Application-Level and Job-Flow Scheduling: An Approach for Achieving Quality of Service in Distributed Computing

Victor Toporkov

Computer Science Department, Moscow Power Engineering Institute,
ul. Krasnokazarmennaya 14, Moscow, 111250 Russia
ToporkovVV@mpei.ru

Abstract. This paper presents the scheduling strategies framework for distributed computing. The fact that architecture of the computational environment is distributed, heterogeneous, and dynamic along with autonomy of processor nodes, makes it much more difficult to manage and assign resources for job execution which fulfils user expectations for quality of service (QoS). The strategies are implemented using a combination of job-flow and application-level techniques of scheduling and resource co-allocation within virtual organizations of Grid. Applications are regarded as compound jobs with a complex structure containing several tasks. Strategy is considered as a set of possible job scheduling variants with a coordinated allocation of the tasks to the processor nodes. The choice of the specific variant 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.

Keywords: scheduling, resource co-allocation, strategy, job, task, critical 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]. Job management issues including resource allocation and scheduling are addressed by a number of research groups. Dealing with the wide range of different approaches to distributed computing, one can pick out two polar and settled trends. First one is based on the usage of the available and non-dedicated resources, where resource brokers are acting as agents between users and processor nodes [2-4]. Several projects such as AppLeS [5], APST [6], Legion [7], DRM [8], Condor-G [9], Nimrod/G [10] and others, which follow this idea, are often associated with application-level scheduling. Another trend is based on the concept of virtual organizations and is mainly aimed at Grid systems [1, 11, 12]. Both trends have their own advantages and disadvantages.

Resource brokers, that are used in the first trend [2-10] are scalable and flexible and can be adapted to a specific application. However resource distribution dedicated

V. Malyshkin (Ed.): PaCT 2009, LNCS 5698, pp. 350–359, 2009.

to the application-level as well as the usage of different criteria by independent users for the respective job scheduling optimization [12], while considering possible competition with other jobs, may deteriorate such integral characteristics as completion time for the batch-job or resource load level [4, 13].

Forming virtual organizations [1] is essentially bordering the scalability of the scheduling framework, although the set of the specific rules for job-flow assignment and resource consuming [14] allows an overall increase in the efficiency of batch-job scheduling and resource usage. Completion time for single jobs can be longer, because the structures of the jobs for some user-specific needs are not taken into account during the scheduling. In order to control the flow of independent jobs, special metaschedulers, managers, Grid-dispatchers [11, 15] are acting as agents between users and local batch-job systems. One can mention that the alternative Grid resource structure has a single central entity, that is controlling the flow of all jobs and no local schedulers are used (commercial platforms DCGrid, LiveCluster, GridMP, Frontier, volunteer projects @Home [16] and CCS system, which supports dedicated resources).

Distinct from existing Grid scheduling solutions [2-16], our approach supposes techniques of dynamic redistribution of job-flows between processor nodes in conjunction with application-level scheduling. 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 [17] 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 [13]. The rare exception is the Maui cluster scheduler, 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 [9] 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. The choice of the strategy depends on the utilization state of processor nodes [4, 13], data storage and replication policies [11, 18, 19], the job structure (computational granularity and data dependencies), user estimations of completion time, resource requirements, and advance reservations [20].

The outline of the paper is as follows. Section 2 presents a framework for integrated job-flow and application-level scheduling. In section 3, we provide details of our approach based on strategies as sets of possible supporting schedules. Simulation studies of coordinated scheduling techniques and results are discussed in Section 4. We conclude and point to future directions in Section 5.

2 Scheduling Framework

In order to implement the effective coordinated scheduling [17] and allocation to heterogeneous resources [13], it is very important to group user jobs into flows according to the strategy selected. A hierarchical structure (Fig. 1) 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 [15]. 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 node manager control.

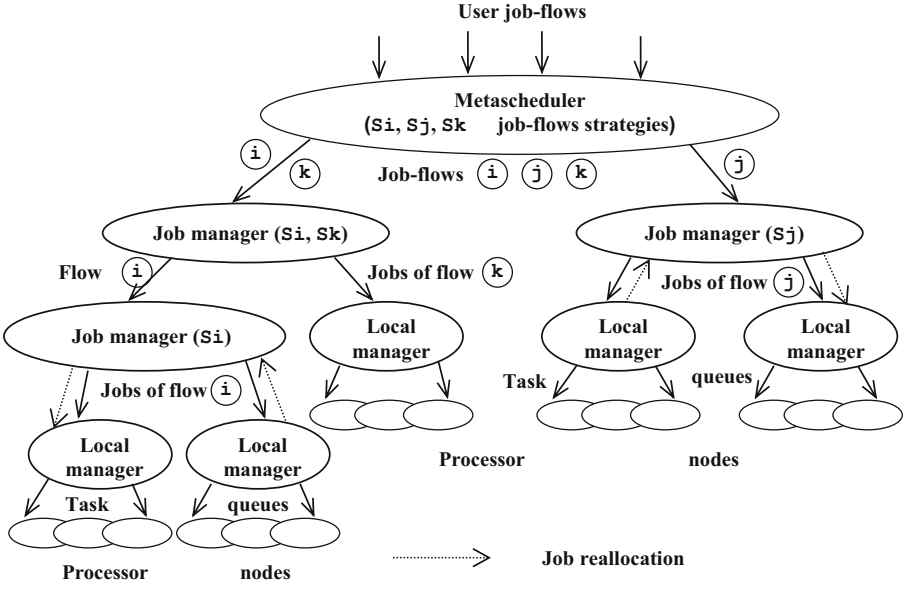


Fig. 1. Hierarchical structure of the scheduling framework

Users submit jobs to the metascheduler (see Fig. 1) 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 [21] and multifactor [22] strategies 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 [23] in a way that allows optimizing scheduling and resource allocation for a set of tasks, comprising the compound job.

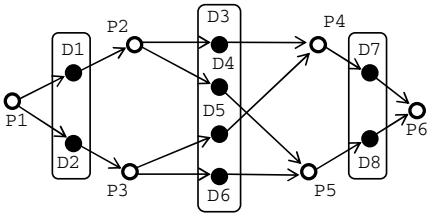
3 Scheduling Strategies

The strategy is a set of possible resource allocation and schedules (distributions) for all N tasks in the job:

```
Distribution:=<<Task 1/Allocation i,[Start 1, End 1]>,
     …, <Task N/Allocation j, [Start N, End N]>>,
```

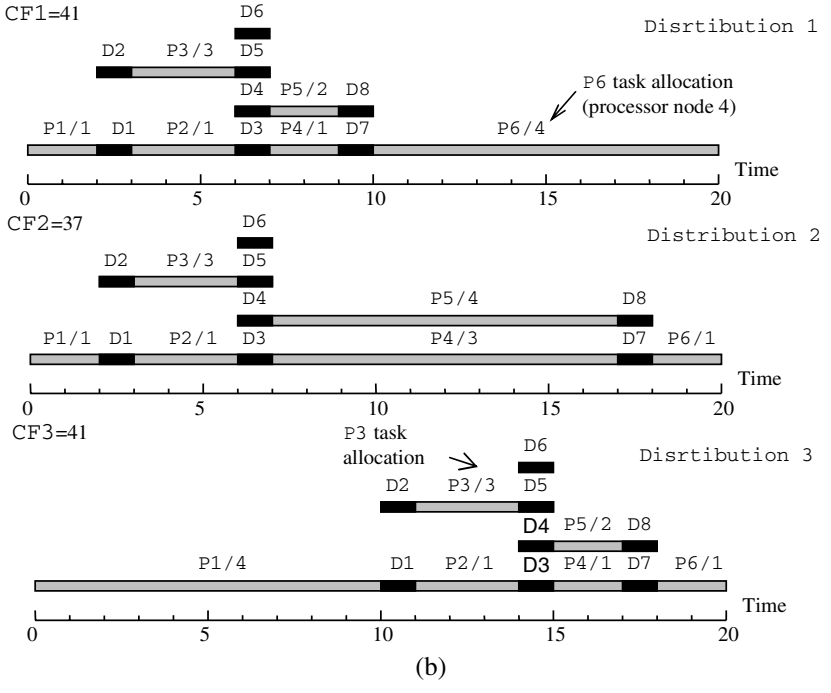
where Allocation i, j is the processor node i, j for Task 1, N; Start 1, N, End 1, N – run time and stop time for Task 1, N execution. Time interval [Start, End] is treated as so called wall time, defined at the resource reservation time [20] in the local batch-job management system.

Figure 2 shows an exemplary information graph of a compound job with user task estimations (Fig. 2, a) and a fragment of the strategy with Distribution variants for schedules and co-allocations (Fig. 2, b). Vertices P1, ..., P6 are corresponding to tasks, D1, ... , D8 – to data transfers. Distribution 2 (see Fig. 2, b) provides minimum of a job execution cost-function CF2=37 equal to the sum of Vij/Ti, i=1,…,N, where Vij is the relative computation volume, and Ti is the real load time of processor node j by task i (rounded to nearest not-smaller integer). Obviously, actual solving time Ti for a task can be different from user estimation Tij. It is to mention, such estimations are also necessary in several methods of priority scheduling including backfilling in Maui cluster scheduler. Cost-functions can be used in economical models [14] of resource distribution in virtual organizations and 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. For that reason Distributions 1, 3 for the job in general (see Fig. 2, b) "cost" more (CF1=CF3=41). The choice of a specific Distribution from the strategy depends on the state and load level of processor nodes, and data storage policies.



Task estimations	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

(a)



(b)

Fig. 2. Job graph with user's estimations (a) and the fragment of the scheduling strategy (b)

A critical works method [23], which was developed for application-level scheduling, 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.

The gist of the method is a multiphase procedure, which is searching for a next critical work – the longest (in terms of estimated execution time) chain of unassigned tasks along with the best combination of available resources, and resolving collisions cased by conflicts between tasks of different critical works competing for the same

resource. As shown on Fig. 2, a, 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:

P1-P2-P4-P6, P1-P2-P5-P6, P1-P3-P4-P6, P1-P3-P5-P6.

Distribution 2 has a collision (see Fig. 2, b), which occurred due to simultaneous attempts of tasks P4 and P5 to occupy processor node 3. This collision is further resolved by the allocation of P4 to the processor node 3 and P5 to the node 4. Such reallocations can be based on virtual organization economics – in order to take higher performance processor node, user should "pay" more. The main positions of the critical works method are described in earlier papers [21-23].

4 Simulations Studies and Results

We have implemented a simulation environment of the scheduling framework to evaluate efficiency indices of different scheduling and co-allocation strategies. In contrast to well-known Grid simulation systems such as ChicSim [11] or OptorSim [24], our simulator 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 with a uniform distribution. 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…066 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.

The strategies types are:

- 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 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 (see Fig. 1). The type S1 has more computational expenses than MS1.

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 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 (Fig. 3, a). This result is obvious: application-level schedules implemented by the critical works method were constructed for available

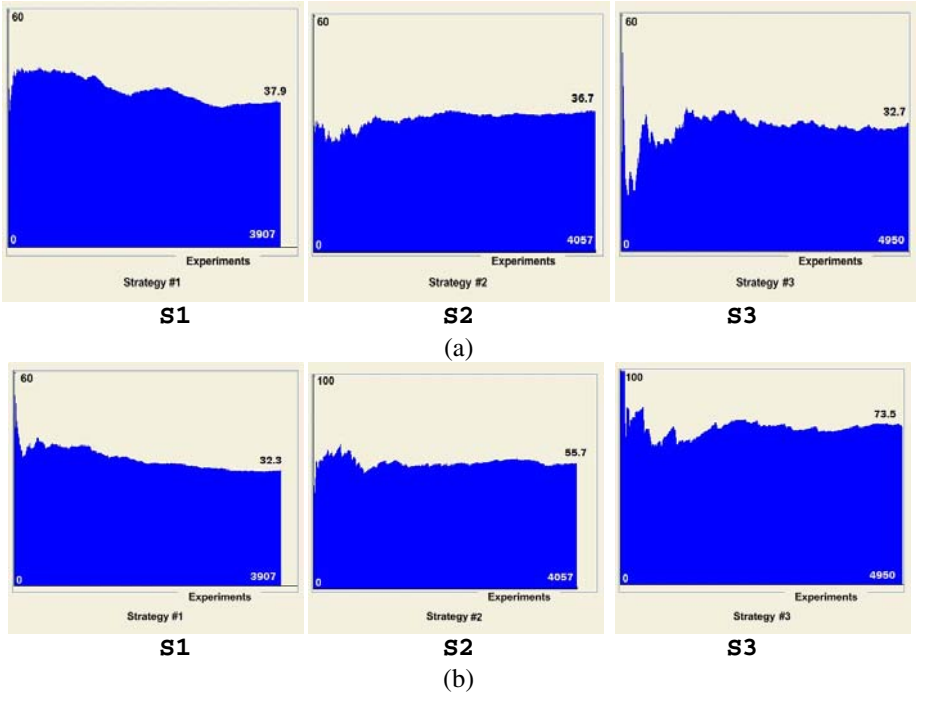


Fig. 3. Simulation results for application-level scheduling: percentage of experiments with admissible schedules (a) and percentage of collisions for "fast" processor nodes (b)

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. 3, b). 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. For each simulation experiment such factors as job completion "cost" (Section 3), task execution time, scheduling forecast errors (start time estimation), strategy live-to-time (time interval of acceptable schedules in a dynamic environment) were studied (Fig. 4). Figure 4, a shows load level statistics of variable performance processor nodes which allows discovering the pattern of the specific resource usage when using strategies 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. 4, a). 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

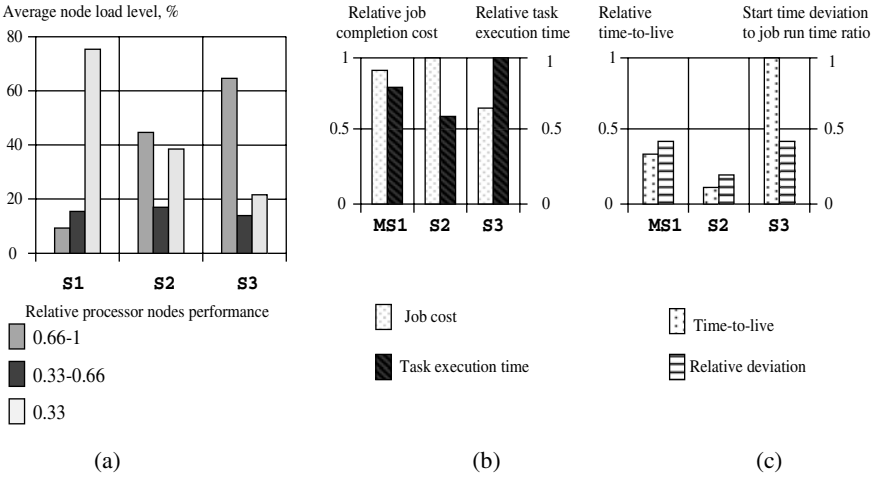


Fig. 4. QoS factors in diverse strategies: processor node load level (a); job completion cost and task execution time (b); time-to-live and start deviation time (c)

worst-case execution time estimations were taken, is shown in Figures 4, b and 4, c. Lowest-cost strategies are the "slowest" ones like `S3` (see Fig. 4, b), they are most persistent in the term of time-to-live as well (see Fig. 4, c). 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`. Less accurate strategies like `MS1` (see Fig. 4, c) provide longer task completion time, than more accurate ones like `S2` (Fig. 4, b), which include more possible events, associated with processor node load level dynamics.

5 Conclusions and Future Work

The existing works in scheduling problems are related to either job 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 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 first-come-first-served (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 can be used (FCFS modifications, least-work-first (LWF), backfilling, gang scheduling etc.) here. Along with it local administering rules can be implemented. One of the most important aspects here

is that advance reservations [17, 20] 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. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. of High Performance Computing Applications 15(3), 200–222 (2001)
2. Thain, D., Tannenbaum, T., Livny, M.: Distributed Computing in Practice: the Condor Experience. Concurrency and Computation: Practice and Experience 17(2-4), 323–356 (2004)
3. Roy, A., Livny, M.: Condor and Preemptive Resume Scheduling. In: Nabrzyski, J., Schopf, J.M., Weglarz, J. (eds.) Grid resource management. State of the art and future trends, pp. 135–144. Kluwer Academic Publishers, Dordrecht (2003)
4. Krzhizhanovskaya, V.V., Korkhov, V.: Dynamic Load Balancing of Black-Box Applications with a Resource Selection Mechanism on Heterogeneous Resources of Grid. In: Malyshkin, V.E. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 245–260. Springer, Heidelberg (2007)
5. Berman, F.: High-performance Schedulers. In: Foster, I., Kesselman, C. (eds.) The Grid: Blueprint for a New Computing Infrastructure, pp. 279–309. Morgan Kaufmann, San Francisco (1999)
6. Yang, Y., Raadt, K., Casanova, H.: Multiround Algorithms for Scheduling Divisible Loads. IEEE Transactions on Parallel and Distributed Systems 16(8), 1092–1102 (2005)
7. Natrajan, A., Humphrey, M.A., Grimshaw, A.S.: Grid Resource Management in Legion. In: Nabrzyski, J., Schopf, J.M., Weglarz, J. (eds.) Grid resource management. State of the art and future trends, pp. 145–160. Kluwer Academic Publishers, Dordrecht (2003)
8. Beiriger, J., Johnson, W., Bivens, H., Humphreys, S., Rhea, R.: Constructing the ASCI Grid. In: 9th IEEE Symposium on High Performance Distributed Computing, pp. 193–200. IEEE Press, New York (2000)
9. Frey, J., Foster, I., Livny, M., Tannenbaum, T., Tuecke, S.: Condor-G: a Computation Management Agent for Multi-institutional Grids. In: 10th International Symposium on High-Performance Distributed Computing, pp. 55–66. IEEE Press, New York (2001)
10. Abramson, D., Giddy, J., Kotler, L.: High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? In: International Parallel and Distributed Processing Symposium, pp. 520–528. IEEE Press, New York (2000)

11. Ranganathan, K., Foster, I.: Decoupling Computation and Data Scheduling in Distributed Data-intensive Applications. In: 11th IEEE International Symposium on High Performance Distributed Computing, pp. 376–381. IEEE Press, New York (2002)
12. Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Multicriteria Aspects of Grid Resource Management. In: Nabrzyski, J., Schopf, J.M., Weglarz, J. (eds.) Grid resource management. State of the art and future trends, pp. 271–293. Kluwer Academic Publishers, Dordrecht (2003)
13. Tracy, D., Howard, J.S., Noah, B., Ladislau, B., et al.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. J. of Parallel and Distributed Computing 61(6), 810–837 (2001)
14. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic Models for Resource Management and Scheduling in Grid Computing. J. of Concurrency and Computation: Practice and Experience 14(5), 1507–1542 (2002)
15. Dail, H., Sievert, O., Berman, F., Casanova, H., et al.: Scheduling in the Grid Application Development Software project. In: Nabrzyski, J., Schopf, J.M., Weglarz, J. (eds.) Grid resource management. State of the art and future trends, pp. 73–98. Kluwer Academic Publishers, Dordrecht (2003)
16. Anderson, D.P., Fedak, G.: The Computational and Storage Potential of Volunteer Computing. In: IEEE/ACM International Symposium on Cluster Computing and Grid, pp. 73–80. IEEE Press, New York (2006)
17. Ioannidou, M.A., Karatza, H.D.: Multi-site Scheduling with Multiple Job Reservations and Forecasting Methods. In: Guo, M., Yang, L.T., Di Martino, B., Zima, H.P., Dongarra, J., Tang, F. (eds.) ISPA 2006. LNCS, vol. 4330, pp. 894–903. Springer, Heidelberg (2006)
18. Tang, M., Lee, B.S., Tang, X., Yeo, C.K.: The Impact of Data Replication on Job Scheduling Performance in the Data Grid. Future Generation Computing Systems 22(3), 254–268 (2006)
19. Dang, N.N., Lim, S.B., Yeo, C.K.: Combination of Replication and Scheduling in Data Grids. Int. J. of Computer Science and Network Security 7(3), 304–308 (2007)
20. Aida, K., Casanova, H.: Scheduling Mixed-parallel Applications with Advance Reservations. In: 17th IEEE International Symposium on High-Performance Distributed Computing, pp. 65–74. IEEE Press, New York (2008)
21. Toporkov, V.: Multicriteria Scheduling Strategies in Scalable Computing Systems. In: Malyshkin, V.E. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 313–317. Springer, Heidelberg (2007)
22. Toporkov, V.V., Tselishchev, A.S.: Safety Strategies of Scheduling and Resource Coallocation in Distributed Computing. In: 3rd International Conference on Dependability of Computer Systems, pp. 152–159. IEEE CS Press, Los Alamitos (2008)
23. Toporkov, V.V.: Supporting Schedules of Resource Co-Allocation for Distributed Computing in Scalable Systems. Programming and Computer Software 34(3), 160–172 (2008)
24. William, H.B., Cameron, D.G., Capozza, L., et al.: OptorSim – A Grid Simulator for Studying Dynamic Data Replication Strategies. Int. J. of High Performance Computing Applications 17(4), 403–416 (2003)