Heuristic Cycle-Based Scheduling with Backfilling for Large-Scale Distributed Environments

Victor Toporkov[1], Anna Toporkova[2],
Alexey Tselishchev[3], Dmitry Yemelyanov[1], and Petr Potekhin[1]

[1] National Research University "MPEI",
ul. Krasnokazarmennaya, 14, Moscow, 111250, Russia
{ToporkovVV,YemelyanovDM,PotekhinPA}@mpei.ru
[2] National Research University Higher School of Economics,
Moscow State Institute of Electronics and Mathematics,
Bolshoy Trekhsvyatitelsky per., 1-3/12, Moscow, 109028, Russia
atoporkova@hse.ru
[3] European Organization for Nuclear Research (CERN),
Geneva, 23, 1211, Switzerland
Alexey.Tselishchev@cern.ch

Abstract. The paper is devoted to comparing the results of an independent job batch scheduling in terms of a virtual organization policy and available resources usage efficiency in large distributed environments like utility Grid. A hybrid approach is proposed on the basis of a cyclic scheduling scheme and backfilling combination. Additionally the paper offers a heuristic shifting procedure which improves jobs execution alternatives selected in the cyclic scheme. The simulation results show that depending on the scheduling efficiency indicator and the level of resource availability each of the approaches is able to provide the best results. Moreover the obtained results are valid under conditions of dynamically varying state of resources and inaccurate user job runtime estimations.

Keywords: Distributed computing, economic scheduling, slot, job, backfilling.

1 Introduction

Some of the most important quality of service (QoS) indicators of a distributed computational environment is a utilization level of the available resources and job start ("response") time. In distributed environments with non-dedicated resources the computational nodes are usually partly utilized by local priority jobs. Thus, the resources available for use can be represented as a set of slots – time intervals during which the individual computational nodes are vacant to execute parts of multiprocessor parallel jobs. Presence of this set of slots (which generally have different start and finish times and difference in performance depending on the node, where the slot is allocated) impedes the problem of a coordinated selection of the resources required to execute the job flow from the computational environment users.

W. Zamojski et al. (eds.), *Proceedings of the Ninth International Conference DepCoS-RELCOMEX*, Advances in Intelligent Systems and Computing 286,
DOI: 10.1007/978-3-319-07013-1_44,

Resource fragmentation also results in a steady decrease of the total computing environment utilization level. Resource management and job scheduling economic models proved to be efficient in such conditions [1-8]. Application-level scheduling, as a rule, does not imply any global resource sharing or allocation policy. Resource brokers [9-15] are usually considered as mediators between users and resource owners. Scheduling and resource management systems in this approach are well-scalable and application-oriented. However, simultaneous application-level scheduling with diverse optimization criteria set by independent users, especially upon possible competition between applications, may deteriorate such QoS characteristics of a distributed environment as total job batch execution time or overall resource utilization. The regulations of a virtual organization (VO) in Grid [16] usually suppose a job flow scheduling. A meta-scheduler or a meta-broker are considered as intermediate chains between the users and a local resource management and job batch processing systems [1, 17-20]. VOs, on one hand, naturally restrict the scalability of resource management systems (though, it is worth remarking here, that there is a good experience [19] of enabling interoperability among meta-schedulers belonging to different VOs). On the other hand, uniform rules of a resource sharing and consumption, in particular based on economic models [1-8], make it possible to improve the job-flow level scheduling and resource distribution efficiency. In some well-known models of a distributed computing environments with non-dedicated resources, only the first fit set of resources is chosen depending on the environment state [18, 21-23], while job scheduling optimization mechanisms are usually not supported. In other models [2, 3, 17] the aspects related to the specifics of the environments with non-dedicated resources (particularly dynamic resource loading, the competition between independent users, users' global and owners' local job flows) are not presented.

In this paper, we propose a combined approach to meta-scheduling in VOs. First of all, we address a problem of an early resources release and "on the fly" rescheduling by combining our original cyclic scheduling scheme (CSS) [24] with backfilling [25]. For overall job-flow execution optimization and a resource occupation time prediction existing schedulers rely on the time specified in the job request, e.g. using Job Submission Description Language (JSDL). However, the reservation time is usually based on the user inaccurate runtime estimates [26]. In case, when the application is completed before the term specified in the resource request, the allocated resources remain underutilized. Second, we introduce a schedule shifting heuristic in CSS. Thus, we outline two main job-flow optimization directions. First, optimal or a suboptimal (under a given VO criteria) scheduling is performed on the basis of a priori information about the computational nodes local schedules and the resource reservation time for each job execution. CSS belongs to this type of systems. Another approach represents scheduling "on the fly" which depends on dynamically updated information about resource utilization. In this case, schedulers are focused on overall resources load maximization and a job start time minimizing. Backfilling may be related to this scheduling model.

The rest of the paper is organized as follows. Section 2 is devoted to analysis of the related works. In Section 3, there is a concept of CSS and backfilling combination as well as of the shifting procedure. Section 4 contains simulation results of the considered scheduling approaches comparison. Finally, section 5 summarizes the paper and describes further research topics.

2 Related Works

Many resource selection and scheduling algorithms, and heuristic-based solutions have been proposed for parallel jobs and tasks with dependencies in distributed environments [3, 18, 19, 21-23, 27-33].

In [3], heuristic algorithms for slot selection, based on user-defined utility functions, are introduced. Slot window allocation is based on the user defined efficiency criterion under the maximum total execution cost constraint. However, the optimization occurs only on the stage of the best found offer selection. The paper [28] presents architecture and an algorithm for performing Grid resources co-allocation without the need for advance reservations based on synchronous subtasks queuing. However, advance reservation is efficient to improve the co-allocation QoS. Advance reservation-based co-allocation algorithms are proposed in [21-23, 29, 30].

First fit resource selection algorithms [21-23] assign any job to the first set of slots matching the resource request conditions without any optimization. Preference-based matchmaking [18] is not focused on the scheduling process. The job is scheduled on the first available resource according to user preferences. In [19], an approach to resource matchmaking among VOs combining hierarchical and peer-to-peer meta-schedulers models is proposed.

The co-allocation algorithms described in [29-31] suppose an exhaustive search and some of them are based on a linear integer programming (IP) [7, 30] or a mixed-integer programming (MIP) model [31]. The co-allocation algorithm presented in [30] uses the 0-1 IP model with the goal of creating reservation plans satisfying user resource requirements. Users can specify a time frame for each resource: the earliest start time, the latest start time and the job duration, where user wants to reserve a time slot. This condition imposes restrictions for slots search only within this time frame. A linear IP-driven algorithm is proposed in [7]. It combines the capabilities of an IP and a genetic algorithm and allows obtaining the best meta-schedule that minimizes the combined cost of all independent users in a coordinated manner. In [31], the authors propose a MIP model which determines the best scheduling for all the jobs in the queue in environments composed of multiple clusters that act collaboratively. The scheduling techniques proposed in [7, 29-33] are efficient compared with other scheduling techniques under given criteria: the minimum processing cost, the overall makespan, resources utilization, load balancing for related tasks [32, 33], etc. However, complexity of the scheduling process is extremely increased by the resources heterogeneity and the co-allocation process, which distributes the tasks of parallel jobs across resource domain boundaries.

In this work, we use algorithms for efficient slot selection based on criteria defined by users, resource owners and VO administrators. The algorithms have linear complexity against the number of all available time-slots and operate on a scheduling interval denoting how far in the future the system may schedule resources [24, 27].

3 A Concept of Combined Cycle-Based Scheduling

CSS was proposed for a model based on a hierarchical job-flow management [24]. Job-flow scheduling is performed in cycles with separate job batches on the basis of dynamically updated computational nodes' local schedules.

Among the major CSS restrictions in terms of an efficient scheduling and resource allocation one may outline the following. First of all, it is not possible to affect execution parameters of an individual job: the search for particular alternatives is performed on the First Fit principle, while choice of the optimal alternatives combination represents only the VO interests. Second, the job batch scheduling is based on an often inaccurate user estimation of a particular job runtime [26]. Third, the job batch scheduling requires allocation of a multiple "nonintersecting" in terms of slots alternatives, and only one alternative is chosen for each job execution.

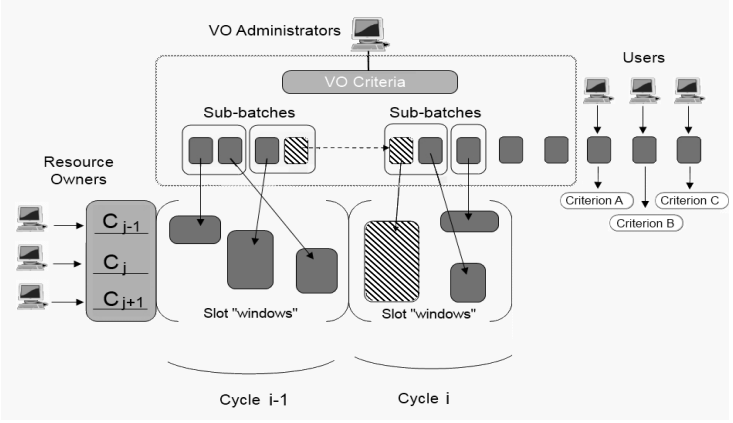

Fig. 1. Cyclic scheduling with batch-slicing

We introduce a modified CSS model: Batch-slicer (Fig. 1). In order to satisfy the user preferences a desirable optimization criterion is introduced into the job request. In Fig. 1: Cj-1, Cj, Cj+1 are slot costs determined by resource owners. We propose initial job batch separation into a set of sub-batches and each sub-batch scheduling at the same given scheduling interval. According to the alternatives search algorithm adopted in CSS [24], at a high resource utilization level the number of the batch jobs' execution alternatives may be relatively small up to just a single alternative for every job. Such a small number of alternatives found may affect the optimal slot

combination selection, and therefore, may reduce overall scheduling efficiency. The job batch "slicing" increases the number of alternatives found for high-priority jobs and diversifies the choice on the slots combination selection stage, and thereby increases the resource sharing efficiency according to VO policy.

Backfilling [25] responds to early resources releases and performs "on the fly" rescheduling which is very important when a user job runtime estimation is significantly different from the actual job execution time. However backfilling has some limitations for distributed computing. The first one is inefficient resource usage by criteria different from average job start time (especially at a relatively low resource load level). The second one is a principal inability to affect the resource sharing quality by defining policies and criteria in VO.

We introduce a combined approach. During every scheduling cycle a set of high priority jobs is allocated from the initial job batch. These jobs are grouped into a separate sub-batch and should be scheduled before other jobs, probably, without compliance with the queue discipline. The scheduling of this sub-batch is further performed by Batch-slicer based on the preliminary known resources utilization schedule. The scheduling of the rest batch jobs is performed by backfilling with the dynamically updated information about the actual computational nodes utilization. The cyclic scheduling method combined with backfilling (Batch-slice-Filling - BSF) combines the main advantages of both Batch-slicer and backfilling, namely the optimization of the most time-consuming jobs execution as well as the efficient resource usage, preferential job execution queue order compliance and a relatively low response time.

A heuristic shifting procedure is proposed for the job execution alternatives selected for advanced reservation. The procedure shifts the alternatives in time towards the beginning of the scheduling interval retaining resource instances in which they are allocated. The shifting procedure is done iteratively for each job of the batch being scheduled. The job selection order is determined according to the start time of the chosen alternatives: first, an attempt to shift the alternatives with the minimal start time is performed. Such order guarantees that when shifting a job all other jobs with an earlier start time are already shifted and hence do not occupy the corresponding nodes. Otherwise, a task with an earlier start time and a lower priority may block the shift of a task with a higher priority and then, in its turn, may be shifted releasing extra slots.

4 Simulation Studies

The experiments are devoted to study scheduling efficiency using the proposed approaches: CSS, BSF, Shifted CSS (CSS with the use of a shifting procedure), and also backfilling (BF). The goal is to compare the scheduling efficiency depending on the number of computational nodes in the domain as well as to investigate schedules

consistency under conditions of inaccurate user job execution time estimations. A series of studies were carried out with the simulation environment [24]. Each experiment includes an input batch of 15 jobs generation as well as the resources structure and local schedules of the computational environment. To analyze the approaches under different conditions the simulation is conducted individually for different numbers of the nodes available {6, 10, 20, 30, 50, 75, 100, 150}. Thus the investigation consists in comparing the scheduling results obtained with the same input data by means of different algorithms.

Scheduling efficiency is considered from the viewpoint of a job batch total slot utilization time T_{proc} minimization, start t_{start} and finish t_{finish} job batch execution times minimization, and minimization of a combined criterion $F = t_{start} + T_{proc}$. For a batch consisting of multiple jobs we consider average parameter values.

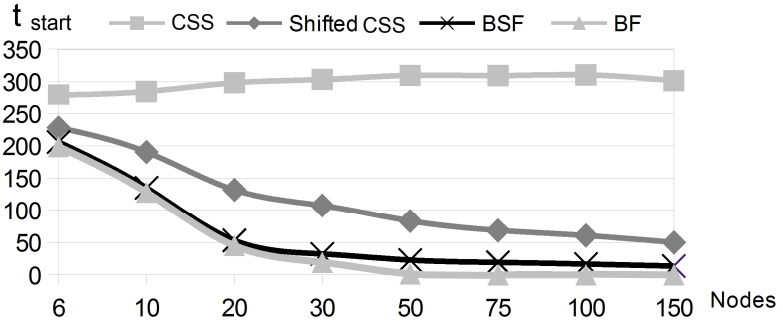

Fig. 2. Average job batch start time

Figure 2 shows average scheduled job start times obtained independently with all considered scheduling approaches depending on the computational environment nodes number. As can be seen from Fig. 2, with increasing amount of available computational nodes backfilling is able to reduce the average job batch start time down to zero (i.e. all batch jobs can start at the very beginning of the scheduling interval). At the same time average job start time obtained with CSS is almost independent of the available nodes number. BSF provides the average job batch start time close to backfilling's by *filling* unused by CSS time slots near beginning of the scheduling interval with relatively low priority jobs. With a relatively large resource level an average job batch start time obtained with Shifted CSS tends to a non-zero value since the most profitable in terms of the optimization criterion resources are generally allocated for more than one job. Thus in case of heterogeneous resource environment it is virtually impossible to start all the batch jobs at the beginning of the scheduling interval using CSS (even with *shifted* variation).

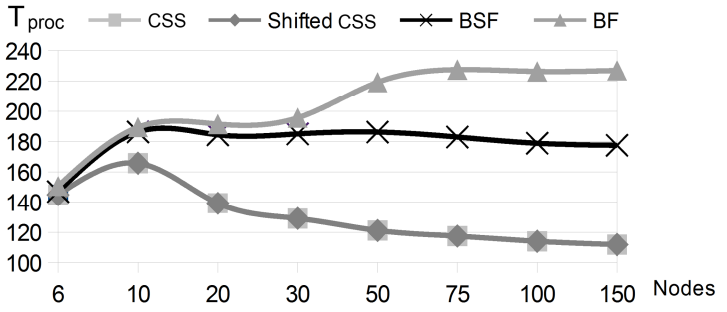

Fig. 3. Average batch jobs processor time usage

Figure 3 shows the advantage of CSS, Shifted CSS, and BSF over backfilling by the VO target optimization criterion T_{proc}. It should be noted that with increasing number of available resources the advantage of CSS and BSF over backfilling also increases. The use of additional heuristics, such as job batch slicing, can provide even greater CSS advantage on the target criterion.

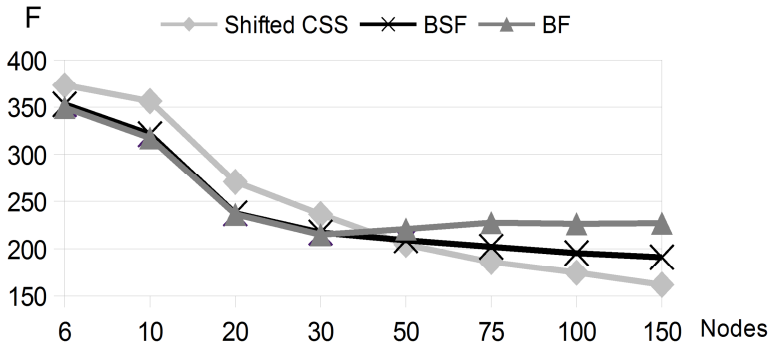

Fig. 4. Average batch jobs F value

Figure 4 shows the value of the combined resource usage efficiency index $F = t_{start} + T_{proc}$. It is important to note that the intersection between the Shifted CSS, BSF and backfilling graphs implies that in case of a relatively low level of available resources backfilling or BSF (they provide almost the same criterion F value) are better as compared with Shifted CSS. With increasing computational environment size Shifted CSS becomes more advantageous in terms of resource usage efficiency.

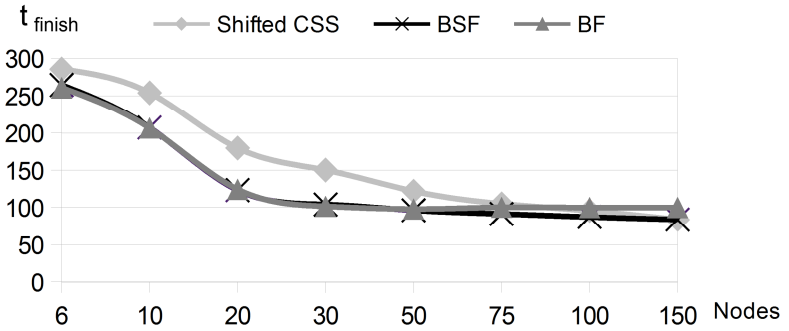

Fig. 5. Average batch jobs finish time

The same conclusion can be drawn if we evaluate the resource usage efficiency by the average batch job finish time (Fig. 5): Shifted CSS provides the best results when computational nodes with a sufficiently large number are available. Thus the use of BSF is justified in virtually any conditions: this combined approach provides competitive to backfilling values of all considered resource usage efficiency indexes, and at the same time optimizes execution performance of the high-priority jobs.

Another experiment studies the scheduling efficiency and consistency when based on user estimated job runtimes in case when these estimations are inaccurate. During the computational environment simulation batch jobs' actual execution time was set as a random variable with uniform distribution, which allows actual job execution time and a user estimation vary by 5 times. Uniform distribution is chosen as it is almost impossible to predict real job execution time on the specified resources. Table 1 contains the simulation results. Results show, that even if the difference between the resource reservation time and the real job execution time is significant the advantage of CSS over backfilling against the VO target optimization criterion not only remains but increases. That is because backfilling does not optimize against criteria different from the start time and implied more compact job packing uses almost all the available resources including those less advantageous against the target criterion.

Table 1. Average batch jobs processor time usage

Approach	Processor time usage	
	Reserved	Real
Backfilling	208	140
CSS	168	112
CSS advantage	19%	20%

5 Conclusions and Future Work

In this work, we compare the scheduling results of a batch of independent jobs in terms of a virtual organization policy and the available resources usage efficiency. Based on a cyclic scheduling scheme and backfilling combination a hybrid approach BSF is proposed. Additionally the shifting procedure is proposed for the alternatives chosen in CSS. The simulation results show that depending on the considered scheduling efficiency index, and depending on the level of the resources available, each of the considered approaches may provide the best results. Backfilling, as a rule, minimizes job start and finish times, while CSS is able, for example, to minimize the job processor time usage (when given the appropriate optimization criterion). In order to ensure compromise scheduling results it is justified to use BSF: scheduling of high priority jobs with CSS and further filling the remaining unassigned resources with backfilling. The results obtained remain valid in a dynamically changing computational environment condition and composition, and in case when user jobs runtime estimations are significantly inaccurate.

Further research will be related to a more precise investigation of dividing the job flow into sub-batches depending on the jobs characteristics and computational environment parameters as well as to studying of rescheduling based on the information about computational nodes current state and performance.

Acknowledgements. This work was partially supported by the Council on Grants of the President of the Russian Federation for State Support of Leading Scientific Schools (SS-362.2014.9), the Russian Foundation for Basic Research (grant no. 12-07-00042).

References

1. Garg, S.K., Konugurthi, P., Buyya, R.: A Linear Programming-driven Genetic Algorithm for Meta-scheduling on Utility Grids. J. Par., Emergent and Distr. Systems 26, 493–517 (2011)
2. Buyya, R., Abramson, D., Giddy, J.: Economic Models for Resource Management and Scheduling in Grid Computing. J. Concurrency and Computation 14(5), 1507–1542 (2002)
3. Ernemann, C., Hamscher, V., Yahyapour, R.: Economic Scheduling in Grid Computing. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2002. LNCS, vol. 2537, pp. 128–152. Springer, Heidelberg (2002)
4. Lee, Y.C., Wang, C., Zomaya, A.Y., Zhou, B.B.: Profit-driven Scheduling for Cloud Services with Data Access Awareness. J. Par. and Distr. Computing 72(4), 591–602 (2012)
5. Garg, S.K., Buyya, R., Siegel, H.J.: Scheduling Parallel Applications on Utility Grids: Time and Cost Trade-off Management. In: 32nd Australasian Computer Science Conference (ACSC 2009), Wellington, New Zealand, pp. 151–159 (2009)
6. Degabriele, J.P., Pym, D.: Economic Aspects of a Utility Computing Service, Trusted Systems Laboratory HP Laboratories Bristol HPL-2007-101. Technical Report, pp. 1-23 (July 3, 2007)

7. Garg, S.K., Yeo, C.S., Anandasivam, A., Buyya, R.: Environment-conscious Scheduling of HPC Applications on Distributed Cloud-oriented Data Centers. J. Parallel and Distributed Computing 71(6), 732–749 (2011)
8. Tesauro, G., Bredin, J.L.: Strategic Sequential Bidding in Auctions Using Dynamic Programming. In: 1st International Joint Conference on Autonomous Agents and Multiagent Systems, Part 2, pp. 591–598. ACM, New York (2002)
9. Thain, D., Tannenbaum, T., Livny, M.: Distributed Computing in Practice: the Condor Experience. J. Concurrency and Computation: Practice and Experience 17(2-4), 323–356 (2004)
10. 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)
11. Yang, Y., Raadt, K., Casanova, H.: Multiround Algorithms for Scheduling Divisible Loads. IEEE Trans. Parallel and Distributed Systems 16(8), 1092–1102 (2005)
12. 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, Boston (2003)
13. Beiriger, J., Johnson, W., Bivens, H.: Constructing the ASCI Grid. In: 9th IEEE Symposium on High Performance Distributed Computing, pp. 193–200. IEEE Press, New York (2000)
14. Frey, J., Foster, I., Livny, M.: 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)
15. 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)
16. 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)
17. 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, Boston (2003)
18. Cafaro, M., Mirto, M., Aloisio, G.: Preference-Based Matchmaking of Grid Resources with CP-Nets. J. Grid Computing 11(2), 211–237 (2013)
19. Rodero, I., Villegas, D., Bobroff, N., Liu, Y., Fong, L., Sadjadi, S.M.: Enabling Interoperability among Grid Meta-Schedulers. J. Grid Computing 11(2), 311–336 (2013)
20. Toporkov, V.: Application-Level and Job-Flow Scheduling: an Approach for Achieving Quality of Service in Distributed Computing. In: Malyshkin, V. (ed.) PaCT 2009. LNCS, vol. 5698, pp. 350–359. Springer, Heidelberg (2009)
21. Aida, K., Casanova, H.: Scheduling Mixed-parallel Applications with Advance Reservations. In: 17th IEEE Int. Symposium on HPDC, pp. 65–74. IEEE CS Press, New York (2008)
22. Ando, S., Aida, K.: Evaluation of Scheduling Algorithms for Advance Reservations. Information Processing Society of Japan SIG Notes HPC-113, 37–42 (2007)
23. Elmroth, E., Tordsson, J.: A Standards-based Grid Resource Brokering Service Supporting Advance Reservations, Coallocation and Cross-Grid Interoperability. J. of Concurrency and Computation 25(18), 2298–2335 (2009)

24. Toporkov, V., Tselishchev, A., Yemelyanov, D., Bobchenkov, A.: Composite Scheduling Strategies in Distributed Computing with Non-dedicated Resources. Procedia Computer Science 9, 176–185 (2012)
25. Moab Adaptive Computing Suite, http://www.adaptivecomputing.com/products/moab-adaptive-computing-suite.php
26. Lee, S.B., Schwartzman, Y., Hardy, J., Snavely, A.: Are User Runtime Estimates Inherently Inaccurate? In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 253–263. Springer, Heidelberg (2005)
27. Toporkov, V., Toporkova, A., Tselishchev, A., Yemelyanov, D.: Slot Selection Algorithms in Distributed Computing with Non-dedicated and Heterogeneous Resources. In: Malyshkin, V. (ed.) PaCT 2013. LNCS, vol. 7979, pp. 120–134. Springer, Heidelberg (2013)
28. Azzedin, F., Maheswaran, M., Arnason, N.: A Synchronous Co-allocation Mechanism for Grid Computing Systems. Cluster Computing 7, 39–49 (2004)
29. Castillo, C., Rouskas, G.N., Harfoush, K.: Resource Co-allocation for Large-scale Distributed Environments. In: 18th ACM International Symposium on High Performance Distributed Compuing, pp. 137–150. ACM, New York (2009)
30. Takefusa, A., Nakada, H., Kudoh, T., Tanaka, Y.: An Advance Reservation-Based Co-allocation Algorithm for Distributed Computers and Network Bandwidth on QoS-Guaranteed Grids. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2010. LNCS, vol. 6253, pp. 16–34. Springer, Heidelberg (2010)
31. Blanco, H., Guirado, F., Lérida, J.L., Albornoz, V.M.: MIP Model Scheduling for Multi-Clusters. In: Caragiannis, I., et al. (eds.) Euro-Par Workshops 2012. LNCS, vol. 7640, pp. 196–206. Springer, Heidelberg (2013)
32. Moise, D., Moise, I., Pop, F., Cristea, V.: Resource CoAllocation for Scheduling Tasks with Dependencies, in Grid. In: The Second International Workshop on High Performance in Grid Middleware (HiPerGRID 2008), Bucharest, Romania, pp. 41–48. IEEE Romania (2008)
33. Olteanu, A., Pop, F., Dobre, C., Cristea, V.: A Dynamic Rescheduling Algorithm for Resource Management in Large Scale Dependable Distributed Systems. Computers and Mathematics with Applications 63(9), 1409–1423 (2012)