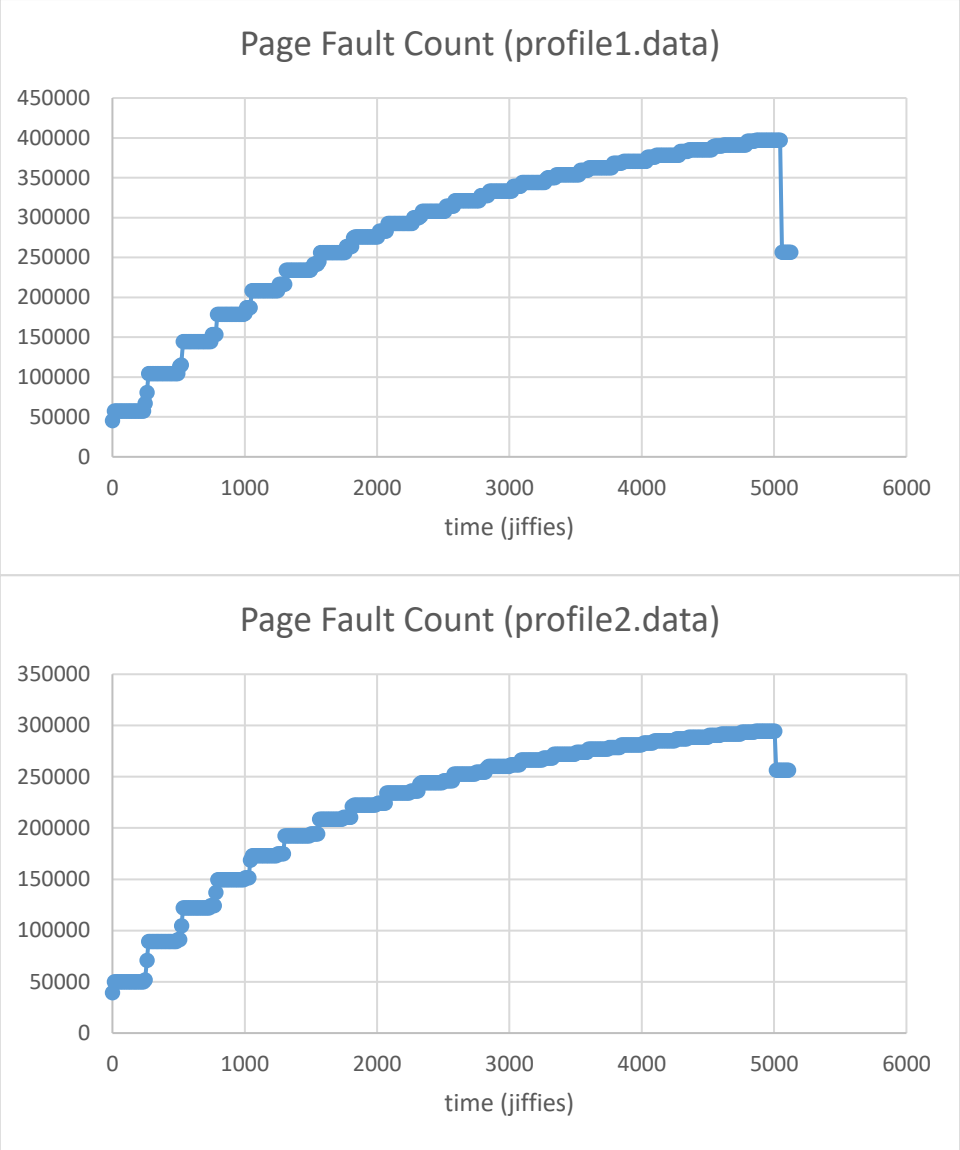
Case study 1:



When user malloc a piece of virtual memory, kernel usually doesn't map the virtual memory to physical memory until user tries to access it, which is the moment that minor page fault happens.

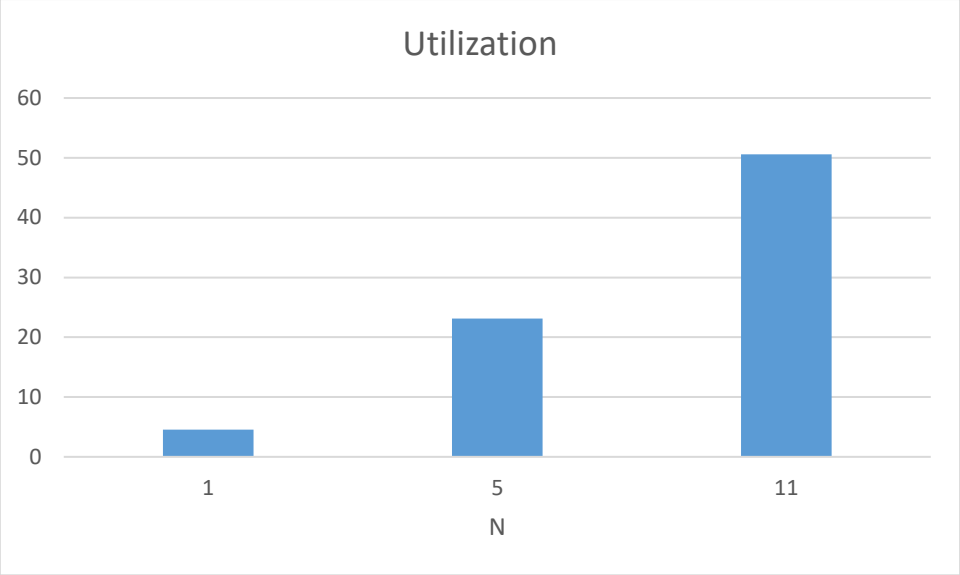
If kernel is running out of physical memory, it might swap some piece of memory out to disk. And when user reclaim that piece of memory, major page fault happens.

There are two kind of access function in work. One is rand_access, which access the virtual memory between buffer[i] and buffer[i]+1024*1024. Another one is local_access, which actually has nothing to do with the virtual memory, and would never trigger page fault.

The total page fault of the first case is much higher than the second case, which makes sense since the second work in the second case has chance to execute local_acess, and other three works have to execute rand_access every time.

The drop in two picture is because one of two tasks exit earlier. We only sample tasks that are running. And the range of drops differ because local_acess doesn't affect the page fault.

Case study 2:



The higher N is, the higher the cpu utilization is. It's because the more tasks we execute, the chance that one of the tasks be context swithed to run is higher, causing the cpu utilization higher.