Design Document: FreeBSD Scheduling

Nithin Ramabathiran (nramabat@ucsc.edu)
Leon Pham (lpham10@ucsc.edu)
Lior Peled (lpeled@ucsc.edu)
Daniel Truong (daktruon@ucsc.edu)

1 Objective

The objective is to modify the current FreeBSD scheduler to implement priority queues and splatter scheduling instead of using its current timeshare scheduler. In addition to modifying the scheduler, benchmarks will be performed to analyze the performance statistics by stressing the scheduler with a program to fork many processes and sort a list.

2 Assumptions

Instead of using the nice() to change the priorities of user threads, we manually set the priority variable "*int pri*" which we assumed does the same thing. For splatter scheduling we set it randomly (based on the correct ranges) and for priority queues we used it to find the highest priority value.

3 Design

The changes to the scheduling algorithm were made in the **runq_add()** and **runq_choose()** functions within the **kern_switch.c** file. The runq_choose() function was changed to implement priority queues to organize threads for scheduling. The runq_add() function was changed to implement splatter_scheduling.

Priority Queue Scheduling

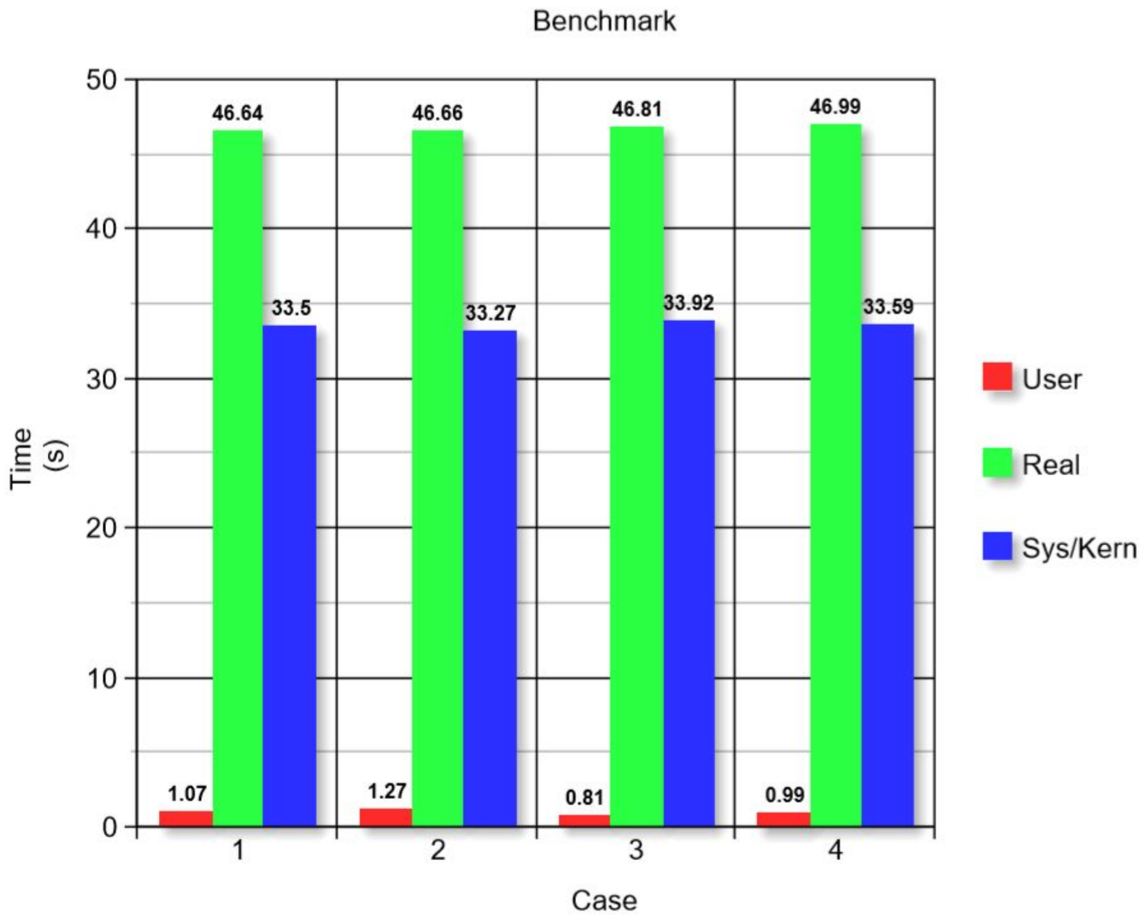
Normally with FIFO scheduling, it just takes the first process in the run queue. We implemented Priority queues by looping through all of the processes to find the one with highest priority, and selecting that highest priority thread.

Splatter Scheduling

We implemented splatter scheduling by generating a random number within the ranges 48- 79, 120 - 223, or 224-255 for REALTIME, TIMESHARE, and IDLE thread types, which are the user threads, respectively. We assigned the random number (divided by 4 to fit into 64 queues) to user threads and then put them into that queue.

4 Benchmarks

The benchmark is performed by running a program which forks many processes to sort a list of numbers. Using **SYSCTL** we added functionality to dynamically switch between the different schedulers. After testing the changes, there was not a significant change in performance between the different methods of scheduling. Scheduling with priority queues, splatter scheduling, priority queues with splatter scheduling all produced similar results.


Benchmark
Time (s)
Case
User
Real
Sys/Kern