Fair CPU Time Accounting in CMP+SMT Processors

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

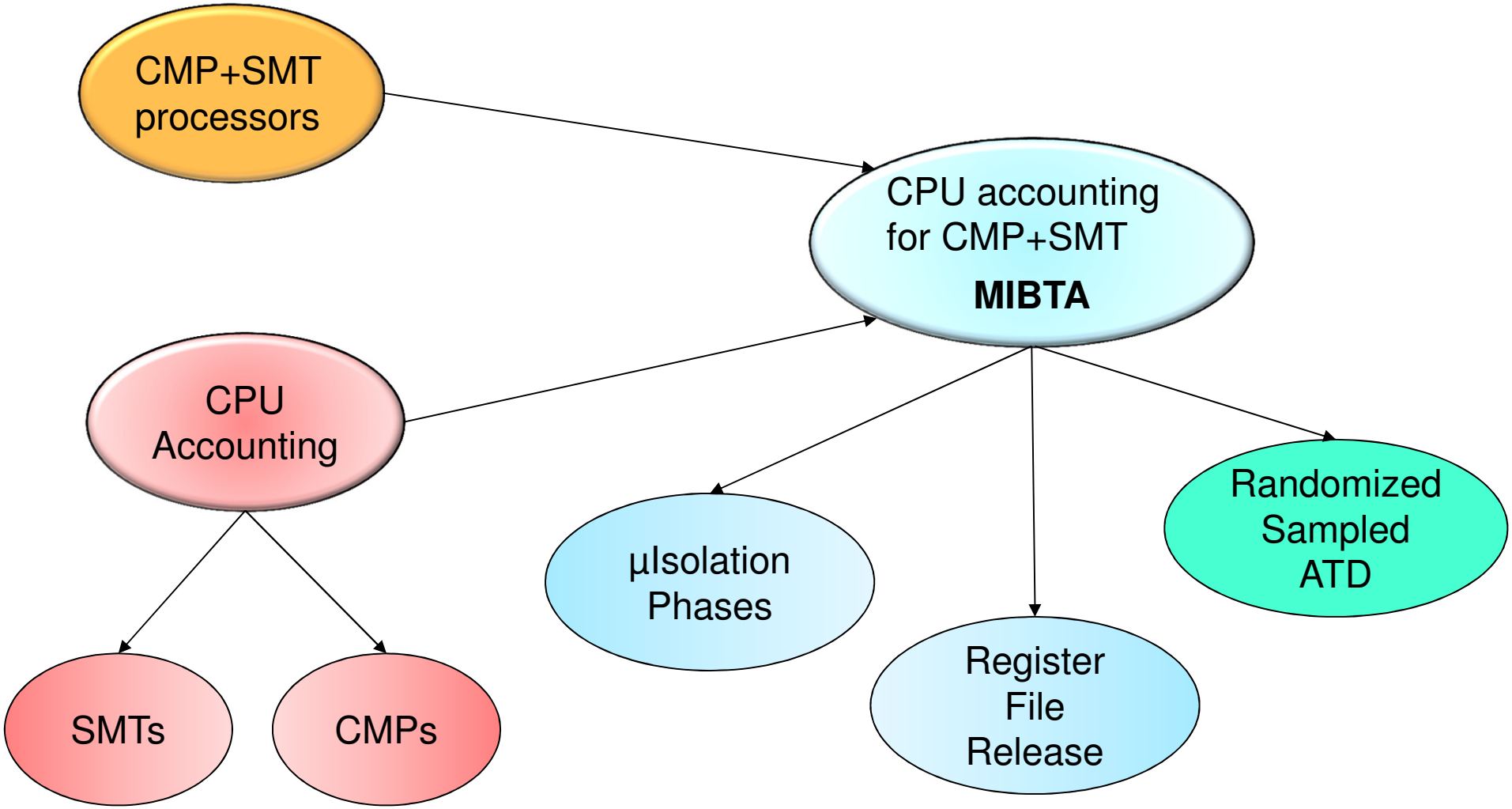
Carlos Luque (UPC/BSC)
Miquel Moreto (ICSI/UPC/BSC)
Francisco J. Cazorla (BSC/IIIA-CISC)
Mateo Valero (UPC/BSC)

Francisco J. Cazorla
Director of the CAOS research group
at BSC (www.bsc.es/caos)

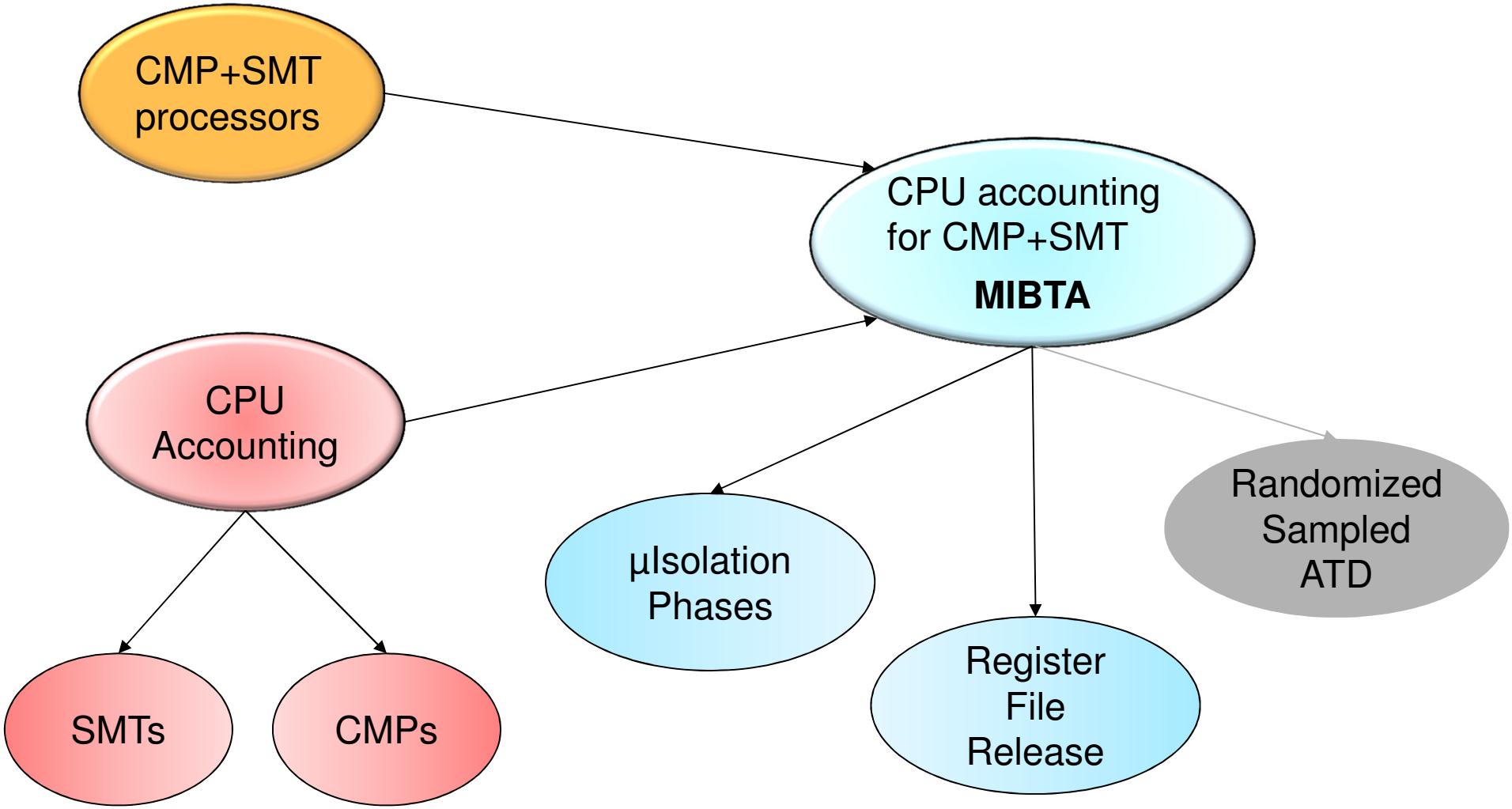
8th HIPEAC
Berlin, Germany
21st January 2013

HiPEAC COMPILATION ARCHITECTURE

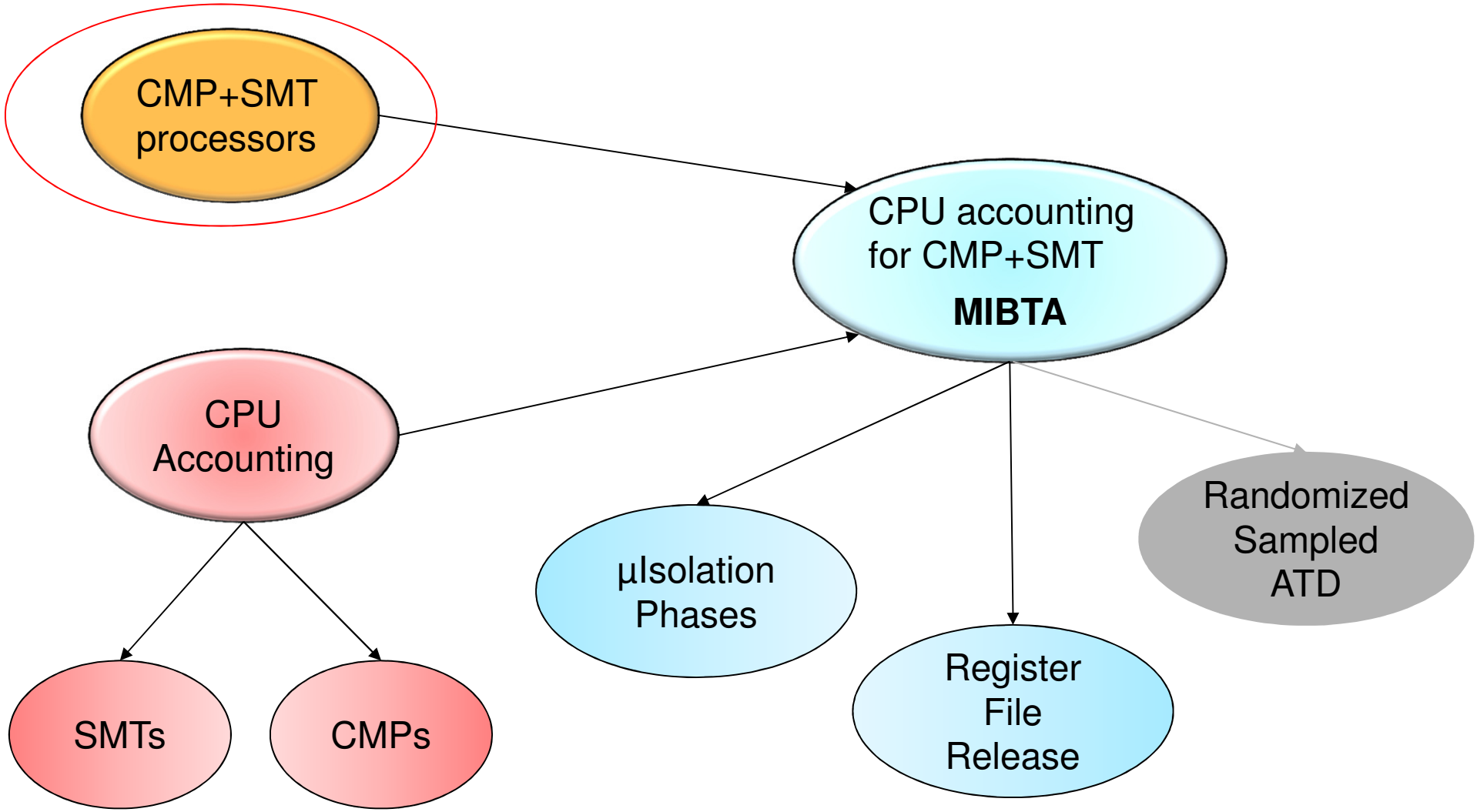
Outline


CMP+SMT processors
CPU accounting for CMP+SMT
MIBTA
CPU Accounting
µIsolation Phases
Randomized Sampled ATD
Register File Release
SMTs
CMPs

Outline


CMP+SMT processors
CPU accounting for CMP+SMT
MIBTA
CPU Accounting
µIsolation Phases
Randomized Sampled ATD
Register File Release
SMTs
CMPs

Outline

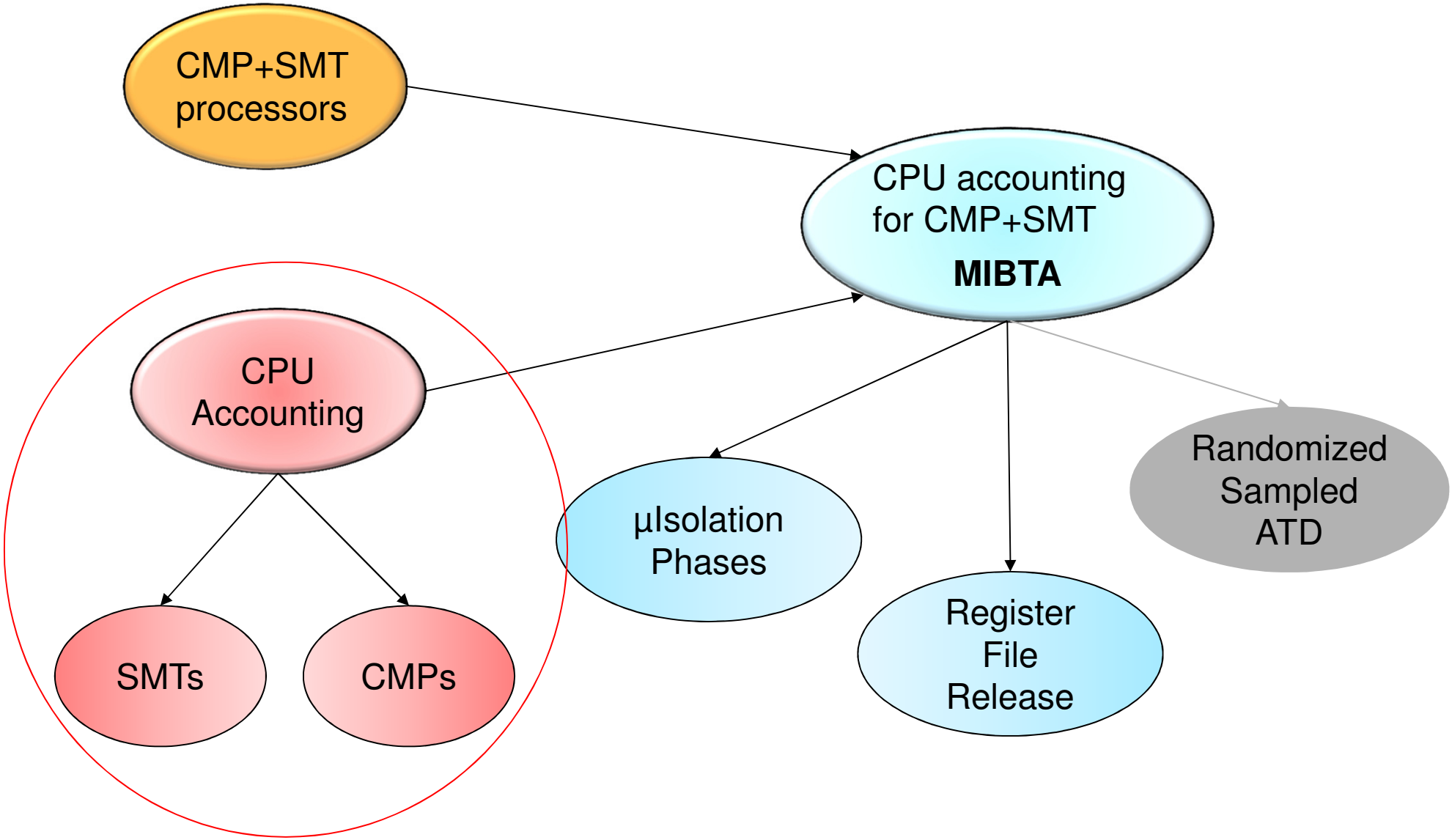

CMP+SMT processors
CPU accounting for CMP+SMT
MIBTA
CPU Accounting
µIsolation Phases
Randomized Sampled ATD
Register File Release
SMTs
CMPs

CMP+SMT processors

- Thread-Level Parallelism (TLP)
  - Overcome the limitations to exploit Instruction-Level Parallelism
  - A wide variety of TLP paradigms (CMP, CGMT, FGMT, SMT)

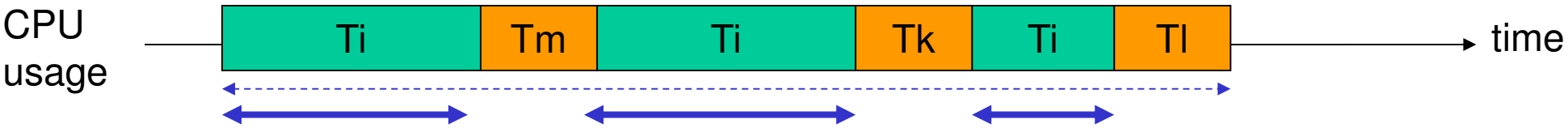
- Processor vendors combine different TLP paradigms
  - Reduce resource underutilization on each core
  - Exploit the available transistors
  - Examples:
    - IBM POWER5/6/7, Intel core i7 (CMP+SMT)
    - Oracle UltraSPARC T1,T2 (CMP+FGMT)
- Multithreaded (MT) processor: processor supporting any TLP paradigm

Outline


CMP+SMT processors
CPU accounting for CMP+SMT
MIBTA
CPU Accounting
SMTs
CMPs
µIsolation Phases
Register File Release
Randomized Sampled ATD

CPU accounting

- CPU Accounting:
  - CPU time accounted to tasks running in a system (TA_i)



- What is CPU Accounting used for?
  - OS task scheduler: maintain fairness between tasks
  - Charge users in data centers
  - Performance tools: statistics of various parameters of a task or a system

- ***Principle of Accounting:*** *the time accounted to a task must always be the same regardless of the workload in which it is executed.*

Measuring CPU accounting

- Single-core: Classical approach
  - Time while the task is running, TR. ($TR_i = TA_i$)
- In MT processors resources are dynamically shared among tasks
  - TA to a task doesn't only depend on the time that task is onto CPU
  - But also on the **progress** that the task makes during that time

C. Luque, M. Moreto, F. J. Cazorla, R. Gioiosa, A. Buyuktosunoglu and M. Valero.

CPU Accounting for Multicore Processors.

In IEEE Transaction on Computers, February 2012.

vould

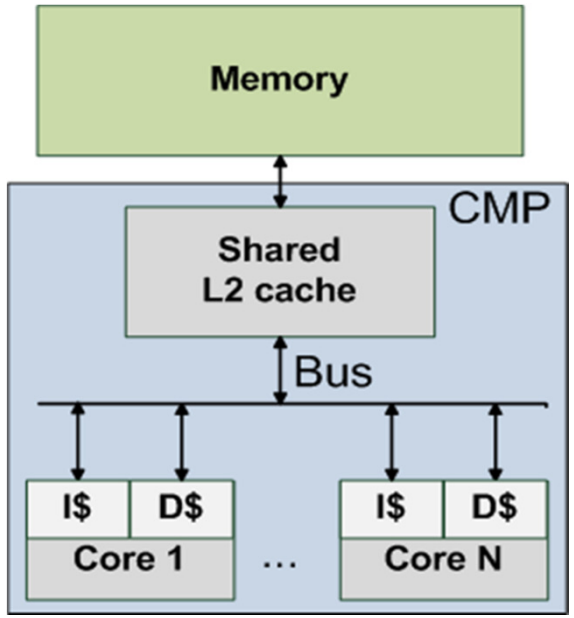
- in isolation (most used baseline. Used in this paper)
- with a fair share of the resources

CPU Accounting in SMTs

- Processor Utilization of Resources Register (IBM POWER5)
  - Decode 1.X: Only one thread can decode up to X instructions per cycle
  - CPU cycles acc. to a task = No. cycles the task decodes instructions
- Scaled PURR (IBM POWER6)
  - CPU acc. scaled to compensate for the impact of throttling and DVFS
- Arndt (US Patent 2006):
  - Decode 2.X
  - CPU cycles acc. to a task ~ No. instructions the task decodes in each cycle
- Eyerman: A Per-thread cycle accounting architecture (ASPLOS 09)
  - Estimates the CPI Stack of each running task based on No. instructions dispatched by a task
    - Extra logic (+15 counters and tables with several R/W ports) spread over all the pipeline and updated on cycle-per-cycle basis
    - Tuned for the case in which the ROB is the bottleneck

CPU Accounting in CMPs

- ITCA: Inter-Task Conflict-Aware Accounting[1,2,3]
  - L2 concentrates the main interaction between tasks
  - On-chip bus, memory bandwidth partially considered



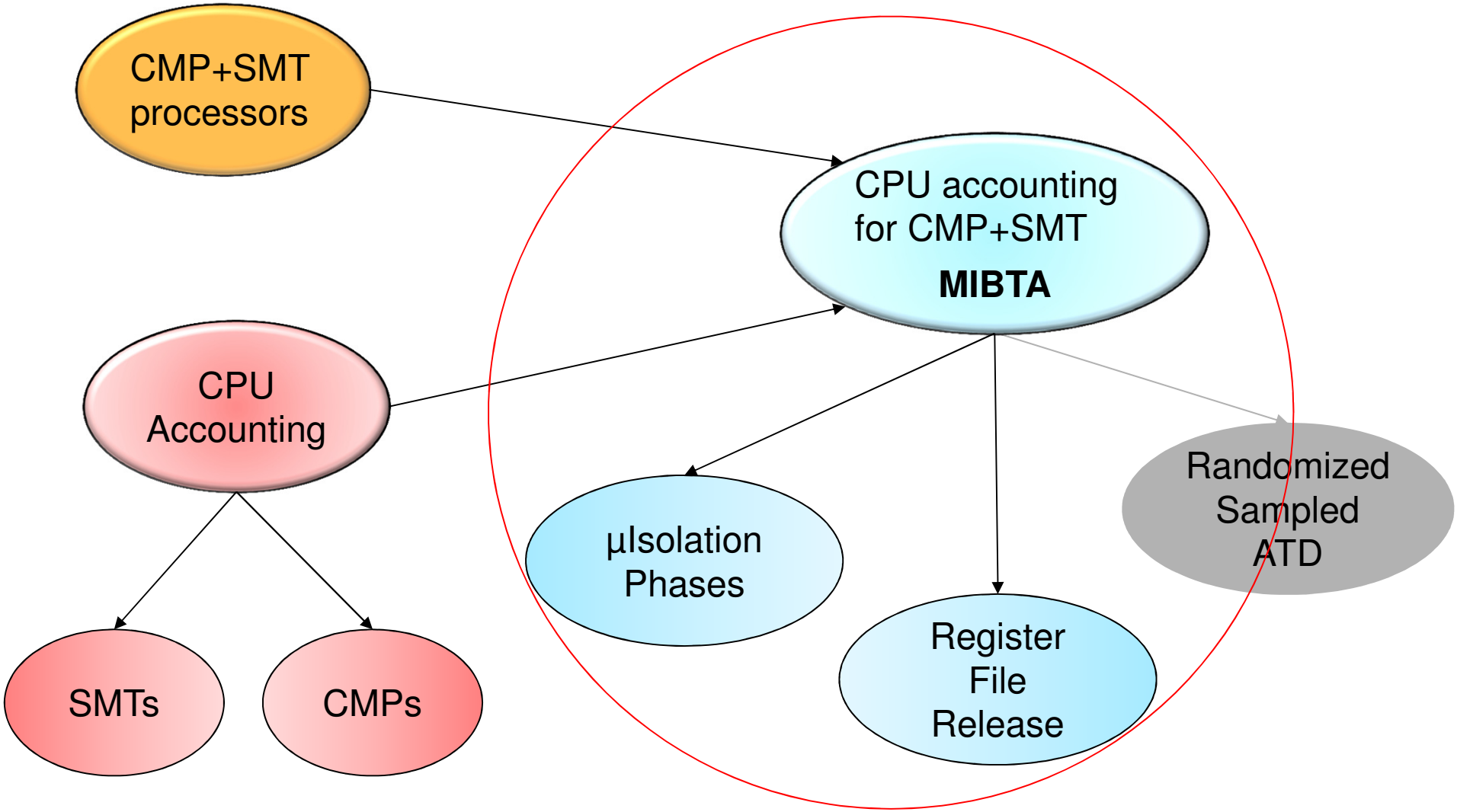
- ITCA principles
  - Keep processor design as simple as possible
  - If task T_B evicts data from a T_A in L2, T_A is said to suffer an inter-task L2 miss
    - ITCA provides support to ensure that the slowdown T_A suffers due to inter-task misses is not added to its CPU accounted cycles
    - ATD: Auxiliary Tag Directory

1 Luque, C. at el, “CPU Accounting in CMP Processors”, CAL Feb 2009

2 Luque, C. at el, “ITCA: Inter-Task Conflict-Aware CPU Accounting for CMPs”, PACT 2009

3 Luque, C. at el, “Accurate CPU Accounting for Multicore Processors”, IEEE Transactions on Computers. Feb 2012

Outline


CMP+SMT processors
CPU accounting for CMP+SMT
MIBTA
CPU Accounting
µIsolation Phases
Randomized Sampled ATD
Register File Release
SMTs
CMPs

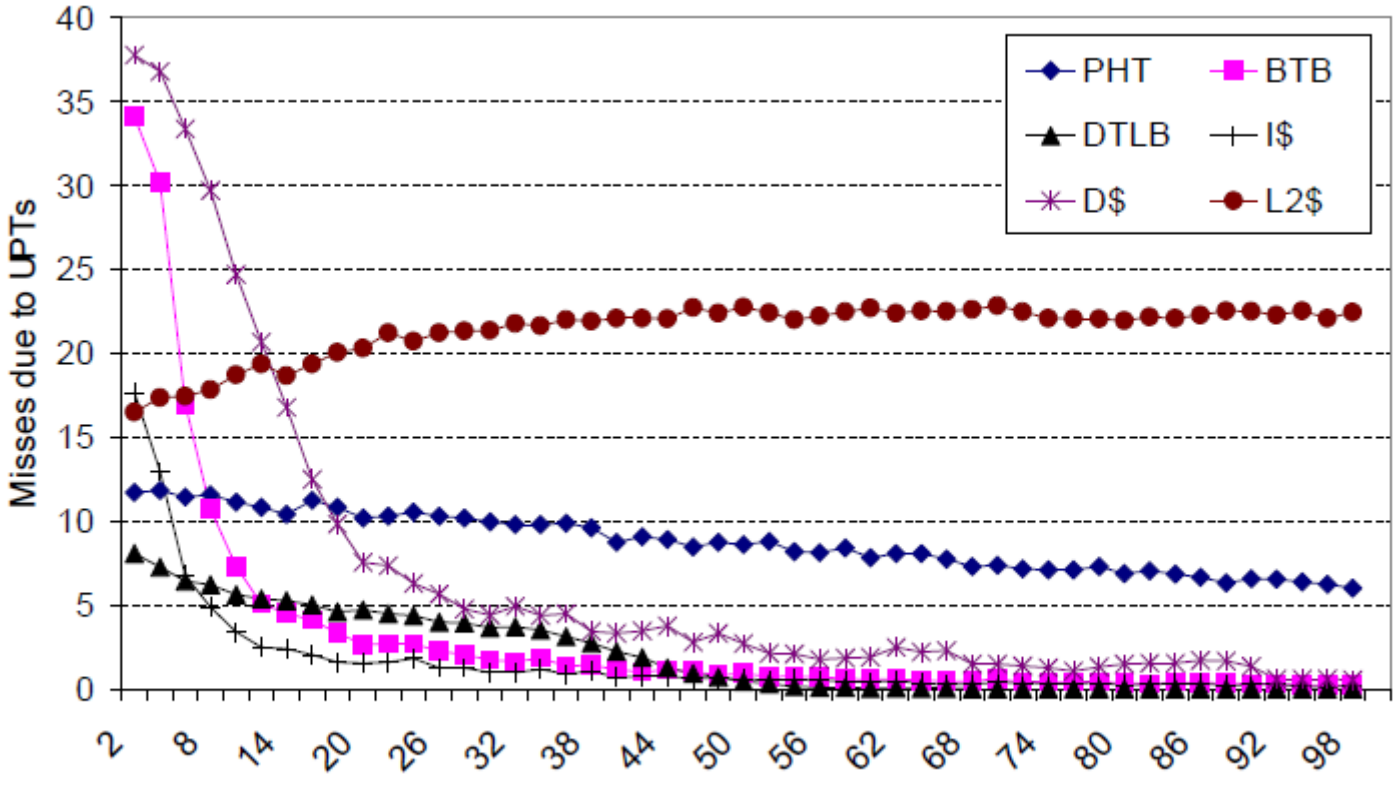
MIBTA: Micro-Isolation-Based Time Accounting

- Previous proposals or combination of them do not work well in CMP+SMT processor
  - Inaccurate
- We developed a new accounting mechanism
  - MIBTA: Micro-Isolation-Based Time Accounting
- MIBTA proposes an integral scalable solution to CMP+SMT processors
  - At SMT level:
    - Time Sampling technique
    - Register File Release
  - At CMP level:
    - Randomized Sampled Auxiliary tag directory, RSA
      - Tracks the interferences on on-cores

MIBTA: SMT level

- Tasks interact in many different resources (IQs, ROB, RFs, …)
  - Tracking all them complicate core design (**it is not a matter of just measuring how many bits data structures require**)
- MIB
  - Do
  - Ins

All tasks but c
(Task Under S



C_{MT}/IPC_{isol}

- MIB
  (alre

age

MT phase i | **Isol phase i**

- Small performance loss due to isolation phases

MIBTA: SMT level: Register File Release

- While in isolation phase the the RF keeps contents of stalled threads[1]
  - TUS enjoys less rename registers than if it runs actually in isolation
  - Its sampled IPC_{isol} is lower than it should be
- MIBTA solution:
  - At the beginning of isolation phase
    - Move architectural registers of the fetch-stalled tasks into the L2
    - Lock those L2 lines
  - Write register values back to the RF at the end of the isolation phase
- TUS enjoys as many rename registers as in isolation
- Complexity:
  - Number L2 lines locked: 4 – 8 depending on the L2 cache size and the number of register
  - Similar technique used in the Intel Sandy Bridge processor

[1] Assuming that the RF is not split into physical and architectural files in which case no change is needed

MIBTA: CMP Level: RSA tag directory

- Based on sampled ATD
  - ATD_i: Copy of the tags of the LLC only accessed by task T_i
  - Hit ATD miss in LLC → inter-task miss
  - Extra logic:The slowdown T_A suffers due to inter-task misses is not added to its CPU accounted cycles
  - Sampled ATD (SATD)
  - RS-ATD

Experimental Setup

- SPEC CPU 2006 benchmarks suite
  - Seven processors configurations: SMT, CMP+SMT
  - Each processor configuration: 26 workloads
- Results
  - The inaccuracy (off estimation) in the CPU accounting:

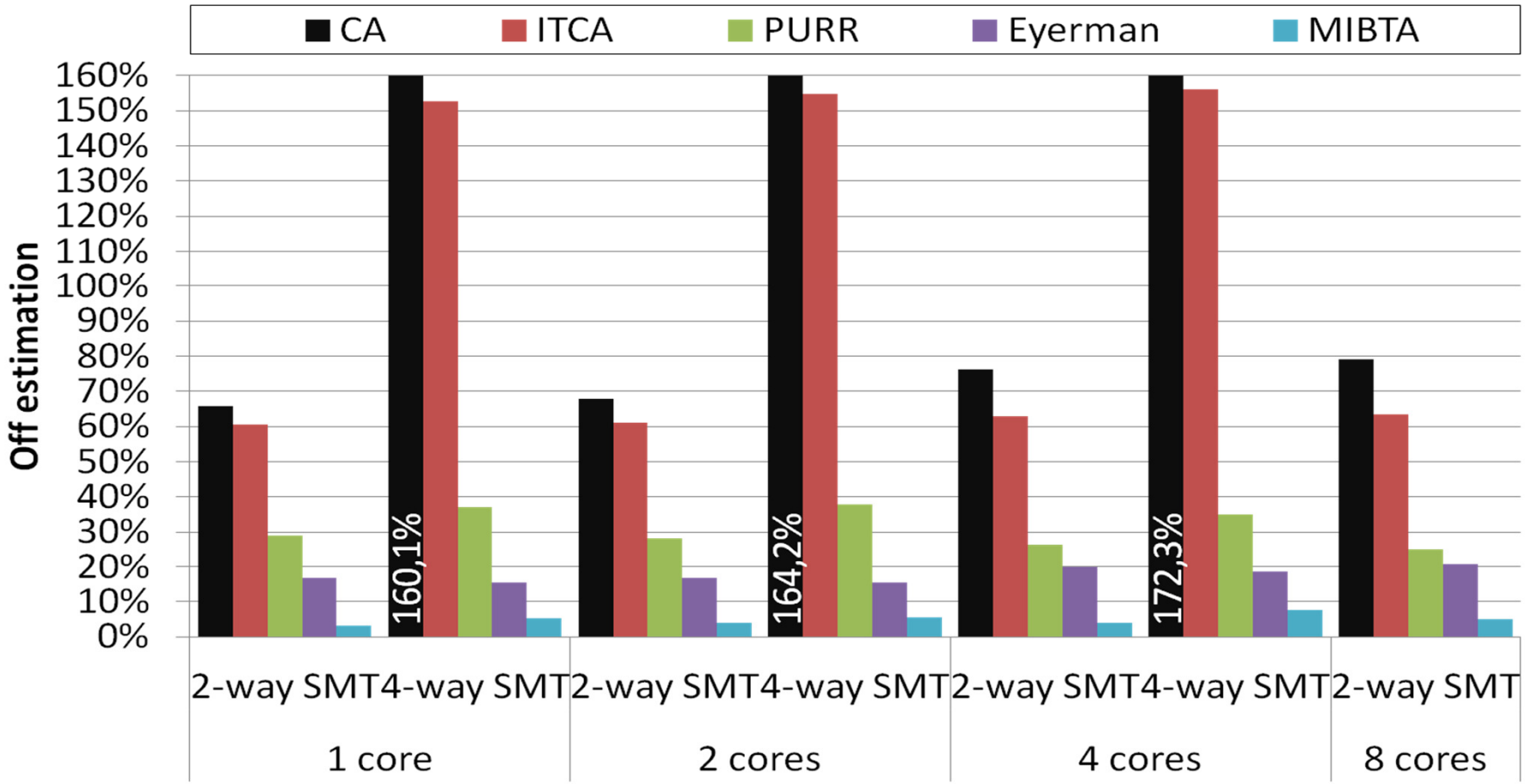
$$Off\ estimation = \left| 1 - \frac{TA^{MT}_{PTh,I_{pth}}}{TR^{isol}_{PTh,I_{pth}}} \right|$$

  - Throughput

$$Throughput = \sum_{i=0}^{N-1} \frac{IPC_i^{MT}}{IPC_i^{isol}}$$ where N number of tasks in the workload

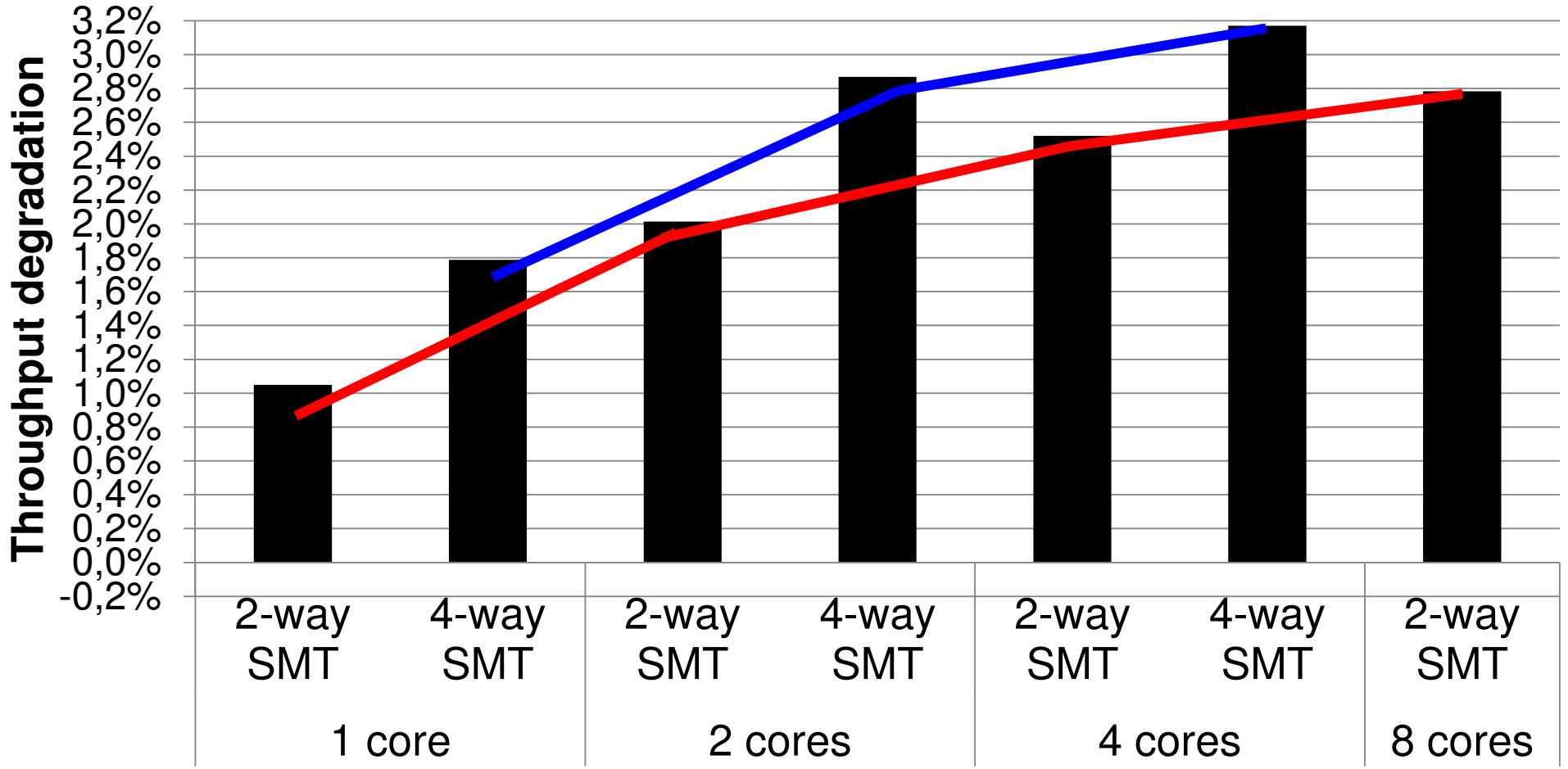
- MITBA
  - Period: 2.6 Million cycles
  - Isolation phase: 100,000 cycles
    - Warmup: 50,000 cycles
    - Actual Isolation phase: 50,000 cycles

Comparison Other Accounting Mechanism



- Techniques targeting CMPs provide worse results than techniques targeting SMT
  - The interaction in SMT cores is much higher than on core-shared resources

Throughput degradation



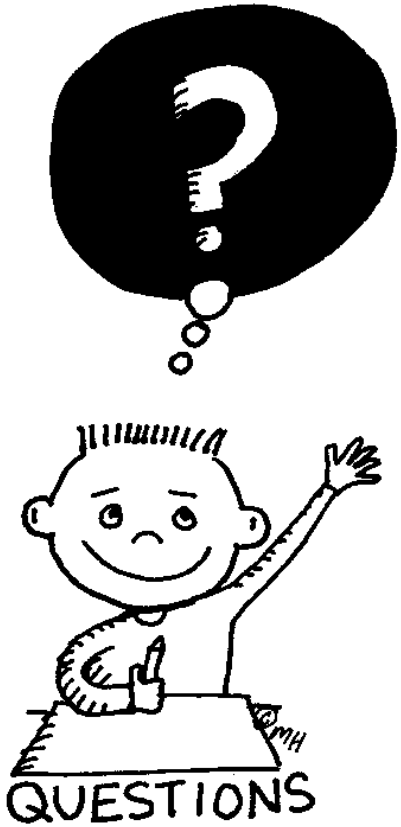
Conclusion

- CPU accounting is a crucial measurement in current Computing Systems

- The current accounting mechanisms are not as accurate as they should be in CMP+SMT processors

- New accounting mechanism for CMP+SMT processors
  - Micro-Isolation-Based Time Accounting, MIBTA
    - High accuracy
    - Low hardware overhead
    - Not depend on the processor architecture

Thanks for the attention!



Fair CPU Time Accounting in CMP+SMT Processors

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Carlos Luque (UPC/BSC)
Miquel Moreto (ICSI/UPC/BSC)
Francisco J. Cazorla (BSC/IIIA-CIS)
Mateo Valero (UPC/BSC)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

8th HIPEAC
Berlin, Germany
21st January 2013

Backup Slides

Eyerman's CPU Accounting for SMT Processors

- A 4-wide dispatch processor has four **dispatch slots** per cycle
- Count the number of **base**, **waiting** and **miss** event dispatch slots **per-task**
  - **Base**: The task made useful progress
  - **Waiting**: The task couldn't progress due to SMT execution
  - **Miss**: The task couldn't progress due to a miss event
- Eyerman's proposal: provide hardware support to identify waiting slots and miss slots due to interferences with other tasks
- Dispatch slots per task can be classified into two possible situations:
  1. The task can dispatch an instruction
     - Correct path → Base slot
     - Wrong path → Branch misprediction slot
     - Requires a **Front-end Miss event Table (FMT)** per task to store base, waiting and miss slots per branch until the branch commits

Hardware Overhead

2. The task can't dispatch an instruction
   - If the task doesn't suffer a miss → **Waiting** slot
   - If the task suffers a miss, we increase the corresponding **miss** counter
     - Front-end misses: instruction L1 and L2 caches, instruction TLB miss
     - Full ROB due to L2 data misses, data TLB misses, long latency units, dependencies, etc.
     - Other misses
   - Front-end misses have more priority than backend misses
   - When there are several front-end misses, we can only have a branch misprediction and an instruction L1, L2, or TLB miss → priority to the branch
   - When there are several back-end misses → priority to the miss associated to the first instruction in the ROB
   - Detect when the ROB would fill in isolation
     - **Virtual-ROB (V-ROB)** counter: in-flight base and waiting slots
     - Start counting miss cycles when V-ROB equals the ROB size

- **Issue**: the ROB is not always the main bottleneck of the arch.

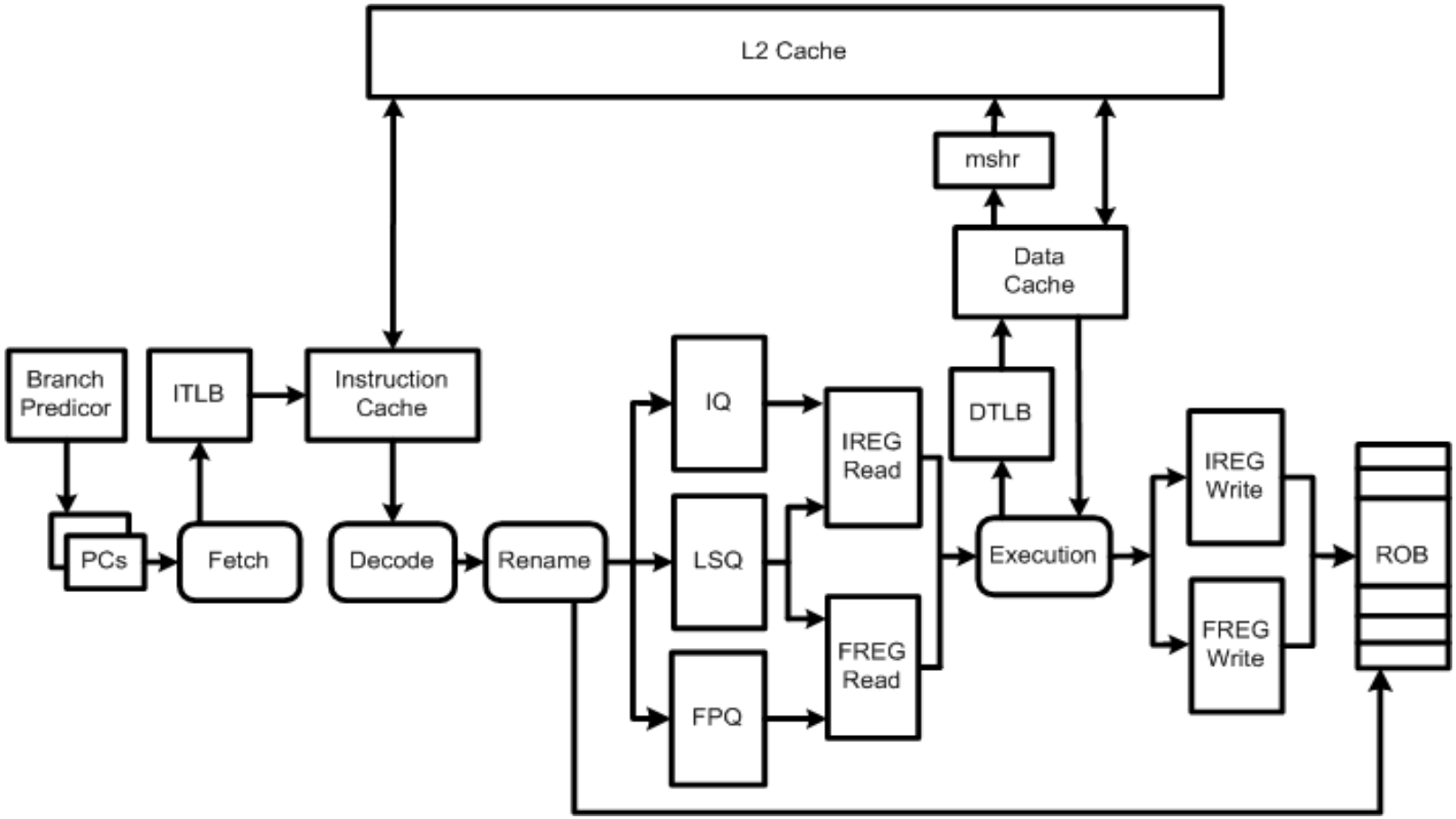
Inter-task interferences

- Detect inter-task L1, L2, TLB and branch predictor interferences:
  - Sampled ATD is required for L2 cache (16/4K sets)
  - Fully-associative per-thread tag directory for data TLB (16/512 entries)
  - Add tag ID per predictor entry
  - Apply correction factors to the miss events components
- Detect difference in MLP
  - Measure current MLP
  - Estimate MLP in isolation using a **Back-end Miss event Table (BMT)**
  - The ratio between these MLPs rescales the LLC cycle component
- Finally, predict CPI in isolation per task

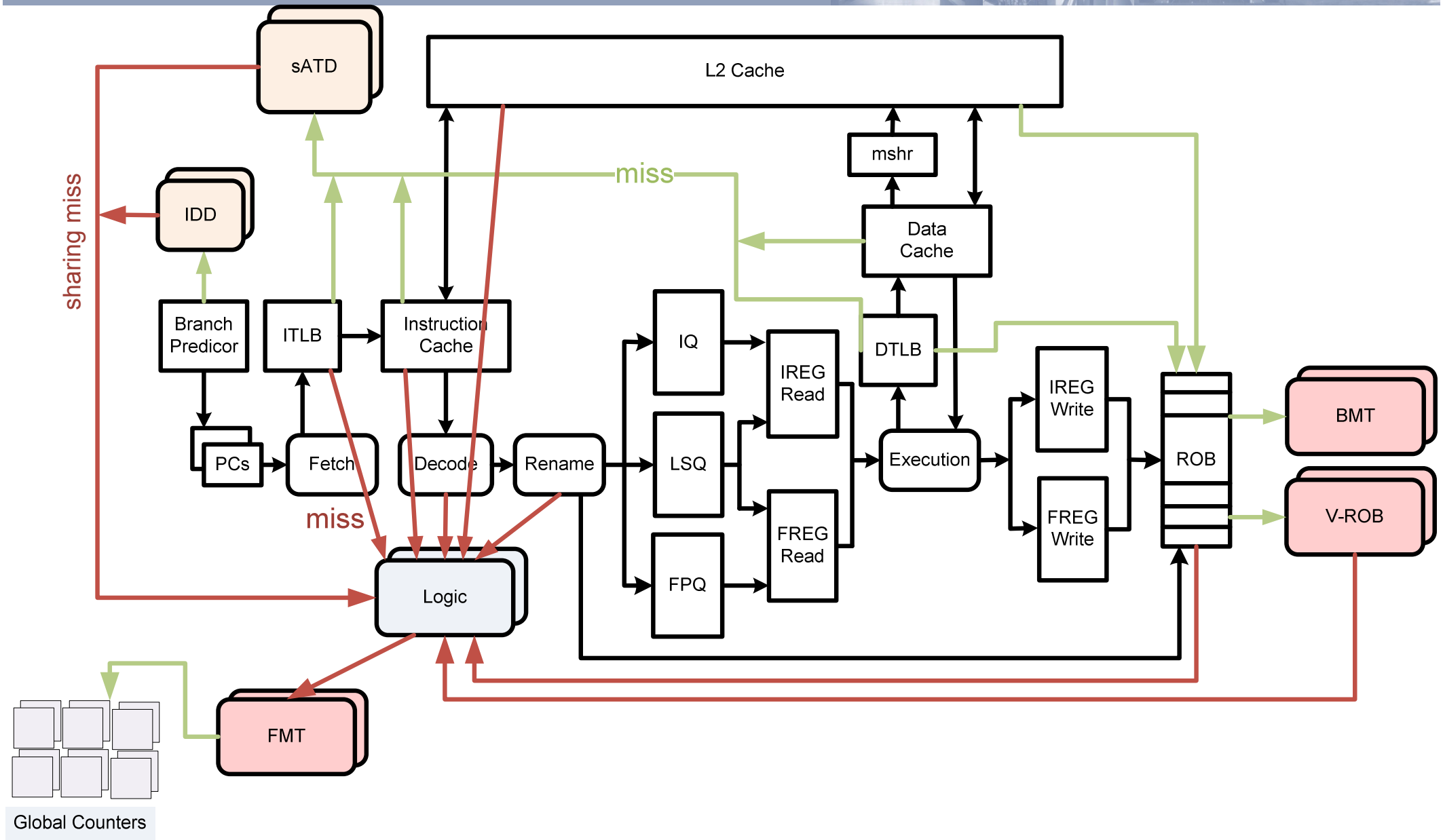
Hardware Overhead

- Nothing is said about other possibilities such as the task cannot dispatch but the ROB is not full and no miss is occurring.
- This could happen when the issue queues get full before the ROB or when there are no available rename registers.
- In this case, we decide to increase the other miss counter.

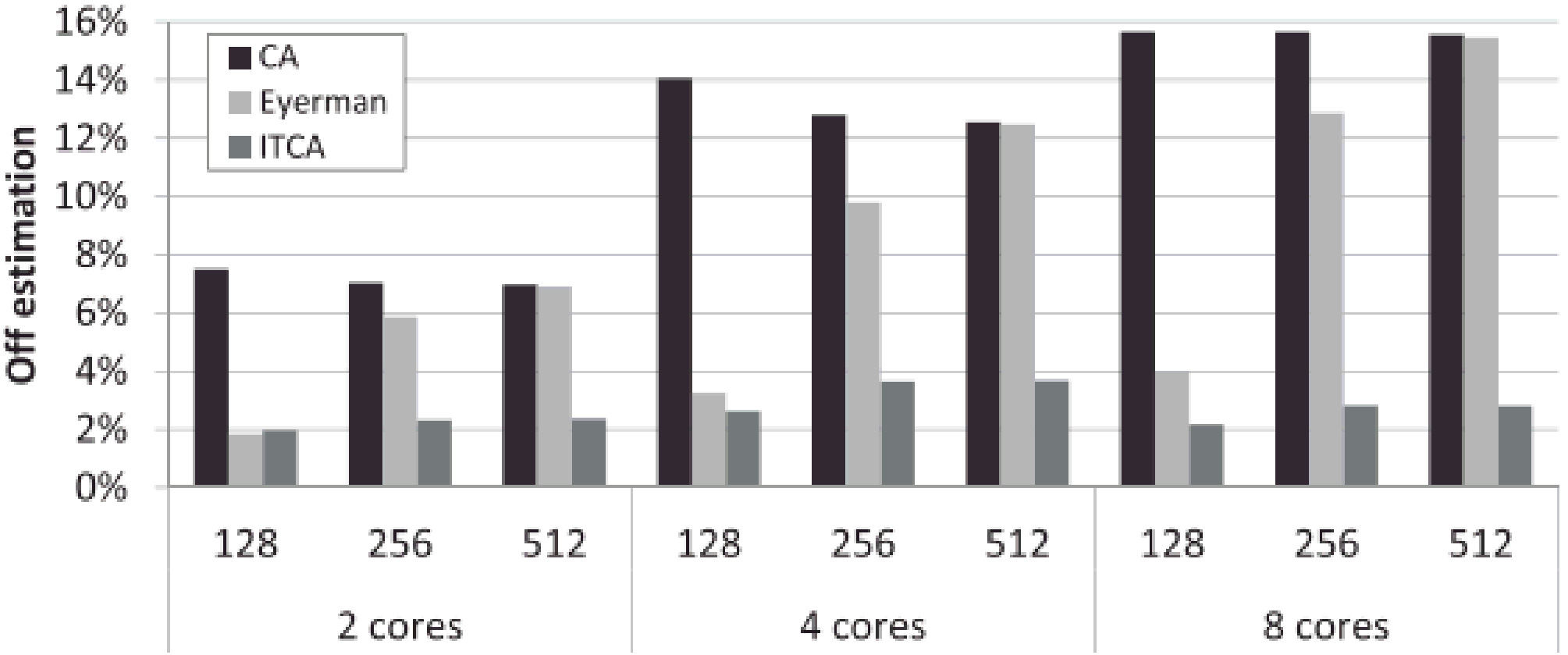
Hardware Overhead



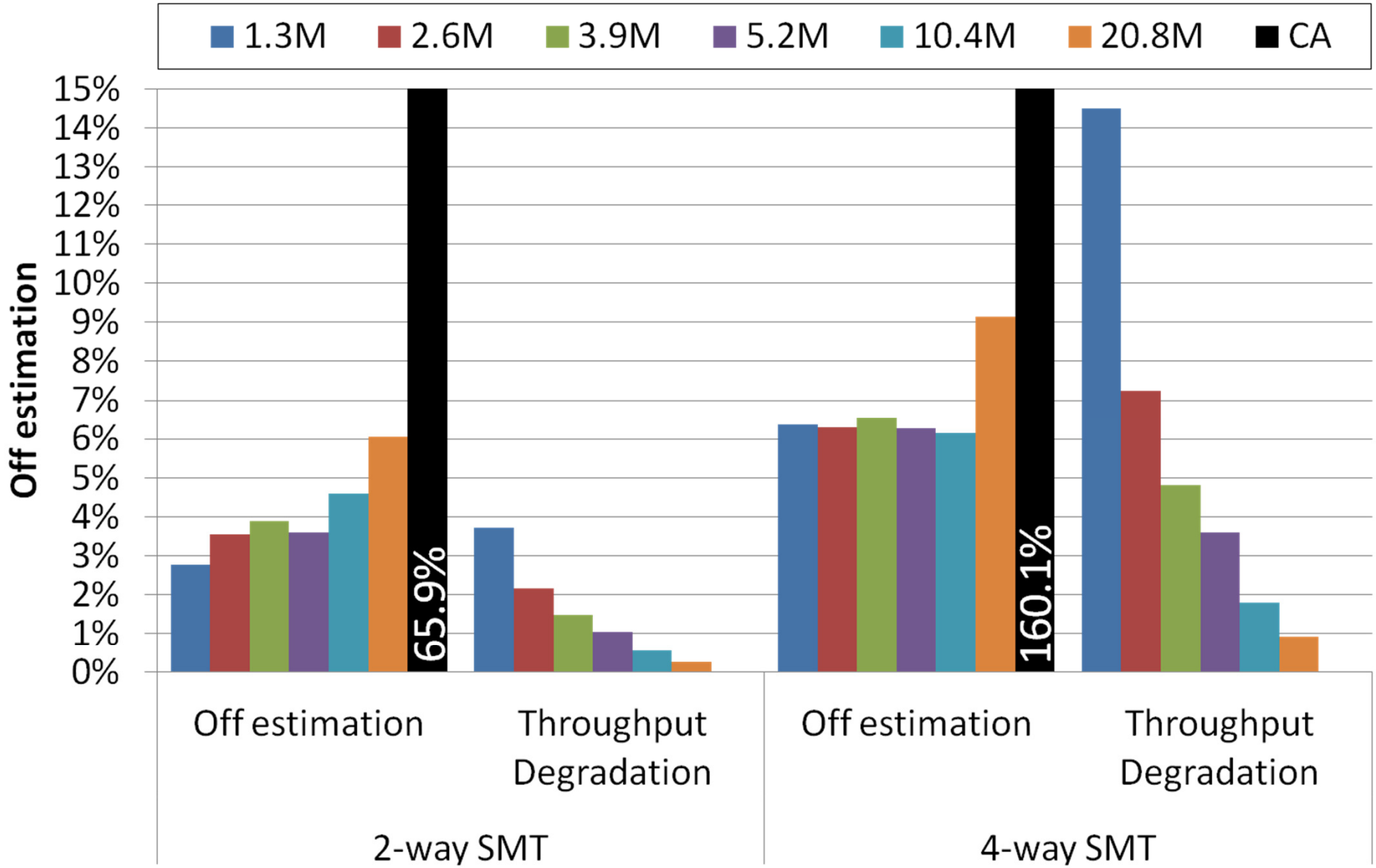
Hardware Overhead



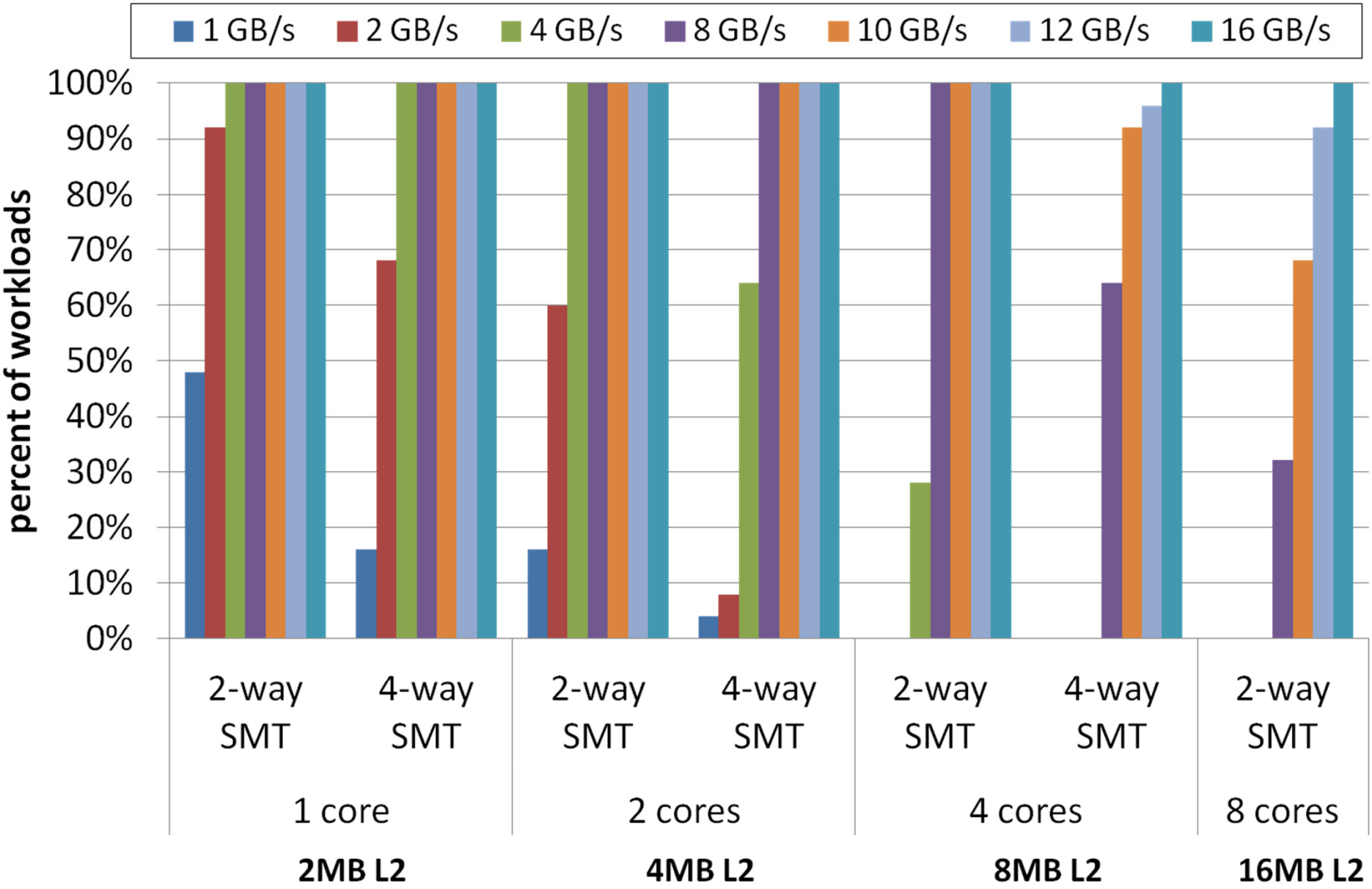
Effect of ROB



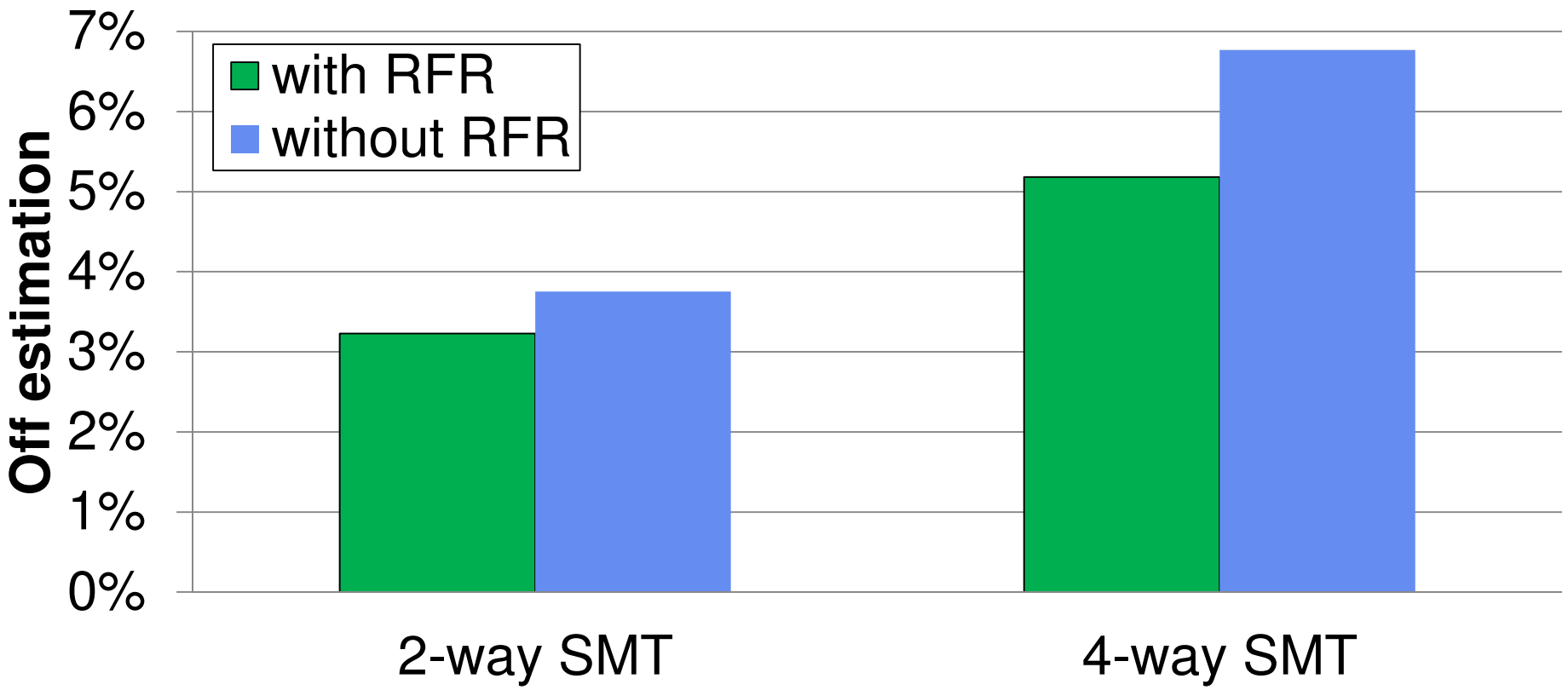
MIBTA in SMT


1.3M
2.6M
3.9M
5.2M
10.4M
20.8M
CA
Off estimation
15%
14%
13%
12%
11%
10%
9%
8%
7%
6%
5%
4%
3%
2%
1%
0%
65.9%
160.1%
Off estimation
Throughput Degradation
Off estimation
Throughput Degradation
2-way SMT
4-way SMT

Memory bandwidth Sensitivity



Register file Release in SMT



Simulation configuration

Core configuration		
	2-way SMT	4-way SMT
Number of core	1,2,4,8	1,2,4
Issue Queue entries	48 int, 48 fp, 48 ld/st	64 int, 64 fp, 64 ld/st
Physical Registers	164 int, 164 fp	256 int, 256 fp
ROB size	256	352
Execution Units	4 int, 2 fp, 2 ld/st	
Fetch Policy	ICOUNT 1.8	
Branch predictor	2K entries, gshare	
Branch Target Buffer	256 entries and 4 ways	
Clock Frequency	2.0GHz	

Cache/Memory Configuration				
Core/s	1	2	4	8
LLC (shared)	2MB	4MB	8MB	16MB
	16 ways, 8 banks, 128 Bytes			
Instruction (per core)	64 KB, 4 ways, 1 bank, 128 Bytes			
Data (per core)	64 KB, 8 ways, 1 bank, 128 Bytes			
ITLB (per core)	128 entries, 8 KB page			
DTLB (per core)	256 entries, 8 KB page			
Latencies	LLC (15), Memory (300)			

MIBTA: CMP Level

- Track interference off-core resources (L2)
  - We track inter-task interferences in the L2
    - Extend for Several million cycles
    - Lead to a bad estimation of $IPC_{isolation}$
- We propose: Randomized Sampled ATD (RSA)
  - Detects inter-task misses in the LLC
  - One per task
  - We add a bit in each entry of MSHR to track inter-task misses
  - Uses the accounting decision provided by the ITCA

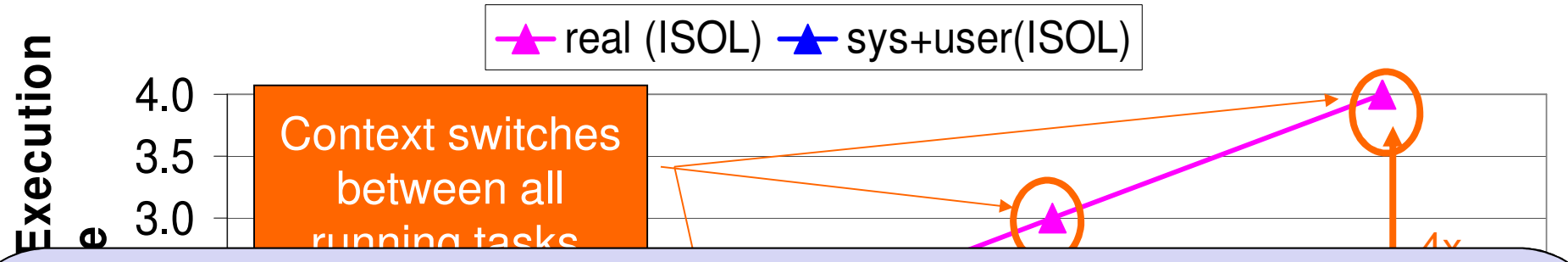
MIBTA: CMP Level: RSA tag directory

- Based on sampled ATD
  - ATD_i: Copy of the tags of the LLC only accessed by task T_i
  - Monitored and non-monitored sets
  - Hit ATD miss in LLC → inter-task miss
- Track the probability of having inter-task misses in sampled sets
  - In MT phase:
    - Track and accumulate the number of inter-tasks misses and total misses
  - In isol phase:
    - In warmup phases: Calculates inter-task Inter-task miss probability (inter-task/total)
    - In actual isolation phase:
      - On a miss in cache and access not in the non-monitored sets: Generate random number
        - If random number < probability => inter-task miss
        - else => intra-task miss

Hardware Overhead

- The hardware overhead of our proposal is as follows:
  - RSA requires 0.97KB:
    - Assuming a 2MB LLC, a sATD with 32 sets,
    - two 64-bit registers,
    - two shifter registers,
    - a LFSR.
  - Four 64-bit special purpose registers (ICR and IIR, MTRC and MTIR) require 0.03KB.
- In total, we have an overhead per task of 1KB.
- At core level, the hardware overhead
  - 0.04KB in a 2-way SMT
  - 0.05KB in a 4-way SMT
    - (one bit per entry in both ROB and MSHR, and three 20-bit registers)

Real Time and Accounted Time (II)

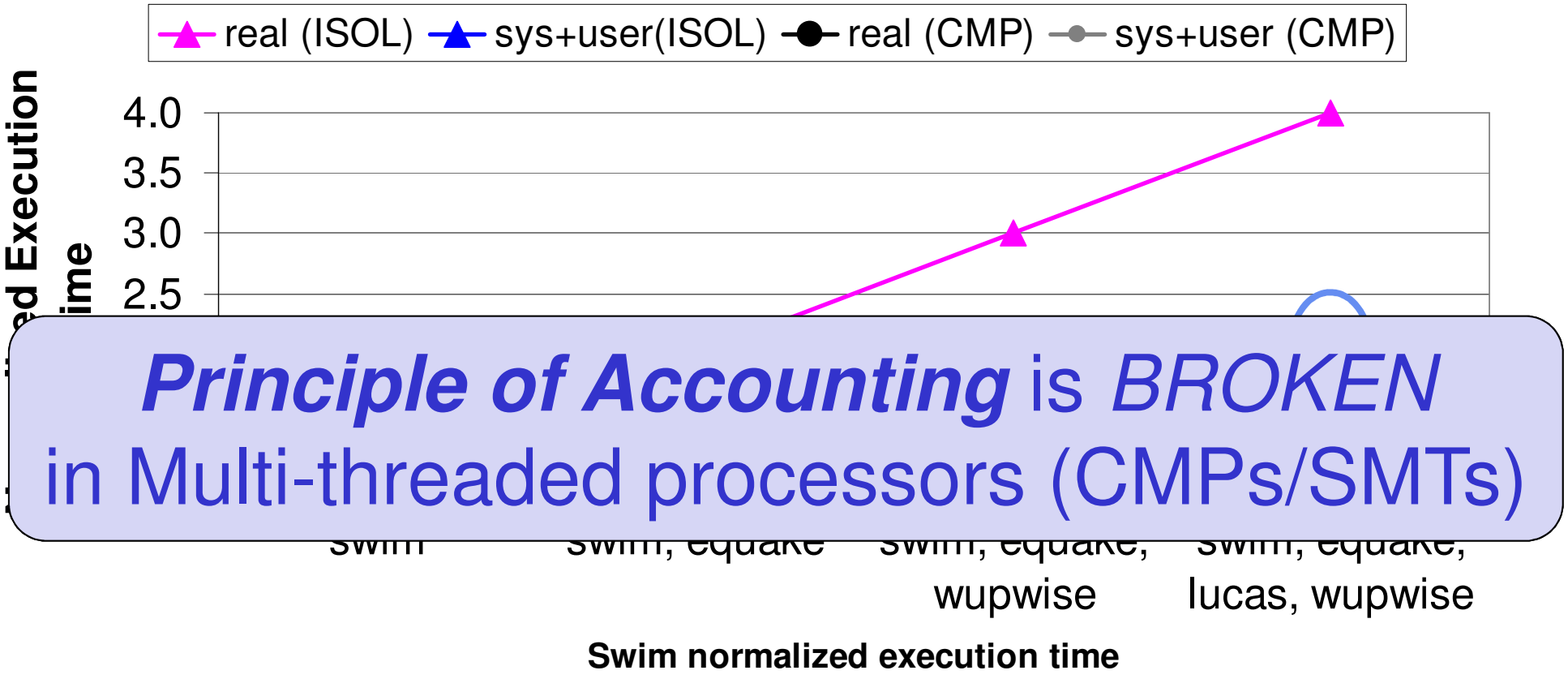


The time accounted to an application is always the same regardless of the workload in which it is executed in single-core, single-threaded processors

Principle of Accounting

- Swim real execution time increases with the number of running tasks.
- Swim accounted time is the same!

Real Time and Accounted Time (III)



Principle of Accounting is *BROKEN*
in Multi-threaded processors (CMPs/SMTs)

- Execution time in an MT processor (a four-core CMP):
  - Swim real execution time increases with the number of running tasks.
  - Swim accounted time also increases with the number of running tasks