ECE 598 – Advanced Operating Systems Lecture 15

Vince Weaver

http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

24 March 2016

Announcements

- Homework #7 will be posted

Virtual Memory Wrapup

Large Pages

- Another way to avoid problems with 64-bit address space
- Larger page size (64kB? 1MB? 2MB? 2GB?)
- Less granularity. Potentially waste space
- Fewer TLB entries needed to map large data structures
- Compromise: multiple page sizes.
  Complicate O/S and hardware. OS have to find free blocks of contiguous memory when allocating large page.

- Transparent usage? Transparent Huge Pages? Alternative to making people using special interfaces to allocate.

Having Larger Physical than Virtual Address Space

- 32-bit processors cannot address more than 4GB
  x86 hit this problem a while ago, ARM just now
- Real solution is to move to 64-bit
- As a hack, can include extra bits in page tables, address more memory (though still limited to 4GB per-process)
- Linus Torvalds hates this.

- Hit an upper limit around 16-32GB because entire low 4GB of kernel addressable memory fills with page tables

ARM 1176 (ARMv6) MMU

- See Chapter 6 of the ARM1176 Technical Reference Manual `http://infocenter.arm.com/help/topic/com.arm.doc.ddi0301h/DDI0301H_arm1176jzfs_r0p7_trm.pdf`
- page sizes: 4KB, 64KB, 1MB, and 16MB
- you can specify access permissions for 64KB large pages and 4KB small pages separately (Subpages)
- micro-TLB

- one 64-entry unified TLB and a lockdown region of eight entries
- you can mark entries as a global mapping, or associated with a specific application space identifier to eliminate the requirement for TLB flushes on most context switches
- access permissions extended to enable Privileged read-only and Privileged or User read-only modes to be simultaneously supported
- hardware page table walks

- separate Secure and Non-secure entries and page tables
- Small fast microTLB. 1 cycle. Has ASID
- Larger slower main TLB, 64-entry 2-way
- Enable MMU by writing to special register in CP15

ARM 1176 (ARMv6) Caches

- L1 instruction cache (32kB?)
- L1 data cache (32kB?). Aliasing if VM enabled (bits 12:13). Need to do page coloring, only use 4kB pages, or limit cache size to 16KB
- TCM (tighly coupled memory) small area of up to 32kb that you can map to arbitrary page location. Fast, cache speeds. Useful for things like FIQ handlers in real time systems.

Devices

- Character devices – read a sequence of characters incoming, like a serial connection
  open/read/write
- Block devices – read chunks of data with random access, can seek back and forth
  open/read/write/lseek
- /dev, mknod

Block Devices

- Disks?
- Ramdisk?

Filesystems

- What is a file?
  Unix is just a stream of bytes. That's not necessarily the only thing. Old systems files were just 80-column punch card images.

- How do you store it?
  In a "file"?
  Everything in root directory? Hiearchical? Database?

- Why not just load everything into memory? Too big?

Share with other processes? Persistent across reboots?

Filesystems metadata

- Name?
  - UNIX – anything that is not / or NUL.
    What about ”-rf” or * or ?
  - Windows, more restrictions. Thins like CON and COM and PTR. Can cause problems.
  - DOS famously had 8.3 filenames.
  - Mac used to use : as directory separator.
- Case sensitive? Bob or bob or BoB?

- Special files?
  - Regular files
  - Directories
  - Char devices, block devices
  - Links (hard, soft)
- Attributes
  - Permissions (read, write, execute)
  - Owner (user, group)
  - Access time (atime, ctime, mtime)
  - Current size

– Locks
– Hidden
– Immutable / System
– Extended attributes / capabilities