Implementing data aware scheduling in Gfarm® using LSF™ scheduler plugin mechanism

WEI Xiaohui[1]，Wilfred W. LI[2], Osamu TATEBE[3]， XU Gaochao[1]，HU Liang[1], JU Jiubin[1]
(1: College of Computer Science and Technology，Jilin University, Changchun 130012,PRC
2: University of California, San Diego, 9500 Gilman Dr. La Jolla, CA 92093, USA
3: Grid Technology Research Center, AIST, Tsukuba, Ibaraki 3058568, Japan)

Abstract：*In high energy physics, astronomy, space exploration, genomics and other disciplines, applications that both access and generate large data sets, called data intensive jobs, increasingly draw our attention. The Data Grids, like Gfarm, seek to harness geographically distributed resources for such large-scale data-intensive problems. However, scheduling is a challenging task in this context. In this paper, we discuss the design and implementation of data aware scheduling and data management system in Gfarm. The system is able to find data-affinity hosts for Gfarm jobs and to adjust the distribution of the data replicas dynamically according to the job load. Using the LSF scheduler plugin mechanism, we do not need to write a new scheduler from scratch or make a lot of changes to an existing scheduler. Moreover, the new policy provided can cooperate with other scheduling policies in the system.*

Keywords：computing grid, data grid, data aware scheduling, LSF, Gfarm

1. Introduction

Grid is considered as the infrastructure for the next generation of Internet. Computing grid and data grid play key roles in grid technologies. Computing grid is designated to facilitate CPU-intensive jobs, whose core functionalities are job scheduling, resource management and job execution. The well known batch systems, such as Condor[1], LSF[2], SGE[3], and PBS[4], etc, focus on local job scheduling and resource management, while Condor-G[5] and CSF[6] work at the grid level.

With the fast developing computer commodity technology, CPU is no longer expensive. Emerging classes of data-intensive applications that both access and generate large data sets are drawing much more attention. High-performance data-intensive computing and networking technology has become a vital part of large-scale scientific research projects in areas such as high energy physics, astronomy, space exploration, and human genome projects. One example is the Large Hadron Collider (LHC) [7] project at CERN. The so-called Data Grids provide essential infrastructure for such applications. Grid Datafarm (Gfarm)[8], for example, is one of them.

Gfarm architecture is designed for global petascale data-intensive computing. It provides a global parallel file system with online petascale storage, scalable I/O bandwidth, and scalable parallel processing, and it can exploit local I/O in a grid of

clusters with tens of thousands of nodes. Gfarm parallel I/O APIs and commands provide a single file system image and manipulate file system metadata consistently.

If a huge amount of data I/O is involved, a network system's performance will be degraded by network congestions without proper data management and job scheduling. In Gfarm, *gfrun* and *gfmpirun* commands are able to allocate the file-affinity hosts for optimum execution of applications based on available metadata. However, the manual method is not scalable in a production environment with a large number of users running jobs concurrently. It is imperative to have an automated job scheduling and data management mechanism.

In this paper, we describe the design and implementation of data aware scheduling in Gfarm by using a LSF scheduler plugin mechanism. It is able to reserve the best hosts for job execution, and performs data stage-in and stage-out. Moreover, it can adjust the distributions of the data replicas based on the actual requirements of jobs, and balance the load for each data replica dynamically. With the plugin approach, the new scheduling policy is provided as a module to be dynamically loaded, and it can cooperate with other scheduling polices in the system as well.

In the following sections, the LSF's scheduler plugin mechanism is introduced briefly, then we discuss the architecture of the data aware schedule module. In section 4, the design principles of the scheduling algorithm and its implementation are explained. In section 5, some related works are discussed. Finally we present our plan for the near future.

2. LSF Scheduler Plugin Mechanism

In the real world, each user has a different requirement. No matter how many scheduling polices are provided, no resource management system can meet all users' needs. Hence, in version 5.0, LSF designed the scheduler plugin feature to allow users to write their own scheduling policies.

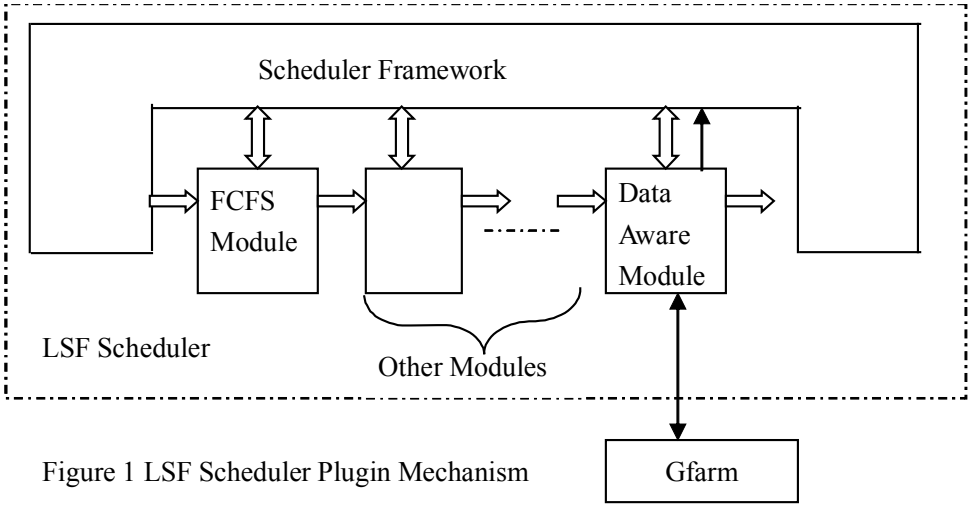

Figure 1 LSF Scheduler Plugin Mechanism

The LSF plugin mechanism consists of the scheduler framework and a number of scheduler plugin modules. See figure 1. The scheduler framework works as a motherboard with slots to hold scheduler plugin modules. The framework maintains

the elementary information, like pending jobs, available hosts etc, for all plugin modules. Plugin modules are able to access those data inside scheduler framework via a LSF scheduler API. The particular scheduling policies are implemented inside plugin modules. FCFS module, for example, provides the first come first serve policy, which is the default policy of LSF.

The plugin modules are loaded dynamically by the framework at run time. The user can indicate which modules to be loaded via configuration. Using this mechanism, the users do not need to write a customized scheduler from scratch, but to provide just a plugin module with the desired policy. In this paper, we describe a plugin module to perform data aware scheduling and data management in Gfarm.

3. Architecture of Data aware Scheduling Module

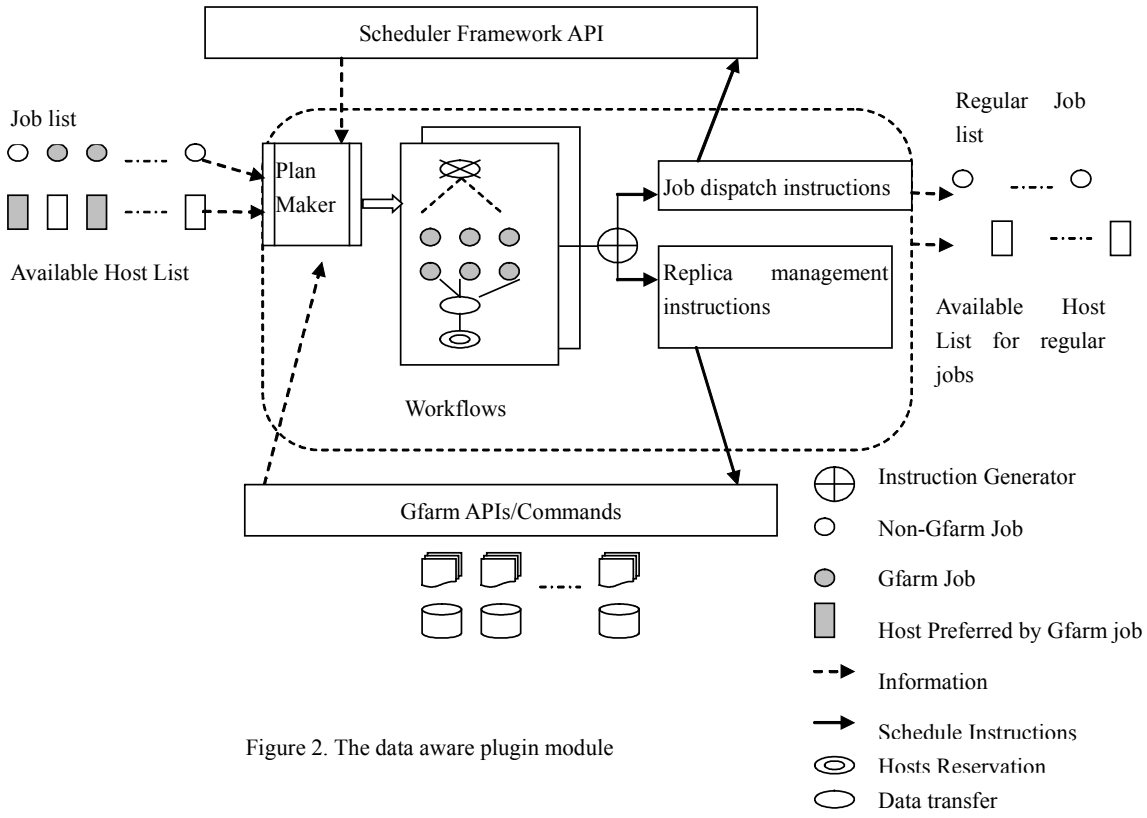

Figure 2. The data aware plugin module

The data aware scheduling module is implemented as a LSF plugin. It communicates with LSF via the LSF scheduler framework API. The module takes the pending jobs and the available hosts in the system as input from LSF and gets the data replica information from Gfarm. To Gfarm, the module is simply a normal Gfarm application. Any Gfarm API and command can be used. The output of the module is a series of schedule decisions, such as host reservation, replica creation, job execution and so on. These decisions are executed by LSF and Gfarm respectively.

The data aware module consists of three components, the Plan Maker, the Workflow Container and the Instruction Generator. See figure 2. The Plan Maker is

responsible for making job scheduling and data management decisions. The decisions are described as workflows and maintained by the Workflow Container. At the end of a scheduling session, the Instruction Generator issues the corresponding LSF or Gfarm commands to execute the workflows.

The data aware module handles Gfarm applications only, and the other jobs are scheduled by other modules. Hence, the Plan Maker has to be able to recognize Gfarm jobs. In our work, a special tag is attached to each Gfarm job at submission time. A tag is a string with the format of "Gfarm Files=*f1,..fn*", which indicates the files to be accessed by the job. This is done via LSF *bsub –ext* command.

The Plan Maker is the brain. It decides when and where to start a Gfarm job, and whether to create a new replica for a data set. The Plan Maker follows the data aware scheduling policy and writes its decisions into the workflows maintained by the Workflow Container. A workflow is a job execution plan for a data replica. Besides the jobs to be launched, a typical workflow consists of host reservation, file transfer (stage-in), and file elimination (stage-out) operations as well. New jobs can be dynamically added into an existing workflow.

The Instruction Generator is the decision executor, generating concrete jobs or data operations based on the workflows. Those instructions are issued to LSF/ Gfarm via appropriate APIs/commands at the end of each scheduling session. More details are provided in the following section.

4. The Data-aware Scheduling Algorithm

In a production cluster, there could be up to thousands of hosts, and tens of thousands of jobs running on it. Our goal is to improve the throughput of the whole system instead of a single job's execution. For data-intensive jobs, the following scenarios will cause a degradation of system performance. First, there are many jobs accessing data files through network. This leads to network congestion and slow down of the execution of every job. Second, the loads of data replicas are imbalanced. Some data files with few replicas are being accessed heavily, while some other data files with more replicas are not frequently accessed.

To alleviate the first problem, jobs should be dispatched to hosts with the required data locally or to those close to the data. This type of scheduling is called data aware or data affinity scheduling. For the same reason, Gfarm insists on users making good use of local disks, although network parallel I/O is also supported. To resolve the second problem, replica management should be able to dynamically balance the load of each data replica based on a job's actual requirement.

Our algorithm resolves these two problems above in the following ways: 1. Besides satisfying a job's resource requirements, like host/OS type, it always selects the data affinity hosts for job execution, 2. It supports data stage-in and stage-out, 3. It adjusts dynamically the number of data replicas and their locations according to the real load of the jobs in the system, 4. It avoids creating multiple data replicas concurrently to reduce the possibilities of the network access conflicts.

Both job scheduling and data replica management strategies are embodied in the algorithm. The algorithm is executed by the Plan Maker periodically. Each execution is called a scheduling session. In each session, more jobs will be inserted into existing workflows, and some new workflows could be created. All the jobs belong to the same workflow have to be started in sequence, and normally they do not start within one schedule session. The jobs from different workflows can run concurrently. See Figure 3.

Algorithm

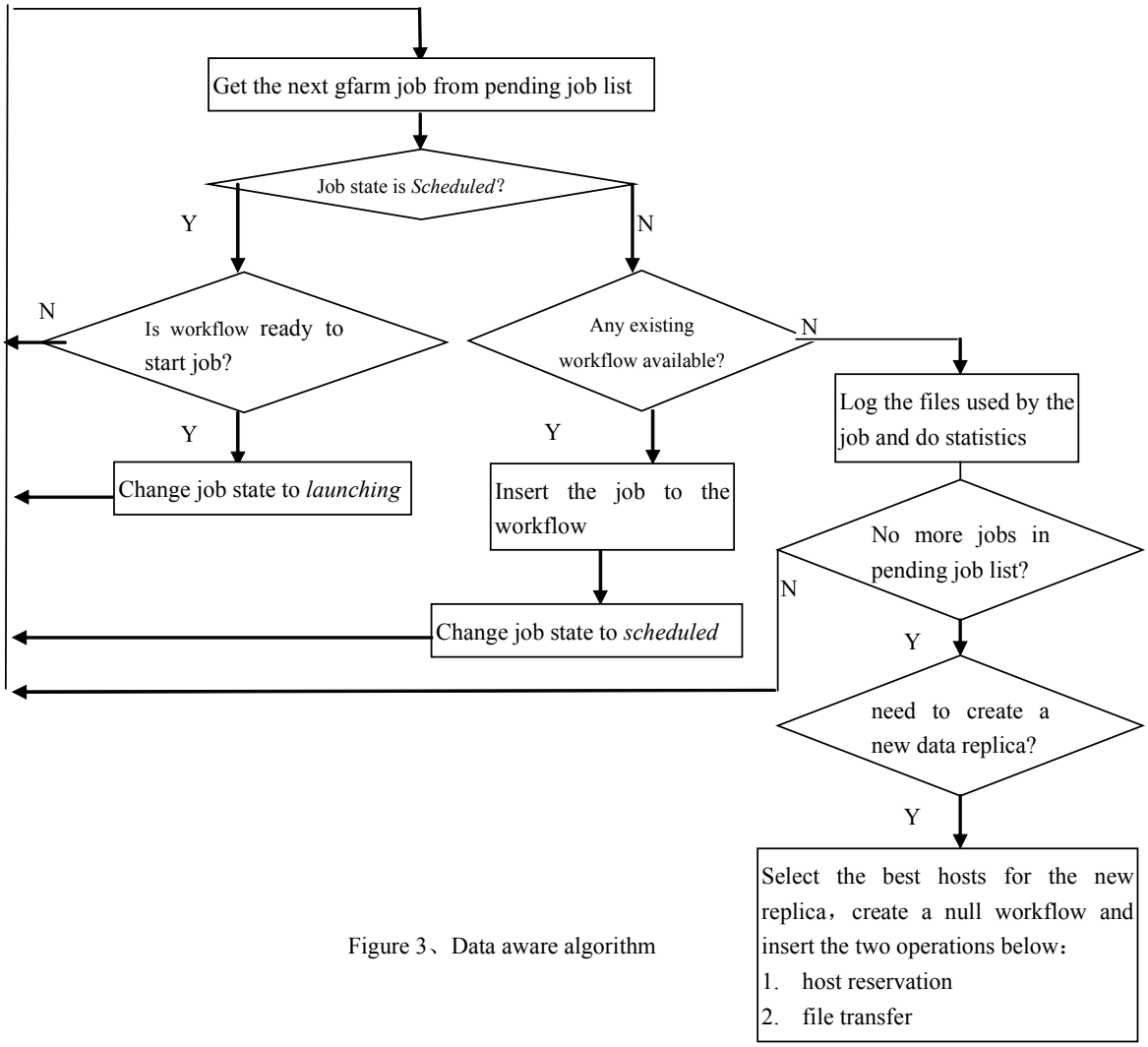

Figure 3、Data aware algorithm

1. Select a Gfarm job from the pending job list. Other jobs are ignored by the algorithm. The pending job list includes all the jobs waiting to run, and it is maintained by LSF scheduler framework.
2. If the job state is *scheduled*, it must belong to an existing workflow. Check the workflow, if a reserved host is available, and there is no job in the workflow to be executed before it, change the job's state to *launching*. The job will be started in this schedule session.

3. If the job state is not *scheduled*, then try to insert it to an existing workflow. If successful, change the job state to *scheduled*. Otherwise, log the data files (also called data set) used by the job and do statistic using the following formula,

$$Sum(ds) = \sum_{i=1..n} Pi * Ti$$

(Pi is the job's priority, ds is the data sets used by the job, Ti is the job's running time）

Note: There is a limit for the number of jobs that a workflow can have. If a workflow is full, no more jobs can be inserted into it unless some jobs are launched. Such a limit can avoid load imbalance among data replicas.

4. If there are more pending jobs, go to 1.
5. After going through all the pending jobs, the algorithm will decide whether to create new replicas in this session according to the statistic results in 3. If there is a data set whose Sum(ds) is larger than a pre-defined value, then the data set needs a new replica. However, in order to decrease the chance of network congestion, only one new replica will be created in a schedule session. The following actions will be taken to perform the replica creation for the data set with the largest Sum(ds) value,
   a. Select the best location (hosts) for the new replica according to job's resource requirement. If there is no host with spare disk space, the obsolete data replicas will be overwritten based on the LRU algorithm.
   b. Create a null workflow
   c. Insert host reservation operation to the workflow to reserve the selected hosts
   d. Insert the data transfer operation to the workflow to create the replica on the selected hosts

At the end of a scheduler session, the Instruction Generator goes through all the workflows: notify LSF to start all the jobs with *launching* state; notify LSF to execute the host reservations for the new replicas; notify Gfarm to start copying data to reserved hosts. Subsequently, the new workflow is used by the Plan Maker to schedule jobs during the next session.

5. Related Works

In [9], job scheduling and data management modules are separate components. In [10], a data aware scheduling mechanism is implemented in a peer-to-peer computing model: job scheduling and data management are implemented as two loose-coupled components. In our work, job scheduling and data management are integrated together. In our case, the scheduler knows both jobs and data well, therefore, is able to make a better plan to improve the system's performance overall.

Plan based scheduling is introduced in some recent works. The focus of [11] is on the scheduling for a single complex task with multiple components in the computing grid. AI plan is used to generate workflows to execute components of a task. Our work is an extension of [11]. Instead of focusing on a single job, workflows

are used to plan the execution of a number of data-intensive jobs. Many works, like [12], introduced methods to balance the load for a parallel job to overcome the performance heterogeneity betweens the nodes in a cluster. In this paper, we achieve dynamic load balancing for data replicas based on the needs of jobs in the system. Moreover, our use of a LSF scheduler plugin mechanism has the following advantages: no need to write a scheduler from scratch, because the data aware scheduling policy is implemented as a plugin module; the new policy can co-operate with other policies, like FCFS, fair share, and preemption etc.

6. Conclusion and Future Work

This paper describes the integration of LSF6.1 and Gfarm1.0.3 on Linux RedHat9.0 using a LSF scheduler plugin mechanism, and the implementation of data aware scheduling and data replica management functionalities. The following features are implemented: 1. A queuing mechanism for Gfarm jobs is provided and to the ability to allocate data-affinity hosts for job execution. 2. The stage-in and stage-out functionality. 3. Dynamic adjustment of the distribution of data replicas according to actual job requirement. 4. Cooperation with other scheduling policies and ease of extension with new scheduling features.

Currently, our team is also working on porting CSF[6] to GT4. CSF is a grid level scheduling framework. In the near future, we are going to introduce data aware scheduling policy into CSF. Our research will identify the different focuses between the grid level data aware scheduling and cluster level data aware scheduling, and how to make the two kind of policies work together efficiently.

7. Acknowledgment

The work of the paper is supported by Jilin University grant 419070200053 and 420010302338, and China NSF grant 60473487.

W. W. Li would like to acknowledge the support from NSF grant INT-0314015 for PRAGMA and NIH/NCRR program grant P41 RR08605 for NBCR.

Thanks Platform to provide LSF 6.1 test bed.

References

[1] Jim Basney and Miron Livny, "Managing Network Resources in Condor"[C]. *Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9)*, Pittsburgh, Pennsylvania, August 2000, pp 298-299.

[2] Songnian Zhou, Xiaohu Zheng, Jingwen Wang *et al*. Utopia: a Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems[J]. SOFTWARE—PRACTICE AND EXPERIENCE, Dec 1993: 23(12), 1305–1336 .

[3] James, P. J, Portable Batch System: Exterernal Reference Specification Altair PBS Pro 5.3[M]. http://www.mta.ca/torch/pdf/pbspro54/pbsproers.pdf, March 2003.

[4] Sun Microsystems, Inc. Sun Grid Engine 5.3 Administration and User's Guide[M]. http://gridengine.sunsource.net/project/gridengine-download/SGE53AdminUserDoc.pdf, April, 2002.

[5] James Frey, Todd Tannenbaum, and Ian Foster *et al*, "Condor-G: A Computation Management Agent for Multi-Institutional Grids" [J], Journal of Cluster Computing volume 5, pages 237-246, 2002.

[6] Platform Computing Co. Open source metascheduling for Virtual Organizations with the Community Scheduler Framework (CSF)[WP]. http://www.cs.virginia.edu/~grimshaw/CS851-2004/Platform/CSF_architecture.pdf , 2004.

[7] MONARC Collaboration. Models of Networked Analysis at Regional Centres for LHC experiments: Phase 2 report. Technical Report CERN/LCB-001[TR], CERN. http://www.cern.ch/MONARC/, 2000.

[8] Osamu Tatebe, Youhei Morita, Satoshi Matsuoka *et al*. Grid Datafarm Architecture for Petascale Data Intensive Computing [C]. Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pp.102-110, 2002.

[9] Kavitha Ranganathan, Ian Foster. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications [C]. Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, http://www.globus.org/research/papers/decouple.pdf, July 2002.

[10] Florian Schintke, Thorsten Schutt, Alexander. A Framework for Self-Optimizing Grids Using P2P Components [C]. Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA'03), http://www.zib.de/reinefeld/Publications/dexa03.pdf , 2003.

[11] Jim Blythe, Ewa Deelman, Yolanda Gil *et al.* The Role of Planning in Grid Computing [C]. 13th International Conference on Automated Planning and Scheduling (ICAPS), Trento, Italy, http://www.isi.edu/~gil/papers/icaps03-submission.pdf , June 2003.

[12] Yoshiaki Sakae, et al. Preliminary Evaluation of Dynamic Load Balancing Using Loop Re-partitioning on Omni/SCASH [C]. The 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, May 2003: 463-471.