How to Avoid Flushing Untagged Caches & TLBs

- StrongARM SA-1100 is a high-performance low-power processor.
- Designed for embedded applications.
- Some less attractive features (for multiprocessing):
  - ➜ virtually indexed, virtually tagged caches,
  - ➜ untagged, hardware-loaded TLB.

ARM Page Tables

- Top-level PT entry can:
  - ➜ map a 1MB *section*;
    protection with section granularity,
  - ➜ point to a second-level PT.
- Second-level PT entry can:
  - ➜ map a 64kB *large* page,
  - ➜ map a 4kb *small* page.

  Protection with 1/4 page granularity.
- Each page or section is tagged with a *domain id*

ARM Domains

Additional access-control feature:

- ➜ 16 different domains,
- ➜ domain access control register (DACR) defines accessibility for each:
  - ➜ no access
  - ➜ access according to page permission bits
  - ➜ access irrespective of page permission bits

Context switch (e.g. Linux):

- load *translation table base register* to replace page table
- flush instruction TLB
- flush data TLB
- flush instruction cache
- flush data cache

⇒ High overhead

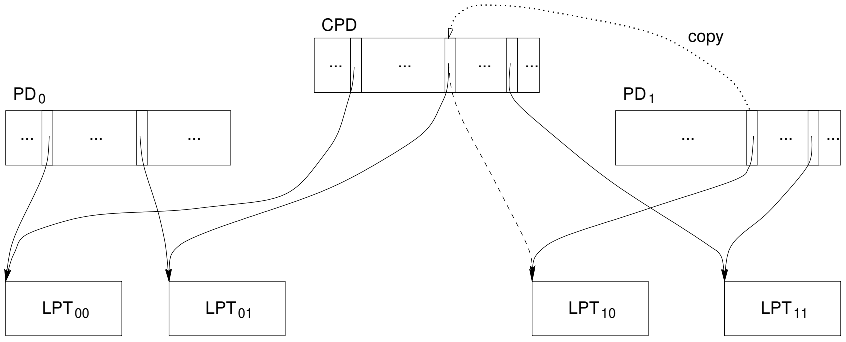
ALTERNATIVE?

- Can avoid flushes if **no overlap of address spaces**
- How????
- Two-part approach [WH00]:
  - ➜ delay flushes as long as possible **in microkernel**
  - ➜ avoid overlaps as much as possible **in server**

DELAYING FLUSHES: IDEA

- Don't change hardware page tables.
- Have translation table base register point to a *cache of top-level page table entries* (CPD).
- Use domain IDs as address-space tag.
- Flush TLB entries and caches when replacing CPD entry.

CACHING PAGE DIRECTORY



CONTEXT SWITCH:

- Set software PT pointer to reference new context's page table
- Load DACR to:
  - ➜ "access according to page permission bits" for new context's domain,
  - ➜ "no access" for other domains
- Go!

No flushes required as long as have separate domain per context.

Cache replacement

- Domains ensure that CPD entries belonging to wrong context raise *domain fault* exception.
- Handled by:
  - ➜ replacing CPD entry
  - ➜ flushing TLBs and caches

  $\Rightarrow$ flush only in case of actual collision.
- Also need to flush if recycling domains.

Avoiding overlap

- Server to allocate process data & stack at unique addresses
  $\Rightarrow$ mini-SASOS (single-address-space OS)
- Need to use separate domains for shared (text). segments
- Aliasing still a problem.

Cool Stuff: Soft Timers [AD99]

Problem: want high-resolution timers

E.g.: Timeouts in L4, prefer high resolution.

- ➜ Kernel has wakeup queues
- ➜ Kernel sets timer interrupt at certain frequency.
- ➜ At each interrupt (tick) checks wakeup queue.
- ➜ Handling interrupts is costly.
- ➜ Handling interrupt pollutes cache.
- ➜ Cannot have very high frequency timer ticks.

What can we do?

Idea: soft timers rather than hardware interrupt

- Check wakeup queues when cost of handling with them is low.
- This is generally the case when in the kernel already.
- Do it at a time where minimal state needs to be saved.
- Suitable times:
  - ➜ just before returning from system call,
  - ➜ at the end of exception handling,
  - ➜ at the end of interrupt handling,
  - ➜ in idle loop.

LIMITATIONS OF SOFT TIMERS:

- No absolute, only *probabilistic* accuracy.
- Actual accuracy depends on frequency of trigger events.
- Can enforce absolute upper bound on delay via timer tick.

THE RETURN OF THE DUMB TERMINAL: SLIM [SLN99]

PROBLEM: HIGH COST-OF-OWNERSHIP OF PCS

- PCs are expensive to maintain/administrate as individual machines.
- Would be cool to have the power of a big iron occasionally.
  ➜ There's still (or again?) some attraction in big central servers.

IDEA: STATELESS THIN-CLIENT ARCHITECTURE

➜ Stateless (other than frame buffer) cheap terminals (graphic displays),
➜ server sends 2D bitmaps,
➜ connected via 100Mbps ethernet.

Supported by very simple, low-level display protocol (5 commands):

① SET: set pixes in rectangle to literal value
② BITMAP: fill rectangle with foreground, background colour
③ FILL: set all pixels to fixed value
④ COPY: copy rectangle
⑤ CSCS: colour-space conversion

Benchmarked GUI apps: photoshop, netscape, framemaker, MPEG, quake

COOL SYSTEM: EROS [SSF99]

IDEA: FAST AND RELIABLE OS

Features:

➜ segregated capabilities,
➜ single-level store,
➜ persistence (via checkpointing),
➜ fast,
➜ mandatory access control,
➜ formal proof of confinement [SW00].

References

[AD99] Mohit Aron and Peter Druschel. Soft timers: Efficient microsecond software timer support for network processing. In *sosp99*, pages 232–246, Kiawah Island, NC, USA, December 1999.

[SLN99] Brian K. Schmidt, Monica S. Lam, and J. Duane Northcutt. The interactive performance of SLIM: a stateless, thin-client architecture. In *Proc. 17th ACM SOSP*, pages 32–47, Kiawah Island, SC, USA, December 1999.

[SSF99] Jonathan S. Shapiro, Jonathan M. Smith, and David J. Farber. Eros: A fast capability system. In *Proc. 17th ACM SOSP*, pages 170–185, Charleston, SC, USA, December 1999.

[SW00] Jonathan S. Shapiro and Samuel Weber. Verifying the EROS confinement mechanism. In *ieeessp*, 2000.

[WH00] Adam Wiggins and Gernot Heiser. Fast address-space switching

on the StrongARM SA-1100 processor. In *Proc. 5th ACAC*, pages 97–104, Canberra, Australia, January 2000.