WebCom Core Information Module

Pushpinder Kaur Chouhan, Archana Patil, Adarsh Patil and John P. Morrison
Department of Computer Science,
University College Cork, Ireland.
{p.chouhan,archana,adarsh,j.morrison}@cs.ucc.ie

Abstract

Grid platforms are very promising but are very challenging to use because of their intrinsic heterogeneity in terms of hardware capacities, software environments and system administrator preferences. To avail of the Grid platforms' advantages easily and efficiently, Grid middlewares are being developed. WebCom-G, based on the WebCom platform, is a grid middleware that hides the complexity of Grid from the end users. WebCom is modular in architecture, constructed from *core* and *user modules*.

This article outlines the functionality of the WebCom *Information Module* and presents a brief overview of WebCom and its other core modules. It also describes module interactions between WebComs running on different machines. The functionality of the core Information Module is presented in detail. In particular the questions: "What information the core information module provides?", "How is the gathered information about the resource is stored within WebCom?", "How stored information is retrieved and transferred within the WebCom and to other WebComs in the environment?", "How this information can be efficiently used by other *core modules* as well as *user modules*?" are addressed.

I. Introduction

Grid [13] is a socially shared, integrated system consisting of unlimited resources shared by multiple administrative domains that grant transparent access to resources under commonly agreed set of rules. The Grid [8] environment is available to solve large scale applications, that are computation and storage intensive. The resources within a Grid environment are heterogeneous, non-persistent and are spread across a multiplicities of users. In most Grid environments, the user applications are submitted in a round robin manner. However, it is also possible for the user to specify their own scheduling algorithms. Existing systems such as NetSolve [4], DIET [1], for example provide no information to the end user about the status of the distributed resources used. In order to make appropriate scheduling, load balancing and fault tolerance decision information on the available resources has to be gathered and maintained. To develop strict models for Quality-of-Service, dynamic and up-to-date information is required. Such information can be utilized in process management models, programming models and service oriented models of computing.

The proliferation of Grid computing technologies will heavily depend on the reliability of the Grid infrastructure deployed. Only with reliable Grid services, a researcher will be able to embrace the modern technology and leverage its capabilities to assist in collaboration and sharing of resources. At the heart of the Grid infrastructure is the information system. This stores and transports information about available resources to Grid users and services. If the information system experiences faults or yields inappropriate information, Grid services will also be detrimentally affected.

Tightly coupled resources such as clusters, clusters of workstations (CoWs) utilize centralized information providers such as the Network Information Service (NIS), which stores information about the individual nodes forming the cluster. In contrast Grid environments acquire resource information through information services hosted by different *virtual organizations*.

Information about the distributed resources is very crucial for the efficient management, scheduling and execution of the applications on these resources. Information retrieval, gathering and management of resources is very important for Grid users. These include the Grid administrators, end users (who exploit the Grid to solve their applications), application vendors (who develop applications for end users) and

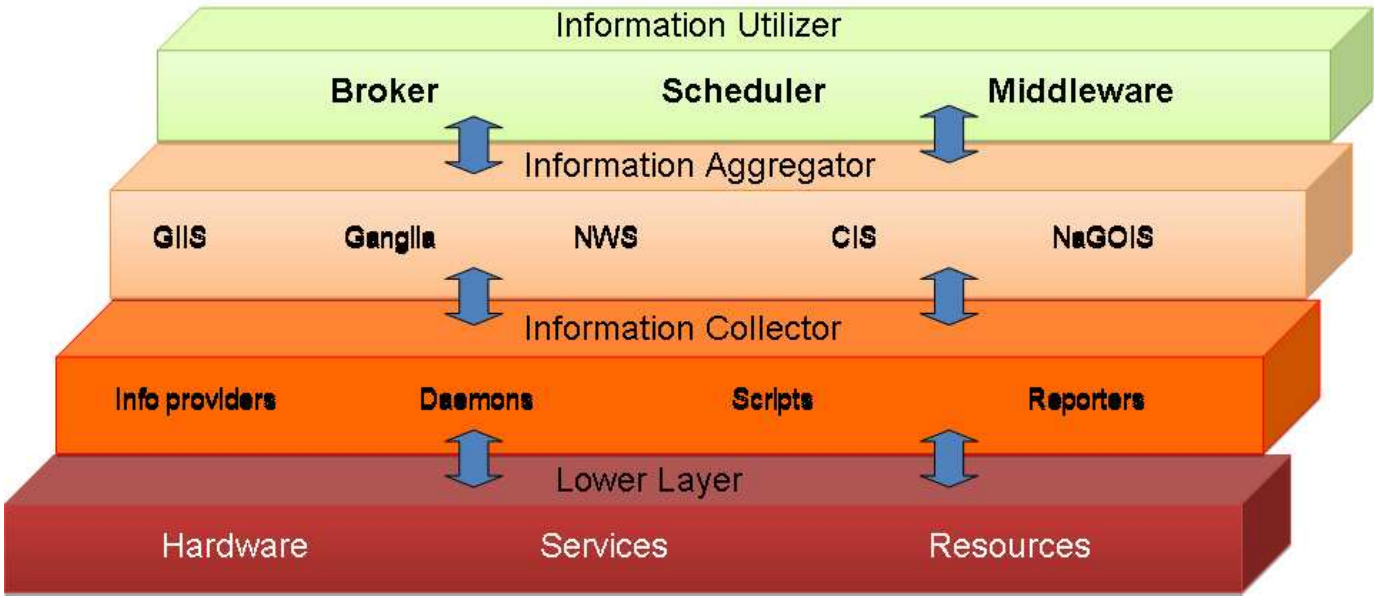


Fig. 1. Overview of a Grid Information System.

Grid developers. Grid systems like Globus [7] and Condor [14] have implemented their own information services for managing the resources on which they are installed. These Grid systems have their individual information services. They access and use the information in similar manner except they provide different techniques and methodologies for doing so.

Fig. 1 presents a general layered overview of typical Gird information systems. These systems can be divided into four layers: the lower layer, the information collector layer, the information aggregator layer and the information utilizer layer. Each layer has a specific task to complete. Different Grid systems use their preferred techniques to accomplish the tasks associated with each layer. The **lower layer** is the foundation layer, which provides the computational, storage and visualization services. The **information collector layer** gathers the information about the status of the underlying hardware, services exposed and available resources. Typically this is done using scripts or daemons or other information providing tools. The information collector gathers and pushes information to the information aggregator. The **information aggregator layer** stores the collected information in a cache. Periodic status information is gathered applying either push or pull techniques. Finally, the information is used by the schedulers, brokers, or Grid middlewares of the **information utilizer layer**. Upon getting a query request from the end user, information utilisers check for the latest high level indexed information and schedule the submitted request.

The remainder of this article presents an information model that gathers the status information of grid resources and displays it in a user friendly manner. We have implemented this as a core module in WebCom [10].

This article is organized as follows: Section II presents an overview of WebCom. Section III presents the functionality of the Information Module and its usage in detail. Finally, Section IV concludes and presents some future work.

II. WebCom

WebCom [11] is a virtual machine that executes applications that are specified as Condensed Graphs (CG) [9]. WebCom separates the application and execution environments by providing both an execution platform and a development platform. The independence provided by separating these two environments facilitates computation in heterogeneous environments; the same CG programs run without change on a range of implementation platforms, from silicon-based Field Programmable Gate Arrays [12] to the

Java-based WebCom. Fault tolerance, load balancing, scheduling and exploitation of available parallelism are handled implicitly by WebCom without explicit programmer intervention.

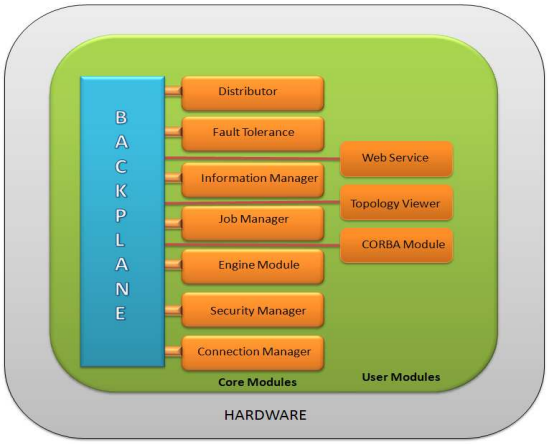


Fig. 2. Overview of WebCom virtual machine residing on the machine.

WebComID = sinai-1176396218502
osName = Linux
osArch = i386
osVersion = 2.6.17
HostName = sinai
HostAddress = 143.239.1.4
AvailableProcessors = 2
Time = Mon May 21 14:51:22 GMT 2007
UserCountry = GB
UserTimezone = Europe/London
JavaVersion = 1.4.2_13
FreeJVM = 262424
CPULoad = 90
UsedMemory = 15
Avgload = 93

Fig. 3. WebComAd_sinai-1176396218502.machine

The execution platform consists of a network of dynamically managed machines, each running WebCom. A WebCom test-bed can assume as a traditional client-server connection model or the more contemporary peer-to-peer model. A WebCom machine that acts as a server, submits task to other WebComs for execution. A WebCom machine that acts as a client carries out the task computation. Each WebCom is pre-installed and initially operates in the role of a client. Upon receipt of a task representing a CG (A task can be partitioned for further distributed execution.), such clients are promoted to act as servers. Once a promoted WebCom returns a result for every such task it has been allocated, it is demoted and acts as a simple client once more.

A WebCom machine is constructed from a number of plugable modules. These are divided into two categories: *Core Modules* and *User Modules*, shown in Fig. 2. To construct a fully functional WebCom, core modules that provide fault tolerance [2], load balancing [3], scheduling [15], a compute engine, security [6] and communications have to be installed. Each of these modules is a plug-in to a core module called the Backplane. User modules are defined by the users and can also be plugged into the WebCom Backplane on-demand.

Communications between WebComs use a messaging system. Modules can send messages between themselves on the local WebCom or to any module on other connected WebComs. Initial configuration will specify where these modules are to be found, either on the local host or on another WebCom host. In execution, these modules are obtained and installed, thus dynamically bootstrapping and configuring each host.

Application execution is initiated by a user on a single WebCom. As nodes become available for execution, they are encapsulated within messages and passed to the distributor module. The distributor decides where the node is to be executed according to installed policies. If it is to be executed locally, it is passed back to the compute engine. For remote execution, information about the WebCom selected to execute the node is incorporated into the message, and the message is passed to the communications manager module for distribution to the selected WebCom.

Once such a node has completed its operation, a result message is created and returned to the originating WebCom. This then incorporates the returned result into the node's graph and the execution proceeds. If a remotely distributed node fails to complete its task, the fault-tolerance module on the distributing WebCom will cause the node to be rescheduled to an alternative compatible WebCom. If no such WebCom is available, the node is retained for subsequent assignment.

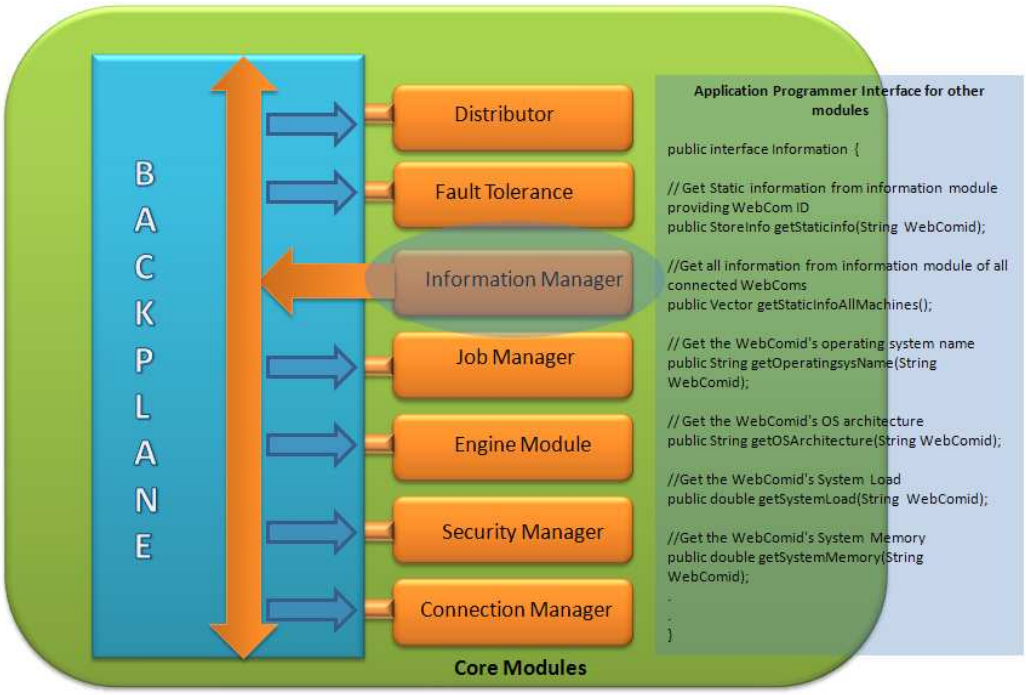


Fig. 4. Passing information between modules on same WebCom.

III. Information Module

The *information module* is an essential core module for WebCom. This module accesses all the basic information of a resource on which WebCom is installed. Static information such as type of operating system, number of processors, etc. are stored in the module cache. Dynamic information, such as CPU load, free memory (RAM) and uptime are accessed periodically or on demand by other modules.

A. Information Gathering

To gather the required information on the status of the resources, daemons and scripts have been used. Information can be collected using two models: Push and Pull. When a WebCom is launched on any resource, it creates an information file (shown in Fig. 3) and stores it in its cache. This file can be used to facilitate the usage of the Condor classAd [5] mechanism within WebCom. The information module also propagates this machine information to other connected WebComs provided their security configuration permits. Information is refreshed at set intervals pushed to other known WebComs. This is the push model. Static information such as operating system, its version and number of processors are sent only once when WebCom is first launched. Dynamic information such as CPU load, free memory and uptime are periodically updated by the daemon and scripts. A pull model can be used by WebCom to obtain this dynamic information. The mechanism employed can be tailored to request only the information that is appropriate. The pull mechanism for propagating information is handled is a point-to-point manner. That is, the requesting WebCom sends a message to a selected client requesting some information. This client replies only to the requester. In contrast, the push model propagates its information to all clients at regular intervals. This interval is specified as a Time To Live (TTL) for the information. When the TTL has expired information is resent.

B. Information Storage

Information gathered about each machine is stored at two locations: 1) Locally and 2) Remotely on other connected WebComs. WebCom stores its system information in the form of a file named as

WebComAd_WebCom_id.machine. An example of such a file is shown in Fig. 3. Information about other WebCom machines is stored in a hash-table. The index into this in hash-table is the unique WebCom id of the client machine. The data associated with this index represents all the gathered information about that WebCom.

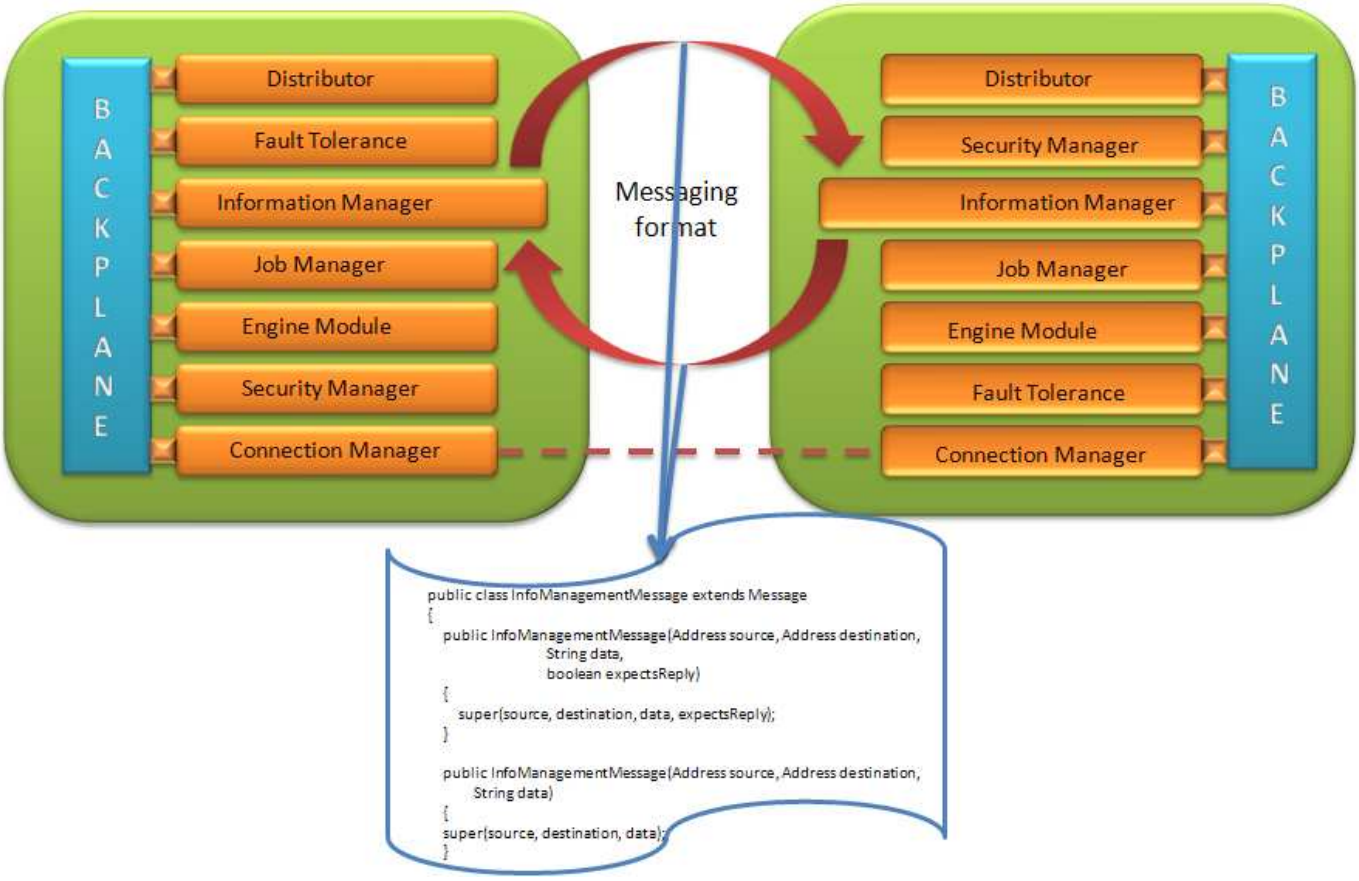


Fig. 5. Passing information between WebComs.

C. Transfer Information

Gathered information can be accessed in three ways: (1) by the core and/or user modules within WebCom, (2) by connected WebComs, or (3) by using the information GUI, shown in Fig. 6. Information required by other modules in WebCom is accessed via Application Program Interfaces (APIs). Fig. 4 shows the API mode availabled by the information module. Remote WebCom access information via the messaging sub-system. The information module has its own message formate and can respond directly to request from other WebComs. Fig. 5 shows how the information between two connected WebCom machines is exchanged. A WebCom sends a request query with a message type which specify the demanding information tag to its peer WebComs. In response to the query the requested WebCom executes the scripts to retrieve its current status and replies with an answer. A query message can ask for load, uptime, free memory, and average load, for example.

D. Information accessing through GUI

The information accessing GUI is user friendly and laid out in a simple easy to use manner. This GUI display the status information of connected WebComs.

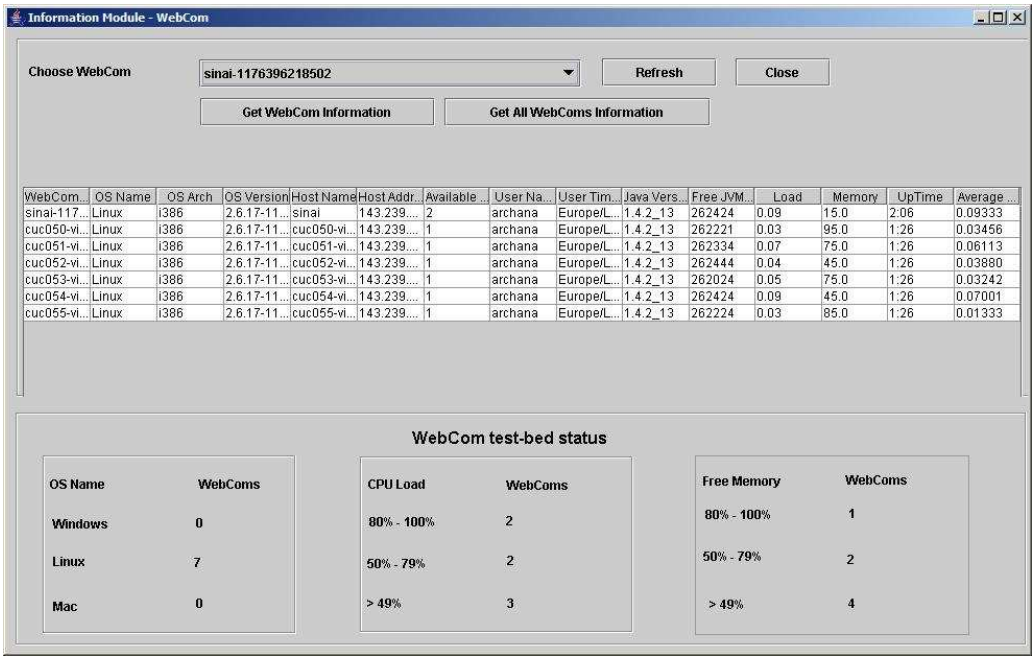


Fig. 6. GUI to access information from connected WebComs.

WebComID displays the WebCom reference name that is generated when a machine join the WebCom test-bed.
OS Name displays the operating system installed on the machine.
OS architecture displays the architecture number of the machine.
Host Name displays the name of machine. The WebComID is different from the Host Name. This can be seen in Fig. 6, where the machine with Host Name “sinai” has a WebComID “sinai-1176396218502”.
Host Address is the IP address of the machine.
Available processors displays the number of processors on the machine.
User Name specify the name of the user that launched WebCom and connected it to the testbed.
Load presents the current CPU load of the machine.
Memoryspecify the amount of memory occupied in the machine.
Uptime tells how long the machine has been running.
Average Load displays the CPU load average of last fifteen minutes.

In addition, User Time Zone, Java Version and Free JVM are also displayed.

The title bar shows that it is a plug-in for WebCom core information module. The Combo Box lists the connected WebComs. *Refresh* button updates the connected WebCom list in the Combo box. *Get WebCom Information* button displays the information of the selected WebCom (in Combo box) as a table with respect to the column names. *Get All WebComs Information* button displays information of all connected WebComs. The rows and columns in the table are not editable. A different panel at the bottom of the frame is a *WebCom test-bed status*. Information here shows how many machines have specific operating systems, the number of machines that have certain amount of free memory and CPU load bands are also displayed. The information presented on the status panel can help in scheduling, load balancing and other decision making. For example, for a data intensive application a suitable WebCom can be selected from those that have more than 80% of free memory space. This GUI can be embedded within the WebCom or run as a standalone application. It is also able to collect information about the WebCom server and connected clients recursively.

E. Utilization of Information

The information about a WebCom machine can be used in different manners to improve the performance and efficiency of a WebCom test-bed. The distributor module can be most benefited by the information module. The distributor decides when and where to execute tasks according to policies and algorithms. These algorithms can be matched to the specific demands of the users. For example, one user may request to execute task based on a least CPU based algorithm, while another user may request the task to be executed in a round robin manner. It is even possible for a single application to use different execution policies. The information used by the policies is provided by the information module. The distributor selects a set of WebCom machines depending upon users demand: For example, only Linux machines or only Windows machines can be used for specific application execution. Thus, information module is the basis for load balancing and scheduling of jobs done by the distributor module.

Information about the availability of free memory, CPU average load, machine, uptime and operating system type helps fault tolerance module to use specific fault survival techniques. The job manager module allows applications to be initiated, suspended, monitored, migrated to different locations on WebCom test-bed and resumed. For these activities the job manager module gets support from distributor module and fault tolerance module. Thus, job manager module indirectly benefits from the information module.

User modules can also benefits from the information gathered by the information module. For example, the topology viewer user module can easily check the connectivity of the WebComs in the testbed. With the use of *Refresh* button dynamic connectivity of the WebComs can be obtained. It is at the implementor discretion as to what uses they make from the available information.

IV. Conclusion

This article presents WebCom's information module in detail. The working of the information module, methods of retrieving information, information storing techniques, and utilization of the information are explained. The WebCom information module provides a number of advantages over most of the existing information services: (1) It does not require administrative privileges to install, use and execute, (2) It can provides information about the system load on SMP machines as well as for systems those are behind gateways/firewall but running WebCom, (3) It provides a simple GUI to display the gathered information, (4) Gathered information can be published in different forms (text, ClassAds and XML). This XML representation of the data can be used to transfer status information between different middlewares providing application independence, and (5) It is Web Service compliant and can work between different middlewares requiring different information format, for example, ClassAds of Condor.

There are several possibilities for future enhancement of the information module with respect to WebCom status and applications submitted to the WebCom. The information about the jobs executing on the WebCom machines, status of these jobs and cost of job on a particular machine will be added in the near future. Information storage techniques will be added to keep the long information history on the machine, so as to use it for WebCom test-bed analysis fault tolerance, load balancing and security.

Acknowledgment

The authors would like to thank Dr. David Power for valuable discussion and insightful ideas.

References

[1] A. Amar, R. Bolze, A. Bouteiller, P. K. Chouhan, A. Chis, Y. Caniou, E. Caron, H. Dail, B. Depardon, F. Desprez, J.-S. Gay, G. Le Mahec, and A. Su. Diet: New developments and recent results. In *CoreGRID Workshop on Grid Middleware (in conjunction with EuroPar2006)*, Dresden, Germany, August 28-29 2006.

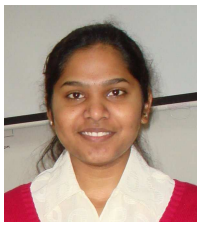
[2] A. Avizienis. Fault-Tolerence: The survival attribute of digital systems. In *IEEE, Vol 66, No. 10*, pages 1109–1125, 1978.

[3] W. Becker. Dynamic Load Balancing for Parallel Database Processing. Faculty Report No. 1997/08, Institute of Parallel and Distributed High-Performance Systems (IPVR), University of Stuttgart, Germany.

[4] H. Casanova and J. Dongarra. NetSolve: a network server for solving computational science problems. In *Supercomputing '96: Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM)*, page 40, 1996.

[5] N. Coleman, R. Raman, M. Livny, and M. Solomon. Distributed policy management and comprehension with classified advertisements. Technical Report UW-CS-TR-1481, University of Wisconsin - Madison Computer Sciences Department, April 2003.
[6] S. N. Foley, B. P. Mulcahy, T. B. Quillinan, M. O'Connor, and J. P. Morrison. Supporting heterogeneous middleware security policies in webcom. *Journal of High Speed Networks*, 15(3):301–313, 2006.
[7] I. Foster. Globus toolkit version 4: Software for service-oriented systems. pages 2–13, 2006.
[8] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.
[9] J. P. Morrison. *Condensed Graphs: Unifying Availability-Driven, Coercion-Driven and Control-Driven Computing*. PhD thesis, Technische Universiteit Eindhoven, 1996.
[10] J. P. Morrison, B. Clayton, D. A. Power, and A. Patil. Webcom-g: Grid enabled metacomputing. *The Journal of Neural, Parallel and Scientific Computation. Special Issue on Grid Computing.*, 2004(12)(2):419–438, April 2004.
[11] J. P. Morrison, B. Coghlan, A. Shearer, S. Foley, D. A. Power, and R. Perrott. Webcom-g: A candidate middleware for grid-ireland. *The International Journal of High Performance Computing Applications*, 2007.
[12] J. P. Morrison, P. J. O'Dowd, and P. D. Healy. Searching RC5 Keyspaces with Distributed Reconfigurable Hardware. ERSA 2003, Las Vegas, June 23-26, 2003.
[13] D. A. Reed, C. L. Mendes, C. da Lu, I. Foster, and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure - Application Tuning and Adaptation*. Morgan Kaufman, San Francisco, CA, second edition, 2003. pp.513-532.
[14] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.
[15] J. Watts and S. Taylor. A Practical Approach to Dynamic Load Balancing. Scalable Concurrent Programming Laboratory, Syracuse University.

Pushpinder K. Chouhan is a Post-doctoral researcher working in Centre for Unified Computing, University College Cork, Cork, Ireland. She got her Research Masters and PhD in Computer Science from ENS-Lyon, France. She did her Bachelor's degree in Computer Science and Engineering from SLIET, Punjab, India. She has reviewed for a number of conference and journal publications. Her areas of interests include Automatic Deployment, Grid & Distributed Computing.



Archana Patil is a Research Scholar working in one of the leading Grid and Distributed Computing research group, Centre for Unified Computing, Cork, Ireland. Her areas of interests include Grid & Distributed Computing and distributed application development using WebCom. She has a bachelor degree in Computer Science and Engineering from G M Institute of Technology (Visvesvaraya Technological University), Davangere, Karnataka, India. She is currently pursuing Research Masters in Department of Computer Science, University College Cork, Cork, Ireland.

Adarsh Patil is a Research Scholar working in one of the leading Grid and Distributed Computing research group, Centre for Unified Computing, Cork, Ireland. His areas of interests include Grid & Distributed Computing and Economic Models in Grid Environment. He has a Bachelor's degree in Computer Science and Engineering from University B.D.T College of Engineering (Kuvempu University), Davangere, Karnataka State, India. He is currently pursuing his PhD in Department of Computer Science, University College Cork, Cork, Ireland. He has reviewed for a number of journal publications and has been a program committee member, session chair and organizing committee member for a number of conferences. He has also been a tutor/demonstrator in the Computer Science Department where he prepared and delivered tutorials in programming and Software Engineering.

John P. Morrison is the founder and director of the Centre for Unified Computing. He is a co-founder and co-director of the Boole Centre for Research in Informatics and a co-founder and co-director of Grid-Ireland. Prof. Morrison is a Science Foundation of Ireland Investigator award holder and has published widely in the field of Parallel Distributed and Grid Computing. He has been the guest editor on many journals including the Journal of Super Computing and the Journal of Scientific Computing. He is on the Editorial Board of Multi-Agent and Grid Systems: An International Journal, published by ISO Press, and the International Journal of Computational Intelligence: Theory and Practice (IJCITP). He is a member of the ACM and a senior member of the IEEE. Prof Morrison is a member of the I2Lab Advisory Board in the University of Central Florida. He has served on dozens of international conference programme committees and is a co-founder of the International Symposium on Parallel and Distributed Computing.