Priyank Kashyap

Round Robin CPU Scheduling

Student Id: 0892582
CSCI-330-M02

Why CPU Scheduling?

CPU scheduling is necessary as all processes require time on the CPU to complete execution and we don't want only one process to hold on to CPU resources and run on it infinitely. We want to make sure all processes run the same amount of time on the CPU and allow us maximize CPU utilization. To achieve that we have various CPU Scheduling algorithms such as

i) First Come First Serve
ii) Shortest Time Left
iii) Round Robin
iv) Priority Algorithm

For the purpose of this project I used Java to simulate a CPU with Round Robin scheduling to allow processes with a given Process ID, Arrival Time and Burst cycle, time on the CPU to run

Goals for Scheduling:

1) CPU Utilization to be maximum :- We do not want our CPU to remain idle at any instance of the simulation
2) Maximize the number of jobs processed
3) Minimize turnaround time
4) Minimize waiting time for processes once they arrive in the ready queue
5) Minimize context switching

Implementation

I have simulated the round robin algorithm using Java. There are 3 classes, Scheduler, Process and reader. Each process that is read in from a CSV file which has a specified format. Once the file is read, the Process objects are created with an id, burst time and arrival time, all which are obtained from the CSV file. The processes are then sorted on the basis of arrival times with the ones first to arrive stored first in a temporary ArrayList after which they are added to a queue to be run on the CPU. The CPU in the program is a variable of the type process. It can have one process at a time. The timer or the clock of the CPU is an integer variable that keeps incrementing as the CPU starts executing the processes and goes on until the CPU finishes executing all of the processes in the ready queue. To make the calculations easier there is another completed queue where the processes are added when they finish execution.

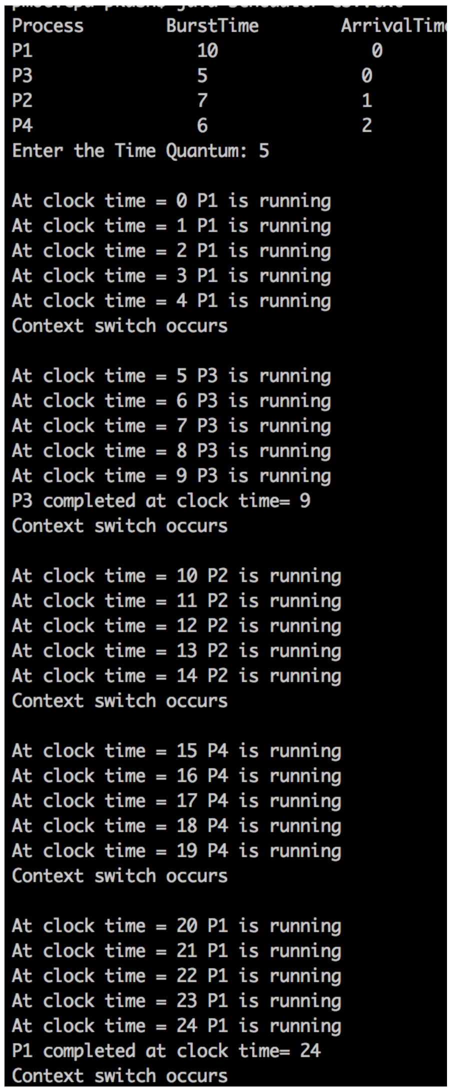
Empirical Evaluation

For the simulation of the round robin algorithm I have created four different processes to be used with different time quantum. Below are the set of processes that I have used for my calculation.

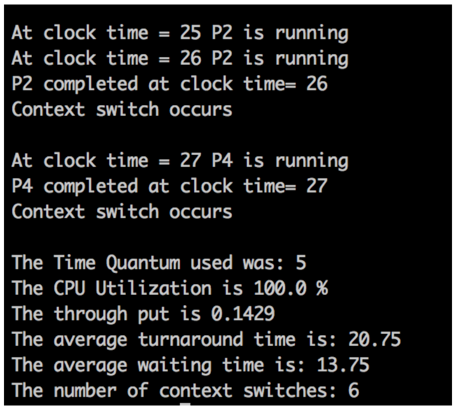
- CPU Utilization – Calculated CPU utilization by counting the number of unit time CPU was being used divided by the total time.
- Throughput – Calculated throughput by dividing the number of processes by the total time taken to complete execution of all the processes.

- Average turnaround time – Calculated the average turnaround time by dividing the total turnaround time by the number of processes.
- Waiting Time – Calculated the waiting time by subtracting the burst time from the turnaround time and took the average of all the waiting times.

1) Time Quantum= 5

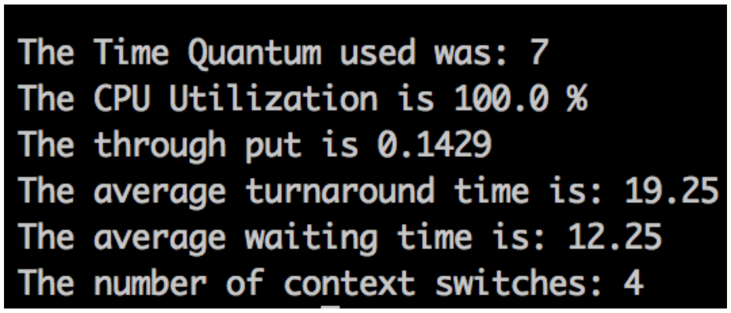


```
Process        BurstTime        ArrivalTim
P1                10               0
P3                5               0
P2                7               1
P4                6               2
Enter the Time Quantum: 5

At clock time = 0 P1 is running
At clock time = 1 P1 is running
At clock time = 2 P1 is running
At clock time = 3 P1 is running
At clock time = 4 P1 is running
Context switch occurs

At clock time = 5 P3 is running
At clock time = 6 P3 is running
At clock time = 7 P3 is running
At clock time = 8 P3 is running
At clock time = 9 P3 is running
P3 completed at clock time= 9
Context switch occurs

At clock time = 10 P2 is running
At clock time = 11 P2 is running
At clock time = 12 P2 is running
At clock time = 13 P2 is running
At clock time = 14 P2 is running
Context switch occurs

At clock time = 15 P4 is running
At clock time = 16 P4 is running
At clock time = 17 P4 is running
At clock time = 18 P4 is running
At clock time = 19 P4 is running
Context switch occurs

At clock time = 20 P1 is running
At clock time = 21 P1 is running
At clock time = 22 P1 is running
At clock time = 23 P1 is running
At clock time = 24 P1 is running
P1 completed at clock time= 24
Context switch occurs
```



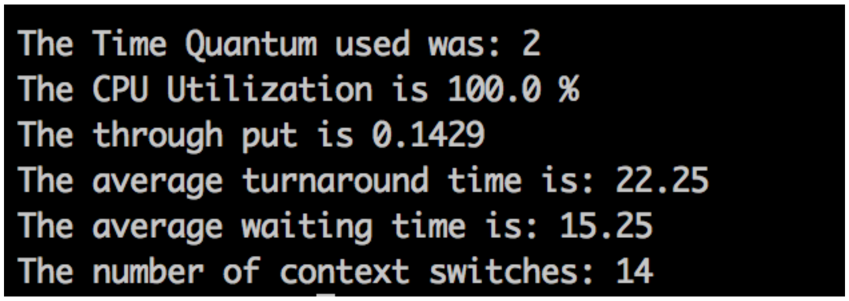
```
At clock time = 25 P2 is running
At clock time = 26 P2 is running
P2 completed at clock time= 26
Context switch occurs

At clock time = 27 P4 is running
P4 completed at clock time= 27
Context switch occurs

The Time Quantum used was: 5
The CPU Utilization is 100.0 %
The through put is 0.1429
The average turnaround time is: 20.75
The average waiting time is: 13.75
The number of context switches: 6
```

2) Time Quantum= 7



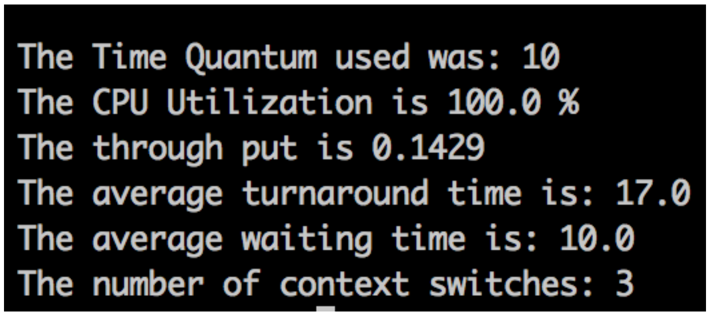
```
The Time Quantum used was: 7
The CPU Utilization is 100.0 %
The through put is 0.1429
The average turnaround time is: 19.25
The average waiting time is: 12.25
The number of context switches: 4
```

3) Time Quantum= 2



```
The Time Quantum used was: 2
The CPU Utilization is 100.0 %
The through put is 0.1429
The average turnaround time is: 22.25
The average waiting time is: 15.25
The number of context switches: 14
```

4) Time Quantum= 10



```
The Time Quantum used was: 10
The CPU Utilization is 100.0 %
The through put is 0.1429
The average turnaround time is: 17.0
The average waiting time is: 10.0
The number of context switches: 3
```

<u>Source Code</u>

```
import java.util.concurrent.ArrayBlockingQueue;
import java.util.Scanner;
import java.util.ArrayList;
import java.io.FileNotFoundException;
import java.io.IOException;
public class Scheduler {
    public static void main(String [] args) throws FileNotFoundException{
        ArrayList <Process> please=null;
        String address=args[0];//Reading in the file or file directory
        ///////////////////////////////////////////////////////////////////

        try{ //Reading the CSV file and storing it in the temporary ArrayList

            reader trial=new reader(address);

            trial.open();

            please=trial.processList();
        }
        catch(IOException e){
            System.out.println("No file");
        }
        ///////////////////////////////////////////////////////////////////////
        System.out.println("Process        BurstTime        ArrivalTime");//Displaying All the
processes in the queue
        for(int j=0;j<please.size();j++){
            System.out.println(please.get(j).getPID()+"                "+ please.get(j).getBurst()+"
"+please.get(j).getArrivalTime());
        }
        ArrayBlockingQueue readyQ=new ArrayBlockingQueue(please.size());
        for(int i=0;i<please.size();i++){
            readyQ.add(please.get(i));
        }
        ///////////////////////////////////////////////////////////////////////
        Scanner in=new Scanner(System.in);
        int clock=0;
        Process current_cpu;
        System.out.print("Enter the Time Quantum: ");
        int n=in.nextInt();//Reading in the desired time quantum
        int timeQuantum= n;
        int contextSwitch=-1;
        int cpu_idle_counter=0;
        int numProcess=readyQ.size();
        ArrayBlockingQueue finishedQ=new ArrayBlockingQueue(numProcess);
        ////////////////////////////////////////////////////////////////////////
        while(!readyQ.isEmpty()){
            current_cpu=(Process) readyQ.poll();
            System.out.println(" ");
            for (int cpuStartTime=clock;(clock-cpuStartTime)<timeQuantum;clock++){ //keeps
running for the time quantum times
                if(current_cpu==null){ // Checking for number of CPU Idle times
                    cpu_idle_counter++;
                }
                System.out.println("At clock time = "+clock+" "+ current_cpu.getPID() +" is
running");
                if(current_cpu.getBurst()==current_cpu.getServiceTime()){ //Process starts running for
the first time
                }

                current_cpu.reduceServiceTime();//Decrease the service Time

                if(current_cpu.getServiceTime()==0){   //Checking if the process finished execution
                    current_cpu.addCompletionTime(clock);
                    finishedQ.add(current_cpu);   //Process is sent to completedQ
                    System.out.println(current_cpu.getPID()+" completed at clock time=
"+current_cpu.getCompletionTime());
                    contextSwitch++; //Increments the context switch value
                    System.out.println("Context switch occurs");
                    clock++;
                    break;
                }
            }
            if(current_cpu.getServiceTime()>0){//Incomplete Processes gets added to the back to the
end of the readyQ again
                readyQ.add(current_cpu);
                contextSwitch++; //Increments context switch value
                System.out.println("Context switch occurs");
            }
        }

        System.out.println("");
        System.out.println("The Time Quantum used was: " + n);

        //Calculate CPU Utilization
        double cpuUtilization= ((clock-cpu_idle_counter)/(clock))*100;
        System.out.println("The CPU Utilization is " + cpuUtilization+ " % ");

        //Calculating Throughput
        double clockDou=clock;
        double size= finishedQ.size();
        double throughPut=size/clockDou;
        System.out.printf("The through put is "+ "%.4f ",throughPut);
        System.out.println();

        //Calculating Average Turnaround Time
        int totalTurnAroundTime=0;
        for(int i=0; i<numProcess;i++){
            Process temp1=(Process) finishedQ.poll();
            int turnAroundTime= temp1.getCompletionTime()-temp1.getArrivalTime();
            temp1.addTurnAroundTime(turnAroundTime);
            totalTurnAroundTime=turnAroundTime+totalTurnAroundTime;
            finishedQ.add(temp1);
        }
        double avgTAT= totalTurnAroundTime/size;
        System.out.println("The average turnaround time is: "+ avgTAT);

        //Average Waiting Time
        int totalWaitTime=0;
        for(int j=0; j<numProcess;j++){
            Process temp2= (Process)finishedQ.poll();
            int waitTime=temp2.getTurnAroundTime()-temp2.getBurst();
            totalWaitTime=totalWaitTime+waitTime;
        }
        double avgWaiting= totalWaitTime/size;
        System.out.println("The average waiting time is: "+ avgWaiting);

        System.out.println("The number of context switches: "+contextSwitch);
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

public class Process {

    private String pid;
    private int arrivalTime;
    private int burst;
    private int contextSwitch;
    private int completeTime;
    private int turnaroundTime;
    private int serviceTime;

    public Process(String processId, int arriveTime, int burstCycle){
        pid=processId;
        arrivalTime=arriveTime;
        burst=burstCycle;
        contextSwitch=0;
        serviceTime=burstCycle;
    }
    /////////////////////////////////////////////////////////////////////
    public int getServiceTime(){
        return serviceTime;
    }

    public String getPID(){
        return pid;
    }

    public int getArrivalTime(){
        return arrivalTime;
    }

    public int getBurst(){
        return burst;
    }

    public int getTurnAroundTime(){
        return turnaroundTime;
    }

    public int getCompletionTime(){
        return completeTime;
    }
    ///////////////////////////////////////////////////////
    public void context(){
        contextSwitch++;
    }

    public void reduceServiceTime(){
        serviceTime--;
    }

    public void addCompletionTime(int time){
        completeTime=time;
    }

    public void addTurnAroundTime(int time){
        turnaroundTime=time;
    }
}


//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;

public class reader {

    private String fileName;
    private int processCounter=0;
    private ArrayList <Process> processKeeper=null;
    private BufferedReader br;
    private String[] value;

    public reader(String name) {
        fileName = name;
    }

    public void open() throws IOException {
        File input = new File(fileName);
        br = new BufferedReader(new FileReader(input));
        read();
    }

    public void read() throws IOException {
        br.readLine();
        String line = "";
        while ((line = br.readLine()) != null) {
            value = line.split(",");
            Process prop=processMaker(value);
            setPro(prop);
        }
        sortArrival();
        br.close();
    }

    public Process processMaker(String[] trial){
        String pid= "P"+trial[0];
        int arrival= Integer.parseInt(trial[1]) ;
        int burst= Integer.parseInt(trial[2]) ;
        Process pro=new Process(pid,arrival,burst);
        return pro;
    }
    public void setPro(Process p1){
        if(processKeeper==null){
            processKeeper=new <Process> ArrayList();
        }
        processKeeper.add(processCounter, p1);
        processCounter++;
    }

    public ArrayList <Process> processList(){
        return processKeeper;
    }
    public void sortArrival(){
        for(int i=1; i<processKeeper.size();i++){
            Process temp1=processKeeper.get(i);
            int tempArr=temp1.getArrivalTime();
            int j=i;
            while(j>0 && processKeeper.get(j-1).getArrivalTime()>tempArr){
                Process temp2=processKeeper.get(j-1);
                processKeeper.set(j, temp2);
                j--;
            }
            processKeeper.set(j, temp1);
        }
    }
}
```