Homework
Part 1: Virtualization – 2) Scheduling

master@d08e360 (20220915-171922)

P. Mainini / E. Benoist / C. Fuhrer / L. Ith

BTI1341 / Fall 2022-23

1 Scheduling Strategies

Suppose we have four jobs A, B, C and D. The time of arrival and duration in seconds for each of them is given in the following table:

Job	Arrival	Duration
A	0	50
B	0	60
C	10	30
D	10	20

1.1 First In, First Out (FIFO)

Calculate the *turnaround-* and *response time* for every task, when the tasks are scheduled using the FIFO strategy. What is the average turnaround- and response time?

1.2 Shortest Time-to-Completion First (STCF)

Calculate the *turnaround-* and *response time* for every task, when the tasks are scheduled using the STCF strategy. What is the average turnaround- and response time?

1.3 Round Robin

Calculate the *turnaround-* and *response time* for every task, when the tasks are scheduled using the round robin strategy with time slices of 5 seconds. What is the average turnaround- and response time?

2 Multi-Level Feedback Queues

Given a MLFQ system with 4 queues and the following 3 tasks:

Job	Arrival	Duration	I/O
A	0	50	–
B	10	30	every 10s
C	10	20	every 2s

2.1 Scheduling With Rules 1-4

Create a diagram containing the 4 queues and draw the scheduling of all tasks until they have completed. Only use MLFQ rules 1-4 (without rules 4a, 4b and 5)!

Assume the scheduling quantum for the per-queue round robin scheduler to be 5s; also, all I/O operations take 5s as well.

2.2 Scheduling With Rules 1-5

Create a diagram containing the 4 queues and draw the scheduling of all tasks until they have completed. This time, use MLFQ rules 1-5 (do still not use rules 4a and 4b)!

Assume that priority boosting occurs every 20 seconds; scheduling quantum and I/O duration is the same as for the previous exercise.

3 Exploring Linux CFS

The goal of this task is to get a bit of hands-on experience with the *Linux CFS* scheduler. Perform the following tasks (in your virtual machine or on your own GNU/Linux system):

1. Take a look at the `nice` and `renice` tools. Apply them to some processes on your system; can you see any effect (e.g. using `top` or `htop`)?
   👉 Try running some synthetic workloads, like e.g. a program which performs an endless loop. Start it alone or multiple times. If your GNU/Linux system runs in a VM, you might also try to add/remove virtual CPUs.

2. Find out the actual values for `sched_latency` and `min_granularity`.
   a) Can you explain them?
   b) Can you modify them? Are there any limits?
   c) If you could modify them, what effects would you expect? How?

3. Take a look at the `perf` tool (usage examples can be found in [Gre]). Try to understand how it works by running it on your own system. For this, you might need to install the `linux-perf` package on your system.

4 Scheduling Simulation ★

The simulation homework from *OSTEP, Chapters 7, 8 and 9* (`scheduler.py`, `mlfq.py` and `lottery.py`) provides additional exercises for the different scheduling strategies. It can be found at [ost] and will further strengthen your skills.

5 Multiprocessor Scheduling ★

If you are interested in the topic of multiprocessor scheduling, read [ADAD] and/or conduct the corresponding simulation homework (`multi.py`) from [ost].

References

[ADAD] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, *OSTEP, Chapter 10, Multiprocessor Scheduling (Advanced)*, http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-multi.pdf.

[Gre] Brendan Gregg, *perf Examples, Section 6.7, Scheduler Analysis*, http://www.brendangregg.com/perf.html#SchedulerAnalysis.

[ost] *GitHub.com, remzi-arpacidusseau/ostep-homework*, https://github.com/remzi-arpacidusseau/ostep-homework.