Dr. Ribeiro
OS Concepts
24 April 2018

Implement Scheduler Assignment

Design:

Our scheduler implements a shortest job first type of scheduler. However, due to the inability to properly estimate the run time of each process, the scheduler uses a random number generator to “simulate” their durations. The scheduler then takes the shortest jobs and runs them first.

While round robin type schedulers are very fair, each job gets an equal chance to run, they are not very fast, a fairly large portion of time is spent on context switching. SJF has a much greater turnaround time than round robin, but there is a chance a very long job will never get a chance to run. To highlight this difference, a set of jobs with a wide range of completion times can be submitted to the scheduler. The round robin scheduler will happily process each and every job, while the SJF scheduler, if shorter jobs are continuously added to the job queue, will never schedule the longer jobs.

Since our scheduler is, in actuality, Monkey Scheduler, however, under no circumstances will our scheduler reliably perform better than round robin. This is due to inefficiencies of our SJF policy using a random number assignment. Therefore, the results above for ours are longer than the round robin in every test that it was able to function properly for.

Our scheduler could be improved by fully implementing SJF with an algorithm for predicting the burst times of the queued jobs. In its current state, jobs are not scheduled against any metric other than the seed fed to the random number generator, which does not (frequently) lead to rapid job processing; all outcomes are accidental.

Implementation:

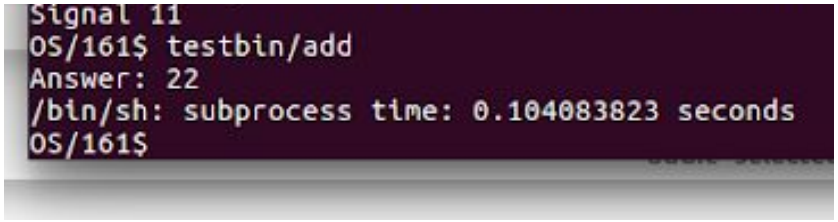
In order to implement the Shortest Job First type scheduler, the estimated burst time of a job must be stored, and there must be a way to sort the job queue based upon that estimated burst time. The estimated time of a thread is stored within it’s queue node; we modified the `threadlistnode` data structure to include an integer variable that holds a randomly generated estimated burst time. For simplicity, this variable is a randomly generated unsigned number modulo 20.

We then added a function to the `threadlist` data structure, `threadlist_sort`, that sorts the run queue by the value of the nodes’ estimated burst time when a new job is scheduled, thus creating a priority queue. While we have essentially implemented Monkey Scheduler with the random assignment of estimated burst times, it is done in the style of a Shortest Job First scheduler through the use of a priority queue to determine the execution order of the jobs.

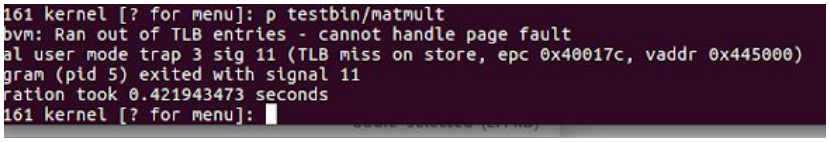
Benchmark:

Default Round Robin:

Add:

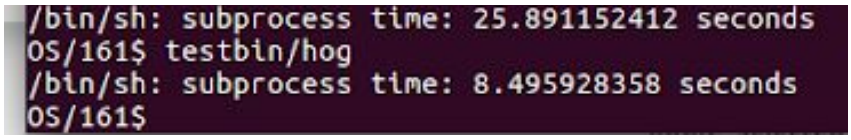


Matmult:

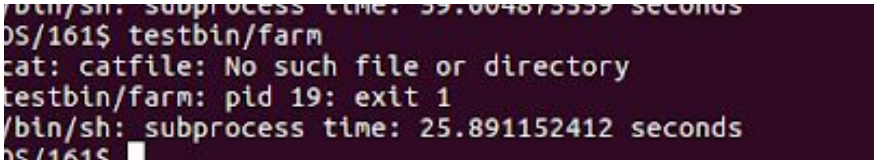


VM didn't have enough memory available to test

Hog:



Farm:



Schedpong:

Implemented Scheduler:

Add:

```
OS/161 kernel [? for menu]: p testbin/add
Answer: 22
Program (pid 2) exited with status 0
Operation took 0.160995087 seconds
```

Matmult:

```
OS/161 kernel [? for menu]: p testbin/matmult
dumbvm: Ran out of TLB entries - cannot handle page fault
Fatal user mode trap 3 sig 11 (TLB miss on store, epc 0x40017c, vaddr 0x44500
0)
Program (pid 2) exited with signal 11
Operation took 0.547820224 seconds
```

Hog:

```
OS/161 kernel [? for menu]: p testbin/hog
Program (pid 2) exited with status 0
Operation took 8.550940873 seconds
```

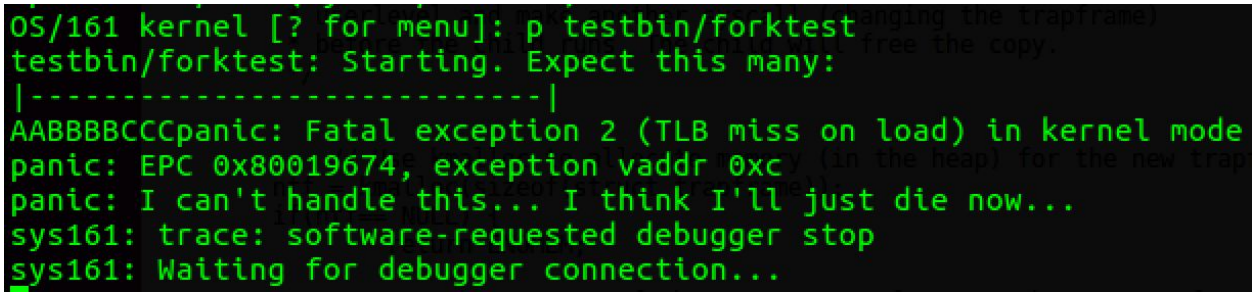
Farm:

```
OS/161 kernel [? for menu]: p testbin/farm
panic: Fatal exception 2 (TLB miss on load) in kernel mode
panic: EPC 0x80019674, exception vaddr 0xc
panic: I can't handle this... I think I'll just die now...
sys161: trace: software-requested debugger stop
sys161: Waiting for debugger connection...
```

SchedPong:

```
OS/161 kernel [? for menu]: p testbin/schedpong
Running with 2 thinkers, 0 grinders, and 1 pong groups of size 6 each.
panic: Fatal exception 2 (TLB miss on load) in kernel mode
panic: EPC 0x80019674, exception vaddr 0xc
panic: I can't handle this... I think I'll just die now...
sys161: trace: software-requested debugger stop
sys161: Waiting for debugger connection...
```

Forktest:



While “add” and “hog” were able to be properly run, “farm” and “schedPong” were not. After some debugging, it was discovered that our scheduler was able to properly schedule and run only a few jobs before breaking. This phenomenon can be seen in the partial execution of ‘testbin/forktest’; some of the child processes are able to finish before the scheduler breaks. Since “add” and “hog” are single processes, they run without issue. “Farm” and “schedPong”, however, spawn several children that must then be scheduled and run.