Building Australia's Fastest Computer

Frank Crawford

ac3

<Frank.Crawford@ac3.com.au>

ABSTRACT

ac3 recently installed the fastest supercomputer in Australia (1.095 TFlops) consisting of a 155 node Beowulf Cluster running Linux, Red Hat 9 and OSCAR.

The paper will outline a number of the issues related to setting up such a large system, including the environmental considerations of heat and noise, and the techniques used to install and manage the cluster in production. This will include details of a number of publicly available packages used in the management tasks, such as monitoring, patching and reporting, as well as the security aspects, both to prevent unauthorised network intrusion and local host based issues.

Finally the paper outlines details about benchmarking the system for Top500.org, including "hpl", the High Performance Linpack benchmarking program. It will also outline a number of issues with the whole Top500.org process.

1. Introduction

In 2000, The Australian Centre for Advanced Computing and Communications was established as the primary site for High Performance Computing within NSW. It is one of a number of centres established throughout Australia, others include VPAC for Victoria, SAPAC for SA and QPSF in Queensland. In addition, the national centre is the Australian Partnership for Advanced Computing located at ANU in Canberra.

ac3 was established by a consortium of seven Universities in NSW and initially housed a NEC SX5/2, a 64 processor SGI Origin 2400 and an 18 node, with a total of 68 processor IBM SP2. Overall, this was a very balanced offering with support for vector computing (NEC SX5), large shared memory multiprocessor computing (SGI Origin 2400) and distributed memory clustered computing (IBM SP2).

In 2003, ac3 decided to decommission the IBM SP2, and replace it with a new cluster. At the same time the University consortium obtained funding for a new system.

1.1 A Linux Beowulf Cluster

While the existing systems at ac3 were all 64 bit systems, an analysis of the user requirements soon made it clear that their needs would be met by different technology. After investigation it was determined that the best fit to the user's requirement would be a Linux Beowulf cluster linked by a high speed network.

In 2002, UTS as the lead for ac3 Research, the consortium of universities involved with ac3, put a proposal for a grant to fund a "Cluster-based High Performance Parallel Computer System". The proposed system consisted of 128 dual 1GHz Pentium III systems, with an aggregate performance of 400-500 GFlops, and worth approximately $1M. The system was envisaged to act as a general High Performance Computing (HPC) resource for NSW researchers in a range of disciplines, and to supplement the facilities then available at the national facility at ANU.

The choice of a Cluster-based system, was based on the recognition that such systems are by far much more cost effective per MFlop than other traditional HPC architectures. In addition, the adoption of cluster systems world-wide, particularly within the US National Science Foundation (NSF) has made available a wide variety of software suitable for researchers in a wide range of disciplines.

A Beowulf cluster is high performance massively parallel system built primarily out of commodity hardware, running a free-software operating system like Linux or FreeBSD and interconnected by a private high-speed network. It consists of a loosely coupled collection of PCs or workstations dedicated to running high-performance computing tasks, and differs from some others cluster types in that the nodes are dedicated to running the cluster jobs, and it is usually connected to the outside world through a single node.

As the system consists of separate components or nodes, and in particular each node has its own individual memory, any parallel programs need to support a distributed memory architecture. The most common programming model for such a system is

Message Passing Interface (MPI), although older implementations such as PVM (Parallel Virtual Machines) are also supported.

Aside from the individual nodes, one of the biggest differentiators in Beowulf cluster performance is the interconnection network linking up the individual nodes. Certain applications need very high speed networks to pass high volumes of small messages between individual threads, whereas other do not require much interprocess communication. As such the interconnect has a major impact on the type of jobs that run successfully on such a cluster.

1.2 Barossa Specifications

The system finally selected was based on a limited tender to a number of commodity PC vendors with previous experience with cluster systems, and consists of 155 x Dell PowerEdge 1750 servers, each with 2 x Intel Xeon 3.06 GHz CPUs. The compute nodes each have 2GB RAM, 1 x 36 GB SCSI disk, while the head nodes (used for system management) have 4GB RAM, 2 x 36 GB SCSI disks (mirrored) and the file server has 2GB RAM, 2 x 36 GB SCSI disks (mirrored) and a fibre channel connection to a Dell/EMC CX200 with 1.2TB usable disk. The original system design is given in Figure 1. The cost of this system was approximately $850K.

The core of the system is an interconnection switch, a Foundry FastIron 1500, with 11 x 16 port Gigabit blades and a Management blade. This switch is capable of non-blocking transfer between any two ports at Gigabit speed.

The system is housed in six racks, five of which contain 32 nodes, the other containing the head nodes, file server, CX200 and Foundry switch. Each rack is configured with 6 x 15A circuits. In addition, each rack contains an additional Dell PowerConnect 3248 switch used for system monitoring and management.

The operating system is based on the Red Hat 9 distribution of Linux, controlled by the package OSCAR, the Open Source Cluster Application Resources. This provides both installation and management functions.

While there is an intention of running a distributed file system using the space available on each of the compute nodes, no suitable candidate has yet been found. As such one of the nodes is a dedicated file server, connected to a small SAN and shared to all the rest of the cluster via NFS. While this is not an optimal solution, it is a stable and workable solution until a stable distributed file system becomes available. To improve performance, the space on the node's local disk is used for job scratch space.

In line with ac3's system naming policy, the system is named after an Australian wine region, in this case the Barossa Region in SA. Other systems at ac3 are clare and hunter.

As barossa (pictured in Figure 2) is intended to be used for High Performance Computing, in particular for distributed computing, the system was ranked for the Top500.org list (`http://www.top500.org`). This involved running the High Performance Linpack program, optimised

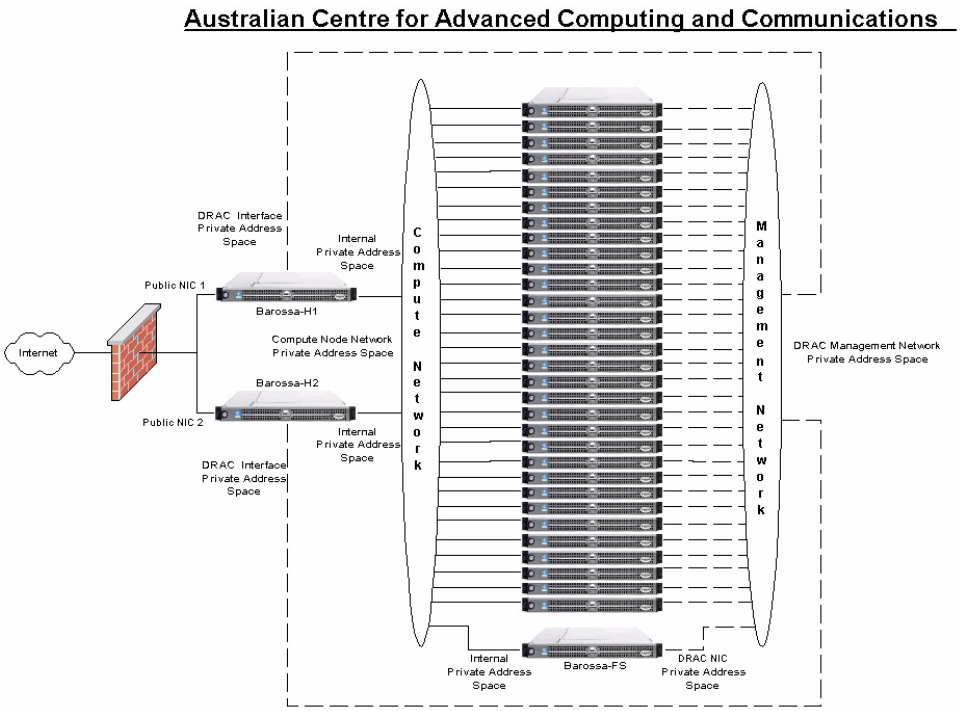

Figure 1: Original System Design for "barossa"

Figure 2: ac3 Beowulf Cluster "barossa"

for this configuration, a process that took two to three weeks. The system was benchmarked at 1095 GFlops (10^9 Floating Point Operations/sec) out of a theoretical maximum of 1860 GFlops, raking it at 108 in the 22nd list published in November 2003. This made it the fastest system in Australia, beating the previous holder, APAC's performance of 825.5 GFlops. Interestingly, in the 21st list published six months earlier, it would have ranked around 48. However, within six months, i.e. for the 23rd list published in June 2004, it had dropped to 207.

2. Environmental Requirements and Issues

While each individual server is nothing spectacular, as a collection they put a strain on ac3's computing facilities. While ac3 has only been established within the last four years and the design of the computer room was industry best practice, the rapid changes to computer architecture has meant changes to computer room requirements.

2.1 Power Issues

The best way to look at the requirements, is in terms of the power requirements for each rack, as this affects both power delivery and also heat dissipation and hence air conditioning. In 2000, most servers were 4 rack units in size or bigger, with 2 rack unit servers just beginning to be deployed. In addition, power requirements for these servers was of the order of 250W/system. Taken together, it was unlikely that there were more than 8 server/rack and hence a maximum power requirement of 2KW/rack or 1.2KW/m^2.

These power requirements could easily be addressed by a single 10A circuit to each rack and the heat can be removed with a single low powered air-conditioner. For redundancy, ac3 supplied at least two circuits to each rack and has 4 air-conditioners each with dual compressor for cooling.

Servers such as Dell's PowerEdge 1750 require 450W/system, and are each 1 rack unit in height, hence 32 or more units can be installed in a rack. This gives a power requirement of nearly 15KW or 9KW/m^2 per rack. This is a 7 fold increase in the power requirements per rack, and a commensurate increase in the air-conditioning requirements.

To supply this power, it now requires multiple 15A (or larger) circuits to each rack, with increased cabling, etc. In fact this is born out by the measured power figures seen for ac3's Beowulf cluster. Measured power requirements for a fully loaded system are approximately 45KW or 9KW/rack for fully populated racks, and is supplied by 3 x 15A circuits (with appropriate redundancy). In addition air-conditioning requirements have gone from 1 unit to 3 units, although this is confused by the issues of air flow requirements.

2.2 Temperature

As a side issue, there are also noticeable differences between an idle and a fully loaded system, with the power requirements varying from 25KW when all nodes are idle to 45KW when they are all fully utilised. Even more, the temperature variation is physically noticeable, with a 2-3degC variation felt at the back of the system, again depending on load. This variation in temperature can be seen in Figure 3, which shows the increase in computer room temperature as the cluster was brought on line. These issues were later corrected by reallocation of air-conditioning units.

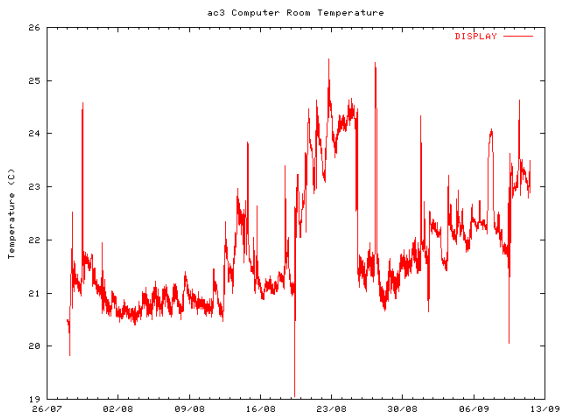

Figure 3: Temperature Variation During barossa Installation

2.3 Ambient Noise

The issue of airflow brings up a separate problem, that of noise. The volume of air passing through the 155 nodes, each of which has 7 small fans produces a very noisy environment. Some measurements of the ambient noise level within

the computer room has level of 82dB in the vicinity of the cluster and a value of 76dB away from the cluster. As a comparison, the NSW OH&S policy recommends an average noise level of 85dB over an 8 hour period, and 140dB as a peak level. Above this level, hearing protection should be worn. This is not a simple issue to address, and takes a major redesign of the individual components (e.g. nodes, racks, etc.).

2.4 Vibration

In addition, either the heating and cooling or some other internal vibrations cause another interesting problem. We have an ongoing issue of daughter and other cards, such as DIMM, gradually working loose and having to be reseated. The effect is that we start getting ECC errors from loose DIMMs or monitoring stops on loose ERA/O cards. This seems to be confined to a few nodes, but does mean ongoing monitoring and maintenance.

2.5 Cable Density

One final note that comes from this increased density is network connectivity. While barossa is a special case, it does have the issue of 150+ UTP cables all coming into the same 7 rack unit space as the centre of the system. This places a premium on the use of cable management and in particular inter-rack cabling. While this is spread over 6 racks, as an example of more extreme density, we have another customer looking at 20 servers in a single rack with 4 x NICs, teamed to give 2 network connections, and a system monitoring connection, meaning that over 100 UTP cables try to fit into a 3 rack unit space!

3. OSCAR Description

As with any computer system, the hardware is only part of the equation for a Beowulf cluster. The other essential component is the system and application software packages. For large Linux clusters there are a few packages available, and the one selected for barossa was OSCAR, the Open Source Cluster Application Resource (`http://oscar.openclustergroup.org`). This package includes components applicable to both system administration and general cluster usage.

In fact, OSCAR is really a collection of other open source packages, built with a set of standard configuration options and grouped together with a specially written installation interface. The packages themselves fall into three separate categories:

- Infrastructure - consisting of ISC DHCPD, OpenSSH and OpenSSL,
- System administration - consisting of the System Installer Suite (SIS) and C3,
- Application support - consisting of MPICH and LAM/MPI, PVM, OpenPBS batch queue manager and Maui batch scheduler.

In addition, due to the common interface available through the use of standard open source systems, there are a few additional system monitoring packages available, in particular, ganglia a distributed monitoring system and clumon, a cluster and OpenPBS monitoring system.

One of the features of OSCAR is a high degree of automation. This is a fairly obvious requirement, given the large number of nodes that require administration. One advantage that clusters have is a very high degree of replication among the nodes. For example in barossa there are 152 compute nodes, all of which have the same configuration.

3.1 Installation Support

One of the major components of OSCAR is the System Installer Suite (SIS) (`http://sisuite.org`), which itself is composed of three separate components, the System Installer, the System Imager and the System Configurator. This suite of packages allows the creation of a number of different system images, the installation of an image onto a “bare-metal” system and the configuration of the newly installed image for the specific machine.

OSCAR hides much of the details of SIS, especially the image creation performed by System Installer, which creates the image, as it creates it from a predefined set of RPMs. However, SIS is capable of using a “golden-client”, i.e. an existing system to clone, which it copies and uses as the basis for other systems.

The basic process for image installation is:

- image creation (System Imager/OSCAR)
- collection of client information (System Installer)
- load images to client (System Imager)
- configuration of host specific details (System Configurator)

The primary information collected for each client is which image is tied to the client and which IP address is associate with that host. While there are a variety of methods available for system installation, the most common is a network install, which involves a PXE boot, installation of a basic Linux kernel, download of the image, via ssh and rdist, and final boot to system image.

3.2 Booting and Netbooting

To enable netbooting of the client it is necessary to obtain the MAC addresses of all hosts, and this is one function that OSCAR adds to the process, a GUI to associate these MAC addresses with hostnames. Following this, System Installer has functions to create the appropriate DHCP configuration file including passing appropriate PXE and installer options.

One interesting feature of SystemImager is the ability for remote management of the PXE boot process. To enable this, it is necessary to configure the systems to have a BIOS boot order of Netboot, followed by hard drive. At this point SystemImager toggles the downloaded image to either be one to install a new image, or fails the PXE boot and hence falls through to booting from the hard disk.

Despite these features, there is still an issue with network loading, in that the traffic to load 155 nodes at once is extreme and as such we are generally limited to loading about 16 nodes at a time. In an effort to alleviate this problem the SIS developers have recently released a multicast/ broadcast enabled client that theoretically should allow all clients to load at once. Unfortunately, at present there still appear to be some problems.

Leaving aside the problems with the multicast client, with the current system it is possible to load 16 clients in a period of about 5 mins, and hence reload the entire cluster within an hour. This is a huge time saving over having to manually access each node for an installation.

3.3 Management

3.3.1 C3

The core of OSCAR's management utilities is the Cluster Command and Control (C3) package from ORNL (`http://www.csm.ornl.gov/torc/C3/`). It is a suite of Python based tools that use ssh to push jobs out to various nodes. By default the package affects all nodes, but there is a simple syntax to specify a subset of nodes.

The functions that C3 implements include ‘cexec’ to run a given command string on the selected nodes of the cluster, ‘cpush’ to push a file from the current system to other, ‘cpushimage’ which pushes an entire image (although not as useful as it sounds) and ‘cshutdown’ which run the shutdown on each node.

3.3.2 OpenSSH

One of the central items with this management structure is the use of ssh and common authorization keys across the cluster. The first time a user logs in, the system runs a script to create keys and inserts authentication for the local host. As all the user's home directories are NFS mounted and the same on all nodes, this allows the user to log into any node without any further authentication.

Using these basic functions more complex management functions are built, such as ‘opium’. This is used to synchronise the password and other system files across the cluster, by using ‘cpush’ push out files who’s checksums don’t match. As this can be run from cron, it means that user accounts are defined on all nodes and they see the same environment on which ever node they use.

3.3.3 Access Restrictions and OpenPBS

Of course letting users log into any node is in fact counter productive in a batch environment, as unscrupulous users can run additional jobs on any node in the cluster. To counter this, all users are blocked from logging in to any compute nodes through the use of /etc/security/ limits.conf, and then specific access is given to the support group and other users as required.

In fact, for jobs to run through PBS, the system does an implicit login, and so each user needs access to their allocated nodes. In addition, ssh is used to start up the necessary functions on any additional nodes which again needs login access. The way this is handled is via system exits within PBS which modifies the restrictions on the fly. Prior to running a job PBS invokes a prologue script as root on one of the allocated nodes, and at the end of the job an epilogue script. By using these exits, it is possible to modify /etc/security/limits.conf before the job starts and then to remove these changes at the end. As this needs to be done on each allocated node, this is also run over ssh.

The use of these system exits are also essential to patch management across the entire cluster. The current method for performing updates is to push the relevant RPM files out to the nodes (usually via NFS), and then toggle a flag for a reboot of the system (if required) when all PBS jobs are completed.

3.3.4 Remote Management

As a final issue with system management, all the servers are equipped with an Embedded Remote Access Option (ERA/O), which allows remote monitoring and management of the server. One of the major management features of the ERA/O is the ability to power the server on or off, independent of the state of the O/S. While Dell supports this through Remote Administrator Server, and a 35MB package, in

ac3's environment, we have chosen to use an open source package called PowerEdge::RAC (`http://www.lanceerplaats.nl/PowerEdge/RAC/`) which contains a Perl module to access most of the functions on this board (and other related Dell remote access cards). The most useful features of this package are the monitoring functions, which will be outlined below.

3.4 Monitoring

With such a large number of separate systems, monitoring is an important function. Currently there are three different packages in use. The first is PowerEdge::RAC, described previously. This package monitors such hardware functions as fan speed, CPU voltage, etc. Additionally, this package also accesses the hardware event log and other hardware information. By expanding on one of the sample files it is possible to plot and monitor all the hardware functions across all nodes. This is now being expanded to include notification of hardware level events that require action. A sample output showing a number of the graphs and other available information is shown below (Figure 4).

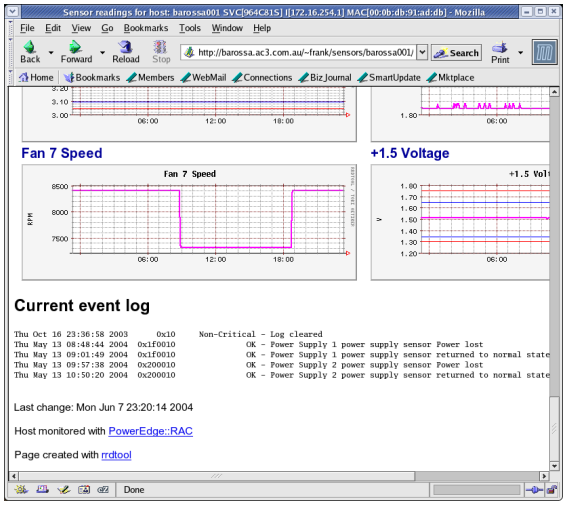

Figure 4: PowerEdge::RAC output

Of more use for monitoring system activity is a package called 'ganglia' (`http://ganglia.sourceforge.net`). This package is a scalable distributed monitoring system, which relies on a multicast-based listen/announce protocol. By default it monitors and reports on CPU utilisation, load average, memory usage, disk space and network performance, and these statistics are reported for the last hour, but all the results are stored in a RRD database, so data can be examined over any period. This gives an easy overview of the system activity (as shown in Figure 5), but can drill down to give details for an individual node.

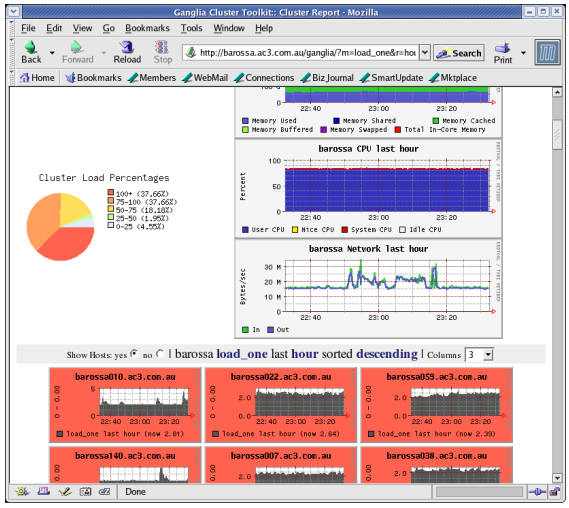

Figure 5: Ganglia output

Further, ganglia is extensible, in that additional monitors can be added by the user.

The final monitoring system available is 'clumon' (`http://clumon.ncsa.uiuc.edu`), which interacts with OpenPBS (and similar schedulers) to report on the queue utilisation and batch jobs submitted by the users. It can further drill down to give details about individual nodes, jobs and queues. This is the main tool available to system users for monitoring their system access. A sample of this can be seen below in Figure 6.

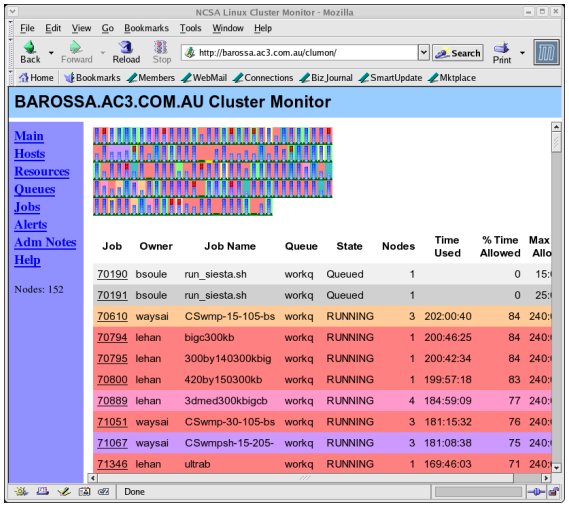

Figure 6: Clumon output

4. Usage Outside of the Cluster

While all the functions outlined are applicable to cluster management, many of them are as applicable to an environment with multiple systems. For example at ac3, we have used the SIS installation suite to clone systems to move from old to new, and also used it to duplicate database servers for a parallel database management system. To do this, it is necessary to install the various RPMs on the

"golden-client" as well as the image server. It is then necessary to prepare the client for imaging, which enables copying of the system. From this, take an image of current client and then use it for the image from SystemImager for the installation of the new client.

Similarly, many of the monitoring utilities can be used across non-related systems, for example, ganglia could be used to track details of different systems. Even further, the PowerEdge::RAC software does not rely on any cluster functions, but can just as easily be used for separate, unrelated systems.

5. Benchmarking

While it may not be the primary reason for the purchase of a high performance system, trying to get a high position within the Top500.org list *http://www.top500.org* is an interesting exercise in itself. This list has come out of work previously done separately by Hans Meuer and Jack Dongarra on the status of high performance and supercomputers through out the world. The list is based on the "best" performance as measured by the LINPACK Benchmark, which is used to solve a dense system of linear equations. For the Top500.org list, a version of the benchmark is used that allows the user to scale the size of the problem and to optimise the software in order to achieve the best performance for a given machine.

An implementation of this can be found at `http://www.netlib.org/benchmark/hpl`, but there is nothing stopping an organisation writing their own from scratch. In fact, Intel have recently released a version of the code that is specifically designed for shared memory systems that they have heavily modified for Intel architectures and compilers. The only real constraint is that the procedure must conform to the standard operation count for LU factorization with partial pivoting. In particular, the operation count for the algorithm must be $^2/_3\ n^3 + O(n^2)$ floating point operations.

What is reported is the maximum performance (R_{max}) in GigaFLOPS, or 10^9 Floating Point Operations per Second, the problem size (N_{max}), and the problem size (N_{half}) for half the maximum performance. The theoretical peak performance (R_{peak}) is also reported. For example for barossa, R_{max} is 1095 GFlops with an N_{max} of 179000, and N_{half} of 40000, while the R_{peak} is 1860.48 GFlops.

While there are a large number of parameters that can be modified to tune hpl (see Appendix 1: HPL.dat), the important ones are:

N: problem size

NB: the block size for the matrix, and

P & Q: the partitioning of the problem across nodes.

Put simply, "N" relates to the size of memory available across the system, "NB" relates to the actual algorithm used for the decomposition, and "P×Q" is equal to the number of CPUs available. Some of the other parameters can have major effect if the wrong selection is made, but once chosen, there is little variation, e.g. "BCAST", which affects the order of message passing.

The best procedure to adopt in choosing suitable values is to set the problem size to a smallish value (e.g. N_{half}) initially, then to try and optimise the other parameters, followed by then verifying them with much larger problem size values. For the barossa runs the problem size of 40000 took approximately 60 secs, while a problem size of 175000 takes nearly 1 hour.

One of the most amazing things about the LINPACK benchmark is that the performance is mostly affected by the quality of the BLAS (Basic Linear Algebra System) libraries chosen. Optimisation of the basic code has little effect. This is because any library chosen should be very heavily optimised before starting any other work. In the case of benchmarking barossa, initially we ran it with the standard Linux libraries, and obtained a value of around 500GFlops, we then made use of Intel's MKL 6.0 and immediately jumped to 850GFlops. A further upgrade to Kazushige Goto's BLAS libraries (`http://www.cs.utexas.edu/users/kgoto/`) which include hand coded optimisations, produced a figure of around 970MFlops. The final step was to switch to Intel's MKL 6.1, which included Kazushige Goto's optimisations, gave a result of near 1035GFlops. The final change was to add 8 additional computation nodes (16 CPUs) to achieve our final peak result of 1095GFlops. However, as you can see from this history, selecting the right library can more than double the performance.

In addition to the BLAS library affecting performance, it also affects the optimum value of NB. This relates to internal issues, and changes will cause some variation. For example with Kazushige Goto's library, the reported best values are around 112, which for MKL the better value turned out to be around 200.

The problem size is the next most important factor. Obviously the larger the problem size, the better the performance, all other factors equal. However, for optimum performance, the problem needs to remain in RAM, and so anything that causes swapping will also cause a dramatic drop in performance. Again, using

barossa as an example, a problem size of 181000 gave a result of 1078GFlops, while 182000 results in 876GFlops.

This also has the side effect that sometimes better performance can be obtained on less hardware. At one point, we considered running with two less nodes, as 150 (15×10) than 152 (18×16). However, (36×9) gave an even better performance, so it was not necessary.

The last item of interest is the partitioning. General experience indicates that P & Q being approximately equal give the best performance, although in barossa's case Q > P gives the best result, with P×Q = 8×36 giving a result of 1027GFlops, while 16×18 gave only 924GFlops.

Of course the overriding factor affecting the performance is the hardware, and this is something that isn't easily changed. Looking through the results in the Top500.org listing, for clusters, the biggest factors are the available memory, and the interconnect. It is obvious from the maximum problem size which clusters are composed primarily nodes with 1GB of RAM. Further, for systems that have only a Fast Ethernet interconnect (i.e. 100Mb/sec), their maximum performance is well below that of Gigabit Ethernet (1Gb/sec) or Myinet.

The peak performance (R_{peak}) is a theoretical maximum, and for barossa it is given by the number of CPUs (152 dual Xeon compute nodes = 304 CPUs) times the processor speed (3.06GHz) times 2 operations per clock tick, giving a value of 152 × 2 × 3.06 × 2 = 1860.48GFlops.

The final output from the run can be seen in Appendix 2: full_run_152.1095, but does bring up a very important point. It isn't obvious from this output, but due to minor difference in each run on each node, no two runs will necessarily generate the same performance figure. The Top500.org results accept the maximum value achieved, and provided it is reasonable, this is used. The variation can be a few percent, for example, when rerunning 'hpl' with the same parameters, values between 1095GFlops and 1081GFlops were obtained.

6. Conclusion

High Performance Computers have often lead the computing environment in both hardware and software, and modern Linux Beowulf clusters are no different. In the deployment of barossa at ac3 we have discovered issues both with the environment requirements for large systems, in particular power, cooling and noise issues, and in suitable software to automate the management and monitoring of the systems.

The variation in LINPACK results brings into question the results published in the Top500.org list, as on close examination, there are a large number of similar systems that are all ranked the same, with exactly the same R_{max} value. Unfortunately, this is both wrong, and turns the Top500.org list into a marketing tool rather than a valid representation, as a large number of the top systems have never been tested and are not even running as a cluster. This is almost like claiming that a warehouse full of workstations should obtain an entry within the list.

Finally, as interesting, the software used for the system is Open Source, but more importantly, consists of a number of Open Source projects, built on top of other open source projects, which use basic open source utilities. Put together they make a flexible and extensible system which can be tailored to any special requirement, in this case the management of the fastest system in Australia.

Acknowledgments

A complex system like this cannot be build by just a single person, and ac3's Beowulf cluster is no different. I'd like to acknowledge the efforts of the following in the construction and management of barossa: Duraid Madina (UNSW), Youzhen Cheng (ac3), Jim Lowe (ac3) and Sergey Omelaenko (Dell).

Appendix 1: HPL.dat

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
9            # of problems sizes (N)
40000 175000 176000 177000 178000 179000 180000 181000 182000 Ns
1            # of NBs
200      NBs
1            # of process grids (P x Q)
8        Ps
38       Qs
16.0         threshold
1            # of panel fact
2        PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
8          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1           BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1 2 0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

Appendix 2: full_run_152.1095

```
================================================================================
HPLinpack 1.0  --  High-Performance Linpack benchmark  --  September 27, 2000
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Labs.,  UTK
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :   40000   175000   176000   177000   178000   179000   180000   181000
          182000
NB     :     200
P      :       8
Q      :      38
PFACT  :   Right
NBMIN  :       8
NDIV   :       2
RFACT  :    Left
BCAST  :  1ringM
DEPTH  :       1
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
ALIGN  : 8 double precision words

--------------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual checks will be computed:
   1) ||Ax-b||_oo / ( eps * ||A||_1  * N        )
   2) ||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  )
```

```
   3) ||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo )
- The relative machine precision (eps) is taken to be         1.110223e-16
- Computational tests pass if scaled residuals are less than          16.0

================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8        40000   200     8    38              78.36          5.445e+02
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0158003 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0144471 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0028370 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       175000   200     8    38            3331.72          1.072e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0673868 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0089810 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0015430 ...... PASSED

================================================================================
HPLinpack 1.0  --  High-Performance Linpack benchmark  --  September 27, 2000
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Labs.,  UTK
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :   40000   175000   176000   177000   178000   179000   180000   181000
         182000
NB     :     200
P      :       8
Q      :      38
PFACT  :   Right
NBMIN  :       8
NDIV   :       2
RFACT  :    Left
BCAST  :  1ringM
DEPTH  :       1
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
ALIGN  : 8 double precision words

--------------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual checks will be computed:
   1) ||Ax-b||_oo / ( eps * ||A||_1  * N        )
   2) ||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  )
   3) ||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo )
- The relative machine precision (eps) is taken to be         1.110223e-16
- Computational tests pass if scaled residuals are less than          16.0

================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8        40000   200     8    38              77.31          5.519e+02
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0175013 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0160024 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0031424 ...... PASSED
================================================================================
```

```
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       175000   200     8    38            3311.22          1.079e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0627572 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0083640 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0014369 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       176000   200     8    38            3360.40          1.082e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0094875 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0077927 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0013420 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       177000   200     8    38            3418.90          1.081e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0043984 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0085455 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0015046 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       178000   200     8    38            3460.28          1.087e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0382668 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0082226 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0014446 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       179000   200     8    38            3490.59          1.095e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0162521 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0086945 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0014488 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       180000   200     8    38            3569.05          1.089e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0039023 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0081021 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0013326 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       181000   200     8    38            3665.82          1.078e+03
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0046288 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0083695 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0014308 ...... PASSED
================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
W11L2R8       182000   200     8    38            4589.01          8.758e+02
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0094461 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0089341 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0015777 ...... PASSED
================================================================================

Finished      9 tests with the following results:
              9 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```