Computational Science Technical Note **CSTN-018**

Trends in Cluster Computing Scheduling and the Missing Cycles

K. A. Hawick and H. A. James

2005

Parallel computer clusters now represent a mainstream resource of commodity compute cycles in many academic departments. We review our experiences over the past 7 years of building and experimenting with compute clusters and speculate about future trends. We focus on scheduling and utilisation and discuss data collected from a complete year of use of one system. We discuss scheduling inefficiencies and the lost or missing cycles that are inevitable when running a multi user resource.

Keywords: scheduling and resource management; recent historical trends in cluster computing

BiBTeX reference:

```
@INPROCEEDINGS{CSTN-018,
        author = {K. A. Hawick and H. A. James},
        title = {Trends in Cluster Computing Scheduling and the Missing Cycles},
        booktitle = {Proc. Int. Conf on Parallel and Distributed Processing Techniques
                and Applications (PDPTA'05)},
        year = {2005},
        pages = {732-738},
        address = {Las Vegas, USA.},
        month = {27-30 June},
        publisher = {CSREA},
        keywords = {scheduling and resource management; recent historical trends in cluster
                computing}
}
```

Technical Report CSTN-018

Trends in Cluster Computing Scheduling and the Missing Cycles

H.A. James and K.A.Hawick
Institute of Information and Mathematical Sciences, Massey University
Albany, North Shore 102-904, Auckland, New Zealand
Email: {h.a.james,k.a.hawick}@massey.ac.nz

Abstract

Parallel computer clusters now represent a mainstream resource of commodity compute cycles in many academic departments. We review our experiences over the past 7 years of building and experimenting with compute clusters and speculate about future trends. We focus on scheduling and utilisation and discuss data collected from a complete year of use of one system. We discuss scheduling inefficiencies and the lost or missing cycles that are inevitable when running a multi user resource.

Keywords: scheduling and resource management; recent historical trends in cluster computing.

1 Introduction

Cluster computing has become a mainstream branch of parallel computing and many organisations report success building cluster systems. Compute clusters are now achieving significant penetration onto the Top 500 Supercomputer list [15]; in the current list, published in November 2004, clusters feature as 58% of the top 500 machines. In fact, five machines in the top 10 are clusters and another three machines are constellations (clusters of clusters). The clusters typically number between 2200 and 4096 processors, connected by Gigabit Ethernet or Myrinet technology. In the June 2004 list the number 3 machine in the world comprised of 2200 commodity Apple Macintosh G5 [1] processors.

Cluster computing remains an attractive route for cost conscious university research groups in yielding systems well suited to throughput computing and a number of quality scheduling software packages are available either freely or at low cost [4]. Those large computer vendors that remain after the big supercomputer vendor shake-out of the 1990's all offer cluster products. Typically these offerings will have well integrated hardware, based around individual or shared-memory processor systems sold by the vendor as workstations or in other products. Certainly it has been our experience that these systems are easy to install and use, but nevertheless there is still a strong trend towards using very cheap PC systems and a do-it-yourself approach to hardware integration. The so-called Beowulf approach [14] and a collection of shelf mounted PC boxes connected with off-the shelf fast Ethernet (and more recently gigabit Ethernets) is still common in university departments.

Our research group built and operated Beowulf clusters in Australia and Wales for simulation work [7, 8]. A team in our present department in New Zealand operates a successful Beowulf cluster for bioinformatics and more recently theoretical chemistry work [2]. We have also worked with adhoc clusters built from Apple workstations; Alpha workstations and closely integrated clusters built from Sun Microsystems components. We are now in a position to review work over the full lifecycle of Beowulf clusters and consider the perceived advantages and disadvantages in terms of science achieved for resource expended.

In this paper we discuss the lifecycle of a Beowulf cluster (section 2) and some operational scheduling issues connected with processor utilisation achieved over a long period of time with what we believe is a typical multi-user mix (section 3). We conclude with what we hope are useful observations for other groups planning or operating similar clusters.

2 Lifecycle of a Cluster

In this section we briefly review some historical trends in Beowulf style clusters from the mid 1990s to the present.

Our original cluster experiment at Adelaide University was named Iofor after a less well known character in the Beowulf saga [14]. It was a nine-node 486 PC based cluster and was unusual, for that time, in that we heavily configured it with hard disks . We used it to scope out some of the cluster software issues for our subsequent systems and it was also used for student projects in I/O. The name was meant to be a pun on "IO". The components were rescued from a second-hand computer shop and a 10MBit/s Ethernet switch was rescued from another lab where it was being replaced by Fast Ethernet. Iofor was only operated for a few months before being relegated to a student lab as a training tool. It was later dismantled and some of the hardware components recovered and used as spares. For the most part its hardware was entirely discarded, the cabinets were incompatible with newer motherboards, its power supplies were barely able to power newer systems, its hard disks too small and its network cards at 10MBit/s were no longer useful other than in home network systems. We would estimate that the cost of disposing of Iofor's components was not significantly different from the cost of acquiring the second hand nodes in the first place.

Our second system at Adelaide was named Perseus (a local in-joke was that it would slay Titan, an aging shared memory system). The Perseus [8] system was built from dual node Pentium processors and was largely funded through collaborative grant with Mark Buntine's Chemistry research group [3]. A significant amount of local effort (in the form of graduate students) was needed to physically build the machine from its component parts. The local university workshop constructed custom shelves and an existing chilled machine room was used to house the system. The full machine ended up with 230 processors in dual node motherboards, each running Linux and with significant local disk and memory. These nodes have now been overtaken by the evolution in PC performance, but collectively the machine is still a valuable asset running simulation codes and in particular the Gaussian computational chemistry package. This machine is nearing the end of its useful lifecycle however and it is not entirely obvious what the best economic route is for it. The cabinets have value, as do the power supplies. The hard disks are perhaps still useful and the memory chips are not too out of date. It is possible to consider replacing the motherboards and processors although either is a labour intensive process and also with some risks in terms of fault tolerance and reliability. The dedicated Fast Ethernet stackable switch gear is still a valuable asset as is the cabling harness and shelving layout.

One attractive feature of systems like Perseus is that it could be incrementally upgraded although in practice few sites seem to do this. As a throughput engine, incremental upgrade is attractive, although for running large simulations that genuinely do use parallel data distribution, heterogeneity across node configurations is usually more of a nuisance than the incremental upgrades are worth. The Amdahl's law effect means that unless node heterogeneity is handled very carefully and both the users and the scheduler "know" about configuration details, then performance is wasted and the fastest nodes are held back by the slowest.

The "Helix" system at Massey University is based on dual Athlon processors and was cleverly designed by Johnson & Messom [2] to use a set of two gigabit switch networks connecting "rows" and "columns" of the processor mesh to provide a machine that is well balanced in computation and communication. The machine scored well enough on the ubiquitous linear algebra benchmark to make it into the Top 500 list in 2002 despite its very low cost. The machine was funded by the Allan Wilson Centre for Molecular Ecology and Evolution and is used extensively for bioinformatics processing. In early 2004 the machine is still a valuable asset and is used for training a new generation of parallel computing students.

Helix like Perseus however will be overtaken by events, and will either need to be replaced or upgraded. Some incremental upgrades to processors and memory capacity of individual nodes has already been performed. This is worthwhile since the machine is mostly used in job throughput mode and the queueing software has been appropriately configured to separate the heterogeneous configuration into homogeneous subsets. The Helix installation is unfortunately at capacity in terms of its physical configuration. The requisitioned Telecoms machine room was not originally designed to house large scale heat producing computer equipment and extra chiller units had to be installed. The physical area available and the heat extraction capability are therefore at their limits and under these circumstances it is more useful to replace nodes over a period of time, and re-deploy the old units as desktops or or a separate "training cluster" for students.

The Helix system has recently been upgraded in two ways: a small number of new 64-bit Opteron CPU nodes have been purchased to satisfy the growing demand for general-use higher-precision numerical simulations, and a group of theoretical chemists have funded a separate series of new 64-bit nodes called "Double Helix". Double Helix currently comprises of 20 dual-processor nodes running a 64-bit Linux kernel. All but one node features 4Gb of shared memory, while the last has 16Gb. The cluster is connected via Gigabit Ethernet. Together with the Helix and smaller "Sisters" cluster (a 16-processor teaching cluster), our local constellation has thus far served the research community well.

Some conclusions regarding the hardware for the Iofor, Perseus and Helix systems have emerged. The choice of dual processor nodes is still a good one. Quad nodes remain too expensive and would involve memory contention effects unless very large (expensive) memory configuration options were chosen for each node. Schedulers such as PBS [10] can be configured to make good choices concerning job partitioning amongst dual processor nodes. The gigabit Ethernet is now essentially a commodity item and appears quite sufficient in bandwidth for gigahertz processing nodes. The Helix dual network design is however a well worth-while addition for the incremental cost of dual network cards. This balance point will have to be reviewed however as processor speeds continue to increase, and gigabit Ethernet may not be sufficient for the next generation of machine. At present Helix nodes use 1 GByte of memory per dual node. This is now barely sufficient since the dual processors compete for this memory. Much of the simulation work we are now pursuing requires more memory. It is likely to make sense to fill next generation nodes with as much memory as is economic and to try to buy it in units that allow later upgrade. In practice it is difficult to buy PC mother board systems that will allow practical or economic use of more than 2 GBytes of memory. Unfortunately, 4GB and beyond is now strongly desirable. The next generation of 64 bit processors will support this more comfortably.

However the major drawback to us is the large memory requirements (4-8Gb) of our simulations and the fact that most of the available 64-bit nodes only have 4Gb shared between two processors. The one 64-bit node with 16Gb of memory is under continual use by the group that owns it, leaving few spare cycles for other members of the university's research community to use. We are currently experimenting with Apple Macintosh G5 technology and in particular their new 64-bit operating system, with the view to constructing a small cluster of well-configured nodes to serve as our simulations framework.

The software typically run on Beowulf systems around the world is almost ubiquitous. Our systems used the Gnu/Linux [5] operating system, compiler and toolkits; parallel library packages such as MPI [12]; scheduling software such as PBS [10] and various application packages as dictated by user needs. Our own research activities generally involve custom written simulation codes for which the Gnu compilation system is entirely adequate, although some work has benefited from advanced Fortran compilers such as that from Portland. We have found there is still plenty of scope for computer science and software engineering research and development work in the arena of software for Beowulf clusters. Our group has had a number of worthwhile student projects including work on scheduling [11] and super-accurate timing and benchmarking [6]. Nevertheless the existing public domain software base is already quite adequate for many cluster application scenarios.

An important non-technical conclusion concerns the support arrangements for Beowulf clusters. Our systems operated as research assets by computing departments benefit from having an enthusiastic group of postgraduate student administrators and support team. This works where students have a good background in operating systems and parallel computing and in particular have experience with Unix. It was our experience in a UK university where the Unix culture entirely been replaced by a Microsoft teaching background that it was very difficult to make good support arrangements for cluster systems. Happily we can report we did finally manage to train up a new generation of Unix-aware graduate students to love and care for our clusters.

3 Schedule Management

Different cluster systems have different user and job mixes. A traditional argument in favour of large supercomputer installations has been better resource utilisation occurs from sharing an expensive asset amongst many research groups. This can be true, but is not always so. Some groups can operate small to medium clusters at very high utilisation when a small number of application codes are involved and their characteristics well understood.

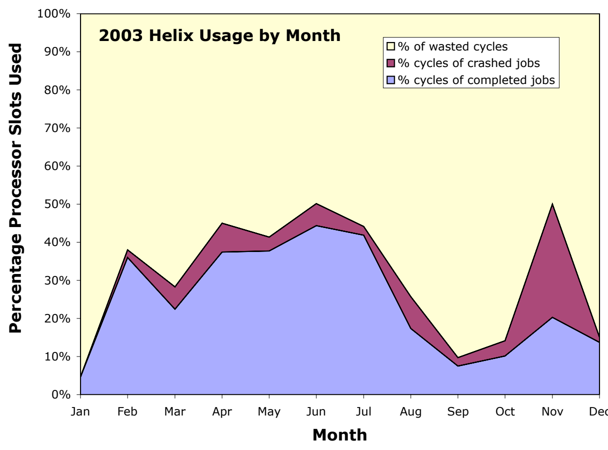

Figure 1: 2003 jobs by month, foreground queue only

Figures 1 and 2 shows the percentage of used, wasted and crashed processor cycles on the Helix system during 2003 and 2004 respectively. There are some interesting trends in these pictures that in insightful for managing a university cluster system. The dips in the overall pattern correspond roughly to teaching semester times. Over all of 2004 some 13% of the processor cycles were lost due to job crashes. This seemed surprisingly high and we analyse these missing cycles in section 4.

Figure 3 shows foreground queue figures for all user jobs. The data is histogrammed into job size. In 2004 our cluster predominantly ran single processor or dual processor jobs, although a small percentage - mostly student exercises used larger numbers of parallel processes. We are able to identify particular job signatures in the spectrum shown. The base distribution from many users doing different work for various durations all the way up to the 2-week queue limit accounts for only a minor part of our system's utilisation. Usage is dominated by the peaks labelled "2,3" and "4,5". These come from computational chemistry jobs [13] and computational physics jobs [9] respectively. The former typically use two processors, the latter are single processor jobs.

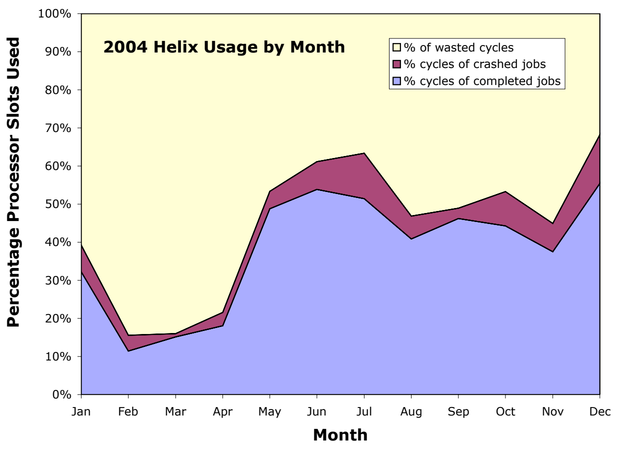

Figure 2: 2004 jobs by month, foreground queue only

In an attempt to better manage the two dominant peaks of the job mix distribution, in mid 2004 an additional " background queue" was introduced onto the Helix system. This could be used to satisfy high-volume users' demands for processing cycles and also reduce the contention between high-volume users and regular users. This background queue featured an elevated 'niceness' level, thus reducing the priority of queued jobs.

The analysis program was then run combining the foreground and background queue statistics between June and August of 2004. This had the effect of: increasing the percentage of processor slots used for completed jobs from 48% to 59%; reducing the percentage of processor slots that were wasted due to job crashes from 8% to 6%; and reducing the percentage of unused processor slots from 43% to 3%.

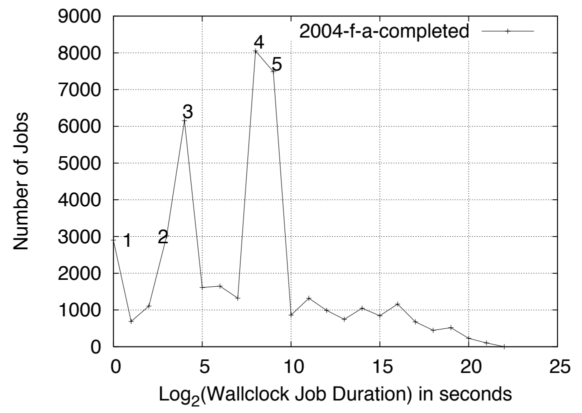

Figure 3: Completed all jobs in 2004, all users, foreground queue

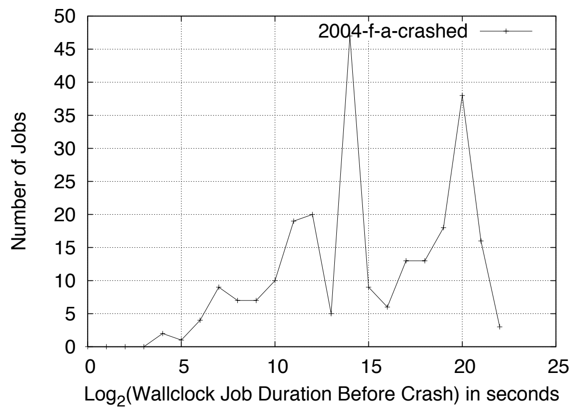

Figure 4: Crashed all jobs in 2004, all users, foreground queue

Interestingly, the method that was chosen to implement the background queue was not simply to have a multi-level queueing structure as is common in operating systems, but to allow the PBS scheduler to begin processes on all processing nodes, but with lowered priority. Typically, cluster managers prefer to only have a single executable job running on each node to combat any contention issues for limited valuable resources, such as disk space or memory utilisation. This doubling-up method had the effect of increasing the effective load on each of the processing nodes, but the advantageous effect of being able to 'soak up' any available cycles due to the higher-priority job being I/O bound or simply unable to use all the processor cycles.

At the time of writing this article, in the light of the percentage of long-running jobs that fail due to resource contentions, the "background queue" experiment is over and it has been removed. We therefore expect to see a rise in the number of unused processor slots but a decline in the number of slots wasted due to job crashes. As so often in computer systems design there is a tradeoff involving economic and non technical issues.

4 The Missing Cycles

The surprisingly high fraction of jobs that did not complete deserve further examination.

Figure 4 shows the job size distribution for just those jobs which crashed. Closer analysis of the two peaks in this distribution revealed that the left hand peak is due to a new simulation code we started running in late 2004, and which uses a lot of dynamic memory allocation. These jobs run well on one of the dual processor nodes providing there is no other program contending for memory on the same node. This problem can be partially alleviated by allocating memory in this program when it first starts running. Other starting programs on the same node may then be blocked due to inadequate memory but the long running job no longer crashes.

The right hand peak is harder to explain but seems to be linked to other resource contention effects. As discussed above dual processor nodes are attractive in terms of cost-performance ratios, but they do seem to be prone to job interference effects. The mean time between failure of individual nodes on our system is too low to explain away the anomalously high job failures. However, when we consider cooperative effects, a temporary hardware glitch on a processor such as overheating can actually drag down jobs on both processors of the dual node, but also any multi-processor parallel jobs that happen to straddle it. Other effects we have observed are associated with temporary disk space filling up on a node. This can also cause a fragile job to crash - sometimes annoyingly just before the critical moment of printout before completion.

We are presently undertaking a detailed statistical analysis of these effects. A useful observation however is that although university clusters are often built "on the cheap" in terms of both hardware and software - there is no substitute for having dedicated acolytes tend the job and intervene to fix unexpected problems such s over full disks and crashed job queues. Modern queue management software is very useful in helping to manage a cluster queue system but cannot anticipate the unanticipatable!

5 Conclusions

We conclude that clusters are clearly here to stay for some time in cash starved university research groups. We have enjoyed building and operating clusters and they certainly provide an effective compute resource. We have summarised our historical experiences of Beowulf systems from "cradle to grave" as it were and discussed some of the lifecycle trends and total cost of ownership issues.

It is inevitable that there are savings to be made by running multi-user multi-project systems, but there are some job interference effects that may cause cluster operators some frustrations. We also observe a down-turn since the 1990s in the amount of user jobs that are truly multi-processor parallel ones. This is no doubt a reflection of the useful compute power of individual processors. Clusters are still very valuable throughput engines. However, care must be taken when designing new clusters ensure that shared-memory nodes do, in fact, have enough physical memory to to avoid the resource contention effects we have described above.

Finally, we offer our believe that managing computational experiments such as long running simulations is hard to completely automate.

Acknowledgements

We thank Massey University and the Allan Wilson Centre for the funding of the "Helix" cluster system, and the University of Adelaide and the Australian Government's Cooperative Research Centres programme for the funding of work on the Iofor and Perseus systems. We owe great thanks to all our colleagues who have helped build and operate the systems we describe here - particularly: Martin Johnson, Andre Barczak, Chris Messom, Lorri O Brien, James Chai, Mark Buntine, Duncan Grove, Ade Fewings, Paul Coddington and Francis Vaughan.

References

[1] Apple Computer. "Apple - Power Mac G5" Available from www.apple.com/powermac

[2] A.L.C. Barczak, C.H. Messom and M.J. Johnson, "Performance Characteristics of a Cost-Effective Medium-Sized Beowulf Cluster Supercomputer," in Res. Lett. Inf. Math. Sci., 2003, Vol.5, pp 1-10. Available from iims.massey.ac.nz/research/letters

[3] M.A. Buntine, V.J. Hall, F.J. Kosovel and E.R.T. Tiekink "The Influence of Crystal Packing on Molecular Geometry: A Crystallographic and Theoretical Investigation of Selected Diorganotin Systems," J. Phys. Chem. A., 102, 2472-2482 (1998). See research group at www.chemistry.adelaide.edu.au/staff/mbuntine.html

[4] T.L. Casavant and J.G. Kuhl. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems, May 1986.

[5] Free Software Foundation, "The GNU Operating System", Available from www.gnu.org also Linux Online, "The Linux Home Page at Linux Online", Available from www.linux.org

[6] D.A. Grove, "Performance Modelling of Message-Passing Parallel Programs", PhD Thesis, the University of Adelaide, June 2003. See also D.A. Grove and P.D. Coddington, "Communication Benchmarking and Performance Modelling of MPI Programs on Cluster Computers" in Proc. of Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS'04), Santa Fe, April 2004.

[7] K.A. Hawick, D.A. Grove and F.A. Vaughan, "Beowulf - A New Hope for Parallel Computing?," Proc. 6th IDEA Workshop, Rutherglen, Victoria, Jan 1999. DHPC cluster computing information available from www.dhpc.adelaide.edu.au/projects/beowulf

[8] K.A. Hawick, D.A. Grove, P.D. Coddington and M.A. Buntine, "Commodity Cluster Computing for Computational Chemistry", in Internet Journal of Chemistry 3, number 4 (2000).

[9] K.A.Hawick and H.A.James, “Ising Model Scaling Behaviour on Small-World Networks,” Technical report CSTN-006. Available from www.massey.ac.nz/˜kahawick/cstn/006/cstn-006.html

[10] R.L. Henderson and D. Tweten. Portable Batch System Requirements Specification. NAS Scientific Computing Branch, NASA Ames Research Center, California, April 1995.

[11] H.A. James, “Scheduling in Metacomputing Systems”, PhD Thesis, the University of Adelaide, July 1999.

[12] Message Passing Interface Forum. MPI: A Message Passing Interface Standard. Available from www.mpi-forum.org.

[13] P. Schwerdtfeger, Markus Pernpointner, Witold Nazarewicz, “Calculation of Nuclear Quadrupole Coupling Constants”, In “Calculation of NMR and EPR Parameters: Theory and Applications” (Editors: M. Kaupp, M. Bühl, V. G. Malkin), Wiley-VCH, Weinheim, 2003; pgs. 279-291. See research group at ifs.massey.ac.nz/staff/schwerdtfeger.shtml

[14] Thomas Sterling, Donald J. Becker, Daniel Savarese, Michael R. Berry and Chance Reschke Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation, Proc. International Parallel Processing Symposium (IPPS’96), April 1996.

[15] Top 500 Supercomputer List. Available at www.top500.org