Machine Problem 3 – CPU scheduling

Deadline:2018/12/16 23:59

I. Goal

The default CPU scheduling algorithm of Nachos is a simple round-robin scheduler with 500 ticks time quantum. The goal of this MP is to replace it with a **multilevel feedback queue.**

II. Assignment

1. Scheduling policy descriptions:
   (a). There are **3 levels of queues**: L1, L2 and L3. L1 is the highest level queue, and L3 is the lowest level queue.
   (b). All processes must have a valid **scheduling priority between 0 to 149**. Higher value means higher priority. So 149 is the highest priority, and 0 is the lowest priority.
   (c). A process with priority between **0~49 is in L3** queue. A process with priority between **50~99 is in L2** queue. A process with priority between **100~149 is in L1** queue.
   (d). **L1 queue uses preemptive SJF**(shortest job first) scheduling algorithm. The job execution time is approximated using the equation: $t(i) = 0.5 \cdot T + 0.5 \cdot t(i-1)$
   (e). **L2 queue uses non-preemptive priority** scheduling algorithm.
   (f). **L3 queue uses round-robin** scheduling algorithm with time quantum **100 ticks**.
   (g). An **aging mechanism** must be implemented, so that the priority of a process is **increased by 10 after waiting for every 1500 ticks**.

2. Scheduling logging information
   (a). Whenever a process is insert into a queue:

Tick [current tick count]: Thread [thread ID] is inserted into queue L[queue level]

   (b). Whenever a process is removed from a queue:

Tick [current tick count]: Thread [thread ID] is removed from queue L[queue level]

   (c). Whenever a process changes its scheduling priority:

Tick [current tick count]: Thread [thread ID] changes its priority from [old value] to [new value]

   (d). Whenever a context switch occurs

Tick [current tick count]: Thread [new thread ID] is now selected for execution

Tick [current tick count]: Thread [prev thread ID] is replaced, and it has executed [tick count] ticks

3. Reminders:
   (a). When a process is interrupted or preempted by others, its **CPU burst time** must stop accumulating. After it resumes running you should restart accumulating its CPU burst time.
   (b). When you select a thread from L1 which has been interrupted or preempted before, **you don't need to recalculate predicted burst time.**

4. Working items:
   (a). **L1 preemptive SJF scheduling algorithm** as described above.
   (b). **L2 non-preemptive priority job scheduling algorithm** as described above.
   (c). **L3 round-robin scheduling algorithm** as described above.
   (d). **An aging mechanism** to move processes among the queues as described above.
   (e). **Output logging information** during your execution as described above.

III. Instructions

1. Copy your code for MP2 to a new folder

   ```
   $ cp -r NachOS-4.0_MP2 NachOS-4.0_MP3
   ```

2. Test your implementation
   - You can run multiple processes and set their **initial priority** by using "**-ep**" argument in NachOS launch process command line.
   - E.g.,: the command below will launch 2 processes: test1 with initial priority 40, and test2 with initial priority 80.

     ```
     $ ../build.linux/nachos -ep test1 40 -ep test2 80
     ```
   - **Create your own test program to verify the correctness. (Remember to modify the Makefile** according the instructions in tutorial slides).

IV. Grading

1. Implementation correctness – 70%
   (a). (25%) L1preemptive SJF scheduling algorithm.
   (b). (15%) L2 non-preemptive priority job scheduling algorithm.
   (c). (15%) L3 round-robin scheduling algorithm.
   (d). (10%) Aging mechanism.
   (e). (5%) Output logging information
2. Report – 15%
   - **Cover page** including team members, Team member contribution.
   - You are not requested to trace any code in this MP, but you must **explain each step** in your implementation in details.
3. Demo – 15%
   - You must prepare one or multiple **test cases by yourself** to **demonstrate the correctness of your implementation for each of the working items**, and explain the correctness of our implementation **based ONLY on the output logging information**.
   - You must demonstrate and prove your correctness in **10 minutes**. (**Exceeding this time limit could result in point deduction!!!**)
   - **One random test case** will be used for correctness verification during the demo.
   - Answer **questions from TAs**.

***Refer to syllabus for late submission penalty.**