Grid Scheduler with Dynamic Load Balancing for Parallel CFD

Stanley Chien*, Yudong Wang, Akin Ecer, and Hasan U. Akay

Computational Fluid Dynamics Laboratory
Purdue School of Engineering and Technology
Indiana University-Purdue University Indianapolis
723 W. Michigan st., Indianapolis, IN 46202, USA

Key words: Grid Scheduling, Dynamic Load Balancing.

Introduction

Load scheduling and balancing of parallel jobs in a heterogeneous computer environment have been studied for many years. Some studies started form system's point of view that do not consider the special property of the parallel applications. They treated each parallel process as a single independent process. It is proved that these kind systems cannot provide a good scheduling for tightly coupled parallel jobs (such as parallel CFD jobs). Some other studies started from the application point of view. Most them restricted the computer systems to be homogeneous. Since it is very costly to use a large-scale heterogeneous computer system, the applications of these systems are very restricted. We studied the load-balancing problem from both the system and application point of views and developed a dynamic load-balancing tool DLB3.0 [1]. DLB 3.0 can best utilize the available computing resources for tightly coupled parallel jobs in a *heterogeneous* computers and multi-user environment. To the best of our knowledge, none of the existing computer system schedulers can do the dynamic load-balancing job that DLB 3.0 can do. Although some schedulers on the market claim that they can schedule the tightly parallel jobs, they just schedule each parallel process as if it is an independent process. We have shown analytically and experimentally that this approach utilizes the computer resources very poorly.

In order to enable more CFD users to use DLB, we need to make the DLB work together with the schedules on the computer clusters. Since the existing computer system schedulers do not provide the environment for dynamic load balancing of tightly coupled parallel jobs, we are developing a new global scheduler for Grid environment.

Requirement to the scheduler

There are three types of jobs that a computer needs to run: sequential jobs, loosely coupled parallel jobs, and tightly coupled parallel jobs. Tightly coupled parallel job means that the parallel processes of the job need frequent communication and synchronization (like a parallel CFD job). A good computer system scheduler needs to support all three types of jobs. There are three ways a computer handles jobs: dedicated mode (it reserves time periods for a user), batch mode (user runs jobs one by one in first come first serve bases) and interactive mode (multi-user mode). A good computer system scheduler needs to support all three types of jobs. There are three kinds of network environments: (1) network managed by one owner, (2) network managed by multiple owners without firewall (can logon any node directly in the network), and (3) network managed by multiple owners with firewall (can logon a node only through a gate node in the network).

The goal of this study is to develop a new scheduler with the following features:

- Efficiently schedule tightly coupled parallel jobs, single jobs, and loosely coupled parallel jobs.

*E-mail: schien@iupui.edu

- Schedule job request in dedicated mode, batched mode (queued jobs), and interactive mode.
- Support heterogeneous computers in multiple computer networks with firewalls.
- Support large parallel jobs using hundreds or thousands networked computers by providing computer system fault tolerance based on application level job migration.
- Support dynamic load balancing for tightly coupled parallel jobs.

Scheduler design

Since Globus [2] supports security in Grid environment and allows each cluster to have its own scheduler, our scheduler will be built under Globus. To allow users to use computers in multiple clusters, we need to determine if the scheduler should be centralized or distributed. Centralized scheduler has been the main stream in off-the-shelf schedulers [3-5]. Centralized scheduler is relatively easier to design and manage. However, when thousands of computers with many different owners are used, a centralized scheduler will need very high political force to implement. To make Grid computing practical, we decided to develop a distributed scheduler. In this distributed scheduler, each cluster (or an owner) runs its own scheduler and the schedulers in clusters can communicate and negotiate for shared resources. If an owner wants to share resources with other clusters, the cluster administrator needs to allow the local scheduler to communicate and negotiate with the schedulers in other clusters.

Figure 1 shows the block structure of our grid scheduler. Only two clusters are shown in the figure. All user commands are written in Globus RSL format. User's requests are sent to the scheduler of the local cluster through the Globus installed on the local cluster. Since a job should provide the user and application dependent information to DLB, we expanded RSL so that it can pass more information required by DLB through Globus. In order to make sure user's application can be executed across different clusters, user's applications should use MPI-G2 [6] for message passing.

If a job is submitted to a local cluster scheduler through Globus and the job needs computers on different remote clusters, the local scheduler will be responsible for assign the needed machines in the local cluster to the job and for starting negotiation with the schedulers on the remote schedulers. Since the remote clusters have firewalls, the negotiation between clusters will go through Globus. Currently we only allow a submitted job to request machines running on the same operating mode. If a job requests the machines that are operating in interactive mode, the job will be executed immediately based on the available resources that the schedulers can find. If a job requests the machines that are operating in batch mode, the job will be queued when the schedulers cannot meet all the conditions requested by that job. The job will be executed only after scheduler can match all requirements from that job. If a job requests the machines that are operating in dedicated mode, the job will be executed only after scheduler can match all requirements from that job. There are two types of requests for machines operating in dedicated mode. One is to request the amount of dedicated time for a number of machines. The other is to request dedicated time for a number of machines starting at a specific time. The scheduler will be able to handle both cases. When a job is running either in dedicated or batched modes or interactive mode, the load balancer in the scheduler will make sure the job uses the available resources optimally. If the requested machines on different clusters operate in different operating modes (dedicated, batched, and interactive), how to handle the request is still under study.

Another feature of this scheduler is fault tolerance. If one computer is down and a job is executed on it, the scheduler will restart the job on other available computers automatically without any human intervention.

Conclusion

A global Grid scheduler is a software tool that provides support to many users who run parallel jobs concurrently on the clusters of heterogeneous computers while maintaining the load balance from both user's point of view and system administrator's point of view. The scheduler

has a self-healing feature that can recover parallel CFD computation from system and hardware related errors. The error recovery is transparent from the user. Applications will be presented at the conference.

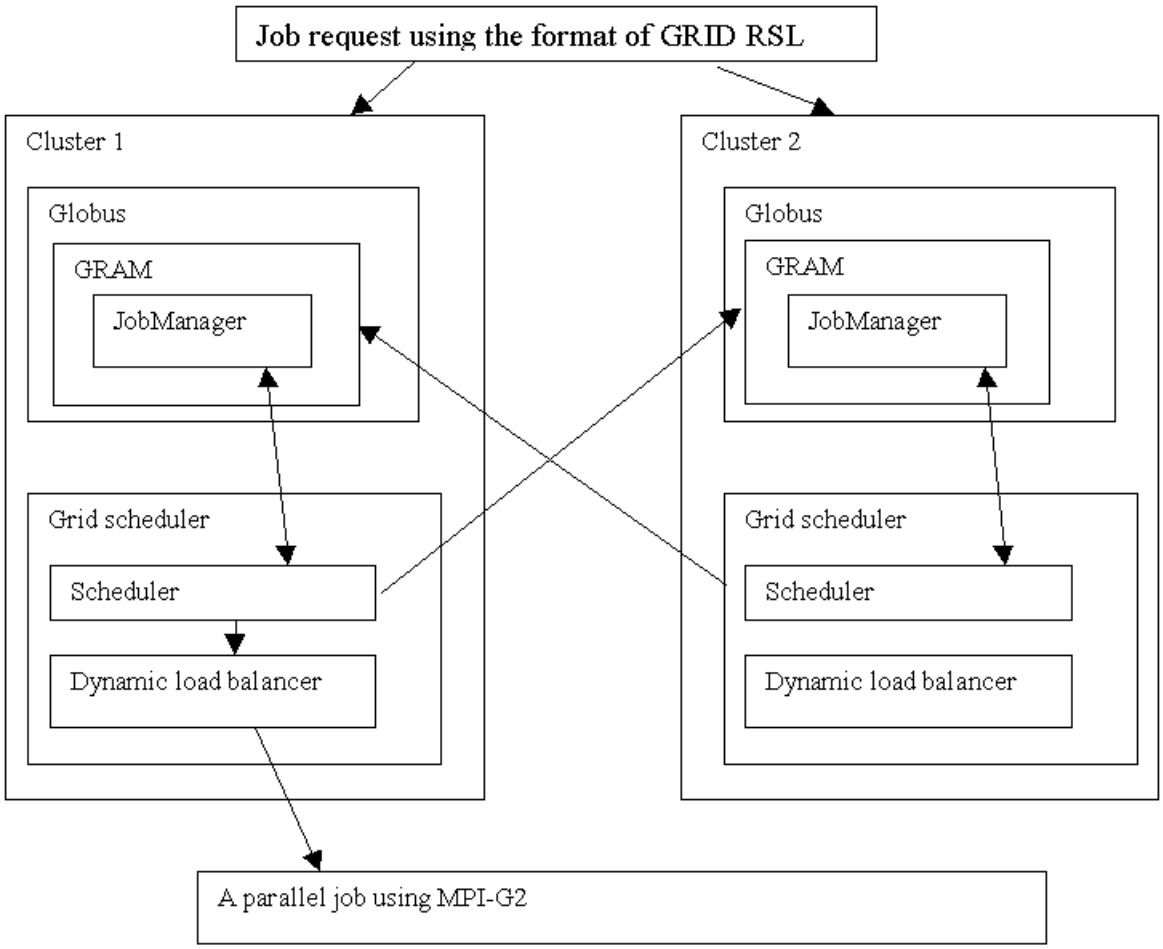

Fig. 1. Structure of the Grid scheduler.

Acknowledgement

The authors greatly appreciate the financial support provided by the NASA Glenn Research Center, under Grant No. NAG3-2260.

References

1. Chien, S., Zhou, J., Ecer, A., and Akay, H.U., "Autonomic System for Dynamic Load Balancing of Parallel CFD," *Proceedings of Parallel Computational Fluid Dynamics 2002,* May 2002, Nara, Japan.
2. The Globus project, http://www.globus.org/toolkit/
3. Condor – High Throughput Computing, http://www.cs.wisc.edu/condor/
4. An introduction to PORTABLE BATCH SYSTEM, http://hpc.sissa.it/pbs/pbs.html
5. Load Sharing Facility, http://wwwinfo.cern.ch/pdp/bis/services/lsf/
6. MPI-G2, http://www3.niu.edu/mpi/