Problems for Resource Brokering in Large and Dynamic Grid Environments

Catalin L. Dumitrescu[1]

CoreGRID Institute on Resource Management and Scheduling
Electrical Eng., Math. and Computer Science, Delft University of Technology
Mekelweg 4, Delft, 2628 CD Delft, the Netherlands
c.dumitrescu@ewi.tudelft.nl

Abstract. Running workloads in a Grid environment may become a challenging problem when no appropriate means are available for resource brokering. Many times resources are provided under various administrative policies and agreements that must be known in order to perform adequate scheduling decisions. Thus, providing suitable solutions for resource management is important if we want to cope with the increased scale and complexity of such distributed system. In this paper we explore the key requirements a brokering infrastructure must meet in large and dynamic Grid environments and illustrate how these requirements are addressed by a specialized infrastructure, DI-GRUBER - a distributed usage service level agreement (uSLA) brokering service. The accuracy function of the brokering infrastructure connectivity and the performance gains when a client scheduling policy is employed are analyzed in high detail. In addition, a performance comparison with a P2P-based distributed lookup service is performed to illustrate the performance differences between two different technologies that address similar problems (Grids that focus on federated resource sharing scenarios and P2Ps that focus on self-organizing distributed resource sharing systems, in which most of the communication is symmetric).

1. Introduction

The motivating scenarios of our work are large grid environments in which virtual organizations (VOs) and agreements appear and vanish with a high frequency (every day or week). Such VOs might be companies requiring outsourcing services over short time intervals or scientific communities that want to participate temporarily in different collaborations with access to other types of resources. In these environments, we distinguish between two types of entities participating: resource *providers* and resource *consumers*. They may be nested: a *provider* may function as a middleman, providing access to resources to which the provider has itself been granted access by some other *provider*. While sharing policies issues can arise at multiple levels in such scenarios, the dynamicity of such an environment is also a problem. *Providers* want to express (and enforce) various sharing policies (what we call *usage service level*

[1] This research work is carried out for the CoreGRID IST project n°004265, funded by the European Commission.

agreements or *uSLAs*) under which resources are made available to *consumers*. *Consumers* want to access and interpret uSLA statements published by *providers*, in order to monitor their agreements and guide their activities. Starting from this environment and interactions model, our main focus is the identification of requirements and the provisioning of the design ingredients for building a scalable distributed resource brokering service that supports uSLA expression, publication, discovery, interpretation, enforcement, and verification in large dynamic Grid environments. We build on much previous work concerning the specification and enforcement of resource uSLAs [1-5], information lookup, scheduling and brokering services [6-8], the GRUBER broker [9], and the DI-GRUBER version [10].

The main contributions of this paper are on three dimensions. First, we identify several requirements a brokering infrastructure has to meet when deployed in large and dynamic Grid environments. We base our judgment on our past experience with the GRUBER framework in the Grid3 [11] context and the enhanced version, DI-GRUBER. Second, we present several novel DI-GRUBER performance measurements, namely the brokering accuracy function of infrastructure components' connectivity, and the gains in performance when using automated decision point scheduling for the clients. Third, we realize a performance comparison with a P2P-based system for file management. The paper also introduces two major technical enhancements to the DI-GRUBER two layer brokering infrastructure: WS-Index Service-based infrastructure discovery [6] and a specific solution for handling infrastructures decision points' scheduling in order to meet the outlined requirements [10]. The first enhancement takes advantage of the WS-Index Service functionalities that acts as a lookup service. Each GRUBER decision point registers itself with a predefined list of WS-Index Services at startup and it is automatically deleted when it no longer provides brokering services. The second enhancement also takes advantage of the WS-Index Service to discover the most appropriate decision point (DP).

2. Brokering Key Requirements for Large and Dynamic Grids

This work targets Grids (and any large distributed systems in general) that may comprise hundreds of institutions and thousands of individual investigators where the participants often join or leave the environment [11]. Moreover, each individual investigator and institution may participate in, and contribute resources to multiple collaborative projects that can vary widely in scale, lifetime, and formality [10, 12]. Such globally distributed systems provide several key benefits over large centralized solutions, in particular: maintenance costs and upgrade operations are more easily handled and there are no single points of failure. Two main environment examples of this class are introduced next.

2.1. Grid Environment Examples

Open Science Grid (previously known as Grid3 [11]) is a multi-virtual organization (multi-VO) environment that sustains production level services required by various physics experiments. The Grid3 infrastructure had comprised more than 30 sites and 4500 CPUs, over 1300 simultaneous jobs and more than 2 TB/day aggregate data traffic. The participating sites were (are) the main resource providers under various

conditions. Thus, we consider that OSG/Grid is a good example of the kind of environments we envisage for the work in this paper. However, with times we believe that this infrastructure can grow. For example, the number of sites can increase by means of new joins; the rate of jobs can jump when new scientific communities will want to solve high computer power consummative applications. Thus, the resource management infrastructure we envisage in this paper targets Grid environments ten to hundred times bigger than today OSG.

The other Grid testbed example is the LHC Computing Project (LCG). LCG targets to build and to maintain a data storage and analysis infrastructure for the entire high energy physics community that will use the LHC (Large Hadron Collider) [13]. The data from the LHC experiments will be distributed around the globe, according to a four-tiered model. Two of the goals of the LCG project include developing and deploying computing services based on a distributed Grid model, and managing acquisition, installation, and capacity planning for the large number of commodity hardware components. The expected size of the entire community is around 5000 scientists in 500 research institutes and universities worldwide. The analysis of the data, including simulations, requires around 100,000 CPUs. Such a distributed system presents a number of significant challenges; the most important one from our point of view is the provisioning of controlled resource sharing mechanisms so that different groups have fair access, based on their needs and contributions, to the infrastructure.

2.2. Resource Brokering Key Requirements

The resource brokering (and scheduling) problem in such Grid environments encompasses intertwined requirements, while the most important three ones in our vision are: *support for brokering of numerous resources*, *an adequate level of accuracy of the brokering infrastructure* and *fault-tolerant brokering*.

- *Support for Brokering of Dynamic and Numerous Resources (scalability):* dynamicity implies in our view that various communities, providers or VOs might join a Grid environment for short (days to weeks) time intervals in order to solve fast various problems. This dynamicity imposes certain technical requirements, such as rapid propagation of information about available resources in the brokering infrastructure and of the new administrative policies under which these resources are made available. When the environment is large (composed of hundreds to thousands simultaneous providers and more than thousands of consumers), the brokering solution must be scalable enough to handle such an infrastructure.
- *Adequate Level of Brokering Accuracy Independent of the Infrastructure:* regarding management information, an important problem is the accuracy of information provided by a brokering service in order to perform adequate scheduling decisions. Even more, for a distributed infrastructure, several operations have to be considered, such as propagation, reconciliation and removal. These operations may occur whenever new decisions are performed and new resources join or leave the environment. The entire brokering infrastructure must become aware of these changes in a timely fashion manner.
- *Fault-tolerant Resource Brokering:* fault tolerance is important from a client point of view. Even when a client cannot contact a brokering decision point, it still expects to perform scheduling operations over the Grid with a lower but acceptable

execution performance. Also, when many clients perform queries, the brokering infrastructure must be able to cope with this request load. Even the reader might think about the P2P networks and their properties to re-organize, we pursue the path of scheduling the brokering decision points as any other resources. Thus, the employment of an adequate strategy becomes important in this approach.

3. Illustrating the Key Requirements in a Concrete Case

We now introduce the main concepts and tools used in this paper. We start with the WS-Index Service (monitoring and discovery service [6]) used as a supporting tool and introduce afterwards our brokering infrastructure (DI-GRUBER [10]) used as a vehicle for proving our assumptions.

3.1. WS-Index Service

WS-Index Service [6] is a standard component of the Globus Toolkit (one of the Grid technologies largely used in science and industry [14]). It provides specialized functions for resource and service monitoring and discovery, and it is used as the central rendezvous point by our brokering infrastructure. While someone might consider the WS-Index Service a bottle-neck, our previous experiments proved that its scalability is well beyond our needs. Thus, WS-Index Service's main function in our infrastructure is to act as a specialized directory of all DI-GRUBER decision points for all clients and the decision points themselves, and for infrastructure management.

3.2. DI-GRUBER (A Distributed Grid Resource uSLA-based Broker)

GRUBER [9] is a prototype Grid V-PEP and S-PEP infrastructure that implements the brokering functionalities required for steering workloads in a distributed environment based on uSLAs. It is able to perform job scheduling based on notions such as sites, VOs, VO groups, and uSLAs at various levels [4]. Currently, GRUBER is implemented as a Grid Web Service using the Globus Toolkit (GT4) technologies [14]. As an additional clarification, GRUBER does not perform job submission by itself, but can be used in conjunction with various grid job submission infrastructures. So far, we have interfaced GRUBER for job execution with the Euryale and Pegasus planners, largely used on Grid3 [11].

However, managing uSLAs within environments that integrate participants and resources spanning many physical institutions is a challenging problem when a centralized infrastructure is employed. A single unified uSLA management decision point providing brokering decisions over hundreds to thousands of jobs and sites can easily become a bottleneck in terms of reliability as well as performance. DI-GRUBER, an extension to the GRUBER prototype, was developed as a distributed uSLA-based resource broker that allows multiple decision points to coexist and cooperate in real-time. DI-GRUBER targets to provide a scalable management service with the same functionalities as GRUBER but in a distributed approach [10]. It is a two layer resource brokering service, capable of working over large Grids, extending

GRUBER with support for multiple brokering decision points that cooperate by periodically exchanging status information.

3.3. DI-GRUBER Enhancements to Meet Previous Requirements

DI-GRUBER was developed as a distributed uSLA-based grid resource broker that allows multiple decision points to coexist and cooperate in real-time. The problem is that without support for dynamic discovery of the brokering infrastructure, some of the advantages offered by this infrastructure become impractical. Here we outline how the interfacing and integration with the WS-Index Service practically fulfills the requirements enumerated in Section 2.

- *Transparent Decision Point Bootstrapping:* As already described, the ability to bring up a decision point is important in a large and dynamic Grid. Our proposed solution uses the functionalities offered by the WS-Index Service for various clients by employing the notion of rendezvous point. In our implementation, each DI-GRUBER decision point registers with a predefined WS-Index Service at startup, while it is automatically deleted when it vanishes.
- *Transparent Client Scheduling:* Further, all decision points and clients can use this registry to find information about the existing infrastructure and select the most appropriate point of contact. When we use the term *most appropriate*, we refer to metrics such as load and number of clients already connected. The scheduling policy employed by each client in selecting a decision point was the *least-used* (*LU*) strategy. Also, whenever a decision point stops responding, its clients query automatically the registry and select a new different decision point to communicate.
- *Failure Handling:* While dynamic DI-GRUBER decision point bootstrapping might be difficult to automate in a generic environment, the solution we have devised is simple. Every time a client fails to communicate or to connect with a decision point, it registers with the WS-Index Service a request fault. Such faults can be consumed by a specialized entity that based on various policies starts dynamically new decision points by means of the WS-GRAM service.
- *Brokering Infrastructure Accuracy Identification:* An important aspect of the work in this paper is to identify the accuracy of our brokering infrastructure function of the connectivity of each decision point to the rest of the network. This analysis falls into the same class of scenarios where a decision point has only partial knowledge, and is on the same path as the analysis of dealing with stalled information, measured and analyzed somewhere else [10].

4. DI-GRUBER Infrastructure Performance Results

Here we report on our latest results [10] while also considering some of our previous results. We used one to ten DI-GRUBER decision points deployed on the PlanetLab nodes [15]. Each decision point maintains a local view of the environment configuration and via periodic exchanges (in the experiments that follow every three minutes) with other decision points acquires the necessary knowledge about recent job dispatch operations or other changes in the system (new resources, new uSLAs).

The three metrics employed for analysis are **Throughput, Response** (or Average Response Time) and **Accuracy**. **Throughput** is defined as the number of requests

completed successfully by the service per time unit. **Response** is defined by the following formula (with RT_i being the individual job time response and N being the number of jobs processed during the execution period): $\boldsymbol{Response = \Sigma_{i=1..N}\ RT_i / N}$. Finally, we define the scheduling accuracy for a specific job (SA_i) as the ratio of free resources at the selected site to the total free resources over the entire grid. **Accuracy** is then the aggregated value of all scheduling accuracies measured for each individual job: $\boldsymbol{Accuracy = \Sigma_{i=1..N}\ (SA_i) / N}$.

For all the experiments, we used synthetic workloads with a constant arrival rate of 1 job/s for each client or as soon as the previous scheduling decision was served that overlaid work for 60 VOs and 10 groups per VO. The experiment duration was one hour in all cases. Each of the 120 submission hosts ("clients") maintained a connection with one decision point; selected either under the *random* or the *least used scheduling* policy. The emulated environment was composed of 300 sites representing 40,000 nodes. The entire configuration was based on Grid3's landscape in terms of number of CPUs, disk space, network connectivity, etc., but ten times larger [10].

4.1. Decision Accuracy with Brokering Network Mesh Connectivity

First, we measure **Accuracy** of the brokering infrastructure function of the decision points' average connectivity. We consider practically three cases: *full connectivity* (DPs see each other), *half connectivity* (each DP collects information only from half of all the others), and *one-fourth connectivity* (each DP collects information only from a quarter of all the others). The results were achieved by means of the DI-GRUBER infrastructure in all three above configurations and are captured in Table 1.

Table 1. DI-GRUBER Accuracy Performance with Mesh Connectivity

	Connectivity	**Util**	**Accuracy**
Requests Handled by GRUBER	*All*	35%	75%
	One half	27%	62%
	One fourth	20%	55%
Total Request	*All*	41%	68%
	One half	30%	60%
	One fourth	21%	50%

We can observe that the performance of the brokering infrastructure drops substantially with connectivity degree of each individual decision point. As an additional note, the **Util** parameter is low because jobs do not start all in the beginning, but are scheduled every second during the entire execution period. In a nutshell, **Accuracy** drops almost linearly with clients' connectivity degree, intuitively.

4.2. Decision Point Scheduling and Performance Gains

Second, we focus on capturing the gains a client can achieve in term of performance when a *least-used* service selection policy is employed vs. the *random* scheduling policy we employed before. The results for the *random* scheduling policy are captured in Fig. 1 and Fig. 2, while the results for *least-used* scheduling policy are captured in Fig. 3 and Fig. 4. As can be observed in the first two figures, the

distributed service provides a symmetrical behavior with the number of concurrent machines that is independent of the state of the Grid (lightly or heavily loaded). Also, with three decision points, **Throughput** increases slowly to about 4 job scheduling requests per second when all testing machines are accessing the service. With 10 decision points, the average **Response** time decreased even further to about 13 seconds, and the achieved **Throughput** reached about 7.5 queries per second [10].

Next two figures report the experiments performed when using the WS-Index Service and *LU* scheduling policy was employed by each client. We must mention that we also used this time a final GT4 release based implementation. As can be easily observed, the results show improvement in terms of both **Response** and **Throughput**. Clients achieved a more stable response time compared with the one in the previous set of tests. The **Response** metric's value is always less than 30 seconds for 3 decision points, and less than 10 seconds for 10 decision points. The **Throughput** metric's value shows even higher improvements, reaching a constant value of 5 queries per seconds for 3 decision points, while going us up as 16 queries per second for 10 decision points.

However, on average, we find modest improvements for 3 decision points (19% higher throughput and 8% lower response time) and significant improvements for 10 decision points (68% higher throughput and 70% lower response times). From this observation we conclude that, practically, the request load was better balanced among the decision points and the infrastructure was able to achieve higher **Throughput** and lower **Response**.

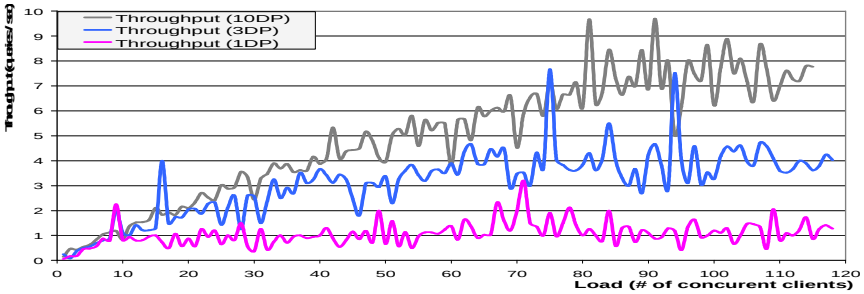

Fig. 1. DI-GRUBER Throughput (1, 3 and 10 Decision Points)

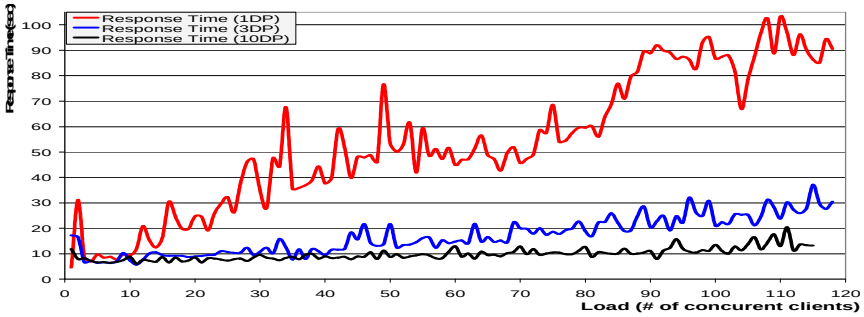

Fig. 2. DI-GRUBER Response (1, 3 and 10 Decision Points)

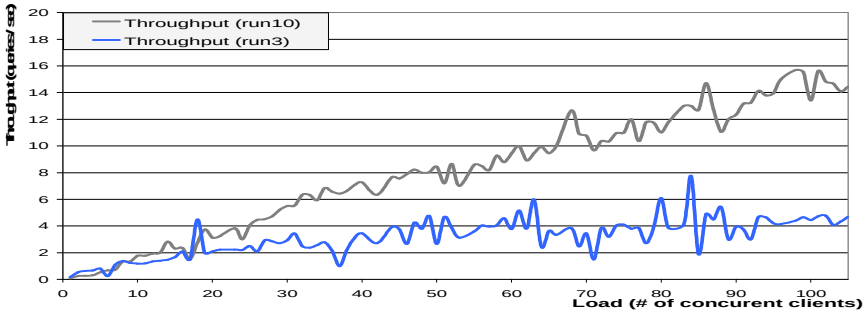

Fig. 3. DI-GRUBER Throughput (3 and 10 Decision Points)

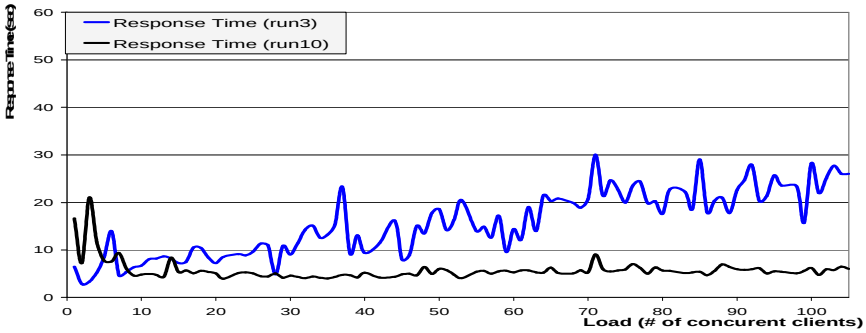

Fig. 4. DI-GRUBER Scalability Response (3 and 10 Decision Points)

4.3. Comparison with a FreePastry-based Lookup Service (PAST)

For convincing the reader that even though DI-GRUBER's transaction throughput seems low compared to '*other transaction processing systems*', we have performed further performance studies by means of DiPerF [16] on PlanetLab for a pretty well know distributed lookup service. The service chosen for testing was the PAST application [7], built on top of the PASTRY substrate.

The chosen setup was very similar to the one used for DI-GRUBER: the same PlanetLab nodes (around 120). This time we used five machines for running permanent PAST nodes, while the rest ones were brought up dynamically, joining and leaving the network in a controlled manner. Again, we used only one of the five nodes as the main contact point (a node situated at the University of Chicago). The rest ones were maintained as backup and to mimic the DI-GRUBER network. The length of the experiment was again one hour, while each joining node requested a lookup and an insert operation every second (or, if the previous operation took more than one second, at soon as the previous operation ended).

Our performance results are presented in Fig. 5. The measurements show that for insert and lookup operations, the PAST's response time is around 2.5 seconds with a higher variance in the beginning (the stabilization of the P2P network), while the throughput goes as up as 27 transaction per second in average. Also, the message lost

rate for this ad-hoc network was pretty high compared with the one of DI-GRUBER. However, the network stabilization delay is higher for the P2P system (first 18% of the experimental time) compared with DI-GRUBER clients' instantaneous network join operation. Our last note is that all operations were performed and measured on the local nodes (insertion followed by lookup); each node was responsible to propagate the results further (thus the higher response time and lower throughput than in the case of employing the continuation).

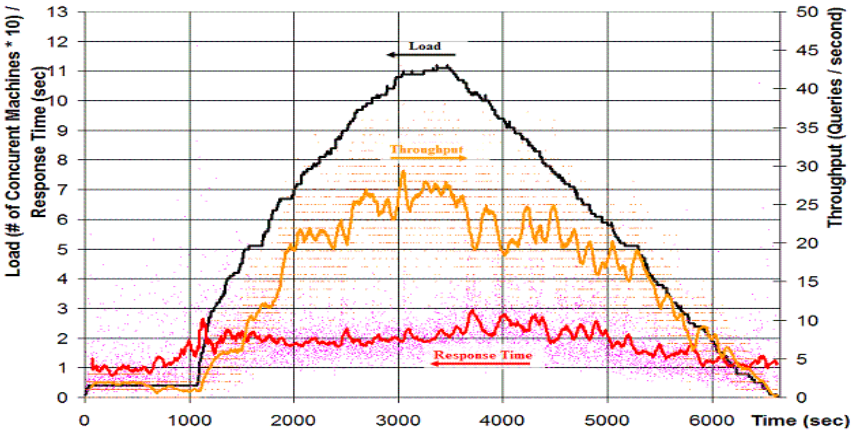

Fig. 5. PAST Network Response Time (left axis) and Throughput (right axis) for a variable Load (left axis * 10) on 120 PlanetLab Nodes

5. Conclusions

Resource management within large VOs that integrate participants and resources spanning multiple physical institutions is a challenging problem. The main question this paper addresses is "*what are the key requirements an already existing management infrastructure should meet in order to support large and dynamic Grid environments?*". The contributions of this paper are represented by results we achieved on three dimensions: we have identified three key requirements for extending a resource management service for large and dynamic Grid environments (and any other distributed systems in general), analyzed these requirements by means of a real infrastructure in a real case scenario, and also compared the performance results of the considered infrastructure with the ones of a P2P-based service.

Our experimental results showed how the brokering accuracy decreases with the loss of connectivity for a single decision point instance, while the performance of the system almost doubles in the 10 decision points' case due to the better repartition of the clients with the DI-GRUBER's nodes. The last set of experiments, the comparison performance tests, convinced us that even though DI-GRUBER's performance may seem low compared with a cluster resource manager, its performance is comparable in a similar environment in terms of response time and throughput with a distributed P2P system that, however, employs less functionality than the Grid counterpart technology.

Acknowledgments: I would like to thank Ian Foster, Michael Wilde, Jens-S. Vöckler, Yong Zhao, Ioan Raicu and Luiz Meyer for their support and discussions during the development of the (DI-)GRUBER infrastructure.

Bibliography

1. Lupu, E., *A Role-based Framework for Distributed Systems Management*, in *Department of Computing*. 1998, University of London: London.
2. Dan, A., et al. *Web Services on Demand: WSLA-driven automated management*, S. Journal, Editor. 2004, IBM. p. 136.
3. Dumitrescu, C. and I. Foster. *Usage Policy-based CPU Sharing in Virtual Organizations*. in *5th International Workshop in Grid Computing*. 2004. Pittsburgh.
4. Dumitrescu, C., M. Wilde, and I. Foster. *A Model for Usage Policy-based Resource Allocation in Grids*. in *6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*. 2005. Stockholm, Sweden.
5. Pearlman, L., et al. *A Community Authorization Service for Group Collaboration*. in *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*. 2002.
6. Czajkowski, K., et al. *Grid Information Services for Distributed Resource Sharing*. in *10th IEEE International Symposium on High Performance Distributed Computing*. 2001: IEEE Computer Society Press.
7. Rowstron, A.I.T. and P. Druschel. *Storage Management and Caching in PAST, a Large-Scale, Persistent P2P Storage Utility*. in *Symposium on Operating Systems Principles*. 2001.
8. Ludwig, H., et al., *A Service Level Agreement Language for Dynamic Electronic Services*. IBM Research Report RC22316 (W0201-112), January 24, 2002.
9. Dumitrescu, C. and I. Foster. *GRUBER: A Grid Resource SLA Broker*. in *Euro-Par*. 2005. Portugal.
10. Dumitrescu, C., I. Raicu, and I. Foster. *DI-GRUBER: A Distributed Approach for Grid Resource Brokering*. in *Super Computing* (*SC'05)*. 2005. Seattle.
11. Foster, I. et al. *The Grid2003 Production Grid: Principles and Practice*. in *IEEE International Symposium on High Performance Distributed Computing*. 2004: IEEE Computer Science Press.
12. Dumitrescu, C., I. Raicu, and I. Foster. *Experiences in Running Workloads over Grid3*. in *Grid and Cooperative Computing* (GCC'05). Beijing. China. 2005.
13. *LHC Computing Project*. 2004.
14. M. Humphrey, G.W., K. Jackson, J. Boverhof, M. Rodriguez, Joe Bester, J. Gawor, S. Lang, I. Foster, S. Meder, S. Pickles, and M. McKeown. *State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations*. in *4th IEEE International Symposium on High Performance Distributed Computing (HPDC-14)*. 24-27 July 2005. NC.
15. Chun B., D.C., T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, *PlanetLab: An Overlay Testbed for Broad-Coverage Services*. ACM Computer Communications Review, 3, July 2003. 33(3).
16. Dumitrescu, C., et al. *DiPerF: Automated DIstributed PERformance testing Framework*. in *5th International Workshop in Grid Computing*. 2004. Pittsburgh.