Gather-Scatter DRAM

In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses

Vivek Seshadri

Thomas Mullins, Amirali Boroumand, Onur Mutlu,

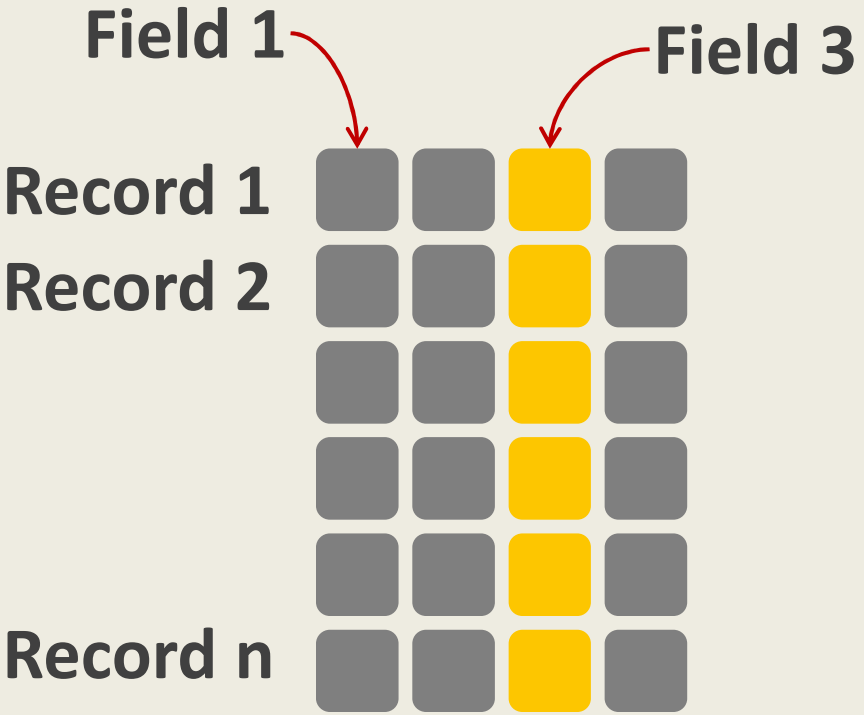
Phillip B. Gibbons, Michael A. Kozuch, Todd C. Mowry

Executive summary

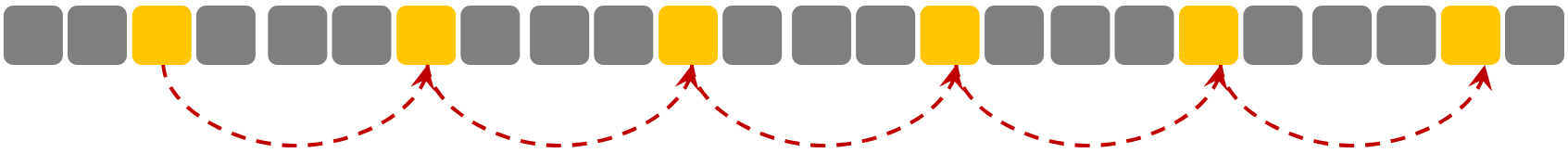
- **Problem: Non-unit strided accesses**
  - Present in many applications
  - In-efficient in cache-line-optimized memory systems
- **Our Proposal: Gather-Scatter DRAM**
  - Gather/scatter values of strided access from multiple chips
  - Ideal memory bandwidth/cache utilization for power-of-2 strides
  - Requires very few changes to the DRAM module
- **Results**
  - In-memory databases: the best of both row store and column store
  - Matrix multiplication: Eliminates software gather for SIMD optimizations

Strided access pattern

In-Memory Database Table



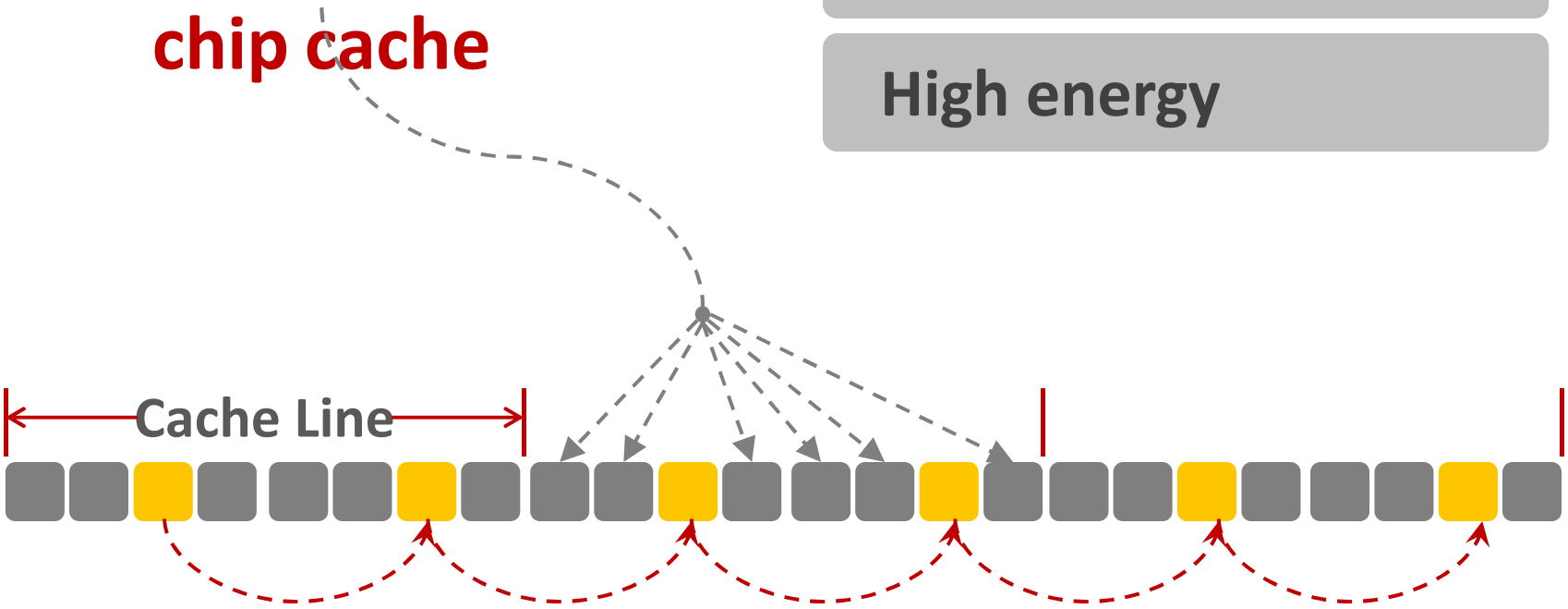
Physical layout of the data structure (row store)



Shortcomings of existing systems

Data **unnecessarily** transferred on the **memory channel** and stored in **on-chip cache**

- High latency
- Wasted bandwidth
- Wasted cache space
- High energy



Prior approaches

Improving efficiency of fine-grained memory accesses

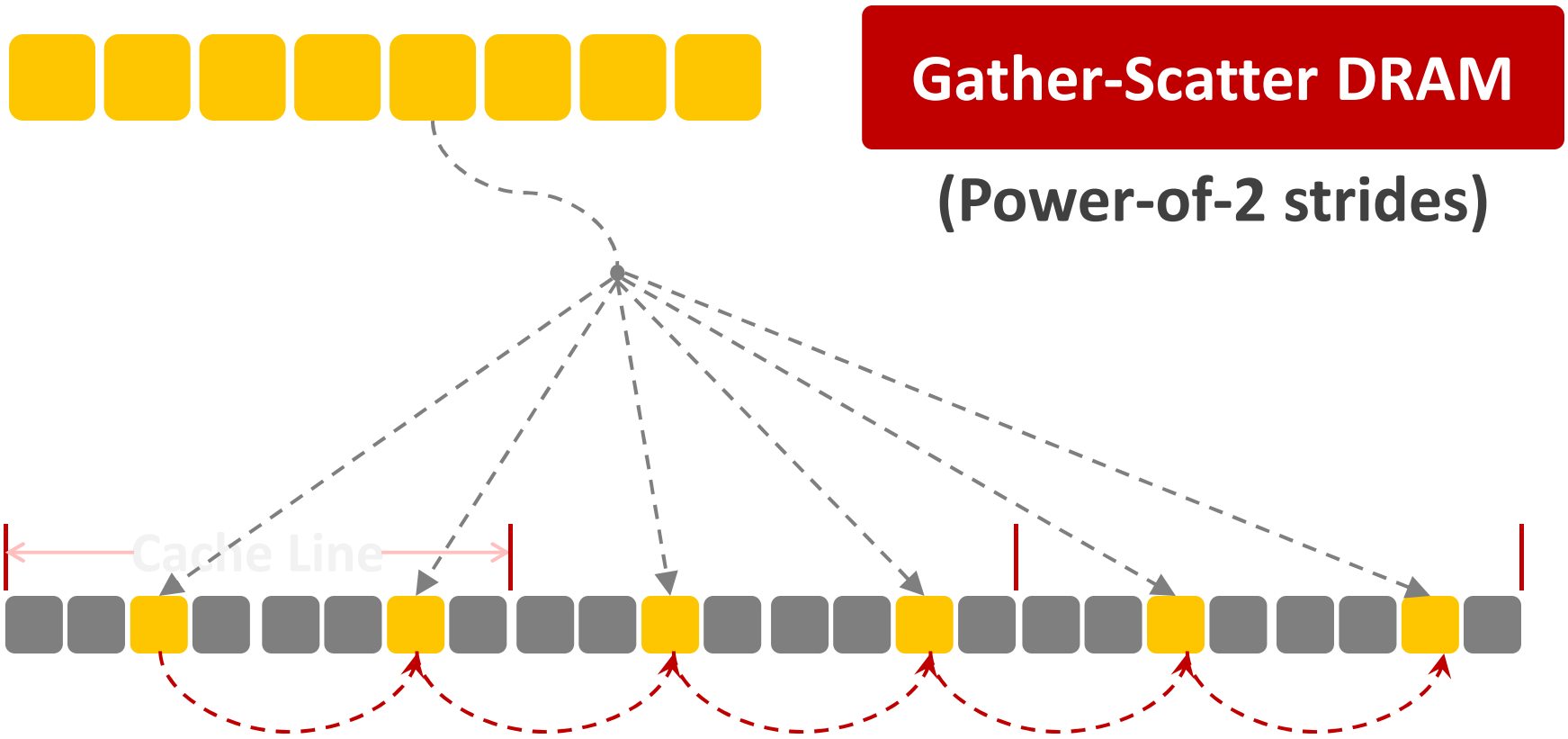
- **Impulse Memory Controller (HPCA 1999)**
- **Adaptive/Dynamic Granularity Memory System (ISCA 2011/12)**

Costly in a commodity system

- **Modules that support fine-grained memory accesses**
  - **E.g., mini-rank, threaded-memory module**
- **Sectored caches**

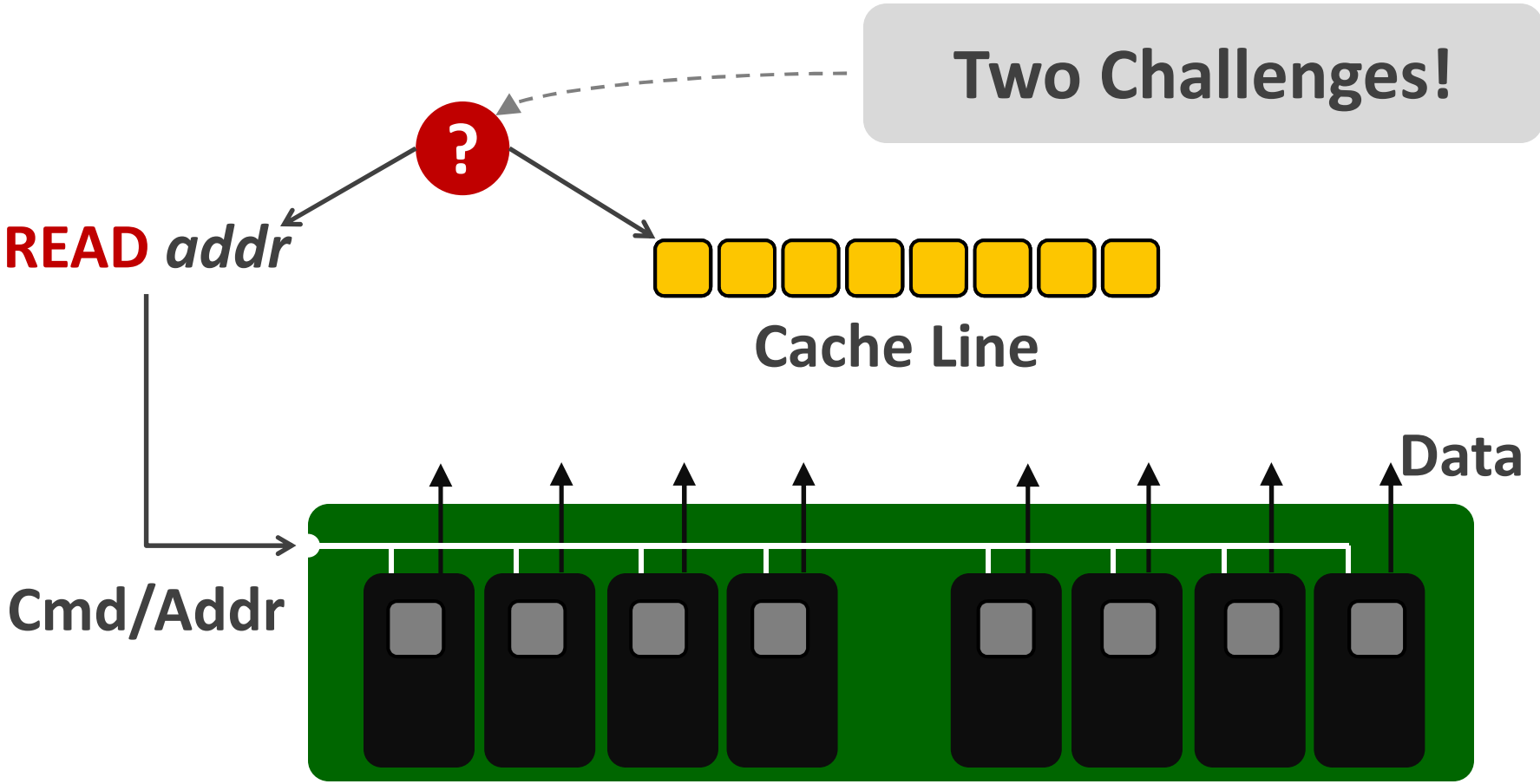
Goal: Eliminate inefficiency

Can we retrieve a only useful data?



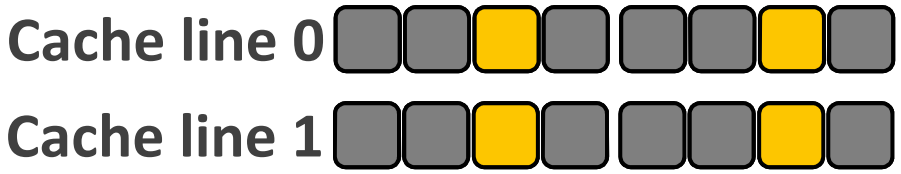
DRAM modules have multiple chips

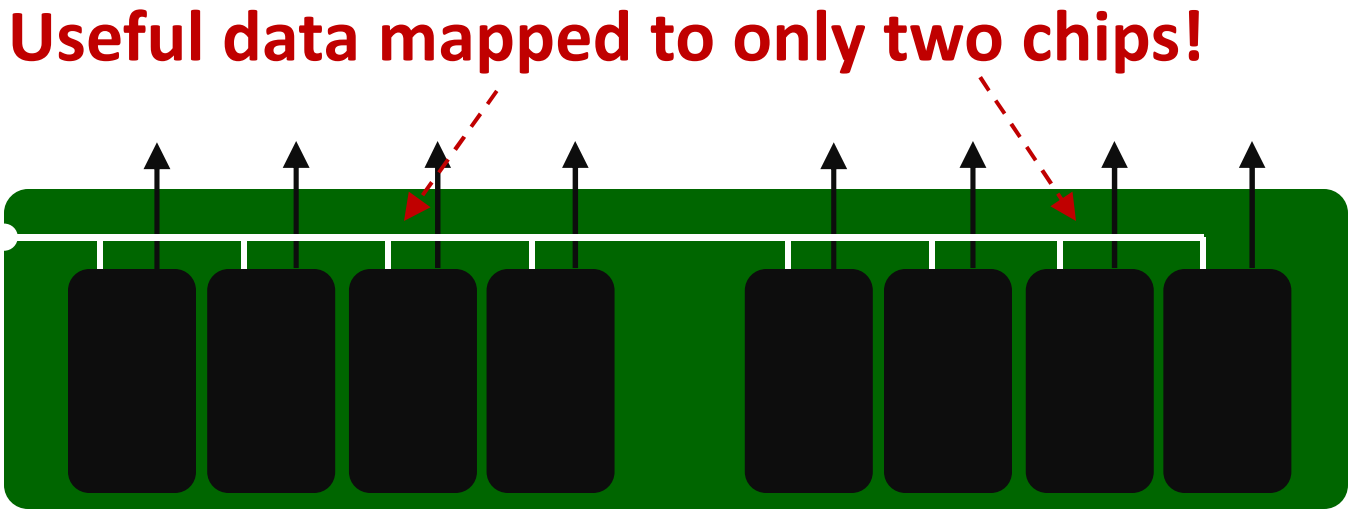
All chips within a “rank” operate in unison!



Challenge 1: Chip conflicts

Data of each cache line is spread across all the chips



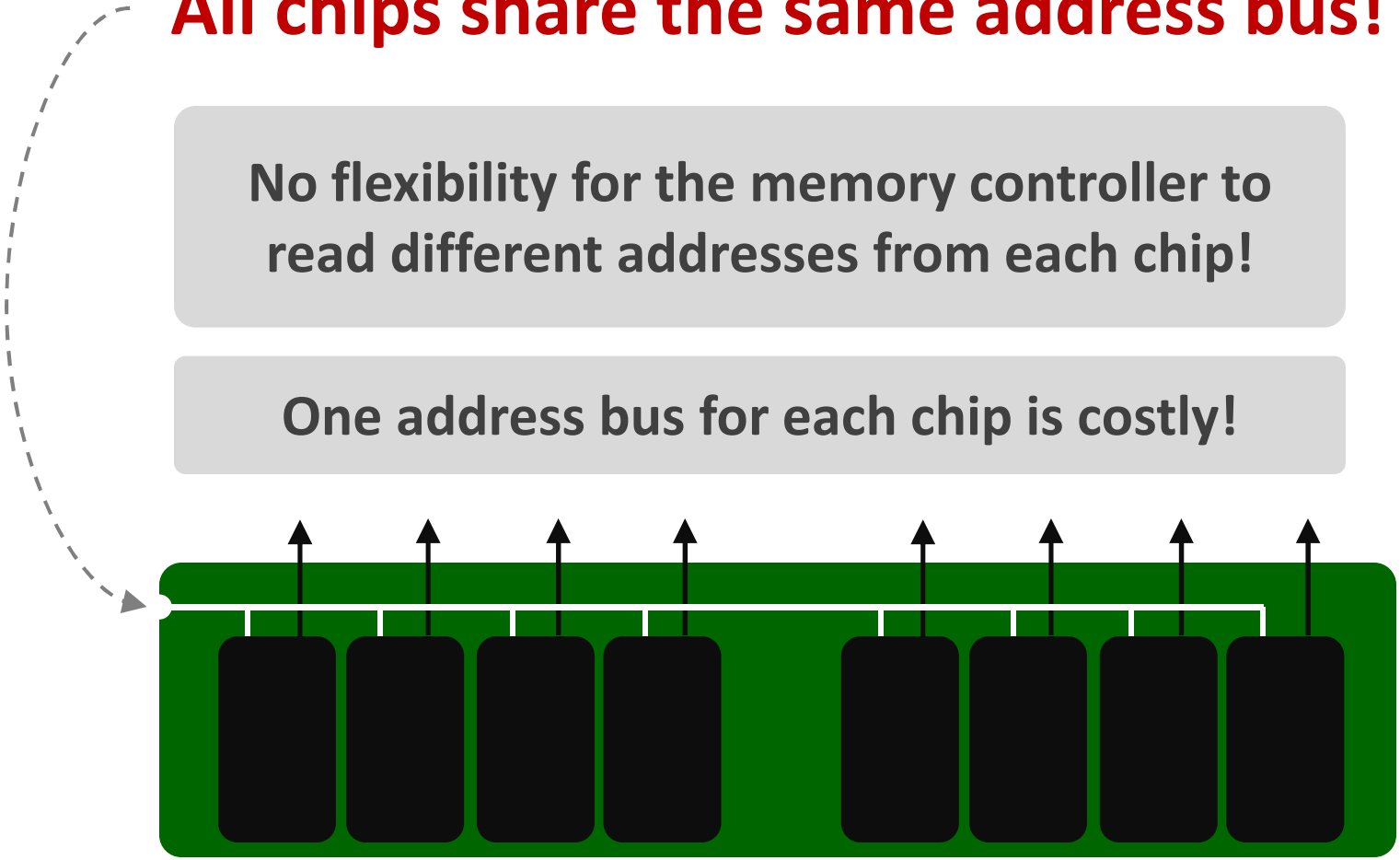


Challenge 2: Shared address bus

All chips share the same address bus!

No flexibility for the memory controller to read different addresses from each chip!

One address bus for each chip is costly!



Gather-Scatter DRAM

Challenge 1: Minimizing chip conflicts

Column-ID-based data shuffling

(shuffle data of each cache line differently)

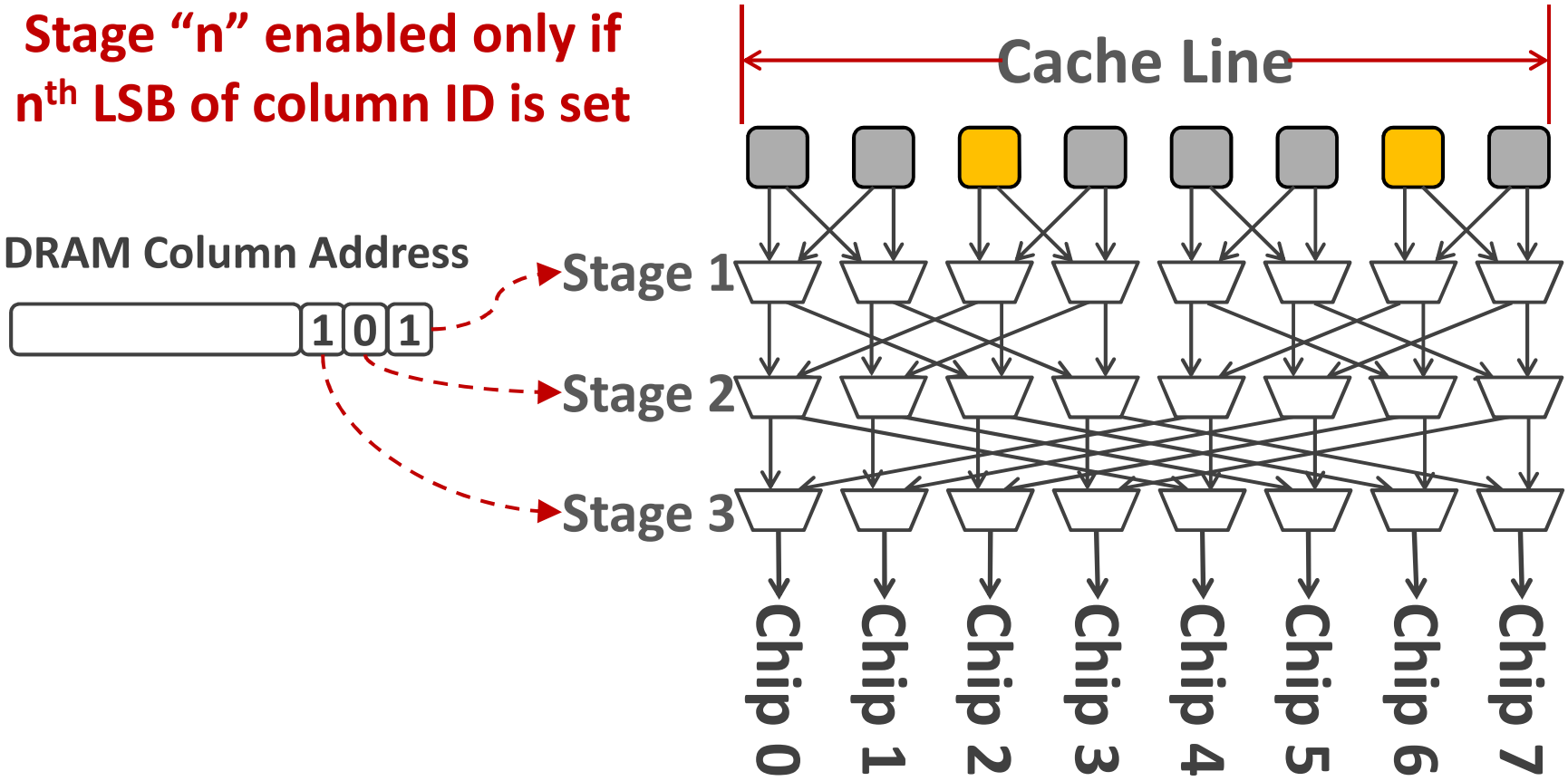
Challenge 2: Shared address bus

Pattern ID – In-DRAM address translation

(locally compute column address at each chip)

Column-ID-based data shuffling

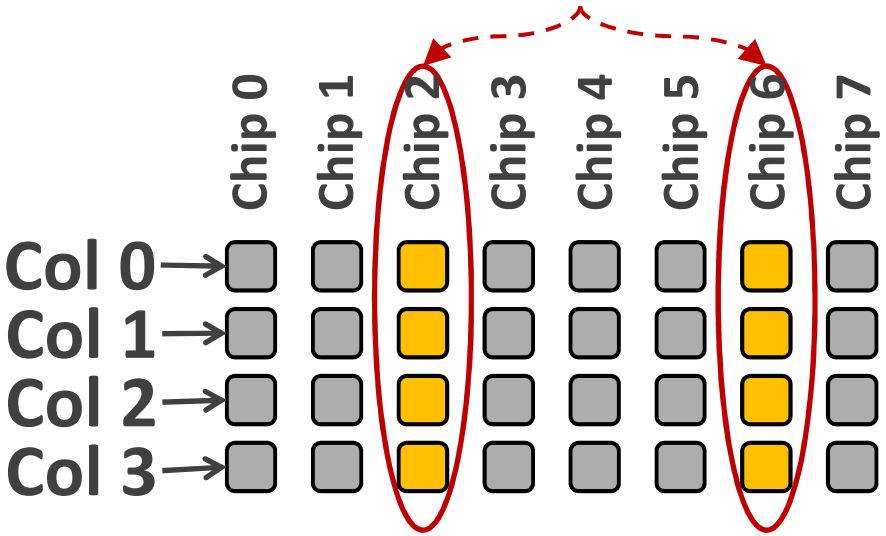
(implemented in the memory controller)



Effect of data shuffling

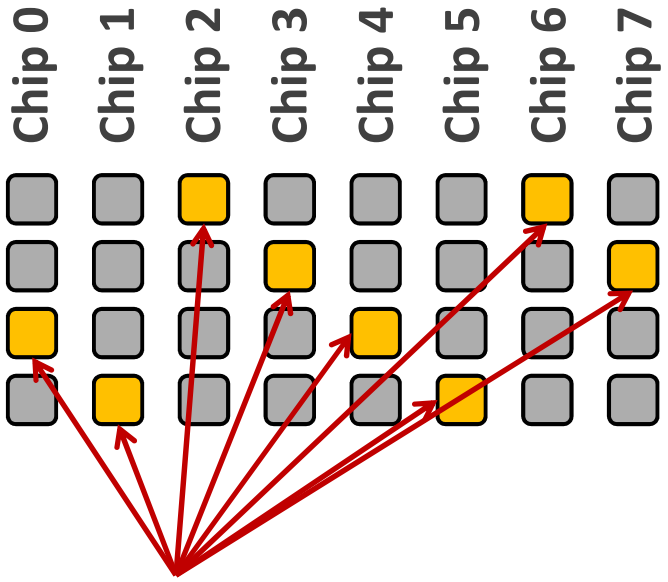
Before shuffling

Chip conflicts



After shuffling

Minimal chip conflicts!



Can be retrieved in a single command

Gather-Scatter DRAM

Challenge 1: Minimizing chip conflicts

Column-ID-based data shuffling

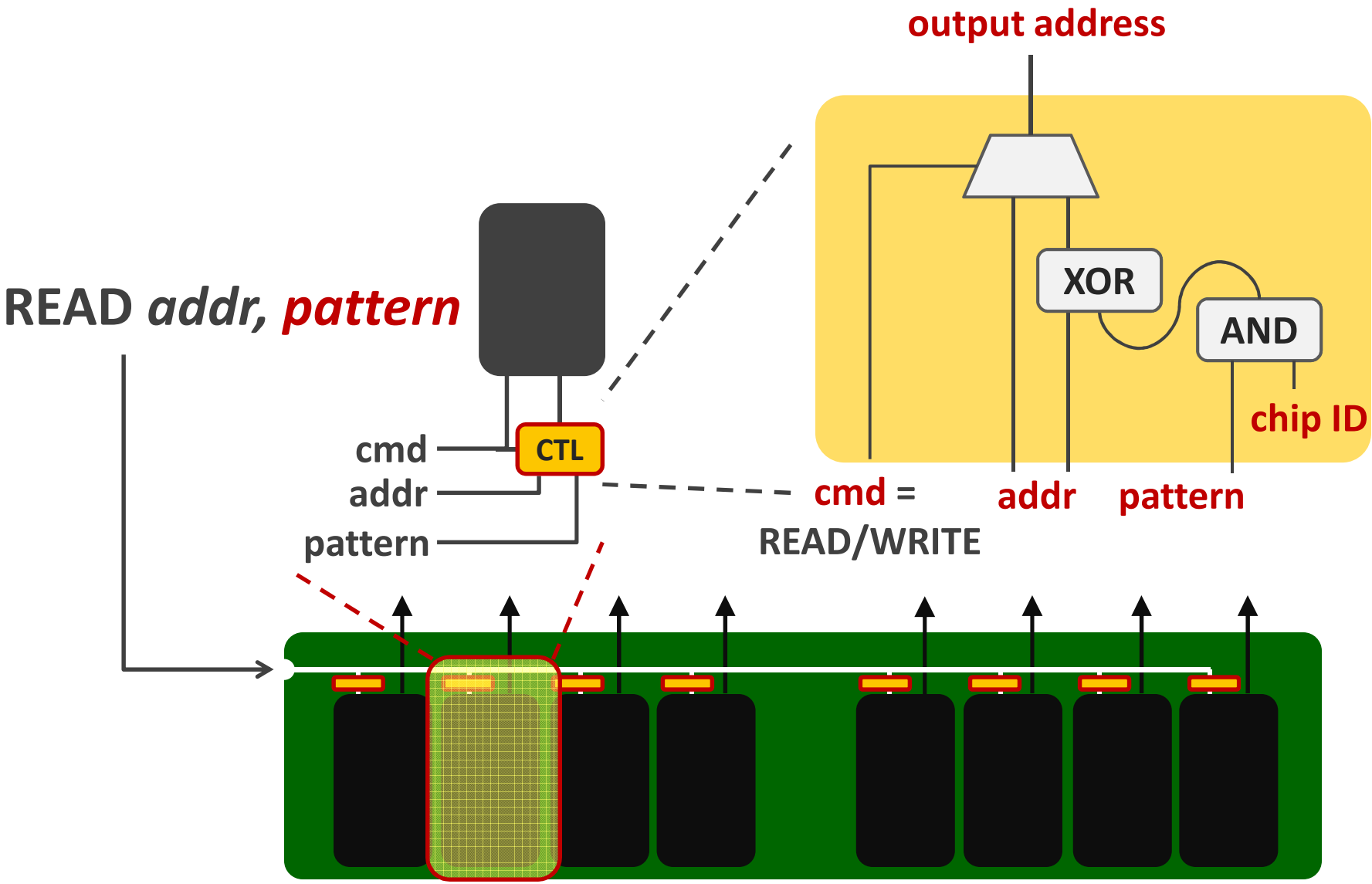
(shuffle data of each cache line differently)

Challenge 2: Shared address bus

Pattern ID – In-DRAM address translation

(locally compute the column address at each chip)

Per-chip column translation logic



Gather-Scatter DRAM (GS-DRAM)

32 values contiguously stored in DRAM (at the start of a DRAM row)

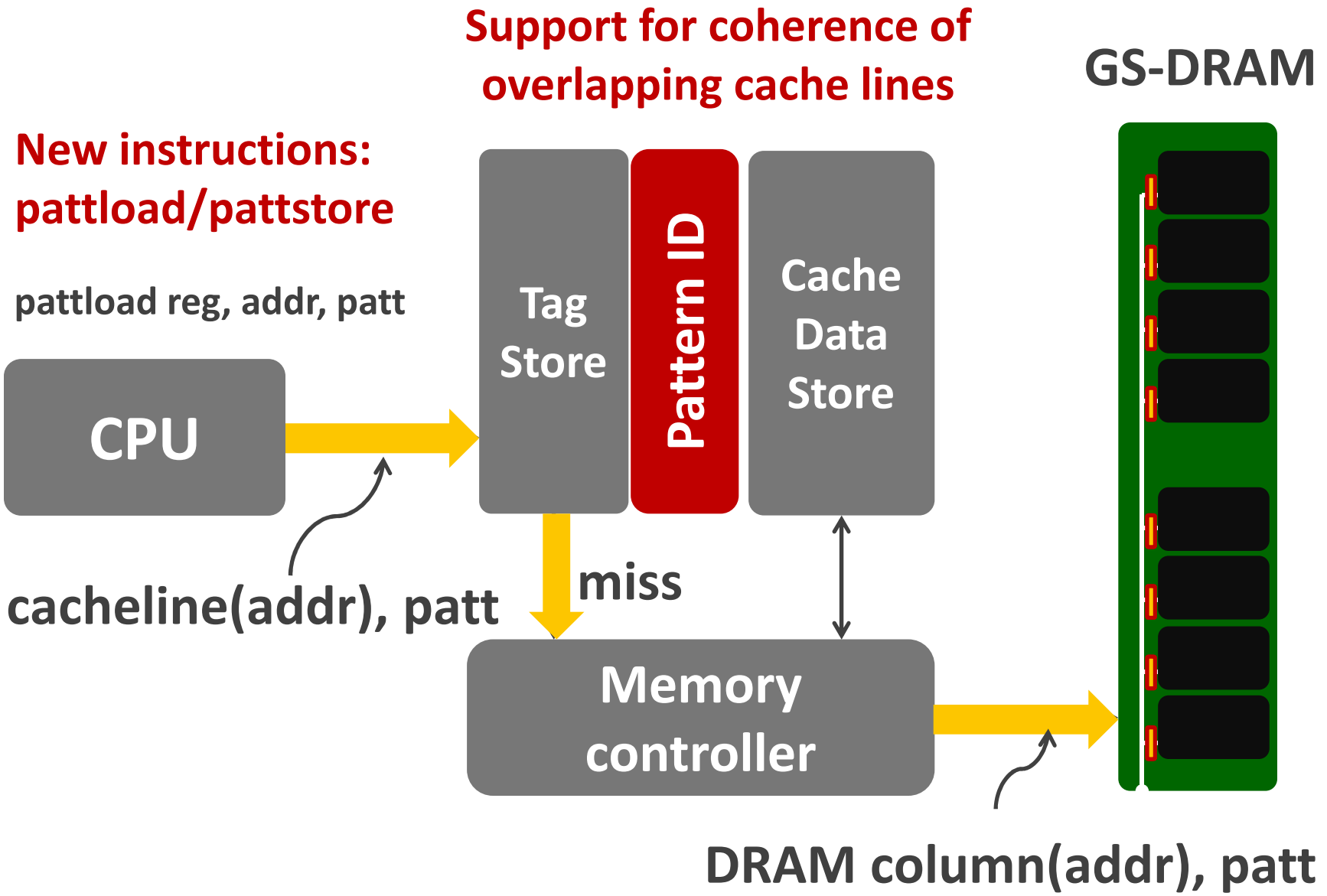
read addr **0**, pattern **0** (stride = 1, default operation)

read addr **0**, pattern **1** (stride = 2)

read addr **0**, pattern **3** (stride = 4)

read addr **0**, pattern **7** (stride = 8)

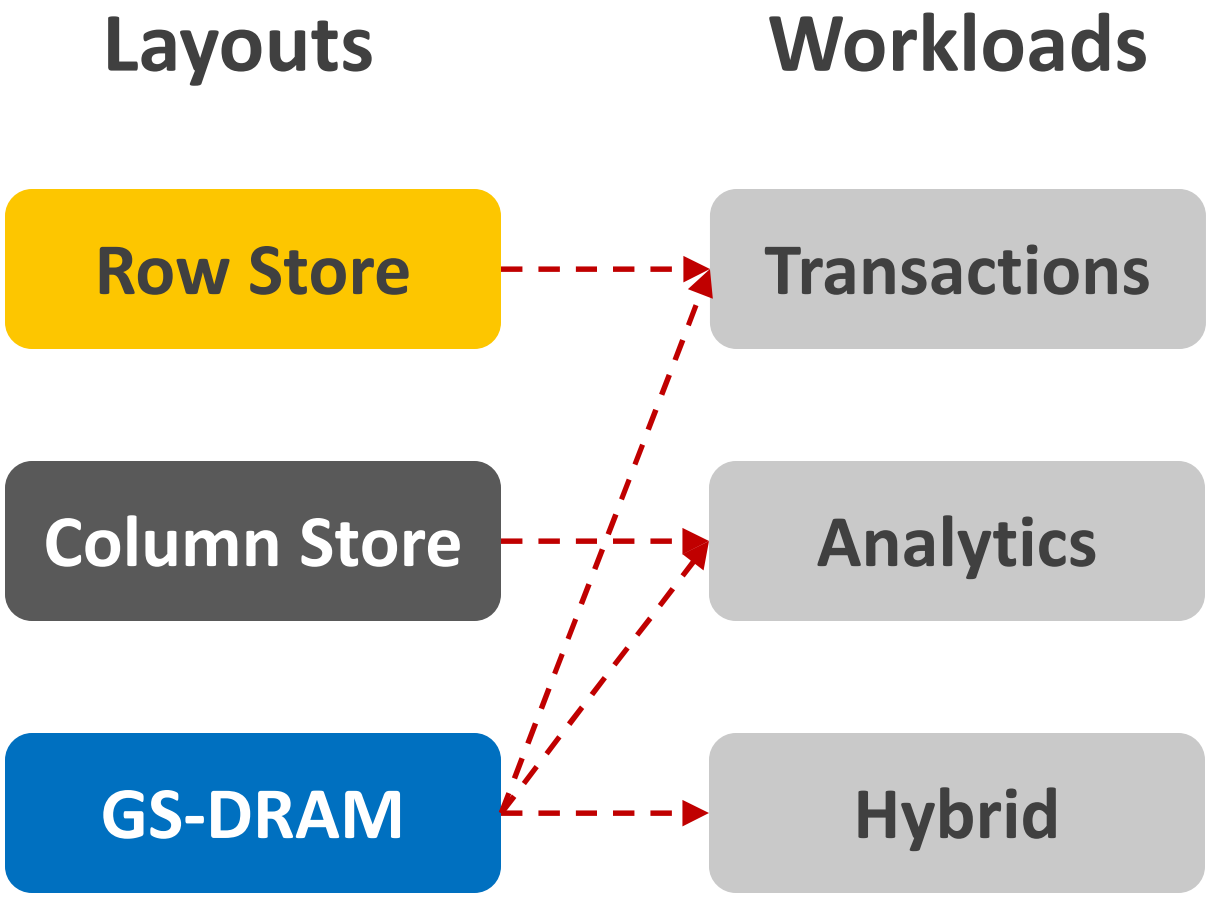
End-to-end system support for GS-DRAM

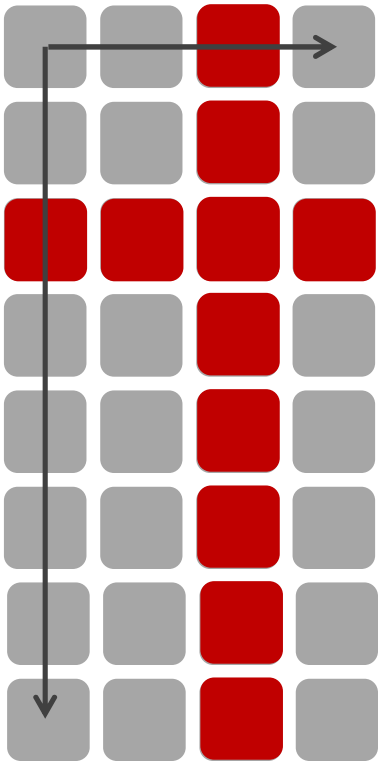


Methodology

- **Simulator**
  - Gem5 x86 simulator
  - Use “prefetch” instruction to implement pattern load
  - Cache hierarchy
    - 32KB L1 D/I cache, 2MB shared L2 cache
  - Main Memory: DDR3-1600, 1 channel, 1 rank, 8 banks
- **Energy evaluations**
  - McPAT + DRAMPower
- **Workloads**
  - In-memory databases
  - Matrix multiplication

In-memory databases

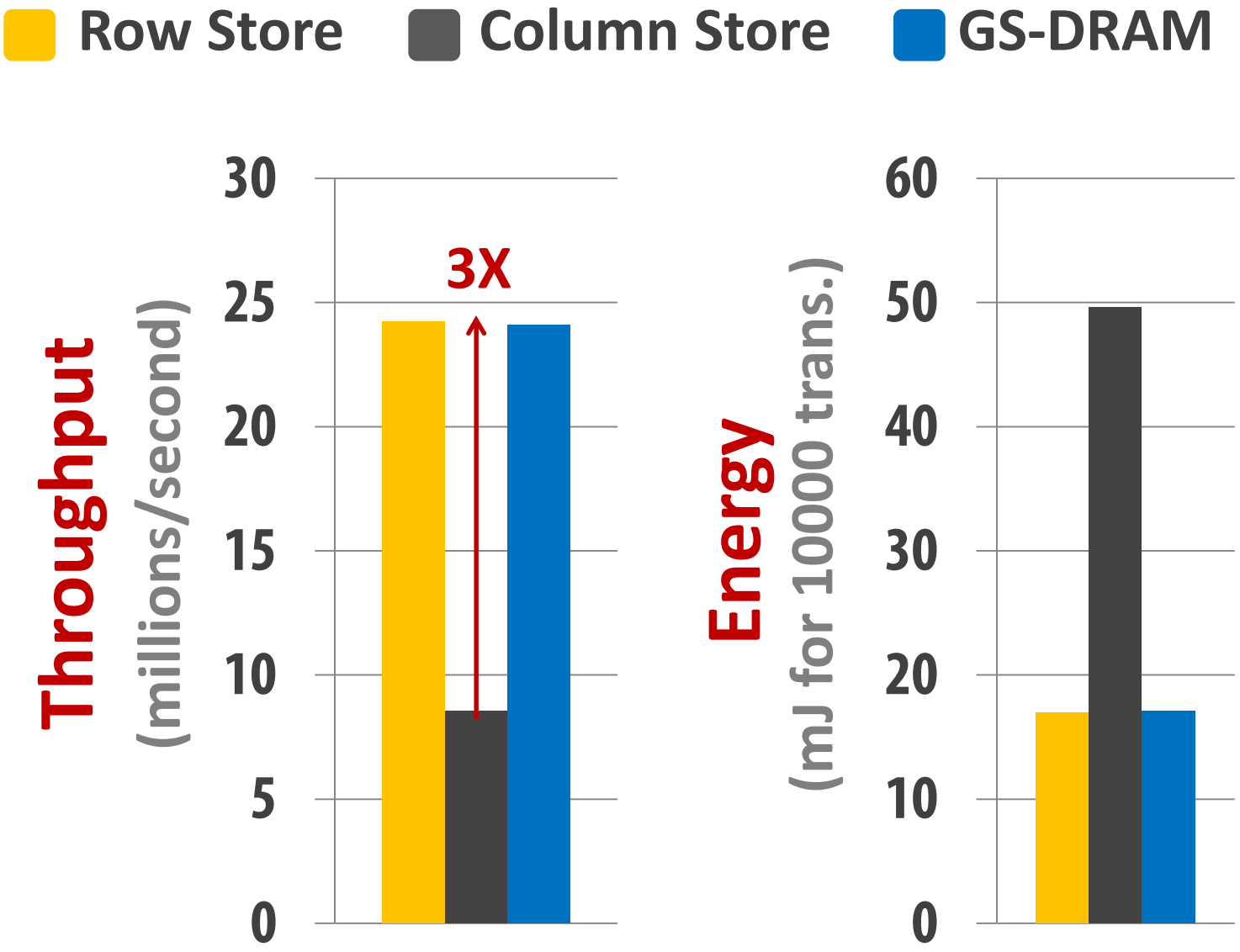




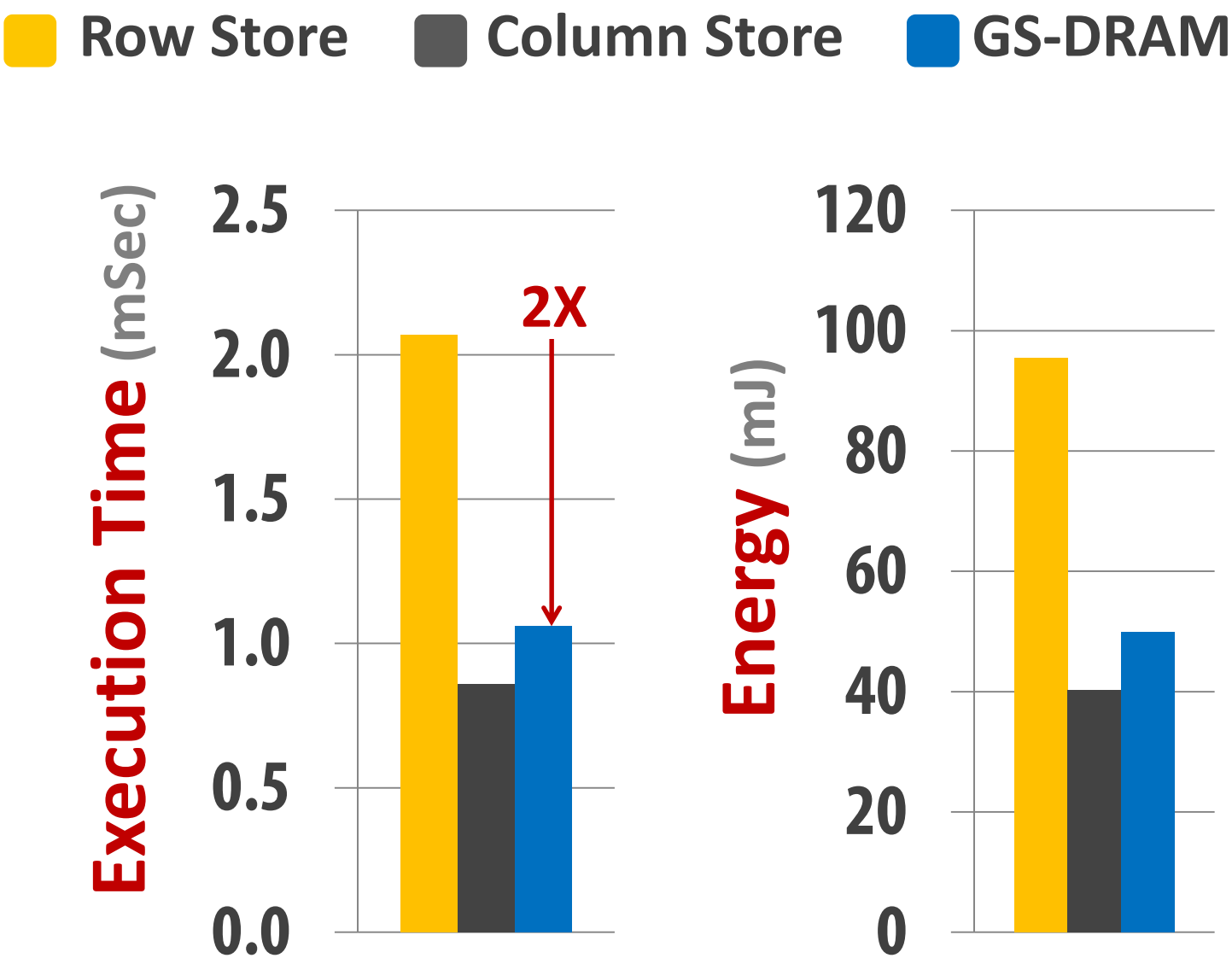
Workload

- **Database**
  - 1 table with million records
  - Each record = 1 cache line
- **Transactions**
  - Operate on a random record
  - Varying number of read-only/write-only/read-write fields
- **Analytics**
  - Sum of one/two columns
- **Hybrid**
  - Transactions thread: random records with 1 read-only, 1 write-only
  - Analytics thread: sum of one column

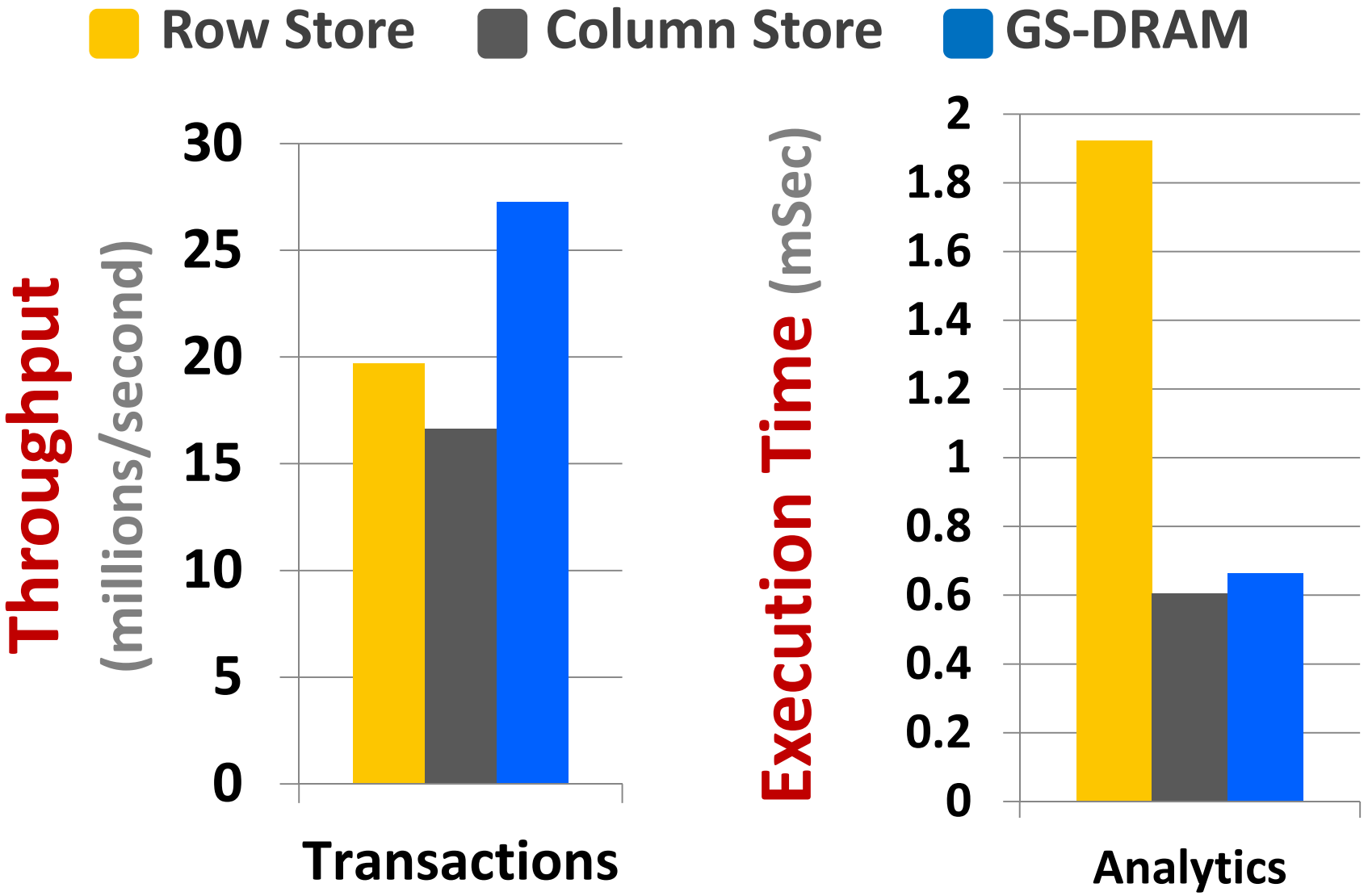
Transaction throughput and energy



Analytics performance and energy



Hybrid Transactions/Analytical Processing



Conclusion

- **Problem: Non-unit strided accesses**
  - Present in many applications
  - In-efficient in cache-line-optimized memory systems
- **Our Proposal: Gather-Scatter DRAM**
  - Gather/scatter values of strided access from multiple chips
  - Ideal memory bandwidth/cache utilization for power-of-2 strides
  - Low DRAM Cost: Logic to perform two bitwise operations per chip
- **Results**
  - In-memory databases: the best of both row store and column store
  - Many more applications: scientific computation, key-value stores

Gather-Scatter DRAM

In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses

Vivek Seshadri

Thomas Mullins, Amirali Boroumand, Onur Mutlu,

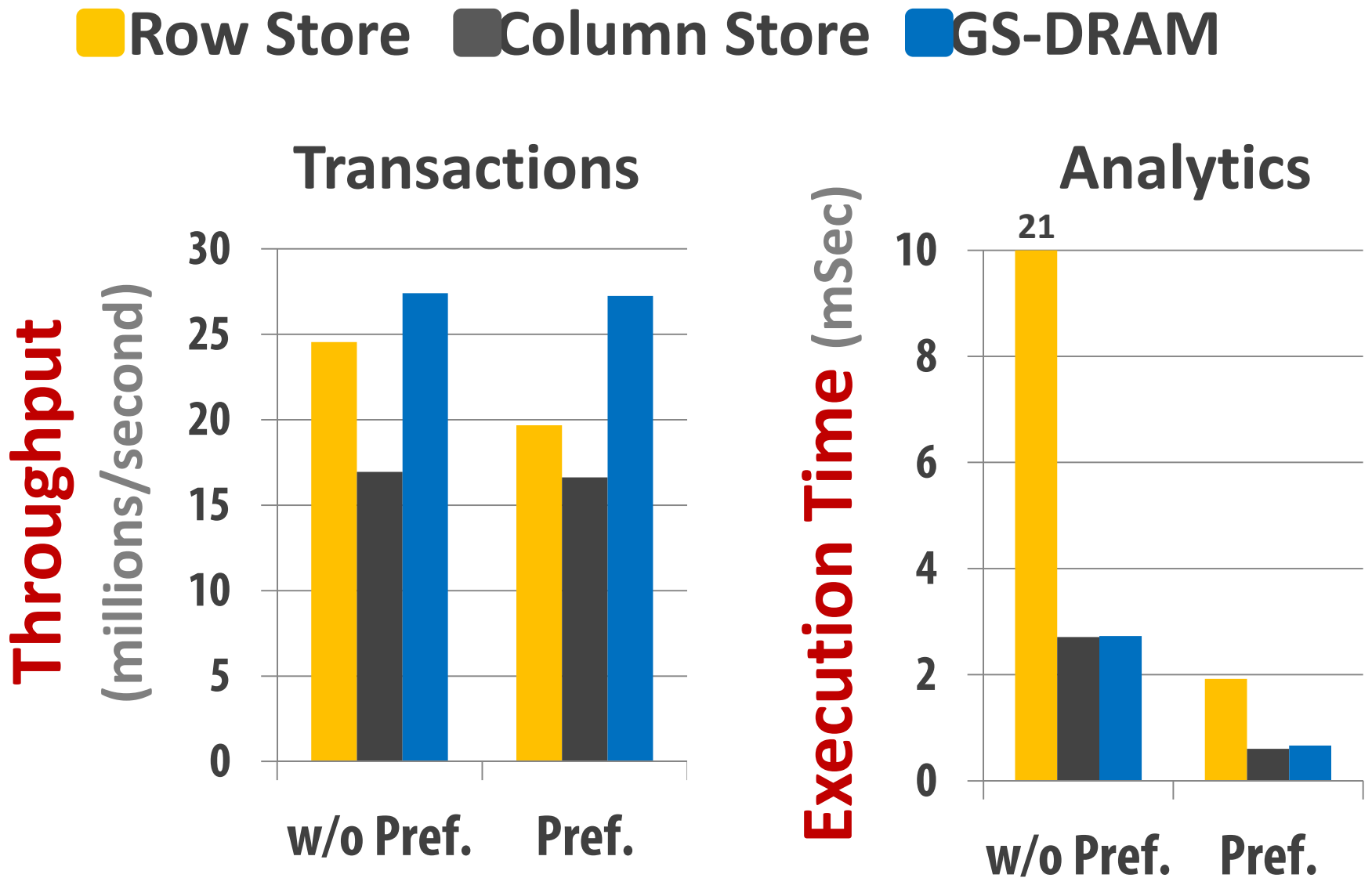
Phillip B. Gibbons, Michael A. Kozuch, Todd C. Mowry

Backup

Maintaining Cache Coherence

- **Restrict each data structure to only two patterns**
  - Default pattern
  - One additional strided pattern
- **Additional invalidations on read-exclusive requests**
  - Cache controller generates list of cache lines overlapping with modified cache line
  - Invalidates all overlapping cache lines

Hybrid Transactions/Analytical Processing



Transactions Results