Self-adaptive Address Mapping Mechanism for Access Pattern Awareness on DRAM

Chundian Li*, Mingzhe Zhang*, Zhiwei Xu*, Xianhe Sun†

* ICT, CAS, China

† Illinois Tech, USA

12/17/2019

中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY

Outline

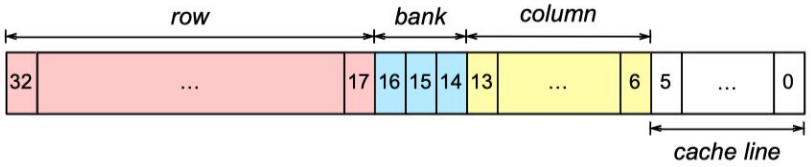
- Introduction & Background
- Motivation
- Design
- Experiments
- Conclusion
- Future work

Introduction

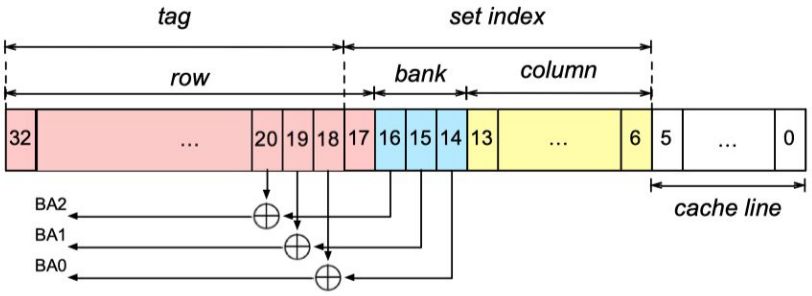
- Memory wall.
- DRAM serve data accesses in two efficient ways.
  - Locality: row buffer.
  - Memory-level parallelism (MLP): channel/bank parallelism.
- Worst case.
  - Neither locality nor concurrency.
  - When and Why?
- Mismatch between data layout and access pattern.
  - Data layout: row-major, column-major, bank-major, etc.
  - Access pattern: stream, stride, random, pointer, etc.
  - (Take regular access patterns in our study).

Background

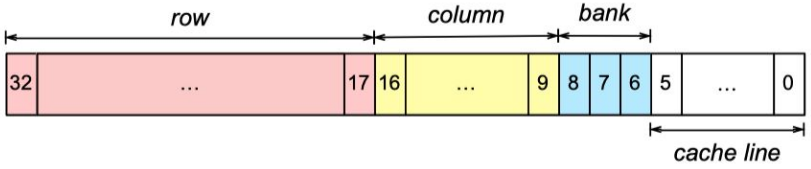
- Layout <- Address Mappings
  - RI: spatial row-buffer locality.
    - XOR: increase MLP potential.
  - CI: bank parallelism.
- How about these mappings?
  - Row bits are in the high zone.
  - Designed for accesses with short distance.
- Problems?
  - If distance is quite long, how?
    - Worst case will appear.
    - Take Matrix Multiplication as an example.
  - XOR can really match all the access patterns?
    - No.



(a) Row-interleaving Address Mapping (Baseline, RI)



(b) XOR Address Mapping (XOR)



(c) Cache-interleaving Address Mapping (CI)

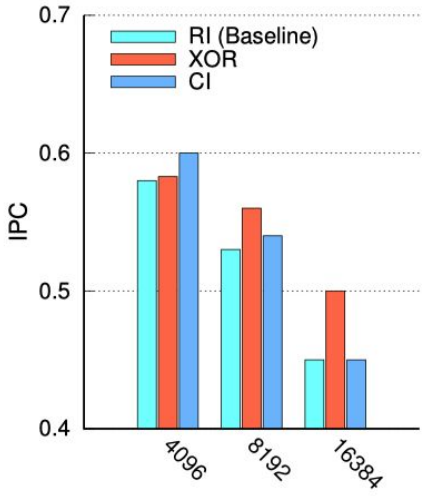
Motivation

- Take three versions and scales of GEMM as cases.
  - Naïve.
  - Cache-friendly: tiling.
  - Highly-optimized: Intel MKI.

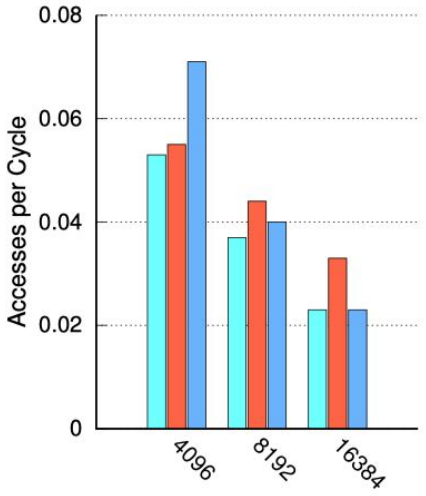
Version	Scale	L1 MR	LLC MR	LLC MPKI
Naive	4096	44.3%	99.9%	143
	8192	43.8%	99.9%	143
	16384	46.5%	90.4%	142
Tiling	4096	30.3%	70.6%	7
	8192	31.8%	70.5%	7
	16384	32.6%	70.9%	8
Intel MKI	4096	9.8%	61.3%	7
	8192	9.8%	62.7%	8
	16384	10.3%	59.9%	9

Motivation

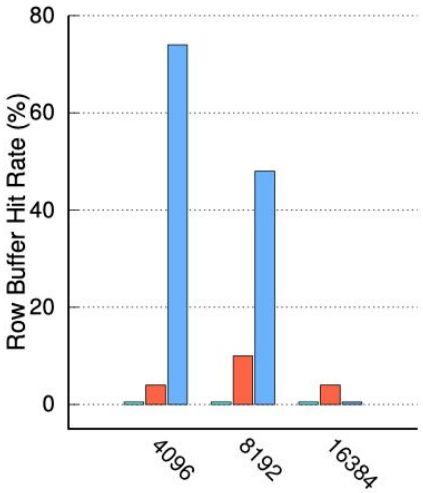
- Observation 1.
  - RI/ XOR/ CI may fail to provide its advantages when they happen to mismatch access patte on DRAM.



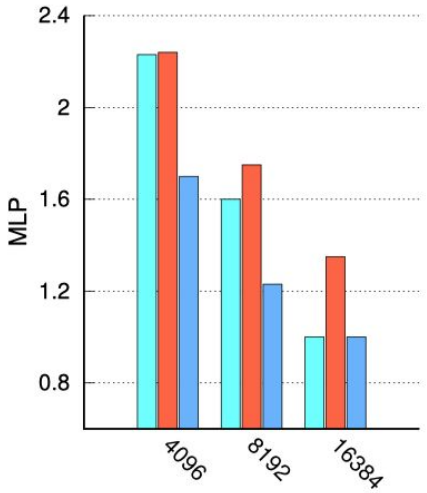
(a) IPC



(b) APC on DRAM



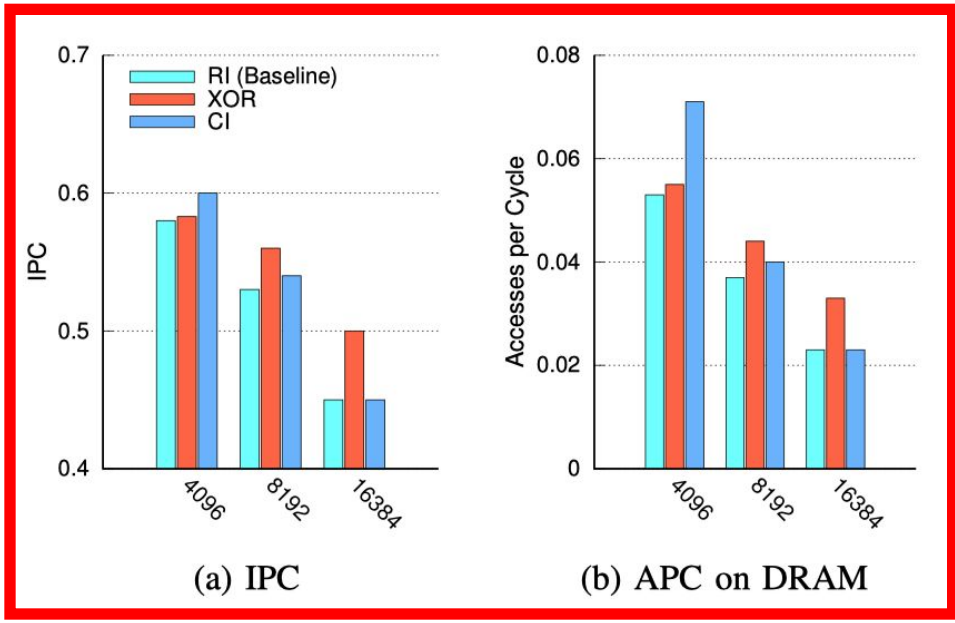
(c) Locality on DRAM

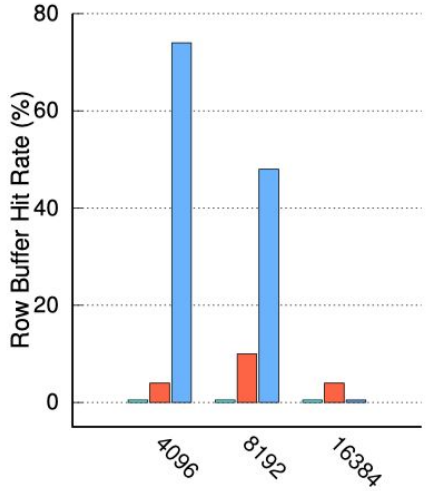


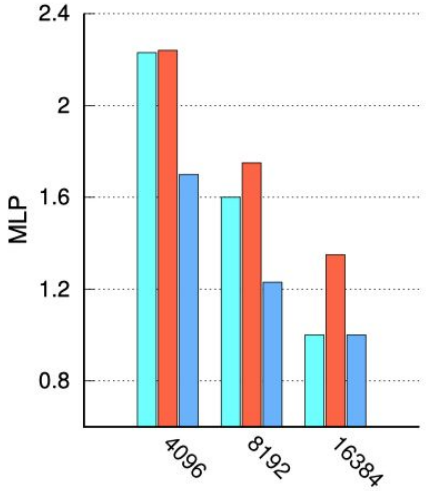
(d) MLP on DRAM

Motivation

- Observation 2.
  - Performance of XOR conquers one of CI, or the other way around on different patterns.

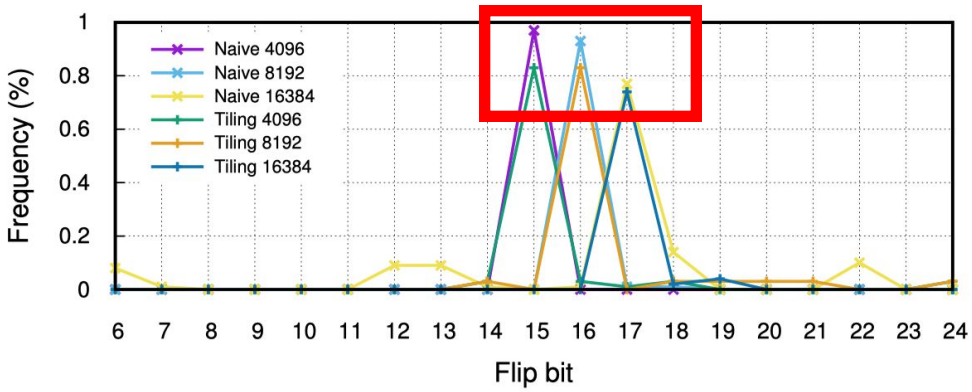




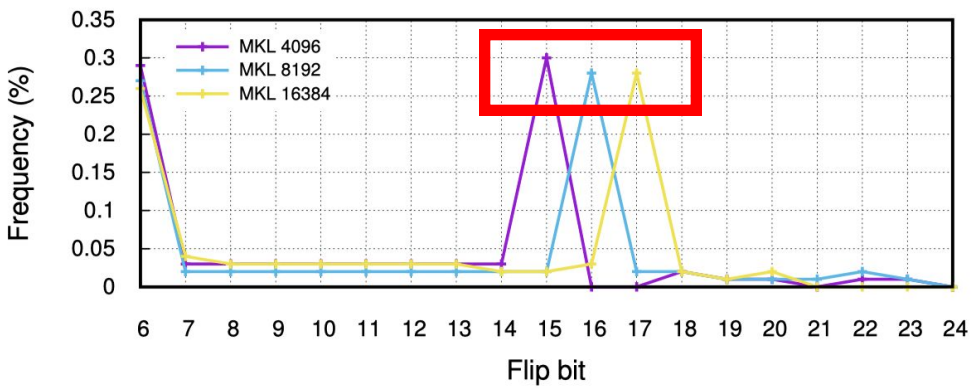


Motivation

- Bit flip:
  - address distance.
- Observation 3.
  - RI/ XOR/ CI may all degrade DRAM performance when bit flips are outstanding.
  - Consecutive accesses span a long distance that disables both locality and MLP.



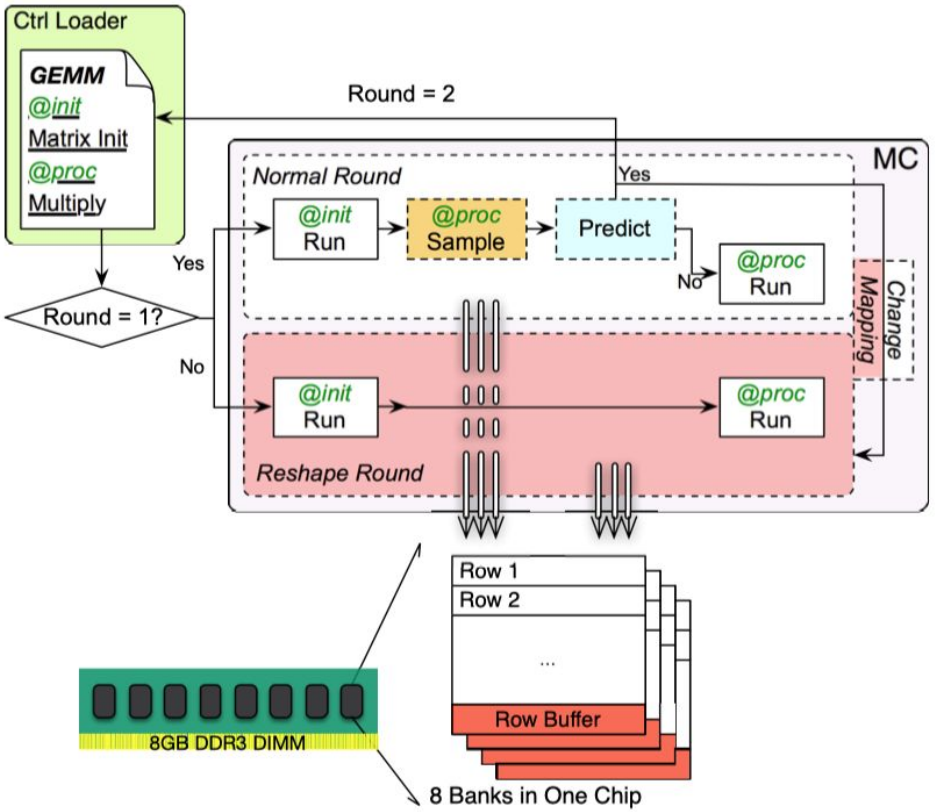
(a) Bit Flips of Naive and Tiling



(b) Bit Flips of Intel MKL

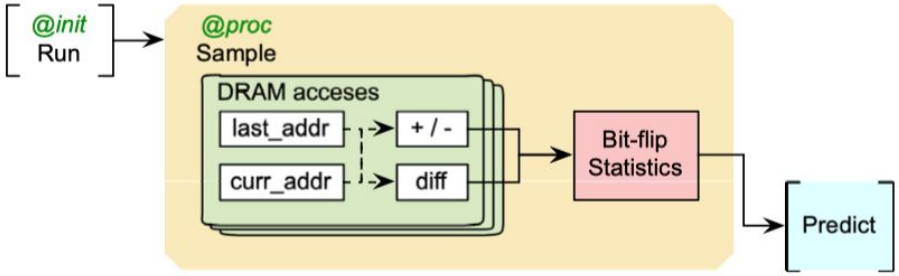
Design

- Two tags.
  - Distinguish two procedures.
  - MC decides when to sample.
- Software-level: Ctrl Loader.
  - Interact with MC.
- Hardware-level: MC Modifications.
  - Flip sampling.
  - Pattern-aware Prediction.

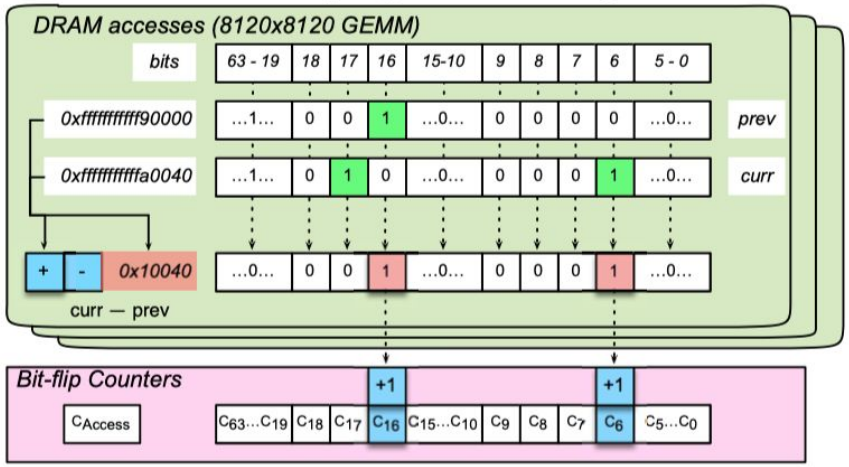


Design

- Flip sampling.
  - Care about adjacent accesses.
  - Light-weight.
  - Little cost.
- Access pattern.
  - Check bit flips for all 64 bits.
  - Decide which bit is outstanding.
  - Reduce side effects of access thrashing.



(a) Sampling Module Design

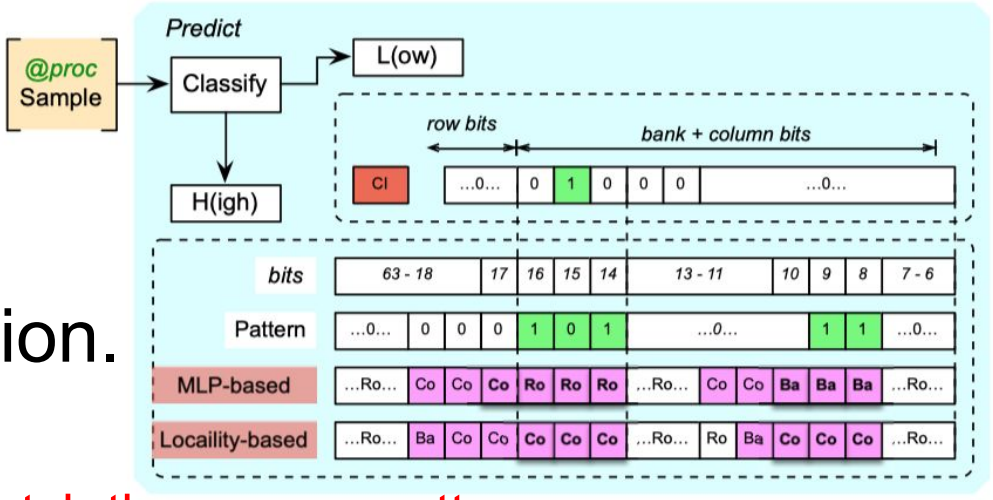


(b) Implementation of Sampling and Example

$$Convergence\ Condition:\ \forall i(\rho_i > \lambda) \rightarrow |\rho_i - \rho_i'| < \sigma.$$

$$Pattern: \{P_i\ |P_i = 1\ if\ \rho_i > \lambda\ otherwise\ 0\}.$$

Design



- Pattern-aware Prediction.
  - Basic idea:
    - Reshape the layout to match the access pattern.
  - Based on prominent flipping.
  - Two strategies. (Aggressiveness control)
    - Locality-based strategy.
    - MLP-based strategy.
- Profit model for this mechanism.

$$Profit(GEMM) \approx 1 - \frac{T(Proc')}{T_{staic}} \approx 1 - \frac{IPC(Proc)}{IPC_{Proc'}}.$$

Experiments

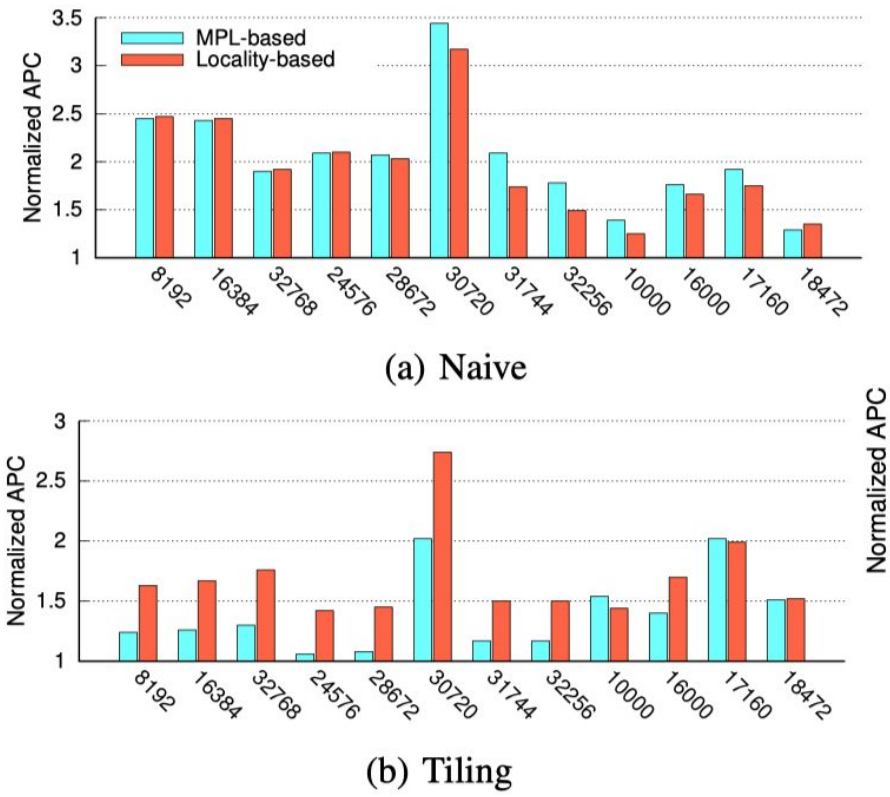
- Testbed.
  - Ramulator + Champsim.
  - Representative benchmarks: diverse scales of GEMM.
  - Baseline: XOR.

Processor	One core, x86-64, 4-wide issue and 3-wide retire, 1.8 GHz, out-of-order
L1 cache	32 KB D-cache, 32 KB I-cache, 8 MSHRs, LRU policy, 4 cycles
L2 cache	128 KB, 16 MSHRs, LRU policy, 4 cycles
LLC cache	4 MB, 48 MSHRs, LRU policy, 15 cycles
MC	FR-FCFS, 16 KB row size, 400MHz, 32-entry queue, open-page policy
DRAM	DDR3-2133, 1-channel, 1-rank, 8 banks 2^{16} rows, 2^{11} columns, 8B bus

Groups	Matrix size	Scales
A	2^N. E.g., 8192.	8192, 16384, 32768
B	Tailing zeros. E.g., 24576 (1100...0)	24576, 28672, 30720, 31744, 32256
C	1000x and random zeros. E.g., 10000 ; 17160 (100001100001)	10000, 16000; 17160, 18472

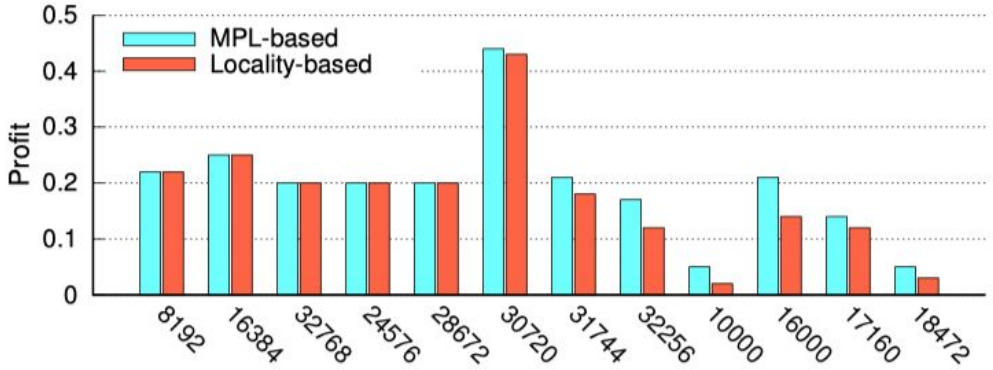
Experiments

- DRAM performance.
  - MLP-based strategy.
    - Naïve: 2.1x.
    - Tiling: 1.4x.
  - Locality-based.
    - Naïve: 1.9x.
    - Tiling: 1.7x.
    - Intel MLK: 1.6x.

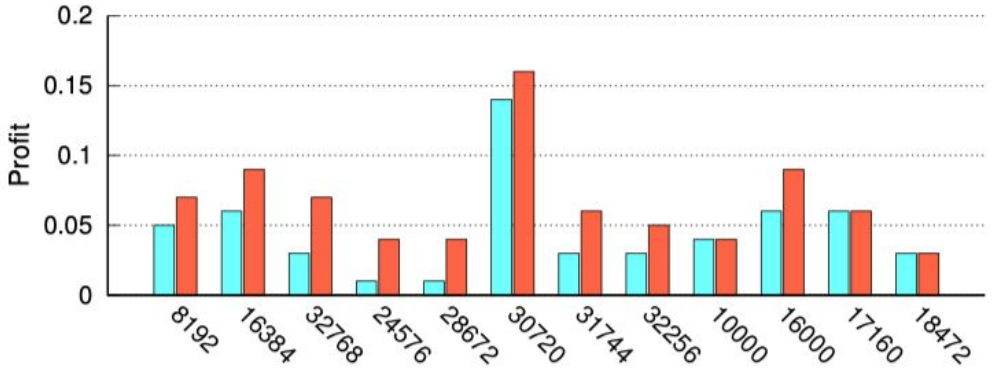


Experiments

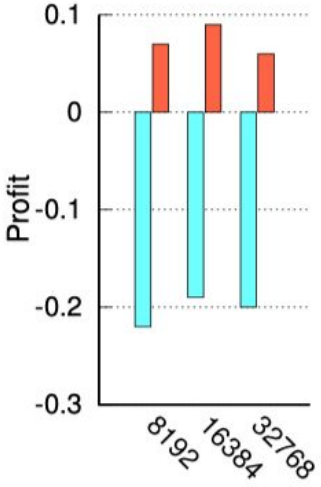
- IPC for whole execution.
  - Execution time decreases by 24%, 8%, and 7% averagely.



(a) Naive



(b) Tiling



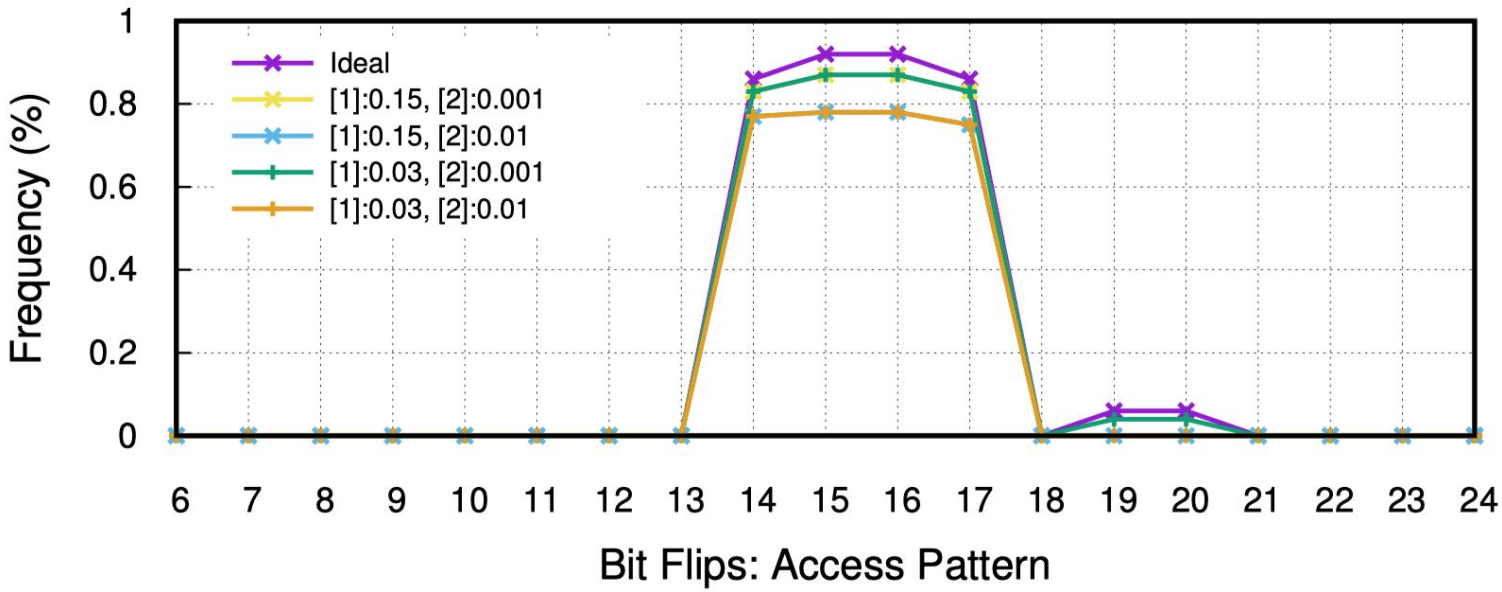
(c) Intel MKL

Experiments

- Sensitivity study.
  - [1]-λ. How much frequency of bit flips is prominent to the access pattern
  - [2]-σ. Speed of reaction.

$$Convergence\ Condition:\ \forall i(\rho_i > \lambda) \rightarrow |\rho_i - \rho_i'| < \sigma.$$

$$Pattern: \{P_i\ |P_i = 1\ if\ \rho_i > \lambda\ otherwise\ 0\}.$$



Conclusion

- Key observation.
  - Inefficiency comes from the mismatch of access patterns and data layout.
  - Worst case: both locality and parallelism are harmed.
- An adaptive address mapping mechanism to be aware of access patterns.
  - Bridging the huge mismatch between access patterns and data layout on DRAM.
  - Adjustable to different access patterns by adopting suitable mappings to gain either locality or bank parallelism.

Future work

- Show potential on other benchmarks.
  - Dig more profit from other applications with regular patterns.
- Fast reshaping.
  - Exploit efficient data movement in 3D-stack DRAM to support fast reshaping on runtime after predicting a suitable mapping.

Thank you.

Q & A.