OPERATING SYSTEMS

Memory Management

Suresh Jamadagni
Department of Computer Science

OPERATING SYSTEMS

Allocation of Frames

Suresh Jamadagni
Department of Computer Science

OPERATING SYSTEMS

Slides Credits for all the PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

OPERATING SYSTEMS

Allocation of Frames

- How do we allocate the fixed amount of free memory among the various processes?

Example:

- Consider a single-user system with 128 KB of memory composed of pages 1 KB in size. Therefore, this system has 128 frames.
- The operating system may take 35 KB, leaving 93 frames for the user process.
- When a user process started execution, under pure demand paging, it would generate a sequence of page faults.
- The first 93 page faults would all get free frames from the free-frame list.
- When the free-frame list is exhausted, a page-replacement algorithm would be used.
- When the process is terminated, the 93 frames would once again be placed on the free-frame list

Allocation of Frames

Minimum number of frames:

- One reason for allocating at least a minimum number of frames involves performance.
- As the number of frames allocated to each process decreases, the page-fault rate increases, slowing process execution
- When a page fault occurs before an executing instruction is complete, the instruction must be restarted. Consequently, we must have enough frames to hold all the different pages that any single instruction can reference
- The minimum number of frames is defined by the computer architecture.
- The maximum number is defined by the amount of available physical memory.
- In between, we are still left with significant choice in frame allocation

OPERATING SYSTEMS

Frame Allocation Algorithms

Equal allocation:

- The easiest way to split m frames among n processes is to give everyone an equal share, **m/n** frames (ignoring frames needed by the operating system)

  Example:

  - If there are 93 frames and 5 processes, each process will get 18 frames.
  - The three leftover frames can be used as a free-frame buffer pool.

Frame Allocation Algorithms

Proportional allocation:

- Allocate available memory to each process according to its size
- Let the size of the virtual memory for process p_i be s_i and define $S = \sum s_i$
- If the total number of available frames is m, allocate a_i frames to process p_i , where a_i is approximately $a_i = s_i / S \times m$
- Adjust each a_i to be an integer that is greater than the minimum number of frames required by the instruction set, with a sum not exceeding m
- **Example**: To allocate 62 frames between two processes, one of 10 pages and one of 127 pages
  - $10/137 \times 62 \approx 4$
  - $127/137 \times 62 \approx 57$

OPERATING SYSTEMS

Global versus Local Allocation

- Page-replacement algorithms can be classified into two broad categories: global replacement and local replacement
- **Global replacement** allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another.
- **Local replacement** requires that each process select from only its own set of allocated frames.
- **Example**: Consider a global replacement scheme wherein high-priority processes are allowed to select frames from low-priority processes for replacement.
- A process can select a replacement from among its own frames or the frames of any lower-priority process. This approach allows a high-priority process to increase its frame allocation at the expense of a low-priority process
- With a local replacement strategy, the number of frames allocated to a process does not change.
- Global replacement generally results in greater system throughput and is therefore the more commonly used method.

OPERATING SYSTEMS

Non-uniform Memory Access

- In systems with multiple CPUs, a given CPU can access some sections of main memory faster than it can access others.
- These performance differences are caused by how CPUs and memory are interconnected in the system.
- Systems in which memory access times vary significantly are known collectively as **non-uniform memory access (NUMA)** systems, and without exception, they are slower than systems in which memory and CPUs are located on the same motherboard
- Managing which page frames are stored at which locations can significantly affect performance in NUMA systems. If we treat memory as uniform in such a system, CPUs may wait significantly longer for memory access than if we modify memory allocation algorithms to take NUMA into account.

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu