Fairshare Scheduling – A Case Study

Hung Bui, Wesley Emeneker, Amy Apon
University of Arkansas
{hbui, ewe, aapon}@uark.edu

Doug Hoffman
Acxiom Corporation
Doug.Hoffman@acxiom.com

Larry Dowdy
Vanderbilt University
Larry.Dowdy@vanderbilt.edu

Abstract

Scheduling and resource management are important in optimizing multiprocessor cluster resource allocation. Resources must be multiplexed to service requests of varied importance, and the policy chosen to manage this multiplexing can have enormous impact on throughput and response time. Fairshare scheduling is a way to manage application performance by dynamically allocating shares of system resources among competing users. The primary objective of this paper is to present an in-depth case study of fairshare scheduling In this case study, an in-depth sensitivity analysis of the various tunable parameters in fair-share scheduling techniques will be provided. The starting points for the study are scheduler log files collected from two production systems, one a production industry cluster and the second a university cluster. The approach to the case study is in two parts. First, using well-known techniques in the field, workload models for the two different environments are built and analyzed. Secondly, after the models are developed, they are presented to a fairshare scheduler under what-if scenarios. The experimental results are examined to evaluate the performance of fairshare scheduling.

1. Introduction

Scheduling computational jobs in distributed systems is a complex, challenging problem. Scheduling theory is concerned with the effective allocation of scarce resources to active entities in the system over time. Resources must be multiplexed to service request of varied importance [1], and the policy that is chosen to manage this multiplexing has a significant effect on the throughput and response time of the system. An ineffective scheduling policy may lead to poor performance of the whole system. For example, problems such as indefinite blocking of an individual job, or a convoy effect in which many jobs can be slowed by having to wait behind a single job, can be caused by a bad scheduling algorithm. Therefore, it is desirable to choose a scheduling algorithm that can maximize system utilization and throughput while at the same time providing acceptable levels of wait time and response time to jobs of varying importance. The fairshare scheduling algorithm is one of the solutions for the problems mentioned above.

The main objective of this paper is to present an in-depth case study of fairshare scheduling in two different operating environments. Fairshare scheduling allows the system to divide its resources "fairly" among competing users or groups. Most uses of fairshare scheduling assume some intuitive understanding of what will be "fair", but the term is often applied in different ways. What seems "fair" to one user may not seem "fair" to another.

The focus of the research is on analyzing the effects of modifying various parameters of the fairshare scheduling policy on the performance of classes of user jobs with different characteristics. The focus is not on determining whether the achieved effect is more or less "fair" to one user group or another, which is generally a business decision for the operation of the overall enterprise.

The approach in this paper is to create a synthetic workload that is derived from measurements on production clusters, and then to use discrete event simulation to study the effect of varying fairshare parameter values on the performance of the system and the resulting impact to different users. First, workload models are built and analyzed. Secondly, after the model workloads are examined, they are used as input for the study of fairshare scheduling. Various what-if scenarios are tested using the simulation model. The results of simulations are used to draw conclusions about the performance of the system using fairshare scheduling. Figure 1 shows the experimental steps used in the paper.

The remainder of this paper is structured as follows:

- Section 2 covers background material for Moab fairshare scheduling.
- Section 3 describes the process of modeling the workload from Acxiom cluster system and University of Arkansas Supercomputer.
- Section 4 focuses on simulation set up.

- Section 5 presents and analyzes simulation results

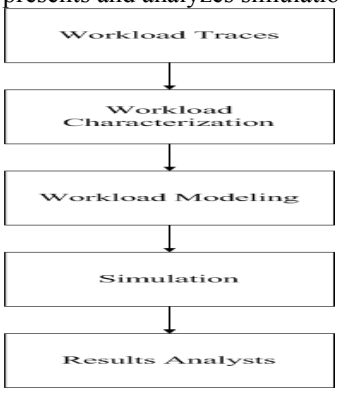

Figure 1: Experiment Process

2. Background

Fairshare scheduling incorporates historical resource utilization into job feasibility and priority decisions [2]. In fairshare scheduling each user or group is assigned a fixed number of shares. The shares represent a fraction of the resources that are available in the system. The most "important" users or groups can be assigned the most shares. A user's job cannot run in the system at all if the user has no share.

Fairshare algorithms typically use the share assignment and a dynamic priority formula to calculate a job's dynamic priority. In addition to more shares, higher priorities can be assigned to the most important users. There are multiple ways of allocating resources if two users with different priority compete for resources. For example, the most important user can have all the resources, or the most important user can get more resources according to some proportional distribution. Or, the two users can get the same level of resources. Fairshare scheduling variations include lottery scheduling, stride scheduling, max-min fairshare scheduling, and hierarchical share scheduling [3] [4] [5]. In each one of these fairshare scheduling algorithms different parameters are used to optimize the "fairness". In this paper, we focus on a case study of fairshare scheduling that is implemented in the Moab scheduler.

2.1 Fairshare parameters

Moab fairshare scheduling allows utilization targets (i.e., shares) to be set for users, groups, and classes. The target utilization is based on the usage during "windows" of time, and shares can be configured at the system level, at the group level, and at the user level. The dynamic priority of a job is a calculation based on the proportion of the target utilization that has been used. Parameters for the calculation of the dynamic priority include the Fairshare_Interval (FS_INTERVAL), Fairshare_Depth (FS_DEPTH), and Fairshare Decay (FS_DECAY) [6], defined as follows:

- **FS_INTERVAL**: Duration of each fairshare window.
- **FS_DEPTH**: Number of fairshare windows factored into the current fairshare utilization calculation.
- **FS_DECAY**: Decay factor applied to weighting the contribution of each fairshare window in the past.

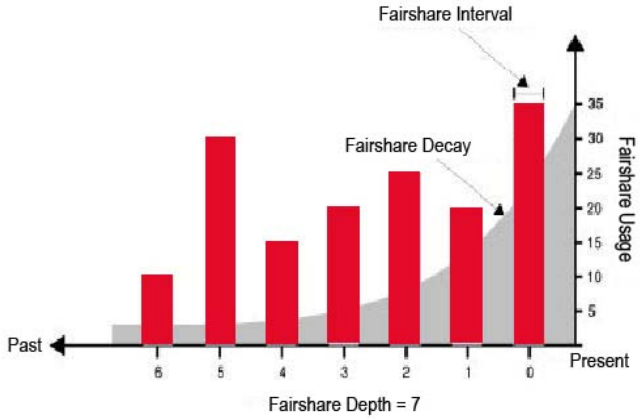

Figure 2: Fairshare Example Graph [from 6]

Figure 2 illustrates how the Moab fairshare system divides the present and past time into a number of distinct fairshare windows. In the figure Window 0 is the present window. Window 1 is the window in the most immediate past, Window 2 happened just before window 1, and so on. The FS_INTERVAL parameter specifies the duration of each window while the FS_DEPTH parameter indicates the number of windows to consider. The total time evaluated by the Moab fairshare scheduling algorithm is FS_INTERVAL * FS_DEPTH.

The FS_DECAY parameter allows the impact of an older fairshare window to be limited according to its age. The value of decay factor is usually is in the range between zero and one. If the FS_DECAY parameter is less than one then utilization during the most recent fairshare window contributes more to a total fairshare usage calculation than utilization during older windows [7]. The formula also allows the FS_DECAY parameter to be set to a value that is greater than one, in which case the utilization in older windows contributes more to the fairshare usage calculation than utilization in the present window.

FS_INTERVAL, FS_DEPTH, and FS_DECAY are global variables that are used for all groups and users in the system. In addition, the parameter FS_TARGET can be applied to each user, group, or account. The FS_TARGET parameters allow fairshare information to affect job priority.

Once all these parameters are known, they can be applied to the formula below to calculate the usage of a user:

$$Usage = \frac{\sum_{i=0}^{N} decay^{i} * t_i}{\sum_{i=0}^{N} decay^{i} * T_i}$$

Equation 1: Usage Calculation

The example below shows how to use FS_INTERVAL, FS_DEPTH and FS_DECAY to calculate the usage of a user and use FS_TARGET to decide the priority of a user.

Fairshare Window	*Total usage of user A*	*Total cluster usage*
0	*60*	*110*
1	*40*	*125*
2	*50*	*100*
3	*80*	*150*

Table 1: Fairshare Example Table

Suppose that parameters are set as follows:
FS_INTERVAL 12:00:00
FS_DEPTH 4
FS_DECAY 0.7
USERCFG [A] FSTARGET = 40
Based on the information provided in Table 1, the fairshare usage for user A would be calculated as follows:
*Usage = (60 + 0.7^1 * 40 + 0.7^2 * 50 + 0.7^3 * 80) / (110 + 0.7^1*125 + 0.7^2*100 +0.7^3*150) = 47.91%*
With these parameters the usage of user A is larger than the FS_TARGET that is set equal to 40. The priority of user A is reduced and the job submitted by user A have to wait in the queue longer than other jobs that belong to a user who has higher priority. However, if the FS_DEPTH is set to 3, the usage of user A is now calculated as:
*Usage = (60 + 0.7^1*40 + 0.7^2 * 50) / (110 + 0.7^1*125+ 0.7^2 * 100) = 38.65%*

With the FS_DEPTH set to 3, the fairshare usage of user A is smaller than the FS_TARGET and the priority of User A is increased. The high usage of User A in Window 3 is no longer counted in the fairshare usage calculation. In general, the longer the total time evaluated by fairshare (FS_INTERVAL * FS_DEPTH), the more history usage of a user or group is taken into account. A policy that includes a history of usage in previous windows allows users or groups who consume less system usage in previous windows to have more priority in the current window. However, the example illustrates how a small change in the value of a single fairshare parameter can change the priority of a user. Depending on how resources are then allocated, the parameters can impact the performance of the system significantly.

3. Workload Modeling

3.1 Workload Trace

Workload modeling always starts with measured data about the workload. This is often recorded as a trace, or log, of workload-related events that happened in a certain system. The workload trace for this paper is a workload trace that has been acquired from the job scheduler monitoring system of two kinds of clusters: Acxiom Corporation Cluster and University of Arkansas Red Diamond Supercomputer

3.1.1 The Acxiom Workload Trace

The workload trace in the case study was measured from a production cluster at Acxiom Corporation during March 2006. The Acxiom's scheduler records job attributes and performance data in an accounting log. Each workload element contains job information, such as job ID, submit time, queue time, runtime, number of compute nodes required, and other attributes. The job runtime is defined as the time between the job start time and the completion of the job. Job queue time is defined as the time between job submission and job start time. Each submitted job places a certain load on the system by demanding resources, which includes dedicated resources such as compute nodes, and shared resources such as network, storage, and other services [8]. Some attributes are intrinsic to the jobs, such as arrival time and nodes required. But job performance related attributes are dependent on specific system configuration and loading conditions, such as queue time and response time. Figure 3 below shows a portion of a log file that was obtained from Acxiom cluster system.

	A	B	C	D	E	F	G	H	I
1	Job ID		DOW	submit date	hour	minute	start datetime	stop datetime	nodes
2	O1719395	HIVEGRID	Tue	3/1/06	0	2	3/1/06 0:02	3/1/06 5:40	1
3	O1721185	HIVEGRID	Wed	3/1/06	5	9	3/1/06 5:09	3/1/06 5:40	4
4	O1721823	HIVEGRID	Wed	3/1/06	9	42	3/1/06 9:42	3/1/06 10:02	1
5	O1722182	RECORDOP	Tue	3/1/06	0	42	3/1/06 0:46	3/1/06 2:53	10
6	O1722787	HIVEGRID	Tue	3/1/06	0	5	3/1/06 0:05	3/1/06 0:08	16
7	O1722788	HIVEGRID	Tue	3/1/06	0	8	3/1/06 0:09	3/1/06 0:09	1
8	O1722831	HIVEGRID	Tue	3/1/06	0	2	3/1/06 0:02	3/1/06 0:03	1
9	O1722838	HIVEGRID	Tue	3/1/06	0	4	3/1/06 0:04	3/1/06 0:04	1
10	O1722839	HIVEGRID	Tue	3/1/06	0	5	3/1/06 0:05	3/1/06 0:08	16
11	O1722840	HIVEGRID	Tue	3/1/06	0	8	3/1/06 0:09	3/1/06 0:09	1
12	O1722842	HIVEGRID	Tue	3/1/06	0	6	3/1/06 0:06	3/1/06 0:09	4
13	O1722845	HIVEGRID	Tue	3/1/06	0	7	3/1/06 0:07	3/1/06 0:08	2
14	O1722848	HIVEGRID	Tue	3/1/06	0	9	3/1/06 0:09	3/1/06 0:17	1
15	O1722849	HIVEGRID	Tue	3/1/06	0	18	3/1/06 0:52	3/1/06 0:55	16
16	O1722850	HIVEGRID	Tue	3/1/06	0	55	3/1/06 0:56	3/1/06 0:56	1
17	O1722851	HIVEGRID	Tue	3/1/06	0	9	3/1/06 0:09	3/1/06 0:49	2
18	O1722852	RECORDOP	Tue	3/1/06	0	11	3/1/06 0:53	3/1/06 0:55	2
19	O1722853	HIVEGRID	Tue	3/1/06	0	20	3/1/06 0:46	3/1/06 0:46	1
20	O1722861	HIVEGRID	Tue	3/1/06	0	35	3/1/06 0:53	3/1/06 0:53	1
21	O1722862	HIVEGRID	Tue	3/1/06	0	53	3/1/06 0:53	3/1/06 0:57	16
22	O1722863	HIVEGRID	Tue	3/1/06	0	57	3/1/06 0:57	3/1/06 0:57	1
23	O1722865	HIVEGRID	Tue	3/1/06	0	47	3/1/06 0:48	3/1/06 0:52	16
24	O1722866	HIVEGRID	Tue	3/1/06	0	52	3/1/06 0:52	3/1/06 0:54	1
25	O1722867	HIVEGRID	Tue	3/1/06	0	52	3/1/06 0:52	3/1/06 0:53	1
26	O1722868	HIVEGRID	Tue	3/1/06	0	43	3/1/06 0:46	3/1/06 0:47	1
27	O1722869	HIVEGRID	Tue	3/1/06	0	46	3/1/06 0:46	3/1/06 0:47	1
28	O1722870	HIVEGRID	Tue	3/1/06	0	47	3/1/06 0:48	3/1/06 0:51	16
29	O1722871	HIVEGRID	Tue	3/1/06	0	51	3/1/06 0:51	3/1/06 0:51	1
30	O1722875	HIVEGRID	Tue	3/1/06	0	48	3/1/06 0:49	3/1/06 0:49	1
31	O1722876	HIVEGRID	Tue	3/1/06	0	49	3/1/06 0:49	3/1/06 0:52	16
32	O1722877	HIVEGRID	Tue	3/1/06	0	53	3/1/06 0:53	3/1/06 0:53	1
33	O1722881	HIVEGRID	Tue	3/1/06	0	50	3/1/06 0:51	3/1/06 0:51	1

Figure 3: Acxiom Workload Trace

3.1.2 University of Arkansas Workload Trace

Red Diamond is a cluster of 128 dual-processor computers with 64-bit Intel® Xeon™ processor. The computers are interconnected with an InfiniBand network. The main purpose of Red Diamond is to provide a high performance environment for researchers at the University of Arkansas. There are about 1500 jobs that are submitted to Red Diamond each month. The log files are stored in a MySQL database. Compared to the Acxiom enterprise cluster workload, the Red Diamond workload is much more diverse in that the application types range from data-intensive parallel applications to computation intensive serial applications. In addition, the job arrival pattern is also different from that of commercial systems. The workload trace file that was used in this case study was obtained during a four-month period from August to November of 2007. Each workload element in this trace file contains

almost the same job information as Acxiom's workload trace. Although the contents of both Acxiom and Red Diamond workloads are similar, there are some differences, such as number of users, number of groups, average arrival time, and average response time.

3.2. Workload Characterization

Workload characterization is a process to construct a concise description of the workload based on the input trace data and other information that may be known about the execution environment. The trade-off in workload characterization is between complexity and predictive power [9]. While the original trace data set contains very detailed information about the user load that is placed on the cluster system, it is difficult to construct a prediction workload from the trace data alone. Factors such as the number of nodes requested by a particular job, the overall run time, and the amount and rate of data read or written may not be uniform over a measurement period [10].

The first step of workload characterization is trace analysis. In trace analysis, the observed workload traces are studied carefully to provide insight to the composition of the workload and to identify job features for workload modeling. Well-known statistical analysis techniques are used to understand the overall characteristics, such as the distributions of job arrivals, node requirements, and runtimes. In this step, jobs can be divided into classes and each class then can be modeled individually. This hierarchical case-by-case approach also significantly simplifies the workload characterization process.

In this paper, two different techniques are used to divide jobs into classes. In the first technique, jobs are divided into classes based on job size, which is the number of compute nodes required by a job running on a system. The second way is to divide jobs based on user group [11]. Each of these techniques provides a different view of the performance of the whole cluster.

3.2.1 Job Size

In both the Acxiom cluster and Red Diamond, nodes are acquired by jobs and are used exclusively by those jobs until job completion. It is an important job attribute as the availability of nodes in the system and the number of nodes requested by a job largely determines the user job queue time in the system [11][12]. With Acxiom's workload, the workload characterization process was performed based on job size basis. In the trace, there are about 30 different groups of user, so that each group does not have many jobs. If one had done the workload characterization based on user group, it would have been difficult to analyze the trace. Therefore, job size is used rather than user group.

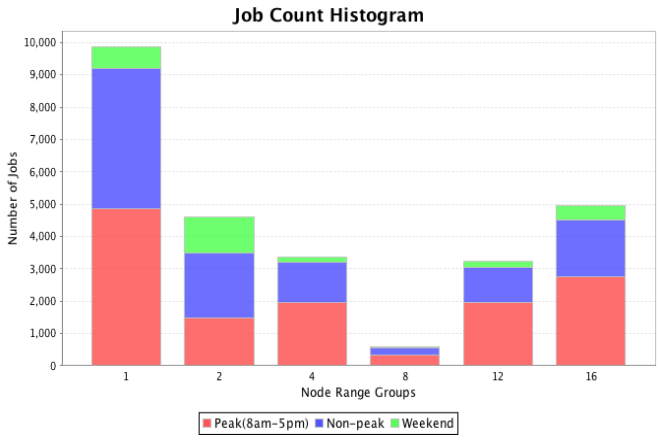

Figure 4: Job Distribution by Job Size

In Figure 4, jobs on the Acxiom cluster are divided into job size buckets and are also color-coded according to their arrival time ranges, i.e. peak-hours (8am-5pm), non-peak-hours (5pm- 8am), and weekend hours. As shown, the one, two, and 16 job size buckets contain the great majority of all jobs. Almost 40% of the jobs request only one node. The number of jobs in the 2_node group and the 16_node are group almost equal to each other. The numbers of jobs in those groups are relatively larger than the numbers of jobs in the 4_node and 12_node group. 8_node group has the smallest amount of jobs; it is about 10 times smaller than number of jobs in the 1_node group.

After dividing jobs into several classes according to job size, the average runtime and average arrival rate of each class were calculated. Both of these statistics can be obtained easily from our workload traces. It is worthy to mention that although the number of jobs in the 1_node group is larger than the number of jobs in other groups, the average run time of the 1_node group is the smallest. The average runtime of the 16_node group is the largest. Table 2 shows the average runtime of each group in the Acxiom workload. These statistics are used later to evaluate the performance of each group in the system.

Group	Average Runtime (minutes)
1_node	5.83
2_node	15.36
4_node	13.27
8_node	14.25
12_node	17.82
16_node	29.64

Table 2: Average Runtime per Group - Acxiom Workload

3.2.2 User-Group

Another way to perform job class separation is to divide jobs based on user group [8]. Usually in a system each individual user belongs to a specific group, and each group has different characteristic from other groups. For instance, for the Red Diamond cluster, three different user groups are defined, the Physics group, the Chemistry group, and others. Each of these groups has distinctive characteristics in term of job size, runtime and arrival rate.

Red Diamond's workload is very different from Acxiom's workload. Using a typical K-means clustering algorithm on the Red Diamond workload identifies about 40 different groups by job size. In contrast, the same algorithm applied to Acxiom's trace identifies about six groups by job size.
Figure 5 shows the jobs distribution of the Red Diamond workload trace according to user group. In the figure, the number of jobs in Others group is about 80% of the whole workload. Based on job count the Physics group is only about 15% of all jobs, and the remaining 5% of jobs are in the Chemistry group.

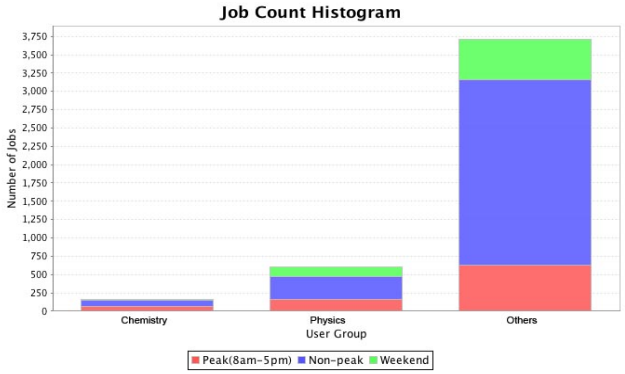

Figure 5: Job Distribution by User-Group

Although the number of jobs in the Physics group is small compared to the Others group, the average runtime and job size of these jobs is much larger. The average runtime of jobs in the Physics group is about 24 times larger than the average runtime of jobs in the Others group. Table 3 shows the average runtime and job size of each group in Red Diamond workload.

Group	Average Runtime (hours)	Average Job Size
Physics	47.8	13
Chemistry	4.89	8
Others	2.63	2

Table 3: Average Runtime per Group - Red Diamond Workload

3.3 Workload Modeling

Several synthetic workloads which have characteristics derived from the original workload traces were developed for use in this study. Although the original workload provides all of the job information needed in this study, it is too detailed to provide insight about the workload and it is also too complicated to be manipulated for performance studies. A useful workload model should be simple enough to be manipulated to represent futuristic or hypothetical workloads. Synthetic workload traces have a number of advantages over original traces such as adjustment ability, controlled modification ability, adding features ability, generalization ability and repetition ability. Workload attributes such as runtime, arrival rate, and number of jobs can be summarized by statistics or represented by a distribution function. Research by Lu shows that the hyper-exponentional distribution is a good match to the measured arrival rate distribution [5]. The average runtime and job size of each group in the synthetic workload are the same as the average runtime and job size of each group in the real workload trace.

The workload characterization module utilizes an XML file to express an output summary from the characterization process. The XML file contains several important attributes of each job class such as average arrival rate, average runtime and number of jobs. The combination of hierarchical clustering techniques and the XML output format allows a user to look at the different groups within the data in a level-by-level manner. This is helpful when performing trend analysis or identifying any abnormal sets of jobs in the data. The XML file can be fed into the Integrated Capacity Planning Environment (ICPE) tool to generate a synthetic workload [13].

4. Experimental Setup

The experiments are set up to study the impact of each fairshare parameter on the performance of the whole system. The measured performance in this study is response time.

When one parameter is examined, its value is changed throughout the experiment and the values of other parameters are fixed to a set value. In this way, it is simple to see how much effect that parameter has on the system. For example, when FS_INTERVAL is studied, its value is increased from 1 to 9, FS_DECAY is set to 0.7, FS_DEPTH is set to 7 windows, and FS_TARGET of each group is set to equal to the percentage runtime of that group. Because the main purpose of this experiment is to learn about FS_INTERVAL, so the values of both FS_DEPTH and FS_DECAY are selected so that they did not have much impact on the system.

It is a bit more complicated to obtain the FS_TARGET. It took several steps to determine this value. First, from the workload file, the total runtime of every job in the whole workload is calculated. Second, also from the workload file, the total runtime of every job in one group is calculated. Then we have:

*FS_TARGET = (total runtime of group/total runtime of the whole workload)*100*

Each experiment is run several times to observe the change in response time.

5. Result and Analysis

In this section, all the results are presented and analyzed. The first three experiments show the effect of each fairshare parameter such as FS_INTERVAL, FSDECAY, and FS_TARGET. Since both FS_INTERVAL and FS_DEPTH represent the evaluated time by fairshare scheduler, this section only shows the result of FS_INTERVAL. The next

four experiments provide the graphs resulting from study the combination of two parameters such as FS_INTERVAL and FS_DEPTH, or FS_INTERVAL and FS_DECAY. Another way to study fairshare is to increase the intensity of the arrival rate of a workload to observe the effect on the performance. This will be shown in the last two experiments.

5.1 Fairshare Interval

The first experiment studies the impact of the FS_INTERVAL parameter on both Red Diamond and Acxiom workloads.

5.1.1 Red Diamond Workload

FS_DEPTH = 7 windows	FS_TARGET [Physics] = 65
FS_DECAY = 0.7	FS_TARGET [Chemistry] = 20
	FS_TARGET [Others] = 15

Table 4: Fairshare Interval on Red Diamond Data

Figure 6 shows the change in average response time of each group as the value of FS_INTERVAL increases. It appears that, as the value of FS_INTERVAL increases, the average response time of both the Chemistry and Others group decrease somewhat, but the average response time of the Physics group increases.

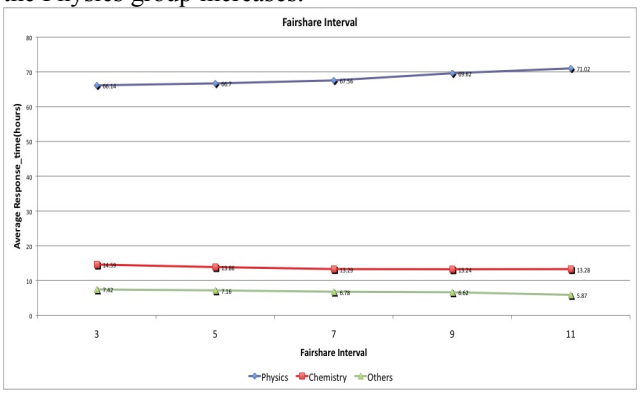

Figure 6: Fairshare Interval on Red Diamond data

As mentioned in the workload characterization section, the average runtime of Physics group is much higher than the average runtime of both Chemistry and Others groups. Also, the average job size of the Physics group is 13 processors and 48 hours. Compared to the average job size of the "Others" group (4 processors and 2.6 hours),it is clear that the Physics group uses significantly more resources. The question is why the Physics group's jobs' average response time increases as the value of FS_INTERVAL increases? As the FS_INTERVAL increases, the total time evaluated by the scheduler in each window is longer, so it makes the total usage of the Physics group in each window larger. As the total usage increases, the fairshare algorithm reduces priority. This reduction in priority increases the wait time of each Physics job. As the average wait time of the Physics group becomes longer, the average wait time of other groups decreases.

Another issue that needs to be addressed is system utilization. In figure 6, when FS_INTERVAL equals 3 days, it appears that the average response times of all three groups are at their smallest. So why can we just set the FS_INTERVAL to 3 and get the lowest average response time? Figure 7 shows how the utilization changes as the FS_INTERVAL increases.

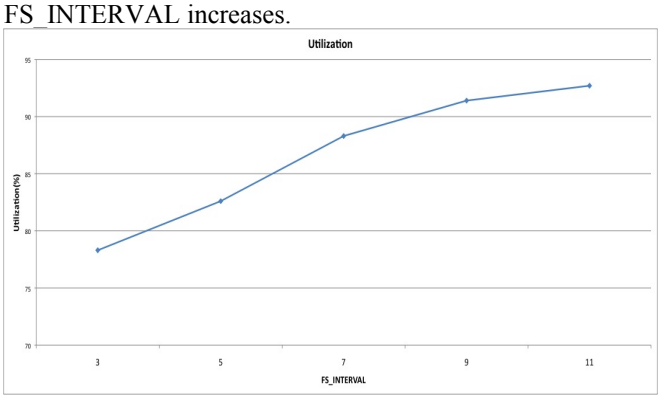

Figure 7: Utilization Red Diamond

In Figure 7, as the FS_INTERVAL increases, the utilization of the system also increases. The reason that the system utilization increases is because the average response time of the Physics group increases. When FS_INTERVAL equals 3, the average response time of the Physics group is small, which means the Physics group has high priority. As mentioned, the Physics jobs require a big average job size. So, for example, some Physics jobs have the highest priority and they all request 50 processors to run. If the system just has 30 processors available, then those jobs have to wait until there are enough processors. Those jobs will block all small jobs with lower priority and make 30 processors stay idle. This behavior lowers system utilization. As FS_INTERVAL increases, the average response time of the Physics group increases. If the Physics group's jobs do not have high priority, system utilization increases.

Therefore, when considering the value of FS_INTERVAL, both average response time and system utilization must be taken into account. It depends on the desired outcome of the system administrator to select the value of FS_INTERVAL.

5.1.2 Acxiom Workload

The Acxiom workload is characterized differently from the Red Diamond workload. However, in both of these workloads there is a group that dominates in term of both runtime and job size. In Red Diamond workload, that group is the Physics group. In the Acxiom workload, that group is the 16_node group. So, as the values of the FS_INTERVAL increases, it is expected that the average response time of

the 16_node group would increase. Figure 8 clearly supports this argument. However, Figure 8 shows not only the increase in response time of the 16_node group, but also the increase in the average response time of the 12_node group. Although, the increase in the 12_node group is not significant, it still shows that the 12_node group is the second dominant group in the workload.

FS_DEPTH =3windows	FS_target[1_node] = 14 FS_target[8_node] = 12
FS_DECAY = 0.7	FS_target[2_node] = 6 FS_target[12_node] = 15
	FS_target[4_node] = 15 FS_target[16_node] = 38

Table 5: Fairshare Interval on Acxiom Workload

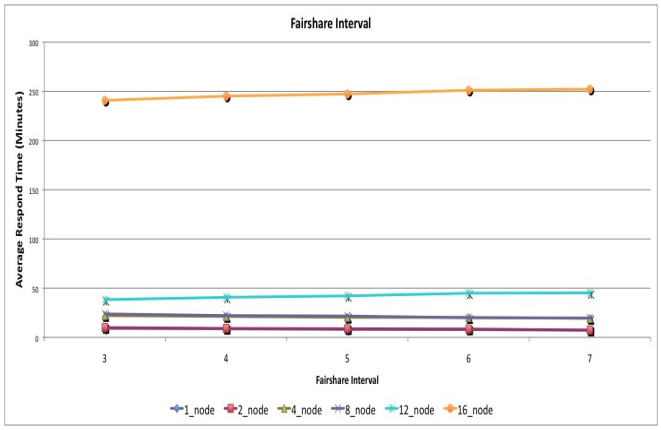

Figure 8: Fairshare Interval on Acxiom Workload

The utilization of the system during this experiment is shown in Figure 9. Similar to the Red Diamond workload, as the FS_INTERVAL increases, the system utilization also increases. The increase in average response time of the 16_node group makes the system utilization also increase.

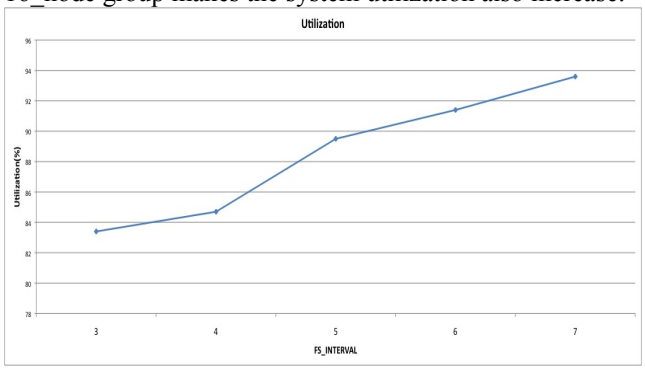

Figure 9: Utilization Acxiom Workload

5.2 Fairshare Decay

The second experiment is to study the impact of the FS_DECAY parameter on both workloads.

5.2.1 Red Diamond Workload

FS_INTERVAL = 7 days	FS_TARGET [Physics] = 65
FS_ DEPTH = 7 windows	FS_TARGET [Chemistry] = 20
	FS_TARGET [Others] = 15

Table 6: Fairshare Decay on Red Diamond Workload

Figure 10 shows the result of increasing the value of FS_DECAY. The result is not much different than the results shown for Interval variation. The only thing that stands out from the graph is that the increase in the average response time of the Physics group is more dramatic than the result with FS_INTERVAL. It increases from 64.32 to 71.61 compared to the last graph that increases from 66.12 to 71.38. From this experiment, the change in FS_DECAY has more impact on the system than the change in FS_INTERVAL.

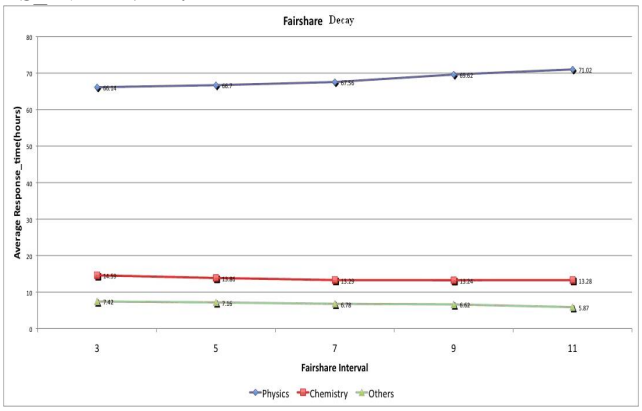

Figure 10: Fairshare Decay on Red Diamond Workload

Similar to previous experiment, as the FS_DECAY increases, the system utilization also increases.

5.2.2 Acxiom Workload

FS_DEPTH = 3 windows	FS_target[1_node] = 14 FS_target[8_node] = 12
FS_INTERVAL = 3 days	FS_target[2_node] = 6 FS_target[12_node] = 15
	FS_target[4_node] = 15 FS_target[16_node] = 38

Table 7: Fairshare Decay on Acxiom Workload

The conclusion with the Red Diamond workload can also be applied to Acxiom workload. The average response time of the 16_node group increases significantly and the average response time of other groups decrease. Figure 11 shows the result of this experiment.

.

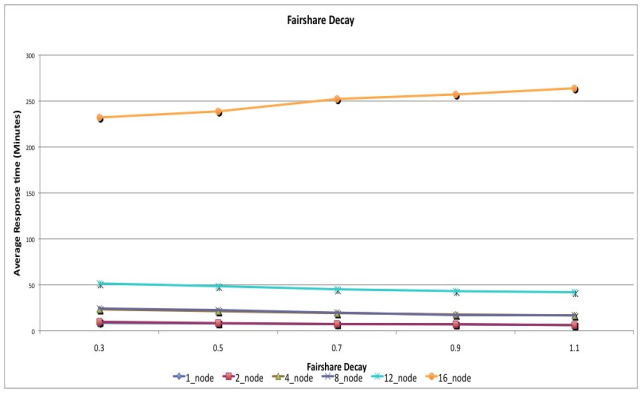

Figure 11: Fairshare Decay on Acxiom Workload

5.3 Fairshare Target

The next parameter that is examined is FS_TARGET. In theory, FS_TARGET may have more impact than other parameters because it has a direct influence on the priority of each group.

5.3.1 Red Diamond Workload

The way to set up this experiment is different compared to the previous experiments. FS_INTERVAL, FS_DEPTH and FS_DECAY are set to constant values. The values of FS_TARGET of each group are changed. In this case, the FS_TARGET of the Physics group will decrease and FS_TARGET of Chemistry and "Others" will increase. However, the sum of all FS_TARGET parameters must equal 100 since it is equal to the percentage runtime of each group.The maximum possible usage of the system is 100%, so the maximum possible FS_TARGET must also be 100.

FS_DEPTH = 7 windows	Run 1 Phys = 85 Chem = 10 Others = 5
FS_INTERVAL = 7 days	Run 2 Phys = 75 Chem = 15 Others = 10
FS_DECAY = 0.7	Run 3 Phys = 65 Chem = 20 Others = 15

Table 8: Fairshare Target on Red Diamond Workload

Figure 12 shows that as the FS_TARGET decreases, the average response time of the Physics group increases. A smaller FS_TARGET means lower priority, makes the jobs of the Physics group wait longer.

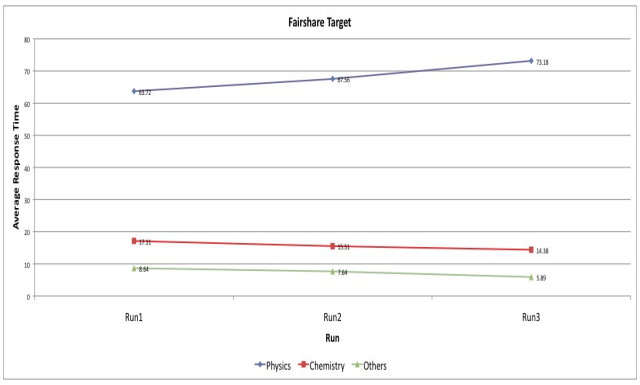

Figure 12: Fairshare target on Red Diamond Workload

Figure 13 shows the system utilization when varying FS_TARGET. The increase in average response time of the Physics group makes the system utilization also increase.

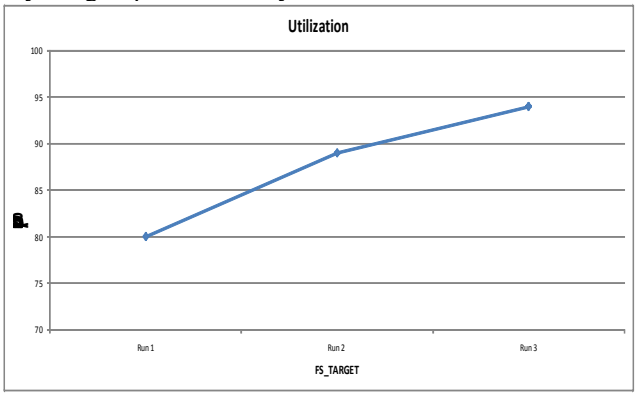

Figure 13: Utilization Red Diamond Workload

5.3.2 Acxiom Workload

This experiment is set up exactly like the previous experiment. The setup is shown in Table 9. Figure 14 shows the impact of changing the FS_TARGET on each group in the Acxiom workload. The results are even more interesting than what was shown in Figure 10. On the first run, the average response time of the 16_node group is even smaller than the average response time of the 12_node group. Having a much higher FS_TARGET is the reason why the average response time of the 16_node group is still smaller than the average response time of the 12_node group even though the average runtime and number jobs of the 16_node group is higher than the 12_node group. This result shows how much impact FS_TARGET has on the performance of each group. As the FS_TARGET of the 16_node group decreases, its average response time increases rapidly. The 12_node group, its average response time decreases, as the FS_TARGET increases.

FS_DEPTH = 3 windows FS_INTERVAL = 3 days
FS_DECAY = 0.7

Group	Run 1	Run 2	Run 3
1_node	FS_TARGET =10	FS_TARGET =12	FS_TARGE T=14
2_node	FS_TARGET =2	FS_TARGET =4	FS_TARGE T=6
4_node	FS_TARGET =11	FS_TARGET =13	FS_TARGE T=15
8_node	FS_TARGET =7	FS_TARGET =9	FS_TARGE T=11
12_node	FS_TARGET =11	FS_TARGET =13	FS_TARGE T=15
16_node	FS_TARGET =58	FS_TARGET =48	FS_TARGE T=38

Table 9: Fairshare Target on Acxiom Workload

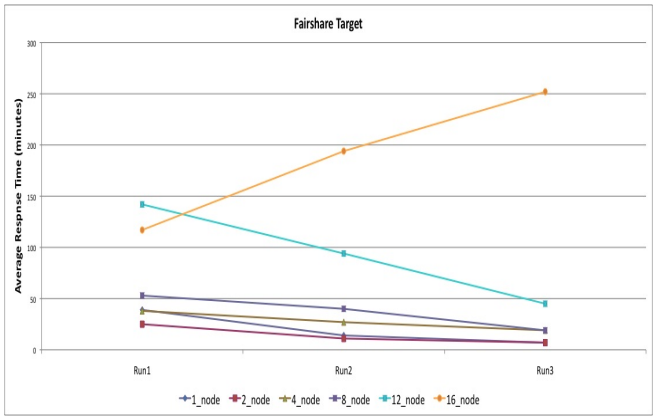

Figure 14: Fairshare Target on Acxiom Workload

Similar to the Red Diamond workload, the increase in average response time of the 16_node group increases system utilization. It now can be concluded that, in both workloads, if the average response time of the group with highest average job size and highest average runtime such as the Physics group or the 16_node group, increases, then the system utilization will also increase. Also, the utilization also increases in this case.

5.4 Fairshare Interval and Fairshare Depth

Up to this point, only one factor at a time has been changed during simulation. Although the results from the previous parts are good, the impact of varying the fairshare parameters individually is relatively small. Variations in response time were small, giving the impression that changing fairshare parameters affects workload execution very little. This section provides a different look at the results. Instead of just FS_INTERVAL or FS_DEPTH, it will be the combination of both parameters. The result analysis of this experiment is based on Red Diamond workload.

FS_DECAY = 0.7	FS_TARGET [Physics] = 65
	FS_TARGET [Chemistry] = 20
	FS_TARGET [Others] = 15

Table 10: Fairshare Interval and Fairshare Depth on Red Diamond Workload

In this experiment both FS_INTERVAL and FS_DEPTH parameters are varied. FS_DECAY and FS_TARGET remain fixed. Figure 15 shows the change in response time of both Physics and Chemistry groups. The x-axis is the FS_INTERVAL, the y-axis is the FS_DEPTH and the z-axis is the average response time. Similar to the graph in section 5.1, the average response time of the Physics group increases and the average response time of the Chemistry group decreases. However, compared to figure 6, it is clear that the gap between two surfaces gets bigger as the parameters increase. This means the change in the average response time of both Physics and Chemistry are more significant than the previous experiments.

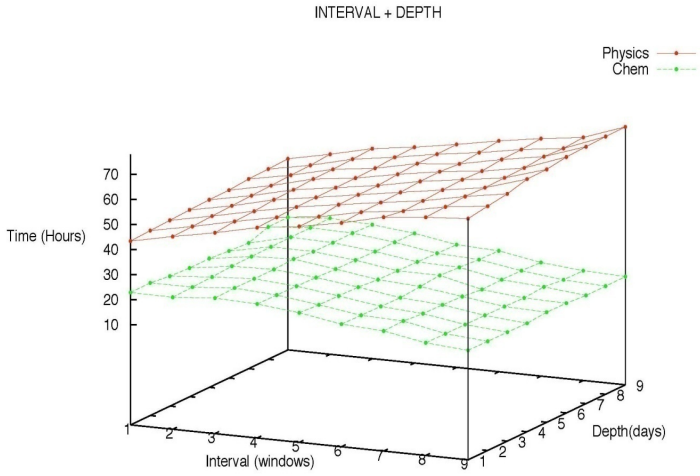

Figure 15: Fairshare Interval and Fairshare Depth on Red Diamond Workload

6. Experimental Summary

The goal of this research is to conduct a performance analysis of fairshare scheduling for two different case study environments. First, a workload model was built and analyzed. The workload models have been built to capture all the essential attributes of the actual workloads. Statistical analysis and graphical techniques have been used to perform the actual workload analysis. The second part is to study the impact of fairshare scheduling on the model workloads. In order to evaluate the impact of each fairshare parameter on the performance, ten experiments have been completed. As the value of each parameter is varied, the results are different. Following are the highlights of all experiments:

- As FS_INTERVAL, FS_DEPTH, and FS_DECAY increase, the average response time of the group, which has a high average runtime, high average job size, and large number of jobs, increases. On the other hand, the average response time of the group that has a smaller average runtime, average job size and number of job, decreases.
- Changes to FS_DECAY have more impact than changes to FS_INTERVAL or FS_DEPTH. When FS_DECAY increases, the change in response time is more significant than when FS_INTERVAL increases.
- Changes to FS_TARGET have the most impact on the performance. One can easily change the whole outcome of a system by changing FS_TARGET.
- Increasing the values of two parameters at once will change the performance more rapidly than just one parameter

7. Conclusion

Fairshare scheduling is a dynamic scheduling algorithm to use in clusters and grid. This research has presented several experiments that study the effect of each fairshare parameter. From the results, although it appears that varying FS_TARGET is the best way to control usage, other paramters such as FS_INTERVAL, FS_DEPTH, and FS_DECAY still have a big part in the whole algorithm.

Acknowledgments

This work is supported by research grant from Acxiom and by the National Science Foundation under MRI Grant #0421099.

References:

[1] B.W.Lampson, "A Scheduling Philosophy for Multiprocessing Systems", *Communications of the ACM,* Volume 11, No. 5, May 1968.

[2] E. Bolker and Y. Ding, "On the Performance Impact of Fair Share Scheduling" *need full reference information here*

[3] D.H.J. Epema and J.F.C.M. de Jongh, "Proportional-Share Scheduling in Single-Server and Multiple-Server Computing Systems", *Performance Evaluation Review,* Vol. 27, Issue 3, December 1999, pp. 7-10.

[4] C.A. Waldspurger and W.E. Weihl, "Lottery Scheduling: Flexible Proportional Share Resource Management", In *Proceedings of the First USENIX Symposium on Operating System Design and Implementation.* Monterey, CA, USA, November 1994.

[5] C.A. Waldspurger and W.E. Weihl, "Stride Scheduling: Deterministic Proportional-Share Resource Management", Technical Report. MIT, June 1995.

[6] Moab cluster suite. http://www.clusterresources.com/pages/products/moab-cluster-suite.php, March 2008. website.

[7] P.Goyal. H. M. Vin, and H.Cheng,"Star-Time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks", *IEEE/ACM Transaction on Networking,* Vol. 5, No. 5, October 1997. Pp. 690-704.

[8] B. Lu. "An Integrated Capacity Planning Environment for Enterprise Grids", *P.h.D Dissertation*, University of Arkansas, 2007. Reference the CMG paper also

[9] Feitelson, D. (2008). Workload Characterization and Modeling. www.cs.huji.ac.il/labs/parallel/workload/

[10] Uri Lublin and Dror G. Feitelson. "The workload on parallel supercomputers: modeling the characteristics of rigid jobs", In *Job Parallel Distributed Computing*, 2003. *Need full reference*

[11] Baochuan Lu, Michael Tinker, Amy Apon, Doug Hoffman, and Lawrence Dowdy," Adaptive automatic grid reconfiguration using workload phase identification". In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, Washington, DC, USA, 2005.

[12] Mark S. Squillante, David D. Yao, and Li Zhang, "The impact of job arrival patterns on parallel scheduling", *SIGMETRICS Perform. Eval. Rev*., 26(4):52–59, 1999.

[13] B. Lu. "An Integrated Capacity Planning for Enterprise Grid", *Ph.D Dissertation*, University of Arkansas, 2007.