A WSDM-Based Architecture for Global Usage Characterization of Grid Computing Infrastructures

Glauco Antonio Ludwig[1], Luciano Paschoal Gaspary[2], Gerson Geraldo Homrich Cavalheiro[1], and Walfredo Cirne[1]

[1] Universidade do Vale do Rio dos Sinos (UNISINOS), Brazil
{glaucol, gersonc}@unisinos.br
[2] Universidade Federal do Rio Grande do Sul (UFRGS), Brazil
paschoal@inf.ufrgs.br
[3] Universidade Federal de Campina Grande (UFCG), Brazil
walfredo@dsc.ufcg.edu.br

Abstract. Current solutions to characterize grid computing usage are limited in three important aspects. First, they do not provide a global, uniform view of the use of infrastructures comprised of heterogeneous grid middleware. Second, they do not allow the specification of policies to publicize the collected information. Third, they do not generate statistics about the applications that are executed on the grid. To fill this gap, we propose an architecture based on the Web Services Distributed Management standard and on access control policies to characterize global usage of grid computing infrastructures, even when such grids are formed by heterogeneous middleware packages. We introduce this architecture and present preliminary results obtained with a prototype.

1 Introduction

The lack of solutions to manage complex systems such as grid computing infrastructures has hampered the widespread use of this technology, specially in the corporate arena. When used in large scale, involving many institutions and participants, it is necessary – in addition to managing faults, configuration, performance, and security – to characterize the use of the grid infrastructure. The objective is sixfold: (*i*) obtain detailed information about the applications executed on the grid (i.e. where they were executed, execution duration, resources consumed, and users who submitted them); (*ii*) identify the execution of malicious applications (an application executing for a very long time could indicate the intention of only consuming grid resources, impeding their legitimate use); (*iii*) determine users who contribute resources for the grid and those who consume most of the computing power available; (*iv*) guarantee a fair scheduling of jobs on the grid, denying or allowing users to execute new applications based on their usage history (users that exceed a usage quota should not have access to the resources of the grid); (*v*) identify stations of the grid which are not contributing in a productive way (a certain station receives jobs to compute and ends up failing to process them very often); and (*vi*) follow the evolution of the grid computing infrastructure (allowing the recognition of usage patterns and trends).

R. State et al. (Eds.): DSOM 2006, LNCS 4269, pp. 124 – 135, 2006.
IFIP International Federation for Information Processing 2006

Currently, there are several solutions [2, 4, 9, 11, 12, 14, 17] that provide the administrator with statistics about the use of grid systems. Most of them are limited to monitor the status of environment resources, neglecting statistical and historical data about the execution of applications on the grid. For example, with the existing solutions it is possible to observe that a certain station had its CPU highly loaded for the past twelve hours. However, one cannot precisely infer the reason for that. In short, these solutions do not relate in a proper way information about the resources and the applications running on top of them.

It is usual to have heterogeneous grid middleware being employed in a large scale, inter-institutional grid computing infrastructure. Each of them (e.g. Globus [7], Condor [3], and OurGrid [13]) uses its own indicators to report job execution. In this context, a problem to be overcome by a characterization and accounting solution is the use of a common format to represent basic accounting and usage data. Existing tools do not handle well such heterogeneity and, therefore, are unable to provide the grid administrator with a uniform, integrated view of the usage of a heterogeneous grid setup.

While institutions involved in a grid computing infrastructure are willing to share usage information with its collaborating sites, each institution has a different security policy to be applied. Some of the data gathered by a characterization and accounting tool can be classified (e.g. details of the station that executed a job) and, therefore, their distribution conflicts with the security policy in place. Here, again, there is no support in current solutions to define and enforce a policy for the distribution of data to the sites comprising the grid.

Considering the above limitations, we propose an architecture to support global usage characterization of grid computing infrastructures composed of different middleware packages such as Globus, Condor, and OurGrid. Our architecture allows the grid administrator to obtain long-term statistics and real-time notifications about major events generated by both the grid resources and applications. Currently, we are focusing on managing the grid applications, filling a not yet explored gap. In addition, we propose a mechanism to allow each institution participating of the grid infrastructure to specify and enforce policies for the publication of usage information generated inside the institution.

In line with current grid technologies, whose components have been organized as web services, the architecture relies on the recent OASIS Web Services Distributed Management (WSDM) standard [15]. Despite the fact this work focuses on the issue of accounting, we regard it as a building block to achieve a more scalable and autonomous management solution, based on the composition/choreography of complementary WSDM-compliant grid management services.

The remainder of the paper is organized as follows. Section 2 discusses related work on usage characterization of grid computing infrastructures. Section 3 introduces the architecture and Section 4 details its components. Section 5 presents preliminary results obtained with a prototype. Section 6 closes the paper with concluding remarks and perspectives for future work.

2 Related Work

Important steps have been taken towards characterization of grid computing infrastructures, with relevant mechanisms being proposed in the past to address specific issues such as distributed resource usage monitoring. Nevertheless, they are not fully prepared to (*i*) provide a global, uniform view of the use of infrastructures comprised of heterogeneous grid middleware, (*ii*) allow the specification of policies to publicize the collected information, and (*iii*) generate statistics about the applications executed on the grid.

Solutions such as Remos [4], visPerf [9], GridRM [2], MonALISA [12], and Ganglia [11] are unable to operate over heterogeneous infrastructures, which share re-sources employing distinct grid computing middleware packages. Since each middleware tends to adopt a proprietary format to represent statistics about applications executed and resources consumed, they are not prepared to collect such statistics and normalize them using a uniform information model. Due to this limitation, it becomes a challenge to provide the grid administrators with a global view of the grid usage.

Regarding selective dissemination of grid usage information, just a few systems – such as visPerf and GridRM – provide access control mechanisms. These mechanisms, however, are limited. visPerf, for example, employs only one credential for the whole grid; users that possess this credential can access all information generated by the infrastructure. GridRM adopts an access control mechanism based on administrative domains. When an institution decides to make part of a grid infrastructure that uses GridRM, it starts to publicize – to the domains of interest – all the information generated by the monitoring agents located within its administrative boundaries. In this context, we claim that neither of the approaches are flexible enough to handle the definition and enforcement of information publicization policies.

MonALISA and Usage Record (UR) [10] are the only solutions that provide statistics related to the execution of applications on the grid. The former does not relate such statistics with the resources consumed, hampering a precise understanding of the grid usage. The latter constitutes an information model that merges data from both applications executed and resources consumed. Since it is not a software architecture, UR does not define how this data must be collected, processed, consolidated, and presented to the grid administrator.

3 Architecture Overview

The architecture proposed is composed of four major components: *Publishers*, *Characterization Service*, *Authorization Service*, and *Management Application*. Figure 1 illustrates a general view of the architecture and the relations between its components.

Publishers are grid-technology-dependent piece of software responsible for monitoring the occurrence of predefined events generated by the components comprising the grid (e.g. a log message informing that a certain application has been submitted). Whenever such an event is observed, the publisher extracts the data of interest from the event, normalizes and send it to the characterization service. Publishers are supposed to be developed and distributed across the infrastructure, in

accordance to the needs of every grid middleware taking part of the overall inter-institutional setup. For example, supposing a scenario composed of two distinct grid middleware packages, say Globus (domain A) and OurGrid (domain B), specific Globus and OurGrid publishers should be installed in both domains – exactly in the location where data about the resources and the running applications is generated.

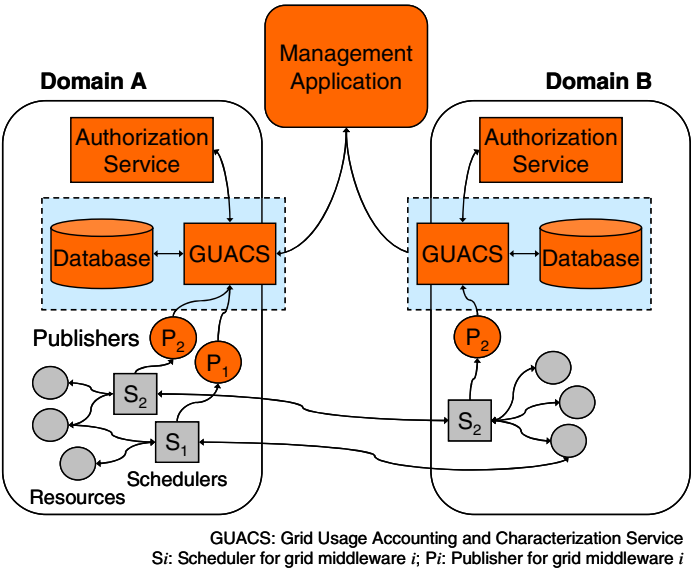


Fig. 1. Conceptual view of the architecture

The *Characterization Service* is a software entity, whose purpose is, among other, to receive data from the publishers and store them in a local, normalized database. By normalized database we mean that regardless of the grid middleware being monitored, the database always stores the same data (i.e. the information model is the same). To address potential scalability problems, the service can be instantiated in several locations (e.g. per institution, per department, etc.); important is to cover the whole grid infrastructure.

In addition to receiving and processing data sent by the publishers, the characterization service also serves a management application (or other management services), which may request for historical and/or real-time information about the resources and the applications running on the grid. Two types of requests are currently supported: *publish/subscribe* and *query/response*.

Whenever a request is received by the characterization service, it is first passed to an *Authorization Service*, which is responsible for granting or not the former permission to reply the request. Based on a role-based access control policy [5], it determines whether or not (and to what extent) a request originated from a certain institution can be replied.

The architecture is also composed of a *Management Application*, from where the administrator can subscribe for/be notified of relevant events generated by the whole grid infrastructure. He/she can also gather historical information from the

characterization service instances and plot several graphs in order to characterize grid usage. Note that, due to the normalized information model employed by the characterization service, all the data gathered – regardless of the grid system used in every institution – has the same format. Therefore, it is possible to draw a picture of the whole grid infrastructure usage in an integrated, uniform way. This *complete view* of the grid will may be compromised if the policy to distribute information employed by the institutions making part of the grid is too restrictive.

It is possible to have several instances of the management application running simultaneously (e.g. one per domain or department). Each of them will present a more or less detailed view of the grid infrastructure, depending on the permissions that the requester has in the characterization services.

4 Components of the Architecture and Implementation

In this section we describe the components of the architecture in more detail, focusing on design and implementation decisions. Recall that the architecture stands on the WSDM specification. Therefore, other associated standards such as WS-Notification [8] and WSRF (Web Services Resource Framework) [6] are also employed.

Table 1. Usage Record format

Field name	Description
Username	User's login name corresponding to user Id in /etc/passwd file.
ProjectName	Name/identifier of the project or charge group associated with this usage.
JobId	Identifier of the job.
Queue	Name of the queue from which the job was executed or submitted.
GridId	User's global unique Id. Distinguish Name in the user's X509 certificate.
FromHost	Name of the host from which the job was submitted.
Host	Name of the host on which the job ran.
StartTime	Date when the job started running in date time format (UTC time zone).
EndTime	Date when the job completed in date time format (UTC time zone).
Processors	Number of processors either used or requested that each center uses (for billing purpose).
NumNodes	Number of nodes used.
CpuTime	CPU time used, summed over all processes in the job.
WallTime	Wall clock time elapsed while the job was in the running state.
Memory	Maximum amount of virtual memory used by all concurrent processes in the job.
Disk	Disk storage used.
Network	Network used (withdrawals) or requested (reservations).
JobName	Job or Application name.
Status	Number representing completion status of the job.
Charge	Total charge of the job in system's allocation unit.

4.1 Information Model

As already mentioned in the previous section, we have adopted a uniform information model to be used by all instances of the characterization service. We use the UR (Usage Record) model [10], proposed by the Usage Record Working Group of the Global Grid Forum. This model is the result of an effort to define a common usage record based on those employed in current grid sites.

Table 1 illustrates the fields included in the Usage Record. As one can see, it merges information related to resource usage (e.g. Processors and NumNodes) and applications (e.g. JobName and Status).

4.2 Publishers

For every grid middleware there must exist a specific publisher, which is able to collect data about the resources and the running applications on that particular system. In our current implementation of Globus and OurGrid publishers, this data is gathered from the log files generated by both grid systems.

Whenever a new event is detected in the grid system (e.g. job started and job completed), the publisher executes a SETRESOURCEPROPERTIES request to the *Characterization Service* (to where it is virtually attached) updating a *resource property* defined within the service (e.g. JOBSTARTED and JOBCOMPLETED). The content of such a request is illustrated in figure 2a and b. Note that each *set* operation comprises the update of several data into the service (through the use of complex types). A simplified XML document is used to express the updates required by each possible SETRESOURCEPROPERTIES request. In addition, as it is the case of the examples presented, the data updated by a set request to a certain property (e.g. JOBCOMPLETED) is complementary to the data supplied by the set request to other property (e.g. JOBSTARTED).

```
<JobStarted>
  <Username> glauco_ludwig </Username>
  < ProjectName> bio_paua </ProjectName>
  <JobId> 1.1.1.node01.unisinos.br </JobId>
  <Queue> node01 </Queue>
  <GridId> glauco_ludwig@unisinos.br </GridId>
  <FromHost> node01.unisinos.br </FromHost>
  <Host> node09.unisinos.br </Host>
  <StartTime> 2005-09-13 17:24:50 </StartTime>
  <JobName> intracel_dynamics </JobName>
</JobStarted>
```

(a) JobStarted

```
<JobCompleted>
  <JobId> 1.1.1.node01.unisinos.br </JobId>
  <EndTime> 2005-09-13 17:30:22 </EndTime>
  <Processors> 2 </Processors>
  <NumNodes> 1 </NumNodes>
  <CpuTime> PT15S </CpuTime>
  <WallTime> PT45M48S </WallTime>
  <Memory> 1234 </Memory>
  <Disk> 560 </Disk>
  <Network> 1.000.000 </Network>
  <Charge> 300 </Charge>
</JobCompleted>
```

(b) JobCompleted

Fig. 2. Format of a SETRESOURCEPROPERTIES request

4.3 Characterization Service

The characterization service operates as an intermediary component between the publishers and the management application. The communication of both publishers and the management application with the service is carried out through a WSDM interface.

Publishers update data into the service through SETRESOURCEPROPERTIES requests, while a management application can (*i*) request for the value of a single property (GETRESOURCEPROPERTY), (*ii*) request for a filtered set of information – through an XPath query – of one or more resource properties (QUERY RESOURCEPROPERTIES), and (*iii*) subscribe for a resource property, as well as receive notifications. Figures 3 and 4 illustrate part of the SOAP envelope of both a subscription to and a notification of change in the resource property JOBFAILED.

```
<wsnt:Subscribe>
  <wsnt:ConsumerReference>
    <wsa:Address> http://www.unisinos.br:8080 </wsa:Address>
    <wsa:ReferenceProperties/>
  </wsnt:ConsumerReference>
  <wsnt:TopicExpressionDialect=”http://docs.oasis-open.org/wsn/2004/06/TopicExpression/Simple”>
    cs:JobFailed
  </wsnt:TopicExpression>
</wsnt:Subscribe>
```

Fig. 3. Format of a WSN-based subscription to JOBFAILED resource property

```
<wsn:Message>
  <wsrf:ResourcePropertyValueChangeNotification xmlns:wsrf=”http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-
   ResourceProperties-1.2-draft-01.xsd”>
    <wsrf:OldValue>
      <cs:JobFailed xmlns:wsrp=”http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.
       Xsd” xmlns=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:fs=”http://ws.apache.org/resource/gaems”>
        …
      </cs:JobFailed>
    </wsrf:OldValue>
    <wsrf:NewValue>
      <cs:JobFailed xmlns:wsrp=”http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.
       Xsd” xmlns=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:fs=”http://ws.apache.org/resource/gaems”>
        <fil:JobId> 2.1.1.node02.unisinos.br </fil:JobId>
        <fil:EndTime> 2005-09-13 15:30:24 </fil:EndTime>
        <fil:Processors> 4 </fil:Processors>
        <fil:NumNodes> 4 </fil:NumNodes>
        <fil:CPUTime> PT15S </fil:CPUTime>
        <fil:WallTime> PT45M48S </fil:WallTime>
        <fil:Memory> 5000 </fil:Memory>
        <fil:Disk> 850 </fil:Disk>
        <fil:Network> 10.000.000 </fil:Network>
        <fil:Charge> 0 </fil:Charge>
      </cs:JobFailed>
    </wsrf:NewValue>
  </wsrf:ResourcePropertyValueChangeNotification>
</wsn:Message>
```

Fig. 4. Format of a WSN-based notification for the JOBFAILED resource property

With reference to the resource properties available in the characterization service, currently six properties are offered: JOBSTARTED, JOBCOMPLETED, JOBABORTED, JOBFAILED, JOBCANCELED, and JOBHISTORY. These properties are detailed in table 2.

Note that the first fifth properties are expected to be subscribed for. Hence, when a job is completed, successfully or not, the management application may receive a real-time notification, which can be used either by the grid system itself (e.g. to reschedule a failed job) or the administrator. The JOBHISTORY property, on the other hand, stores multiple instances of the Usage Record, which can be retrieved by a management application through GETRESOURCEPROPERTY or QUERYRESOURCEPROPERTIES requests.

The characterization service has been implemented based on the Muse framework [1] from the Apache Software Foundation.

Table 2. Resource properties available in the characterization service

Resource property	Operations supported	Information available
JobStarted	Subscribe/publish	Username, ProjectName, JobId, Queue, GridId, FromHost, Host, StartTime, JobName
JobCompleted, JobAborted, JobFailed, JobCanceled	Subscribe/publish	JobId, EndTime, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge
JobHistory	GetResourceProperty, QueryResourceProperties	Username, ProjectName, JobId, Queue, GridId, FromHost, Host, StartTime, EndTime, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, JobName, Status, Charge[1]

[1] The information available matches those of the Usage Record. The JobHistory property stores multiple instances of the record.

4.4 Authorization Service

The authorization service is invoked by the characterization service whenever it receives a request from a management application. Depending on the identification of the requesting institution and the policies defined by the owner of the characterization service, three situations may occur: a response is not sent; a response with partial content is sent; or a complete response is sent.

The authorization model employed by the service to specify and enforce information distribution policies is the RBAC (Role-Based Access Control) [5]. RBAC was chosen because it is widely accepted and simplifies the management of policies. In this format, policies (e.g. permissions and restrictions) are associated to roles, and institutions are assigned to the proper roles.

Table 3 illustrates the organization proposed for the policy repository. Institutions may belong to one or more roles. Each role, on its turn, comprises the set of the Usage Record fields allowed to be distributed.

Table 3. Example of policies specified for a hypothetical institutional characterization service

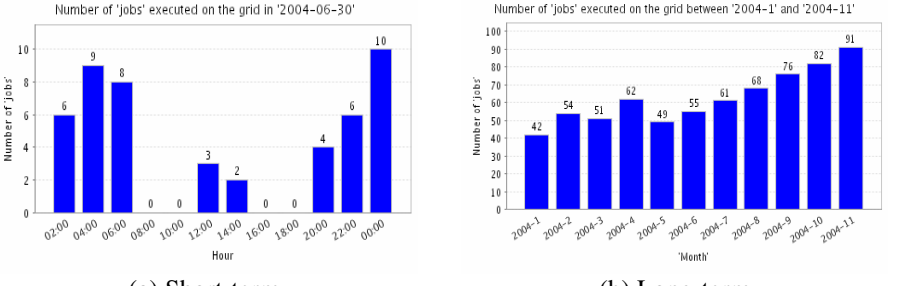
Host/institution	Role	Policies[1]
University A	University	JobStarted, JobCompleted, JobAborted, JobFailed, JobCanceled, JobHistory Username, ProjectName, JobId, Queue, GridId, FromHost, Host, StartTime, EndTime, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, JobName, Status, Charge
Company B	Enterprise	JobHistory JobId, GridId, FromHost, StartTime, EndTime, CpuTime, JobName, Status, Charge
Company C	Enterprise	JobHistory JobId, GridId, FromHost, StartTime, EndTime, CpuTime, JobName, Status, Charge
Other	Default	None

[1] Information allowed to be distributed.

4.5 Management Application

In the management application, the grid administrator can configure the characterization services it is going to interact with, as well as dynamically subscribe for/be notified of relevant events generated by them. In the case of the notifications received, they are presented in an event console. As for the historical information retrieved from one or more characterization service instances, several plots can be automatically generated, providing an integrated view of the usage of the grid computing infrastructure (see figure 5 for a simple example).

Our current prototype was implemented in Linux, using the Java programming language, the HSQLDB database, and the JFreeChart library (for the generation of plots). We are also working on the design and implementation of a release of the management application to run integrated to the HP Open View management platform.



(a) Short-term (b) Long-term

Fig. 5. Plot of job execution statistics

5 Experimental Evaluation

To prove concept and technical feasibility of the architecture, we have instantiated it in a real setup, composed by three administrative domains: A, B, and C. In two of them, A and B, both Globus and OurGrid middleware packages were deployed, while in domain C only OurGrid was used. One instance of the *Characterization Service* was installed and executed in each domain. In this environment, we were able to evaluate the impact of policy definition on the dissemination of grid usage data among the domains (explored in a paper being currently prepared by our research group). In addition, we were able to stress both the *publish/subscribe* mechanism (properties JOBSTARTED, JOBCOMPLETED, JOBABORTED, JOBFAILED, and JOBCANCELED) and the requests for *historical* data (property JOBHISTORY). Data retrieved by requesting the latter allowed the management application to draw plots offering an integrated view of the whole grid infrastructure.

A preliminary performance evaluation of the architecture has also been carried out, restricting the experiment to one *Publisher*, one instance of the *Characterization Service,* and the *Management Application*. Each of these components was executed in a different PC with 2 Pentium4 2.4 GHz processors, 1 GB of RAM memory, and GNU/Linux Red Hat 8.0 operating system, which were interconnected through a 100 Mbps switch.

To obtain statistically sound results, each experiment was repeated 400 times. The *end-to-end delay*, i.e. the time interval measured between the moment when an event is generated by the grid middleware and the moment when it is reported to the management application, was in average 245.62 ms. The standard deviation of the observed measurements was 3 ms. Although this a hardware-dependent result, it represents a good estimative of what one can expect in terms of the processing overhead imposed by the management plane. We consider this result acceptable, since the management application will be almost immediately informed about the occurrence of important events and will be able to react, in a timely manner, to interruptions in the execution of applications.

6 Conclusion and Future Work

The use of grid computing infrastructures is consolidated in academic environments, but in corporate environments their deployment is not as accelerated as originally expected. We attribute this to the lack of security and, specially, management mechanisms to provide a reliable, secure, and controllable grid computing environment. In this paper we proposed an architecture – based on a management standard highly conformant with the current web services orientation of grid technologies – to characterize and account, in a uniform and integrated way, usage of grid computing infrastructures composed of heterogeneous middleware packages.

The architecture allows the grid administrator, in addition to receiving real-time notifications about major events, to obtain long-term statistics about jobs executed, resources used, and so on. This information allows one to precisely characterize grid usage (e.g. in terms of top grid consumers, types of jobs executed, stations most used). Existing solutions are limited, since they only gather complementary

information such as CPU load and memory use of the grid machines. From a grid management perspective, the statistics the architecture is able to provide are important (*i*) to assess the volume of accesses to the grid infrastructure, the communications established, (*ii*) to draw a global grid usage profile, and (*iii*) to optimize and plan the capacity of the grid.

Another important aspect of the architecture to be highlighted is its ability to selectively distribute information taking into account the policies defined by every institution comprising the grid computing infrastructure. Although a complete view of the grid may not be offered if the policies are too restrictive, we believe it can be avoided through negotiation and agreements between the institutions.

A major contribution of this work relies on the usage of WSDM, which consists of a common approach for managing all components of a grid environment, including resources and services. As far as we are aware, this is one of the first research papers to propose (and report a real implementation of) a grid management service compliant with this recently standardized specification.

As future work we intend to better evaluate the architecture developed, stressing it under different number of (and simultaneous) publishers, subscriptions, and policies. Besides, we will extend the architecture with complementary services to support additional grid management functionality (fault, configuration, performance, and security), following a *plug-and-play* design.

Acknowledgments

This work has been developed in collaboration with HP Brazil R&D.

References

1. Apache Web Services – Muse Project Home Page (2006). http://ws.apache.org/muse/.
2. Baker, M. and Smith, G. (2003). GridRM: An Extensible Resource Monitoring System. IEEE International Conference on Cluster Computing, pp. 207-215.
3. Condor Project Home Page (2005). http://www.cs.wisc.edu/condor.
4. Dinda, P., Gross, T., Karrer, J. et al. (2001). The Architecture of the Remos System. IEEE International Symposium on High Performance Distributed Computing, pp. 383-394.
5. Ferraiolo, D. F., Sandhu, R., Gavrila, S. et al. (2001). Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security, v. 4, n. 3, pp. 224-274.
6. Foster, I., Czajkowski, K., Ferguson, D. E. et al. (2005). Modeling and Managing State in Distributed Systems: the Role of OGSI and WSRF. Proceedings of the IEEE, v. 93, issue 3, pp. 604-612.
7. Globus Toolkit Home Page (2005). http://www.globus.org.
8. Graham, S., Hull, D., and Murray, B. (2006). Web Services Base Notification 1.3 (WS-BaseNotification). OASIS Public Review Draft. http://www.oasis-open.org/committees/download.php/18546/wsn-ws_base_notification-1.3-spec-pr-03.doc.
9. Lee, D., Dongarra, J., and Ramakrishna, R. et al. (2003). VisPerf: Monitoring Tool for Grid Computing. International Conference on Computational Science, pp. 233-243.

10. Mach, R., Lepro-Metz, R., and Jackson, S. (2005). Usage Record – Format Recommendation. Global Grid Forum Usage Record Working Group. http://www.psc.edu/ ~lfm/ PSC/Grid/UR-WG/UR-Spec.v1.pdf.
11. Massie, M. L., Chun, B. N., and Culler, D. E. (2004). The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. Parallel Computing, v. 30, n. 7, pp. 817-840.
12. Newman, H. B., Legrand, I. C., Galvez, P. et al. (2003). MonALISA: A Distributed Monitoring Service Architecture. Computing in High Energy and Nuclear Physics.
13. OurGrid Project Home Page (2005). http://www.ourgrid.org.
14. Tierney, B., Aydt, R., Gunter, D. et al. (2002). A Grid Monitoring Architecture. Global Grid Forum Performance Working Group. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-3.pdf.
15. Vambenepe W. (2005). Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 1. OASIS Standard. http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf.
16. Vinoski, S. (2005). Web Service References. IEEE Internet Computing, v. 9, no. 3, pp. 90-93.
17. Wolski, R., Spring, N., and Hayes, J. (1999). The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. Journal of Future Generation Computing Systems, v. 15, n. 5-6, pp. 757-768.