TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science, Institute for System Architecture, Operating Systems Group

Paper Reading Group:

Interaction of Architecture and Operating System Design

T. Anderson, H. Levy, B. Bershad, E. Lazowska

Marcus Völp

Context

- `85 80386 – paged virtual memory
- `89 80486
- **`91 Interaction of Hardware and Operating System Design**
  - **We need IA64-RSEs, the Page Global Extension, tagged TLBs, physically indexed Caches and sysenter!**
- `93 Pentium 5
- `95 – P6 (Pentium Pro, PII, PIII): paging global extension (PGE)
  - (Dr Dobb's Microprocessor Resources)
- `02 Understanding and Improving Operating System Effects in Control Flow Prediction
  - collect separate branch correlation information for user and kernel code
  - use separate branch prediction tables for user and kernel code

- **Why do I present a somehow „obsolete“ paper? - lets see!**

Trends

Hardware Trends (1991):

- move towards simple, directly-executed instruction sets
  - Now: microcode, java instructions on a risc, Transmeta Crusoe (pico-Java + x86)
- open-architecture philosophy
- migration from hardware to software
  - Now: VT, TE, ...
- performance-oriented approach
  - Yes!
  - But are Spec CPU and similar single-application runs the right benchmarks?
  - Is VMEntry /VMExit what we really need?
  - Recent Trend: Once in there it stays forever?

OS Trends (1991)

- fast local communication
  - do we need fast IPC (or can we work around slow one)
- virtual memory
  - Yes!
- distributed / parallel programming
- Microkernels / Unix as just one interface

Motivation

- Simulation overlooked OS influence
  - Asplos
- Unix has driven the design
  - Linux still does
- Strong OS research focus on performance
  - optimization down to where HW is the bottleneck
    - Jochen : IPC 25 cycles + HW Costs
  - “may not perform adequately in the future given current architectural trends”
- Investigate
  - null system call
  - trap
  - page-table entry change
  - context switch

IPC / RPC

- RPC (procedure call semantics)
- null RPC processing time:

		½ Ping = 240000 Cyc (1.6 Ghz thx Björn)	
– Network transfer time	17%	(~ 77500 Cyc.)	~32%
– Interrupt processing	30%	(~2500 Cyc)	~ 1%
– Wakeup receiving thread (UL)	17%	(~80000 ORE -> Client (x2?))	~66%
– Stub processing	29%		
– Checksums	7%	(in HW)	

- IPC call + return / LRPC
  - 2 kernel entries + exits
  - 2 context switches
    - state safe / restore
    - TLB misses
  - Jochen: 25 cycles + HW time for a single path + TLB misses

System Calls and Interrupt Handling

- Mismatch between HW trap and software requirements of system calls

- Kernel entry / exit 4.5 µs 230 cyc.
  Call preparation 3.1 µs
  Call / return to C 8.2 µs 196 / 296 cyc.

- IPC (P4 – short) [thx. Ron]
  - Client marshalling 92 cyc.
  - IPC 2 ways 1600 cyc. (ca. 150 cyc. sysenter + 80 cyc. sysexit)
  - Server unmarshal. 104 cyc.
  - Server marshalling 88 cyc.
  - Client unmarshal. 208 cyc.

- limitations by the hardware (1991):
  - 4 entry write buffer => stalls account to 30% overhead
    (Intel Core 20 Store buffer entries, 32 load buffer)
    [Inside Intel Core Microarchitecture – Hot Chips 2006]
  - data copying – limited by small caches

Virtual Memory

- Virtual memory “mis”-used to implement
  - Copy-on-write relies on quick traps and page permission changes
  - Distributed shared memory
  - Checkpointing
  - Garbage Collection
  - Recoverable Virtual Memory
  - Transaction Locking

- Imprecise Faults - Exposure of processor Pipeline state
  - Rarely found in todays CPUs
  - ARM 1136JF-S
    - MMU generates precise data aborts,
    - watchpoints are imprecise but restartable (reported on an instruction boundary)
    - external data aborts to CP15, Strongly Ordered Memory, PC, CSPR, SWP load part are precise, others are imprecise

- HW reports insufficient information
  - need to decode faulting instruction to determine cause + address
  - still required on ARM

TLBs + Page tables

- !!! WE ALL NEED TAGGED TLBS + PHYSICALLY INDEXED CACHES !!!
- Are multi-level page-tables the right thing to do: GPTs?
- TLB shootdown
  - Itanium: global TLB purge
  - memory based TLBs perform better than processor based TLBs [Patricia Teller – TLB Performance in Multiprocessors (1991)]
  - Why can't PTE caches participate in the cache coherence protocol?
  - No TLB invalidations at all?

Threads and Multiprocessing

- Issue: safe / restore register state

- SPARC – Register Window
- Itanium Register Stack Engine
  - Backing Store Pointer to automatically spill / fill the 96 rotating registers

- Isn't is just sufficient to add thread IDs when we rename registers?
  - yes perhaps we need some lazy spills + fills

How do Operating Systems and Applications behave?

Mach 3.0

Time (sec)	AS Switch	Th Switch	Syscall	Emul Inst.	KTLB Miss	Other Exc	% in OS
1.4	1277	1418	1898	13807	22931	2824	20
80.9	16208	19068	16561	213781	378159	19309	5
99.2	41355	50865	70495	492179	1136756	144122	12
150	128874	144919	160233	1601813	1865436	187804	16
29.9	24589	25830	26904	164436	423607	29796	16
28.8	1723	2211	1308	1406792	12675	3385	18
26.3	1785	3963	1372	1341130	18038	4045	19

Anderson's Conclusion

- Operating systems are decomposed into mircokernel-based systems;
  - When will Linus accept this!
- Architectures have made IPC more costly

- Operating systems are requiring more use of memory management
- Handling MMU events has become more difficult

- Operating systems are moving towards fine grain multithreading
- Architectures are adding more state, making fine grain threads more expensive

my Conclusions

- Processor architectures are moving targets
- Operating system impacts on architectural performance are not negligible

- We can influence hardware developments, provided we can justify the performance improvements?
- How can hardware based mechanisms solve your problem better than the software solution you are proposing?
  - Parallel comparisons are trivial with CAM: OS-level TLB for capspace lookups?
  - What would happen if we implement delayed preemption in the interrupt controller?
    - delay the incoming interrupt and deliver it latest with the timer?
    - increased cost to enter a DP section (~ program an LAPIC register)
  - What to do with instructions like:
    - is this memory locally cached?
    - is the page in the TLB and has the entry sufficient permissions?
    - lock this cacheline / wait until it is unlocked?