Designing of Vague Logic Based Fair-Share CPU Scheduler: VFS CPU Scheduler

Supriya Raheja, ITMU, Gurgaon, India
Reena Dadhich, University of Kota, Kota, India
Smita Rajpal, Alpha Global IT, Toronto, Ontario, Canada

ABSTRACT

CPU scheduler is the primary component of an operating system which supports multitasking. Fair Share CPU scheduler uses static policy to share the CPU time among different levels of system. Static sharing of CPU time can affect the performance of system. Moreover, Fair share scheduler does not consider the impreciseness and uncertainty related to tasks. The objective is to design a vague based fair share scheduler (VFS). VFS scheduler extends the research in the field of fair sharing of CPU time where the functions of VFS scheduler are threefold: firstly, it deals with the impreciseness of task. Secondly, it dynamically shares the CPU time among users as well as among tasks. Thirdly, it supplies the dynamic priority to each task. It improves the performance of system. The VFS scheduler has three modules, VIS-DCS vague inference system for sharing CPU time; VIS-DP vague inference system for assigning dynamic priority to each task and scheduling algorithm to schedule the tasks. The novelty of this approach is to introduce the vague logic based CPU scheduler. Simulation results show that the VFS scheduler has improved performance over the fair share scheduler.

Keywords: Fair Share Scheduler, Scheduler, Vague Based Fair Share Scheduler, Vague Inference System, Vague Set Theory

1. INTRODUCTION

UNIX based operating systems usually uses fair share scheduling algorithm for concurrent execution of multiple tasks. Fair share divides the CPU time among different groups, then to different users and so on (Tanenbaum and Woodfhull, 2008; Dhamdhere,2006). But, this sharing of CPU time is fixed or we can say static. For example if there are two groups with two active users, then according to fixed CPU sharing, 50% of CPU time will be assigned to each group. Within each group 25% of CPU time will be given to each user and so on.

The static sharing affects the performance of system. Moreover, operating system is not aware of exact attributes of tasks like burst time, priority of tasks, finishing time etc (Alam et al., 2011; Raheja et al., 2014) before execution. There can be the probability of impreciseness present in the decisions of CPU Scheduler. Developments in fair share scheduler do not
exactly dealing with this impreciseness and inexactness. Alam et al. (2009) have discussed one fuzzy logic based approach but have not illustrated the improvements.

This paper is written with the main objective of dealing with impreciseness and inexactness. A vague logic based fair share scheduler (VFS) with three modules: two vague inference systems VIS-DCS, VIS-DP and new fair-share scheduling algorithm has been proposed. It deals with the impreciseness and uncertainty; dynamically shares CPU time amongst users as well as tasks; and it also provides dynamic priority to each task of active users. Additionally, it also improves the performance of system over FS scheduler in terms of average response time, average normalized turnaround time and number of context switches.

The paper is organized as follows. Section 2 discusses the related work to the fair-share scheduling. Section 3 outlines basics of vague set theory which itself includes the relationship between two vague member function. Section 4 defines the VFS scheduler in detail. Section 5, describes the simulation results. Finally, Section 6 summarizes the conclusion.

2. STATE OF ART

The main work related to our research address the problem of static sharing of CPU time and the use of fair policies to favouring the users as well as tasks.

A common criticism regarding scheduling techniques discussed so far is the relationship between user and task. Fairness is one of the vital requirements of any scheduler. It is considered that each task belongs to a different user and tried to assign equal CPU services to all tasks (Sabin et al., 1996; Nie et al., 2011). Earlier schedulers were to be designed fairly in context of tasks. However, system usually supports various groups of related tasks. As the time goes, it becomes clear that the scheduler should be fair between users as well as tasks. Fair Share scheduling technique has addressed this problem. For example UNIX and other multiuser systems make groups of tasks that belong to a particular user.

Fair share scheduler permits CPU time to be shared fairly among system groups or users in a system (Stallings, 2014, Bui et al., 2010). It is the fraction of CPU time that should be allocated to group of tasks that belongs to the same user. Let us consider an example where fair share scheduling will be applicable. Suppose a society group whose members are using one multiuser system. They are further divided into two groups. One is the society head and other is the secretaries of this group. The society head uses the system to perform important and intensive work. Whereas many secretaries use the system for less intensive work such as collecting information etc. The secretaries consumes more CPU time as compare to the society head as they are many. But the society head performs the important task over other tasks. However, if the system allows only 30% of the CPU time to secretaries’ work and the remaining 70% for society head, the society head would not suffer. In this manner, fair share scheduling ensures the fairness of CPU share.

Let us suppose three different users (U1, U2, and U3) in a system, where each is concurrently running one task. The scheduler divides the CPU time in such a way that each user can get 33.3% as its fair share. Suppose U3 starts another task, then the tasks running by U1 and U2 still get the same CPU share (33.3%) but the tasks of U3 now receive 16.7% of CPU share. Further, fair share (FS) scheduling allows division of users into groups and assigned the CPU share to groups as well. Scheduler first assigns the share of CPU time to groups then to users within group and later on among tasks of those users (Stallings, 2014).

In a non-preemptive multi-tasking system like UNIX, CPU scheduling technique adjusts the priority of each task in a way that size of time quantum changes accordingly (Moonju, 2008). However, an additional new arrived task will affect all the other ready tasks in a system, as the CPU resource will be redistributed between all tasks (Zaim, 2003; Caprita et al., 2005). Kay and Lauder (1988) introduced a hierarchical
Deontic Logic Based Ontology Alignment Technique for E-Learning
Lazarus Jegatha Deborah, Ramachandran Baskaran and Arputharaj Kannan (2012). 
*International Journal of Intelligent Information Technologies* (pp. 56-72). 
www.igi-global.com/article/deontic-logic-based-ontology-alignment/69390?camid=4v1a