Best-Offset Hardware Prefetching

Pierre Michaud

March 2016

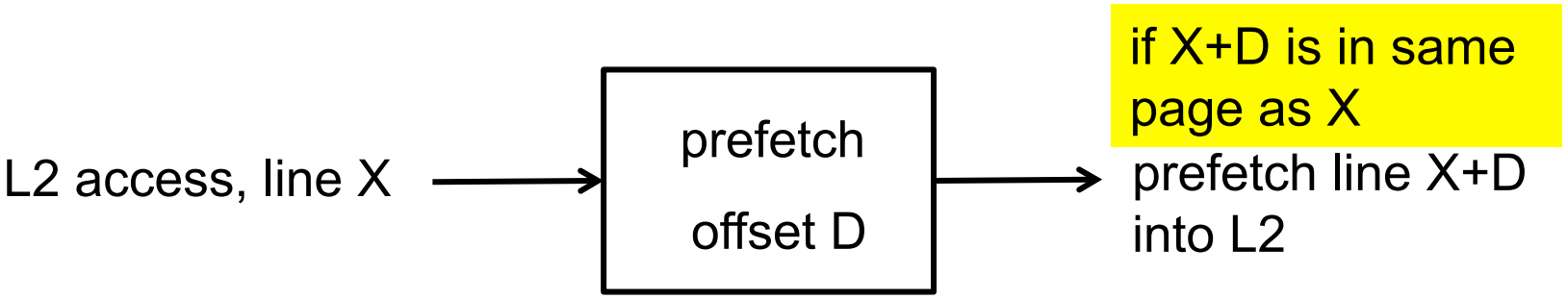
BOP: yet another data prefetcher

- Contribution: offset prefetcher with new mechanism for setting the prefetch offset dynamically
  - Improvement over Sandbox prefetcher (Pugsley et al., HPCA 2014)
- Good performance on the SPEC CPU benchmarks
  - tuned BOP won the 2015 Data Prefetching Championship
- Simple hardware

Offset prefetching (L2 cache)

next-line prefetching ➔ offset = 1

Offset prefetching with physical addresses

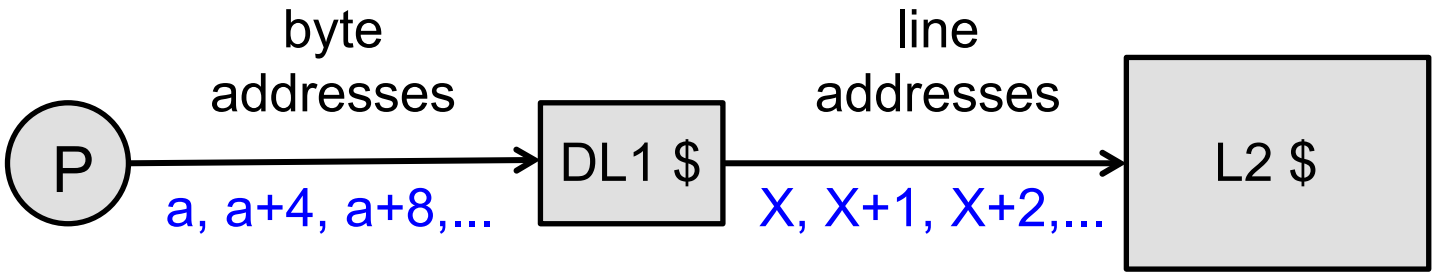


offset prefetching works better with large pages
(or with virtual addresses)

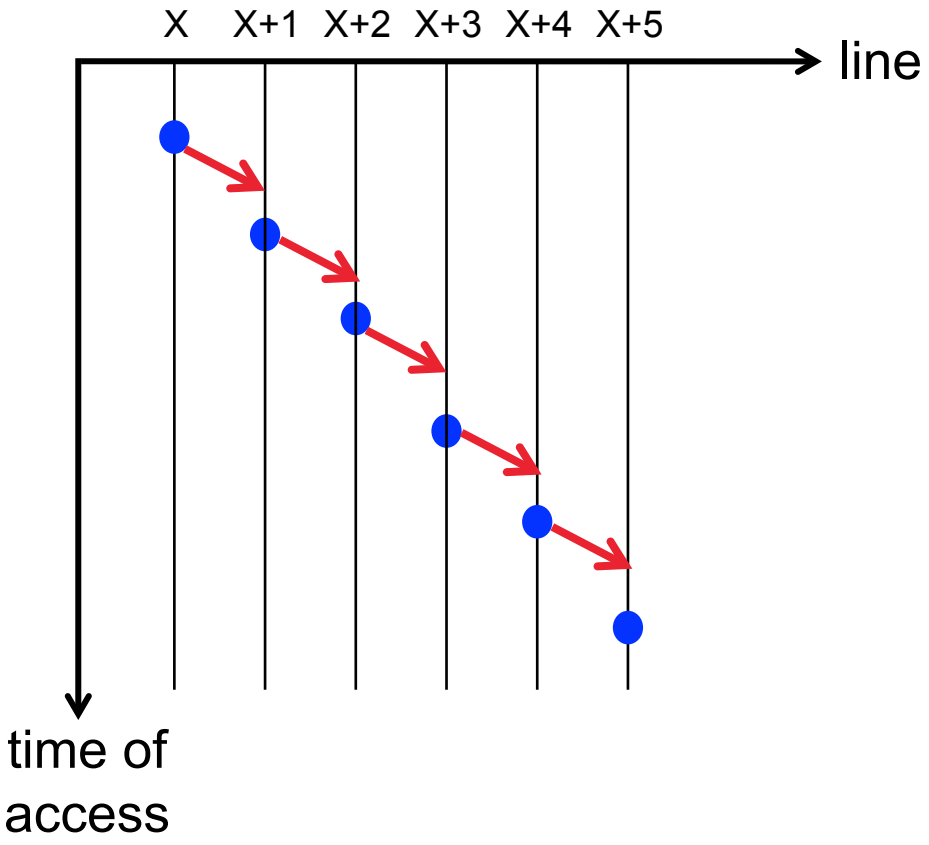
Offset prefetching is not new

- Not mainstream either (at least in academia)
  - Ki & Knowles, "Adaptive data prefetching using cache information", ICS 1997
  - Pugsley et al., "Sandbox prefetching: safe run-time evaluation of aggressive prefetchers", HPCA 2014
  - other ?
- Different from stream prefetching
  - does not try to detect streams
- Different from delta-correlation prefetching
  - delta-correlation predicts which line will be accessed next
  - offset prefetching predicts that a line will be accessed soon

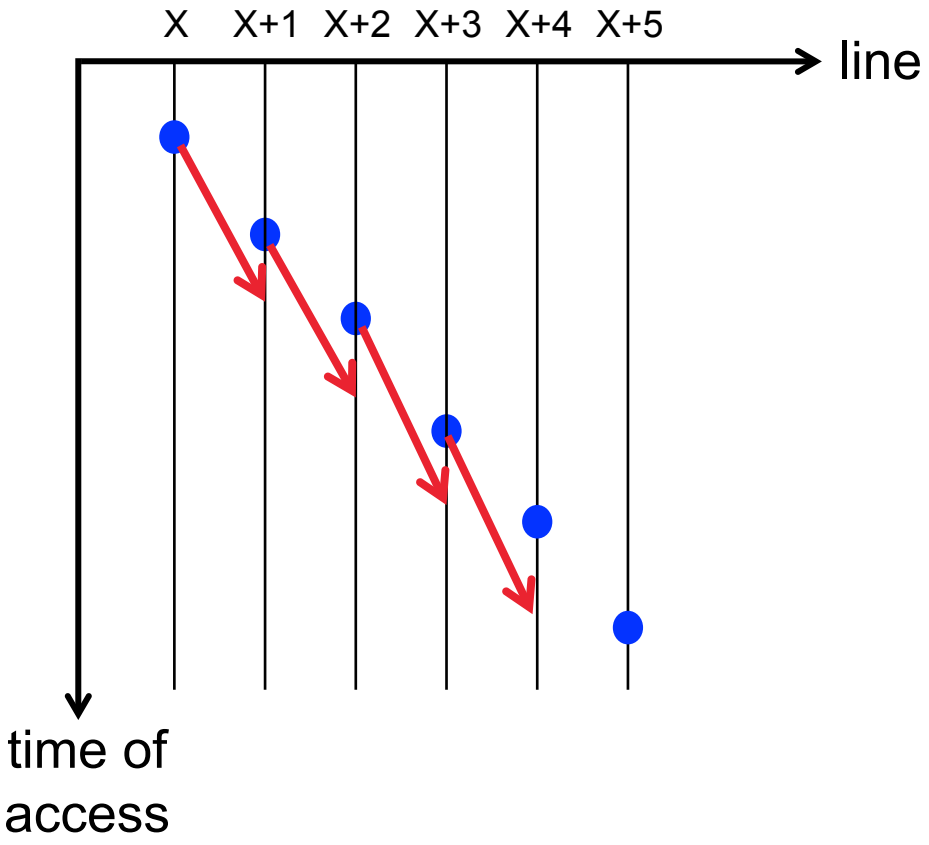
Sequential stream



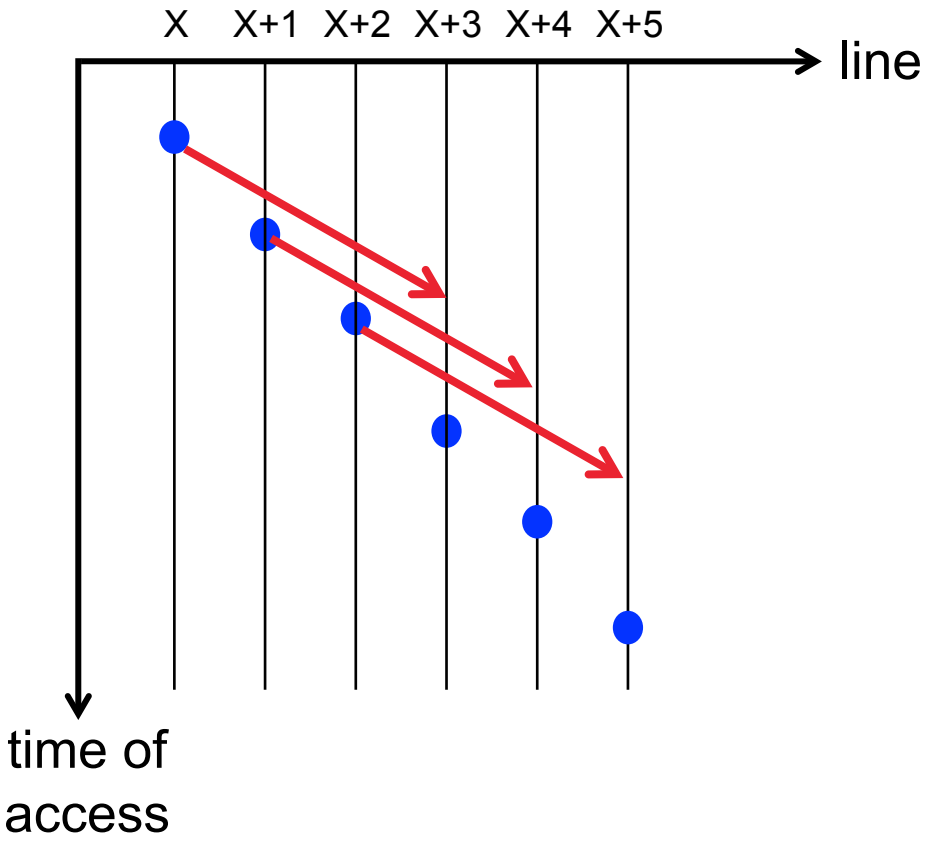
100% prefetch coverage with offset=1



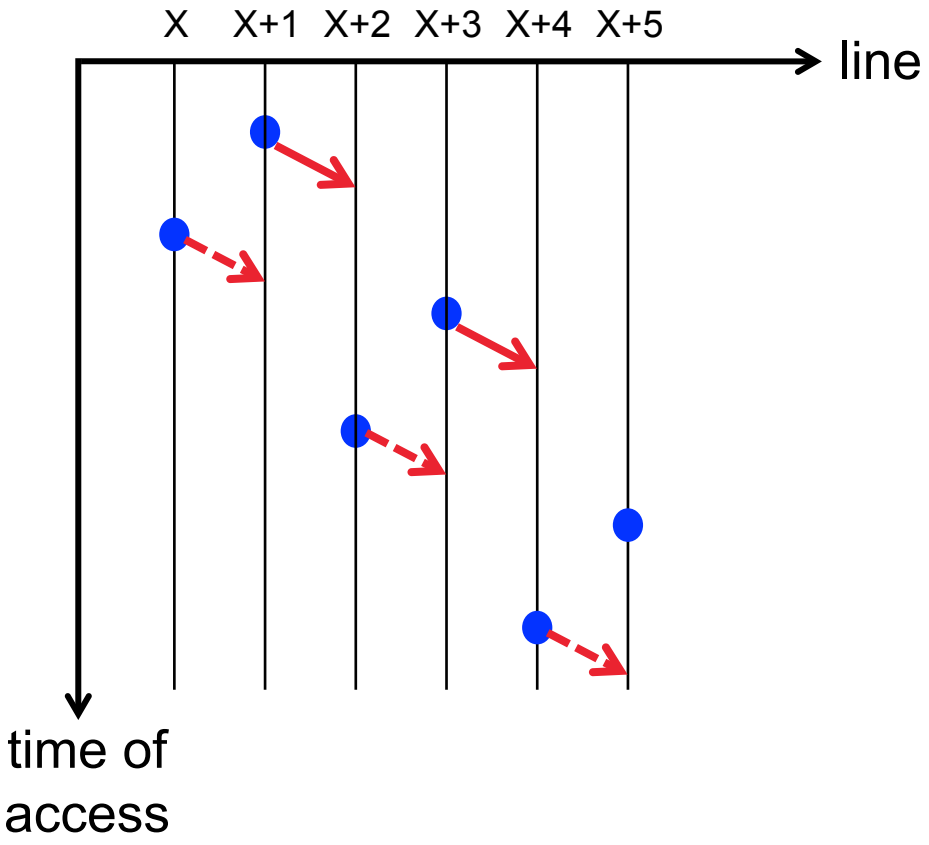
Late prefetches hurt performance



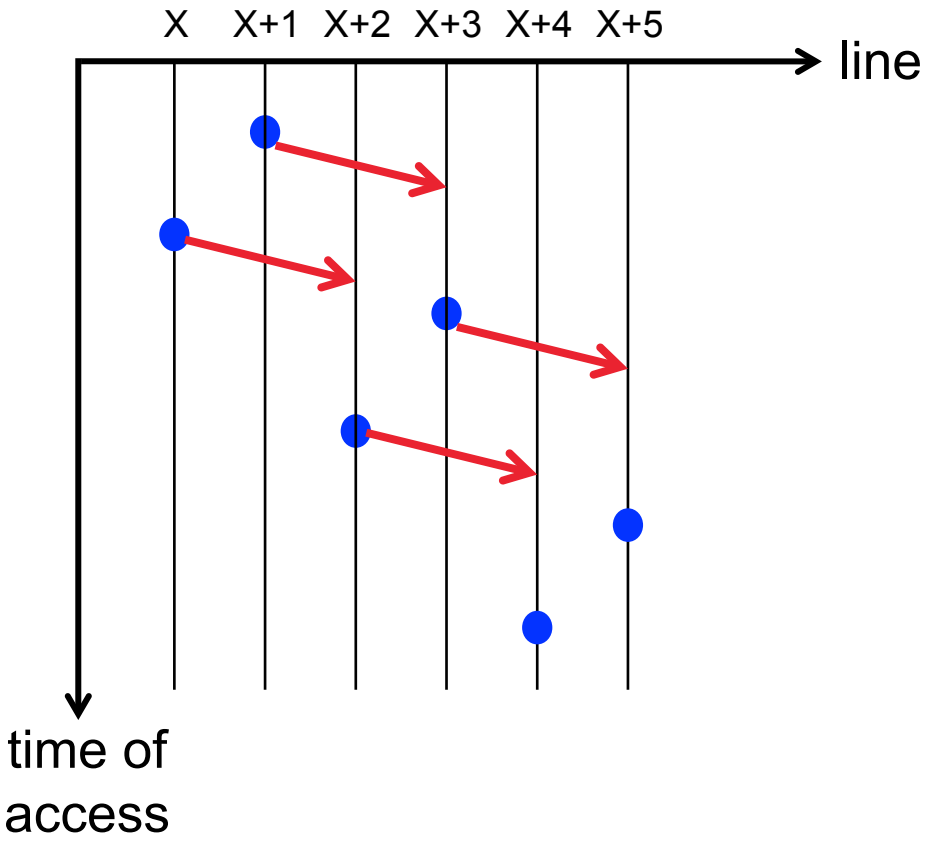
Greater offset makes timely prefetches



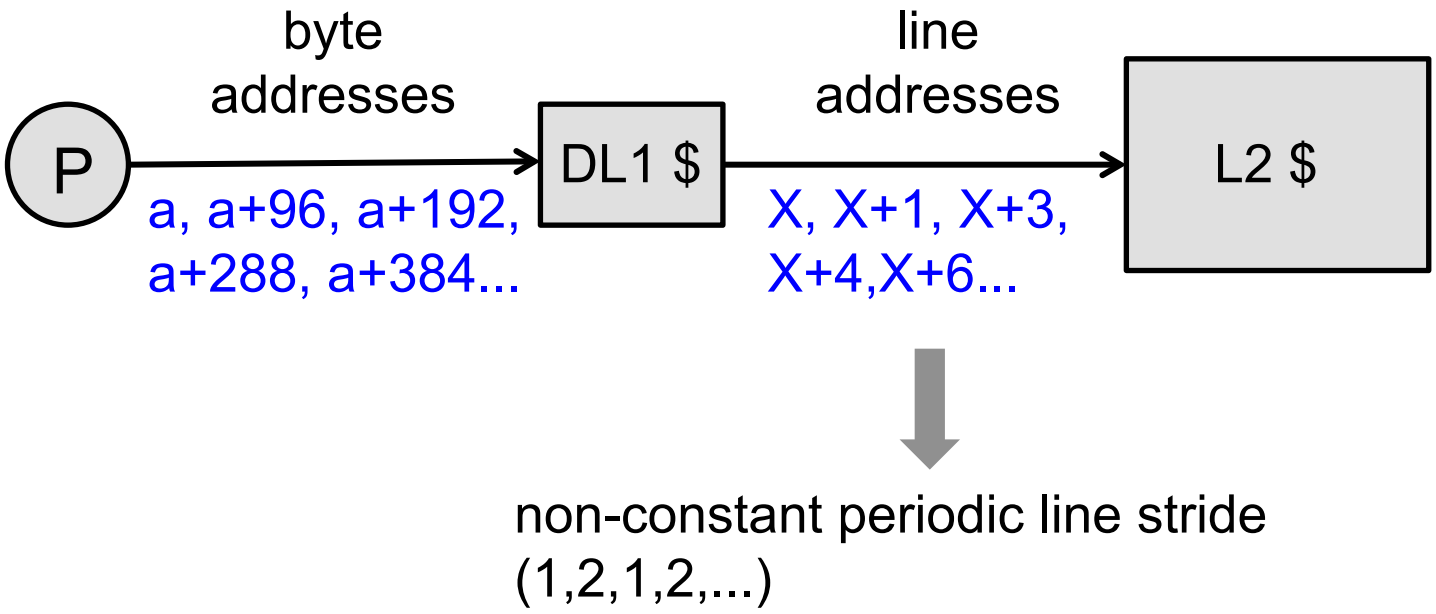
Scrambled stream hurts prefetch coverage



Greater offset less sensitive to scrambling

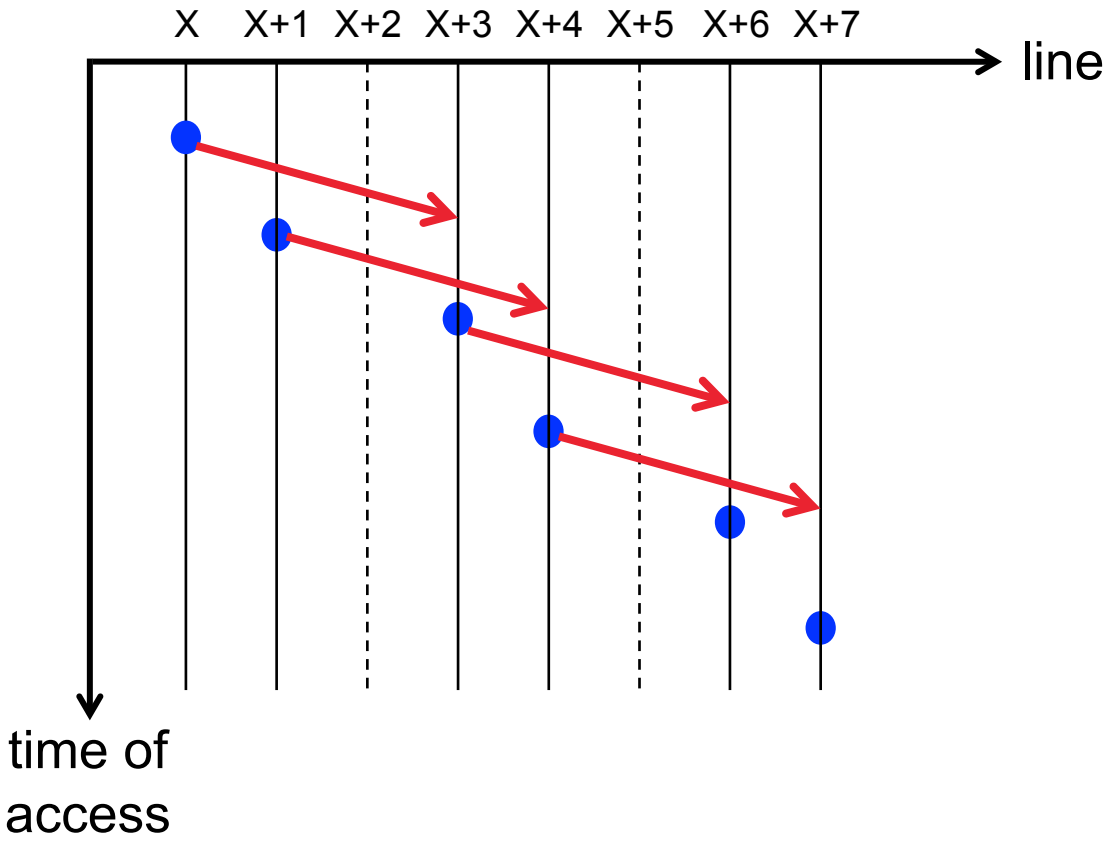


Periodic strides



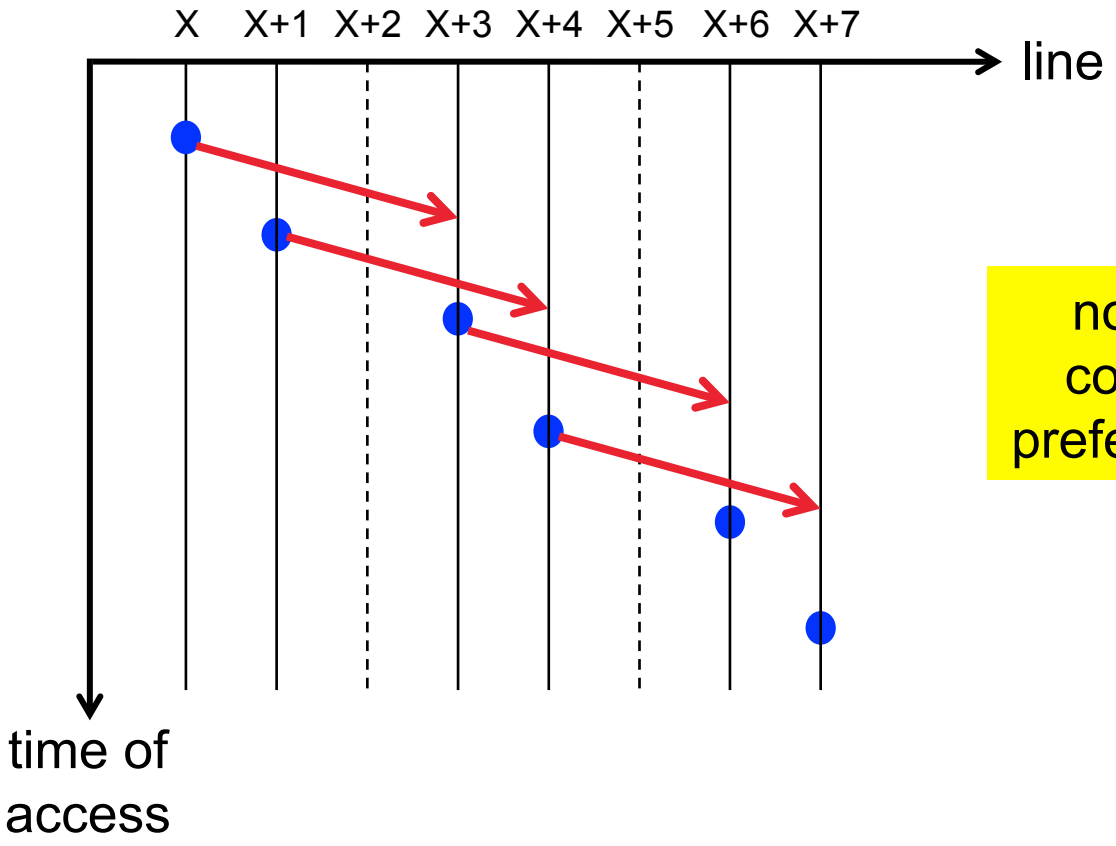
Offset = sum of strides in a period

or multiple of that number (for timeliness)



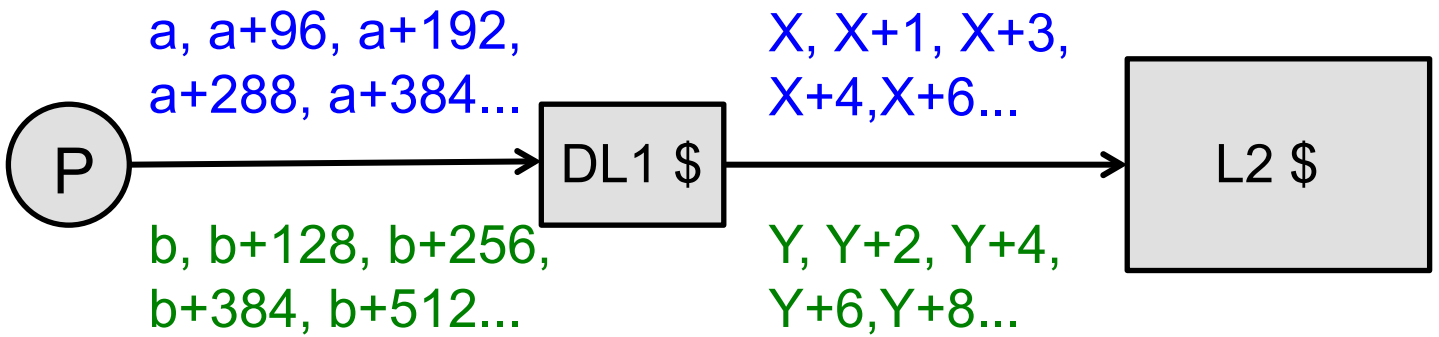
Offset = sum of strides in a period

or multiple of that number (for timeliness)



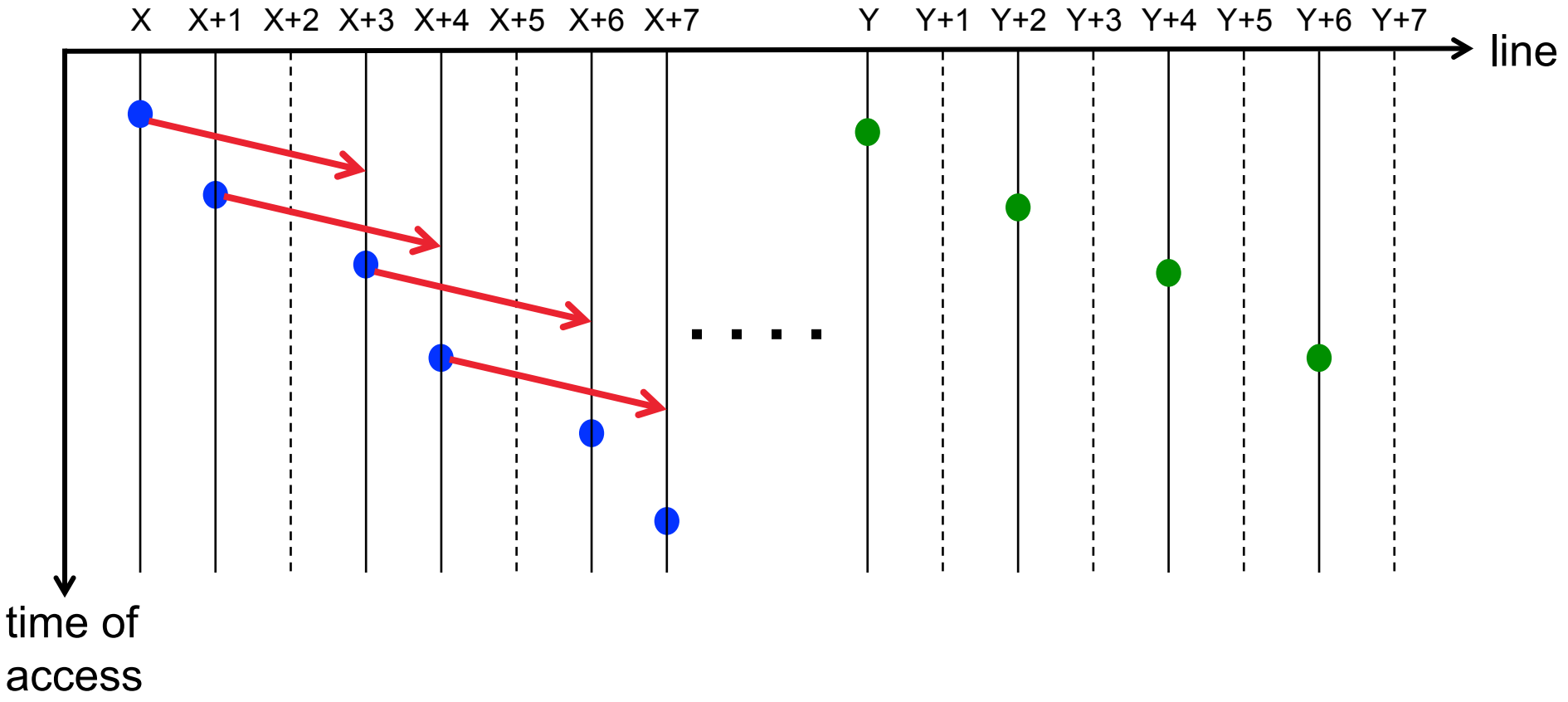
no need for complicated prefetcher here !

Interleaved streams



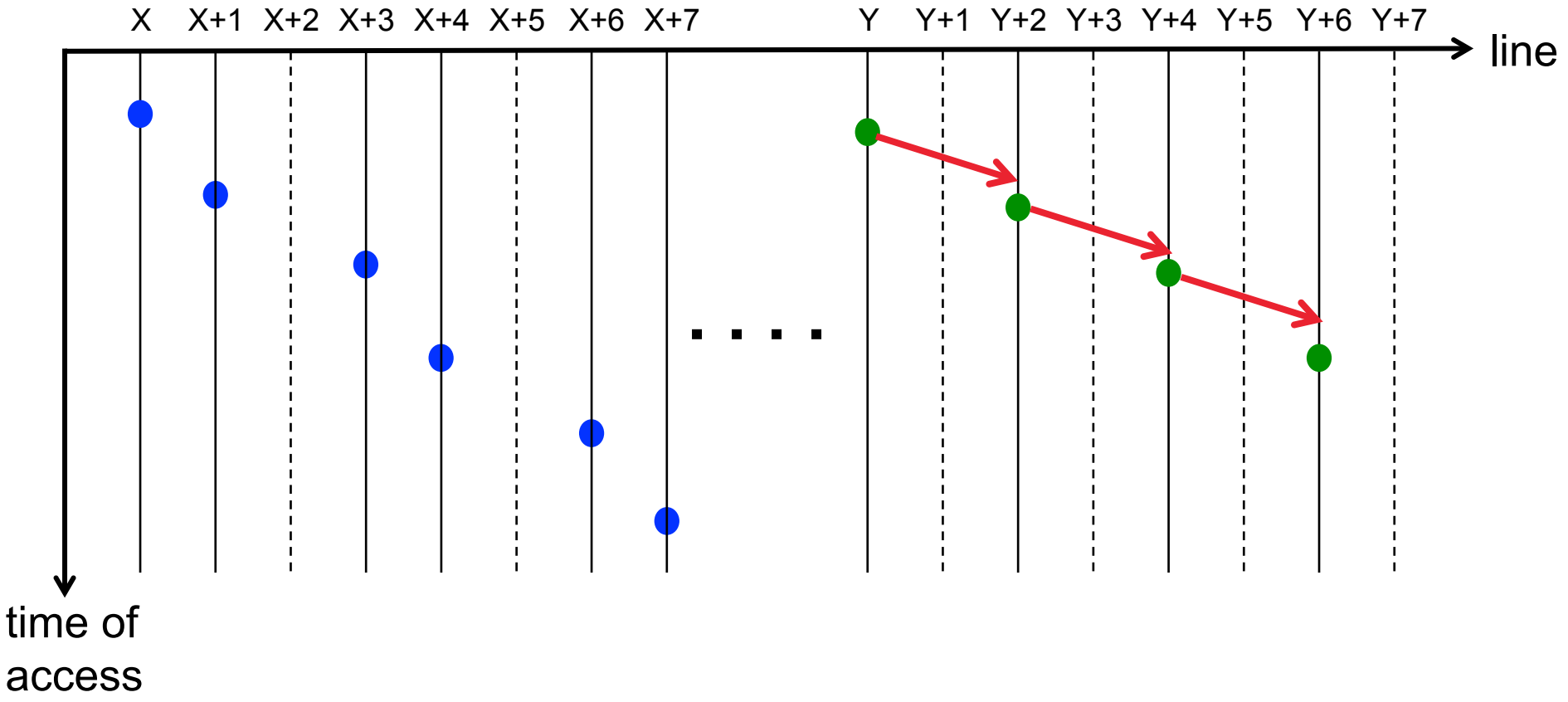
Interleaved streams

offset = multiple of 3



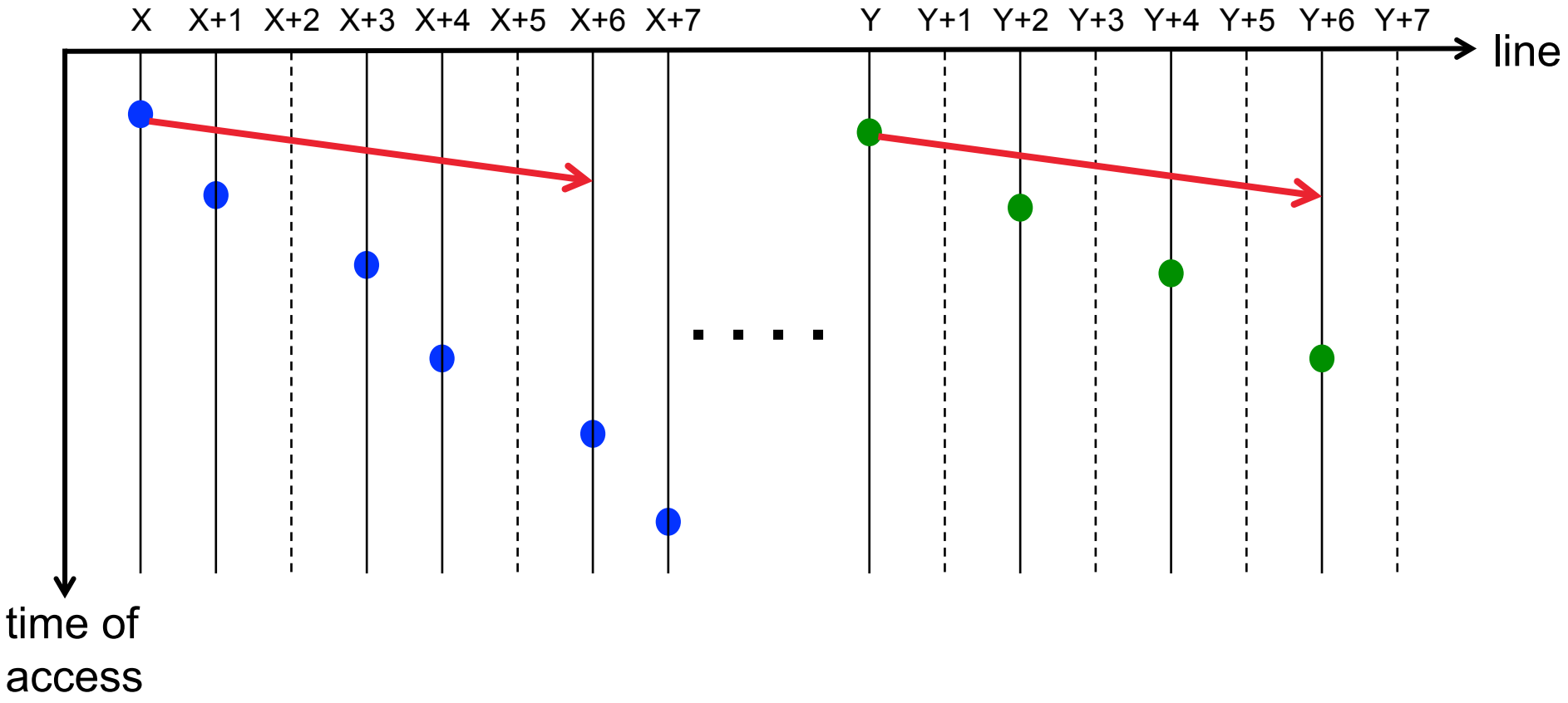
Interleaved streams

offset = multiple of 2



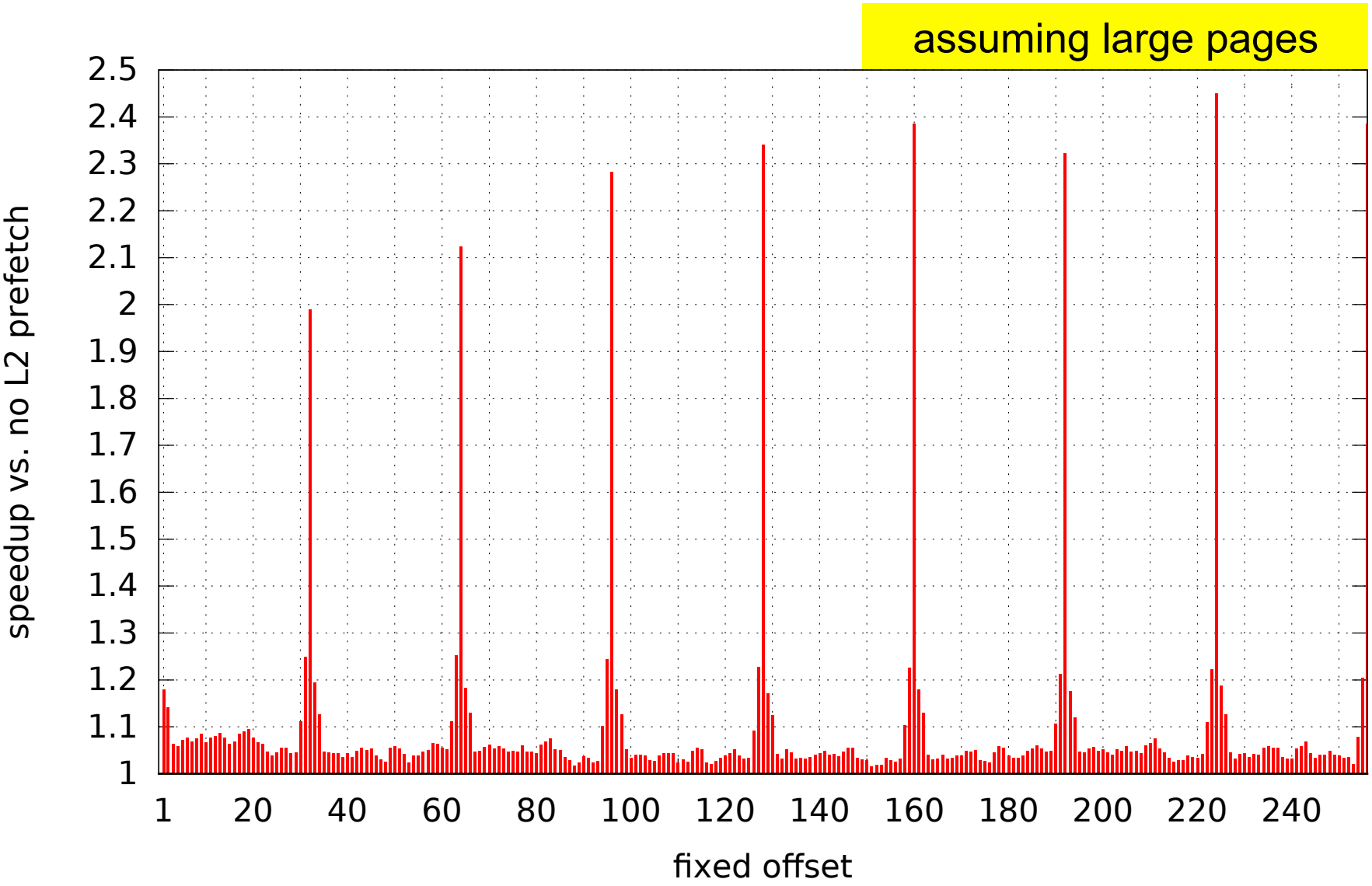
Interleaved streams

prefetch both streams with offset = multiple of 6

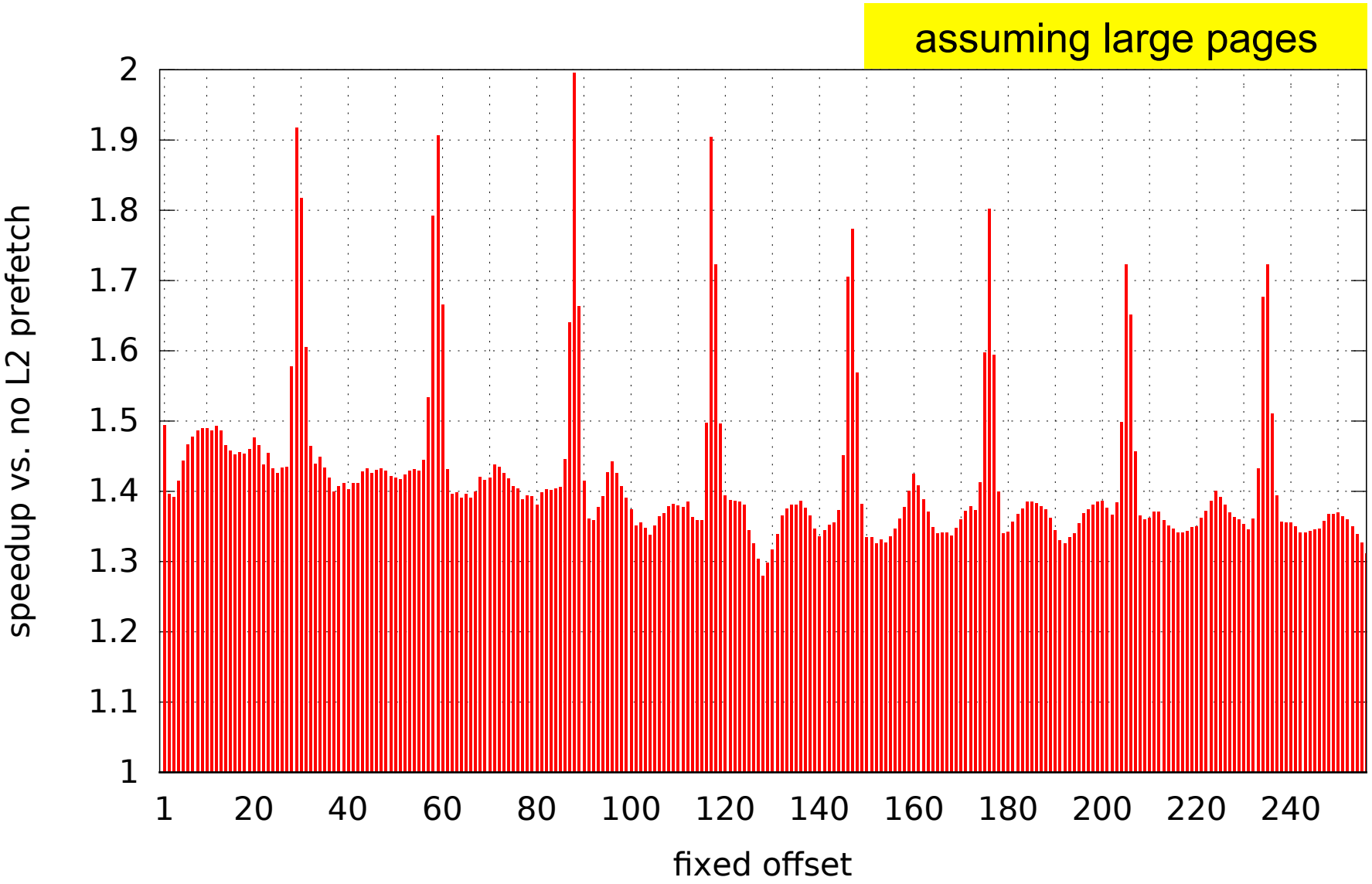


What about a fixed offset ?

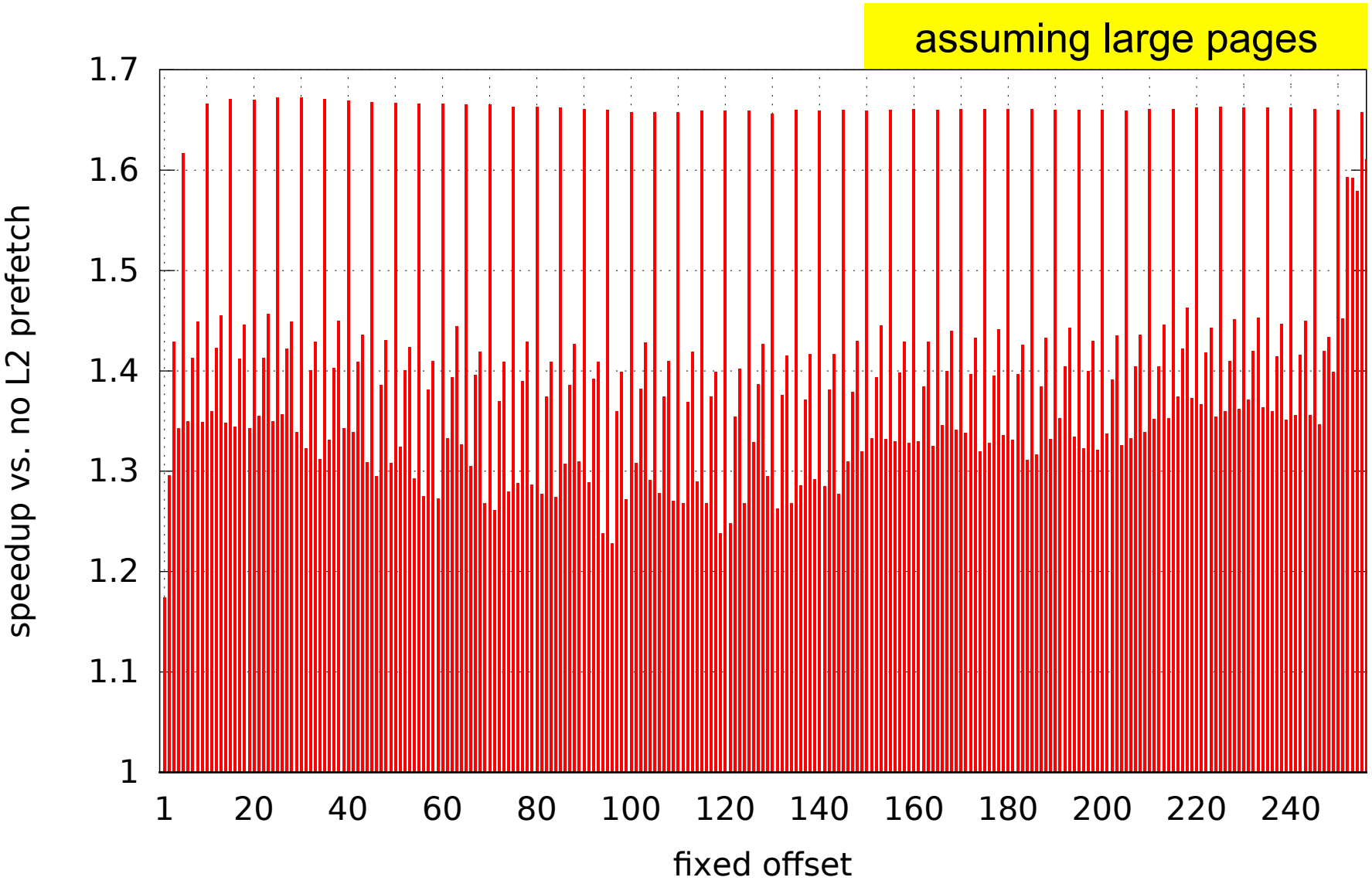
benchmark *milc*



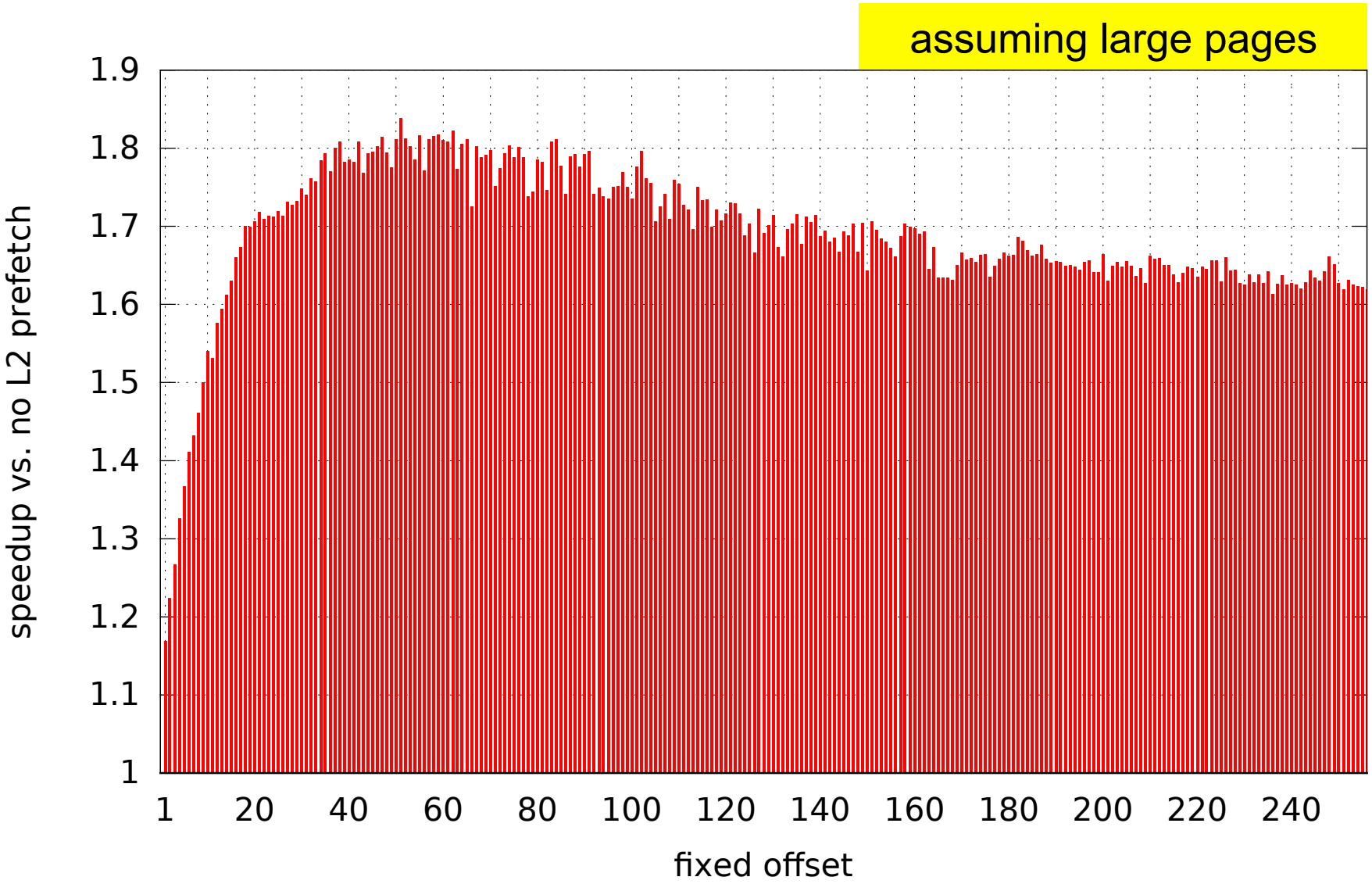
benchmark *GemsFDTD*



benchmark *lbm*



benchmark *libquantum*



Learnings

- The best offset depends on the application
  - full-fledged offset prefetchers select the offset dynamically
- The best offset may be > 100
  - when not limited by 4KB page boundaries
- Prefetch timeliness is essential for performance
  - high prefetch coverage is not sufficient

Dynamic offset selection

- Define a list of possible offsets
  - e.g., all numbers between -10 and +30
  - e.g., numbers between 1 and 255 with no prime factor greater than 5
- Define a mechanism for evaluating offsets
- Want simple hardware

Sandbox Prefetcher (SBP)

- Pugsley et al., HPCA 2014
- Introduces Sandbox method
  - evaluate offset by recording fake-prefetch addresses in Sandbox
  - on L2 cache access, check Sandbox ➔ if hit, increment score for offset
- Multi-degree prefetcher ➔ multiple prefetches per cache access
  - all the offsets with high enough coverage are potential candidates
  - smaller offsets first
- Prefetch timeliness not considered, only coverage

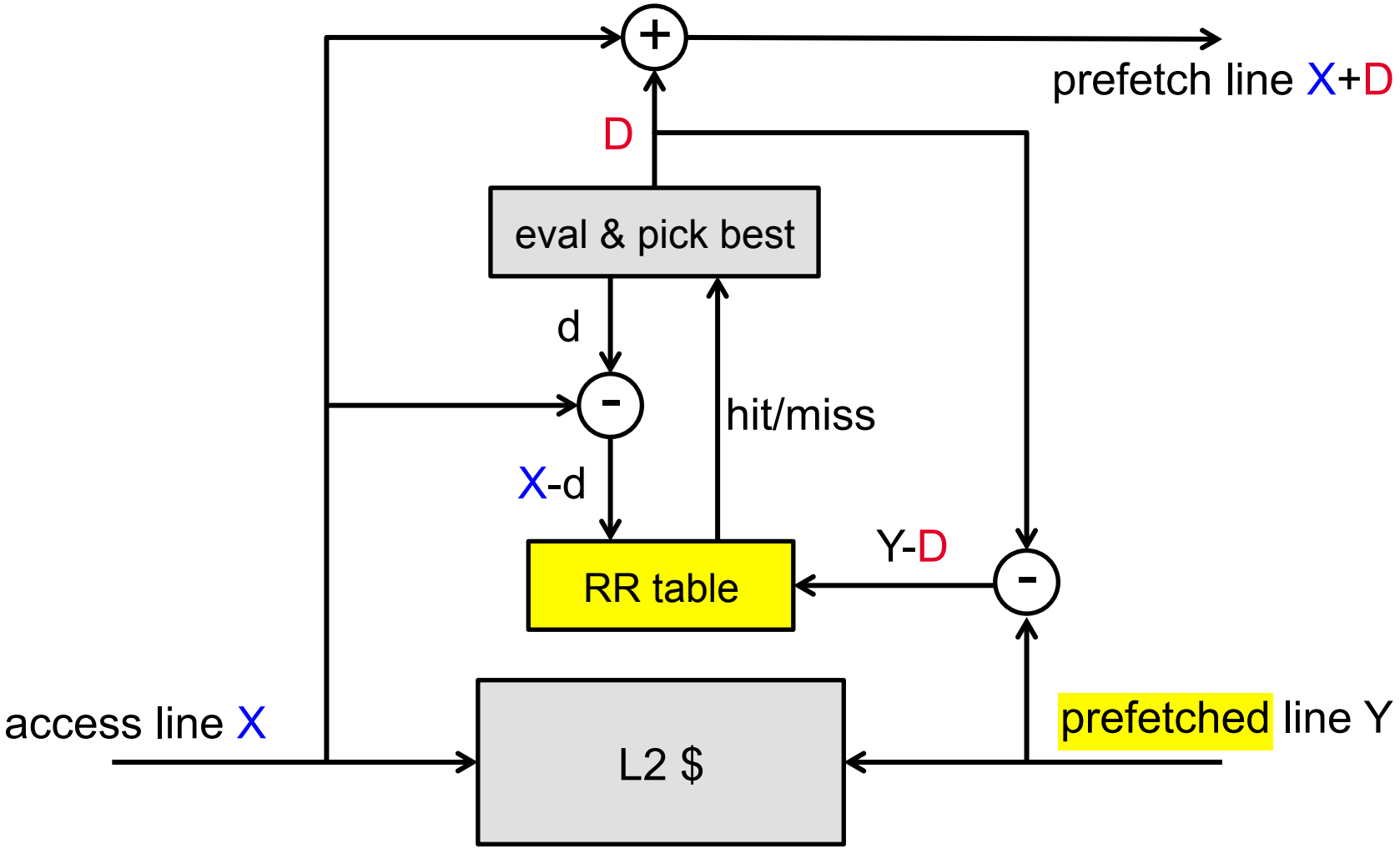
Best-Offset Prefetcher (BOP)

- Try to identify the single best offset
- Degree-one prefetch
  - one cache access ➔ one prefetch request
- New method for evaluating offsets
- Take into account both coverage and timeliness

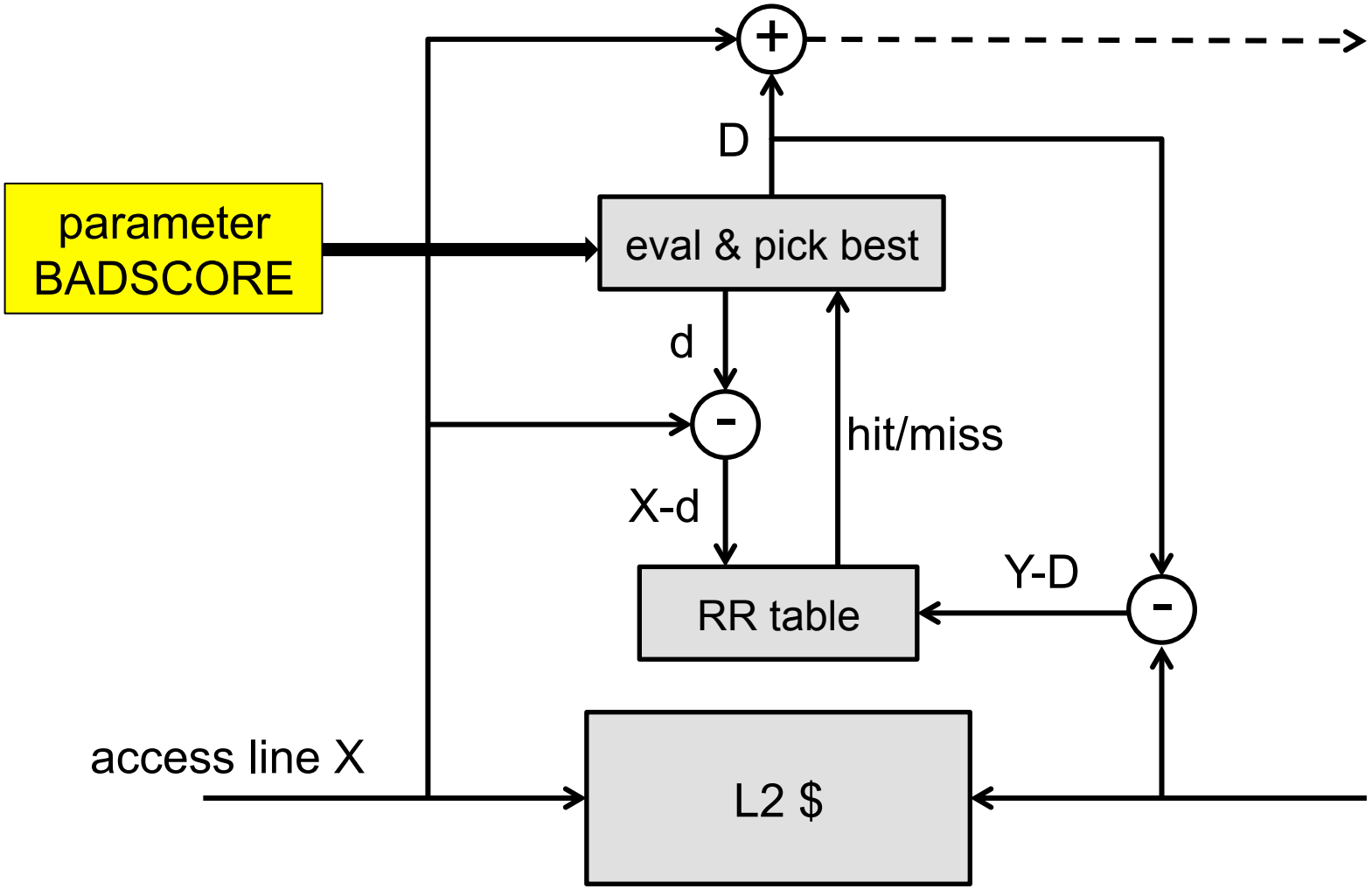
New method for evaluating offsets

- When a prefetch completes, store in a recent requests (RR) table the base address of the prefetch
  - prefetched line is X+D, base address is X
- To evaluate offset d, upon access to X, check if X-d is in RR table
  - if hit in RR table, increment score for offset d
- Evaluate all the offsets in the list, one by one
- When learning phase finished, pick offset with highest score, update prefetch offset D
  - then new learning phase starts

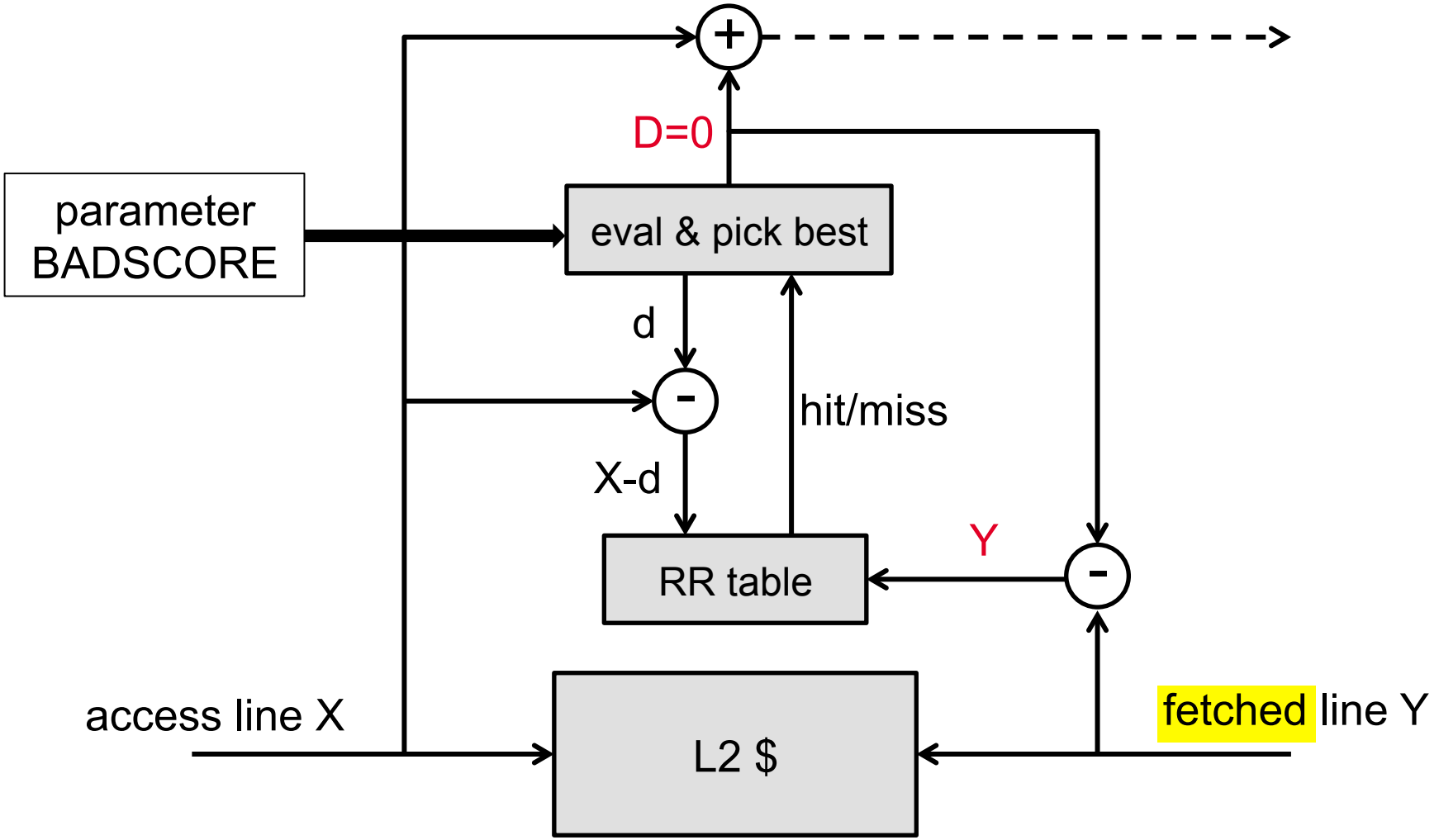
Best-Offset Prefetcher (BOP)



Turn prefetch off if best offset score is too low



Offset evaluation continues while prefetch is off



Hardware

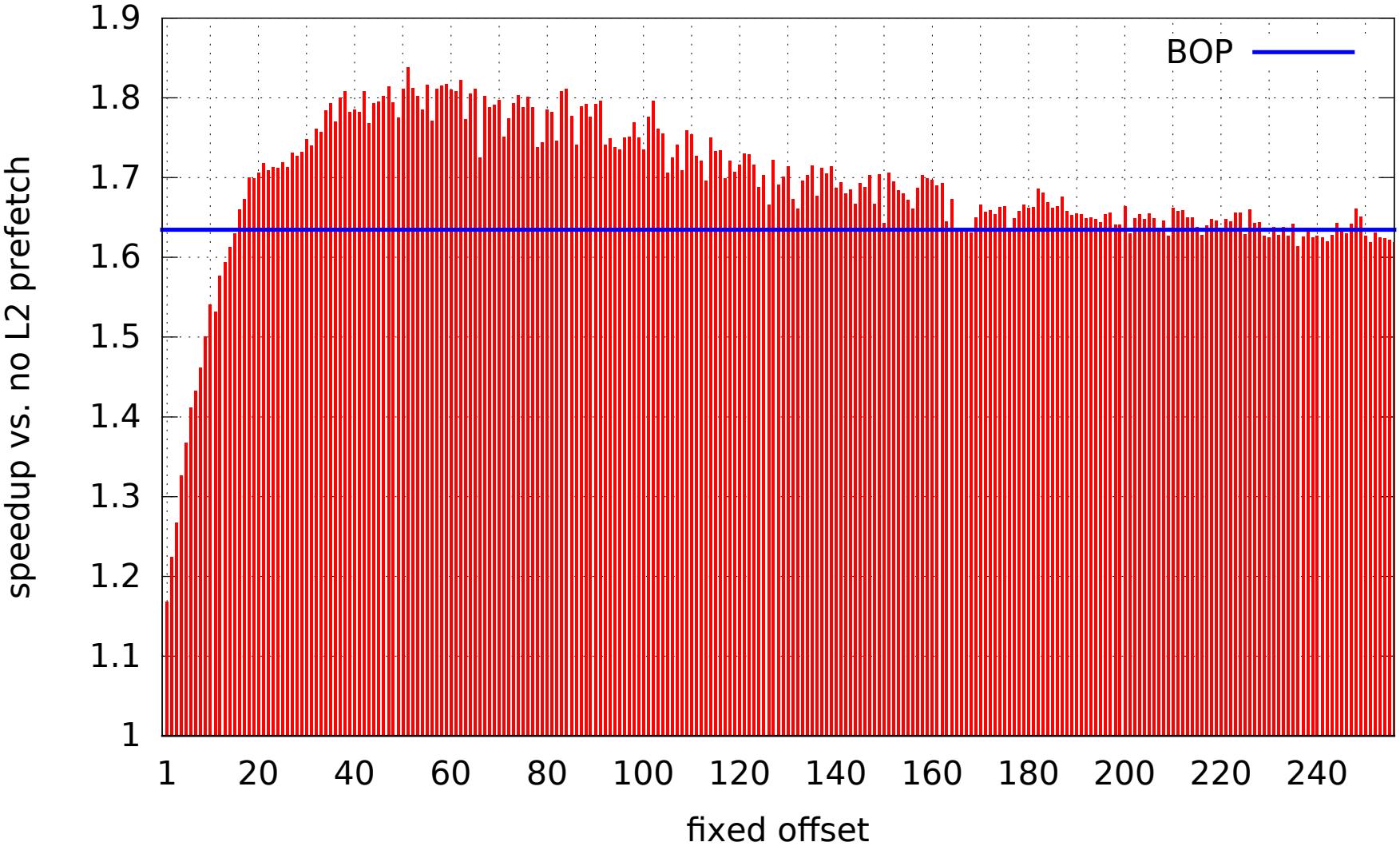
- One score per offset in the list
  - in the paper, 52 offsets, 5-bit scores ➔ 260 bits
- RR table
  - several possible implementations
  - in the paper, direct-mapped, 256 entries, 12-bit tags ➔ 3072 bits
- 3 adders
  - e.g., 64B line, 2MB page ➔ 15-bit adders
- Misc. logic
  - iterate on the list, increment scores, find highest score,...

BOP's main weakness

- Tradeoff between prefetch coverage and timeliness
  - small offsets give higher prefetch coverage
  - large offsets hide memory latency better

- BOP selects the offset yielding the most timely prefetches
  - most of the time, this is OK
  - but sometimes a smaller offset gives better performance

benchmark *libquantum*



Conclusion

- Prefetch timeliness is essential for performance
- BOP is very effective on the SPEC CPU benchmarks
  - though not always optimal
- Simple hardware
- BOP is degree-one prefetcher ➔ one prefetch per L2 access
  - multi-degree prefetching not the right solution for timeliness issues
- Maybe we don't need multi-degree prefetching at all
  - if we obtain 100% prefetch coverage with degree-2 prefetching, it means that we are doubling the memory traffic

thanks for your attention