HAMMAD KHAN MUSAKHEL

21801175

CS342-002

PROJECT 2 REPORT

28/03/2021

REPORT

The project massively catches programmer's attention towards the function of concurrent threads and data access. In my report, I will depict different scheduling algorithms and the average waiting time of indexed W threads in accordance with those scheduling algorithms. The W threads will have initially random-exponentially distributed values in terms of: inter-arrival time between two consecutive busts of a W thread and the length of each W thread burst.

In the first example, we will be assessing the scheduling behavior of W threads with values given in the project assignment report: schedule 3 75 100 200 1000 1500 ALGS.

N (no. of threads) = 3

Burst count = 75

minB = 100

avgB = 200

minA = 1000

avgA = 1500

Algorithms = FFS/ SJF/ PRIO / VRUNTIME

Form of retrieving burst length and inter-arrival time would be randomly generated as commanded in the project assignment report.

W Threads	PRIO Algorithm (waiting time ms)	FCFS Algorithm (Waiting time ms)	SJF Algorithm (Waiting time ms)	VRUNTIME Algorithm (waiting time ms)
W thread 1	56059	1792791	770770	1252251
W thread 2	1590459	1688687	944954	1720719
W thread 3	3154121	1577576	841184	2159157

The above table has values for 3 W threads running concurrently with a scheduler thread that schedules the execution of each W thread according to the mentioned algorithms. There are trends associated with each algorithm as well. Aside from the average waiting increasing from

W thread 1 to W thread 2, we also see a trend in the scheduling on PRIO; when the scheduler is executing, output is given when execution of a W thread's burst is being done. 'Consumer retrieved item' is the burnt length of the bursts and also the thread index of the thread being executed is depicted. We notice at the that W thread 1 is completely executed first and then W thread 2 and then W thread 3 as Priority is given to the thread with smaller index. In FCFS, the scheduling is very random as all the threads are adding one by one in a random order as the W thread sleeps for the inter-arrival time. However, the trend in time s noticed as time reducing from W thread 1 to W thread 3. In SJF, there is clear trend. The trend is seen in the output of the program with this algorithm as the 'Consumer retrieved item' aka the burst length is seen increasing thus the bursts with smaller burst length are executed first and the ones with bursts bigger are executed later. This algorithm is pretty effect as well, however, there is no time trend as the burst length generated is random and bursts are added randomly from each thread in the queue. In VRUNTIME, the trend is observed in the time as the W thread with higher index obtains a large vruntime value which makes it lower in priority to run. The trend on the output is that 'vruntime of the thread selected to executed' is seen to be increasing as the program proceeds and W thread with lower index are to exceed first due to the formula of vruntime (t(0.7 + 0.3i); thus, we also see the time trend increasing from lower indexed W thread to high indexed W thread.

The trends are mostly due to the scheduling algorithms. We can see that the most efficient algorithm is SJF in this case. The allotment of burst length and inter arrival time is randomly generated. In contrast, PRIO and VRUNTIME make it more in favor of W threads with lower indexes. The total execution time of PRIO and VRUNTIME is close. The claims made regarding the output trends have been testified with random values of N, Bcount, minB, avgB, minA, avgA.

The trends also stay true when the other form of execution where a file is used to generate inter arrival time and burst length.

Some other tables are also provided rot observe the behavior of randomly distributed numbers:

N (no. of threads) = 5

Burst count = 75

minB = 150

avgB = 250

minA = 1200

avgA = 1600

W threads	VRUNTIME Algorithm (waiting time ms)
W thread 1	2442440
W thread 2	3540537
W thread 3	4150146
W thread 4	5479474
W thread 5	5960955

The trend of increasing VRUNTIME is observed as mentioned above.

The claims made regarding the trends have been stated after numerous executions with different values of command N, avgB, avgA, etc.

I chose to depict the behavior of VRUNTIME as it seemed the most complicated of all the scheduling algorithms.

THE ABOVE CALCULATIONS HAVE BEE OBTAINED BY RUNNING THE .C PROGRAM ON THE TERMINAL OF UNIX BASED SYSTEM (macOS) AND CLION. IT DOES COMPILE AND RUN. ALL THE RELEVANT LIBRARIES TO REFERENCE FUNCTIONS HAVE BEEN INCLUDED IN THE .C FILE.