Hot Topics — Cool Systems

Cool System: EROS [SSF99]

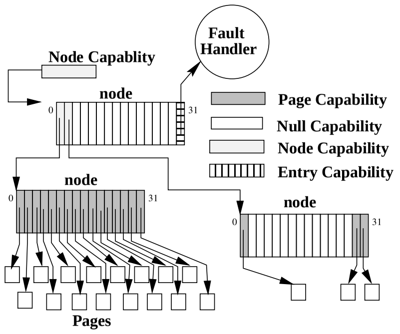
IDEA: FAST, SECURE, RELIABLE OS

Features:

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

EROS is a re-design of KeyKOS [BFF+92].

EROS MEMORY AND ACCESS MANAGEMENT



Clists form page table

EROS ACCESS RIGHTS MANAGEMENT

- Limit propagation and support revocation of rights by:
  - "weak capabilities":
    ➜ reading/writing any cap via a *weak* cap makes it R/O and *weak*
    ➜ can obtain *transitive read-only access*
  - indirection:
    ➜ *Reference monitor* (similar to L4 *chief* mediates cap transfer
    ➜ inserts *forwarding objects* to capabilities
    ➜ implements security policy
    ➜ on change of policy can revoke caps by revoking forwarding object

EROS PERFORMANCE

Benchmark	*Linux-Normalized*	*Speedup*
Pipe Latency	5.66 μs / 8.34 μs	32.3%
Pipe Bandwidth	281 MB/s / 260 MB/s	8.07%
Create Process	0.664 ms / 1.92 ms	65.3%
Ctxt Switch	1.19 μs / 1.26 μs	5.5%
Grow Heap	20.42 μs / 31.74 μs	35.7%
Page Fault	3.67 μs / 687 μs	99.5%
Trivial Syscall	1.6 μs / 0.7 μs	-128%

Note: No data on cost of indirection for MAC.

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

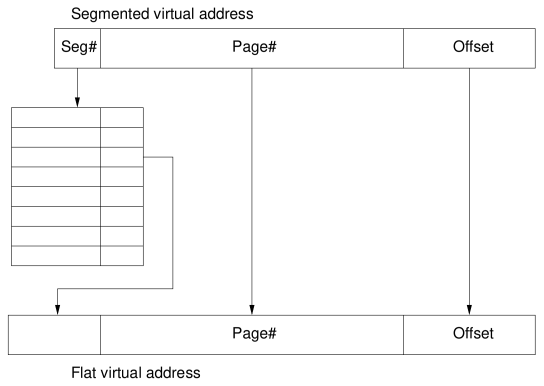
Benchmarked GUI apps: photoshop, netscape, framemaker, MPEG, quake

How to Roll Your Own Tagged TLB and Caches

CONTEXT SWITCH WITH UNTAGGED TLB:

➜ load *translation table base register* to replace page table
➜ flush instruction TLB
➜ flush data TLB
➜ flush instruction cache (if virtual)
➜ flush data cache (if virtual)
⇒ High overhead

SIMULATING TLB TAGS BY SEGMENTATION:



SEGMENT REGISTERS AS ASID TAGS:

- Fill all segment registers with process' ASID
- Problem if flat VAS is only 32-bit (Pentium)
- Partial solution [Lie95]:
  ➜ split AS into *large* (3GB) and *small* area
  ➜ small area split into smaller (128MB) *slots*
  ➜ put small processes into a slot
  ➜ can do fast context switches between one large and all small processes

ON NON-SEGMENTED ARCHITECTURE (STRONGARM)

- StrongARM SA-1100 is a high-performance low-power processor
- Designed for embedded applications
- Some less attractive features (for multitasking):
  ➜ 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

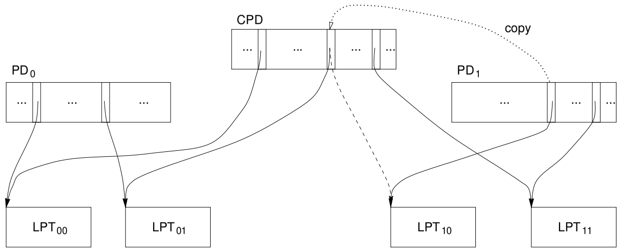
HOW TO AVOID FLUSHING?

- 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.

References

[AD99] Mohit Aron and Peter Druschel. Soft timers: Efficient microsecond software timer support for network processing. In *Proc. 17th SOSP*, pages 232–246, Kiawah Island, NC, USA, Dec 1999.

[BFF+92] Alan C. Bromberger, A. Peri Frantz, William S. Frantz, Ann C. Hardy, Norman Hardy, Charles R. Landau, and Jonathan S. Shapiro. The KeyKOS nanokernel architecture. In *Proc. W. Microkernels & other Kernel Arch.*, pages 95–112, Seattle, WA, USA, Apr 1992.

[Lie95] Jochen Liedtke. Improved address-space switching on Pentium processors by transparently multiplexing user

address spaces. Technical Report 933, GMD SET-RS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany, Nov 1995.

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

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

[SW00] Jonathan S. Shapiro and Samuel Weber. Verifying the EROS confinement mechanism. In *Symp. Security & Privacy*, 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, Jan 2000. IEEE CS Press.