Teaching the Grid:
Learning Distributed Computing with the M-grid Framework

Robert J. Walters, David E. Millard, Philip Bennett,
David Argles, Stephen Crouch, Lester Gilbert, Gary Wills

School of Electronics and Computer Science,
University of Southampton,
Southampton, UK
{rjw,dem,pb903,da,stc,lg3,gbw}@ecs.soton.ac.uk

Abstract: A classic challenge within Computer Science is to distribute data and processes so as to take advantage of multiple computers tackling a single problem in a simultaneous and coordinated way. This situation arises in a number of different scenarios, including Grid computing which is a secure, service-based architecture for tackling massively parallel problems and creating virtual organizations. Although the Grid seems destined to be an important part of the future computing landscape, it is very difficult to learn how to use as real Grid software requires extensive setting up and complex security processes. M-grid mimics the core features of the Grid, in a much simpler way, enabling the rapid prototyping of distributed applications. We describe m-grid and explore how it may be used to teach foundation Grid computing skills at the Higher Education level and report some of our experiences of deploying it as an exercise within a programming course.

Introduction

Distributed computing is a classic challenge within computer science. The objective is to coordinate a number of distributed processes so they function as one system. In recent times this has been described as a Grid of computers, and there has been a technological trend towards a service-based architecture known as the Grid.

Learning about Grid technology, and the principles of distributed computing, is challenging due to the difficulty of demonstrating the principles in a way meaningful to undergraduate students. One application of Grid technologies which we would like our students to understand and be able to exploit is a computational Grid (Altair Engineering Inc, 2004; Litzkow & Livny, 1990; Livney, Basney, Raman, & Tannenbaum, 1997) whereby a computationally intensive task is distributed around a network of machines. Such a system accepts tasks from (outside) users, distributes them to available machines for processing, collects the results, and passes these back to the users. At the heart of such a system is a mechanism whereby tasks arriving from users can be distributed to and executed on other machines. In most Grid systems (Altair Engineering Inc, 2004; Erwin & Snelling, 2001; Foster, Kesselman, & Tuecke, 2001; Frey, Tannenbaum, Livney, Foster, & Tuecke, 2002; Gridsystems SA, 2003; Litzkow & Livny, 1990; Livney, Basney, Raman, & Tannenbaum, 1997), this is achieved using software installed on all of the machines which form the Grid. This software is crafted to ensure the system has the very best performance and security but the associated cost is that the software system is extensive and requires considerable configuration. This is not a problem where the Grid is being set up by experts using dedicated hardware but it can be an insurmountable barrier for the interested potential user. One particular problem with any implementation of a computational Grid is that the machines performing the computation receive the code from a remote machine and this is dangerous. This is true even when there is no malicious intent. It is entirely possible for unintentional flaws in the incoming software to cause real problems for the machine on which it is executed. Established Grid systems have extensive security infrastructures associated with them to ameliorate this problem.

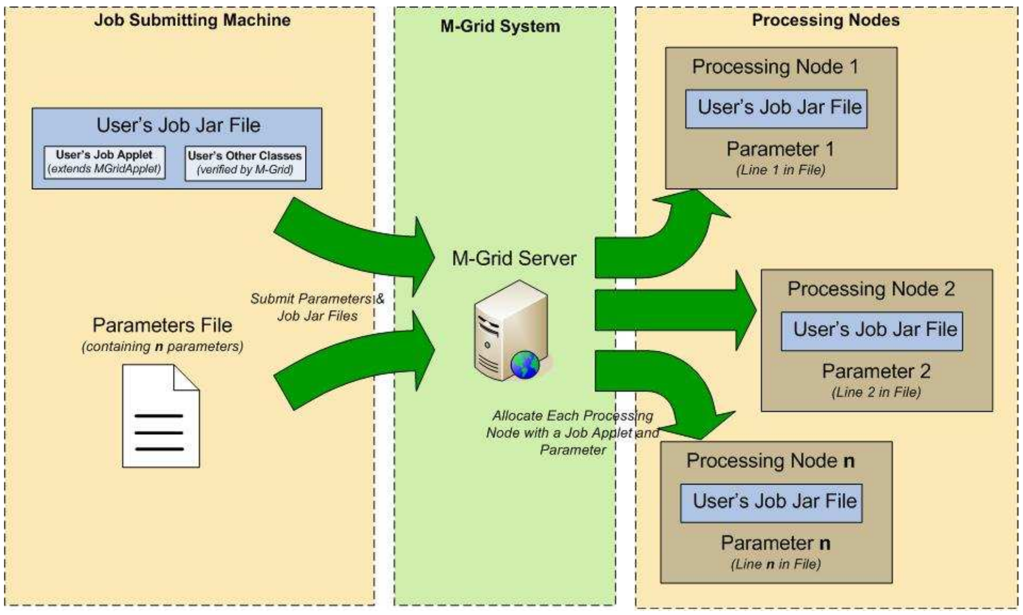

Job Submitting Machine
M-Grid System
Processing Nodes
User's Job Jar File
User's Job Applet (extends MGridApplet)
User's Other Classes (verified by M-Grid)
Parameters File (containing n parameters)
Submit Parameters & Job Jar Files
M-Grid Server
Allocate Each Processing Node with a Job Applet and Parameter
Processing Node 1
User's Job Jar File
Parameter 1 (Line 1 in File)
Processing Node 2
User's Job Jar File
Parameter 2 (Line 2 in File)
Processing Node n
User's Job Jar File
Parameter n (Line n in File)

Figure 1: an overview of m-grid

We have found that students find thinking and reasoning about distributed systems extremely difficult. The complexity and security which accompanies all of the established Grid systems serves to cause significant installation and configuration overhead and, more importantly, obscure the issues we wish them to learn about even further.

Background

The Grid is a relatively new technology, hence there is relatively little published work on teaching best practice. To address this the ACM Special Interest Group on Computer Science Education has initiated a project called GridForce to aid the dissemination material required for Grid technology education (Kumar, Shumba, Ramamurthy, & D'Antonio, 2005).

However, the Grid has emerged out of the Parallel and Distributed computing community. Teachers trying to teach distributed computing in the early 1990's faced similar problems to those teachers now trying to teach Grid computing, Stewart suggests that the main reason why distributed computing was originally confined to graduate courses, was that such a course '*requires expensive hardware and the very latest software development tools*' (Stewart, 1994).

A number of tools were built for teaching distributed computing. Ben-Ari and Silverman describe a client application that uses an integrated GUI environment in which students can develop distributed programmes, called DPLab (Ben-Ari & Silverman, 1999). The focus is slightly different from m-grid in that DPLab primarily focuses on implementing and teaching high-level algorithms (e.g., the Byzantine Generals), rather than an overall systems approach. Bynum and Camp developed a programming language that allow the students to simulate concurrent (parallel) programming and also found that in order to assist the students in learning these difficult concepts it was necessary to do this through practical experience (Bynum & Camp, 1996). Cunha and Lourenço report on teaching a course on Parallel and Distributed processing, to undergraduates (Cunha & Lourenço, 1998). The course was taught for a number of years, they suggest that an essential element of the approach is the emphasis on having a tightly coupled interplay between the practical implementation of the theoretical abstractions, and the use of an effective platform to support the implementation.

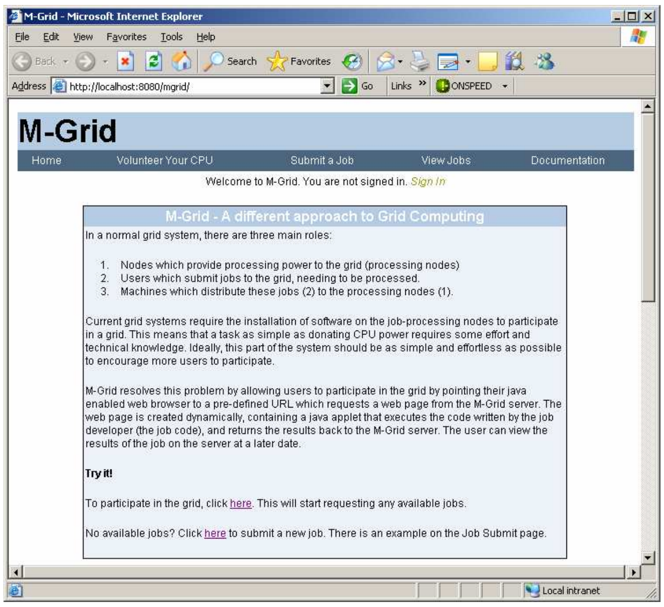
The m-grid framework described in this paper provides a practical environment for students to learn about Grid technology and distributed computing using widely available and familiar software.

M-grid

In order to create a computational Grid, we need to find a way to execute code on remote machines without all of the usual security implications. In fact, although it was never intended to be used in this way, a Java applet running within a web browser provides a suitable mechanism. All of the well known browsers implement a “sandbox” in which to run applets which are downloaded as part of web pages. The purpose of the sandbox is to contain and constrain the actions of applets to such an extent that they are prevented from causing damage to the host machine, even if they try. As a result, downloading and running an applet is not considered to be dangerous. The idea that lead to the creation of m-grid is the realization that, provided the task can be encoded into an applet suitable for execution inside the sandbox of a web browser, all that is needed for the task to be executed on a remote machine was a Java enabled browser directed at a web page in which the task is embedded as an applet.

Figure 1 shows an overview of m-grid. The complete system comprises three types of element: users who interact with the central system using a web browser to upload their task, the coordinating system which distributes tasks to nodes and marshals the results, and the nodes which perform the actual computations. A proof of concept system was built using ASP (Buser et al., 2003; Walters & Crouch, 2005). More recently, the system has been redeveloped using JSP ("The Apache Jakarta Tomcat 5.5 Servlet/JSP Container", 2004; Bergsten, 2003).

A new user’s first contact with m-grid will be its homepage (Figure 2) which provides information about how to use the system and links to the important pages of the system. If the user wishes to create a processing node, they follow the “Volunteer your processor” link to a page which is created dynamically by the system. If m-grid has work awaiting allocation to a processing node, the page will include a visible applet. This applet has a presence on the page in which the machine owner can follow the progress of the calculation. When the system is free, the system generates a placeholder page until the node is needed.
Figure 3 shows an example node page containing an applet.


M-Grid - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/mgrid/ Go Links ONSPEED
M-Grid
Home Volunteer Your CPU Submit a Job View Jobs Documentation
Local intranet

Welcome to M-Grid. You are not signed in. *Sign In*

M-Grid - A different approach to Grid Computing

In a normal grid system, there are three main roles:

1. Nodes which provide processing power to the grid (processing nodes)
2. Users which submit jobs to the grid, needing to be processed.
3. Machines which distribute these jobs (2) to the processing nodes (1).

Current grid systems require the installation of software on the job-processing nodes to participate in a grid. This means that a task as simple as donating CPU power requires some effort and technical knowledge. Ideally, this part of the system should be as simple and effortless as possible to encourage more users to participate.

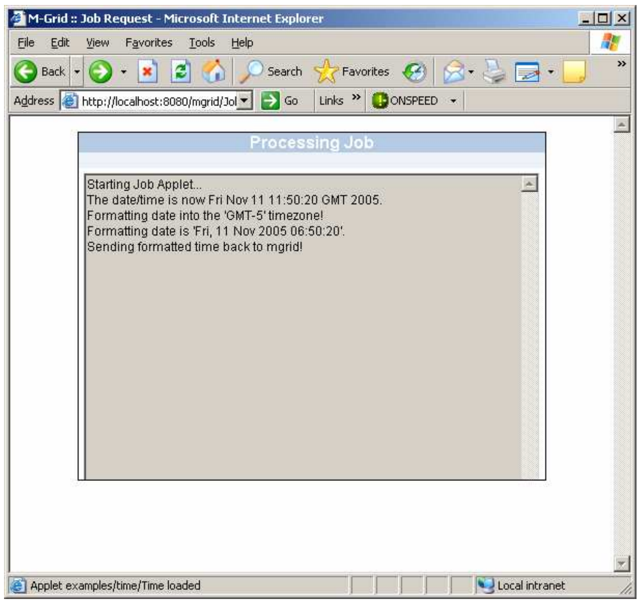
M-Grid resolves this problem by allowing users to participate in the grid by pointing their java enabled web browser to a pre-defined URL which requests a web page from the M-Grid server. The web page is created dynamically, containing a java applet that executes the code written by the job developer (the job code), and returns the results back to the M-Grid server. The user can view the results of the job on the server at a later date.

Try it!

To participate in the grid, click here. This will start requesting any available jobs.

No available jobs? Click here to submit a new job. There is an example on the Job Submit page.

Figure 2: The m-grid homepage

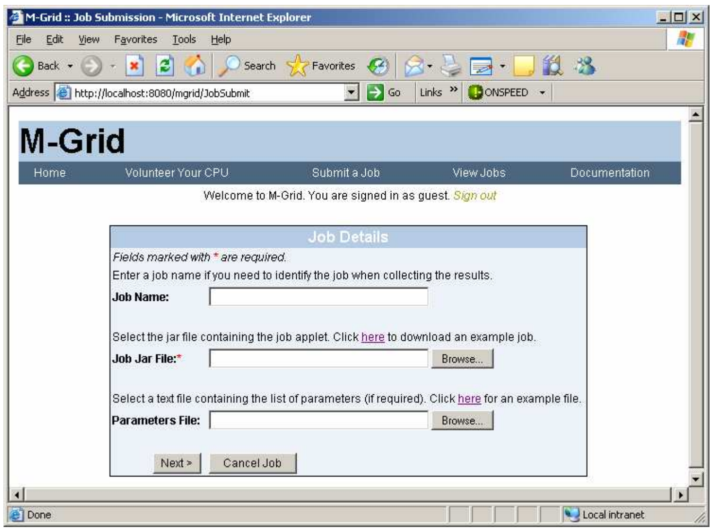

M-Grid :: Job Request - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/mgrid/Jol Go Links ONSPEED
Processing Job
Applet examples/time/Time loaded
Local intranet

```
Starting Job Applet...
The date/time is now Fri Nov 11 11:50:20 GMT 2005.
Formatting date into the 'GMT-5' timezone!
Formatting date is 'Fri, 11 Nov 2005 06:50:20'.
Sending formatted time back to mgrid!
```

Figure 3: An m-grid applet executing

The other two important pages in the system are the job submission page (Figure 4) where users are guided through submitting a job to m-grid and the results page (Figure 5) where they can view and download the results. This revised implementation of m-grid incorporates improvements, for example:

1) Users are permitted to submit tasks which require more than one JAVA class and/or additional resources.
2) The creation of the applet code for submission to m-grid is smoothed by the provision of the MGridApplet class which permits users to develop their code locally using Appletviewer (or similar) and then submit their code to m-grid unchanged.
3) M-grid is robust against applets which throw uncaught exceptions or never complete their tasks. Details of uncaught exceptions are returned to users in place of results. Unfinished tasks are (eventually) allocated to further nodes.


M-Grid :: Job Submission - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/mgrid/JobSubmit Go Links ONSPEED
M-Grid
Home
Volunteer Your CPU
Submit a Job
View Jobs
Documentation
Job Details
Browse...
Browse...
Next >
Cancel Job
Done
Local intranet

Welcome to M-Grid. You are signed in as guest. Sign out

*Fields marked with * are required.*

Enter a job name if you need to identify the job when collecting the results.

Job Name:

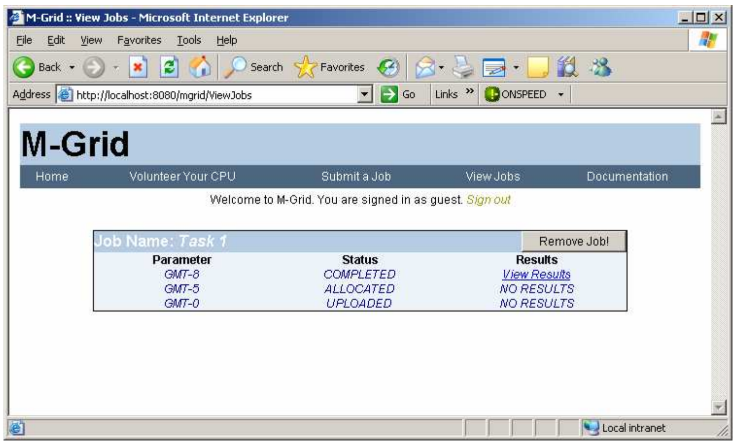
Select the jar file containing the job applet. Click here to download an example job.

Job Jar File:*

Select a text file containing the list of parameters (if required). Click here for an example file.

Parameters File:

Figure 4: The job submission page


M-Grid :: View Jobs - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/mgrid/ViewJobs Go Links ONSPEED
M-Grid
Home Volunteer Your CPU Submit a Job View Jobs Documentation
Remove Job!
Local intranet

Welcome to M-Grid. You are signed in as guest. *Sign out*

Job Name: *Task 1*

Parameter	Status	Results
GMT-8	*COMPLETED*	*View Results*
GMT-5	*ALLOCATED*	*NO RESULTS*
GMT-0	*UPLOADED*	*NO RESULTS*

Figure 5: The results page

M-grid as a Teaching Tool

Students of computing are required to understand the basic principles of distributed systems. This includes their advantages, their potential pitfalls , and the complexity of breaking a single algorithm into separate parts that work together towards a shared goal. In our experience it seems that students find it difficult to visualize the system as a whole and also find it hard to relate to the context of individual distributed processes.

One of the issues with teaching distributed systems, especially Grid based systems, is that it is often impossible to show students a real example or allow them to experiment with it. This is because the tools for writing the software are very complex, and the middleware required can have prohibitively high overheads in terms of security authorization and system set-up and configuration.

We have attempted to use m-grid to get around these problems of complexity and allow students to experience genuine distribution. We believe that m-grid could be used in a teaching scenario in three different ways:

- **As a Demonstrator** – M-grid can be used to show students a distributed system in action. Because of the simple infrastructure (a web browser) it is possible for students to participate directly in the demonstration by volunteering their laptops as m-grid nodes. Thus the students see jobs submitted to the central server and then distributed across a classroom. As m-grid returns the results of each node separately it is also possible for them to directly see the consequences of the various approaches which might be taken to breaking the problem up.
- **To Explore Algorithm Distribution** – Not all students have the expertise (or the time within a particular module) to write distributable program code (even in m-grid), but these students may still benefit from understanding the consequences of dividing up a problem space for distribution. M-grid's parameter files support these types of students, and allow them to experiment with different numbers of nodes and configurations without having to write new applets.
- **To Code Distributed Programs** – Those students who are capable of writing their own Java applets can create programs in the m-grid framework to solve any appropriate problem in a distributed way. This would probably take more time than the other options, but would provide a rare hands-on activity. It has benefits over other distributed frameworks in that it is simulating a real world scenario, and therefore should have more credibility with students than other, more forced or artificial approaches.

M-grid used as a demonstrator works well in a lecture or seminar scenario. Exploring algorithm distribution would make an effective lab session, and coding a distributed program would be an appropriately challenging activity for self-study or coursework.

In the next section we will describe some of our early experiences with deploying m-grid as a demonstrator, and as a code framework for teaching.

Experiences of Deploying m-grid

At the University of Southampton, we have a number of modules within our undergraduate computing courses that touch on distributed programming. We have successfully used m-grid as a demonstrator within lectures and seminars in these courses. M-grid is distributed with a small number of sample applets and although these are very simple, purely by running they demonstrate the framework in operation. The audience can watch the submission of the applet and the eventual production of the results on the main lecture theatre projector. They can also see the applet instances appear and perform their work on the nodes (machines belonging to members of the audience).

More recently we have tried to use m-grid as way of allowing students to write distributed code. The course in which we did this is a first year programming principles course based on the Java language. The course is self streamed, with the top flight students opting out of the main lecture series and choosing to attend a workshop series where they are set weekly challenges (Jenkins & Davy, 2000). These challenges are designed to stretch the students' java knowledge and introduce them to particular programming challenges, such as GUI programming, language design, artificial intelligence, and distributed programming.

We chose to set the upper stream a challenge that involved writing an applet for the m-grid framework. Our intention was they would learn something about designing algorithms for distributed systems, and also gain an appreciation of the trade-off between increased processing power and network overhead.

We had a number of learning and teaching issues with setting the challenge:

- Choosing an appropriate problem to solve - we need a problem that lends itself to being broken up into chunks that can be tackled simultaneously and it must be a problem that can be understood by first-year University students.
- Choosing an engaging problem - the upper stream challenges are entirely voluntary, and the only reason for students to tackle them is if they have some sort of "hook" that has relevance and captures their interest and imagination.
- The difficulty of distributing data with applets - m-grid uses applets to send processes to its nodes. This means that those processes are subject to all the restrictions of an applet. Probably the most awkward of these is the fact that applets can only create network connections to their home server, which in m-grid's case is the m-grid coordinator. This makes it difficult to write distributed programs that operate on large data sets, as the data must be wrapped up in the applet *jar* file.
- Issues with amalgamating results programmatically - m-grid returns the results from each node separately. This means that the student uploading the applet has to gather and make sense of the results. This restricts the type of problem that can be given out, as the results must be reasonably simple to recombine.

Given these considerations we set a challenge which operated on a data set small enough to be distributed in the jar file of each applet. After a brief introduction to the m-grid framework, the students were given the following challenge:

> *"For this challenge I'd like you to write an applet that analyses an image. There is a simple version in the development kit called PixelCount, which counts black and white pixels in an image. The image needs to be in the jar file for PixelCount to find it when it's running via the m-grid server (although it should see it in the local directory if you run it via appletviewer). Try creating your own version:*
>
> - *That does a frequency count on all the colours*
> - *That runs a more complex statistical analysis (mode, median, mean, standard dev, etc)*
> - *If you are feeling brave how about trying to use some form of Hough Transform to find features (lines or circles) within the image?*

The m-grid system returns results separately from each copy of your applet so you will have to combine them externally."

The workshop series has too few students to draw any quantative conclusions about our m-grid deployment, but we did generate several comments about the m-grid framework that will inform its future development.

Conclusions and Future Work

Our experiences of deploying m-grid for teaching have led us to a number of observations about the current framework. In particular there a number of restrictions (due to the framework design and use of applets) that reduces the types of problem that can be addressed and therefore the potential learning activities.

Problem	**Potential Solution**
The system requires users to submit applets manually for distribution, and then manually combine their results. This makes it difficult to create chains of processes, as there always needs to be a human element to stitch m-grid applets' inputs and outputs together.	To allow automatic inputs we need to create some means for tasks (applets) to be automatically added to the m-grid framework. Theoretically this could be achieved using the existing web interface, but a Java RMI call to the m-grid server, or even a Web Service interface, would make this much simpler. To allow automatic compilation of results we need to create an API on the server that allows results to be queried automatically. Alternatively we could use callback functions or return values from the automatic task API to send results back to the invoking software.
Applet restrictions mean that it is impossible for applets to access large data stores beyond the servers from which the applets were served. This makes it impractical to create distributed tasks that operate on large data sets, as the entire data sets must be distributed with each applet.	It may be necessary to add some kind of data service to the m-grid server to allow m-grid applets to access data via some proxy system. This could be as simple as proxying URL requests, or could be some more sophisticated system that automatically breaks up data within jar files submitted to the server, and sends only relevant parts to the m-grid nodes, perhaps according to some parameters sent with the submitted applet.
Although the m-grid system does employ a username/password login security feature, it does not illustrate to students the more core security issues addressed by existing Grid infrastructures; in particular mutual authentication and encryption.	This could be achieved by developing integrated, automated security support via the use of PKI (Public Key Infrastructure) certification to handle mutual authentication, and the adoption of https as the transport protocol to address encryption. If done correctly this would have the added benefit of illustrating security at an easily approachable level, but also allowing more advanced demonstration by the configuration of those security features on the browser and the web server.

We are also currently planning a larger m-grid teaching activity with third year students. It is our intention to set an activity such that it will be possible to evaluate the effectiveness of m-grid for conveying the principles of distribution.

Distributed computing is a difficult topic to teach. Because of the sophistication of its operation, and of the complexity of the middleware systems available, it is prohibitively difficult to demonstrate, let alone use in a student centered activity within the context of an undergraduate laboratory exercise. The m-grid framework makes it possible for students to experience a distributed system with minimum overhead, as it uses java applets as nodes, and a web server as a central broker. In this paper we have explained how we believe m-grid could be used for learning and teaching. We have described our experiences so far and used these to critique the existing framework as a teaching aid. We have concluded that while it does make distributed systems more accessible it also limits the type of activity that can be distributed, and have suggested some directions for future work that will address this.

M-grid is a simple tool that makes complex problems more accessible. By using m-grid it is possible to expose students to real examples of distributed computing that are analogous to cutting edge architectures such as the Grid; its Java/Web approach allows students to experience, modify and create genuine distributed systems.

References

Altair Engineering Inc. (2004). Portable Batch System.

The Apache Jakarta Tomcat 5.5 Servlet/JSP Container. (2004). from http://jakarta.apache.org/tomcat/tomcat-5.5-doc/index.html

Ben-Ari, M., & Silverman, S. (1999). *DPLab: an environment for distributed programming.* Paper presented at the 4th annual SIGCSE/SIGCUE ITiCSE conference on innovation and technology in computer science education, Cracow, Poland.

Bergsten, H. (2003). *JavaServer Pages*: O'Reilly and Associates.

Buser, D., Kauffman, J., Llibre, J. T., Francis, B., Sussman, D., Ullman, C., et al. (2003). *Beginning Active Server Pages 3.0*. Indianapolis: Wiley Publishing, Inc.

Bynum, B., & Camp, I. (1996). *After you, Alfonse: A mutual exclusion toolkit.* Paper presented at the Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education, Philadelphia.

Cunha, J., & Lourenço, J. (1998). *An integrated course on parallel and distributed processing.* Paper presented at the Twenty-ninth SIGCSE technical symposium on computer science education, Atlanta, Georgia.

Erwin, D. W., & Snelling, D. F. (2001). UNICORE: A Grid Computing Environment. *Lecture Notes in Computer Science, 2150*, 825-839.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scaleable Virtual Organization. *International Journal of Supercomputer Applications and High Performance Computing, 15*(3), 200-222.

Frey, J., Tannenbaum, T., Livney, M., Foster, I., & Tuecke, S. (2002). Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Journal of Cluster Computing, 5*, 237-246.

Gridsystems SA. (2003). Overview to InnerGrid. from http://gridsystems.com/pdf/IGIntro.pdf/

Jenkins, T., & Davy, J. (2000, 23rd -25th August). *Dealing With Diversity in Introductory Programming.* Paper presented at the LTSN-ICS 1st Annual Conference, Heriot-Watt University, Edinburgh.

Kumar, A. N., Shumba, R. K., Ramamurthy, b., & D'Antonio, L. (2005). *Emerging areas in computer science education.* Paper presented at the 36th SIGCSE technical symposium on Computer science education, Baltimore.

Litzkow, M., & Livny, M. (1990). *Experience with the Condor Distributed Batch System.* Paper presented at the IEEE Workshop on Experimental Distributed Systems, Huntsville, AL.

Livney, M., Basney, J., Raman, R., & Tannenbaum, T. (1997). Mechanisms for High Throughput Computing. *SPEEDUP Journal, 11*, 36-40.

Stewart, C. (1994). Distributed systems in the undergraduate curriculum. *ACM SIGCSE Bulletin, 26*(4), 17 - 20.

Walters, R. J., & Crouch, S. (2005). M-Grid: Using Ubiquitous Web Technologies to Create a Computational Grid. *Lecture Notes in Computer Science, 3470*, 59-67.