Enabling a Network Simulation Application on Grid Infrastructure

A. Menychtas[1], D. Apostolopoulos[1], D. Kyriazis[1], K. Christodoulopoulos[2], H. Avramopoulos[1] and T. Varvarigou[1]

[1]Department of Electrical and Computer Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str, 15773 Athens, Greece
[2]Computer Engineering and Informatics Dept. and Research Academic Computer Technology Institute, University of Patras, Rio, Greece
E-mail: a_menychtas@telecom.ntua.gr, apostold@mail.ntua.gr, dkyr@telecom.ntua.gr, kcristodou@ceid.upatras.gr, hav@mail.ntua.gr, dora@telecom.ntua.gr

Abstract

The advent of heterogeneous and distributed environments, such as Grid environments, made feasible the solution of computational intensive problems in a reliable and cost-effective way. We demonstrate the operation of a mechanism and evaluate its performance and effectiveness for the Network Simulation application in a Grid environment. The network simulator (ns-2), that was implemented, is used for simulation of high speed optical networks. In order to be able to meet the requirements of commercial business processes on the Grid, and especially network simulation tasks, we present a toolkit for centralized job submission and access. The approach has been implemented within the framework of the GRIA IST project that was originally designed for industrial based applications.

Keywords: Grid Computing, Network Simulation, Optical Networks, Centralized Access, Grid Portals

1. Introduction

Grid computing is increasingly being viewed as the next phase of distributed computing. Built on pervasive Internet standards, Grid computing enables organizations to share computing and information resources across department and organizational boundaries in a secure, highly efficient manner. Grids support the sharing, interconnection and use of diverse resources in dynamic computing systems that can be sufficiently integrated to deliver computational power to applications that need it in a transparent way [Foster I. (2001)], [Leinberger W. (1999)].

The Grid provides a cost-effective way for occasional users to access high-end systems and software. At the same time, those that need such systems enough to

invest in equipment and expertise can sell any spare capacity over the Grid to other users. These benefits make the Grid a natural forum for e-commerce in computing capabilities, and many observers believe these "B2B" applications will drive the development of the Grid from a research tool to become the next generation Web infrastructure. One of the Grid middleware that enables such actions is the outcome of the GRIA IST project [GRIA Project], [Taylor Steve (2004)]. The performed work allows the usage of this middleware by engineers in order to submit simulations to the Network Simulator ns-2 platform [Network Simulator].

In this case, the Grid acts as an organization that connects and provides directly available heterogeneous resources using Internet technologies, across widely variant devices enabling the simulation process to be completed automatically in an electronic way. It makes use of business models, processes and semantics to allow resource owners and users to discover each other and negotiate terms for access to high-value resources, by implementing an overall business process to find, procure and utilize resources capable of carrying out high-value, expert-assisted computations.

Enabling applications on the Grid is a challenging research topic as proved from the results of the GRIA IST project, in which the GRIA middleware was implemented. In brief, the overall procedure is being initiated from the engineer while the simulation process occurs in one or more Grid resources. The above description proves that the Network Simulator ns-2 meets the definition standards of a fully automated Grid application as it eliminates the manual process - the engineer's responsibilities are limited to the uploading of the source files and the downloading of the final outcome. Furthermore, the manual process is being replaced with an automated process of allocating the Grid resources, completing the tender and the accounting procedures and executing the submitted job. These procedures are described further on in our paper.

Moreover, the main subject of our study concerns a toolkit especially designed and implemented for any Grid application. The toolkit includes capabilities like centralized access and usage statistics generation.

2. Toolkit Overview

The aforementioned toolkit provides a Grid application solution for the network simulation scenario as well as added functionality like system control and user management, that are being described in detail further on.

2.1. Toolkit Architecture

In general, the toolkit can be seen as an application inside a servlet container (currently implemented using Apache Tomcat [Apache Tomcat]) combining four

modules. These modules are not purely separated but are distinguished based on their role in the overall architecture of the system.

The first one is the Client (a series of java [Java Sun] classes) that implements the API. This part is where the overall Grid layer is hidden. All actions like security encryptions, allocation protocols and tender mechanisms have been implemented and are compatible with the underlying layer of the supplier.

The second module, which is also a collection of java classes, implements a module concerning user accounting and statistics. This module is used for the logging of all user actions and storing the job submissions and allocations that each user has performed.

The third and fourth modules are sets of JSP pages [JavaServer Pages] that provide the user an administrative interface correspondingly. These pages are designed in order to maximize the usability of the toolkit from the end-user side.

Regarding the network simulation software (NS), it is located on the supplier side under the local resource broker. In order to proceed with the implementation of the broker, we have adopted the OpenMosix toolkit [OpenMosix].

On the following Figure (Figure 1) we present the aforementioned modules along with their role and the interactions they have with other parts of the system. The administrator registers the suppliers while all users are able to request allocations and submit jobs to them.

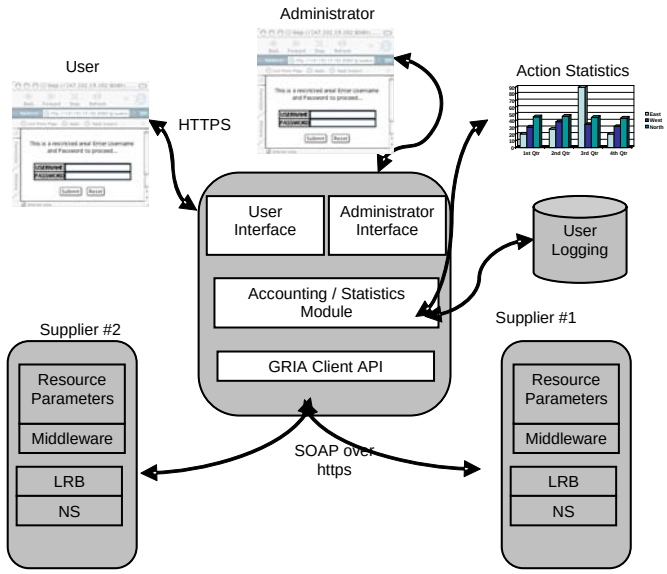

Figure 1. *Toolkit Architecture*

2.2. Implementation

The scheme of this architecture is designed following the specifications under the Web Services Resource Framework (WSRF) [OASIS Web Services Resource Framework] and therefore it is a federation of Web Services managing stateful resources. In general, the toolkit provides a package of four Web Services that provide the capability to the service provider to access shared remote computation and data storage, subject to a well-defined business process.

In detail and firstly, the Account Service supports the creation and management of accounts. Secondly, the Resource Allocation Service allows remote users to request and be granted (or denied) allocations of computation and data storage capacity at the service provider site. The service incorporates a resource capacity model, which is used to determine whether capacity is available to meet a requested allocation. Moreover, the Data Storage Service allows remote users to upload and download data files to the service provider. The Data Service also supports delegation of access rights (for read or read-write access) to other users or service providers. Finally, the Job Execution Service allows remote users to start, monitor or kill computational jobs, executed by the service provider. The Job Service can be configured to support multiple applications (mapped to different web service endpoints), fetching input from and writing output to data storage services.

The above mentioned Services are designed to run on a Tomcat/AXIS [Apache Axis] platform, using the WS-Security [OASIS Web Services Security] and Process-Based Access Control (PBAC) features supplied by the GridServIT package, which is an infrastructure toolkit enabling easy deployment of applications on to a Grid or similar service-orientated architecture. It is based on Web Services technology (building on Axis) and provides access to context information for a request at both detailed and shortcut levels. GridServIT also adds a number of desirable security features such as Secure Communication using WS-Security and X.509 PKI [Public-Key Infrastructure], Authentication using the WS-Security SOAP header [SOAP Header element] and Authorization using Process based access control.

A client-side toolkit and Java API is available and can be used to simplify the programming of applications that use Grid Services. The Client API means that client side applications can be easily written and managed.

2.3. Core Functionality

The main reason for implementing the toolkit is that a simple command line client causes many problems when used by business end-users, usually not computer professionals. At the same time, the toolkit provides homogeneous working conditions to all individual users, which simplifies administration and problem solving. Furthermore, all user actions can be monitored and supervised and thus avoid

unwanted usage and charges. Last but not least, the extraction of statistics about system utilization is an extra feature that provides valuable information about the enterprise needs and their employees actions. The basic capabilities of the architectural components that were named above are being described more analytical in the following paragraphs of this section.

In particular, one of the most interesting issues of any Grid middleware is the Job Submission Procedure. The toolkit provides a specific interface in order to assure and maximize the efficiency and usability of the Grid middleware. The Job Submission Procedure steps are depicted in figure 2 and described in detail below.

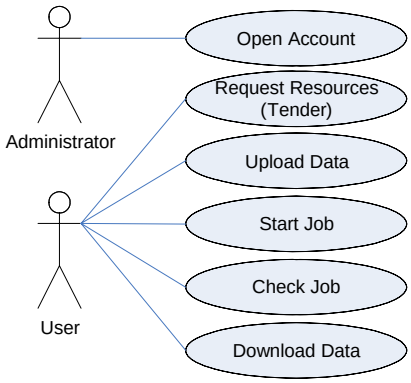

***Figure 2**. Job Submission Procedure*

Tender is the first step of the Job Submission Procedure. The job requirements are sent to the suppliers and the selection of the most suitable one for the specific job is being completed. This process consists of two steps. Firstly, the e-client user fills in a form with the requirement parameters of the job such as the start and end date of the allocation and the size of the data. Based on these parameters, the toolkit interacts with all the "known" suppliers (suppliers that the administrator has open accounts) and returns their results (price for the specific job and other parameters similar to the parameters mentioned before). Moreover, the toolkit includes a simple scheduler [Jackson L. (2002)], which examines the results of the tender process and sets as default the most appropriate. Thus the user can accept the suggestion or choose another one. With the completion of the tender process, the user has an allocation (-conversation) in a supplier and he is able to use the supplier resources.

Upload Data is an action that has to be performed after the tender process has been completed and an allocation conversation has been opened. Using the toolkit, the user uploads the input file(s) of the job and selects in which allocation the data will be uploaded. The data in the suppliers is represented as a data conversation.

Setting the Job and defining the execution parameters is the next step in the Job Submission Procedure. When all the data have been uploaded (input data may also be

output data from previous job executions) and the allocation for the job execution has been defined, the job execution can be started.

During the job submission procedure, the users can check the Job Status. Users can check the status of the job(s) sent for execution by browsing a table - available via the toolkit - with all the active jobs (-conversations) and the status of them. After a successful job execution (the status of the job would be finished / completed) the users can download the output data of the submitted job or use them as input for a new job submission.

Finally, the toolkit allows the users to check the active conversations providing a list with all the conversations and their description. These conversations can be accounts, allocations, data or job(s). Deletion of some conversations is also visible (for a specific group of users) except if there is an allocation with active job or data.

2.4. Added Functionality

In order to assure the efficiency and the reliability of the e-client, the toolkit also includes an administration section. The administrators can perform several actions divided into three major categories as described below.

In the User Management section, the administrators can add new users, change their attributes or their access level, and view the details of the accounts. The user access level gets specific values according to the users' roles like Administrator, Read-Write User, Read User or No Access. Moreover, the administrator can view a list of the users, change user details or change the access level of a user. Users cannot be deleted but they can only be restricted of any access (No Access level) in order to keep the user instance for logging purposes in a database.

The System Control section allows the administrators to add new accounts in the suppliers, check their status and balance and delete them. The new account process contains information relative to the supplier. Each account is also described from a status parameter. When a new account is opened, its status is pending-credit-checks. The users can tender using this account only if the supplier approves it and the status changes to open.

Furthermore, the administrators can check the status by browsing a list with the active accounts that the e-client uses, their description and their status. The administrators can either close these accounts (the status changes to account-usage-finished and the users cannot tender to them any more) or delete the accounts from the e-client system.

Finally, the administrators of the toolkit can check the balance of the accounts by browsing a table with all the accounts, the events that charge them and the balance of each one. Additional information for each account is provided containing all the allocation conversations that the users opened and confirmed in the suppliers, or the payments. Moreover in this page details about the events, like the date and the cost of

each one, are being provided.

The toolkit allows the presentation of Statistics for several time intervals, actions or users. The statistics can be customized depending on the administrator's preferences (ex. per user, per action, for a specific time period or any combination of these). The statistics section provides the administrator with monitoring information concerning the user's actions and how they interact using the toolkit. Also, using the statistics information there can be determined which users submit jobs, which user accounts are inactive and thus change the user rights.

2.5. Local resource Broker

When the submitted job reaches the selected supplier a Local Resource Broker is responsible for the distribution of the workload to all the available nodes of the specific supplier. In this implementation the LRB that was used is the OpenMosix cluster software.

OpenMosix is a Linux kernel [Linux Kernel] extension for single-system image (SSI) clustering [Single-system image]. This kernel extension turns a network of ordinary computers into a supercomputer for Linux applications. Once you have installed OpenMosix, the nodes in the cluster start talking to one another and the cluster adapts itself to the workload. Processes originating from any one node, if that node is too busy compared to others, can migrate to any other node. OpenMosix continuously attempts to optimize the resource allocation. OpenMosix achieves this with a kernel patch for Linux, creating a reliable, fast and cost-efficient SSI clustering platform that is linearly scalable and adaptive. With OpenMosix' Auto Discovery, a new node can be added while the cluster is running and the cluster will automatically begin to use the new resources.

There is no need to program applications specifically for OpenMosix. Since all OpenMosix extensions are inside the kernel, every Linux application automatically and transparently benefits from the distributed computing concept of OpenMosix. The cluster behaves much as does a Symmetric Multi-Processor (SMP) [Single-system image], but this solution scales to well over a thousand nodes which can themselves be SMPs.

The OpenMosix Community is very active, contributing add-on applications and sharing helpful information with all users. The OpenMosix Add-Ons and Community page lists these shared applications that is all licensed under GPL [GNU General Public License].

3. Network Simulation

The Network Simulator ns-2 is a discrete event based simulator targeted at networking research. It provides substantial support for simulation of TCP, routing,

and multicast protocols over wired and wireless networks. Ns-2 is freely distributed, open source software. Therefore, it gives the ability to reuse and extend existing components and its rich infrastructure for accommodating new protocols and architectures. The increase use of ns-2 by network researchers and its extensible design influenced the choice of using ns as the base platform for our simulation.

The design of ns-2 is such that simulation of very large networks is difficult, if not impossible, due to excessive memory and CPU time requirements. The PADS research group at Georgia Tech has developed extensions and enhancements to ns-2 (called PDNS - Parallel/Distributed ns) to allow a network simulation to be run in a parallel and distributed fashion, on a network of workstations [PDNS]. By distributing the network model on several machines, the memory requirements on any single system can be substantially smaller than the memory used in a single-computer simulation. The overall execution time of the simulation is at least as fast at the original single-computer, and can be several times faster, while it can also support proportionally larger network models by distributing the model on multiple systems. In order to do so, PADS group has modified the ns event processing infrastructure, and defined extensions to the TCL script syntax providing a way to describe distributed simulation executions. However, extensions that were written for ns-2 are not always compatible with PDNS and the additional complexity that the user has to deal with (manually divide the workload on the cluster of computers) is not always comparable to the gains of the parallel execution.

To evaluate the applicability of the ns-2 simulation platform in the presented Grid toolkit we have experimented with Optical Burst Switching (OBS) simulations [OWns]. Optical Burst Switching is a concept which lies between grain optical circuit switching and fine grain optical packet switching [Qiao C. (1999)]. Optical burst switching improves the utilization of the wavelengths by aggregating the traffic at the ingress of the network according to a particular parameter (commonly the destination and secondary the class of service - CoS). From the aggregation of packets, a burst is created and this is the granularity that is handled in OBS. The specific implementation is an open source ns-2 extension that enables full network OBS simulations with typical traffic sources such as CBR, exponential and Pareto over TCP or UDP protocols and the extraction of various statistics.

The key characteristic of the experiments that we wanted to perform is that they require the execution of ns-2 many times with different but simple data inputs (in particular, the network topology as well as various traffic parameters). What is usually done in this kind of experiments is change one traffic parameter, execute the simulation, obtain the results and iterate the whole process. The output is stored in a file that we can later process in order to evaluate the effect of the examined parameter on the performance. The iterative process that was just described can be considered as a batch process: instances of the same program need to be run on one processor with different inputs, while no communication is required between them.

From the aforementioned characteristics we can elicit that presented toolkit for centralized access and task submission to a cluster is an ideal solution for the simulation experiments that we wanted to perform. The parallel ns (PDNS) addresses a different kind of problem (experiments with large network models) which is inappropriate for our case. The simulations that were actually conducted using the toolkit proved the advantages of the proposed architecture when compared to "single-pc" model. Since the submitted to the cluster tasks had the form of batch processes that were CPU intensive, the execution time was vastly improved, converging to the optimum parallel execution performance (s-pc-time/N, where N is the number of the cluster's Working Nodes and s-pc-time is the execution time on a single pc).

The job submission procedure for each simulation was simplified through the portal functionality. The user didn't have to deal with design and execution parameters (as in the case of PDNS) which were handled effectively and transparently by the toolkit, and was able to monitor the simulations and the output in real time. Finally, using the framework the simulation resources can be accessed by more (than one) users, a promising attribute for the efficient utilization of the available resources.

4. Conclusions and Future Work

In conclusion, in this paper we presented a toolkit for automating the procedure of submitting network simulation jobs to a cluster that improves the Grid Middleware in terms of time saving for operations allowing each user to be only a few clicks away from each job submission and thus enabling centralized access to Network Simulation Applications on the Grid. Furthermore, it provides a centralized management system for all enterprise users, a rapid and easy to install application independent from the number of local users as well as statistics and accounting.

Additionally, the efficiency of a Job scheduler can be highly optimized when the criterion of execution time is estimated a priori which poses a research potential for our group. Moreover, the toolkit can be enhanced with an efficient fault tolerant mechanism [Litke Antonios (2005)] based on replication scheme in order to maximize the reliability of e-business application usage over a Grid infrastructure.

Finally, our future plans on this field of interest include the migration of the toolkit from the Apache Tomcat servlet container to the GridSphere [Novotny J. (2003)], [GridSphere] portal framework in order to take advantage of the useful tools, services, and core Grid portlets that provides, in addition to the pluggable access to Grid services using the Portlet Services concept. Furthermore, it is open source similar to the current infrastructure allowing the implementation of an abstract and generic Grid Portlets model.

5. References

Apache Axis implementation: http://ws.apache.org/axis

Apache Tomcat Servlet Container: http://tomcat.apache.org

Foster I., Kesselman C., Tuecke S. (2001), "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal Supercomputer Applications, Vol. 15, No. 3.

GNU General Public License: http://www.gnu.org/copyleft/gpl.html

GRIA Project: http://www.gria.org

GridSphere project: http://www.gridsphere.org

Jackson L. E. and Rouskas G. N. (2002), "Deterministic Preemptive Scheduling of Real Time Tasks", IEEE Computer, vol. 35, no. 5, pp. 72-79

Java Sun: http://java.sun.com

JavaServer Pages (JSP) technology: http://java.sun.com/products/jsp

Leinberger W., Kumar V. (1999), "Information Power Grid: The new frontier in parallel computing," IEEE Concur., Vol. 7, No. 4, pp. 75-84, October-December.

Linux Kernel: http://www.kernel.org

Litke Antonios, Tserpes Konstantinos, Dolkas Konstantinos, Varvarigou Theodora (2005), "A Task Replication and Fair Resource Management Scheme for Fault Tolerant Grids", European Grid Conference

Network Simulator: http://www.isi.edu/nsnam/ns

Novotny J., Russell M., Wehrens O. (2003), "GridSphere: A Portal Framework For Building Collaborations". Proceedings of the 1st International Workshop on Middleware for Grid Computing. Rio de Janeiro Brazil

OASIS Web Services Resource Framework (WSRF): http://www.oasis-open.org/committees/wsrf

OASIS Web Services Security: http://www.oasis-open.org/committees/wss

OpenMosix cluster software: http://openmosix.sourceforge.net

OWns - Optical WDM Network Simulator: http://dawn.cs.umbc.edu/owns

PDNS - Parallel/Distributed NS: http://www.cc.gatech.edu/computing/compass/pdns

Public-Key Infrastructure (X.509): http://www.ietf.org/html.charters/pkix-charter.html

Qiao C. and Yoo M. (1999), "Optical burst switching (obs)-a new paradigm for an optical internet," Journal High Speed Networks, vol. 8, pp. 69–84

Single-system image clustering: http://en.wikipedia.org/wiki/Single-system_image

SOAP Header element: www.w3schools.com/soap/soap_header.asp

Symmetric Multi-Processing: http://en.wikipedia.org/wiki/Symmetric_multiprocessing

Taylor Steve, Surridge Mike, Marvin Darren (2004) "Grid Resources for Industrial Applications", ICWS'04