COCSC09-OPERATING SYSTEM

CPU SCHEDULING PROJECT

CONTENTS

Amogh Garg
2020UCO1688

OBJECTIVE:

Implement a CPU Scheduling Algorithm (FCFS, SJF, Round Robin, etc) on any one of the below datasets.
Project Data sets that can be used for implementation are:

1. https://github.com/alibaba/clusterdata
2. https://github.com/google/cluster-data
3. http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains
4. https://github.com/r5k5/cloud-scheduling/tree/master/Dataset

I have performed CPU Scheduling on the third (highlighted) dataset.

THEORY:

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way. It is basically the pre-emptive version of First come First Serve CPU Scheduling algorithm.

Round Robin CPU Algorithm generally focuses on Time Sharing technique.

The period of time for which a process or job is allowed to run in a pre-emptive method is called time **quantum**.

Each process or job present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time, then the process will **end** else the process will go back to the **waiting table** and wait for the its next turn to complete the execution.

IMPLEMENTATION:

Round-Robin CPU scheduling algorithm has been implemented on the following dataset: Dataset

The dataset can be downloaded from the given link and can be made available for use after extracting the files.

After studying the parameters given in the dataset, it is assumed that all the processes arrive at the same time i.e 0^{th} second and the parameter "Timestamp" denotes the CPU Burst-time of that process in milliseconds. The scheduling algorithm has been implemented on one of the CSV files, however it can be implemented on other files also just by changing the path of the file in "pd.read_csv" function. The code written in Python programming language can be found below.

CODE:

```
#Importing Libraries

import numpy as np

import pandas as pd

# Reading the CSV file- "2013-7/4.csv"

# Change path here

# NOTE: If path of the file is changed, change the value of n in the main function accordingly

df=pd.read_csv('D:/Scheduling Project/rnd/rnd/2013-7/4.csv',header=0,sep=";+\t")

# First five data-points

df.head()

# Calculating some mean measures of the dataset

df.describe()


# General Information about the dataset
```

```
df.info()
# Function for finding the waiting time
def findWaitingTime(processes, n, bt,wt, quantum):
        rem_bt = [0] * n

        # Copy the burst time into rt[]
        for i in range(n):
                rem_bt[i] = bt[i]
        t = 0 # Current time

        # Keep traversing processes in round
        # robin manner until all of them are
        # not done.
        while(1):
                done = True

                # Traverse all processes one by
                # one repeatedly
                for i in range(n):

                        # If burst time of a process is greater
                        # than 0 then only need to process further
                        if (rem_bt[i] > 0) :
                                done = False # There is a pending process

                                if (rem_bt[i] > quantum) :

                                        # Increase the value of t i.e. shows
                                        # how much time a process has been processed
                                        t += quantum

                                        # Decrease the burst_time of current
                                        # process by quantum
                                        rem_bt[i] -= quantum
```

```
                # If burst time is smaller than or equal
                # to quantum. Last cycle for this process
                else:

                        # Increase the value of t i.e. shows
                        # how much time a process has been processed
                        t = t + rem_bt[i]

                        # Waiting time is current time minus
                        # time used by this process
                        wt[i] = t - bt[i]

                        # As the process gets fully executed
                        # make its remaining burst time = 0
                        rem_bt[i] = 0

        # If all processes are done
        if (done == True):
                break
# Function for finding the Turnaround time
def findTurnAroundTime(processes, n, bt, wt, tat):

    # Calculating turnaround time
    for i in range(n):
        tat[i] = bt[i] + wt[i]
#Function for finding the average-time
def findavgTime(processes, n, bt, quantum):
    wt = [0] * n
    tat = [0] * n

    # Function to find waiting time
    # of all processes
    findWaitingTime(processes, n, bt,wt, quantum)
```

```
        # Function to find turn around time
        # for all processes
        findTurnAroundTime(processes, n, bt,wt, tat)
        # Display processes along with all details
        print("Processes \tBurst Time[ms]\t\t Waiting","Time[ms]\t Turn-Around Time[ms]")
        total_wt = 0
        total_tat = 0
        for i in range(n):
                total_wt = total_wt + wt[i]
                total_tat = total_tat + tat[i]
                print(" ", i + 1, "\t\t", bt[i],"\t\t", wt[i], "\t\t", tat[i])
        print("\nAverage waiting time[ms] = %.5f "%(total_wt /n) )
        print("Average turn around time[ms] = %.5f "% (total_tat / n))
# Main Function for performing round-robin scheduing on the dataset
if __name__ =="__main__":
   #  8253 is the number of data-points in "2013-7/4.csv"
        proc = list(range(0,8253))
        n = 8253
        # Burst time of all processes
        burst_time = []
        for x in df['Timestamp [ms]']:
                burst_time.append(x)
        # Time quantum
        quantum = 1000000000;
        findavgTime(proc, n, burst_time, quantum)
#Scroll to the bottom of the result to see average time result
```

OUTPUT:

The link for the output and the code is:

https://github.com/amoghgarg20/Machine-Learning/blob/main/CPU%20Scheduling.ipynb

(Copy the link and paste on web-browser)

RESULT:

The following result was observed after successful implementation of the algorithm on “rnd/2013-7/4.csv”.

```
Average waiting time[ms] = 9792974830439.95117
```

```
Average turn-around time[ms] = 9794348750753.98633
```

SUBMITTED BY:

NAME: AMOGH GARG

ROLL NUMBER: 2020UCO1688

BRANCH- COE

SECTION- 3