Assignment 2 - Design and Implementation of Lottery Scheduler for FreeBSD System

Assignment By: Ahmed SalahEldin Farouk Elkashef 1410216

General Overview:

This assignment aims to implement lottery scheduling in FreeBSD that assigns each process's threads some number of tickets (500), then randomly draws a ticket among those allocated to "ready" threads to decide which one to run. That thread is then allowed to run for a set time quantum, during which I/o may occur, or after which it is interrupted by a timer interrupt and this whole procedure is repeated. The number of tickets assigned to each thread determines the likelihood that it'll run at each scheduling decision, and thus (over the long term) the relative amount of time that it gets to run. Threads that are more likely to get chosen each time will get chosen more often, and thus will get more CPU time.

Steps to achieve lottery scheduling:

- Change the mechanism that adds new threads to run queues so that they get added to new runqs in order to get chosen and scheduled randomly, this mechanism only applies for user processes.

- Change the way a thread is chosen, by producing a new array of random numbers with the size of 10 numbers for each runq, those random numbers are updated every 10 scheduling decisions. Thus, we introduced a new variable "num_sched_decisions" that counts each time a scheduler successfully decides.

- Add a gift() system call, that takes a target process, and a number of tickets. Deducts those tickets from the calling process and donates them to the target process. This system call only works when gifting tickets is applicable and does not contradict with any of the restrictions of ticket transfer.

- add three new runqs for the user processes (interactive, timsharing, indle) that will be dealt with using the lottery scheduling

- Used the arc4random() pseudo-random number generator, the function fills up the pool of random number 10 times and gets restarted again to fill new 10 random numbers and replace the existing ones. For more elaboration:

https://www.freebsd.org/cgi/man.cgi?query=arc4random&sektion=3

Adding to our new run queues - Responsible function: tdq_runq_add:

The function is split into two sections that handles the process if it's a root process or a user process

If root process: it acts the same way

If user process: it adds gives each thread 500 tickets, sits the flag "ticketed" to true, and decides if it's a timeshare, interactive, or idle. Finally, it adds them to our newly created ones.

Choosing from our new run queues - Responsible function: tdq_choose:

This function utilizes new functions inside of it that do the job:

tdq_rand: this function populates the pool of random numbers that will be later used to decide which thread to be chosen.

runq_choose_from_lottery: this function takes that random number from tdq_rand, loops through the 64 insider queues of each runq, subtracts the number of tickets from each thread of those insider queues from that random number, and when the random number finally reaches zero, it picks that thread. Using this method, we can guarantee that the chosen thread is most probably chosen in a random way.

Adding a gift system call - Responsible function: sys_gift.c, syscalls.master, Symbol.map, sys_pipe.c:

Register the new system call in syscalls.master & Symbol.map & sys_pipe.c, then create the function itself in the file sys_gift.c

General restrictions on Ticket transfers and manipulation in Gift(PID,tickets) & nice(tickets):

- No thread should have more than 100,000 tickets, and no less than 1 ticket

- determine if a thread should have its tickets increased or decreased depending on its "nice" value. if it's negative, we increase the tickets, if positive, we decrease the tickets.

- if gift(0,0) is called, we return the number of tickets for the calling process