Team:

syakter (Samin Akter)
jhkung (Jonathan Kung)
stesuh (Steven Suh)
emgwong (Emma Wong)

CMPS 111 Spring 2019
Asg 2: Scheduler - DESIGN DOC

OVERVIEW

The purpose of this assignment is to modify the FreeBSD scheduler to run priority queues and implement splatter scheduling.

GOALS

As outlined in the assignment pdf itself, we need to support the following cases:

- Case 1: FIFO queues and current FreeBSD scheduling
- Case 2: Priority queues and current FreeBSD scheduling
- Case 3: FIFO queues and splatter scheduling
- Case 4: Priority queues and splatter scheduling

And make sure that the following requirements are met:

- Random generator should perform the same when tested with seed values
- Random numbers should be uniformly distributed
- Write benchmarks to compare performance statistics of different cases
    - Runtime
    - Page faults
- Include bar graph to show performance comparison among the 4 cases

PROCESS

For this assignment we spent the most time figuring out how the kernel

worked before implementing priority queues and splatter scheduling.

There were somethings that we ended up not having to alter given our approach such as the nice() call and the sched_ule.c file. We ended up just making modifications to the kern_switch.c file, which was all that was needed based on our design.

ARCHITECTURE

In order to switch between cases we implemented a kernel flag called CMPS_111_SCHED which when set to 1 represents case 1, and when set to 2 represents case 2, etc. In the runq_add function, we check which case was requested by getting the environment variable, which then determines what next course of action to take within the code.

Before doing anything, we check if the thread is a user thread using its process id number.

If we are given case 3 or 4 we know that splatter scheduling is required so the variable pri is set to return a random runq within its given type. The random runq priority number is determined using the random() function. If CMPS_111_SCHED is 1 or 2, we select the runq based on the thread's priority(as is the base case). We finally set the runq using the function runq_setbit.

Once we know which runq we want to add the thread to, we check CMPS_111_SCHED again to see whether we want to add the thread to the queue in FIFO or priority order.

For the priority queue implementation, we insert into a runq which is always sorted from highest priority to lowest priority. This way, the highest priority thread is at the head of the queue, and the lowest priority thread is always at the end of the queue. It's important to note that this way of insertion is only for Case 1 or 2. With this design, it is not necessary to modify the current FIFO scheduler that selects a thread to run. In the insertion process, we use a for each loop to go through the threads in the runq. If the priority of the thread being added is greater than the current thread, the new thread is inserted

before the current thread. If the queue is empty, the thread is simply placed at the head.

ANALYSIS

Note that the benchmark code we used was posted on the CMPS 111-SPRING-19-01 Piazza by Axel Eduardo Garcia, and code written by Andrew Purcell. The link to the post can be found here: https://piazza.com/class/jtt1omu5bb2sd?cid=347

Case 1:

- Runtime:
  - -User: 52.191s
  - -System: 8.869s
- Page faults:
  - Soft: 303
  - Hard: 0
- Voluntary Context Switches: 36
- Involuntary Context Switches: 4777

Case 2:

- Runtime:
  - User: 51.880s
  - System: 8.778s
- Page faults:
  - Soft: 303
  - Hard: 0
- Voluntary Context Switches: 37
- Involuntary Context Switches: 4681

Case 3:

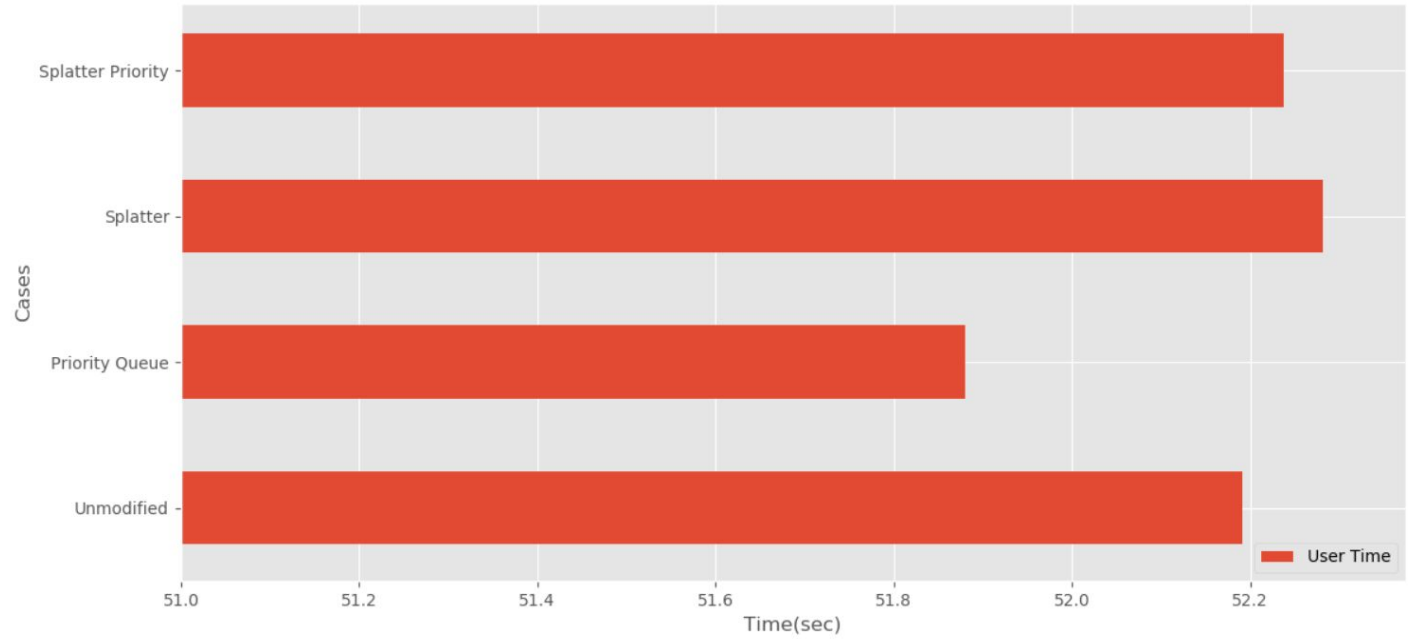
- Runtime:
  - User: 52.281s
  - System: 8.616s
- Page faults:
  - Soft: 303
  - Hard: 0
- Voluntary Context Switches: 36

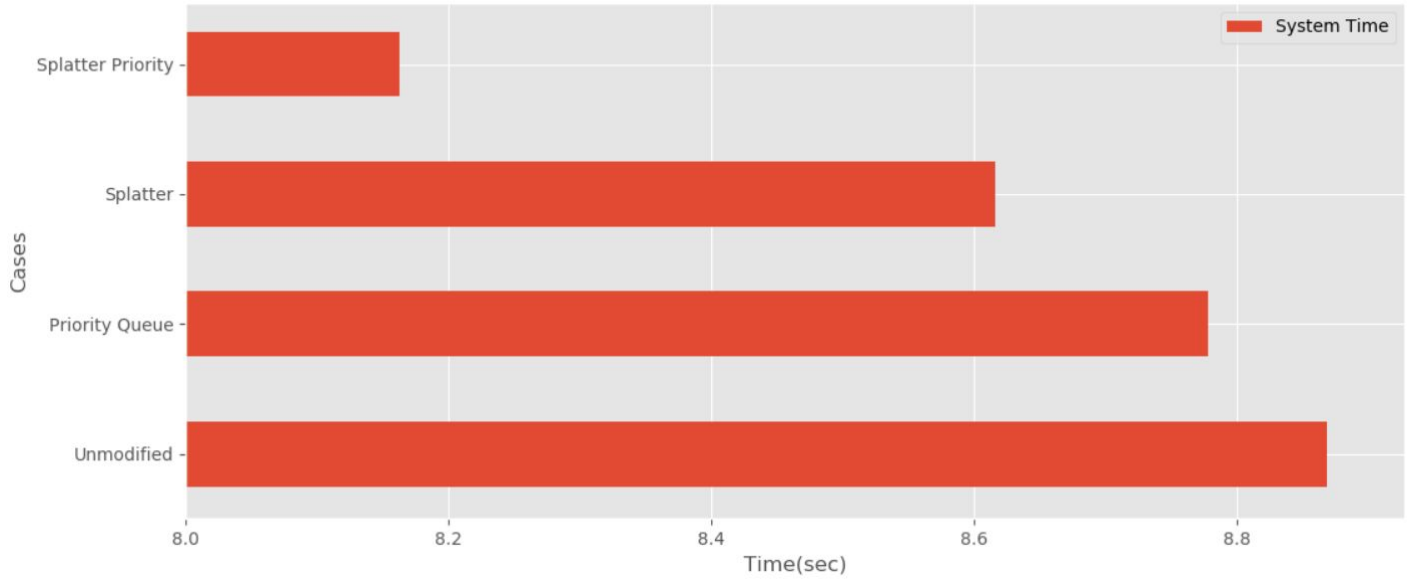
- Involuntary Context Switches: 4799

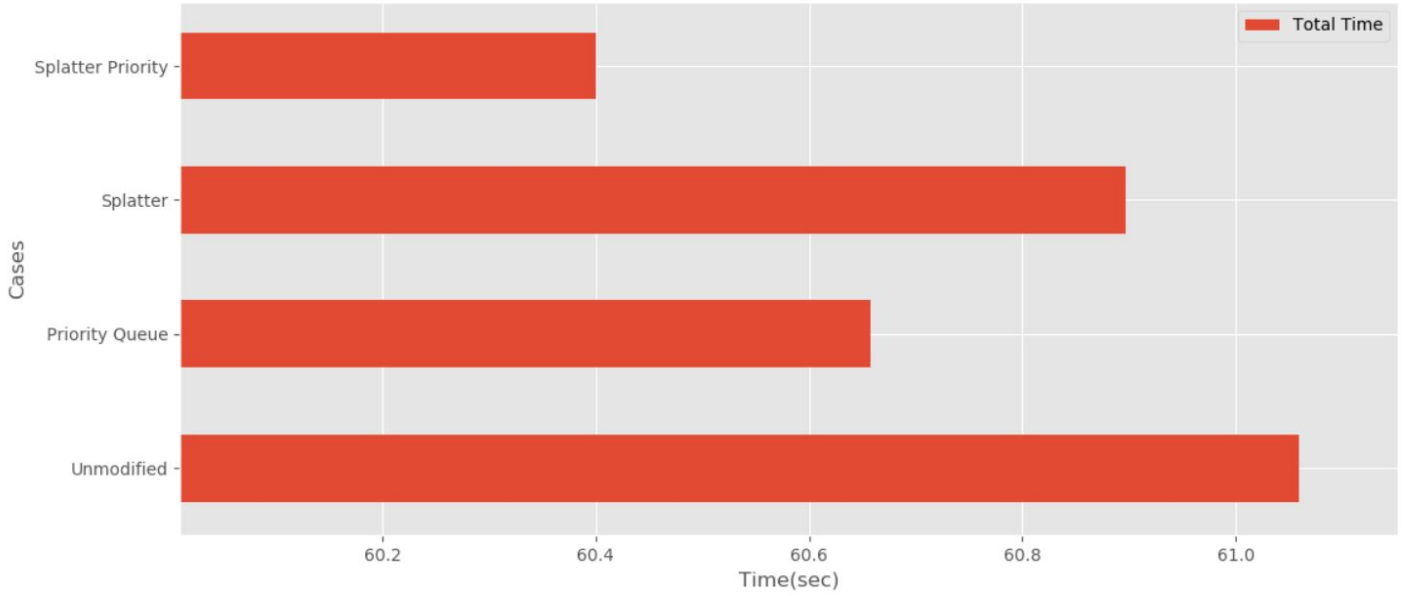
Case 4:

- Runtime:
    - User: 52.237s
    - System: 8.163s
- Page faults:
    - Soft: 303
    - Hard: 0
- Voluntary Context Switches: 37
- Involuntary Context Switches: 4638

VISUALIZATION




System Time
Splatter Priority
Splatter
Priority Queue
Unmodified
Cases
8.0
8.2
8.4
8.6
8.8
Time(sec)


Total Time
Splatter Priority
Splatter
Priority Queue
Unmodified
Cases
60.2
60.4
60.6
60.8
61.0
Time(sec)