# Report For Staircase Scheduler

YUKUI YE
SUID: 439644268

## 1: My comment on the implemented staircase scheduler:

When a process come into the staircase field, its priority equal to its minPriority, after it reached the maxTimeslice, its priority increased by 1, but before it reaches the bottom of the staircase, each descend down will only cost 1 timeslice which is smaller than the dynamic changed timeslice in the top level, And it will move back to one level below its previous priority, and minPriority increased by1 after falling off the bottom of the staircase, another special thing is when process minPriority reaches the bottom of the staircase, it will competing itself with all the process at this level, but will increase its maximum time slice each time. It is kind of amazing to see that this situation lead to a never expired array. And it gives priority to interactive processes and seldom starve any processes. It maximizes overall CPU utilization as well as maximizing interactive performance.

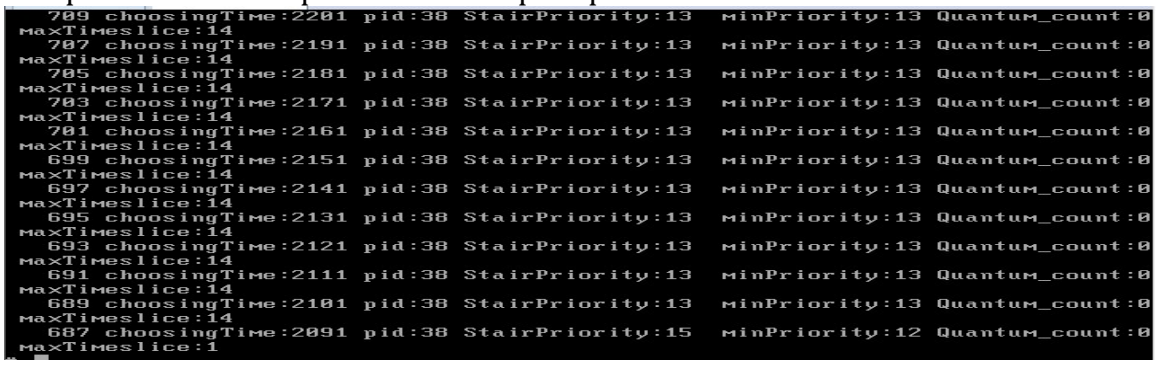
## 2: Comparison between the existing 4.4BSD scheduler and the staircase scheduler:

(1)staircase scheduler has dynamic maximum time slice while 4.4BSD has a fixed time slice based on 10 system ticks calculated by every 4seconds;

(2)The way of calculating priority :In 4.4BSD priority is always changing based on cpu utilization , and there has a certain formula to calculate its priority. After it spent out its time slice but have not done , it will recalculate its priority and insert it into the tail of relative run queue. While , for Staircase scheduler, the priority is the variable we declared , and it increased only by 1 each time.

(3)4.4BSD has property that preemptive multitasking with dynamic priority adjustment which will lead to potential starvation of low-priority jobs. While staircase scheduler has advantage when all the process reaches its minimum priority which is 15(55 normal), then it keeps this priority never

## 3:Representative samples of KTRdump output:

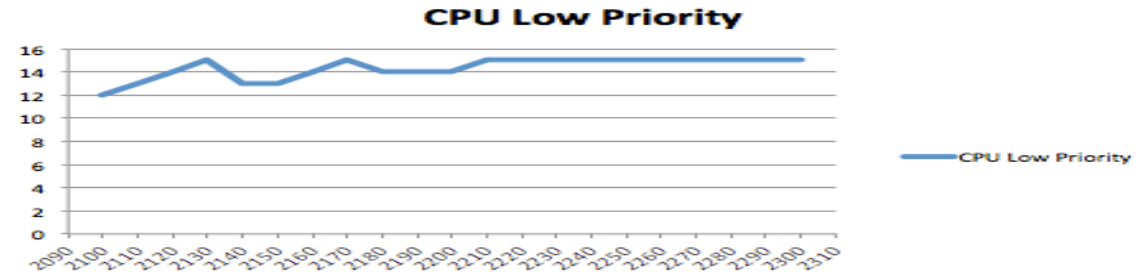



## 4:Graphs(Time VS Priority and Time VS Max. Time slice length) for each type process mentioned in task2:

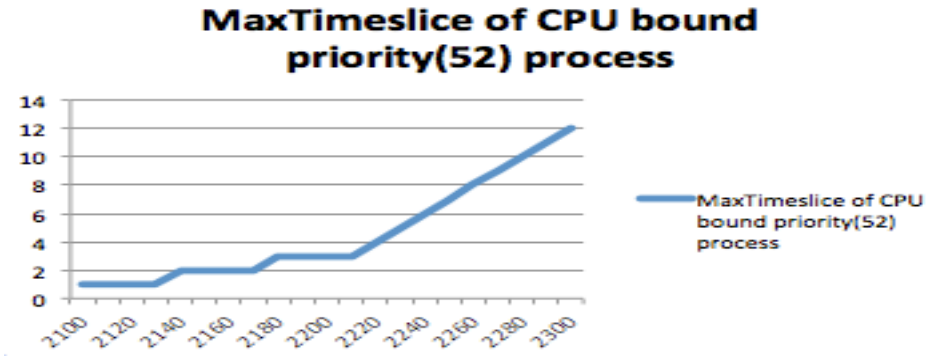
PART1: Time VS Priority:

(1)CPU-BOUND process with worse priority 52(12 in staircase)

*PART1: Time VS Priority:*

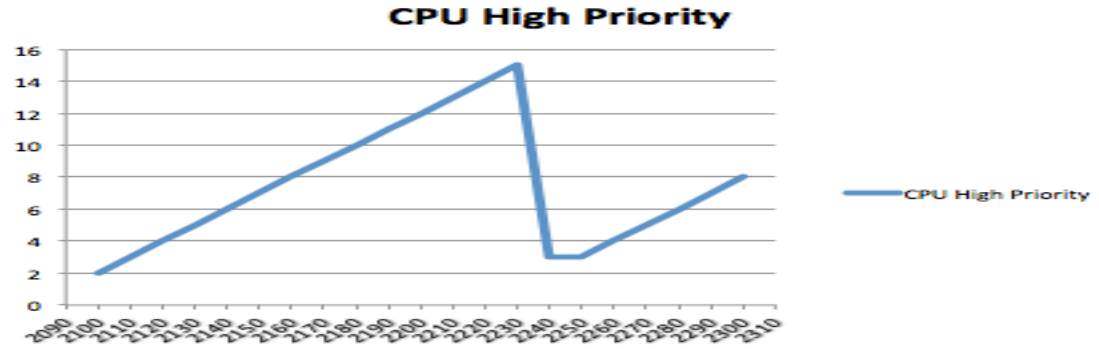



*PART1: Time VS MaxTimeslice:*

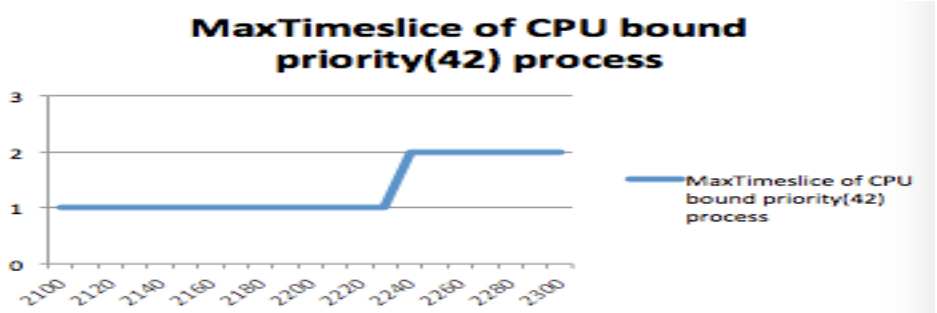



(2)CPU-BOUND process with good priority 42(2in staircase)

*PART1: Time VS Priority:*

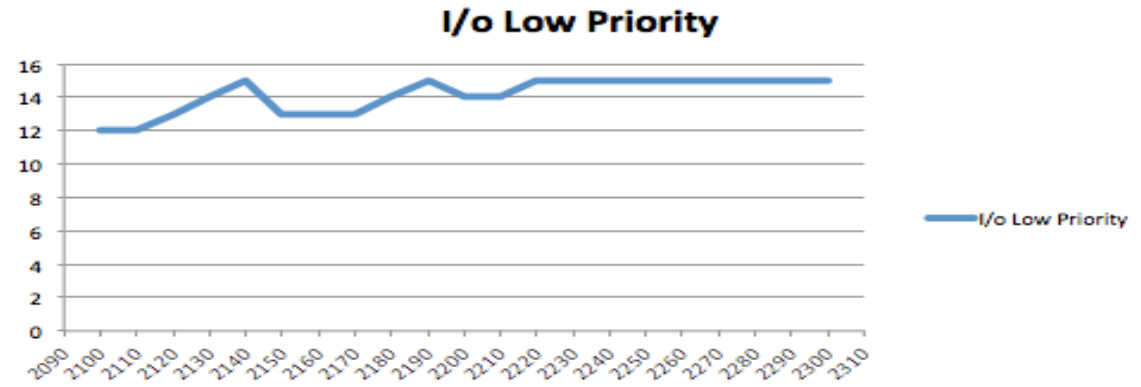



*PART1: Time VS MaxTimeslice:*

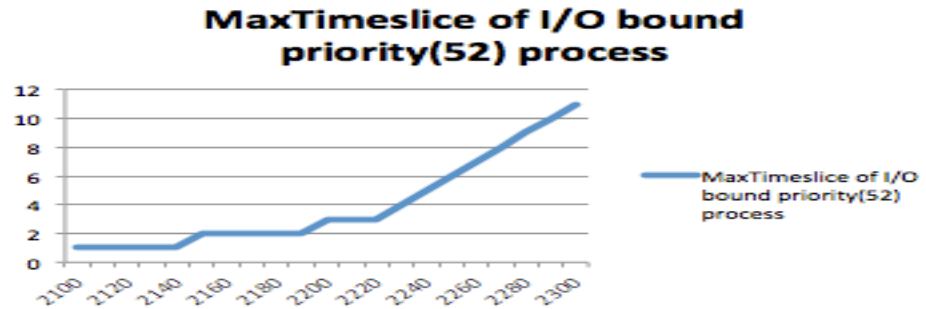



(3)I/O-BOUND process with worse priority 52(12 in staircase)

*PART1: Time VS Priority:*

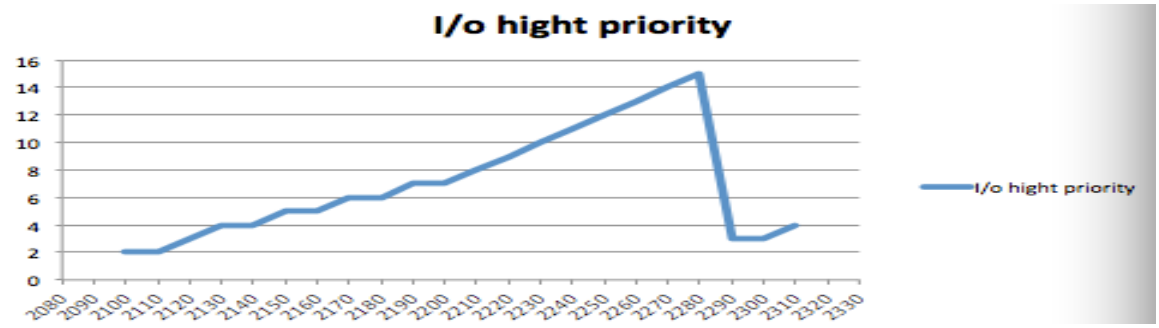



*PART1: Time VS MaxTimeslice:*

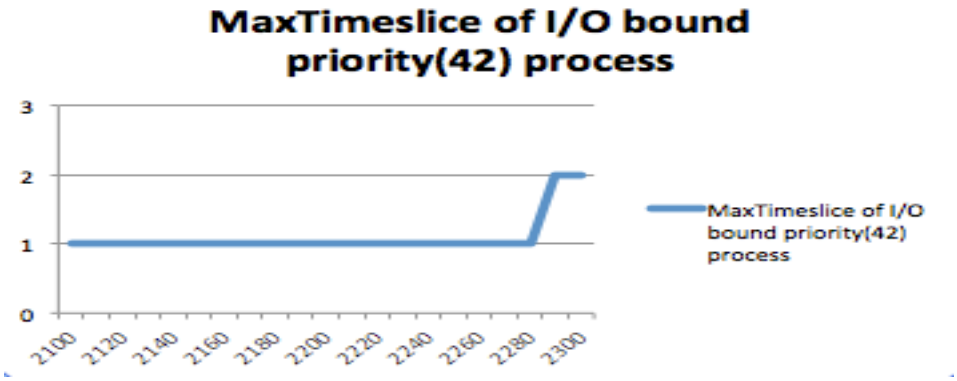



(4)I/O-BOUND process with good priority 42(2 in staircase)

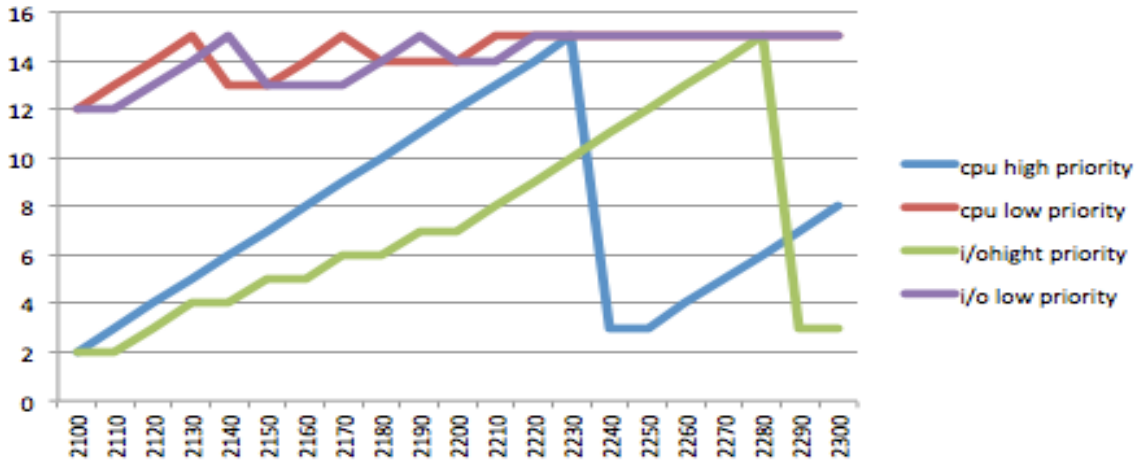
*PART1: Time VS Priority:*




*PART1: Time VS MaxTimeslice:*



5:Analysis on the graphs in the form of starvation and CPU hogging. You also need include your observation of the staircase scheduler for each kind of process.
Before this, I need to clear one confusion about drawing the graph, that is whenever I started the priority from which point let's say started at '12',the start point of the line always from 0, but it will stopped at 12 then do the changes it should be.




Precondition: No preemptive and starvation happened since there is no many processes running in the synchronization multiply processor system.

(1)CPU high priority VS I/O high priority (blue & green lines)

Both of them started at low number priority(high priority),the difference is I/O bound process change slower than CPU bound process, probably due to the interaction between user and computer, it takes time to get user input. The similarity between them is that after they reaches the bottom of the staircase, its priority number dramatically drop from 15 to one level lower than original priority. And it loop one more time slice than previous maximum time slice. That is why they have horizon line at the bottom.

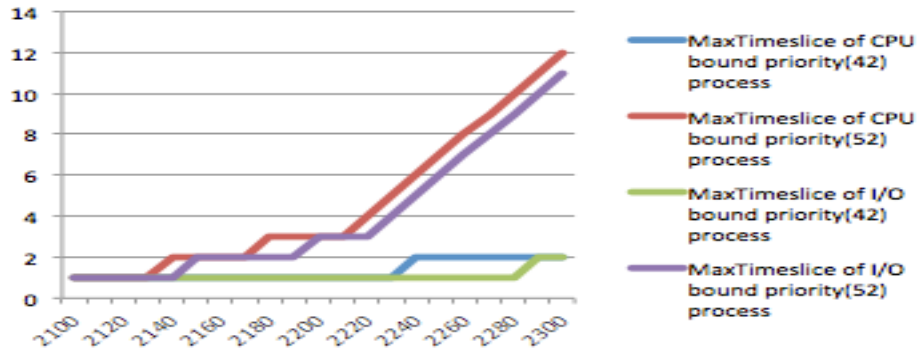
(2)CPU low priority VS I/O low priority(red & purple lines)

Comparing with these two lines, it all started at priority 12(52—worse priority). Similar as previous explanation, cpu bound process change fast than i/o bound process, both of them , when their minPriority reaches the bottom of the staircase, it remains their priority at 15. After they go over from start point priority to the bottom priority of staircase, it jumps over to the next level just behind the previous priority. and increase every time just like before.

(3)process with high priority VSprocess with low priority(blue & red) (green&purple)

high priority with lower number would have dramatically drop down when it reaches the bottom, then jump to one level below the previous minPriority. Low priority with high number do the same thing but when its minPriority reaches the bottom of the staircase, it will remain its priority , never change it, this situation lead to an unexpired array. Besides , from the" maxtimeslice vs time" graph, we can tell that higher number priority's maxtimeslice would continue to increase which is a sharp increasing line.

Time vs MaxTimeslice:




Higher number priority process would have bigger change than lower number priority, since it does not take long time for them to reach the bottom of the staircase and then they come back one level before the minPriority. After their minPriority reaches the bottom of the staircase, we can see from the graph, the maximum time slice continue to increase every 10 ticks. During the time from minPriority to bottom priority, their maximum Time slice remain the same.

(4) After all the process's minPriority reaches the bottom of the staircase, then they will have the same priority, it does not matter which is I/O process or which is CPU bound process, there would have no preemptive situation happened. Besides, I think we do not have process who is in starvation, at least I can not tell from the graphs.