Colby Boyer (cs162-aj) 3/9

Topic: Demand Paging, Thrashing, Working Sets

*The programmer's view of memory is abstracted away by using a memory mapping mechanism. This mechanism makes it easy for multiple users to share a single memory space, and for these users to be shuffled around in memory. An example of a memory mapping mechanism is virtual memory.

*Advantages of Virtual Memory: It allows only portions of a user process to be loaded into memory. This reduces resource use and makes the system more flexible. VA gives the impression of a much larger address space than physical memory can provide. It also makes other process's use of physical memory transparent. The programmer does not need to worry about accommodating other processes.

*2(Multi)-Level Memory: To give the user the appearance of fast (expensive) memory while retaining the cost and volume of cheap (slow) memory. Memory access time can range from nanoseconds (registers) to minutes (tape drives). Multi-level memory provides performance gains because of the principle of locality.

*Principle of Locality:
- *Temporal Locality – Information just accessed is likely to be accessed again
- *Spacial Locality – Nearby information is likely to be used

Paging:

*A page table uses a valid bit to determine if a page fault is to happen. A page fault traps to the OS.
*What happens at page fault?:
- *Program traps to OS
- *Checks to see if the page exists. If not throw error.
- *Find place to put the page. This may require the removal of an existing page.
- *Update page table
- *Update outside memory -swapping page in and out
- *Update memory map
- *Flush TLB

*Find page on secondary storage: (This is needed when VM is larger than PM)
- *Transfer Page
- *Update Page table
- *Update Map/filesystem (inodes)
- *Update Core map
- *resume execution

*These steps takes a lot of time and can slow the system down and one of the goals is to make the page fault unnoticeable to the user. Often the operating system overlaps execution of one process and memory transfers of another process to hide the latency. Also, it makes switching processes correctly tricky due to the complexity, and paging depends on hardware to work/be easy to implement.

Terminology:
Page out – remove page
Page out a process – remove from memory
Page in – load into memory

Algorithms:

Page Fetch:

*Demand paging: Starts up a process with none of the pages loaded. Pages are loaded when they are first referenced. The downside of this algorithm is that can be inefficient and time will be wasted waiting for each page to be loaded 1 by 1, since loading a page has a significant overhead. The upside is that it is simple to implement.

*Request Paging: User can ask for what pages it wants. However, the programmer does not always know best. An example is overlays, where the programmer performs segmentation by hand (very painful). The IBM Cell architecture does this and its difficult to work with.

*Prefetch or Prepaging: Requests pages before they are reference by the process. This allows for several pages to be transferred at once; reduces overhead/cuts time delay. However, the OS has to guess what pages will be needed, which is difficult to do and does not work well. To take advantage of transferring many pages at once, the last working set of pages is transferred.

Page Replacement:

*Random (RAND): Replace a random page in physical memory.
*FIFO: Replace the page that has been in memory for the longest.
*Least Recently Used (LRU): As the name implies. The idea is to use past history of page use to determine what pages should be replaced
*MIN: Replaces the page that will be used the least in the future. Cannot implement due to causality.

Real and Virtual Time:

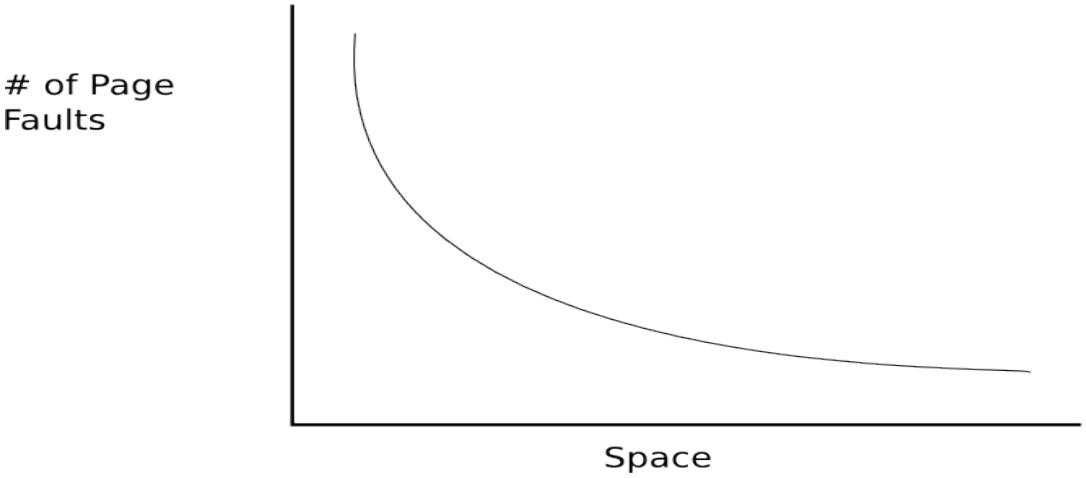
*Virtual Time – Time as measured by the process(time spend on CPU). Unit of time are memory accesses.
*Real Time – Wall Clock

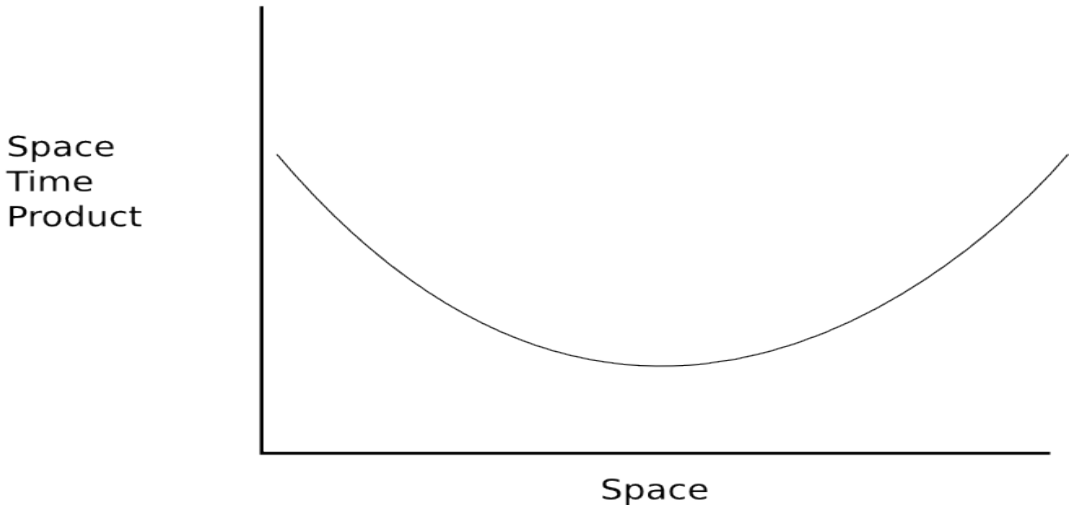
Cost of page faults:

*CPU overhead is around 3000 instructions (measured in 70s). This most likely 5-10 times higher in a modern OS.
*System may be idle, more I/O usage, memory delays, and real time delays are all effects of page faults.

Page Replacement Metrics:

*# of page faults vs space used
*Space Time Product (SPT) vs Space





*SPT is the integral of the amount of space used by the program over time. $\int space(t)dt$

*Discrete time: $\Sigma m(i)(1+\dot{f}(i)\dot{PFT})$ PFT is page fault time.