Paging with Multiprogramming

CS 537 - Introduction to Operating Systems

Global Allocation

- Processes compete for pages against one another
- Allows a process to use as many pages as it needs
  - if one is using 10 pages and another a 100, memory is allocated efficiently
  - if a processes changes from using 10 to using 100 pages, its new state can be met

Global Allocation

- Problem if one page is a memory "hog"
  - imagine database program that is just searching and another process that is only using a few pages
  - the database program will "steal" all of the pages even though it's not really using them
- A process may perform differently from one run to the next because of external factors
  - first time it runs with "good" processes
  - next time it runs with a "hog"

Local Allocation

- Assign a certain number of pages to each process to use for paging
- Can base this number on needs of process
  - a larger process can be allocated more frames
- This prevents memory "hog" problem
- What if a process doesn't need all of its frames?
  - another process that could use them won't get them

Virtual Time

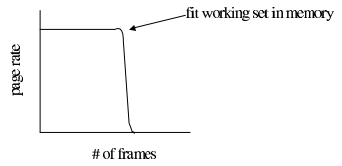
- Give each process its own clock
- Last reference of a page is based on the clock of the process that is using the page
  - not on a global time
- Using this, a process's pages are not punished because the process is context switched out

Virtual Time

- Keep the clock for a process in its PCB
- start(p)
  - PCB[p].lastStart = now
- stop(p)
  - PCB[p].virtualTime += now - PCB.lastStart

Working Set

- Minimum number of pages a process needs in memory to execute satisfactorily



- $W_\tau(p)$ = set of pages process p touched during the last τ seconds of virtual time

Thrashing

- If a processes working set is not in memory, the process will *thrash*
- Process spends most of its time reading in pages from disk
- Virtually no useful work will get done
- Better to kill a process than to let it thrash

Working Set Size

- How many pages make up the working set of a process?
  - measure each process carefully
    - requires running the process before hand
    - requires process behave the same way
  - ask the user
    - requires a user to be truthful
  - monitor the process
    - Page fault frequency monitoring
    - clock algorithm

Page Fault Frequency

- Give each process initial number of pages
- If process page faults are above a set threshold
  - give the process a new frame
- If process page faults are below a set threshold
  - take away a frame from the process
- If all of the process have too high of a page fault rate
  - kill some process

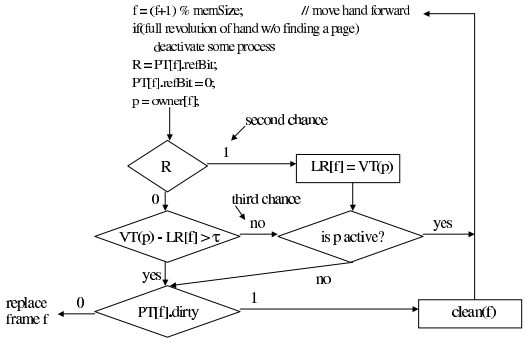
Clock Algorithm

- Use clock algorithm discussed before for doing replacement
  - modified slightly
- If the "hand" is moving too fast, kill some process
- If the "hand" is moving slowly enough, start some process

Clock Algorithm

- Some terminology
  - VT(p) = virtual time of a process
  - LR[f] = time of last reference to a frame
    - based on virtual time of owner process
  - owner[f] = process that owns a frame
  - PT[f] = page table entry for a frame

Clock Algorithm



Clock Algorithm

- Allows a process's frame ownership to grow
- A page is only taken away from a process if the process hasn't used it for a long time