M-grid: Using Ubiquitous Web Technologies to create a Computational Grid

Robert John Walters and Stephen Crouch

Declarative Systems and Software Engineering Group
Department of Electronics and Computer Science
University of Southampton, UK.
SO17 1BJ
{rjw1,stc}@ecs.soton.ac.uk

Abstract. There are many potential users and uses for grid computing. However, the concept of sharing computing resources excites security concerns and, whilst being powerful and flexible, at least for novices, existing systems are complex to install and use. Together these represent a significant barrier to potential users who are interested to see what grid computing can do. This paper describes m-grid, a system for building a computational grid which can accept tasks from any user with access to a web browser and distribute them to almost any machine with access to the internet and manages to do this without the installation of additional software or interfering with existing security arrangements.

1. Introduction

Probably the most widespread application of grid technology is a computational grid [2, 13, 14] which provides a network for the distribution of computational power available from connected machines to users with work to perform. On identifying a task, a user uses local software to present their task to the grid system which then arranges for the task to be executed (usually elsewhere and often distributed amongst a number of machines) and delivers the results back to the user. Existing systems require the installation of software onto the machines which are connected to form the grid.

The grid software [2, 7, 9-11, 13-15] provides the necessary infrastructure for the users to inject their tasks into the system, for the system to distribute tasks to nodes within the grid and to return the results. This software also implements mechanisms to transmit tasks, input data and results from machine to machine as necessary and to execute tasks when they arrive at remote machines. The effort associated with installing and configuring this software is considerable. Aside from the effort associated with installing, configuring and coordinating the software, this can be a problem for two reasons. The first is that, depending on their environment, users may need to obtain permissions to install software onto the machines to be used. Even where this is not a problem, there is a second issue of the risk for those machines which will perform the computations of executing an unknown program which is delivered by the

grid software but whose ultimate source is a third party. Handling these two issues adds considerable complexity to the task of installing a computational grid. M-grid offers a solution which avoids these issues by using software and associated security which are ubiquitous.

The essence of a computational grid is that it provides a mechanism whereby a collection of computers with processing capability is made available to users with a computational tasks to perform. Each system has its own features and strengths, but they all have the following three roles performed by one or more of the machines which form the grid:

- Nodes which provide processing power to the grid.
- Users who supply tasks to be performed
- Coordinator(s) who distribute incoming tasks to the processing nodes and see that results are returned as appropriate.

In a classical computational grid, there will be many nodes providing processing power and relatively few users each of whom has a need for large amounts of processing power to perform either a few large tasks (which are often broken into pieces for execution) or a great many relatively smaller ones. The coordination role is often performed by a single machine. The coordinator is at the heart of the system and has two main activities: collecting tasks and distributing them to available nodes. Discharging these obligations implies a subsidiary task of collecting the results and delivering them to the users.

The coordinating machine(s) need to be able to communicate with others in the grid and, in the case of the nodes providing processing, supply code (and associated data) for execution. This is typically achieved by installing grid software on each of the machines taking part which handles communications between the component parts of the grid. Installing this software can be problematic because of what it does. Owners of computers are concerned to ensure that new software installed will not disrupt, compromise or damage their system. Hence, before they are prepared to install it, they subject it to scrutiny to check that it's behaviour is acceptable. Areas of concern include ensuring that the software won't damage the machine on which it is installed, won't interfere with existing applications which operate on the machine and doesn't seek to destroy data or collect and disclose confidential information. This might be established directly by examination of the software, or indirectly by the certification by some trusted party. The aspect of the grid whereby the coordinator sends code to other nodes for execution is particularly problematic since the nature and detail of the code to be executed is not known at the time the grid infrastructure is being installed. Thus, in accepting code to execute, a node has to rely on the coordinator to only send code which is acceptable. This requires trusting relationships between the node, the coordinator and the client [12]. Establishing these trusting relationships can be both difficult and time consuming.

2. Why is m-grid so easy?

In all of the above, the real nub of the problem is the danger associated with executing code which arrives via the network. These tasks arrive via the coordinator from clients who may not even be known to the processing node. In accepting this work, the processing node has to trust that it can execute the code safely. In order to do this, the node must be prepared to trust in the other elements of the system. In the case of the client, the node has to either trust it directly or be prepared to rely on the coordinator to vet clients and the tasks they supply appropriately. In addition, the node then has to be prepared to trust the coordinator not to interfere with the tasks (such as by adding in its own unacceptable code).

We identified that, although the motivation is quite different, a browser which downloads and runs an applet contained within a web page is doing exactly what causes the trouble in a computational grid - the server sends code which is executed on the client machine in the browser [8, 16]. The enabling technology for this activity was developed to further the capability of the world wide web to provide a rich and comprehensive experience to the user of the browser. It works by permitting the creation of elements in web pages which use local processing to enhance the appearance of web pages and interact with the user. In order to do this, the browser provides, in the form of a specialised Java virtual machine, an execution environment which is carefully crafted to ensure there can be no risk to the local host from the actions of an applet [5, 8, 16]. Because the activity of the applet is constrained and contained, there is no risk to browser or the hosting machine associated with running the code. We realised that we could use the same mechanism to implement the nodes of a computational grid. Any machine with a Java enabled web browser could provide processing capability to a grid system if the tasks were distributed as applets embedded within a web page. Such a grid does not need its own communications infrastructure as it could use the mechanisms already in place for use by the web browser, nor is there any need to establish relationships of trust between the parties involved as the activities of applets are constrained to acceptable limits by the execution environment provided by the browser.

Since virtually any connected machine will have a suitable browser installed, virtually any connected machine worldwide could take part in such a grid as a computational node without any software installation or security implications.

2.1 How m-grid works in outline

M grid uses applet technology to distribute work to client machines. Many of the usual objections and problems associated with joining a grid are avoided because, there is no need to install software on the target machine. M grid uses the existing web browser. Problems of security don't arise either since applets execute within the carefully constrained environment provided by the browser which is designed to protect their hosts from errant behaviour by applets. Hence, there is no need for the processing node to establish a trusting relationship with the supplier of the task or its ultimate source since the "sandbox" in which the applet executes protects the host from errant behaviour by an applet (even if it is malicious).

The only code required to create m-grid is in the provision of the coordinator role which supplies the mechanism for clients to inject their tasks into the system, distributes them to the nodes and returns the results. This is achieved using two web pages:

- A job distribution page. Anyone willing to supply processing capability to the grid does so by opening the job distribution page in a java enabled browser.
- A job submission page. Those with tasks to be performed use a form on this page to upload them and await the results.

3. Setting up m-grid

Since all that is required to become a processing node of m grid is to open the job request page in a browser and clients need only to submit their tasks to the job submission page, there is no setup for these beyond directing a browser to the appropriate web page. This is unlikely to be difficult for even the most naïve of computer users. Such setup effort as there is for m grid is in the creation of the coordinator web pages and their publication for which a machine running suitable web server software is required.

Today, most people who might be interested to setup their own m grid will have access to a suitable web server and resources which would enable them to construct and maintain simple web pages, even if they don't do it themselves. The only complexity in the m grid web pages is the code embedded within them which elicits tasks from clients and distributes them to the nodes. The manner in which this code is written is dependent on the capabilities of the web server software being used. For our first implementation, we have used Active Server Pages (ASP) which is provided by Microsoft in their Internet Information Server (IIS) [4].

However, we have already generated suitable code so the task of generating these pages may be reduced to pasting the ASP code into an appropriate existing web page, or adding any desired presentation to our "vanilla" m grid files. Creating these pages this way requires no more than the most elementary skills in web page creation.

With the web pages in place, the m grid can commence operation as soon as the first clients and nodes access the pages.

4. Joining and using m-grid

Since there is no software to install and no security or trust relationships to establish between the various elements of m grid, joining is trivial. Almost any machine with a web browser able to view the m grid web pages can take part.

Becoming a node offering to perform processing couldn't be easier. All that is required is to navigate a Java enabled browser to the job distribution web page and leave it. Any browser able to host an applet is suitable regardless of the underlying hardware or operating system. The page is created dynamically and displays some information about m grid. From time to time the server will insert an applet into the page which has a computational task encoded within it. Once complete the applet re-

turns its results (and disappears). There is nothing for the machine owner/operator to do. To remove the machine from the grid, all that is necessary is to close the browser or redirect it to another page.
The user with a job to be processed by m grid only needs a browser able to access the form on the job submission page. Once a job has been uploaded to the server using this form, m grid passes the task out as an applet to an available node. The results are collected and returned to the user in due course. An enhancement which we are already implementing is to permit the user uploading a job to leave the job submission page as soon as the upload is completed instead of waiting for the outcome. They can then return to the job submission page later and look up the result of their computation.

5. M grid in more detail

Since the clients and computational nodes in m grid use nothing more than standard features available in popular web browsers, there is little to say about these elements of the system. However, there are a number of considerations concerning the coordinating role.
The coordinating role runs on a machine which hosts the m grid web pages. The server needs to collect and store jobs pending distribution to a node, arrange for each job to be sent (in due course) to just one node and to manage the results that the nodes return. This can only be achieved with a server which has the capability to generate dynamic content in the pages it supplies. These types of feature are typically offered as a scripting language and are available from all but the most minimal web servers [3, 4, 6]. Any of these could be used to create the m grid pages. There is no one dominant technology which is universally available. Therefore, we shall need to develop different versions of the m grid coordinating pages for each of the competing systems. For the first implementation, have used ASP which is available on Microsoft web servers (Internet Information Server, IIS and Personal Web Server, PWS). One of these servers is either included with or available as a free download for most recent versions of Windows. We felt that few potential users of m grid would have difficulty gaining access to a machine running an ASP enabled web server.
In due course, we shall implement further versions using alternative technologies such as JSP/Tomcat [1, 3] so that the m grid coordinator pages may be hosted on other popular web serving software.
As indicated above, the coordinating role in m grid comprises a small number of web pages published using a web server on the coordinating machine. Computational nodes set a browser to view the jobRequest page and receive their jobs as applets embedded in this page. The coordinating role achieves its objective of getting each job it receives executed by manipulating the contents of the jobRequest page so that each job is distributed to just one of the available nodes. In order to do this, the applet element to the jobRequest page is generated dynamically by the server each time it receives a request for the page.

The lifecycle of a job by m grid has four visible stages: the two halves of the interaction with the job submission page by the client and the two halves of the interaction with the job execution page at the processing node:

1. The client submits a job via jobSubmit page.

2. The client (eventually) collects results using the jobCollectResults page.

3. A node requests a job via jobRequest page.

4. The node submits the results from a job via the jobSubmitResults page.

Let us first examine how a task is submitted to m-grid. This is depicted in Figure 1. The client with a task which they require evaluated directs their browser to the jobSubmit.asp page where they insert the details required into the form, including the location of their applet class file. On pressing the upload button the task (encapsulated in an applet class file, see later) and associated details are uploaded to the server and stored in a collection of tasks awaiting processing.

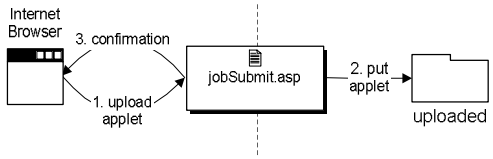

Fig. 1. Submitting an applet job to m-grid

The second part of the client's interaction with the server is to collect their results. In the prototype implementation of m grid, the user had to wait until computation was completed at which point the output was displayed in a box on the jobSubmit page. This has been amended so that the user need no longer wait for the outcome of their calculation, although they may if they wish. A returning user wishing to collect results is presented with a list of completed tasks. Clicking on any of these displays their output. This operation is shown in Figure 2. In order to keep m grid as simple as possible, it has no security. Anyone accessing the system would be able to view the results of any task completed by the system. (Of course a user concerned to keep their work secret could submit tasks which encrypt their results.)

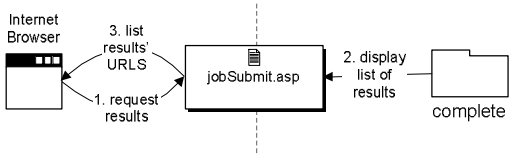

Fig. 2. Collecting results from a task

The remaining two visible pieces to the job's lifecycle concern the distribution of the job to a processing node and the subsequent return of the results to the coordinator. The first of these operations is illustrated in Figure 3. Along with some static content,

the jobRequest.asp page includes some content which is generated dynamically. In the absence of outstanding jobs, this dynamic element is a simple message to say there are no jobs available. If there is a task waiting, it is inserted into the page as an applet. The task is then moved from the collection of uploaded tasks to a collection of tasks in progress.
Once the applet has completed, its results are uploaded to the server (using another ASP page) as illustrated in Figure 4. The task and its results are stored in a collection of completed tasks.
A feature of the jobRequest page is that it causes browsers to refresh the page periodically to check for new tasks so long as they are not executing an applet.

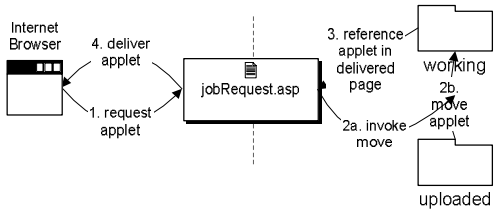

Fig. 3. Obtaining a task to perform

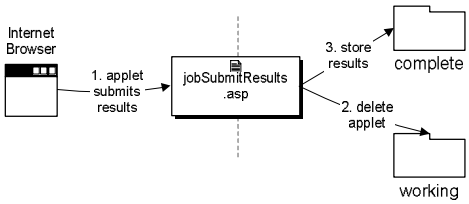

Fig. 4. Returning results

6. Issues and Future work

M grid has limitations. Some are a necessary consequence of the way that m grid operates but we plan to address others. For example, since m grid distributes tasks to nodes as applets, these tasks are restricted by the constraints which apply to applets which include being forbidden to use the local file system or to establish communications other than with the host from which the applet was downloaded.
Other limitations arise from our present implementation. One is that, with its current reliance on ASP in the coordinator web pages means that these need to be hosted on a Microsoft web server. Since Microsoft software is so widely available, we felt using ASP for the coordinator role is not a severe restriction. Nevertheless, we intend to create alternative versions of these pages which are suitable for deployment on other widely available web servers.
A further concern is the constraints which m grid places on the tasks which clients can upload to m grid. In its present form, a task to be executed by m grid must be up-

loaded as an applet contained in a single class file and the applet must include some m grid specific code (associated with the return of results). This requires the potential user to be able to embed their task into an applet along with the m grid specific code. This sounds like a considerable imposition onto the user, but in fact the users of any computational grid system have to submit their tasks in the form of a program compatible with the grid framework.

We are already looking investigating how we can eliminate the need for the m grid specific code by either recompiling user supplied code after applying a source code transformation or providing a wrapper around the users' code. Of the restrictions to what our users are permitted to do, these arise from the nature of applets and the environment in which they execute. For example, applets are forbidden access to the local file system. Whilst some of these are inconvenient to clients creating tasks, they cannot easily be relaxed. They represent the price we pay for being able to dispatch our jobs to other machines without formality.

One final issue which we have yet to investigate fully is performance. The most obvious reason for a user to pass a job on to a computational grid is because they wish to take advantage of the processing power available from the grid. It seems reasonable to expect that the execution environment in which applets operate, in which their actions are constantly monitored to ensure they don't abuse their privilege of execution by breaching the conditions on which it is granted, would place a job executing as an applet at a performance disadvantage compared with an unrestrained application running in an open environment. However, our results to date suggest that this performance penalty is modest and quite acceptable in this context.

7. Conclusion

There is increasing interest in grid computing and there are many potential users of these technologies. However, existing grid software requires considerable installation and configuration effort, including placing software onto each machine which is to take part in the grid. Aside from any difficulty which may arise in the installation and configuration of the machines to be used, this raises issues in many environments (even if the software is to be installed on dedicated machines) and permissions need to be obtained before installation can commence.

Issues of security are usually addressed by the use of certification and trust techniques. These are crafted to ensure that so far as is possible, malicious tasks are not accepted, and assure system owners that they will not suffer harm by joining the grid and executing jobs. They also attempt to permit identification of responsibility if things go wrong. However, not only is establishing these trusting relationships complex and time consuming to set up, ultimately they still require machine owners to accept a degree of risk. By using applets, which execute in a protected environment within a web browser, m grid is able to operate without needing any of the usual software and security infrastructure.

M grid provides a means for potential uses to experiment with grid technologies by allowing them to set up a computational grid quickly and easily with an absolute minimum of formality, software installation and time and effort from users. It re-

quires nothing in the way of software installation on the computational nodes or the clients and on the coordinating machine all that is needed is a web server and permission to publish web pages. Of course there is a price to pay for this convenience and this comes in the form of restrictions on the tasks which m grid can perform to those which an applet can perform when executing within the controlled environment provided by a web browser.

8. References

1. The Apache Jakarta Tomcat 5.5 Servlet/JSP Container. See: http://jakarta.apache.org /tomcat/tomcat-5.5-doc/index.html (2004)
2. Altair Engineering Inc.: Portable Batch System. See: http://www.openpbs.org/ (2004)
3. Bergsten, H.: JavaServer Pages. O'Reilly and Associates (2003)
4. Buser, D., Kauffman, J., Llibre, J.T., Francis, B., Sussman, D., Ullman, C., Duckett, J.: Beginning Active Server Pages 3.0. Wiley Publishing, Inc. (2003)
5. Chen, E.: Poison Java. IEEE Spectrum. Vol. 36 (1999) 38-43
6. Converse, T.: PHP and MySQL Bible. John Wiley & Sons Inc (2004)
7. Erwin, D.W., Snelling, D.F.: UNICORE: A Grid Computing Environment. Lecture Notes in Computer Science, Vol. 2150. Springer-Verlag (2001) 825-839
8. Flanagan, D.: Java in a Nutshell. O'Reilly and Associates, (2002).
9. Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. Int. J. Supercomputer Applications and High Performance Computing, Vol. 11 (1997) 115-128
10. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scaleable Virtual Organization. Int J Supercomputer Applications and High Performance Computing, Vol. 15 (2001) 200-222
11. Frey, J., Tannenbaum, T,. Livney, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. J. Cluster Computing, Vol. 5 (2002) 237-246
12. Grandison, T., Sloman, M.: A survey of Trust in Internet Applications. IEEE Communications Surveys & Tutorials Vol. 3 (2000)
13. Litzkow, M., Livny, M.: Experience with the Condor Distributed Batch System. IEEE Workshop on Experimental Distributed Systems, Huntsville, AL (1990)
14. Livny, M., Basney, J., Raman, R., Tannenbaum, T.: Mechanisms for High Throughput Computing. SPEEDUP Journal Vol. 11 (1997) 36-40
15. Gridsystems S.A.: Overview to InnerGrid. See: http://www.gridsystems.com /pdf/IGIntro.pdf (2003)
16. Sun Microsystems, Inc.: Sun Microsystems, Java Technology. See: http://java.sun.com/ (2004)