RowClone

Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri

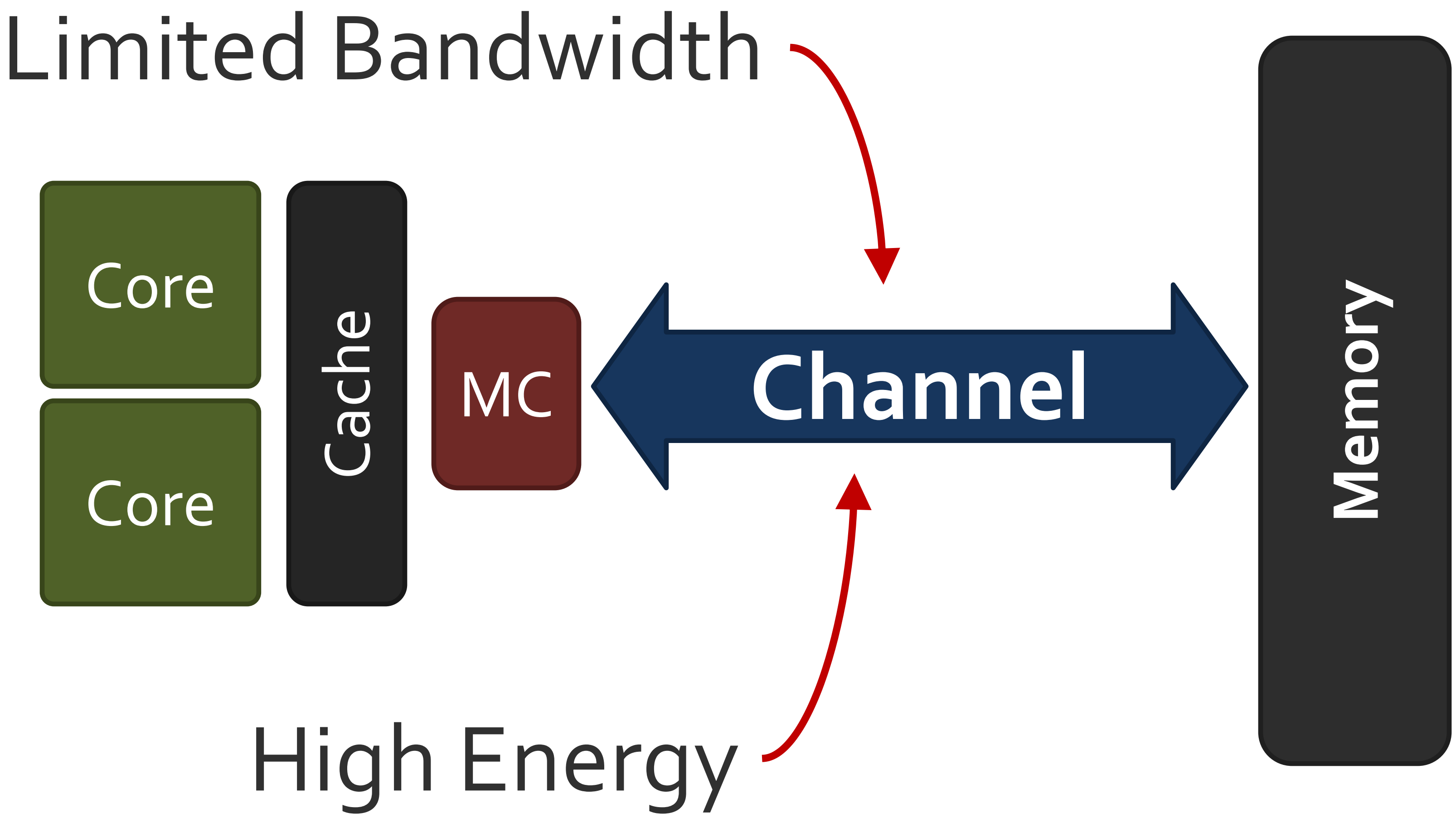
Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, T. C. Mowry

SAFARI Carnegie Mellon intel

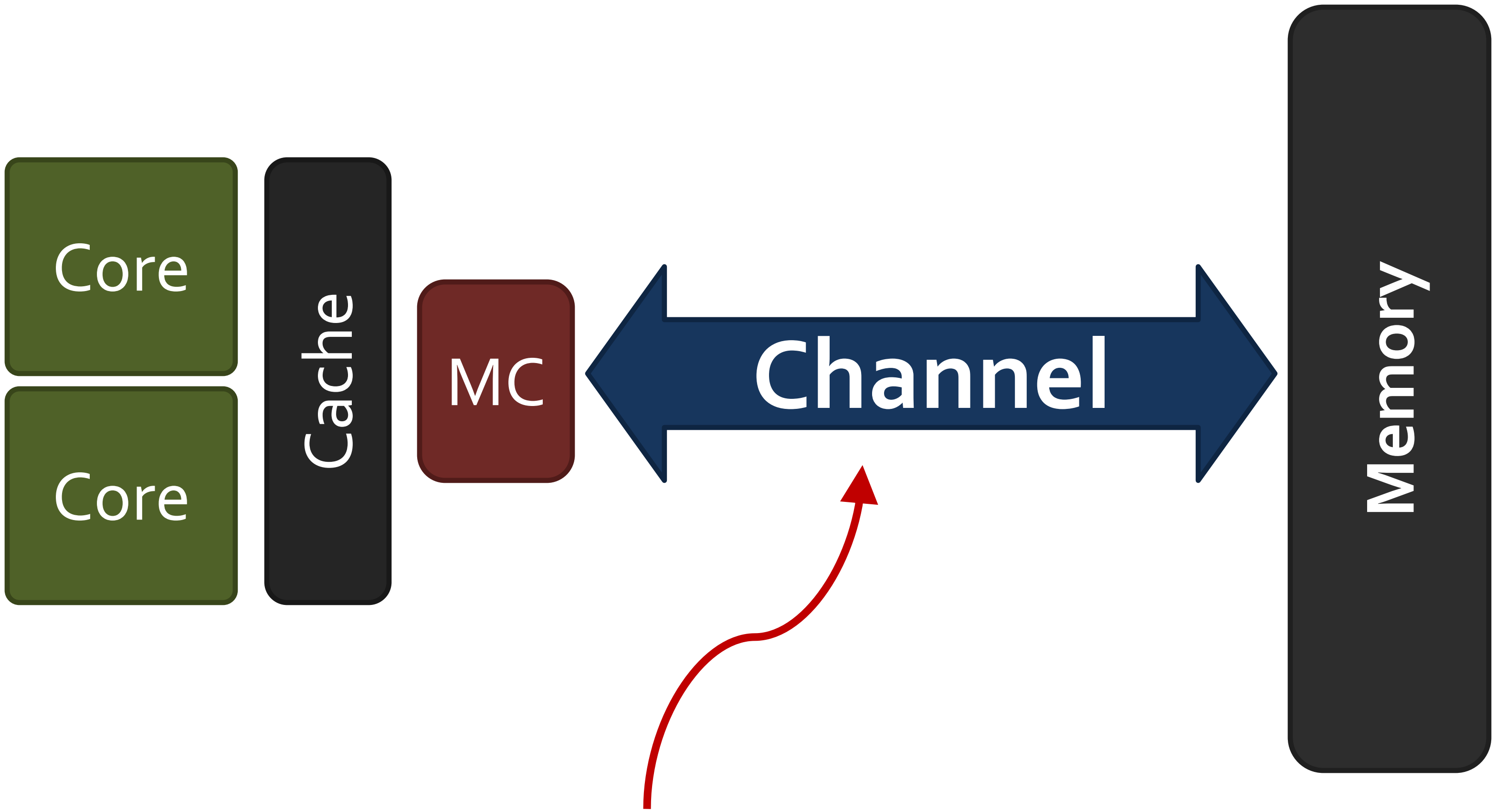
Executive Summary

- Bulk data copy and initialization
  - Unnecessarily move data on the memory channel
  - Degrade system performance and energy efficiency
- **RowClone** – perform copy in DRAM with low cost
  - Uses row buffer to copy large quantity of data
  - **Source row → row buffer → destination row**
  - 11X lower latency and 74X lower energy for a bulk copy
- Accelerate Copy-on-Write and Bulk Zeroing
  - Forking, checkpointing, zeroing (security), VM cloning
- Improves performance and energy efficiency at low cost
  - 27% and 17% for 8-core systems (0.01% DRAM chip area)

Memory Channel – Bottleneck



Goal: Reduce Memory Bandwidth Demand

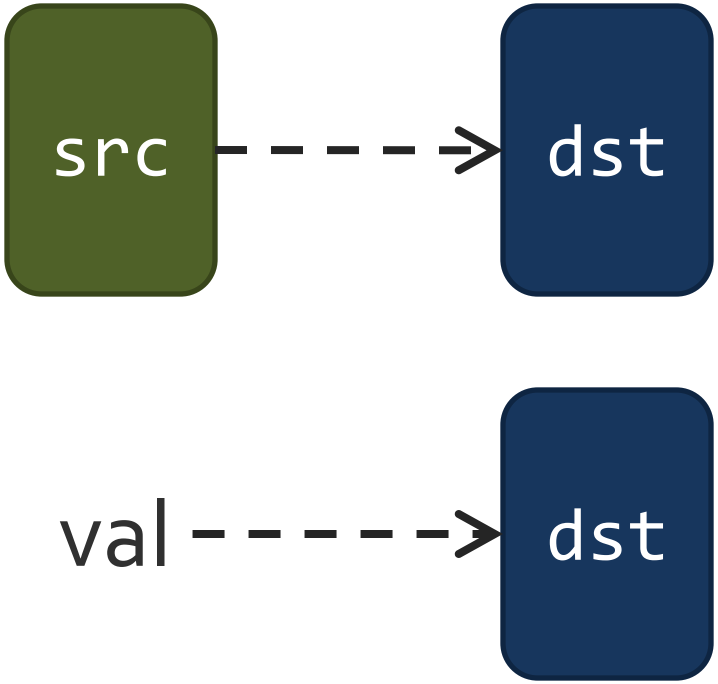


Reduce unnecessary data movement

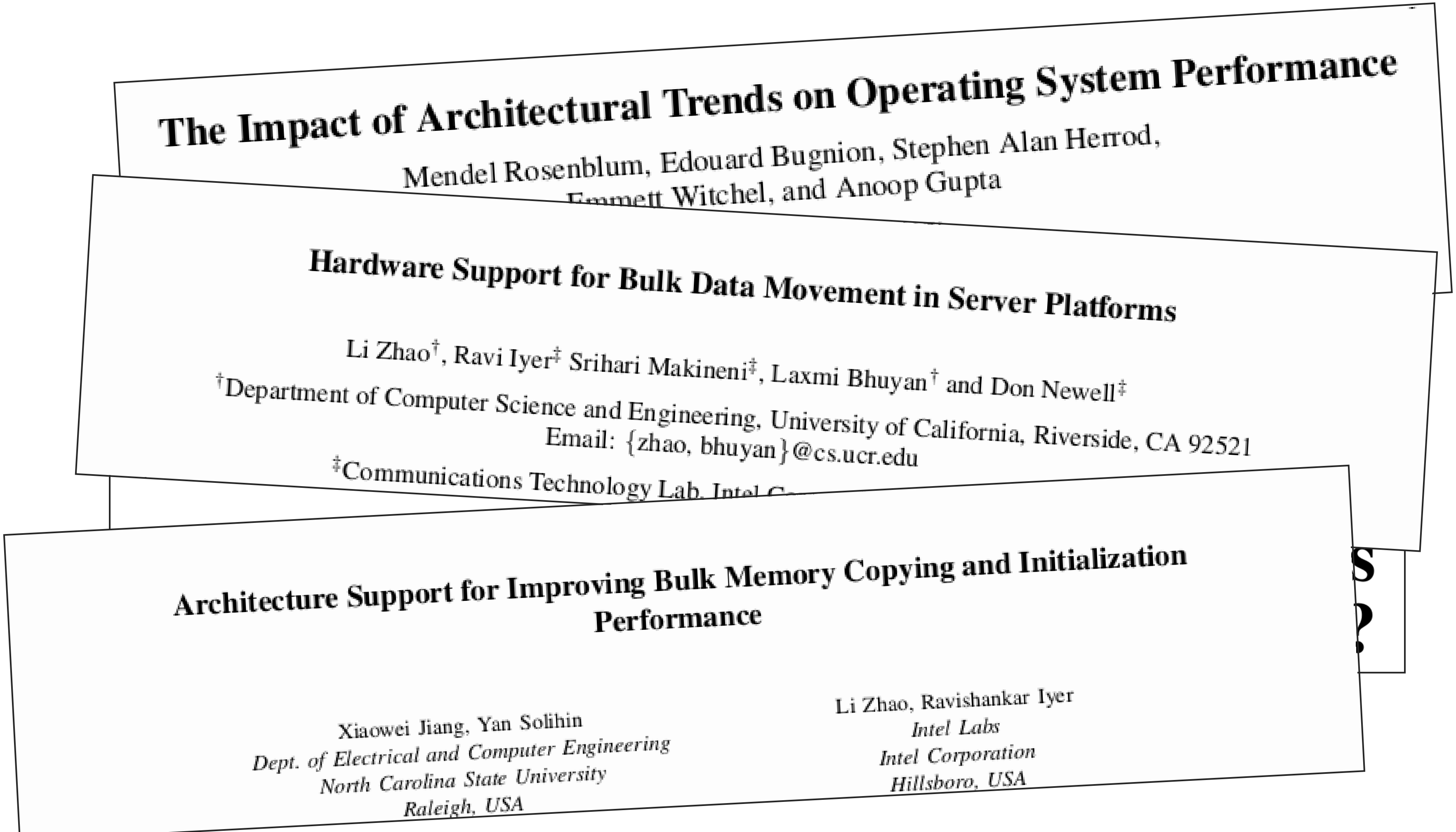
Bulk Data Copy and Initialization

Bulk Data Copy

Bulk Data Initialization



Bulk Data Copy and Initialization



The Impact of Architectural Trends on Operating System Performance

Mendel Rosenblum, Edouard Bugnion, Stephen Alan Herrod,
Emmett Witchel, and Anoop Gupta

Hardware Support for Bulk Data Movement in Server Platforms

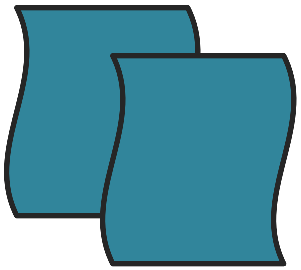
Li Zhao[†], Ravi Iyer[‡] Srihari Makineni[‡], Laxmi Bhuyan[†] and Don Newell[‡]
[†]Department of Computer Science and Engineering, University of California, Riverside, CA 92521
Email: {zhao, bhuyan}@cs.ucr.edu
[‡]Communications Technology Lab, Intel C

Architecture Support for Improving Bulk Memory Copying and Initialization Performance

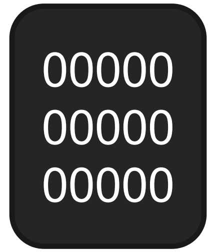
Xiaowei Jiang, Yan Solihin
Dept. of Electrical and Computer Engineering
North Carolina State University
Raleigh, USA

Li Zhao, Ravishankar Iyer
Intel Labs
Intel Corporation
Hillsboro, USA

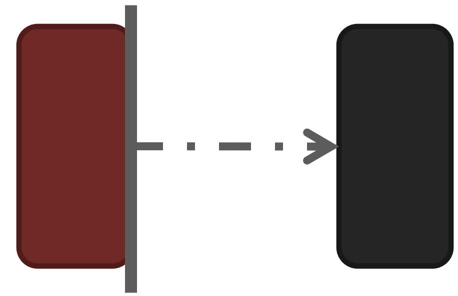
Bulk Copy and Initialization – Applications



Forking

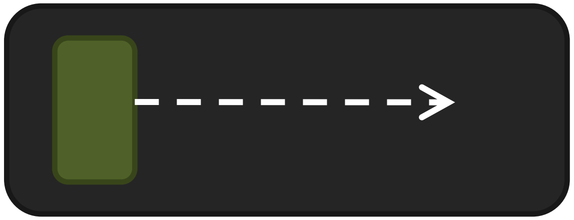


Zero initialization
(e.g., security)



Checkpointing

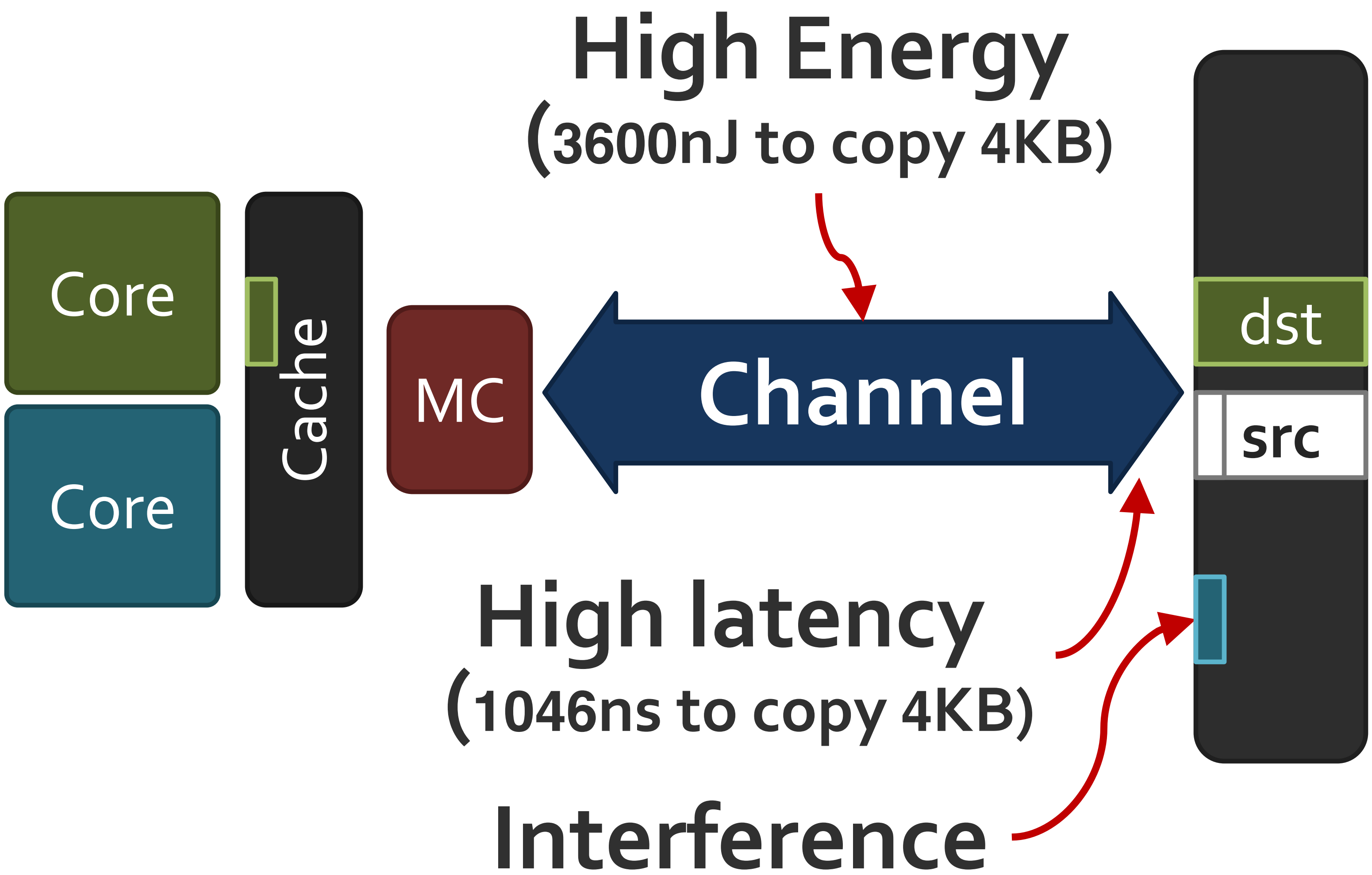
VM Cloning
Deduplication



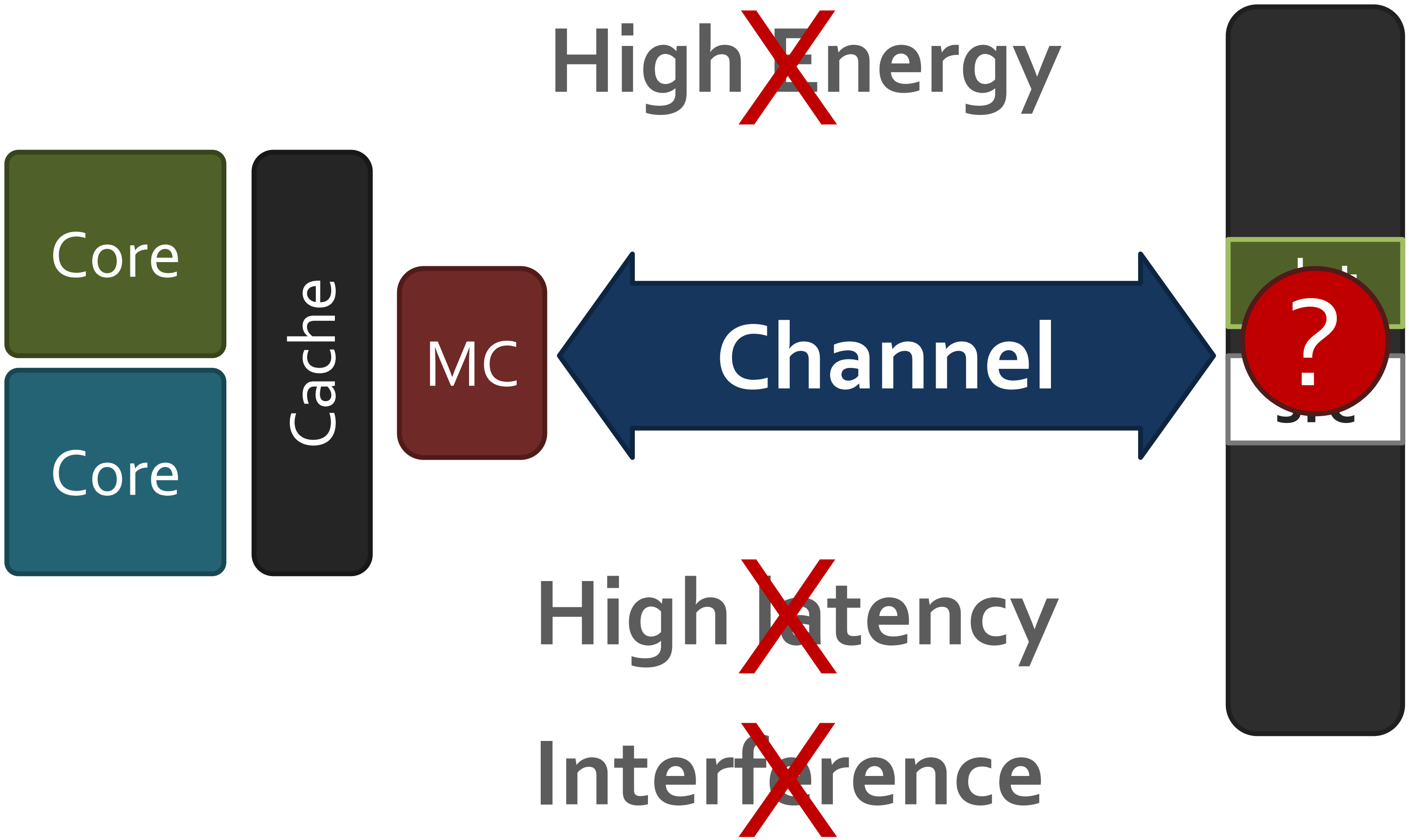
Page Migration

• • •
Many more

Shortcomings of Existing Approach



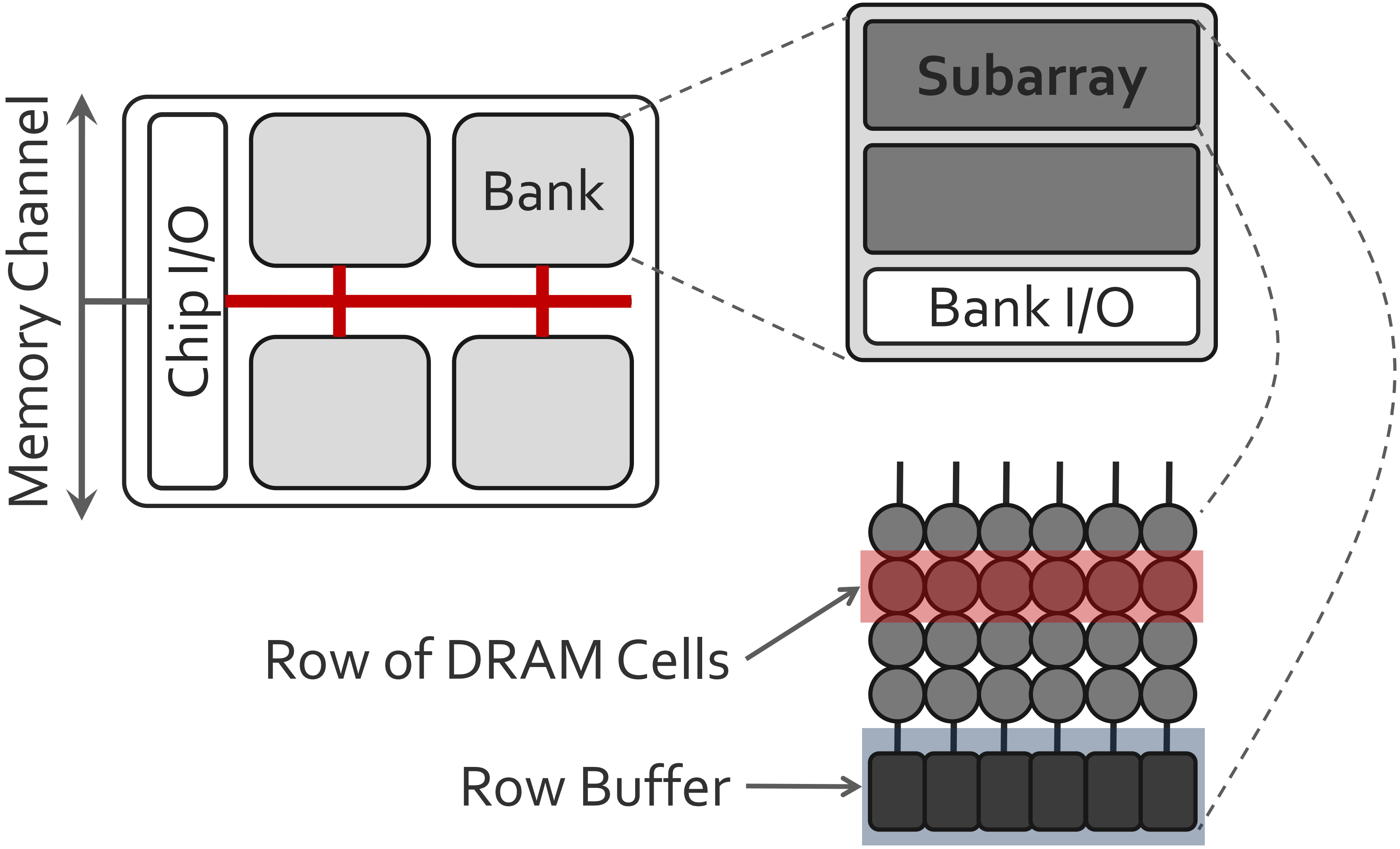
Our Approach: In-DRAM Copy with Low Cost



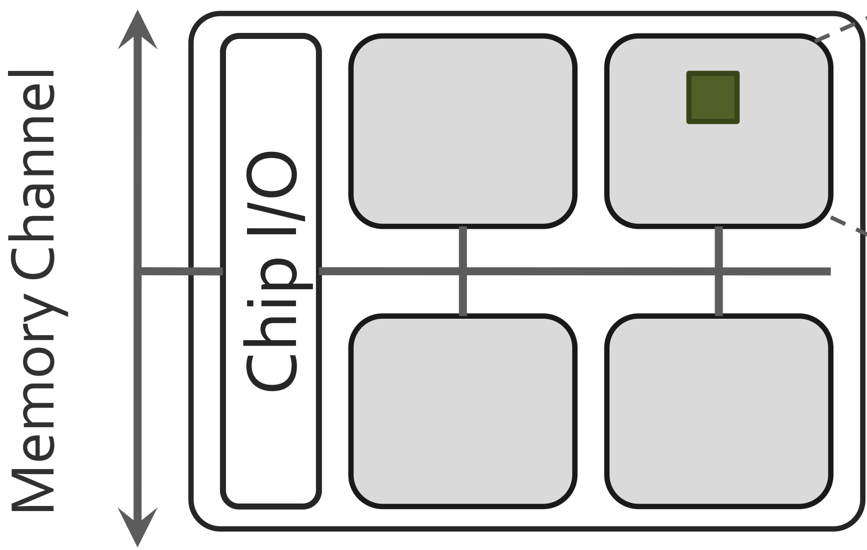
Outline

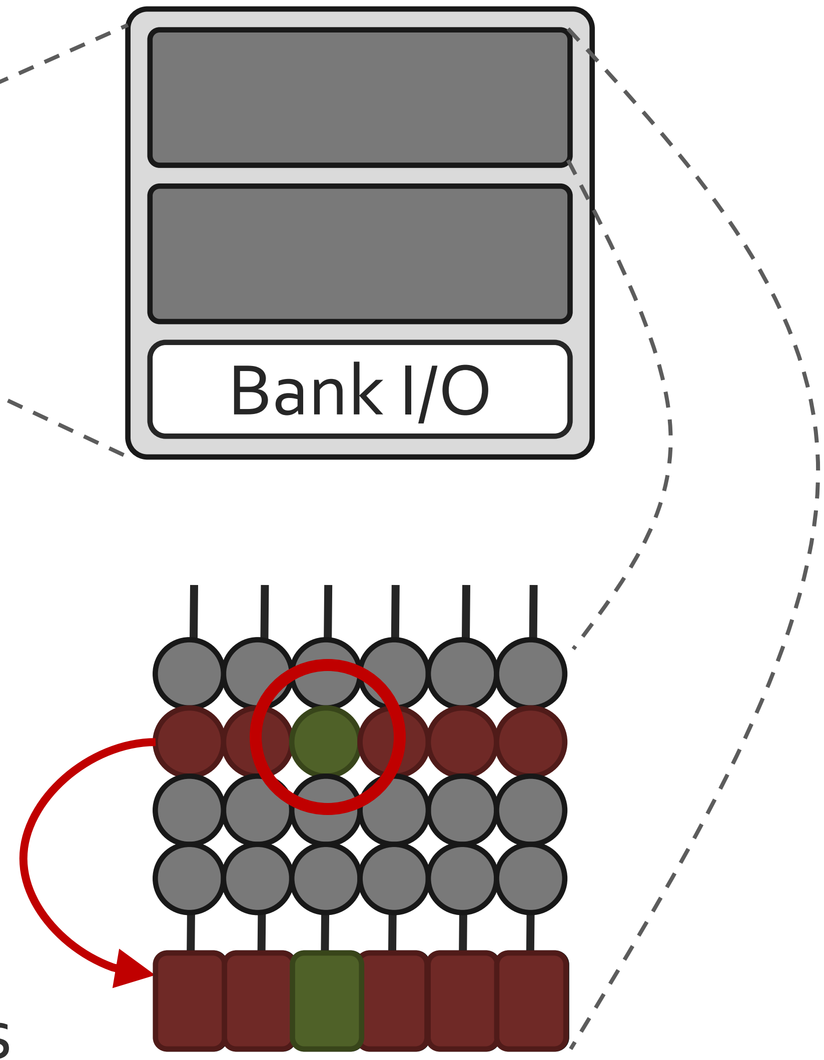
- ✓ Introduction
- DRAM Background
- RowClone
  - Fast Parallel Mode
  - Pipelined Serial Mode
- End-to-end Design
- Evaluation

DRAM Chip Organization



DRAM Read Operation

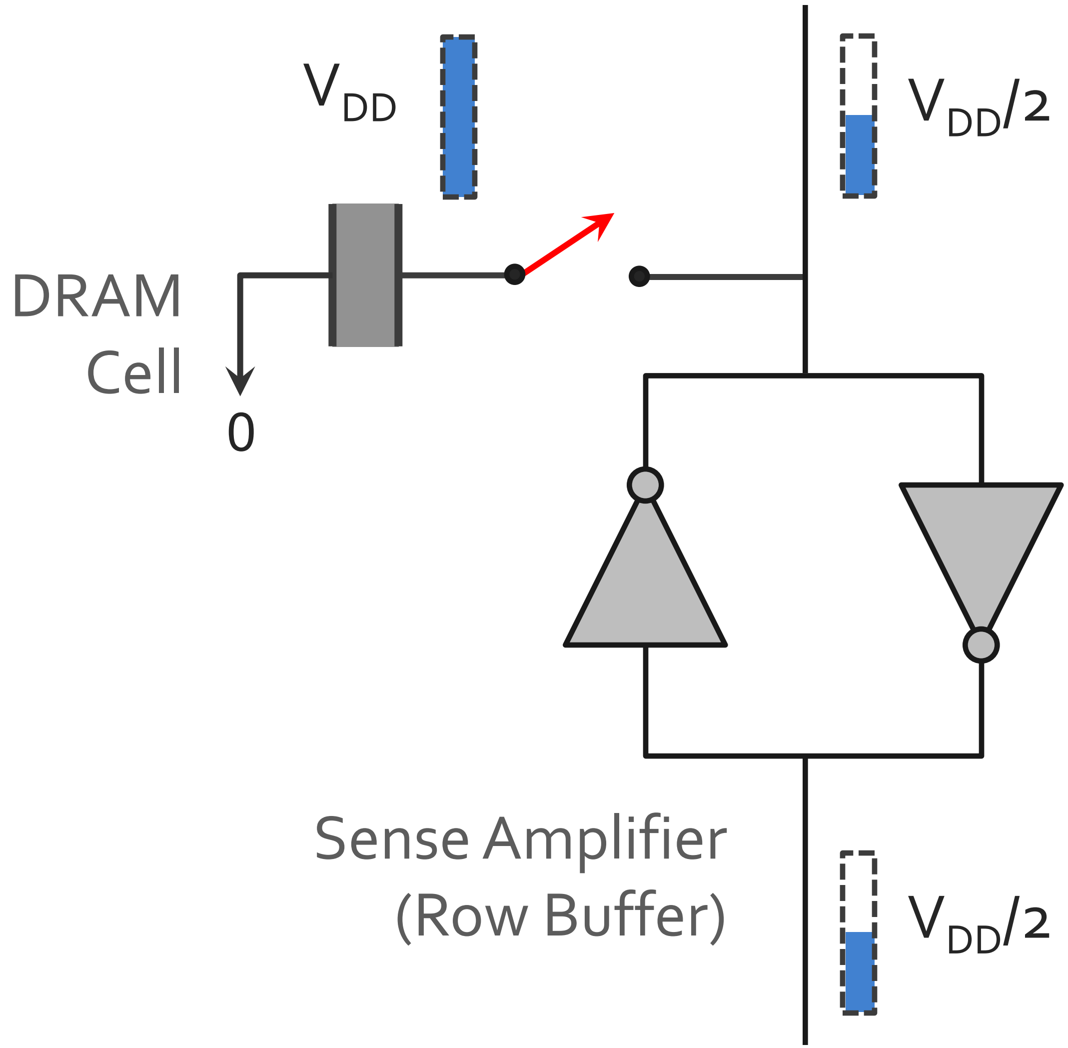




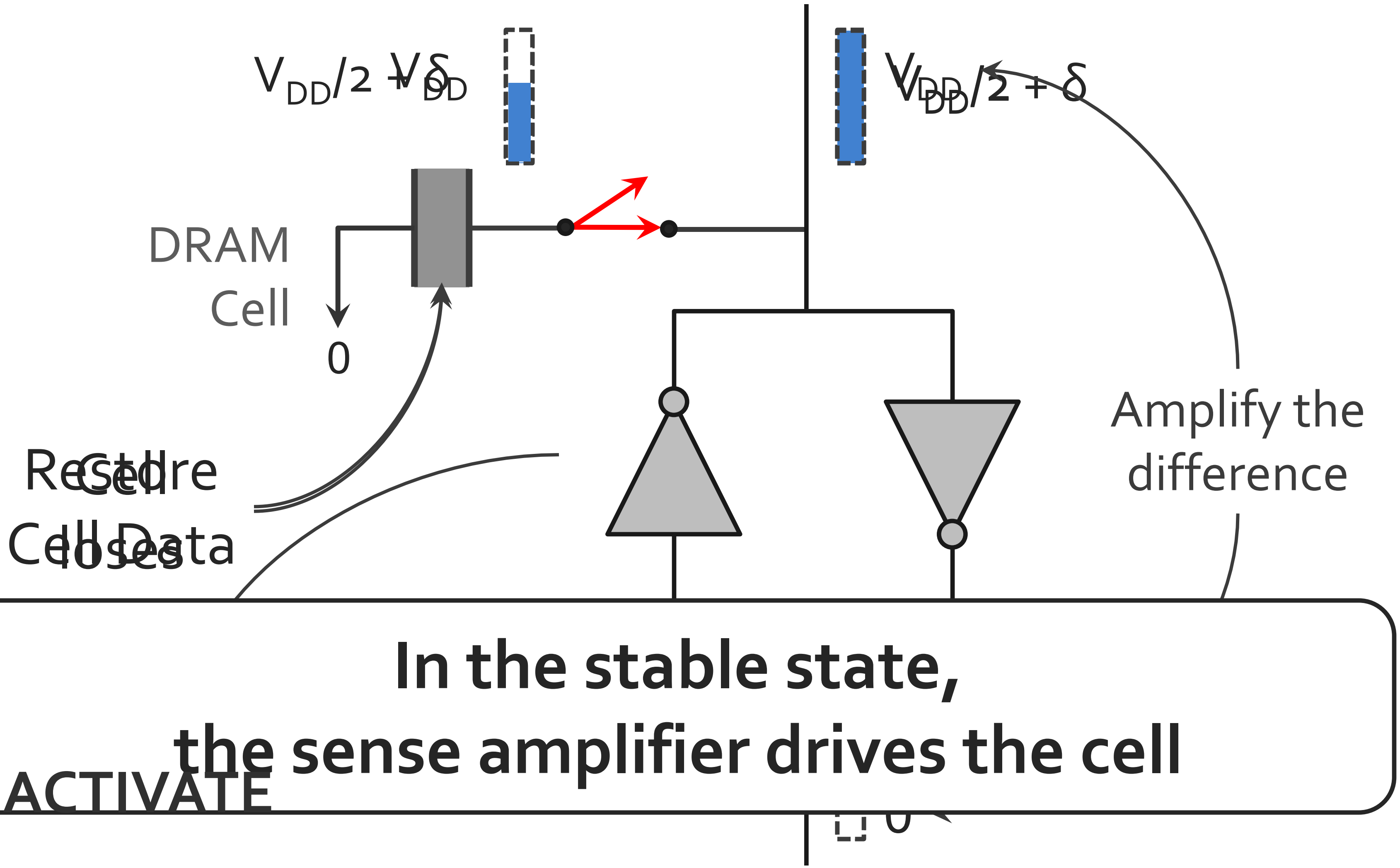
ACTIVATE: Copy data from row to row buffer

READ: Transfer data to channel using the shared bus

DRAM Cell Operation



DRAM Cell Operation

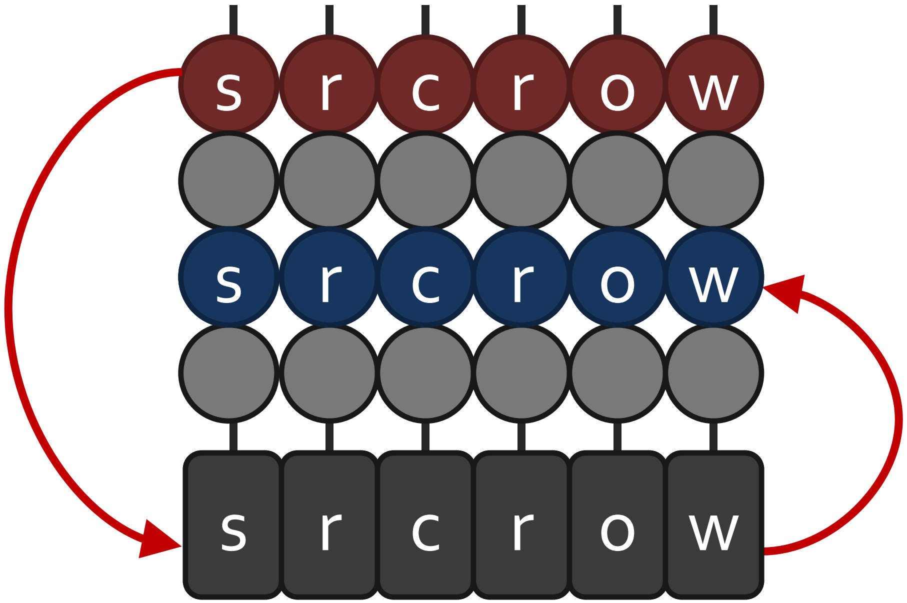


In the stable state,
the sense amplifier drives the cell

Outline

- ✓ Introduction
- ✓ DRAM Background
- RowClone
  - Fast Parallel Mode
  - Pipelined Serial Mode
- End-to-end Design
- Evaluation

RowClone: Fast Parallel Mode (FPM)

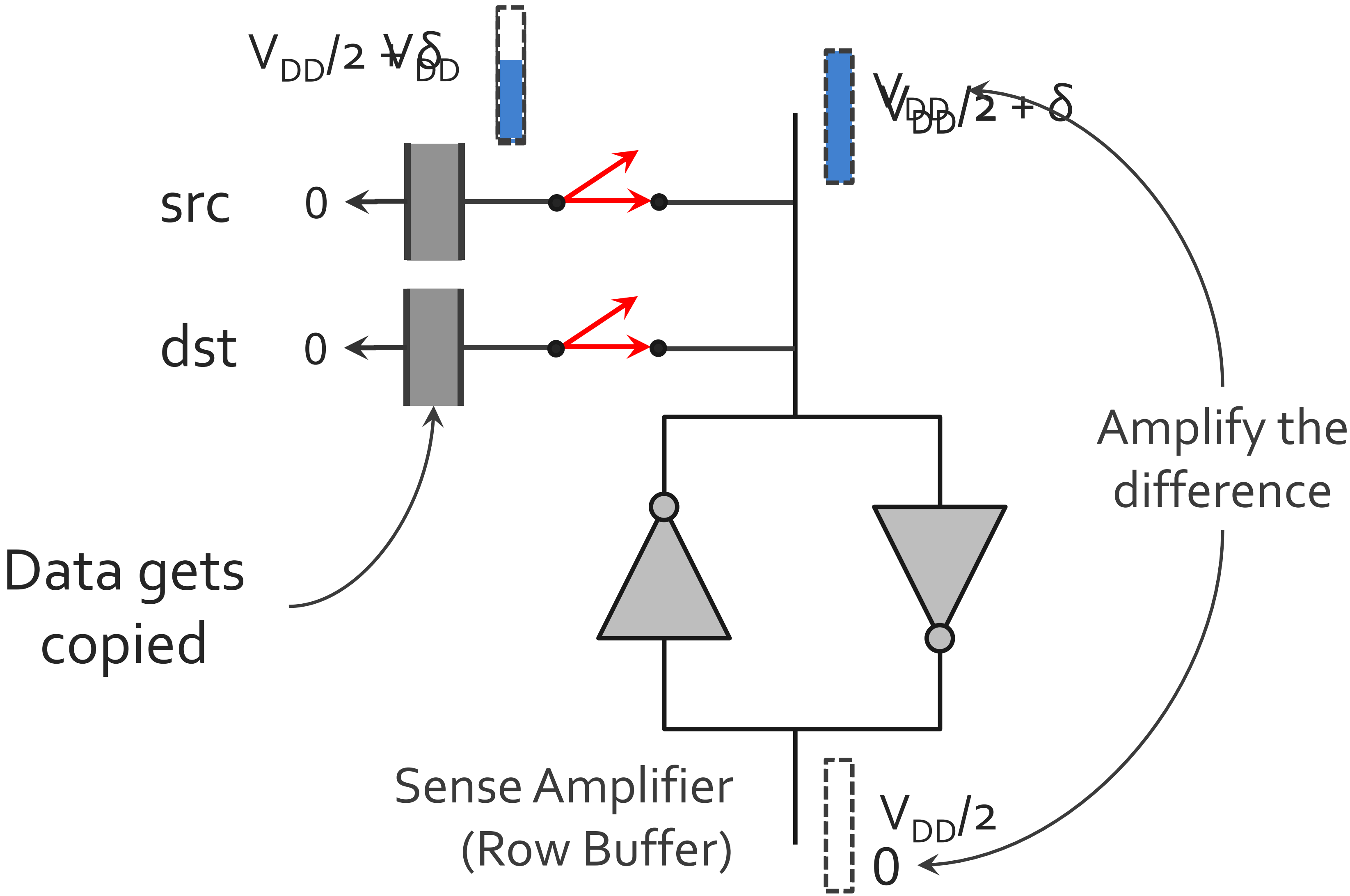


Row Buffer

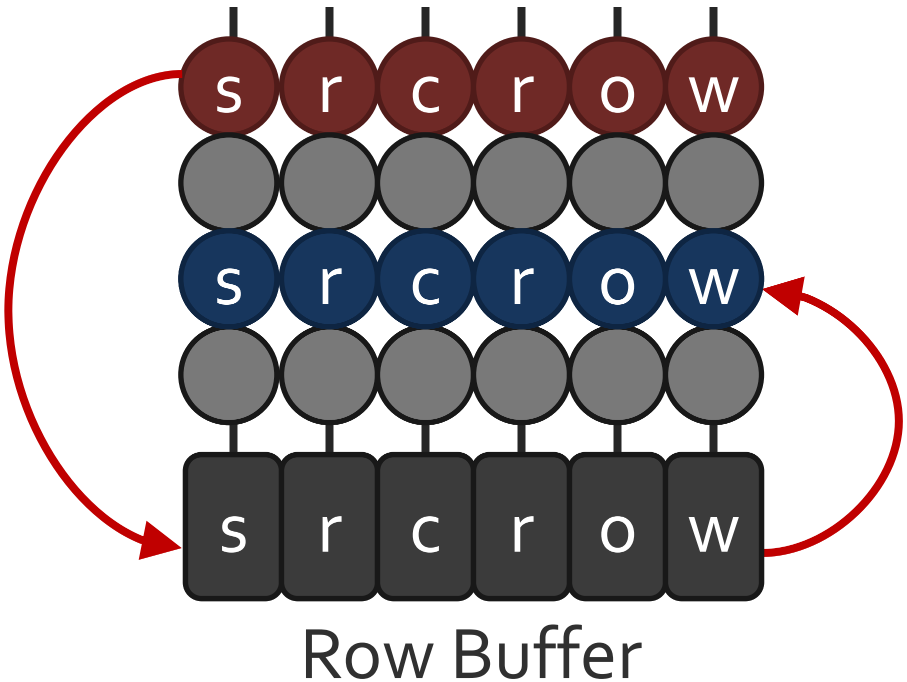
✓ 1. Source row to row buffer

? 2. Row buffer to destination row

Fast Parallel Mode: Implementation



Fast Parallel Mode: Implementation



1. **Activate** src row (copy data from src to row buffer)

2. **Activate** dst row (disconnect src from row buffer, connect dst – copy data from row buffer to dst)

Fast Parallel Mode: Benefits

Bulk Data Copy

Latency **11x** ↓

1046ns to 90ns

Energy **74x** ↓

3600nJ to 40nJ

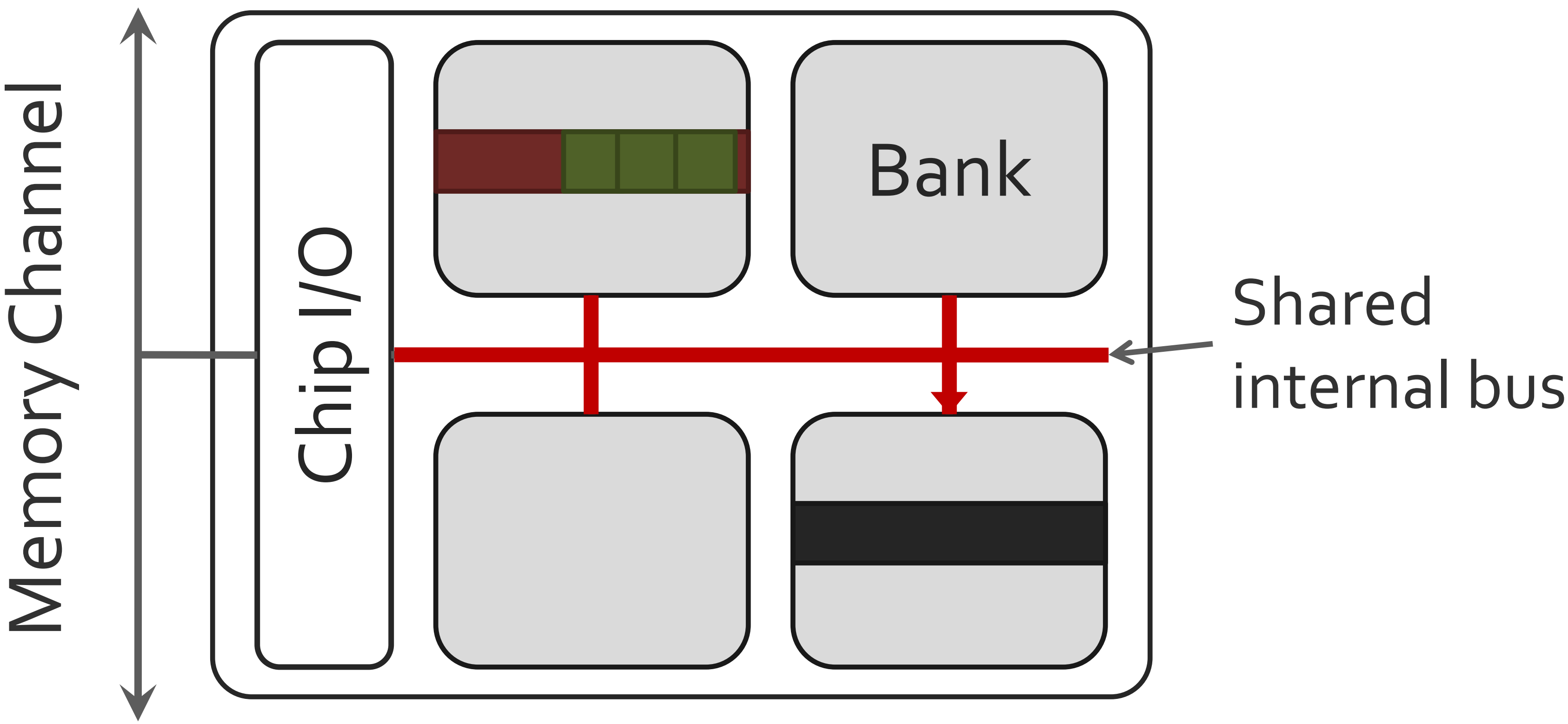
No bandwidth consumption

Very little changes to the DRAM chip

Fast Parallel Mode: Constraints

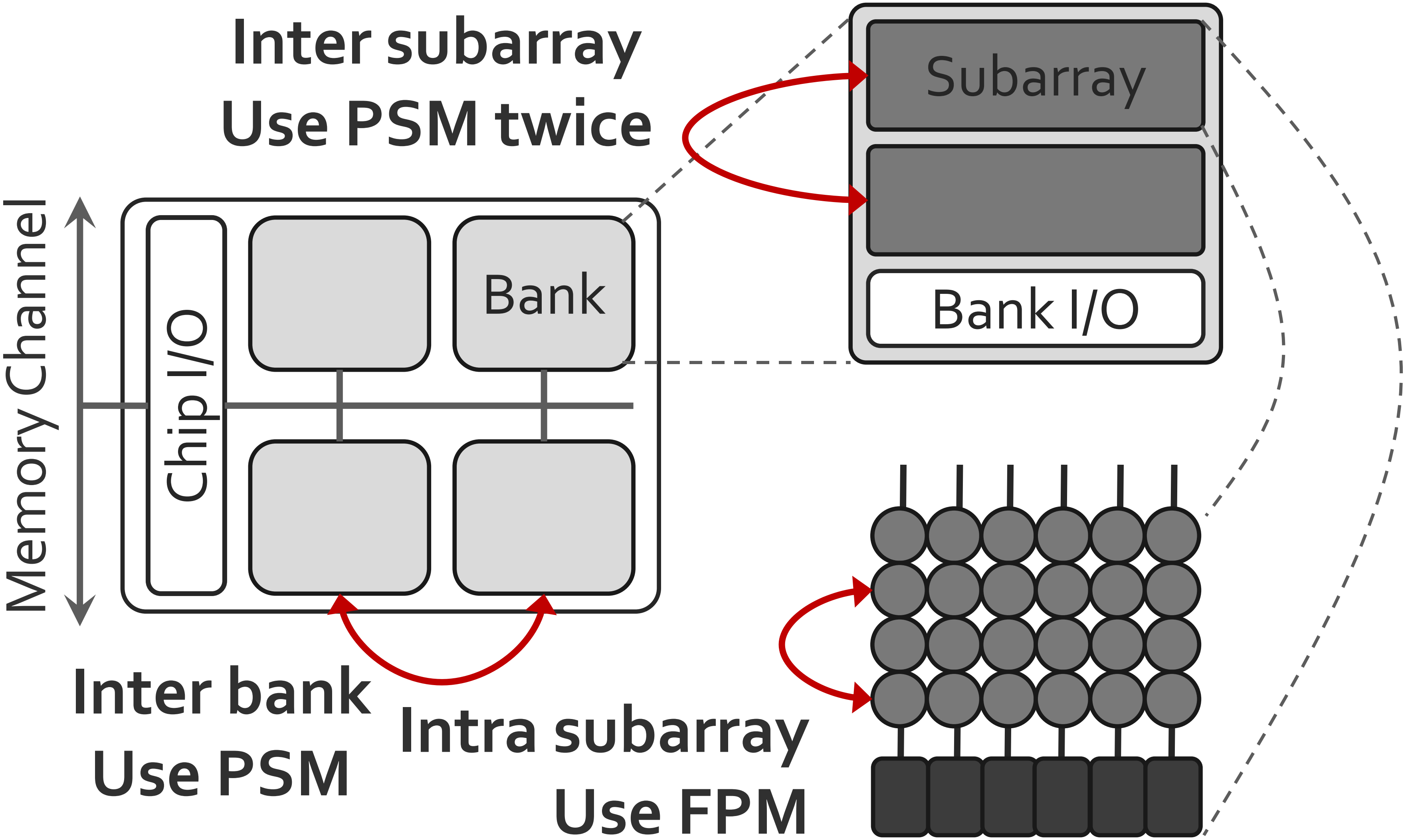
- Location of source/destination
  - Both should be in the same subarray
- Size of the copy
  - Copies *all* the data from source row to destination

RowClone: Pipelined Serial Mode (PSM)



Overlap the latency of the read and the write
1.9X latency reduction, 3.2X energy reduction

Bulk Copy using RowClone

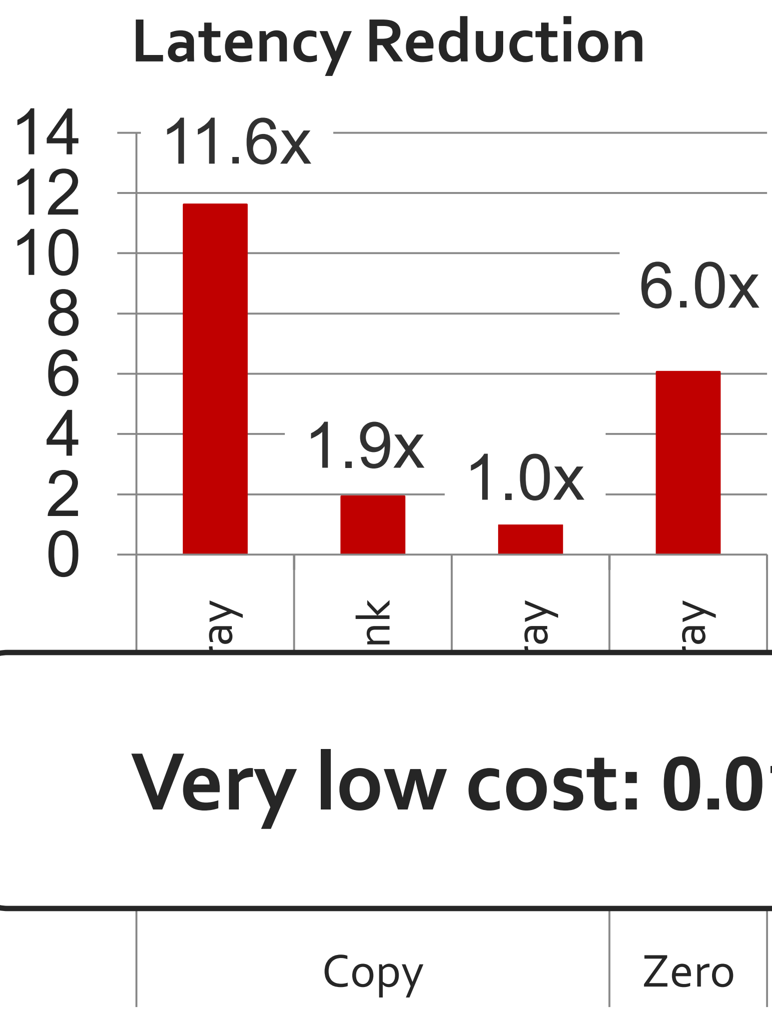


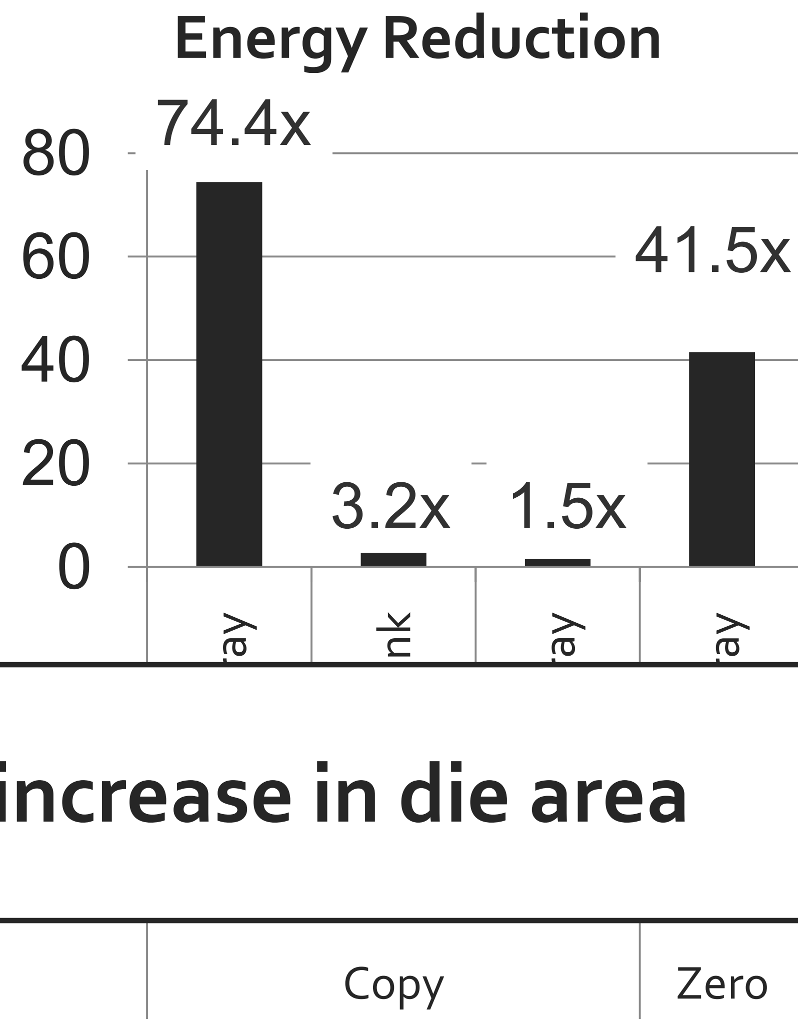
Bulk Initialization

- Initialization with arbitrary data
  - Initialize one row
  - Copy the data to other rows

- Zero initialization (most common)
  - Reserve a row in each subarray (always zero)
  - Copy data from reserved row (FPM mode)
  - **6.0X** lower latency, **41.5X** lower DRAM energy
  - 0.2% loss in capacity

Latency and Energy Benefits





Very low cost: 0.01% increase in die area

Outline

- ✓ Introduction
- ✓ DRAM Background
- ✓ RowClone
  - Fast Parallel Mode
  - Pipelined Serial Mode
- End-to-end Design
- Evaluation

End-to-end System Design

- Application
- Operating System
- ISA
- Microarchitecture
- DRAM (RowClone)

How does the software communicate occurrences of bulk copy/initialization to hardware?

How to ensure cache coherence?

How to maximize use of the Fast Parallel Mode?

Handling data reuse after zero initialization?

1. Hardware/Software Interface

- Two new instructions
  - memcopy and meminit
  - Similar instructions present in existing ISAs

- Microarchitecture Implementation
  - Checks if instructions can be sped up by RowClone
  - Export instructions to the memory controller

2. Managing Cache Coherence

- RowClone modifies data in memory
  - Need to maintain coherence of cached data

- Similar to DMA
  - Source and destination in memory
  - Can leverage hardware support for DMA

- Additional optimizations

3. Maximizing Use of the Fast Parallel Mode

- Make operating system subarray-aware

- Primitives amenable to use of FPM
  - **Copy-on-Write**
    - Allocate destination in same subarray as source
    - Use FPM to copy
  - **Bulk Zeroing**
    - Use FPM to copy data from reserved zero row

4. Handling Data Reuse After Zeroing

- Data reuse after zero initialization
  - Phase 1: OS zeroes out the page
  - Phase 2: Application uses cachelines of the page

- RowClone
  - Avoids misses in phase 1
  - But incurs misses in phase 2

- **RowClone-Zero-Insert (RowClone-ZI)**
  - Insert clean zero cachelines

Outline

- ✓ Introduction
- ✓ DRAM Background
- ✓ RowClone
  - Fast Parallel Mode
  - Pipelined Serial Mode
- ✓ End-to-end Design
- ▪ Evaluation

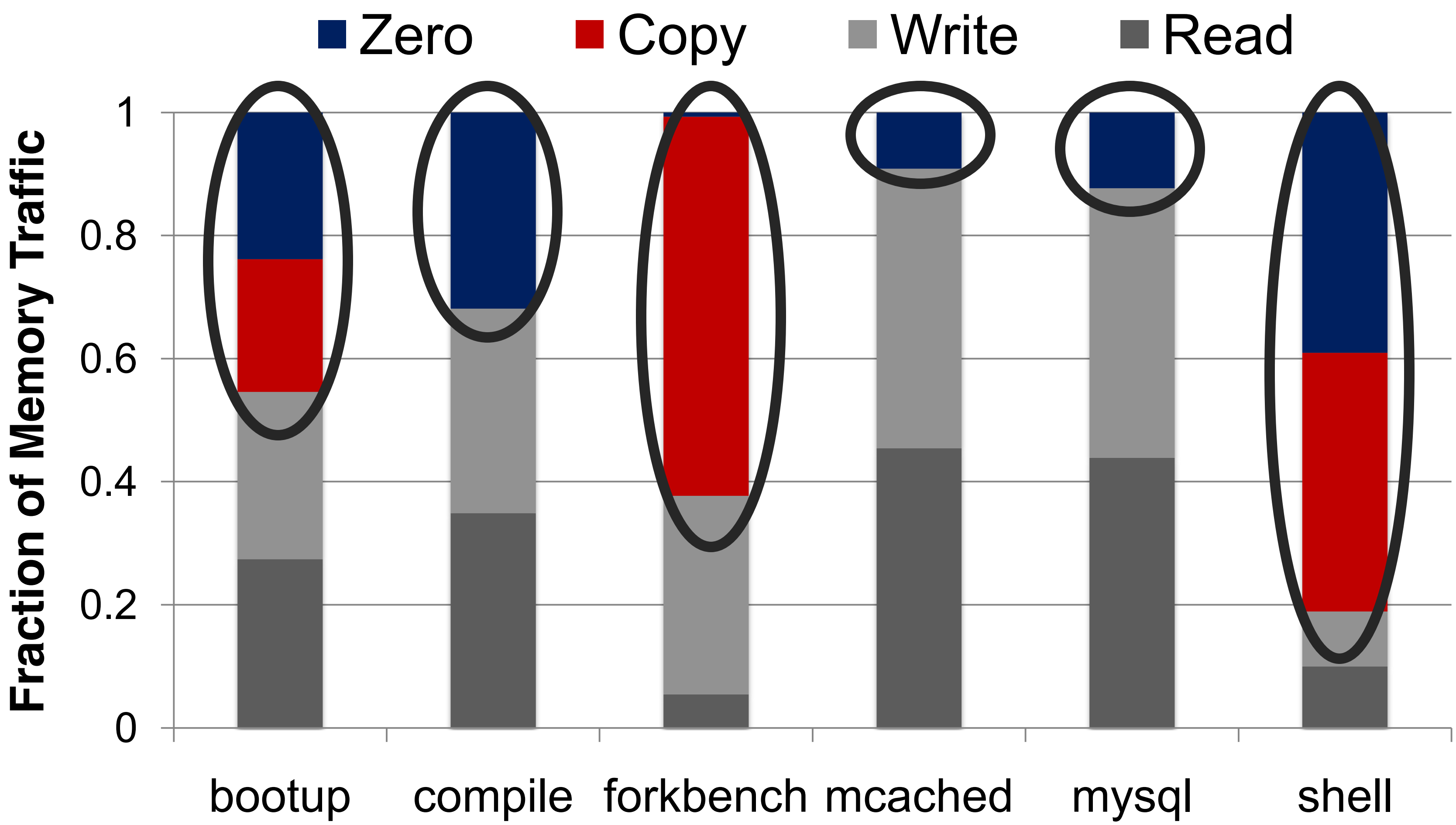
Methodology

- Out-of-order multi-core simulator
- 1MB/core last-level cache
- Cycle-accurate DDR3 DRAM simulator
- 6 Copy/Initialization intensive applications
  +SPEC CPU2006 for multi-core
- Performance
  - Instruction throughput for single-core
  - Weighted Speedup for multi-core

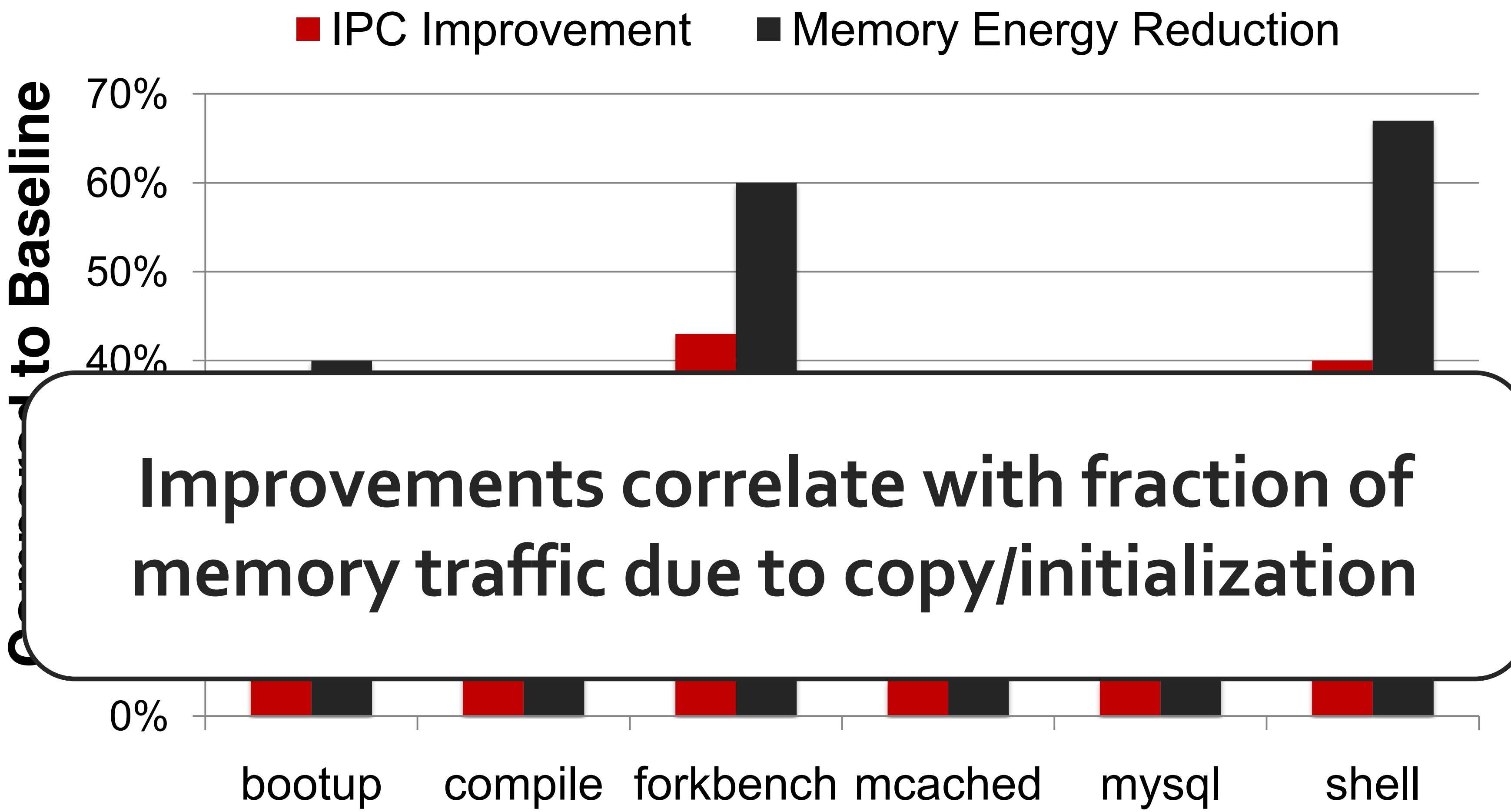
Copy/Initialization Intensive Applications

- **System bootup** (Booting the Debian OS)
- **Compile** (GNU C compiler – executing cc1)
- **Forkbench** (A fork microbenchmark)
- **Memcached** (Inserting a large number of objects)
- **MySql** (Loading a database)
- **Shell** script (`find` with `ls` on each subdirectory)

Memory Traffic due to Copy/Initialization



Single-Core – Performance and Energy

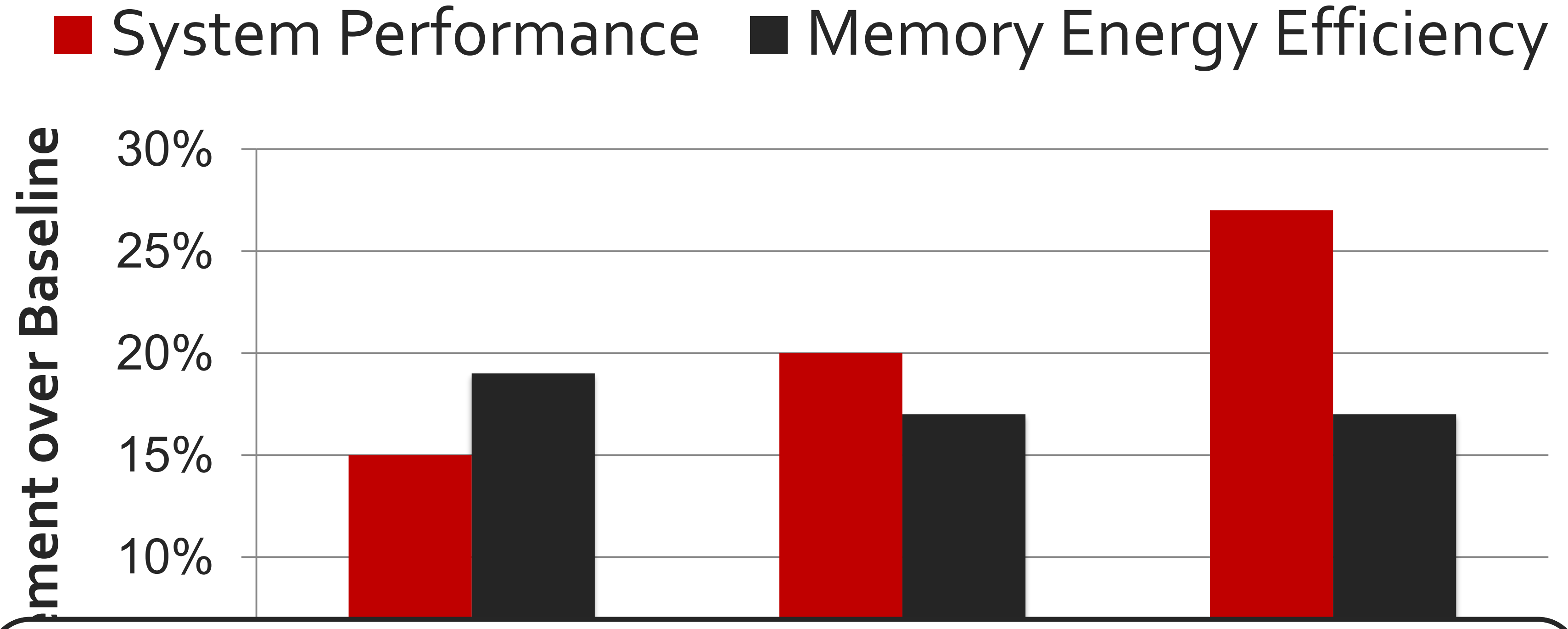


Improvements correlate with fraction of memory traffic due to copy/initialization

Multi-Core Systems

- Reduced bandwidth consumption benefits all applications.
- Run copy/initialization intensive applications with memory intensive SPEC applications.
- Half the cores run copy/initialization intensive applications. Remaining half run SPEC applications.

Multi-Core Results: Summary



Consistent improvement in energy/instruction

Other Results and Discussion in the Paper

- Discussion on interleaving and copy granularity
- Detailed analysis of the fork benchmark
- Detailed multi-core results and analysis
- Results with the PSM mode
- Analysis of RowClone-ZI
- Comparison to memory-controller-based DMA

Conclusion

- Bulk data copy and initialization
  - Unnecessarily move data on the memory channel
  - Degrade system performance and energy efficiency
- **RowClone** – perform copy in DRAM with low cost
  - Uses row buffer to copy large quantity of data
  - **Source row → row buffer → destination row**
  - 11X lower latency and 74X lower energy for a bulk copy
- Accelerate Copy-on-Write and Bulk Zeroing
  - Forking, checkpointing, zeroing (security), VM cloning
- Improves performance and energy efficiency at low cost
  - 27% and 17% for 8-core systems (0.01% chip area overhead)

RowClone

Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri

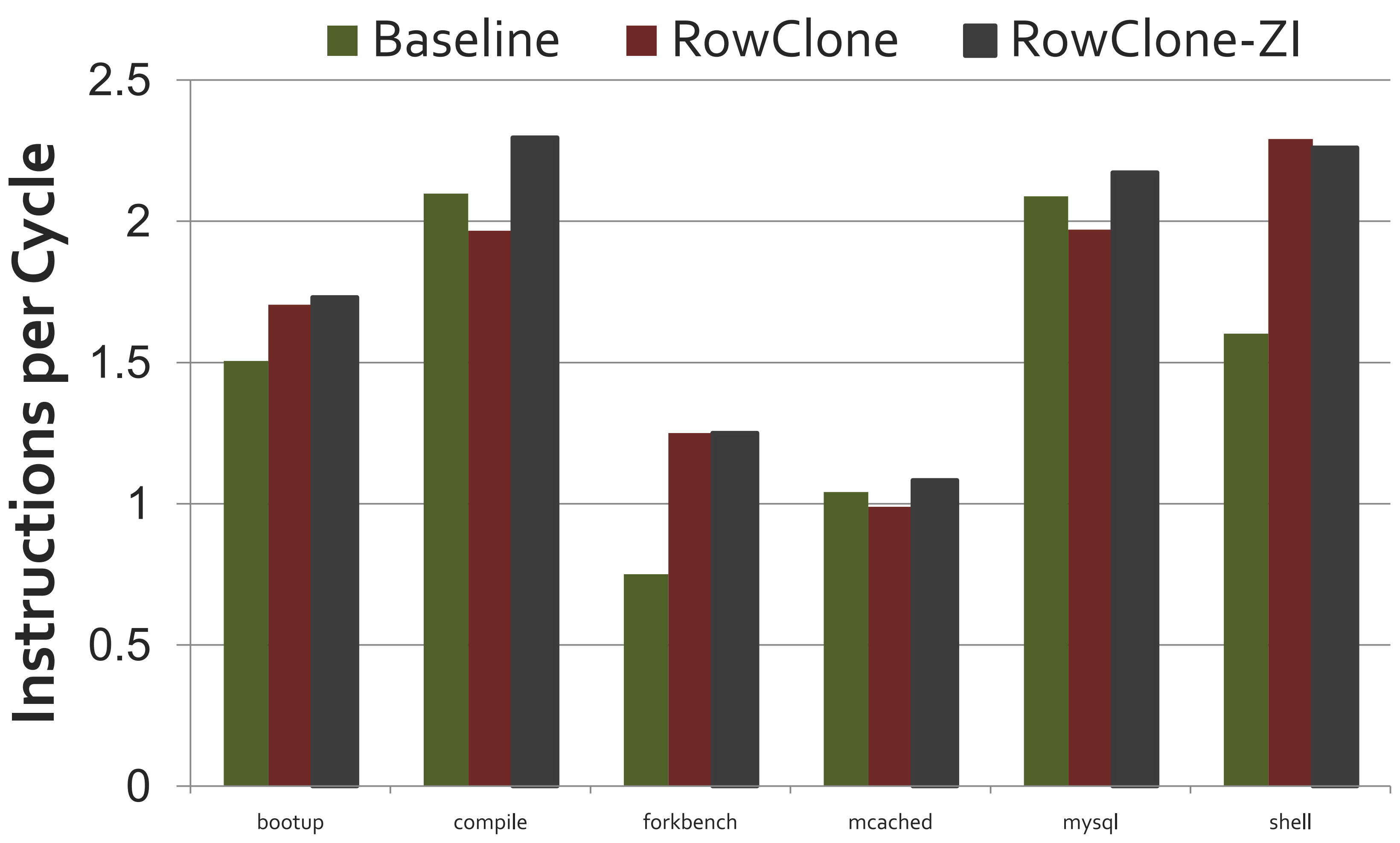
Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, T. C. Mowry

Backup Slides

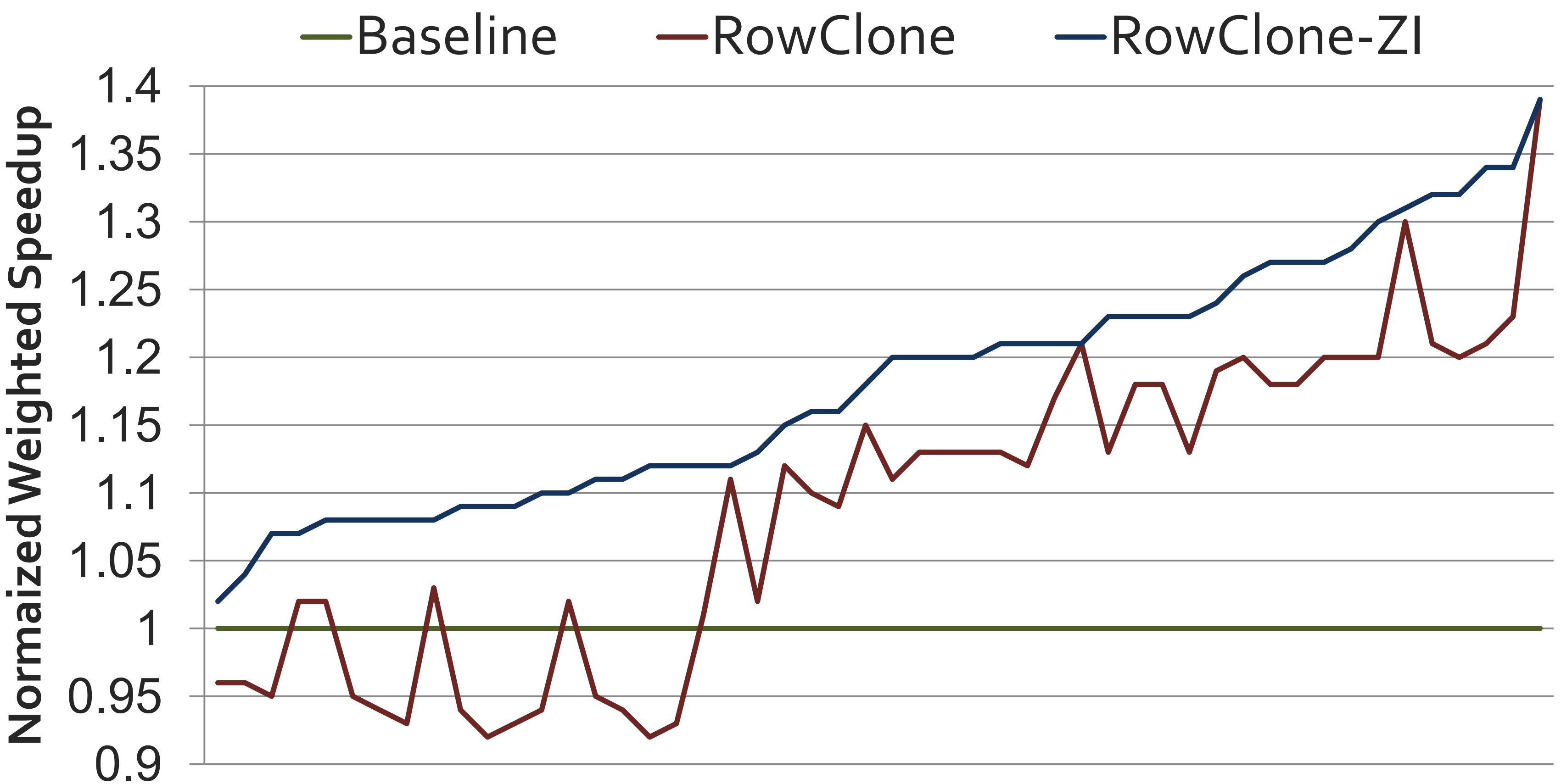
Multi-core Metrics

	2-core	4-core	8-core
# Workloads	138	50	40
Weighted Speedup	15%	20%	27%
Instruction Throughput	14%	15%	25%
Harmonic Speedup	13%	16%	29%
Max Slowdown Reduction	6%	12%	23%
Bandwidth/Instruction Reduction	29%	27%	28%
Energy/Instruction Reduction	19%	17%	17%

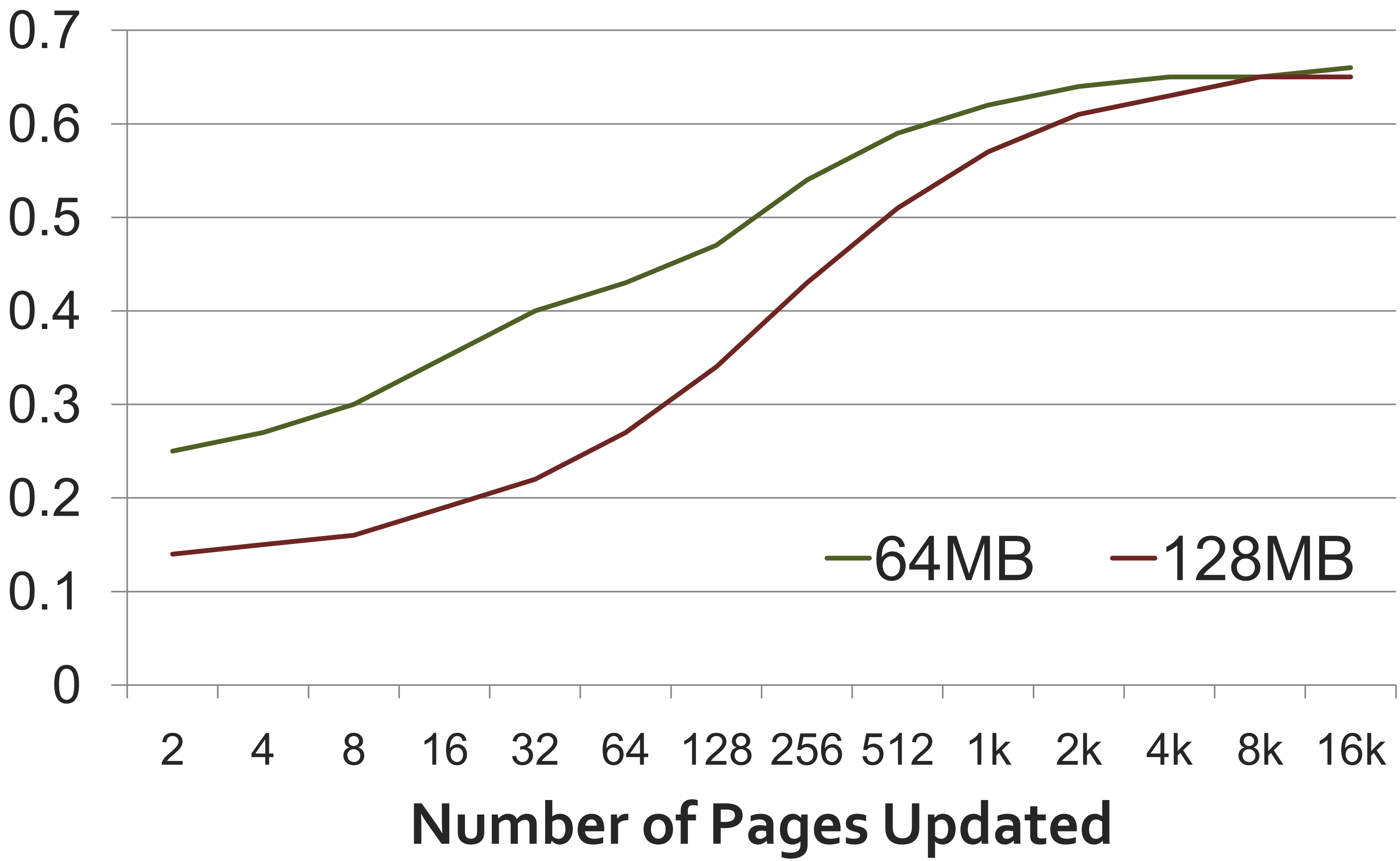
RowClone-ZI Single-Core



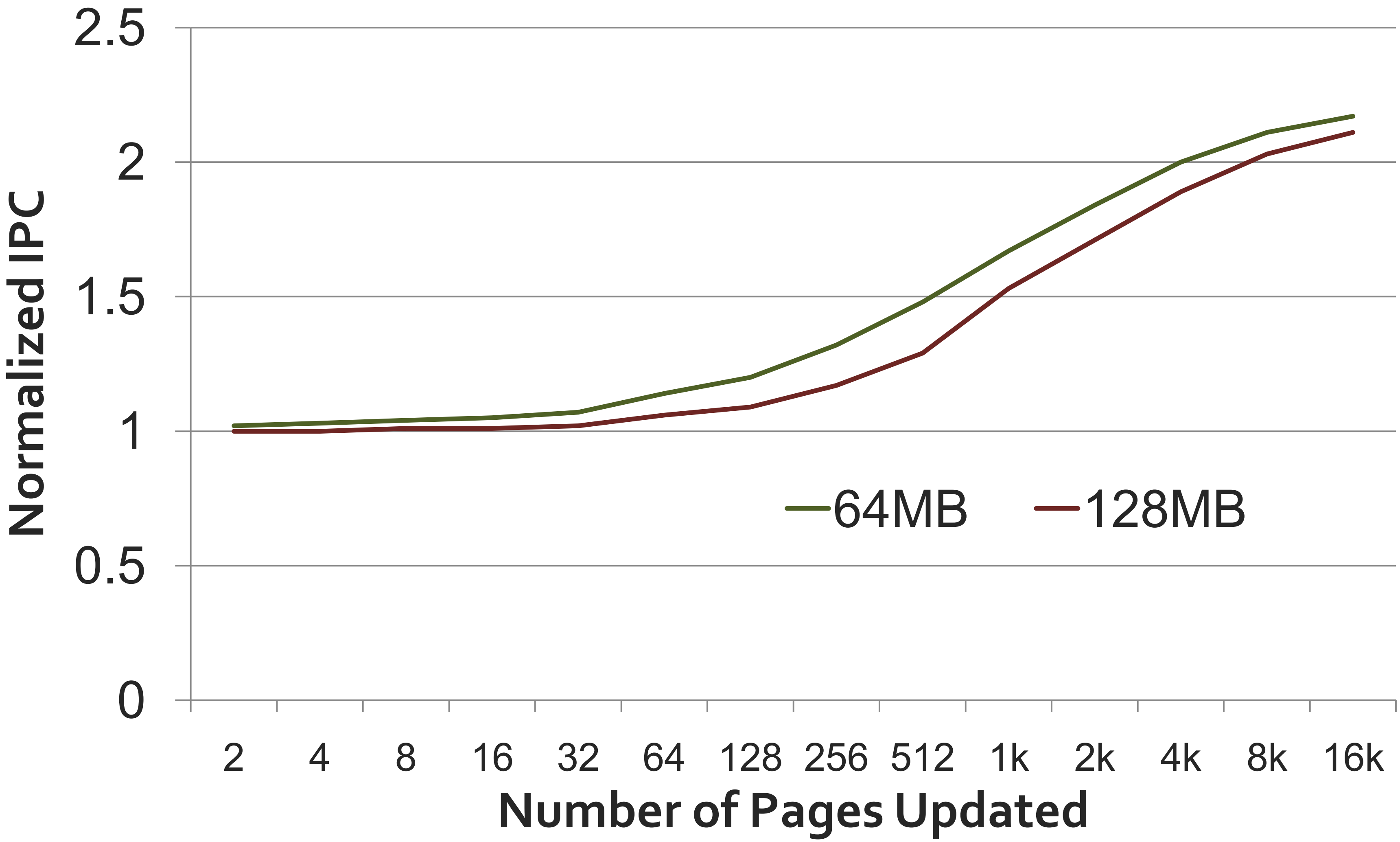
RowClone-ZI Multi-Core



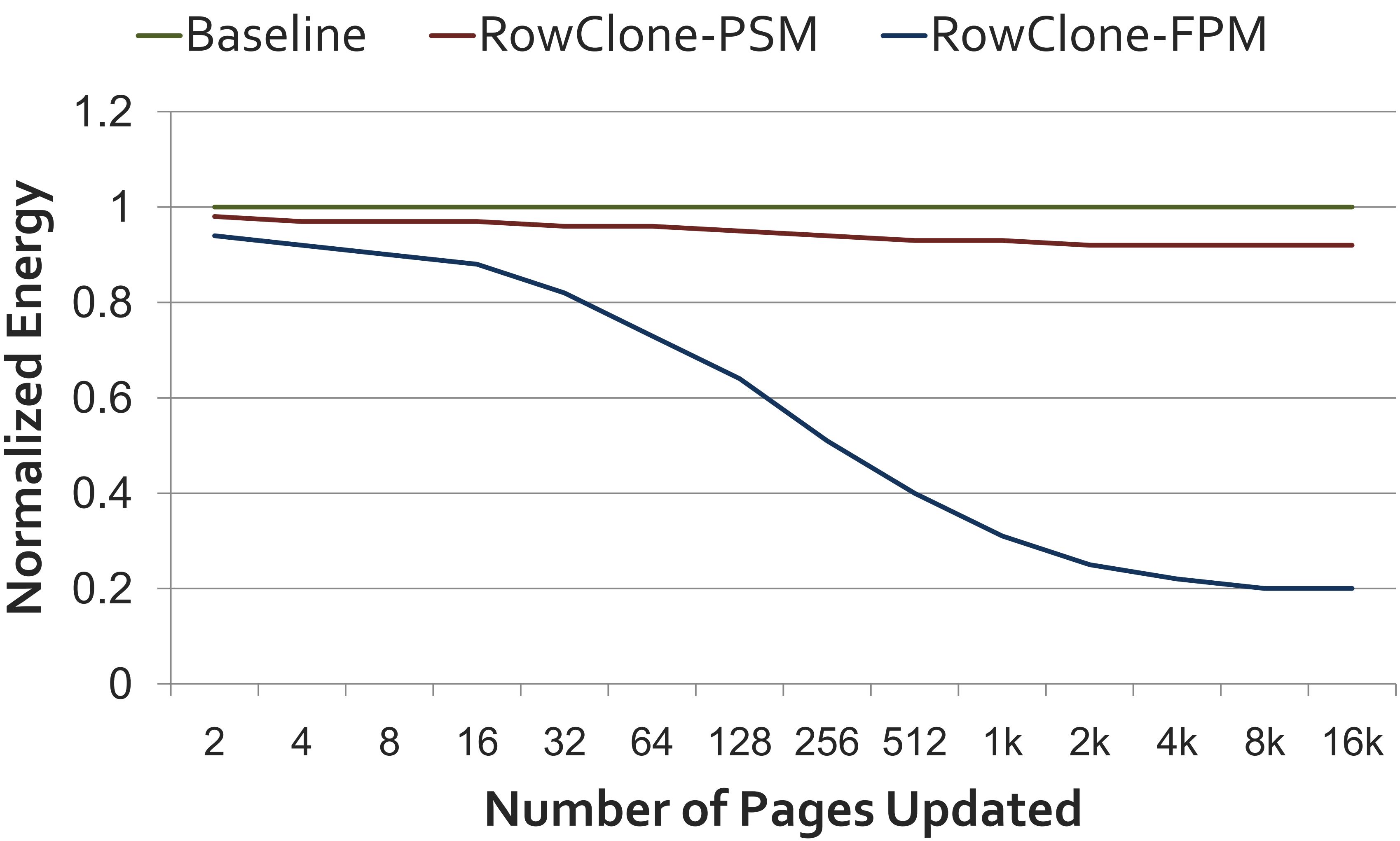
Forkbench – Fraction of Memory Traffic



Forkbench – Performance



Forkbench – Energy



Comparison to Prior Work

- Copy engines (Zhao et al. 2005, Jiang et al. 2009)
  - Addresses cache pollution, pipeline stalls due to copy
  - But requires data transfer over the memory channel

- IRAM (Patterson et al. 1997)
  - Compute + memory using same technology
  - Exploit high DRAM bandwidth
  - Goal: Wider range of SIMD operations
  - High cost

Why is FPM not done today?

- Copy/Initialization is important
  - But not well known
- Opportunity to perform in DRAM
  - Not well known
- This paper: Proof of concept
  - More challenges to be addressed