UDC 004.9
DOI: 10.15827/2311-6749.21.2.2

Modeling the reliability of geographically distributed computer systems

A.I Tikhomirov [1], *Ph.D. (Engineering), tema4277@rambler.ru*
A.V. Baranov [1], *Ph.D. (Engineering), Anton.Baranov@jscc.ru*
P.N. Telegin [1], *Ph.D. (Engineering), pnt@jscc.ru*

[1] *Joint Supercomputer Center of the Russian Academy of Sciences, Moscow, 119334, Russian Federation*

The paper presents an overview of platforms for simulation of distributed computing networks. The authors have chosen a modeling platform that has the required functionality to study the problem of fault tolerance of a distributed job management system, as well as to study the problem of scheduling user jobs for geographically distributed resources. There is an investigation on the GridSim modeling platform for building a distributed network model. Two models were implemented: one is for investigating fault tolerance of a distributed network job management system problem, and the other one is for investigating auction methods for scheduling user jobs for geographically distributed resources. The authors have formulated the advantages and disadvantages of distributed network modeling. The paper presents an alternative approach requiring less computational resources.

Keywords: *modeling platform, toolbox, distributed computing network modeling, GridSim.*

With the increasing complexity of software and hardware systems used to implement a geographically distributed network of supercomputer centers (GDN), it is necessary to develop new methods allowing to adequately reaching reliability and efficiency. Currently, the abstract models not based on specific hardware units are most efficient for analyzing reliability. At the first design stage, these models make it possible to determine the optimal reliability of logical subsystems. At the subsequent design stages, in order to move from abstract functional blocks in the reliability model to more specific subsystems that can be brought to hardware and software implementation, these logical subsystems should be considered in more detail. In other words, there is a problem of determining the reliability of a parallel system consisting of a large number of elements.

The question of GDN efficiency is related to the task of scheduling user computing jobs for geographically distributed computational resources. Like the task of assessing reliability, the task of assessing efficiency can be solved using abstract functional blocks. However, the results will be of limited use, since most GDNs are characterized by such basic properties as heterogeneity and non-expropriation of resources. These properties make the task of evaluating effectiveness difficult.

Both problems can be solved using analytical and simulation methods. In analytical modeling, a distributed network is represented as a queuing network. Optimizing the parameters of queuing networks is often a time-consuming and resource-intensive job. Therefore, we used time-consuming analytical models.

Simulation modeling is the most efficient method for studying distributed networks, and sometimes it is practically the only available method for obtaining information about the network behavior, especially at the design stage. Simulation modeling can use either a network layout, or a modeling platform. Creating a layout of a distributed network that simulates the real state as precisely as possible is a difficult and sometimes completely unsolvable task. Other members of the scientific community can rarely verify the results obtained using the simulation model, and the model itself is rarely used for another research.

The use of a simulation modeling platform should reduce the time for developing a model and conducting an experiment on the one hand, and should standardize the model building process on the other hand, which in turn will allow verification and use of the results in other studies. Indeed, the modeling platform contains a set of software tools for the network model description. Using these tools allows concentrating on exploring the model characteristics. At the same time, it is reasonable to assume that the possibility of previously obtained results reproduction will speed up new studies, as well as increase the efficiency of the existing ones.

The paper reviews existing modeling platforms for distributed networks. It also justifies the choice of a modeling platform for the task of calculating GDN reliability and efficiency indicators.

Choosing a modeling platform

The paper [1] formulates a number of basic properties of the modeling platform, the implementation of which guarantees its efficiency. First, the platform should provide usability, so that a researcher can focus on the study of the problem being solved, and not on the issues of modeling a real system. Second, the launch of the model and the simulation itself must be performed quickly, because the research process may require a large number of model experiments. Third, an accurate (detailed) description of a real system requires that the modeling platform provide

flexibility in setting, configuring, and possibility to add new model parameters without rebuilding the platform implementation. Finally, the simulation platform must be scalable and support the ability to simulate tens of thousands of resources and the same number of computational jobs.

Let us formulate the basic requirements for the simulation platform:

– the possibility to model the basic elements (storage resources, users, brokers, networks);
– the modeling speed that should significantly exceed the real system speed;
– the possibility to obtain modeling statistics for individual elements and for the operation of the model as a whole;
– the comparability of the modeling results with the real situation;
– the possibility to simulate equipment malfunctions.

The requirements depend on the problem to be solved; this paper examines the problem of scheduling computational jobs for the GDN resources, therefore, as special requirements.

In addition to the basic ones, we will highlight special requirements that are important when studying the task of scheduling jobs for GDN resources:

– the possibility to explore auction scheduling methods;
– the possibility to prioritize jobs;
– the possibility to simulate the real flow of jobs (real system log);
– the possibility to simulate background network traffic;
– the possibility to simulate the background load of computing resources (since non-expropriation resources are being studied).

There are two classes of modeling platforms: emulators and simulators [2]. A simulator is software enabling to model a real system, displaying part of the real phenomena and the properties in a virtual environment. An emulator is a tool for reproducing the behavior of a device or a program in real time. In other words, an emulator replaces the properties and functions of an original, doing real work, and a simulator only simulates these properties and functions without actually functioning. The best-known systems emulators are MicroGrid and Grid eXplorer. The papers [3, 4] note that emulators have less flexibility in comparing scheduling algorithms. This is primarily due to the lower speed of experiments in comparison with simulator systems, as well as the complexity of reconfiguring a network model.

There is a large number of different simulation systems for simulating a distributed network. Simulators are developed using different programming languages, have different architectures and assume different mechanisms for describing and working with a model. This means that a network model prepared in one simulator cannot be converted to a model in another simulator. In other words, there is no common standard for network modeling. Most simulator systems are highly specialized, i.e. they should only be used to investigate specific characteristics of the model, and that they include special tools that allow quick preparing of a model for investigating these characteristics. The correct choice of the system allows describing the studied GDN model fast and in detail, and concentrating on solving the AI specific problem. The paper [5] provides an extensive classification of simulator systems according to various criteria including: the presence of a user interface for the simulator, a programming language used to implement a model, an operating system that is required to run the simulator, a simulator software architecture, etc. Most often, the classification of simulator systems is used in accordance with the type of a distributed network [6].

Table 1

Research area and simulator examples

Network type	Field of study	Examples
Computing network	The efficiency of using various algorithms and strategies for scheduling user jobs for distributed network resources	• GridSim [7] • SimGrid [8]
Data network	The efficiency of various strategies for replication and data retrieval in the data network	• OptorSim • Monarc • ChicSim • GridNet
Service oriented network	The effectiveness of cloud services	• CloudGrid [9] • GreenCloud 10]
Communication network	The efficiency of algorithms for forwarding network protocol packets. The efficiency of routing and multicast over wired and wireless networks	• NS 2 • DaSSF • OMNeT ++ • OPNET

Table 2

Instruments and investigated characteristics

Network type	Key tools of modeling	The investigated characteristics of the model
Computing network	• basic scheduling algorithms • auction methods • topology toolkit • policies and resource allocation of compute nodes	• utilization of network resources • average job execution time • number of unfulfilled jobs
Data network	• data replication protocols • strategies and algorithms for finding replica locations	• data access time • disposal of storage • network utilization
Service oriented network	• tools to simulate starting and stopping virtual network resources	no data
Communication network	• Basic routing algorithms (DropTail, RED, etc.) • base of network protocols • databases of models of network devices provided by key manufacturers	no data

This paper presents the choice of a modeling platform for studying various auction models and their parameters, therefore, in the future, we will consider the first class of simulators – computer network simulators. As already noted, the most well-known class simulation systems are GridSim and SimGrid.

There follows the description of the criteria for the comparison.

Table 3

Comparison of simulators

Simulator	Year	Basis	Programming language	Research area
GridSim	2002	library SimJava2	Java	Economic scheduling methods [5]
S imGrid	1999	Designed from the ground up	C	Working with custom jobs such as DAG (Directed acyclic graph) and MPI (Message Passing Interface).

Both simulators are software libraries. The network model is created using a programming language. The comparison is be based on the following criteria.

1. Modeling heterogeneous network resources;
2. Modeling a heterogeneous communication environment with a complex topology;
3. Modeling different resource allocation policies. RR or BackFill can be used for uniprocessor it is FIFO, for multiprocessors (cluster systems).
4. Modeling auction mechanisms for scheduling computational tasks for distributed resources. The simulator must provide the appropriate abstractions (resource broker, auctioneer, bid, etc.) to implement auction methods;
5. Adding user-developed algorithms for scheduling jobs;
6. Simulation of load based on the logs, workload from supercomputers;
7. Modeling network resource failures [11];
8. Modeling background network traffic based on probabilistic distribution.

The disadvantage of both simulators is the lack of a graphical interface, which makes it possible to track the process of the scheduling algorithms. For a graphical presentation of the results, all information can be output to a file after simulating using the gnuplot utility.

Table 4 shows from that the GridSim simulator meets all the requirements most fully. It is important to note that this simulator was used to simulate the Nimrod\G resource broker [12].

Modeling failures of distributed network resources in the GridSim simulator

The GridSim simulator has the functionality required to simulate computational failures. At the same time, it is possible to simulate the failure of a part of the SuperComputing Unit (CU) computing resources and the failure of the entire CU. The failure detection mechanism implemented in GridSim is based on sending periodic signals

to the CU. Signals are sent both by the metascheduler in order to form a list of available computing resources, and by users in order to control job execution (Fig. 1).

Table 4

Comparison of simulator functionality

Comparison criteria	GridSIm	SimGrid
Modeling heterogeneous resources	+	+
Modeling communication heterogeneous environment	+	-
Modeling different resource allocation policies	There is Already implemented: FCFS, RR and Backfil varieties l	There is order of arrival (FIFO) and its variations (FRFO)
Modeling auction mechanisms	There are basic abstractions	no
Adding custom algorithms	possible	
Modeling load baseon logs	possible	possible
Network failure simulation	possible	no data
Modeling background network traffic	yes	no data

Let us consider a job-scheduling scenario implemented in GridSim.

1. The user contacts the metascheduler to obtain the CU identifier with the available amount of computing resources required to process the user task.

2. Having received the CU identifier, the user places the job in the local job management system (LJMS) of this CU.

3. LJMS allocates for a job computational resources from among those available in accordance with t implemented job-scheduling algorithm.

4. The user controls task execution periodically referring to the LJMS of the CU.

Let us consider the algorithms for the LJMS implemented in GridSim.

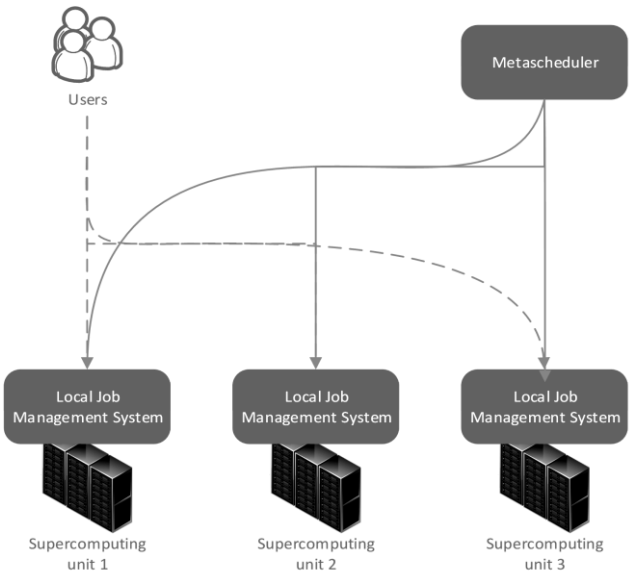

Fig. 1. Failure detection mechanism in distributed network resources implemented in GridSim. Dashed lines shows how users control the execution of the job by periodically polling the local job control system of the CU, the metascheduler generates a list of available resources also periodically polling the local job control system

– a scheduling algorithm in order of receipt (FIFO);

– a cyclic scheduling algorithm (RR, Round Robin).

The main difference between the presented scheduling algorithms for LJMS is processing of emerging failures on CU resources. In the case of resource failure, the jobs allocated by the FIFO scheduling algorithm immediately terminate abnormally and receive the corresponding error code. The RR cyclic scheduling algorithm reallocates jobs whose resources are faulty to other available resources within the same CU. In other words, unlike the FIFO algorithm, the cyclical scheduling algorithm does not rigidly fix computational resources for a specific task. If there are no available resources, then the users of these tasks will receive the corresponding error codes as in the previous case.

Both LJMS scheduling algorithms will work in the same way in the event of a malfunction of all the resources of the CU, namely, all tasks processed in this CU will terminate abnormally with the corresponding error code. Note that the realistic behavior in GDN would be not to assign an error code to an abnormally terminated job, since the entire CU is out of order and is inaccessible. However, in GridSim, the absence of a job completion code will lead to endless user waiting for the job to complete. Let us describe the main parameters of the considered model with failures:

Job parameters:
- the number of jobs: 50;
- the duration of each job (million instructions): 42,000,000;
- the volume of initial data/results (bytes): 100000.
- each job requires 2 CUs to complete.

Computing resource parameters:
- the number of CUs: 3;
- the number of computing modules (CM) in each CU: 10;
- the number of processors in each CU: 2;
- performance of each processor (millions of instructions per second): 50,000;
- a cyclic scheduling algorithm is selected as the LJMS algorithm;

The main characteristics of failure modeling are the number of VMs on which the failure occurred, the failure time and duration. In our experiment, all these parameters were set by a hyperexponential distribution with the following characteristics (mean, standard deviation, flow):
- the number of failed VMs (number of VMs/2, number of VMs, 4);
- the failure time (25, 100, 4);
- the failure duration (20, 25, 4).

Test results are displayed in text format and can be saved to a file. Table 5 shows just a part of the output of the simulation results, since the output of all simulation statistics takes up a lot of space.

Table 5

The results of modeling the fault tolerance of the system

Job ID	Job completion status	ID of the resource that ran the job	Cost	Job execution time	Time to receive the job result
0	Success	85	2571.87	857.29	862.04
4	Success	40	2572.47	857.49	862.42
1	Success	85	2572.32	857.44	864.81
2	Success	90	2574.00	858.00	874.86
3	Success	90	2572.65	857.55	880.7

Modeling a system for scheduling user jobs for geographically distributed resources of a distributed network

In addition to modeling failures, the GridSim simulator has functionality that allows exploring the work of different algorithms for scheduling user jobs. The toolkit available in the simulator allows flexible describing of the computational and communication resources that are a part of the studied GDN model. For example, different computing resources can have different cost of use, performance, architecture, operating system, resource allocation policy: total time or space, time zone, etc. Each simulated resource must be connected to a router.

Users who submit the jobs on computing resources can be simulated with different application or quality of service requirements. These requirements include the network data transfer rate (connection speed), the maximum time allowed for a job to start, the time lag between job submissions, and a scheduling strategy such as optimizing costs and/or time to complete jobs. Economic requirements (deadline and budget) can be set for each user, which restricts the job execution. For example, a user may be simulated willing to spend as much money as required, or a user willing to spend the exact amount.

The simulated jobs, in turn, are characterized by the duration of execution (in millions of instructions), the amount of input and output data (in bytes).

In the studied example, the authors model a network with homogeneous resources, all of them have the same performance. Two supercomputer centers (SCC) are modeled. The supercomputer center includes 3 computing units, each has a configuration of 100 processors. Each processor in the computational unit is rated at 1000 MIPS. The following numbers of users were simulated: 100, 200, and 300. Each user created 100 jobs, each job required one processor to execute. The size of the each job initial data is 600 MB, the output data is 300 MB, the processing time varied during the experiment.

The Dutch auction was used as an algorithm for scheduling computational jobs for remote resources. The auction scheduling algorithm assumes competition of participants for the right to use resources; during competition, the cost of using resources is determined. The Dutch auction is an auction for a decrease, the auctioneer starts the auction with a knowingly high price, after which he begins to lower the bid. When only one member is ready to buy an item, the item is sold. Our experiment had the established characteristics of the Dutch auction: the maximum round duration was 1 minute, and the maximum number of rounds was 10.

The parameters of the computational resources and the parameters of the jobs of the model under study were set as in the previous experiment. No background network traffic was modeled. The background loading of the resources of the computing unit was not modeled. As in the previous case, the simulation result was presented in text format, due to its cumbersomeness. Figure 2 shows only a part of the simulation results.

```
Resource_2 bidding for auction 0 round 1 and price 4.6470160484313965
Resource_1 bidding for auction 0 round 1 and price 4.9272847175598145
Resource_3 bidding for auction 0 round 1 and price 8.039090156555176
Auction results
Winner ID: 9
Price for job execution 0: 4.6470160484313965
```

Fig. 2. The result of the job scheduling system modeling with the Dutch auction scheduling algorithm

About the boundary of simulation platforms

We can formulate the following limitations of the simulation platforms from the conducted research:

1. High entry level. On the one hand, simulators are versatile, since they allow supplementing the model with custom parameters. On the other hand, the researcher needs to spend a lot of time to understand the simulator structure and must have knowledge and programming skills using specific programming languages. In addition to knowing the programming language, the user must understand how to work correctly with platform abstractions. All this significantly increases the entry threshold. It takes a long time from the moment an idea appears to the first model close to the real one.
2. No model reuse. The user must implement most of the scheduling algorithms on his own using a programming language; this is a source of errors. It would simplify a single model space in which users could place their models and implemented algorithms. It is reasonable to assume that the ideas of the open source community would improve the quality of the models. The examples of such platforms are: Vagrant Cloud, Docker Hub, GitHub. These examples demonstrate users' ability to use or improve the ideas of other users, which significantly saves time and allows getting new ideas (a user does not need to spend time studying a large amount of documentation, but simply finds a similar problem and tries to use the idea).
3. Installation and initial configuration complexity. To work with the simulator, it is necessary to install and configure compilers (*gcc* in our case) or virtual machines (*jvm* in the example). It also raises the entry barrier. Using modeling tools in the cloud would also make them easier to use.
4. Large volume of resources. The performed experiment requires a large volume of computing resources to simulate a real network.
5. Ease of use. Modeling tools have no familiar interactivity; most of them do not have a graphical interface. The user can visualize the results only after finishing modeling using third-party visualization utilities. While in some studies, for example, comparing different scheduling algorithms, visualization of the algorithm is of interest.
6. Lack of examples. The documentation of most simulators is compiled according to the rules for describing a programming language library. However, it does not contain a large number of examples.
7. Complexity of choosing a simulator system. Before starting to use the simulator, the user should spend a lot of time studying and choosing the simulator from a large number of existing solutions. The decision choice can be based only on the analysis of scientific articles, since there is no sufficient information on the site.

The listed disadvantages have a negative impact on such properties of the modeling platform as usability and ease of model configuration. The low-level implementation of the scheduling algorithms is a source of errors, and the lack of open space for the exchange of models in the scientific environment limits the applicability of modeling platforms.

At the same time, the functionality available in GridSim for modeling and detecting failures that have occurred makes it possible to use this simulator to study distributed system reliability. The set of standard scheduling algorithms implemented in GridSim, can be supplemented with other scheduling algorithms if necessary.

Due to the limitations of simulation platforms, an alternative approach is also being developed to determine the numerical parameters of program execution.

Determining the numerical reliability parameters

We consider the problem of determining the optimal choice of the sizes of subtasks when executing programs with parallelization according to data with a known a priori equipment reliability.

The job is divided into M subtasks, which are executed on N computing nodes of one or more computing systems. Each node receives a data "chunk", processes it and sends the results.

The nodes are combined into clusters. A set of clusters within a single computing center forms a computing facility. Computing installations in different computing centers are aggregated into a geographically distributed network.

A computing cluster contains N identical nodes (U_1, ... U_N).

The probability of the job completion at the computing node i will be denoted as P (U_i). The time of organizing computations on the computational node of the cluster will be designated as TO. The computation organization time consists of data transmission to the $T1$ node, the TS program start time and the data transmission time from the $T2$ node.

The execution time of the entire job on the cluster is designated as TP. A problem of dimension S is divided into M subtasks, each of which is assigned to a node.

Subtask execution time is

$$t = \frac{TP}{M} + T1 + T2 + TS.$$

Then the job execution time on the cluster will be

$$T = \left[M / N\right] \cdot \left(\frac{TP}{M} + T1 + T2 + TS \right).$$

Probability of job completion on the computing node is denoted by P. In case if the subtask at the node is not completed (with probability $1 - P$), a restart is performed.

An incomplete subtask takes the following time

$$tf = \frac{TP}{N} + T1 + TS + TO,$$

where TO is the time it takes to detect a failure while executing a subtask. If $M > N$, then the execution occurs in tiers according to the "master-slave" scheme. Tier execution time in case of unfinished jobs is the following:

$$T = \frac{TP}{\min(M, N)} + T1 + TS + \max(T1, TO).$$

The computing facility consists of L homogeneous clusters (C_1, ..., C_L). Each cluster C_i has performance E (C_i) when performing this job. The average cluster performance is:

$$E_m = \frac{\sum_{k=1}^{L} E(C_i)}{L}.$$

Normalized cluster performance is:

$$\tilde{E}(C_i) = \frac{E(C_i)}{E_m}.$$

When splitting jobs for M subtasks, execution also takes place according to the scheme "master-slave". In this case, the number of tiers l is first selected, and then the problem is divided into l subtasks J_1, ... J_l. Each J_i subtask is divided into subtasks in proportion to the cluster performance so that the execution on each cluster is the same:

$$Size_i = \tilde{E}(C_i) = \frac{S}{l \cdot L}.$$

Further execution proceeds in the same way as on the cluster nodes. If the subtask ends abnormally (with probability $P(C_i)$), the size of the remaining job is increased by $Size_i$ and there is a redistribution already only between the remaining clusters.

Let us consider a distributed computer network consisting of K computing devices (S_1, ... S_k).

In this case, the execution is similar, except that the times $T1$ and $T2$ significantly depend on the CU, it should be taken into account when choosing the size of the subtask.

There is a developed simulation program to determine numerical estimates for the optimal choice of the subtask sizes at the level of parallel program execution for the equipment used with known a priori reliability. The program determines the optimal number of subtasks depending on the expected number of failures per unit interval, the maximum number of nodes used, and the overhead of organizing the execution of subtasks. The criterion for optimality is the minimum execution time.

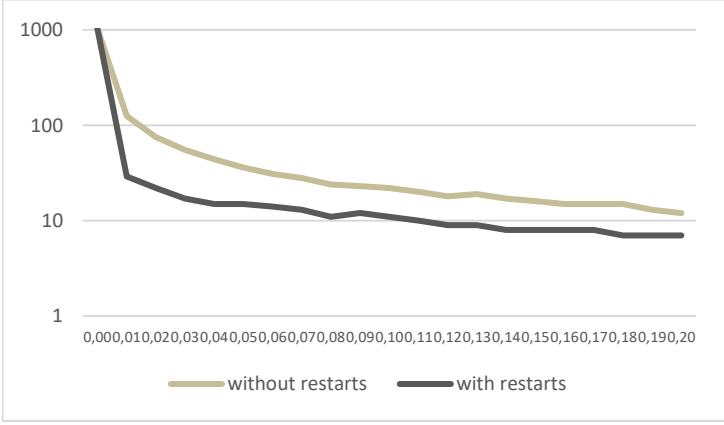

Fig. 3. The dependence of the number of jobs on waiting for failure

We consider two models. In the failure event all jobs are restarted or only those that have not completed. Modeling has shown that in the case of absolute reliability, if it is necessary to use the number of subtasks with the maximum acceleration factor (which is obvious), then with an increase in the expected number of failures, the optimal number of jobs decreases (Figure 3). So, with the optimal number of subtasks equal to 1024 with absolute reliability, while waiting for 0.2 failure on a unit interval, the number of subtasks decreases to 12 when restarting incomplete subtasks and to 7 when starting all subtasks. A simulation program will be expanded for jobs running time evaluation on CUs and a GDNs.

Conclusion

There are the following limitations of simulation platforms:
- high entry level,
- impossibility to reuse a model,
- complexity of the modeling platform installation,
- the need for a large number of resources,
- ease of use,
- insufficient examples,
- difficult choice of a simulator system.

The functionality of modeling and detecting failures in GridSim makes it possible to use this simulator to study the reliability of a distributed system.

The authors are developing an alternative approach to studying the reliability of a whole distributed system as well as its parts. The approach is to simulate job execution on a geographically distributed system with limited reliability.

Acknowledgements: The reported study was funded by state assignment, project 0580-2021-0014 and RFBR, projects no. 19-07-01088 and no. 18-29-03236. The research involved using the Supercomputer MVS-10P installed in JSCC RAS.

References

1. Legrand A.R., Casanova H., Marchal L. Scheduling distributed applications: The SimGrid simulation framework. *Proc. IEEE/ACM Intern. Symposium CCGrid*, 2003, pp. 138–145. DOI: 10.1109/CCGRID.2003.1199362.
2. Korsukov A.S. Automation of preparation and implementation of simulation modeling in an integrated cluster system. *Modern technologies. System analysis. Modeling*, 2012, no. 3, pp. 98–103 (in Russ.).
3. Dias de Assunção M., Streitberger W., Eymann T., Buyya R. Enabling the simulation of service-oriented computing and provisioning policies for autonomic utility grids. *Proc. GECON. Lecture Notes in Computer Science*, 2007, vol. 4685, pp. 136–149. DOI: 10.1007/978-3-540-74430-6_11.
4. Jarvis S.A., Spooner D.P., Mudalige G.R., Foley B.P., Cao J. and Nudd G.R., *Performance Evaluation of Parallel and Distributed Systems*, vol. 1, chap. Performance Prediction Techniques for Large-scale Distributed Environments. Ould-Khaoua M. and Min G. (Eds.), Nova Sci., 2005, pp. 269-288.
5. Prajapati H.B., Shah V.A. Analysis perspective views of grid simulation tools. *J. of Grid Computing*, 2015, vol. 13, no. 2, pp. 177–213. DOI: 10.1007/s10723-015-9328-9.
6. Argungu S.M., Arif S., Hasbullah O.M. Compute and data grids simulation tools: A comparative analysis. In: *Emerging Trends in Intelligent Computing and Informatics* by Saeed F., Mohammed F., Gazem N., 2020, pp. 533–544. DOI: 10.1007/978-3-030-33582-3_50.
7. *GridSim: A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing*. Available at: http://www.buyya.com/gridsim/ (accessed April 22, 2021).
8. *Simulation of Distributed Computer Systems*. Available at: https://simgrid.org/ (accessed April 22, 2021).
9. *CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*. Available at: http://www.cloudbus.org/cloudsim/ (accessed April 22, 2021).
10. *Greencloud - The Green Cloud Simulator*. Available at: https://greencloud.gforge.uni.lu/ (accessed April 22, 2021).
11. Caminero A., Sulistio A., Caminero B., Carrion C., Buyya R. Extending GridSim with an architecture for failure detection. *Proc. Intern. Conf. Parallel and Distributed Systems*, 2007, pp. 1–8. DOI: 10.1109/ICPADS.2007.4447756.
12. 12. Buyya R., Abramson D., Giddy J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. *Proc. IV Intern. Conf. and Exhibition on HPC in Asia-Pacific Region*, 2000, vol. 1, pp. 283–289. DOI: 10.1109/HPC.2000.846563.

УДК 004.9
DOI: 10.15827/2311-6749.21.2.2

Моделирование надежности территориально распределенных вычислительных систем

А.И. Тихомиров [1], *к.т.н., tema4277@rambler.ru*
А.В. Баранов [1], *к.т.н., Anton.Baranov@jscc.ru*
П.Н. Телегин [1], *к.т.н., pnt@jscc.ru*

[1] *Межведомственный суперкомпьютерный центр Российской академии наук, Москва,119334, Россия*

В статье представлен обзор актуальных платформ моделирования распределенных вычислительных сетей. Осуществлен выбор платформы моделирования, обладающий необходимым функционалом для исследования задачи отказоустойчивости системы управления заданиями распределенной сети, а также исследования задачи планирования пользовательских заданий на территориально удаленные ресурсы распределенной сети. Исследован реализованный в платформе моделирования GridSim функционал для построения модели распределенной сети. Реализованы две модели: для исследования задачи отказоустойчивости системы управления заданиями распределенной сети, для исследования аукционных методов планирования пользовательских заданий на территориально удаленные ресурсы распределенной сети. Сформулированы достоинства недостатки моделирования распределенной сети. Предложен альтернативный подход к моделированию, требующий меньших вычислительных ресурсов.

Ключевые слова: *платформа моделирования, набор инструментов, моделирование распределённой вычислительной сети, gridsim.*

Благодарности. *Работа выполнена в рамках государственного задания (тема 0580-2021-0014) и при финансовой поддержке РФФИ, проекты № 19-07-01088 и 18-29-03236. В исследованиях использовался суперкомпьютер МВС-10П, установленный в МСЦ РАН.*

Литература

1. Legrand A.R., Casanova H., Marchal L. Scheduling distributed applications: The SimGrid simulation framework. Proc. IEEE/ACM Intern. Symposium CCGrid, 2003, pp. 138–145. DOI: 10.1109/CCGRID.2003.1199362.
2. Корсуков А.С. Автоматизация подготовки и проведения имитационного моделирования в интегрированной кластерной системе // Современные технологии. Системный анализ. Моделирование. 2012. № 3. С. 98–103.
3. Dias de Assunção M., Streitberger W., Eymann T., Buyya R. Enabling the simulation of service-oriented computing and provisioning policies for autonomic utility grids. Proc. GECON. Lecture Notes in Computer Science, 2007, vol. 4685, pp. 136–149. DOI: 10.1007/978-3-540-74430-6_11.
4. Jarvis S.A., Spooner D.P., Mudalige G.R., Foley B.P., Cao J. and Nudd G.R., Performance Evaluation of Parallel and Distributed Systems, vol. 1, chap. Performance Prediction Techniques for Large-scale Distributed Environments. Ould-Khaoua M. and Min G. (Eds.), Nova Sci., 2005, pp. 269-288.
5. Prajapati H.B., Shah V.A. Analysis perspective views of grid simulation tools. J. of Grid Computing, 2015, vol. 13, no. 2, pp. 177–213. DOI: 10.1007/s10723-015-9328-9.
6. Argungu S.M., Arif S., Hasbullah O.M. Compute and data grids simulation tools: A comparative analysis. In: Emerging Trends in Intelligent Computing and Informatics by Saeed F., Mohammed F., Gazem N., 2020, pp. 533–544. DOI: 10.1007/978-3-030-33582-3_50.
7. GridSim: A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing. URL: http://www.buyya.com/gridsim/ (дата обращения: 22.04.2021).
8. Simulation of Distributed Computer Systems. URL: https://simgrid.org/ (дата обращения: 22.04.2021).
9. CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. URL: http://www.cloudbus.org/cloudsim/ (дата обращения: 22.04.2021).
10. Greencloud – The Green Cloud Simulator. URL: https://greencloud.gforge.uni.lu/ (дата обращения: 22.04.2021).
11. Caminero A., Sulistio A., Caminero B., Carrion C., Buyya R. Extending GridSim with an architecture for failure detection. Proc. Intern. Conf. Parallel and Distributed Systems, 2007, pp. 1–8. DOI: 10.1109/ICPADS.2007.4447756.
12. Buyya R., Abramson D., Giddy J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. Proc. IV Intern. Conf. and Exhibition on HPC in Asia-Pacific Region, 2000, vol. 1, pp. 283–289. DOI: 10.1109/HPC.2000.846563.