Building Computational Grids Using Ubiquitous Web Technologies

Robert John Walters, Stephen Crouch, Phillip Bennett

Building Computational Grids using Ubiquitous Web Technologies

Robert John Walters, Stephen Crouch and Phillip Bennett

University of Southampton, Highfield, UK. SO17 1BJ
{rjw1, stc, pb903}@ecs.soton.ac.uk

Abstract. Grid computing is an exciting development which promises to be the enabling technology for many users with periodic requirements for massive computing power. There are a number of Grid computing infrastructures available which are fully featured, powerful, efficient and secure. However, for novice users, these systems are not easy to setup and use which presents a significant barrier to their adoption. M-grid offers an alternative approach which permits the creation of a computational grid able to accept tasks from any user with access to the web and distribute them to machines running standard a web browser without any security implications.

Keywords. Lightweight grid computing, computational grid, Applet, m-grid.

1 Introduction

One of the most widespread applications of grid technology is a computational grid [1-3] whereby a network of machines is organized to permit the distribution of processing power to users with computationally intense tasks. Such a system accepts tasks from users, distributes them for processing and collects the results to pass back to the users. To achieve this, software is installed on all of the machines involved. It is often also necessary to install software on users' machines to present their tasks to the system. This software [1-7] is crafted to ensure the resulting grid offers the very best performance and security but is usually quite extensive and requires considerable configuration. This is not a problem for experts using dedicated hardware but it can be an insurmountable barrier for the interested potential user who needs a light weight system which they can install and use with a minimum of initial effort.

2 M-Grid in Outline

The heart of a computational grid is a mechanism for tasks to be sent to other machines for execution. This is risky for the receiving machine ("node") since it involves executing code supplied by a third party (Fig. **1**). This code might be poor quality; or even be malicious. The traditional solution is to ensure tasks are only ac-

cepted from trustworthy users. Nodes then trust coordinating systems only to distribute safe code [8].

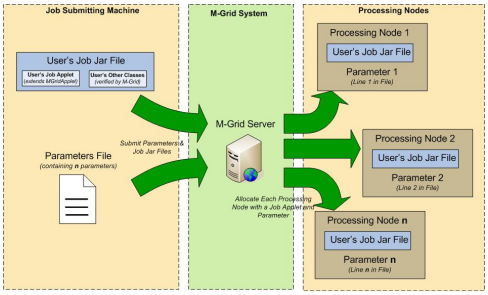

Fig. 1. Distribution of Applets to Nodes

M-grid avoids these issues by using a mechanism for remote execution of code which is available on nearly every machine: the sandbox used by web browsers to execute java applets [9-11]. No software needs to be installed as the sandbox is available within the web browsing software on most machines. The sandbox insulates the host from bad behaviour by applets. Preventing applets from damaging of interfering with their hosts, places some limitations on applets but it was felt that these are outweighed by the fact they can be run on virtually any machine without formality. The coordinating machine supplies tasks to nodes as applets dynamically embedded into web pages. All that is necessary for a machine to join m-grid as a computational node is a java enabled browser. Users with jobs to execute upload these (as java applets) to m-grid using a web based interface. A proof of concept system [12] was built using the ASP technology provided within the Microsoft IIS web server supplied the Windows operating system [13-15]. The system has now been completely redeveloped using JSP [11, 15, 16]. At the same time, some additional features have been added, enabling users to specify that tasks be executed many times with different parameters, some resilience to the failure of nodes to return results and optional security features.

3 Joining and using M-Grid

There are three roles to a computational grid: the computation nodes, the users with tasks to execute and the coordinator who arbitrates between the other two. Since there is no software to install and no security/trust relationships to establish, joining m-grid as a computational node is trivial; almost any machine can take part. All that is required is open the m-grid page with a Java-enabled browser.

A user who wishes to use m-grid to execute a task first embeds their task into an applet (see below). They then upload this to the m-grid using their web browser.

The coordinator role is the only element which requires any effort to establish. The co-ordinating machine needs to run the Tomcat Java Application container (available free from Apache Software Foundation) [16]. All that then needs to be

done is for the m-grid application files to be copied to the correct directory and the URL of the m-grid home page to be advertised.

3.1 The Lifecycle of an m-grid Task

A user with task to perform follows the link to the job submission page from the m-grid home page which guides them through the submission process of specifying the jar file containing their applet and an optional file of (sets of) parameters. When a processing node becomes available, the co-ordinator selects a job from those waiting for execution, embeds it in a web page and serves this up to the node. Where a file of parameters is supplied, each line is supplied to a separate instance of the applet. An applet/parameter pair is referred to as a "job instance".

On completion, the applet returns its results to the coordinator which identifies and stores the results ready to be returned to owner upon request. The results are retained by the coordinator until they are deleted by the task owner.

3.2 Constructing a Task for Submission to m-grid

In common with most computational grid infrastructures, users wishing to m-grid to execute a task for them must supply it in a form which palatable to the grid infrastructure. In the case of m-grid this means that it must be an applet in a single jar file which can execute within the sandbox of a standard browser. M-grid adds some further requirements which concerned with marshalling outputs at the co-ordinator.

The m-grid task development kit provides resources needed to create applets in the required form. The main class of the applet must inherit from the supplied "MGridApplet" class (which itself inherits from the standard applet class). This class encapsulates the various methods for the developer to read parameters and write results. For development, output methods write onto face of the applet. When submitted to m-grid these methods are substituted with others which also marshal and return the output to the task owner (via the coordinator).

3.3 Submitting a task to m-grid

A user with a task to submit to m-grid navigates to the m-grid job submission page and uploads their jar file containing the applet together with an optional text file containing the necessary parameters (if required).

Once received by the m-grid coordinator, classes relating to methods provided by the MGridApplet class are substituted with alternatives which include additional functionality required for the applet to read its parameters and return results. The jar file is also checked for all necessary class files and the class containing the main method is identified. A unique job number is allocated for each set of parameters in the file. This job number, the modified user jar file and the parameters are then stored to await execution. Where no parameters file is supplied, a single job instance is stored. This process is shown in Fig. **2**.

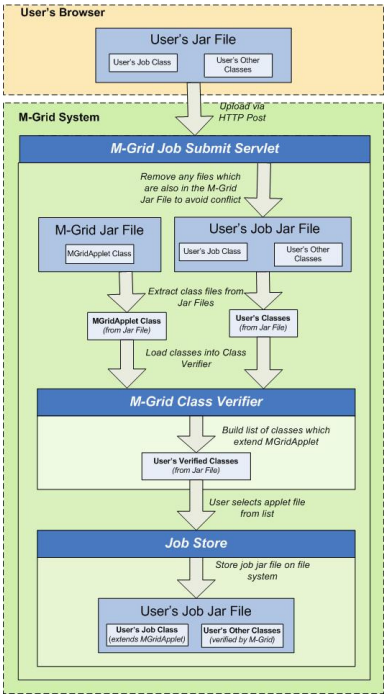

Fig. 2. Uploading a task to m-grid

3.4 Execution of a Task

When a node becomes available, the coordinator checks for job instances awaiting execution. It there are no waiting instances, a place holder page is served states that there is no work awaiting execution and causes the node to check again after a short delay. If there are job instances awaiting execution, one is selected, embedded in a page and served up to the node.

The coordinator monitors the status of job instances. Any job instance taking an excessive period of time to produce a result will be downloaded to further nodes as they become available. The coordinator accepts and records the first result it receives; repeat results arrive are discarded. See Fig. **3**.

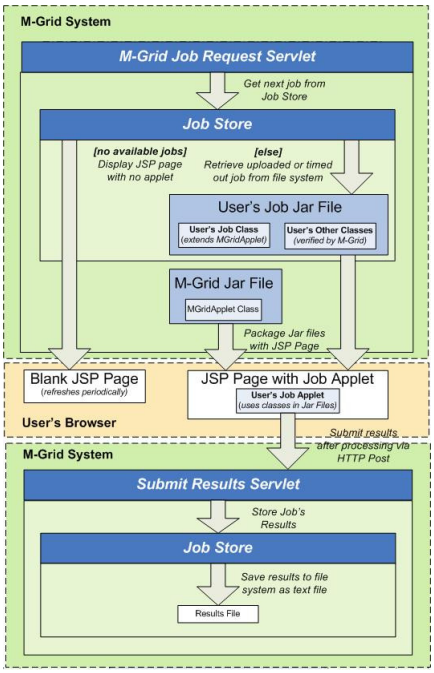

Fig. 3. Job instance execution

4 M-Grid in Action

A new user's first contact with m-grid will be the homepage**.** If they wish to volunteer processing capability, all they need to do is to follow the link to "Volunteer your CPU". That's all. If m-grid has work to be done, they will see applets appear in the page, work and send back results. If m-grid is idle the page says so and periodically checks for tasks. When the user decides that they no longer wish to be supporting m-grid's computation they simply close the browser window or navigate to a different page. Fig. **4** shows an m-grid application executing. A user with a task follows the "Submit a job" link to the page shown in Fig. 5. Using this page they first name to their task for later identification. They then give m-grid the location of the jar file containing their applet and a text file of parameters, if required.

The user may then use the "View Jobs" page to see a list of their tasks. The page displays the name of the task and the status of all the job instances within the task. In the example shown in Fig. , there is one task named Task 1 of three job instances, of which the second instance has been allocated (downloaded) to a node and the third is waiting. The first has completed and the link points to a text file containing the outcome, unless an exception occurred in which case details of the event will appear instead.

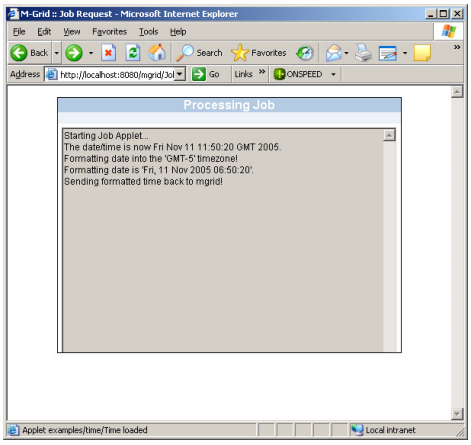

Fig. 4. An m-grid applet executing

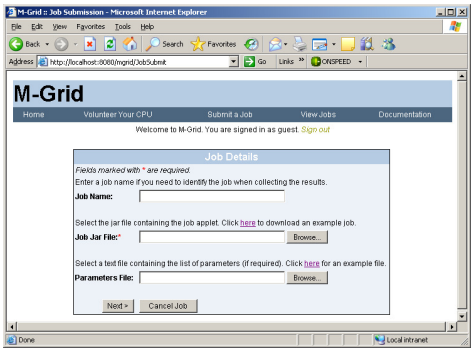

Fig. 5. The job submission page

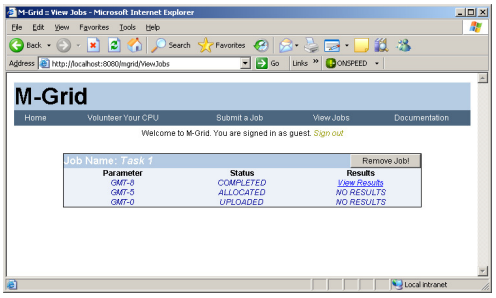

Fig. 6. The Results Page

5 Issues and Further Work

M-grid has its limitations which will prevent it from challenging more conventional grid infrastructures. Most are inherited from the applet mechanism which it uses. For example, m-grid user applications/tasks cannot use of the local file system on the machine where it is executing and an applet's communications are limited. These constraints are imposed on applets in return for being allowed the privilege to execute without formality. The implementation of m-grid described here has improved on the first implementation in a number of respects:

As m-grid is now a web application running within a container, the previous constrain that the coordinator role be executed on a machine running a Microsoft operating system has been relaxed. M-grid may now be run on any platform for which a suitable application container is available.

Users are now permitted to use any number of classes (provided they are packed into a single jar file).

This implementation is able to survive and complete tasks properly if an appl doesn't complete, for whatever reason. This same behaviour permits it to handle tasks which throw exceptions.

The new implementation permits users to execute a piece of code many times with different parameters to inject this into the system as a single task.

The requirements placed on the user when building their applet have also been simplified by encapsulating them all into the single MGridApplet pure virtual class from which all m-grid applets inherit.

One issued which hasn't been investigated in detail is the performance penalty associated with executing tasks as applets inside a web browser. However, casual observation suggests that the actual performance penalty is modest.

6 Conclusion

Mainstream grid software is highly sophisticated and provides the very best performance and security. These are large, complex systems which demand significant quantities of hardware, time and effort to establish. However, this is a real barrier for the potential user who is curious about the potential of grid computing and would like to know more.

Much of the difficulty associated with building grid systems arises from the need to install and configure software on all of the machines which will take part. The problem is not just the logistics of reaching the machines the software. It is also many users don't have full control of their machines. Hence a "trial" installation of a grid infrastructure can require obtaining permissions as well as the physical installation. By using the applet mechanism, m-grid avoids most of these difficulties.

The new implementation of m-grid permits a novice potential user to set up a computational grid with very little effort and minimal cooperation from others who have access to java enabled browsers. They can then experiment with the operation of a computational grid. By the time they reach the limits of the m-grid, they will be in a strong position to select, install and operate one of the "full size" grid infrastructures.

References

1. Altair Engineering Inc: Portable Batch System. (2004)
2. Litzkow, M., Livny, M.: Experience with the Condor Distributed Batch System. In: Workshop on Experimental Distributed Systems. IEEE (1990)
3. Livney, M., Basney, J., Raman, R., Tannenbaum, T.: Mechanisms for High Throughput Computing. SPEEDUP Journal 11, 36-40 (1997)
4. Erwin, D.W., Snelling, D.F.: UNICORE: A Grid Computing Environment. Lecture Notes in Computer Science 2150, 825-839 (2001)
5. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scaleable Virtual Organization. International Journal of Supercomputer Applications and High Performance Computing 15, 200-222 (2001)
6. Frey, J., Tannenbaum, T., Livney, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. Journal of Cluster Computing 5, 237-246 (2002)
7. http://gridsystems.com/pdf/IGIntro.pdf/
8. Grandison, T., Sloman, M.: A survey of Trust in Internet Applications. IEEE communications Surveys & Tutorials 3, (2000)
9. Chen, E.: Poison Java. IEEE Spectrum 36, 38-43 (1999)
10. Flannagan, D.: Java in a Nutshell. O'Reilly and Associates (2002)
11. http://java.sun.com/
12. Walters, R.J., Crouch, S.: M-grid: Using Ubiquitous Web Technologies to create a Computational Grid. Lecture Notes in Computer Science 3470, 59-67 (2005)
13. Buser, D., Kaufman, J., Llibre, J.T., Francis, B., Sussman, D., Ullman, C., Duckett, J.: Beginning Active Server Pages 3.0. Wiley Publishing Inc (2003)
14. Microsoft Corporation, http://www.microsoft.com
15. Bergsten, H.: JavaServer Pages. O'Reilly and Associates (2003)
16. The Apache Jakarta Tomcat 5.5 Servlet/JSP Container. (2004)