Coordination of Application-Level and Job-Flow Scheduling Techniques in Distributed Computing

Victor V. Toporkov
Computer Science Department, Moscow Power Engineering Institute,
ul. Krasnokazarmennaya 14, Moscow, 111250 Russia
Phone: +7(495)3627145 Fax: +7(495)3625506
E-mail: ToporkovVV@mpei.ru

ABSTRACT

This paper presents an integrated approach for scheduling in distributed computing with strategies as sets of job supporting schedules generated by a critical works method. The strategies are implemented using a combination of job-flow and application-level techniques of scheduling within virtual organizations of Grid. Applications are regarded as compound jobs with a complex structure containing several tasks co-allocated to processor nodes. The choice of the specific schedule depends on the load level of the resource dynamics and is formed as a resource request, which is sent to a local batch-job management system. We propose scheduling framework and compare diverse types of scheduling strategies using simulation studies.

Key Words**:** distributed computing, scheduling, metascheduler, strategy, job, task, work.

1. Introduction

The fact that a distributed computational environment is heterogeneous and dynamic along with the autonomy of processor nodes makes it much more difficult to manage and assign resources for job execution at the required quality level [1].

When constructing a computing environment based on the available resources, e.g. in the model which is used in X-Com system [2], one normally does not create a set of rules for resource allocation as opposed to constructing clusters or Grid-based virtual organizations. This reminds of some techniques, implemented in Condor project [3, 4]. Non-clustered Grid resource computing environments are using similar approach. For example, @Home projects which are based on BOINC system are using cycle stealing, i.e. either idle computers or idle cycles of a specific computer.

Another still similar approach is related to the management of distributed computing based on resource broker assignment [5-11]. Besides Condor project [3, 4], one can also mention several application-level scheduling projects: AppLeS [6], APST [7], Legion [8], DRM [9], Condor-G [10], and Nimrod/G [11].

It is known, that scheduling jobs with independent brokers, or application-level scheduling, allows adapting resource usage and optimizing a schedule for the specific job, for example, decreasing its completion time. Such approaches are important, because they take into account details of job structure and users resource load preferences [5]. However, when independent users apply totally different criteria for application optimization along with job-flow competition, it can degrade resource usage integral quality performance, system throughput, and processor nodes load balance and job completion time.

Alternative way of scheduling in distributed computing based on virtual organizations includes a set of specific rules for resource use and assignment that regulates mutual relations between users and resource owners [1]. In this case only job-flow level scheduling and allocation efficiency can be increased. Grid-dispatchers [12] or metaschedulers are acting as managing centres like in the GrADS project [13].

Inseparability of the resources makes it much more complicated to manage jobs in a virtual organization, because the presence of local job-flows launched by owners of processor nodes should be taken into account. Dynamical load balance of different job-flows can be based on economical principles [14] that support fairshare division model for users and owners. Actual job-flows presence requires forecasting resource state and their reservation [15], for example by means of Maui cluster scheduler simulation approach or methods, implemented in systems such as GARA, Ursala, and Silver [16].

The above-mentioned works are related to either job-flow scheduling problems or application-level scheduling.

Fundamental difference between them and the approach described is that the resultant dispatching strategies are based on the integration of job-flows management methods and compound job scheduling methods on processor nodes. It allows increasing the quality of service for the jobs and distributed environment resource usage efficiency.

It is considered, that *the job* can be compound (multiprocessor) and *the tasks*, included in the job, are heterogeneous in terms of computation volume and resource need. In order to complete the job, one would co-allocate the tasks to different nodes. Each task is executed on a single node and it is supposed, that the local management system interprets it as a job accompanied by a resource request.

On one hand, the structure of the job is usually not taken into account. The rare exception is the Maui cluster scheduler [16], which allows for a single job to contain several parallel, but homogeneous (in terms of resource requirements) tasks. On the other hand, there are several resource-query languages. Thus, JDL from WLMS defines alternatives and preferences when making resource query, ClassAds extensions in Condor-G [10] allows forming resource-queries for dependant jobs. The execution of compound jobs is also supported by WLMS scheduling system of gLite platform, though the resource requirements of specific components are not taken into account.

What sets our work apart from other scheduling research is that we consider coordinated application-level and job-flow management as a fundamental part of the effective scheduling strategy within the virtual organization.

Environment state of distribution, dynamical state of its configuration, user's and owner's preferences cause the need of building multifactor and multicriteria job managing strategies [17-20]. Availability of heterogeneous resources, data replication policies [12, 21, 22] and multiprocessor job structure for efficient co-allocation between several processor nodes should be taken into account.

In this work, the multicriteria strategy is regarded as *a set of supporting schedules* in order to cover possible events related to resource availability.

The outline of the paper is as follows.

In section 2, we provide details of application-level scheduling with a critical works method and strategies as sets of possible supporting schedules.

Section 3 presents a framework for integrated job-flow and application-level scheduling.

Simulation studies of coordinated scheduling techniques and results are discussed in Section 4.

We conclude and point to future directions in Section 5.

2. Application-Level Scheduling Based on a Critical Works Method

The application-level scheduling strategy is a set of possible supporting schedules for all tasks in the job [18]. Figure 1 shows some examples of job graphs in strategies with different degrees of distribution, task details, and data replication policies [19].

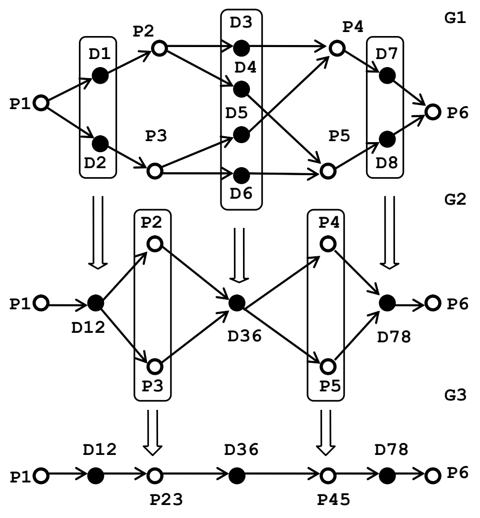


Figure 1: Examples of job graphs

The first type strategy S1 allows scheduling with fine-grain computations and multiple data replicas, the second type strategy S2 is one with fine-grain computations and a bounded number of data replicas, and the third type S3 implies coarse-grain computations and constrained data replication. The vertices P1, …, P6, P23, and P45 correspond to tasks, while D1, …, D8, D12, D36, and D78 correspond to data transmissions. The transition from graph G1 to graphs G2 and G3 is performed through lumping of tasks and reducing of the parallelism level.

The job graph is parameterized by prior estimates of the duration Tij of execution of a task Pi for a processor node nj of the type j, of relative volumes Vij of computations on a processor of the type j, etc. (Table 1).

It is to mention, such estimations are also necessary in several methods of priority scheduling including backfilling in Maui cluster scheduler.

Table 1: User's task estimations

Tij, Vij	Tasks					
	P1	P2	P3	P4	P5	P6
Ti1	2	3	1	2	1	2
Ti2	4	6	2	4	2	4
Ti3	6	9	3	6	3	6
Ti4	8	12	4	8	4	8
Vij	20	30	10	20	10	20

Figure 2 shows fragments of strategies of types S1, S2, and S3 for jobs in Fig. 1. The duration of all data transmissions is equal to one unit of time for G1, while the transmissions D12 and D78 require two units of time and the transmission D36 requires four units of time for G2 and G3.

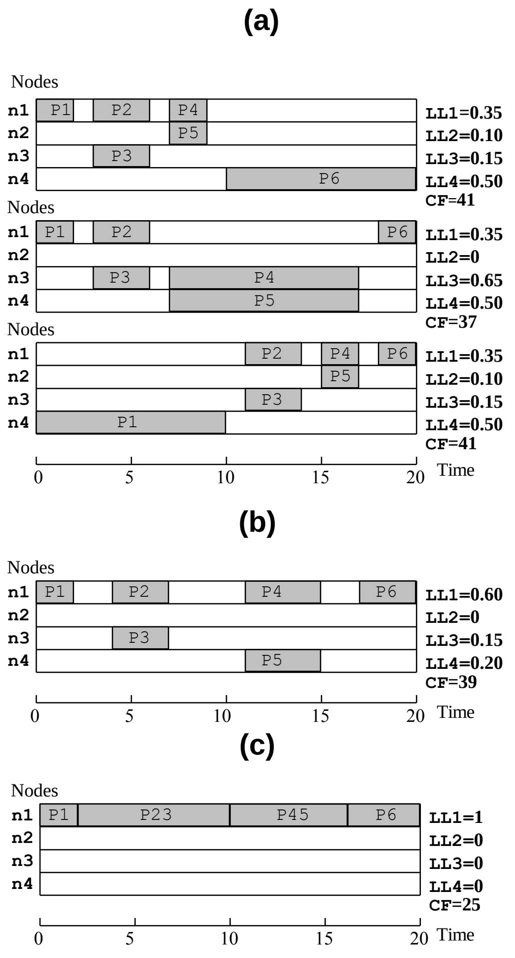


Figure 2. Fragments of scheduling strategies S1 (a), S2 (b), S3 (c)

We assume that the lumping of tasks is characterized by summing of the values of corresponding parameters of constituent subtasks (see Table 1).

Supporting schedules in Fig. 2, a present a Pareto-optimal strategy of the type S1 for tasks Pi, i=1, …, 6 in G1. The Pareto relation is generated by a vector of criteria CF, LLj, j=1, …, 4.

A job execution cost-function CF is equal to the sum of Vij/Ti, where Ti is the real load time of processor node j by task Pi rounded to nearest not-smaller integer. Obviously, actual solving time Ti for a task can be different from user estimation Tij (see Table 1).

The processor node load level LLj is the ratio of the total time of usage of the node of the type j to the job run time.

Schedules in Fig. 2, b and Fig. 2, c are related to strategies S2 and S3.

Strategies are generated with *a critical works method* [20].

The gist of the method is a multiphase procedure. The first step of any phase is scheduling of *a critical work* – the longest (in terms of estimated execution time Tij for task Pi) chain of unassigned tasks along with the best combination of available resources. The second step is resolving collisions cased by conflicts between tasks of different critical works competing for the same resource.

For example, there are four critical works 12, 11, 10, and 9 time units long (including data transfer time) on fastest processor nodes of the type 1 for the job graph G1 in Fig. 1, a (see Table 1):

(P1, P2, P4, P6), (P1, P2, P5, P6),

(P1, P3, P4, P6), (P1, P3, P5, P6).

The schedule with CF=37 has a collision (see Fig. 2, a), which occurred due to simultaneous attempts of tasks P4 and P5 to occupy processor node n4. This collision is further resolved by the allocation of P4 to the processor node n3 and P5 to the node n4. Such reallocations can be based on virtual organization economics – in order to take higher performance processor node, user should "pay" more. Cost-functions can be used in economical models [14] of resource distribution in virtual organizations. It is worth noting that full costing in CF is not calculated in real money, but in some conventional units (quotas), for example like in corporate non-commercial virtual organizations. The essential point is different – user should pay additional cost in order to use more powerful resource or to start the task faster. The choice of a specific schedule from the strategy depends on the state and load level of processor nodes, and data storage policies.

The critical works method was developed for application-level scheduling [19, 20]. However, it can be further refined to build multifactor and multicriteria strategies for job-flow distribution in virtual organizations. This method is based on dynamic programming and therefore uses some integral characteristics, for example total resource usage cost for the tasks that compose the job. However the method of critical works can be referred to the priority scheduling class. There is no conflict between these two facts, because the method is dedicated for task co-allocation of compound jobs.

Priority scheduling based on queues is not an efficient way of multiprocessor jobs co-allocating, in our opinion. Besides, there are several well-known side effects of this approach in the cluster systems such as LL, NQE, LSF, PBS and others. For example, traditional First-Come-First-Serve (FCFS) strategy leads to idle standing of the resources. Another strategy, which involves job ranking according to the specific properties, such as computational complexity, for example Least-Work-First (LWF), leads to a severe resource fragmentation and often makes it impossible to execute some jobs due to the absence of free resources. In distributed environments these effects can lead to unpredictable job execution time and thereby to unsatisfactory quality of service.

In order to avoid it many projects have components that make schedules, which are supported by preliminary resource reservation mechanisms [15, 16]. One to mention is Maui cluster scheduler, where backfilling algorithm is implemented.

Remote Grid resource reservation mechanism is also supported in GARA, Ursala and Silver projects [16]. Here, only one variant of the final schedule is built and it can become irrelevant because of changes in the local job-queue, transporting delays etc.

The strategy is some kind of preparation of possible activities in distributed computing based on supporting schedules (see Fig. 2) and reactions to the events connected with resource assignment and advance reservations [15, 16]. The more factors considered as formalized criteria are taken into account in strategy generation, the more complete is the strategy in the sense of coverage of possible events [18, 19]. The choice of the supporting schedule [20] depends on the utilization state of processor nodes, data storage and relocation policies specific to the environment, structure of the jobs themselves and user estimations of completion time and resource requirements.

3. Metascheduling Framework

In order to implement the effective scheduling and allocation to heterogeneous resources, it is very important to group user jobs into flows according to the strategy type selected and to coordinate job-flow and application-level scheduling.

A hierarchical structure (Fig. 3) composed of a job-flow metascheduler and subsidiary job managers, which are cooperating with local batch-job management systems, is a core part of a scheduling framework proposed in this paper.

The advantages of hierarchically organized resources managers are obvious, e.g., the hierarchical job-queue-control model is used in the GrADS metascheduler [13] and X-Com system [2]. Hierarchy of intermediate servers allows decreasing idle time for the processor nodes, which can be inflicted by transport delays or by unavailability of the managing server while it is dealing with the other processor nodes. Tree-view manager structure in the network environment of distributed computing allows avoiding deadlocks when accessing resources. Another important aspect of computing in heterogeneous environments is that processor nodes with the similar architecture, contents, administrating policy are grouped together under the job manager control.

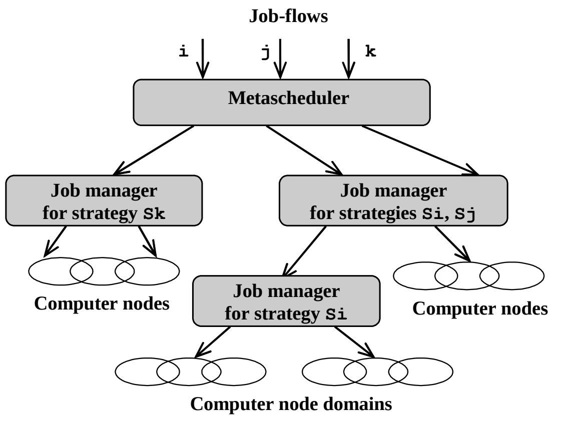

Job-flows
i
j
k
Metascheduler
Job manager for strategy Sk
Job manager for strategies Si, Sj
Computer nodes
Job manager for strategy Si
Computer nodes
Computer node domains

Figure 3: Components of metascheduling framework

Users submit jobs to the metascheduler (see Fig. 3) which distributes job-flows between processor node domains according to the selected scheduling and resource co-allocation strategy `Si`, `Sj` or `Sk`. It does not mean, that these flows cannot “intersect” each other on nodes. The special reallocation mechanism is provided. It is executed on the higher-level manager or on the metascheduler-level. Job managers are supporting and updating strategies based on cooperation with local managers and simulation approach for job execution on processor nodes.

Innovation of our approach consists in mechanisms of dynamic job-flow environment reallocation based on scheduling strategies. The nature of distributed computational environments itself demands the development of multicriteria and multifactor strategies [17, 18] of coordinated scheduling and resource allocation.

The dynamic configuration of the environment, large number of resource reallocation events, user’s and resource owner’s needs as well as virtual organization policy of resource assignment should be taken into account. The scheduling strategy is formed on a basis of formalized efficiency criteria, which sufficiently allow reflecting economical principles [14] of resource allocation by using relevant cost functions and solving the load balance problem for heterogeneous processor nodes. The strategy is built by using methods of dynamic programming in a way that allows optimizing scheduling and resource allocation for a set of tasks, comprising the compound job. In contrast to previous works, we consider the scheduling strategy as *a set of admissible supporting schedules* (see Fig. 2). The choice of the specific variant depends on the load level of the resource dynamics and is formed as a resource query, which is sent to a local batch-job processing system.

One of the important features of our approach is resource state forecasting for timely updates of the strategies. It allows implementing mechanisms of adaptive job-flow reallocation between processor nodes and domains, and also means that there is no more fixed task assignment on a particular processor node. While one part of the job can be sent for execution, the other tasks, comprising the job, can migrate to the other processor nodes according to the updated co-allocation strategy. The similar schedule correction procedure is also supported in the GrADS project [13], where multistage job control procedure is implemented: making initial schedule, its correction during the job execution, metascheduling for a set of applications. Downside of this approach is the fact, that it is based on the creation of *a single schedule*, so the metascheduler stops working when no additional resources are available and job-queue is then set to waiting mode. The possibility of strategy updates allows user, being integrated into economical conditions of virtual organization, to affect job start time by changing resource usage costs. In fact it means that the job-flow dispatching strategy is modified according to new priorities and this provides competitive functioning and dynamic job-flow balance in virtual organization with inseparable resources.

4. Simulations Studies and Results

We have implemented an original simulation environment of the metascheduling framework (see Fig. 3) to evaluate efficiency indices of different scheduling and co-allocation strategies. In contrast to well-known Grid simulation systems such as ChicSim [12] or OptorSim [23], our simulator MetaSim generates multicriteria strategies as a number of supporting schedules for metascheduler reactions to the events connected with resource assignment and advance reservations.

Strategies for more than 12000 jobs with *a fixed completion time* were studied. Every *task of a job* had *randomized* completion time estimations, computation volumes, data transfer times and volumes. These parameters for various tasks had difference which was equal to 2, ..., 3. Processor nodes were selected in accordance to their relative performance. For the first group of “fast” nodes the relative performance was equal to 0.66, …, 1, for the second and the third groups 0.33, …, 0.66 and 0.33 (“slow” nodes) respectively. A number of nodes was conformed to a job structure, i.e. a task parallelism degree, and was varied from 20 to 30.

We have studied the strategies of the types `S1` – with fine-grain computations and active data replication policy; `S2` – with fine-grain computations and a remote data access; `S3` – with coarse-grain computations and static data storage; `MS1` – with fine-grain computations, active data replication policy, and the best- and worst execution time estimations (a modification of the strategy `S1`). The strategy `MS1` is less complete than the strategy `S1` in the sense of coverage of events in distributed environment. However the important point is the generation of a strategy by efficient and economic computational procedures of the metascheduler. The type S1 has more computational expenses than MS1 especially for simulation studies of integrated job-flow and application-level scheduling. Therefore, in some experiments with integrated scheduling we compared strategies MS1, S2, and S3.

4.1 Application-Level Scheduling Study

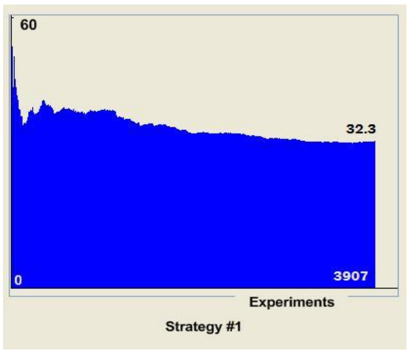
We have conducted the statistical research of the critical works method for application-level scheduling with above-mentioned types of strategies S1, S2, S3.

The main goal of the research was to estimate a forecast possibility for making application-level schedules with the critical works method without taking into account independent job flows. For 12000 randomly generated jobs there were 38% admissible solutions for S1 strategy, 37% for S2, and 33% for S3. This result is obvious: application-level schedules implemented by the critical works method were constructed for available resources non-assigned to other independent jobs. Along with it there is a conflict distribution for the processor nodes that have different performance ("fast" are 2-3 times faster, than "slow" ones): 32% for "fast" ones, 68% for "slow" ones in S1, 56% and 44% in S2, 74% and 26% for S3 (Fig. 4). This may be explained as follows. The higher is the task state of distribution in the environment with active data transfer policy, the lower is the probability of collision between tasks on a specific resource.

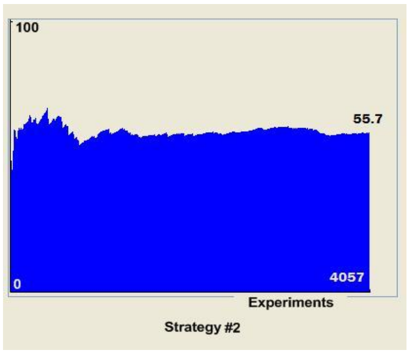
In order to implement the effective scheduling and resource allocation policy in the virtual organization we should coordinate application and job-flow levels of the scheduling.

4.2 Job-Flow and Application-Level Scheduling Study

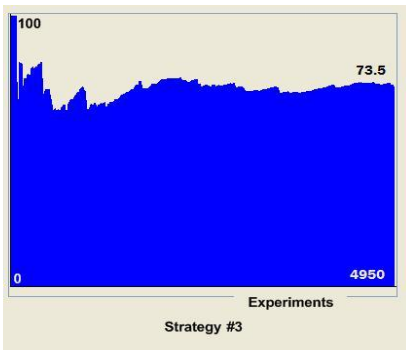
For each simulation experiment such factors as job completion "cost", task execution time, scheduling forecast errors (start time estimation), strategy live-to-time (time interval of acceptable schedules in a dynamic environment), and average load level for strategies S1, MS1, S2, and S3 were studied.



S1



S2



S3

Figure 4: Percentage of collisions for "fast" processor nodes in application-level scheduling

Figure 5 shows load level statistics of variable performance processor nodes which allows discovering the pattern of the specific resource usage when using strategies S1, S2, and S3 with coordinated job-flow and application-levels scheduling.

The strategy S2 performs the best in the term of load balancing for different groups of processor nodes, while the strategy S1 tries to occupy "slow" nodes, and the strategy S3 - the processors with the highest performance (see Fig. 5).

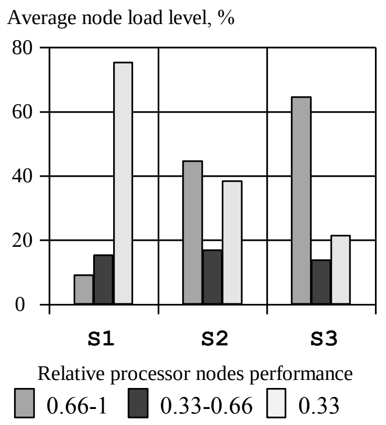


Figure 5: Processor node load level in strategies S1, S2, and S3

Factor quality analysis of S2, S3 strategies for the whole range of execution time estimations for the selected processor nodes as well as modification MS1, when best- and worst-case execution time estimations were taken, is shown in Figures 6 and 7.

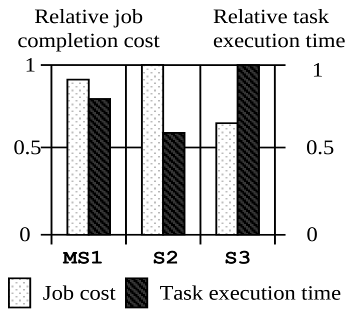


Figure 6: Job completion cost and task execution time in strategies MS1, S2, and S3

Lowest-cost strategies are the "slowest" ones like S3 (see Fig. 6); they are most persistent in the term of time-to-live as well (see Fig. 7).

The strategies of the type S3 try to monopolize processor resources with the highest performance and to minimize data exchanges.

Withal, less persistent are the "fastest", most expensive and most accurate strategies like S2.

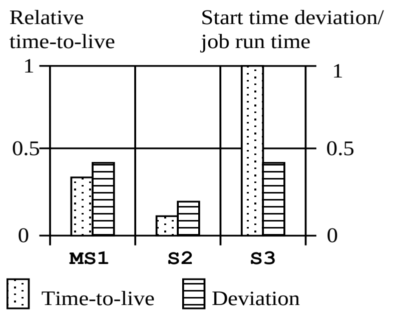


Figure 7: Time-to-live and start deviation time in strategies MS1, S2, and S3

Less accurate strategies like MS1 (see Fig. 7) provide longer task completion time, than more accurate ones like S2 (Fig. 6), which include more possible events, associated with processor node load level dynamics.

5. Conclusions and Future Work

The related works in scheduling problems are devoted to either job scheduling problems or application-level scheduling. The gist of the approach described is that the resultant dispatching strategies are based on the integration of job-flows and application-level techniques. It allows increasing the quality of service for the jobs and distributed environment resource usage efficiency.

Our results are promising, but we have bear in mind that they are based on simplified computation scenarios, e.g., in our experiments we use FCFS management policy in local batch-job management systems. Afore-cited research results of strategy characteristics were obtained by simulation of global job-flow in a virtual organization. Inseparability condition for the resources requires additional advanced research and simulation approach of local job passing and local processor nodes load level forecasting methods development.

Different job-queue management models and scheduling algorithms (FCFS modifications, LWF, backfilling, gang scheduling, etc.) can be used here. Along with it local administering rules can be implemented.

One of the most important aspects here is that advance reservations have impact on the quality of service. Some of the researches (particularly the one in Argonne National Laboratory) show, that preliminary reservation nearly always increases queue waiting time.

Backfilling decreases this time. With the use of FCFS strategy waiting time is shorter than with the use of LWF. On the other hand, estimation error for starting time forecast is bigger with FCFS than with LWF. Backfilling that is implemented in Maui cluster scheduler includes advanced resource reservation mechanism and guarantees resource allocation. It leads to the difference increase between the desired reservation time and actual job starting time when the local request flow is growing.

Some of the quality aspects and job-flow load balance problem are associated with dynamic priority changes, when virtual organization user changes execution cost for a specific resource.

All of these problems require further research.

Acknowledgments. This work was supported by the Russian Foundation for Basic Research (grant no. 09-01-00095) and by the State Analytical Program “The higher school scientific potential development” (project no. 2.1.2/6718).

References:

[1] I. Foster, C. Kesselman, and S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” Int. J. of High Performance Computing Applications, Vol. 15, No. 3, 2001, pp. 200 - 222.

[2] V.V. Voevodin, “The Solution of Large Problems in Distributed Computational Media,” Automation and Remote Control, Vol. 68, No. 5, 2007, pp. 32 - 45.

[3] D. Thain, T. Tannenbaum, and M. Livny, “Distributed Computing in Practice: the Condor Experience,” Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, 2004, pp. 323 - 356.

[4] A. Roy and M. Livny, “Condor and Preemptive Resume Scheduling,” In: J. Nabrzyski, J.M. Schopf, and J.Weglarz (eds.): *Grid resource management. State of the art and future trends*, Kluwer Academic Publishers, 2003, pp. 135 - 144.

[5] V.V. Krzhizhanovskaya and V. Korkhov, “Dynamic Load Balancing of Black-Box Applications with a Resource Selection Mechanism on Heterogeneous Resources of Grid,” In: *9th International Conference on Parallel Computing Technologies*, Springer, Heidelberg, LNCS, Vol. 4671, 2007, pp. 245 - 260.

[6] F. Berman, “High-performance Schedulers,” In: I. Foster and C. Kesselman (eds.): *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, 1999, pp. 279 - 309.

[7] Y. Yang, K. Raadt, and H. Casanova, “Multiround Algorithms for Scheduling Divisible Loads,” IEEE Transactions on Parallel and Distributed Systems, Vol. 16, No. 8, 2005, pp. 1092 - 1102.

[8] A. Natrajan, M.A. Humphrey, and A.S. Grimshaw, “Grid Resource Management in Legion,” In: J. Nabrzyski, J.M. Schopf, and J.Weglarz (eds.): *Grid resource management. State of the art and future trends*, Kluwer Academic Publishers, 2003, pp.145 - 160.

[9] J. Beiriger, W. Johnson, H. Bivens et al., “Constructing the ASCI Grid,” In: *9th IEEE Symposium on High Performance Distributed Computing*, IEEE Press, New York, 2000, pp. 193 - 200.

[10] J. Frey, I. Foster, M. Livny et al., “Condor-G: a Computation Management Agent for Multi-institutional Grids,” In: *10th International Symposium on High-Performance Distributed Computing*, IEEE Press, New York, 2001, pp. 55 - 66.

[11] D. Abramson, J. Giddy, and L. Kotler, “High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?” In: *International Parallel and Distributed Processing Symposium*, IEEE Press, New York, 2000, pp. 520 - 528.

[12] K. Ranganathan and I. Foster, “Decoupling Computation and Data Scheduling in Distributed Data-intensive Applications,” In: *11th IEEE International Symposium on High Performance Distributed Computing*, IEEE Press, New York, 2002, pp. 376 - 381.

[13] H. Dail, O. Sievert, F. Berman et al., “Scheduling in the Grid Application Development Software project,” In: J. Nabrzyski, J.M. Schopf, and J.Weglarz (eds.): *Grid resource management. State of the art and future trends*, Kluwer Academic Publishers, 2003, pp. 73 - 98.

[14] R. Buyya, D. Abramson, J. Giddy et al., “Economic Models for Resource Management and Scheduling in Grid Computing,” J. of Concurrency and Computation: Practice and Experience, Vol. 14, No. 5, 2002, pp. 1507 – 1542.

[15] K. Aida and H. Casanova, “Scheduling Mixed-parallel Applications with Advance Reservations,” In: *17th IEEE International Symposium on High-Performance Distributed Computing*, IEEE Press, New York, 2008, pp. 65 - 74.

[16] D.B. Jackson, “GRID Scheduling with Maui/Silver,” In: J. Nabrzyski, J.M. Schopf, and J.Weglarz (eds.): *Grid resource management. State of the art and future trends*, Kluwer Academic Publishers, 2003, pp. 161 - 170.

[17] K. Kurowski, J. Nabrzyski, A. Oleksiak, and J. Weglarz, “Multicriteria Aspects of Grid Resource Management,” In: J. Nabrzyski, J.M. Schopf, and J.Weglarz (eds.): *Grid resource management. State of the art and future trends*, Kluwer Academic Publishers, 2003, pp. 271 - 293.

[18] V. Toporkov, “Multicriteria Scheduling Strategies in Scalable Computing Systems,” In: *9th International Conference on Parallel Computing Technologies*, Springer, Heidelberg, LNCS, Vol. 4671, 2007, pp. 313 - 317.

[19] V.V. Toporkov and A.S. Tselishchev, “Safety Strategies of Scheduling and Resource Co-allocation in Distributed Computing,” In: *3rd International Conference on Dependability of Computer Systems*, IEEE CS Press, 2008, pp. 152 – 159.

[20] V.V. Toporkov, “Supporting Schedules of Resource Co-Allocation for Distributed Computing in Scalable Systems,” Programming and Computer Software, Vol. 34, No. 3, 2008, pp. 160 – 172.

[21] M. Tang, B.S. Lee, X. Tang, et al., “The Impact of Data Replication on Job Scheduling Performance in the Data Grid,” Future Generation Computing Systems, Vol. 22, No. 3, 2006, pp. 254 - 268.

[22] N.N. Dang, S.B. Lim, and C.K. Yeo, “Combination of Replication and Scheduling in Data Grids,” Int. J. of Computer Science and Network Security, Vol. 7, No. 3, 2007, pp. 304 - 308.

[23] W.H. Bell, D. G. Cameron, L. Capozza et al., “OptorSim – A Grid Simulator for Studying Dynamic Data Replication Strategies,” Int. J. of High Performance Computing Applications, Vol. 17, No. 4, 2003, pp. 403 - 416.