ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

Performance Enhancement of CPU Scheduling by Hybrid Algorithms Using Genetic Approach

Jyotirmay Patel, A.K.Solanki

Abstract - **Scheduling is a fundamental operating-system function. The concept is to have computer resources shared by a number of processes. The productivity of a computer solely depends on the use of CPU scheduling algorithm in a multiprogrammed operating system. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to an operating-system's design and constitutes an important topic in the computing research. The problem of scheduling which computer process run at what time on the central processing unit (CPU) or the processor is explored. Some CPU scheduling algorithms has been elaborated and assessed on the basic CPU scheduling objectives i.e; average waiting time etc. These will form the base parameters in making a decision for the suitability of the given algorithm for a given objective.**

Many algorithms have been developed for the CPU scheduling of a modern multiprogramming operating system. Our research work involves the design and development of new CPU scheduling algorithm (the Hybrid Scheduling Algorithm using genetic approach). This work involves a software tool which produces a comprehensive simulation of a number of CPU scheduling algorithms. The tool's results are in the form of scheduling performance metrics. We will discuss the use of genetic algorithm to provide efficient scheduling algorithm. The work shows that genetic approach will be efficient for sequencing problems. Result of the work shows that proposed genetic algorithm demands less average waiting time.

Index terms: CPU scheduling, Genetic algorithm, Hybrid algorithms, waiting time.

I. INTRODUCTION

When a computer is multiprogrammed, it frequently has multiple processes competing for the CPU at the same time[1]. When more than one process is in the ready state and there is only one CPU available, the operating system must decide which process to run first. The part of operating system that makes the choice is called short term scheduler or CPU scheduler. The algorithm that it uses is called scheduling algorithm.

There are several scheduling algorithms. Different scheduling algorithms have different properties and the choice of a particular algorithm may favor one class of processes over another. Many criteria have been suggested for comparing CPU scheduling algorithms and deciding which one is the best algorithm. Some of the criteria include (i)Fairness(i)CPU utilization(iii)Throughput (iv)Turnaround time (v)Waiting time (vi)Response time[2]. It is desirable to maximize CPU utilization and throughput, to minimize turnaround time, waiting time and response time and to avoid starvation of any process.

II. SCHEDULING ALGORITHMS: AN OVERVIEW

A. First-Come, First-Served (FCFS)

Processes are assigned the CPU in the order they request it.

B. Round-Robin (RR):

Each process is given a limited amount of CPU time, called a time slice, to execute. If the required CPU burst of the process is less than or equal to the time slice, it releases the CPU voluntarily. Otherwise, the scheduler will preempt the running process after one time slice and put it at the back of the ready queue, then dispatch another process from the ready queue.

C. Shortest-Job-First (SJF) Non-preemptive:

When the CPU is available, it is allocated to the process that has the smallest next CPU burst. *SJF Preemptive:* When the CPU is available, it is allocated to the process that has the shortest remaining CPU burst. When a process arrives at the ready queue, it may have a shorter remaining CPU burst than the currently running process. Accordingly, the scheduler will preempt the currently running process.

D. Multilevel Feedback Queues (MLFQ):

There are several ready queues, each with different priority (figure 1). When the CPU is available, the scheduler selects a process from the highest-priority, non-empty ready queue. Within a queue, it uses RR scheduling. The scheduler adjusts the priority of a process dynamically, for example, to reflect resource requirements (e.g., being blocked awaiting an event) and the amount of resources consumed by the process (e.g., CPU time)[3][4]. Processes are moved between ready queues based on changes in their priority. When a process other than the currently running process attains a higher priority, the scheduler will preempt the currently running process and add it to the appropriate ready queue.

E. Priority Scheduling (PS):

The PS algorithm associates with each process a priority and the CPU is allocated to the process based on their priorities. Usually, lower numbers are used to represent higher priorities. The process with the highest priority is allocated first. If there are multiple processes with same priority, typically the FCFS is used to break tie.

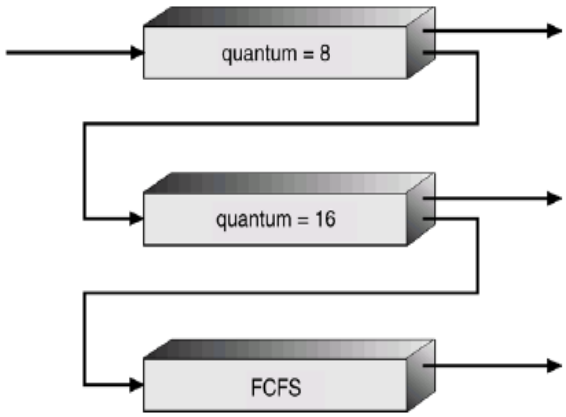

Figure 1: Multilevel Feedback Queues

F. Lottery Scheduling

The Basic idea is to give processes lottery tickets for CPU time. Whenever a scheduling decision has to be made, a lottery ticket is chosen at random and the process holding the ticket gets the CPU[5].

G. Guaranteed scheduling

In this a ratio of actual CPU time a process had and its entitled CPU time is calculated. The process with this lowest ratio is scheduled.

III. PROBLEM STATEMENT

Suppose there are finite k processes (1, 2, 3..k) each will be executed one by one at a time on CPU. Let us consider that the complete execution time of each job on CPU is known. The problem is to find the order so that the total waiting time and total turnaround time is minimal.

A. Assumption

There is a pool of ready processes competing for the processor. The processes are independent and competing for resources. The work of the scheduler is to allocate the available resource of the CPU to the different processes in such a way that it can maximize the some performance criteria.

B. Experimental Environment

Individual solutions are randomly generated to form an initial population. Successive generations of reproduction and crossover produce increasing numbers of individuals in solution regions. The algorithm favors the fittest individuals, as parents will have more offspring in the next generation. To achieve minimum waiting time the fitness function defined here is based on Shortest Job First algorithm[9]. Fitness function checks the jobs, which occurs after the crossover point. If the jobs present after crossover point has minimum CPU burst then fitness function marks it as fit to generate new generation[10][11]. The crossover point considered is (number of jobs)/2, which is rounded towards infinity. Inversion operator is applied to the fit individuals and new generation is formed. The process is repeated till a termination criterion is reached in our case it is minimum waiting time.

IV. DESCRIPTION OF THE EXPERIMENT

Let there are 5 jobs The number of possible permutations are 5!. The number of randomly selected individuals varies depending on the number of jobs and number of machines. Here randomly 24 sequences are selected out of 120 for 5 jobs[6][7]. Considering the number of jobs as 5 the crossover point is 3. Suppose job 2 requires minimum time on CPU. So fitness function checks last two genes of the individuals, if one of the two genes is 2 then that individual can form offspring. Let us consider following two individuals, which are marked as fit to generate next generation.

5 4 3 2 1 and 5 3 1 2 4

After crossover: 2 4 3 5 3

But this is not valid individual as job 1 is not there in new individual and job 3 appears twice. Such individuals are discarded .

V. PROPOSED ALGORITHM

Choose initial random population
do

Calculate the entity fitness using fitness function (based on minimum time)

first-rate pairs of best-ranking entity using fitness function to reproduce

Breed new generation through crossover and inversion

While terminating condition becomes true.

VI. COMPUTATIONAL RESULTS

There are four jobs (1, 2, 3, 4), which requires processing time as (1 14 3 4). The initial population for this example is as shown in table 1

Table 1: Initial Population

2	4	3	1
4	3	2	1
2	3	4	1
3	4	2	1
4	1	2	3
1	3	2	4
3	2	1	4
4	1	3	2
3	2	4	1
4	2	3	1
2	3	1	4
3	1	2	4
1	4	3	2
4	3	2	1
4	1	2	3
3	2	4	1
2	1	4	3

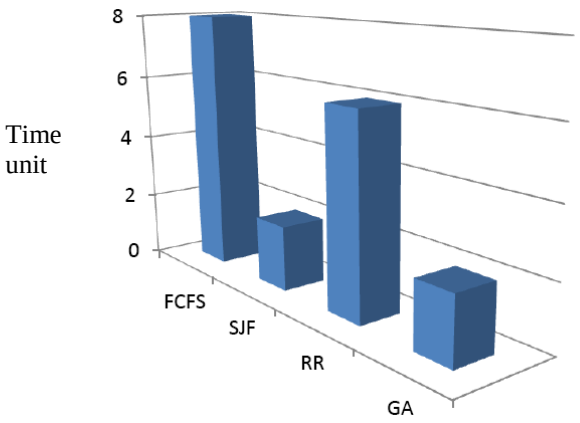

Figure 2: Comparison Graph

After applying genetic operators' crossover, fitness function and inversion we get final population. Out of the above final population we get minimum waiting for job sequence 1, 3, 4, 2 as 3.25 time unit. Following graph focus on comparison based on average waiting time of algorithms. A time unit on Z-axis and on Y-axis FCFS, SJF, RR and GA algorithms has been presented (Figure 2).

VII. CONCLUSION AND FUTURE SCOPE

The problem of scheduling which computer process run at what time on the central processing unit (CPU) or the processor is explored. Some CPU scheduling algorithms has been briefed. The simplicity of the methods used supports the assumption that GA's can provide a highly flexible and user friendly, near optimal solution to the general job sequencing problem. The Genetic algorithms outperform the conventional procedures in solving optimization problems. The new representation has initially been tested on a data to evaluate its effectiveness. Quite promising results are obtained. The simulation results clearly show that the proposed approach is able to find optimized solution. The experiment carried out is efficient to find best sequence. From result we even conclude that with evolutionary technique we may get more than one best sequence with minimum waiting time. This work can be extended so that technique can be implemented for dynamic scheduling and for similar sequencing problem.

REFERENCES

[1] Milenkovic, M., *Operating System Concepts and Design*, McGraw Hill, International Edition, 1992.
[2] Silberschatz,A. and P.B. Galvin,1997. Operating System concepts, Fifth Edition, John Wiley & Sons, Inc.,
[3] Stallings,W., 1996. Computer Organization and Architecture; Designing for Performance, Fourth Edition, Prentice Hall
[4] Andrew S. Tanenbaum , and Albert S. Woodfhull , Operating Systems Design and Implementation,Second Edition,2005.
[5] Yavatkar, R. and K . Lakshman,1995. A CPU Scheduling Algorithm for Continuous M edia Applications, In Proceedings of the 5th International Workshop on Network and Operating System Support.
[6] Dorn, J. and Girsch, M.(1994)Genetic Operators Based on Constraint Repair, ECAI'94 Workshop on Applied Genetic and other Evolutionary algorithms, Amsterdam.
[7] Umale, J.; Mahajan, S.; , "Optimized Grid Scheduling using Two Level Decision Algorithm (TLDA)," *Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on* , vol., no., pp.78-82, 28-30 Oct. 2010
[8] Kashani, M.H.; Sarvizadeh, R.; , "A novel method for task scheduling in distributed systems using Max-Min Ant Colony Optimization," *Advanced Computer Control (ICACC), 2011 3rd International Conference on* , vol., no., pp.422-426, 18-20 Jan. 2011
[9] Kamalapur, S.; Deshpande, N.; , "Efficient CPU Scheduling: A Genetic Algorithm based Approach," *Ad Hoc and Ubiquitous Computing, 2006. ISAUHC '06. International Symposium on* , vol., no., pp.206-207, 20-23 Dec. 2006
[10] Iotfii, H.; Broumandnia, A.; Lotfi, S.; , "Task graph scheduling in multiprocessor systems using a coarse grained genetic algorithm," *Computer Technology and Development (ICCTD), 2010 2nd International Conference on* , vol., no., pp.179-185, 2-4 Nov.2010.
[11] Jahanshahi, M.; Meybodi, M.R.; Dehghan, M.; , "A new approach for task scheduling in distributed systems using learning automata," *Automation and Logistics, 2009. ICAL '09. IEEE International Conference on* , vol., no., pp.62-67, 5-7 Aug. 2009.

Jyotirmay Patel is Research Scholar, in Institute of Computer Application, Bhagwant University, Rajasthan, India. He received MCA degree from M.G.Kashi Vidyapith, Varanasi-India and M.Tech.(IT) from Karnataka state open university-India. His research interest is in Operating Systems. He is life member of professional societies like CSI, ISTE, IETE.

A.K.Solanki is working as professor, in the Department of Computer Sc. & Engg, Bundelkhand University, He received a Ph.D. in Computer and Engineering in 2005, M.E. Degree in Computer Science & Engineering in 1996, and MBA degree in 2008. His research interests in Data mining and Web mining