VECIMS 2006 – IEEE International Conference on
Virtual Environments, Human-Computer Interfaces, and Measurement Systems
La Coruña - Spain, 10-12 July 2006

Dynamic Simulation for Grid Computing Systems

A. Amato[1], M. Calabrese[1], V. Di Lecce[1] and V. Piuri[2]
[1] DIASS - Politecnico di Bari - V.le del Turismo 8, 74100 Taranto – Italy
[2] DTI - University of Milan - Via Bramante 65, 26013 Crema (CR) – Italy
[email: a.amato@poliba.it, calabrese.marco79@libero.it, dilecce@poliba.it, piuri@dti.unimi.it]

Abstract *- Grid computing systems are emerging as a consequence of the growing internet connectivity in combination with the need of shared resources to deploy large-scale scientific applications. In such a context, heterogeneity, decentralization, location, access and availability of resources need to be dealt with suitable simulation tools. In particular, overloading conditions may be critical and difficult to analyse when the grid is requested to guarantee affordable high performances. To deal with such a challenging task, a virtual simulation environment provided with a suitable graphical interface has been developed as the means for comparative analysis with real test bed activities.*

Keywords *– grid computing systems, overloading condition, graphical simulation tool*

I. INTRODUCTION

Distributed environments are becoming a challenge in the context of heavy computation. So far, many different middleware frameworks like Globus, Alchemi Unicore and others [1] provide scalable solutions to grid resource management. A great effort towards standardisation has led the grid community to foster and adopt the Open Grid Services Architecture [2], but much is still to be done. Grid systems may differ from each other by several components such as topology, resources, delivered services, scheduling policies and job submission frequency, hence it is difficult to find a useful model for grid designers. That's why numerous grids are designed for specific data or computational extensive applications rather than for general purposes. Specific grid applications can range in fact from molecular modelling for drug design to traffic simulation or distributed measurement systems for environmental monitoring.

The basics characterizing even elementary grid simulators can be summarized in this way: resource definition, user input jobs to be processed by each resource (also called gridlets), scheduling policy which maps gridlets to resources and some performance parameters to evaluate through simulation. A visual environment indeed helps designers to have a rapid visual sketch of their models and helps them to follow, step by step, the distance between the predicted responses and the experimental ones.

The Grid Computing and Distributed System Laboratory (GRIDS) has developed the *GridSim Toolkit* [1][3] to support modelling on large-scale distributed systems of heterogeneous resources. *GridSim* shows a user-friendly Graphical User Interface (GUI) which enables users to define realistic distributed computing scenarios, but it lacks in giving dynamic information about the ongoing processes. In particular, *GridSim* doesn't focus on dynamic queuing which is a relevant parameter in QoS-based scenarios (in [4] a detailed analysis on QoS in grid environments can be found).

On the other hand, some graphical tools (like G-Monitor [5]) provide friendly web-based interface to monitor, control and steer applications job, but they indicate, in general, only the actual state of the node (ready, busy, waiting, etc…) without plotting their parameters over time.

The lack of virtual simulation environments displaying run-time grid system evolution often prevents designers from attempting a deep analysis in grid performance evaluation. This affects a wider understanding of the processes involved in scheduling activities on grid systems.

To overcame these problems, this work presents Grid Dynamic Simulator (GDS), a GUI simulation tool developed with the aim of exploring the step-by-step behaviour of grid computing environments under different workload configurations. The research outcomes GDS may provide range from dynamic analysis of grid systems during saturation to comparison among scheduling performances. Moreover, GDS can be the basis for further studies on application domains different from the grid one such as Pear-to-Pear systems (P2P) or telecommunications networks.

In order to test the coherence between GDS platform and real test beds a local grid based on Globus 3.0 Toolkit [6] has been set up. The comparison between predicted and real behaviours is very useful indeed to heighten the quality of the model used for simulation. The grid used for our experiments consists of three machine classes equipped with different cpu speed availability and memory space. Each node has been equipped with Unix-like Operative Systems.

This research is partially supported by a grant from Italian Agency for Space (ASI) and Italian Ministry of Education, University and Research (MIUR) for the Grid.it project [7].

II. DOMAIN ANALYSIS AND TERMINOLOGY

Every geographically distributed computing infrastructure for advanced science and engineering, called *grid* for the first time in the mid 1990s [8], can be regarded as a definite set of nodes which can have computational, storage and data transfer capabilities. Each node shares its resources to make it possible for user to run applications on different locations. Grid systems have been widely implemented throughout the world for data-intensive and parallel applications overcoming the traditional client-server architecture by proposing a new one where each node can play the role of a service maker or a client requester. From this point of view, some authors have pointed out [9] the strict connection between grid and peer-

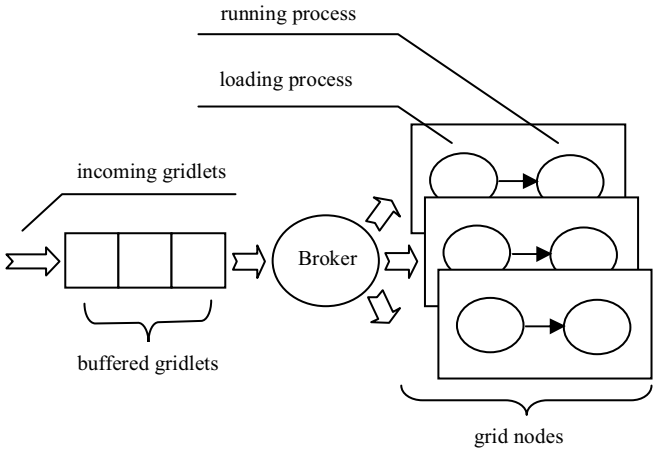

Fig. 1a. Adopted model for system simulation

to-peer (P2P), though the former is still more hierarchical than the latter. This means that the benefit that comes out from research modelling on grid environments may be then extended with little effort to P2P networks, for example in the field of Quality of Service analysis.

Some defining metrics of single-node computation speed are: Million Instruction (MI), Million Instruction Per Second (MIPS), Floating Point Operations Per Second (FLOPS) or SPEC[1]-like ratings. Each one of these metrics cannot completely describe the real behaviour of each node, however, MIPS, our present choice, is the most used. Every computational node may be subdivided in as many Processing Elements (PE) as there are CPUs available, but such level of detail has not been considered as necessary in this work.

Any user job submitted to any grid node is commonly called *gridlet*. If more than one gridlet are being simultaneously hosted by a node, they are supposed to run in time-shared mode and scheduled under round-robin policy. This affects the total MIPS amount which must be shared (equally) among all the gridlets running on the node during each simulation step. Each node has of course a limited number of free memory space and it is characterised by upload and download rate. On the other hand, each gridlet can be described at least by its computational load (measured in MI) and its run-time memory occupation (measured in MB or GB). Gridlets are supposed to be independent, that is, they don't have to be synchronised each other during their running. This assumption is not very restrictive, in fact it comprises a wide spectrum of scientific *Bag-of-Task*[10] applications like Nimgorod and SETI@home and, in general, every application which has to run different procedures on the same dataset or the same procedure on different datasets.

Finally, our attention has been focused on those grid settings characterised by the predominance of requests against available resources, which inevitably carries out a bottleneck configuration. These situations are some of the most interesting ones because they allow analysing policies to recover from saturation. In this context, a clear interface may give an immediate outline of the congested nodes, thus simplifying system understanding.

III. PROPOSED SIMULATION TOOL

GDS can be mainly divided into two components: a brokering core and a graphical interface. The former is the block that performs all the processes concerning gridlet submission and execution, the latter is the front-end block that enables user to define parameters for simulation and to constantly monitor the system behaviour.

A. Brokering core and simulation model

The *broker* is responsible for discovering available resources, deploying and monitoring gridlets, managing the process of synchronization and output collection. These activities take time and should be considered to compute total completion time, but, of course, their weight is inversely proportional to the time taken by bare gridlet processing. The brokering core, among all its tasks, also accomplishes the management of node behaviours during simulation, which depends on the model adopted for each single node. Considering that each state that every process (gridlet in our case) is supposed to encounter during their life cycle are essentially *loading*, *waiting* and *running*, a simple model has been carried out for node simulation. In such a model each node is supposed to handle both loading and running state for each scheduled gridlet, while the waiting state, to a first approximation, has not been considered. Only one loading per node at a time is allowed; all the unserved requests are buffered in FIFO mode. If more than one gridlet are present on a node in the running state at the same time, they are supposed to share CPU power equally in round-robin

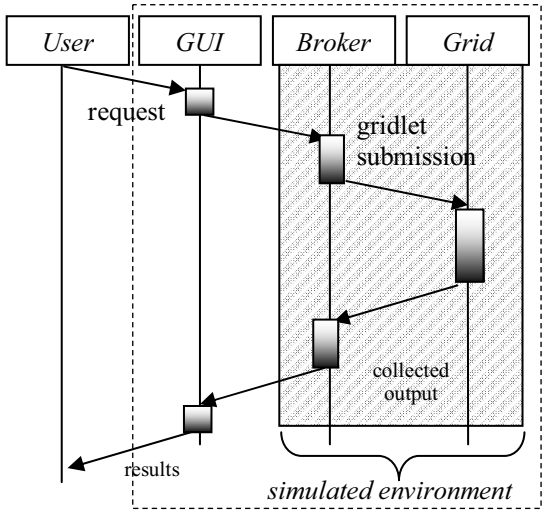

Fig. 1b. Sequential diagram of GDS architecture

[1] SPEC is the acronym for Standard Performance Evaluation Corporation, which is a foundation that operates in the field of performance evaluation for the newest generation of high-performance computers.

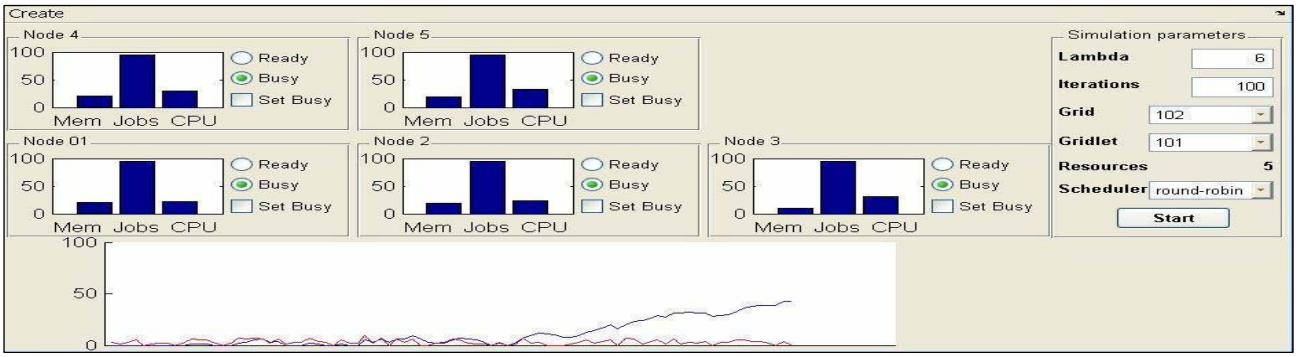

Fig. 3a. Screenshot of 5-node grid during saturation due to high lambda. Every node has reached its maximum job capacity thus entering a busy state, although allocated memory is negligible.

fashion. Fig. 1a gives a sketch of the adopted model.

To sum up, the brokering core mediates access to distributed environment, hiding grid complexity to grid users. It is also responsible for correct management of applications running on nodes and should offer an effective schedule to minimise the queue of pending gridlets. An outline of GDS architecture is presented in Fig. 1b.

B. Graphical User Interface

Dealing with the graphical representation of a large amount of dimensions is a challenging task. In the case of a computer grid in fact, the need of having a unique visual sketch of each process on each node can be hardly fulfilled. Assuming that parameter visualization depends on the user category, the choice to personalize the GUI for grid designers has been made. That's why a console-like panel reporting the instant snapshots of the nodes during simulation has been chosen. The buffer length (as displayed below the frames of the grid nodes) gives an aggregated estimation of the whole simulation process.

GDS GUI shows a right-side frame for simulation parameter settings (see Fig. 2). The user is given the possibility to modify the following parameters: λ (Poisson distribution parameter), the number of iterations before stopping, set of gridlets and nodes to be used for simulation

Fig. 2. Simulation parameters frame

(selecting their id), scheduling policy for gridlet-to-node mapping. The λ parameter defines the mean rate at which gridlets (taken from a given set) are submitted to the grid.

Once started with the simulation, the GUI main frame loads as many sub-frames as there are nodes involved in the simulation, while a poissonian load generator creates new gridlets according to the value of λ. Each sub-frame shows (by histograms) up-to-date information about memory occupation, cpu loading, number of job already scheduled on that node. A *ready* and a *busy* flag are provided to let user know respectively if the node is ready to load a new gridlet (that is, it has full bandwidth availability for download) and if the node has reached its maximal job number capability. If the node is in a busy state it is no more prone to accept new loads until it goes back again below its job number threshold. Finally, a switch button gives the user the ability to manually set one node to busyness, in order to follow evolving system state.

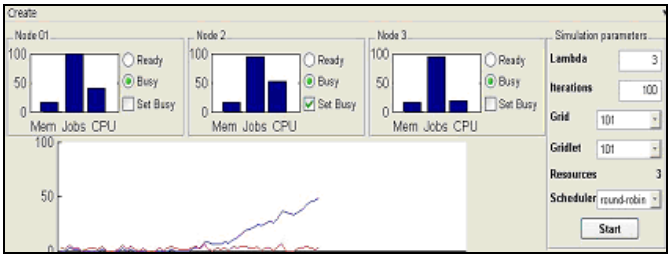

Fig. 3b. Screenshot of 3-node grid during saturation caused by an external asynchronous busy-setting on node 2

Fig. 3a illustrates a 5-node grid under saturation conditions. The Poisson parameter is set to 6, slightly greater than the number of resources. As screenshot shows in the plot downward the central area, the buffer occupation (higher curve) is diverging at the moment of the screenshot, and, as soon as the simulation goes on, the system will certainly progressively degrade its performances due to the overloading caused by lambda greater than the number of resources. The second curve represents the number of gridlets that call for service at each time step. Fig. 3b shows another configuration in which the grid system enters in the saturation zone for external reasons.

The displayed screenshot has been captured after that one node has been manually set to busy, thus not receiving gridlet loads any more. The Poisson parameter is set to 3, that is a value equal to the actual number of resources. Node 2 has been put off shortly after the beginning of the simulation. It is fairly evident that the buffer occupation (diverging curve) increases very sharply.

C. Post-processing analysis

During simulation steps, GDS keeps trace of scheduled gridlets, pending requests and node status in a log file.

On the one hand, this storing activity allows to evaluate simulation reliability by comparing simulation results with those coming from the experimental test bed (as it is explained in the next section). On the other hand, the logger also takes care of saving the set of resources and gridlets available for each experiment, to make it possible for designers to compare, for example, grid response to the same input batch but with different resources or scheduling policies. It's plain that if the grid grows in elements and complexity also the total amount of available data will be difficult to be managed with. For this reason, data representation is still an open research topic that would deserve some further specific investigation, which however goes beyond the purpose of this paper.

Fig. 4a compare the effect of two different scheduling policies when applied with the same input batch to the same grid system made of non-homogeneous nodes. Z-axis represents time expressed in simulation steps (each step is nominally equivalent to one second) while the XY-plane defines the instant buffered gridlet load (expressed in terms of MI) and the estimated finish time. This last one is the sum of the estimated times by which each node is supposed to finish its work, assuming that no other gridlet will enter the system. The buffer holds every unscheduled request until they are submitted. The adopted scheduling policies are very simple ones: "first free" submits any gridlet to the first available node, whereas "less busy" delivers the load to the node with minimal estimated finish time.

Each scheduling action, marked in the following plots either with diamond-shaped sign or with star-shaped sign in dependence of the scheduling type ("less busy" and "first free" respectively), determines a change in the state space, thus adding one node and its connecting arc into the graph.

In this simulation three gridlets of different size are supposed to be submitted all at once repeatedly in the first six simulation steps. Three nodes have been used to simulate the grid. Their speed has been chosen of 20, 30 and 40 MIPS respectively. It's clear from the plot that the two policies have similar effects on the grid state throughout processing except for the last steps where a slight difference occurs and can be observed in Fig. 4a.

It's interesting to note that in the same input conditions but with cpu speeds all equal to 30 MIPS (thus maintaining the total MIPS amount invariant between the two simulations) the grid outperforms the previous result in both scheduling policies (which gives almost perfectly coincident results in this case as fig. 4b displays).

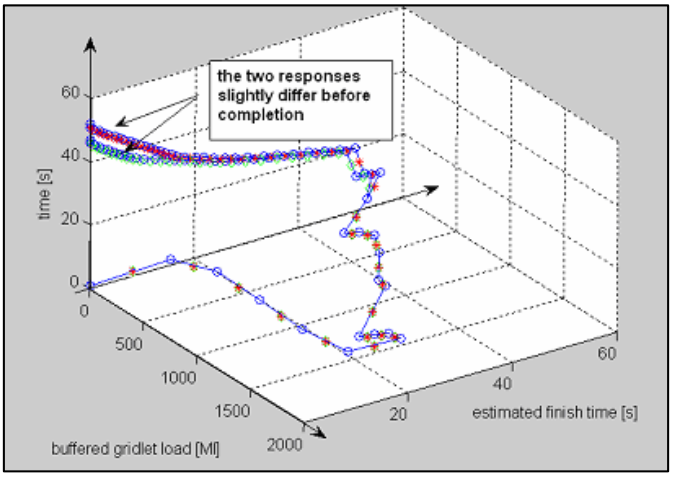

Fig. 4a. 3D-view comparison between two different policies applied to a non-homogeneous grid. The two responses differ only short before completion.

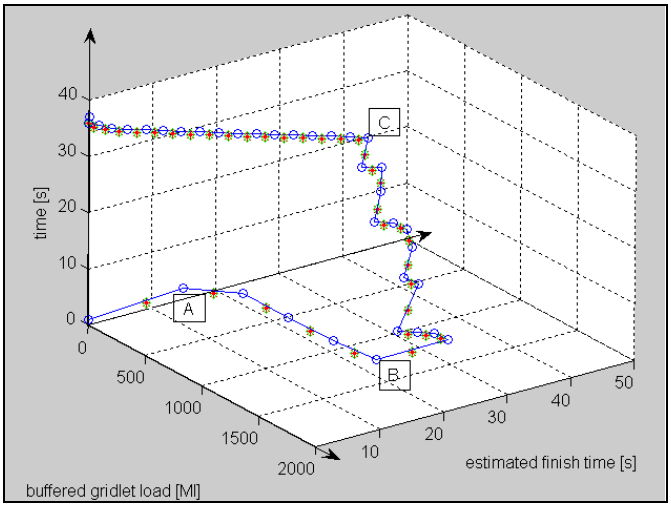

Fig. 4b. 3D-view comparison between the same two policies of the previous simulation applied to a homogeneous grid with the total MIPS amount equal to the one depicted in figure 4a. The two responses are practically coincident. Three labelled text boxes (A,B,C) highlight three key situations: when buffer starts being filled (A), when it reaches its maximum extent (B) and when it turns back to an empty state (C).

IV. RESULTS

To validate the effectiveness of GDS, a local grid environment based on Globus 3.2 Toolkit has been used. *GridBus Resource Broker*, a friendly environment for medium-level grid management, has been utilised to abstract from mere Globus APIs.

The experiments have been performed in this way: for each experiment a batch of gridlets, identical one another, but increasing in number has been submitted to the grid test bed. Three nodes have been chosen with the same speed , the same memory space and the same transfer bandwidth. This choice

comes out from the idea that, given a certain scheduling policy, if the resources are similar, the predicted run-time matchings between gridlet requests and computing nodes will be easily close to those real ones. In other words, unpredictability rises as much as difference among node resources rises. To operate in the best conditions, the possible system variance has been lowered at maximum extent by choosing a set of similar nodes. In our simulations and test bed the MI / MIPS ratio was set to 7. The node were set with a local buffer capacity of 2 gridlets.

Plotting the completion times at which the grid test bed satisfied all the batch requests, a characteristic curve has been drawn. The curve has been compared with the one given from simulation. In the simulated curve the shape is, of course, sharper. Without going into the detailed argumentation of technical issues, irrelevant for the purpose of this paper, it is fairly evident that the total completion time for each batch submission is the maximum among all the completion times of the single nodes.

Differently from simulated situation, in the real test it can happen that slight differences between nodes appear during computation. Furthermore, it has been supposed that the whole available CPU speed was equally shared among the gridlets but it isn't exactly so and some background noise due to OS running alters and smoothes the real curve. The gap between two successive "plateaus" (the ladder rungs) is about the time value used by a node to serve one single gridlet (MI/MIPS ratio). The curve maintains a slight slope along its plateaus due to the broker's overhead for gridlet synchronization. As Fig. 5 shows, the experimental curve and the simulated curve fit quite well, apart from certain offsets and smoothing trends which are due to some specific issues of the operating environment.

By means of several repeated experiments, the average test curve has been estimated for the 3-node grid configuration described above. A comparison between the test and the simulated curve has shown an average distance always below the value of 10% for high gridlet numbers.

This result is encouraging and indicates that using simulations to understand in advance the real system behaviours is possible and also effective.

V. CONCLUSION

In this work Dynamic Grid Simulator, a GUI tool that enables dynamic monitoring of grid systems over time, has been presented. This research activity springs from the necessity of having a friendly environment where it is possible to simulate distributed computing applications before making them run on real grid systems. A first modelling effort has been made with the aim of selecting the right parameters and defining the actual processes that have great influence in real situations. For the sake of simplicity no synchronism among gridlets has been taken into account. This could be enough for many large-scale scientific applications but it can be overcome with a bit further effort in terms of modelling and programming. It's our intention to enhance such a tool in order to make it more abstract from grid domain, thus making it possible to apply it to other research fields like freight transportation and telecommunication networks.

Another key issue which deserves to be explored, as we have stated before, is the problem of effective simulation data representation which can be carried out by using advanced 3d-view techniques.

The outcomes of our experiments have proved that in case of homogeneous resources, within a certain degree of approximation, it is possible to produce simulated behaviours not too far from those observable in real test beds.

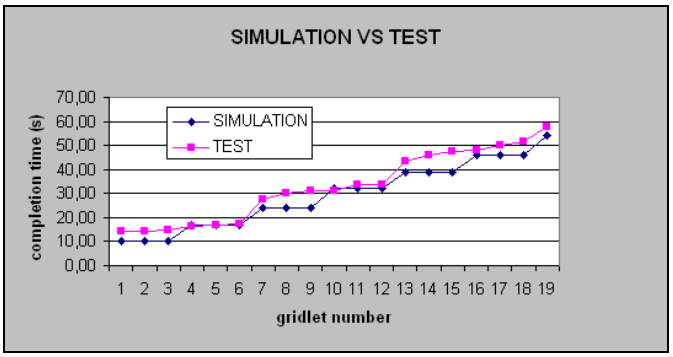

Fig. 5. Synopsis of the simulated and the experimental curve

REFERENCES

[1] Srikumar Venugopal, Rajkumar Buyyal, Lyle Winton. "A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids". *Journal of Concurrency and Computation: Practice and Experience*, Wiley Press, USA (accepted in Jan. 2005).

[2] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, J. Von Reich. "The Open Grid Services Architecture, Version 1.0", Informational Document, *Global Grid Forum (GGF)*, January 29, 2005.

[3] Rajkumar Buyya1 and Manzur Murshed, "GridSim: a toolkit for the modelling and simulation of distributed resource management and scheduling for grid computing", *Concurrency Computation: Pract. Exper. 2002*; 14 pp. 1175-1220.

[4] Rashid J. Al-Ali, Ali ShaikhAli, Omer F. Rana and David W. Walzer, "Supporting QoS-Based Discovery in Service-Oriented Grids", *Proceedings of the International Parallel and Distributed Processing Symposium,* 2003.

[5] Martin Placek and Rajkumar Buyya, "G-Monitor: A Web Portal for Monitoring and Steering Application Execution on Global Grids", *International Workshop on Challenges of Large Applications in Distributed Environments*. June 2003.

[6] Available at the link http://www.globus.org/toolkit/docs/.

[7] Further information on the Grid.it project may be found at the link http://www.grid.it/.

[8] Ian Foster, Carl Kesselman. "The Grid: Blueprint for a New Computing Infrastructure". Morgan Kaufmann publishers Inc., 1998.

[9] Domenico Talia and Paolo Trunfio, "Toward a Synergy between P2P and grids", *IEEE Internet Computing*, vol. 07, no. 4, pp. 96, 94-95, July/August 2003.

[10] W. Cirne, F. Brasileiro, J. Sauve, N. Andrade, D. Paranhos, E. Santos-Neto and R. Medeiros. "Grid computing for bag of tasks applications". *Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and E-Government*, Sept. 2003.