A Grid Application Development Platform for WebCom-G*

John P. Morrison, Sunil John, David A. Power, Neil Cafferkey and Adarsh Patil
Centre for Unified Computing,
Dept. Computer Science,
National University of Ireland,
Cork,
Ireland.
{j.morrison, s.john, d.power, n.cafferkey, adarsh}@cs.ucc.ie
http://www.cuc.ucc.ie

Abstract

Grid Computing is becoming more popular. The traditional role of the Internet as being an information repository is evolving to become a resource repository. People using the Internet will want to do more than just look for information, they will want to exploit the resources available. Grid Computing provides platforms to facilitate such requirements. Various Grid middlewares have been developed to offer access to vast resources ranging from computing power to application functionality to specialised physical resources.

This paper details the programming model used for heterogeneous environments exploited by WebCom-G. In addition to describing the tools and methodologies used for this computing environment, a brief outline of WebCom-G (WebCom Grid) and some of its capabilities is given.

1 Introduction

Grid Computing is evolving to mean the sharing of geographically distributed resources. The complexity of various Grid systems is increasing rapidly. This rate of increase can be seen across all areas of Grid Computing, from raw processing power to networking infrastructure to new mechanisms for exploiting these resources, such as web services. However, Grid is unpredictable, since required resources may not always be available. Resource availability, machine heterogeneity, software heterogeneity and the highly dynamic environment in which Grid Computing resides all contribute to this unpredictability.

There are a number of Grid middlewares readily available such as Globus [5], Legion[14], Alchemi[7], GridBus[3] and Simgrid[4]. To successfully utilise such systems, developers are required to know system details ranging from resource configuration to service availability to hardware configuration. When creating Grid applications, proper management of the underlying resources has to be adopted. Resource availability, Fault Tolerance, Load Balancing, Scheduling all have to be taken into account. These requirements significantly increase the burden of responsibility on Grid application developers.

To become widely accepted to those who are not computing experts, Grid Computing must become easy to use. In effect the use of Grid technologies must become hidden from the end user. Various Grid middlewares give access to resources at different abstraction levels. These resources constitute significant investments in time and effort by experienced experts and provide a valuable foundation for the construction of Grid applications. A challenge is to

*This work is partly funded by Science Foundation Ireland under the WebCom-G project and the Higher Education Authority under the CosmoGrid project.

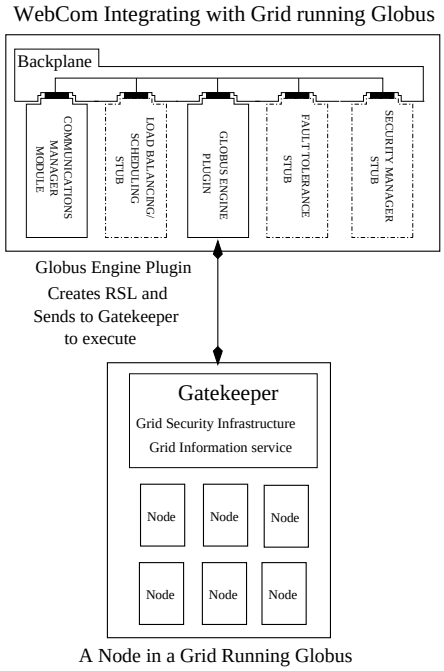

Figure 1: *The WebCom-G architecture containing modules for Scheduling, Load Balancing, Fault Tolerance, Security, Communication and Computation Engine. Each module plugs into a Backplane.*

leverage these resources in a manner consistent with hiding the detail from the uninitiated user community.

As Grid Computing becomes more established, emphasis is being placed on the creation of complete problem solving environments (PSEs) and the provision of different languages and APIs. These tools have to be effective in order to cope with dynamically changing heterogeneous environments.

This paper describes an application development framework based on the Condensed Graph model of Computing. A methodology to translate high level languages and proprietary specification languages to Condensed Graphs and extension of these methodologies for creation of other Condensed Graphs is discussed. A mechanism for creating Condensed Graphs from traditional applications is presented in Section 4. The WebCom-G PSE, WebCom-G$_{IDE}$ presented in Section 5, can be used to manually optimise these graphs. It can also be used to create new applications from previously created graphs thus facilitating the exploitation of the different resources available.

Firstly, Section 2 briefly outlines the main components of WebCom-G and Section 3 describes the job submission mechanism supported by WebCom-G. Finally, conclusions and future work is presented in Section 6.

2 WebCom-G - a Grid Middleware

WebCom-G[9, 12, 13] is a fledgling Grid Operating System. The WebCom-G project is a Grid enabled version of the WebCom[11] metacomputer. It separates the application from the execution platform. This is achieved by providing a multi-layer system implementation. Each layer has particular responsibilities in achieving task execution. These range from an application Execution Engine, to layers that carry out Scheduling, Load Balancing, Fault Tolerance and Security. Each layer in this architecture is implemented as a distinct module within WebCom-G. Modules are provided for Scheduling, Load Balancing, Fault Tolerance, Security, Communication and Computation Engine. These modules form *plug-ins* to a core module called the *Backplane*. The Backplane is responsible for bootstrapping particular instances of WebCom-G and also inter module communication. Different Computation Engine modules are available for different middlewares including COM, DCOM, .NET, Corba, EJB, and Globus. Figure 1 illustrates a particular configuration of WebCom-G that with a Globus Execution Engine plugin installed. This will execute Globus tasks presented to WebCom-G as part of a Condensed Graph application.

Applications expressed as Condensed Graphs are not implementation dependent thus the programmer is free to concentrate on expressing a solution to the problem rather than on its implementation.

Condensed Graphs [8] represent programs as directed acyclic graphs, in which nodes represent tasks

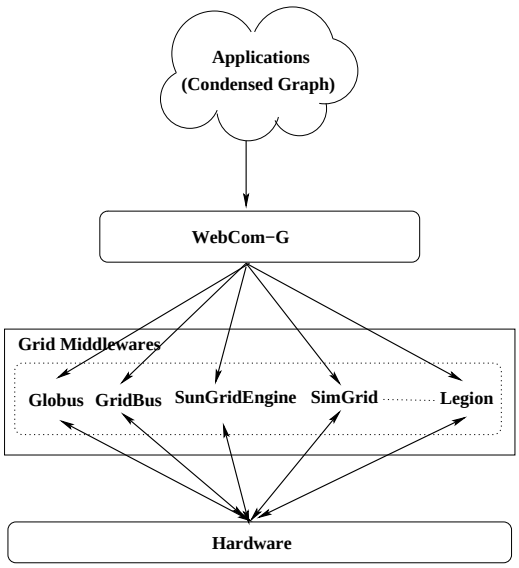

Figure 2: *Top level view of WebCom-G. Applications expressed as Condensed Graphs execute on the WebCom-G platform. WebCom-G can then target computations to different Grid middlewares.*

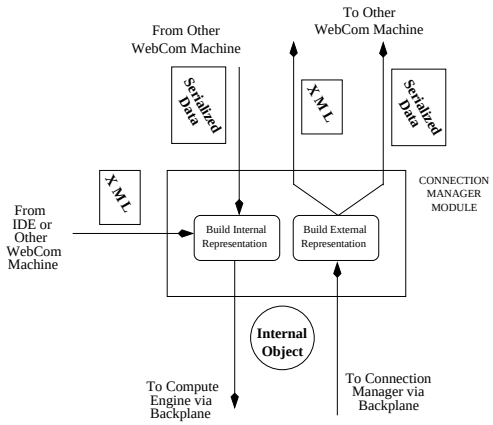

Figure 3: *Architecture of the Web Service Connection Manager. XML or serialized object descriptions of a Condensed Graph are converted to the internal representation and passed on for execution. Likewise, when sending an internal representation it may be converted to XML, serialized object or both.*

and arcs represent inter task sequencing constraints. In addition, the semantics of Condensed Graphs express data-driven, demand-driven and control-driven computations using a single, uniform, formalism. Tasks can range in complexity from simple instructions to invocation of services offered by various middlewares. By representing Condensed Graphs as nodes in other graphs, arbitrarily complex and abstract applications can be developed. Representing a graph as a single node with a defined set of inputs is equivalent to the provision of a standalone service.

WebCom-G contains a mechanism for targeting nodes of a Condensed Graph, i.e. tasks, to specific execution domains. Combining support for executing native applications and targeting mechanisms facilitated the integration of different Grid middlewares, see Figure 2. Each Grid middleware supported by WebCom-G has a unique computation engine plugin. A WebCom-G installation can simultaneously support multiple plugins of the same type. Thus, for example, different middleware tasks can be executed by the same compute engine.

3 Submission of Jobs

A number of mechanisms exist to facilitate job submission and communication between WebCom-G machines. One such mechanism uses Web Services to invoke the job. The XML representation of a job is communicated to WebCom-G via a specific Communication Manager Module. This is responsible for taking the XML description of the Condensed Graph and for building the internal representation of that graph. Once this internal representation is constructed it is forwarded to the execution engine. This is depicted in Figure 3.

The second way of submitting jobs is via a native job transport mechanism involving Object Serialization. WebCom-G machines may submit jobs to each other via this mechanism. This is outlined in Figure 4.

However, this will fail if the graph description is not resident on both machines. Such a failure precipitates the following actions:

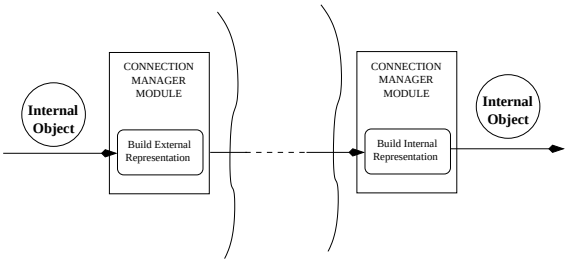

Figure 4: *Jobs being sent between WebCom-G machines are converted to a format suitable for transmission, either XML or Serialized data. The receiving machine is responsible for reconstructing the job and passing it on to its execution engine.*

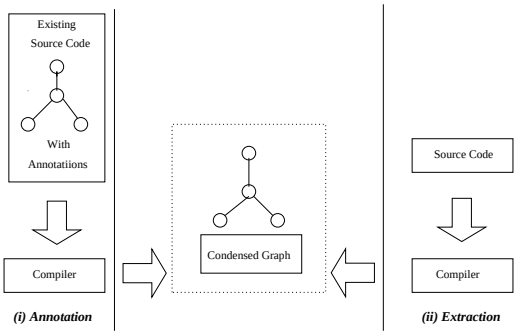

Figure 5: *Native applications are supported by two mechanisms: Annotation and Extraction.*

- The sender uses XML to communicate an XML representation of the graph to the receiver,
- The receiver builds a representation of the task from the XML description and
- Task data is communicated via object serialization.

Once a task description has been constructed, subsequent communication of data for similar tasks may be carried out by object serialization alone. This is useful as the communication cost incurred by sending large XML descriptions may be significant.

A third submission mechanism uses a lightweight command line interface to invoke jobs already resident on a machine. This job submission interface is intended to be used when scripting single application executions during a development cycle.

4 Converting High Level Languages to Condensed Graphs

Two approaches to supporting native applications in WebCom-G[10] are provided: *Extraction* and *Annotation*. Extraction is the process of translating higher level languages into a Condensed Graph representation. Annotation allows the developer to place hints in the source code that direct a special compiler on how to best construct the resulting Condensed Graph representation of the application. These two methods are shown in Figure 5.

The first approach, Extraction, is a process of extracting the parallelism at compile-time. Specifically, this is achieved by translating the applications into Condensed Graphs. This methodology is suitable for legacy applications as well as for proprietary specification languages like Globus RSL. For traditional high level languages, a translator performs a data dependency analysis to identify the parallel blocks within the source code. Condensed Graph conversion rules are applied and the output is written in XML format. This step is illustrated in Figure 6.

In a Globus environment, the Extraction process extracts the jobs specified in the RSL and expresses them and their sequencing constraints as a series of nodes and arcs of a Condensed Graph. WebCom-G subsequently executes the resulting Graphs. This methodology benefits from the inherent features of WebCom-G such as Fault Tolerance, Load Balancing and Scheduling. For example, using the Fault Tolerance capability, if a particular job execution fails, WebCom-G re-schedules the job for execution at a later time.

The Condensed Graphs compiler works in either fully automatic or semi automatic translation mode. Using fully automatic translation with appropriate analysis and interpretation, the uncovered parallel blocks will be converted into the Condensed Graph format. With semi-automatic translation, the uncovered data blocks will be presented to the application

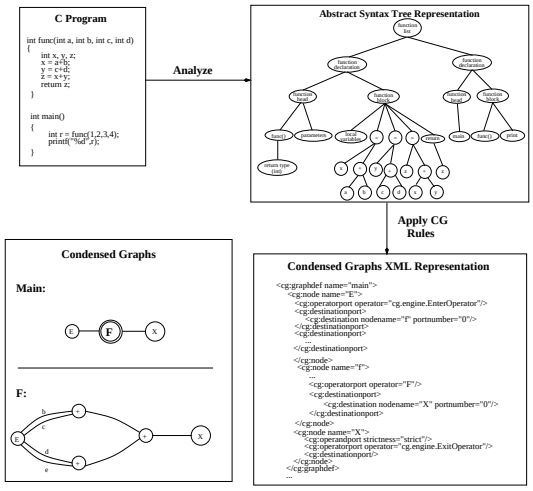

Figure 6: *Example translation of sequential C code to Condensed Graphs(CG) XML representation. The C program is analyzed to produce an Abstract Syntax Tree(AST) representation. Applying CG rules to the AST results in the XML representation.*

developer through the WebCom-G Integrated Development Environment thus enabling further optimisations to be performed.

The second approach, Annotation, allows the developer to identify parallel blocks within the source code by including some Condensed Graph specific annotations. Specifically, this aims to identify coarse grain parallelism. The Condensed Graph APIs are developed for this purpose. Source code containing these annotations can be later optimised using the translation mechanisms mentioned previously. Thus, translating legacy applications into Condensed Graphs enables non Grid aware applications to become grid aware.

The two translation approaches described provide for encapsulating legacy applications and releasing them to a Service Repository. These services are exposed to and may be exploited by the different development environments mentioned. The Condensed Graph XML format acts as a suitable interface for services in this repository, Figure 7. By utilising services uncovered, the development environment does not need to maintain state information about the target execution environment. This is done at run-time by the appropriate execution environment itself. Services, possibly representing complex independent applications, can easily be combined to construct more powerful applications. These applications may use multiple bindings and implementations for their services.

5 WebCom-G$_{IDE}$ Problem Solving Environment

WebCom-G$_{IDE}$ is a development tool that enables visual application development and optimisation. From within the IDE, a user can create, load, save and execute applications. Condensed Graphs are displayed and edited graphically within the IDE.

The IDE also shows a palette of components that may be used to construct Condensed Graphs. This palette includes middleware and legacy services whose details have been added to a central database by middleware interrogators. The IDE checks the database periodically, and dynamically adds any new components found to the palette. The palette is arranged hierarchically, with separate categories for different middlewares and different classes of components.

A new application begins as a single empty Condensed Graph. An application may consist of many Condensed Graphs, and each Condensed Graph has its own window within the GUI.

Nodes are added to an application by dragging them from the palette onto one of the application's development windows. The path of input and result data through a graph is then controlled by placing edges between the various nodes.

Native application support outlined in Section 4 forms one constituent part of *WebCom-G$_{IDE}$*. The translation support can be applied to the specification languages like Globus RSL[6]. Using the Condensed Graph compiler, the jobs and their sequencing constraints can be extracted and converted into Condensed Graph format. The Condensed Graph representation, in XML format, provides a way to execute tasks in different middlewares seamlessly. An sample

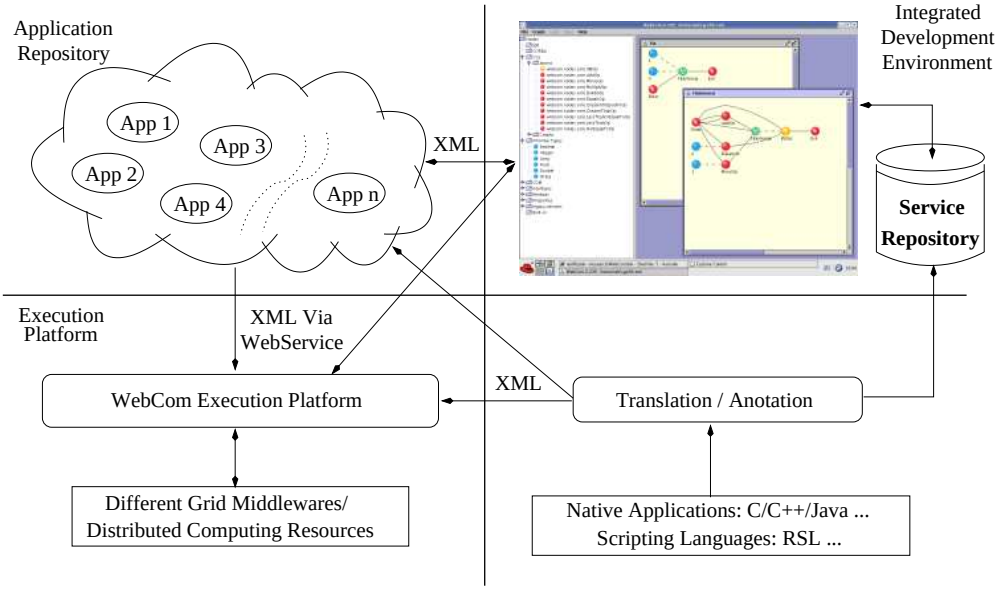

Figure 7: *Interactions between different components in the WebCom-G system. Legacy applications may be translated/annotated to populate the IDE service and application repositories. The IDE can be used to create further applications. Applications can be submitted to the Execution platform via web services.*

illustration of the IDE is given in Figure 8.

6 Conclusions and Future Work

This paper presents ongoing research into the area of integrating a number of Grid middleware service providers at the application level. It outlines the WebCom-G execution platform and its constituent modules. The benefits of such a modularised execution platform to application developers include the removal of Fault Tolerance, Scheduling, Security and Load Balancing operations from the task of the developer since they are vested in the WebCom-G abstract machine. This allows them to concentrate on specifying a problem solution and not problem implementation. An application development environment based on WebCom-G enables traditional application development mechanisms to utilise computational resources made available via different Grid middlewares. It provides two mechanisms that support execution of legacy code. An application development environment for WebCom-G application construction incorporates support for legacy code through the processes of Annotation and Extraction and results in that code having the ability to both invoke Grid services and to be invoked as a Grid service.

Currently Computation Engine modules exist for COM, DCOM, .NET, J2EE, CORBA and Globus. Future work includes expanding WebCom-G to support other Grid systems such as Netsolve[2], DIET[1] and others.

References

[1] Distributed Interactive Engineering Toolbox. http://graal.ens-lyon.fr/DIET/.

[2] Sudesh Agrawal, Jack Dongarra, Keith Seymour, and Sathish Vadhiyar. NetSolve: Past, Present, and Future - a Look at a Grid Enabled

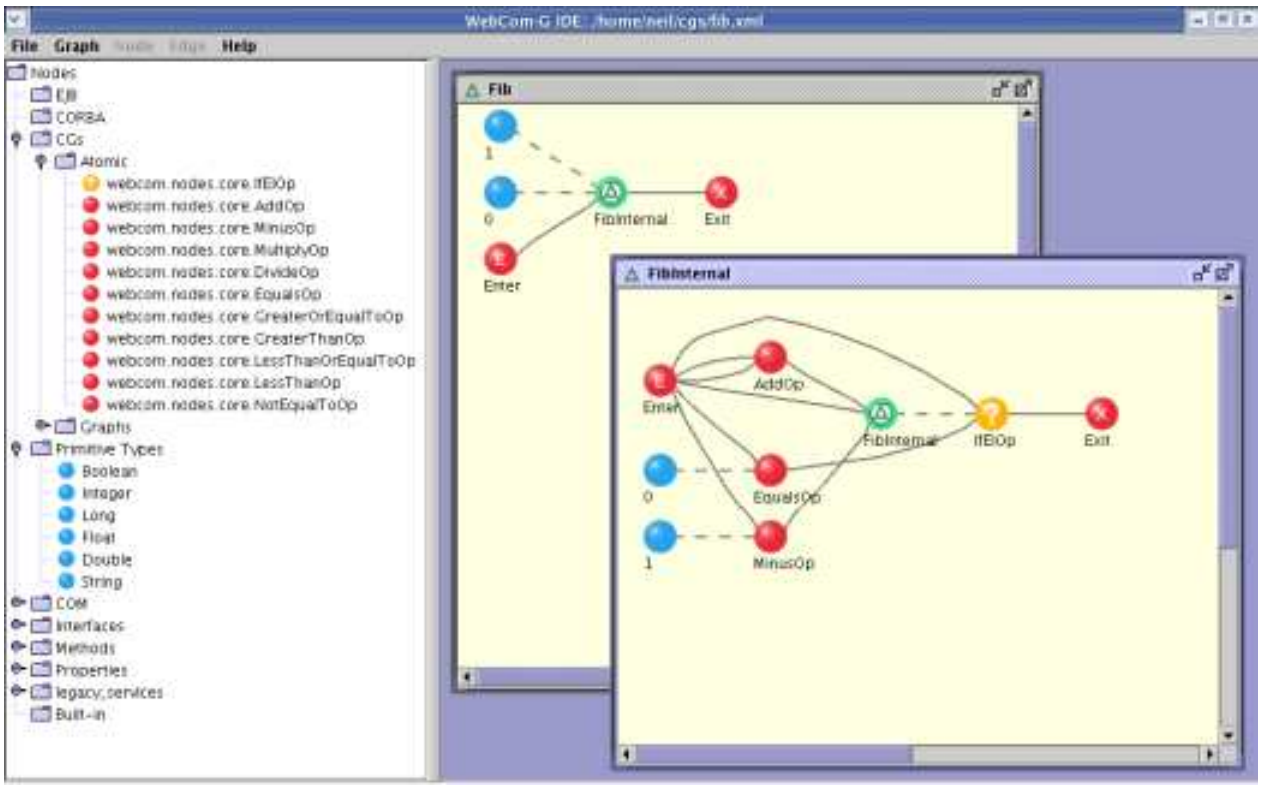

Figure 8: *Sample Screen shot of the IDE, showing the Graphs used to generate a Fibonacci sequence.*

Server. Grid Computing: Making the Global Infrastructure a Reality, Edited by F. Berman, G. Fox, and A. Hey, Wiley Publisher, 2003, ISBN 0-470-85319-0. .

[3] Rajkumar Buyya and Srikumar Venugopal. The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. Proceedings of the First IEEE International Workshop on Grid Economics and Business Models (GECON 2004, April 23, 2004, Seoul, Korea), 19-36pp, ISBN 0-7803-8525-X, IEEE Press, New Jersey, USA.

[4] Henri Casanova. Simgrid: A Toolkit for the Simulation of Application Scheduling. 1st International Symposium on Cluster Computing and the Grid,May 15 - 18, 2001,Brisbane,Australia.

[5] I. Foster and C. Kesselman. The Grid: Blueprint for a New Computing Infrastructure. Published by Morgan Kaufmann Publishers inc. ISBN:1-55860-475-8.

[6] Globus. *Globus RSL.* http://www.globus.org/gram/rsl_spec1.html.

[7] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal. Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework. Book Chapter:High Performance Computing: Paradigm and Infrastructure, Laurence Yang and Minyi Guo (editors), Wiley Press, New Jersey, USA. Fall 2004. (in print).

[8] John P. Morrison. *Condensed Graphs: Unifying Availability-Driven, Coercion-Driven and*

Control-Driven Computing. PhD thesis, Eindhoven, 1996.

[9] John P. Morrison, Brian Clayton, David A. Power, and Adarsh Patil. WebCom-G: Grid Enabled Metacomputing. The Journal of Neural, Parallel and Scientific Computation. Special issue on Grid Computing. Guest Editors: H.R. Arabnia, G. A. Gravvanis and M.P. Bekakos. Accepted for publication April 2004.

[10] John P. Morrison, Sunil John, and David A. Power. Supporting Native Applications in WebCom-G. Distributed and Parallel Systems Cluster and Grid Computing Series: The Kluwer International Series in Engineering and Computer Science. Editors: Zoltan Juhasz et al, Vol. 777, September, 2004.

[11] John P. Morrison, James J. Kennedy, and David A. Power. WebCom: A Volunteer-Based Metacomputer. The Journal of Supercomputing, Volume 18(1): 47-61, January 2001.

[12] John P. Morrison, David A. Power, and James J. Kennedy. An Evolution of the WebCom Metacomputer. The Journal of Mathematical Modelling and Algorithms: Special issue on Computational Science and Applications, 2003(2), pp 263-276, Editor: G. A. Gravvanis.

[13] David A. Power, Adarsh Patil, Sunil John, and John P. Morrison. WebCom-G. Proceedings of the international conference on parallel and distributed processing techniques and applications (PDPTA 2003), Las Vegas, Nevada, June 23-26, 2003.

[14] Geoff Stoker, Brian S. White, Ellen Stackpole, T. J. Highley, and Marty Humphrey. Toward Realizable Restrected Delegation in Computational Grids. European High Performance Computing and Networking. Amsterdam, June 25-27, 2001.