Adaptive Data Prefetching for High Performance Processors

Lionel Vincent, Stéphane Mancini,
Henri-Pierre Charles, Suzanne Lesecq

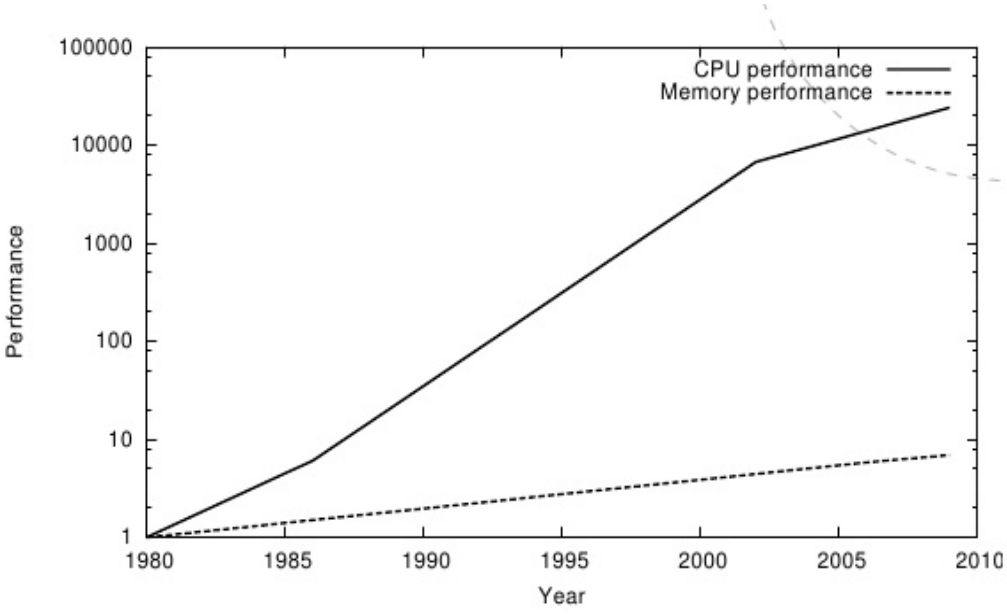
21/01/2015

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01)

Context

Memory Wall

– limits the computational performance of processors

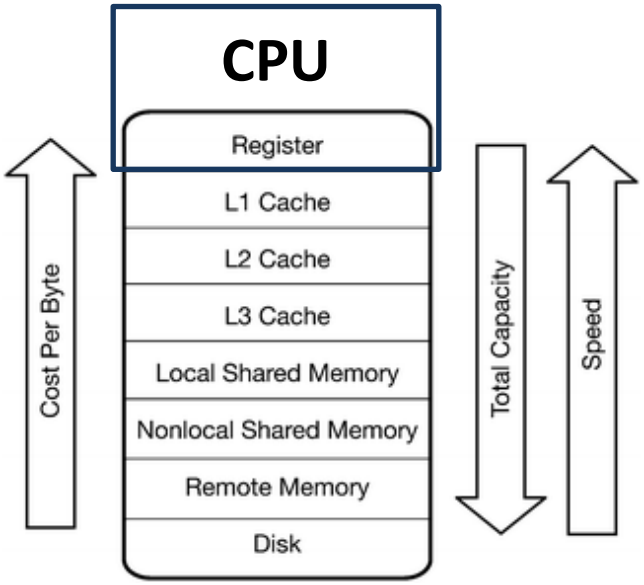


Memory hierarchies :

⇒ improve the average latency

⇒ Spatial and timing locality

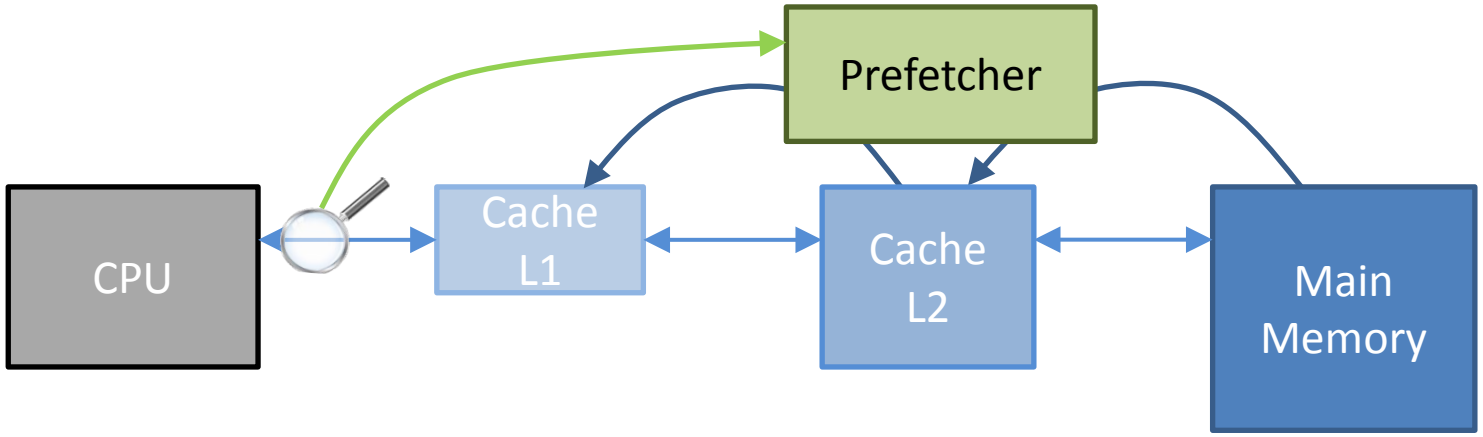
⇒ But not optimal : prefetch

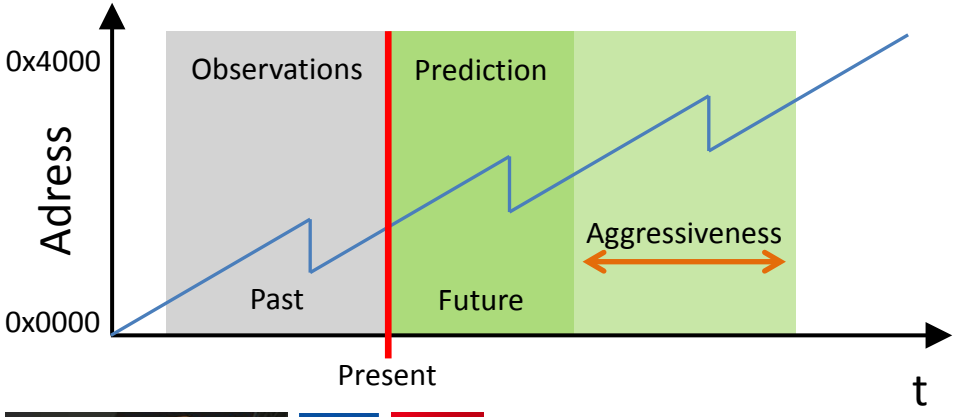


Data Prefetching

Data prefetch : improve memory hierarchies performance

⇒ Predict the future memory accesses to minimize the cache misses





Metrics :

- **Timely** : Timeliness
- **Useful** : Accuracy (% of useful prefetches),
  Coverage (% of misses removed)
- **Low overhead** : Pollution

Existing solutions

- Software
  - Manual hints in code : **tedious**
  - Automatic insertion of instruction at compilation time : **weak efficiency**
- Hardware
  - Sequential prefetching (80's) : weak efficiency → **no pattern**
  - Stride prefetching (90's) : data and application dependent → **no adaptivity**
  - Adaptive prefetching (2000's) : basic and arbitrary rules → **limited by the original prefetcher capabilities**
- Hybrid
  - Combination of existing software and hardware solutions

```
For i=0 to n do
    Prefetch a[i];
    Prefetch b[i];
    c[i]=a[i]+b[i];
end
```

Existing solutions

- Software
  - Manual hints in code : **tedious**
  - Automatic insertion of instruction at compilation time : **weak efficiency**
- Hardware
  - Sequential prefetching (80's) : weak efficiency → **no pattern**
  - Stride prefetching (90's) : data and application dependent → **no adaptivity**
  - Adaptive prefetching (2000's) : basic and arbitrary rules → **limited by the original prefetcher capabilities**
- Hybrid
  - Combination of existing software and hardware solutions

```
lw    $t0, 0($gp)
bltz  $t0, def
slti  $t1, $t0, 3
beq   $t1, $zero, def
sll   $t0, $t0, 2
add   $t2, $t0, $gp
lw    $t2, 1064($t2)
```

Existing solutions

- Software
  - Manual hints in code : **tedious**
  - Automatic insertion of instruction at compilation time : **weak efficiency**
- Hardware
  - Sequential prefetching (80's) : weak efficiency → **no pattern**
  - Stride prefetching (90's) : data and application dependent → **no adaptivity**
  - Adaptive prefetching (2000's) : basic and arbitrary rules → **limited by the original prefetcher capabilities**
- Hybrid
  - Combination of existing software and hardware solutions

0x..00	0x..04	0x..08	0x..12
0x..16	0x..20	0x..24	0x..28
0x..32	0x..36	0x..40	0x..44
0x..48	0x..52	0x..56	0x..60
0x..64	0x..68	0x..72	0x..76

Existing solutions

- Software
  - Manual hints in code : **tedious**
  - Automatic insertion of instruction at compilation time : **weak efficiency**
- Hardware
  - Sequential prefetching (80's) : weak efficiency → **no pattern**
  - Stride prefetching (90's) : data and application dependent → **no adaptivity**
  - Adaptive prefetching (2000's) : basic and arbitrary rules → **limited by the original prefetcher capabilities**
- Hybrid
  - Combination of existing software and hardware solutions

PC	Address
…	…
0x..100	0x..04
0x..104	0x..48
0x..108	0x..64
0x..100	0x..36
0x..104	0x..52
0x..108	0x..48
0x..100	0x..68
0x..104	0x..56
0x..108	0x..32
…	…

0x..00	0x..04	0x..08	0x..12
0x..16	0x..20	0x..24	0x..28
0x..32	0x..36	0x..40	0x..44
0x..48	0x..52	0x..56	0x..60
0x..64	0x..68	0x..72	0x..76

Existing solutions

- Software
  - Manual hints in code : **tedious**
  - Automatic insertion of instruction at compilation time : **weak efficiency**
- Hardware
  - Sequential prefetching (80's) : weak efficiency → **no pattern**
  - Stride prefetching (90's) : data and application dependent → **no adaptivity**
  - Adaptive prefetching (2000's) : basic and arbitrary rules → **limited by the original prefetcher capabilities**
- Hybrid
  - Combination of existing software and hardware solutions

Objectives

- Develop data prefetching mechanisms natively adaptive to data and application variations
  - Evaluate the performance improvement margin beyond the Stride Prefetcher

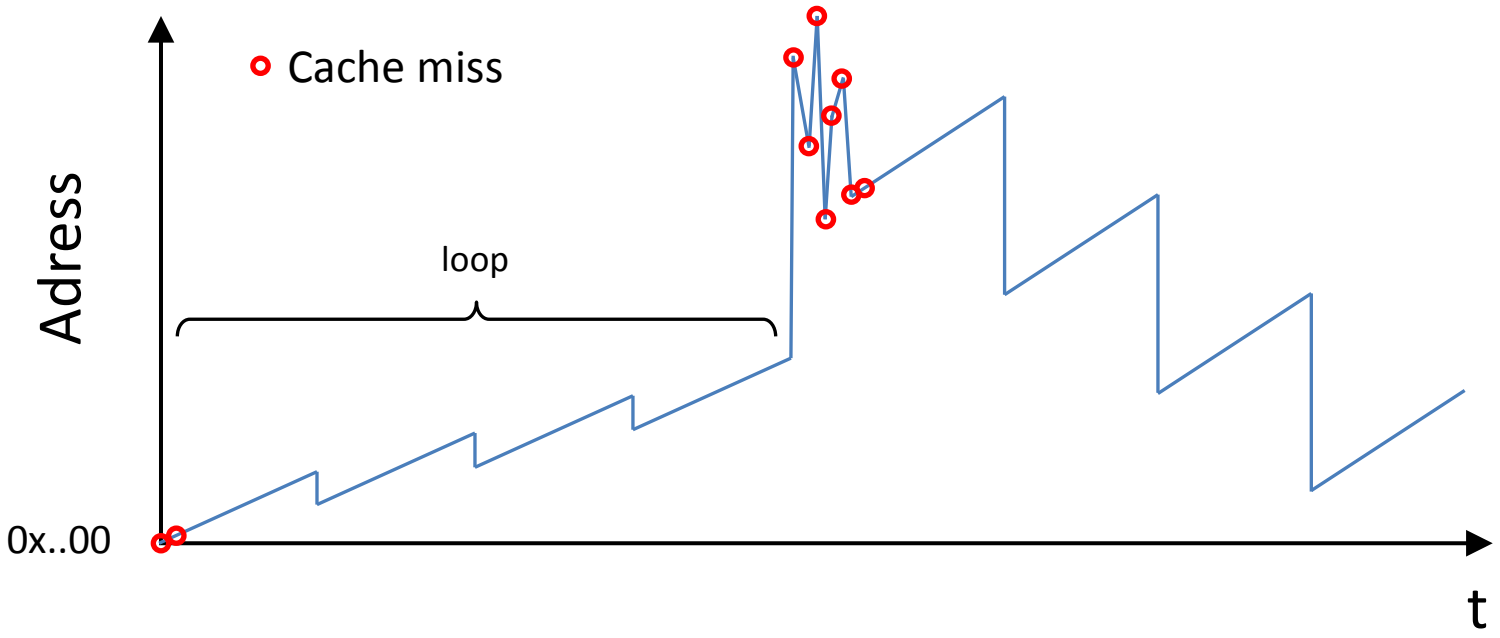
    ⇒Is it useful to develop alternative prefetcher mechanisms ?
  - Design a prefetcher taking part of Control theory and Dynamic Compilation capabilities

    ⇒ On going work

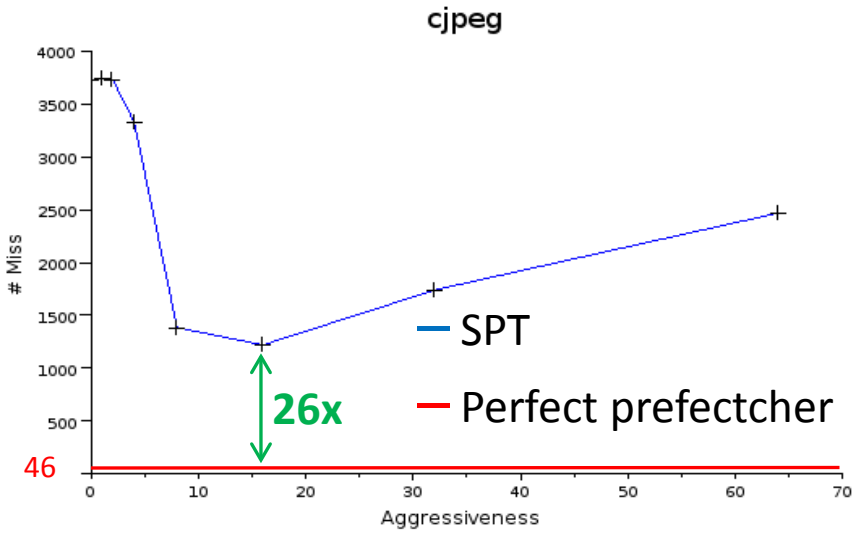
Perfect prefetcher

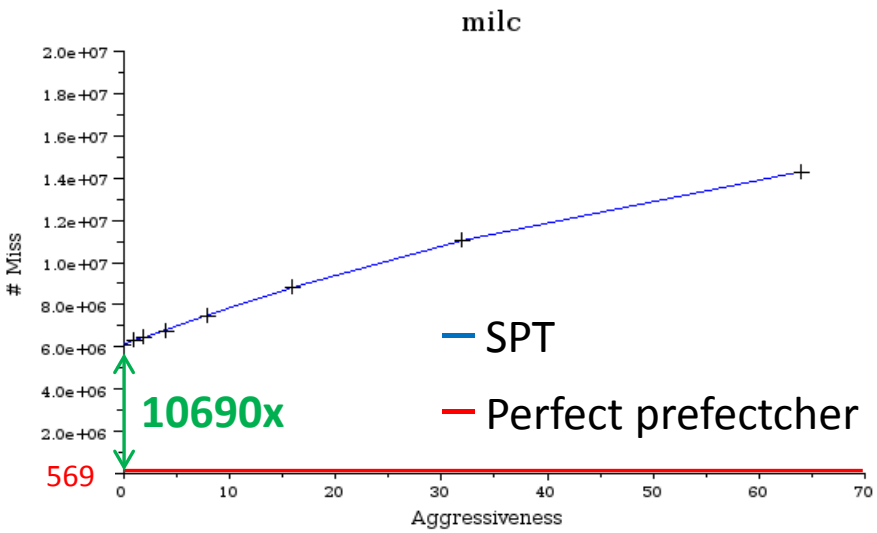
- Prediction is only possible in loops

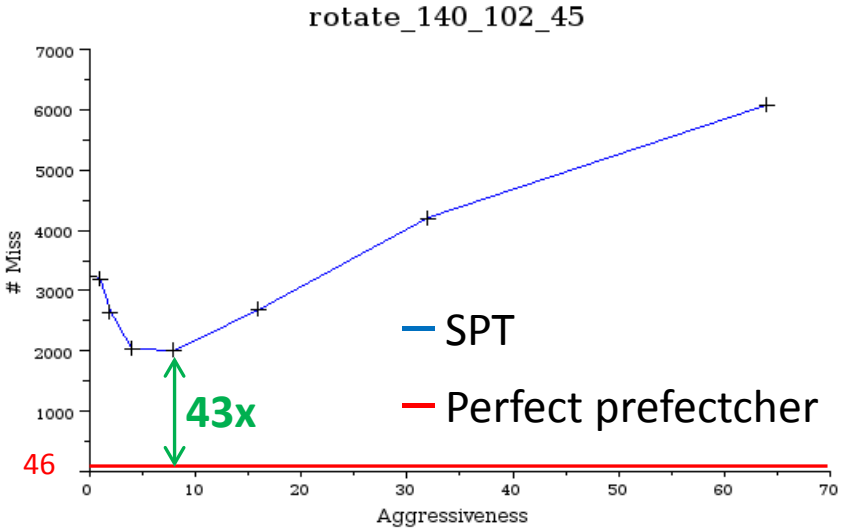
  ⇒Perfect prefetcher predicts all accesses in loops excepted the 2 first ones and extraordinary ones

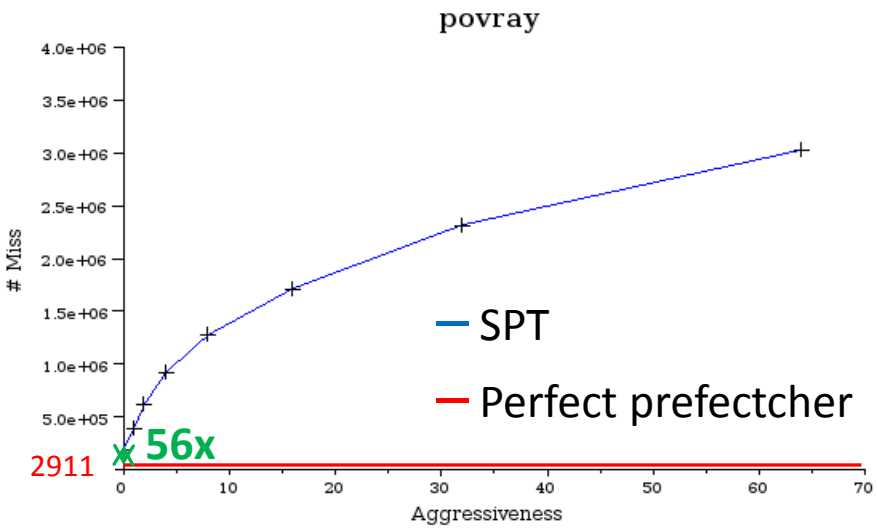


Results


cjpeg
Miss
Aggressiveness
SPT
Perfect prefectcher
26x
46


milc
Miss
Aggressiveness
SPT
Perfect prefectcher
10690x
569


rotate_140_102_45
Miss
Aggressiveness
SPT
Perfect prefectcher
43x
46


povray
Miss
Aggressiveness
SPT
Perfect prefectcher
56x
2911

Adaptive prefetcher

- Loops detection :
  - **PC patterns**
  - Instruction opcodes
  - Hints from dynamic compilation

  ⇒Nest loop : **Inner loops** are easier to detect and represent the major part of improvement

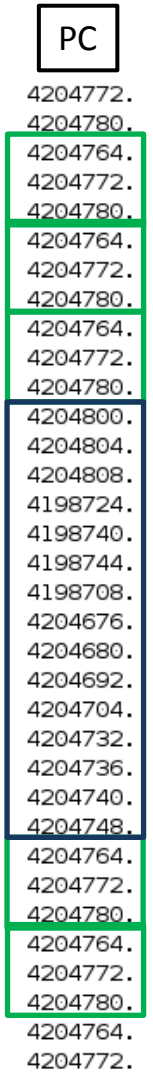
- Estimation of memory access patterns for each PC
  - Stride
  - **Polynomial**

Adaptive prefetcher

- Loops detection :
  - **PC patterns**
  - Instruction opcodes
  - Hints from dynamic compilation

  ⇒Nest loop : **Inner loops** are easier to detect and represent the major part of improvement

- Estimation of memory access patterns for each PC
  - Stride
  - **Polynomial**

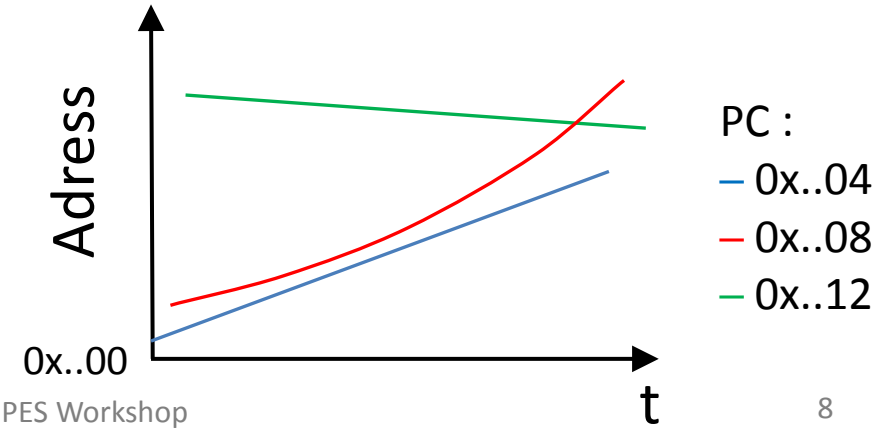


Adaptive prefetcher

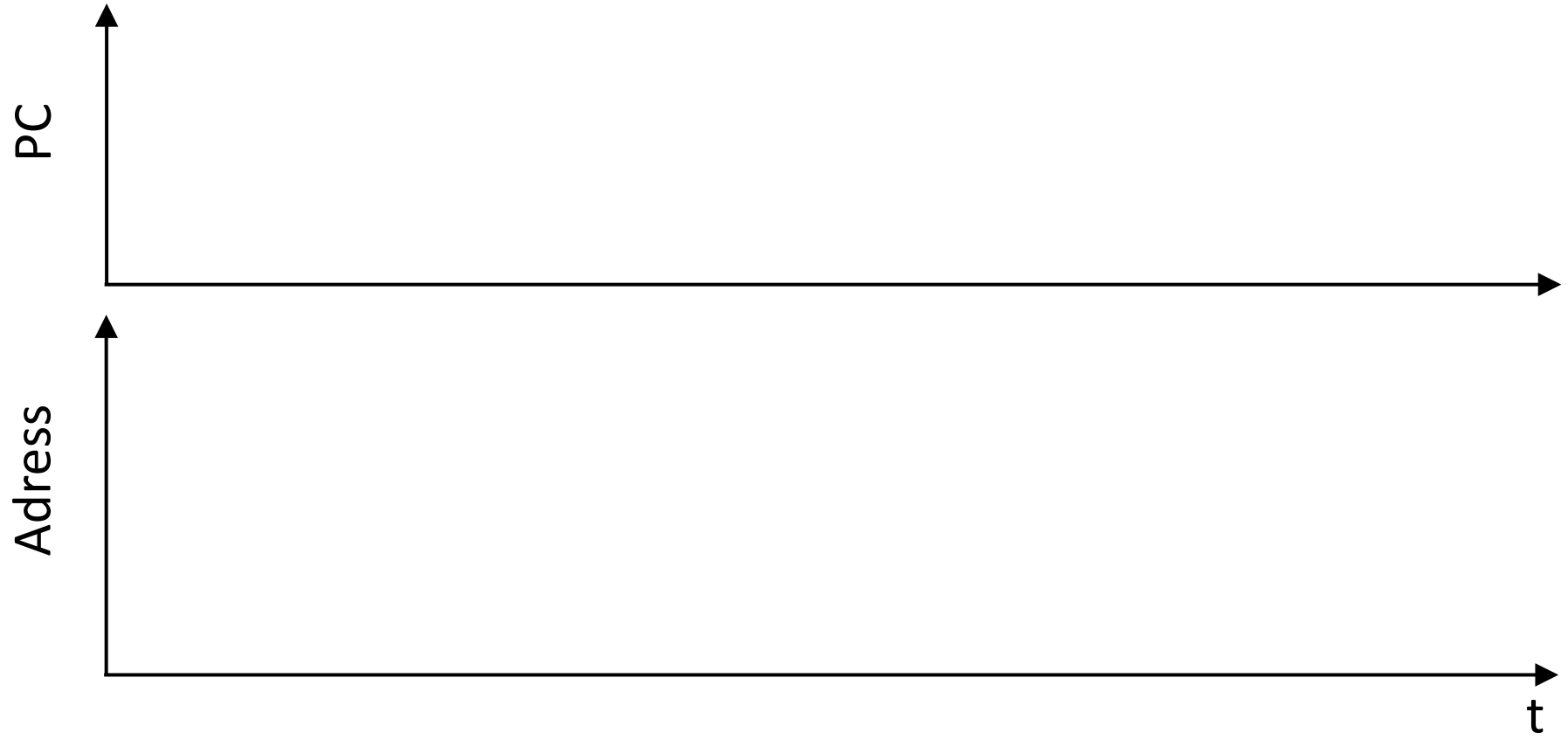
- Loops detection :
  - **PC patterns**
  - Instruction opcodes
  - Hints from dynamic compilation

  ⇒Nest loop : **Inner loops** are easier to detect and represent the major part of improvement

- Estimation of memory access patterns for each PC
  - Stride
  - **Polynomial**

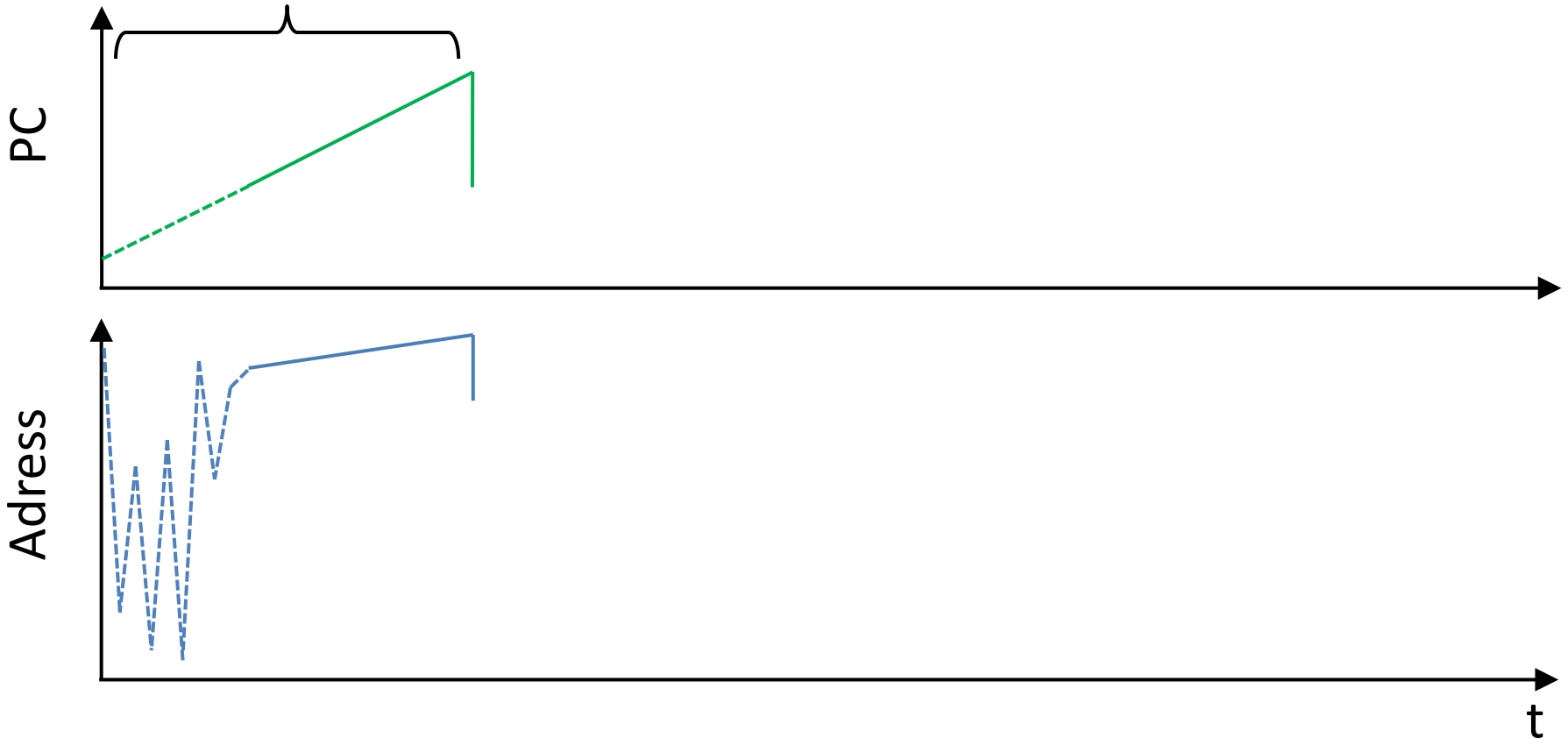


Adaptive prefetcher with inner loop detection



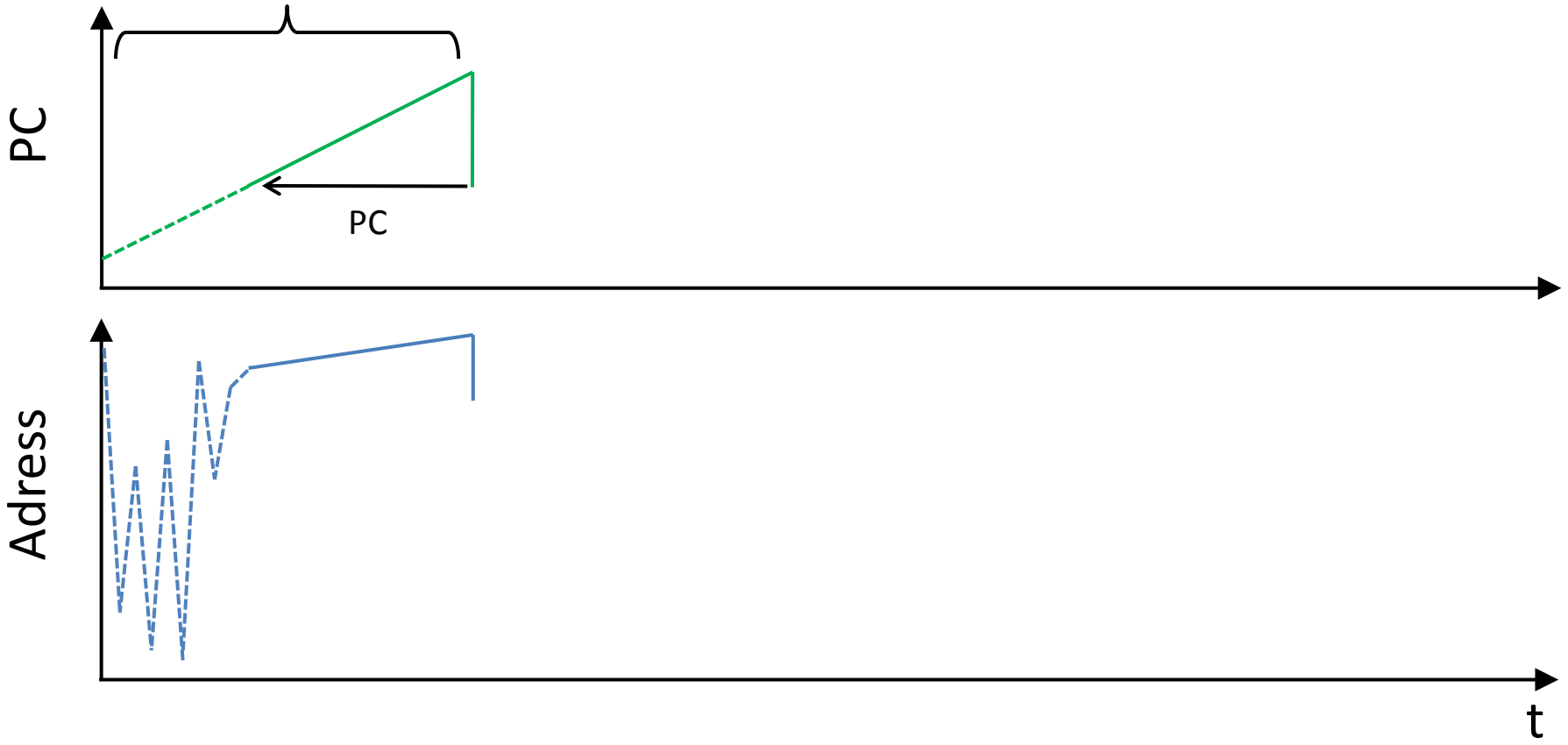
Adaptive prefetcher with inner loop detection

Before and during 1st iteration :
PCs and Addresses are stored each in a dedicated buffer



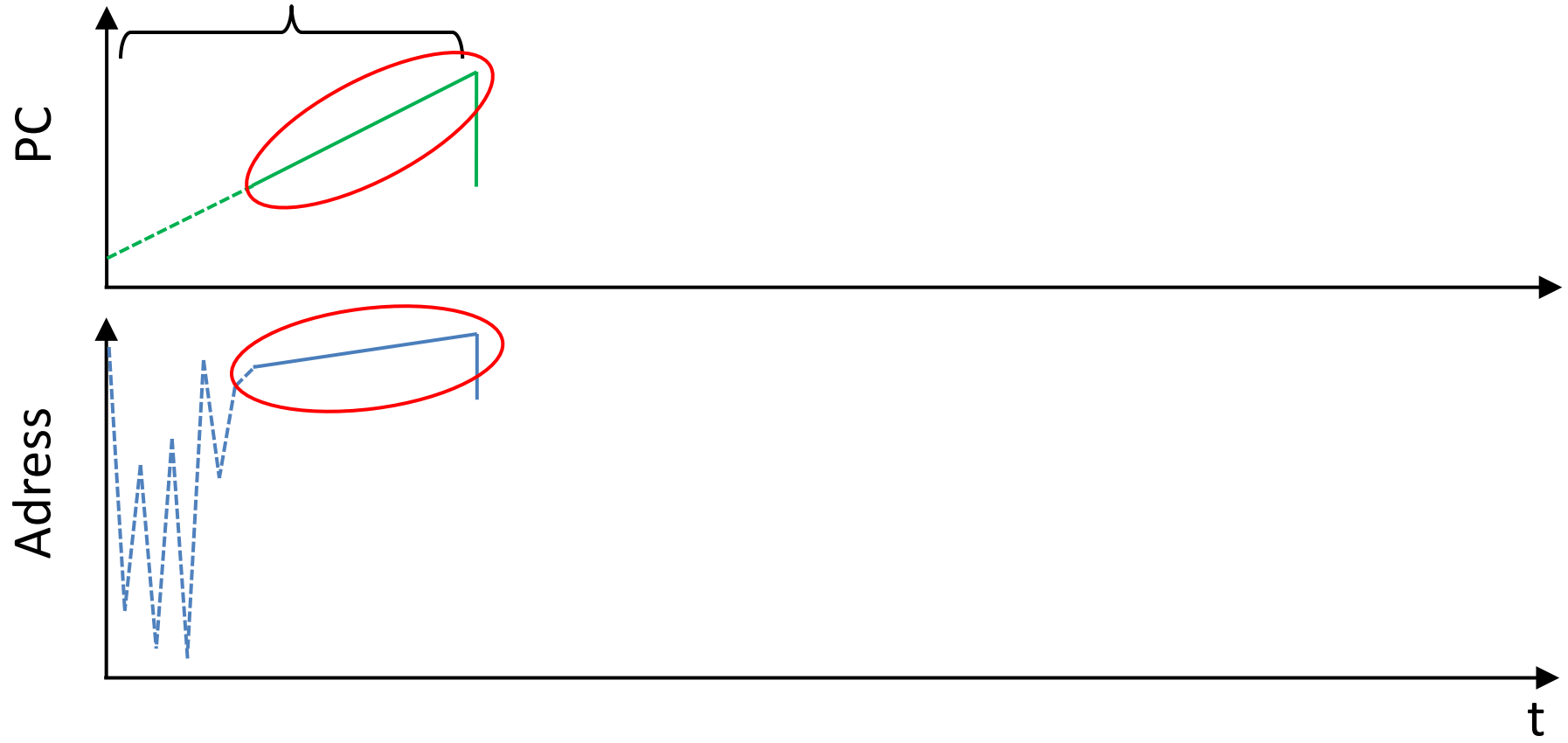
Adaptive prefetcher with inner loop detection

Before and during 1st iteration :
PCs and Addresses are stored each in a dedicated buffer



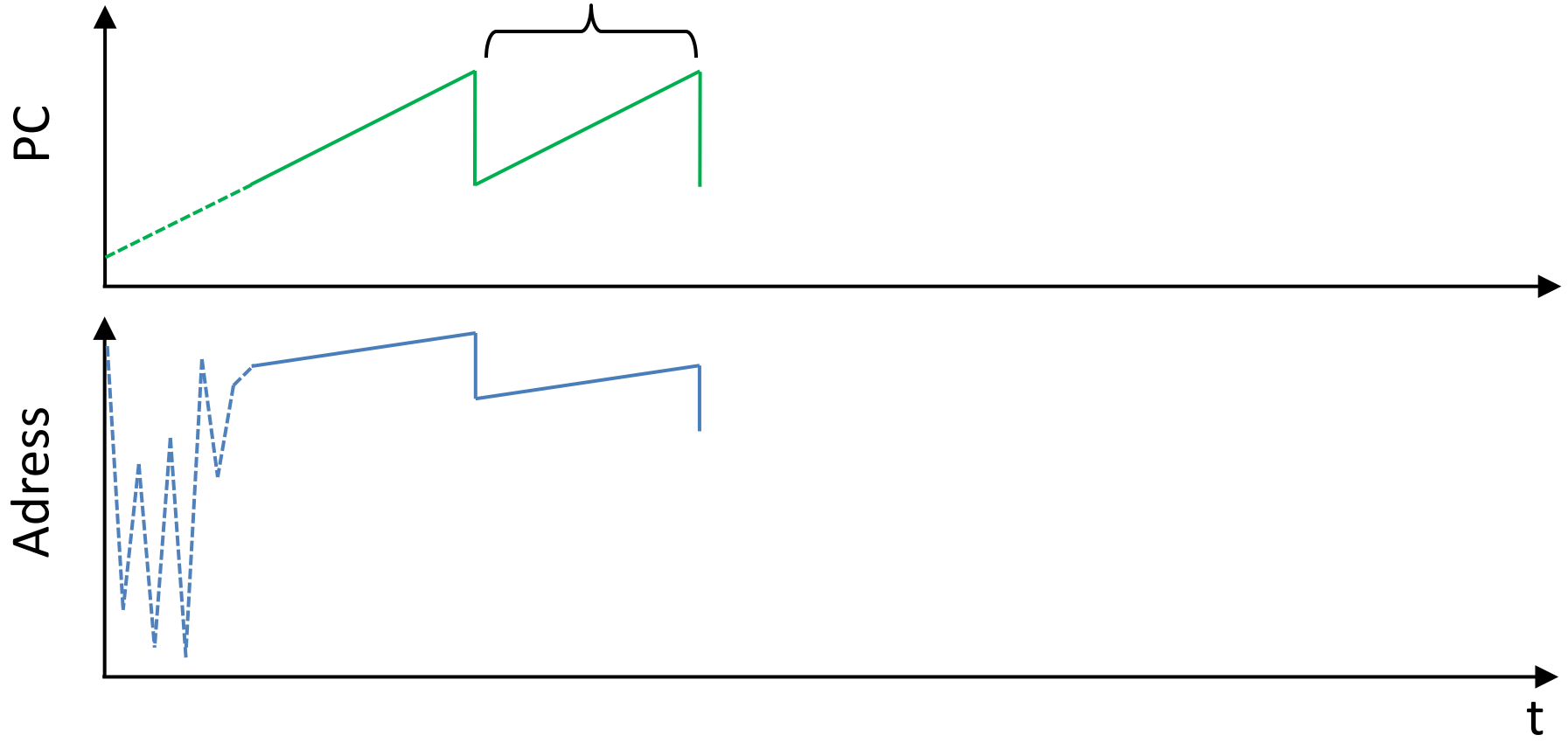
Adaptive prefetcher with inner loop detection

Before and during 1st iteration :
PCs and Addresses are stored each in a dedicated buffer



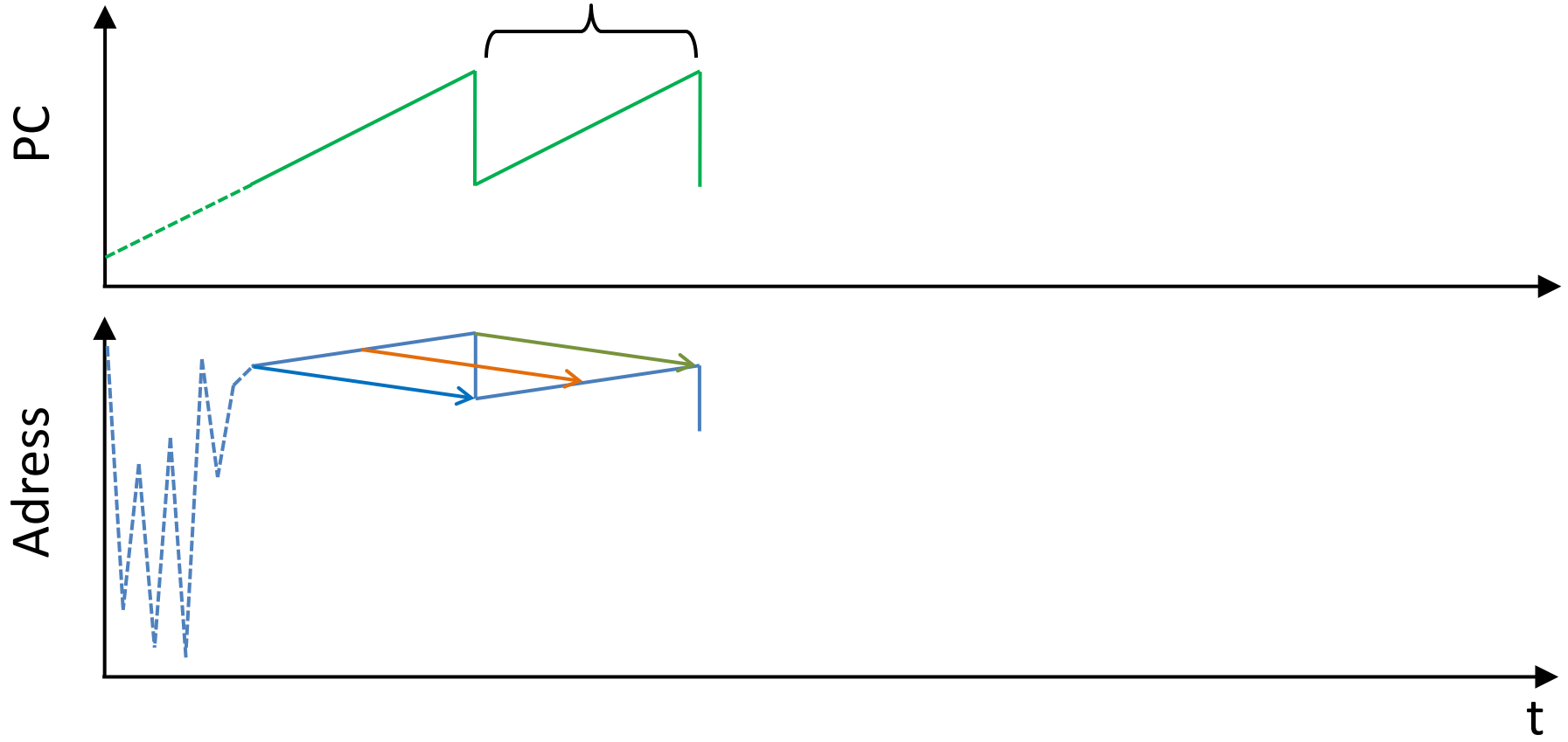
Adaptive prefetcher with inner loop detection

2nd iteration :
PC Pattern detected
⇒ Address Patterns computed
(Linear)



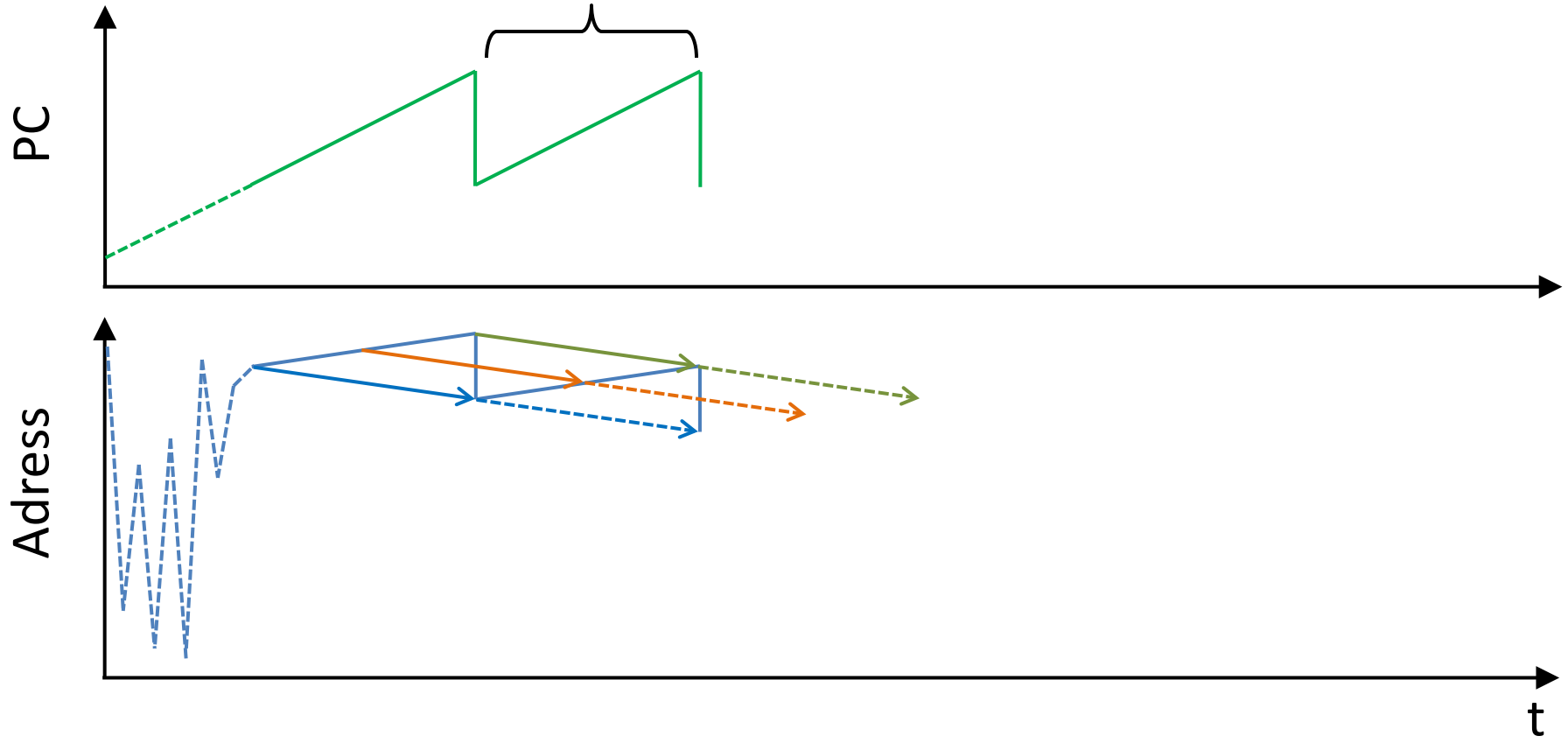
Adaptive prefetcher with inner loop detection

2nd iteration :
PC Pattern detected
⇒ Address Patterns computed
(Linear)



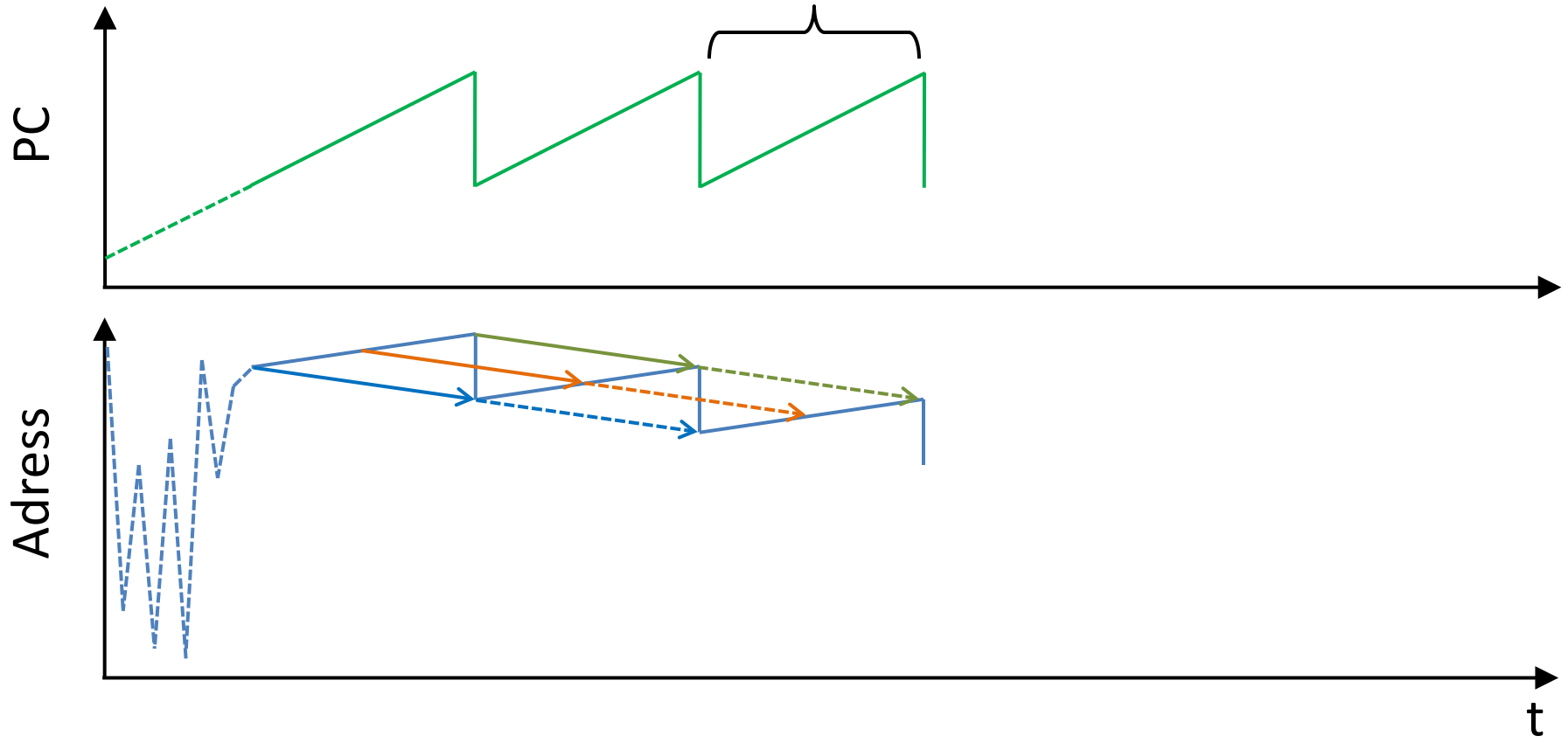
Adaptive prefetcher with inner loop detection

2nd iteration :
PC Pattern detected
⇒ Address Patterns computed
(Linear)



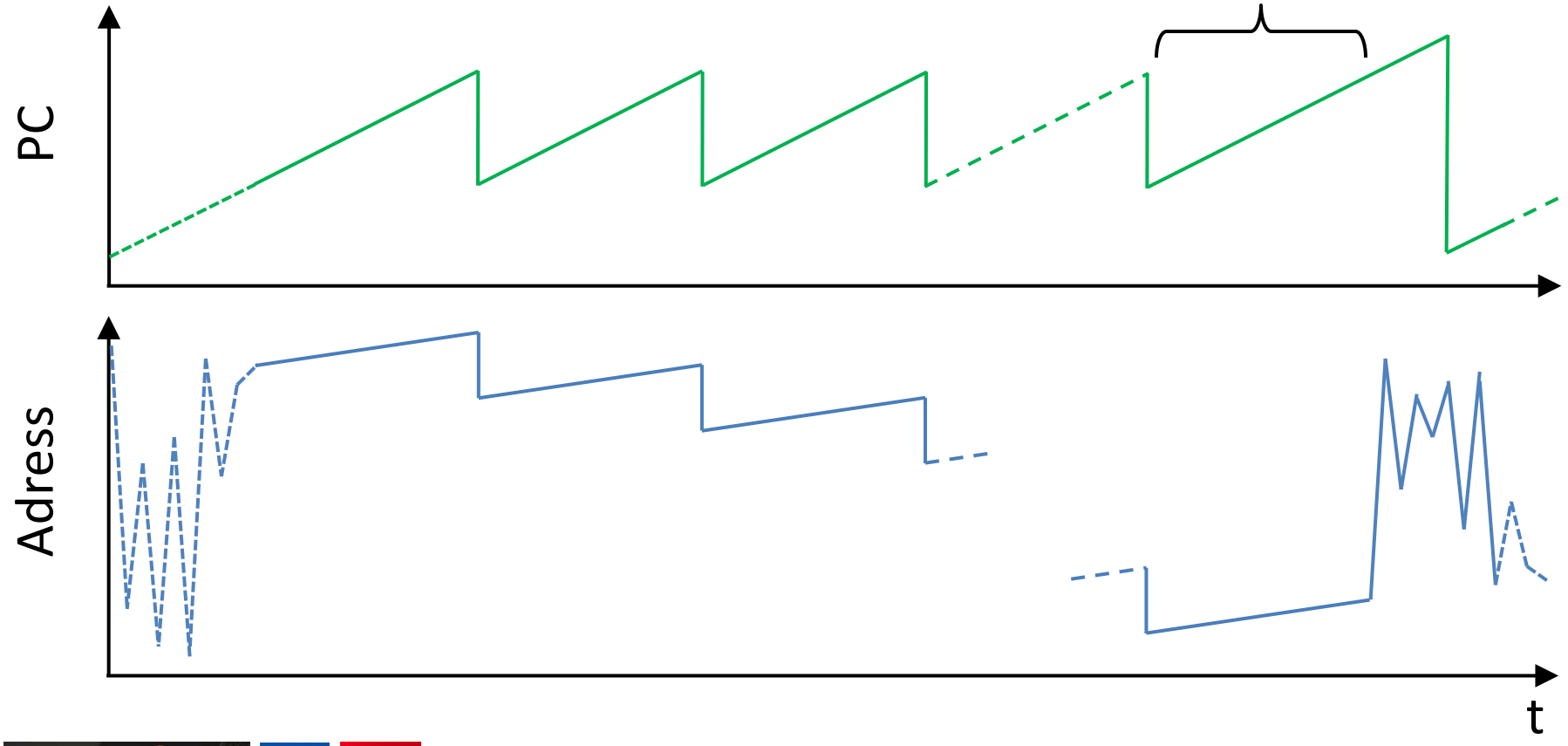
Adaptive prefetcher with inner loop detection

3rd iteration :
PC Pattern confirmed
⇒ Polynomial Address Patterns
(No more than 2nd order)

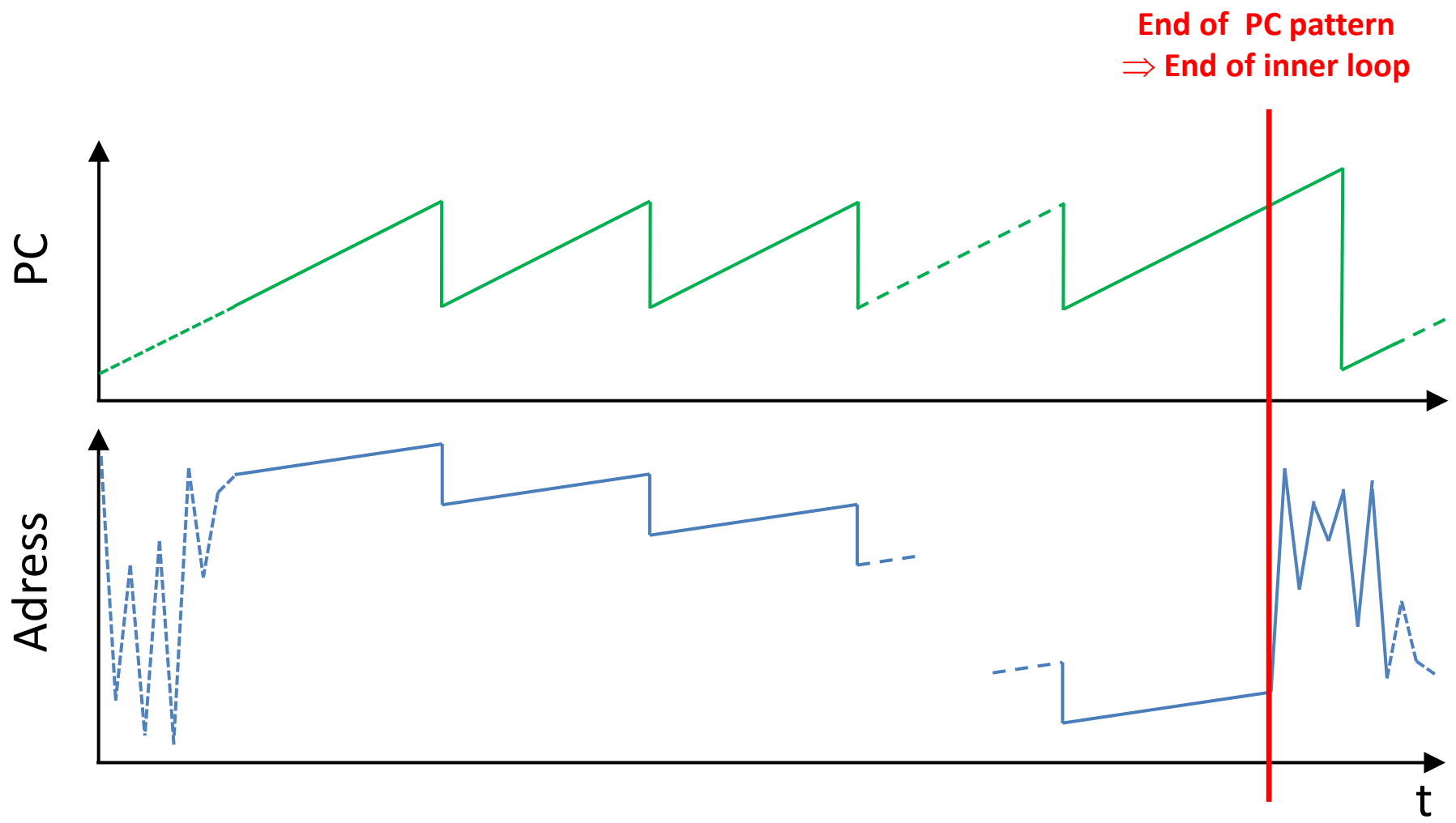


Adaptive prefetcher with inner loop detection

n^{th} iteration :
PC Pattern confirmed
⇒ Polynomial Address Patterns
(No more than $(n-1)^{nd}$ order)

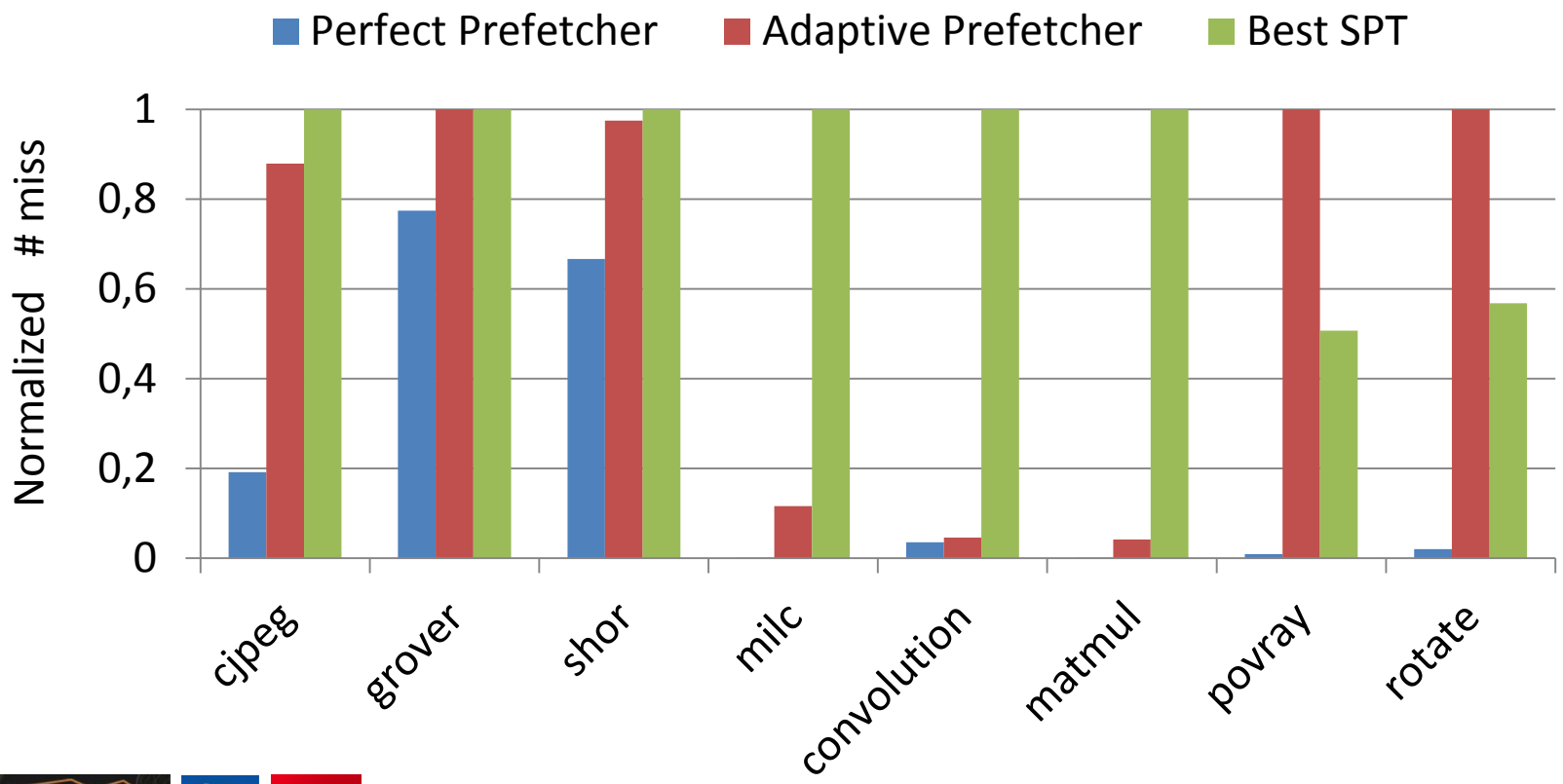


Adaptive prefetcher with inner loop detection



Results

- Framework:
  - RISC Processor
  - L1 cache : 16kB, 16b word size, Fully Associative



Conclusion and Future Works

Conclusion :

- Comparison performance Stride/Perfect prefetcher

  ⇒ Substantial improvement margin

- Adaptive prefetcher based on loop detection

  ⇒ Promising results

Perspectives :

- Improve loop detection with use of opcodes and “compilette”

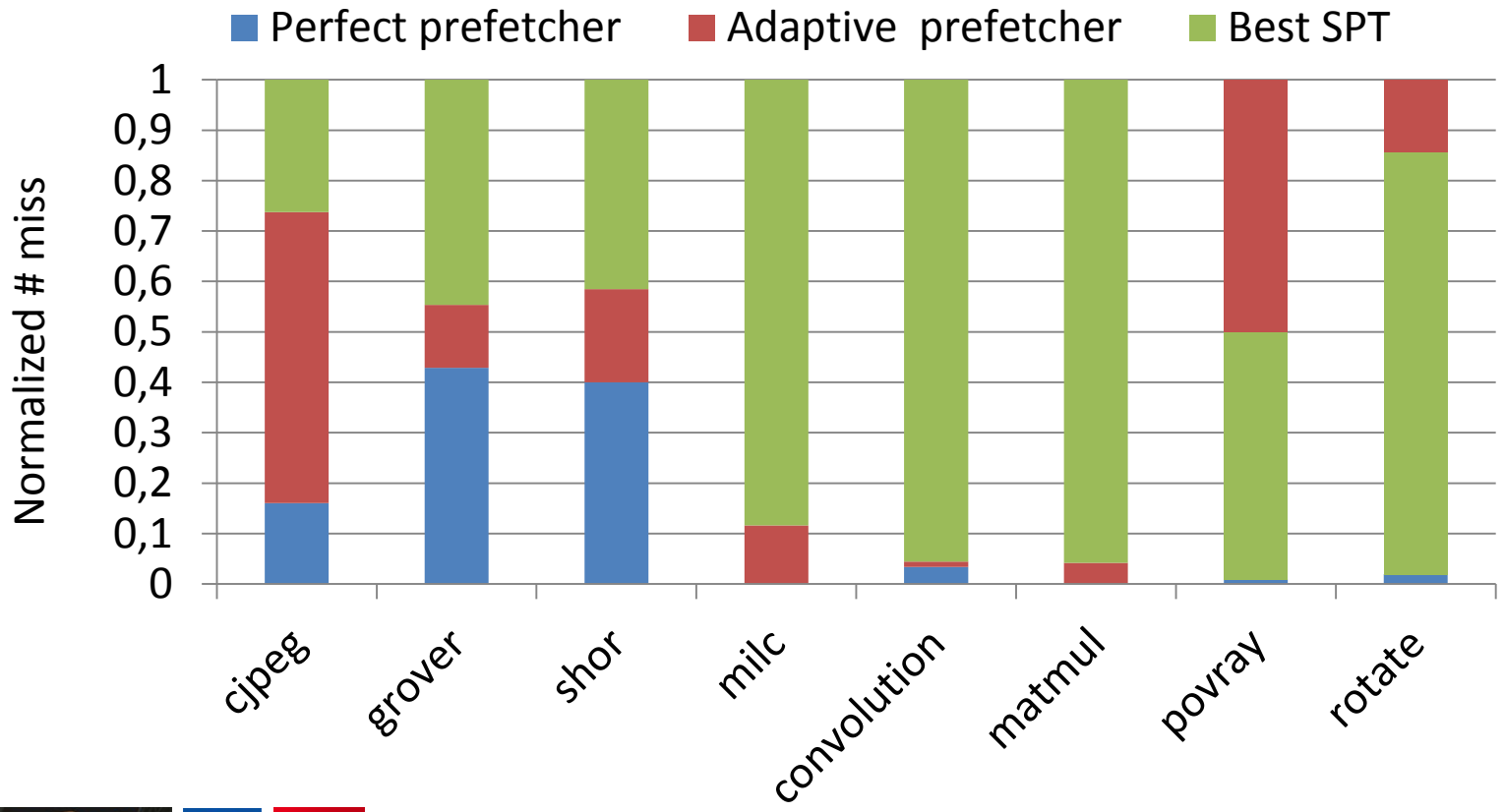
- Access pattern identification using more complex models

Thank you

for your attention

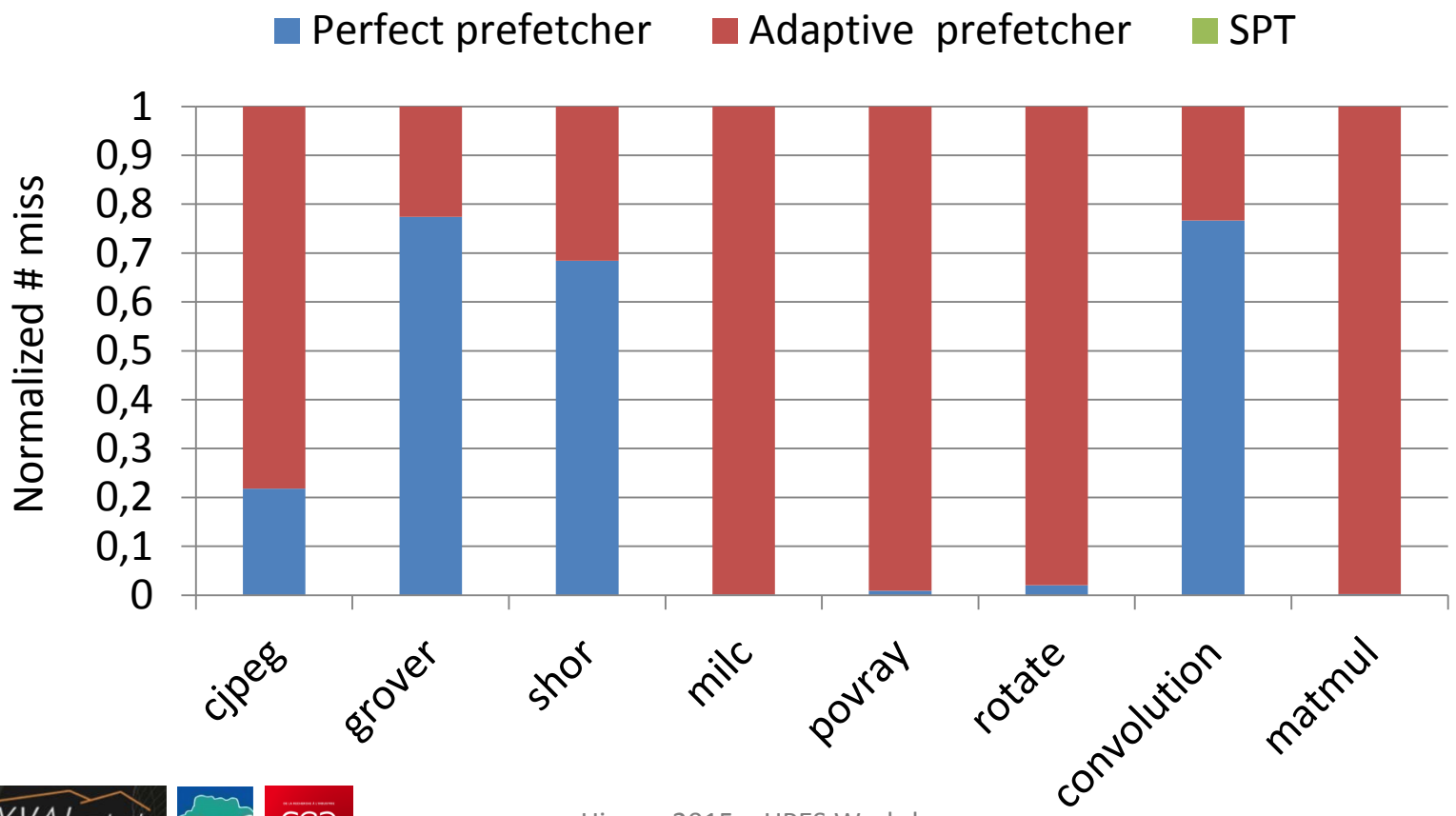
Results

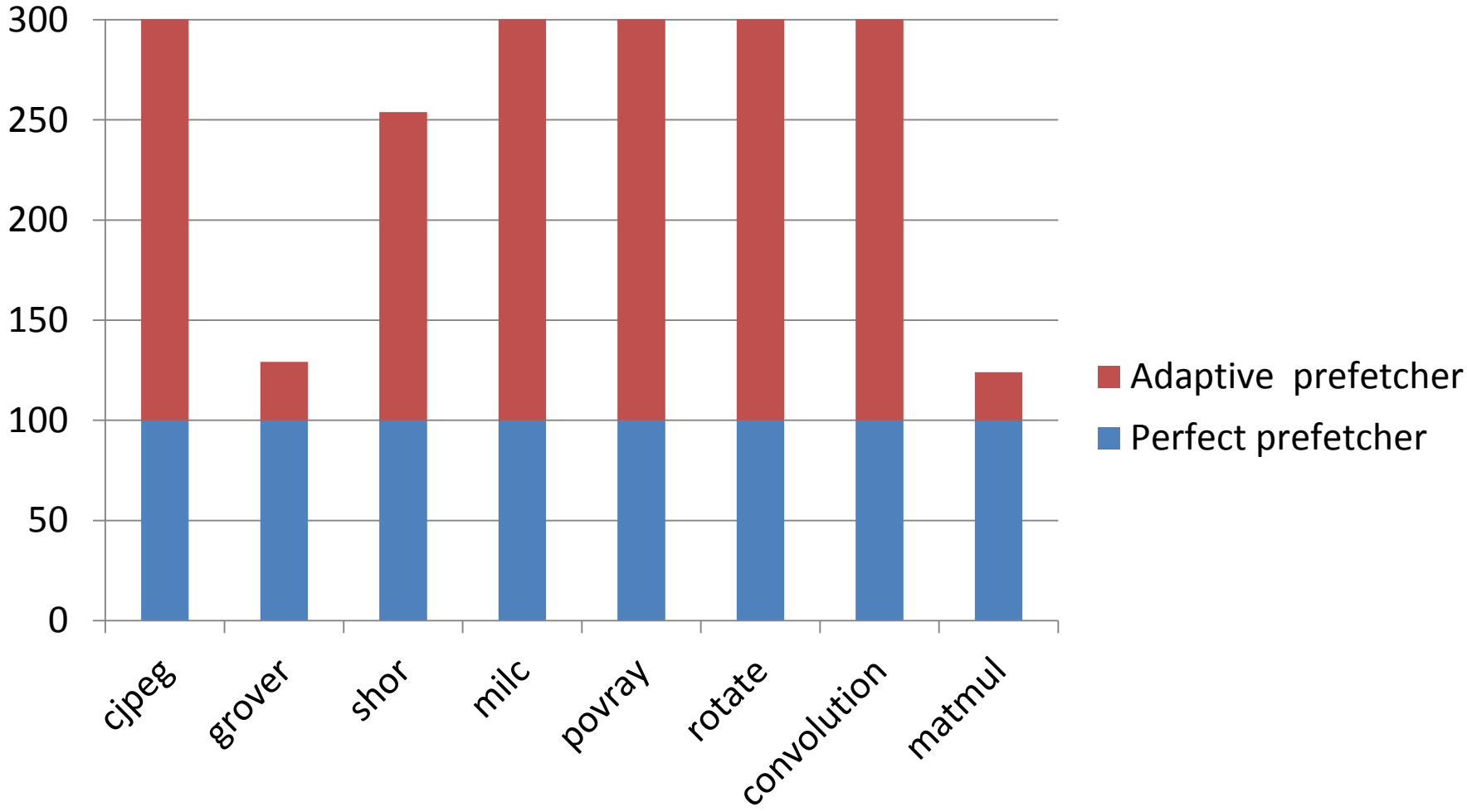
- Framework:
  - RISC Processor
  - L1 cache : 16kB, 16b word size, Fully Associative



Results

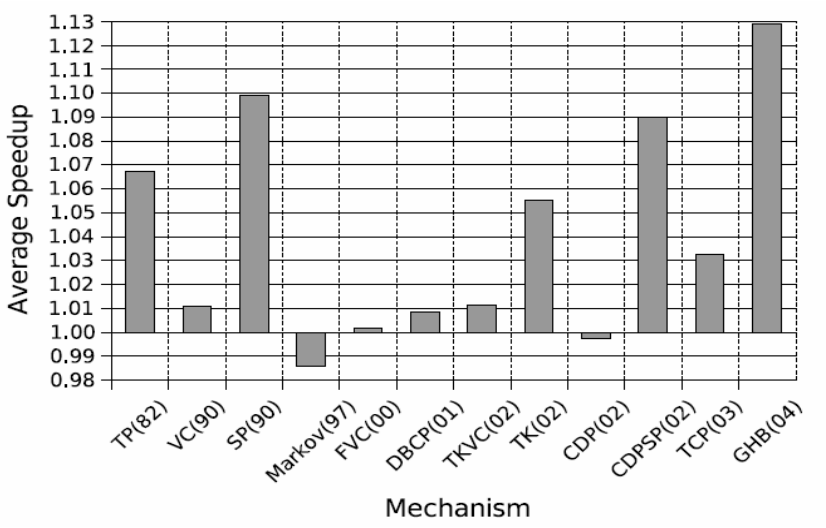
- Framework:
  - RISC Processor
  - L1 cache : 16kB, 16b word size, Fully Associative

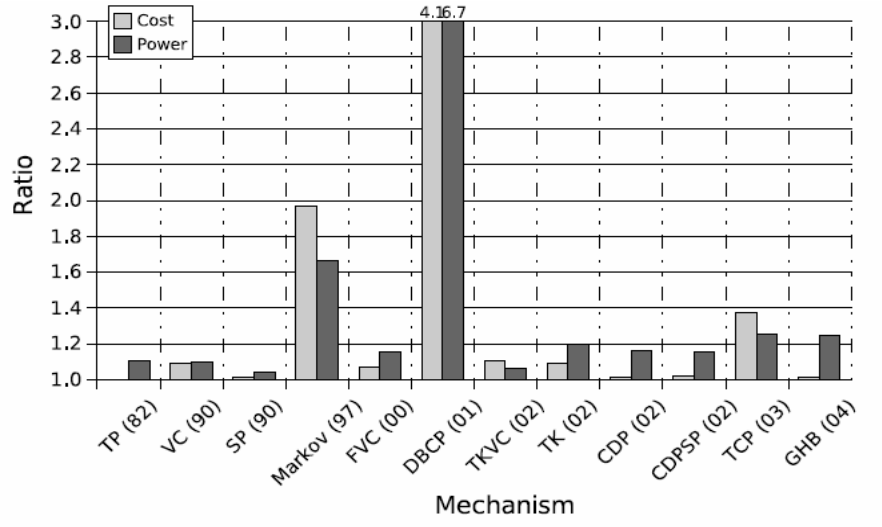



300
250
200
150
100
50
0
cjpeg
grover
shor
milc
povray
rotate
convolution
matmul
Adaptive prefetcher
Perfect prefetcher

Performance Comparison

- Garcia Pérez 2004 :
  - Retro-engineering of several prefetchers
  - Uniform framework of evaluation (benchmark, model, trace)





⇒Today, no prefetcher is more efficient than the 25 years old Stride Prefetcher

First works

- Generation of access memory trace : QEMU
  - Pluggins tcg (Tiny Code Generator) : DineroIV (cache simulator)

l1-dcache

Metrics	Total	Instrn	Data	Read	Write	Misc
Demand Fetches	51	0	51	38	13	0
Fraction of total	1.0000	0.0000	1.0000	0.7451	0.2549	0.0000
Demand Misses	2	0	2	1	1	0
Demand miss rate	0.0392	0.0000	0.0392	0.0263	0.0769	0.0000

Multi-block refs	0
Bytes From Memory	256
(/ Demand Fetches)	5.0196
Bytes To Memory	128
(/ Demand Writes)	9.8462
Total Bytes r/w Mem	384
(/ Demand Fetches)	7.5294

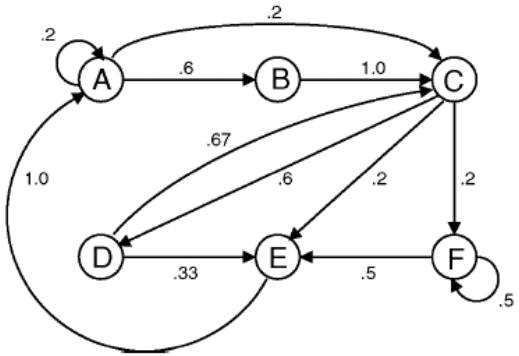
Summary:

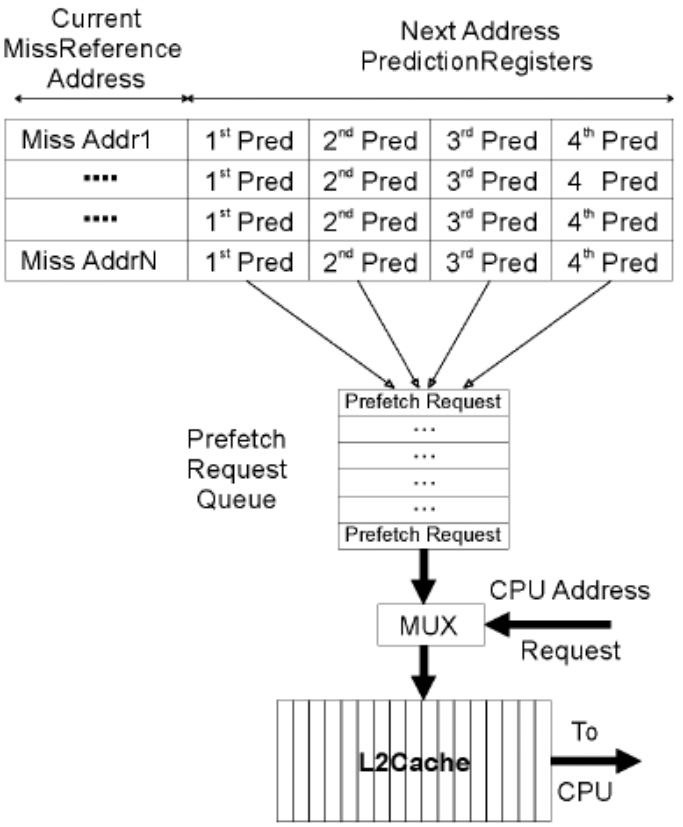
instruction fetch: 74866 access in cache level 1
instruction fetch: 14411 access in RAM
data read: 37 access in cache level 1
data read: 1 access in RAM
data write: 12 access in cache level 1
data write: 1 access in RAM

```
...
i 0x000000000000844c 0x00000000 (0x0000000000000000) CPU #0 0x000000000000844c
i 0x0000000000008450 0x00000000 (0x0000000000000000) CPU #0 0x0000000000008450
w 0x00000000f6fff7b4 0x00000004 (0x0000000000000000) CPU #0 0x0000000000008450
i 0x0000000000008454 0x00000000 (0x0000000000000000) CPU #0 0x0000000000008454
i 0x00000000000084bc 0x00000000 (0x0000000000000000) CPU #0 0x00000000000084bc
r 0x00000000f6fff7b4 0x00000004 (0x0000000000000000) CPU #0 0x00000000000084bc
i 0x00000000000084c0 0x00000000 (0x0000000000000000) CPU #0 0x00000000000084c0
i 0x00000000000084c4 0x00000000 (0x0000000000000000) CPU #0 0x00000000000084c4
i 0x0000000000008458 0x00000000 (0x0000000000000000) CPU #0 0x0000000000008458
r 0x00000000f6fff7b4 0x00000004 (0x0000000000000000) CPU #0 0x0000000000008458
i 0x000000000000845c 0x00000000 (0x0000000000000000) CPU #0 0x000000000000845c
i 0x0000000000008460 0x00000000 (0x0000000000000000) CPU #0 0x0000000000008460
i 0x0000000000008464 0x00000000 (0x0000000000000000) CPU #0 0x0000000000008464
...
```

Bibliographie

- Techniques
  - Hardware :
    - Stride/stream : principe de localité spatiale
    - Correlation-based prefetching : principe de localité temporelle
    - Joseph 1999 : *Prefetching using Markov predictor*
      - Très lourd à stocker : plusieurs MBytes





Bibliographie

- Techniques
  - Hardware :
    - Nesbit 2005 : *Data cache prefetching using a Global History Buffer*
      - Prefetch pour les niveaux supérieurs (L2, L3)
      - Longueur GHB ?

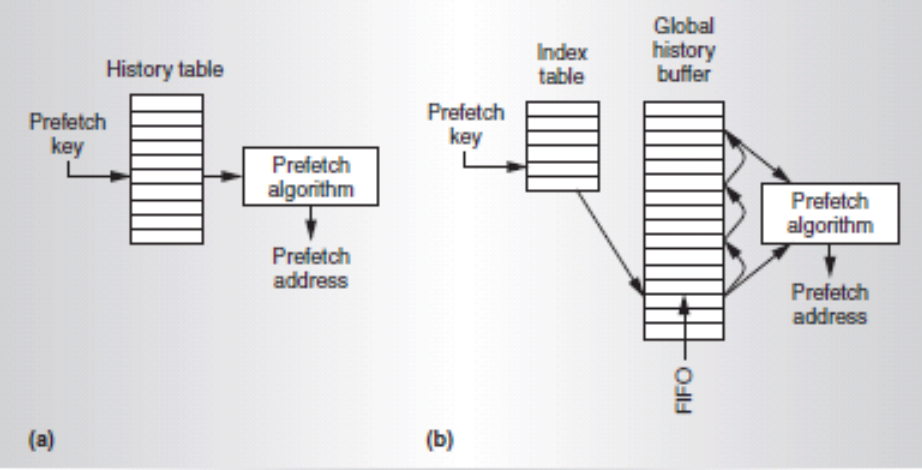


Figure 1. Table-based data cache prefetch methods: conventional prefetch table (a); global history buffer prefetch structure (b).

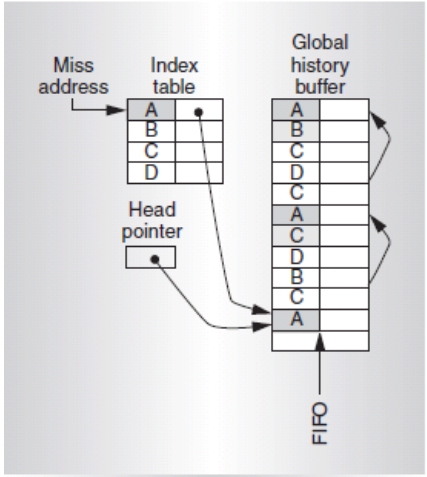


Figure 2. GHB global/address correlation (G/AC) prefetcher.