Australian Journal of Basic and Applied Sciences, 6(10): 1-10, 2012
ISSN 1991-8178

Scheduling In Heterogeneous Grid-Systems

Hussein. A. Al-Ofeishat

Computer Engineering Department, Al-Balqa Applied University, Jordan

Abstract: We propose a hierarchical scheduling method in GRID- system, which will allow us to achieve higher efficiency of scheduling through considering the number of physical links in the processor's nodes, the possibility of supporting communication channels processors of duplex mode, the proposal of the structure and algorithms functionality of the central and local brokers of GRID-systems, and Carrying out a comparative analysis of the proposed scheduling method with known methods. Analyzing the main scheduling methods in Grid-systems, the hierarchical scheduling method is selected as the most effective approach for computing Grid-systems. we propose an improved hierarchical scheduling method for Grid-systems, focused on one of the three optimization criteria chosen by the user. The Increase of the effectiveness of the proposed approach is provided using two modified scheduling algorithms - for homogeneous and heterogeneous nodes. Developed dynamic scheduling algorithm for heterogeneous Grid-systems nodes, which, in comparison with the known approaches can increase the scheduling efficiency by taking into account the cost of transfer through the formation of queues sub-tasks, as well as by improving the model of data delivery between processors in a Grid-system node. Experimental studies performed, confirm the higher efficiency of the proposed approach in comparison with the known.

Key words: GRID-systems, processors, homogeneity, acceleration coefficient, coefficient efficiency

INTRODUCTION

GRID-system (Foster and C. Kesselman, 2004) is a hardware and software infrastructure that provides reliable, stable, and inexpensive access to high-performance computing resources. As a computational medium, *GRID-system* uses a set of resources of different types and different computer systems and networks, including multiprocessor systems and clusters, which represent *GRID-systems* computing nodes. To date, a number of methods are developed for multiprocessor systems and clusters (Mu'alem, W. and D.G. Feitelson, 2001; Toporkov, V., 2009; Lifka, D., 1995; Garg, S.K., *et al*., 2009), which are used to solve the problem of parallel tasks scheduling.

These methods are divided into two groups. The first group is based on the idea of resources space division, (space-sharing) between jobs, according to which each task receives necessary resources for the required time in exclusive mode. Methods of the second group use time-sharing of processors between multiple jobs. This means that several jobs at any given time can share the same computing resources.

Among the most important GRID-system features, the complicating of scheduling tasks can be identified as the following (Zhu, Y., 2003):

The homogeneity (heterogeneity) and independence: Homogeneity and independence in terms of *GRID*-systems *user* are represented by different parameters such as resource parameters, the model and runtime optimization criteria.

The dynamics of the environment: *GRID-systems* are characterized by a dynamic environment; therefore, the performance of available resources is continuously changing. Scheduling algorithm should be adaptive to such dynamic behavior (resource reservation and rescheduling).

These reasons are imperative to develop new and better ways and means of scheduling tasks.

1. Analysis Of Algorithms For Scheduling Heterogeneous Grid – Systems:

In (Behnamian, J., *et al*., 2010; Chen, R.M., *et al*., 2010; AL-Khateeb, A., *et al*., 2009), given fairly detailed analysis of the known scheduling algorithms in *GRID-systems.*

Algorithm (Heterogeneous Earliest-Finish-Time) refers to the most popular scheduling algorithms for heterogeneous nodes (Topcuoglu, H., *et al*., 2002) and heterogeneous payroll algorithm proposed in (You, S.Y., *et al*., 2004).

In (You, S.Y., *et al*., 2004) presented description of the HEFT algorithm and a comparative analysis of this algorithm with well-known approaches for heterogeneous systems, such as LMT (Levelized - Min time) (Iverson, M., *et al*., 1995), MH (Mapping Heuristic) (Hesham El-Rewini, Ted G. 1990), DLS (Dynamic Level Scheduling-) (Sih, G.C. and E.A. Lee, 1993) and the CPOP (Critical-Path-on-a-Processor) (You, S.Y., *et al*., 2004). during queues formation in HEFT algorithm computing the value of "urgency" of vertices is defined as the sum of weights of vertices underlying on the critical path, starting from this vertex to the end node. The

Corresponding Author: Hussein. A. Al-Ofeishat, Computer Engineering Department, Al-Balqa Applied University, Jordan
E-mail: husseinmns@yahoo.com

weights of vertices in the HEFT use the average execution time of vertices on processors with different performance heterogeneous node GRID-system. The weights of the arcs in the graph during queues formation are not counted. Appointment produced to the processor, which provides the minimum completion time of vertices. To determine the optimal processor all the node processors *GRID-s* systems are considered, whether they are busy or free at the scheduled time.

The decision is made by modeling the performance of appointed vertices on each processor. It is assumed that the heterogeneous node is full, data transfer speed between processors is the same, and distribution of different messages from one processor to another can be performed simultaneously. That is why the HEFT algorithm is one of the most popular for heterogeneous *GRID-systems* nodes. This is confirmed by studies in ASKALON (Foster and C. Kesselman, 2004), where this algorithm is compared with genetic algorithm and shows better results.

However, this algorithm has some disadvantages, such as:

- During queue formation of graph vertices, it does not include the of transfers cost;
- Does not account the different transfers speed;
- The algorithm uses a simplified model of data transfer (sending different messages from one processor to another is performed simultaneously), which does not include the actual delay and reduces the accuracy of the application execution;
- Considerable difficulty in the appointment, because when the decision is made, all node processors in Grid-systems are searched using simulation.

In (You, S.Y., *et al*., 2004) a description is provided for heterogeneous scheduling algorithm and given a comparative analysis of this algorithm with HEFT. This algorithm differs from HEF in method of queue formation of vertices. Here, computing the value of "urgency" of vertices is defined as the sum of the weights of vertices underlying on the critical path, starting from this vertices until end node, plus the sum of weights of connections between them. Thus, the authors have tried to eliminate the first of the above-mentioned drawbacks of the HEFT algorithm. The procedure is similar to that used in HEFT. Thus, all other drawbacks are not correct. However, comparative analysis is presented in (Sih, G.C. and E.A. Lee, 1993), and shows that when performing Gauss task of different dimensions, the proposed payroll heterogeneous scheduling algorithm is more effective than the HEFT algorithm.

2. Description Of The Proposed Scheduling Algorithm For Heterogeneous GRID-systems nodes:

In this paper we will solve the following scheduling optimization problem.

In general, the scheduling problem is reduced to determining the sum of the *i-th node* Grid-system, which provides minimum execution time for a given application:

$$T_i = \sum_{i=1}^{m} \sum_{l=1}^{P_i} t_{jli} * X_{jli} + S_i + \max\{Tr, Tf_i\} \quad (1)$$

Where:

t_{jli} - Runtime *j-th* sub-problem in the *l-th* processor the *i-th* node of Grid-system;

S_i - Delivery time of input data and application results to/from the *i-th* node Grid-system;

Tr - Application preparation time to perform in the system nodes;

Tf_i - Release time of *the i-th* node of Grid-system to perform a given application in exclusive mode;

$X_{jli} = 1$, If the *j-th* sub-problem is performed *l-th* processor *the i-th* node,

Otherwise, $X_{jli} = 0$.

In this paper we consider two ways to select the optimal node of Grid system.

In the first case, *the i-th* node of Grid-system is required to be at minimal cost (C_i), which provides the implementation of a given application for a limited time (T_z). A mathematical model of this problem can be written as follows.

Find

$$\min_{i=1,R} \{C_i\} \quad (2)$$

With

$$T_i \le Tz(i = 1, K) \quad (3)$$

Initially a subset of nodes and task execution time are defined that corresponds to (3). Among them it is necessary to find a node in which the application is executing with a minimum cost (2).

In the second case, *the i-th* node of Grid-system is determined with minimal cost, which provides a minimum total execution time for the specified application. A mathematical model of this problem can be written as follows. Find the node of Grid-system, satisfying (1).

$$C_i \le C_{\min} \tag{4}$$

Initially in this case, according to (1), subsets of nodes are determined, providing applications executed in minimal time. Then among them, according to (4), the node with the minimal cost is selected.

The main purpose of the proposed algorithm is to improve its efficiency in comparison with the HEFT algorithm and heterogeneous scheduled algorithm.

For this purpose, we propose the following:

1) Use a method of queue formation that determines the period of the delayed vertices. In the nodes, the homogeneous algorithm of the delayed time of execution is defined as the difference between the critical path, the length of the graph and the length of the longest path starting from a given vertex and ending with the final considering the vertex weights, but without calculation of the arcs weights. The algorithm of the heterogeneous nodes, instead of vertex weights uses average execution time of vertices on processors with different performance heterogeneous Grid-system node.

2) Improve the model of data delivery between processors in the Grid-system node while the destination is executed.

Consider the basic steps of the proposed scheduling algorithm for heterogeneous Grid-system nodes:

1) Create vertex "parents" for each array on the basis of a problem graph.

2) Immersion of each vertex, which has no "parents" to the processor on which it will be execute.

3) Each of the processors in turn recognize the "optimal" and performs an immersion. Thus, each processor computes the indicator of start vertex execution time. The processor that has the lowest load will be acknowledged," is really the optimal" and it performs the "real Immersion".

4) Each "parent" performs the transfer of their node-holders to the optimal node. A table of "engaged channel nodes" is created before the start of the transfer. Initially, each processor has a clock cycle, equal to the number of physical links. If transfer is performed in the current cycle involving this processor, then the index in the table is decremented. If the current processor clock cycle rate is equal to zero, it means that all links are busy, so the transfer should start later. The exception, if the duplex transmission is allowed, then the current transfer goes to the old transfer on the same connection, but in the opposite direction.

5) After the immersion of each vertex, it is removed from the queue. If the vertex is unable to load due to the lack of data, then it is skipped and the next iteration performs a new attempt to load the vertex. Thus, the algorithm is as accurate as possible adhering to the queue.

6) Once the queue becomes empty, the algorithm terminates.

3. The Structure Of Resource Broker's GRID-systems:

The hierarchical scheduling system uses the central and local brokers.

The central broker is a class that describes the work of central broker scheduler for hierarchical Grid-system.

The block diagram of a central broker is shown in Fig. 1.

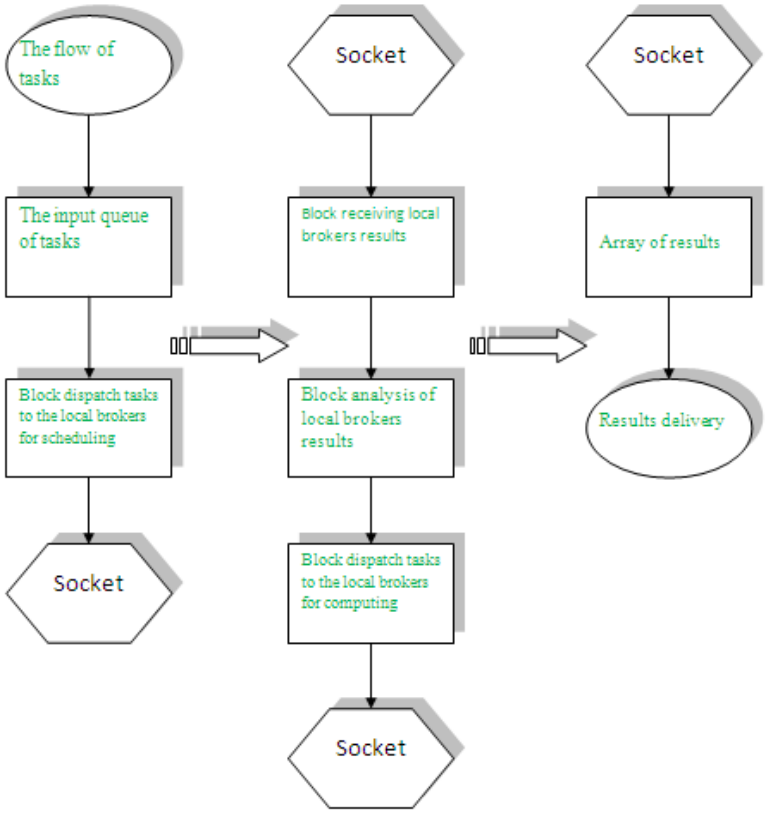

Fig. 1: The block diagram of a central broker

According to its job, a central broker must perform the following functions:

1. Receive tasks that come from users
2. Setting the tasks in the queue
3. Dispatch tasks to a local broker for scheduling
4. Receive and analyze the results
5. Deliver the scheduled results to the user
6. Receive instructions from the user about the chosen problem solution
7. Data transfer to a local broker to compute
8. Receive computation results
9. Transfer the results to the user
10. Receive messages from local brokers about errors in their work
11. Tracking the performance of local brokers
12. Making a decision upon the occurrence of errors

The algorithms diagram of the central broker is shown in Fig. 2.

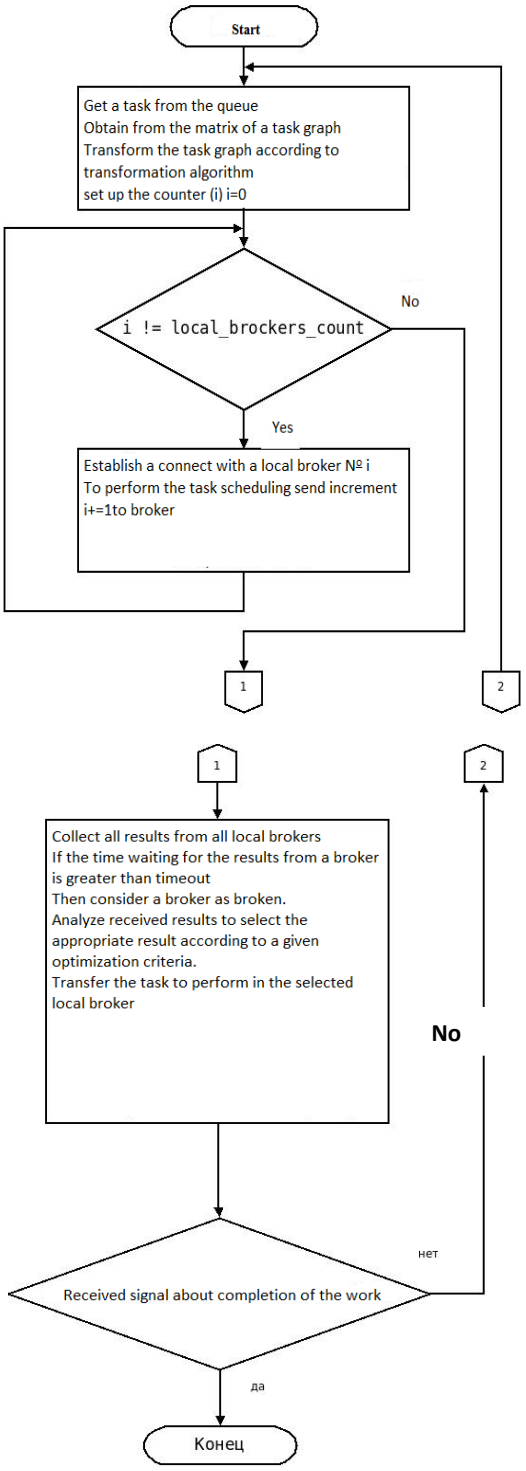

Fig. 2: The algorithms diagram of the central broker

The flow is generated through a set of connected tasks. The stepwise algorithm of the generator of input tasks is described below. The generator of input tasks generates a matrix of connected tasks with a given connection, and the vector of vertex weights. Figure 3 shows a diagram of the input queue.

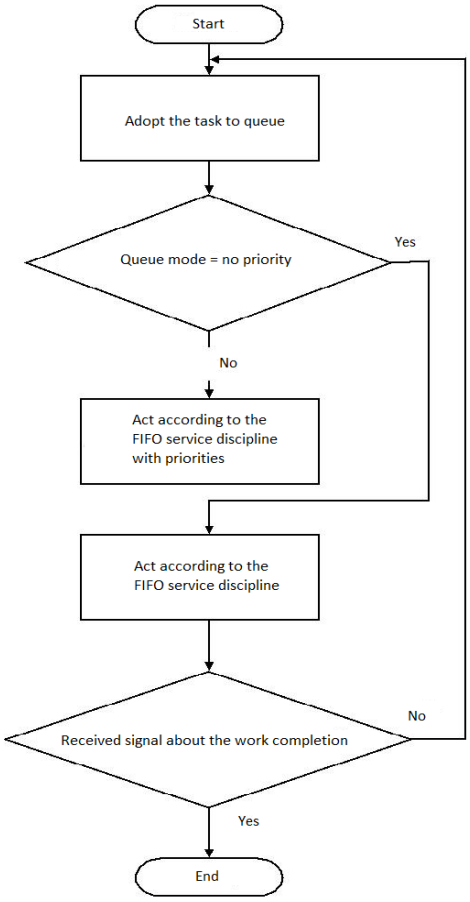

Fig. 3: The scheme of the input queue

Generating A Graph Of Algorithm Tasks:

1) State the following characteristics: the number of vertices, the minimum weight of a vertex, the maximum weight of a vertex, the minimum weight of edge, the maximum weight of the edge, and coherence.

2) Perform the creation of a given number of vertices, choosing weights randomly within a given limit

3) Calculate the sum of connection weights using a formula of coherence: $\Sigma SW = C*\Sigma ver/(1-C)$

4) Randomly choose a source-vertex, selecting a number from 0 to the number of vertices - 1. Then randomly select the receiver with a number greater than the number of the source

5) Randomly select an edge, weigh and carry it from source to destination. If an edge already exists, then the generated weight is added to the weight of the existing edge.

6) From the sum of connection weights, subtract the generated weight of an edge.

7) The algorithm performs until the sum of connection weights become a positive number.

After the selection of a task from a queue, it is sent to all local brokers to undergo scheduling. After being performed, the planning results are sent to a central broker, then the results are transmitted back to the central broker, after a collection of all the results for one task, the central broker analyzes received information and decides about providing an available resource (node). Users are required to specify the recommended resource to execute the task, and other options. After the clients decision, performed tasks transmit to execution according to the chosen load variant.

3.6.2. Specifications Of A Local Broker:

Local broker - a class that describes the work of a local broker's hierarchical system scheduler of Grid-systems, that perform the following functions:

1. Receive tasks from a central broker and schedule them
2. Perform the scheduling according to the systems topology
3. Transmission of scheduling results to the central broker
4. Receive tasks and data to execute
5. Preparation of a work schedule for its system part

6. Monitoring the schedule execution
7. Notifying the central broker about an error
8. Transfer results of computations to the central broker

The block diagram of a local broker is shown in Fig. 4.

After the central broker receives a message, scheduling is performed. The scheduling algorithm depends on whether the node is homogeneous or not. Furthermore, space in the schedule is reserved for this task, given that it is decided to be solved in this node. Then the scheduling results are transmitted to the central broker. If computation is decided to be performed in a given node, then the local broker accepts the task and the data for it.

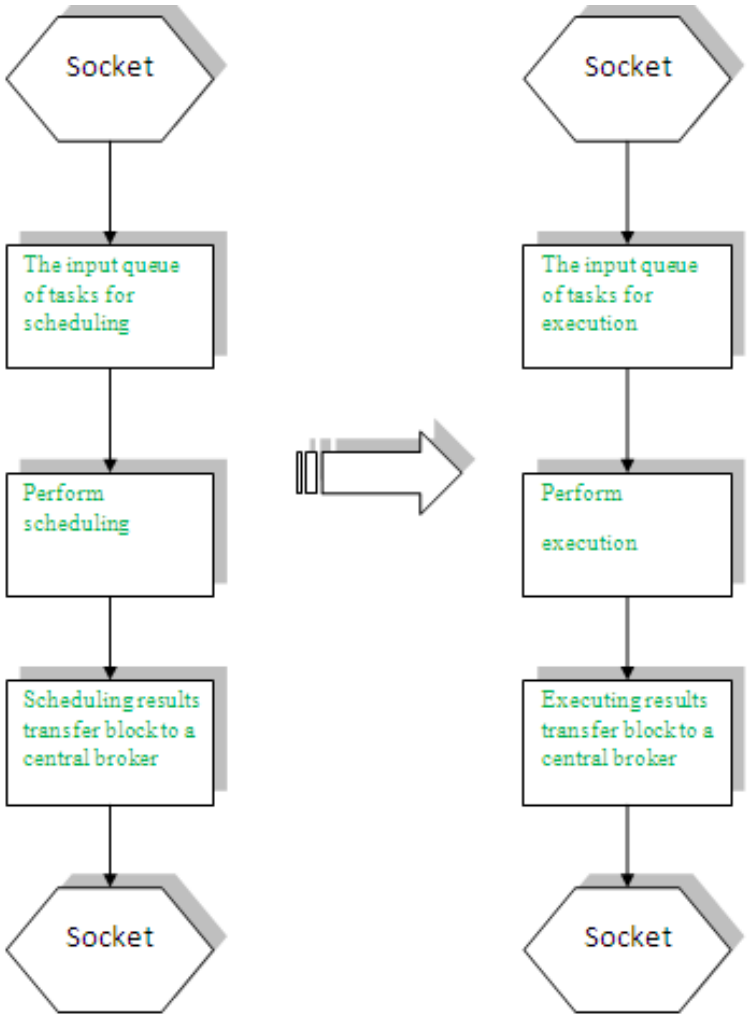

Fig. 4: The block diagram of a local broker

3.8.2. Comparative Analysis Of Scheduling Algorithms For Heterogeneous Nodes Grid-system:

The main type of experiments that are performed using the software model consist of a sequential generation of large numbers of different connectivity graphs and collect statistics data about acceleration coefficient and coefficient efficiency, depending on the tasks connectivity, and on the number of processors in the nodes.

The first stage of collecting statistical data includes the establishment of the local broker the HEFT algorithm and produces the collection of statistical data. In the second stage HEFT was replaced by the proposed algorithm and again the statistical data collection has been made.

The results of the programming model are shown in Tables 1 and 2.

Table 1: The acceleration coefficient when HEFT algorithm works on a local broker

	The number of processors								
C(%)	20	30	40	50	60	70	80	90	100
50	1,85	2,375	2,9	3	3,1	3,15	3,2	3,3	3,4
60	1,83	2,115	2,4	2,5	2,6	2,6	2,6	2,7	2,8
70	1,7	1,9	2,1	2,2	2,3	2,4	2,5	2,6	2,7
80	1,62	1,76	1,9	2	2,1	2,15	2,2	2,25	2,3
90	1,53	1,565	1,6	1,75	1,9	1,95	2	2,1	2,2

Table 2: The acceleration coefficient when proposed algorithm works on a local broker

	The number of processors								
C(%)	20	30	40	50	60	70	80	90	100
50	1,79	1,92	1,82	2,11	2,18	2,37	2,33	2,54	2,53
60	1,89	2,13	2,14	2,38	2,40	2,59	2,55	2,71	2,64
70	1,98	2,28	2,36	2,60	2,62	2,86	2,89	3,09	3,08
80	2,12	2,51	2,68	2,92	2,94	3,08	3,00	3,20	3,19
90	2,15	2,79	3,21	3,46	3,49	3,69	3,66	3,87	3,85

The average values of the acceleration coefficients are given in Table 3. Figure 5 and Figure 6 show the simulation results with different tasks of connectivity and different numbers of processors.

Table 3: The average values of the acceleration coefficients depending on the number of processors

	The number of processors								
	20	30	40	50	60	70	80	90	100
The average values of acceleration coefficients (HEFT)	1,71	1,94	2,18	2,29	2,40	2,45	2,50	2,59	2,68
The average values of acceleration coefficients (proposed algorithm)	1,99	2,32	2,44	2,69	2,73	2,92	2,89	3,08	3,06

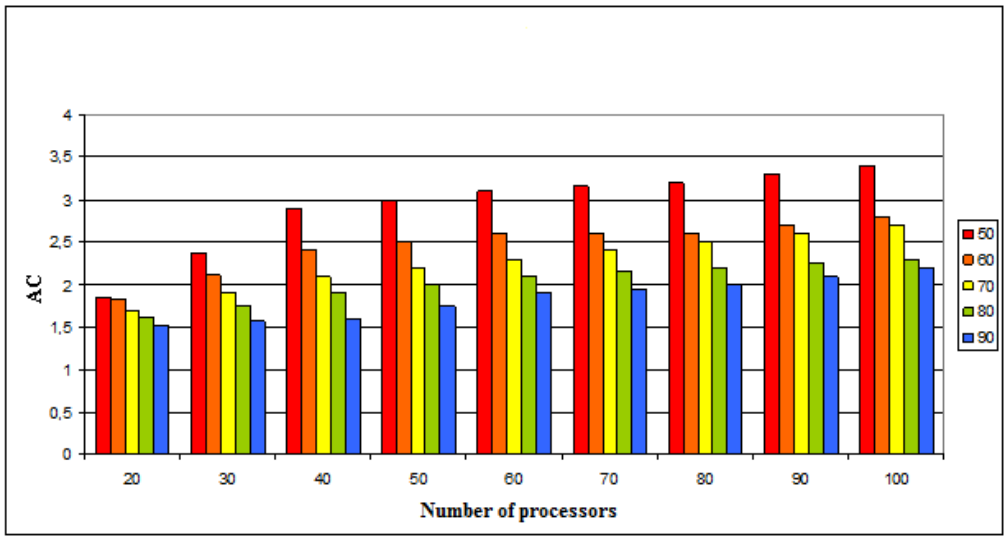

Fig. 5: The dependence of acceleration coefficient on the number of processors with different tasks connectivity when using the HEFT algorithm

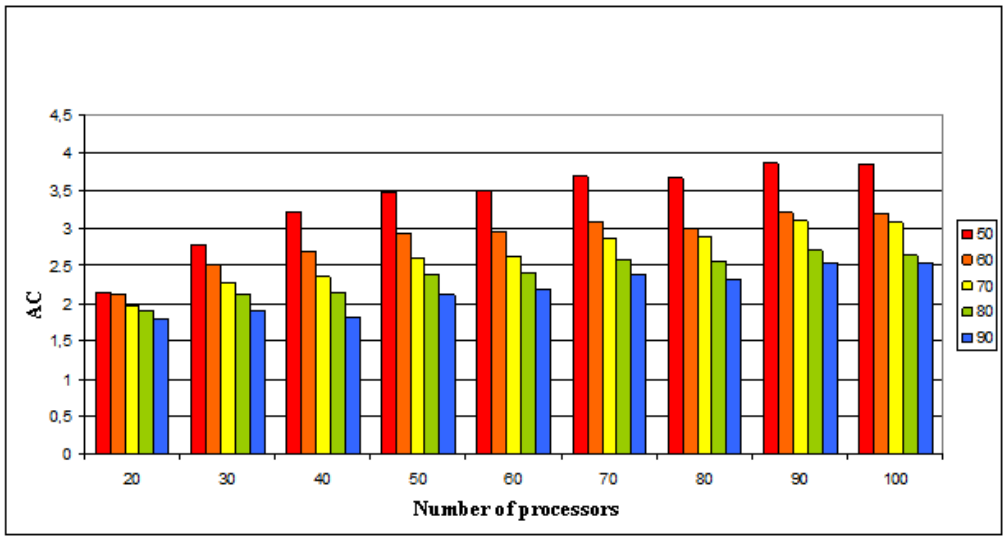

Fig. 6: The dependence of acceleration coefficient on the number of processors with different tasks connectivity when using the proposed algorithm

According to data obtained when using the proposed scheduling algorithm the use of classical HEFT algorithm gives benefits in 11% as average, which is essential especially when dealing with complex tasks.

As a result of the statistics of modeling and analysis, we can conclude the following:

1. The presence of three physical links in each processor node can increase the acceleration coefficient by 10-20%
2. Calculating the duplex channel processor increases the acceleration coefficient by 5-15%
3. In general, the growth of acceleration coefficient compared to the basic algorithm presented in [23], about 20-25%
4. Closest to the optimal acceleration coefficient is observed around 30% of the task connectivity
5. When the ratio of the task dimension 5:1, the proposed algorithm increases 18% to system performance and 15% to system efficiency. At a ratio of 10:1, these indicators increase to 25% and 20% respectively
6. The simulation using star topology, results in performance growths when using the proposed algorithm compared with a base about 17%, and the growth of efficiency coefficient equals 13%.
7. Using the proposed scheduling algorithm for heterogeneous nodes is better than the use of HEFT algorithm in 11%

Conclusions:

1. Analyzing the main scheduling methods in Grid-systems, the hierarchical scheduling method is selected as the most effective approach for computing Grid-systems.
2. We propose an improved hierarchical scheduling method for Grid-systems, focused on one of the three optimization criteria chosen by the user. The Increase of the effectiveness of the proposed approach is provided using two modified scheduling algorithms - for homogeneous and heterogeneous nodes.
3. Developed dynamic scheduling algorithm for heterogeneous Grid-systems nodes, which, in comparison with the known approaches can increase the scheduling efficiency by taking into account the cost of transfer through the formation of queues sub-tasks, as well as by improving the model of data delivery between processors in a Grid-system node.
4. Experimental studies performed, confirm the higher efficiency of the proposed approach in comparison with the known.

REFERENCES

AL-Khateeb, A., R. Abdullah, N. Rashid, 2009. Job Type Approach for Deciding Job Scheduling in Grid Computing Systems // Journal of Computer Science, 5: 745_750.

Behnamian, J., M. Zandieh, & S.M.T.F. Ghomi, 2010. Due windows group scheduling using an effective hybrid optimization approach. International Journal of Advanced Manufacturing Technology, 46(5-8): 721-735, ISSN (printed): 0268-3768.

Chen, R.M., C.L. Wu, C.M. Wang, & S.T. Lo, 2010. Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. Expert systems with applications, 37(3): 1899-1910, ISSN: 0957-4174.

Foster and C. Kesselman, 2004. "The Grid: Blueprint for a NewComputing Infrastructure," Elsevier Inc., Singapore, Second Edition.

Garg, S.K., R. Buyya, H.J. Siegel, 2009. Scheduling parallel applications on utility Grids: time and cost trade-off management // Proc. of the 32nd Australasian Computer Science Conf. (ACSC 2009). Wellington, New Zealand, pp: 151-159.

Hesham El-Rewini, Ted G. 1990. Lewis Scheduling Parallel Program Tasks onto Arbitrary Target Machines, J. Parallel and Disrubuted Computing., 9: 138-153.

Iverson, M., F. Ozguner and G. Follen, 1995. Parallelizing Existing Applications in a Distributed Heterogeneous Environment, Proc. Int. Conf. Parallel Processing, 2(93): 100.

Lifka, D., 1995. The ANL/IBM SP scheduling system. In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (eds.), pp. 295–303, Springer-Verlag, 1995. Lect. Notes Comput. Sci., 949.

Mu'alem, W. and D.G. Feitelson, 2001. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. IEEE Trans. Parallel & Distributed Syst., 12(6): 529-543.

Sih, G.C. and E.A. Lee, 1993. A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures, IEEE Trans. Parallel and Distributed Systems, 4(2): 175-186.

Topcuoglu, H., S. Hariri, M.Y. Wu, 2002. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, IEEE Transactions on Parallel and Distributed Systems, 13(3): 260-274.

Toporkov, V., 2009. Application-level and job-flow scheduling: an approach for achieving quality of service in distributed computing // Proc. of the 10th Intern. Conf. on Parallel Computing Technologies. Heidelberg: Springer, LNCS. 5698: 350-359.

Wieczorek, M., R. Prodan and T. Fahringer, 2005. Scheduling of Scientific Workflows in the ASKALON Grid Environment, in ACM SIGMOD Record, 34(3): 56-62.

You, S.Y., H.Y. Kim, D.H. Hwang, S.C. Kim, 2004. Task Scheduling Algorithm in GRID Considering Heterogeneous Environment, in Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA '04: 240-245, Nevada, USA.

You, S.Y., H.Y. Kim, D.H. Hwang, S.C. Kim, 2004. Task Scheduling Algorithm in GRID Considering Heterogeneous Environment, in Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA '04: 240-245.

Zhu, Y., 2003. A Survey on Grid Scheduling Systems, Department of Computer Science, Hong Kong University of science and Technology.