Operating Systems Course Assignment 2

Year: 2016/2017

1 Linux Scheduler

The highlight of any OS is the scheduler. Linux comes with a number of different scheduling algorithms (policies in Linux terminology) that can be set [1].

The supported schedulers include deadline, Round Robin, FIFO, CSF, among others.[2] There are even more algorithms available from hackers and linux gurus, such as the Linux Brain-Fuck Scheduler [3]

The scheduling policy has impact on the performance of an application. Depending on the application type, you can gain/lose a lot when it comes to performance.

1.1 Performance evaluation

When comparing different scheduling policies for a given application, there are multiple metrics that are usually considered, for example:

1. Throughput: How many units of work/threads/processes finish per unit time.
2. Latency: On average, how long does it take for an application to finish?
3. tail latency: sorting applications by how long they take, how long does it take for, e.g., the application that is slower than 95% of all other applications, to finish?
4. Waiting time: How long does an application stay in the Waiting queue on average?
5. distributions: Plotting the distributions of the above numbers, instead of providing single numbers, e.g., the average or the 95th percentile latency.

The above set are just an example, but there are many more that can be considered.

2 Your task

Write a multi-threaded program to test and compare two or more of the Linux scheduling policies available in the main Linux kernel or by others. Your program should run in user space. You should include performance metrics that

[1] See for example: http://man7.org/linux/man-pages/man7/sched.7.html

[2] See this presentation:http://retis.sssup.it/~jlelli/talks/rts-like14/SCHED_DEADLINE.pdf

[3] See:https://en.wikipedia.org/wiki/Brain_Fuck_Schedulerandhttp://ck.kolivas.org/patches/bfs/

make sense for your choice of the benchmarked algorithms, and you do not need to confine yourself to the ones above.

Your program should be adequate to what you are trying to test and should be capable of running for time sufficient to draw conclusions.

Deliver a well written report, and some beautiful code :). Use the department's gitlab, your online github repo, or some other code sharing facility to share code with both teachers.

Two different policies is the absolute minimum. Anything more will be a bonus. Considering how the same policy's configuration affects the scheduling is a bonus too.

For inspiration, take a look at `http://cs.unm.edu/~eschulte/classes/cs587/data/bfs-v-cfs_groves-knockel-schulte.pdf`, BUT DO NOT USE latt.c.

3 Deadline

14 December, 2016