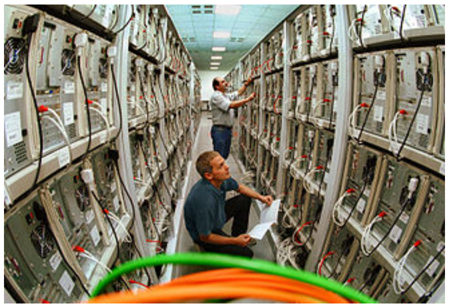
WIKIPEDIA

Computer cluster

A **computer cluster** is a set of loosely or tightly connected computers that work together so that, in many aspects, 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.


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

The components of a cluster are usually connected to each other through fast local area networks, 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[1] and the same operating system, although in some setups (e.g. using Open Source Cluster Application Resources (OSCAR)), different operating systems can be used on each computer, or different hardware.[2]

Sun Microsystems Solaris Cluster, with In-Row cooling

Clusters 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.[3]

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.[4] Prior to the advent of clusters, single unit fault tolerant mainframes with modular redundancy were employed; but the lower upfront cost of clusters, and increased speed of network fabric has favoured the adoption of clusters. In contrast to high-reliability mainframes clusters are cheaper to scale out, but also have increased complexity in error handling, as in clusters error modes are not opaque to running programs.[5]

Contents

Basic concepts

A simple, home-built Beowulf cluster.

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.[6] 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.[6]

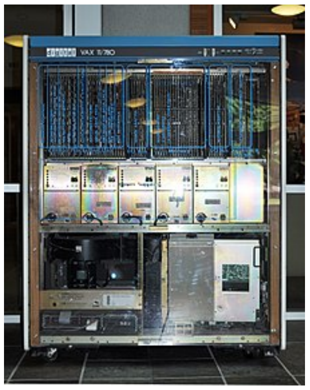
Computer clustering relies on a centralized management approach which makes the nodes available as orchestrated shared servers. It is distinct from other approaches such as peer to peer or grid computing which also use many nodes, but with a far more distributed nature.[6]

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-node Stone Soupercomputer.[7] The developers used Linux, the Parallel Virtual Machine toolkit and the Message Passing Interface library to achieve high performance at a relatively low cost.[8]

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.[9]

History

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.[10] 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.


A VAX 11/780, c. 1977

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.

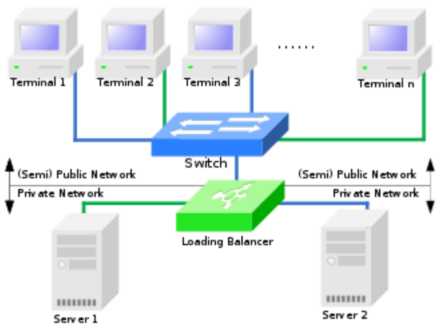
The first production system designed as a cluster was the Burroughs B5700 in the mid-1960s. This allowed up to four computers, each with either one or two processors, to be tightly coupled to a common disk storage subsystem in order to distribute the workload. Unlike standard multiprocessor systems, each computer could be restarted without disrupting overall operation.

The first commercial loosely coupled 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.[11] 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.

Attributes of clusters

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.



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

"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.[12] 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.[12]

Computer clusters are used for computation-intensive purposes, rather than handling IO-oriented operations such as web service or databases.[13] 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.

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*) allows for scalability, and in high performance situations, low frequency of maintenance routines, resource consolidation (e.g. RAID), and centralized management. Advantages include enabling data recovery in the event of a disaster and providing parallel data processing and high processing capacity.[14][15]

In terms of scalability, clusters provide this in their ability to add nodes horizontally. This means that more computers may be added to the cluster, to improve its performance, redundancy and fault tolerance. This can be an inexpensive solution for a higher performing cluster compared to scaling up a single node in the cluster. This property of computer clusters can allow for larger computational loads to be executed by a larger number of lower performing computers.

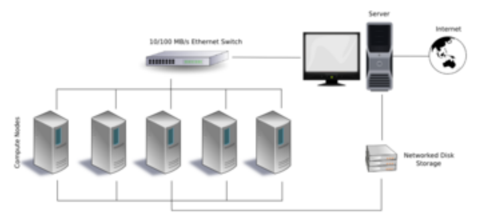
When adding a new node to a cluster, reliability increases because the entire cluster does not need to be taken down. A single node can be taken down for maintenance, while the rest of the cluster takes on the load of that individual node.

If you have a large number of computers clustered together, this lends itself to the use of distributed file systems and RAID, both of which can increase the reliability and speed of a cluster.

Design and 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 cluster, 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.[13] 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



A typical Beowulf configuration.

network of the organization.[13] 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.[13]

A 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.[16]

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 are still much less costly (purchase cost).[2]

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.[17] 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.[17]

Data sharing and communication

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.

Message passing and communication

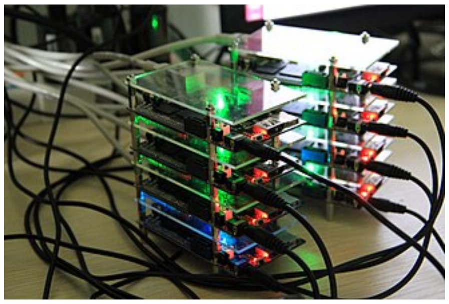
Two widely used approaches for communication between cluster nodes are MPI (Message Passing Interface) and PVM (Parallel Virtual Machine).[18]

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.[18][19]

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.[18] MPI is now a widely available communications model that enables parallel programs to be written in languages such as C, Fortran, Python, etc.[19] Thus, unlike PVM which provides a concrete implementation, MPI is a specification which has been implemented in systems such as MPICH and Open MPI.[19][20]

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.[21] In some cases this provides an advantage to shared memory architectures with lower administration costs.[21] This has also made virtual machines popular, due to the ease of administration.[21]


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

Task scheduling

When a large multi-user cluster needs to access very large amounts of data, task scheduling becomes a challenge. 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.[22] This is an area of ongoing research; algorithms that combine and extend MapReduce and Hadoop have been proposed and studied.[22]

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.[23][24] 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.[23]

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.[23]

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.

Software development and administration

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.[25]

Automatic parallelization of programs remains 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.[25][26]

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.[19][27] 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.[19]

Application checkpointing can be used to restore a given state of the system when a node fails during a long multi-node computation.[28] 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.[28]

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).

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.

See also

Basic concepts

- Clustered file system
- Heartbeat private network
- High-availability cluster
- Single system image
- Symmetric multiprocessing

Distributed computing

- Distributed computing
- Distributed data store
- Distributed operating system
- Distributed shared memory

Specific systems

- DEGIMA (computer cluster)
- K computer
- Microsoft Cluster Server
- Red Hat Cluster Suite
- Rocks Cluster Distribution
- Solaris Cluster
- Veritas Cluster Server

Computer farms

- Compile farm
- Render farm
- Server farm

References

1. "Cluster vs grid computing" (https://stackoverflow.com/questions/9723040/what-is-the-difference-between-cloud-grid-and-cluster). *Stack Overflow*.
2. Graham-Smith, Darien (29 June 2012). "Weekend Project: Build your own supercomputer" (http://www.pcauthority.com.au/Feature/306972,weekend-project-build-your-own-supercomputer.aspx). *PC & Tech Authority*. Retrieved 2 June 2017.

3. Bader, David; Pennington, Robert (May 2001). "Cluster Computing: Applications" (https://web.archive.org/web/20071221011621/http://www.cc.gatech.edu/~bader/papers/ijhpca.html). Georgia Tech College of Computing. Archived from the original (http://www.cc.gatech.edu/~bader/papers/ijhpca.html) on 2007-12-21. Retrieved 2017-02-28.
4. "Nuclear weapons supercomputer reclaims world speed record for US" (https://www.telegraph.co.uk/technology/9338651/Nuclear-weapons-supercomputer-reclaims-world-speed-record-for-US.html). The Telegraph. 18 Jun 2012. Retrieved 18 Jun 2012.
5. Gray, Jim; Rueter, Andreas (1993). *Transaction processing : concepts and techniques* (https://archive.org/details/transactionproce0000gray). Morgan Kaufmann Publishers. ISBN 978-1558601901.
6. *Network-Based Information Systems: First International Conference, NBIS 2007*. p. 375. ISBN 3-540-74572-6.
7. William W. Hargrove, Forrest M. Hoffman and Thomas Sterling (August 16, 2001). "The Do-It-Yourself Supercomputer" (http://www.sciam.com/article.cfm?id=the-do-it-yourself-superc). *Scientific American*. **265** (2). pp. 72–79. Retrieved October 18, 2011.
8. Hargrove, William W.; Hoffman, Forrest M. (1999). "Cluster Computing: Linux Taken to the Extreme" (https://web.archive.org/web/20111018122713/http://climate.ornl.gov/~forrest/linux-magazine-1999/). *Linux Magazine*. Archived from the original (http://climate.ornl.gov/~forrest/linux-magazine-1999/) on October 18, 2011. Retrieved October 18, 2011.
9. Yokokawa, Mitsuo; et al. (1–3 August 2011). *The K computer: Japanese next-generation supercomputer development project*. International Symposium on Low Power Electronics and Design (ISLPED). pp. 371–372. doi:10.1109/ISLPED.2011.5993668 (https://doi.org/10.1109%2FISLPED.2011.5993668).
10. Pfister, Gregory (1998). *In Search of Clusters* (https://archive.org/details/insearchofcluste00pfis/page/36) (2nd ed.). Upper Saddle River, NJ: Prentice Hall PTR. p. 36 (https://archive.org/details/insearchofcluste00pfis/page/36). ISBN 978-0-13-899709-0.
11. Hill, Mark Donald; Jouppi, Norman Paul; Sohi, Gurindar (1999). *Readings in computer architecture*. pp. 41–48. ISBN 978-1-55860-539-8.
12. Sloan, Joseph D. (2004). *High Performance Linux Clusters* (https://archive.org/details/highperformancel0000sloa). ISBN 978-0-596-00570-2.
13. Daydé, Michel; Dongarra, Jack (2005). *High Performance Computing for Computational Science - VECPAR 2004*. pp. 120–121. ISBN 978-3-540-25424-9.
14. "IBM Cluster System : Benefits" (https://web.archive.org/web/20160429022854/http://www-03.ibm.com/systems/clusters/benefits.html). IBM. Archived from the original (http://www-03.ibm.com/systems/clusters/benefits.html) on 29 April 2016. Retrieved 8 September 2014.
15. "Evaluating the Benefits of Clustering" (https://web.archive.org/web/20160422092651/https://technet.microsoft.com/en-us/library/cc778629%28v%3Dws.10%29.aspx). Microsoft. 28 March 2003. Archived from the original (https://technet.microsoft.com/en-us/library/cc778629(v=ws.10).aspx) on 22 April 2016. Retrieved 8 September 2014.
16. Hamada, Tsuyoshi; et al. (2009). "A novel multiple-walk parallel algorithm for the Barnes–Hut treecode on GPUs – towards cost effective, high performance N-body simulation". *Computer Science - Research and Development*. **24** (1–2): 21–31. doi:10.1007/s00450-009-0089-1 (https://doi.org/10.1007%2Fs00450-009-0089-1). S2CID 31071570 (https://api.semanticscholar.org/CorpusID:31071570).
17. Mauer, Ryan (12 Jan 2006). "Xen Virtualization and Linux Clustering, Part 1" (http://www.linuxjournal.com/article/8812). *Linux Journal*. Retrieved 2 Jun 2017.
18. Milicchio, Franco; Gehrke, Wolfgang Alexander (2007). *Distributed services with OpenAFS: for enterprise and education* (https://books.google.com/books?id=_HtHG2Ca5AEC). pp. 339–341. ISBN 9783540366348.
19. Prabhu, C.S.R. (2008). *Grid and Cluster Computing* (https://books.google.com/books?id=evcgB7Qlix4C). pp. 109–112. ISBN 978-8120334281.

20. Gropp, William; Lusk, Ewing; Skjellum, Anthony (1996). "A High-Performance, Portable Implementation of the MPI Message Passing Interface". *Parallel Computing*. **22** (6): 789–828. CiteSeerX 10.1.1.102.9485 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.102.9485). doi:10.1016/0167-8191(96)00024-5 (https://doi.org/10.1016%2F0167-8191%2896%2900024-5).
21. Patterson, David A.; Hennessy, John L. (2011). *Computer Organization and Design*. pp. 641–642. ISBN 978-0-12-374750-1.
22. K. Shirahata; et al. (30 Nov – 3 Dec 2010). *Hybrid Map Task Scheduling for GPU-Based Heterogeneous Clusters*. Cloud Computing Technology and Science (CloudCom). pp. 733–740. doi:10.1109/CloudCom.2010.55 (https://doi.org/10.1109%2FCloudCom.2010.55). ISBN 978-1-4244-9405-7.
23. "Alan Robertson Resource fencing using STONITH" (https://web.archive.org/web/20210105195318/https://mirrors.sinuspl.net/www.linux-ha.org/heartbeat/ResourceFencing_Stonith.pdf) (PDF). *IBM Linux Research Center, 2010*. Archived from the original (https://mirrors.sinuspl.net/www.linux-ha.org/heartbeat/ResourceFencing_Stonith.pdf) (PDF) on 2021-01-05.
24. Vargas, Enrique; Bianco, Joseph; Deeths, David (2001). *Sun Cluster environment: Sun Cluster 2.2* (https://books.google.com/books?id=oRjWAZS5iZMC). Prentice Hall Professional. p. 58. ISBN 9780130418708.
25. Aho, Alfred V.; Blum, Edward K. (2011). *Computer Science: The Hardware, Software and Heart of It* (https://books.google.com/books?id=S7QU9RRLYIYC). pp. 156–166. ISBN 978-1-4614-1167-3.
26. Rauber, Thomas; Rünger, Gudula (2010). *Parallel Programming: For Multicore and Cluster Systems* (https://books.google.com/books?id=wWogxOmA3wMC). pp. 94–95. ISBN 978-3-642-04817-3.
27. Francioni, Joan M.; Pancake, Cherri M. (April 2000). "A Debugging Standard for High-performance computing" (http://dl.acm.org/citation.cfm?id=1239906). *Scientific Programming*. Amsterdam, Netherlands: IOS Press. **8** (2): 95–108. doi:10.1155/2000/971291 (https://doi.org/10.1155%2F2000%2F971291). ISSN 1058-9244 (https://www.worldcat.org/issn/1058-9244).
28. Sloot, Peter, ed. (2003). *Computational Science-- ICCS 2003: International Conference*. pp. 291–292. ISBN 3-540-40195-4.

Further reading

- Baker, Mark; et al. (11 Jan 2001). "Cluster Computing White Paper". arXiv:cs/0004014 (https://arxiv.org/abs/cs/0004014).
- Marcus, Evan; Stern, Hal (2000-02-14). *Blueprints for High Availability: Designing Resilient Distributed Systems* (https://archive.org/details/blueprintsforhig00marc). John Wiley & Sons. ISBN 978-0-471-35601-1.
- Pfister, Greg (1998). *In Search of Clusters* (https://archive.org/details/insearchofcluste00pfis). Prentice Hall. ISBN 978-0-13-899709-0.
- Buyya, Rajkumar, ed. (1999). *High Performance Cluster Computing: Architectures and Systems*. **1**. NJ, USA: Prentice Hall. ISBN 978-0-13-013784-5.
- Buyya, Rajkumar, ed. (1999). *High Performance Cluster Computing: Architectures and Systems*. **2**. NJ, USA: Prentice Hall. ISBN 978-0-13-013785-2.

External links

- IEEE Technical Committee on Scalable Computing (TCSC) (https://web.archive.org/web/20190219183441/https://www.ieeetcsc.org/)
- Reliable Scalable Cluster Technology, IBM (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=%2Fcom.ibm.cluster.rsct.doc%2Frsctbooks.html)

- Tivoli System Automation Wiki (https://www.ibm.com/developerworks/wikis/display/tivoli/Tivoli+System+Automation)
- Large-scale cluster management at Google with Borg (https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/43438.pdf), April 2015, by Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune and John Wilkes