Runtime analysis for various scheduling algorithms

Ques1 :

SCHED_NPREEMPT_FCFS is implemented by not calling yield() on a timer interrupt in both usertrap() and kerneltrap() functions. So, timer interrupt induced preemption will be disabled. This makes SCHED_NPREEMPT_FCFS to behave similar to non-preemptive round-robin.

a. **Statistics of SCHED_NPREEMPT_FCFS on batch1.txt**
   Batch execution time: 11024
   Average turn-around time: 11021
   Average waiting time: 9913
   Completion time: avg: 13790, max: 13795, min: 13787

   Statistics of SCHED_PREEMPT_RR on batch1.txt
   Batch execution time: 11150
   Average turn-around time: 11137
   Average waiting time: 10017
   Completion time: avg: 12513, max: 12523, min: 12503

   The turnaround time is almost equal to batch execution time for both the scheduling algorithms, with each process completing the execution almost at same time. The difference between the minimum and maximum completion time is very low, indicating a fair scheduling.
   Batch execution time (and thus turnaround time) for both algorithms is almost equal because process in SCHED_NPREEMPT_FCFS gives up cpu periodically due to sleep and thus behaving similar to SCHED_PREEMT_RR which gives up cpu due to both timer interrupt and sleep.

b. **Statistics of SCHED_NPREEMPT_FCFS on batch2.txt**
Batch execution time: 12192
Average turn-around time: 12188
Average waiting time: 10968
Completion time: avg: 12864, max: 12869, min: 12860

Statistics of SCHED_PREEMPT_RR on batch2.txt
Batch execution time: 11140
Average turn-around time: 11130
Average waiting time: 10016
Completion time: avg: 11724, max: 11733, min: 11715

Observations in this case are similar to part (a) above except the fact that here process gives up cpu using yield call whereas in above part sleep call was made. On comparing batch1 and batch2 on same algorithm, batch2 will have slightly higher wait time because the process spends more time in RUNNABLE state due to yield() call because yield() call the process to RUNNABLE state whereas sleep will send the process to SLEEPING state (in SLEEP state, there is no waiting time).

c. **Statistics of SCHED_NPREEMPT_FCFS on batch7.txt**
Batch execution time: 11348
Average turn-around time: 6230
Average waiting time: 5095
Completion time: avg: 6709, max: 11828, min: 1567

Statistics of SCHED_PREEMPT_RR on batch7.txt
Batch execution time: 11243
Average turn-around time: 11230
Average waiting time: 10105
Completion time: avg: 12479, max: 12490, min: 12463

In SCHED_NPREEMT_FCFS, processes does not give up cpu while RUNNING until they complete their execution. Thus, the turnaround and waiting time of this algorithm is almost half of batch execution time. To be precise if t is the execution time of any process (total processes = 10), then batch execution time will be approximately 10t, average turnaround time will be approximately 5.5t and average waiting time will be approximately 4.5t. For the SCHED_PREEMT_RR, there is no sleep or yield call hence all processes will give up cpu only on timer interrupt and all progress cyclically. Hence all of them finish almost at same time leading to equal values of turnaround and batch execution time. In SCHED_PREEMT_RR fair scheduling is ensured.

Ques 2 :

Statistics of SCHED_NPREEMPT_SJF on batch2.txt

Batch execution time: 11299
Average turn-around time: 10759
Average waiting time: 9629
Completion time: avg: 11337, max: 11875, min: 11199
CPU bursts: count: 59, avg: 191, max: 384, min: 1
CPU burst estimates: count: 51, avg: 178, max: 289, min: 1
CPU burst estimation error: count: 49, avg: 90

Statistics of SCHED_NPREEMPT_SJF on batch3.txt

Batch execution time: 46532
Average turn-around time: 38637
Average waiting time: 33984
Completion time: avg: 39345, max: 47239, min: 24713
CPU bursts: count: 209, avg: 222, max: 382, min: 1
CPU burst estimates: count: 201, avg: 219, max: 325, min: 1
CPU burst estimation error: count: 199, avg: 27

The CPU burst estimation error is lesser in batch3.txt than in batch2.txt (90 for batch2.txt and 27 for batch3.txt).
The ratio :
(the average CPU burst estimation error per estimation instance)/(the average CPU burst length) also decreases in batch3.txt . This ratio is approximately 0.5 for batch2.txt and less than 0.13 for batch3.txt.
This can be explained by looking at the number of CPU burst estimates made in batch2.txt and batch3.txt (batch3.txt is more computationally intensive than batch2.txt). Observe that both batch2.txt and batch3.txt create 10 identical processes (identical in a batch; different for the two batches) and it can be assumed that the actual cpu bursts are same (say, t) in a given batch for different processes. With this assumption the estimated cpu burst at time (n) comes out to be :

s(n) = t.(1-(½)^n) which converges to t as n tends to infinity.

Batch2.txt makes 51 CPU burst estimates while batch3.txt makes 201 CPU burst estimates. Thus the estimated CPU burst converges very close to the actual value in case of batch3.txt and the contribution to CPU-burst estimation error in later iterations is much lesser in batch3.txt than in batch2.txt, thus bringing down the average error and the ratio defined above.

Ques 3:

Statistics of SCHED_NPREEMPT_FCFS on batch4.txt

Batch execution time: 8354
Average turn-around time: 8351
Average waiting time: 7516
Completion time: avg: 9049, max: 9053, min: 9046

Statistics of SCHED_NPREEMPT_SJF on batch4.txt

Batch execution time: 8561
Average turn-around time: 6198
Average waiting time: 5342
Completion time: avg: 6745, max: 9106, min: 4128
CPU bursts: count: 60, avg: 142, max: 273, min: 1
CPU burst estimates: count: 50, avg: 138, max: 220, min: 56
CPU burst estimation error: count: 50, avg: 64

The batch execution time for both the scheduling algorithms is similar, but the turnaround time of SJF is smaller than FCFS. This happens because SCHED_NPREEMPT_FCFS behaves similar to non-preemptive round-robin and has a very poor turnaround time as all the processes end their execution at almost the same time. On the other hand, it can be mathematically proven that SCHED_PREEMPT_SJF achieves the lowest turnaround time which is in accordance with the observed values.

Ques 4:

Statistics of SCHED_PREEMPT_RR on batch5.txt

Batch execution time: 11950
Average turn-around time: 11938
Average waiting time: 10738
Completion time: avg: 12505, max: 12516, min: 12478

Statistics of SCHED_PREEMPT_UNIX on batch5.txt

Batch execution time: 11357
Average turn-around time: 7428
Average waiting time: 6287
Completion time: avg: 8007, max: 11938, min: 3956

Statistics of SCHED_PREEMPT_RR on batch6.txt

Batch execution time: 11199
Average turn-around time: 11185
Average waiting time: 10065
Completion time: avg: 12046, max: 12059, min: 12027

Statistics of SCHED_PREEMPT_UNIX on batch6.txt

Batch execution time: 12822
Average turn-around time: 8344
Average waiting time: 7062
Completion time: avg: 9050, max: 13531, min: 4386

SCHED_PREEMPT_RR has a much higher turnaround time than SCHED_PREEMPT_UNIX. This is because for SCHED_PREEMPT_RR all the processes end their execution at almost the same time. On the other hand, In unix scheduling as the priorities are different, the low priority tasks get completed first, so certain processes get completed very fast and some may complete at the very end. So the average turnaround time comes in between them which will be smaller than batch execution time.
The maximum and minimum completion time of SCHED_PREEMPT_RR have a very small difference thus indicating a fair scheduling, while the maximum and minimum completion time of SCHED_PREEMPT_UNIX has a large difference which means there are some processes which end sooner while others that complete execution at the very end.

Implementation Details of Algorithms:

SCHED_NPREEMPT_SJF:

The SCHED_NPREEMPT_SJF is implemented as follows:

- It iterates over the proc array and find out the RUNNABLE batch process with minimum cpu_burst estimate time. In the iteration, if it encounters a RUNNABLE non-batch process, it returns immediately and schedules that non-batch process.
- If no RUNNABLE non-batch process is found, it schedules the RUNNABLE batch process with min_cpu_burst.
- If two batch processes have same minimum cpu burst estimate, we break the tie by selecting the process with longer waiting time.
- If we do not find any RUNNABLE process then we loop over the proc table again till we find a suitable process according to the criteria described in above points.
- Cpu bursts are calculated when the process transitions from RUNNING state to RUNNABLE (in yield), SLEEP (in sleep), ZOMBIE (in exit).

SCHED_PREEMPT_UNIX:

The SCHED_PREEMPT_UNIX is implemented as follows:

At each loop over the proc array, it does the following

- It loops over the proc table and selects the RUNNABLE non-batch process for execution when it finds one and keeps looping to update the dynamic priorities of batch processes.
- If it does not find any RUNNABLE non-batch process, it selects the batch process with minimum dynamic priority.
- If two batch processes have equal minimum dynamic priority, then we select the process with maximum waiting time. All these steps are performed in single iteration of proc table.
- If we do not find any RUNNABLE process then we loop over the proc table again till we find a suitable process according to the criteria described in above points.
- cpu usage is updated by SCHED_PARAM_CPU_USAGE if a process gives up cpu using yield else we update it by SCHED_PARAM_CPU_USAGE/2 in sleep call.