SCHEDULING ALGORITHMS PERFORMANCE REPORT

In this report, we have evaluated the performance of the Stride Scheuler, Lottery Scheduler, and XV6 default scheduler. we have performed the fundamental analysis using the uptime() to calculate the turnaround time of the process and the average turnaround time on four different tasks: stress, uniq, find, cat README | uniq. To test this we have developed a test program in which we have created four child processes using fork system call and then replaced the first child process with stressfs using exec system call, similarly replaced the second child process with uniq, third child process with find and fourth child process with cat README.

In these, we have measured the **turnaround time** of the processes. The units were ticks.

In the stride scheduler, the Constant is 100000

Default Scheduler vs Stride Scheduler

Task	**Default Scheduler**	**Stride Scheduler**	**Tickets assigned in Stride scheduler**
stressfs	628 ticks	773 ticks	40000
uniq	225 ticks	740 ticks	2
find	60 ticks	106 ticks	2
Cat README | uniq	490 ticks	739 ticks	2

Default Scheduler vs Lottery Scheduler

For testing, in lottery scheduling also a same number of tickets were assigned to child processes as in stride scheduling.

Default Scheduler vs Lottery Scheduler

Task	**Default Scheduler**	**Lottery Scheduler**	**Tickets assigned in Lottery scheduler**
stressfs	628 ticks	909 ticks	40000
uniq	225 ticks	909 ticks	2
find	60 ticks	862 ticks	2
Cat README | uniq	490 ticks	909 ticks	2

Stride Scheduler vs Lottery Scheduler

In lottery vs stride scheduling 40000 tickets were assigned to stressfs, 2 tickets were assigned to uniq, 2 tickets were assigned to find, 3 tickets were assigned to cat.

Stride Scheduler vs Lottery Scheduler

Task	**Stride Scheduler**	**Lottery Scheduler**	**Tickets Assigned**
stressfs	532 ticks	719 ticks	40000
uniq	512 ticks	65 ticks	2
find	532 ticks	683 ticks	2
Cat README | uniq	516 ticks	711 ticks	3

From the observations, we can see that different schedulers allocate different amounts of CPU time to different processes.

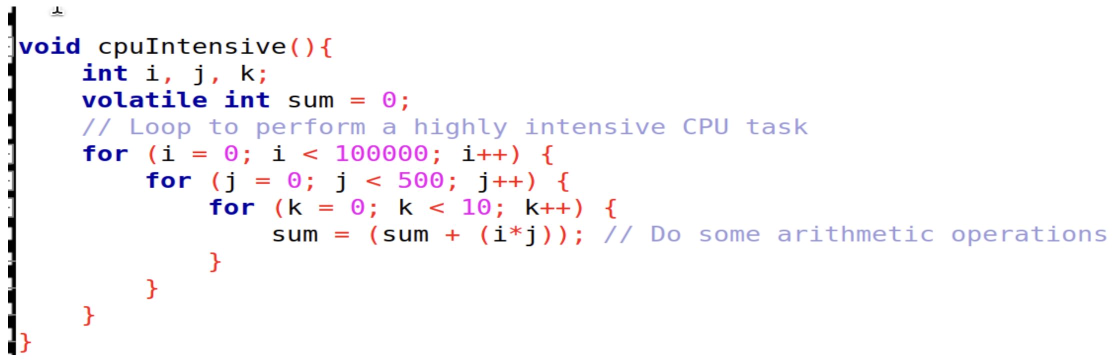
Stride scheduler and lottery scheduler allocate CPU time based on the number of tickets assigned to each process. The stride scheduler uses a fixed number of tickets, whereas the lottery scheduler uses a random winning ticket number and selects a process with that ticket to allocate CPU. As a result, in table 3 lottery scheduler might have picked the winning ticket number of uniq more times as a result it has completed quickly compared to the stride scheduler.

In the default scheduler, the allocation of CPU time is not based on tickets or any other scheduling algorithm. It is determined by the order in which the processes are created and scheduled to run and each process works in a round-robin fashion with a time slice of 1 tick.

But when these commands run sequentially then there will be a subtle difference in the turnaround time of the process as they will be scheduled one after another and one process will be running in CPU at an instant of time. Hence, irrespective of the scheduler almost similar turnaround times can be observed when using different schedulers.

We have developed a test program in which we have created 4 child processes and each process will execute CPU-intensive tasks with the same workload. Now, when the scheduler has changed the difference in turnaround time of each child is observed.

Each child is executing the below CPU-intensive task.



```
void cpuIntensive(){
    int i, j, k;
    volatile int sum = 0;
    // Loop to perform a highly intensive CPU task
    for (i = 0; i < 100000; i++) {
        for (j = 0; j < 500; j++) {
            for (k = 0; k < 10; k++) {
                sum = (sum + (i*j)); // Do some arithmetic operations
            }
        }
    }
}
```

Below are the metrics observed when running the task with different schedulers.

Default Scheduler vs Stride Scheduler

In stride scheduler the Constant is 100000 and the default tickets for each process when it was created is 80. In this case, we have set the child 1 tickets to 2, child 2 tickets to 2500, child 3 tickets to 200 and child 4 tickets to 1500.

Default Scheduler vs Stride Scheduler

Task	**Default Scheduler**	**Stride Scheduler**	**Tickets in Stride scheduler**
Child 1	786 ticks	1039 ticks	2
Child 2	812 ticks	466 ticks	2500
Child 3	789 ticks	882 ticks	200
Child 4	768 ticks	514 ticks	1500

Default Scheduler vs Lottery Scheduler

For testing, in lottery scheduling also a same number of tickets were assigned to child processes as in stride scheduling.

Default Scheduler vs Lottery Scheduler

Task	**Default Scheduler**	**Lottery Scheduler**	**Tickets in Lottery scheduler**
Child 1	786 ticks	1195 ticks	2
Child 2	812 ticks	581 ticks	2500
Child 3	789 ticks	1026 ticks	200
Child 4	768 ticks	612 ticks	1500

Stride Scheduler vs Lottery Scheduler

In lottery vs stride scheduling 200 tickets were assigned to child 1, 10 tickets were assigned to child 2, 3000 tickets were assigned to child 3, 3 tickets were assigned to child 4.

Stride Scheduler vs Lottery Scheduler

Task	**Stride Scheduler**	**Lottery Scheduler**	**Tickets Assigned**
Child 1	498 ticks	541 ticks	200
Child 2	916 ticks	917 ticks	10
Child 3	475 ticks	454 ticks	3000
Child 4	938 ticks	1035 ticks	3

From the above observations it can be seen that in default scheduler all processes took the almost same time to complete as all were running same program and default scheduler works in round-robin fashion with a time slice of 1 tick. When using the Stride scheduler it can be seen that child 2 got completed quickly as more tickets were assigned to it and Child 4 got completed later with little difference in time. As xv6 is running on two cores, the Child 4 also got scheduled parallely with Child 2. Hence, we observed a little difference in turnaround time.

Similarly, in the Lottery scheduler Child 2 and Child 4 completed almost at the same time as they were assigned a higher number of tickets.