Recourse Management Using a Fair Share Scheduler

Suja Cherukullapurath Mana *scheruk1@masonlive.gmu.edu*
Computer Science Department, George Mason University
Fairfax,VA 22030

Abstract

Resource management is a vital task of all operating systems. It is the responsibility of operating system to ensure that all programs requesting resources are getting resources in a timely manner. Various recourse allocation strategies are there which provide guidance for operating systems to make resource allocation decisions. This article studies about the resource management using a fare share scheduler. The fair share scheduler ensures that resources are allocated to programs in an efficient manner and this ensures fairness in resource allocation.

Keywords: Fairness, Fair Share Scheduling.

1. INTRODUCTION

Fairness is an important, challenging feature of any operating system resource scheduler, especially when there are a large number of programs waiting for resources. Real-time interactive and multimedia application servers will have a large number of service request occurring at the same time. A fair share scheduler is essential in this case to ensure that no processes starve indefinitely for resources. The fair share scheduler ensures that a specific portion of resources will be allocated to all programs [1]. The fair share algorithm requires that each user should be able to specify their required share of resources [1] and also no program should prevent the scheduler to allocate resources to other programs [1].

Similarly in a virtual machine environment where more than one operating system are running on a single machine, proper resource management is necessary to ensure good results.

Main objectives of a fair share scheduler are to ensure fairness, fast response time and load spreading without making any programs wait for too long. The users can view the fair share scheduler as a scheduler which ensures that all resources will be allocated in a fair manner. The resource allocation is defined based on the share and usage history [6]. A diagram showing users view of fair share scheduler is shown below [6]. In the diagram the word ‘Share’ means users authority to do work on a particular machine. A user with more shares can do more work than a user with fewer shares. The term ‘Usage’ indicates a measure of amount of work that each user performed in a particular system. So as per the user centric view, each user can expect to see that as their usages are increasing dynamically, their response time is getting worse’ [6].

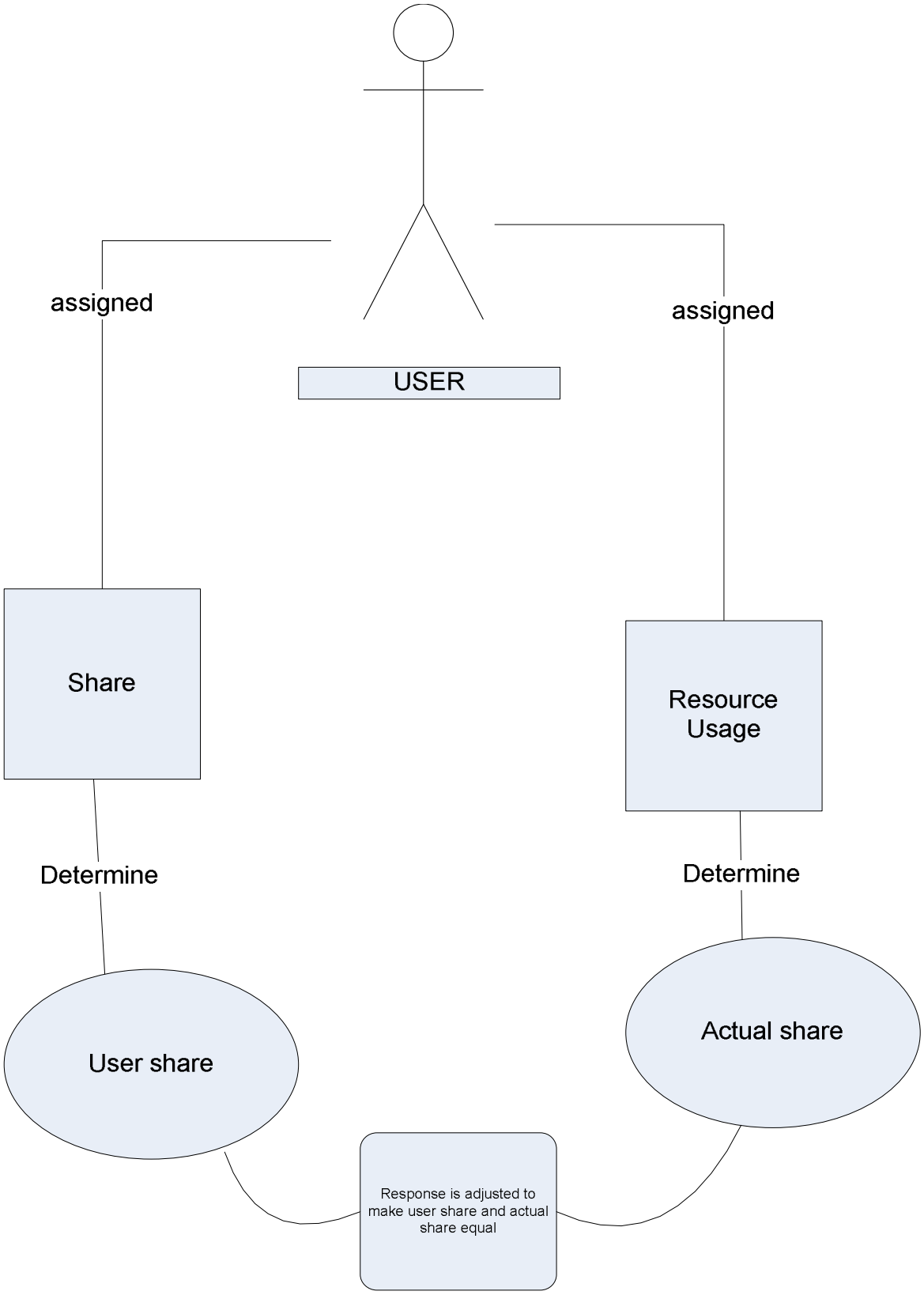

FIGURE 1: A user view of fair share scheduler

In some systems the resource allocation is done by a 'fixed budget' model [6]. As per this model of resource sharing, users have fixed budget allotted to them for resource usage. The budget will be reduced as they are using resources. In a nutshell this model ensures that processes whose owner is ready to pay more will get hier share of resources. To have better control over the allocation, certain constrains are being used . For example , there will be an upper limit for daily disk space usage , daily and weekly connect limits are also available[6].

The paper is organized in such a way that in the following sections it describes about the methodology of fair share scheduler , advantages and disadvantages of these methodologies, then some related works and research gaps followed by conclusion and future works.

2. METHODOLOGIES

The ' Fare share scheduling' ensures that each user is receiving required resources in a fair manner. That is the operating system is not just dividing available resources among available user, but doing the resource allocating on a need based manner. Various methodologies are being studied to ensure fare share allocations.

One methodology [6] describes concept of share from a user and a program perspectives. The user level scheduling involve steps like

- Usage of each user is updated by adding charges incurred by each processes since last update and then adjust it by appropriate constant[6]
- Update resource consumption records[6]

The program level scheduler involve following steps

- . Activation of process: Update the cost encounter by currently running process and then select a processes with lowest priority for running [6]
- Adjust priorities of process: According to the usage, share and number of active processes, adjust the priority of currently running processes.

Another method of determining share of each processes towards resource usage is the lottery scheduling [8] . Lottery scheduling is an efficient way of implementing 'proportional share' resource management ensuring that each process will get resources proportional to their shares. In this scheduling resource rights are represented by lottery tickets and resources are allocated to a user with a winning ticket [8]. This scheduling methodology is fair probabilistically [8]. Resource allocation to each user is proportional to number of lotteries each processes hold. The representation of resource right as lottery tickets is also helpful for modular resource management. To ensure that a user using only a fraction of its allocated resource is not preventing from getting CPU , a compensation ticket will be given to such users. This compensation tickets will raise the priority of user[8].

The following figure shows how lottery based scheduling works [8].

. Total tickets = 20

10^{th} ticket is selected randomly for comparison.

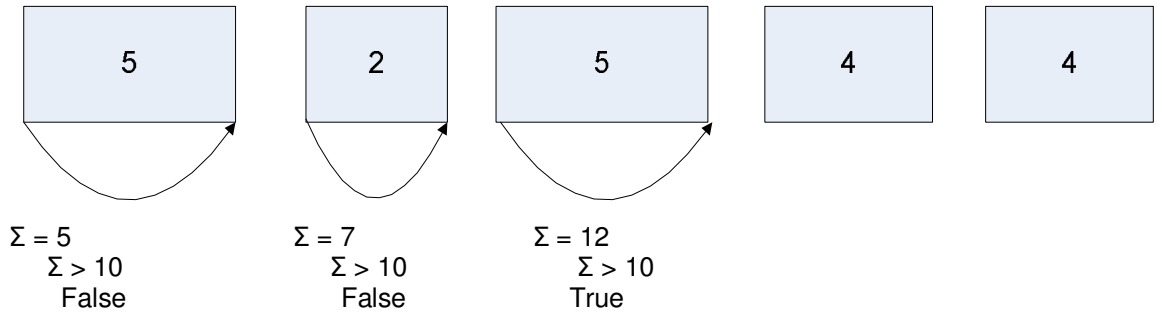

FIGURE 2: Lottery based scheduling example

Here five clients with tickets compete for resources. 10^{th} ticket is randomly selected for comparison. Tickets sum is calculated until the comparison value is reached. In this case the third client is the winner [8]. Since the number of tickets each client possess is varying , ordering them in the descending order of number of tickets will reduce the average search time to a great extend.

Another fair share algorithm is the Virtual Time Round Robin which combine the benefits of round robin and fair queuing [9] algorithms. Implementation simplicity is the main advantage of this algorithm. The implementation of this algorithm involve following three steps[9].

- Clients are ordered in the run queue according to their share and their position in the queue changes only when their share changes [9].
- 'Starting from the beginning of the queue , run each clients in a round robin manner for a fixed time quantum'[9].
- If any client receive more than its proportional share during the execution of step 2 , further executions will be skipped and the execution start again from the beginning of the queue[9].

2.1 Advantages and Disadvantages of Fair Share Methodology

Fair share scheduler is very effective in allocating resources in a reasonable way. These methodologies ensure fairness in resource allocation. It ensures that all processes will get resources that they want at some reasonable amount of time. Thus it increases response time for each process. Also this scheduler ensures s that all resources of the machine are well utilized. The fair share scheduler also increases the system performance even during peak load time. In a nutshell it always guarantees a predictable performance level. But some methodologies like lottery scheduling some time results in selecting clients with more number of tickets always which will cause some other users to wait more than expected. Implementing mechanisms like' moving forward' [8] can be very effective in resolving this problem.

3. RELATED STUDIES

Due to fact that resource management is a critical function of operating system, many studies have been performed to find an optimal scheduling algorithm which will allocate resources fairly. The paper [1] talks about a surplus fair share scheduling. This scheduler is very effective for allocating resources for a symmetric multiprocessor environment. The paper discusses about a weight re adjustment algorithm and surplus share scheduler to allocate operating system resources fairly [1]. Even though this method is very efficient and practical , it results in a large scale increases in the scheduling overhead .

Generalized process sharing is discussed in papers[2,3,4] . The generalized process sharing is based on a weight assigned to each processes. Resources will be allocated to processes based on its weight. Even though these schedulers are efficient in uniprocessor environment, it may cause indefinite starvation in multi processor systems. The paper [5] introduced the lottery scheduling which provide proper control over the execution rates of computation and also ensures proportional share, modular resource management. In [7] the paper introduces 'a FaRes system level mechanism for monitoring and utilizing that information to ensure fairness among set of Virtual Machines' sharing a set of resources.

The main difference between other resource management techniques and fair share scheduler is that it ensures very high fairness in resource allocation. In the case of priority scheduling a low priority processes will always need to starve. But in the case of fair share scheduler that processes is ensured to get resource at some point of time. Also it ensure that the most needed process will get resource soon and in a timely manner. Similarly in the case of round robin all processes will get a fixed time quantum of resources. So even if a high priority processes is not yet done with its execution, it has to relinquish its resources forcefully. Thus fairness will be lost in

this kind of resource allocation. On the other hand the fair share scheduler ensures that the process's resource requirements are satisfied properly.

4. CONCLUSION

Resource management using the fair share scheduler is very much successful in distributing resources effectively among multiple processes competing for resources. The 'share 'for each user can be determined by various methodologies and implementation overhead for these methodologies are very less compared to poor resource distribution penalties. Future work includes using this scheduler for thread scheduling and memory management. By implementing similar methodologies for thread and memory management I believe the operating system efficiency can be increased to a great extend.

5. REFERENCES

[1] A. Chandra, M.Adler, P. Goyaly and P. Shenoy , "Surplus Fair Scheduling: A Proportional-Share CPU Scheduling Algorithm for Symmetric Multiprocessors " , (paper presented in Proceedings of fifth symposium on operating system design and implementation at Boston, Massachusetts , December 9-11 2002)

[2] P. Goyal, X. Guo, and H.M. Vin." A Hierarchical CPUScheduler for Multimedia Operating Systems. In Proceedingsof Operating System Design and Implementation*"* (paper presented at *OSDI'96, Seattle*, pages 107–122, October 28-31 1996.)

[3] K. Duda and D. Cheriton. "Borrowed Virtual Time (BVT)Scheduling: Supporting Lantency-sensitive Threads in aGeneral-Purpose Scheduler ".(paper presented in Proceedings of the ACMSymposium on Operating Systems Principles (SOSP'99),Kiawah Island Resort, SC, pages 261–276, December 1999 .)

[4] J. Nieh and M S. Lam." The Design, Implementation andEvaluation of SMART: A Scheduler for Multimedia Applications ".(paper presented in *Proceedings of the sixteenth ACM symposiumon Operating systems principles (SOSP'97), Saint-\Malo, France*, pages 184–197, December 1997.)

[5] Carl A. Waldspurger _ William E. Weihl "Lottery Scheduling: Flexible Proportional-Share Resource Management'**",** Comm. ACM 31(1) (Jan 1988.) , pp. 44–55,

[6] A Cray X-MP , Henry**,** G. J. **"**The Fair Share Scheduler**",** Bell System Technical Journal, October,(1984).

[7] Adit Ranadive, Ada Gavrilovska, Karsten Schwan , "FaReS: Fair Resource Scheduling for VMM-Bypass InfiniBand Devices" ,(paper presented in the Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing ,Australia May 2010)

[8] C. A. Waldspurger and W. E. Weihl, "Lotteryscheduling: Flexible proportional-share resource management".(paper presented in 1st Symp. Operating Systems Design & Implementation, pp. 1–11, USENIX, Nov 1994.)

[9] J. Nieh, C. Vaill, and H. Zhong, "Virtual-Time RoundRobin: An O(1) proportional share scheduler"(paper presented in . USENIX Ann. Technical Conf.at Boston, Massachusetts pp. 245–259, Jun25-30 2001.)