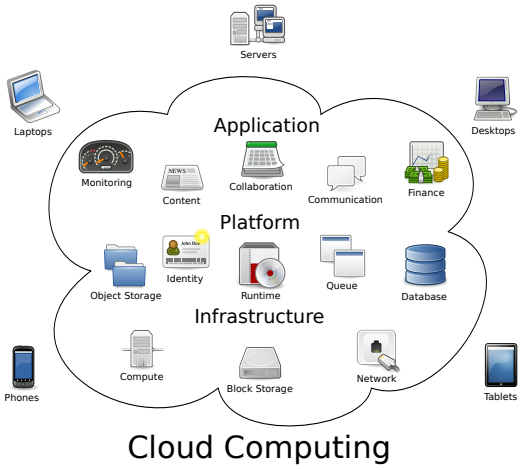
Cloud Computing

Contents

Chapter 1

Cloud computing



Cloud computing metaphor: For a user, the network elements representing the provider-rendered services are invisible, as if obscured by a cloud.

Cloud computing is a recently evolved computing terminology or metaphor based on utility and consumption of computing resources. Cloud computing involves deploying groups of remote servers and software networks that allow centralized data storage and online access to computer services or resources. Clouds can be classified as public, private or hybrid.[1][2]

1.1 Overview

Cloud computing[3] relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network.[2] At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.

Cloud computing, or in simpler shorthand just "the cloud", also focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. This can work for allocating resources to users. For example, a cloud computer facility that serves European users during European business hours with a specific application (e.g., email) may reallocate the same resources to serve North American users during North America's business hours with a different application (e.g., a web server). This approach should maximize the use of computing power thus reducing environmental damage as well since less power, air conditioning, rack space, etc. are required for a variety of functions. With cloud computing, multiple users can access a single server to retrieve and update their data without purchasing licenses for different applications.

The term "moving to cloud" also refers to an organization moving away from a traditional CAPEX model (buy the dedicated hardware and depreciate it over a period of time) to the OPEX model (use a shared cloud infrastructure and pay as one uses it).

Proponents claim that cloud computing allows companies to avoid upfront infrastructure costs, and focus on projects that differentiate their businesses instead of on infrastructure.[4] Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand.[4][5][6] Cloud providers typically use a "pay as you go" model. This can lead to unexpectedly high charges if administrators do not adapt to the cloud pricing model.[7]

The present availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing have led to a growth in cloud computing.[8][9][10]

Cloud vendors are experiencing growth rates of 50% per annum.[11]

1.2 History of cloud computing

1.2.1 Origin of the term

The origin of the term *cloud computing* is unclear. The expression *cloud* is commonly used in science to describe a large agglomeration of objects that visually appear from

a distance as a cloud and describes any set of things whose details are not inspected further in a given context.[12] Another explanation is that the old programs to draw network schematics surrounded the icons for servers with a circle, and a cluster of servers in a network diagram had several overlapping circles, which resembled a cloud.[13]

In analogy to above usage the word *cloud* was used as a metaphor for the Internet and a standardized cloud-like shape was used to denote a network on telephony schematics and later to depict the Internet in computer network diagrams. With this simplification, the implication is that the specifics of how the end points of a network are connected are not relevant for the purposes of understanding the diagram. The cloud symbol was used to represent the Internet as early as 1994,[14][15] in which servers were then shown connected to, but external to, the cloud.

References to cloud computing in its modern sense appeared early as 1996, with the earliest known mention in a Compaq internal document.[16]

The popularization of the term can be traced to 2006 when Amazon.com introduced the Elastic Compute Cloud.[17]

1.2.2 The 1950s

The underlying concept of cloud computing dates to the 1950s, when large-scale mainframe computers were seen as the future of computing, and became available in academia and corporations, accessible via thin clients/terminal computers, often referred to as "dumb terminals", because they were used for communications but had no internal processing capacities. To make more efficient use of costly mainframes, a practice evolved that allowed multiple users to share both the physical access to the computer from multiple terminals as well as the CPU time. This eliminated periods of inactivity on the mainframe and allowed for a greater return on the investment. The practice of sharing CPU time on a mainframe became known in the industry as time-sharing.[18] During the mid 70s, time-sharing was popularly known as RJE (Remote Job Entry); this nomenclature was mostly associated with large vendors such as IBM and DEC. IBM developed the VM Operating System to provide time-sharing services.

1.2.3 The 1990s

In the 1990s, telecommunications companies, who previously offered primarily dedicated point-to-point data circuits, began offering virtual private network (VPN) services with comparable quality of service, but at a lower cost. By switching traffic as they saw fit to balance server use, they could use overall network bandwidth more effectively. They began to use the cloud symbol to denote the demarcation point between what the provider was responsible for and what users were responsible for. Cloud computing extends this boundary to cover all servers as well as the network infrastructure.[19]

As computers became more prevalent, scientists and technologists explored ways to make large-scale computing power available to more users through time-sharing. They experimented with algorithms to optimize the infrastructure, platform, and applications to prioritize CPUs and increase efficiency for end users.[20]

Since 2000 cloud computing has come into existence. In early 2008, OpenNebula, enhanced in the RESERVOIR European Commission-funded project, became the first open-source software for deploying private and hybrid clouds, and for the federation of clouds.[21] In the same year, efforts were focused on providing quality of service guarantees (as required by real-time interactive applications) to cloud-based infrastructures, in the framework of the IRMOS European Commission-funded project, resulting in a real-time cloud environment.[22] By mid-2008, Gartner saw an opportunity for cloud computing “to shape the relationship among consumers of IT services, those who use IT services and those who sell them”[23] and observed that “organizations are switching from company-owned hardware and software assets to per-use service-based models” so that the “projected shift to computing ... will result in dramatic growth in IT products in some areas and significant reductions in other areas.”[24]

Microsoft Azure became available in late 2008.

In July 2010, Rackspace Hosting and NASA jointly launched an open-source cloud-software initiative known as OpenStack. The OpenStack project intended to help organizations offer cloud-computing services running on standard hardware. The early code came from NASA’s Nebula platform as well as from Rackspace’s Cloud Files platform.[25]

On March 1, 2011, IBM announced the IBM SmartCloud framework to support Smarter Planet.[26] Among the various components of the Smarter Computing foundation, cloud computing is a critical piece.

On June 7, 2012, Oracle announced the Oracle Cloud.[27] While aspects of the Oracle Cloud are still in development, this cloud offering is posed to be the first to provide users with access to an integrated set of IT solutions, including the Applications (SaaS), Platform (PaaS), and Infrastructure (IaaS) layers.[28][29][30]

1.3 Similar concepts

Cloud computing is the result of evolution and adoption of existing technologies and paradigms. The goal of cloud computing is to allow users to take benefit from all of these technologies, without the need for deep knowledge

about or expertise with each one of them. The cloud aims to cut costs, and helps the users focus on their core business instead of being impeded by IT obstacles.[31]

The main enabling technology for cloud computing is virtualization. Virtualization software separates a physical computing device into one or more "virtual" devices, each of which can be easily used and managed to perform computing tasks. With operating system–level virtualization essentially creating a scalable system of multiple independent computing devices, idle computing resources can be allocated and used more efficiently. Virtualization provides the agility required to speed up IT operations, and reduces cost by increasing infrastructure utilization. Autonomic computing automates the process through which the user can provision resources on-demand. By minimizing user involvement, automation speeds up the process, reduces labor costs and reduces the possibility of human errors.[31]

Users routinely face difficult business problems. Cloud computing adopts concepts from Service-oriented Architecture (SOA) that can help the user break these problems into services that can be integrated to provide a solution. Cloud computing provides all of its resources as services, and makes use of the well-established standards and best practices gained in the domain of SOA to allow global and easy access to cloud services in a standardized way.

Cloud computing also leverages concepts from utility computing to provide metrics for the services used. Such metrics are at the core of the public cloud pay-per-use models. In addition, measured services are an essential part of the feedback loop in autonomic computing, allowing services to scale on-demand and to perform automatic failure recovery.

Cloud computing is a kind of grid computing; it has evolved by addressing the QoS (quality of service) and reliability problems. Cloud computing provides the tools and technologies to build data/compute intensive parallel applications with much more affordable prices compared to traditional parallel computing techniques.<ref name=HAM2012/

Cloud computing shares characteristics with:

- Client–server model — *Client–server computing* refers broadly to any distributed application that distinguishes between service providers (servers) and service requestors (clients).[32]
- Grid computing — "A form of distributed and parallel computing, whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large tasks."
- Mainframe computer — Powerful computers used mainly by large organizations for critical applications, typically bulk data processing such as: census; industry and consumer statistics; police and secret intelligence services; enterprise resource planning; and financial transaction processing.
- Utility computing — The "packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility, such as electricity."[33][34]
- Peer-to-peer — A distributed architecture without the need for central coordination. Participants are both suppliers and consumers of resources (in contrast to the traditional client–server model).

1.4 Characteristics

Cloud computing exhibits the following key characteristics:

- **Agility** improves with users' ability to re-provision technological infrastructure resources.
- **Application programming interface** (API) accessibility to software that enables machines to interact with cloud software in the same way that a traditional user interface (e.g., a computer desktop) facilitates interaction between humans and computers. Cloud computing systems typically use Representational State Transfer (REST)-based APIs.
- **Cost** reductions claimed by cloud providers. A public-cloud delivery model converts capital expenditure to operational expenditure.[35] This purportedly lowers barriers to entry, as infrastructure is typically provided by a third party and does not need to be purchased for one-time or infrequent intensive computing tasks. Pricing on a utility computing basis is fine-grained, with usage-based options and fewer IT skills are required for implementation (in-house).[36] The e-FISCAL project's state-of-the-art repository[37] contains several articles looking into cost aspects in more detail, most of them concluding that costs savings depend on the type of activities supported and the type of infrastructure available in-house.
- **Device and location independence**[38] enable users to access systems using a web browser regardless of their location or what device they use (e.g., PC, mobile phone). As infrastructure is off-site (typically provided by a third-party) and accessed via the Internet, users can connect from anywhere.[36]
- **Maintenance** of cloud computing applications is easier, because they do not need to be installed on each user's computer and can be accessed from different places.
- **Multitenancy** enables sharing of resources and costs across a large pool of users thus allowing for:

  - **centralization** of infrastructure in locations with lower costs (such as real estate, electricity, etc.)
  - **peak-load capacity** increases (users need not engineer for highest possible load-levels)
  - **utilisation and efficiency** improvements for systems that are often only 10–20% utilised.[39][40]

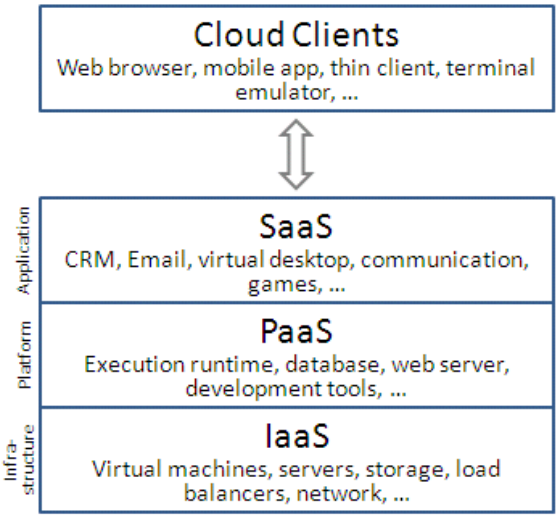
- **Performance** is monitored, and consistent and loosely coupled architectures are constructed using web services as the system interface.[36][41][42]
- **Productivity** may be increased when multiple users can work on the same data simultaneously, rather than waiting for it to be saved and emailed. Time may be saved as information does not need to be re-entered when fields are matched, nor do users need to install application software upgrades to their computer.[43]
- **Reliability** improves with the use of multiple redundant sites, which makes well-designed cloud computing suitable for business continuity and disaster recovery.[44]
- **Scalability and elasticity** via dynamic ("on-demand") provisioning of resources on a fine-grained, self-service basis in near real-time[45][46] (Note, the VM startup time varies by VM type, location, OS and cloud providers[45]), without users having to engineer for peak loads.[47][48][49]
- **Security** can improve due to centralization of data, increased security-focused resources, etc., but concerns can persist about loss of control over certain sensitive data, and the lack of security for stored kernels. Security is often as good as or better than other traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford to tackle.[50] However, the complexity of security is greatly increased when data is distributed over a wider area or over a greater number of devices, as well as in multi-tenant systems shared by unrelated users. In addition, user access to security audit logs may be difficult or impossible. Private cloud installations are in part motivated by users' desire to retain control over the infrastructure and avoid losing control of information security.

The National Institute of Standards and Technology's definition of cloud computing identifies "five essential characteristics":

> *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
>
> *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
>
> *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
>
> *Rapid elasticity.* Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear unlimited and can be appropriated in any quantity at any time.
>
> *Measured service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.
>
> —National Institute of Standards and Technology[2]

1.5 Service models

Cloud computing providers offer their services according to several fundamental models:[2][51]



1.5.1 Infrastructure as a service (IaaS)

See also: Category:Cloud infrastructure

In the most basic cloud-service model & according to the IETF (Internet Engineering Task Force), providers of IaaS offer computers – physical or (more often) virtual machines – and other resources. (A hypervisor, such as Xen, Oracle VirtualBox, KVM, VMware ESX/ESXi, or Hyper-V runs the virtual machines as guests. Pools of hypervisors within the cloud operational support-system can support large numbers of virtual machines and the ability to scale services up and down according to customers' varying requirements.) IaaS clouds often offer additional resources such as a virtual-machine disk image library, raw block storage, and file or object storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles.[52] IaaS-cloud providers supply these resources on-demand from their large pools installed in data centers. For wide-area connectivity, customers can use either the Internet or carrier clouds (dedicated virtual private networks).

To deploy their applications, cloud users install operating-system images and their application software on the cloud infrastructure. In this model, the cloud user patches and maintains the operating systems and the application software. Cloud providers typically bill IaaS services on a utility computing basis: cost reflects the amount of resources allocated and consumed.[53][54][55]

1.5.2 Platform as a service (PaaS)

Main article: Platform as a service
See also: Category:Cloud platforms

In the PaaS models, cloud providers deliver a computing platform, typically including operating system, programming language execution environment, database, and web server. Application developers can develop and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. With some PaaS offers like Microsoft Azure and Google App Engine, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually. The latter has also been proposed by an architecture aiming to facilitate real-time in cloud environments.[56] Even more specific application types can be provided via PaaS, e.g., such as media encoding as provided by services as bitcodin transcoding cloud[57] or media.io.[58]

1.5.3 Software as a service (SaaS)

Main article: Software as a service

In the business model using software as a service (SaaS), users are provided access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis or using a subscription fee.

In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs. This eliminates the need to install and run the application on the cloud user's own computers, which simplifies maintenance and support. Cloud applications are different from other applications in their scalability—which can be achieved by cloning tasks onto multiple virtual machines at run-time to meet changing work demand.[59] Load balancers distribute the work over the set of virtual machines. This process is transparent to the cloud user, who sees only a single access point. To accommodate a large number of cloud users, cloud applications can be *multitenant*, that is, any machine serves more than one cloud user organization.

The pricing model for SaaS applications is typically a monthly or yearly flat fee per user,[60] so price is scalable and adjustable if users are added or removed at any point.[61]

Proponents claim SaaS allows a business the potential to reduce IT operational costs by outsourcing hardware and software maintenance and support to the cloud provider. This enables the business to reallocate IT operations costs away from hardware/software spending and personnel expenses, towards meeting other goals. In addition, with applications hosted centrally, updates can be released without the need for users to install new software. One drawback of SaaS is that the users' data are stored on the cloud provider's server. As a result, there could be unauthorized access to the data. For this reason, users are increasingly adopting intelligent third-party key management systems to help secure their data.

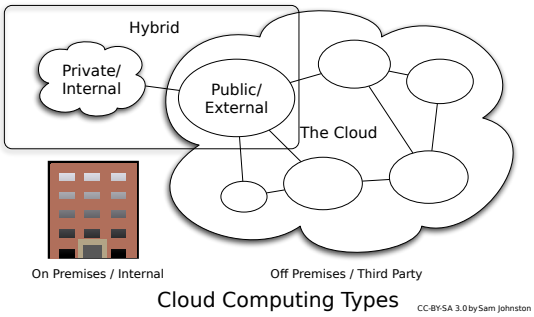
1.6 Cloud clients

See also: Category:Cloud clients

Users access cloud computing using networked client devices, such as desktop computers, laptops, tablets and smartphones. Some of these devices – *cloud clients* – rely on cloud computing for all or a majority of their applications so as to be essentially useless without it. Examples are thin clients and the browser-based Chromebook. Many cloud applications do not require specific software

on the client and instead use a web browser to interact with the cloud application. With Ajax and HTML5 these Web user interfaces can achieve a similar, or even better, look and feel to native applications. Some cloud applications, however, support specific client software dedicated to these applications (e.g., virtual desktop clients and most email clients). Some legacy applications (line of business applications that until now have been prevalent in thin client computing) are delivered via a screen-sharing technology.

1.7 Deployment models



Cloud computing types

1.7.1 Private cloud

Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party, and hosted either internally or externally.[2] Undertaking a private cloud project requires a significant level and degree of engagement to virtualize the business environment, and requires the organization to reevaluate decisions about existing resources. When done right, it can improve business, but every step in the project raises security issues that must be addressed to prevent serious vulnerabilities.[62] Self-run data centers[63] are generally capital intensive. They have a significant physical footprint, requiring allocations of space, hardware, and environmental controls. These assets have to be refreshed periodically, resulting in additional capital expenditures. They have attracted criticism because users "still have to buy, build, and manage them" and thus do not benefit from less hands-on management,[64] essentially "[lacking] the economic model that makes cloud computing such an intriguing concept".[65][66]

1.7.2 Public cloud

A cloud is called a "public cloud" when the services are rendered over a network that is open for public use. Public cloud services may be free.[67] Technically there may be little or no difference between public and private cloud architecture, however, security consideration may be substantially different for services (applications, storage, and other resources) that are made available by a service provider for a public audience and when communication is effected over a non-trusted network. Saasu is a large public cloud. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure at their data center and access is generally via the Internet. AWS and Microsoft also offer direct connect services called "AWS Direct Connect" and "Azure ExpressRoute" respectively, such connections require customers to purchase or lease a private connection to a peering point offered by the cloud provider.[36]

1.7.3 Hybrid cloud

Hybrid cloud is a composition of two or more clouds (private, community or public) that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect collocation, managed and/or dedicated services with cloud resources.[2]

Gartner, Inc. defines a hybrid cloud service as a cloud computing service that is composed of some combination of private, public and community cloud services, from different service providers.[68] A hybrid cloud service crosses isolation and provider boundaries so that it can't be simply put in one category of private, public, or community cloud service. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service.

Varied use cases for hybrid cloud composition exist. For example, an organization may store sensitive client data in house on a private cloud application, but interconnect that application to a business intelligence application provided on a public cloud as a software service.[69] This example of hybrid cloud extends the capabilities of the enterprise to deliver a specific business service through the addition of externally available public cloud services. Hybrid cloud adoption depends on a number of factors such as data security and compliance requirements, level of control needed over data, and the applications an organization uses.[70]

Another example of hybrid cloud is one where IT organizations use public cloud computing resources to meet temporary capacity needs that can not be met by the private cloud.[71] This capability enables hybrid clouds to employ cloud bursting for scaling across clouds.[2] Cloud bursting is an application deployment model in which an application runs in a private cloud or data center and "bursts" to a public cloud when the demand for computing capacity increases. A primary advantage of cloud bursting and a hybrid cloud model is that an organization only pays for extra compute resources when they are needed.[72] Cloud bursting enables data centers to cre-

ate an in-house IT infrastructure that supports average workloads, and use cloud resources from public or private clouds, during spikes in processing demands.[73]

1.7.4 Others

Community cloud

Community cloud shares infrastructure between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third-party, and either hosted internally or externally. The costs are spread over fewer users than a public cloud (but more than a private cloud), so only some of the cost savings potential of cloud computing are realized.[2]

Distributed cloud

Cloud computing can also be provided by a distributed set of machines that are running at different locations, while still connected to a single network or hub service. Examples of this include distributed computing platforms such as BOINC and Folding@Home. An interesting attempt in such direction is Cloud@Home, aiming at implementing cloud computing provisioning model on top of voluntarily shared resources [74]

Intercloud

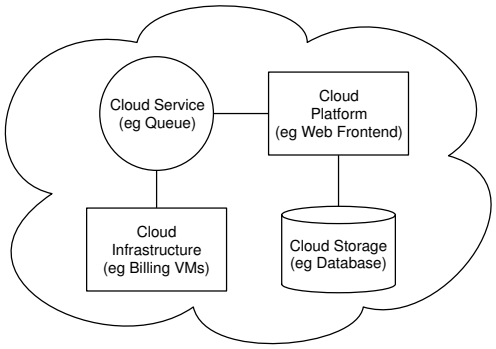
Main article: Intercloud

The Intercloud[75] is an interconnected global "cloud of clouds"[76][77] and an extension of the Internet "network of networks" on which it is based. The focus is on direct interoperability between public cloud service providers, more so than between providers and consumers (as is the case for hybrid- and multi-cloud).[78][79][80]

Multicloud

Main article: Multicloud

Multicloud is the use of multiple cloud computing services in a single heterogeneous architecture to reduce reliance on single vendors, increase flexibility through choice, mitigate against disasters, etc. It differs from hybrid cloud in that it refers to multiple cloud services, rather than multiple deployment modes (public, private, legacy).[81][82]



Cloud computing sample architecture

1.8 Architecture

Cloud architecture,[83] the systems architecture of the software systems involved in the delivery of cloud computing, typically involves multiple *cloud components* communicating with each other over a loose coupling mechanism such as a messaging queue. Elastic provision implies intelligence in the use of tight or loose coupling as applied to mechanisms such as these and others.

1.8.1 Cloud engineering

Cloud engineering is the application of engineering disciplines to cloud computing. It brings a systematic approach to the high-level concerns of commercialization, standardization, and governance in conceiving, developing, operating and maintaining cloud computing systems. It is a multidisciplinary method encompassing contributions from diverse areas such as systems, software, web, performance, information, security, platform, risk, and quality engineering.

1.9 Security and privacy

Main article: Cloud computing issues

Cloud computing poses privacy concerns because the service provider can access the data that is on the cloud at any time. It could accidentally or deliberately alter or even delete information.[84] Many cloud providers can share information with third parties if necessary for purposes of law and order even without a warrant. That is permitted in their privacy policies which users have to agree to before they start using cloud services.[85] Solutions to privacy include policy and legislation as well as end users' choices for how data is stored.[84] Users can encrypt data that is processed or stored within the cloud to prevent unauthorized access.[84]

According to the Cloud Security Alliance, the top three

threats in the cloud are “Insecure Interfaces and API’s”, Data Loss & Leakage”, and “Hardware Failure” which accounted for 29%, 25% and 10% of all cloud security outages respectively - together these form shared technology vulnerabilities. In a cloud provider platform being shared by different users there may be a possibility that information belonging to different customers resides on same data server. Therefore Information leakage may arise by mistake when information for one customer is given to other.[86] Additionally, Eugene Schultz, chief technology officer at Emagined Security, said that hackers are spending substantial time and effort looking for ways to penetrate the cloud. “There are some real Achilles’ heels in the cloud infrastructure that are making big holes for the bad guys to get into”. Because data from hundreds or thousands of companies can be stored on large cloud servers, hackers can theoretically gain control of huge stores of information through a single attack — a process he called “hyperjacking”.

There is the problem of legal ownership of the data (If a user stores some data in the cloud, can the cloud provider profit from it?). Many Terms of Service agreements are silent on the question of ownership.[87]

Physical control of the computer equipment (private cloud) is more secure than having the equipment off site and under someone else’s control (public cloud). This delivers great incentive to public cloud computing service providers to prioritize building and maintaining strong management of secure services.[88] Some small businesses that don't have expertise in IT security could find that it’s more secure for them to use a public cloud.

There is the risk that end users don't understand the issues involved when signing on to a cloud service (persons sometimes don't read the many pages of the terms of service agreement, and just click “Accept” without reading). This is important now that cloud computing is becoming popular and required for some services to work, for example for an intelligent personal assistant (Apple’s Siri or Google Now).

Fundamentally private cloud is seen as more secure with higher levels of control for the owner, however public cloud is seen to be more flexible and requires less time and money investment from the user.[89]

1.10 The future

According to Gartner’s Hype cycle, cloud computing has reached a maturity that leads it into a productive phase. This means that most of the main issues with cloud computing have been addressed to a degree that clouds have become interesting for full commercial exploitation. This however does not mean that all the problems listed above have actually been solved, only that the according risks can be tolerated to a certain degree.[90] Cloud computing is therefore still as much a research topic, as it is a market offering.[91] What is clear through the evolution of Cloud Computing services is that the CTO is a major driving force behind Cloud adoption.[92] The major Cloud technology developers continue to invest billions a year in Cloud R&D, in 2011 Microsoft for example committed 90% of its $9.6bn R&D budget to Cloud[93]

1.11 See also

- Category:Cloud computing providers
- Category:Cloud platforms
- Cloud computing comparison
- Cloud management
- Cloud research
- Cloud storage
- Edge computing
- Fog computing
- Grid computing
- eScience
- iCloud
- Mobile cloud computing
- Personal cloud
- Robot as a Service
- Service-Oriented Architecture
- Synaptop
- Ubiquitous computing
- Web computing

1.12 References

[1] Hassan, Qusay (2011). “Demystifying Cloud Computing”. *The Journal of Defense Software Engineering* (CrossTalk) **2011** (Jan/Feb): 16–21. Retrieved 11 December 2014.

[2] “The NIST Definition of Cloud Computing”. National Institute of Standards and Technology. Retrieved 24 July 2011.

[3] “Know Why Cloud Computing Technology is the New Revolution”. By Fonebell. Retrieved 8 January 2015.

[4] “What is Cloud Computing?". *Amazon Web Services*. 2013-03-19. Retrieved 2013-03-20.

[5] "Baburajan, Rajani, "The Rising Cloud Storage Market Opportunity Strengthens Vendors," infoTECH, August 24, 2011". It.tmcnet.com. 2011-08-24. Retrieved 2011-12-02.

[6] Oestreich, Ken, (2010-11-15). "Converged Infrastructure". *CTO Forum*. Thectoforum.com. Retrieved 2011-12-02.

[7] "Where's The Rub: Cloud Computing's Hidden Costs". 2014-02-27. Retrieved 2014-07-14.

[8] "Cloud Computing: Clash of the clouds". *The Economist*. 2009-10-15. Retrieved 2009-11-03.

[9] "Gartner Says Cloud Computing Will Be As Influential As E-business". Gartner. Retrieved 2010-08-22.

[10] Gruman, Galen (2008-04-07). "What cloud computing really means". *InfoWorld*. Retrieved 2009-06-02.

[11] "The economy is flat so why are financials Cloud vendors growing at more than 90 percent per annum?". FSN. March 5, 2013.

[12] Liu, [edited by] Hongji Yang, Xiaodong (2012). "9". *Software reuse in the emerging cloud computing era*. Hershey, PA: Information Science Reference. pp. 204–227. ISBN 9781466608979. Retrieved 11 December 2014.

[13] Schmidt, Eric; Rosenberg, Jonathan (2014). *How Google Works*. Grand Central Publishing. p. 11. ISBN 978-1-4555-6059-2.

[14] Figure 8, "A network 70 is shown schematically as a cloud", US Patent 5,485,455, column 17, line 22, filed Jan 28, 1994

[15] Figure 1, "the cloud indicated at 49 in Fig. 1.", US Patent 5,790,548, column 5 line 56–57, filed April 18, 1996

[16] Antonio Regalado (31 October 2011). "Who Coined 'Cloud Computing'?". *Technology Review* (MIT). Retrieved 31 July 2013.

[17] "Announcing Amazon Elastic Compute Cloud (Amazon EC2) - beta". Amazon.com. 2006-08-24. Retrieved 2014-05-31.

[18] Strachey, Christopher (June 1959). "Time Sharing in Large Fast Computers". *Proceedings of the International Conference on Information processing, UNESCO*. paper B.2.19: 336–341.

[19] "July, 1993 meeting report from the IP over ATM working group of the IETF". CH: Switch. Retrieved 2010-08-22.

[20] Corbató, Fernando J. "An Experimental Time-Sharing System". *SJCC Proceedings*. MIT. Retrieved 3 July 2012.

[21] Rochwerger, B.; Breitgand, D.; Levy, E.; Galis, A.; Nagin, K.; Llorente, I. M.; Montero, R.; Wolfsthal, Y.; Elmroth, E.; Caceres, J.; Ben-Yehuda, M.; Emmerich, W.; Galan, F. "The Reservoir model and architecture for open federated cloud computing". *IBM Journal of Research and Development* **53** (4): 4:1–4:11. doi:10.1147/JRD.2009.5429058.

[22] Kyriazis, D; A Menychtas; G Kousiouris; K Oberle; T Voith; M Boniface; E Oliveros; T Cucinotta; S Berger (November 2010). "A Real-time Service Oriented Infrastructure". *International Conference on Real-Time and Embedded Systems (RTES 2010)* (Singapore).

[23] Keep an eye on cloud computing, Amy Schurr, Network World, 2008-07-08, citing the Gartner report, "Cloud Computing Confusion Leads to Opportunity". Retrieved 2009-09-11.

[24] Gartner (2008-08-18). "Gartner Says Worldwide IT Spending On Pace to Surpass Trillion in 2008".

[25] "OpenStack History".

[26] "Launch of IBM Smarter Computing". Retrieved 1 March 2011.

[27] "Launch of Oracle Cloud". Retrieved 28 February 2014.

[28] "Oracle Cloud, Enterprise-Grade Cloud Solutions: SaaS, PaaS, and IaaS". Retrieved 12 October 2014.

[29] "Larry Ellison Doesn't Get the Cloud: The Dumbest Idea of 2013". Forbes.com. Retrieved 12 October 2014.

[30] "Oracle Disrupts Cloud Industry with End-to-End Approach". Forbes.com. Retrieved 12 October 2014.

[31] HAMDAQA, Mohammad (2012). *Cloud Computing Uncovered: A Research Landscape*. Elsevier Press. pp. 41–85. ISBN 0-12-396535-7.

[32] "Distributed Application Architecture". Sun Microsystem. Retrieved 2009-06-16.

[33] "It's probable that you've misunderstood 'Cloud Computing' until now". TechPluto. Retrieved 2010-09-14.

[34] Danielson, Krissi (2008-03-26). "Distinguishing Cloud Computing from Utility Computing". Ebizq.net. Retrieved 2010-08-22.

[35] "Recession Is Good For Cloud Computing – Microsoft Agrees". CloudAve. Retrieved 2010-08-22.

[36] "Defining 'Cloud Services' and "Cloud Computing"". IDC. 2008-09-23. Retrieved 2010-08-22.

[37] "e-FISCAL project state of the art repository".

[38] Farber, Dan (2008-06-25). "The new geek chic: Data centers". CNET News. Retrieved 2010-08-22.

[39] "Jeff Bezos' Risky Bet". *Business Week*.

[40] He, Sijin; L. Guo, Y. Guo, M. Ghanem,. "Improving Resource Utilisation in the Cloud Environment Using Multivariate Probabilistic Models". 2012 2012 IEEE 5th International Conference on Cloud Computing (CLOUD). pp. 574–581. doi:10.1109/CLOUD.2012.66. ISBN 978-1-4673-2892-0.

[41] He, Qiang, et al. "Formulating Cost-Effective Monitoring Strategies for Service-based Systems." (2013): 1-1.

[42] A Self-adaptive hierarchical monitoring mechanism for Clouds Elsevier.com

[43] Heather Smith (23 May 2013). *Xero For Dummies*. John Wiley & Sons. pp. 37–. ISBN 978-1-118-57252-8.

[44] King, Rachael (2008-08-04). "Cloud Computing: Small Companies Take Flight". *Bloomberg BusinessWeek*. Retrieved 2010-08-22.

[45] Mao, Ming; M. Humphrey (2012). "A Performance Study on the VM Startup Time in the Cloud". *Proceedings of 2012 IEEE 5th International Conference on Cloud Computing (Cloud2012)*: 423. doi:10.1109/CLOUD.2012.103. ISBN 978-1-4673-2892-0.

[46] Dario Bruneo, Salvatore Distefano, Francesco Longo, Antonio Puliafito, Marco Scarpa: Workload-Based Software Rejuvenation in Cloud Systems. IEEE Trans. Computers 62(6): 1072-1085 (2013)

[47] "Defining and Measuring Cloud Elasticity". KIT Software Quality Departement. Retrieved 13 August 2011.

[48] "Economies of Cloud Scale Infrastructure". Cloud Slam 2011. Retrieved 13 May 2011.

[49] He, Sijin; L. Guo; Y. Guo; C. Wu; M. Ghanem; R. Han. "Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning". 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA). pp. 15–22. doi:10.1109/AINA.2012.74. ISBN 978-1-4673-0714-7.

[50] Mills, Elinor (2009-01-27). "Cloud computing security forecast: Clear skies". CNET News. Retrieved 2010-08-22.

[51] Voorsluys, William; Broberg, James; Buyya, Rajkumar (February 2011). "Introduction to Cloud Computing". In R. Buyya, J. Broberg, A.Goscinski. *Cloud Computing: Principles and Paradigms*. New York, USA: Wiley Press. pp. 1–44. ISBN 978-0-470-88799-8.

[52] Amies, Alex; Sluiman, Harm; Tong, Qiang Guo; Liu, Guo Ning (July 2012). "*Infrastructure as a Service Cloud Concepts*". ***Developing and Hosting Applications on the Cloud***. IBM Press. ISBN 978-0-13-306684-5.

[53] "Amazon EC2 Pricing". Retrieved 7 July 2014.

[54] "Compute Engine Pricing". Retrieved 7 July 2014.

[55] "Microsoft Azure Virtual Machines Pricing Details". Retrieved 7 July 2014.

[56] Boniface, M. et al. (2010), *Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds*, 5th International Conference on Internet and Web Applications and Services (ICIW), Barcelona, Spain: IEEE, pp. 155–160, doi:10.1109/ICIW.2010.91

[57] bitcodin cloud transcoding platform

[58] media.io

[59] Hamdaqa, Mohammad. *A Reference Model for Developing Cloud Applications*.

[60] Chou, Timothy. *Introduction to Cloud Computing: Business & Technology*.

[61] "HVD: the cloud's silver lining". Intrinsic Technology. Retrieved 30 August 2012.

[62] "Is The Private Cloud Really More Secure?". CloudAndCompute.com. Retrieved 12 October 2014.

[63] "Self-Run Private Cloud Computing Solution - GovConnection". *govconnection.com*. 2014. Retrieved April 15, 2014.

[64] Foley, John. "Private Clouds Take Shape". *InformationWeek*. Retrieved 2010-08-22.

[65] Haff, Gordon (2009-01-27). "Just don't call them private clouds". CNET News. Retrieved 2010-08-22.

[66] "There's No Such Thing As A Private Cloud". *InformationWeek*. 2010-06-30. Retrieved 2010-08-22.

[67] Rouse, Margaret. "What is public cloud?". Definition from Whatis.com. Retrieved 12 October 2014.

[68] http://blogs.gartner.com/thomas_bittman/2012/09/24/mind-the-gap-here-comes-hybrid-cloud/

[69] "Business Intelligence Takes to Cloud for Small Businesses". CIO.com. 2014-06-04. Retrieved 2014-06-04.

[70] http://www.techradar.com/news/internet/cloud-services/hybrid-cloud-is-it-right-for-your-business-$-$1261343

[71] Metzler, Jim; Taylor, Steve. (2010-08-23) "Cloud computing: Reality vs. fiction," Network World.

[72] Rouse, Margaret. "Definition: Cloudbursting," May 2011. SearchCloudComputing.com.

[73] Vizard, Michael. "How Cloudbursting 'Rightsizes' the Data Center", (2012-06-21). Slashdot.

[74] Vincenzo D. Cunsolo, Salvatore Distefano, Antonio Puliafito, Marco Scarpa: Volunteer Computing and Desktop Cloud: The Cloud@Home Paradigm. IEEE International Symposium on Network Computing and Applications, NCA 2009, pp 134-139

[75] Bernstein, David; Ludvigson, Erik; Sankar, Krishna; Diamond, Steve; Morrow, Monique (2009-05-24). "Blueprint for the Intercloud – Protocols and Formats for Cloud Computing Interoperability". IEEE Computer Society. pp. 328–336. doi:10.1109/ICIW.2009.55. ISBN 978-1-4244-3851-8.

[76] "Kevin Kelly: A Cloudbook for the Cloud". Kk.org. Retrieved 2010-08-22.

[77] "Intercloud is a global cloud of clouds". Samj.net. 2009-06-22. Retrieved 2010-08-22.

[78] "Vint Cerf: Despite Its Age, The Internet is Still Filled with Problems". Readwriteweb.com. Retrieved 2010-08-22.

[79] "SP360: Service Provider: From India to Intercloud". Blogs.cisco.com. Retrieved 2010-08-22.

[80] Canada (2007-11-29). "Head iaaan the clouds? Welcome to the future". *The Globe and Mail* (Toronto). Retrieved 2010-08-22.

[81] Rouse, Margaret. "What is a multi-cloud strategy". SearchCloudApplications. Retrieved 3 July 2014.

[82] King, Rachel. "Pivotal's head of products: We're moving to a multi-cloud world". ZDnet. Retrieved 3 July 2014.

[83] "Building GrepTheWeb in the Cloud, Part 1: Cloud Architectures". Developer.amazonwebservices.com. Retrieved 2010-08-22.

[84] "Cloud Computing Privacy Concerns on Our Doorstep".

[85] "Sharing information without a warrant". Retrieved 2014-12-05.

[86] Chhibber, A (2013). "SECURITY ANALYSIS OF CLOUD COMPUTING". *International Journal of Advanced Research in Engineering and Applied Sciences* **2** (3): 2278-6252. Retrieved 27 February 2015.

[87] Maltais, Michelle (26 April 2012). "Who owns your stuff in the cloud?". *Los Angeles Times*. Retrieved 2012-12-14.

[88] "Security of virtualization, cloud computing divides IT and security pros". Network World. 2010-02-22. Retrieved 2010-08-22.

[89] "The Bumpy Road to Private Clouds". Retrieved 2014-10-08.

[90] http://blog.kaseya.com/blog/2014/10/06/realistic-look-cloud-computing/

[91] Smith, David Mitchell. "Hype Cycle for Cloud Computing, 2013". Gartner. Retrieved 3 July 2014.

[92] http://www.hello-cirro.co.uk/evolution-of-cloud-computing/

[93] http://cloudtimes.org/2011/04/12/microsoft-says-to-spend-90-of-rd-on-cloud-strategy/

1.13 External links

Chapter 2

Grid computing

Grid computing is the collection of computer resources from multiple locations to reach a common goal. The **grid** can be thought of as a distributed system with non-interactive workloads that involve a large number of files. Grid computing is distinguished from conventional high performance computing systems such as cluster computing in that grid computers have each node set to perform a different task/application.[1] Grid computers also tend to be more heterogeneous and geographically dispersed (thus not physically coupled) than cluster computers.[2] Although a single grid can be dedicated to a particular application, commonly a grid is used for a variety of purposes. Grids are often constructed with general-purpose grid middleware software libraries.

Grid size varies a considerable amount. Grids are a form of distributed computing whereby a **"super virtual computer"** is composed of many networked loosely coupled computers acting together to perform large tasks. For certain applications, "distributed" or "grid" computing, can be seen as a special type of parallel computing that relies on complete computers (with onboard CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private or public) by a conventional network interface, such as Ethernet. This is in contrast to the traditional notion of a supercomputer, which has many processors connected by a local high-speed computer bus.

2.1 Overview

Grid computing combines computers from multiple administrative domains to reach **a common goal**,[3] **to solve a single task**, and may then disappear just as quickly.

One of the main strategies of grid computing is to use middleware to divide and apportion pieces of a program among several computers, sometimes up to many thousands. Grid computing involves computation in a distributed fashion, which may also involve the aggregation of large-scale clusters.

The size of a grid may vary from small—confined to a network of computer workstations within a corporation, for example—to large, public collaborations across many companies and networks. "The notion of a confined grid may also be known as an intra-nodes cooperation whilst the notion of a larger, wider grid may thus refer to an inter-nodes cooperation".[4]

Grids are a form of distributed computing whereby a "super virtual computer" is composed of many networked loosely coupled computers acting together to perform very large tasks. This technology has been applied to computationally intensive scientific, mathematical, and academic problems through volunteer computing, and it is used in commercial enterprises for such diverse applications as drug discovery, economic forecasting, seismic analysis, and back office data processing in support for e-commerce and Web services.

Coordinating applications on Grids can be a complex task, especially when coordinating the flow of information across distributed computing resources. Grid workflow systems have been developed as a specialized form of a workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or a workflow, in the Grid context.

2.2 Comparison of grids and conventional supercomputers

"Distributed" or "grid" computing in general is a special type of parallel computing that relies on complete computers (with onboard CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private, public or the Internet) by a conventional network interface producing commodity hardware, compared to the lower efficiency of designing and constructing a small number of custom supercomputers. The primary performance disadvantage is that the various processors and local storage areas do not have high-speed connections. This arrangement is thus well-suited to applications in which multiple parallel computations can take place independently, without the need to communicate intermediate results between processors.[5] The high-end scalability of geographically dispersed grids is generally favorable, due to the low need for connectivity between nodes relative to the capacity of the public Internet.

There are also some differences in programming and de-

ployment. It can be costly and difficult to write programs that can run in the environment of a supercomputer, which may have a custom operating system, or require the program to address concurrency issues. If a problem can be adequately parallelized, a "thin" layer of "grid" infrastructure can allow conventional, standalone programs, given a different part of the same problem, to run on multiple machines. This makes it possible to write and debug on a single conventional machine, and eliminates complications due to multiple instances of the same program running in the same shared memory and storage space at the same time.

2.3 Design considerations and variations

One feature of distributed grids is that they can be formed from computing resources belonging to multiple individuals or organizations (known as multiple administrative domains). This can facilitate commercial transactions, as in utility computing, or make it easier to assemble volunteer computing networks.

One disadvantage of this feature is that the computers which are actually performing the calculations might not be entirely trustworthy. The designers of the system must thus introduce measures to prevent malfunctions or malicious participants from producing false, misleading, or erroneous results, and from using the system as an attack vector. This often involves assigning work randomly to different nodes (presumably with different owners) and checking that at least two different nodes report the same answer for a given work unit. Discrepancies would identify malfunctioning and malicious nodes. However, due to the lack of central control over the hardware, there is no way to guarantee that nodes will not drop out of the network at random times. Some nodes (like laptops or dialup Internet customers) may also be available for computation but not network communications for unpredictable periods. These variations can be accommodated by assigning large work units (thus reducing the need for continuous network connectivity) and reassigning work units when a given node fails to report its results in expected time.

The impacts of trust and availability on performance and development difficulty can influence the choice of whether to deploy onto a dedicated cluster, to idle machines internal to the developing organization, or to an open external network of volunteers or contractors. In many cases, the participating nodes must trust the central system not to abuse the access that is being granted, by interfering with the operation of other programs, mangling stored information, transmitting private data, or creating new security holes. Other systems employ measures to reduce the amount of trust "client" nodes must place in the central system such as placing applications in virtual machines.

Public systems or those crossing administrative domains (including different departments in the same organization) often result in the need to run on heterogeneous systems, using different operating systems and hardware architectures. With many languages, there is a trade off between investment in software development and the number of platforms that can be supported (and thus the size of the resulting network). Cross-platform languages can reduce the need to make this trade off, though potentially at the expense of high performance on any given node (due to run-time interpretation or lack of optimization for the particular platform). There are diverse scientific and commercial projects to harness a particular associated grid or for the purpose of setting up new grids. BOINC is a common one for various academic projects seeking public volunteers; more are listed at the end of the article.

In fact, the middleware can be seen as a layer between the hardware and the software. On top of the middleware, a number of technical areas have to be considered, and these may or may not be middleware independent. Example areas include SLA management, Trust and Security, Virtual organization management, License Management, Portals and Data Management. These technical areas may be taken care of in a commercial solution, though the cutting edge of each area is often found within specific research projects examining the field.

2.4 Market segmentation of the grid computing market

For the segmentation of the grid computing market, two perspectives need to be considered: the provider side and the user side:

2.4.1 The provider side

The overall grid market comprises several specific markets. These are the grid middleware market, the market for grid-enabled applications, the utility computing market, and the software-as-a-service (SaaS) market.

Grid middleware is a specific software product, which enables the sharing of heterogeneous resources, and Virtual Organizations. It is installed and integrated into the existing infrastructure of the involved company or companies, and provides a special layer placed among the heterogeneous infrastructure and the specific user applications. Major grid middlewares are Globus Toolkit, gLite, and UNICORE.

Utility computing is referred to as the provision of grid computing and applications as service either as an open grid utility or as a hosting solution for one organization or a VO. Major players in the utility computing market are Sun Microsystems, IBM, and HP.

Grid-enabled applications are specific software applications that can utilize grid infrastructure. This is made possible by the use of grid middleware, as pointed out above.

Software as a service (SaaS) is "software that is owned, delivered and managed remotely by one or more providers." (Gartner 2007) Additionally, SaaS applications are based on a single set of common code and data definitions. They are consumed in a one-to-many model, and SaaS uses a Pay As You Go (PAYG) model or a subscription model that is based on usage. Providers of SaaS do not necessarily own the computing resources themselves, which are required to run their SaaS. Therefore, SaaS providers may draw upon the utility computing market. The utility computing market provides computing resources for SaaS providers.

2.4.2 The user side

For companies on the demand or user side of the grid computing market, the different segments have significant implications for their IT deployment strategy. The IT deployment strategy as well as the type of IT investments made are relevant aspects for potential grid users and play an important role for grid adoption.

2.5 CPU scavenging

CPU-scavenging, **cycle-scavenging**, or **shared computing** creates a "grid" from the unused resources in a network of participants (whether worldwide or internal to an organization). Typically this technique uses desktop computer instruction cycles that would otherwise be wasted at night, during lunch, or even in the scattered seconds throughout the day when the computer is waiting for user input or slow devices. In practice, participating computers also donate some supporting amount of disk storage space, RAM, and network bandwidth, in addition to raw CPU power.

Many volunteer computing projects, such as BOINC, use the CPU scavenging model. Since nodes are likely to go "offline" from time to time, as their owners use their resources for their primary purpose, this model must be designed to handle such contingencies.

2.6 History

The term *grid computing* originated in the early 1990s as a metaphor for making computer power as easy to access as an electric power grid. The power grid metaphor for accessible computing quickly became canonical when Ian Foster and Carl Kesselman published their seminal work, "The Grid: Blueprint for a new computing infrastructure" (1999).

CPU scavenging and volunteer computing were popularized beginning in 1997 by distributed.net and later in 1999 by SETI@home to harness the power of networked PCs worldwide, in order to solve CPU-intensive research problems.

The ideas of the grid (including those from distributed computing, object-oriented programming, and Web services) were brought together by Ian Foster, Carl Kesselman, and Steve Tuecke, widely regarded as the "fathers of the grid".[6] They led the effort to create the Globus Toolkit incorporating not just computation management but also storage management, security provisioning, data movement, monitoring, and a toolkit for developing additional services based on the same infrastructure, including agreement negotiation, notification mechanisms, trigger services, and information aggregation. While the Globus Toolkit remains the de facto standard for building grid solutions, a number of other tools have been built that answer some subset of services needed to create an enterprise or global grid.[7]

In 2007 the term cloud computing came into popularity, which is conceptually similar to the canonical Foster definition of grid computing (in terms of computing resources being consumed as electricity is from the power grid). Indeed, grid computing is often (but not always) associated with the delivery of cloud computing systems as exemplified by the AppLogic system from 3tera.

2.7 Fastest virtual supercomputers

- As of June 2014, Bitcoin Network – 1166652 PFLOPS.[8]
- As of April 2013, Folding@home – 11.4 x86-equivalent (5.8 "native") PFLOPS.[9]
- As of March 2013, BOINC – processing on average 9.2 PFLOPS.[10]
- As of April 2010, MilkyWay@Home computes at over 1.6 PFLOPS, with a large amount of this work coming from GPUs.[11]
- As of April 2010, SETI@Home computes data averages more than 730 TFLOPS.[12]
- As of April 2010, Einstein@Home is crunching more than 210 TFLOPS.[13]
- As of June 2011, GIMPS is sustaining 61 TFLOPS.[14]

2.8 Projects and applications

Main article: List of distributed computing projects

Grid computing offers a way to solve Grand Challenge problems such as protein folding, financial modeling, earthquake simulation, and climate/weather modeling. Grids offer a way of using the information technology resources optimally inside an organization. They also provide a means for offering information technology as a utility for commercial and noncommercial clients, with those clients paying only for what they use, as with electricity or water.

Grid computing is being applied by the National Science Foundation's National Technology Grid, NASA's Information Power Grid, Pratt & Whitney, Bristol-Myers Squibb Co., and American Express.

One cycle-scavenging network is SETI@home, which was using more than 3 million computers to achieve 23.37 sustained teraflops (979 lifetime teraflops) as of September 2001.[15]

As of August 2009 Folding@home achieves more than 4 petaflops on over 350,000 machines.

The European Union funded projects through the framework programmes of the European Commission. BEinGRID (Business Experiments in Grid) was a research project funded by the European Commission[16] as an Integrated Project under the Sixth Framework Programme (FP6) sponsorship program. Started on June 1, 2006, the project ran 42 months, until November 2009. The project was coordinated by Atos Origin. According to the project fact sheet, their mission is "to establish effective routes to foster the adoption of grid computing across the EU and to stimulate research into innovative business models using Grid technologies". To extract best practice and common themes from the experimental implementations, two groups of consultants are analyzing a series of pilots, one technical, one business. The project is significant not only for its long duration, but also for its budget, which at 24.8 million Euros, is the largest of any FP6 integrated project. Of this, 15.7 million is provided by the European commission and the remainder by its 98 contributing partner companies. Since the end of the project, the results of BEinGRID have been taken up and carried forward by IT-Tude.com.

The Enabling Grids for E-sciencE project, based in the European Union and included sites in Asia and the United States, was a follow-up project to the European DataGrid (EDG) and evoled into the European Grid Infrastructure. This, along with the LHC Computing Grid[17] (LCG), was developed to support experiments using the CERN Large Hadron Collider. A list of active sites participating within LCG can be found online[18] as can real time monitoring of the EGEE infrastructure.[19] The relevant software and documentation is also publicly accessible.[20] There is speculation that dedicated fiber optic links, such as those installed by CERN to address the LCG's data-intensive needs, may one day be available to home users thereby providing internet services at speeds up to 10,000 times faster than a traditional broadband connection.[21]

The European Grid Infrastructure has been also used for other research activities and experiments such as the simulation of oncological clinical trials.[22]

The distributed.net project was started in 1997. The NASA Advanced Supercomputing facility (NAS) ran genetic algorithms using the Condor cycle scavenger running on about 350 Sun Microsystems and SGI workstations.

In 2001, United Devices operated the United Devices Cancer Research Project based on its Grid MP product, which cycle-scavenges on volunteer PCs connected to the Internet. The project ran on about 3.1 million machines before its close in 2007.[23]

As of 2011, over 6.2 million machines running the open-source Berkeley Open Infrastructure for Network Computing (BOINC) platform are members of the World Community Grid, which tops the processing power of the current fastest supercomputer system (China's Tianhe-I).[24]

2.8.1 Definitions

Today there are many definitions of *grid computing*:

- In his article "What is the Grid? A Three Point Checklist",[3] Ian Foster lists these primary attributes:
  - Computing resources are not administered centrally.
  - Open standards are used.
  - Nontrivial quality of service is achieved.
- Plaszczak/Wellner[25] define grid technology as "the technology that enables resource virtualization, on-demand provisioning, and service (resource) sharing between organizations."
- IBM defines grid computing as "the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a vast array of other computing resources over the Internet. A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across 'multiple' administrative domains based on their (resources) availability, capacity, performance, cost and users' quality-of-service requirements".[26]
- An earlier example of the notion of computing as utility was in 1965 by MIT's Fernando Corbató. Corbató and the other designers of the Multics operating system envisioned a computer facility operating "like a power company or water company".[27]
- Buyya/Venugopal[28] define grid as "a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically

distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements".

- CERN, one of the largest users of grid technology, talk of **The Grid**: "a service for sharing computer power and data storage capacity over the Internet."[29]

2.9 See also

2.9.1 Related concepts

- Sensor grid
- Jungle computing
- Code mobility
- Cloud computing

2.9.2 Alliances and organizations

- Open Grid Forum (Formerly Global Grid Forum)
- Object Management Group

2.9.3 Production grids

- European Grid Infrastructure
- Enabling Grids for E-sciencE
- INFN Production Grid
- NorduGrid
- OurGrid
- Sun Grid
- Techila
- Xgrid

2.9.4 International projects

2.9.5 National projects

- GridPP (UK)
- CNGrid (China)
- D-Grid (Germany)
- GARUDA (India)
- VECC (Calcutta, India)
- IsraGrid (Israel)
- INFN Grid (Italy)
- PL-Grid (Poland)
- National Grid Service (UK)
- Open Science Grid (USA)
- TeraGrid (USA)
- Grid5000 (France)

2.9.6 Standards and APIs

- Distributed Resource Management Application API (DRMAA)
- A technology-agnostic information model for a uniform representation of Grid resources (GLUE)
- Grid Remote Procedure Call (GridRPC)
- Grid Security Infrastructure (GSI)
- Open Grid Services Architecture (OGSA)
- Open Grid Services Infrastructure (OGSI)
- A Simple API for Grid Applications (SAGA)
- Web Services Resource Framework (WSRF)

2.9.7 Software implementations and middleware

- Advanced Resource Connector (NorduGrid's ARC)
- Altair PBS GridWorks
- Berkeley Open Infrastructure for Network Computing (BOINC)
- DIET
- Discovery Net
- European Middleware Initiative
- gLite
- Globus Toolkit
- GridWay
- OurGrid
- Portable Batch System (PBS)
- Platform LSF
- LinuxPMI
- ProActive

- Platform Symphony
- SDSC Storage resource broker (data grid)
- Simple Grid Protocol
- Sun Grid Engine
- Techila Grid
- UNICORE
- Univa Grid Engine
- Xgrid
- ZeroC ICE IceGrid

2.9.8 Monitoring frameworks

- GStat

2.10 See also

- Jungle computing

2.11 References

[1] Grid vs cluster computing

[2] What is grid computing? - Gridcafe. E-sciencecity.org. Retrieved 2013-09-18.

[3] "What is the Grid? A Three Point Checklist".

[4] "Pervasive and Artificial Intelligence Group :: publications [Pervasive and Artificial Intelligence Research Group]". Diuf.unifr.ch. May 18, 2009. Retrieved July 29, 2010.

[5] Computational problems - Gridcafe. E-sciencecity.org. Retrieved 2013-09-18.

[6] "Father of the Grid".

[7] Alaa, Riad; Ahmed, Hassan; Qusay, Hassan (31 March 2010). "Design of SOA-based Grid Computing with Enterprise Service Bus". *INTERNATIONAL JOURNAL ON Advances in Information Sciences and Service Sciences* **2** (1): 71–82. doi:10.4156/aiss.vol2.issue1.6.

[8] bitcoinwatch.com (15 June 2014). "Bitcoin Network Statistics". *Bitcoin*. Staffordshire University. Retrieved June 15, 2014.

[9] Pande lab. "Client Statistics by OS". *Folding@home*. Stanford University. Retrieved April 23, 2013.

[10] "BOINCstats – BOINC combined credit overview". Retrieved March 3, 2013.

[11] "MilkyWay@Home Credit overview". BOINC. Retrieved April 21, 2010.

[12] "SETI@Home Credit overview". BOINC. Retrieved April 21, 2010.

[13] "Einstein@Home Credit overview". BOINC. Retrieved April 21, 2010.

[14] "Internet PrimeNet Server Distributed Computing Technology for the Great Internet Mersenne Prime Search". *GIMPS*. Retrieved June 6, 2011.

[15]

[16] Home page of BEinGRID

[17] Large Hadron Collider Computing Grid official homepage

[18] "GStat 2.0 – Summary View – GRID EGEE". Goc.grid.sinica.edu.tw. Retrieved July 29, 2010.

[19] "Real Time Monitor". Gridportal.hep.ph.ic.ac.uk. Retrieved July 29, 2010.

[20] "LCG – Deployment". Lcg.web.cern.ch. Retrieved July 29, 2010.

[21] "Coming soon: superfast internet"

[22] Athanaileas, Theodoros, et al. (2011). "Exploiting grid technologies for the simulation of clinical trials: the paradigm of in silico radiation oncology". *SIMULATION: Transactions of The Society for Modeling and Simulation International* (Sage Publications) **87** (10): 893–910. doi:10.1177/0037549710375437.

[23]

[24] BOINCstats

[25] P Plaszczak, R Wellner, *Grid computing*, 2005, Elsevier/Morgan Kaufmann, San Francisco

[26] IBM Solutions Grid for Business Partners: Helping IBM Business Partners to Grid-enable applications for the next phase of e-business on demand

[27] Structure of the Multics Supervisor. Multicians.org. Retrieved 2013-09-18.

[28] "A Gentle Introduction to Grid Computing and Technologies" (PDF). Retrieved May 6, 2005.

[29] "The Grid Café – The place for everybody to learn about grid computing". CERN. Retrieved December 3, 2008.

2.11.1 Bibliography

- Buyya, Rajkumar; Kris Bubendorfer (2009). *Market Oriented Grid and Utility Computing*. Wiley. ISBN 978-0-470-28768-2.
- Benedict, Shajulin; Vasudevan (2008). "A Niched Pareto GA approach for scheduling scientific workflows in wireless Grids". *Journal of Computing and Information Technology* **16**: 101. doi:10.2498/cit.1001122.

- Davies, Antony (June 2004). "Computational Intermediation and the Evolution of Computation as a Commodity" (PDF). *Applied Economics* **36** (11): 1131. doi:10.1080/0003684042000247334.
- Foster, Ian; Carl Kesselman (1999). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers. ISBN 1-55860-475-8.
- Plaszczak, Pawel; Rich Wellner, Jr (2006). *Grid Computing "The Savvy Manager's Guide"*. Morgan Kaufmann Publishers. ISBN 0-12-742503-9.
- Berman, Fran; Anthony J. G. Hey; Geoffrey C. Fox (2003). *Grid Computing: Making The Global Infrastructure a Reality*. Wiley. ISBN 0-470-85319-0.
- Li, Maozhen; Mark A. Baker (2005). *The Grid: Core Technologies*. Wiley. ISBN 0-470-09417-6.
- Catlett, Charlie; Larry Smarr (June 1992). "Metacomputing". *Communications of the ACM* **35** (6).
- Smith, Roger (2005). "Grid Computing: A Brief Technology Analysis" (PDF). CTO Network Library.
- Buyya, Rajkumar (July 2005). "Grid Computing: Making the Global Cyberinfrastructure for eScience a Reality" (PDF). *CSI Communications* (Mumbai, India: Computer Society of India (CSI)) **29** (1).
- Berstis, Viktors. "Fundamentals of Grid Computing". IBM.
- Elkhatib, Yehia (2011). *Monitoring, Analysing and Predicting Network Performance in Grids* (Ph.D.). Lancaster University.
- Ferreira, Luis; et al. "Grid Computing Products and Services". IBM.
- Ferreira, Luis; et al. "Introduction to Grid Computing with Globus". IBM.
- Jacob, Bart; et al. "Enabling Applications for Grid Computing". IBM.
- Ferreira, Luis; et al. "Grid Services Programming and Application Enablement". IBM.
- Jacob, Bart; et al. "Introduction to Grid Computing". IBM.
- Ferreira, Luis; et al. "Grid Computing in Research and Education". IBM.
- Ferreira, Luis; et al. "Globus Toolkit 3.0 Quick Start". IBM.
- Surridge, Mike; et al. "Experiences with GRIA – Industrial applications on a Web Services Grid" (PDF). IEEE.
- Stockinger, Heinz; et al. (October 2007). "Defining the Grid: A Snapshot on the Current View" (PDF). *Supercomputing* **42**: 3. doi:10.1007/s11227-006-0037-9.
- Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies
- The Grid Technology Cookbook
- Francesco Lelli, Eric Frizziero, Michele Gulmini, Gaetano Maron, Salvatore Orlando, Andrea Petrucci and Silvano Squizzato. *The many faces of the integration of instruments and the grid*. International Journal of Web and Grid Services 2007 – Vol. 3, No.3 pp. 239 – 266 Electronic Edition
- Poess, Meikel; Nambiar, Raghunath (2005). *Large Scale Data Warehouses on Grid*.
- Pardi, Silvio; Francesco Palmieri (October 2010). "Towards a federated Metropolitan Area Grid environment: The SCoPE network-aware infrastructure". *Future Generation Computer Systems* **26**. doi:10.1016/j.future.2010.02.0039.

2.12 External links

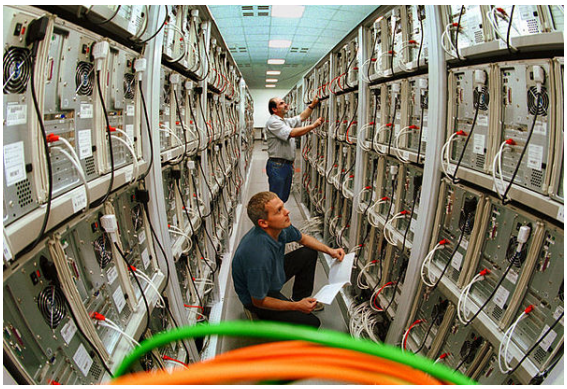
- GridCafé—an layperson's introduction to grid computing and how it works
- SuGI-Portal—more on grids.

Chapter 3

Computer cluster

Not to be confused with data cluster or computer lab.

A **computer cluster** consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. Unlike grid computers, computer clusters have each node set to perform the same task, controlled and scheduled by software.[1]



Technicians working on a large Linux cluster at the Chemnitz University of Technology, Germany

Sun Microsystems Solaris Cluster

The components of a cluster are usually connected to each other through fast local area networks ("LAN"), with each *node* (computer used as a server) running its own instance of an operating system. In most circumstances, all of the nodes use the same hardware[2] and the same operating system, although in some setups (i.e. using Open Source Cluster Application Resources (OSCAR)), different operating systems can be used on each computer, and/or different hardware.[3]

They are usually deployed to improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.[4]

Computer clusters emerged as a result of convergence of a number of computing trends including the availability of low cost microprocessors, high speed networks, and software for high-performance distributed computing. They have a wide range of applicability and deployment, ranging from small business clusters with a handful of nodes to some of the fastest supercomputers in the world such as IBM's Sequoia.[5] The applications that can be done however, are nonetheless limited, since the software needs to be purpose-built per task. It is hence not possible to use computer clusters for casual computing tasks.[6]

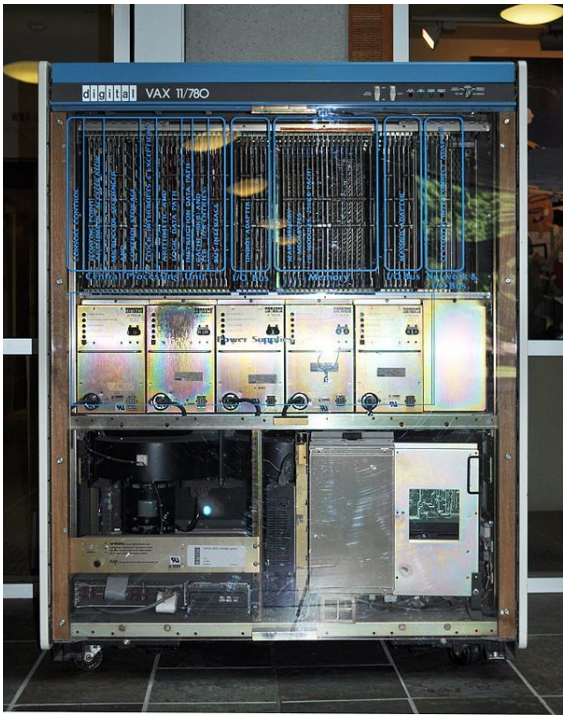
3.1 Basic concepts

The desire to get more computing power and better reliability by orchestrating a number of low cost commercial off-the-shelf computers has given rise to a variety of architectures and configurations.

The computer clustering approach usually (but not always) connects a number of readily available computing nodes (e.g. personal computers used as servers) via a fast local area network.[7] The activities of the computing nodes are orchestrated by "clustering middleware", a software layer that sits atop the nodes and allows the users to treat the cluster as by and large one cohesive computing unit, e.g. via a single system image concept.[7]

Computer clustering relies on a centralized management approach which makes the nodes available as orchestrated shared servers. It is distinct from other approaches such as peer to peer or grid computing which also use many nodes, but with a far more distributed nature.[7]

A simple, home-built Beowulf cluster.



A VAX 11/780, c. 1977

A computer cluster may be a simple two-node system which just connects two personal computers, or may be a very fast supercomputer. A basic approach to building a cluster is that of a Beowulf cluster which may be built with a few personal computers to produce a cost-effective alternative to traditional high performance computing. An early project that showed the viability of the concept was the 133 nodes Stone Soupercomputer.[8] The developers used Linux, the Parallel Virtual Machine toolkit and the Message Passing Interface library to achieve high performance at a relatively low cost.[9]

Although a cluster may consist of just a few personal computers connected by a simple network, the cluster architecture may also be used to achieve very high levels of performance. The TOP500 organization's semiannual list of the 500 fastest supercomputers often includes many clusters, e.g. the world's fastest machine in 2011 was the K computer which has a distributed memory, cluster architecture.[10][11]

3.2 History

Main article: History of computer clusters
See also: History of supercomputing

Greg Pfister has stated that clusters were not invented by any specific vendor but by customers who could not fit all their work on one computer, or needed a backup.[12] Pfister estimates the date as some time in the 1960s. The formal engineering basis of cluster computing as a means of doing parallel work of any sort was arguably invented by Gene Amdahl of IBM, who in 1967 published what has come to be regarded as the seminal paper on parallel processing: Amdahl's Law.

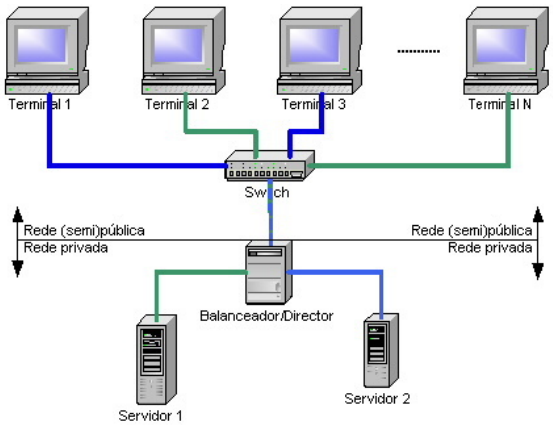
The history of early computer clusters is more or less directly tied into the history of early networks, as one of the primary motivations for the development of a network was to link computing resources, creating a de facto computer cluster.

The first commercial clustering product was Datapoint Corporation's "Attached Resource Computer" (ARC) system, developed in 1977, and using ARCnet as the cluster interface. Clustering per se did not really take off until Digital Equipment Corporation released their VAXcluster product in 1984 for the VAX/VMS operating system (now named as OpenVMS). The ARC and VAXcluster products not only supported parallel computing, but also shared file systems and peripheral devices. The idea was to provide the advantages of parallel processing, while maintaining data reliability and uniqueness. Two other noteworthy early commercial clusters were the *Tandem Himalayan* (a circa 1994 high-availability product) and the *IBM S/390 Parallel Sysplex* (also circa 1994, primarily for business use).

Within the same time frame, while computer clusters used parallelism outside the computer on a commodity network, supercomputers began to use them within the same computer. Following the success of the CDC 6600 in 1964, the Cray 1 was delivered in 1976, and introduced internal parallelism via vector processing.[13] While early supercomputers excluded clusters and relied on shared

memory, in time some of the fastest supercomputers (e.g. the K computer) relied on cluster architectures.

3.3 Attributes of clusters



A load balancing cluster with two servers and N user stations (Galician).

Computer clusters may be configured for different purposes ranging from general purpose business needs such as web-service support, to computation-intensive scientific calculations. In either case, the cluster may use a high-availability approach. Note that the attributes described below are not exclusive and a "computer cluster" may also use a high-availability approach, etc.

"Load-balancing" clusters are configurations in which cluster-nodes share computational workload to provide better overall performance. For example, a web server cluster may assign different queries to different nodes, so the overall response time will be optimized.[14] However, approaches to load-balancing may significantly differ among applications, e.g. a high-performance cluster used for scientific computations would balance load with different algorithms from a web-server cluster which may just use a simple round-robin method by assigning each new request to a different node.[14]

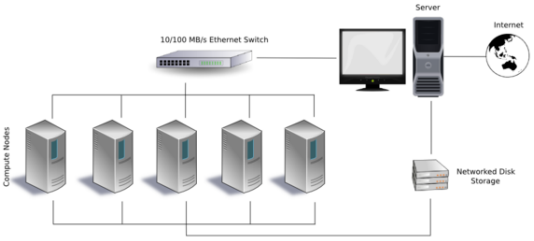
Computer clusters are used for computation-intensive purposes, rather than handling IO-oriented operations such as web service or databases.[15] For instance, a computer cluster might support computational simulations of vehicle crashes or weather. Very tightly coupled computer clusters are designed for work that may approach "supercomputing".

"High-availability clusters" (also known as failover clusters, or HA clusters) improve the availability of the cluster approach. They operate by having redundant nodes, which are then used to provide service when system components fail. HA cluster implementations attempt to use redundancy of cluster components to eliminate single points of failure. There are commercial implementations of High-Availability clusters for many operating systems. The Linux-HA project is one commonly used free software HA package for the Linux operating system.

3.4 Benefits

Clusters are primarily designed with performance in mind, but installations are based on many other factors; fault tolerance (*the ability for a system to continue working with a malfunctioning node*) also allows for simpler scalability, and in high performance situations, low frequency of maintenance routines, resource consolidation, and centralized management.[16][17]

3.5 Design and Configuration



A typical Beowulf configuration

One of the issues in designing a cluster is how tightly coupled the individual nodes may be. For instance, a single computer job may require frequent communication among nodes: this implies that the cluster shares a dedicated network, is densely located, and probably has homogeneous nodes. The other extreme is where a computer job uses one or few nodes, and needs little or no inter-node communication, approaching grid computing.

In a Beowulf system, the application programs never see the computational nodes (also called slave computers) but only interact with the "Master" which is a specific computer handling the scheduling and management of the slaves.[15] In a typical implementation the Master has two network interfaces, one that communicates with the private Beowulf network for the slaves, the other for the general purpose network of the organization.[15] The slave computers typically have their own version of the same operating system, and local memory and disk space. However, the private slave network may also have a large and shared file server that stores global persistent data, accessed by the slaves as needed.[15]

By contrast, the special purpose 144 node DEGIMA cluster is tuned to running astrophysical N-body simulations

using the Multiple-Walk parallel treecode, rather than general purpose scientific computations.[18]

Due to the increasing computing power of each generation of game consoles, a novel use has emerged where they are repurposed into High-performance computing (HPC) clusters. Some examples of game console clusters are Sony PlayStation clusters and Microsoft Xbox clusters. Another example of consumer game product is the Nvidia Tesla Personal Supercomputer workstation, which uses multiple graphics accelerator processor chips. Besides game consoles, high-end graphics cards too can be used instead. The use of graphics cards (or rather their GPU's) to do calculations for grid computing is vastly more economical than using CPU's, despite being less precise. However, when using double-precision values, they become as precise to work with as CPU's, and still be much less costly (purchase cost).[19]

Computer clusters have historically run on separate physical computers with the same operating system. With the advent of virtualization, the cluster nodes may run on separate physical computers with different operating systems which are painted above with a virtual layer to look similar.[20] The cluster may also be virtualized on various configurations as maintenance takes place. An example implementation is Xen as the virtualization manager with Linux-HA.[21]

3.6 Data sharing and communication

3.6.1 Data sharing

A NEC Nehalem cluster

As the computer clusters were appearing during the 1980s, so were supercomputers. One of the elements that distinguished the three classes at that time was that the early supercomputers relied on shared memory. To date clusters do not typically use physically shared memory, while many supercomputer architectures have also abandoned it.

However, the use of a clustered file system is essential in modern computer clusters. Examples include the IBM General Parallel File System, Microsoft's Cluster Shared Volumes or the Oracle Cluster File System.

3.6.2 Message passing and communication

Main article: Message passing in computer clusters

Two widely used approaches for communication between cluster nodes are MPI, the Message Passing Interface and PVM, the Parallel Virtual Machine.[22]

PVM was developed at the Oak Ridge National Laboratory around 1989 before MPI was available. PVM must be directly installed on every cluster node and provides a set of software libraries that paint the node as a "parallel virtual machine". PVM provides a run-time environment for message-passing, task and resource management, and fault notification. PVM can be used by user programs written in C, C++, or Fortran, etc.[22][23]

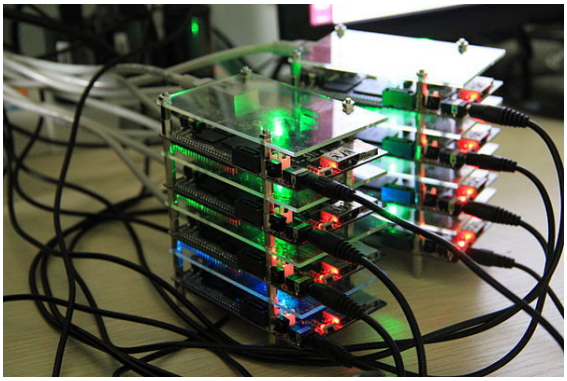
MPI emerged in the early 1990s out of discussions among 40 organizations. The initial effort was supported by ARPA and National Science Foundation. Rather than starting anew, the design of MPI drew on various features available in commercial systems of the time. The MPI specifications then gave rise to specific implementations. MPI implementations typically use TCP/IP and socket connections.[22] MPI is now a widely available communications model that enables parallel programs to be written in languages such as C, Fortran, Python, etc.[23] Thus, unlike PVM which provides a concrete implementation, MPI is a specification which has been implemented in systems such as MPICH and Open MPI.[23][24]

3.7 Cluster management

One of the challenges in the use of a computer cluster is the cost of administrating it which can at times be as high as the cost of administrating N independent machines, if the cluster has N nodes.[25] In some cases this provides an advantage to shared memory architectures with lower administration costs.[25] This has also made virtual machines popular, due to the ease of administration.[25]

3.7.1 Task scheduling

When a large multi-user cluster needs to access very large amounts of data, task scheduling becomes a challenge.



Low-cost and low energy tiny-cluster of Cubieboards, using Apache Hadoop on Lubuntu

In a heterogeneous CPU-GPU cluster with a complex application environment, the performance of each job depends on the characteristics of the underlying cluster. Therefore, mapping tasks onto CPU cores and GPU devices provides significant challenges.[26] This is an area of ongoing research; algorithms that combine and extend MapReduce and Hadoop have been proposed and studied.[26]

3.7.2 Node failure management

When a node in a cluster fails, strategies such as "fencing" may be employed to keep the rest of the system operational.[27][28] Fencing is the process of isolating a node or protecting shared resources when a node appears to be malfunctioning. There are two classes of fencing methods; one disables a node itself, and the other disallows access to resources such as shared disks.[27]

The STONITH method stands for "Shoot The Other Node In The Head", meaning that the suspected node is disabled or powered off. For instance, *power fencing* uses a power controller to turn off an inoperable node.[27]

The *resources fencing* approach disallows access to resources without powering off the node. This may include *persistent reservation fencing* via the SCSI3, fibre channel fencing to disable the fibre channel port, or global network block device (GNBD) fencing to disable access to the GNBD server.

3.8 Software development and administration

3.8.1 Parallel programming

Load balancing clusters such as web servers use cluster architectures to support a large number of users and typically each user request is routed to a specific node, achieving task parallelism without multi-node cooperation, given that the main goal of the system is providing rapid user access to shared data. However, "computer clusters" which perform complex computations for a small number of users need to take advantage of the parallel processing capabilities of the cluster and partition "the same computation" among several nodes.[29]

Automatic parallelization of programs continues to remain a technical challenge, but parallel programming models can be used to effectuate a higher degree of parallelism via the simultaneous execution of separate portions of a program on different processors.[29][30]

3.8.2 Debugging and monitoring

The development and debugging of parallel programs on a cluster requires parallel language primitives as well as suitable tools such as those discussed by the *High Performance Debugging Forum* (HPDF) which resulted in the HPD specifications.[23][31] Tools such as TotalView were then developed to debug parallel implementations on computer clusters which use MPI or PVM for message passing.

The Berkeley NOW (Network of Workstations) system gathers cluster data and stores them in a database, while a system such as PARMON, developed in India, allows for the visual observation and management of large clusters.[23]

Application checkpointing can be used to restore a given state of the system when a node fails during a long multi-node computation.[32] This is essential in large clusters, given that as the number of nodes increases, so does the likelihood of node failure under heavy computational loads. Checkpointing can restore the system to a stable state so that processing can resume without having to recompute results.[32]

3.9 Some implementations

The GNU/Linux world supports various cluster software; for application clustering, there is distcc, and MPICH. Linux Virtual Server, Linux-HA - director-based clusters that allow incoming requests for services to be distributed across multiple cluster nodes. MOSIX, LinuxPMI, Kerrighed, OpenSSI are full-blown clusters integrated into the kernel that provide for automatic process migration among homogeneous nodes. OpenSSI, openMosix and Kerrighed are single-system image implementations.

Microsoft Windows computer cluster Server 2003 based on the Windows Server platform provides pieces for High Performance Computing like the Job Scheduler, MSMPI library and management tools.

gLite is a set of middleware technologies created by the Enabling Grids for E-sciencE (EGEE) project.

slurm is also used to schedule and manage some of the largest supercomputer clusters (see top500 list).

3.10 Other approaches

Although most computer clusters are permanent fixtures, attempts at flash mob computing have been made to build short-lived clusters for specific computations. However, larger scale volunteer computing systems such as BOINC-based systems have had more followers.

3.11 See also

3.12 References

[1] Grid vs cluster computing

[2] Cluster vs grid computing

[3] Hardware of computer clusters not always needing to be the same, probably depends on software used

[4] Bader, David; Robert Pennington (June 1996). "Cluster Computing: Applications". Georgia Tech College of Computing. Retrieved 2007-07-13.

[5] "Nuclear weapons supercomputer reclaims world speed record for US". The Telegraph. 18 Jun 2012. Retrieved 18 Jun 2012.

[6] Grid and cluster computing, limitations

[7] *Network-Based Information Systems: First International Conference, NBIS 2007* ISBN 3-540-74572-6 page 375

[8] William W. Hargrove, Forrest M. Hoffman and Thomas Sterling (August 16, 2001). "The Do-It-Yourself Supercomputer". *Scientific American* **265** (2). pp. 72–79. Retrieved October 18, 2011.

[9] William W. Hargrove and Forrest M. Hoffman (1999). "Cluster Computing: Linux Taken to the Extreme". *Linux magazine*. Retrieved October 18, 2011.

[10] TOP500 list To view all clusters on the TOP500 select "cluster" as architecture from the sublist menu.

[11] M. Yokokawa et al *The K Computer*, in "International Symposium on Low Power Electronics and Design" (ISLPED) 1-3 Aug. 2011, pages 371-372

[12] Pfister, Gregory (1998). *In Search of Clusters* (2nd ed.). Upper Saddle River, NJ: Prentice Hall PTR. p. 36. ISBN 0-13-899709-8.

[13] *Readings in computer architecture* by Mark Donald Hill, Norman Paul Jouppi, Gurindar Sohi 1999 ISBN 978-1-55860-539-8 page 41-48

[14] *High Performance Linux Clusters* by Joseph D. Sloan 2004 ISBN 0-596-00570-9 page

[15] *High Performance Computing for Computational Science - VECPAR 2004* by Michel Daydé, Jack Dongarra 2005 ISBN 3-540-25424-2 pages 120-121

[16] "IBM Cluster System : Benefits". *http://www-03.ibm.com/''. IBM. Retrieved 8 September 2014.*

[17] "Evaluating the Benefits of Clustering". *http://www.microsoft.com/''. Microsoft. 28 March 2003. Retrieved 8 September 2014.*

[18] Hamada T. *et al.* (2009) A novel multiple-walk parallel algorithm for the Barnes–Hut treecode on GPUs – towards cost effective, high performance N-body simulation. *Comput. Sci. Res. Development* 24:21-31. doi:10.1007/s00450-009-0089-1

[19] GPU options

[20] Using Xen

[21] Maurer, Ryan: Xen Virtualization and Linux Clustering

[22] *Distributed services with OpenAFS: for enterprise and education* by Franco Milicchio, Wolfgang Alexander Gehrke 2007, ISBN pages 339-341

[23] *Grid and Cluster Computing* by Prabhu 2008 8120334280 pages 109-112

[24] Gropp, William; Lusk, Ewing; Skjellum, Anthony (1996). "A High-Performance, Portable Implementation of the MPI Message Passing Interface". *Parallel Computing*. CiteSeerX: 10.1.1.102.9485.

[25] *Computer Organization and Design* by David A. Patterson and John L. Hennessy 2011 ISBN 0-12-374750-3 pages 641-642

[26] K. Shirahata, et al *Hybrid Map Task Scheduling for GPU-Based Heterogeneous Clusters* in: Cloud Computing Technology and Science (CloudCom), 2010 Nov. 30 2010-Dec. 3 2010 pages 733 - 740 ISBN 978-1-4244-9405-7

[27] Alan Robertson *Resource fencing using STONITH*. IBM Linux Research Center, 2010

[28] *Sun Cluster environment: Sun Cluster 2.2* by Enrique Vargas, Joseph Bianco, David Deeths 2001 ISBN page 58

[29] *Computer Science: The Hardware, Software and Heart of It* by Alfred V. Aho, Edward K. Blum 2011 ISBN 1-4614-1167-X pages 156-166

[30] *Parallel Programming: For Multicore and Cluster Systems* by Thomas Rauber, Gudula Rünger 2010 ISBN 3-642-04817-X pages 94–95

[31] *A debugging standard for high-performance computing* by Joan M. Francioni and Cherri Pancake, in the "Journal of Scientific Programming" Volume 8 Issue 2, April 2000

[32] *Computational Science-- ICCS 2003: International Conference* edited by Peter Sloot 2003 ISBN 3-540-40195-4 pages 291-292

3.13 Further reading

- Mark Baker, et al., *Cluster Computing White Paper* , 11 Jan 2001.
- Evan Marcus, Hal Stern: *Blueprints for High Availability: Designing Resilient Distributed Systems*, John Wiley & Sons, ISBN 0-471-35601-8
- Greg Pfister: *In Search of Clusters*, Prentice Hall, ISBN 0-13-899709-8
- Rajkumar Buyya (editor): *High Performance Cluster Computing: Architectures and Systems*, Volume 1, ISBN 0-13-013784-7, and Volume 2, ISBN 0-13-013785-5, Prentice Hall, NJ, USA, 1999.

3.14 External links

- IEEE Technical Committee on Scalable Computing (TCSC)
- Reliable Scalable Cluster Technology, IBM
- Tivoli System Automation Wiki

Chapter 4

Supercomputer

"High-performance computing" redirects here. For narrower definitions of HPC, see high-throughput computing and many-task computing. For other uses, see Supercomputer (disambiguation).

A **supercomputer** is a computer that has world-

The Blue Gene/P supercomputer at Argonne National Lab runs over 250,000 processors using normal data center air conditioning, grouped in 72 racks/cabinets connected by a high-speed optical network[1]

class computational capacity. In 2015, such machines can perform quadrillions of floating point operations per second.[2]

Supercomputers were introduced in the 1960s, made initially and, for decades, primarily by Seymour Cray at Control Data Corporation (CDC), Cray Research and subsequent companies bearing his name or monogram. While the supercomputers of the 1970s used only a few processors, in the 1990s machines with thousands of processors began to appear and, by the end of the 20th century, massively parallel supercomputers with tens of thousands of "off-the-shelf" processors were the norm.[3][4] As of November 2014, China's Tianhe-2 supercomputer is the fastest in the world at 33.86 petaFLOPS (PFLOPS), or 33.86 quadrillion floating point operations per second.

Systems with massive numbers of processors generally take one of two paths: In one approach (e.g., in distributed computing), a large number of discrete computers (e.g., laptops) distributed across a network (e.g., the Internet) devote some or all of their time to solving a common problem; each individual computer (client) receives and completes many small tasks, reporting the results to a central server which integrates the task results from all the clients into the overall solution.[5][6] In another approach, a large number of dedicated processors are placed in close proximity to each other (e.g. in a computer cluster); this saves considerable time moving data around and makes it possible for the processors to work together (rather than on separate tasks), for example in mesh and hypercube architectures.

The use of multi-core processors combined with centralization is an emerging trend; one can think of this as a small cluster (the multicore processor in a smartphone, tablet, laptop, etc.) that both depends upon and contributes to the cloud.[7][8]

Supercomputers play an important role in the field of computational science, and are used for a wide range of computationally intensive tasks in various fields, including quantum mechanics, weather forecasting, climate research, oil and gas exploration, molecular modeling (computing the structures and properties of chemical compounds, biological macromolecules, polymers, and crystals), and physical simulations (such as simulations of the early moments of the universe, airplane and spacecraft aerodynamics, the detonation of nuclear weapons, and nuclear fusion). Throughout their history, they have been essential in the field of cryptanalysis.[9]

4.1 History

Main article: History of supercomputing

The history of supercomputing goes back to the 1960s, with the Atlas at the University of Manchester and a series of computers at Control Data Corporation (CDC), designed by Seymour Cray. These used innovative designs and parallelism to achieve superior computational peak performance.[10]

The Atlas was a joint venture between Ferranti and the Manchester University and was designed to operate at processing speeds approaching one microsecond per instruction, about one million instructions per second.[11] The first Atlas was officially commissioned on 7 December 1962 as one of the world's first supercomputers – con-

A Cray-1 preserved at the Deutsches Museum

sidered to be the most powerful computer in the world at that time by a considerable margin, and equivalent to four IBM 7094s.[12]

The CDC 6600, released in 1964, was designed by Cray to be the fastest in the world by a large margin. Cray switched from germanium to silicon transistors, which he ran very fast, solving the overheating problem by introducing refrigeration.[13] Given that the 6600 outran all computers of the time by about 10 times, it was dubbed a *supercomputer* and defined the supercomputing market when one hundred computers were sold at $8 million each.[14][15][16][17]

Cray left CDC in 1972 to form his own company, Cray Research.[15] Four years after leaving CDC, Cray delivered the 80 MHz Cray 1 in 1976, and it became one of the most successful supercomputers in history.[18][19] The Cray-2 released in 1985 was an 8 processor liquid cooled computer and Fluorinert was pumped through it as it operated. It performed at 1.9 gigaflops and was the world's fastest until 1990.[20]

While the supercomputers of the 1980s used only a few processors, in the 1990s, machines with thousands of processors began to appear both in the United States and Japan, setting new computational performance records. Fujitsu's Numerical Wind Tunnel supercomputer used 166 vector processors to gain the top spot in 1994 with a peak speed of 1.7 gigaFLOPS (GFLOPS) per processor.[21][22] The Hitachi SR2201 obtained a peak performance of 600 GFLOPS in 1996 by using 2048 processors connected via a fast three-dimensional crossbar network.[23][24][25] The Intel Paragon could have 1000 to 4000 Intel i860 processors in various configurations, and was ranked the fastest in the world in 1993. The Paragon was a MIMD machine which connected processors via a high speed two dimensional mesh, allowing processes to execute on separate nodes; communicating via the Message Passing Interface.[26]

4.2 Hardware and architecture

Main articles: Supercomputer architecture and Parallel computer hardware

Approaches to supercomputer architecture have taken

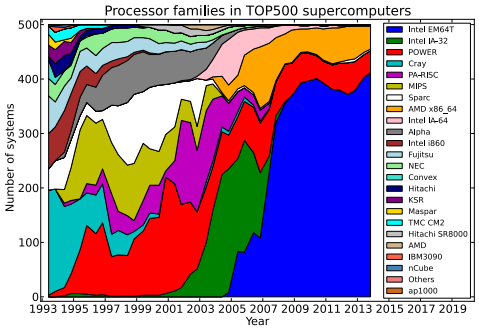
A Blue Gene/L cabinet showing the stacked blades, each holding many processors

dramatic turns since the earliest systems were introduced in the 1960s. Early supercomputer architectures pioneered by Seymour Cray relied on compact innovative designs and local parallelism to achieve superior computational peak performance.[10] However, in time the demand for increased computational power ushered in the age of massively parallel systems.

While the supercomputers of the 1970s used only a few processors, in the 1990s, machines with thousands of processors began to appear and by the end of the 20th century, massively parallel supercomputers with tens of thousands of "off-the-shelf" processors were the norm. Supercomputers of the 21st century can use over 100,000 processors (some being graphic units) connected by fast connections.[3][4] The Connection Machine CM-5 supercomputer is a massively parallel processing computer

capable of many billions of arithmetic operations per second.[27]

Throughout the decades, the management of heat density has remained a key issue for most centralized supercomputers.[28][29][30] The large amount of heat generated by a system may also have other effects, e.g. reducing the lifetime of other system components.[31] There have been diverse approaches to heat management, from pumping Fluorinert through the system, to a hybrid liquid-air cooling system or air cooling with normal air conditioning temperatures.[20][32]



The CPU share of TOP500

Systems with a massive number of processors generally take one of two paths. In the grid computing approach, the processing power of a large number of computers, organised as distributed, diverse administrative domains, is opportunistically used whenever a computer is available.[5] In another approach, a large number of processors are used in close proximity to each other, e.g. in a computer cluster. In such a centralized massively parallel system the speed and flexibility of the interconnect becomes very important and modern supercomputers have used various approaches ranging from enhanced Infiniband systems to three-dimensional torus interconnects.[33][34] The use of multi-core processors combined with centralization is an emerging direction, e.g. as in the Cyclops64 system.[7][8]

As the price, performance and energy efficiency of general purpose graphic processors (GPGPUs) have improved,[35] a number of petaflop supercomputers such as Tianhe-I and Nebulae have started to rely on them.[36] However, other systems such as the K computer continue to use conventional processors such as SPARC-based designs and the overall applicability of GPGPUs in general-purpose high-performance computing applications has been the subject of debate, in that while a GPGPU may be tuned to score well on specific benchmarks, its overall applicability to everyday algorithms may be limited unless significant effort is spent to tune the application towards it.[37] However, GPUs are gaining ground and in 2012 the Jaguar supercomputer was transformed into Titan by retrofitting CPUs with GPUs.[38][39][40]

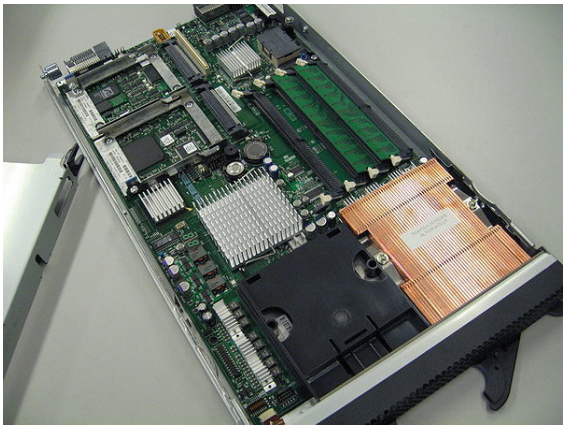
High performance computers have an expected life cycle of about three years.[41]

A number of "special-purpose" systems have been designed, dedicated to a single problem. This allows the use of specially programmed FPGA chips or even custom VLSI chips, allowing better price/performance ratios by sacrificing generality. Examples of special-purpose supercomputers include Belle,[42] Deep Blue,[43] and Hydra,[44] for playing chess, Gravity Pipe for astrophysics,[45] MDGRAPE-3 for protein structure computation molecular dynamics[46] and Deep Crack,[47] for breaking the DES cipher.

4.2.1 Energy usage and heat management

See also: Computer cooling and Green 500

A typical supercomputer consumes large amounts of electrical power, almost all of which is converted into heat, requiring cooling. For example, Tianhe-1A consumes 4.04 megawatts of electricity.[48] The cost to power and cool the system can be significant, e.g. 4 MW at $0.10/kWh is $400 an hour or about $3.5 million per year.



An IBM HS20 blade

Heat management is a major issue in complex electronic devices, and affects powerful computer systems in various ways.[49] The thermal design power and CPU power dissipation issues in supercomputing surpass those of traditional computer cooling technologies. The supercomputing awards for green computing reflect this issue.[50][51][52]

The packing of thousands of processors together inevitably generates significant amounts of heat density that need to be dealt with. The Cray 2 was liquid cooled, and used a Fluorinert "cooling waterfall" which was forced through the modules under pressure.[20] However, the submerged liquid cooling approach was not practical for the multi-cabinet systems based on off-the-shelf processors, and in System X a special cooling system that combined air conditioning with liquid cooling was developed in conjunction with the Liebert company.[32]

In the Blue Gene system, IBM deliberately used low power processors to deal with heat density.[53] On the other hand, the IBM Power 775, released in 2011, has closely packed elements that require water cooling.[54] The IBM Aquasar system, on the other hand uses *hot water cooling* to achieve energy efficiency, the water being used to heat buildings as well.[55][56]

The energy efficiency of computer systems is generally measured in terms of "FLOPS per Watt". In 2008, IBM's Roadrunner operated at 3,76 MFLOPS/W.[57][58] In November 2010, the Blue Gene/Q reached 1,684 MFLOPS/W.[59][60] In June 2011 the top 2 spots on the Green 500 list were occupied by Blue Gene machines in New York (one achieving 2097 MFLOPS/W) with the DEGIMA cluster in Nagasaki placing third with 1375 MFLOPS/W.[61]

Because copper wires can transfer energy into a supercomputer with much higher power densities than forced air or circulating refrigerants can remove waste heat,[62] The ability of the cooling systems to remove waste heat is a limiting factor.[63][64] As of 2015, many existing supercomputers have more infrastructure capacity than the actual peak demand of the machine – people conservatively designed the power and cooling infrastructure to handle more than the theoretical peak electrical power consumed by the supercomputer. Designs for future supercomputers are power-limited – the thermal design power of the supercomputer as a whole, the amount that the power and cooling infrastructure can handle, is somewhat more than the expected normal power consumption, but less than the theoretical peak power consumption of the electronic hardware.[65]

4.3 Software and system management

4.3.1 Operating systems

Main article: Supercomputer operating systems

Since the end of the 20th century, supercomputer operating systems have undergone major transformations, based on the changes in supercomputer architecture.[66] While early operating systems were custom tailored to each supercomputer to gain speed, the trend has been to move away from in-house operating systems to the adaptation of generic software such as Linux.[67]

Since modern massively parallel supercomputers typically separate computations from other services by using multiple types of nodes, they usually run different operating systems on different nodes, e.g. using a small and efficient lightweight kernel such as CNK or CNL on compute nodes, but a larger system such as a Linux-derivative on server and I/O nodes.[68][69][70]

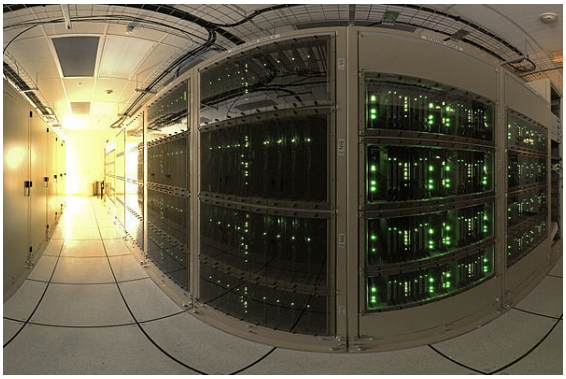
While in a traditional multi-user computer system job scheduling is, in effect, a tasking problem for processing and peripheral resources, in a massively parallel system, the job management system needs to manage the allocation of both computational and communication resources, as well as gracefully deal with inevitable hardware failures when tens of thousands of processors are present.[71]

Although most modern supercomputers use the Linux operating system, each manufacturer has its own specific Linux-derivative, and no industry standard exists, partly due to the fact that the differences in hardware architectures require changes to optimize the operating system to each hardware design.[66][72]

4.3.2 Software tools and message passing

Main article: Message passing in computer clusters
See also: Parallel computing and Parallel programming model

The parallel architectures of supercomputers often dictate the use of special programming techniques to exploit their speed. Software tools for distributed processing include standard APIs such as MPI and PVM, VTL, and open source-based software solutions such as Beowulf.



Wide-angle view of the ALMA correlator.[73]

In the most common scenario, environments such as PVM and MPI for loosely connected clusters and OpenMP for tightly coordinated shared memory machines are used. Significant effort is required to optimize an algorithm for the interconnect characteristics of the machine it will be run on; the aim is to prevent any of the CPUs from wasting time waiting on data from other nodes. GPGPUs have hundreds of processor cores and are programmed using programming models such as CUDA.

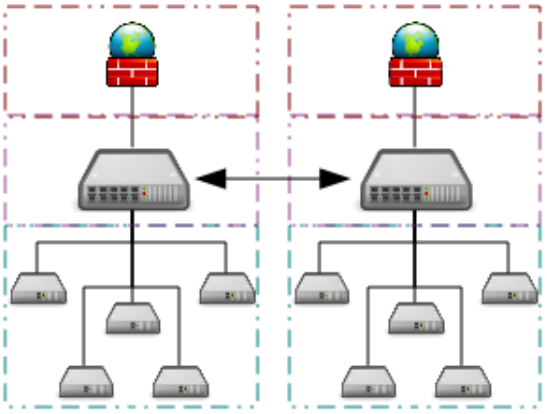
Moreover, it is quite difficult to debug and test parallel programs. Special techniques need to be used for testing and debugging such applications.

4.4 Distributed supercomputing

4.4.1 Opportunistic approaches

Main article: Grid computing

Opportunistic Supercomputing is a form of networked grid computing whereby a "super virtual computer" of many loosely coupled volunteer computing machines performs very large computing tasks. Grid computing has been applied to a number of large-scale embarrassingly parallel problems that require supercomputing performance scales. However, basic grid and cloud computing approaches that rely on volunteer computing can not handle traditional supercomputing tasks such as fluid dynamic simulations.



Example architecture of a grid computing system connecting many personal computers over the internet

The fastest grid computing system is the distributed computing project Folding@home. F@h reported 43.1 PFLOPS of x86 processing power as of June 2014. Of this, 42.5 PFLOPS are contributed by clients running on various GPUs, and the rest from various CPU systems.[74]

The BOINC platform hosts a number of distributed computing projects. As of May 2011, BOINC recorded a processing power of over 5.5 PFLOPS through over 480,000 active computers on the network[75] The most active project (measured by computational power), MilkyWay@home, reports processing power of over 700 teraFLOPS (TFLOPS) through over 33,000 active computers.[76]

As of May 2011, GIMPS's distributed Mersenne Prime search currently achieves about 60 TFLOPS through over 25,000 registered computers.[77] The Internet PrimeNet Server supports GIMPS's grid computing approach, one of the earliest and most successful grid computing projects, since 1997.

4.4.2 Quasi-opportunistic approaches

Main article: Quasi-opportunistic supercomputing

Quasi-opportunistic supercomputing is a form of distributed computing whereby the "super virtual computer" of a large number of networked geographically disperse computers performs computing tasks that demand huge processing power.[78] Quasi-opportunistic supercomputing aims to provide a higher quality of service than opportunistic grid computing by achieving more control over the assignment of tasks to distributed resources and the use of intelligence about the availability and reliability of individual systems within the supercomputing network. However, quasi-opportunistic distributed execution of demanding parallel computing software in grids should be achieved through implementation of grid-wise allocation agreements, co-allocation subsystems, communication topology-aware allocation mechanisms, fault tolerant message passing libraries and data pre-conditioning.[78]

4.5 Performance measurement

4.5.1 Capability vs capacity

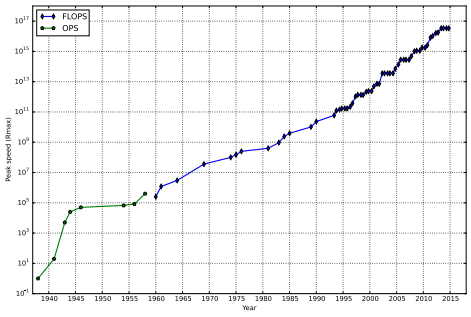
Supercomputers generally aim for the maximum in *capability computing* rather than *capacity computing*. Capability computing is typically thought of as using the maximum computing power to solve a single large problem in the shortest amount of time. Often a capability system is able to solve a problem of a size or complexity that no other computer can, e.g. a very complex weather simulation application.[79]

Capacity computing, in contrast, is typically thought of as using efficient cost-effective computing power to solve a small number of somewhat large problems or a large number of small problems.[79] Architectures that lend themselves to supporting many users for routine everyday tasks may have a lot of capacity, but are not typically considered supercomputers, given that they do not solve a single very complex problem.[79]

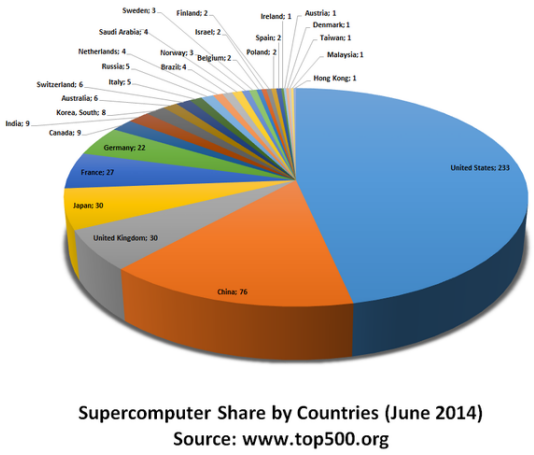
4.5.2 Performance metrics

See also: LINPACK benchmarks

In general, the speed of supercomputers is measured and benchmarked in "FLOPS" (*FLoating point Operations Per Second*), and not in terms of "MIPS" (Million Instructions Per Second), as is the case with general-purpose computers.[80] These measurements are commonly used with an SI prefix such as tera-, combined into the shorthand "TFLOPS" (10^{12} FLOPS, pronounced *teraflops*), or peta-, combined into the shorthand "PFLOPS" (10^{15} FLOPS, pronounced *petaflops*.) "Petascale" supercom-



Top supercomputer speeds: logscale speed *over 60 years*



Distribution of top 500 supercomputers among different countries as of June 2014

puters can process one quadrillion (10^{15}) (1000 trillion) FLOPS. Exascale is computing performance in the exaFLOPS (EFLOPS) range. An EFLOPS is one quintillion (10^{18}) FLOPS (one million TFLOPS).

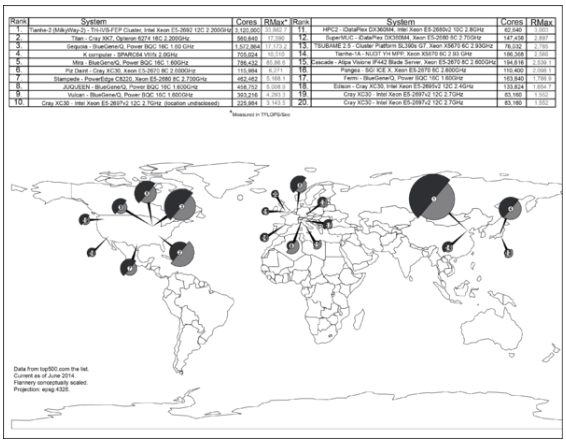
No single number can reflect the overall performance of a computer system, yet the goal of the Linpack benchmark is to approximate how fast the computer solves numerical problems and it is widely used in the industry.[81] The FLOPS measurement is either quoted based on the theoretical floating point performance of a processor (derived from manufacturer's processor specifications and shown as "Rpeak" in the TOP500 lists) which is generally unachievable when running real workloads, or the achievable throughput, derived from the LINPACK benchmarks and shown as "Rmax" in the TOP500 list. The LINPACK benchmark typically performs LU decomposition of a large matrix. The LINPACK performance gives some indication of performance for some real-world problems, but does not necessarily match the processing requirements of many other supercomputer workloads, which for example may require more memory bandwidth, or may require better integer computing performance, or may need a high performance I/O system to achieve high levels of performance.[81]

4.5.3 The TOP500 list

Main article: TOP500

Since 1993, the fastest supercomputers have been ranked on the TOP500 list according to their LINPACK benchmark results. The list does not claim to be unbiased or definitive, but it is a widely cited current definition of the "fastest" supercomputer available at any given time.

This is a recent list of the computers which appeared at the top of the TOP500 list,[82] and the "Peak speed" is given as the "Rmax" rating. For more historical data see History of supercomputing.



Top 20 Supercomputers in the World as of June 2013

4.6 Largest Supercomputer Vendors according to the total Rmax (GFLOPS) operated

Source : TOP500

4.7 Applications of supercomputers

The stages of supercomputer application may be summarized in the following table:

The IBM Blue Gene/P computer has been used to simulate a number of artificial neurons equivalent to approximately one percent of a human cerebral cortex, containing 1.6 billion neurons with approximately 9 trillion connections. The same research group also succeeded in using a supercomputer to simulate a number of artificial neurons equivalent to the entirety of a rat's brain.[89]

Modern-day weather forecasting also relies on supercom-

puters. The National Oceanic and Atmospheric Administration uses supercomputers to crunch hundreds of millions of observations to help make weather forecasts more accurate.[90]

In 2011, the challenges and difficulties in pushing the envelope in supercomputing were underscored by IBM's abandonment of the Blue Waters petascale project.[91]

4.8 Research and development trends

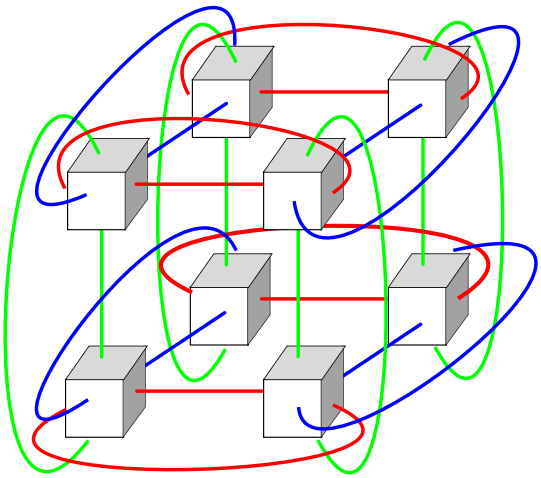


Diagram of a 3-dimensional torus interconnect used by systems such as Blue Gene, Cray XT3, etc.

Given the current speed of progress, industry experts estimate that supercomputers will reach 1 EFLOPS (10^{18}, one quintillion FLOPS) by 2018. In China industry experts estimate machines will start reaching 1,000-petaflop performance by 2018.[92] Using the Intel MIC multi-core processor architecture, which is Intel's response to GPU systems, SGI plans to achieve a 500-fold increase in performance by 2018, in order to achieve one exaFLOPS. Samples of MIC chips with 32 cores, which combine vector processing units with standard CPU, have become available.[93] The Indian government has also stated ambitions for an exaFLOPS-range supercomputer, which they hope to complete by 2017.[94] In November 2014, it was reported that India is working on the Fastest supercomputer ever which is set to work at 132 EFLOPS.[95]

Erik P. DeBenedictis of Sandia National Laboratories theorizes that a zettaFLOPS (10^{21}, one sextillion FLOPS) computer is required to accomplish full weather modeling, which could cover a two-week time span accurately.[96] Such systems might be built around 2030.[97]

4.9 See also

- ACM/IEEE Supercomputing Conference
- Jungle computing
- Nvidia Tesla Personal Supercomputer
- Supercomputing in China
- Supercomputing in Europe
- Supercomputing in India
- Supercomputing in Japan
- Supercomputing in Pakistan
- Ultra Network Technologies
- Testing high-performance computing applications

4.10 Notes and references

[1] "IBM Blue gene announcement". 03.ibm.com. 26 June 2007. Retrieved 9 June 2012.

[2] http://www.top500.org/lists/2014/11/

[3] Hoffman, Allan R.; et al. (1990). *Supercomputers: directions in technology and applications*. National Academies. pp. 35–47. ISBN 0-309-04088-4.

[4] Hill, Mark Donald; Jouppi, Norman Paul; Sohi, Gurindar (1999). *Readings in computer architecture*. pp. 40–49. ISBN 1-55860-539-8.

[5] Prodan, Radu; Fahringer, Thomas (2007). *Grid computing: experiment management, tool integration, and scientific workflows*. pp. 1–4. ISBN 3-540-69261-4.

[6] DesktopGrid

[7] *Performance Modelling and Optimization of Memory Access on Cellular Computer Architecture Cyclops64* K Barner, GR Gao, Z Hu, Lecture Notes in Computer Science, 2005, Volume 3779, Network and Parallel Computing, Pages 132–143

[8] *Analysis and performance results of computing betweenness centrality on IBM Cyclops64* by Guangming Tan, Vugranam C. Sreedhar and Guang R. Gao The Journal of Supercomputing Volume 56, Number 1, 1–24 September 2011

[9] Lemke, Tim (8 May 2013). "NSA Breaks Ground on Massive Computing Center". Retrieved 11 December 2013.

[10] *Hardware software co-design of a multimedia SOC platform* by Sao-Jie Chen, Guang-Huei Lin, Pao-Ann Hsiung, Yu-Hen Hu 2009 ISBN pages 70-72

[11] *The Atlas*, University of Manchester, retrieved 21 September 2010

[12] Lavington, Simon (1998), *A History of Manchester Computers* (2 ed.), Swindon: The British Computer Society, pp. 41–52, ISBN 978-1-902505-01-5

[13] *The Supermen*, Charles Murray, Wiley & Sons, 1997.

[14] *A history of modern computing* by Paul E. Ceruzzi 2003 ISBN 978-0-262-53203-7 page 161

[15] Hannan, Caryn (2008). *Wisconsin Biographical Dictionary*. pp. 83–84. ISBN 1-878592-63-7.

[16] John Impagliazzo, John A. N. Lee (2004). *History of computing in education*. p. 172. ISBN 1-4020-8135-9.

[17] Richard Sisson, Christian K. Zacher (2006). *The American Midwest: an interpretive encyclopedia*. p. 1489. ISBN 0-253-34886-2.

[18] *Readings in computer architecture* by Mark Donald Hill, Norman Paul Jouppi, Gurindar Sohi 1999 ISBN 978-1-55860-539-8 page 41-48

[19] *Milestones in computer science and information technology* by Edwin D. Reilly 2003 ISBN 1-57356-521-0 page 65

[20] *Parallel computing for real-time signal processing and control* by M. O. Tokhi, Mohammad Alamgir Hossain 2003 ISBN 978-1-85233-599-1 pages 201–202

[21] "TOP500 Annual Report 1994". Netlib.org. 1 October 1996. Retrieved 9 June 2012.

[22] N. Hirose and M. Fukuda (1997). *Numerical Wind Tunnel (NWT) and CFD Research at National Aerospace Laboratory*. Proceedings of HPC-Asia '97. IEEE Computer Society. doi:10.1109/HPC.1997.592130.

[23] H. Fujii, Y. Yasuda, H. Akashi, Y. Inagami, M. Koga, O. Ishihara, M. Syazwan, H. Wada, T. Sumimoto, Architecture and performance of the Hitachi SR2201 massively parallel processor system, Proceedings of 11th International Parallel Processing Symposium, April 1997, Pages 233–241.

[24] Y. Iwasaki, The CP-PACS project, Nuclear Physics B – Proceedings Supplements, Volume 60, Issues 1–2, January 1998, Pages 246–254.

[25] A.J. van der Steen, Overview of recent supercomputers, Publication of the NCF, Stichting Nationale Computer Faciliteiten, the Netherlands, January 1997.

[26] *Scalable input/output: achieving system balance* by Daniel A. Reed 2003 ISBN 978-0-262-68142-1 page 182

[27] Steve Nelson (3 October 2014). "ComputerGK.com : Supercomputers".

[28] Xue-June Yang, Xiang-Ke Liao, et al in Journal of Computer Science and Technology. "The TianHe-1A Supercomputer: Its Hardware and Software". 26, Number 3. pp. 344–351.

[29] *The Supermen: Story of Seymour Cray and the Technical Wizards Behind the Supercomputer* by Charles J. Murray 1997 ISBN 0-471-04885-2 pages 133–135

[30] *Parallel Computational Fluid Dyynamics; Recent Advances and Future Directions* edited by Rupak Biswas 2010 ISBN 1-60595-022-X page 401

[31] *Supercomputing Research Advances* by Yongge Huáng 2008 ISBN 1-60456-186-6 pages 313–314

[32] *Computational science – ICCS 2005: 5th international conference* edited by Vaidy S. Sunderam 2005 ISBN 3-540-26043-9 pages 60–67

[33] Knight, Will: "IBM creates world's most powerful computer", *NewScientist.com news service*, June 2007

[34] N. R. Agida et al. (2005). "Blue Gene/L Torus Interconnection Network | IBM Journal of Research and Development" (PDF). *Torus Interconnection Network*. 45, No 2/3 March–May 2005. p. 265.

[35] Mittal et al., "A Survey of Methods for Analyzing and Improving GPU Energy Efficiency", ACM Computing Surveys, 2014.

[36] Prickett, Timothy (31 May 2010). "Top 500 supers – The Dawning of the GPUs". Theregister.co.uk.

[37] Hans Hacker et al in *Facing the Multicore-Challenge: Aspects of New Paradigms and Technologies in Parallel Computing* by Rainer Keller, David Kramer and Jan-Philipp Weiss (2010). *Considering GPGPU for HPC Centers: Is It Worth the Effort?*. pp. 118–121. ISBN 3-642-16232-0.

[38] Damon Poeter (11 October 2011). "Cray's Titan Supercomputer for ORNL Could Be World's Fastest". Pcmag.com.

[39] Feldman, Michael (11 October 2011). "GPUs Will Morph ORNL's Jaguar Into 20-Petaflop Titan". Hpcwire.com.

[40] Timothy Prickett Morgan (11 October 2011). "Oak Ridge changes Jaguar's spots from CPUs to GPUs". Theregister.co.uk.

[41] "The NETL SuperComputer". p. 2.

[42] Condon, J.H. and K.Thompson, "Belle Chess Hardware", In *Advances in Computer Chess 3* (ed.M.R.B.Clarke), Pergamon Press, 1982.

[43] Hsu, Feng-hsiung (2002). "Behind Deep Blue: Building the Computer that Defeated the World Chess Champion". Princeton University Press. ISBN 0-691-09065-3.

[44] C. Donninger, U. Lorenz. The Chess Monster Hydra. Proc. of 14th International Conference on Field-Programmable Logic and Applications (FPL), 2004, Antwerp – Belgium, LNCS 3203, pp. 927 – 932

[45] J Makino and M. Taiji, *Scientific Simulations with Special Purpose Computers: The GRAPE Systems*, Wiley. 1998.

[46] RIKEN press release, *Completion of a one-petaFLOPS computer system for simulation of molecular dynamics*

[47] Electronic Frontier Foundation (1998). *Cracking DES – Secrets of Encryption Research, Wiretap Politics & Chip Design*. Oreilly & Associates Inc. ISBN 1-56592-520-3.

[48] "NVIDIA Tesla GPUs Power World's Fastest Supercomputer" (Press release). Nvidia. 29 October 2010.

[49] Balandin, Alexander A. (October 2009). "Better Computing Through CPU Cooling". Spectrum.ieee.org.

[50] "The Green 500". Green500.org.

[51] "Green 500 list ranks supercomputers". *iTnews Australia*.

[52] Wu-chun Feng (2003). "Making a Case for Efficient Supercomputing | ACM Queue Magazine, Volume 1 Issue 7, 10 January 2003 doi 10.1145/957717.957772" (PDF).

[53] "IBM uncloaks 20 petaflops BlueGene/Q super". The Register. 22 November 2010. Retrieved 25 November 2010.

[54] Prickett, Timothy (15 July 2011). ""The Register": IBM 'Blue Waters' super node washes ashore in August". Theregister.co.uk. Retrieved 9 June 2012.

[55] "HPC Wire 2 July 2010". Hpcwire.com. 2 July 2010. Retrieved 9 June 2012.

[56] Martin LaMonica (10 May 2010). "CNet 10 May 2010". News.cnet.com. Retrieved 9 June 2012.

[57] "Government unveils world's fastest computer". *CNN*. Archived from the original on 10 June 2008. performing 376 million calculations for every watt of electricity used.

[58] "IBM Roadrunner Takes the Gold in the Petaflop Race".

[59] "Top500 Supercomputing List Reveals Computing Trends". IBM... BlueGene/Q system .. setting a record in power efficiency with a value of 1,680 MFLOPS/W, more than twice that of the next best system.

[60] "IBM Research A Clear Winner in Green 500".

[61] "Green 500 list". Green500.org. Retrieved 9 June 2012.

[62] Saed G. Younis. "Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic". 1994. p. 14.

[63] "Hot Topic – the Problem of Cooling Supercomputers".

[64] Anand Lal Shimpi. "Inside the Titan Supercomputer: 299K AMD x86 Cores and 18.6K NVIDIA GPUs". 2012.

[65] Curtis Storlie; Joe Sexton; Scott Pakin; Michael Lang; Brian Reich; William Rust. "Modeling and Predicting Power Consumption of High Performance Computing Jobs". 2014.

[66] *Encyclopedia of Parallel Computing by David Padua 2011 ISBN 0-387-09765-1 pages 426–429*

[67] *Knowing machines: essays on technical change* by Donald MacKenzie 1998 ISBN 0-262-63188-1 page 149-151

[68] *Euro-Par 2004 Parallel Processing: 10th International Euro-Par Conference* 2004, by Marco Danelutto, Marco Vanneschi and Domenico Laforenza ISBN 3-540-22924-8 pages 835

[69] *Euro-Par 2006 Parallel Processing: 12th International Euro-Par Conference*, 2006, by Wolfgang E. Nagel, Wolfgang V. Walter and Wolfgang Lehner ISBN 3-540-37783-2 page

[70] *An Evaluation of the Oak Ridge National Laboratory Cray XT3* by Sadaf R. Alam etal *International Journal of High Performance Computing Applications* February 2008 vol. 22 no. 1 52–80

[71] Open Job Management Architecture for the Blue Gene/L Supercomputer by Yariv Aridor et al. in *Job scheduling strategies for parallel processing* by Dror G. Feitelson 2005 ISBN 978-3-540-31024-2 pages 95–101

[72] "Top500 OS chart". Top500.org. Retrieved 31 October 2010.

[73] "Wide-angle view of the ALMA correlator". *ESO Press Release*. Retrieved 13 February 2013.

[74] "Folding@home: OS Statistics". Stanford University. Retrieved 17 June 2014.

[75] "BOINCstats: BOINC Combined". BOINC. Retrieved 28 May 2011Note this link will give current statistics, not those on the date last accessed.

[76] "BOINCstats: MilkyWay@home". BOINC. Retrieved 28 May 2011Note this link will give current statistics, not those on the date last accessed

[77] "Internet PrimeNet Server Distributed Computing Technology for the Great Internet Mersenne Prime Search". *GIMPS*. Retrieved 6 June 2011.

[78] Kravtsov, Valentin; Carmeli, David; Dubitzky, Werner; Orda, Ariel; Schuster, Assaf; Yoshpa, Benny. "Quasi-opportunistic supercomputing in grids, hot topic paper (2007)". *IEEE International Symposium on High Performance Distributed Computing*. IEEE. Retrieved 4 August 2011.

[79] *The Potential Impact of High-End Capability Computing on Four Illustrative Fields of Science and Engineering* by Committee on the Potential Impact of High-End Computing on Illustrative Fields of Science and Engineering and National Research Council (28 October 2008) ISBN 0-309-12485-9 page 9

[80] Xingfu Wu (1999). *Performance Evaluation, Prediction and Visualization of Parallel Systems*. pp. 114–117. ISBN 0-7923-8462-8.

[81] Dongarra, Jack J.; Luszczek, Piotr; Petitet, Antoine (2003), "The LINPACK Benchmark: past, present and future", *Concurrency and Computation: Practice and Experience* (John Wiley & Sons, Ltd.): 803–820

[82] Intel brochure – 11/91. "Directory page for Top500 lists. Result for each list since June 1993". Top500.org. Retrieved 31 October 2010.

[83] "The Cray-1 Computer System" (PDF). Cray Research, Inc. Retrieved 25 May 2011.

[84] Joshi, Rajani R. (9 June 1998). "A new heuristic algorithm for probabilistic optimization". Department of Mathematics and School of Biomedical Engineering, Indian Institute of Technology Powai, Bombay, India. Retrieved 1 July 2008. (subscription required (help)).

[85] "Abstract for SAMSY – Shielding Analysis Modular System". OECD Nuclear Energy Agency, Issy-les-Moulineaux, France. Retrieved 25 May 2011.

[86] "EFF DES Cracker Source Code". Cosic.esat.kuleuven.be. Retrieved 8 July 2011.

[87] "Disarmament Diplomacy: – DOE Supercomputing & Test Simulation Programme". Acronym.org.uk. 22 August 2000. Retrieved 8 July 2011.

[88] "China's Investment in GPU Supercomputing Begins to Pay Off Big Time!". Blogs.nvidia.com. Retrieved 8 July 2011.

[89] Kaku, Michio. Physics of the Future (New York: Doubleday, 2011), 65.

[90] "Faster Supercomputers Aiding Weather Forecasts". News.nationalgeographic.com. 28 October 2010. Retrieved 8 July 2011.

[91] Washington Post 8 August 2011

[92] Kan Michael (31 October 2012). "China is building a 100-petaflop supercomputer, InfoWorld, 31 October 2012". infoworld.com. Retrieved 31 October 2012.

[93] Agam Shah (20 June 2011). "SGI, Intel plan to speed supercomputers 500 times by 2018, ComputerWorld, 20 June 2011". Computerworld.com. Retrieved 9 June 2012.

[94] Dillow Clay (18 September 2012). "India Aims To Take The "World's Fastest Supercomputer" Crown By 2017, POPSCI, 9 September 2012". popsci.com. Retrieved 31 October 2012.

[95] Prashanth G N (13 November 2014). "India working on building fastest supercomputer". Deccan Herald. Retrieved 28 November 2014.

[96] DeBenedictis, Erik P. (2005). "Reversible logic for supercomputing". *Proceedings of the 2nd conference on Computing frontiers*. pp. 391–402. ISBN 1-59593-019-1.

[97] "IDF: Intel says Moore's Law holds until 2029". *Heise Online*. 4 April 2008.

4.11 External links

- A Tunable, Software-based DRAM Error Detection and Correction Library for HPC
- Detection and Correction of Silent Data Corruption for Large-Scale High-Performance Computing

Chapter 5

Multi-core processor

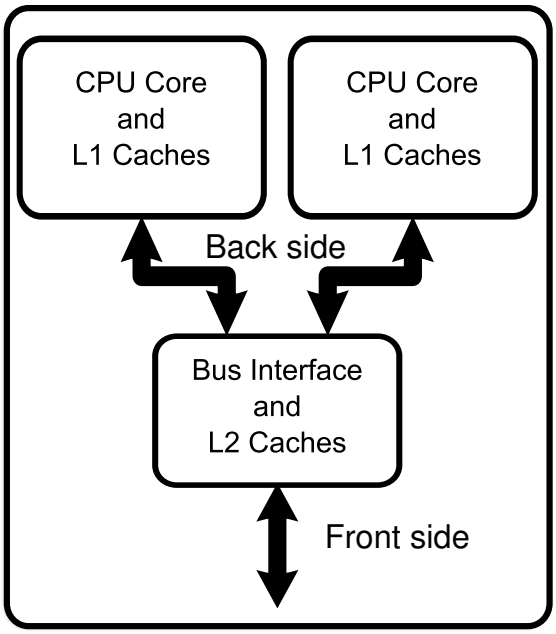
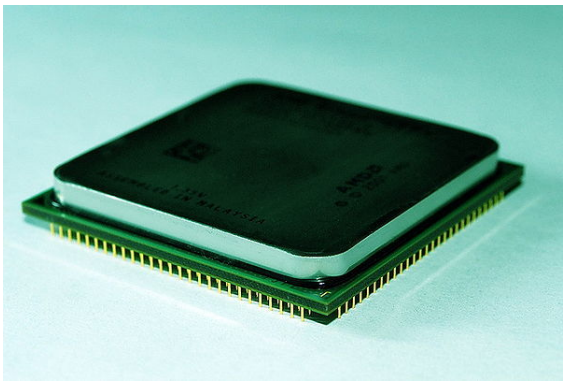


Diagram of a generic dual-core processor, with CPU-local level 1 caches, and a shared, on-die level 2 cache.

An Intel Core 2 Duo E6750 dual-core processor.

A **multi-core processor** is a single computing component with two or more independent actual processing units (called "cores"), which are the units that read and execute program instructions.[1] The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing.[2] Manufacturers typically integrate the cores onto a single integrated circuit die (known as a chip multiprocessor or CMP), or onto multiple dies in a single chip package.



An AMD Athlon X2 6400+ dual-core processor.

Processors were originally developed with only one core. In the mid 1980s Rockwell International manufactured versions of the 6502 with two 6502 cores on one chip as the R65C00, R65C21, and R65C29,[3][4] sharing the chip's pins on alternate clock phases. Other multi-core processors were developed in the early 2000s by Intel, AMD and others.

Multi-core processors may have two cores (dual-core CPUs, for example AMD Phenom II X2 and Intel Core Duo), four cores (quad-core CPUs, for example AMD Phenom II X4, Intel's i5 and i7 processors), six cores (hexa-core CPUs, for example AMD Phenom II X6 and Intel Core i7 Extreme Edition 980X), eight cores (octo-core CPUs, for example Intel Xeon E7-2820 and AMD FX-8350), ten cores (for example, Intel Xeon E7-2850), or more.

A multi-core processor implements multiprocessing in a single physical package. Designers may couple cores in a multi-core device tightly or loosely. For example, cores may or may not share caches, and they may implement message passing or shared memory inter-core communication methods. Common network topologies to interconnect cores include bus, ring, two-dimensional mesh,

and crossbar. *Homogeneous* multi-core systems include only identical cores, heterogeneous multi-core systems have cores that are not identical. Just as with single-processor systems, cores in multi-core systems may implement architectures such as superscalar, VLIW, vector processing, SIMD, or multithreading.

Multi-core processors are widely used across many application domains including general-purpose, embedded, network, digital signal processing (DSP), and graphics.

The improvement in performance gained by the use of a multi-core processor depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can be run in parallel simultaneously on multiple cores; this effect is described by Amdahl's law. In the best case, so-called embarrassingly parallel problems may realize speedup factors near the number of cores, or even more if the problem is split up enough to fit within each core's cache(s), avoiding use of much slower main system memory. Most applications, however, are not accelerated so much unless programmers invest a prohibitive amount of effort in re-factoring the whole problem.[5] The parallelization of software is a significant ongoing topic of research.

5.1 Terminology

The terms *multi-core* and *dual-core* most commonly refer to some sort of central processing unit (CPU), but are sometimes also applied to digital signal processors (DSP) and system-on-a-chip (SoC). The terms are generally used only to refer to multi-core microprocessors that are manufactured on the *same* integrated circuit die; separate microprocessor dies in the same package are generally referred to by another name, such as *multi-chip module*. This article uses the terms "multi-core" and "dual-core" for CPUs manufactured on the *same* integrated circuit, unless otherwise noted.

In contrast to multi-core systems, the term *multi-CPU* refers to multiple physically separate processing-units (which often contain special circuitry to facilitate communication between each other).

The terms *many-core* and *massively multi-core* are sometimes used to describe multi-core architectures with an especially high number of cores (tens or hundreds).[6]

Some systems use many soft microprocessor cores placed on a single FPGA. Each "core" can be considered a "semiconductor intellectual property core" as well as a CPU core.

5.2 Development

While manufacturing technology improves, reducing the size of individual gates, physical limits of semiconductor-based microelectronics have become a major design concern. These physical limitations can cause significant heat dissipation and data synchronization problems. Various other methods are used to improve CPU performance. Some *instruction-level parallelism* (ILP) methods such as superscalar pipelining are suitable for many applications, but are inefficient for others that contain difficult-to-predict code. Many applications are better suited to *thread level parallelism* (TLP) methods, and multiple independent CPUs are commonly used to increase a system's overall TLP. A combination of increased available space (due to refined manufacturing processes) and the demand for increased TLP led to the development of multi-core CPUs.

5.2.1 Commercial incentives

Several business motives drive the development of multi-core architectures. For decades, it was possible to improve performance of a CPU by shrinking the area of the integrated circuit, which drove down the cost per device on the IC. Alternatively, for the same circuit area, more transistors could be used in the design, which increased functionality, especially for CISC architectures. Clock rates also increased by orders of magnitude in the decades of the late 20th century, from several megahertz in the 1980s to several gigahertz in the early 2000s.

As the rate of clock speed improvements slowed, increased use of parallel computing in the form of multi-core processors has been pursued to improve overall processing performance. Multiple cores were used on the same CPU chip, which could then lead to better sales of CPU chips with two or more cores. Intel has produced a 48-core processor for research in cloud computing; each core has an X86 architecture.[7] Intel has loaded Linux on each core.[8]

5.2.2 Technical factors

Since computer manufacturers have long implemented symmetric multiprocessing (SMP) designs using discrete CPUs, the issues regarding implementing multi-core processor architecture and supporting it with software are well known.

Additionally:

- Using a proven processing-core design without architectural changes reduces design risk significantly.
- For general-purpose processors, much of the motivation for multi-core processors comes from greatly

diminished gains in processor performance from increasing the operating frequency. This is due to three primary factors:

1. The *memory wall*; the increasing gap between processor and memory speeds. This, in effect, pushes for cache sizes to be larger in order to mask the latency of memory. This helps only to the extent that memory bandwidth is not the bottleneck in performance.
2. The *ILP wall*; the increasing difficulty of finding enough parallelism in a single instruction stream to keep a high-performance single-core processor busy.
3. The *power wall*; the trend of consuming exponentially increasing power with each factorial increase of operating frequency. This increase can be mitigated by "shrinking" the processor by using smaller traces for the same logic. The *power wall* poses manufacturing, system design and deployment problems that have not been justified in the face of the diminished gains in performance due to the *memory wall* and *ILP wall*.

In order to continue delivering regular performance improvements for general-purpose processors, manufacturers such as Intel and AMD have turned to multi-core designs, sacrificing lower manufacturing-costs for higher performance in some applications and systems. Multi-core architectures are being developed, but so are the alternatives. An especially strong contender for established markets is the further integration of peripheral functions into the chip.

5.2.3 Advantages

The proximity of multiple CPU cores on the same die allows the cache coherency circuitry to operate at a much higher clock-rate than is possible if the signals have to travel off-chip. Combining equivalent CPUs on a single die significantly improves the performance of cache snoop (alternative: Bus snooping) operations. Put simply, this means that signals between different CPUs travel shorter distances, and therefore those signals degrade less. These higher-quality signals allow more data to be sent in a given time period, since individual signals can be shorter and do not need to be repeated as often.

Assuming that the die can physically fit into the package, multi-core CPU designs require much less printed circuit board (PCB) space than do multi-chip SMP designs. Also, a dual-core processor uses slightly less power than two coupled single-core processors, principally because of the decreased power required to drive signals external to the chip. Furthermore, the cores share some circuitry, like the L2 cache and the interface to the front side bus (FSB). In terms of competing technologies for the available silicon die area, multi-core design can make use of proven CPU core library designs and produce a product with lower risk of design error than devising a new wider core-design. Also, adding more cache suffers from diminishing returns.

Multi-core chips also allow higher performance at lower energy. This can be a big factor in mobile devices that operate on batteries. Since each and every core in multi-core is generally more energy-efficient, the chip becomes more efficient than having a single large monolithic core. This allows higher performance with less energy. The challenge of writing parallel code clearly offsets this benefit.[9]

5.2.4 Disadvantages

Maximizing the usage of the computing resources provided by multi-core processors requires adjustments both to the operating system (OS) support and to existing application software. Also, the ability of multi-core processors to increase application performance depends on the use of multiple threads within applications.

Integration of a multi-core chip drives chip production yields down and they are more difficult to manage thermally than lower-density single-chip designs. Intel has partially countered this first problem by creating its quad-core designs by combining two dual-core on a single die with a unified cache, hence any two working dual-core dies can be used, as opposed to producing four cores on a single die and requiring all four to work to produce a quad-core. From an architectural point of view, ultimately, single CPU designs may make better use of the silicon surface area than multiprocessing cores, so a development commitment to this architecture may carry the risk of obsolescence. Finally, raw processing power is not the only constraint on system performance. Two processing cores sharing the same system bus and memory bandwidth limits the real-world performance advantage. It has been claimed that if a single core is close to being memory-bandwidth limited, then going to dual-core might give 30% to 70% improvement; if memory bandwidth is not a problem, then a 90% improvement can be expected; however, Amdahl's law makes this claim dubious.[10] It would be possible for an application that used two CPUs to end up running faster on one dual-core if communication between the CPUs was the limiting factor, which would count as more than 100% improvement.

5.3 Hardware

5.3.1 Trends

The general trend in processor development has moved from dual-, tri-, quad-, hex-, oct-core chips to ones with tens or even thousands of cores. In addition, multi-

core chips mixed with simultaneous multithreading, memory-on-chip, and special-purpose "heterogeneous" cores promise further performance and efficiency gains, especially in processing multimedia, recognition and networking applications. There is also a trend of improving energy-efficiency by focusing on performance-per-watt with advanced fine-grain or ultra fine-grain power management and dynamic voltage and frequency scaling (i.e. laptop computers and portable media players).

5.3.2 Architecture

The composition and balance of the cores in multi-core architecture show great variety. Some architectures use one core design repeated consistently ("homogeneous"), while others use a mixture of different cores, each optimized for a different, "heterogeneous" role.

The article "CPU designers debate multi-core future" by Rick Merritt, EE Times 2008,[11] includes these comments:

> Chuck Moore [...] suggested computers should be more like cellphones, using a variety of specialty cores to run modular software scheduled by a high-level applications programming interface.
>
> [...] Atsushi Hasegawa, a senior chief engineer at Renesas, generally agreed. He suggested the cellphone's use of many specialty cores working in concert is a good model for future multi-core designs.
>
> [...] Anant Agarwal, founder and chief executive of startup Tilera, took the opposing view. He said multi-core chips need to be homogeneous collections of general-purpose cores to keep the software model simple.

5.4 Software effects

An outdated version of an anti-virus application may create a new thread for a scan process, while its GUI thread waits for commands from the user (e.g. cancel the scan). In such cases, a multi-core architecture is of little benefit for the application itself due to the single thread doing all the heavy lifting and the inability to balance the work evenly across multiple cores. Programming truly multithreaded code often requires complex co-ordination of threads and can easily introduce subtle and difficult-to-find bugs due to the interweaving of processing on data shared between threads (thread-safety). Consequently, such code is much more difficult to debug than single-threaded code when it breaks. There has been a perceived lack of motivation for writing consumer-level threaded applications because of the relative rarity of consumer-level demand for maximum use of computer hardware. Although threaded applications incur little additional performance penalty on single-processor machines, the extra overhead of development has been difficult to justify due to the preponderance of single-processor machines. Also, serial tasks like decoding the entropy encoding algorithms used in video codecs are impossible to parallelize because each result generated is used to help create the next result of the entropy decoding algorithm.

Given the increasing emphasis on multi-core chip design, stemming from the grave thermal and power consumption problems posed by any further significant increase in processor clock speeds, the extent to which software can be multithreaded to take advantage of these new chips is likely to be the single greatest constraint on computer performance in the future. If developers are unable to design software to fully exploit the resources provided by multiple cores, then they will ultimately reach an insurmountable performance ceiling.

The telecommunications market had been one of the first that needed a new design of parallel datapath packet processing because there was a very quick adoption of these multiple-core processors for the datapath and the control plane. These MPUs are going to replace[12] the traditional Network Processors that were based on proprietary micro- or pico-code.

Parallel programming techniques can benefit from multiple cores directly. Some existing parallel programming models such as Cilk Plus, OpenMP, OpenHMPP, FastFlow, Skandium, MPI, and Erlang can be used on multi-core platforms. Intel introduced a new abstraction for C++ parallelism called TBB. Other research efforts include the Codeplay Sieve System, Cray's Chapel, Sun's Fortress, and IBM's X10.

Multi-core processing has also affected the ability of modern computational software development. Developers programming in newer languages might find that their modern languages do not support multi-core functionality. This then requires the use of numerical libraries to access code written in languages like C and Fortran, which perform math computations faster than newer languages like C#. Intel's MKL and AMD's ACML are written in these native languages and take advantage of multi-core processing. Balancing the application workload across processors can be problematic, especially if they have different performance characteristics. There are different conceptual models to deal with the problem, for example using a coordination language and program building blocks (programming libraries or higher-order functions). Each block can have a different native implementation for each processor type. Users simply program using these abstractions and an intelligent compiler chooses the best implementation based on the context.[13]

Managing concurrency acquires a central role in developing parallel applications. The basic steps in designing parallel applications are:

Partitioning The partitioning stage of a design is intended to expose opportunities for parallel execution. Hence, the focus is on defining a large number of small tasks in order to yield what is termed a fine-grained decomposition of a problem.

Communication The tasks generated by a partition are intended to execute concurrently but cannot, in general, execute independently. The computation to be performed in one task will typically require data associated with another task. Data must then be transferred between tasks so as to allow computation to proceed. This information flow is specified in the communication phase of a design.

Agglomeration In the third stage, development moves from the abstract toward the concrete. Developers revisit decisions made in the partitioning and communication phases with a view to obtaining an algorithm that will execute efficiently on some class of parallel computer. In particular, developers consider whether it is useful to combine, or agglomerate, tasks identified by the partitioning phase, so as to provide a smaller number of tasks, each of greater size. They also determine whether it is worthwhile to replicate data and computation.

Mapping In the fourth and final stage of the design of parallel algorithms, the developers specify where each task is to execute. This mapping problem does not arise on uniprocessors or on shared-memory computers that provide automatic task scheduling.

On the other hand, on the server side, multi-core processors are ideal because they allow many users to connect to a site simultaneously and have independent threads of execution. This allows for Web servers and application servers that have much better throughput.

5.4.1 Licensing

Vendors may license some software "per processor". This can give rise to ambiguity, because a "processor" may consist either of a single core or of a combination of cores.

- Microsoft has stated that it would treat a socket as a single processor.[14][15]
- Oracle Corporation counts an AMD X2 or an Intel dual-core CPU as a single processor but uses other metrics for other types, especially for processors with more than two cores.[16]

5.5 Embedded applications

Embedded computing operates in an area of processor technology distinct from that of "mainstream" PCs. The same technological drivers towards multi-core apply here too. Indeed, in many cases the application is a "natural" fit for multi-core technologies, if the task can easily be partitioned between the different processors.

In addition, embedded software is typically developed for a specific hardware release, making issues of software portability, legacy code or supporting independent developers less critical than is the case for PC or enterprise computing. As a result, it is easier for developers to adopt new technologies and as a result there is a greater variety of multi-core processing architectures and suppliers.

As of 2010, multi-core network processing devices have become mainstream, with companies such as Freescale Semiconductor, Cavium Networks, Wintegra and Broadcom all manufacturing products with eight processors. For the system developer, a key challenge is how to exploit all the cores in these devices to achieve maximum networking performance at the system level, despite the performance limitations inherent in an SMP operating system. To address this issue, companies such as 6WIND provide portable packet processing software designed so that the networking data plane runs in a fast path environment outside the OS, while retaining full compatibility with standard OS APIs.[17]

In digital signal processing the same trend applies: Texas Instruments has the three-core TMS320C6488 and four-core TMS320C5441, Freescale the four-core MSC8144 and six-core MSC8156 (and both have stated they are working on eight-core successors). Newer entries include the Storm-1 family from Stream Processors, Inc with 40 and 80 general purpose ALUs per chip, all programmable in C as a SIMD engine and Picochip with three-hundred processors on a single die, focused on communication applications.

5.6 Hardware examples

5.6.1 Commercial

- Adapteva Epiphany, a many-core processor architecture which allows up to 4096 processors on-chip, although only a 16 core version has been commercially produced.
- Aeroflex Gaisler LEON3, a multi-core SPARC that also exists in a fault-tolerant version.
- Ageia PhysX, a multi-core physics processing unit.
- Ambric Am2045, a 336-core Massively Parallel Processor Array (MPPA)
- AMD

  - A-Series, dual-, triple-, and quad-core of Accelerated Processor Units (APU).
  - Athlon 64, Athlon 64 FX and Athlon 64 X2 family, dual-core desktop processors.
  - Athlon II, dual-, triple-, and quad-core desktop processors.
  - FX-Series, quad-, 6-, and 8-core desktop processors.
  - Opteron, dual-, quad-, 6-, 8-, 12-, and 16-core server/workstation processors.
  - Phenom, dual-, triple-, and quad-core processors.
  - Phenom II, dual-, triple-, quad-, and 6-core desktop processors.
  - Sempron X2, dual-core entry level processors.
  - Turion 64 X2, dual-core laptop processors.
  - Radeon and FireStream multi-core GPU/GPGPU (10 cores, 16 5-issue wide superscalar stream processors per core)
- Analog Devices Blackfin BF561, a symmetrical dual-core processor
- ARM MPCore is a fully synthesizable multi-core container for ARM11 MPCore and ARM Cortex-A9 MPCore processor cores, intended for high-performance embedded and entertainment applications.
- ASOCS ModemX, up to 128 cores, wireless applications.
- Azul Systems
  - Vega 1, a 24-core processor, released in 2005.
  - Vega 2, a 48-core processor, released in 2006.
  - Vega 3, a 54-core processor, released in 2008.
- Broadcom SiByte SB1250, SB1255 and SB1455.
- ClearSpeed
  - CSX700, 192-core processor, released in 2008 (32/64-bit floating point; Integer ALU)
- Cradle Technologies CT3400 and CT3600, both multi-core DSPs.
- Cavium Networks Octeon, a 32-core MIPS MPU.
- Freescale Semiconductor QorIQ series processors, up to 8 cores, Power Architecture MPU.
- Hewlett-Packard PA-8800 and PA-8900, dual core PA-RISC processors.
- IBM
  - POWER4, a dual-core processor, released in 2001.
  - POWER5, a dual-core processor, released in 2004.
  - POWER6, a dual-core processor, released in 2007.
  - POWER7, a 4,6,8-core processor, released in 2010.
  - POWER8, a 12-core processor, released in 2013.
  - PowerPC 970MP, a dual-core processor, used in the Apple Power Mac G5.
  - Xenon, a triple-core, SMT-capable, PowerPC microprocessor used in the Microsoft Xbox 360 game console.
- Kalray
  - MPPA-256, 256-core processor, released 2012 (256 usable VLIW cores, Network-on-Chip (NoC), 32/64-bit IEEE 754 compliant FPU)
- Sony/IBM/Toshiba's Cell processor, a nine-core processor with one general purpose PowerPC core and eight specialized SPUs (Synergystic Processing Unit) optimized for vector operations used in the Sony PlayStation 3
- Infineon Danube, a dual-core, MIPS-based, home gateway processor.
- Intel
  - Atom, single and dual-core processors for netbook systems.
  - Celeron Dual-Core, the first dual-core processor for the budget/entry-level market.
  - Core Duo, a dual-core processor.
  - Core 2 Duo, a dual-core processor.
  - Core 2 Quad, 2 dual-core dies packaged in a multi-chip module.
  - Core i3, Core i5 and Core i7, a family of multi-core processors, the successor of the Core 2 Duo and the Core 2 Quad.
  - Itanium 2, a dual-core processor.
  - Pentium D, 2 single-core dies packaged in a multi-chip module.
  - Pentium Extreme Edition, 2 single-core dies packaged in a multi-chip module.
  - Pentium Dual-Core, a dual-core processor.
  - Teraflops Research Chip (Polaris), a 3.16 GHz, 80-core processor prototype, which the company originally stated would be released by 2011.[18]
  - Xeon dual-, quad-, 6-, 8-, 10- and 15-core processors.[19]

- Xeon Phi 57-core, 60-core and 61-core processors.

- IntellaSys
  - SEAforth 40C18, a 40-core processor[20]
  - SEAforth24, a 24-core processor designed by Charles H. Moore

- NetLogic Microsystems
  - XLP, a 32-core, quad-threaded MIPS64 processor
  - XLR, an eight-core, quad-threaded MIPS64 processor
  - XLS, an eight-core, quad-threaded MIPS64 processor

- Nvidia
  - GeForce 9 multi-core GPU (8 cores, 16 scalar stream processors per core)
  - GeForce 200 multi-core GPU (10 cores, 24 scalar stream processors per core)
  - Tesla multi-core GPGPU (10 cores, 24 scalar stream processors per core)

- Parallax Propeller P8X32, an eight-core microcontroller.

- picoChip PC200 series 200–300 cores per device for DSP & wireless

- Plurality HAL series tightly coupled 16-256 cores, L1 shared memory, hardware synchronized processor.

- Rapport Kilocore KC256, a 257-core microcontroller with a PowerPC core and 256 8-bit "processing elements".

- SiCortex "SiCortex node" has six MIPS64 cores on a single chip.

- Sun Microsystems
  - MAJC 5200, two-core VLIW processor
  - UltraSPARC IV and UltraSPARC IV+, dual-core processors.
  - UltraSPARC T1, an eight-core, 32-thread processor.
  - UltraSPARC T2, an eight-core, 64-concurrent-thread processor.
  - UltraSPARC T3, a sixteen-core, 128-concurrent-thread processor.
  - SPARC T4, an eight-core, 64-concurrent-thread processor.
  - SPARC T5, a sixteen-core, 128-concurrent-thread processor.

- Texas Instruments
  - TMS320C80 MVP, a five-core multimedia video processor.
  - TMS320TMS320C66, 2,4,8 core dsp.

- Tilera
  - TILE64, a 64-core 32-bit processor
  - TILE-Gx, a 72-core 64-bit processor

- XMOS Software Defined Silicon quad-core XS1-G4

5.6.2 Free

- OpenSPARC

5.6.3 Academic

- MIT, 16-core RAW processor
- University of California, Davis, Asynchronous array of simple processors (AsAP)
  - 36-core 610 MHz AsAP
  - 167-core 1.2 GHz AsAP2
- University of Washington, Wavescalar processor
- University of Texas, Austin, TRIPS processor
- Linköping University, Sweden, ePUMA processor

5.7 Benchmarks

The research and development of multicore processors often compares many options, and benchmarks are developed to help such evaluations. Existing benchmarks include SPLASH-2, PARSEC, and COSMIC for heterogeneous systems.[21]

5.8 Notes

1. ^ Digital signal processors (DSPs) have used multi-core architectures for much longer than high-end general-purpose processors. A typical example of a DSP-specific implementation would be a combination of a RISC CPU and a DSP MPU. This allows for the design of products that require a general-purpose processor for user interfaces and a DSP for real-time data processing; this type of design is common in mobile phones. In other applications, a growing number of companies have developed multi-core DSPs with very large numbers of processors.

2. ^ Two types of operating systems are able to use a dual-CPU multiprocessor: partitioned multiprocessing and symmetric multiprocessing (SMP). In a partitioned architecture, each CPU boots into separate segments of physical memory and operate independently; in an SMP OS, processors work in a shared space, executing threads within the OS independently.

5.9 See also

- Race condition
- Multicore Association
- Hyper-threading
- Multitasking
- PureMVC MultiCore – a modular programming framework
- XMTC
- Parallel Random Access Machine
- Partitioned global address space (PGAS)
- Thread
- CPU shielding
- GPGPU
- CUDA
- OpenCL (Open Computing Language) – a framework for heterogeneous execution
- Ateji PX – an extension of the Java language for parallelism
- BMDFM (Binary Modular Dataflow Machine) – Multi-core Runtime Environment

5.10 References

[1] Margaret Rouse (March 27, 2007). "Definition: multi-core processor". TechTarget. Retrieved March 6, 2013.

[2] CSA Organization

[3] "Rockwell R65C00/21 Dual CMOS Microcomputer and R65C29 Dual CMOS Microprocessor". Rockwell International. October 1984.

[4] "Rockwell 1985 Data Book". Rockwell International Semiconductor Products Division. January 1985.

[5] Aater Suleman (May 20, 2011). "What makes parallel programming hard?". FutureChips. Retrieved March 6, 2013.

[6] Programming Many-Core Chips. By András Vajda, page 3

[7] Ryan Shrout (December 2, 2009). "Intel Shows 48-core x86 Processor as Single-chip Cloud Computer". Retrieved March 6, 2013.

[8] "Intel unveils 48-core cloud computing silicon chip". BBC. December 3, 2009. Retrieved March 6, 2013.

[9] Aater Suleman (May 19, 2011). "Q & A: Do multicores save energy? Not really.". Retrieved March 6, 2013.

[10] Ni, Jun. "Multi-core Programming for Medical Imaging". Retrieved 17 February 2013.

[11] Rick Merritt (February 6, 2008). "CPU designers debate multi-core future". EE Times. Retrieved March 6, 2013.

[12] Multicore packet processing Forum

[13] John Darlinton, Moustafa Ghanem, Yike Guo, Hing Wing To (1996), "Guided Resource Organisation in Heterogeneous Parallel Computing", *Journal of High Performance Computing* **4** (1): 13–23

[14] Multicore Processor Licensing

[15] Compare: "Multi-Core Processor Licensing". *download.microsoft.com*. Microsoft Corporation. 2004-10-19. p. 1. Retrieved 2015-03-05. On October 19, 2004, Microsoft announced that our server software that is currently licensed on a per-processor model will continue to be licensed on a per-processor, and not on a per-core, model.

[16] Compare: "The Licensing Of Oracle Technology Products". OMT-CO Operations Management Technology Consulting GmbH. Retrieved 2014-03-04.

[17] Maximizing network stack performance

[18] 80-core prototype from Intel

[19] 15 core Xeon

[20] "40-core processor with Forth-based IDE tools unveiled"

[21] "COSMIC Heterogeneous Multiprocessor Benchmark Suite"

5.11 External links

- What Is A Processor Core?
- Embedded moves to multicore
- Multicore News blog
- IEEE: Multicore Is Bad News For Supercomputers

Chapter 6

Graphics processing unit

Not to be confused with Graphics card.
"GPU" redirects here. For other uses, see GPU (disambiguation).

A **graphics processing unit** (**GPU**), also occasionally called **visual processing unit** (**VPU**), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing, and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. In a personal computer, a GPU can be present on a video card, or it can be on the motherboard or—in certain CPUs—on the CPU die.[1]

GeForce 6600GT (NV43) GPU

The term GPU was popularized by Nvidia in 1999, who marketed the GeForce 256 as "the world's first 'GPU', or Graphics Processing Unit, a single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that are capable of processing a minimum of 10 million polygons per second". Rival ATI Technologies coined the term visual processing unit or VPU with the release of the Radeon 9700 in 2002.

6.1 History

Arcade system boards have been using specialized graphics chips since the 1970s. Fujitsu's MB14241 video shifter was used to accelerate the drawing of sprite graphics for various 1970s arcade games from Taito and Midway, such as *Gun Fight* (1975), *Sea Wolf* (1976) and *Space Invaders* (1978).[2][3][4] The Namco Galaxian arcade system in 1979 used specialized graphics hardware supporting RGB color, multi-colored sprites and tilemap backgrounds.[5] The Galaxian hardware was widely used during the golden age of arcade video games, by game companies such as Namco, Centuri, Gremlin, Irem, Konami, Midway, Nichibutsu, Sega and Taito.[6][7] In the home video game console market, the Atari 2600 in 1977 used a video shifter called the Television Interface Adaptor.

6.1.1 1980s

See also: Video Display Controller, List of home computers by video hardware and Sprite (computer graphics)

In 1985, the Commodore Amiga featured a GPU advanced for a personal computer at the time. It supported line draw, area fill, and included a type of stream processor called a blitter which accelerated the movement, manipulation and combination of multiple arbitrary bitmaps. Also included was a coprocessor with its own (primitive) instruction set capable of directly invoking a sequence of graphics operations without CPU intervention. Prior to this and for quite some time after, many other personal computer systems instead used their main, general-purpose CPU to handle almost every aspect of drawing the display, short of generating the final video signal.

In 1986, Texas Instruments released the TMS34010, the first microprocessor with on-chip graphics capabilities. It could run general-purpose code, but it had a very graphics-oriented instruction set. In 1990-1991, this chip

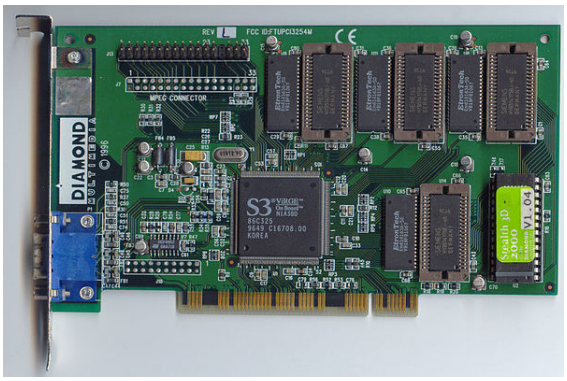
would become the basis of the Texas Instruments Graphics Architecture ("TIGA") Windows accelerator cards.

In 1987, the IBM 8514 graphics system was released as one of the first video cards for IBM PC compatibles to implement fixed-function 2D primitives in electronic hardware. The same year, Sharp released the X68000, which used a custom graphics chipset[8] that was powerful for a home computer at the time, with a 65,536 color palette and hardware support for sprites, scrolling and multiple playfields,[9] eventually serving as a development machine for Capcom's CP System arcade board. Fujitsu later competed with the FM Towns computer, released in 1989 with support for a full 16,777,216 color palette.[10]

In 1988, the first dedicated polygonal 3D graphics boards were introduced in arcades with the Namco System 21[11] and Taito Air System.[12]

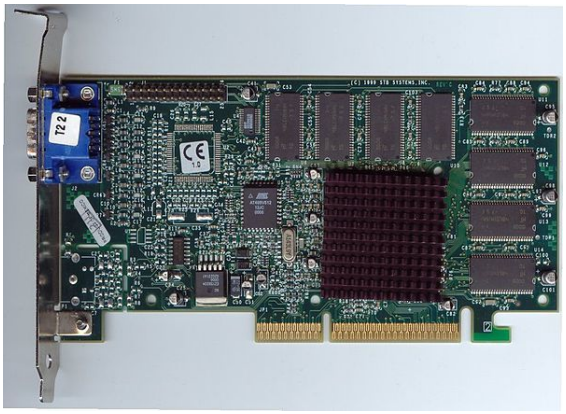
6.1.2 1990s

Tseng Labs ET4000/W32p



S3 Graphics ViRGE

In 1991, S3 Graphics introduced the *S3 86C911*, which its designers named after the Porsche 911 as an implication of the performance increase it promised. The 86C911 spawned a host of imitators: by 1995, all major PC graphics chip makers had added 2D acceleration support to their chips. By this time, fixed-function *Windows accelerators* had surpassed expensive general-purpose graphics coprocessors in Windows performance, and these coprocessors faded away from the PC market.



Voodoo3 2000 AGP card

Throughout the 1990s, 2D GUI acceleration continued to evolve. As manufacturing capabilities improved, so did the level of integration of graphics chips. Additional application programming interfaces (APIs) arrived for a variety of tasks, such as Microsoft's WinG graphics library for Windows 3.x, and their later DirectDraw interface for hardware acceleration of 2D games within Windows 95 and later.

In the early- and mid-1990s, CPU-assisted real-time 3D graphics were becoming increasingly common in arcade, computer and console games, which led to an increasing public demand for hardware-accelerated 3D graphics. Early examples of mass-marketed 3D graphics hardware can be found in arcade system boards such as the Sega Model 1, Namco System 22, and Sega Model 2, and the fifth-generation video game consoles such as the Saturn, PlayStation and Nintendo 64. Arcade systems such as the Sega Model 2 and Namco Magic Edge Hornet Simulator were capable of hardware T&L (transform, clipping, and lighting) years before appearing in consumer graphics cards.[13][14] Fujitsu, which worked on the Sega Model 2 arcade system,[15] began working on integrating T&L into a single LSI solution for use in home computers in 1995.[16][17][18]

In the PC world, notable failed first tries for low-cost 3D graphics chips were the S3 *ViRGE*, ATI *Rage*, and Matrox *Mystique*. These chips were essentially previous-generation 2D accelerators with 3D features bolted on. Many were even pin-compatible with the earlier-generation chips for ease of implementation and minimal cost. Initially, performance 3D graphics were possible only with discrete boards dedicated to accelerating 3D functions (and lacking 2D GUI acceleration entirely) such as the PowerVR and the 3dfx *Voodoo*. However, as manufacturing technology continued to progress, video, 2D GUI acceleration and 3D functionality were all integrated into one chip. Rendition's *Verite* chipsets were among the first to do this well enough to be worthy of note. In 1997, Rendition went a step further by collaborating with Hercules and Fujitsu on a "Thriller Conspiracy" project which combined a Fujitsu FXG-1 Pino-

lite geometry processor with a Vérité V2200 core to create a graphics card with a full T&L engine years before Nvidia's GeForce 256. This card, designed to reduce the load placed upon the system's CPU, never made it to market.

OpenGL appeared in the early '90s as a professional graphics API, but originally suffered from performance issues which allowed the Glide API to step in and become a dominant force on the PC in the late '90s.[19] However, these issues were quickly overcome and the Glide API fell by the wayside. Software implementations of OpenGL were common during this time, although the influence of OpenGL eventually led to widespread hardware support. Over time, a parity emerged between features offered in hardware and those offered in OpenGL. DirectX became popular among Windows game developers during the late 90s. Unlike OpenGL, Microsoft insisted on providing strict one-to-one support of hardware. The approach made DirectX less popular as a standalone graphics API initially, since many GPUs provided their own specific features, which existing OpenGL applications were already able to benefit from, leaving DirectX often one generation behind. (See: Comparison of OpenGL and Direct3D.)

Over time, Microsoft began to work more closely with hardware developers, and started to target the releases of DirectX to coincide with those of the supporting graphics hardware. Direct3D 5.0 was the first version of the burgeoning API to gain widespread adoption in the gaming market, and it competed directly with many more-hardware-specific, often proprietary graphics libraries, while OpenGL maintained a strong following. Direct3D 7.0 introduced support for hardware-accelerated transform and lighting (T&L) for Direct3D, while OpenGL had this capability already exposed from its inception. 3D accelerator cards moved beyond being just simple rasterizers to add another significant hardware stage to the 3D rendering pipeline. The Nvidia *GeForce 256* (also known as NV10) was the first consumer-level card released on the market with hardware-accelerated T&L, while professional 3D cards already had this capability. Hardware transform and lighting, both already existing features of OpenGL, came to consumer-level hardware in the '90s and set the precedent for later pixel shader and vertex shader units which were far more flexible and programmable.

6.1.3 2000 to 2006

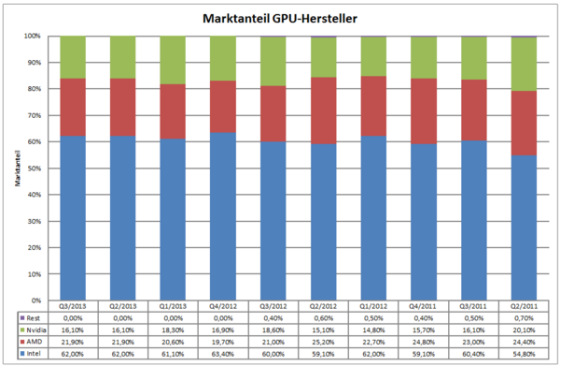
With the advent of the OpenGL API and similar functionality in DirectX, GPUs added shading to their capabilities. Each pixel could now be processed by a short program that could include additional image textures as inputs, and each geometric vertex could likewise be processed by a short program before it was projected onto the screen. Nvidia was first to produce a chip capable of programmable shading, the *GeForce 3* (code named NV20). By October 2002, with the introduction of the ATI *Radeon 9700* (also known as R300), the world's first Direct3D 9.0 accelerator, pixel and vertex shaders could implement looping and lengthy floating point math, and in general were quickly becoming as flexible as CPUs, and orders of magnitude faster for image-array operations. Pixel shading is often used for things like bump mapping, which adds texture, to make an object look shiny, dull, rough, or even round or extruded.[20]

6.1.4 2006 to present

With the introduction of the GeForce 8 series, which was produced by Nvidia, and then new generic stream processing unit GPUs became a more generalized computing device. Today, parallel GPUs have begun making computational inroads against the CPU, and a subfield of research, dubbed GPU Computing or GPGPU for *General Purpose Computing on GPU*, has found its way into fields as diverse as machine learning,[21] oil exploration, scientific image processing, linear algebra,[22] statistics,[23] 3D reconstruction and even stock options pricing determination. Over the years, the energy consumption of GPUs has increased and to manage it, several techniques have been proposed.[24]

Nvidia's CUDA platform was the earliest widely adopted programming model for GPU computing. More recently OpenCL has become broadly supported. OpenCL is an open standard defined by the Khronos Group which allows for the development of code for both GPUs and CPUs with an emphasis on portability.[25] OpenCL solutions are supported by Intel, AMD, Nvidia, and ARM, and according to a recent report by Evan's Data, OpenCL is the GPGPU development platform most widely used by developers in both the US and Asia Pacific.

6.1.5 GPU companies



GPU manufacturers market share

Many companies have produced GPUs under a number of brand names. In 2009, Intel, Nvidia and AMD/ATI were the market share leaders, with 49.4%, 27.8% and

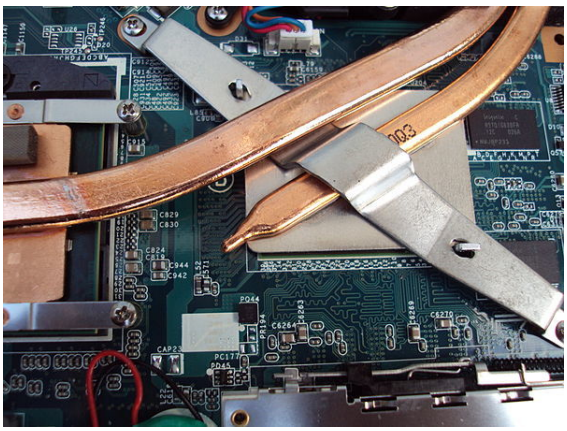
20.6% market share respectively. However, those numbers include Intel's integrated graphics solutions as GPUs. Not counting those numbers, Nvidia and ATI control nearly 100% of the market as of 2008.[26] In addition, S3 Graphics[27] (owned by VIA Technologies) and Matrox[28] produce GPUs.

6.2 Computational functions

Modern GPUs use most of their transistors to do calculations related to 3D computer graphics. They were initially used to accelerate the memory-intensive work of texture mapping and rendering polygons, later adding units to accelerate geometric calculations such as the rotation and translation of vertices into different coordinate systems. Recent developments in GPUs include support for programmable shaders which can manipulate vertices and textures with many of the same operations supported by CPUs, oversampling and interpolation techniques to reduce aliasing, and very high-precision color spaces. Because most of these computations involve matrix and vector operations, engineers and scientists have increasingly studied the use of GPUs for non-graphical calculations.

In addition to the 3D hardware, today's GPUs include basic 2D acceleration and framebuffer capabilities (usually with a VGA compatibility mode). Newer cards like AMD/ATI HD5000-HD7000 even lack 2D acceleration; it has to be emulated by 3D hardware.

6.2.1 GPU accelerated video decoding



The ATI HD5470 GPU (above) features UVD 2.1 which enables it to decode AVC and VC-1 video formats

Most GPUs made since 1995 support the YUV color space and hardware overlays, important for digital video playback, and many GPUs made since 2000 also support MPEG primitives such as motion compensation and iDCT. This process of hardware accelerated video decoding, where portions of the video decoding process and video post-processing are offloaded to the GPU hardware, is commonly referred to as "GPU accelerated video decoding", "GPU assisted video decoding", "GPU hardware accelerated video decoding" or "GPU hardware assisted video decoding".

More recent graphics cards even decode high-definition video on the card, offloading the central processing unit. The most common APIs for GPU accelerated video decoding are DxVA for Microsoft Windows operating system and VDPAU, VAAPI, XvMC, and XvBA for Linux-based and UNIX-like operating systems. All except XvMC are capable of decoding videos encoded with MPEG-1, MPEG-2, MPEG-4 ASP (MPEG-4 Part 2), MPEG-4 AVC (H.264 / DivX 6), VC-1, WMV3/WMV9, Xvid / OpenDivX (DivX 4), and DivX 5 codecs, while XvMC is only capable of decoding MPEG-1 and MPEG-2.

Video decoding processes that can be accelerated

The video decoding processes that can be accelerated by today's modern GPU hardware are:

- Motion compensation (mocomp)
- Inverse discrete cosine transform (iDCT)
  - Inverse telecine 3:2 and 2:2 pull-down correction
- Inverse modified discrete cosine transform (iMDCT)
- In-loop deblocking filter
- Intra-frame prediction
- Inverse quantization (IQ)
- Variable-length decoding (VLD), more commonly known as slice-level acceleration
- Spatial-temporal deinterlacing and automatic interlace/progressive source detection
- Bitstream processing (Context-adaptive variable-length coding/Context-adaptive binary arithmetic coding) and perfect pixel positioning.

6.3 GPU forms

6.3.1 Dedicated graphics cards

Main article: Video card

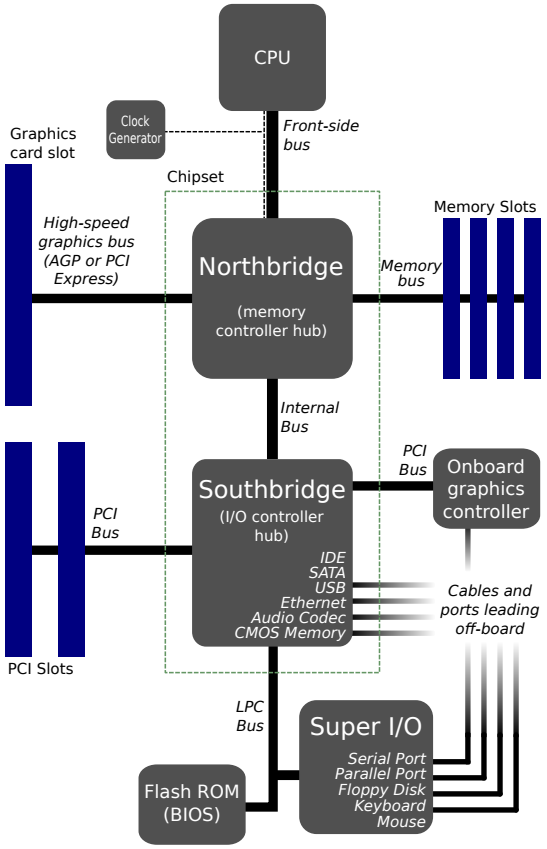
The GPUs of the most powerful class typically interface with the motherboard by means of an expansion

slot such as PCI Express (PCIe) or Accelerated Graphics Port (AGP) and can usually be replaced or upgraded with relative ease, assuming the motherboard is capable of supporting the upgrade. A few graphics cards still use Peripheral Component Interconnect (PCI) slots, but their bandwidth is so limited that they are generally used only when a PCIe or AGP slot is not available.

A dedicated GPU is not necessarily removable, nor does it necessarily interface with the motherboard in a standard fashion. The term "dedicated" refers to the fact that dedicated graphics cards have RAM that is dedicated to the card's use, not to the fact that *most* dedicated GPUs are removable. Dedicated GPUs for portable computers are most commonly interfaced through a non-standard and often proprietary slot due to size and weight constraints. Such ports may still be considered PCIe or AGP in terms of their logical host interface, even if they are not physically interchangeable with their counterparts.

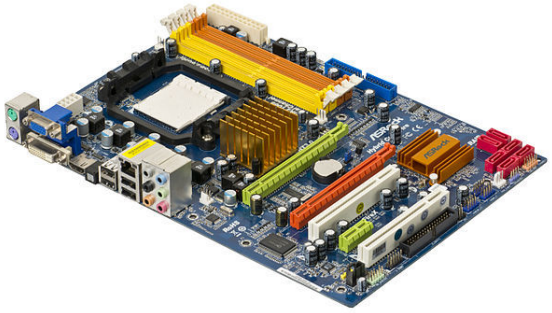
Technologies such as SLI by Nvidia and CrossFire by AMD allow multiple GPUs to draw images simultaneously for a single screen, increasing the processing power available for graphics.

6.3.2 Integrated graphics solutions



Layout

Integrated graphics solutions, *shared graphics solutions*, or *integrated graphics processors* (IGP) utilize a portion of a computer's system RAM rather than dedicated graphics memory. IGPs can be integrated onto the motherboard as part of the chipset, or within the same die as CPU (like AMD APU or Intel HD Graphics). Some of AMD's IGPs use dedicated sideport memory on certain motherboards. Computers with integrated graphics account for 90% of all PC shipments.[29] These solutions are less costly to implement than dedicated graphics solutions, but tend to be less capable. Historically, integrated solutions were often considered unfit to play 3D games or run graphically intensive programs but could run less intensive programs such as Adobe Flash. Examples of such IGPs would be offerings from SiS and VIA circa 2004.[30] However, modern integrated graphics processors such as AMD Accelerated Processing Unit and Intel HD Graphics are more than capable of handling 2D graphics or low stress 3D graphics.



A motherboard with integrated graphics, which has HDMI, VGA and DVI outs.

As a GPU is extremely memory intensive, an integrated solution may find itself competing for the already relatively slow system RAM with the CPU, as it has minimal or no dedicated video memory. IGPs can have up to 29.856 GB/s of memory bandwidth from system RAM, however graphics cards can enjoy up to 264 GB/s of bandwidth between its RAM and GPU core. This bandwidh is what is referred to as the memory bus and can be performance limiting. Older integrated graphics chipsets lacked hardware transform and lighting, but newer ones include it.[31][32]

6.3.3 Hybrid solutions

This newer class of GPUs competes with integrated graphics in the low-end desktop and notebook markets. The most common implementations of this are ATI's HyperMemory and Nvidia's TurboCache.

Hybrid graphics cards are somewhat more expensive than integrated graphics, but much less expensive than dedicated graphics cards. These share memory with the system and have a small dedicated memory cache, to make up for the high latency of the system RAM. Technologies within PCI Express can make this possible. While these

solutions are sometimes advertised as having as much as 768MB of RAM, this refers to how much can be shared with the system memory.

6.3.4 Stream Processing and General Purpose GPUs (GPGPU)

Main articles: GPGPU and Stream processing

It is becoming increasingly common to use a general purpose graphics processing unit as a modified form of stream processor. This concept turns the massive computational power of a modern graphics accelerator's shader pipeline into general-purpose computing power, as opposed to being hard wired solely to do graphical operations. In certain applications requiring massive vector operations, this can yield several orders of magnitude higher performance than a conventional CPU. The two largest discrete (see "Dedicated graphics cards" above) GPU designers, ATI and Nvidia, are beginning to pursue this approach with an array of applications. Both Nvidia and ATI have teamed with Stanford University to create a GPU-based client for the Folding@home distributed computing project, for protein folding calculations. In certain circumstances the GPU calculates forty times faster than the conventional CPUs traditionally used by such applications.[33][34]

GPGPU can be used for many types of embarrassingly parallel tasks including ray tracing. They are generally suited to high-throughput type computations that exhibit data-parallelism to exploit the wide vector width SIMD architecture of the GPU.

Furthermore, GPU-based high performance computers are starting to play a significant role in large-scale modelling. Three of the 10 most powerful supercomputers in the world take advantage of GPU acceleration.[35]

NVIDIA cards support API extensions to the C programming language such as CUDA ("Compute Unified Device Architecture") and OpenCL. CUDA is specifically for NVIDIA GPUs whilst OpenCL is designed to work across a multitude of architectures including GPU, CPU and DSP (using vendor specific SDKs). These technologies allow specified functions (kernels) from a normal C program to run on the GPU's stream processors. This makes C programs capable of taking advantage of a GPU's ability to operate on large matrices in parallel, while still making use of the CPU when appropriate. CUDA is also the first API to allow CPU-based applications to directly access the resources of a GPU for more general purpose computing without the limitations of using a graphics API.

Since 2005 there has been interest in using the performance offered by GPUs for evolutionary computation in general, and for accelerating the fitness evaluation in genetic programming in particular. Most approaches compile linear or tree programs on the host PC and transfer the executable to the GPU to be run. Typically the performance advantage is only obtained by running the single active program simultaneously on many example problems in parallel, using the GPU's SIMD architecture.[36][37] However, substantial acceleration can also be obtained by not compiling the programs, and instead transferring them to the GPU, to be interpreted there.[38][39] Acceleration can then be obtained by either interpreting multiple programs simultaneously, simultaneously running multiple example problems, or combinations of both. A modern GPU (*e.g.* 8800 GTX or later) can readily simultaneously interpret hundreds of thousands of very small programs.

6.3.5 External GPU (eGPU)

An external GPU is a graphics processor located outside of the housing of the computer. External Graphics Processors are often used with laptop computers. Laptops might have a substantial amount of RAM and a sufficiently powerful Central Processing Unit(CPU), but often lack a powerful graphics processor (and instead have a less powerful, but energy efficient on-board graphics chip). On-board graphics chips are often not powerful enough for playing the latest games, or for other tasks (video editing, ...).

Therefore it is desirable to be able to attach to some external PCIe bus of a notebook. That may be an x1 2.0 5Gbit/s expresscard or mPCIe (wifi) port or a 10Gbit/s/16Gbit/s Thunderbolt1/Thunderbolt2 port. Those ports being only available on certain candidate notebook systems.[40][41]

External GPU's have had little official vendor support. Promising solutions such as Silverstone T004 (aka ASUS XG2)[42] and MSI GUS-II[43] were never released to the general public. MSI's Gamedock [44] promising to deliver a full x16 external PCIe bus to a purpose built compact 13" MSI GS30 notebook. Lenovo and Magma partnering in Sep-2014 to deliver official Thunderbolt eGPU support.[45]

This has not stopped enthusiasts from creating their own DIY eGPU solutions.[46][47] expresscard/mPCIe eGPU adapters/enclosures are usually acquired from BPlus (PE4C, PE4L, PE4C),[48] or EXP GDC.[49] native Thunderbolt eGPU adaptere/enclosures acquired from One Stop Systems,[50] AKiTiO,[51] Sonnet (often rebadge as Other World Computing — OWC) and FirmTek.

6.4 Sales

In 2013, 438.3 million GPUs were shipped globally and the forecast for 2014 was 414.2 million.[52]

6.5 See also

- Brute force attack
- Computer graphics
- Computer hardware
- Computer monitor
- Central processing unit
- Physics processing unit (PPU)
- Ray tracing hardware
- Video card
- Video Display Controller
- Video game console
- Virtualized GPU

6.5.1 Hardware

- Comparison of AMD graphics processing units
- Comparison of Nvidia graphics processing units
- Comparison of Intel graphics processing units
- Intel GMA
- Larrabee
- Nvidia PureVideo - the bit-stream technology from Nvidia used in their graphics chips to accelerate video decoding on hardware GPU with DXVA.
- UVD (Unified Video Decoder) - is the video decoding bit-stream technology from ATI Technologies to support hardware (GPU) decode with DXVA.

6.5.2 APIs

- OpenGL API
- DirectX Video Acceleration (DxVA) API for Microsoft Windows operating-system.
- Mantle (API)
- Video Acceleration API (VA API)
- VDPAU (Video Decode and Presentation API for Unix)
- X-Video Bitstream Acceleration (XvBA), the X11 equivalent of DXVA for MPEG-2, H.264, and VC-1
- X-Video Motion Compensation, the X11 equivalent for MPEG-2 video codec only

6.5.3 Applications

- GPU cluster
- Mathematica includes built-in support for CUDA and OpenCL GPU execution
- MATLAB acceleration using the Parallel Computing Toolbox and MATLAB Distributed Computing Server,[53] as well as 3rd party packages like Jacket.
- Molecular modeling on GPU
- Deeplearning4j, open-source, distributed deep learning for Java. Machine vision and textual topic modelling toolkit.

6.6 References

[1] Denny Atkin. "Computer Shopper: The Right GPU for You". Retrieved 2007-05-15.

[2] "mame/8080bw.c at master · mamedev/mame · GitHub". *GitHub*.

[3] "mame/mw8080bw.c at master · mamedev/mame · GitHub". *GitHub*.

[4] "Arcade/SpaceInvaders – Computer Archeology". *computerarcheology.com*.

[5] "mame/galaxian.c at master · mamedev/mame · GitHub". *GitHub*.

[6] "mame/galaxian.c at master · mamedev/mame · GitHub". *GitHub*.

[7] "MAME - src/mame/drivers/galdrvr.c". *archive.org*. Archived from the original on 3 January 2014.

[8] http://nfggames.com/games/x68k/

[9] "musem ~ Sharp X68000". Old-computers.com. Retrieved 2015-01-28.

[10] "Hardcore Gaming 101: Retro Japanese Computers: Gaming's Final Frontier". *hardcoregaming101.net*.

[11] "System 16 - Namco System 21 Hardware (Namco)". *system16.com*.

[12] "System 16 - Taito Air System Hardware (Taito)". *system16.com*.

[13] "System 16 - Namco Magic Edge Hornet Simulator Hardware (Namco)". *system16.com*.

[14] "MAME - src/mame/video/model2.c". *archive.org*. Archived from the original on 4 January 2013.

[15] "System 16 - Sega Model 2 Hardware (Sega)". *system16.com*.

[16] http://www.hotchips.org/wp-content/uploads/hc_archives/hc07/3_Tue/HC7.S5/HC7.5.1.pdf

[17] http://www.fujitsu.com/downloads/MAG/vol33-2/paper08.pdf

[18] "Fujitsu Develops World's First Three Dimensional Geometry Processor". *fujitsu.com*.

[19] 3dfx Glide API

[20] Søren Dreijer. "Bump Mapping Using CG (3rd Edition)". Retrieved 2007-05-30.

[21] "Large-scale deep unsupervised learning using graphics processors". Dl.acm.org. 2009-06-14. doi:10.1145/1553374.1553486. Retrieved 2014-01-21.

[22] "Linear algebra operators for GPU implementation of numerical algorithms", Kruger and Westermann, International Conf. on Computer Graphics and Interactive Techniques, 2005

[23] "ABC-SysBio—approximate Bayesian computation in Python with GPU support", Liepe et al., Bioinformatics, (2010), 26:1797-1799

[24] "A Survey of Methods for Analyzing and Improving GPU Energy Efficiency", Mittal et al., ACM Computing Surveys, 2014.

[25] "OpenCL - The open standard for parallel programming of heterogeneous systems". *khronos.org*.

[26] "GPU sales strong as AMD gains market share". *techreport.com*.

[27] "Products". S3 Graphics. Retrieved 2014-01-21.

[28] "Matrox Graphics - Products - Graphics Cards". Matrox.com. Retrieved 2014-01-21.

[29] Gary Key. "AnandTech - µATX Part 2: Intel G33 Performance Review". *anandtech.com*.

[30] Tim Tscheblockov. "Xbit Labs: Roundup of 7 Contemporary Integrated Graphics Chipsets for Socket 478 and Socket A Platforms". Retrieved 2007-06-03.

[31] Bradley Sanford. "Integrated Graphics Solutions for Graphics-Intensive Applications". Retrieved 2007-09-02.

[32] Bradley Sanford. "Integrated Graphics Solutions for Graphics-Intensive Applications". Retrieved 2007-09-02.

[33] Darren Murph. "Stanford University tailors Folding@home to GPUs". Retrieved 2007-10-04.

[34] Mike Houston. "Folding@Home - GPGPU". Retrieved 2007-10-04.

[35] "Top500 List - June 2012 | TOP500 Supercomputer Sites". Top500.org. Retrieved 2014-01-21.

[36] John Nickolls. "Stanford Lecture: Scalable Parallel Programming with CUDA on Manycore GPUs".

[37] S Harding and W Banzhaf. "Fast genetic programming on GPUs". Retrieved 2008-05-01.

[38] W Langdon and W Banzhaf. "A SIMD interpreter for Genetic Programming on GPU Graphics Cards". Retrieved 2008-05-01.

[39] V. Garcia and E. Debreuve and M. Barlaud. Fast k nearest neighbor search using GPU. In Proceedings of the CVPR Workshop on Computer Vision on GPU, Anchorage, Alaska, USA, June 2008.

[40] "eGPU candidate system list". *Tech-Inferno Forums*.

[41] Neil Mohr. "How to make an external laptop graphics adaptor". *TechRadar*.

[42] "[THUNDERBOLT NEWS] Silverstone T004... Now the ASUS XG2". *Tech-Inferno Forums*.

[43] "MSI's GUS II: External Thunderbolt GPU". *notebookreview.com*.

[44] "MSI eGPU dock in the works for GS30?". *Tech-Inferno Forums*.

[45] "Lenovo + Magma partnership delivers official Thunderbolt eGPU support". *Tech-Inferno Forums*.

[46] "DIY eGPU on Tablet PC's: experiences, benchmarks, setup, ect...". *tabletpcreview.com*.

[47] "Implementations Hub: TB, EC, mPCIe". *Tech-Inferno Forums*.

[48] BPlus eGPU adapters

[49] "□□-□□□□-□□□". *taobao.com*.

[50] Jim Galbraith (28 March 2014). "Expo Notes: Thunderbolt takes over". *Macworld*.

[51] "US$200 AKiTiO Thunder2 PCIe Box (16Gbps-TB2)". *Tech-Inferno Forums*.

[52] "Graphics chips market is showing some life". *TG Daily*. August 20, 2014. Retrieved August 22, 2014.

[53] "MATLAB Adds GPGPU Support". 2010-09-20.

6.7 External links

- NVIDIA - What is GPU computing?
- The *GPU Gems* book series
- - a Graphics Hardware History
- General-Purpose Computation Using Graphics Hardware
- How GPUs work
- GPU Caps Viewer - Video card information utility
- OpenGPU-GPU Architecture(In Chinese)
- ARM Mali GPUs Overview
- GPU Rendering Magazine

Chapter 7

OpenMP

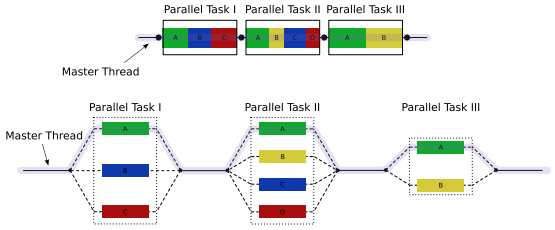
OpenMP (**Open M**ulti-**P**rocessing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran,[4] on most processor architectures and operating systems, including Solaris, AIX, HP-UX, Linux, Mac OS X, and Windows platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.[3][5][6]

OpenMP is managed by the nonprofit technology consortium *OpenMP Architecture Review Board* (or *OpenMP ARB*), jointly defined by a group of major computer hardware and software vendors, including AMD, IBM, Intel, Cray, HP, Fujitsu, Nvidia, NEC, Red Hat, Texas Instruments, Oracle Corporation, and more.[1]

OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer.

An application built with the hybrid model of parallel programming can run on a computer cluster using both OpenMP and Message Passing Interface (MPI), or more transparently through the use of OpenMP extensions for non-shared memory systems.

7.1 Introduction



An illustration of multithreading where the master thread forks off a number of threads which execute blocks of code in parallel.

See also: Fork–join model

OpenMP is an implementation of multithreading, a method of parallelizing whereby a master *thread* (a series of instructions executed consecutively) *forks* a specified number of slave *threads* and the system divides a task among them. The threads then run concurrently, with the runtime environment allocating threads to different processors.

The section of code that is meant to run in parallel is marked accordingly, with a preprocessor directive that will cause the threads to form before the section is executed.[4] Each thread has an *id* attached to it which can be obtained using a function (called omp_get_thread_num()). The thread id is an integer, and the master thread has an id of *0*. After the execution of the parallelized code, the threads *join* back into the master thread, which continues onward to the end of the program.

By default, each thread executes the parallelized section of code independently. *Work-sharing constructs* can be used to divide a task among the threads so that each thread executes its allocated part of the code. Both task parallelism and data parallelism can be achieved using OpenMP in this way.

The runtime environment allocates threads to processors depending on usage, machine load and other factors. The runtime environment can assign the number of threads based on environment variables, or the code can do so using functions. The OpenMP functions are included in a header file labelled omp.h in C/C++.

7.2 History

The OpenMP Architecture Review Board (ARB) published its first API specifications, OpenMP for Fortran 1.0, in October 1997. October the following year they released the C/C++ standard. 2000 saw version 2.0 of the Fortran specifications with version 2.0 of the C/C++ specifications being released in 2002. Version 2.5 is a combined C/C++/Fortran specification that was released in 2005.

Version 3.0 was released in May 2008. Included in the new features in 3.0 is the concept of *tasks* and the *task* construct.[7]

Version 3.1 of the OpenMP specification was released July 9, 2011.[8]

Version 4.0 of the specification was released in July 2013.[9] It adds or improves the following features: support for accelerators; atomics; error handling; thread affinity; tasking extensions; user defined reduction; SIMD support; Fortran 2003 support.[10]

7.3 The core elements

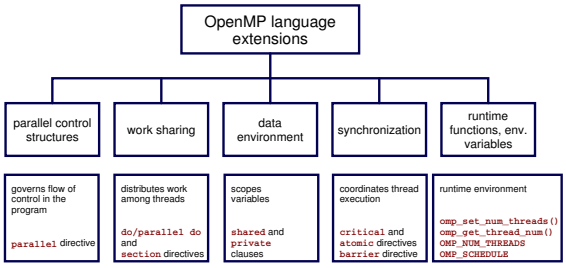


Chart of OpenMP constructs.

The core elements of OpenMP are the constructs for thread creation, workload distribution (work sharing), data-environment management, thread synchronization, user-level runtime routines and environment variables.

In C/C++, OpenMP uses #pragmas. The OpenMP specific pragmas are listed below.

7.3.1 Thread creation

The pragma *omp parallel* is used to fork additional threads to carry out the work enclosed in the construct in parallel. The original thread will be denoted as *master thread* with thread ID 0.

Example (C program): Display "Hello, world." using multiple threads.

#include <stdio.h> int main(void) { #pragma omp parallel printf("Hello, world.\n"); return 0; }

Use flag -fopenmp to compile using GCC:

$gcc -fopenmp hello.c -o hello

Output on a computer with two cores, and thus two threads:

Hello, world. Hello, world.

However, the output may also be garbled because of the race condition caused from the two threads sharing the standard output.

Hello, wHello, woorld. rld.

7.3.2 Work-sharing constructs

Used to specify how to assign independent work to one or all of the threads.

- *omp for* or *omp do*: used to split up loop iterations among the threads, also called loop constructs.
- *sections*: assigning consecutive but independent code blocks to different threads
- *single*: specifying a code block that is executed by only one thread, a barrier is implied in the end
- *master*: similar to single, but the code block will be executed by the master thread only and no barrier implied in the end.

Example: initialize the value of a large array in parallel, using each thread to do part of the work

int main(int argc, char *argv[]) { const int N = 100000; int i, a[N]; #pragma omp parallel for for (i = 0; i < N; i++) a[i] = 2 * i; return 0; }

7.3.3 OpenMP clauses

Since OpenMP is a shared memory programming model, most variables in OpenMP code are visible to all threads by default. But sometimes private variables are necessary to avoid race conditions and there is a need to pass values between the sequential part and the parallel region (the code block executed in parallel), so data environment management is introduced as *data sharing attribute clauses* by appending them to the OpenMP directive. The different types of clauses are

Data sharing attribute clauses

- *shared*: the data within a parallel region is shared, which means visible and accessible by all threads simultaneously. By default, all variables in the work sharing region are shared except the loop iteration counter.
- *private*: the data within a parallel region is private to each thread, which means each thread will have a local copy and use it as a temporary variable. A private variable is not initialized and the value is not maintained for use outside the parallel region. By default, the loop iteration counters in the OpenMP loop constructs are private.
- *default*: allows the programmer to state that the default data scoping within a parallel region will be either *shared*, or *none* for C/C++, or *shared*, *firstprivate*, *private*, or *none* for Fortran. The *none* option forces the programmer to declare each variable in

the parallel region using the data sharing attribute clauses.

- *firstprivate*: like *private* except initialized to original value.
- *lastprivate*: like *private* except original value is updated after construct.
- *reduction*: a safe way of joining work from all threads after construct.

Synchronization clauses

- *critical*: the enclosed code block will be executed by only one thread at a time, and not simultaneously executed by multiple threads. It is often used to protect shared data from race conditions.
- *atomic*: the memory update (write, or read-modify-write) in the next instruction will be performed atomically. It does not make the entire statement atomic; only the memory update is atomic. A compiler might use special hardware instructions for better performance than when using *critical*.
- *ordered*: the structured block is executed in the order in which iterations would be executed in a sequential loop
- *barrier*: each thread waits until all of the other threads of a team have reached this point. A work-sharing construct has an implicit barrier synchronization at the end.
- *nowait*: specifies that threads completing assigned work can proceed without waiting for all threads in the team to finish. In the absence of this clause, threads encounter a barrier synchronization at the end of the work sharing construct.

Scheduling clauses

- *schedule(type, chunk)*: This is useful if the work sharing construct is a do-loop or for-loop. The iteration(s) in the work sharing construct are assigned to threads according to the scheduling method defined by this clause. The three types of scheduling are:

1. *static*: Here, all the threads are allocated iterations before they execute the loop iterations. The iterations are divided among threads equally by default. However, specifying an integer for the parameter *chunk* will allocate chunk number of contiguous iterations to a particular thread.
2. *dynamic*: Here, some of the iterations are allocated to a smaller number of threads. Once a particular thread finishes its allocated iteration, it returns to get another one from the iterations that are left. The parameter *chunk* defines the number of contiguous iterations that are allocated to a thread at a time.
3. *guided*: A large chunk of contiguous iterations are allocated to each thread dynamically (as above). The chunk size decreases exponentially with each successive allocation to a minimum size specified in the parameter *chunk*

IF control

- *if*: This will cause the threads to parallelize the task only if a condition is met. Otherwise the code block executes serially.

Initialization

- *firstprivate*: the data is private to each thread, but initialized using the value of the variable using the same name from the master thread.
- *lastprivate*: the data is private to each thread. The value of this private data will be copied to a global variable using the same name outside the parallel region if current iteration is the last iteration in the parallelized loop. A variable can be both *firstprivate* and *lastprivate*.
- *threadprivate*: The data is a global data, but it is private in each parallel region during the runtime. The difference between *threadprivate* and *private* is the global scope associated with threadprivate and the preserved value across parallel regions.

Data copying

- *copyin*: similar to *firstprivate* for *private* variables, *threadprivate* variables are not initialized, unless using *copyin* to pass the value from the corresponding global variables. No *copyout* is needed because the value of a threadprivate variable is maintained throughout the execution of the whole program.
- *copyprivate*: used with *single* to support the copying of data values from private objects on one thread (the *single* thread) to the corresponding objects on other threads in the team.

Reduction

- *reduction(operator | intrinsic : list)*: the variable has a local copy in each thread, but the values of the local copies will be summarized (reduced) into a global shared variable. This is very useful if a particular operation (specified in *operator* for this particular clause) on a datatype that runs iteratively so that its value at a particular iteration depends on its value

at a prior iteration. Basically, the steps that lead up to the operational increment are parallelized, but the threads gather up and wait before updating the datatype, then increments the datatype in order so as to avoid racing condition. This would be required in parallelizing numerical integration of functions and differential equations, as a common example.

Others

- *flush*: The value of this variable is restored from the register to the memory for using this value outside of a parallel part
- *master*: Executed only by the master thread (the thread which forked off all the others during the execution of the OpenMP directive). No implicit barrier; other team members (threads) not required to reach.

7.3.4 User-level runtime routines

Used to modify/check the number of threads, detect if the execution context is in a parallel region, how many processors in current system, set/unset locks, timing functions, etc.

7.3.5 Environment variables

A method to alter the execution features of OpenMP applications. Used to control loop iterations scheduling, default number of threads, etc. For example *OMP_NUM_THREADS* is used to specify number of threads for an application.

7.4 Sample programs

In this section, some sample programs are provided to illustrate the concepts explained above.

7.4.1 Hello World

A basic program that exercises the *parallel*, *private* and *barrier* directives, and the functions omp_get_thread_num and omp_get_num_threads (not to be confused).

C

This C program can be compiled using gcc-4.4 with the flag -fopenmp

#include <omp.h> #include <stdio.h> #include <stdlib.h> int main (int argc, char *argv[]) { int th_id, nthreads; #pragma omp parallel private(th_id) { th_id = omp_get_thread_num(); printf("Hello World from thread %d\n", th_id); #pragma omp barrier if (th_id == 0) { nthreads = omp_get_num_threads(); printf("There are %d threads\n",nthreads); } } return EXIT_SUCCESS; }

C++

This C++ program can be compiled using GCC: g++ -Wall -fopenmp test.cpp
NOTE: The IOstreams library is not thread-safe. Therefore, for instance, *cout* calls must be executed in critical areas or by only one thread (e.g. masterthread).

#include <iostream> using namespace std; #include <omp.h> int main(int argc, char *argv[]) { int th_id, nthreads; #pragma omp parallel private(th_id) shared(nthreads) { th_id = omp_get_thread_num(); #pragma omp critical { cout << "Hello World from thread " << th_id << '\n'; } #pragma omp barrier #pragma omp master { nthreads = omp_get_num_threads(); cout << "There are " << nthreads << " threads" << '\n'; } } return 0; }

Fortran 77

Here is a Fortran 77 version.

PROGRAM HELLO INTEGER ID, NTHRDS INTEGER OMP_GET_THREAD_NUM, OMP_GET_NUM_THREADS C$OMP PARALLEL PRIVATE(ID) ID = OMP_GET_THREAD_NUM() PRINT *, 'HELLO WORLD FROM THREAD', ID C$OMP BARRIER IF (ID .EQ. 0) THEN NTHRDS = OMP_GET_NUM_THREADS() PRINT *, 'THERE ARE', NTHRDS, 'THREADS' END IF C$OMP END PARALLEL END

Fortran 90 free form

Here is a Fortran 90 free form version.

program hello90 use omp_lib integer:: id, nthreads !$omp parallel private(id) id = omp_get_thread_num() write (*,*) 'Hello World from thread', id !$omp barrier if (id == 0) then nthreads = omp_get_num_threads() write (*,*) 'There are', nthreads, 'threads' end if !$omp end parallel end program

7.4.2 Clauses in work-sharing constructs (in C/C++)

The application of some OpenMP clauses are illustrated in the simple examples in this section. The piece of code below updates the elements of an array *b* by performing a simple operation on the elements of an array *a*. The parallelization is done by the OpenMP directive *#pragma omp*. The scheduling of tasks is dynamic. Notice how the iteration counters *j* and *k* have to be made private, whereas the primary iteration counter *i* is private by default. The task of running through *i* is divided among multiple threads, and each thread creates its own versions of *j* and *k* in its execution stack, thus doing the full task allocated to it and updating the allocated part of the array *b* at the same time as the other threads.

#define CHUNKSIZE 1 /*defines the chunk size as 1 contiguous iteration*/ /*forks off the threads*/ #pragma omp parallel private(j,k) { /*Starts the work sharing construct*/ #pragma omp for schedule(dynamic, CHUNKSIZE) for(i = 2; i <= N-1; i++) for(j = 2; j <= i; j++) for(k = 1; k <= M; k++) b[i][j] += a[i-1][j]/k + a[i+1][j]/k; }

The next piece of code is a common usage of the *reduction* clause to calculate reduced sums. Here, we add up all the elements of an array *a* with an *i*-dependent weight using a for loop, which we parallelize using OpenMP directives and reduction clause. The scheduling is kept static.

#define N 10000 /*size of a*/ void calculate(long *); /*The function that calculates the elements of a*/ int i; long w; long a[N]; calculate(a); long sum = 0; /*forks off the threads and starts the work-sharing construct*/ #pragma omp parallel for private(w) reduction(+:sum) schedule(static,1) for(i = 0; i < N; i++) { w = i*i; sum = sum + w*a[i]; } printf("\n %li",sum);

An equivalent, less elegant, implementation of the above code is to create a local sum variable for each thread ("loc_sum"), and make a protected update of the global variable *sum* at the end of the process, through the directive *critical*. Note that this protection is critical, as explained elsewhere.

... long sum = 0, loc_sum; /*forks off the threads and starts the work-sharing construct*/ #pragma omp parallel private(w,loc_sum) { loc_sum = 0; #pragma omp for schedule(static,1) for(i = 0; i < N; i++) { w = i*i; loc_sum = loc_sum + w*a[i]; } #pragma omp critical sum = sum + loc_sum; } printf("\n %li",sum);

7.5 Implementations

OpenMP has been implemented in many commercial compilers. For instance, Visual C++ 2005, 2008, 2010, 2012 and 2013 support it (OpenMP 2.0, in Professional, Team System, Premium and Ultimate editions[11][12][13]), as well as Intel Parallel Studio for various processors.[14] Oracle Solaris Studio compilers and tools support the latest OpenMP specifications with productivity enhancements for Solaris OS (UltraSPARC and x86/x64) and Linux platforms. The Fortran, C and C++ compilers from The Portland Group also support OpenMP 2.5. GCC has also supported OpenMP since version 4.2.

Compilers with an implementation of OpenMP 3.0:

- GCC 4.3.1
- Mercurium compiler
- Intel Fortran and C/C++ versions 11.0 and 11.1 compilers, Intel C/C++ and Fortran Composer XE 2011 and Intel Parallel Studio.
- IBM XL C/C++ compiler[15]
- Sun Studio 12 update 1 has a full implementation of OpenMP 3.0[16]

Several compilers support OpenMP 3.1:

- GCC 4.7[17]
- Intel Fortran and C/C++ compilers 12.1[18]

Compilers supporting OpenMP 4.0:

- GCC 4.9.0 for C/C++, GCC 4.9.1 for Fortran[19][20]
- Intel Fortran and C/C++ compilers 15.0[21]

Auto-parallelizing compilers that generates source code annotated with OpenMP directives:

- iPat/OMP
- Parallware
- PLUTO
- ROSE (compiler framework)
- S2P by KPIT Cummins Infosystems Ltd.

A number of profilers and debuggers have specific support for OpenMP:

- Allinea DDT - debugger for OpenMP and MPI codes
- Allinea MAP - profiler for OpenMP and MPI codes
- ompP - profiler for OpenMP
- VAMPIR - profiler for OpenMP and MPI codes

7.6 Pros and cons

Pros:

- Portable multithreading code (in C/C++ and other languages, one typically has to call platform-specific primitives in order to get multithreading).
- Simple: need not deal with message passing as MPI does.
- Data layout and decomposition is handled automatically by directives.
- Scalability comparable to MPI on shared-memory systems.[22]
- Incremental parallelism: can work on one part of the program at one time, no dramatic change to code is needed.
- Unified code for both serial and parallel applications: OpenMP constructs are treated as comments when sequential compilers are used.
- Original (serial) code statements need not, in general, be modified when parallelized with OpenMP. This reduces the chance of inadvertently introducing bugs.
- Both coarse-grained and fine-grained parallelism are possible.
- In irregular multi-physics applications which do not adhere solely to the SPMD mode of computation, as encountered in tightly coupled fluid-particulate systems, the flexibility of OpenMP can have a big performance advantage over MPI.[23][24]
- Can be used on various accelerators such as GPGPU.[25]

Cons:

- Risk of introducing difficult to debug synchronization bugs and race conditions.[26][27]
- Currently only runs efficiently in shared-memory multiprocessor platforms (see however Intel's Cluster OpenMP and other distributed shared memory platforms).
- Requires a compiler that supports OpenMP.
- Scalability is limited by memory architecture.
- No support for compare-and-swap.[28]
- Reliable error handling is missing.
- Lacks fine-grained mechanisms to control thread-processor mapping.
- High chance of accidentally writing false sharing code.
- Multithreaded Executables often incur longer startup times than single threaded applications, therefore if the running time of the program is short enough there may be no advantage to making it multithreaded.

7.7 Performance expectations

One might expect to get an N times speedup when running a program parallelized using OpenMP on a N processor platform. However, this seldom occurs for these reasons:

- When a dependency exists, a process must wait until the data it depends on is computed.
- When multiple processes share a non-parallel proof resource (like a file to write in), their requests are executed sequentially. Therefore each thread must wait until the other thread releases the resource.
- A large part of the program may not be parallelized by OpenMP, which means that the theoretical upper limit of speedup is limited according to Amdahl's law.
- N processors in a symmetric multiprocessing (SMP) may have N times the computation power, but the memory bandwidth usually does not scale up N times. Quite often, the original memory path is shared by multiple processors and performance degradation may be observed when they compete for the shared memory bandwidth.
- Many other common problems affecting the final speedup in parallel computing also apply to OpenMP, like load balancing and synchronization overhead.

7.8 Thread affinity

Some vendors recommend setting the processor affinity on OpenMP threads to associate them with particular processor cores.[29][30][31] This minimizes thread migration and context-switching cost among cores. It also improves the data locality and reduces the cache-coherency traffic among the cores (or processors).

7.9 Benchmarks

There are some public domain OpenMP benchmarks for users to try.

- NAS parallel benchmark
- OpenMP validation suite
- OpenMP source code repository
- EPCC OpenMP Microbenchmarks

7.10 Learning resources online

- Tutorial on llnl.gov
- Reference/tutorial page on nersc.gov
- Tutorial in CI-Tutor

7.11 See also

- Cilk and Cilk Plus
- Message Passing Interface
- Concurrency (computer science)
- Heterogeneous System Architecture
- Parallel computing
- Parallel programming model
- POSIX Threads
- Unified Parallel C
- X10 (programming language)
- Parallel Virtual Machine
- Bulk synchronous parallel
- Grand Central Dispatch – comparable technology for C, C++, and Objective-C by Apple
- Partitioned global address space
- GPGPU
- CUDA – Nvidia
- AMD FireStream
- Octopiler
- OpenCL – Standard supported by Apple, Nvidia, Intel, IBM, AMD/ATI and many others.
- OpenACC – a standard for GPU acceleration, which is planned to be merged into openMP

7.12 References

[1] "About the OpenMP ARB and". OpenMP.org. 2013-07-11. Retrieved 2013-08-14.

[2] OpenMP 4.0 Specification Released

[3] "OpenMP Compilers". OpenMP.org. 2013-04-10. Retrieved 2013-08-14.

[4] Gagne, Abraham Silberschatz, Peter Baer Galvin, Greg. *Operating system concepts* (9th ed.). Hoboken, N.J.: Wiley. pp. 181–182. ISBN 9781118063330.

[5] OpenMP Tutorial at Supercomputing 2008

[6] Using OpenMP - Portable Shared Memory Parallel Programming - Download Book Examples and Discuss

[7] "OpenMP Application Program Interface, Version 3.0". openmp.org. May 2008. Retrieved 2014-02-06.

[8] "OpenMP Application Program Interface, Version 3.1". openmp.org. July 2011. Retrieved 2014-02-06.

[9] "OpenMP 4.0 API Released". OpenMP.org. 2013-07-26. Retrieved 2013-08-14.

[10] "OpenMP Application Program Interface, Version 4.0". openmp.org. July 2013. Retrieved 2014-02-06.

[11] Visual C++ Editions, Visual Studio 2005

[12] Visual C++ Editions, Visual Studio 2008

[13] Visual C++ Editions, Visual Studio 2010

[14] David Worthington, "Intel addresses development life cycle with Parallel Studio", SDTimes, 26 May 2009 (accessed 28 May 2009)

[15] "XL C/C++ for Linux Features", (accessed 9 June 2009)

[16] "Oracle Technology Network for Java Developers | Oracle Technology Network | Oracle". Developers.sun.com. Retrieved 2013-08-14.

[17] "openmp - GCC Wiki". Gcc.gnu.org. 2013-07-30. Retrieved 2013-08-14.

[18] Submitted by Patrick Kennedy... on Fri, 09/02/2011 - 11:28 (2011-09-06). "Intel® C++ and Fortran Compilers now support the OpenMP* 3.1 Specification | Intel® Developer Zone". Software.intel.com. Retrieved 2013-08-14.

[19] "GCC 4.9 Release Series - Changes". www.gnu.org.

[20] "openmp - GCC Wiki". Gcc.gnu.org. 2013-07-30. Retrieved 2013-08-14.

[21] "OpenMP* 4.0 Features in Intel Compiler 15.0". Software.intel.com.

[22] Amritkar, Amit; Tafti, Danesh; Liu, Rui; Kufrin, Rick; Chapman, Barbara (2012). "OpenMP parallelism for fluid and fluid-particulate systems". *Parallel Computing* **38** (9): 501. doi:10.1016/j.parco.2012.05.005.

[23] Amritkar, Amit; Tafti, Danesh; Liu, Rui; Kufrin, Rick; Chapman, Barbara (2012). "OpenMP parallelism for fluid and fluid-particulate systems". *Parallel Computing* **38** (9): 501. doi:10.1016/j.parco.2012.05.005.

[24] Amritkar, Amit; Deb, Surya; Tafti, Danesh (2014). "Efficient parallel CFD-DEM simulations using OpenMP". *Journal of Computational Physics* **256**: 501. Bibcode:2014JCoPh.256..501A. doi:10.1016/j.jcp.2013.09.007.

[25] Frequently Asked Questions on OpenMP

[26] Detecting and Avoiding OpenMP Race Conditions in C++

[27] Alexey Kolosov, Evgeniy Ryzhkov, Andrey Karpov 32 OpenMP traps for C++ developers

[28] Stephen Blair-Chappell, Intel Corporation, Becoming a Parallel Programming Expert in Nine Minutes, presentation on ACCU 2010 conference

[29] Chen, Yurong (2007-11-15). "Multi-Core Software". *Intel Technology Journal* (Intel) **11** (4). doi:10.1535/itj.1104.08.

[30] "OMPM2001 Result". SPEC. 2008-01-28.

[31] "OMPM2001 Result". SPEC. 2003-04-01.

7.13 Further reading

- Quinn Michael J, Parallel Programming in C with MPI and OpenMP McGraw-Hill Inc. 2004. ISBN 0-07-058201-7
- R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. McDonald, Parallel Programming in OpenMP. Morgan Kaufmann, 2000. ISBN 1-55860-671-8
- R. Eigenmann (Editor), M. Voss (Editor), OpenMP Shared Memory Parallel Programming: International Workshop on OpenMP Applications and Tools, WOMPAT 2001, West Lafayette, IN, USA, July 30–31, 2001. (Lecture Notes in Computer Science). Springer 2001. ISBN 3-540-42346-X
- B. Chapman, G. Jost, R. van der Pas, D.J. Kuck (foreword), Using OpenMP: Portable Shared Memory Parallel Programming. The MIT Press (October 31, 2007). ISBN 0-262-53302-2
- Parallel Processing via MPI & OpenMP, M. Firuziaan, O. Nommensen. Linux Enterprise, 10/2002
- MSDN Magazine article on OpenMP
- SC08 OpenMP Tutorial (PDF) - Hands-On Introduction to OpenMP, Mattson and Meadows, from SC08 (Austin)
- OpenMP Specifications
- Parallel Programming in Fortran 95 using OpenMP (PDF)

7.14 External links

- Official website , includes the latest OpenMP specifications, links to resources, and a lively set of forums where questions about OpenMP can be asked and are answered by the experts and implementors.
- GOMP is GCC's OpenMP implementation, part of GCC
- IBM Octopiler with OpenMP support
- Blaise Barney, Lawrence Livermore National Laboratory site on OpenMP
- ompca, an application in REDLIB project for the interactive symbolic model-checker of C/C++ programs with OpenMP directives
- Combining OpenMP and MPI (PDF)
- Mixing MPI and OpenMP

Chapter 8

Message Passing Interface

Message Passing Interface (**MPI**) is a standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computers. The standard defines the syntax and semantics of a core of library routines useful to a wide range of users writing portable message-passing programs in different computer programming languages such as Fortran, C, C++ and Java. There are several well-tested and efficient implementations of MPI, including some that are free or in the public domain. These fostered the development of a parallel software industry, and encouraged development of portable and scalable large-scale parallel applications.

8.1 History

The message passing interface effort began in the summer of 1991 when a small group of researchers started discussions at a mountain retreat in Austria. Out of that discussion came a Workshop on Standards for Message Passing in a Distributed Memory Environment held on April 29–30, 1992 in Williamsburg, Virginia. At this workshop the basic features essential to a standard message-passing interface were discussed, and a working group established to continue the standardization process. Jack Dongarra, Rolf Hempel, Tony Hey, and David W. Walker put forward a preliminary draft proposal in November 1992, this was known as MPI1. In November 1992, a meeting of the MPI working group was held in Minneapolis, at which it was decided to place the standardization process on a more formal footing. The MPI working group met every 6 weeks throughout the first 9 months of 1993. The draft MPI standard was presented at the Supercomputing '93 conference in November 1993. After a period of public comments, which resulted in some changes in MPI, version 1.0 of MPI was released in June 1994. These meetings and the email discussion together constituted the MPI Forum, membership of which has been open to all members of the high performance computing community.

The MPI effort involved about 80 people from 40 organizations, mainly in the United States and Europe. Most of the major vendors of concurrent computers were involved in MPI along with researchers from universities, government laboratories, and industry.

The MPI standard defines the syntax and semantics of a core of library routines useful to a wide range of users writing portable message passing programs in Fortran and C.

MPI provides parallel hardware vendors with a clearly defined base set of routines that can be efficiently implemented. As a result, hardware vendors can build upon this collection of standard low-level routines to create higher-level routines for the distributed-memory communication environment supplied with their parallel machines. MPI provides a simple-to-use portable interface for the basic user, yet powerful enough to allow programmers to use the high-performance message passing operations available on advanced machines.

As an effort to create a "true" standard for message passing, researchers incorporated the most useful features of several systems into MPI, rather than choose one system to adopt as a standard. Features were used from systems by IBM, Intel, nCUBE, PVM, Express, P4 and PARMACS. The message passing paradigm is attractive because of wide portability and can be used in communication for distributed-memory and shared-memory multiprocessors, networks of workstations, and a combination of these elements. The paradigm is applicable in multiple settings, independent of network speed or memory architecture.

Support for MPI meetings came in part from ARPA and US National Science Foundation under grant ASC-9310330, NSF Science and Technology Center Cooperative agreement number CCR-8809615, and the Commission of the European Community through Esprit Project P6643. The University of Tennessee also made financial contributions to the MPI Forum.

8.2 Overview

MPI is a language-independent communications protocol used to program parallel computers. Both point-to-point and collective communication are supported. MPI "is a message-passing application programmer interface,

together with protocol and semantic specifications for how its features must behave in any implementation."[1] MPI's goals are high performance, scalability, and portability. MPI remains the dominant model used in high-performance computing today.[2]

MPI is not sanctioned by any major standards body; nevertheless, it has become a *de facto* standard for communication among processes that model a parallel program running on a distributed memory system. Actual distributed memory supercomputers such as computer clusters often run such programs. The principal MPI-1 model has no shared memory concept, and MPI-2 has only a limited distributed shared memory concept. Nonetheless, MPI programs are regularly run on shared memory computers. Designing programs around the MPI model (contrary to explicit shared memory models) has advantages over NUMA architectures since MPI encourages memory locality.

Although MPI belongs in layers 5 and higher of the OSI Reference Model, implementations may cover most layers, with sockets and Transmission Control Protocol (TCP) used in the transport layer.

Most MPI implementations consist of a specific set of routines (i.e., an API) directly callable from C, C++, Fortran and any language able to interface with such libraries, including C#, Java or Python. The advantages of MPI over older message passing libraries are portability (because MPI has been implemented for almost every distributed memory architecture) and speed (because each implementation is in principle optimized for the hardware on which it runs).

MPI uses Language Independent Specifications (LIS) for calls and language bindings. The first MPI standard specified ANSI C and Fortran-77 bindings together with the LIS. The draft was presented at Supercomputing 1994 (November 1994)[3] and finalized soon thereafter. About 128 functions constitute the MPI-1.3 standard which was released as the final end of the MPI-1 series in 2008.[4]

At present, the standard has several versions: version 1.3 (commonly abbreviated *MPI-1*), which emphasizes message passing and has a static runtime environment, MPI-2.2 (MPI-2), which includes new features such as parallel I/O, dynamic process management and remote memory operations,[5] and MPI-3.0 (MPI-3), which includes extensions to the collective operations with nonblocking versions and extensions to the one-sided operations.[6] MPI-2's LIS specifies over 500 functions and provides language bindings for ANSI C, ANSI C++, and ANSI Fortran (Fortran90). Object interoperability was also added to allow easier mixed-language message passing programming. A side-effect of standardizing MPI-2, completed in 1996, was clarifying the MPI-1 standard, creating the MPI-1.2.

MPI-2 is mostly a superset of MPI-1, although some functions have been deprecated. MPI-1.3 programs still work under MPI implementations compliant with the MPI-2 standard.

MPI-3 includes new Fortran 2008 bindings, while it removes deprecated C++ bindings as well as many deprecated routines and MPI objects.

MPI is often compared with Parallel Virtual Machine (PVM), which is a popular distributed environment and message passing system developed in 1989, and which was one of the systems that motivated the need for standard parallel message passing. Threaded shared memory programming models (such as Pthreads and OpenMP) and message passing programming (MPI/PVM) can be considered as complementary programming approaches, and can occasionally be seen together in applications, e.g. in servers with multiple large shared-memory nodes.

8.3 Functionality

The MPI interface is meant to provide essential virtual topology, synchronization, and communication functionality between a set of processes (that have been mapped to nodes/servers/computer instances) in a language-independent way, with language-specific syntax (bindings), plus a few language-specific features. MPI programs always work with processes, but programmers commonly refer to the processes as processors. Typically, for maximum performance, each CPU (or core in a multi-core machine) will be assigned just a single process. This assignment happens at runtime through the agent that starts the MPI program, normally called mpirun or mpiexec.

MPI library functions include, but are not limited to, point-to-point rendezvous-type send/receive operations, choosing between a Cartesian or graph-like logical process topology, exchanging data between process pairs (send/receive operations), combining partial results of computations (gather and reduce operations), synchronizing nodes (barrier operation) as well as obtaining network-related information such as the number of processes in the computing session, current processor identity that a process is mapped to, neighboring processes accessible in a logical topology, and so on. Point-to-point operations come in synchronous, asynchronous, buffered, and *ready* forms, to allow both relatively stronger and weaker semantics for the synchronization aspects of a rendezvous-send. Many outstanding operations are possible in asynchronous mode, in most implementations.

MPI-1 and MPI-2 both enable implementations that overlap communication and computation, but practice and theory differ. MPI also specifies *thread safe* interfaces, which have cohesion and coupling strategies that help avoid hidden state within the interface. It is relatively easy to write multithreaded point-to-point MPI code, and some implementations support such code. Multithreaded collective communication is best accomplished with multiple copies of Communicators, as described below.

8.4 Concepts

MPI provides a rich range of abilities. The following concepts help in understanding and providing context for all of those abilities and help the programmer to decide what functionality to use in their application programs. Four of MPI's eight basic concepts are unique to MPI-2.

8.4.1 Communicator

Communicator objects connect groups of processes in the MPI session. Each communicator gives each contained process an independent identifier and arranges its contained processes in an ordered topology. MPI also has explicit groups, but these are mainly good for organizing and reorganizing groups of processes before another communicator is made. MPI understands single group intracommunicator operations, and bilateral intercommunicator communication. In MPI-1, single group operations are most prevalent. Bilateral operations mostly appear in MPI-2 where they include collective communication and dynamic in-process management.

Communicators can be partitioned using several MPI commands. These commands include MPI_COMM_SPLIT, where each process joins one of several colored sub-communicators by declaring itself to have that color.

8.4.2 Point-to-point basics

A number of important MPI functions involve communication between two specific processes. A popular example is MPI_Send, which allows one specified process to send a message to a second specified process. Point-to-point operations, as these are called, are particularly useful in patterned or irregular communication, for example, a data-parallel architecture in which each processor routinely swaps regions of data with specific other processors between calculation steps, or a master-slave architecture in which the master sends new task data to a slave whenever the prior task is completed.

MPI-1 specifies mechanisms for both blocking and non-blocking point-to-point communication mechanisms, as well as the so-called 'ready-send' mechanism whereby a send request can be made only when the matching receive request has already been made.

8.4.3 Collective basics

Collective functions involve communication among all processes in a process group (which can mean the entire process pool or a program-defined subset). A typical function is the MPI_Bcast call (short for "broadcast"). This function takes data from one node and sends it to all processes in the process group. A reverse operation is the MPI_Reduce call, which takes data from all processes in a group, performs an operation (such as summing), and stores the results on one node. Reduce is often useful at the start or end of a large distributed calculation, where each processor operates on a part of the data and then combines it into a result.

Other operations perform more sophisticated tasks, such as MPI_Alltoall which rearranges n items of data processor such that the nth node gets the nth item of data from each.

8.4.4 Derived datatypes

Many MPI functions require that you specify the type of data which is sent between processors. This is because these functions pass variables, not defined types. If the data type is a standard one, such as int, char, double, etc., you can use predefined MPI datatypes such as MPI_INT, MPI_CHAR, MPI_DOUBLE.

Here is an example in C that passes an array of ints and all the processors want to send their arrays to the root with MPI_Gather:

int array[100]; int root, total_p, *receive_array; MPI_Comm_size(comm, &total_p); receive_array=malloc(total_p*100*sizeof(*receive_array)); MPI_Gather(array, 100, MPI_INT, receive_array, 100, MPI_INT, root, comm);

However, you may instead wish to send data as one block as opposed to 100 ints. To do this define a "contiguous block" derived data type.

MPI_Datatype newtype; MPI_Type_contiguous(100, MPI_INT, &newtype); MPI_Type_commit(&newtype); MPI_Gather(array, 1, newtype, receive_array, 1, newtype, root, comm);

Passing a class or a data structure cannot use a predefined data type. MPI_Type_create_struct creates an MPI derived data type from MPI_predefined data types, as follows:

int MPI_Type_create_struct(int count, int blocklen[], MPI_Aint disp[], MPI_Datatype type[], MPI_Datatype *newtype)

where count is a number of blocks, also number of entries in blocklen[], disp[], and type[]:

- blocklen[] — number of elements in each block (array of integer)
- disp[] — byte displacement of each block (array of integer)
- type[] — type of elements in each block (array of handles to datatype objects).

The disp[] array is needed because processors require the variables to be aligned a specific way on the memory. For example, Char is one byte and can go anywhere on the memory. Short is 2 bytes, so it goes to even memory addresses. Long is 4 bytes, it goes on locations divisible by 4 and so on. The compiler tries to accommodate this architecture in a class or data structure by padding the variables. The safest way to find the distance between different variables in a data structure is by obtaining their addresses with MPI_Get_address. This function calculates the displacement of all the structure's elements from the start of the data structure.

Given the following data structures:

typedef struct{ int f; short p; } A; typedef struct{ A a; int pp,vp; } B;

Here's the C code for building an MPI-derived data type:

void define_MPI_datatype(){ //The first and last elements mark the beg and end of data structure int blocklen[6]={1,1,1,1,1,1}; MPI_Aint disp[6]; MPI_Datatype newtype; MPI_Datatype type[6]={MPI_LB, MPI_INT, MPI_SHORT, MPI_INT, MPI_INT, MPI_UB}; //You need an array to establish the upper bound of the data structure B findsize[2]; MPI_Aint findsize_addr, a_addr, f_addr, p_addr, pp_addr, vp_addr, UB_addr; int error; MPI_Get_address(&findsize[0], &findsize_addr); MPI_Get_address(&(findsize[0]).a, &a_addr); MPI_Get_address(&((findsize[0]).a).f, &f_addr); MPI_Get_address(&((findsize[0]).a).p, &p_addr); MPI_Get_address(&(findsize[0]).pp, &pp_addr); MPI_Get_address(&(findsize[0]).vp, &vp_addr); MPI_Get_address(&findsize[1],&UB_addr); disp[0]=a_addr-findsize_addr; disp[1]=f_addr-findsize_addr; disp[2]=p_addr-findsize_addr; disp[3]=pp_addr-findsize_addr; disp[4]=vp_addr-findsize_addr; disp[5]=UB_addr-findsize_addr; error=MPI_Type_create_struct(6, blocklen, disp, type, &newtype); MPI_Type_commit(&newtype); }

8.5 MPI-2 concepts

8.5.1 One-sided communication

MPI-2 defines three one-sided communications operations, Put, Get, and Accumulate, being a write to remote memory, a read from remote memory, and a reduction operation on the same memory across a number of tasks, respectively. Also defined are three different methods to synchronize this communication (global, pairwise, and remote locks) as the specification does not guarantee that these operations have taken place until a synchronization point.

These types of call can often be useful for algorithms in which synchronization would be inconvenient (e.g. distributed matrix multiplication), or where it is desirable for tasks to be able to balance their load while other processors are operating on data.

8.5.2 Collective extensions

This section needs to be developed.

8.5.3 Dynamic process management

The key aspect is "the ability of an MPI process to participate in the creation of new MPI processes or to establish communication with MPI processes that have been started separately." The MPI-2 specification describes three main interfaces by which MPI processes can dynamically establish communications, MPI_Comm_spawn, MPI_Comm_accept/MPI_Comm_connect and MPI_Comm_join. The MPI_Comm_spawn interface allows an MPI process to spawn a number of instances of the named MPI process. The newly spawned set of MPI processes form a new MPI_COMM_WORLD intracommunicator but can communicate with the parent and the intercommunicator the function returns. MPI_Comm_spawn_multiple is an alternate interface that allows the different instances spawned to be different binaries with different arguments.[7]

8.5.4 I/O

The parallel I/O feature is sometimes called MPI-IO,[8] and refers to a set of functions designed to abstract I/O management on distributed systems to MPI, and allow files to be easily accessed in a patterned way using the existing derived datatype functionality.

The little research that has been done on this feature indicates the difficulty for good performance. For example, some implementations of sparse matrix-vector multiplications using the MPI I/O library are disastrously inefficient.[9]

8.6 Implementations

8.6.1 'Classical' cluster and supercomputer implementations

The MPI implementation language is not constrained to match the language or languages it seeks to support at runtime. Most implementations combine C, C++ and assembly language, and target C, C++, and Fortran programmers. Bindings are available for many other languages, including Perl, Python, R, Ruby, Java, CL.

The initial implementation of the MPI 1.x standard was MPICH, from Argonne National Laboratory (ANL) and Mississippi State University. IBM also was an early implementor, and most early 90s supercomputer companies either commercialized MPICH, or built their own implementation. LAM/MPI from Ohio Supercomputer Center was another early open implementation. ANL has continued developing MPICH for over a decade, and now offers MPICH 2, implementing the MPI-2.1 standard. LAM/MPI and a number of other MPI efforts recently merged to form Open MPI. Many other efforts are derivatives of MPICH, LAM, and other works, including, but not limited to, commercial implementations from HP, Intel, and Microsoft.

8.6.2 Python

MPI Python implementations include: pyMPI, mpi4py,[10] pypar,[11] MYMPI,[12] and the MPI submodule in ScientificPython. pyMPI is notable because it is a variant python interpreter, while pypar, MYMPI, and ScientificPython's module are import modules. They make it the coder's job to decide where the call to MPI_Init belongs. Recently the well known Boost C++ Libraries acquired Boost:MPI which included the MPI Python Bindings.[13] This is of particular help for mixing C++ and Python.

8.6.3 OCaml

The OCamlMPI Module[14] implements a large subset of MPI functions and is in active use in scientific computing. An eleven thousand line OCaml program was "MPI-ified" using the module, with an additional 500 lines of code and slight restructuring and ran with excellent results on up to 170 nodes in a supercomputer.[15]

8.6.4 Java

Although Java does not have an official MPI binding, several groups attempt to bridge the two, with different degrees of success and compatibility. One of the first attempts was Bryan Carpenter's mpiJava,[16] essentially a set of Java Native Interface (JNI) wrappers to a local C MPI library, resulting in a hybrid implementation with limited portability, which also has to be compiled against the specific MPI library being used.

However, this original project also defined the mpiJava API[17] (a de facto MPI API for Java that closely followed the equivalent C++ bindings) which other subsequent Java MPI projects adopted. An alternative, less-used API is MPJ API,[18] designed to be more object-oriented and closer to Sun Microsystems' coding conventions. Beyond the API, Java MPI libraries can be either dependent on a local MPI library, or implement the message passing functions in Java, while some like P2P-MPI also provide peer-to-peer functionality and allow mixed platform operation.

Some of the most challenging parts of Java/MPI arise from Java characteristics such as the lack of explicit pointers and the linear memory address space for its objects, which make transferring multidimensional arrays and complex objects inefficient. Workarounds usually involve transferring one line at a time and/or performing explicit de-serialization and casting at both sending and receiving ends, simulating C or Fortran-like arrays by the use of a one-dimensional array, and pointers to primitive types by the use of single-element arrays, thus resulting in programming styles quite far from Java conventions.

Another Java message passing system is MPJ Express.[19] Recent versions can be executed in cluster and multicore configurations. In the cluster configuration, it can execute parallel Java applications on clusters and clouds. Here Java sockets or specialized I/O interconnects like Myrinet can support messaging between MPJ Express processes. It can also utilize native C implementation of MPI using its native device. In the multicore configuration, a parallel Java application is executed on multicore processors. In this mode, MPJ Express processes are represented by Java threads.

8.6.5 Matlab

There are a few academic implementations of MPI using Matlab. Matlab has their own parallel extension library implemented using MPI and PVM.

8.6.6 R

R implementations of MPI include Rmpi[20] and pbdMPI,[21] where Rmpi focuses on manager-workers parallelism while pbdMPI focuses on SPMD parallelism. Both implementations fully support Open MPI or MPICH2.

8.6.7 Common Language Infrastructure

The two managed Common Language Infrastructure (CLI) .NET implementations are Pure Mpi.NET[22] and MPI.NET,[23] a research effort at Indiana University licensed under a BSD-style license. It is compatible with Mono, and can make full use of underlying low-latency MPI network fabrics.

8.6.8 Hardware implementations

MPI hardware research focuses on implementing MPI directly in hardware, for example via processor-in-memory, building MPI operations into the microcircuitry of the RAM chips in each node. By implication, this approach

is independent of the language, OS or CPU, but cannot be readily updated or removed.

Another approach has been to add hardware acceleration to one or more parts of the operation, including hardware processing of MPI queues and using RDMA to directly transfer data between memory and the network interface without CPU or OS kernel intervention.

8.6.9 mpicc

mpicc is a program which helps the programmer to use a standard C programming language compiler together with the Message Passing Interface (MPI) libraries, most commonly the OpenMPI implementation which is found in many TOP-500 supercomputers, for the purpose of producing parallel processing programs to run over computer clusters (often Beowulf clusters). The mpicc program uses a programmer's preferred C compiler and takes care of linking it with the MPI libraries.[24][25]

8.7 Example program

Here is a "Hello World" program in MPI written in C. In this example, we send a "hello" message to each processor, manipulate it trivially, return the results to the main process, and print the messages.

/* "Hello World" MPI Test Program */ #include <mpi.h> #include <stdio.h> #include <string.h> #define BUFSIZE 128 #define TAG 0 int main(int argc, char *argv[]) { char idstr[32]; char buff[BUFSIZE]; int numprocs; int myid; int i; MPI_Status stat; /* MPI programs start with MPI_Init; all 'N' processes exist thereafter */ MPI_Init(&argc,&argv); /* find out how big the SPMD world is */ MPI_Comm_size(MPI_COMM_WORLD,&numprocs); /* and this processes' rank is */ MPI_Comm_rank(MPI_COMM_WORLD,&myid);
/* At this point, all programs are running equivalently, the rank distinguishes the roles of the programs in the SPMD model, with rank 0 often used specially... */ if(myid == 0) { printf("%d: We have %d processors\n", myid, numprocs); for(i=1;i<numprocs;i++) { sprintf(buff, "Hello %d! ", i); MPI_Send(buff, BUFSIZE, MPI_CHAR, i, TAG, MPI_COMM_WORLD); } for(i=1;i<numprocs;i++) { MPI_Recv(buff, BUFSIZE, MPI_CHAR, i, TAG, MPI_COMM_WORLD, &stat); printf("%d: %s\n", myid, buff); } } else { /* receive from rank 0: */ MPI_Recv(buff, BUFSIZE, MPI_CHAR, 0, TAG, MPI_COMM_WORLD, &stat); sprintf(idstr, "Processor %d ", myid); strncat(buff, idstr, BUFSIZE-1); strncat(buff, "reporting for duty", BUFSIZE-1); /* send to rank 0: */ MPI_Send(buff, BUFSIZE, MPI_CHAR, 0, TAG, MPI_COMM_WORLD); } /* MPI programs end with MPI Finalize; this is a weak synchronization point */ MPI_Finalize(); return 0; }

When run with two processors this gives the following output.[26]

0: We have 2 processors 0: Hello 1! Processor 1 reporting for duty

The runtime environment for the MPI implementation used (often called mpirun or mpiexec) spawns multiple copies of the program, with the total number of copies determining the number of process *ranks* in MPI_COMM_WORLD, which is an opaque descriptor for communication between the set of processes. A single process, multiple data (SPMD) programming model is thereby facilitated, but not required; many MPI implementations allow multiple, different, executables to be started in the same MPI job. Each process has its own rank, the total number of processes in the world, and the ability to communicate between them either with point-to-point (send/receive) communication, or by collective communication among the group. It is enough for MPI to provide an SPMD-style program with MPI_COMM_WORLD, its own rank, and the size of the world to allow algorithms to decide what to do. In more realistic situations, I/O is more carefully managed than in this example. MPI does not guarantee how POSIX I/O would actually work on a given system, but it commonly does work, at least from rank 0.

MPI uses the notion of process rather than processor. Program copies are *mapped* to processors by the MPI runtime. In that sense, the parallel machine can map to 1 physical processor, or N where N is the total number of processors available, or something in between. For maximum parallel speedup, more physical processors are used. This example adjusts its behavior to the size of the world N, so it also seeks to scale to the runtime configuration without compilation for each size variation, although runtime decisions might vary depending on that absolute amount of concurrency available.

8.8 MPI-2 adoption

Adoption of MPI-1.2 has been universal, particularly in cluster computing, but acceptance of MPI-2.1 has been more limited. Issues include:

1. MPI-2 implementations include I/O and dynamic process management, and the size of the middleware is substantially larger. Most sites that use batch scheduling systems cannot support dynamic process management. MPI-2's parallel I/O is well accepted.
2. Many MPI-1.2 programs were developed before MPI-2. Portability concerns initially slowed, although wider support has lessened this.
3. Many MPI-1.2 applications use only a subset of that

standard (16-25 functions) with no real need for MPI-2 functionality.

8.9 Future

Some aspects of MPI's future appear solid; others less so. The MPI Forum reconvened in 2007, to clarify some MPI-2 issues and explore developments for a possible MPI-3.

Like Fortran, MPI is ubiquitous in technical computing, and it is taught and used widely.

Architectures are changing, with greater internal concurrency (multi-core), better fine-grain concurrency control (threading, affinity), and more levels of memory hierarchy. Multithreaded programs can take advantage of these developments more easily than single threaded applications. This has already yielded separate, complementary standards for symmetric multiprocessing, namely OpenMP. MPI-2 defines how standard-conforming implementations should deal with multithreaded issues, but does not require that implementations be multithreaded, or even thread safe. Few multithreaded-capable MPI implementations exist. Multi-level concurrency completely within MPI is an opportunity for the standard.

8.10 See also

- MPICH
- Open MPI
- OpenMP
- OpenHMPP HPC Open Standard for Manycore Programming
- Microsoft Messaging Passing Interface
- Global Arrays
- Unified Parallel C
- Co-array Fortran
- occam (programming language)
- Linda (coordination language)
- X10 (programming language)
- Parallel Virtual Machine
- Calculus of communicating systems
- Calculus of Broadcasting Systems
- Actor model
- Allinea DDT Debugging tool for MPI programs
- Allinea MAP Performance profiler for MPI programs
- Bulk Synchronous Parallel BSP Programming
- Partitioned global address space
- Caltech Cosmic Cube

8.11 References

[1] Gropp, Lusk & Skjellum 1996, p. 3

[2] High-performance and scalable MPI over InfiniBand with reduced memory usage

[3] Table of Contents — September 1994, 8 (3-4). Hpc.sagepub.com. Retrieved on 2014-03-24.

[4] MPI Documents. Mpi-forum.org. Retrieved on 2014-03-24.

[5] Gropp, Lusk & Skjellum 1999b, pp. 4–5

[6] MPI: A Message-Passing Interface Standard Version 3.0, Message Passing Interface Forum, September 21, 2012. http://www.mpi-forum.org. Retrieved on 2014-06-28.

[7] Gropp, Lusk & Skjelling 1999b, p. 7

[8] Gropp, Lusk & Skjelling 1999b, pp. 5–6

[9] Sparse matrix-vector multiplications using the MPI I/O library

[10] mpi4py

[11] pypar

[12] Now part of Pydusa

[13] Boost:MPI Python Bindings

[14] OCamlMPI Module

[15] Archives of the Caml mailing list > Message from Yaron M. Minsky. Caml.inria.fr (2003-07-15). Retrieved on 2014-03-24.

[16] mpiJava

[17] mpiJava API

[18] MPJ API

[19] MPJ Express

[20] Yu, H. (2002). "Rmpi: Parallel Statistical Computing in R". *R News*.

[21] Chen, W.-C., Ostrouchov, G., Schmidt, D., Patel, P., and Yu, H. (2012). "pbdMPI: Programming with Big Data -- Interface to MPI".

[22] Pure Mpi.NET

[23] MPI.NET

[24] Woodman, Lawrence. (2009-12-02) Setting up a Beowulf Cluster Using Open MPI on Linux. Techtinkering.com. Retrieved on 2014-03-24.

[25] mpicc. Mpich.org. Retrieved on 2014-03-24.

[26] Using OpenMPI, compiled with gcc -g -v -I/usr/lib/openmpi/include/ -L/usr/lib/openmpi/include/ wiki_mpi_example.c -lmpi and run with mpirun -np 2 ./a.out.

8.12 Further reading

- This article is based on material taken from the Free On-line Dictionary of Computing prior to 1 November 2008 and incorporated under the "relicensing" terms of the GFDL, version 1.3 or later.
- Aoyama, Yukiya; Nakano, Jun (1999) *RS/6000 SP: Practical MPI Programming*, ITSO
- Foster, Ian (1995) *Designing and Building Parallel Programs (Online)* Addison-Wesley ISBN 0-201-57594-9, chapter 8 *Message Passing Interface*
- Viraj B., Wijesuriya 2010-12-29 *Daniweb: Sample Code for Matrix Multiplication using MPI Parallel Programming Approach*
- *Using MPI* series:
  - Gropp, William; Lusk, Ewing; Skjellum, Anthony (1994). *Using MPI: portable parallel programming with the message-passing interface*. Cambridge, MA, USA: MIT Press Scientific And Engineering Computation Series. ISBN 0-262-57104-8.
  - Gropp, William; Lusk, Ewing; Skjellum, Anthony (1999a). *Using MPI, 2nd Edition: Portable Parallel Programming with the Message Passing Interface*. Cambridge, MA, USA: MIT Press Scientific And Engineering Computation Series. ISBN 978-0-262-57132-6.
  - Gropp, William; Lusk, Ewing; Skjellum, Anthony (1999b). *Using MPI-2: Advanced Features of the Message Passing Interface*. MIT Press. ISBN 0-262-57133-1.
- Gropp, William; Lusk, Ewing; Skjellum, Anthony (1996). "A High-Performance, Portable Implementation of the MPI Message Passing Interface". *Parallel Computing*. CiteSeerX: 10.1.1.102.9485.
- Pacheco, Peter S. (1997) *Parallel Programming with MPI*. 500 pp. Morgan Kaufmann ISBN 1-55860-339-5.
- *MPI—The Complete Reference* series:
  - Snir, Marc; Otto, Steve; Huss-Lederman, Steven; Walker, David; Dongarra, Jack (1995) *MPI: The Complete Reference*. MIT Press Cambridge, MA, USA. ISBN 0-262-69215-5
  - M Snir, SW Otto, S Huss-Lederman, DW Walker, J (1998) *MPI—The Complete Reference: Volume 1, The MPI Core*. MIT Press, Cambridge, MA. ISBN 0-262-69215-5
  - Gropp, William; Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Marc Snir (1998) *MPI—The Complete Reference: Volume 2, The MPI-2 Extensions*. MIT Press, Cambridge, MA ISBN 978-0-262-57123-4
- Parallel Processing via MPI & OpenMP, M. Firuziaan, O. Nommensen. Linux Enterprise, 10/2002
- Vanneschi, Marco (1999) *Parallel paradigms for scientific computing* In Proc. of the European School on Computational Chemistry (1999, Perugia, Italy), number 75 in *Lecture Notes in Chemistry*, pages 170–183. Springer, 2000.

8.13 External links

- "MPI Examples - Message Passing Interface". Hakan Haberdar, University of Houston. Retrieved October 2012.
- Message Passing Interface at DMOZ
- Tutorial on MPI: The Message-Passing Interface (PDF)
- A User's Guide to MPI (PDF)

Chapter 9

CUDA

CUDA (after the Plymouth Barracuda[1]), which stands for **Compute Unified Device Architecture**, is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce.[2] CUDA gives developers direct access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs.

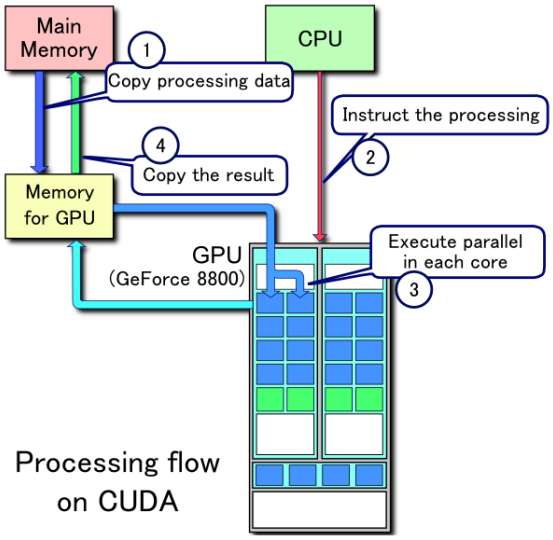
Using CUDA, the GPUs can be used for general purpose processing (i.e., not exclusively graphics); this approach is known as GPGPU. Unlike CPUs, however, GPUs have a parallel throughput architecture that emphasizes executing many concurrent threads slowly, rather than executing a single thread very quickly.

The CUDA platform is accessible to software developers through CUDA-accelerated libraries, compiler directives (such as OpenACC), and extensions to industry-standard programming languages, including C, C++ and Fortran. C/C++ programmers use 'CUDA C/C++', compiled with "nvcc", NVIDIA's LLVM-based C/C++ compiler.[3] Fortran programmers can use 'CUDA Fortran', compiled with the PGI CUDA Fortran compiler from The Portland Group.

In addition to libraries, compiler directives, CUDA C/C++ and CUDA Fortran, the CUDA platform supports other computational interfaces, including the Khronos Group's OpenCL,[4] Microsoft's DirectCompute, OpenGL Compute Shaders and C++ AMP.[5] Third party wrappers are also available for Python, Perl, Fortran, Java, Ruby, Lua, Haskell, R, MATLAB, IDL, and native support in Mathematica.

In the computer game industry, GPUs are used not only for graphics rendering but also in game physics calculations (physical effects such as debris, smoke, fire, fluids); examples include PhysX and Bullet. CUDA has also been used to accelerate non-graphical applications in computational biology, cryptography and other fields by an order of magnitude or more.[6][7][8][9][10]

CUDA provides both a low level API and a higher level API. The initial CUDA SDK was made public on 15 February 2007, for Microsoft Windows and Linux. Mac OS X support was later added in version 2.0,[11] which supersedes the beta released February 14, 2008.[12] CUDA works with all Nvidia GPUs from the G8x series onwards, including GeForce, Quadro and the Tesla line. CUDA is compatible with most standard operating systems. Nvidia states that programs developed for the G8x series will also work without modification on all future Nvidia video cards, due to binary compatibility.



Example of CUDA processing flow
1. *Copy data from main mem to GPU mem*
2. *CPU instructs the process to GPU*
3. *GPU execute parallel in each core*
4. *Copy the result from GPU mem to main mem*

9.1 Background

See also: GPU

The GPU, as a specialized processor, addresses the demands of real-time high-resolution 3D graphics compute-intensive tasks. As of 2012, GPUs have evolved into highly parallel multi-core systems allowing very efficient manipulation of large blocks of data. This design is more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel, such as:

- push-relabel maximum flow algorithm

- fast sort algorithms of large lists
- two-dimensional fast wavelet transform
- molecular dynamics simulations

9.2 Advantages

CUDA has several advantages over traditional general-purpose computation on GPUs (GPGPU) using graphics APIs:

- Scattered reads – code can read from arbitrary addresses in memory
- Unified virtual memory (CUDA 4.0 and above)
- Unified memory (CUDA 6.0 and above)
- Shared memory – CUDA exposes a fast shared memory region that can be shared amongst threads. This can be used as a user-managed cache, enabling higher bandwidth than is possible using texture lookups.[13]
- Faster downloads and readbacks to and from the GPU
- Full support for integer and bitwise operations, including integer texture lookups

9.3 Limitations

- CUDA does not support the full C standard, as it runs host code through a C++ compiler, which makes some valid C (but invalid C++) code fail to compile.[14][15]
- Interoperability with rendering languages such as OpenGL is one-way, with OpenGL having access to registered CUDA memory but CUDA not having access to OpenGL memory.
- Copying between host and device memory may incur a performance hit due to system bus bandwidth and latency (this can be partly alleviated with asynchronous memory transfers, handled by the GPU's DMA engine)
- Threads should be running in groups of at least 32 for best performance, with total number of threads numbering in the thousands. Branches in the program code do not affect performance significantly, provided that each of 32 threads takes the same execution path; the SIMD execution model becomes a significant limitation for any inherently divergent task (e.g. traversing a space partitioning data structure during ray tracing).
- Unlike OpenCL, CUDA-enabled GPUs are only available from Nvidia[16]
- No emulator or fallback functionality is available for modern revisions
- Valid C/C++ may sometimes be flagged and prevent compilation due to optimization techniques the compiler is required to employ to use limited resources.
- A single process must run spread across multiple disjoint memory spaces, unlike other C language runtime environments.
- C++ Run-Time Type Information (RTTI) is not supported in CUDA code, due to lack of support in the underlying hardware.
- Exception handling is not supported in CUDA code due to performance overhead that would be incurred with many thousands of parallel threads running.
- CUDA (with compute capability 2.x) allows a subset of C++ class functionality, for example member functions may not be virtual (this restriction will be removed in some future release). *[See CUDA C Programming Guide 3.1 – Appendix D.6]*
- In single precision on first generation CUDA compute capability 1.x devices, denormal numbers are not supported and are instead flushed to zero, and the precisions of the division and square root operations are slightly lower than IEEE 754-compliant single precision math. Devices that support compute capability 2.0 and above support denormal numbers, and the division and square root operations are IEEE 754 compliant by default. However, users can obtain the previous faster gaming-grade math of compute capability 1.x devices if desired by setting compiler flags to disable accurate divisions, disable accurate square roots, and enable flushing denormal numbers to zero.[17]

9.4 Supported GPUs

Compute capability table (version of CUDA supported) by GPU and card. Also available directly from Nvidia:

'*' - OEM-only products

A table of devices officially supporting CUDA:[16]

9.5 Version features and specifications

For more information please visit this site: http://www.geeks3d.com/20100606/

gpu-computing-nvidia-cuda-compute-capability-comparative-table and also read Nvidia CUDA programming guide.[20]

9.6 Example

This example code in C++ loads a texture from an image into an array on the GPU:

texture<float, 2, cudaReadModeElementType> tex; void foo() { cudaArray* cu_array; // Allocate array cudaChannelFormatDesc description = cudaCreateChannelDesc<float>(); cudaMallocArray(&cu_array, &description, width, height); // Copy image data to array cudaMemcpyToArray(cu_array, image, width*height*sizeof(float), cudaMemcpyHostToDevice); // Set texture parameters (default) tex.addressMode[0] = cudaAddressModeClamp; tex.addressMode[1] = cudaAddressModeClamp; tex.filterMode = cudaFilterModePoint; tex.normalized = false; // do not normalize coordinates // Bind the array to the texture cudaBindTextureToArray(tex, cu_array); // Run kernel dim3 blockDim(16, 16, 1); dim3 gridDim((width + blockDim.x - 1)/ blockDim.x, (height + blockDim.y - 1) / blockDim.y, 1); kernel<<< gridDim, blockDim, 0 >>>(d_data, height, width); // Unbind the array from the texture cudaUnbindTexture(tex); } //end foo() __global__ void kernel(float* odata, int height, int width) { unsigned int x = blockIdx.x*blockDim.x + threadIdx.x; unsigned int y = blockIdx.y*blockDim.y + threadIdx.y; if (x < width && y < height) { float c = tex2D(tex, x, y); odata[y*width+x] = c; } }

Below is an example given in Python that computes the product of two arrays on the GPU. The unofficial Python language bindings can be obtained from *PyCUDA*.[21]

import pycuda.compiler as comp import pycuda.driver as drv import numpy import pycuda.autoinit mod = comp.SourceModule(""" __global__ void multiply_them(float *dest, float *a, float *b) { const int i = threadIdx.x; dest[i] = a[i] * b[i]; } """) multiply_them = mod.get_function(“multiply_them”) a = numpy.random.randn(400).astype(numpy.float32) b = numpy.random.randn(400).astype(numpy.float32) dest = numpy.zeros_like(a) multiply_them(drv.Out(dest), drv.In(a), drv.In(b), block=(400,1,1)) print dest-a*b

Additional Python bindings to simplify matrix multiplication operations can be found in the program *pycublas*.[22]

import numpy from pycublas import CUBLASMatrix A = CUBLASMatrix(numpy.mat([[1,2,3]],[[4,5,6]],numpy.float32)) B = CUBLASMatrix(numpy.mat([[2,3]],[4,5],[[6,7]],numpy.float32)) C = A*B print C.np_mat()

9.7 Language bindings

- Common Lisp - cl-cuda
- Fortran – FORTRAN CUDA, PGI CUDA Fortran Compiler
- F# - Alea.CUDA
- Haskell – Data.Array.Accelerate
- IDL – GPULib
- Java – jCUDA, JCuda, JCublas, JCufft, CUDA4J
- Lua – KappaCUDA
- Mathematica – CUDALink
- MATLAB – Parallel Computing Toolbox, MATLAB Distributed Computing Server,[23] and 3rd party packages like Jacket.
- .NET – CUDA.NET, Managed CUDA, CUDAfy.NET .NET kernel and host code, CURAND, CUBLAS, CUFFT
- Perl – KappaCUDA, CUDA::Minimal
- Python – Numba, NumbaPro, PyCUDA, KappaCUDA, Theano
- Ruby – KappaCUDA
- R – gputools

9.8 Current and future usages of CUDA architecture

- Accelerated rendering of 3D graphics
- Accelerated interconversion of video file formats
- Accelerated encryption, decryption and compression
- Distributed calculations, such as predicting the native conformation of proteins
- Medical analysis simulations, for example virtual reality based on CT and MRI scan images.
- Physical simulations, in particular in fluid dynamics.
- Neural network training in machine learning problems
- Distributed computing
- Molecular dynamics
- Mining cryptocurrencies

9.9 See also

- Allinea DDT - A debugger for CUDA, OpenACC, and parallel applications
- OpenCL - A standard for programming a variety of platforms, including GPUs
- BrookGPU – the Stanford University graphics group's compiler
- Array programming
- Parallel computing
- Stream processing
- rCUDA – An API for computing on remote computers
- Molecular modeling on GPU

9.10 References

[1] [Mark Ebersole, Nvidea educator, 2012 presentation]

[2] NVIDIA CUDA Home Page

[3] CUDA LLVM Compiler

[4] First OpenCL demo on a GPU on YouTube

[5] DirectCompute Ocean Demo Running on Nvidia CUDA-enabled GPU on YouTube

[6] Giorgos Vasiliadis, Spiros Antonatos, Michalis Polychronakis, Evangelos P. Markatos and Sotiris Ioannidis (September 2008). "Gnort: High Performance Network Intrusion Detection Using Graphics Processors" (PDF). *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID)*.

[7] Schatz, M.C., Trapnell, C., Delcher, A.L., Varshney, A. (2007). "High-throughput sequence alignment using Graphics Processing Units". *BMC Bioinformatics*. 8:474: 474. doi:10.1186/1471-2105-8-474. PMC 2222658. PMID 18070356.

[8] Manavski, Svetlin A.; Giorgio Valle (2008). "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment". *BMC Bioinformatics* **9**: S10. doi:10.1186/1471-2105-9-S2-S10. PMC 2323659. PMID 18387198.

[9] Pyrit – Google Code http://code.google.com/p/pyrit/

[10] Use your Nvidia GPU for scientific computing, BOINC official site (December 18, 2008)

[11] Nvidia CUDA Software Development Kit (CUDA SDK) – Release Notes Version 2.0 for MAC OS X

[12] CUDA 1.1 – Now on Mac OS X- (Posted on Feb 14, 2008)

[13] Silberstein, Mark; Schuster, Assaf; Geiger, Dan; Patney, Anjul; Owens, John D. (2008). "Proceedings of the 22nd annual international conference on Supercomputing - ICS '08". pp. 309–318. doi:10.1145/1375527.1375572. ISBN 978-1-60558-158-3. |chapter= ignored (help)

[14] NVCC forces c++ compilation of .cu files

[15] C++ keywords on CUDA C code

[16] "CUDA-Enabled Products". *CUDA Zone*. Nvidia Corporation. Retrieved 2008-11-03.

[17] Whitehead, Nathan; Fit-Florea, Alex. "Precision & Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPUs". Nvidia. Retrieved November 18, 2014.

[18] Cores perform only single-precision floating-point arithmetics. There is 1 double-precision floating-point unit.

[19] No more than one scheduler can issue 2 instructions at once. The first scheduler is in charge of the warps with an odd ID and the second scheduler is in charge of the warps with an even ID.

[20] Appendix F. Features and Technical Specifications PDF (3.2 MiB), Page 148 of 175 (Version 5.0 October 2012)

[21] PyCUDA

[22] pycublas

[23] "MATLAB Adds GPGPU Support". 2010-09-20.

9.11 External links

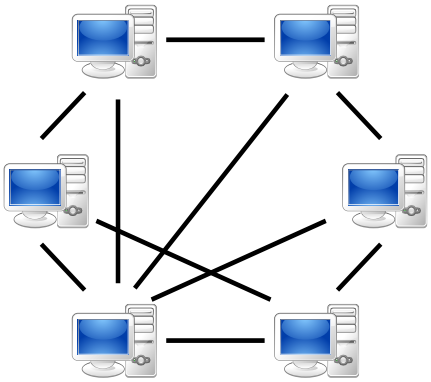
- Official website
- CUDA Community on Google+
- A little tool to adjust the VRAM size

Chapter 10

Peer-to-peer

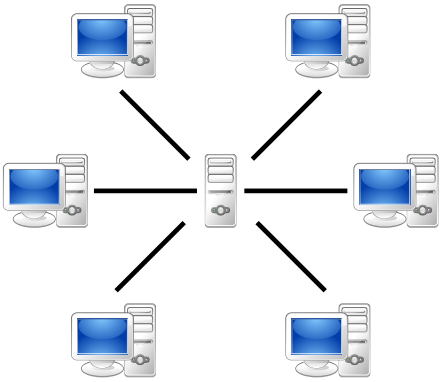
Not to be confused with Point-to-point (telecommunications).
This article is about peer-to-peer computer networks. For other uses, see Peer-to-peer (disambiguation).

Peer-to-peer (**P2P**) computing or networking is a distributed application architecture that partitions tasks or work loads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.



*A **peer-to-peer (P2P) network** in which interconnected nodes ("peers") share resources amongst each other without the use of a centralized administrative system*

Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.[1] Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model in which the consumption and supply of resources is divided. Emerging collaborative P2P systems are going beyond the era of peers doing similar things while sharing resources, and are looking for diverse peers that can bring in unique resources and capabilities to a virtual community thereby empowering it to engage in greater tasks beyond those that can be accomplished by individual peers, yet that are beneficial to all the peers.[2]



*A network based on the **client-server model**, where individual clients request services and resources from centralized servers*

While P2P systems had previously been used in many application domains,[3] the architecture was popularized by the file sharing system Napster, originally released in 1999. The concept has inspired new structures and philosophies in many areas of human interaction. In such social contexts, peer-to-peer as a meme refers to the egalitarian social networking that has emerged throughout society, enabled by Internet technologies in general.

10.1 Historical development

While P2P systems had previously been used in many application domains,[3] the concept was popularized by file sharing systems such as the music-sharing application Napster (originally released in 1999). The peer-to-peer movement allowed millions of Internet users to connect "directly, forming groups and collaborating to become user-created search engines, virtual supercomputers, and

filesystems." [4] The basic concept of peer-to-peer computing was envisioned in earlier software systems and networking discussions, reaching back to principles stated in the first Request for Comments, RFC 1.[5]

Tim Berners-Lee's vision for the World Wide Web was close to a P2P network in that it assumed each user of the web would be an active editor and contributor, creating and linking content to form an interlinked "web" of links. The early Internet was more open than present day, where two machines connected to the Internet could send packets to each other without firewalls and other security measures.[4] This contrasts to the broadcasting-like structure of the web as it has developed over the years.[6] As a precursor to the Internet, ARPANET was a successful client-server network where "every participating node could request and serve content." However, ARPANET was not self-organized and it lacked the ability to "provide any means for context or content based routing beyond 'simple' addressed based routing."[7]

Therefore, a distributed messaging system that is often likened as an early peer-to-peer architecture was established: USENET. USENET was developed in 1979 and is a system that enforces a decentralized model of control. The basic model is a client-server model from the user or client perspective that offers a self-organizing approach to newsgroup servers. However, news servers communicate with one another as peers to propagate Usenet news articles over the entire group of network servers. The same consideration applies to SMTP email in the sense that the core email relaying network of Mail transfer agents has a peer-to-peer character, while the periphery of e-mail clients and their direct connections is strictly a client–server relationship.

In May 1999, with millions more people on the Internet, Shawn Fanning introduced the music and file-sharing application called Napster.[7] Napster was the beginning of peer-to-peer networks, as we know them today, where "participating users establish a virtual network, entirely independent from the physical network, without having to obey any administrative authorities or restrictions."[7]

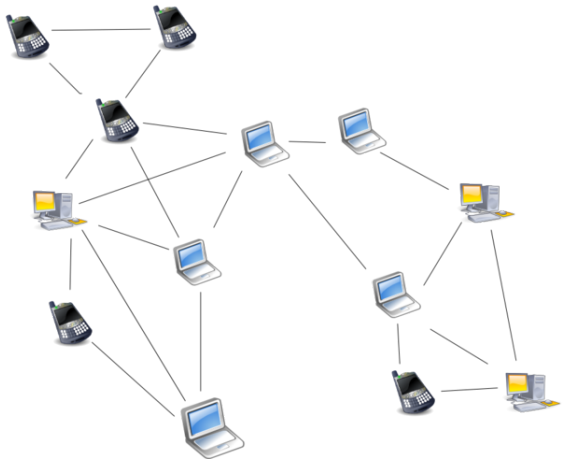
10.2 Architecture

A peer-to-peer network is designed around the notion of equal *peer* nodes simultaneously functioning as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the client–server model where communication is usually to and from a central server. A typical example of a file transfer that uses the client-server model is the File Transfer Protocol (FTP) service in which the client and server programs are distinct: the clients initiate the transfer, and the servers satisfy these requests.

10.2.1 Routing and resource discovery

Peer-to-peer networks generally implement some form of virtual overlay network on top of the physical network topology, where the nodes in the overlay form a subset of the nodes in the physical network. Data is still exchanged directly over the underlying TCP/IP network, but at the application layer peers are able to communicate with each other directly, via the logical overlay links (each of which corresponds to a path through the underlying physical network). Overlays are used for indexing and peer discovery, and make the P2P system independent from the physical network topology. Based on how the nodes are linked to each other within the overlay network, and how resources are indexed and located, we can classify networks as *unstructured* or *structured* (or as a hybrid between the two).[8][9][10]

Unstructured networks



*Overlay network diagram for an **unstructured P2P network**, illustrating the ad hoc nature of the connections between nodes*

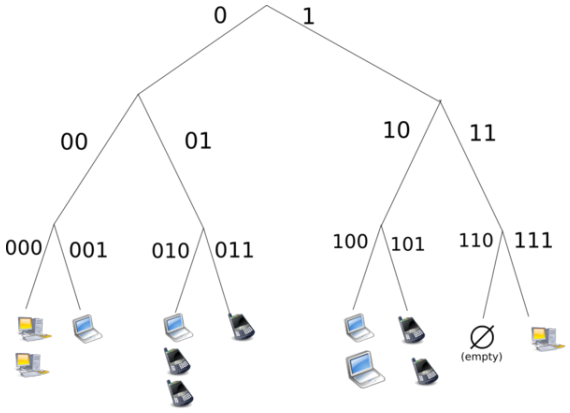
Unstructured peer-to-peer networks do not impose a particular structure on the overlay network by design, but rather are formed by nodes that randomly form connections to each other.[11] (Gnutella, Gossip, and Kazaa are examples of unstructured P2P protocols).[12]

Because there is no structure globally imposed upon them, unstructured networks are easy to build and allow for localized optimizations to different regions of the overlay.[13] Also, because the role of all peers in the network is the same, unstructured networks are highly robust in the face of high rates of "churn"—that is, when large numbers of peers are frequently joining and leaving the network.[14][15]

However the primary limitations of unstructured networks also arise from this lack of structure. In particular, when a peer wants to find a desired piece of data in the network, the search query must be flooded through the network to find as many peers as possible that share the

data. Flooding causes a very high amount of signaling traffic in the network, uses more CPU/memory (by requiring every peer to process all search queries), and does not ensure that search queries will always be resolved. Furthermore, since there is no correlation between a peer and the content managed by it, there is no guarantee that flooding will find a peer that has the desired data. Popular content is likely to be available at several peers and any peer searching for it is likely to find the same thing. But if a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful.[16]

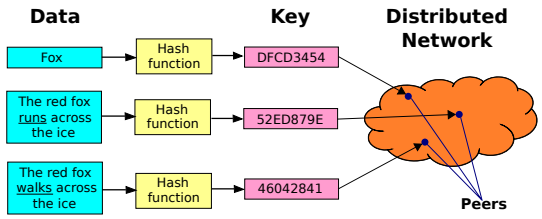
Structured networks



*Overlay network diagram for a **structured P2P network**, using a distributed hash table (DHT) to identify and locate nodes/resources*

In *structured peer-to-peer networks* the overlay is organized into a specific topology, and the protocol ensures that any node can efficiently[17] search the network for a file/resource, even if the resource is extremely rare.

The most common type of structured P2P networks implement a distributed hash table (DHT),[18][19] in which a variant of consistent hashing is used to assign ownership of each file to a particular peer.[20][21] This enables peers to search for resources on the network using a hash table: that is, (*key*, *value*) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key.[22][23]



Distributed hash tables

However, in order to route traffic efficiently through the network, nodes in a structured overlay must maintain lists of neighbors that satisfy specific criteria. This makes them less robust in networks with a high rate of *churn* (i.e. with large numbers of nodes frequently joining and leaving the network).[15][24] More recent evaluation of P2P resource discovery solutions under real workloads have pointed out several issues in DHT-based solutions such as high cost of advertising/discovering resources and static and dynamic load imbalance.[25]

Notable distributed networks that use DHTs include BitTorrent's distributed tracker, the Kad network, the Storm botnet, YaCy, and the Coral Content Distribution Network. Some prominent research projects include the Chord project, Kademlia, PAST storage utility, P-Grid, a self-organized and emerging overlay network, and CoopNet content distribution system. DHT-based networks have also been widely utilized for accomplishing efficient resource discovery[26][27] for grid computing systems, as it aids in resource management and scheduling of applications.

Hybrid models

Hybrid models are a combination of peer-to-peer and client-server models.[28] A common hybrid model is to have a central server that helps peers find each other. Spotify is an example of a hybrid model. There are a variety of hybrid models, all of which make tradeoffs between the centralized functionality provided by a structured server/client network and the node equality afforded by the pure peer-to-peer unstructured networks. Currently, hybrid models have better performance than either pure unstructured networks or pure structured networks because certain functions, such as searching, do require a centralized functionality but benefit from the decentralized aggregation of nodes provided by unstructured networks.[29]

10.2.2 Security and trust

Peer-to-peer systems pose unique challenges from a computer security perspective.

Like any other form of software, P2P applications can contain vulnerabilities. What makes this particularly dangerous for P2P software, however, is that peer-to-peer applications act as servers as well as clients, meaning that they can be more vulnerable to remote exploits.[30]

Routing attacks

Also, since each node plays a role in routing traffic through the network, malicious users can perform a variety of "routing attacks", or denial of service attacks. Examples of common routing attacks include "incorrect lookup routing" whereby malicious nodes deliberately forward requests incorrectly or return false results, "in-

correct routing updates" where malicious nodes corrupt the routing tables of neighboring nodes by sending them false information, and "incorrect routing network partition" where when new nodes are joining they bootstrap via a malicious node, which places the new node in a partition of the network that is populated by other malicious nodes.[31]

Corrupted data and malware

See also: Data validation and Malware

The prevalence of malware varies between different peer-to-peer protocols. Studies analyzing the spread of malware on P2P networks found, for example, that 63% of the answered download requests on the Limewire network contained some form of malware, whereas only 3% of the content on OpenFT contained malware. In both cases, the top three most common types of malware accounted for the large majority of cases (99% in Limewire, and 65% in OpenFT). Another study analyzing traffic on the Kazaa network found that 15% of the 500,000 file sample taken were infected by one or more of the 365 different computer viruses that were tested for.[32]

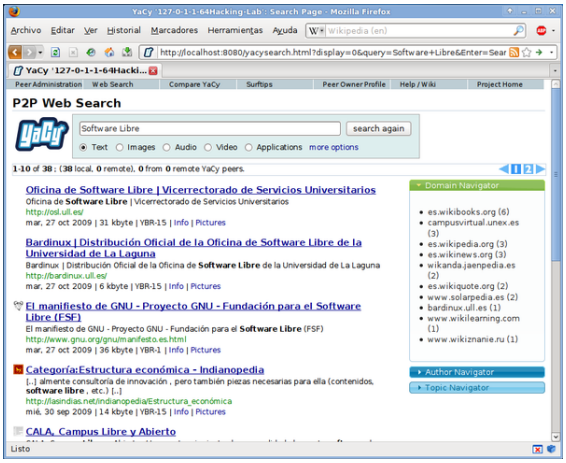
Corrupted data can also be distributed on P2P networks by modifying files that are already being shared on the network. For example, on the FastTrack network, the RIAA managed to introduce faked chunks into downloads and downloaded files (mostly MP3 files). Files infected with the RIAA virus were unusable afterwards and contained malicious code. The RIAA is also known to have uploaded fake music and movies to P2P networks in order to deter illegal file sharing.[33] Consequently, the P2P networks of today have seen an enormous increase of their security and file verification mechanisms. Modern hashing, chunk verification and different encryption methods have made most networks resistant to almost any type of attack, even when major parts of the respective network have been replaced by faked or nonfunctional hosts.[34]

10.2.3 Resilient and scalable computer networks

See also: Wireless mesh network and Distributed computing

The decentralized nature of P2P networks increases robustness because it removes the single point of failure that can be inherent in a client-server based system.[35] As nodes arrive and demand on the system increases, the total capacity of the system also increases, and the likelihood of failure decreases. If one peer on the network fails to function properly, the whole network is not compromised or damaged. In contrast, in a typical client–server architecture, clients share only their demands with the system, but not their resources. In this case, as more clients join the system, fewer resources are available to serve each client, and if the central server fails, the entire network is taken down.

10.2.4 Distributed storage and search



Search results for the query "software libre", using YaCy a free distributed search engine that runs on a peer-to-peer network instead making requests to centralized index servers (like Google, Yahoo, and other corporate search engines)

There are both advantages and disadvantages in P2P networks related to the topic of data backup, recovery, and availability. In a centralized network, the system administrators are the only forces controlling the availability of files being shared. If the administrators decide to no longer distribute a file, they simply have to remove it from their servers, and it will no longer be available to users. Along with leaving the users powerless in deciding what is distributed throughout the community, this makes the entire system vulnerable to threats and requests from the government and other large forces. For example, YouTube has been pressured by the RIAA, MPAA, and entertainment industry to filter out copyrighted content. Although server-client networks are able to monitor and manage content availability, they can have more stability in the availability of the content they choose to host. A client should not have trouble accessing obscure content that is being shared on a stable centralized network. P2P networks, however, are more unreliable in sharing unpopular files because sharing files in a P2P network requires that at least one node in the network has the requested data, and that node must be able to connect to the node requesting the data. This requirement is occasionally hard to meet because users may delete or stop sharing data at any point.[36]

In this sense, the community of users in a P2P network is completely responsible for deciding what content is available. Unpopular files will eventually disappear and become unavailable as more people stop sharing them. Pop-

ular files, however, will be highly and easily distributed. Popular files on a P2P network actually have more stability and availability than files on central networks. In a centralized network a simple loss of connection between the server and clients is enough to cause a failure, but in P2P networks the connections between every node must be lost in order to cause a data sharing failure. In a centralized system, the administrators are responsible for all data recovery and backups, while in P2P systems, each node requires its own backup system. Because of the lack of central authority in P2P networks, forces such as the recording industry, RIAA, MPAA, and the government are unable to delete or stop the sharing of content on P2P systems.[37]

10.3 Applications

10.3.1 Content delivery

In P2P networks, clients both provide and use resources. This means that unlike client-server systems, the content serving capacity of peer-to-peer networks can actually *increase* as more users begin to access the content (especially with protocols such as Bittorrent that require users to share, refer a performance measurement study[38]). This property is one of the major advantages of using P2P networks because it makes the setup and running costs very small for the original content distributor.[39][40]

10.3.2 File-sharing networks

Many file peer-to-peer file sharing networks, such as Gnutella, G2, and the eDonkey network popularized peer-to-peer technologies.

- Peer-to-peer content delivery networks.
- Peer-to-peer content services, e.g. caches for improved performance such as Correli Caches[41]
- Software publication and distribution (Linux distribution, several games); via file sharing networks.

Copyright infringements

Peer-to-peer networking involves data transfer from one user to another without using an intermediate server. Companies developing P2P applications have been involved in numerous legal cases, primarily in the United States, over conflicts with copyright law.[42] Two major cases are *Grokster vs RIAA* and *MGM Studios, Inc. v. Grokster, Ltd.*.[43] In both of the cases the file sharing technology was ruled to be legal as long as the developers had no ability to prevent the sharing of the copyrighted material.

10.3.3 Multimedia

- The P2PTV and PDTP protocols.
- Some proprietary multimedia applications, such as Skype and Spotify, use a peer-to-peer network along with streaming servers to stream audio and video to their clients.
- Peercasting for multicasting streams.
- Pennsylvania State University, MIT and Simon Fraser University are carrying on a project called LionShare designed for facilitating file sharing among educational institutions globally.
- Osiris is a program that allows its users to create anonymous and autonomous web portals distributed via P2P network.

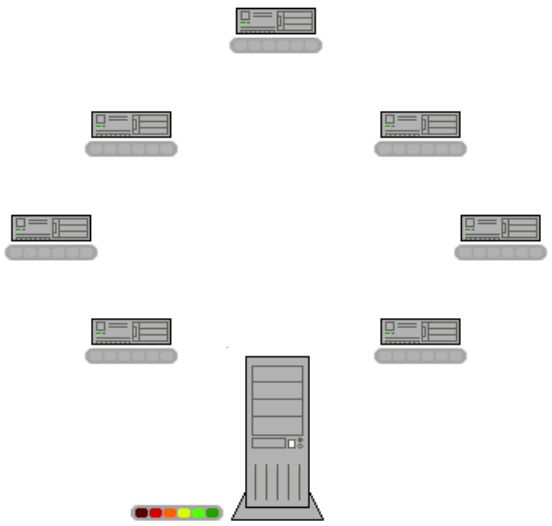
10.3.4 Other P2P applications

- Tradepal and M-commerce applications that power real-time marketplaces.
- Bitcoin and alternatives such as Peercoin and Nxt are peer-to-peer-based digital cryptocurrencies.
- I2P, an overlay network used to browse the Internet anonymously.
- Infinit is an unlimited and encrypted peer to peer file sharing application for digital artists written in C++.
- Netsukuku, a Wireless community network designed to be independent from the Internet.
- Dalesa, a peer-to-peer web cache for LANs (based on IP multicasting).
- Open Garden, connection sharing application that shares Internet access with other devices using Wi-Fi or Bluetooth.
- Research like the Chord project, the PAST storage utility, the P-Grid, and the CoopNet content distribution system.
- JXTA, a peer-to-peer protocol designed for the Java platform.
- Midpoint and CurrencyFair are peer-to-peer foreign currency exchange marketplace.
- The U.S. Department of Defense is conducting research on P2P networks as part of its modern network warfare strategy.[44] In May, 2003, Anthony Tether, then director of DARPA, testified that the U.S. military uses P2P networks.

10.4 Social implications

See also: Social peer-to-peer processes

10.4.1 Incentivizing resource sharing and cooperation



***The BitTorrent protocol**: In this animation, the colored bars beneath all of the 7 clients in the upper region above represent the file being shared, with each color representing an individual piece of the file. After the initial pieces transfer from the seed (large system at the bottom), the pieces are individually transferred from client to client. The original seeder only needs to send out one copy of the file for all the clients to receive a copy.*

Cooperation among a community of participants is key to the continued success of P2P systems aimed at casual human users; these reach their full potential only when large numbers of nodes contribute resources. But in current practice P2P networks often contain large numbers of users who utilize resources shared by other nodes, but who do not share anything themselves (often referred to as the "freeloader problem"). Freeloading can have a profound impact on the network and in some cases can cause the community to collapse.[45] In these types of networks "users have natural disincentives to cooperate because cooperation consumes their own resources and may degrade their own performance." [46] Studying the social attributes of P2P networks is challenging due to large populations of turnover, asymmetry of interest and zero-cost identity.[46] A variety of incentive mechanisms have been implemented to encourage or even force nodes to contribute resources.[47]

Some researchers have explored the benefits of enabling virtual communities to self-organize and introduce incentives for resource sharing and cooperation, arguing that the social aspect missing from today's P2P systems should be seen both as a goal and a means for self-organized virtual communities to be built and fostered.[48] Ongoing research efforts for designing effective incentive mechanisms in P2P systems, based on principles from game theory, are beginning to take on a more psychological and information-processing direction.

Privacy and anonymity

Some peer-to-peer networks (e.g. Freenet) place a heavy emphasis on privacy and anonymity—that is, ensuring that the contents of communications are hidden from eavesdroppers, and that the identities/locations of the participants are concealed. Public key cryptography can be used to provide encryption, data validation, authorization, and authentication for data/messages. Onion routing and other mix network protocols (e.g. Tarzan) can be used to provide anonymity.[49]

10.5 Political implications

10.5.1 Intellectual property law and illegal sharing

Although peer-to-peer networks can be used for legitimate purposes, rights holders have targeted peer-to-peer over the involvement with sharing copyrighted material. Peer-to-peer networking involves data transfer from one user to another without using an intermediate server. Companies developing P2P applications have been involved in numerous legal cases, primarily in the United States, primarily over issues surrounding copyright law.[42] Two major cases are *Grokster vs RIAA* and *MGM Studios, Inc. v. Grokster, Ltd.*.[43] In both of the cases the file sharing technology was ruled to be legal as long as the developers had no ability to prevent the sharing of the copyrighted material. To establish criminal liability for the copyright infringement on peer-to-peer systems, the government must prove that the defendant infringed a copyright willingly for the purpose of personal financial gain or commercial advantage.[50] Fair use exceptions allow limited use of copyrighted material to be downloaded without acquiring permission from the rights holders. These documents are usually news reporting or under the lines of research and scholarly work. Controversies have developed over the concern of illegitimate use of peer-to-peer networks regarding public safety and national security. When a file is downloaded through a peer-to-peer network, it is impossible to know who created the file or what users are connected to the network at a given time. Trustworthiness of sources is a potential security threat that can be seen with peer-to-peer systems.[51]

10.5.2 Network neutrality

Peer-to-peer applications present one of the core issues in the network neutrality controversy. Internet service providers (ISPs) have been known to throttle P2P file-sharing traffic due to its high-bandwidth usage.[52] Compared to Web browsing, e-mail or many other uses of the internet, where data is only transferred in short intervals and relative small quantities, P2P file-sharing often consists of relatively heavy bandwidth usage due to ongoing file transfers and swarm/network coordination packets. In October 2007, Comcast, one of the largest broadband Internet providers in the USA, started blocking P2P applications such as BitTorrent. Their rationale was that P2P is mostly used to share illegal content, and their infrastructure is not designed for continuous, high-bandwidth traffic. Critics point out that P2P networking has legitimate legal uses, and that this is another way that large providers are trying to control use and content on the Internet, and direct people towards a client-server-based application architecture. The client-server model provides financial barriers-to-entry to small publishers and individuals, and can be less efficient for sharing large files. As a reaction to this bandwidth throttling, several P2P applications started implementing protocol obfuscation, such as the BitTorrent protocol encryption. Techniques for achieving "protocol obfuscation" involves removing otherwise easily identifiable properties of protocols, such as deterministic byte sequences and packet sizes, by making the data look as if it were random.[53] The ISP's solution to the high bandwidth is P2P caching, where an ISP stores the part of files most accessed by P2P clients in order to save access to the Internet.

10.6 Current research

Researchers have used computer simulations to aid in understanding and evaluating the complex behaviors of individuals within the network. "Networking research often relies on simulation in order to test and evaluate new ideas. An important requirement of this process is that results must be reproducible so that other researchers can replicate, validate, and extend existing work." [54] If the research cannot be reproduced, then the opportunity for further research is hindered. "Even though new simulators continue to be released, the research community tends towards only a handful of open-source simulators. The demand for features in simulators, as shown by our criteria and survey, is high. Therefore, the community should work together to get these features in open-source software. This would reduce the need for custom simulators, and hence increase repeatability and reputability of experiments." [54]

10.7 See also

- Client–queue–client
- Cultural-Historical Activity Theory (CHAT)
- Decentralized computing
- Friend-to-friend
- List of P2P protocols
- Segmented downloading
- Semantic P2P networks
- Sharing economy
- Wireless ad hoc network
- USB dead drop

10.8 References

[1] Rüdiger Schollmeier, *A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications*, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002).

[2] Bandara, H. M. N. D; A. P. Jayasumana (2012). "Collaborative Applications over Peer-to-Peer Systems – Challenges and Solutions". *Peer-to-Peer Networking and Applications*. doi:10.1007/s12083-012-0157-3.

[3] D. Barkai, *Peer-to-Peer Computing*, Intel Press, 2002.

[4] Oram, A. (Ed.). (2001). Peer-to-peer: Harnessing the Benefits of a Disruptive Technologies. O'Reilly Media, Inc.

[5] RFC 1, *Host Software*, S. Crocker, IETF Working Group (April 7, 1969)

[6] Berners-Lee, Tim (August 1996). "The World Wide Web: Past, Present and Future". Retrieved 5 November 2011.</hat Is This "Peer-to-Peer" About? (pp. 9-16). Springer Berlin Heidelberg.

[7] Steinmetz, R., & Wehrle, K. (2005). 2. What Is This "Peer-to-Peer" About? (pp. 9-16). Springer Berlin Heidelberg.

[8] Ahson, Syed A.; Ilyas, Mohammad, eds. (2008). *SIP Handbook: Services, Technologies, and Security of Session Initiation Protocol*. Taylor & Francis. p. 204. ISBN 9781420066043.

[9] Zhu, Ce; et al., eds. (2010). *Streaming Media Architectures: Techniques and Applications: Recent Advances*. IGI Global. p. 265. ISBN 9781616928339.

[10] Kamel, Mina; et al. (2007). "Optimal Topology Design for Overlay Networks". In Akyildiz, Ian F. *Networking 2007: Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet: 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007 Proceedings*. Springer. p. 714. ISBN 9783540726050.

[11] Filali, Imen; et al. (2011). "A Survey of Structured P2P Systems for RDF Data Storage and Retrieval". In Hameurlain, Abdelkader; et al. *Transactions on Large-Scale Data- and Knowledge-Centered Systems III: Special Issue on Data and Knowledge Management in Grid and P2P Systems*. Springer. p. 21. ISBN 9783642230738.

[12] Zulhasnine, Mohammed; et al. (2013). "P2P Streaming Over Cellular Networks: Issues, Challenges, and Opportunities". In Pathan; et al. *Building Next-Generation Converged Networks: Theory and Practice*. CRC Press. p. 99. ISBN 9781466507616.

[13] Chervenak, Ann; Bharathi, Shishir (2008). "Peer-to-peer Approaches to Grid Resource Discovery". In Danelutto, Marco; et al. *Making Grids Work: Proceedings of the CoreGRID Workshop on Programming Models Grid and P2P System Architecture Grid Systems, Tools and Environments 12-13 June 2007, Heraklion, Crete, Greece*. Springer. p. 67. ISBN 9780387784489.

[14] Jin, Xing; Chan, S.-H. Gary (2010). "Unstructured Peer-to-Peer Network Architectures". In Shen; et al. *Handbook of Peer-to-Peer Networking*. Springer. p. 119. ISBN 978-0-387-09750-3.

[15] Lv, Qin; et al. (2002). "Can Heterogenity Make Gnutella Stable?". In Druschel, Peter; et al. *Peer-to-Peer Systems: First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*. Springer. p. 94. ISBN 9783540441793.

[16] Shen, Xuemin; Yu, Heather; Buford, John; Akon, Mursalin (2009). *Handbook of Peer-to-Peer Networking* (1st ed.). New York: Springer. p. 118. ISBN 0-387-09750-3.

[17] Typically approximating O(log N), where N is the number of nodes in the P2P system

[18] Other design choices include overlay rings and d-Torus. See for example Bandara, H. M. N. D.; Jayasumana, A. P. (2012). "Collaborative Applications over Peer-to-Peer Systems – Challenges and Solutions". *Peer-to-Peer Networking and Applications* **6** (3): 257. doi:10.1007/s12083-012-0157-3.

[19] R. Ranjan, A. Harwood, and R. Buyya, "Peer-to-peer based resource discovery in global grids: a tutorial," *IEEE Commun. Surv.*, vol. 10, no. 2. and P. Trunfio, "Peer-to-Peer resource discovery in Grids: Models and systems," *Future Generation Computer Systems* archive, vol. 23, no. 7, Aug. 2007.

[20] Kelaskar, M.; Matossian, V.; Mehra, P.; Paul, D.; Parashar, M. (2002). "A Study of Discovery Mechanisms for Peer-to-Peer Application"{{inconsistent citations}}

[21] Dabek, Frank; Zhao, Ben; Druschel, Peter; Kubiatowicz, John; Stoica, Ion (2003). "Towards a Common API for Structured Peer-to-Peer Overlays". *Peer-to-Peer Systems II*. Lecture Notes in Computer Science **2735**: 33–44. doi:10.1007/978-3-540-45172-3_3. ISBN 978-3-540-40724-9.

[22] Moni Naor and Udi Wieder. Novel Architectures for P2P Applications: the Continuous-Discrete Approach. Proc. SPAA, 2003.

[23] Gurmeet Singh Manku. Dipsea: A Modular Distributed Hash Table. Ph. D. Thesis (Stanford University), August 2004.

[24] Li, Deng; et al. (2009). Vasilakos, A.V.; et al., eds. *An Efficient, Scalable, and Robust P2P Overlay for Autonomic Communication*. Springer. p. 329. ISBN 978-0-387-09752-7.

[25] Bandara, H. M. N. Dilum; Jayasumana, Anura P. (January 2012). "Evaluation of P2P Resource Discovery Architectures Using Real-Life Multi-Attribute Resource and Query Characteristics". *IEEE Consumer Communications and Networking Conf. (CCNC '12)*.

[26] Ranjan, Rajiv; Harwood, Aaron; Buyya, Rajkumar (1 December 2006). "A Study on Peer-to-Peer Based Discovery of Grid Resource Information" (PDF){{inconsistent citations}}

[27] Ranjan, Rajiv; Chan, Lipo; Harwood, Aaron; Karunasekera, Shanika; Buyya, Rajkumar. "Decentralised Resource Discovery Service for Large Scale Federated Grids" (PDF).

[28] Darlagiannis, Vasilios (2005). "Hybrid Peer-to-Peer Systems". In Wehrle, Klaus. *Peer-to-Peer Systems and Applications*. Springer. ISBN 9783540291923.

[29] Yang, Beverly; Garcia-Molina, Hector (2001). "Comparing Hybrid Peer-to-Peer Systems" (PDF). *Very Large Data Bases*. Retrieved 8 October 2013.

[30] Vu, Quang H.; et al. (2010). *Peer-to-Peer Computing: Principles and Applications*. Springer. p. 8. ISBN 978-3-642-03513-5.

[31] Vu, Quang H.; et al. (2010). *Peer-to-Peer Computing: Principles and Applications*. Springer. pp. 157–159. ISBN 978-3-642-03513-5.

[32] Goebel, Jan; et al. (2007). "Measurement and Analysis of Autonomous Spreading Malware in a University Environment". In Hämmerli, Bernhard Markus; Sommer, Robin. *Detection of Intrusions and Malware, and Vulnerability Assessment: 4th International Conference, DIMVA 2007 Lucerne, Switzerland, July 12-13, 2007 Proceedings*. Springer. p. 112. ISBN 9783540736134.

[33] Sorkin, Andrew Ross (4 May 2003). "Software Bullet Is Sought to Kill Musical Piracy". New York Times. Retrieved 5 November 2011.

[34] Singh, Vivek; Gupta, Himani (2012). *Anonymous File Sharing in Peer to Peer System by Random Walks* (Technical report). SRM University. 123456789/9306.

[35] Lua, Eng Keong; Crowcroft, Jon; Pias, Marcelo; Sharma, Ravi; Lim, Steven (2005). "A survey and comparison of peer-to-peer overlay network schemes".

[36] Balakrishnan, Hari; Kaashoek, M. Frans; Karger, David; Morris, Robert; Stoica, Ion (2003). "Looking up data in P2P systems". *Communications of the ACM* **46** (2): 43–48. doi:10.1145/606272.606299. Retrieved 8 October 2013.

[37] "Art thou a Peer?". *www.p2pnews.net*. 14 June 2012. Retrieved 10 October 2013.

[38] Sharma P., Bhakuni A. & Kaushal R."Performance Analyis of BitTorrent Protocol. National Conference on Communications, 2013 doi:10.1109/NCC.2013.6488040

[39] Li, Jin (2008). "On peer-to-peer (P2P) content delivery" (PDF). *Peer-to-Peer Networking and Applications* **1** (1): 45–63. doi:10.1007/s12083-007-0003-1.

[40] Stutzbach, Daniel; et al. (2005). "The scalability of swarming peer-to-peer content delivery". In Boutaba, Raouf; et al. *NETWORKING 2005 -- Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems* (PDF). Springer. pp. 15–26. ISBN 978-3-540-25809-4.

[41] Gareth Tyson, Andreas Mauthe, Sebastian Kaune, Mu Mu and Thomas Plagemann. Corelli: A Dynamic Replication Service for Supporting Latency-Dependent Content in Community Networks. In Proc. 16th ACM/SPIE Multimedia Computing and Networking Conference (MMCN), San Jose, CA (2009).

[42] Glorioso, Andrea; et al. (2010). "The Social Impact of P2P Systems". In Shen; et al. *Handbook of Peer-to-Peer Networking*. Springer. p. 48. ISBN 978-0-387-09750-3.

[43] John Borland, Judge: File-Swapping Tools are Legal , http://news.cnet.com/Judge-File-swapping-tools-are-legal/2100-1027_3-998363.html/

[44] Walker, Leslie (2001-11-08). "Uncle Sam Wants Napster!". *The Washington Post*. Retrieved 2010-05-22.

[45] Krishnan, R., Smith, M. D., Tang, Z., & Telang, R. (2004, January). The impact of free-riding on peer-to-peer networks. In System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on (pp. 10-pp). IEEE.

[46] Feldman, M., Lai, K., Stoica, I., & Chuang, J. (2004, May). Robust incentive techniques for peer-to-peer networks. In Proceedings of the 5th ACM conference on Electronic commerce (pp. 102-111). ACM.

[47] Vu, Quang H.; et al. (2010). *Peer-to-Peer Computing: Principles and Applications*. Springer. p. 172. ISBN 978-3-642-03513-5.

[48] P. Antoniadis and B. Le Grand, "Incentives for resource sharing in self-organized communities: From economics to social psychology," Digital Information Management (ICDIM '07), 2007

[49] Vu, Quang H.; et al. (2010). *Peer-to-Peer Computing: Principles and Applications*. Springer. pp. 179–181. ISBN 978-3-642-03513-5.

[50] Majoras, D. B. (2005). Peer-to-peer file-sharing technology consumer protection and competition issues. Federal Trade Commission, Retrieved from http://www.ftc.gov/reports/p2p05/050623p2prpt.pdf

[51] The Government of the Hong Kong Special Administrative Region, (2008). Peer-to-peer network. Retrieved from website: http://www.infosec.gov.hk/english/technical/files/peer.pdf

[52] Janko Roettgers, 5 Ways to Test Whether your ISP throttles P2P, http://newteevee.com/2008/04/02/5-ways-to-test-if-your-isp-throttles-p2p/

[53] Hjelmvik, Erik; John, Wolfgang (2010-07-27). "Breaking and Improving Protocol Obfuscation" (PDF). ISSN 1652-926X.

[54] Basu, A., Fleming, S., Stanier, J., Naicken, S., Wakeman, I., & Gurbani, V. K. (2013). The state of peer-to-peer network simulators. ACM Computing Surveys (CSUR), 45(4), 46.

10.9 External links

- Ghosh Debjani, Rajan Payas, Pandey Mayank P2P-VoD Streaming: Design Issues & User Experience Challenges Springer Proceedings, June 2014
- Glossary of P2P terminology
- Foundation of Peer-to-Peer Computing, Special Issue, Elsevier Journal of Computer Communication, (Ed) Javed I. Khan and Adam Wierzbicki, Volume 31, Issue 2, February 2008
- Anderson, Ross J. "The eternity service". *Pragocrypt* **1996**.
- Marling Engle & J. I. Khan. Vulnerabilities of P2P systems and a critical look at their solutions, May 2006
- Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. ACM Computing Surveys, 36(4):335–371, December 2004.
- Biddle, Peter, Paul England, Marcus Peinado, and Bryan Willman, The Darknet and the Future of Content Distribution. In *2002 ACM Workshop on Digital Rights Management*, November 2002.
- John F. Buford, Heather Yu, Eng Keong Lua P2P Networking and Applications. ISBN 0123742145, Morgan Kaufmann, December 2008
- Djamal-Eddine Meddour, Mubashar Mushtaq, and Toufik Ahmed, "Open Issues in P2P Multimedia Streaming", in the proceedings of the 1st Multimedia Communications Workshop MULTICOMM 2006 held in conjunction with IEEE ICC 2006 pp 43–48, June 2006, Istanbul, Turkey.
- Detlef Schoder and Kai Fischbach, "Core Concepts in Peer-to-Peer (P2P) Networking". In: Subramanian, R.; Goodman, B. (eds.): *P2P Computing: The*

Evolution of a Disruptive Technology, Idea Group Inc, Hershey. 2005

- Ramesh Subramanian and Brian Goodman (eds), *Peer-to-Peer Computing: Evolution of a Disruptive Technology*, ISBN 1-59140-429-0, Idea Group Inc., Hershey, PA, USA, 2005.
- Shuman Ghosemajumder. Advanced Peer-Based Technology Business Models. *MIT Sloan School of Management, 2002.*
- Silverthorne, Sean. *Music Downloads: Pirates- or Customers?*. Harvard Business School Working Knowledge, 2004.
- Glasnost test P2P traffic shaping (Max Planck Institute for Software Systems)

Chapter 11

Mainframe computer

For other uses, see Mainframe (disambiguation).

Mainframe computers (colloquially referred to as "big iron"[1]) are computers used primarily by corporate and governmental organizations for critical applications, bulk data processing such as census, industry and consumer statistics, enterprise resource planning and transaction processing.

An IBM System z9 mainframe

The term originally referred to the large cabinets called "main frames" that housed the central processing unit and main memory of early computers.[2][3] Later, the term was used to distinguish high-end commercial machines from less powerful units.[4] Most large-scale computer system architectures were established in the 1960s, but continue to evolve.

11.1 Description

Modern mainframe design is generally less defined by single-task computational speed (typically defined as MIPS rate or FLOPS in the case of floating point calculations), and more by:

- Redundant internal engineering resulting in high reliability and security
- Extensive input-output facilities with the ability to offload to separate engines
- Strict backward compatibility with older software
- High hardware and computational utilization rates through virtualization to support massive throughput.

Their high stability and reliability enables these machines to run uninterrupted for decades.

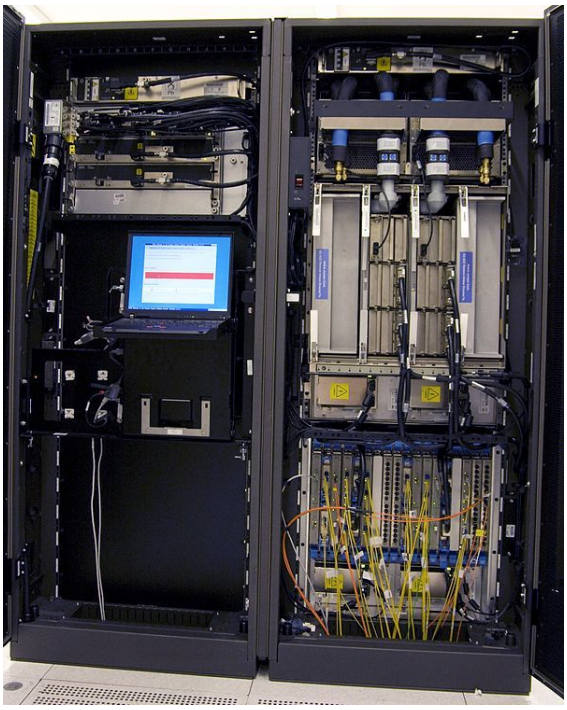
Software upgrades usually require setting up the operating system or portions thereof, and are non-disruptive only when using virtualizing facilities such as IBM's z/OS and Parallel Sysplex, or Unisys's XPCL, which support workload sharing so that one system can take over another's application while it is being refreshed. Mainframes are defined by high availability, one of the main reasons for their longevity, since they are typically used in applications where downtime would be costly or catastrophic. The term reliability, availability and serviceability (RAS) is a defining characteristic of mainframe computers. Proper planning and implementation is required to exploit these features, and if improperly implemented, may serve to inhibit the benefits provided. In addition, mainframes are more secure than other computer types: the NIST vulnerabilities database, US-CERT, rates traditional mainframes such as IBM zSeries, Unisys Dorado and Unisys Libra as among the most secure with vulnerabilities in the low single digits as compared with thousands for Windows, Unix, and Linux.[5]

In the late 1950s, most mainframes had no explicitly interactive interface. They accepted sets of punched cards, paper tape, or magnetic tape to transfer data and

programs. They operated in batch mode to support back office functions, such as customer billing, and supported interactive terminals almost exclusively for applications rather than program development. Typewriter and Teletype devices were also common control consoles for system operators through the 1970s, although ultimately supplanted by keyboard/display devices. By the early 1970s, many mainframes acquired interactive user interfaces[NB 1] and operated as timesharing computers, supporting hundreds of users simultaneously along with batch processing. Users gained access through specialized terminals or, later, from personal computers equipped with terminal emulation software. By the 1980s, many mainframes supported graphical terminals, and terminal emulation, but not graphical user interfaces. This format of end-user computing reached mainstream obsolescence in the 1990s due to the advent of personal computers provided with GUIs. After 2000, most modern mainframes have partially or entirely phased out classic "green screen" terminal access for end-users in favour of Web-style user interfaces.

The infrastructure requirements were drastically reduced during the mid-1990s when CMOS mainframe designs replaced the older bipolar technology. IBM claimed that its newer mainframes could reduce data center energy costs for power and cooling, and that they could reduce physical space requirements compared to server farms.[6]

11.2 Characteristics



Inside an IBM System z9 mainframe

Modern mainframes can run multiple different instances of operating systems at the same time. This technique of virtual machines allows applications to run as if they were on physically distinct computers. In this role, a single mainframe can replace higher-functioning hardware services available to conventional servers. While mainframes pioneered this capability, virtualization is now available on most families of computer systems, though not always to the same degree or level of sophistication.

Mainframes can add or hot swap system capacity without disrupting system function, with specificity and granularity to a level of sophistication not usually available with most server solutions. Modern mainframes, notably the IBM zSeries, System z9 and System z10 servers, offer two levels of virtualization: logical partitions (LPARs, via the PR/SM facility) and virtual machines (via the z/VM operating system). Many mainframe customers run two machines: one in their primary data center, and one in their backup data center—fully active, partially active, or on standby—in case there is a catastrophe affecting the first building. Test, development, training, and production workload for applications and databases can run on a single machine, except for extremely large demands where the capacity of one machine might be limiting. Such a two-mainframe installation can support continuous business service, avoiding both planned and unplanned outages. In practice many customers use multiple mainframes linked either by Parallel Sysplex and shared DASD (in IBM's case), or with shared, geographically dispersed storage provided by EMC or Hitachi.

Mainframes are designed to handle very high volume input and output (I/O) and emphasize throughput computing. Since the late-1950s,[NB 2] mainframe designs have included subsidiary hardware[NB 3] (called *channels* or *peripheral processors*) which manage the I/O devices, leaving the CPU free to deal only with high-speed memory. It is common in mainframe shops to deal with massive databases and files. Gigabyte to terabyte-size record files are not unusual.[7] Compared to a typical PC, mainframes commonly have hundreds to thousands of times as much data storage online, and can access it much faster. Other server families also offload I/O processing and emphasize throughput computing.

Mainframe return on investment (ROI), like any other computing platform, is dependent on its ability to scale, support mixed workloads, reduce labor costs, deliver uninterrupted service for critical business applications, and several other risk-adjusted cost factors.

Mainframes also have execution integrity characteristics for fault tolerant computing. For example, z900, z990, System z9, and System z10 servers effectively execute result-oriented instructions twice, compare results, arbitrate between any differences (through instruction retry and failure isolation), then shift workloads "in flight" to functioning processors, including spares, without any impact to operating systems, applications, or users. This hardware-level feature, also found in HP's NonStop sys-

tems, is known as lock-stepping, because both processors take their “steps” (*i.e.* instructions) together. Not all applications absolutely need the assured integrity that these systems provide, but many do, such as financial transaction processing.

11.3 Market

IBM mainframes dominate the mainframe market at well over 90% market share.[8] Unisys manufactures ClearPath Libra mainframes, based on earlier Burroughs products and ClearPath Dorado mainframes based on Sperry Univac OS 1100 product lines. In 2002, Hitachi co-developed the zSeries z800 with IBM to share expenses, but subsequently the two companies have not collaborated on new Hitachi models. Hewlett-Packard sells its unique NonStop systems, which it acquired with Tandem Computers and which some analysts classify as mainframes. Groupe Bull's DPS, Fujitsu (formerly Siemens) BS2000, and Fujitsu-ICL VME mainframes are still available in Europe. Fujitsu, Hitachi, and NEC (the “JCMs”) still maintain mainframe hardware businesses in the Japanese market.[9][10]

The amount of vendor investment in mainframe development varies with market share. Fujitsu and Hitachi both continue to use custom S/390-compatible processors, as well as other CPUs (including POWER and Xeon) for lower-end systems. Bull uses a mixture of Itanium and Xeon processors. NEC uses Xeon processors for its low-end ACOS-2 line, but develops the custom NOAH-6 processor for its high-end ACOS-4 series. IBM continues to pursue a different business strategy of mainframe investment and growth. IBM has its own large research and development organization designing new, homegrown CPUs, including mainframe processors such as 2012's 5.5 GHz six-core zEC12 mainframe microprocessor. Unisys produces code compatible mainframe systems that range from laptops to cabinet sized mainframes that utilize homegrown CPUs as well as Xeon processors. IBM is rapidly expanding its software business, including its mainframe software portfolio, to seek additional revenue and profits.[11]

Furthermore, there exists a market for software applications to manage the performance of mainframe implementations. In addition to IBM, significant players in this market include BMC,[12] Compuware,[13][14] and CA Technologies.[15]

11.4 History

Several manufacturers produced mainframe computers from the late 1950s through the 1970s. The group of manufacturers was first known as "IBM and the Seven Dwarfs":[16]:p.83 usually Burroughs, UNIVAC, NCR, Control Data, Honeywell, General Electric and RCA, although some lists varied. Later, with the departure of General Electric and RCA, it was referred to as IBM and the BUNCH. IBM's dominance grew out of their 700/7000 series and, later, the development of the 360 series mainframes. The latter architecture has continued to evolve into their current zSeries mainframes which, along with the then Burroughs and Sperry (now Unisys) MCP-based and OS1100 mainframes, are among the few mainframe architectures still extant that can trace their roots to this early period. While IBM's zSeries can still run 24-bit System/360 code, the 64-bit zSeries and System z9 CMOS servers have nothing physically in common with the older systems. Notable manufacturers outside the USA were Siemens and Telefunken in Germany, ICL in the United Kingdom, Olivetti in Italy, and Fujitsu, Hitachi, Oki, and NEC in Japan. The Soviet Union and Warsaw Pact countries manufactured close copies of IBM mainframes during the Cold War; the BESM series and Strela are examples of an independently designed Soviet computer.

An IBM 704 mainframe (1964)

Shrinking demand and tough competition started a shakeout in the market in the early 1970s — RCA sold out to UNIVAC and GE sold its business to Honeywell; in the 1980s Honeywell was bought out by Bull; UNIVAC became a division of Sperry, which later merged with Burroughs to form Unisys Corporation in 1986.

During the 1980s, minicomputer-based systems grew more sophisticated and were able to displace the lower-end of the mainframes. These computers, sometimes called *departmental computers* were typified by the DEC VAX.

In 1991, AT&T Corporation briefly owned NCR. During the same period, companies found that servers based on microcomputer designs could be deployed at a fraction of the acquisition price and offer local users much greater control over their own systems given the IT policies and practices at that time. Terminals used for interacting with mainframe systems were gradually replaced by personal computers. Consequently, demand plummeted and new mainframe installations were restricted mainly to financial services and government. In the early

1990s, there was a rough consensus among industry analysts that the mainframe was a dying market as mainframe platforms were increasingly replaced by personal computer networks. InfoWorld's Stewart Alsop famously predicted that the last mainframe would be unplugged in 1996; in 1993, he cited Cheryl Currid, a computer industry analyst as saying that the last mainframe "will stop working on December 31, 1999",[17] a reference to the anticipated Year 2000 problem (Y2K).

That trend started to turn around in the late 1990s as corporations found new uses for their existing mainframes and as the price of data networking collapsed in most parts of the world, encouraging trends toward more centralized computing. The growth of e-business also dramatically increased the number of back-end transactions processed by mainframe software as well as the size and throughput of databases. Batch processing, such as billing, became even more important (and larger) with the growth of e-business, and mainframes are particularly adept at large scale batch computing. Another factor currently increasing mainframe use is the development of the Linux operating system, which arrived on IBM mainframe systems in 1999 and is typically run in scores or hundreds of virtual machines on a single mainframe. Linux allows users to take advantage of open source software combined with mainframe hardware RAS. Rapid expansion and development in emerging markets, particularly People's Republic of China, is also spurring major mainframe investments to solve exceptionally difficult computing problems, e.g. providing unified, extremely high volume online transaction processing databases for 1 billion consumers across multiple industries (banking, insurance, credit reporting, government services, etc.) In late 2000 IBM introduced 64-bit z/Architecture, acquired numerous software companies such as Cognos and introduced those software products to the mainframe. IBM's quarterly and annual reports in the 2000s usually reported increasing mainframe revenues and capacity shipments. However, IBM's mainframe hardware business has not been immune to the recent overall downturn in the server hardware market or to model cycle effects. For example, in the 4th quarter of 2009, IBM's System z hardware revenues decreased by 27% year over year. But MIPS shipments (a measure of mainframe capacity) increased 4% per year over the past two years.[18] Alsop had himself photographed in 2000, symbolically eating his own words ("death of the mainframe").[19]

In 2012, NASA powered down its last mainframe, an IBM System z9.[20] However, IBM's successor to the z9, the z10, led a New York Times reporter to state four years earlier that "mainframe technology — hardware, software and services — remains a large and lucrative business for I.B.M., and mainframes are still the back-office engines behind the world's financial markets and much of global commerce".[21] As of 2010, while mainframe technology represented less than 3% of IBM's revenues, it "continue[d] to play an outsized role in Big Blue's results".[22]

11.5 Differences from supercomputers

A supercomputer is a computer that is at the frontline of current processing capacity, particularly speed of calculation. Supercomputers are used for scientific and engineering problems (high-performance computing) which are data crunching and number crunching,[23] while mainframes are used for transaction processing. The differences are as follows:

- Mainframes are often approximately measured in millions of instructions per second (MIPS),[24] but supercomputers are measured in floating point operations per second (FLOPS) and more recently by traversed edges per second or TEPS.[25] Examples of integer operations include moving data around in memory or checking values. Floating point operations are mostly addition, subtraction, and multiplication with enough digits of precision to model continuous phenomena such as weather prediction and nuclear simulations. In terms of computational ability, supercomputers are more powerful.[26]
- Mainframes are built to be reliable for transaction processing as it is commonly understood in the business world: a commercial exchange of goods, services, or money. A typical transaction, as defined by the Transaction Processing Performance Council,[27] would include the updating to a database system for such things as inventory control (goods), airline reservations (services), or banking (money). A transaction could refer to a set of operations including disk read/writes, operating system calls, or some form of data transfer from one subsystem to another. This operation doesn't count toward the processing power of a computer. Transaction processing is not exclusive to mainframes but also used in the performance of microprocessor-based servers and online networks.

In 2007,[28] an amalgamation of the different technologies and architectures for supercomputers and mainframes has led to the so-called gameframe.

11.6 See also

- Computer types
- Failover
- Gameframe
- Channel I/O
- Cloud computing

11.7 Notes

[1] In some cases the interfaces were introduced in the 1960s but their deployment became more common in the 1970s

[2] E.g., the IBM 709 had channels in 1958

[3] sometimes computers, sometimes more limited

11.8 References

[1] "IBM preps big iron fiesta". The Register. July 20, 2005.

[2] "mainframe, n". *Oxford English Dictionary* (on-line ed.).

[3] Ebbers, Mike; O'Brien, W.; Ogden, B. (2006). "Introduction to the New Mainframe: z/OS Basics" (PDF). IBM International Technical Support Organization. Retrieved 2007-06-01.

[4] Beach, Thomas E. "Computer Concepts and Terminology: Types of Computers". Retrieved November 17, 2012.

[5] "National Vulnerability Database". Retrieved September 20, 2011.

[6] "Get the facts on IBM vs the Competition- The facts about IBM System z "mainframe"". IBM. Retrieved December 28, 2009.

[7] "Largest Commercial Database in Winter Corp. TopTen Survey Tops One Hundred Terabytes". *Press release*. Retrieved 2008-05-16.

[8] "IBM Tightens Stranglehold Over Mainframe Market; Gets Hit with Antitrust Complaint in Europe". CCIA. 2008-07-02. Retrieved 2008-07-09.

[9] GlobalServer : Fujitsu Global. Fujitsu.com. Retrieved on 2013-07-17.

[10] □□□□□□□□□□□AP8800E:□□. Hitachi.co.jp. Retrieved on 2013-07-17.

[11] "IBM Opens Latin America's First Mainframe Software Center". Enterprise Networks and Servers. August 2007.

[12] "Mainframe Automation Management". Retrieved 26 October 2012.

[13] "Mainframe Modernization". Retrieved 26 October 2012.

[14] "Automated Mainframe Testing & Auditing". Retrieved 26 October 2012.

[15] "CA Technologies".

[16] Bergin, Thomas J (ed.) (2000). *50 Years of Army Computing: From ENIAC to MSRC*. DIANE Publishing. ISBN 0-9702316-1-X.

[17] Alsop, Stewart (Mar 8, 1993). "IBM still has brains to be player in client/server platforms". *InfoWorld*. Retrieved Dec 26, 2013.

[18] "IBM 4Q2009 Financial Report: CFO's Prepared Remarks". IBM. January 19, 2010.

[19] "Stewart Alsop eating his words". *Computer History Museum*. Retrieved Dec 26, 2013.

[20] Cureton, Linda (11 February 2012). *The End of the Mainframe Era at NASA*. NASA. Retrieved 31 January 2014.

[21] Lohr, Steve (March 23, 2008). "Why Old Technologies Are Still Kicking". *The New York Times*. Retrieved Dec 25, 2013.

[22] Ante, Spencer E. (July 22, 2010). "IBM Calculates New Mainframes Into Its Future Sales Growth". *The Wall Street Journal*. Retrieved Dec 25, 2013.

[23] High-Performance Graph Analysis Retrieved on February 15, 2012

[24] Resource consumption for billing and performance purposes is measured in units of a Million service units (MSU), but the definition of MSU varies from processor to processor in such a fashion as to make MSU's/s useless for comparing processor performance.

[25] The Graph 500 Retrieved on February 19, 2012

[26] World's Top Supercomputer Retrieved on December 25, 2009

[27] Transaction Processing Performance Council Retrieved on December 25, 2009.

[28] Cell Broadband Engine Project Aims to Supercharge IBM Mainframe for Virtual Worlds

11.9 External links

- IBM Systems Mainframe Magazine
- IBM eServer zSeries mainframe servers
- Univac 9400, a mainframe from the 1960s, still in use in a German computer museum
- Lectures in the History of Computing: Mainframes (archived copy from the Internet Archive)
- Articles and Tutorials at Mainframes360.com: Mainframes
- Mainframe Tutorials and Forum at mainframewizard.com: Mainframes

Chapter 12

Utility computing

Utility computing is a service provisioning model in which a service provider makes computing resources and infrastructure management available to the customer as needed, and charges them for specific usage rather than a flat rate. Like other types of on-demand computing (such as grid computing), the utility model seeks to maximize the efficient use of resources and/or minimize associated costs. Utility is the packaging of computing resources, such as computation, storage and services, as a metered service. This model has the advantage of a low or no initial cost to acquire computer resources; instead, computational resources are essentially rented.

This repackaging of computing services became the foundation of the shift to "on demand" computing, software as a service and cloud computing models that further propagated the idea of computing, application and network as a service.

There was some initial skepticism about such a significant shift.[1] However, the new model of computing caught on and eventually became mainstream.

IBM, HP and Microsoft were early leaders in the new field of utility computing, with their business units and researchers working on the architecture, payment and development challenges of the new computing model. Google, Amazon and others started to take the lead in 2008, as they established their own utility services for computing, storage and applications.

Utility computing can support grid computing which has the characteristic of very large computations or a sudden peaks in demand which are supported via a large number of computers.

“Utility computing” has usually envisioned some form of virtualization so that the amount of storage or computing power available is considerably larger than that of a single time-sharing computer. Multiple servers are used on the “back end” to make this possible. These might be a dedicated computer cluster specifically built for the purpose of being rented out, or even an under-utilized supercomputer. The technique of running a single calculation on multiple computers is known as distributed computing.

The term "grid computing" is often used to describe a particular form of distributed computing, where the supporting nodes are geographically distributed or cross administrative domains. To provide utility computing services, a company can “bundle” the resources of members of the public for sale, who might be paid with a portion of the revenue from clients.

One model, common among volunteer computing applications, is for a central server to dispense tasks to participating nodes, on the behest of approved end-users (in the commercial case, the paying customers). Another model, sometimes called the Virtual Organization (VO), is more decentralized, with organizations buying and selling computing resources as needed or as they go idle.

The definition of “utility computing” is sometimes extended to specialized tasks, such as web services.

12.1 History

Utility computing merely means “Pay and Use”, with regards to computing power. Utility computing is not a new concept, but rather has quite a long history. Among the earliest references is:

IBM and other mainframe providers conducted this kind of business in the following two decades, often referred to as time-sharing, offering computing power and database storage to banks and other large organizations from their world wide data centers. To facilitate this business model, mainframe operating systems evolved to include process control facilities, security, and user metering. The advent of mini computers changed this business model, by making computers affordable to almost all companies. As Intel and AMD increased the power of PC architecture servers with each new generation of processor, data centers became filled with thousands of servers.

In the late 90’s utility computing re-surfaced. InsynQ (), Inc. launched [on-demand] applications and desktop hosting services in 1997 using HP equipment. In 1998, HP set up the Utility Computing Division in Mountain View, CA, assigning former Bell Labs computer scientists to begin work on a computing power plant, incorporating multiple utilities to form a software stack. Ser-

vices such as “IP billing-on-tap” were marketed. HP introduced the Utility Data Center in 2001. Sun announced the Sun Cloud service to consumers in 2000. In December 2005, Alexa launched Alexa Web Search Platform, a Web search building tool for which the underlying power is utility computing. Alexa charges users for storage, utilization, etc. There is space in the market for specific industries and applications as well as other niche applications powered by utility computing. For example, PolyServe Inc. offers a clustered file system based on commodity server and storage hardware that creates highly available utility computing environments for mission-critical applications including Oracle and Microsoft SQL Server databases, as well as workload optimized solutions specifically tuned for bulk storage, high-performance computing, vertical industries such as financial services, seismic processing, and content serving. The Database Utility and File Serving Utility enable IT organizations to independently add servers or storage as needed, retask workloads to different hardware, and maintain the environment without disruption.

In spring 2006 3tera announced its AppLogic service and later that summer Amazon launched Amazon EC2 (Elastic Compute Cloud). These services allow the operation of general purpose computing applications. Both are based on Xen virtualization software and the most commonly used operating system on the virtual computers is Linux, though Windows and Solaris are supported. Common uses include web application, SaaS, image rendering and processing but also general-purpose business applications.

12.2 See also

- Edge computing

12.3 References

[1] *On-demand computing: What are the odds?*, ZD Net, Nov 2002, retrieved Oct 2010

[2] Garfinkel, Simson (1999). Abelson, Hal, ed. *Architects of the Information Society, Thirty-Five Years of the Laboratory for Computer Science at MIT*. MIT Press. ISBN 978-0-262-07196-3.

Decision support and business intelligence 8th edition page 680 ISBN 0-13-198660-0

12.4 External links

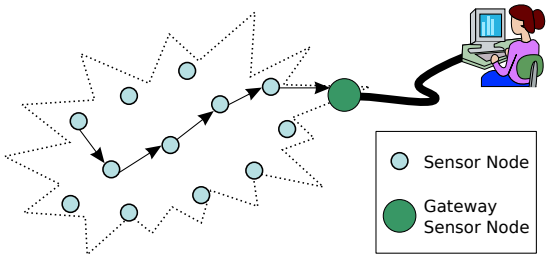
- How Utility Computing Works
- Utility computing definition

Chapter 13

Wireless sensor network

"WSN" redirects here. For other uses, see WSN (disambiguation).

A **wireless sensor network (WSN)** (sometimes called a **wireless sensor and actor network (WSAN)**[1]) of spatially distributed autonomous sensors to *monitor* physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling *control* of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.



Typical multi-hop wireless sensor network architecture

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.[2][3]

In computer science and telecommunications, wireless sensor networks are an active research area with numerous workshops and conferences arranged each year, for example IPSN, SenSys, and EWSN.

13.1 Applications

13.1.1 Process Management

13.1.2 Area monitoring

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors detect enemy intrusion; a civilian example is the geo-fencing of gas or oil pipelines.

13.1.3 Health care monitoring

The medical applications can be of two types: wearable and implanted. Wearable devices are used on the body surface of a human or just at close proximity of the user. The implantable medical devices are those that are inserted inside human body. There are many other applications too e.g. body position measurement and location of the person, overall monitoring of ill patients in hospitals and at homes. Body-area networks can collect information about an individual's health, fitness, and energy expenditure.[4]

13.1.4 Environmental/Earth sensing

There are many applications in monitoring environmental parameters,[5] examples of which are given below. They share the extra challenges of harsh environments and reduced power supply.

Air pollution monitoring

Wireless sensor networks have been deployed in several cities (Stockholm[citation needed], London[citation needed] and Brisbane[citation needed]) to monitor the concentration of dangerous gases for citizens. These can take advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas.

Forest fire detection

A network of Sensor Nodes can be installed in a forest to detect when a fire has started. The nodes can be equipped with sensors to measure temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the firefighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

Landslide detection

A landslide detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. Through the data gathered it may be possible to know the occurrence of landslides long before it actually happens.

Water quality monitoring

Water quality monitoring involves analyzing water properties in dams, rivers, lakes & oceans, as well as underground water reserves. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the permanent deployment of monitoring stations in locations of difficult access, without the need of manual data retrieval.[6]

Natural disaster prevention

Wireless sensor networks can effectively act to prevent the consequences of natural disasters, like floods. Wireless nodes have successfully been deployed in rivers where changes of the water levels have to be monitored in real time.

Chemical agent detection

The U.S. Department of Homeland Security has sponsored the integration of chemical agent sensor systems into city infrastructures as part of its counterterrorism efforts. In addition, DHS is supporting the development of crowdsourced sensing systems that will draw upon chemical agent detectors embedded in mobile phones.[7]

13.1.5 Industrial monitoring

Machine health monitoring

Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionality.[8]

Wireless sensors can be placed in locations difficult or impossible to reach with a wired system, such as rotating machinery and untethered vehicles.

Data logging

Main article: Data logging

Wireless sensor networks are also used for the collection of data for monitoring of environmental information, this can be as simple as the monitoring of the temperature in a fridge to the level of water in overflow tanks in nuclear power plants. The statistical information can then be used to show how systems have been working. The advantage of WSNs over conventional loggers is the “live” data feed that is possible.

Water/Waste water monitoring

Monitoring the quality and level of water includes many activities such as checking the quality of underground or surface water and ensuring a country’s water infrastructure for the benefit of both human and animal.It may be used to protect the wastage of water.

Structural Health Monitoring

Main article: Structural health monitoring

Wireless sensor networks can be used to monitor the condition of civil infrastructure and related geo-physical processes close to real time, and over long periods through data logging, using appropriately interfaced sensors.

13.2 Characteristics

The main characteristics of a WSN include:

- Power consumption constraints for nodes using batteries or energy harvesting
- Ability to cope with node failures (resilience)

- Mobility of nodes
- Heterogeneity of nodes
- Scalability to large scale of deployment
- Ability to withstand harsh environmental conditions
- Ease of use
- Cross-layer design

Cross-layer is becoming an important studying area for wireless communications. In addition, the traditional layered approach presents three main problems:

1. Traditional layered approach cannot share different information among different layers , which leads to each layer not having complete information. The traditional layered approach cannot guarantee the optimization of the entire network.
2. The traditional layered approach does not have the ability to adapt to the environmental change.
3. Because of the interference between the different users, access confliction, fading, and the change of environment in the wireless sensor networks, traditional layered approach for wired networks is not applicable to wireless networks.

So the cross-layer can be used to make the optimal modulation to improve the transmission performance, such as data rate, energy efficiency, QoS (Quality of Service), etc.. Sensor nodes can be imagined as small computers which are extremely basic in terms of their interfaces and their components. They usually consist of a *processing unit* with limited computational power and limited memory, *sensors* or MEMS (including specific conditioning circuitry), a *communication device* (usually radio transceivers or alternatively optical), and a power source usually in the form of a battery. Other possible inclusions are energy harvesting modules,[9] secondary ASICs, and possibly secondary communication interface (e.g. RS-232 or USB).

The base stations are one or more components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user as they typically forward data from the WSN on to a server. Other special components in routing based networks are routers, designed to compute, calculate and distribute the routing tables.

13.3 Platforms

13.3.1 Hardware

Main article: sensor node

One major challenge in a WSN is to produce *low cost* and *tiny* sensor nodes. There are an increasing number of small companies producing WSN hardware and the commercial situation can be compared to home computing in the 1970s. Many of the nodes are still in the research and development stage, particularly their software. Also inherent to sensor network adoption is the use of very low power methods for radio communication and data acquisition.

In many applications, a WSN communicates with a Local Area Network or Wide Area Network through a gateway. The Gateway acts as a bridge between the WSN and the other network. This enables data to be stored and processed by devices with more resources, for example, in a remotely located server.

13.3.2 Software

Energy is the scarcest resource of WSN nodes, and it determines the lifetime of WSNs. WSNs may be deployed in large numbers in various environments, including remote and hostile regions, where ad hoc communications are a key component. For this reason, algorithms and protocols need to address the following issues:

- Lifespan is increased
- Robustness and fault tolerance
- Self-configuration

Lifetime maximization: Energy/Power Consumption of the sensing device should be minimized and sensor nodes should be energy efficient since their limited energy resource determines their lifetime. To conserve power, wireless sensor nodes normally power off both the radio transmitter and the radio receiver when not in use.

Operating systems

Operating systems for wireless sensor network nodes are typically less complex than general-purpose operating systems. They more strongly resemble embedded systems, for two reasons. First, wireless sensor networks are typically deployed with a particular application in mind, rather than as a general platform. Second, a need for low costs and low power leads most wireless sensor nodes to have low-power microcontrollers ensuring that mechanisms such as virtual memory are either unnecessary or too expensive to implement.

It is therefore possible to use embedded operating systems such as eCos or uC/OS for sensor networks. However, such operating systems are often designed with real-time properties.

TinyOS is perhaps the first[10] operating system specifically designed for wireless sensor networks. TinyOS

is based on an event-driven programming model instead of multithreading. TinyOS programs are composed of *event handlers* and *tasks* with run-to-completion semantics. When an external event occurs, such as an incoming data packet or a sensor reading, TinyOS signals the appropriate event handler to handle the event. Event handlers can post tasks that are scheduled by the TinyOS kernel some time later.

LiteOS is a newly developed OS for wireless sensor networks, which provides UNIX-like abstraction and support for the C programming language.

Contiki is an OS which uses a simpler programming style in C while providing advances such as 6LoWPAN and Protothreads.

RIOT implements a microkernel architecture. It provides multithreading with standard API and allows for development in C/C++. RIOT supports common IoT protocols such as 6LoWPAN, IPv6, RPL, TCP, and UDP.[11]

ERIKA Enterprise is an open-source and royalty-free OSEK/VDX Kernel offering BCC1, BCC2, ECC1, ECC2, multicore, memory protection and kernel fixed priority adopting C programming language.

13.3.3 Online collaborative sensor data management platforms

Online collaborative sensor data management platforms are on-line database services that allow sensor owners to register and connect their devices to feed data into an online database for storage and also allow developers to connect to the database and build their own applications based on that data. Examples include Xively and the Wikisensing platform. Such platforms simplify online collaboration between users over diverse data sets ranging from energy and environment data to that collected from transport services. Other services include allowing developers to embed real-time graphs & widgets in websites; analyse and process historical data pulled from the data feeds; send real-time alerts from any datastream to control scripts, devices and environments.

The architecture of the Wikisensing system is described in [12] describes the key components of such systems to include APIs and interfaces for online collaborators, a middleware containing the business logic needed for the sensor data management and processing and a storage model suitable for the efficient storage and retrieval of large volumes of data.

13.4 Simulation of WSNs

At present, agent-based modeling and simulation is the only paradigm which allows the simulation of complex behavior in the environments of wireless sensors (such as flocking).[13] Agent-based simulation of wireless sensor and ad hoc networks is a relatively new paradigm. Agent-based modelling was originally based on social simulation.

Network simulators like OPNET, OMNeT++, NetSim, Worldsens (WSNet and WSIM),[14][15] and NS2 can be used to simulate a wireless sensor network.

13.5 Other concepts

13.5.1 Distributed sensor network

If a centralised architecture is used in a sensor network and the central node fails, then the entire network will collapse, however the reliability of the sensor network can be increased by using a distributed control architecture. Distributed control is used in WSNs for the following reasons:

1. Sensor nodes are prone to failure,
2. For better collection of data
3. To provide nodes with backup in case of failure of the central node

There is also no centralised body to allocate the resources and they have to be self organised.

13.5.2 Data integration and Sensor Web

The data gathered from wireless sensor networks is usually saved in the form of numerical data in a central base station. Additionally, the Open Geospatial Consortium (OGC) is specifying standards for interoperability interfaces and metadata encodings that enable real time integration of heterogeneous sensor webs into the Internet, allowing any individual to monitor or control Wireless Sensor Networks through a Web Browser.

13.5.3 In-network processing

To reduce communication costs some algorithms remove or reduce nodes' redundant sensor information and avoid forwarding data that is of no use. As nodes can inspect the data they forward, they can measure averages or directionality for example of readings from other nodes. For example, in sensing and monitoring applications, it is generally the case that neighboring sensor nodes monitoring an environmental feature typically register similar values. This kind of data redundancy due to the spatial correlation between sensor observations inspires techniques for in-network data aggregation and mining

13.6 See also

- Ad hoc On-Demand Distance Vector Routing
- Ambient intelligence
- Backpressure routing
- Barrier resilience
- body area network
- List of ad hoc routing protocols
- MQ Telemetry Transport
- MyriaNed
- optical wireless communications
- Sensor grid
- Smart, Connected Products

13.7 References

[1] .F. Akyildiz and I.H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges,"; Ad Hoc Networks, vol. 2, no. 4, pp. 351-367, Oct. 2004.

[2] Dargie, W. and Poellabauer, C., "Fundamentals of wireless sensor networks: theory and practice", John Wiley and Sons, 2010 ISBN 978-0-470-99765-9, pp. 168–183, 191–192

[3] Sohraby, K., Minoli, D., Znati, T. "Wireless sensor networks: technology, protocols, and applications", John Wiley and Sons, 2007 ISBN 978-0-471-74300-2, pp. 203–209

[4] Peiris, V. (2013). "Highly integrated wireless sensing for body area network applications". *SPIE Newsroom*. doi:10.1117/2.1201312.005120.

[5] J.K.Hart and K.Martinez, "Environmental Sensor Networks: A revolution in the earth system science?", Earth Science Reviews, 2006

[6] Spie (2013). "Vassili Karanassios: Energy scavenging to power remote sensors". *SPIE Newsroom*. doi:10.1117/2.3201305.05.

[7] Monahan, Torin, Mokos, Jennifer T. (2013). "Crowdsourcing Urban Surveillance: The Development of Homeland Security Markets for Environmental Sensor Networks" (pdf). *Geoforum* **49**: 279–288. doi:10.1016/j.geoforum.2013.02.001.

[8] Tiwari, Ankit et al., Energy-efficient wireless sensor network design and implementation for condition-based maintenance, ACM Transactions on Sensor Networks (TOSN), http://portal.acm.org/citation.cfm?id=1210670

[9] Magno, M.; Boyle, D.; Brunelli, D.; O'Flynn, B.; Popovici, E.; Benini, L. (2014). "Extended Wireless Monitoring Through Intelligent Hybrid Energy Supply". *IEEE Transactions on Industrial Electronics* **61** (4): 1871. doi:10.1109/TIE.2013.2267694.

[10] TinyOS Programming, Philip Levis, Cambridge University Press, 2009

[11] Oliver Hahm, Emmanuel Baccelli, Mesut Günes, Matthias Wählisch, Thomas C. Schmidt, *RIOT OS: Towards an OS for the Internet of Things*, In: Proc. of the 32nd IEEE INFOCOM. Poster Session, Piscataway, NJ, USA:IEEE Press, 2013.

[12] Silva, D.; Ghanem, M.; Guo, Y. (2012). "WikiSensing: An Online Collaborative Approach for Sensor Data Management". *Sensors* **12** (12): 13295. doi:10.3390/s121013295.

[13] Muaz Niazi, Amir Hussain (2011). A Novel Agent-Based Simulation Framework for Sensing in Complex Adaptive Environments. IEEE Sensors Journal, Vol.11 No. 2, 404–412. Paper

[14] Shafiullah Khan, Al-Sakib Khan Pathan, Nabil Ali Alrajeh. "Wireless Sensor Networks: Current Status and Future Trends". 2012. p. 236.

[15] Ruiz-Martinez, Antonio. "Architectures and Protocols for Secure Information Technology Infrastructures". 2013. p. 117.

13.8 External links

- IEEE 802.15.4 Standardization Committee

13.9 Further reading

- Mark Fell. "Roadmap for the Emerging Internet of Things - Its Impact, Architecture and Future Governance". Carré & Strauss, United Kingdom, 2014.

Chapter 14

Internet of Things

The **Internet of Things** (**IoT**) is the network of physical objects or "things" embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices. Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing Internet infrastructure.

Typically, IoT is expected to offer advanced connectivity of devices, systems, and services that goes beyond machine-to-machine communications (M2M) and covers a variety of protocols, domains, and applications.[1] The interconnection of these embedded devices (including smart objects), is expected to usher in automation in nearly all fields, while also enabling advanced applications like a Smart Grid.[2]

Things, in the IoT, can refer to a wide variety of devices such as heart monitoring implants, biochip transponders on farm animals, electric clams in coastal waters,[3] automobiles with built-in sensors, or field operation devices that assist fire-fighters in search and rescue.[4] These devices collect useful data with the help of various existing technologies and then autonomously flow the data between other devices.[5] Current market examples include smart thermostat systems and washer/dryers that utilize wifi for remote monitoring.

Besides the plethora of new application areas for Internet connected automation to expand into, IoT is also expected to generate large amounts of data from diverse locations that is aggregated at a very high velocity, thereby increasing the need to better index, store and process such data.[6][7]

14.1 Early history

As of 2014, the vision of the Internet of Things has evolved due to a convergence of multiple technologies, ranging from wireless communication to the Internet and from embedded systems to micro-electromechanical systems (MEMS).[4] This means that the traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the Internet of Things (IoT).

The concept of a network of smart devices was discussed as early as 1982, with a modified Coke machine at Carnegie Mellon University becoming the first internet-connected appliance,[8] able to report its inventory and whether newly loaded drinks were cold.[9] Mark Weiser's seminal 1991 paper on ubiquitous computing, "The Computer of the 21st Century", as well as academic venues such as UbiComp and PerCom produced the contemporary vision of IoT.[5][10] In 1994 Reza Raji described the concept in *IEEE Spectrum* as "[moving] small packets of data to a large set of nodes, so as to integrate and automate everything from home appliances to entire factories".[11] However, only in 1999 did the field start gathering momentum. Bill Joy envisioned Device to Device (D2D) communication as part of his "Six Webs" framework, presented at the World Economic Forum at Davos in 1999.[12]

The concept of the Internet of Things first became popular in 1999, through the Auto-ID Center at MIT and related market-analysis publications.[13] Radio-frequency identification (RFID) was seen as a prerequisite for the Internet of Things in the early days. If all objects and people in daily life were equipped with identifiers, computers could manage and inventory them.[14][15] Besides using RFID, the *tagging* of things may be achieved through such technologies as near field communication, barcodes, QR codes and digital watermarking.[16][17]

In its original interpretation, one of the first consequences of implementing the Internet of Things by equipping all objects in the world with minuscule identifying devices or machine-readable identifiers would be to transform daily life in several positive ways.[18][19] For instance, instant and ceaseless inventory control would become ubiquitous.[19] A person's ability to interact with objects could be altered remotely based on immediate or present needs, in accordance with existing end-user agreements.[14] For example, such technology could grant motion-picture publishers much more control over end-user private devices by enforcing remotely copyright restrictions and digital restrictions management, so the ability of a customer who bought a Blu-ray disc to watch the movie becomes dependent on so-called "copyright

holder's" decision, similar to Circuit City's failed DIVX.

14.2 Applications

According to Gartner, Inc. (a technology research and advisory corporation), there will be nearly 26 billion devices on the Internet of Things by 2020.[20] ABI Research estimates that more than 30 billion devices will be wirelessly connected to the Internet of Things (Internet of Everything) by 2020.[21] As per a recent survey and study done by Pew Research Internet Project, a large majority of the technology experts and engaged Internet users who responded—83 percent—agreed with the notion that the Internet/Cloud of Things, embedded and wearable computing (and the corresponding dynamic systems [22]) will have widespread and beneficial effects by 2025.[23] It is, as such, clear that the IoT will consist of a very large number of devices being connected to the Internet.[24]

Integration with the Internet implies that devices will utilize an IP address as a unique identifier. However, due to the limited address space of IPv4 (which allows for 4.3 billion unique addresses), objects in the IoT will have to use IPv6 to accommodate the extremely large address space required. [25] [26] [27] [28] [29] Objects in the IoT will not only be devices with sensory capabilities, but also provide actuation capabilities (e.g., bulbs or locks controlled over the Internet).[30] To a large extent, the future of the Internet of Things will not be possible without the support of IPv6; and consequently the global adoption of IPv6 in the coming years will be critical for the successful development of the IoT in the future. [26] [27] [28] [29]

The ability to network embedded devices with limited CPU, memory and power resources means that IoT finds applications in nearly every field.[31] Such systems could be in charge of collecting information in settings ranging from natural ecosystems to buildings and factories,[30] thereby finding applications in fields of environmental sensing and urban planning.[32]
On the other hand, IoT systems could also be responsible for performing actions, not just sensing things. Intelligent shopping systems, for example, could monitor specific users' purchasing habits in a store by tracking their specific mobile phones. These users could then be provided with special offers on their favorite products, or even location of items that they need, which their fridge has automatically conveyed to the phone.[33][34] Additional examples of sensing and actuating are reflected in applications that deal with heat, electricity and energy management, as well as cruise-assisting transportation systems.[35]

However, the application of the IoT is not only restricted to these areas. Other specialized use cases of the IoT may also exist. An overview of some of the most prominent application areas is provided here. Based on the application domain, IoT products can be classified broadly into five different categories: smart wearable, smart home, smart city, smart environment, and smart enterprise. The IoT products and solutions in each of these markets have different characteristics.[36]

14.2.1 Media

In order to home into the manner in which the Internet of Things (IoT), the Media and Big Data are interconnected, it is first necessary to provide some context into the mechanism used for media process. It has been suggested by Nick Couldry and Joseph Turow that Practitioners in Advertising and Media approach Big Data as many actionable points of information about millions of individuals. The industry appears to be moving away from the traditional approach of using specific media environments such as newspapers, magazines, or television shows and instead tap into consumers with technologies that reach targeted people at optimal times in optimal locations. The ultimate aim is of course to serve, or convey, a message or content that is (statistically speaking) in line with the consumer's mindset. For example, publishing environments are increasingly tailoring messages (advertisements) and content (articles) to appeal to consumers that have been exclusively gleaned through various data-mining activities.[37]

The media industries process Big Data in a dual, interconnected manner:

- Targeting of consumers (for advertising by marketers)
- Data-capture

According to Danny Meadows-Klue, the combination of analytics for conversion tracking, with behavioural targeting and programmatic marketing has unlocked a new level of precision that enables display advertising to be focussed on the devices of people with relevant interests.[38] Big Data and the IoT work in conjunction. From a media perspective, Data is the key derivative of device inter connectivity, whilst being pivotal in allowing clearer accuracy in targeting. The Internet of Things therefore transforms the media industry, companies and even governments, opening up a new era of economic growth and competitiveness. The wealth of data generated by this industry (i.e. Big Data) will allow Practitioners in Advertising and Media to gain an elaborate layer on the present targeting mechanisms utilised by the industry.

14.2.2 Environmental monitoring

Environmental monitoring applications of the IoT typically utilize sensors to assist in environmental protection by monitoring air or water quality,[3] atmospheric or soil conditions,[39] and can even include areas like monitoring the movements of wildlife and their habitats.[40] Development of resource[41] constrained devices connected

to the Internet also means that other applications like earthquake or tsunami early-warning systems can also be used by emergency services to provide more effective aid. IoT devices in this application typically span a large geographic area and can also be mobile.[30]

14.2.3 Infrastructure management

Monitoring and controlling operations of urban and rural infrastructures like bridges, railway tracks, on- and offshore- wind-farms is a key application of the IoT.[42] The IoT infrastructure can be used for monitoring any events or changes in structural conditions that can compromise safety and increase risk. It can also be utilized for scheduling repair and maintenance activities in an efficient manner, by coordinating tasks between different service providers and users of these facilities.[30] IoT devices can also be used to control critical infrastructure like bridges to provide access to ships. Usage of IoT devices for monitoring and operating infrastructure is likely to improve incident management and emergency response coordination, and quality of service, up-times and reduce costs of operation in all infrastructure related areas.[43] Even areas such as waste management stand to benefit from automation and optimization that could be brought in by the IoT.[44]

14.2.4 Manufacturing

Network control and management of manufacturing equipment, asset and situation management, or manufacturing process control bring the IoT within the realm on industrial applications and smart manufacturing as well.[45] The IoT intelligent systems enable rapid manufacturing of new products, dynamic response to product demands, and real-time optimization of manufacturing production and supply chain networks, by networking machinery, sensors and control systems together.[30]

Digital control systems to automate process controls, operator tools and service information systems to optimize plant safety and security are within the purview of the IoT.[42] But it also extends itself to asset management via predictive maintenance, statistical evaluation, and measurements to maximize reliability.[46] Smart industrial management systems can also be integrated with the Smart Grid, thereby enabling real-time energy optimization. Measurements, automated controls, plant optimization, health and safety management, and other functions are provided by a large number of networked sensors.[30]

14.2.5 Energy management

Integration of sensing and actuation systems, connected to the Internet, is likely to optimize energy consumption as a whole.[30] It is expected that IoT devices will be integrated into all forms of energy consuming devices (switches, power outlets, bulbs, televisions, etc.) and be able to communicate with the utility supply company in order to effectively balance power generation and energy usage.[47] Such devices would also offer the opportunity for users to remotely control their devices, or centrally manage them via a cloud based interface, and enable advanced functions like scheduling (e.g., remotely powering on or off heating systems, controlling ovens, changing lighting conditions etc.).[30] In fact, a few systems that allow remote control of electric outlets are already available in the market, e.g., Belkin's WeMo,[48] Ambery Remote Power Switch,[49] Budderfly [50] etc.

Besides home based energy management, the IoT is especially relevant to the Smart Grid since it provides systems to gather and act on energy and power-related information in an automated fashion with the goal to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity.[47] Using Advanced Metering Infrastructure (AMI) devices connected to the Internet backbone, electric utilities can not only collect data from end-user connections, but also manage other distribution automation devices like transformers and reclosers.[30]

14.2.6 Medical and healthcare systems

IoT devices can be used to enable remote health monitoring and emergency notification systems. These health monitoring devices can range from blood pressure and heart rate monitors to advanced devices capable of monitoring specialized implants, such as pacemakers or advanced hearing aids.[30] Specialized sensors can also be equipped within living spaces to monitor the health and general well-being of senior citizens, while also ensuring that proper treatment is being administered and assisting people regain lost mobility via therapy as well.[51] Other consumer devices to encourage healthy living, such as, connected scales or wearable heart monitors, are also a possibility with the IoT.[52]

14.2.7 Building and home automation

IoT devices can be used to monitor and control the mechanical, electrical and electronic systems used in various types of buildings (e.g., public and private, industrial, institutions, or residential).[30] Home automation systems, like other building automation systems, are typically used to control lighting, heating, ventilation, air conditioning, appliances, communication systems, entertainment and home security devices to improve convenience, comfort, energy efficiency, and security.[53][54]

14.2.8 Transportation

The IoT can assist in integration of communications, control, and information processing across various transportation systems. Application of the IoT extends to all aspects of transportation systems, i.e. the vehicle, the infrastructure, and the driver or user. Dynamic interaction between these components of a transport system enables inter and intra vehicular communication, smart traffic control, smart parking, electronic toll collection systems, logistic and fleet management, vehicle control, and safety and road assistance.[30]

14.2.9 Large scale deployments

There are several planned or ongoing large-scale deployments of the IoT, to enable better management of cities and systems. For example, Songdo, South Korea, the first of its kind fully equipped and wired smart city, is near completion. Nearly everything in this city is planned to be wired, connected and turned into a constant stream of data that would be monitored and analyzed by an array of computers with little, or no human intervention.

Another application is a currently undergoing project in Santander, Spain. For this deployment, two approaches have been adopted. This city of 180000 inhabitants, has already seen 18000 city application downloads for their smartphones. This application is connected to 10000 sensors that enable services like parking search, environmental monitoring, digital city agenda among others. City context information is utilized in this deployment so as to benefit merchants through a spark deals mechanism based on city behavior that aims at maximizing the impact of each notification.<ref name=BUTLER-"SmartCity">Rico, Juan (22–24 April 2014). “Going beyond monitoring and actuating in large scale smart cities”. *NFC & Proximity Solutions - WIMA Monaco*.</ref>

Other examples of large-scale deployments underway include the Sino-Singapore Guangzhou Knowledge City;[55] work on improving air and water quality, reducing noise pollution, and increasing transportation efficiency in San Jose, California;[56] and smart traffic management in western Singapore.[57]

Another example of a large deployment is the one completed by New York Waterways in New York City to connect all their vessels and being able to monitor them live 24/7. The network was designed and engineered by Fluidmesh Networks, a Chicago based company developing wireless networks for mission critical applications. The NYWW network is currently providing coverage on the Hudson River, East River, and Upper New York Bay. With the wireless network in place, NY Waterway is able to take control of its fleet and passengers in a way that was not previously possible. New applications can include security, energy and fleet management, digital signage, public Wi-Fi, paperless ticketing and much more.

14.3 Unique addressability of things

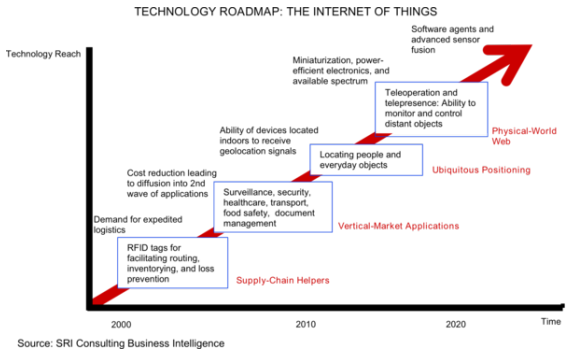
The original idea of the Auto-ID Center is based on RFID-tags and unique identification through the Electronic Product Code however this has evolved into objects having an IP address or URI.

An alternative view, from the world of the Semantic Web[58] focuses instead on making all things (not just those electronic, smart, or RFID-enabled) addressable by the existing naming protocols, such as URI. The objects themselves do not converse, but they may now be referred to by other agents, such as powerful centralized servers acting for their human owners.

The next generation of Internet applications using Internet Protocol Version 6 (IPv6) would be able to communicate with devices attached to virtually all human-made objects because of the extremely large address space of the IPv6 protocol. This system would therefore be able to scale to the large numbers of objects envisaged.[59]

A combination of these ideas can be found in the current GS1/EPCglobal EPC Information Services[60] (EPCIS) specifications. This system is being used to identify objects in industries ranging from aerospace to fast moving consumer products and transportation logistics.[61]

14.4 Trends and characteristics



Technology Roadmap: Internet of Things

14.4.1 Intelligence

Ambient intelligence and autonomous control are not part of the original concept of the Internet of Things. Ambient intelligence and autonomous control do not necessarily require Internet structures, either. However, there is a shift in research to integrate the concepts of the Internet of Things and autonomous control,[62] with initial outcomes towards this direction considering objects as the driving force for autonomous IoT.[63][64] In the future

the Internet of Things may be a non-deterministic and open network in which auto-organized or intelligent entities (Web services, SOA components), virtual objects (avatars) will be interoperable and able to act independently (pursuing their own objectives or shared ones) depending on the context, circumstances or environments. Autonomous behavior through collecting and reasoning context information plays a significant role in IoT. Modern IoT products and solutions in the marketplace use variety of different technologies to support such context-aware automation.[65]

Embedded intelligence[66] presents an "AI-oriented" perspective of Internet of Things, which can be more clearly defined as: leveraging the capacity to collect and analyze the digital traces left by people when interacting with widely deployed smart things to discover the knowledge about human life, environment interaction, as well as social inter connection and related behaviors.

14.4.2 Architecture

The system will likely be an example of event-driven architecture,[67] *bottom-up* made (based on the context of processes and operations, in real-time) and will consider any subsidiary level. Therefore, model driven and functional approaches will coexist with new ones able to treat exceptions and unusual evolution of processes (Multi-agent systems, B-ADSc, etc.).

In an Internet of Things, the meaning of an event will not necessarily be based on a deterministic or syntactic model but would instead be based on the context of the event itself: this will also be a semantic web.[68] Consequently, it will not necessarily need common standards that would not be able to address every context or use: some actors (services, components, avatars) will accordingly be self-referenced and, if ever needed, adaptive to existing common standards (*predicting everything* would be no more than defining a "global finality" for everything that is just not possible with any of the current *top-down* approaches and standardizations). Some researchers argue that sensor networks are the most essential components of the Internet of Things.[69]

Building on top of the Internet of Things, the Web of Things is an architecture for the application layer of the Internet of Things looking at the convergence of data from IoT devices into Web applications to create innovative use-cases.

14.4.3 Complex system

In semi-open or closed loops (i.e. value chains, whenever a global finality can be settled) it will therefore be considered and studied as a Complex system[70] due to the huge number of different links and interactions between autonomous actors, and its capacity to integrate new actors. At the overall stage (full open loop) it will likely be seen as a chaotic environment (since systems have always finality).

14.4.4 Size considerations

The Internet of objects would encode 50 to 100 trillion objects, and be able to follow the movement of those objects. Human beings in surveyed urban environments are each surrounded by 1000 to 5000 trackable objects.[71]

14.4.5 Space considerations

In an Internet of Things, the precise geographic location of a thing—and also the precise geographic dimensions of a thing—will be critical. Open Geospatial Consortium, "OGC Abstract Specification" Currently, the Internet has been primarily used to manage information processed by people. Therefore, facts about a thing, such as its location in time and space, have been less critical to track because the person processing the information can decide whether or not that information was important to the action being taken, and if so, add the missing information (or decide to not take the action). (Note that some things in the Internet of Things will be sensors, and sensor location is usually important. Mike Botts et al., "OGC Sensor Web Enablement: Overview And High Level Architecture") The GeoWeb and Digital Earth are promising applications that become possible when things can become organized and connected by location. However, challenges that remain include the constraints of variable spatial scales, the need to handle massive amounts of data, and an indexing for fast search and neighbour operations. If in the Internet of Things, things are able to take actions on their own initiative, this human-centric mediation role is eliminated, and the time-space context that we as humans take for granted must be given a central role in this information ecosystem. Just as standards play a key role in the Internet and the Web, geospatial standards will play a key role in the Internet of Things.

14.4.6 Sectors

There are three core sectors of the IoT: enterprise, home, and government, with the Enterprise Internet of Things (EIoT) being the largest of the three. By 2019, the EIoT sector is estimated to account for nearly 40% or 9.1 billion devices.[72]

14.4.7 A Basket of Remotes

According to the CEO of Cisco, the remote control market is expected to be a $USD 19 trillion market.[73] Many IoT devices have a potential to take a piece of this market. Jean-Louis Gassée (Apple initial alumni team, and

BeOS co-founder) has addressed this topic in an article on Monday Note,[74] where he predicts that the most likely problem will be what he calls the “Basket of remotes” problem, where we'll have hundreds of applications to interface with hundreds of devices that don't share protocols for speaking with one another.

There are multiple approaches to solve this problem, one of them called the "predictive interaction",[75] where cloud or fog based decision makers will predict the user's next action and trigger some reaction.

For user interaction, new technology leaders are joining forces to create standards for communication between devices. While AllJoyn alliance is composed the top 20 World technology leaders, there are also big companies that promote their own protocol like CCF from Intel.

This problem is also a competitive advantage for some very technical startup companies with fast capabilities.

- AT&T Digital Life provides one solution for the “basket of remotes” problem. This product features home-automation and digital-life experiences. It provides a mobile application to control their closed ecosystem of branded devices;
- Nuve has developed a new technology based on sensors, a cloud-based platform and a mobile application that allows the asset management industry to better protect, control and monitor their property.[76]
- Muzzley motd controls multiple devices with a single application[77] and has had many manufacturers use their API[78] to provide a learning ecosystem that really predicts the end-user next actions. Muzzley is known for being the first generation of platforms that has the ability to predict form learning the end-user outside World relations with “things”.
- my shortcut[79] is an approach that also includes a set of already-defined devices and allow a Siri-Like interaction between the user and the end devices. The user is able to control his or her devices using voice commands;[80]
- Realtek “IoT my things” is an application that aims to interface with a closed ecosystem of Realtek devices like sensors and light controls.

Manufacturers are becoming more conscious of this problem, and many companies have begun releasing their devices with open APIs. Many of these APIs are used by smaller companies looking to take advantage of quick integration.

14.5 Sub systems

Not all elements in an Internet of Things will necessarily run in a global space. Domotics running inside a Smart House, for example, might only run and be available via a local network.

14.6 Frameworks

Internet of Things frameworks might help support the interaction between “things” and allow for more complex structures like Distributed computing and the development of Distributed applications. Currently, some Internet of Things frameworks seem to focus on real time data logging solutions like Jasper Technologies, Inc. and Xively (formerly Cosm and before that Pachube): offering some basis to work with many “things” and have them interact. Future developments might lead to specific Software development environments to create the software to work with the hardware used in the Internet of Things. Companies such as B-Scada,[81][82] ThingWorx,[83][84][85] IoT-Ticket.com, Raco Wireless,[86][87] nPhase,[88] Carriots,[89][90] EVRYTHNG,[91] and Exosite[92][93][94] are developing technology platforms to provide this type of functionality for the Internet of Things.

The XMPP standards foundation XSF is creating such a framework in a fully open standard that isn't tied to any company and not connected to any cloud services. This XMPP initiative is called Chatty Things.[95] XMPP provides a set of needed building blocks and a proven distributed solution that can scale with high security levels. The extensions are published at XMPP/extensions

The independently developed MASH IoT Platform was presented at the 2013 IEEE IoT conference in Mountain View, CA. MASH's focus is asset management (assets=people/property/information, management=monitoring/control/configuration). Support is provided for design thru deployment with an included IDE, Android client and runtime. Based on a component modeling approach MASH includes support for user defined things and is completely data-driven.[96]

REST is a scalable architecture which allows for things to communicate over Hypertext Transfer Protocol and is easily adopted for IoT applications to provide communication from a thing to a central web server. MQTT is a publish-subscribe architecture on top of TCP/IP which allows for bi-directional communication between a thing and a MQTT broker.

14.7 Criticism and controversies

While many technologists tout the Internet of Things as a step towards a better world, scholars and social observers have doubts about the promises of the ubiquitous computing revolution.

14.7.1 Privacy, autonomy and control

Peter-Paul Verbeek, a professor of philosophy of technology at the University of Twente, Netherlands, writes that technology already influences our moral decision making, which in turns affects human agency, privacy and autonomy. He cautions against viewing technology merely as a human tool and advocates instead to consider it as an active agent.[97]

Justin Brookman, of the Center for Democracy and Technology, expressed concern regarding the impact of IoT on consumer privacy, saying that "There are some people in the commercial space who say, 'Oh, big data — well, let's collect everything, keep it around forever, we'll pay for somebody to think about security later.' The question is whether we want to have some sort of policy framework in place to limit that."[98]

Editorials at WIRED have also expressed concern, one stating 'What you're about to lose is your privacy. Actually, it's worse than that. You aren't just going to lose your privacy, you're going to have to watch the very concept of privacy be rewritten under your nose.'[99]

The American Civil Liberties Union (ACLU) expressed concern regarding the ability of IoT to erode people's control over their own lives. The ACLU wrote that "There's simply no way to forecast how these immense powers -- disproportionately accumulating in the hands of corporations seeking financial advantage and governments craving ever more control -- will be used. Chances are Big Data and the Internet of Things will make it harder for us to control our own lives, as we grow increasingly transparent to powerful corporations and government institutions that are becoming more opaque to us."[100]

Researchers have identified privacy challenges faced by all stakeholders in IoT domain, from the manufacturers and app developers to the consumers themselves, and examined the responsibility of each party in order to ensure user privacy at all times. Problems highlighted by the report[101] include:

- User consent – somehow, the report says, users need to be able to give informed consent to data collection. Users, however, have limited time and technical knowledge.
- Freedom of choice – both privacy protections and underlying standards should promote freedom of choice. For example, the study notes,[102] users need a free choice of vendors in their smart homes; and they need the ability to revoke or revise their privacy choices.
- Anonymity – IoT platforms pay scant attention to user anonymity when transmitting data, the researchers note. Future platforms could, for example, use TOR or similar technologies so that users can't be too deeply profiled based on the behaviors of their "things".

14.7.2 Security

A different criticism is that the Internet of Things is being developed rapidly without appropriate consideration of the profound security challenges involved and the regulatory changes that might be necessary.[103] According to the BI (Business Insider) Intelligence Survey conducted in the last quarter of 2014, 39% of the respondents said that security is the biggest concern in adopting Internet of Things technology.[104] In particular, as the Internet of Things spreads widely, cyber attacks are likely to become an increasingly physical (rather than simply virtual) threat.[105] In a January 2014 article in *Forbes*, cybersecurity columnist Joseph Steinberg listed many Internet-connected appliances that can already "spy on people in their own homes" including televisions, kitchen appliances, cameras, and thermostats.[106] Computer-controlled devices in automobiles such as brakes, engine, locks, hood and truck releases, horn, heat, and dashboard have been shown to be vulnerable to attackers who have access to the onboard network. (These devices are currently not connected to external computer networks, and so are not vulnerable to Internet attacks.)[107]

The U.S. National Intelligence Council in an unclassified report maintains that it would be hard to deny "access to networks of sensors and remotely-controlled objects by enemies of the United States, criminals, and mischief makers… An open market for aggregated sensor data could serve the interests of commerce and security no less than it helps criminals and spies identify vulnerable targets. Thus, massively parallel sensor fusion may undermine social cohesion, if it proves to be fundamentally incompatible with Fourth-Amendment guarantees against unreasonable search."[108] In general, the intelligence community views Internet of Things as a rich source of data.[109]

14.7.3 Design

Given widespread recognition of the evolving nature of the design and management of the Internet of Things, sustainable and secure deployment of Internet of Things solutions must design for "anarchic scalability."[110] Application of the concept of anarchic scalability can be extended to physical systems (i.e. controlled real-world objects), by virtue of those systems being designed to account for uncertain management futures. This "hard anarchic scalabilty" thus provides a pathway forward to fully realize the potential of Internet of Things solutions by selectively constraining physical systems to allow for all management regimes without risking physical failure.

Brown University computer scientist Michael Littman has argued that successful execution of the Internet of

Things requires consideration of the interface's usability as well as the technology itself. These interfaces need to be not only more user friendly but also better integrated: "If users need to learn different interfaces for their vacuums, their locks, their sprinklers, their lights, and their coffeemakers, it's tough to say that their lives have been made any easier."[111]

14.7.4 Environmental impact

A concern regarding IoT technologies pertains to the environmental impacts of the manufacture, use, and eventual disposal of all these semiconductor-rich devices. Modern electronics are replete with a wide variety of heavy metals and rare-earth metals, as well as highly toxic synthetic chemicals. This makes them extremely difficult to properly recycle. Electronic components are often simply incinerated or dumped in regular landfills, thereby polluting soil, groundwater, surface water, and air. Such contamination also translates into chronic human-health concerns. Furthermore, the environmental cost of mining the rare-earth metals that are integral to modern electronic components continues to grow. With production of electronic equipment growing globally yet little of the metals (from end-of-life equipment) being recovered for reuse, the environmental impacts can be expected to increase.

Also, because the concept of IoT entails adding electronics to mundane devices (for example, simple light switches), and because the major driver for replacement of electronic components is often technological obsolescence rather than actual failure to function, it is reasonable to expect that items that previously were kept in service for many decades would see an accelerated replacement cycle, if they were part of the IoT. For example, a traditional house built with 30 light switches and 30 electrical outlets might stand for 50 years, with all those components still being original at the end of that period. But a modern house built with the same number of switches and outlets set up for IoT might see each switch and outlet replaced at five-year intervals, in order to keep up-to-date with technological changes. This translates into a ten-fold increase in waste requiring disposal.

While IoT devices can serve as energy-conservation equipment, it is important to keep in mind that everyday good habits can bring the same benefits. Practical, fundamental considerations such as these are often overlooked by marketers eager to induce consumers to purchase IoT items that may never have been needed in the first place.

14.8 See also

- Web of Things
- Algorithmic Regulation
- Cloud manufacturing
- Connected Revolution
- Data Distribution Service
- Digital Object Memory
- Hyper Text Coffee Pot Control Protocol
- Industry 4.0
- INSTEON
- RoboEarth
- Smart, Connected Products
- SMPTE ST2071
- Skynet (Terminator)
- Transreality gaming
- Wearable technology

14.9 References

[1] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, D. Boyle: *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence.* Elsevier, 2014, ISBN 978-0-12-407684-6.

[2] O. Monnier: *A smarter grid with the Internet of Things.* Texas Instruments, 2013.

[3] http://molluscan-eye.epoc.u-bordeaux1.fr/index.php?rubrique=accueil&lang=en/

[4] I. Wigmore: "Internet of Things (IoT)". TechTarget, June 2014.

[5] Farooq, M.U.; Waseem, Muhammad; Khairi, Anjum; Mazhar, Sadia (2015). "A Critical Analysis on the Security Concerns of Internet of Things (IoT)". *International Journal of Computer Applications (IJCA)* **11**. Retrieved 18 February 2015.

[6] Violino, Bob. "The 'Internet of things' will mean really, really big data". *InfoWorld.* Retrieved 9 July 2014.

[7] Hogan, Michael. "The 'The Internet of Things Database' Data Management Requirements". *ScaleDB.* Retrieved 15 July 2014.

[8] "The "Only" Coke Machine on the Internet". Carnegie Mellon University. Retrieved 10 November 2014.

[9] "Internet of Things Done Wrong Stifles Innovation". InformationWeek. 7 July 2014. Retrieved 10 November 2014.

[10] Weiser, Mark (1991). "The Computer for the 21st Century". *Scientific American* **265** (3): 94–104. Retrieved 5 November 2014.

[11] Raji, RS (June 1994). "Smart networks for control". *IEEE Spectrum*.

[12] Jason Pontin: ETC: Bill Joy's Six Webs. In: *MIT Technology Review*, 29 September 2005. Retrieved 17 November 2013.

[13] Analyst Anish Gaddam interviewed by Sue Bushell in *Computerworld*, on 24 July 2000 ("M-commerce key to ubiquitous internet")

[14] P. Magrassi, T. Berg, *A World of Smart Objects*, Gartner research report R-17-2243, 12 August 2002

[15] Commission of the European Communities (18 June 2009). "Internet of Things — An action plan for Europe" (PDF). COM(2009) 278 final.

[16] Techvibes *From M2M to The Internet of Things: Viewpoints From Europe* 7 July 2011

[17] Dr. Lara Sristava, European Commission Internet of Things Conference in Budapest, 16 May 2011 *The Internet of Things - Back to the Future (Presentation)*

[18] P. Magrassi, A. Panarella, N. Deighton, G. Johnson, *Computers to Acquire Control of the Physical World*, Gartner research report T-14-0301, 28 September 2001

[19] Casaleggio Associati *The Evolution of Internet of Things* February 2011

[20] "Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020". Gartner. 12 December 2013. Retrieved 2 January 2014.

[21] More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020, ABI Research

[22] Fickas, S.; Kortuem, G.; Segall, Z. (13–14 Oct 1997). "Software organization for dynamic and adaptable wearable systems". *International Symposium on Wearable Computers*: 56–63. doi:10.1109/ISWC.1997.629920.

[23] Main Report: An In-depth Look at Expert Responses|Pew Research Center's Internet & American Life Project

[24] http://www.theconnectivist.com/2014/05/infographic-the-growth-of-the-internet-of-things/

[25] Kushalnagar, N; Montenegro, G; Schumacher, C (August 2007). "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals". *IETF RFC 4919*.

[26] Sun, Charles C. (1 May 2014). "Stop using Internet Protocol Version 4!". *Computerworld*.

[27] Sun, Charles C. (2014-05-01). "Stop Using Internet Protocol Version 4!". CIO. Retrieved 2015-01-28.

[28] Sun, Charles C. (2 May 2014). "Stop using Internet Protocol Version 4!". *InfoWorld*.

[29] Sun, Charles C. (1 May 2014). "Stop using Internet Protocol Version 4!". *IDG News India*.

[30] Ersue, M; Romascanu, D; Schoenwaelder, J; Sehgal, A (4 July 2014). "Management of Networks with Constrained Devices: Use Cases". *IETF Internet Draft < draft-ietf-opsawg-coman-use-cases>*.

[31] Vongsingthong, S.; Smanchat, S. (2014). "Internet of Things: A review of applications & technologies". *Suranaree Journal of Science and Technology*.

[32] Mitchell, Shane; Villa, Nicola; Stewart-Weeks, Martin; Lange, Anne. "The Internet of Everything for Cities: Connecting People, Process, Data, and Things To Improve the 'Livability' of Cities and Communities". Cisco Systems. Retrieved 10 July 2014.

[33] Narayanan, Ajit. "Impact of Internet of Things on the Retail Industry". *PCQuest*. Cyber Media Ltd. Retrieved 20 May 2014.

[34] CasCard; Gemalto; Ericsson. "Smart Shopping: spark deals". *EU FP7 BUTLER Project*.

[35] Kyriazis, D.; Varvarigou, T.; Rossi, A.; White, D.; Cooper, J. (4–7 June 2013). "Sustainable smart city IoT applications: Heat and electricity management & Eco-conscious cruise control for public transportation". *IEEE International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. doi:10.1109/WoWMoM.2013.6583500.

[36] Perera, Charith; Liu, Harold; Jayawardena, Srimal. "The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey". *Emerging Topics in Computing, IEEE Transactions on*. PrePrint. doi:10.1109/TETC.2015.2390034. Retrieved 1 February 2015.

[37] Couldry, Nick; Turow, Joseph (2014). "Advertising, Big Data, and the Clearance of the Public Realm: Marketers' New Approaches to the Content Subsidy". *International Journal of Communication* **8**: 1710–1726.

[38] Meadows-Klue, Danny. "A new era of personal data unlocked in an "Internet of Things"". *http://www.digitalstrategyconsulting.com". Digital Strategy Consulting. Retrieved 26 January 2015.*

[39] Li, Shixing; Wang, Hong; Xu, Tao; Zhou, Guiping (2011). "Application Study on Internet of Things in Environment Protection Field". *Lecture Notes in Electrical Engineering Volume* **133**: 99–106. doi:10.1007/978-3-642-25992-0_13.

[40] FIT French Project. "Use case: Sensitive wildlife monitoring". Retrieved 10 July 2014.

[41] http://en.wikipedia.org/wiki/Resource

[42] Gubbi, Jayavardhana; Buyya, Rajkumar; Marusic, Slaven; Palaniswami, Marimuthu (24 February 2013). "Internet of Things (IoT): A vision, architectural elements, and future directions". *Future Generation Computer Systems* **29** (7): 1645–1660. doi:10.1016/j.future.2013.01.010.

[43] Chui, Michael; Löffler, Markus; Roberts, Roger. "The Internet of Things". *McKinsey Quarterly*. McKinsey & Company. Retrieved 10 July 2014.

[44] Postscapes. "Smart Trash". Retrieved 10 July 2014.

[45] Severi, S.; Abreu, G.; Sottile, F.; Pastrone, C.; Spirito, M.; Berens, F. (23–26 June 2014). "M2M Technologies: Enablers for a Pervasive Internet of Things". *The European Conference on Networks and Communications (EUCNC2014)*.

[46] Tan, Lu; Wang, Neng (20–22 August 2010). "Future Internet: The Internet of Things". *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)* **5**: 376–380. doi:10.1109/ICACTE.2010.5579543.

[47] Parello, J.; Claise, B.; Schoening, B.; Quittek, J. (28 April 2014). "Energy Management Framework". *IETF Internet Draft <draft-ietf-eman-framework-19>*.

[48] "Why Wemo?". Belkin. Retrieved 30 January 2015.

[49] "Professional 4-Port Remote Power Switch - Phone Control + Web Control". Ambery.

[50] "Budderfly - The Power to Manage ALL YOUR ENERGY".

[51] Istepanian, R.; Hu, S.; Philip, N.; Sungoor, A. (30 August – 3 September 2011). "The potential of Internet of m-health Things "m-IoT" for non-invasive glucose level sensing". *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. doi:10.1109/IEMBS.2011.6091302.

[52] Swan, Melanie (8 November 2012). "Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0". *Sensor and Actuator Networks* **1** (3): 217–253. doi:10.3390/jsan1030217.

[53] Alkar, A.Z.; Buhur, U. (November 2005). "An Internet based wireless home automation system for multifunctional devices". *IEEE Transactions on Consumer Electronics* **51** (4): 1169–1174. doi:10.1109/TCE.2005.1561840.

[54] Spiess, P.; Karnouskos, S.; Guinard, D.; Savio, D.; Baecker, O.; Souza, L.; Trifa, V. (6–10 July 2009). "SOA-Based Integration of the Internet of Things in Enterprise Services". *IEEE International Conference on Web Services (ICWS)*: 968–975. doi:10.1109/ICWS.2009.98.

[55] "Sino-Singapore Guangzhou Knowledge City: A vision for a city today, a city of vision tomorrow". Retrieved 11 July 2014.

[56] "San Jose Implements Intel Technology for a Smarter City". Retrieved 11 July 2014.

[57] Coconuts Singapore. "Western Singapore becomes testbed for smart city solutions". Retrieved 11 July 2014.

[58] Dan Brickley et al., c. 2001

[59] Waldner, Jean-Baptiste (2008). *Nanocomputers and Swarm Intelligence*. London: ISTE. pp. p227–p231. ISBN 1-84704-002-0.

[60] "EPCIS - EPC Information Services Standard". GS1. Retrieved 2 January 2014.

[61] Miles, Stephen B. (2011). *RFID Technology and Applications*. London: Cambridge University Press. pp. 6–8. ISBN 978-0-521-16961-5.

[62] Uckelmann, Dieter; Isenberg, Marc-André; Teucke, Michael; Halfar, Harry; Scholz-Reiter, Bernd (2010). "An integrative approach on Autonomous Control and the Internet of Things". In Ranasinghe, Damith; Sheng, Quan; Zeadally, Sherali. *Unique Radio Innovation for the 21st Century: Building Scalable and Global RFID Networks*. Berlin, Germany: Springer. pp. 163–181. ISBN 978-3-642-03461-9. Retrieved 28 April 2011.

[63] Kortuem, G.; Kawsar, F.; Fitton, D.; Sundramoorthy, V. (Jan–Feb 2010). "Smart objects as building blocks for the Internet of things". *IEEE Internet Computing*: 44–51. doi:10.1109/MIC.2009.143.

[64] Kyriazis, D.; Varvarigou, T. (21–23 Oct 2013). "Smart, Autonomous and Reliable Internet of Things". *International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN)*. doi:10.1016/j.procs.2013.09.059.

[65] Perera, Charith; Liu, Harold; Jayawardena, Srimal; Chen, Min. "A Survey on Internet of Things From Industrial Market Perspective". *Access, IEEE* **2**: 1660–1679. doi:10.1109/ACCESS.2015.2389854. Retrieved 1 February 2015.

[66] "Living with Internet of Things, The Emergence of Embedded Intelligence (CPSCom-11)". Bin Guo. Retrieved 6 September 2011.

[67] Philippe Gautier, « RFID et acquisition de données évènementielles : retours d'expérience chez Bénédicta », pages 94 à 96, Systèmes d'Information et Management - revue trimestrielle N°2 Vol. 12, 2007, ISSN 1260-4984 / ISBN 978-2-7472-1290-8, éditions ESKA.

[68] "3 questions to Philippe Gautier, by David Fayon, march 2010"

[69] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos (2013). "Context Aware Computing for The Internet of Things: A Survey". *Communications Surveys Tutorials, IEEE* **PP** (n/a): 1–44. doi:10.1109/SURV.2013.042313.00197.

[70] Gautier, Philippe; Gonzalez, Laurent (2011). *L'Internet des Objets… Internet, mais en mieux*. foreword by Gérald Santucci (European commission), postword by Daniel Kaplan (FING) and Michel Volle. Paris: AFNOR editions. ISBN 978-2-12-465316-4.

[71] Waldner, Jean-Baptiste (2007). *Nanoinformatique et intelligence ambiante. Inventer l'Ordinateur du XXIeme Siècle*. London: Hermes Science. pp. p254. ISBN 2-7462-1516-0.

[72]

[73] Cisco CEO says it will be a 19 trillion dollar market

[74] Jean-Louis Gassée opinion

[75] intel predictive interaction analysis

[76] IoT for the Asset Management Industry

[77] Integrations with a world of IoT's like Nest, Belkin WeMo and others

[78] API's for joining the ecosystem

[79] his shortcust website

[80] TechCrunch debuts a Siri-Like IoT app

[81] Boccamazzo, Allison (28 January 2015). "B-Scada Launches New IoT Initiative at ITEXPO 2015". *TMCnet*.

[82] "B-Scada Takes SCADA to the Cloud". *Automation.com*. Retrieved 13 February 2015.

[83] Rizzo, Tony (12 March 2013). "ThingWorx Drives M2M and IoT Developer Efficiency with New Platform Release". *TMCnet*.

[84] Bowen, Suzanne. "ThingWorx CEO Russell Fadel on M2M and the Connected World". *DIDX Audio Podcast Newspaper*. Retrieved 9 April 2013.

[85] "ThingWorx".

[86] Bowen, Suzanne. "Raco Wireless John Horn on the Connected World and M2M". *DIDX Audio Podcast Newspaper*. Retrieved 9 April 2013.

[87] Fitchard, Kevin (26 February 2013). "T-Mobile's M2M provider Raco goes international with Sprint, Telefónica deals". *GigaOm*.

[88] Bowen, Suzanne. "Interview with nPhase (Qualcomm - Verizon) Steve Pazol on M2M". *DIDX Audio Podcast Newspaper*. Retrieved 9 April 2013.

[89] "What is Carriots". *Carriots official site*. Retrieved 10 October 2013.

[90] Higginbotham, Stacey. "Carriots is building a PaaS for the Internet of Things". *GigaOM*. Retrieved 26 April 2013.

[91] "IoT Startup EVRYTHNG Secures $7M Series A From Atomico, BHLP, Cisco And Dawn". *Techcrunch*.

[92] Katharine Greyson (22 October 2013). "Is Minn.'s next big thing the Internet of Things?". Minneapolis-Saint Paul Business Journal.

[93] "Exosite: Extending Big Data to Next Generation Of Cloud Solutions". CIOReview.

[94] Bill Wong (1 May 2014). "Dev Kits Light Up The Internet Of Things". Electronic Design.

[95] XMPP IoT systems

[96] http://www.youtube.com/user/MASHPlatform "YouTube channel"

[97] Verbeek, Peter-Paul. "Moralizing Technology: Understanding and Designing the Morality of Things." Chicago: *The University of Chicago Press,* 2011.

[98] Diane Cardwell, At Newark Airport, the Lights Are On, and They're Watching You, *The New York Times,* 2014.02.17

[99] Webb, Geoff (5 February 2015). "Say Goodbye to Privacy". WIRED. Retrieved 15 February 2015.

[100] Catherine Crump and Matthew Harwood, The Net Closes Around Us, *TomDispatch*, 25 March 2014

[101] Perera, Charith; Ranjan, Rajiv; Wang, Lizhe; Khan, Samee; Zomaya, Albert (2015). "Privacy of Big Data in the Internet of Things Era". *IEEE IT Professional Magazine*. PrePrint (Internet of Anything). Retrieved 1 February 2015.

[102] Perera, Charith; Zaslavsky, Arkady (8 March 2014). "Improve the sustainability of Internet of Things through trading-based value creation". *Internet of Things (WF-IoT), 2014 IEEE World Forum on*: 135–140. doi:10.1109/WF-IoT.2014.6803135. Retrieved 1 February 2015.

[103] Christopher Clearfield Why The FTC Can't Regulate The Internet Of Things, Forbes, 18 September 2013

[104] http://www.businessinsider.in/ We-Asked-Executives-About-The-Internet-Of-Things-And-Their-Answers articleshow/45959921.cms

[105] Christopher Clearfield "Rethinking Security for the Internet of Things" Harvard Business Review Blog, 26 June 2013/

[106] Joseph Steinberg (27 January 2014). "These Devices May Be Spying On You (Even In Your Own Home)". Forbes. Retrieved 27 May 2014.

[107] http://www.popsci.com/cars/article/2010-05/ researchers-hack-car-computers-shutting-down-brakes-engine-and-more

[108] Disruptive Technologies Global Trends 2025. National Intelligence Council (NIC), April 2008, P. 27.

[109] Spencer Ackerman. CIA Chief: We'll Spy on You Through Your Dishwasher. Wired, 15 March. 2012.

[110] Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures (2000), Dissertation - Doctor of Philosophy in Information and Computer Science

[111] Littman, Michael and Samuel Kortchmar. "The Path To A Programmable World". *Footnote*. Retrieved 14 June 2014.

14.10 Further reading

- Atzori, Luigi and Iera, Antonio and Morabito, Giacomo. "The internet of things: A survey". Computer Networks, Elsevier, The Netherlands, 2010.
- Carsten, Paul (2015). "Lenovo to stop pre-installing controversial software". Reuters.
- Chaouchi, Hakima. The Internet of Things. London: Wiley-ISTE, 2010.

- Chabanne, Herve, Pascal Urien, and Jean-Ferdinand Susini. RFID and the Internet of Things. London: ISTE, 2011.
- "Disruptive Technologies Global Trends 2025". U.S. National Intelligence Council (NIC).
- Fell, Mark (2014). "Roadmap for the Emerging Internet of Things - Its Impact, Architecture and Future Governance". Carré & Strauss, United Kingdom.
- Fell, Mark (2013). "Manifesto for Smarter Intervention in Complex Systems". Carré & Strauss, United Kingdom.
- Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami (September 2013). "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions". Future Generation Computer Systems, Elsevier, The Netherlands.
- Hersent, Olivier, David Boswarthick and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Chichester, West Sussex: Wiley, 2012.
- "Internet of Things in 2020: A Roadmap for the future". EPoSS.
- IERC - European Research Cluster on the Internet of Things: Documents and Publications
- Michahelles, Florian, et al. Proceedings of 2012 International Conference on the Internet of Things (IOT) : 24–26 October 2012 : Wuxi, China. Piscataway, N.J.: IEEE, 2012.
- "What is the Internet of Things? An Economic Perspective". Auto-ID Labs.
- Pfister, Cuno. Getting Started with the Internet of Things. Sebastapool, Calif: O'Reilly Media, Inc, 2011.
- Uckelmann, Dieter, Mark Harrison and Florian Michahelles. Architecting the Internet of Things. Berlin: Springer, 2011.
- Weber, Rolf H., and Romana Weber. Internet of Things: Legal Perspectives. Berlin: Springer, 2010.
- Zhou, Honbo. The Internet of Things in the Cloud: A Middleware Perspective. Boca Raton: CRC Press, Taylor & Francis Group, 2013.

14.11 External links

- "A New Economic Vision for Addressing Climate Change (Internet of things - part II)". (2014-06-02) and "Monopoly Capitalism vs. Collaborative Commons (Internet of things - part I)". (2014-04-07)
- Pew Internet canvas of experts, prognosticating on the nature, application, and impact of the Internet of Things in 2025
- "The Creepy New Wave of the Internet (Internet of things)". (2014-11-02), *New York Review of Books*
- Pros of IOT, possible advantages of IOT
- IBM. "IBM view on the Internet of Things". IBM.
- www.internet-of-things.eu
- The IoT Council

14.12 Text and image sources, contributors, and licenses

14.12.1 Text

- **Cloud computing** *Source:* http://en.wikipedia.org/wiki/Cloud%20computing?oldid=650566982 *Contributors:* Zundark, ChangChienFu, Heron, Jose Icaza, Jdlh, Michael Hardy, Mahjongg, Rw2, Haakon, Ronz, Julesd, Andrewman327, DJ Clayworth, Tpbradbury, Furrykef, Saltine, Fvw, Dbabbitt, Tkdcoach, Rossumcapek, Chealer, Lapax, Rursus, SC, Jleedev, Superm401, Tobias Bergemann, Lysy, Martinwguy, Giftlite, Metapsyche, Smjg, Graeme Bartlett, Ryanrs, HangingCurve, Mckaysalisbury, DavidLam, Utcursch, SoWhy, Pgan002, SarekOfVulcan, Beland, Bumm13, Sfoskett, Xinconnu, Axelangeli, Now3d, ShortBus, Chem1, Thorwald, Mike Rosoft, Slady, Discospinster, Rich Farmbrough, Hydrox, YUL89YYZ, Bender235, ESkog, Neko-chan, Syp, Shanes, Jpgordon, Shax, Sanjiv swarup, Richi, CKlunck, Justinc, Mdd, Alansohn, Gary, Csabo, Richard Harvey, Tobych, Free Bear, Kessler, Diego Moya, Andrewpmk, Ricky81682, Ashley Pomeroy, Snowolf, Wtmitchell, Velella, Wtshymanski, Stephan Leeds, RubenSchade, LFaraone, Blaxthos, Richwales, Walshga, Oleg Alexandrov, Skrewler, Stuartyeates, Brianwc, Lordfaust, Firsfron, Woohookitty, Mindmatrix, RHaworth, TheNightFly, Ruud Koot, WadeSimMiser, Trödel, Cbdorsett, GregorB, Dogsbody, SPACEBAR, Littlewild, Mandarax, SqueakBox, BD2412, Pmj, Jorunn, Rjwilmsi, Nightscream, Wooptoo, Salix alba, MZMcBride, Vegaswikian, Bodhran, ElKevbo, Bubba73, The wub, Nicolas1981, FayssalF, Makru, Windchaser, Jmc, Nogburt, Crazycomputers, Jacob1044, A.K.Karthikeyan, Intgr, David H Braun (1964), Ahunt, Imnotminkus, Chobot, DVdm, Guliolopez, Wavelength, RussBot, Bhny, Stephenb, Manop, SteveLoughran, Rsrikanth05, Bovineone, Tungsten, SamJohnston, LandoSr, Gram123, NawlinWiki, Dialectric, Grafen, Welsh, Hogne, Akropp, Dethomas, PhilipC, Moe Epsilon, Tony1, Jerome Kelly, Wizzard, Jeh, Sarathc, Bikeborg, Yonidebest, Rolf-Peter Wille, Zzuuzz, Sissyneck, Timwayne, E Wing, Juliano, JLaTondre, DoriSmith, Allens, Katieh5584, Snaxe920, Otto ter Haar, Bernd in Japan, Liujiang, Tom Morris, Victor falk, Kimdino, DanStern, Luk, Mgaffney, Palapa, SmackBot, Ashley thomas80, JoshDuffMan, McGeddon, Gigs, PhilJackson, CastAStone, C.Fred, Elminster Aumar, Davewild, WookieInHeat, Jab843, AnOddName, Lainagier, Yamaguchi??, Gilliam, Ohnoitsjamie, Skizzik, Samveen, Kawana, Rmosler2100, Chris the speller, Bidgee, Ebhakt, Thumperward, Siddii, RayAYang, Deli nk, Jerome Charles Potts, Dlohcierekim's sock, Letdorf, Nbarth, Colonies Chris, A. B., John Reaves, Scwlong, Wynand.winterbach, Nabeez, Mike hayes, Tped, Frap, StefanB sv, Jacob Poon, OSborn, Uozef, Billytkid, GVnayR, LuchoX, Abrahami, Speedplane, Valenciano, Preetesh.rao, Dreadstar, Drphilharmonic, DMacks, Shswanson, Vina-iwbot, Bejnar, Vasiliy Faronov, Spiritia, KenCavallon, Acrooney, ArglebargleIV, AbdullahHaydar, Harryboyles, Gandalf44, JzG, Kuru, Oskilian, Tomhubbard, Gobonobo, Darktemplar, Robofish, JoshuaZ, Kashmiri, Minna Sora no Shita, IronGargoyle, Ckatz, Kompere, Beetstra, Mr Stephen, Ehheh, Larrymcp, Optakeover, Waggers, TastyPoutine, Dr.K., Kvng, Belfry, Keahapana, Hu12, Meitar, Quaeler, Spo0nman, Jonasalmeida, IvanLanin, UncleDouggie, Rnb, Mjboniface, Majora4, Courcelles, Dlohcierekim, Chris55, Patrickwooldridge, FatalError, JForget, VoxLuna, Ourhistory153, Randhirreddy, Earthlyreason, Eric, JohnCD, Bill.albing, Kmsmgill, NaBUru38, Flood6, Sanspeur, WeisheitSuchen, Alexamies, Myasuda, Metatinara, Jehfes, Rotiro, Yaris678, Cydebot, Mblumber, MC10, UncleBubba, Anthonyhcole, GRevolution824, Dancter, Clovis Sangrail, Christian75, Ameliorate!, Kozuch, Neustradamus, Casliber, Malleus Fatuorum, Thijs!bot, Epbr123, Kubanczyk, Dschrader, Wikid77, Vicweast, Shoaibnz, Ugarit, Vondruska, Vertium, John254, James086, Edchi, EdJohnston, Nick Number, Bob.gourley@comcast.net, Heroeswithmetaphors, Tree Hugger, Dawnseeker2000, Escarbot, Porqin, MrMarmite, Seaphoto, Shirt58, Marokwitz, Smartse, Dinferno, Silver seren, MrKG, Lbecque, DaudSharif, Tangurena, Dougher, Barek, MER-C, Dsp13, Jldupont, MB1972, Mwarren us, Rms77, Ispabierto, Greensburger, East718, Ny156uk, Spojrzenie, Magioladitis, Swikid, Bongwarrior, Lmbhull, JamesBWatson, Mathematrucker, GaryGo, Steven Walling, ForthOK, Jeffsnox, Hamiltonstone, Be-nice:-), Pleft, Kibbled bits, Cpl Syx, Balaarjunan, SBunce, JaGa, Kgfleischmann, Philg88, Pikolas, Zevnik, Curtbeckmann, Pisapatis, Dezrtluver, CliffC, Iamthenewno2, Casieg, CitizenB, Parveson, Jack007, Xiler, Bus stop, Vermtt, Miguelcaldas, Alankc, Mariolina, Linuxbabu, JonathonReinhart, Tgeairn, J.delanoy, PCock, Trusilver, Anandcv, Vpfaiz, Uncle Dick, Maurice Carbonaro, Jesant13, Ginsengbomb, Mathglot, Jarrad Lewis, Tsmitty31, Betswiki, Tonyshan, Staceyeschneider, NewEnglandYankee, Quantling, BostonRed, Biglovinb, Olegwiki, Bshende, KylieTastic, Raspalchima, HenryLarsen, Paulmmn, Songjin, Bonadea, Pegordon, Swolfsg, Idioma-bot, Laurenced, Martin.ashcroft, Imtiyazali4all, Bobwhitten, Obdurodon, Huygens 25, Vranak, 28bytes, VolkovBot, Jeff G., Dogbertwp, Edeskonline, Bkengland, Priyo123, FatUglyJo, Nyllo, Philip Trueman, A.Ward, TXiKiBoT, Itangalo, Vipinhari, Technopat, Guillaume2303, Anonymous Dissident, Danielchalef, Markus95, Markfetherolf, GcSwRhIc, Monkey Bounce, Piperh, Rich Janis, Felipebm, Martin451, Broadbot, Willit63, Amog, Figureskatingfan, Everything counts, SpecMode, Johnpltsui, Andy Dingley, Finngall, Haseo9999, Lamro, Garima.rai30, The Seventh Taylor, Falcon8765, VanishedUserABC, Nelliejellynoonaa, Sapenov, LittleBenW, Jimmi Hugh, Logan, OsamaK, Biscuittin, SieBot, Skyrail, Moonriddengirl, EwokiWiki, Doctorfree, Sakaal, Dawn Bard, Timothy Cooper, Navywings, Yintan, SuzanneIAM, Kpalsson, Jerryobject, Fishtron, Keilana, Chmyr, Heyitscory, Bentogoa, Happysailor, Flyer22, Jojalozzo, Nopetro, Snideology, Yerpo, Reservoirhill, OsamaBinLogin, Dominik92, Xe7al, North wiki, Techman224, Vykk, Rosiestep, Fuddle, StaticGull, Classivertsen, Bijoysr, WikiLaurent, Superbeecat, Laser813, Shinerunner, Denisarona, Motyka, Dlrohrer2003, Martarius, Simonmartin74, Elassint, ClueBot, GorillaWarfare, Wasami007, The Thing That Should Not Be, Cdhkmmaes, Nnemo, Czarkoff, Axorc, Jasapir, Drmies, VQuakr, Mild Bill Hiccup, Myokobill, Allenmwnc, Enc1234, LizardJr8, Bob bobato, Darren uk, Esthon, Auntof6, 718 Bot, Pointillist, Jonathan.robie, Loadbang, Stuart.clayton.22, Ktr101, Excirial, Pumpmeup, Alexbot, Jusdafax, Sajeer50, Hfoxwell, Eeekster, Nasonmedia, Muhandes, SunnySideOfStreet, Technobadger, 842U, Cmartell, M.O.X, Razorflame, Jinlye, SchreiberBike, Five-toed-sloth, Craig.Coward, Jakemoilanen, Vdmeraj, PCHS-NJROTC, Johnuniq, Vigilius, DumZiBoT, Jack Bauer00, Steveozone, Darkicebot, Beltman R., Lorddunvegan, XLinkBot, AgnosticPreachersKid, Roxy the dog, Njkool, Stickee, Sponsion, Feinoha, Chanakal, Bpgriner, C. A. Russell, Avoided, Fergus Cloughley, Imllorente, Skarebo, WikHead, Galzigler, Mifter, PcCoffee, Jbeans, Eleven even, Jht4060, NonNobisSolum, Richard.McGuire88, Sandipk singh, RealWorldExperience, Y2l2, Edepa, B Fizz, Dbrisinda, Deineka, Bazj, Addbot, American Eagle, TimFreeman701, Ramu50, Mortense, Realtimer, Sean R Fox, Mabdul, IXavier, VijayKrishnaPV, Fcalculators, Mkdonqui, Amore proprio, Tanhabot, Barmijo, TutterMouse, Fieldday-sunday, Scientus, Shakeelrashed, CanadianLinuxUser, Ethoslight, Kristiewells, Cst17, Mohamed Magdy, MrOllie, Download, Robert.Harker, Hatfields, Glane23, Mhodapp, Glass Sword, JimDelRossi, Favonian, Optatus, Stbrodie1, Terrillja, Numbo3-bot, Superkillball, Cybercool10, HandThatFeeds, Ashleymcneff, Tide rolls, דוד ש, Avono, NeD80, Hunyadym, Luckas Blade, Teles, Cloudcoder, Jarble, Mlavannis, Shri ram r, HerculeBot, Enaiburg, Gamber34, Legobot, Avlnet, Jerichochang97, Luckas-bot, BaldPark, ZX81, Yobot, Evagarfer, Themfromspace, Dfxdeimos, Legobot II, Librsh, Jamalystic, Bruce404, Asieo, Indigokk, Reshadipoor, Washburnmav, Identity20, Adam Hauner, Imeson, Javaeu, Thesurfpup, Achimew, Lerichard, Knoxi171, ByM4k5, Tiburondude, Aburreson, Jean.julius, Sweerek, Peter Flass, Sql er2, WikiScrubber, Sivanesh, IANYL, Deicool, AnomieBOT, Momoricks, Dmichaud, Pgj1997, 1exec1, Cronos4d, ThaddeusB, Jim1138, IHSscj, JackieBot, Iamdavinci, CloudComputing, Yaraman, Mbblake, AdityaTandon, Csigabi, Felixchu, Materialscientist, RobertEves92, JamesLWilliams2010, The High Fin Sperm Whale, Citation bot, Jkelleyy, OllieFury, Shan.rajad23, ArthurBot, Quebec99, YoungManBlues, NW's Public Sock, PavelSolin, LemonairePaides, Mwmaxey, Xqbot, L200817s, Alexlange, Lairdp, Avneralgom, Capricorn42, Rakesh india, Surajpandey10, Pontificalibus, Nfr-Maat, Nasnema, Poliverach, Gkorland, Ohspite, Ramnathkc, Wlouth, Tatatemc, Chadastrophic, Dbake, NFD9001, Emrekenci, Anna Frodesiak, Explorer09, BrianWren, Peterduffell, Macholl, Anamika.search, EricTheRed20, Michael.owen4, MarkCPhinn, NocturneNoir, Miym, J04n, GrouchoBot, Onmytoes4eva, Frosted14, Popnose, Protection-

TaggingBot, Rsiddharth, Omnipaedista, Andyg511, DarrenOP, RibotBOT, Mattg82, TonyHagale, Jbekuykendall, Jwt1801, Mathonius, Amaury, Yodaspirine, Vmops, WootFTW, Liyf, Mobilecloud, Figaronline, BrennaElise, Shadowjams, E0steven, Chaheel Riens, Jef4444, Person1988, A.amitkumar, Dan6hell66, RetiredWikipedian789, Mmanie, FrescoBot, Imtiyaz 81, Adlawlor, Yuchan0803, Zachnh, Manusnake, Blackguard SF, Cajetan da kid, Paj mccarthy, Ronen.hamias, Mark Renier, CloudBot, Sariman, Jakeburns99, Jesse.gibbs.elastra, W Nowicki, Estahl, Pottersson, Freddymay, Recognizance, Nakakapagpabagabag, MichealH, Gratridge, Ashakeri3596, Ummelgroup, Sebastiangarth, HJ Mitchell, Pete maloney, Scott A Herbert, Zhanghaisu, Berny68, Wireless Keyboard, HamburgerRadio, Yinchunxiang, Lmp90, Rickyphyllis, Acandus, Vasq0123, Winterst, Monkeyfunforidiots, Pinethicket, I dream of horses, Elockid, HRoestBot, Samuraiguy, Jonesey95, Eddiev11, AcuteSys, MJ94, PMstdnt, JLRedperson, Li Yue Depaul, Tinton5, Skyerise, Ggafraser, A8UDI, Dannymjohnson, Nimbus prof, RedBot, Janniscui, Manishekhawat, Bigfella1123, SpaceFlight89, Aneesh1981, Troy.frericks, Σ, Natishalom, Agemoi, Piandcompany, Noisalt, Cloudnitc, Jandalhandler, Devinek22, Undiscovered778, No1can, Ras67, Maasmiley, Abligh, Reconsider the static, AstralWiki, Juliashapiro, SW3 5DL, Niri.M, Msileinad, Hughesjp, Skovigo, Kjohnthomas, Jburns2009, JanaGanesan, ConcernedVancouverite, Trappist the monk, Declan Clam, Iminvinciblekris, SchreyP, Rajeshkamboj, Avermapub, KotetsuKat, Sanmurugesan, Markus tauber, Burrows01, Lotje, Kieransimkin, EDC5370, Dinamik-bot, Vrenator, Danielrs, LilyKitty, Richramos, Clarkcj12, Robscheele, SeoMac, Miracle Pen, Çalıştay, Ansumang, Ycagen, Aoidh, Eco30, Reaper Eternal, Crysb, Whitehouseseo, Info20072009, Jeffrd10, Pmell, Imnz730, Cemgurkok, Suffusion of Yellow, Taicl13, Tbhotch, Latha as, Colleenhaskett, Balvord, Hutch8700, MarshallWilensky, Nesjo, DARTH SIDIOUS 2, OmbudsTech, MidgleyC, Mean as custard, Sumdeus, RjwilmsiBot, Veerapatr.k, TjBot, DexDor, Alph Bot, Tdp610, Jon.ssss, Nasserjune, Amartya71, VernoWhitney, Darkyeffectt, Chriswriting, Vpolavia, Beleg Tâl, Ianmolynz, DuineSidhe, DSP-user, Learn2009, Aurifier, Jineshvaria, Pmorinreliablenh, Bamtelim, Adjectivore, Lipun4u, R39132, Nookncranny, J36miles, EmausBot, Editorfry, Understated1, Acather96, Pjposullivan, Deepalja, WikitanvirBot, Gozzilli, Rutger72, PaulQSalisbury, Logical Cowboy, Timtempleton, Eusbls, DragonflyReloaded, Macecanuck, Ajraddatz, Heracles31, Noloader, Jbadger88, Dewritech, Clutterpad, Ianlovesgolf, Gmm24, GoingBatty, RA0808, Snydeq, AoV2, Vanished user zq46pw21, Tisane, Sp33dyphil, Fzk chc, Luigi Baldassari, Solarra, Njcaus, Moswento, Sam Tomato, Wikipelli, 623des, K6ka, Tmguru, AsceticRose, Srastogis, Anirudh Emani, Zafar142003, Thecheesykid, Francis.w.usher, Hardduck, Vishwaraj.anand00, QuentinUK, Manasprakash79, Jtschoonhoven, BobGourley, Bongoramsey, Fæ, Josve05a, DeleoB, Ppachwadkar, Stevenro, Aaditya025, Trinidade, Wackywace, 9Engine, TunaFreeDolfin, Tom6, Kronostar, Wtsao, Aavindraa, A930913, GZ-Bot, H3llBot, Chintan4u, Eniagrom, Amanisdude, Kyerussell, Machilde, Mr little irish, Tolly4bolly, Jay-Sebastos, Thexman20, Coasterlover1994, Scott.somohano, L Kensington, Ready, Mayur, Aflagg99, Donner60, Skumarvimal, Zfish118, MillinKilli, Puffin, Ego White Tray, Orange Suede Sofa, Rangoon11, Bill william compton, MainFrame, JohnnyJohnny, StartupMonkey, ClamDip, Kenny Strawn, MaxedoutnjNJITWILL, Alex5678, Msfitzgibbonsaz, Shajulin, Nurnware, EmilyEgnyte, DASHBotAV, NatterJames, Jhodge88, JohnJamesWIlson, 28bot, JonRichfield, Frozen Wind, Petrb, RS-HKG, Dyepoy05, ClueBot NG, Sharktopus, Horoporo, Michaelmas1957, DellTechWebGuy, Jack Greenmaven, Slwri, VicoSystems1, Businesstecho, DrFurey, Dadomusic, Cloudcto, Amoebat, MelbourneStar, This lousy T-shirt, Qarakesek, CPieras, Satellizer, A520, Markqu, Bulldog73, Kkh03, Ethicalhackerin2010, Bped1985, Stickyboi, Candace Gillhoolley, Happyinmaine, Leifnsn, Hemapi, Lord Roem, Gxela, Nikoschance, The Master of Mayhem, Shawnconaway, Einasmadi, Qusayfadhel, Certitude1, Tylerskf, PJH4-NJITWILL, IDrivebackup, Lionheartf, O.Koslowski, Kevin Gorman, Dimos2k, ScottSteiner, Helotrydotorg, Alanmonteith, Digestor81, Widr, Scottonsocks, WikiPuppies, Gawali.jitesh, Andersoniooi, Rebuker, Chuahcsy, Carl presscott, Qoncept, Keynoteworld, Fearlessafraid, Johananl, Helpful Pixie Bot, OAnimosity, Leoinspace, Iste Praetor, HMSSolent, Tastic007, Titodutta, Calabe1992, DBigXray, Lavanyalava, Aditya.smn, Whitehatpeople, Elauminri, Angrywikiuser, BG19bot, RLSnow, SocialRadiusOly, MikeGeldens, Oluropo, Krenair, Cloudxtech, Sfiteditor, ValStepanova, Joerajeev, Robert.ohaver, Cornelius383, Dpsd 02011, Jimsimwiki, Markoo3, Rijinatwiki, Bhargavee, Softdevusa, Northamerica1000, Nadeemamin, Jayvd, San2011, Shaysom09, Lee.kinser, GhostModern, Om23, Panoramak, Avillaba, Hallows AG, Wiki13, Stevevictor134, MusikAnimal, Frze, Er.madnet, BDavis27, TylerFarell, Itzkishor, Mark Arsten, Compfreak7, Kirananils, EEldridge1, Leinsterboy, Dmcelroy1, Philopappos86, ThomasTrappler, StrategicBlue, Joydeep, Anne.naimoli, Jbucket, JMJeditor, Blvrao, Mychalmccabe, JDC321, Watal, Latticelattuce, Wretman, Frdfm, Phil.gagner, Elasticprovisioner, DPL bot, Andromani, Tramen12, Torturedgenius, Phazoni, Chapmagr, Jmillerfamily, TJK4114, Subbumv, Dinhthaihoang, Charvoworld, Mpcirba, Smileyranger, Klilidiplomus, Ssabihahmed, Achowat, Wannabemodel, ChambersDon, Fylbecatulous, BrianWo, Knodir, EricEnfermero, JoeBulsak, BattyBot, 21hattongardens, Eduardofeld, Kridjsss, Dlowenberger, 1337H4XX0R, Kunitakako, Elbarcino, ChannelFan, Haroldpolo, Cloudfest, Teammm, Xena77, Anhtrobote, Pratyya Ghosh, Tuwa, Mdann52, D rousslan, MPSTOR, Pea.hamilton, Crackerspeanut12, Mrt3366, Cloudreviewer, ChrisGualtieri, LarryEFast, Jackoboss, Valentina Ochoa, Beanilkumar, Mediran, EliyahuStern, Beer2beer, EuroCarGT, Prodirus, Fb2ts, SimonBramfitt, Xxyt65r, Mheikkurinen, Nithdaleman, Jags707, EagerToddler39, Davidogm, Padmaja cool, Zingophalitis, Zeeyanwiki, Mcsantacaterina, Shierro, Weternuni, Webclient101, Raushan Shahi, Sjames1, Mogism, Gotocloud, Derekvicente, Nozomimous, Anderson, Cerabot, Chishtigharana, Lone boatman, Fabrice Florin (WMF), PonmagalMalar, Naturelover007, TwoTwoHello, Thewebartists013, TechyOne, Aloak1, Vikas gupta70, Arnavrox, Frosty, SFK2, MartinMichlmayr, Sk8trboi199, Os connect, Jamesx12345, Shubhi choudhary, Joe1689, Sriharsh1234, Viralshah0704, Millycylan, Lorenrb, Kevin12xd, Choor monster, Drjoseph7, BurritoBazooka, OSRules, Waynej6, Avacam, Khan.sharique1994, Amitgupta2792, Zimzaman, Faizan, Chiefspartan, Epicgenius, Shivalikisam, FallingGravity, Ruinjames, Ramanrawal, Acaliguiran, Vanamonde93, S.sameermanas, JaredRClemence, I am One of Many, John-readyspace, Whitecanvas, Jlamus, Manishrai tester, FunkyMonk101, Carrot Lord, Jp4gs, Melonkelon, Mangai Vellingiri, Lena322, Craig developer, Danieljohnc, Alfy32, Solomon35, Rkocher, 5c0tt-noe, Thinkcd, Lsteinb, Tentinator, Marinac93, Aaronito, Evano1van, Kapils1255, Olynickjeff, Marcio10Luiz, Cookingwithrye, Jopgro, Fsandlinux, Backendgaming, 5andrew1, Nextlevelwb, Maura Driscoll, Flat Out, Halkemp, Couth, Dreamfigure, Murus, Saqibazmat, Jugaste, Babitaarora, Bloonstdfan360, Comp.arch, Kewi69, Metadox, Jbrucb, Podger.the, Henhuawang, Katepressed, Tbilisi2013, Asarada, Rzicari, AcidBlob, Rodie151, Fatdan786, Max1685, Mariatim, Ginsuloft, ArmitageAmy, Didi.hristova, IMMS, Insomniac14, Corey Rose, Acalycine, Jackmcbarn, Dudewhereismybike, PracticalScrum, TDBA, Bob Staggert, Rkanojia, Pcpded, Jora8488, Deeepak1300, Harshac89, Goofette, WikiJuggernaut, Gracecheung08, Cloud guru28, Lucy1982, CloudBurster, PierreCoyne, Sweetsadamali, LookToLuke, Justuj, Mareep, Wlwells67833, JaconaFrere, Theworm4321, JenniferAndy, Skr15081997, Rax sa, Ssgmu55, Nclemen2, 7Sidz, Sofia Lucifairy, Abhinavgupta007, Sfroberts, Musabhai2, Ajabak, Edwardsmith285, Nyashinski, MtthwAndrn, Shanhuyang, Nassaraf, Amenychtas, Uk1211, Deepika Sreerama, Monkbot, Cjbwin, Allanamiller, Sylvesta101, Lucyloo10, Kalpesh radadiya, Dansullivanpdx, Bingoarunprasath, BethNaught, Security.successfactors, Jblews, Mboxell, Mannanseo, MorePix, ThatWriterBloke, Ipsrsolutions, JoelAaronSeely, Biblioworm, Science.Warrior, Daniel.moldovan, Schwarrrtz, Gk9999, Northbridge Secure, Twoosh, Cloudwizard, Thandi moyo, Wrightandru, Ferozahmed0382, Mkchendil, Tonysmith2014, Manjaribalu, Ramvalleru, Daniela C DeMaria, Brandon Connor, Andy7809, Chicodoodoo, MONISHA GEORGE, MVMeena, Garywfchan, Sahit1109, Tweeks-va, Ss.jarvisboyle, Cubexsweatherly, Stephenzhang.cs, Srinivas.dgm, Dtechinspiration, Amagi82, Lalith269, Headinthecloud, Shantan 1995, Edavinmccoy, Zellabox, Abcdudtc, Enkakad, Nei Wg Khang, Selvarajrajkanna, FervourPriyanka, Naveenwilder, Alexsmith22, Sanjeev.rawat86, Yinongchen, Amit Seo Expert, Robbetto, Awm165, Nemesis2473 and Anonymous: 2475

- **Grid computing** *Source:* http://en.wikipedia.org/wiki/Grid%20computing?oldid=644497125 *Contributors:* The Anome, Tarquin, Jan Hidders, LA2, Christopher Mahan, Heron, Mrwojo, Edward, DopefishJustin, Kku, Rw2, Ronz, Setu, Srikrishnan, Rotem Dan, Kaihsu,

- **Computer cluster** *Source:* http://en.wikipedia.org/wiki/Computer%20cluster?oldid=640418649 *Contributors:*

- **Supercomputer** *Source:* http://en.wikipedia.org/wiki/Supercomputer?oldid=650571618 *Contributors:*

unen Baka, Robbot, Dale Arnett, Hankwang, Fredrik, Scott McNay, Donreed, Romanm, Modulatum, Lowellian, Geoff97, Texture, Asparagus, Tobias Bergemann, Davedx, Ancheta Wis, Kevin Saff, Alexwcovington, Giftlite, DavidCary, Akadruid, Pretzelpaws, Haeleth, Inkling, Ævar Arnfjörð Bjarmason, Herbee, Monedula, Wwoods, Tuqui, Rookkey, Rchandra, Sdeox, Matt Crypto, Bosniak, Ryanaxp, Chowbok, Jasper Chua, Antandrus, Beland, Margana, Robert Brockway, Piotrus, Balcer, Bk0, Sam Hocevar, Ojw, Muijz, Moxfyre, Yaos, N328KF, Imroy, RossPatterson, Discospinster, Rich Farmbrough, Florian Blaschke, Adam850, Mani1, Dyl, Stbalbach, Bender235, ESkog, ZeroOne, Violetriga, CanisRufus, RoyBoy, Vipul, Gershwinrb, Femto, Circeus, Harley peters, Smalljim, Giraffedata, Anr, Roy da Vinci, Anonymous Cow, Sukiari, Bijee, Hellis, ClementSeveillac, Alansohn, Liao, Etxrge, Duffman, Guy Harris, Xalfor, Johndelorean, Laug, Suruena, Evil Monkey, Humble Guy, Bsadowski1, Gunter, MIT Trekkie, Kleinheero, TheCoffee, Johntex, HenryLi, Forderud, Oleg Alexandrov, Nuno Tavares, Richard Arthur Norton (1958-), Nick Drake, Torqueing, Scm83x, Cooperised, Elvarg, Toussaint, TNLNYC, Paxsimius, Qwertyus, RxS, Squideshi, Rjwilmsi, MJSkia1, Koavf, Wikibofh, T0ny, Linuxbeak, Tangotango, Bruce1ee, Tawker, SMC, Nneonneo, Quietust, NeonMerlin, Bubba73, Matt Deres, Platyk, Harmil, Nivix, RexNL, KFP, Alphachimp, Chobot, Simesa, YurikBot, Wavelength, TexasAndroid, RobotE, Hede2000, Splash, Samuel Curtis, Stephenb, Manop, Shell Kinney, Gaius Cornelius, Eleassar, NawlinWiki, Wiki alf, Mipadi, Buster79, Długosz, Aufidius, Nutiketaiel, Slarson, Hogne, Jpbowen, Ndavies2, Voidxor, MySchizoBuddy, Amwebb, Roche-Kerr, BOT-Superzerocool, Gadget850, DeadEyeArrow, Bota47, Trcunning, Searchme, Georgewilliamherbert, Closedmouth, Th1rt3en, Peyna, Trainor, Dmuth, JLaTondre, Katieh5584, Rwwww, AndrewWTaylor, Oldhamlet, Yakudza, SmackBot, Mmernex, Erik the Appreciator, Henriok, Bd84, Pgk, Darkstar1st, Chych, Agentbla, Nil Einne, Tim@, Gilliam, Ppntori, Andy M. Wang, Anastasios, Cowman109, Anwar saadat, Andyzweb, Hitman012, Bluebot, MK8, Adam M. Gadomski, Thumperward, SchfiftyThree, Hibernian, CSWarren, Krallja, John Reaves, Rama's Arrow, Can't sleep, clown will eat me, Samrawlins, Jahiegel, Proofreader, Onorem, Gamester17, JonHarder, Rrburke, Metageek, Roaming, Fuhghettaboutit, Nakon, Nick125, Drphilharmonic, Er Komandante, Zahid Abdassabur, Kuru, Meteshjj, Soumya92, Gobonobo, Statsone, JH-man, Homfrog, JoshuaZ, Joffeloff, Antonielly, CredoFromStart, Niroht, Chrisch, JHunterJ, Beetstra, Mr Stephen, MainBody, Nils Blümer, EEPROM Eagle, Peyre, Delta759, Hgrobe, Balderdash707, NName591, Iridescent, Olegos, Doc Daneeka, Joseph Solis in Australia, Shoeofdeath, Newone, Tony Fox, Stoakron97, DarlingMarlin, Patrickwooldridge, J Milburn, JForget, CmdrObot, Page Up, JohnCD, Yarnalgo, NickW557, Shandris, Ravensfan5252, Doctorevil64, Johnlogic, Equendil, MC10, Chuck Marean, Cec, Ttiotsw, ST47, Jayen466, Myscrnnm, Jedonnelley, Codetiger, Cowpriest2, Kozuch, Editor at Large, Wexcan, Epbr123, Ryansca, Kubanczyk, Keraunos, N5iln, Mojo Hand, James086, Edal, Ekashp, Dgies, Zachary, Akata, Thadius856, AntiVandalBot, Gioto, Bkkeim2000, Seaphoto, LinaMishima, List of marijuana slang terms, Krtek2125, Qwerty Binary, [illegible], Erxnmedia, JAnDbot, Gatemansgc, MER-C, CosineKitty, Ericoides, Arch dude, Chanakyathegreat, IanOsgood, Owenozier, TAnthony, LittleOldMe, .anacondabot, SteveSims, Pedro, Bongwarrior, VoABot II, Dannyc77, JamesBWatson, PeterStJohn, Some fool, Artlondon, Sanket ar, Sink257, 28421u2232nfenfcenc, Dck7777, Vssun, DerHexer, Edward321, Christopher.booth, TheRanger, Patstuart, DGG, Gwern, DancingPenguin, Isamil, MartinBot, Jsbillings, Keith D, CommonsDelinker, Qrex123, Artaxiad, J.delanoy, Ali, Javawizard, Maurice Carbonaro, Thegreenj, Afskymonkey, Cdamama, Cpiral, Dontrustme, Igottalisp, FrummerThanThou, AlienZen, McSly, Ignatzmice, Samtheboy, Elcombe2000, SJP, SriMesh, Vaibhavahlawat1913, Kovo138, Adamd1008, Cometstyles, Quadibloc, Jamesontai, Vanished user 39948282, Treisijs, Headsrfun, LogicDictates, Jaqiefox, Davidweiner23, Idioma-bot, Funandtrvl, VolkovBot, Jeff G., Torswin, Philip Trueman, TXiKiBoT, Tovojolo, Sean D Martin, T-bonham, Melsaran, Corvus cornix, Martin451, Raryel, Gererd+, Maxim, RadiantRay, VanishedUserABC, Wikineer, Mary quite contrary, Michael Frind, Colorvision, K25125, JamesBondMI6, SieBot, Wowiamgood123, Sonicology, Calliopejen1, Tiddly Tom, ToePeu.bot, Alcachi, Dawn Bard, Caltas, RJaguar3, Mihaigalos, Maddiekate, Mwaisberg, Toddst1, Quest for Truth, Flyer22, JCLately, Lord British, Shaw SANAR, Xe7al, AnonGuy, Lightmouse, KathrynLybarger, Coffeespoon, Arthur a stevens, CharlesGillingham, Chillum, Fishnet37222, ClueBot, Danbert8, IanBrock, Editor4567, Rilak, Arakunem, Jbaxt7, Cp111, Lbeben, DocumentN, The 888th Avatar, SCOnline, Namazu-tron, Shainer, Jusdafax, Monobi, GoRight, 12 Noon, Tomtzigt, [illegible], Shethpratik, Rajesh.krissh, ViveCulture, Dekisugi, OrbitalAnalyst, La Pianista, Marcoacostareyes, SoxBot III, 5950FX, Harman malhotra, Ali azad bakhsh, Alchemist Jack, Vsm01, Seuakei, Finchsnows, Zrs 12, Zodon, MichaelsProgramming, SheckyLuvr101, Dsimic, Mojska, TreyGeek, 7im, Addbot, Ramu50, Some jerk on the Internet, Mchu amd, Aomsyz, Leszek Jańczuk, Fluffernutter, LokiiT, Download, Truaxd, Mjr162006, Glane23, Chzz, Torla42, ChenzwBot, Jasper Deng, Fireaxe888, Wikicojamc, Fleddy, Cadae, Tide rolls, David0811, Jarble, ZotovBST, Peturingi, Luckas-bot, Yobot, OrgasGirl, Cbtarunjai87, Fraggle81, Nirvana888, Shinkansen Fan, Jean.julius, Tempodivalse, AnomieBOT, Indulis.b, Catin20, Jim1138, Jawz44, Shieldforyoureyes, AdjustShift, Materialscientist, Citation bot, Air55, Obersachsebot, Xqbot, Capricorn42, Drilnoth, KarlKarlson, Mhdrateln, Jmundo, NFD9001, Gap9551, Champlax, Miym, Abce2, RibotBOT, Mathonius, New guestentry, Der Falke, VB.NETLover, Ahmedabadprince, Eugene-elgato, SchnitzelMannGreek, CES1596, Lionelt, FrescoBot, Nvgranny, Vinceouca, Gino chariguan, JEIhrig, DrilBot, Pinethicket, I dream of horses, Jj05y, 10metreh, Loyalist Cannons, Calmer Waters, Yahia.barie, Skyerise, Jschnur, RedBot, Ongar the World-Weary, MondalorBot, RandomStringOfCharacters, Xeworlebi, SkyMachine, FoxBot, TobeBot, DixonDBot, Jonkerz, Lotje, Seanoneal, Hefiz, Diannaa, WikiTome, ThinkEnemies, Sirkablaam, TareqMahbub, DARTH SIDIOUS 2, NKOzi, Jfmantis, RjwilmsiBot, Mrsnuggless, Apeman2001, Leadmelord, Salvio giuliano, Metaferon, EmausBot, Bonanza123d, Autarchprinceps, Gfoley4, JteB, Jmencisom, Wikipelli, K6ka, Maycrow, Thecheesykid, Vincentwilliamse, ZéroBot, Fæ, Mar4d, AOC25, Gz33, Sfraza, Noodleki, MonoAV, Donner60, ChuispastonBot, VictorianMutant, Sonicyouth86, Petrb, Mikhail Ryazanov, Cswierkowski, ClueBot NG, Supercomputtergeek, Thatdumisdum, Michaelmas1957, Gareth Griffith-Jones, Jack Greenmaven, GioGziro95, CocuBot, MelbourneStar, This lousy T-shirt, Satellizer, Joefromrandb, Jmarcelo95, Jessvj, Widr, Kant66, جدامجدس, Helpful Pixie Bot, Somatrix, HMSSolent, BG19bot, Donkeyo4, FuFoFuEd, Imgaril, Prosa100, Wiki13, Vivek prakash81, Lgmora, Jeancey, Maxrangeley, Jasonas77, Elsie3456, Nuclearsavage, Pratyya Ghosh, LarryEFast, Tow, BrightStarSky, Webclient101, Mogism, Akshayranjan1993, PeerBaba, Skydoc28, Frosty, Graphium, CCC2012, Faizan, Forgot to put name, Ruby Murray, Chris troutman, Comp.arch, Ugog Nizdast, Nakitu, Lizia7, Skr15081997, Frankhu2016, Crossswords, Leegrc, TonyM101, Akjprao, SantiLak, Ilikepeak, Mama meta modal, Biblioworm, Youjustfailed, Ninjacyclops, TerryAlex, Sophie.grothendieck, Hardikjain2002, ChamithN, AYOBLAB, Tuneix, Python.kochav, Silversparkcontributions, Newwikieditor678 and Anonymous: 894

- **Multi-core processor** *Source:* http://en.wikipedia.org/wiki/Multi-core%20processor?oldid=650161229 *Contributors:* Edward, Mahjongg, Nixdorf, Ixfd64, 7265, CesarB, Ronz, Julesd, Charles Matthews, Dragons flight, Furrykef, Bevo, Mazin07, Jakohn, Donreed, Altenmann, Nurg, Auric, Bkell, Ancheta Wis, Centrx, Giftlite, DavidCary, Gracefool, Solipsist, Falcon Kirtaran, Kiteinthewind, Ludootje, Cynical, Qiq, Ukexpat, GreenReaper, Alkivar, Real NC, MattKingston, Monkeyman, Reinthal, Archer3, Rich Farmbrough, Florian Blaschke, Sapox, SECProto, Berkut, Dyl, Bender235, Narcisse, RoyBoy, Dennis Brown, Neilrieck, WhiteTimberwolf, Bobo192, Fir0002, SnowRaptor, Matt Britt, Hectoruk, Gary, Liao, Polarscribe, Guy Harris, Hoary, Evil Prince, Lerdsuwa, Bsadowski1, Gene Nygaard, Marasmusine, Simetrical, Woohookitty, Henrik, Mindmatrix, Aaron McDaid, Splintax, Pol098, Vossanova, Qwertyus, JIP, Ketiltrout, SMC, Smithfarm, CQJ, Bubba73, Yamamoto Ichiro, Skizatch, Ian Pitchford, Master Thief Garrett, Crazycomputers, Superchad, Dbader, DaGizza, SirGrant, Hairy Dude, TheDoober, Epolk, Stephenb, Rsrikanth05, NawlinWiki, VetteDude, Thiseye, Rbarreira, Anetode, DGJM, Falcon9x5, Addps4cat, Closedmouth, Fram, Andyluciano, JLaTondre, Carlosguitar, Mark hermeling, SmackBot, Mmernex, Stux, Henriok, JPH-FM, Jagged 85, Powo, Pinpoint23, Thumperward, Swanner, Hibernian, JagSeal, E946, Shalom Yechiel, Frap, KaiserbBot, AcidPenguin9873, JonHarder, Kcordina, Aldaron, Fuhghettaboutit, Letowskie, DWM, Natamas, Kellyprice, Fitzhugh, Sonic Hog, A5b, Homo

sapiens, Lambiam, Kyle wood, JzG, Pgk1, Littleman TAMU, Ulner, WhartoX, Disavian, Danorux, Soumyasch, Joffeloff, Gorgalore, Guy2007, Fernando S. Aldado, 16@r, JHunterJ, NJA, Peyre, Vincecate, Hu12, Quaeler, Iridescent, Pvsuresh, Tawkerbot2, CmdrObot, Plasticboob, CBM, Nczempin, Jesse Viviano, Michaelbarreto, Shandris, Evilgohan2, Neelix, Babylonfive, ScorpSt, Cydebot, Myscrnnm, Steinj, Drtechmaster, Kozuch, Thijs!bot, Hervegirod, Mwastrod, Bahnpirat, Squater, Dawnseeker2000, Sherbrooke, AlefZet, AntiVandalBot, Widefox, Seaphoto, Shalewagner, DarthShrine, Chaitanya.lala, Leuko, Od1n, Gamer2325, Arch dude, IanOsgood, Stylemaster, Aviadbd, Geniac, Coffee2theorems, Ramurf, Vintei, JamesBWatson, CattleGirl, CountingPine, Midgrid, EagleFan, David Eppstein, DerHexer, Gimpy530, Gwern, Gjd001, Oren0, Bdsatish, Red66, Ehoogerhuis, Sigmajove, Felipe1982, J.delanoy, Jspiegler, GuitarFreak, NerdyNSK, Techedgeezine, Acalamari, Barts1a, Thucydides411, EMG Blue, Chrisforster, Mikael Häggström, Hubbabridge, LA Songs, SlightlyMad, Haoao, Remi0o, 28bytes, Monkeyegg, Imperator3733, Smkoehl, Gwib, Klower, Taurius, HuskyHuskie, ITMADOG, Haseo9999, Cmbay, Nono1234, ParallelWolverine, Mike4ty4, JasonTWL, Vjardin, Winchelsea, Rockstone35, Caltas, Jerryobject, Flyer22, FSHL, Rogergummer, Lord British, Rupert baines, Ddxc, Ttrevers, Noelhurley, Radical.bison, WikipedianMarlith, ClueBot, Binksternet, GorillaWarfare, Starkiller88, Rilak, Czarkoff, Taroaldo, Wikicat, LizardJr8, Cirt, DragonBot, Drewster1829, Goodone121, Karlhendrikse, Coralmizu, Alejandrocaro35, Time2zone, Jeffmeisel, Msrill, Versus22, Un Piton, DumZiBoT, Чръный человек, Parallelized, Zodon, Airplaneman, Dsimic, Thebestofall007, Addbot, Proofreader77, Hcucu, Scientus, CanadianLinuxUser, MrOllie, LaaknorBot, Markuswiki, Jasper Deng, IOLJeff, Imirman, Tide rolls, Jarble, Ettrig, Legobot, Yobot, SabbaZ, TaBOT-zerem, Jack Boyce, Goldenthree, Danperryy, SwisterTwister, Mdegive, AnomieBOT, Enisbayramoglu, Decora, Jim1138, Piano non troppo, DaveRunner, GFauvel, Flewis, Materialscientist, RobertEves92, Threadman, Gnumer, Joshxyz, MacintoshWriter, LilHelpa, TheAMmollusc, Miym, Abce2, Bikeman333, Barfolomio, Adavis444, Gordonrox24, Elemesh, Gastonhillar, Prari, Hemant wikikosh, FrescoBot, Picklecolor2, StaticVision, RaulMetumtam, MBbjv, Winterst, Elockid, UkillaJJ, Skyerise, Meaghan, FoxBot, Sulomania, Ellwd, Glenn Maddox, Gal872875, Sreven.Nevets, NagabhushanReddy, Gg7777, Jesse V., DARTH SIDIOUS 2, Truthordaretoblockme, Beyond My Ken, WildBot, Virtimo, Helwr, EmausBot, Az29, Keithathaide, Super48paul, P3+J3^u!, Tommy2010, Serketan, Erpert, Alpha Quadrant (alt), NGPriest, Bmmxc damo, Steedhorse, L Kensington, Donner60, Jsanthara, DASHBotAV, Rmashhadi, Cswierkowski, ClueBot NG, Jeff Song, Gilderien, Chharper1, Braincricket, Widr, MerlIwBot, Nodulation, OpenSystemsPublishing, Minadasa, Cdog44, Hz.tiang, Charlie6WIND, WinampLlama, Op47, Harizotoh9, Nolansdad95120, Glacialfox, Simonriley, DigitalMediaSage, Sha-256, Michael Anon, Snippy the heavily-templated snail, NimbusNiner, DavidLeighEllis, Koza1983, Christian CHABRERIE, Geoyo, Aniru919, Lagoset, Sofia Koutsouveli, RoninDusette, Kyle1009, ComsciStudent, DorothyGAlvarez and Anonymous: 643

- **Graphics processing unit** *Source:* http://en.wikipedia.org/wiki/Graphics%20processing%20unit?oldid=650152153 *Contributors:* Taw, Wayne Hardman, Heron, Edward, DopefishJustin, Mahjongg, Nixdorf, Karada, Egil, Andres, Harvester, Lee Cremeans, Furrykef, Tempshill, Wernher, Thue, Topbanana, Stormie, Optim, Robbot, Chealer, Vespristiano, Playwrite, Academic Challenger, Tobias Bergemann, Alan Liefting, Alf Boggis, Paul Pogonyshev, Everyking, Alison, Lurker, DJSupreme23, Gracefool, Rchandra, AlistairMcMillan, Egomaniac, Khalid hassani, Gadfium, Utcursch, Pgan002, Aughtandzero, Quadell, Lockeownzj00, Beland, MFNickster, Simoneau, Trilobite, Imroy, Pixel8, AlexKepler, Berkut, Alistair1978, Pavel Vozenilek, Gronky, Indrian, Evice, Billlion, TOR, CanisRufus, RoyBoy, Drhex, Polluks, Matt Britt, Richi, Kjkolb, Markpapadakis, Kaf, Varuna, Murphykieran, Mc6809e, Hohum, Angelic Wraith, Velella, Suruena, Sciurinæ, Bjorke, Freyr, Marasmusine, Kelly Martin, Woohookitty, Jannex, Ae-a, Macronyx, SCEhardt, Isnow, M412k, Toussaint, Kbdank71, Josh Parris, Tbird20d, Sdornan, Sango123, StuartBrady, FlaBot, Mirror Vax, Arnero, Viznut, Chobot, ShadowHntr, YurikBot, Jtbandes, Locke411, Yyy, ALoopingIcon, Virek, RicReis, Qviri, Panscient, Zephalis, Mike92591, MaxDZ8, Wknight94, Delirium of disorder, Arthur Rubin, D'Agosta, E Wing, Red Jay, David Biddulph, Mikkow, Nekura, Veinor, FearTec, SmackBot, Colinstu, AFBorchert, Bigbluefish, Unyoyega, Jagged 85, Renku, KVDP, Jrockley, Eskimbot, Scott Paeth, Jpvinall, Gilliam, Bluebot, TimBentley, GoldDragon, QTCaptain, Thumperward, Jerome Charles Potts, Octahedron80, Anabus, Tsca.bot, Can't sleep, clown will eat me, Harumphy, Frap, JonHarder, Ruw1090, Easwarno1, Theonlyedge, Cybercobra, Melter, Nakon, Trieste, HarisM, Nitro912gr, Swaaye, Salamurai, HeroTsai, Soumya92, Disavian, Wibbble, Joffeloff, Codepro, Aleenf1, Vuurmeester, Phranq, Cxk271, Sjf, Hu12, Stargaming, Agelu, ScottHolden, Stoakron97, Aeons, Tawkerbot2, Jafet, Braddodson, SkyWalker, Xcentaur, Zarex, Mattdj, Nczempin, Jsmaye, Jesse Viviano, Shandris, Lazulilasher, Sahrin, Pi Guy 31415, Phatom87, Danrok, JJC1138, Gogo Dodo, Scissorhands1203, Soetermans, Mr. XYZ, Tawkerbot4, Bitsmart, Thijs!bot, Mentifisto, Eberhart, AntiVandalBot, Konman72, Gioto, SEG88, Flex Flint, Johan.Seland, Skarkkai, Serpent's Choice, JAnDbot, MER-C, Jdevesa, Arch dude, Kremerica, RubyQ, Vidsi, AndriusG, RBBrittain, Gbrose85, Michaelothomas, Nikevich, I JethroBT, Marmoulak, David Eppstein, Crazyideas21, Frampis, El Krem, UnfriendlyFire, Trusader, R'n'B, J.delanoy, Pharaoh of the Wizards, ChrisfromHouston, Jesant13, Smite-Meister, Gzkn, Xbspiro, M-le-mot-dit, Urzadek, Jo7hs2, EconomistBR, Sugarbat, Spiesr, Canadianbob, Martial75, Lights, VolkovBot, MrRK, TXiKiBoT, Like.liberation, Tr-the-maniac, Tandral, Cody-7, Broadbot, Haseo9999, Squalk25, AlleborgoBot, Glitchrf, SieBot, 4wajzkd02, Yulu, Garde, Djayjp, Flyer22, Nopetro, Oxymoron83, Lightmouse, Earthere, Twsl, Pinkadelica, Gillwill, WikipedianMarlith, Accessory, ClueBot, The Thing That Should Not Be, Placi1982, Rilak, Nnemo, Dpmuk, Jappalang, Hexmaster, Niceguyedc, Alexbot, Socrates2008, Technobadger, Arjayay, Jotterbot, Ark25, Muro Bot, Vapourmile, GlasGhost, Andy16666, Socks 01, Tigeron, 5900FX, GeoffMacartney, DumZiBoT, Rreagan007, Salam32, Frood, JeGX, Noctibus, Eleven even, Zodon, Veritysense, NonNobisSolum, Dsimic, Osarius, Addbot, Willking1979, Ronhjones, MrOllie, Download, LaaknorBot, Aunva6, Peti610botH, Fiftyquid, Jarble, Xowets, Ben Ben, Legobot, Publicly Visible, Luckas-bot, Yobot, Ptbotgourou, Becky Sayles, GateKeeper, Sg227, 4thotaku, AnomieBOT, Masterofwiki666, Galoubet, Materialscientist, Clark89, LilHelpa, JanEnEm, PavelSolin, Xqbot, Holden15, Erud, Victorbabkov, CoolingGibbon, P99am, Braxtonw1, J04n, Winstonliang, =Josh.Harris, Robert SkyBot, FrescoBot, IvarTJ, Umawera, Math1337, Jusses2, Vincentfpgarcia, RedBot, Akkida, Rzęsor, Hitachi-Train, Yogi m, Ale And Quail, Ravenperch, Jesse V., John Buchan, DARTH SIDIOUS 2, Onel5969, Dewritech, Dcirovic, Serketan, Cogiati, Vitkovskiy Roman, Handheldpenguin, Veikk0.ma, Romdanen, Tomy9510, Topeil, Evan-Amos, Des3dhj, ClueBot NG, Matthiaspaul, Dholcombe, Widr, Tijok, MarcusBritish, Helpful Pixie Bot, Largecrashman, Wbm1058, KLBot2, Aayush.nitb, Kangaroopower, Sqzx, MusikAnimal, Joydeep, Diculous, Alanau8605, Isenherz, Tagremover, Comatmebro, Dymatic, Stocbuster, Codename Lisa, Webclient101, Makecat-bot, Ckoerner, Nonnompow, Andrei.gheorghe, Frosty, Calinou1, OSXiOSMacFan, EdwardJK, Jmankovecky, Reatlas, Mahbubur-r-aaman, Hallowin, Eyesnore, Nigma2k, Dannyniu, CrystalCanine, Comp.arch, Papagao, Sibekoe, Jdog147123, ScotXW, UltraFireFX, Kral Petr, Mansoor-siamak, ChamithN, Newwikieditor678 and Anonymous: 460
- **OpenMP** *Source:* http://en.wikipedia.org/wiki/OpenMP?oldid=649201759 *Contributors:* The Anome, Llywrch, Minesweeper, Julesd, Selket, Secretlondon, Chealer, Hadal, BenFrantzDale, Rheun, Chowbok, Tietew, Jin, Corti, Mike Schwartz, Minghong, Jonsafari, Liao, Schapel, Suruena, Tedp, Foreignkid, Forderud, Siafu, Firsfron, Rchrd, Paxsimius, Qwertyus, Rjwilmsi, FlaBot, Dave1g, David H Braun (1964), Michael Suess, Sbrools, Visor, RussBot, Samsarazeal, Bisqwit, CarlHewitt, Jmore, SmackBot, AnOddName, Mcld, Bluebot, Deli nk, Frap, Nixeagle, BWDuncan, A5b, Soumyasch, Michael miceli, Woon Tien Jing, Mojoh81, Phsilva, Paul Foxworthy, Raysonho, Wws, Ezrakilty, MaxEnt, Michaelbarnes, Pipatron, Un brice, Dgies, Stannered, Yellowdesk, SeRo, JAnDbot, Bakken, Cic, User A1, Gwern, Aldinuc, JeromeJerome, Salahuddin66, Abasher, Idioma-bot, Markusaachen, Lear's Fool, RedAndr, Khazadum, Ohiostandard, SieBot, Scarian, Jerryobject, EnOreg, Denisarona, Dex1337, Wpoely86, Alexbot, DumZiBoT, Dsimic, Sameer0s, Deineka, Addbot, Kne1p, Ridgeview, Lebenworld, Enerjazzer, LaaknorBot, Wikomidia, Luckas-bot, Yobot, AnomieBOT, Amritkar, Nicolaas Vroom, Citation bot, Eumolpo,

LilHelpa, TheAMmollusc, SamuelThibault, Joehms22, Palatis, Daniel Strobusch, Nameandnumber, FrescoBot, Openmpexpert, Winterst, Timonczesq, JnRouvignac, Jfmantis, Ruudmp, Streapadair, Anubhav16, HalcyonDays, ZéroBot, Fabrictramp(public), AManWithNoPlan, Ipsign, Shajulin, Mikhail Ryazanov, Filiprino, Timflutre, Farnwang, SchlitzaugeCC, Skappes, OhioGuy814, Eranisme, Aleks-ger, ScotXW and Anonymous: 168

- **Message Passing Interface** *Source:* http://en.wikipedia.org/wiki/Message%20Passing%20Interface?oldid=644892073 *Contributors:* AxelBoldt, The Anome, Edward, Nealmcb, Michael Hardy, Modster, Egil, Emperorbma, Grendelkhan, Jnc, Raul654, Nnh, GPHemsley, Phil Boswell, Unknown, EpiVictor, Mirv, Rege, Superm401, Alerante, Thv, Uday, BenFrantzDale, Ketil, Jacob grace, Erik Garrison, Hellisp, Jin, AlexChurchill, AliveFreeHappy, CALR, Rich Farmbrough, Gronky, Vicarage, Liao, Oleszkie, Sligocki, Cdc, Thaddeusw, Stillnotelf, EmmetCaulfield, Suruena, Drbreznjev, Blaxthos, Forderud, Nuno Tavares, DavidBiesack, Bluemoose, Qwertyus, Ketiltrout, Rjwilmsi, Earin, Drrngrvy, FlaBot, Dave1g, Windharp, Michael Suess, YurikBot, Bovineone, CarlHewitt, Romanc19s, SvenDowideit, Flooey, BrianDominy, Bsod2, Boggie, JJL, SmackBot, Emeraldemon, Davepape, El Cubano, DHN-bot, Bsilverthorn, Frap, Adamantios, Sspecter, Cybercobra, Warren, Mwtoews, Sigma 7, ArglebargleIV, Fprincipe, Stardust85, Phuzion, Wizard191, Iridescent, Paul Foxworthy, Raysonho, Phatom87, Hebrides, Omicronpersei8, Rodrigo.toro, Un brice, Dgies, Byornski, Vetter, Danger, Lfstevens, Jazzydee, Magioladitis, JamesBWatson, JenniferForUnity, Juedsivi, A3nm, Gwern, MartinBot, R'n'B, Katalaveno, BagpipingScotsman, M-le-mot-dit, Tonyskjellum, Aquaeolian, Hulten, Idioma-bot, LokiClock, AlnoktaBOT, MusicScience, Qxz, Lordofcode, Lucadjtoni, Gloomy Coder, Khazadum, Jmath666, Winterschlaefer, Synthebot, VanishedUserABC, Glennklockwood, Kaell, Boy1jhn, Jerryobject, Sliwers, Syed Zafar Gilani, OKBot, Uniomni, ClueBot, SummerWithMorons, Avenged Eightfold, PipepBot, Foxj, Katmairock, Niceguyedc, Sheepe2004, Locus99, Leonard^Bloom, Hrafnkell.palsson, 2, DumZiBoT, BarretB, Kula85, Dsimic, Addbot, Ghettoblaster, Db1618, Lebenworld, MrOllie, Jarble, Luckas-bot, Yobot, Les boys, DanKidger, Tempodivalse, Windwisp, Jim1138, RobertEves92, Xqbot, TheAMmollusc, BulldogBeing, Binary Runner, Nonugoel, DenisKrivosheev, W Nowicki, Andrewhayes, Modamoda, Mreftel, Jesse V., Marie Poise, RjwilmsiBot, John of Reading, Immunize, Keithathaide, Japs 88, GoingBatty, ZéroBot, Flies 1, Codingking, Erget2005, Ipsign, Cswierkowski, ClueBot NG, Satellizer, Tyrantbrian, Hklimach, Webelity, Blelbach, Helpful Pixie Bot, Wbm1058, BG19bot, Griggy1, Wenzeslaus, Abs0ft2781, Compfreak7, Skappes, Sofia Koutsouveli, BradfordBaze, Joelmoniz, Fwyzard and Anonymous: 174
- **CUDA** *Source:* http://en.wikipedia.org/wiki/CUDA?oldid=649366577 *Contributors:* AxelBoldt, Boud, Michael Hardy, Ixfd64, Stevenj, Jeffq, Connelly, Jason Quinn, Gracefool, Vadmium, Chowbok, Simoneau, Saariko, Imroy, Qutezuce, Bender235, DrYak, Cwolfsheep, Mathieu, AshtonBenson, LeGreg, Schapel, Rebroad, ReyBrujo, Kenyon, Oleg Alexandrov, Mahanga, Asav, Tabletop, Eyreland, Qwertyus, Kbdank71, Rjwilmsi, Strait, Brighterorange, FlaBot, Rbonvall, Skierpage, Nehalem, Dadu, Noclador, Jnareb, Iamfscked, Gaius Cornelius, Bovineone, DavidConrad, Arichnad, MX44, TDogg310, ThomasBradley, Falcon9x5, Rwalker, Sarathc, Clith, Kendroberts, Cedar101, JLaTondre, Btarunr, Lomacar, Itub, SmackBot, Aths, Elonka, Reedy, Henriok, Mcld, Oscarthecat, Jnelson09, Frap, Gamester17, Rrburke, VMS Mosaic, Qmwne235, Ripe, VincentH, Arstchnca, StanfordProgrammer, Keredson, Lvella, FleetCommand, Raysonho, Jesse Viviano, NaBUru38, Shandris, Cydebot, Rifleman 82, Dancter, Jamitzky, Alaibot, Thijs!bot, Frozenport, Kamal006, Davidhorman, Openlander, Gioto, Widefox, Nidomedia, Lordmetroid, Markthemac, Fellix, Wootery, Penubag, Paranoidmage, Nyq, JamesBWatson, Hans Lundmark, Cic, Nk126, Curdeius, User A1, Nicholas wilt, JeromeJerome, Nikpapag, Yegg13, Jack007, Algotr, Asjogren, Flatterworld, Potatoswatter, Adamd1008, VolkovBot, Dan.tsafrir, MenasimBot, TXiKiBoT, Rei-bot, Rican7, Ilia Kr., Draceane, Synthebot, AMAMH, Julekmen, SieBot, Jerryobject, Quest for Truth, Paul.adams.jr, EnOreg, AlanUS, FxJ, DEEJAY JPM, ClueBot, Fyyer, Vergil 577, Razimantv, Tosaka1, Auntof6, Houyunqing, Netvope, Excirial, Socrates2008, Ykhwong, Miathan6, Ark25, GlasGhost, Aprock, Andy16666, Tigeron, Perchy22, DumZiBoT, InternetMeme, XLinkBot, SilvonenBot, RealityDysfunction, StewieK, Mschatz, Dsimic, Pozdneev, Addbot, Mortense, DOI bot, Svetlin, Lebenworld, Chris TC01, MrOllie, AndersBot, SpBot, AgadaUrbanit, Apple, Jimsve, Luckas-bot, Yobot, Agni451, Legobot II, Ahumber, Azylber, Torydude, Nyat, AnomieBOT, Ciphers, Hairhorn, Rubinbot, 1exec1, Henry Merriam, Amritkar, Fahadsadah, Materialscientist, Slsh, Anttonij, Churchill17, Jgottula, Majorcabuff, PavelSolin, Xqbot, Erud, Drpepperwithice, Animist, Nexus26, Weichaoliu, P99am, Control.valve, Tugaworld, Foobarhoge, Sotirisioannidis, FrescoBot, Opencl, Hyju, Komissarov Andrey, Citation bot 1, Royalstream, The GP-you Group, Winterst, Qyqgpower, RedBot, MastiBot, Anonauthor, GoneIn60, Diblidabliduu, Gpucomputingguru, Sokka54, VneFlyer, Joelfun96, Aoidh, Gaurav.p.chaturvedi, Jesse V., Jfmantis, Brkt, EmausBot, Tuankiet65, Dewritech, DanielWaterworth, Basheersubei, KermiDT, Ὁ οἶστρος, AManWithNoPlan, Salmanulhaq, Higgs Teilchen, Babək Akifoğlu, Atcold, Topeil, Psilambda, Minoru-kun, Research 2010, N8tingale, Cswierkowski, ClueBot NG, Misancer, Tayboonl, Ranga.prasa, BG19bot, Daarien, Uwsbel, Dumbbell1023, Gargnano, AlexReim, AdventurousSquirrel, Knecknec, Aerisch, Abledsoe78, Celtechm, Rezonansowy, Ksirrah, OsCeZrCd, Ginsuloft, Oranjelo100, ScotXW, Roynalnaruto, Kral Petr, Monkbot, Nzoomed, Realnot, Maxgeier, Vollmer1995, John W Herrick and Anonymous: 414
- **Peer-to-peer** *Source:* http://en.wikipedia.org/wiki/Peer-to-peer?oldid=648803348 *Contributors:* Damian Yerrick, AxelBoldt, Kpjas, Wesley, The Anome, RoseParks, Rjstott, Andre Engels, Greg Lindahl, Youssefsan, Aldie, M, SimonP, Ben-Zin, Ellmist, Heron, Dk, Branko, Olivier, Chuq, Jim McKeeth, Edward, Ubiquity, K.lee, Michael Hardy, Kwertii, Lexor, Lousyd, Shellreef, Kku, Liftarn, Gabbe, Collabi, Delirium, Eric119, Minesweeper, CesarB, Mkweise, Ahoerstemeier, Copsewood, Haakon, Mac, Ronz, TUF-KAT, Yaronf, Kingturtle, Ping, LittleDan, Julesd, Pratyeka, Glenn, Sir Paul, Rossami, Rl, Jonik, Bramp, Conti, Schneelocke, Mydogategodshat, Frieda, Timwi, MatrixFrog, Viajero, Wik, IceKarma, Rvalles, Maximus Rex, Sweety Rose, Furrykef, Itai, Bhuston, Meembo, SEWilco, Omegatron, Ed g2s, Bloodshedder, Dysfunktion, MadEwokHerd, Johnleemk, Jamesday, Owen, Chuunen Baka, Robbot, Paranoid, MrJones, Sander123, Korath, Tomchiukc, ZimZalaBim, Tim Ivorson, Postdlf, Texture, Yacht, TittoAssini, Qwm, Mushroom, Anthony, Cyrius, Moehre, Jrash, RyanKoppelman, Rossgk, Connelly, Giftlite, DocWatson42, Fennec, DavidCary, Laudaka, ShaunMacPherson, Mintleaf, Wolfkeeper, Netoholic, Lupin, Bkonrad, Niteowlneils, Endlessnameless, FrYGuY, Gracefool, AlistairMcMillan, Softssa, VampWillow, Benad, Jrdioko, Neilc, PeterC, Fys, Toytoy, Knutux, Lockeownzj00, Thomas Veil, ArneBab, Lord dut, Hgfernan, Secfan, Maximaximax, Vbs, Wiml, Korou, Ihsuss, Jareha, Lee1026, Cynical, Joyous!, Kevyn, DMG413, Ivo, The stuart, Shiftchange, Mormegil, Tom X. Tobin, DanielCD, Lifefeed, Discospinster, 4pq1injbok, Sharepro, Solitude, Rich Farmbrough, Rhobite, Iainscott, Alexkon, H0riz0n, Jon Backenstose, Inkypaws, Jsnow, Morten Blaabjerg, Deelkar, S.K., Loren36, Mjohnson, CanisRufus, Gen0cide, Koenige, Tverbeek, PhilHibbs, Diomidis Spinellis, Sietse Snel, Just zis Guy, you know?, Eltomzo, Grick, LBarsov, Velociped, BrokenSegue, Johnteslade, Cwolfsheep, SpeedyGonsales, VBGFscJUn3, Minghong, Idleguy, Wrs1864, Haham hanuka, Merope, Conny, Ifny, Liao, Mo0, Falsifian, CyberSkull, Gwendal (usurped), Andrewpmk, Cctoide, Sl, Apoc2400, Antoniad, Gaurav1146, Elchupachipmunk, Snowolf, Eekoo, Melaen, Gbeeker, Totof, Raraoul, ReyBrujo, Stephan Leeds, Evil Monkey, Tony Sidaway, Computerjoe, Versageek, Gene Nygaard, Ringbang, Netkinetic, MiguelTremblay, Ceyockey, Adrian.benko, AlexMyltsev, Mahanga, Kelly Martin, Mindmatrix, Vorash, The Belgain, Jersyko, Morton.lin, Deeahbz, Splintax, Abab99, Ilario, Ruud Koot, The Wordsmith, MONGO, Mangojuice, Wtfunkymonkey, Rchamberlain, CharlesC, Waldir, Sendai2ci, Wayward, Toussaint, Karam.Anthony.K, Palica, Gerbrant, Aarghdvaark, Zephyrxero, David Levy, Kbdank71, Phoenix-forgotten, Canderson7, CortlandKlein, Sjakkalle, Kumarbhatia, Rjwilmsi, Quale, Strait, PinchasC, Tawker, Forage, Edggar, Kry, Peter Tribe, LjL, Bhadani, -lulu-, FlaBot, Authalic, Ground Zero, RexNL, Ewlyahoocom, Mike Van Emmerik, Valermos, RobyWayne, Bmicomp, Chobot, Garas, Bgwhite, Manu3d, Dadu, Cuahl, YurikBot, Wavelength, Borgx, Pip2andahalf, RussBot, Wellreadone, Akamad, Gaius Cornelius, CambridgeBayWeather, Bovineone, Salsb, Richard Allen, Msoos, Johann Wolfgang, Nick, Retired username, Mikeblas, RL0919, Amwebb,

Matthewleslie, Nethgirb, Mavol, Wangi, DeadEyeArrow, Atbk, Bota47, Charleswiles, Xpclient, Nlu, Bikeborg, Boivie, FF2010, Zzuuzz, 2bar, Lt-wiki-bot, Ninly, Bayerischermann, Icedog, Closedmouth, Abune, GraemeL, Adammw, Fram, Scoutersig, Rearden9, Sunil Mohan, Bluezy, Carlosguitar, Maxamegalon2000, Teryx, GrinBot, BiH, Aimini, Prab, Elbperle, Veinor, Zso, SmackBot, Evansp, Xkoalax, Reedy, Unyoyega, Augest, Od Mishehu, Cutter, Vald, Bomac, Echoghost, Arny, KelleyCook, HalfShadow, Preeeemo, Gilliam, Ohnoitsjamie, Folajimi, Skizzik, Chris the speller, Bluebot, Coinchon, Jprg1966, Emufarmers, Thumperward, Victorgrigas, Carlconrad, Octahedron80, Trek00, DHN-bot, Konstable, Audriusa, Royboycrashfan, Kzm, Mirshafie, Neiltheffernaniii, Милан Јелисавчић, Preada, TCL, UltraLoser, Nixeagle, JonHarder, Rrburke, Zak123321, Vironex, NoIdeaNick, Radagast83, E. Sn0 =31337=, Bslede, Jiddisch, Funky Monkey, Bernino, Allyant, Jeremyb, Sigma 7, LeoNomis, Cjdkoh, Ck lostsword, Alcuin, Pilotguy, Kukini, Ricky@36, Mbauwens, P2pauthor, SashatoBot, Nishkid64, Harryboyles, Tazmaniacs, Rjdainty1, Gobonobo, Aaronchall, Joshua Andersen, InfinityB, Musicat, Generic69, Scyth3, Flamingblur, F15 sanitizing eagle, Loadmaster, Silvarbullet1, JHunterJ, Mauro Bieg, Tigrisnaga, Ryulong, Rosejn, Peyre, Caiaffa, Teemuk, Fan-1967, Iridescent, Michaelbusch, BrainMagMo, Sschluter, Sirius Wallace, Thommi, Az1568, Courcelles, Gounis, Pjbflynn, Tawkerbot2, FatalError, SkyWalker, JForget, CmdrObot, Bane004, Zarex, Miatatroll, Pmerson, GargoyleMT, Requestion, Pgr94, Kennyluck, Cldnails, Arangana, Phatom87, Ooskapenaar, Jackiechen01, ECELonghorn, Steel, Gogo Dodo, Feedloadr, Farshad83, Vanished user 8jq3ijalkdjhviewrie, DumbBOT, FastLizard4, Kozuch, Nuwewsco, Daniel Olsen, Lo2u, Gimmetrow, Thijs!bot, Epbr123, Coelacan, Oldiowl, Nick Number, Wikidenizen, AntiVandalBot, Lord JoNil, Ingjerdj, Bondolo, Caper13, Bigjimr, Leuko, Davewho2, Dustin gayler, CosineKitty, Albany NY, BrotherE, Geniac, SteveSims, Magioladitis, Antelan, Bongwarrior, VoABot II, Dekimasu, Yandman, JamesBWatson, CobaltBlue, Radio Dan, Mupet0000, Gabriel Kielland, Pausch, M 3bdelqader, 4nT0, Cpl Syx, Kgfleischmann, Deathmolor, RayBeckerman, Stephenchou0722, Aliendude5300, Sachdevj, MartinBot, LinuxPickle, Rettetast, Akkinenirajesh, Webpageone.co.uk, Mattsag, LedgendGamer, Tgeairn, Brunelstudy, Mthibault, Feierbach, Hopper96, Icseaturtles, Karrade, LordAnubisBOT, Touisiau, Wuyanhuiyishi, Aervanath, Cometstyles, SirJibby, Warlordwolf, Remember the dot, Ghacks, Lawman3516, Ahtih, Stanleyuroy, Idioma-bot, Funandtrvl, Soali, Jimmytharpe, VolkovBot, ABF, Hansix, Jeff G., Indubitably, I'mDown, Philip Trueman, Smywee, Hpfreak26, Tourist-Philosopher, Someguy1221, Coldfire82, Una Smith, Lradrama, LotharZ, Seb26, Jackfork, LeaveSleaves, Buryfc, Anishsane, Haseo9999, Gillyweed, SmileToday, VanishedUserABC, Hardistyar, Kbrose, Exile.mind, SieBot, Kwirky88, BotMultichill, Gerakibot, Caltas, Rexguo, Terribim, ACNS, Dattebayo321, Bentogoa, Flyer22, Permacultura, Reinderien, Matthewedwards, Bagatelle, Cshear, Lightmouse, Hobartimus, Kos1337tt, Mattycl, Creative1980, DRTllbrg, Jludwig, Phelfe, Tomdobb, Stedjamulia, Celique, Tuxa, Atif.t2, Augman85, ClueBot, Mlspeten, Binksternet, The Thing That Should Not Be, Cambrasa, Ewawer, Ice77cool, Yamakiri, Alexbot, Diegocr, Abrech, Vivio Testarossa, Kihoiu, Dilumb, Rhododendrites, SchreiberBike, Wvithanage, Cyko 01, Classicrockfan42, ClanCC, Miami33139, XLinkBot, Mmv-ru, Petchboo, OWV, Cwilso, Harisankarh, Little Mountain 5, Cmr08, Lewu, Moose mangle, Harjk, Lajena, Addbot, Imeriki al-Shimoni, VCHunter, Qnext-Support, Tothwolf, Larrybowler, Cuaxdon, MrOllie, Jreconomy, ManiaQ, RogersMD, Jakester23jj, Evildeathmath, Tide rolls, Avono, Gail, SasiSasi, Vincent stehle, Arm-1234, Legobot, Yobot, Tohd8BohaithuGh1, Old Death, Dfe6543, Preston.lee, Knownot, RedMurcury1, Koman90, AnomieBOT, Bwishon, Kristen Eriksen, Sonia, Dinesh smita, FenrirTheWolf, Neilapalmer, Materialscientist, CoMePrAdZ, Citation bot, Teilolondon, LilHelpa, MC707, Ludditesoft, Lairdp, Laboriousme, Mrdoomino, Tad Lincoln, Jmundo, Miym, Mobilon, Abce2, Frosted14, Kevinzhouyan, Zicko1, Felix.rivas, Bahahs, Coolblaze03, Alainr345, Shadowjams, Nyhet, Tknew, Dougofborg, FrescoBot, Sky Attacker, Sae1962, Eliezerb, Drew R. Smith, Tom235, Tiger Brown, Pinethicket, I dream of horses, Btrest, A8UDI, Gabrielgmendonca, Ocexyz, Patrickzuili, Shielazhang, CountZer0, TobeBot, Irvine.david, Vrenator, TBloemink, Stjones86, Jeffrd10, Tyofcore, XDnonameXD, RjwilmsiBot, TjBot, Dangerousrave, Noodles-sb, Slon02, DASHBot, P2prules, EmausBot, WikitanvirBot, Snied, Yoelzanger, K6ka, AvicBot, Juststreamit, Josve05a, Fred Gandt, Wayne Slam, Layona1, Donner60, Senator2029, DASHBotAV, Mattsenate, Rocketrod1960, Helpsome, Will Beback Auto, ClueBot NG, Jack Greenmaven, Lokeshyadav99, Satellizer, Mesoderm, RichardOSmith, Widr, G8yingri, Helpful Pixie Bot, Manja Neuhaus, Lowercase sigmabot, BG19bot, Desmarie17, Absalom23, Metricopolus, RentalicKim, Editerjhon, Skpande, Lesldock, Chazza1113, TRBurton, ChrisGualtieri, ZappaOMati, Ducknish, Profilemine, Codename Lisa, Hmainsbot1, GrayEagle1, Nonnompow, Lugia2453, Rcomrce, Razibot, Epicgenius, Pronacampo9, Tigstep, Maxwell bernard, Nshunter, Cp123127, Cecilia Hecht, Sahil sharma2119, Myconix, Lesser Cartographies, Ginsuloft, Ekilson, AlyssaG92, CBCompton, J grider65, Cespo4, Ppcoinwikipeercoin, Jkielty82, SwiftCrimson, Rosesollere, Drkhataniar, Monkbot, Cazer78, V-apharmd, Vanished user 31lk45mnzx90, W.phillips7, AkashValliath, Emhohensee, Nelsonkam, Gautamdebjani, IPUpfficia, Jesuufamtobie and Anonymous: 1101

- **Mainframe computer** *Source:* http://en.wikipedia.org/wiki/Mainframe%20computer?oldid=648523331 *Contributors:* Damian Yerrick, AxelBoldt, Kpjas, Bryan Derksen, Robert Merkel, Timo Honkasalo, David Merrill, William Avery, Roadrunner, Maury Markowitz, Hephaestos, Leandrod, Edward, Ubiquity, RTC, AdSR, JohnOwens, Tannin, CesarB, Ahoerstemeier, Stevenj, Ugen64, Rob Hooft, OliD, Boson, Dmsar, Reddi, Fuzheado, Darkhorse, Ed g2s, Wernher, Dcsohl, Pakaran, Rossumcapek, Jni, Serek, Robbot, RedWolf, Altenmann, Romanm, Rfc1394, Smb1001, Dng88, Hadal, Mushroom, Ancheta Wis, Takanoha, Giftlite, Mintleaf, Intosi, Sukoshisumo, Everyking, AlistairMcMillan, VampWillow, Bgoldenberg, Bobblewik, Neilc, Comatose51, Chowbok, Slowking Man, Rdsmith4, Lvl, Icairns, Sfoskett, Sam Hocevar, Neutrality, KeithTyler, Avihu, Karl Dickman, Hobart, EagleOne, Metahacker, RossPatterson, Solitude, Loganberry, Pluke, ArnoldReinhold, Martpol, Dyl, Kbh3rd, Charm, Tverbeek, Bobo192, Wood Thrush, Chessphoon, Matt Britt, Jerryseinfeld, Cavrdg, Towel401, Hectigo, Patsw, Alansohn, Polarscribe, Guy Harris, Atlant, Geo Swan, Ricky81682, Sligocki, Samohyl Jan, Velella, Helixblue, Wtshymanski, Harej, Humble Guy, Gunter, Pauli133, Dan East, Alem Dain, Forderud, Brookie, Nuno Tavares, Ruud Koot, Tabletop, Isnow, Toussaint, Mandarax, Slgrandson, Graham87, Qwertyus, FreplySpang, Glasreiniger, Deasmi, Jclemens, Reisio, Rjwilmsi, Scandum, Bubba73, Ian Dunster, FlaBot, RexNL, Gurch, BjKa, Brendan Moody, Bmicomp, Chobot, Karch, Hall Monitor, UkPaolo, YurikBot, Borgx, Angus Lepper, RussBot, AVM, Jengelh, RadioFan, Stephenb, Wimt, SamJohnston, The Hokkaido Crow, Ugur Basak, NawlinWiki, Joel7687, Vanderaj, Megapixie, Mikeblas, Alex43223, Nate1481, Takeel, Jhinman, Navstar, Zzuuzz, Closedmouth, GraemeL, JLaTondre, Rwwww, Finell, SmackBot, Jared555, Rokfaith, KocjoBot, Senordingdong, Chairman S., Sloman, Gilliam, Skizzik, Anwar saadat, Bluebot, Geneb1955, Thom2002, Cbh, Roscelese, Nossac, BBCWatcher, DHN-bot, Da Vynci, Can't sleep, clown will eat me, Erzahler, Onorem, Kcordina, Nonforma, Jmlk17, Jsavit, Lillycrop, Weregerbil, Lambiam, DHR, Kuru, JohnCub, Slakr, Mathewignash, Waggers, Anonymous anonymous, Peyre, Phuzion, JeffW, Iridescent, Wjejskenewr, Chunawalla, DJ HEAVEN, UncleDouggie, Linkspamremover, Tawkerbot2, The Letter J, Raysonho, Wafulz, Dycedarg, Page Up, Baiji, Basawala, Nilfanion, Mblumber, Gogo Dodo, JFreeman, Itsphilip, Dan.j.evans@btinternet.com, Sirianoftatton, Tawkerbot4, Kozuch, Thijs!bot, Epbr123, Kubanczyk, Ultimus, N5iln, Marek69, A3RO, James086, Apantomimehorse, AntiVandalBot, Gioto, Widefox, RDT2, Edokter, Mk*, Karthik sripal, MichaelR., JAnDbot, Arch dude, Esc2006, Goldenglove, Robert Buzink, .anacondabot, Sawney bean, Casmith 789, VoABot II, Sanoj1234, DerHexer, Excesses, Bieb, Gwern, MartinBot, Munier, Jim.henderson, Rettetast, CommonsDelinker, Tgeairn, J.delanoy, Bogey97, NightFalcon90909, Foober, Mahadeva, Chriswiki, NewEnglandYankee, Pterre, Kraftlos, Christopher Kraus, Vachari, Shoessss, DH85868993, WarFox, DorganBot, Idioma-bot, Signalhead, X!, VolkovBot, Nburden, Franck Dernoncourt, Philip Trueman, TXiKiBoT, Oshwah, Jazzgalaxy, Defect17, Walor, T-bonham, 01griste, Anna Lincoln, The Wilschon, Leafyplant, Modal Jig, Dain69, MartinPackerIBM, Shafi.jam, AlleborgoBot, SieBot, Dwandelt, Portalian, WereSpielChequers, RJaguar3, Flyer22, Oda Mari, Elcobbola, Ferret, AnonGuy, Tombomp, Makikiwiki, Dajja78, Jonlandrum, Tony Webster, Fishnet37222, ClueBot, Robenel, Rilak, Mazagnet, Arakunem, Rlbarton, DragonBot, Copyeditor42, Excirial, A plague of rainbows, Sandeep.bhalekar, Dickguertin, Duster.Cleaner, Katanada, XLinkBot, SFFrog, Duncan, SilvonenBot, Airplaneman,

TreyGeek, Addbot, Pyfan, Friginator, AkhtaBot, Ted.macneil, Download, Chzz, GrnScrn, Lightbot, Legobot, Luckas-bot, Yobot, OrgasGirl, Cflm001, Legobot II, Amirobot, Nallimbot, Peter Flass, AnomieBOT, Lucerne2001, Neptune5000, 9258fahsflkh917fas, Crecy99, RandomAct, Materialscientist, Zigoman, ArthurBot, FreeRangeFrog, Xqbot, Vanished user xlkvmskgm4k, Earlypsychosis, RibotBOT, Doulos Christos, Chatul, Milesaaway, Prari, Jc3s5h, Pinethicket, I dream of horses, Calmer Waters, Gkhankz, Ryoohkies, Akolyth, Cinemageddon, Antipastor, Mrdoggyhead, Statham1234, Skakkle, DARTH SIDIOUS 2, Dexter Nextnumber, Alph Bot, Lopifalko, IBMSPECIALIST, TGCP, Indubitabletc, Thexchair, Sreenvasan, Slightsmile, Cmlloyd1969, Wikipelli, Dcirovic, Kiralexis, TyA, L Kensington, MainFrame, ChuispastonBot, ClueBot NG, MelbourneStar, Vsnares, Rangeenbasu, Strike Eagle, BG19bot, Abuo98, Compfreak7, Jimwthompson, Crossreference16, Lukethecreator, Camberleybates, Vikas.ramesh.saxena, Mrt3366, Ccbowman, EuroCarGT, Jethro B, Mogism, WikiEXBOB, Pvtcal, Jamesx12345, BLUEmainframe, Ugog Nizdast, Cokkie7550, My name is not dave, Ginsuloft, Freewayfan99, JaconaFrere, BruceHellmer, Monkbot, Supersonik45, OMPIRE, GeorginaMat and Anonymous: 555

- **Utility computing** *Source:* http://en.wikipedia.org/wiki/Utility%20computing?oldid=640768750 *Contributors:* The Anome, Delirium, Skysmith, Ronz, Phr, Bevo, Robbot, Metapsyche, Mikeroodeus, Everyking, Wmahan, Beland, GreatTurtle, Cretog8, Shenme, Kjkolb, Pearle, Bodhran, Jehochman, Piet Delport, SteveLoughran, Bovineone, SamJohnston, Dipskinny, JoeBruno, THB, Zwobot, Jeh, Raistolo, Rwwww, SmackBot, El Baby, CSWarren, Colonies Chris, Weregerbil, Soumyasch, Kompere, Hu12, UncleDouggie, Randhirreddy, No1lakersfan, Mblumber, Hft, Kozuch, Khcw77, RichardVeryard, Calaka, RobotG, Isilanes, Myanw, Kgfleischmann, Angwill, Mannjc, Chad Vander Veen, Public Menace, SpigotMap, Belovedfreak, Snrjefe, Suyambuvel, Bonadea, RJASE1, Mifam, Softtest123, Eleckyt, Jojalozzo, Andymrhodes, Megacat, Classivertsen, Datacenterguy, ClueBot, Applicationit, SpikeToronto, Jonah Stein, XLinkBot, JimParkerRogers, Addbot, Barmijo, Scientus, Kristiewells, MrOllie, Epatrocinio, Tlausser, Luckas-bot, Yobot, Soggyc, 4twenty42o, Miym, Nakakapagpabagabag, Roman Doroshenko, Miracle Pen, Ycagen, RjwilmsiBot, Emmess2005, WikitanvirBot, Emmess2006, RA0808, Liquiddatallc, DASHBotAV, ClueBot NG, Verbamundi, Helpful Pixie Bot, Jagruti.gh, Xavier.parmantier and Anonymous: 73
- **Wireless sensor network** *Source:* http://en.wikipedia.org/wiki/Wireless%20sensor%20network?oldid=649794023 *Contributors:* Edward, Michael Hardy, Kku, Glenn, Palfrey, Samw, MariusG, Populus, Omegatron, Jni, Cyrius, DavidCary, Mboverload, Ezod, Hgfernan, Joyous!, D6, Discospinster, Rich Farmbrough, JoeSmack, Calavera, Tgeller, Jfcarr, Photonique, Snowolf, Velella, Versageek, Tr00st, Shimeru, Stemonitis, Mindmatrix, Wolfey, Robert K S, Kgrr, Sega381, Toussaint, Tslocum, Jwoodger, MauriceKA, Qwertyus, Rjwilmsi, Tomtheman5, CMorty, Chobot, Antilived, Bgwhite, TheNatealator, Grubber, Gaius Cornelius, Janbeutel, Kkmurray, Nelson50, Wikimaniac17, Wikimaniac18, Katieh5584, Zvika, SmackBot, Dickcaro, Jxjimmy, McGeddon, Grey Shadow, Powo, Diom1982, Commander Keane bot, Skizzik, Bluebot, Pwightman, Jacques.Bovay, Mogman1, Rrelf, Frap, JonHarder, Kittybrewster, Zvar, Allan McInnes, Fitzhugh, FilippoSidoti, ManiacK, Twocs, Canadianshoper, TastyPoutine, Kvng, Iridescent, CmdrObot, Tarchon, Tobes00, Srangwal, Usman one, Haensel, Mblumber, Mato, Cricketgirl, Nitin ravin, Flowerpotman, Herorev, Omicronpersei8, Thijs!bot, Adimallikarjunareddy, Pruetboonma, Nick Number, Dawnseeker2000, Escarbot, AntiVandalBot, Arcturus4669, Anna.foerster, Dougher, Kkim86, AndreasWittenstein, Jh.kang, Barek, Txomin, Muneeb.ali, Gerculanum, Jheiv, Magioladitis, JamesBWatson, JoergBertholdt, David Eppstein, Ingle, WrlssMn, Celaine, Tinyos, Tamer ih, Jenson589, Haffner, J.delanoy, Mange01, Trusilver, Jiuguang Wang, MrBell, Wsnplanet, Kudpung, Grandsonofmaaden, Elrayis, Jeepday, KirkMartinez, Unbound, Rustyguts, Sahyagiri, Netrangerrr, Bonadea, Akarim awwad, Noure04, Ishitasharan, 28bytes, VolkovBot, Umar420e, Philip Trueman, David ocr, Sanajcs, Rohit.nadig, Shu.lei, AlleborgoBot, Dtaverson, Foyh, SieBot, Krawi, Ienlaul, Smithderek2000, Mikebar, Lostgravity, Flyer22, Cialo, Ali asin, Luciole2013, Stoneygirl45, Karl2620, Santafen, Sphilbrick, Bravekermit, 6MarketRoad, Gailyh, ClueBot, Mh-en, Rustic, Rpagliari, Mild Bill Hiccup, GamaFranco, Uncle Milty, SuperHamster, Niceguyedc, Trivialist, Auntof6, Athropos, 3poutsis, Alexbot, PixelBot, Sun Creator, 7&6=thirteen, Dekisugi, Ruzzelli, Aitias, Scaifegibson, SoxBot III, DumZiBoT, XLinkBot, Gnowor, BodhisattvaBot, MensaDropout, Fd42, NellieBly, Cmr08, Alexius08, Lukasz Tlomak, Addbot, DOI bot, MrOllie, Pmod, Laurenmacdonald, Celia.periza, Teles, Zorrobot, Yobot, Ptbotgourou, Francesco Betti Sorbelli, Nallimbot, SwisterTwister, AnomieBOT, Applefat, Jim1138, David1972, Citation bot, Srinivas, Rainman0100, GB fan, LilHelpa, Xqbot, Thiliniishaka, Jmjornet, Shulini, Vvdounai, Miym, Mdsattd, Uncommonj, Alivalizadeh, Madison Alex, Daneshwiki, ????, DanTheSeeker, Citation bot 1, Aboulis, 10metreh, SSchnelbach, RedBot, Rsgray9999, Full-date unlinking bot, Lissajous, Niazim1, Ganjishyam, Maegeri, Japangyro, دال‌با, Praveenkumar88, Sensorwizard1, EmausBot, Qasim Sidd 1987, John of Reading, Ashih.ieeecs, Sourensinha, Sitharama.iyengar1, RA0808, RenamedUser01302013, Solarra, Dnshaw1337, Eken7, Bruno Vernay, Ari81, Niki1984, Thomaswatteyne, Mn mom8429, Rafikmol, Adbatson, Ipsign, Bomazi, Marcbisscheroux, 28bot, ClueBot NG, Hossein172, Wgrace, Tip1424, Organicdev, Davnav, W roonn, Helpful Pixie Bot, BG19bot, JamesQueue, Rijinatwiki, Hallows AG, Sumeshka, Sowsnek, Sasi4289, Mrrfid, 220 of Borg, Cryptos2k, Owoo1, Mrt3366, Zeeyanwiki, TedStepanski, Compte.wiki, SenseOr, SFK2, Graphium, Jamesx12345, Catseyetigereye, Simonetech, Jyheo0, Shashank16392, We07, GingerGeek, Brzydalski, Lesser Cartographies, Rahamatkar s, JaconaFrere, Hossein.ranjbaran.it, Mostafaazami, Adele0622, Wsnsw, Yzhu1, Cesarmarch, Betafive, Fellmark, LindseyH140, Gallusuberalles and Anonymous: 444
- **Internet of Things** *Source:* http://en.wikipedia.org/wiki/Internet%20of%20Things?oldid=650618544 *Contributors:* Damian Yerrick, Deb, Ubiquity, Kku, Ihcoyc, Glenn, Bearcat, Ancheta Wis, Chowbok, Beland, Discospinster, Brianhe, Vsmith, Giraffedata, Ynhockey, Wtmitchell, Wtshymanski, JonSangster, Woohookitty, Ruud Koot, Winterdragon, Ashmoo, Rjwilmsi, Allen Moore, Jezarnold, Ahunt, Bgwhite, Wavelength, AVM, Gaius Cornelius, SamJohnston, Welsh, Red Jay, SmackBot, McGeddon, KVDP, WDavidStephenson, Chris the speller, George Church, Deli nk, Rrelf, Seduisant, Decltype, BullRangifer, DMacks, Ozhiker, Robofish, IronGargoyle, Novangelis, Dl2000, Dansiman, Tamlyn, Patrickwooldridge, Nhumfrey, Loopkid, Ibadibam, Steveliang, Jane023, Sovanyio, Michael Fourman, Deepak.harsha, Widefox, Ivazquez, Shambolic Entity, Thickicesong, Barek, Vladounet, SirDuncan, JamesBWatson, DGG, Jim.henderson, Gaming4JC, Funandtrvl, Deor, Wcrosbie, TooTallSid, Piperh, Billinghurst, Andy Dingley, Fdacosta, Michael Frind, Kevin.anchi, Kbrose, Mikebar, Dawn Bard, Jojalozzo, Svick, Firefly4342, Fergussa, Jbw2, Caskinner, Fangjian, Mild Bill Hiccup, Wurtis65, Rockfang, PixelBot, Muhandes, BirgerH, Rhododendrites, MPH007, Apparition11, MasterOfHisOwnDomain, XLinkBot, Koumz, WikHead, Dubmill, Good Olfactory, Addbot, Mortense, Innv, Texperience, Bte99, Kapaleev, Pgautier-neuze, Jarble, Rcalix1, Legobot, Yobot, Enviro1, Bjoertvedt, Vini 17bot5, Jean.julius, Banjohunter, MihalOrela, AnomieBOT, Seanlorenz, Rejedef, Mihal Orela, Jo3sampl, Bluerasberry, Mquigley8, Citation bot, Xqbot, Philip sheldrake, Gap9551, Solphusion, Ita140188, Omnipaedista, SassoBot, Smallman12q, OtherAdam, Jugdev, FrescoBot, Goldzen, Jokek, Potted1, Sae1962, PeterEastern, Jersey92, Zednik, PigFlu Oink, DrilBot, Winterst, Joebigwheel, Max Harms, Xcvista, Alkapole, Wikitanvir, Jandalhandler, Anna Comnena, Tooncoppens, Ahmed31, Newton09, Peterkaptein, RjwilmsiBot, EmausBot, John of Reading, Smarty9002, WikitanvirBot, GoingBatty, Hscharler, K6ka, AvicBot, ZéroBot, Jonathan Wheeler, Jrtknight, Lcnbst, BetweenMyths, Internetofthings, ✄, Rajrsingh, Javamen, ClueBot NG, Cwmhiraeth, Jmcfarland27, Dimos2k, Cyborg4, Albertojuanse, Helpful Pixie Bot, Bingoal, KLBot2, IoTCruiser, BG19bot, Virtualerian, MusikAnimal, Techman220, Paganinip, Metaprinter, Semanticwebbing, Xenva, Majorbolz, Khalid aldkhaeel, BattyBot, Pbierre, Dwu42, Jsalatas, Internet2Guru, JerDoug, Xbao, ChrisGualtieri, Arcandam, EuroCarGT, IjonTichyIjonTichy, Tomvanvu, Mogism, Makecat-bot, Morfusmax, ElleCP, Rodgerlea, Jlthames2, GeminiDrive, Ivan.v.gerasimov, ReidWender, Arshdeepbahga, Jbirdwell34, Ruby Murray, Csepartha, Joshua Simi, New worl, Murus, Buffbills7701, Jens Haupert, Sensingasaservice, Nuvolaio, Spredge, JoachimLindborg, Skiaustin, Urnhart, Dcautela, Gehrhorn, Andylesavage, Sunny2888, Stamptrader, Hbb9, Carl J. Garcia, Iot coi, Kevalsingh, Wyn.junior, Rotunda2013, RicardoCuevasGarcia, Lagoset, Posicks, Gramamoo,

Monkbot, Stefenev, Mikedeanklein, Jechma, Sofia Koutsouveli, Sightestrp, Avidwriterforever, Elgrancid, Nataliard, Mr.freely, Chwagen, Scienceteacher6410, Drudgeart, Ccofrnzl, Thetechgirl, Oiyarbepsy, Sarasedgewick, Xorain, Gkort23, Cosimomalesci, OdinS Rafael, Mdaliakhtar, Casjonker, Edavinmccoy, Nikhil.sharma2301, Joe00961, Gonzalo.massa, Dwheedon, Fellmark, Wendyavis, SoSivr, Sanjeev.rawat86, Androslaxbos, Jordan.Manser and Anonymous: 211

14.12.2 Images

- **File:2x2x2torus.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/3f/2x2x2torus.svg *License:* CC BY 2.5 *Contributors:* Drawn by ???????? *Original artist:* ????????
- **File:6600GT_GPU.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/4/44/6600GT_GPU.jpg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Berkut
- **File:A790GXH-128M-Motherboard.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/0/0c/A790GXH-128M-Motherboard.jpg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Evan-Amos
- **File:AMD_HD5470_GPU.JPG** *Source:* http://upload.wikimedia.org/wikipedia/en/8/88/AMD_HD5470_GPU.JPG *License:* CC0 *Contributors:*

  Self created

  Original artist:

  highwycombe (talk)
- **File:Ambox_important.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/b4/Ambox_important.svg *License:* Public domain *Contributors:* Own work, based off of Image:Ambox scales.svg *Original artist:* Dsmurat (talk · contribs)
- **File:ArchitectureCloudLinksSameSite.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/ef/ArchitectureCloudLinksSameSite.png *License:* Public domain *Contributors:* Own work *Original artist:* Driquet
- **File:Athlon64x2-6400plus.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/fb/Athlon64x2-6400plus.jpg *License:* CC BY 3.0 *Contributors:* Own work *Original artist:* Babylonfive David W. Smith
- **File:Balanceamento_de_carga_(NAT).jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/3e/Balanceamento_de_carga_%28NAT%29.jpg *License:* CC BY-SA 2.5 *Contributors:* ? *Original artist:* ?
- **File:Beowulf.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/8/8c/Beowulf.jpg *License:* GPL *Contributors:* ? *Original artist:* User Linuxbeak on en.wikipedia
- **File:Beowulf.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/4/40/Beowulf.png *License:* Public domain *Contributors:* Own work *Original artist:* Mukarramahmad
- **File:BlueGeneL_cabinet.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/a/a7/BlueGeneL_cabinet.jpg *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:Bus_icon.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/ca/Bus_icon.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:CUDA_processing_flow_(En).PNG** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/59/CUDA_processing_flow_%28En%29.PNG *License:* CC BY 3.0 *Contributors:* Own work *Original artist:* Tosaka
- **File:CloudComputingSampleArchitecture.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/7/79/CloudComputingSampleArchitecture.svg *License:* GFDL *Contributors:* Scalable Vector Graphic created by Sam Johnston using OminGroup's OmniGraffle *Original artist:* Sam Johnston, Australian Online Solutions Pty Ltd
- **File:Cloud_computing.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/b5/Cloud_computing.svg *License:* CC BY-SA 3.0 *Contributors:* Created by Sam Johnston using OmniGroup's OmniGraffle and Inkscape (includes Computer.svg by Sasa Stefanovic) *Original artist:* Sam Johnston
- **File:Cloud_computing_layers.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/3c/Cloud_computing_layers.png *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Cloud_computing_types.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/8/87/Cloud_computing_types.svg *License:* CC BY-SA 3.0 *Contributors:* wikipedia *Original artist:* Sam Joton
- **File:Commons-logo.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* ? *Contributors:* ? *Original artist:* ?
- **File:Computer-aj_aj_ashton_01.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/c1/Computer-aj_aj_ashton_01.svg *License:* CC0 *Contributors:* ? *Original artist:* ?
- **File:Cray-1-deutsches-museum.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/f7/Cray-1-deutsches-museum.jpg *License:* CC BY 2.5 *Contributors:* Own work *Original artist:* Clemens PFEIFFER
- **File:Crystal_Clear_app_browser.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/fe/Crystal_Clear_app_browser.png *License:* LGPL *Contributors:* All Crystal icons were posted by the author as LGPL on kde-look *Original artist:* Everaldo Coelho and YellowIcon
- **File:Crystal_Clear_app_kedit.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/e8/Crystal_Clear_app_kedit.svg *License:* LGPL *Contributors:* Sabine MINICONI *Original artist:* Sabine MINICONI
- **File:Cubieboard_HADOOP_cluster.JPG** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/27/Cubieboard_HADOOP_cluster.JPG *License:* Public domain *Contributors:* http://dl.cubieboard.org/media/a10-cubieboard-hadoop/IMG_0774.JPG *Original artist:* Cubie Team
- **File:DHT_en.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/9/98/DHT_en.svg *License:* Public domain *Contributors:* Jnlin *Original artist:* Jnlin

- **File:DIAMONDSTEALTH3D2000-top.JPG** *Source:* http://upload.wikimedia.org/wikipedia/en/f/f8/DIAMONDSTEALTH3D2000-top.JPG *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Dstealth32.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/22/Dstealth32.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Swaaye at English Wikipedia
- **File:Dual_Core_Generic.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/ec/Dual_Core_Generic.svg *License:* Public domain *Contributors:* Transferred from en.wikipedia; transferred to Commons by User:Liftarn using CommonsHelper. *Original artist:* Original uploader was CountingPine at en.wikipedia
- **File:E6750bs8.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/a/af/E6750bs8.jpg *License:* Public domain *Contributors:* Transferred from en.wikipedia; transferred to Commons by User:Liftarn using CommonsHelper. *Original artist:* Original uploader was GuitarFreak at en.wikipedia
- **File:Edit-clear.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/f/f2/Edit-clear.svg *License:* Public domain *Contributors:* The *Tango! Desktop Project*. *Original artist:*
  The people from the Tango! project. And according to the meta-data in the file, specifically: "Andreas Nilsson, and Jakub Steiner (although minimally)."
- **File:Flag_of_Australia.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/b/b9/Flag_of_Australia.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Flag_of_Brazil.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/0/05/Flag_of_Brazil.svg *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Flag_of_Canada.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/c/cf/Flag_of_Canada.svg *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Flag_of_France.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/c/c3/Flag_of_France.svg *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Flag_of_Germany.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/b/ba/Flag_of_Germany.svg *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Flag_of_India.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/4/41/Flag_of_India.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Flag_of_Japan.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/9/9e/Flag_of_Japan.svg *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Flag_of_Russia.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/f/f3/Flag_of_Russia.svg *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Flag_of_the_Netherlands.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/20/Flag_of_the_Netherlands.svg *License:* Public domain *Contributors:* Own work *Original artist:* Zscout370
- **File:Flag_of_the_People'{}s_Republic_of_China.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/fa/Flag_of_the_People%27s_Republic_of_China.svg *License:* Public domain *Contributors:* Own work, http://www.protocol.gov.hk/flags/eng/n_flag/design.html *Original artist:* Drawn by User:SKopp, redrawn by User:Denelson83 and User:Zscout370
- **File:Flag_of_the_Republic_of_China.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/7/72/Flag_of_the_Republic_of_China.svg *License:* Public domain *Contributors:* [1] *Original artist:* User:SKopp
- **File:Flag_of_the_United_States.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/a/a4/Flag_of_the_United_States.svg *License:* PD *Contributors:* ? *Original artist:* ?
- **File:Folder_Hexagonal_Icon.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/4/48/Folder_Hexagonal_Icon.svg *License:* Cc-by-sa-3.0 *Contributors:* ? *Original artist:* ?
- **File:Fork_join.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/f1/Fork_join.svg *License:* CC BY 3.0 *Contributors:* w:en:File:Fork_join.svg *Original artist:* Wikipedia user A1
- **File:Front_Z9_2094.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/21/Front_Z9_2094.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Ing. Richard Hilber
- **File:IBM_704_mainframe.gif** *Source:* http://upload.wikimedia.org/wikipedia/commons/7/7d/IBM_704_mainframe.gif *License:* Attribution *Contributors:* ? *Original artist:* Lawrence Livermore National Laboratory
- **File:IBM_Blue_Gene_P_supercomputer.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/d3/IBM_Blue_Gene_P_supercomputer.jpg *License:* CC BY-SA 2.0 *Contributors:* originally posted to **Flickr** as Blue Gene / P *Original artist:* Argonne National Laboratory's Flickr page
- **File:IBM_HS20_blade_server.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/20/IBM_HS20_blade_server.jpg *License:* CC BY-SA 2.0 *Contributors:* http://www.flickr.com/photos/jemimus/66531212/ (original size version) *Original artist:* Robert Kloosterhuis
- **File:Inside_Z9_2094.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/6/6d/Inside_Z9_2094.jpg *License:* Public domain *Contributors:* Transferred from de.wikipedia; transferred to Commons by User:Mewtu using CommonsHelper.
  Original artist: Ing. Richard Hilber. Original uploader was Rhilber at de.wikipedia
- **File:Internet_map_1024.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/d2/Internet_map_1024.jpg *License:* CC BY 2.5 *Contributors:* Originally from the English Wikipedia; description page is/was here. *Original artist:* The Opte Project
- **File:Internet_of_Things.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/5a/Internet_of_Things.png *License:* Public domain *Contributors:* Apendix F of Disruptive Technologies Global Trends 2025 page 1 Figure 15 (Background: The Internet of Things) *Original artist:* SRI Consulting Business Intelligence/National Intelligence Council
- **File:MEGWARE.CLIC.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/c5/MEGWARE.CLIC.jpg *License:* CC-BY-SA-3.0 *Contributors:* http://www.megware.com *Original artist:* MEGWARE Computer GmbH

- **File:Marktanteil_GPU-Hersteller.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/20/Marktanteil_GPU-Hersteller.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Mark H.
- **File:Mergefrom.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/0/0f/Mergefrom.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Motherboard_diagram.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/bd/Motherboard_diagram.svg *License:* CC-BY-SA-3.0 *Contributors:* Transferred from en.wikipedia; transferred to Commons by User:Moxfyre using CommonsHelper. *Original artist:* user:Moxfyre. Original uploader was Moxfyre at en.wikipedia
- **File:Nec-cluster.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/1a/Nec-cluster.jpg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Hindermath
- **File:OpenMP_language_extensions.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/9/9b/OpenMP_language_extensions.svg *License:* Public domain *Contributors:* en:Image:Omp lang ext.jpg *Original artist:* en:User:Khazadum, User:Stannered
- **File:Openmp.png** *Source:* http://upload.wikimedia.org/wikipedia/en/2/27/Openmp.png *License:* Fair use *Contributors:*
  From http://www.openmp.org/drupal/node/view/16 *Original artist:* ?
- **File:P2P-network.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/3f/P2P-network.svg *License:* Public domain *Contributors:* Own work *Original artist:* User:Mauro Bieg
- **File:Processor_families_in_TOP500_supercomputers.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/ef/Processor_families_in_TOP500_supercomputers.svg *License:* CC BY-SA 3.0 *Contributors:* Own work, intending to create a vector version of File:Top500.procfamily.png *Original artist:* Moxfyre
- **File:Question_book-new.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:*
  Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:*
  Tkgd2007
- **File:SPEC-1_VAX_05.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/ec/SPEC-1_VAX_05.jpg *License:* CC BY-SA 3.0 *Contributors:* Photo by Joe Mabel *Original artist:* Joe Mabel
- **File:Server-based-network.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/fb/Server-based-network.svg *License:* LGPL *Contributors:* derived from the Image:Computer n screen.svg which is under the GNU LGPL *Original artist:* User:Mauro Bieg
- **File:Structured_(DHT)_peer-to-peer_network_diagram.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/7/79/Structured_%28DHT%29_peer-to-peer_network_diagram.png *License:* CC0 *Contributors:* Inkscape *Original artist:* Mesoderm
- **File:Sun_Microsystems_Solaris_computer_cluster.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/7/7c/Sun_Microsystems_Solaris_computer_cluster.jpg *License:* CC BY 2.0 *Contributors:* Flickr *Original artist:* ChrisDag
- **File:Supercomputer_Share_Top_500_by_Country_Jun_2014.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/eb/Supercomputer_Share_Top_500_by_Country_Jun_2014.png *License:* CC BY-SA 3.0 *Contributors:* Made using Microsoft Excel 2013 and importing data from www.top500.org *Original artist:* Dsfarcturus
- **File:Supercomputing-rmax-graph2.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/b8/Supercomputing-rmax-graph2.svg *License:* CC0 *Contributors:* Own work *Original artist:* Morn
- **File:Top20supercomputers.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/4/4e/Top20supercomputers.png *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* Python.kochav
- **File:Torrentcomp_small.gif** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/3d/Torrentcomp_small.gif *License:* CC-BY-SA-3.0 *Contributors:* https://en.wikipedia.org/wiki/BitTorrent → smaller file-size GIF for BitTorrent article, cleaned up the dithered and ugly pixels. I made this in Photoshop to replace the monstrous 1.77 MB GIF currently residing on that article's page. *Original artist:* Wikiadd
- **File:Unstructured_peer-to-peer_network_diagram.png** *Source:* http://upload.wikimedia.org/wikipedia/en/f/fa/Unstructured_peer-to-peer_network_diagram.png *License:* CC0 *Contributors:*
  Inkscape
  Original artist:
  Mesoderm
- **File:Voodoo3-2000AGP.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/8/88/Voodoo3-2000AGP.jpg *License:* CC-BY-SA-3.0 *Contributors:* Transferred from en.wikipedia; transferred to Commons by User:JohnnyMrNinja using CommonsHelper. *Original artist:* Original uploader was Swaaye at en.wikipedia
- **File:WSN.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/21/WSN.svg *License:* Public domain *Contributors:* Own work *Original artist:* Original by Adi Mallikarjuna Reddy V (en:User:Adimallikarjunareddy), converted to SVG by tiZom
- **File:Wide-angle_view_of_the_ALMA_correlator.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/50/Wide-angle_view_of_the_ALMA_correlator.jpg *License:* CC BY 4.0 *Contributors:* http://www.eso.org/public/images/eso1253a/ *Original artist:* ESO
- **File:Wiki_letter_w_cropped.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/1c/Wiki_letter_w_cropped.svg *License:* CC-BY-SA-3.0 *Contributors:*
- Wiki_letter_w.svg *Original artist:* Wiki_letter_w.svg: Jarkko Piiroinen
- **File:Wikibooks-logo-en-noslogan.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/df/Wikibooks-logo-en-noslogan.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.
- **File:Yacy-resultados.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/f1/Yacy-resultados.png *License:* GFDL *Contributors:* Trabajo propio/captura de pantalla *Original artist:* User:Hack-Master

14.12.3 Content license

- Creative Commons Attribution-Share Alike 3.0