Towards a Resource Reservation Approach for an Opportunistic Computing Environment

Eliza Gomes, M.A.R. Dantas

Research Laboratory of Distributed Systems (LaPeSD)
Department of Informatics and Statistic (INE)
Federal University of Santa Catarina (UFSC)
88040-900 - Florianopolis,SC - Brazil

E-mail: eliza.gomes@posgrad.ufsc.br, mario.dantas@ufsc.br

Abstract.

Advanced reservation has been used in grid environments to provide quality of service (QoS) and to guarantee resources available at the execution time. However, in grid subtypes, such as opportunistic grid computing, it is a challenge provides QoS and guarantee of availability resources. In this article, we propose a new advanced reservation approach which offers to a user the possibility to select resources in advance for a future utilization. Therefore, the main goal of this proposal is to offer a best effort feature to a user from an opportunistic configuration. In these types of environments, it is not possible to provide QoS, because, usually, there are no guarantees of resources availability and, consequently, the execution of users applications. In addition, this research work provides a way to organize executions, what it can improve the scheduling and system operations. Experimental results, carried out through a case study, shown the efficiency and relevance of our proposal.

1. Introduction

In recent years we have seen increasing use of supercomputers to execute applications which require high performance computing [1]. On the other hand, personal computers have increasingly computational power to, often, execute simple applications. In the face of this scenario, to cluster these devices, as opportunistic configurations, has become a widely paradigm used (and required) by industrial and academic communities. Opportunistic grid configurations are a sub-type of environment, which utilize idle computational resources from personal machines, companies and laboratories, to form, dynamically, a computational grid [2]. Examples of middleware for opportunistic grid are the projects: SETI@home [3], BOINC [4] and InteGrade [5].

Opportunistic systems can be characterized as meta-schedulers, because they receive requests from users and scheduling their applications to different local schedulers. However, these execution requests of tasks are, usually, randomly and disorderly. In other words, requests may not exist for long periods and then a system receives several requests. This scenario illustrates that in any system can become hard to process users' requests.

This article presents a proposal of an implementation of an advanced reservation of resources to opportunistic grid environments. The advanced reservation concept can be understood as the ability to user to reserves resource in advanced to be used at specific time in the future

[6]. Resources commonly reserved are: CPU and amount of memory, disk space and links [7]. The InteGrade project was utilized as a middleware to opportunistic grid, where we added our contribution.

This proposal comprises an implementation of an additional module in the InteGrade architecture which enables users to request resources in advance. The main goal is to offer to user a best effort, once is not possible to guarantee QoS, due to dynamism of the environment. In addition, our proposal aims to improve the applications scheduling, in looking for an advanced reservation slot, somehow, organizes executions, once the applications will be organized in queue, waiting for the beginning of the execution.

The proposal was evaluated through a case study which simulates the module behavior. Experimental results show the efficiency of the mechanism and a best effort offered to user.

This article is organized as follows: section 2 provides a review related to the concepts of co-scheduling, co-allocation and co-reservation to build a meta-scheduler while section 3 presents related works. Section 4 describes the proposed architecture. In section 5 the implementation is simulated through case study. Section 6 concludes the article and suggests some future works.

2. Co-Scheduling, Co-Reservation and Co-Allocation

Meta-schedulers are characterized by receiving requests from users and schedule applications to local cluster scheduler. They also have the function of orchestrate multi-cluster configurations [8]. The use of advance reservation in multi-schedulers requires efficient algorithms to co-scheduling, co-reservation and co-allocation.

2.1. Co-Scheduling

Scheduling determines where and when jobs are executed and how many resources will be allocated. Co-scheduling can be understood as the ability to schedule jobs to execute at the same time at different processing nodes [8]. There are two types of co-scheduling [9], Gang and Loosely.

- Gang: schedule process of a job at the same time at different allocated nodes or processors to this job.
- Loosely: originated from NOW (Network of Workstations) environments, seeks to solve the problems to scheduling serial and parallel workload together. This approach is based on the local scheduling and on communication organization of the involved nodes.

2.2. Co-Reservation

The co-reservation function is characterized by the coordination use of cluster resources in sequence as a grid configuration [8]. Usually are adopted timeslot tables to show the percentage of available resources allocated during a determinate period of time. These tables help to control current allocations and future reserves [10]. Figure 1 shows an example of timeslot table with As representing current allocations and Rs as future reservations.

2.3. Co-Allocation

Co-allocation can be defined as the simultaneous use of grid resources through multi-clusters [8]. According to Netto and Buyya [7], co-allocation is the resource allocation process from multiple administrative domains in a coordinated.

3. Related Research Work

In this section, we present some related works that present some related characteristic related to our research.

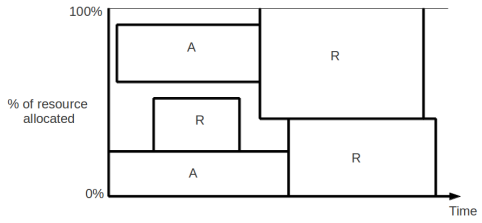

Figure 1. Timeslot Table

Advance reservation of resources method presented at Ferreira and Dantas [11] research work enables user to reserve resources between different clusters in a multi-clusters environment. The proposal uses semantic integration of multiple ontologies to match users requests with description of the system configurations. Furthermore, it includes fuzzy logic to control the distribution of tasks among clusters, since applications are subdivided into tasks and it can be executed in several clusters.

Sulistio et. al. [12] present a data structure based on Segment Tree and Calendar Queue, to provide efficiently management of advance reservations. This data structures, named as GarQ (Grid advanced reservation Queue), found available nodes, add requests and delete existing reservations efficiently and quickly, as shown by the experimental results. Furthermore, The authors have developed a mechanism to search for free time intervals closest to the requested reservation, if it was reject before.

Melo Martins et. al. [13] research presents a management mechanism design to opportunistic grid computing environments to handling the execution of applications with time deadlines set by users during their submission in the system. This mechanism is based in dynamic scheduling and re-scheduling and it was developed to execute in InteGrade middleware. In additional, the research proposal uses the advanced reservation mechanism to reserves the choose node to execute the submitted application if the node is busy, in other words, executing other application. As experimental results the authors carried out simulations in different opportunistic computing scenarios.

4. Proposal

In this section, it is presented our proposed architecture which provides an advanced reservation for opportunistic environments. Since grid environments are dynamic, an advanced reservation feature becomes important aspect, because it improves the scheduling process from an application [14], once the scheduler will know in advance when and, perhaps, where an application will be executed and an availability of resources for applications.

Researches such as [15] and [16]have the objective to reach the quality of service (QoS) in grid environments using advanced reservation. The main goal of these works is to guarantee that the resources will be available at some time in the future. In contrast to these proposals, our research work aims to achieve the best possible quality of service, once we are developing a proposal for an opportunistic environment. In other words, our proposal does not guarantee the availability and, therefore the application execution. Nevertheless, our proposal is relevant since we seek to improve both usability of the system to a user, as to organization and operation of the system.

The InteGrade project [5] was employed to develop our proposal. InteGrade is an opportunistic grid computing middleware whose purpose is to use idle computing capacity of workstations, academic laboratories and personal computers to execute applications which require high performance computing [17]. Its main goal is to offer to grid users high performance computing without degrading the performance of their machines, always prioritizing the

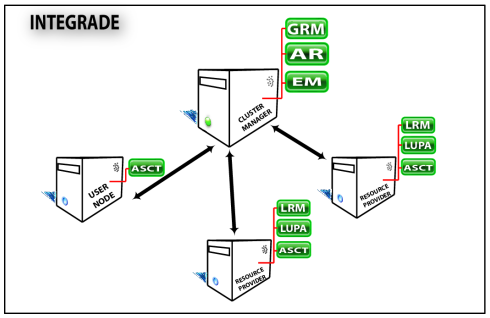

Figure 2. InteGrade Architecture

execution of local applications. The InteGrade was chosen for the development of this research because no advanced reservation exists in its basic architecture, it is a project developed through the partnership between Brazilian universities. Furthermore, it has open source and easy access, and the availability of efficient support.

4.1. InteGrade

Currently, InteGrade executes three types of applications [18]:

- *Sequential Applications:* single applications, with only one executable binary which require only one node to execute.
- *Parametric or Bag-of-Tasks Applications:* several application copies are distributed to different cluster's nodes. Each copy processes different sets of parameters and input files independently of the data of the others nodes.
- *Parallel Applications:* parallel applications in the InteGrade are based in BSP and MPI models.

The basic architecture of InteGrade grid is a cluster, in other words, a set of machines connected to a local network and organized hierarchically [18]. Figure 2 shows the major components of the InteGrade cluster. Each cluster has a Cluster Manager node which is responsible for the application scheduling and the storage information of each node belonging to the cluster. It has a set of nodes which offers computing resources to the cluster, named Resources Provider nodes. Furthermore, the cluster has User Nodes in which user can submit application to execution without provide their resources to the cluster.

The InteGrade architecture is composed of modules which they are responsible for cluster operation. The modules, depicted in Figure 2, are:

- *ASCT - Application Submission and Control Tool:* it is a friendly graphical users interface which enable user to submit applications to the cluster [17]. Users can specify hardware requirements and software platform, and monitor their application execution.
- *LRM - Local Resource Manager:* it is executed each cluster node. It collects information about the node status, such as memory, disk, CPU and network utilization. This information is send periodically to the GRM. It is also responsible for instantiating and executing applications schedule to its node.
- *GRM - Global Resource Manager:* it is a manager cluster. GRM receives information collected by LRMs about quantity and availability of resources belonging to the cluster, and it uses them to scheduling applications. It is responsible to communication with GRMs of others clusters.

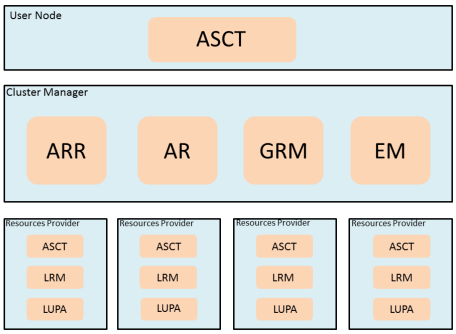

Figure 3. ARR Module

- *AR - Application Repository:* storage applications which are submitted by the ASCT to be executed later in the cluster.
- *EM - Execution Manager:* maintains information about applications submitted to cluster, such as submission and execution state, input and output parameters [18].
- *LUPA - Local Usage Pattern Analyzer:* it executes each cluster node and establishes usage pattern through analysis of resources use [19]. These patterns are used to predict idle period of the node and improve the applications scheduling.

4.2. Proposed Architecture

To enhance the InteGrade architecture, we proposed an advanced reservation of resources (ARR) module. This module enables users to require a reservation in advance resources from a cluster to be used in the future. The Figure 3 shows the insertion of the ARR module in the Cluster Manager in order to maintain the basic structure of InteGrade.

The ARR is composed of service modules which are responsible for its operation. These modules are: Reservation Service, Execution Service and Notification Service.

The Reservation Service deals with the management of reservation request. In other words, it verifies if there is a free time interval and reallocates the reservation if an interval resquested is not available. To reach this goal, the service receives from User Node information necessary for executing an application, originally provided by InteGrade. Examples are: application type (sequential, parametric or parallel), quantity for resources to application execution, identification of application in the Application Repository and the information relating to advanced reservation and stored them in the database.. The information required to advanced reservation is:

- *start-time*: execution start time;
- *end-time*: execution end time;
- *user*: user name to handle stored reservations;
- *email*: user e-mail to notifications.

The Execution Service is responsible for requesting the beginning of application execution. Then, the service searches the execution information in the database and sends it to Cluster Manager which starts the procedures to application execution.

The Notification Service warns users, by e-mail, when the process failure or in the initialization or completion of application execution.

The following section shows the application execution process in InteGrade using the Advanced Reservation of Resource module.

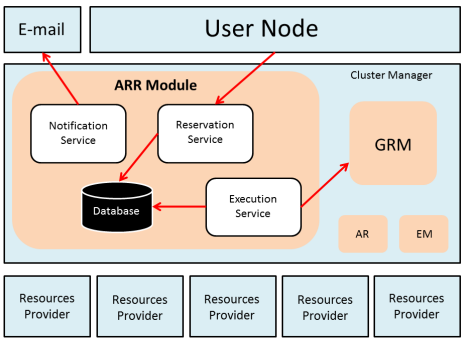

Figure 4. ARR Architecture

4.3. Execution Process

The execution process starts when user registers the application on AR. Then, user can choose between immediate and scheduled execution. Both for immediate execution and for scheduled execution, user specifies information such as quantity of resources to execution, application type (sequential, parametric or parallel) and input files. When user chooses scheduled execution, it is necessary, additionally, to specify information such as start execution time, end execution time and e-mail.

In the next step, user submits the execution scheduled request. At this moment, the ASCT sends information regarding the resource specifications to GRM. This module searches for nodes, in the cluster, which match with the resource requested by user. If the resources do not exist, user is notified by the ASCT. Otherwise, the ASCT sends the information regarding reservation to the Reservation Service. Then, the module verifies in the database whether there is, or not, reservation in the requested period. If there is not, the reservation is successful. Otherwise, the Reservation Service suggests an available period of time closest to the specific what it can be accepted or not. Upon the specification of period, the information regarding the resources, application, reservation and input files is stored in the database and the reservation is successful.

When executing the application, the Execution Service retrieves the information regarding the resources, in the database, and sends it to the GRM. The GRM verifies whether the node previously selected is still available to application execution. If the node still to belong to the cluster, though it is executed an application, the GRM saves the application status and verifies if there are in the cluster other available nodes to complete the execution. This occurs because we assume that a scheduled execution takes priority over immediate execution. In case of the node does not belong to the cluster, the GRM verifies whether there are other nodes with characteristics which match with user's requirements.

After selecting the node, the GRM sends an execution request to LRM of the candidate node. At this point, the LRM verifies, if indeed, the node has available resources. This procedure is necessary because the GRM maintains an approximate vision of the availability of nodes resources belonging to the cluster. If the node has not available resources, the LRM notifies the GRM which searches, again, for candidate node. However, if the candidate node is available, LRM requests the application to the Application Repository, it requests to Execution Service the input files and it executes the application. The LRM sends a notification of acceptance and after the completion of the execution to the Notification Service which notifies user through e-mail.

Table 1. Characteristics of machines

Machine	Quantity of Memory (MB)	Quantity of CPU
node 1	3072	1
node 2	4096	1
node 3	2048	1

Table 2. Reservations stored in the database

User	Start Time	End Time	CPU Usage (%)	Free Memory (MB)	Application Type	Reserved Node
A	09:00am	11:00am	< 70	≥ 2000	Parametric	node 2
B	12:00pm	02:00pm	< 50	≥ 1024	Sequential	node 3
C	06:00pm	07:00pm	without specifications	without specifications	Parallel	node 1
D	12:00pm	02:00pm	< 50	≥ 1024	Sequential	node 2

5. Experimental Results

In this section, we tested our proposal through of a case study which illustrates the InteGrade behaviour with the implementation of Advance Reservation of Resources module.

5.1. Environment

The cluster environment is composed of one Cluster Manager (GRM) and three Resources Provider (LRM). The Table 1 describes the quantity of memory of each LRM that belong to the cluster.

The GRM implements a heuristics to the alternation of the nodes which will meet the requisitions. The goal is to prioritize nodes which do not received requisitions recently, thereby the GRM minimizes the possibilities of the requisitions be sent to the same node [20]. The Table 2 shows the reservations stored in the database.

Although user to provide resources specifications in the reservation request, the reservation module reserves the selected node, instead of the resources. The goal of this method is to have no more than one reservation per node what would overload it.

5.2. Case Study

In this subsection we present a case study that simulates the system behavior with conflicts of the new reserves. The Table 3 presents four requests to be carried by four users.

The reservation request of user G is accepted without restrictions, once there is not reservation with the same period stored in the database. Then, in this case, the node 3 is reserved. The reservation request of user E is at the same time of user B, but it is accepted. This occurs because the environment is composed of three nodes there is still the node 1 available at this time. Then, the node 1 is reserved to user E.

On the other hand, reservation of user F is refused by the system because the period between 12:00pm and 02:00pm all nodes are reserved. Therefore there would be no sufficient resources to satisfy user F request. To solve this problem, the Reservation Service offers the reallocation of reservation to user. In this case, the module suggests the period between 03:00pm and 05:00pm, in other words, the free time interval closest to the one chosen by user. If user accepts the

Table 3. Reservations requested by users

User	Start Time	End Time	CPU Usage (%)	Free Memory (MB)	Application Type
G	08:00pm	09:00pm	< 50	> 2300	Parallel
E	12:00pm	02:00pm	< 70	> 1024	Sequential
F	12:00pm	02:00pm	< 70	> 1024	Parallel
A	10:00pm	11:00pm	< 70	> 1024	Parametric

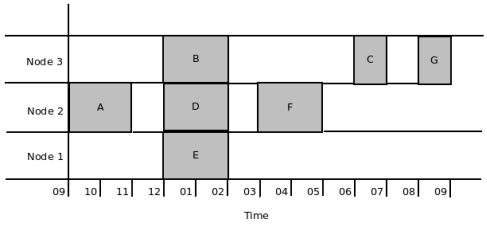

Figure 5. Timeslot table

suggestion, the node 2 will be reserved. The reservation request of user A is not accepted, since this user already has a reservation stored in the database. We adopt as a rule, that only one reservation can be stored in the database per user, this procedure avoids an only user reserves several periods of time and precludes the use of this mechanism by other users of the cluster.

As result, the Figure 5 shows the reservation allocation through timeslot table. The execution queue is sorted, firstly, by start time and then by request order.

6. Conclusions and Future Works

In this article, it was proposed an architecture which consisted in an approach of advance reservation of resources in opportunistic grid computing environments. The main goal was to improve applications scheduling and to provide a best-effort feature to users. The proposal was developed using the InteGrade project opportunistic grid computing middleware.

Therefore, a review from the InteGrade architecture was presented. In a second moment, it was presented the proposal and its operation with the InteGrade. It was, also, demonstrated an execution process of an application, when a user chose a scheduled execution. The approach was evaluated successfully through case study, where an InteGrade simulation behavior was executed in a real environment, where the combined configuration used an advanced reservation.

As a future work we intend to add in ARR module a service which manipulates the information collected by LUPA about usage patterns of each node and it generates idle periods prediction of entire the cluster to help users in the chosen the most appropriated period to execute their applications.

Acknowledgment

The authors would like to thank the attention of Professor Alfredo Goldman and his disposition to discuss part of the work presented in this article.

References

[1] Top500. http://www.top500.org/, January 2013.
[2] F. Kon and A. Goldman. Grid computing: Fundamental concepts and concrete cases. *Proceedings of the Tomasz Kowaltowski e Karin Breitman (Org.)*, pages 55–104, 2008.
[3] SETI@home. http://setiathome.ssl.berkeley.edu/, January 2013.
[4] BOINC. http://boinc.berkeley.edu/, January 2013.
[5] InteGrade. http://www.integrade.org.br, January 2013.
[6] W. Smith, I. Foster, and V. Taylor. Scheduling with advanced reservations. In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 127–132. IEEE, 2000.
[7] M.A.S. Netto and R. Buyya. Resource co-allocation in grid computing environments. *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications. IGI Global*, 2009.
[8] J. Qin, M.A.R. Dantas, and M. Bauer. Application programmers interactions enhancing a meta-scheduler in multi-clusters environment. Unpublished.
[9] A.C. Sodan. Loosely coordinated coscheduling in the context of other approaches for dynamic job scheduling: a survey. *Concurrency and Computation: Practice and Experience*, 17(15):1725–1781, 2005.
[10] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on*, pages 27–36. IEEE, 1999.
[11] DJ Ferreira, MAR Dantas, and M.A. Bauer. An ontological-fuzzy approach to advance reservation in multi-cluster grids. In *Journal of Physics: Conference Series*, volume 256, page 012021. IOP Publishing, 2010.
[12] A. Sulistio, U. Cibej, S.K. Prasad, and R. Buyya. Garq: An efficient scheduling data structure for advance reservations of grid resources. *International Journal of Parallel, Emergent and Distributed Systems*, 24(1):1–19, 2009.
[13] M.R. Melo Martins, B.T. Pereira Gomes, J.F. Gonçalves, and F.J. da Silva e Silva. Execution management of applications with runtime restrictions on opportunistic grids environments. In *PESARO 2012, The Second International Conference on Performance, Safety and Robustness in Complex Systems and Applications*, pages 1–7, 2012.
[14] L. Tomás, C. Carrión, B. Caminero, and A. Caminero. Meta-scheduling in advance using red-black trees in heterogeneous grids. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE, 2010.
[15] M. Siddiqui, A. Villazón, and T. Fahringer. Grid capacity planning with negotiation-based advance reservation for optimized qos. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 103. ACM, 2006.
[16] A. Di Stefano, G. Morana, and D. Zito. Advanced reservation in grid using pbs. In *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2008. WETICE'08. IEEE 17th*, pages 216–221. IEEE, 2008.
[17] Andrei Goldchleger, Fabio Kon, Alfredo Goldman, Marcelo Finger, and Germano Capistrano Bezerra. Integrade: Object-oriented grid middleware leveraging idle computing power of desktop machines. *Concurrency and Computation: Practice and Experience*, 16(5):449–459, March 2004.
[18] F.J. da Silva e Silva, F. Kon, A. Goldman, M. Finger, R.Y. de Camargo, F.M. Costa, et al. Application execution management on the integrade opportunistic grid middleware. *Journal of Parallel and Distributed Computing*, 70(5):573–583, 2010.
[19] Marcelo Finger, Germano C. Bezerra, and Danilo M. R. Conde. Resource use pattern analysis for predicting resource availability in opportunistic grids. *Concurrency and Computation: Practice and Experience*, 2009.
[20] Andrei Goldchleger. Integrade: A middleware system to opportunistic grid computing. Master's thesis, University of Sao Paulo, December 2004.