Round Robin CPU Scheduling

Brayden Sorenson
CSCI 330 - M02
Professor Gass
Dec. 5, 2016

Overview

This project simulates the Round Robin CPU scheduling algorithm that may be used by a computer’s operating system. The purpose of this project is to analyze how different time quantums affect the overall performance of an operating system that uses Round Robin scheduling.

CPU Scheduling

When using a multitasking operating system, many processes must compete for execution by the computer’s CPU. A scheduling algorithm is used to determine the order in which processes are executed, and for how long. The scheduling algorithm impacts the overall performance of the system.

The goals of CPU scheduling are to maximize CPU utilization, maximize throughput, minimize turnaround time, minimize waiting time, and minimize response time.

Utilization is the percentage of time that the CPU is processing a task compared to the amount of time that it is idle. *Throughput* is the number of processes that are completed for a certain unit of time. *Turnaround time* is the total amount of time it takes for a process to finish executing. *Waiting time* is the amount of time the process spends in the ready queue, which means the process is outside of the CPU waiting for execution. *Response time* is the amount of time it takes until the CPU begins to execute a process after that process has requested execution.

How Round Robin Works

Round Robin is a scheduling algorithm that divides CPU execution time equally among all processes that are ready to be executed. The algorithm uses a specified time quantum, which indicates the amount of time each process gets to spend executing on the CPU. Each process executes in the order that it arrives in the ready queue. If one CPU cycle is not enough, it is inserted at the end of the queue and waits until it can be executed again. This cycle continues for all processes until they are complete.

Implementation

For this program, a comma separated value (CSV) text file is used to represent different processes that are ready to run. Each process has a process ID, arrival time, and burst time. The burst time is an estimate of the amount of time a process will take to finish execution. The program reads these processes from the file and stores them in a list. The list is then sorted by arrival time. If two processes arrive at the same time, then the order they are executed in is decided arbitrarily. Then, the processes are selected from the list and added to the ready queue, preserving the order of arrival time.

Next, each process is sent to the CPU, which decreases the burst time of the process by the specified time quantum. The CPU class simulates the execution time by putting the current program thread to sleep for the duration equal to the time quantum. This is done because there isn't any executable code associated with each process. If the process's remaining burst time is less than the time quantum, then the thread is put to sleep for appropriate length of ratio of remaining burst time to time quantum.

How to Run the Program

In order to run the program, a text file must be created that describes the processes to be run. A default processes.txt file is included in the zipped folder. This file is an ordinary comma separated values (CSV) file with the format of

pid,arrivaltime,bursttime

The program can be run on a machine with a Java Development Kit installed. First, open a terminal and navigate to where the program is installed on the computer. After changing the current directory to the *RoundRobin* folder, use the following commands:

```
javac CPUscheduling\*.java
java CPUscheduling.Simulator [file path] [time quantum]
```

The first line compiles all of the .java files into runnable .class files, and only has to be done once. The second line runs the program. Replace [file path] with the path to the text file that contains the processes. Make sure the name of the file is included at the end. For example, Replace [time quantum] with the amount of time each process can be run on the CPU. This time is in milliseconds.

Analyzing Output

Example output

```
Process created          pid=1     time=2016/12/07 13:41:07:854
Process created          pid=2     time=2016/12/07 13:41:07:866
Process created          pid=3     time=2016/12/07 13:41:07:867
Process created          pid=4     time=2016/12/07 13:41:07:869
Processes loaded and created.
Allocating the ready queue...
Ready queue allocated. Sending for dispatch
Context switch complete            time=2016/12/07 13:41:07:877
Process executing        pid=1     time=2016/12/07 13:41:07:878
Context switch started             time=2016/12/07 13:41:07:882
Context switch complete            time=2016/12/07 13:41:07:883
Process executing        pid=3     time=2016/12/07 13:41:07:884
Process completed        pid=3     time=2016/12/07 13:41:07:888
```

```
Context switch started                          time=2016/12/07 13:41:07:889
Context switch complete                         time=2016/12/07 13:41:07:890
Process executing            pid=2              time=2016/12/07 13:41:07:891
Context switch started                          time=2016/12/07 13:41:07:896
Context switch complete                         time=2016/12/07 13:41:07:897
Process executing            pid=4              time=2016/12/07 13:41:07:898
Context switch started                          time=2016/12/07 13:41:07:902
Context switch complete                         time=2016/12/07 13:41:07:903
Process executing            pid=1              time=2016/12/07 13:41:07:904
Context switch started                          time=2016/12/07 13:41:07:907
Context switch complete                         time=2016/12/07 13:41:07:908
Process executing            pid=2              time=2016/12/07 13:41:07:909
Context switch started                          time=2016/12/07 13:41:07:912
Context switch complete                         time=2016/12/07 13:41:07:913
Process executing            pid=4              time=2016/12/07 13:41:07:914
Context switch started                          time=2016/12/07 13:41:07:917
Context switch complete                         time=2016/12/07 13:41:07:918
Process executing            pid=1              time=2016/12/07 13:41:07:919
Process completed            pid=1              time=2016/12/07 13:41:07:922
Context switch started                          time=2016/12/07 13:41:07:924
Context switch complete                         time=2016/12/07 13:41:07:925
Process executing            pid=2              time=2016/12/07 13:41:07:926
Context switch started                          time=2016/12/07 13:41:07:929
Context switch complete                         time=2016/12/07 13:41:07:930
Process executing            pid=4              time=2016/12/07 13:41:07:931
Process completed            pid=4              time=2016/12/07 13:41:07:934
Context switch started                          time=2016/12/07 13:41:07:935
Context switch complete                         time=2016/12/07 13:41:07:937
Process executing            pid=2              time=2016/12/07 13:41:07:938
Process completed            pid=2              time=2016/12/07 13:41:07:941
Context switch started                          time=2016/12/07 13:41:07:943
All processes have been fully executed.

Total context switches= 11
pid=3          wait time=6    ms      turnaround time=8    ms
pid=1          wait time=48   ms      turnaround time=53   ms
pid=2          wait time=54   ms      turnaround time=61   ms
pid=4          wait time=39   ms      turnaround time=45   ms

CPU utilization=              100%
Avg. wait time=               36.750    ms
Avg. turnaround time=         41.750    ms
Avg. throughput=              10.438    ms/process
```

The simulator prints logs of certain states while the program is running. The first rows show the time that each process is created in the system. Next, the output will notify the user that the processes are being added to the ready queue and sent to the short term scheduler. After this, logs will be generated when each process enters execution in the CPU, when a context switch begins, and when a context switch is completed.

Once all processes have finished execution, a summary of all events is generated. This includes the total number of context switches, total wait time for each process, and turnaround time for each process. The CPU utilization will always be at 100% for the Round Robin scheduling algorithm.

As the time quantum is changed when running the same set of processes, the average wait time, turnaround time, and throughput will also change. Here are the outputs of five different time quantums for the same processes.txt file:

TQ	Output
1 ms	Total context switches= 20 pid=3 wait time=23 ms turnaround time=25 ms pid=1 wait time=79 ms turnaround time=84 ms pid=2 wait time=96 ms turnaround time=103 ms pid=4 wait time=84 ms turnaround time=90 ms CPU utilization= 100% Avg. wait time= 70.500 ms Avg. turnaround time= 75.500 ms Avg. throughput= 18.875 ms/process
2 ms	Total context switches= 11 pid=3 wait time=7 ms turnaround time=9 ms pid=1 wait time=56 ms turnaround time=61 ms pid=2 wait time=66 ms turnaround time=73 ms pid=4 wait time=53 ms turnaround time=59 ms CPU utilization= 100% Avg. wait time= 45.500 ms Avg. turnaround time= 50.500 ms Avg. throughput= 12.625 ms/process

4 ms	TQ = 4 ms
	Total context switches= 7
	pid=3 wait time=5 ms turnaround time=7 ms
	pid=1 wait time=45 ms turnaround time=50 ms
	pid=2 wait time=40 ms turnaround time=47 ms
	pid=4 wait time=40 ms turnaround time=46 ms
	
	CPU utilization= 100%

	Avg. wait time= 32.500 ms Avg. turnaround time= 37.500 ms Avg. throughput= 9.375 ms/process
6 ms	Total context switches= 5 pid=1 wait time=9 ms turnaround time=14 ms pid=3 wait time=5 ms turnaround time=7 ms pid=2 wait time=37 ms turnaround time=44 ms pid=4 wait time=13 ms turnaround time=19 ms CPU utilization= 100% Avg. wait time= 16.000 ms Avg. turnaround time= 21.000 ms Avg. throughput= 5.250 ms/process
8 ms	Total context switches= 4 pid=1 wait time=5 ms turnaround time=10 ms pid=3 wait time=3 ms turnaround time=5 ms pid=2 wait time=4 ms turnaround time=11 ms pid=4 wait time=4 ms turnaround time=10 ms CPU utilization= 100% Avg. wait time= 4.000 ms Avg. turnaround time= 9.000 ms Avg. throughput= 2.250 ms/process

The effect the time quantum has over the average throughput can be seen by analyzing these outputs. As the time quantum increases, the throughput also increases. (Note that the throughput increases as the number of ms/process decreases.) However, as the time quantum gets larger, the ready queue would get longer in a real situation where more processes may be continually added. This means that there will be a longer response time for each process as more processes are added to the ready queue. In turn, the interaction with applications may be delayed.

As the time quantum is decreased, the number of context switches increases. Every context switch takes time, resulting in longer times for processes to be completed. As a result, the average wait time becomes longer and the throughput is decreased. However, response time is increased because time is more equally divided among all processes.

In order to find an “ideal” time quantum, the system goals should be considered, along with the side effects that result from longer and shorter time quantums.