Запропоновано інформаційну технологію розподілу завдань для GRID-систем, яка заснована на використанні імітаційного середовища моделювання GRASS. GRASS відтворює процес функціонування в часі елементарних подій, які протікають в GRID-системі із збереженням логіки їх взаємодії. Дане рішення дозволяє проводити обчислювальні експерименти, що реалізують різні методи розподілу, з подальшим вибором найбільш ефективного вирішення на основі збору, аналізу та інтерпретації результатів моделювання

Ключові слова: розподілені системи, GRID-система, планувальник завдань (брокер), інформаційна технологія, середа GRASS, обчислювальні ресурси, політики розподілу

Предложена информационная технология распределения заданий для GRID-систем, основанная на использовании имитационной среды моделирования GRASS. GRASS воспроизводит процесс функционирования во времени элементарных событий, которые протекают в GRID-системе с сохранением логики их взаимодействия. Данное решение позволяет проводить вычислительные эксперименты, реализующие разные методы распределения, с последующим выбором наиболее эффективного решения на основе сбора, анализа и интерпретации результатов моделирования

Ключевые слова: распределенные системы, GRID-система, планировщик заданий (брокер), информационная технология, среда GRASS, вычислительные ресурсы, политики распределения

UDC 004.051: 004.75
DOI: 10.15587/1729-4061.2016.71892

DEVELOPMENT OF INFORMATION TECHNOLOGY OF TASKS DISTRIBUTION FOR GRID-SYSTEMS USING THE GRASS SIMULATION ENVIRONMENT

T. Filimonchuk
Assistant*
E-mail: tetiana.filimonchuk@nure.ua
M. Volk
PhD, Associate Professor*
E-mail: maksym.volk@nure.ua
I. Ruban
Doctor of Engineering Science, Professor, Head of Department*
E-mail: ihor.ruban@nure.ua
V. Tkachov
PhD, Assistant*
E-mail: vitalii.tkachov@nure.ua
*Department of Electronic Computers
Kharkiv National University of Radioelectronics
Nauka ave., 14, Kharkiv, Ukraine, 61166

1. Introduction

GRID is a geographically distributed infrastructure, which is created on the basis of a set of heterogeneous network resources (cluster, server, separate PCs) and is used to solving scientific tasks on large computing powers. GRID is a coordinated, standardized and open environment that enables performing optimal distribution of computing resources to run the tasks, accessing at the entrance of the system.

GRID-systems are divided into three types: voluntary, scientific and commercial. Each of these types has a number of advantages and disadvantages, is characterized by a large number of decisions, implementations, forms of organization of calculations. One of the relevant objectives is the development of application software, the main function of which is providing the user of GRID-systems with convenient interface between the application and the required computing resources.

2. Analysis of scientific literature and the problem statement

There are many methods and algorithms of distribution of tasks on computing resources. A huge niche in this regard is given to programs that carry out the distribution of tasks for resources - Task Scheduler (brokers). The main objective of the broker is the construction of the distribution plan that meets the requirements of the task suppliers. When building a distribution plan, an optimization of values of the parameters of the objective function is carried out, through the use of which, a reduction in run-time of accessed tasks proceeds in the system, which leads to efficient use of computing resources.

Often, when distributing the tasks, a first suitable resource without any optimization is selected [1–4]. Other algorithms do not take into account the peculiarities of the environment with inalienable resources, i.e. a monitoring of the computing resource utilization dynamics is not proceeded, the task supplier's competition is not taken into account [5, 6]. In [6], the criteria for the selection of the slots on the basis of utility functions are provided, which specifies a task supplier. However, even here, the optimization is performed at the level of selection of the best available computational resources. Many algorithms are not focused on advanced reservation, and are based on the priority of the tasks in a queue [7, 8]. However, the use of advanced reservation [1–3, 9, 10] gives a gain in a performance time of the task pool, which in turn, can reduce the downtime of computational resources in the system.

Many approaches of distribution of tasks on computing resources [10, 11] are guided by the use of models of integer linear or mixed programming. In this case, the task suppliers can set preferred time frames of use of computing resources. In [12] it is proposed to use a combination of integer linear programming model with a genetic algorithm, which enables composing the distribution plan and taking into account the cost of computing resources for specific groups of suppliers. In [13] a model of distribution of tasks is proposed, which enables re-planning "in the air" by means of information about the actual dynamics of the system load.

Planning methods described in [4, 9–13] use a number of criteria (the value of the use of resources, the level of load of computing resources, the use of nearby resources for related objectives in the task). In the studies [14–17] the methods of planning, which focus on the preferences of task suppliers, administrators of virtual organizations or suppliers of computing resources are suggested to be used.

An optimization criterion is entered into a resource request format which takes into account the interests of users. This criterion is used for the search of alternatives for performance of tasks, i. e., the owners of computing resources get the possibility to manage the workload of computer systems. The policy of partition of tasks into separate subtasks is also implemented, which can be run on diverse computing resources that enables improving the efficiency of the system load.

In [18] the selection of computing resources for the tasks, entering into the system, is performed by means of logical-probabilistic algorithm. The proposed algorithm is focused on multi-level planning of tasks for specified quality criteria (Cost, lead-time, reliability index). Planning is carried out in four steps based on the mechanism of regulation of demand and supply of resources.

In [19] a classification of incoming tasks is proposed, which can be used as a superstructure for the task scheduler (broker). Using the proposed superstructure increases the efficiency of the use of computing resources by optimizing the distribution of computing resources.

There are also those methods [14, 15], in which there is the possibility of entering a predetermined selection criteria (E. g., via JSDL language), but they are characterized by a linear time complexity, i. e., they depend on the number of computational resources, which are available at the current planning time.

The conducted analysis shows that the disadvantage of existing GRID-systems is the use of a single broker, which is oriented to the certain class of objectives and at distribution of incoming tasks, it uses one distribution policy. However the GRID is primarily a heterogeneous system [20] and, in addition to that, the tasks, accessing for performing, may be of a heterogeneous nature. Most of today's planners do not take into account the cost of computing resources, which also leads to inefficient use of resources.

3. The purpose and objectives of the study

The purpose of this work is the developing of the information technology for distribution of tasks for computing resources in a heterogeneous GRID-system.

To achieve this goal it is necessary to solve the following tasks:

– to develop a mathematical model of task representations and computational resources in GRID-systems;

– to develop a method of selection of the best distribution option;

– to carry out a series of experiments on the distribution of tasks on the basis of the proposed method in the GRASS simulation modeling environment.

4. The use of simulation modeling for the distribution of tasks in GRID-systems

Let us justify the expediency of the proposed GRASS simulation environment (GRID Advanced Simulation System) for the research of application of different task planning strategies.

There are many sets of modeling environments of the GRID-systems. The most popular of them are SimGrid, GridSim, MicroGrid, ChicSim, OptSim, Alea [21, 22]. The comparative analysis shows that most of them are characterized by specialization, limitations of structures, absence of accessible and open versions. Also to work with such systems the knowledge of specific programming languages is required.

Creating a GRID-system requires investments, which are not always available, so a cheaper option is the use of simulation modeling. Simulation modeling is based on the building of a mathematical model, which can be used to research the properties of the real system.

GRASS system [23] is a computer model of GRID-system which enables developing a framework of distributed computing systems taking into account the problems that are solved by it and to explore its properties. This model enables:

– to reproduce the process of functioning in time of elementary events occurring in the system with the preservation of their interaction logic;

– to produce a series of numerical experiments, which will enable collecting, analyzing the modeling results and a comparison of the received data with the actual behavior of the object.

The GRID-system tasks do not run on their arrival, but at a certain time of the day. Therefore, we can use the task and resource pool from the real GRID-system and with the help of GRASS environment to simulate and get the best distribution plan, which can be offered to a real system. The received best plan minimizes not only the final performing time of pool tasks, but allows to increase of efficiency of the use of computing resources of the GRID system. However, in this case, a problem arises that a GRASS environment simulates a work of a real GRID-system and its result can be obtained only after a certain time, when the distribution plan may be no longer relevant. This problem can be solved in two ways:

– start of GRASS environment is carried out parallel on several processors (according to the number of present distribution policies in the environment);

– introduction of a temporary scaling for process modeling.

The first solution can reduce a modeling time, as the first received plan is the best solution in time for a given task pool. However, this solution is a long-term option, it cannot be always possible to apply in practice. The second solution, based on the introduction of a temporary factor, enables

reducing of distribution time (transition from hours to seconds). After receiving the plans on all distribution methods, it is required to make the transition from seconds to hours and obtain a real modeling time on the basis of which to analyze and select the best distribution plan.

5. Technologies of tasks distribution using GRASS environment

GRID-system is a complex object, containing a set of comprising of interconnected heterogeneous resources, whose operating is subjected to a number of predetermined rules. One of the main tasks of this system is coordinating of resource distribution, for the solution of the provided tasks. The distribution model in GRID–system can be constructed on the basis of two sets: a set computing resources R and a set of tasks Z, and also a distribution policy q, i. e., $G = \{R, Z, q\}$.

Tasks, accessing the GRID-system, form a stream $\{Z_i, i=1, 2,...,M\}$, where I is the serial task number, and M is the number of tasks. Each task includes a number of parameters required for its launching in the environment (1):

$$Z_i = \{ar_i^z, os_i^z, pc_i^z, ps_i^z, ms_i^z, dc_i^z, pr_i^z\}, \ \forall i = 1...M, \quad (1)$$

where ar_i is the architecture; os_i is the operating system; pc_i is the processor count; ps_i is the processor speed; ms_i is the memory size; dc_i is the disk capacity; pr_i is the priority.

Any task is a package of objectives $\{z, a \ 1,2,...,A\}$ $(z_a \in Z_i)$, combined by a specific theme. Each task is a separate performing program. The task for its performing can be started when all requested resources will be selected for all tasks.

Resources also form the set $\{R_j, j = 1,2,...,N\}$, where j is the number of computing resource, and N is the number of resources in GRID-system. Any computing resource, accessing GRID-system is described with a number of features which are represented by a tuple (2):

$$R_j = \{ar_j^r, os_j^r, pc_j^r, ps_j^r, ms_j^r, dc_j^r\}, \quad \forall j = 1...N, \quad (2)$$

In different distribution systems the input tuples (1) and (2) may have insignificant differences – due to the fact that the systems develop and during the work with them, arises the need for introduction of addictions, related to requirements of task suppliers.

An important place in the GRID-systems is assigned to task schedulers (brokers), whose responsibility is to establish a schedule of use of computing resources. There are currently more than twenty well known schedulers for GRID-systems: Portable Batch System (PBS), Sun Grid Engine (SGE), TORQUE, Condor, LoadLever, MAUI Scheduler, etc. The listed above schedulers do not provide the user with a unique efficient task distribution method. With their help it is only possible to use one simple algorithm, which is started on the condition that all the tasks and resources of the system are given to an advanced predetermined general description.

Currently, there is no universal task scheduler, as the tasks in GRID-systems are heterogeneous. Therefore, the additional options should be taken into account at distribution, which will enable increasing the efficiency of use of computing resources of the system. For example, the analysis of objectives in the task in terms of their connectivity will enable to reduce the performing time of the task pool in the system by eliminating the losses in time, which are caused by data exchange between separate objectives in the task.

Let us extend the model of GRID-system through the introduction of 1 and 2 parameters in the tuples, which will reduce the downtime of computing resources in GRID-system (3), (4):

$$Z_i = \{ar_i^z, os_i^z, pc_i^z, ps_i^z, ms_i^z, dc_i^z, pr_i^z, ca_i^z, rt_i^z\},$$

$$\forall i = 1...M, \quad (3)$$

$$R_j = \{ar_j^r, os_j^r, pc_j^r, ps_j^r, ms_j^r, dc_j^r, bw_j^r, d_j^r\}, \quad \forall j = 1...N, \quad (4)$$

where ca_i (coefficient of association) is the coefficient of connectivity of objectives in the task; rt_i (run time) is the time of task performing (time of use of the resource); bw_j (bandwidth) is the total bandwidth (from the broker to the resource) including the network condition at the current time; d (delay) is the total packet transmission delay time based on network condition at the current time.

Let us also introduce a set of distribution methods in the model instead of a single distribution policy.

$$Q_k = \{mn_k, lp_k\}, \ \forall k = 1...k, \quad (5)$$

where mn is the application distribution algorithm on computing resources; lp is the list of input parameters taken into account at distribution.

As mentioned above, the disadvantage of the GRID-system is the use of a single broker, which is focused on a certain class of objectives. The proposed GRASS environment enables operating several distribution algorithms at the moment: First-Come First-Served (FCFS), Last In First Out (LIFO), Highest Priority First (HPF), a linear programming method (Simplex), distribution method on free resource, (Smart), a backfill method (Backfill), and a task distribution method on the computing resources based on network traffic (Backfill mod). Backfill mod is an upgrade of a backfill method, which, unlike the existing one, enables performing distribution of tasks, accessing GRID-system depending on objective associations in each of the tasks.

To select an effective distribution plan, a selection method of the best distribution plan has been developed, which was implemented in GRASS modeling environment. It is required:

– to form pools of tasks and resources for different classes of tasks (either to form in GRASS environment using specified generators, either use the pools of tasks of the real GRID-systems;

– to select a policy (method) of the distribution;

– to run the simulation in the GRASS environment;

– to analyze the log files for all of the policies (methods) of distribution;

– to select the best plan for the distribution on the basis of accepted rules and restrictions.

The proposed task distribution policies in GRASS environment for different input pools of tasks and resources can provide a gain in time and reduce a downtime of computing resources, so it is necessary to set accurate limits for the access.

Let us observe the operation of the information technology of task distribution [24] in a heterogeneous system, which

uses a simulation modeling environment GRASS (Fig. 1). The proposed technique involves the following stages.

Stage 1. Experiment parameters setting. To run an experiment it is required to set a configuration file plugins.xml, which describes interconnections between the module names in the system and the names of the files and libraries, as well as paths to configuration files [25]. A plugins.xml file allows setting parameters, passed to the modules at boot time, which can be used to set the operation mode or initializing of internal values: for tasks, computational resources and methods of distribution.

The result of this step is reading and decoding of configuration file (plugins.xml), loading the main system plugins to start a distribution process.

Stage 2. A distribution policy selection. The process of distribution of tasks is performed after the tasks_count parameter validation of the AlgorythmLoader plugin. As the task number in the queue exceeds the number, set in the tasks_count parameter, a call of distribution algorithm proceeds (5), which is defined in the DistributionAlgorythm parameter. According to the selected distribution method [26], a selection of resources for each task in the queue will be processed. A module of task distribution, which is a part of GRASS environment, contains a number of methods, which emulate different distribution algorithms, each of which uses its own set of parameters (3) and (4) for distribution. FCFS and LIFO methods do not use additional parameters, as they are the simplest policies, serving visual comparison with other methods.

Stage 3. Uploading of information about computing resources and tasks into GRASS modeling environment.

A task, accessing the system, is a packet of objectives, which is a separate executable program. The processor can execute only one task at each moment of time, and the task can be started for execution only in case of selection of computing resources for all objectives of the task. Any task, accessing into GRASS environment can be divided into two components: the characteristics (parameters) of the task and the task (as a .exe file, input data files, databases, etc.).

Tasks, accessed to the GRASS environment, form a flow, (link 1, Fig. 1) $\{Z_i, i=1,2,...,M\}$, where i is the task number, and M is the number of tasks. Having entered the system, a division of tasks on proposed above parameters is proceeded. At the initial stage the primarily value is a task information for the system (3), which operates the data, required for the search of computing resources, suitable for run [26]. The file (tasks_data.xml), containing this information, is set to advance predetermined format and enables selecting the certain resource to run in this system (Fig. 2).

```
<tasks>
    <task time="1000">
            <parameter cpu_arch="AMD"/>
            <parameter cpu_count="64"/>
            <parameter cpu_speed="1000"/>
            <parameter os="Windows XP"/>
            <parameter physical_memory="16"/>
            <parameter priority ="18"/>
            <parameter timeout="180000"/>
            <parameter hdd_size="16" />
            </task>
</tasks>
```

Fig. 2. A file fragment tasks_data.xml

GRASS system enables not only loading of advanced generated task file using the the SimpleTasksManager plugin, but also within the distribution process, to load task forming generator, using the SimpleTasksGenerator plugin from the tasks_model.xml file (Fig. 3).

Currently, the following generators are provided in GRASS environment:

– integers: (Uniform), (Constant), (Exponent) and (Normal);

– for lists: the sequential selection of one value from an advanced proposed list (ListLinear) and a random selection (ListRandom).

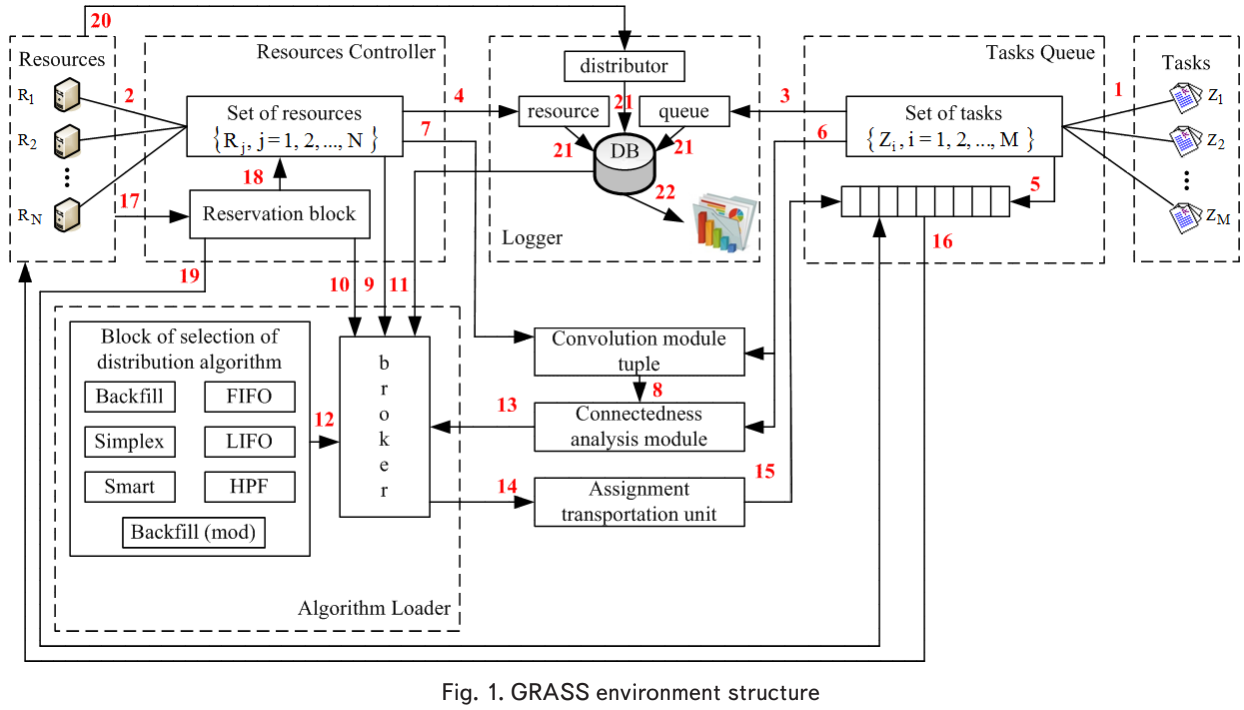

Fig. 1. GRASS environment structure

```
<delay generator="Uniform">
<parameter a="1500"/>
<parameter b="2300"/>
</delay>
<memory generator="Normal">
<double mean="1000000"/>
<double sigma="5000"/>
</memory>
<priority generator="Constant">
<double value="100"/>
</priority>
<time generator="Exponential">
<double gamma="0.5"/>
</time>
<OS generator="ListLinear">
<parameter values="stringlist">
<string value="Linux 2.6.2"/>
<string value="Linux 2.4.2"/>
<string value="Windows 98"/>
<string value="Windows 2000"/>
<string value="Windows XP"/>
<string value="Windows Vista"/>
<string value="Windows 7"/>
<string value="Solaris"/>
<string value="Mac Os X"/>
</parameter>
</OS>
```

Fig. 3. Fragment of file tasks_model.xml to generate a pool of tasks

Simultaneously with the tasks, an information about the available resources accesses the system (communication 2 in Fig. 1) $\{R_j, j=1,2,...,N\}$, where j is the order number of the resource, and N is the number of resources in GRID-system. Resources, as well as tasks, are also set to the specified format (resources_data.xml) (Fig. 4).

```
<resources>
    <resource name="name1" time="480">
            <parameter cpu_arch="AMD"/>
            <parameter cpu_count="8"/>
            <parameter cpu_speed="1000"/>
            <parameter os="Windows XP"/>
            <parameter physical_memory="16"/>
            <parameter hdd_size="64"/>
            <parameter free_cpu="8"/>
    </resource>
```

Fig. 4. Fragment of file resources data.xml

Just as in the task formation, we can use the SimpleResourcesManager and SimpleResourcesGenerator plugins. The first plug–in is used to download resources from the resources data.xml file, which was formed in advance, the second plugin will load the selected resource generator, which is specified in the configuration file model.xml. The generators of forming of computing resources are similar to generators of tasks (Fig. 3). Parallel to the work of this stage, a launch of the 7^{th} step is fulfilled, which is responsible for maintaining the log-files in the GRASS environment.

Stage 4. Formation of additional parameters for the most efficient distribution. All tasks, accessed to the system, are placed to the task queue (link 5, Fig. 1) and parallel to it, a transfer of information about each task is proceeded (a tuple 3) into a tuple convolution module and an association analysis module (link 6, Fig. 1). Tuple convolution module computes the generalized evaluation criteria for each task, which enables to manage the process of distribution of tasks on computing resources more efficiently and will show what part of the resource the task takes while performing [27]. At the same time, a transfer of information of resources, available in the system, is proceeded (link 7, Fig. 1). The result of the tuple convolution module is a list of resources, on which each task can be distributed.

Then information is received at the association analysis module, (link 8, Fig. 1), where the analysis of objectives in the task is performed. If objectives in the task have high connectivity (the exchange of information between tasks in the course of their implementation is required) or for this task a transfer of large amount of input and output data is required, a call of method checkQueueStrict() will be proceeded, which will compare the requirements of the task with available computing resources.

If computing resources for running of the current task are available in the system, it remains in the queue and receives a Waiting status, and an appropriate computing resources will be selected to suit it. If these resources are not found, the task gets a Cancelled status, after which it leaves the queue i.e., this status indicates that the task cannot be run in this configuration.

In this case, at distribution of objectives out of the task, the focus on selection of computing resources will be done in such a way, to reduce the time of data transferring for task objectives.

If objectives in the task are not interconnected, a method checkQueueBase() will be called in the system, which, similar to the method checkQueueStrict() will perform a selection of computing resources to distribute individual objectives out of the task to various computing resources and the task remains in the queue at Waiting status up to the time of its running on computing resource.

Stage 5. Task distribution module (broker) receives the information which enables to make decisions for distribution of tasks on computing resource or resources, performing on which will be the most optimal for it both in hardware characteristics, so in characteristics of performance. For distribution, the broker must receive the following information:

– information on the resources (4), which are currently present in the system (link 9, Fig. 1);

– information on the resources that are currently reserved (link 10, Fig. 1) and the time of their release;

– information from the database of the previous starting of the task (link 11, Fig. 1);

– policy (method) of distribution (link 12, Fig. 1);

– information about connectivity of objectives in the task (link 13, Fig. 1).

On the basis of the received data, a distribution is performed. The result of the work of the module is the plan (link 14, Fig. 1), which defines a computing resource to each task, that is suitable according to the previously mentioned requirements. Any task, accessing to the Grass environment will be distributed, the only possible exception is when the computing resources will not be found in the system.

Stage 6. A process of task sending to the resource or resources, identified in the distribution plan. To do this, an inquiry into the queue of tasks is carried out from the unit which is in charge of transportation (link 15, Fig. 1), which,

according to plan of distribution selects a specific task and assigns it a status of Running. Further, a task (second component), is sent to the selected resource for the next starting (link 16, Fig. 1). As the actions with resources took place, (in this case a resource for task performing is selected), it is required to notify a system about it (link 17, Fig. 1). A redundancy block sends information to the system that the resource cannot be used for the subsequent start, until the finish of task performing (link 18, Fig. 1). Once again, actions with resources emerge, (for example, a release of the resource emerges), the backup unit will receive information about the occurred event (link 17, Fig. 1) and will notify the system of the possibility of using of the resource for distribution (link 18, Fig. 1), and will send a message in the queue of tasks to delete the task from the queue due to the end of its execution (link 19, Fig. 1).

Stage 7. Collection of statistical information about the distribution. To obtain statistical information about the distribution and its following analysis, the module starts up, which is responsible for log-files. In Grass environment a log-file conducting module (Logger) is responsible for collection of statistics. This module provides a flexible and centralized logging of all plugins of GRASS modeling environment.

The system has several types of log-files, which record a series of actions that enables a quick obtaining of information form the database in case of necessity.

At the task accessing into system, an automatic recording of them performs into the queue.log file (link 3, Fig. 1), and at resource accessing (link 4, Fig. 1) – it performs into resources.log. file. At the time of task starting on the resource or at the end of task performing, a fixation of information of this action performs in the raspred.log file (link 20, Fig. 1), i. e., a recording of the file performs. All records from the log files are sent to the database of the system (link 21, Fig. 1), data from which can be used next for sampling the specified parameters or for graphical information display for a certain period of time (link 22, Fig. 1).

The modeling process continues as long as there are tasks in the queue.

6. Peculiarities of GRASS simulation system

GRASS environment is a project with an open source software, which implements modular environment of GRID systems [23], is cross platformed, implemented in C++ language using a cross platform Qt4, Boost libraries.

GRASS modeling environment has both a console and a graphical user interface that enables to observe the simulation of the process in real time. The graphical interface provides the user with more options, as it displays and visualizes statistics of the simulation environment, the status of the queue of tasks and resources, and demonstrates the performance of tasks by computing resources. In addition to the interactive observation of the simulation process, GRASS environment enables to write a report for the following more detailed analysis. The report is implemented using scripts that make it easy to change both its appearance and content.

GRASS modeling environment has a modular structure: It consists of a core and dynamically loaded modules (plug-ins). Each module performs a highly specialized task, referring if necessary to the other modules of the system. The core provides means of inter-module interaction and provides the boot and system configuration. Each module has a unique string identifier (name or ID), which provides a set of interfaces to interact with the other system components. Each interface of the module has also a name and can be implemented by any number of modules. Thus, to achieve any interface of the module it is only necessary to know its name and the name of the interaction interface.

GRASS module environment is a dynamic linked library dll (dynamic linked library), which implements the factory method that creates a sample of module class.

Module class implements a Framework::IPlugin interface, which enables working universally with it, not taking into account peculiarities of implementations.

To create a flexible modular system an easy possibility of changing the set of plug-ins and their parameters is required without modifying the core or the libraries of modules. For this purpose, GRASS environment uses the system of configuration files based on XML.

The developed software system is the basis for the continuation of research in the field of GRID-systems.

7. Discussion of the results of research of application of the proposed information technology

In the study of information technology of distribution of tasks, a series of experiments were conducted. Their aim was to show the rationality not of using of a single broker for all incoming tasks on the system input, but to select the best, focusing on task classes.

2 pools were formed with the GRASS environment: of tasks and resources. Pool of tasks includes 300 tasks, each of which is described by a tuple (3), a resource pool is formed in accordance with the tuple (4), and includes 80 computing resources. Each task is characterized by the requirements, specified by the job providers.

Using different distribution policies a simulation was conducted. The simulation results are compared with each other, and on the basis of the analysis a decision of the best distribution for a particular task pool is made. Fig. 5 shows the run time of all GRASS environment methods for a particular task pool, and Fig. 6 shows the percentage of downtime of computing resources.

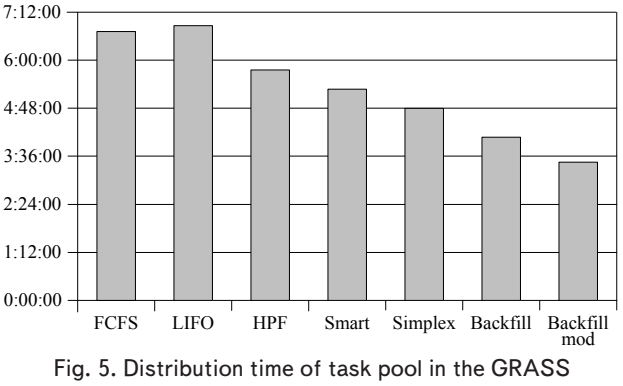

Fig. 5. Distribution time of task pool in the GRASS environment

All tasks, accessing the GRID-system, can be divided into 4 classes, which are characterized by:

– a large amount of input data and a large amount of output data;

– a small amount of input data and a large volume of output data;

– a large of input data and small amount output data;
– a small amount of input data and a small amount of output data.

Table 1 and Fig. 7 show the simulation results of all methods of GRASS environment in accordance with the following classification of tasks.

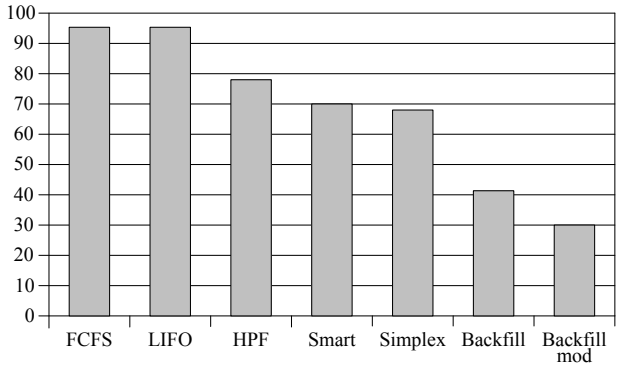

Fig. 6. Downtime percentage of computing resources in GRASS environment

Table 1

Run time of GRASS environment methods for various classes of tasks

Distribution methods	Classes of tasks			
	1st	2nd	3rd	4th
FCFS	6:50:21	5:16:28	5:23:53	2:32:42
LIFO	6:54:43	5:58:50	6:01:52	2:37:53
HPF	6:33:45	6:00:35	6:08:32	2:40:22
Smart	6:28:55	6:03:51	6:10:44	3:09:49
Simplex	6:33:46	5:33:46	4:56:21	2:57:55
Backfill	5:19:54	5:01:49	3:28:43	3:42:46
Backfill (mod)	4:48:39	3:58:23	2:47:33	3:48:01

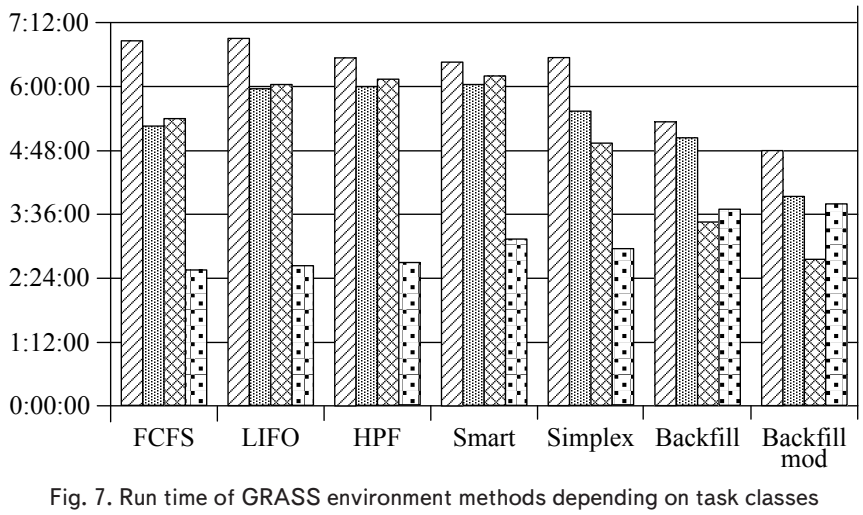

Fig. 7. Run time of GRASS environment methods depending on task classes

In the course of the experiments, a dependence of distribution results to task classes was revealed. There is no unique method of distribution at present, which could enable getting the best plan on any pool of tasks. But in case, when the GRID-system characteristics and the input task stream are known, we can carry out modeling and select distribution method for a certain task pool.

The advantage of GRASS simulation environment is that it operates on an algorithm that analyzing the input stream of tasks, chooses the distribution method that gives the best distribution of the specified requirements of task provider.

8. Conclusions

As the result of conducted research, methods of task distribution in heterogeneous GRID-systems were analyzed. From the analysis we can conclude that the existing schedulers on the market have a number of drawbacks, the main of them is the focus on a specific class of tasks. This can be explained by the fact that these planners were originally intended for cluster systems, which in turn, were later included in the GRID-infrastructure and tasks continued to be carried out locally.

In the process of research a modification of mathematical models of representation of tasks and resources in GRID-system was proposed. The use in representation model tasks a connectivity factor (step 4) enables the selection of computing resources with a minimizing of time for data exchange between tasks. Introduction to the model of representation the computing resources of value, taking into account the amount of traffic and transmission delays of information on the channel, enable to reduce the performance time of the task pool, which will increase the efficiency of the use of computational resources in GRID-system.

Based on a mathematical model of representing of resources and tasks, a method of selection of the best distribution option on the basis of analysis of data obtained during the experiment was developed. The proposed GRASS environment enables to operate multiple distribution algorithms: FCFS, LIFO, HPF, Simplex, Smart, Backfill, Backfill mod that enables to simulate a distribution for a particular pool task at the various distribution policies and to analyze the simulation results in the future. Due to the fact that the proposed system is modular, there are no obstacles for the implementation and connection of new distribution algorithms.

In the course of research, a series of experiments on the distribution of tasks to computing resources in the GRASS simulation-modeling environment for different distribution policies was conducted. The results obtained during the experiments show a decrease in task pool performance time by 24 % and increase of efficiency of use of computing resources by 32 % for a number of the distribution policies (methods), implemented in the GRASS environment.

References

1. Aida, K. Scheduling Mixed-parallel Applications with Advance Reservations [Text] / K. Aida, H. Casanova // Proceedings of the 17th international symposium on High performance distributed computing – HPDC '08, 2008. – P. 65–74. doi: 10.1145/1383422.1383432
2. Ando, S. Evaluation of Scheduling Algorithms for Advance Reservations [Text] / S. Ando, K. Aida // Information Processing Society of Japan SIG Notes. HPC-113, 2007. – P. 37–42.
3. Elmroth, E. A Standards-based Grid Resource Brokering Service Supporting Advance Reservations, Coallocation and Cross-Grid Interoperability [Text] / E. Elmroth, J. Tordsson // Concurrency and Computation: Practice and Experience. – 2009. – Vol. 21, Issue 18. – P. 2298–2335. doi: 10.1002/cpe.1441
4. Cafaro, M. Preference-Based Matchmaking of Grid Resources with CP-Nets [Text] / M. Cafaro, M. Mirto, G. Aloisio // Journal of Grid Computing. – 2013. – Vol. 11, Issue 2. – P. 211–237. doi: 10.1007/s10723-012-9235-2
5. Kurowski, K. Multicriteria Aspects of Grid Resource Management [Text] / K. Kurowski, J. Nabrzyski, A. Oleksiak, J. Weglarz // International Series in Operations Research & Management Science, 2003. – P. 271–293. doi: 10.1007/978-1-4615-0509-9_18
6. Ernemann, C. Economic Scheduling in Grid Computing [Text] / C. Ernemann, V. Hamscher, R. Yahyapour; D. G. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.) // Lecture Notes in Computer Science. – 2002. – Vol. 2537. – P. 128–152. doi: 10.1007/3-540-36180-4_8
7. Rodero, I. Enabling Interoperability among Grid Meta-Schedulers [Text] / I. Rodero, D. Villegas, N. Bobroff, Y. Liu, L. Fong, S. Sadjadi // Journal of Grid Computing. – 2013. – Vol. 11, Issue 2. – P. 311–336. doi: 10.1007/s10723-013-9252-9
8. Azzedin, F. A Synchronous Co-allocation Mechanism for Grid Computing Systems [Text] / F. Azzedin, M. Maheswaran, N. Arnason // Cluster Computing. – 2004. – Vol. 7, Issue 1. – P. 39–49. doi: 10.1023/b:clus.0000003942.73875.29
9. Castillo, C. Resource Co-allocation for Large-scale Distributed Environments [Text] / C. Castillo, G. N. Rouskas, K. Harfoush // 18th ACM International Symposium on High Performance Distributed Compuing, ACM, 2009. – P. 137–150.
10. Takefusa, A. An Advance Reservation-based Co-allocation Algorithm for Distributed Computers and Network Bandwidth on QoS-guaranteed Grids [Text] / A. Takefusa, H. Nakada, T. Kudoh, Y. Tanaka; E. Frachtenberg, U. Schwiegelshohn (Eds.) // Lecture Notes in Computer Science. – 2010. – Vol. 6253. – P. 16–34. doi: 10.1007/978-3-642-16505-4_2
11. Blanco, H. MIP Model Scheduling for Multiclusters [Text] / H. Blanco, F. Guirado, J. L. Lérida, V. M. Albornoz // Lecture Notes in Computer Science. – 2012. – Vol. 7640. – P. 196–206. doi: 10.1007/978-3-642-36949-0_22
12. Garg, S. K. A Linear Programming-driven Genetic Algorithm for Meta-scheduling on Utility Grids [Text] / S. K. Garg, P. Konugurthi, R. Buyya // International Journal of Parallel, Emergent and Distributed Systems. – 2011. – Vol. 26, Issue 6. – P. 493–517. doi: 10.1080/17445760.2010.530002
13. Olteanu, A. A Dynamic Rescheduling Algorithm for Resource Management in Large Scale Dependable Distributed Systems [Text] / A. Olteanu, F. Pop, C. Dobre, V. Cristea // Computers and Mathematics with Applications. – 2012. – Vol. 63, Issue 9. – P. 1409–1423. doi: 10.1016/j.camwa.2012.02.066
14. Toporkov, V. Slot Selection Algorithms in Distributed Computing [Text] / V. Toporkov, A. Toporkova, A. Tselishchev, D. Yemelyanov // Journal of Supercomputing. – 2014. – Vol. 69, Issue 1. – P. 53–60. doi: 10.1007/s11227-014-1210-1
15. Toporkov, V. Slot Selection Algorithms in Distributed Computing with Non-dedicated and Heterogeneous Resources [Text] / V. Toporkov, A. Toporkova, A. Tselishchev, D. Yemelyanov; V. Malyshkin (Ed.) // Lecture Notes in Computer Science. – 2013. – Vol. 7979. – P. 120–134. doi: 10.1007/978-3-642-39958-9_10
16. Toporkov, V. V. Metodi i evristiki planirovaniya v raspredelenih vichisleniyah s neotchujdaemimi resursami [Text] / V. V. Toporkov, A. V. Bobchenkov, D. M. Yemelyanov, A. S. Tselishchev // Vestnik UUrGU, seriya «Vichislitelnaya matematika i informatika». – 2014. – Vol. 3, Issue 2. – P. 43–62.
17. Toporkov, V. V. Metaplanirovanie vichisleniy v raspredelennih sredah s neotchuzhdaemimi resursami [Text] / V. V. Toporkov, D. M. Yemelyanov, A. S. Toporkova // Informacionnie tehnologii v nauke, obrazovanii i upravlenii. IT + S&E`16, 2016. – P. 22–31.
18. Kostromin, R. O. Modeli, metodi i sredstva upravleniya vichisleniyami v integrirovannoy klasternoy sisteme [Text] / R. O. Kostromin // Fundamentalnii issledovaniya. – 2015. – Vol. 6. – P. 35–38.
19. Feoktistov, A. G. Metodoligia konceptualizacii i klassifikacii potokov zadaniy masshtabiruemih prilozheniy v raznorodnoi raspredelennoy vichislitelnoi srede [Text] / A. G. Feoktistov // Sistemi upravleniya, svjazi i bezopasnosti. – 2015. – Vol. 4. – P. 1–25.
20. Venugopal, S. A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids [Text] / S. Venugopal, R. Buyya, L. Winton // Concurrency and Computation: Practice and Experience. – 2006. – Vol. 18, Issue 6. – P. 685–699. doi: 10.1002/cpe.974

21. Astrikov, D. U. Modelirovanie sistemi planirovaniya raspredelennogo visokoproizvoditelnogo vichislitelnogo kompleksa [Text] / D. U. Astrikov, D. A. Kuzmin, A. I. Panacuk // Dokladi Akademii nauk Vissheiy shkoli Rossiyskoi federazii. – 2014. – Vol. 2-3, Issue 23-24. – P. 34–41.
22. Milukov, V. V. Modelirovanie fragmentov GRID-sistemi v simulatore GridSim [Text] / V. V. Milukov, U. V. Sosnovskiy // Optimizaciya virobnichih procesiv. – 2013. – Vol. 14. – P. 218–222.
23. Volk, M. A. Arhitektura imitatsionnoy modeli GRID-sistemyi, osnovannaya na podklyuchaemyih modulyah [Text] / M. A. Volk, A. S. Gorenkov, R. N. Gridel // Sistemi obrobki informatsii. – 2010. – Vol. 1, Issue 82. – P. 17–20.
24. Filimonchuk, T. V. Informatsionnaya tehnologiya raspredeleniya zadaniy na vyichislitelnyie resursyi v GRID-sistemah [Text] / T. V. Filimonchuk, V. N. Tkachev // Informatika, matematicheskoe modelirovanie, ekonomika. – 2015. – Vol. 1. – P. 204–209.
25. Volk, M. A. Struktura programmnogo kompleksa imitatsionnogo modelirovaniya elementov GRID-sistem dlya nauchnyih issledovaniy [Text] / M. A. Volk // Sistemi obrobki Informatsii. – 2009. – Vol. 3, Issue 77. – P. 125–128.
26. Volk, M. A. Modul raspredeleniya zadaniy v GRID-sistemah [Text] / M. A. Volk, M. A. Filimonchuk, T. V. Filimonchuk // Sistemi obrobki informatsii. – 2012. – Vol. 2, Issue 100. – P. 177–182.
27. Volk, M. A. Obobschennyiy kriteriy otsenki zadaniya dlya tehnologii planirovaniya zadaniy v GRID. In 3 volumes. Vol. 2 [Text] / M. A. Volk, T. V. Filimonchuk // Informatika, matematicheskoe modelirovanie, ekonomika, 2013. – P. 172–176.