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?

Gather-Scatter DRAM
(Power-of-2 strides)
DRAM modules have multiple chips.

All chips within a “rank” operate in unison!

Two Challenges!
Challenge 1: Chip conflicts

Data of each cache line is spread across all the chips!

Cache line 0

Cache line 1

Useful data mapped to only two 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)

Stage “n” enabled only if n\textsuperscript{th} LSB of column ID is set

DRAM Column Address

Chip 0, Chip 1, Chip 2, Chip 3, Chip 4, Chip 5, Chip 6, Chip 7

Stage 1, Stage 2, Stage 3

Cache Line
Effect of data shuffling

Before shuffling

Col 0 → Chip 2
Col 1 → Chip 3
Col 2 → Chip 5
Col 3 → Chip 6

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

READ \textit{addr, pattern}

\textit{chip ID}

\textit{output address}

\textit{XOR}

\textit{AND}

\textit{READ/WRITE}

\textit{cmd}

\textit{addr}

\textit{pattern}

\textit{cmd = READ/WRITE}
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

Support for coherence of overlapping cache lines

New instructions: pattload/pattstore

pattload reg, addr, patt

cacheline(addr), patt

miss

GS-DRAM

DRAM column(addr), patt
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

Layouts
- Row Store
- Column Store
- GS-DRAM

Workloads
- Transactions
- Analytics
- Hybrid
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

- **Row Store**
- **Column Store**
- **GS-DRAM**

**Throughput (millions/second):**
- Row Store: ~25
- GS-DRAM: ~30

**Energy (mJ for 10000 trans.):**
- Column Store: ~50
- GS-DRAM: ~15

GS-DRAM shows a 3X improvement in throughput compared to Column Store.
Analytics performance and energy

- **Row Store**
- **Column Store**
- **GS-DRAM**

**Execution Time (mSec)**
- 2X improvement

**Energy (mJ)**
Hybrid Transactions/Analytical Processing

Throughput (millions/second)

Transactions

Execution Time (mSec)

Analytics

Row Store  |  Column Store  |  GS-DRAM
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

Throughput (millions/second)

Transactions

Analytics

Execution Time (mSec)

Row Store  Column Store  GS-DRAM
Transactions Results

Execution time for 10000 trans.