4-2008

Capacity Planning of a Commodity Cluster in an Academic Environment: A Case Study

Linh B. Ngo
Clemson University, lngo@clemson.edu

Amy W. Apon
Clemson University, aapon@clemson.edu

Baochuan Lu
University of Arkansas - Main Campus

Hung Bui
University of Arkansas - Main Campus

Nathan Hamm
Vanderbilt University

See next page for additional authors

Capacity Planning of a Commodity Cluster in an Academic Environment: A Case Study

Baochuan Lu[1], Linh Ngo[1], Hung Bui[1], Amy Apon[1],Nathan Hamm[2], Larry Dowdy[2], Doug Hoffman[3] and Denny Brewer[3]

[1] Computer Science and Computer Engineering, University of Arkansas
Fayetteville AR 72701, USA
[2] Electrical Engineering and Computer Science, Vanderbilt University
Nashville TN 37240, USA
[3] Acxiom Corporation
Conway AR 72032, USA

Abstract. In this paper, the design of a simulation model for evaluating two alternative supercomputer configurations in an academic environment is presented. The workload is analyzed and modeled, and its effect on the relative performance of both systems is studied. The Integrated Capacity Planning Environment (ICPE) toolkit, developed for commodity cluster capacity planning, is successfully applied to the target environment. The ICPE is a tool for workload modeling, simulation modeling, and what-if analysis. A new characterization strategy is applied to the workload to more accurately model commodity cluster workloads. Through "what-if" analysis, the sensitivity of the baseline system performance to workload change, and also the relative performance of the two proposed alternative systems are compared and evaluated. This case study demonstrates the usefulness of the methodology and the applicability of the tools in gauging system capacity and making design decisions.

1 Introduction

Performance engineering includes performance analysis, performance tuning, and capacity planning. It should be an integral part of the development and operation of any large computer system. Commodity clusters are special large scale distributed systems that pose unique challenges to performance engineering, largely due to their multi-level workloads, shared services, and intricate dependencies among interacting components. Inadequate capacity planning (e.g., designing insufficient resources) can have a significant impact on system performance. In contrast, understanding and taking full advantage of the dynamic nature of cluster systems where resources are allocated and db-allocated dynamically can help to alleviate potential performance bottlenecks. Therefore, it is imperative to study the performance of commodity cluster systems and to develop rules and techniques that will guide planning, management, and operation of these systems.

The Integrated Capacity Planning Environment (ICPE) toolkit is used to address the capacity planning needs of large-scale distributed system, such as commodity cluster systems. The components of this toolkit include trace analysis, workload characterization, simulation, and animation [1]. The purpose of trace analysis is to gain insight about workload patterns, including workload trend analysis as a function of time. The workload characterization module uses predefined methods to characterize jobs from the actual measurement trace into job classes. These job classes provide a concise, representative model of the workload that is used by the simulation engine. As a generalization and simplification of the actual workload, the workload model can be modified easily to represent futuristic workloads which, along with system configuration parameters, constitute what-if scenarios. The simulation engine is used to answer what-if questions by simulating what-if scenarios.

In this paper, the ICPE methodology and tools are applied to a specific distributed system, the University of Arkansas Red Diamond supercomputer system, which runs primarily scientific computation jobs from two major science departments, chemistry and physics. The workload includes long serial jobs, large parallel jobs, and other miscellaneous jobs that are relatively small and short. When a job arrives, it request a number of processing units (i.e., CPU cores) which are allocated and used until the job finishes execution. Since funding sources are different for the two departments, jobs are divided into groups according to the user department, and the scheduling policy is set so departments have a fair share of the Red Diamond resource and usage is allocated proportional to the funding over time.

Two alternative systems are proposed to significantly enhance Red Diamond's capacity. To make an informed purchasing decision, the performance of the two hypothetical systems must be evaluated and compared under projected future workloads. Section 2 discusses issues workload modeling through trace analysis. The system simulation model design is presented in Section 3. In the end, the design and the results of various what-if analyses are illustrated in Section 4.

2 Trace Analysis and Workload Modeling

The performance of a computer system is a function of its underlying hardware and the workload it executes. The goal of workload modeling is to build models of the real workload that are detailed enough to be representative of their impact on system performance and are simple enough to be manipulated easily in order to model hypothetical workloads. Therefore, jobs are characterized into a small number of job classes, each capturing the primary characteristics of specific, identifiable segment of the workload. Identifying these job classes is a challenge, since few actual systems are similar to each other.

To build realistic workload models the real workload must be measured and analyzed. In the target system, jobs submitted to the system are measured and recorded in a database table, which is considered a trace of the actual work-

load. Each row of the table contains attributes of a job such as its arrival time, start time, end time, user group, and nodes requirement. The number of nodes required by a job is defined as its job size. The particular trace under study in this case study is recorded from the Red Diamond supercomputer. The trace includes jobs submitted from August 2007 to November 2007. The number of nodes in each of the 4471 jobs ranges from 1 to 64. In a previous case study, a trace analysis module of the ICPE tool was created for analyzing the Acxiom enterprise grid. In order to apply this tool to the new data presented here, the job trace is converted into standard trace format (STF) [2].

The primary goal of trace analysis is to gain insight about the workload and its characteristics. Of particular interest are those time periods that demonstrate high workload variability, measured in terms of the job's runtime distribution or the job's arrival process. High variability negatively impacts the system performance. The trace analysis module of ICPE is capable of generating various graphs from a given trace. Representative graphs are presented here.

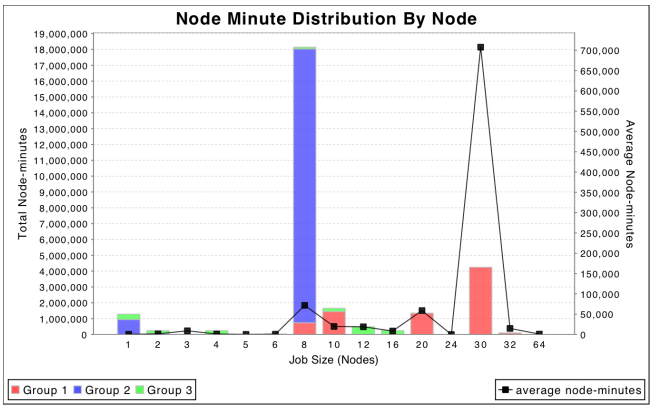

Fig. 1: Load Distribution among Node and User Groups

Figure 1 shows the resource demand ($nodes \times runtime = total\ CPU\ time$) from different job sizes with the demand color-coded according to user groups. Jobs of the same size require the same number of nodes. User group information (e.g., chemistry, physics, other) of a job is determined by the group ID of the job. As shown, different user groups show distinctive resource demand patterns. Overall, group 3 (i.e., other) jobs places the lowest demand on the system and requests nodes ranging from 1 to 16. Group 2 jobs (i.e., physics) place the highest demand, but mainly require one or eight nodes. Group 1 jobs (i.e., chemistry) require a large number of nodes, from 1 to 32, and consume about 1/3 of the overall resource. Figure 1 also shows that if the resource demand of each job is evaluated individually, jobs requiring 30 nodes have the highest average resource demand (second y axis). This observation demonstrates that jobs from different

user groups place distinctive resource demands on the system, which further justifies the decision to classify jobs based on user groups. In addition, users in the target system are most interested in their own user group's performance metrics.

The workload model that is developed not only classifies jobs into groups and classes according to the user group and number of nodes selected, but also describes the job demands on the system in terms of inter-arrival times of the job stream and runtimes for each of the jobs. A typical workload model includes the inter-arrival time distribution, the runtime distribution, and node requirements for each job class. A negative exponential distribution is commonly used as the statistical distribution model for inter-arrival time and runtime with a mean equal to that from the measurement. An exponential distribution is often assumed in analytic queueing models because the mathematical properties of the model can be provably solved with that assumption. However, performance models that rely only on exponential distributions for the inter-arrival and runtimes are often inaccurate because the real workload distributions are not exponential, but often are heavy-tailed, or self-similar. The measured workload for Red Diamond is analyzed to determine the statistical properties of the inter-arrival and runtime times.

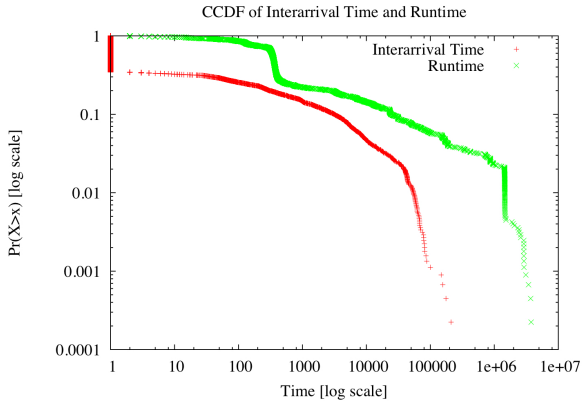

Fig. 2: Log-log CCDF of Inter-arrival Time and Runtime

Figure 2 shows the log-log complementary cumulative distribution functions of the runtime time and inter-arrival time distributions. In the figure, there is a linear region toward the end that spans several orders of magnitude in the runtime figure indicating that the runtime distribution is somewhat heavy-tailed. This region occurs on the graph between time values of 400 and 1000000 for run-

times. Although no such linear region is observable in the inter-arrival curve, the distribution is nevertheless skewed, with the mean inter-arrival time being much smaller than its standard deviation. This observation is also shown in Table 1. Based on this observation, the per-class runtime distribution is not modeled well by an exponential distribution. A two-stage hyper-exponential distribution is selected for the model of the per-class runtimes.

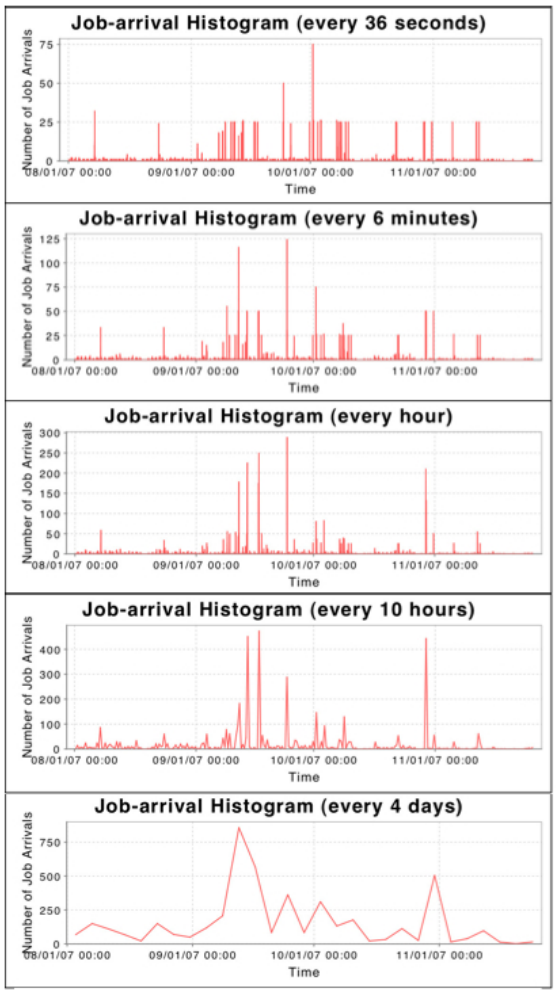

Fig. 3: Burstiness of Arrivals at Different Time Scales

The arrival process is also analyzed for its burstiness, or self-similarity. Large fluctuations in the load indicate a heavy-tailed distribution, similar to what is found in the runtime distributions. To test for self-similarity, the job arrival

Parameters	Class 1	Class 2	Class 3
IAT	16 hr	3.9 hr	45.4 min
IAT stdv	27 hr	13.9 hr	3.8 hr
RT	2 day	3.6 day	2.6 hr
RT stdv	5.5 day	7.8 day	28.1 hr
Node	1:21% 2:1%	1:74%	1:83%
Requirement	4:7.5% 6:1%	2:0.1%	2:4.45%
	8:25%	8:25%	4:5.5%
	10:18%		5:0.4%
	12:0.5%		6:1.5%
	16:3%		8:2.2%
	20:13%		10:1.6%
	30:4%		12:0.7%
	32:3%		16:0.6%
	64:3%		32:0.05%

Table 1: Workload Parameters

histogram is plotted using different levels of aggregation (i.e., various bin sizes). Figure 3 shows job arrival histograms with bin sizes that cover five orders of magnitude from the same Red Diamond measurement log. In each sub-figure of Figure 3 there are definitely observable "peaks" in the data, and this is as true for the small bin size of 36 seconds as it is for the large bin size of every 4 days. This burstiness of arrivals using different scales indicates that self-similarity exists. This burstiness implies that job arrivals are not independent (i.e., that the probability of many jobs arriving practically at once is non-negligible) [3]. A daily cycle pattern is obvious from the hourly job arrival histogram shown in Figure 4. Due to this observed burstiness, job inter-arrival distribution is also modeled by hyper-exponential distribution. The daily cycle pattern is modeled by a slot weight method [4]. This method adjusts the generated arrival times so that the job arrival histogram follows the same pattern as in the actual trace. Experiments show that the slot weight method is effective in increasing the burstiness of arrivals, but its effect on the inter-arrival distribution requires further study.

Figure 5 shows the daily arrival histogram (per hour), which is calculated by averaging over the complete trace. As shown, job arrivals during both weekdays and weekends exhibit clear patterns, suggesting that the arrival process is non-stationary, meaning that the probability of an arrival is dependent on the time that the event occurs. One approach to modeling this behavior is to use a non-stationary Poisson process. The steady-state cyclical properties can be studied for each cyclical period [5]. A second approach is to model the daily cycles directly using heuristic, rather than statistical techniques. In this case study, the slot weight method is selected. The slot weight method is a heuristic technique and practical technique that is intuitive and easy to implement [4].

The slot weight method reproduces non-stationary arrival processes as follows. First, the job arrival histogram is found by placing the arrival times into intervals for a selected interval size. The number of job arrivals is counted in

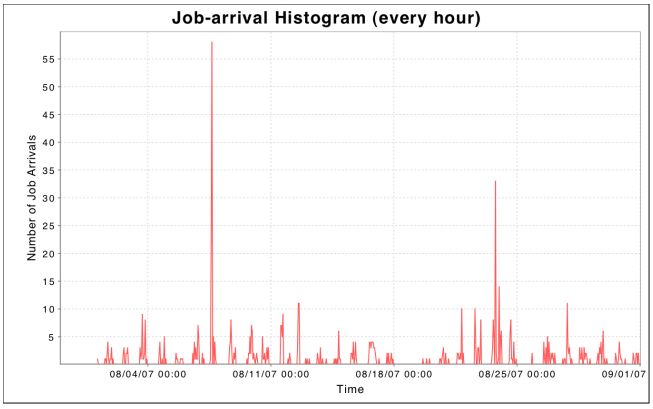

Fig. 4: Daily Cycles in a Week

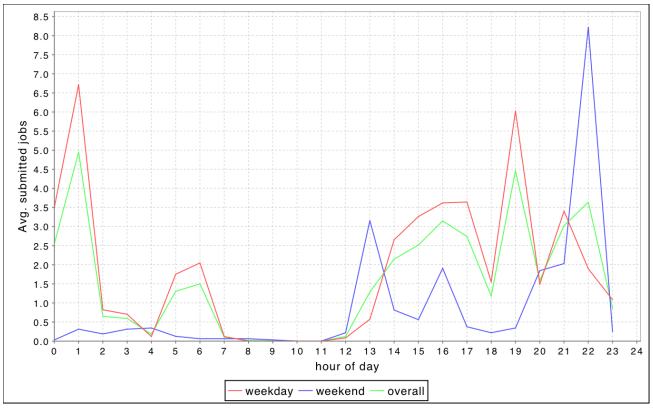

Fig. 5: Daily Arrival Pattern

each interval. This job arrival histogram is used to represent the density function of the empirical distribution of job arrivals. In this case study the interval size selected is one hour for each day, because the patterns are observed for a daily period in the measurement data. The slot weight method in this case computes a 24-valued vector where each value corresponds to one hour of the daily cycle. The resulting 24-valued vector is used by the slot-weight method [4] to reproduce the arrival pattern within a synthetic workload, which is then used as input to the simulation engine.

In our previous case studies, jobs are first divided into classes based on the number of nodes (the size of the job), and then the runtime of each class is calculated secondly, which handles any potential correlation between job size and runtime. In this case study, however, jobs are first categorized according to its user group in the Red Diamond supercomputer because user group based performance metrics are of more interest. The drawback of this approach is that the job sizes of each resulting job class can span a large range as shown in the last row of Table 1. Modeling the job size of each job class by a number (i.e., the mode of the samples) is inadequate. Two possible solutions to this problem have been evaluated. First, each user group based job class can be further divided into subclasses according to the job size. The drawback of this approach is that it can complicate the workload description by significantly increasing the number job classes. For example, if there are three job classes (i.e., chemistry, physics, and other) and each requires six different numbers of nodes, the final model would contain 18 job classes. This not only complicates the workload model for what-if analysis, but also makes modeling the arrival time and runtime distributions more difficult because the number of job samples in each class becomes smaller, which limits the accuracy and confidence within each class.

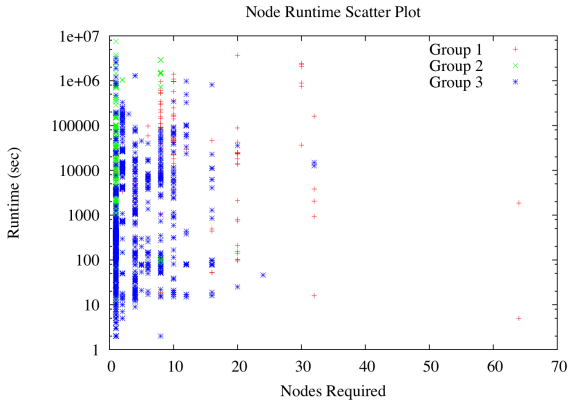

Fig. 6: Correlation Between Node and Runtime

Another approach is to model the node requirement with an empirical distribution as shown in the bottom section of Table 1. However, if the runtime and the size of a job are correlated, it must be modeled explicitly. To examine the correlations between runtime and node requirement, a scatter plot between the runtime and the node requirement is generated and shown in Figure 6. Though no strong correlation is obvious, further analysis is required. If the runtime and the job size are directly related, it can be modeled through correlation functions between the two attributes. Sensitivity analysis of the system performance to the correlation function parameters then can be used to evaluate the effect of the correlation on system performance. If the effect is significant, the correlation function must be used as an input model parameter.

In the current workload model, correlation between a job's runtime and its node requirement is not modeled. Instead, the node requirement of each job class is represented by an empirical distribution functions and the job class' runtime is modeled by an independent two stage hyper-exponential distribution with parameters derived from actual measurements.

Table 1 summarizes the baseline workload model parameters including the average inter-arrival time (IAT), average runtime (RT), standard deviations (IAT stdv and RT stdv), and node requirement (job size) for all three job classes. The node requirement of each class is specified as an empirical distribution with a notation of nodes:percentage. For example, 1:21% indicates that in class 1 20% of class 1 jobs require one node. In contrast, 77% of class 2 jobs require one node.

In summarize, the resulting workload model consists of three job classes each corresponding to a user group in the Red Diamond supercomputing. Within each class, the job arrival rate and service rate are modeled by two stage hyper-exponential distributions. The parameters are derived from the actual workload trace by matching the first two moments of the measurement using Morse's method [6]. The node requirement of each class is modeled by an empirical distribution shown in Table 1.

3 Simulation Design

In the ICPE, a discrete-time, discrete-event simulation model is used to represent the target system. Figure 7 shows a conceptual diagram of the simulator. The model is implemented using Java Simulation Library (JSL) [7]. This model supports both trace-driven simulation and distribution-driven simulation. For the trace driven simulation, jobs are read from a trace file of a real system. In this case study, the standard trace format (STF) used in the workload characterization component is used as input to the simulation model. For the distribution driven simulation, synthetic jobs are generated from the job classes from the workload characterization. Both trace driven and distribution-driven simulations are supported without changing the core simulation model.

In the simulation model, each job has an arrival time, a required number of nodes, and a runtime. The job scheduler is a separate simulation process that is responsible for scheduling jobs in the job queue. Based on the number of

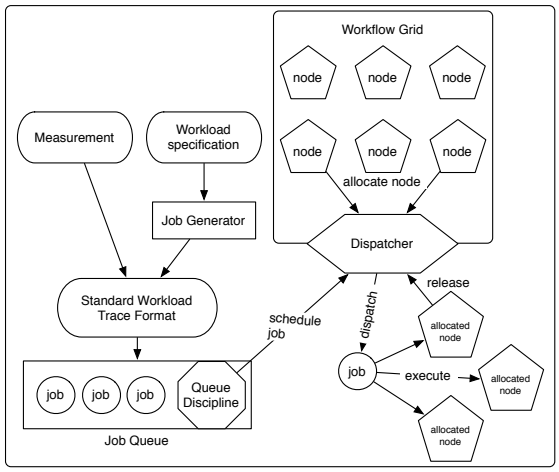

Fig. 7: Architectural View of Simulator

currently available nodes in the system, the number of requested nodes, and the scheduling policy, a decision is made by the simulator regarding the launch time of each job. If the system has enough free nodes, these nodes are allocated to the job and the job is launched, provided it is the next job in the queue according to the current scheduling discipline. Otherwise, the job waits in the job queue. Whenever the system state changes (e.g., when a job completes or arrives), the scheduler process is invoked and seeks to schedule as many jobs as possible. The scheduling discipline is specified as a parameter to the simulator. The scheduling discipline used is first-come-first-with-fillin (FCFSFILLIN), which allows a later arriving job with a smaller number of required nodes to leap-frog in front of an earlier arrived job that is waiting for a larger number of required nodes. This discipline yields more opportunistic results than the queueing discipline employed in the real system, which is a fair-share policy according to preset rules for balancing the usage among different user groups. Thus, it is expected that the simulated results provide an upper bound on the performance of the actual system.

Each node is modeled as a separate simulation process. If a node has a job assigned to it, it remains dedicated to the job for the lifetime of the job (i.e., for an amount of time equal to the service time specification of the job). When a node finishes processing a job, it disassociates itself with the job and releases itself back to the free node pool. Because each node is simulated individually, modeling nodes with heterogeneous characteristics (i.e., different processor speeds or memory sizes) is straightforward. The output of the simulation is a trace file containing statistics of each simulated job. Because this trace file is in the same STF format, the same graphing and visualization tools are available as in the preliminary analysis phase. Statistics for each of the job classes (e.g., queu-

ing time, response time) and performance metrics for system components (e.g., throughput, utilization) are derived by post processing the simulation trace files.

The ICPE simulator takes the following parameters: workload model, number of nodes in the system, queueing discipline, simulation length, number of replications, and the transient period length. The simulation length and the number of replications needed are determined by pilot studies. The underlying random number generator is an industrial strength generator developed by LEcuyer [8] and is now used in many commercial simulation packages. The random number generator provides a practically unlimited number of independent random number streams and handles the synchronization of random numbers between replications within a simulation. In other words, the set of random streams is automatically advanced between a replication to the next, which ensures no sub-streams overlap, i.e. the sub-streams are independent. As a result no seed is needed for running simulations.

4 What-If Analyses and Results

Two types of what-if scenarios are analyzed. The first type focuses on the original system with the baseline workload and studies the effect of increasing the arrival intensity of each individual workload class. The baseline model is also modified to properly load the hypothetical systems to study their relative performance under various loading conditions.

In the first case, the intensity factor of each individual user groups is varied from 1.2 to 2.0 to study its effect on per user group performance. The results are shown in Figure 8, Figure 9, and Figure 10. Note that the scale of the y-axes of the figures are different. Figure 8 shows that an increase of arrival intensity for class 1 has more impact on its own performance than on the other two groups. A 100% arrival intensity increase of class 1 results in an almost 7-fold increase in class 1's average wait time. Figure 9 shows that the arrival intensity increase of group 2 has a dramatic impact on group 1. Doubling the class 2 intensity causes the class 1 average wait time to increase about 26 times. However, increasing group 3's arrival intensity has almost no effect on any class until the level goes up to 2.0, as shown in Figure 10. This indicates that class 1 is the most sensitive and that class 3 is the least sensitive with respect to the workload arrival intensity.

number of cores	avg. clock speed
256	3.2 GHz
1184	2.76 GHz
1256	2.66 GHz

Table 2: Two Alternative Systems

The second set of what-if analysis involves two hypothetical systems considered for purchase, a system with 1184 cores, and a system with 1256 cores.

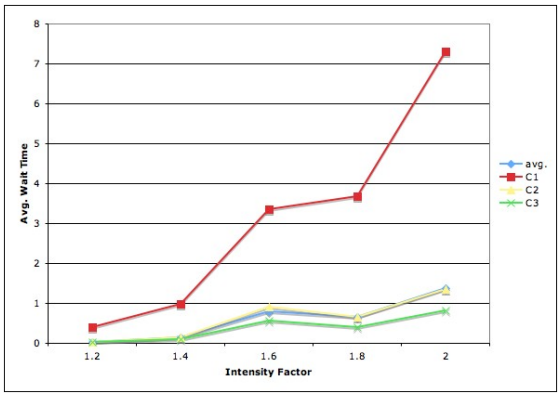

Fig. 8: Increase Class 1 Arrival Intensity

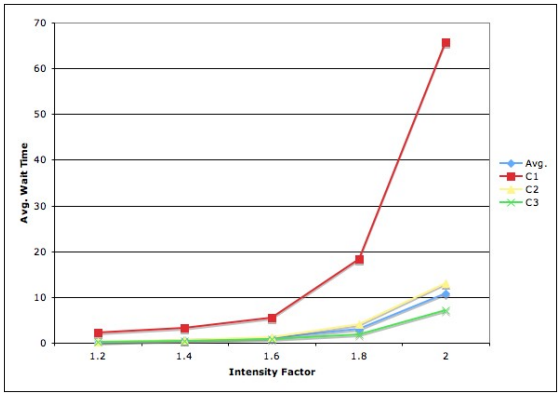

Fig. 9: Increase Class 2 Arrival Intensity

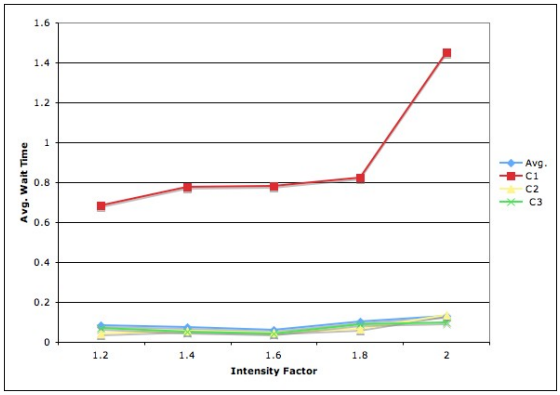

Fig. 10: Increase Class 2 Arrival Intensity

Table 2 lists the number of cores and the average CPU clock rate of the current system and the two hypothetical system configurations. The 1256-core system consists of 1256 cores all running at 2.66GHz. The 1184-core system consists of 1024 cores running at 2.83 GHz and 160 cores running at 2.33GHz. The average CPU clock rate of 2.76GHz is used for easy comparison for the 1184-core system. As shown in Table 1, the 256 cores is the baseline system, which has the fastest CPU clock rate. The other two hypothetical systems are much larger (more than four times) than the original systems but with slower clock rates. In the two hypothetical systems, the 1184-core one contains fewer cores but its CPU clock rate is slightly higher. If the number of cores times the average CPU clock rate is used to estimate the overall system capacity, the theoretical performance of the 1184-core system can be calculated based on the number of cores, the clock speed, and the number of floating point operations per cycle, which is 4 for this architecture. The calculated peak performance of the 1184 core system is $1184 \times 2.76 \times 4 = 13.07$ Tflops. The calculated peak performance of he 1256 core system is $1256 \times 2.66 \times 4 = 13.36$ Tflops. However, because of the intricacy of the systems determining the performance difference between the two systems is not straightforward. Therefore, simulations are used to compare the relative performance of the two hypothetical systems by subjecting the two systems to the same workload model.

As this future system will be used by the same research community, it is reasonable to assume that the future workload characteristic will resemble that of the current workload. Therefore, the baseline workload model derived from the current workload measurement is used to approximate the projected workload conditions. The current measured workload is characterized into 8 job classes each with different nodes requirement and hyper-exponential inter-arrival time and runtime. The intensity factor for job arrival rate is adjusted correspondingly to properly load the system. The service time acceleration factor is chosen to account for the different clock speed of the hypothetical systems and the fact that their CPUs conduct four floating points per cycle as supposed to two per cycle in the current system. Through experiments, the simulation length and number of replications are determined to be 240 days and 100 respectively. Therefore, each simulation experiment is run 100 replications, and each performance metric is calculated along with a 95% confidence interval.

The performance from the two alternative systems is simulated and compared. Figure 11 shows the average queue time of the two systems with 95% confidence interval when the arrival intensities of all job classes increase uniformly by a factor from 5 to 9. As shown, the 1256-core system consistently yields shorter wait time than the 1184-core system and the gap grows increasingly larger as job intensity increases. Figure 12 shows the system utilization of the two hypothetical systems under increasing workloads. As shown, the 1256-core system is always less loaded than the 1184 system.

Another metric studied is the job response time, which is defined as the total wall-clock time from when a job is submitted until it finishes execution. Response time has two components: 1) runtime, during which the job is actually

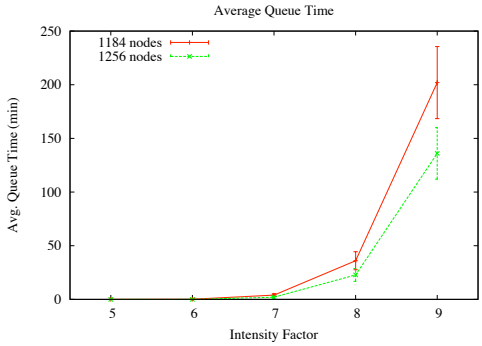

Fig. 11: Average Queue Time Comparison

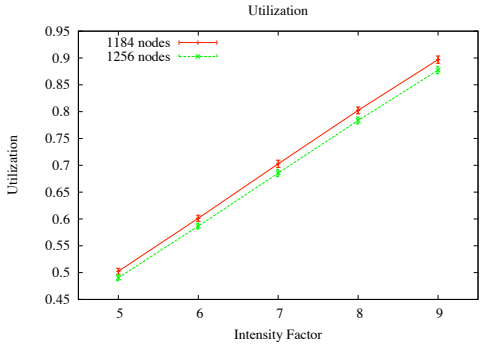

Fig. 12: System Utilization Comparison

running on all of its allocated nodes, and 2) the wait time, in which the job is waiting to be initially scheduled and launched. Figure 13 shows the average response time from both systems under various arrival intensities. As shown, the two figures cross as the intensity factor increases, indicating that the 1184-core yields shorter average response time when the job arrival intensity is less than 8 (i.e., when the system utilization is less than 80%) but that the 1256-core system yield shorter response time above the 8 intensity level. This crossover occurs because jobs incur shorter average queue times in the 1256-core system as shown in Figure 11.

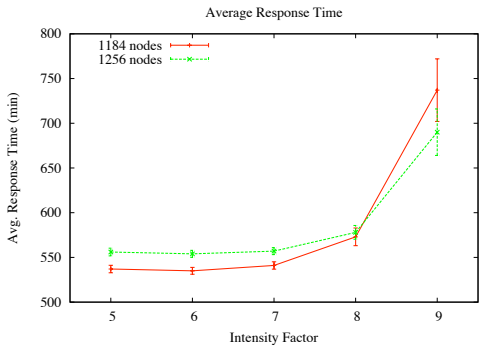

Fig. 13: Response Time Comparison

Both the wait time predictions and the response time predictions are useful metrics when making a business decision about what cluster configuration should be purchased. The business decision takes into account the anticipated workload intensity, the performance metrics at that intensity, cost of the system, and other factors.

5 Conclusions

This paper demonstrates that the ICPE methodology can be successfully applied to commodity cluster capacity planning efforts. The relative performance of two hypothetical supercomputer configurations, which have equivalent overall capacity, can be compared and contrasted to demonstrate under which workloads each is preferable. Under the scaled current workload, the system with more nodes performs better in terms of average wait time, but worse in terms of response time if the workload intensity factor is above a certain level. Therefore, depending on the projected workload level, different user groups will favor different systems. This demonstrates the effectiveness and versatility of the methodology and of the ICPE tool in performance modeling of large-scale distributed systems.

Acknowledgments

This work is supported by research grant from Acxiom Corporation.

References

1. Lu, B.: Integrated capacity planning environment for commodity cluster systems. Ph.D. Dissertation, University of Arkansas (2008)
2. Lu, B., Apon, A.W., Dowdy, L.W., Robinson, F., Hoffman, D., Brewer, D.: A case study on grid performance modeling. In: Parallel and Distributed Computing Systems. (2006) 607–615
3. Feitelson, D.G.: Metrics for parallel job scheduling and their convergence. In: JSSPP '01: Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing, London, UK, Springer-Verlag (2001) 188–206
4. Lublin, U., Feitelson, D.G.: The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. Parallel Distrib. Comput. **63**(11) (2003) 1105–1122
5. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. Mc Graw Hill (2000)
6. Morse, P.M.: Queues, Inventories and Maintenance. John Willey (1967)
7. Rossetti, M.: Jsl: An open-source object-oriented framework for discrete-event simulation in java. International Journal of Simulation and Process Modeling (2007)
8. L'Ecuyer, P., Simard, R., Chen, E.J., Kelton, W.D.: An object-oriented random-number package with many long streams and substreams. Oper. Res. **50**(6) (2002) 1073–1075