DEPARTMENT OF COMPUTER SCIENCE, HANYANG UNIVERSITY

MLFQ and Stride scheduling

Operating Systems: Three Easy Pieces

Programming Project with xv6

Due date: April 23, 2017

1 IMPLEMENTATION SPECIFICATION

1. Original xv6 scheduler (default scheduler in xv6)
   Xv6 uses RR (Round Robin) scheduling as a default scheduling algorithm. When a timer interrupt occurs, a currently running process is switched over to the next runnable process. The time interval between consecutive timer interrupts is called a *tick*. The default value set for the tick in xv6 is about 10ms.

2. MLFQ (Multi-level feedback queue) scheduling
   - MLFQ consists of three queues with each queue applying the round robin scheduling.
   - The scheduler chooses a next ready process from MLFQ. If any process is found in the higher priority queue, a process in the lower queue cannot be selected until the upper level queue becomes empty.
   - Each queue has different time quantum.
     - The highest priority queue: 5 ticks
     - Middle priority queue: 10 ticks
     - The lowest priority queue: 20 ticks
   - To prevent starvation, priority boosting needs to be performed periodically.
     - The priority boosting is the only way to move the process upward.
     - Frequency of the priority boosting: 100 ticks

- MLFQ should always occupy at least 20% of CPU share.
- We assume that there is no *gaming the scheduler* situation.

3. Stride scheduling
   - If a process wants to get a certain amount of CPU share, then it invokes a new system call to set the amount of CPU share.
   - When a process is newly created, it initially enters MLFQ. The process will be managed by the stride scheduler only if the set_cpu_share() system call has been invoked.
   - The total sum of CPU share requested from processes in the stride queue can not exceed 80% of CPU time. Exception handling needs to be properly implemented to handle oversubscribed requests.

4. Required system calls
   Following system calls should be newly implemented for this project, and TAs will assume all these system calls are implemented in your xv6 kernel when testing your kernel:
   - yield: yield the cpu to the next process
     - int sys_yield(void)
   - getlev: get the level of current process ready queue of MLFQ. Returns one of the index of MLFQ (0/1/2)
     - int sys_getlev(void)
   - set_cpu_share: inquires to obtain cpu share(%).
     - int sys_set_cpu_share(void) - wrapper
     - int set_cpu_share(int)

5. Scheduling scenario

Event	MLFQ CPU %	Stride CPU %	Total CPU %
Create process1	100	0	100
Create process2	100	0	100
Process2.set_cpu_share(20)	80	20	100
Create process3	80	20	100
Process3.set_cpu_share(50)	30	70	100
Process1.set_cpu_share(30) **- FAIL**	30	70	100
Process1.exit	30	70	100
Process2.exit	50	50	100
Process3.exit	100	0	100