Towards Scalable Simulation of Large Scale Distributed Systems

Ciprian Dobre*, Florin Pop*, Valentin Cristea*
*Faculty of Automatics and Computer Science, University Politehnica of Bucharest, Romania

E-mails: {ciprian.dobre, florin.pop, valentin.cristea}@cs.pub.ro

Abstract

The field of modeling and simulation was long-time seen as a viable alternative to develop new algorithms and technologies and to enable the development of large-scale distributed systems, where analytical validations are prohibited by the nature of the encountered problems. The use of discrete-event simulators in the design and development of large scale distributed systems is appealing due to their efficiency and scalability. In this paper we focus on the challenge to enable scalable, high-level, online simulation of applications, middleware, resources and networks to support scientific and systematic study of Grid and P2P applications and environments. We describe alternatives to designing and implementing simulators to be used in the validation of distributed systems, particularly Grid and P2Ps.

Keywords: Distributed Systems, Grid Computing, Modeling and Simulation, Performance Analysis.

1. Introduction

In the broad area of distributed systems researchers often ask questions such as which scheduling algorithm is best suitable for deploying an application on a Grid system or which caching strategy serves better a community of users that are working in distributed data analysis. Answers to such questions are obtained in several ways. A solution is to develop purely analytical or mathematical models, but this often leads to NP-complete problems, such as routing, partitioning or scheduling, for which no analytical solution can be found. Another solution consists in conducting live experiments. Unfortunately there are no standard approaches to conduct live experiments on large scale distributed systems. Real-world experiments can be time-intensive, since the execution of applications could last for hours, days, month or even more. And to make things worse, live experiments are limited to testbeds (the particular capabilities of the testbed can affect the outcome) and the obtained results can not be reproduced by others (which is in fact the basis for scientific advances). Because of such problems *simulation* can prove to be a more elegant solution to conduct experiments. In this paper we focus on the challenge to enable scalable, high-level, online simulation of applications, middleware, resources and networks to support scientific and systematic study of Grid and P2P applications and environments. We present the characteristics of large scale distributed systems such as Grids and P2Ps in terms of modeling and simulation. We present the design considerations of simulation models designed for evaluating distributed systems solutions. Such models present special features, as they incorporate specific components and preserve characteristics necessary to design realistic simulation experiments of complex distributed system architectures, consisting of many resources and various technologies, working together to provide a common set of characteristics.

The rest of this paper is organized as follows. Section 2 presents the characteristics of distributed systems and their influence on the development of simulation models. Section 3 emphasize on the specific requirements pose by large scale distributed systems, as large scale distributed systems, such as Grids, on the simulation models. Finally, in Section 4 we present the conclusions.

2. Distributed systems and their influence on simulation models

Simulation is based on the use of formalized models of real-world systems that not only characterize the relationships inherent in the system, either mathematically, logically, or symbolically, but also are recognizable and executable by computers. In order to be useful, a simulation model designed for evaluating large scale distributed systems should incorporate several components and characteristics that are specific to the real-world systems.

The original distributed system architecture consists of a collection of autonomous machines connected by communication networks and running software systems designed to produce an integrated and consistent computing environment. Distributed systems represent a solution to enable people to cooperate and coordinate their activities more effectively and efficiently. The key characteristics of a distributed system are: resource sharing, openness, concurrency, scalability, fault-tolerance and transparency (Coulouris, et al, 1994).

In a distributed system, the resources – hardware, software and data – can easily be shared among users. For example, a database server can be shared among a group of users. The resource sharing characteristic is ensured by the use of networking components connecting various resources. This characteristic is also important for a simulation model. An adequate model should incorporate a wide set of networking components and protocols to facilitate the modeling of data communication between distributed resources. The simulated entities, included in the model, should also easily share various resources (e.g., a database server could be modeled as being accessible to various tasks running in different locations).

The openness characteristic of a distributed system is related to specifying key software interfaces to the system and making them available to software developers so that the system can be extended in many ways. A modeling design preserves this characteristic by following an object-oriented programming approach. The user should also be presented with APIs for interfacing with existing simulated components. He should easily extend the modeling framework with its own modeling components. In addition, the simulated tasks should be able to access various modeling actors such as the networking stack, modeled database servers or processing units by using standard interfaces.

The processing concurrency can be achieved by sending requests to multiple machines connected by networks at the same time. For the simulation model this property is equivalent to many simulated processes competing for the same computational resources, as well as concurrent data transfers competing for the same networking resources. A simulation model should therefore incorporate special mechanisms to allow the modeling of concurrent data transfers and processes competing for the same resources. As an example, in MONARC (Dobre&Cristea, 2007) concurrency is achieved using a specialized interrupt mechanism.

The scalability characteristic is important because a distributed system composed of a small number of machines should be easily extended to a large number of machines to increase the processing power. In order to preserve this property an adequate simulation model should consider the case of multiple resources connected by simulated networking entities. A modeled distributed system should, therefore, consist of many simulated machines, each composed of both computing and storage elements. A model should furthermore consider the case of a large number of simulated machines, and even allow the dynamical addition of many others. This could be accomplished using an object-oriented design for the simulation model, which could translate in allowing the addition of many instances into a simulation experiment. The number of simulated resources should only be limited by the physical resources available in the system where the simulation is being executed. The number of threads necessary to handle the various simulated actors is one physical limitation. The solution consists in allowing one thread to simultaneously handle multiple such actors, grouping them all together based on some logic. For example, all jobs being concurrently executed on a single processing unit could be handled in the simulation model by one single physical thread. The same idea could be applied in case of the network model, were all packets or flows traversing one network link could be handled, if necessary, by only one thread. An alternative (more expensive) solution consists in researching and using some form of distributed simulation algorithm such that to use the resources of multiple workstations.

Conducting simulation experiments is time consuming for several reasons. First, the design of sufficiently detailed models requires in depth modeling skills and usually extensive model development efforts. The availability of sophisticated modeling tools today significantly reduces development time by standardized model libraries and user friendly interfaces. Second, once a simulation model is specified, the simulation run can take exceedingly long to execute. This is due either to the objective of the simulation, or the nature of the simulated model. For statistical reasons it might for example be necessary to perform a whole series of simulation runs to establish the required confidence in the performance meters obtained by the simulation, or in other words make confidence intervals sufficiently small (Weske, 2001).

Possibilities to resolve these shortcomings can be found in several methods, one of which is the use of statistical knowledge to prune the number of required simulation runs. Statistical methods like variance reduction can be used to avoid the generation of "unnecessary" system evolutions. Statistical significance can be preserved with a smaller number of evolutions given the variance of a single random estimate can be reduced. Importance sampling methods can be effective in reducing computational efforts as well. Naturally, however, faster simulations can be obtained by using more computational resources, particularly multiple processors operating in parallel. It seems obvious, at least for simulation models reflecting real life distributed systems, consisting of components operating in parallel, that this inherent parallelism could be exploited to make better use of all physical computing resources more effectively. One way of copping up with the increasingly power demand coming from the simulation scenarios nowadays is to make use of more processor units, running on different architectures and dispersed around a larger area, in other words one way of keeping up with the simulating scenarios is to distribute the simulation application. Unfortunately, despite over two decades of research, the technology of distributed simulations has not significantly impressed the general simulation community (Fujimoto, 1993). Considerable efforts and expertise are still required to develop efficient simulation programs. There are no "golden rules" that a programmer can follow to guarantee an efficient program.

The fault-tolerance characteristic refers to the capability of distributed systems to detect and recover from faults occurring in various layers of the systems. The faults can be of various types, occurring in hardware or software; their occurrences can be transient or permanent. In distributed systems the failure of one machine can be tolerated, for example, if its functionality can be easily replaced by another redundant stand-by machine. So, machines connected by networks can be seen as redundant resources. A software system can be installed on multiple machines so that in the face of hardware faults or software failures, the faults or failures can be detected and tolerated by other machines. In order to validate fault-tolerance solutions for distributed systems a simulation model should at least incorporate the capability to simulate faults in various levels (applications, processing units, network links), to make use of various fault detection schemes or fault recovery procedures. Considering for example the case of a scheduler allocating jobs to be executed on the underlying distributed resources of a modeled system, a possible fault recovery solution that might be evaluated with a well-designed simulation model consists in allowing the scheduling algorithm to take immediate actions to use the remaining running resources.

Distributed systems can provide many forms of transparency such as location transparency, which allows local and remote information to be accessed in a unified way, failure transparency, which enables the masking of failures automatically, and replication transparency, which allows duplicating software/data on multiple machines invisibly. In order to preserve such characteristics into the designed experiment, a simulation model could include several possible mechanisms. For example, in case of network failures, the model could include algorithms that would automatically reroute, if possible, the transiting networking transfers. In case of data handling services failing, replication mechanisms could be used to ensure data consistency. Automation mechanisms could also run in the simulation scenario to ensure data consistency among replicas, as well as to save the data for longer-term usage (write the data in simulated tape deposits for example). A simulated scheduling algorithm could ensure, using various mechanisms, the correct execution of simulated jobs in case of failures occurring in the underlying resources under failure-transparency environments.

A useful simulation model must incorporate many, if not all, of the components and characteristics of Grid and P2P systems. An important aspect of a distributed system is the architecture that defines the system components, specifying the purpose and function of these components, and indicating their interactions. The analysis of the Grid and P2P architectures is a crucial aspect for developing useful simulation models. The functional requirements of the architecture influence the decision process on what the simulation model should comprise in terms of simulation entities and what properties and characteristics must be preserved.

Table 1. The characteristics of distributed systems and their influence on a simulation model.

Characteristic	**Possible influence on a simulation model**
Resource sharing	Use of networking components and data sharing entities
Openness	Inclusion of easily extendable object-oriented modeling infrastructure and standard interfaces that allow access to the fabric components inside a running simulation experiment
Concurrency	Inclusion of mechanisms to model concurrent processes and networking transfers (possible based on some interrupt mechanisms)
Scalability	The adoption of an object-oriented simulation model and the use of advanced internal structures to make better use of available physical resources of the underlying stations
Fault-tolerance	Mechanisms to model the occurrence of faults and the possibility to include mechanisms to detect and recover from occurring faults
Transparency	Use of advanced routing algorithms, data replication algorithms, and scheduling algorithms to consider failure-transparency

3. Characteristics of large scale distributed systems

Large scale distributed systems, such as Grids, are complex systems that present specific characteristics. We are mainly referring to the characteristics of Grids because the hype around Grid computing is not about replacing current technologies but harnessing them all together (cluster computing, web services, P2P, etc.) to work as one unified utility. Grid computing incorporates many of these technologies, together with their characteristics.

According to Bote-Lorenzo (2002), the characteristics of a Grid system can be summarized into 10 main features. First of all, a Grid must be able to deal with a number of resources ranging from just a few to millions. The large scale characteristic of a Grid brings up the very serious problem of avoiding potential performance degradation as the grid size increases. In order to consider this characteristic, a simulation framework for Grids should allow the modeling of scenarios consisting of a large number of nodes (ranging from few hundreds to thousands). The model should also allow the dynamical addition of other nodes into a running simulation experiment. This could be possible by following an object-oriented approach, allowing the addition of many instances of a simulated resource. In this sense, the simulation model should be scalable, the number of resources being limited only by the amount of the physical resources of the system where the simulation is being executed. Careful consideration on the modeling engine implementation, the implementation of advanced structures and algorithms at this level, should allow the experimenting with scenarios involving a large number of resources.

Another characteristic of a Grid system is the geographical distribution of its resources. The resources pertaining to a grid may be located at distant places. A simulation model should consider that the underlying simulated system is composed of many resources organized into sites, each one being located in geographical distributed locations. The simulated networking stacks should also include special WAN components that connect such distributed farms of resources.

The heterogeneity, the vast range of technologies comprising a Grid system, both software and hardware, is one other characteristic. A Grid hosts both software and hardware resources, ranging from data, files, software components or programs to sensors, scientific instruments, display devices, computers, super-computers and networks. The simulation model can include various hardware components, each having different characteristics. For example, the simulation model can include both mobile components and several other types of hard servers. The diversity in hardware architectures can also be achieved by composing many different components. For example, a dual-processor server can be modeled by using two single-processor modeling nodes, together with several communication constructs. The software heterogeneity could also be modeled using various probability distributions that should be included the simulated model. For example, in case of a data transfer application, the effective quality of the transfer is subject to many influences coming from the software itself or from the underlying networking resources. The actual random fluctuations appearing in the data transfers can be modeled by generating various interrupts in the transmission according to various probability distributions. The network heterogeneity should also be considered by the network model, by means of diverse characteristics as well as protocols being used.

Resource sharing is one other characteristic. Resources in a Grid belong to many different organizations that allow other organizations (i.e. users) to access them. Resources, other than the local ones, can be used by applications, promoting efficiency and reducing costs. Yet, this is also one of the main stops in large-scale acceptance of Grid computing, due to problems such as server-hugging or enterprise politics. The simulation model should preserve this characteristic by adopting a mature networking model, with components that simulate the connectivity among the resources being modeled.

Table 2. The influence of the Grid characteristics on a simulator.

Grid characteristic	Influence on the simulation framework
Large scale	Careful design consideration for the simulation model: the use of advanced internal structure could allow the modeling of experiments with many incorporated resources.
Geographical distribution	The inclusion of sites, geographically distributed, in the simulation model. The sites should be connected by special WAN modeled links.
Heterogeneity	Use of various models for hardware components; software architectures captured using probability distributions.
Resource sharing	Represented in the network model.
Multiple administration	Inclusion of a distributed scheduler.
Resource coordination	Resource coordination mechanisms.
Dependable access	Implementation of DAG scheduling algorithms.
Consistent access	Use of standard methods to access the resources.
Pervasive access	The scheduling framework detecting faults and taking appropriate actions.

The resource sharing characteristic is related to the multiple administrations feature. Each organization may establish different security and administrative policies under which their owned resources can be accessed and used. As a result, the already challenging network security problem is complicated even more with the need of taking into account all different policies. This characteristic can translate in the simulation model in the adoption of a distributed scheduler component. Each regional center can contain a local scheduler, and each scheduler can use its own policy to handle the locally available resources.

One other characteristic is resource coordination. The resources of a Grid must be coordinated in order to provide aggregated computing capabilities. A Grid aggregates many resources and therefore provides an aggregation of the individual resources into a higher capacity virtual resource. The capability of individual resources is preserved. As a consequence, from a global standpoint the Grid enables running larger applications faster (aggregation capacity), while from a local standpoint the Grid enables running new applications. In a simulation model, if the resource coordination capability is considered, a data replication scenario, for example, can be better simulated by using many geographically disparate sites with multiple simulated database servers, thus making good use of the underlying Grid resources. Such resource coordination mechanisms should be part of an adequate simulation model.

One important aspect provided by any Grid system is the transparent access, meaning the user should see the Grid as a single virtual computer. The Grid provides single-sign-on access to any user accessing the system. The possibility to ensure transparency in the simulation model was described in the analysis of the distributed systems presented in the previous section.

A Grid must also assure dependable access or the guaranty to deliver services under established Quality of Service (QoS) requirements. The need for dependable service is fundamental since users require assurances that they will receive predictable, sustained and often high levels of performance. In order to preserve this characteristic, a simulation model should also allow the definition of QoS metrics. In order to impose such metrics various politics could be implemented in several components. For example, the scheduler algorithm should consider deadline restrictions, the restrictions that job definitions impose (Dobre et al. 2009). They should also allow the modeling of DAG scheduling algorithms, where the submitted job could have dependencies specified. In order to preserve QoS requirements, a simulation model should also include a monitoring component that reports when problem appear. This monitoring capability could be implemented, for example, with the help of a resource catalogue (Pop et al. 2008).

According to the next characteristic, consistent access, a Grid must be constructed with standard services, protocols and interfaces, thus hiding the heterogeneity of the resources while allowing its scalability. Without such standards, application development and pervasive use would not be possible. Every resource being simulated should extend a specific object abstraction in order to preserve this characteristic. The simulation model should also provide standard methods to access the simulated resources.

Finally, pervasive access means that a Grid must grant access to available resources by adapting to a dynamic environment in which resources do fail. This does not imply that resources are everywhere or universally available but that the Grid must tailor its behavior as to extract the maximum performance from the available resources. In the presence of faults in a simulation experiment the scheduling algorithm could take appropriate actions to use the remaining resources. To this date, not many simulators for distributed systems allow the evaluation of fault-tolerance solutions.

The characteristics of the Grid systems must influence the development process of a simulation model specifically designed for Grid technologies. To summarize the specific elements of the simulation model that enables the correct modeling of a Grid environment we can refer to the study conducted in Bagchi (2005). As such, the author presents the features that must be implemented by a simulation model in order to allow the correct modeling of a Grid environment. The identified set of features consists of: multi-tasking IT resources, job decomposition, task parallelization, heterogeneous resources, resource scheduling, and resource provisioning.

The simulation model must incorporate processing units, database servers, network links, and data storage devices (multi-tasking IT resources). The modeled processing unit should consider the case of several tasks being concurrently processed by the resource. An interrupt mechanism could ensure, for example, the modeling of concurrency. In this case, the time needed to complete a task is proportional with the number of other competing tasks. A detailed simulation of the task management within each resource is far too time-consuming when considering Grid environments, with hundreds of resources simulated for the duration of weeks. Instead, when a new task is submitted to a resource, the simulation framework should perform a good approximation of the multi-tasking behavior by re-estimating the completion time of all tasks being processed by that resource.

A workload could be composed of several jobs (job decomposition), each one having multiple resource requirements. In this sense, a simulated application could be composed of several jobs, handling various actions with several resource requirements. A job can, for example, be programmed by the user to request data from a database server, to perform some computation using the obtained records and then send the results for further processing to another job. The tasks performed by a job can be correlated with the tasks performed by another one. The dependencies between the jobs can be specified in the form of DAG structures in the simulation model.

A job can be furthermore decomposed into several tasks, each one representing a single resource requirement within the job. Each task in a job may be parallelizable. In the simulation model this can be accomplished if considering a job as being composed of several parallel jobs, each one modeling the action of some task. A simulated job can start new simulated jobs, each one performing specific tasks. This, correlated with the job decomposition characteristic implemented using the DAG structures, can be used to handle the case of task parallelization.

Grid resources are heterogeneous by their nature. Therefore, the processing time of a task on a resource is subject to performance benchmarks. In the simulation model, the processing entities, as well as other simulated entities, must also be defined in terms of benchmarking units. The computation, data and network models should consider resources with various characteristics and their parameters be generically defined in order to simulate the heterogeneity of resources in the simulation experiments.

Also, a Grid simulator must be able to model scheduling policies used by resource brokers to determine on which resource a task will be executed. The simulation scheduler could provide a more advanced scheduling implementation, such as a meta-scheduler, allowing the execution the simulated jobs in a distributed manner, using all sites available. The meta-scheduler could also incorporate a wide-range of user-defined scheduling algorithms. Local scheduling is essential for a Grid simulation experiments. In the same time, user-defined scheduling algorithms should be easily added in a simulation experiment.

A simulation model should also incorporate the ability to provision resources for processing particular types of tasks. The provisioning policies could be either calendar-based or based on a more dynamic policy. The simulation model should consider the existence of background jobs to handle the execution of specific resource provisioning tasks. Such jobs can be used in dependency with other simulated jobs. In addition, the database server can perform programmed actions, being modeled as a special task in the simulation model. It can simulate special operations such as data archiving on a calendar based designed policy. The automation of resource provision is particularly important to the simulation model because it can be used to experiment with various data replication algorithms and provide flexibility to the simulation scenarios.

4. Conclusions

In this paper we described new alternatives to designing and implementing simulators to be used in the validation of distributed system technologies, particularly Grid and P2P related. We presented the challenges and solutions to the problem of enabling scalable, high-level, online simulation of applications, middleware, resources and networks to support scientific and systematic study of grid applications and environments. We described new alternatives to modeling, designing and implementing simulation instruments for distributed system technologies. Such systems consist of many resources and various technologies, ranging from data transferring to scheduling and data replication, with resources working together to provide a common set of characteristics. Therefore, in order to be useful, a simulation model must preserve several characteristics and include a wide range of possible modeled entities.

5. References

[1] G. Coulouris, J. Dollimore, T. Kindberg, "*Distributed Systems – Concepts and Design"*, Addison-Wesley, second edition, 1994.

[2] C. Dobre, V. Cristea, "*A Simulation Model for Large Scale Distributed Systems*", Proc. of the 4th International Conference on Innovations in Information Technology, Dubai, United Arab Emirates, November 2007.

[3] S. Bagchi, "*Simulation Of Grid Computing Infrastructures: Challenges and Solutions*", Winter Simulation Conference, pp. 1773-1780, ACM, 1994.

[4] M. Bote-Lorenzo, Y. Dimitriadis, & E. Gomez-Sanchez, "*Grid characteristics and uses: a grid definition"*, Tehnical Report CICYT, Univ. of Valladolid, Spain, 2002.

[5] R.M. Fujimoto, "*Parallel discrete event simulation: Will the field survive?*", ORSA Journal on Computing, 5(3), 213-230, 1993.

[6] Ciprian Dobre, Florin Pop, Valentin Cristea, "*Simulation Framework for the Evaluation of Dependable Distributed Systems*", Scalable Computing: Practice and Experience, Scientific International Journal for Parallel and Distributed Computing (SCPE), Volume 10, Number 1, pp. 13–23.

[7] Florin Pop, Ciprian Dobre, Valentin Cristea, "*Evaluation of Multi-Objective Decentralized Scheduling for Applications in Grid Environment*", Proceedings of 2008 IEEE 4th International Conference on Intelligent Computer Communication and Processing, pp. 231-238, August 28-30, 2008, Cluj-Napoca, Romania, Published by IEEE Computer Society, ISBN: 978-1-4244-2673-7

[8] Weske, M. and Wirtz, G. 2001. Integrated Modeling of Distributed Software Systems and Workflow Applications. In Proceedings of the 34th Annual Hawaii international Conference on System Sciences (Hicss-34)-Volume 9 - Volume 9 (January 03 - 06, 2001). HICSS. IEEE Computer Society, Washington, DC, 9035.