Cache-Related Preemption and Migration Delays: Empirical Approximation and Impact on Schedulability

OSPERT 2010, Brussels
July 6, 2010

Andrea Bastoni

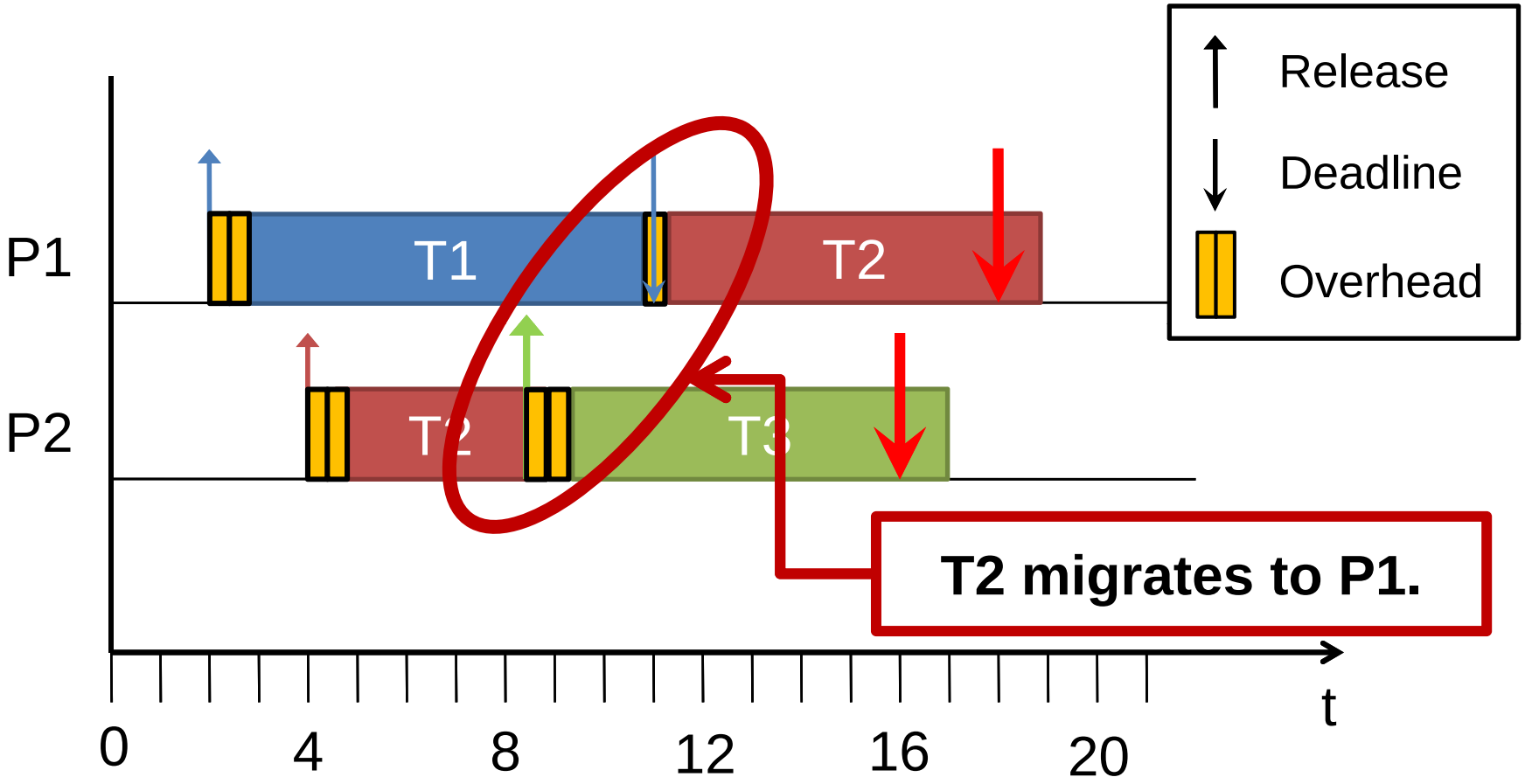
University of Rome
"Tor Vergata"

Björn B. Brandenburg
James H. Anderson

The University of North Carolina
at Chapel Hill

Work supported by AT&T, IBM, and Sun Corps.; NSF grants CNS 0834270, CNS 0834132, and CNS 0615197; ARO grant W911NF-09-1-0535; and AFOSR grant FA 9550-09-1-0549.

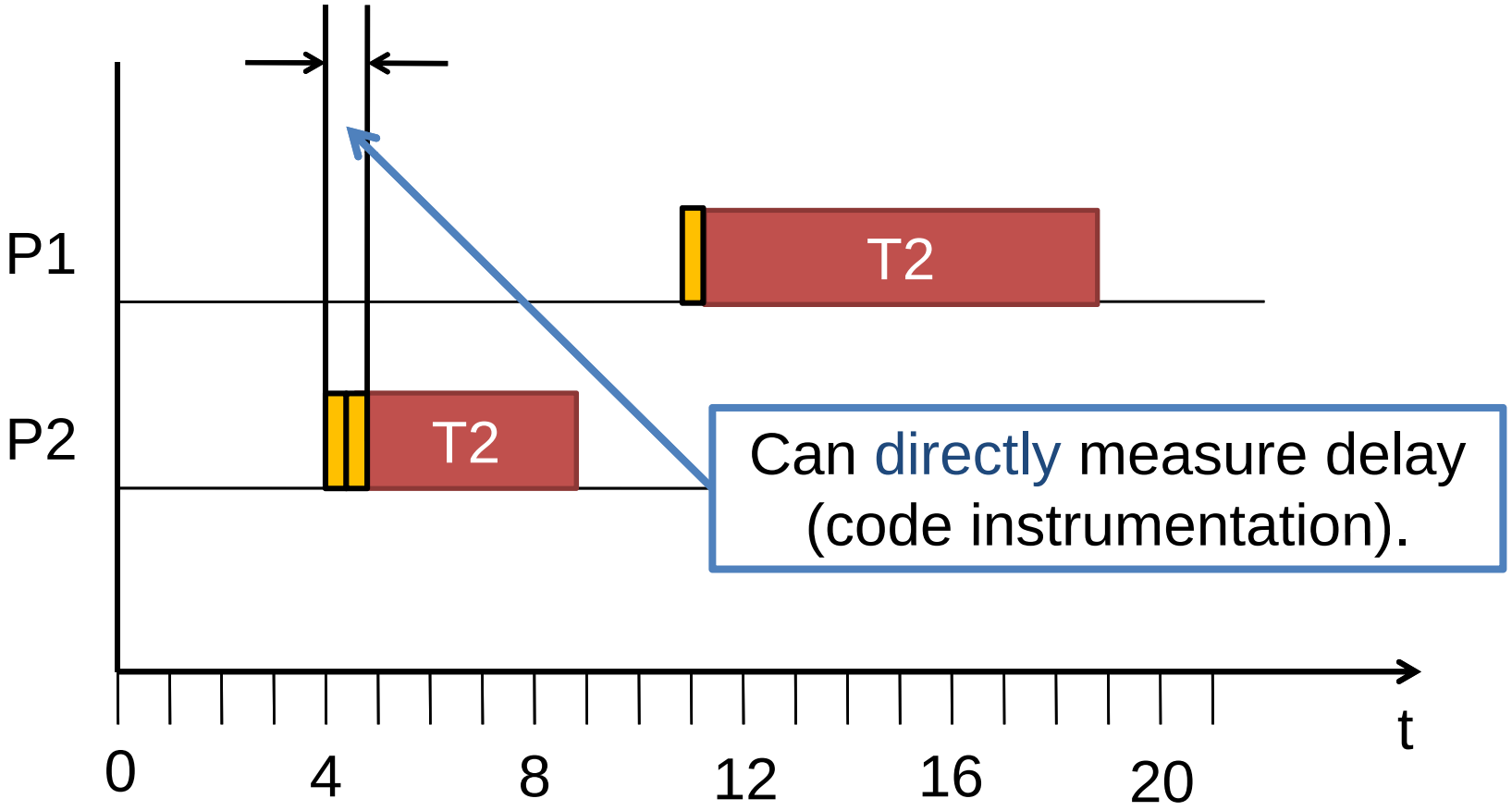
The Problem



Multiprocessor Global EDF schedule with overheads.

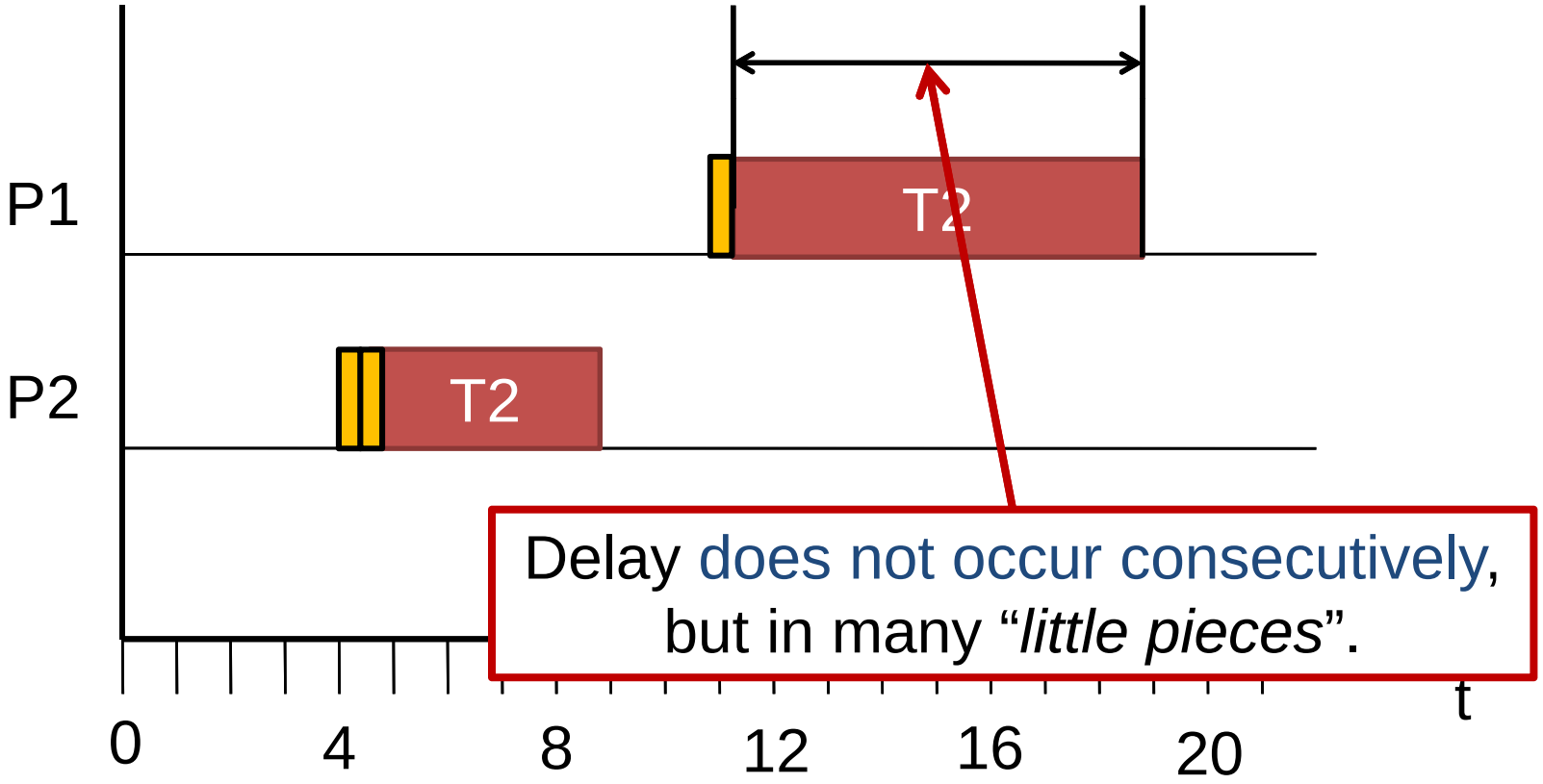
The Problem

Kernel overheads (e.g., release overhead, scheduling overhead, etc.) are "easy" to measure.



The Problem

Overheads due to preemption / migrations are not!

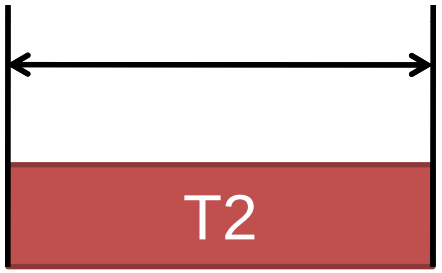


Outline

- Cache-related preemption and migration delays (CPMD).
- Two methods to measure CPMD.
- Experimental results and discussion.
- Impact on schedulability (sketch).

Cache-Related Preemption and Migration Delays (CPMD)

- Cache-related preemption and migration delays:
  - Delays due to additional cache misses when resuming execution after a preemption or a migration.



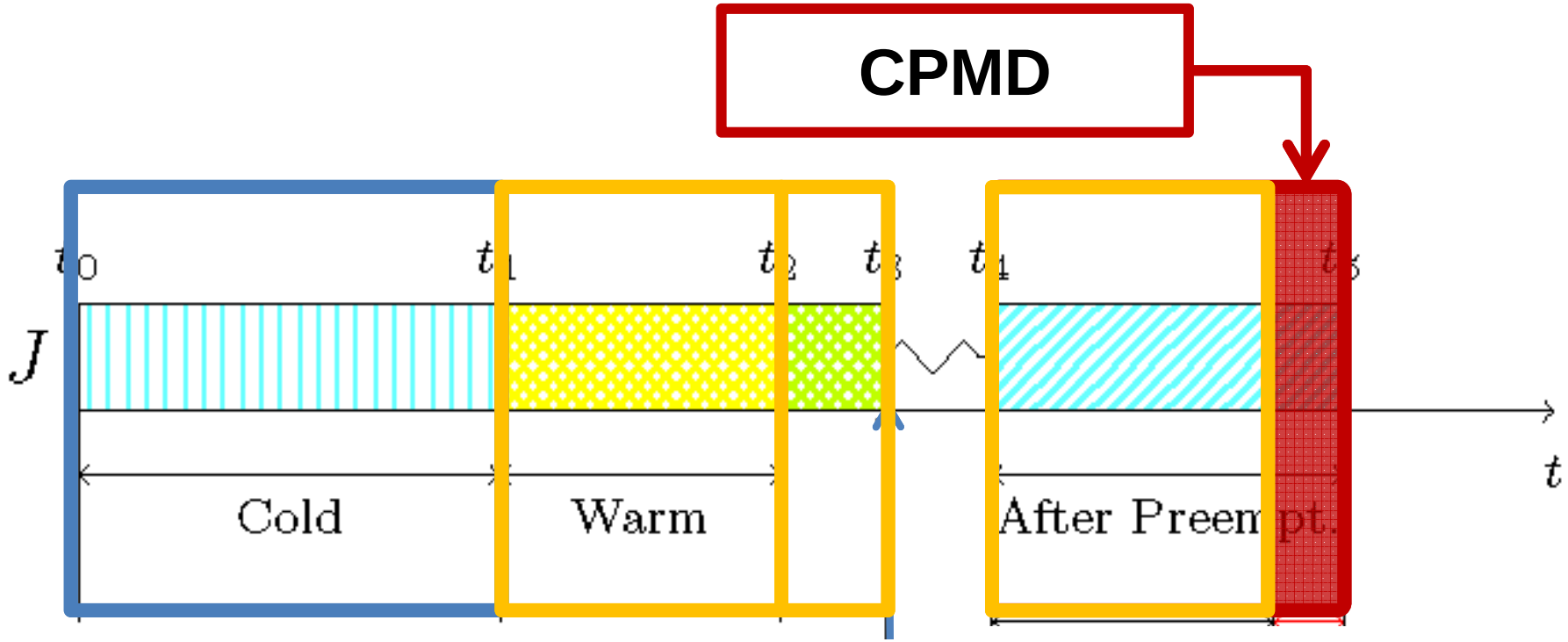
Data in local cache. → (Re)fetch.

Data in remote cache. → Invalidate.

- Heavily dependent on *working set size* (WSS).
- No effective WCET analysis techniques available for current multiprocessors with cache hierarchies.
- Need to rely on empirical measurements.

Detecting CPMD

- In this study: empirical approximation of CPMD.

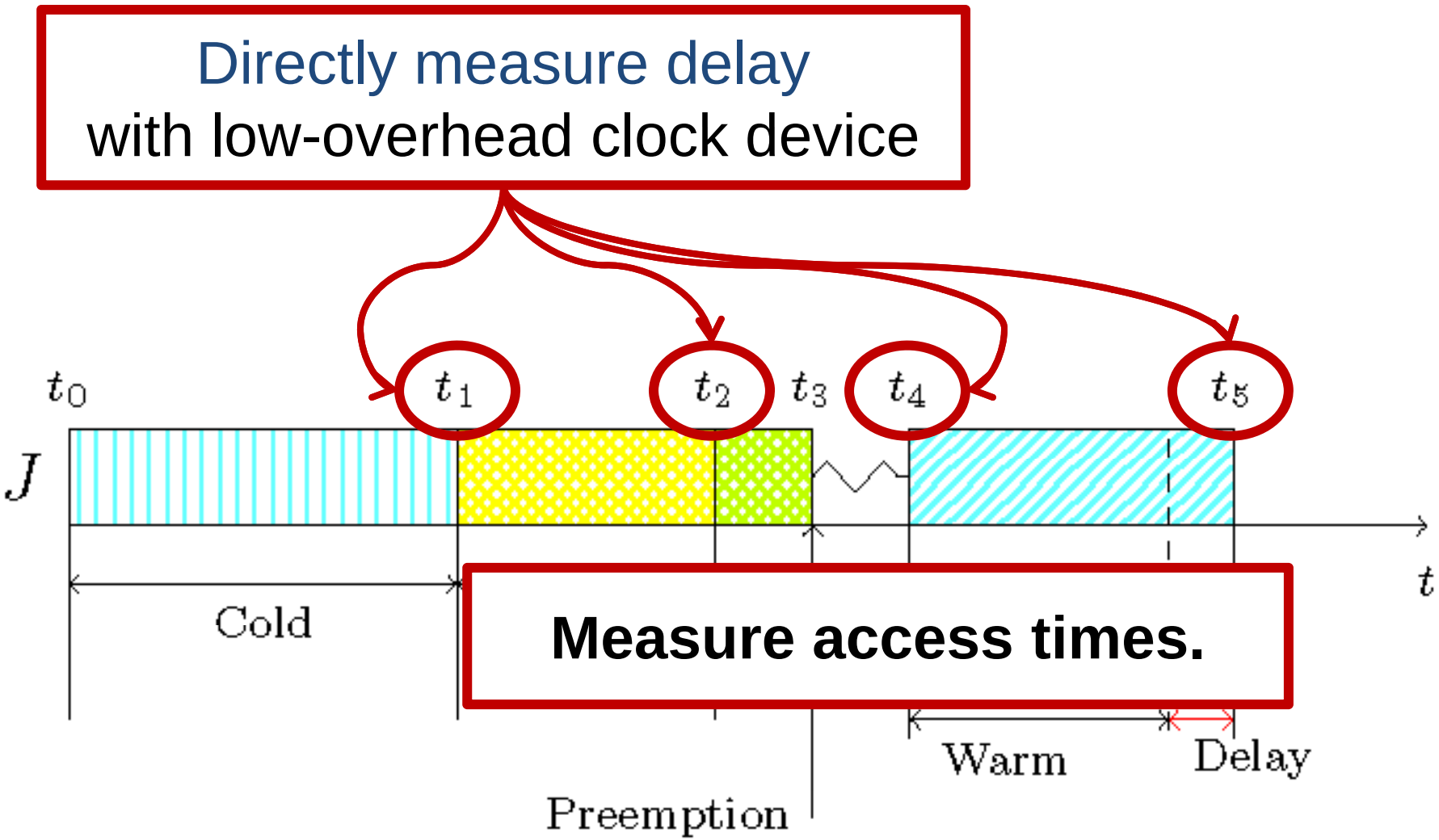


Complete access to Working Set (WS), cache-cold.

Complete access to J Working Set (WS), cache-warm.

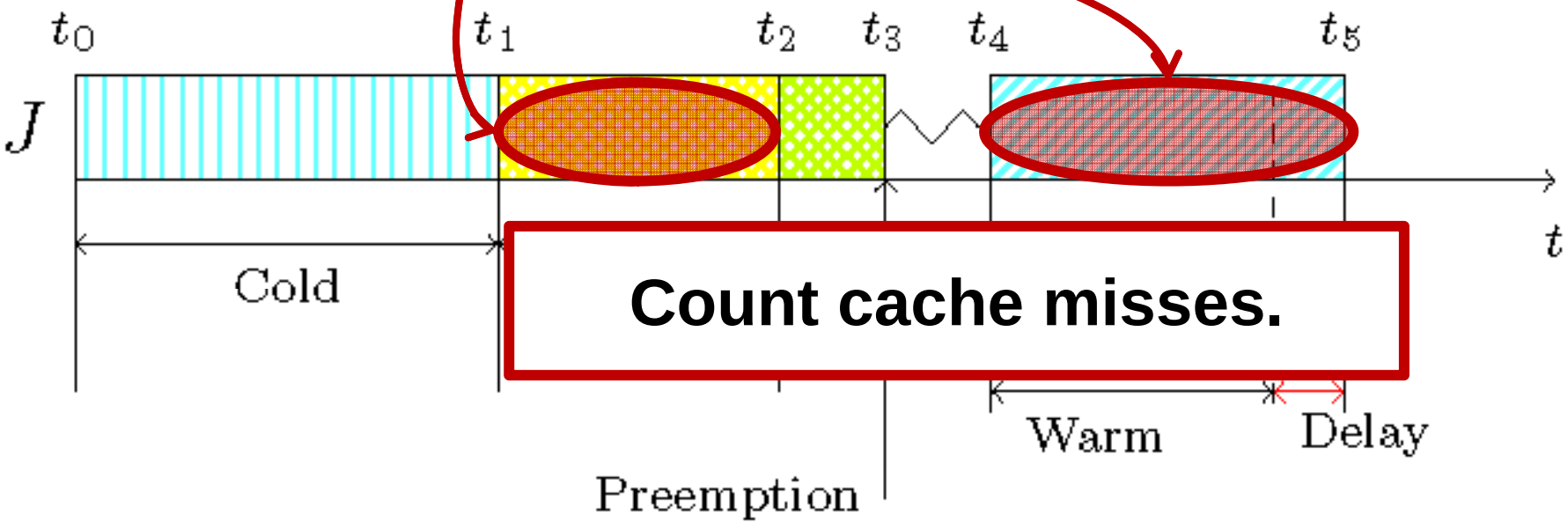
Complete access to J Working Set (WS), after preemption.

Measuring CPMD



Measuring CPMD

Indirectly measure delay
with hardware perf. counters



Outline

- Cache-related preemption and migration delays (CPMD).
- Two methods to measure CPMD.
- Experimental results and discussion.
- Impact on schedulability (sketch).

Schedule-Sensitive Method

- On-line recording of delays:
  - Execute instrumented synthetic tasks under desired scheduling policy.
  - Wide range of WSS, TSS, and read/write ratios.
- Can reveal dependencies on:
  - Scheduling policy.
  - Task set size (TSS).
- Cannot explicitly control preempt./migrat.
  - P/M depends on the scheduling policy.
  - Not every job yields a valid measure.
  - A job may be preempted too early, too late, or not at all.

Synthetic Method

- Fine-grained control on measurement process:
  - Artificially trigger preemptions and migrations.
  - Explicit control on preemptions and different types of migrations (through L2 cache, L3 cache, and memory).
- Fixed-prio scheduling policy (e.g., SCHED_FIFO).
  - Single high-prio tasks access wide range of WSS.
  - Wide range of read/write ratios.
- Every job yields a valid measure.
- Cannot detect dependencies on:
  - Scheduling policy, TSS.

Implementation

- Operating System:

LINUX  + LITMUS^RT
Linux Testbed for Multiprocessor Scheduling in Real-Time Systems

- UNC's real-time Linux extension.
- Developed as kernel patch (currently based on Linux 2.6.32).
- Code is available at http://www.cs.unc.edu/~anderson/litmus-rt/.

Implementation Issues

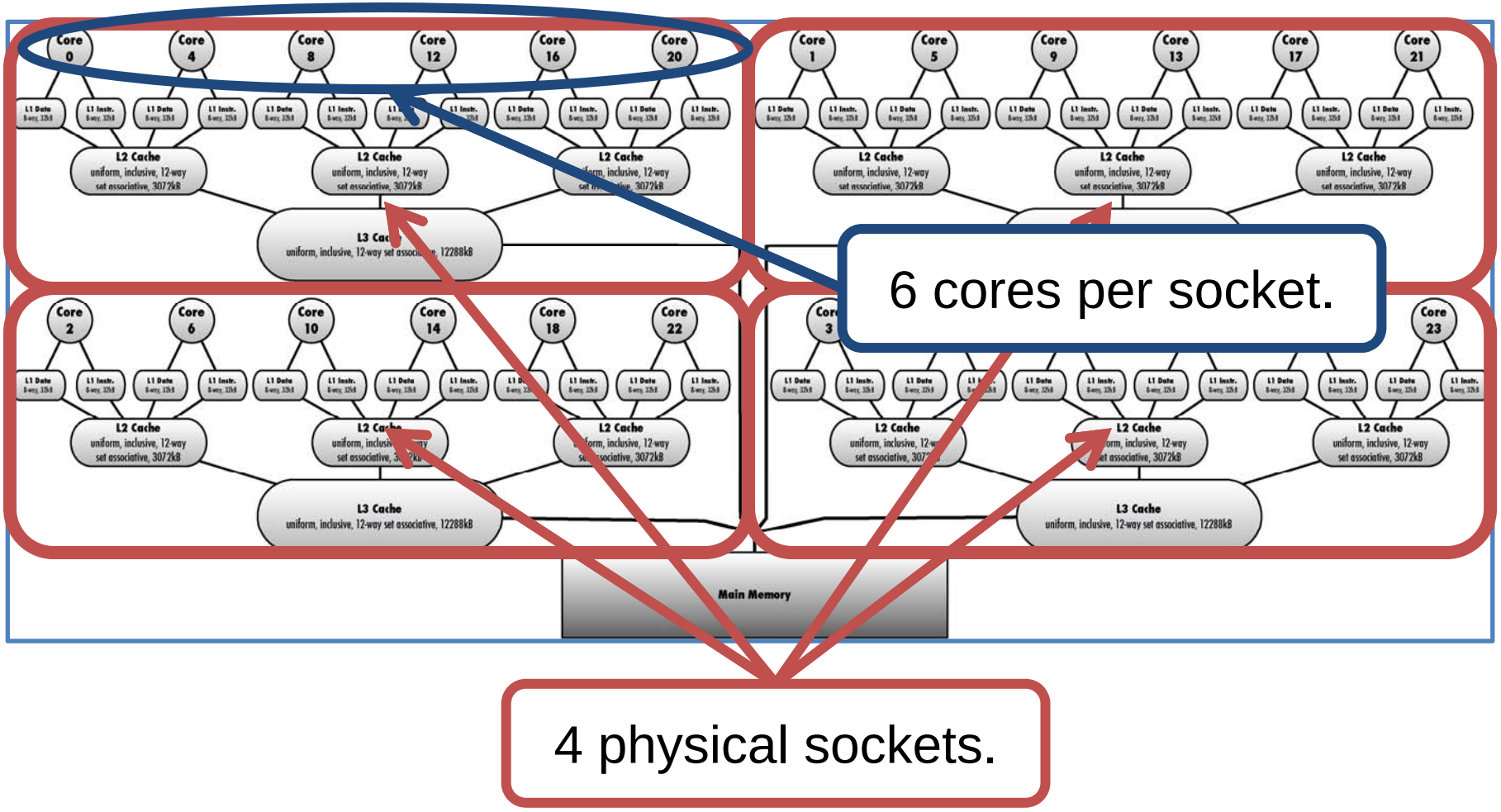
- Low-overhead clock device.
  - Time-stamp counter (TSC). Per-core clock device.
- Clock skew among cores.
  - WS access times only based on samples from the same processor.
- Interrupts interference.
  - Interrupts disabled during WS access.
- How to detect when a preempt./migration occurred.
  - Low-overhead kernel-user communication mechanism.
  - Per-task memory page shared with the kernel.

Outline

- Cache-related preemption and migration delays (CPMD).
- Two methods to measure CPMD.
- Experimental results and discussion.
- Impact on schedulability (sketch).

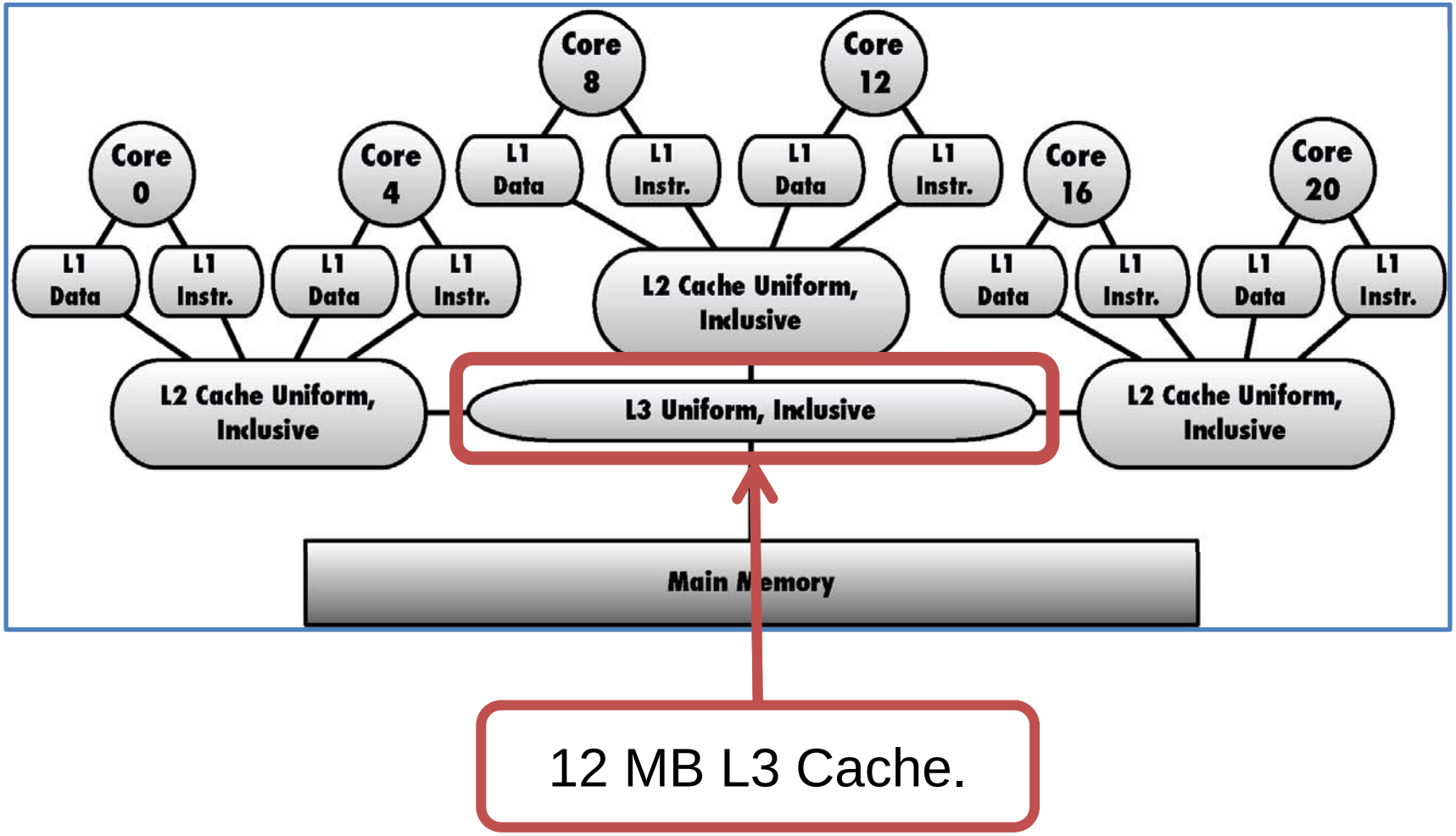
Test Platform

- Intel Xeon L7455 “Dunnington”:



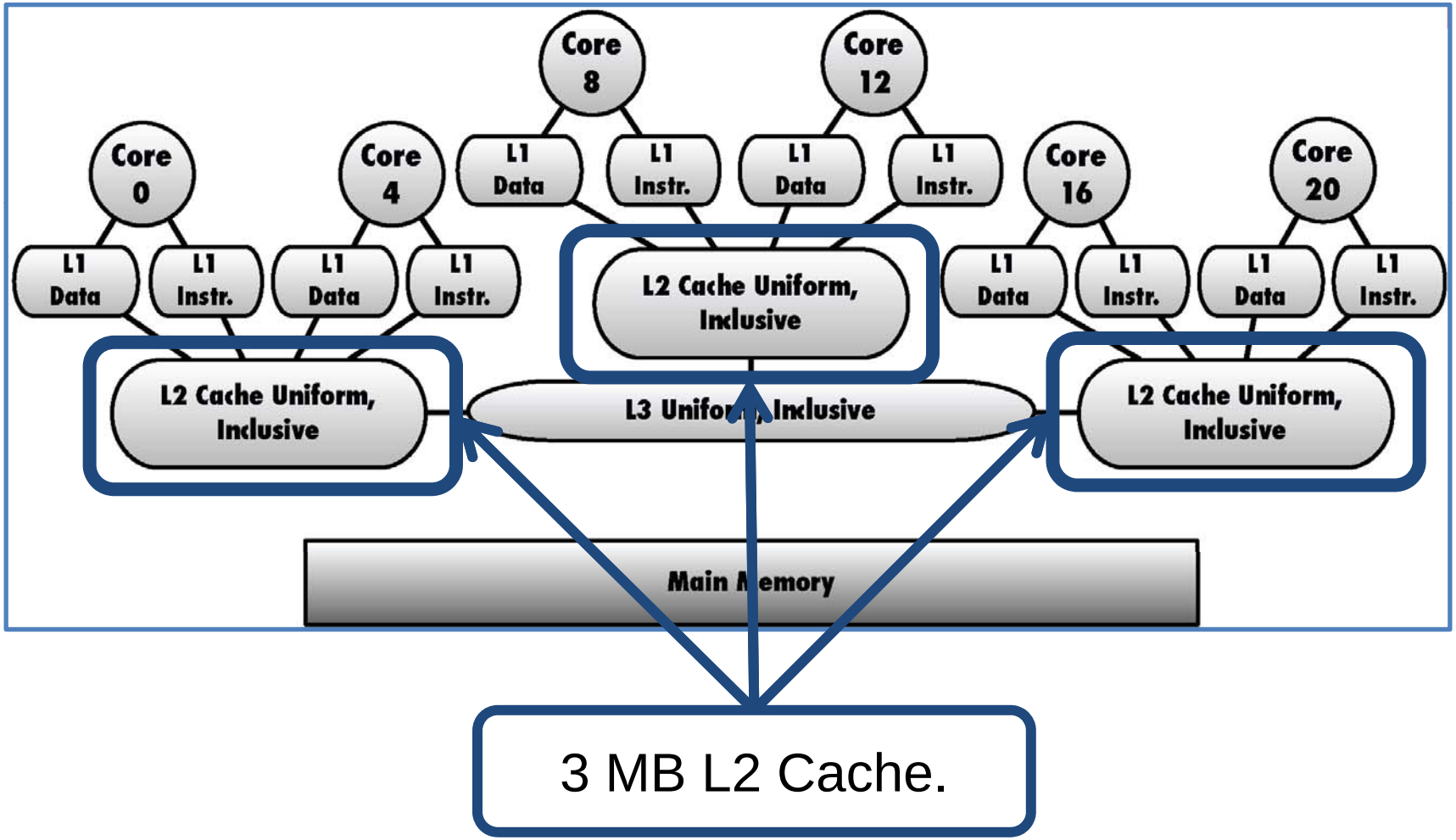
Test Platform

- Intel Xeon L7455 “Dunnington”:



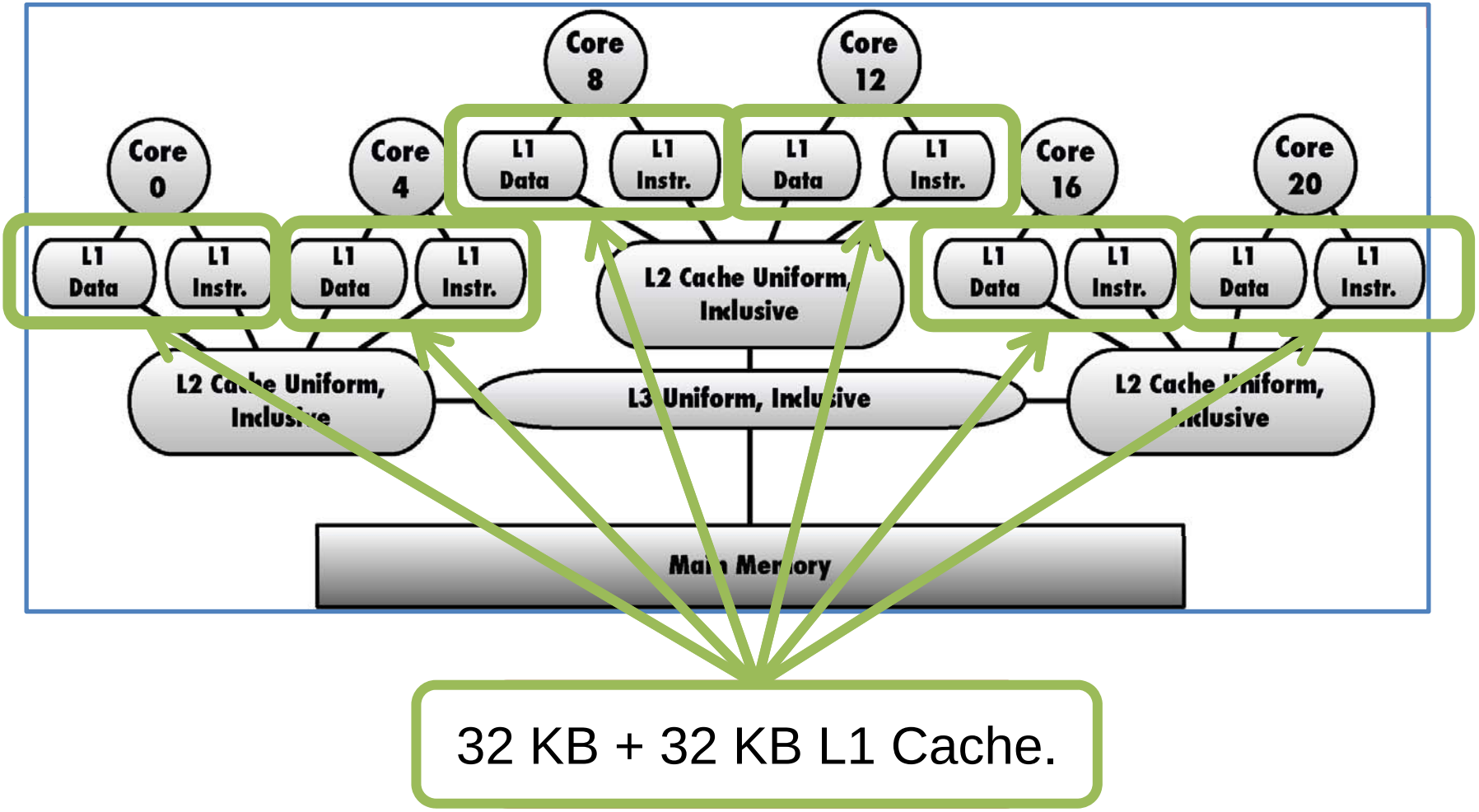
Test Platform

- Intel Xeon L7455 “Dunnington”:



Test Platform

- Intel Xeon L7455 “Dunnington”:



Study Setup

- Schedule-Sensitive method:
  - G-EDF algorithm (but can be applied to other algos).
  - TSS between 25 and 250, tasks randomly generated.
    - Uniform distribution. Periodic tasks with utilizations in [0.001, 0.1] and periods in [10,100] ms.
  - WSS in the range from 4 KB to 2048 KB.
  - Per-WSS write ratio 1/2 and 1/4.
- Synthetic method:
  - Single SCHED_FIFO task at the highest priority.
  - WSS in the range from 4 KB to 12 MB.
  - Per-WSS write ratio in the range from 0 to 1.
  - Preemption length uniformly distributed in [0,50] ms.

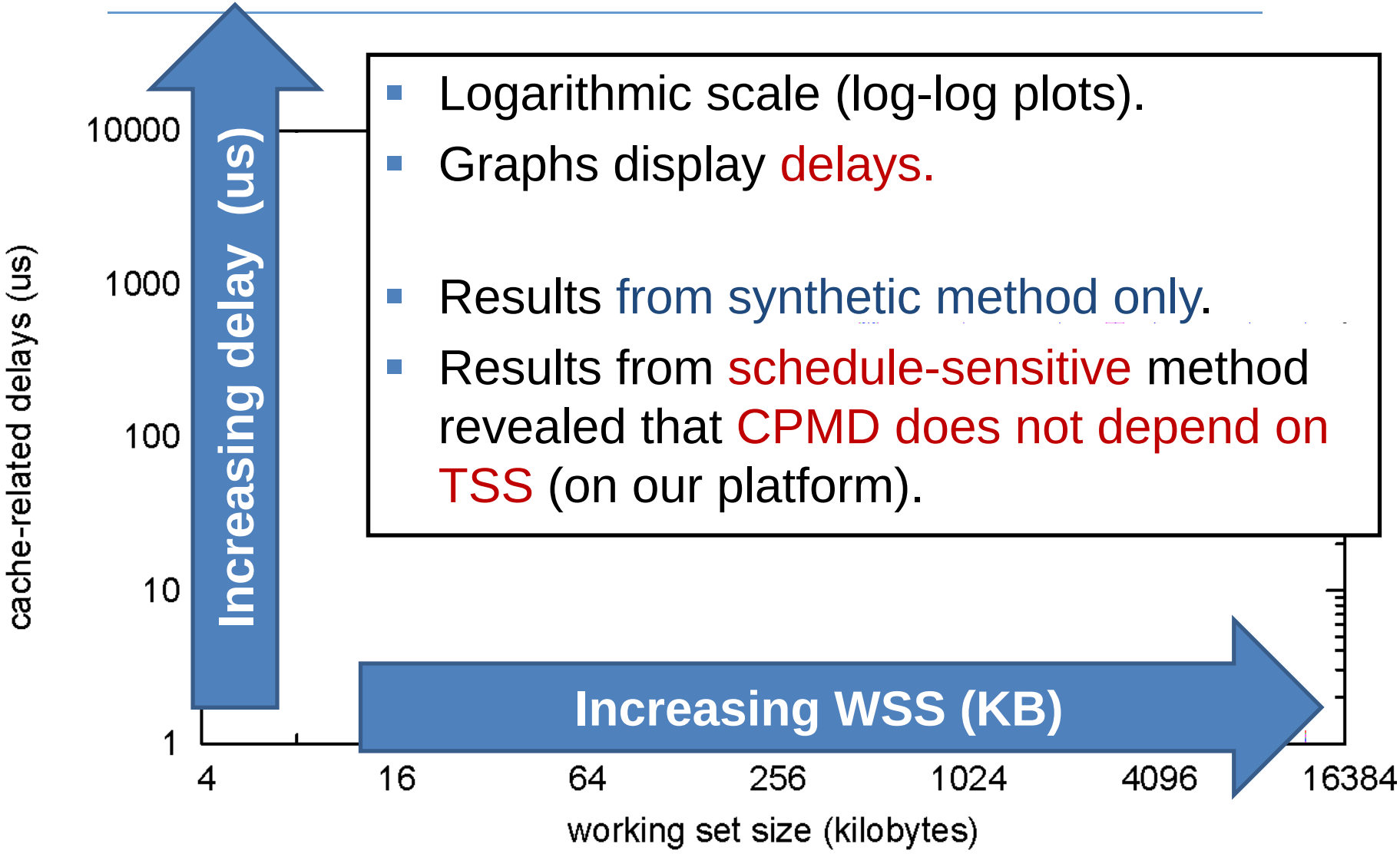
Study Setup

- Schedule-Sensitive method:
  - G-EDF algorithm (but can be applied to other algos).
  - TSS between 25 and 250 tasks randomly generated.
    - Uni
      [0.0
  - WSS
  - Per-W
- Synthet
  - Single SCHED_FIFO task at the highest priority.
  - WSS in the range from 4 KB to 12 MB.
  - Per-WSS write ratio in the range from 0 to 1.
  - Preemption length uniformly distributed in [0,50] ms.

Tested *two* configurations for each method:

- **Idle system.**
- **Under load.**

Results

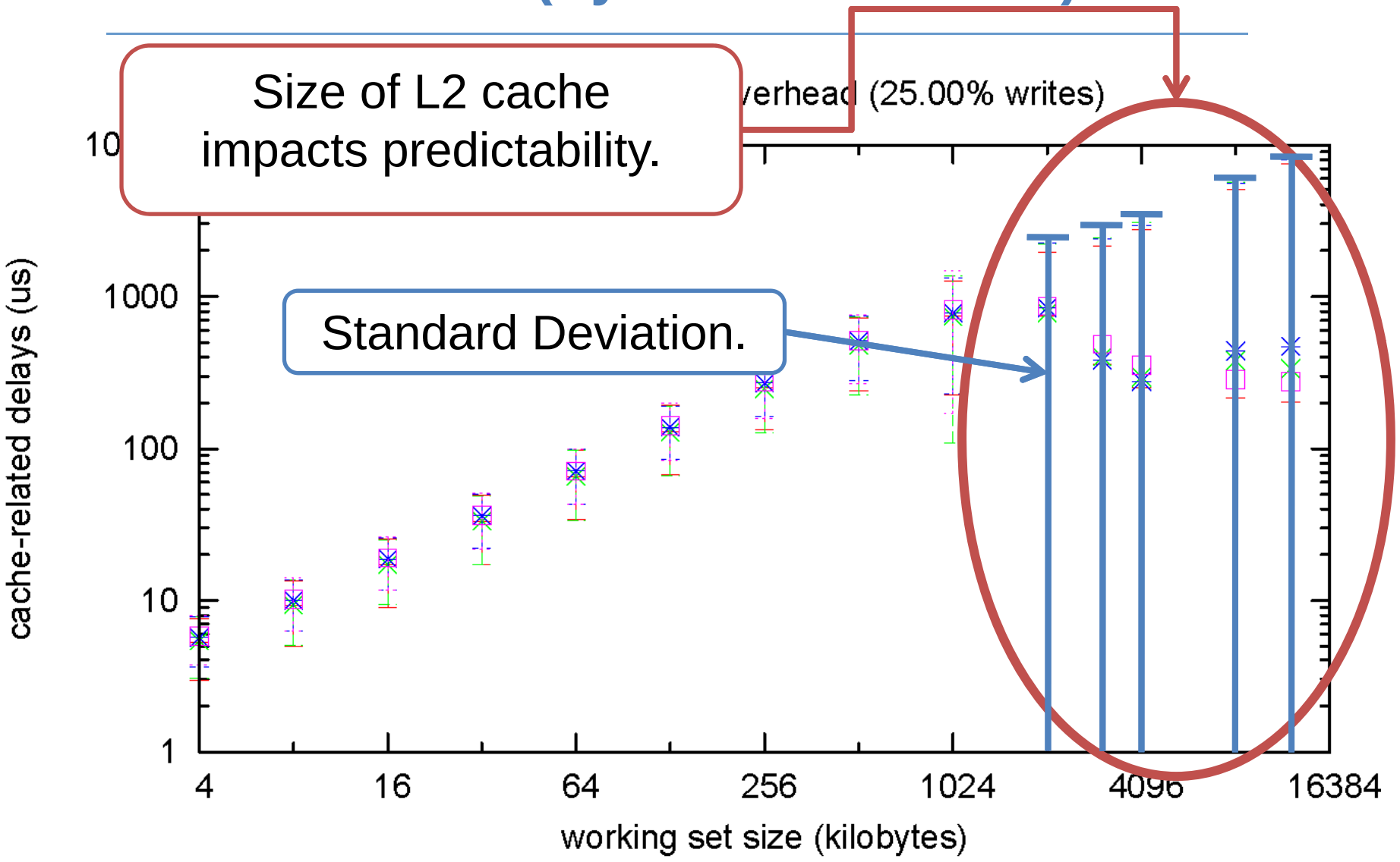


- Logarithmic scale (log-log plots).
- Graphs display delays.

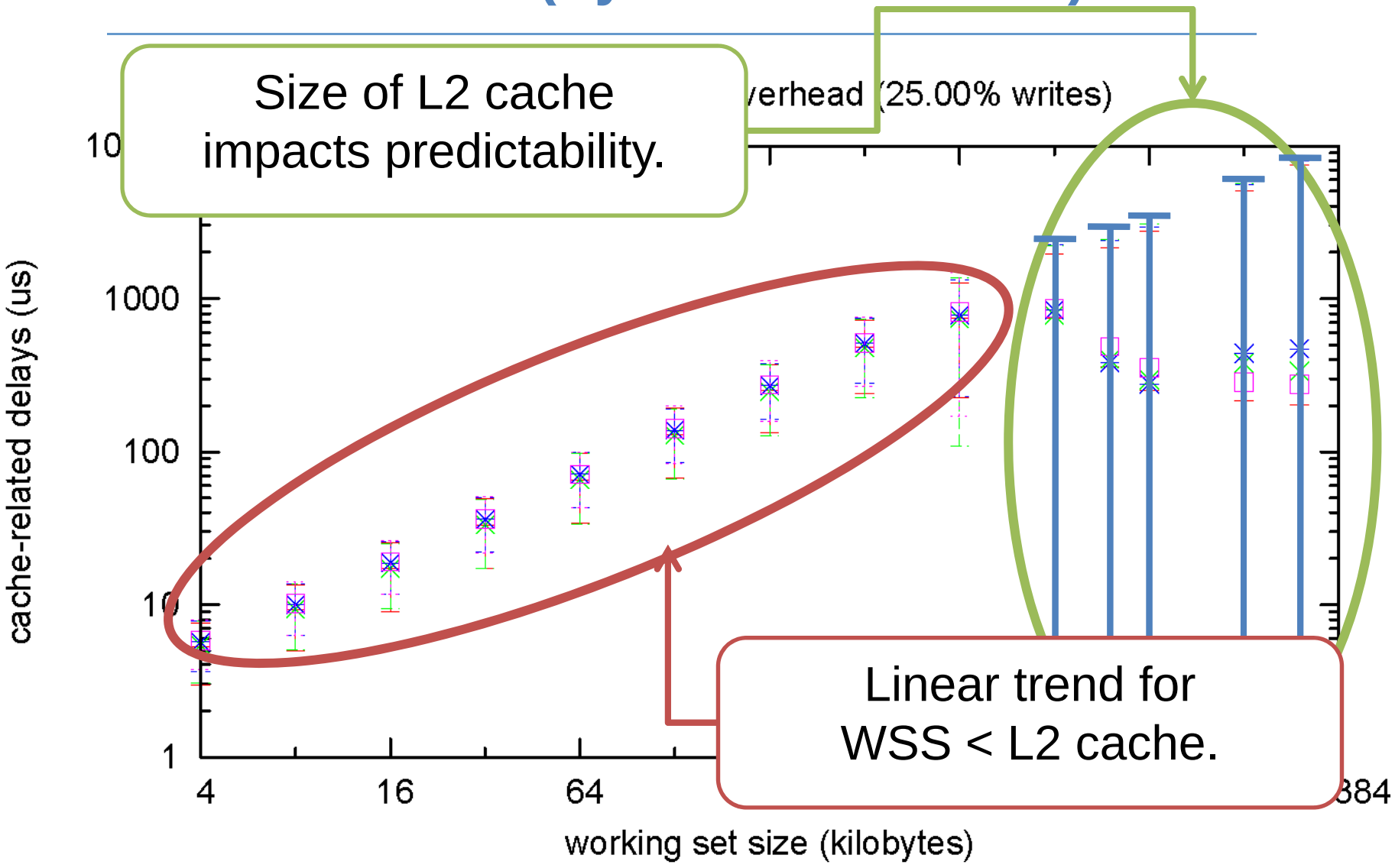
- Results from synthetic method only.
- Results from schedule-sensitive method revealed that CPMD does not depend on TSS (on our platform).

“Higher is worse”.

Results (System under load)

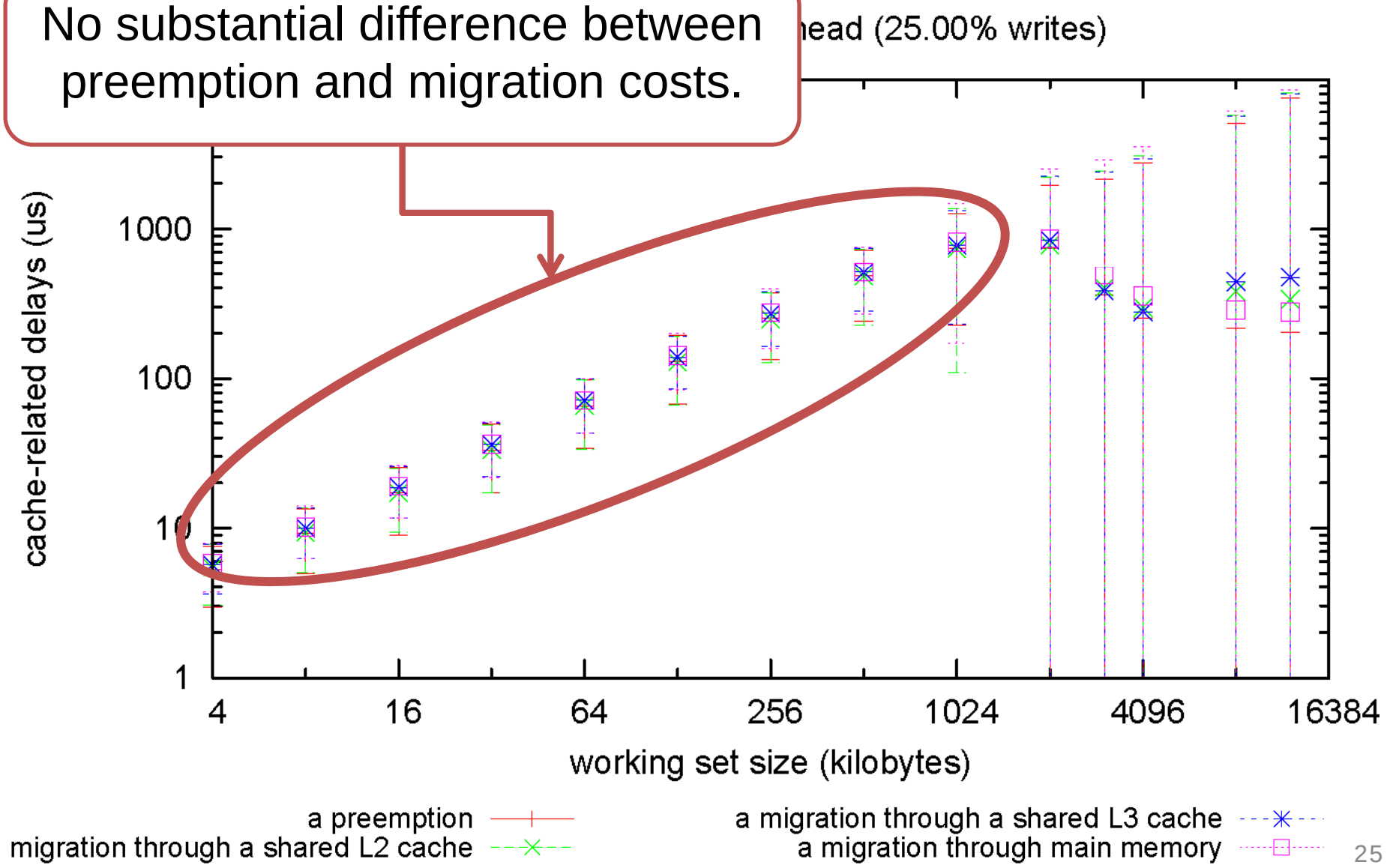


Results (System under load)



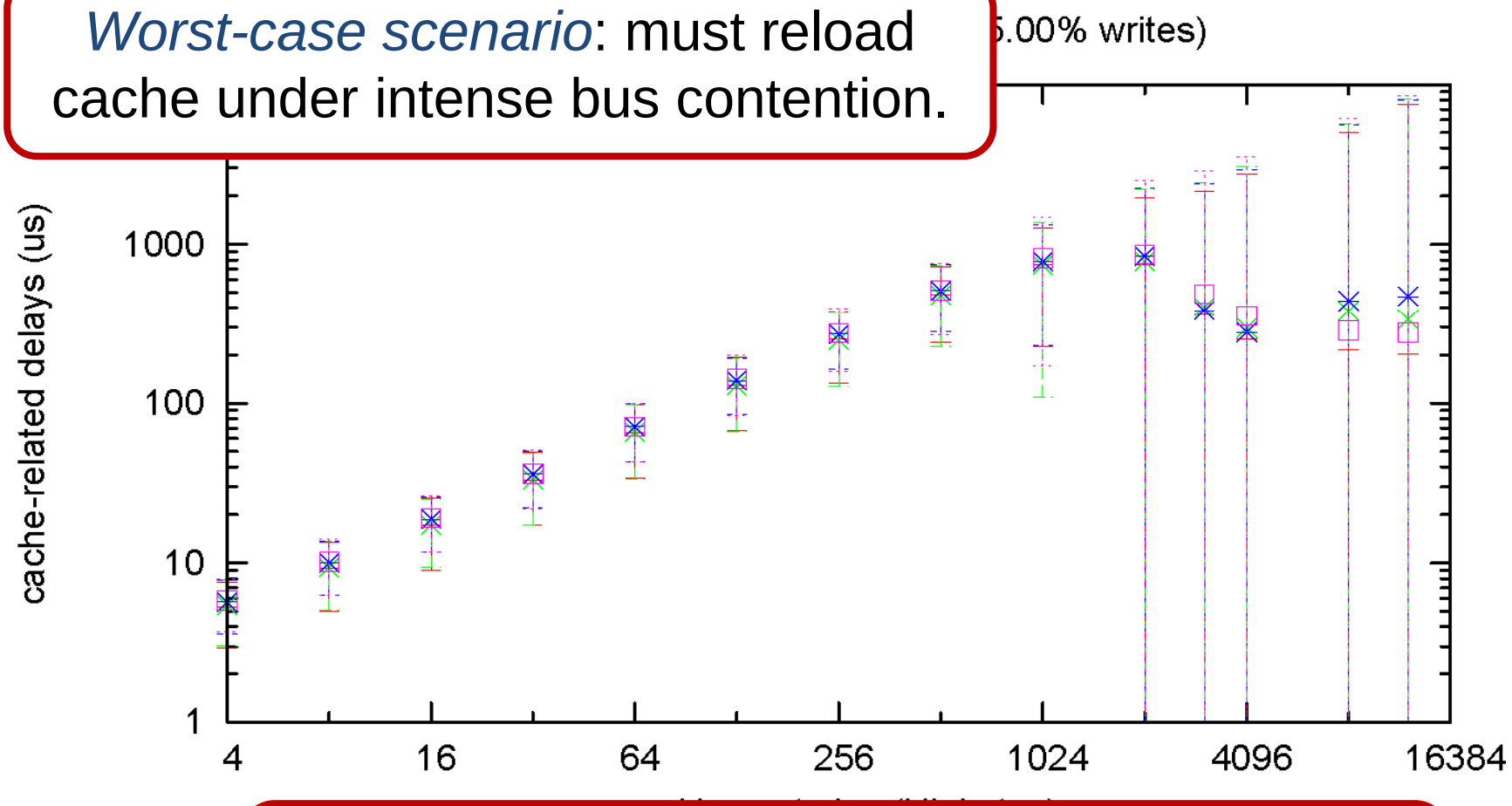
Results (System under load)

No substantial difference between preemption and migration costs.



Results (System under load)

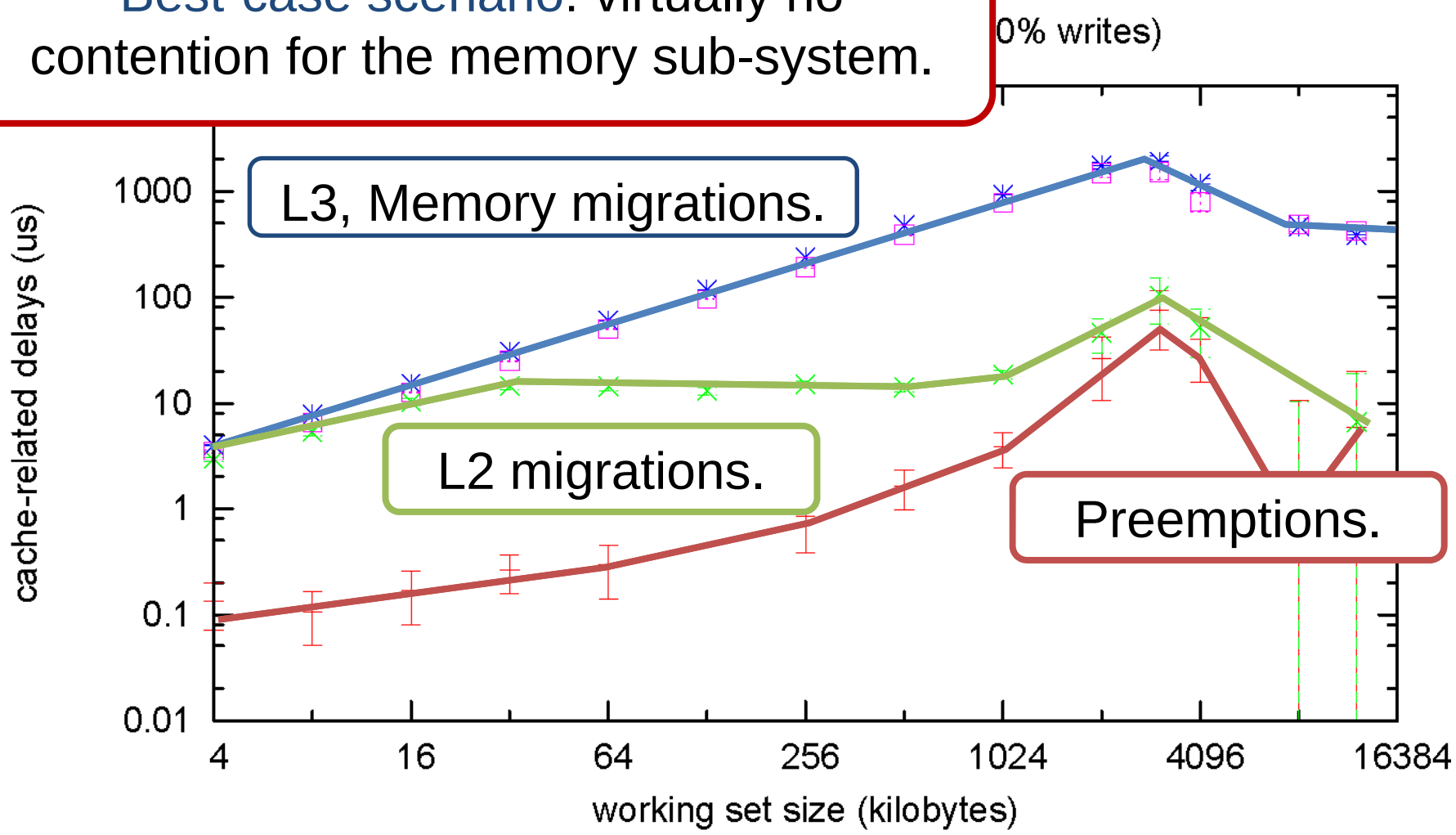
Worst-case scenario: must reload cache under intense bus contention.



Applies unless **no** background activity **and** only small WSS for all tasks.

Results (Idle system)

Best-case scenario: virtually no contention for the memory sub-system.

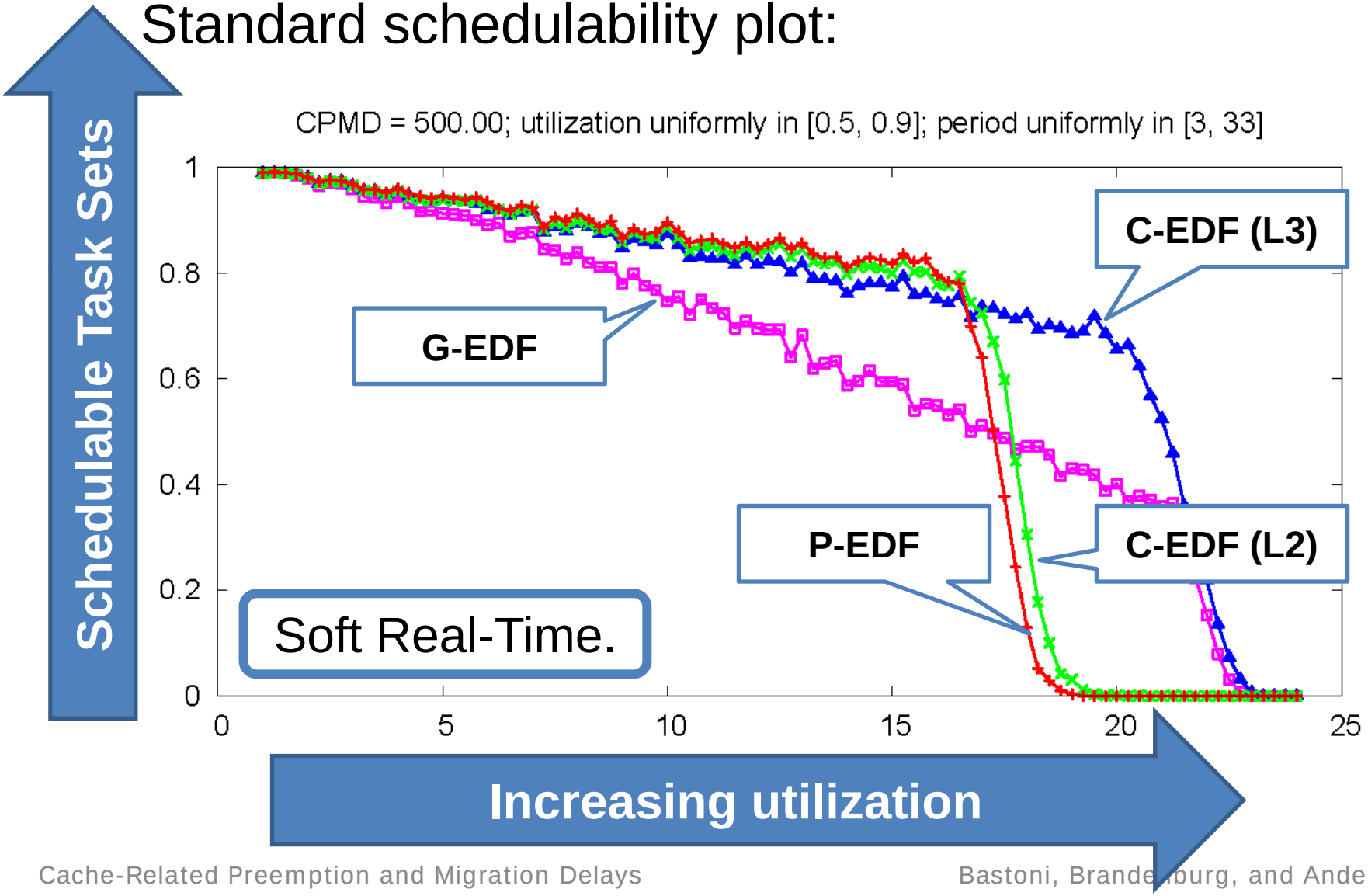


Outline

- Cache-related preemption and migration delays (CPMD).
- Two methods to measure CPMD.
- Experimental results and discussion.
- Impact on schedulability (sketch).

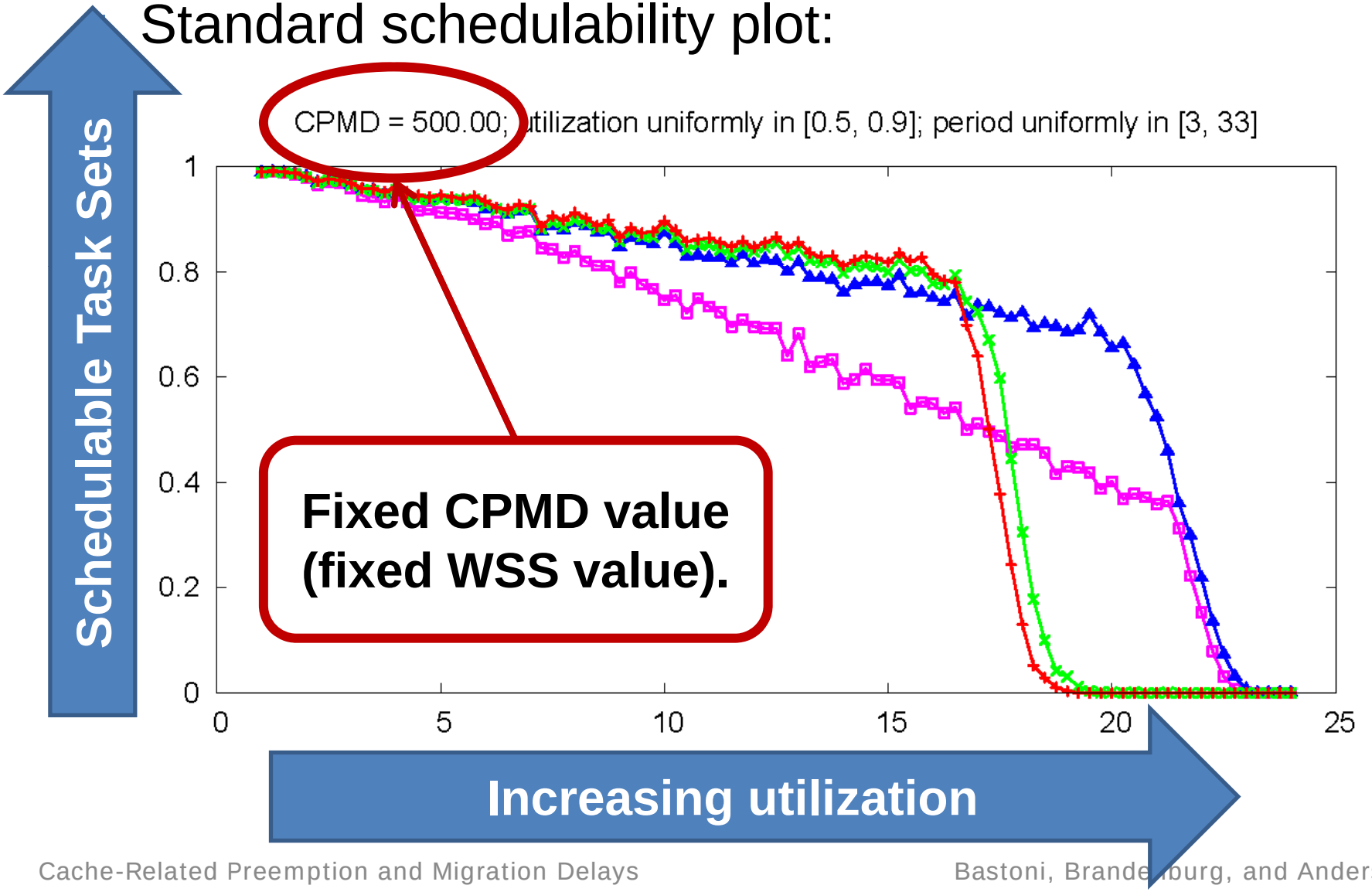
Impact on Schedulability

Standard schedulability plot:



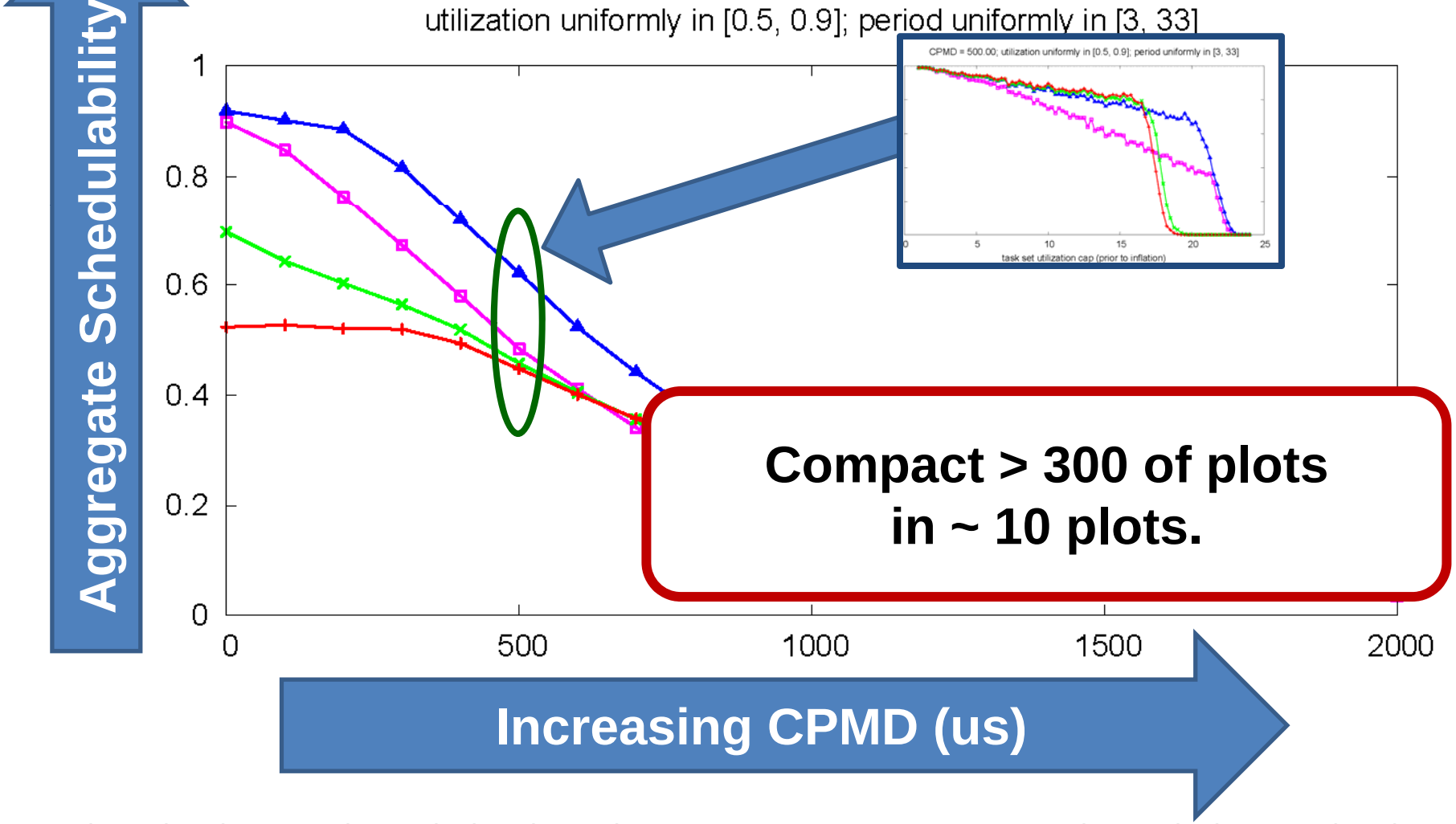
Impact on Schedulability

Standard schedulability plot:

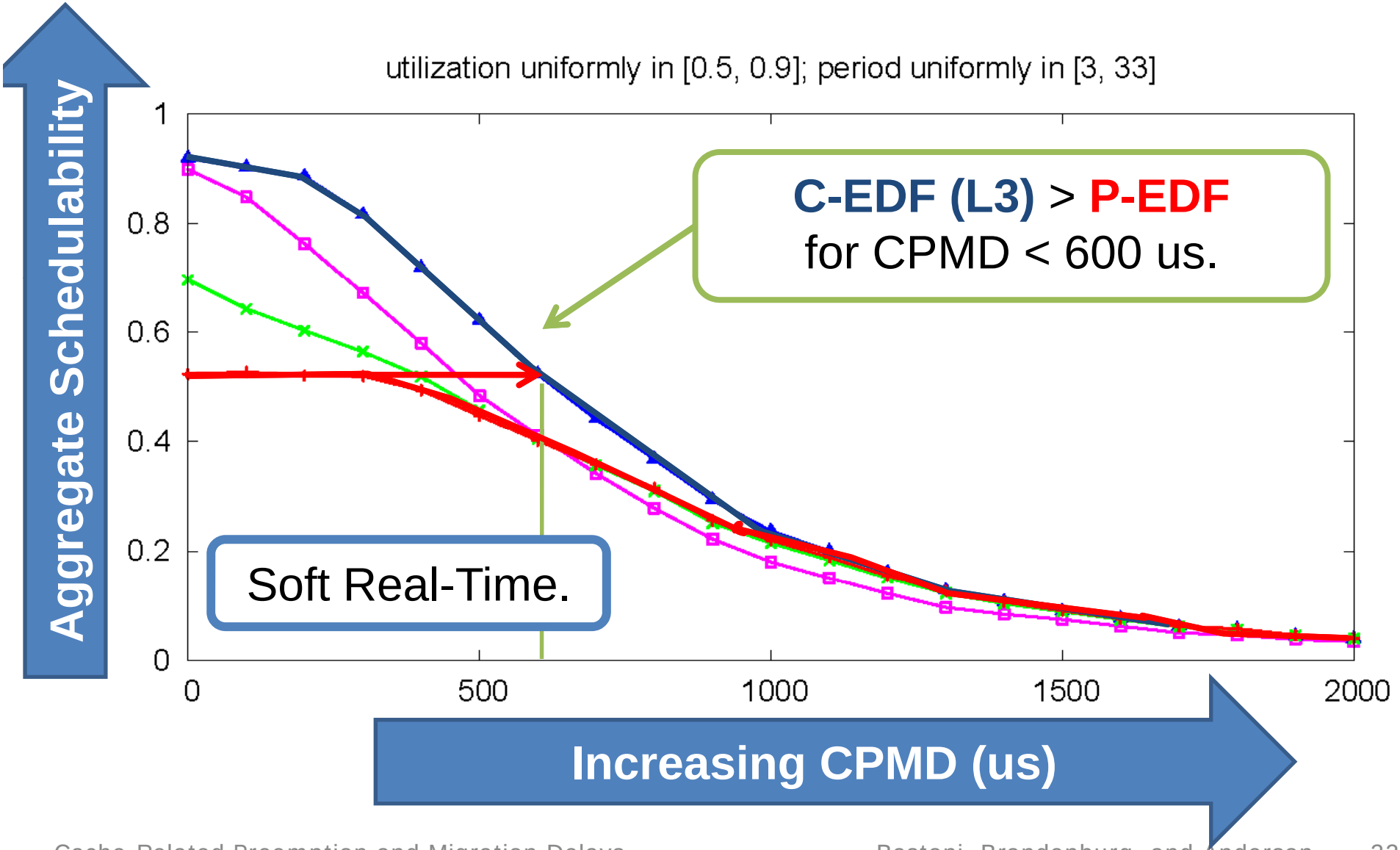


Impact on Schedulability

Weighted schedulability plot:



Evaluation of Scheduling Algorithms



Conclusions

- Empirical approximation of CPMD:
  - Schedule-sensitive method.
  - Synthetic method.

- CPMD strongly impacts evaluation of schedulers:
  - Preemptions are not necessarily (much) cheaper than migrations (for worst-case overheads).
  - If there is memory bus contention, then this is also true for average-case overheads.

Future Work

- Validate TSC-based results with performance counters.
- Apply the methodologies on NUMA and embedded platforms.
- Investigate impact of bus locking on CPMD.
  - For example: DMA transfers, atomic instructions etc.

LITMUS^{RT}

Linux Testbed for Multiprocessor Scheduling in Real-Time Systems

http://www.cs.unc.edu/~anderson/litmus-rt/

Thank You!