"A Survey on Cache Analysis for Real-Time System", from M. Lv, N. Guang, W. Yi, J. Reineke, R. Wilhelm

Valentin Touzeau

February 21, 2016

Motivations

Real-time systems

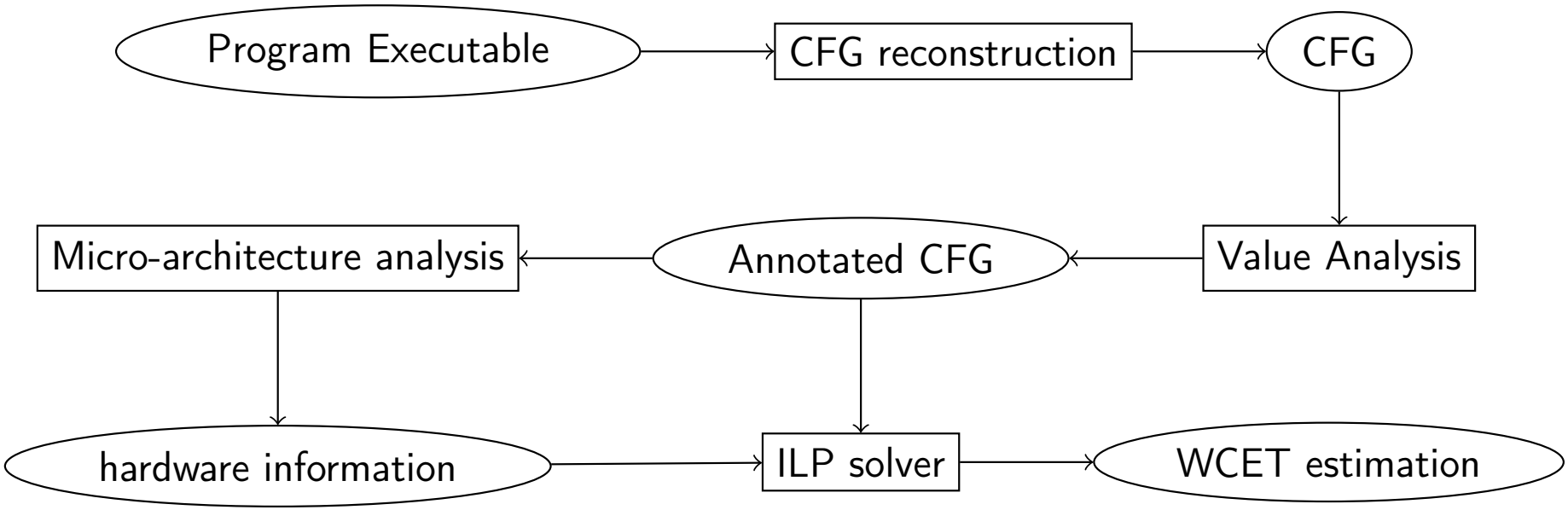
- Logically correct
- Satisfying timing requirements (bounded WCET)

Why cache matters ?

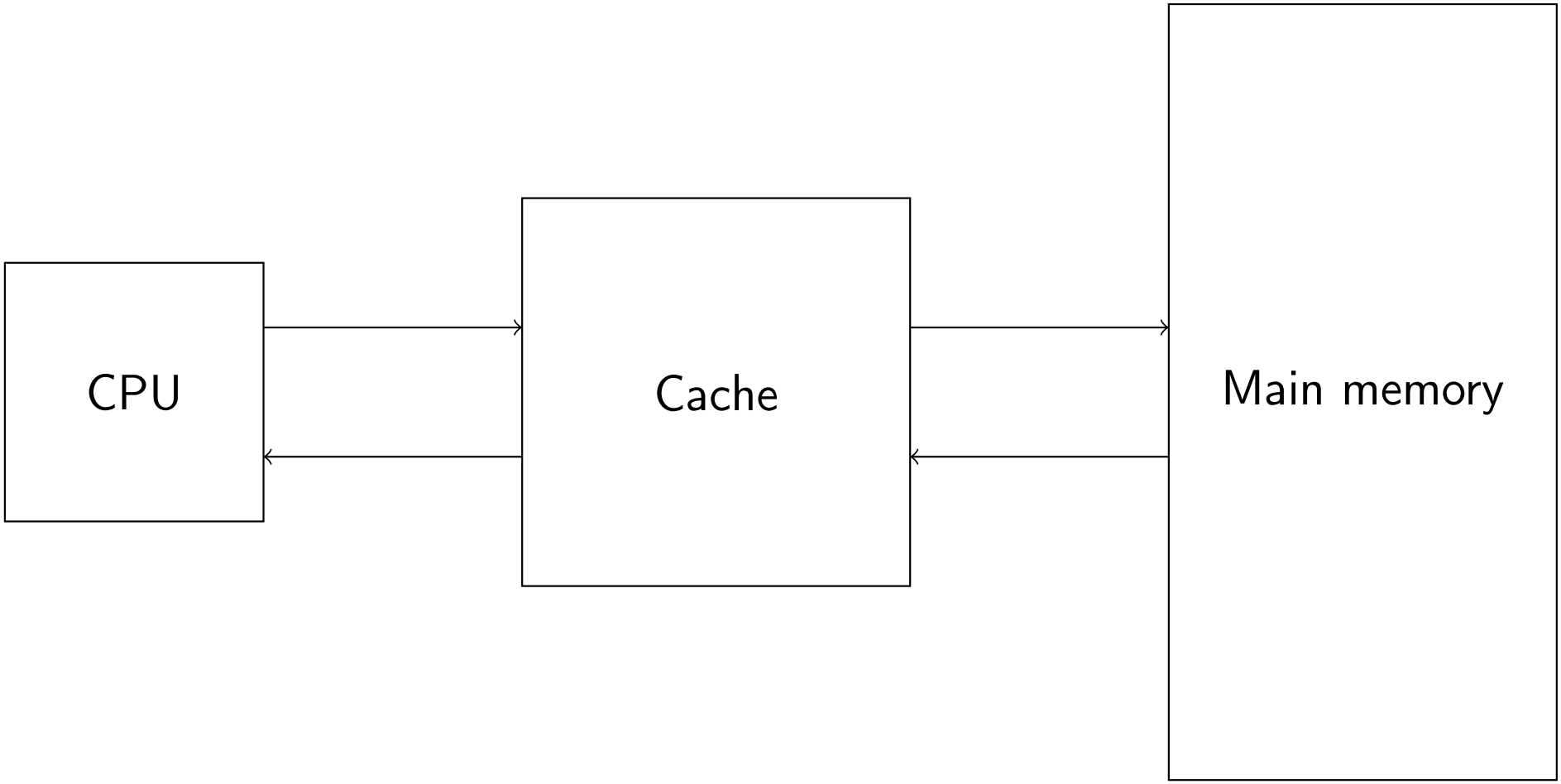
Caches have the biggest influence on WCET [HP11]

- Cache hit : $\sim$ 2 cycles,
- Cache miss : $\sim$ 100 cycles

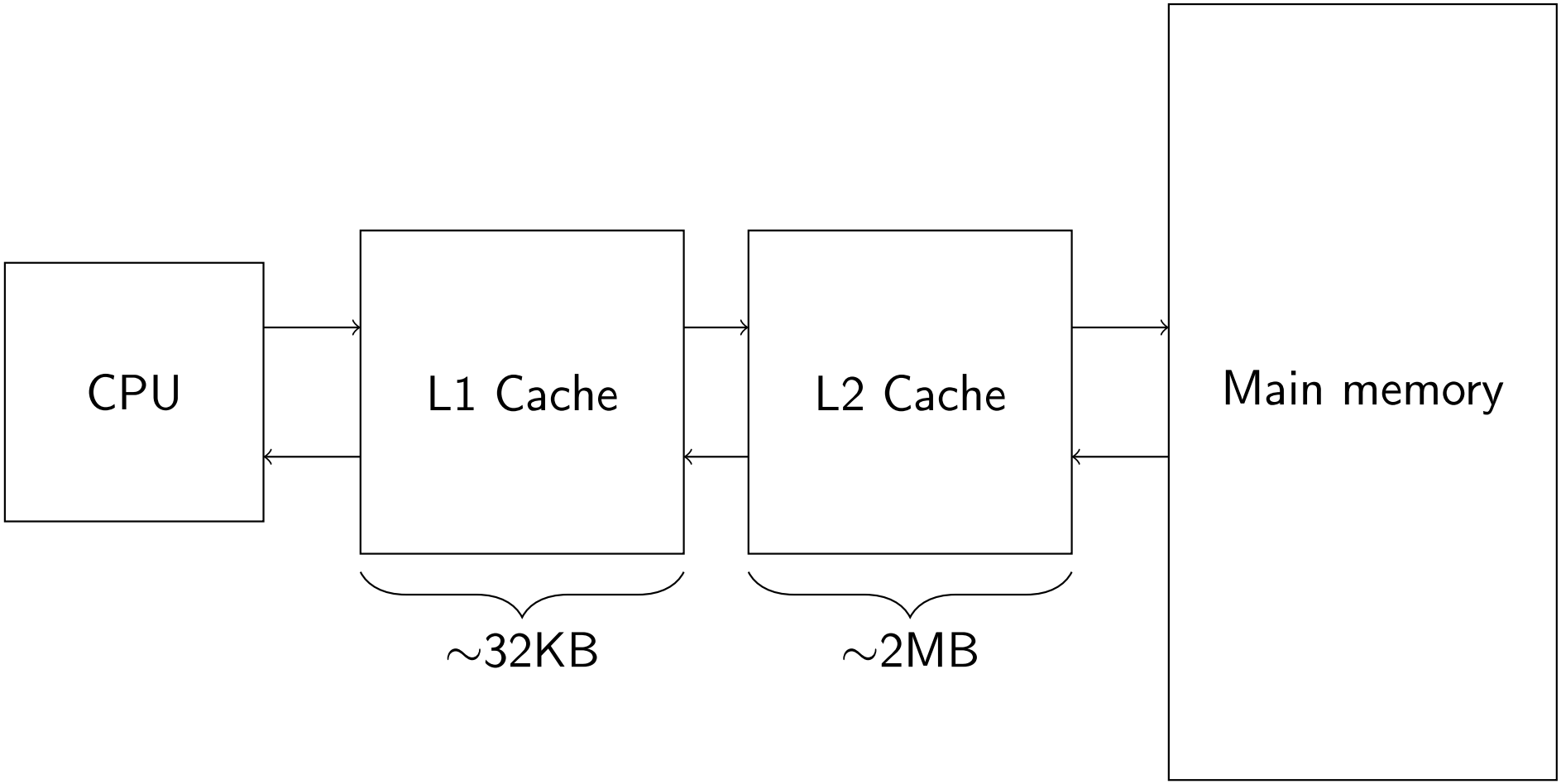
WCET Analysis Framework



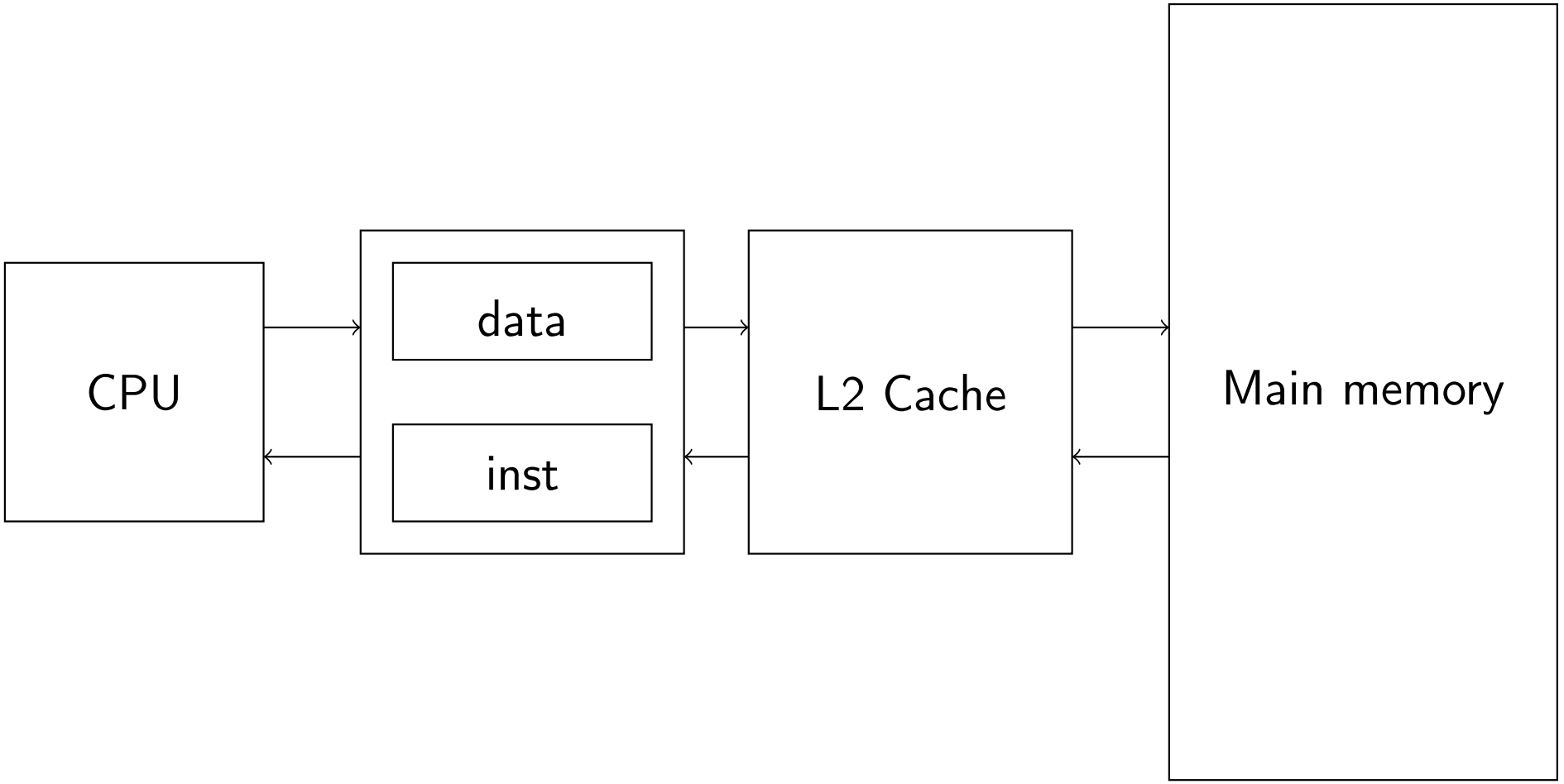
Caches



Caches



Caches



Caches

Blocks

Insts/Data are transferred by blocks of fixed size

Cache sets

Caches are split into independent sets. A block is assigned to a unique set.

Ways

Each block can contain k blocks (k is called associativity of the cache).

Replacement policy

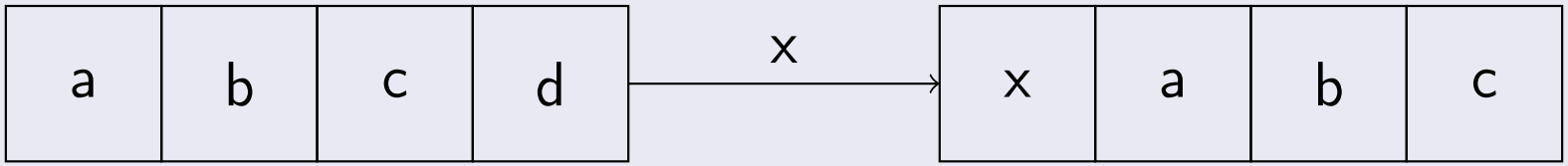
Selection of the block to evict on a miss.

- LRU is easier to analyze
- PLRU, MRU and FIFO are cheaper and easier to implement

Least Recently Used policy

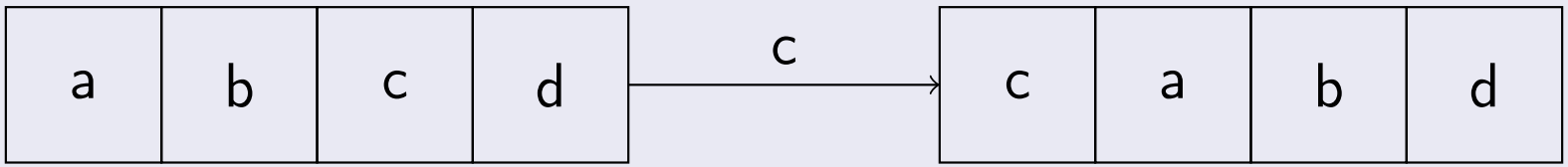
Miss

All blocks are shifted, and LRU is evicted.



Hit

Only younger blocks are shifted.



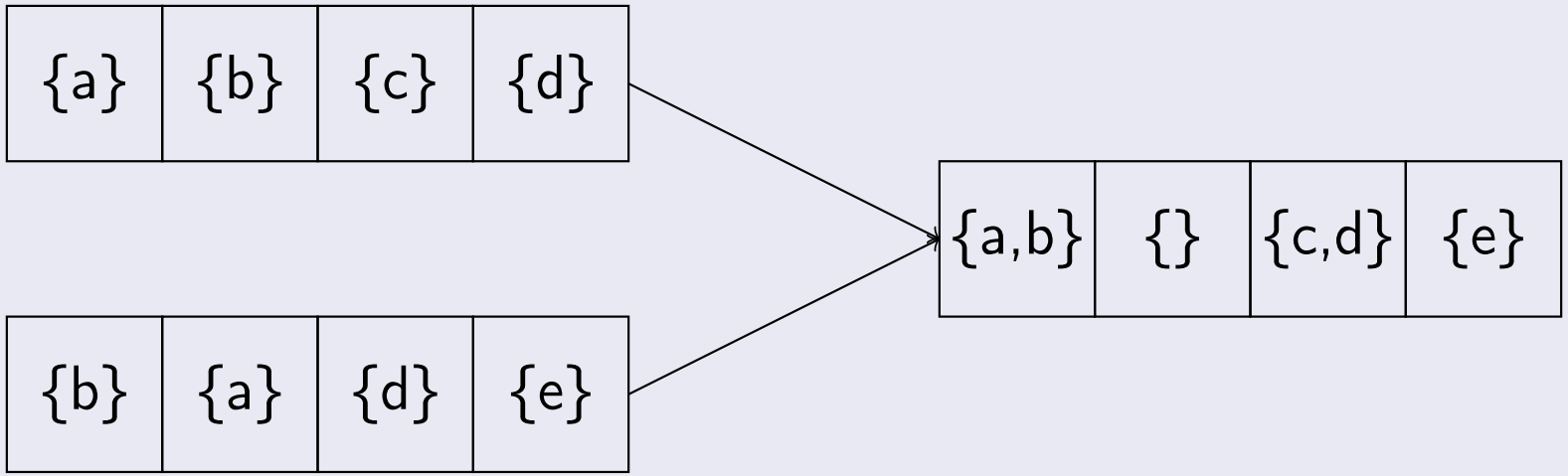
May/Must Analysis [Fer97]

Abstraction of cache states

May/Must overapproximates block ages :

- May : safe lower bound
- Must : safe upper bound

Example : May join



Precision lost on loops

Dealing with loops

Usually (for small enough loops), first iteration loads blocks in the cache (cold miss), and further iterations lead to hits.

Solutions

- Virtual unrolling [MAWF98] : Analyze first iteration separately.
- Persistence Analysis : Remember blocks that have been visited and evicted. All other blocks only suffer one miss at most. [FW99], [Cul13], [HJR11]

Which one is better ?

- Not comparable for precision
- Hard to compare in term of cost : expensive micro-architectural analysis VS more constraints during path enumeration.

Data caches

Issues

- Some addresses are not known statically (heap allocated data).
- Memory access pattern are usually unknown.
- Data can be written.

Analysis must deal conservatively with many potentially accessed block.

Approaches

- Discover access pattern (cache miss equation [GMM99]) : successful on a small subset of programs, and costly.
- Adapt Persistence Analysis to data to bound the number of misses [SS07] : fast, but very pessimistic estimations
- ...

Write policies

Write-through/Write-back

- Write-through : Data is updated both in the cache and the backing store.
- Write-back : Data is modified in the cache and marked as dirty. On eviction, a dirty block is written to the backing store.

Analysis of write-back caches need to track dirty bits.

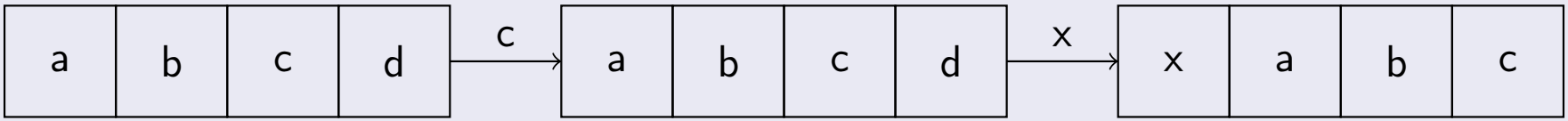
Write-allocate/No-write-allocate

- Write-allocate : On a write-miss, data is loaded into the cache before being updated.
- No-write-allocate : Data is directly written to the backing store.

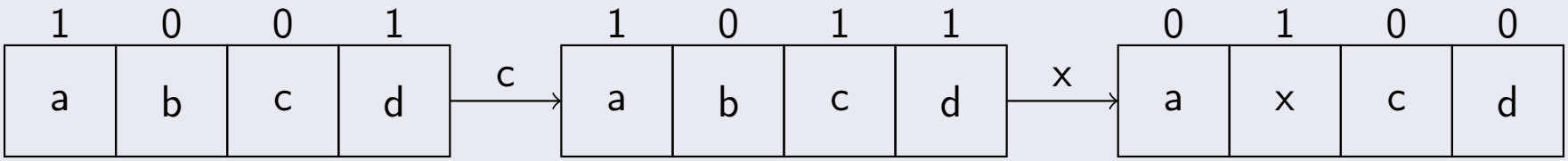
Usually, write-back caches are write-allocate and write-through are no-write-allocate.

Non LRU policies

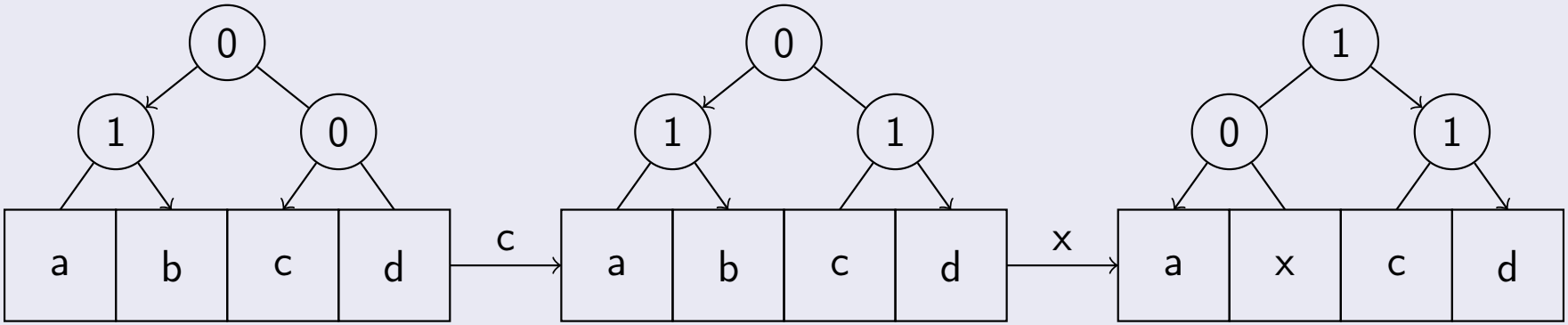
FIFO



MRU



PLRU



Non LRU policies

Classification

(Always-hit, Always-miss, Not-classified) classification is not adapted

- FIFO : accesses easily exhibit alternate hit/miss behavior
- MRU : many accesses are K-misses.

Irregular cache update

Under LRU, an accessed block stays in the cache for at least k-1 accesses.

- FIFO/MRU : an accessed block can be evicted in only 1 or 2 accesses.
- PLRU : A block can stay forever, even if not accessed.

Initial state sensitivity

- Uncertainty about initial cache content is harder to remove.
- Empty initial cache state is not always the worst case.

Multi-level caches

Issues

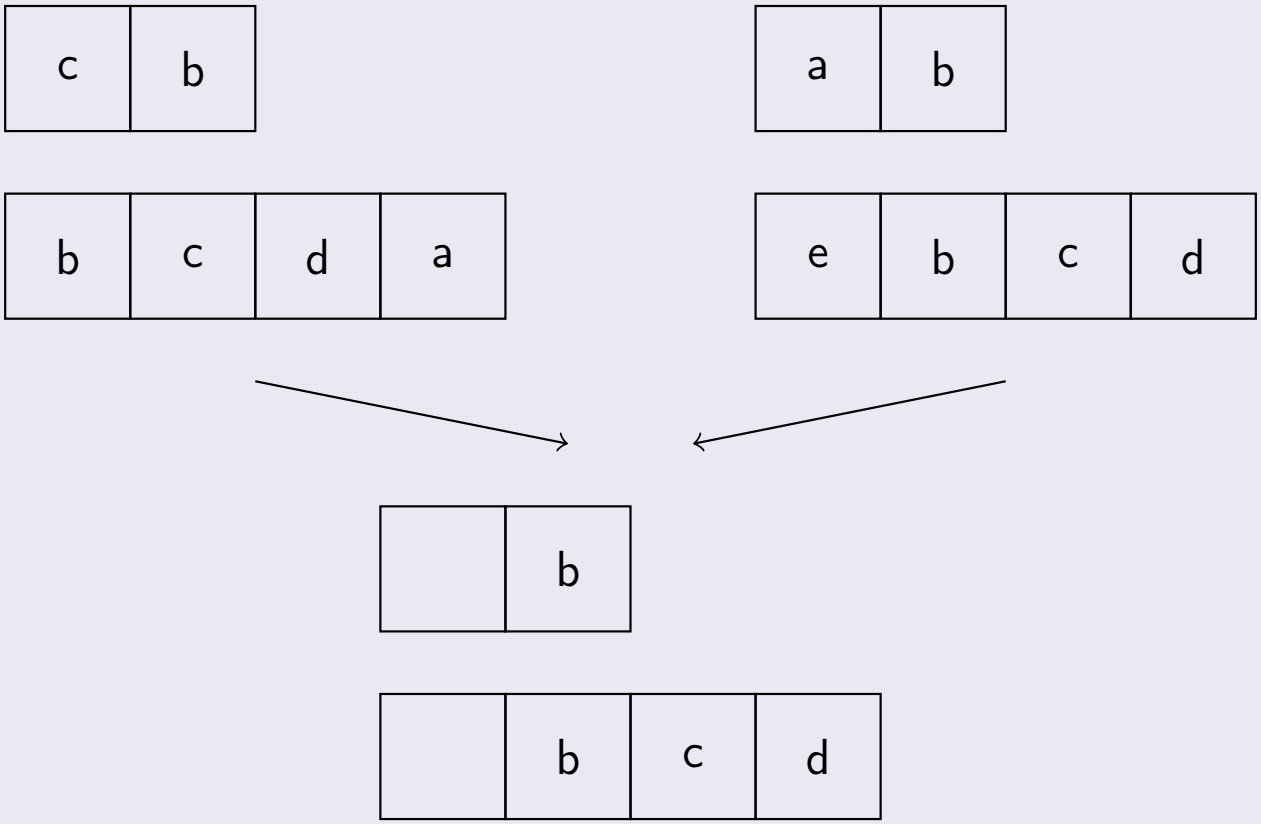
- Except for the first level, instructions and data are not distinguished.
- Uncertainty propagates between levels.

Approaches

- Separate approach
- Integrated approach

Why one need integrated approach ?

Example : separate approach



What can one tell about block a ?

Separate approach conclude there is no guarantee *a* is in the cache after joining.

Inclusiveness

Exclusive caches

A block is stored in one cache level at most. A hit at L_{n+1} exchange lines between levels L_n and L_{n+1}.

Strictly inclusive caches

Level L_n is included in level L_{n+1}. Thus, an eviction at level L_{n+1} can lead to an eviction at level L_n.

Cache related preemption delay

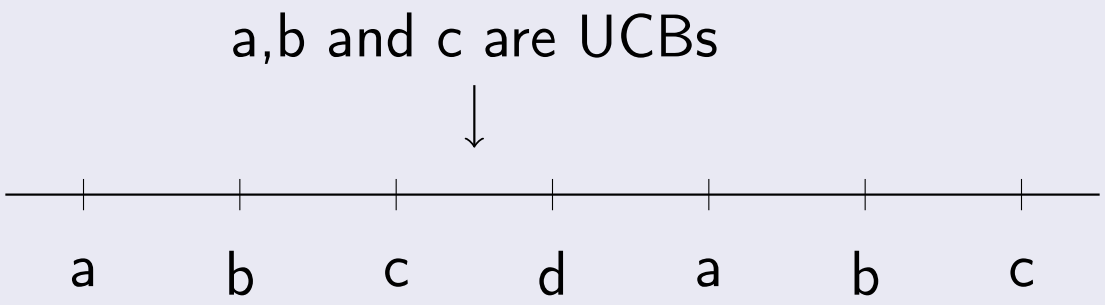
Issue

- Preemption causes block eviction. Thus, when resuming a preempted task, it suffers from additional misses.
- Preemption might be nested

Analysis the preempted

Useful cache block : block that is cached and may be reused before it is evicted. Bounding the number of UCBs bounds the number of additional misses due to preemption.

Example for LRU

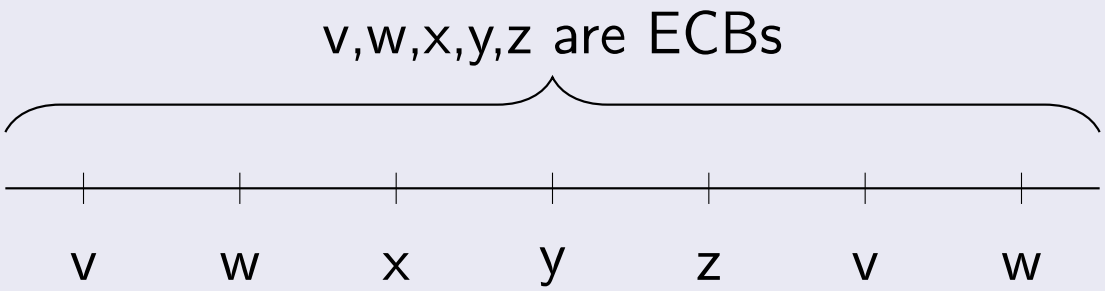


Cache related preemption delay II

Analysis the preempting task

Evicting cache block : block that is accessed in the preempting task.

Example for LRU



Combining both approach : resilience analysis [AMR10]

Resilience of a UCB m is the difference between k and the age of m before its next access, ie. the number of ECBs it can suffers before it becomes useless.

Shared caches in multi-cores

Issues

- Same as preemption issues
- No priority fixed
- Interleaving is unknown

Approach is similar to preemption

1. Analyze task A in isolation
2. Bound the interference that can evict "useful" blocks.

Not considering timing of the conflicting accesses $\Rightarrow$ precision lost

Other approaches

- Exclude unfeasible conflict by adding constraint in the ILP [ZY09].
- Explore task overlapping [LDM+12].
- Gain precision using model checking instead of AI (scalability ?).

Conclusion

Future work

- Design precise analysis for non-LRU policies.
- Application to other domain than WCET, e.g.side-channels
- Design timing-predictable embedded systems

Questions ?

References I


Sebastian Altmeyer, Claire Maiza, and Jan Reineke, *Resilience analysis: tightening the CRPD bound for set-associative caches*, Proceedings of the ACM SIGPLAN/SIGBED 2010 conference on Languages, compilers, and tools for embedded systems, LCTES 2010, Stockholm, Sweden, April 13-15, 2010 (Jaejin Lee and Bruce R. Childers, eds.), ACM, 2010, pp. 153–162.


Christoph Cullmann, *Cache persistence analysis for embedded real-time systems*, Ph.D. thesis, Saarland University, 2013.


Christian Ferdinand, *Cache behavior prediction for real-time systems*, Pirrot, 1997.


Christian Ferdinand and Reinhard Wilhelm, *Efficient and precise cache behavior prediction for real-time systems*, Real-Time Systems **17** (1999), no. 2-3, 131–181.


Somnath Ghosh, Margaret Martonosi, and Sharad Malik, *Cache miss equations: a compiler framework for analyzing and tuning memory behavior*, ACM Trans. Program. Lang. Syst. **21** (1999), no. 4, 703–746.


Bach Khoa Huynh, Lei Ju, and Abhik Roychoudhury, *Scope-aware data cache analysis for WCET estimation*, 17th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2011, Chicago, Illinois, USA, 11-14 April 2011, IEEE Computer Society, 2011, pp. 203–212.


John L. Hennessy and David A. Patterson, *Computer architecture, fifth edition: A quantitative approach*, 5th ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.


Yun Liang, Huping Ding, Tulika Mitra, Abhik Roychoudhury, Yan Li, and Vivy Suhendra, *Timing analysis of concurrent programs running on shared cache multi-cores*, Real-Time Systems **48** (2012), no. 6, 638–680.

References II

Florian Martin, Martin Alt, Reinhard Wilhelm, and Christian Ferdinand, *Analysis of loops*, Compiler Construction, 7th International Conference, CC'98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings (Kai Koskimies, ed.), Lecture Notes in Computer Science, vol. 1383, Springer, 1998, pp. 80–94.

Rathijit Sen and Y. N. Srikant, *WCET estimation for executables in the presence of data caches*, Proceedings of the 7th ACM & IEEE International conference on Embedded software, EMSOFT 2007, September 30 - October 3, 2007, Salzburg, Austria (Christoph M. Kirsch and Reinhard Wilhelm, eds.), ACM, 2007, pp. 203–212.

Wei Zhang and Jun Yan, *Accurately estimating worst-case execution time for multi-core processors with shared direct-mapped instruction caches*, 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2009, Beijing, China, 24-26 August 2009, IEEE Computer Society, 2009, pp. 455–463.