

Scheduling and Resource Management in Computational Mini-Grids

July 1, 2002

Project Description

The concept of grid computing is becoming a more and more important one in the high performance computing world. At Clemson University, we have built a campus-wide grid, or “mini-grid” , that is similar to a true computational grid but has a number of distinct architectural advantages. In this proposal, we seek support to investigate scheduling algorithms that will exploit these unique architectural features.

The structure of the clemson computational mini-grid is shown in figure 1. The original four clusters in the grid were provided by support from the NSF via MRI award EIA-0079734 and this year a fifth cluster was added for high performance visualization through a grant from the Keck foundation. The 5 clusters in the grid now consist of 792 processors accessed by a large number of researchers on campus, including the Center for Advanced Engineering Fibers and Films(CAEFF), the Clemson University Genomics Institute(CUGI), and the Parallel Architecture Research Lab (PARL). While the grid has been successful in producing research results for all the groups involved, this type of architecture presents new challenges in resource scheduling that must be met if the full potential of the grid is to be realized.

Results from prior NSF support

Center for Advanced Engineering Fibers and Films

EEC-9731680, 1998-2006 (\$23,400,000)

This award established an NSF Engineering Research Center (ERC) for Advanced Engineering Fibers and Films at Clemson University with the Massachusetts Institute of Technology as a partner institution. The Center provides an integrated research and education environment for the systems-oriented study of fibers and films. CAEFF develops polymeric materials suitable for fiber and film applications through experiments, mathematical models coupling molecular and continuum information, and three-dimensional images created in the virtual domain. The Center’s integrative, centralized testbed (comprising film extrusion equipment, fiber spinning equipment, on-line instrumentation, computer simulation and scientific visualization facilities, and characterization equipment) links actual fibers and films to their computational and

visual models. CAEFF's charge as an ERC includes serving as a model for systems-oriented, integrated research and education both on campus and for the larger scientific community. The Center champions the swift transfer of research results to the classroom, innovative curriculum developments, real-world applications and industry perspectives, and holistic education spanning all student populations, from kindergarten to continuing education. The Center is particularly interested in making these experiences available to student groups traditionally underrepresented in science and engineering, in order to better utilize the nation's diverse human resources.

Acquisition of a Computational Mini-Grid Supercomputing Facility

EIA-0079734, \$1,116,727, 2000-2003

This award under the MRI program funded the purchase of computational equipment that comprises a prototype mini-grid facility used for numerical computation. The Clemson computational minigrid consists of four interconnected Beowulf clusters with a total of 528 1 Ghz Pentium III processors, a quarter terabyte of memory, and nearly nine terabytes of storage. Each node in each cluster is connected by dual fast ethernet connections to the cluster switch. The cluster switches are connected together via multiple trunked gigabit ethernet connections. Custom in-house software allows nodes in clusters to be "loaned" to other clusters in the grid, allowing all the grid resources to be temporarily allocated to a single problem. The grid is used for running high performance simulations of film and fiber problems, bioinformatics applications, and research into scheduling and operating system issues in parallel computing.

Mini-grid Architecture

At first glance, the mini-grid may appear to be distinguished from a conventional computational grid only in scope. A mini-grid is limited to a campus-wide setting, while a conventional grid is national or global in scope. However, a mini-grid also has two distinctive architectural features:

1. The grids are composed entirely of Beowulf Clusters
2. The internal networks of the client clusters are interconnected through dedicated links.

This adds an additional layer of complexity in producing software for these grids. Resources may be shared through the private network links, so an understanding of some grid services must be integrated into the system software at the node level of the client clusters, rather than simply at the head of the cluster. However, this arrangement provides two primary advantages:

1. Finer granularity control of resources within the grid
2. Predictable, reliable bandwidth between grid resources (as opposed to internet connected grids).

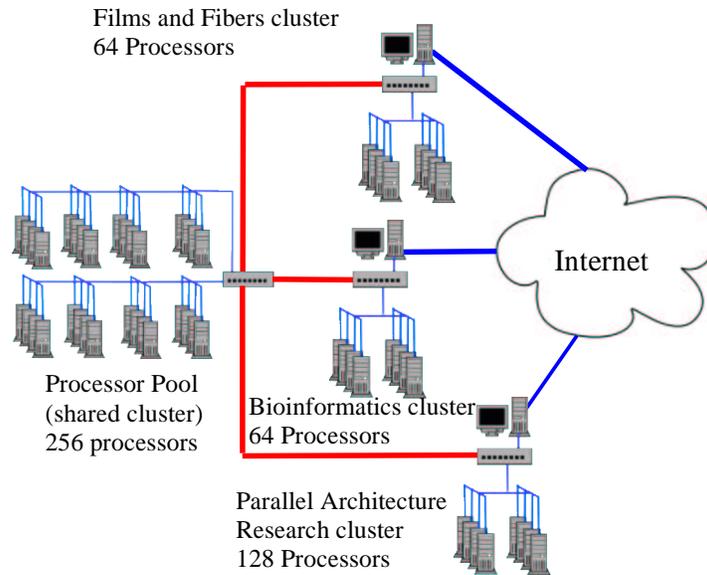


Figure 1: The Clemson Computational Mini-Grid

The additional granularity of control comes from the fact that is possible to allocate resources to a particular job on a per node basis, rather than having to allocate entire clusters or supercomputers connected to the grid to a single task. In the Clemson mini-grid, we have developed the facility to “loan” nodes between clusters, allowing for jobs to span multiple clusters. A sample allocation is shown in figure 2.

The existence of dedicated bandwidth between the clusters on the grid lowers the cost of having jobs on the grid span multiple clusters. The fact that the bandwidth between the clusters is predictable makes it possible for a scheduler to accurately account for the cost of using remote resources in making scheduling decisions.

The Grid Scheduling Problem

Before the issue of grid scheduling can be examined, the scheduling problem on a single cluster must be examined. On an average cluster or supercomputer, users typically submit work to a local queue on the head of the cluster. There may be one or more local queues, differentiated by some characteristic of the jobs submitted to them (e.g. length of job, priority of job, etc.). Typical queue implementations for clusters include OpenPBS or PBSPro[7], NQS, or LSF. The queues all run under control of a local scheduler, which may be integrated with the queuing system, or may be a separate piece of software, such as the popular Maui [8] scheduler. The queuing software then interfaces to the local dispatching method on the cluster (e.g. mpirun, bproc) to begin the execution of jobs. This situation is illustrated in figure 3. The local scheduler

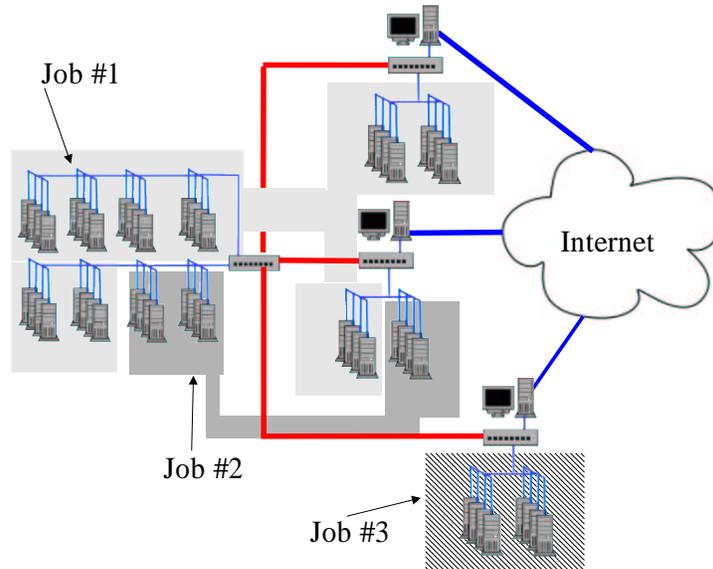


Figure 2: Typical Mini-Grid Job Allocation

typically has an algorithm or set of algorithms by which it operates, which incorporate local policies governing user priority, length of job priority, etc. To complicate matters, there is no standard interface between the queuing system and the user, the queuing system and the scheduler, or the queuing system and the job dispatching facility.

A large body of work exists which explores issues of dynamic resource allocation or load balancing in parallel computers, or more recently specifically in clusters such as [11, 1, 3]. Many of the findings of this work are implemented in excellent fashion in the Maui [8] scheduler, which will be used as one of the local schedulers in our work. While Maui does an excellent job of scheduling a queued workload on a given set of resources, multiple instances of Maui can not interact to perform meta-scheduling, the need we propose to fill. Very recently, a team of researchers at BYU have begun to extend Maui to deal with meta-scheduling issues, and begun the development of a meta-scheduler for heterogeneous resources on computational grids [10]. While this meta-scheduler addresses some of the same challenges, we feel the mini-grid environment is different enough to require a separate effort.

In a grid environment, scheduling is substantially more complicated. Each cluster in the grid has the environment described above, although each may have different local scheduling policies. Further, there is an authentication issue. Different users may or may not have accounts on each cluster, or may have different IDs on each cluster.

Much of the work on scheduling in traditional computational grids relates to the Globus project, the standard toolkit for building grid software. In particular, grid schedulers and resource allocators described in [2] and [6]. This work presents some insights

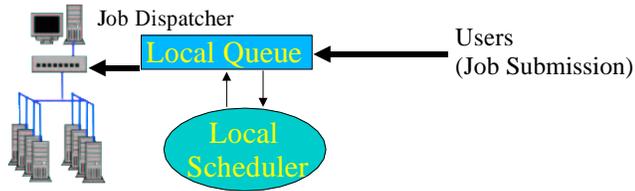


Figure 3: Scheduling on a Single Cluster

into dealing with local schedulers and policies, but not in an environment where nodes could be shared between different computers on the grid. Our work will apply specifically to a mini-grid environment, and be complementary to the techniques described.

The ability to share resources that exists in the mini-grid environment will confuse all existing schedulers. Most schedulers assume that although the workload is constantly changing, the resource pool is essentially static. The set of processors on which the scheduler operates remains the same, with the exception of occasional outages for hardware maintenance. In the mini-grid environment, as processors are borrowed from or loaned to other clusters, the set of resources with which the scheduler operates is dynamically changing as well as the work load. This situation calls for another level of scheduling to decide how the grid resources are allocated.

Research Approach

We propose to examine scheduling in a mini grid environment where node sharing is possible. The end product of this research will be a prototype scheduler for a mini-grid architecture, and data on the effectiveness of various algorithms, heuristics, and policies in this type of environment.

The scheduler we propose will not be a replacement for local scheduling on an individual cluster. Rather, we see the proposed scheduler as a higher level entity, a meta-scheduler. The meta-scheduler will coordinate the actions of local schedulers on each cluster in the grid, and work with a resource allocation system to move nodes to each clusters as the local workload dictates, with consideration to certain grid-wide performance goals. The proposed situation is shown in figure 4. A group of clusters in a mini-grid is depicted, each with it's local queue, scheduler, and resource allocation software. The meta-scheduler is shown interacting with the local schedulers and resource allocator to manage the mini-grid. We have defined several subtasks necessary to reach these goals, each of which is detailed below.

Development of Interfaces

One barrier for the creation of grid schedulers is the lack of standardization of the interfaces between the queuing system and the user for job submission, the queuing system and the scheduler, and the queuing system and the resource allocator and job dispatching facility in a Beowulf cluster. As part of this work, we plan to define simple, clean interfaces for each of the above. We intend to develop our software around these interfaces, distribute them freely to the community, and incorporate them into the Scyld Beowulf OS.

Most current implementations of resource management systems integrate queuing, resource allocation, and scheduling. The first step of our proposed research will be to decouple these functions with well defined interfaces that will allow different tools to be combined to form a complete resource management system. Once this infrastructure is complete, our meta-scheduler can interact with the local scheduler, queue manager, and allocator on the nodes to perform grid-wide scheduling.

Continued Development of Grid Node Allocator

A key underlying element in building a functional scheduler is a mechanism for resource allocation. In PARL, we have developed *balloc*, the Beowulf node allocator. Balloc provides the underlying mechanism by which nodes are exchanged between clusters. Balloc has been developed as an integral part of the Scyld Beowulf OS, a version of the Linux operating system modified for Beowulf clusters. In the Scyld OS, cluster nodes are installed with a slave version of the OS that contacts the cluster head to receive work. A balloc daemon runs on every cluster head throughout the mini-grid. The daemons maintain contact with each other to determine the availability of nodes throughout the grid. Upon receiving a request, balloc daemons can reassign slave nodes which are not currently in use to a different cluster head. Modifications have been made to the standard “mpirun” mechanism for starting parallel jobs to use balloc.

While balloc is currently functional, it requires improvements to its interface for use with a scheduler and a queuing system, and additions to maintain a sane system state in the event of job crashes, etc. Balloc also requires that a slave node be assigned to only one cluster in the grid at a time. This becomes inconvenient if the node provides some kind of specialized service in its home cluster (e.g. its disks are used by a parallel filesystem), but is otherwise available for loan to another cluster. Continuing development of balloc will allow nodes to be used by multiple clusters simultaneously.

Development of Beowulf/Grid Event Driven Simulator

In order to properly evaluate scheduling alternatives in the mini-grid environment, we will develop an event-driven simulator of Beowulf clusters and a mini-grid environment. The simulator will be based on an existing event-driven simulation package. The simulator will be configurable to support mini-grids of an arbitrary number of clusters of arbitrary size. Among the proposed features of the simulator are:

- Sequential or Parallel job execution

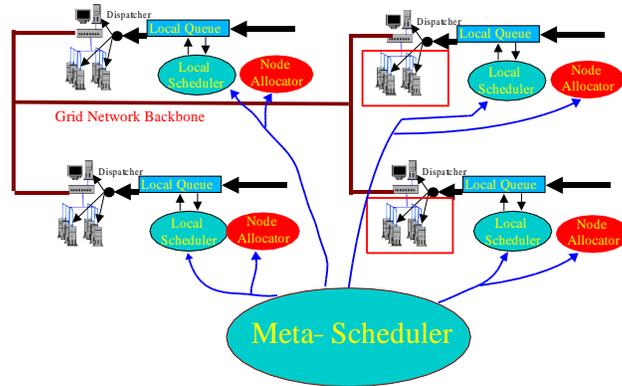


Figure 4: Proposed Role of the Metascheduler

- Multiple queues on each simulated cluster
- Local scheduling policies on/between each queue
- Sharing of nodes between clusters
- Multiple CPUs per node
- Variable interconnect speeds between nodes and between clusters
- Optional preemption of running jobs

We will use the simulator as a testbed for the evaluation of scheduling algorithms and policies. Workloads taken from our existing mini-grid will be used, as well as randomly generated workloads where a probability distribution function is used to generate job inter-arrival times, size (number of CPUs required), length of run, memory requirements, I/O requirements, and grain size (ratio of computation/communication). We intend to base our testing of the simulator on much of the existing work in characterizing parallel workloads, for instance [5] and [4].

Exploration of Scheduling Algorithms and Policies

Once the testbed is in place, the core of this work will be to evaluate various scheduling alternatives for a mini-grid environment. We will focus on the scheduling of typical message-passing parallel jobs. While we will examine the applicability of scheduling techniques developed for computational grids, for individual Beowulf clusters and other parallel computers, as well as algorithms from other domains, we believe that this application has a number of unique factors which will require development of new approaches. Among the challenges are:

- Interfacing with the local scheduler

- Complying with local priorities and policies
- Making local scheduling decisions when the available resource pool is dynamic
- Building a distributed scheduling algorithm with fault tolerance suitable for a mini-grid environment

Each of these challenges is significant. Most schedulers are designed to have complete control of resources, and to make all decisions about the order in which a workload executes. Having resources which exist outside of the control of the scheduler, or which the scheduler is only aware of part of the time, greatly complicates the situation.

Local priority is also a significant issue. In a mini-grid environment, many of the clusters that comprise the grid are “owned” by a particular research group, making scheduling a difficult issue from both an algorithmic and political standpoint. More sophisticated priority algorithms are required, in which certain classes of jobs have higher priority only on certain specific resources or certain quantities of resources. This makes the situation more complicated than simply stating that one job has a higher priority than another; a job may have a higher priority on one node but a lower priority on another. Avoiding deadlock in this situation, and preventing starvation of large jobs that must span multiple clusters in order to execute is a significant challenge.

Another substantial change from previous scheduling work is we believe that these algorithms must be truly distributed. In most situations, the scheduler is a centralized entity with complete access to system state. In a mini-grid environment, where various clusters may be unavailable due to network problems or maintenance, the meta-scheduling system should continue to function. This requires an ability for topology discovery and self-organization of the schedulers in the absence of a central authority.

We intend to develop algorithms to address all of these challenges. Though our system software currently does not support checkpointing or migration of jobs in the Clemson mini-grid, we intend to explore the issues of preemptible and migratable jobs in the development of algorithms. We anticipate that any algorithm addressing all these issues will require a number of heuristics and policy decisions, and we plan to investigate the impact of these policy decisions and the performance of the system with a range of values for the heuristics. Among the specific algorithms targeted, we will attempt to extend the backfill and fair-use algorithms employed by the Maui scheduler [8] to deal with a dynamic resource pool. The attribute tags used to define jobs in a queue will be extended to include their suitability to run on local versus borrowed processors in the grid. We hope that our study will ultimately produce substantially new algorithms in addition to adaptations of existing ones.

Implementation of Prototype Scheduler

Once the resource allocation software is in place, and we have determined a potential scheduling mechanism with the simulator, we will build a prototype meta-scheduler and employ it on our mini-grid to validate our simulation results. The meta-scheduler will interact with an existing local scheduler and queue software, such as the Maui scheduler or the scheduling mechanism built into the Portable Batch System. The meta-scheduler will also interface with the balloc resource allocation software.

Though the meta-scheduler is shown in figure 4 as logically a single entity, the implementation will be a distributed one. A piece of the meta-scheduler will exist on each of the clusters in the grid, and will self-organize into a complete meta-scheduler based on whatever clusters are currently functional within the mini-grid.

Impact

We envision mini-grids as an extremely powerful intermediate level between individual computational resources and national or global scale computational grids. Beowulf clusters are proliferating at an enormous rate. In almost any campus environment, multiple clusters typically already exist. While mini-grids are limited geographically (by the need for dedicated network links), this type of architecture could be applied at any university campus, government lab, or large industrial lab. Collectively these settings make up many if not most of the nodes in the larger computational grid. We see the mini-grid approach as complementary to the computational grid approach, where each mini-grid becomes a powerful node of the computational grid.

In order for this vision to come to fruition, mini-grids must have effective scheduling mechanisms, and well-defined interfaces to these mechanisms in order to be integrated with both computational grid and existing cluster software. We believe this effort to be a significant step toward making mini-grids a viable approach.