Performance Effects of Cache Sharing (And how to measure them)

Vlastimil Babka

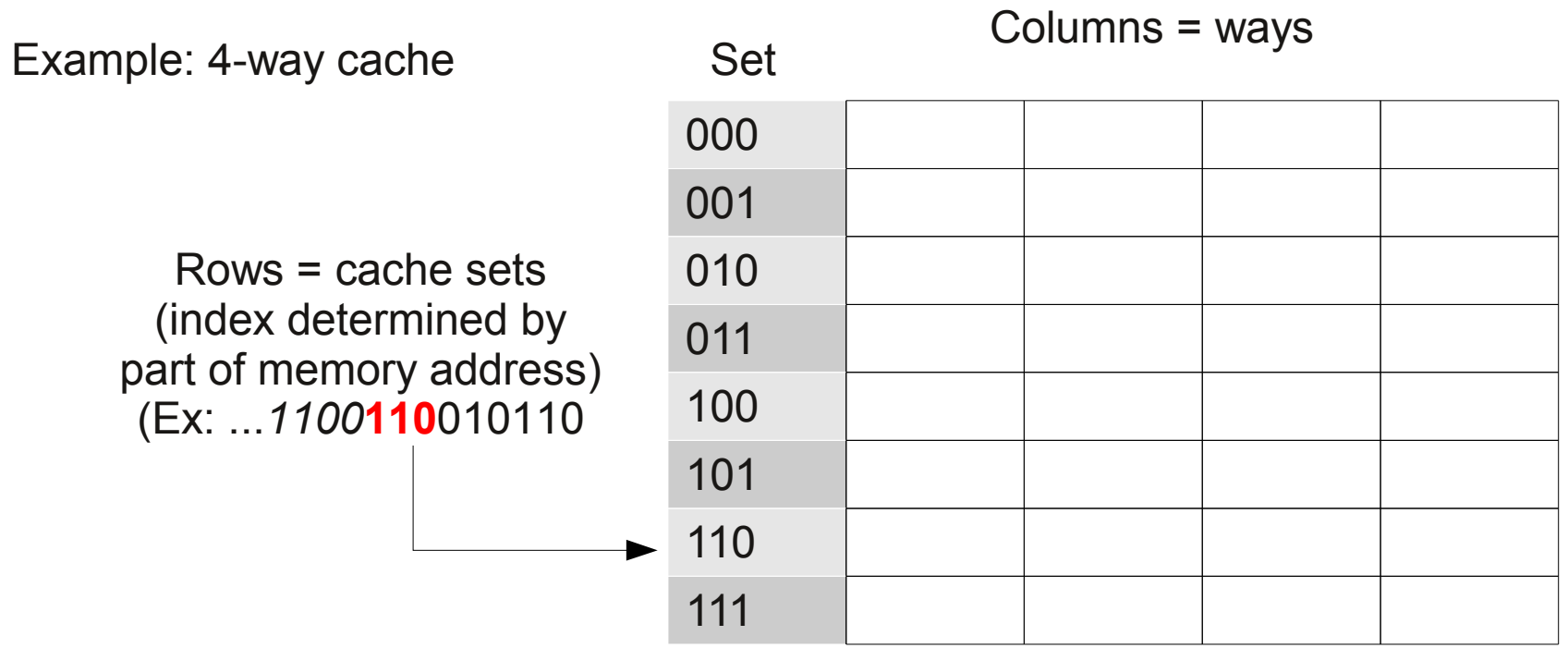
Res Informatica I2 Seminar

DEPARTMENT OF DISTRIBUTED AND DEPENDABLE SYSTEMS
FACULTY OF MATHEMATICS AND PHYSICS
CHARLES UNIVERSITY IN PRAGUE

http://d3s.mff.cuni.cz

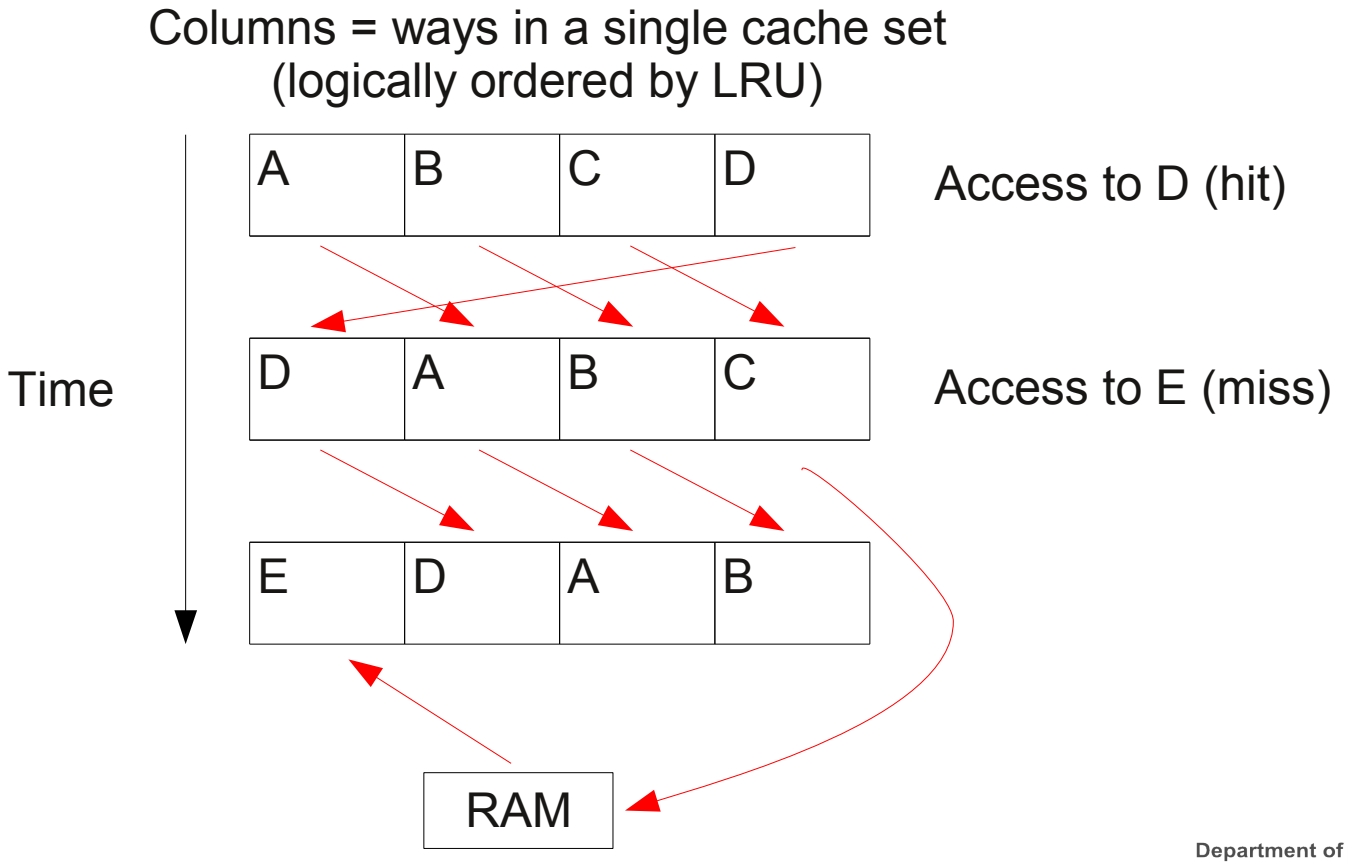
Processor Cache Essentials: Associativity

- Cache = a number of fixed size (64B) blocks (lines) holding least recently accessed data
- Typically set-associative, with (pseudo-)LRU replacement policy



Processor Cache Essentials: LRU Policy

- Example: 4-way cache with LRU
- A,B,C,D,E – memory blocks that map to the same cache set

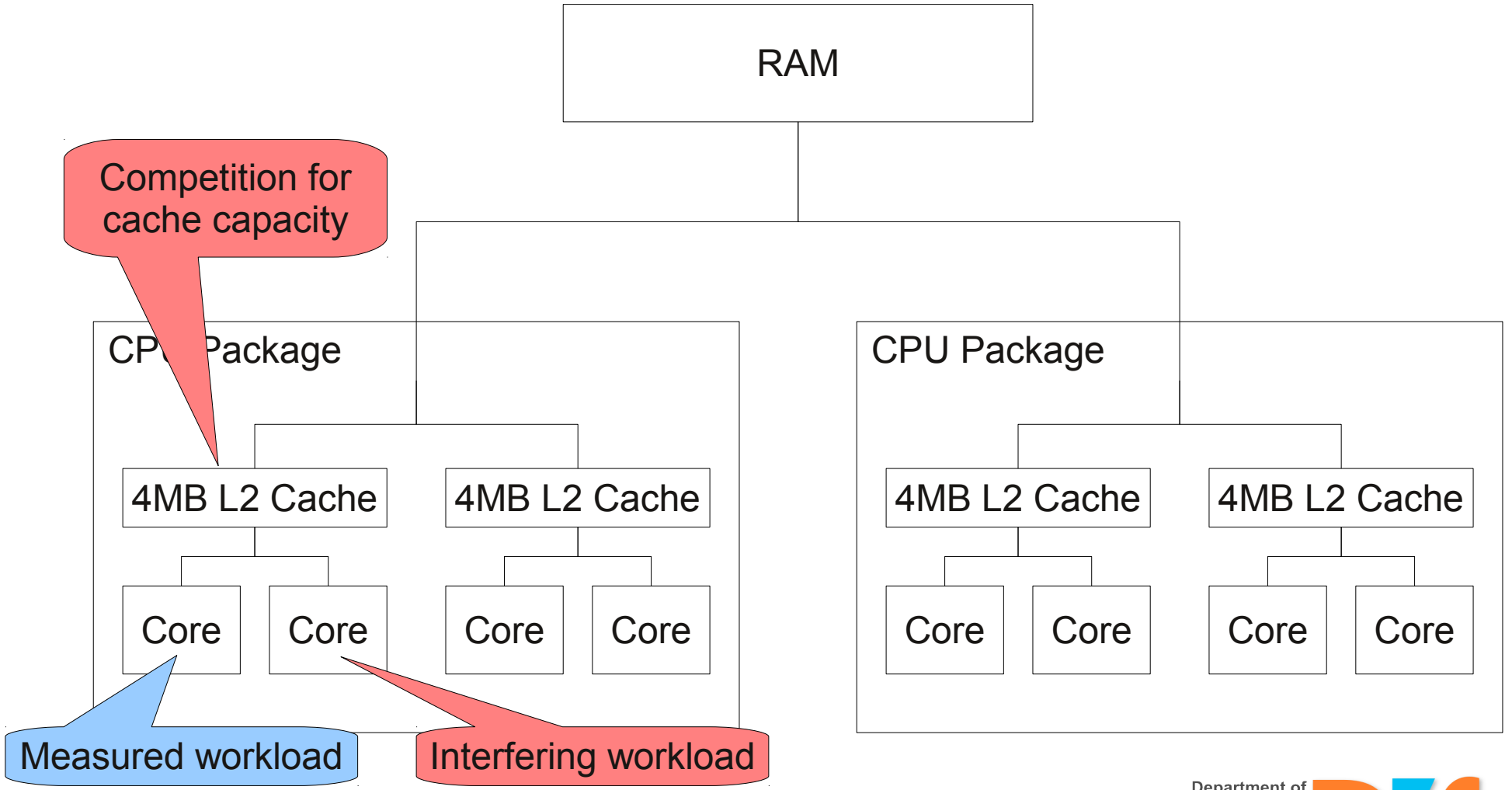


Hardware Prefetching

- Idea: predict what cache line will soon be requested and fetch it to the cache in advance
  - Mask the latency of a cache miss
- Implementation: detect and track linear access patterns (possibly with a stride)
- Intel L2: up to 12 (16?) upstream, 4 (?) downstream sequences can be tracked
  - Prefetch up to 8 accesses ahead
  - Bounded by 4 KB pages

Experiment Platform and Setup

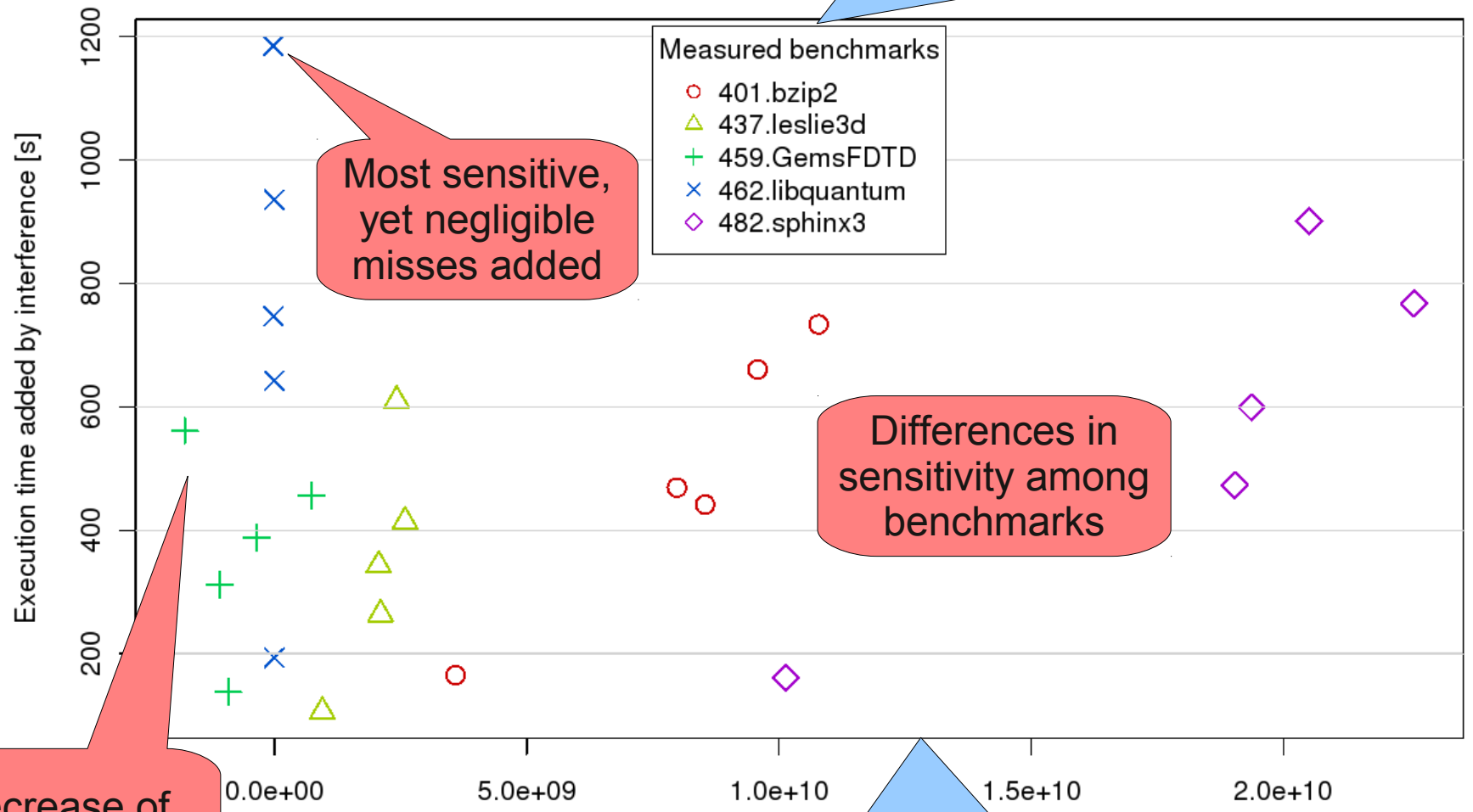
- Dual Quad-Core Intel Xeon E5345



Two benchmarks sharing an L2 cache

Execution time (seconds) added due to parallel execution of another benchmark

Subset of SPEC CPU2006 used in pairs as both measured and interfering workload. Points of same color+shape differ in interfering workload.



Most sensitive, yet negligible misses added

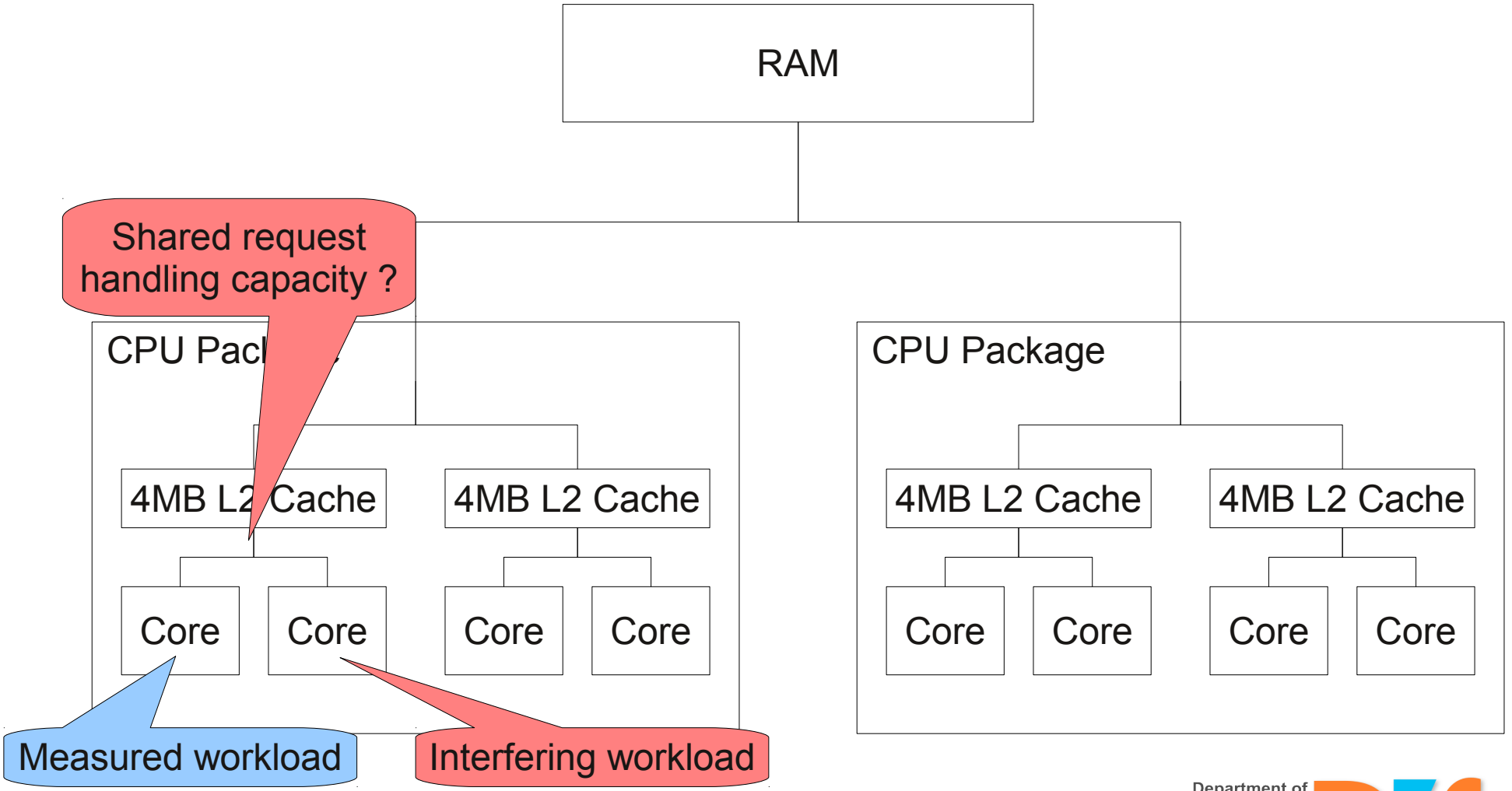
Differences in sensitivity among benchmarks

Decrease of misses?

L2 cache misses added due to parallel execution of another benchmark

Department of Distributed and Dependable Systems

Factor: Request Handling Capacity



Factor: Request Handling Capacity

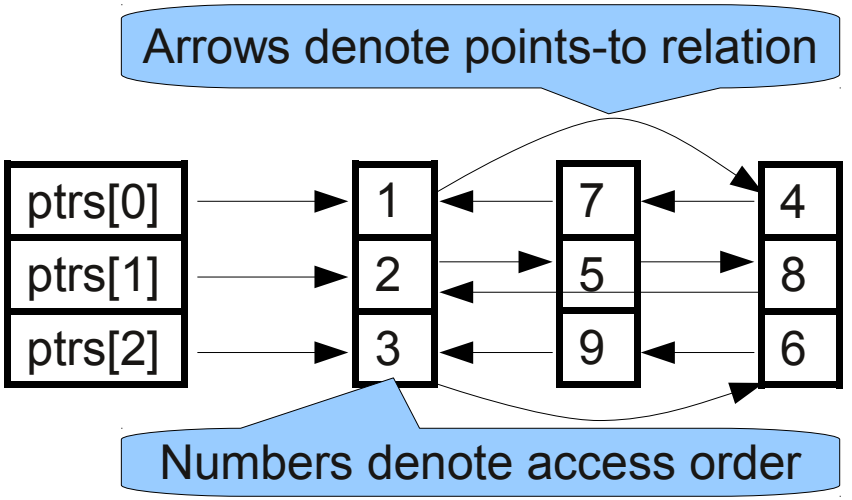
- Limits concurrent access to the L2 cache
- Sparse details in vendor documentation

Experiment:

- Measured workload that hits in the L2 cache
- Interfering workload issuing multiple parallel memory accesses
  - Amount of parallelism controlled
  - Hits or misses in shared L2 cache
    - But minimize competition for cache capacity
  - Minimize prefetching

Interfering Workload Overview

- Multipointer random walks
  - Accesses via one pointer **dependent**
  - Accesses via different pointers **independent**
  - Number of pointers used controls parallelism



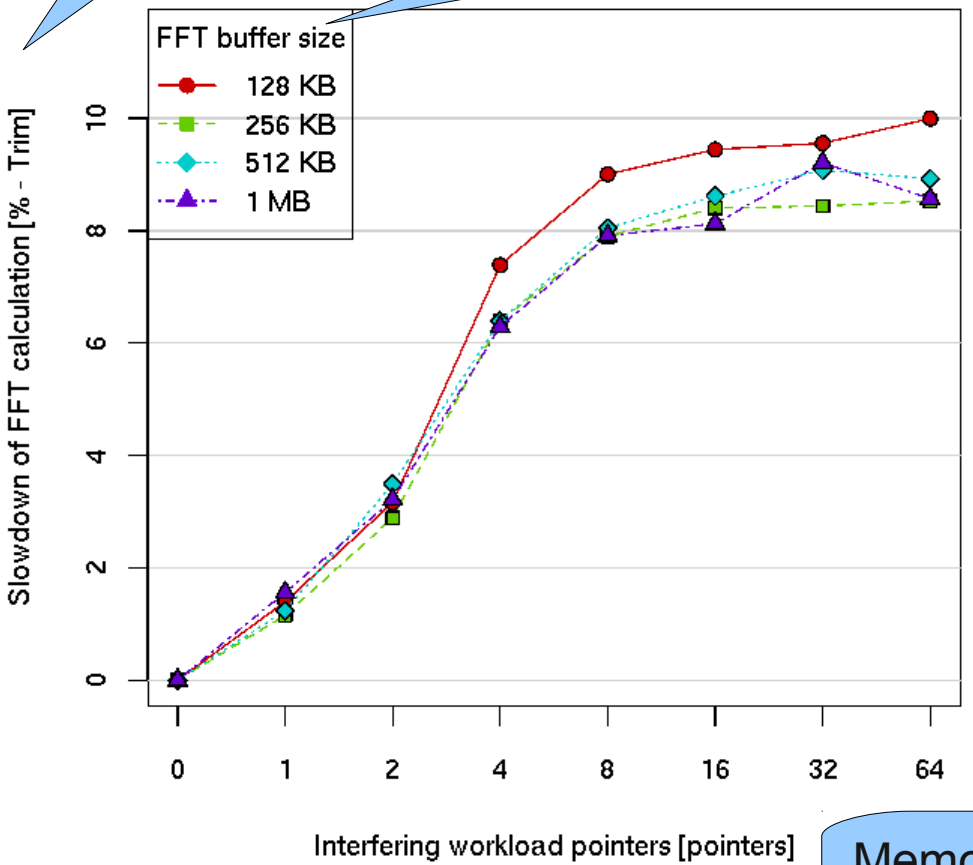
- Randomized to minimize prefetching etc.
- Hits in L2 cache: exceeding L1 cache capacity
- Miss in L2 cache: each pointer uses only addresses mapping to the same cache set
  - Need to deal with physical L2 cache indexing

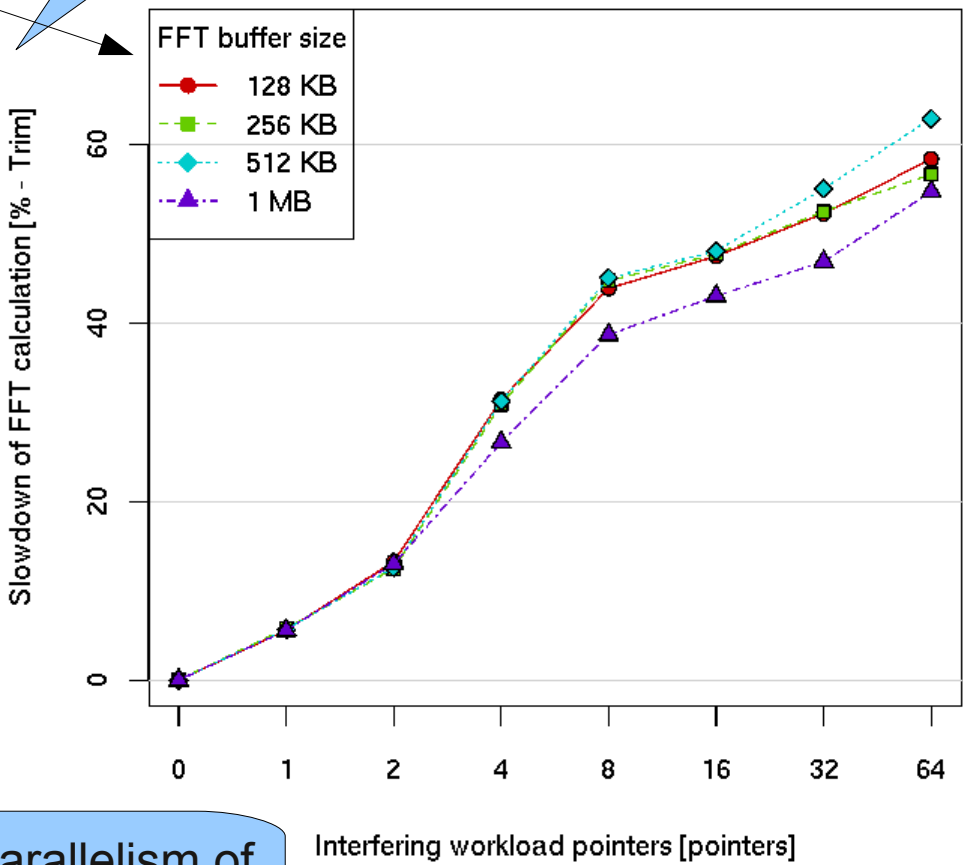
Results: L2 Request Handling (FFTW)

Slowdown with interference **hitting** in L2 cache

Buffer fits in L2 cache – FFTW **hits**

Slowdown with interference **missing** in L2 cache

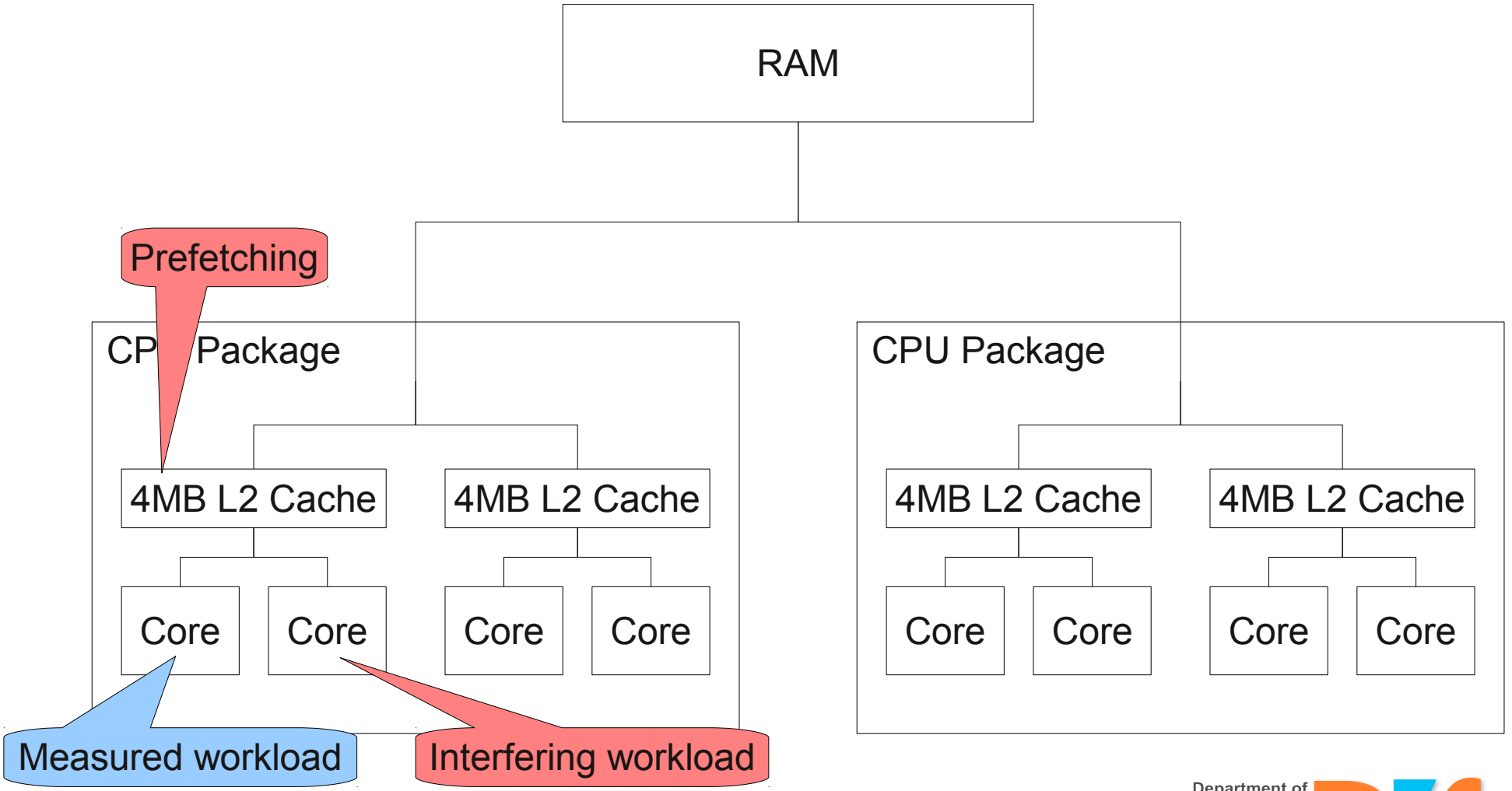




Memory parallelism of **interfering** workload

Department of Distributed and Dependable Systems

Factor: Hardware Prefetch Competition



Factor: Hardware Prefetch Competition

- Limited number of prefetch stream trackers
- Prefetch misses handled with lower priority
- Some prefetches are inefficient

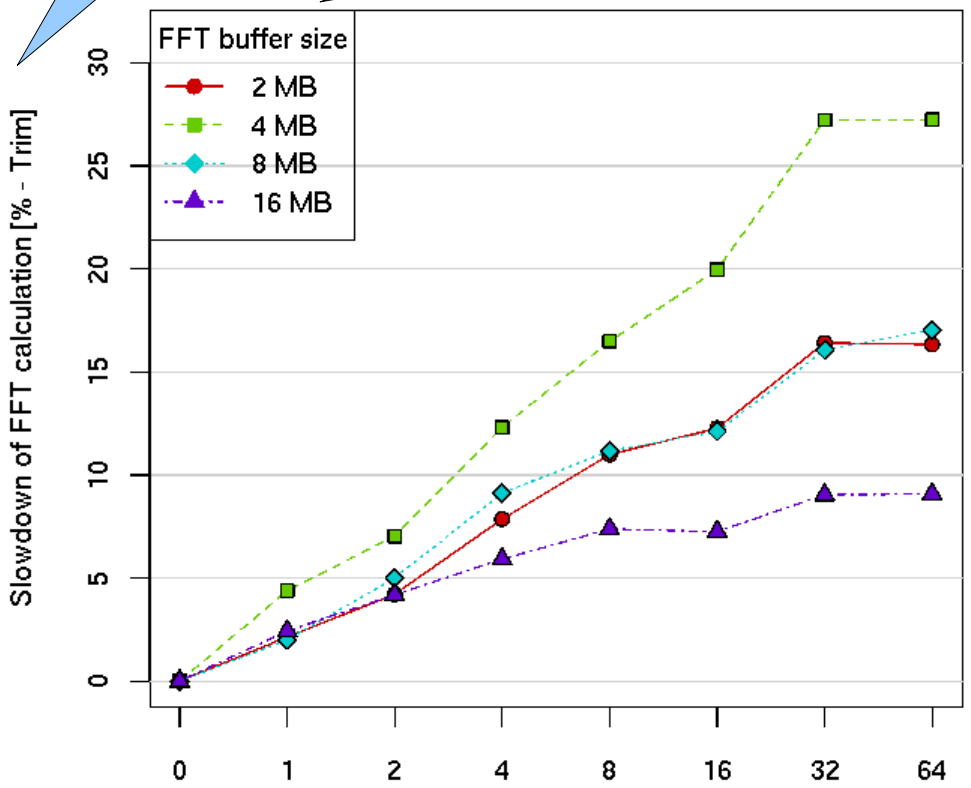
Experiment:

- Measured workload that misses in L2 cache may benefit from prefetches and thus be sensitive
- Interfering workload hitting in shared L2 cache
  - Competition for capacity still minimal
- Caveat: implies also sharing of request handling capacity

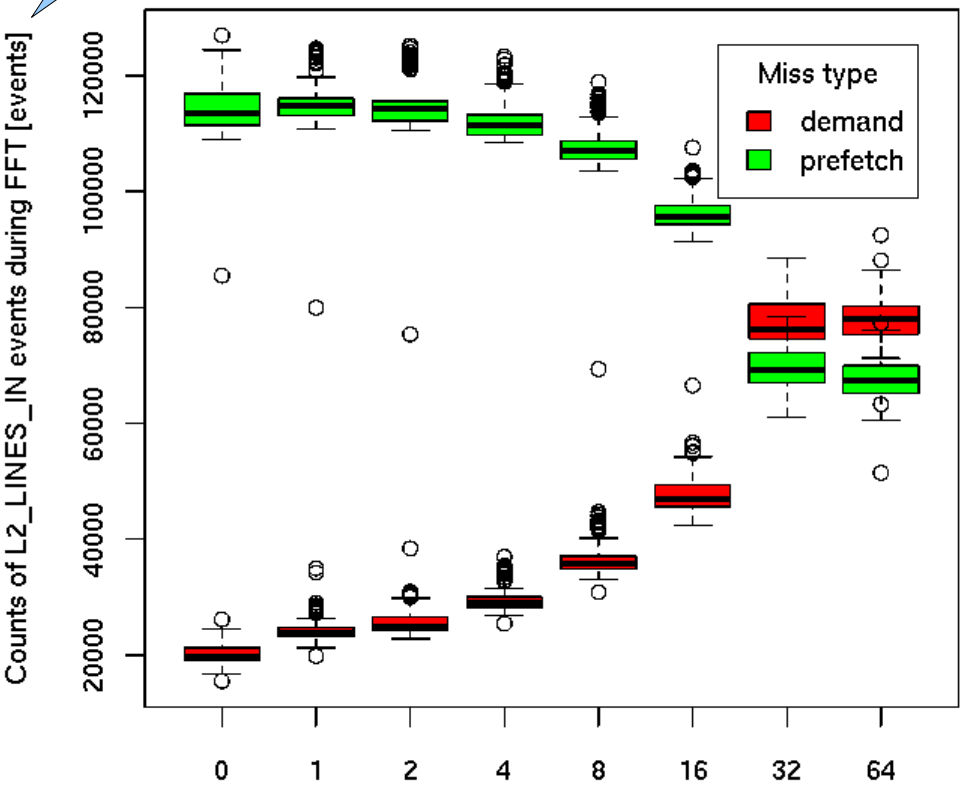
Request Handling + Prefetch (FFT)

Slowdown with interference **hitting** in L2 cache

Buffer exceeds L2 cache – FFTW **misses**



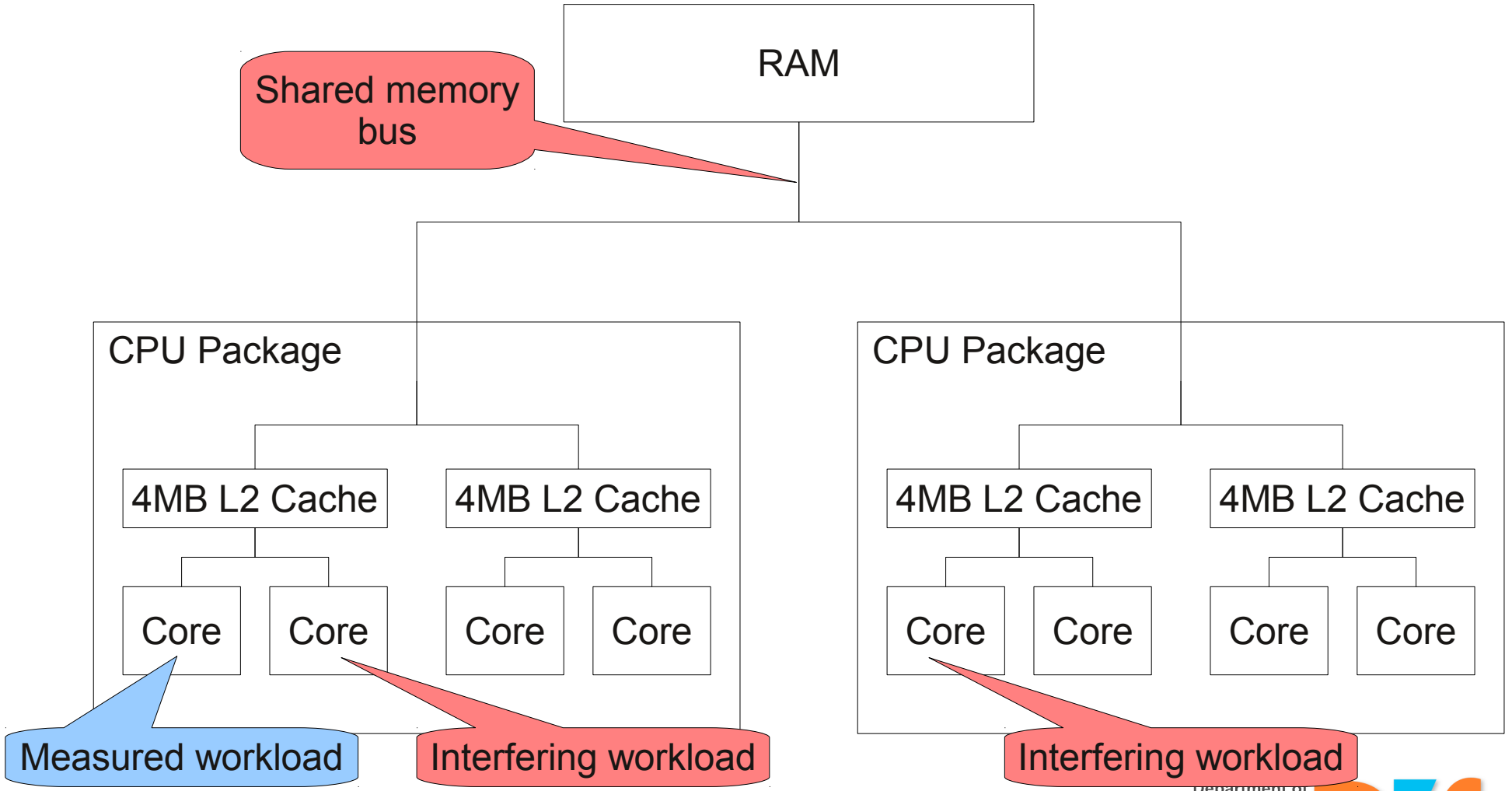
Number of L2 misses with 4 MB buffer



Memory parallelism of **interfering** workload

Department of Distributed and Dependable Systems

Factor: Memory Bus Contention



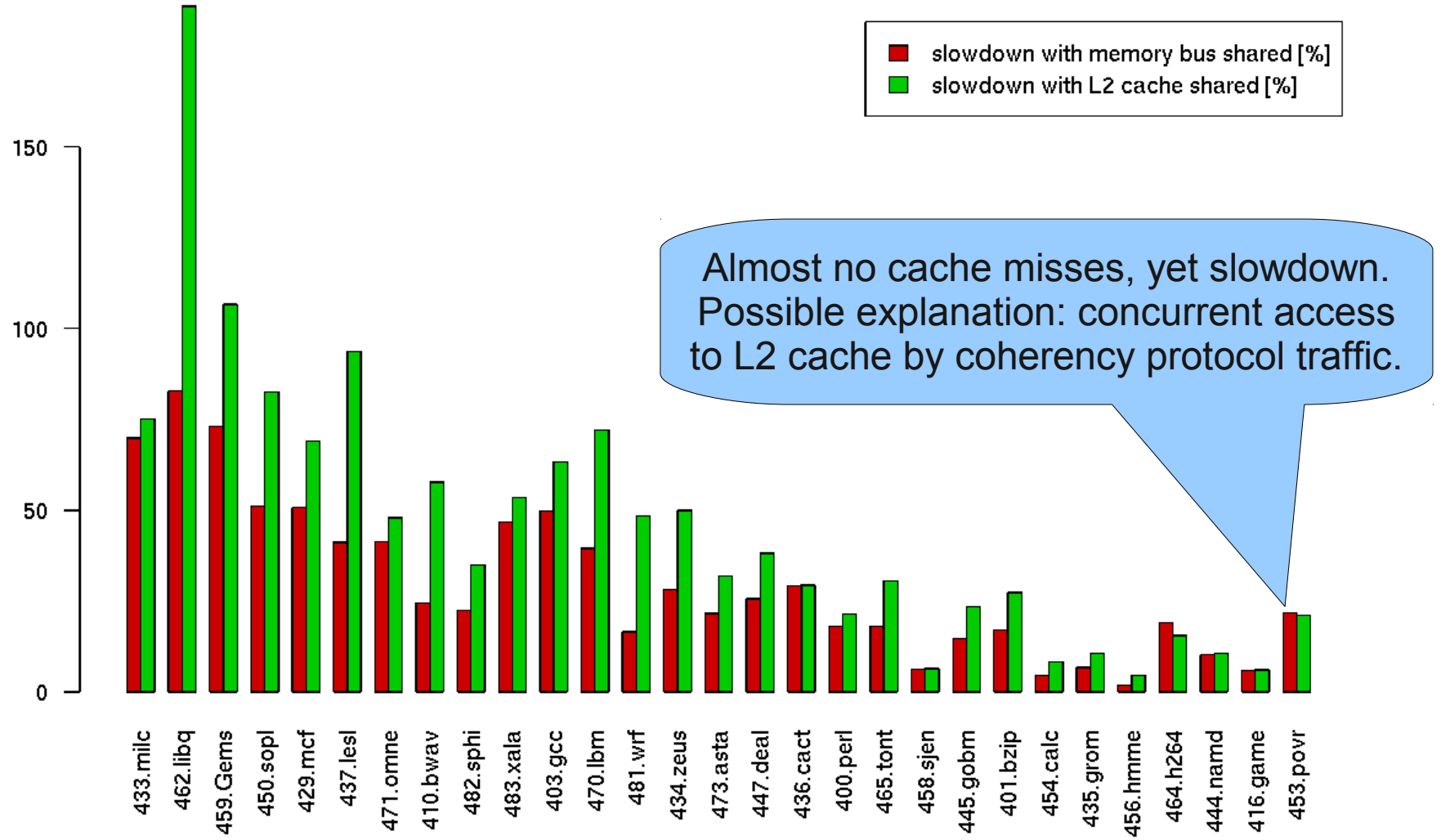
Factor: Memory Bus Contention

- Associated with cache misses
  - Not necessarily due to shared cache capacity
  - With shared cache, influence cannot be isolated from cache request handling capacity and hardware prefetch competition

Experiment: random multipointer walk missing in L2 cache as interfering workload

- Running either on core sharing L2 cache and memory bus, or just memory bus
- Minimize competition for cache capacity

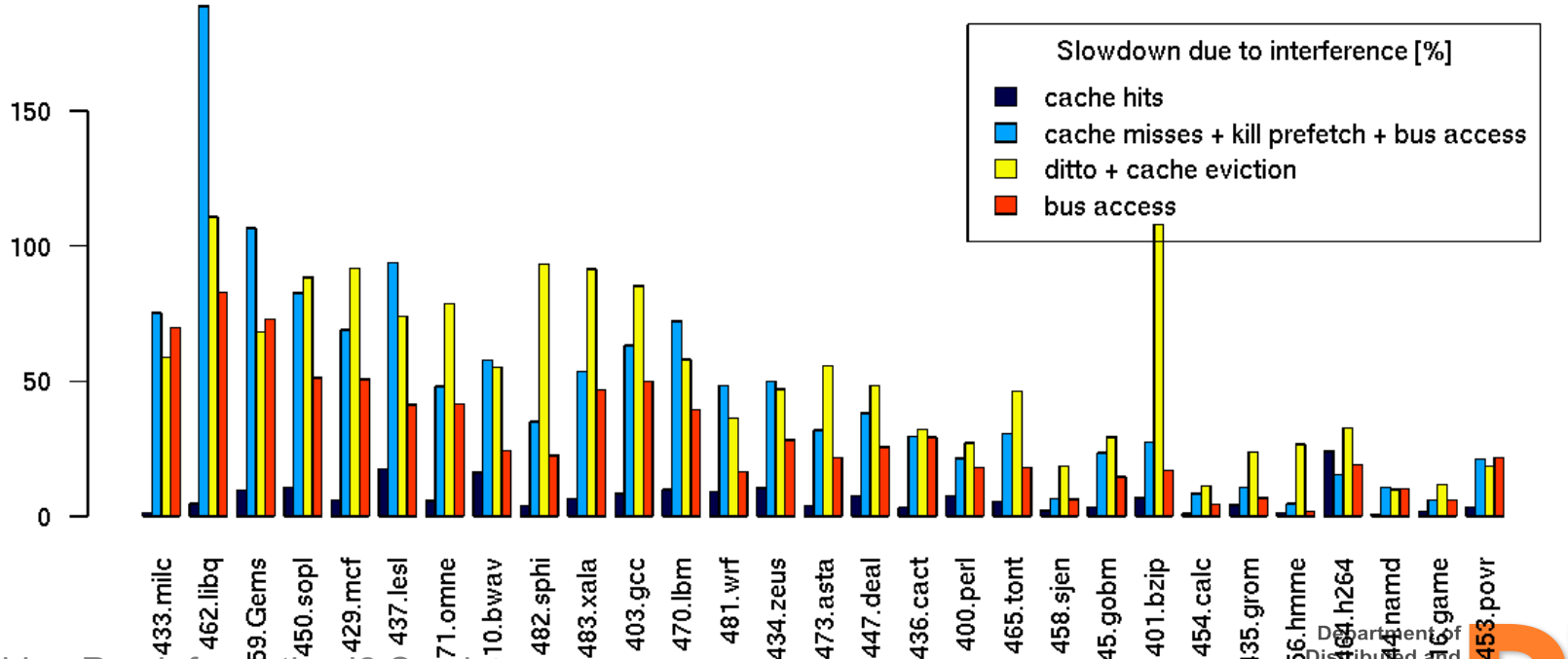
Results: Memory Bus (SPEC CPU2006)



SPEC CPU2006 benchmarks sorted by isolated miss rate in descending order

Results summary (SPEC CPU2006)

Nature of interfering workload	Slowdown		
	Min	Max	Median
Hits in L2 cache + kills prefetch	0.5 %	24 %	6 %
Misses in L2 cache + kills prefetch + accesses memory bus	4.5 %	190 %	35 %
Ditto + evicts from cache	10 %	111 %	48 %
Accesses memory bus + causes coherency requests	2 %	83 %	22 %



Thank you ...

Related publications / more details:

•Babka V., Marek L., Tůma P.: *When Misses Differ: Investigating Impact of Cache Misses on Observed Performance*, ICPADS 2009, Shenzhen, China, IEEE, Dec 2009

• Q-ImPrESS Deliverable D3.3: *Resource Usage Modeling* http://www.q-impress.eu/wordpress/publications/

• Babka, V., Libic, P., Tuma, P.: *Timing Penalties Associated with Cache Sharing*, MASCOTS 2009, London, UK, IEEE, Sep 2009

• Babka, V., Tuma, P.: *Investigating Cache Parameters of x86 Family Processors*, SPEC Benchmark Workshop 2009, Austin, TX, USA, Springer, Jan 2009

• Babka, V.: *Cache Sharing Sensitivity of SPEC CPU2006 Benchmarks*, Tech. Report No. 2009/3, Charles University https://d3s.mff.cuni.cz/publications/Babka-tr-cpu2006-sensitivity.pdf

• Measurement framework: http://d3s.mff.cuni.cz/projects/performance_evaluation/#resource_sharing

Department of Distributed and Dependable Systems