Simulation of Hierarchical Real-time Task Scheduling Algorithms Using DGridSim

(Poster Abstract)

Mustafa Müjdat Atanak
Anadolu University
Department of Electrical and Electronics Engineering
26470 Eskisehir, Turkey
mmatanak@anadolu.edu.tr

Safai Tandoğan
Anadolu University
Computer Research and Application Center
26470 Eskisehir, Turkey
standogan@anadolu.edu.tr

Atakan Doğan
Anadolu University
Department of Electrical and Electronics Engineering
26470 Eskisehir, Turkey
atdogan@anadolu.edu.tr

ABSTRACT
Real-time Data Grid applications are emerging in many disciplines of science and engineering. In order to run these applications while meeting the real-time constraints associated with them, the Data Grid infrastructure should be designed to respect these constraints and allocate its computing, networking, storage, and the other resources accordingly. In this study, a process oriented and discrete-event driven all-in-one Data Grid simulator (DGridSim) supporting real-time operation through advance reservation mechanism is used to test the real time performance of a task scheduling algorithm.

Categories and Subject Descriptors
C.4 [**Computer Systems Organization**]: Performance of Systems – *measurement techniques, modeling techniques*

General Terms
Performance

Keywords
Data Grid system, real-time, task scheduling

1. INTRODUCTION
Grid systems provide researchers of various fields with the required hardware and software infrastructure of high performance computation, data storage, and communication resources for running their applications. A central component of all Grid systems is the *resource management system*, which defines the *scheduler organization* as well as the *scheduling policy* among the other functionalities [1], [2].

SIMUTools 2010 March 15–19, Torremolinos, Malaga, Spain.

As far as the scheduler organization of the Grid is concerned, there are three different implementations: centralized, hierarchical and distributed. In this study, the hierarchical scheduler organization is assumed to respect to the typical organization of Grid systems where the sites comprising the Grid have their own local resource management systems and there exists a set of global services on top of the local services serving for all Grid users. The scheduling policy administers how all related resources for running user tasks are allocated in order to optimize some performance metric(s). It is a well-known fact that the scheduling policy plays a crucial role in the performance of Grid systems.

Evaluating the performance of various scheduling policies requires repeatable and controlled experiments to be performed on Grid environments. This is a difficult task to achieve on real Grid systems because of their heterogeneous and dynamic nature. As a result, there are several simulators written for Grid simulation in the literature [3].

This work presents the results of the analysis of the performance of a task scheduling algorithm. All the tests are performed with DGridSim, a real-time Grid simulator with support of advance reservation for all Grid resources.

2. DGRIDSIM
DGridSim is written in C++ programming language using CSIM20 (for C++) discrete event simulator library [4]. DGridSim makes it possible to test different job distribution, data replication and data distribution algorithms under large number of data requests and jobs with real-time requirements. Much of the effort is spent on implementing the simulated Data Grid network as close to the real life counterpart as possible, while keeping the simulator execution time as small as possible.

DGridSim supports the advance reservation of the computing bandwidths of the compute nodes, storage spaces of the storage elements and bandwidths of the links on the Grid system. Note that the advance reservation of all related resources is required to promise the completion of tasks before their deadlines.

DGridSim models the Grid system by a collection of sites interconnected by an internet. Every site runs a collection of services related to that site. One of the sites serves as the Global Management Site (GMS) and runs the collection of services that

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8676
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8676

are global to the Grid system. Each site holds a number of computing elements to run the jobs submitted by the scheduler and storage elements to store the related data items. In this model, a site can serve as a network attached hard disk (just set the number of compute nodes to be zero) or a purely computational site (just set the number of storage elements to be zero) or a cluster of computing elements connected to a single storage element. In DGridSim, all the transfers are viewed as flows and all the bandwidths on the route are assumed to be reserved beforehand for the duration of the transfer.

3. WORKING PRINCIPLES

3.1 Grid Scheduling

Upon receiving a new job, Grid Scheduling Service (GSS) fetches the replica locations from Replica Location Service and the reservation tables from Reservation Service. Then, GSS calls for a scheduling algorithm that tries to find out a site for running the job. Once the submission site is determined, GSS sends the job to Job Dispatch Manager and removes it from the queue. It is up to Job Dispatch Manager to forward the job to the related Site Job Submission Service. It is also possible that the scheduling algorithm cannot find a site on which the job's deadline will be met. In such a case, the job stays in the queue for further scheduling trials until its deadline is passed. After that, job is considered as failed and it is removed from the queue.

3.2 Site Scheduling

Once Site Job Submission Service receives the job, it places the job in a queue and informs Site Scheduling Service (SSS) that implements either an online or offline scheduling algorithm. It is up to SSS to locate available resources for running the job.

Scheduling Service first fetches the computing element reservation tables from Reservation Service and tries to find a computing element to run the job before its deadline. If an available computing element can be found, a data request composed of the job's deadline and the related Logical File Names (LFNs) are passed to Replica Manager. Replica Manager puts the request into a queue and triggers Replica Management Service. Replica Management Service calls for Replica Location Service to find the possible source locations for all LFNs. The list of possible sources is sent back to Replica Management Service. Replica Optimization Service (ROS) is called with this list. Replica Optimization Service calculates the minimum required bandwidth, (request data size) / (deadline – current time) and gets the topology containing all the available links satisfying this minimum bandwidth for the reservation interval, [(current time), (deadline)] from Grid Information Service. Using this information, it tries to find an optimal source and optimal path for each transfer request. Replica Management Service calls Reservation Service to reserve source/destination storage elements and the links between them for the specified interval provided that ROS has found a deadline-satisfying path for every transfer request related to the job.

File Transfer Service handles the file transfers using these advance-reserved resources. If overall scheduling is successful, SSS sends the job to Job Invoke Manager and removes it from the queue. If the scheduling fails at any time during the process, the job stays in the queue for further scheduling trials. SSS removes the job if its deadline is passed. Once Job Invoke Manager receives the job, it *holds* for the duration between the start time of the job and the current time. It then sends the job to the computing element where it is executed until the completion.

4. RESULTS

Using the simulator developed, a set of simulation studies were conducted to verify that the simulator is operating as expected. These tests evaluate the distribution of real-time jobs to computing elements. In each run of the simulation, a Data Grid system is randomly generated by changing the network topology and site resources. Furthermore, a user submits jobs with a random frequency, each of which may require a random number of data items chosen from a data item set.

The base tests are conducted with a Grid system of 10 sites. The read/write bandwidths of the storage elements are assumed to be between 1 Gb/s and 3 Gb/s; the internet links have bandwidths between 500 Mb/s and 1.5 Gb/s. One hundred data items are placed on one of the storage elements. Data item sizes are between 0.5 GB and 0.75 GB. The user submits 500 jobs, each having an average deadline value of 40.0 seconds from the time they are submitted. The size of the jobs is measured in millions of instructions. In the base tests, the average size of the jobs is assumed to be one million of instructions. On the average, one job is submitted to the Grid system every second. The results of the simulation studies are presented in Table 1. Each data shown is the average of 10 simulation runs of the percentage of the real-time job requests that are successfully completed before the deadline. This percentage will be referred as the *success rate (SR)*. The success rate in the base test is 65%.

Table 1. The results of the first set of tests.

Job Size	SR (%)	Deadline	SR (%)	No Jobs	SR (%)
3	65	20	65	1000	32
50	61	10	61	250	92
100	47	5	43	100	93

5. ACKNOWLEDGEMENTS

This work is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under contract number 108E232.

6. REFERENCES

[1] Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., and Tuecke, S. 1998. A resource management architecture for metacomputing systems. Lecture Notes in Computer Science 1459.

[2] Krauter, K., Buyya, R., and Maheswaran, M. 2002. A taxonomy and survey of grid resource management systems for distributed computing. Software—Practice & Experience 32, 2) 135-164.

[3] Quetier, B. and Cappello, F. 2005. A survey of grid research tools: simulators, emulators and real life platforms. 17th IMACS World Congress (IMACS 2005).

[4] User's Guide: CSIM20 Simulation Engine (C++ Version), http://www.mesquite.com.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8676
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8676