Two stage strategy of job scheduling in grid environment based on the dynamic programming method

Volodymyr V. Kazymyr, Olga A. Prila

***Abstract*—**The problem of the effective usage of the grid environment for solving different types of computing tasks of large dimension is researched in the paper. We study the problem of optimal task scheduling at an affordable set of resources on the one hand and the equitable distribution of resources between the tasks that come into the input queue of a centralized workflow management system, on the other hand. Two stage strategy of task scheduling in grid environment that takes into account user-defined QoS requirements, structural features and execution dynamicity of the task is presented. The dynamic programming method application to the workflow scheduling problem is proposed in the paper and the effectiveness evaluation experimental results of the proposed decision are given.

***Keywords*—**Grid, workflow, scheduling, QoS

I. Introduction

Currently grid technologies are actively developed and applied to solving of complex high-dimensional problems. The problem of task adoption for effective execution in the grid environment is rather complex itself. Abstract features of using of the grid-infrastructure for different types of tasks' execution based on their structural but not functional features can be determined.

The actively developing field in the grid-computing is the technology of workflow execution in grid-environment. The workflow represents the task as a sequence of subtasks with certain synchronization scheme. The presence of several parallel blocks in such tasks allows to execute them on different resources for more efficient problem solution. To provide such a solution several factors should be taken into account and the most important of them is the expenses for data exchange between utilized resources. Workflow scheduling is an NP-complete problem in general [1].

Another component that is very important for grid end-users is the ability of a grid system to provide its consumers with the required quality of service (QoS). It results in the need to allocate task on a set of resources which is most suitable for its execution depending on the QoS information provided by the resources. While for non-commercial community grids it is limited to estimate completion time (ECT), commercial utility grids can also operate with costs of calculations and some other parameters.

The paper is dedicated to the research of the aspects of scheduling different types of tasks in grid-environment paying much attention for the workflow scheduling problem. Another research problem is the strategy of the central queue of grid tasks processing according to their priorities and QoS requirements.

The complex two stage strategy for tasks' queue processing and task scheduling next is proposed in the paper.

V. V. Kazymyr, Dr. Sc. Prof. Chernihiv State Technological University, Shevchenko street, 95, Chernihiv-27, Ukraine, 14027; e-mail: v.vkazymyr@gmail.com.

O. A. Prila, postgraduate, the assistant lecturer, Chernihiv State Technological University, Shevchenko street, 95, Chernihiv-27, Ukraine, 14027; e-mail: olga.prila1986@gmail.com.

II. The Formalization Of The Task Of Scheduling Different Types Of Jobs In Grid Environment

One of the factors influencing the performance of the grid-network is planning efficiency. Taking into account the heterogeneity of grid-resources, as well as the structural features of tasks, the following factors should be understood under the scheduling efficiency.

1) Equable load of all the grid computing elements.
2) The minimal tasks' downtime in the run queue.
3) The minimal execution time of tasks on a dedicated set of resources, including the time required for data transfer between computing blocks.

The classification of the tasks which are calculated in the grid-environment according to the structural criterion has been suggested by the authors [2].

The execution effectiveness of the task represented by a single computing unit or a set of consistent, depends on the effectiveness of its program implementation, planning strategies of the low-level grid brokers and local scheduler.

If the task is represented by a set of similar tasks with different input data, scheduling optimization reduces to decomposition of the task according to the current options of grid-infrastructure.

The presence of parallel blocks in workflow tasks allows executing them on different resources for more efficient problem solution. To provide such a solution the expenses for data exchange between utilized resources should be taken into account.

The expenses for data exchange can be eliminated by clustering several blocks of workflow for the same resource. There is a concept of linear and nonlinear clustering [1], when serial or parallel blocks are grouped respectively. The optimization problem is reduced to finding the optimal solution between parallelization and clustering.

Workflow task is generally modeled by a precedence-constrained task graph, which is a directed acyclic graph with nodes representing the subtasks and the directed edges representing the execution dependencies between them as well as the amount of communication (see Fig. 1).

For effective workflow scheduling the following subtask parameters must be defined:

{ECT, Memory, {T}},

where ECT - estimated completion time;

Memory – memory requirements;

{T} – set of links to other modules (one-way communication between nodes).

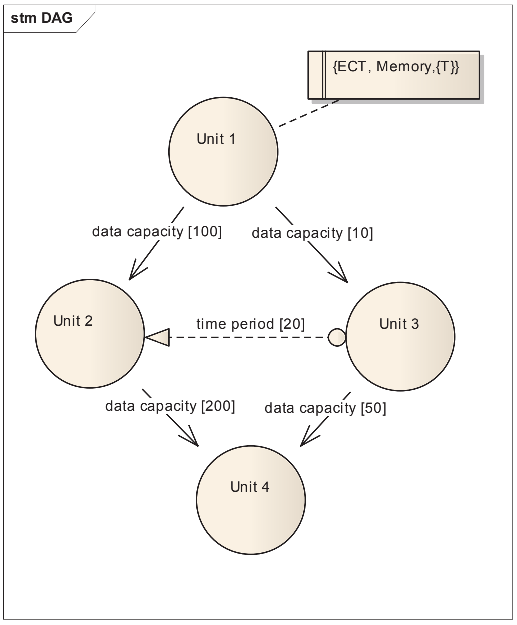


Fig. 1 the example of workflow task structure

Each relationship is defined by the data capacity parameter – the amount of data transferred between the units. The relationship between Unit2 and Unit3 determines the need for periodic synchronization of data between the units that are executed in parallel.

The complex type of the task which includes the characteristics of several types should be considered separately. There may also occur a particular case of type three when the task is decomposed into totally independent parallel subtasks. In the latter case the process of adaptation to grid is significantly simplified due to the absence of necessity of synchronization between the calculation blocks.

For effective execution in the grid-environment the following limitations are imposed for the workflow structure.

1) Lack of loops and branches. These limits are determined by the task structure presentation in the form of an acyclic directed graph. However, the task structure having loops can be converted to DAG by adding an additional level. Branching can be handled at the level of metascheduler through the dynamic approach to planning.
2) High level of task granularity. The dimension of calculations should be much higher in relation to the dimension of the transmitted data [1]. In [3] the granularity problem is defined as follows:

$$g = \min_{x=1:\nu}\{\tau_x / \max_j\{c_{x,j}\}\}, \tag{1}$$

where τ_x – computational complexity of node n_x;

$c_{x,j}$ – dimensionality of the data being transferred between nodes n_x and n_j;

ν – the number of computing nodes of the task.

Grid-network structure can be presented as a complete directed graph where vertices define the resources, and the weight of arcs define bandwidth computer network (see Fig. 2).

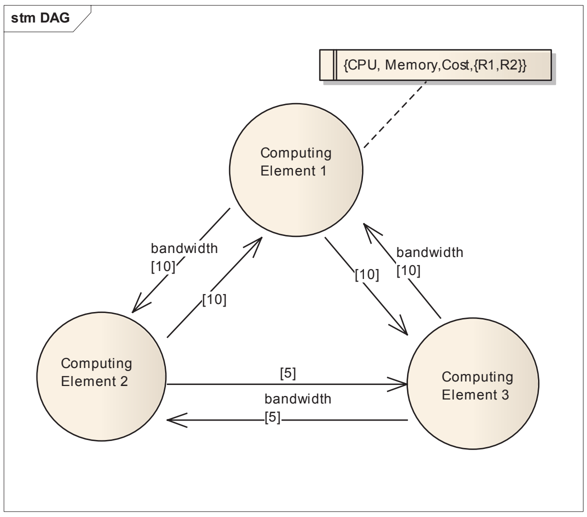


Fig. 2 grid-structure example

Each unit of the grid-net structure is characterized by the following compulsory parameters.

{CPU, Memory, Cost, {R1, R2}},

where CPU – computational power, Memory – memory characteristics, Cost – usage cost, R1 – receive data network bandwidth и R2 – data transmission network bandwidth.

The optimization task presupposes working out an optimal variant of the stream of jobs arrangement on the available set of resources.

The classification of jobs scheduling according to the following criteria is presented in [4]:

$$\{\alpha \mid \beta \mid \gamma \mid \delta\}, \quad \beta = \{\beta_1 \mid \beta_2 \mid \beta_3\}, \tag{2}$$

where α – determines the characteristics of the distributed environment (homogeneous / heterogeneous);

β – the job specifications and the presence of limitations in the job structure;

β_1 – presence of relation between the job computing units;

β_2 – homogeneity / heterogeneity of the job computing units;

β_3 – presence of time limitations of the job computing unit irrespective of the results of the units connected with it;

γ – determines the optimization criterion and the type of the objective function;

δ – determines the expenditure function of the distributed environment resources interaction in job performance.

According to the classification suggested the job scheduling in the grid-environment is determined the following way

$$\{R \mid PREC, \varnothing, r_i \mid C_{\max} \mid JP\}, \qquad (3)$$

where R – determines the heterogeneity of the distributed environment resources; the time of the job computing unit performance is the power function of the distributed environment unit;

$PREC$ – the presence of relations between the job computing units;

$\varnothing$ – the job units have different computing complexity;

r_i – there are time limitations as for the beginning of the job computing unit;

$C_{\max}$ – the objective of the job scheduling is the minimization of the time of job performance;

JP – the expenditures on the distributed environment units interaction are determined by the parameters of the network bandwidth, as well the job specifications (the level of transmission data between the computing units).

The suggested classification does not take into account the multicriterion characteristic of the objective function. However, besides the job performance time, the competitive criterion, computing cost, is significant for the commercial grid-environment. The task of jobs streams scheduling in the grid-environment is generally NP-total task.

At the middleware level grid does not provide full support for the tasks of different types. For instance, ARC Nordugrid (http://www.nordugrid.org/) and gLite (http://glite.cern.ch/), which are on the list of the main providers of grid middleware in EMI (http://www.eu-emi.eu/) and which are widely used by the Ukrainian national grid-infrastructure make use of the following formats of the grid-tasks specifications: JSDL [5], xRSL [6] and JDL [7]. Among the mentioned JDL-format is the only to introduce the notion of the task type (Job, DAG и Collection), which still has certain limitations – the determination of the periodic synchronization between the units and the data transmission levels is impossible. JSDL and xRSL formats provide just means of determining the parameters of some tasks; however, the jobs stream life-cycle, as well as the relations between certain tasks, is not supported.

Middleware grid brokers realize simplified scheduling strategies and do not allow performing jobs stream scheduling taking into consideration the tasks specifications and QoS parameters. For instance, ARC Nordugrid broker realizes the following strategies of the available computing resources selection: RandomBroker, BenchmarkBroker, FastestQueueBroker, DataBroker [6]. The latter means that the mechanisms of complex grid-tasks scheduling are to be realized and arranged beyond the middleware grid level.

III. The Structure Of The Metascheduler Of The Centralized Workflow Management System

An important component of the use of the grid-environment is to provide its consumers with the required level of quality of service (QoS).

In addition to finding the optimal schedule of workflow task on the set of available computing resources, the important aspects of the metascheduler are: a) the strategy of processing the input queue of tasks in accordance with their priorities; b) the choice of the scheduling algorithm according to the structural features of the problem; c) dynamic control of task execution; d) accounting the dynamicity of grid-network as well as the level of quality of service of grid resources.

Below we consider the metascheduler implementation aspects which are an integral part of the centralized workflow management system. Lack of the centralized approach, which consists in the possible occurrence of bottleneck, is assumed to be eliminated through the scalable workflow management system architecture.

This paper introduces a two-stage strategy for task scheduling in grid environment. The first stage involves the processing of the input queue of tasks in accordance with their priorities and QoS requirements. The second phase involves task scheduling at the affordable set of resources taking into account the structural features of the task.

Below we consider the approaches to scheduling the workflow task on the set of available heterogeneous grid-resources as well as strategies for handling the input queue of tasks of different types accounting the dynamicity of their execution.

IV. Dynamic Programming Method Application TO The Problem Of Workflow Scheduling

In [1, 8, 9] the classification and the results of the algorithms' effectiveness and complexity evaluation are given.

The methods of search in the space of states and methods of mathematical programming can produce optimal solutions, but in general are characterized by high computational complexity of the algorithm. Heuristic approaches can give effective solutions in polynomial time, but in general these approaches do not provide the optimal solution, as the average, the worst and the best performance of these algorithms is unknown [1].

Clustering (DSC, CASS-II [10]) and replication (TDS, TANH [11]) approaches aimed at reducing the time required for data transfer between nodes by placing tasks that require the exchange of large amounts of data on a single resource or duplicate blocks, respectively [9]. The disadvantages of these approaches is the difficulty of accounting the heterogeneity of the subtasks, and the lack of opportunities to use several resources grid-network for parallel blocks task.

An important aspect of the use of commercial grid-environment is the need to optimize the characteristics of mutually exclusive - resource cost and execution time taking into account the significance of the coefficients of each of the characteristics. Most heuristic scheduling algorithms focus on improving one of the criteria. Today, the only workflow scheduling algorithm that solves the multiobjective optimization problem is the LOSS / GAIN algorithm [8].

Many of the existing scheduling algorithms impose some restrictions on the structure of the problem, the structure of grid-network optimization criteria.

Dynamic programming method is one of the methods of mathematical programming, applied to the problem with optimal substructure. Optimal substructure problem assumes that the optimal solution of its constituent smaller subtasks can be used to solve the original problem [12].

The algorithm introduces the concept of levels in the structure of the problem of the "work flow", which is determined by a variety of tasks that can be performed simultaneously at a certain stage of the task. For example, a workflow structure shown in Fig. 1 may be allocated to the following levels: 1) {Unit 1}; 2) {Unit 2, Unit 3}; 3) {Unit 4}.

Optimal solution contains optimal solutions at every level, and, therefore, the task has the property of optimality [16].

The objective function of the algorithm can be determined by several parameters that have some weight. Accordingly, the objective function might look as follows:

$$y = k_1 \cdot time + k_2 \cdot \cos t \quad (4)$$

where k_1, k_2 - user-defined coefficients of QoS parameters;

time – task execution time;

cost - the cost of computing resources usage.

The flowchart of the algorithm is shown in Fig. 3.

As it can be seen from the flowchart, at each level for each allocation variant the optimal solution is saved regarding the allocation cost as well as the cost of interaction with the blocks of the previous levels. Inefficient solutions for each location are discarded and will not be further considered. At the last step of the algorithm the global optimal solution is determined moving backward from the bottom up through the levels. It is recommended to store a copy of the accommodation plan ordered by the value of the objective function, which can be used for dynamic rescheduling problem if necessary.

In the case of existence of the “through the level” communication link the “dummy" block of zero computational complexity is assumed to be added.

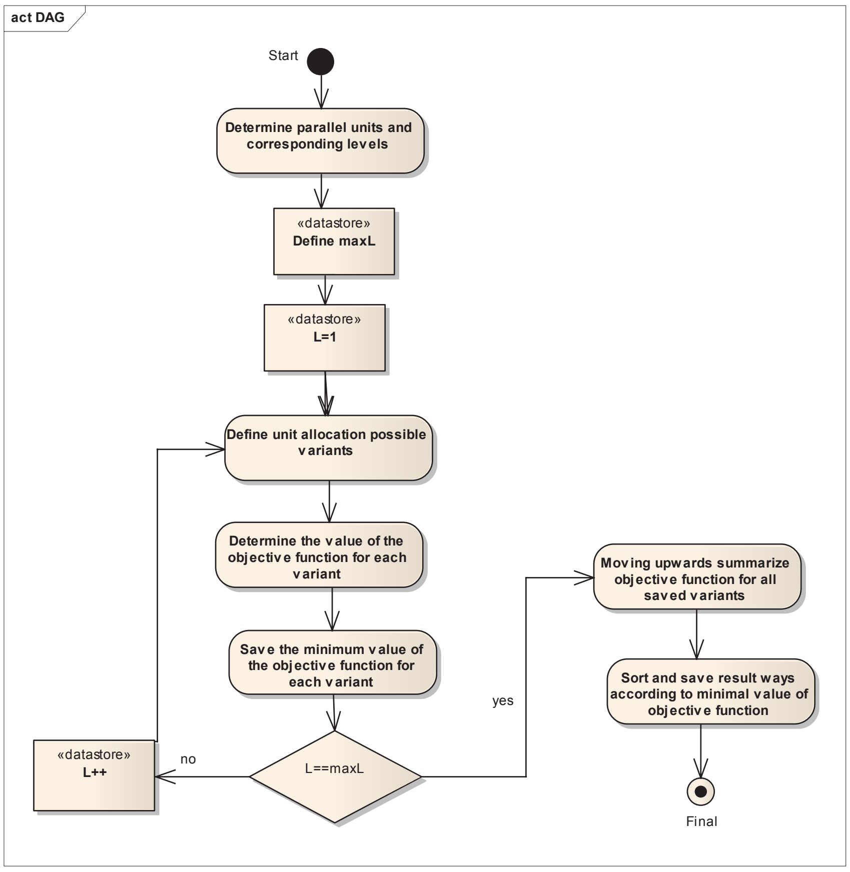

act DAG
Start
Determine parallel units and corresponding levels
«datastore» Define maxL
«datastore» L=1
Define unit allocation possible variants
Determine the value of the objective function for each variant
Save the minimum value of the objective function for each variant
L==maxL
no
«datastore» L++
yes
Moving upwards summarize objective function for all saved variants
Sort and save result ways according to minimal value of objective function
Final

Fig. 3 the QoS-based scheduling algorithm flowchart

Consider the example of the algorithm for the workflow shown in Figure 2 and the network structure shown in Figure 3.

For the simplicity we take the objective function as

$$f(\text{time}) \rightarrow \min \qquad (5)$$

The input data is represented in a table structure (see Table 1-2).

Table 1. Workflow structure

	1	2	3	4	ECT
1	#	100	10	#	500k
2	#	#	#	200	2000k
3	#	#	#	50	100k
4	#	#	#	#	100k

Table 2. Grid network structure

	0	1	2	CPU
0	0	10	10	100k/time
1	10	0	5	50k/time
2	10	5	0	50k/time

Step 1. Determined levels:

Level 1: Unit 1

Level 2: Unit 2, Unit 3

Level 3: Unit 4.

Step 2. For set №1 define all possible variants of unit allocation:

U1CE1, U1CE2, U1CE3.

Step 3. Determine the value of the objective function for each variant.

U1CE1: 500k / 100k = 5;

U1CE2: 500k / 50k = 10;

U1CE3: 500k / 50k = 10;

Step 4. Save the value of the objective function for each variant.

Step 5. Turn to set № 2. Repeat steps 2-4. First, we define the computational cost for U2 and U3 (for ease of computation), and then for all possible combinations of the location of the previous set of blocks.

a) U2CE1U3CE2: 2000k/100k + 100k/50k = 22;

b) U2CE1U3CE3: 2000k/100k + 100k/50k = 22;

c) U2CE2U3CE1: 2000k/50k + 100k/100k = 41;

d) U2CE2U3CE3: 2000k/50k + 100k/50k = 42;

e) U2CE3U3CE1: 2000k/50k + 100k/100k = 41;

f) U2CE3U3CE2: 2000k/50k + 100k/50k = 42;

The calculated objective function for each combination of set №2 is presented at table 3.

Table 3. The objective function value for each combination of set №2

	U1CE1 (5)	U1CE2 (10)	U1CE3 (10)	Min
U2CE1 U3CE2 (22)	22+5+0+10/10	22+10+100/10+0	22+10+100/10+10/5	28(U1CE1)
U2CE1 U3CE3 (22)	22+5+0+10/10	22+10+100/10+10/5	22+10+100/10+0	28(U1CE1)
U2CE2 U3CE1 (41)	41+5+100/10+0	41+10+0+10/10	41+10+100/5+10/10	52(U1CE2)
U2CE2 U3CE3 (42)	42+5+100/10+10/10	42+10+0+10/5	42+10+100/5+0	54(U1CE2)
U2CE3 U3CE1 (41)	41+5+100/10+0	41+10+100/5+10/10	41+10+0+10/10	52(U1CE3)
U2CE3 U3CE2 (41)	42+5+100/10+10/10	42+10+100/5+0	42+10+0+10/5	54(U1CE3)

Step 6. Turn to set № 2. Repeat steps 2-4.

U4CE1: 100k / 100k = 1;

U4CE2: 100k / 50k = 2;

U4CE3: 100k / 50k = 2;

In table 4 we present only the results of calculations.

Table 4. The objective function value for each combination of set №3

	a (28)	b (28)	c (52)	d (54)	e (52)	f (54)	Min()
U4CE1 (1)	28+1+0+50/10	28+1+0+50/5	52+1+200/10+0	54+1+200/10+50/5	52+1+200/10+0	54+1+200/10+50/10	34(a)
U4CE2 (2)	28+2+200/10+0	28+2+200/10+50/5	52+2+0+50/10	54+2+0+50/5	52+2+200/5+50/10	54+2+200/5+0	59(в)
U4CE3 (2)	28+2+200/10+50/5	28+2+200/10+0	52+2+200/5+50/10	54+2+200/5+0	52+2+0+50/10	54+2+0+50/5	59(д)

Step 7. Select the minimum value of objective function for block 3.

According to calculations

min f(x) = 34 for (U4CE1, U2CE1U3CE2, U1CE1).

V. The Strategy Of Input Queue Of Tasks Processing

We have considered the issues of planning a separate task represented as a workflow at an affordable set of heterogeneous resources. In this section, we will discuss approaches to processing and scheduling of various types of tasks coming into a single input queue of workflow management system.

In [13] the following existing multiple workflow scheduling strategies are presented:

1) Scheduling and execution DAGS that are in the input queue one after another. The disadvantage of this approach is the ineffective grid resources utilization, inability to reflect the priorities and the required level of QoS.
2) Scheduling and execution of DAGs in accordance with the criterion of total estimated runtime. Processing order may be different: the priority for tasks with a minimum execution time or the maximum. Such approach does not solve the problem of effective resources utilization and QoS considering.
3) Combining multiple DAGs into a single DAG with a further usage of existing methods of single workflow task scheduling in a heterogeneous environment.

The four main approaches of merging DAGS are determined:

1) Combining DAGS by adding a new entry and new exit “empty” nodes (C1);
2) A composite graph is created in the same way as before, but the scheduling is made by the levels for independent parallel tasks (level-based ordering) (C2);
3) Scheduling and execution of different computational units of workflow tasks occur in the style of round-robin: if on the previous step the task of one workflow was planned and carried out, then on the next step it will be considered as the ready task of another workflow (C3).
4) When combining DAGs into a single workflow structure the estimated execution time of workflow is taken into account and merging by introducing additional nodes occurs at the appropriate level (C4).

Two fairness policies of resources distribution based on calculating the delay of each workflow while choosing the next workflow for scheduling have been introduced in [13].

However the merging DAGs approaches and fairness policies can be applied only to the workflows with the same priorities.

In [14] a ServeOnTime strategy is proposed and its efficiency in comparison with the classical approach FCFS is shown. The strategy is based on adding the new arrived workflow task to the exiting task of executing workflow. Such an approach ignores the QoS requirements, and underutilized resources associated with the occurrence of "gaps" (waiting for completion of the tasks of the previous level and data transfer). In [15] a GapSearchScheduling algorithm is presented. The algorithm is based on finding and filling such gaps by tasks, the execution time of which is less than the gap size.

In [16] the input queue deadline coordinator structure is presented. The deadline driven (DD) coordinator orders DAGs considering deadlines specified by users. DAG with earlier deadline is processed first. DAG priority is computed as inversely to deadline value. The DD-coordinator should verify that the deadline is realistic.

However, the solution is not complete. Deadline set by the user must be considered in relation to the estimated execution time and the arrival time of all tasks.

We suggest the following scheme of tasks priority evaluation:

$$P = U_p + 1/t_{in} \qquad (6)$$

where U_p – user task priority set by the policy of appropriate virtual organization;

t_{in} –arrival time of task queued for execution.

When sending a task to perform, the user can set the desired values for the following QoS parameters: restriction to the task time execution (deadline), cost limit of computing (maximal cost), as well as the significance of the coefficients of these parameters.

Taking into account the dynamicity of the grid environment structure, compliance with user-defined QoS-parameters can not be guaranteed, but finding the optimal solutions based on established significance coefficients is guaranteed. Defining the actual values of QoS parameters is possible by the use of simulation model of task execution process.

Guaranteed compliance deadline is possible only for the tasks submitted with advanced reservation policy, and the preliminary assessment time regarding to the task execution time is set by the workflow management system administrator. In the case of low QoS level of available resources replication approach can be used in addition.

Internal xml task specification format has been developed. The format allows describing the tasks of different types, as well as to determine the volume of data transferred between the computational units of workflow task and periodic synchronization between the parallel blocks. When the task is sent to a specific computing resource the task format is converted to those required by the corresponding middleware.

There are static and dynamic approaches to task scheduling. Static approach assumes the availability of information about the current state of grid-network resources and sequence blocks execute tasks prior to computation. Dynamic approach takes into account the dynamic grid-network resources, as well as handles branching in the structure of the problem. However, this greatly complicates the planning process.

The paper introduces the use of a hybrid approach to planning, which is to use static methods for primary distribution, followed by the dynamic regulation of the primary distribution, taking into account the dynamics of the task and the state of network resources. Such a scheme is implemented at the level of the metascheduler through periodic survey of the state of the network resources, control units perform tasks and rescheduling tasks when needed.

The system is supposed to have the following task queues.

1) Single Block And Data Parallel Queue – contains the tasks of the first and the second type. In case of the resource failure, the task is resubmitted to the same queue with the highest priority.
2) Workflow Queue – contains workflow tasks.
3) Workflow Tasks Rescheduling Queue – contains computational units of different workflow tasks requiring rescheduling. Computing unit of any workflow is put into this queue if the resource that was scheduled for the unit failed.

Queue processing and tasks scheduling is made in the order of their priorities. Rescheduling is processed first, and the rescheduling task is assigned to a suitable free resource or the nearest "gap" in the schedule of resource employment.

If several workflow tasks have the same priority, they are merged into a single DAG according to policy C4.

Single DAG scheduling is made using the method presented in Section 5, with further drafting task's schedule taking into account the synchronization between units and graphics of resources employment.

While scheduling the workflow if the amount of the free resources is less than the width of the DAG then the estimated execution time on the set of available resources is compared with the estimated execution time on the greatest possible variety of resources. If the difference in execution time is less than the waiting time of deallocation, the task is assigned to the available resources, or the task is waiting for the release of resources employed and the optimal schedule for its implementation since the liberation of resources is prepared.

The tasks of the first and second type are assigned either to a suitable free resource, or to the nearest appropriate "gap" in the graph of the resource. "Gap" is considered appropriate if the estimated execution time of the task is less than gap size of not more than 80%.

VI. Scheduling Algorithms Effectiveness Evaluation

The experiments were carried out only for the analysis of the effectiveness of the proposed single DAG scheduling method on the available set of resources. Effectiveness evaluation of the proposed strategy for processing the queue of tasks taking into account the dynamics of their implementation requires additional research.

To investigate the properties of scheduling algorithms the GridSim toolkit [17] was expanded by adding new entities required for modeling the processes of planning and execution of workflows in the grid-environment. Implemented modules class diagrams are shown in Fig. 4-5.

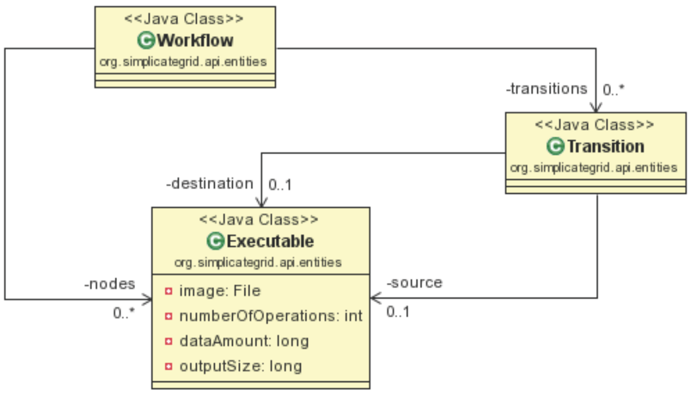
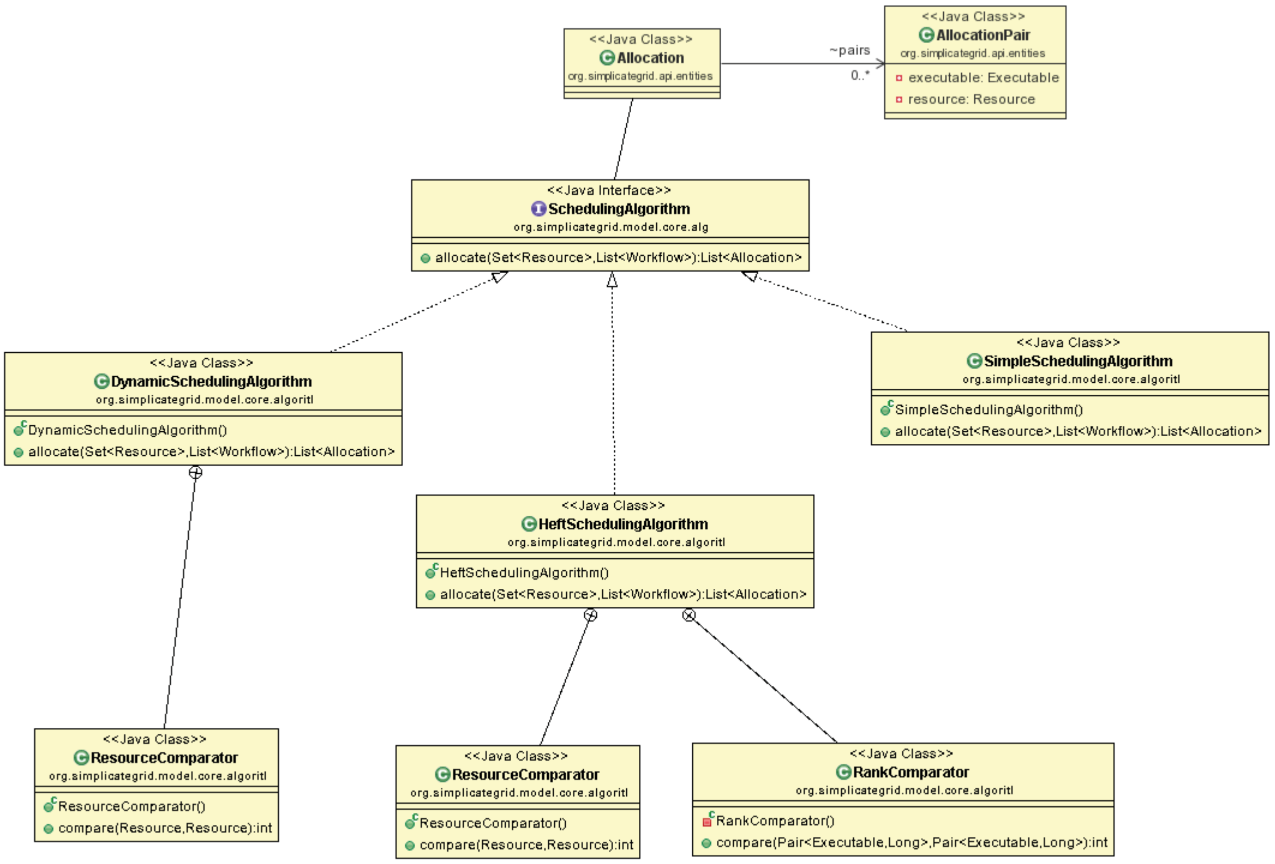


Fig. 4 the class diagram of the workflow specification module



<<Java Class>>
Allocation
org.simplicategrid.api.entities
~pairs
0..*
<<Java Class>>
AllocationPair
org.simplicategrid.api.entities
executable: Executable
resource: Resource
<<Java Interface>>
SchedulingAlgorithm
org.simplicategrid.model.core.alg
allocate(Set<Resource>,List<Workflow>):List<Allocation>
<<Java Class>>
DynamicSchedulingAlgorithm
org.simplicategrid.model.core.algoritl
DynamicSchedulingAlgorithm()
allocate(Set<Resource>,List<Workflow>):List<Allocation>
<<Java Class>>
SimpleSchedulingAlgorithm
org.simplicategrid.model.core.algoritl
SimpleSchedulingAlgorithm()
allocate(Set<Resource>,List<Workflow>):List<Allocation>
<<Java Class>>
HeftSchedulingAlgorithm
org.simplicategrid.model.core.algoritl
HeftSchedulingAlgorithm()
allocate(Set<Resource>,List<Workflow>):List<Allocation>
<<Java Class>>
ResourceComparator
org.simplicategrid.model.core.algoritl
ResourceComparator()
compare(Resource,Resource):int
<<Java Class>>
ResourceComparator
org.simplicategrid.model.core.algoritl
ResourceComparator()
compare(Resource,Resource):int
<<Java Class>>
RankComparator
org.simplicategrid.model.core.algoritl
RankComparator()
compare(Pair<Executable,Long>,Pair<Executable,Long>):int

Fig. 5 the class diagram of the metascheduler module

The experiments were carried out for randomly generated workflow tasks of varying complexity. The example of randomly generated graph is shown in Fig. 6.

Obviously, the real workflows of specific domains have fewer nodes and links. However, the use of more complex workflows allows identifying bottlenecks in the studied algorithms.

The grid-environment experimental model was presented by four compute nodes with the following characteristics: CE1 = {10 mips, 100, 0, {100, 50}}; CE2 = {25 mips, 100, 0, {100, 20}}; CE3 = {35 mips, 100, 0, {80, 40}}; CE4 = {47 mips, 100, 0, {100, 30}}.

The effectiveness of a scheduling algorithm in the experiments was determined by: 1) the objective function value; 2) the computational complexity of the algorithm.

In order to simplify the objective function was defined by task execution time, excluding the economic costs.

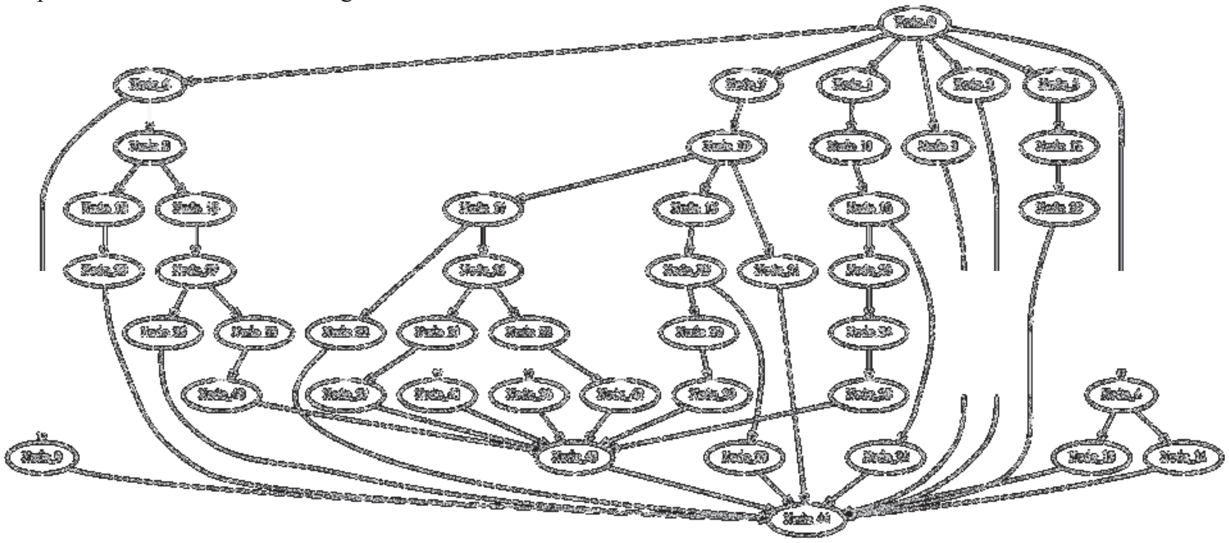


Fig. 6 generated workflow structure example

Table 5 shows the results of the experiments for the tasks of different complexity and the following scheduling algorithms:

1 - heuristic algorithm HEFT, 2 - scheduling algorithm based on dynamic programming method, 3 - random selection of the resource to perform the task of computing unit, ready to run, excluding the cost of accommodation.

Table 5. Experimental results

Number of nodes	Number of links	Number of levels	Scheduling time, ms	Runtime, s	Algorithm	The ratio of the (number of nodes / number of links)
10	13	5	3	518,4	1	0,769231
25	34	7	9	6859,91	1	0,735294
50	72	9	30	7241,28	1	0,694444
10	13	5	5	408	2	0,769231
25	34	7	12	5833,44	2	0,735294
50	72	9	43	6167,76	2	0,694444
10	13	5	1	576	3	0,769231
25	34	7	5	7873,92	3	0,735294
50	72	9	20	8203,2	3	0,694444

Fig. 7-8 shows the results of the algorithms performance criteria evaluation depending on the complexity of the workflow structure. Proposed algorithm showed a higher efficiency for the workflow runtime criterion. Scheduling time of the proposed method is higher than for other algorithms, but it is incomparably less the workflow runtime in the grid-environment that justifies the appropriateness of the proposed solution.

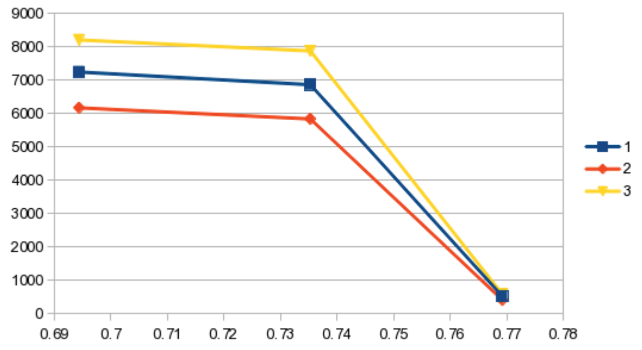


Fig. 7 execution time depending to the complexity of the workflow structure

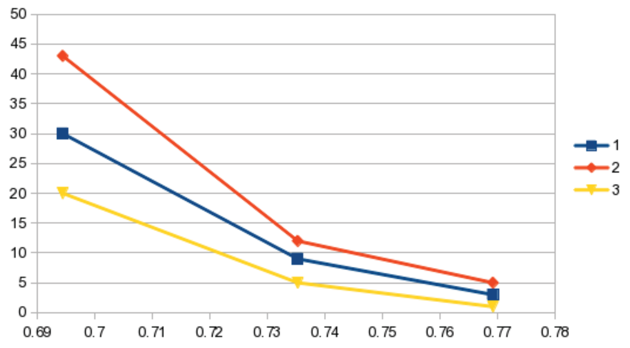


Fig. 8 scheduling time depending to the complexity of the workflow structure

VII. Conclusion

The paper discusses the features of using grid environment to perform various types of computational tasks of high dimension. The metascheduler structure of the centralized workflow management system and two stage scheduling strategy that takes into account QoS requirements, the dynamicity execution and structural features of the task were proposed. The dynamic programming method application to the workflow scheduling problem was shown in the paper. Experimental results proved the effectiveness of the proposed workflow scheduling method. The effectiveness evaluation of the proposed queue processing and rescheduling strategy where not presented in the paper and require additional research.

References

[1] Forti A. DAG Scheduling for grid computing systems / A. Forti // Ph.D. Thesis, University of Udine, Department of Mathematics and Computer Science. – Italy, 2005 – 2006. – P. 43-46, 52-55.

[2] Kazymyr V. Grid workflow design and management system / V. Kazymyr, O. Prila, V. Rudyi // International Journal “Information Technologies & Knowledge”. – 2013. – Vol. 7, N 3. – P. 241 – 255.

[3] Gerasoulis A. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors / A. Gerasoulis, T. Yang // Journal of Parallel and Distributed Computing. – 1992. – N 16. – P. 276 – 291.

[4] Tompkins M.F. Optimization techniques for task allocation and scheduling in distributed multi-agent operations / M.F Tompkins // Diss. Massachusetts Institute of Technology. – 2003. – P. 20-23.

[5] Job Submission Description Language (JSDL) Specification, Version 1.0, GFD-R.136. – 2008. – P. 5-10.

[6] Extended Resource Specification Language, Reference Manual for ARC versions 0.8 and above, Nordugrid-Manual-4. – 2013. – P. 13-28.

[7] Job description language attributes specification for the gLite Workload Management System, WMS-JDL.doc. – 2011. – P.7-10, 38-40.

[8] Yu J. Workflow Scheduling Algorithms for Grid Computing, Metaheuristics for Scheduling in Distributed Computing Environments / Yu J., Buyya R., Ramamohanarao K.; Xhafa F., Abraham A. (Ed.). – Berlin, Germany: Springer, 2008. – P. 111-149.

[9] Fangpeng D. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems / D. Fangpeng, Akl.G. Selim // School of Computing, Queen’s University Kingston, Ontario, Technical Report. – 2006. – N 504. – P. 7-32.

[10] Liou J. CASS: an efficient task management system for distributed memory architectures / J. Liou, M.A. Palis // International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '97). - Taipei, Taiwan, 1997. – P 289 – 295.

[11] Bajaj R. Improving Scheduling of Tasks in a Heterogeneous Environment / R. Bajaj, D.P. Agrawal // IEEE Transactions on Parallel and Distributed Systems. – 2004. – Vol. 15, N 2. – P. 107 – 118.

[12] Introduction to algorithms, third ed. / Thomas H.C., Leiserson C.E., Ronald L.R. [et al.]. – [3 ed.]. – Cambridge, Massachusetts London, England: The MIT Press, 2009. – P. 357 – 414.

[13] H. Zhao, R. Sakellariou: Scheduling Multiple DAGs onto Heterogeneous Systems. Proceedings of the 20th international conference on Parallel and distributed processing, p.159-159, April 25-29, 2006.

[14] L. Zhu, Z. Sun, W. Guo, Y. Jin, W. Sun, W. Hu: Dynamic Multi DAG Scheduling Algorithm for Optical Grid Environment. Network Architectures, Management, and Applications, V 6784(1), 2007.

[15] Bittencourt, L.F., Madeira, E.R.M.: Towards the Scheduling of Multiple Workflows on Computational Grids. Journal of Grid Computing, 8, pp. 419–441, 2010.

[16] Melnyk A. Multiple DAGs Scheduling with Deadline Driven Coordinator in Grid / A. Melnyk // Second International Conference “Cluster Computing”. – Lviv, Ukraine, 2013. – June 3–5. – P. 127 – 130.

[17] Buyya R. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing / R. Buyya, M. Manzur // The Journal of Concurrency and Computation: Practice and Experience (CCPE). – 2002. – Vol. 14, Is. 13–15. – P. 1179 – 1219.