Sun[tm] Based Beowulf Cluster

By Börje Lindh - Sun Microsystems AB, Sweden

Sun BluePrints™ OnLine - December 2001

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 816-3636-10
Revision 1.0, 12/03/01
Edition: December 2001

Please
Recycle

Sun™ Based Beowulf Cluster

For users who are doing numerical calculations, building compute clusters is fairly common. This document describes what a compute cluster is, what a Beowulf cluster is, and what you should keep in mind when building a compute cluster. It also describes how to build an optimized compute cluster based on Sun hardware and software.

What is a Compute Cluster?

Traditionally a cluster is a solution of multiple computers which provide one or more highly available services. For example, this solution might be a database, a product data management (PDM) server, or some other vitally important business service.

A high-performance compute (HPC) cluster is completely different. It is a group of computing resources seen as a single resource by the end user. A compute cluster is used to run traditional HPC applications across the resource, parallelized applications using message passing technology, or when a large number of data sets is considered, throughput applications. It is also used to run any combination of these applications.

For example, a chip developer wants to verify part of a chip design. This application must make many simulations in which there is different input data, simulate a number of clock cycles, and then verify that the output data is correct. Instead of running these computations on a workstation, the chip developer can submit many simulations to a large compute cluster and receive all results in one hour instead of several days.

Or, assume that you are working on a large code development project and must recompile 1,000 source files for a complete rebuild. Instead of doing it all on your workstation, you can submit all of the jobs to a large compute cluster, and have the build ready in minutes instead of hours.

Another popular class of applications to run on a compute cluster is computational fluid dynamics (CFD) simulations, in which the problem can be divided easily into multiple parts that can be run in parallel on different nodes, thus decreasing the total simulation time dramatically.

A Beowulf cluster is a popular variant of a compute cluster that is based on PC hardware and free open source software. Subsequent sections of this article discuss Beowulf clusters in detail.

Different Types of Compute Jobs

Compute jobs can be classified into three different groups:

- Single threaded—The traditional UNIX® process with one address space and one execution thread.
- Multithreaded—One address space but multiple execution threads that can run in parallel if multiple CPUs are available in the machine. A multithreaded program can handle both fine grain and course grain parallelism.
- Multiprocess—Multiple processes executing the same program simultaneously. Each process can be single threaded or multithreaded. Communication between processes is done using message passing. This solution can be used only for course grain parallelism. Otherwise, the communication becomes dominant.

Currently, most computer programs are still single threaded, which yields sufficient performance for most uses. Only when the program execution takes too long does the programmer make the extra effort required to parallelize the program.

A multithreaded program only has one address space (just as in a single-threaded program), but there are multiple execution threads, each with its own stack and so forth. Parallelizing a program into a multithreaded program can be done two different ways. One possibility is to direct the compiler to parallelize the program. In some cases, this method might be enough to achieve sufficient performance, but in most cases more work is needed. The second possibility is to help the compiler by adding compiler directives to the program. Compiler directives are hints that tell the compiler what parts of the code are safe to run in parallel. You can also use compiler directives in some parts of a program and let the compiler do the other parts. Currently, the most common development environment for shared memory parallelization is OpenMP.

Single-threaded and multithreaded applications can be developed using standard compilers; no special runtime environment is required. Multithreaded applications running in a single address space must be run on a multiprocessor computer if you want to utilize more than one CPU for a single job.

For some applications, depending on the structure of the application, it is better to use the message passing approach. This approach does, however, require more programming skills because the programmer must manually structure the code and insert all subroutine calls for parallelization, synchronization, and so forth. In the past, the parallel virtual machine (PVM) subroutine library was commonly used, but currently most parallel programs are written using an implementation of the message passing interface (MPI). Open source software implementations of MPI, such as MPICH, and commercial versions, such as Sun HPC ClusterTools™ 4.0 software are both available. TABLE 1 lists the software required for these applications.

TABLE 1 Software Required for Single-Threaded, Multithreaded, and Multiprocessed Applications

Type	Single threaded	Multithreaded	Multiprocess
Development	Forte™ software	Forte software	Forte software plus Sun HPC ClusterTools 4.0 software
Execution	Solaris™ Operating Environment (Solaris OE)	Solaris OE	Solaris OE plus Sun HPC ClusterTools software

Of course, it is possible to mix methods and create an application that is both multithreaded and multiprocessed; that is, multiple processes in which each process is multithreaded, running on different nodes.

Building a Compute Cluster

A production compute cluster is an configuration of a number of machines into a single computing resource. Instead of starting a job on a specific machine (or host), the user submits a job to a queue. The queuing system runs the job on the best available machine. It is possible, although not very common, to run interactive jobs through the queuing system, too. If a user submits the task of running an `xterm`, the queuing system will start the task on a lightly-loaded host.

A compute cluster consists of the following parts:

- Network—Ethernet, Myrinet, and so forth
- File sharing—Network File System (NFS), Andrew file system (AFS), or distributed file system (DFS)
- Queuing system—Sun™ Grid Engine hardware, parallel batch system (PBS), or Platform Computing's Platform LSF (load-sharing facility)
- Message passing (if used)—an MPI library (Sun HPC ClusterTools 4.0 software or MPICH)
- Compiler (if needed)—Forte 6.2 software or GNU
- Maintenance tools—JumpStart™ software, automatic patch installation tools, and so forth
- Administration tools—hardware health checking tools (SunVTS™ software, Sun™ Management Center software (hereafter called Sun MC), resource allocation (Solaris™ BandWidth Manager software, disk quota, Network Information System (NIS), and so forth)
- Terminal servers for the consoles

Job execution depends on how the queuing system is configured. You can optimize for the use of expensive software licenses, maximize total resource utilization, prioritize a certain group of users, and so forth.

Note that the optimal application development environment may be different than the optimal production environment. In a development environment, response time is more important than throughput.

To benefit the user and system administrator the most, the cluster must have these characteristics

- Powerful
- Simple to use
- Easy to program
- Simple to administer
- Easy to add more resources
- Good price and performance

Computing Resources Needed

Different organizations require different computing resources; no optimal solution fits all. Some commercial applications, especially in the CFD and crash simulation areas are available in parallel versions (both multithreaded and MPI). If a parallel version of the application is available and the input data is such that the performance scales well, you can improve performance significantly by running the job in parallel. A CFD simulation runs seven times faster on eight CPUs compared to one CPU. Thus, the results are available overnight instead of in two days. In most organizations the majority of the jobs are single threaded. Some applications scale well to a modest number of CPUs; a few applications scale to a large number of CPUs. The exact numbers differ among organizations, but a similar distribution is often seen. FIGURE 1 shows a typical workload distribution.

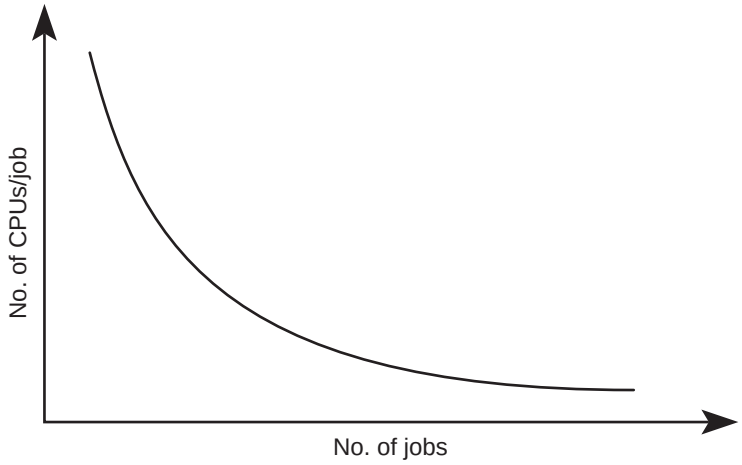

FIGURE 1 Typical Workload Distribution Graph—No. of CPUs Per Job Versus No. of Jobs

Price Per CPU

The price per CPU is generally greater in a large computer than in a small one. The reason is that a single CPU machine is much simpler to design, cheaper to build, and is often produced in a large series of machines. However, if the application is multithreaded and you want to run it in parallel, you must run it on a parallel machine—symmetric multiprocessor (SMP) or Cache-Coherent, Non-Uniform Memory Architecture (CC-NUMA). Also, a larger machine can have more internal memory, which the application might require. A large SMP is easier to use and administrate than a cluster of smaller machines. If you use the same chip in a 750 MHz Sun Blade™ 1000 workstation and a 750 MHz Sun Fire™ 6800 server the price per CPU in the Sun Fire 6800 server is much higher, because the system is much more complex. FIGURE 2 shows a plot of price per CPU versus machine size.

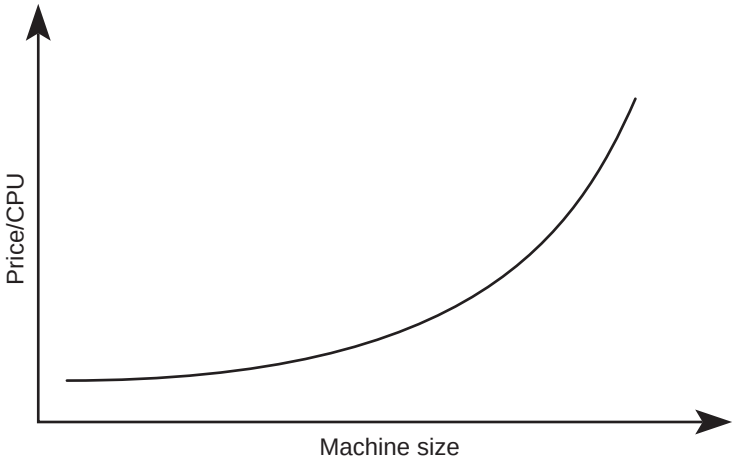

FIGURE 2 Cost of CPU—Price Per CPU Versus Machine Size

Optimal Solution Economics

From an economical standpoint, the optimal solution is to run all jobs on the smallest and least expensive machines possible. Single-threaded jobs should be run on single CPU machines. Multithreaded jobs that only scale to a few CPUs run most economically on a small or medium-size machine with several CPUs. Only jobs that scale to many CPUs, and for which the execution wall time (real time) is very important, such as weather simulations and large crash simulations, should be run on a large machine or cluster.

These requirements indicate that the optimal compute cluster for a large organization is probably many small single CPU machines, a few medium-size machines, and one or more large parallel machines. FIGURE 3 shows the number of CPUs per job versus the number of jobs.

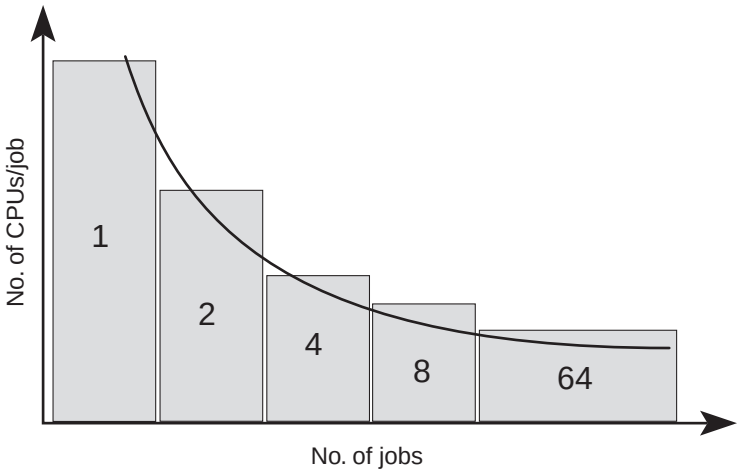

FIGURE 3 Optimal Computer Cluster Graph—No. of CPUs Per Job Versus No. of Jobs

Similarly, for some applications, such as electronic design automation (EDA), memory is the most important factor. Most EDA simulation tools are single threaded and can utilize only one CPU, but you might need 10 Gbytes or more of memory for a large verification.

A very important consideration is the optimal use of expensive licenses. The annual license fee for an EDA tool is often in the order of $10000 per CPU; thus, it is important to run as many jobs as you have licenses for, and to run these jobs on the fastest CPUs available. When looking at a solution for multiprocess applications, you must also look at the communication characteristics. An application with minimal communication needs, such as a CFD calculation with a static grid, might

run very well over a slow interconnect like Ethernet. If the communication needs are greater, such as those of CFD with a moving grid (when simulating the interior of a cylinder in an engine), a low-latency interconnect like Myricom's Myrinet or an SMP, in which the processes can communicate through shared memory, will perform much better.

Often, many unused computing resources are available within an organization. In a common computer aided design (CAD) or software development environment, the CPUs in the workstations are used only 5 to 10 percent of the time on average. Usually, these machines are unused at nights, during weekends, and on holidays. Even during normal working hours, users attend meetings, make phone calls, eat lunch, and so forth. Often, all of this CPU time is wasted. By setting up a compute cluster on these workstations, a large portion of the wasted resources can be made available to the users in a very simple way. To the organizations, this means that their existing investments can yield 10 to 20 percent more. You should note though, that this configuration requires sufficient network bandwidth between the computing resources to transfer the data needed for computing.

One example of a very successful compute cluster is at Saab Automobile, where 130 CAD workstations are used for background CFD simulations.

Beowulf Solution

Beowulf is one way to build a compute cluster. A Beowulf cluster is configured with PC hardware running the Linux operating system and other open source software. Often, standard Fast Ethernet is used for the interconnect, but sometimes Gigabit Ethernet or Myrinet are used. The file sharing is NFS and the compilers are GNU. If parallel jobs are to be processed, the open source MPICH MPI software implementation is used. For batch queuing, PBS is the most commonly used software. Because the clusters use inexpensive PC hardware and free open source software, the initial cost for building a Beowulf cluster is very low. Note, however, that Solairs OE and SPARC™ product combinations that cost less than $1,000 are available, too. So the perception that only a cluster based on PC hardware is inexpensive is incorrect. The drawbacks of using the PC solution are that you must support the solution yourself and most PC hardware is still 32 bit, so you cannot run any applications that require much memory. The current PC hardware and the Linux operating system cannot handle many CPUs efficiently in an SMP environment. This limitation means that the traditional Beowulf implementation of a compute cluster

only fits the low end of the computing spectrum, plus some parallel applications that also scale well over a slow interconnect. TABLE 2 summarizes the typical PC-based Beowulf components.

TABLE 2 PC-Based Beowulf Cluster Components

Service	Component
Architecture	X86
Operating System	Linux
Network	Ethernet or Myricom Myrinet
File sharing	NFS
Queuing system	PBS or Sun Grid Engine hardware
Message passing interface	MPICH
Compiler	GNU
Management tools—job management	PBS or Sun Grid Engine hardware

Beowulf Cluster on SPARC Hardware

You could just as easily build a Beowulf cluster based on Sun hardware. The Linux operating system is available for SPARC, as are the GNU compilers, MPICH, and PBS.

TABLE 3 SPARC hardware Beowulf Cluster Components

Service	Component
Architecture	SPARC architecture
Operating System	Linux
Network	Ethernet or Myricom Myrinet
File sharing	NFS
Queuing system	PBS or Sun Grid Engine hardware
Message passing interface	MPICH
Compiler	GNU
Management tools—job management	PBS or Sun Grid Engine hardware

SUN Supported Beowulf Cluster

The SPARC hardware cluster solution shares one characteristic with the Beowulf PC cluster solution that many commercial customer would like to avoid—a single vendor does not support the hardware and software components. However, by exchanging the software to supported versions every thing except the compiler would be free of charge.

TABLE 4 Sun Supported Beowulf Cluster Components

Service	Components
Architecture	SPARC architecture
Operating system	Solaris OE
Network	Ethernet or Myricom Myrinet
File sharing	NFS
Queuing system	Sun Grid Engine hardware
Message passing interface	Sun HPC ClusterTools 4.0 software
Compiler	Forte software
Management tools Hardware monitoring Operating environment monitoring Resource monitoring Job management	 SunVTS software, Sun MC software) Sun MC software Solaris™ Resource Manager software Solaris BandWidth Manager software Sun Grid Engine hardware

How To Build Your Compute Cluster

To build an optimal compute cluster, you must:

1. Identify your current computing needs. For each type of job you must know:
   a. Type: single threaded, multithreaded, or multiprocess
   b. Memory requirement per process
   c. Execution time per process
   d. If multiprocess, communication intensity. (Do you need SMP or fast link?)
2. Estimate how the requirements will change in the near future.
3. Find out if any applications are available only on a specific platform (you can have multiple platforms in your cluster.
4. Find out if any existing machines (workstations, and so forth) can be used as a computing resource, either as is or with a slight modification (such as adding more memory).
5. Make the cluster manageable as a single resource:
   a. Develop or configure tools to “soft update” or migrate the operating system.
   b. Develop or configure tools to automatically manage patch installation.
   c. Make any license server scalable and highly available.
   d. Develop or configure tools to run applications from any compute node and other nodes from which you want users to be able launch jobs.
   e. Develop or configure tools to make your compute cluster manageable from a single point of control (that is, from a single display).
6. Decide on, install, and configure a queuing system (Sun Grid Engine hardware, PBS, and so forth). You need one machine as the queue master. If you do not already have file sharing , you need it between all machines plus access for all the services (NIS, and so forth). Depending on your applications, you may also need to install other software such as MPI libraries.
7. Train the users to submit their jobs to a queue instead of running everything interactively.
8. Decide whether additional computing resources are needed, and try to fill the needs with the smallest machines possible. That is, make single CPU machines handle the single CPU workload (if they can have enough memory), use dual

CPU machines for the dual CPU load, and so forth. To minimize administration when you add machines that are to be used as a computing resource only, the machines should be as similar as possible, network installed, and not have local data storage (other than temporary files).

9. If some applications are parallelized, but are moderately communication intensive, you should consider connecting a few machines through a fast interconnect such as Myrinet or include an SMP.

10. If some applications are parallel, but are heavily communication intensive, add one or more larger machines to the cluster. The same is true if there are multithreaded applications in the application mix.

Advantages of a Sun Based Cluster

The advantages of building a compute cluster based on Sun products instead of the Beowulf PC cluster components are:

- Uniform software environment across all nodes in the cluster, from the smallest blade to the largest SMP. This uniformity enables both horizontal scaling (adding more nodes) and vertical scaling (adding more resources per node).
- Full 64-bit support of both hardware and software processes requiring very large memories—more than 500 Gbytes of physical memory for a single process.
- Solaris supports very large CPU counts; more than 100 CPUs with a single operating system image and good scaling.
- Available Sun workstations can be used in the cluster.
- Good administrative tools are available.
- A very large set of optimized, commercial software packages is available.
- All components are tightly integrated and supported by a single vendor.
- The Prism™ multiprocess debugger is available at no cost to develop MPI codes. No free equivalent exists for a PC-based Beowulf cluster.
- The availability of source code for software tools is sometimes cited as an advantage of the pure Beowulf solution. However, the source code for the Sun Grid Engine hardware and Sun HPC ClusterTools 4.0 software, including Sun MPI software and the Prism debugger, are available at no cost.
- The addition of a few larger nodes to the mix allows the cluster to handle capability computing in addition to the capacity computing that has been traditionally associated with Beowulf clusters, while maintaining the cost effectiveness of the capacity solution.

These advantages mean that you can build a compute cluster with nodes as simple as the Netra™ X1 server. This configuration allows up to 40 CPUs per rack, and is very effective in an environment running only single-threaded jobs requiring up to 2 Gbytes per process. If the memory requirements are high, nodes like the Sun Fire 280R server or the Sun Blade 1000 workstation can handle up to 8 Gbytes.

If you need to run multithreaded on more that two CPUs, you can add larger nodes such as the Sun Fire 3800, the Sun Fire 4800, or Sun Fire 6800 servers (24 CPUs with 192 Gbytes of memory) to the compute cluster.

For parallel nodes with heavy communication requirements, the performance increase from communicating over the high-bandwidth SMP backplane should more than offset the greater cost of these nodes on a per-CPU basis.

The biggest computer currently available from Sun is the 104 CPU Sun Fire™ 15K server with 576 Gbytes of internal memory. Thus, you can build a cluster with mixed capabilities that is precisely tailored to the expected job mix in your environment, and meets your computing requirements in the most cost effective manner.

Grid Computing

Building a compute cluster for a single organization like a company or a university campus is standard procedure today. In the future, we see two different trends. One trend is to make the computing resources more easily available over the Web. The other trend is to connect separate clusters over the Internet and make them appear to the users (GLOBUS, LEGION, and so forth) as a single computing resource. Much work must still be done in this area because input and output data must be transmitted securely over the Internet; only some users should be allowed access to this data. Sun is leveraging its heritage in network computing to help define this future.

Conclusion

The popular Beowulf clusters are only one implementation of compute clusters. Despite its popularity, it is a very limited solution that does not address all computing needs. Compute clusters built with Sun hardware and software pick up where a Beowulf solution falls short. They subsume the Beowulf capabilities and extend them to meet the demands of your environment.

Building an optimal cluster is a complex task. This article described how to determine the optimal cluster solution for an organization, and how it can be built. The first challenge is to make the hardware framework, including compute nodes, network, and common services such as file and licenses scalable to fit all types of applications. The second, and more complicated, challenge is to make the software framework manageable and usable as a single resource. A special focus on the management of operating system, patches, applications, licenses, hardware resources, jobs, and users is needed. Development and support of all of the components by a single vendor is a definite advantage.

The most important consideration, however, is for organizations to use of all the computing resources currently available before they invest in new equipment. This goal can be achieved quite easily with available tools at no cost, and can greatly increase the productivity of the engineering community within an organization.

Compute Cluster Software

Sun HPC ClusterTools 4.0 Software

The Sun HPC ClusterTools 4.0 software is Sun's current MPI implementation, which is based on the MPI development environment that Sun acquired from Thinking Machines. In addition to the MPI libraries, this configuration contains a parallel file system (PFS), a Sun™ Scalable Scientific Subroutine Library (Sun S3L), and Prism software. Prism software is a very powerful debugger and data visualizer. The Sun HPC ClusterTools 4.0 software supports jobs running on up to 2048 CPUs and 64 nodes.

Forte 6.2 Software

Although the GNU compilers are available free for the Solairs OE, HPC users are usually interested in getting the maximum performance from their system. For these users, the Forte development environment is a better choice, because the Forte compilers generally produce more efficient code on SPARC systems with the Solairs OE.

The Forte 6.2 software is the current version of Sun's development environment for single-threaded and multithreaded applications. Forte software supports C, C++, Fortran, and the OpenMP programming model for shared memory parallelization. A

very extensive, optimized subroutine library called Perflib is included, plus a complete development environment with context sensitive editors, source browser, debugger, and so forth.

The Forte documentation, like other Sun product documentation, is available at `http://docs.sun.com`.

Compiler Optimization

For HPC users who are writing their own applications or compiling applications available from others in source format, it is important that the compilers generate, as the default. the correct code in the shortest time possible,. This requirement enables rapid program development. However, the generated code is far from optimal. To generate optimized code, which might run several times faster, the programmer must direct the compiler to do so. On the GNU compilers, -O is the basic flag for this directive. For the Forte compiler, `-fast` is a set of options that are usually safe to use for code optimization.

If available, you should use routines from heavily optimized performance libraries such as BLAS, rather than writing these routines again.

The compiler documentation includes information on compiler flags. For more information, you can contact Sun.

Sun Grid Engine Hardware

Sun Grid Engine hardware is Sun's batch queuing system. It is based on Codine/GRD from Genias, which was bought by Sun in 2000. The current version (5.2.3) is available free at `http://www.sun.com` for the Solaris OE and SPARC hardware; for the Linux operating system and *x*86 hardware. Because the source code is also available, binaries for other operating systems can be built too. However, only the Solaris OE on SPARC hardware is supported by Sun.

Myrinet

Myrinet is a low-latency cluster interconnect from Myricom. Drivers are available for the Solaris OE and Sun HPC ClusterTools 4.0 software. For further information, see `http://www.myri.com`.

PBS

PBS is available for most platforms. For more information, see `http://www.pbspro.com`.

Author's Bio: Börje Lindh

Börje has over 16 years of UNIX experience. He joined Sun in 1994 as a Systems Engineer, and is currently a Technical Computing Specialist at Sun Sweden.