A Job Scheduling Simulator in Data Grid Based on GridSim

XING Chang-ming，LIU Fang-ai，CHEN Kun
Information Science and Technology College, Shandong Normal University, Jinan 250014, China
xingchm@tom.com，liufangai@yahoo.com.cn, chenkunmy@126.com

Abstract

The job scheduling strategy influences the QoS of data grid immediately. This paper firstly summarizes and defines data grid models and the process of job scheduling, simultaneously analyzes the time and cost of job execution in the data grid, then proposes a design proposal of the job scheduling simulator of data grid based on a grid simulator named GridSim, and introduces the architecture, process and key technologies of the job scheduling simulator. Finally, it proves that this job scheduling simulator can satisfy the need of research on the data grid optimization theories.

1. Introduction

Along with the rapid development of grid technology, the data grid has being gained wide attention as an important branch of grid computing [1]. The data-intensity job is a main type of jobs in the data grid, and the job scheduling strategy is one of key technologies affecting execution efficiency of data-intensity jobs. Recently, new job scheduling strategies was proposed continuously. Before employment of a practical job scheduling strategy, it needs enough experiments to verify besides strict theoretical analysis. As is known, there are massive funds needed to build an actual grid environment. And it is always hard to configure and realize; even some new thoughts and methods are unable to test under the actual network environment. Thus, the simulator becomes an effective tool to evaluate performance of related strategies. Typical grid simulators mainly have: GridSim[2], OptorSim[3], Monarc[4], SimGrid[5], and MicroGrid[6]. Paper [7] carried on a detailed analysis and comparison of these simulators; and indicated that all of them had some deficiency, which reduced the flexibility of applications. Specially, these simulators cannot carry on the simulation of the data grid job scheduling directly. However, these simulators and related theories provide a good foundation platform, which can be improved to provide more plenteous functions. Considering from the concrete applications, GridSim has a variety of applications in the field of grid researching, whose developing team is leaded by Rajkumar Buyya from Melbourne University of Australia. It has both a rich function library as the most major characteristic and an open as well as expandable network architecture which is easy to add own algorithm by users to carry on simulation experiments. Therefore, this paper chooses the GridSim simulator as the basis of development. In view of the data grid job scheduling problem, it studies the design proposal and realization methods of the data grid job scheduling simulator.

The edition before 3.3 of GridSim provided rich simulation functions in point of the computation grid; after 4.0 edition, GridSim released its data grid simulation function, which expansion methods was introduced detailed by paper [7]. However, the expansion edition of GridSim only provided simulations of replica creation, replica location, replica access as well as complex query of file attributes. So far, the newest edition GridSim4.2 had still not provided realization of data grid job scheduling module, which is to say, the user cannot carry on the simulation of the data grid job scheduling strategy directly. In view of this question, this paper extended GridSim, mainly finished the following work: carrying on the detailed analysis of the data grid job scheduling process, proposing a design scheme of the data grid job scheduling simulator based on GridSim4.0, giving its realization methods of essential parts, and verifying that it can provide supports for research of the data grid optimization theory using experiment simulations.

2. The job scheduling models in data grid

In the data grid research field, US and Europe have conducted more thorough research, and promoted some experiment systems, the most famous projects of

978-1-4244-3930-0/09/$25.00 ©2009 IEEE

which are the European Data Grid Project and the American Grid3 Project. This part we analyze these systems as the basement, combine with related research of paper [8], carries on the induction definition of the data grid model and the job scheduling flow.

2.1. The data grid model

The data grid is a data-intensity job computing environment constituted of M computing resources $R = \{r_1, r_2, r_M\}$ and P data hosts $D = \{d_1, d_2, d_3\}$ [8]. In the data grid, the computing resource is usually the high performance computing platform, for example, cluster systems. The data host is used to store each kind of data files, which may be both the special-purpose storage device and the storage device contained in the computing resource. But even the storage device contained in the computing resource can also be regarded as a data host independently. Computing resource R and data host D as well as physical link connecting them constitute the entire data grid environment.

2.2. Process of job scheduling

The data grid mainly processes the data-intensity job, most of which belong to type of Bag-of-Jobs (BoT) [9]. Jobs of BoT exist generally in the high energy physics, meteorological and the biological computing. The BOT job can be decomposed of some independent sub-job, and each sub-job's execution usually needs to read multiple data files. Recording the job as J, because a job can be decomposed of some independent sub-job, J can be decomposed like this, $J = \{j_1, j_2, , j_N\}$,N>>M. Generally, executing job j need a computing resource (recorded as Rj) and K data files (recorded as F^j), and K data files distribute in data host D. Regarding $f \in F^j$, $D_f \subseteq D$, and D_f are all hosts of storing file f.

In the data grid environment, the user firstly submits its jobs to the data grid job scheduler (Job Scheduler). Then the job scheduler select resources for jobs and carry on the job scheduling according to the scheduling algorithm. Paper [10] summarizes the main job schedulers, including Nimrod/G, AppLes, Condor-G and so on, so this article no longer introduces. After the user submits job j to the job scheduler, the job scheduler j selects resources recorded as $S_j = \{R_j, D_j\}$, in which, $r \in R$ is the computing resource for executing job j and $D^j \subseteq \bigcup_{f \in F^j} D_f$ is data hosts to provide data files for the job j.

After the user submits jobs to the job scheduler, job scheduling flow is as follows: (1)querying GIS, collecting information of computing resource, including speed, network bandwidth, resource load and so on; (2) querying RC, collecting storage information of files, including replica position, replica number, size and so on; (3) describing parameters of service quality according to information and users, deciding time and location of job on the basis of job scheduling strategy; (4) submitting jobs to the computing resource; (5) the computing resource requesting files needed to the data hosts; (6) data host transmitting files for the computing resource; (7) all resources ready, the job executing; (8) job execution finished, returning results to the job scheduler.

2.3. Analysis of job execution time

The job execution time is an important performance indicator of the job scheduling strategy. After job j is submitted to computing resource r, the execution time of job j is mainly concerned with load status, processing speed as well as the network condition of computing resource r. We record the execution time of job j as $T(j, r)$.

$T(j, r) = T_w(j, r) + T_t(F^J) + T_e(j, r)$, in which, $Tw(j, r)$ is job j's waiting time in the job queue of r; $T_t(F^j)$ is the transmission time of F^j from d_j to r; $T_e(j, r)$ is computing time of job j executing in r. Paper [8] proposed two kind of job execution model shown in Figure 1, in which T_{f_1}, T_{f_2}, T_{f_n} are transmission time of file f1, f2, fn.

In model (a), job execution is independent with data transmission, which is, data communication and job computing serializing. Only when the data needed are all ready, the job executes. Thus, $T_t(F^j) = \max_{f \in F^j} (T_t(f, d^f, r)$, the file transmission time decided by the slowest transmission of F^j d. In model (b), the job execution and the partial files' transmission overlaps, which is, the data communication and the job computing parallelization. It is one kind of more actual job execution model, but the job execution time is also influenced by the data transmission.

For ease of realization, we firstly use model (a) to design data grid job scheduling simulator followed, and next step we research model. Model (a) is a special form of model (b). This model maximizes the data communication influence of the job execution time, which can reflect performance of data choice strategy in the job scheduling process.

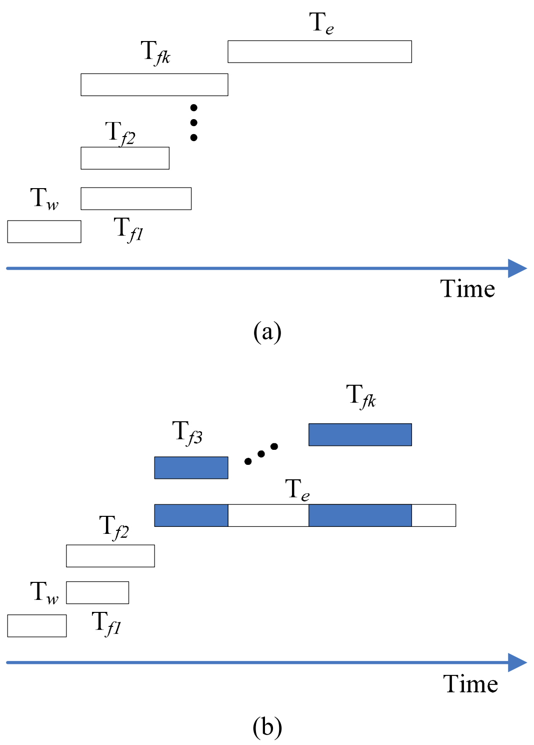


Figure 1. Model of Job Execution

2.4. Analysis of job execution cost

Under the distributed environment, the economic models are generally introduced to motivate users sharing resources, which makes job execution cost become another indicator of the job scheduling strategy. In data grid, when job j submits to the computing resource r, the job execution cost is recorded as C (j, r), which can be composed of two parts: the job execution cost of data host $\mathrm{C}(\mathrm{F}^{\mathrm{j}},\mathrm{r})$ and the job execution cost of computing resource $\mathrm{C}_{\mathrm{e}}(\mathrm{j},\mathrm{r})$.

And $C(F^j,r)=\sum_{f\in F^j} access\_\cos t(f,df)$, indicating cost of computing resource r requesting Fj to execute job j; $\mathrm{C}_{\mathrm{e}}(\mathrm{j},\mathrm{r})=I_j * C_r$, indicating instruction numbers of executing job j * each instruction cost of computing resource r. We neglects the cost of file transmission, because in the practical applications, the user downloading a file only need to pay the corresponding cost to the file provider, but not the cost of file transmission .

3. Main framework of simulator

Designing a practical network simulation platform, principles such as accuracy, open and usability should be followed. Therefore, for the ease of employment, followed the above principles, this paper designs and realizes a general, simple, open data grid job scheduling simulator. This simulator can satisfy the diverse request of users. As long as compiling job scheduling algorithms according to the standard interface, setting the simulation parameters and the output results, this simulator can load these job scheduling algorithms dynamically to carry on the experiments, and return direct-viewing output result to the user.

The data grid job scheduling simulator based on GridSim can be divided into five functional modules: Grid resources creation module, network topology creation module, grid user creation module, job scheduling module as well as statistical result output module. Various modules' flow is shown in Figure 2.

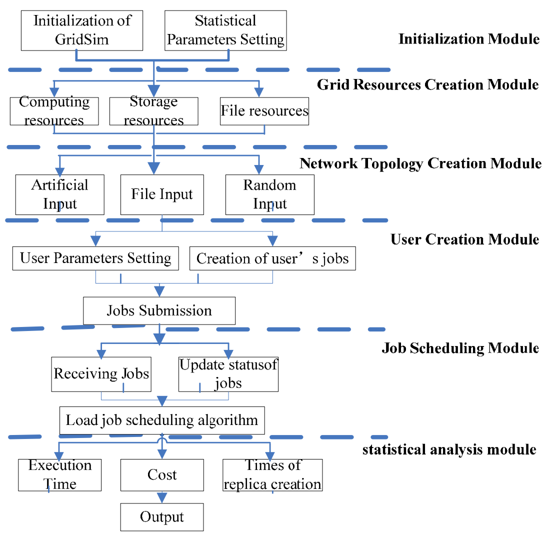


Figure 2. Modules of the System

The initialization module is responsible for the initialization operation of entire simulation, including initializating GridSim package and setting system parameters. Grid resources creation module creates resources for the simulation of grid job scheduling algorithms, including computing resource, storage resources as well as file resources. The network topology creation module is responsible to create the network topology. The topology can be edited by the user artificially, or be loaded automatically by files and randomized algorithms. The grid user creation module is responsible to create grid users, create a certain amount of jobs for each grid user, and simultaneously describe parameters of job execution. The job scheduling module firstly receives jobs submitted by users, update status of grid resources and user jobs, then execute assigned job scheduling algorithm to carry on the simulation. The statistical analysis module records the performance indicators of the job scheduling algorithm, and outputs them in the form of text or files to analyze and evaluate.

Figure 2 also gives the entire process of data grid job scheduling simulation: After starting GridSim, firstly carries on initialization of GridSim packages and configure system parameters including user quantity, resource quantity, statistical parameters and so on; then creates resources, network environment, users; after user submitting jobs to the job scheduling module, loads job scheduling algorithm to carry on the simulation; finally analyzes the simulation result by the statistical analysis module and returns the result to the user.

4. Key components of system

Grid simulator GridSim is based on the discrete event package SimJava2, defines multiple network entities, including the user, the resource, the job, the replica supervisor, the replica catalog and so on. These entities in GridSim communicate through event transmission mechanism. In order to realize simulation of the data grid job scheduling, some related components should be defined on the basis of existing entities. Below the paper introduce them separately.

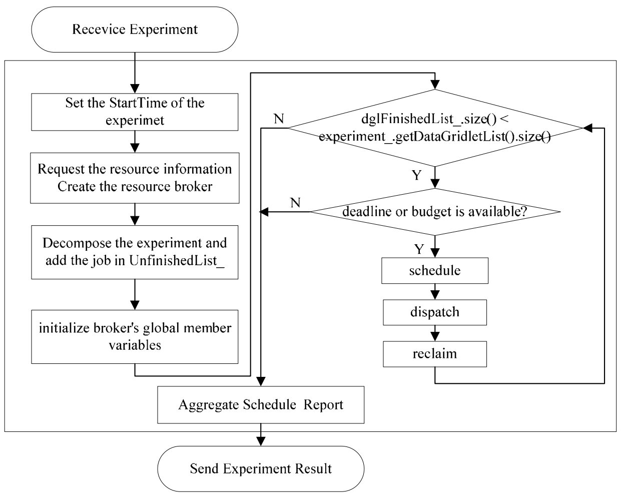
4.1. The user broker

In the actual grid environment, the user broker is equal to the grid job scheduling. The user broker is used to receive jobs submitted by users, execute the job scheduling strategy to submit jobs to corresponding resources. Meanwhile it is responsible to log job status and execution. In this simulator, a class named DataGridBroker is defined as the user agent entity. In this class, there are following methods:

- schedule() load scheduling algorithm, carry on the job scheduling;
- dispatch() according to the scheduling result, distribute jobs to corresponding resources;
- reclaim() recycle executed jobs, update resources status;
- body() implements all related methods, its flow is shown in Figure 5.

Moreover, this class defines the following several variables, the explanation are as follows:

- resIDList_: Record resources entity ID list;
- brokerDataGridResourceList_: Record resource broker list;
- dglUnfinishedList_: Record unfinished work;
- dglFinishedList_: Record finished work;
- dataGridletDispatched_: Record submitted work;
- dataGridletReturned_: Record returned work;
- expenses_: Record current cost of the job.



Recevice Experiment
Set the StartTime of the experimet
Request the resource information Create the resource broker
Decompose the experiment and add the job in UnfinishedList_
initialize broker's global member variables
N
dglFinishedList_.size() < experiment_.getDataGridletList().size()
Y
N
deadline or budget is available?
Y
schedule
dispatch
reclaim
Aggregate Schedule Report
Send Experiment Result

Figure 3. Flowchart of the User Broker Entity

4.2. The resource broker

The resource broker records resource attributes and the status information, predicts cost of job executing in this resource according to its load information, which provides the reference for the resource scheduling. In this simulator, a class named BrokeDataGridResource is defined to realize related functions of the resource broker. To convenient, this class extends the class BrokeResource in the gridbroke, and simultaneously defines two methods of getExpectedFileTime() and the getExpectedFileCost(). It realizes prediction of the transmission time and cost of data-intensity jobs. Because the length is limited, the process to realize the resources agent is no long to be introduced detailed in this paper.

4.3. Status of jobs and data statistics

Through analysis above, the execution of data-intensity jobs can be regarded as two parts of data transmission and job computing. Accordingly, this paper defines the data grid job status as the five statuses of created, blocked, ready, executing and success. Created: status after the job creates; Blocked: the job has submitted but its data needed have not yet been prepared; Ready: the data needed have already be prepared, but the computing resource has not yet gained; Executing: status of executing the job; Success: status after the job executed successfully.

This simulator mainly statistics the job execution time, the execution cost, and the resource distribution result. Execution time includes job submission time, beginning time of data transmission end time of data transmission, beginning time of job computing, end time of job computing as well as actual computing time. Execution cost includes file transmission cost and job computing cost. Resource distribution includes the computing resource distributed to the job as well as the file request.

5. Implement of job scheduling

Based on the grid entity defined above, simulating process of data grid job scheduling is shown in Figure 4. Firstly it sets experiment parameters, initializes GridSim package, then employs the user entity to create jobs; after created job, submit jobs to the user broker in the form of event transmission, then carries out the operations of job scheduling, distributing and recycling in turn.

This simulator employs a method of message inquiry to realize the job scheduling simulation. After the user broker receives a new job, it inquires each known computing resource broker about time and cost if the job executing in this resource. After all resource brokers return their time and cost information, the user broker carries out the job scheduling strategy to distribute jobs.

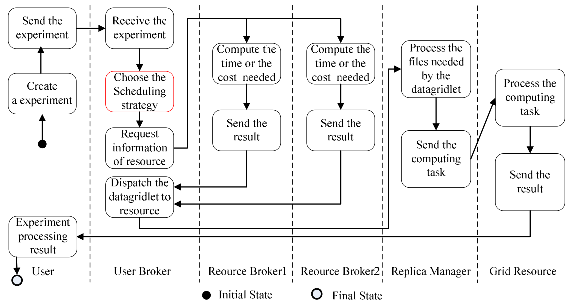

Send the experiment
Receive the experiment
Create a experiment
Choose the Scheduling strategy
Request information of resource
Dispatch the datagridlet to resource
Compute the time or the cost needed
Send the result
Compute the time or the cost needed
Send the result
Process the files needed by the datagridlet
Send the computing task
Process the computing task
Send the result
Experiment processing result
User
User Broker
Reource Broker1
Reource Broker2
Replica Manager
Grid Resource
Initial State
Final State

Figure 4. Simulation Flowchart of the Job Scheduling

6. Experiments

This simulator realizes the simulation of data grid job scheduling based on GridSim. Below a random scheduling algorithm and a cost-optimal scheduling algorithm are taken for examples to show the employment of this simulator and prove its validity.

Table 1. Parameters

num_resource	10
baud_rate (bits/sec)	10000-50000
num_user	1
num_router	4
topology	star
resource distribution	random
Machine Cost (G$/PE time unit)	30-50
Rating for a Machine MIPS	377-754
File Cost (G$/MB)	30-50

To simulate a job scheduling strategy using this simulator, the user should sets related parameters of the simulator, adds the new job scheduling strategy to the user broker, and then starts the simulator to carry on the performance analysis of the strategy. Figure 5 and 6 are the time comparison diagram and the cost comparison diagram separately using two scheduling strategies of the random scheduling and the cost-optimal scheduling separately, whose number of jobs is between 5-40 and resources uses the space-share model.

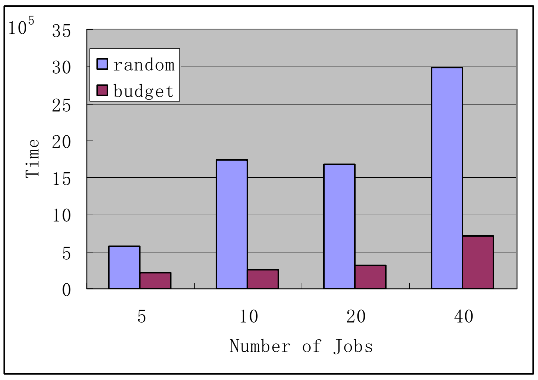


Figure 5. Time Comparison Diagram

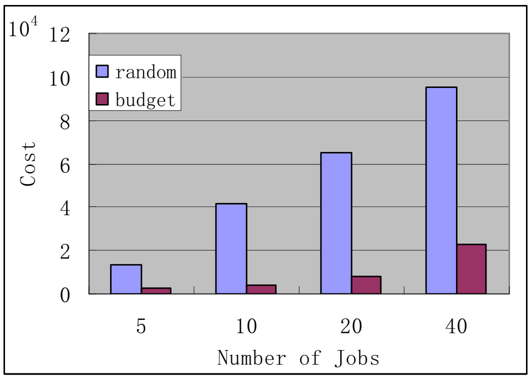


Figure 6. Cost Comparison Diagram

From Figure 5 and 6, obviously the performance of the cost-optimal scheduling algorithm is better than that of the random scheduling algorithm. Specially, the superiority of the cost-optimal scheduling algorithm is bigger along with job number's increase, which is consistent with anticipated results.

7. Conclusions

This paper studied the job scheduling of data grid, analyzed and summarized the scheduling process of data-intensity jobs, analyzed and defined data statistics of the job scheduling process. Based on the result, it expanded the GridSim simulator, added a new function of simulating the data grid job scheduling, and proved its feasibility and rationality by experiment simulation. Next step, some related problem on evaluating job scheduling strategies of the data grid will be researched.

Acknowledgements This work is supported by the National Natural Science Foundation of China (No: 90612003)and the Shandong Natural Science Foundation(No. Y2007G11).

References

[1]Chervenak A, Foster I, Kesselman C, "The data grid: towards architecture for the distributed management and analysis of large scientific data sets", *Network and computer Applications*, 2001,23, pp.187-200.

[2]Buyya R, Murshed M. "GridSim: A toolkit for the modeling and simulation of distributed resource management andscheduling for grid computing", *Concurrency and Computation: Practice and Experience 2002*,14, pp.1175–1220.

[3]Bell W, Cameron D, Capozza L, Millar P, Stockinger K, Zini F. "Simulation of dynamic Grid replication strategies in OptorSim", *Proceedings of the 3rd International Workshop on Grid Computing (GRID)*, Baltimore, U.S.A. IEEE CS Press: Los Alamitos, CA, U.S.A., 18 November 2002.

[4]Mihai Dobre C, Stratan C. "Monarc simulation framework", *Proceedings of the RoEduNet International Conference,*Timisoara, Romania, May 2004,pp. 27–28.

[5]Legrand A, Marchal L, Casanova H. "Scheduling distributed applications: The SimGrid simulation framework", *Proceedings of the 3rd International Symposium on Cluster Computing and the Grid*, Tokyo, Japan, May 2003, pp. 12–15.

[6]Song HJ, Liu X, Jakobsen D, "The MicroGrid: A scientific tool for modeling computational grids", *Proceedings of IEEE Supercomputing Conference*, Dallas, U.S.A., November 2000,pp. 4–10.

[7]Sulistio A, Buyya R. "A toolkit for modeling and simulating data Grids: an extension to GridSim", *Concurrency and Computation: Practice and Experience* 2008,20, pp.1591–1609.

[8]Srikumar Venugopal and Rajkumar Buyya, "An SCP-based Heuristic approach for Scheduling Distributed Data-Intensive Applications on Global Grids", *Journal of Parallel and Distributed Computing*, Elsevier Press, Amsterdam, 2008,68(4),pp. 471-487.

[9] Kyong Hoon Kim, Rajkumar Buyya and Jong Kim, "Power Aware Scheduling of Bag-of-Jobs Applications with Deadline Constraints on DVS-enabled Clusters", *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*, IEEE CS Press, Los Alamitos, USA, 2007,pp.14-17.

[10]Srikumar Venugopal, Krishna Nadiminti, Hussein Gibbins and Rajkumar Buyya, "Designing a Resource Broker for Heterogeneous Grids", *Software: Practice and Experience*, Wiley Press, New York, USA,2008, 38(8), pp. 793-825.