01 Jan 2001

Workstation Clusters for Parallel Computing

J. Stone

Fikret Erçal
Missouri University of Science and Technology, ercal@mst.edu

Recommended Citation

J. Stone and F. Erçal, "Workstation Clusters for Parallel Computing," *IEEE Potentials*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2001.
The definitive version is available at https://doi.org/10.1109/45.954655

Workstation clusters for parallel computing

© Jeff Nishinaka/PhotoDisc

Workstation clusters have become an increasingly popular alternative to traditional parallel supercomputers for many workloads requiring high performance computing. The use of parallel computing for scientific simulations has increased tremendously in the last ten years, and parallel implementations of scientific simulation codes are now in widespread use. There are two dominant parallel hardware/software architectures in use today: distributed memory, and shared memory. Systems implementing shared memory provide cooperating processes with a shared memory address space that can be accessed by all processors. In shared memory systems, parallel processing occurs through the use of shared data structures, or through emulation of message passing semantics in software. Distributed memory systems are composed of a number of interconnected computational nodes, which do not share memory, but can communicate with each other through a high-performance network of some kind. Parallelism is achieved on distributed memory systems with multiple copies of the parallel program running on different nodes, sending messages to each other to coordinate computations. The messages used in a distributed memory parallel program typically contain application data, synchronization information, and other data that controls the execution of the parallel program.

Workstation clusters can be built from similar workstations networked together using a high performance interconnect of some kind. Since they can be built from common off-the-shelf components, workstation clusters often enjoy a tremendous price/performance advantage over traditional supercomputers. With PC's and workstations now available in the $1000 price range, clusters have become an extremely compelling way to run computationally demanding simulations used in science and engineering. The choice of workstation and interconnect are the two most important decisions in planning a cluster, since they tend to have the greatest effect on price, performance and manageability.

Two widely used parallel message passing systems, MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) allow the same software to run on clusters of workstations and traditional parallel supercomputers. Before they caught on, most parallel programs were written using vendor-specific message passing systems not portable to other platforms. As MPI and PVM gained success, the long sought after goal of portable parallel programs became a reality. They also helped to legitimize workstation clusters as a practical alternative to traditional parallel computing hardware for many kinds of parallel computation. Many different, freely available, implementations of MPI will run on most Unix systems. In fact, many workstation vendors offer commercial MPI implementations tuned to run at peak performance on their hardware and operating systems. There is at least one commercial implementation of MPI for Windows NT. PVM is also freely available for a large number of Unix systems as well as Windows NT.

The most important question to answer when building a cluster is what kinds of applications will be run on it? Applications that do a lot of message passing, or have very little tolerance for message passing latency, require a higher performance interconnect than those that do not. The balance of processor speed and interconnect speed is an important consideration when building a workstation cluster. To get the best performance for the money (when acquiring new hardware), evaluate the requirements for the applications to be executed on the system. In general it is much easier to design a workstation cluster tuned for a specific type of application or a small domain of applications than for broader general-purpose parallel computing tasks.

Another important concern in building a workstation cluster is how well it will integrate into the existing comput-

John Stone and Fikret Ercal

ing environment. It is easier to manage workstation hardware and software already familiar at a given site than hardware and software that are significantly different. This is one of the biggest drawbacks to traditional parallel supercomputers. They have historically been "special" requiring new administration skills and additional training. Also, traditional parallel supercomputers have a much smaller selection of software packages than for general-purpose workstations. By applying the same or similar workstations used in an organization, one can utilize existing software and expertise.

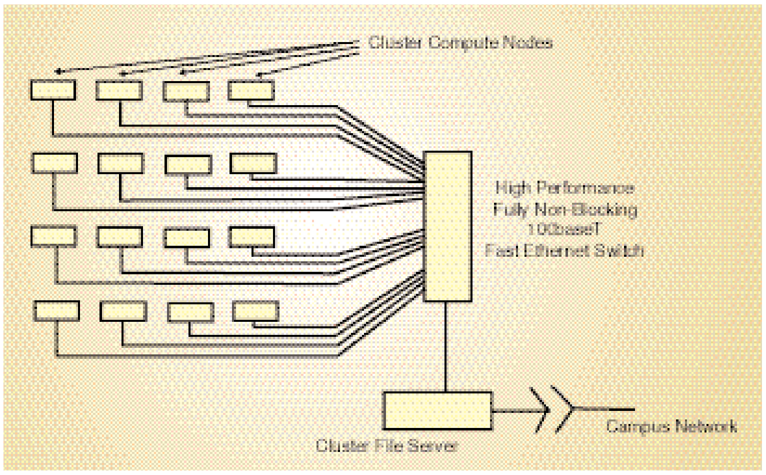

Fig. 1 Cluster Interconnect Configuration

Unlike a workstation laboratory environment, a parallel workstation cluster must meet performance requirements beyond those required in most general purpose computing environments. To understand why, one must consider how parallel computing software is typically used:

• The cluster should perform as a parallel computing resource, achieving higher performance than possible using assprted workstations configured in a more standard way.

• The nodes in the cluster are always used in groups, not individually as in a general purpose workstation laboratory.

• Users run jobs on the cluster through job execution scripts or applications. Users should never need to login to compute nodes. Batch queuing systems are frequently employed along with the job execution facilities included with MPI and PVM.

• Large clusters can be a challenge to maintain if they depend on many external services. It is best to dedicate a server or servers to the cluster rather than using existing servers that may not be capable of providing the performance or reliability necessary to run a large number of cluster compute nodes. If servers are dedicated for use by the workstation cluster, it can also be much easier to coordinate software upgrades and hardware maintenance.

Cluster compute node hardware

Using the guidelines just described, we can now discuss some example hardware configuration choices and their implications. Several general machine and network characteristics are of particular interest when choosing hardware for cluster compute nodes. These include processor speed, cache size, memory bandwidth, memory capacity, network bandwidth, and network latency. The individual effects of these characteristics on overall performance depend greatly on application requirements. Some applications perform very well on a cluster that maximizes only one or two of these characteristics. Parallel molecular dynamics simulations in structural biology are an example of an application that performs well even on systems with limited memory bandwidth and modest memory capacity. Other applications such as computational fluid dynamics simulations are demanding in almost all of the areas listed. Another factor to take into account is the scale of the problem or simulation that will be run on a cluster. The scale of a problem often has significant effects on its network bandwidth demands and network latency tolerance. Larger data sets usually provide an application with greater tolerance for network latency as more data is typically exchanged between compute nodes in a given phase of communication.

Once application and hardware performance characteristics have been taken into consideration, other factors such as integration into the computing environment and maintainability must be considered. It is often the case that several different types of hardware will meet application performance criteria but that some of the options will be vastly preferable due to pre-existing local expertise, physical space, cooling, or other factors. Institutions with site licenses for software development tools or high performance numerical libraries often prefer cluster hardware that can utilize the existing licensed software. It is worth noting that some applications stand to benefit significantly from the use of advanced compilers and numerical libraries. Since software efficiency improvements from compilers and libraries act as a multiplicative factor on sequential performance, their benefits tend to scale with the size of a cluster. For a cluster of more than 32 nodes, an advanced compiler or numerical library costing a few thousand dollars may actually be a better choice than purchasing additional compute nodes if it benefits the majority of the applications that the cluster will run.

Physical space limitations, power, and cooling are all considerations when building large clusters. Buying hardware which can be rack mounted can be advantageous when space is at a premium, but adds to the cost of compute nodes, eroding some of the price/performance advantages that commodity hardware typically offers. Software checkpoint capabilities are often used at the application level to avoid the need for uninterruptible power when building a cluster on a budget. Power and cooling capacity can become a problem when building large clusters, and is a site-specific issue for which there aren't many short cuts.

It is worth considering what the lifespan of a cluster's compute nodes will be, and what to do with them when they are no longer fast enough for the intended applications. One strategy that has been successfully employed by several institutions is to recycle compute nodes as desktop workstations after approximately two years of service. Two years is enough time to allow for a doubling in processor speed in newly purchased equipment and is short enough that recycled compute nodes will still be viable desktop computers. If compute nodes are to be recycled as desktop computers, compatibility with and similarity to machines and software in the existing computational environment becomes even more important.

Typical cluster configurations

A typical workstation cluster configuration found in many institutions is that of a group of 16 or more identical compute nodes, interconnected using a 100baseT Ethernet switch, with one ore more file servers or "master" nodes providing file service, job execution, and scheduling functions. Some clusters use more sophisticated networking components such as gigabit Ethernet or Myrinet, which provide increased bandwidth, decreased latency, or both. Some institutions have made creative use of multiple network interfaces per node and multiple switched networks as an inexpensive alternative to gigabit Ethernet and Myrinet for achieving improved bandwidth and latency while retaining the price/performance advantages of commodity hardware.

Cluster compute nodes usually run the minimum system software necessary to perform their tasks. Each node is installed with a minimalistic operating system and any components of the MPI or PVM software that must be installed or configured locally to each node. All other software including compilers, debuggers, MPI headers and libraries are installed on the cluster file server. Cluster compute nodes may only access files that reside local to each node or on the cluster file server. The use of minimal operating system software on compute nodes benefits both performance and security. One successful system that takes this philosophy to its logical extreme is the Scyld Beowulf distribution (www.scyld.com), which essentially stores nothing on the compute nodes, net booting them from a master node that provides them with everything they need.

The cluster file server or "master node" is usually installed with the full complement of development tools, libraries and other software. This is similar to workstations in general-purpose workstation laboratories. The server often contains a significant amount of local disk storage. This space is made available for cluster users as a temporary storage area for large data files and programs. The storage area on the cluster server is visible to all compute nodes, and is the only shared storage area available to all the cluster's nodes.

In normal use, cluster users copy data files and programs binaries to the storage area during execution of jobs on the cluster. While jobs are running, data may be read from and written to this area. The contents of the storage are not automatically erased on a regular basis; however, there are no guarantees made as to the long-term availability of data left on the cluster server. Cluster users typically depend on other computer resources for long-term storage of their programs and data.

MPICH

A popular version of MPI found on many clusters is the "MPICH" distribution from Argonne National Labs and Mississippi State University. In a cluster environment, MPICH operates by spawning processes on the nodes based on a *machines* file and parameters given to the *mpirun* command. The *machines* file lists each of the nodes in the cluster by name. When a user invokes the mpirun command, *mpirun* parses the machines file and spawns processes on each node in the cluster. This is done in the order they are listed in the machines file. If the user requests more nodes than are physically available, mpirun can be made to spawn multiple processes per machine until the requested number of processes have been started.

The *mpirun* command spawns processes on the nodes in the cluster using one of two methods:

• Processes are spawned using the rsh command or a functional equivalent.

• Processes are spawned using a special daemon process that runs on each node in the cluster.

If the mpirun command fails to successfully spawn one of the processes, the parallel program may hang indefinitely if it is not written carefully. If a user manually kills the hung *mpirun* command by pressing ctrl-c, zombie processes may be left running on the nodes. They may degrade the performance of subsequent parallel jobs to run on the cluster. In the pathological case, a large number of such processes may consume all of the available memory on a given compute node preventing any jobs from executing.

LAM/MPI

The LAM (Local Area Multicomputer) MPI distribution uses a slightly different method for spawning processes on the compute nodes. The user first creates a group of server daemons running on the compute nodes. These daemons are spawned using the *rsh* command. The daemons themselves act as an "allocation" of the nodes in the cluster. Once these daemons are started, all subsequent MPI jobs the user starts get run on that set of compute nodes.

The LAM daemons are similar to the ones used by the MPICH distribution but rather than running as root, the LAM daemons run as the user invoking the job. The user must create and remove these daemons through the use of various LAM provided utilities.

PVM

Like the MPI distributions, PVM uses *rsh* to spawn processes on the compute nodes of a workstation cluster. PVM generates daemons on each of the machines a user intends to use based on a machine list supplied by the user and other mechanisms. Unlike most of the MPI distributions, PVM allows a user to add or delete machines from a running parallel program on the fly. PVM adds and deletes nodes from its working set by spawning and removing invocations of *pvmd*, which runs on each node in the active working set.

Read more about it

• Message Passing Interface Forum. *MPI: A message-passing interface standard*. Technical Report CS-94-230, Computer Science Department, University of Tennessee, Knoxville, TN, May 5 1994. Also appeared in *International Journal of Supercomputing Applications*, Vol. 8, No. 3/4, 1994.

• Al Geist, Adam Beuelin, Jack Dongarra, Weicheng Jiang, R obert Manchek, and Vaidy Sunderam. *PVM: Parallel Virtual Machines, A User's Guide and Tutorial for Net worked Parallel Computing*. MIT Press, 1994.

• Thomas L. Sterlin, John Salmon, Donald J. Becker, Daniel F. Savarese. *How To Build a Beowulf: A guide to the Implementation and Application of PC Clusters*. MIT Press, 1999.

• Implementations of MPI and PVM area available from these web sites: www.mpi.nd.edu/lam (LAM MPI), www.mcs.anl.gov/mpi/mpich (MPICH), www.epm.ornl.gov/pvm (PVM).

• Many large built clusters have web pages worth reading, some are linked from this useful site: www.beowulf.org

• Both NCSA and PSC are now building large supercomputer-class clusters of high-performance workstation systems: www.ncsa.uiuc.edu, www.psc.edu

About the authors

John Stone got his BS/MS in Comp. Sci. from the Univ. of Missouri-Rolla. He is a Senior Research Programmer with the Beckman Institute for Advanced Science and Technology (University of Illinois).

Dr. Fikret Ercal is a professor in Computer Science at the University of Missouri-Rolla.