White Paper

February 23, 2000

Prepared by *CustomSystems* High Performance Computing Segment

Compaq Computer Corporation

Contents

Considerations in Specifying Beowulf Clusters

Abstract: Compaq Computer Corporation has substantial experience and success in deploying supercomputer solutions. We offer complete turnkey solutions or any level of consulting or design support our customers require.

Our supercomputer offerings include tightly coupled systems such as the *AlphaServer SC* running *Compaq Tru64 UNIX* as well as the increasingly popular Beowulf clusters built from combinations of *Alpha* or Intel® processors using the Linux operating system.

This white paper focuses on Beowulf clusters built on the Alpha platform and provides an overview of the factors to consider in determining the best configuration for a given application and budget. The choices one makes in configuring a cluster are highly dependent on the class of application to be run. With an understanding of the analytical process required to specify a Beowulf cluster, you will be able to accelerate the speed at which you can deploy a solution.

For more information on Compaq's high-performance capabilities visit
www.compaq.com/hpc
or
www.compaq.com/solutions/customsystems/hps/index.html

COMPAQ

Introduction

Compaq Computer Corporation has substantial experience and success in deploying supercomputer solutions. We offer complete turnkey solutions or any level of consulting or design support our customers require.

Our supercomputer offerings include tightly coupled systems such as the *AlphaServer SC* running *Compaq Tru64 UNIX* as well as the increasingly popular Beowulf clusters built from combinations of Alpha or Intel processors using the Linux operating system.

This white paper focuses on Beowulf clusters employing the Alpha platform and provides an overview of the factors to consider in determining the best configuration for a given application and budget. The choices one makes in configuring a cluster are highly dependent on the class of applications to be run. With an understanding of the analytical process required to specify a Beowulf cluster, you will be able to accelerate the speed at which you can deploy a solution.

1. Overview

Beowulf is *not* a particular product. It is a concept for clustering varying numbers of small, relatively inexpensive computers running the Linux operating system. The goal of Beowulf clustering is to create a parallel-processing supercomputer environment at a price well below that of conventional supercomputers.

Beowulf Origins

In the summer of 1994 Thomas Sterling and Don Becker, working at The Center of Excellence in Space Data and Information Sciences (CESDIS) under the sponsorship of the Earth and Space Sciences (ESS) project, built a cluster computer consisting of 16 DX4 processors connected by channel-bonded Ethernet. They called their machine *Beowulf.*

The first Beowulf was built to address problems associated with the large data sets that are often involved in ESS applications. The machine was an instant success and their idea of providing COTS (Commodity off the Shelf) systems to satisfy specific computational requirements quickly spread through NASA and into the academic and research communities. The High Performance Computer community refers to such machines as *Beowulf Class Cluster Computers*.

[Reference: http://www.beowulf.org/intro.html]

Creating parallel virtual supercomputers

Beowulf is one approach to clustering commodity hardware components to form a parallel virtual supercomputer. It is a system that usually consists of one system-management node and one or more compute nodes connected via Ethernet or some other network or system area network interconnect.

It is ideal for tackling very complex problems that can be split up and run simultaneously in separate computers. And that's a key point: not every problem can be approached in parallel so not every problem will benefit from the Beowulf approach.

In the taxonomy of parallel computers, Beowulf clusters fall somewhere between MPP (Massively Parallel Processors, like the nCube, CM5, Convex SPP, Cray T3D, Cray T3E, etc.) and NOWs (Networks of Workstations).

Beowulf clusters benefit from developments in both classes. MPPs are typically larger and have a lower latency and higher bandwidth system interconnect network than a Beowulf cluster. Programmers need to consider locality, load balancing, granularity, and communication overheads in order to obtain the best performance. Even on shared-memory machines, many programmers develop their programs in a message-passing style. Programs that do not require fine-grain computation and communication can usually be ported and run effectively on Beowulf clusters.

Programming a NOW is usually an attempt to harvest unused cycles on an already installed base of workstations in a lab or on a campus. Programming in this environment requires algorithms that are extremely tolerant of load balancing problems and large communication latency. Any program that runs on a NOW will run at least as well on a Beowulf cluster.

A Beowulf class cluster computer is distinguished from a NOW by several subtle but significant characteristics. First, the nodes in the cluster are dedicated to the cluster. This helps ease load balancing problems because the performance of individual nodes is not subject to external factors. Also, since the interconnection network is isolated from the external network, the network load is determined only by the application being run on the cluster.

This eases the problems associated with unpredictable latency in NOWs. All the nodes in the cluster are within the administrative jurisdiction of the cluster. For example, the interconnection network for the cluster is not visible from the outside world so the only authentication needed between processors is for system integrity. On a NOW, one must be concerned with network security.

Another distinguishing feature is the Beowulf software that provides a global process ID. This enables a mechanism for a process on one node to send signals to a process on another node of the system, all within the user domain. This is not allowed on a NOW.

Finally, operating system parameters can be tuned to improve performance. For example, a workstation should be tuned to provide the best interactive feel (instantaneous responses, short buffers, etc.), but in a cluster the nodes can be tuned to provide better throughput for coarser-grain jobs because they are not interacting directly with users.
[Reference: http://www.linuxpowered.com/html/tutorials/Beowulf-HOWTO.html]

2. Building a Beowulf Cluster

To build a Beowulf cluster you need the following components:

- ➔ Some number of compute nodes
- ➔ System management and configuration node
- ➔ Linux operating system
- ➔ Storage devices (several options are available)
    - – diskless boot support
    - – central file stores
    - – individual node disks
- ➔ A system interconnect to tie all the compute nodes together
- ➔ System considerations
- ➔ Middleware: the message-passing protocols for coordination and data dissemination
- ➔ A parallelized application to run on the system

Compute nodes – How many? What type?

In order to determine the number and type of compute nodes, you need to ascertain the performance requirements for the class of application that you will run. To do this requires the use of some common metrics and the ability to understand those metrics in terms of peak and sustained performance. Don’t let peak performance -- which might never be attainable by a real application -- sway a decision to go a certain way when sustained performance is what really counts. Any of the decisions made when configuring a Beowulf cluster rely on a comprehensive understanding of the application to be run. Metrics to consider include but are not limited to

- ➔ Floating Point Operations per Second (FLOPS)
- ➔ SPECfp performance
- ➔ SPECint performance
- ➔ Cache size
- ➔ Cache bandwidth

- ➔ Total memory
- ➔ Memory bandwidth
- ➔ Storage bandwidth
- ➔ System interconnect throughput
- ➔ System interconnect latency
- ➔ Bisection bandwidth
- ➔ The effect of interconnect topologies such as meshes, cubes, and fat trees on application design

As you can see, one can't leap blindly into the creation of a Beowulf cluster. Compaq consulting engineers with experience in building large-scale Beowulf clusters are ready to work with customers and partners seriously considering the application of Beowulf clustering.

Linux operating system

The Linux operating system and the Alpha Linux community have supported Compaq's Alpha products since November of 1995 when Red Hat Software, Inc., released the first full non-Intel distribution of Linux for *Alpha*. *Alpha* Linux is a true 64-bit implementation that allows for more accurate mathematics as well as larger address spaces.

Compaq engineers have been part of the Linux community since May of 1994, making major contributions to the development of *Alpha* Linux . Compaq employees have also participated in many Linux activities. In fact, Compaq was the first system vendor to join Linux International, a vendor organization devoted to the promotion of Linux, and is a Sponsoring Corporate Member.

Other Linux distributions include (but are not limited to) Debian, SuSE, Caldera, TurboLinux, and Slackware. Compaq intends to qualify its hardware with the major Linux distributions to give our customers and partners the greatest flexibility to meet their needs.

Storage devices

Some applications might demand several terabytes of central file storage (connected via Fibre Channel, SCSI, UltraSCSI, etc.) whereas others might suffice with the storage available in the individual nodes. Another system configuration option is diskless boot support. Storage bandwidth must be considered based on the requirements of your application and all configuration options.

System Interconnects

Software control messages and data both move over the system interconnect. Any network interconnect -- local area network (LAN) or system area network (SAN) -- can be used to connect all the nodes to each other although some interconnects are faster than others. Here is a list of potential interconnects for a Beowulf cluster supported by the *Compaq CustomSystems* High Performance Computing group available today and in the future:

Interconnect	Network Type	HPC Configuration Support
Fast Ethernet	LAN	Standard
Gigabit Ethernet	LAN	Future
Myrinet	SAN	Standard
Quadrics	SAN	Future
ServerNet2	SAN	Future
ATM	LAN	Project only
FDDI	LAN	Project only
HiPPI	SAN	Project only

COMPAQ

Characterizing System Area Networks
There are two very important measures used to characterize system interconnects. The first is speed. How fast does data pass through the network? Choices of speed run from Fast Ethernet at 10 megabits per second to specialized network technologies clocking 1 gigabyte per second or higher.

The other measurement used to characterize system interconnects is latency. How long does it take to prepare a packet of information for transmission and get it onto the network? How long does it take to propagate through the network and all it's various potential stages or hierarchies? Latencies can range from milliseconds in some older technologies to just a few microseconds in newer ones.

The two measurements – speed and latency -- taken together define "throughput" -- the total amount of useable aggregate data that can be moved from one system to the other. Clearly, throughput can greatly affect the overall performance of a cluster.

Scalability
You can start with a small cluster. When you need to support more users or need more nodes to bring to bear on a problem, you can simply add them to the existing cluster. Although there may be some small amount of degradation in communications efficiency as more nodes are added to the cluster, scaling is essentially linear. If you double the number of nodes in your cluster, you basically double the performance.

Another aspect of scalability involves interconnecting clusters: not only do system interconnects connect the nodes within a Beowulf cluster but they can also connect small clusters together to make larger ones.

Theoretically, the smallest Beowulf cluster you can build would comprise two nodes. And, theoretically, the largest you can build is limited only by the system area network's switching scheme. Compaq has implemented Beowulf clusters as small as 8 nodes and as large as over 1,450 nodes.
[Reference: http://www.cs.sandia.gov/cplant/]

Other interconnect considerations
You might want to connect your cluster to the Internet or, on the other hand, maintain absolute security with an isolated system. Physical size of the cluster may also affect the system interconnect.

System Considerations

Design/development Debug
There are a number of critical tools necessary for the implementation of a successful HPTC cluster solution. The first is a compiler which can take advantage of the architectural features of the processor. Next, a debugger such as Etnus TotalView allows the developer to graphically display the code and assists in finding the problem areas or sections of code to be further tuned for performance. A profiler is also necessary to assist in finding the performance bottlenecks in the overall system including the system interconnect.

Job Control
Once an application has been developed or ported to a Beowulf cluster, the application must be started and run on a portion or all of the cluster. You must understand for your particular needs the requirements for system partitioning, how jobs are started and run, and how a queue of jobs can be setup to run automatically.

Checkpoint Restart
Many applications running on even very large HPTC clusters will require many hours, days, or weeks of execution time to run to completion. A failure in one part of the system could corrupt a job execution run, forcing a restart. The solution is to periodically "checkpoint" the current state, writing the intermediate data calculations available at the end of the interval to a disk subsystem. This usually takes a small amount of time to write out the data with the compute functions temporarily paused, the time dependent on the storage architecture. If there is a system failure of one of the computing components, then the failing component can be taken out of the cluster and the job restarted with the data available from the previous period's

checkpoint save. If there are no failures, then the system did take a bit longer to perform the periodic checkpointing function.

Performance Monitoring
Even if a considerable amount of time is spent during the debug phase to tune the application for best performance, a performance monitoring function is still necessary to watch the cluster performance over time. With potentially multiple job streams running concurrently on the system, each taking differing amounts of CPU or memory, there may be situations where the applications are not running at the expected efficiency. The performance-monitoring tool can assist in detecting these situations.

Partitioning
Once you've created a cluster, do you have to dedicate all cluster nodes to solving one problem for one user? The answer is no. With the proper software to manage it, your cluster can be partitioned dynamically and broken up into some number of smaller "virtual" Beowulfs supporting some larger number of users or compute tasks.

Middleware: the message passing protocols for coordination and data dissemination

Coordination and communication between the processing nodes so they can truly work together is an obvious key requirement of parallel-processing clusters. In order to accommodate this coordination, developers have created software to carry out the coordination and hardware to send and receive the coordinating messages.

Messaging architectures such as MPI or Message Passing Interface, and PVM or Parallel Virtual Machine, allow the programmer to ensure that control and data messages take place as needed during operation. Several approaches to message passing are discussed below.

OpenMP
The OpenMP application program interface (API) supports multi-platform shared-memory programming on UNIX platforms (and Microsoft Windows NT architectures). Jointly defined by a group of major computer hardware and software vendors, OpenMP is a portable, scalable model that gives shared-memory programmers a simple and flexible interface for developing parallel applications for platforms ranging from the desktop to the supercomputer. The OpenMP compiler will automatically set up the application to run on nodes within the multiprocessor system targeted.

An increasing number of parallel machines make use of the shared memory architecture. In this type of platform, each processor has access to a global memory store and processors communicate with one another by accessing the shared memory. This communication paradigm simplifies programming multiprocessor machines by removing the requirement for explicit communications.

MPI
MPI is a message-passing application programmer interface with protocol and semantic specifications for how its features must behave in any implementation (such as a message buffering and message delivery progress requirement). MPI includes point-to-point message passing and collective (global) operations. These are all scoped to a user-specified group of processes.

In addition, MPI supplies abstractions for processes at two levels. At the first level, processes are named according to the rank of the group in which the communication is being performed. At the second level, virtual topologies allow for graph or Cartesian naming of processes that help relate in a convenient, efficient way the application semantics to the message passing semantics. Communicators, which house groups and communication context (scoping) information, provide an important measure of safety that is necessary and useful for building up library-oriented parallel code.

MPI provides a substantial set of libraries for the writing, debugging, and performance testing of distributed programs. [Reference: http://www-unix.mcs.anl.gov/mpi/index.html]

Our system currently offers MPICH, a portable implementation of the MPI standard developed cooperatively by the University of Mississippi and the Argonne National Laboratory**.** MPICH provides for simpler portability while still maintaining high performance.
[Reference: http://www-unix.mcs.anl.gov/mpi/mpich]

PVM
PVM, or Parallel Virtual Machine, started out as a project at the Oak Ridge National Laboratory and was developed further at the University of Tennessee. PVM is a complete distributed computing system, allowing programs to span several machines across a network. PVM utilizes a Message Passing model which allows developers to distribute programs across a variety of machine architectures and across several data formats. PVM essentially collects the network's workstations into a single virtual machine. XPVM, the graphical interface for PVM is also now available.

For the truly adventurous, UNIX@CSI now supports JPVM, a Java implementation of the Parallel Virtual Machine Architecture. JPVM combines the distributed power of the simplistic PVM command structure with the strengths of Java's elegant Object Oriented Programming techniques. For those already familiar with Java, JPVM can provide a quick method of developing distributed programs.
[Reference: http://unlser1.unl.csi.cuny.edu/%7Erigbyc/parallel.html]

LINDA
LINDA is a parallel-processing model developed originally by a research team at Yale University, and now offered commercially by Scientific Computing Associates. The LINDA model uses a distributed data structure topology (called “tuple-space” in LINDAspeak). This topology can be implemented with either Fortran or C. The LINDA “language” consists of a small library of functions that allows the user to control access to the distributed memory system as well as to start and stop processes. Using these functions, pre-existing sequential programs can be easily transformed into parallel programs. UNIX@CSI currently holds a twenty-station license for LINDA.
[Reference: http://unlser1.unl.csi.cuny.edu/%7Erigbyc/parallel.html]

SHMEM (Cray function)
The SHMEM library provides a shared-memory model for programming parallel computer systems. It allows a node to read and write data stored in other nodes' memories. In addition to the basic set of primitives for accessing remote memory, the SHMEM library provides several collective parallel operations.

Cray Research originally created the SHMEM interface for use on the T3D supercomputer. The HPVM SHMEM library provides a compatible interface to the Cray SHMEM library for a subset of the operations. The library includes remote get, put, swap and synchronize routines. It also includes several collective operations including reduction, broadcast, and barrier.

Since SHMEM was developed for the T3D, one of the most common uses for HPVM SHMEM is the porting of applications written for SHMEM on the T3D and T3E. However, any application with irregular communication patterns may benefit from the get/put interface. Further, the HPVM SHMEM library may be a good target for the conversion of cache-coherent shared memory programs to a message-passing version that can run efficiently on a cluster of commodity computers and networks.
[Reference: http://www-csag.ucsd.edu/projects/hpvm/doc/hpvmdoc_82.html]

A parallelized application or applications to run on the system

You need to consider application design in relation to the computing topology. Here are some examples of applications that can take advantage of the parallel-computing environment.

➔ Climate/Weather/Ocean Modeling and Simulation
➔ Computational Chemistry and Materials Science
➔ Computational Electromagnetics and Acoustics
➔ Computational Electronics and Nanoelectronics

- ➔ Computational Fluid Dynamics
- ➔ Computational Structural Mechanics
- ➔ Environmental Quality Modeling and Simulation
- ➔ Integrated Modeling and Test Environments
- ➔ Military Forces Modeling and Simulation (FMS)
- ➔ Pharmacokinetics
- ➔ Reservoir Modeling and Simulation
- ➔ Seismic Processing and Interpretation
- ➔ Signal/Image Processing

3. Achieving performance goals

What performance do you need in your Beowulf cluster and how do you achieve it?
There are several system factors that affect performance including

- ➔ Number of processors
- ➔ CPU speed
- ➔ Memory size
- ➔ Cache size
- ➔ Storage architecture and total storage capacity
- ➔ Interconnect bandwidth and latency

In general, Compaq strongly encourages benchmarking the application or a key piece of the application on the various platforms, processors and architectures under considerations. Compaq provides *Solution Centers* and engineering labs for consultation, benchmarking, and characterization.
[Reference: www.compaq.com/solutions/customsystems/hps/index.html]

Number of processors

For the most part, the number of processors depends on how the application can be subdivided into a number of different tasks. Yet, how the application subdivision is not the only metric for deciding how many processors. Overall performance requirements can figure in, too. For instance, you can do a BLAST search by dividing the database into 8 pieces and running it on an 8-node Beowulf. But if you need to finish twice as fast (and money isn't an over-riding consideration), you could run it on 16 nodes.

Also factor in the application source: is it “home-grown” or commercially available? The number of processors will be dependent on interconnect type and bandwidth and how much message passing the application requires. Choice of platform architecture is also a consideration, e.g., SMP versus uniprocessor. Some applications from vendors may require particular system architectures.

CPU speed

Unlike the "sequential world" where processor speed is considered the single most important factor, processor speed in the "parallel world" is just one of several factors that will determine overall system performance and efficiency. While peak performance is a good first-level indication of suitability, it may not be the best way to measure systems in regard to the class of application that they will support. Sustained performance can be a better metric instead. While it is nice to have your system listed on the Top 500 Supercomputers list which ran one very specific piece of benchmarking code, your application’s more general code will not usually be able to take advantage of that level of performance. The system’s sustained performance is usually more critical for most applications.
[Reference: http://www.netlib.org/benchmark/top500/top500.list.html]

Memory size

In a Beowulf cluster**,** the data set may be required to reside in memory along with all the messaging traffic destined for the other processors. If this is the case, memory needs to be large enough to contain all those necessary elements as well as the operating system and it’s necessary components. It is best to keep the bulk of the data as close to the processor as possible and larger memory sizes increase the likelihood. As with most aspects of the cluster, requisite memory size is highly dependent on the class of application and its relation to cache.

Cache size

Cache size is highly dependent on the class of application you are running. The cache is a small piece of very fast memory sitting “between” the processor and memory. The memory controller in the processor will first try to find a requested piece of memory in the cache because this is the fastest access memory. If it is located in the cache, the memory operation completes quickly and processing can continue. If not, then a longer access to main memory is required and the application may be held up waiting for the memory operation to complete.

To make more efficient use of the HPTC system, it is important to tune the application to utilize the cache as much as possible. Trying to do all operations in cache requires a lot of tuning of the application. There are some applications, however, with fundamental data sets that are too large to fit into cache so cache size may be irrelevant.

Storage architecture/total storage capacity

The appropriate storage architecture is critical element in cluster performance. Storage capacity includes hard disk storage as well as offline storage. The storage architecture is application-dependent not only for size but also for disk subsystem performance. A typical requirement is writing out all memory to disk as quickly as possible for checkpoint functions (described earlier). In this case, storage I/O bandwidth is very important.

Interconnect bandwidth and latency

When analyzing your specific application, determine if it is CPU limited (compute bound) or interconnect limited (I/O bound). The requirements of your Beowulf may be quite different depending upon your needs. For example, a compute-bound problem may work well with a few, very fast CPUs and a low-speed, higher-latency network. An I/O bound problem may work better with slower CPUs and a faster interconnect. [Reference: http://www.netlib.org/benchmark/top500/top500.list.html]

4. Performance capabilities: platforms and interconnects

Compaq platforms

Performance numbers for the *Compaq* systems shown below were obtained from measurements using individual servers running *Compaq Tru64 UNIX* or Windows NT. Performance measurements running Linux in cluster environments are not available at this time. Mathematical analysis suggests, however, that while clustering compute nodes will not provide linear scaling, scaling of close to linear performance may be expected.

Platform	Processor	Peak FLOPS (MFLOPS)	SPECfp95 [3] (#proc, result)	SPECint95 [3] (#proc, result)	McCalpin (Copy) Memory Bandwidth [1] (#proc, MB/s)
AlphaServer					
DS10/DS10L	EV6, 466 MHz	932	1P, 48	1P, 25	881
XP1000	EV6, 500 MHz	1,000	1P, 66	1P, 38	900
DS20	EV6, 500 MHz	1,000	1P, 59	1P, 28	1077
ES40	EV6, 500 MHz EV67, 667 MHz	1,000 1,334	1P, 58 1P, 83 [4]	1P, 27 1P, 38 [4]	1004 2,550 [2]
Intel-based *ProLiant 5000*	PentiumPro, 200 MHz				126

[1] http://www.cs.virginia.edu/stream/ref.html; [2] Results not yet posted on McCalpin results website;
[3] http://www.spec.org/osg/cpu95/results/cpu95.html; [4] Estimated results, not yet posted on SPEC website

Interconnects
Performance data for the interconnects (using their theoretical bandwidth and latencies) listed below have been provided by the manufacturers of the interconnects. As with the platform data, more performance analysis is required in a clustered Linux environment to determine the effects of scaling or application type.

	Bandwidth (MB/s)		**Latency (MPI, one way, in ì s)**
	Sustained	**Theoretical peak**	
System Area Networks			
Myrinet	140	250	18
Servernet2 [5]	173	480	13
Quadrics	208	680	5
Local Area Networks			
Fast Ethernet	~7	12	~40
Gigabit Ethernet	~50-100	120	~32

[5] Available Q2 2000.

5. Conclusion

The choices one makes in specifying a Beowulf cluster are highly dependent on the requirements of the application to be run. With an understanding of the analytical process required to specify a Beowulf cluster, you will be able to accelerate the speed at which you can deploy a solution.

To further speed your successful Beowulf deployment, Compaq has a range of *Alpha* or Intel systems characterized and configured for Beowulf clustering with support for major Linux distributions. Compaq has assisted customers on a number of different Beowulf configurations including some of the world’s largest at computing facilities such as

➔ Sandia National Labs
➔ Los Alamos National Labs
➔ Argonne National Lab
➔ Jefferson Lab
➔ Idaho National Energy and Environmental Lab
➔ National Research Council of Canada
➔ Southampton High Performance Computing Centre
➔ Clarkson University
➔ Geneva (Switzerland) Observatory
➔ Eglin Air Force Base

In addition to start-of-the-art hardware we provide architectural consulting; configuration design; pre-testing of solutions; and factory integration of various Linux operating systems, selected development tools, and your specified applications. We also support a *High Performance Solutions Center* and engineering labs for consultation, benchmarking, characterization, and prototype development.

For more information on Compaq’s high-performance capabilities, visit www.compaq.com/hpc or www.compaq.com/solutions/customsystems/hps/index.html

Considerations in Specifying Beowulf Clusters
White Paper prepared by CustomSystems High Performance Computing Segment

First Edition (February 2000)

11QE-0200A-USEN Rel#.000/2000 00 00 00

Rev 2. 22-Feb-00

Rev 3. 23-Feb-00

COMPAQ