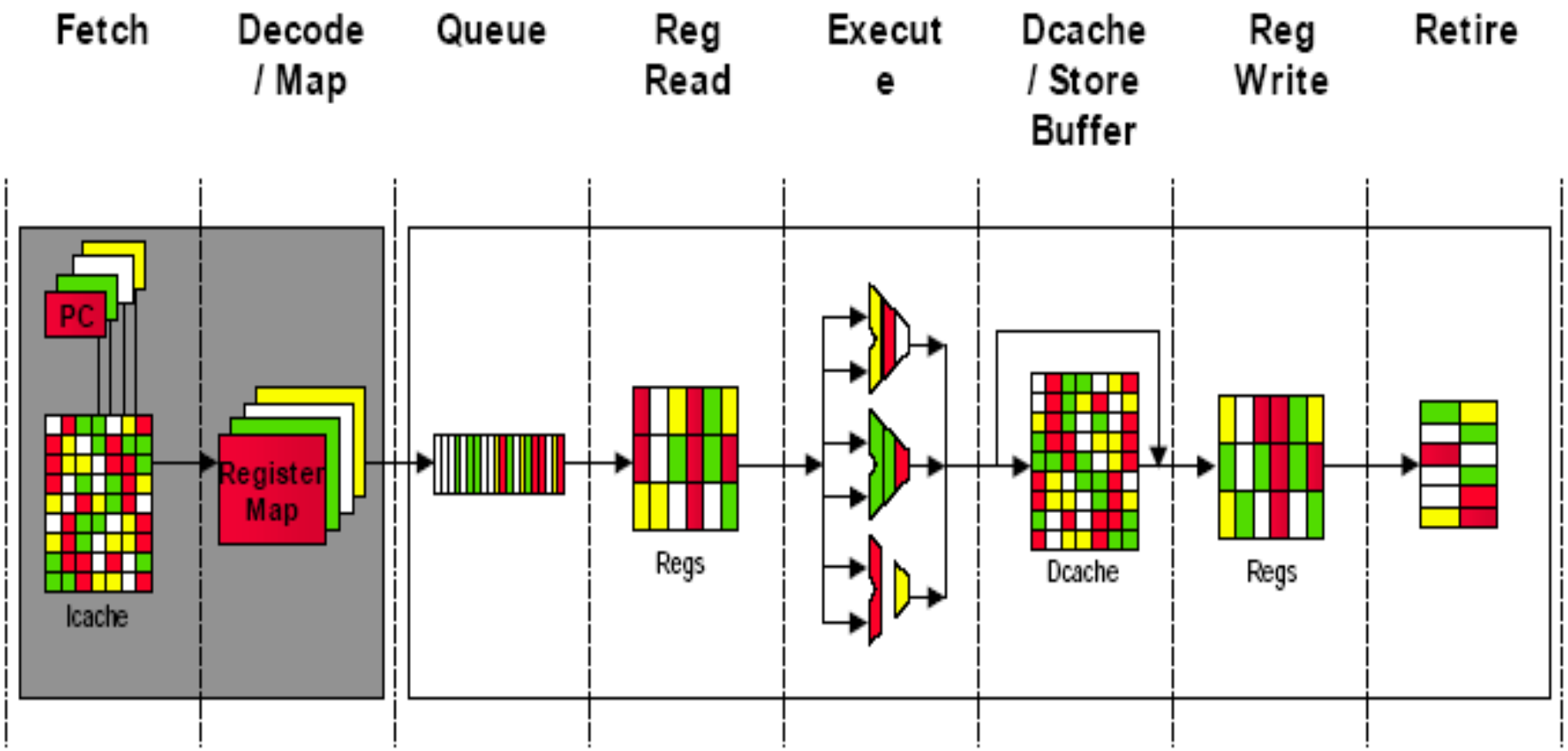
Multithreaded Architectures

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

EE-739: Processor Design

Lecture 13 (4 March 2015) CADSL

SMT Pipeline

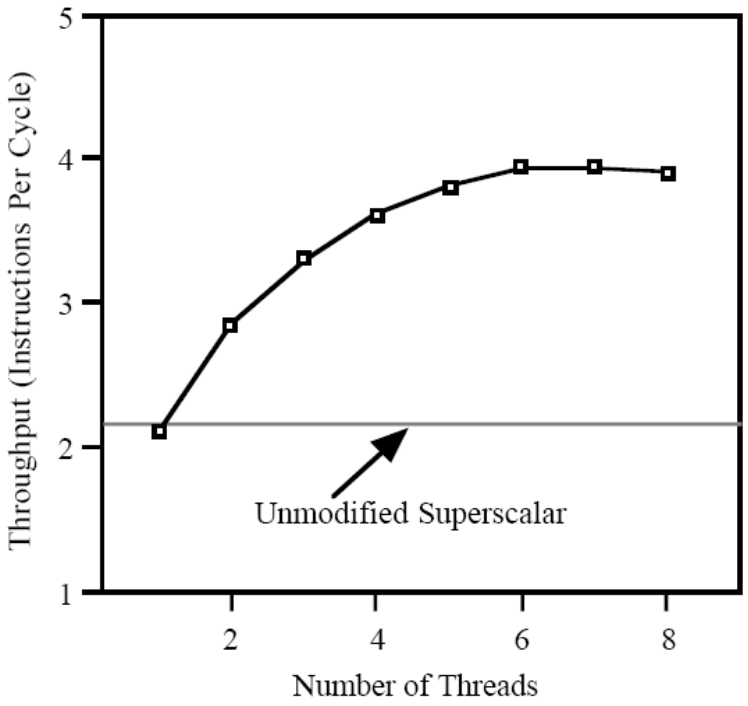


SMT Architecture

- Straightforward extension to conventional superscalar design.
  - multiple program counters and some mechanism by which the fetch unit selects one each cycle,
  - a separate return stack for each thread for predicting subroutine return destinations,
  - per-thread instruction retirement, instruction queue flush, and trap mechanisms,
  - a thread id with each branch target buffer entry to avoid predicting phantom branches, and
  - a larger register file, to support logical registers for all threads plus additional registers for register renaming.
    - The size of the register file affects the pipeline and the scheduling of load-dependent instructions.

SMT Performance

Tullsen '96



Metric	Number of Threads		
	1	4	8
out-of-registers (% of cycles)	3%	7%	3%
I cache miss rate	2.5%	7.8%	14.1%
-misses per thousand instructions	6	17	29
D cache miss rate	3.1%	6.5%	11.3%
-misses per thousand instructions	12	25	43
L2 cache miss rate	17.6%	15.0%	12.5%
-misses per thousand instructions	3	5	9
L3 cache miss rate	55.1%	33.6%	45.4%
-misses per thousand instructions	1	3	4
branch misprediction rate	5.0%	7.4%	9.1%
jump misprediction rate	2.2%	6.4%	12.9%
integer IQ-full (% of cycles)	7%	10%	9%
fp IQ-full (% of cycles)	14%	9%	3%
avg (combined) queue population	25	25	27
wrong-path instructions fetched	24%	7%	7%
wrong-path instructions issued	9%	4%	3%

Fetch Policies

- Basic: Round-robin: RR.2.8 fetching scheme, i.e., in each cycle, two times 8 instructions are fetched in round-robin policy from two different 2 threads,
  - superior to different other schemes like RR.1.8, RR.4.2, and RR.2.4
- Other fetch policies:
  - BRCOUNT scheme gives highest priority to those threads that are least likely to be on a wrong path,
  - MISSCOUNT scheme gives priority to the threads that have the fewest outstanding D-cache misses
  - IQPOSN policy gives lowest priority to the oldest instructions by penalizing those threads with instructions closest to the head of either the integer or the floating-point queue
  - ICOUNT feedback technique gives highest fetch priority to the threads with the fewest instructions in the decode, renaming, and queue pipeline stages

Design Challenges in SMT- Decode

- Primary tasks
  - Identify source operands and destination
  - Resolve dependency
- Instructions from different threads are not dependent
- Tradeoff → Single thread performance

Design Challenges in SMT- Rename

- Allocate physical register
- Map AR to PR
- Makes sense to share logic which maintain the free list of registers
- AR numbers are disjoint across the threads, hence can be partitioned
  - High bandwidth al low cost than multi-porting
- Limits the single thread performance

Design Challenges in SMT- Issue

- Tomasulo's algorithm
- Wakeup and select
- Clearly improve the performance
- Selection
  - Dependent on the instruction from multiple threads
- Wakeup
  - Limited to intrathread interaction
  - Make sense to partition the issue window
- Limit the performance of single thread

Design Challenges in SMT- Execute

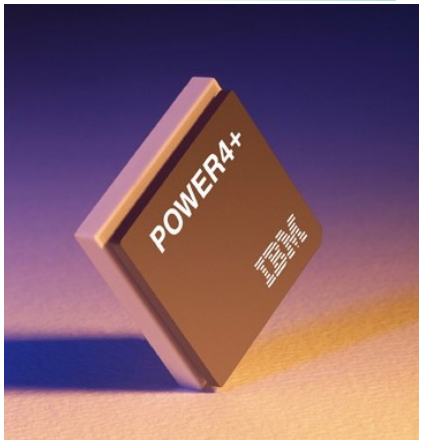
- Clearly improve the performance
- Bypass network
- Memory
  - Separate LS queue

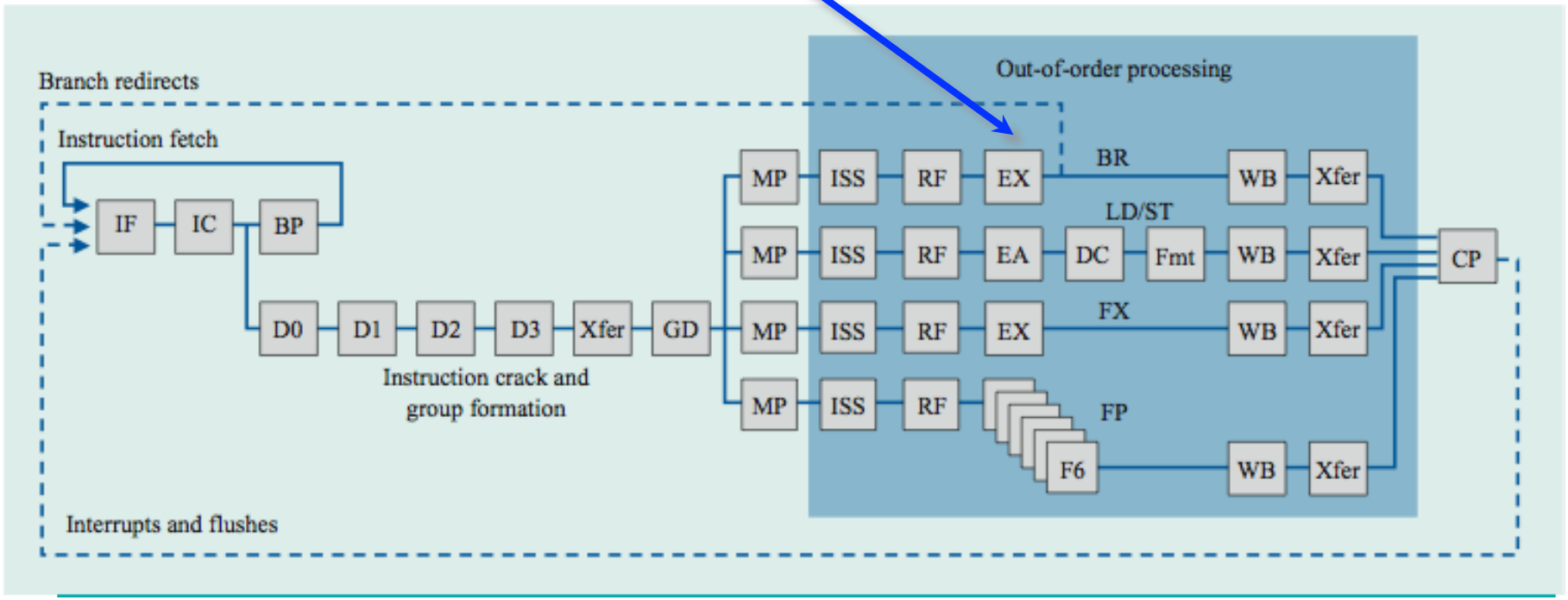
Commercial Machines with MT Support

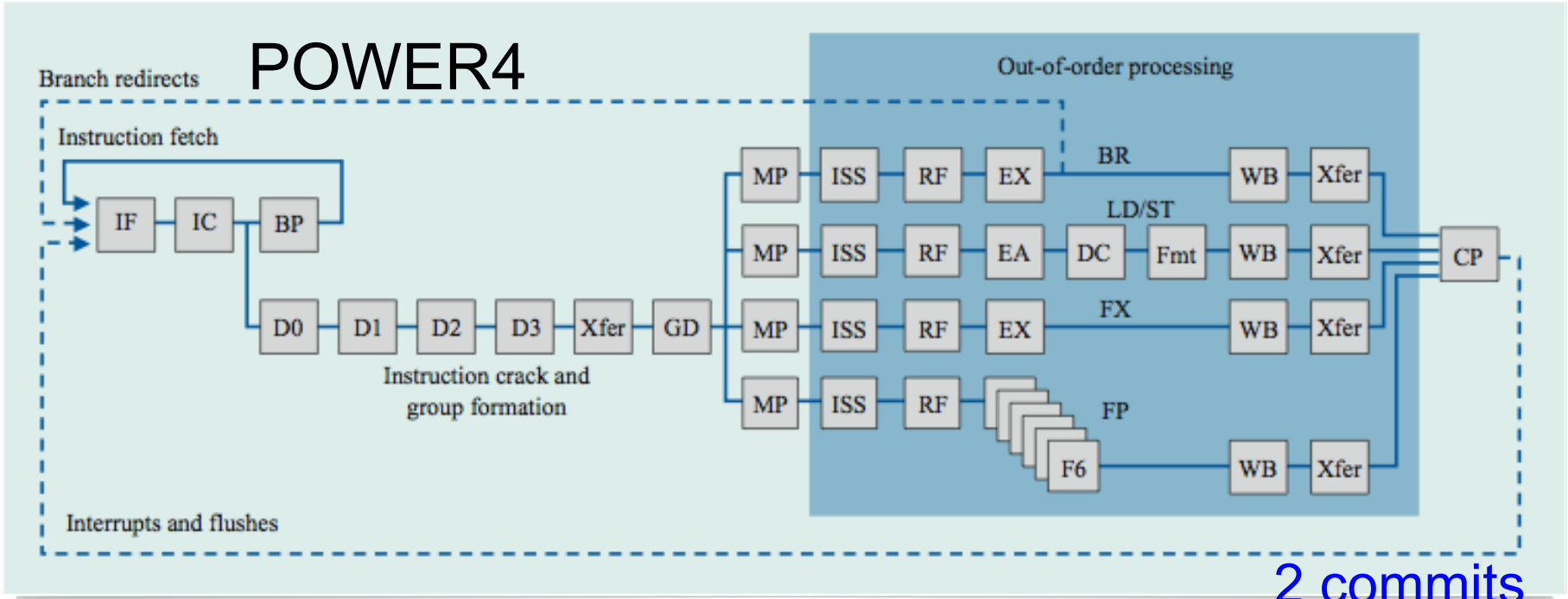
- Intel Hyperthreding (HT)
  - Dual threads
  - Pentium 4, XEON
- Sun CoolThreads
  - UltraSPARC T1
  - 4-threads per core
- IBM
  - POWER5

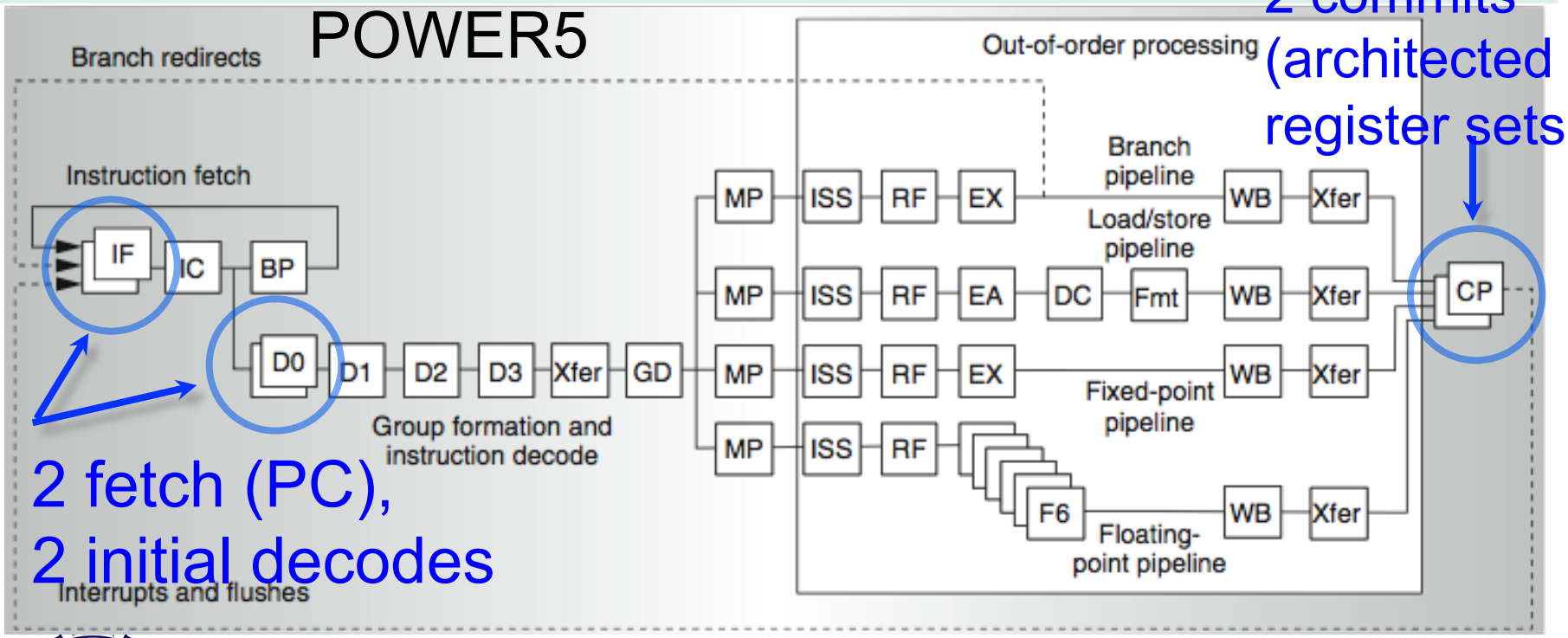
IBM POWER4

Single-threaded predecessor to POWER5. 8 execution units in out-of-order engine, each may issue an instruction each cycle.





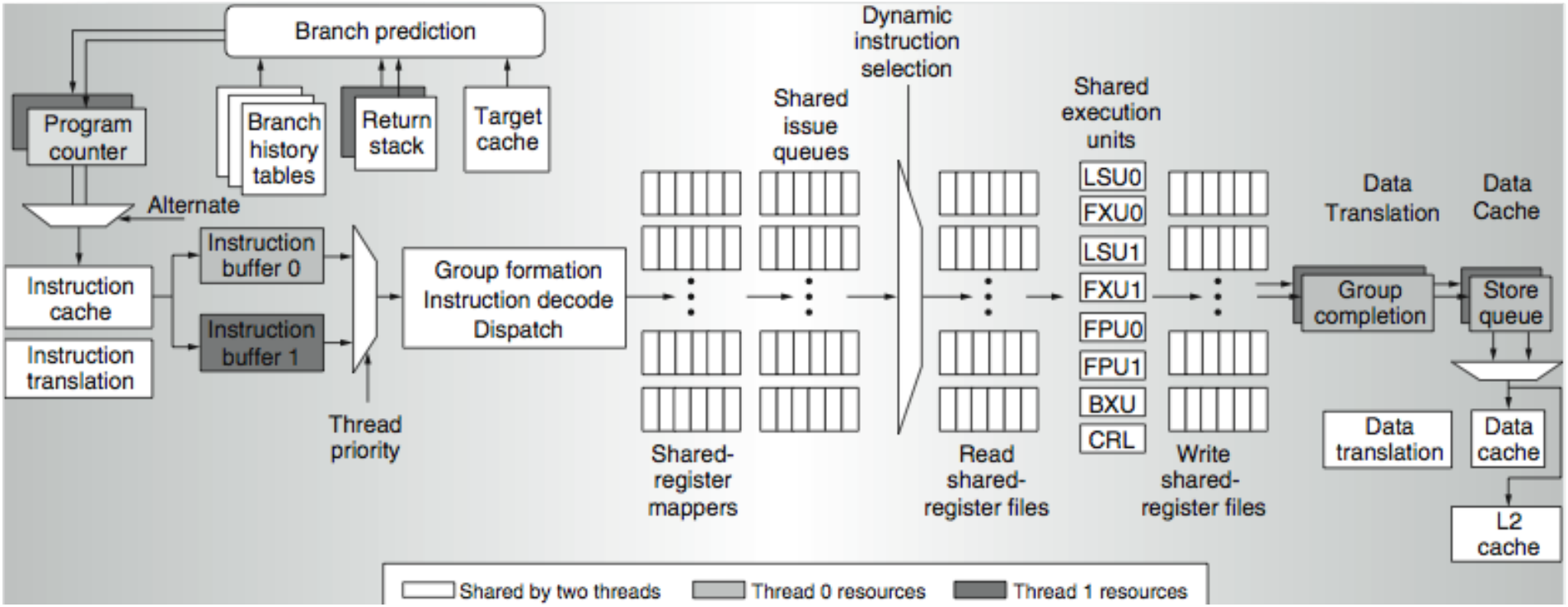




2 commits (architected register sets)

2 fetch (PC),
2 initial decodes

POWER5 data flow ...



Why only 2 threads? With 4, one of the shared resources (physical registers, cache, memory bandwidth) would be prone to bottleneck

Changes in POWER5 to support SMT

- Increased associativity of L1 instruction cache and the instruction address translation buffers
- Added per thread load and store queues
- Increased size of the L2 and L3 caches
- Added separate instruction prefetch and buffering per thread
- Increased the number of virtual registers from 152 to 240
- Increased the size of several issue queues
- The POWER5 core is about 24% larger than the POWER4 core because of the addition of SMT support

IBM Power5

http://www.research.ibm.com/journal/rd/494/mathis.pdf

Table 1 Workloads selected for the study.

Workload	*Computation type*	*SMT gain (%)*
Sentence passing	Integer	41.2
Data compression	Integer	38.6
Programming language	Integer	26.3
3D Multi-grid Solver	Floating-point	21.6
Circuit Routing	Integer	19.8
Seismic Wave Simulation	Floating-point	15.3
Object-oriented Database	Integer	12.5
Neural Network	Floating-point	11.2

IBM Power5

http://www.research.ibm.com/journal/rd/494/mathis.pdf

Table 5 Sources and percentages of data from each source by workload and mode.

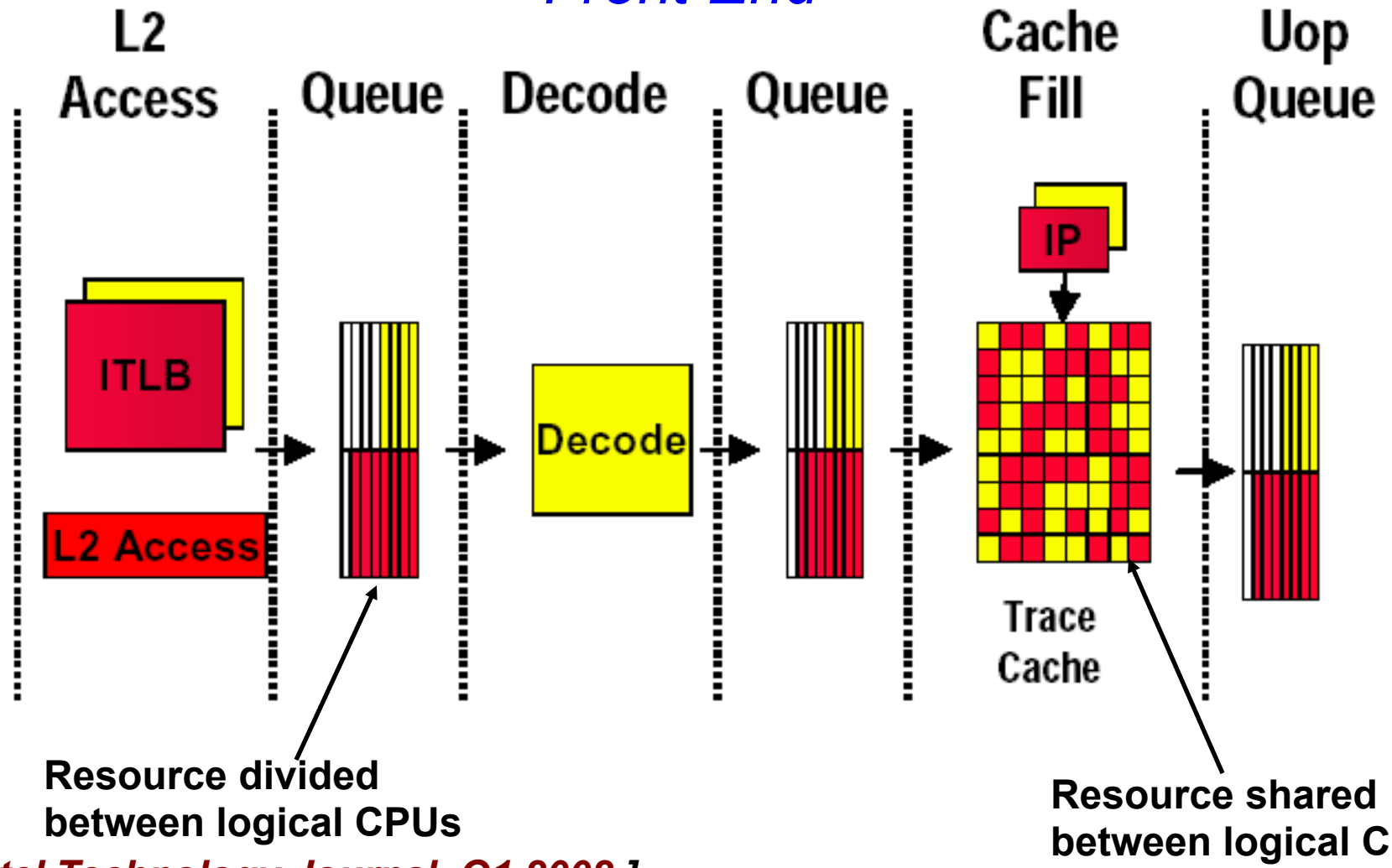
Workload	*Mode*	*Data source*		
		L2 cache (%)	*L3 cache (%)*	*Main memory (%)*
Sentence parsing	ST	94	6	0
Sentence parsing	SMT	92	8	0
Data compression	ST	100	0	0
Data compression	SMT	100	0	0
Programming language	ST	94	5	0
Programming language	SMT	97	3	0
3D Multi-grid Solver	ST	95	2	3
3D Multi-grid Solver	SMT	88	6	6
Circuit Routing	ST	79	20	1
Circuit Routing	SMT	72	27	1
Seismic Wave Simulation	ST	86	6	8
Seismic Wave Simulation	SMT	82	4	14
Object-oriented Database	ST	85	14	1
Object-oriented Database	SMT	88	11	1
Neural Network	ST	50	50	0
Neural Network	SMT	49	51	0

Pentium-4 Hyperthreading (2002)

- First commercial SMT design (2-way SMT)
  - Hyperthreading == SMT
- Logical processors share nearly all resources of the physical processor
  - Caches, execution units, branch predictors
- Die area overhead of hyperthreading ~ 5%
- When one logical processor is stalled, the other can make progress
  - No logical processor can use all entries in queues when two threads are active
- Processor running only one active software thread runs at approximately same speed with or without hyperthreading

Pentium-4 Hyperthreading

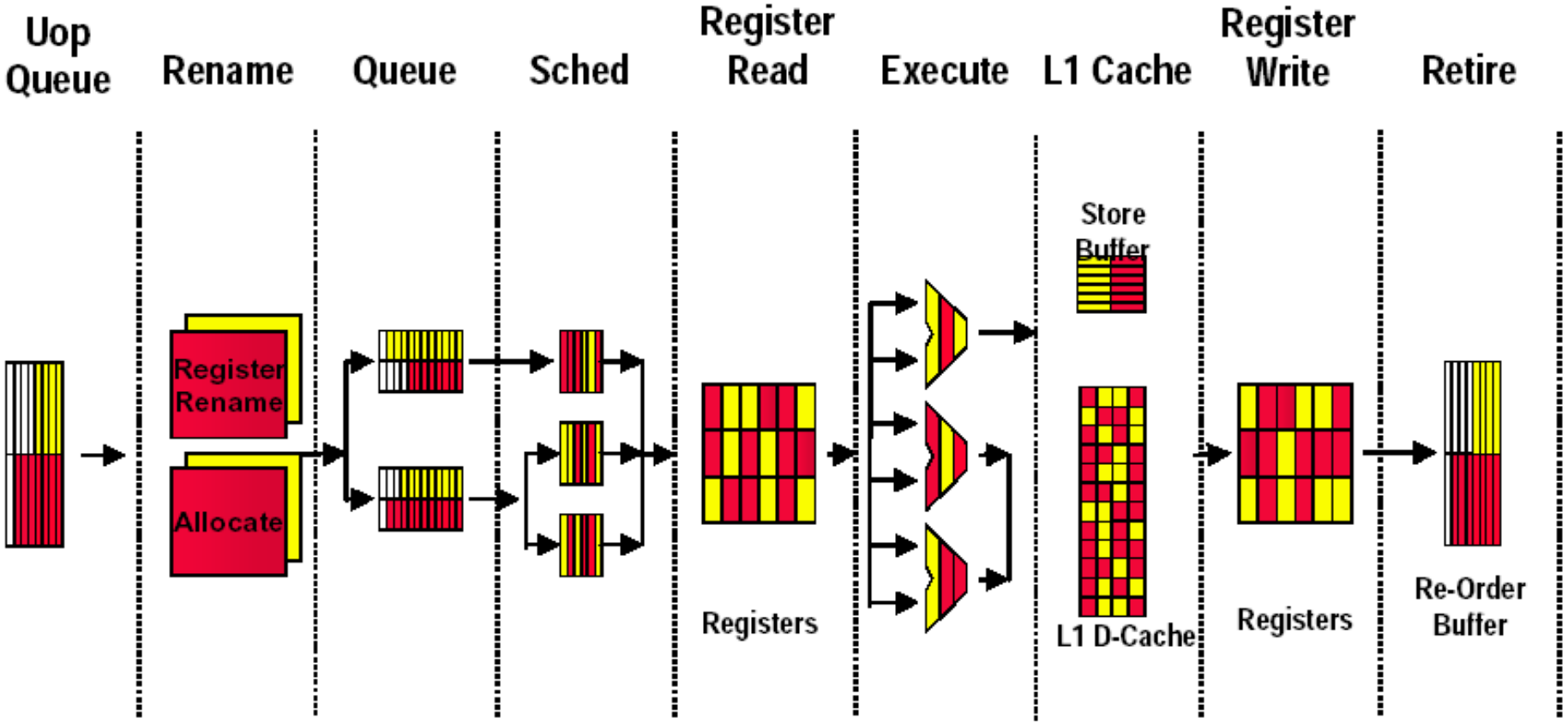
Front End



[Intel Technology Journal, Q1 2002]

Pentium-4 Hyperthreading

Execution Pipeline



[Intel Technology Journal, Q1 2002]

Initial Performance of SMT

- P4 Extreme Edition SMT yields 1.01 speedup for SPECint_rate benchmark and 1.07 for SPECfp_rate
  - Pentium 4 is dual threaded SMT
  - SPECRate requires that each SPEC benchmark be run against a vendor-selected number of copies of the same benchmark
- Running on P4 each of 26 SPEC benchmarks paired with every other (26^2 runs) speed-ups from 0.90 to 1.58; average was 1.20
- POWER5, 8 processor server 1.23 faster for SPECint_rate with SMT, 1.16 faster for SPECfp_rate
- POWER5 running 2 copies of each app speedup between 0.89 and 1.41
  - Most gained some
  - FP apps had most cache conflicts and least gains

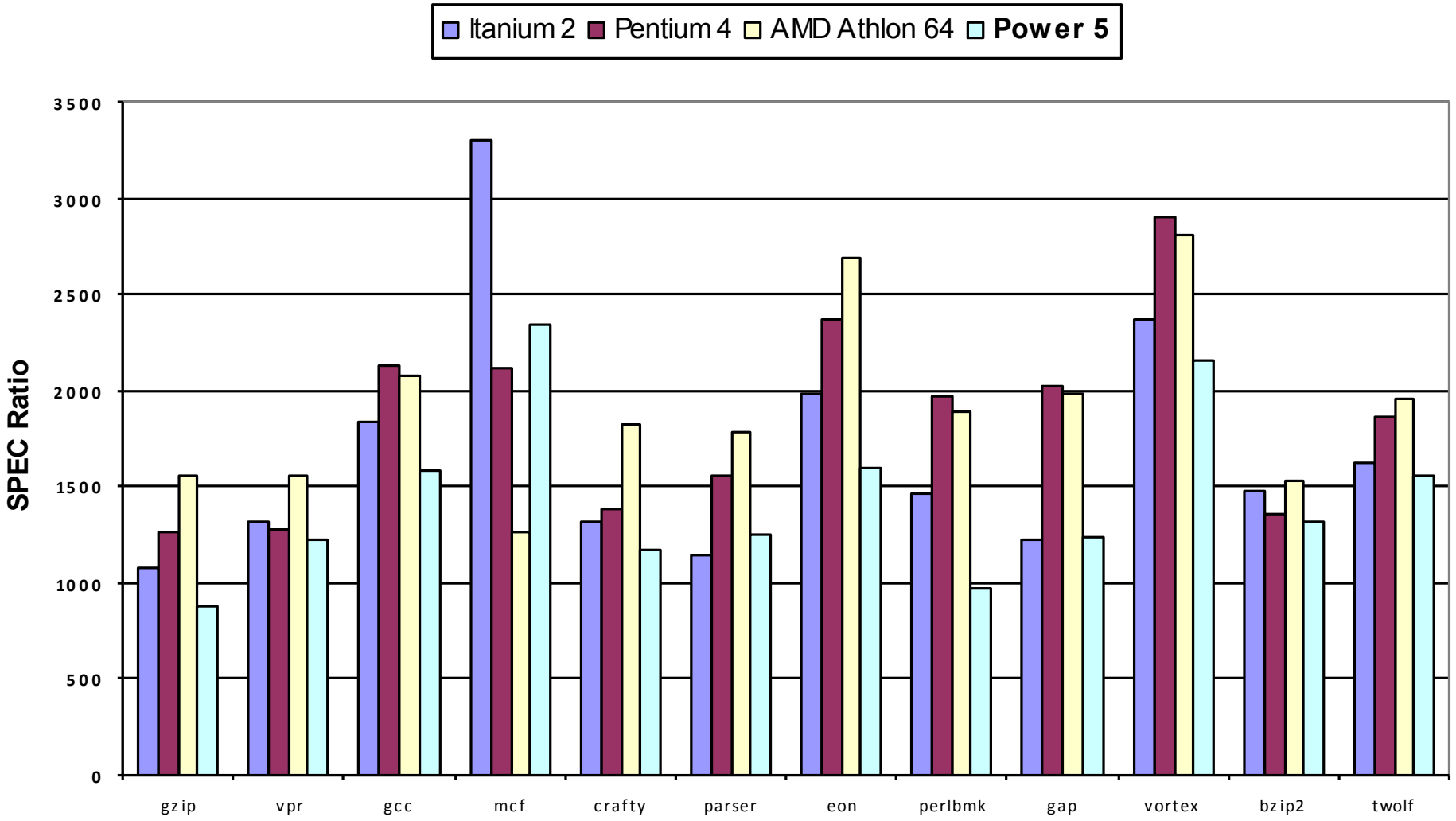
Initial Performance of SMT

- P4 Extreme Edition SMT yields 1.01 speedup for SPECint_rate benchmark and 1.07 for SPECfp_rate
  - Pentium 4 is dual threaded SMT
  - SPECRate requires that each SPEC benchmark be run against a vendor-selected number of copies of the same benchmark
- Running on P4 each of 26 SPEC benchmarks paired with every other (26^2 runs) speed-ups from 0.90 to 1.58; average was 1.20
- POWER5, 8 processor server 1.23 faster for SPECint_rate with SMT, 1.16 faster for SPECfp_rate
- POWER5 running 2 copies of each app speedup between 0.89 and 1.41
  - Most gained some
  - FP apps had most cache conflicts and least gains

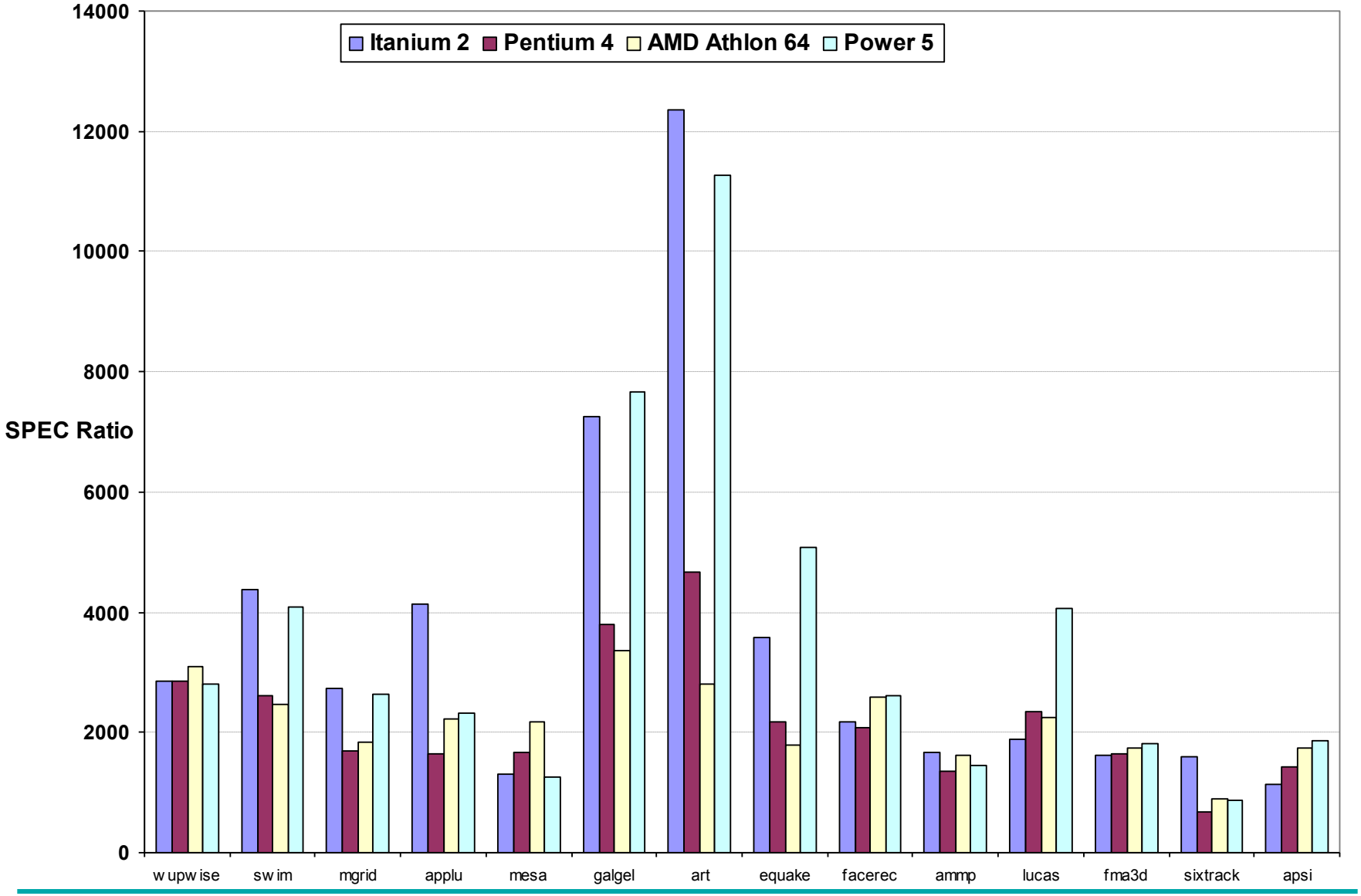
Head to Head ILP competition

Processor	Micro architecture	Fetch / Issue / Execute	FU	Clock Rate (GHz)	Transis-tors Die size	Power
Intel Pentium 4 Extreme	Speculative dynamically scheduled; deeply pipelined; SMT	3/3/4	7 int. 1 FP	3.8	125 M 122 mm^2	115 W
AMD Athlon 64 FX-57	Speculative dynamically scheduled	3/3/4	6 int. 3 FP	2.8	114 M 115 mm^2	104 W
IBM POWER5 (1 CPU only)	Speculative dynamically scheduled; SMT; 2 CPU cores/chip	8/4/8	6 int. 2 FP	1.9	200 M 300 mm^2 (est.)	80W (est.)
Intel Itanium 2	Statically scheduled VLIW-style	6/5/11	9 int. 2 FP	1.6	592 M 423 mm^2	130 W

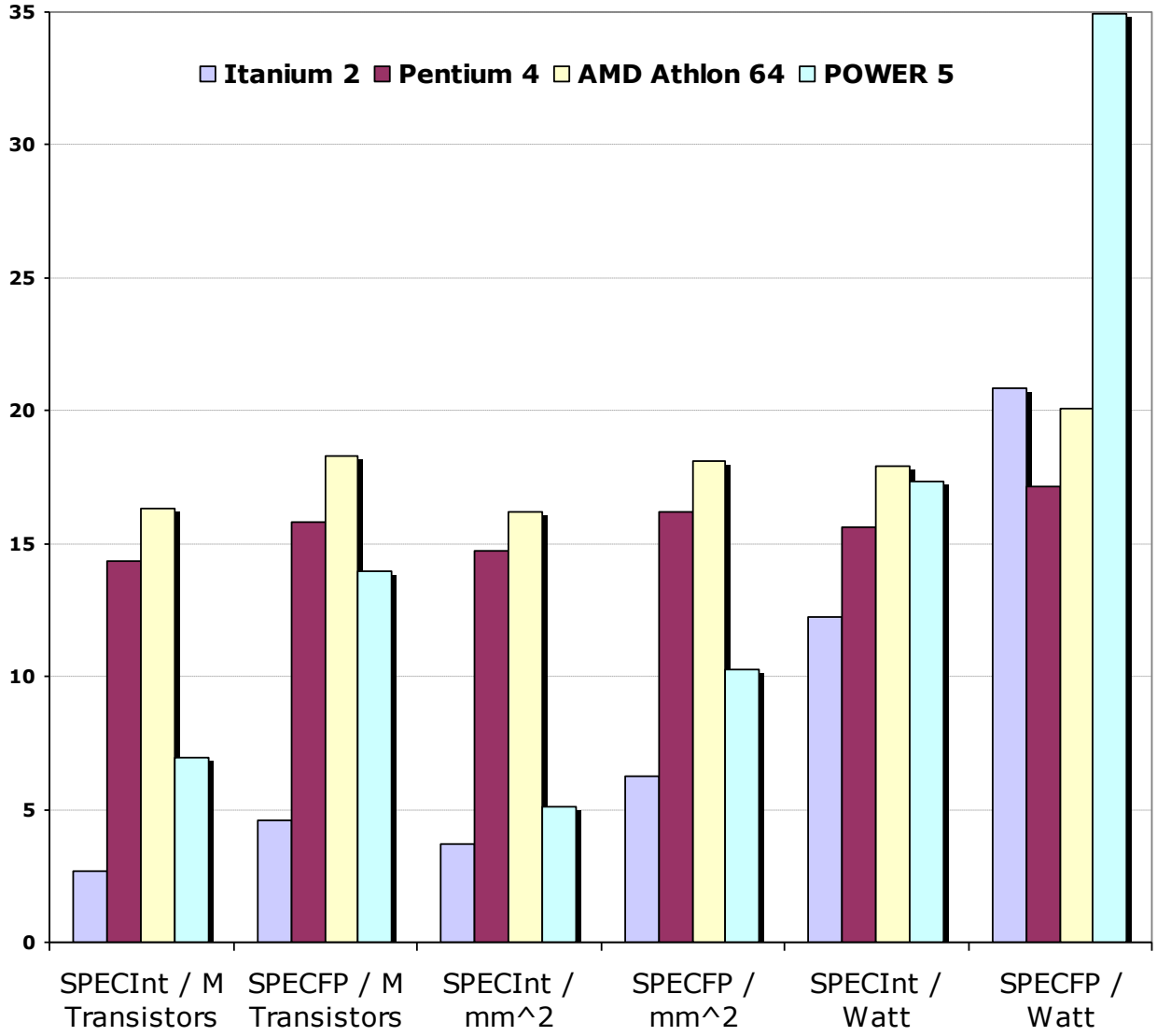
Performance on SPECint2000



Performance on SPECfp2000



Normalized Performance: Efficiency



Rank	Itanium 2	Pentium 4	Athlon	Power 5
Int/Trans	4	2	1	3
FP/Trans	4	2	1	3
Int/area	4	2	1	3
FP/area	4	2	1	3
Int/Watt	4	3	1	2
FP/Watt	2	4	3	1

No Silver Bullet for ILP

- No obvious over all leader in performance
- The AMD Athlon leads on SPECInt performance followed by the P4, Itanium 2, and POWER5
- Itanium 2 and POWER5, which perform similarly on SPECFP, clearly dominate the Athlon and P4 on SPECFP
- Itanium 2 is the most inefficient processor both for FP and integer code for all but one efficiency measure (SPECFP/Watt)
- Athlon and P4 both make good use of transistors and area in terms of efficiency
- IBM POWER5 is the most effective user of energy on SPECfp and essentially tied on SPECint

Limits to ILP

- Doubling issue rates above today's 3-6 instructions per clock, say to 6 to 12 instructions, probably requires a processor to
  - issue 3 or 4 data memory accesses per cycle,
  - resolve 2 or 3 branches per cycle,
  - rename and access more than 20 registers per cycle, and
  - fetch 12 to 24 instructions per cycle.
- The complexities of implementing these capabilities is likely to mean sacrifices in the maximum clock rate
  - E.g, widest issue processor is the Itanium 2, but it also has the slowest clock rate, despite the fact that it consumes the most power!

Limits to ILP

- Most techniques for increasing performance increase power consumption
- The key question is whether a technique is *energy efficient*: does it increase power consumption faster than it increases performance?
- Multiple issue processors techniques all are energy inefficient:
  1. Issuing multiple instructions incurs some overhead in logic that grows faster than the issue rate grows
  2. Growing gap between peak issue rates and sustained performance
- Number of transistors switching = f(peak issue rate), and performance = f(sustained rate),
  growing gap between peak and sustained performance
  → increasing energy per unit of performance

OS Code Vs. User Code

- Operating systems are usually huge programs that can overwhelm the cache and TLB due to code and data size.
- Operating systems may impact branch prediction performance, because of frequent branches and infrequent loops.
- OS execution is often brief and intermittent, invoked by interrupts, exceptions, or system calls, and can cause the replacement of useful cache, TLB and branch prediction state for little or no benefit.
- The OS may perform spin-waiting, explicit cache/TLB invalidation, and other operations not common in user-mode code.

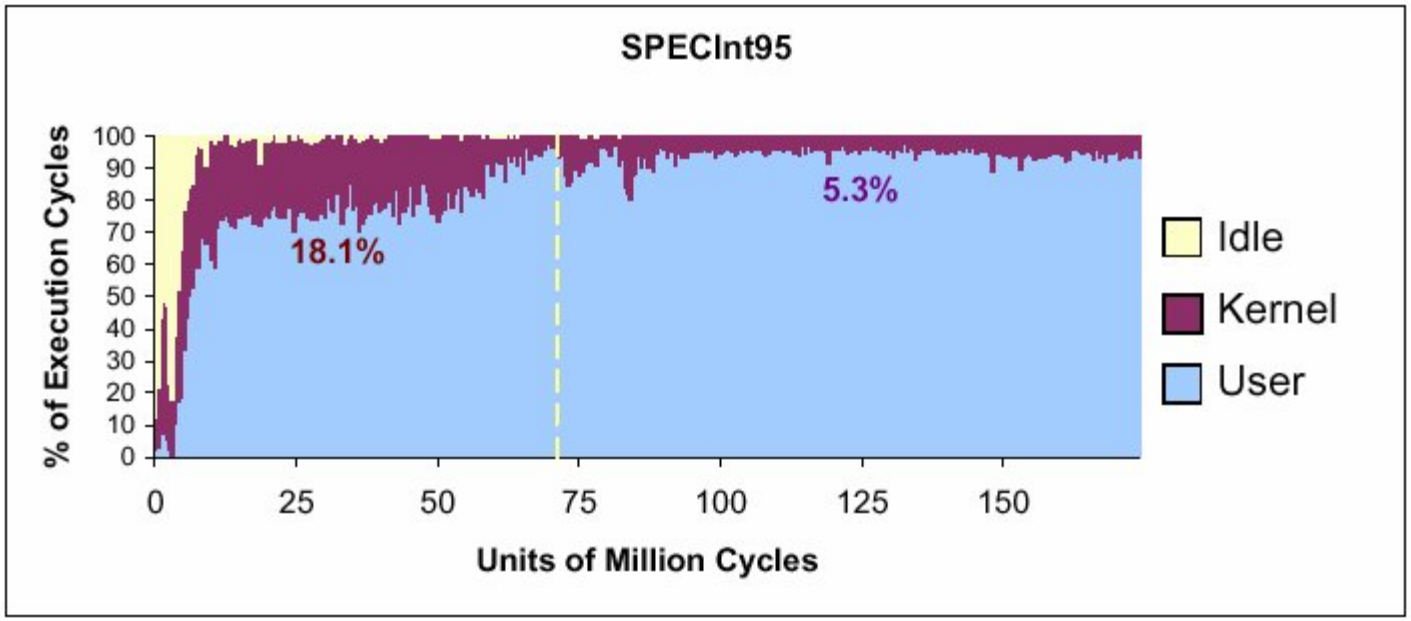
OS Modifications for SMT

Only required modifications considered not OS optimizations for SMT:

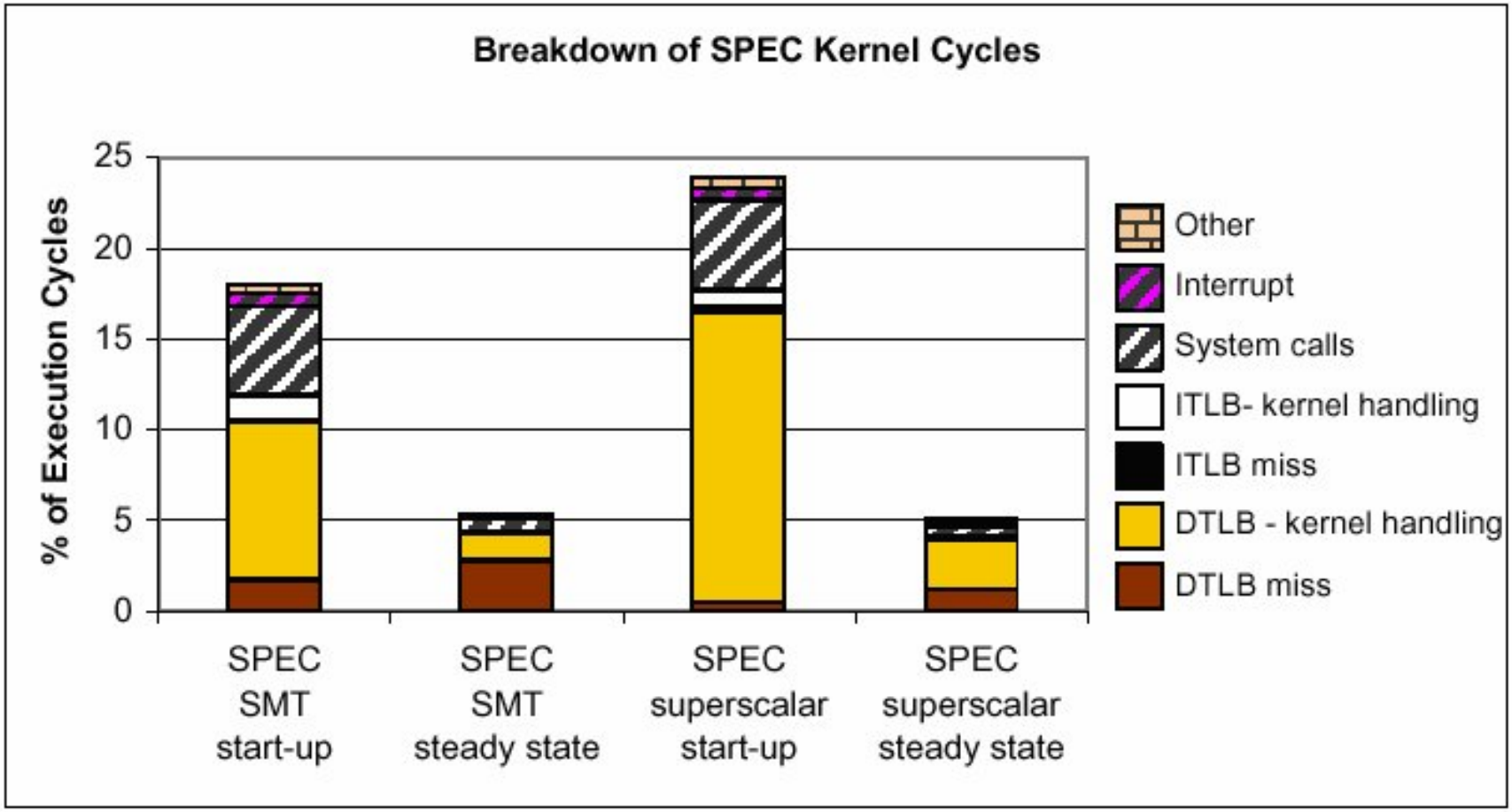
- OS task scheduler must support multiple threads in running status:
  - Shared-memory multiprocessor (SMP) aware OS (including Digital Unix) has this ability but each thread runs on a different CPU in SMP systems.
  - An SMT processor reports to such an OS as multiple shared memory CPUs.
- TLB-related code must be modified:
  - Mutual exclusion support to access to address space number (ASN) tags of the TLB by multiple threads simultaneously.
  - Modified ASN assignment to account for the presence of multiple threads.
  - Internal CPU registers used to modify TLB entries replicated per context.
- No OS changes required to account for the shared L1 cache of SMT vs. the non shared L1 for SMP.

SPECInt Workload Execution Cycle Breakdown

- Percentage of execution cycles for OS Kernel instructions:
  - During program startup: 18%, mostly due to data TLB misses.
  - Steady state: 5% still dominated by TLB misses.



Breakdown of kernel time for SPECInt95



SPECInt95 Dynamic Instruction Mix

Instruction Type	Program Start-up			Steady State		
	User	Kernel	Overall	User	Kernel	Overall
Load	19.5	16.5 (51%)	19.2 (5%)	20.0	12.2 (35%)	19.7 (1%)
Store	12.3	19.0 (57%)	13.1 (10%)	9.6	11.8 (68%)	9.7 (3%)
Branch	15.1	15.9	15.3	14.8	15.0	14.9
Conditional	(64%) 65.9	(56%) 65.3	(63%) 65.8	(56%) 68.3	(26%) 59.9	(54%) 68.0
Unconditional	19.5	14.1	18.8	18.3	6.5	17.8
Indirect Jump	14.7	11.7	14.3	13.3	5.5	13.0
PAL call/return	.01	8.9	1.1	.01	28.1	1.2
Total	100.0	100.0	100.0	100.0	100.0	100.0
Remaining Integer	50.0	48.6	49.7	53.3	61.0	53.5
Floating Point	3.1	0.0	2.7	2.3	0.0	2.2

- Percentage of dynamic instructions in the SPECInt workload by instruction type.
- The percentages in parenthesis for memory operations represent the proportion of loads and stores that are to physical addresses.
- A percentage breakdown of branch instructions is also included.
- For conditional branches, the number in parenthesis represents the percentage of conditional branches that are taken.

SPECInt95 Total Miss rates & Distribution of Misses

	Miss Percentages									
	Branch Target Buffer		L1 Instruction Cache		L1 Data Cache		L2 Cache		Data TLB	
	User	Kernel	User	Kernel	User	Kernel	User	Kernel	User	Kernel
Total miss rate	30.5	75.2	1.8	8.4	3.2	18.8	0.9	10.5	0.5	3.2
Cause of misses	**Percentage of Misses Due to Conflicts (sums to 100%)**									
Intrathread conflicts	51.0	**4.9**	6.5	**.2**	14.6	**.8**	34.7	**1.2**	17.5	**.2**
Interthread conflicts	39.5	**1.1**	33.7	**.2**	59.5	**5.2**	18.9	**.7**	64.5	**.8**
User-kernel conflicts	**1.8**	**1.7**	**4.6**	**3.2**	**5.7**	**6.6**	**6.2**	**.9**	**8.2**	**8.8**
Invalidation by the OS			**40.9**	**10.7**						
Compulsory			.01	.01	7.3	.3	.5	36.6		

- The miss categories are percentages of all user and kernel misses.
- Bold entries signify kernel-induced interference.
- User-kernel conflicts are misses in which the user thread conflicted with some type of kernel activity (the kernel executing on behalf of this user thread, some other user thread, a kernel thread, or an interrupt).

Metrics for SPECInt95 with and without the Operating System for both SMT and Superscalar.

Metric	SMT			Superscalar		
	SPEC only	SPEC+OS	Change	SPEC only	SPEC+OS	Change
IPC	5.9	5.6	-5%	3.0	2.6	-15%
Average # fetchable contexts	7.7	7.1	-8%	1.0	0.8	-20%
Branch misprediction rate (%)	8.1	9.3	15%	5.1	5.0	-2%
Instructions squashed (% of instructions fetched)	15.1	18.2	21%	31.8	32.3	2%
L1 Icache miss rate (%)	1.0	2.0	190%	0.1	1.3	1300%
L1 Dcache miss rate (%)	3.2	3.6	15%	0.6	0.5	-15%
L2 miss rate (%)	1.1	1.4	27%	1.0	1.8	72%
ITLB miss rate (%)	0.0	0.0		0.0	0.0	
DTLB miss rate (%)	0.4	0.6	36%	0.04	0.05	25%

- The maximum issue for integer programs is 6 instructions on the 8-wide SMT, because there are only 6 integer units.

SMT and Memory

- Large demands on memory
- SMT require more bandwidth from primary cache (MT allows more load and store)
- Complex MESI(modified , exclusive, shared and invalid) cache-coherence protocol

Thank You