MOKSLO TAIKOMIEJI TYRIMAI LIETUVOS KOLEGIJOSE, 2021 Nr.17 (2), p. 123–129
ISSN 1822-1068 / eISSN 2335-8904, http://ojs.kaunokolegija.lt/index.php/mttlk/index

SELF-LEARNING OF PARAMETER WEIGHTS FOR TASK SCHEDULING IN GRID COMPUTING ENVIRONMENT

Donatas Sandonavičius[1], Aušra Gadeikytė[1], Giedrius Paulikas[1], Mindaugas Vaitkūnas[1], Gytis Vilutis[1], Gintaras Butkus[2]

[1]Kaunas University of Technology, [2]Kaunas University of Applied Sciences

Abstract. The Grid computing environment is very important for solving scientific problems. To get the best performance from Grid, it is important to know where to send tasks. This paper is about one of the suggested methods for a Grid resource broker to find the best resources for the task. This method requires defining the parameters of the resources and knowing the importance of the weights of parameters. This paper also presents the self-learning method of parameter weights.

Keywords: Grid, Cloud, Quality of Service, Resource Broker, Self-learning of parameter weights.

Introduction

Grid and Cloud networks provide SaaS (Software as a Service) services that process various user tasks, for example, video rendering - a popular and computing-intensive task. Because different Grid vendors share different sets of resources, the timing of the service itself also differs. Considering the fact that different sets of resources have different payment plans, the quality of service for users is determined mainly by these two components: price and execution time. It is important for the user to choose the Grid supplier with the best price-to-service ratio. Large Clouds, like Grids, combine a large number of computing resource clusters and vendors themselves outsource tasks to the least loaded and best fitting cluster. Therefore, the problem of fast task execution is relevant for both end-users and SaaS service providers.

Suboptimal selection of grid resources hinders the execution of user tasks. This article presents the Quality of Grid Service (QoGS) method that uses resource parameters to select the most appropriate resource. However, different types of tasks require different weights of parameters (WoP). Proper selection of WoP determines the quality of the service received in the Grid. Most frequently, the quality of the service is defined by the following parameters: computation time, data transfer rate, data security and parallelism (Sulaiman, Halim, Lebbah, Waqas, Tu, 2021; Wang, Wang, 2021; Lavanya, Shanthi, Saravanan, 2020). The effective selection of resources ensures faster execution of computing tasks and more precise results. As resource availability changes over time depending on Grid load, constant updating of WoP is required to ensure the quality of Grid services.

The problem of task scheduling is pertinent to computing Grid networks and SaaS Clouds (Mahato, Sandhu, Singh, Kaushal, 2020; Rawat, Dimri, Gupta, 2020; Ankita, Sahana, 2020; Gabri, Agrawal, Srinivas, 2020; You, Luo, He, 2020; Abualigah, Diabat, 2021). Grid resources are usually selected by the Estimated Response Time (ERT) method (McBride, Krznaric, Darlington, van der Aa, Aggarwal, Colling, 2006). ERT method selects resources by comparing them only by two parameters: the queue length for the resource and CPU speed. If the resource selection is wrong, Grid will have both free and overloaded resources. Some resources will have long queues of tasks. That results in a lower quality of service for the user because the tasks take longer to complete. Temporary certificates are sent with the task to Grid, and they determine how long the task can be run on Grid. As a result, some tasks may be lost altogether due to the expiration of their certificates. The selection of resources impacts not only the time of the task execution but also the time of the other tasks. The main reason that prevents researchers from creating the optimal resource selection method is the lack of data on queue waiting time and task execution time (Sharma, Kumar, Jain, 2020; Chen, Yuan, Wang, Luo, Luo, 2020).

The main focus of this article is to define the method for evaluating parameters of Grid resources that affect the quality of the service provided by the Grid. The evaluation of parameters is used to select the best-suited resources for the user's tasks. This improves the efficiency of Grid resource utilization, shortens the average task completion time and reduces the number of tasks rejected.

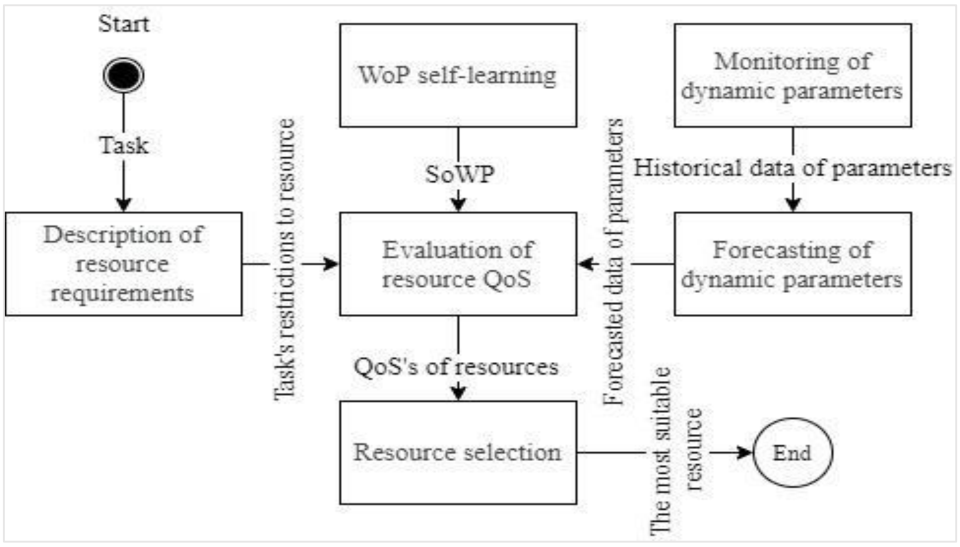

Start
Task
Description of resource requirements
Task's restrictions to resource
WoP self-learning
SoWP
Evaluation of resource QoS
QoS's of resources
Resource selection
Monitoring of dynamic parameters
Historical data of parameters
Forecasting of dynamic parameters
Forecasted data of parameters
The most suitable resource
End

Fig. 1. QoGS method

Grid resource selection method

ERT (Estimated Response Time) is a common Grid computing method for selecting resources, and it performs well for identical tasks (Fig. 1). However, this is rarely the case. Grid usually operates with diverse tasks that exercise an uneven load of Grid resources. The ERT method selects resources using only two parameters: J_{NR} - the length of the task queue for the resource and *CPS* - processor speed/number of processors. That's not enough for selecting the most appropriate resource because the task may be sent to the resource where the waiting time is too long.

This article presents a new method for resource selection, called QoGS (Quality of Grid Service), which chooses resources according to the quality of the service (QoS) of the resource. The Resource QoS is calculated after evaluating resource parameters and is used to select the most suitable resource for task execution.

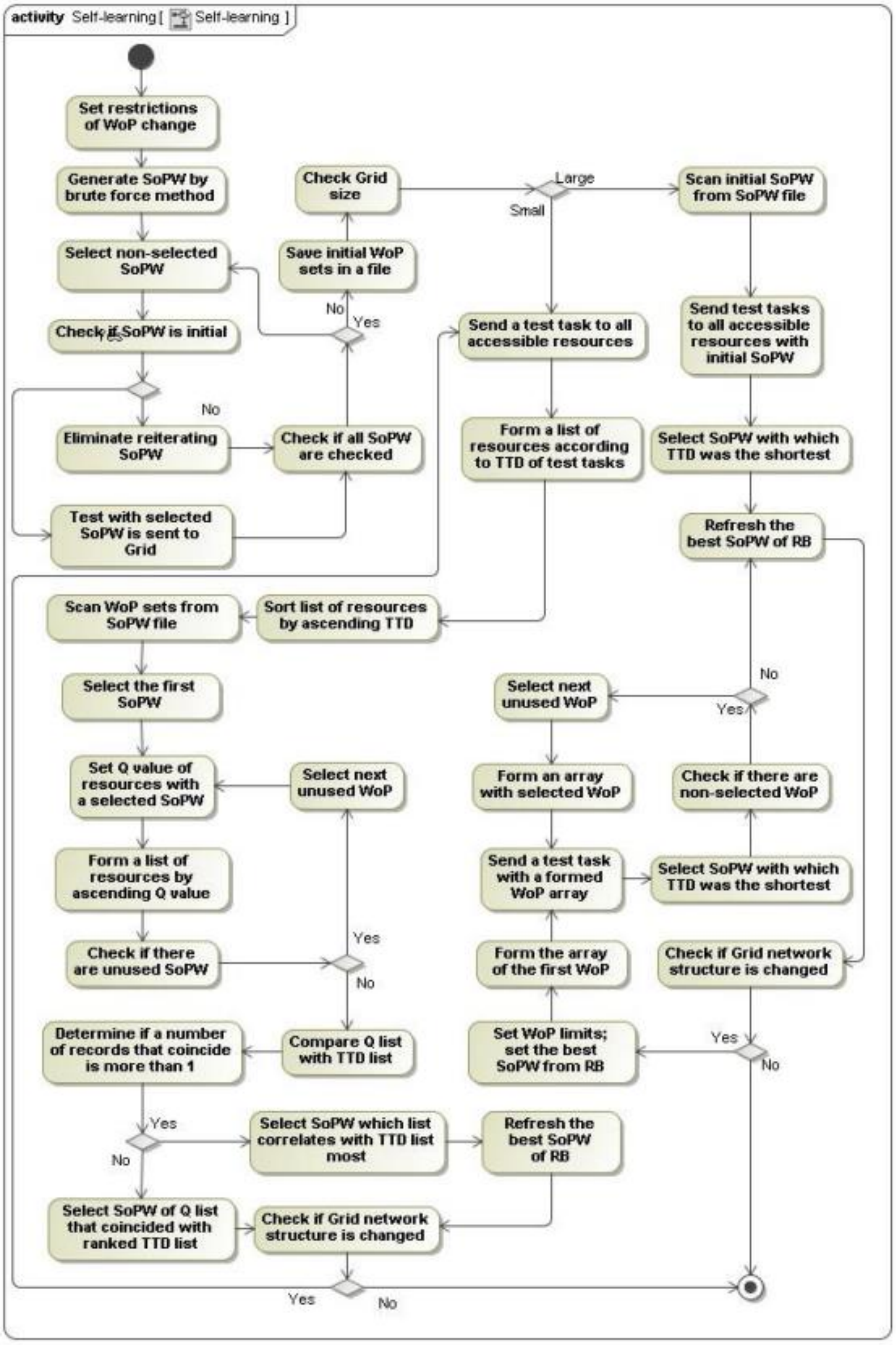
QoGS method has six main components (Figure 1): "Description of resource requirements", "WoP self-learning", "Monitoring of dynamic parameters", "Forecasting dynamic parameters", "Evaluation of resource QoS" and "Resource selection".

Before sending a task to Grid, the user may set its resource requirements. The task with the attached requirements is sent to the Resource Broker (RB). "Monitoring of dynamic parameters" and "Forecasting dynamic parameters" output, combined with task resource requirements, allow RB to select the most appropriate resource for the task execution. SARIMA method is used for dynamic parameter forecasting (see a detailed discussion about this component in Sutiene, Vilutis, Sandonavicius, 2011). Forecasting the load of available resources is a major task in both Cloud and Grid environments (Gao, Wang, Shen, 2020; Masdari, Khoshnevis, 2020; Qionga, Zhiyongc, Xiaolua, 2020; Gadhavi, Bhavsar, 2019). Before selecting a resource, RB evaluates the QoS of all resources using sets of parameters weights (SoPW). If the user did not specify the SoPW in task requirements, SoPW are taken from the "WoP self-learning" component (see Section "Self-learning of QoGS SoPW"). Self-learning in this component is activated by RB at defined time intervals and provides the currently best SoPW for the "Evaluation of resource QoS" component. The latter uses static parameters, resource requirements, SoWP and dynamic parameter forecasting to compute QoS for each resource (Pilkauskas, Plestys, Vilutis, Sandonavicius, 2011). "Evaluation of resource QoS" emits the resource QoS values, the resource with the highest QoS is selected, and eventually, the task is sent to that resource. The main focus of the QoGS method is on the "Evaluation of resource QoS" component that is responsible for selecting adequate resources.

By default, coefficients are set by the user. If WoP in resource brokers were selected and adjusted by "WoP self-learning", it would be easier for users to describe tasks before sending them to the Grid. This is presented in the next section.

Self-learning of QoGS SoPW

The selection of resources is directly related to the correct SoPW. Setting WoP is not enough; it is necessary to correct SoPW in time because the Grid size and behaviour change (Fig. 2). If the user's main criterion for the quality of service is determined by the shortest time to the results of the task (TTD - Time To Delivery: the time interval from sending the task to the execution), it is necessary to take into consideration many parameters of resources when selecting for the best (Vilutis, Sandonavičius, 2008). The most suitable SoPW for each Grid network may differ. Therefore, RB requires the self-learning mechanism of determining WoP (Figure 2). These WoP are used in the formula (1) calculating *Q*.



activity Self-learning [Self-learning]
Set restrictions of WoP change
Generate SoPW by brute force method
Select non-selected SoPW
Check if SoPW is initial
No
Eliminate reiterating SoPW
Test with selected SoPW is sent to Grid
Check if all SoPW are checked
Yes
No
Save initial WoP sets in a file
Check Grid size
Large
Small
Scan initial SoPW from SoPW file
Send a test task to all accessible resources
Send test tasks to all accessible resources with initial SoPW
Form a list of resources according to TTD of test tasks
Select SoPW with which TTD was the shortest
Refresh the best SoPW of RB
Scan WoP sets from SoPW file
Sort list of resources by ascending TTD
Select the first SoPW
Set Q value of resources with a selected SoPW
Select next unused WoP
Form a list of resources by ascending Q value
Check if there are unused SoPW
Yes
No
Determine if a number of records that coincide is more than 1
Compare Q list with TTD list
Yes
No
Select SoPW which list correlates with TTD list most
Refresh the best SoPW of RB
Select SoPW of Q list that coincided with ranked TTD list
Check if Grid network structure is changed
Yes
No
Select next unused WoP
No
Yes
Form an array with selected WoP
Check if there are non-selected WoP
Send a test task with a formed WoP array
Select SoPW with which TTD was the shortest
Form the array of the first WoP
Check if Grid network structure is changed
Set WoP limits; set the best SoPW from RB
Yes
No

Fig. 2. Grid QoGS self-learning algorithm

$$Q_{nk} = \begin{cases} 1, when\ N = 1\ and\ V_{nj}^{M} \le V_{kj}^{M_R}\ and\ V \ge V_{ki}^{D_R}, i = \overline{1,I}, j = \overline{1,J}, n = \overline{1,N} \\ \dfrac{\sum_{i=1}^{I}\left(\omega_{ki}^{D} \cdot \frac{V_{ni}^{D}}{V_{i}^{D\,max}}\right) + \sum_{j=1}^{J}\left(\omega_{kj}^{M} \cdot \frac{V_{j}^{M\,max} - V_{nj}^{M}}{V_{j}^{M\,max} - V_{j}^{M\,min}}\right)}{\sum_{i=1}^{I}\omega_{ki}^{D} + \sum_{j=1}^{J}\omega_{kj}^{M}}, \begin{matrix} when\ V_{nj}^{M} \le V_{kj}^{M_R} \\ and\ V_{ni}^{D} \ge V_{ki}^{D_R}, N > 1 \\ i = \overline{1,I}, j = \overline{1,J} \end{matrix} \\ 0, when\ V_{nj}^{M} > V_{kj}^{M_R}\ or\ V_{ni}^{D} < V_{ki}^{D_R}, i = \overline{1,I}, j = \overline{1,J}, n = \overline{1,N} \end{cases} \quad (1)$$

where: n – resource number; $n = \overline{1,N}$, N – resource number in Grid; k – task number; V_{ni}^{D} – the value of the *i*-th parameter of the *n*-th resource ("*D*" marks the parameter, and when the value of this parameter increases, the qualitative estimate increases); V_{j}^{max} – the highest value of the *i*-th parameter in the Grid network ($V_{i}^{D\,max} = \max_{n=1,\dots,N} V_{ni}^{D}$); V_{nj}^{M} – the values of the *j*-th parameter of the *n*-th resource ("*M*" marks the parameter and when the value of this parameter increases, the qualitative estimate decreases); $V_{j}^{M\,max}$ – the highest and $V_{j}^{M\,min}$ the lowest values of the *j*-th parameter in Grid network ($V_{j}^{M\,max} = \max_{n=1,\dots,N} V_{nj}^{M}, V_{j}^{M\,min} = \min_{n=1,\dots,N} V_{nj}^{M}$); $V_{ki}^{D_R}$ and $V_{kj}^{M_R}$ – limits of values for the *i*-th and *j*-th user-set parameters, which indicate the requirements set for the execution of the *k*-th task; ω_{ki}^{D} and ω_{kj}^{M} – weights of the *i*-th and *j*-th parameters that are set for the *k*-th task.

The easiest way of obtaining the best SoPW is to send tasks with all possible SoPW to the Grid network and observe which SoPW results in the fastest execution of tasks. This approach is very time-consuming and was discussed in Vilutis, Butkiene, Lagzdinyte-Budnike, Sandonavicius, Paulikas (2013). This paper presents an improved and more detailed method, which also suggests checking the Grid network size. When the Grid network is small, and the amount of WoP in the set is large, the Grid network experiences an unnecessary load of testing tasks, and a large number of tasks end up at the same resources. Therefore, this way is only suitable for large Grid networks. For small ones, another self-learning mechanism that does not saturate the network by test tasks is needed. In the case of a small Grid network (less than 100 resources), to avoid overloading the Grid network by test tasks, it is recommended to execute self-learning with the least amount of test tasks. When the amount of resources in the Grid network is large, sending test tasks with selected SoPW is used, and only then WoP parameters are specified. The WoP that are selected by self-learning of QoGS require correction. However, the disadvantage of these self-learning methods is that self-learning uses only a small amount of available statistical data, whereas to employ the QoGS method, all SoPW are required. A well-known method of brute force can be used to obtain the best-fitting SoPW.

The application of the brute force method in QoGS needs the following data: WoP change restrictions and step size for the change. It's best to restrict WoP change to [0;10] with a step size of 1. To find the most suitable SoPW, a complete reselection of all SoPW is performed and used in the Grid network simulation to find the SoPW that can use the Grid network resources most efficiently. After the best SoPW finding research, the following restrictions are recommended for applying the QoGS method in the Grid network to search for the best SoPW:

- the queue length for jobs J_{NR} is provided with WoP value between [5;10];
- the time of jobs queued T_{Q}^{C} is provided with WoP value between [5;10];
- the central processor speed *CPS* is provided with a WoP value between [1;5];
- the amount of working nodes *WNA* is provided with WoP value between [1;5];
- other parameters have WoP values between [1;5].

It is possible to expand WoP restrictions up to [0;20] or [0;50], but it will not improve the operation of the QoGS method much. Meanwhile, due to the much bigger amount of the SoPW used in the brute force, the search for the best SoPW slows down significantly.

This is why there are cases when values that define resource QoGS according to the formula (1) are equal while SoPW are different.

The brute force method can also be used partially because normalization of the parameters V_i and V_j is performed with the help of the denominator in the formula (1). The condition for the rejection of SoPW is this: SoPW is rejected if another SoPW was already used where all ω_i equally affected the values of the parameters V_i when the resource Q was processed [20].

For example, when the WoP amount increases from 3 to 4, the amount of repetitive SoPW increases from 158 to 720.

After rejecting repetitive SoPW, the selected SoPW is called the partial brute force SoPW. When the QoGS method is applied for the first time, it is necessary to find the best SoPW. So, at first, all

WoP are set (values of 1 are recommended), and test tasks are sent with the partial brute force SoPW. If the number of parameters is large, a very large number of test tasks are sent to the network (for example, when parameters are 4, about 9280 test tasks are sent). This high load for the Grid network makes the execution of self-learning attractive. The brute force method is used only for the first time, while later, only the test tasks necessary for specifying parameters are sent to the network. When running learning for the first time, it is recommended to use the partial brute force SoPW method.

The study of rejection of repetitive SoPW was performed in this research. To estimate how many sets were rejected, parameter quantities of 2, 3 and 4 were used with ranges from 1 to 10 and step 1. The results are given in Table 1.

Table 1. Rejection of repetitive SoPW parameter sets

Number of parameters	Total number of SoPW	Rejected SoPW	Rejected percentage
2	100	36	36
3	1000	158	15.8
4	10000	720	7.2

When the number of parameters is very small (2), the number of rejected variants is as high as 36 per cent. With 4 parameters, the number of rejected variants is 720, which is 7.2 per cent of the generated SoPW. The increased number of parameters generates an increasing number of repetitive SoPW that will not be sent to the Grid and would only overload it if not discarded. This SoPW rejection reduces the number of test tasks with SoPW that are sent to the Grid. This way, it reduces the load on the network with testing tasks and eliminates duplicate SoPW variants with repetitive WoPs.

When the rejection of repetitive SoPW is finished, the way of SoPW self-learning is selected (Figure 2). If the Grid network is large (>100 resources), it is proposed to select self-learning and send tasks with all selected partial brute force SoPW. Since many test tasks are sent to the Grid network during self-learning, it is not recommended to execute self-learning while specifying SoPW. If the structure of the Grid network has changed, SoPW should be specified.

Correction of SoPW values is performed differently when SoPW is determined by the partial brute force method. This correction is performed by forming small SoPW arrays where all SoPW differ by values of the same WoP (Figure 2). Test tasks are sent by the first SoPW array; then, resources are selected with the help of the QoGS method. When the results of test tasks are received, the SoPW, which resulted in the fastest processing of the test task (the shortest time TTD from sending the task until its execution), is selected. During the next iteration, another SoPW array is formed, where SoPW differ by the values of the next WoP, whereas the value of the first parameter in all SoPW coincides with the value of the newly received WoP. These iterations are repeated until all the best SoPW values are set. When the WoP correction algorithm is used, the chances that the best SoPW will not be found are low.

For small Grid networks, self-learning is not suitable if tasks are sent together with the SoPW generated during the application of the partial brute force. A small Grid network would be overloaded with test tasks. Many tasks with different SoPW would be sent to the same resources, making obtaining the most suitable SoPW more difficult. Thus another self-learning way that uses the minimal amount of test tasks is employed for the Grid networks with less than 100 resources. It reduces the amount of both the sent tasks and the workload of the Grid network.

Self-learning uses qualitative weight parameters (ω_W, ω_L, ω_C, ω_N), where values are within the range of [0; 10]. When the number of accessible resources is around 30, using “the minimum amount of test tasks” compared to sending tasks with partial SoPW reduces the number of the tasks by approximately 99.7 per cent (from 9280 to 30). The main advantage of this algorithm is that the number of test tasks sent to the Grid network is equal to the number of resources in the network. The method with minimal test tasks follows this procedure: after receiving the data about test jobs, resources are ranked by TTD of test tasks. This list is considered the benchmark. Next, the computing of the service quality for resources Q (using the SoPW generated by the partial brute force method) and ranking (descending) according to the computed values Q is performed. Based on these values, resources are sorted in descending order. The calculation results are written in a matrix. The best SoPW from the matrix is considered the one that, according to the list of ranked resources, is the closest or even equal to the list of benchmark resources. If several duplicate SoPW are found, the SoPW with the value Q of the resources that best correlates with the time of test tasks TTD is selected.

Deploying these self-learning ways, the best SoPW is determined, recorded into RB and used in the QoGS method. However, if the network structure changes, it is necessary to renew SoPW.

Self-learning, which uses the minimum amount of test tasks, suits well when the users of the Grid services have access (certificate) to a small number of resources that can handle tasks, and they want to obtain the best SoPW by themselves.

Conclusions

A QoGS method has been developed for selecting resources for future tasks in the Grid. Using this new method, a resource is selected based on its qualitative parameters and the assessment of the quality of the service. The method allows shortening the queues and task execution time.

Algorithms for weighting coefficients are proposed. Determining the coefficients allows the selection of the most appropriate resources for the task at hand.

References

1. Sulaiman, M., Halim, Z., Lebbah, M., Waqas, M., Tu, S. (2021). An Evolutionary Computing-Based Efficient Hybrid Task Scheduling Approach for Heterogeneous Computing Environment. Journal of Grid Computing, volume 19, Article number: 11. 552-4.
2. Wang, J., Wang, L. (2021). A Computing Resource Allocation Optimization Strategy for Massive Internet of Health Things Devices Considering Privacy Protection in Cloud Edge Computing Environment. Journal of Grid Computing, volume 19, Article number: 17.
3. Lavanya, M., Shanthi, B., Saravanan, S. (2020). Multi-objective task scheduling algorithm based on SLA and processing time suitable for cloud environment. Computer Communications, Volume 151, 183-195.
4. Mahato, D. P., Sandhu, J. K., Singh, N. P., Kaushal, V. (2020). On scheduling transaction in grid computing using cuckoo search-ant colony optimization considering load. Cluster Computing 23, 1483–1504.
5. Rawat, P. S., Dimri, P., Gupta, P. (2020). Learning-Based Task Scheduling Using Big Bang Big Crunch for Cloud Computing Environment. Recent Advances in Computer Science and Communications, Volume 13, Number 2, 137-146.
6. Ankita, Sahana, S. K. (2020). Evolutionary based hybrid GA for solving multi-objective grid scheduling problems. Microsystem Technologies volume 26, 1405–1416.
7. Gabri, L. R., Agrawal, Y., Srinivas, BK. (2020). A Survey on Grid Computing Scheduling Algorithms. International Journal for Research in Applied Science & Engineering Technology (IJRASET) Volume 8 Issue VI, 731-736.
8. You, M., Luo, W., He, M. (2020). Resource scheduling of information platform for general grid computing framework. International Journal of Web and Grid Services, Volume 16, Issue 3, 254-272.
9. Abualigah, L., Diabat, A. (2021). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Computing volume 24, 205–223.
10. McBride, D., Krznaric, M., Darlington, J., van der Aa, O., Aggarwal, M., Colling, D. (2006). Running a Production Grid Site at the London e-Science Centre. e-Science and Grid Computing. Amsterdam, The Netherlands, 153-153.
11. Sharma, M., Kumar, R., Jain, A. (2020). A Proficient Approach for Load Balancing in Cloud Computing-Join Minimum Loaded Queue: Join Minimum Loaded Queue. International Journal of Information System Modeling and Design (IJISMD) 11(1), 25.
12. Chen, M., Yuan, J., Wang, N., Luo, Y., Luo, P. (2020). Two-Sided Matching Scheduling Using Multi-Level Look-Ahead Queue of Supply and Demand. International Conference on Service-Oriented Computing ICSOC 2020: Service-Oriented Computing, 525-532.
13. Sutiene, K., Vilutis, G., Sandonavicius, D. (2011). Forecasting of GRID Job Waiting Time from Imputed Time Series. Electronics and electrical engineering ISSN 1392 – 1215. No. 8(114).
14. Gao, J., Wang, H., Shen, H. (2020). Machine-Learning-Based Workload Prediction in Cloud Computing. 2020 29th International Conference on Computer Communications and Networks (ICCCN), 1-9.
15. Masdari, M., Khoshnevis, A. (2020). A survey and classification of the workload forecasting methods in cloud computing. Cluster Computing volume 23, 2399–2424.
16. Qionga, S., Zhiyongc, T., Xiaolua, Z. (2020). Workload prediction of cloud computing based on SVM and BP neural networks. Journal of Intelligent & Fuzzy Systems, vol. 39, no. 3, 2861-2867.
17. Gadhavi, L. J., Bhavsar, M. D. (2019). Efficient Resource Provisioning Through Workload Prediction in the Cloud System. Smart Trends in Computing and Communications. Smart Innovation, Systems and Technologies, vol 165. Springer, Singapore, 317-325.
18. Pilkauskas, V., Plestys, R., Vilutis, G., Sandonavicius, D. (2011). Improvement of WMS Functionality, Aiming to Minimize Processing Time of Jobs in Grid Computing. Electronics and electrical engineering ISSN 1392 – 1215. No. 7(113).
19. Vilutis, G., Sandonavičius, D. (2008). The complex evaluation of parameters influence on QoS. ITI 2008: proceedings of the 30th International Conference on Information Technology Interfaces, June 23-26, Cavtat/Dubrovnik, Croatia, 905-910.
20. Vilutis, G., Butkiene, R., Lagzdinyte-Budnike, I., Sandonavicius, D., Paulikas K. (2013). The QoGS Method Application for Selection of Computing Resources in Intercloud. Elektronika Ir Elektrotechnika, 19(7), 98-103.

UŽDUOČIŲ PASKIRSTYMAS SKAIČIUOJAMAJAME GRIDE NAUDOJANT PARAMETRŲ SVORINIŲ KOEFICIENTŲ PRI(SI)TAIKYMĄ

Santrauka

Skaičiuojamieji Grid tinklai plačiai naudojami didelių skaičiavimų uždaviniams spręsti. Tai aktualu sprendžiant mokslines problemas. Straipsnyje pristatomas metodas, kuris sutrumpina į Grid tinklą išsiųstos užduoties rezultato gavimo laiką. Siekiant, kad užduotis Grid tinkle būtų įvykdyta per trumpiausią laiką, labai svarbu žinoti, į kurį resursą ją pasiųsti. Šiame straipsnyje pristatomas į Grid resursų parinkimą orientuotas metodas, padedantis brokeriui parinkti tinkamiausią resursą užduočiai spręsti. Tinkamiausio resurso parinkimui yra taikomas savaiminis pritaikymas, kurio esmė bei rezultatai pristatomi šiame straipsnyje.

Reikšminiai žodžiai: Grid tinklai, debesų tinklai, paslaugų kokybė, išteklių brokeris, savarankiškas parametrų svorių mokymasis.

Informacija apie autorius

dr. Donatas Sandonavičius. Kauno technologijos universiteto Informatikos fakulteto Taikomosios informatikos katedros lektorius. Mokslinių tyrimų kryptys: kompiuterių tinklai, GRID ir debesų tinklai.
El. pašto adresas: donatas.sandonavicius@ktu.lt

Aušra Gadeikytė. Kauno technologijos universiteto Informatikos fakulteto Taikomosios informatikos katedros lektorė. Mokslinių tyrimų kryptys: kompiuterių tinklai, GRID ir debesų tinklai, baigtinių elementų metodai.
El. pašto adresas: ausra.gadeikyte@ktu.lt

dr. Giedrius Paulikas. Kauno technologijos universiteto Informatikos fakulteto Taikomosios informatikos katedros lektorius. Mokslinių tyrimų kryptys: kompiuterių tinklai, GRID ir debesų tinklai, genetiniai algoritmai.
El. pašto adresas: giedrius.paulikas@ktu.lt

dr. Mindaugas Vaitkūnas. Kauno technologijos universiteto Informatikos fakulteto Taikomosios informatikos katedros lektorius. Mokslinių tyrimų kryptys: kompiuterių tinklai, GRID tinklai.
El. pašto adresas: mindaugas.vaitkunas@ktu.lt

dr. Gytis Vilutis. Kauno technologijos universiteto Informatikos fakulteto Taikomosios informatikos katedros docentas. Mokslinių tyrimų kryptys: kompiuterių tinklai, GRID ir debesų tinklai.
El. pašto adresas: gytis vilutis@ktu.lt

Gintaras Butkus. Kauno kolegijos Technologijų fakulteto Informatikos katedros lektorius. Mokslinių tyrimų kryptys: GRID tinklai.
El. pašto adresas: gintaras.butkus@go.kauko.lt