OPERATNG SYSTEMS LAB

XV6

Assignment

Lottery scheduling

Lottery scheduling is a probabilistic scheduling algorithm for processes in an operating system. Processes are each assigned some number of lottery tickets, and the scheduler draws a random ticket to select the next process. The distribution of tickets need not be uniform; granting a process more tickets provides it a relatively higher chance of selection. This technique can be used to approximate other scheduling algorithms, such as Shortest job next and Fair-share scheduling.

Lottery scheduling solves the problem of starvation. Giving each process at least one lottery ticket guarantees that it has a non-zero probability of being selected at each scheduling operation.

Lottery scheduling : example

1. We have two processes A and B. A has 60 tickets (ticket number 1 to 60) and B have 40 tickets (ticket no. 61 to 100).
2. Scheduler picks a random number from 1 to 100. If the picked no. is from 1 to 60 then A is executed otherwise B is executed.

Steps

You need to do the following things to implement lottery scheduling in XV6

1. Make a **system call** which allows you to set the tickets for a process.
2. Code to **generate a random number**.
3. In the scheduler function count the **total number of tickets** for all processes that are **runnable.**
4. Generate a **random number between 0 and the total tickets** calculated above.

5. When looping through the processes keep the **counter of the total number of tickets passed**.
6. Just when the counter becomes greater the random value we got, run the process.
7. Put a break at the end of for loop so that we don't execute the processes following the process we just run.

Allocate default value of tickets when a process is created

In the proc.c file, enter the following line of code in "**allocproc**" function just below "**found**".

```
p->tickets = 10;
```

This sets the default ticket value to 10.

Implementing The Scheduler

- For lottery scheduling, we need the **total number of tickets of the processes that are in the RUNNABLE state**. To calculate this we put a for loop inside the outer for loop just **before** the inside for loop executes. This calculates the total number of tickets.
- Now generate a random number between 0 and the total number of tickets. After getting the random number execute the for loop that runs processes.
- When this for loop loops over processes, keep counting the total number of tickets passed of processed that are in a RUNNABLE state. As soon as the total tickets passed get higher than the random number, run that process.
- Now, after the process runs, we need to put a break at the end of for loop which executes all processes. This is because:
  If we don't break the value of total tickets passed will keep on increasing and will always be higher than the random number.
- Also, after we run a winning process we need to recompute the total number of tickets of all RUNNABLE processes as that value might have changed.