Spectral Clustering Scheduling Techniques for Tasks with Strict QoS Requirements

Nikos Doulamis, Panagiotis Kokkinos, and Emmanouel Varvarigos

Department of Computer Engineering and Informatics, University of Patras and Research Academic Computer Technology Institute, Patras, Greece
ndoulam@cs.ntua.gr

Abstract. Efficient task scheduling is fundamental for the success of the Grids, since it directly affects the Quality of Service (QoS) offered to the users. Efficient scheduling policies should be evaluated based not only on performance metrics that are of interest to the infrastructure side, such as the Grid resources utilization efficiency, but also on user satisfaction metrics, such as the percentage of tasks served by the Grid without violating their QoS requirements. In this paper, we propose a scheduling algorithm for tasks with strict timing requirements, given in the form of a desired start and finish time. Our algorithm aims at minimizing the violations of the time constraints, while at the same time minimizing the number of processors used. The proposed scheduling method exploits concepts derived from spectral clustering, and groups together for assignment to a computing resource the tasks so to a) minimize the time overlapping of the tasks assigned to a given processor and b) maximize the degree of time overlapping among tasks assigned to different processors. Experimental results show that our proposed strategy outperforms greedy scheduling algorithms for different values of the task load submitted.

1 Introduction

Task scheduling is fundamental for the success of the Grids, especially with regards to its ability to support real-life commercial applications, by directly affecting a) the efficiency with which Grid resources are used and b) the Quality of Service (QoS) offered to the users. Evaluating a task scheduling algorithm should be based not only on resource utilization metrics, but also on user satisfaction factors, such as the percentage of tasks that are served by the Grid without violating their QoS requirements, e.g., their start and finish times [1]. Currently, several open source schedulers have been developed for clusters of servers such as Maui [2] and portable batch system (PBS) [3]. However, the primary objective of most existing approaches is to improve overall system performance (e.g., resource utilization), while the QoS experienced by Grid users is, at best, a secondary consideration [4], [5].

Without QoS guarantees (e.g., given in the form of task deadlines that should not be violated), users may be reluctant to pay for Grid services or contribute resources to the Grid, reducing its economic impact. On the other hand, designing a scheduling algorithm that satisfies only the end-to-end users' QoS, without taking into account Grid utilization efficiency, would result in a wasteful Grid architecture, that uses

E. Luque, T. Margalef, and D. Benítez (Eds.): Euro-Par 2008, LNCS 5168, pp. 478–488, 2008.

many more processors than necessary in order to satisfy users' QoS requirements. Equivalently, processor utilization would be relatively small, meaning that only a small percentage of the available resources would be exploited. Therefore, we need scheduling and resource allocation schemes that are able to simultaneously meet these two sometimes contradictory requirements: *optimize Grid utilization efficiency* while *simultaneously guaranteeing the tasks' strict QoS requirements* (e.g., deadlines).

Several computing toolkits and systems have been developed to meet the task QoS requirements in a Grid computing architecture. Globus is probably the most well known [6]. Additionally Condor-G is an enhanced version of Condor that uses the Globus toolkit to manage Grid jobs [7]. The Nimrod-G [8] is a Grid aware version of the Nimrod, which provides a simple declarative parametric modeling language for expressing a parametric experiment. A dynamic Grid resource allocation method is adopted in [9] using market economy notions (the G-commerce architecture). Finally, a new scheduling algorithm developed in the framework of the GrADS (Grid Application Development Software) tool has been proposed in [10]. A survey of state of the art methods for Grid scheduling is presented in [11].

In general, scheduling parallel and distributed applications is a known NP-complete problem. For this reason, several heuristic algorithms have been proposed for task scheduling. Some approaches use genetic algorithms to maximize the overall system performance [12], [13], while others use Directed Acyclic Graphs (DAG) for scheduling on heterogeneous or homogeneous computing environments [14], [15]. Performance evaluation results for these algorithms are presented in [16]. However, all the aforementioned approaches try to maximize overall system performance, (that is, Grid resource utilization), without respecting task deadlines (that is, user's QoS). Advance reservation of resources, which is the ability of the scheduler to guarantee the availability of resources at a particular time in the future, is one mechanism Grid providers may employ in order to offer specific QoS guarantees to the users [4]. However, these algorithms lack scalability, as they are unable to efficiently perform task scheduling in short time for large numbers of Grid resources. Using concepts from computational geometry, [1] solves the scalability problem for task scheduling under a user's satisfaction framework. The scalability problem is also addressed in [17]. Furthermore, fair scheduling algorithms and reservation schemes have been discussed in [5].

The main drawback of the above mentioned approaches is that scheduling is performed either in the direction of maximizing overall system performance (resource utilization efficiency) or minimizing the degradation of user's QoS requirements satisfaction. As mentioned before, a successful Grid scheduling algorithm should actually take into account both directions. This problem is addressed in this paper, by proposing a novel task scheduling algorithm that assigns tasks to processors so that a) *the time overlapping between tasks assigned to the same processor are minimized* (users QoS requirements are met to the degree possible), while simultaneously b) *maximizing overall Grid utilization efficiency*.

As we show in this paper, the two above mentioned objectives can be described by a matrix representation and then the proposed optimal scheduling strategy can be obtained by introducing the notions of generalized eigenvalues through the use of the Ky-Fan theorem [19]. The Ky-Fan theorem states that an optimal schedule that satisfies both aforementioned criteria can be derived as a solution of the largest

eigenvectors of the two matrices that represent the two conditions. Therefore, we have a scheduling algorithm of polynomial order with respect to the number of tasks, that simultaneously satisfies the users' QoS and the system's performance conditions.

The paper is organized as follows. Section 2 discusses the proposed scheduling algorithm for jointly optimizing resource utilization efficiency and tasks' QoS requirements. The solution of the joint optimization problem is given in Section 3. In Section 4, we discuss a lower bound on the number of processors required to achieve no task overlapping (no QoS violations) and propose objective criteria for evaluating scheduling efficiency. Experimental results and comparisons with other approaches are given in Section 5, while Section 6 concludes the paper.

2 Joint Optimization of Resource Performance and QoS Requirements

Let us denote by T_i, i=1,2,..,N, the tasks that request service in a Grid infrastructure consisting of M processors. Let us also denote by ST_i the desired *Start Time* for Task T_i and by FT_i its desired *Finish Time*. In this paper, we assume that the tasks are scheduled in a *non-preemptable, non-interruptible way*. Under this assumption, if a task has been assigned for execution on a processor and another task requests service on an overlapping time interval, then, the second task should either be assigned to another processor (which is not reserved at the requested time interval) or undergo violation of its QoS, i.e., its start or finish time or both of them.

We denote by σ_{ij} the non-overlapping measure between tasks T_i and T_j. Assuming that the task i Start and Finish Time, ST_i and FT_i, are hard constraints that should not be violated, a proper selection for the non-overlapping measure σ_{ij} is to take zero values when tasks T_i and T_j overlap in time and positive non-zero values when they do not overlap.

$$\sigma_{ij} = \begin{cases} \alpha, & \text{if } T_i, T_j \text{ are non-overlapping in time} \\ 0, & \text{if } T_i, T_j \text{ overlap in time} \end{cases} \tag{1}$$

where $\alpha > 0$.

Let us assume, without loss of generality that the *Start time* ST_i and *Finish time* FT_i for all tasks that are to be scheduled are within a *time horizon T,* which can be considered as the time interval within which one instance of the scheduling algorithm is executed. Let us denote by C_r the set of tasks assigned for execution on processor r.

As stated in Section 1, an efficient scheduling scheme for a commercially successful Grid should assign all the N pending tasks to the M processors so as to a) *minimize the tasks' QoS violations*, while simultaneously b) maximizing the *overall utilization* of the M processors, so that the Grid resources do not stay *idle* most of the time. The first requirement, in terms of the scheduling algorithm, means that the tasks assigned to a *given* processor should present *minimal overlapping*. The second requirement

indicates that the task overlapping among *different* processors should be *maximized*, that is, the utilization of all processors in Grid should be as high as possible. These two requirements can be written as

$$Q_r = \frac{\sum_{i \in C_r, j \in C_r} \sigma_{ij}}{\sum_{i \in C_r, j \in V} \sigma_{ij}}, \quad P_r = \frac{\sum_{i \in C_r, j \notin C_r} \sigma_{ij}}{\sum_{i \in C_r, j \in V} \sigma_{ij}}, \tag{2}$$

where $V = \{T_i\}_{i=1,\ldots,N}$ the set of tasks that request service in a Grid infrastructure.

The denominator of equations (2) is used for normalization purposes. Otherwise, optimizing would favor the trivial solution of one task per processor. Parameter Q_r expresses a measure of the overall QoS violation for the tasks assigned to the r^{th} processor. Instead, parameter P_r expresses the Grid utilization. Taking into account all the M processors of the Grid, we can define a measure Q for the total tasks' QoS violation and a measure P for the overall processor utilization as

$$Q = \sum_{r=1}^{M} Q_r \ , \ P = \sum_{r=1}^{M} P_r \ . \tag{3}$$

An efficient scheduler that tries to meet user QoS requirements should maximize Q and simultaneously minimize P. However, it is clear that

$$P + Q = M \ . \tag{4}$$

Equation (4) shows that the *maximization* of Q simultaneously yields a *minimization* of P and vice versa. Hence, in our problem, the two aforementioned optimization objectives require in fact the use of identical means and they can be met simultaneously. This is intuitively satisfying, since scheduling a set of tasks in a way that makes efficient use of resources is also expected to help meet the QoS requirements of the set of tasks that are scheduled. Therefore, it is enough to optimize (maximize or minimize) only one of the two criteria. In our case, we select to minimize variable P. Thus,

$$\hat{C}_r : \min P = \min \sum_{r=1}^{M} \frac{\sum_{i \in C_r, j \notin C_r} \sigma_{ij}}{\sum_{i \in C_r, j \in V} \sigma_{ij}}, \text{ for all } r=1,\ldots,M, \tag{5}$$

where $\hat{C}_r$ is the set of tasks assigned for execution on processor r.

3 The Proposed Task Scheduling Policy

3.1 Matrix Representation

Optimizing equation (5) is still a NP-complete problem, even for the special case of M=2 processors. To overcome this difficulty, we transform the problem of (5) into a matrix based representation. Let us denote by $\mathbf{\Sigma} = [\sigma_{ij}]$ a matrix which contains the values of the non-overlapping measures σ_{ij} for all NxN combinations of tasks T_i and

T_j. Let us now denote as $\mathbf{e}_r = [\cdots e_r^u \cdots]^T$ an Nx1 *indicator vector* whose u-th entry is given by

$$e_r^u = \begin{cases} 1, \text{ if Task } T_u \text{ is assigned to processor } r \\ 0, \text{ otherwise} \end{cases} \tag{6}$$

The indicator vector $\mathbf{e}_r$ points out which of the N tasks are executed on processor r. That is, indices of tasks executed on processor r are marked with one, while the remaining indices take zero values. Since the Grid infrastructure consists of M processors, M different indicator vectors $\mathbf{e}_r$, $r = 1,2,...,M$ are defined, each indicating the tasks assigned for execution on each processor. This way, we can express the left hand of (5) with respect to vectors $\mathbf{e}_r$, $r = 1,2,...,M$. However, we also need to express the right hand of (5) as a function of $\mathbf{e}_r$. For this reason, we denote by $\mathbf{L} = diag(\cdots l_i \cdots)$ the diagonal matrix, whose elements l_i express the cumulative non-overlapping degree of task T_i with the remaining tasks. That is,

$$l_i = \sum_j \sigma_{ij} . \tag{7}$$

Using matrices $\mathbf{L}$ and $\mathbf{\Sigma}$, we can express equation (5) as,

$$\hat{\mathbf{e}}_r, \forall r : \min P = \min \sum_{r=1}^{M} \frac{\mathbf{e}_r^T (\mathbf{L} - \mathbf{\Sigma}) \mathbf{e}_r}{\mathbf{e}_r^T \mathbf{L} \mathbf{e}_r} . \tag{8}$$

3.2 Optimization in the Continuous Domain

Let us form the indicator matrix $\mathbf{E} = [e_1 \cdots e_M]$, the columns of which correspond to the M processors in the Grid, while the rows to the N tasks, then the rows of $\mathbf{E}$ have only one unit entry and the remaining entries are zero. Optimization of (8) under the discrete representation of matrix $\mathbf{E}$ is still a NP hard problem. However, if we relax the indicator matrix $\mathbf{E}$ to take values in the continuous domain, we can solve the problem in polynomial time. We denote by $\mathbf{E}_R$ the *relaxed version of the indicator matrix* $\mathbf{E}$, i.e. a matrix whose rows take real values instead of binary values as is the case of the indicator matrix $\mathbf{E}$. Then, it can be proven that (8) can be rewritten as

$$P = M - trace(\mathbf{Y}^T \mathbf{L}^{-1/2} \mathbf{\Sigma} \mathbf{L}^{-1/2} \mathbf{Y}) , \tag{9a}$$

$$\text{subject to } \mathbf{Y}^T \mathbf{Y} = \mathbf{I} . \tag{9b}$$

$\mathbf{Y}$ is a matrix that is related to the matrix $\mathbf{E}_R$ through

$$\mathbf{L}^{-1/2} \mathbf{Y} = \mathbf{E}_R \mathbf{\Lambda} , \tag{10}$$

where $\mathbf{\Lambda}$ is any $M \times M$ matrix. In this paper, we select $\mathbf{\Lambda}$ to be equal to the identity matrix, $\mathbf{\Lambda}=\mathbf{I}$. Then, the relaxed indicator matrix $\mathbf{E}_R$ is given as

$$\mathbf{E}_R = \mathbf{L}^{-1/2} \mathbf{Y} . \tag{11}$$

Minimization of (11) is obtained through the Ky-Fan theorem [19]. The Ky-Fan theorem states that the maximum value of the $trace(\mathbf{Y}^T\mathbf{L}^{-1/2}\mathbf{\Sigma}\,\mathbf{L}^{-1/2}\mathbf{Y})$ subject to the constraint of $\mathbf{Y}^T\mathbf{Y}=\mathbf{I}$ is equal to the sum of the M ($M<N$) *largest eigenvalues of matrix* $\mathbf{L}^{-1/2}\mathbf{\Sigma}\,\mathbf{L}^{-1/2}$. This maximum value is provided for the matrix

$$\mathbf{Y}=\mathbf{U}\cdot\mathbf{R}, \tag{12}$$

where $\mathbf{U}$ is a $N\times M$ matrix whose columns are the *eigenvectors* corresponding to the M largest eigenvalues of matrix $\mathbf{L}^{-1/2}\mathbf{\Sigma}\,\mathbf{L}^{-1/2}$ and $\mathbf{R}$ is an *arbitrarily rotation matrix* (i.e., orthogonal with determinant of one). Again, a simple approach is to select matrix $\mathbf{R}$ as the identity matrix, $\mathbf{R}=\mathbf{I}$, that is $\mathbf{Y}=\mathbf{U}$. Therefore, we have that the optimal relaxed indicator matrix $\hat{\mathbf{E}}_R$ in the continuous domain is given as

$$\hat{\mathbf{E}}_R=\mathbf{L}^{-1/2}\mathbf{U}. \tag{13}$$

3.3 Discrete Approximation

The optimal matrix $\hat{\mathbf{E}}_R$ given by equation (13) does not have the form of the indicator matrix $\mathbf{E}$ since the values of $\hat{\mathbf{E}}_R$ are continuous, while $\mathbf{E}$'s entries are binary. Recall that a unit entry indicates the processor a task is assigned to for execution, under the non-interruptible, non-preemptable assumption. Consequently, the problem is how to round the continuous values of $\hat{\mathbf{E}}_R$ in a discrete form that approximates matrix $\mathbf{E}$.

One simple approach, regarding the rounding process, is to set the maximum value of each row of matrix $\hat{\mathbf{E}}_R$ to be equal to 1 and let the remaining values be equal to 0. However, such an approach yields unsatisfactory performance when there is no dominant maximum value for each row of $\hat{\mathbf{E}}_R$. Furthermore, it handles the rounding process as N (that is the number of tasks) independent problems. An alternative approach, which is adopted in this paper, is to treat the N rows of matrix $\hat{\mathbf{E}}_R$ as M-dimensional feature vectors. Each one of these feature vectors indicates the association degree of each task and the respective M processor of the Grid. Then, we apply the k-means to form the indicator matrix $\mathbf{E}$.

4 Lower Bound - Scheduling Efficiency

An important aspect, which determines the scheduling efficiency is the task granularity g, and the task arrival rate λ defined as

$$\lambda=\frac{N}{T},\ g=\frac{D}{T}, \tag{14}$$

where N is the number of tasks requesting service over the corresponding time interval T and D the average task delay.

Given a granularity g and a rate λ, the *lower bound* of Grid resources required for achieving no task overlapping is given by the following equation

$$B = \frac{ND}{T} = N \cdot g \le M_{opt}, \tag{15}$$

where M_{opt} refers to minimum number of processors required for achieving no task overlapping by an optimal (exhaustive search) scheduling algorithm. The lower bound of (15) is achieved in the extreme case that the tasks request execution intervals of a constant duration D that appear one right after the other, completely filling the gaps within the time horizon T. Thus, this lower bound is usually smaller than the M_{opt}.

Given the lower bound B on the number of processors required for no overlapping, the scheduling efficiency is defined as

$$e(A) = \frac{B}{M(A)} \text{ or } \varepsilon(A) = \frac{\lceil B \rceil}{M(A)}, \tag{16}$$

where A refers to the algorithm used to approximate the exhaustive search policy and M(A) is the number of processors required for achieving no task overlapping when algorithm A is used. $e(A)$ is the scheduling efficiency, while $\varepsilon(A)$ is the rounded efficiency for algorithm A. The $\lceil \cdot \rceil$ indicates the ceil operator.

5 Experimental Results

Two different algorithms were implemented in this paper and compared with respect to their scheduling efficiency. The first algorithm is the *proposed scheme,* presented in Section 2. The second scheme is a *greedy approach*, which, for each task, a locally optimum choice is selected. In particular, the algorithm assigns each task to a processor, so that no task overlapping is encountered, by taking into account the current local load of the processors.

Our proposed algorithm is recursively applied assuming different number of processors in Grid. Then, we select the minimum number of processors that provide *no task overlapping* that is no violation of the tasks' QoS. This number M(*Proposed Algorithm*) is used for evaluating the scheduling efficiency. In the greedy algorithm, each time a newly considered task overlaps with the already assigned tasks, then a new resource is activated and this task is assigned to this resource. The number of resources that have been activated after scheduling all tasks, without overlaps, is denoted by M(*Greedy*). We assume that the tasks' *Start and Finish Times* ST_i and FT_i are uniformly distributed within the time horizon T and that the average tasks' duration is constant and equals D. Experiments where the task duration varies significantly from task to task, have also been performed, but are not included here due to space limitations.

Fig. 1(a) presents the efficiency e [see equation (16)] versus the task granularity g for different values of lower bound B. As is observed, the efficiency increases as the granularity decreases for low values of g. However, the ratio of improvement decreases, meaning that the efficiency converges as g increases. We also observe from

Fig. 1 that for values of granularity greater than $g \geq 0.2$ the efficiency also increases as g increases. This is due to singularity issue, since in this case the minimum number of processors required for achieving no task overlapping equals the number of tasks N. In Fig. 1(b), we compare the continuous and rounded efficiency e and ε for the lower bound B=1. As expected, the rounded efficiency is a discontinuous function and several peaks are encountered due to the ceiling operator $\lceil \cdot \rceil$ involved in (16). However, in general terms, the overall behavior resembles that of the continuous case.

In Fig. 2(a), we depict the effect of the number of tasks N (equivalently, of the task arrival rate λ, for a given time window T) on the efficiency ε for different values of the granularity g. As we observe, the rounded efficiency presents a periodically discontinuous behavior that depends on the granularity value. This periodicity is due to the ceiling operator involved in the rounded efficiency ε [see equation (16)]. Next, we examined the effect of the number of iterations of the k-means algorithm used for estimating the indicator matrix **E** –that is tasks' partitioning– from the relaxed matrix

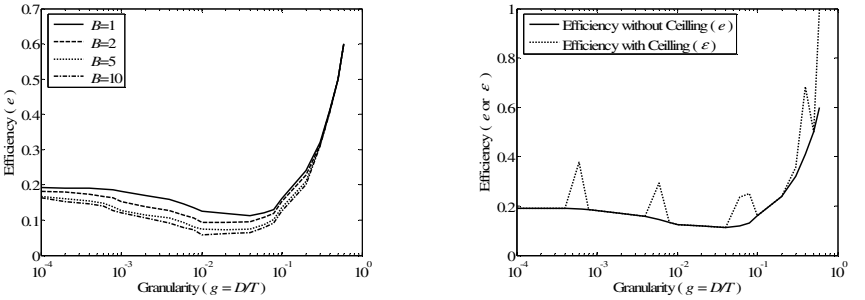

Fig. 1. (a) Efficiency (e) versus granularity (g) for different values of lower bound B in case that the proposed scheduling policy is used. (b) Comparison of the continuous and rounded efficiency (e and ε) for lower bound B=1.

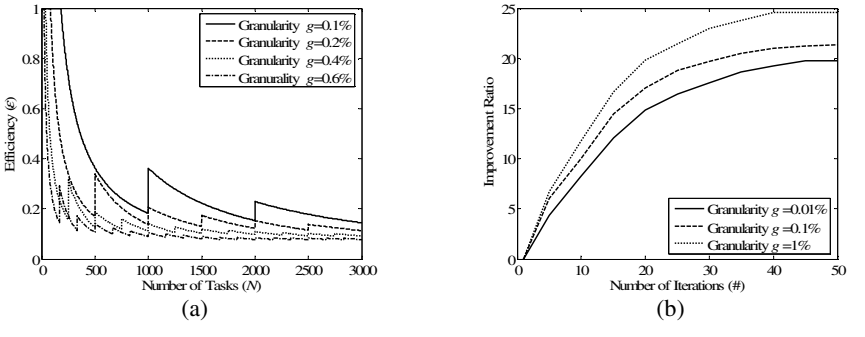

Fig. 2. (a) Efficiency ε versus the number of tasks for different values of granularity in case of B=1. (b) Improvement ratio of the efficiency versus the number of iterations of the proposed algorithm.

$\mathbf{E}_R$ computing by the Ky-Fan Theorem (see Section 3). In particular, Fig. 2(b) presents the improvement ratio versus the number of iterations for different granularity values, assuming B=1. We observe that as the number of iterations increases the scheduling efficiency increases for all granularity values. However, convergent is achieved for large number of iterations.

Fig. 3 presents the comparison results between the proposed algorithm for iterations of 1 and 50 and the greedy scheduling scheme. As we observe, the proposed algorithm exhibits better efficiency at any value of granularity. At low task load (low values of B) the improvement is more evident than for high values of B. Additionally, for low granularity values the improvement is smaller. This is because in this case, task durations are very small compared to the time window and thus both algorithms can schedule more effectively the tasks.

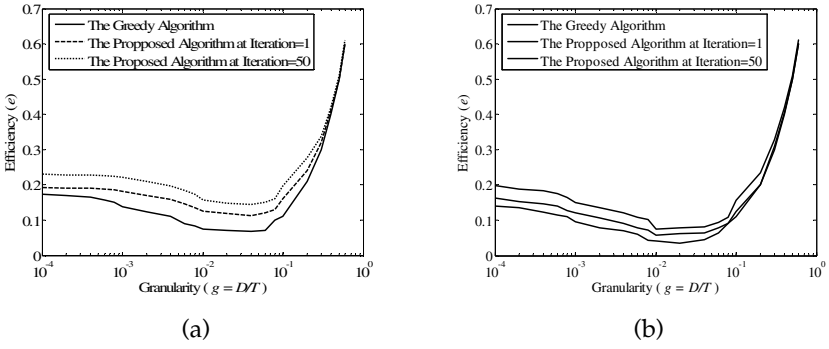

Fig. 3. Comparison of the proposed method for different iterations with the greedy algorithm. (a) B=1. (b) B=10.

6 Conclusions

We proposed an efficient scheduling strategy that maximizes Grid utilization efficiency, while resulting in a minimal degradation of the QoS offered to the submitted tasks. These two objectives are transformed into a matrix representation and then the scheduling problem is solved by introducing the notions of generalized eigenvalues through the use of the Ky-Fan theorem. Optimization using eigenvectors has the advantage that scheduling is performed in polynomial complexity.

Experimental results and comparisons with a greedy scheduling policy are presented to indicate the efficiency of the proposed scheme. In particular, we investigate the number of processors required for achieving no task overlapping (no QoS violations) under the two scheduling policies. We also define a lower bound on the minimum number of processors required and we estimate the scheduling efficiency of an algorithm as the ratio of the lower bound over the number of processors achieved by the algorithm. Comparison with the greedy scheduling policy demonstrates the efficiency of the proposed scheme for all granularities and different assumptions on the number and durations of the tasks. In addition, as the number of iterations of the proposed algorithm increases better scheduling efficiency is achieved. Algorithm

convergence is achieved even for a small number of iterations, e.g., 30. We find that task granularity affects more significantly the scheduling efficiency rather than the task arrival rate. Finally, efficiency is better at low values of granularity, however, convergence is noticed for very low granularities.

Acknowledgment

This work has been supported by the European Commission through the IP Phosphorus project.

References

[1] Castillo, C., Rouskas, G., Harfoush, K.: On the Design of Online Scheduling Algorithms for Advance Reservations and QoS in Grids. In: Int'l Parallel and Distributed Processing Symp., pp. 1–10 (2007)

[2] Jackson, D., Snell, Q., Clement, M.: Core Algorithms of the Maui Scheduler. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 2001. LNCS, vol. 2221, pp. 87–102. Springer, Heidelberg (2001)

[3] Bode, B., et al.: The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters. Usenix Conf. (2000)

[4] Al-Ali, R.J., et al.: Analysis and provision of QoS for distributed grid applications. Journal of Grid Computing 2(2), 163–182 (2004)

[5] Doulamis, N., Doulamis, A., Varvarigos, E., Varvarigou, T.: Fair Scheduling Algorithms in Grids. IEEE Trans, on PDS 18(11), 1630–1648 (2007)

[6] Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications 11(2), 115–128 (1997)

[7] Thain, D., Tannenbaum, T., Livny, M.: Condor and the Grid. In: Berman, F., Hey, A.J.G., Fox, G. (eds.) Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Chichester (2003)

[8] Abramson, D., Giddy, J., Kotler, L.: High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid. In: Int'l Parallel and Distributed Processing Symp. (2000)

[9] Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: G-commerce: Market Formulations Controlling Resource Allocation on the Computational Grid. In: Int'l Parallel and Distributed Processing Symp. (2001)

[10] K. Cooper et al., "New Grid Scheduling and Rescheduling Methods in the GrADS Project," Int'l Parallel and Distributed Processing Symp., pp. 199-207, 2004.

[11] Maheswaran, M., Krauter, K., Buyya, R.: A taxonomy and survey of grid resource management systems for distributed computing. Software: Practice and Experience 32(2), 135–164 (2002)

[12] Shu, W., et al.: A Grid Computing Task Scheduling Method Based on Target Genetic Algorithm. The Sixth World Congress on Intelligent Control and Automation 1, 3528–3532 (2006)

[13] Ye, G., Rao, R., Li, M.: A Multiobjective Resources Scheduling Approach Based on Genetic Algorithms in Grid Environment. In: Fifth International Conference on Grid and Cooperative Computing Workshops, pp. 504–509 (2006)

[14] Topcuoglu, H., Hariri, S., Wu, M.-Y.: Performance Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. Transactions on Parallel and Distributed Systems 2(13), 260–274 (2002)
[15] Mandal, A., et al.: Scheduling Strategies for Mapping Application Workflows onto the Grid. In: Symp. on High Performance Distributed Computing, pp. 125–134 (2005)
[16] Zhang, Y., Koelbel, C., Kennedy, K.: Relative Performance of Scheduling Algorithms in Grid Environments. In: Int'l Conf. on Cluster Computing and the Grid, pp. 521–528 (2007)
[17] Zhang, Y., et al.: Scalable Grid Application Scheduling via Decoupled Resource Selection and Scheduling. In: Int'l Conf. on Cluster Computing and the Grid, pp. 568–575 (2006)
[18] Varvarigos, E., Doulamis, N., Doulamis, A., Varvarigou, T.: Timed/Advance Reservation Schemes and Scheduling Algorithms for QoS Resource Management in Grids. In: Di Martino, B., Dongarra, J., Hoisie, A., Yang, L.T., Zima, H. (eds.) Engineering the Grid, pp. 355–378. American Scientific Publishers (2006)
[19] Nakic, I., Veselic, K.: Wielandt and Ky-Fan Theorem for Matrix Pairs. Linear Algebra and its Applications 369(17), 73–77 (2003)