CS 202: Advanced Operating Systems
University of California, Riverside

Lab #2: Modifying xv6 Scheduler

Due: 11/10/2021, Wednesday, 11:59 p.m. (Pacific time)

In this assignment, you will implement the lottery scheduler and the stride scheduler. The goal is to implement only basic operations of these schedulers (i.e., no ticket transfers, compensation tickets, etc). It is up to you to design the interface to assign the number of tickets, but it should be sufficient to illustrate that the schedulers work correctly. Assume a single CPU core for xv6.

To get started, look at the file "kernel/proc.c". In there you will find a simple round robin CPU scheduler implemented. You should change the logic there to use lottery and stride scheduling.

After you complete the schedulers, here is how you can demo your work for each one separately.

First, write a system call, `sched_statistics()`, which prints out the number of times each process in the system has been scheduled to run (i.e., a rough estimation of the number of ticks used by each process).

Next, run the following user-level program with 3 different ticket values:

```
#include "kernel/types.h"
#include "kernel/stat.h"
#include "user/user.h"
int main(int argc, char *argv[])
{
    FUNCTION_SETS_NUMBER_OF_TICKETS(30);    // write your own function here
    int i,k;
    const int loop=100000; // adjust this parameter depending on your system speed
    for(i=0;i<loop;i++)
    {
        asm("nop");  // to prevent the compiler from optimizing the for-loop
        for(k=0;k<loop;k++)
        {
           asm("nop");
        }
    }
    sched_statistics(); // your syscall
    exit(0);
}
```

Output should look like this:

For example: The program prog1.c has 30 tickets, add prog2.c with 20 tickets and prog3.c with 10 tickets. Run all of them simultaneously:

Input

$ prog1&;prog2&;prog3

Run 3 test programs, each with a different number of tickets. Whenever one of these processes finishes execution, your system call should print the number of times these processes have been scheduled to run. The first output of your syscall (the one from the earliest finished process) is what you need to focus on.

Output: (first syscall output; progs may be printed in different order)

Ticks value of each program

prog1 : xxx

prog2 : xxx

prog3 : xxx

Use the number of ticks to calculate the allocated ratio per process: (number of ticks per program) / (total number of ticks by all 3 programs)

Approximately prog1 ratio will be 1/2, prog2 ratio will be 1/3 and prog3 will be 1/6

Compare your implementations of lottery scheduler and stride scheduler. In case of the stride scheduler, the stride values are inversely proportional to the number of tickets provided.

See Section 5 of the stride scheduling paper (http://dl.acm.org/citation.cfm?id=889650 or http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TM-528.pdf) on how to compare and evaluate these two schedulers. Try to reproduce Figure 8 and Figure 9 for your implementations.

Notes:

xv6 doesn't have a random generator. You would need your own pseudo-random generator to implement lottery scheduling.

Stride scheduling in this lab refers to the "Basic" one in the paper (Sec 2.1). No need to implement the "Dynamic" and "Hierarchical" stride scheduling algorithms.

How to compile:

Use preprocessor directives to switch between lottery and stride schedulers. For example:

```
    ......xv6 code......
```

```
#ifdef LOTTERY
your code for lottery scheduler
#endif
#ifdef STRIDE
Your code for strider scheduler
#endif
......xv6 code......
```

You need to modify the Makefile to let you can toggle it on/off in command line:

After lines 72-73 "`CFLAGS = -fno-pie ...... endif`", insert 2 new lines:

```
LAB2 = LOTTERY
CFLAGS += -D$(LAB2)
```

So your code should be compiled for lottery scheduler by "`make clean; make LAB2=LOTTERY`" and for stride scheduler by "`make clean; make LAB2=STRIDE`".

What to submit:

You need to submit the following:

(1) The **entire XV6 source code** with your modifications ('make clean' to reduce the size before submission)

(2) **A report (in PDF)** with detailed explanation what changes you have made, the comparison figures, and how you produce the figures.

(3) A **demo video** showing that all the functionalities you implemented can work as expected, as if you were demonstrating your work in-person. Demonstrate both lottery and stride schedulers.

Grades breakdown:

- Working lottery scheduler (with clear explanation): 40 pts
- Working strider scheduler (with clear explanation): 40 pts
- Comparison figures and clear explanation of how to produce them: 20 pts

Total: 100 pts