Managing High Volume Astronimical Data with Heterogeneous Beowulf Clusters

John Salmon, Daniel F. Savarese, and Thomas Sterling
{johns, dfs, tron}@cacr.caltech.edu
Center for Advanced Computing Research
California Institute of Technology
Mail Code 158-79 Pasadena, CA 91125

Abstract

In cooperation with the European Southern Observatory (ESO), Caltech has investigated the application of Beowulf clusters to the management and analysis of data generated by large astronomical instruments, exemplified by the Very Large Telescope (VLT) in Cerro Paranal, Chile. The VLT consists of four 8-meter telescopes that can operate independently or in unison and will generate 20 terabytes of image data per year when fully operational in 2001. Over 1 terabyte of data will be generated in 1999 as the VLT sees first light. This data is archived and made available to the astronomical community for online reduction and analysis. Several hundred data reduction requests may be active at any one time. The sheer quantity of data combined with the large number of clients necessitates the application of parallelism to the responsive reduction of data. Using characterstic workloads, we find that a Beowulf cluster can generate astronomical data reductions with near-linear scaling until it becomes limited by available network and memory bandwidth. We report our experimental results and propose approaches to overcoming the scaling limitations.

1 Introduction

Beowulf clusters [5] are commonly applied to perform computational simulations, where a parallel computer program models a physical process, often utilizing both spatial and temporal partitioning. The use of clusters in this capacity is often straightforward because the programming mechanisms closely mirror those of vendor-supplied distributed memory supercomputers. However, the limitations of the parts incorporated in commodity clusters, especially the network latency, require that some applications be restructured. Nevertheless, a useful range of scientific simulations can be performed effectively on Beowulf systems [6].

The successful application of low-cost cluster computing as a platform for the execution of science codes has led research institutions to explore the use of these systems in other capacities. Up to now, computational science has been the almost exclusive user, but observational and experimental science may be opening a new spectrum of cluster computing applications. Observational and experimental science are generating larger quantities of data every year as more powerful scientific instruments are built. Particle colliders and large telescopes are the principal examples, now capable of generating several terabytes of data a year. As near real-time analysis of data becomes more important to effectively use observational instruments, the ability to move large quantities of data and process them becomes necessary. For example, telescope observation runs can become much more effective if data can be reduced quickly in order to dynamically decide what objects merit closer inspection. Even though clusters can be utilized to simultaneousiy run a coexisting set of applications [3, 1], cursory evidence indicates they are most effective when dedicated to a specific class of task rather than an arbitrary mixture. It is the effectiveness of a dedicated cluster that stands to benefit instrument data analysis.

Of importance equal to rapid data analysis is the processing of the multi-terabyte data archives that accumulate over time. Both of these applications stand to benefit from parallelism, potentially provided by commodity cluster computing. The California Institute of Technology, in conjunction with the European Southern Observatory, is pioneering the evaluation of Beowulf-class computing systems for processing the large volumes of data to be produced by the Very Large Telescope (VLT) in Cerro Paranal, Chile. By 2001, this telescope will generate 20 terabytes of image data per year. This data will be archived and made available to the broader scientific community for online reduction and analysis. We present the results of our initial experiments that attempt to characterize the ability of a Beowulf system to provide dynamic data processing for the high volume astronomical data to be generated by the VLT.

2 ESO Data Archive Workload Synthesis

The ESO data archiving architecture [2] consists of a central data repository, ostensibly a high performance file server, a reduction block scheduler (RBS), and a set of data reduction systems (DRS). The RBS directs the execution of data reduction requests, assigning work to the data reduction systems. Data reduction systems do not share work or communicate with each other, resulting in an almost embarassingly parallel application. We investigated the performance and overheads of using a Beowulf cluster as the host for performing reduction block scheduling and executing data reduction systems. We modified the existing sequential system to perform data reductions in parallel. One cluster node served as the RBS and a set of cluster nodes, varying in size and connected with 100Mbps Ethernet, executed data reduction systems. Each data reduction system is an instance of the ESO Munich Image Data Analysis System (MIDAS), an image processing and data reduction tool that is widely used in the astronomical community [4]. The central data repository was simulated using NFS, which would in practice be replaced with a dedicated high-speed file transfer mechanism.

In order to simulate a realistic computational load, we obtained from ESO a small set of reduction blocks and associated data. The wide range of instrumentation and scientific programs undertaken by ESO makes the definition of a perfectly representative workload problematic. Nevertheless, the set of reduction blocks and data files is approximately representative of a typical workload. The six reduction blocks we studied consisted of one master bias generation, one flat-field generation and four *science* reductions, each of which reads multiple image files and photometry tables, producing a new set of processed images and tables. All input images have dimensions of 2086 x 2046 pixels. Altogther, executing the suite of six reduction blocks causes 328 MB to be transferred from the server to computational nodes, and 314 MB to be written back. A synthetic workload of 30 sets of these six reduction block sequences, totalling 180 reduction blocks and 19 GB, was timed on between 1 and 50 processors of varying types to characterize the performance domain.

We used an existing Caltech Beowulf system for our tests. A diagram of the system is shown in Figure 1. This system consists of a "front end" and an assortment of "compute nodes." The front end is a dual processor 300Mhz PentiumII with 512KB cache and 256MB of memory on a 66Mhz memory bus. It has three SCSI disks — two Seagate "Barracuda 4LP" ST34371W and one "Barracuda 18" ST118273LW. All data for the tests was kept on the larger 7200rpm, 18.1GB Ultra-2 SCSI Barracuda 18 drive. The OS and its utilities are on the Barracuda 4LP drives. All three drives share the same controller, an Adaptec AHA-2940 integrated with the motherboard. Interprocessor communication is through 100baseT Fast Ethernet with a maximum bidirectional throughput of 100Mbps. The front end is equipped with two ethernet cards, allowing it to communicate with two independent networks of computational processors.

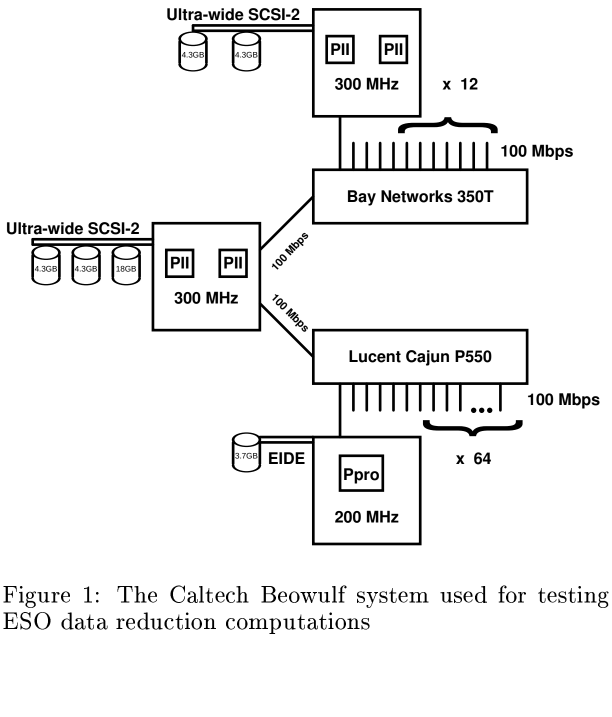

Figure 1: The Caltech Beowulf system used for testing of ESO data reduction computations

Two distinct types of computational nodes were employed in the test. Twelve nodes are identical to the front-end, except that they are not equipped with an 18GB drive and they use only one Fast Ethernet NIC. They are interconnected through a 16 port BayStack 350T 100baseT Ethernet switch. Temporary storage for DRS processes was located on the Barracuda 4LP drive.

The second class of nodes consisted of 200Mhz PentiumPro systems. These were equipped with a 3.1GB, 5200rpm Western Digital WDC AC3100H EIDE drive which holds the OS, utilities and all local temporary files. They were interconnected through two Lucent Cajun P550 switches which can support up to 120 separate 100baseT connections. In our tests, we used up to 50 of these nodes.

All systems were running RedHat Linux 5.2, with various upgrades including version 2.2.5-ac6 of the Linux kernel, version 0.4.22 of kernel NFS support and the egcs-1.1.2 compiler suite. MIDAS was compiled locally using g77 from egcs-1.1.1 and patch levels through patch98NOVpl1.5-1.6.tar.Z.

3 Cluster Performance Heterogeneity

An increasingly common characteristic of clustered computing systems is performance heterogeneity rather than instruction set architecture heterogeneity. Throughout the lifetime of a cluster, new components are added that retain binary software compatibility, but take advantage of price/performance improvements in the marketplace. Consequently, any cluster with an expected lifespan of more than a year or two should be expected to include components with differing performance attributes. We have attempted to quantify this impact on the ESO data analysis tasks by starting with a simple analytic model, hypothesizing that the timing data we obtain should obey a simple relation like:

$$T = \frac{T_1}{P_{\text{eff}}} + c \tag{1}$$

That is, the time to complete some task in parallel should consist of a parallel component which is reduced by the effective parallelism P_{eff} plus a constant overhead, c. The speedup is thus

$$\text{speedup} = \frac{T_1}{T} = P_{\text{eff}} \frac{1}{1 + P_{\text{eff}} c / T_1} \tag{2}$$

which is effectively linear for $P_{\text{eff}} \lesssim \frac{T_1}{c}$.

In practice, we expect the constant overhead to come from the sequential bottleneck imposed by the central file server. In the architecture under investigation, all data files are read and written from a single server. This server has a limited network bandwidth (one or two 100Mbps Ethernet channels, depending on the mix of compute nodes), and even more limited disk bandwidth. Adding additional processors in the configuration of Figure 1 cannot improve the I/O performance. (Other architectures are possible which allow for parallel I/O as well as parallel computation). For a workload of constant size, the server must send and receive a constant amount of data which will add a fixed amount of time to the calculation, no matter how much parallelism is available. If the data are found to obey Eqn 1, then the non-zero constant c imposes an upper limit on usable parallelism. For sufficiently large P_{eff}, the overall time-to-solution is approximately constant, and is unaffected by the use of additional hardware.

N_{DRS}	N_{PII}	N_{SMP}	N_{Ppro}	elapsed
50	0	0	50	129.9
32	0	0	32	185.2
24	12	0	12	166.0
24	12	12	9	197.4
16	0	0	16	271.6
16	8	8	0	267.6
12	12	0	0	238.5
9	0	0	9	428.8
8	8	0	0	326.5
8	4	4	0	478.1
4	4	0	0	613.0
1	1	0	0	2170.0

Table 1: Average time to process the standard suite of six RBs, measured on parallel systems consisting of different mixes of processors. N_{DRS} is the number of DRS processes employed in each test. N_{PII} is the number of Pentium II systems used. N_{SMP} is the number of SMP Pentium II systems executing two DRS processes simultaneously. N_{Ppro} is the number of Pentium Pro systems used.

The value P_{eff} represents the effective parallelism of the Beowulf system. In a homogeneous system comprised of uniprocessors, P_{eff} is equal to the number of processors. In the experimental system, (and very likely in any evolving production Beowulf), one will have a mix of different generations of components, possibly including both uniprocessor and SMP (symmetric multi-processor) nodes. We capture this heterogeneity by defining the effective parallelism as:

$$P_{\text{eff}} = \sum_s N_s q_s \tag{3}$$

where N_s is the number of processors of species s, and q_s is the uniprocessor performance of species s. For SMPs, it is rarely the case that adding a second processor doubles the performance of the system. Contention for limited memory bandwidth is usually the cause. and the phenomenon is incorporated into our model by classifying the second (or more) processor on each SMP node as a different species and choosing an appropriate value of $q_s < 1$.

4 Experimental Results

We timed the execution of 30 independent RB sequences on several different hardware configurations. The results are tabulated in Table 1.

In order to test the validity of Eqn 1, we show a plot $T - T_1/P_{\text{eff}}$ against the number of DRS processes in Figure 2. If Eqn 1 is valid, then the data in Figure 2 should lie in a horizontal straight line. Two adjustable parameters, q_{SMP} and q_{Ppro} allow us to calibrate the performance of the individual processing components. The value $q_{\text{Ppro}} = 0.66$ corresponds to the ratio of clock speeds (200 Mhz vs. 300 Mhz) for the two types of processors. Apparently, differences in cache size and memory architecture between the two systems do not influence the performance of the reduction blocks. The value of $q_{\text{SMP}} = 0.3$ is rather low, but should come as no surprise. In SMP systems only the processor is "doubled." The rest of the system, i.e., the memory bus, PCI bus, hard disks and external I/O remains constant. Contention for these resources make the utility of adding additional processors to the same bus/motherboard an uncertain proposition.

Figure 2 illustrates that the sequential overhead associated with running the set of six RBs is very close to 60 sec. As many as 50 DRS processes were run simultaneously, and the overhead remained nearly constant. This is very good news. It means there are no significant non-linear effects like network contention, disk thrashing or context switching that further degrade overall performance. The I/O system delivers data at a relatively constant rate essentially independent of the number of simultaneously active clients.

Since the same set of RBs execute on a single processor in $T_1 = 2170$ seconds, we conclude that effective parallelism for the astronomical date reductions tasks is up to about

$$P_{\text{eff}} \lesssim \frac{T_1}{c} \approx 35. \tag{4}$$

is supportable by this particular implementation. That is, for $P_{\text{eff}} \lesssim 35$, the total time to solution is dominated by parallel computation. For larger P_{eff}, the total time is dominated by the sequential component, c, and will not be improved by the simply adding more processors.

If we make the reasonable assumption that the sequential bottleneck is due to bandwidth limitations in reading and writing data to the front end's disk, we can estimate the effective, delivered bandwidth. The six RBs cause slightly more than 300MB to be transferred to and from the disk. This suggests that the overall performance of the I/O system, including Ethernet NICs, switches, NFS software, PCI bus, SCSI bus and disk drive is roughly 5 MB/s bidirectional (300 MB in 60 sec). This compares favorably with the theoretical limit of 12 MB/s imposed by 100Mbps Ethernet.

This effective throughput can almost certainly be improved by addressing the individual components. Every factor of two improvement allows for effective utilization of twice as many parallel processors. Even the unoptimized case supports $P_{\text{eff}} \approx 35$, so modest, and very practical improvements in the networking and I/O throughput will enable effective use of parallelism in the hundred processor range.

5 Conclusions

The architecture studied here is a completely standard "base case," with essentially no optimization, tuning or customization. For example, NFS was used for data transfer because of its ease of use, despite its well-known performance drawbacks. Despite these limitations, parallel efficiency is close to 50% on 50 processors. Nevertheless, the next generation of processors is even faster, and processors speeds increase faster than individual disk and network bandwidths. In order to be prepared for the future, we should investigate approaches to reducing sequential bottlenecks today. Indications are that the sequential bottleneck arises from data transfer to and from the server. Opportunities for improvement fall into two categories: improve the performance of the hardware and software components involved in the transfer or reduce the amount of data transferred.

Improved internal bandwidth within server RAID systems offer greatly improved bandwidth to secondary storage by utilizing many disks in parallel. Both hardware and software RAID solutions are known to work with commodity Beowulf systems with bandwidths well in excess of the 5 MB/s seen in our single-disk tests. We have performed preliminary tests using

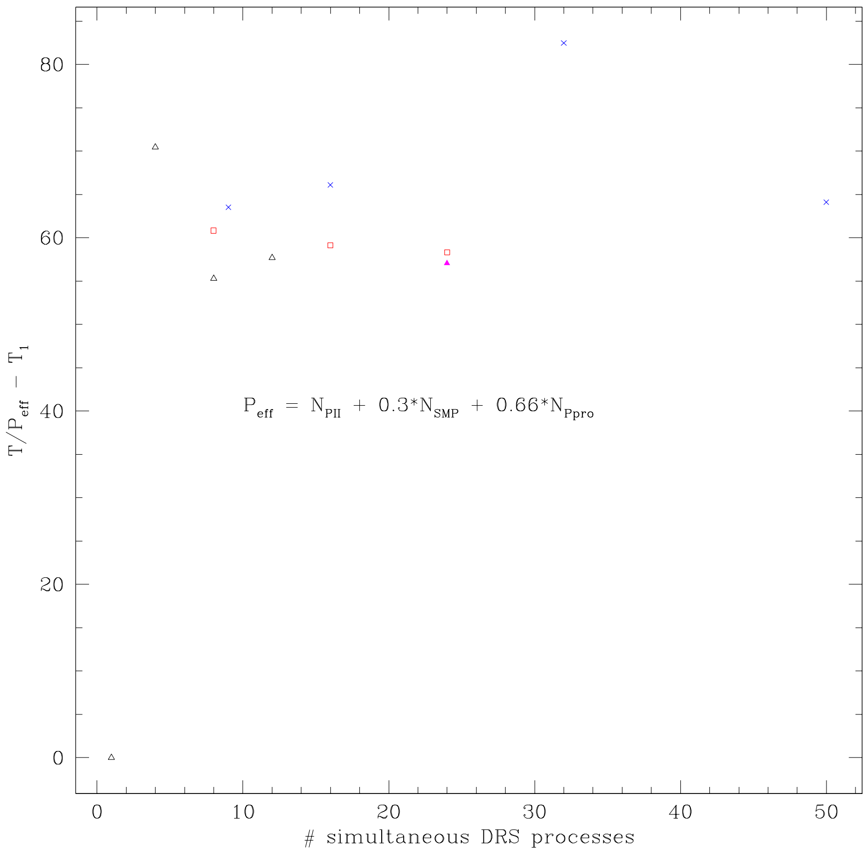

Figure 2: The sequential component C vs N_{DRS}, the number of simultaneous DRS processes for machine configurations and timings in Table 1.

Linux-based software RAID systems incorporating 8 Ultra-2 SCSI drives delivering 80 MB/s reading and 53 MB/s writing.

Improved bulk data transfer protocols Improving the internal data bandwidth is likely to expose the limitations of the NFS implementation. NFS is not particularly well suited the transfer of large chunks of data. Although it performs tolerably, it places a heavy burden on both the network and the server's processor. A connection-oriented protocol like rcp would likely perform better and scale more readily to larger systems.

Gigabit Ethernet into server Improving the internal bandwidth will confront the limitation imposed by the single 100 Mbps link between the server and the compute nodes. In Figure 1, all data to and from the server passes over one (or possibly two) 100 Mbps links. Fortunately, commodity network switches now routinely come equipped with one or two Gigabit Ethernet links to augment the more numerous 100 Mbps links. These links are designed to support an architecture very similar to that shown in Figure 1, but with the links between the server and the switches operating over Gigabit Ethernet at (in theory) ten times the performance of the other links in the system. The switches themselves have more than enough internal bandwidth to keep all the 100 Mbps links fully fed. Gigabit Ethernet is a relatively new technology, and hard numbers are difficult to obtain.

Reusing/caching data files The science RBs tend to reuse the same reference frames over and over. In a job mix with a large fraction of RBs that use reference frames, the overall throughput could be significantly improved if the reference frames were cached on the compute nodes.

Optimal placement of RBs If data is cached, then some RBs will perform better if dispatched to nodes that already have the reference data cached. Currently, the dispatcher ignores this, but the information is available, and could be put to use. This optimization will be important only if data caching is implemented as well.

Local executables One very simple improvement would be to keep executables (e.g., MIDAS) on a local (i.e., non-NFS) filesystem. In the experimental architecture, network traffic is incurred each time MIDAS is started, which occurs for each reduction block. The overhead appears to have been small, but it is easily eliminated in a production system.

Reuse executable In the original RBS, when a DRS process completes one reduction it waits for another set of commands to appear on its input socket. Thus, the cost of loading the DRS is amortized over many reductions. There is, however, the possibility of unintended corruption of internal data structures, memory leaks, and disk leaks. In a well-tested production environment, these risks are low, and it may be worth saving the overhead of loading the DRS for every RB.

Improved bandwidth to nodes As the performance of commodity networks and processors improves, and as the price comes down, it may become possible to use Gigabit Ethernet everywhere. This is a long-term solution, however, since current systems appear to perform very well with only 100 Mbps to each node.

As the ESO Very Large Telescope comes online and starts producing data, it will be possible to better evaluate the impact of the various performance improvement options that will be explored. If Beowulf clusters can prove effective in managing terabyte archives of astronomical data, then it should become possible to apply them to processing other types of instrument data. The first steps being explored by Caltech and ESO involve retrofitting existing data analysis software and adapting general-purpose software such as NFS in an effort to develop a prototype and reach an initial assessment. Future steps will require the identification of those functional characteristics of most critical importance in the underlying system software. The promise of clustered computing for instrument data analysis will demand the development of specialized system software to provide a supporting infrastructure. The development of specialized systems, such as bullkdata transfer protocols, that focus on satisfying the requirements of instrument data analysis should prove broadly applicable and warrants investigation.

References

[1] Acharya, A., Edjlali, G., and Saltz, J. The Utility of Exploiting Idle Workstations for Parallel Computation. In *Proceedings of ACM SIGMETRICS '97* (June 1997).

[2] Albrecht, M., and Quinn, P. ESO VLT Science Archive Research Environment. In *Proceedings of the Workshop on Interfaces to Scientific Data* (1998).

[3] Arpaci, R. H., Dusseau, A., Vahdat, A. M., Liu, L. T., Anderson, T. E., and Patterson, D. A. The Interaction of Parallel and Sequential Workloads on a Network of Workstations. In *Proceedings of ACM SIGMETRICS'95/PERFORMANCE'95 Joint International Conference on Measurement and Modeling of Computer Systems* (1995), pp. 267–278.

[4] Grosbol, P. J., and Ponz, J. D. The MIDAS System. In *Acquisition, Processing and Archiving of Astronomical Images*, G. Longo and G. Sedmak, Eds. OAC and FORMEZ, 1990, p. 109.

[5] Sterling, T., Becker, D., Savarese, D., Dorband, J. E., Ranawake, U. A., and Packer, C. V. BEOWULF: A Parallel Workstation for Scientific Computation. In *Proceedings of the International Conference on Parallel Processing* (1995).

[6] Sterling, T., Savarese, D., Becker, D., Fryxell, B., and Olson, K. Communication Overehead for Space Science Applications on the Beowulf Parallel Workstation. In *Proceedings of the Fourth IEEE Symposium on High Performance Distributed Computing* (1995).