Tomson Li, Yifan Wang, Lien Zhu

CSE 522

Project Proposal

1. **Propose a modification of, significant use of, or extension to, the behavior of the Linux kernel or of user space infrastructure that uses the Linux kernel.**

We propose a new scheduling algorithm that intentionally delivers inefficient performance. It will aim to prioritize processes with **~~longest execution time estimates and~~** lowest priority. We will also experiment with the kernel provided preemption configurations with our scheduler to determine any relations between the scheduler and preemption setting, **namely by testing performance with preemption and with preemption completely disabled**.

1. **Clearly explain the purpose of such modification, use, or extension.**

This extension and significant use of the Linux kernel may serve as a sort of worst case, performance lowest bound. For instance, if we implement what is theoretically the worst scheduling algorithm (within some constraints i.e., at least one process must be scheduled if one is available, no extraneous time wasting other than poor prioritization) this can be a point of comparison for kernel developers who may want to create a new scheduling algorithm. Reviewers may be able to use our scheduler as a comparison point.

If the above points are nullified, then at the very least this project will serve to enhance our understanding of the kernel scheduling algorithms.

1. **Identify any kernel and/or user space files that will require modification or extension.**

Understand code under /kernel/sched/. Specifically: /kernel/sched/core.c, fair.c, and rt.c for understanding the scheduling classes for both CFS and real-time scheduling class (FIFO and RR). Under /kernel/sched, add our worst-case scheduling class and integrate into schedule core. We may need to add more macro in the kernel to support our newly added scheduling class.

1. **Identify any kernel and/or user space data structures that will be changed or extended.**

The schedule class structure would be changed to accommodate newly added scheduling class.

1. **Identify any kernel and/or user space concepts, control-flow paths, and/or data-flow paths that will need to be changed or extended.**

In the schedule core, we would need to add entry to use our scheduling class when scheduling (picking the next task).

1. **Propose a set of test cases that demonstrate and test the modification, use, or extension.**

Run multiple programs to exhaust our CPU and evaluate the performance to our scheduling class/system with preemption configuration on and **completely** off. In addition, to understand the performance of our scheduling class when utilization is low, i.e., the difference between CFS/RT scheduling class and our worst case scheduling class when resources is sufficient.

1. **Identify any additional kernel modules or user space programs that will be produced to implement, or to validate and test, the modification, use, or extension.**

Design and compile a user space program that can satisfy our test case. A kernel module may be needed to support our change to new scheduling class (pending).

1. **Include an eight-week planned schedule for the project, including detailed design, implementation, initial evaluation, project presentation, and project write up phases.**

|  |  |
| --- | --- |
| 2/5-2/11 | Identify class project topic, discuss project scope, and proposal writeup. |
| 2/12-2/18 | Background knowledge review, including CPU schedule mechanism, common process schedule algorithm, e.g. FIFO, **Earliest** Deadline, and Round Robin. |
| 2/19-2/25 | Read /kernel/sched/core.c, study CFS algorithm, which will be the primary schedule strategy we study and modify.  Receive proposal feedback, discuss feasible strategies that can be used on creating the worst-case scheduler. |
| 2/26-3/11  (Two weeks) | Experiment with different worst-case strategy on the process scheduler. Come up at least two to three feasible worst-case schedular. |
| 3/12-3/18 | Spring break.  Observe the efficiency of modified worst-case scheduler with various tasks, e.g. IO heavy tasks, compute heavy tasks. Conduct the experiment with preemption on/off to acquire the lower-bound of the process running time. |
| 3/19-3/25 | Conduct the similar experiment on different scheduler, RR, FIFO, CFS. Compare our experiment results with that from the original kernel-default schedular. Discuss experiment results and forms a conclusion on the optimization of efficiency of the system default scheduler. |
| 3/26-4/1 | Milestone writeup. |