Load Value Approximation

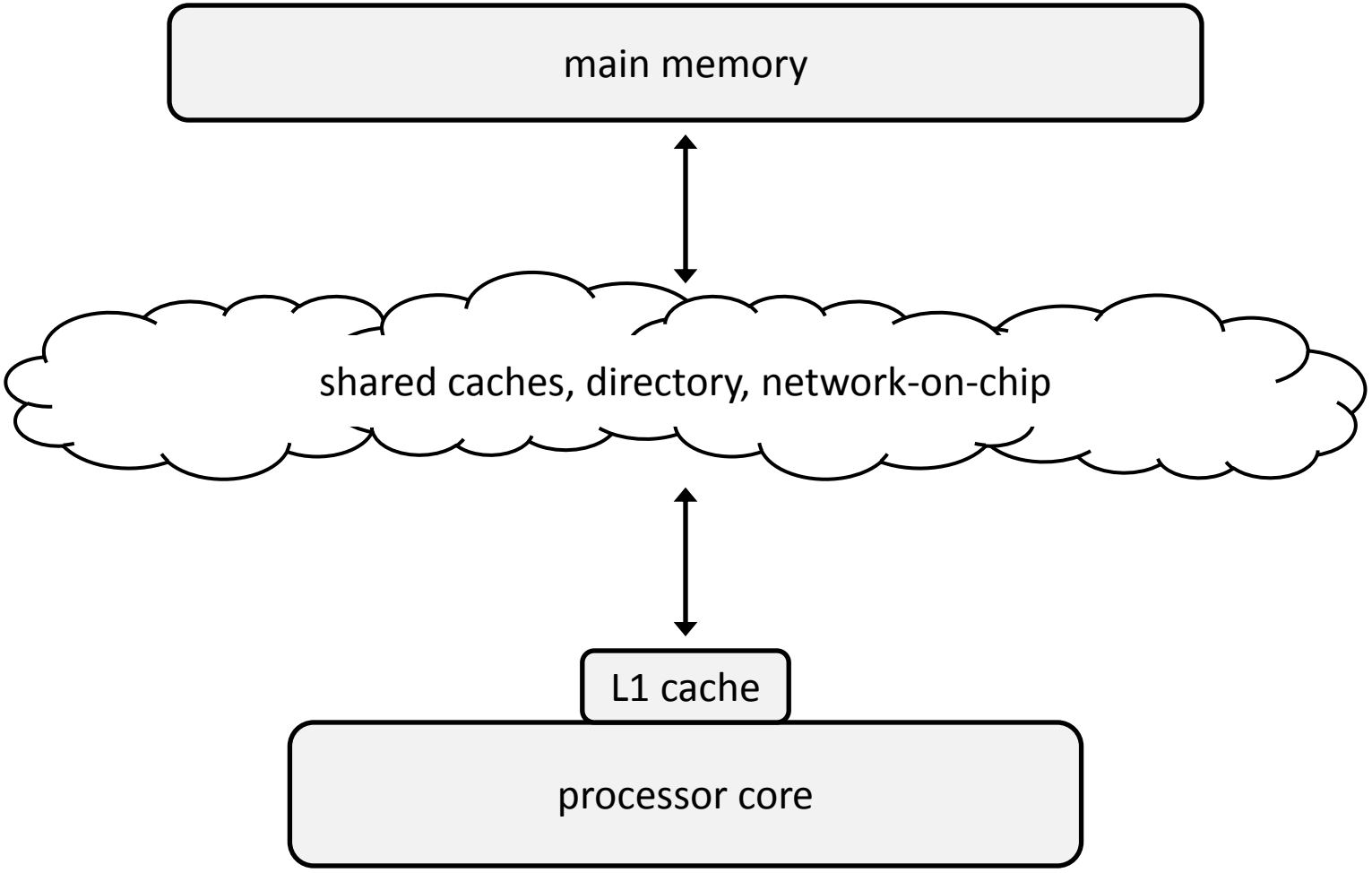
Joshua San Miguel

Mario Badr

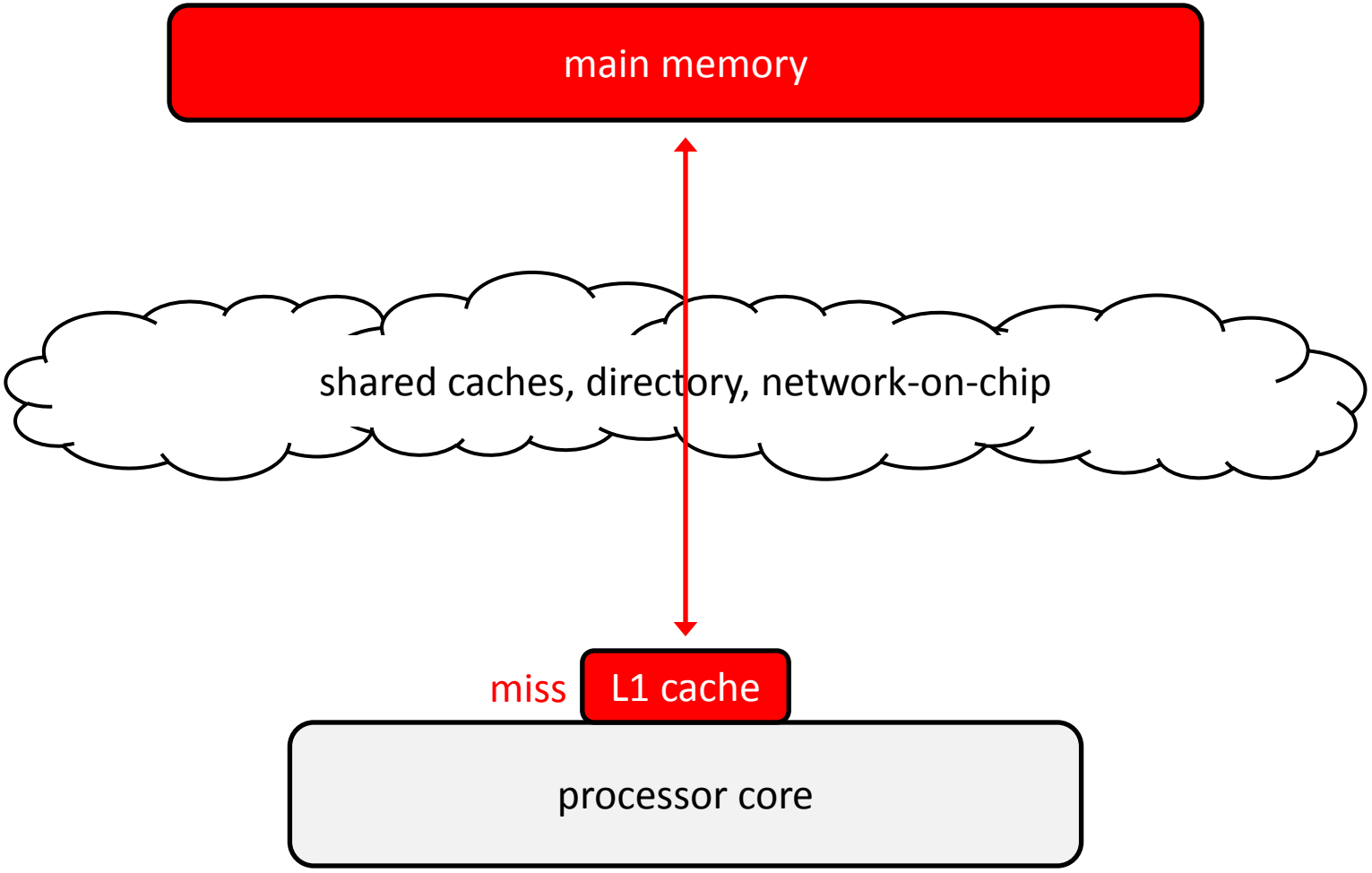
Natalie Enright Jerger

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

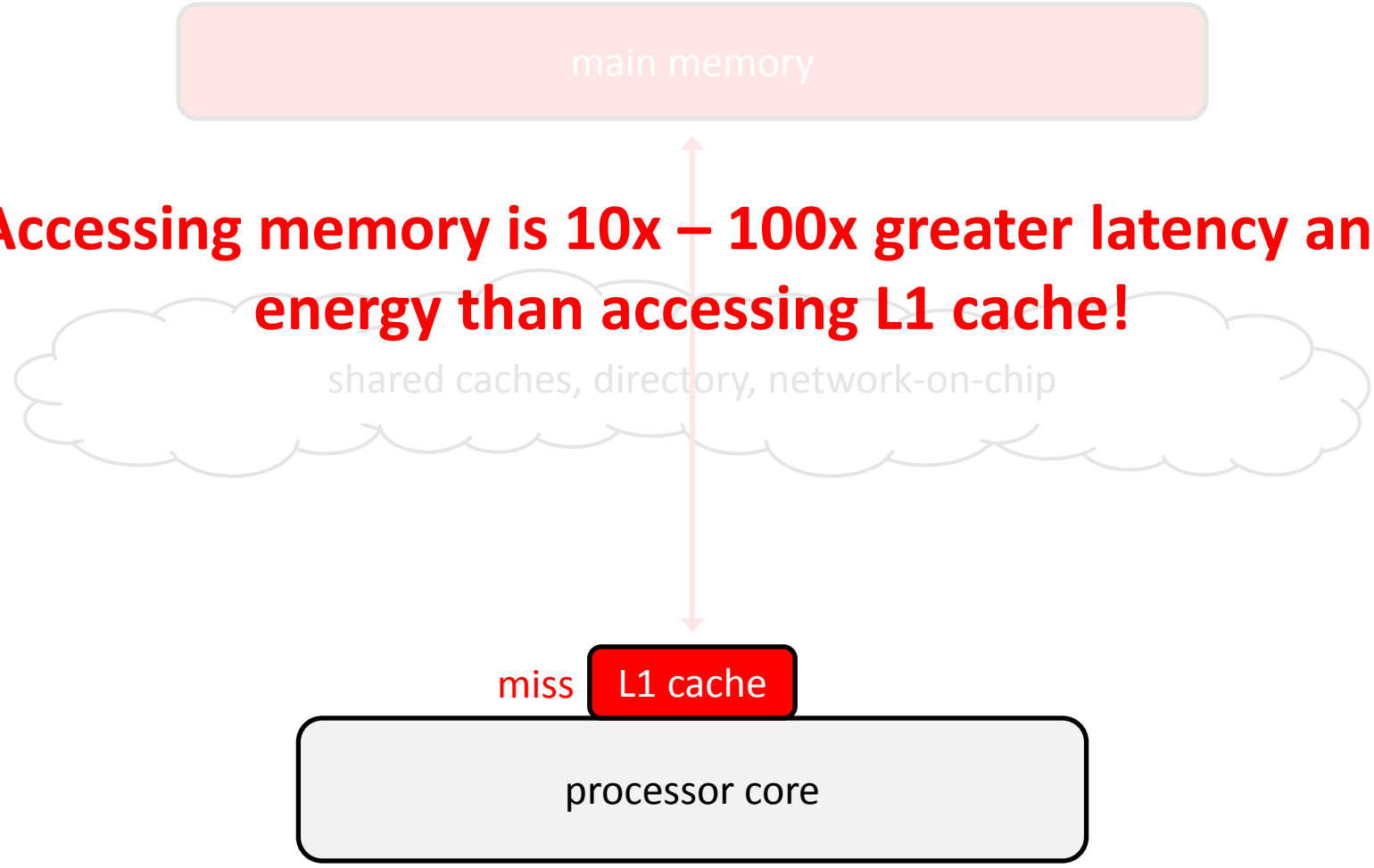
Accessing Memory



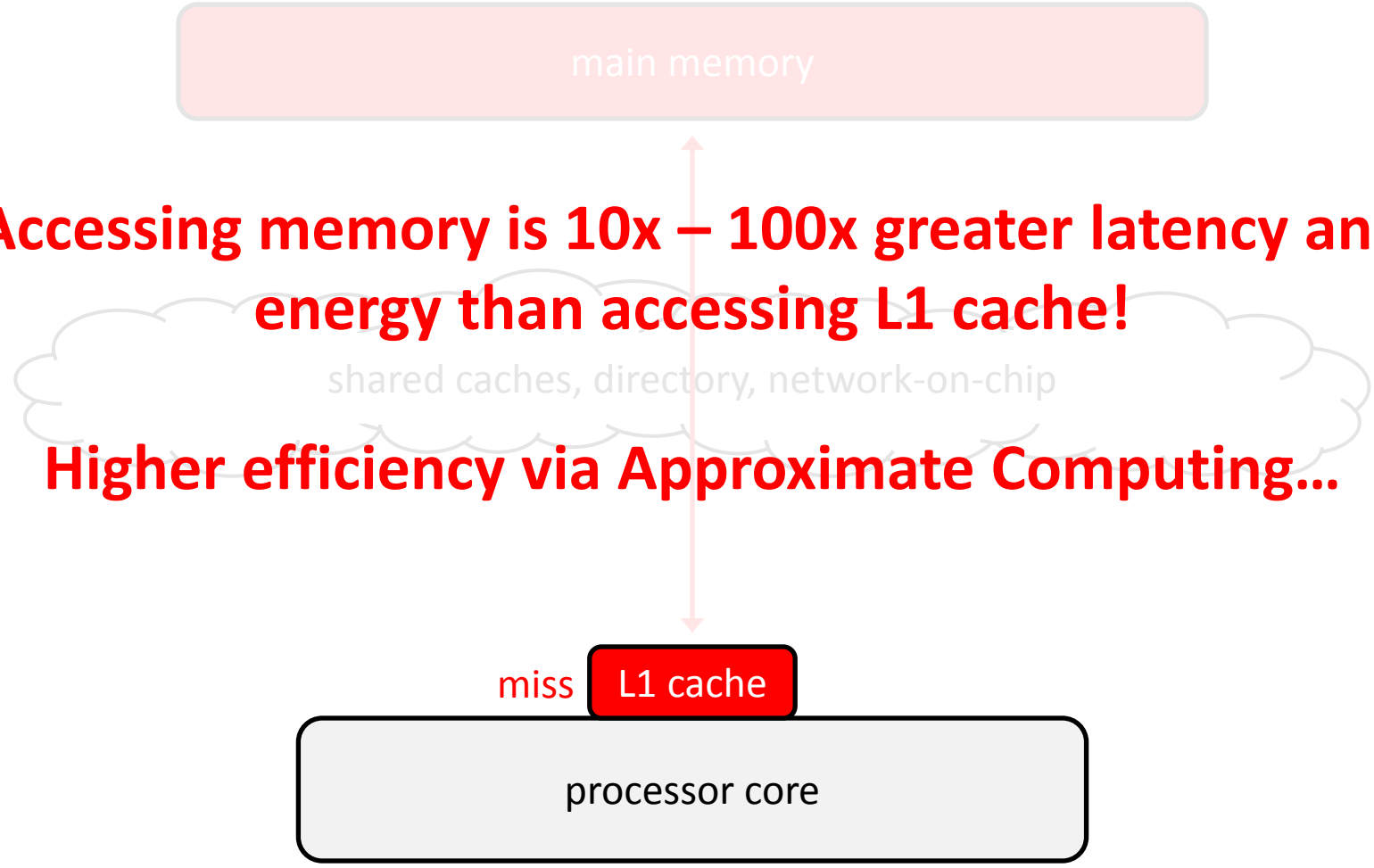
Accessing Memory



Accessing Memory

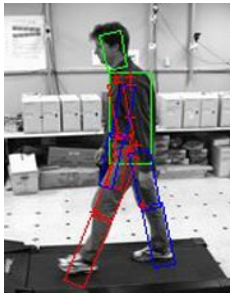
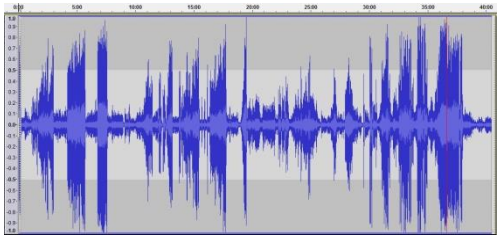
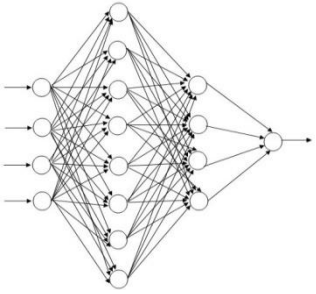
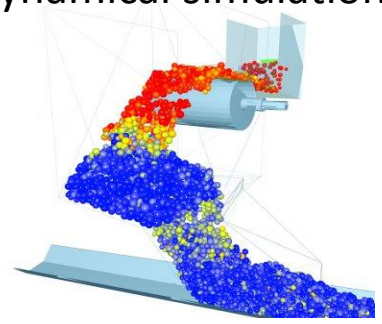


Accessing Memory

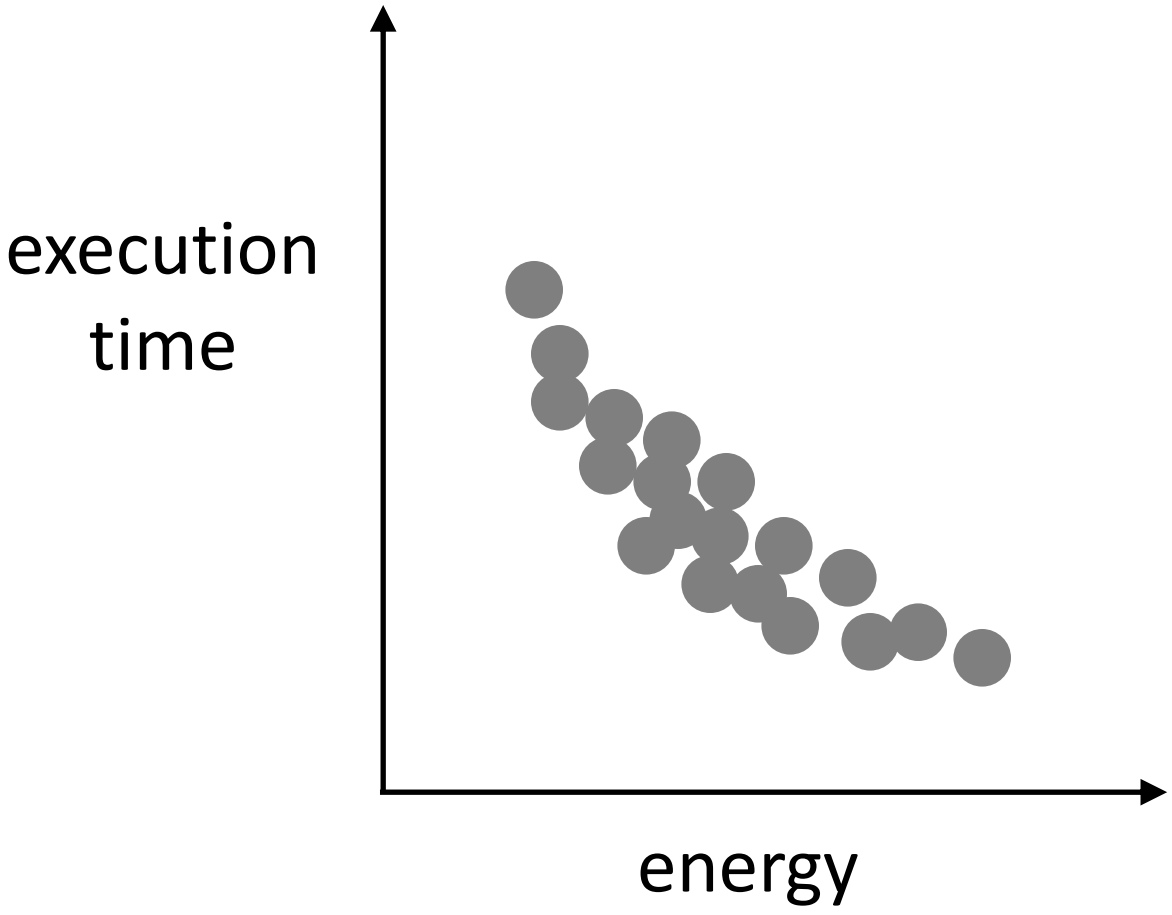


Approximate Computing

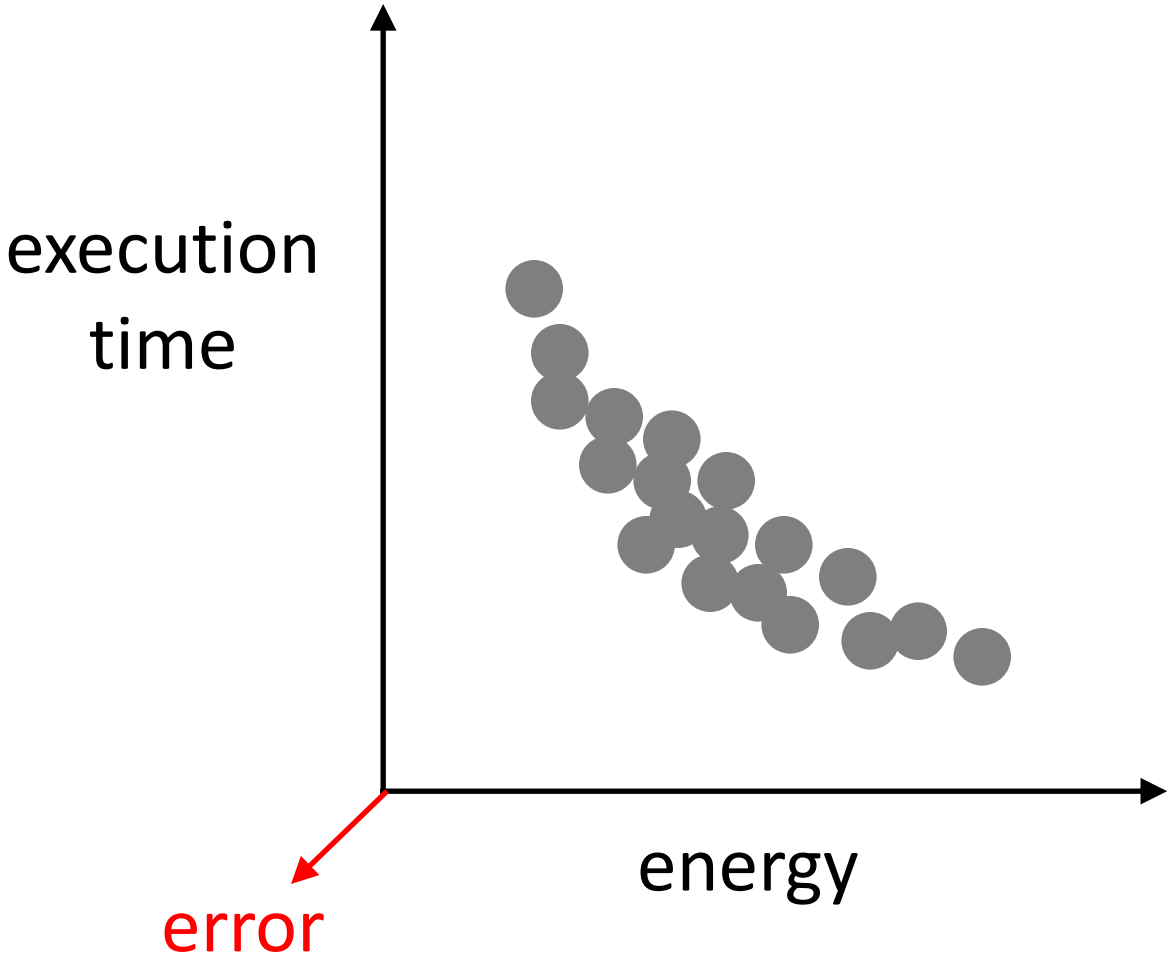
Not all computations need to be precise.

Data mining  http://www.zentut.com/	Computer vision  http://www.cc.gatech.edu/~cnieto6/	Audio and video processing  http://themusicparlour.blogspot.ca/
Gaming  http://www.businessweek.com/	Machine learning  http://www.analyticbridge.com/	Dynamical simulation  http://www.scientific-computing.com/

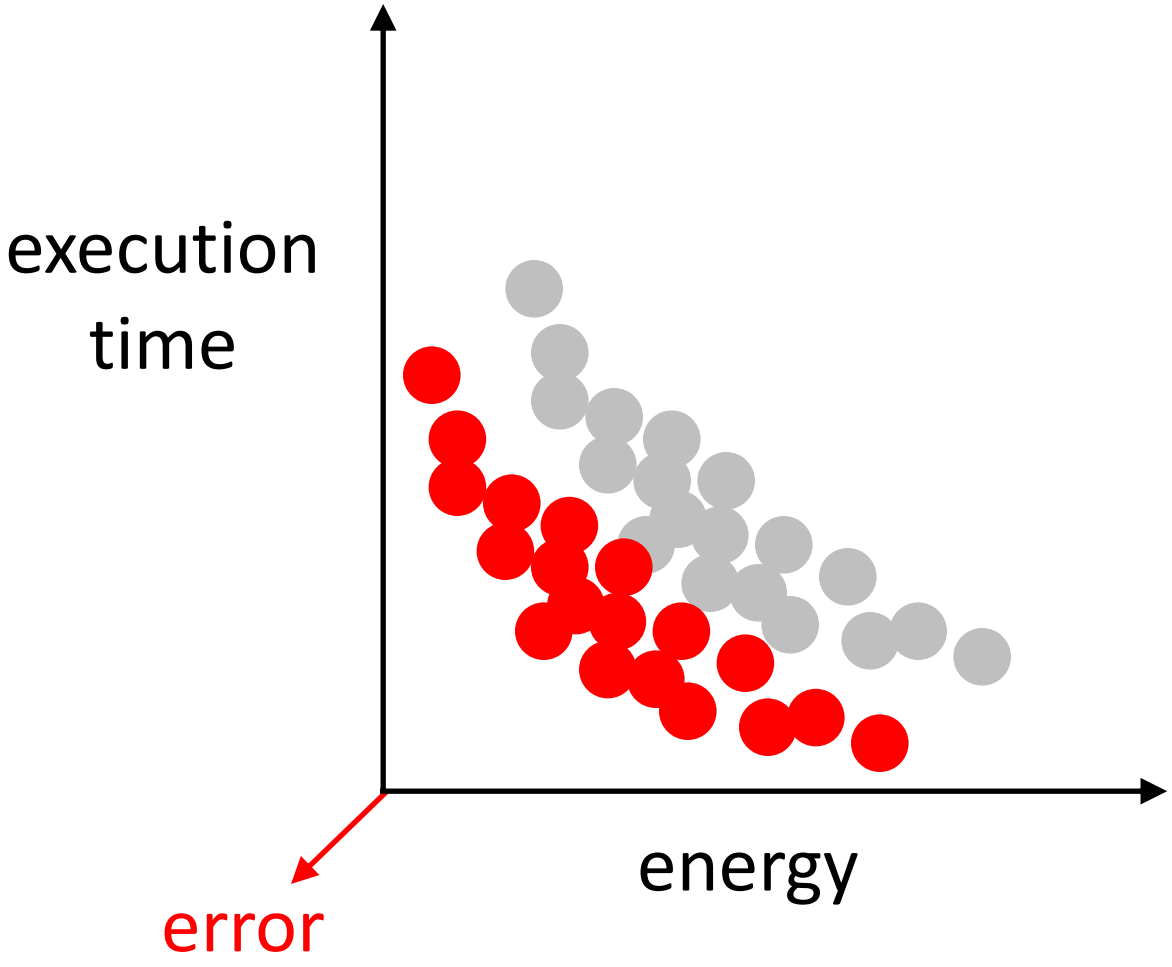
Approximate Computing



Approximate Computing



Approximate Computing



Approximate Computing

Many applications can tolerate approximate data.

- 40% to nearly 100% of data footprint is approximate [Sampson, MICRO 2013].

Approximate Computing

Many applications can tolerate approximate data.

- 40% to nearly 100% of data footprint is approximate [Sampson, MICRO 2013].

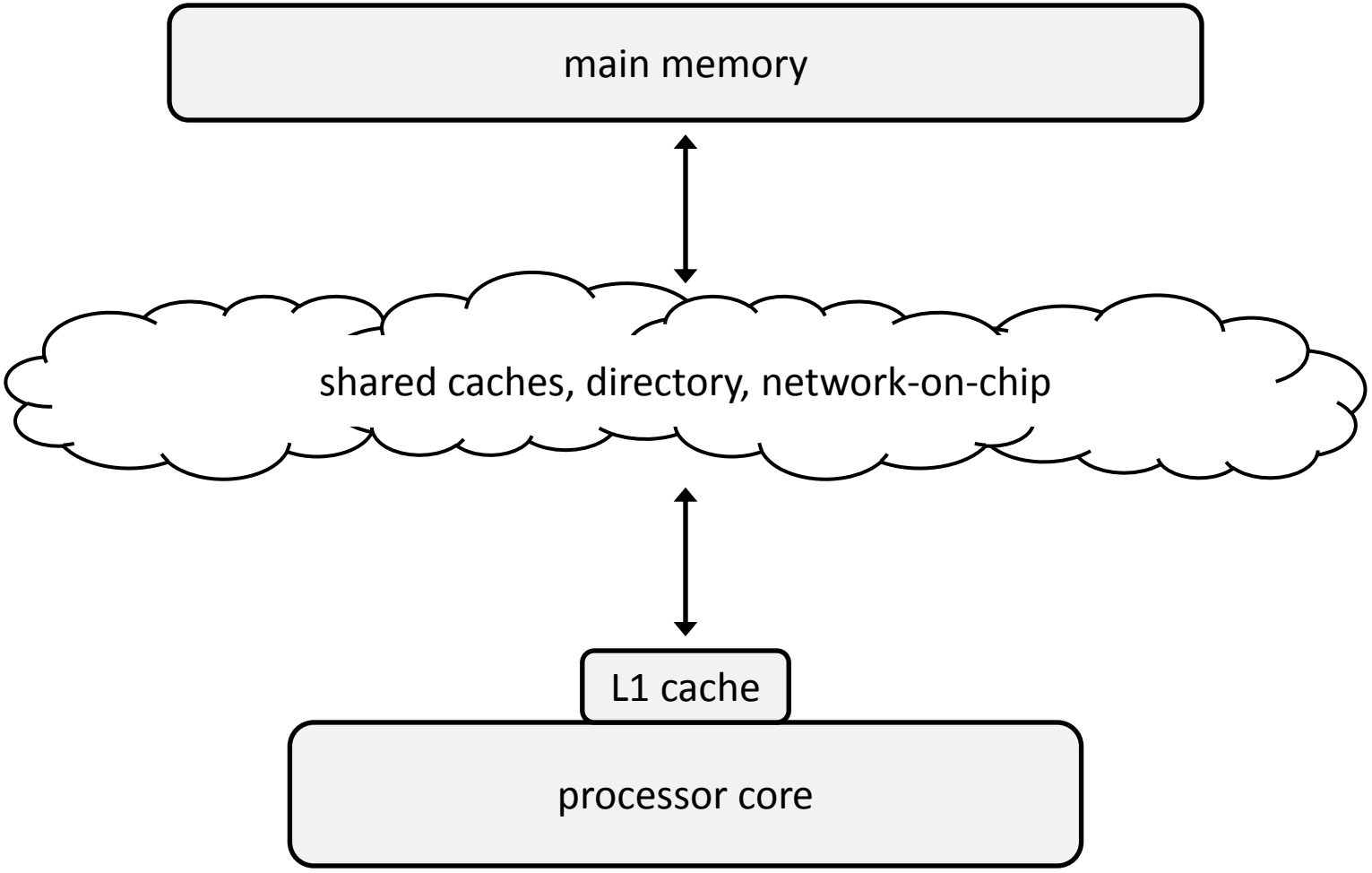
Approximate value locality:

- **Many data values are similar to or can be approximated from previously seen values.**

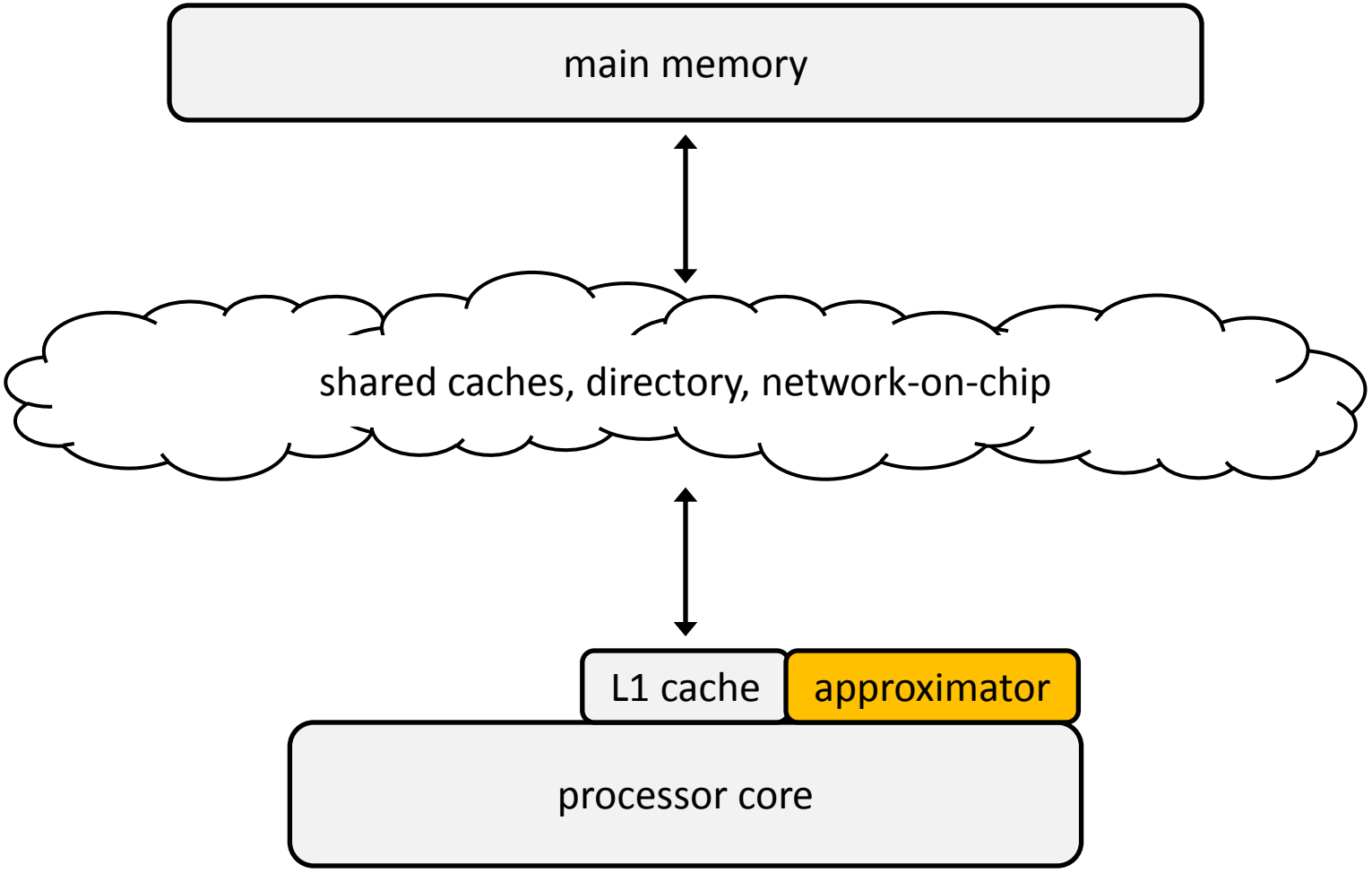
Outline

- Load Value Approximation
  - Non-Speculative Operation
- Approximator Design
  - Relaxed Confidence Windows
  - Approximation Degree
- Methodology
- Evaluation

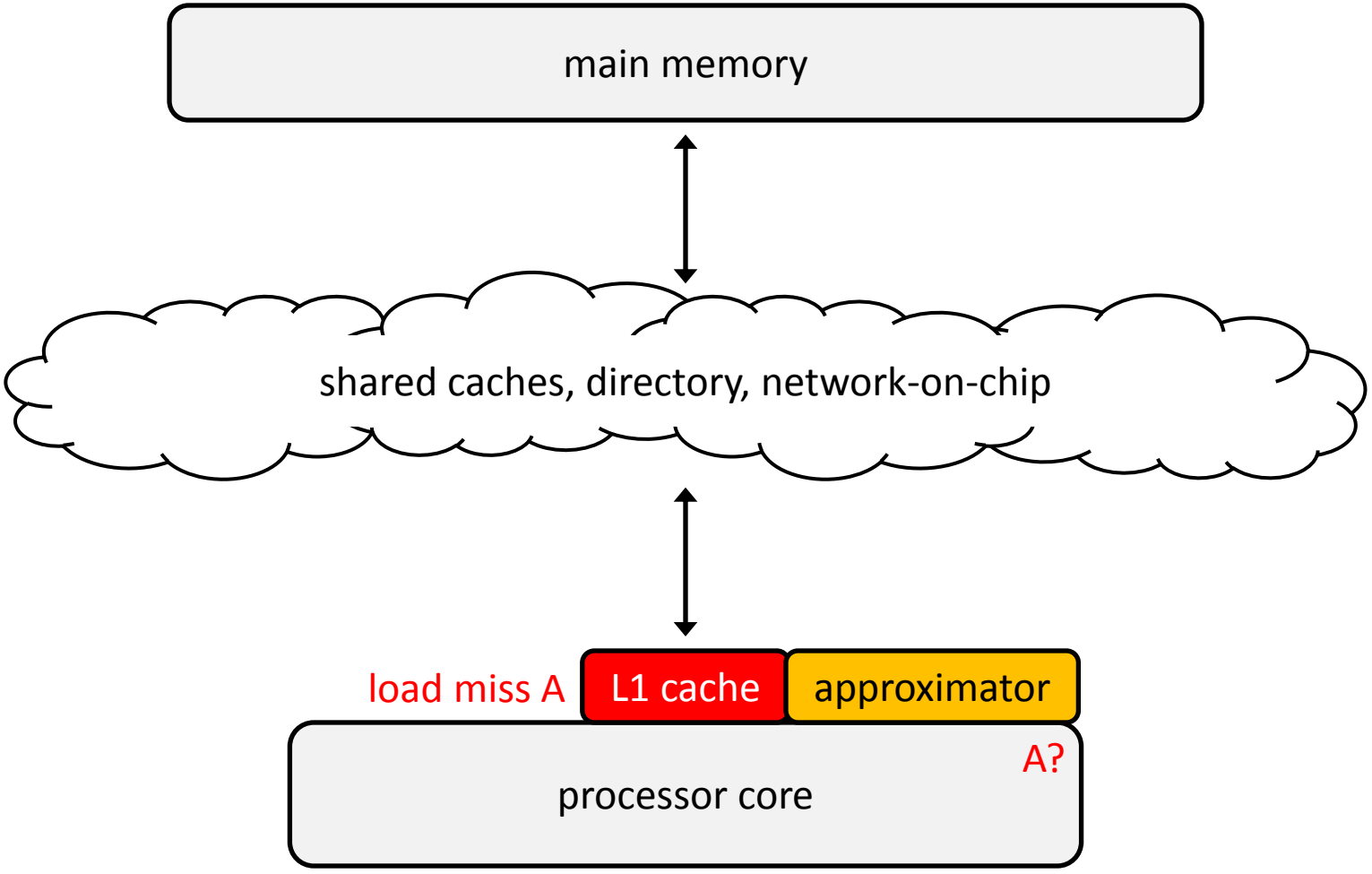
Load Value Approximation



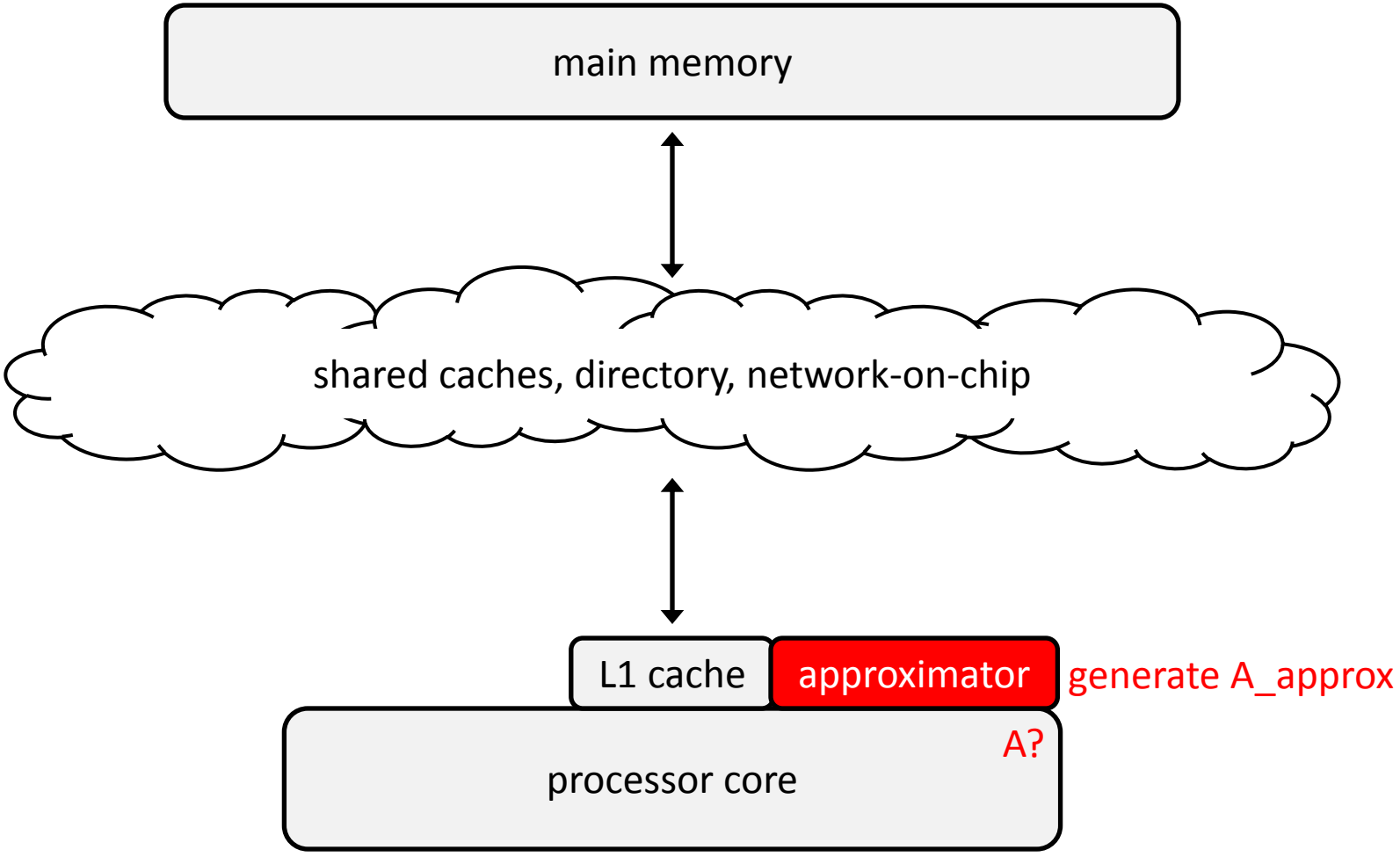
Load Value Approximation



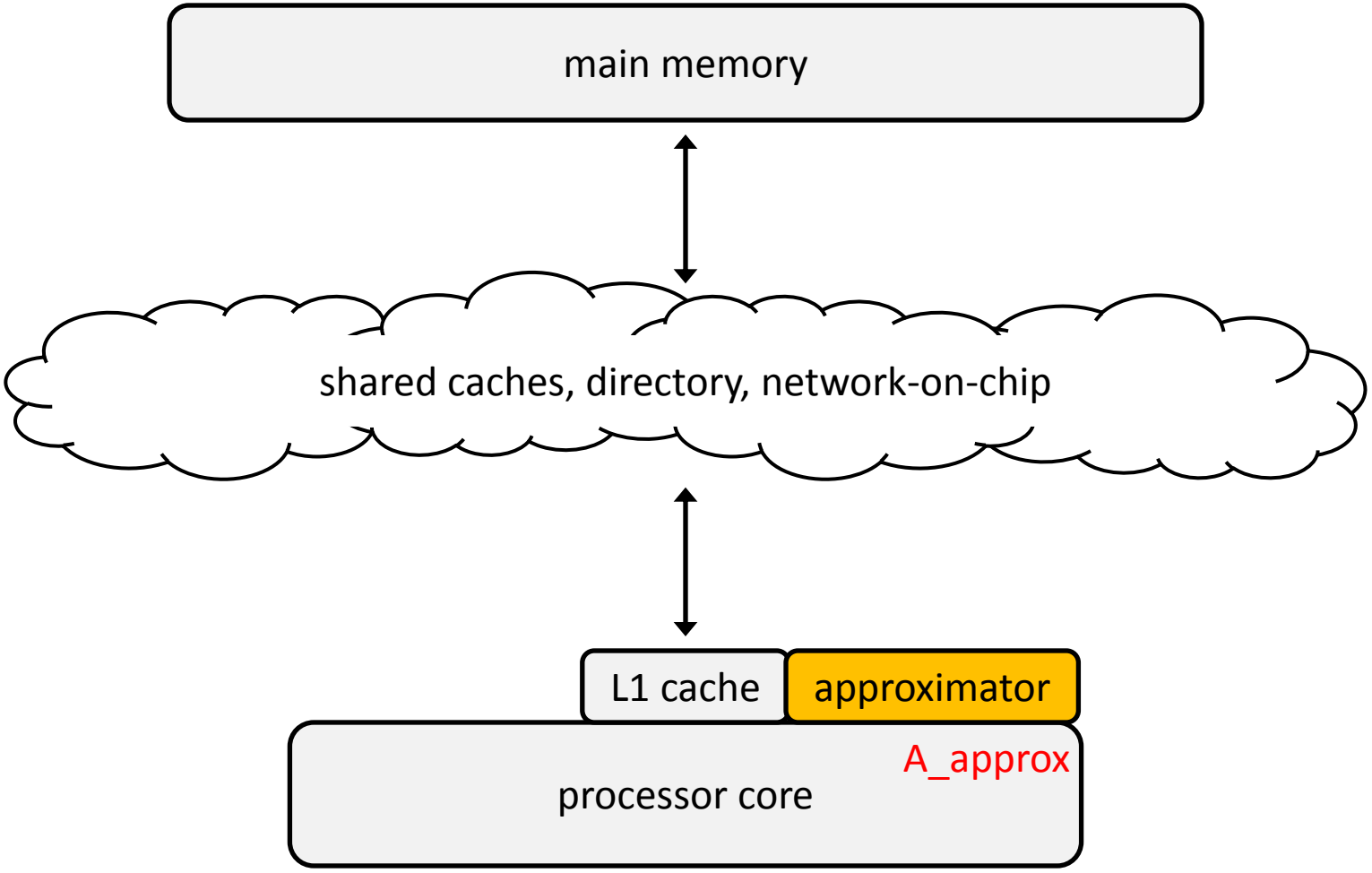
Load Value Approximation



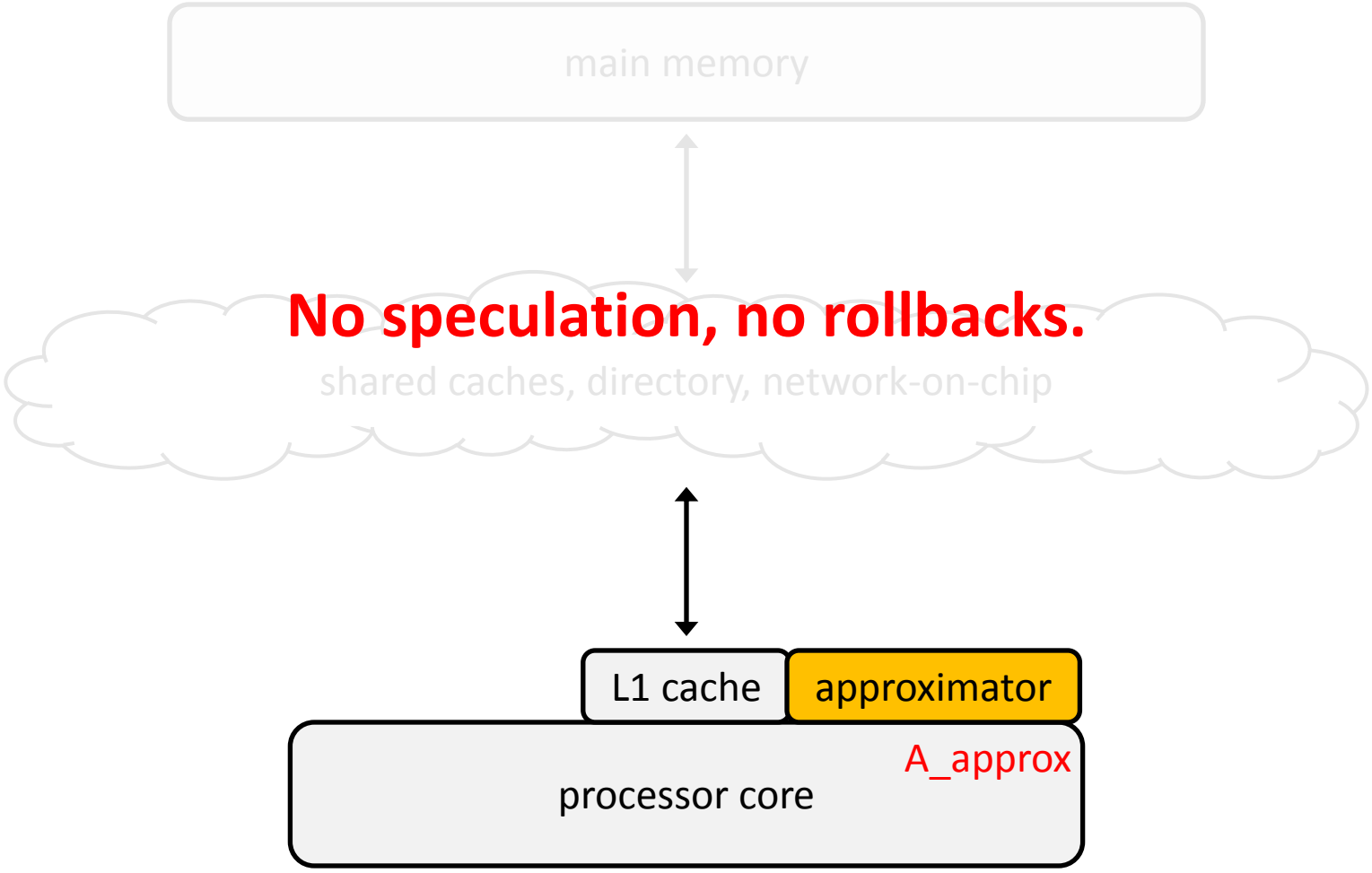
Load Value Approximation



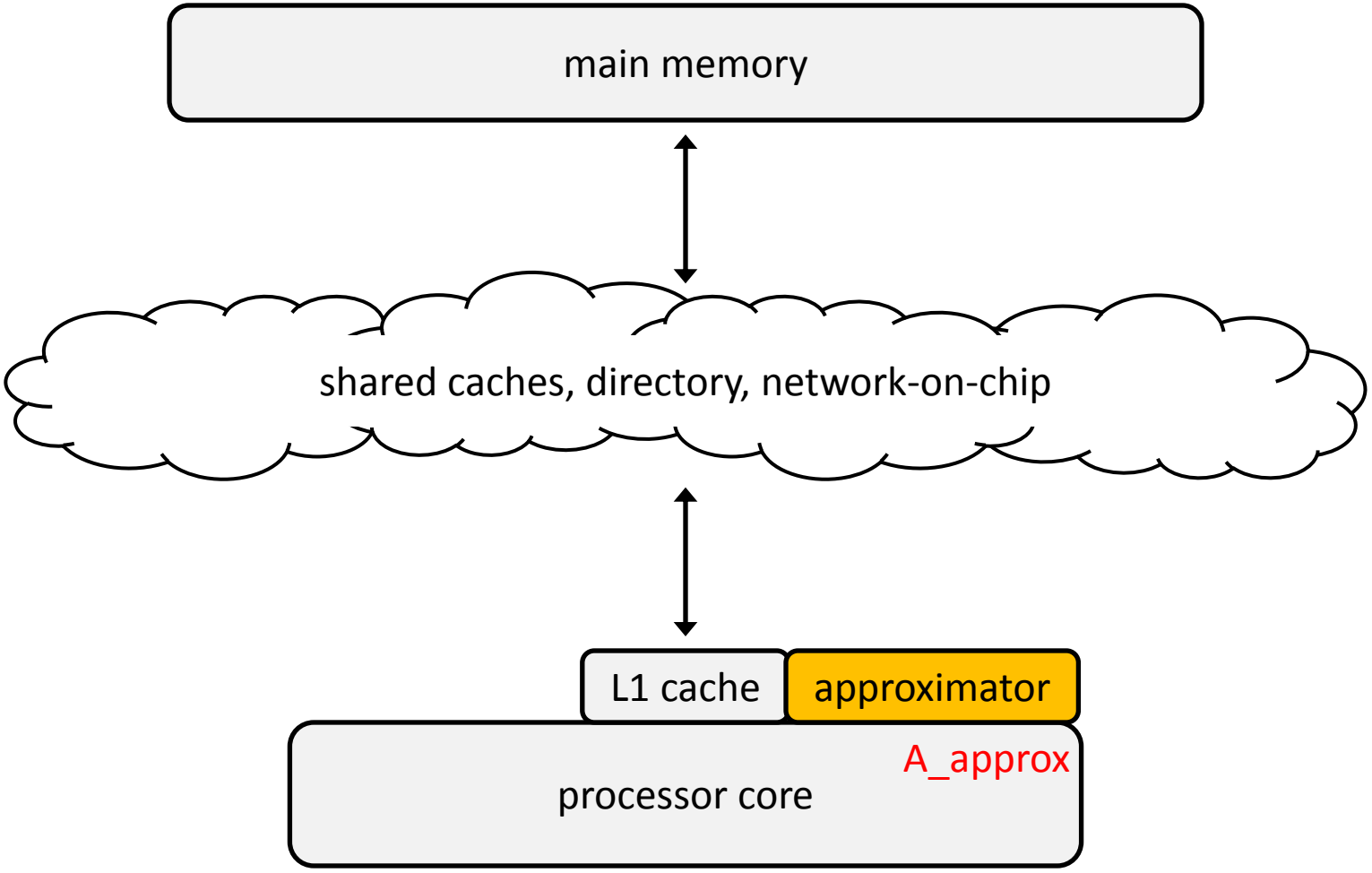
Load Value Approximation



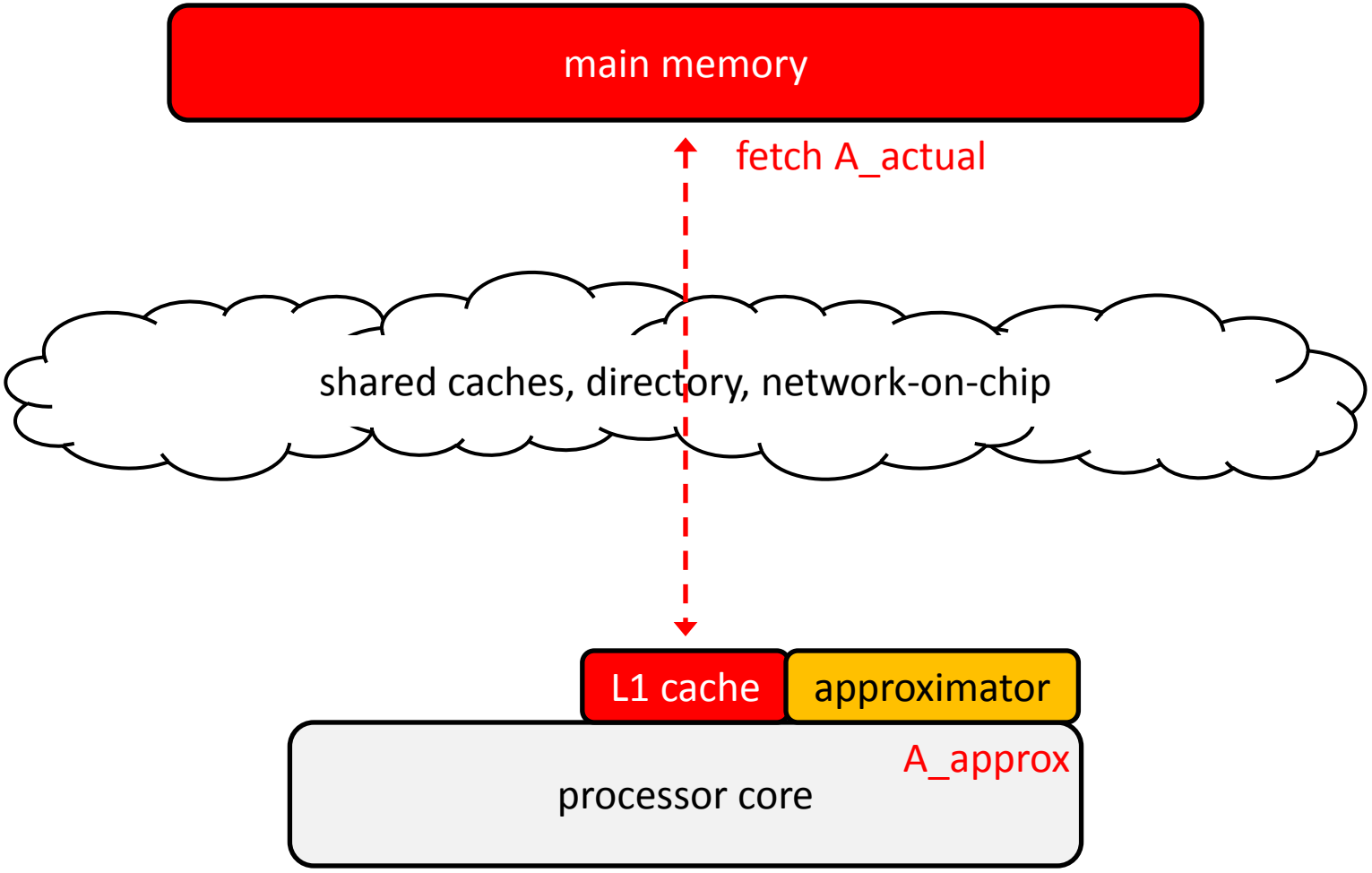
Load Value Approximation



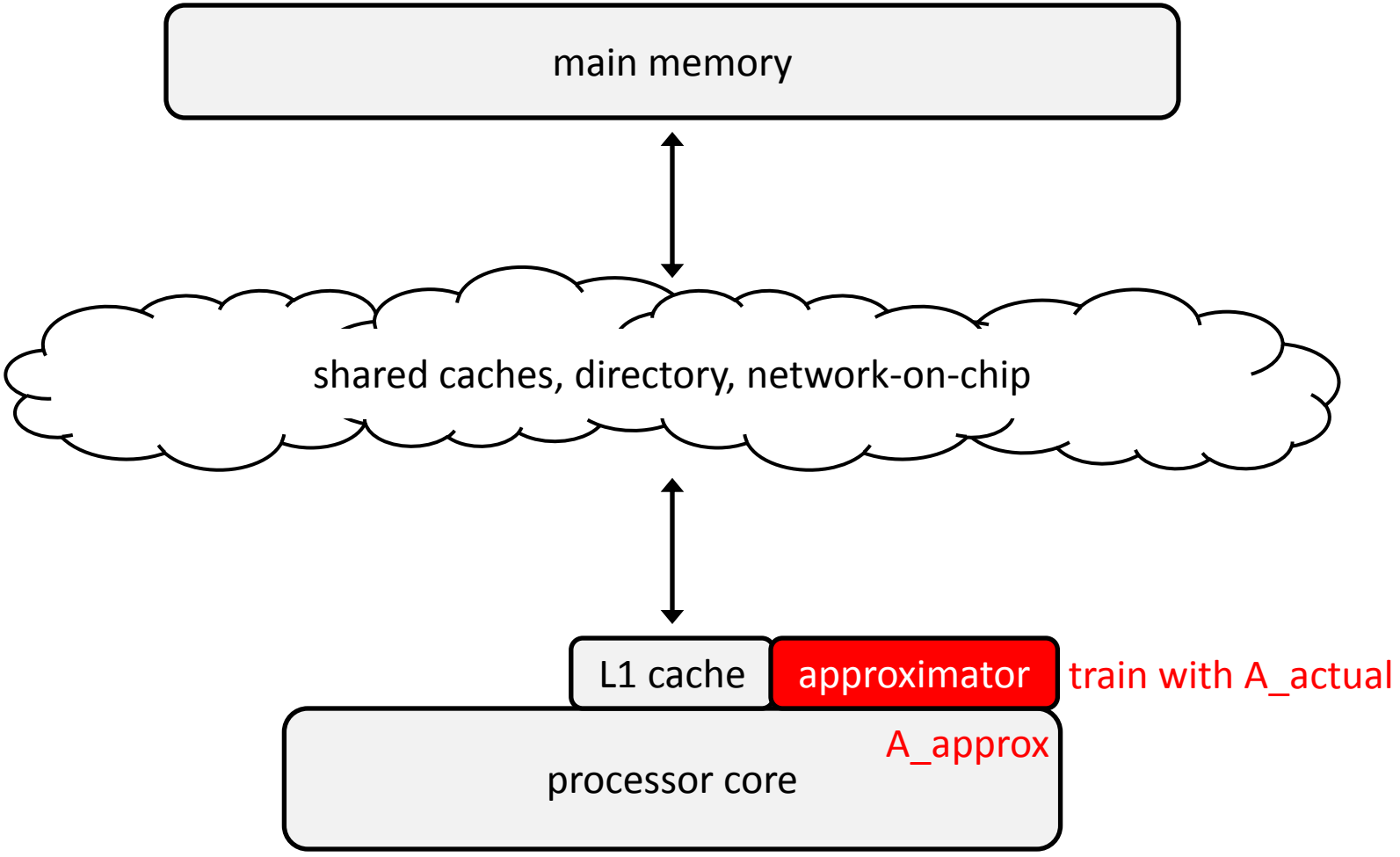
Load Value Approximation



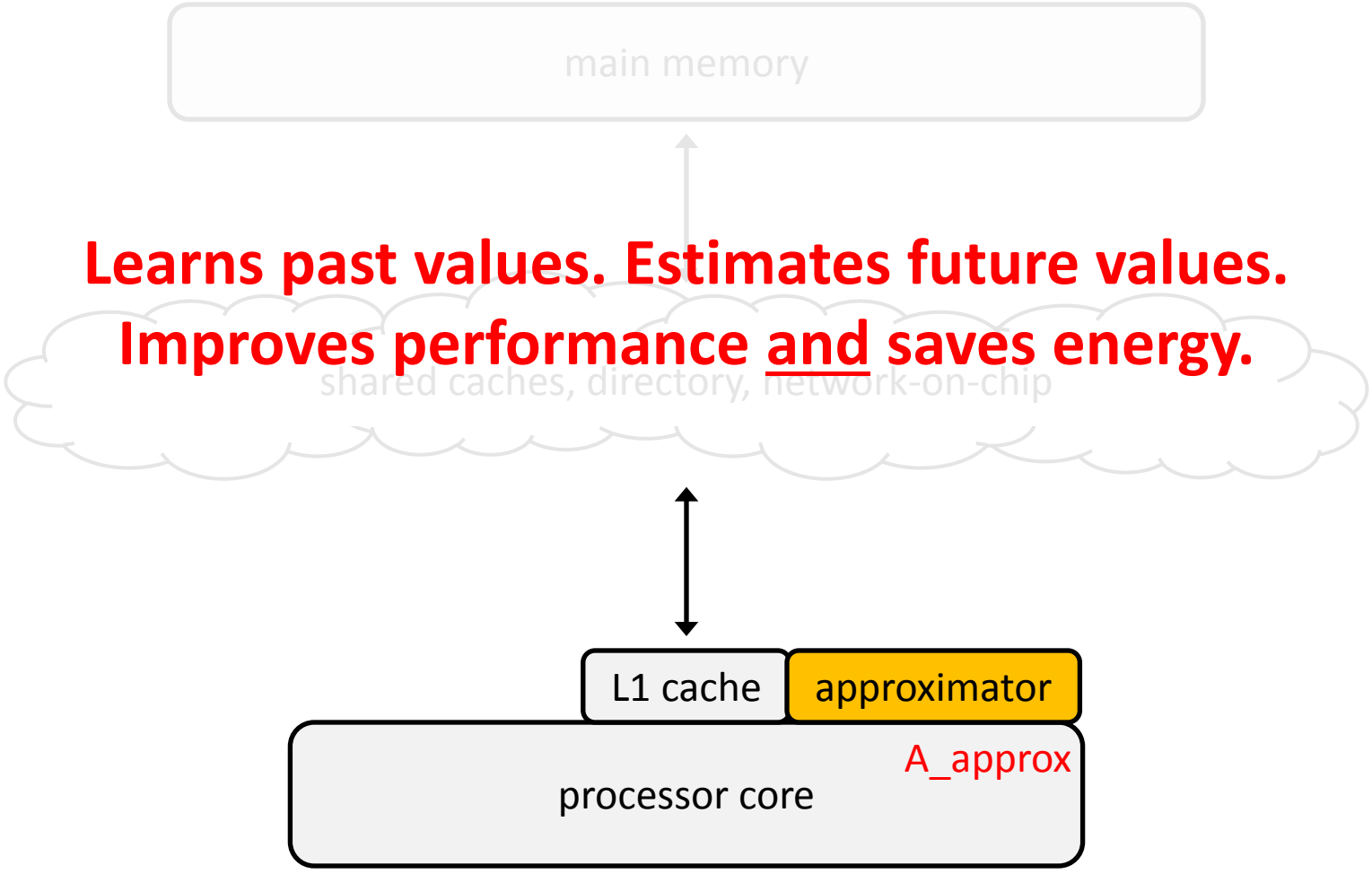
Load Value Approximation



Load Value Approximation

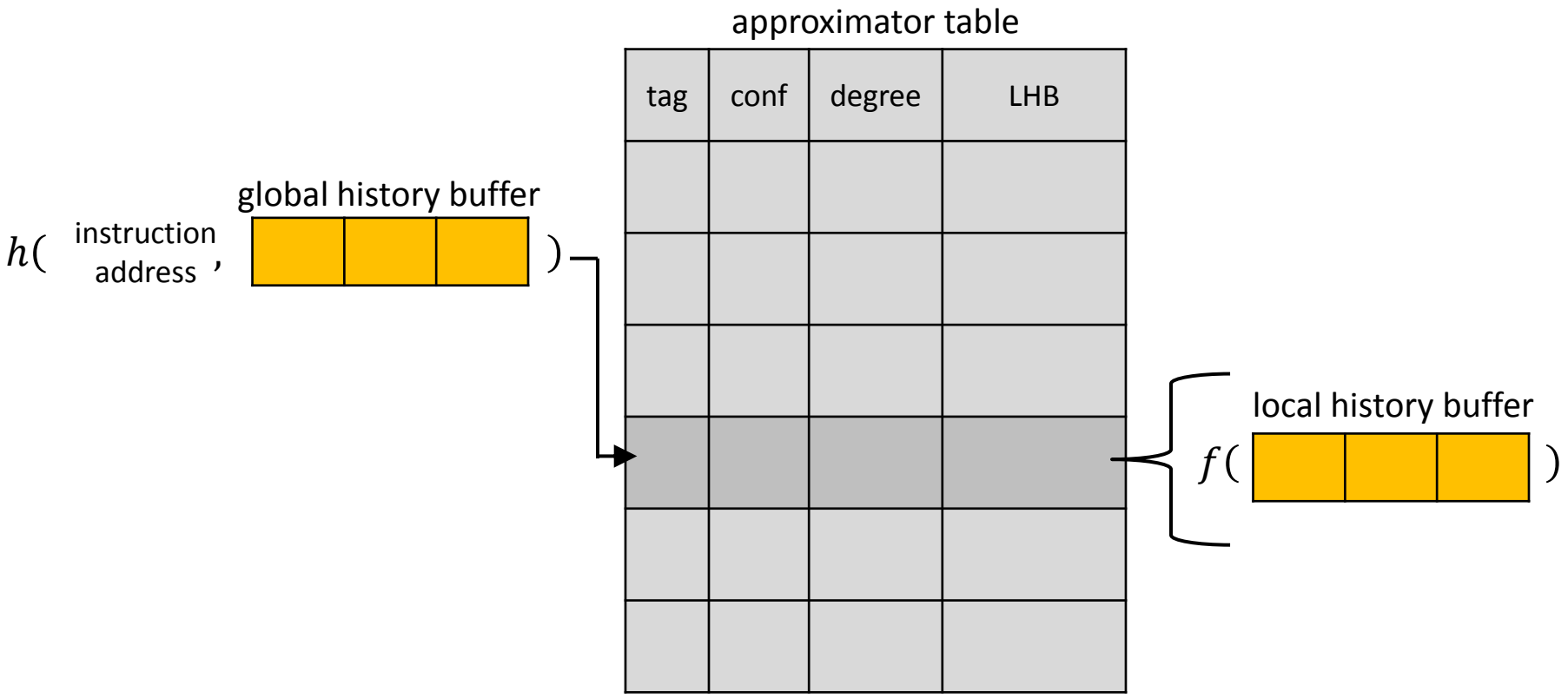


Load Value Approximation

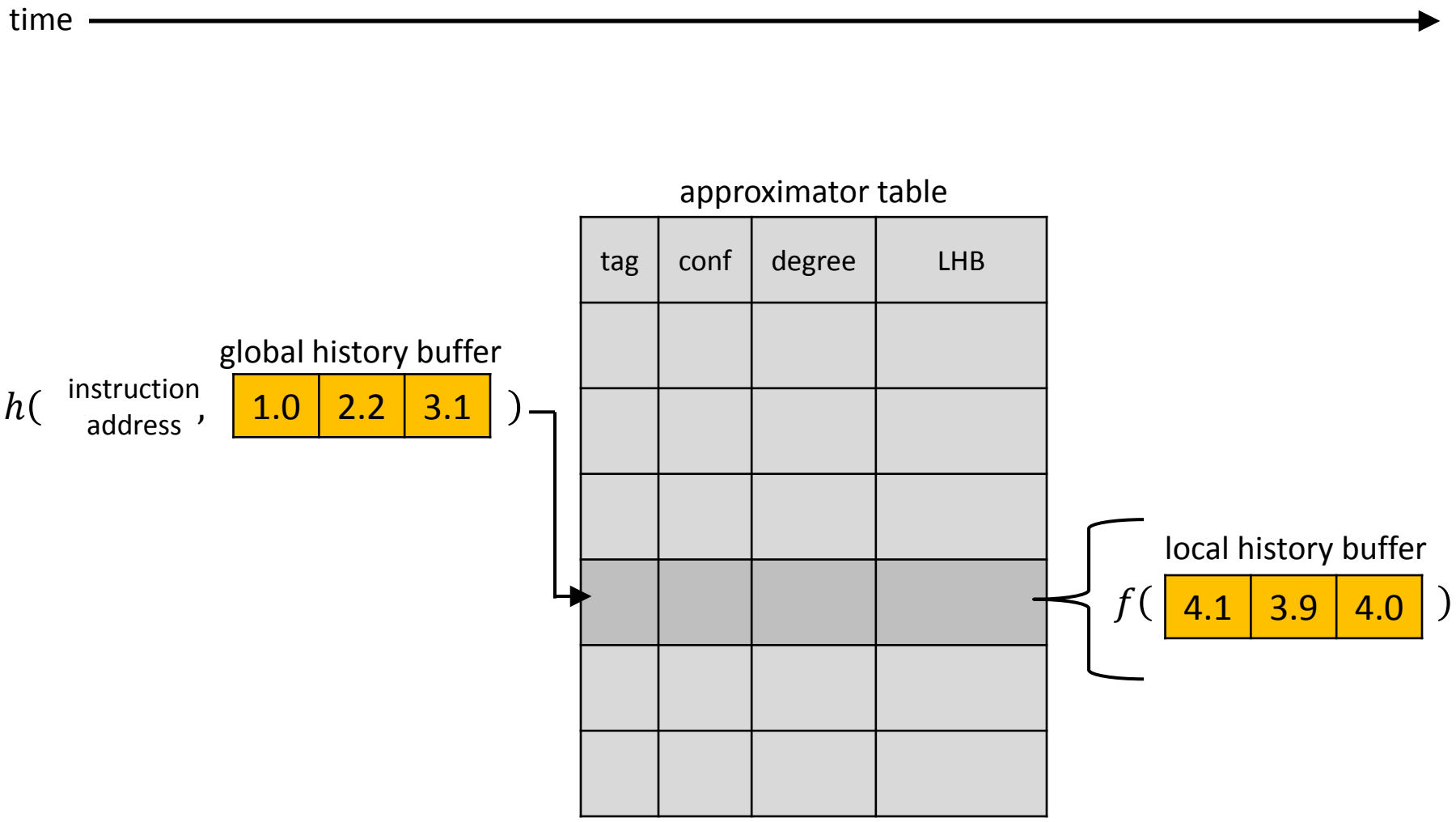


Learns past values. Estimates future values.
Improves performance and saves energy.

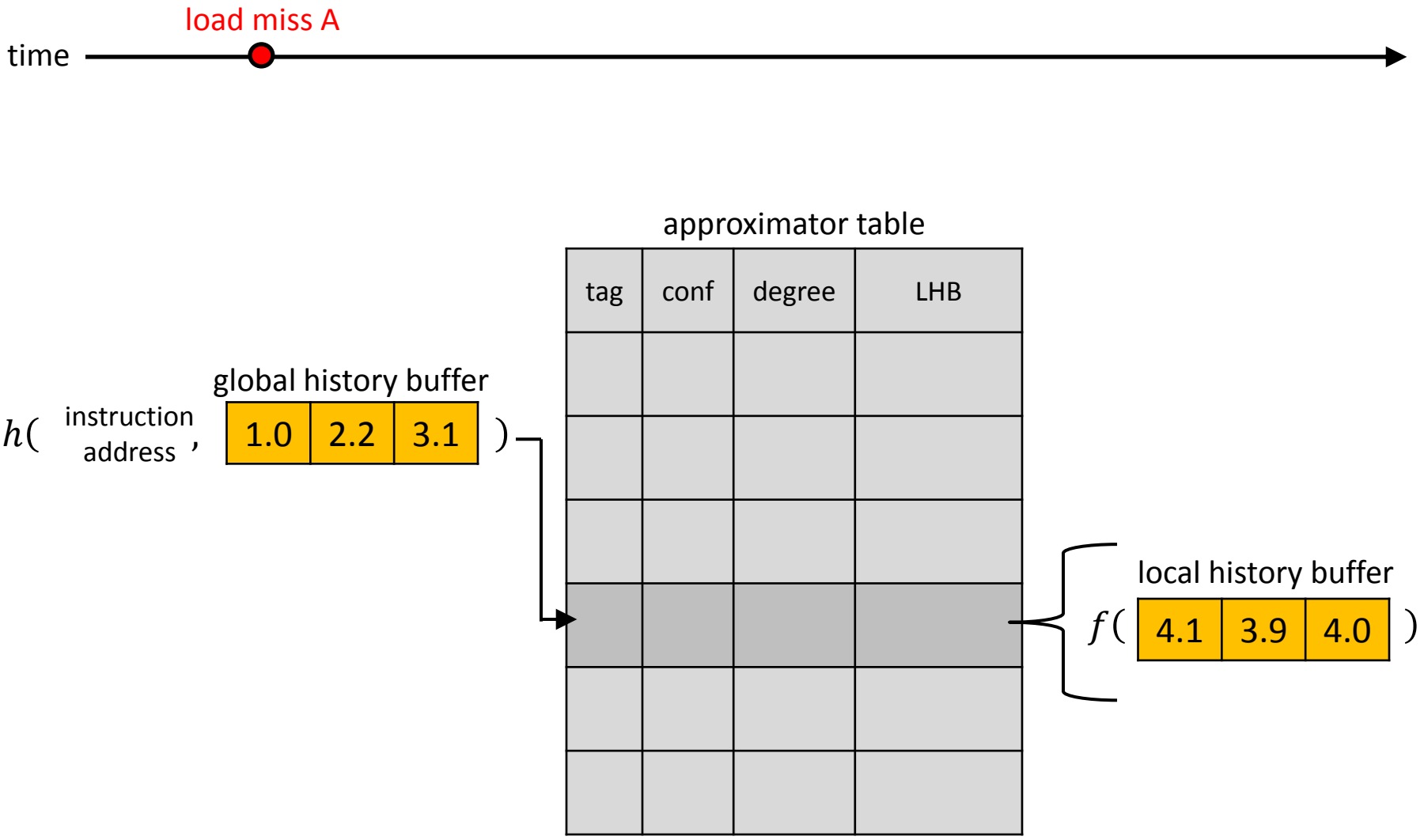
Approximator Design



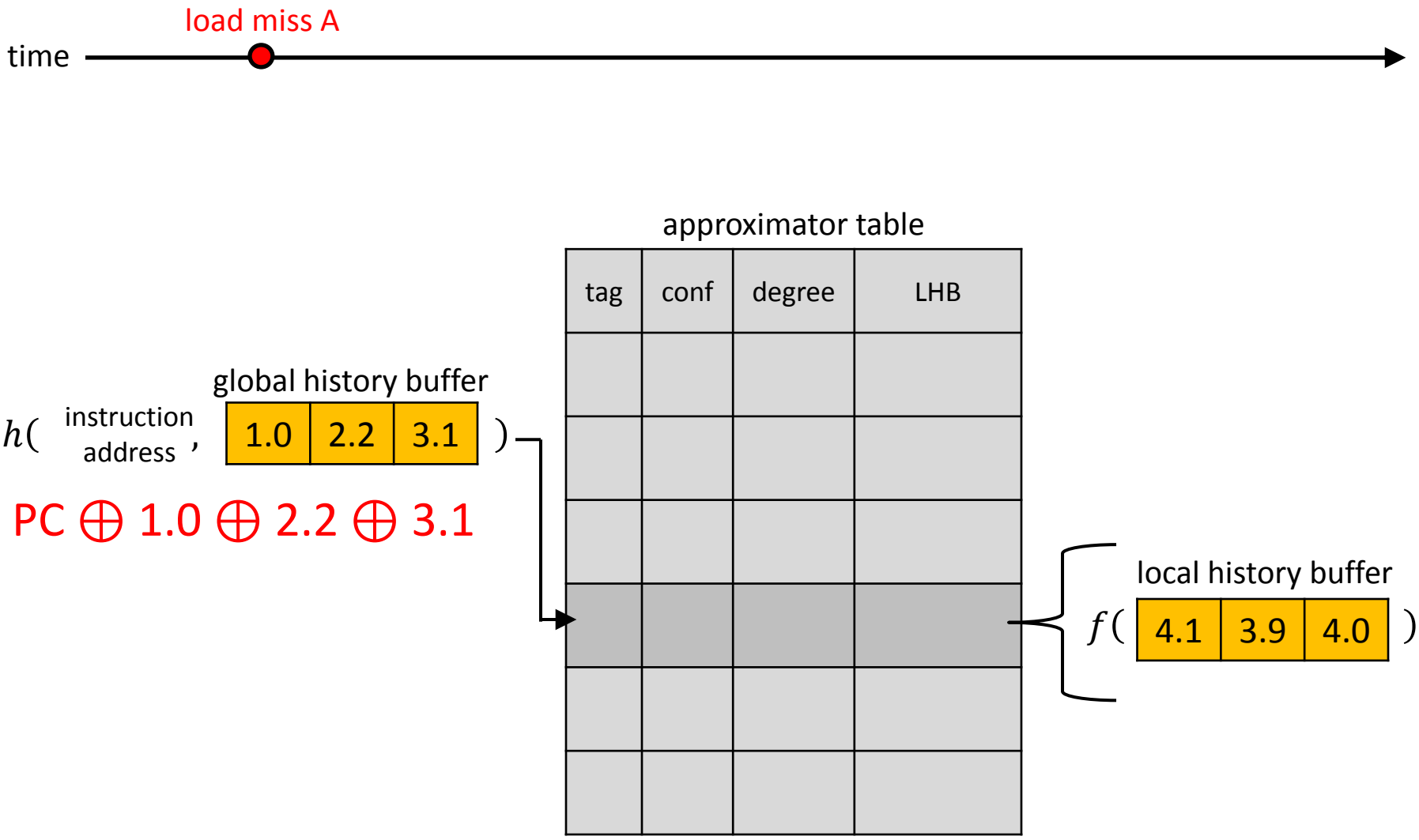
Approximator Design



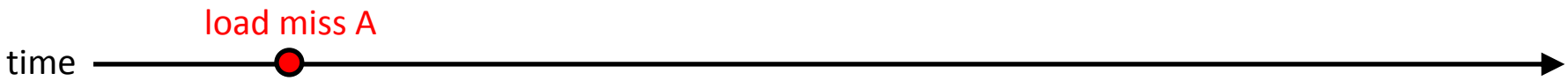
Approximator Design



Approximator Design



Approximator Design



approximator table

tag	conf	degree	LHB

$h($ instruction address , global history buffer: 1.0 | 2.2 | 3.1 $)$

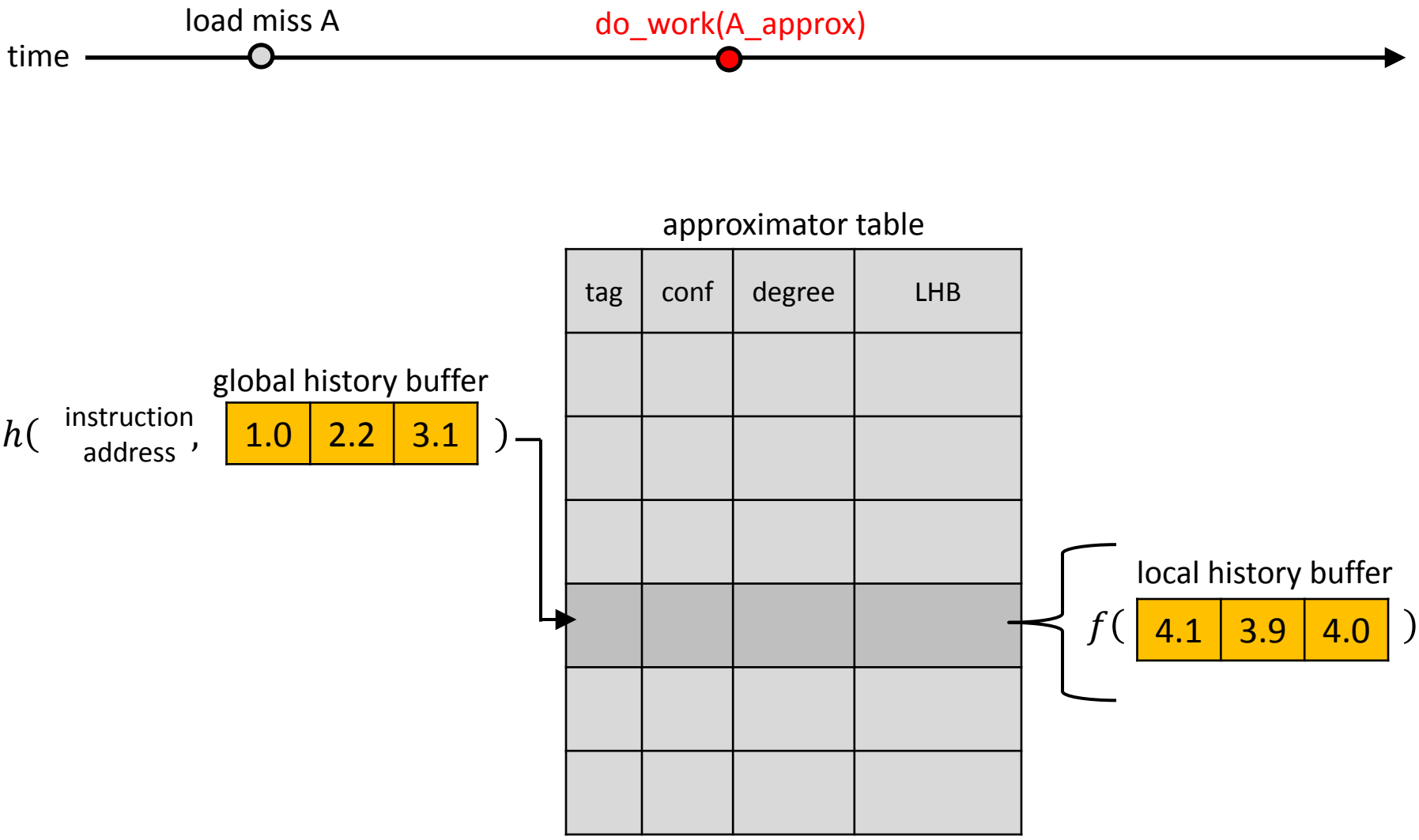
PC ⊕ 1.0 ⊕ 2.2 ⊕ 3.1

local history buffer: $f($ 4.1 | 3.9 | 4.0 $)$

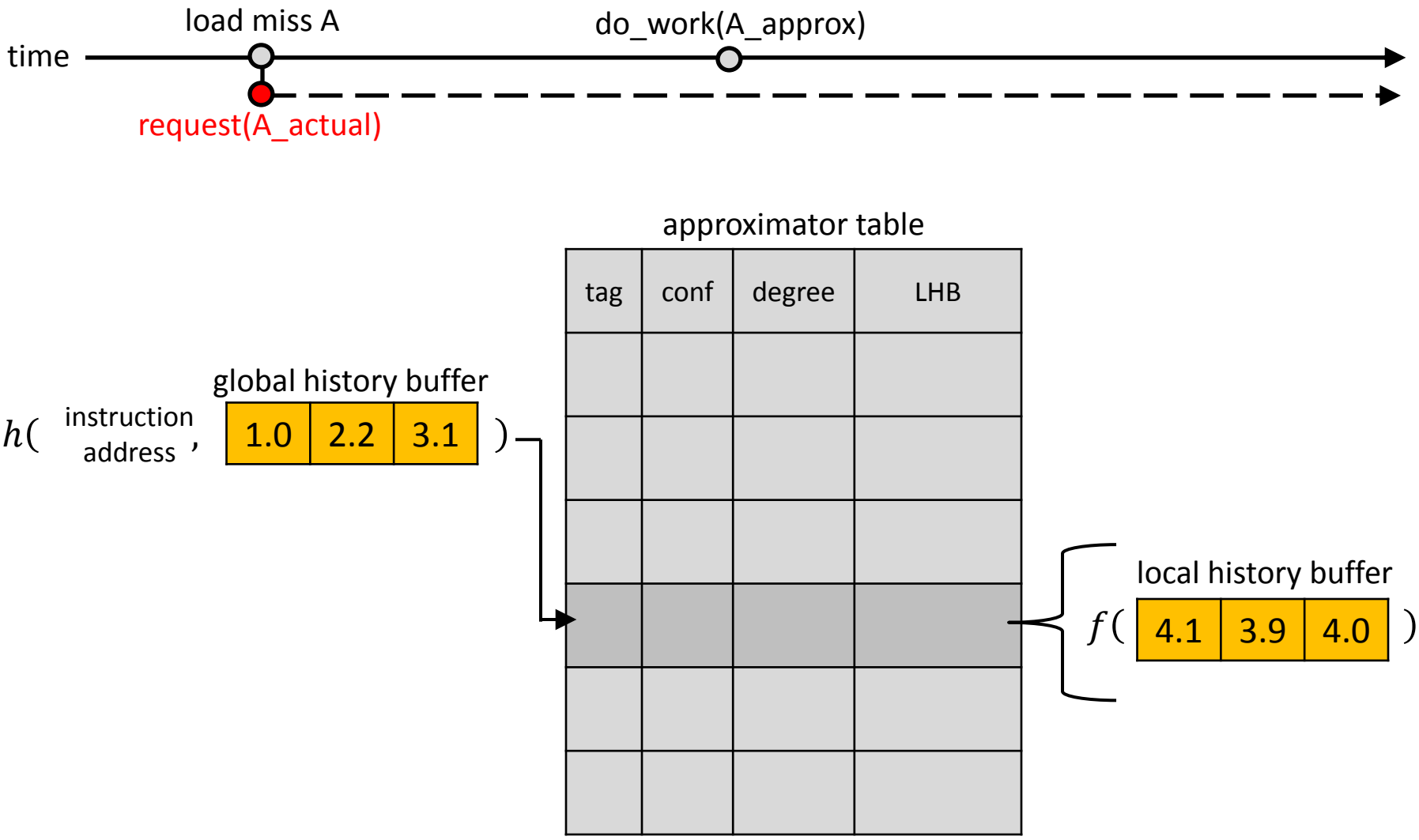
(4.1 + 3.9 + 4.0) / 3

A_approx = 4.0

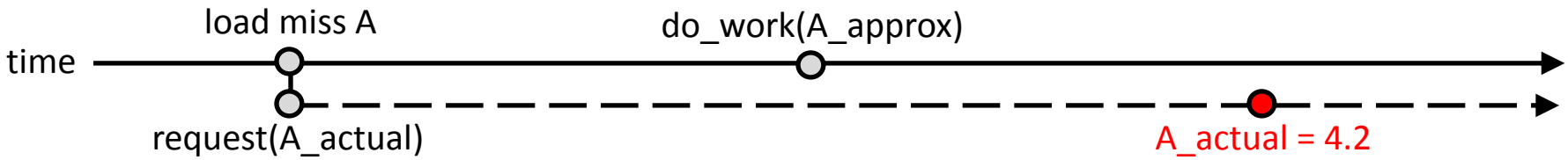
Approximator Design



Approximator Design



Approximator Design



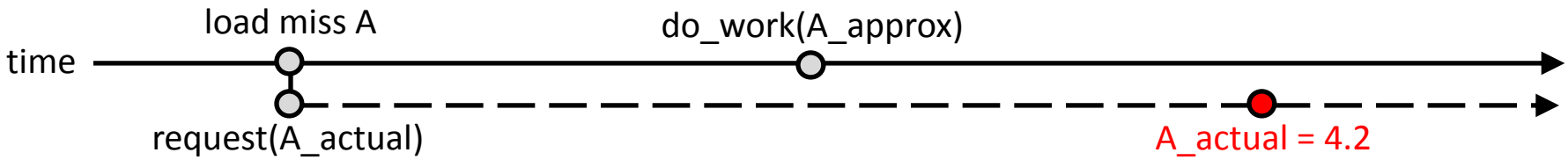
approximator table

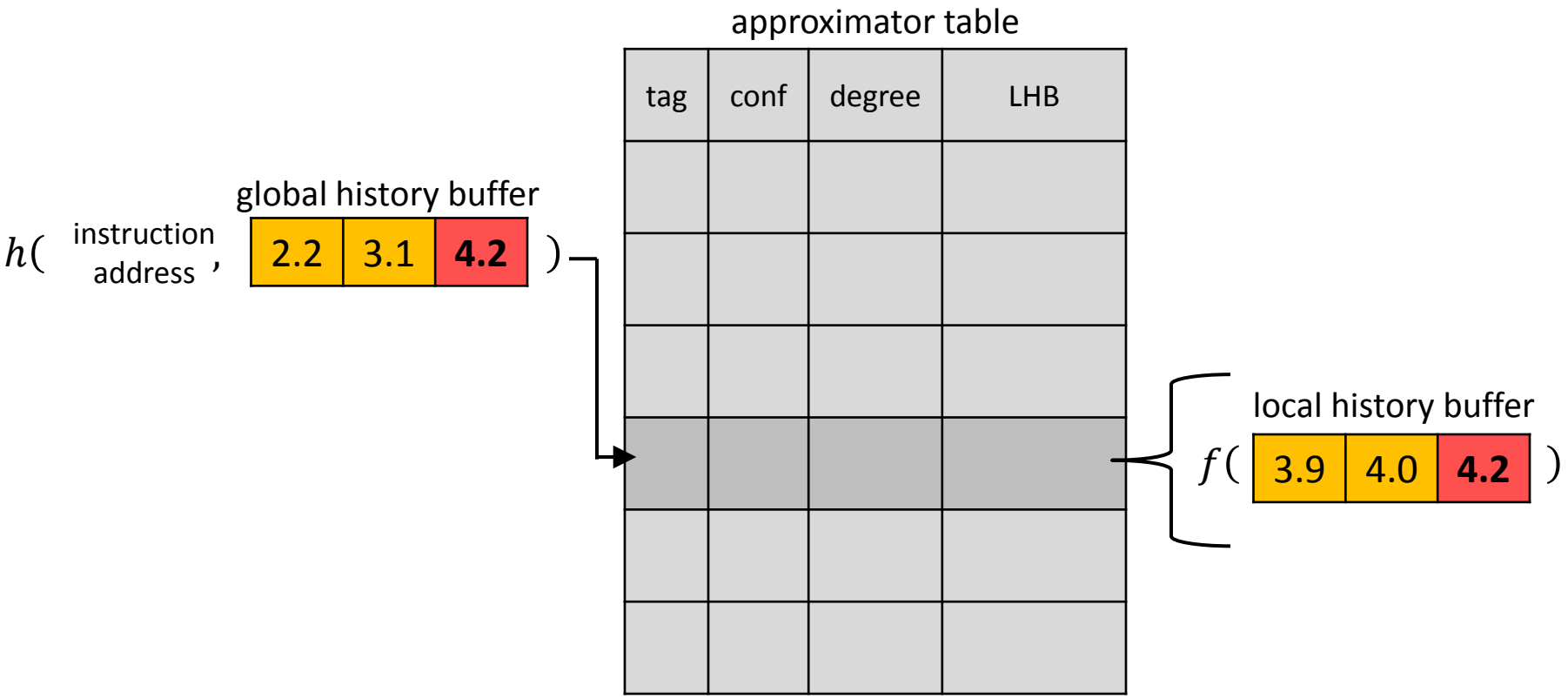
tag	conf	degree	LHB

$h($ instruction address , global history buffer: 1.0 | 2.2 | 3.1 $)$

$f($ local history buffer: 4.1 | 3.9 | 4.0 $)$

Approximator Design





Approximator Design – Other Considerations

- Floating-point precision
- History buffer sizes
- Stale values

More details in paper.

Approximator Design

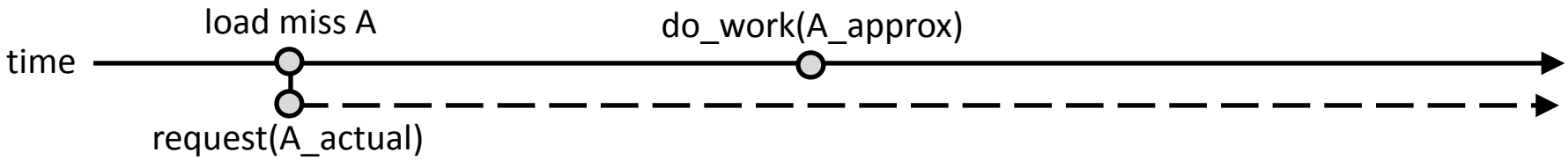
Relaxed Confidence Windows

- How do we avoid making bad approximations?
- Trade-off performance and error.

Approximation Degree

- Do we need to fetch the actual value from memory every time?
- Trade-off energy and error.

Relaxed Confidence Windows



approximator table

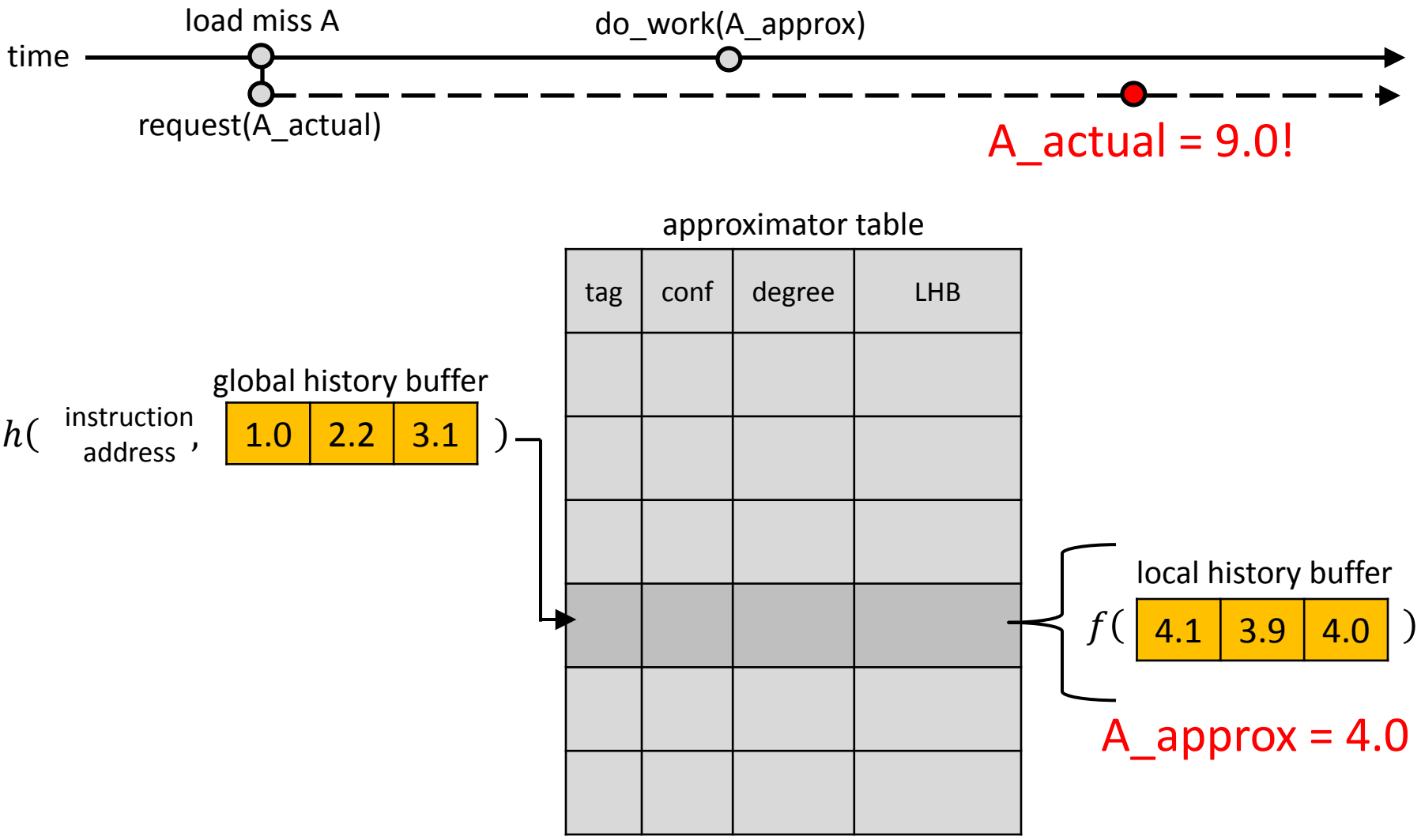
tag	conf	degree	LHB

$h($ instruction address , global history buffer: 1.0 | 2.2 | 3.1 $)$

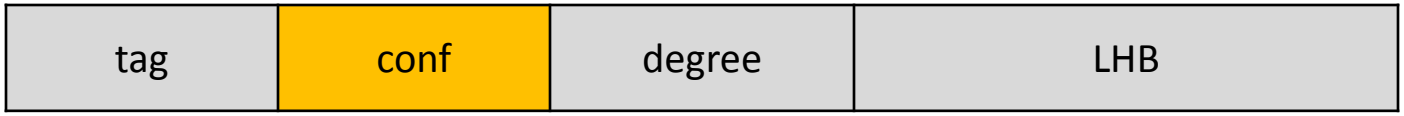
local history buffer: $f($ 4.1 | 3.9 | 4.0 $)$

A_approx = 4.0

Relaxed Confidence Windows



Relaxed Confidence Windows



When approximating:

if *conf* >= 0: use *A_approx*

else: don't use *A_approx*

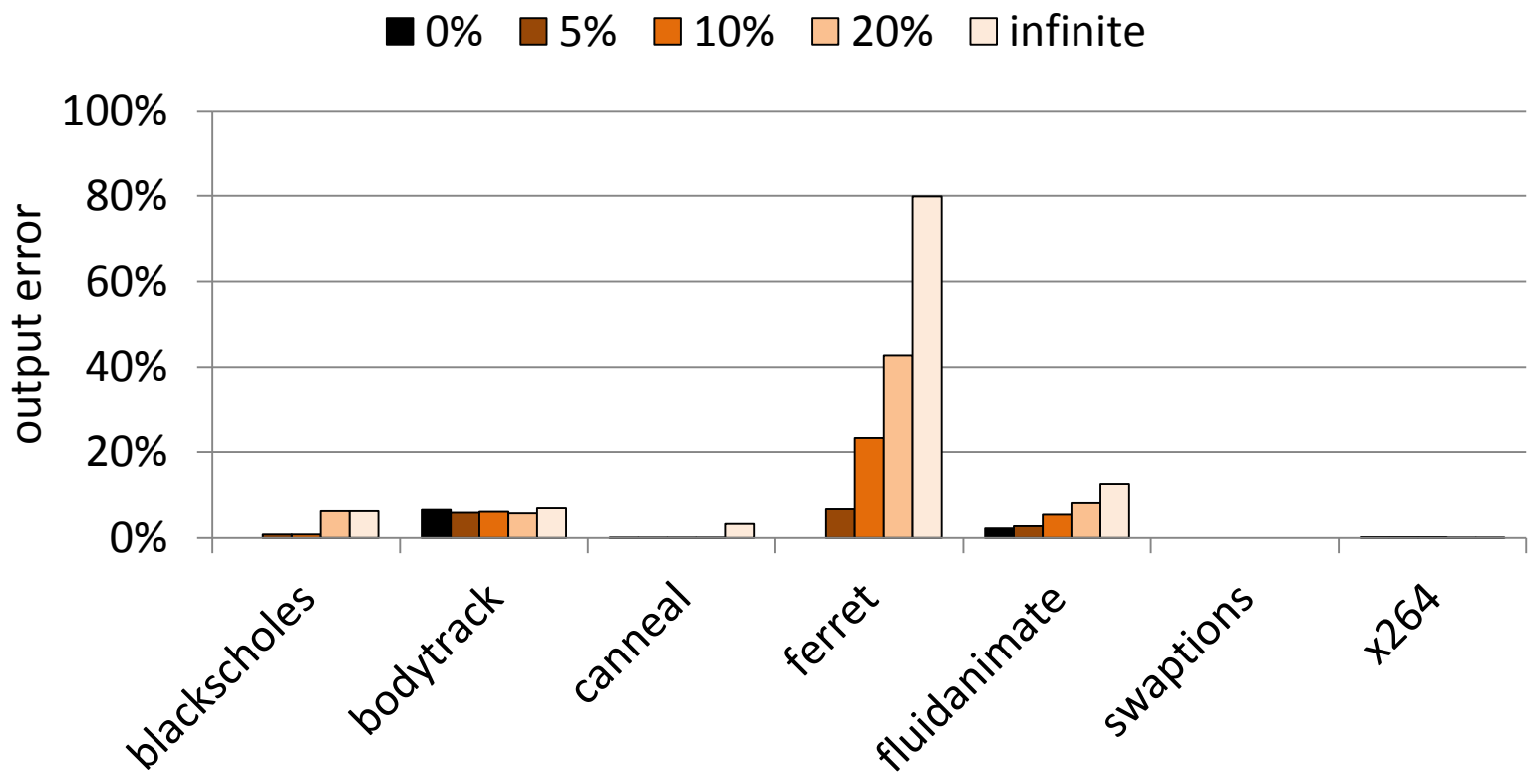
When updating:

if *A_approx*, *A_actual* differ by <= ***CONF_WINDOW%***: *conf*++

else: *conf*--

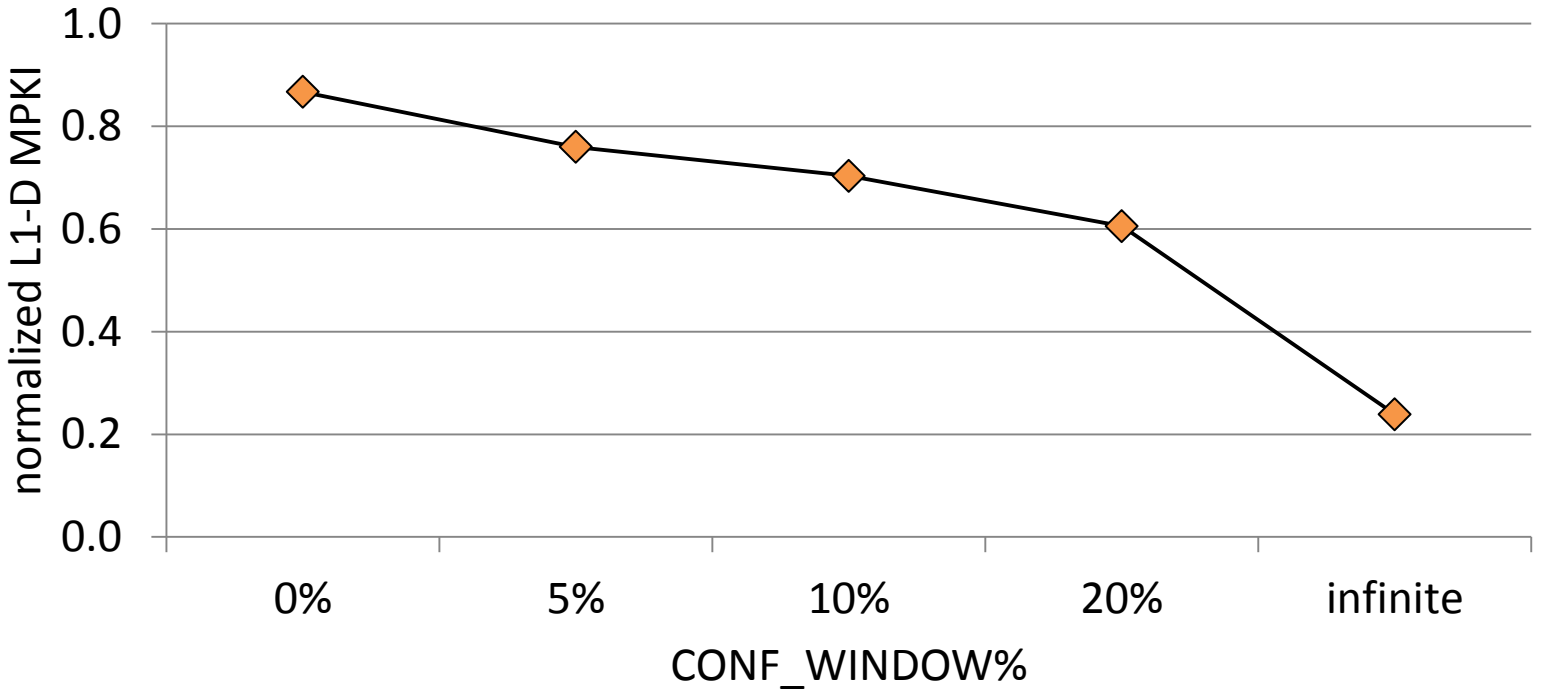
Relaxed Confidence Windows – Output Error

Varying *CONF_WINDOW%*:



Relaxed Confidence Windows – L1-D MPKI

Varying *CONF_WINDOW%*:



Approximator Design

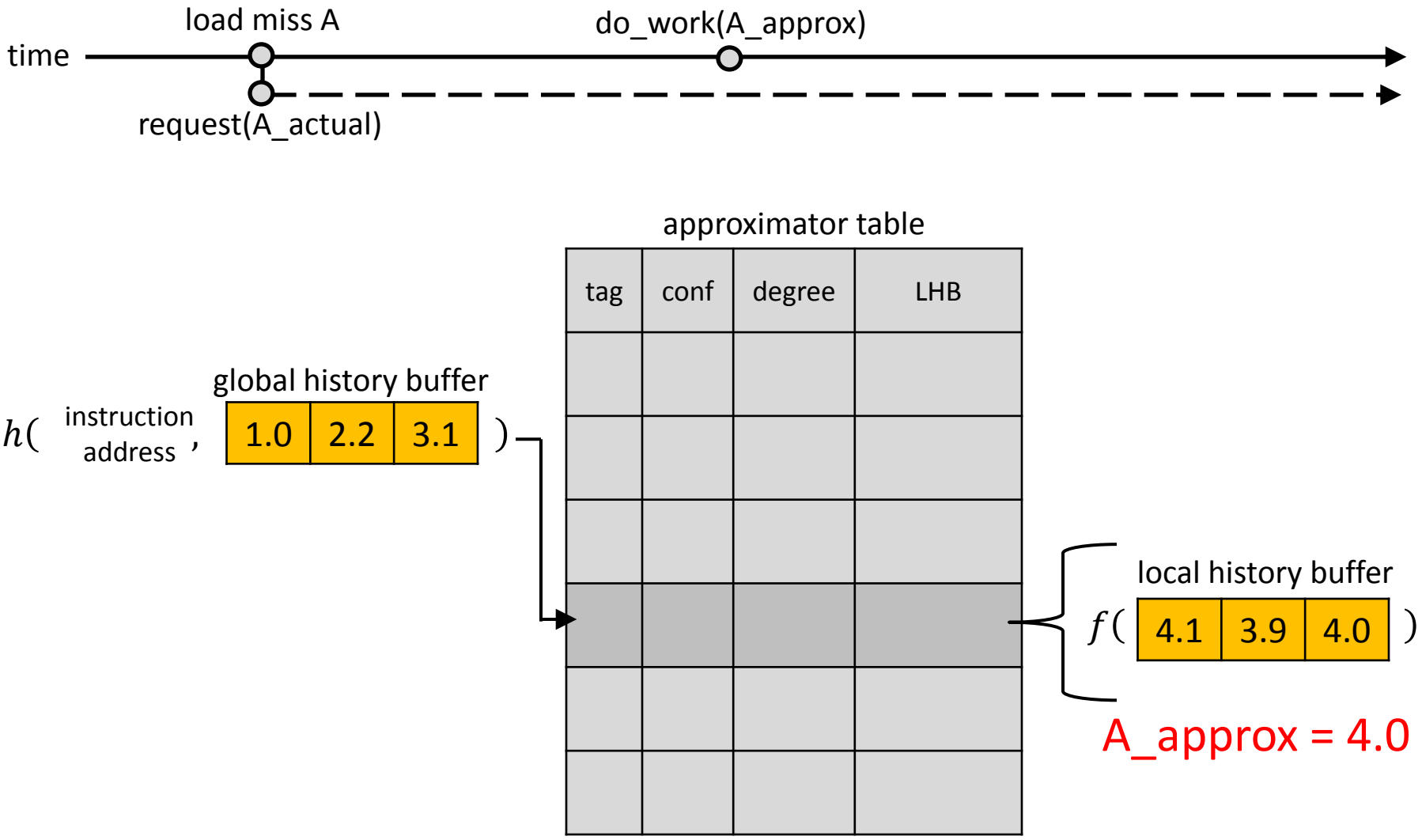
Relaxed Confidence Windows

- How do we avoid making bad approximations?
- Trade-off performance and error.

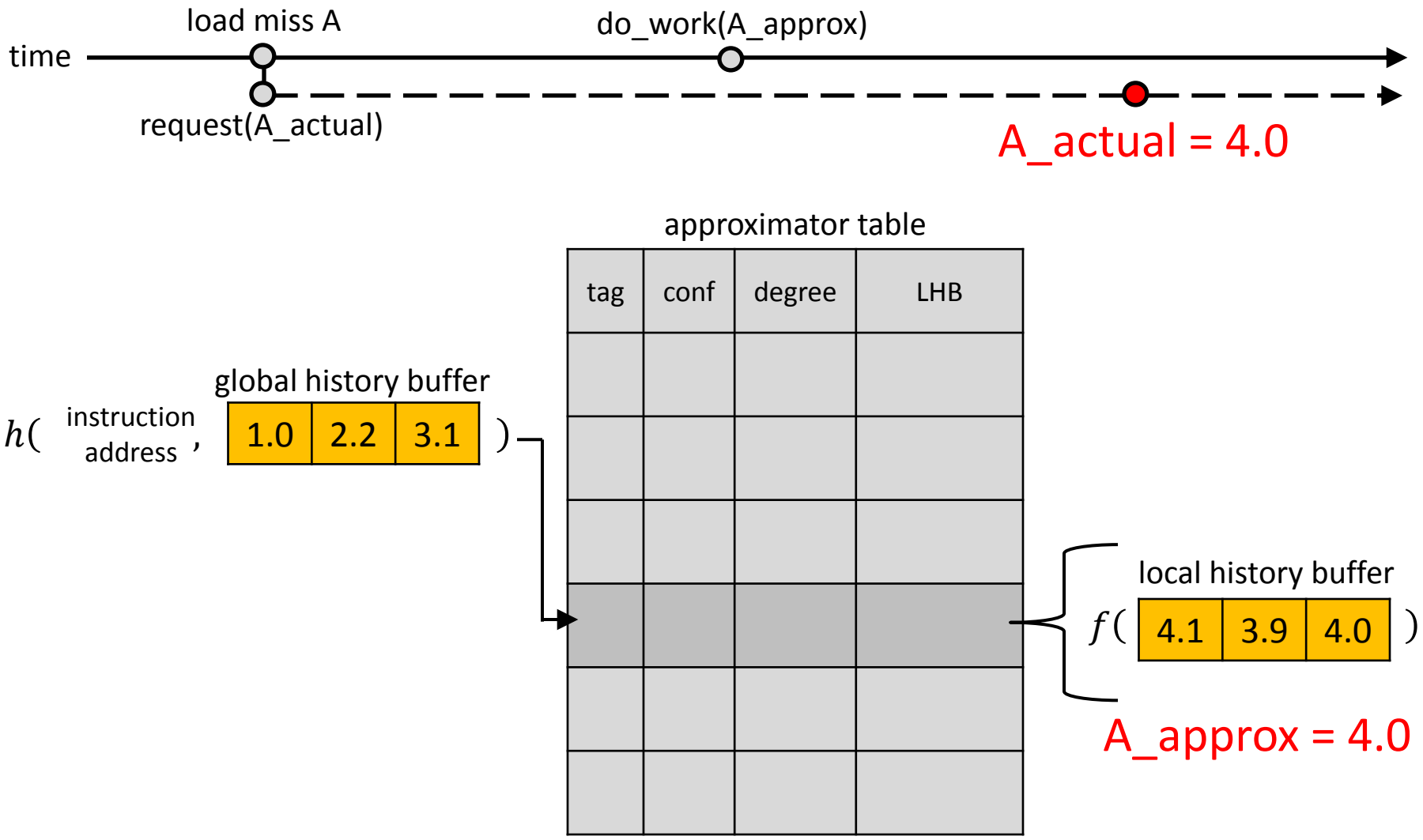
Approximation Degree

- **Do we need to fetch the actual value from memory every time?**
- **Trade-off energy and error.**

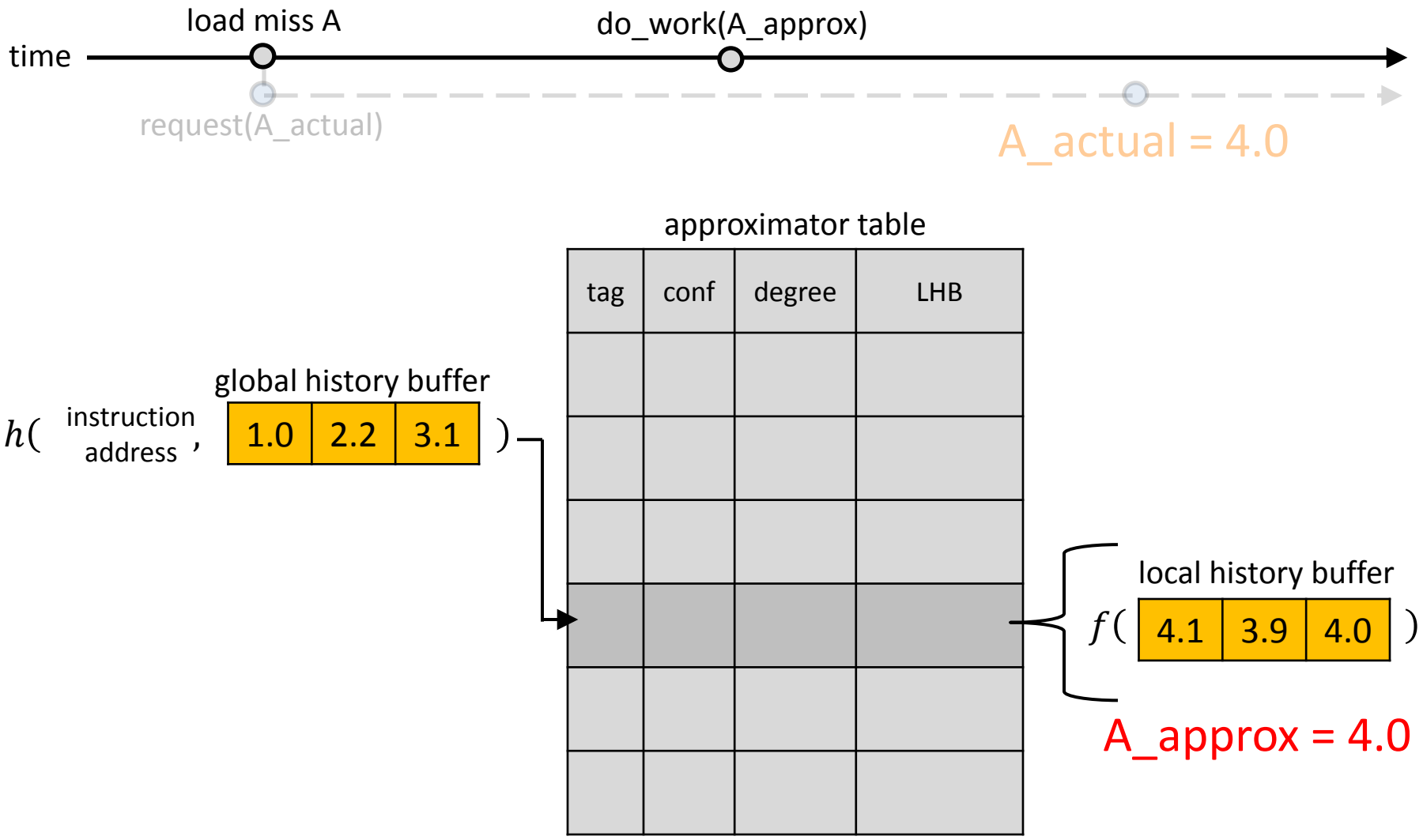
Approximation Degree



Approximation Degree



Approximation Degree



Approximation Degree

When approximating:

if *degree* == ***APPROX_DEGREE***: fetch *A_actual*

else: don't fetch *A_actual*

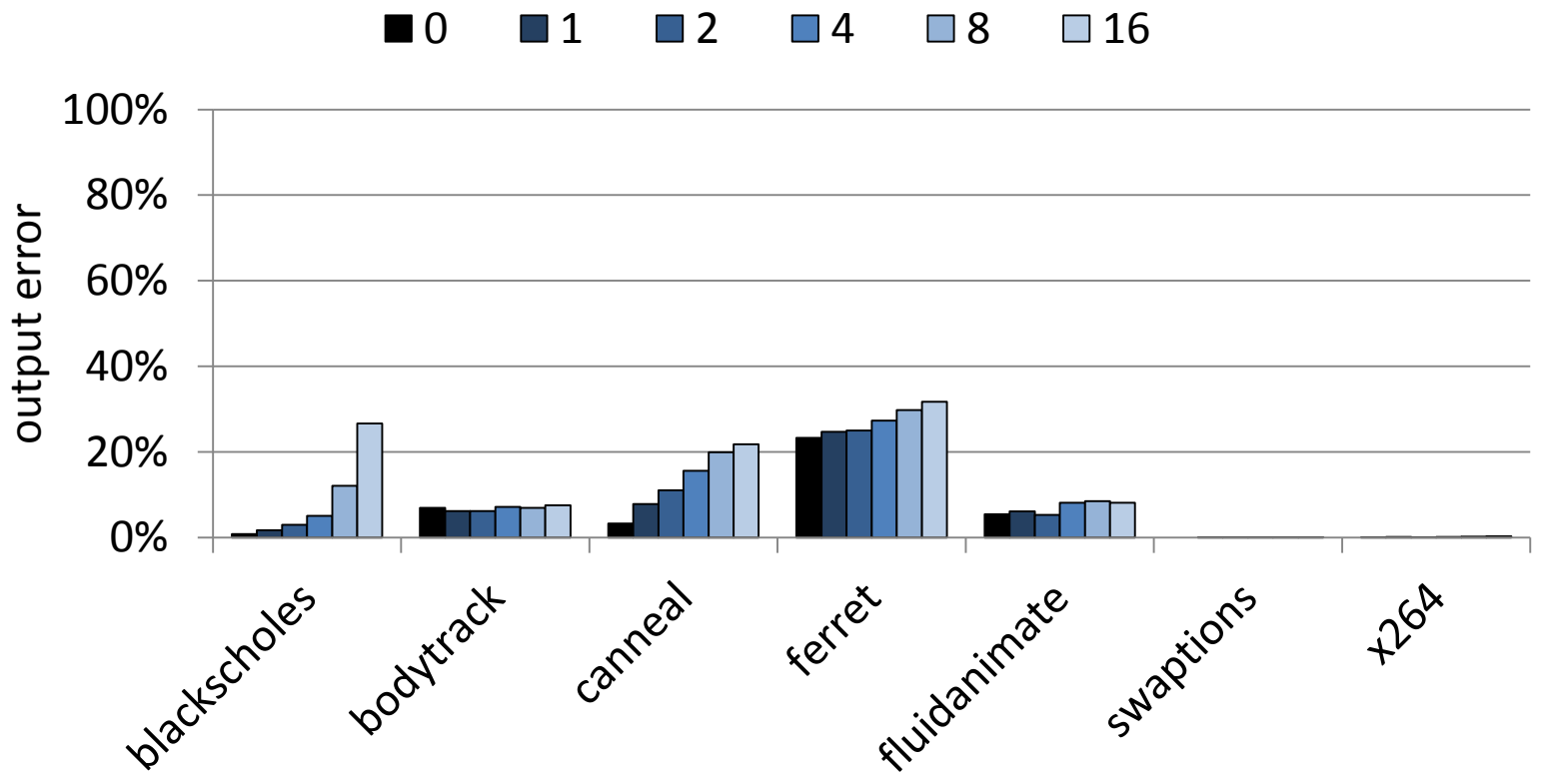
When updating:

if *degree* == ***APPROX_DEGREE***: *degree* = 0

else: *degree*++

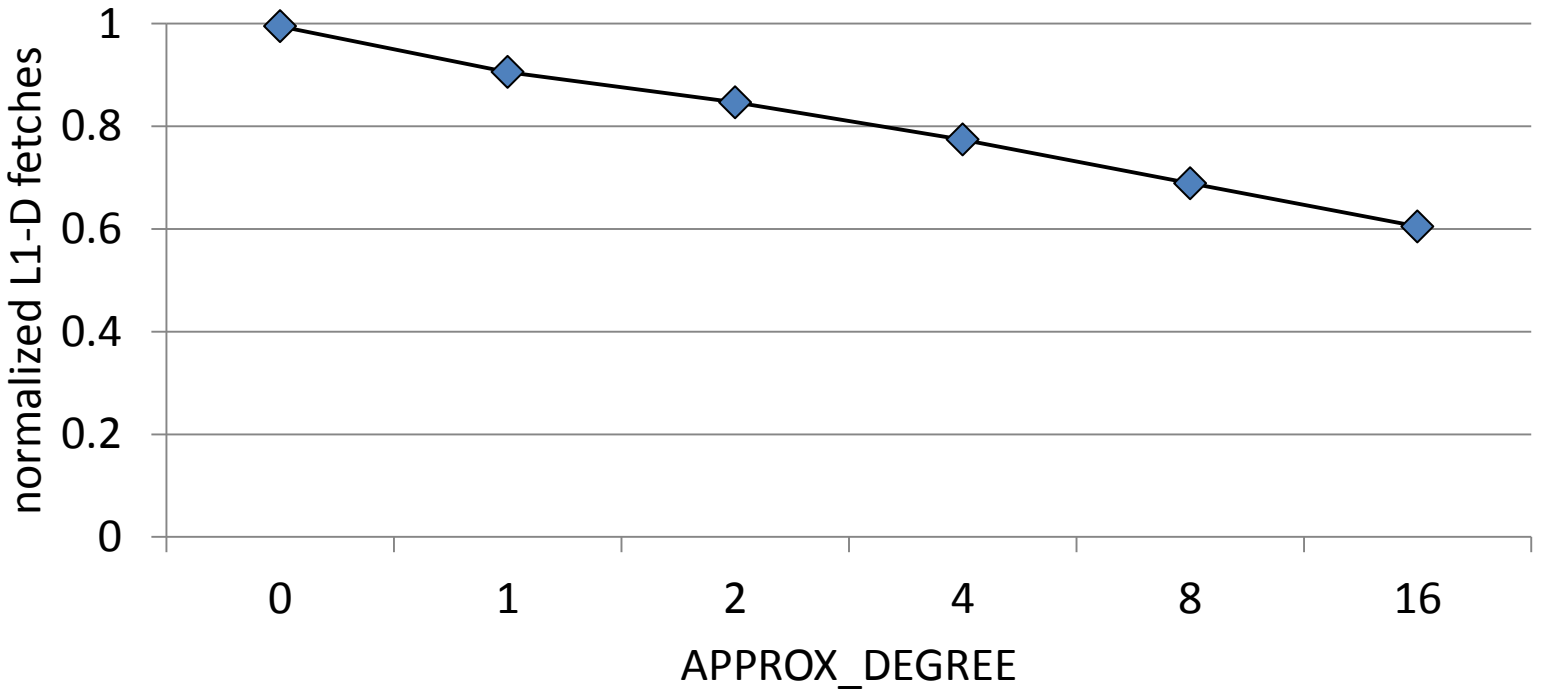
Approximation Degree – Output Error

Varying *APPROX_DEGREE*:



Approximation Degree – L1-D Fetches

Varying *APPROX_DEGREE*:



Methodology

Multi-threaded approximate applications

- PARSEC benchmark suite [Bienia, Princeton 2011]
- Programmer annotations and ISA extensions [Esmaeilzadeh, ASPLOS 2012]

Approximator design space exploration

- Pin dynamic binary instrumentation tool [Luk, PLDI 2005]

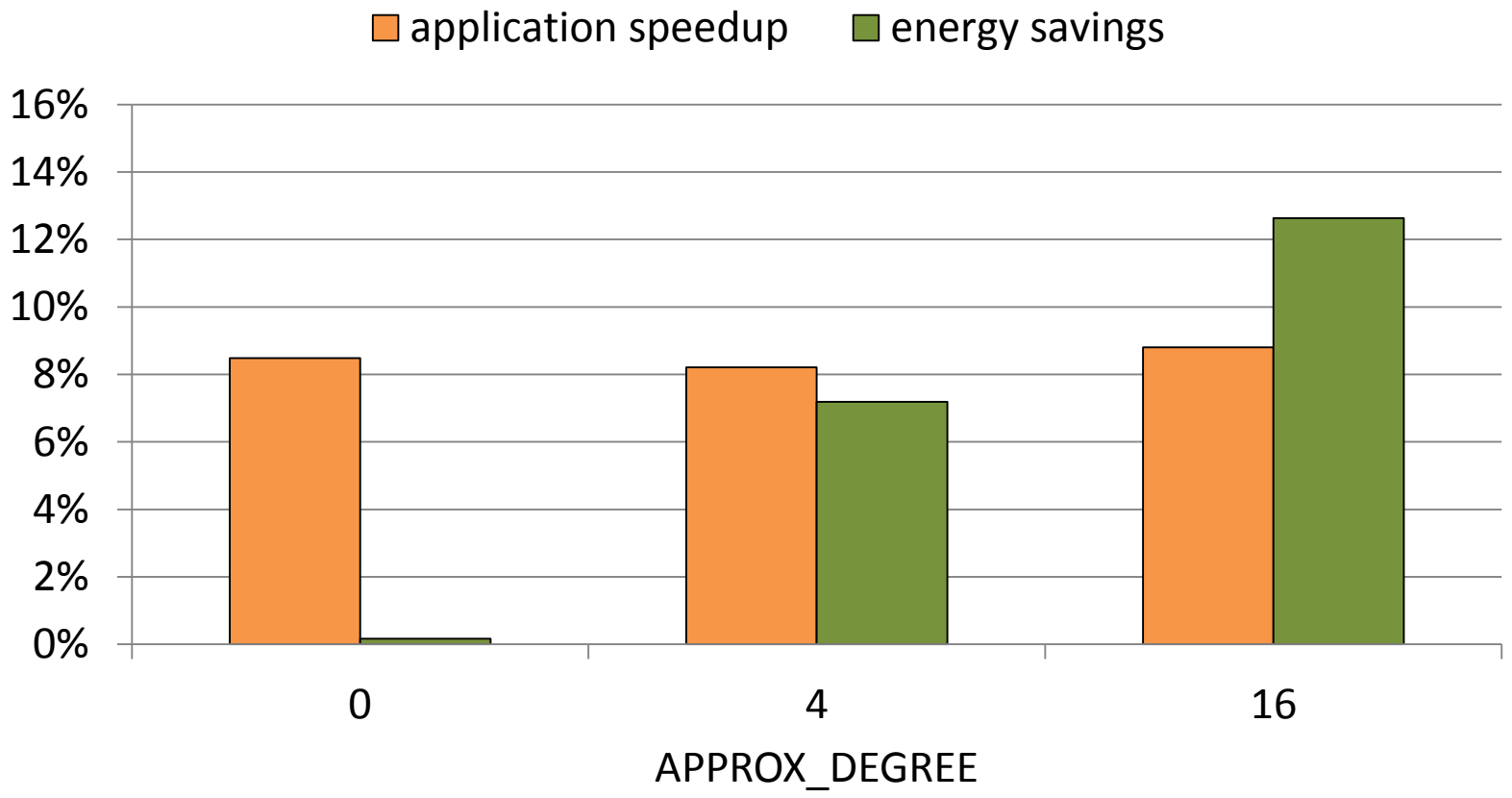
Full-system simulation

- FeS2 cycle-level x86 simulator [Neelakantam, ASPLOS 2008]

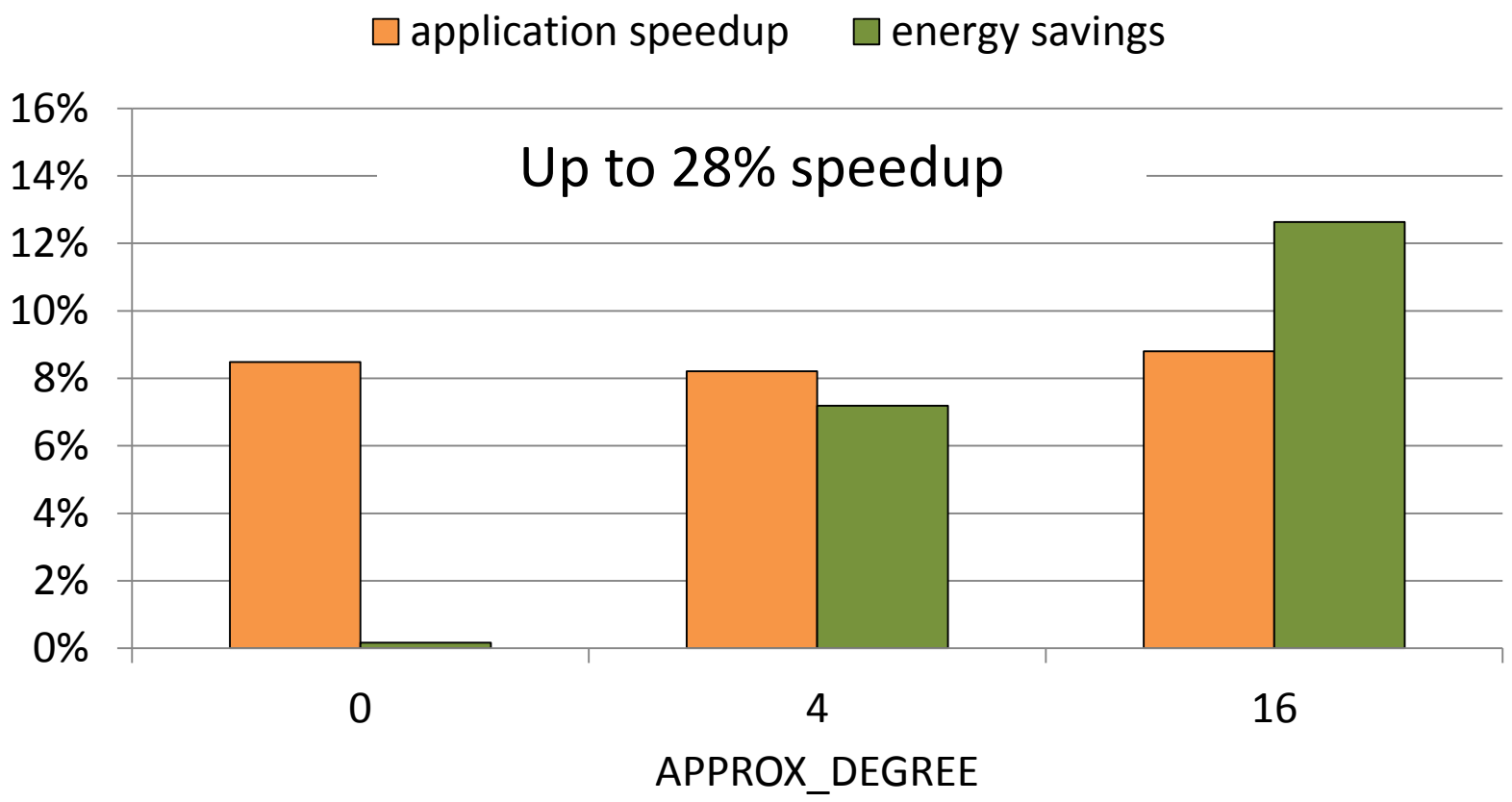
Approximator, cache and memory energy consumption

- CACTI modeling tool [Thoziyoor, HP 2008]

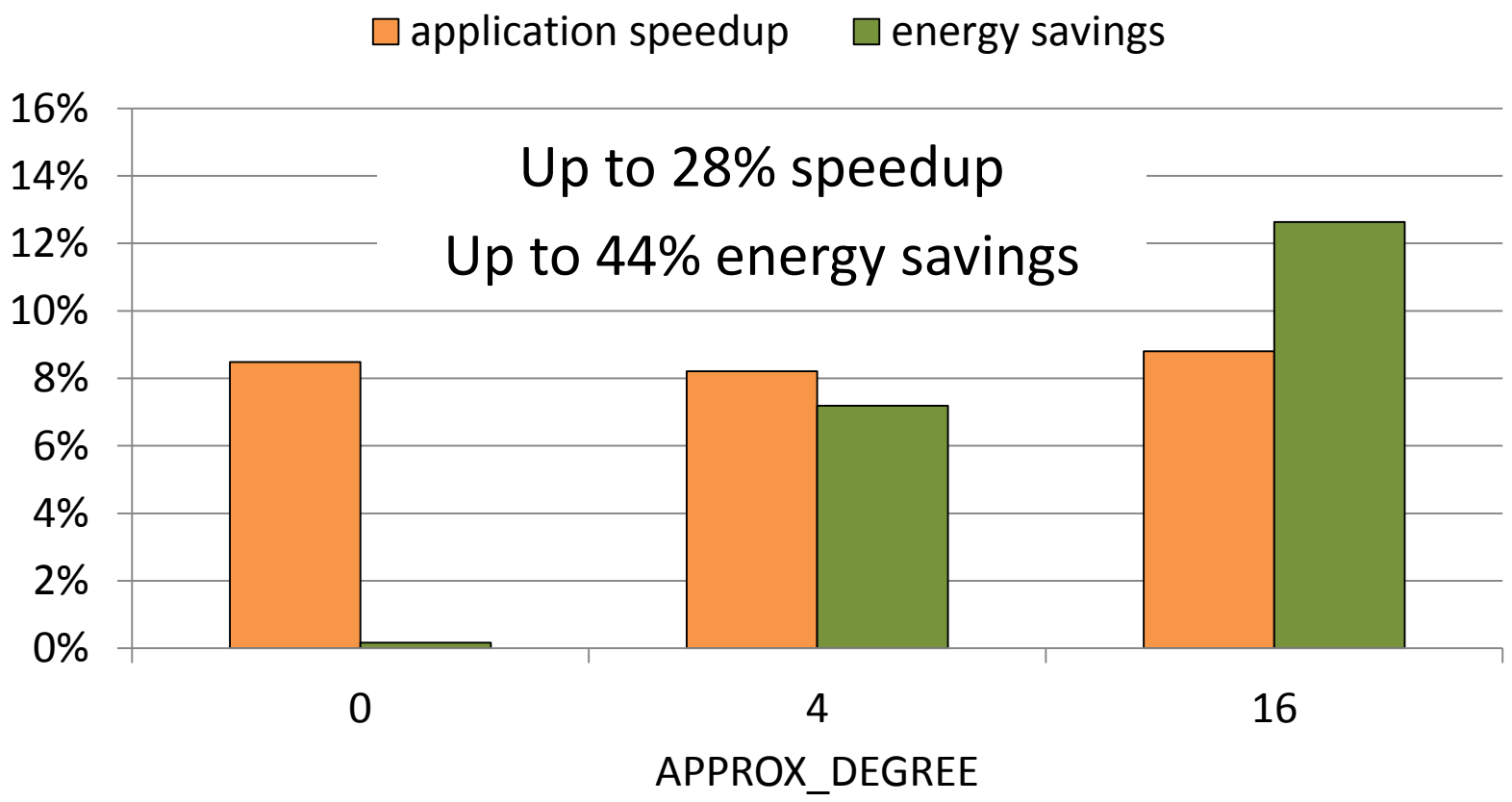
Evaluation


application speedup
energy savings
16%
14%
12%
10%
8%
6%
4%
2%
0%
0
4
16
APPROX_DEGREE

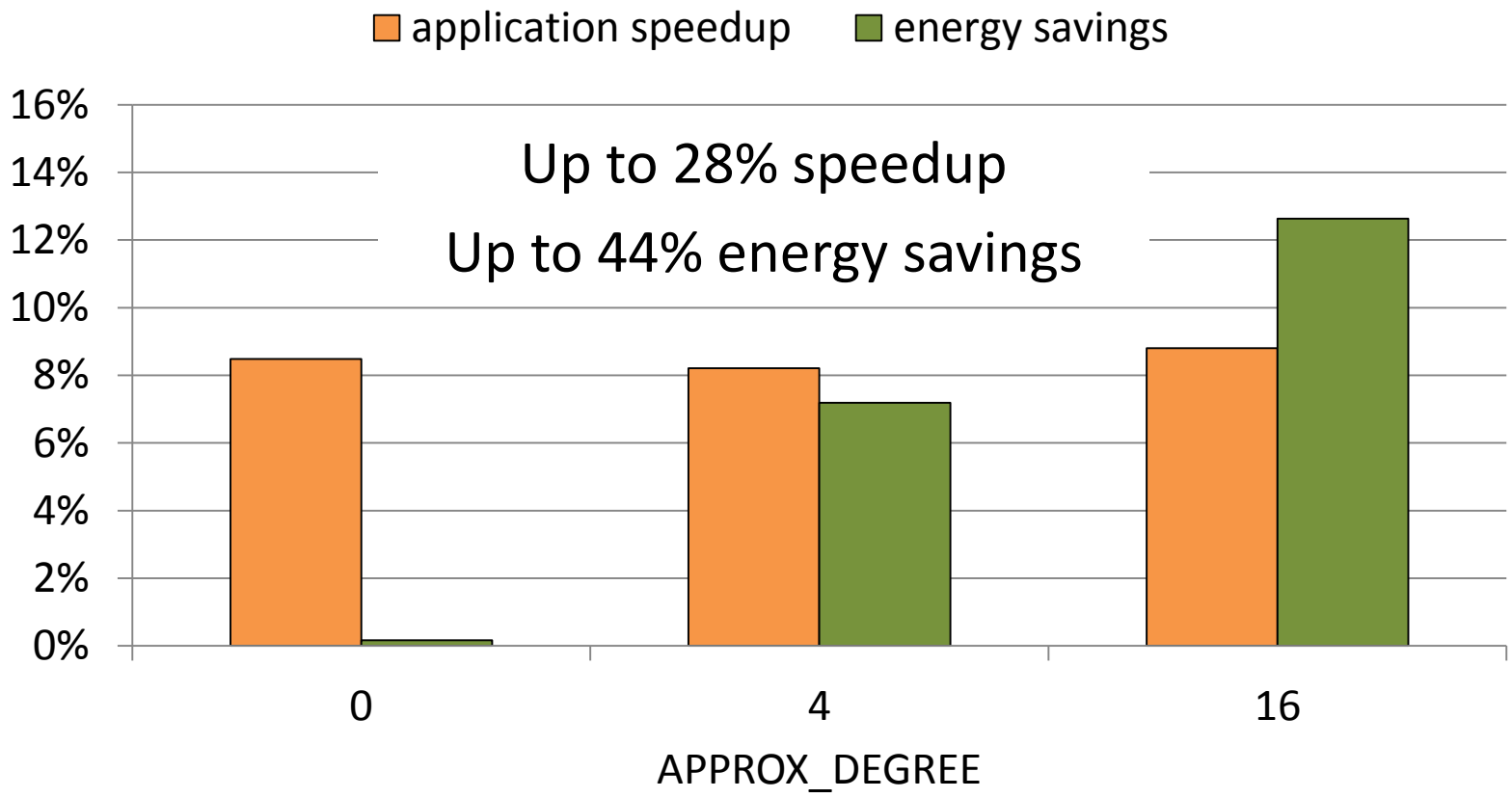
Evaluation


application speedup
energy savings
16%
14%
12%
10%
8%
6%
4%
2%
0%
Up to 28% speedup
0
4
16
APPROX_DEGREE

Evaluation


application speedup
energy savings
16%
14%
12%
10%
8%
6%
4%
2%
0%
Up to 28% speedup
Up to 44% energy savings
0
4
16
APPROX_DEGREE

Evaluation



Reduces L1-D MPKI by 30% over traditional value predictor and prefetcher.

Conclusion

Load Value Approximation

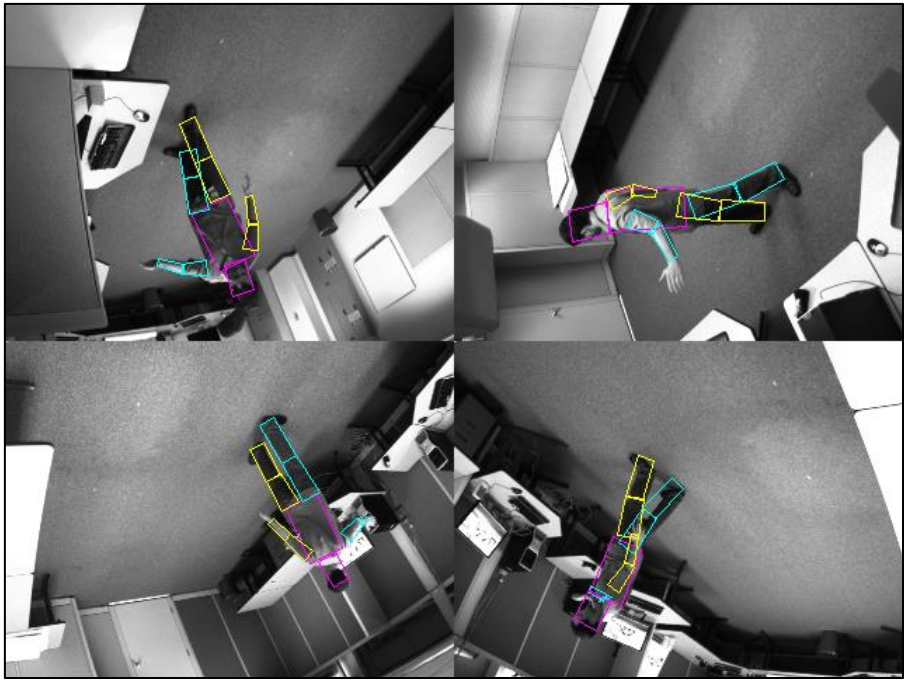
- Approximate Value Locality
- Non-Speculative
- Relaxed Confidence Windows
- Approximation Degree

↑ performance

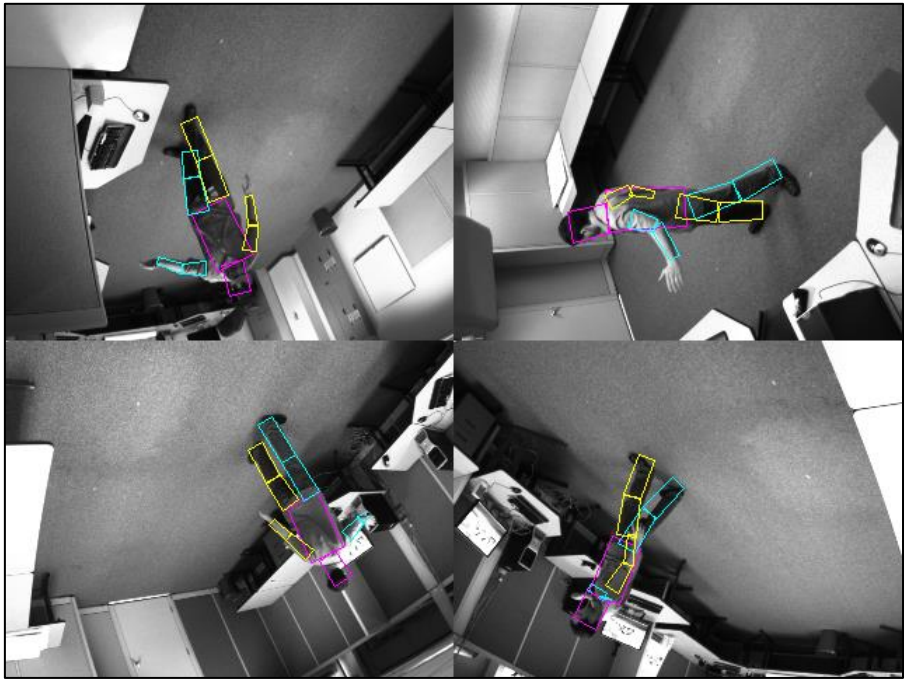
↓ energy consumption

low output error

Conclusion



baseline (precise)



load value approximation