International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:8, 2012

MICOSim: A simulator for modelling economic scheduling in Grid computing

Mohammad Bsoul, Iain Phillips, and Chris Hinde

Abstract—This paper is concerned with the design and implementation of MICOSim, an event-driven simulator written in Java for evaluating the performance of Grid entities (users, brokers and resources) under different scenarios such as varying the numbers of users, resources and brokers and varying their specifications and employed strategies.

Keywords—Grid computing; Economic Scheduling; Simulation; Event-Driven; Java.

I. Introduction

In Grid computing, the entities such as users, brokers and resources employ strategies that their performance need to be evaluated under different circumstances. Unfortunately, it is nearly impossible to evaluate the performance of different entities in a repeatable and controllable manner for different scenarios such as changing the number of entities in real Grid environments. The reason is that the availability of resources and their load change with time and it is impossible for an individual user to control actions of other users on the Grid.

Thus, a simulator is needed for evaluating the performance of different entity strategies under different scenarios. Those scenarios comprise:

- Varying the numbers of users, resources and brokers.
- Varying the entities' specifications such as varying the numbers of jobs that are owned by different users and jobs' lengths and classes, and varying the numbers of processors the resources have and processors' speeds.
- Varying the strategies that are employed by various users, brokers and resources.
- Varying the strategies' parameters like the parameters they send to other entities such as prices, completion times and deadlines.

After this introduction, Section II discusses related work including some of its limitations. Section III and IV introduce the implemented simulator's components and their interactions, respectively. Finally, section V concludes this paper.

II. Related work

Simulation is defined as "*attempting to predict aspects of the behavior of some system by creating an approximate model for it*" [1]. Building simulators has a number of advantages: there is no need for building a real system, conducting more easily controlled experiments and allowing to run a huge number of experiments. A number of simulators have been implemented such as Bricks, SimGrid, GangSim, OptorSim and GridSim.

Bricks [2] is a Java-based simulator developed at the Tokyo Institute of Technology in Japan and is used for comparing scheduling algorithms and frameworks in client-server like global computing systems under different circumstances such as varying the workload. However, it uses centralized scheduling which is known to have drawbacks such as inscalability and single point of failure.

SimGrid [3] is an event-driven simulator developed in the university of California and it deals with single-client multi-server scheduling. However, because it can only be used for simulating a single client, it is hard to simulate a group of competing users that each employs its own strategy.

SimGrid 2 [4] and 3 [5] are enhanced versions of SimGrid which have new features involving simulating distributed scheduling agents in dynamic distributed environments and supporting more network models.

GangSim [6] simulates environments where there is a large number of institutions and users that control a huge number of computers and storage systems. In GangSim, the allocation of resources is decided from the interaction between virtual organisations (VOs). In MICOSim, the allocation of resources is determined from the interaction between individual entities.

OptorSim [7] is a Data Grid simulator written in Java for evaluating replica optimiser algorithms in various grid configurations. Furthermore, it is used for evaluating an economic model using a Peer to Peer auction protocol [8]. The economic model is used to choose replicas for running jobs and to determine where to create replicas dynamically in Grid sites by employing a file revenue prediction function.

Finally, Gridsim [9] is a Java-based toolkit and is used for modelling and simulation of entities in Grid environments. It is built on top of SimJava which is a discrete event simulation engine that runs entities in separate threads. It is mainly concern is Grid economy where there are users (buyers), resources (sellers) and brokers for discovering the available resources and allocating them to user jobs. Threads have a drawback which is platform dependency, so the program runs differently under various operating systems platforms [10]. GridSim does not support that the brokers have their own strategies too in order to maximise their own utilities.

Therefore, a simulator that supports decentralized scheduling in Grid computing has to be implemented. In this simulator, all entities can have their own utilities and can interact with each other using tender model. In this paper, an event-

M. Bsoul is with the Department of Computer Science and Applications, The Hashemite University, P.O. Box 150459, Zarqa 13115, Jordan email: mbsoul@hu.edu.jo.

I. Phillips and C. Hinde are with the Department of Computer Science, Loughborough University, Garendon Wing, Holywell Park, Loughborough, LE11 3TU, United Kingdom.

driven simulator, MICOSim, that was implemented in Java and overcomes the limitations of the above simulators is described. In this simulator, threads were not used because of their drawback which is platform dependency. Additionally, this simulator was built from scratch.

III. MICOSim components

MICOSim's basic components are TheSystem, Entity, Entitystrategy and Scenario. TheSystem is the class that is responsible for handling the interaction (communications) between different entities. Entity is the class that different entities are created from. Also, each entity has an Entitystrategy that is an abstract class that contains definitions of methods that their bodies are the same for all strategies. It also contains abstract methods that are missing their bodies and that are defined in the subclasses inherited from this class such as user, broker and resource strategies. The definitions of those abstract methods in the subclasses rely on the specification of the created strategy. Scenario is the class that indicates the specifications of the performed simulation.

A. TheSystem

TheSystem is responsible for the interactions occur between various entities. For example, it informs the entities' strategies to take actions according to their order in its event list. Additionally, it forwards the messages between different entities like informing the entity that won the award of executing the job. An event list is a Vector class which contains the events that should happen through the simulation ordered by the occurrence time. An event is represented by an object which comprise the event's occurrence time, the entity's class and the entity's number. At the begining of the simulation, the list contains events that their relevant actions are initiating negotiations for user jobs. When the simulation starts, the system removes the first event from the list and executes the relevant action. Any new events that occur as a result are inserted on the list at the appropriate point. This continues until the event list becomes empty. If two or more events have the same occurrence time, then their relevant actions are executed sequentially.

B. Entity

Entity is an object that can send and receive both jobs and bids. However, the sub-classes that are created from Entity class have some of the capabilities the entity has. For example, users can only send jobs and receive bids, while resources can do the opposite. On the other hand, brokers can send and receive both jobs and bids, but cannot execute jobs as resources. Three classes of entities are created from Entity: users, brokers and resources. All entities have common things such as name which is unique, number which is used in communications to specify the recipient of job or bid parameters and strategy which is the course of action to achieve entity's goal(s).

The next sections describe the parameters and methods that are specific to each category of entities.

1) User: User is the entity that sends jobs and receives bids. The parameters that are specific to a user are:

- Number of jobs: The number of jobs the user has.
- Job IDs: The IDs (numbers) of the jobs belong to the user.
- Job lengths: The length of jobs belong to the user in Million Instructions (MI).
- Jobs' classes: The class of each job belongs to the user. The class shows the computer resources it requires when running.
- Job numbers of negotiations: How many negotiations occurred between the user and the brokers for every job.
- The number of jobs that were executed.
- Job costs: The cost the user paid for executing each of its jobs.
- The number of brokers the user knows about.
- Historical performance of each broker: Each broker has an integer value between min (very poor) and max (very good). Initially, each broker is given a value between min and max. This value is used to determine with which brokers the user keeps negotiating. This value is decreased or increased by the strategy that is employed by the user based on the occurrence of specific conditions.

Each user has also methods for informing the broker that the user accepted its bid, updating the historical performance of the broker that submitted the job that has just finished its execution, increasing the number of jobs that were executed when a job of the user is completed, increasing the number of negotiations occurred between the user and a broker and creating a new event for a new job if the submission of jobs is dynamic (e.g. the submission time of each job is determined based on the completion of the previous job).

2) Broker: The broker sends and receives jobs and bids. Every broker has a number of parameters and methods that are used during the negotiations. The broker parameters are:

- The job parameters (jobs which are owned by users who the broker acts on behalf) mentioned above like IDs, lengths and classes in addition to the IDs of the users sent them. The broker needs to keep information about the jobs submitted to it, so it can pass them later to resources which the broker will interact with.
- Jobs' numbers of negotiations: How many negotiations occurred between the broker and users, and between the broker and resources for every job submitted to the broker.
- Cost per MI: The cost resulted from acting on behalf of a job owner measured in MI.
- Cost per time unit: A continuous cost that results from maintaining the broker's requirements such as maintaining the broker's software and keeping the security up to date, and it is measured per time unit.
- Revenue: The entire amount of income before any deductions are made.
- Profit: The excess of income over expenses. The expenses are represented by cost per MI and cost per time unit.
- The number of resources the broker knows about.
- Historical performance of each resource: The same as

International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:8, 2012

historical performance in users, except it is for a resource and not for a broker.

The broker has four methods. The first method increases the number of negotiations occurred between it and a resource. The second method informs the resource that the broker accepted its bid. The third method updates the historical performance of the resource that did not meet its sent completion time for a job. Finally, the fourth method increases the broker's profit and revenue when its bid is accepted by a user.

3) Resource: The resource is an entity that receives jobs in order to execute them. Moreover, it sends bids to brokers that submitted jobs to it . As users and brokers, each resource has its parameters and methods. The resource parameters are:

- Number of processors: The number of processors the resource has.
- Number of free processors: The number of processors that are unallocated to any job.
- Processors' speeds measured in MIPS (Million Instruction per second).
- Next availabilities: The next availability for a processor is the time when the processor will be available (free). In other words, it is the time when the job currently executing on the processor in addition to the jobs that currently in the execution queue if there are any will be completed.
- Total number of jobs that were executed on the resource.
- Total number of jobs that are waiting in the execution queue or currently executing on the resource.
- Jobs' numbers of negotiations: How many negotiations took place between the resource and brokers for every job submitted to the resource.
- Cost per MI: The cost resulted from executing the job measured in MI.
- Cost per time unit: A continuous cost that results from maintaining the resource's requirements like sustaining its machines, software and security and it is measured per time unit.
- Revenue: The same as revenue in brokers.
- Profit: The same as profit in brokers.

Each resource comprises the following methods that are called when the resource:

- receives a job to update its profit, revenue, number of free processors, the availability of the processor allocated to the job and the number of jobs waiting in the execution queue or currently executing. Additionally, an object that contains information about the submission of the job is created. For example, it contains information about when the job was submitted and when it is going to be received by its owner.
- finishes executing a job for updating the number of executed jobs on the resource and the numbers of jobs that are waiting in the execution queue or currently executing.

C. Entitystrategy

As mentioned above, Entitystrategy contains definitions for the methods that their bodies are the same for all inherited strategies, and misses definitions for the abstract methods that their contents rely on the characteristics of the inherited classes. Entitystrategy has two defined methods for copying:

- job parameters from the entity (user or broker) that employs it in order for TheSystem to send them to a group of entities (broker(s) or resource(s)).
- bid parameters from the entity (broker or resource) that employs it in order for TheSystem to send them to a group of entities (user(s) or broker(s)).

On the other hand, Entitystrategy has also four abstract methods. The first method returns an object containing the parameters of the strategy (subclass) for a particular job. The second method determines how the strategy behaves. The third method is used for updating the availability of the processor allocated to a job. Finally, the fourth method is used to calculate the expected time needed to finish executing the job. The last two methods are only used by resource strategies.

There are three kinds of strategies that are user, broker and resource strategies. The sections below describe the parameters of each kind of strategy.

1) User strategy parameters: Each strategy comprises a number of parameters such as current deadlines and prices of jobs belong to the user who employs the strategy, to when the user wants to wait for each job before making a decision and other parameters that are specific to the strategy.

2) Broker strategy parameters: Each strategy controls a number of parameters that are:

- The broker's (that employs the strategy) current prices, deadlines, and completion times for users' jobs that acts on behalf of them.
- Expiry dates of broker's sent bids.
- A parameter that specifies if the broker sent a new bid in the last negotiation or not.
- Waiting time which determines until when the broker wants to wait before making a decision.
- Other variables that are specific to the strategy.

3) Resource strategy parameters: The parameters that are included in each resource strategy are:

- The resource's (that employs the strategy) current prices and completion times for users' jobs submitted by brokers to the resource which employs it.
- The job classes that are acceptable.
- A parameter that specifies if the resource sent a new bid in the last negotiation or not.
- Expiry dates of resource's sent bids.
- Other variables that are specific to the strategy.

D. Scenario

This class indicates the characteristics of the performed experiment involving:

- The kind of submission (e.g. static, dynamic).
- Number of users and number and lengths of jobs owned by each of them.
- Number of brokers.
- Number of resources and number and speeds of processors belong to each of them.
- The strategies employed by different entities.

- Costs per MI and costs per time unit of brokers and resources.
- Strategies' related parameters such as increment in price, initial price and maximum acceptable price.

IV. MICOSim components' interaction

Fig. 1 shows the interaction occurs between MICOSim's components. First of all, Scenario's parameters are passed to TheSystem. Then, TheSystem checks the first event in its eventlist to know which entity should start the negotiation first. Next, that entity employs its strategy to know the course of action to be taken. Then, the strategy adds a new event to the TheSystem event list that specifies with what entities it wants to negotiate. Furthermore, the strategy copies the needed parameters to the appropriate object and that will be used by the entities that will respond to the negotiation. Then, TheSystem deletes the current event and checks the next event to see what should happen next. This continues until the eventlist is empty.

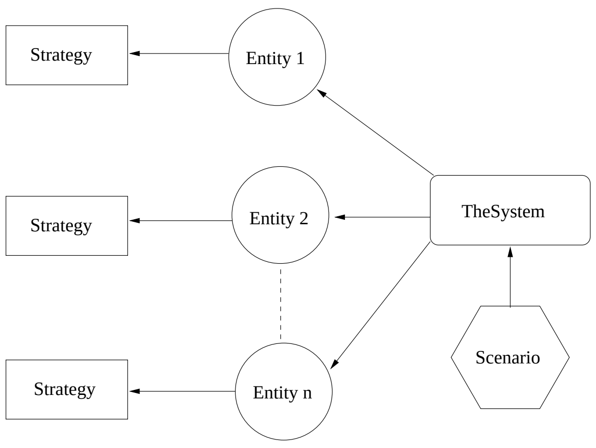

Fig. 1. Interaction between MICOSim's components.

Fig. 2 shows the negotiation occurs between a user and a broker. First, the user employs its strategy and then copies the required parameters for negotiation to a job description. Then, the job description is passed to TheSystem which in turn forwards it to the appropriate broker when its time to take an action comes. When the broker employs its strategy, it copies the required parameters to a bid description which is passed to the TheSystem. Eventually, the TheSystem sends the bid description to the user when its time to take an action is reached.

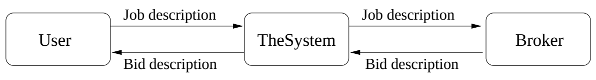

Fig. 2. Negotiation between a user and a broker.

On the other hand, Fig. 3 shows the negotiation happens between a broker and a resource. What happens is similar to what mentioned above for Fig. 2 except that the broker passes a job description to the TheSystem, and the resource passes a bid description to the TheSystem.

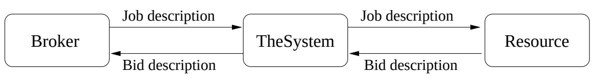

Fig. 3. Negotiation between a broker and a resource.

V. Conclusion

In this paper, a simulator called MICOSim, for modelling economic scheduling in Grid computing using tender model has been described. It has the ability to simulate Grid entities under different scenarios. The users can have different number of jobs with different lengths measured by MI, while resources can have different number of processors with different speeds measured by MIPS. Additionally, the users can employ various strategies with different parameters such as prices and deadlines as well as resources that can employ various strategies with different parameters like prices and completion times. MICOSim also simulates brokers with different interests for acting on behalf of users.

This paper has also presented the MICOSim's components including the required parameters and methods for each of them. Finally, the interaction occurs between various components of MICOSim has been introduced.

This object-oriented simulator can be easily extended to support more models such as commodity and auction models. This can be achieved by modifying some of the classes which compose the simulator.

References

[1] ProModel Corporation, "What is simulation?" Available at http://www.promodel.com/simulation.asp.
[2] A. Takefusa, S. Matsuoka, K. Aida, H. Nakada, and U. Nagashima, "Overview of a performance evaluation system for global computing scheduling algorithms," in *HPDC '99: Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 97–104.
[3] H. Casanova, "Simgrid: A toolkit for the simulation of application scheduling," *ccgrid*, vol. 00, pp. 430–437, 2001.
[4] A. Legrand, L. Marchal, and H. Casanova, "Scheduling distributed applications: the simgrid simulation framework," *ccgrid*, vol. 00, pp. 138–145, 2003.
[5] A. Legrand, "Simgrid 3.0 is out," Available at http://gforge.inria.fr/forum/forum.php?forum_id=234.
[6] C. Dumitrescu and I. T. Foster, "Gangsim: a simulator for grid scheduling studies." in *CCGRID*, Cardiff, UK, 2005, pp. 1151–1158.
[7] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini, "Optorsim: A Grid simulator for studying dynamic data replication strategies," *IJHPCA*, vol. 17, no. 4, pp. 403–416, Winter 2003.
[8] ——, "Simulation of dynamic grid replication strategies in optorsim," in *GRID '02: Proceedings of the Third International Workshop on Grid Computing*. London, UK: Springer-Verlag, 2002, pp. 46–57.
[9] R. Buyya and M. Murshed, "Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing." *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.
[10] A. Holub, "Programming java threads in the real world, part 1," Available at http://www.javaworld.com/jw-09-1998/jw-09-threads.html.