Mach 3.0 Machine-Dependent Layer

Alessandro Forin

af@cs.cmu.edu

Summary

- The PMAP module

- Other machine dependencies

The Pmap Module

- Implements the pmap_t data type, encapsulating VM support hardware

- "Keeps the hardware happy"

- Can lazily and/or incompletely implement some of the required functionality

- Has strong effects on performance of the overall system

Some Common MMUs

- Flat tables (Vax)

- Trees: Encore, Sun3, i386

- Hash tables: RT & RS6000

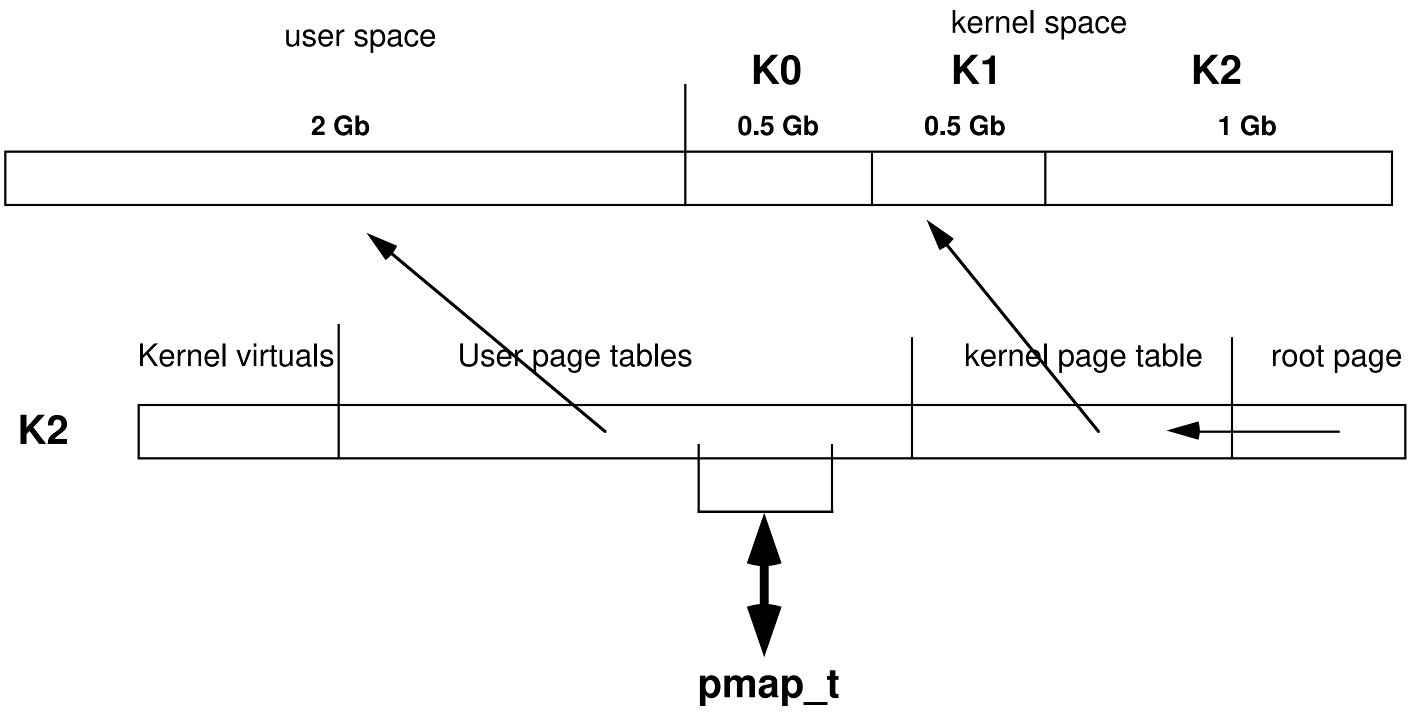
- Software-reloaded: MIPS

Example: Mips

- Software-reloaded TLB
- K0 / K1 wired segments
- TLB 'process IDs'
- Small number of 'wired' TLB entries

Chosed to use virtual arrays of page tables
chiefly because of small hardware assist

Page Tables (mips)



Pmap: The Big Ones

pmap_create   pmap_destroy

pmap_enter pmap_remove pmap_remove_all

pmap_protect

pmap_activate pmap_deactivate

pmap_zero_page pmap_copy_page

Pmap: Bootstrapping

pmap_steal_memory   pmap_free_pages

pmap_startup   pmap_init   [pmap_bootstrap]

Optional: pmap_next_page   pmap_virtual_space

Pmap: Ref/Mod Bits

pmap_clear_reference   pmap_is_referenced

pmap_clear_modify   pmap_is_modified

pmap_page_protect

Pmap: Miscellaneous

pmap_kernel pmap_reference

pmap_extract pmap_access

pmap_collect

pmap_change_wiring

pmap_phys_address pmap_phys_to_frame

pmap_statistics pmap_resident_count

Optional: pmap_copy pmap_attribute

Pmap: Activation

PMAP_ACTIVATE and PMAP_DEACTIVATE install the translations of the given pmap on the current CPU.

- Vax TLB flushes

- MultiP bookeeping and synch for TLB shootdown

Pmap_enter/remove

Pmap_enter:

get necessary ptepages
lock [and Shoot]
enter (v,pmap) in PV-list
if (new mapping)
  check non-cachable bit
  enter mapping
  update (our) TLB
  flush cache if needed
else
  apply change
  update (our) TLB
unlock

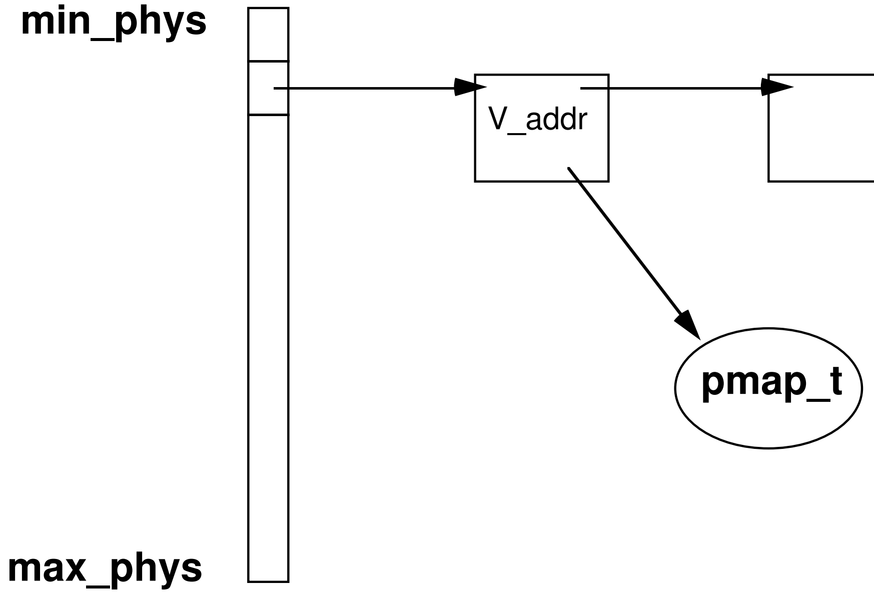
Pmap_remove:

lock [and Shoot]
check if ptepage exists
remove mapping from PV-list
invalidate (our) TLB
zero pte
unlock

The PV-list

Some functions operate on physical addresses

==> Needs a Physical-to-Virtual mapping



Locking

Uniprocessors can get away (almost) without any locking

On MultiP there are three cases:

- operations that only affect the pmap (pmap_extract&co)
  only lock the pmap
- operations that that modify the PV-list use a single, global
  complex lock. pmap_enter&co lock like
  - read_lock the global lock
  - lock the pmap
  - lock the PV-list

  pmap_remove_all&co lock like
  - write_lock the global lock (implies PV-list)
  - lock each pmap as needed

Cache Coherency

You have a cache coherency problem when

- You do DMA in the kernel and the cache does not snoop the bus

- You modify code (kernel or user) and there are separate I & D caches, not kept coherent by the hardware

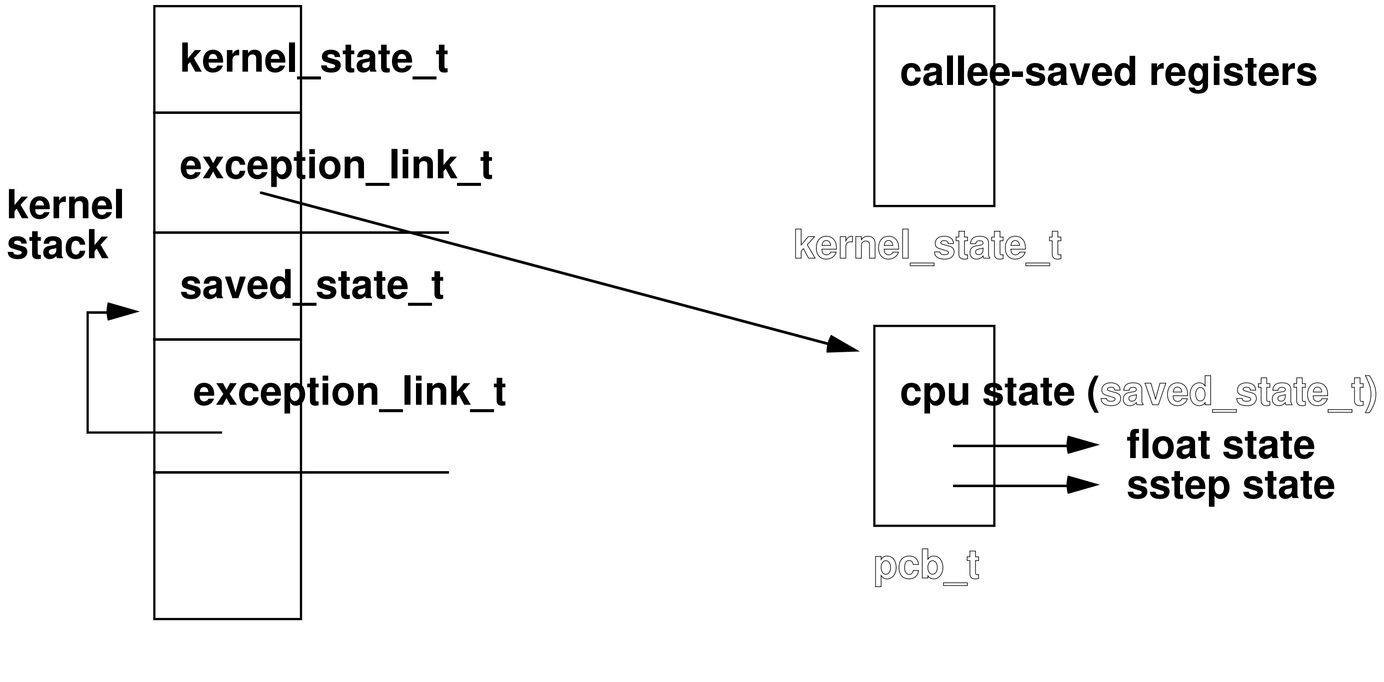
TLB Coherency

You always have a TLB coherency problem on a multiP

===> Shootdown algorithm

..because noone will ever implement snooping TLBs

Context-switching



Primitives: stack_handoff, Switch_context, load_context

Syscall Trapping

- In syscall path, check if syscall number is emulated
- Lookup emulation PC in thread->task->emulation_vector
- Pass along to user the trapping PC (in register if possible, on stack otherwise)
- Return from trap

Interrupts

- Interrupts to users and to kernel are different

- Dispatch them quickly

- Beware of interrupt nesting

Coprocessors

The FPU can effectively be dispatched asynchronousy on many architectures.

- Disable coprocessor access by default
- On trap, save current content, restore proper content
- On context switch, disable FPU access unless this is the current "owner"

Problems on multiprocessors