The Cluster Challenge: 6 Students, 26 Amps, 44 Hours

Robert Beck, Antoine Filion, Paul Greidanus, Gordon Klok, Chris Kuethe, Paul Lu, Cameron Macdonell, Andrew Nisbet, Stephen Portillo

University of Alberta, Edmonton, Alberta, T6G 2E8, Canada
ccteam@cs.ualberta.ca

Abstract. The inaugural Cluster Challenge competition at the Supercomputing 2007 conference saw six teams, each with six undergraduate students, compete to complete a computational science workload within 44 hours. The HPCC benchmarks, GAMESS, POP, and POV-Ray applications formed the workload. One constraint on the cluster was that it had to draw less than 26 amps total at 120 volts. The University of Alberta team placed first in the 2007 challenge.

We describe our SGI and Linux-based cluster environment, our experiences with hardware and software in preparation for the challenge, and our experiences during the challenge itself. To seasoned cluster administrators and users, many of our observations are familiar, but the Cluster Challenge is an interesting data point in the state (both good and bad) of cluster technology (hardware and software).

1 Introduction

The University of Alberta had one of six teams of undergraduate students competing at the Cluster Challenge [12] at the Supercomputing 2007 conference in Reno, Nevada. The purpose of the challenge was to motivate and educate undergraduate students in cluster computing, and to demonstrate the maturity of high-performance computing tools. The Alberta team won the Cluster Challenge with a cluster built from Altix XE310 systems provided by vendor partner SGI.

In this paper, we discuss some of the techniques that worked for the University of Alberta team and some which did not. Some of the most important lessons we learned were:

1. Know your application at a non-trivial level of detail (e.g., compiler flags, how to restart jobs).
2. Measure power consumption. In our case, many of our initial assumptions were not borne out by real testing.
3. Consider memory, as well as processors. For certain applications, having nodes with extra memory is beneficial.

1.1 The Cluster Challenge

The Cluster Challenge was structured as a throughput contest. The two biggest constraints under which teams operated were:

1. The cluster had to be built and run by six students with no postsecondary degree.
2. Power was restricted to two 120 volt circuits, with a maximum of 13 amps of draw allowed on each circuit.

Within those parameters, the six students forming a team had to build a cluster with a vendor partner. By November, the students had to ship their cluster to the Supercomputing 2007 Conference site in Reno, then set up and run the cluster at the competition. The competition itself was to run a set of jobs consisting of the High-Performance Computing Challenge (HPCC) benchmark suite [5, 16], Persistence of Vision Raytracer (POV-Ray) [9], Parallel Ocean Program (POP) [8, 17] and General Atomic and Molecular Electronic Structure System (GAMESS) [4, 18] applications. The teams knew what applications would be used in the competition ahead of time and could use the time before the competition to acquire knowledge about the applications and tune their cluster. The actual problem set data, and the point value for completing each problem set, were given to the students when the "flag dropped" to start the competition.

1.2 The Cluster Hardware

The University of Alberta cluster was supplied by SGI, and was built using four Altix XE310 servers. Each of the 1U servers had two nodes sharing a single power supply, which gave us an advantage by having less waste due to power supply inefficiencies. The nodes each had two quad-core Intel Xeon 5355 CPUs, running at 2.67 GHz, with 8 MB of L2 cache. The system had a total of 64 cores (i.e., 4 servers, 2 nodes per server, 2 processors per node, 4 cores per processor). Six of our nodes had 16 GB of RAM, and two of them had 24 GB of RAM, which helped us keep problem sets in memory and off of the 250 GB SATA hard drive. Each node also had a 4X DDR (20 Gb/s) Infiniband interface built onto the motherboard. The cluster had both a gigabit Ethernet switch and a Voltaire ISR 9024 Infiniband switch.

The unique nature of our cluster hardware (e.g., two nodes share one power supply) offered a significant advantage in density. This was a great advantage with respect to power consumption, as we were able to use more of our power budget towards memory, and the interconnects.

2 Preparation for the Competition

2.1 Setup and System Software

We installed the head node using Scientific Linux 4.5, and then we installed OSCAR 5.0 [11, 15], which provided us with a good set of tools, pre-configured

and ready to run. OSCAR includes SystemImager/SystemInstaller which built a node image for cluster nodes on the head node, then quickly deployed cluster nodes by booting them off the network using the Preboot Execution Environment (PXE), and imaging the nodes using the SystemImager software. This rapid deployment allowed us to make changes to the entire cluster quickly and easily, as opposed to manually installing each system. The OSCAR package includes a job scheduler, cluster management tools, and monitoring software.

For the scheduler, we elected to use Sun Grid Engine (SGE) [10], since our team was more familiar with SGE's interface than the default Maui/Torque scheduler. The bundled cluster management tools (C3) were useful for executing commands on all nodes simultaneously, and copying minor changes to the entire cluster without performing a full re-image of each system. OSCAR includes the monitoring tool Ganglia [3], which proved useful for monitoring and displaying cluster activity in real time. Ganglia was also used at the competition, shown via an LCD projector and screen, both to keep us aware of the activity on the cluster and to show our activity to those visiting our booth.

The Intel C compiler version 10.0.23 was chosen to compile the applications after a comparison of run-times against applications compiled with the GNU C compiler (gcc) distributed with Scientific Linux 4.5. We did most (but not all) of our benchmarking for these purposes using POV-Ray, after considering our platform and published literature [13].

A plethora of choices face a cluster architect when choosing an MPI library for a cluster and a good deal of preparatory work was done testing these various choices with the challenge applications and our cluster hardware. We tested LAM, MPICH, MVAPICH2, and OpenMPI, eventually choosing MVAPICH2 because it was the only one which would compile with Infiniband support in our chosen version of Linux. We used the OFED [7] packaging to compile and build MVAPICH2 and related tools.

We chose to implement a small firewall in front of our cluster, and to have the firewall perform Network Address Translation [NAT], using RFC 1918 addresses for our cluster itself. The firewall ensured that we need not worry about security issues from the conference floor network, and also allowed our cluster to be built, run, and tested as a self contained unit, using consistent IP addresses. This made moving the cluster from network to network much simpler, as it required only a change to one IP address on the firewall, instead of the entire cluster. The Firewall machine was built on a small embedded PC using OpenBSD 4.2.

2.2 Power Measurement

Given the power requirements of the competition, considerable effort was expended maximising performance per amp. Power consumption was measured for both the cluster as a whole as well as individual removable components with a variety of equipment including a simple Kill-a-watt meter, and sophisticated Fluke 43B power quality meters. Whenever the possibility to save power by the removal of a component was available, its power draw was measured and these

Component	Amps
Fan Speed Full	3.24–3.34
Fan Speed Auto Server	3.15–3.20
Fan Speed Auto Desktop	3.12–3.16

Table 1. Effect on power consumption of BIOS fan settings

savings carefully weighed against the utility that component provided. We measured both the power consumption of the components of a single system, as well as the total power consumption of the cluster, both loaded and without load.

In measuring the power consumption of the components of the systems, one useful discovery was that, over the entire cluster, we could save nearly a full amp by switching the fan control in the BIOS. Instead of the default setting of "full speed" all the time to "desktop" mode, where the fan speed would vary with temperature, power could be saved (Table 1).

We had initially suspected that removal of the disk drives from the compute nodes might save us a good deal of power and contemplated using a diskless, or flash disk based setup. To our surprise, our measurements, even under load, showed that disks consumed little in the way of power, which led us to abandon our initial thoughts of removing disks to save power. Finally, our testing showed us that each node's eight Fully Buffered memory DIMMs were a major consumer of power. The memory drew almost an amp per node. In the end, it was felt that the presence of all the memory and hard disks was more important to performance then any power savings we could achieve by running diskless, or with reduced memory.

The measurement of the total power consumption of the cluster under load was performed while running the POP application on sample data sets, as well as the HPCC benchmark suite on the cluster. Our initial conclusion from these experiments was that we would be able to use seven nodes (i.e., 56 cores) for the competition without exceeding the power requirements. As discussed later, only 48 cores were used for the HPCC benchmarks during the competition.

2.3 Application Characterisation

We spent most of our preparation time prior to the competition becoming familiar with the applications which would be run at the Challenge. We searched for sample data sets wherever they could be found to test each application, as well as for previously published results on their performance [17]. Ensuring that our build of the application was correct, and would produce accurate answers, was important. For example, in our experience, POP required a careful choice of compiler flags to produce correct results. We also used sample data sets to characterise the load generated and resource requirements of each application.

HPC Challenge Tuning the HPCC benchmarks is a complicated problem because there are many choices in supporting libraries and parameters. The team

choose to use HPCC built with Intel compilers and linked against the Intel Math Kernel Library. To our detriment, not all tuning opportunities were thoroughly investigated. For example, the Fast Fourier Transform (FFT) code included with HPCC was not replaced with an optimised external library, and we did not test all of the available basic linear algebra systems. Prior to leaving for the competition, we settled on parameters P=7, Q=8, N=92000 and NB=128, which resulted in a top LINPACK score of 338 Gflops on 56 cores. At the competition, our performance was lower (see below).

POV-Ray: Graphics Rendering Several of the team members had prior experience with the POV-Ray ray tracer, which is used for rendering still images, or videos. For the competition, we were told that the POV-Ray jobs would be rendering frames of video, which can be distributed to many processors for rendering with no inter-node communication. Our work with POV-Ray consisted of comparing the speed obtained with different compilers and compiler optimisation flags, as well as ensuring we could partition a POV-Ray job up by frames and distribute it effectively to available nodes via the SGE job scheduler.

POP: Ocean Modelling We had some difficulty finding data sets to work with for POP, as data sets are not widely available on the Internet. In this case, much of the work was done using the small sample data sets that are distributed with POP, as well as a few limited samples found on the Internet. The application was discussed at some length with domain expert Dr. Paul Myers of the University of Alberta, who provided his expertise in ocean modelling to the students so that they could gain an understanding of how to interpret the POP output for correctness. Some external sources [2] and the POP Users Guide [1] were also used for this purpose. On our hardware, it was observed that POP would scale reasonably well up to 32 cores, and it was decided that it would be run on 32 cores or less, per job, at the competition.

GAMESS: Quantum Chemistry We worked with domain experts in this area to gain an understanding of the application. GAMESS developers Dr. Mariusz Klobukowski of the University of Alberta and Dr. Mark Gordon of Iowa State University (who, coincidentally, visited the University of Alberta in September 2007 for a series of lectures unrelated to the competition), both took time to meet with us and provided insight into how the application worked. They also provided some key hints on how to use GAMESS's estimate of memory requirements and how to interpret both GAMESS input and output files. As well, team member Antoine Filion was taking a chemistry course in the Fall 2007 term and was shown by Melissa Gajewski (his teaching assistant and a Ph.D. candidate with Dr. Klobukowski) how to restart GAMESS jobs from intermediate results, should one be terminated. All of their advice proved valuable at the competition.

As part of our pre-competition testing, we observed that GAMESS had scalability concerns with many problem sets and, whenever possible, we worked to

minimise the necessity of running inter-node GAMESS jobs. We observed that GAMESS performed much better if it could be confined to a single node and could then exploit the better performance of shared memory for interprocess communication. As such, GAMESS benefited greatly from being run on fewer nodes with more memory available to it.

As a result of working with GAMESS ahead of time, we chose to obtain extra memory for two "fat memory" nodes, which were increased from 16 to 24 GB of RAM. We then used the *consumable resources* feature of Sun Grid Engine to ensure that the GAMESS jobs were run on the large memory nodes, and not on the other nodes.

3 The Competition

The competition itself consisted of a set-up day in Reno, with some direction from the competition organisers. We were given information on points breakdown for each of the problem sets, but not the actual problem sets, during the set-up period. We also found out that the HPCC benchmark would be worth points, but this needed to be completed before we could get the other data sets.

3.1 Starting with HPCC

We had come to Reno with our HPCC parameters chosen and which results we should expect, however a complication arose due to differences in the power measuring equipment (i.e., our meters used pre-competition versus the power distribution units (PDU) at the competition). During the set-up day, we discovered that we could not run HPCC on the full 56 cores as we had expected due to the extremely high power consumption during the LINPACK portion of the benchmark.

We were also plagued by an issue related to our high-speed interconnect during the HPCC portion of the contest. Our Infiniband stack locked up, requiring us to throw away the results computed so far and costing us over 90 minutes of time before we received datasets for the remaining applications. In the end, it was decided to proceed with a smaller value for N, which resulted in a lower LINPACK number, but allowed us to complete the HPCC benchmark quicker in order to not lose time on the three other applications in the contest. The HPCC numbers achieved (227 Gflops) compared poorly to the best numbers produced by other teams and was less than half of the theoretical peak of the cluster.

3.2 Scheduling Strategy

At the competition, but before the official 44 hours started, we were given an estimated run-time of each application's data sets to allow use to gauge the difficulty of each job, as well as the point value for completing each job.

The team took a greedy approach to scheduling, attempting to get the most points per CPU minute as quickly as possible, based upon a straightforward

spreadsheet (Table 2) constructed at the competition to compute the estimated CPU-minutes-per-point for each data set.

Job	Points	Run-time (CPU-min)	CPU-min/point	Scheduling Priority
POP A	5	33409.30	6681.86	14
POP B	5	31446.56	6289.31	12
POP C	5	31879.79	6375.96	13
POP D	5	21443.72	4288.74	10
POP E	5	29488.77	5897.75	11
POV pollen6	3	1347.47	449.16	1
POV SquidAtom	4	16797.60	4199.40	9
POV POV_SFAI	4	16228.80	4057.20	7
POV Scene005	4	14025.60	3506.40	5
GAMESS 1	4	12179.20	3044.80	2
GAMESS 3	4	12345.60	3086.40	3
GAMESS 4	4	12972.80	3243.20	4
GAMESS 6	4	16780.80	4195.20	8
GAMESS 7	4	14316.80	3579.20	6

Table 2. Spreadsheet table used to determine scheduling priority

We used our calculated priorities based on the provided run-time estimates to determine our scheduling order for jobs in SGE. The jobs were not necessarily completed in that order due to the resource requirements of each job. Also, some jobs took significantly less time to run on our cluster than the estimate provided.

3.3 Power

Power measurements before the competition proved to be slightly different than the readings obtained by the particular PDU used at the contest. Consequently, many of our load tests were re-done during the set-up day before the competition started. As discussed earlier, we were surprised to find that, when measured with the contest equipment, we could field 64 cores for the application phase, but that we could only run 48 cores for the HPCC component of the challenge.

While using 64 cores, it was necessary to constantly monitor the system to stay as close as possible to the 13 amps limit (to achieve the best performance) while not going over (to avoid disqualification). The Cluster Challenge organisers provided a mechanism (via email and a back panel LED on the PDU unit) to indicate when the limit was exceeded. We actively monitored the cluster under load, and made use of the *software controlled clock modulation* [6] of our Intel Xeon X5355 processors. With this mechanism, we were able to manually intervene to control power consumption without interrupting job activity, by scaling back the CPU speed and suppressing spikes that would have pushed us over the 13 amps limit on each of the provided circuits.

3.4 Power Outage

During the competition, a unplanned power outage occurred at the Reno Convention Centre where we were competing. At this time, the cluster was running two GAMESS jobs, and various POV-Ray jobs. One GAMESS job had just started (within minutes of the outage) and had achieved no useful results yet so it was simply resubmitted. Another job had run for 4 hours, however this job was successfully restarted from intermediate results, using the advice gained in our pre-competition preparations. Some 30 frames of the POV-Ray animation were lost and needed to be recomputed. But, in the end, relatively little work (some tens of minutes of computation) was actually lost. And, with the SGE scheduler running, jobs were restarted within minutes of the power being restored and the systems restarting.

3.5 Stragglers at the end

Nearing the end of set of jobs to be run, we were left with one problematic job that would not complete–the GAMESS run which had been restarted from the power outage. This job was completed successfully by rescheduling it simultaneously on free nodes. Specifically, the same job was started simultaneously on 8, 16, and 32 cores. In the end, it was the 8 core job that finished first, even though it was started later than some of the other jobs scheduled on more nodes. This tactic of scheduling multiple instances of a slow job at the end of a workload is something to be considered in cases of stragglers in a workload, as noted by other systems [14]. This final problematic job finished with approximately 8 hours left in the competition.

4 Concluding Remarks

The Supercomputing 2007 Cluster Challenge was an excellent opportunity for the participating students to learn about cluster computing. Working with a restricted power limit meant that some considerable work was done in characterising the power profile of the system. While there were some power savings to be had, our initial assumptions about saving power by disabling or removing devices proved not to be worthwhile on our system. Our experience shows that systems software, such as OSCAR and the bundled tools, allow less-experienced users to more-easily build and run a clustered computing environment. Finally, while far from having exhaustive knowledge, we gained a useful understanding of several applications, and how to apply that knowledge to running real jobs on a real cluster. Knowledge about application scalability and nodes-versus-memory trade-offs were important to our success.

5 Acknowledgments

Thank you to SGI, our vendor partner, to the organisers of the Cluster Challenge, and to the many individuals who helped us with the applications.

References

1. Parallel Ocean Program (POP) User Guide. http://climate.lanl.gov/Models/POP/UsersGuide.pdf, 2003.
2. The PGI Guide to POP. http://www.pgroup.com/resources/pop/pop201_pgi42.htm, 2004.
3. Ganglia Monitoring Software. http://ganglia.sourceforge.net/, 2007.
4. General Atomic and Molecular Electronic Structure System. http://www.msg.chem.iastate.edu/gamess/, 2007.
5. HPC Challenge Benchmark. http://icl.cs.utk.edu/hpcc, 2007.
6. IA-32 Intel Architecture Software Developers Manual Volume 3A: System Programming Guide. http://http://developer.intel.com/design/processor/manuals/253668.pdf, 2007.
7. Open Frameworks Enterprise Distribution. http://www.openib.org, 2007.
8. Parallel Ocean Program. http://climate.lanl.gov/Models/POP, 2007.
9. Persistence of Vision Ray Tracer. http://www.povray.org, 2007.
10. Sun Grid Engine. http://gridengine.sunsource.net/, 2007.
11. The OSCAR Project. http://oscar.openclustergroup.org/, 2007.
12. Supercomputing 2007. SC07 Cluster Challenge. http://sc07.supercomputing.org/?pg=clusterchlng.html, 2007.
13. Nicolas Calimet. Getting the best performance of POV-Ray 3.6 for Unix. http://pov4grasp.free.fr/articles/fastpov1/, 2004.
14. Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. of 6th Symposium on Operating Systems Design and Implementation (OSDI '04)*, pages 137–150, San Francisco, California, USA, December 2004.
15. Benoît des Ligneris, Stephen Scott, Thomas Naughton, and Neil Gorsuch. Open Source Cluster Application Resources (OSCAR): design, implementation and interest for the [computer] scientific community. In *Proc. of 17th Annual International Symposium on High Performance Computing Systems and Applications (HPCS)*, Sherbrooke, Québec, Canada, May 11–14 2003. http://citeseer.ist.psu.edu/740761.html.
16. Jack J. Dongarra and Piotr Luszczek. Introduction to the HPCChallenge Benchmark Suite. Technical Report ICL-UT-05-01, Innovative Computing Laboratory, University of Tennessee, 2005.
17. Darren J. Kerbyson and Phillip W. Jones. A Performance Model of the Parallel Ocean Program. *International Journal of High Performance Applications*, 19:261–276, 2005.
18. M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S.J. Su, T.L. Windus, M. Dupuis, and J.A. Montgomery, Jr. The General Atomic and Molecular Electronic Structure System. *Journal of Computational Chemistry*, 14:1347–1363, 1993.