CS423 MP3 Report

Boyin Zhang : bzhang70

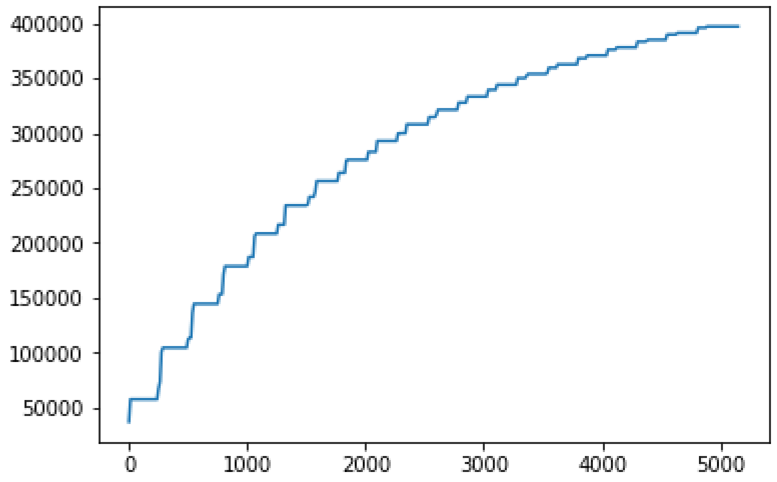
Implementation Details and Design Decisions

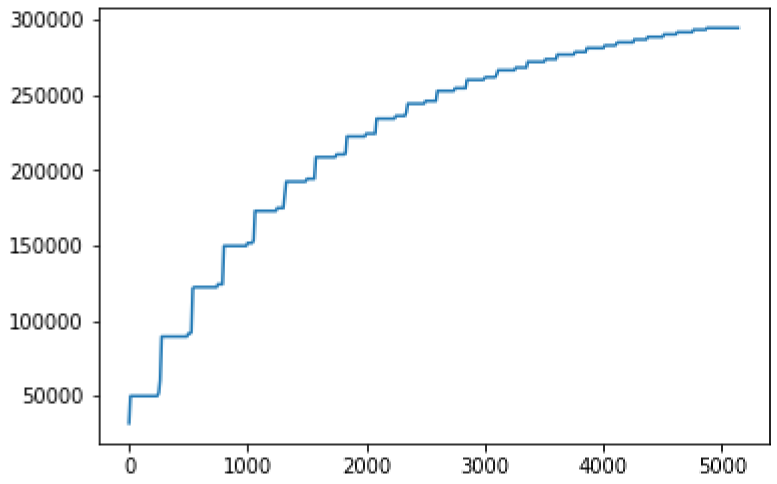
In this MP, I implemented a profiler to measure the major, minor page fault count and CPU utilization of registered user processes. For most of designing part, I just followed the instructions. Specifically, the profiler uses a character device to map the profiler buffer memory allocated in the kernel address space to the virtual address space of a requesting user-level process and uses the mmap operation to map the kernel memory to the address space of a user-level process.

For the implementation parts, I was also following the implementation overview part of the instruction. Having methods like **mp3_read()** and **mp3_write()** for basic /prop/mp3/status read/write, **registration()** and **unregistration()** for the incoming and outcoming processes. **wq_function()** along with **timer_function()**, a delayed work queue to measure and update the major page fault, minor page fault, cpu_time and then write them into the buffer. Finally, I have **cdev_mmap()** using vmalloc_to_pfn(virtual_address) and remap_pfn_range() to map the physical pages of the buffer to the virtual address space of the user process.

Graphs and logical analysis

Case Study 1:
Work process 1: 1024MB Memory, Random Access, and 50,000 accesses per iteration
Work process 2: 1024MB Memory, Random Access, and 10,000 accesses per iteration
Work process 3: 1024MB Memory, Random Locality Access, and 50,000 accesses per iteration
Work process 4: 1024MB Memory, Locality-based Access, and 10,000 accesses per iteration

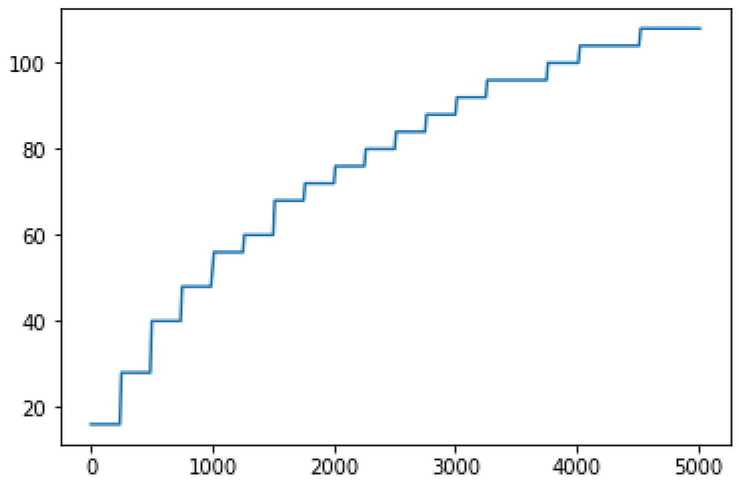




The first graph is the page fault(y-axis) over time(in jiffies, x-axis) for process 1 and 2, the second graph is the page fault over time for process 3 and 4.

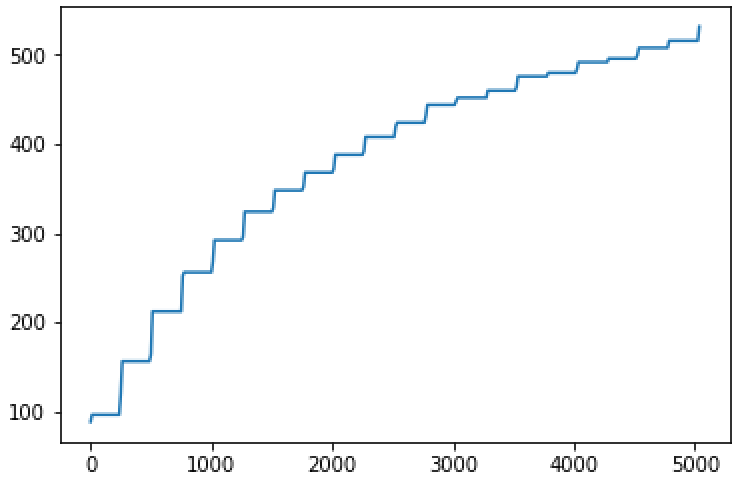
We can observe from the comparision between the 2 figures that the page fault count of the combination of processes of locality-based memory access pattern and random access pattern is much smaller than the combination of two random memory accessing processes. Also, we can derive from the data in the two files that the completion time of one process of locality-based access and the other of random access is much larger than that of two processes of random access pattern.

Case study 2:
For the 3 digures below, x-axis is the time in jiffies, and y-axis is the total cpu utilization of all processes in ms, where N= 1, 5, 11 for the 3 figures.

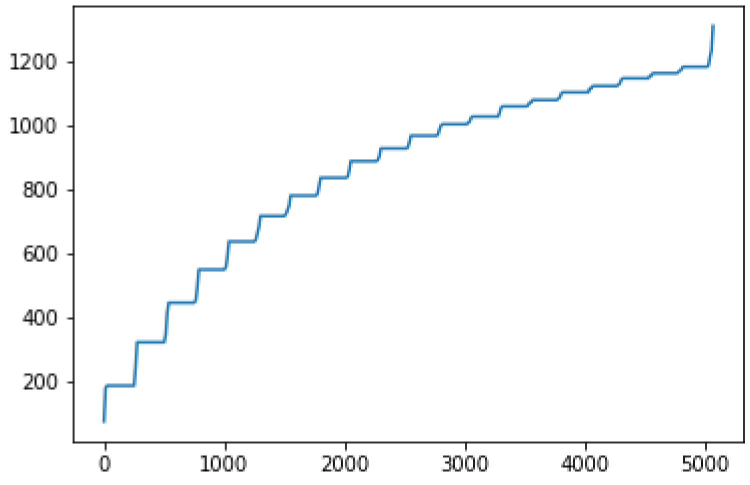


N = 1

We found that with the improvement of number of processes, the cpu utilization is increasing highly corelated with the number of processes. I think the reason is that the number of processes is not big enough to cause thrashing so that the cpu utilizaiton is always increasing with the degree of multiprogramming. In addition, the completion time is increasing slowly with the increase of N.



N = 5



N = 11