2001-09-01

Improving Cluster Utilization Through Set Based Allocation Policies

Quinn O. Snell
snell@cs.byu.edu

Julio C. Facelli

Brian D. Haymore

David B. Jackson

Improving Cluster Utilization Through Set Based Allocation Policies

David B. Jackson,[&] Brian D. Haymore,[&] Julio C. Facelli,[&,*] and Quinn O. Snell[¶]

[&]Center for High Performance Computing, University of Utah, Salt Lake City, UT, USA.
jacksond@supercluster.org, brian@chpc.utah.edu, facelli@chpc.utah.edu

[¶]Fulton Supercomputing Center, Brigham Young University, Provo UT, USA.
snell@cs.byu.edu

Abstract

While clusters have already proven themselves in the world of high performance computing, some clusters are beginning to exhibit resource inefficiencies due to increasing hardware diversity. Much of the success of clusters lies in the use of commodity components built to meet various hardware standards. These standards have allowed a great level of hardware backwards compatibility that is now resulting in a condition referred to as hardware 'drift' or heterogeneity. The hardware heterogeneity introduces problems when diverse compute nodes are allocated to a parallel job, as most parallel jobs are not self-balancing. This paper presents a new method that allows the batch scheduling system to intelligently select the best resource set for a parallel job in order to minimize the adverse effects of hardware drift and increase overall performance of the cluster. The performance improvements of this technique are evaluated in terms of parallel job efficiency and scheduling resource utilization and overall system performance. Using the emulation capabilities of the Maui Scheduler, this paper evaluates a number of variations of the resource set allocation algorithm on true cluster throughput and utilization using a recorded trace workload from a production cluster.

1. Introduction

Clusters have been accepted as a viable alternative to mainstream supercomputing platforms for a broad subset of compute intensive workloads. The success of clusters has been, in part, due to their use of commodity components, including compute and interconnect hardware. The commodity nature of these components has resulted in inexpensive and standardized building blocks that allow a high degree of backwards compatibility. This compatibility has allowed sites to realize extensive cost savings by extending existing systems rather than requiring an 'en masse' replacement with newer hardware.

While this ability to extend existing systems has resulted in cost savings, it has also resulted in a greater degree of hardware 'drift'. This drift occurs as newer compute nodes are incorporated into an existing system resulting in large performance discrepancies between the system's fastest and slowest nodes. This discrepancy becomes critical to machine utilization when nodes spanning this performance space are allocated to parallel jobs which do not utilize internal load balancing techniques, which is the largest percentage of the parallel jobs running today. These results in jobs constrained to run no faster than the slowest processor allocated, as tasks running on faster processors are forced to wait for intermediate data supplied by the slowest processors. While there has been significant research reporting scheduling techniques to take into account varying hardware architectures [3, 9] and Grid environments [1, 8] much less work has been reported in commodity clusters.

This paper proposes techniques to minimize the negative impact of hardware drift found in heterogeneous clusters by utilizing a simple set based node allocation algorithm for batch schedulers. While the proposed techniques are applicable to any type of hardware drift, including memory and networking based drift, this paper will focus on processor speed based drift.

In batch systems, the class of algorithms known as 'node allocation' algorithms are responsible for determining where a job should be run either now or in the future. The set-based algorithms described in this

* Correspondent Author at 155 S 1452 E Rm 405, Salt Lake City, Utah 84112-0190.

paper select these nodes by determining the set of all nodes which can support a given job, generating common attribute subsets from this set, and then selecting the optimal subset for the job. The effectiveness of this aproach is evaluated using the emulation capabilities of the Maui scheduler [5] using resource and workload traces covering a 30 day period at the University of Utah's Center for High Performance Computing. This algorithm is shown to significantly improve overall system utilization under a variety of scheduling environments and configurations.

The term 'resource set' is used in this paper to describe a set of compute nodes which possess a common set of attributes which impact the ability of the node to perform the work required by a parallel job task. These sets can be delineated along various attribute dimensions such as node processor speed, network interconnect, memory, etc. Nodes within the same resource set should be able to complete a comparable job task in a similar amount of time.

2. Current Approaches

Sites experiencing processor speed drift are currently using one of the following approaches:

- Do not address the problem.
- Partition the system into processor speed based sets. Allowing jobs to run only in one of the available partitions.
- Allow users to specify required processor speed on a per job basis.

The first approach is currently used at most sites because these sites A) do not realize the problem exists, B) do not realize the extent of the problem, or C) have determined that the drawbacks of existing solutions outweigh the benefits. Partitioning the compute nodes along processor speed or other performance boundaries guarantees that jobs will run on a homogeneous set of compute nodes; eliminating the processor speed drift issue. However, this partitioning introduces extensive system fragmentation diminishing many of the price/performance gains acquired by clustering nodes in the first place. The third approach, allowing specification of processor speed by users, allows users with self-balancing jobs to avoid the drawbacks of partitioning. However, for those aware of the processor drift problem, it still effectively partitions the system as users are forced to choose a particular processor speed at submit time. For those users unaware of the problem, no particular processor speed is specified and their jobs will run the risk of spanning resource sets.

This paper proposes an alternative approach containing elements of the third method mentioned above. However, rather than having a user specify a particular node attribute value required by the job, such as 'processor speed=500MHz', we propose using an enhanced scheduler node allocation algorithm which allows the user to specify the node attribute of interest. This algorithm then allows the scheduler to determine the best set of nodes with a common value for this particular attribute at job run time.

For example, when a user specifies that the job requires 16 processors with a common processor speed, the allocation algorithm determines, at run time, the best set of processors meeting these needs. This may be a set of sixteen 500 MHz processors instead of waiting several days for sixteen 700 MHz processors. Like previously described approaches, the algorithm guarantees a set of homogeneous processors that are not subject to the inefficiencies of processor speed drift. However, unlike other methods, this algorithm does not 'artificially' fragment compute resources by requiring a user to select a specific processor speed at submit time.

While it appears intuitively clear that the proposed change will result in better overall system throughput, several key questions remain:

- How much improvement will be realized from the proposed algorithm in real world workloads?
- How much improvement is seen in systems where some form of processor speed partitioning is currently used? What improvements will be seen in systems where no partitioning is used?
- Should the algorithm prevent a job from starting if adequate total processors are available but not enough processors are available within a single processor speed set?
- How should the algorithm select processor sets when more than one set is available? In the specific case of processor speed, should the algorithm always select the fastest processor set? The largest? The smallest?

3. Experiments

The effectiveness of the 'set based' resource allocation algorithm was tested using the workload found on the University of Utah's Center for High Performance Computing Linux cluster, ICEBox [4]. As of May 2001, this cluster consists of 294 processors ranging in speed from 350 MHz to 1.33 GHz. This cluster is made available to a wide user community running parallel scientific codes ranging from QCD to weather modeling. The ICEBox cluster in some ways more closely represents a federation of clusters than a single, large cluster. It is composed of multiple groups of compute nodes dedicated to specific programs for which special QoS (Quality of Service) are established.

A distinguishing characteristic of CHPC is the

allocation tracking system used. In many clusters, users are not 'charged' for jobs or are charged a simple flat rate for CPU time used. Even in the later case, the user is not charged for the resources dedicated, but rather, only for the resources used, resulting in users generally desiring the fastest nodes possible and not caring about job efficiency. CHPC utilizes the Maui scheduler and the Qbank [7] allocation management system enabling them to charge processor speed dependent rates based on compute resources dedicated to the job. Consequently, users are motivated to have their jobs request compute nodes that can be utilized efficiently. This charging approach has resulted in a distinct effort by many users to maximize the efficiency of their runs and has resulted in a high percentage of jobs requesting specific processor speeds to avoid the inefficiencies associated with processor drift.

To conduct these experiments, the emulation capabilities of the Maui Scheduler [6] were used. In emulation mode, the Maui scheduler can replicate the compute resources, batch queue, and scheduling configuration in virtually every detail, allowing highly accurate scheduling performance statistics to be obtained. The results reported here are based on a 6000+ job workload trace representing a 30 day timeframe. Preliminary results in a smaller trace, 10 days, were qualitatively similar indicating that the results reported here are representative of the average workload for the system. The jobs in the trace ranged in size from 1 to 80 processors with jobs requiring less than one to over 40 hours of wall time. Figure 1 shows a size/duration distribution breakdown of these jobs. The current CHPC scheduling configuration was used with two notable exceptions. First, the Qbank allocation system was disabled since its effect was in essence already imposed upon the collected job trace information. Secondly, Maui resource reservation and QoS capabilities were disabled to allow focusing on the impact of the new scheduling algorithm and prevent artificial impact associated with their use. CHPC job prioritization, backfill, fairness, task distribution, and other policies were kept in place.

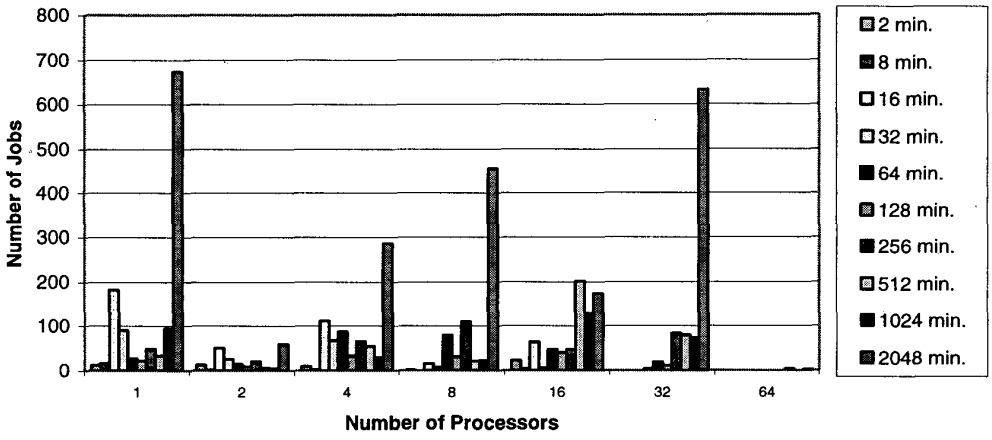

Figure 1: Job Size/Duration Distribution on ICEBox

To test the effectiveness of 'resource set' based allocation algorithms, Maui's 'NODESET' policy was enabled, allowing nodes to be allocated to jobs in sets. A set 'attribute' such as 'node feature', 'processor speed', 'memory', or 'networking interface' may be specified as well as an optional 'attribute subset', allowing the job to provide a list of values for the selected attribute which are acceptable. This feature allows a job to specify that it should run on any set of nodes with a common attribute value or only on nodes sets containing a specified subset of attribute values.

Several simulation runs were conducted. The first experiment attempted to evaluate the effect of resource set count and node performance discrepancy on cluster throughput. The second focused on the value of various methods of handling processor speed drift. Finally, the third and fourth experiments evaluated various adjustments to the resource set based allocation algorithm and their impact on its effectiveness.

For the purpose of these experiments, the emulation was configured assuming that job run times were linearly scalable with the minimum processor speed of the nodes allocated. Hence a job trace recorded for a job running on 500 MHz nodes would have its execution time halved if run on 1 GHz nodes. While it is clear this linear speedup assumption does not take into account

issues such as data and network constraints on job runtime, it does provide a starting point from which to collect information. The fact that these factors are not incorporated into the emulation does not invalidate the collected results but rather cause the emulation to merely somewhat exaggerate the benefits of these algorithms.

The Maui Scheduler provides a suite of statistics that can be used to observe scheduling performance. For this paper, the metrics of scheduling efficiency and average job efficiency are used [2]. Scheduling efficiency is defined as the ratio of resources dedicated to jobs divided by the total resources available. Average job efficiency, as reported by Maui, is defined as the ratio of resources utilized by jobs divided by the total resources dedicated to jobs. This definition provides a job size-duration weighted job efficiency value.

Experiment 1: *Effects of Resource Set Count and Performance Disparity on Average Job Processor Efficiency.*

In this experiment, the impact on performance due to changes in the hardware, or number of processor generations existing within a cluster, was evaluated. To evaluate resulting job efficiency, the CHPC workload was applied to two emulated clusters; the first representing the state of the ICEBox cluster shortly after its initial build and the other representing its current state as of May 2001. When initially used in production, ICEBox consisted of only three types of processors ranging in speed from 350 MHz to 550 MHz. Within 2 years, this cluster had evolved to include 9 distinct processor speed sets ranging in speed from 350 MHz to 1,330 MHz.

In each emulation, identical scheduler policies were applied and the simulator was configured to scale the completion time of all jobs by the speed of the slowest processor allocated to the job.

The results indicate that the addition of distinct processor sets resulted in nearly 15% lower average job efficiency. While the addition of nodes in the current cluster allowed the scheduler more nodes to choose from, resulting in slightly higher scheduling efficiency (~2.5%), this was not enough to compensate for the job efficiency losses. The overall efficiency of the cluster (scheduling efficiency * job efficiency) actually dropped as more nodes were added from 92.1% to 80.3%. Detailed analysis of the simulation indicated that both the percentage of 'mixed node' jobs and the average efficiency losses per job increased as the cluster evolved.

Experiment 2: *Impact of Various Processor Speed Specification Approaches on Scheduling, Job and System Performance.*

This experiment evaluated the performance impact of different processor speed specification mechanisms. All utilized the current ICEBox node configuration and workload described in Experiment 1. In the first test run, the scheduler ran through the workload using its default behavior. In the second run, the scheduler was instructed to ignore processor speed based node feature requests. In the third run, all jobs that specified a specific processor speed were modified to instead specify a processor set. In the fourth run, all jobs were required to utilize processor sets regardless of whether or not they requested specific processor speeds in the initial job trace.

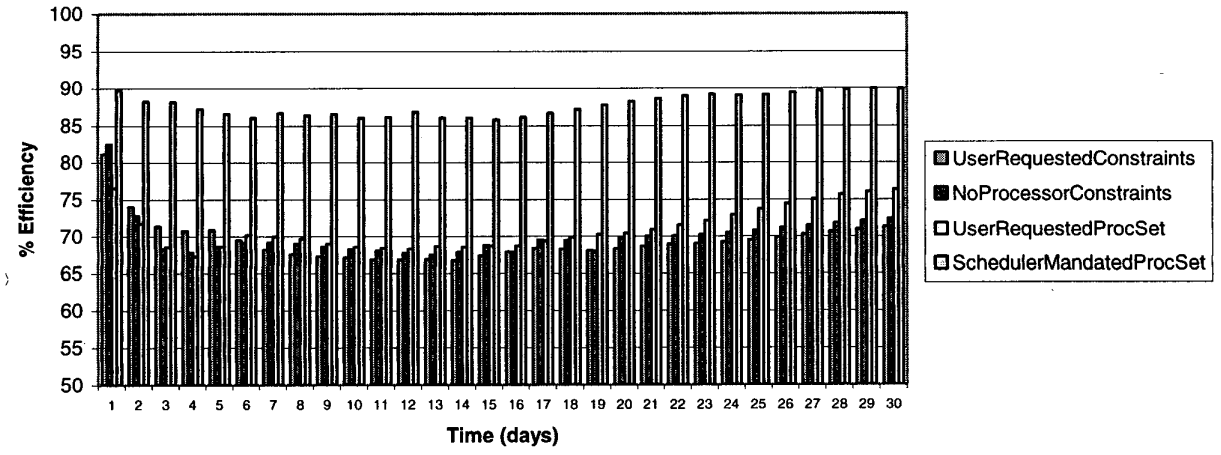

Figure 2: Overall System Efficiency (Scheduler Efficiency * Job Efficiency)

Figures 2 shows the resulting cumulative system efficiency (Scheduler Efficiency * Job Efficiency) evaluated at 24-hour intervals. The mandated resource set based approach clearly outperforms the specific processor speed request in all simulations, with a minimum overall system efficiency improvement close to 20%. While the

improvement of job efficiency is obvious for the mandated resource set scheduling, the improvement of the total system efficiency is not intuitively obvious. Because in this case the scheduler has much less freedom in its scheduling decisions and thus it may not be able to use available resources effectively. The results show that the scheduler efficiency for the first three cases, with non-mandated resource set is approximately 97%. When resource set scheduling is mandated this efficiency decrease to approximately 87%. This decrease in scheduler efficiency is greatly compensated by the increase in job efficiency that increase from less than 70% for the non mandated cases to 100% when the resource set requests are mandated. This situation leads to the significant overall system performance increases depicted in Figure 2.

Experiment 3: *Scheduling Efficiency Associated with Various Set Selection Approaches.*

In many cases, the scheduler will have an option of more than one suitable resource set on which to run a given job. Three possible algorithms were evaluated:

- Select the fastest processor set available (BestResource).
- Select processors within the largest available set (WorstFit).
- Select processors within the smallest available set (BestFit).

In this experiment, the workload with mandated resource set was applied to the current cluster system. Identical scheduling policies, with the exception of the resource set selection policy, were used in the simulations. Figure 3 shows resulting scheduling efficiency measured at 24-hour intervals. Note that in these experiments all jobs run on homogeneous sets of processors resulting always in 100% job efficiencies. Thus the data of interest is the scheduler efficiency (i.e., how well the algorithm maximizes the number of remaining feasible resource sets for the scheduler to utilize in the future).

Figure 3 shows that the BestFit algorithm outperforms the other two algorithms considered. While the success of this algorithm was anticipated, the size of the improvement was surprising, significantly leading the field both in terms of scheduling efficiency and more importantly in job throughput.

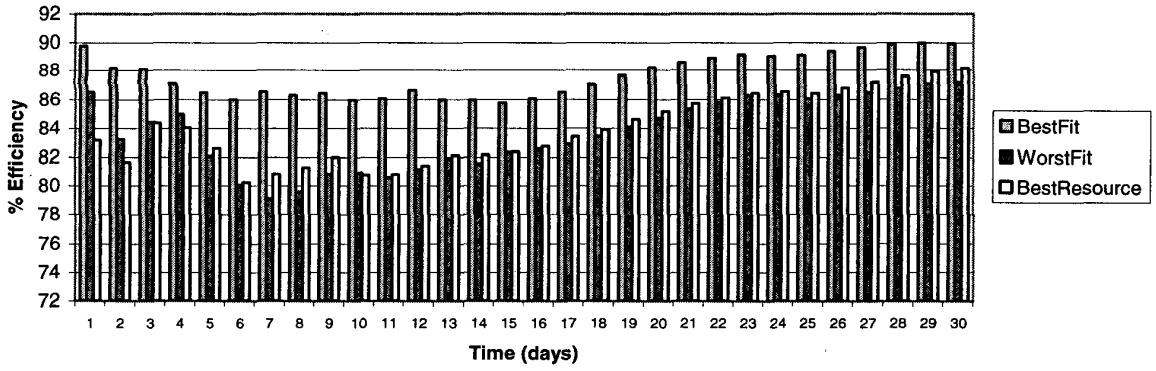

Figure 3: Cumulative Scheduling Efficiency for Various Set Selection Algorithms.

Experiment 4: *System Efficiency for Best Effort Vs Forced Set Based Resource Allocation.*

The use of set based resource allocation is clearly beneficial when adequate homogeneous resources are available. However, delaying jobs, which could run on a heterogeneous processor mix, until a later time at which a complete homogeneous set is available may not always be the best approach. In this experiment, the CHPC job trace is applied to the current ICEBox cluster under a new 'best effort' approach. In this algorithm, the scheduler will allocate homogeneous processor sets whenever possible. However, if such a set is not available but a heterogeneous set is, the algorithm allocates the heterogeneous resources. The results of this test are compared to a standard 'forced' set algorithm in which jobs must always run on homogeneous processors sets. Figure 4 shows the resulting system efficiency.

Surprisingly, at no point does the best effort algorithm achieve the system efficiency found in the forced set based algorithm. While the best effort approach appears to be a good idea on the surface, it suffers from the same resource fragmentation issues exhibited in the 'User Specified Processor Set Request' test run of Experiment 3. The high scheduling efficiency allowed through the best effort scheduling algorithm results in highly fragmented resource sets. Thus, on the

occasions when homogeneous resource sets cannot be found, the available resources are scattered across multiple processors sets resulting in low efficiencies for the jobs allocating these resources.

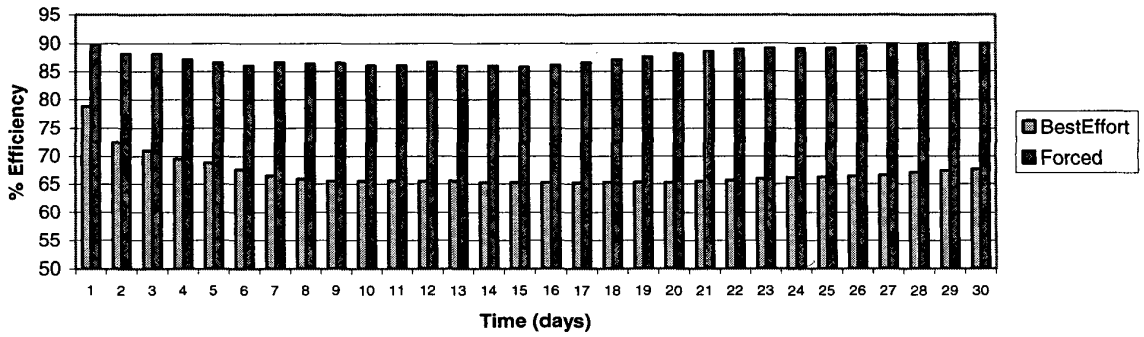

Figure 4: System Efficiency for Best Effort Vs Forced Set Based Resource Allocation

4. Conclusions

Linux clusters are highly efficient and cost effective in the HPC arena. However, over time, these systems evolve to possess multiple generations of hardware. This leads to significant wasted resources under batch systems without resource set aware algorithms. Resource set aware node allocation algorithms can avoid these potential inefficiencies by allocating homogeneous resource sets as needed to non-load balancing parallel jobs. The resource set allocation policies currently in place in the Maui Scheduler can significantly outperform non-resource set allocation policies in terms of true system utilization and job throughput.

While this paper showed that significant throughput and system efficiency gains can be accomplished by way of a resource set based allocation algorithm, further gains are still anticipated with added algorithm sophistication. Further research would be valuable in terms of validating these algorithms on different hardware configurations and under different and more extensive workloads. Finally, extending the study to research the use of resource sets based on non-processor based attributes such as node memory or network adapters would be of great value to sites running a high percentage of memory and/or communication bound jobs.

5. Bibliography

[1] F. Berman, "High-Performance Schedulers", in *The Grid*, I. Foster and C. Kesselman (eds.), Morgan Kauffmann, San Francisco, 1999.

[2] D. G. Feitelson and L. Rudolph, "Metrics and Benchmarking for Parallel Job Scheduling". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), Springer-Verlag, 1998, Lect. Notes Comput. Sci. vol. 1459, pp.1-24.

[3] R. F. Freund *et al.*, "Scheduling Resources in Multi-User, Heterogeneous, Computing Environments with SmartNet". Heterogeneous Computing Workshop, International Parallel Processing Symposium, 1998.

[4] B. D. Haymore, ICEBox Overview, Technical Report, CHPC, University of Utah, 2000, Available at http://www.chpc.utah.edu/ice.

[5] D. B. Jackson, Maui Scheduler Homepage, Technical Report, Supercluster.org, 2001, Available at http://www.supercluster.org/maui.

[6] D. B. Jackson and Q. O. Snell, "Emulation Based HPC Scheduling Analysis". Proceedings of the International and Parallel Distributed Processing Symposium, 2001.

[7] S. Jackson, Qbank Overview and Documentation, Technical Report, EMSL, Pacific Northwest National Laboratory, 2000, Available at http://www.emsl.pnl.gov:2080/docs/mscf/qbank-2.8.

[8] M. Livny and R. Raman, "High-Throughput Resource Management", in *The Grid*, I. Foster and C. Kesselman (eds.), Morgan Kauffmann, San Francisco, 1999.

[9] M. Maheswaran and H. J. Siegel, "A Dynamic Matching and SchedulingAlgorithms for Heterogeneous Computing Systems". Heterogeneous Computing Workshop, International Parallel Processing Symposium, 1998.