A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling *

Virginia Lo, Jens Mache, and Kurt Windisch

Department of Computer and Information Science
University of Oregon, Eugene, OR 97403
{lo, jens, kurtw}@cs.uoregon.edu

Abstract. Two basic approaches are taken when modeling workloads in simulation-based performance evaluation of parallel job scheduling algorithms: (1) a carefully reconstructed trace from a real supercomputer can provide a very realistic job stream, or (2) a flexible synthetic model that attempts to capture the behavior of observed workloads can be devised. Both approaches require that accurate statistical observations be made and that the researcher be aware of the applicability of a given trace for his or her experimental goals.
In this paper, we compare a number of real workload traces and synthetic workload models currently used to evaluate job scheduling and allocation strategies. Our results indicate that the choice of workload model *alone* – real workload trace versus synthetic workload models – did not significantly affect the relative performance of the algorithms in this study (two scheduling algorithms and three static processor allocation algorithms). Almost all traces and models gave the same ranking of algorithms from best to worst. However, two specific workload characteristics were found to significantly affect algorithm performance: (a) proportion of power-of-two job sizes and (b) degree of correlation between job size and job runtime. When used in the experimental evaluation of resource management algorithms, workloads differing in these two characteristics may lead to discrepant conclusions.

1 Introduction

Simulation-based performance evaluation of parallel job scheduling strategies is traditionally carried out using a synthetic workload model to generate a stream of incoming jobs with associated job characteristics. Despite the acknowledged rigor of simulation testing using stochastically generated workloads, there has been a pressing need for more realistic performance evaluation to further validate algorithm performance. Recently, the use of massively parallel machines for high performance computing has grown rapidly, both in numbers and in the maturity of the user communities. Systems administrators at national labs and

* This research was sponsored by NSF grant MIP-9108528.

supercomputing center sites have collected large amounts of workload trace data and released them for use in the evaluation of new resource management algorithms. Thus, researchers in performance evaluation have at their disposal two valid methods for conducting simulations:

> (1) Use of **real workload traces** gathered from scientific production runs on real supercomputers and carefully reconstructed for use in simulation testing.
> (2) Use of **synthetic workload models** that use probability distributions to generate workload data. We refer to the earliest synthetic workload models as "naive" because they were based on little or no knowledge of real trace characteristics (due to the fact that real traces didn't exist). Recently, more realistic synthetic models have been developed in which the model and its parameters have been abstracted through careful analysis of real workload data from production machines.

Both approaches require that many assumptions and accurate statistical observations be made, and that the modeler understands well the profile of the targeted, real workload. Inaccurate assumptions or minor perturbations in any proposed model may yield a workload that provides a poor evaluation of scheduling strategies on the targeted system. Yet, there exists very little published literature that offers guidance to researchers concerning the use of real workload traces and synthetic workload models for experimentation with scheduling algorithms.

Our work aims to fill this void. Our goals are to determine the degree of influence of workload choice and workload characteristics on performance and, where possible, to isolate the causes of observed differences in performance results. We will begin to address issues such as those listed below and to propose some rules-of-thumb to help guide the use of these workload traces and models in simulation testing of scheduling algorithms.

- **Real workload traces versus synthetic workload models:** When should one use real traces and when should one use synthetic workloads? Do the results of simulation with these two types of workloads reinforce each other? If not, is it due to biases in the workload traces or inadequacies in the synthetic model?
- **Universality of workload traces:** Given a choice among real workload traces from different sites, how does one know which trace to use? If performance evaluation results are discrepant, how does one know which results are valid?
- **Sensitivity of scheduling algorithm performance to workload characteristics:** What specific workload characteristics might bias performance results?

The experiments reported in this paper compare many real workload traces and synthetic workload models, both analytically and through simulation, observing their effects on the performance evaluation of several classes of scheduling

and allocation strategies. Included were two scheduling algorithms: First Come First Served and ScanUp [16], a multi-level queuing algorithm, and three static allocation strategies: First Fit [24], Frame Sliding [3] , and Paging [17].

The real traces were captured from four production machines in use for scientific computing at research labs and supercomputer sites around the world (two IBM SP-2s, an Intel Paragon, and a Cray T3E). The synthetic models include "naive" models and those developed by Downey [6, 5] and Feitelson [7] based on their careful analyses of traces from production machines.

We conducted five distinct experiments: (a) real workload traces versus synthetic workload models; (b) realistic synthetic workload models versus naive synthetic models; (c) a comparison of real workload traces across disjoint time periods (at the same site); (d) a comparison of real workload traces across sites; and (e) effects of specific workload characteristics: power-of-two jobsizes and correlation between jobsize and runtime. These experiments led us to the following general observations; more details are discussed in Section 5.

- The choice of workload *alone* did not significantly affect the relative performance of the resource management algorithms. Almost all workloads (real or synthetic, across sites, and for different time periods at the same site) ranked the scheduling and allocation algorithms in the same order from best to worst with respect to response time and system utilization. The choice of workload did yield differences in more subtle aspects of algorithm performance.
- Two critical workload characteristics were found to significantly affect algorithm performance: (a) proportion of power-of-two job sizes and (b) degree of correlation between job size and job run time. When used in the experimental evaluation of resource management algorithms, workloads differing in these two characteristics may lead to discrepant conclusions.

As we shall see, for both real traces and synthetic models, it is critical that one be aware of specific trace characteristics and the applicability of a given trace for the researcher's experimental goals.

The remainder of this paper is organized as follows: Section 2 gives the background for this study and surveys related work; Section 3 discusses real workload traces and synthetic workload models, and describes the specific traces and models used in this study. In Section 4 we describe our experiments, including the resource management strategies and performance metrics used. Section 5 discusses our experimental results, and Section 6 gives our conclusions.

2 Background and Related Work

This project was motivated by our desire to improve the quality of our own work in performance evaluation of scheduling algorithms and to help facilitate the comparability of (sometimes contradictory) results obtained in the scheduling community. We would like to help develop a benchmark suite of real workload traces and synthetic models for use in the resource management community.

Our focus is on job-oriented resource management for distributed memory parallel machines, and on job-oriented workloads, where a job consists of a collection of one or more computational tasks to be run in parallel. The *job scheduling strategy* involves the decision about which of many queued jobs is next to be allocated resources. Job scheduling policies range from the classic First-Come First Served algorithm to complex, multi-level queue models such as those implemented by scheduling systems such as NQS, Load Leveler, PBS or EASY [4, 14, 12, 13]. The *job allocation strategy* selects the set of processors to be allocated to the job based on its jobsize request. We restrict our attention to static allocation strategies rather than adaptive strategies: the former simply allocate the requested number of processors.[1] Static allocation strategies fall into two classes: contiguous and non-contiguous, based on whether or not the set of allocated processors are directly connected by links in the interconnection network. Research in job scheduling and processor allocation is thoroughly surveyed in [8]; more recent work in this area includes that reported in [10] as well as our own [17, 22, 18]. This project is distinguished from our previous work in that we evaluate the experimental method, not the scheduling techniques themselves.

Some of the first researchers to use real traces from production machines to drive their simulations include [19, 1, 18]. At the same time, analysis of this emerging body of trace data was conducted by Feitelson and Downey in the development of realistic synthetic workload models. Feitelson's model combines observations of five different parallel supercomputers in the evaluation of gang scheduling strategies [7], and Downey's model is based on detailed analysis of the SDSC Paragon, utilized in experiments evaluating adaptive job scheduling algorithms [6, 5]. The synthetic workload models developed by Downey and Feitelson are used in our experiments and are discussed in more detail in Section 4.

Several studies statistically analyzed workload traces from production use of real parallel supercomputers. Feitelson and Nitzberg analyzed the workload from an Intel iPSC/860 located at NASA Ames, providing the first widely available workload measurements from a real system [9]. Windisch et. al. [21] continued this effort by analyzing traces from the Intel Paragon at the San Diego Supercomputer Center (SDSC), and comparing the workload to that of the NASA iPSC/860. The workload characteristics analyzed in these studies included general job mix, resource usage, system utilization and multiprogramming level, runtime distribution, job submission statistics, and interarrival time distributions. Overall, the profiles of the two workloads were surprisingly similar. Hotovy analyzed the evolution of the workload on an IBM SP-2 at the Cornell Theory Center (CTC), concluding that workloads change in significant ways over time, requiring adaptations in the scheduling mechanisms for efficient operation [11].

A few of the many studies of job scheduling and processor allocation algorithms have offered insights into the effects of workload characteristics on those

[1] Adaptive allocation strategies for moldable jobs allocate a number of processors that is a function of the number of free processors in the system and of characteristics of the job.

algorithms. Of special interest to our study is Krueger's work on scheduling and allocation performance under workloads exhibiting negative correlations between jobsize and runtimes [16]. As we shall see, our work further explains some of the phenomena he observed. The only recent study that we know of that has focused on the experimental methodology itself, i.e., the effect choice of workload has on scheduling performance results, is that of Chiang et. al. [2]. They compare the performance of several scheduling strategies over a wide range of workload parameters and conclude that the discrepancies among various studies are due to differences in the (synthetic) workloads used for performance evaluation. Neither Krueger nor Chiang studied the effect of real workload traces on performance.

3 Workloads

A workload trace is a record of resource usage data about a stream of parallel and sequential jobs that were submitted to and run on a given message-passing parallel machine. Each job arrives, executes on one or more nodes for a period of time, and then departs the system. The resources requested by these jobs typically include jobsize (the number of requested processors), runtime, memory requirements, I/O devices such as disks or network interfaces, and software resources. Furthermore, every job in a workload is associated with an arrival time, indicating when it was submitted to the scheduler for consideration.

Real workload traces captured from production machines can potentially provide a very high level of realism when used directly in performance evaluation experiments. However, this usage comes with a number of caveats in the interpretation of performance results. A given trace is the product of an existing scheduling policy and thus is biased or affected by that policy. Human factors must be considered when analyzing or utilizing workload data: human users often adapt their resource demands to the system or the scheduling policies, in ways that do not reflect the actual needs of the job. Finally, workload traces may not discriminate between a job's direct resource needs and the resources utilized by the operating system on behalf of the job; furthermore, traces may not distinguish between inherent versus contention-related resource usage.

Synthetic workload models offer the convenience of a much more manageable experimental medium that is free of the idiosyncratic site-specific behavior of production traces. This is both its advantage and a potential point of criticism – the use of a tractable synthetic model provides a neutral, more universal experimental environment but does not yield the realism and pragmatic testing desired by some researchers. Sometimes the mathematical modeling smoothes out perturbations that it is desirable to investigate.

Therefore, researchers should be cognizant of the details surrounding the trace and consider how applicable a given trace is as a model of the target system. They need to be mindful of their specific experimental goals; performance evaluation by a site administrator at one of the supercomputing centers has a much more focused and pragmatic goal than that performed by an academic researcher developing new algorithms for future machines and environments.

3.1 Real Workload Traces

The traces described below were collected from a variety of machines at several national labs and supercomputing sites in the United States and Europe. The type of workload at all the sites consisted of scientific applications ranging from numerical aerodynamic simulations to elementary particle physics. Trace data was collected through a batch scheduling agent such as the Network Queuing System, Load Leveler, PBS, or EASY. Traces that have been used for trace-driven simulation were sanitized to remove user specific information and pre-processed to correct for system downtimes, sporadic missing data, and then reformatted for use in the simulator. In some cases, trace data may also be manipulated in order to study specific phenomena (e.g. selective filtering to remove interactive jobs). We briefly summarize the machine architecture, user environment, and scheduling policies in force at each site. Our experiments used traces from the first four machines in the list.

- **SDSC Intel Paragon**: The San Diego Supercomputer Center houses a 416 node Paragon machine. The scheduling policies are implemented through the Network Queuing System (NQS) which queues jobs according to power-of-two jobsizes, maximum runtime, and memory requirements (16 MB vs. 32 MB nodes). The (static) allocation algorithm is a block-based non-contiguous strategy. The SDSC traces were taken from 1995-1996, in three-month groups. [21, 20]
- **CTC IBM SP-2**: The Cornell Theory Center IBM SP-2 machine has 512 nodes connected by a high performance switch. The traces come from two periods: July 1994-March 1995 with scheduling managed by IBM's LoadLeveller and July 1995-Feb 1996 under LoadLeveller and Easy [11].
- **NASA Ames IBM SP-2**: The NASA Ames IBM SP-2 machine has 160 nodes connected by a high performance switch. The traces cover two years, from August 1995 to August 1997. The scheduling policies are implemented through the Portable Batch System (PBS) [15].
- **KFA Cray T3E**: The Forschungszentrum Jülich has a 512 node CRAY T3E. Jobs are submitted through NQS. Job queues differ in maximum (power-of-two) jobsize and maximum runtime. The (static) allocation algorithm is contiguous. The traces were taken from March 1997 - September 1997.
- Other traces include those from the iPSC/860 at NASA Ames NAS, a 128 node hypercube; the ETH Paragon, a 96 node mesh; the Cray T3D at Pittsburgh Supercomputing Center, a 512 node torus-based machine; Argonne National Laboratory's 128 node IBM SP-1, a 512 node CM-5; and 126 node shared memory BBN Butterfly at Lawrence Livermore National Laboratory's BBN Butterfly.

3.2 Synthetic Workload Models

Workload modeling of the nature required for resource management in distributed memory parallel machines is in its infancy due to prior lack of trace

data available for analysis. For the most part, performance evaluation studies have relied on "naive" synthetic models, using classic probability distributions such as exponential and uniform.

Dror Feitelson [7] analyzed trace data from five of the above machines, specifically the NASA iPSC/860, ANL IBM SP-1, SDSC Paragon, LLNL Butterfly, and ETH Paragon. His goal was to derive probabilistic models for use in his experiments with gang scheduling algorithms. Feitelson proposed a harmonic distribution to model jobsizes, and then hand tailored the model to emphasize small job sizes and other "interesting sizes" that he observed across all five machines. These included powers-of-two, squares, multiples of 10, and full system size requests.

He used a two-stage hyperexponential distribution to model runtimes and enforced a linear relationship between jobsize and the probability of using the distribution with the higher mean. This decision was based on observations of a strong correlation between jobsize and job runtime on the NASA iPSC/860 and a weak correlation on the ANL SP-1 and SDSC Paragon. (Our results relating to correlation of jobsize and runtime are discussed in Section 5.)

Feitelson also modeled user job submission behavior, using a Zipf distribution to model repeated submissions of the same job (job *runlength*). We note that Feitelson chose to use an exponential distribution to model interarrival times; our own studies of workload data have shown this assumption to be strongly justified [21].

Allen Downey [6, 5] focused his analysis on two machines, the SDSC Paragon and the CTC SP-2. He proposed a uniform-log distribution for modeling job lifetimes (approximated by the product of runtime and number of processors). From his analysis of the SDSC trace data, he observed that this distribution is accurate except for very small and very large jobs. Downey's curve-fitting analysis eliminated the smallest 10% and largest 10% of jobs. He used this model to predict queue waiting times and verified the accuracy of his model using the SDSC workload data [2]. From his comparison of the SDSC and CTC workloads, he noticed that the same (uniform-log) model is accurate for both machines/sites, but the specific parameters differ significantly. Downey warned that researchers will need to be very careful about use of these models, recommending that a workload derived from one system should not be used to evaluate another. We will discuss this precaution further in Section 5.

In a second study of the same machines, Downey derived a uniform-log distribution for job cluster size (which we call jobsize) by smoothing out the observed step function that characterized the raw trace data. He argued that the power-of-two cluster sizes responsible for the step function reflected the power-of-two NQS job queues, not the actual cluster size requirements of the jobs. This distribution and additional parameters involving variance in job parallelism were used to develop a stochastic model for simulation of several known adaptive

[2] Downey acknowledges that his results are prejudiced because he uses the same SDSC trace to derive the prediction model and to evaluate it.

strategies.[3] Downey's goal was to analyze the performance of these strategies in terms of their sensitivity to specific workload characteristics.

Finally, the types of "naive" workload models that have been used over the past decade of scheduling research include the following: The vast majority of researchers have used exponential distributions to model interarrival times. Jobsizes have been modeled using a variety of probabilistic models including uniform, uniform with step functions, normal, geometric, and exponential, while runtimes have been modeled with exponential and hyperexponential distributions.

4 Experiments

4.1 Experimental Method and ProcSimity Simulator

In our experiments we observed the performance of various resource management algorithms through simulation using real workload traces and compared those results with simulations of the same algorithms using synthetic models. The real workload traces that we used were the first four described in Section 3. In all cases, we used batch jobs only, removing the interactive jobs from the trace, since our focus is on batch scheduling. (Interactive jobs tend to have drastically different workload characteristics.) The synthetic models used in these experiments were those of Feitelson, Downey, and two naive models. The specific workload information that was used as input to the scheduling and allocation algorithms included job arrival time, jobsize and job runtime. Our simulator modeled a stream of jobs that arrive, execute for a period of time, and then depart the system. We did not model message-passing behavior in this study.

We conducted five distinct experiments:

- real workload traces versus synthetic workload models
- realistic synthetic workload models versus naive synthetic models
- a comparison of real workload traces across disjoint time periods (at the same site)
- a comparison of real workload traces across sites
- effects of specific workload characteristics: power-of-two jobsizes and correlation between jobsize and runtime.

For all five experiments, we simulated a mesh topology. This mesh was used in all of the experiments involving real workload traces, even if the trace was captured from a machine with a different architecture. For the studies involving the synthetic workloads and the SDSC workloads, we simulated a 16×22 mesh to match the size of the SDSC Paragon batch partition. For studies involving the CTC workload, we used a 16×27 mesh to match the size (but not the

[3] The type of allocation strategies studied by Downey are different from those studied in this paper. Downey focused on allocation for moldable jobs, where a job's cluster size is based on the number of available processors or job characteristics. We focus on static allocations in which the jobsize is a fixed request.

topology) of the CTC SP-2 machine. For the experiment investigating power of two jobsizes, we used a 32×32 mesh.

All experiments were conducted using ProcSimity [23], a simulation tool we developed for evaluating job scheduling and processor allocation algorithms for distributed memory parallel machines. ProcSimity models a variety of network topologies and several current flow control and routing technologies. ProcSimity supports both synthetic job streams and trace-driven simulation. Our simulator has been in use for several years at the University of Oregon and is currently in use at a number of research sites including the Ministry of International Trade and Industry in Japan, ETH Zurich, and academic institutions in the United States.

4.2 Resource Management Strategies

We evaluated the performance of two scheduling algorithms and three static allocation strategies, varying the type of workload used to drive the simulations. A given job scheduling strategy determines which of many queued jobs is admitted to the system for execution. The allocation strategy selects a subset of the physical processors for allocation to that job. We restricted our attention to static allocation strategies in which the number of processors assigned to a job is fixed for the lifetime of the job.

Job Scheduling Strategies

- **FCFS** is the classic First Come First Served Scheduling Algorithm. FCFS is a simple, single queue algorithm commonly used as a standard of comparison or as a default algorithm.
- **ScanUp** [16] is a multi-queue job scheduling strategy in which jobs are queued by jobsize. The scheduler services one queue at a time, from smallest to largest, serving only those jobs that arrived in a given queue *before* it selected that queue (thereby avoiding starvation). When it completes serving the largest job queue, it begins again with the smallest. ScanUp was shown to outperform a wide range of scheduling algorithms.

Allocation Strategies

Static allocation strategies can be classified as contiguous or non-contiguous, based on whether or not the set of allocated processors are directly connected by communication links in the interconnection network.

- **Frame Sliding** [3] searches for a rectangular block of processors by sliding a window of the desired size across the mesh in horizontal and vertical strides based on the width and height of the requested rectangle. Frame Sliding is a contiguous strategy with marginal performance.
- **First Fit** [24] searches for a contiguous block of processors starting at a reference point (e.g. lower left hand corner of the mesh). First Fit was shown to have the best performance among all contiguous strategies in [17].

- **Paging** allocates processors by scanning the free list of processors in a fixed order and allocating them to the job without regard to their contiguity. Paging was shown to have the best performance among all non-contiguous allocation strategies in [17][4]. Because we do not model message-passing contention in this study, its performance is the same as that of the whole class of purely non-contiguous algorithms.

4.3 Performance Metrics

Performance was measured using the following metrics:

- **average processor utilization**: the percentage of processors allocated to jobs at any given time, averaged over the entire workload.
- **average response time**: the elapsed time from when a job arrives for scheduling to when it completes execution, averaged over the entire workload. Response time includes both time spent in waiting queues and time spent in execution.
- **slowdown ratio**: $\frac{average\ response\ time}{average\ runtime}$. This metric normalizes average response time so that results are more easily compared across workloads. This metric is not the same as average slowdown. The differences and the reason we chose slowdown ratio are discussed below.
- **system load**: $\lambda\ \frac{E[runtime*jobsize]}{N}$ where λ is the arrival rate and N is the system size (total number of processors). This metric measures the offered load relative to the size of the system, and appears as the independent variable in the graphs of experimental results.
- **sustainable load**: the system load value below which average job response times remain within *reasonable* bounds.

While we measured average response times in all experiments, the graphs presented in this paper use *slowdown ratio* as the dependent variable (on the y axis). An alternative metric is *average slowdown*, the expected value of the quotient response time divided by runtime. We found that average slowdown is a heavy-tailed distribution with very large values for outlier data points. These outliers turned out to be jobs with very short runtimes (in the order of seconds) whose response time was huge (in the order of tens of hours) because they were blocked in the waiting queue behind a large and long-running job. We considered using order statistics (e.g. displaying results for the 90% quantile), but found slowdown ratio to be more suitable for this paper.[5]

It is important to realize that sustainable load, the system load value below which average response time remains within *reasonable* bounds, is an important focal point in performance evaluation. This critical point is visible as the "knee" in the graphs of slowdown ratio and system utilization. Below the critical point,

[4] Since that time we have developed a superior algorithm called MC [18] that is contiguous when a contiguous block exists, but non-contiguous otherwise.

[5] We plan to look further into the relative merits and utility of various performance metrics.

the system load is at manageable levels so that the increase in job response time is gradual and utilization continues to improve. At the knee, the job response time suddenly begins to grow rapidly toward infinity. By the same token, the system utilization levels off since the system is saturated with work. Thus, in evaluating the relative performance of resource management strategies, we focus our analysis on the phenomena observed near this saturation point.

Table 1. Ranking of Scheduling and Allocation Algorithms

Strategy	Synthetic models				SDSC traces	CTC traces	NAS traces
	N1	N2	D	F	S1 - S8	C1 - C2	N1 - N3
Paging/ScanUp	1	1	1	1	2	1	1
Paging/FCFS	2	2	2	2	1	2	2
FF/ScanUp	3	3	3	3	3	3	3
FS/ScanUp	4	4	4	4	4	4	4
FF/FCFS	5	5	5	5	5	5	5
FS/FCFS	6	6	6	6	6	6	6

5 Results

5.1 Real Workload Traces versus Synthetic Workload Models

Our experiments showed that the choice of workload trace *alone* did not affect the relative performance of the selected resource management algorithms. Almost all workloads (real or synthetic, across sites, and for different time periods at the same site) ranked the algorithms in the same order from best to worst with respect to slowdown ratio and system utilization. See Table 1. In addition, all workloads strongly discriminated among the three allocation algorithms, with non-contiguous Paging clearly outperforming First Fit, and First Fit clearly outperforming Frame Sliding. The distinctions between the two scheduling algorithms were consistent across workloads but not as pronounced, with ScanUp usually outperforming First Come First Served. In cases where rankings were inconsistent, we conducted further experiments to identify causes for the differences. These are discussed in Section 5.5.

5.2 Realistic Synthetic Models versus Naive Synthetic Models

Table 2 gives the probability model and values of associated parameters for each of the four synthetic models that we tested. Two of these models, Downey and Feitelson, were realistic models derived through careful analysis of real workload traces; two were naive models widely used in the scheduling literature. Downey did not model job runtimes directly but derived a uniform log model for job

Table 2. Models, means, and parameters for the four stochastic workloads

Model	Jobsize	Runtime	Inter-arrivals
Naive-1	uniform μ = 98.2 nodes	exponential μ = 1.0-8.0	exponential μ = 1.0
Naive-2	exponential μ = 47.1 nodes	exponential μ = 1.0-8.0	exponential μ = 1.0
Downey	uniform log μ = 61.26 nodes	uniform log μ = 7142.72 sec.	exponential observed μ = 967 sec. varied
Feitelson	harmonic hand-tailored μ = 22.75 nodes	2-stage hyper-exponential μ = 1289.49 sec.	exponential μ = varied

lifetimes. We also determined that a uniform log model was accurate for runtimes by using linear regression over job runtimes from the SDSC Paragon Trace (method of least squares). The maximum runtime was limited to 12 hours, the limit for the trace from which Downey's model was derived. We did not model the repeated job submissions used by Feitelson (runlength).

As shown in Table 1, all four synthetic models ranked the scheduling and allocation algorithms in the same order from best to worst. In Figure 1 it is interesting to note that results from the two realistic synthetic models were similar to each other as were results from the two naive models. This was true despite the fact that each modeled jobsize and job runtime with very different probability distributions. For example, with respect to system utilization, the two Naive models show the noncontiguous allocation strategies clearly outperforming the contiguous ones, with the choice of scheduling algorithm having a lesser effect. This can be seen in Figure 1 in which the algorithms cluster into two groups based on allocation strategy. In contrast, the Downey and Feitelson models show the scheduling algorithm having a more pronounced effect on performance.

The Naive models also differed from Downey and Feitelson's realistic synthetic models with respect to performance under increasing system loads. With the Naive models, performance of the system degrades more gradually as system loads approach 1.0, while with the two realistic synthetic models, slowdown ratios increase much more rapidly and earlier. As discussed in Section 5.5, these differences might be attributed to differences in two specific characteristics of the workload models.

Thus, it appears that the choice of synthetic model *alone* does not affect the overall ranking of scheduling and allocation strategies, despite the fact that they may use very different probability distributions in their models. The choice of synthetic model does affect more subtle aspects of algorithm performance.

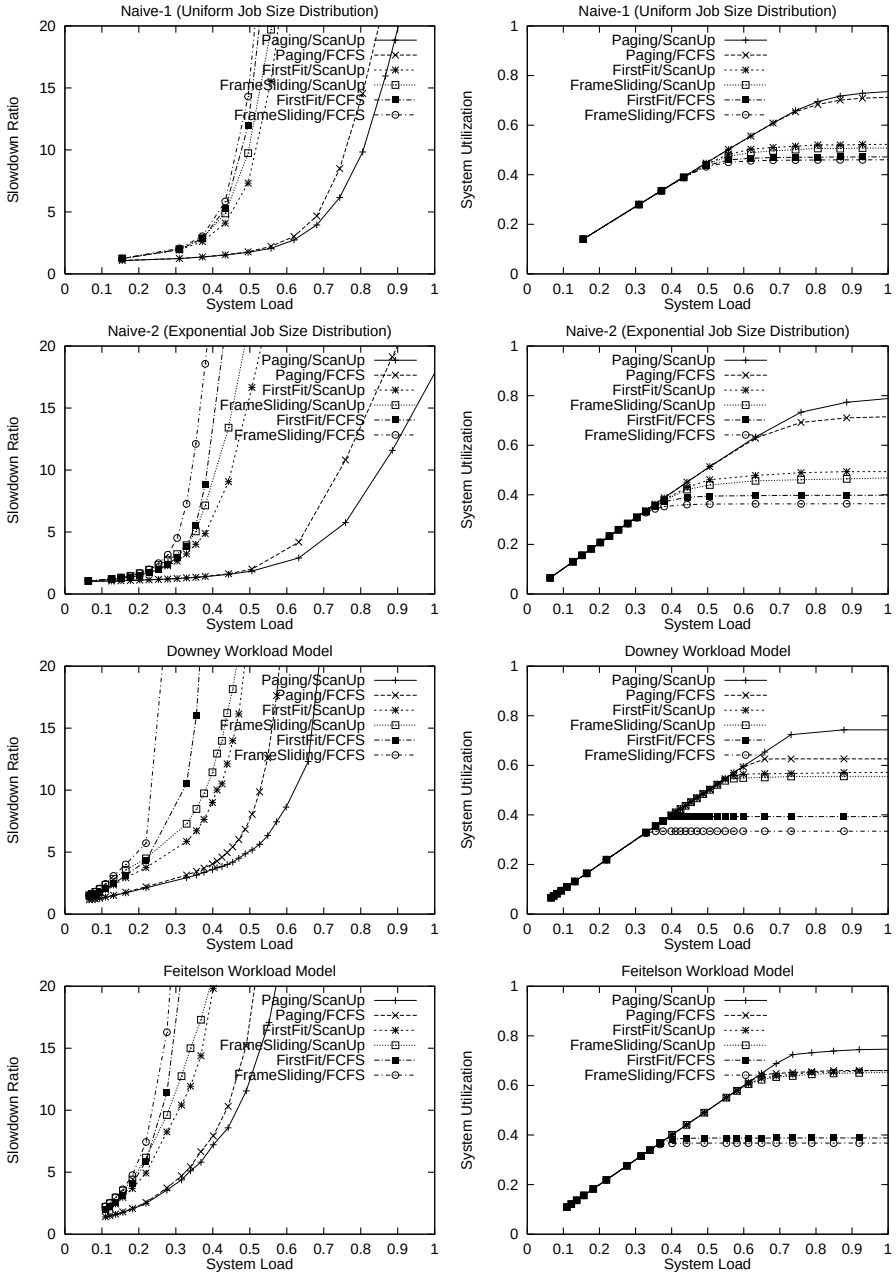

Fig. 1. Four Synthetic Workload Models: Naive1, Naive2, Downey, and Feitelson

5.3 Real Workload Traces over Disjoint Time Periods (at a Single Site)

In this experiment, we compared performance results from the SDSC Paragon traces over eight quarters in 1995 and 1996. We observed highly consistent results for all eight quarters, both in ranking and detailed behavior of scheduling strategies. Due to space limitations we only show graphs for the first quarter of 1995 and the last quarter of 1996 (see Figure 2). The strong consistency in performance results is especially notable given the variation in workload characteristics among the eight quarterly profiles. Table 3 shows the variation in means for jobsize, runtime, and interarrival times and for system load among the quarterly workloads.

We did the same comparison for the CTC SP-2 over two eight month periods in 1994-95 and 1995-96, respectively. The results were similar to those of the SDSC experiments. This is especially interesting because the scheduler changed from LoadLeveller to LoadLeveller/EASY.

We conclude that workload traces from the same site but different time periods are consistent in their evaluation of scheduling algorithms because the workload profile at a given site tend to be fairly stable over time (assuming a mature production site). However, as we discuss below in Section 5.5, there are other workload characteristics that are critical for performance.

Table 3. SDSC Workload Characteristics by Quarter

Quarter	Mean Jobsize (nodes)	Mean Runtime (secs)	Mean Interarrival (secs)	Mean Sys.Load
1995 Q1	22	8689	1100	.705
1995 Q2	21	8305	1042	.744
1995 Q3	29	8859	1516	.741
1995 Q4	32	8245	1836	.752
1996 Q1	23	12722	1590	.782
1996 Q2	18	11376	1130	.865
1996 Q3	26	9660	1598	.698
1996 Q4	15	10945	1301	.545

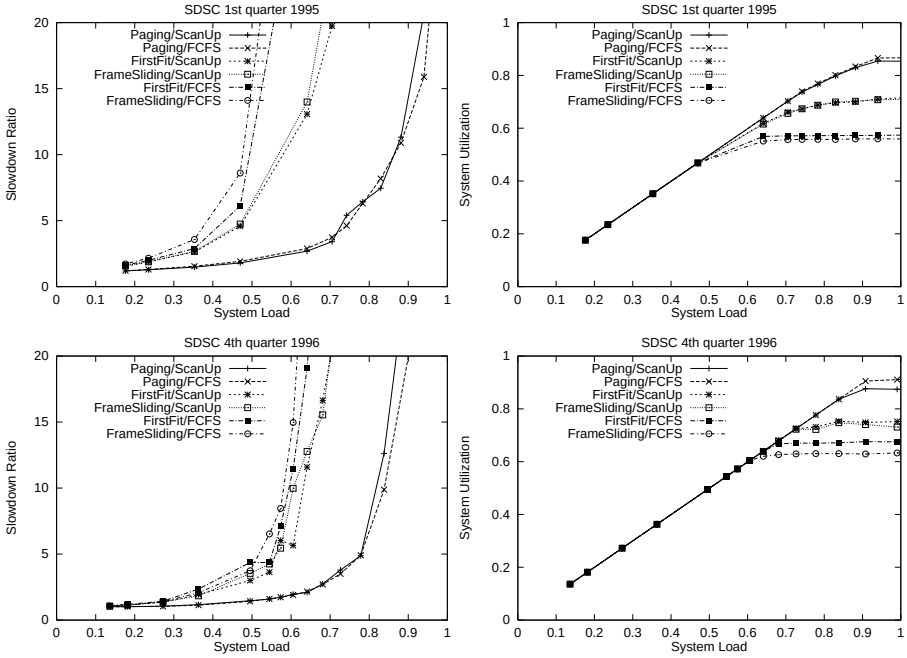

Fig. 2. SDSC 1st Quarter 1995 and 4th Quarter 1996

5.4 A Comparison of Real Workload Traces across Sites and Machines

Our goal for this set of experiments was to compare performance results based on real workload traces from different machines at different sites. The workload traces came from the following machines: SDSC Paragon, NASA Ames NAS IBM SP-2, Cornell Theory Center IBM SP-2, and KFA Cray T3E (see descriptions of each machine and user environment in Section 3).[6]

This set of experiments showed that, with one exception, the ranking of the selected scheduling and allocation algorithms was not affected by the specific workload trace used. (See Table 1.) The exception occurred on the SDSC Paragon for which FCFS slightly outperformed ScanUp as the top ranking algorithm; for the NAS and CTC workloads, ScanUp was the best algorithm as was true for all the synthetic traces as well.

All three workloads showed clear discrimination for non-contiguous allocation algorithms over contiguous algorithms; however, the degree of discrimination varied among the different workloads. It is also interesting to look at the performance of the FCFS algorithm alone. For the contiguous allocation algorithms (First Fit and Frame Sliding), FCFS achieved utilization levels of at best 40% on the NAS and CTC machines, but reached over 60% on the SDSC. See Figure 3.

Looking at the profiles of the workloads in Table 5, we found distinct differences in two areas, motivating the last set of experiments.

5.5 Effects of Specific Workload Characteristics

Effects of Proportion of Power-of-Two Jobsizes. One of the most obvious differences among workloads was the proportion of power-of-two jobsizes. Downey's model smoothed out the power-of-two step function while Feitelson's emphasized these sizes. Among the production traces, the proportion of power-of-two jobsizes was very high, ranging from 84.2% for SDSC to 100% for the iPSC/860, a hypercube machine.

We looked into the effect power-of-two jobsizes have on performance evaluation by creating three synthetic traces: one in which jobsize is taken from an exponential distribution, one forced to have a minimum of 50% power-of-two jobsizes, and one forced to have 100% power-of-two jobsizes. The traces for 50% and 100% sets were created so that the step function matched the initial exponential distribution when smoothed. As seen in Table 4, as power-of-two dominance increases, so do utilization levels for all resource management algorithms. As a result, the sustainable load also increases with increasing dominance of power-of-two jobsizes.

Another interesting phenomenon is the fact that with a 100% power-of-two job mix, it is the scheduling strategy (not allocation strategy) that determines

[6] We are still running experiments with KFA T3E.

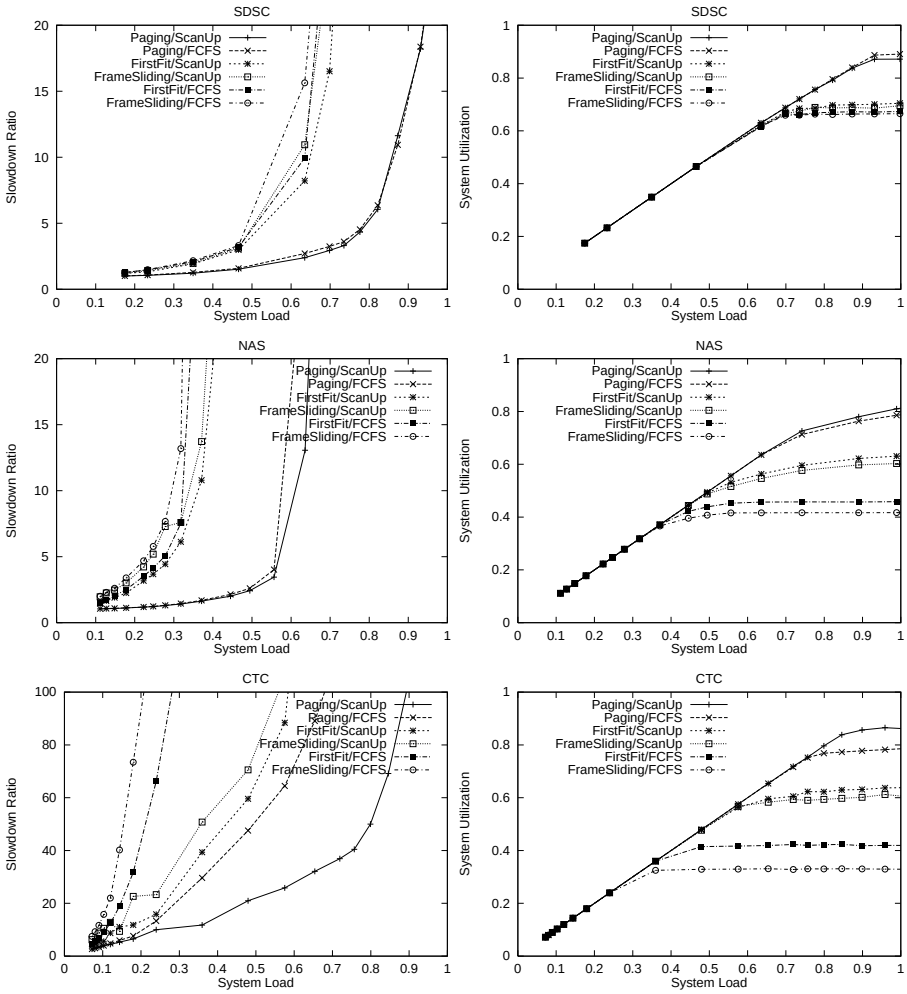

Fig. 3. SDSC Paragon versus NAS SP-2 versus CTC SP-2

performance. ScanUp outperforms First Come First Serve, regardless of allocation strategy. This result is consistent with that of [16] in their experiments with scheduling and allocation algorithms for the hypercube.

Table 4. Effects of Power-of-two Jobsizes on System Utilization and Sustainable Load. Minimum and maximum utilization levels are across all algorithms. Sustainable loads are shown for the worst performing algorithm FS/FCFS and the best performing algorithm Paging/ScanUp.

Percent Power-of-2	Min Util.	Max Util.	Sust.Load FS/FCFS	Sust.Load Paging/ScanUp
3.4%	41%	78%	.36	.74
51.7%	45%	84%	.39	.82
100%	63%	90%	.48	.93

Effects of Degree of Correlation Between Jobsize and Runtime. Degree of correlation between jobsize and runtime is of interest in the scheduling community because it reflects certain assumptions about the work model and the type of scheduling algorithms needed. The *fixed work model* and the notion of adaptive scheduling for *moldable jobs* carry an implicit assumption that jobsize and runtime are negatively correlated since they assume that the more processors given to a job, the more quickly it will finish execution. The *independent work model* presumes that jobsize is unrelated to job runtime (zero correlation).

We note that a common assumption for many large production workloads is that jobsize and runtime are positively correlated. Table 5 shows the range of correlation coefficients relating jobsize to job runtime for some of the real workloads and synthetic models reported in this paper. We used Pearson's r, which presumes a linear relationship between the variables, to compute the correlations. In reality, this relationship does not necessarily hold. Feitelson [7] noted a strong positive correlation for the NAS iPSC/860 trace but found much weaker relationships for other traces.

To test the effect of correlation on performance evaluation, we used the SDSC workload as a base and manipulated the data to achieve correlations of -1, 0, and +1 (see Figure 4). Our experiments show that for highly positively correlated workloads, Krueger's ScanUp algorithm, the best performing strategy in all other studies, performed worse than FCFS for all three allocation strategies!!

The explanation lies in the correlation. Recall that ScanUp uses multi-level queues, with each queue associated with a specific jobsize, from small to large. Thus, with a strongly positive correlation, large, long-running jobs arrive in the large-jobs queue while small, short-running jobs arrive in the small-jobs queue. While ScanUp is serving the large-jobs queue, smaller sized jobs arrive at the other queues. However, because all the large jobs are also long-running, these smaller queues fill up and the response time for these smaller jobs increases

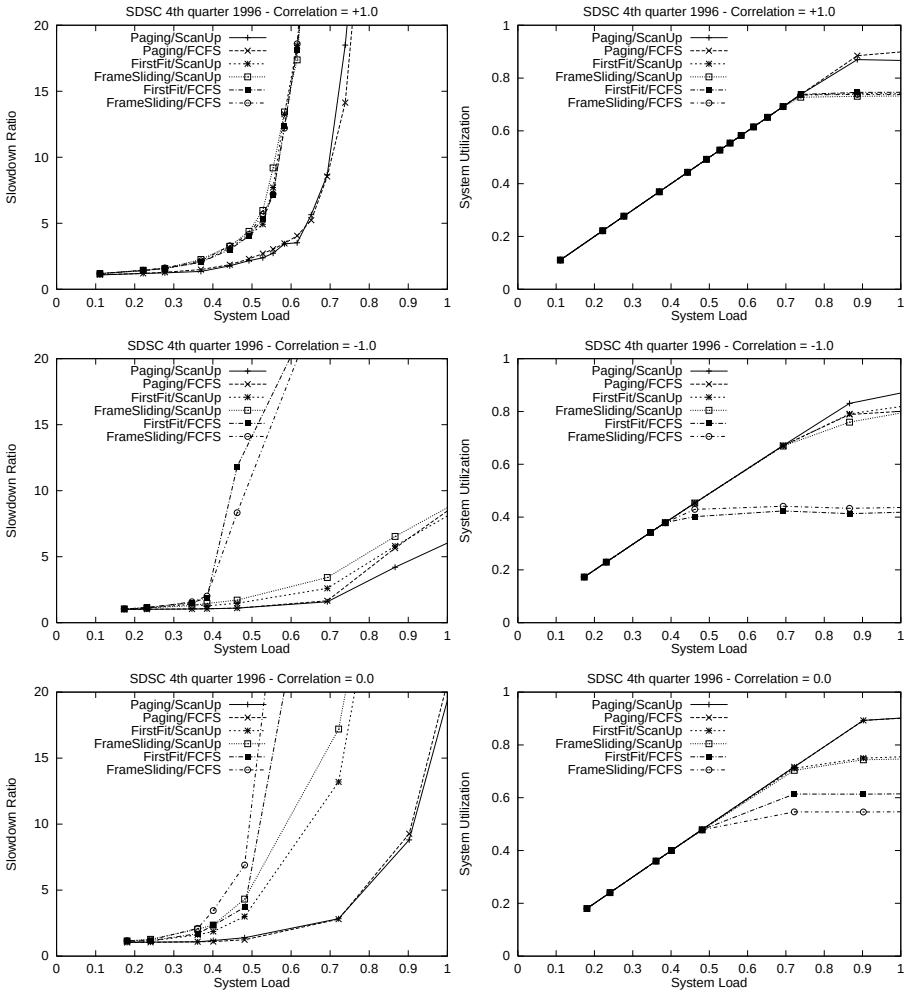

Fig. 4. Effects of Correlation between Jobsize and Runtime

Table 5. Trace Characteristics

Trace	Runtime and Jobsize Correlation Coefficient	Percent Power-of-2 Jobs
NAS iPSC/860	strongly pos.	100.0%
SDSC Paragon	0.20 - 0.35	84.2%
CTC SP-2	≈ 0	88.5%
NAS SP-2	≈ 0	51.2%
Feitelson	0.19	84.9%
Downey	0	16.9%
Naive1	0	6.3%
Naive 2	0	20.2%

rapidly, resulting in poor performance. In addition, when ScanUp serves several large jobs in a row, fragmentation of the processor space is high, also diminishing utilization. Thus, for positively correlated workloads, ScanUp suffers.

Looking back at Krueger's performance evaluation experiments for ScanUp, we see that he simulated ScanUp under non-correlated and negatively correlated workload models, the latter based on the fixed work model. Thus, our results are complementary to Krueger's and together show that the performance of algorithms is critically dependent on the correlation.

The effect of correlation between jobsize and runtime on performance results is also illustrated by examining Krueger's statement that "scheduling is more important than allocation." Our results confirm what Krueger concluded, however, only for negatively correlated workloads. In that case, ScanUp did best under all three allocation strategies. However, for non-negatively correlated workloads, scheduling did not dominate.

The key observation is that correlation between jobsize and runtime has strong effects on performance results. Thus, researchers need to take this factor into consideration when choosing or designing workloads and when evaluating algorithm performance.

6 Conclusions and Future Work

In this paper we investigated the use of real workload traces and synthetic workload models for performance evaluation of several parallel job scheduling algorithms. Our long term goal is the development of guidelines for the effective use of traces and models in scheduling research.

Our experiments showed that the choice of workload *alone* – real trace versus synthetic model – did not significantly affect the relative performance of the selected resource management algorithms (FCFS and ScanUp scheduling; First Fit, Frame Sliding, and Paging allocation). Almost all workloads ranked the algorithms in the same order from best to worst with respect to response time, slowdown, and system utilization.

It also appears that the choice of synthetic model *alone* does not affect the overall ranking of these scheduling and allocation strategies, despite the fact that they may use very different probability distributions in their models. The choice of synthetic model does affect more subtle aspects of algorithm performance.

We saw that workload traces from the same site but different time periods are consistent in their evaluation of scheduling algorithms because the workload profile at a given site tends to be fairly stable over time (assuming a mature production site).

However, our experiments revealed clear differences in performance using real workload traces from different machines at different sites. Our investigation of the causes of this inconsistent behavior led us to two factors which significantly affect performance evaluation results:

- As the **proportion of power-of-two jobs** in the workload increases, so does system utilization. As a result, the sustainable load also increases with increasing dominance of power-of-two jobsizes.
- We also found that **correlation between jobsize and runtime** has strong effects on performance results. Scheduling algorithms that did well on a workload with strong positive correlation did worse on a negatively correlated workload, and vice versa.

Taken together, these results show that great care must be taken in the use of both realistic workload traces and synthetic workload models. Naive synthetic workload models are useful in qualitative performance analysis, giving a high level evaluation of algorithm performance. Realistic synthetic workload models provide more detailed performance analysis. Both provide a convenient experimental medium in which parameters are more easily controlled. It is critical, however, that the researcher be aware of the profile of the workload produced by manipulation of the parameters in the context of his experimental goals.

Real workload traces provide a much more realistic simulation testbed but again precautions are necessary. The idiosyncrasies of a trace from one site may make it unsuitable for algorithm testing at another site. In addition, real traces must be carefully prepared for use in simulation testing of algorithms to remove biases that affect performance results.

The experiments reported here open up more questions than they answer. Some specific areas that we plan to investigate further include:

- Extension of these experiments to a broader range of scheduling algorithms. Will these same results hold up when applied to adaptive scheduling? to gang scheduling?
- Extension of these experiments to see if they hold up when evaluating algorithms whose performance is more similar than those we selected for this study? Will the various workloads differ in their ability to discriminate among very similar algorithms?
- Investigation of other workload characteristics that affect performance, such as effect of including interactive jobs and periodic job submissions patterns (such as day/night submissions and repeated submissions).

– Development of a benchmark suite of real workload traces and synthetic workload models for distribution throughout the scheduling community. Use of standardized suites would improve the ability of scheduling researchers to compare experimental results and to reach more solid conclusions about algorithm performance.

7 Acknowledgments

We would like to thank the following people who provided workload traces and information regarding the traces: Reagan Moore, San Diego Supercomputer Center; Steve Hotovy, Cornell Theory Center; James Jones, NASA Ames NAS; and Bernd Mohr, KFA. Thanks to Vimala Appayya and Drina Guzman from the University of Oregon for help with processing the traces. We also would like to thank the referees for their careful reading of the manuscript and their very insightful comments.

References

[1] R. H. Arpaci, A. C. Dusseau, A. M. Vahdat, L. T. Liu, T. E. Anderson, and D. A. Patterson. The interaction of parallel and sequential workloads on a network of workstations. In *Proceedings SIGMETRICS'95*, 1995.

[2] S. H. Chiang, R. K. Mansharamini, and M. K. Vernon. Use of application characteristics and limited preemption for run-to-completion parallel processor scheduling policies. In *Proceedings SIGMETRICS'94*, 1994.

[3] P. J. Chuang and N. F. Tzeng. An efficient submesh allocation strategy for mesh computer systems. In *Proceedings IEEE International Conference on Distributed Computer Systems*, 1991.

[4] Intel Corp. Paragon Network Queuing System manual. October 1993.

[5] A. B. Downey. A parallel workload model and its implications for processor allocation. In *Proceedings HPDC'97*, 1997.

[6] A. B. Downey. Predicting queue times on space-sharing parallel computers. In *Proceedings of the 3rd Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '97*, 1997.

[7] D. Feitelson. Packing schemes for gang scheduling. In *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '96*, 1996.

[8] D. G. Feitelson. A survey of scheduling in multiprogrammed parallel systems. Technical Report RC 19790 (87657), IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598, October 1994.

[9] D. G. Feitelson and B. Nitzberg. Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860. In *Proceedings of the 1st. Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '95*, April 1995.

[10] D. G. Feitelson and L. Rudolph, editors. *Job Scheduling Strategies for Parallel Processing*. Springer Lecture Notes in Computer Science, 1995-1997.

[11] S. Hotovy. Workload evolution on the Cornell Theory Center IBM SP2. In *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '96*, 1996.

[12] http://science.nas.nasa.gov/Software/PBS/pbshome.html. PBS portable batch system.

[13] http://www.llnl.gov/liv_comp/dpcs/. LLNL distributed production control system.

[14] http://www.tc.cornell.edu/Papers/abdullah.jul96/index.html. Extensible Argonne scheduler system (EASY).

[15] J. Jones. NASA Ames NAS, Personal communication, 1997.

[16] P. Krueger, T. Lai, and V. A. Dixit-Radiya. Job scheduling is more important than processor allocation for hypercube computers. *IEEE Transactions on Parallel and Distributed Systems*, 5(5):488–497, May 1994.

[17] V. M. Lo, K. Windisch, W. Liu, and B. Nitzberg. Non-contiguous processor allocation algorithms for mesh-connected multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 8(7):712–726, July 1997.

[18] J. Mache and V. Lo. Minimizing message-passing contention in fragmentation-free processor allocation. In *Proceedings of the 10th International Conference on Parallel and Distributed Computing Systems*, 1997.

[19] J. Subhlok, T. Gross, and T. Suzuoka. Impacts of job mix on optimizations for space sharing schedulers. In *Proceedings of Supercomputing '96*, 1996.

[20] M. Wan, R. Moore, G. Kremenek, and K. Steube. A batch scheduler for the Intel Paragon MPP system with a non-contiguous node allocation algorithm. In *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '96*, 1996.

[21] K. Windisch, V. Lo, D. Feitelson, B. Nitzberg, and R. Moore. A comparison of workload traces from two production parallel machines. In *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation*, 1996.

[22] K. Windisch, V. M. Lo, and B. Bose. Contiguous and non-contiguous processor allocation algorithms for k-ary n-cubes. In *Proceedings of the International Conference on Parallel Processing*, 1995.

[23] K. Windisch, J. V. Miller, and V. M. Lo. Procsimity: an experimental tool for processor allocation and scheduling in highly parallel systems. In *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, February 1995.

[24] Y. Zhu. Efficient processor allocation strategies for mesh-connected parallel computers. *Journal of Parallel and Distributed Computing*, 16:328–337, 1992.