CSE451 Memory Management Introduction Spring 2001

Gary Kimura
Lecture #13
April 23, 2001

Today

- Midterm Results
- Memory Management
  - Overview
  - Virtual and physical address space
  - A cursory look at three memory management techniques

Your job this week

- Readings in Silberschatz
  - Chapter 8 and 9
- Homework #4
  - Out today Monday April 23, 2001
  - Due next Monday April 30, 2001
  - Silberschatz questions 8.6, 8.10
  - And a third question to be posted on the class web page

Midterm Results

- Average score 30.25 stdev 6.6

Problem	Average	Max	Problem	Average	Max
#1	1.6	3.0	#5	2.8	5.0
#2	2.7	3.0	#6	4.6	5.0
#3	6.3	9.0	#7	2.8	4.0
#4	3.8	6.0	#8	5.4	15.0

Simple Programs, Simple Memory

- Remember back to simple programs and the memory model they use.
- They live in a virtual world, a linear address space not based on physical memory (i.e., reality).

Address Space Introduction

- First memory isn't all scattered around with little nice names
- It is a set of sequentially numbered cells (bytes).
- Physical memory is also a set of sequentially numbered cells (bytes).
- In early systems your virtual world was identical to your physical world
- Let's draw a picture to help illustrate the concept

Address space in more Hardware terms

- A logical or virtual address is the memory location generated by the CPU
- A physical address is the memory location of a cell as seen by the memory unit
- Need hardware support to quickly and transparently translate Virtual to Physical

Memory Management Units

- A hardware memory management unit is typically stuck between the CPU and main memory. So each memory access maps a Virtual Address to a Physical Address
- Virtual address space is typically larger than physical memory
- Hardware units are usually biased toward either paging or segmentation

Memory Management techniques

- What do we do when the sum of all the virtual memory doesn't fit into physical memory?
  - Swapping: not everyone is allowed into memory at the same time
  - Segmentation: divide a program into logical units and allow units to be swapped in and out or memory
  - Paging: everyone is allowed only part of their program into memory at a time
- Paging is real the one in vogue today, but we'll quickly look at the other two to better understand the problem

Swapping

- Swapping is where executing programs are temporarily copied out of memory to a backing store such as a disk drive.
- This frees up physical memory for other programs to be swapped in.
- Swapping is similar in concept to a context switch however it is a lot more expensive.

Segmentation

- Another scheme for resolving virtual to physical addresses. Each function or program is assigned to a segment
- Hardware support needed
  - Segment table: Each entry in the table corresponds to a segment and contains the physical base and limit of the segment
  - Protection and sharing: Two processes can share segments.
  - Fragmentation: It may not allows be possible to find a big enough hole in physical memory to load a new segment.

Paging

- Divide up virtual and physical memory into pages (usually in the 4KB or 8KB range)
- Hardware support needed
  - Page table: map all virtual pages to physical pages. Page tables can be large.
  - TLB (Translation Look-aside Buffer): cache of recently accessed page table entries.
  - Page protection: indicates the state of each logical and physical page. Is the mapping valid and who has access to the page

Logical and physical memory in the paging model

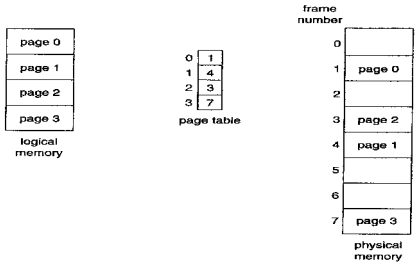

Figure 8.13 Paging model of logical and physical memory.

TLB Hardware

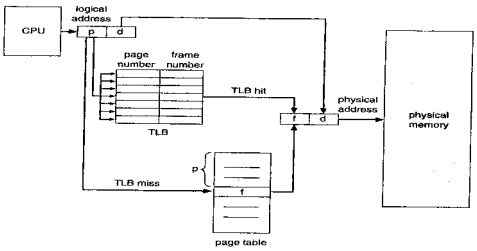

Figure 8.16 Paging hardware with TLB.

Paging (continued)

- Multilevel paging: Used to reduce the amount of memory needed to store the page table.
- Inverted page tables: Each physical page has a table entry identifying the process.
- Shared pages: Physical pages can show up in multi page table entries.
  - This allows for sharing executable code pages.
  - It also can be used for copy-on-write pages.

Issues (addressed in later lectures)

- Fragmentation: Essentially wasted memory that cannot be used to store real data
  - Internal fragmentation: Where each allocation unit potentially has a bit of wasted memory .
  - External fragmentation: where physical memory is being divided into various sized holes.
- Which pages or segments should be loaded and/or removed from physical memory?
- How does the system allocate or grow the physical memory supporting their virtual address space and how is that represented in the system?
- Kernel address space has additional issues
  - Must all the kernel code and data always be in physical memory?

Next Time

- Virtual Memory and a more in-depth look at paging