Study Set for Lecture 06: CPU Scheduling

Due Oct 5 at 11:59pm **Points** 10 **Questions** 15
Available Sep 29 at 12pm - Dec 8 at 8pm 2 months **Time Limit** None
Allowed Attempts Unlimited

Instructions

Review lecture notes from **lect06_CPU_Scheduling.pdf**. The archive of the accompanying code is in **lect06code.zip**.

Then answer the questions from this study set and submit them to gain access to the further part of the course.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	99 minutes	10 out of 10
LATEST	Attempt 2	99 minutes	10 out of 10
	Attempt 1	1,586 minutes	0 out of 10 *

* Some questions not yet graded

ⓘ Correct answers are hidden.

Score for this attempt: **10** out of 10
Submitted Oct 4 at 7:25pm
This attempt took 99 minutes.

Question 1 — 0 / 0 pts

Explain what a CPU burst is and how it is used by CPU scheduler.

Next, compute a series of predictions for CPU bursts for a process with the following sequence of CPU bursts:

```
10, 4, 6, 2, 10, 8, 2, 8, 10, 4, 6
```

Assume alpha = 0.75
Start with tau_0 = 8

Show all intermediate details of your computation.

NOTE:

Assume that you are receiving each of the burst values in the sequence one at a time.

So, the first value estimate is computed as:

```
tau_1 = 0.75 * 10 + (1 - 0.75) * 8
```

Your Answer:

A CPU burst is the time that a process has control of the CPU, and the time is how long it has the CPU to complete. The scheduler uses the burst to determine which process should go when, depending on what scheduling it's using.

tau(n+1) - predicted value for the (n+1)th CPU burst

t(n) - actual length of n-th CPU burst

tau(n) - previous average

0 <= alpha <= 1 average coefficient

FORMULA:

taun(n+1) = (alpha)(t(n)) + (1-(alpha))(tau(n))

tau1 = (.75*10) + ((.25)8) = 7.5 + 2 = 9.5

tau2 = (.75*4) + ((.25)9.5) = 3 + 2.375 = 5.375

tau3 = (.75*6) + ((.25)5.375) = 4.5 + 1.34375 = 5.84375

tau4 = (.75*2) + ((.25)5.84375) = 1.5 + 1.4609375 = 2.9609375

tau5 = (.75*10) + ((.25)2.9609375) = 7.5 + 0.740234375 = 8.240234375

tau6 = (.75*8) + ((.25)8.240234375) = 6 + 2.060058594 = 8.060058594

tau7 = (.75*2) + ((.25)8.060058594) = 1.5 + 2.015014648 = 3.515014648

tau8 = (.75*8) + ((.25)3.515014648) = 6 + 0.878753662 = 6.878753662

tau9 = (.75*10) + ((.25)6.878753662) = 7.5 + 1.719688416 = 9.219688416

tau10 = (.75*4) + ((.25)9.219688416) = 3 + 2.304922104 = 5.304922104

tau11 = (.75*6) + ((.25)5.304922104) = 4.5 + 1.326230526 = 5.826230526

Question 2

0 / 0 pts

Prove that in exponential averaging the further a value is in the past, the less impact it has on the average value.

Your Answer:

In exponential averaging the further the value the less impact it has since both alpha and (1-alpha) are less than or equal to 1, therefore each successive term has less weight than its predecessor, because each one divides to be smaller and smaller.

Question 3

0 / 0 pts

List and describe scheduling criteria for scheduling algorithms.

Your Answer:

Different **CPU scheduling algorithms (https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/)**

have different properties and choice of a particular algorithm depends on the various factors. Many criteria have been suggested for comparing CPU scheduling algorithms.

The criteria include the following:

1. **CPU utilisation –**
   The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically CPU utilisation can range from 0 to 100 but in a real time system it varies from 40 to 90 percent depending on the load upon the system.
2. **Throughout –**
   A measure of the work done by CPU is the number of processes being executed and completed per unit time. This is called throughput. The throughput may vary depending upon the length or duration of processes.
3. **Turnaround time –**
   For a particular process, an important criteria is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in ready queue, executing in CPU and waiting for I/O.
4. **Waiting time –**
   A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue.
5. **Response time –**
   In an interactive system turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to user. Thus another criteria is the time taken from submission of process of request until the first response is produced. This measure is called response time.

There are various CPU Scheduling algorithms such as-

- **First Come First Served (FCFS) (https://www.google.com/amp/s/www.geeksforgeeks.org/program-for-fcfs-cpu-scheduling-set-1/amp/)**
- **Shortest Job First (SJF) (https://www.google.com/amp/s/www.geeksforgeeks.org/program-for-shortest-job-first-or-sjf-cpu-scheduling-set-1-non-preemptive/amp/)**

- Longest Job First (LJF) (https://www.google.com/amp/s/www.geeksforgeeks.org/longest-job-first-ljf-cpu-scheduling-algorithm/amp/)
- Priority Scheduling (https://www.google.com/amp/s/www.geeksforgeeks.org/program-for-priority-cpu-scheduling-set-1/amp/)
- Round Robin (RR) (https://www.google.com/amp/s/www.geeksforgeeks.org/program-round-robin-scheduling-set-1/amp/)
- Shortest Remaining Time First (SRTF) (https://www.google.com/amp/s/www.geeksforgeeks.org/introduction-of-shortest-remaining-time-first-srtf-algorithm/amp/)
- Longest Remaining Time First (LRTF) (https://www.google.com/amp/s/www.geeksforgeeks.org/longest-remaining-time-first-lrtf-cpu-scheduling-algorithm/amp/)

Question 4

0 / 0 pts

Evaluate CPU scheduling algorithm based entirely on process priorities.

Are there any potential pitfalls? If yes, how can the algorithm be changed to address them?

Your Answer:

The pitfall of scheduling based entirely on priority is that low priority processes may never execute.

The solution is to use "aging". This means as time progresses the priority of the process increases.

Each process is given a priority number (int) which corresponds to how how important it is to the cpu. The lower the number means the higher the priority.

Question 5

0 / 0 pts

Describe with details the algorithm of using a roulette wheel algorithm to select processes from a number of queues with different priorities.

Your Answer:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
srandom(time(NULL));
double r = ((double)random())/RAND_MAX;
printf("r = %lf\n", r);
if (r < 0.1)
printf("Hit 1st slice\n");
else if (r < 0.25)
printf("Hit 2nd slice\n");
else if (r < 0.5)
printf("Hit 3rd slice\n");
else if (r < 0.65)
printf("Hit 4th slice\n");
else if (r < 0.85)
printf("Hit 5th slice\n");
else
printf("Hit 6th slice\n");
}
```

This example above will split the percentages in slices (like a wheel). This will help us change the probability of each slice that will be hit when we spin the wheel or throw a dart etc.

Question 6

0 / 0 pts

List and describe with details at least three approaches to selecting processes from priority-based multi-level queues.

Your Answer:

- There are several different criteria to consider when trying to select the "best" scheduling algorithm for a particular situation and environment, including:
  - **CPU utilization** - Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles. On a real system CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)
  - **Throughput** - Number of processes completed per unit time. May range from 10 / second to 1 / hour depending on the specific processes.
  - **Turnaround time** - Time required for a particular process to complete, from submission time to completion. (Wall clock time.)
  - **Waiting time** - How much time processes spend in the ready queue waiting their turn to get on the CPU.
    - (**Load average** - The average number of processes sitting in the ready queue waiting their turn to get into the CPU. Reported in 1-minute, 5-minute, and 15-minute averages by "uptime" and "who".)
  - **Response time** - The time taken in an interactive program from the issuance of a command to the ***commence*** of a response to that command.
- In general one wants to optimize the average value of a criteria (Maximize CPU utilization and throughput, and minimize all the others.) However some times one wants to do something different, such as to minimize the maximum response time.
- Sometimes it is most desirable to minimize the ***variance*** of a criteria than the actual value. I.e. users are more accepting of a consistent predictable system than an inconsistent one, even if it is a little bit slower.

Question 7

0 / 0 pts

Design a CPU scheduler that supports "real-time" (RT) and "regular" (OTHER) processes. The RT processes should be preferred over OTHER.

Explain all details of the architecture and the algorithms that will achieve that objective.

Your Answer:

Real-time (“soft” R-T)
- FIFO, RR, DEADLINE (experimental)
- FIFO – no time slice
- RR – time slice
- DEADLINE – complicated; see man page
- highest priority process always runs first
- “r-t” processes assigned static priorities 1 to 99

Question 8

0 / 0 pts

Design and describe with details a priority-based scheduling algorithm that uses a 4-level feedback queue.

The algorithm should address process entrance (i.e., the starting queue), process promotion and demotion, queue selection, and process selection from the selected queue.

Your Answer:

In **computer science** (https://en.wikipedia.org/wiki/Computer_science), a **multilevel feedback queue** is a **scheduling** (https://en.wikipedia.org/wiki/Scheduling_(computing)) algorithm. Solaris 2.6 Time-Sharing (TS) scheduler implements this algorithm.[1] (https://en.wikipedia.org/wiki/Multilevel_feedback_queue#cite_note-1) The MacOS and Microsoft Windows schedulers can both be regarded as examples of the broader class of multilevel feedback queue schedulers.[2] (https://en.wikipedia.org/wiki/Multilevel_feedback_queue#cite_note-2) This scheduling algorithm is intended to meet the following design requirements for **multimode systems** (https://en.wikipedia.org/w/index.php?title=Multimode_systems&action=edit&redlink=1):

1. Give preference to short jobs.
2. Give preference to **I/O bound** (https://en.wikipedia.org/wiki/I/O_bound) processes.

3. Separate processes into categories based on their need for the processor.

Three queues:

● Q0 – time quantum 8 ms

● Q1 – time quantum 16 ms

● Q2 – anything longer than 24 ms

Scheduling:

● Processes in Q2 considered only if Q0 and Q1 empty

● Processes in Q1 considered only if Q0 empty ● A new job enters queue Q0 which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q1.

● At Q1 job is again scheduled FCFS and receives 16 additional milliseconds.

● Only if Q0 is empty!

● If it still does not complete, it is preempted and moved to queue Q2.

Question 9

0 / 0 pts

Consider the following task workload pattern:

```
<Process> <CPU Burst Time>
   P1            7
   P2           10
   P3            3
   P4            8
   P5           15
   P6            2
   P7            4
```

Assuming the FCFS scheduling policy, compute the average waiting time. Use the following format to illustrate the details of your computation:

```
T01: P1(3)
T04: P3(6)
T10: P6(5)
T15: P4(...)
...
```

Your Answer:

T01: p1(7)

T7: P2(10)

T17: P3(3)

T20: P4(8)

T28: P5(15)

T43: P6(2)

T45: P7(4)

T49: done

Question 10

0 / 0 pts

Consider the following task workload pattern:

```
<Process><CPU Burst Time><Arrival Time>
   P1            7              1
   P2           10              3
   P3            1              5
   P4            8             11
   P5           15             14
   P6            2             15
   P7            2             21
```

Assuming the SRTF scheduling policy, compute the average waiting time. Use the following format to illustrate the details of your computation:

```
T01: P1(3)
T04: P3(6)
T10: P6(5)
T15: P4(...)
...
```

Your Answer:

T01: P1(7)

T03: P1(5) Q: P2(10)

T05: P3(1) Q: P1(3) P2(10)

T06: P1(3) Q: P2(10)

T09: P2(10)

T11: P2(8) Q: P4(8)

T14: P2(5) Q: P4(8) P5(15)

T15: P6(2) Q: P2(4) P4(8) P5(15)

T17: P2(4) Q: P4(8) P5(15)

T21: P7(2) Q: P4(8) P5(15)

T23: P4(8) Q:P5(15)

T31: P5(15)

T46: done

Shortest-Job-First (SJF) Scheduling

● Associate with each process the length of its next CPU burst.

● Use CPU bursts to schedule the process with the shortest time

Two schemes:

non-preemptive -

● once CPU given to the process it cannot be interrupted until completes its CPU burst

preemptive-

● if a new process arrives with CPU burst length less than remaining time of current executing process, preempt

● this scheme is know as the Shortest-Remaining-Time-First (SRTF)

Question 11

0 / 0 pts

Assume that the a scheduler uses a round robin scheme with a quantum time of 3.

Assuming that only the following processes compete for CPU, compute the average turnaround time for time quanta 1, 2, 3, 4, 5, and 6:

```
<Process><CPU Burst Time>
   P1            6
   P2            3
   P3            1
   P4            7
```

1. Show the detailed timeline of scheduling events for each time quantum value.
2. Show an ordered list of the average turnaround times along with their corresponding time quanta.

Your Answer:

T0 : idle

T01: P1(6)

T02 : P1(5) Q: P2(3)

T03: P1(4) Q: P2(3) P3(1)

T04: P2(3) Q: P3(1) P4(7) P1(3)

T05: P2(2) Q: P3(1) P4(7) P1(3)

T06: P2(1) Q: P3(1) P4(7) P1(3)

T07: P3(1) Q: P4(7) P1(3)

T08: P4(7) Q:P1(3)

T09: P4(6) Q:P1(3)

T010: P4(5) Q:P1(3)

T011: P1(3) Q:P4(4)

T012: P1(2) Q:P4(4)

T013: P1(1) Q:P4(4)

T014: P4(4)

T015: P4(3)

T016: P4(2)

T017: P4(1)

T018: done

Each process gets a small unit of CPU time (time quantum)

● usually 10-100 milliseconds

● After this time has elapsed, the process is preempted (interrupted) and added to the end of the ready queue.

● If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units at once. No process waits more than (n - 1)q time units.

Performance:

● q large ⇒ FIFO

● q small ⇒ better system response

- however, q must be still relatively large with respect to context switch (otherwise overhead is too high) -- see next slides

● There might be another algorithm used for selection of the process that will execute in the next time quantum

- the ready queue can be a FIFO or a priority queue

Question 12

0 / 0 pts

Consider the following task workload pattern:

```
<Process><CPU Burst Time><Arrival Time>
   P1            7              1
   P2           10              3
   P3            1              5
   P4            8             11
   P5           15             14
   P6            2             15
   P7            2             21
```

Assuming the SJF scheduling policy, compute the average waiting time. Use the following format to illustrate the details of your computation:

```
T01: P1(3)
T04: P3(6)
T10: P6(5)
T15: P4(...)
...
```

Your Answer:

T01: P1(7)

T03: P1(5) Q: P2(10)

T05: P1(3) Q: P2(10) P3(1)

T08: P3(1) Q: P2(10)

T09: P2(10)

T11: P2(8) Q: P4(8)

T14: P2(5) Q: P4(8) P5(15)

T15: P2(4) Q:P4(8) P5(15) P6(2)

T19: P6(2) Q:P4(8) P5(15)

T21: P7(2) Q:P4(8) P5(15)

T23: P4(8) Q:P5(15)

T33: P5(15)

T48: done

Question 13

0 / 0 pts

Explain how Completely Fair Queueing (CFQ) algorithm is used to implement Completely Fair Scheduler (CFS).

Your Answer:

CFQ is one of the input/output scheduler for the Linux kernel and is the current default scheduler in the Linux kernel.

What is kernel ?
Kernel is the central part of an operating system. It manages the operation between the hardware and the software. Every operating system has a kernel, for example the Linux kernel.

Completely Fair Queuing (CFQ) scheduler:
The Completely Fair Queuing (CFQ) scheduler is the I/O scheduler. The CFQ scheduler maintains a scalable per-process requests submitted by the processes into the number of per-process I/O queue and then allocate time for each of the queues to access the resource. This is done on the basis of the I/O priority of the given process.
In case of asynchronous requests, all the requests from all the processes are batched together into fewer queues according to their process's I/O priority.

The CFQ scheduler divides the processes into 3 separates class:

1. Real time (highest priority)
2. Best effort
3. Idle (lowest priority)

The real-time and best-effort scheduling classes are further subdivided into eight-eight I/O priorities.These priorities are 0 to 7. Zero(0) being the highest and 7 the lowest and 4 is the default priority within the class.

Process in the real time class are always performed before processes in best effort class, which is always performed before processes in the idle class. This means that due to the higher priority of the real time class, rest both class have to starve of the processor time. Processes are assigned to the best effort class by default.

When there is no other I/O pending in the system then the idle scheduling class are only serviced. Thus, it is very important to only set the I/O scheduling class of a process to idle if I/O from the process is not at all required for making further progress.

Question 14

0 / 0 pts

Describe and compare approaches to evaluate scheduling algorithms.

Your Answer:

Algorithm Evaluation

● Deterministic modeling

- takes a particular predetermined workload and defines the performance of each algorithm for that workload

● Queueing models

- Queueing Theory - Mathematical analytical models using probabilistic distributions of job arrivals, CPU bursts, etc.

● Simulation

- simulators using probabilistic distributions or trace tapes (captures of real data)

● Implementation

- measuring performance of a live system

Question 15

10 / 10 pts

I have submitted answers to all questions in this study set.

- ◉ True

False

Quiz Score: **10** out of 10