MODELING OF TASK SCHEDULING IN DESKTOP GRID SYSTEMS AT THE INITIAL STAGE OF DEVELOPMENT

I.I. Kurochkin [1,2,a], E.A. Gerk [2,b]

[1] *Institute for Information Transmission Problems of Russian Academy of Sciences, Bolshoy Karetny per. 19, build.1, Moscow, 127051, Russia*

[2] *The National University of Science and Technology MISiS, Leninskiy prospekt 4, Moscow, 119049, Russia*

E-mail: [a] kurochkin@iitp.ru, [b] gerkevgeny@gmail.com

The paper presents an overview of modern methods for task scheduling in desktop grid systems, estimates of the quality of methods, including: the time of execution of all tasks, the level of resource utilization. Heuristic approach to task scheduling is considered, which allows ensuring high performance and reliability of such systems at the early stages of development. A comparative analysis of the results of computational experiments performed with the help of the GridSim high performance computing simulation tool for various desktop grid system configurations is carried out.

Keywords: desktop grid, grid system, task scheduling, GridSim, simulation of desktop grid

1. Introduction

The desktop grid is a grid system that unites personal devices (personal computers, laptops, smartphones, etc.) using telecommunications networks and uses their idle computing resources for calculations. In the early 2000s, the desktop grid implied the unification of only personal computers. Today, the desktop grid is a heterogeneous system that combines not only various non-specialized computing devices, but also multiprocessor computing systems with a large amount of hardware and software heterogeneity. The idea of recycling idle resources of a large number of personal computers is extremely attractive. Such a solution, which requires almost no costs, has enormous potential. The technology of organization desktop grid has the following advantages: ease of deployment; low cost; high scalability (hundreds of thousands computing nodes); high potential peak performance; low financial cost of creation and maintenance. But there are also a number of features characteristic of such grid systems: high hardware and software heterogeneity; lack of information about the availability of nodes; low reliability of nodes. And the simultaneous provision of energy efficiency, safety, reliability and high performance is a pressing issue for modern grid systems. These goals conflict with each other and require increased attention to the problem of scheduling tasks at the desktop grid.

2. Overview of existing task scheduling algorithms

In papers [1] [2], a wide range of heuristics and algorithms were proposed for solving separate scheduling problems in grid systems. These heuristics use additional information, such as reliability ratings, estimation of availability periods, node failure functions, etc. The MET (Minimum Execution Time) algorithm proposed in [3] assigns each task in an arbitrary order to the compute node with the minimum time to complete this task without regard to its availability. Ignoring the availability of a computational node when assigning tasks (jobs) to it can lead to a load imbalance in the grid system. In turn, MCT (Minimum Completion Time), also proposed in [3], assigns each task in an arbitrary order to the compute node with the earliest completion time for this task (the sum of the compute node ready time and the task execution time at the computation node), thereby taking into account the current resource load when assigning the next task. This algorithm allows you to balance the load on the computational nodes of the grid system, but at the same time leads to the execution of tasks on less fast computational nodes. The Min-min algorithm, described in [4], selects the machine with the minimum completion time and assigns the task according to the MCT. The main difference between this algorithm and the MCT algorithm is that Min-min considers all unselected tasks while making an assignment decision and MCT considers only one randomly selected task. This advantage allows you to build a more efficient map of task performance, comprehensively assessing the performance of the MCT algorithm. The Max-min algorithm [4], in contrast to the Min-min algorithm, selects the machine with the minimum completion time, and the task with the maximum completion time. The RASA heuristic uses the Min-min algorithm to assign the first task (job), if the number of available resources is odd, otherwise it uses the Max-min algorithm. Further, sequential alternation of the specified algorithms is applied when assigning a task to the available nodes of the system. Heuristics LBMM (Load Balanced Min-Min) allows to reduce the total time to complete all tasks and increase the degree of utilization of resources.

3. Proposed approach to scheduling tasks

The proposed approach to scheduling jobs – Sort-Median (SM). The main objective of the proposed heuristics is to reduce the overall execution time of all tasks and increase the level of utilization of the resources of the grid system. Let $T=\{t_i : i=1, 2, \ldots, n\}$ – a set of computing tasks in the desktop grid, with each task t_i has a length $length_{ti}$ in FLOP. Let $H=\{h_j : j=1, 2, \ldots, m\}$ – a set of computational nodes, with each node h_j have performance $power_{hj}$ in FLOPS. Then the schedule for performing the set of tasks T on the set of computing nodes H of the system is determined by the following expression:

$$S = \{s_{ij} : i \in N, j \in M\}, \qquad (1)$$

where N is the set of numbers of computing tasks to be performed on the nodes of the grid system, according to schedule S, and M is the set of numbers of computational nodes to which these tasks will be assigned. We will also define a matrix in which we will store estimates for the completion of tasks (Completion Time, CT):

$$c_{ij} = e_{ij} + r_j, \qquad (2)$$

where e_{ij} is the task execution time t_i on the h_j node, r_j is the release time of the h_j resource.

The description of this algorithm can be divided into the following steps:

1. CT matrix initialization — for each task, the execution time on each computing node is estimated;
2. bypassing a set of tasks is performed if it is not empty:
   - 2.1. The CT matrix is sorted and the SCT (Sorted Completion Time) matrix contains the medians in rows (for each computational task) – this characteristic, unlike the average value, reflects how “expensive” the miscalculation of a particular task will cost, if it is not counted first;
   - 2.2. assigns the task t_k with the maximum median to the computation node h_p with the minimum completion time;
   - 2.3. the r_p value is updated, as well as the column with the index p of the matrix CT;
   - 2.4. removes the t_k task sent for rendering from the set.

Unlike the previously proposed Min-Min or Min-Max algorithms, this algorithm allows you to get a comprehensive assessment when assigning tasks to a computational node, namely, to show the degree of efficiency of calculating a particular task in the first place.

4. Grid system modeling tools

To model distributed computing systems, a special class of software is used. Depending on the method of reproducing the processes of functioning of distributed systems, these tools are divided into two classes: emulators and simulators. Simulators include the following systems: OptorSim [5], GridSim [6], WorkflowSim, SimGrid, etc. These systems provide an opportunity to obtain statistical data on the most important characteristics of the simulated environment. The GridSim platform allows users to simulate the operation of a grid system with the ability to simulate the characteristics of resources and computer networks in various configurations. With the help of GridSim, it is possible to carry out reproducible experiments that are difficult to implement in the present environment of dynamic grid systems.

5. Results and discussions

As part of the studies conducted on the basis of the GridSim software, an application was developed for modeling the operation of the grid system and testing the tasks scheduling algorithms. The following modeling scenarios were highlighted (Table 1), the main task of which is to evaluate the effectiveness of the proposed approach to scheduling in comparison with existing algorithms according to the above criteria and the behavior of the grid system at the early stages of its development, when the number of computing nodes is sufficiently small.

Table 1. Scenarios for desktop grid simulation

Number	Number of computational nodes	Number of tasks	Performance of computational nodes ($GFLOPS$)	Complexity of tasks ($TFLOP$)
1	10	1000	N(50,10)	N(700, 50)
2	50	6000	N(50,10)	N(700, 50)
3	100	15000	N(50,10)	N(700, 50)

For a comparative analysis of the proposed scheduling algorithm with existing ones, we introduce two quality assessments: the level of utilization of the resources of the grid system (Grid Utilization, GU) and the total execution time (Makespan). Let the execution time of all tasks:

$$Makespan = \max\{r_j | \forall\ 1 \le j \le m\}. \tag{3}$$

Then the level of utilization of the resources of the grid system:

$$GU = \frac{\sum_{j=1}^{m} r_j}{m \times Makespan}. \tag{4}$$

The proposed quality estimates were calculated for the following scheduling algorithms: Max-Min, Min-Min, RASA, Sort-Median. The diagram of the total task execution time (Figure 1) shows the advantage of the proposed Sort-Median heuristics for all grid modeling scenarios.

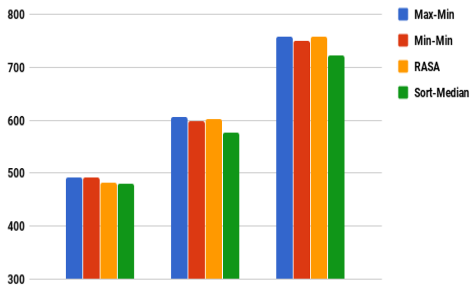

Figure 1. Total execution time

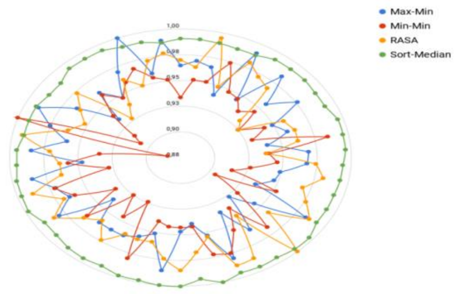

Figure 2. Resource utilization chart on nodes

When scaling the grid system and increasing the number of tasks, it was possible to reduce the total time to complete all tasks. The indicators of the resource utilization diagram (Figure 2) show a significant advantage of the proposed scheduling algorithm, which allows you to evenly distribute the load among all the computational nodes of the system. Despite the fact that the Sort-Median algorithm has improved the performance of the grid system and increased the level of resource utilization, using it in the proposed form for scheduling tasks on a real grid system is unacceptable. This approach allows to take into account the variation in the performance of resources and the complexity of computing tasks, but does not take into account the possibility of failure of the computational nodes, as well as their return of erroneous results. These restrictions make it difficult to complete tasks on time, they require additional time spent on re-execution of tasks. Therefore, it is necessary to modify the Sort-Median algorithm to ensure the reliability of the calculations. Since the planning of tasks is considered at the early stages of the development of the grid system, when there are no statistics on the activity of computational nodes, reputational methods lose their relevance. The use of various prediction heuristics is also difficult, since it is difficult to estimate the number and reliability of computational nodes that can join or leave the system at any time. Therefore, premature scheduling of all tasks on all compute nodes is the wrong approach. It is advisable to use short-term planning: break up the set of all tasks into small subsets and make a decision about planning a specific subset of tasks after returning a certain percentage of correct results. This approach will allow assessing the state of computational nodes set at each scheduling iteration and make a decision on the assignment of tasks.

Let the size of each subset be equal to the number of nodes in the grid system, and each new iteration of short-term scheduling occurs when returning 35% of the valid results of the subset. It is proposed to calculate the delay for each node:

$$d_j = \frac{\sum_{i=1}^{q}(e_{ij} - a_{ij})}{q}, \tag{5}$$

where a_{ij} is the actual time of execution of the task t_i on the h_j node, and q is the number of correctly calculated tasks by the h_j node. Then we introduce two types of penalties for each node: for delay and for failure (failure or incorrect answer). The size of the penalty for failure for each computing node is determined by the expression:

$$p_j = \sum_{i=1}^{g} e_{ij}, \tag{6}$$

where g is the number of failures and incorrect answers for the h_j node. Define the resulting matrix as a weighted sum:

$$f_{ij} = c_{ij} + w(d_j + p_j), \tag{7}$$

The weighting factor w is the proportion of correct calculated results from all tasks. Accordingly, as the number of tasks not counted decreases, the importance of the above penalties increases during planning. The modified heuristics of the algorithm Sort-Median called PSM (Penalty Sort-Median). Let us determine the characteristics of scenarios with the presence of failures in the operation of the grid system for comparing the heuristics of Sort-Median and PSM. The simulated percentage of failover computing nodes will vary from 20 to 35. The results show a significant decrease in the total runtime of the script (about 60%) when using short-term planning and the PSM algorithm in comparison with the Sort-Median algorithm. The main disadvantage of this solution is a decrease in the level of utilization of resources due to an increase in their idle time.

6. Conclusion

An approach using Sort-Median heuristics for scheduling tasks in desktop grid is proposed. A comparative analysis of the algorithms showed the advantage of the proposed approach. Sort-Median utilizes more than 99% of the grid system's resources in all cases, and also improves the performance of the grid system by 3%. Penalty Sort-Median (PSM) heuristic was also proposed using short-term scheduling, which is important for desktop grids at the early stages of their development.

7. Acknowledgement

This work was supported by the Russian Foundation for Basic Research (grants No. 18-29-03264, 18-57-06003).

References

[1] Casanova H. F. Dufossé, Y. Robert, F. Vivien. Scheduling parallel iterative applications on volatile resources // Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International. – IEEE, 2011. – pp. 1012-1023.

[2] Wang X. Ch. Sh. Yeo, R. Buyya, J. Su. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm // Future Generation Computer Systems. – 2011. – Vol. 27. – № 8. – pp. 1124-1134.

[3] Freund R. F. Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet //Heterogeneous Computing Workshop, 1998. (HCW 98) Proceedings. 1998 Seventh. – IEEE, 1998. – pp. 184-199.

[4] Anousha S., Ahmadi M. An improved Min-Min task scheduling algorithm in grid computing // International Conference on Grid and Pervasive Computing. – Springer, Berlin, Heidelberg, 2013. – pp. 103-113.

[5] Korenkov, V. V., & Nechaevskiy, A. V. (2009). DataGrid simulation packages. System Analysis in Science and Education (Online), ISSN, 2071-9612, pp.21-35.

[6] Buyya R. Gridsim – Toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing / R. Buyya, M. Murshed // Concurrency and computation: practice and experience. – 2002. – Vol. 14. – pp. 1175-1220.