UDC 004.724.2

LOAD BALANCING ALGORITHM FOR GRID TASK SCHEDULING IMPROVEMENT

Pavlo Svirin

Institute of Applied System Analysis, NTUU "KPI", Kyiv, Ukraine

In order to satisfy the users' requirements for the productivity and efficiency of tasks implementation the grid-system should implement the effective algorithm of tasks distribution between the computing resources accessible at the moment. The primary purpose of such load balancing in the grid-system is to reduce the time necessary for the user's task implementation. Moreover, it will provide efficient usage of the computing resources and eliminate such a situation that some resources stand idle when the other are overloaded with users' tasks implementation.

Introduction

Despite the load balancing algorithms in computing resources in Grid are being studied for a long time and the availability of many ready algorithmic decisions as well as software implementations, the intensive development of Grid technologies and improvement of middleware makes the problem of load balancing and up-to date and the interest towards research activities in this area does not decrease. The main purpose of such load balancing in Grid is to decrease the overall execution time for the user's task and ensure utilization efficiency of the computing resources.

The main tasks that require Grid are the following:

- large number of tasks with low requirements regarding the resources. Such tasks are executed over a short period of time;
- large number of tasks with high requirements regarding the resources that are executed over a long period of time.

Examples of such tasks are as follows:

- ALICE experiment data processing. Usually such task requires 1 processor; the data are transferred to the resource in the course of calculation, maximum running time is 24 hours. Number of such tasks can range up to hundreds of thousands;
- Calculation of molecular dynamics tasks. This category of tasks requires a big number of processors, transfers a small amount of data for calculation, maximum running time of such tasks is up to months. Number of such tasks can range up to thousands.

Hence, the use of a single strategy for distribution of various categories tasks is not efficient. The solutions to this problem are specialized Grid systems such as AliEn Grid, WeNMR. However, the number of tasks categories is extremely important and it is not possible to develop a system for each category of task.

General information

Task scheduling is a well-known problem for distributed computing. Extensive research on this problem applied to Grid environments has been conducted by Buyya [1], Foster [2], Heiss [3]. Nevertheless, every specific Grid environment may require its specific broker algorithm according to the tasks being scheduled or restrictions of the environment.

The common form for the resource state evaluation is (1):

$$P = f(x_1, x_2, ..., x_n) \quad (1)$$

where P is an evaluation rank for a computational resource, f is some function which implements a resource evaluation algorithm of a broker, $x_1..x_n$ are computational resource parameters.

Thus, the task of finding an optimal resource for task execution means finding an optimal resource rank according to the algorithm used. This can be a maximum or minimum value for some set of computational resources.

If there are some parameters that can be omitted white resource state evaluation, then (1) can be rewritten as (2):

$$P = f(x_1, x_2, ..., x_n, y_1, y_2, ... y_m), \quad (2)$$

where $x_1..x_n$ are mandatory parameters for resource state evaluation, $y_1 \ldots y_m$ – not mandatory parameters.

Global task scheduling approaches and problems

Global Grid segment architecture can be:

1) centralized – this approach has on central queue for whole of the Grid segment. Grid users submit their tasks to this queue and they are distrib-

uted among the computing resources. Although this approach has more options for task execution control, the central queue can be a single point which may cause failure for the whole Grid segment. Example for this architecture is gLite.

2) decentralized – in this approach there is no central queue. Grid users pass tasks directly to the queues of available computing resource. Such architecture is more stable, nevertheless, it is more complicated to control the resource load and task scheduling. Example for this architecture can be Nordugrid ARC.

The main problems for task scheduling are:

- tasks are being distributed to resources inappropriate for execution;
- broker does not take into account current status of a resource;
- the task is being executed for too long time because of too slow resource and is dropped because of maximum execution time exceeded;
- Broker delivers tasks to a resource with long task queue;
- Some of the resources stay without load after all of the tasks have been dispatched, some stay with long queue.

Problem definition for UNG

Ukrainian Grid infrastructure is made by the use of ARC (Advanced Resource Connector) middleware also known as project NorduGrid [4].

In ARC both 0.8 version and new ARC 2.0 version use full maximum decentralization as the main principle therefore the personal broker is installed on every workbench of the Grid network user. The broker's function is to opt for the best resource to execute the user's task in the Grid network.

Currently in UNG the random selection of the resource is used, however it does not take into account the current state of the existing resources. For more efficient distribution of load among the resources personal brokers which take into account both the current state of the resources and the load balancing policy should be developed. Nordugrid ARC package contains the simplest policies therefore the suggested methods can be used not only in UNG but also for other segments and virtual organizations having specific and general tasks.

The specifics of Ukrainian Grid infrastructure are the following:

- 38 clusters with low computational performance [5];
- Only 2 high computational performance resources are available;
- All resources are managed by ARC;
- Various calculation subjects: molecular dynamics, physics, chemistry, astronomy etc., a high number of virtual organizations.

Specifics of brokers in Nordugrid ARC:

- Availability of only simplified policies for tasks distribution
- The system is targeted at ATLAS [6-9] experiment data processing with prevailing short tasks having small amounts of data. The broker that draws a conclusion regarding the target resource taking into account the amount of required data in the computational resource cache was developed specifically for this experiment. In such way it decreases the data transfer time.

Therefore Ukrainian segment lacks brokers suitable for efficient distribution of tasks of all categories.

In reality the tasks that require 10-30 processors are sent to the cluster of the Cybernetics institute and they await for days in a queue to be executed. Shorter tasks can also be directed there and also wait in queue.

Hence the goal of the optimal broker for UNG is:

- submit shorter tasks to weaker resources;
- submit longer tasks to more powerful resources;
- choose the resources with the earliest start time for the task.

Methods for resource brokers design for Nordugrid ARC

General sequence of stages in the process of task submission is the following (Fig. 1):

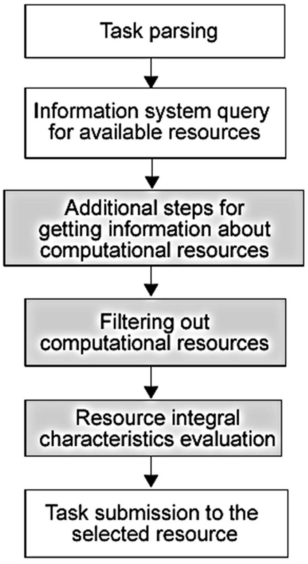

Fig. 1 Steps performed while submitting to a computational resource.

1. Task is being parsed, checks for description correctness are performed;
2. Information system is queried for a list of available computational resources is discovered;
3. The algorithm queries external data sources for additional information about the available resources from the list fetched on step 2.
4. The algorithm performs checks for computational resources and filters out those which do not satisfy task requirements or other restrictions.
5. Resource integral characteristics evaluation – the algorithms performs rank calculation, then the resource list is sorted according to this rank.
6. Task submission – the task description together with its data is passed to the selected computational resource for execution.

Steps 3-5 are to be implemented in order to develop a personal broker algorithm.

The following is required to design an efficient broker:

- To define the task area and environment requirements for which the broker can carry out efficient distribution between the resources;
- To ensure information receipt from the information system;
- To define the criteria for rejection of the resource candidate;
- To define the set of characteristics to be used for load evaluation of the computational cluster;
- To develop the sequence of steps for ranking the clusters list and rejection of clusters not suitable for task requirements.
- To develop the procedure for tasks sending sequence in case of need to simultaneously distribute few tasks.

There are two methods of resources ranking:

- Consecutive evaluation and rejection of resources based on each criteria;
- Calculation of integral characteristic for each resource and further ranking according to this characteristic.

The second approach has already been discussed in few articles, in particular [10], in which authors have developed an integral characteristics of the resource based on its specific capacity, memory space, number of processors, processors loading during the preceding minute and the rate of network load for the resource.

In Nordugrid ARC resource broker is located on the client side because this platform uses decentralized approach. Thus, every Grid client can use its own broker implementation (Fig. 2)

Nordugrid ARC 2.0 platform enables use of web services for implementation of special purpose services. Therefore, it is possible to implement a service that will return an extended information about the resource. Monitoring systems such as Nagios etc. can also be another source of extended information. The article [11] reviews the option of the broker development for Ukrainian Grid segment [12]. The brokers suggested in current article do not require installation of services beyond the platform Nordugrid ARC.

From the architectural point of view the brokers for Nordugrid ARC can be divided into:

- constant – modules compiled into the program
- variable – such modules can be changed without recompilation of the entire code of client's software. For instance it is possible to occasionally download broker updates within the frameworks of a certain project. The algorithm code in this case is written in scripting language Python. During running the program of task submission the list of candidates-resources is submitted to the present script and the script returns the ranked resources lists at the output.

From the algorithm point of view brokers can be divided as follows:

- brokers calculating overall estimation coefficient for a resource. It may take into account both static and dynamic resource characteristics as well as static task requirements for the resource which enables balancing the load between the resources according to a certain policy;
- brokers using table method – resources ranking is carried out in accordance with already accumulated data regarding the duration of certain types of tasks existing in the project.

Main information sources about Nordugrid ARC resources are:

- AREX – service controlling the tasks execution. Returns the list of static and dynamic characteristics described by GLUE 2 schema;
- Infosys – information system Nordugrid ARC for version 0.8 and lower. It is based on OpenLDAP and contains the list of registered computing resources and their current state for static and dynamic characteristics;
- ISIS – represents a registry of services that enable obtaining the information about the resources.

Possible additional sources are:

- Additional services of Nordugrid Arc platform.
- Monitoring systems Network Weather Service, Ganglia and others;
- Other sources.

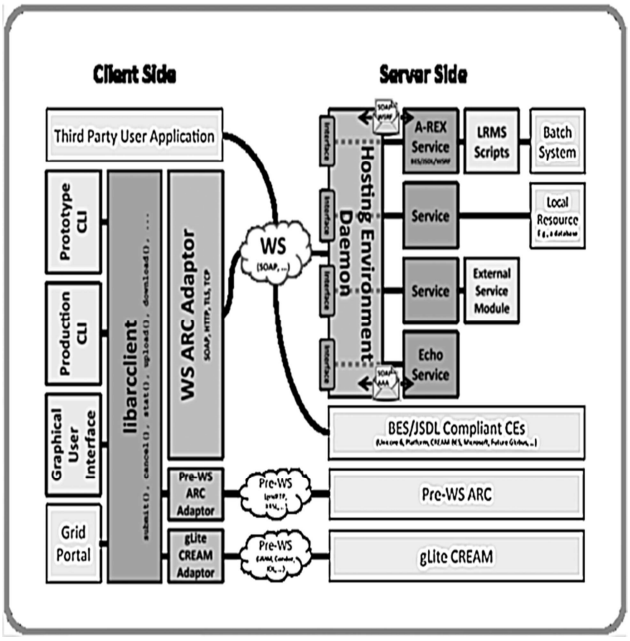

Fig.2 Nordugrid ARC architecture

For example, the widely used in the Grid environments information system Ganglia provides the following data:

- the number of operating resource nodes at the current time;
- resource network status;
- recent cluster load.

Simulation overview

In order to simulate the possible enhancements for Nordugrid ARC broker we performed a set of simulations in order to research the efficiency of different approaches. We used Alea3 and metacenter.mwf task file for broker simulation with 1000 tasks to process.

Random

This is a default static algorithm in Nordugrid ARC which is used in UNG. This approach selects randomly an available resource and then passes the task to it.

While using this approach the set of tasks has been processed in more than 400 days with unbalanced load (some resources are overloaded for a certain period of time, some remain free and have no incoming tasks in the queue) (Fig. 3).

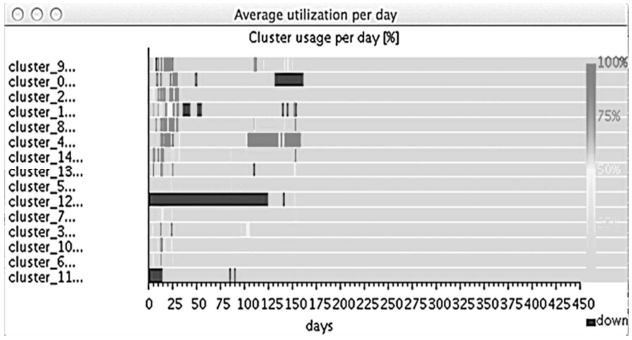

Fig.3 Resource load for Random policy

Best suitable

This broker algorithm uses productivity rank from the information system for making a decision. Available computing resources are ranked by this parameter and the resource with the best productivity rank receives the task.

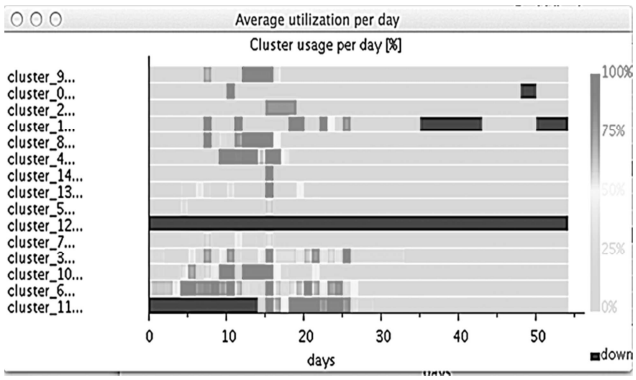

Fig.4 Resource load for Best Suitable policy

The set of tasks has been processed in about 55 days (Fig. 4), which shows much better efficiency than Random algorithm, although this approach is still static and does not take into account current state of the resource.

Earliest Start time prediction

This dynamic broker uses execution time estimations from task registry service, which stores task execution time evaluations for every task type in the domain of tasks and also a task tracker service which tracks execution percentages for every task.

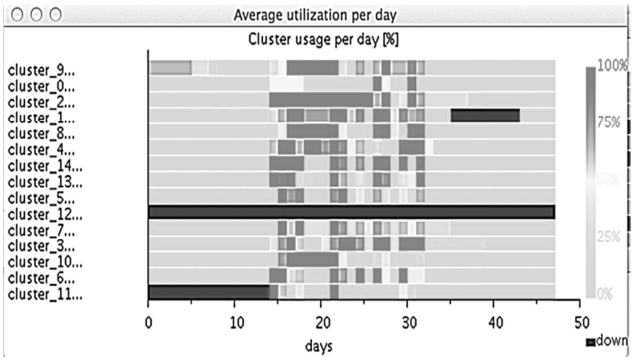

Fig.5 Resource load for Earliest Start policy

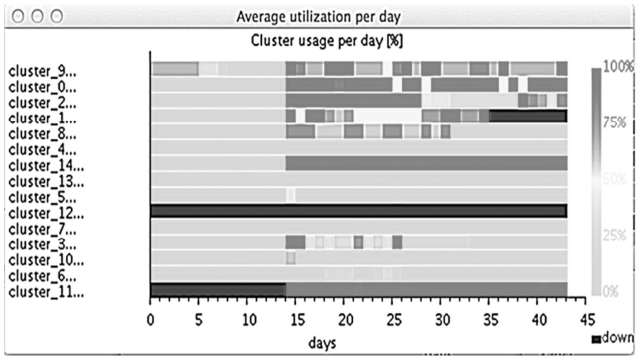

Fig.6 Resource load for Earliest Start policy with rescheduling

The earliest start time on a certain resource is evaluated by (3):

$$TL_j = \sum_{i=1}^{n} l(t_i)(1 - p_i) \quad (3)$$

where TL_j – estimated time left until tasks that were submitted to the j-th resource will be accomplished, n – number of tasks delivered to the resource, $l(t_i)$ – a function which estimates the length of the task i, p_i – completed percentage for the task i.

The simulation showed considerable improvement in productivity comparing to the previous one together with more balanced load across the computing resources. The set of tasks has been processed in 47 days (Fig. 5). Nevertheless, this approach needs additional software and task evaluations for different CPU types. However, it can be used for the sets of tasks where most of the task types are known and thus their execution times are already evaluated.

Resource load for Earliest Start policy with rescheduling

This is an extension for the algorithm described in previous section.

This broker uses execution time estimations from task registry together with task transfer to free resources from the long queue of another resources.

The rescheduling approach uses server-requested policy, when a server which has no load asks its neighbor resources for task which stand in a long queue and, thus, have to wait for a long time before their execution starts. If there is such task – it is transferred to a free resource which starts its execution immediately [13].

The simulation showed further improvements for productivity: the same set of tasks has been processed in 43 days (Fig. 6) which gives another 10% boost.

The rescheduling feature uses the first free resource available. In order to extend this approach it is possible to evaluate the free resource to find a free resource with the best productivity. This improvement gives decreases task set processing time to 41.8 days (3% boost comparing to the previous approach). (Fig. 7)

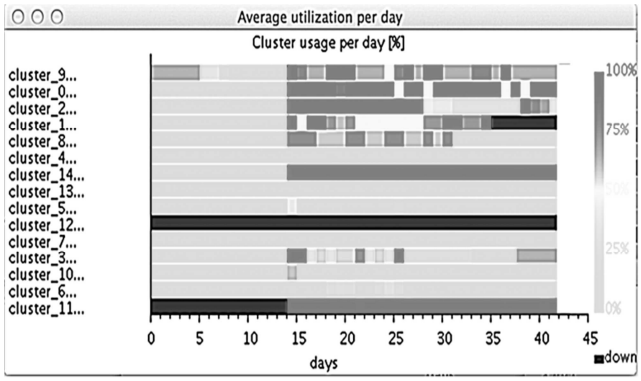

Fig.7 Resource load for Earliest Start policy with rescheduling to a resource with the best productivity.

The software required to run this broker is the same as for the previous approach, however the tracker has to track the task URI change after the rescheduling has been completed. The data schema for services is shown on Fig.8. TaskTypes table stores the task types list together with the function which estimates execution time. Such functions can be written in some interpreted language like Lua, which can be interpreted via C API interface and can represent a function calculating the task time using input parameters or just returning a constant number. Task table tracks the URI change for a task and its completed percentage.

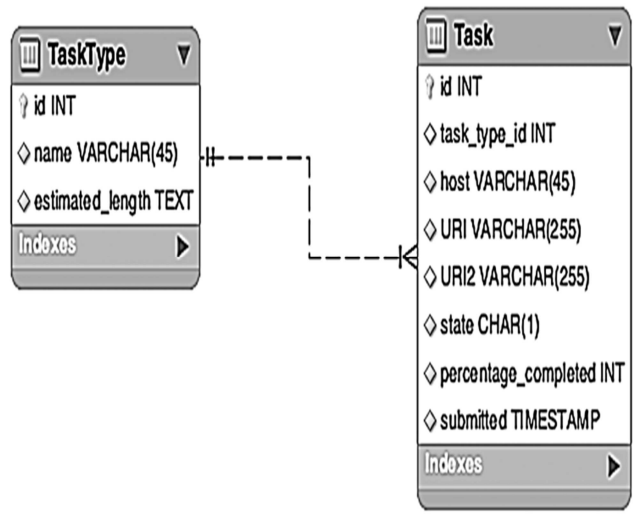

Fig.8 Database schema for the service that stores data about rescheduled tasks

The process of task result retrieving now has two stages:

- the user contacts the computing resource where the task had been initially submitted to;
- the resource returns the new URI where the task has been transfered to and executed;
- the user contacts the computing resource using this new URI and retrieves the task results.

Conclusions

This article reviews the principles of resource brokers design for Nordugrid ARC as well as review of possible information sources for ranking the resources list by broker. The options for implementation of the load distribution algorithm are presented.

The simulation results show that every broker has its own advantages and disadvantages. For example, more effective ones need additional software and complex structure which may be inacceptable in certain situations.

Obtained results enable making the choice of sources that allow appropriate evaluation for the resource current state and choose the best broker algorithm or its architecture according to the conditions of computing environment or virtual organization requirements.

The approaches shown in this article also allow to develop other implementations or using other additional sources of information.

References

1. Buyya R., Abramson D., Giddy J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid // Fourth International Conference on High Performance Computing in Asia. – China: Pacific Region(HPC Asia 2000) Beijing, – 2000. – P. 283–289.
2. Foster I., Kesselman C., Tuecke S. The anatomy of the grid: Enabling scalable virtual organizations // Cluster Computing and the Grid, IEEE International Symposium on. – Los Alamitos, CA, USA: IEEE Computer Society, 2001 – Vol.15, No.3 – p. 200-222.
3. Heiss H.-U., Schmitz M. Decentralized dynamic load balancing: The particles approach // Information Sciences – 1995 – № 84. – P. 115–128.
4. Nordugrid ARC website. http://www.nordugrid.org
5. Grid Monitor website. http://gridmon.bitp.kiev.ua/
6. A. Read, A. Taga, F. Ould-Saada, K. Pajchel, B. H. Samset, D. Cameron. Complete Distributed Computing Environment for a HEP Experiment: Experience with ARC-Connected Infrastructure for ATLAS. http://www.nordugrid.org/documents/chep07-atlas.pdf
7. Kennedy J. ATLAS Production System. http://www.etp.physik.uni-muenchen.de /dokumente/talks/jkennedy_dpg07.pdf
8. Werner J. Grid computing in High Energy Physics using LCG: the BaBar experience. http://www.gridpp.ac.uk/papers/ahm06_werner.pdf
9. L. Boyanov, P. Nenkova. On the employment of LCG Grid middleware. http://ecet.ecs.ru.acad.bg/ cst05/Docs/cp/SII/II.11.pdf
10. Chao-Tung Y., Sung-Yi C., Tsui-Ting C. A Grid Resource Broker with Network Bandwidth-Aware Job Scheduling for Computational Grids. // Advances in Grid and Pervasive Computing. – 2007 - Vol. 4459. - pp.1 - 12.
11. Petrenko A., Svistunov S., Svirin P. Grid site load evaluation algorithm // Proceedings of «System Analysis And Information Technologies», May 23-28 2011, Kyiv, Ukraine – p. 388. [in Ukrainian]
12. Zagorodniy, A., Zinovyev, G., Martynov, E., Svistunov, S. Ukrainskiy akademicheskiy Grid. Ukrayins'ko makedons'kiy naukoviy zbirnyk – 2009 – 4 - pp. 140-150. [in Ukrainian]
13. Livny M., Melman M. Load Balancing in Homogeneous Broadcast Distributed Systems // Proc. ACM Computer Network Performance Symposium (April 1982) - pp. 47-55.

Received in final form November 11, 2013