Performance Modeling and Analysis for Resource Scheduling in Data Grids*

Yajuan Li[1], Chuang Lin[1], Quanlin Li[2], and Zhiguang Shan[3]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
{yjli, clin}@csnet1.cs.tsinghua.edu.cn

[2] Department of Industrial Engineering, Tsinghua University, Beijing 100084, China
liquanlin@tsinghua.edu.cn

[3] Public Technical Service Department, State Information Center, Beijing 100045, China
shanzg@mx.cei.gov.cn

Abstract. Data Grids normally deal with large data-intensive problems on geographically distributed resources; yet, most current research on performance evaluation of resource scheduling in Data Grids is based on simulation techniques, which can only consider a limited range of scenarios. In this paper, we propose a formal framework via Stochastic Petri Nets to deal with this problem. Within this framework, we model and analyze the performance of resource scheduling in Data Grids, allowing for a wide variety of job and data scheduling algorithms. As a result of our research, we can investigate more scenarios with multiple input parameters. Moreover, we can evaluate the combined effectiveness of job and data scheduling algorithms, rather than study them separately.

1 Introduction

A Data Grid [1] connects a collection of computational and data-resources distributed geographically among multiple sites, and enables users to share these resources. To use a Data Grid, users typically submit jobs. In order for a job to be executed, two types of resources are required: computing facilities, data access and storage. The Grid must make scheduling decisions for each job based on the current state of these resources. Different job and data scheduling algorithms may bring different performance for the Data Grid.

Many research works have been done on the performance evaluation of Data Grids, but most of which use simulation techniques, which can only analyze a limited range of scenarios. For example, in [2], a special Data Grid simulator, called OptorSim, was designed to study the complex nature of a typical Grid environment and evaluate various data replication algorithms; in [3], a simulation work was developed to study

* This work is supported by the National Natural Science Foundation of China (No. 90412012, 60429202, 60372019 and 60373013), NSFC and RGC (No. 60218003), and the National Grand Fundamental Research 973 Program of China (No.2003CB314804).

H. Jin, D. Reed, and W. Jiang (Eds.): NPC 2005, LNCS 3779, pp. 32 – 39, 2005.

dynamic replication strategies; in [4], a discrete event simulator, called ChicagoSim, was constructed to evaluate the performance of different combinations of job and data scheduling algorithms. Furthermore, many related works are based on a single factor of job or data scheduling. In [2][3][5], performance is analyzed with the assumption that jobs have been allocated to certain computing elements. While in [6-9], performance is analyzed with the assumption that data have been accessed. The research to study the combined effectiveness of job and data scheduling strategies has been pointed out to be very complex [10].

We propose a formal performance evaluation framework that addresses the above mentioned issues. Within this framework, we can investigate more scenarios with multiple input parameters. Moreover, we can evaluate the combined effectiveness of job and data scheduling algorithms, rather than study them separately.

The rest of the paper is organized as follows. Section 2 describes the general and extensible scheduling architecture of Data Grids that we use for our modeling and analysis. Section 3 presents the performance model, while Section 4 analyzes the performance of the model. We conclude and point to future directions in Section 5.

2 Architecture

Our study is based on a general and extensible Data Grid scheduling architecture, which is inspired by the work presented in [4], and depicted in figure 1. The logic of the architecture can be encapsulated in three distinct modules:

- **Server.** Each server comprises a number of processors and storage. Due to the heterogeneousness of Grid environments, different server may have a different number of processors. The processors of a server can only access the local storage.
- **Client.** Each client submits jobs to schedulers. Then each job can be allocated to any of the servers and further dispatched to any of the processors of a server. Each job requires some specific data be available at the local storage before it can be executed.
- **Scheduler.** It is the core of the system and can be classified into three schedulers: external scheduler (ES), local scheduler (LS), and data scheduler (DS). (1) **External scheduler**. In the system, jobs can be classified depending on their different priority levels. Each job is submitted to some ES in terms of its priority. Once an ES receives a job, it immediately makes a decision on which server the job should be assigned to, according to some scheduling algorithm. It may use the global information, such as load of each server, and/or location of the data required by a job, as input to its decisions. (2) **Local scheduler**. When a job is delivered to some server, it is managed by the local scheduler of that server. The LS determines how to schedule the jobs allocated to it, according to its associated scheduling algorithm. It only uses the local information, such as load of each local processor, to guide its decisions. (3) **Data scheduler**. Each DS is responsible for determining if and from which server to replicate data according to some algorithm. When a job is allocated to some server, the DS in that server will query whether the data required to run the job is already present at the local storage. If not, the DS will use the global information, such as the

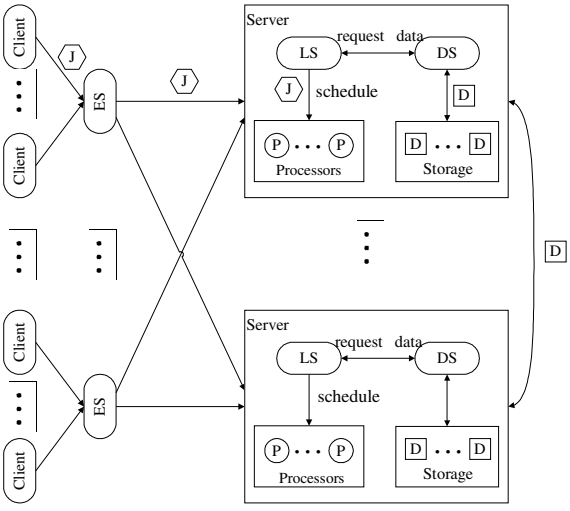

Fig. 1. A Data Grid Scheduling Architecture

availability of the required data in a remote server, and/or the distance between two servers, toreplicate the required data from some remote server to the local storage, before the job is executed.

3 SPN Model

To study the performance of resource scheduling in Data Grids, we adopt the modeling and analysis method, which allows for the performance evaluation in various scenarios. We choose the Stochastic Petri Net (SPN) [11] as the base for our study, since it is a powerful graphical and mathematical tool that is able to handle prioritized, concurrent, asynchronous, stochastic and nondeterministic events. In figure 2, we propose a SPN model of the Data Grid scheduling system.

Suppose there are n classes of jobs, the jobs in each class have the same priority level. The priority level values range from 1 (the highest priority) to n (the lowest one). Jobs with priority level i are denoted by r_i. In accordance, the clients in the system are classified into n categories. Each client in class i submits jobs r_i to ES_i according to a Poisson distribution with the same mean arrival rate.

The system consists of k servers, each of which contains a depository with an infinite capacity for storing data, and may have different compute power. To consider a general case, we assume that server x comprises m_x processors, for $1 \le x \le k$, and each processor provides the exponential distributed service durations with different mean rates for different priority-level jobs. In each processor, there are n waiting queues of jobs, each for one priority level and with an infinite capacity. Jobs in the same waiting queue are managed in FIFO (First-In-First-Out) order. If a job is in the turn to be scheduled, it can be executed only when the processor is free and its required data is available. Each processor can provide service for at most one job at any

time, and the jobs from different waiting queues are selected for service according to their priorities, i.e., jobs with higher priorities have higher priorities to be executed.

There are *n* external schedulers, each for one priority level; *k* local schedulers, each distributed in one server; and *k* data schedulers, each for one server.

The meanings of the transitions and the places are described as follows, where variable *i* identifies priority level *i* ($1 \le i \le n$), *x* and *y* denote server *x* ($1 \le x \le k$) and storage *y* ($1 \le y \le k$) respectively, *j* indicates processor *j* of server *x* ($1 \le j \le m_x$), *z* indicates client *z* ($1 \le z \le l_i$).

- Places. f_i: the external job assigner, which holds jobs r_i; a_{ix}: the transmission link from ES_i to server *x*; f_{ix}: the local job assigner of server *x*, which holds jobs r_i; q_{ij}^x: the waiting queue, which holds jobs r_i in processor *j* of server *x*; w_{ij}^x: the running state of a job r_i at processor *j* of server *x*; v_j^x: the available state of processor *j* of server *x*; g_{ij}^x: the place holding execution results of jobs r_i in processor *j* of server *x*; sd_{ij}^{xy}: the place identifying for the processor *j* of server *x* whether the storage *y* possesses the data required by jobs r_i; dm_{ij}^x: the logical module of data manager *x*, which is responsible for jobs r_i on the local processor *j*; td_{ij}^{xy}: the transmission link for data required by jobs r_i, from storage *y* to processor *j* of server *x*; ls_{ij}^x: the place holding data required by jobs r_i, that is already allocated to processor *j* of server *x*.

- Transitions. c_{iz}: the exponential transition representing that client *z* submits jobs r_i, with mean firing rate λ_i; u_{ix}: the immediate transition denoting that ES_i dispatches jobs r_i to server *x*, according to some ES algorithm; e_{ix}: the exponential transition denoting the job transmission from ES_i to server *x*, with mean firing rate β_{ix}; d_{ij}^x: the immediate transition representing that LS_x allocates jobs r_i to the local processor *j* of server *x*, according to some LS algorithm; h_{ij}^x: the immediate transition which transfers jobs r_i in processor *j* of server *x*, from waiting state to execution state; s_{ij}^x: the exponential transition denoting that processor *j* of server *x* runs jobs r_i, with mean firing rate μ_{ij}^x; ud_{ij}^{xy}: the immediate transition representing that the data monitor of processor *j* of server *x*, which collects data information for jobs r_i from storage *y* once the state of storage *y* changes; rd_{ij}^{xy}: the immediate transition representing that DS_x schedules data required by jobs r_i, from storage *y* to processor *j*, according to some DS algorithm; od_{ij}^{xy}: the exponential transition denoting that the data transmission for jobs r_i, from storage *y* to processor *j* of server *x*, with mean firing rate δ_{ij}^{xy}.

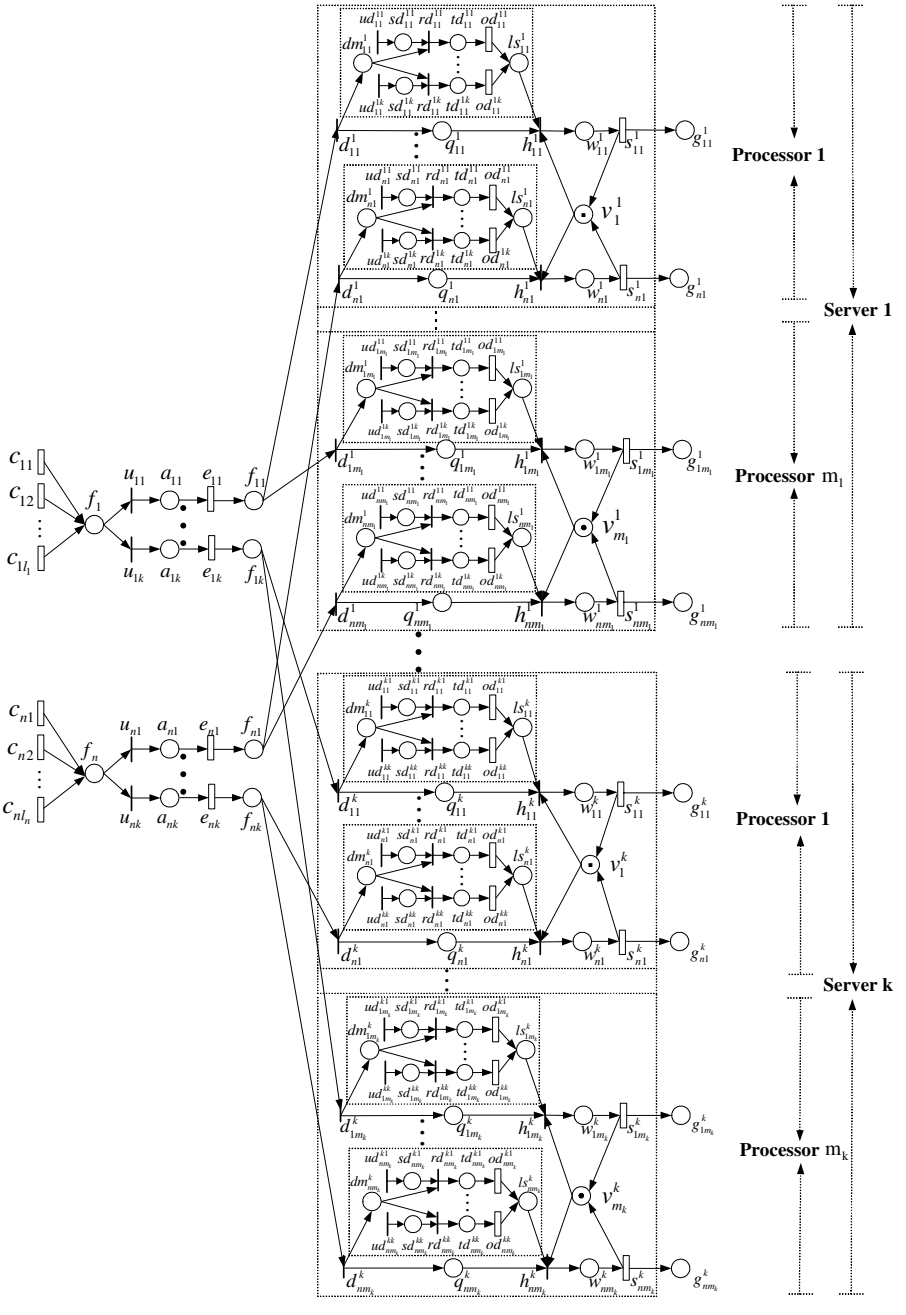

Fig. 2. A SPN Model of the Data Grid Scheduling System

4 Performance Evaluation

In SPN models, performance evaluation is based on steady-state probabilities. Since the model is very large and complicated, we adopt an approximate analysis technique to reduce the complexity of the model solution, presented as the following steps.

(1) **Refinement.** To simplify a complicated model into a relatively compact model, by deleting immediate transitions and transferring the enabling predicates associated with these immediate transitions to some exponential transitions.

(2) **Decomposition.** To decompose a model into several sub-models, by using independence and interdependence relations of the sub-models. A refined sub-model of the original model is generally described in figure 3, denoted as A_{ij}^{xy}, which represents the module that job r_i is submitted to processor j of server x, and its required data is replicated from storage y. The refined complete model is composed of $\sum_{i=1}^{k}\left(n\times k\times m_i\right)$ sub-models, which are independent with each other in structure. The interdependence relation of these sub-models is embodied by the enabling predicates associated with transitions.

(3) **Iteration.** For each sub-model, import parameters are from other sub-models; after computed, the solution result is again exported to other sub-models.

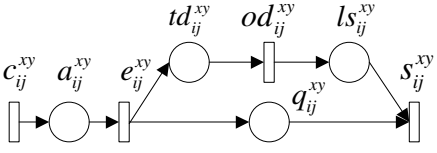

Fig. 3. A Refined Sub-Model of the Data Grid Scheduling System

It follows from the steady-state probabilities that system performance measures can be obtained.

(1) The average throughput for transition t in steady state is:

$$T\left(t\right)=\sum_{M\in H(t)}\Pr\left[M\right]\times\theta\left(t,M\right),$$

where Pr[M] is the steady-state probability of marking M, $\theta\left(t,M\right)$ is the firing rate of transition t in marking M, and $H(t)$ is the subset of reachable markings that enable t.

(2) The average number of tokens for place q in steady state is:

$$N\left(q\right)=\sum i\times\Pr\left[M\left(q\right)=i\right].$$

Following the above, we consider two important metrics of the Data Grid scheduling system: average system throughput and average job completion duration.

(1) **Average system throughput:**

$$T=\sum_{i=1}^{n}T_i,$$

where T_i is the throughput for job r_i and obtained by

$$T_i = \sum_{x=1}^{k} \sum_{j=1}^{m_x} \sum_{y=1}^{k} T\left(s_{ij}^{xy}\right).$$

(2) **Average job completion duration:**

$$JCD = \frac{\sum_{i=1}^{n} JCD_i \times T_i}{T},$$

where JCD_i is the job completion duration for job r_i and acquired by

$$JCD_i = SD_i + \sum_{x=1}^{k} PR_{ix} \times TD_{ix} + DT_i,$$

where SD_i: the submission duration of job r_i from client to ES_i, and

$$SD_i = \frac{1}{\sum_{x=1}^{k} \sum_{j=1}^{m_x} \sum_{y=1}^{k} T\left(c_{ij}^{xy}\right)},$$

PR_{ix}: the probability of job r_i being allocated to server x, and

$$PR_{ix} = \frac{\sum_{j=1}^{m_x} \sum_{y=1}^{k} T\left(s_{ij}^{xy}\right)}{\sum_{x=1}^{k} \sum_{j=1}^{m_x} \sum_{y=1}^{k} T\left(s_{ij}^{xy}\right)},$$

TD_{ix}: the transfer duration of job r_i from ES_i to LS_x, and

$$TD_{ix} = \frac{\sum_{j=1}^{m_x} \sum_{y=1}^{k} N\left(a_{ij}^{xy}\right)}{\sum_{j=1}^{m_x} \sum_{y=1}^{k} T\left(e_{ij}^{xy}\right)},$$

DT_i: the delay time of job r_i for all servers, and

$$DT_i = \frac{\sum_{x=1}^{k} \sum_{j=1}^{m_x} \sum_{y=1}^{k} N\left(td_{ij}^{xy}\right)}{\sum_{x=1}^{k} \sum_{j=1}^{m_x} \sum_{y=1}^{k} T\left(od_{ij}^{xy}\right)} + \frac{\sum_{x=1}^{k} \sum_{j=1}^{m_x} \sum_{y=1}^{k} N\left(q_{ij}^{xy}\right)}{\sum_{x=1}^{k} \sum_{j=1}^{m_x} \sum_{y=1}^{k} T\left(s_{ij}^{xy}\right)}.$$

5 Conclusions and Future Work

In this paper, we construct the SPN model based on a general and extensible scheduling architecture of Data Grids, and further evaluate the system performance. The

performance metrics considered in this paper include the system throughput and the job completion duration experienced in system.

In future work, we want to develop an analysis tool to evaluate the performance of practical Grids. Particularly, this tool is planned to be able to plug in different algorithms for selecting the best server, the best processor, and the best replication. Another area for further research is to study the sensitivities with respect to all system parameters, which will be helpful to come up with more reasonable schemes for system designs.

References

1. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., Tuecke, S.: The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. J. Network and Computer Applications Vol. 23. No. 3. (2000) 187-200
2. William, H.B., David, G.C., Luigi, C., Paul, M.A., Kurt, S., Floriano, Z.: Simulation of Dynamic Grid Replication Strategies in OptorSim. In: Proceedings of the Third International Workshop on Grid Computing. Lecture Notes in Computer Science, Vol. 2536. Springer-Verlag, London, UK (2002) 46-57
3. Ranganathan, K., Foster, I.: Identifying Dynamic Replication Strategies for a High-Performance Data Grid. In: Proceedings of the Second International Workshop on Grid Computing. Lecture Notes in Computer Science, Vol. 2242. Springer-Verlag, London, UK (2001) 75-86
4. Ranganathan, K., Foster, I.: Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids. Journal of Grid Computing, Vol. 1. No. 1. (2003) 53-62
5. Venugopal, S., Buyya, R., Lyle, J.W.: A Grid Service Broker for Scheduling Distributed Data-oriented Applications on Global Grids. In: Proceedings of the 2nd Workshop on Middleware for Grid Computing. ACM Press, USA (2004) 75-80
6. Hamscher, V., Schwiegelshohn, U., Streit, A., Yahyapour, R.: Evaluation of Job-Scheduling Strategies for Grid Computing. In: Proceedings of the First IEEE/ACM International Workshop on Grid Computing. Lecture Notes in Computer Science, Vol. 1971. Springer-Verlag, London, UK (2000) 191-202
7. James, H.A., Hawick, K.A., Coddington, P.D.: Scheduling Independent Tasks on Metacomputing Systems. Technical Report DHPC-066. University of Adelaide, Australia (1999)
8. Shirazi, B.A., Husson, A.R., Kavi, K.M. (eds.): Scheduling and Load Balancing in Parallel and Distributed Systems. IEEE Computer Society Press (1995)
9. Subramani, V., Kettimuthu, R., Srinivasan, S., Sadayappan, P.: Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing. IEEE Computer Society Press, Los Alamitos (2002) 359-367
10. Desprez, F., Vernois. A.: Simultaneous Scheduling of Replication and Computation for Data-Intensive Applications on the Grid. Technical Report RR2005-01 (2005)
11. Gianfranco, B.: Introduction to Stochastic Petri Nets. In: Lectures on Formal Methods and Performance Analysis: first EEF/Euro summer school on trends in computer science. Springer-Verlag, Berlin Heidelberg New York (2002)