CS341: Operating System

Virtual Memory: Page replacement, thrashing, Working Set

Lect 34: 29th Oct 2014
Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

Outline

• Frame allocation
• Working Set

Page Replacement Algorithm

• Input: Reference string, number of frame
• Output: number of page fault

• Timing information was missing in Reference string
• How to allocate the frame for process?
  – Static (fixed size for process life time)
  – Dynamic

Allocation of Frames

• Each process needs minimum number of frames
• Example: IBM 370 – 6 pages to handle SS MOVE instruction:
  – instruction is 6 bytes, might span 2 pages
  – 2 pages to handle from
  – 2 pages to handle to

Graph of Page Faults Versus The Number of Frames

Allocation of Frames

• Maximum of course is total frames in the system
• Two major allocation schemes
  – fixed allocation
  – priority allocation
• Many variations

Allocation of Frames

• Each process needs minimum number of frames
• Example: IBM 370 – 6 pages to handle SS MOVE instruction:
  – instruction is 6 bytes, might span 2 pages
  – 2 pages to handle from
  – 2 pages to handle to
**Fixed Allocation**

- Equal allocation
  - Allocate equal number of frame to each process
  - For example, if there are 100 frames (after allocating frames for the OS) and 5 processes, give each process 20 frames
  - Keep some as free frame buffer pool
- Proportional allocation
  - Allocate according to the size of process
  - Dynamic as degree of multiprogramming, process sizes change

**Priority Allocation Based on Page Fault Rate**

- Use a proportional allocation scheme using priorities rather than size
- If process $P_i$ generates a page fault,
  - select for replacement one of its frames
  - select for replacement a frame from a process with lower priority number

**Cache Friendly and Cache Thrashing Apps**

1. Reading a data with Temporal locality (same data reference many time)
2. Reading data in streaming manner, referred one time

**Global vs. Local Allocation**

- Global replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another
  - But then process execution time can vary greatly
  - But greater throughput so more common
  - Interference: Interferer, Sufferer, Pinning/Quota
- Local replacement – each process selects from only its own set of allocated frames
  - More consistent per-process performance
  - But possibly underutilized memory

**Non-Uniform Memory Access**

- So far all memory accessed equally
- Many systems are NUMA – speed of access to memory varies
  - Consider system boards containing CPUs and memory, interconnected over a system bus

**Non-Uniform Memory Access**

- Optimal performance comes
  - Allocating memory "close to" the CPU on which the thread is scheduled
  - And modifying the scheduler to schedule the thread on the same system board when possible
- Solved by Solaris by creating lgroups
  - Structure to track CPU / Memory low latency groups
  - Used my schedule and pager
  - When possible schedule all threads of a process and allocate all memory for that process within the lgroup
Thrashing
• If a process does not have "enough" pages, the page-fault rate is very high
  – Page fault to get page
  – Replace existing frame
  – But quickly need replaced frame back
  – This leads to:
    • Low CPU utilization
    • Operating system thinking that it needs to increase the degree of multiprogramming
    • Another process added to the system

  • Thrashing: a process is busy swapping pages in and out

Demand Paging and Thrashing
• Why does demand paging work?
  Locality model
  – Process migrates from one locality to another
  – Localities may overlap
• Why does thrashing occur?
  \[ \sum \text{size of locality} > \text{total memory size} \]
  – Limit effects by using local or priority page replacement
  – Modeling locality by Working Set Window

Working-Set Model
• Working-set window (\(\Delta\)): A fixed number of page references
  Or example: 10,000 instructions
• \(WSS_i\) (working set of Process \(P_i\)) = total number of pages referenced in the most recent \(\Delta\) (varies in time)
  – if \(\Delta\) too small will not encompass entire locality
  – if \(\Delta\) too large will encompass several localities
  – if \(\Delta = \infty\) \(\Rightarrow\) will encompass entire program

Working-Set Model
\[ D = \sum WSS_i = \text{total demand frames} \]

• Approximation of locality
• Number of frames in locality : \(m\)
• if \(D > m\) \(\Rightarrow\) Thrashing
• Policy if \(D > m\), then suspend or swap out one of the processes
**Keeping Track of the Working Set**
- Approximate with interval timer + a reference bit
- Example: $\Delta = 10,000$
  - Timer interrupts after every 5000 time units
  - Keep in memory 2 bits for each page
  - Whenever a timer interrupts copy and sets the values of all reference bits to 0
  - If one of the bits in memory = 1 $\Rightarrow$ page in working set
- Why is this not completely accurate?
- Improvement = 10 bits and interrupt every 1000 time units

**Page-Fault Frequency**
- More direct approach than WSS
- Utility based frame allocation
  - How much is getting utilized by allocating extra frames?
  - If Page fault is getting lower then beneficial
- Establish “acceptable” page-fault frequency (PFF) rate and use local replacement policy
  - If actual rate too low, process loses frame
  - If actual rate too high, process gains frame

**Page-Fault Frequency**

![Page-Fault Frequency Diagram](image)

**Working Sets and Page Fault Rates**
- Direct relationship between working set of a process and its page-fault rate
- Working set changes over time
- Peaks and valleys over time

![Working Sets and Page Fault Rates Diagram](image)