The Impact of Information Availability and Workload Characteristics on the Performance of Job Co-allocation in Multi-clusters

William M. Jones[†] Walter B. Ligon III Nishant Shrivastava
864-656-7367 864-656-1224 864-656-5909
wjones@parl.clemson.edu walt@parl.clemson.edu nshriva@parl.clemson.edu

Parallel Architecture Research Lab
Department of Electrical and Computer Engineering
Clemson University
105 Riggs Hall
Clemson, SC 29634-0915
http://www.parl.clemson.edu/

Keywords – parallel job scheduling, multiple computational clusters, workload effects, network contention, multi-site scheduling

Abstract

In this paper, we utilize a bandwidth-centric job communication model that captures the interaction and impact of simultaneously co-allocating jobs across multiple clusters. We make use of a parallel job model that seeks to capture both local and global communication access patterns. By doing so, we are able to explore scheduling strategies that attempt to improve average job turnaround time by selectively mapping jobs across cluster boundaries in a process known as job co-allocation.

In this research, we focus on scheduling strategies that make use of available information such as network link utilization, per-processor bandwidths, and job communication topology in order to make intelligent decisions regarding application partition sizes and job placement. We provide results that help to establish the relationship between the quantity of information available a priori to the scheduler and its ability to improve overall system performance. Additionally, we demonstrate the dramatic impact that salient workload characteristics can have on the effectiveness of co-allocation.

[1]This work was supported in part by the ERC Program of the National Science Foundation under Award Number EEC-9731680. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation. Special thanks to Louis W. Pang for his contributions to the simulator. [†]Corresponding author.

1 Introduction

Computational multi-clusters (Figure 1) are an important emerging class of supercomputing architectures. As multi-cluster systems become more prevalent, techniques for efficiently exploiting these resources become increasingly significant. A critical aspect of exploiting these resources is the challenge of scheduling [5]. In order to maximize job throughput, multi-cluster schedulers must simultaneously leverage the collective computational resources of each of its participating clusters. By doing so, jobs that would otherwise wait for nodes to become available on a single cluster can potentially run earlier by aggregating disjoint resources throughout the multi-cluster in a process known as *co-allocation* or *multi-site scheduling* (Figure 1). This procedure can result in dramatic reductions in queue waiting times.

The main caveat of this approach is that by mapping jobs across cluster boundaries, inter-cluster network resources are also consumed. If the inter-cluster network links become too saturated with traffic, any co-allocated jobs may experience degraded runtime performance due to the communication bottleneck present in the network infrastructure. This degradation in runtime performance can potentially offset the benefit of performing job co-allocation in the first place. More precisely, the increase in job runtime due to link saturation can rapidly outweigh the decrease in queue waiting time, thus resulting in poorer overall system performance.

Multi-cluster schedulers must make use of all available information pertaining to job communication structure as well as network topology and utilization in order to improve job throughput while mitigating any negative impact to job

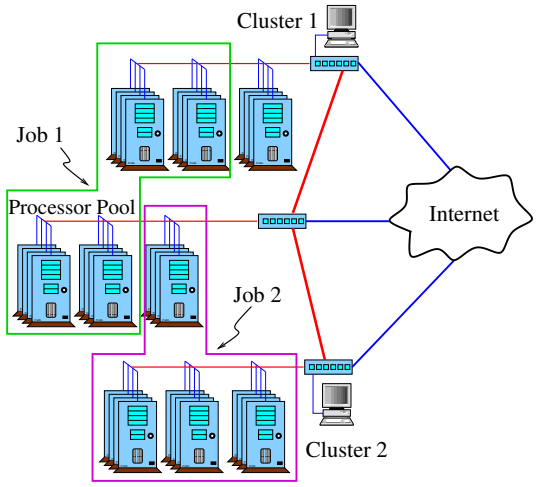

Figure 1. A multi-cluster with co-allocated jobs

runtime performance due to network congestion. Additionally, these schedulers must make reasonable co-allocation decisions in the absence of specific job and network information, as this information is not always available.

In this research, we utilize a bandwidth-centric job communication model that captures the interaction and impact of simultaneously co-allocating jobs across multiple clusters. We also make use of a parallel job model that seeks to capture both local (nearest-neighbor) and global (all-to-all) communication access patterns. By doing so, we are able to explore scheduling strategies that attempt to improve average job turnaround time by selectively mapping jobs across cluster boundaries.

Previous work in the area of job co-allocation tends to characterize jobs by either specifying that all communications require a fixed amount of time to travel between clusters [1], or by assigning co-allocated jobs a fixed execution-time penalty, [2]. This type of characterization is not sensitive to the time-varying contention for bandwidth in the inter-cluster communication links and the impact it has on the execution time of co-allocated jobs that share network resources. We take a different approach by considering that as jobs become co-allocated or co-allocated jobs terminate, there is a continual change in the available inter-cluster bandwidth. Therefore, in our work, the duration of wide area communication is a function of the time-varying network bandwidth utilization among clusters participating in the multi-cluster, which in turn affects the execution time of co-allocated jobs. This research aims to extend the work presented in [1] and [2] by replacing the static communication model with a more dynamic view of job communication that is *bandwidth-centric*.

We find that schedulers designed to allocate node resources across cluster boundaries can result in rather poor overall performance over a wide range of workload characterizations and multi-cluster configurations due to the interaction simultaneously co-allocated jobs experience as they contend for inter-cluster network bandwidth. Therefore in this research, we focus on scheduling strategies that make use of available information such as network link utilization, per-processor bandwidths, and job communication topology in order to make intelligent decisions regarding application partition sizes and job placement. We also provide results that help to establish the relationship between the quantity of information available a priori to the scheduler and its ability to improve overall system performance. Additionally, we demonstrate the dramatic impact that salient workload characteristics can have on the effectiveness of co-allocation over simple job migration.

2. The Model

In this section we characterize the parallel job model as well as the multi-cluster architecture. We provide a *very* brief explanation (due to space constraints) of the communication model used, as well as a strategy to account for the time-varying inter-cluster network utilization. This general methodology is a modified version of our previous work [9] and is fully documented in [8]. The interested reader can obtain a detailed treatment of the complete modeling methodology on our website [7].

2.1. Multi-cluster and Parallel Job Models

In the research presented in this paper, we consider a multi-cluster to be a collection of arbitrarily sized clusters with globally homogeneous nodes. Each cluster has its own internal ideal switch. Additionally, the clusters are connected to one another through a single dedicated link to a central ideal switch. Each node in the multi-cluster has a single processor and a single network interface card. Jobs can be co-allocated in a multi-cluster by allocating nodes from different clusters to the same job in order to better meet collective needs across the multi-cluster. The model used assumes that jobs are non-malleable. In other words, each job requires a fixed number of processors for the life of the job, and the scheduler may not adjust this number. Additionally, neither execution-time migration nor gang-scheduling is employed in mapping jobs the multi-cluster, i.e. once the job is mapped to a particular set of nodes, the job remains on these nodes for the lifetime of its execution.

A job's execution time, T_E, is a function of two components, the computation time, T_P, and the communication time, T_C. The initial value of these two quantities is considered to represent the total execution time that the job would experience on a *single dedicated cluster* with an ideal

switch. They therefore form a basis for the best-case execution time of a given job when it is co-allocated in the multi-cluster. The computation portion of the execution time does not vary, however the communication time is considered dynamic, since the communication time of simultaneously co-allocated jobs may be lengthened due to the utilization of any shared inter-cluster network links.

2.2. Communication Characterization

In order to capture both local and global communication characteristics, each job modeled in this paper is assumed to perform both nearest neighbor (2D mesh) and all-to-all communication patterns throughout its execution.

During co-allocation, nodes must communicate across cluster boundaries. This communication requires a certain amount of bandwidth in the inter-cluster network links. A job's performance will deteriorate if it does not receive the amount of bandwidth it requires to run at full speed. Each time a new job is co-allocated or a co-allocated job terminates, an algorithm is applied in order to determine the amount of bandwidth ultimately allotted to each job on each link. The amount of bandwidth each job receives is limited by the most saturated link over which it spans.

As these inter-cluster state changing events occur, the remaining execution and communication times are recalculated based on a number of factors, including available network bandwidth. Due to these recalculations, the job's end-event can slide forward (later) or backward (earlier) in time, reflecting either a degradation or improvement in saturation levels of the inter-cluster links over which it spans. In this paper, jobs are characterized by a per-processor bandwidth (PPBW) that describes the total bandwidth that is required to sustain full-rate execution. Furthermore, this PPBW is decomposed into two parts; the fraction that is due to global all-to-all communications and the remaining bandwidth that is due to the 2D mesh nearest neighbor communication access patterns. The parameter ρ is used throughout this paper to denote the percentage that is associated with the all-to-all comunication. (The full description is rather lengthy and is therefore ommitted here due to space constraints, but can be found in [9].)

This procedure provides a dynamic view of job communication by accounting for the slowdown a job experiences due to the time-varying utilization of the inter-cluster network links.

3 Simulation

In this paper, we present results based on two distinct workload characterizations. The first is based on actual workload trace files taken from several different supercomputing centers, while the second is a completely synthetic workload used to contrast this work with our previous work as well as to highlight a few critical issues with the realistic workload. The remainder of this section provides an overview of our workload generation techniques.

3.1 Realistic Workload Generation

The workload on a cluster, in general, can be specified using three characteristics, namely; job arrival process, job size distribution and job runtime distribution. For generating a realistic workload in our simulator, we use statistical distribution-based models, derived from the actual workload traces obtained from various supercomputing sites. One of the main advantages of using a statistical model to synthetically generate a workload based on actual trace files is the ability to create a stream of jobs that is sufficiently long as to ensure that the system is operating in a stable steady-state [4]. Additionally, when using the jobs present in the actual workload, the simulation is limited to the number of jobs in the particular trace. This quantity may well fall short of the number needed to obtain convergence in the performance metrics used, especially when the multi-cluster system is heavily loaded [3]. We have therefore made use of the statistical modeling approach proposed by Lublin and Feitelson [11].

3.2 Job Arrival Process

In practice, job arrival processes tend to have daily, weekly and yearly cycles. For the purpose of simplicity, we consider only the daily and weekly cycles (Figure 2). Arrivals during the peak hours (8 a.m. to 7 p.m., Mondays through Fridays) are modeled using a Gamma distribution and the average daily cycle is modeled using another Gamma distribution. The daily cycle is divided in 24 slots and the slots are given different weights corresponding to the relative arrival rates within each slot. In the weekly cycle, the maximum job arrival rate is observed on Wednesdays while, Sundays and Saturdays have the lowest job arrival rates.

While the Gamma distribution could be used to model the weekly cycle as well, we made use of a Weibull distribution since it provides a better fit, particularly at the tail.

3.3 Job Size and Runtime

Jobs are divided into three categories: serial, power-of-two and jobs of other sizes (Figure 4). Sizes of non-serial jobs are modeled using a two-stage Uniform distribution. Additionally, job run-times are modeled using a Hyper-Gamma distribution which is a combination of two Gamma distributions (Figure 3). It should be noted that this workload characterization assumes that there is a weak correlation between job sizes and their respective run-times. A

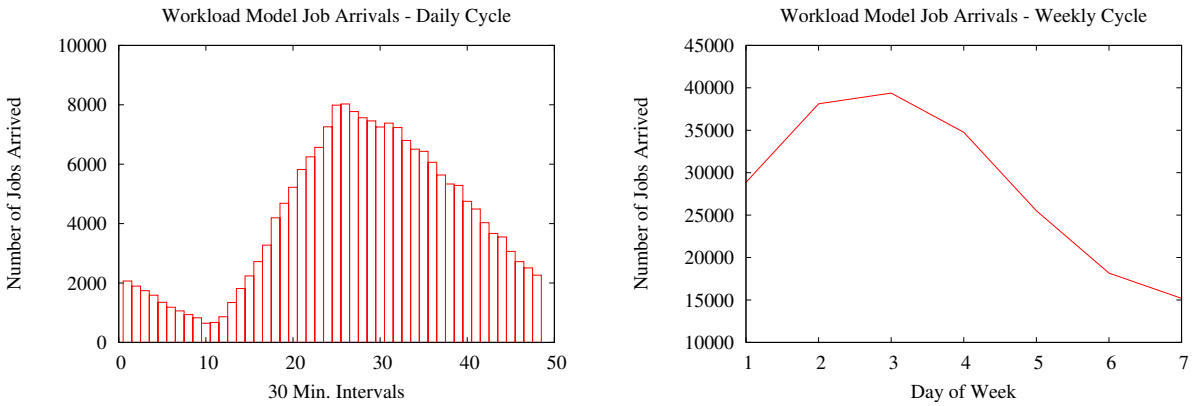

Figure 2. Daily and weekly arrival cycles

detailed description of this methodology can be found in [11].

3.4 Workload Generator Parameters

In order to obtain the necessary parameters to drive the statistical models used for job arrival, size, and runtime distributions, we made use of five workload traces provided by the Parallel Workload Archive, [12], namely:

The distribution parameters were extracted from each of the original workload traces by making use of several built-in functions in MatLab. They were then used to provide a parameterization for the statistical models used in our workload generator. The specific parameters used to drive the Hyper-Gamma, Gamma, Weibull, and Two-stage Uniform distributions are listed in Chapter 3 of [8] due to space constraints. (This text can be found at [7]).

3.5 Completely Synthetic Workload

Our second workload makes use of a much simpler characterization. This workload is used in contrast to several of the characteristics present in the previously described realistic workload. This workload assumes that the the arrival process of jobs to each cluster, C_i, has a Poisson distribution with rate λ_i. Additionally, we assume that a job's initial service time, T_E is exponential with parameter $(\mu_i)^{-1}$. The number of nodes that a job requires is given by a uniform distribution $D_i^{nodes} \sim UNIF[n_1^i, n_2^i]$. For the results presented in this paper, the synthetic workload parameters are as follows: $D_i^{nodes} \sim UNIF[10, 50]$, $\lambda_i = 1/150$, and $(\mu_i)^{-1} = 450$.

In both workloads, the fraction of *the total execution time* that initially represents computation is set to a constant, $K_i = .7$, for all jobs. Additionally, since this research seeks to study the effects of varying levels of local and global communication, we have also required that each job be rectangular. That is to say, since a job performs 2D nearest-neighbor communication, we have assumed that the job can be decomposed into a whole number of rows and columns. This implies that the actual job sizes are determined as $n_T = rows \cdot cols = \lfloor\sqrt{N}\rfloor \cdot \lceil\sqrt{N}\rceil$. where, N is the original job size specified by the workload generator. While this does affect the size distribution to some extent, the general shape of the distribution remains the same.

4 Scheduling Strategies

In this section, we describe our scheduling strategies in detail. We also suggest several measures used to evaluate the relative performance of job co-allocation strategies.

4.1 Job Selection Policy

Each of the strategies described in this paper uses the classic **F**irst-**C**ome-**F**irst-**S**erved (FCFS) job selection policy with a slight modification. Specifically, the queue is traversed from head to tail looking for the first job that will fit into the available node sets. This policy is known as **F**it-**P**rocessors-**F**irst-**S**erved (FPFS) ([13]). Traditionally, backfilling, [14] is used in many production grid schedulers, such as Maui. This method will attempt to run jobs in FCFS order, but in the event that the job at the head of the queue *cannot* run due to insufficient resources, it will traverse the queue from head to tail searching for the first job that *can* run given the currently available free resources, provided that by doing so, the start time of the job at the head of the queue is not delayed. This technique is typically used as a means by which to provide a degree of flexibility in backfilling node-time holes in the schedule, while guaranteeing that no starvation takes place.

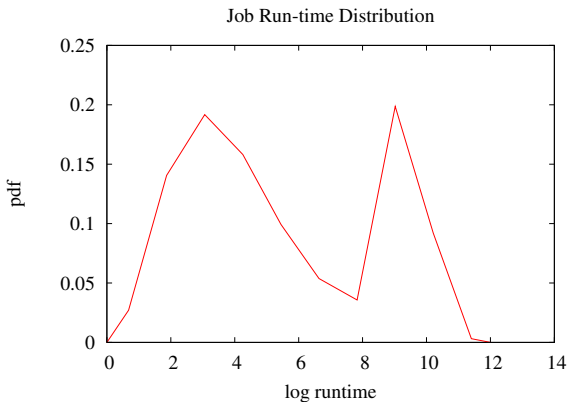

Figure 3. Realistic job run-time distribution

Figure 4. Realistic job size distribution

By making use of the bandwidth-centric communication model, only an estimate of a job's end event is known at any instant in time, since a job's end event can slide forward and backward in time depending on the communication contention in the inter-cluster network links. This makes it difficult to guarantee that the highest priority job's reservation in EASY backfilling will be meet, since we do not terminate jobs; therefore, we do not employ EASY backfilling.

4.2 Co-allocation Strategies

First-Fit The **First-Fit** strategy performs job co-allocation by assigning node resources starting with the cluster with the largest number of free nodes. It then spans as many clusters as necessary to satisfy the job's node requirement. By employing this technique, the number of inter-cluster links over which a given job will span is minimized. It should be noted that this strategy does not make use of any available information regarding a job's communication characterization nor information related to network link saturation. In this sense, it is the most naive.

LSLT Only The **LSLT Only** strategy performs job co-allocation in the same order as First-Fit; however it first identifies all clusters that have links that are saturated beyond a configurable threshold (the Link Saturation Level Threshold i.e. LSLT). It then discounts each of these as potential candidates for job co-allocation; therefore, it will continue to utilize node resources on a given cluster for co-allocation while its network link remains unsaturated. As soon as saturation occurs on a particular network link, this algorithm will then discontinue use of the associated cluster until its saturation level drops. This implies that a link can only be over-saturated to the extent due to a single job's bandwidth utilization. Based on our previous research [9], we found that a threshold value of 70% proved to provide the best 'overall' performance; therefore in this paper, the threshold level has been statically set to 70%. While this algorithm attempts to mitigate network saturation by discounting clusters with saturated network links, it does so without regard to the communication characterization of the given job.

LSLT+CS The **LSLT+CS** performs job co-allocation under the same conditions as the LSLT Only algorithm; however it imposes an additional constraint. In particular, it attempts to co-allocate a sufficiently "large chunk" (e.g. 85% of the node requirement) of the job onto a single cluster. If successful, it will then place the remaining nodes of the job on the remaining clusters, starting with the cluster with the largest number of free nodes first.

This module attempts to capitalize on two primary observations. Since jobs may contain a large degree of all-to-all global communication, the individual bandwidth requirements during co-allocation are minimized when a job is partitioned into a few pieces, one large and perhaps a few small ones. This is in contrast to bisecting the job which results in the maximum bandwidth requirement. However, it may not always be possible to co-allocate a job by partitioning it into at least one "large" piece; which can result in underutilized node resources.

While the algorithm attempts to mitigate network saturation by enforcing a "chunk size" requirement, it does so under the assumptions that the job's communication patterns are primarily all-to-all, and that its PPBW is sufficiently large so as to cause extreme congestion in the inter-cluster links if mapped without regard to partition sizes. Based on our previous research [9], the chunk-size has been statically set to 85%.

ICS After the scheduler has backfilled as many jobs as possible via local execution and remote job migration, it begins considering the remaining jobs for co-allocation. The ICS (Integer Constraint Satisfaction) scheduler ensures that no inter-cluster saturation occurs due to co-allocated jobs. This strategy is an on-line algorithm, in that each time a scheduling decision is made, the scheduler only has access to the information related to the jobs that have arrived thus far, i.e. not for jobs that will arrive in the future.

In order to map jobs onto the multi-cluster in such a way that completely prevents the slowdown associated with over-saturated inter-cluster network links, it is necessary to first determine the range of nodes that a job could potentially acquire on each cluster as a function of the job's bandwidth characterization, as well as the available inter-cluster link bandwidth.In order to accomplish this task, the scheduler must have access to the job's communication characterization, including its per-processor bandwidth requirements requirements as well as current link utilizations. With this information, this scheduler can determine a partitioning that will guarantee that no link saturation occurs, while simultaneously making use of as many available node resources as possible. We solve this graph partitioning problem by using a custom integer constraint satisfaction algorithm. Note that job co-allocation is also subject to the constraints imposed by each local backfill schedule over which it spans. Additionally, it is important to note in our simulations, that our scheduler is assumed to have access to accurate information regarding a job's execution time and bandwidth characterization. This implies that when I backfill schedule is generated, it never changes in response to jobs that finish early or that run longer than expected.

(The interested reader may wish to explore the modeling, design, and implementation in our previous work [9].)

4.3 Baselines for comparison

In order to establish an "reasonable" upper and lower bound for the job turnaround time metric, three baseline simulations were conducted to identify these levels. The first is run under the assumption that the inter-cluster network links have unlimited bandwidth capacities. This configuration, **Zero Comm.** (Zero Communication Cost), represents a "best-case", since there is no slowdown associated with job co-allocation. The second strategy is referred to as **Migration Only**. This strategy only performs job migration, i.e. no job co-allocation. Jobs that are migrated do not contend for inter-cluster network resources. Therefore their ultimate execution times are also unaffected by their bandwidth requirements. Additionally, a third bound, **No Share**, is included that represents the the performance of the multi-cluster when all jobs that arrive to a given cluster *must* run locally. In this configuration, no resource sharing can take place.

The *Zero Comm.* and *Migration Only* therefore appear as horizontal "limits" in the included figures, since they are unaffected by a job's communication characterizations. We have omitted the *No Share* performance in the graphs since we are more interested in the behavior of our scheduling techniques in an operating range between that of *Zero Comm.* and *Migration Only*; however, we have provided it in the text.

5 Experimentation and Results

Each cluster in the multi-cluster consists of 100 homogeneous computational nodes and has a 1000 Mbps inter-cluster network link to the central switch. The workload presented to each cluster consists of 400,000 jobs. Such a large number of jobs were required in order to achieve convergence in the job turnaround time performance metric [3] and to ensure that the queueing system operates in a stable state.

Unless otherwise noted, in these experiments ρ (the fraction of the PPBW that is associated all-to-all communication versus nearest-neighbor) is distributed uniformly $[0, 1]$ in order to provide a "mixture" of jobs ranging from purely local (nearest-neighbor) to purely global (all-to-all) communication characterizations. Additionally, in order to constrain the number of varying parameters, we have held the number of clusters to four.

5.1 Response to Communication Intensity

In our first set of experiments, we observe the relative behavior of our scheduling strategies as a function of increasing job communication intensity. To study the impact of this communication, the jobs are characterized by a per-processor bandwidth, $PPBW$. In this case, we have chosen to hold every job's PPBW constant for a particular run of the simulator. This produces a varying bisection bandwidth, $BSBW$, due to the varying node sizes of jobs within the workload. In these experiments, the ICS scheduler has access to the PPBW and ρ for every job in the presented workload. This represents a best-case scenario, in the sense that each time the ICS scheduler calculates possible partition sizes, it does so with full a priori knowledge of the bandwidth requirements for the given job. The results for these experiments is shown in Figures 5 and 6.

It is worth noting that while the LSLT and LSLT+CS strategies perform fairly well for very small PPBWs, their performance rapidly degrades as job PPBW increases. This is primarily due to the fact that as the PPBW increases, the more over-saturated the associated inter-cluster links become. The decrease in queue waiting time provided by job co-allocation is rapidly overshadowed by the increase in job

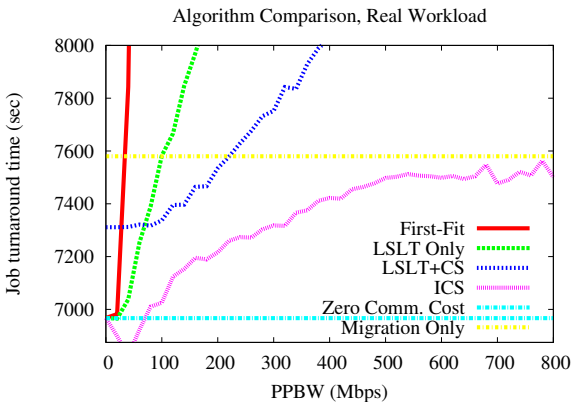

Figure 5. Realistic workload

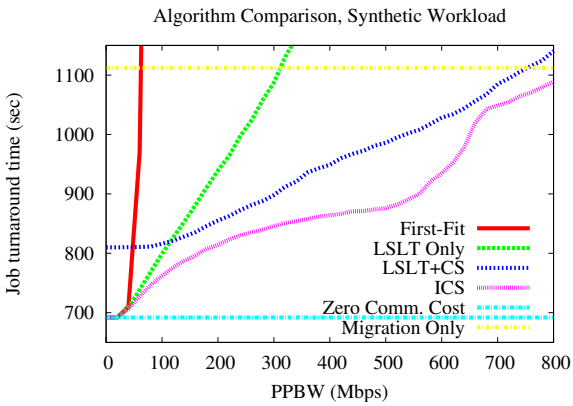

Figure 6. Synthetic workload

execution time. Since these strategies do not have access to a job's communication characterization, they are unable to effectively manage available network resources, especially as the PPBW increases. On the other hand, the ICS scheduler performs much better, even as the PPBW increases. In fact, its performance will never be worse than that of Migration Only. This is due to the fact that as the PPBW increases, the available partition sizes it calculates grow smaller and smaller, and in the limit, the ICS strategy reverts to Migration Only.

Since actual workloads tend to contain a mixture of jobs, with varying PPBWs, a secondary experiment was run to determine the average performance of our algorithms over a range of PPBWs. In particular, in this secondary experiment, jobs were characterized with PPBWs that were uniformly distributed over the interval $[0, 300]$ Mbps. This interval was chosen because is represents a reasonable range of possible sustainable bandwidths on the PCI bus with current GigE NICs. The results of this experiment are summarized in Table 1.

Strategy	Realistic	Synthetic
Migration Only	0%	0%
LSLT+CS	1%	22%
ICS Best	8%	31%
Zero Comm. Cost	9%	36%

Table 1. Improvement over migration only

Note that in the case of the realistic workload, the possible range of improvement over Migration Only is 9%, as opposed to 36% in the case of the synthetic workload. Therefore, while the ICS scheduler improves the performance in both cases, there isn't as much room for improvement with the realistic workload as there is with the synthetic workload. This is due in part by the large number of very small jobs (in the node dimension) present in the realistic workload. Job co-allocation capitalizes on the presence of fragmented node resources present throughout the multi-cluster; however, when a large number of very small jobs are present, many jobs can simply be migrated in order to fill in any "holes" located on remote clusters. This means that co-allocation is not as important when a large portion of the arriving workload are very small jobs. Workload characteristics strongly influence the effectiveness of parallel job scheduling [6], [10]. The effects of some of these key factors are explored in section 5.3.

Furthermore, it is worth noting that Migration Only affords a substantial improvement over No Share (i.e. when all jobs arriving to a cluster must run locally). Specifically, in Migration Only is roughly 60% and 72% faster than No Share for the realistic and synthetic workloads, respectively. It should be noted that the Migration Only model does not account for any overhead such as data-staging that could otherwise result in poorer performance.

5.2 Response to Information Availability

In our second set of experiments, we observe the relative behavior of our scheduling strategies as a function of increasing availability of job communication characterization information. In these experiments, we randomly select jobs from the workload stream to "expose" their PPBW and ρ information to the ICS scheduler. By varying this exposure probability from 0 to 1, we can observe how much information about the arriving workload is necessary to improve performance by a given amount. For example, in the case of the synthetic workload, the ICS scheduler needs roughly 10% of the incomming jobs to expose their communication characterizations in order to improve average job turnaround time to within 50% is it's best performace

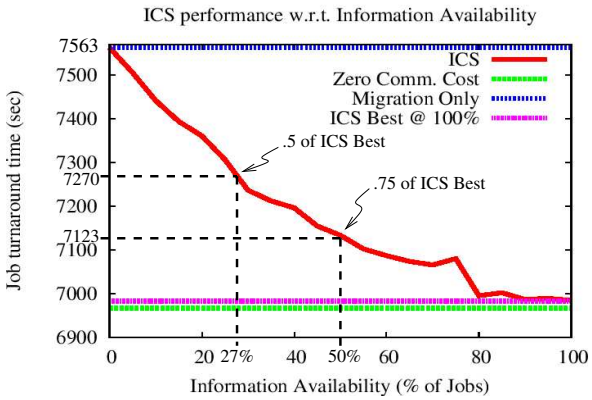

Figure 7. Realistic workload

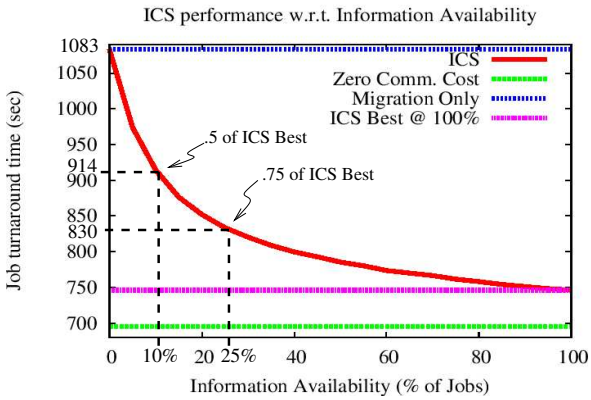

Figure 8. Synthetic workload

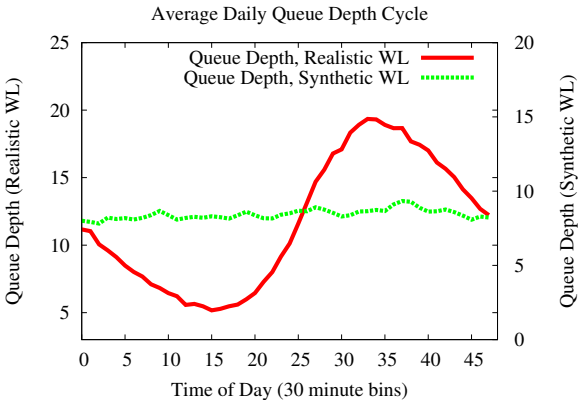

Figure 9. Effect of non-uniform arrival process

(Figure 8). While in the case of the realistic workload, to achieve the same relative level of improvement, the ICS scheduler needs 27% of the jobs to expose their communication characterizations (Figure 7).

These results suggest that the scheduler does not need to have access to the communication characterizations of every job in order to improve the performance of job co-allocation. In fact, for certain workloads, the scheduler only needs a priori communication knowledge of a relatively small fraction of the arriving workload in order to dramatically narrow the gap between the performance of Migration Only and Zero Communication Cost.

5.3 Response to Workload Characteristics

One common pitfall in parallel job scheduling research is to assert the general applicability of a given scheduling strategy based on the performance on a single workload [4].

In this section we demonstrate that the effectiveness of job co-allocation is tightly coupled to salient workload characteristics.

For example, our synthetic workload's arrival process is Poisson, with a constant rate; whereas the realistic workload's arrival rate varies throughout the day. This subjects the system to varying levels of loading, which in turn correlates closely with parameters such as waiting time in the queue and queue depth (Figure 9).

The number of waiting jobs can affect the scheduler's ability to find enough "good" candidates for co-allocation. For example, during periods of lower queue depth, there may not be sufficient jobs in the queue that meet the scheduler's criteria for co-allocation. Likewise, during periods of intense activity, there may be more jobs waiting that meet the scheduler's criteria than it can presently make use of for co-allocation. In this case, the additional jobs do not significantly improve the likelihood of successful job co-allocation and therefore simply lead to larger queue waiting times.

In our last set of experiments, we have modified our workloads to demonstrate that such key factors as arrival process and size distributions can have a dramatic effect on the performance on job co-allocation. Figures 10, 11, 12, and 13 show the performance of the ICS strategy with a modified versions of our original workloads. Workload "A" (Figure 10) shows that by simply replacing the realistic arrival process with a constant rate Poisson process, the potential room for improvement, i.e. the gap between simple Migration Only and Zero Communication Cost, increases from 9% in the case of the unmodified workload, to roughly 28%. Likewise, workload "B" (Figure 11) shows that by replacing the Poisson arrival process of the unmodified synthetic workload with the arrival process of the realistic workload, the potential room for improvement decreases from 36% to 9%. The varying arrival rate of the realistic

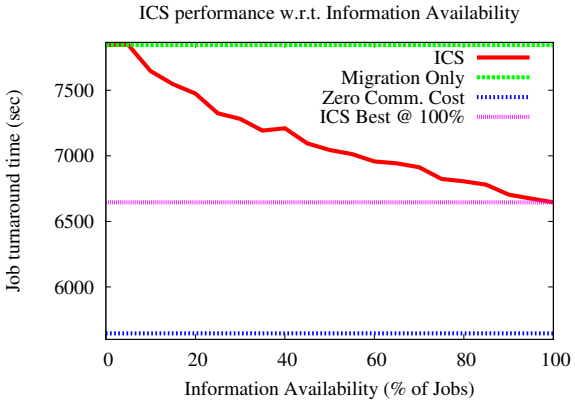

Figure 10. (WL A) Realistic workload with Poisson arrival process

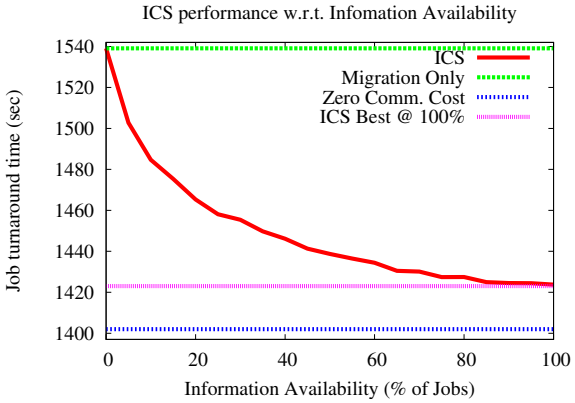

Figure 11. (WL B) Synthetic workload with realistic arrival process

workload has a strong impact on the potential effectiveness of co-allocation.

Additionally, the presence of a large number of small jobs in the arriving workload affects co-allocation effectiveness. Workload "C" (Figure 12) shows the impact of removing all jobs that are smaller than 10 nodes from the realistic workload. In this case, the room for improvement increases from 8% to 14%. Workload "D" (Figure 13) shows the combined impact of removing the small jobs as well as replacing the varying arrival process with the constant-rate Poisson process. In the case, Zero Communication Cost is a full 45% better than Migration Only, indicating a large potential for improvement using co-allocation.

Table 2 summarizes the improvement that is obtained over simple Migration Only, where columns R and S (Realistic and Synthetic) represent the original unmodified workloads, and A, B, C, and D represent the modified workloads. Note that that performance of the ICS strategy is also strongly correlated to the salient workload characteristics.

Strategy	R	S	A	B	C	D
Mig.	0%	0%	0%	0%	0%	0%
ICS	8%	31%	15%	8%	14%	27%
Zero	9%	36%	28%	9%	24%	45%

Table 2. Improvement over migration only

6 Conclusions and Future Work

In this paper, we utilize a bandwidth-centric job communication model that captures the interaction and impact of simultaneously co-allocating jobs across multiple clusters. We make use of a parallel job model that seeks to capture both local and global communication access patterns. By doing so, we are able to explore scheduling strategies that attempt to improve average job turnaround time by selectively mapping jobs across cluster boundaries in a process known as job co-allocation.

In this research, we focus on scheduling strategies that make use of available information such as network link utilization, per-processor bandwidths, and job communication topology in order to make intelligent decisions regarding application partition sizes and job placement. We also provide results that help to establish the relationship between the quantity of information available a priori to the scheduler and its ability to improve overall system performance.

One of the primary conclusions of this paper is that while job co-allocation can dramatically improve the overall performance of the multi-cluster system, its ability to do so depends heavily on the characteristics of the arriving workload stream. Job co-allocation capitalizes on aggregating underutilized node resources from throughout the grid to run parallel jobs earlier. It's ability to improve performance beyond that of simple job migration depends on there being more large (wide) jobs compared to small (narrow) jobs, since small jobs can simply be migrated to remote clusters to make use of any underutilized local node resources; thereby rendering co-allocation unnecessary. Additionally, we demonstrate that the nature of the arrival process can have a dramatic effect on the ability to improve system performance using co-allocation, particularly the daily cycle of job arrivals.

Finally, we demonstrate that having a priori knowledge of a job's communication characterization, including both intensity and topology, is integral to the performance of job co-allocation, especially as the average per-processor bandwidths grow larger.

We are currently working to integrate traditional conser-

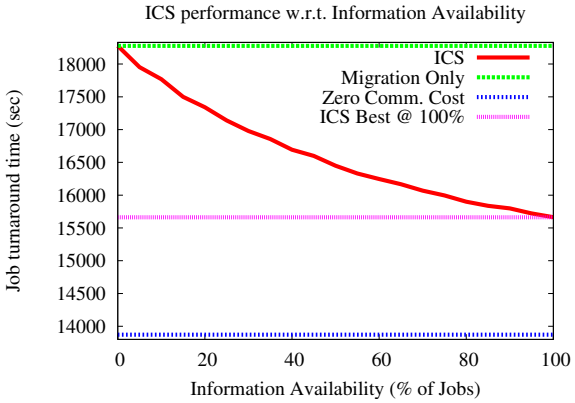

Figure 12. (WL C) Realistic workload with small jobs removed

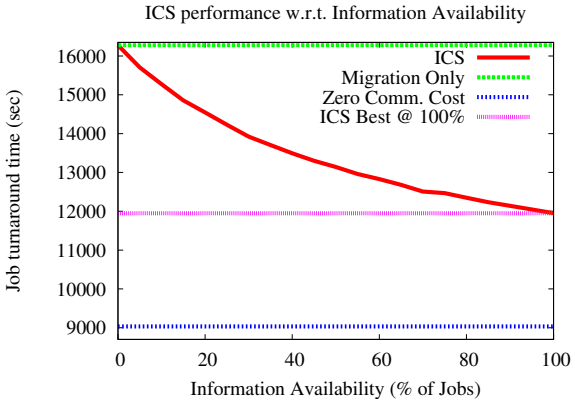

Figure 13. (WL D) Realistic workload with small jobs removed and Poisson arrival process

vative backfilling into our scheduling algorithms in order to ensure fairness among participating clusters during periods of disparate loading conditions. We also plan on developing models to address the impact that data staging to-and-from remote clusters has on the overall performance of the system. Additionally, we are planning to integrate models for job check-pointing that will allow us to address issues such as run-time job migration/re-coallocation, as a means by which to recover from bad scheduling decisions that cause unnecessary network congestion.

References

[1] A. I. D. Bucar and D. H. J. Epema. The Performance of Processor Co-Allocation in Multicluster Systems. In *3rd International Symposium on Cluster Computing and the Grid*, pages 302–309, May 2003.

[2] C. Ernemann, V. Hamscher, A. Streit, R. Yahyapour, and U. Schwiegelshohn. On Adgantages of Grid Computing for Parallel Job Scheduling. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02) Berlin Germany May 21*, pages 31–38, 2002.

[3] D. G. Feitelson. Metrics for Parallel Job Scheduling and Their Convergence. In *Job Scheduling Strategies for Parallel Processing*, volume 2221, pages 188–206, 2001.

[4] D. G. Feitelson and E. Frachtenberg. Pitfalls in parallel job scheduling evaulation. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*. Springer Verlag, 2005.

[5] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn. Parallel job scheduling – a status report. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*. Springer Verlag, 2005.

[6] E. Frachtenberg, D. G. Feitelson, J. Fernandez, and F. Petrini. Parallel job scheduling under dynamic workloads. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 208–227. Springer Verlag, 2003. Lect. Notes Comput. Sci. vol. 2862.

[7] BeoSim Website. http://www.parl.clemson.edu/beosim.

[8] W. M. Jones. Improving Parallel Job Scheduling Performance in Multi-clusters Through Selective Job Co-allocation. PhD Dissertation – Clemson University, December 2005.

[9] W. M. Jones, L. W. Pang, D. Stanzione, and W. B. Ligon III. Characterization of Bandwidth-aware Meta-schedulers for Co-allocating Jobs Across Multiple Clusters. In *Journal of Supercomputing, Special Issue on the Evaluation of Grid and Cluster Computing Systems*, volume 34, pages 135–163. Springer Science and Business Media B.V, November 2005.

[10] V. Lo, J. Mache, and K. Windisch. A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 25–46. Springer Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.

[11] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *J. Parallel Distrib. Comput.*, 63:1105–1122, 11 2003.

[12] Parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload/.

[13] J. Sinaga, H. Mohamed, and D. H. J. Epema. A Dynamic Co-Allocation Service in Multicluster Systems. In *10th Workshop on Job Scheduling Strategies for Parallel Processing (in conjunction with Sigmetrics-Performance 2004)*, New York, June 2004.

[14] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Characterization of backfilling strategies for parallel job scheduling. In *IEEE International Conference on Parallel Processing Workshops*, pages 514–519, August 2002.