



BEOWULF COMPUTER CLUSTERS

Andrew Duggan

TESSELLA SUPPORT SERVICES PLC

Issue V1.R2.M1

October 2001



BEOWULF COMPUTER CLUSTERS

More and more organizations are turning to Beowulf clusters to provide solutions to their highly computationally intensive problems. Why? The major reason is that powerful Beowulf clusters are available at only a fraction of the cost of the high performance computer systems available from commercial vendors. Beowulf clusters are being exploited in many application areas such as powerful web and database servers and high quality rendering. Simulations are being run of high energy and plasma physics, atmospheric chemistry and high temperature superconductors. Complex calculations on N-body problems, astronomical data, computational fluid dynamics and light scattering are also being achieved. This article seeks to understand what Beowulf computers are and how they can solve your computation problems.

Introduction: Who was Beowulf and What is a Beowulf Cluster?

Beowulf was the eponymous hero of a poem written around 1000 AD which tells the tale of a Scandinavian nobleman who fought many courageous battles. In his youth he defeated the monster Grendel, which had been ravaging the court of the King of Denmark. When Grendel's mother returned to avenge her son's death, Beowulf killed her too. After ruling his land for 50 peaceful years, Beowulf died whilst defeating a dragon with the help of Wiglaf who then succeeded Beowulf to the throne of Geatland.

Rather less dramatically, in the summer of 1994, Thomas Sterling and Don Becker, working under a NASA contract, built a parallel processing computer¹ consisting of 16 486 DX4 processors. The idea was to show that 'off the shelf' computers could be used to tackle serious computational problems (this machine was used to model the fragmentation of comet Shoemaker-Levy 9 during its close encounter with Jupiter). Sterling and Becker named their machine Beowulf and the name has since been used to describe the genre of cluster computers built from commodity or 'off the shelf' components. These components are necessarily relatively low cost because of their high sales volume. The computer used a UNIX-like operating system called Linux which runs on, amongst others, the x86 series of processors such as Intel's Pentium III and AMD's Athlon, which are found in many desktop PCs. It was developed by individuals across the world and is available free of charge.

¹ Parallel processing refers to the concept of speeding up the execution of a program by dividing the program into multiple fragments that can execute simultaneously, each on its own processor.

Since the early days the Beowulf concept has been developed and scaled almost to absurdity. The second fastest computer in the world, the ASCI Red TFLOPS supercomputer, made by Intel and the American DoE, which hit the headlines when it was first tested in December 1996, was a Beowulf type machine consisting of 9,216 Pentium Pro processors. It was the first computer to reach the ‘Holy Grail’ of 1 Terra-flop, or, 1,000,000,000,000 (10^{12}) floating point operations a second. While few people would have the time, inclination or network cables to link 9000 computers it demonstrates the computing power available from low cost, commodity components.

The performance/price ratio available from Beowulf machines reached an all time high at the Oak Ridge National Laboratory in America. Scientists wished to use statistical modelling to study the ecology of the United States, but were denied the modest funding needed to produce a Beowulf machine. Undeterred and having already invested a lot of time in the project, the scientists commandeered ‘old’ machines destined for the scrap heap as clerical staff in the laboratory had their machines upgraded. Soon the group had amassed over 100 mostly 486 machines at no cost which they used to produce a Beowulf machine. This ugly duckling machine of cast out components has since produced the most detailed climatic map of the USA to date. The average Beowulf machine today, if there is such a thing, is most likely to be made of between 8 and 32 PIII/Athlon processors for a cost of between 5 and 50 thousand pounds.

The take home message is: “if you want to run serious computations then parallel processing on a Beowulf class machine consisting of off the shelf components can provide a very serious, low cost solution.”

What is Parallel Computation?

We have said that Beowulf machines use parallel processing methods. Parallel processing is not restricted to Beowulf machines. All of today’s fastest computers employ parallel processing technology in one form or another. What exactly does this mean, and would it work for your computing problem? Parallel computing is not always appropriate and for some applications there is little benefit to be made from switching to a parallel system. The method of parallel computing and some possible problems are discussed below.

The analogy of a supermarket is often used in describing parallel computation. Consider a large supermarket with a line of tills at the front. Each till is a CPU and each customer a computer program. The items in the customers trolley are the instructions the CPU must process to run the program.

Imagine the case where only one till is open, each customer has to queue and be served one at a time. This is a single-tasking operating system model, such as MS-DOS. Imagine now that the cashier scans one item each, in turn, of several customers. Everybody moves through the queue, though more slowly than if they were the only customer. This is a multi-tasking operating system example such as single CPU UNIX or Windows NT. In the third scenario, several tills are opened. Each customer is processed by a separate cash register and the overall line moves much quicker. This is called SMP - Symmetric Multi Processing. Note in this case that although there are extra cash registers open you will not get through the line any quicker than the previous example if you are the only customer at the shop. The computer example here is UNIX or NT with multiple processors.

Better use of the system is developed through what is known as 'threading'. If you have a large amount of shopping you might be able to logically break up your shopping between multiple tills, say frozen goods at one till, fruit and vegetables at another and tins at the third. When the cashiers need to get sub-totals to work out the bill they can exchange information quickly by looking and talking to each other. In theory with three tills like this you should get served three times faster, but there are pitfalls. If you only have a small basket, sorting out the different items and adding up the totals at the end could take more time than would be gained by the extra tills. If you have far more frozen food than not then two of the tills will be sitting idle waiting for the third. This scenario models a multi-threaded program running on a multi CPU motherboard with either NT or UNIX.

In order to serve customers more quickly the store adds more tills. There is no space at the front so the new tills are put in at the back of the shop. Now the cashiers have to phone in their subtotals to gather the total bill and it takes longer to take part of the order to the back of the shop. Even with this extra overhead though, if communication between the cashiers is minimised and you have a really big order then you can get a huge increase in speed by using all the cashiers at the front and the back of the shop. This is a classical parallel situation utilised in Beowulf machines, where the CPUs can be in different machines and communicate via messages.

For parallel computing to be successful, your application must have enough concurrency to make good use of multiple processors. This is not just a matter of identifying portions of the program that can execute independently. As we have seen, co-ordinating processes over different CPUs, especially in different machines, has an inherent time overhead, sometimes called network latency. If this is not managed carefully applications may run slower on a parallel system. For example, a section of code may take four seconds to run on a single CPU. It could run in one second on

four parallel CPUs, but there would be no increase in speed if it took three seconds or more for these machines to co-ordinate their actions. Porting code to take advantage of a parallel system is not a trivial task but if handled correctly the benefits of increased speed can be enormous.

How does Beowulf work?

At the simplest level, Beowulf is a collection of extensions to the standard Linux kernel. Thus, at its heart a Beowulf system is only slightly different from a network of UNIX machines that you may already have. A Network Of Workstations (NOW) using system wide logins, such as NIS, with users able to execute remote commands on different machines could in some sense be called a Beowulf system. Beowulf uses several distributed application programming environments to pass messages between the different computation nodes. The most commonly used are PVM (Parallel Virtual Machine) and MPI (Message Passing Interface) which take the form of library function calls that must be added into the code. These environments, especially PVM, are designed to be used on a heterogeneous system of workstations with apparently little in common and could be used in NOWs. There are however several subtle, yet significant differences between these systems and Beowulf systems.

In the usual Beowulf scheme, as is common in NOWs, every node is responsible for running its own copy of the operating system kernel, with nodes generally sovereign and autonomous at this level. However in order to present a more uniform image to both the applications and the users, the Beowulf extensions allow a loose ensemble of nodes to participate in a more coordinated way through a number of global namespaces. For example, normal UNIX processes 'belong' to the kernel running them and have a unique identifier within that context. In the Beowulf system it is possible to have a process ID which is unique across an entire cluster.

A Beowulf cluster will typically have only one connection with the outside world, and will communicate via its own private intranet. This means that there is only a single point for users to log into. In the initial Beowulf machine, 100 Mbit Fast Ethernet was considered too expensive for the system intranet, so a technique was devised, named channel bonding, which joined multiple low cost networks into a single logical network with a higher bandwidth. Thus in the original Beowulf machine each node had two network cards. The current low cost of switched 100Mbit fast Ethernet and 1000Mbit Gigabit Ethernet has reduced the need for this channel bonding solution, but it can still be used for higher network traffic applications.

NOW systems have historically only been successfully used in this manner at night

when there is little other demand on the individual machines and the local network. A Beowulf machine is perceived to be a single user machine, running only one user application at a time. This gives uninterrupted processing time, spread evenly across all the nodes, to a computationally intensive job.

Acquiring a Beowulf System.

So you have decided on a Beowulf system. What exactly do you need to buy? Unfortunately building a cluster is not a simple undertaking. The components making up the cluster should be tuned for the best performance on its primary applications. Decisions have to be made about the disk size, amount of memory, number of CPUs per node, the number of nodes and how they will be connected. In most cases this becomes a fine balancing act within the constraints of the project hardware budget. Even before making a decision about hard disk size there is a choice between two basic system types.

In the first type, a server node serves the whole file system to all the slave nodes which are diskless. The main advantage of this configuration is that new nodes are cheaper and easier to install into the system. Since the slave nodes do not contain any information about themselves administration is easy as only the configuration files on the server have to be altered. You only install the operating system or any other software once, on the server node. The disadvantages are the increased network traffic from NFS and the more complex initial setup. The diskless clients also need to be started from a boot floppy.

The other extreme is to have everything stored on each client. With this configuration each node needs its own copy of the operating system and all the software used. The advantage of this setup is that network traffic is solely due to user applications and the system can be split into component machines if necessary. The disadvantage of this system is the very complicated administration. Most systems lie somewhere in the middle of these extremes, having separate disks, containing a basic operating system with local swap and scratch space, whilst sharing common system and user files.

Choosing the right amount of memory is also crucial and can determine whether single or dual processor machines are used. If there isn't enough memory to hold the jobs you are going to run then you will lose performance due to extensive swapping. Every page of information which has to be read to and from swap space loses valuable execution time. Node based swap space does reduce this problem to some extent.

Finally processor speed and network speed need to be considered. For example, an

application may work better on a cluster with more/faster processors but slower network, or fewer processors with a higher bandwidth network may give a better throughput. If the speed of a loop is CPU rather than I/O limited it might be a good candidate for parallelisation. Remember though if you take a 10 minute loop and break it into 20 parts and data transfers require several seconds per part, then things are going to get tight.

Although it has been stated that there is no such thing as an average Beowulf machine, for the purposes of this discussion, a complete shopping list for a four node cluster, containing eight CPUs connected by a switched fast Ethernet is shown over the page. The master node is the only machine with a monitor. The other nodes contain graphics cards purely as an aid to the setup of the system, with the monitor cable swapped as necessary. Switch boxes can also be used. The three slave nodes have a 20 GB hard disk to contain kernel files and node local scratch space. The master node contains extra storage space for cluster wide access e.g. user files and a second network card which provides the only connection to the outside world. The prices were obtained at the time of going to press from a typical hardware supplier for reasonable quality components. However, IT hardware prices change rapidly (often downwards) so this is a rough guide only.

These components would produce a Beowulf cluster best suited to applications limited by CPU power. The dual 933 MHz processor setup leads to smaller amounts of memory per CPU, which could slow down applications due to swapping between physical and virtual memory. Similarly the switched, fast Ethernet system could produce a bottleneck for more network hungry applications. The computational capacity of the whole system can be simply increased by adding more and more nodes.

Hardware Item	Master Node	3 Slave Nodes	Total Cost / £
Midi Tower Case	1	3	168
Network Cable	2	3	15
3.5" 1.44 Mb Drive	1	3	32
52x EIDE CD ROM	1	N/A	23
8 Mb AGP Graphics Card	1	3	100
20 GB 8.7ms UDMA 100 HDD	N/A	3	216
30 GB 8.7ms UDMA 100 HDD	1	N/A	87
256 Mb SDRAM	2	6	472
19" Monitor	1	N/A	275
Dual FCPGA Motherboard	1	3	480
100 Mbit Fast Ethernet Card	2	3	150
Intel 933EB MHz Pentium III	2	6	1312
Mouse	1	N/A	12
Keyboard	1	N/A	15
100 Mbit Fast Ethernet Switch	N/A	N/A	450
Total Cost (ex. VAT)			£ 3807

Conclusion.

Improvements in the hardware of desktop PCs are continuing constantly and the cost of complex network hardware is falling as it becomes more widely utilised. These two factors have led to the situation where specialised calculations can be performed on a cluster of relatively inexpensive machines available from high street suppliers. Properly co-ordinated, these machines challenge the performance of high end computers for a much lower financial outlay.

Tessella Support Services plc
Creating Software for Science and Engineering

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software
Data Capture
Simulation Software
Advanced Graphics
Systems Support
Database Applications

Other Technical Supplements available include:

- | | |
|---|--|
| <input type="checkbox"/> Archiving of Electronic Info | <input type="checkbox"/> Object Oriented Programming |
| <input type="checkbox"/> Active Server Pages | <input type="checkbox"/> Pocket PC |
| <input type="checkbox"/> Automated GUI Testing | <input type="checkbox"/> Portable GUI Development |
| <input type="checkbox"/> Bayesian Statistics | <input type="checkbox"/> Printer Technology Guide |
| <input type="checkbox"/> Beowulf Clusters | <input type="checkbox"/> Real Time Systems |
| <input type="checkbox"/> C++ | <input type="checkbox"/> Regression Testing |
| <input type="checkbox"/> Client-Server Technology | <input type="checkbox"/> Security and the Internet |
| <input type="checkbox"/> COM | <input type="checkbox"/> Simulation |
| <input type="checkbox"/> Computational Fluid Dynamics | <input type="checkbox"/> Soft Computing |
| <input type="checkbox"/> Computer Image Processing | <input type="checkbox"/> Software Design Methodologies |
| <input type="checkbox"/> Decision Support Systems | <input type="checkbox"/> Software Development Cycle |
| <input type="checkbox"/> Electronic Data Capture | <input type="checkbox"/> Software Documentation |
| <input type="checkbox"/> Electronic Lab Notebooks | <input type="checkbox"/> Software Portability |
| <input type="checkbox"/> Excel | <input type="checkbox"/> Software Re-engineering |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> Software Specification |
| <input type="checkbox"/> Federal Drug Administration | <input type="checkbox"/> SQL |
| <input type="checkbox"/> FORTRAN 90 | <input type="checkbox"/> UNIX Inter-Process Comms |
| <input type="checkbox"/> Grid Computing | <input type="checkbox"/> UNIX Systems Performance |
| <input type="checkbox"/> High Throughput Screening | <input type="checkbox"/> UNIX Workstations |
| <input type="checkbox"/> Instrumentation | <input type="checkbox"/> Visual Basic 6 |
| <input type="checkbox"/> Integrated Lab Systems | <input type="checkbox"/> WAP |
| <input type="checkbox"/> J2EE | <input type="checkbox"/> Web Services |
| <input type="checkbox"/> Java | <input type="checkbox"/> Windows 2000 Services |
| <input type="checkbox"/> Lims | <input type="checkbox"/> XML |
| <input type="checkbox"/> Linux | <input type="checkbox"/> X Windows |
| <input type="checkbox"/> Microsoft Net | |



INVESTOR IN PEOPLE

Tessella Support Services plc

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: info@tessella.com Web Address: <http://www.tessella.com>