Can DRAM Do More Than Just Store Data?

Vivek Seshadri
Final Year Ph.D. Student
Onur Mutlu, Todd C. Mowry
Phillip B. Gibbons, Michael A. Kozuch

http://www.istc-cc.cmu.edu/
Can DRAM do more?
Yes!
Copy/Init

Gather/Scatter
strided access patterns

Bitwise AND/OR
Outline of the Talk

1. Gather-Scatter DRAM
   – accelerating strided access patterns

2. RowClone
   – bulk data copy-initialization in DRAM

3. Bulk bitwise AND/OR in DRAM
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
Goal: Eliminate Inefficiency

Can we retrieve a cache line with only useful data?
DRAM Modules of 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

Column-ID-based Data Shuffling
(minimizing chip conflicts – for $2^n$ strides)

Pattern ID – In-DRAM Address Translation
(flexible column selection for different strides)
Column-ID-based Data Shuffling

Stage “n” enabled only if nth LSB of column ID is set

DRAM Column Address

Stage 1
Stage 2
Stage 3

Cache Line

Zero chip conflicts!
Gather-Scatter DRAM

Column-ID-based Data Shuffling
(minimizing chip conflicts – for $2^n$ strides)

Pattern ID – In-DRAM Address Translation
(flexible column selection for different strides)
Per-Chip Column Translation Logic

READ \textit{addr}, \textit{pattern}

\begin{align*}
\text{cmd} &= \text{READ/WRITE} \\
\text{addr} &= \text{output address} \\
\text{pattern} &= \text{XOR} \\
\text{chip ID} &= \text{AND}
\end{align*}
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)
Leveraging GS-DRAM

• On-chip Cache Support
  – identifying non-contiguous cache lines
  – maintaining cache coherence

• ISA support
  – new load/store instructions with pattern
  – x86 (new address mode with pattern)

• Software support
  – specify data structures (pages) that require shuffling
  – convey pattern ID to processor using new load/store instructions
Methodology

- Gem5 x86 simulator
- Use “prefetch” instruction to implement pattern load
- 32KB L1 D/I cache
- 2MB shared L2 cache
- Main Memory: DDR3-1600, 1 channel, 1 rank, 8 banks
- FR-FCFS scheduling policy
- GS-DRAM: support for stride of 8
Evaluation: In-memory Databases

Layouts
- Row Store
- Column Store
- GS-DRAM

Workloads
- Transactions
- Analytics
- Hybrid
Transaction Throughput and Energy

Throughput (millions/second)

Energy (mJ for 10000 trans.)

- Row Store
- Column Store
- GS-DRAM
Analytics Performance and Energy

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

Execution Time (mSec)

- 0.0
- 0.5
- 1.0
- 1.5
- 2.0
- 2.5

Energy (mJ)

- 0
- 20
- 40
- 60
- 80
- 100
- 120
Hybrid Transactions/Analytical Processing

Throughput (millions/second)

- Row Store
- Column Store
- GS-DRAM

Execution Time (mSec)

- w/o Pref.
- Pref.
Gather-Scatter DRAM: Summary

• Many data structures exhibit multiple access patterns
  — Only one access pattern has good spatial locality

• Gather-Scatter DRAM
  — memory controller gathers/scatters strided accesses
  — near ideal bandwidth/cache utilization for $2^n$ strides

• In-memory databases
  — GS-DRAM provides the best of both a row store and column store
Outline of the Talk

1. Gather-Scatter DRAM
   - accelerating strided access patterns

2. RowClone
   - bulk data copy/initialization in DRAM

3. Bulk bitwise AND/OR in DRAM
DRAM Cell Operation

- **wordline**
- **capacitor**
- **access transistor**
- **bitline**
- **enable**
- **Sense Amp**
DRAM Cell Operation

- **wordline**: Raise wordline
- **capacitor**: Cell loses charge to bitline
- **access transistor**: Sense amp
- **Sense Amp**: Deviation in bitline voltage
- **bitline**: Connects cell to bitline

\[ \frac{1}{2} V_{DD} + \delta \]
RowClone: In-DRAM Bulk Data Copy

activate source

activate destination

data gets copied source to destination

$V_{DD}$ $V_{DD} + \delta$

Sense Amp

enable sense amp
RowClone: Summary of Results

- **Bulk Row-to-Row Copy (8KB)**
  - 11X reduction in latency
  - 74X reduction in energy

- **8-core systems**
  - 27% performance
  - 17% memory energy efficiency
Outline of the Talk

1. Gather-Scatter DRAM
   – accelerating strided access patterns

2. RowClone
   – bulk data copy/initialization in DRAM

3. Bulk bitwise AND/OR in DRAM
Triple-Row Activation

activate all three rows

0

0

0

0

0

V_{DD} + \delta

enable sense amp

Sense Amp
Values of all cells get overwritten

Output = \( AB + BC + CA \)

\[ = C \ (A \ OR \ B) + \sim C \ (A \ AND \ B) \]
Bulk Bitwise AND/OR in DRAM

\[ \text{Result} = \text{row } A \ \text{AND/OR} \ \text{row } B \]

1. **Copy** data of \textit{row } A \ \text{to row } t1
2. **Copy** data of \textit{row } B \ \text{to row } t2
3. **Initialize** data of \textit{row } t3 \ \text{to } 0/1
4. **Activate** \textit{rows } t1/t2/t3 \ simultaneously
5. **Copy** data of \textit{row } t1/t2/t3 \ \text{to Result row}

Use RowClone to perform copy and initialization operations completely in DRAM!
Throughput of Bitwise Operations

- Intel-AVX (one core)
- In-DRAM (Cons.) (1 bank)
- In-DRAM (Aggr.) (1 bank)
- In-DRAM (Aggr.) (2 banks)

![Graph showing throughput of bitwise operations with different working set sizes and memory bank configurations.](image)
In-memory Bitmap Indices

- Predicates for records stored as bitmaps (e.g., age > 18)
- Bitwise operations to evaluate query conditions
- FastBit: real-world bitmap implementation
  - Bitwise OR: 33% of execution time for some queries
  - In-DRAM mechanism: 30% performance improvement
Summary

• DRAM can be more than a storage device

• In-DRAM techniques
  – Bulk data copy/initialization
  – Gather/Scatter strided access patterns
  – Bulk bitwise AND/OR operations

• High performance at low cost
  – Order of magnitude improvement in performance
  – Changes only to the peripheral logic
Can DRAM Do More Than Just Store Data?

Vivek Seshadri  
Final Year Ph.D. Student  
Onur Mutlu, Todd C. Mowry  
Phillip B. Gibbons, Michael A. Kozuch

http://www.istc-cc.cmu.edu/