Functionalities in Grid Computing with Active Services

R.B.Leite[1], F.S.G. de Oliveira[1], C.G.Ribeiro[1], J.C. de Oliveira[1]
B.Schulze[1], E.R.M.Madeira[2]
{fgomes, const, jauvane, schulze}@lncc.br, edmundo@ic.unicamp.br

1. LNCC, Av. Getúlio Vargas 333, 25651-070 Quitandinha, Petrópolis - RJ
2. IC / Unicamp, PoBox 6176, 13083-970, Campinas - SP

Abstract

In this paper we discuss architectural aspects of middleware for grid computing based on an infrastructure of distributed clusters and/or distributed services, an access portal for a demonstration project in progress, and also some security issues. We observe, in recent works, activities in the direction of open service architectures for grid services and for web services. We also see advantages in adding facilities offered by distributed object computing environments with their interoperable references and services. Open service architectures introduce the possibility of composing structures of nested services. Additionally we discuss some security issues for instance intrusion detection.

1. Introduction

The Middleware is a layer between the operating system and the applications that provides open distributed processing support, allowing applications development, usage and maintenance. The Middleware layer is a shell over the operating system, adding new functionalities to support the distribution of applications. The environment is open in the sense that not only the systems are open but the services are open as well. Exporters offer services and importers search for services, independently of the identity and location of the exporter. An open service environment is also characterized by the unlimited access in the sense the environment is always able to accept a new user. On the other hand, in this environment, an exporter is autonomous to decide how to perform the service. The Middleware deals with this dichotomy: openness and privacy.

This support offered by the Middleware matches with the requirements for grid computing, distribution, openness and privacy. Two Middleware approaches can be applied to grid environments, based on object orientation and on service containers.

The former approach supports infrastructure and application objects that use object-oriented mechanisms, like inheritance, polymorphism, encapsulation, and so on. Examples of infrastructure objects are transactional, concurrency and naming service. CORBA (Common Object Request Broker Architecture) [1] is an example of this kind of Middleware. The last approach supports containers with infrastructure and application objects coexisting together. The Web Services standardization [2] is an example of this kind of Middleware.

Both kinds of Middleware provide access and location transparencies. The access transparency masks differences of heterogeneous computer systems with different operating systems and the use of different languages. The location transparency hides the location of an interface (object).

CORBA specifies the IDL (Interface Definition Language) to define services in an object-oriented environment. Web Services use WSDL (Web Services Definition Language), that is based on the XML (Extensible Markup Language), to define the services, and may use SOAP (Simple Object Access Protocol) to transfer data over HTTP or other protocol. Web Services are strongly based on Internet protocols.

The GGF (Global Grid Forum) is developing the most important specification on Grid services, the OGSA (Open Grid Services Architecture) based on emerging Web Services standards. This architecture offers interfaces to discover services, create dynamic service instances, manage service life-time, notify events and manage the Grid environment.

Structure: Section 2 presents requirements in Middleware for Grid Computing. Section 3 discusses related works. Section 4 discusses Middleware with nested services, and an access portal: Web Services allowing internet access with simplified access list, no separate communication servers, simplified security management, limited bandwidth, and (mobile) agents in management, monitoring security and intrusion, collocating services, load-balancing, among others. Section 5 introduces statistical network formulation and intrusion detection. Section 6 mentions performance evaluation and Section 7 presents conclusions and future activities.

2. Middleware for Grid Computing

In this section we discuss some requirements in Middleware for Grid Computing. Besides the transparencies in access and location, Grid computing needs the following basic services from the Middleware:

- Naming;
- Creation of transient services;
- Service invocation;
- Service discovery;
- Multiple protocol bindings;
- Life-time management;
- Change management to allow different versions of the same service;
- Event notification;
- Instrumentation and Monitoring;
- Allocation management to allow task distribution;
- Security; and
- Concurrency control.

CORBA provides almost all these services through its Service Objects (infrastructure objects). For instance, there are the: Naming, Trader (offers service discovery), Property Management (can offer version support), event notification (can be used as base to develop a life-time management), Security and Concurrency. The current version of CORBA does not define support to Instrumentation, Monitoring and Allocation Management, but it is possible and relatively simple to develop these objects. On the other hand, CORBA provides few protocol bindings. Interactions between different CORBA environments are supported.

Web Services provides some of these services, like creation of transient services, naming (OGSA defines GSH – Grid Service Handle, and GSR – Grid Service Reference), service discovery (through the UDDI – Universal Description, Discovery and Integration, and the WS-Inspection), multiple protocol binding, life-time and change management, security and event notification. Like CORBA, Instrumentation, Monitoring and Allocation Management also are possible to be implemented. Currently, Web Services do not provide Concurrency Control, but it will be offered in the future, always following the container approach. The interactions between different Web Services are not completely defined yet.

New functionalities could be incorporated to the Grid Computing Middleware. For instance, task migration for load balance is a point to be developed, as well as the workflow support (called orchestration in Web Services). Also, communications between object of the same application during their executions should be supported, specially in strongly coupled applications.

Nowadays, the Middleware normally allocates automatically the computing resources to the execution of a task. However, advanced Grids can require more complex policies of resource allocation. In this case, the Middleware should provide to the grid administrator the functionality of defining and applying allocation policies. This Policy Management requires the development of new task schedulers.

3. Related Works

In this section we discuss some related works and the possibility to use nested application services. There are several Middlewares for Grid computing nowadays, following different approaches and focusing on some different aspects.

Classic grid implementations support scheduling with dispatch queues. The basic computational units are processes and the management of the resources' availability is obtained by monitoring the queues' throughput and number of waiting processes. The more frequent communication mechanism is MPI (message passing interface) [3] which imposes restrictions in inter-clusters communication, which should be possible for instance, with a directory service for registration and localization of separate process instances in different clusters in the grid.

Globus-3 [4] and Nimrod/G [5] are based on Web Services. The first one offers the main services described in Section 2, while the last extends management and scheduling functions and defines new Scheduler, Dispatcher and Job Wrapper. Legion [6] is based on the object technology and defines an IDL to allow the use of different programming languages. Legion also offers the main Middleware services and permits the use of object-oriented concepts, like inheritance. There are CORBA based implementation taking advantage of the functionalities of this technology, as for example [7], allowing the task migration for load balancing. Discover [8] is based on Web Applications and CORBA. The interaction between the client and the server is based on Web Applications, taking advantage of the simplicity of the Internet protocols. In the server side, the interactions between the several service objects (Name Server, Registry, and so on) is based on CORBA, taking advantage of its robustness.

Yet another aspect is scalability, for handling large amounts of events and specially when they are asynchronous, happening at random and eventually simultaneously. For ORBs there is an specification [9] where the infrastructure dynamically adapts (scales) to allow larger queuing of (simultaneous) incoming and pending outgoing requests.

Web Services as much as object-oriented approaches can be used to develop Middleware platforms for Grid computing. However, each approach takes advantage of different aspects. Web Services is based on Internet protocols, that are *de facto* standards.

New tools using these protocols are often developed. Web Services intrinsically use XML, that allows the service specification in a higher level. On the other hand, object-orientation based platforms permit the use of inheritance, encapsulation, polymorphism, and so on. The infrastructure objects are already defined and implemented, like the Transactional and Concurrency Objects. This kind of platform is more mature.

If we consider services as the main approach for having different kind of computing facilities, we may have a recursive structure of services being implemented by using other existing services, as represented in Figure 1. The Globus-3 is being developed following OGSA and will allow structured services.

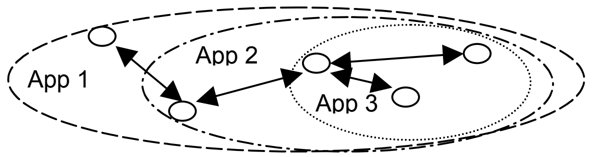

Figure 1. Nested applications based on services.

The presence of more and more web infrastructure leads to web services [10] of all sizes and colors and strongly dependent on interoperability and portability, which is present in CORBA as well. Looking into the direction of services we may consider an alternative grid structure different from the traditional one which aims at a flat structure where the backbone bandwidth should be identical to the clusters' intranet one [11].

4. Large Scale Workbenches

In this section, we have an infrastructure and Middleware allowing nested services, and an access portal to a demonstration project that we are currently working on.

There is the setup of a demonstration project for integrating (Brazilian national) high performance computing centers using a grid computing middleware. The project involves the development of an access portal with a submission service, securtiy services, migration tools, monitoring management, among others. The project will have to cope with shared and limited bandwidth in the internet backbone. The grid platforms used in this effort, up to now, are the GridEngine [12] and the Globus toolkit. The use of OGSA is quite appropriate for the reasons of limited bandwidth, use of web servers for internet (and user) access, no need for separate access list of the individual client machines, no separate communication servers (and port), and simplified centralized security management.

In a more realistic structure we expect a hierarchy of clusters as in Figure 2, with clusters (for the nodes) and the network backbone (for the edges). Consequently the structure is not flat and contiguous with the clusters intranet but rather an heterogeneous structure.

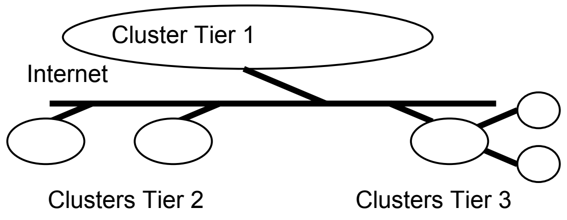

Figure 2. Cluster hierarchy with a network backbone.

In order to deal with different and limited bandwidths, we expect to have (mobile) agents [13] support, as represented in Figure 3, and benefit from collocated client and server [14] communication facilities. This aspect is associated with the relationship service in CORBA. Mobile agents can be used for: management [15], monitoring security conditions and flaws, intrusion detection [16], deployment of (Grid) Web Services (information and code), optimization of performance by collocating services, load-balancing, resource availability, among others.

The access portal and submission service are being implemented as a web service using Java servlets and XML for a Jakarta tomcat webserver. The submission service has a web interface for declaring the code to be submitted, passing parameters, negotiating the necessary resources, and dispatching the execution. It has to rely on security infrastructure for secure connection, authentication, and authorization. In Figure 4 there is a representation of the demonstration project (in progress) with the local grid and connections to be established to other HPC centers.

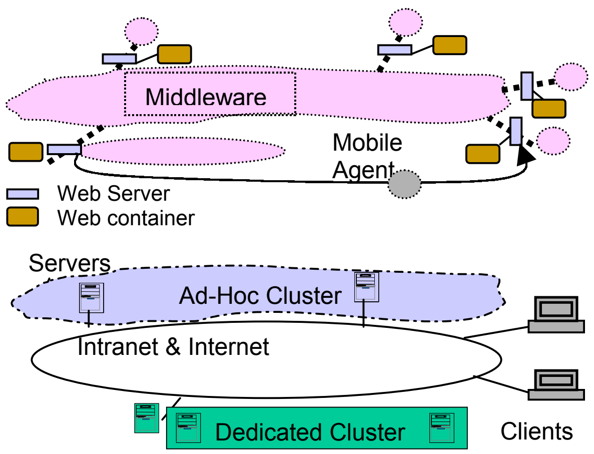

Figure 3. Upper part: Middleware supporting Web Services and (mobile) agents, Lower part: Infrastructure.

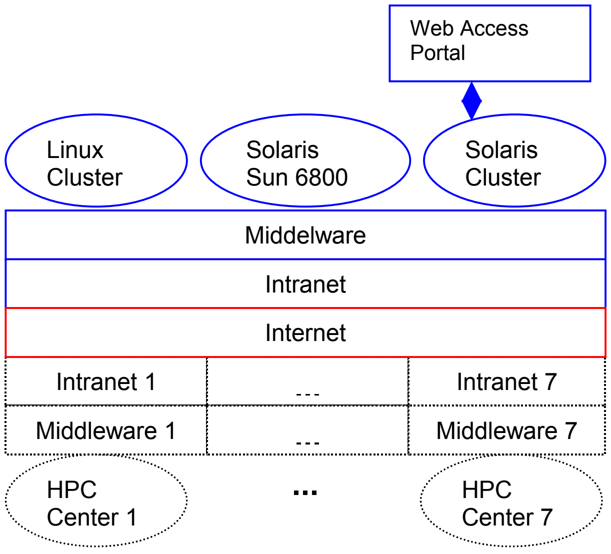

Figure 4. The demonstration project: the local grid (solid) and future connections to HPC centers (dashed).

5. Security Issues and Services – Distributed Denial of Service Attacks

In this Section we discuss the formulation of statistical network modeling in intrusion detection. Security in Grids is a fundamental matter. Grid implementations support security based on services that provide functionalities such as: secure connection protocols, public key infrastructure, and certification authorities. However, additional functionalities may be necessary based on real life grid experiences, as for instance intrusion detection, among others.

Security services in distributed systems constitute a new area in computer science, although there has been considerable research about this matter. Security services have seen a significant evolution, as well as their counter parts like worms, viruses and distributed denial of service attacks. Fundamental components in security architectures are firewall functionalities, based basically on a set of rules that tell the kernel what packets are to be let through the IP stack. Firewall vendors guarantee their products are not vulnerable to attacks, but in practice this is not true. As they incorporate all the complexity of the rules to describe the security policies in a virtual organization, they become more complex. A general concern about security services is that they must be tested against distributed denial of service attacks. The behavior of a distributed system or traffic network may be modeled by agent based systems, expert systems or statistically based systems.

We introduce basic concepts of statistical network modeling [17], where traffic modeling studies are related to collecting, measuring and analyzing traffic variables. The major point is to investigate the estimation of a set of interest variables. The estimation consists of the analysis of the data set generated by the observation of these variables, while the observations are measurements of network quantities. Basically a node is chosen in the network and some characteristics of the connections between this node and others connected to it need to be quantified and measured.

Let H be the number of host machines connected to a single one. The H hosts will be denoted by the set $X = (x_1, \ldots, x_H)$. Let $u_j = u(x_j)$ be a variable of interest at the j^{th} node. A variable of interest could be the transit package time between nodes, inter arrival times of connections to a server, etc. In fact we have a multivariable problem. Let S the total number of interest variables and $U = [u_1, \ldots, u_S]^T$ be a S-dimensional random variable. Then $y_{rj} = u_{rj} + \varepsilon_{rj}$ will be the r^{th} observation at the j^{th} node. For each node Y_j we will take r_j observations, i.e. the set $Y_j^i = [y_{1j}, \ldots, y_{r_j j}]^T$. We assume that no changes in the value we are estimating occur during the collection of the observations. The error terms ε_{ij} are random variables assumed to have zero mean and to be uncorrelated with variance σ^2, independent of the node. The covariance matrix is

$$K = E\left[(U - E[U])(U - E(U))^T\right] = [\kappa_{mn}] \quad 1 \le m \le n \le S$$

This set of nodes is called the experimental design of size S. The predictor will assign a weight to each node, and the prediction will be a weighted sum of the observations. Thus, we are considering a linear predictor in this work. The experimental design of size *n* lower than *S* denoted by ξ_n is defined to be

$$\xi_n = \{p_j, x_j\}^n, p_j = \frac{r_j}{N}; N = \sum_{j=1}^{N} r_j; \quad x_j \in X, n \le S$$

The quantities p_j denote the weight on the node or the proportion of the observations to be taken at node *j*. The values of x_j are the design points of the experiment, i.e. the nodes at which the observations are to be made. Note that $\sum_N p_j = 1$, so the set of p_j can be seen like the probability density function of the *j*th component of the random vector *U*. The estimators are obtained from the observation vector. Then

$$\hat{Y}(\xi_n) = \begin{bmatrix} \frac{1}{r_1}\sum_{i=1}^{r_1} y_{i1} \\ \vdots \\ \frac{1}{r_n}\sum_{i=1}^{r_n} y_{in} \end{bmatrix} = \begin{bmatrix} \hat{Y}_1 \\ \vdots \\ \hat{Y}_n \end{bmatrix}.$$

Each value of r_j corresponds to the sample size of each node defined in ξ_n. The estimator above gives the sample average for each node. This vector will be the prediction vector. The averaging could be thought as a method of "denoising" (such as $E[\varepsilon_j] = 0$), or reducing the variance of the estimate. Instead of averaging, one could consider a robust estimate of location, such as the median, but we will focus on averaging in this work. Another way to reduce the impact on the network would be to make our measurements during non-peak hours. However, if we aim at measuring delays, for example, this would defeat the initial intent. For some quantities (inter arrivals package time, for example), we have no choice but to make our measurements during peak hours. It is important to keep in mind that we are predicting the activity across the entire collection of nodes from measurements collected at a subset of nodes. As mentioned earlier, this is often necessary due to lack of resources or a desire to reduce the impact of the data collection on the network. We want to design our experiment – select the monitored nodes – and to estimate it as accurate as possible. To this end we need a way to measure the accuracy of our estimate. Given $\hat{U}$ an estimate of *U*, the matrix of expected squares residuals associated to the experiment ξ_n is

$$D(\xi_n, \hat{U}) = E\left[(\hat{U} - U)(\hat{U} - U)^T\right].$$

The quantity $D(\xi_n, \hat{U})$ gives us the accuracy of the estimator $\hat{U}$.

An intrusion detection service should have an early detection of Distributed Denial of Service Attacks (DDSA). We work with the Management Information Base (MIB) traffic variables collected in the hosts participating in the attack. Cabrera et al [18] uses this approach to analyze a database generated from a network operation monitoring of these variables. They use a simple Autoregressive Moving Average (ARMA) model of lag *A* to explain the relation between observations Y_j and variables U_j. Then

$$y_{ij} = \sum_{a=0}^{A} \left[\alpha_a y_{i-a,j} + \beta_a u_{i-a,j} + \gamma_a \varepsilon_{i-a,j}\right].$$

The determination of constants $\alpha_a's, \beta_a's, \gamma_a's$ and the model lag A involves using statistical tools for testing its significant information about the variable y_j.

A DDSA is characterized by pieces in the database of anomalous behavior, so the variables don't obey the model above and the values of $D(\xi_n, \hat{U})$ are exceptionally high. The parameters of the model are fitted using a database sample without attack interference.

The Network Management System program measures a total of 91 MIB variables ($U = [u_1, \ldots, u_{91}]^T$), for intervals of 2 hours at a sample rate of 5 seconds. The topology suggested to this experiment is a master system connected to 5 slaves nodes (for instance Web servers) represented by the set $X = (x_1, \ldots, x_5)$. These slave nodes are connected to the target node (a single node). An attack is triggered from the master node. It installs the attack tool software in the slave nodes producing a denial of service into the target (single) node. The U variables are observed in the connections between the slaves with the target node.

Further work is needed to validate this methodology under more realistic traffic loads. A realistic network condition involves multiple domains environment. This work is in progress at this stage.

6. Evaluation

Here we discuss shortly first performance evaluations. For evaluating the behavior of different technologies and implementations involved, we start with a basic benchmark that compares the execution of a program in a collocated run on a single node, with a distributed run over specified computing nodes. We are currently working on the comparison of two study cases for a parallel program based on MPI and for a distributed program based on ORBs. The examples are different, i.e., they are not the same computation, but both of them are either executed collocated on a single computing node, or spread over several computing nodes. This should provide some input to establish comparison figures and identify critical points, and bottlenecks. Further effort is needed in selecting a set of benchmarks.

7. Conclusion

We point out that a (heterogeneous) grid computing infrastructure altogether with a Middleware based on open (grid and web) service architectures and the addition of mobile agents support in the context of web infrastructure allows for instance the possibility of composing applications based on nested services and other functionalities as monitoring agents (management, security, intrusion detection, among others), deployment of new services, migration of services for load-balancing, for collocation and so on. This also contributes to the building of an access portal and dealing with some security issues as intrusion detection.

In this work we considered computing nodes as part of a dedicated cluster with closely coupled nodes and geographically distributed clusters, meanwhile another kind of clustering can be observed in what we call ad-hoc clusters with volunteered loosely coupled computing nodes. Other aspects in this 2nd kind of cluster are for instance security, communication and persistence.

Acknowledgments: This work is supported by Brazilian funding from CNPq and PCI/MCT.

References

1. OMG Common Request Broker: Architecture and Specification, Version 3.0.2, 2002 www.omg.org
2. W3C Architecture Domain, "Web Services", www.w3.org/2002/ws
3. The Message Passing Interface Standard, www-unix.mcs.anl.gov/mpi
4. I.Foster, et al. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed System Integration", 2002 www.globus.org/research/papers/ogsa.pdf
5. R.Buyya, D.Abramson, and J.Giddy, "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", *Fourth IEEE Int. Conference on High Performance* Computing *in the Asia-Pacific Region*, pp. 283-289, 2000
6. A.S.Grimshaw, A.Ferrari, F.C.Knabe and M.Humphrey, "Wide-Area Computing: Resource Sharing on a Large Scale", *IEEE Computer*, 32 (5), pp. 29-37, 1999
7. S.Vanhastel, F.D.Turck and P.Demeester, "Design of a Generic Platform for Efficient and Scalable Cluster Computing based on Middleware Technology", *1st IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 40-47, 2001
8. V.Mann and M.Parashar, "Middleware Support for Global Access to Integrated Computational Collaboratories", *10th IEEE Int. Symposium on High Performance Distributed Computing*, pp. 35-46, 2001
9. A.Gokhale and D.C.Schmidt, "Measuring and Optimizing CORBA Latency and Scalability Over High-speed Networks", *IEEE Transactions on Computing*, Vol.47 No.4 April 1998
10. S.Kleijnen and S.Raju, "An Open Web Services Architecture", *ACM Queue Tomorrow's Computing Today*, February 2003, www.acmqueue.org/issue
11. K.Schmaranz, "On Second Generation Distributed Component Systems", *Journal of Universal Computer Science - JUCS,* Vol.8 No.1 (2002) 97-116
12. GridEngine, Sun Microsystems Inc., gridengine.sunsource.net
13. B.Schulze and E.R.M.Madeira, "Contracting and Moving Agents in Distributed Applications Based on a Service-Oriented Architecture", *Mobile Agents LNCS 1219* pp. 74-85, Springer, April 1997
14. B.Schulze and E.R.M.Madeira, "Migration Transparency in Agent Systems", *IEICE Trans. on Communications, IEICE/IEEE Joint Special Issue on Autonomous Decentralized Systems*, Vol. E83-B No.5 (2000/5) 942-950
15. B.Schulze et al, "Momenta: Service Management using Mobile Agents in a CORBA Environment", *Journal of Network and Systems Management* Vol.9 No.2 June 2001, pp. 203-222
16. W.A.Jansen, "Intrusion Detection with Mobile Agents", *Computer Communications* No.25 (2002) pp. 1392-1401
17. D.J.Marchette, Computer Intrusion Detection and Network Monitoring: A Statistical Approach, Springer-Verlag 2001
18. J.B.Cabrera et al, "Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables – A Feasibility Study", *Proceedings 7th IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, WA – May 2001
19. W.Dillon and M.Goldstein, "Multivariate Analysis Methods and Applications", J. Wiley & Sons 1984
20. T.Amemiya, "Multivariate Regression and Simultaneous Equation Models when the Dependent Variables are Truncated Normal", *Econometrica*, Vol. 42 No. 6 November 1974