Lab 1

There are two parts to this assignment: (1) Add a system call to xv6, and (2) Change the scheduler to use stride scheduling.

Part 1 - System Call Implementation:

Add a new system call info(int param) that takes one integer parameter with value 1, 2 or 3. Depending on the value, it returns:

(1) A count of the processes in the system;

(2) A count of the total number of system calls that a process has done so far;

(3) The number of memory pages the current process is using.

Part 2 - Modifying xv6 scheduler:

In this part, you will implement stride scheduling. Implement only basic operation (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 scheduler works.

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

After you complete the scheduler, here is how you can demo your work.

Run the following program with for 3 different ticket values:

```
#include "types.h"
#include "stat.h"
#include "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=43000;
      for(i=0;i<loop;i++)
      {
            asm(nop);       //in order to prevent the compiler from optimizing the for loop
            for(k=0;k<loop;k++)
            {
                  asm(nop);
            }
      }
      exit();
}
```

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

Assuming that you run 3 test programs, each with a different number of tickets, when each of these programs finishes execution, your system call should print the number of times the processes was scheduled to run (number of ticks).

Output: (not necessarily in the same)

Ticks value of each program

prog1 : xxx

prog2 : xxx

prog3 : xxx

Use the number of tickets 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

Use the same program and run it using three different number of tickets for example (100,50,250) then the stride values are inversely proportional to the number of tickets strides (100,200,40). Results of dividing 10,000 by the number of tickets for each process (no need for dynamic total ticket value).

Show that over time:

prog3 will run 5 times more than prog2 & about 2.5 times more than prog1.

prog1 will run about 2 times more than prog 2.

To do a more complete evaluation of the schedulers, tracking performance metrics such as response time and fairness over a number of workloads. Refer the papers for more ideas.

What to submit:

You need to submit the following:

(1) A zip file of the entire XV6 source code you modified

(2) A PDF document with detailed explanation what changes you have made and necessary screenshots to show your work and results.

Grades breakdown:

info() system call: 45%

- count of the processes in the system: 15%
- report the number of system calls this program has invoked: 15%
- report the number of memory pages the current process is using: 15%

Modification the xv6 scheduler: 55%

- Stride scheduler: 40%
- more complete evaluation of the schedulers: 15%

Total: 100%