A new approach for dispatching task flows in GRID systems with inalienable resources

Taras A. Uzdenov[1,2]

[1]*Zhytomyr Polytechnic State University, 103 Chudnivska Str., Zhytomyr, 10005, Ukraine*

[2]*G.E. Pukhov Institute for Modelling in Energy Engineering of NAS of Ukraine, 15 General Naumova Str., Kyiv, 03164, Ukraine*

Abstract. This paper presents a new approach for solving the problem of dispatching task flows with known complexity in GRID systems that have inalienable resources with determined performance. The proposed method is simple to implement and is compared with the commonly used FCFS method. An example of a practical problem that can be solved using this method is provided.

Keywords: GRID systems, dispatching task flows, inalienable resources, performance, complexity, FCFS method, practical problem, task scheduling

1. Introduction

Every modern organization has a number of computers on which its staff works, their use is not the most efficient, as most of them tasks performed on them do not take up 10–20% of the maximum performance of the PC. Therefore, it makes sense to use free resources to solve other problems.

Therefore, the idea arose to create on the basis of such resources computer systems that would allow other tasks to be performed in parallel with the current ones for each of the nodes. Such systems are called GRID systems with non-alienable resources. Such systems are also known as Desktop GRID.

Thus, Desktop GRID is a GRID system that uses non-specialized computing resources as computing nodes, but disparate computing nodes (computers, laptops, smartphones, etc.) using local and global networks and special software.

Back in 1999, the first large-scale project of distributed voluntary computing SETI @ home was launched. Today, Desktop Grid is part of the high-performance computing industry along with clusters and GRID.

One of the main tasks in creating a GRID system with inalienable resources, as well as for GRID systems, is the task of task scheduling. Therefore, in GRID systems, a planning mechanism must be implemented. It is necessary for the distribution of tasks for execution between the nodes of the system, in order to minimize the execution time and balance the load of the system.

One of the problems that needs to be solved when developing software for a GRID system with inalienable resources is the task of task scheduling. Task scheduling is quite complex and now there is no clear and unambiguous solution [8, 12]. Analysis of scheduling methods used in real systems shows that most systems use mainly the FCFS method [2]. In addition, were analyzed a number of publications in recent years on this topic , in which developers offer new methods and various modifications of known ones. They are compared with FCFS and SJF (Shortest Job First) [1, 3, 5, 6, 9, 11, 13, 15, 16]. But since in real systems the FCFS method is most often used, it was decided to make a comparison with it in this study as well.

This is due to the fact that this method is very simple and reliable in both development and operation. The use of other methods significantly complicates the system, making it less reliable. Since such systems are quite unstable, it is clear why developers abandon complex methods and prefer FCFS. This leads to the conclusion that it is necessary to develop new methods, the main characteristics of which should be simplicity and better performance compared to FCFS.

2. New approach

This article further studies the methods developed on the basis of the new approach outlined in [17]. In particular, a simple practical task that could be performed on a GRID system with non-alienable resources is considered, and the FSA method is used to distribute tasks between nodes.

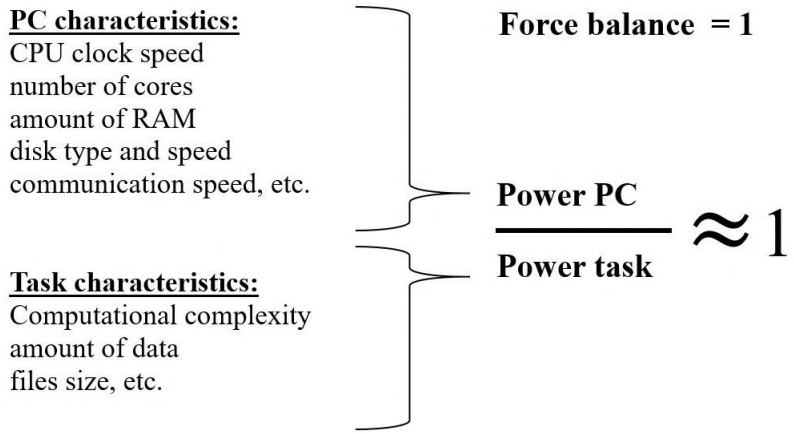

Figure 1: New approach.

In figure 1 schematically shows the main essence of the proposed in [17] approach. The proposed methods are developed on the basis of a new approach, which proposes to consider tasks as one force, and the nodes on which they should be performed as another force, like Newton's third law. And distribute the tasks in such a way as to maintain a balance of forces. Given that such concepts as the strength of the task and the strength of the node are quite abstract concepts, the author proposes to use the concepts of task power and node power. And to carry out distribution already according to balance of capacities.

This choice is not accidental, and can be explained as follows. It is known from physics that power is equal to the ratio of work to time. And the work, in turn, is equal to the force

multiplied by the path to be traversed. Given that when performing a task on a computing node, the time to determine the power of the task will be equivalent to the time to determine the power of the node, and the path that the task must pass is equal to the path that the node must pass, such a replacement is quite logical and acceptable. In the developed methods, these concepts are quantities relative and therefore can be calculated in different ways, depending on different characteristics of tasks (volume, computational complexity, algorithmic complexity, etc.) and different characteristics of nodes (CPU clock speed, RAM volume, communication channel speed and others). The power of the task means the totality of all the characteristics of the task, and the power of the PC means the totality of all the characteristics of the PC, compiled in some way. Moreover, if for a PC it is still possible to use some general formula for calculation, then for the power of the problem such a formula will change each time, depending on the type of problem that needs to be solved.

The approach described above is quite simple and effective to use, but it actually divides the scheduling task into three subtasks:

1. Calculation powers of tasks
2. Calculation powers of nodes
3. Distribution (FSA method)

Both the first and the second subtasks are quite complex and today there are no unambiguous and universal solution for they. The fact is that any task has a number of characteristics, which have already been written above, and to compare them and somehow reduce to one value is quite difficult.

This requires the development of additional methods that would provide such an opportunity. On the other hand, the task of calculating the powers of nodes is no less complex and also requires a separate study and solution. But at the same time there are a number of tasks for which the calculation of capacity will not cause much difficulty. This is well illustrated by the example of a simple practical problem described in the last section.

2.1. Formulation of the problem

Suppose there is a GRID system with $\square$ task and $\square$ nodes. By nodes we mean a computational element. We introduce the concept of task power $\square_{\square}$ and node power $\square_{\square}$. Therefore, we have the set of power of tasks $\square = \{\square_1, \square_2, \square_3, \ldots, \square_{\square}\}$ and the set of power of nodes $\square = \{\square_1, \square_2, \square_3, \ldots, \square_{\square}\}$. We need to optimally distribute tasks across nodes.

Schematically, a given task is shown in figure 2.

2.2. Flow Scheduling Algorithm (FSA)

The method of flow scheduling has the following form:

1) calculate the power of tasks and the power of nodes
2) choose the $\square$-th task
3) find the pair $\square - \square$, for which the ratio of the power of the task $\square_{\square}$ to the power of the node $\square_{\square}$ will be as close as possible to unity
4) send the $\square$-th task to the $\square$-th node

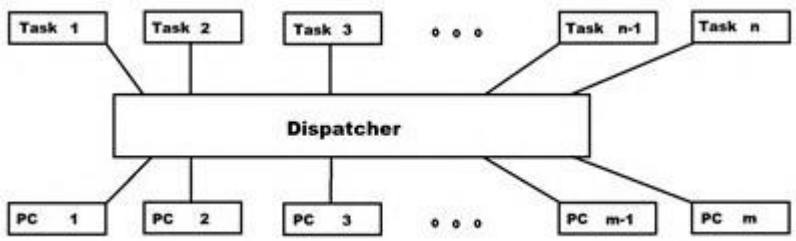

Figure 2: Scheduling task.

5) recalculate the power without □□ and □□
6) if there are unsent tasks, then go to point 2
7) completion

2.3. Flow Scheduling Algorithm Parallel (FSA_P)

The described approach is universal and allows to develop not only methods for distribution of consecutive tasks but also for tasks which can be parallelized. Below is the following method:

1) calculate the power of the nodes, and the total power □□□□
2) select the □-th task from the task queue
3) distribute it proportionally, according to the power of the nodes
4) send for execution
5) if there are unallocated tasks in the queue, go to point 2
6) completion

3. SoPware package

To create software that allows you to test and investigate the effectiveness of the proposed methods, developed its client-server architectural model (figure 3).

This model is based on the WCF service [10] for software that requires distributed computing in computer networks and the Internet, as well as to create a Desktop GRID.

Based on this model, a software package for simulating the operation of a GRID system with inalienable resources and the study of scheduling algorithms was developed, which was named SgridAR-1 [18].

A number of tests were performed using the developed simulator. To determine their effectiveness, the FCFS scheduling method was chosen because it is most often used for GRID systems with inalienable resources. In addition, other methods cannot be used because in this case they do not involve a change in the power of the nodes, and other characteristics, such as time quantum or priority, it was decided not to enter.

Figure 4 highlights nine areas in the Server window, which during testing displays information about the results of the simulation:

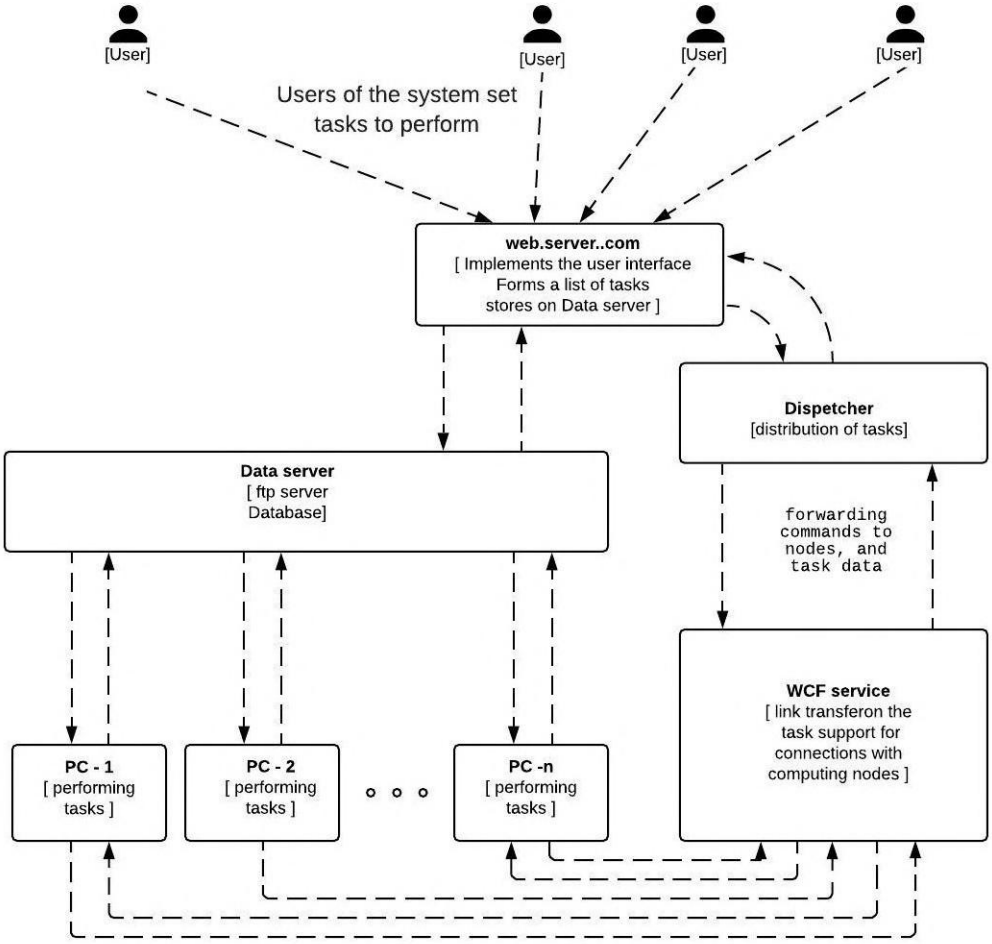

Figure 3: Architectural model of Desktop GRID.

1) the total volume of all tasks to be solved
2) total execution time of all tasks for each algorithm
3) the area in which messages about the beginning and end of calculations are displayed, as well as information about which algorithm is currently working
4) in this area the conditions for testing are set: the number of tasks, their minimum and maximum value, the choice of algorithm
5) list of all generated tasks, their scope and power, status (performed or not), execution time for each algorithm
6) after completing the test, this area receives summary information about the operating time of each algorithm, after which in area 2 the information is updated to perform a new test, and the button "Export to Excel" becomes active
7) the number of Clients to run

8) the maximum and minimum value of the power of the system nodes, after pressing the "Change Power PC" button, the system will automatically change the power in any way for all nodes, in the range from minimum to maximum values
9) information about the nodes connected to the system, their capacity and status (free or busy)

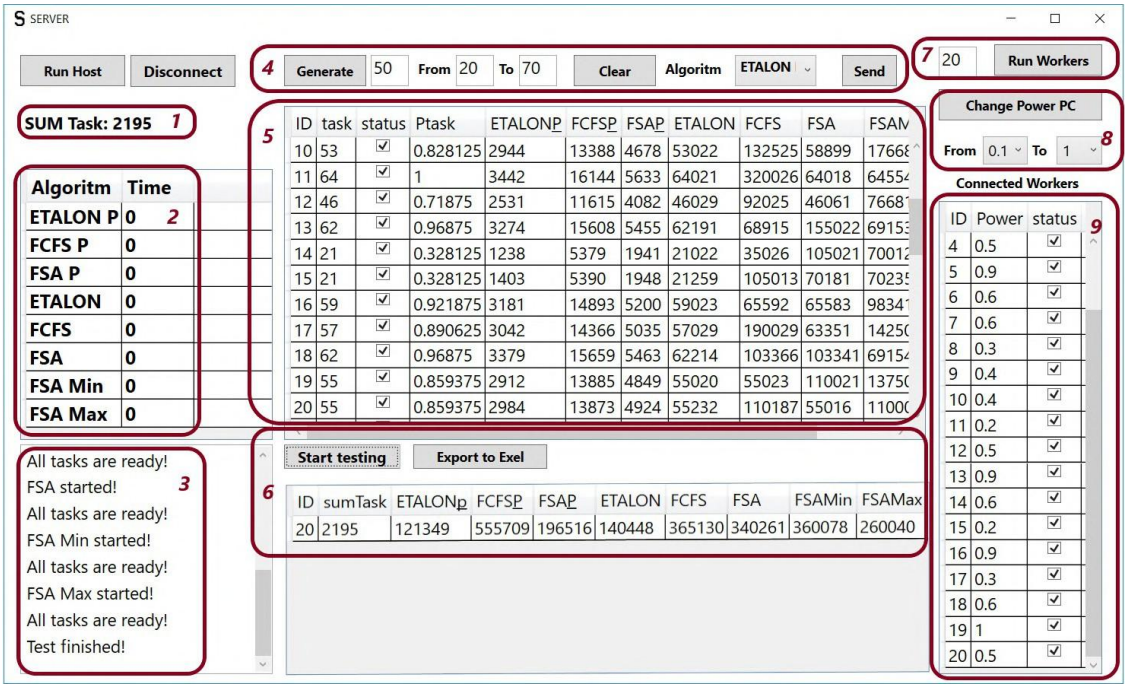

Figure 4: SGridAR-1 server window.

SGridAR-1 allows to show work of the offered method and to compare results of its work with other, well-known, methods of dispatching. With the help of this software you can conduct experiments and explore the work of algorithms, changing the number and size of tasks, the number and power of the PC.

In this program the mechanism of generation of tasks was implemented. The task of which is to create a tasks to be executed in an arbitrary way, specifying the time required for the task to be performed. Power of tasks are calculated in proportion to the given time. The power of the same nodes is also generated arbitrarily. But depending on their size slow down the timer.

The implemented visual interface clearly shows how the GRID system works. This set of programs can be used not only for research but also for the educational process.

4. Test results

The SGridAR-1 system implements a testing mechanism that provides for the execution of all tasks generated by the system, using different algorithms for the distribution of tasks between nodes. That is, first the tasks are distributed according to algorithm 1, then according to algorithm 2 and so on until the last task.

The following scheduling methods are introduced into the system: ETALON_P; FCFS_P; FSA_P; ETALON; FSA, FSA_Min, FSA_Max.

Parallel methods:

- ETALON_P – the method is taken as a reference. This is a FCFS method that does not take into account the power of the personal computer (PC) and tasks. A system is modeled in which all nodes are the same in power. Parallel tasks are possible
- FCFS_P – FCFS method, which takes into account the power of the PC and the ability to distribute one task to several PCs
- FSA_P is a proposed method that takes into account the power of PCs and the ability to distribute one task to multiple PCs

The following methods:

- ETALON – the method is taken as a reference. This is a FCFS method that does not take into account the power of the PC and the task cannot be distributed
- FCFS – a method that takes into account the power of the PC and the task can not be distributed between nodes
- FSA – the proposed methods, which take into account the power of the PC and the task can not be distributed
- FSA_Min – modified FSA method combined with Min-Min method [7]
- FSA_Max – modified FSA method combined with Max-Min method [14]

For comparative experiments, 100 tasks with a runtime of 1 to 10 seconds were generated and 10 nodes were run on which they should be executed. Testing was as follows: first, the calculation of the total execution time of all tasks for 1 method, entered into the system, at an average power of the system 10%, then programmatically increased the system power by 5% and again performed calculations for 1 method. Then again increased the power of the system by 5% and performed the calculation by 1 method. And so on until the power of the system has grown to 100%. Thus, testing was performed for each of the methods. Each time performing the same tasks, changing only the average power of the system.

A total of 19 average power tests were performed for each of the planning methods introduced into the system. A total of 152 tests were conducted.

Such studies were conducted to show how the average power of the system affects the efficiency of the methods.

In figures 5-7 are diagrams based on test results. Based on the results obtained, it can be concluded that the FSA and FSA_P methods give better results than the FCFS and FCFS_P methods.

As can be seen in figure 5, the curve of the FSA_P method has a smooth shape, and the curve of the FCFS_P method is broken. This allows us to conclude that the FSA_P method is well predictable and, knowing the power of the system and the amount of tasks, it is possible to predict the completion of calculations. However, solving problems by the FCFS_P method, it is quite difficult to predict the completion of calculations, because a lot depends on which node, which task will be performed.

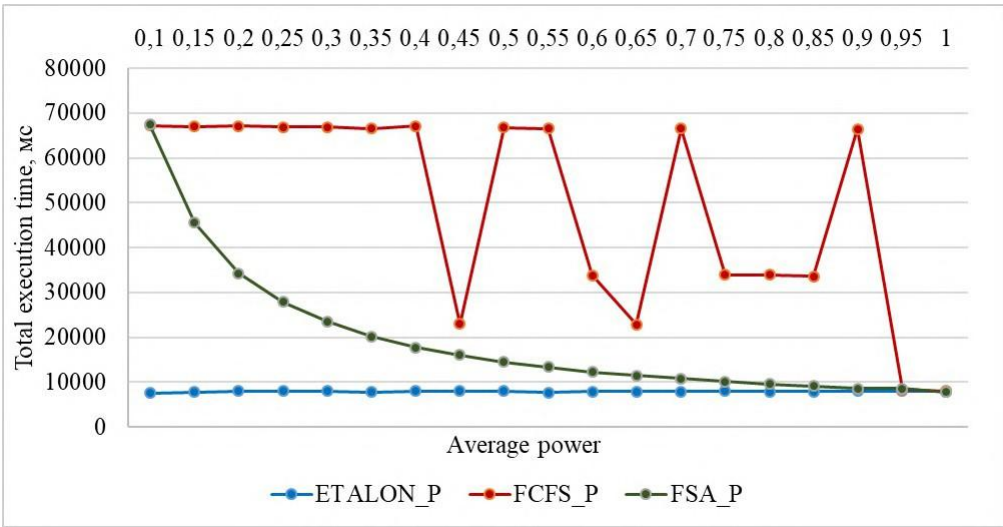

Figure 5: Test results for parallel methods.

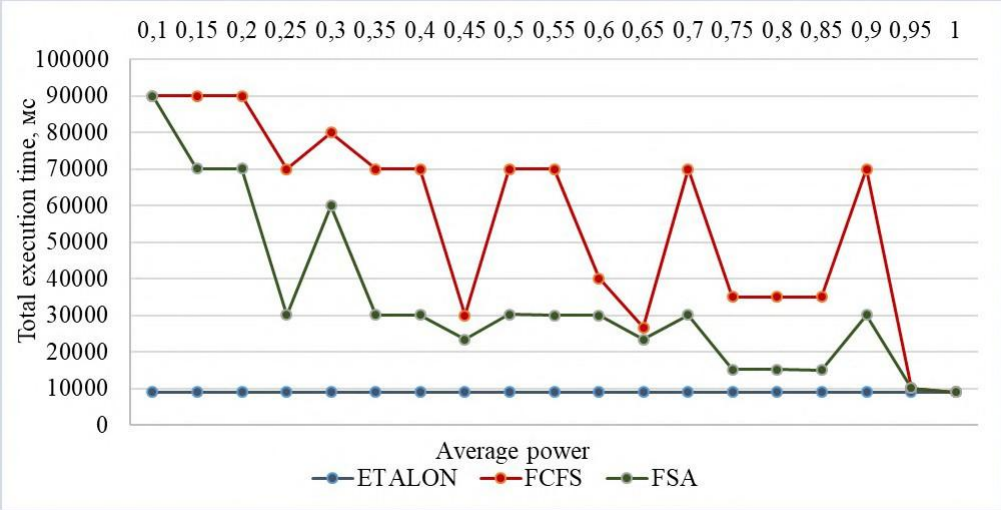

Figure 6: Test results for following methods.

In figure 6, it is seen that the sequential FSA method has a smoother curve than the FCFS method, which also indicates a better predictability.

In figure 7, which shows additional methods FSA Min and FSA Max in comparison with the FSA method, it is seen that the differences in the results are insignificant, but this is only for the situation when the number of tasks exceeds the number of nodes.

As can be seen from the graphs, the system also has ETALON and ETALON_P methods, and at first glance it may seem that their efficiency is much better than the FSA and FSA_P methods. But this is not the case. The fact is that the methods ETALON and ETALON_P are shown not for comparison with others, but to demonstrate the reference state of the system,

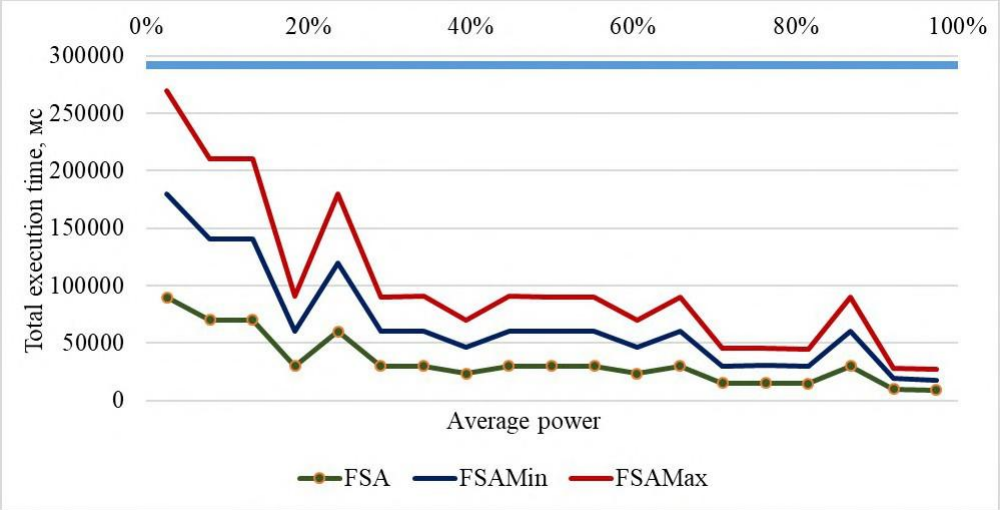

Figure 7: Test results for methods FSA, FSA_Min, FSA_Max.

when all the nodes in it have a capacity of 100%. As can be seen from the graphs, when the average power increases to 1, the execution time by different methods approaches the maximum possible reference value.

Figures 5, 6 show that the behavior of the FCFS method is very unstable, large jumps in the results. This is because the distribution by the FCFS method very much depends on a random factor, and which node will have what task. This is quite logical, because if, for example, a node with a capacity of 0.1 receives a task with a capacity of 1, then the execution time in this case will increase significantly, because there may be a situation that other nodes will work and will wait too long to complete this task.

5. Practical task

It is proposed to consider such a task. If, for example, we have a web resource with a large number of images (for example, a portfolio, an online store, etc.), and we want to promote it in search engines in order to attract more users and increase revenue, then we need will perform page optimization according to search engine rules (for example, using Google PageSpeed [4]), which includes image optimization.

In figure 8 schematically shows the problem described above. Let's say we have 1000 images that need to be optimized according to Google Page Speed. Total files size is 5 GB. If you perform this task on one PC, it will take a long time (maybe even a full day). It all depends on the speed of the PC on which to do it.

Therefore, to speed up this task, it would be advisable to divide it into different PCs that we have. To do this, we will need to determine which method will be used for distribution.

In fact, as mentioned above, there is a new method of FSA, based on the described approach. The main difficulty of this method is to calculate the power of tasks and nodes. But for this practical example, this problem is quite simple to solve. Given that all images differ from each

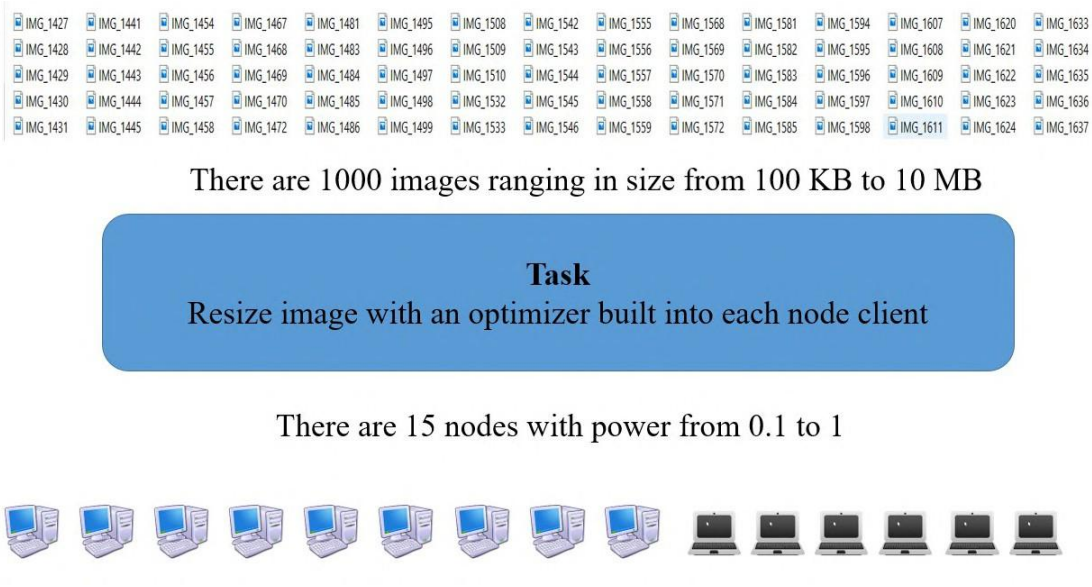

Figure 8: Practical task.

other only in size, the largest image ($\square_{\square\square\square}$) is assigned power ($\square_{\square\square\square} = 1$), and the remaining ($\square - 1$) images are calculated by the power of the tasks by the formula:

$$\square_{\square} = \frac{\square_{\square} \cdot \square_{\square\square\square}}{\square_{\square\square\square}}$$

Given that ($\square_{\square\square\square} = 1$) we can shorten the expression:

$$\square_{\square} = \frac{\square_{\square}}{\square_{\square\square\square}} \tag{1}$$

On the other hand, we have $\square$ PCs (computing nodes), which differ from each other, for example, only the clock speed of the processor. Then, similarly to the calculation of the power of the problem, we can find the power of the nodes, according to the proportion.

The largest by the clock frequency of the node ($\square_{\square\square\square}$) is assigned power ($\square_{\square\square\square} = 1$), and the remaining ($\square - 1$) power of the nodes is calculated by the formula:

$$\square_{\square} = \frac{\square_{\square} \cdot \square_{\square\square\square}}{\square_{\square\square\square}}$$

Given that ($\square_{\square\square\square} = 1$) we can shorten the expression:

$$\square_{\square} = \frac{\square_{\square}}{\square_{\square\square\square}} \tag{2}$$

Then the algorithm of the method of scheduling FSA tasks can be reduced to the following form:

1) calculate $\square = \square_1, \square_2, \square_3, \ldots, \square_{\square}$ by formula (1)

2) calculate $\square = \square_1, \square_2, \square_3, \ldots, \square_\square$ by formula (2)
3) choose the $\square$-th task
4) find the pair $\square - \square$, for which the condition

$$\min \left\{ \left| \frac{\square}{\square_\square} - 1 \right| \right\}$$

is fulfilled.
5) send the $\square$-th task to the $\square$-th node
6) recalculate the powers without $\square_\square$ and $\square_\square$
7) if there are unsent tasks, then go to point 2
8) completion

Thus, we showed that calculating the power of tasks and the power of nodes is not such a difficult task when you need to distribute the image between nodes in order to optimize the size. If you use different nodes, but different in speed of connection to the network, then the power of the nodes must be calculated in some other way, because not always the node with the processor with the highest clock speed will have the highest power. And if this point is not taken into account, then the distribution will not be as effective as in the first case.

As mentioned above, this method is universal and the distribution of tasks according to it makes it possible to significantly speed up the execution of the task queue and thus increase the efficiency of GRID systems with inalienable resources compared to the FCFS method.

6. Conclusions

The results of the study of the effectiveness of the proposed methods showed that their use for task allocation in GRID systems with non-alienable resources provides a significant reduction in task queue time compared to the FCFS scheduling method, provided that the number of tasks exceeds the number of nodes.

All the proposed methods are quite stable and well-predicted, which means that their use in GRID systems will give advantages not only in time and performance, but also allow more efficient planning of the system work. The FCFS method works well, but for GRID systems that have different power resources, its performance depends heavily on a random fact that can be considered a significant drawback.

The main difficulty in using these methods is the need to somehow reduce all the characteristics of tasks and nodes to one relative value. But there are a number of tasks for which this can be done quite easily. As shown in the example of the problem of image optimization, it is quite easy to calculate both the power of tasks and the power of nodes.

References

[1] Carastan-Santos, D., De Camargo, R.Y. and Trystram, D., 2019. One can only gain by replacing EASY Backfilling, a simple scheduling policies case study. *19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, pp.1–10. Available from: https://doi.org/10.1109/CCGRID.2019.00010.

[2] Choi, S., Kim, H., Byun, E. and Hwang, C., 2006. *A Taxonomy of Desktop Grid Systems Focusing on Scheduling*. KU-CSE-2006-1120-02.

[3] Dheenadayalan, K., Muralidhara, V.N. and Srinivasaraghavan, G., 2016. Storage Load Control Through Meta-Scheduler Using Predictive Analytics. In: N. Bjørner, S. Prasad and L. Parida, eds. *Distributed Computing and Internet Technology*. Cham: Springer International Publishing, pp.75–86.

[4] Google Docs, 2021. PageSpeed Insights. Available from: https://developers.google.com/speed/docs/insights/v5/about.

[5] Haruna, A.A., Jung, L.T. and Zakaria, N., 2015. Design and Development of Hybrid Integrated Thermal Aware Job Scheduling on Computational Grid Environment. *2015 International Symposium on Mathematical Sciences and Computing Research*. pp.13–17. Available from: https://doi.org/10.1109/ismsc.2015.7594020.

[6] Kaur, M., 2017. Multi-objective Evolution-Based Scheduling of Computational Intensive Applications in Grid Environment. *Proceedings of the International Conference on Data Engineering and Communication Technology*. pp.457–467. Available from: https://doi.org/10.1007/978-981-10-1678-3_44.

[7] Kokilavani, T. and Amalarethinam, D.I.G., 2011. Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. *International Journal of Computer Applications*, 20(2), pp.43–49.

[8] Kropyvnytska, V.B., Klim, B.V., Romanchuk, A.G. and Slabinoga, M.O., 2011. Investigation of scheduling algorithms in computer systems. *Rozvidka ta rozrobka naftovykh i hazovykh rodovyshch*, 2(39), pp.93–105.

[9] Kumar, P.S., Parthiban, L. and Jegatheeswari, V., 2019. Privacy and security issues in cloud computing using idyllic approach Latha Parthiban. *Networking and Virtual Organisations*, 21(1), pp.30–42. Available from: https://doi.org/10.1504/IJNVO.2019.101146.

[10] Microsoft Docs, 2021. *What Is Windows Communication Foundation — WCF*. Available from: https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf.

[11] Naithani, P., 2018. Genetic Algorithm Based Scheduling To Reduce Energy Consumption In Cloud. *2018 Fifth International Conference on Parallel, Distributed and Grid Computing*. IEEE, pp.616–620. Available from: https://doi.org/10.1109/pdgc.2018.8745801.

[12] Node, S., 2021. Desktop grids, connecting everyone to science. Available from: https://sciencenode.org/feature/desktop-grids-connecting-everyonescience.php.

[13] Pujiyanta, A. and Nugroho, L.E., 2018. Planning and Scheduling Jobs on Grid Computing. *2018 International Symposium on Advanced Intelligent Informatics*. IEEE, pp.162–166. Available from: https://doi.org/10.1109/icic47613.2019.8985978.

[14] Ramyachitra, D. and Kumar, P.P., 2016. Frog leap algorithm for homology modelling in grid environment. *Journal of Emerging Technologies and Innovative Research*, 7(1).

[15] Sahana, S., 2019. Evolutionary based hybrid GA for solving multi-objective grid scheduling problem. *Microsystem Technologies*. Available from: https://doi.org/10.1007/s00542-019-04673-z.

[16] Thet, Y., Hlaing, H. and Yee, T.T., 2019. Static Independent Task Scheduling on Virtualized Servers in Cloud Computing Environment. *2019 International Conference on Advanced Information Technologies*. IEEE, pp.55–59. Available from: https://doi.org/10.1109/aitc.2019.8920865.

[17] Uzdenov, T., 2021. A New Task Scheduling Algorithm for GRID Systems with Non-alienable Resources. In: A. Zaporozhets and V. Artemchuk, eds. *Systems, Decision and Control in Energy II*. Cham: Springer International Publishing, pp.207–220. Available from: https://doi.org/10.1007/978-3-030-69189-9_12.

[18] Uzdenov, T., 2021. Simulator of Task Sheduling in Geographicaly Distributed Computer systemswith Non-Alienable Resources. *Electronic Modeling*, 42(1).