Slide 1

Administrivia

- (Via e-mail.)

Slide 2

“Beyond Physical Memory” — Problem

- We’ve assumed so far that all address-space pages for all processes fit into physical memory.
- What if they don’t? if the address space is really big, or if there are a lot of processes?
- (This used to be a much bigger problem, but current systems often have a *lot* of memory. But they don’t always. So might be good to know why systems with less memory still work but aren’t fast.)

Slide 3

"Beyond Physical Memory" — Old Solution

- Once upon a time — explicit "overlays".
- Idea was, e.g., to split up program into logical pieces not all needed at the same time, keep in memory only the pieces in use, explicitly move them into physical memory as needed (and remove them when not).
- Sounds painful and error-prone, no? but probably better than nothing.

Slide 4

"Beyond Physical Memory" — Better Solution

- Traditional solution: Keep some pages somewhere where we have closer-to-unlimited space, even if it's slower.
- Traditionally that means (spinning) disk, though now could mean something less slow such as solid-state device.
- So basically we're using disk to hold the full complement of address-space pages for multiple processes, and using RAM ("physical memory" in this section) to hold the parts in active use. (Caching to the rescue again!)

Slide 5

Swapping, Continued

- Where on disk? “Swap space”, logically partitioned into page-size chunks.
- Swap space can be one big file, or even a whole disk partition. (More about this when we talk about I/O.)
- Worth noting that size of swap space does constrain how much of this virtualized memory can be supported.

Slide 6

Mechanisms for Swapping

- In a sense, managing swapping is much like managing TLB misses, though unlike with those it always involves the O/S:
- Page-table entry also has bit indicating that page is valid but not present in memory. (Could combine with bit that says “not valid”.)
- In translating a virtual address, MMU still first tries TLB; on a miss, translates using page table. If page valid but not present, generates a “page fault” interrupt, transfers to O/S’s handler for those.
- (Recall that TLB misses can be handled in hardware or software. Simpler to understand if done by hardware, but principles the same.)

Slide 7

Page Faults

- Handling page faults done by O/S handler for page-fault interrupts. (Note that terminogy not always consistent — “page fault” can mean “page not in memory” or “error”.)
- Where to find page in swap space? Could be stored in place of page-frame number in PTE, or could be stored elsewhere.
- O/S finds a free page frame, issues I/O to fetch page, blocks process while waiting for it to complete.
- When I/O completes, O/S can update PTE and retry instruction that generated page fault.

Slide 8

Page Faults, Continued

- “Find a free page frame” . . . Not so fast, you say? Right.
- Physical memory is acting as a cache for the global state of memory (all valid pages of all address spaces), and as with other caches, if it fills up . . .
- Must consider what existing entry to remove — “page-replacement policy” or “page-replacement algorithm”.

Slide 9

Page Replacement Policies

- As noted, sometimes need more physical memory than is free, so must choose something in physical memory and "evict" it. Choose how?
- As with other kinds of caching, goal is reduce number of "cache misses" (here, page faults).
- Textbook goes into some detail about measuring frequency of misses and calculating performance cost. Interesting but for reasons of time we'll skip details. Noteworthy, however, that disk I/O is so much slower than processor speeds or even access to RAM that even a few page faults can affect performance.
- So how to do that . . . (To be continued.)

Slide 10

Finding Free Page Frames

- (Okay, one more thing today . . .)
- Often beneficial to have a background process maintaining a pool of free pages rather than waiting for memory to fill completely — "swap daemon" or "paging daemon".
- Same strategy of using background processes can be useful in other contexts too.

Slide 11

Minute Essay

- Questions? Is this making sense? Once again I feel like if you understand the big picture so much is common sense!