Execution Management of Applications with Runtime Restrictions on Opportunistic Grids Environments

Marcio Rodrigo Melo Martins, Berto de Tácio Pereira Gomes, Jesseildo Figueredo Gonçalves
Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade
Av. dos Portugueses s/n, Campus Universitário do Bacanga, CEP. 65085-580, São Luís-MA-Brasil
marciorodrigomm, bertodetacio, jesseildo@gmail.com

Francisco José da Silva e Silva
Universidade Federal do Maranhão, Departamento de Informática
fssilva@deinf.ufma.br

***Abstract*—An opportunistic grid computing environment takes advantage of idle computing cycles of regular computers and workstations that can be spread across several administrative domains for running high performance applications. Opportunistic grids are usually constructed from personal computers that do not need to be dedicated for executing grid applications. The grid workload must coexist with local applications executions, submitted by the nodes regular users. Thus, its execution environment is typically dynamic, heterogeneous and unpredictable failures occur frequently. In addition, the resources of an opportunistic grid can be used at any time for the execution of local tasks, making it difficult to preview the conclusion of the tasks running on the grid nodes. These characteristics hinder the successful execution of applications for which there are time restrictions related to its completion. This paper presents a management mechanism specifically designed for opportunistic grid computing environments for handling the execution of applications with time deadlines set by users during their submission to the system. The proposed mechanism is based on a dynamic scheduling and rescheduling approach and was evaluated using a simulated model considering various typical scenarios of opportunistic grids. The results demonstrated the benefits of the proposed approach in comparison to traditional scheduling approaches applied in opportunistic grids.**

***Keywords*-opportunistic grids; application scheduling; soft deadline; simulation.**

I. Introduction

A computer grid is a computing system that coordinates distributed resources using standard protocols and interfaces to enable integration and sharing of computational resources, such as computing power, software, peripherals and data on corporate networks and between institutions. The computer grid technology currently receives great attention from both academia and industry since it have established itself as an attractive alternative for executing a wide range of applications that require massive computational power or process large volumes of data in various areas, such as computational biology, weather and market simulations.

Currently, corporations and universities typically have hundreds or thousands of desktop machines, which are used by workers as their personal workstations or by students in instructional and research laboratories. When analyzing the usage of each of these machines one concludes that they sit idle for a significant amount of time. Even when the computer is in use, it normally has a large portion of idle resources. Opportunistic grid middleware enables the use of the existing computing infrastructure available in laboratories and offices in universities, research institutes, and companies to execute computationally intensive applications. They are usually constructed from personal computers that do not need to be dedicated for executing grid applications. The grid workload must coexist with local applications executions, submitted by the nodes regular users.

The process of application scheduling in a grid system consists in assigning the applications tasks to available resources in accordance to specific goals, such as to minimize the applications response time and/or maximize the use of computational resources. However, the construction of a scheduling strategy focused on opportunistic grid environments is a challenging task due to several features present in these computing environments, such as: (a) instability, that arises from the fact that nodes are not dedicated and applications do not run in a controlled environment; (b) high heterogeneity of computing nodes and network links, usually comprising resources spread across different administrative domains; (c) the middleware goal of not interfering with the regular use of resources, which requires migration and rescheduling of applications when resources become unavailable. In addition, the resources of an opportunistic grid can be used at any time for the execution of local tasks, making it difficult to preview the conclusion of the tasks running on the grid nodes. In particular, these characteristics hinder the successful execution of applications for which there are time restrictions related to its completion.

This paper presents a management mechanism specifically designed for opportunistic grid computing environments for handling the execution of applications with time deadlines set by users during their submission to the system. The proposed mechanism is based on a dynamic scheduling and rescheduling approach and was evaluated using a simulated model considering various typical scenarios of opportunistic grids. The results demonstrated the benefits of the proposed approach in comparison to traditional scheduling approaches applied in those environments. The text is structured as follows: Section II describes the related work that influenced the development of this work. Section III presents our proposed mechanism, its concepts, components, and implementation. Section IV describes the results of several experiments that were conducted for evaluating the proposed approach. Finally, in Section V, we describe the conclusions derived from this work and presents future perspectives arising from this initial effort.

II. Related Work

Currently, there are several works related to the development of scheduling algorithms for computer grids aiming different performance goals, such as minimizing the completion time of application or to maximize the use of grid resources, without taking into account the users requirements [1] [2] [3] [4]. Other studies have also emerged with the purpose of meeting user defined Quality of Service (QoS) metrics [5] [6] [7]. These works emphasizes the difficulties of providing support for execution time constraints of submitted applications (*deadlines*). They present new approaches to scheduling, or the adequacy of existing approaches, in order to consider QoS requirements and metrics for the evaluation of scheduling strategies and/or fault tolerance mechanisms in computer grids.

Buyya et al. [5] present a scheduling algorithm for Parameter Sweep applications on global grids. The algorithm goal is to map applications to resources considering, whenever possible, the best cost-benefit ratio between the applications execution time and the cost to perform these computations. Thus, if multiple machines offer the same completion time for a given application, it should be scheduled to the one that offers the least cost (QoS requirement). The algorithm is called *DBC cost-time optimization*. The grid user optionally provides the following parameters when submitting an application for execution: the time she/he is willing to wait until the completion of the application execution (deadline), and the price she/he is willing to pay for the realization of the computation (budget). The algorithm follows an on-line approach. It runs until there are applications to be scheduled able to be executed according to the provided deadline and cost, as defined by its users. The algorithm was evaluated using the GridSim simulator.

Yu and Buyya [6] present a genetic algorithm for scheduling workflow applications. The purpose of this algorithm is to run applications within a certain period with the lowest possible cost (QoS). In order to evaluate the proposed approach, the authors implemented the algorithm and compared it with a set of non-genetic heuristics for two different types of workflow applications (balanced and unbalanced) using the GridSim simulator.

Kyong Kim et al. [7] address an issue that is becoming increasingly important in cluster computing environments: the need to reduce energy consumption for running applications, which gave rise to the term power-aware scheduling. The authors present two scheduling algorithms for Bag-of-Tasks applications, a shared space policy resource allocation, and a time-shared based one. The shared space policy allows the execution of only one task at a time in a given processing unit, while the time-shared policy allows multiple tasks to share the same processing unit, running in alternating time slices. The algorithms goal is to meet time constraints set for the execution of applications, minimizing the energy consumption in cluster systems. An adopted assumption is that the grid nodes have support to Dynamic Voltage Scaling (DVS), a power management technique that allows the dynamic adjustment of the voltage used by a given hardware component. According to the authors, the proposed algorithms can reduce energy consumption by adjusting the levels of voltage used by the grid nodes.

What distinguishes our work from the above described approaches is the specific attention to opportunistic grid environments. In this way, we consider the possible existence of local workload on the grid nodes, on which we have no control. This prevents the accurate prediction of the tasks execution time, forcing the use of a mechanism to monitor the tasks execution progress and the possible need for rescheduling and migrating them. Moreover, we also take into consideration that desktop grids are subject to various types of failures and may exhibit physical problems (from both, nodes and network links), logical errors (in the application or the communication protocols, for example) and suffer invasions of malicious software. Another common failure type is related to the resources dynamism, which are usually not dedicated and can suddenly become unavailable, even when performing computations on behalf of the grid. In addition, applications are usually composed of long-running tasks, which can take hours or even days to run, exacerbating the possibility of failure [8]. In order to circumvent that, we integrated to our solution an application execution autonomic fault tolerance mechanism developed for opportunistic grids environments [9]. This mechanism is based on the use of an autonomic checkpoint approach, which dynamically adjusts the time interval between successive checkpoints of a running task based on the node Mean Time Between Failures (MTBF) history, thus contributing to increase the success rate of the applications execution and, at the same time, reducing the cost involved in the checkpoint process.

Finally, we argue that the dynamism and instability of the resources that comprise opportunistic grids environments make them inappropriate for running applications with severe running time restrictions (hard deadlines) and, therefore, our approach is geared to meet soft applications execution time constraints (soft deadlines).

III. The Proposed Mechanism

The development of the approach presented in this paper was done in the context of the project InteGrade [1] [10], a multi-institutional effort to develop a middleware for opportunistic grids. In this middleware, the user informs constraints and preferences to be taken in consideration when submitting an application for execution. Constraints define minimum requirements for the selection of nodes, such as a specific hardware and software platform. Preferences are used to define an order in the selection of resources for the application execution, such as running it on machines with preferably more than 1 GB of main memory available. In this work, we also consider the users preferences with respect to time constraints for the application execution. The user is then able to specify whether the application being submitted belongs to one of the classes *nice* or *soft-deadline*. In the latter case, the user must provide a deadline for executing the desired application. For soft-deadline applications, the goal is the completion of applications within the informed extend of time. For the nice applications, the goal is just the application successful completion.

The application execution management mechanism for opportunistic grids proposed in this paper consists of the following components:

1) A prediction mechanism for the time execution of applications on the grid nodes;
2) An on-line scheduling heuristic which maps the application tasks to nodes that could potentially meet the application deadline as specified by it's user;
3) A mechanism that tracks the tasks execution progress on each grid node, checking whether or not is necessary to move them to other nodes in order to meet the specified deadline;
4) An adaptive application fault tolerance mechanism based on checkpoint, whose goal is to ensure the successful execution of applications, even in the event of failures.

Two of these four components were based on previous work, they are: the execution time prediction mechanism and the adaptive fault tolerance mechanism. The adopted execution time prediction mechanism is based on [11]. In this paper, the author presents two approaches for predicting the execution time of applications. In the first approach, the calculation of the estimated runtime is based on records of the application previous runs. The second approach, used in our work, is based on knowledge concerning the application execution model. The application code is analyzed, estimating the execution time of each task according to the capacity of the grid resources. Sun and Wu [12] developed a mathematical model to predict performance for a non-dedicated distributed environment. Their work was based on Gong et al. [13]. The prediction model took into consideration that the workstations comprising the distributed environment may be privately owned, which is exactly the case of opportunistic grid computing. In this way, parallel tasks submitted to the grid compete for execution with local sequential jobs submitted by the machines owners. The model also considers systems with heterogeneous machine utilization and heterogeneous service distribution and separates the influence of machine utilization, sequential job service rate, and parallel task allocation on the parallel completion time. A tacit assumption of the proposed model is that the parallel task can be partitioned freely into small pieces and it does not considered the effects of synchronization, communication, process migration, or granularity of parallelism. The adaptive application fault tolerance mechanism of is based on [9]. In this work, the authors present an autonomic strategy for application execution fault tolerance for opportunistic grids. The mechanism is based on two levels of adaptations: (a) parametric reconfiguration of fault tolerance strategies (checkpointing and replication); and (b) structural changes of the fault tolerance mechanism, by fully replacing the used technique. In our work, we explored the autonomic reconfiguration of the checkpoint technique, which dynamically adjusts the time interval between successive checkpoints of a running task based on the node MTBF history.

A component running on each grid node called Local Resource Manager (LRM) is responsible for tracking the execution progress of a task scheduled for its grid resource. This component receives from the grid scheduler the task execution request along with its respective constraints and properties, including its class (nice or soft-deadline) and the given deadline for its completion. Running tasks should regularly report the LRM about their execution progress using a well-defined API. On each received notification, the LRM estimates if the completion of the application should occur within the time limit and, if not, notifies the application through a callback method, forcing its suspension and the save of its state in a stable storage (checkpoint). The task execution request is then forwarded to the scheduler, which will map it to another grid node that could meet the requested deadline, if available.

The scheduling heuristic proposed in this paper follows an on-line approach, allowing the mapping of more than one task to a given grid node. The mapping of soft-deadline tasks is performed taking into account the nodes Mean Time Between Failure (MTBF) and their processing capacity. Soft-deadline tasks are scheduled to nodes that have been identified as stable (high MTBF) and whose capacity allows to conclude a task within the specified time constraint, as informed by the user during the application submission. Due to the cost imposed by the use of a checkpoint approach

[1]Homepage: http://www.integrade.org.br

and the possible local workload, a node is considered able for accomplishing the task if its capacity allows the task execution within the estimated deadline increased with a margin of 10%. If the local workload exceeds this initial estimative, the task execution monitoring mechanism (as described in the previous paragraph) is responsible for identifying the node inability in meeting the deadline and for re-submitting the task for a new scheduling.

The proposed approach includes a mechanism for advanced resource reservation that works as follows: when performing a task mapping, if there are no nodes available that meet the above criteria, the algorithm seeks for busy nodes that could satisfy the provided deadline and that the already running tasks meet the following conditions: (a) if the task class is nice, it will be suspended to make way for the soft-deadline task, reserving the resource for the later execution of the suspended nice task, that will be performed based on the last saved checkpoint; (b) if the task class is soft-deadline, the algorithm checks if the predicted remaining time for its execution increased with the time necessary to execute the task being scheduled is sufficient for accomplishing the provided deadline of the latter. If this condition holds, the resource is reserved for the execution of the task at hand, that will be carry out after the execution of the task already in place. Finally, if there are no resources that meet the above criteria, the user will have the application submission refused.

Nice tasks are scheduled for nodes considered less stable (with lower MTBF). This is done by ordering the available nodes according to a decreasing MTBF order and selecting the last one. If there are no nodes available, the algorithm searches for nodes running nice tasks and reserves the first node found for the later execution of the task being scheduled. Finally, if all the grid nodes are running soft-deadline tasks, the algorithm randomly chooses one, reserving it for executing the task after the already running computation has finish its execution.

IV. Evaluation

We evaluated our proposal through simulations that took into consideration various typical opportunistic grids scenarios, using as the evaluation metric the amount of soft-deadline applications executed within the user informed execution time restrictions. Since this work was done in the context of project InteGrade, we compared the results obtained with our approach with the regular InteGrade scheduling algorithm, that works as follows. The InteGrade scheduling algorithm [14] follows an on-line approach. It uses a filter to select resources based on constraints and preferences provided by users during the process of submitting applications. Constraints define minimum requirements for the selection of machines, such as hardware and software platforms, resource requirements such as minimum memory requirements. Preferences define the order used for choosing the resources, like rather executing on a faster CPU than on a slower one. The tasks that make up an application are then mapped to the nodes according to the ordered list of resources. If requirements and preferences are not specified, the algorithm maps the tasks to random chosen grid resources. The algorithm can map more than one task per node.

For performing the simulations, we used the AGST (Autonomic Grid Simulation Tool) [2] [15], an object-oriented discrete event simulator. The simulator provides, among others, tools for modeling grid resources and their network interconnections, grid applications and their submissions, the occurrence of resource faults, resources local workload, the use of workload and fault traces following the SWF (Standard Workload Format) [3] and FTA (Failure Trace Archive) [4] standards, a database model for storing relevant simulation generated data. Nevertheless, AGST major contribution is the definition and implementation of a simulation model based on the MAPE-K [16] autonomic management cycle, that can be used to simulate the monitoring, analysis and planning, control and execution functions, allowing the simulation of an autonomic computing grid. This is an important feature, since in our work we adopted an autonomic fault-tolerance mechanism.

A. Performed Experiments

The simulated grid environment comprises 100 machines within the same administrative domain, interconnected through a 100 Mbps network. In this environment, the average processing power is equivalent to a Pentium IV machine with 2.4 GHz (2,770 MIPS, considering the TSCP benchmark [5]), since we considered this as a good representative for personal computers. In order to take into consideration the environment resource heterogeneity, grid nodes were synthetically generated through an uniform distribution. We performed several simulations considering two heterogeneity factors: $U(1.385; 4.155)$ MIPS and $U(791; 4.746)$ MIPS. Using the first factor, the the processing power of the fastest machine is about 3 times greater then the processing power of the slowest one. In the later case, the difference is approximately 6 times.

The simulations took into consideration the existence of local workload in the grid resources. The defined workload model was based on Conde [17]. In this work, data regarding the use of resources (CPU and memory) from several machines belonging to laboratories of the Computer Science Department at the University of São Paulo were collected and stored in a trace file. By reading and analyzing these files, it was possible to simulate the workloads for both weekdays and for weekends. We developed an application

[2]http://www.lsd.ufma.br/∼agst
[3]http://www.cs.huji.ac.il/labs/parallel/workload/swf.html
[4]http://fta.inria.fr/
[5]http://home.comcast.net/∼tckerrigan/bench.html

that generates workload vectors, each one having 24 positions representing the 24 hours of a day. The vectors were passed as parameters to the AGST, that simulates the nodes workload.

For the grid workload, we synthetically generated a total of 770 regular grid applications for each experiment, varying their size (in millions of instructions) through an uniform distribution $U(39.888\times 10^3; 159.552\times 10^3)$ MI. Considering the average processing power of the grid machines (2,770 MIPS), each application execution would take 4 to 16 hours. We performed simulations taking into consideration several applications arrival rate per second: 0.002 (0.12 applications per minute); 0.004 (0.24 applications per minute) and 0.006 (0.36 applications per minute). During the simulations, we also varied the amount of soft-deadline applications comprising the simulated application set, using the following parameters: 25%, 50%, 75% and 100%. Table I summarizes the parameters used in the performed simulations.

Table I
SUMMARY OF THE SIMULATION PARAMETERS

machines (nodes)	100
heterogeneity factor	factor 3 = $U(1.385; 4.155)$ and factor 6 = $U(791; 4.746)$
regular grid applications	770 (tasks)
applications arrival rate	0.12; 0.24 and 0.36 (app/min)
percentage of soft applications	25; 50; 75 and 100 (%)
applications size in MI	4 to 16 hours = $U(39.888 \times 10^3; 159.552 \times 10^3)$

We performed a total of 48 simulations, by combining the two simulated scheduling strategies (our approach and the regular InteGrade one), the three applications arrival rate, the four amount of soft-deadline applications and the two environment resource heterogeneity factors. For each simulation, we generated 20 sets of 770 applications, leading to a total of 960 experiments.

Figure 1 presents the results obtained with the two scheduling strategies when using a 0.002 (0.12 applications per minute) arrival rate and an environment resource heterogeneity factor of the fastest machine being about 3 times greater than the processing power of the slowest one. As one can see, our proposed approach meets the runtime execution constraint of almost 100% of the soft-deadline submitted applications, even when 100% of the submitted application is of that class. This is approximately 25% better than the regular InteGrade algorithm, which accomplished almost 80% of the applications deadlines. In this case, both approaches presented a good performance, since the grid workload is relatively low.

Figure 2 shows the result in a scenario where the grid workload is higher, using an application arrival rate of 0.006 (0.36 applications per minute), maintaining the environment resource heterogeneity factor of the fastest machine being

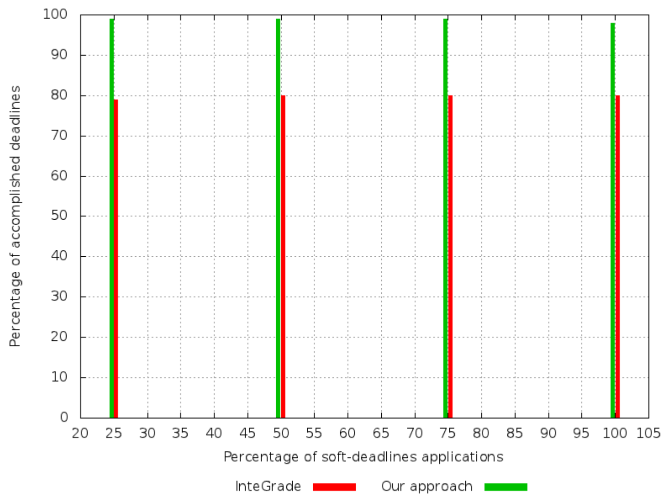

Figure 1. 0.12 applications per minute, heterogeneity factor 3

about 3 times greater than the processing power of the slowest one. As can be seen, in this case the regular InteGrade approach presents a worse result than was seen in the previous simulation, leading only to a 20% of deadlines accomplished when 50% of the submitted applications were soft-deadline. Our approach presented a much better result, meeting almost 50% of the informed deadlines, which represents a gain of 150%.

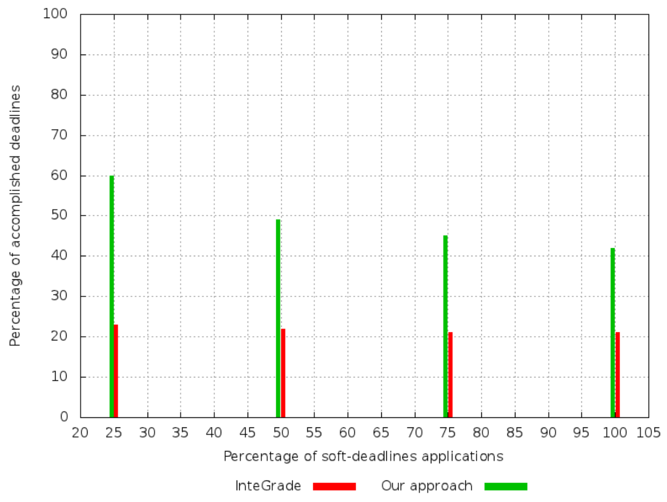

Figure 2. 0.36 applications per minute, heterogeneity factor 3

In another simulated scenario, we maintained the application arrival rate of 0.006 (0.36 applications per minute), altering the environment resource heterogeneity factor by having the fastest machine processing power being about 6 times greater than the slowest one. In this case, for a total of 50% of soft-deadline applications on each application set, our approach accomplished 60% of the requested deadlines, while the regular InteGrade achieved only 25%. This indicates that our approach can take advantage of the greater resource environment heterogeneity for achieving a higher performance.

Space constraints prevent us from showing the results of all the 48 performed simulations, but from the previous described ones, we can conclude that our approach performs much better than the regular InteGrade, considering the objective of running applications with runtime restrictions on opportunistic grid environments.

V. Conclusions and Future Work

Opportunistic grids execution environments are typically dynamic, heterogeneous, unpredictable and highly prone of failures. In addition, the resources of an opportunistic grid can be used at any time for the execution of local tasks, making it difficult to preview the conclusion of the tasks running on the grid nodes. These characteristics hinder the successful execution of applications for which there are time restrictions related to its completion.

This paper presented a new approach to application execution management considering user defined run time restrictions (soft deadlines) developed specifically for opportunistic grid environments. The proposed approach consists of a mechanism for predicting the execution time of applications in grid nodes, an on-line scheduling heuristic, a mechanism that tracks the execution progress of tasks running on the grid nodes and an adaptive fault tolerance mechanism based on the use of checkpoint. When properly combined, these mechanisms comprise a management model that allows applications to run within their time restrictions whenever possible, even in an environment as dynamic as the one typically provided by opportunistic grids. The proposed approach has been properly evaluated in a simulated environment, with experimental results demonstrating significant improvements when compared to traditional scheduling approaches used on computer grids.

In the future, we intend to explore and evaluate our proposal considering other classes of applications commonly used on computer grids, such as Bag-of-Tasks, Workflows and Parameter Sweep.

Acknowledgments

This work is supported by the Brazilian Federal Research Agency, CNPq, grant No.478853/2009-2, as well as by the state of Maranhão Research Agency (FAPEMA).

References

[1] D. da Silva, W. Cirne, and F. Brasileiro, "Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids," in *Euro-Par 2003 Parallel Processing*, ser. Lecture Notes in Computer Science, H. Kosch, L. Böszörményi, and H. Hellwagner, Eds. Springer Berlin / Heidelberg, 2003, vol. 2790, pp. 169–180. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45209-6_26

[2] H. Casanova, D. Zagorodnov, F. Berman, and A. Legrand, "Heuristics for scheduling parameter sweep applications in grid environments," in *Proceedings of the 9th Heterogeneous Computing Workshop*, ser. HCW '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 349–364. [Online]. Available: http://portal.acm.org/citation.cfm?id=795691.797922

[3] L. de Assis, "Uma heurística de escalonamento adaptativa à disponibilidade da informação para aplicações bag-of-tasks data-intensive em grids computacionais," Master's thesis, Universidade Federal de Campina Grande, Campina Grande, Paraíba, Brasil, Setembro 2009. [Online]. Available: http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2009/Dissertacao_LeonardodeAssis.pdf

[4] E. Santos-Neto, W. Cirne, F. Brasileiro, and A. Lima, "Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Springer Berlin / Heidelberg, 2005, vol. 3277, pp. 54–103. [Online]. Available: http://dx.doi.org/10.1007/11407522_12

[5] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal, "Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm," *Software: Practice and Experience*, vol. 35, no. 5, pp. 491–512, April 2005. [Online]. Available: http://dx.doi.org/10.1002/spe.646

[6] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Sci. Program.*, vol. 14, no. 3,4, pp. 217–230, December 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1376960.1376967

[7] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters," in *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, ser. CCGRID '07. Washington, DC, USA: IEEE Computer Society, May 2007, pp. 541–548. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2007.85

[8] S. S. Sathya and K. S. Babu, "Survey of fault tolerant techniques for grid," *Computer Science Review*, vol. 4, no. 2, pp. 101–120, May 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574013710000134

[9] A. E. B. Viana, B. de Tácio P. Gomes, J. F. Gonçalves, L. R. Coutinho, and F. J. da Silva e Silva, "Design and evaluation of autonomic fault tolerance strategies using the agst autonomic grid simulator," in *LatinAmerican Conference on High Performance and Distributed Computing (CLCAR '11)*, Colina, Mexico, Sep 2011.

[10] F. J. da Silva e Silva, F. Kon, A. Goldman, M. Finger, R. Y. de Camargo, F. C. Filho, and F. M. Costa, "Application execution management on the integrade opportunistic grid middleware," *Journal of Parallel and Distributed Computing*, vol. 70, no. 5, pp. 573 – 583, May 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731510000171

[11] Y. Liu, “Survey on grid scheduling,” Department of Computer Science, University of Iowa, for PhD Qualifying Exam, April 2004.

[12] X.-H. Sun and M. Wu, “Grid harvest service: A system for long-term, application-level task scheduling,” *Parallel and Distributed Processing Symposium, International*, vol. 0, p. 25a, april 2003. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/IPDPS.2003.1213102

[13] L. Gong, X.-H. Sun, and E. F. Watson, “Performance modeling and prediction of nondedicated network computing,” *IEEE Transactions on Computers*, vol. 51, no. 9, pp. 1041–1055, September 2002. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TC.2002.1032624

[14] A. Goldchleger, F. Kon, A. Goldman, M. Finger, and G. C. Bezerra, “Integrade: Object-oriented grid middleware leveraging idle computing power of desktop machines,” *Concurrency and Computation: Practice and Experience*, vol. 16, no. 5, pp. 449–459, March 2004. [Online]. Available: http://dx.doi.org/10.1002/cpe.824

[15] B. de Tácio Pereira Gomes and F. J. da Silva e Silva, “Agst - autonomic grid simulation tool - a simulator of autonomic functions based on the mape-k model.” in *SIMULTECH*. SciTePress, 2011, pp. 354–359. [Online]. Available: http://dblp.uni-trier.de/db/conf/simultech/simultech2011.html#GomesS11

[16] M. C. Huebscher and J. A. McCann, “A survey of autonomic computing-degrees, models, and applications,” *ACM Comput. Surv.*, vol. 40, no. 3, pp. 7:1–7:28, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1380584.1380585

[17] D. Conde, “Análise de padrões de uso em grades computacionais,” Master’s thesis, Departamento de Ciência da Computação - Universidade de São Paulo, Brasil, SP, Janeiro 2008, retrieved: 28/11/2011. [Online]. Available: http://www.integrade.org.br/files/danconde_dissertacao.pdf