Ensuring Fairness Among Participating Clusters During Multi-site Parallel Job Scheduling

William M. Jones[†] and Walter B. Ligon, III
Electrical and Computer Engineering Department
Clemson University
105 Riggs Hall, Clemson, SC 29634-0915
{wjones, walt}@parl.clemson.edu
http://www.parl.clemson.edu

Abstract

Multi-cluster schedulers can dramatically improve average job turn-around time performance by making use of fragmented node resources available throughout the grid. By carefully mapping job's across potentially many clusters, jobs that would otherwise wait in the queue for local cluster resources can begin execution much earlier; thereby improving system utilization and reducing average queue waiting time.

In this paper, we demonstrate that these multi-site scheduling techniques can be successfully integrated with fairness policies to ensure that participation in the multi-cluster is beneficial under extremely disparate workload intensities. Furthermore, we demonstrate that the trade-off between fairness and performance is relatively small.

1. Introduction

As cluster computing becomes more commonplace, industrial and academic research parks can often be found purchasing several clusters to meet their needs, either for an individual group or for several disjoint departments. These geographically co-located clusters can be connected via an interconnection network to form a larger computational resource known as a multi-cluster or super-cluster. The internal networks of the clusters are bridged together through dedicated links. In doing so, there exists predictable, reliable bandwidth among cluster resources, as contrasted with Internet-connected grids. This configuration allows considerable flexibility for distributing parallel jobs among available clusters; however, it also dramatically increases the complexity of effectively managing both computing and networking resources.

As multi-cluster systems become more prevalent, techniques for efficiently exploiting these resources become increasingly significant. A critical aspect of exploiting these systems is the challenge of scheduling. In particular, multi-cluster schedulers must address not only node resource and inter-cluster network utilization, but also fairness among participating clusters. Intelligent schedulers can make use of information related to job communication structure and inter-cluster bandwidth availability to improve average job response time by selectively mapping parallel jobs across potentially many clusters in a process known as job co-allocation or multi-site scheduling [1], [2].

However, one of the caveats of this type of resource sharing is that more heavily-loaded clusters can easily overwhelm the other participants with excess jobs to the extent that their participation in the multi-cluster no longer results in a net performance improvement for each cluster. Therefore, in this paper we demonstrate that these multi-site scheduling techniques can be successfully integrated with fairness policies to ensure that participation in the multi-cluster is beneficial under extremely disparate workload intensities. Furthermore, we demonstrate that the trade-off between fairness and performance is relatively small.

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 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 [4]. The interested reader

[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. [†]Corresponding author.

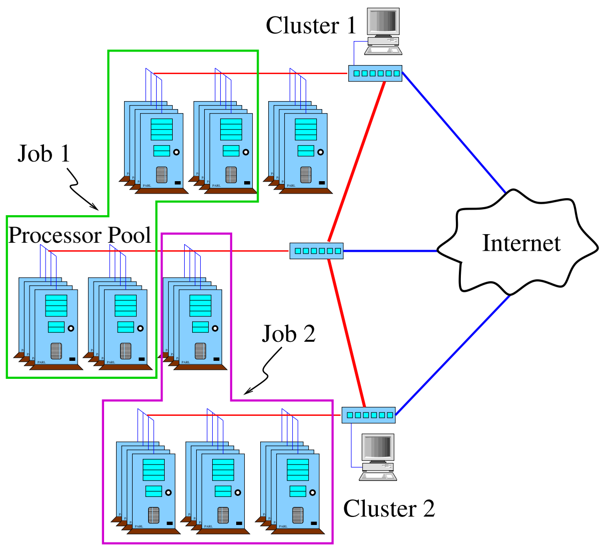

Figure 1. Job Co-allocation

can obtain a detailed treatment of the complete modeling methodology on our website.

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. 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.

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. (The full description is rather lengthy and can be found in [4].)

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. Addressing Fairness

In this paper, we qualitatively view fairness as an indication of how well an individual cluster's jobs perform compared to their performance in isolation. For example, one would generally choose to participate in a multi-cluster provided that there is some tangible benefit in doing so. One benefit that immediately comes to mind is being able to run jobs at a scale larger than is possible on the originating cluster. While this is an extremely interesting case, it is not the focus of this paper. The second obvious benefit is the improvement to queue waiting time, thus resulting in shortened average job turn-around time. Note that this definition of fairness differs from others used in the field [6].

In our previous work [4], we evaluate the relative performance of multi-cluster parallel job schedulers under that assumption that all participating clusters are the same size and that all experience the same arrival workload intensity. This implies that each cluster is contributing not only

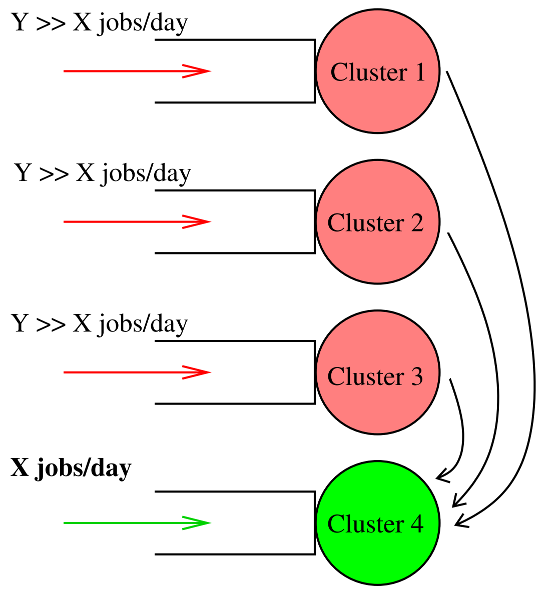

Figure 2. Heavily-loaded clusters overwhelm participating cluster

the same number of resources to the multi-cluster but also the same workload distribution. Under these assumptions, we demonstrated that the migration and co-allocation techniques we employed result in not only an improved average job turnaround time for the multi-cluster as a whole, but also for each cluster individually; therefore, participation was beneficial for each cluster.

Now suppose, for example, that out of four clusters participating, three have an arrival rate that far outstrips that of the remaining cluster. Without the benefit of mechanisms to ensure fairness, nothing would prevent the remaining cluster from being overwhelmed by the excess jobs from the first three clusters (Figure 2). In fact, there is a point where the remaining cluster will experience even worse performance than if it were not participating in the multi-cluster at all. That is to say, on average, it is contributing more node-time to the grid than it is receiving from the grid. This behavior is not acceptable; therefore, a change in the way migration and co-allocation is performed is absolutely necessary to ensure that any individual cluster does not become overwhelmed with jobs from more heavily-loaded participating clusters.

In order to accomplish goal, we have made use of traditional job backfilling (Section 3.1) and job reservation schedules. Backfilling is commonly used to achieve fairness among jobs on a single cluster and to prevent starvation. In this research, we make use of job backfilling to achieve fairness among participating clusters.

3.1. Conservative Backfilling

A primitive scheduler may run jobs in a strict **F**irst-**C**ome-**F**irst-**S**erved (FCFS) order. While this approach ensures fairness, it does not make effective use of idle resources. Backfilling is a technique by which jobs are selected to run out of order to improve resource utilization and lower the average job response time. For example, suppose that jobs are being serviced in FCFS fashion. Now suppose that the job at the head of the queue can not run due to a lack of node resources; but that other jobs in the queue could potentially run given access to the currently available resources.

This general technique has been extensively studied, as it evidenced by many papers published in [3]. Typically a job is only allowed to run out of order provided that in doing so, the start times of the jobs ahead of it in the queue will not be delayed. One special case of backfilling, known as EASY or aggressive, only guarantees that the job at the head of the queue is not delayed, whereas a more conservative approach could potentially guarantee that every job ahead of the given candidate for backfilling is not delayed. The number of jobs that obtain a reservation in the backfill schedule is often referred to as the backfill depth.

The primary trade-off of backfilling is the issue of fairness versus performance. It should be noted that backfilling assumes accurate job execution time estimates in order to guarantee that jobs will start on-time; otherwise, the scheduler may be forced to kill jobs that exceed their estimated execution times by more than an allowable amount. In this paper, we employ conservative backfilling (with backfill depth of 10) to control non-local node resource usage (Figure 3). A backfill depth of 10 was chosen as a moderately conservative depth for two primary reasons. It gives local jobs more security with their reservation schedules in that none of the first 10 local jobs waiting in the queue can be pushed back in time as a result of non-local node usage. This helps provide local jobs more priority over non-local jobs thereby increasing the degree of fairness they experience as participants in the multi-cluster. Secondly, since we are running simulations on the order of 400K jobs per cluster, keeping the backfill depth to 10 keeps the per-simulation time manageable. In the future, we will address various backfill depths to determine the impact to fairness versus performance.

3.2. Local Backfill Schedules

The general technique employed to achieve fairness among participating clusters is to perform backfilling at *two distinct logical levels*. First, the scheduler on each cluster performs job backfilling locally, thereby inherently giving priority to local jobs. A local backfill schedule is cre-

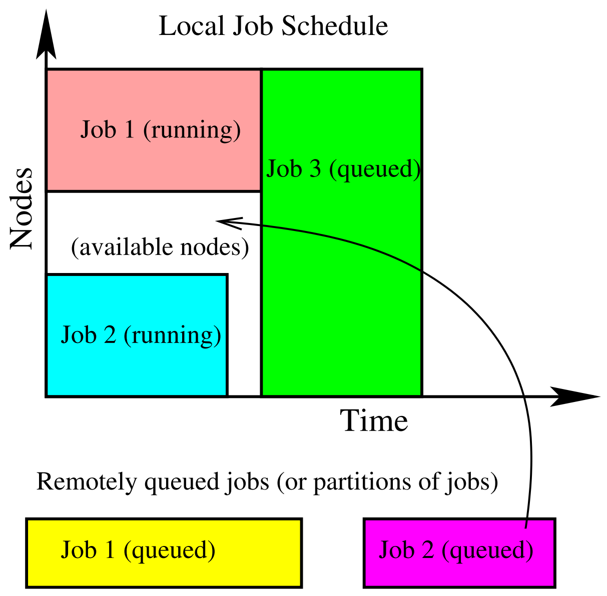

Figure 3. Backfilling

ated during this process that uniquely determines the earliest start times for a configurable number of jobs waiting in the local job queue. Precisely, if there are not presently enough free resources to start a candidate job in the local job queue, the queue is traversed from head to tail until a job is found that can start subject to both the number of available resources as well as the local reservation schedule. As each job is considered, if it is unable to start due to these constraints, a reservation for the candidate job is made in the local job schedule at the earliest start time subject to the resource usage of the jobs already in the schedule.

3.3. Global Backfilling

After each of the local clusters backfills and starts as many local jobs as possible, the remaining local jobs on each cluster have reservations in their corresponding local backfill schedules. These local schedules are then passed to the global scheduler for use during the migration and co-allocation phases of the current scheduling iteration. As the global scheduler traverses the global job queue in FCFS order, it searches for candidate jobs that will not violate any of the local reservation schedules. The net effect here is that when performing migration or co-allocation, the global scheduler does not have access to all currently available node resources on each cluster. In fact, the scheduler only has access to place a non-local job (or portion of a non-local job) on a given cluster, subject to the local reservation schedules. This means that running non-local jobs will NOT delay the earliest start times of any of the local jobs already in the local reservation schedules.

3.4. Bandwidth-aware Co-allocation

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 strategy used in this paper 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. (The interested reader may wish to explore the implementation, performance and limitations of schedulers that relax this constraint in our previous work [4].)

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 [4]. 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.

4. Experimentation

In this paper, we assume that 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. The number of nodes that a job requires is given by a uniform distribution $D_i^{nodes} \sim UNIF[10, 50]$. We also assume that the fraction of *the total execution time* that initially represents computation is set to a constant, $K_i = .7$, for all jobs. While this parameterization is generally considered to be unrealistic, it does provide an easily scalable characterization to generate a disparity among workload intensities. We are currently working on integrating a synthetic workload generator that is based on actual workload traces to improve the degree of realism.

In order to create a disparity among the workload intensities of the participating clusters, the arrival rates are adjusted accordingly. In each of these simulations, three of the four participating clusters experience an increasing arrival rate, while the fourth maintains a fixed arrival rate. By doing so, we are able to observe the effect that the first three overloaded clusters have on the fourth cluster.

In the first simulation, the size of each of the four clusters is 100 nodes. They are kept the same in order to demonstrate the effect solely due to the disparate workload intensities. In the second simulation, the clusters are sized so that cluster 1, 2, and 3 each have 100 nodes, and cluster 4 has 50 nodes. In this simulation, the smaller cluster has a fixed arrival rate while the others experience an increase in workload intensity. In the third simulation, the clusters are sized so that cluster 1 has 200 nodes, cluster 2 and 3 each have 100 nodes, and cluster 4 has 50 nodes. As with the other simulations, the fourth cluster has a fixed arrival rate, while the others do not.

In order to compare the performance of the fourth cluster while participating in the grid, an initial baseline study was performed for the fourth cluster in isolation. This baseline is labeled as “No Share on FC” (on **F**ixed **C**luster) in Figures 4 - 6. Note that it appears as a horizontal line on these graphs since it’s arrival rate is being held constant while in isolation (i.e. not participating in the grid). Note that all four clusters are initially parameterized with the same proportional arrival rate (based on their respective sizes), approximately 570 jobs/week/cluster (i.e. near the origin of the plots on the left-hand side). Then, three of the four clusters are parameterized with increasing arrival rates, while the fourth cluster’s arrival rate is held constant. This allows us to determine how well the fourth, less heavily-loaded cluster, fares in the presence of increasing pressure from its neighboring participants.

In order to demonstrate that the integration of the backfilling technique indeed improves the fairness experienced by the fourth cluster, two distinct versions of the scheduler, referred to as ICS (Integer Constraint Satisfaction) and ICS+B (ICS+**B**ackfill) are evaluated. The ICS scheduler does not employ the fairness mechanisms discussed in the paper, while ICS+B does.

The first item in each plot, labeled “ICS on G” (ICS on **G**rid), shows the average job turnaround time experience by each cluster using the “unfair” version of the ICS scheduler. Note that since the original version made no provisions to protect local jobs during the migration and co-allocation

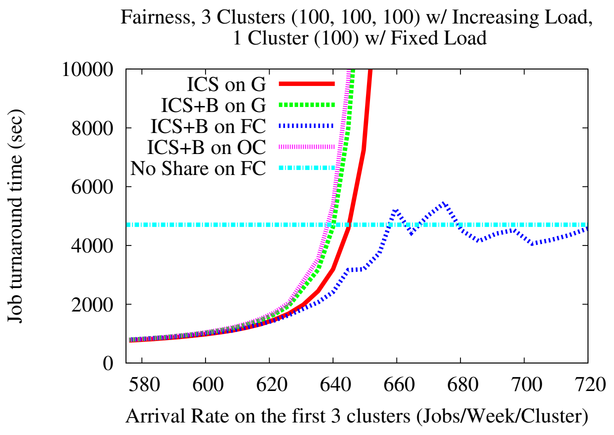

Figure 4. Effects of disparate loads with equally-sized clusters

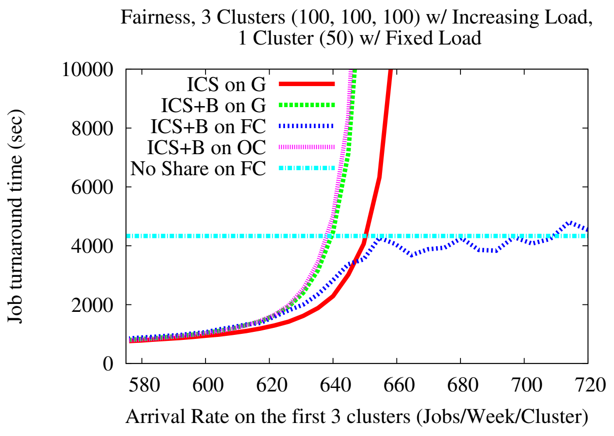

Figure 5. Effects of disparate loads with a disparity in cluster sizes participant

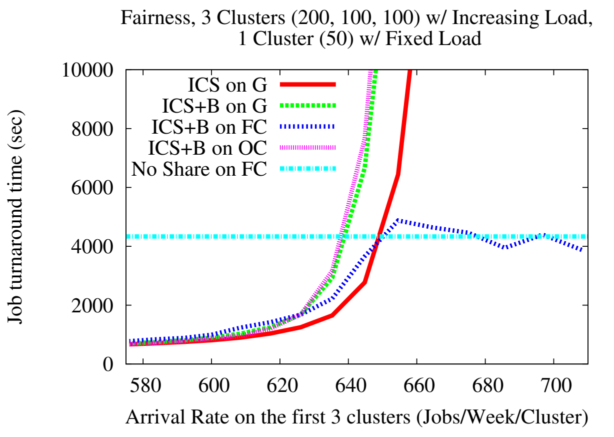

Figure 6. Effects of disparate loads with larger disparity in cluster sizes

phases of the scheduling iteration, it is typically the case that the average turnaround time on each cluster is roughly the same. This is due to the fact that when traversing the global job queue, the scheduler simply sees the grid as a single collection of resources, and maps jobs without regard to the impact that non-local node allocation will have on any locally waiting jobs. We have therefore only shown the average performance for the grid as a whole under the original ICS scheduler.

In the case of the backfilling ICS scheduler, three measures are provided. Specifically, the performance of the jobs on the fourth cluster is shown separately (ICS+B on FC i.e. on **F**ixed **C**luster); while the other three clusters performance is shown as an averaged quantity (ICS+B on OC i.e. on **O**ther **C**lusters). Additionally, the overall performance on all four clusters is shown in the figures as “ICS+B on G”.

5. Observations

The primary observation is that the backfilling strategy does a good job in keeping the fourth cluster from becoming too overwhelmed with the excess jobs from the first three clusters. As the arrival rates on the clusters 1, 2, and 3 place an enormous pressure on the system, the local backfill schedule on cluster 4 prevents its performance from degrading much beyond that of its isolation performance (No Share). In essence, even when the odds are stacked against the fourth cluster, its performance is no worse than it would be if it were not participating in the multi-cluster. On the other hand, when the relative arrival rates are more balanced across the clusters, each cluster more noticeably benefits from its participation.

The second observation is that there is a trade-off between overall performance and fairness; an expected result. With the backfill fairness policy in place, the first three heavily-loaded clusters experience a slight degradation in performance. This is due to the fact that the fourth, less heavily-loaded cluster, is preventing any additional non-local node allocation from taking place. While the trade-off *is* noticeable, it is much less prevalent when the clusters have more balanced arrival workloads. The trade-off becomes more noticeable as the disparity in workload intensities increases. In this case however, this slight decrease in overall performance comes with an enormous improvement in fairness to the less heavily-loaded cluster.

It should be noted that the non-monotonic behavior shown in Figures 4 - 6 is not due to a deficiency in sample size (as we are averaging across 400K jobs / cluster), but rather on two distinct considerations. Firstly, since our scheduler is an on-line algorithm, it does not have access related to the future arrival of jobs; therefore, when it makes a decision to backfill a non-local job, it can not prevent it from consuming resources that could otherwise be used by a local job that arrives in the near future. For this reason, the performance is sometimes worse than that of the ”No Share on Fixed Cluster”. Secondly, the arrival rate on the other three clusters is pushed extremely close to the departure rate, thus resulting in a system that nears the point of instability. This results in a relatively large variability in the performance on the other three clusters, and therefore also affects the performance of the fixed cluster. This was done however to illustrate that even when in the presence of nearly unstable adjacent clusters, a typical cluster still fares well under extreme conditions.

6. Conclusions and Future Work

We have demonstrated that conservative backfilling can be employed in tandem with co-allocating job schedulers to improve fairness among participating clusters. Additionally, we have shown that the trade-off between performance and fairness is relatively small.

While backfilling proves to be an effective mechanism to achieve fairness among participating clusters, it does have a few limitations. Most importantly, it does not prevent non-local jobs from consuming resources that would otherwise be available for local jobs that have not yet arrived to the system. Backfilling alone only provides security for jobs that are already in the queue. Therefore, in order to improve the effectiveness and generality our fairness policies, we intend to leverage more sophisticated accounting-based mechanisms in our future work.

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, 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.

[3] Job Scheduling Strategies for Parallel Processing. http://www.cs.huji.ac.il/˜feit/parsched/.

[4] 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.

[5] D. Lifka. The ANL/IBM SP Scheduling Systems. In *Proc. of the 1st Workshop on Job Scheduling Strategies for Parallel Processing*, volume 949, pages 295–303. LNCS, 1995.

[6] R. Yahyapour and U. Schwiegelshohn. Fairness in parallel job scheduling. In *Journal of Scheduling*, pages 297–320. 2000.