A Geographically-Distributed, Assignment-Structured Undergraduate Grid Computing Course

Mark A. Holliday, Barry Wilkinson, Jeffrey House, and Samir Daoud
Department of Mathematics and Computer Science
Western Carolina University
Cullowhee, NC 28723
+1 (828) 227-3951
{holliday, abw, jhouse, sdaoud}@cs.wcu.edu

Clayton Ferner
Department of Computer Science
University of North Carolina at Wilmington
601 South College Road
Wilmington, NC 28403
+1 (910) 962-7129
cferner@uncw.edu

ABSTRACT

Grid computing is now mature enough and important enough to be studied as a full course at the undergraduate level for upper-level computer science majors. We have developed such a course, including a set of lecture slides, assignments, and assignment handouts specifically targeted for this audience. The sequence of assignments is a key part of the course. Some of the assignments are modifications of pre-existing work and others are completely new. We describe the key decisions we made about the course organization and content and describe the assignments. An important feature of the course is that it was geographically distributed with copies of the grid software installed at three campuses. Those campuses plus three others were receiving sites and included students and faculty associated with nine universities.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *client/server, distributed applications, distributed databases, network operating systems.*

General Terms

Design, Experimentation, Security, Standardization.

Keywords

Grid Computing, Assignments, Globus, Web Services, Grid Services.

1. INTRODUCTION

Grid computing is an approach to distributed computing over the Internet that uses open standards [2]. The goal is to virtualize the resources at geographically distributed and often heterogeneous sites. Such resources can include computational devices, data storage, and instrumentation and sensors. Grid computing should lead to more efficient use of these resources by allowing remote and transparent access. Increasing Internet bandwidth and the recent development of key open standards are making the development of production grids increasingly feasible.

We believe that the field of grid computing has matured to the point where a course aimed at the undergraduate, upper-level computer science major is both feasible and desirable. Graduate level courses on grid computing which consist of a seminar reading and presenting a series of papers in the area have been ongoing at a number of universities. However, a true undergraduate course should be different. Such a course should provide a coherent set of lectures that describe the organization of the software that implements grid computing. Just as importantly, such a course should provide a series of hands-on programming assignments so that the students can directly use the key components of the grid infrastructure and develop their own grid-enabled applications.

We have been working over the last year to develop such a course. We received funding from the University of North Carolina Office of the President [9, 10] and the National Science Foundation [11] to support this effort. The first offering of the course is ending at the time of this writing. Undergraduate, grid computing courses for computer science majors are still quite novel. In fact, we chose to offer the course to other universities in North Carolina over the state-supported tele-conferencing network. The response was larger than we expected.

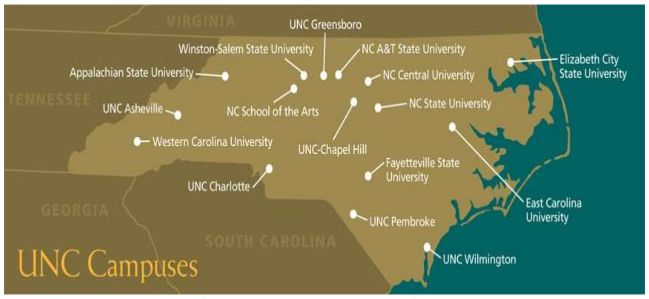

Figure 1: A map of all the University of North Carolina (UNC) campuses. The course originated at the Western Carolina University campus of UNC and was received at Appalachian State University, NC State University, UNC Asheville, UNC Greensboro, UNC Wilmington. In addition, NC Central University, Cape Fear Community College and the private institution Elon University had students or faculty participating.

Over forty students and faculty participated in the course. In addition to the students from our own university, students and faculty at eight other universities participated in the course remotely via the video network. Figure 1, which is a map of the University of North Carolina system, includes the locations of most of the universities involved. The enthusiastic response to our course clearly indicates the need and interest for such a course.

It is our premise that students best learn with hands-on experiences that include working software that they use and modify. In fact, because of the importance of the programming assignments in such a course we structured the course around the assignments. The course involves a series of six assignments including detailed assignment handouts and extensive lecture slides. This paper describes these assignments and thus also explains the course organization.

This paper is organized with the next section addressing some preliminary decisions we had to make about course coverage and expectations of student backgrounds. The third section has a subsection for each of the six assignments. In the fourth section we discuss our experiences. In the last section we review related work and conclude.

2. PRELIMINARY DECISIONS

Deciding on what *not* to have as the first assignment was an early key decision. What we chose not to cover in the lectures or the assignments are the details of installing the grid software. Installation is complex and understanding the details would not be appropriate for an undergraduate course, especially at the start of such a course. Nevertheless, we did develop a detailed installation guide handout that we posted on the website for students to read if they wish.

The posted installation guide served another important purpose which is to ensure consistency across the installations at all the sites. Some of the remote sites had just a few students so we gave those students accounts on our local machines that they could logon to remotely. However, two of the remote sites (NC State University and UNC Wilmington) had about thirteen students each in the course. At those remote sites, the faculty or system administrator performed their own installations or had preexisting installations. Our installation guide provided a point of reference to help ensure consistent installation procedures and resulting grid environments across the sites.

Another issue concerning the installation guide involved what to describe installing. The Globus Toolkit of the Globus Alliance [7] has become the most widely-used basic, open source grid software. However, an installation guide for only the current version of the Globus Toolkit is insufficient. A working grid requires many other pieces of software. Fortunately, the National Science Foundation Middleware Initiative (NMI) [8] has developed a distribution containing many open source grid software contributions. We developed our installation guide based on downloading the NMI distribution and installing selected key parts of it.

Though there is some proprietary grid software, the grid environment is one in which open source software based on open standards is the predominant case. Furthermore, much of the software includes several other features: it is written in the Java programming language, Linux is the operating system, and standard open source software from other application areas (especially software developed as part of the Apache web server project) is used. There are certainly significant efforts with respect to other languages and operating systems, but a representative introduction to current grid software should involve Linux and Java. Consequently we decided that because the course audience consists of upper-level computer science majors we did not think it unreasonable to expect the students to have had some experience with both of these.

Though grid environments are often used to access high-performance computing resources and large capacity storage resources, the hardware requirements needed to install an open source grid environment are not exceptional. At our university we used four 1.3 GHz Pentium IV machines each with 512 megabytes of main memory and a 30 gigabyte disk.

Another key decision we had to make was how the students would use the software. We would characterize a grid as having at least the following key parts:

1. a graphical portal that requires authentication and displays available resources
2. a graphical means to create a workflow (in other words, a) to create job instances out of selected resources, b) to place those job instances at specified sites, and c) to identify execution ordering dependencies between the job instances)
3. the actual procedure of submission and distribution of the jobs that form the workflow
4. monitoring and visualization of the progression of the execution of the workflow and visualization of the results upon completion of the workflow

Because computer science majors make up the audience for this course, we decided to focus on the implementation issues rather than on the user's perspective. Consequently, we chose to use a bottom-up approach. The assignments and lectures start with the lowest level concepts (i.e. web services) and build up to higher levels of software. Thus, the way to use the software is command-level for most of the course. Only after we reach the top level do the students see and start using graphical interfaces for accessing the grid.

3. THE ASSIGNMENTS

Each of the assignments includes a detailed handout available on the course website. Here we merely identify the key points. The lectures for the course cover the material needed for each assignment. The lectures also include introductory material motivating the use of grids and providing a historical context. The last several lectures were by guest speakers on specialized topics.

3.1 Assignment One: Web Services

The standards which grid software implement have been changing rapidly. However, those standards and software appear to be converging to a stable state. A key feature of that state is that the grid is implemented by grid services and that grid services are an extension of web services. Web services, in turn, are based on a set of well-defined standards [12]. Thus, we decided that the series of assignments should start with an assignment where the students learn how to implement and execute a web service.

A web service is often implemented using a Java servlet which in turn requires the presence of a servlet container. Our assignment one uses the widely used Apache Tomcat servlet container. One instance of Tomcat is running on the machine before the student starts the assignment.

All of the assignments have two halves. In the first half the student is guided through the steps of running a version of the software being demonstrated. In the second half the student modifies or extends the provided client and server software and then repeats the steps. Assignment one involves creating all the needed server source files, creating the server executable files, creating the client source file, and compiling and running the client.

The server source files are partially provided. The example service is a MyMath class that includes a method returning the square of the number which is passed to it as an argument. A source file with only that class is provided. The student then uses the Apache Axis software to generate all the other needed server source files. These source files include the WSDL (Web Services Description Language) describing the service, Java interfaces used by the client, a Java stub class to be called by the client, and a Java class to use in locating the service.

The student then compiles the service source files using the Java compiler. We provide and explain the source file for an example client that accesses the server. The student then compiles the client source file and runs the client. The output can be examined to see that the server worked.

The student then extends the MyMath class by adding another method that checks whether the number passed as an argument is even or odd. This assignment and its handout were motivated by one developed by Amy Apon [1].

3.2 Assignment Two: Grid Services

Assignment one does not require the use of any grid software since it involves only a web service. Assignment two adds the complexity of requiring some grid software. However, it only uses the most basic layer of grid software which is implemented by the Globus Toolkit (we are using version 3.2 which is the most recent version).

The assignment two grid service is another mathematics-related class that has methods for changing the value of a state variable by addition or subtraction as well as a method for returning the current value of the state variable. In contrast to the web service, this grid service maintains some state between method calls.

Being a grid service makes the assignment more complex in several ways. One is that Tomcat is not used directly as the servlet container. Instead the servlet container is started through Globus by the student as part of the assignment and terminated by the student as part of the assignment. Since multiple students are using the same machine there will be multiple containers running and the students must ensure that each container is listening on a different TCP port. The students need to learn how to determine which TCP ports are free.

A second way that a grid service is more complex is that deployment of the service is more explicit than it was in assignment one. As in assignment one, the base server source file is provided. However, the WSDL file (actually Grid WSDL) is also provided and described instead of merely generated and used. The student then builds all the other required source files, compiles them, and deploys the resulting service using the Apache Ant build system. Ant is similar to the UNIX make program but based on XML.

At this point the student turns his or her focus to the client software. The source code for an example client is provided and described. The student compiles this code to create the client executable, starts the container on a free port and runs the client. The student then learns to end the session including terminating the container. As in assignment one, the student then extends the server and redoes all the steps. This assignment is a modification of the one in Bojo Sotomayor's tutorial [5] though our handout is significantly more detailed.

3.3 Assignment Three: Job Submission

At this point the student has the basic understanding of how to define, create, deploy, and use a generic grid service. In assignment three we focus on how to use a particular, very important, grid service: the grid service that is used to submit a job. That grid service is called the Globus Resource Allocation Manager (GRAM).

The student first is introduced to the Resource Specification Language-2 (RSL-2). RSL-2 is the language that GRAM uses starting with version 3.2 of the Globus Toolkit. It is an XML schema (RSL-1 was not) with some standard attributes such as the file path to the executable for the job that is being submitted and the file paths for standard input, standard output, and standard error. The RSL-2 file used to submit the executable file /bin/echo is examined.

At this point the student is ready to start the sequence for submitting a job to the grid. As discussed for assignment two, the student must first identify a free port and then start a container on one of those free ports. It would be convenient for the next step to be for the student to submit the job. However, an intermediate step is needed that involves security.

An important part of Globus is its implementation of the Grid Security Infrastructure (GSI). In a grid when a user requests a service that service is often not on the same machine as the user. In fact the service is often on a machine at a different site and that other site may belong to a different organization. Thus, mutual authentication of both the user and the service is essential. Mutual authentication is done by GSI via certificates and the Secure Socket Layer (SSL) protocol. In particular, each user and service in the grid is identified via a certificate in the X.509 certificate format. Each certificate has been signed by a Certificate Authority (CA) through the use of digital signatures and public key cryptography. The mutual authentication is initiated by the user entering a passphrase.

The presence of these certificates and the mutual authentication process have so far not been visible to the user. However, at this point in the use of GRAM, the presence of GSI in the background becomes apparent. In particular, GSI includes a delegation capability which is an extension to the SSL protocol. The service requested by the user often needs to, in turn, make a request of other services in order to complete the user's request. Each of these secondary requests requires mutual authentication and thus an entering of a passphrase by the user. Delegation avoids the need to reenter the passphrase through the creation of a proxy certificate.

Thus, before the student can perform the actual job submission, a proxy certificate must be created to be used by GSI when the job submission takes place. The proxy certificate creation is done by the student executing the grid-proxy-init command. After that command the student can submit the job. GRAM is invoked using the managed-job-globusrun command. That command takes as its argument the RSL-2 file that identifies the job to be run and other information (such as the file to be used for standard output).

The job runs and the student can see that it ran by looking at the output of /bin/echo in the file designated for standard output. The student then repeats this sequence but with a new executable job that she has written and compiled The new job is a Java program that implements the functionality of /bin/echo. In addition to creating the new executable job the student must edit the RSL-2 file appropriately.

Just before assigning assignment three we decided to add another major step. In the additional step the student combines assignment two and assignment three. First, the student writes a grid service called Item, which provides inventory information about the items being sold in a store. Second, the student deploys an instance of the Item grid service. Next the student writes a client called Shopper that accesses that instance of the Item grid service. All of these steps are like parts of assignment two, but are non-trivial since many grid-related files have to be changed as well as the writing of the source code for the grid service and the client.

The next phase of this step involves GRAM job submission and the previous steps of assignment three. First, the student creates a RSL-2 file to submit via GRAM one instance of the Shopper client. That client, once submitted, interacts with the deployed Item grid service as before. Then, the student modifies the RSL-2 file to submit via GRAM four instances of the Shopper client. All four instances of the client then concurrently access the one instance of the deployed Item grid service.

3.4 Assignment Four: Job Distribution

The Globus Toolkit only provides part of the software needed for a grid environment. Assignment four is the first assignment that goes beyond the Globus Toolkit. In particular, GRAM provides only basic job submission. More sophisticated job submission and distribution is done by software such as Condor-G (Grid-enabled Condor) [7]. Fortunately, the National Science Foundation Middleware Initiative (NMI) grid distribution includes many programs including Condor-G (Grid-enabled Condor).

In assignment four the student uses Condor-G to submit a job. The student first learns the format of the submit description text file that Condor-G requires. An example of such a file is provided. The student then creates a proxy certificate, learns how to check the status of the condor pool of machines, and uses the submit description text file and the condor-submit command to submit the job. The student then monitors and manages the progress of the job and the status of the condor pool and verifies the output of the job when the job completes. As in assignment three the student then repeats this sequence but with a job that he or she writes and compiles.

3.5 Assignment Five: Parallel Programming

The previous assignments illustrate how grids can be used to solve important classes of problems. These classes include cases where jobs are single executions but they are embedded within a larger problem such as the parameter sweep of a design space or a step within a workflow. However, grids can also be used for parallel programming. Message Passing Interface (MPI) is widely used to support parallel programming [3]. A version of MPI, called MPICH-G2, works with Globus. Assignment five involves the students writing some parallel programs using MPICH-G2 and running them on our grid.

3.6 Assignment Six: A Workflow Editor

By the end of assignment five the student has an understanding of how the internals of a significant part of a grid environment work. It is now time for the student to turn to the view seen by a user. As mentioned earlier, the user is authenticated via a grid portal and then uses the portal to specify which resources to execute within a workflow. The workflow is then submitted and the user can monitor the progression of the workflow and use visualization to examine the results when the workflow completes.

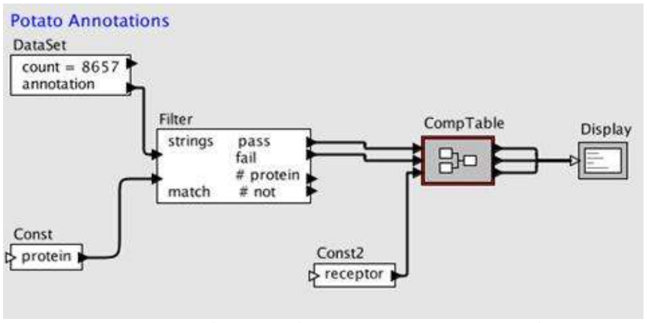

Figure 2: Creation of a workflow [9] using the workflow editor developed at the Department of Computer Science of UNC-Wilmington.

We show the user viewpoint to the student by examining the workflow editor developed at the Department of Computer Science of UNC-Wilmington [9]. An example use of this workflow editor is shown in Figure 2. This example illustrates the flexibility of the UNC-W software. The module selected by the user and placed in the workflow need not be a regular job. Instead it can be the data set resulting from a database query (as in the top left rectangle of the figure), a particular constant value (as in the bottom left rectangle of the figure), or useful general modules such as a filter (left middle rectangle) or a statistical analyzer (right middle rectangle). Assignment six involves the students using this workflow editor in an example grid problem.

4. EXPERIENCES

The course that is structured around the above assignments is now finished except for the final exam. Before the semester started we had carefully tested the first four assignments. We tested them ourselves on our own machines. We also conducted a trial run of using them in a workshop attended by UNC computer science faculty. We were fairly confident at that point that they would work as expected. The one surprise from the pre-course workshop was how memory intensive the servlet container is. If more than a few containers are started on a single machine with 512 megabytes of main memory, thrashing occurs.

During the course those four assignments worked as we expected. Our one mistake was the last minute addition of the last step to assignment three (the store grid service and the shopper client). Completing this step is quite complex and turned out to be beyond the skills of most of the students. We extended the deadline for assignment three in an attempt to give the students sufficient time to complete it. As a result we did not have enough time for all of the remaining assignments. Our solution was to post assignment five (MPI programming) but not to require the students to complete it.

Our experience with assignment three highlights an issue that we debated about the nature of the assignment handouts. If we want the assignment handout to be as useful as possible as a resource for the students then it should contain a great deal of detail. On the other hand, too much detail can make the assignment too straightforward and not require sufficient independent thinking on the part of the student.

The grid software was installed at two of the remote sites as well as at our site. Even though the three sites had different hardware configurations, students were able to complete the assignments at all three sites. We found that supporting the student use of the software was very labor-intensive. We had two students at our university dedicated to providing support for software issues in the course. Their time was completely occupied by answering questions and resolving problems. Consequently, we did not have time to accomplish one of our goals in the course. That goal was to connect the grid software at the three sites into a single operational grid, instead of three independent grids.

5. RELATED WORK AND CONCLUSIONS

The study of grid computing education is in its early stages. A significant step was the set of papers presented at the *International Workshop on Grid Education* at the *4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid04)* in April of 2004.

The paper [4] by Bini Ramamurthy on the GridForce project is especially noteworthy. The GridForce project involves developing a two course sequence on grid computing at the senior undergraduate/first year graduate student level. The laboratories in the first course involve web services and grid services. However, their approach to grid services is different than ours and aimed at more advanced students. The first grid service laboratory does not use Globus (or Apache Axis); instead it uses a “minimal grid framework”. The students build the framework themselves. The second grid service laboratory requires the students to install Globus themselves. The GridForce approach is a useful complement to our approach. It would be attractive to students who complete our course and want to investigate grid computing further.

Bojo Sotomayor has written *The Globus Tookit 3 Programmer's Tutorial* [5]. An early section of the tutorial shows how to write “your first grid service in five easy steps.” Our assignment two is an adaptation of this section to our environment with our handout expanding on what we feel are the key points.

The work [1] by Amy Apon at the University of Arkansas is also related. Like us, she recommends starting with a web services assignment and then a grid services assignment. Our web services assignment was motivated by hers. Her grid service assignment, like ours, is a modification of the Bojo Sotomayor example.

In conclusion, we believe that the course we have developed is a significant contribution to the study of grid computing education. The sequence of assignments builds upon earlier work. The overall sequence of assignments is new. Some assignments are completely new. Other assignments are based on earlier work but for all the assignments we have developed unusually detailed handouts to help explain the concepts and to ensure that the student will avoid common pitfalls. The lecture slides are completely new. Materials appropriate for the undergraduate computer science major are significantly different from those appropriate for graduate students or professionals. We believe (and are supported by our experiences) that we have found the right level of assignments and materials for the undergraduate audience.

All of the course materials including lecture slides, assignment slides, the grid software installation handout, and the assignment handouts are available on the web at http://cs.wcu.edu/~abw/CS492F04/index.html.

6. ACKNOWLEDGEMENTS

Our thanks to the National Science Foundation (DUE 0410667) and the Office of the President of the University of North Carolina System for their financial support of this project. Our thanks also to Mark Baker and the Distributed Systems Group at the University of Portsmouth in England for the time Mark Holliday spent there while working on this paper.

7. REFERENCES

[1] Apon, A., Mache, J., Yara, Y., and Landrus, L., *Classroom Exercises for Grid Services*, Proc. of the Linux Cluster Institute Int. Conf. on High Performance Computing, Austin, TX, USA, May 2004.

[2] Foster, I. and Kesselman, C. *The Grid 2: Blueprint for a New Computing Infrastructure, Second Edition*, Morgan Kaufmann, 2004.

[3] Gropp, W., Lusk, E., and Skjellum A., *Using MPI Portable Parallel Programming with the Message-Passing Interface, Second Edition*, MIT Press, 1999.

[4] Ramamurthy, B., *GridForce: A Comprehensive Model for Improving the Technical Preparedness of our Workforce for the Grid*, Int. Workshop on Grid Education (CCGrid04), April 2004, Chicago, IL. USA.

[5] Sotomayor, B., “The Globus Toolkit 3 Programmer's Tutorial,” http://www.casa-sotomayor.net/gt3-tutorial/multiplehtml/index.html.

[6] The Condor Project Homepage, http://www.cs.wisc.edu/condor/.

[7] The Globus Alliance: The Globus Toolkit, http://www-unix.globus.org/toolkit/.

[8] The National Science Foundation Middleware Initiative, http://www.nsf-middleware.org/.

[9] Wilkinson, B., et. al. (UNC-Wilmington, lead), *Fostering Undergraduate Research Partnerships through a Graphical User Environment for the North Carolina Computing Grid*, University of North Carolina Office of the President, 2004-2006.

[10] Wilkinson, B., Holliday, M., et. al. (Appalachian State Univ. lead), *A Consortium to Promote Computational Science and High Performance Computing*, University of North Carolina Office of the President, 2004-2006.

[11] Wilkinson, B., Holliday, M., and Luginbuhl, D., *Introducing Grid Computing into the Undergraduate Curriculum*, National Science Foundation, DUE 0410667, 2004-2006.

[12] World Wide Web (W3C) Consortium, Web Services Activity, http://www.w3.org/2002/ws/.