Luke Harrison
CSCI 330 M04

CPU Scheduling and Round Robin

CPU scheduling is the operating systems attempt at optimizing the usage of the computers CPU by processes. It also attempts to increase throughput and decrease metrics such as average wait time, turnaround time, and response time. Without a scheduling algorithm the cpu would not know which processes it should execute to produce the best results. One of the most popular CPU scheduling algorithms is known as the round robin scheduling algorithm which attempts to give processes the best average response time and avoid a phenomena known as starvation. Starvation occurs when processes do not get to run on the cpu because of their low priority or because there are too many processes with very large burst times ahead of them. The way the round robin scheduling algorithm attempts to remedy this is by giving every process an equal amount of time to run on the cpu before being replaced by the next process. This allotted time to each process is known as a time quantum. By adjusting this time quantum one could produce a number of different results which could be both beneficial and detrimental. The technique to choosing this time quantum is to choose one which is not too large or too small. These benefits and drawbacks associated with choosing large and small time quantums will be discussed in more detail throughout this paper.

Round Robin Implementation

Program overview:

1. At the programs execution a single scheduler object is created using the parameters provided by the user (process file location, time quantum, context switch time).
2. When this scheduler object is created its fields are initialized and the process csv file is parsed.
3. For each row in the csv file a process object is created and placed into the process list (a list containing all processes).
4. The process creator then loops through this list and places the processes which have an arrival time of zero into the ready queue (this also occurs every time the cpu's clock is increased by one time unit).
5. Finally each process is executed for as long as the specified time quantum and context switched with the next process when it has finished its turn or has run for its required burst time.

The round robin CPU scheduling simulator is implemented in java and contains the following classes (for a full list of methods see source code):

- Driver.java - The class containing the main method used for running the simulation
- Process.java - The class that defines the fields and methods for process objects
    - Important Fields:
        - id - The process's ID.
        - currentBurstTime - The process's current burst time.
        - arrivalTime - The time when the process arrives at the ready queue.
        - completionTime - The time when the process finishes its execution.
        - waitTime - The current number of time units this process has spent waiting.
        - contextSwitches - The total number of context switches performed for this process.
    - Important Methods:
        - run(int currentTime) - decreases a process's burst time by one and calls terminate method if the process's current burst time is reduced to zero.
        - terminate(int currentTime) - sets the processes completion time to the currentTime and completed to true.
        - waiting() - Increases the process's wait time by one.
        - contextSwitch() - Increases the process's contextSwitches by one.
- Scheduler.java - The class that defines the actual round robin scheduling algorithm
    - Data Structures:
        - readyQueue (Linked List Queue) - The FIFO ready queue used to determine which process should be executed on the CPU next.
        - processList (Array List) - The list of all processes that need to placed into the ready queue when they're arrival time is reached.
    - Important Fields
        - clock - The CPU clock that keeps track of how many time units have passed.
        - idleClock - Represents the number of time units the CPU has been idle for.
        - timeQuantum - The specified time quantum for the round robin scheduling algorithm.
        - contextSwitchTime - The specified amount of time it takes for a single context switch
    - Important Methods:
        - runNextProcess() - Acts as the cpu running a single process for the specified time quantum or until the process is finished executing

- createProcesses() - Loops through the process list and checks and places processes with arrival times that match the current clock into the ready queue.
- printAllMetrics() - Calls all other methods which calculate and print the scheduling algorithm and individual process' metrics.
- contextSwitch(Process p) - performs a context switch on a single process p by increasing the idle time of all processes and the clock.
- isComplete() - checks if the scheduling algorithm has terminated all of its processes.

In order to use this code yourself you only need to do four simple steps:

1. Create an object of the Scheduler class
    a. Scheduler s = new Scheduler(filename.csv, time quantum, context switch time)
2. Call the runNextProcess method on this object
    a. s.runNextProcess()
3. Repeat this until the scheduler has completed execution of all of its processes
    a. while(!s.isComplete())
4. Print out metrics for current state or end state of scheduler
    a. Current state: s.printSchedulerState()
    b. End state: s.printAllMetrics()

Running simulation from command line:

1. Compile: javac Driver.class
2. Command line inputs (in this order):
    a. csv file
    b. time quantum >= 1
    c. context switch time >= 0
    d. show intermediate steps true/false
3. Execute: java Driver test.txt 4 3 false
4. Execute (with intermediate steps): java Driver test.txt 4 3 true

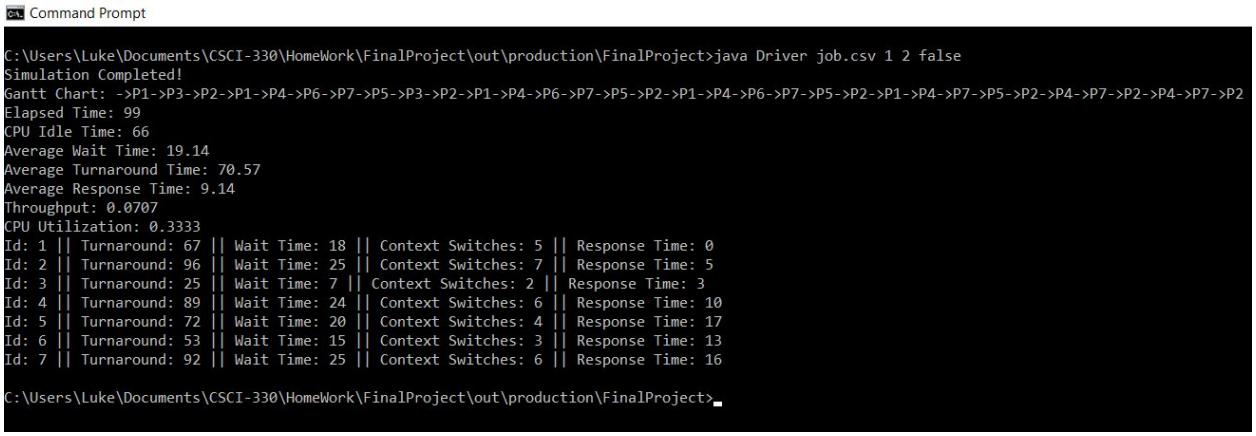
Analysis

Process Table:

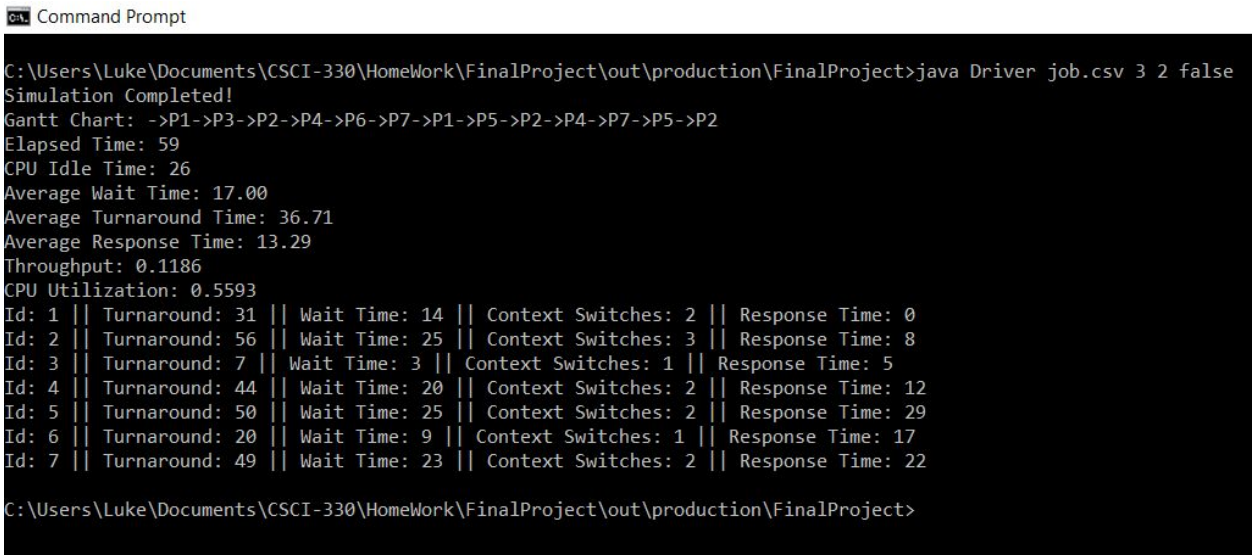
Process Id	Arrival Time	Burst Time
1	0	5

2	1	7
3	0	2
4	2	6
5	4	4
6	2	3
7	2	6

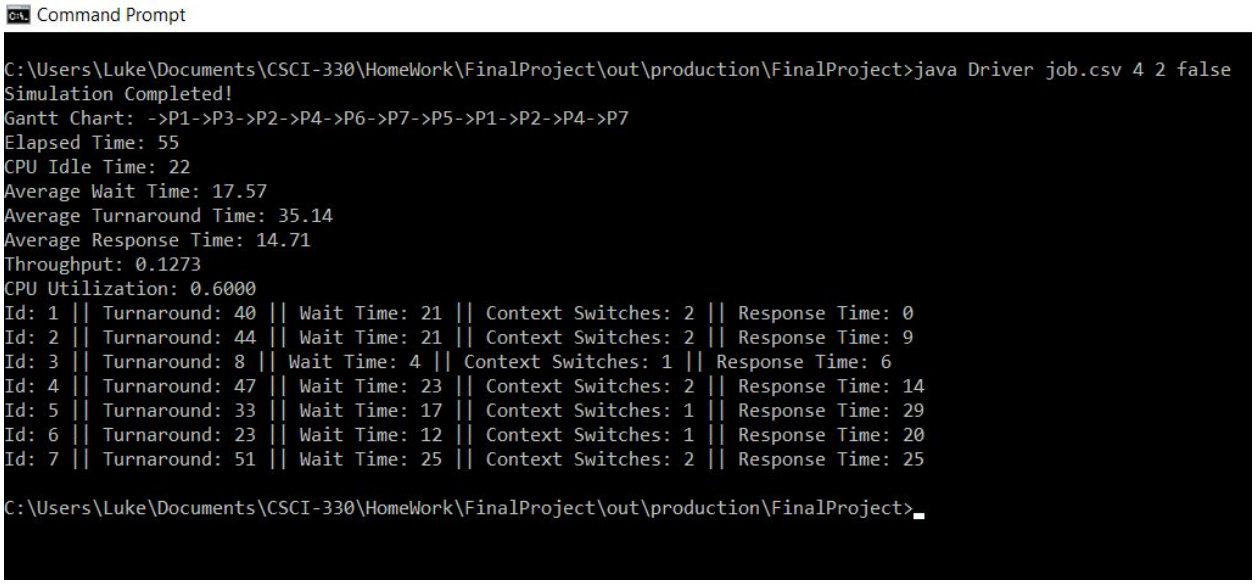
Time quantum 1:



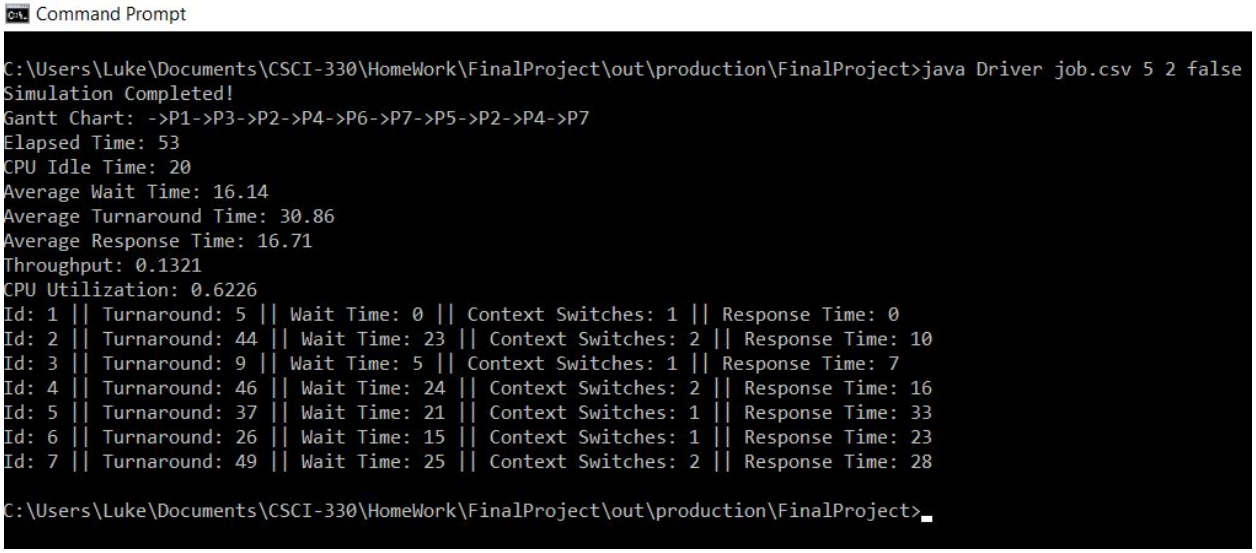
Time quantum 3:



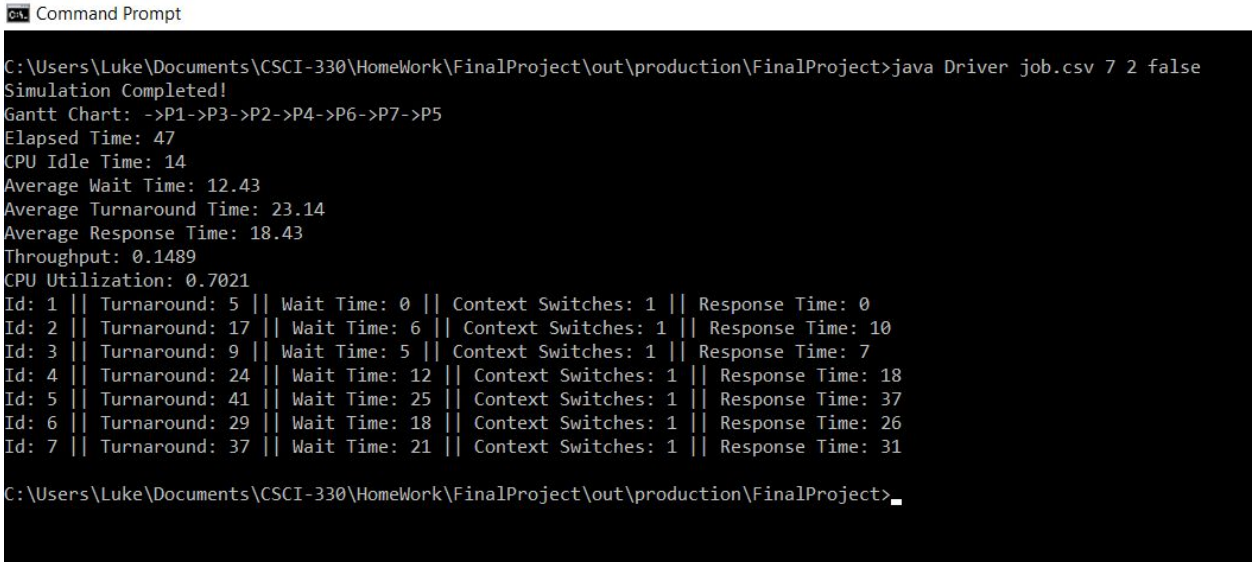
Time quantum 4:



Time quantum 5:



Time quantum 7:



Conclusion:

A time quantum of 1 gives the best average response time for this list of processes, but it also results in too many context switches. This becomes a problem because it increases the average times for all other metrics like wait time, turnaround time and CPU idle time. It also causes a reduction in throughput and CPU utilization. A time quantum of 7 gives the best results for all metrics besides response time. At first glance this may seem like the best time quantum to use based on the results, but this causes another issue known as the convoy effect. Since the response time is so low for this time quantum, this will cause processes that arrive later to have to wait longer to execute. Furthermore when a time quantum is equal to the longest burst time the round robin algorithm become a first come first serve algorithm. The time quantums that give the best average results are 3, 4 and 5. A time quantum of 3 gives the best average response time of the three but sacrifices on all other metrics. A time quantum of 5 gives the worst average response time of the 3 but has the better CPU utilization, average wait time, and throughput. Lastly a response time of 4 gives the best all around average of all metrics. In summary choosing the best time quantum would depend on the situation and task at hand, but for this specific instance the best performing time quantum is 4 because it provides the most balanced results of the five quantums.