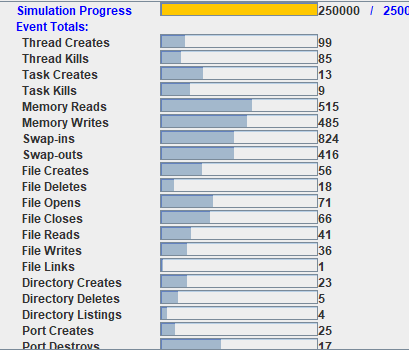
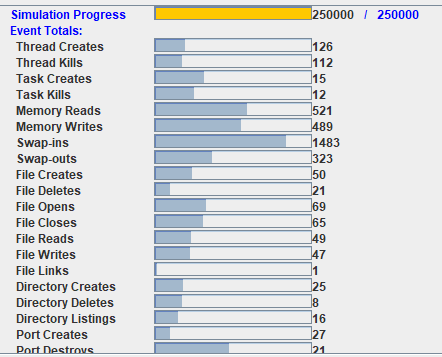
Statistics Viktoryia Strylets 111748510

The number of pages swapped in and swapped out:

Student statistics



Demo statistics:



Swapped-in Demo.jar: 1483 vs Osp.jar 910

Swapped-out OSP : 323 vs 416

Main reason of different numbers of swapped-ins and swapped- outs between two programs

**This is the difference in the Page replacement algorithms**.

Demo uses the simplistic algorithm for page replacement which is like the FIFO algorithm. It scans the entire frame table and chooses the first replaceable frame which is also the frame that has been in memory the longest. The main drawback of the algorithm it doesn’t recognizes the pages that heavily used throughout the life of the program. Those pages will be repeatedly paged in and out by the FIFO algorithm. Osp uses M2HC page replacement algorithm which is a more efficient compare to the previous algorithm. It **protects frequently referenced pages by setting the use bit to 1 at each reference.**

If the page replacement algorithm is not optimal then there will also be the problem of thrashing. If the number of pages that are replaced by the requested pages will be referred in the near future, then there will be a greater number of swap-in and swap-out and therefore the OS has to perform more replacements then usual which causes performance deficiency.

**Another reason is a different Fetch Policy used by programs.**

Demo uses the Prepaging Fetch Policy - brings in more pages than needed, which means more swap-ins . The theoretically it is more efficient to bring in pages that reside contiguously on the disk, but the extra pages not necessary and often not referenced. So, the efficiency not definitely established.

Osp uses Demand paging only brings pages into main memory when a reference is made to a location on the page (paging on demand only). Many page faults when a process is started but should decrease as more pages are brought in.

Moreover, it explains the more efficient statistic of the Demo program at the start of the program compare to the OSP.

**Different cleaning policy**

Moreover, algorithm perform proactive page cleaning by swapping every 10th page it finds out on swapping device. It utilizes the times when the swap device is idle and reduce the time needed to handle page faults by increasing the supply of clean pages.

Numbers shown for the next three statistics are averaged for 10 snapshots produced in each log file.

**CPU Utilization: Demo.jar 76.5128% vs Osp 85.5673%**

**Average service time per thread: Demo.jar 32105.363 vs Osp 29233.457**

**Average normalized service time per thread: Demo.jar 0.057458147 vs Osp.jar 0.659090349**

From the statistics we can see that the statistics for the OSP program slightly high in every aspect.

We can conclude that the paging policies used in the Osp and more important M2HC page replacement algorithm had better performance than the simple FIFO algorithm in the Demo.jar, because the page faults are very expensive and affects many aspects including CPU utilization, turnaround time of thread and average normalized service time.

By minimizing number of page-faults OSP increases the average service time of the threads because when the page fault occurs OS traps this fault and the interrupt handler services the fault by initiating a disk-read request. The read operation is very costly and slow. So, at the page fault the turnaround time of the thread is increasing by cost of page fault. Once page is brought in from disk to main memory, page-table entry is updated and the process which faulted is restarted.