FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang[1], **Lois Orosa**[2], Xiangjun Peng[3,1], Yang Guo[1], Saugata Ghose[4,5], Minesh Patel[2], Jeremie S. Kim[2], Juan Gómez Luna[2], Mohammad Sadrosadati[6], Nika Mansouri Ghiasi[2], Onur Mutlu[2,5]

1 National University of Defense Technology

2 ETH zürich

3 香港中文大學 The Chinese University of Hong Kong

4 University of Illinois Urbana-Champaign

5 Carnegie Mellon University

6 IPM Institute for Research in Fundamental Sciences

SAFARI

MICRO 2020

Executive Summary

- **Problem:** DRAM latency is a **performance bottleneck** for many applications
- **Goal: Reduce DRAM latency** via in-DRAM cache
- **Existing in-DRAM caches:**
  - Augment DRAM with **small-but-fast regions** to implement caches
  - **Coarse-grained** (i.e., multi-kB) in-DRAM data relocation
  - Relocation **latency increases** with **physical distance** between slow and fast regions
- **FIGARO Substrate:**
  - **Key idea:** use the **existing shared global row buffer** among subarrays within a DRAM bank **to provide** support for in-DRAM **data relocation**
  - **Fine-grained** (i.e., multi-byte) in-DRAM data relocation and **distance-independent** relocation latency
  - Avoids complex modifications to DRAM by using **existing structures**
- **FIGCache:**
  - **Key idea:** cache **only small, frequently-accessed portions of different DRAM rows** in a designated region of DRAM
  - Caches only the **parts of each row** that are expected to be accessed in the **near future**
  - **Increases row hits** by packing more **frequently-accessed** data into FIGCache
  - Improves system **performance** by **16.3%** on average
  - Reduces **DRAM energy** by **7.8%** on average
- **Conclusion:**
  - FIGARO enables **fine-grained data relocation** in-DRAM at low cost
  - FIGCache **outperforms state-of-the-art coarse-grained** in-DRAM caches

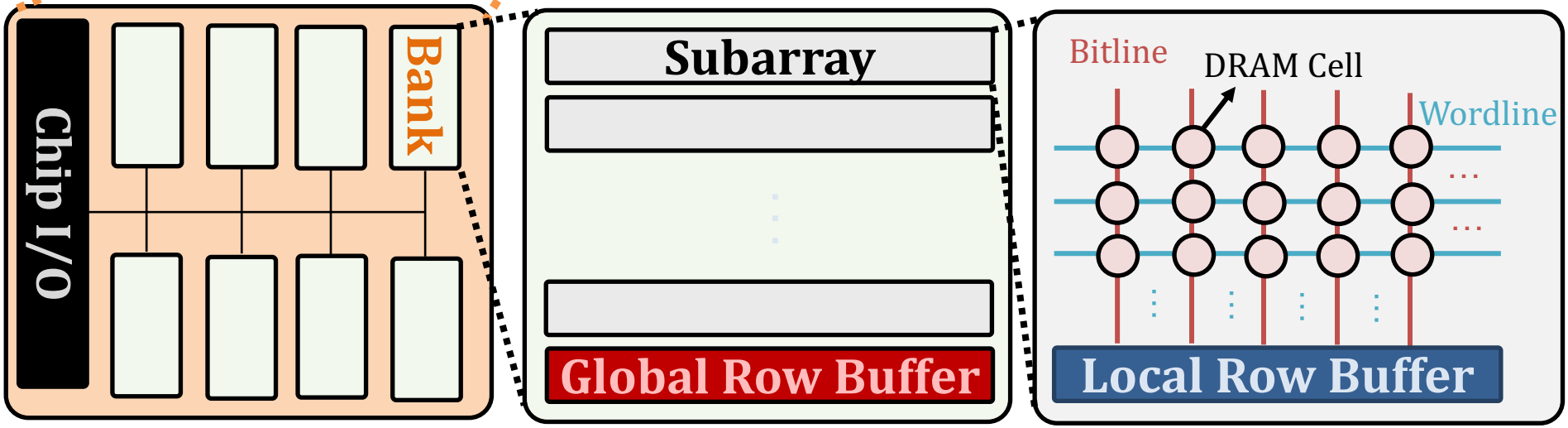
Outline

- Background
- Existing In-DRAM Cache Designs
- FIGARO Substrate
- FIGCache: Fine-Grained In-DRAM Cache
- Experimental Methodology
- Evaluation
- Conclusion

Outline

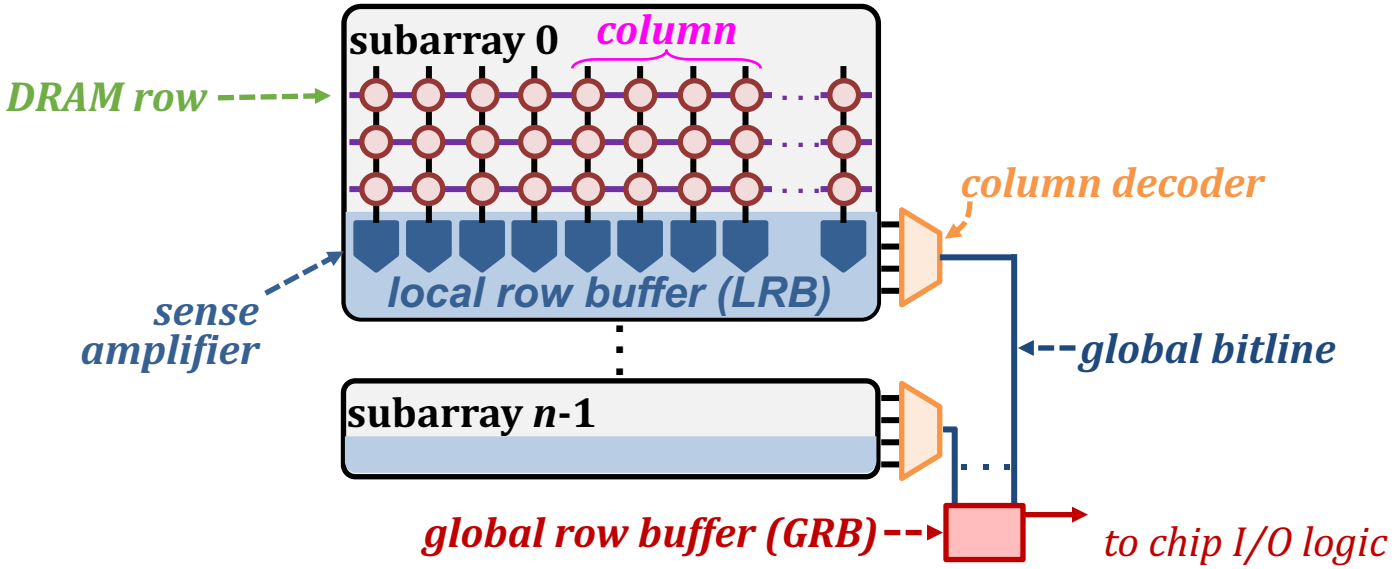
- **Background**
- Existing In-DRAM Cache Designs
- FIGARO Substrate
- FIGCache: Fine-Grained In-DRAM Cache
- Experimental Methodology
- Evaluation
- Conclusion

DRAM Organization



DRAM Chip *DRAM Bank* *DRAM Subarray*

Bank and Subarray Organization



- Each subarray contains 512– 2048 rows of DRAM cells
  - DRAM rows are connected to a local row buffer (LRB)
- All of the LRBs in a bank are connected to a shared global row buffer (GRB)
- The GRB is connected to the LRBs using a set of global bitlines
- The GRB has smaller width (e.g., 8B) than the LRBs (e.g., 1kB)
- A single column (i.e., a small number of bits) of the LRB is selected using a column decoder to connect to the GRB

DRAM Operation

DRAM controller issues **4** main **memory commands**

1) **ACTIVATE**: activates the DRAM row containing the data
   - Latches the selected DRAM row into the LRB

2) **PRECHARGE**: prepares all bitlines for a subsequent ACTIVATE command to a different row

3) **READ**: reads a column of data
   - One column of the LRB is selected using the column decoder
   - The GRB then drives the data to the chip I/O logic

4) **WRITE**: writes a column of data into DRAM

Outline

- Background
- **Existing In-DRAM Cache Designs**
- FIGARO Substrate
- FIGCache: Fine-Grained In-DRAM Cache
- Experimental Methodology
- Evaluation
- Conclusion

In-DRAM Cache Design

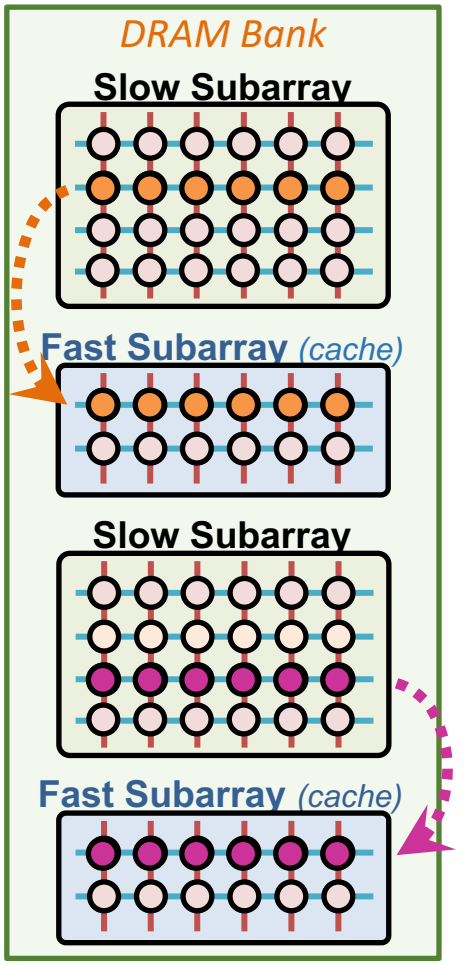
- **Key idea:** Introduce heterogeneity in DRAM

- **Slow subarrays**
  - Same latency and capacity as regular (i.e., slow) DRAM

- **Fast subarrays (shorter bitlines)**
  - Fast access latency and small capacity
  - Used as in-DRAM cache for hot data
  - Many fast subarrays interleaved among slow subarrays
  - Inclusive cache

- **Data relocation between normal and fast subarrays**
  - DRAM row granularity (multi-kilobyte)
  - Relocation latency increases as the physical relocation distance increases

Common in-DRAM cache organization



Inefficiencies of In-DRAM Caches

1) Coarse-grained:
Caching an entire row at a time
hinders the potential of in-DRAM cache

2) Area overhead and complexity:
Many fast subarrays interleaved
among normal subarrays

Outline

- Background
- Existing In-DRAM Cache Designs
- **FIGARO Substrate**
- FIGCache: Fine-Grained In-DRAM Cache
- Experimental Methodology
- Evaluation
- Conclusion

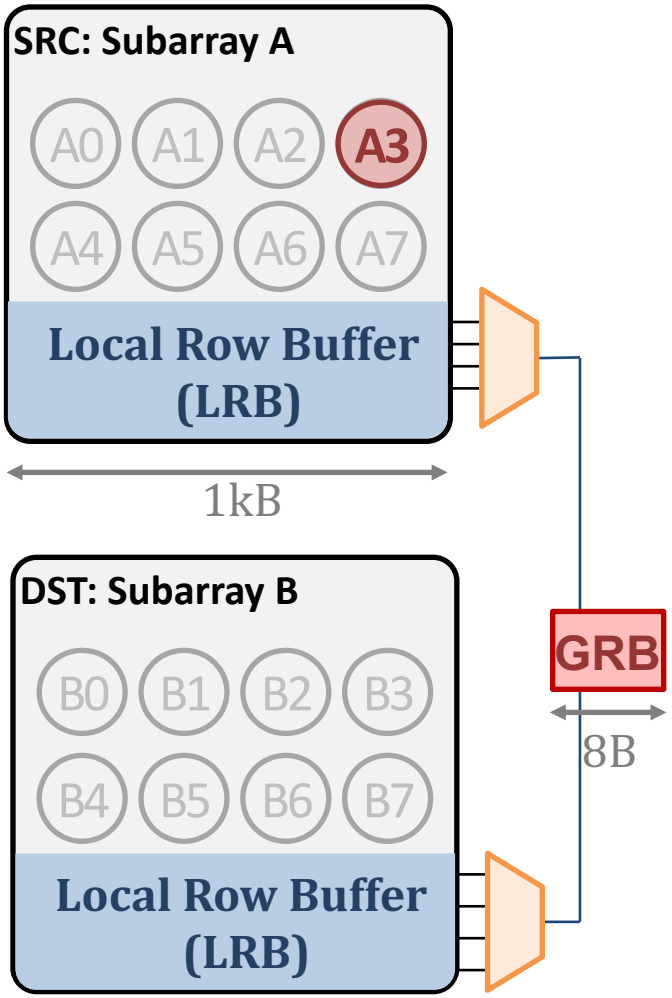
Observations and Key Idea

- **Observations:**

1) All local row buffers (LRBs) in a bank are connected to a single shared global row buffer (GRB)

2) The GRB has smaller width (e.g., 8B) than the LRBs (e.g., 1kB)

- **Key Idea:** use the existing shared GRB among subarrays within a DRAM bank to perform fine-grained in-DRAM data relocation



FIGARO Overview

FIGARO: Fine-Grained In-DRAM Data Relocation Substrate

- Relocates data across subarrays within a bank
- Column granularity within a chip
  - Cache-block granularity within a rank
- New RELOC command to relocate data between LRBs of different subarrays via the GRB

Transferring Data via FIGARO

❶ ACTIVATE subarray A

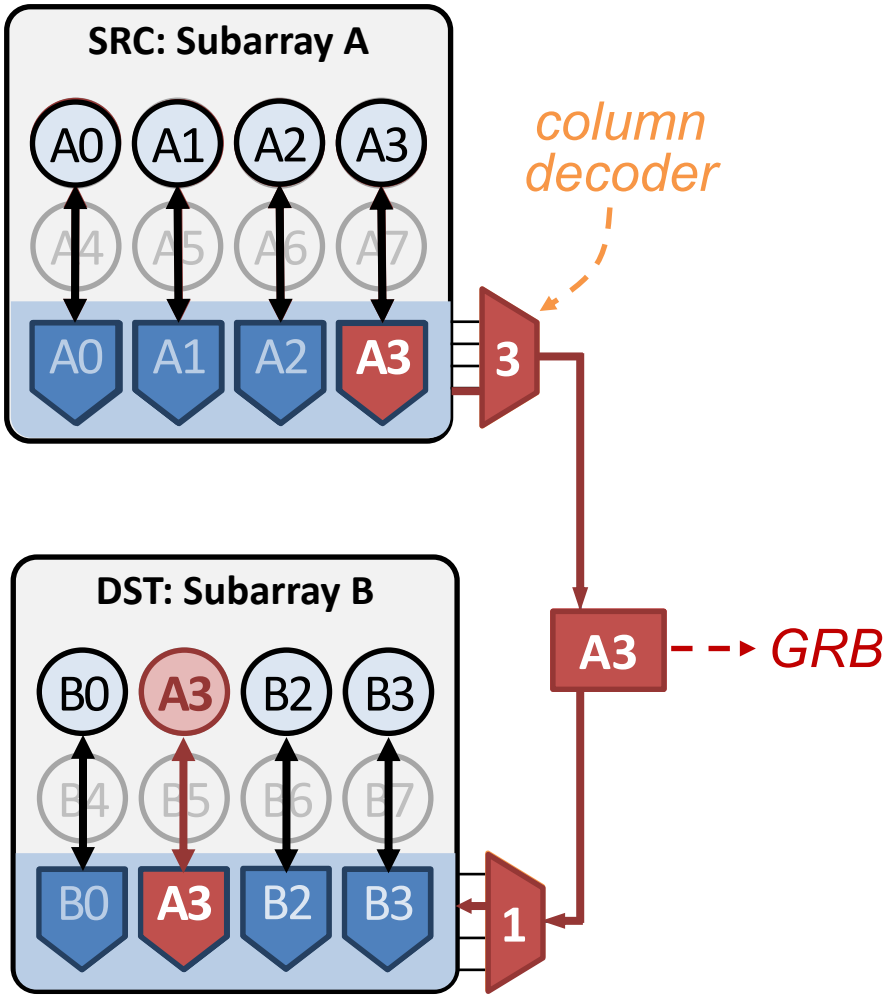
- Copies data from row to LRB

❷ RELOC A col 3 → B col 1

- Selects Column 3 in Subarray A
- Loads A3 into the GRB
- Selects Column 1 in Subarray B

❸ ACTIVATE subarray B

- Overwrites A3 in column 1



Key Features of FIGARO

- **Fine-grained:** column/cache-block level data relocation
- **Distance-independent latency**
  - The relocation latency depends on the length of global bitline
  - Similar to the latency of read/write commands
- **Low overhead**
  - Additional column address MUX, row address MUX, and row address latch per subarray
  - 0.3% DRAM chip area overhead
- **Low latency and low energy consumption**
  - Low latency (63.5ns) to relocate one column
    - » Two ACTIVATEs, one RELOC, and one PRECHARGE commands
  - Low energy consumption (0.03uJ) to relocate one column

More FIGARO Details in the Paper

- Enabling Unaligned Data Relocation
- Circuit-level Operation and Timing
- SPICE Simulations
- Other Use Cases for FIGARO

Outline

Background

Existing In-DRAM Cache Designs

FIGARO Substrate

FIGCache: Fine-Grained In-DRAM Cache

Experimental Methodology

Evaluation

Conclusion

FIGCache Overview

- **Key idea:** Cache only small, frequently-accessed portions of different DRAM rows in a designated region of DRAM

- **FIGCache** (Fine-Grained In-DRAM Cache)
  - Uses FIGARO to relocate data into and out of the cache at the fine granularity of a row segment
  - Avoids the need for a large number of fast (yet low capacity) subarrays interleaved among slow subarrays
  - Increases row buffer hit rate

- **FIGCache Tag Store (FTS)**
  - Stores information about which row segments are currently cached
  - Placed in the memory controller

- **FIGCache In-DRAM Cache Designs**
  - Using 1) fast subarrays, 2) slow subarrays, or 3) fast rows in a subarray

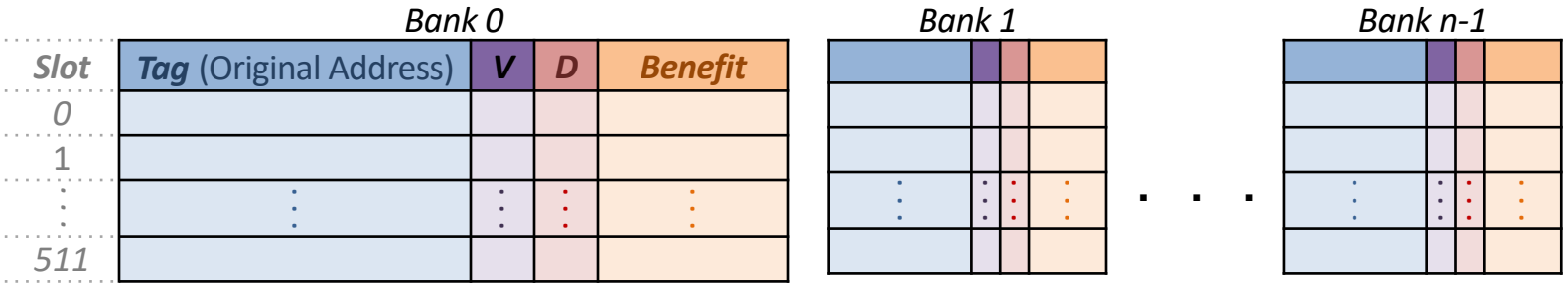
Benefits of FIGCache

Fine-grained (cache-block) caching granularity

Low area overhead and manufacturing complexity

FIGCache Tag Store (FTS)

- The memory controller stores information about which row segments are in cache
- Fully associative cache
- FIGCache Tag Store (FTS)



- *Benefit*: Benefit counter (used for cache replacement)

Insertion and Replacement

- FIGCache insertion policy
  - *Insert-any-miss:* every FIGCache miss triggers a row segment relocation into the cache

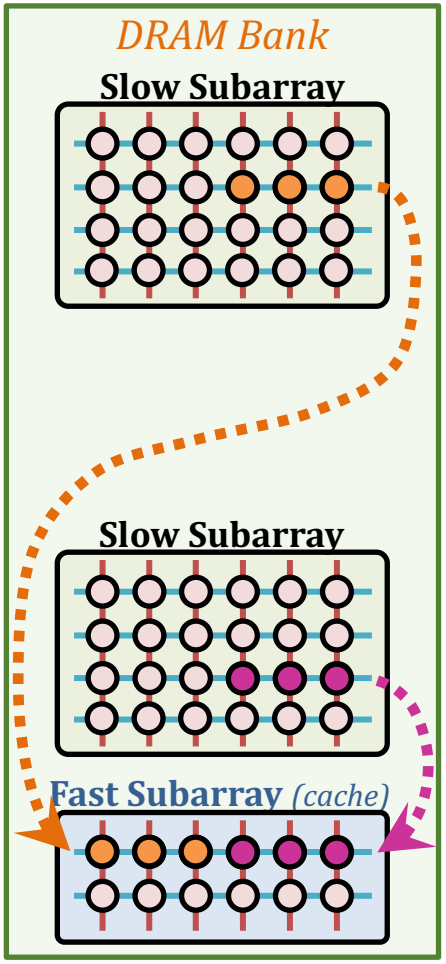
- FIGCache replacement policy
  1) Sum all *Benefit* values from all segments of the row
  2) The row with least *Benefit* is selected for eviction

  - Row granularity replacement

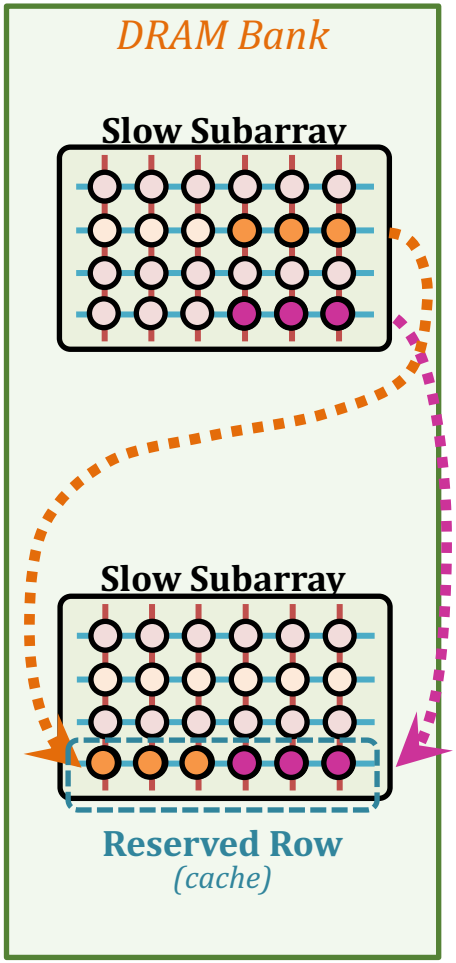
- We experimentally find that FIGCache insertion and replacement policies are rather effective

FIGCache Designs

FIGCache using
Fast Subarrays



FIGCache using
Slow Subarrays
(i.e., existing DRAM chips)



Outline

- Background
- Existing In-DRAM Cache Designs
- FIGARO Substrate
- FIGCache: Fine-Grained In-DRAM Cache
- **Experimental Methodology**
- Evaluation
- Conclusion

Experimental Methodology

- **Simulator**
  - Ramulator open-source DRAM simulator [Kim+, CAL'15] [https://github.com/CMU-SAFARI/ramulator]
  - Energy model: McPAT, CACTI, Orion 3.0, and DRAMPower
- **System configuration**
  - 8 cores, 3-wide issue, 256-entry instruction window
  - L1 4-way 64KB, L2 8-way 256KB, L3 LLC 16-way 2MB per core
  - DRAM DDR4 800MHz bus frequency
- **FIGCache default parameters**
  - Row segment size: 1/8th of a DRAM row (16 cache blocks)
  - Fast subarray reduces tRCD by 45.5%, tRP by 38.2%, and tRAS by 62.9%
  - In-DRAM cache size: 64 rows per bank
- **Workloads**
  - 20 eight-core multiprogrammed workloads from SPEC CPU2006, TPC, BioBench, Memory Scheduling Championship

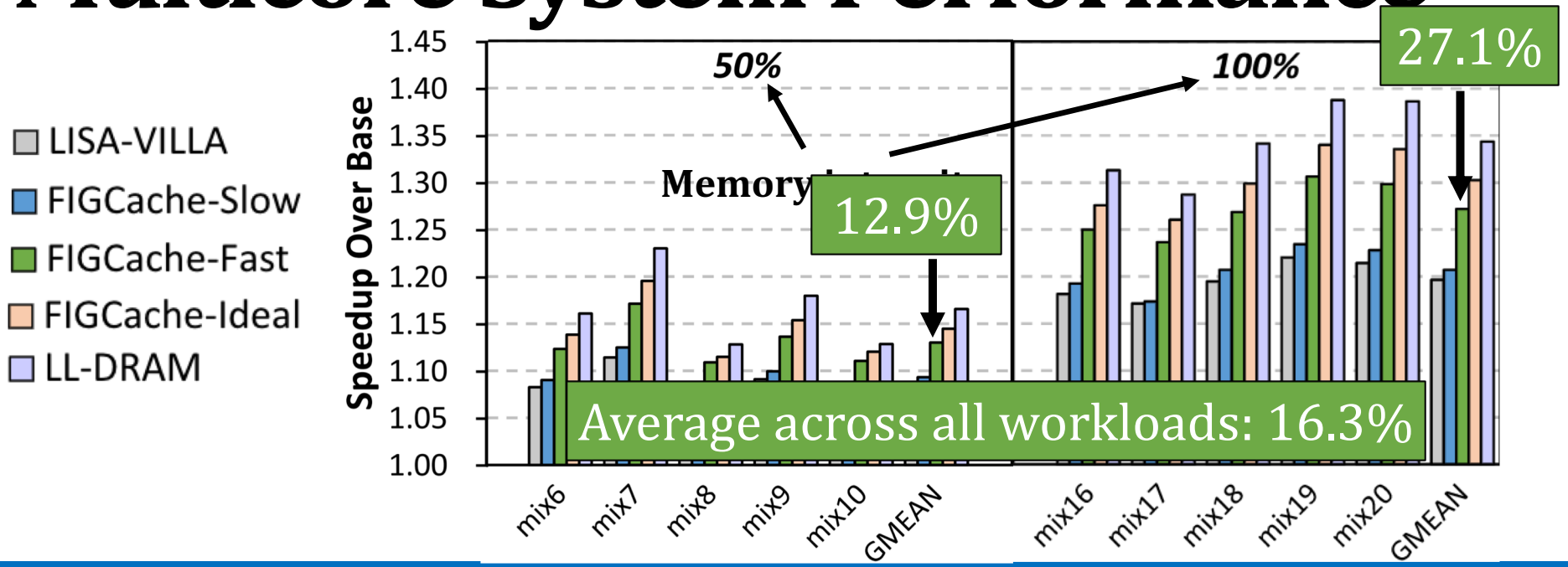
Comparison Points

- **Baseline:** conventional DDR4 DRAM
- **LISA-VILLA:** State-of-the-art in-DRAM Cache [Chang+, HPCA '16]
- **FIGCache-slow:** Our in-DRAM cache with cache rows stored in slow subarrays
- **FIGCache-fast:** Our in-DRAM cache with cache rows stored in fast subarrays
- **FIGCache-ideal:** An unrealistic version of FIGCache-Fast where the row segment relocation latency is zero
- **LL-DRAM:** System where all subarrays are fast

Outline

- Background
- Existing In-DRAM Cache Designs
- FIGARO Substrate
- FIGCache: Fine-Grained In-DRAM Cache
- Experimental Methodology
- **Evaluation**
- Conclusion

Multicore System Performance

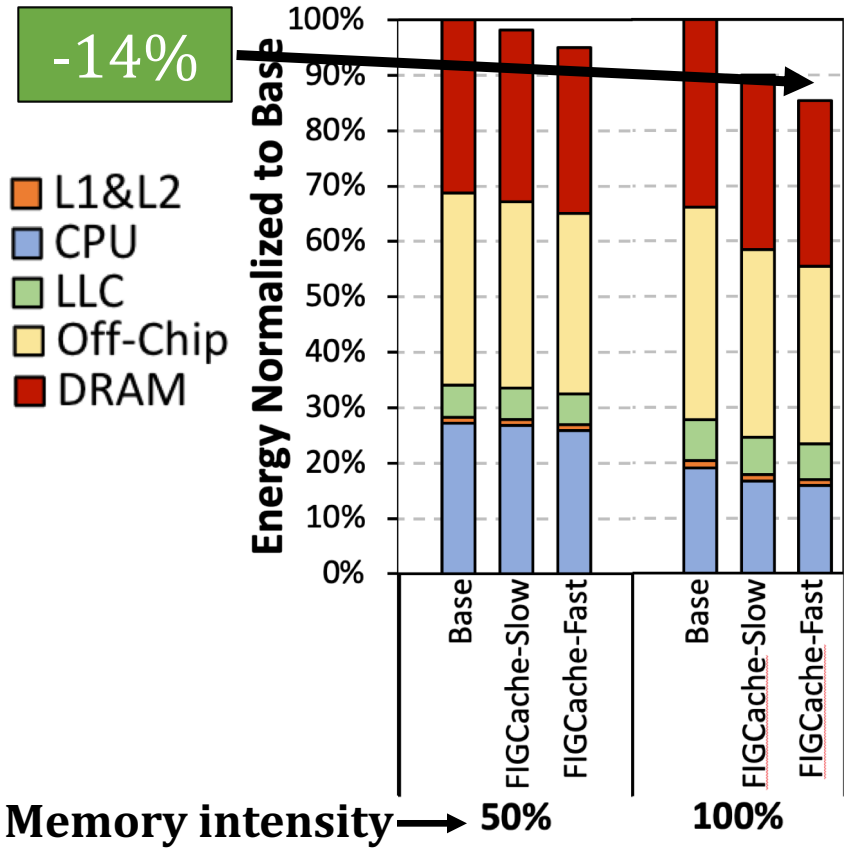


The benefits of FIGCache-Fast and FIGCache-Slow increase as workload memory intensity increases

Both FIGCache-slow and FIGCache-fast outperform LISA-VILLA

FIGCache-Fast approaches the ideal performance improvement of both FIGCache-Ideal and LL-DRAM

Multicore System Energy Savings

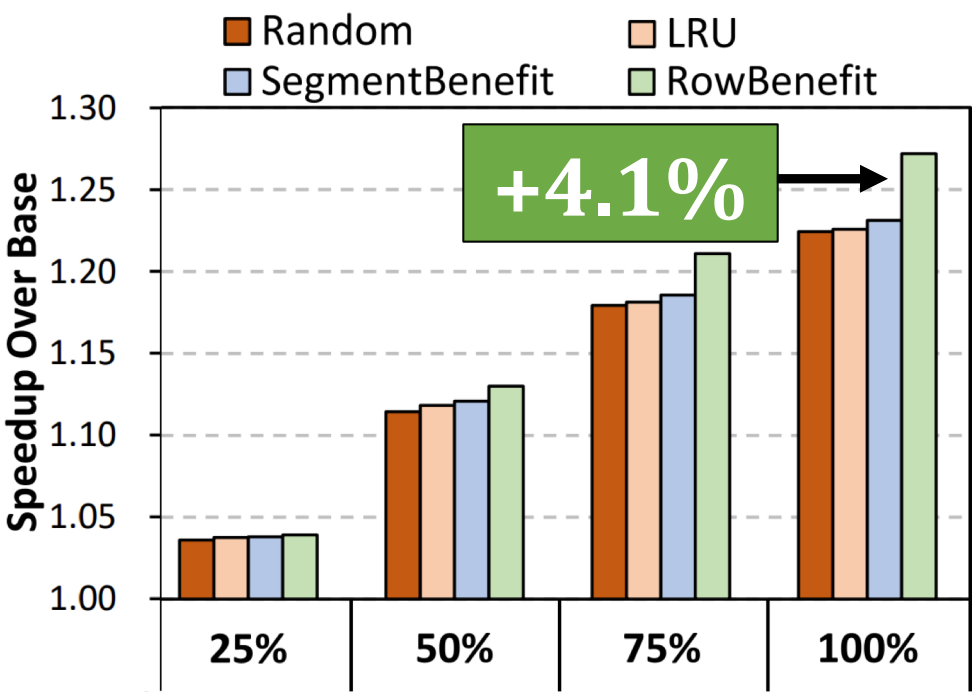


- **FIGCache-Slow** and **FIGCache-Fast** consume less energy than Base

- **Energy reduction** comes from:
  1) Improved DRAM row buffer hit rate
  2) Reduced execution time that saves static energy across each component

FIGCache is effective at reducing system energy consumption

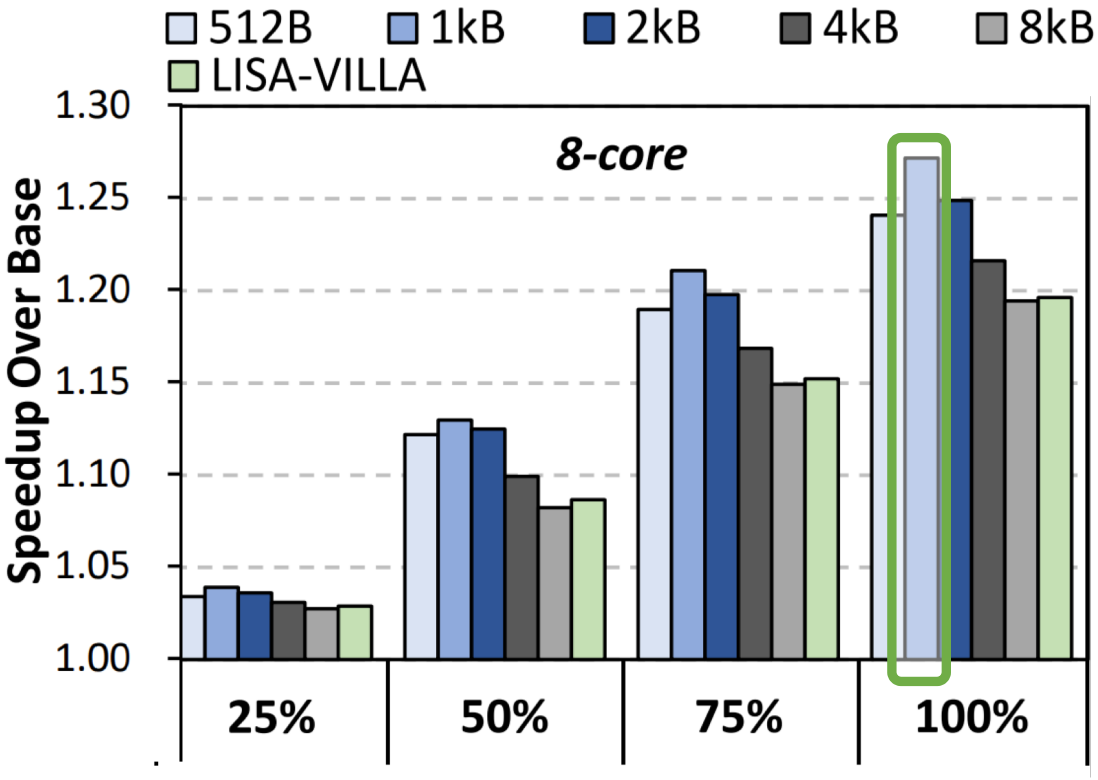
FIGCache Replacement Policies



- **RowBenefit**: FIGCache replacement policy
- **SegmentBenefit**: traditional benefit-based policy [Lee+,HPCA '13]
- **Observations:**
  1) FIGCache outperforms Base with all replacement policies
  2) RowBenefit outperforms all the other policies

RowBenefit replacement policy is effective at capturing temporal locality

Different Row Segment Sizes



FIGCache 1KB
is the best configuration

Performance highly depends on the row segment size

More Results in the Paper

- More Detailed Results
- Single-core Workloads
- In-DRAM Cache Hit Rate
- DRAM Row Buffer Hit Rate
- Performance with Different Row Segment Insertion Thresholds
- Performance with Different Cache Capacities

Outline

- Background
- Existing In-DRAM Cache Designs
- FIGARO Substrate
- FIGCache: Fine-Grained In-DRAM Cache
- Experimental Methodology
- Evaluation
- **Conclusion**

Executive Summary

- **Problem:** DRAM latency is a **performance bottleneck** for many applications
- **Goal: Reduce DRAM latency** via in-DRAM cache
- **Existing in-DRAM caches:**
  - Augment DRAM with **small-but-fast regions** to implement caches
  - **Coarse-grained** (i.e., multi-kB) in-DRAM data relocation
  - Relocation **latency increases** with **physical distance** between slow and fast regions
- **FIGARO Substrate:**
  - **Key idea:** use the **existing shared global row buffer** among subarrays within a DRAM bank **to provide** support for in-DRAM **data relocation**
  - **Fine-grained** (i.e., multi-byte) in-DRAM data relocation and **distance-independent** relocation latency
  - Avoids complex modifications to DRAM by using **existing structures**
- **FIGCache:**
  - **Key idea:** cache **only small, frequently-accessed portions of different DRAM rows** in a designated region of DRAM
  - Caches only the **parts of each row** that are expected to be accessed in the **near future**
  - **Increases row hits** by packing more **frequently-accessed** data into FIGCache
  - Improves system **performance** by **16.3%** on average
  - Reduces **DRAM energy** by **7.8%** on average
- **Conclusion:**
  - FIGARO enables **fine-grained data relocation** in-DRAM at low cost
  - FIGCache **outperforms state-of-the-art coarse-grained** in-DRAM caches

FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang[1], **Lois Orosa**[2], Xiangjun Peng[3,1], Yang Guo[1], Saugata Ghose[4,5], Minesh Patel[2], Jeremie S. Kim[2], Juan Gómez Luna[2], Mohammad Sadrosadati[6], Nika Mansouri Ghiasi[2], Onur Mutlu[2,5]

1 National University of Defense Technology

2 ETH zürich

3 香港中文大學 The Chinese University of Hong Kong

4 University of Illinois Urbana-Champaign

5 Carnegie Mellon University

6 IPM Institute for Research in Fundamental Sciences

SAFARI

MICRO 2020