Job Communication Characterization and its Impact on Meta-scheduling Co-allocated Jobs in a Mini-grid

William M. Jones[†]
864-656-7367
wjones@parl.clemson.edu

Louis W. Pang[†]
864-656-7367
plouis@parl.clemson.edu

Dan Stanzione[‡]
703-292-8121
dstanzio@nsf.gov

Walter B. Ligon III[†]
864-656-1224
walt@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

[‡]National Science Foundation
Division of Graduate Education
4201 Wilson Blvd Suite 907
Arlington, VA 22230

Abstract

In this paper, we present a bandwidth-centric parallel job communication model that takes into account inter-cluster network utilization as a means by which to capture the interaction and impact of simultaneously co-allocated jobs in a mini-grid. Our model captures the time-varying utilization of shared inter-cluster network resources in the grid. We compare our dynamic model with previous research that utilizes a fixed execution time penalty for co-allocated jobs. We have found that the fixed penalty model is more generous in its prediction of job turnaround time than our dynamic communication model. Additionally, we see that the penalty co-allocated jobs may experience without causing a severe performance degradation decreases as the number of clusters increases.

1 Introduction

Clusters of commodity processors have become fixtures in research laboratories around the world. In many larger laboratories, universities, and research parks, multiple clusters likely exist on the same campus. This co-location of several resource collections naturally lends itself to the formation of a mini-grid [7].

A mini-grid is distinguished from a traditional computational grid in that the mini-grid utilizes a dedicated interconnection network between grid resources with a known topology and predictable performance characteristics. This type of networking infrastructure allows for the possibility of mapping jobs across cluster boundaries in a process known as *co-allocation* or *multi-site scheduling*. In this paper, we develop a parallel job model that takes both computation and communication into account as a means by which to explore co-allocating grid schedulers that exploit these unique architectural features. The main focus of this paper is to present an in-depth explanation of our communication model and its associated algorithms as well as to provide a brief study the impact of co-allocation in a mini-grid as a function of job communication characteristics and grid scheduling routines.

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], [2] or by assigning co-allocated jobs a fixed penalty [3], [4]. 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 mini-grid, 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*.

2 Computational Mini-Grids

At first glance, mini-grids may appear to be distinguished from conventional computational grids only in scope. A mini-grid is limited to a campus-wide setting, for example, while a traditional grid is national or even global in scope. However, upon further inspection, a mini-grid has a distinctive architectural feature; the internal networks of the clusters are bridged together through dedicated links. This has several important implications. First, there exists predictable, reliable bandwidth between grid resources, as contrasted with Internet connected grids. This should allow for a scheduler capable of making better decisions in co-allocating jobs across these resources. Additionally, the mini-grid architecture makes finer grain control of the resources within the grid more practical. When fast, low latency links are available, loaning a single node or a small number of nodes from one cluster to another is a low overhead operation.

An example of such a grid is the Clemson computational mini-grid (Figure 1) which consists of four interconnected Beowulf clusters, ranging in size from 64 to 256 processors. Each node in each cluster is connected by dual fast Ethernet connections to the cluster switch. The cluster switches are, in turn, connected together via multiple trunked gigabit Ethernet connections. Nodes can be "loaned" to other clusters in the grid, thus allowing grid resources to be temporarily reallocated to better meet collective needs across the grid.

The initial motivation for this research was an interest in developing new scheduling and resource management software for our own computational mini-grid. A discrete event-driven simulator, known as *Beosim* (Section 5), was developed in order to study the effects of various scheduling routines and explore the behavior of a mini-grid under a variety of workload characterizations.

3 The Model

In this section we characterize the parallel job model as well as the mini-grid architecture. We provide a detailed explanation of the communication model used, as well as a strategy to account for the time-varying inter-cluster network utilization.

3.1 Mini-grid Model

We consider the mini-grid to be a collection of arbitrary sized clusters with globally homogeneous nodes. Each cluster has its own internal ideal switch. Additionally, the clusters are connected to one another via a central ideal switch. This implies that each cluster has a single dedicated link to the central switch. We assume that each node in the mini-grid has a single processor and a single network interface card.

3.2 Parallel Job Model

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 [9] is employed in mapping the job onto the mini-grid, 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 execution time of a given job when it is co-allocated in the mini-grid. In particular, $T_E = T_P + T_C$. 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.

3.3 Communication Characterization

Each job modeled in this study performs all-to-all global communication patterns throughout its execution, and in general, each node in a given job, j, is characterized by an average per-processor bandwidth requirement, $PPBW_j$. Since jobs can be co-allocated in the mini-grid, nodes must therefore communicate across cluster boundaries. This communication will require a certain about 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. In order to determine when the inter-cluster links become saturated, we must first identify how much bandwidth a job will require in order to run at full speed. The amount of bandwidth, BW_i^j, required by job j on inter-cluster link i is given by equation 1, where n_T^j is the total number of nodes required by job j and n_i^j is the number of nodes allocated to job j on cluster C_i. The first factor in this equation represents the fraction of the messages generated by a single node on cluster, C_i, that are destined for non-local nodes.

$$BW_i^j = \left(\frac{n_T^j - n_i^j}{n_T^j - 1} \right) (PPBW_j * n_i^j) \qquad (1)$$

The second factor is total bandwidth required by all the nodes associated with job j on cluster C_i. Once a job

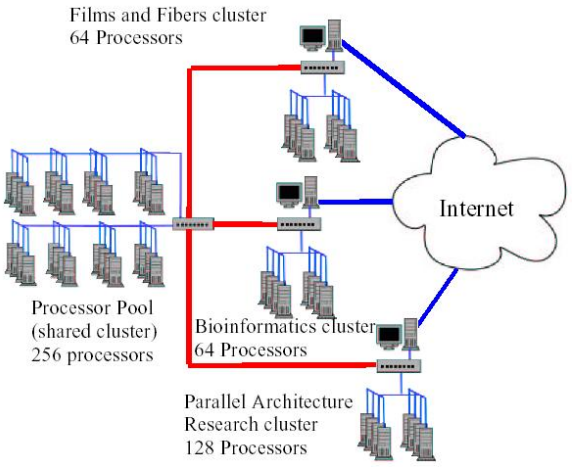

Figure 1. Clemson mini-grid

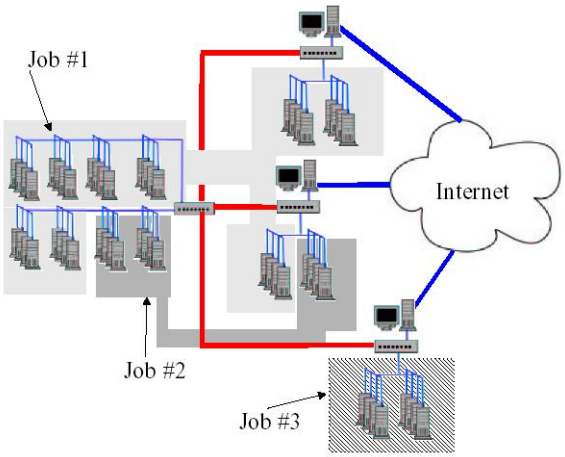

Figure 2. Job co-allocation in a mini-grid

has been mapped to the mini-grid, the required bandwidth, BW_i^j, is calculated for each link, i. This amount is then aggregated with the bandwidth required by every other job that shares this link. Using this quantity, the co-allocated jobs' residual times to completion are recalculated for each event that causes a state change in any inter-cluster links. The details for these recalculations are provided in the subsections that follow.

3.4 Job Co-allocation

We consider co-allocation (Figure 2) to be the mapping of a job across cluster boundaries. One possible reason that a job might be co-allocated is due to the natural fragmentation that occurs within each cluster in the node dimension. Suppose for example that a job is waiting in a cluster's ready queue. This job may require more nodes than are presently available on its particular cluster, but collectively there may be enough available nodes elsewhere in the grid to accommodate the job. The job would be considered co-allocated if it were mapped onto nodes that were "borrowed" from other clusters.

3.5 Intra-cluster Bandwidth Saturation

When jobs are co-allocated, their effective execution times may be altered due to the inherent bottleneck that is created in the mini-grid's interconnection network. The degree to which the job's runtime is affected depends on several factors. Clearly the time-varying utilization of the dedicated links connecting the clusters plays an important role. Additionally, the amount of communication that each co-allocated job produces can considerably affect not only its own execution time, but also that of every other co-allocated job that shares any network resources with it.

The first step in determining the impact of co-allocation is to identify the presence and location of communication bottlenecks in the inter-cluster links. The residual time to completion for a particular job can change in response to two events:

- a new job is co-allocated in the mini-grid
- a co-allocated job terminates and thus frees network resources

Each inter-cluster link, i, is characterized by a maximum bandwidth rating, BW_i^{max}. An initial measure of the saturation of each link is calculated by taking the ratio of the maximum available bandwidth to the total bandwidth required for every job that spans that link. The saturation ratio is given by equation 2

$$BW_i^{sat} = \frac{BW_i^{max}}{\sum_{\forall j \in J_i} BW_i^j} \tag{2}$$

where set J_i is the set of all jobs that span link i. If $BW_i^{sat} \geq 1.0$ then link i is *not* saturated, otherwise if $(0.0 \leq BW_i^{sat} < 1.0)$, then link i *is* saturated. If a given link i is saturated, then each job in J_i will not be able to receive the amount of bandwidth it requires to run at full speed. In order to calculate the impact on each job due to co-allocation, the fraction of bandwidth each job receives compared to the amount it requires must be determined.

Each time a new job is co-allocated or when a co-allocated job terminates, the algorithm below is applied in order to determine the amount of bandwidth ultimately alloted to each job on each link. In the following algorithm, equation 3 is used to account for the residual saturation level of the inter-cluster links due only to the jobs that have not

yet been constrained.

$$BW_i^{uc\_sat} = \frac{BW_i^{avail}}{\sum_{\forall j \in J_{uc}} BW_{(i,j)}^{alloted}} \quad (3)$$

Step 1: *Initialization* - For every job j, let $BW_{(i,j)}^{alloted} = BW_i^j$. For every link i, let $BW_i^{avail} = BW_i^{max}$. Let the unconstrained set of nodes, $J_{uc} = J$ (set of all jobs). Let the set, J_i be the set of all jobs that span link i.

Step 2: *Saturation detection* - For every link, calculate $BW_i^{uc\_sat}$. While there exists at least one $BW_i^{uc\_sat} < 1.0$, continue, else goto **Step 5**.

Step 3: *Saturation correction* - Identify the link with the smallest $BW_i^{uc\_sat}$ (most saturated link) from **Step 2**, and globally reduce the alloted bandwidth of every job in $J_i \cap J_{uc}$ by a factor of BW_i^{sat}.

Step 4: *Update state* - Remove each of the modified jobs from the set J_{uc}. For each of the modified jobs, remove their alloted bandwidth from the available bandwidth, BW_i^{avail} on each link over which they span. Goto **Step 2**.

Step 5: *Termination* - DONE.

After this algorithm is applied, the alloted bandwidth, $BW_{(i,j)}^{alloted}$, for each job will either be its initially requested bandwidth, BW_i^j for full speed execution, or it will be some fraction of its required bandwidth. If the job is alloted its required bandwidth, it will not experience any slowdown associated with communication for the duration of time between the current event and the next inter-cluster state changing event. However, if the job does not receive its required bandwidth, it will experience a slowdown in its residual communication time that is proportional to the disparity between its required and alloted inter-cluster bandwidths.

3.6 Co-allocated Job Communication Slowdown

Each affected job's bandwidth allotment on each link is reduced in order to accommodate the most saturated link over which it spans. This bottleneck uniquely determines the disparity between the job's required and alloted bandwidths on each link. This implies that a job's ratio of the alloted to required bandwidth is the same for each link over which the job spans. Equation 4 formalizes the bandwidth slowdown associated with job j, where link k may be any link over which the job spans.

$$BW_{(k,j)}^{sd} = \frac{BW_{(k,j)}^{alloted}}{BW_k^j} \quad (4)$$

3.7 Residual Execution Times

Now that the communication slowdown factor is known, the residual execution time, T_E^R, of a job can be calculated as a function of both the residual communication and computation times (T_C^R and T_P^R respectively). Its associated end-event can then be rescheduled in the simulator to account for the state change in the inter-cluster network. In particular, equations 5 and 6 illustrate the calculation required to determine the residual execution time of job j, where the primed terms represent quantities from the previous inter-cluster state changing event, while the non-primed values represent quantities for the current state change event.

$$T_E^R = \overbrace{(T_C^{R'} - T_C^{\Delta})(BW_{(k,j)}^{sd'})(BW_{(k,j)}^{sd})^{-1}}^{T_C^R} + \underbrace{(T_P^{R'} - T_P^{\Delta})}_{T_P^R} \quad (5)$$

where

$$T_P^{\Delta} = \frac{\Delta T}{T_E^{R'}} T_P^{R'} \quad , \quad T_C^{\Delta} = \frac{\Delta T}{T_E^{R'}} T_C^{R'} \quad (6)$$

The T_P^{Δ} and T_C^{Δ} terms represent the times spent doing computation and communication respectively during the interval since the last state change event. These quantities can then be subtracted from the previous residual computation and communication times. The communication term is then scaled to take into account the slowdown due to its most saturated inter-cluster network link, as seen in equation 5.

When a job is initially co-allocated, its residual computation (T_P^R), communication (T_C^R), and execution (T_E^R) times are initialized to T_P, T_C, and T_E respectively from its original profile. As inter-cluster state changing events occur, the residual times are recalculated based on equations 5 and 6. Due to these recalculations, the job's end-event can slide forward or backward in time, reflecting either a degradation or improvement in saturation levels of the inter-cluster links over which it spans.

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.

3.8 Workload Generation

Although our simulator is capable of ingesting actual workload trace-files, in this paper we make use of synthetically generated workloads. In particular, we assume 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 $D_i^{nodes} \sim UNIF[n_1^i, n_2^i]$. The fraction of *the total execution time* that initially represents computation is set to a constant, K_i, for all jobs.

4 Meta-scheduling Algorithms

In general, we consider a meta-scheduler to be the software, or collection of software, that decides where, when, and how to schedule jobs in a grid. A meta-scheduler is expected to work in conjunction with the local schedulers working on each individual cluster. In this paper, we assume that the meta-scheduler is globally aware of the state of the mini-grid. In order to address the impact of job co-allocation on the performance of the mini-grid, the following strategies and policies are analyzed.

4.1 Strategies and Policies

To establish a base-line performance, the first algorithm treats the mini-grid as if it were a collection of disjoint clusters (i.e., the grid didn't exist). In this scenario all jobs that arrive to a given cluster run in their entirety on their *home* cluster. We refer to this strategy as **NO_SHARING**.

The second algorithm is slightly more flexible. Although it does not perform any job co-allocation, it does allow jobs to run in their entirety on any cluster in the mini-grid. It will attempt to allocate resources to a job in best fit fashion. We refer to this strategy as **MIGRATION_ONLY**.

The third algorithm performs job co-allocation by employing a best fit allocation of available grid resources spanning as many clusters as necessary to satisfy the job's node requirement. By employing best fit, the number of inter-cluster links over which a given job will span is minimized. We refer to this strategy as **BFFF**.

Each of the three strategies described above 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 (we call this *FCFS_SCAN*). Traditionally a policy commonly known as *EASY backfilling* [8] is used in many production grid schedulers, such as Maui [6]. 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.

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. In our initial experiments, we use the FCFS_SCAN approach described above as a first pass in evaluating the impact of communication on job co-allocation.

5 Simulation

Beosim [7] is a discrete event driven simulator designed to model a mini-grid as a collection of (possibly heterogeneous) Beowulf clusters connected via a dedicated interconnection network. Beosim can be driven via synthetic workload distributions that are characterized through the use of randomly generated arrival and service processes. Additionally, Beosim has the capability of ingesting actual workload trace-files from a variety of supercomputing centers.

In order to study the effects of various scheduling routines, Beosim was designed to allow the scheduler modules to be swapped either "on the fly", or from run to run, in order to compare and contrast their respective performance impacts.

5.1 Experimental Parameters

This subsection describes the set of experimental parameters used in all simulations in this paper. Each cluster in the mini-grid 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 4,000,000 jobs. Such a large number of jobs were required in order to achieve convergence in the job turnaround time performance metric [5]. The number of nodes each job requires is taken from a uniform distribution $UNIF[10, 90]$ (nodes). The job arrival process is Poisson with the inter-arrival time taken from an exponential distribution with parameter 150 (sec). The base execution time of each job is taken from an exponential distribution with parameter 225 (sec). For simplicity, the computation fraction is uniformly set to $K = 0.7$ for all jobs that arrive to the mini-grid.

In order to study to impact of communication, the jobs must be characterized by a per processor bandwidth $PPBW$. We chose to hold every job's bisection bandwidth constant for a particular run of the simulator. This produces a varying $PPBW$ due to the varying node sizes of jobs in the workload. We calculate the $PPBW$ given the bisection bandwidth, $BSBW$, using equation 7, which is obtained from our model in equation 1.

$$PPWB_j = BSBW\left(\frac{4(N_T^j - 1)}{(N_T^j)^2}\right) \quad (7)$$

5.2 Experimental Setup

For the research presented in this paper, we were interested in comparing the results obtained from our dynamic communication model with that of previous research that utilizes a fixed penalty for co-allocated jobs [3], [4]. Using the parameters specified in Section 5.1, we conducted three distinct simulations, a 2, 4, and 8 cluster simulation. In each simulation the bisection bandwidth of the job workload is varied over a particular range, where the average job turnaround time and average co-allocated job penalty in the grid is measured. The penalty is calculated as the ratio of how long the job actually ran to its original execution time.

In order to establish an upper and lower bound for the job turnaround time metric, two baseline simulations were conducted to identify these levels. The first is the **NO_SHARING** strategy described in Section 4.1. The second is the BFFF strategy with the computation fraction K set to 1, which we refer to as **NO_COMM_PENALTY**. Since the NO_SHARING strategy does not allow job migration or co-allocation, it represents a "worst-case" scenario that can be seen as an upper bound for average job turnaround time. In the NO_COMM_PENALTY simulations, the original execution times for all jobs that arrive to the grid is purely computational. This implies that job co-allocation can be performed with no impact to average job turnaround time. This scenario essentially assumes that the inter-cluster links have unlimited bandwidth capacities, therefore NO_COMM_PENALTY represents a "best-case" scenario that can be seen as a lower bound for average job turnaround time. Both the NO_SHARING as well as the NO_COMM_PENALTY strategies therefore appear as horizontal "limits" in each of the figures, since they are unaffected by a job's bisection bandwidth.

Additionally, for each of the three primary simulations, a third strategy, **MIGRATION_ONLY**, is shown. 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 bisection bandwidth.

6 Results

In this section, the results from the three primary simulations are presented. In each figure, the bisection bandwidth (BSBW) of the jobs arriving to the grid is varied over an area of interest. In particular, the BSBW ranges were chosen to show the behavior of the average job turnaround time in the grid, as it approaches the NO_SHARING performance. The BFFF strategy is used in conjunction with the dynamic communication model developed in this paper to generate the data associated with the **BFFF_DYNAMIC_COMM** curve.

For each iteration of the simulation (i.e. for a particular BSBW), the average penalty co-allocated jobs experience is also calculated. This penalty in then fed into another instance of the simulation using the fixed penalty model, where every job that is co-allocated experiences an increase in its original execution time by a factor equal to the measured co-allocation penalty. The data that was generated by using the fixed penalty model is shown as **BFFF_FIXED_COMM** in each figure.

6.1 Observations

The first general observation we make is that in every simulation, the use of the dynamic communication model is not nearly as generous as the fixed penalty model. Although the actual measured co-allocation penalties from the dynamic communication runs are fed directly into the fixed penalty model (i.e. the average co-allocation penalty is identical in both cases), there is a significant difference between the average job turnaround times predicted with each model. The dynamic model accounts for the time varying utilization of the inter-cluster network links, and therefore captures some of the essence of simultaneously co-allocated job interactions in the network.

Additionally, in each scenario (2, 4, 8 clusters), the ability to initially migrate jobs (MIGRATION_ONLY) to a remote cluster provides a rather large performance gain over NO_SHARING. In fact, as the number of clusters increase, so does the performance gain associated with simple job migration. Certainly the gain measured here underestimates the impact associated with activities such a data staging, etc, but we feel that this overhead is relatively small in the mini-grid context can can be accounted for in future work.

By plotting the co-allocation penalty versus turnaround time, it becomes obvious that as the number of clusters increases, the average penalty (using both the dynamic and fixed communication models) that co-allocated job may experience decreases from around 1.2 to 1.25 in the two cluster case, to around 1.13 to 1.2 in the eight cluster case, when compared to the MIGRATION_ONLY strategy. Additionally, when compared to the NO_SHARING strategy, the acceptable co-allocation penalty also decreases, from 1.35 to 1.4 in the two cluster case, to around 1.25 to 1.35 in the eight cluster case.

7 Conclusions and Future Work

In this paper, we have presented a bandwidth-centric job communication model that is capable of taking into account

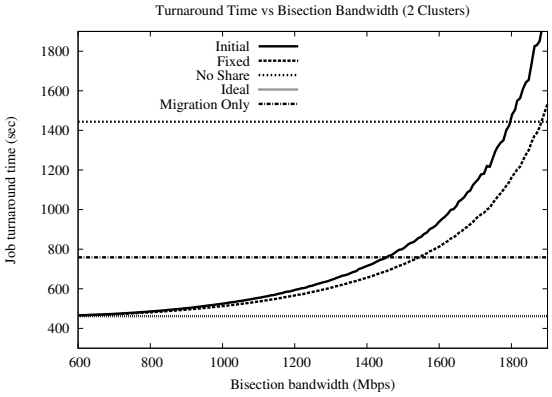

Figure 3. Job Turnaround – 2 Clusters

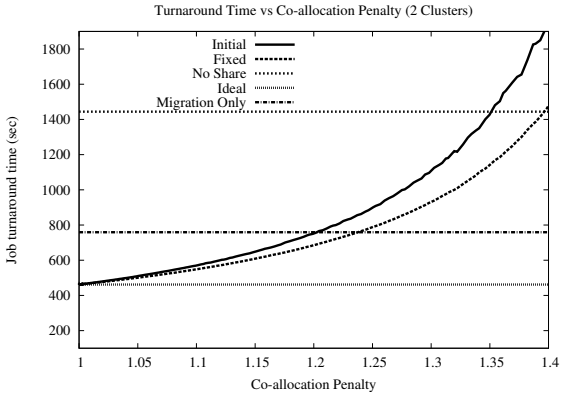

Figure 4. Job Turnaround – 2 Clusters

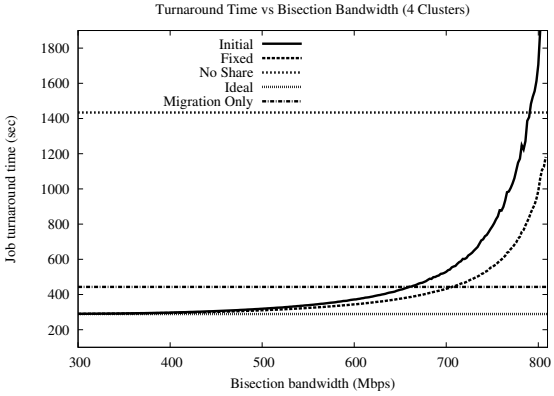

Figure 5. Job Turnaround – 4 Clusters

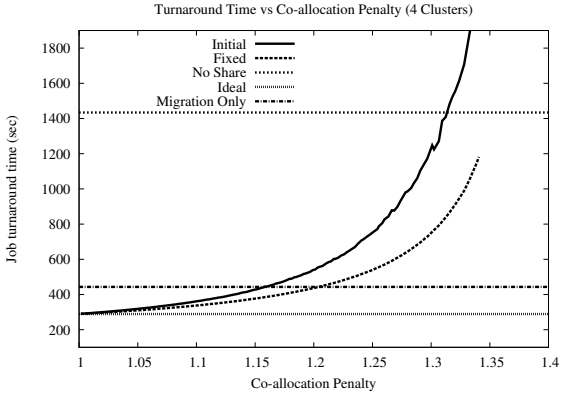

Figure 6. Job Turnaround – 4 Clusters

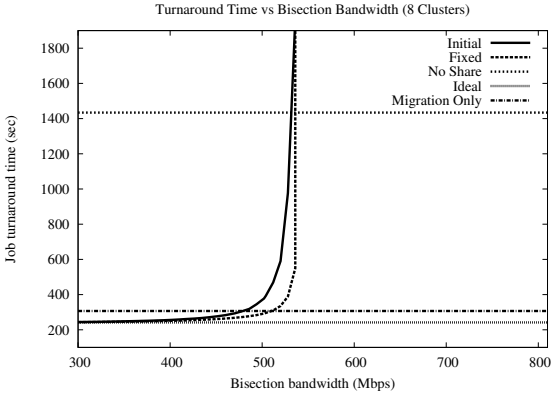

Figure 7. Job Turnaround – 8 Clusters

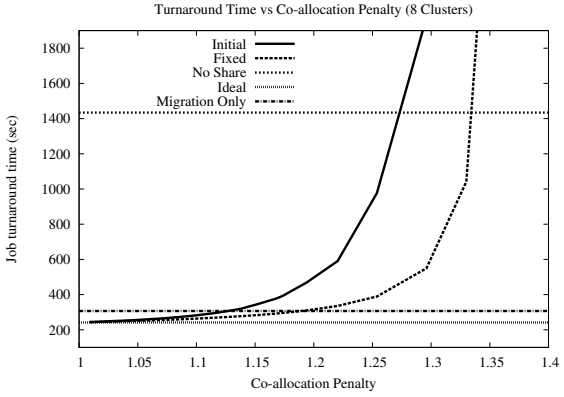

Figure 8. Job Turnaround – 8 Clusters

time-varying network utilization as a means by which to capture the interaction and impact of simultaneously co-allocated jobs in a mini-grid. We have compared our dynamic model with previous research that utilizes a fixed execution time penalty for co-allocated jobs. We have found that the fixed penalty model is more generous in its prediction of job turnaround time than is our dynamic communication model. Additionally, we see that the penalty that co-allocated jobs can experience without causing a severe performance impact decreases as the number of clusters increase.

One of our primary goals is to develop intelligent meta-schedulers that can take advantage of the unique architectural features present in a mini-grid. Our future work will include developing meta-schedulers that monitor the saturation levels of the inter-cluster network in order to make more informed decisions about when and where to map jobs across cluster boundaries. We also plan to augment our communication model to capture the essence of the tree-based algorithms that are commonly used to implement many collective communication patterns. Additionally, we plan to study the overhead associated with job migration as a means by which improve our job communication model.

8 Acknowledgments

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.

References

[1] A. I. D. Bucar and D. H. J. Epema. The Influence of Communication on the Performance of Co-Allocation. In *7th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 66–86. in conjunction with ACM Sigmetrics 2001, June 2001.

[2] 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.

[3] C. Ernemann, V. Hamscher, A. Streit, and R. Yahyapour. Enhanced Algorithms for Multi-Site Scheduling. In *Grid Computing - GRID 2002, Third International Workshop, Baltimore, MD, USA, November 18, 2002, Proceedings*, pages 219–231, 2002.

[4] 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.

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

[6] D. Jackson, Q. Snell, and M. Clement. Core Algorithms of the Maui Scheduler. In *7th Workshop on Job Scheduling Strategies for Parallel Processing*. In conjunction with ACM Sigmetrics 2001, June 2001.

[7] W. M. Jones, L. Pang, and D. Stanzione. Computational Mini-Grid Research at Clemson University. Technical Report PARL 2002-009, PARL, Clemson University, November 2002.

[8] 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.

[9] Y. Zhang, H. Franke, J. Moreira, and A. Sivasubramaniam. An Integrated Approach to Parallel Scheduling Using Gang-Scheduling, Backfilling, and Migration. In *IEEE Transactions On Parallel and Distributed Systems*, volume 14, pages 236–247, March 2003.