6th IDEA Workshop, Rutherglen
Victoria, Australia, 27-29 January, 1999.

23

Beowulf – A New Hope for Parallel Computing?

K. A. Hawick, D. A. Grove and F. A. Vaughan

Advanced Computational Systems Cooperative Research Centre
Department of Computer Science, University of Adelaide
SA 5005, Australia
Tel +61 08 8303 4519, Fax +61 08 8303 4366

Email: khawick@cs.adelaide.edu.au

Abstract

The Beowulf model for clusters of commodity computers[15, 17, 18] has become very popular over the last year, particularly amongst university research groups and other organisations less able to justify large procurements. The Beowulf concept is usually applied[16] to clusters of Personal Computers running Linux, but other platforms and operating systems can also be considered as providing similar functionality [13, 14]. We discuss the criteria for what constitutes a Beowulf cluster, describe our initial experiments with prototype clusters of PCs and iMacs, and review possibilities for construction of a large well-networked system. We have experimented with the configuration and set up of a heterogeneous cluster of PCs on various network fabrics including conventional and fast Ethernet. Design of a scalable network structure is particularly important for a scalable system. We discuss some options for using switch-based infrastructure built from off the shelf switch components. A range of economic as well as technical issues complicate the design of a suitable Beowulf system for a given set of applications. We consider two major requirements scenarios: running a large number of serial applications; and running a small number of highly parallel applications. Of particular interest to us is the situation where individual nodes must be heavily configured with disk and memory. Parallel computing research issues seem particularly relevant to Beowulf clusters, including those of fault tolerance, scheduling and dynamic problem decomposition. We describe our ongoing research activities in these areas and speculate over the impact Beowulf style clusters will have on parallel computing.

Keywords: Beowulf; cluster; scalable; parallel computing; heterogeneous.

1 Introduction

Beowulf is a project to produce parallel Linux clusters from off-the-shelf hardware and freely available software. Conceived in 1994 at the Goddard Space Flight Center[2], there are now dozens of Beowulf-class systems in use in Government and at Universities worldwide. Many of these organisations have joined to form a Beowulf consortium[19] who actively share information and software for Beowulf systems. Some members include: Caltech, Los Alamos National Laboratory, Oak Ridge National Laboratory, Sandia National Laboratory, Duke, Oregon, Clemson and Washington Universities, The US National Institute of Health (NIH), as well as DESY in Germany, Kasetsart University in Thailand. NAS, Goddard Space Flight Center, Ames and various NASA sites and divisions have built major Beowulf systems. Other small systems have also been built at the University of Southern Queensland and the University of Adelaide[22] amongst many other sites.

The original Beowulf parallel workstation prototyped by NASA combined sixteen 486DX PC's with dual Ethernet networks, 0.5 GByte of main memory, and 20 GBytes of storage, and providing up to eight times the disk I/O bandwidth of conventional workstations. Since the Beowulf design uses commodity hardware components and freely available systems software, NASA's project has demonstrated how the price/performance ratio of this route is attractive for many academic and research organisations.

This paper outlines some of the technical and operational details surrounding design of a Beowulf system. It also presents a discussion of some of the open research issues that surround parallel computing systems in general and

particularly how they relate to Beowulf like systems, as well as describing how a Beowulf prototype system can further research objectives in parallel computing. We describe some of the possible processor hardware configurations and the economic issues in section 2 and focus on network architecture issues in section 3. Since our immediate target system has to run the Gaussian chemistry package we list some performance figures on various platforms in section 4. We briefly review some of the parallel computing research issues that Beowulf clusters present in section 5 and summarise the outlook for parallel Beowulf-style computing in section 6.

2 Economics and Configurations

One of the most difficult tasks in designing and commissioning a Beowulf cluster is tracking the cost/performance benefits from the multitude of different possible configuration options. In this section we discuss the different component cost options.

Broadly the design choices in order of importance for performance are:

1. Processor/Platform (eg PC, iMac, Alpha, O2,...)

2. Network infrastructure (Ethernet, Fast Ethernet, Myrinet, SCI,...)

3. Disk configuration (Diskless, EIDE or SCSI interface...)

4. Operating system (Linux or Solaris or other...)

For the purposes of this discussion we assume the operating system software is of negligible cost *per se*, although there are of course maintainance issues involving the expertise of local personnel that might also be important. It is of course possible to complicate this further by admitting the possibility of a heterogeneous system.

The main options we have considered are Intel processors or iMac/G3 processors, with each node having a modest disk configuration for local virtual memory, and a relatively flat structured network technology such as switched fast Ethernet. Tables 1 and 2 below show the current (January 1999) tax exclusive prices and educational discount prices available in Australia for a PC-based configuration and an iMac based configuration.

Component	Cost (Aus$)	Comments
chassis/case	135	mini tower
motherboard	529	Dual processor
PII processor	436	350 MHz
PII processor	702	400 MHz
PII processor	1121	450 MHz
Network Card(s)	106	(Intel 10/100B)
Memory	1060	256MB SDRAM (2x128)
Disk	873	9GB SCSI
Disk	233	4GB EIDE
CD-ROM	94	32x IDE
Floppy	31	IDE
Graphics Card	53	4MB S3 AGP

Table 1: Typical PC Hardware Components for a Beowulf System Node

The typical cost of a usable compute node with dual processors and EIDE disk will range between $3k and $4k. Additional costs for a server node will include keyboard ($61), mouse ($20), and monitor which can range from $250 to $2000 depending upon size and quality.

Other minor sundry items can be usefully added to a server such as additional disk controllers, ports and tape drives. These will vary considerably with desired configuration, but a SCSI tape drive and subsystem will add between $2k and $3k to the cost of the server.

In general Apple equipment is not as easily available in custom configurations and is typically bought pre-assembled. This is reflected in the difficulties in obtaining component price information. In general a mini-tower configuration will cost between $4k and $6k for a 400MHz node configuration with a single G3 processor.

6th IDEA Workshop, Rutherglen
Victoria, Australia, 27-29 January, 1999.

25

Mac Configuration	Approx Cost (Aus$)	Comments
G3 Processor	3495	300 MHz, 64MB, 6GB disk
G3 Processor	4495	350 MHz, 64MB, 6GB disk
G3 Processor	5495	350 MHz, 128MB, 12GB disk
G3 Processor	6695	400 MHz, 128MB, 9GB disk
Memory	1018	256MB G3 RAM

Table 2: Typical Apple Mac Hardware Components for a Beowulf System Node

Other considerations that need to be factored into the overall cost of a system include the housing infrastructure. It is common in research groups to choose either to utilise existing legacy rack structure or more commonly to use commodity shelving and provide each node with its own case and power supply. This route is preferred since it allows the nodes to be reused, as desktops for example, at a later date and also minimises the custom infrastructure required.

For our target system we have primarily considered running Linux or Solaris operating system software on each node. Both of these are available at cost of media only. We have run Linux on Intel/PII and PowerPC/G3 processors but Solaris only on Intel platforms. The choice between Linux and Solaris does not seem to present any particular advantages either way for Intel. Since we believe it is likely we may wish to experiment with a heterogeneous hardware cluster it is marginally preferable to have homogeneity of operating systems and therefore Linux is our chosen system. Current research into kernel improvements for Beowulf like systems are more common for Linux than Solaris.

We have experimented with Gnu compilers for Fortran as well as proprietary compilers from Absoft and from the Portland group. These represent the only software costs for our planned system. Other utilities and management software are already available as part of the commonly bundled Linux releases or as public domain software on the Web. There is scope for an improved set of system management utilities however.

We are still considering the maintainance (personnel effort) and life cycle (upgrade costs) for our planned cluster. These do not appear to be appreciably different for those of other compute platforms, but do appear to offer more flexibility over buying a monolithic system as one inflexible purchase order.

3 Network Architectures

There are a number of technologies available at the time of writing for linking together a set of compute nodes in a Beowulf like cluster. These include conventional Ethernet running at 10 Mbit/s, Fast Ethernet running at 100Mbit/s or less common faster technologies such as Myrinet or SCI, and other technologies more usually associated with supercomputers such as HiPPI.

Conventional or Fast Ethernet can be run on a hub or analogue repeater system. Nodes may have flat structured access either through a cascaded set of repeaters, or through a hierarchy of switches. Repeaters share the 10Mbit/s amongst nodes, whereas switches will provide a dedicated high bandwidth backplane that will allow pairs of nodes to communicate independently at the full speed rating. Switches typically have slightly higher latencies than repeaters which is a potential limitation for tightly coupled parallel applications, but is acceptable for a job tasking environment. The effective latency for applications codes which must use the kernel to send messages is typically of order 100us, and in a practical system will dominate switch latency. Switches provide more attractive technical solution but are significantly more expensive with a price ratio of approximately 10.

Myrinet is probably the only credible alternative to Fast Ethernet at the time of writing. It provides 1.28Gbit/s (full duplex bandwidth), but at a typical cost of $2k-$3k per network card and $9375 for an 8-port switch, is somewhat overpriced. It is also unclear how to inter link switches at similar bandwidths. In technological terms, SCI provides the best performance at a rate of 4Gbit/s, but has no Linux driver support at the time of writing and is approximately twice as expensive as Myrinet hardware at present. Scalability of the network infrastructure becomes more complex for large clusters where maintaining the full rating of the backplane becomes technologically more difficult (and necessarily more expensive).

The choice of network architecture is perhaps the most difficult aspect of a cluster design to change and therefore requires careful consideration. Upgrade routes will typically involve complete replacement of the switching infrastructure and network cards in each node. Commodity units such as PC network cards can at least be recycled in

26

6th IDEA Workshop, Rutherglen
Victoria, Australia, 27-29 January, 1999.

other systems. A number of research groups, including the original Beowulf project group at NASA have investigated the effects of various network combinations, topologies and switching technologies. The original Beowulf project used channel bonding software to make use of two conventional Ethernet card interfaces and double the bandwidth available. This approach yields good performance but necessarily involves special kernel modifications and is perhaps more difficult to maintain than using a single card. Other work [7, 8] has considered using combinations of switch and torus configurations to achieve a better compromise bandwidth than is possible from a flat switched structure.

Generally, commodity technology for relatively small systems up to 24 nodes is widely available from switch manufacturers and is a good solution for Fast Ethernet speeds. For higher numbers of nodes some increase in inter switch bandwidth is required to maintain the full rates between arbitrary pairs of nodes. Commodity products to link switches is becoming available for this.

The balance point for a typical application is a useful concept for judging whether processing nodes are adequately networked. Some years ago the T800 transputer[12] was thought to be well balanced having a processing capacity of the order of 1MFlop/s and a channel bandwidth of circa 1Mbit/s. This ratio is approximately matched by present processing nodes (eg 350-400MHz PowerPC G3 or Intel Pentium II processors), which may perform at between 100 and 200MFlop/s. Processing speed is growing and it is likely that over the next year, the "economic sweet spot" for processing power will exceed ability of Fast Ethernet technology to feed processors with data.

At present Fast Ethernet technology is by far the most economic for a Beowulf system, particularly for applications that are compute bound rather than communications bound. The cost of Myrinet may decrease significantly and it may then be feasible to upgrade all or part of a cluster to use that system. At the time of writing SCI and other technologies such as ATM or HiPPI do not show any signs of becoming economically viable. Gigabit Ethernet may be the next technology worth tracking and is very likely to allow inter switch upgrades. The network technology viability is definitely driven by commodity economics rather than any technological superiority considerations.

4 Performance Review

We have conducted a broad range of performance measures on Intel and Apple processor platforms as well as other Sun, SGI and DEC workstation products. We do not report on these in detail here but summarise some broad conclusions.

We are especially interested in the single node performance for scientific applications as well as likely degradation effects of running tightly coupled parallel applications across multiple nodes. We have employed the well known computational chemistry program, Gaussian[1], as the main target application for our planned cluster. This program is used to calculate chemical configurations as represented by the energy eigenvalues of matrices describing molecular configurations. The code is generally dominated by floating point performance but utilises significant amounts of local file store for intermediate calculations. Although parallel versions of this code are available for a shared memory parallel architecture, our planned cluster usage mode is to run a relatively large number of separate compute jobs running under a queuing management system such as DQS, rather than a single massive computation. This means that at present we are less concerned with parallel communications overhead and that single node compute performance is our primary concern. This should provide good throughput and a cost effective utilisation of the overall machine but poorer turnaround times on individual jobs.

Some preliminary benchmark figures for Gaussian (G94) on various platforms are show in table 3. The figures are for the total execution time for a standard suite of Gaussian test jobs (Numbers 1, 28, 94, 155, 194, 296 and 302 from the Gaussian G94 User manual). These figures therefore report on the general performance of Gaussian using: a dual Pentium II (233MHz) system; a G3/266MHz iMac; a 266 MHz DEC Alpha 600 Series Workstation; and a 20 Processor Power Challenge. The workstations were all dedicated, whereas the Power Challenge was under its typical load and is therefore somewhat slower than would be the case if in dedicated mode. The PC and iMac were using the Linux Operating system and Gnu Fortran compiler, the DEC Alpha used Digital Unix and DEC Fortran compiler; the Power Challenge used IRIX and SGI Fortran.

These preliminary figures do strongly indicate the good performance of the Dual Pentium II system but emphasise what are probably memory and disk overheads for a dual processor system and two jobs. The alpha performance is surprisingly poor; the amount of cpu time used, which is mainly dependent on floating point performance, is significantly longer than the Pentium II. It is likely the disparity between cpu time and wall time consumed for the alpha is indicative of using a remote file system bulk storage, although this remains unresolved. The Power Challenge

6th IDEA Workshop, Rutherglen
Victoria, Australia, 27-29 January, 1999.

27

Test Description	PII/233MHz Time(secs)	Alpha 233 Time(secs)	Power Challenge Time(secs)
G94 (cpu)	633	1105	852.1
G94 (wall)	698	1810	1209
2xG94 (cpu)	709	n/a	-
2xG94 (wall)	790	n/a	-

Table 3: Typical Gaussian test job suite performance on various platforms.

is an interesting comparison point as this is the platform currently heavily used by users including the chemistry group.

5 Open Research Issues and Experiments

Although many Beowulf systems are already used operationally, work is ongoing at NASA and other institutions researching issues such as parallel I/O, distributed file systems, out-of-core algorithms[4] and distributed shared memory software libraries[5] and optimising applications algorithms that will further enhance the performance of Beowulf systems.

The software and design methodology for Beowulf systems are already sufficiently advanced that for some applications a Beowulf can be set up in virtually turnkey mode, with no special set up or changes to the applications code nor special systems software enhancements. However, recent workshops held by NASA[10] have identified a number of research issues in parallel computing that remain outstanding and which would advance the usefulness of Beowulf systems even further. These include:

- distributed I/O, making use of multiple and innovative network interfaces and topologies to alleviate bisection bandwidth, latency and global synchronisation limitations.

- parallel and distributed storage, particularly in a portable form

- systems software reliability and robustness improvements

- instrumentation, monitoring and ensemble management tools to allow large scale Beowulf systems to be operated easily

- new and improved scheduling software to allow heterogeneous Beowulf system to be constructed and reliably run, with the heterogeneity largely transparent to application level users.

- making use of commodity components in interesting configurations, to allow for example dual headed machines or other customisation in configurations to suit particular applications communities.

These issues can be broadly divided into: compute performance; storage performance; I/O and communications; and systems integration, as described in the following sections.

5.1 Compute Performance

Linux is now available for a number of processor architectures, including the Alpha processor as well as the Intel x86 family. PC motherboards are now available with dual or quad processor options, providing the potential for considerable processing power on each compute node.

It is not yet clear how well the memory bus, operating system and associated services can make use of multiple processors on a Beowulf node. A number of projects report building systems with dual nodes. This is still relatively recent and performance scalability is not yet fully characterised. On the basis of past experience with parallel computing architectures it is likely there is a tradeoff point at which it may be more useful to allocate resources on a new separate compute node rather than add nodes to the shared memory and bus structure of an existing node. This needs to be measured for a particular class of target application. Our own preliminary measurements indicate that dual nodes are

28

6th IDEA Workshop, Rutherglen
Victoria, Australia, 27-29 January, 1999.

viable with mainstream shared memory buses able to keep pace with processor speeds. It is not yet clear this is the case for quad processor nodes however.

Many of the Beowulf systems described so far are homogeneous, with processing nodes consisting of the same processor type and clock speed. In practice, it is often desirable to be able to add nodes incrementally to an operational system, and the economics will mandate the price/performance for a particular type and specification of a node will change. It is important to understand how a heterogeneous Beowulf cluster will behave and how its maintainance requirements will change.

We believe it is almost inevitable that a practical system will be built incrementally with new nodes with different clock speeds and memory cache structures. Queuing system software that can manage the metadata to characterise the performance of different nodes and enable static load balancing is therefore desirable. For a dedicated system running entirely under control of a scheduling manager static scheduling is adequate. For system with open user access to the nodes, dynamic load balancing and active load measurement is important[6].

5.2 Distributed Storage Performance

A number of Beowulf clusters have been built using diskless compute nodes. This is a convenient design when the machine is targeted at compute intensive jobs where NFS cross mounted disks and the switch interconnect is adequate. The advantage is that the nodes are very simple to maintain and do not have file systems that can become easily damaged if the nodes are shut down incorrectly. Diskless nodes are easy to setup and design however. It is possible to boot them entirely using floppy disks, or even more conveniently using network adaptor boot roms. The disadvantage of this approach is that the server node is heavily loaded for all file transactions and that if low bandwidth Ethernet is used for the interconnect structure it may become saturated.

There are a variety of hardware and software options for managing storage across the nodes of a cluster. PC hardware options include the cheaper IDE hard disk drives or the more expensive but faster under load SCSI based disk drives[20]. More exotic storage devices such as tape drives or even tape silos with associated control roots are also possible options.

Beowulf nodes can have the full operating system kernel and installation on their own hard disk, which may reduce network congestion for program loading and other system activities. The disk space for user or applications to work in can either be cross mounted from a central server or can be a mix of local and central disk space. It may be attractive to give each node some disk space to work locally in for temporary calculations.

A number of sophisticated distributed file-system research software has been developed for use with clusters of workstations. The use of these may be optimal for certain applications. It is also an interesting possibility to use a Beowulf cluster configured with slow but cheap processors as a RAID system in its own right. The attachment of a dedicated processor allows a number of data management options not possible on dumb disk controllers.

5.3 Parallel I/O and Communications

A critical aspect of a Beowulf cluster that determines its performance is the underlying switch fabric connecting the nodes. A key feature of the original Beowulf demonstrator was the use of channel bonding in software to enable the use of dual Ethernet interface cards[3]. This was important to achieve better performance than a single 10Mbit/s card could provide at a time when 100Mbit/s cards were not cost effective. Performance was analysed in terms of pair-wise exchange of token across the Ethernets of up to 8kBytes, and combinations of single and double channels. Further work was carried out on alternative network topologies for Beowulf clusters [8] in which it was found that in many application cases more complex topologies outperformed the channel bonded scheme. This conclusion was reached on the basis of sensitivity analysis of packet size and traffic density.

The original Beowulf design made use of a private system-wide area network to enhance the network performance of the machine. NASA also built a Beowulf system using groups of metanodes, which were internally connected using Fast Ethernet, but with 1.28Gbit/s Myrinet crossbar used to connect metanodes together[9].

Various experiments were carried out by NASA and the Beowulf consortium into techniques and systems software that would aid in managing a large Beowulf cluster. One was the Ethernet driver software to allow channel bonding and the use of more than one Ethernet interface by each node. Another was the necessary software to allow global process

6th IDEA Workshop, Rutherglen
Victoria, Australia, 27-29 January, 1999.

29

ID's to be used across the cluster. This is very convenient for process management and running parallel programs in SPMD mode.

NASA found that the network performance significantly constrained the performance of the processing nodes of a Beowulf by up to a factor of four [11]. The need to address this deficiency led to their experiments in channel bonding of 10Mbit/s Ethernet and also to investigate the performance of 100Mbit/s – now known as Fast Ethernet.

Latency is less critical for some applications where relatively rare or loosely coupled communications takes place between nodes. For a set of relatively independent computational jobs, a scheduling software package can manage the compute nodes and the latency will have almost no effect on the performance perceived by users.

5.4 Distributed Systems Integration

The Beowulf cluster concept blurs the boundary between the notion of a tightly coupled parallel computer and a distributed cluster system. Approaches to the issue of reliability in parallel systems are either based on the provision of redundant components and special purpose fault detection hardware or on the supposition that the box is unusable if any single component processor fails. Both these approaches drive up the cost of a parallel system, requiring special custom hardware or requiring high reliability specifications for individual components to ensure low probabilities of overall system failure.

Distributed clusters of processors are almost bound to have a component fail or become temporarily inaccessible across a network at some point during their operational lifetime. The probability of a single processor failure is very high and consequently the need for software to handle such occurrences is important. Fault tolerance in distributed cluster systems is an active area of research.

Practical Beowulf systems can be built either from dedicated processors or from distributed workstations that may in fact be located on individuals' desks throughout an organisation. Since it is entirely likely that the same network infrastructure technology is used for both, it is an attractive possibility to allow individual workstation nodes to dynamically join and leave an operating cluster without disruption to the overall operations. Queuing and performance models for this sort of scenario also present an interesting research area.

6 Discussion and Conclusions

We have discussed some of the key issues for designing a Beowulf system and laid out some of the economic and performance data valid for early 1999. In summary, we conclude that PC compute nodes with Intel processors running at 350-400MHz clock speed appear to give the best price/performance at the time of writing.

The optimal region of the design space is shifting as the balance between best economic processing capability and communications bandwidth changes. While Fast Ethernet is probably the best economic choice at the time of writing it is likely that some faster network technology such as Myrinet or Fast Ethernet will supplant it over the next year.

A number of research issues are still open and opportunities exist for investigation of storage and communications behaviour and optimisation for single and multi application use of a Beowulf style cluster. We intend to focus effort on software designs for managing dynamic and heterogeneous systems.

Acknowledgments

References

[1] "Gaussian 94", M.J. Frisch et al, 1994, http://www.gaussian.com

[2] "BEOWULF: A Parallel Workstation for Scientific Computation", Donald J. Becker et al, Proc International Conference on Parallel Processing, 95.

[3] "Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation", Donald J. Becker et al. Proc High Performance and Distributed Computing, 1995.

[4] "Support for Out of Core Applications on Beowulf Workstations", Matthew M. Cettei et al, Proc IEEE Aerospace, 1998.

[5] "Hrunting: A Distributed Shared Memory System for the Beowulf Parallel Workstation", Jason A. Crawford, Clark M. Mobarry, Proc IEEE Aerospace, 1998.

[6] "Resource Descriptions for Job Scheduling in DISCWorld", H.A.James and K.A.Hawick Proc 5th IDEA Workshop, Fremantle, Feb 1998.

[7] "The P-Mesh – A Commodity-based Scalable Network Architecture for Clusters", B.Nitzberg et al, Proc. 32nd Hawaii International Conference on System Sciences, Wailea, Maui, Jan 1999.

[8] "A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation", Chance Reschke et al, Proc High Performance and Distributed Computing, 1996.

[9] "Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs", Daniel Ridge et al, Proc IEEE Aerospace, 1997.

[10] "An Assessment of Beowulf-class Computing for NASA Requirements: Initial Findings from the First NASA Workshop on Beowulf-class Clustered Computing", Thomas Sterling et al, Proc IEEE Aerospace, 1998 .

[11] "Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation", Thomas Sterling et al, Proc International Parallel Processing Symposium, 1996.

[12] "Domain Growth in Alloys", Kenneth A. Hawick, PhD Thesis Edinburgh University, 1991.

[13] "A Case for NOW (Networks of Workstations)", T.E. Anderson et al., IEEE Micro, 1995 (February), p54-64.

[14] "Linux Parallel Processing HOWTO", H. Dietz, 1998.

[15] "I'm Not Going to Pay a Lot for This Supercomputer", J. Hill, M. Warren, P. Goda, Linux Journal (Dec 1997).

[16] "Beowulf Supercomputer Howto Draft", J. Radajewski, 1998.

[17] "Off-the-shelf Supercomputing", S. Shankland, 1998.

[18] "Parallel Supercomputing with Commodity Components", M.S. Warren et al, PDPTA'97.

[19] "The Beowulf Consortium", http://www.beowulf.org/consortium, Web page hosted by NASA.

[20] "Beowulf Bulk Data Server", http://www.beowulf.org/bds

[21] "Advanced Computational Systems Cooperative Research Centre", http://acsys.adelaide.edu.au/.

[22] "Iofor: The Distributed High Performance Computing (DHPC) Group's Beowulf Project", http://acsys.adelaide.edu.au/projects/beowulf