CSE 4/521
Introduction to Operating Systems

Lecture 18 – Virtual Memory IV
(Allocation of Frames, Thrashing, Operating System Examples)
Summer 2018
Overview

• Allocation of Frames
• Thrashing
• Operating System Examples
Recap

• Page Replacement Algorithms
  • FIFO, Belady’s Anomaly, Optimal Algorithm, LRU Algorithm, Second Chance Algorithms, Counting Algorithms
Questions

1. You have devised a new page-replacement algorithm that you think may be optimal. In some test cases, Belady’s anomaly occurs. Is the new algorithm optimal? (Medium)

2. In which circumstances would LFU produce less page-faults than LRU? (Medium)

3. In which circumstances would MFU produce less page-faults than LRU? (Medium)
Overview

• Allocation of Frames
• Thrashing
• Operating System Examples
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*

**Maximum** allocation is total frames in the system.

Two major allocation schemes:
- Fixed allocation
- Priority allocation

Many variations.
Allocation of Frames: Fixed Allocation

- **Equal allocation** – 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

\[
s_i = \text{size of process } p_i
\]
\[
S = \sum s_i
\]
\[
m = \text{total number of frames}
\]
\[
a_i = \text{allocation for } p_i = \frac{s_i}{S} \times m
\]

\[
m = 62
\]
\[
s_1 = 10
\]
\[
s_2 = 127
\]
\[
a_1 = \frac{10}{137} \times 62 \approx 4
\]
\[
a_2 = \frac{127}{137} \times 62 \approx 57
\]
Allocation of Frames: Priority Allocation

• 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
Allocation of Frames:
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

- **Local replacement** — each process selects from only its own set of allocated frames
  - More consistent per-process performance
  - But possibly underutilized memory
Allocation of Frames: Non-Uniform Memory Access

- So far all memory accessed equally
- Many systems are NUMA – speed of access to memory varies
- Optimal performance comes from allocating memory “close to” the CPU on which the thread is scheduled
  - Solved by Solaris by creating lgroups
    - Structure to track CPU / Memory low latency groups
    - When possible, schedule all threads of a process and allocate all memory for that process within the lgroup
Overview

- Allocation of Frames
- Thrashing
- Operating System Examples
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
Thrashing

![Diagram showing CPU utilization vs. degree of multiprogramming, with a region marked as 'thrashing'.]
Thrashing: 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 effect of thrashing by using local or priority page replacement
Thrashing: Locality in a Memory-Reference Pattern
Thrashing: 
Working-Set Model

- \( \Delta \equiv \) working-set window \( \equiv \) a fixed number of page references (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

- \( D = \sum WSS_i \equiv \) total demand for frames
  - Approximation of locality

- if \( D > m \) \( \Rightarrow \) Thrashing (\( m \) is total number of available frames)

- Policy: If \( D > m \), then suspend or swap out one of the processes
Thrashing:
 Keeping Track of the Working Set

• Approximate with interval timer + a reference bit

• Example: Δ = 10,000
  • Timer interrupts after every 5000 time units
  • Keep in memory 2 bits for each page
  • Whenever a timer interrupts, copy and set the values of all reference bits to 0
  • If one of the bits in memory = 1 ⇒ page in working set

• It is not completely accurate though.

• Possible Improvement: 10 bits and interrupt every 1000 time units
Thrashing: Page-Fault Frequency

- More direct approach than WSS
- 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 rate
```

```
number of frames
```

- Increase number of frames if rate is too low
- Decrease number of frames if rate is too high

Upper bound and lower bound to control page-fault rate.
Thrashing: 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
Overview

• Allocation of Frames
• Thrashing
• Operating System Examples
Operating System Examples: Windows

- Uses demand paging with clustering. Clustering brings in pages surrounding the faulting page.
- Processes are assigned working-set minimum and working-set maximum.
- Working set minimum is the minimum number of pages the process is guaranteed to have in memory.
- A process may be assigned as many pages up to its working set maximum.
- When the amount of free memory in the system falls below a threshold, automatic working set trimming is performed to restore the amount of free memory.
- Working set trimming removes pages from processes that have pages in excess of their working set minimum.
Operating System Examples: Solaris

- Maintains a list of free pages to assign faulting processes
- **Lotsfree** – threshold parameter (amount of free memory) to begin paging
- **Desfree** – threshold parameter to increasing paging
- **Minfree** – threshold parameter to being swapping
- **Paging is performed by pageout process**
- **Pageout** scans pages using modified clock algorithm
- **Scanrate** is the rate at which pages are scanned. This ranges from slowscan to fastscan
- **Pageout** is called more frequently depending upon the amount of free memory available
- **Priority paging** gives priority to process code pages
Operating System Examples: Solaris 2 Page Scanner

- Solaris 2 Page Scanner
- minfree
- slowscan 8192
- fastscan
- desfree
- lotsfree

scan rate

8192 fastscan

100 slowscan

minfree
desfree
lotsfree

amount of free memory
Credits for slides

Silberschatz, Galvin and Gagne