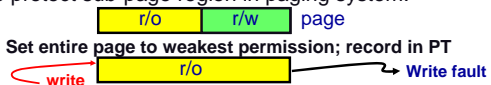


CS 140: Operating Systems

Lecture 12: Paging/VM Tricks

Mendel Rosenblum

Flashback: Faults + resumption = power

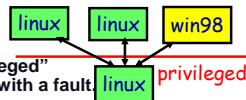
- ◆ The ability to resume after a fault lets the OS emulate a huge number of things.
("every problem can be solved with layer of indirection")
- ◆ Example: sub-page protection
Nice thing about segmentation: lets us protect arbitrary sized byte ranges.
- ◆ To protect sub-page region in paging system:


Set entire page to weakest permission; record in PT

Any access that violates perm will cause an access fault
Fault handler checks if page special, and if so, if access allowed. Continue or raise error, as appropriate

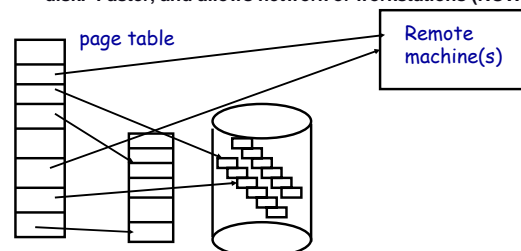
Fault resum. lets us lie about many things

- ◆ Emulate reference bits:
Set page permissions to "invalid".
On any access will get a fault: Mark as referenced
- ◆ Emulate non-existent instructions:
Give inst an illegal opcode. When executed will cause "illegal instruction" fault. Handler checks opcode: if for fake inst, do, otherwise kill.
- ◆ Run OS on top of another OS!
Slam OS into normal process
When it does something "privileged" the real OS will get woken up with a fault.
If op allowed, do it, otherwise kill.
IBM's VM/370. vmware.com



Distributed shared memory

- ◆ Virtual memory allows us to go to memory or disk
But, can use the same idea to go anywhere! Even to another computer. Page across network rather than to disk. Faster, and allows network of workstations (NOW)



DSM "details" (still about 10,000 ft up)

- ◆ Simplest approach: each page has one owner (trivially coordinates multiple updates to same page)
- ◆ Page table tracks resident/non-resident for page on disk: the block holding it for page on network: the owning machine
Difference? The latter may be wrong.
Truth in a distributed setting tends to be expensive. (coordinating all machines does **not** scale.)
So, location is frequently just a hint. If not at the location, use more expensive algorithm to find it.
- ◆ Page fault:
on disk? Fetch. On network, find owner, claim page.
- ◆ Tradeoffs in page size: large page -> "false sharing"

Swapping mechanics

- ◆ Allocate contiguous region on disk (ideal: across disks)
- ◆ When to allocate space?
When page allocated? (BSD 4.3)
When you need to page?
- ◆ What about code pages?
Get from file? (BSD 4.3)
Page to swap? (Why?)
- ◆ When to swap process?
4.3 BSD 'swapper' (pid 0): 20 seconds idle (gone home), when thrashing, take longest resident of 4 largest ones.
- ◆ Note: Early systems used swapping for protection!

