More on Virtual Memory

CS 256/456
Dept. of Computer Science, University of Rochester

Recap of the Last Class

- **Virtual memory** - separation of user logical memory from physical memory.
  - Only part of the program address space needs to be in physical memory for execution.
  - Copy-on-write: allows for more efficient process creation.
  - Memory-mapped I/O.

- Page replacement algorithm: the algorithm that picks the victim page.
  - Metrics: page-fault rate; implementation cost/feasibility.
  - FIFO, Optimal, LRU, LRU approximation.

Counting-based Page Replacement

- Least frequently used page-replacement algorithm
  - the page with smallest access count (within a period of time) is replaced

- Implementation?

Transparent Sharing of Memory Pages

- You are asked to implement a memory management system that transparently share pages of the same content over multiple processes.
  - pay some overhead is fine, but not too excessive.
  - discovering pages with the same content.
  - copy-on-write.

- How often do pages have the same content?
Frame Allocation among Processes

- **Global** replacement - process selects a replacement frame from the set of all frames; one process can take a frame from another.
- **Local** replacement - each process selects from only its own set of allocated frames.
- Pros and cons?

Thrashing

- If a process does not have "enough" pages, the page-fault rate is very high. This leads to low CPU utilization.
- **Thrashing** ≡ a process is mostly busy with swapping pages in and out.

Locality In A Memory-Reference Pattern

Working-Set Model

- data access locality.
- \(WSS(P)\) (working set of Process \(P\)) = total number of pages referenced in the most recent \(\Delta\) (working-set window)
  - if \(\Delta\) too small, we will not encompass sufficient locality.
  - if \(\Delta\) too large, we will over-estimate localities.
  - if \(\Delta = \infty\) ⇒ will encompass entire program.

Page reference table:

```
... 2 6 1 5 7 7 7 5 1 6 2 3 4 1 2 3 4 4 3 4 4 4 4 4 4 4 ...
```

\[WS(t_1) = (1, 2, 5, 6, 7)\]
\[WS(t_2) = (3, 4)\]
**Working-Set Model**

- Policy:
  - try to allocate enough frames for each process’s working set.
  - if $\sum WSS_i > m$, then suspend one of the processes.

- How to keep track of the process working set size in practice?

**Page-Fault Frequency Scheme**

- Establish “acceptable” page-fault rate range.
  - If actual rate too low, process loses frame.
  - If actual rate too high, process gains frame.

**Other Considerations**

- When to swap out pages?
- Prepping
  - swap in pages that are expected to be accessed in the future
- Page size selection
  - fragmentation
  - page table size
  - TLB reach

**TLB Reach**

- **TLB Reach** - the amount of memory accessible from the TLB.
  - $\text{TLB Reach} = (\text{TLB Size}) \times (\text{Page Size})$

- Advantage of a large TBL reach?

- Increasing the TLB reach:
  - **Increase the Page Size.** This may lead to an increase in fragmentation as not all applications require a large page size.
  - **Provide Multiple Page Sizes.** This allows applications that require larger page sizes the opportunity to use them without an increase in fragmentation.
  - Implementation?
Performance Impact on User Programs

- Assume:
  - page size is 4KB
  - the program is allocated 1023 physical frames.
- Program structure
  - int A[][] = new int[1024][1024];
  - Program 1
    - for (j = 0; j < A.length; j++)
      - for (i = 0; i < A.length; i++)
        - A[i,j] = 0;
    - 1024 x 1024 page faults
  - Program 2
    - for (i = 0; i < A.length; i++)
      - for (j = 0; j < A.length; j++)
        - A[i,j] = 0;
    - 1024 page faults

Page Locking

- Pages must sometimes be locked into memory.
- Consider I/O. Pages that are used for copying a file from a device must be locked from being selected for eviction by a page replacement algorithm.

Memory Management in a MicroKernel

- Where is the page replacement algorithm?

Memory Management in ExoKernel

- In a normal microkernel OS, the external pager runs in a separate protected process, often owned by the superuser
- In ExoKernel, an external pager runs (as a library) within the address space of each process
  - the kernel memory manager allocates memory across processes
  - the external pager for each process manages memory for itself (so it can use the page replacement algorithm that best fits itself)
  - the external pager cannot modify its own page table at will
Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).