Global Journals LaTeX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Optimized Round Robin CPU Scheduling Algorithm

Dr. Sukumar Babu[1], Neelima Priyanka[2] and Dr. P. Suresh Varma[3]

[1] Vijaya Institute Of Technology For Women, JNTUK

Received: 14 December 2011 Accepted: 5 January 2012 Published: 15 January 2012

Abstract

One of the fundamental function of an operating system is scheduling. There are 2 types of uni-processor operating system in general. Those are uni-programming and multi-programming. Uni-programming operating system execute only single job at a time while multiprogramming operating system is capable of executing multiple jobs concurrently. Resource utilization is the basic aim of multiprogramming operating system. There are many scheduling algorithms available for multi-programming operating system. But our work focuses on design and development aspect of new and novel scheduling algorithm for multi-programming operating system in the view of optimization. We developed a tool which gives output in the form of experimental results with respect to some standard and new scheduling algorithms e.g. First come first serve, shortest job first, round robin, optimal and a novel cpu scheduling algorithm etc.

Index terms— operating system, uni-processor, uni programming, multi-programming, resource utilization, scheduling, fcfs, sjf, priority, round robin, a novel CPU s

1 Introduction

cheduling is the heart of any computer system since it contain decision of giving resources between possible processes. Sharing of computer resources between multiple processes is also called as scheduling. The CPU is one of the primary computer resources, so its scheduling is essential to an operating system's design **??**1]. Efficient resource utilization is achieved by sharing system resources amongst multiple users and system processes **??**2]. Optimum resource sharing depends on the efficient scheduling of competing users and system processes for the processor, which renders process scheduling an important aspect of a multiprogramming operating system. As the processor is the most important resource, process scheduling, which is called CPU scheduling, becomes all the more important in achieving the above mentioned objectives **??**1]. Part of the reason for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU. Another main reason is the need for processes to perform I/O operations in the normal course of computation. Since I/O operations ordinarily require orders of magnitude more time to complete than do CPU instructions, multiprograming systems allocate the CPU to another process whenever a process invokes an I/O operation[3]. a) Goals for Scheduling FCFS is the simplest scheduling algorithm. For this algorithm the ready queue is maintained as a FIFO queue. PCB (Process Control Block) of a process submitted to the system is linked to the tail of the queue. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.

ii. Shortest Job First Scheduling Algorithm (SJF) For this algorithm the ready queue is maintained in order of CPU burst length, with the shortest burst length at the head of the queue. A PCB of a process submitted to the system is linked to the queue in accordance with its CPU burst length. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.

In this algorithm, priority is associated with each process and on the basis of that priority CPU is allocated to the processes. Higher priority processes are executed first and lower priority processes are executed at the

end. If multiple processes having the same priorities are ready to execute, control of CPU is assigned to these processes on the basis of FCFS **??**1]. Priority Scheduling can be preemptive and nonpreemptive in nature. iv. Round Robin Scheduling Algorithm (RR) For this algorithm the ready queue is maintained as a FIFO queue. A PCB of a process submitted to the system is linked to the tail of the queue. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. Processes being executed are preempted on expiry of a time quantum, which is a system defined variable. A preempted process's PCB is linked to the tail of the ready queue. When a process has completed its task, i.e. before the expiry of the time quantum, it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue. The proposed algorithm A Novel CPU Scheduling algorithm is both preemptive and nonpreemptive in nature. In this algorithm a new factor called condition factor(F) is calculated by the addition of burst time and arrival time ie., F = Burst Time + Arrival Time.

This factor f is assigned to each process and on the basis of this factor process are arranged in ascending order in the ready queue. Process having shortest condition factor (F) are executed first and process with next shortest factor (F) value is executed next. By considering the arrival time the new algorithms acts as preemptive or non-preemptive.

Proposed CPU scheduling algorithm reduces waiting time, turnaround time and response time and also increases CPU utilization and throughput.

The working procedure of A Novel Preemptive Scheduling of Preemptive and Non Preemptive algorithm is as given below:

? Take the list of processes, their burst time and arrival time. ? Find the condition factor F by adding arrival time and burst time of processes. ? First arrange the processes, burst time, condition factor based on arrival time ascending order. ? Iterate step a, b until burst time becomes zero.

a. If arrival time of first and second process are equal the arrange them based on their condition factor f. b. Decrement the burst time and increment arrival time by 1 ? When burst time becomes find the waiting time and turnaround time of that process. ? Average waiting time is calculated by dividing total waiting time with total number of processes. ? Average turnaround time is calculated by dividing total turnaround time by total number of processes.

[6].

2 b) Work Procedure of Efficient CPU Scheduling Algorithm

The proposed algorithm ERR is pre-emptive in nature. In this algorithm all the processes are sorted in ascending order in the ready queue. ERR scheduling algorithm maintains a small unit of time called time quantum or time slice is assigned to each process. According to that time quantum processes are executed and if time quantum of any process expires before its complete execution, it is put at the end of the ready queue and control of the CPU is assigned to the next shortest incoming process. In this algorithm ready queue is a circular queue. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue sets of timer to interrupt after assigned time quantum, and dispatches the process.

The average waiting time and average turnaround time obtained from ERR is better than existing CPU scheduling algorithms.

ERR is fair in scheduling and effective in time sharing environment. In ERR scheduling, CPU is given to each process for equal time period, no process has to wait for long time for the CPU.

Working Procedure of the Proposed Algorithm ERR is discussed below: ? First collect all the list of processes, their burst time, arrival time and time quantum. ? Arrange processes and their burst time in ascending order based on their arrival time. ? Repeat 1 and 2 until all process complete their execution 1. If the current time is equal to arrival time of the ready queue process And if burst time of current process is greater than time slice then execute for time slice period else execute for burst time period. III.

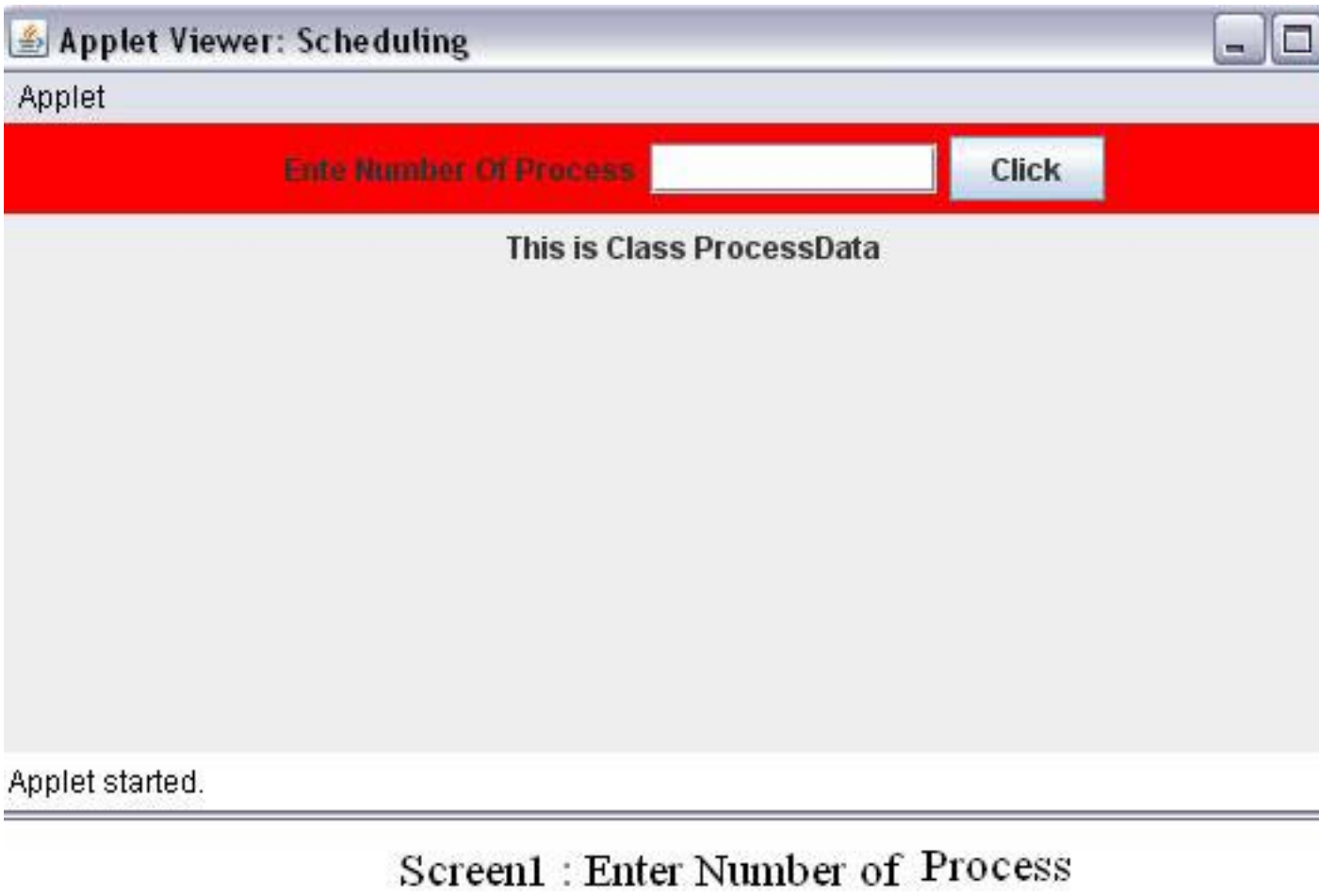
3 Global

4 Performance Comparison between Existing and New Algorithms

The simulation provides an interactive GUI interface to the user through which a user can input data related to different processes then applying different algorithms based on the algorithm choices given in the simulator. The simulator employs JAVA swings. The front end user interfaces are developed using java awt's and swings. The parent screen allows the user to give number of processes to be in execution.

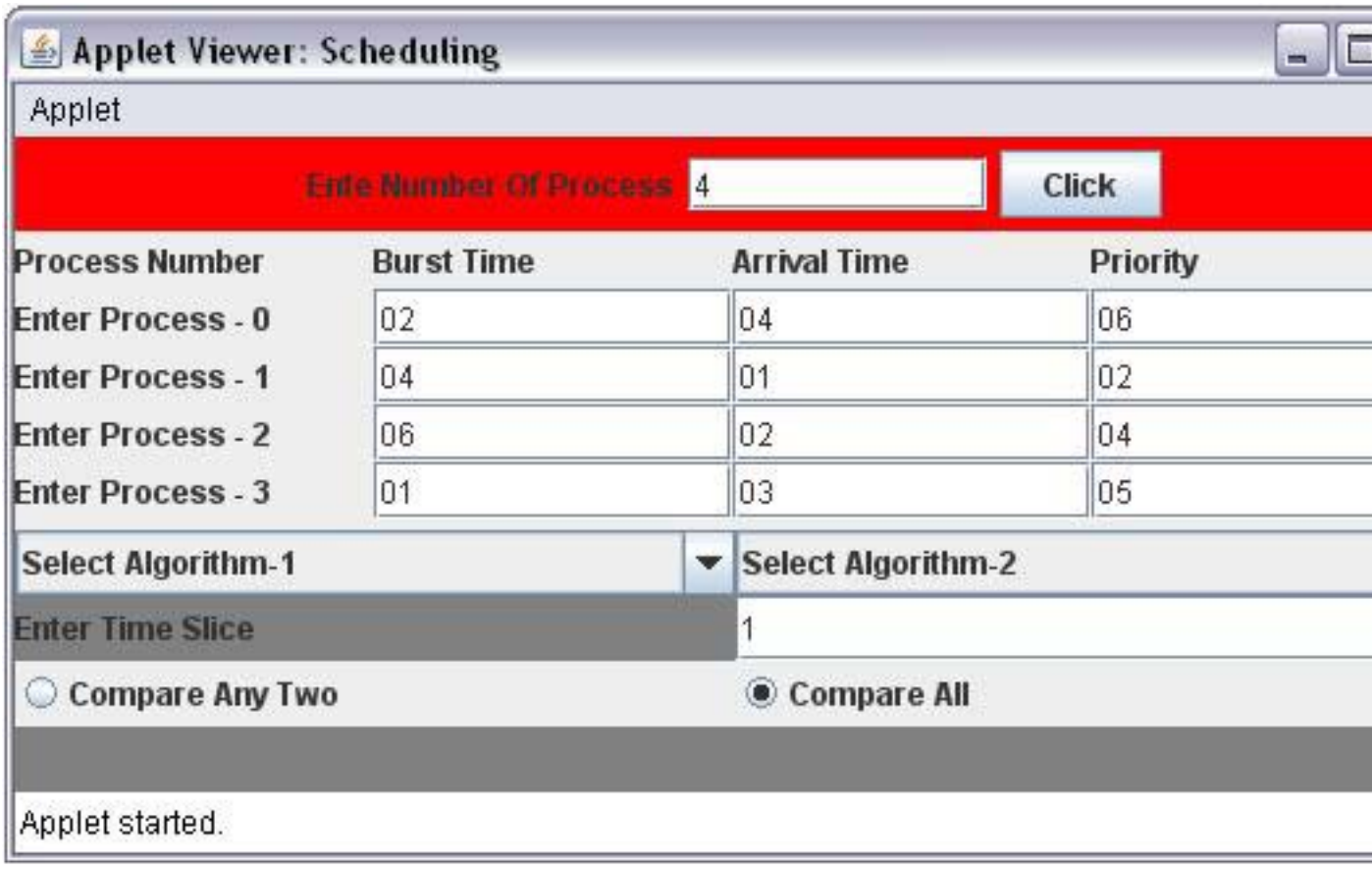

OPEN
ASSOCIATION
OF RESEARCH
SOCIETY, USA
RESEARCH | DIVERSITY | ETHICS

Figure 1:



Screen1 : Enter Number of Process

Figure 2:



Screen 2: Enter Burst , Arrival and Priority of processes

Figure 3:

4 PERFORMANCE COMPARISON BETWEEN EXISTING AND NEW ALGORITHMS