A COMPARIASION OF JOB DURATION UTILIZING HIGH PERFORMANCE COMPUTING ON A DISTRIBUTED GRID

Michael Austin, JerNettie Burney, & Robyn Evans
Mentor: Je'aime Powell Principal Investigator: Dr. Linda B. Hayden
1704 Weeksville Road, Box 672
Elizabeth City, North Carolina 27909

Abstract--*The Polar Grid team was tasked with testing the central manager system on Elizabeth City State University to ensure that it was prepared for grid computing. This was achieved by installing the Condor 7.4.0 client on iMac workstations computers located in Dixon hall, Lane hall, and E.V Wilkins on the campus of Elizabeth City State University. Condor allowed jobs to be submitted to the central manager and distributed to one or more nodes. The job that the team submitted to Condor was the compiled Sieve of Eratosthenes in C++ code. This code generated prime numbers from 0 to 500,000, and was essential in testing the job submission process. The compiled code that was used in the script file was submitted to the central manager through Condor. These jobs were then distributed to available nodes for processing.*

After each successful job submission, log files were created to record statistical data. The data was of the elapsed time it took to process its entire submission along with the time it took to process each individual job. The data from these tables were imported in to Mini Tab, which was a statistical analysis software package. An analysis of variance (ANOVA) was then performed to determine if the elapsed times of the submissions varied within a 5 percent level of significance. From ANOVA, statistical evidence proved that by increasing the number of nodes, the elapsed time would decrease; therefore showing a performance increase..

Keywords – ANOVA, cluster, Condor, grid, Kitoto, node, script, VikeGrid

I. NATURE AND BACKGROUND OF THE STUDY

A. Introduction

B. Statement and Background of the Problem

The Polar Grid team's ultimate goal was to find a Titanic prime. A Titanic prime is a prime number that has at least a 1,000 digits. This prime number takes a massive amount of computing power in order to find it in a short amount of time.

The first step toward achieving this goal was to find a prime number generator. The prime number generator that was chosen to use was derivative of the Sieve of Eratosthenes. This prime number generator is capable of producing numbers in between 0 to 500,000.

The code generated was used to test the grid. In the use of this prime number generator we tested how long it took to complete fifty jobs with various numbers of workers.

C. Hypothesis

The team hypothesized that the elapsed time for a single node to complete a submitted job would take much longer than multiple nodes collaborating to complete the same job. This reason being that if the nodes worked together collaboratively instead of individually, the amount of time it will take for a job to finish would be much quicker than if a single node was working by itself.

D. Definitions of terms

- **Cluster –** the network of small computers that is under the control of a larger computer
- **Grid** – a network of computers that work together, but are topographically dispersed
- **Script** – defines an executable file
- **Condor** – a batch scheduler used to communicate with VikeGrid

- **Node** – a core
- **VikeGrid** – a grid located on the campus of Elizabeth City State University that uses bothe desktop and cluster resources
- **Kitoto** – name of the central managr that manages VikeGrid
- **ANOVA –** acronym meaning the analysis of variance; a program in Minitab used as a statistical method for making simultaneous comparisons between two or more means and attempts to prove a significant relationship between variables

II. Review of Literature

A. Prior Research

Different studies on Titanic prime numbers have been conducted over the years by different mathematicians and scientist to find an equation that could generate thousands of primes in a small amount of time. With the aide of computer scientist, algorithms using various programming language codes were created to generate these numerous amounts of primes using computing power. These different algorithms, using codes to generate primes at a quickly are called "sieves." These sieves can be used in numerous programming language such as Java, C++, C#, and Fortran to name a few. There are different sieves that can be utilized, eachof which has a different performance, a different method for how to compute the primes, along with its limitations of how many numbers it can compute. One example is the Sieve of Eratosthenes, named after a Greek mathematician who created an algorithm for calculating primes. His algorithim is based on. This was one of the earliest sieves created, and can generate primes from zero to 500,000 within in a few minutes. The next sieve was that of named Atkin that could generate more primes in a less amount of time. For the team's research the main focus was on using the Sieve of Eratosthenes. The objective was not to find the fastest generator, but any generator that could be used to test out the cluster system Kitoto.

This research was an extension of the 2008 – 2009 academic Polar Grid Team's research, "Implementation of a Static Cluster". The previous Polar Grid Team conducted their research on how to permanently install a Condor-based test cluster. The team's main focus was on network topography, naming schemes, user management and compatibility concerns. The target machines included the SunFire V480 management server and several SunBlade 150 workstations, which utilized Solaris 10 as their primary operating system and the Condor High Throughput Computing software was utilized as a job scheduler (Brown, Jr., Jefferson, Jr., & Vick, 2009).

This year's team expanded that research in order to install Condor onto the Macintosh and Windows environments. This year's team was able to install condor easily with less configuration issues so that more time could be devoted to the main objective (Brown, Jr., Jefferson, Jr., & Vick, 2009).

B. Related Literature

The team did research as to what a cluster system function is and the history from which it comes. The first resource dealt with research about cluster computing (Baker, Buyya and Hawick). The importance of this book was to understand where a cluster computer comes from and the different works that happen behind the scenes to make the computer do its different tasks. In order to understand how it works, one must first understand what it is and the different operations it can achieve.

The next resource told of the different uses that the marketplace has for high performance computing (HPC) (Strohmaier, Dongarra and Meuer). This resource was needed for a broader interpretation for why does the science community have a need for the knowledge of high performance computing. It was very important that the team understood that HPC goes further than what was researched in the duration of the project. The reason being for this is in the future, teams can try out a different aspect of HPC to excel their knowledge in all the different things HPC can accomplish.

Finally, the last resource informed the team of the different technologies that are at work within the cluster computer (Petrini, Feng and Hoisie). Though it may not have seemed that important, these processes have a very huge effect when something goes wrong with a cluster computer. If a team didn't know the different processes that are used to carry out tasks given to the computer, that team cannot reevaluate the steps that were taken to see if there was an error carried out when the computer was commanded to do a specific thing or even if the task was given to the right process. Therefore, not only must the team know of the architecture of cluster computers, but they must understand the micro architecture to gain a higher perception of how to use a cluster computer.

III. Methodology

A. Definition of the population

There were two types of data collected for this research project: elapsed time it took to process the entire submission and the time it took to process each individual job. Overall there were a total of eight submission files.

The team was able to access these datasets by submitting the Sieve of Eratosthenes job—used to test the titanic prime generator found earlier—to Condor and uploading the log files for this job on to a Condor Log Analyzer. The analyzer allowed the team to upload the log files that were generated by Condor, and get back both graphics and an explanation of what happened over the course of the submission. From there, the log times were uploaded into a Microsoft Office Excel 2007 worksheet, and the means for the eight submission files were calculated. This information was then uploaded into MiniTab—a statistics package developed in 1972 at Pennsylvania State University by researchers Barbara F. Ryan, Thomas A. Ryan, Jr., and Brian L. Joiner (What is Minitab, and where can I access it?, 2005 - 2010). The analysis of variance, also known as ANOVA—a feature in Minitab used to tests whether the treatment means differ from each other—was used to see if the elapsed time for a single node to complete a submitted job took longer than multiple nodes collaborating to complete the same job (ID 121: What are the differences between Analysis of Mean?, 2010).

B. *Procedure*

Throughout the course of this study, there were many procedures utilized to ensure the accuracy of results.

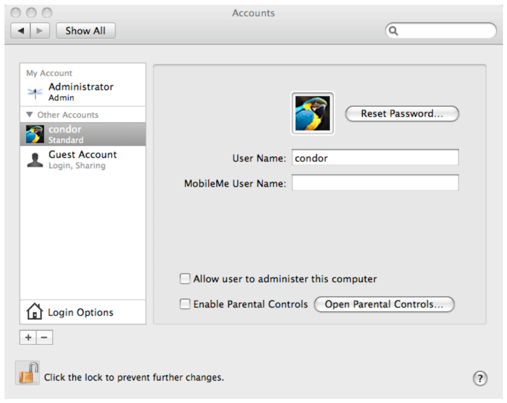
The group started off by downloading Condor 7.4.0 for the MAC OSX computers located in the Center of Excellence in Remote Sensing Education and Research (CERSER) lab in Lane Hall. The 7.4.0 version of Condor was chosen for two reasons: it was the current stable version of Condor and the team wanted to keep the grid homogeneous.

The next step was to create a Condor account for each computer. To do this, the group logged in as administrators and went to "System Preferences" and clicked on "Accounts". From here, the account was set up, by authenticating and clicking on the "+" icon. The appropriate information—such as the name, short name, and the password—was entered and the account was created. See figure 1.

In addition to creating the Condor account, the group decided to changed the energy settings while in the System Preferences pane. The energy saver settings thus became, "NEVER put the computer to sleep" when it is inactive and "put the display to sleep when the computer is inactive" after an hour. These setting changes were made in order to keep the computers active during the submission and test run for the jobs.

Next, Condor 7.4.0 was installed. To do this, the "Terminal" window in on the Mac side of the computer was utilized. Terminal is referred to as the "gateway command line" in Mac OS X and is a utility program that "provides an interface with a shell, or command interpreter" that is used to "enter data, send commands, and receive output from the mainframe computers" (McElhearn, 2005).

The "su" command, was then used to switch to the administrator account, and then the "bash" command was used to call the bash shell. The group then changed the permission for the condor directory by typing "chmod a+rx ~condor." From here, the directory was changed and a new directory, named condor, was created. To make sure that there was nothing in the directory, the "ls" command was used. The Condor 7.4.0 file was then copied from the "Downloads" folder and pasted into the "/condor/" directory. The working directory was then printed and all the items inside were printed. The file was then extracted by using "tar –xzf condor-7.4.0-MacOSX0.4-x86-dynamic-unstripped.tar.gx ."



User Name: condor
MobileMe User Name:

Fig. 1. Creation of the Condor Account

Condor was set up, by changing to the "/condor/condor-7.4.0" directory and installing the file by using "./condor_configure --install – install-dir=/condor/condor-7.4.0 – local-dir=/condor/codor-7.4.0/var --owner=condor." Symbolic links were then created. To do this, "ln –s /condor/condor-7.4.0/etc/condor_config ~condor/" was used.

To grant access to the condor commands, the path was changed to "$PATH:/condor/condor-7.4.0/bin/:/condor/" and then the files were edited using the "nano" command.

The "nano" command is used as an alternative for the vi editor. The nano editor is a easier way to edit fils in terminal, offering a screen-based interface that can be navigated using the arrow keys on a keyboard (Peterson, Part 1: Getting Started, 2009).

In the "condor_config" file, the following changes were made:

- Central Manager: CONDOR_HOST = kitioto.ecsu.edu
- Collector_Name: VikeGrid
- Allow write: *

The "nano" command was also used to edit the "condor_config.local" file. The following changes were made to the file:

- Change Host: kitoto.ecsu.edu
- Collector Name: VikeGrid

Next, the status of condor was checked and a pearl script was created. To create the pearl script, the “nano” command was once again, as the following script was created:

```
#!/bin/bash
# Ensure network is all setup
sleep 100

# Ensure condor environment is
loaded
source /condor/condor-
7.4.0/condor.sh

# Start condor
/condor/condor-
7.4.0/sbin/condor_master
```

From here, everyone was given the rights to read, write, and execute “/condor/condor-7.4.0/condor.sh” file and the file was then added to CRON. To do this the following code was used:

“chmod 755 /usr/sbin/start_condor” and press [Enter]
 “ls –l /usr/sbin/start_condor” and press [Enter]
Type “echo “@reboot root /usr/sbin/start_condor” >> /etc/crontab

The XCODE, which is a feature on the Mac side of all of the computers that can compile C++ code, needed to be installed so the C++ source code could be compiled. To do this, the XCODE software was installed and “cd” command—used to move through drectories—was used to copy the XCODE from the developer tools to the hard drive, and the installation process was run (Peterson, Part 5: Shells, 2009).

Complete installation instructions can be found in the appendix section of the paper.

C. Collection of Data

It became a major obligation to try to find a prime number generator to use as a means to test the performance of VikeGrid. The team found out that there are a series of prime number generators called a “sieve.” A sieve is an algorithm made by a programmer that generates an increasingly productive amount of prime numbers. The team required Java, C, or C++ code to be able to submit the jobs to VikeGrid. The sieve that the team decided on using is the Sieve of Eratosthenes named after a Greek mathematician who formulated this equation to generate prime numbers.

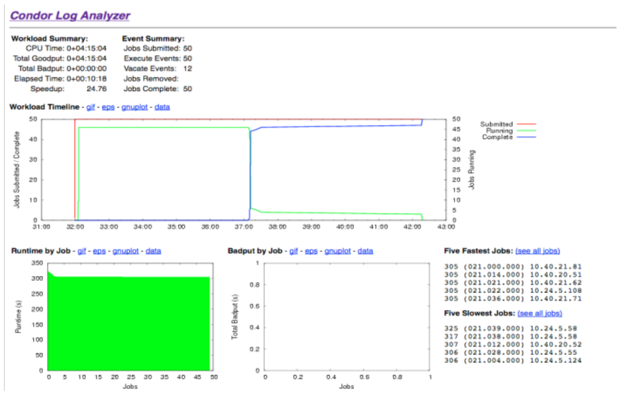
Once the team found the C++ code for the sieve, they compiled and ran the C++ code. The code was then modified then expanded generate prime numbers between 0 and 500,000.

The next step was to create a job submission script in order to utilize the condor pool named VikeGrid. The major factors that were needed to form this script was the output file for the C++ code, the arguments for the specified nodes that were desired to submit to VikeGrid, and the newly named log files after every submission. The information that was needed to create a submission file was:

- Output file
- The log file
- The queue which tells condor how many times condor will submit the jobs
- Arguments which defined specific number of nodes in which to submit jobs

The job submittal process began by establishing how many nodes were desired to run the jobs. It was determined that the team would do 1 node, then increase it to 2 nodes, then 4, 8, 16, 24,36, and lastly 48 nodes. After the jobs were done the log files were saved according to how many nodes were tested.

Using Condor Log Analyzer—Figure 2—the team found online, they were able to get the exact time it each node took to finish a job. These times, were then placed into Minitab in order to use the ANOVA feature to get a statistical representation of the different submission times.



Condor Log Analyzer
Workload Timeline - gif - eps - gnuplot - data
Jobs Submitted / Complete
Jobs Running
Submitted
Running
Complete
Runtime by Job - gif - eps - gnuplot - data
Runtime (s)
Jobs
Badput by Job - gif - eps - gnuplot - data
Total Badput (s)
Jobs

```
Workload Summary:
     CPU Time: 0+04:15:04
Total Goodput: 0+04:15:04
 Total Badput: 0+00:00:00
 Elapsed Time: 0+00:10:18
      Speedup:      24.76

Event Summary:
 Jobs Submitted: 50
 Execute Events: 50
 Vacate Events:  12
 Jobs Removed:
 Jobs Complete:  50

Five Fastest Jobs: (see all jobs)
305 (021.000.000) 10.40.21.81
305 (021.014.000) 10.40.20.51
305 (021.021.000) 10.40.21.62
305 (021.022.000) 10.24.5.108
305 (021.036.000) 10.40.21.71

Five Slowest Jobs: (see all jobs)
325 (021.039.000) 10.24.5.58
317 (021.038.000) 10.24.5.58
307 (021.012.000) 10.40.20.52
306 (021.028.000) 10.24.5.55
306 (021.004.000) 10.24.5.124
```

Fig. 2. Condor Log Analyzer

D. Statistical Methods and Tests that will be used to Analyze the Data

To compare the job submission times, a function in MiniTab, the analysis of variance, also known as ANOVA, was utilized. ANOVA is a statistical method for making simultaneous comparisons between two or more means. The function attempts to prove a significant relationship between variables—that there was or was not a change. To use ANOVA, a null-hypothesis is needed. The null hypothesis states:

$$H_o = \mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5 = \mu_6 = \mu_7 = \mu_8 \textbf{ (No Change)}$$
$$H_1 \neq \mu_1 \neq \mu_2 \neq \mu_3 \neq \mu_4 \neq \mu_5 \neq \mu_6 \neq \mu_7 \neq \mu_8 \textbf{ (Change)}$$

In the group's study, ANOVA was performed to determine if there was statistically enough variance between the means. This was tested within a 5% significance to note a difference in the job submission times. If the p-value –the probability that the variation between conditions may have occurred by chance, so the smaller the p-value, the more significant it is—of the ANOVA table was higher than the level of significance, then the hypotheses would be rejected (Overview of ANOVA, 2004). However, if the p-value was lower than 0.05, the hypotheses could be accepted.

The results of the ANOVA chart were placed in a Tukey or chart—Figure 3. This allowed the group to statistically see the difference between the eight job submission times.

```
One-way ANOVA: Number of Workers, Elapsed Time

Source  DF   SS    MS    F     P
Factor   1  1165  1165  8.22  0.012
Error   14  1983   142
Total   15  3148

S = 11.90   R-Sq = 37.01%   R-Sq(adj) = 32.51%


                             Individual 95% CIs For Mean Based on
                             Pooled StDev
Level            N   Mean  StDev  ---------+---------+---------+---------+
Number of Workers  8  17.13  16.83              (--------*--------)
Elapsed Time      8   0.06   0.07  (--------*--------)
                             ---------+---------+---------+---------+
                                 0        10        20        30

Pooled StDev = 11.90
```

Fig. 3. ANOVA Tukey chart

IV. Analysis of Data

A. *Results of the Statistical Analysis of Data*

After conducting the necessary research, it was discovered that a time difference is present when submitting a job over *x* nodes.

Also, when the 48 nodes were tested, only 46 nodes would respond, so the team used 46 nodes instead of the original 48.

B. *Tables, figures etc. used for data analysis*

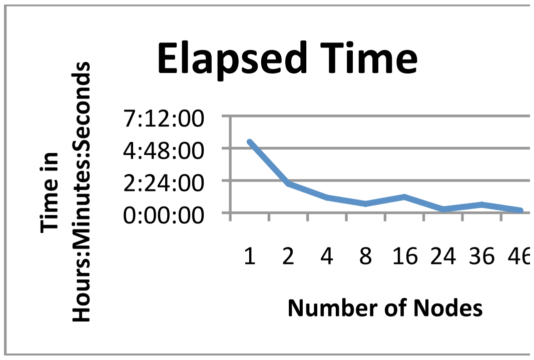


Fig. 4. Graph of the Elapsed Time it took to complete a job over *x* nodes

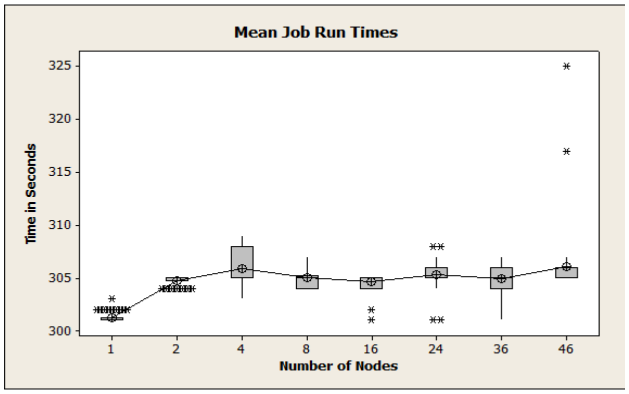


Fig. 5. Graph of the average time it took to complete a job.

C. *Decision about the hypothesis*

The team hypothesized that the elapsed time for a single node to complete a submitted job would take much longer than multiple nodes collaborating to complete the same job.

V. Summary, Conclusions, and Recommendations

A. *Conclusions Resulting from Statistical Analysis of the Data*

The Polar Grid team's ultimate goal was to find a Titanic prime—a number with at least a 1,000 digits. The first step toward achieving this goal was to find a prime number generator—the Sieve of Eratosthenes. The code generated was then used to test VikeGrid. In the use of this prime number generator the team tested how long it took to complete fifty jobs with various numbers of workers.

Before the study, the team hypothesized that the elapsed time for a single node to complete a submitted job would take much longer than multiple nodes collaborating to complete the same job—the reasoning behind this being that if the nodes worked together collaboratively instead of individually, the amount of time it will take for a job to finish would be much quicker than if a single node was working by itself.

After the study was completed, the team looked at the variance (the difference between points) that the ANOVA produced. When the graph was examined, the team concluded that it would be quicker for forty-six nodes to split a job then to just run the job on one node. However, the group noted that it was quicker for one node to run a job submission multiple times then to have the job split between *x* nodes and run multiple times.

Examining the ANOVA table, it became statistically clear that there was a difference in the elapsed time it takes for a job to run. This showed that the team's original hypothesis was correct. However the group noted that this was not the case for the 16 nodes test run and the 36 nodes test run. The team was surprised at this, because it was hypothesized that the times would decreases as the number of nodes increased.

Although this was true, it was not constant; for nodes 16 and 36, it took longer to split the job and run it over multiple nodes. The group was surprised by this and would like to further research what may have caused this.

B. *Shortcomings*

Over the course of the study, the team noticed some of the computers were unable to be completely installed with condor due to installation errors the team made—mainly typing errors. For example, in the "condor_config" file, instead of setting the Central Manager to, CONDOR_HOST = kitoto.ecsu.edu, the team originally set it equal to CONDOR_OST = kitoto.ecsu.edu. To fix this issue, the group went back and reinstalled Condor from scratch.

There was also an issue in checking the status of Kitoto due to the sbin paths being incorrect. The group originally had to log onto to Kitoto to check the status, but once the correct paths were installed the issue was fixed.

Xcode was needed to compile the Sieve of Erotosanes code, but due to the lab's Xcode being the incorrect version, this became an issue. To correct this issue, the Xcode was updated on all of the computers.

The team also needed to figure out the correct way to submit scripts to condor, and how to specify the number of nodes to be used in the submission of a job. To do this, the group had to specify the name(s) of the node(s) the job would use to run; this specification was made in the condor submission file.

Unfortunately, this year's Polar Grid team was not able to complete the ultimate research goal—finding a titanic prime number; the team was only able to test the job submissions for only one trial.

C. *Future Works*

The team would like for the future team's research of VikeGrid to use Titanic prime number, a prime number with thousands of number places, so that further testing of Kitoto can be done.

The Polar Grid team would like to have more trials done then have a comparison of the different times. This is so that there can be certainty that the times are correct and no errors were made.

These job submissions were only tested on the Mac side of each of the computers. For future teams, there should be tests done on the Linux and Windows side of the computers.

References

[1] *Computer Hope*. (2010). Retrieved from http://www.computerhope.com/unix/uchmod.htm

[2] *Developer tools xcode*. (2010). Retrieved from http://developer.apple.com/technologies/tools/xcode.html

[3] *ID 121: What are the differences between Analysis of Mean?* (2010). Retrieved April 07, 2010, from Minitab - Software for Quality Improvement: http://www.minitab.com/en-US/support/answers/answer.aspx?ID=121&P=0&R=255&M=43&S=44

[4] McElhearn, K. (2005). Chapter 2: Using Terminal. In K. McElhearn, *The Mac OS X Command Line: Unix Under the Hood* (pp. 9 - 10). Alameda, CA: Sybex.

[5] *Overview of ANOVA*. (2004). Retrieved April 8, 2010, from Improved Outcomes Software: http://www.improvedoutcomes.com/docs/WebSiteDocs/Statistics/Overview_of_ANOVA.htm

[6] Peterson, R. (2009). Part 1: Getting Started. In R. Peterson, *Ubuntu 9.04 Server: Administration and Reference* (p. 76). Alameda, CA: Surfing Turtle Press.

[7] Peterson, R. (2009). Part 5: Shells. In R. Peterson, *Ubuntu 9.04 Server: Administration and Reference* (p. 483). Alameda, Ca: Surfing Turtle Press.

[8] Vernon Brown Jr., M. J. (2008 - 2009). *Implementation of a Static Cluster.* Elizabth City, NC.

[9] *What is Minitab, and where can I access it?* (2005 - 2010). Retrieved April 06, 2010, from Indiana Univeristy - University Information TEchnology SErvices: http://kb.iu.edu/data/cagq.html

Appendix

```
/* Sieve Of Erathosthenes by Denis Sureau */
/* Arguments -- 1 is the number you want to check up to and 2 is the name of the output file */
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

void eratosthenes(int top, ostream & out )
{
  int* all = new int[1000000];
  int idx = 0;

  out << "1 ";

  int prime = 3;

  while(prime <= top)
  {
    bool skip = false;
    for(int x = 0; x < top; x++)
    {
      if(all[x] == prime) {
    prime += 2;
    skip = true;
        break;
      }
    }

    if( skip == true ) {
    continue;
    }

    out << prime << " ";
    int j = prime;
    while(j <= (top / prime))
    {
      all[idx++] = prime * j;
      j += 1;
    }

//skip:
    prime+=2;
  }
  out << std::endl;
  return;
}

int main(int argc, char **argv)
{
  ofstream output;
  int top = 500000;
  if( argc > 1 ) {
    top = atoi( argv[1] );
    output.open( argv[2] );
  }
  else {
    output.open( "output_primes.txt" );
  }

  eratosthenes(top, output);
  return 0;
}
```

/* Pearl Script */

```
#!/bin/bash
# Ensure network is all setup
sleep 100

# Ensure condor environment is loaded
source /condor/condor-7.4.0/condor.sh

# Start condor
/condor/condor-7.4.0/sbin/condor_master
```

/* Condor Submission File with Specific Nodes */

```
Universe = vanilla
Executable = primes

output = primecondor.out
error = primecondor.error
log = primecondor.log
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
Requirements = Name =="slot1@lan111-04m.local" || Name =="slot2@lan111-06m.local"

Queue 3
```