

Analyzing and Visualizing a Large Data Set Using Grid

Bina Ramamurthy

CSE4/587 Information Structures

Due Date: 4/18/2004 by mid-night.

Purpose:

1. Design and develop a solution to analyze a large set of real data from a pharmaceutical experiment.
2. Understand the components and operation of a condor-based (High Throughput Computing) grid (CSECCR) built using recyclable Sparc 4 machines.
3. Design and implement a Java application and submit script to execute the solution developed in step 1.
4. Learn to use database and graphing tools with grid-based jobs.
5. Understand the process of utilizing CPU cycles offered by CSECCR grid.

Preparation:

1. Get a clear understanding of condor-based CSECCR grid you will be using for this project. (See notes given below).
2. Understand the technology underlying Condor: its architecture and application models. See <http://www.cs.wisc.edu/condor/>
3. Understand the role of submit scripts.
4. Learn to use various tools such as GnuPlot for drawing graphs of various relationships among the data.
5. Make sure you have an account on johnlee.ccr.buffalo.edu by logging into it using secure shell from any of the cse machines. Your username name is same as UBIT username and the password is your person number.

Technology details:

You will work with CSECCR grid shown in Figure 1. The grid is primarily composed of Sun Sparc4s which form the compute nodes. The 40 compute nodes form an internal private Class C network with a grid front end. All nodes run Solaris 8, and middleware is configured for a 'shared file system' oriented job execution. There are a total of around 40 CPU's and the total memory is around 2.5 GB. The front end has another external interface, through which jobs are submitted.

The grid also has various middleware solutions installed in it for educational research. The middleware is primarily composed of NMI (NSF Middleware Initiative) components. Middleware components installed range from Globus, Condor-G and PBS (Portable Batch System) and NWS (Network Weather Service). PBS is for job management and scheduling, Globus and Condor for resource management, NWS and Ganglia for distributed resource monitoring. You shall be using Condor as your primary grid middleware. This means that both the *job management* part and the *resource management* part of the grid is taken care by Condor daemons. Condor has various commands which let's you submit

jobs, monitor and manipulate the job queue, assign job preferences etc as described later in the Getting Started with CSECCR Grid.

Assignment (What to do?):

You are given a data from a gene expression experiment. The gene expressed by their DNA after treatment with a certain drug for a set of patients is recorded and provided in the database. This data is available at /projects/bina/PD_pt1_14.xls. This will be ported to a relational database connected to the CSECCR grid. (You don't have to do this. Vijay will do this.) The data contains 14 patient's information, different types of genes, their expression levels at various times (1 hours, 3hours, 1 day, 1 week etc.) after treatment with some experimental drug. This provides a "time series". You analysis and graphing can be (i) as simple as line drawings of gene expression over time, (ii) average of gene expression for a specific time over patients (iii) repeated measures ANOVA, (iv) application sophisticated algorithm such as that of Markovitz to choose best responding gene etc. It is our goal to provide all possible analysis. It is up to the expert to interpret the analysis. For this purpose you may provide a simple portal for visualizing the results of the analysis.

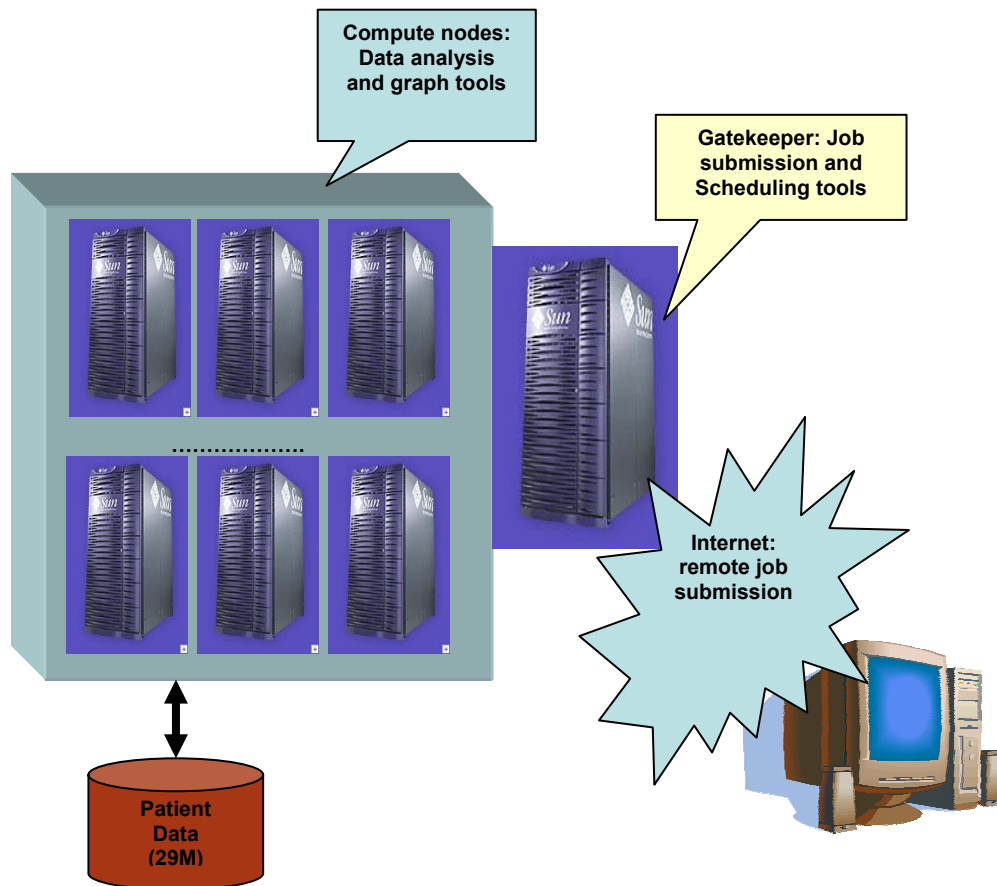


Figure 1: CSECCR Condor-based grid

Use Scenario: A scientist who has gene expression data or similar data in that domain will populate the database with the data. He/she will then choose a set of analysis to be carried

out on the data. This could be selected from a menu of available analysis tool. Once the scientist is satisfied with the selection he/she submit the job to the grid for processing. The grid software will orchestrate the analysis of the data but organizing needed resources and tools. On completion the scientist will be notified and he/she may review the results before initiating further action/analysis/repeat experimentation. A portal that facilitates all the operations/features discussed above will be an ideal solution. The requirement for the project within the scope of this course is to submit a Java or C-based program to analyze the data and to obtain outputs in the form of graphs (visuals).

Your assignment consists of three parts: (i) design and implement a Java/C/C++ program that reads in the data from the data source connected to the grid, (ii) interface Java/gnuplot API and and (iii) prepare graphs for visualization. It is enough if you generate a few representative set of graphs as a proof of concept. If you prepare the same graph for every single data set you may run of limited disk space that may be allocated to you. You will work in groups of not more than two people.

Analysis and Design:

Data Analysis: Study the excel file that contains the data. Some of the columns are easy to understand and many of them of domain-specific information which is quite common with such data collections. One of the sub tables you will be working with is the gene-expression over a period of time. Locate this on the excel sheet and study it. In the second worksheet of the excel workbook the averages for the same data are given. You will work with these sets of data for specific genes of your choice. You can draw line graphs, average line graphs, and other sophisticated graphs of your choice.

Grid side: You should design your application in Java/C/C++ and use appropriate external and built-in API for generating graphs. Alternatively you may generate graph data files and visualize these explicitly using appropriate tools such as gnuplot.

Client side: Ideally we would like to see a portal implemented to carry out the analysis and visualization. Due to lack of time this aspect is left as an optional exercise.

Implementation steps and details:

- 1. Working with Condor-based grid:** You will be given accounts on the CSECCR grid described above. You will login and verify that you have an account. If not send mail to bina@cse.buffalo.edu Log into the account and prepare a simple Java program or C program and prepare job submission script for as described in the “Getting Started” section above. Submit the job and monitor its progress using command line operations or the web-based monitoring tool Ganglia (<http://johnlee.ccr.buffalo.edu/ganglia>)
- 2. Working with the data source:** We will convert the data available in the excel file into a relational database and make it accessible through jdbc/odbc interface. However

this may work for Java programs. For C++/C programs you may use a simple subset of the data in a file in your local disk space.

3. **Working with graphing tool:** Your java program will read the data from the data source, process it and generate data for graphing. You may use Java 2D API or free ware gnuplot API for Java at <http://www.is.informatik.uni-duisburg.de/projects/java-unidu/api/de/unidu/is/gnuplot/package-summary.html>
4. What are we interested in? We are interested in monitoring and benchmarking the power of the condor-grid we have assembled. We expect that you will computationally intensive analysis of the data. An example of is repeated measure ANOVA, the details of which can be found <http://www.utexas.edu/cc/docs/stat40.html>
5. What will you learn? Working with grid-based job preparation, submission, monitoring, and managing data for data and computationally intensive problems.
6. For Java-based job, you will need the class files (ex: helloWorld.java), shell script with java command (hello.sh), and a submit script (hello.submit) specifying resource requirements. You will `condor_submit hello.submit` to schedule the job on the grid. See “Getting Started with CSECCR Grid”, document for details on how to prepare and submit a job.

Submission Details:

Create a `project2` directory in your project space. Copy your source code and outputs from CSRCCR grid into this directory. Add an experience report that details how you accomplished tasks outlined in this project. Call it `ExpReport2`. This report should also outline how we can use your program to obtain your outputs.

```
zip -r project2.zip project2
```

Use the electronic submission program that corresponds to your class (cse4/587). For instance students in cse587 will submit by typing `submit_cse587 proejct2.zip` at the Unix prompt.

Getting Started with CSECCR grid
Prepared by Karthikram Ramamani
kv8@cse.buffalo.edu

A typical job for the CSECCR grid will be written a high level language, say, Java or C++, compiled and preprocessed, if necessary to prepare the executable. A submit script is prepared that declares all the requirements for scheduling and execution of the job. Then the job is submitted using Condor submit command. Condor provides commands for monitoring and controlling the executing job.

Preparing the executables:

For java program, compile the programs using javac. No special preprocessing is needed. Prepare a submit script and shell script using the sample scripts provided in your home directory.

For C/C++ programs, compile using

```
condor_compile gcc sourceName -lm -o executableName
```

Job Submissions via Condor:

The condor_submit command is used to submit jobs to the Condor scheduler. The argument to the command is a submit script file which specifies the job preferences. Matchmaking in Condor is done on the basis of this submit file or *ClassAds*. You will find example submit files in your home directory. Modify it as per your needs. Certain attributes are mandatory for proper job submissions, so go through the Condor manuals before you make any considerable changes to the submit files. The load on the cluster at any given time can be monitored using Ganglia Distributed Monitoring, by connecting to the apache server at <http://johnlee.ccr.buffalo.edu/ganglia>

Any error conditions that arise while executing your jobs are logged in the log file you specify. If your jobs are go into idle state for long periods, check your submit files for incompatibilities. 'condor_q -analyze' command gives the analysis of the submitted job.

Condor Quick Reference:

condor_compile

The condor_compile command assists in linking jobs with the Condor libraries so that Condor features like migrating and check-pointing are made use of. For your Java project you need not make use of condor_compile.

condor_q

The condor_q command displays the Condor job queue at the instant. The job id, running time, job status etc are displayed. The command has various options which make the output more descriptive. (Go through Condor manual for preferences)

condor_rm

This command along with the job id, removes the specified job out of the queue. Use this command to delete your jobs from the queue.

condor_status

This command displays the hosts running Condor, their status (Claimed/Unclaimed), host info. Command has options for displaying host specific information like *machineAds*, architecture etc.

condor_submit

This command is used to submit jobs to Condor. Takes as argument a submit file, which specifies the ClassAd (User Job Preferences).

condor_history

Lists the history of jobs submitted to Condor along with their exit status.

condor_hold & condor_release

These commands hold and release the job specified respectively.

Sample Condor Submission.

Any condor submission will require a submit script which specifies your job classAd (job preferences). Samples submit script is given below. Note that “kv8” has to be replaced by your username.

```
*Executable = /home/kv8/submit.sh
Output =     /home/kv8/submit.out
Log =       /home/kv8/submit.log
Error=      /home/kv8/submit.error
*Universe = java
*Requirements= Arch == "SUN4x"
*Rank      = Memory >= 30
*Arguments= helloworld
*Queue
```

*indicates the attribute is mandatory

Submit script above says the execution universe is a java universe, class name is helloworld, specifies the executable to move, the job requirements and the rank. You also request the job to be queued for submission. A sample *submit.sh* for above job would be:

```
#!/bin/sh
java helloworld
```

Note that all references to files are absolute paths. Please avoid using relative path names in your submit scripts. Go through the Condor manual at <http://cs.wisc.edu/condor> for detailed information on Condor.

CSECCR Grid Etiquettes:

The server you log into is johnlee.ccr.buffalo.edu. Your username will be your UB IT NAME and your initial password will be your UB person number. Make sure you change the password immediately after you log in.

Considering the infrastructure of the Grid, during times of heavy load, the grid is bound to be slow. Make sure you don't submit too many jobs at a time and flood the job queue. When your job finishes running, you will be notified about the exit status of the jobs via email. You can monitor the status of your jobs in the queue using Condor commands. Feel free to remove your jobs from the queue if you don't need them. We would certainly appreciate saving computational cycles.

PLEASE wait for your jobs submissions to finish, before you fire up other jobs. This would ensure smooth network traffic and optimal performance for all users. During times of high network traffic, redundant jobs from a user shall be paused or removed if necessary.

Use JohnLee strictly for job submissions only. Do not log into other servers from JohnLee or initiate Netscape connections. **Your disk quota is a hard limit of 25 MB only. Avoid using relative path names in any job submission scripts you write. Please specify files using absolute path names, wherever you use them.**