Scheduling in Grid Computing Systems

August 10, 2005

Phillipa Sessini
CPSC 531 Term Project
University of Calgary
Alberta,Canada

1 Abstract

Grid computing involves employing multiple machines to solve complex computing problems. For my project I have simulated a grid computing system based on a cluster of machines located at the University of Calgary. Data was collected from this system over a period of seven days. I have examined scheduling of jobs to nodes in the computing cluster. In particular I have simulated single site, centralized scheduling algorithms[3]. This means that I assumed all jobs enter one queue and are then assigned to machines based on scheduling criteria.

Through my analysis of the collected data I was able to conclude that the arrival rate of jobs entering the grid computing system followed a Poisson distribution with $\lambda = 3.125$ jobs per half hour. I found that the running time of processes followed a negative exponential distribution with a mean of 496.3minutes. Using these distributions I simulated this system using GPSS-H. From my results I can conclude that in a system where the majority of jobs require less than 4 processors it is reasonable to allow these jobs to contend for 4-processor machines. Also, using a most nodes available selection strategy versus a random selection strategy does not seem to enhance performance significantly.

2 Introduction

In large scale computing systems such as grid computing systems, there are often large amounts of resources available to be used for computing jobs. Since these resources can cost up to millions of dollars[6] maximizing their utilization is an important problem. Scheduling in a grid computing system is not as simple as scheduling on a multi-processor machine because of several factors. These factors include the fact that grid resources are sometimes used by paying customers[2] who have interest in how their jobs are being scheduled. Also, grid computing systems usually operate in remote locations[6] so scheduling tasks for the clusters may be occurring over a network. It is because of these reasons that looking at scheduling in grid computing is an interesting and important problem to examine.

3 Methodology

3.1 Data Collection

Data was collected from the Western Canada Research Grid. In particular data was taken from the Lattice computing cluster located at the University of Calgary. This cluster of machines consists of approximately 144 processors based on HP AlphaServer technology[4]. This is described further in the System Architecture section. The data was collected over a period of seven days.

3.2 Statistical Analysis of Data

3.2.1 Arrival Rate and Interarrival Times

To determine the distribution of arrival times in the system I have examined jobs arriving in half hour intervals over the period of one day. The frequency distribution of this is plotted in Table A1. From this data it can be determined that the average number of jobs arriving in a half hour (λ) is 3.125 jobs. The variance is 32.54 and the standard deviation is 5.70 jobs. Using this information we can perform a χ^2-test to determine if the arrival rate follows a Poisson distribution.

Table 1: χ^2-test for Poisson distribution of job arrival rates in a grid computing system.

i	O_i	E_i	$\frac{(O_i-E_i)^2}{E_i}$
0-1	22	8.70	20.33
2	9	10.30	0.16
3	3	10.73	5.57
4	3	8.38	3.45
5	2	5.24	2.00
6-37	9	4.66	4.05
			$\chi^2 = 35.57$

Using the table on page 584 of [1] $\chi^2_{v,\alpha} \doteq 55.8$ where $\alpha = 0.05$ and $v = 46$. Since $\chi^2 < \chi^2_{v,\alpha}$ at the 0.05 level of significance there is not enough evidence to assume that the arrival rate does not follow a Poisson distribution. Therefore by the relationship of the negative exponential distribution and the Poisson distribution we can infer that the interarrival times will follow a negative exponential distribution with a mean of $1/\lambda$

3.2.2 Running Time of Jobs

Based on the data collected on the second of my seven day observation period, I have derived a distribution for the running time of processes within the system. Selecting to examine only one day rather than the whole seven days does not change my result for the distribution because the data exhibited self similarity and trends seen on a weekly level were similar to what was seen in a single day. Also, because of the volume of data collected examining one day is more feasible than examining the entire week.

Since the run times of jobs varied from 0-7762 minutes I have chosen to group the data to perform a Kolmogorov-Smirnov Test. Ungrouped frequency data can be seen in Table A2. From this table I have derived the following statistical measures for this data:

$$\mu = 496.3333$$

$$\sigma^2 = 2043326.423$$

$$\sigma = 1429.4497$$

Using these values I was able to perform the Kolmogorov-Smirnov Test as follows:

Table 2: Kolomogorov-Smirnov Test for Exponential distribution on run-time of processes in a grid computing system.

Range	f	Percent	Observed CDF	Expected CDF	Δ
0-499	56	0.7778	0.7778	0.6341	0.1437
500-999	8	0.1111	0.8889	0.8664	0.0225
1000-1499	3	0.0417	0.9306	0.9512	0.0207
1500-1999	1	0.0139	0.9444	0.9822	0.0377
2000-2499	0	0.0000	0.9444	0.9935	0.0490
2500-2999	0	0.0000	0.9444	0.9976	0.0532
3000-3499	0	0.0000	0.9444	0.9991	0.0547
3500-3999	1	0.0139	0.9583	0.9997	0.0413
4000-4499	1	0.0139	0.9722	0.9999	0.0277
4500-4999	0	0.0000	0.9722	1.0000	0.0277
5000-5499	0	0.0000	0.9722	1.0000	0.0278
5500-5999	0	0.0000	0.9722	1.0000	0.0278
6000-6499	0	0.0000	0.9722	1.0000	0.0278
6500-6999	0	0.0000	0.9722	1.0000	0.0278
7000-7499	0	0.0000	0.9722	1.0000	0.0278
7500-7999	2	0.0278	1.0000	1.0000	0.0000
Total:	72			Max:	0.1437

From this table we can see that $D = 0.1437$. Using the formula on page 586 of [1] we can see that $D_{0.05,N} = \frac{1.36}{\sqrt{N}}$. Since $N = 72$, $D_{0.05,N} = 0.1603$. Thus $D < D_{0.05,N}$ and there is not enough evidence to assume that the running time of processes in the grid computing system do not follow an exponential distribution.

3.3 System Architecture

In my project I have simulated a cluster consisting of the following groups of nodes:

- **Group A** 36 4-processor nodes of type A
- **Group B** 20 4-processor nodes of type B
- **Group C** 1 4-processor node of type C
- **Group D** 50 single processor nodes

For simplicity I have assumed that all jobs can go to any group provided the nodes of that group have enough processors to meet the requirements of the job. For example, a job requiring 4 processors could be serviced by nodes of type A,B or C. However, if a machine requires more than 1 node, all its nodes must be of the same type. That is, a job cannot be serviced by more than one group. Also, jobs requiring only 1 processor may be serviced by any of the groups. This hardware configuration is based on [4]

3.4 Algorithms Implemented

For my project I have implemented the random and most nodes available(MNA) selection strategies(described in Appendix B1). I have also employed a FCFS job scheduling algorithm. I also tried two variations of each selection strategy. In one variation nodes requiring less than 4 processors were not allowed to contend for 4-processor machines (the "restricted" variation) and in the other variation, nodes requiring less than 4 processors were allowed to contend for 4-processor machines if all the single processor machines were in use("non-restricted").

3.5 Implementation of Model

For my model each group of processors will be represented by a different storage type. The individual processors within the group will be represented by the number of each storage type. When jobs enter the system, the number of processors they will use(p_j) will be assigned to a transaction parameter. The amount of nodes used by the job (approximately $\lceil p_j/4 \rceil$) will also be assigned to a transaction parameter. The number of processors used by a

process is modeled based on the distribution seen in observed data(Table A3). Interarrival rates and running times of jobs are modeled based on the distributions observed in the collected data.

4 Analysis of Simulation Results

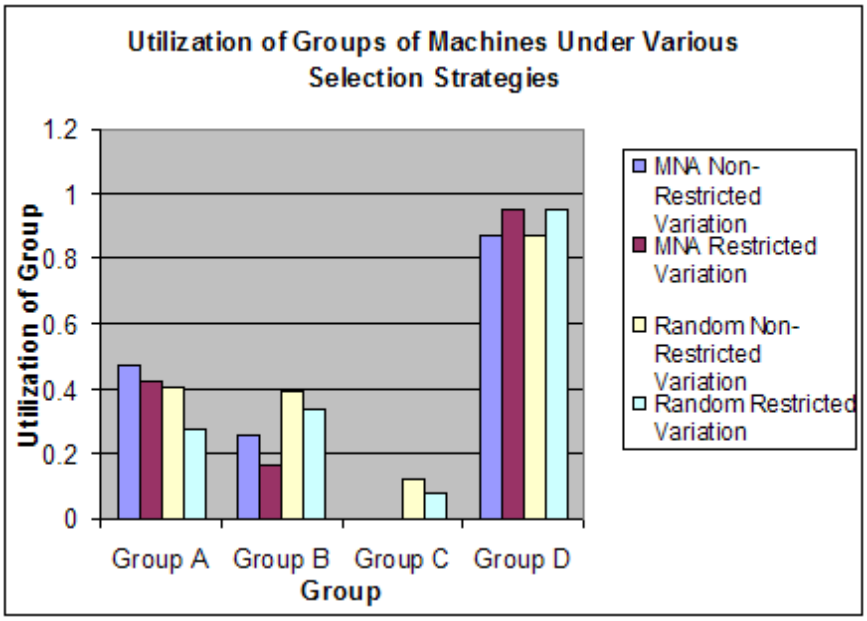

Figure 1: Average utilization of each group of nodes under various selection strategies

Figure 1 provides us with some insight into the load balancing capabilities of the random as well as the most nodes available selection strategies. It can be seen that in the most nodes available strategy, the group of nodes consisting of the smallest number of nodes (group 3 with 1 node in this case) does not get utilized until all the other groups are full, leaving it under-utilized unless system load is sufficiently high. The random selection strategy on the other hand provides much better balancing of utilization among the groups based on the amount of nodes they have. In all cases Group D, the group of single processors is utilized significantly more. This is because approximately 90 percent of the jobs require 1,2 or 3 processors. These jobs are directed

to the single processor machines to be fulfilled first before contending for 4-processor machines in the non-restricted case. In the restricted case we can see that utilization of these machines is increased. However this is not without a cost as can be seen when examining queuing delays of the restricted variants of these strategies.

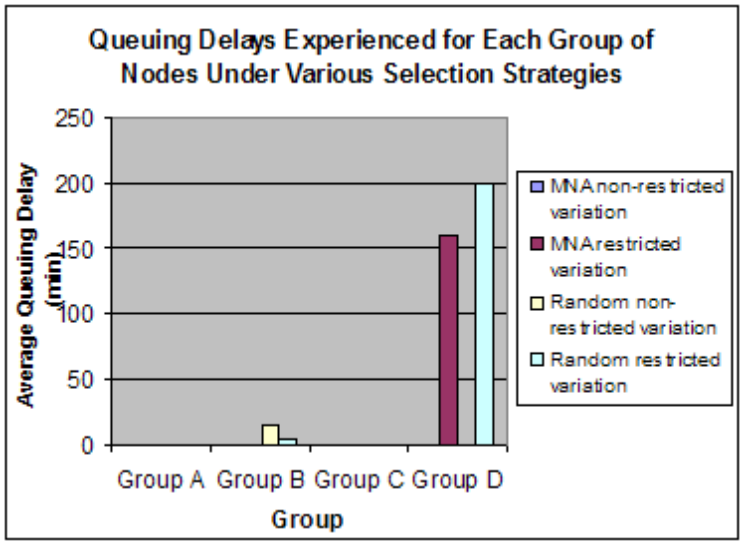

Figure 2: Average queuing delay for tasks going to each group of nodes under various selection strategies

When considering queuing delays of the various selection strategies (Figure 2) it becomes apparent that restricted variants of the selection strategies are not a good choice. When jobs requiring less than 4 processors are not allowed to contend for 4-processor machines there is a build up of jobs in the single processor group (Group D). In this case it may be because such a high percentage of the jobs did not require four or more processors. In a system where the majority of jobs require four or more processors, limiting jobs that do not require a full 4-processor node to single processor machines might provide better performance. Similar to queuing delays, over all response times (Figure 3) are increased with the restriction put on nodes that require fewer than four processors.

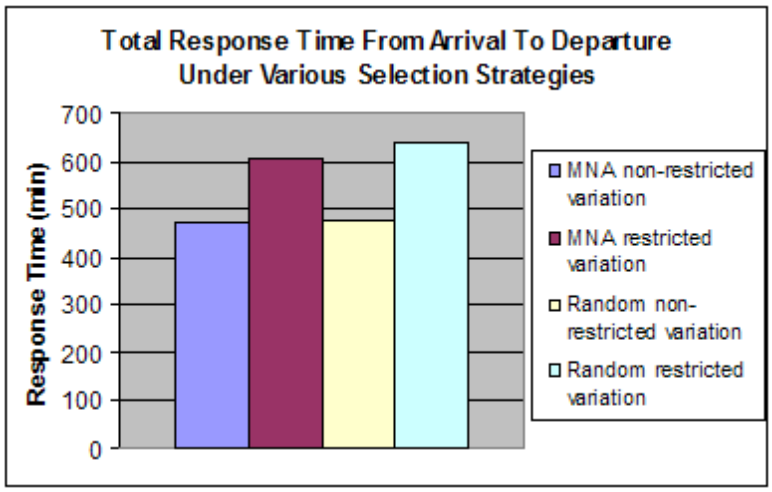

Figure 3: Average total response time calculated from job arrival to completion as observed in GPSS-H simulation results.

5 Conclusions

From the data collected in GPSS-H simulations I can conclude that the random and most nodes available selection strategies work approximately as well as each other. Restricting nodes which required fewer than four processors to single processor machines did not perform very well and introduced high queuing delays for single processor machines. This could be as a result of the fact that the majority of jobs arriving into the system did not require an entire 4-processor node.

Future research projects on this topic could include, examining when the restriction condition begins to perform better. For example, what percentage of jobs requiring four or more processors do we need to have before it becomes advantageous to restrict jobs requiring fewer processors to single processor machines. Also, examining the effects of memory/storage requirements on the flow of jobs through the system would be worth examining. However due to time contraints this was not feasible on this project.

6 Acknowledgements

I would like to thank the Western Canada Research Grid for providing me with the data for my project. Also, I would like to acknowledge Mark Fox for answering my many questions about grid computing.

References

[1] Banks J., Carson J., Nelson B. and Nicol D.: *Discrete-Event System Simulation 4th Edition*. Prentice Hall, New Jersey (2005)

[2] Buyya R., Steve Chapin S., and DiNucci D.: *Architectural Models for Resource Management in the Grid*. GRID 2000 IEEE/ACM International Workshop on Grid Computing (2000)

[3] Hamscher V., Schwiegelshohn U., Streit A., and Yahyapour R.: *Evaluation of Job-Scheduling Strategies for Grid Computing*. GRID 2000 IEEE/ACM International Workshop on Grid Computing (2000)

[4] Western Canada Research Grid: U of C Hardware.
http://www.westgrid.ca/support/Facilities#head-a37ce88f53df3ac15b3dd85bedbf6a892ef655ab).
Last modified: 2005-06-14

[5] Western Canada Research Grid: WestGrid Resource Allocation Policy.
http://www.westgrid.ca/downloads/RACFinal.pdf.
Accessed on: 2005-07-16.

[6] Western Canada Research Grid: http://westgrid.ca.
Accessed on: 2005-07-16.

Appendices

A Definitions

Cluster - the set of all groups of nodes in the system.

Group - for the purposes of this project group will refer to a collection of machines of the same type to which a job can be submitted.

Node - a computer. It may consist of 1 or more processors.

Selection strategy - the process by which a node is selected for a job to go to.

Scheduling algorithm - the process that selects which job in the queue should be considered next.

B Algorithm Descriptions

B.1 Selection Strategies

Random
This is a semi-random selection procedure to decide which node a job should go to. It is not totally random because jobs requiring only 1 processor will automatically be fielded to the group of single processor nodes to save the 4-processor nodes for larger jobs. In contrast, jobs requiring more than 1 processor will randomly select a group of 4 processor nodes to join. This is done based on a distribution created by taking $P(X = G_i) = \frac{|G_i|}{T}$ where $|G_i|$ is the size of the group to be joined(in nodes) and T is the total number of nodes in the cluster.
Most Nodes Available
Similar to the random algorithm, jobs requiring only 1 processor will, by default, go to the single processor machines. Jobs requiring more than 1 processor will select a group of 4 processor machines to go based on which group has the most nodes available.

B.2 Scheduling Algorithms

First Come First Served (FCFS)
In this algorithm jobs are serviced in the order that they will arrive.

C Raw Data

Due to privacy restrictions on my data I am unable to provide raw data in this report.

D Frequency Tables

Table A1: table illustrating the variation of f, the frequency of x, the number of jobs being submitted in a half hour interval.

x	f	xf	x^2f	x	f	xf	x^2f
0	17	0	0	19	0	0	0
1	5	5	5	20	0	0	0
2	9	18	36	21	0	0	0
3	3	9	27	22	0	0	0
4	3	12	48	23	0	0	0
5	2	10	50	24	0	0	0
6	3	18	108	25	0	0	0
7	3	21	147	26	0	0	0
8	1	8	64	27	0	0	0
9	0	0	0	28	0	0	0
10	0	0	0	29	0	0	0
11	0	0	0	30	0	0	0
12	1	12	144	31	0	0	0
13	0	0	0	32	0	0	0
14	0	0	0	33	0	0	0
15	0	0	0	34	0	0	0
16	0	0	0	35	0	0	0
17	0	0	0	36	0	0	0
18	0	0	0	37	1	37	1369
				total	48	150	1998

Table A2: table illustrating the variation of f, the frequency of x, the run time of a job submitted to the cluster.

x	f	xf	x^2f	x	f	xf	x^2f
0	11	0	0	163	0	0	0
1	10	10	10	...			
2	4	8	16	178	0	0	0
3	5	15	45	179	1	179	32041
4	1	4	16	180	0	0	0
5	2	10	50	181	0	0	0
6	2	12	72	182	0	0	0
7	2	14	98	183	1	183	33489
8	1	8	64	184	0	0	0
9	0	0	0	...			
10	5	50	500	190	0	0	0
11	2	22	242	191	1	191	36481
12	1	12	144	192	0	0	0
13	0	0	0	...			
14	1	14	196	211	0	0	0
15	0	0	0	212	1	212	44944
...				213	0	0	0
63	0	0	0	...			
64	1	64	4096	320	0	0	0
65	0	0	0	321	1	321	103041
...				322	0	0	0
142	0	0	0	...			
143	1	143	20449	505	0	0	0
144	0	0	0	506	1	506	256036
...				507	0	0	0
150	0	0	0	...			
151	1	151	22801	522	0	0	0
152	0	0	0	523	1	523	273529
...				524	0	0	0
161	0	0	0	...			
162	1	162	26244	528	0	0	0

(continued on next page)

x	f	xf	x^2f	x	f	xf	x^2f
529	1	529	279841	1828	0	0	0
530	0	0	0	1829	1	1829	3345241
...				1830	0	0	0
578	0	0	0	...			
579	1	579	335241	3766	0	0	0
580	0	0	0	3767	1	3767	14190289
...				3768	0	0	0
584	0	0	0	...			
585	1	585	342225	4042	0	0	0
586	0	0	0	4043	1	4043	16345849
...				4044	0	0	0
599	0	0	0	...			
600	1	600	360000	7739	0	0	0
601	0	0	0	7740	1	7740	59907600
...				7741	0	0	0
675	0	0	0	...			
676	1	676	456976	7761	0	0	0
677	0	0	0	7762	1	7762	60248644
...							
684	0	0	0				
685	1	685	469225				
686	0	0	0				
...							
1370	0	0	0				
1371	1	1371	1879641				
1372	0	0	0				
...							
1377	0	0	0				
1378	2	2756	3797768				
1379	0	0	0				

Table A3: Frequency distribution of x, the number of processors required by a process.

x	f	percentage of total	cumulative percentage
0	0	0	0
1	744	83.69	83.69
2	46	5.17	88.86
3	1	0.11	88.98
4	26	2.92	91.90
5	0	0.00	91.90
6	0	0.00	91.90
7	0	0.00	91.90
8	64	7.20	99.10
9	0	0.00	99.10
10	0	0.00	99.10
11	0	0.00	99.10
12	0	0.00	99.10
13	0	0.00	99.10
14	0	0.00	99.10
15	0	0.00	99.10
16	8	0.90	100.00
Total:	889	100.00	

E Simulation Results

Note that Group A is simulated by storage and queue 1, Group B by storage and queue 2, Group C by storage and queue 3, and Group D by storage and queue 4.

Table A4: Most nodes available selection strategy, non-restricted variation, table of storage utilization.

Storage:	1	2	3	4
Run 1	0.416	0.197	0	0.862
Run 2	0.473	0.215	0	0.874
Run 3	0.523	0.37	0	0.883
Average	0.470666667	0.260666667	0	0.873

Table A5: Most nodes available selection strategy, non-restricted variation, table of queuing delays(minutes).

Queue	1	2	3	4	SYS	WAIT
Run 1	0	0	0	0	471.749	0
Run 2	0	0	0	0	464.488	0
Run 3	0.033	0.029	0	0	485.692	0.007
Average	0.011	0.009666667	0	0	473.9763333	0.002333333

Table A6: Most nodes available selection strategy, restricted variation, table of storage utilization.

Storage	1	2	3	4
Run 1	0.36	0.1	0	0.914
Run 2	0.42	0.162	0	0.944
Run 3	0.491	0.233	0	1
Average	0.423666667	0.165	0	0.952666667

Table A7: Most nodes available selection strategy, restricted variation, table of queuing delays(minutes).

Queue	1	2	3	4	SYS	WAIT
Run 1	0	0	0	20.651	485.91 18.392	
Run 2	0	0	0	106.534	546.454	94.787
Run 3	0	0	0	353.287	782.592	315.014
Average	0	0	0	160.1573333	604.9853333	142.731

Table A8: Random selection strategy, non-restricted variation, table of storage utilization.

Storage:	1	2	3	4
Run 1	0.367	0.281	0.099	0.862
Run 2	0.401	0.376	0.146	0.872
Run 3	0.444	0.516	0.135	0.884
Average	0.404	0.391	0.126666667	0.872666667

Table A9: Random selection strategy, non-restricted variation, table of queuing delays(minutes).

Queue	1	2	3	4	SYS	WAIT
Run 1	0	0.176	0	0	471.759	0.01
Run 2	0.053	0	487.523	0	465.389	0.943
Run 3	0	15.926	0	0	486.807	1.194
Average	0.017666667	5.367333333	162.5076667	0	474.6516667	0.715666667

Table A10: Random selection strategy, restricted variation, table of storage utilization.

Storage:	1	2	3	4
Run 1	0.285	0.235	0.073	0.914
Run 2	0.295	0.379	0.156	0.944
Run 3	0.255	0.391	0	1
Average	0.278333333	0.335	0.076333	0.952667

Table A11: Random selection strategy, restricted variation, table of queuing delays(minutes).

Queue	1	2	3	4	SYS	WAIT
Run 1	0	0	0	20.651	485.91	18.392
Run 2	0	0	0	106.534	546.454	94.787
Run 3	0	11.887	0	470.804	873.92	420.263
Average	0	3.962333333	0	199.3296667	635.428	177.814