Seminar in Computer Architecture

Meeting 2: Example Review: RowClone

Prof. Onur Mutlu

ETH Zürich

Fall 2020

24 September 2020

Suggested Paper Discussion Format

- Problem & Goal
- Key Ideas/solution
- Novelty
- Mechanisms & Implementation
- Major Results
- Takeaways/Conclusions

~20-25 minute Summary

- Strengths
- Weaknesses
- Alternatives
- New ideas/problems
- Brainstorming and Discussion

~10 min Critique plus
~10 min Discussion

Presentation Schedule

- We will have 11 sessions of presentations

- 2 presentations in each of the 11 sessions
  - Max 50 minutes total for each presentation+discussion
  - We will take the entire 2 hours in each meeting

- Each presentation
  - One student presents one paper and leads discussion
  - Max 25 minute summary+analysis
  - Max 10 minute critique
  - Max 10 minute discussion+brainstorming+feedback
  - Should follow the suggested guidelines

Algorithm for Presentation Preparation

- Study Lecture 1b again for presentation guidelines

- Read and analyze your paper thoroughly
  - Discuss with anyone you wish + use any resources
- Prepare a draft presentation based on guidelines

- Meet mentor(s) and get feedback
  - Revise the presentation and delivery
- Meet mentor(s) again and get further feedback
  - Revise the presentation and delivery
- Meetings are mandatory – you have to schedule them with your assigned mentor(s). We may suggest meeting times.
- Practice, practice, practice

Example Paper Presentations

Learning by Example

- A great way of learning
- We will do at least one today

Structure of the Presentation

- Background, Problem & Goal
- Novelty
- Key Approach and Ideas
- Mechanisms (in some detail)
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Background, Problem & Goal

Novelty

Key Approach and Ideas

Mechanisms (in some detail)

Key Results: Methodology and Evaluation

Summary

Strengths

Weaknesses

Thoughts and Ideas

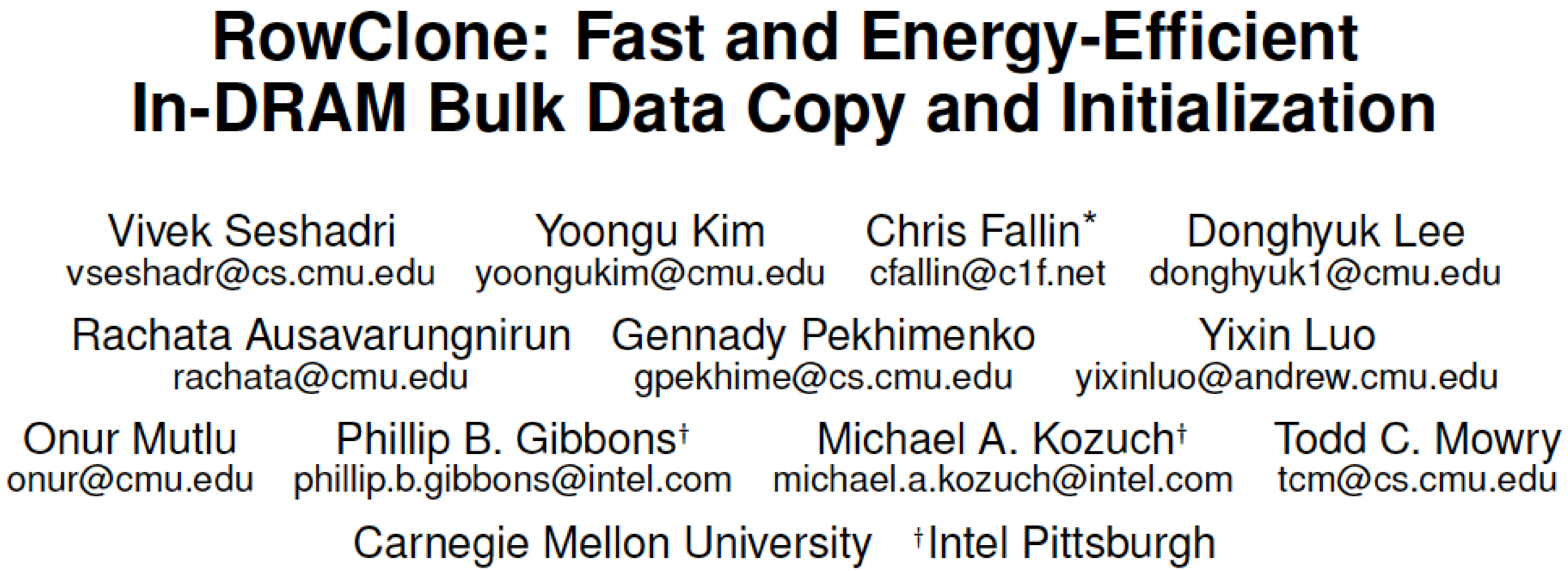
Takeaways

Open Discussion

Example Paper Presentation

Let’s Review This Paper

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry, **"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"** *Proceedings of the 46th International Symposium on Microarchitecture (**MICRO**)*, Davis, CA, December 2013. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]



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

Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com
Michael A. Kozuch† michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu

Carnegie Mellon University †Intel Pittsburgh

RowClone

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

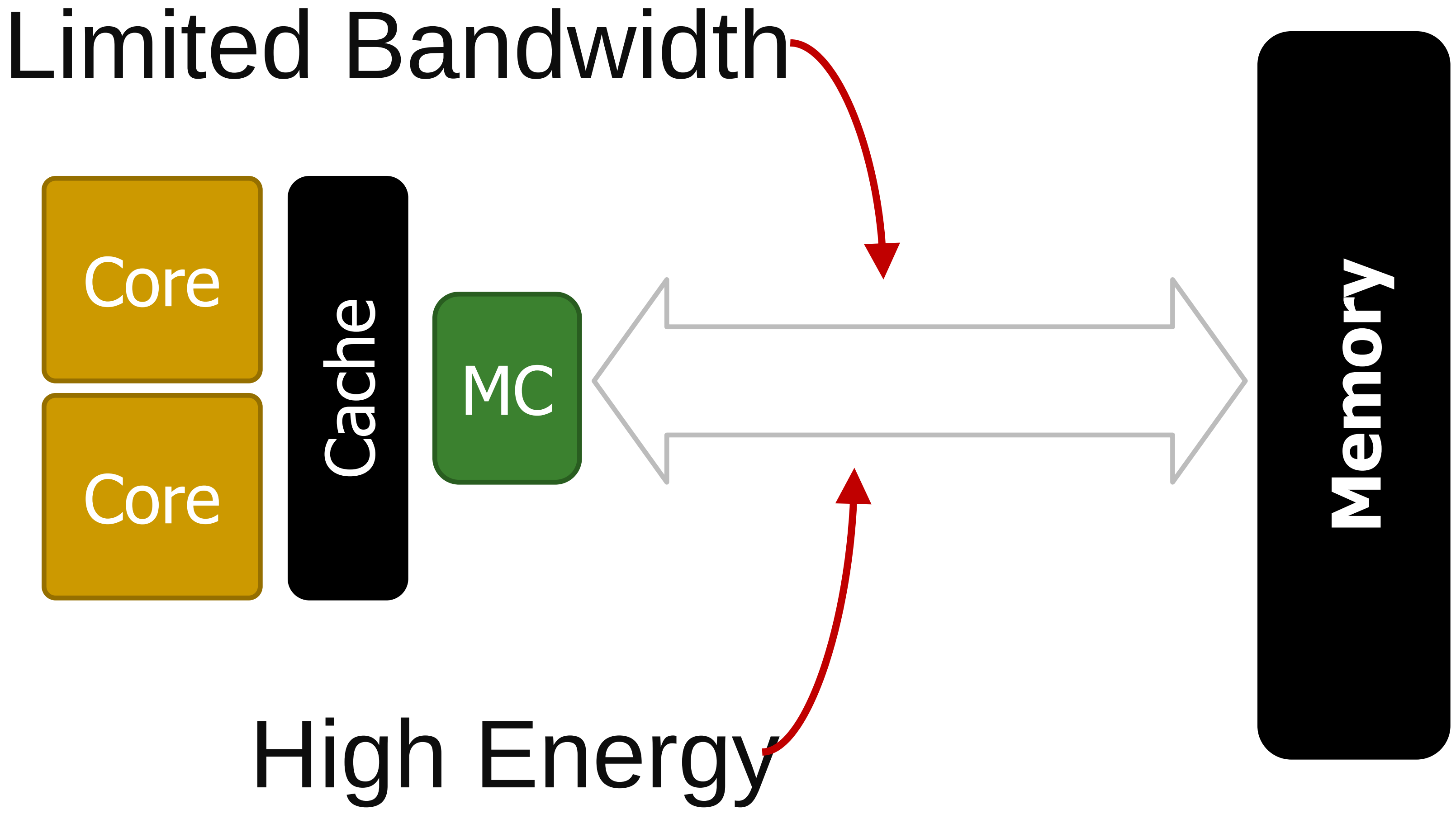
Vivek Seshadri

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

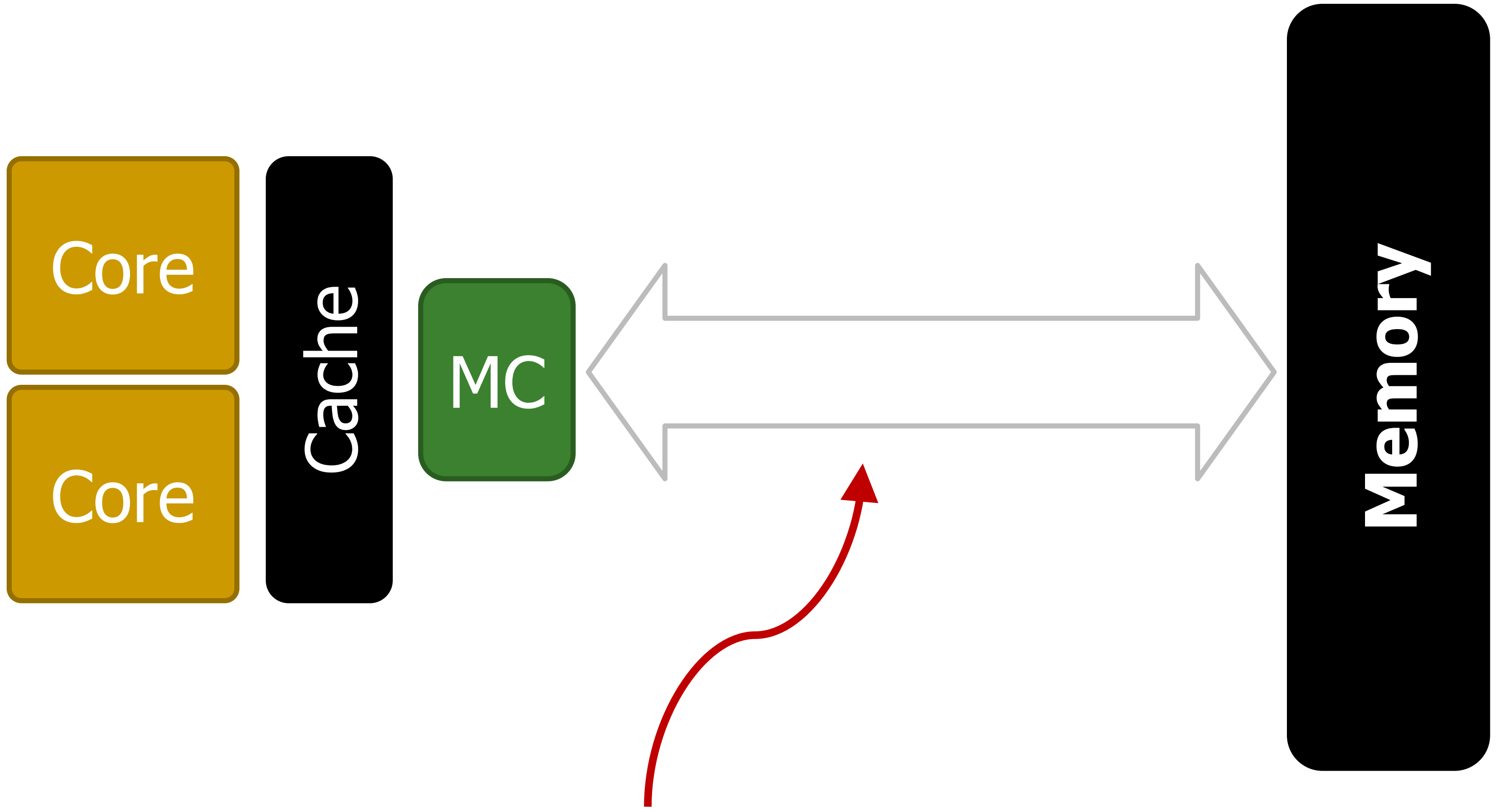
SAFARI Carnegie Mellon intel

Background, Problem & Goal

Memory Channel – Bottleneck



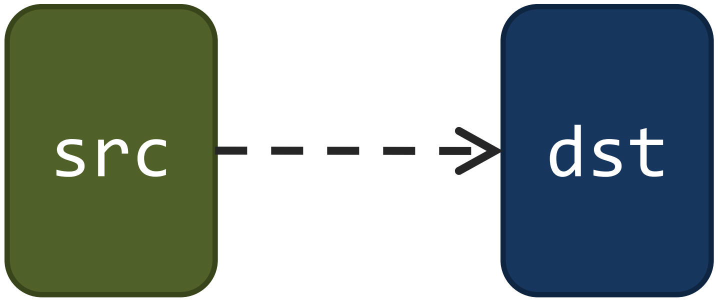
Goal: Reduce Memory Bandwidth Demand



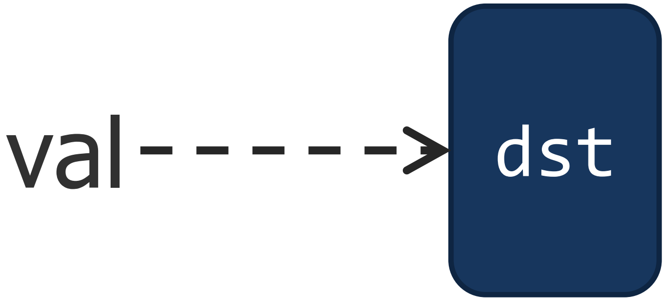
Reduce unnecessary data movement

Bulk Data Copy and Initialization

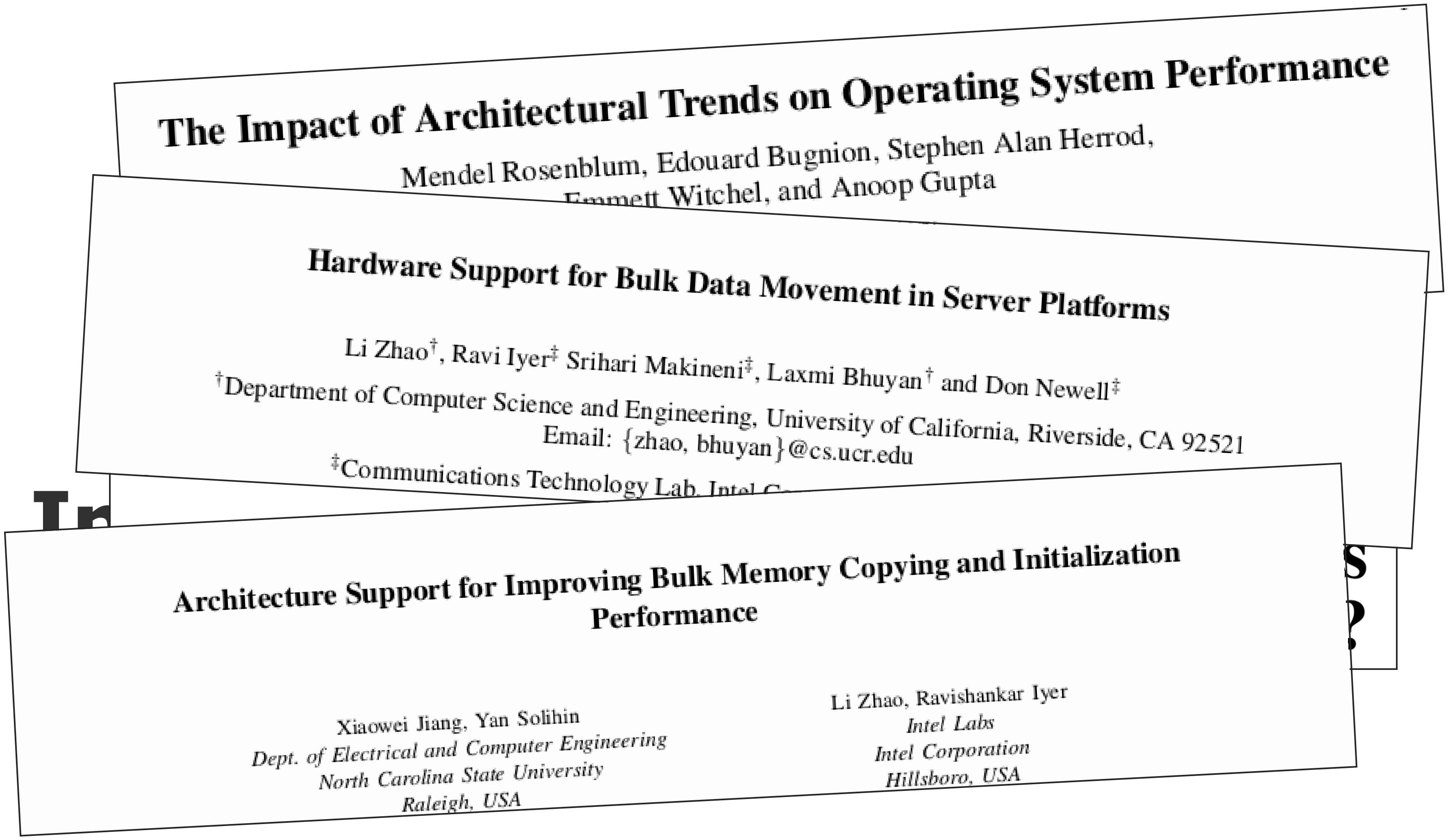
Bulk Data Copy



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

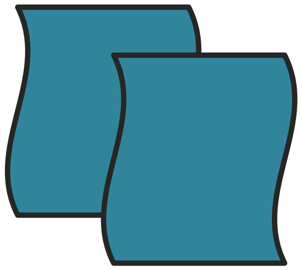
Architecture Support for Improving Bulk Memory Copying and Initialization Performance

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

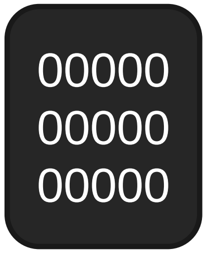
Li Zhao, Ravishankar Iyer
Intel Labs
Intel Corporation
Hillsboro, USA

Bulk Data Copy and Initialization

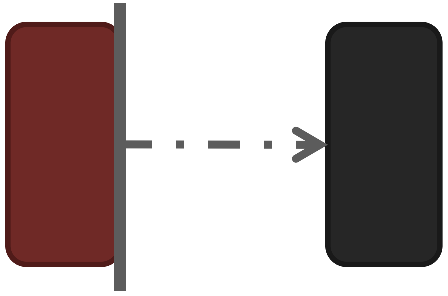
memmove & *memcpy*: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking

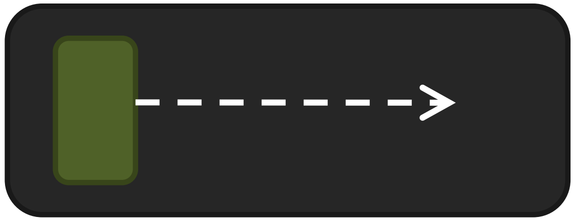


Zero initialization (e.g., security)



Checkpointing

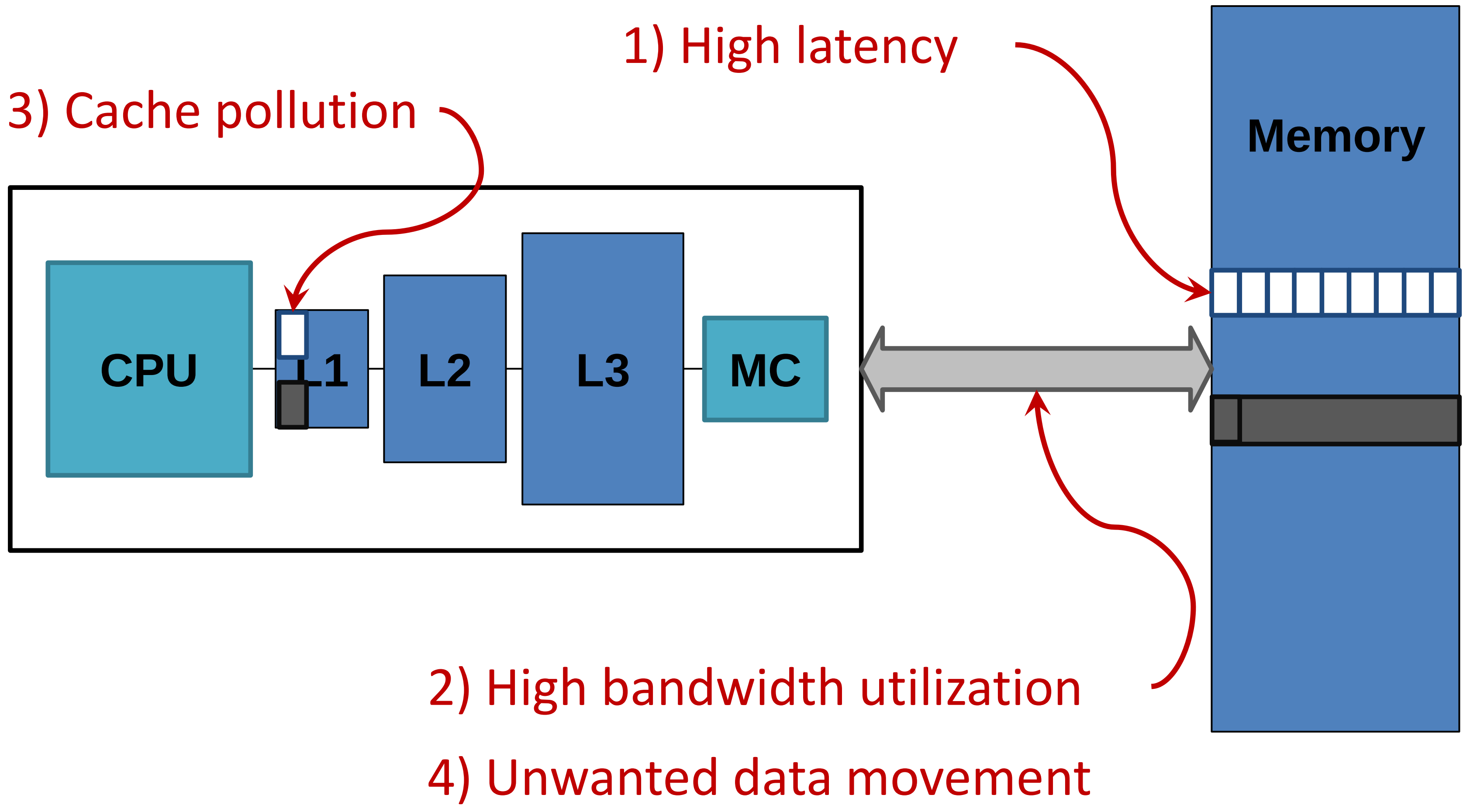
VM Cloning Deduplication



Page Migration

. . .
Many more

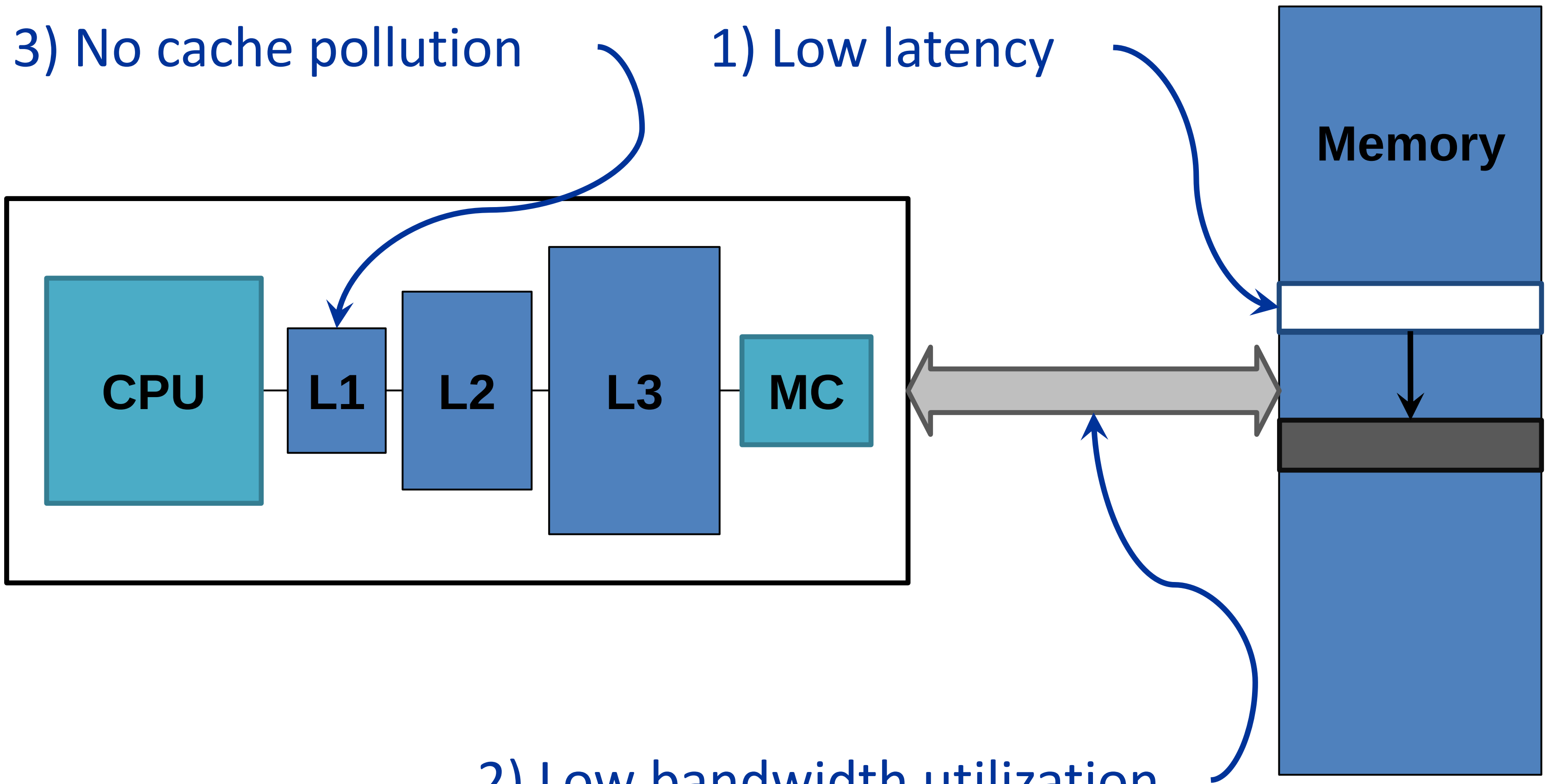
Shortcomings of Today's Systems



1046ns, 3.6uJ (for 4KB page copy via DMA)

Novelty, Key Approach, and Ideas

RowClone: In-Memory Copy



2) Low bandwidth utilization

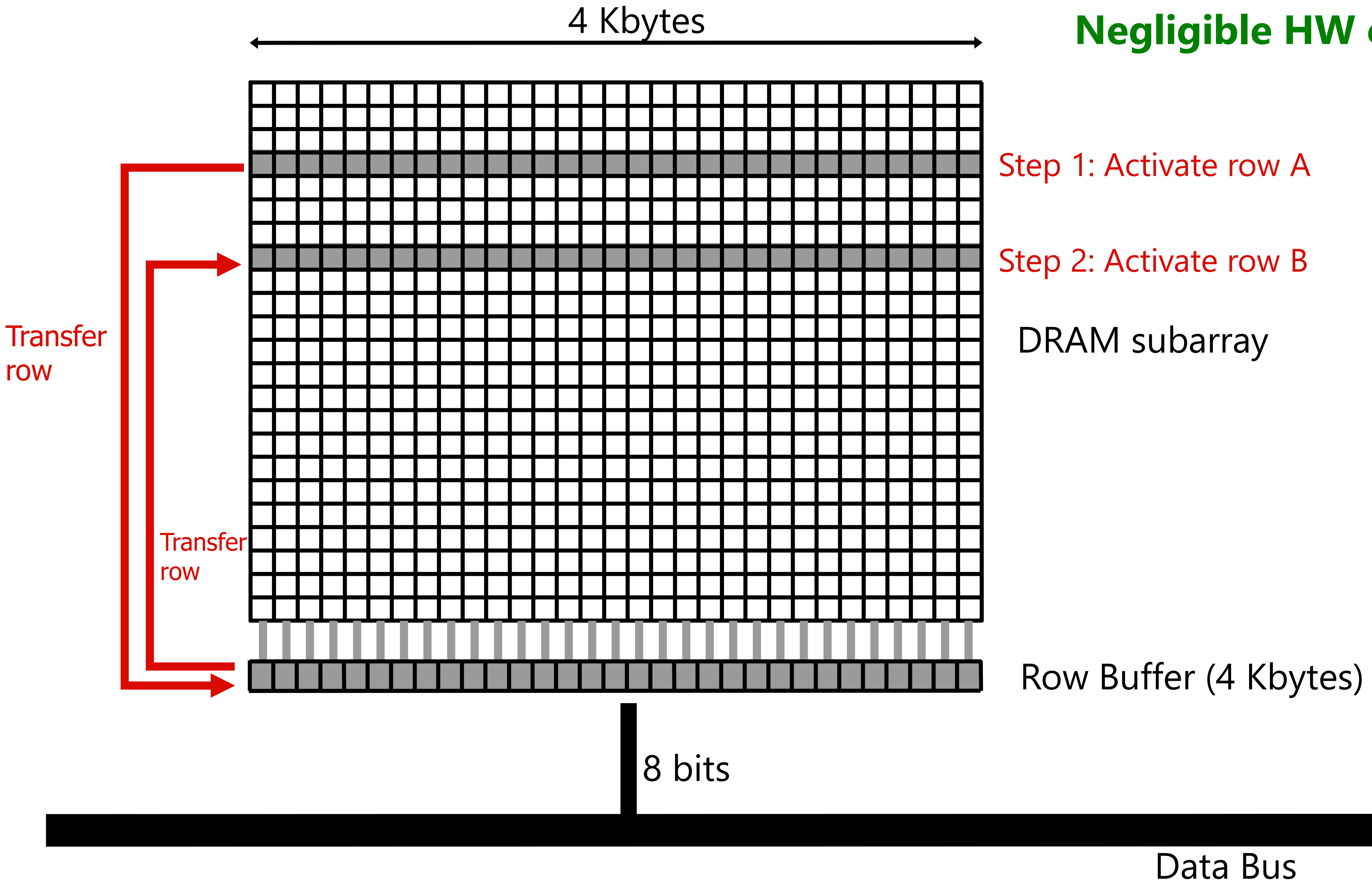
4) No unwanted data movement

1046ns, 3.6uJ → 90ns, 0.04uJ

RowClone: In-DRAM Row Copy

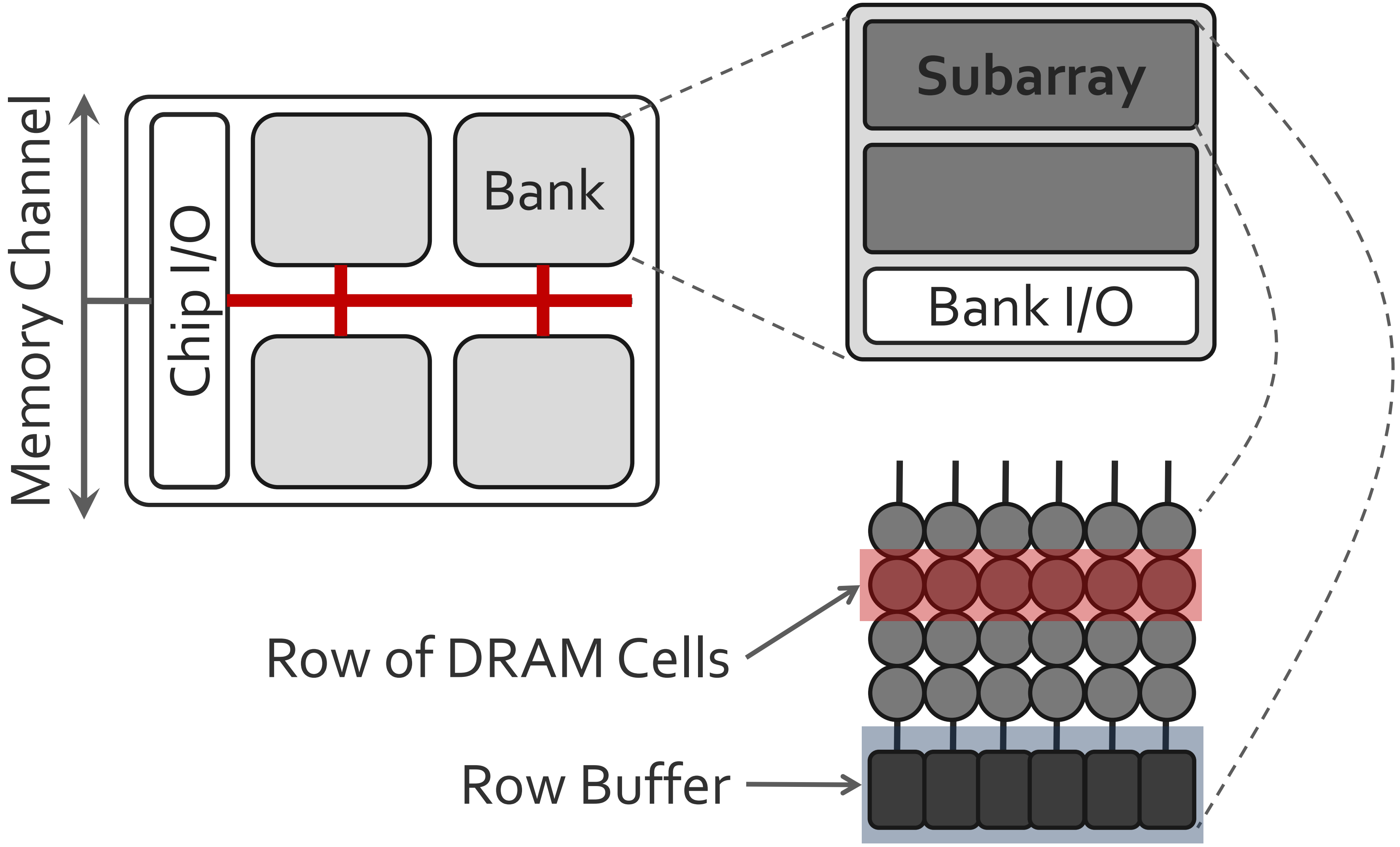
Idea: Two consecutive ACTivates

Negligible HW cost



Mechanisms (in some detail)

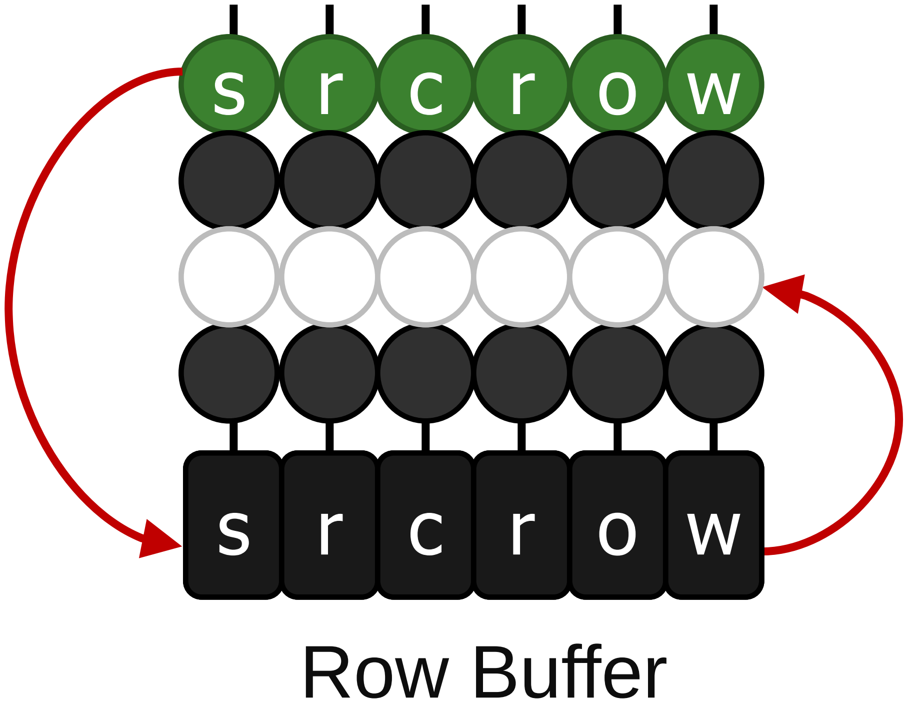
DRAM Chip Organization



RowClone Types

- Intra-subarray RowClone (row granularity)
  - Fast Parallel Mode (FPM)
- Inter-bank RowClone (byte granularity)
  - Pipelined Serial Mode (PSM)
- Inter-subarray RowClone

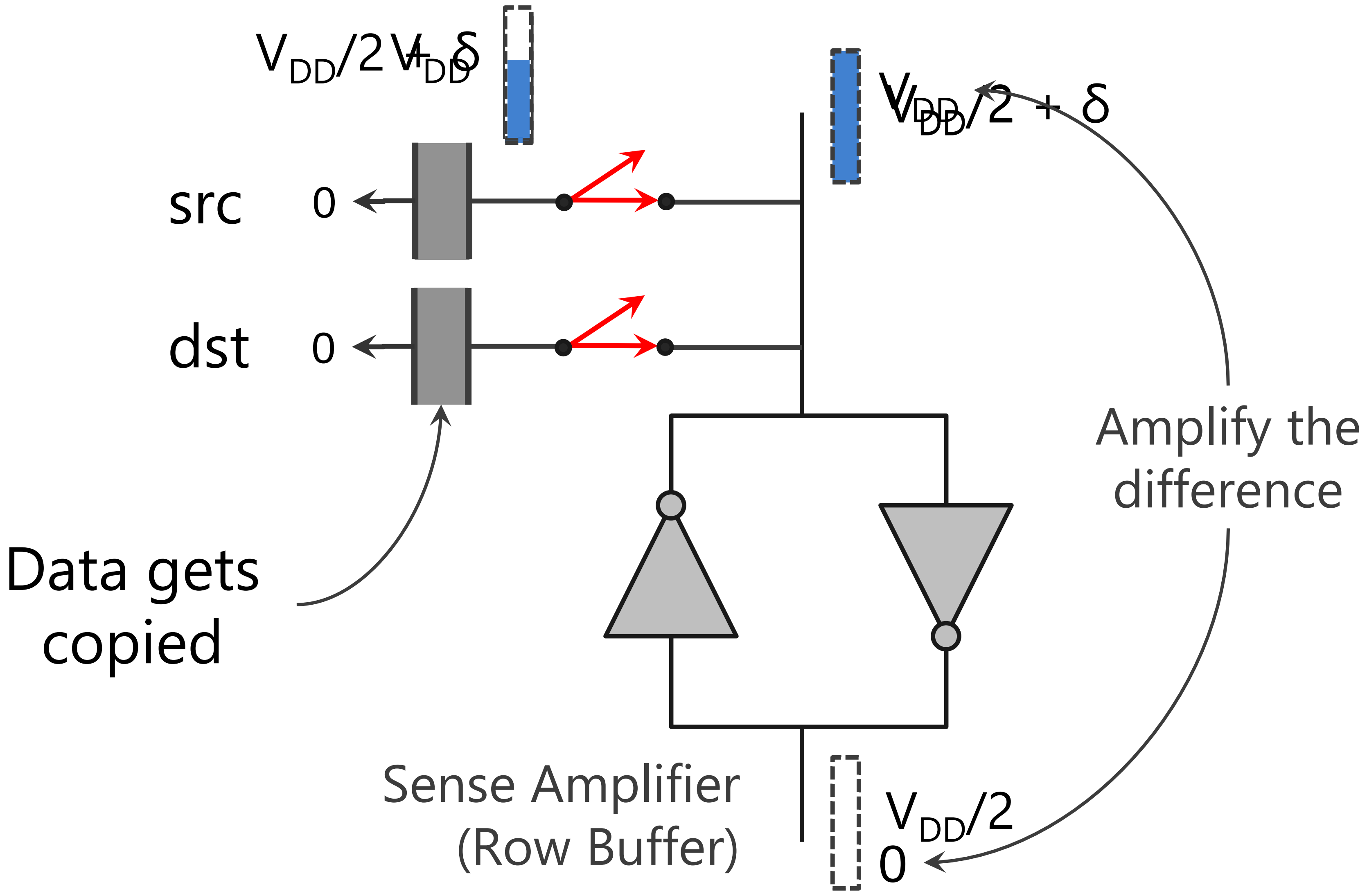
RowClone: Fast Parallel Mode (FPM)



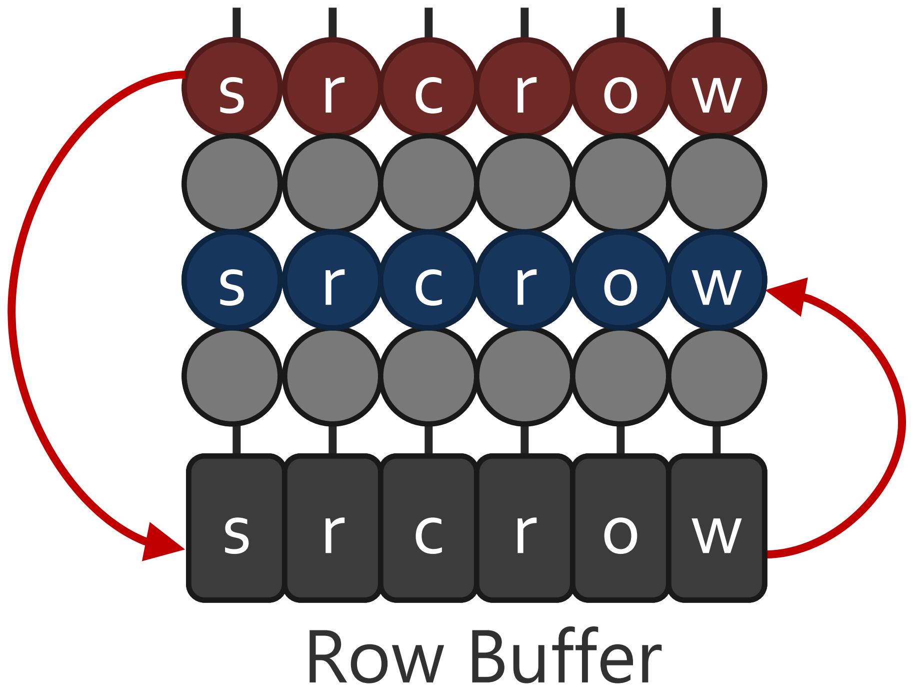
✓ **1. Source row to row buffer**

? **2. Row buffer to destination row**

RowClone: Intra-Subarray (I)



RowClone: Intra-Subarray (II)



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

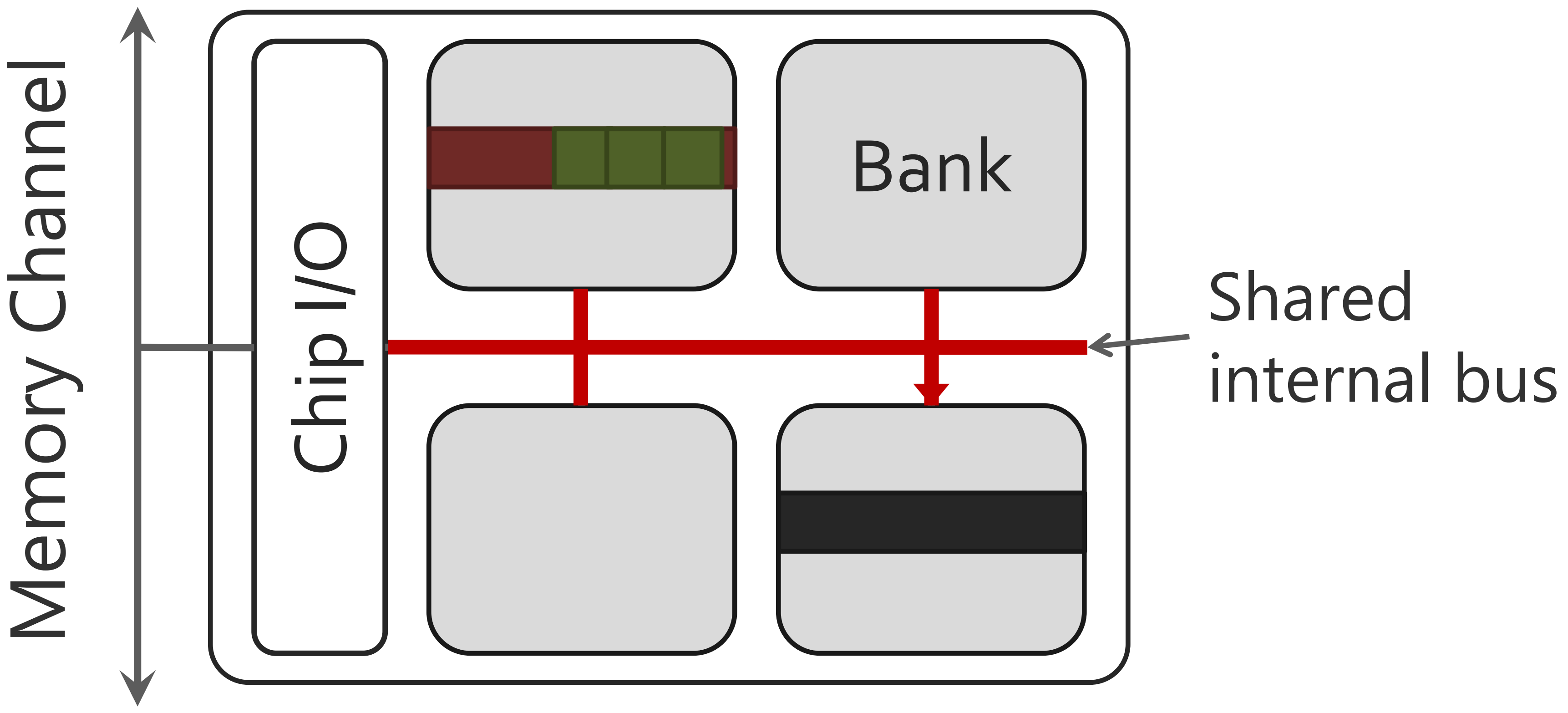
No bandwidth consumption

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: Inter-Bank

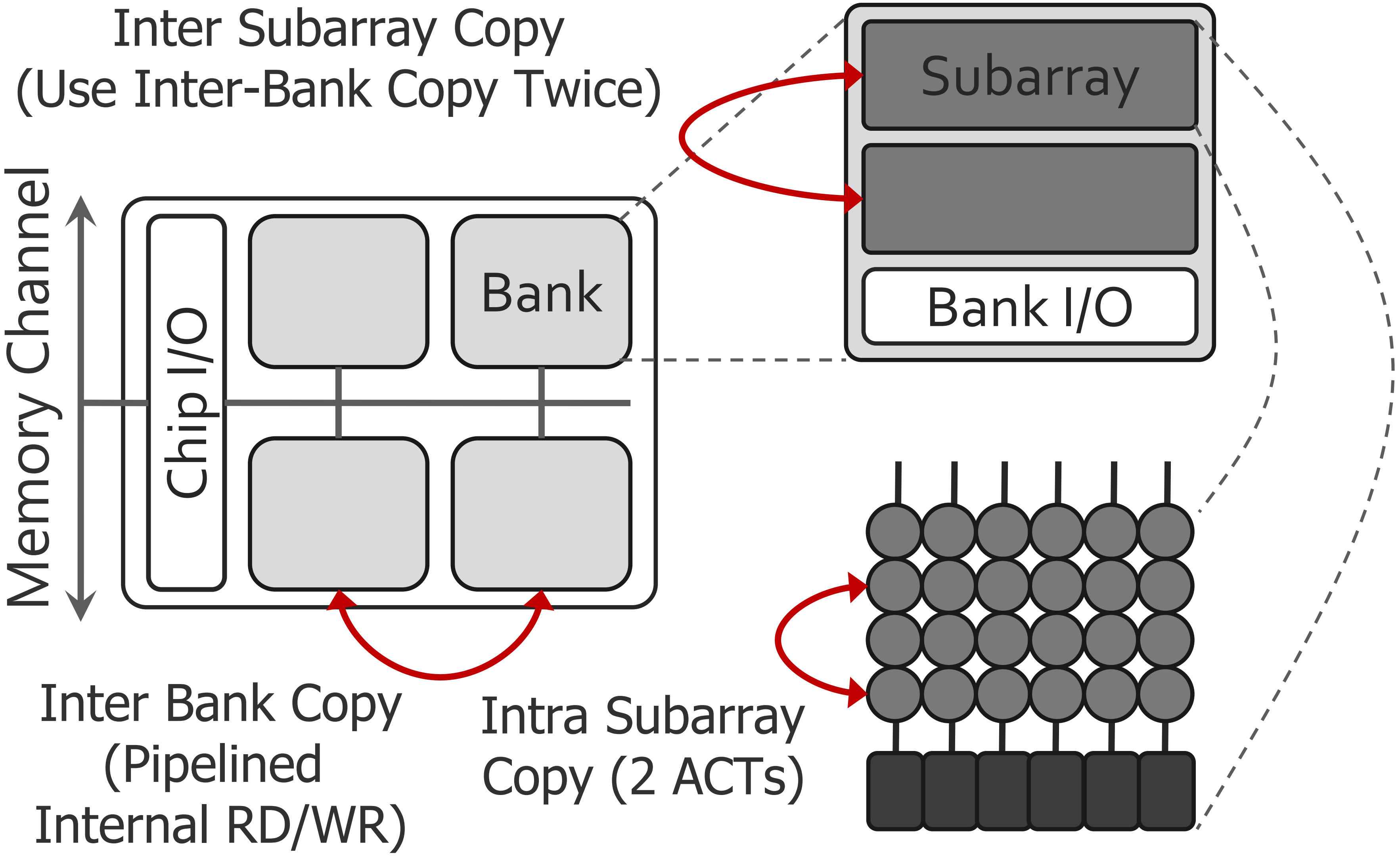


Overlap the latency of the read and the write

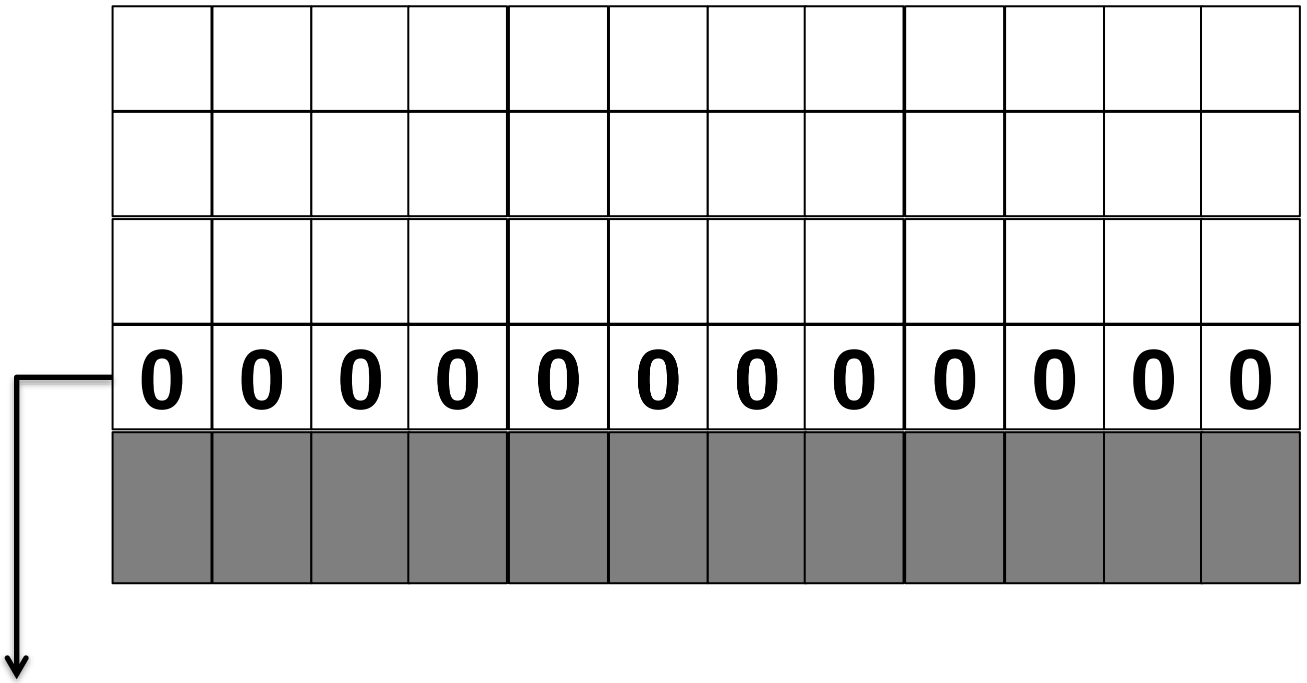
1.9X latency reduction, 3.2X energy reduction

Generalized RowClone

0.01% area cost



RowClone: Fast Row Initialization



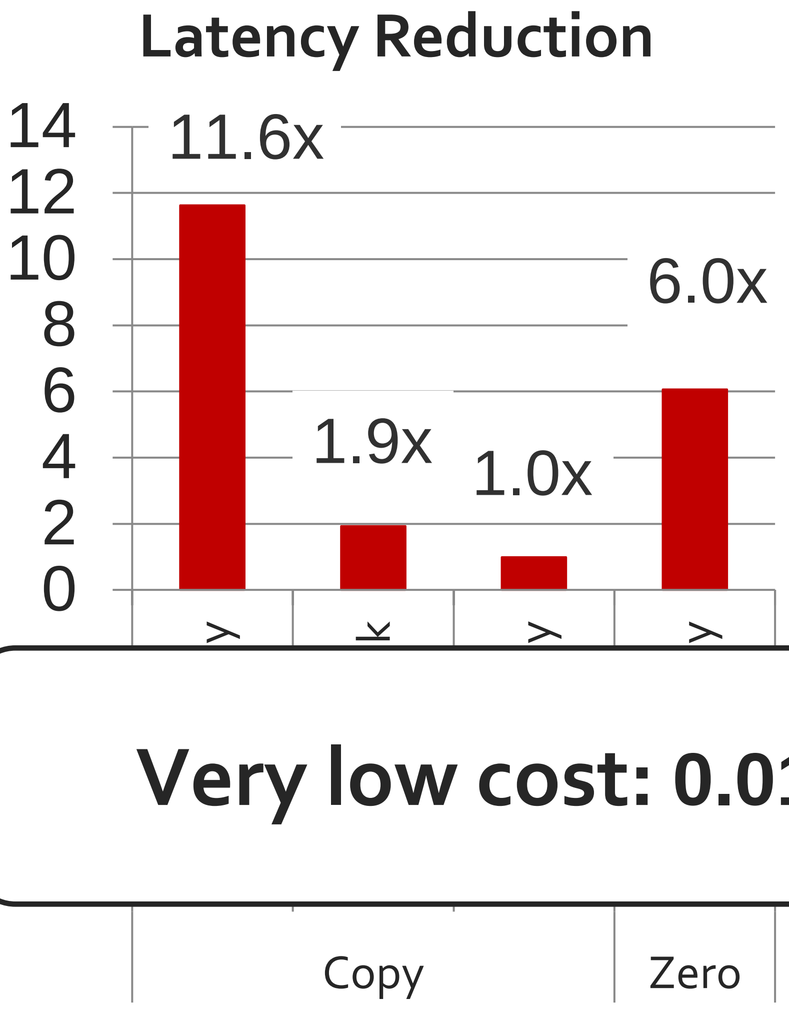
Fix a row at Zero
(0.5% loss in capacity)

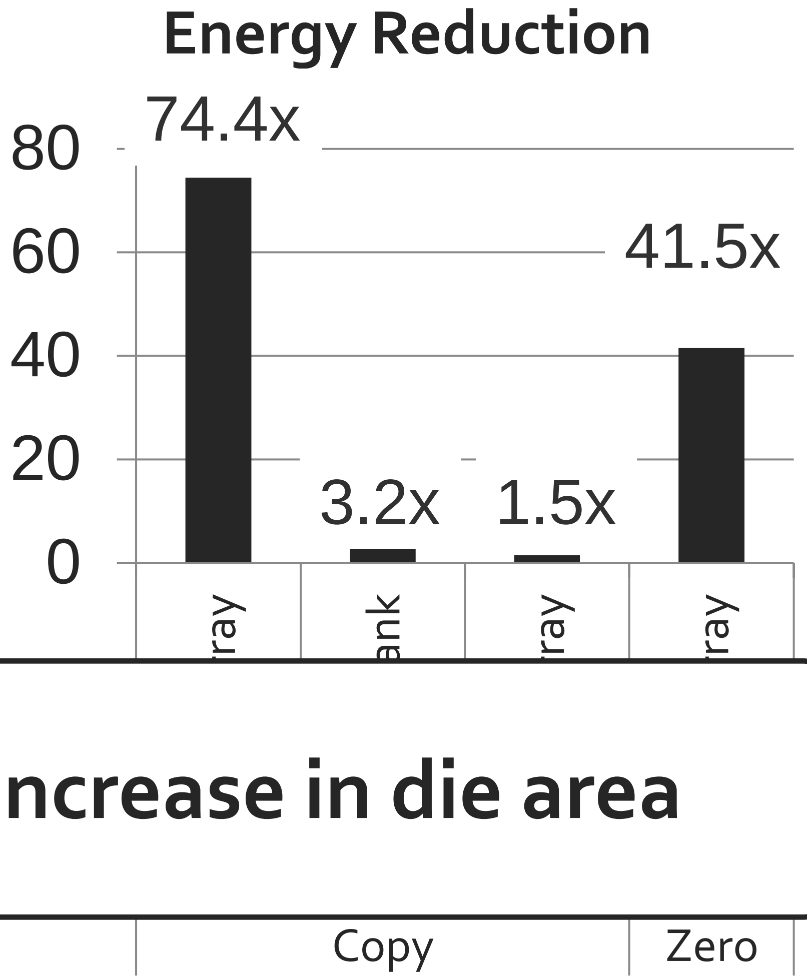
RowClone: Bulk Initialization

- 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

RowClone: Latency & Energy Benefits



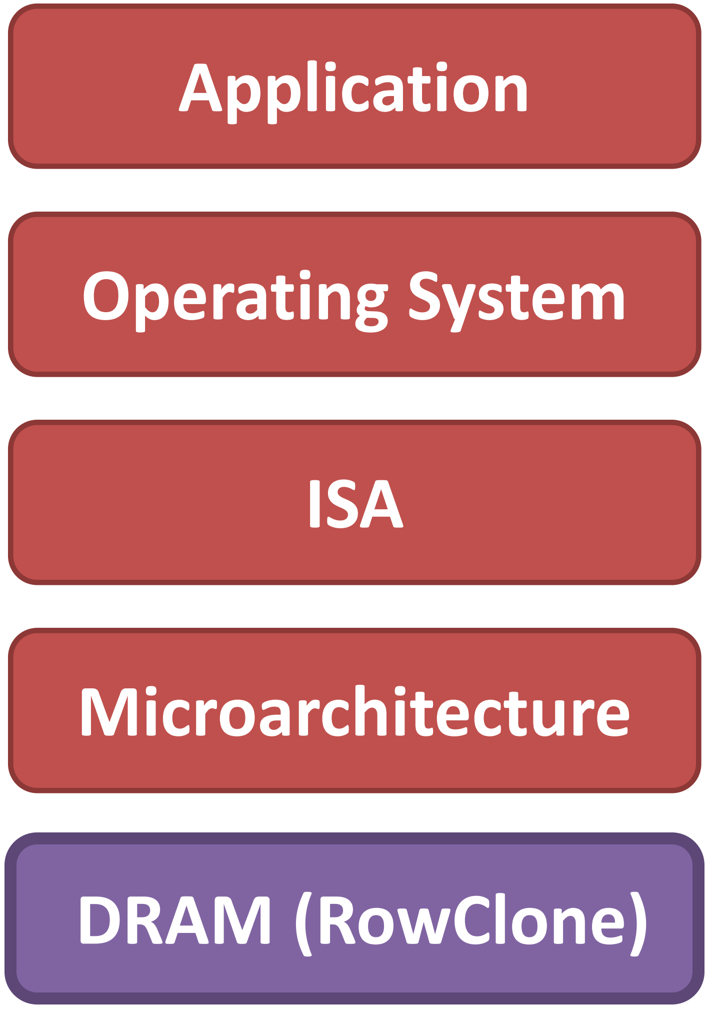


Very low cost: 0.01% increase in die area

SAFARI

System Design to Enable RowClone

End-to-End System Design



How to communicate occurrences of bulk copy/initialization across layers?

How to ensure cache coherence?

How to maximize latency and energy savings?

How to handle data reuse?

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

Key Results: Methodology and Evaluation

Methodology

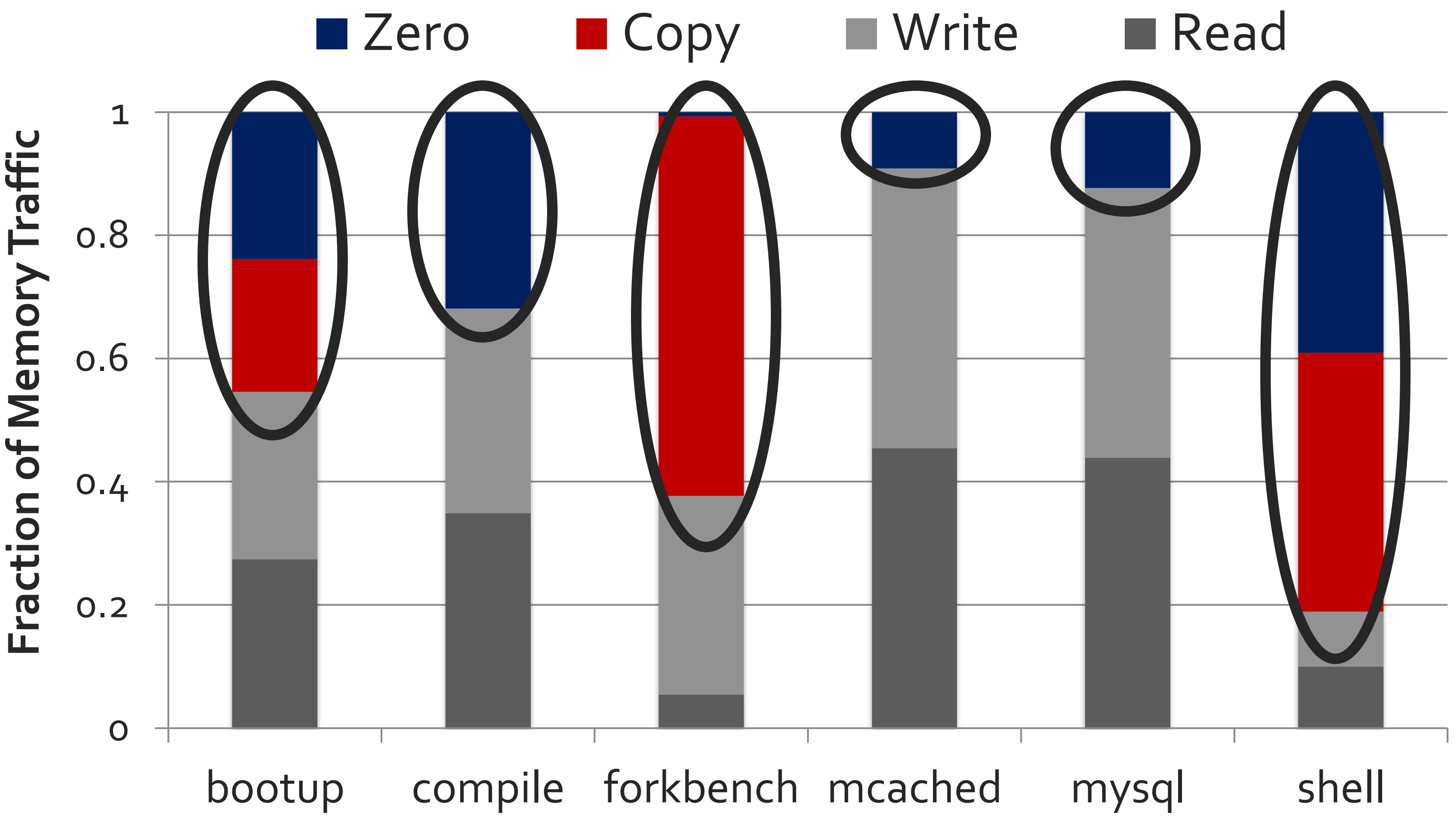
- Out-of-order multi-core simulator
- 1MB/core last-level cache
- Cycle-accurate DDR3 DRAM simulator
- 6 Copy/Initialization intensive applications

  +SPEC CPU2006 for multi-core
- Performance
  - Instruction throughput for single-core
  - Weighted Speedup for multi-core

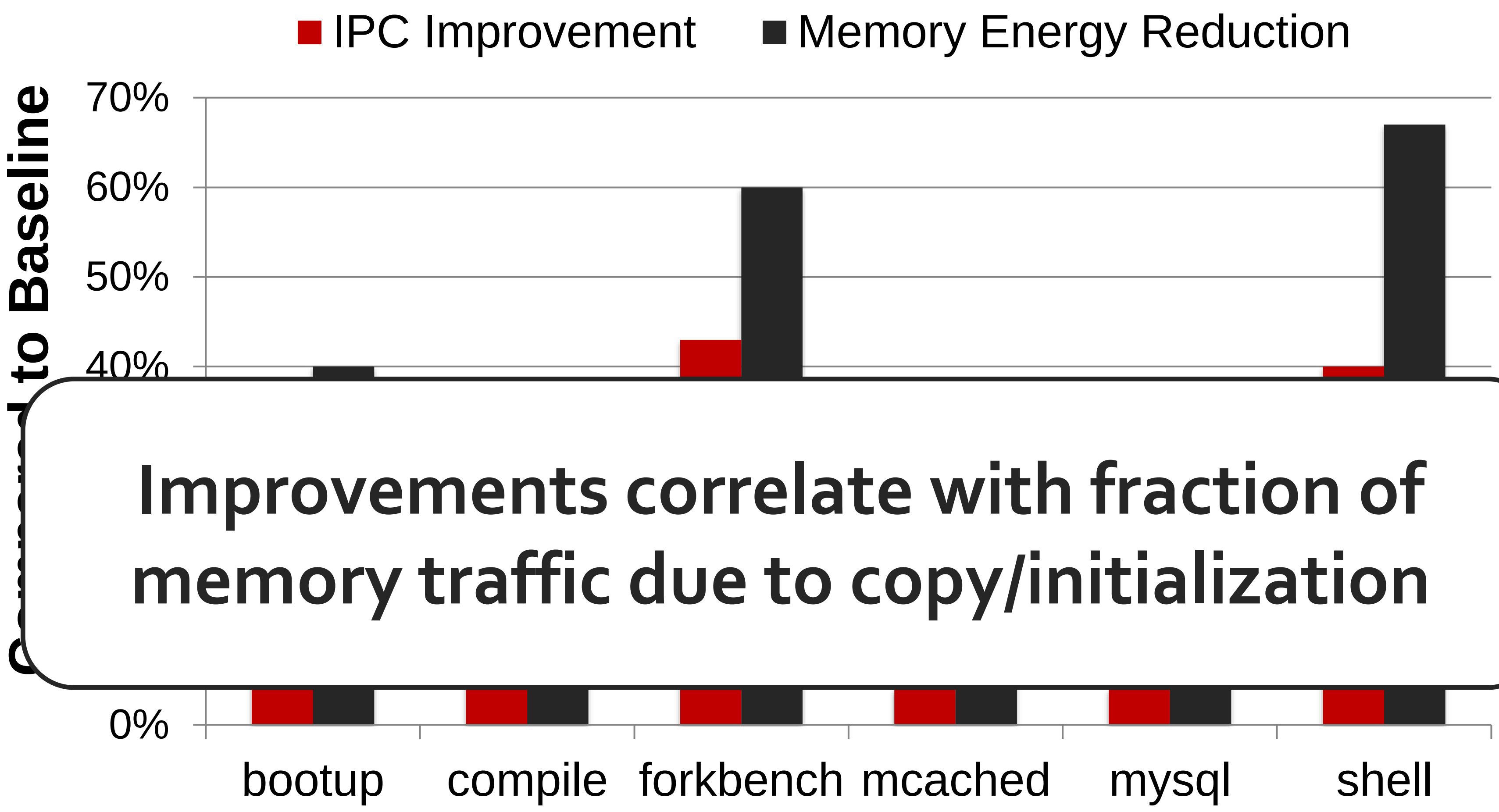
Copy/Initialization Intensive Applications

- **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)

Copy and Initialization in Workloads



Single-Core – Performance and Energy

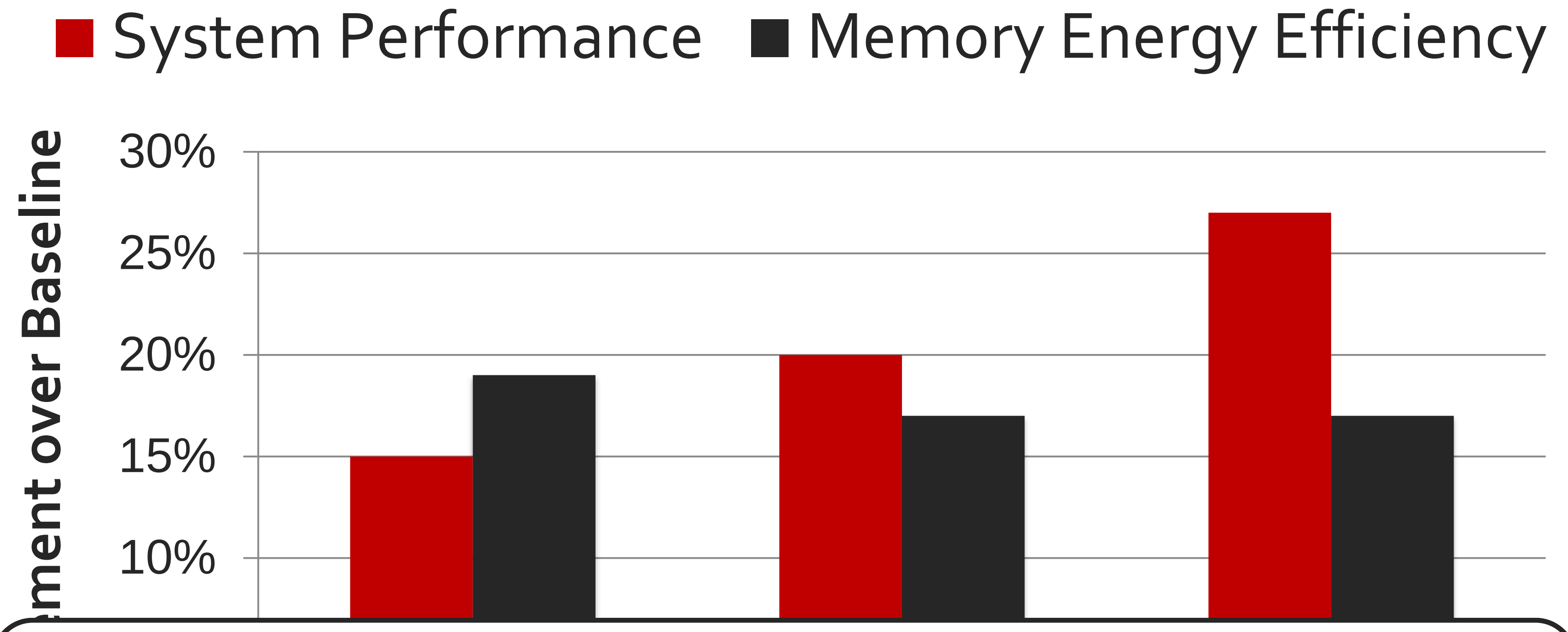


Improvements correlate with fraction of memory traffic due to copy/initialization

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

Summary

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)

Strengths

Strengths of the Paper

- Simple, novel mechanism to solve an important problem
- Effective and low hardware overhead
- Intuitive idea!
- Greatly improves performance and efficiency (assuming data is mapped nicely)
- Seems like a clear win for data initialization (without mapping requirements)
- Makes software designer's life easier
    - If copies are 10x-100x cheaper, how to design software?

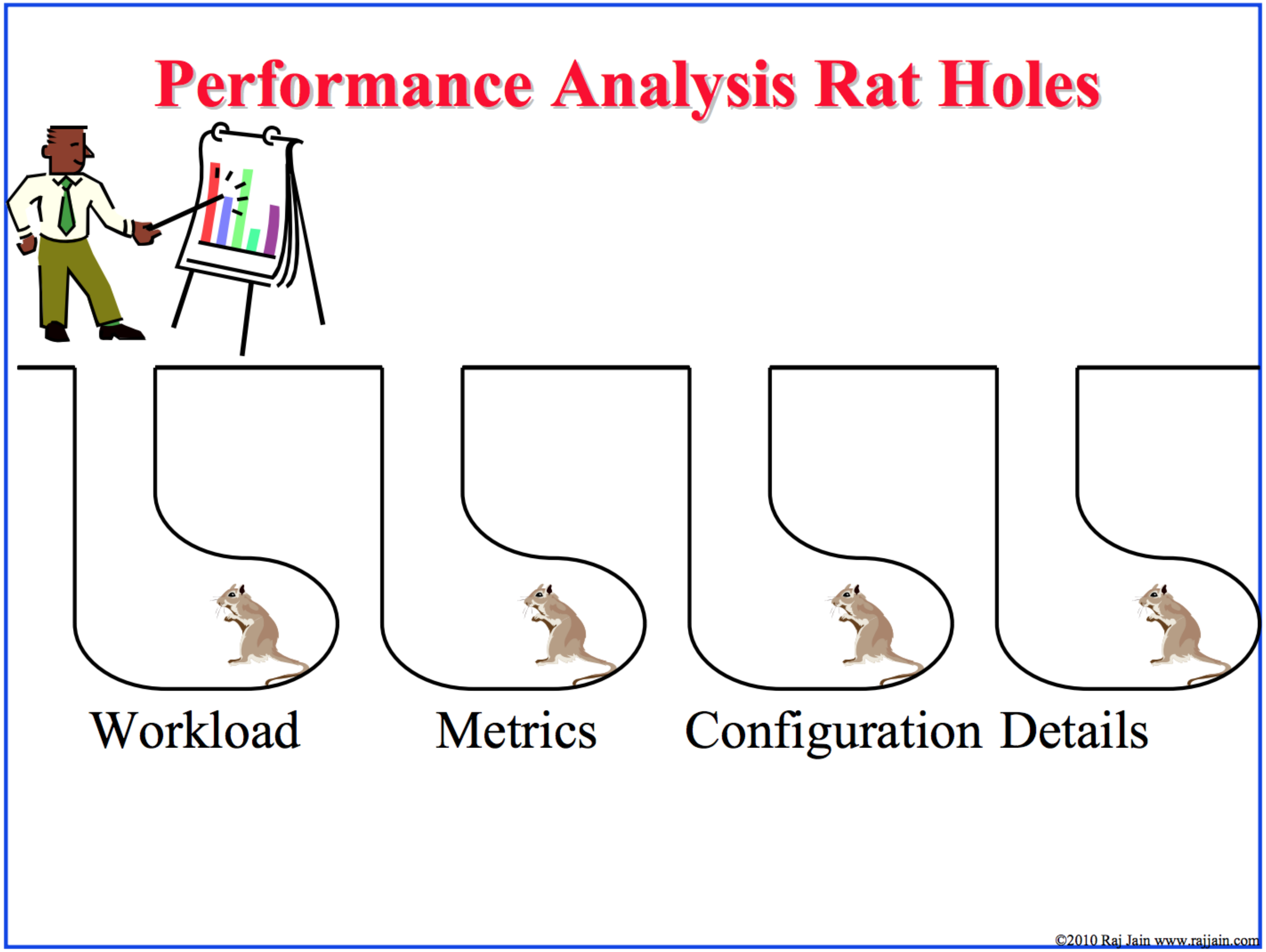
- Paper tackles many low-level and system-level issues
- Well-written, insightful paper

Weaknesses

Weaknesses

- Requires data to be mapped in the same subarray to deliver the largest benefits
  - Helps less if data movement is not within a subarray
  - Does not help if data movement is across DRAM channels
- Inter-subarray copy is very inefficient
- Causes many changes in the system stack
  - End-to-end design spans applications to circuits
  - Software-hardware cooperative solution might not always be easy to adopt
- Cache coherence and data reuse cause real overheads
- Evaluation is done solely in simulation
- Evaluation does not consider multi-chip systems
- Are these the best workloads to evaluate?

Recall: Try to Avoid Rat Holes


Performance Analysis Rat Holes
Workload
Metrics
Configuration
Details
©2010 Raj Jain www.rajjain.com

Source: https://www.cse.wustl.edu/~jain/iucee/ftp/k_10adp.pdf

Thoughts and Ideas

Extensions and Follow-Up Work

- Can this be improved to do faster inter-subarray copy?
  - Yes, see the LISA paper [Chang et al., HPCA 2016]

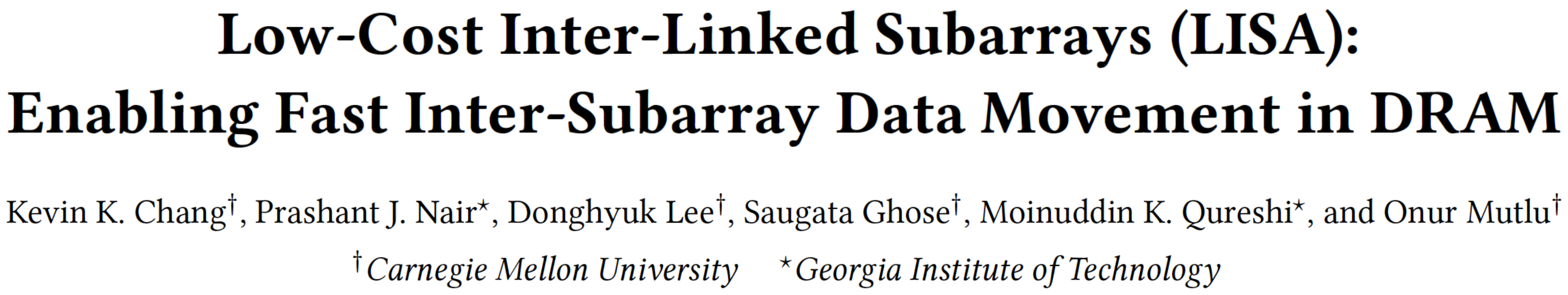
- Can we enable data movement at smaller granularities within a bank?
  - Yes, see the FIGARO paper [Wang et al., MICRO 2020]

- Can this be improved to do better inter-bank copy?
  - Yes, see the Network-on-Memory paper [CAL 2020]

- Can similar ideas and DRAM properties be used to perform computation on data?
  - Yes, see the Ambit paper [Seshadri et al., MICRO 2017]

LISA: Fast Inter-Subarray Data Movement

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
  "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
  *Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (**HPCA**)*, Barcelona, Spain, March 2016.
  [Slides (pptx) (pdf)]
  [Source Code]

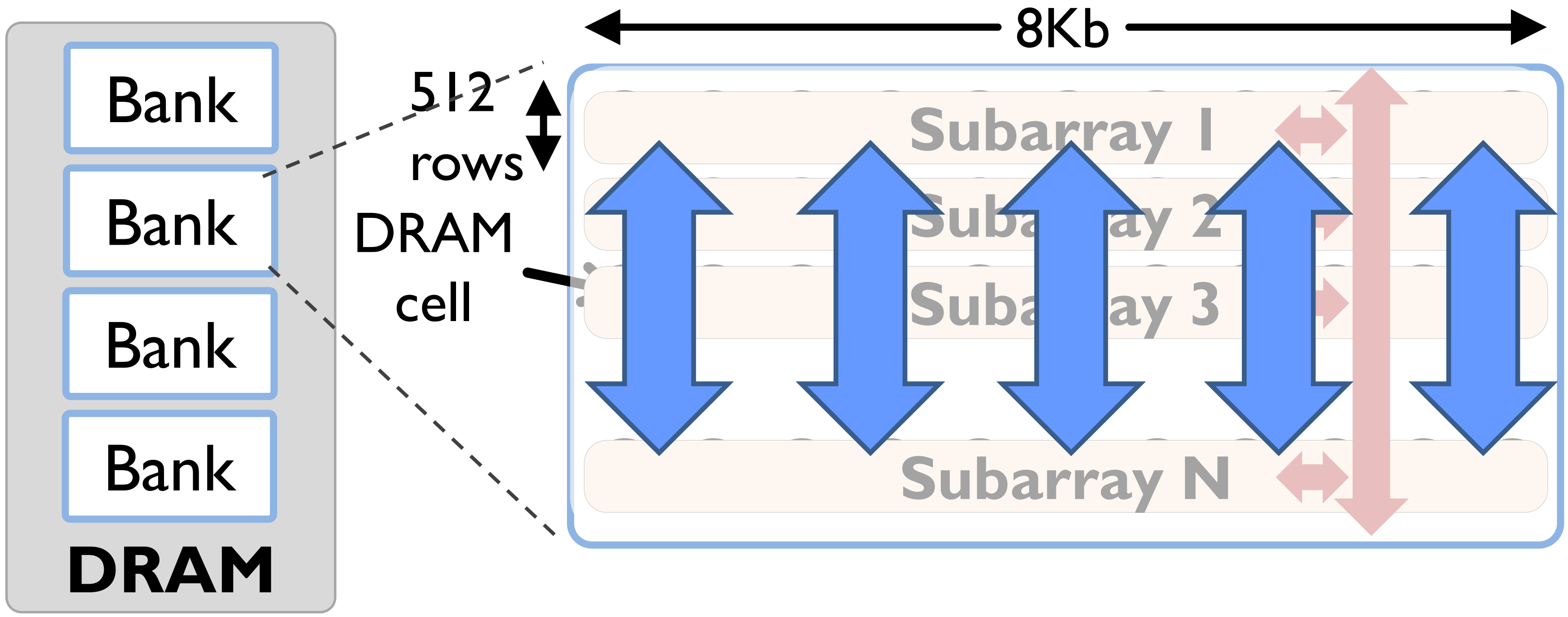


Low-Cost Inter-Linked Subarrays (LISA):
Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang[†], Prashant J. Nair[*], Donghyuk Lee[†], Saugata Ghose[†], Moinuddin K. Qureshi[*], and Onur Mutlu[†]

[†]*Carnegie Mellon University* [*]*Georgia Institute of Technology*

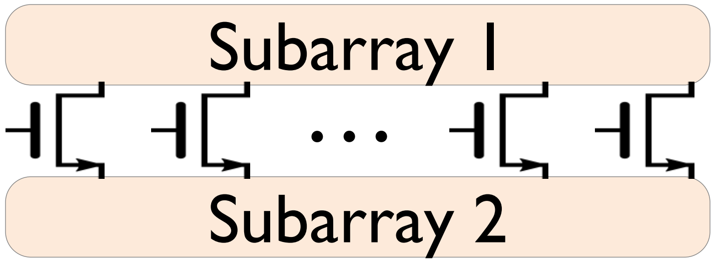
Moving Data Inside DRAM?



Goal: Provide a new substrate to enable wide connectivity between subarrays

Key Idea and Applications

- **Low-cost Inter-linked subarrays (LISA)**
  - Fast bulk data movement between subarrays
  - Wide datapath via isolation transistors: 0.8% DRAM chip area



- LISA is a **versatile substrate** → new applications

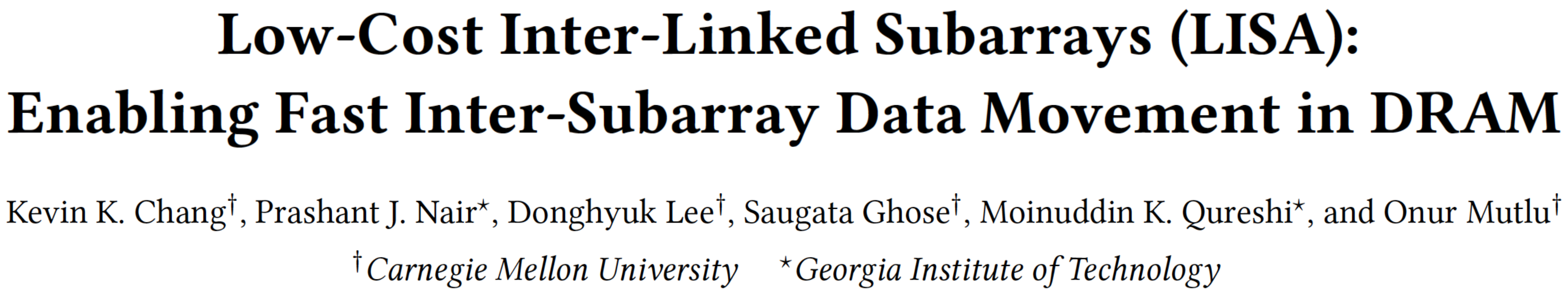
Fast bulk data copy: Copy latency 1.363ms→0.148ms (9.2x)
→ 66% speedup, -55% DRAM energy

In-DRAM caching: Hot data access latency 48.7ns→21.5ns (2.2x)
→ 5% speedup

Fast precharge: Precharge latency 13.1ns→5.0ns (2.6x)
→ 8% speedup

More on LISA

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
"Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (***HPCA***), Barcelona, Spain, March 2016.
[Slides (pptx) (pdf)]
[Source Code]



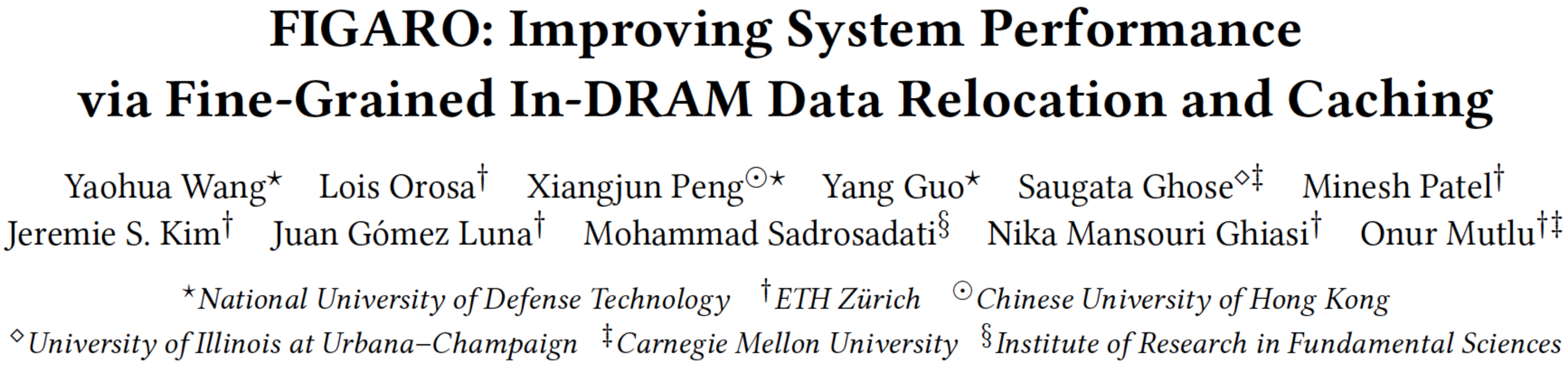
Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang[†], Prashant J. Nair[*], Donghyuk Lee[†], Saugata Ghose[†], Moinuddin K. Qureshi[*], and Onur Mutlu[†]

[†]*Carnegie Mellon University* [*]*Georgia Institute of Technology*

FIGARO: Fine-Grained In-DRAM Copy

- Yaohua Wang, Lois Orosa, Xiangjun Peng, Yang Guo, Saugata Ghose, Minesh Patel, Jeremie S. Kim, Juan Gómez Luna, Mohammad Sadrosadati, Nika Mansouri Ghiasi, and Onur Mutlu, **"FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching"** *Proceedings of the 53rd International Symposium on Microarchitecture* (***MICRO***), Virtual, October 2020.



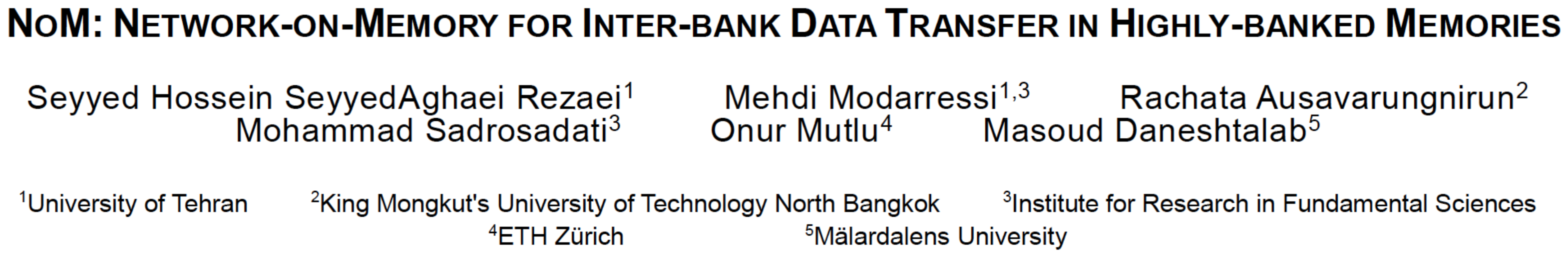
FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang[⋆] Lois Orosa[†] Xiangjun Peng[⊙⋆] Yang Guo[⋆] Saugata Ghose[⋄‡] Minesh Patel[†] Jeremie S. Kim[†] Juan Gómez Luna[†] Mohammad Sadrosadati[§] Nika Mansouri Ghiasi[†] Onur Mutlu[†‡]

[⋆]*National University of Defense Technology* [†]*ETH Zürich* [⊙]*Chinese University of Hong Kong* [⋄]*University of Illinois at Urbana–Champaign* [‡]*Carnegie Mellon University* [§]*Institute of Research in Fundamental Sciences*

Network-On-Memory: Fast Inter-Bank Copy

- Seyyed Hossein SeyyedAghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungnirun, Mohammad Sadrosadati, Onur Mutlu, and Masoud Daneshtalab,
"NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories"
IEEE Computer Architecture Letters ***(CAL)***, to appear in 2020.



NoM: Network-on-Memory for Inter-bank Data Transfer in Highly-banked Memories

Seyyed Hossein SeyyedAghaei Rezaei[1] Mehdi Modarressi[1,3] Rachata Ausavarungnirun[2]
Mohammad Sadrosadati[3] Onur Mutlu[4] Masoud Daneshtalab[5]

[1]University of Tehran [2]King Mongkut's University of Technology North Bangkok [3]Institute for Research in Fundamental Sciences
[4]ETH Zürich [5]Mälardalens University

In-DRAM Bulk Bitwise AND/OR

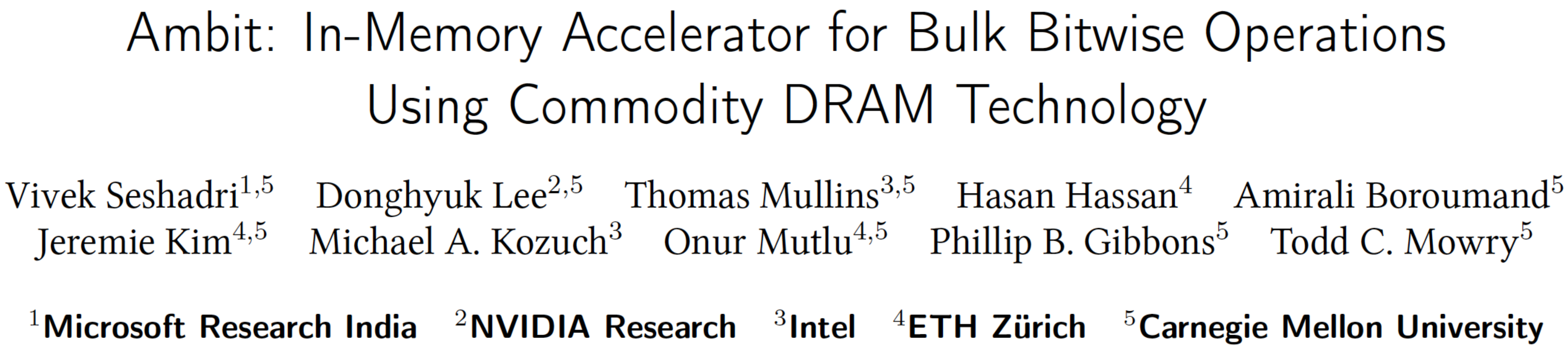
- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
  "Fast Bulk Bitwise AND and OR in DRAM"
  IEEE Computer Architecture Letters (***CAL***), April 2015.

Fast Bulk Bitwise AND and OR in DRAM

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*,
Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*
*Carnegie Mellon University †Intel Pittsburgh

Ambit: Bulk-Bitwise in-DRAM Computation

- Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
Proceedings of the 50th International Symposium on Microarchitecture (***MICRO***), Boston, MA, USA, October 2017.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]

- 

Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

Vivek Seshadri[1,5] Donghyuk Lee[2,5] Thomas Mullins[3,5] Hasan Hassan[4] Amirali Boroumand[5]
Jeremie Kim[4,5] Michael A. Kozuch[3] Onur Mutlu[4,5] Phillip B. Gibbons[5] Todd C. Mowry[5]

[1]Microsoft Research India [2]NVIDIA Research [3]Intel [4]ETH Zürich [5]Carnegie Mellon University

SAFARI

In-DRAM Bulk Bitwise Execution Paradigm

- Vivek Seshadri and Onur Mutlu,
  "In-DRAM Bulk Bitwise Execution Engine"
  Invited Book Chapter in Advances in Computers, to appear in 2020.
  [Preliminary arXiv version]

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch

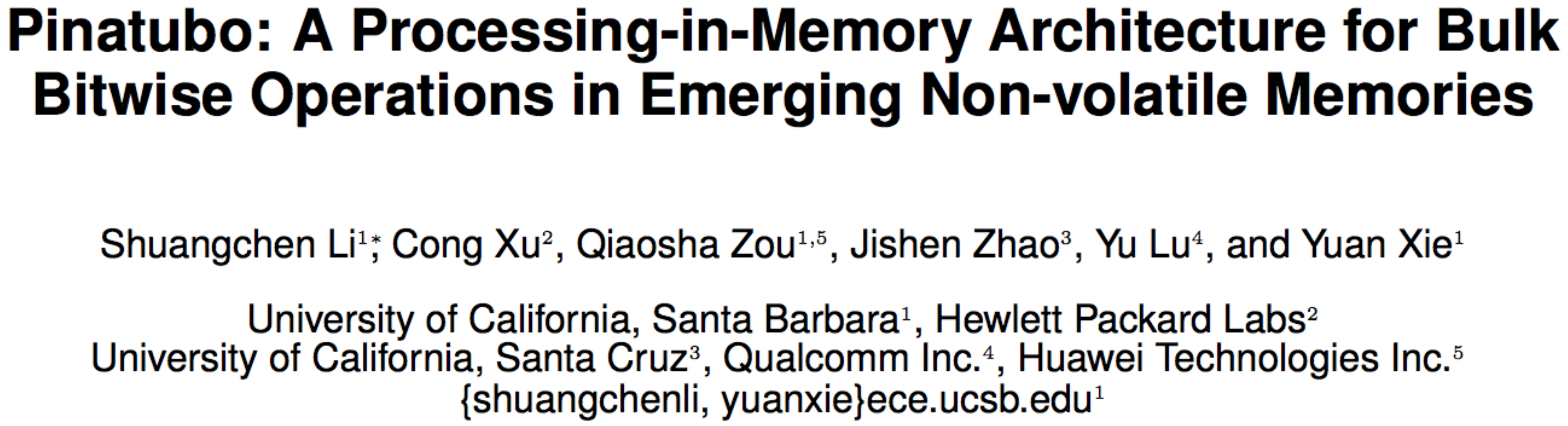
Extensions and Follow-Up Work (II)

- Can this idea be evaluated on a real system? How?
  - Yes, see the Compute DRAM paper [MICRO 2019]

- Can similar ideas be used in other types of memories? Phase Change Memory? RRAM? STT-MRAM?
  - Yes, see the Pinatubo paper [DAC 2016]

- Can we have more efficient solutions to
  - Cache coherence (minimize overhead)
  - Data reuse after copy and initialization

Pinatubo: PCM RowClone and Bitwise Ops



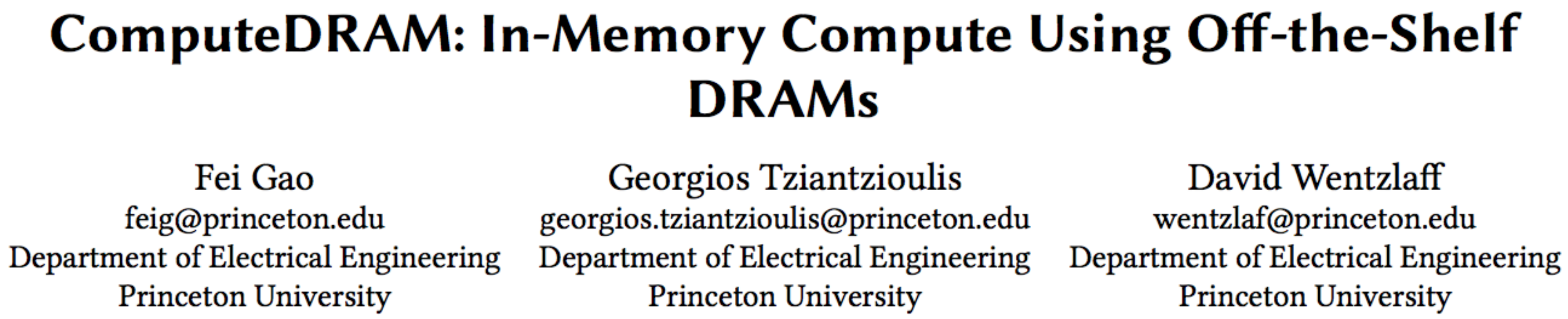
Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li[1]*, Cong Xu[2], Qiaosha Zou[1,5], Jishen Zhao[3], Yu Lu[4], and Yuan Xie[1]

University of California, Santa Barbara[1], Hewlett Packard Labs[2]
University of California, Santa Cruz[3], Qualcomm Inc.[4], Huawei Technologies Inc.[5]
{shuangchenli, yuanxie}ece.ucsb.edu[1]

https://cseweb.ucsd.edu/~jzhao/files/Pinatubo-dac2016.pdf

RowClone Demonstration in Real DRAM Chips



ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao
feig@princeton.edu
Department of Electrical Engineering
Princeton University

Georgios Tziantzioulis
georgios.tziantzioulis@princeton.edu
Department of Electrical Engineering
Princeton University

David Wentzlaff
wentzlaf@princeton.edu
Department of Electrical Engineering
Princeton University

https://parallel.princeton.edu/papers/micro19-gao.pdf

Takeaways

Key Takeaways

- A novel method to accelerate data copy and initialization

- Simple and effective

- Hardware/software cooperative

- Good potential for work building on it to extend it
  - To different granularities
  - To make things more efficient and effective
  - Many works have already built on the paper (see LISA, FIGARO, Ambit, ComputeDRAM, and other works in Google Scholar)

- Easy to read and understand paper

Open Discussion

Discussion Starters

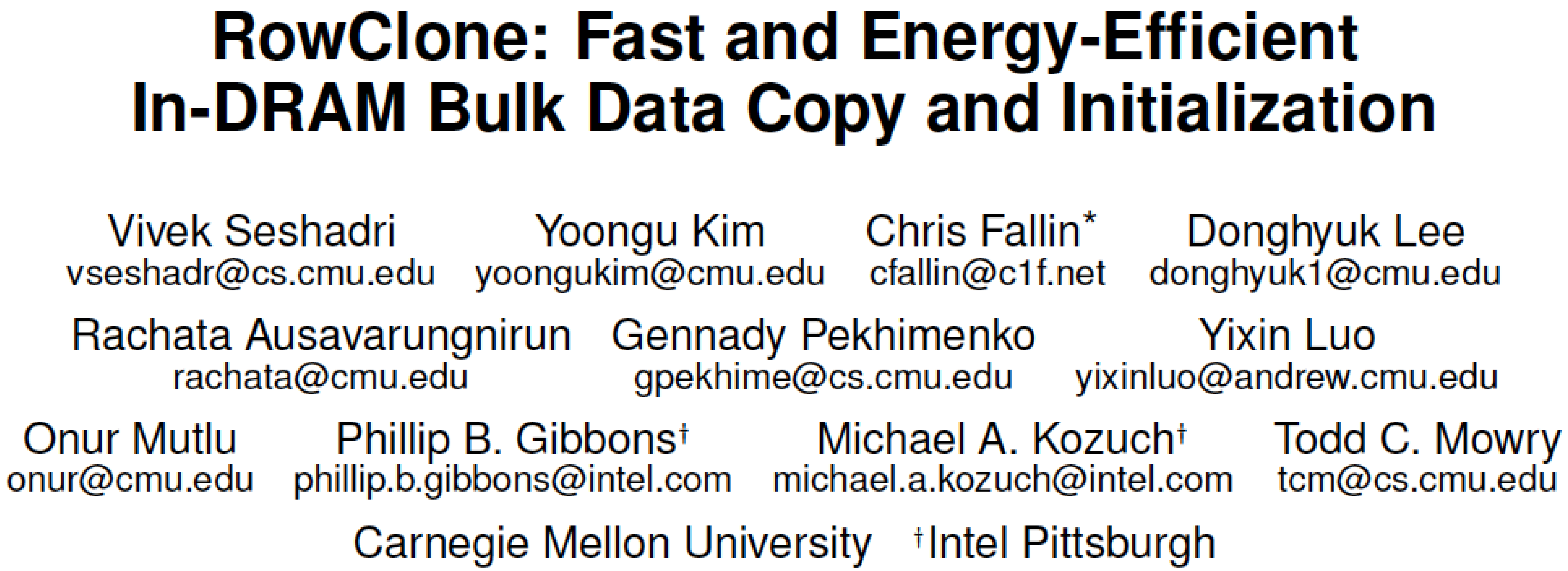
- Thoughts on the previous ideas?
- How practical is this?
- Will the problem become bigger and more important over time?
- Will the solution become more important over time?
- Are other solutions better?
- Is this solution clearly advantageous or opposite in some cases?

General Issues

- Data mapping and interleaving
- Data coherence between caches and DRAM
- Data reuse
- All are issues with Processing in Memory

More on RowClone

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry, **"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"** *Proceedings of the 46th International Symposium on Microarchitecture (**MICRO**)*, Davis, CA, December 2013. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]



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

Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin[*] cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons[†] phillip.b.gibbons@intel.com
Michael A. Kozuch[†] michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu
Carnegie Mellon University [†]Intel Pittsburgh

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

Some History

Historical Perspective

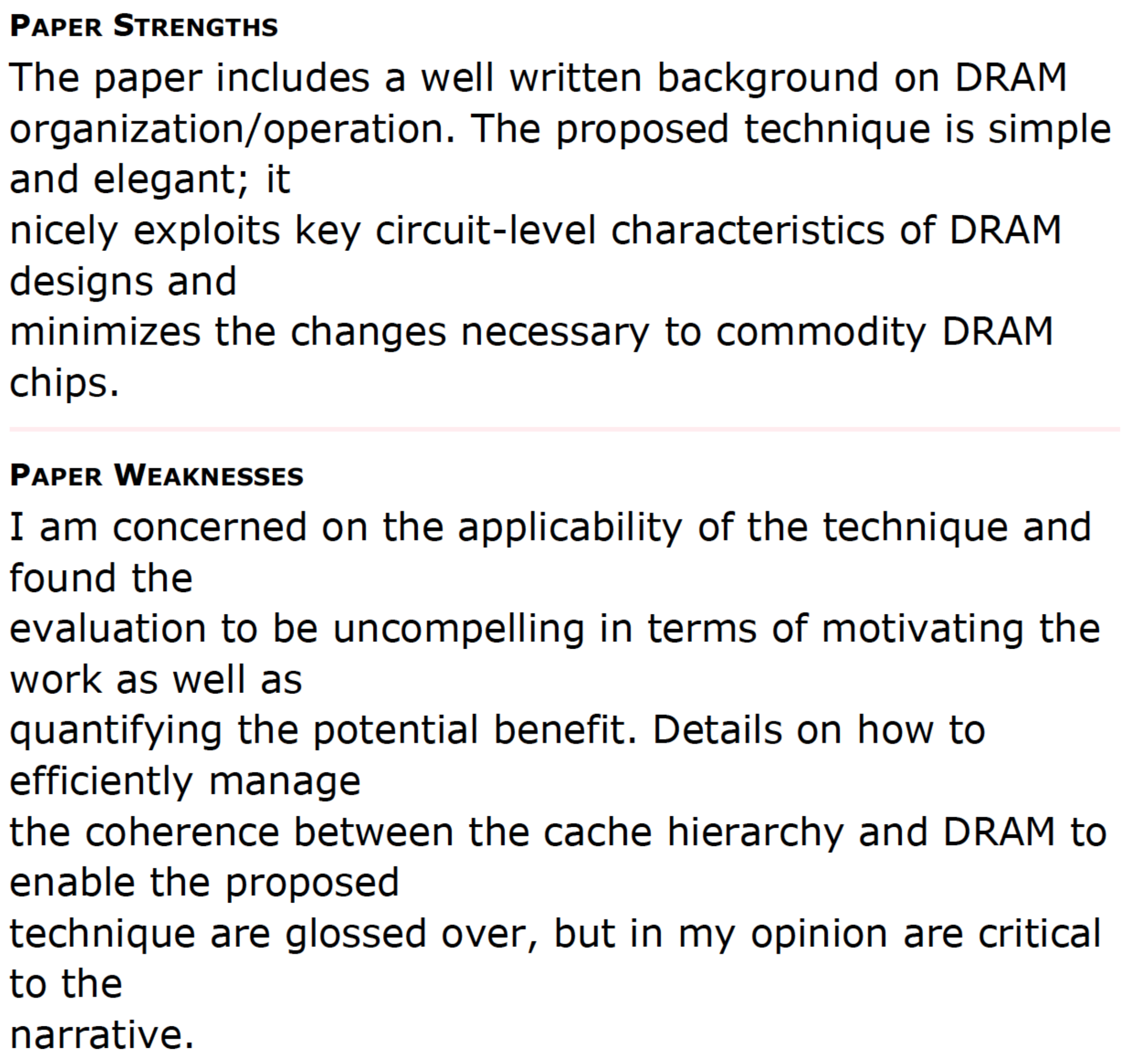
- This work is perhaps the first example of “minimally changing DRAM chips” to perform data movement and computation
  - Surprising that it was done as late as 2013!

- It led to a body of work on in-DRAM (and in-NVM) computation with “hopefully small” changes

- Work building on RowClone still continues

- Initially, it was dismissed by some reviewers
  - Rejected from ISCA 2013 conference

One Review (ISCA 2013 Submission)



PAPER STRENGTHS

The paper includes a well written background on DRAM organization/operation. The proposed technique is simple and elegant; it
nicely exploits key circuit-level characteristics of DRAM designs and
minimizes the changes necessary to commodity DRAM chips.

PAPER WEAKNESSES

I am concerned on the applicability of the technique and found the
evaluation to be uncompelling in terms of motivating the work as well as
quantifying the potential benefit. Details on how to efficiently manage
the coherence between the cache hierarchy and DRAM to enable the proposed
technique are glossed over, but in my opinion are critical to the
narrative.

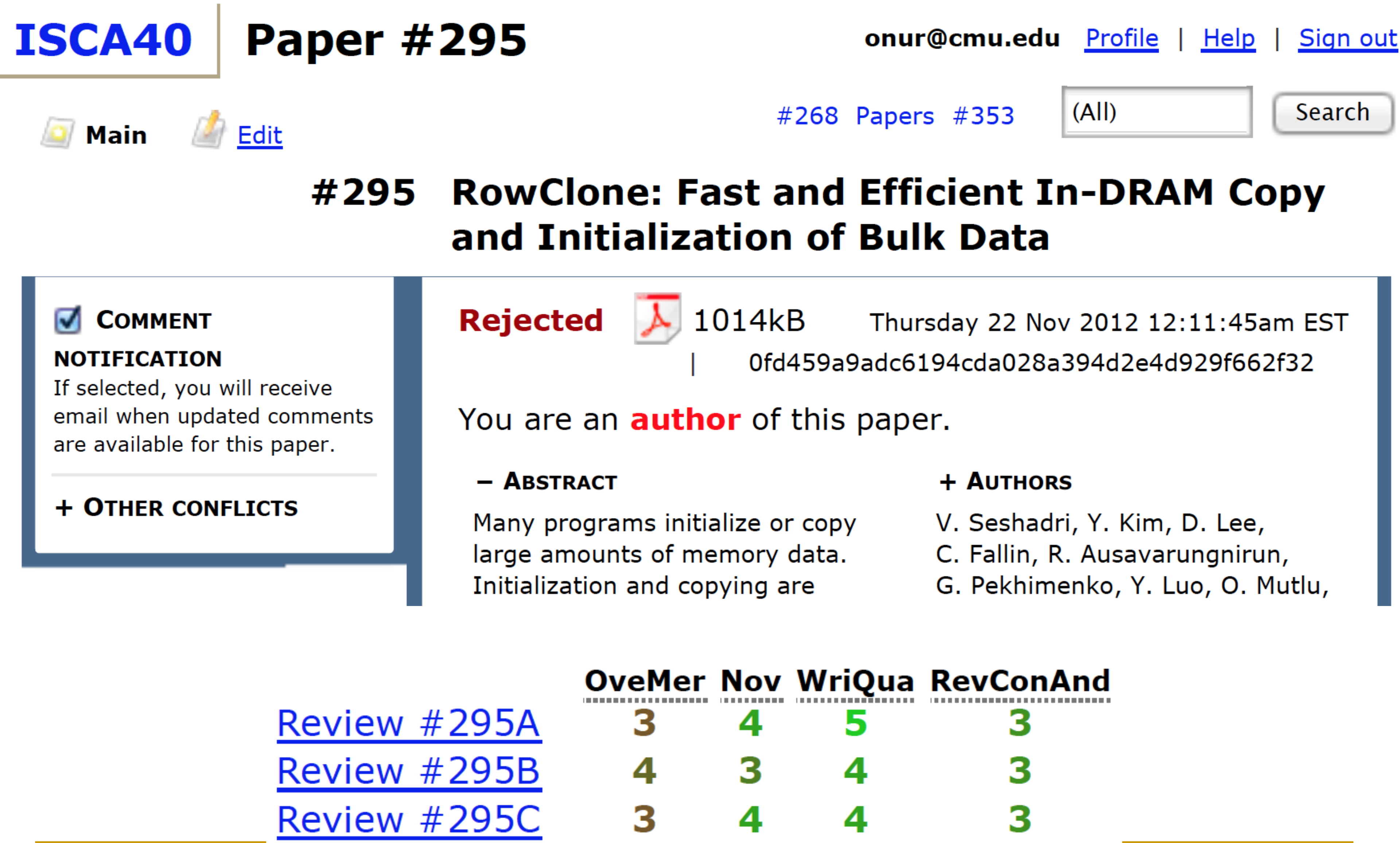
Another Review and Rebuttal

DETAILED COMMENTS

The paper proposes a simple and not new idea, block copy in a DRAM, and the creates a complete

Reviewer B mentions that our idea is "not new". An explicit reference by the reviewer would be helpful here. While the reviewer may be referring to one of the patents that we cite in our paper (citations 2, 6, 25, 26, 27 in the paper), these patents are at a superficial level and do *not* provide a concrete mechanism. In contrast, we propose three concrete mechanisms and provide details on the most important architectural and microarchitectural modifications required at the DRAM chip, the memory controller, and the CPU to enable a system that supports the mechanisms. We also analyze their latency, hardware overhead, power, and performance in detail. We are not aware of any prior work that achieves this.

ISCA 2013 Submission



ISCA40 **Paper #295** onur@cmu.edu Profile | Help | Sign out

Main Edit #268 Papers #353 (All) Search

#295 RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data

☑ **COMMENT NOTIFICATION**
If selected, you will receive email when updated comments are available for this paper.

+ **OTHER CONFLICTS**

Rejected 1014kB Thursday 22 Nov 2012 12:11:45am EST | 0fd459a9adc6194cda028a394d2e4d929f662f32

You are an **author** of this paper.

− **ABSTRACT**
Many programs initialize or copy large amounts of memory data. Initialization and copying are

+ **AUTHORS**
V. Seshadri, Y. Kim, D. Lee, C. Fallin, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu,

	OveMer	Nov	WriQua	RevConAnd
Review #295A	3	4	5	3
Review #295B	4	3	4	3
Review #295C	3	4	4	3

Yet Later… in ISCA 2015…

Profiling a warehouse-scale computer

Svilen Kanev† Harvard University
Juan Pablo Darago† Universidad de Buenos Aires
Kim Hazelwood† Yahoo Labs
Parthasarathy Ranganathan Google
Tipp Moseley Google
Gu-Yeon Wei Harvard University
David Brooks Harvard University

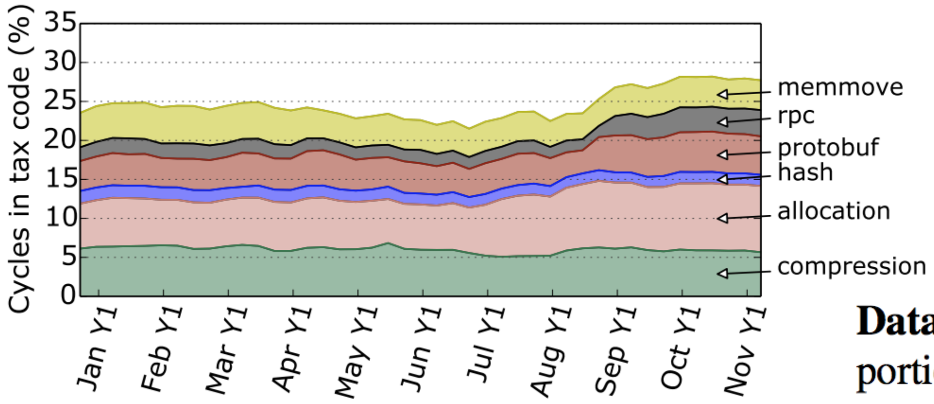


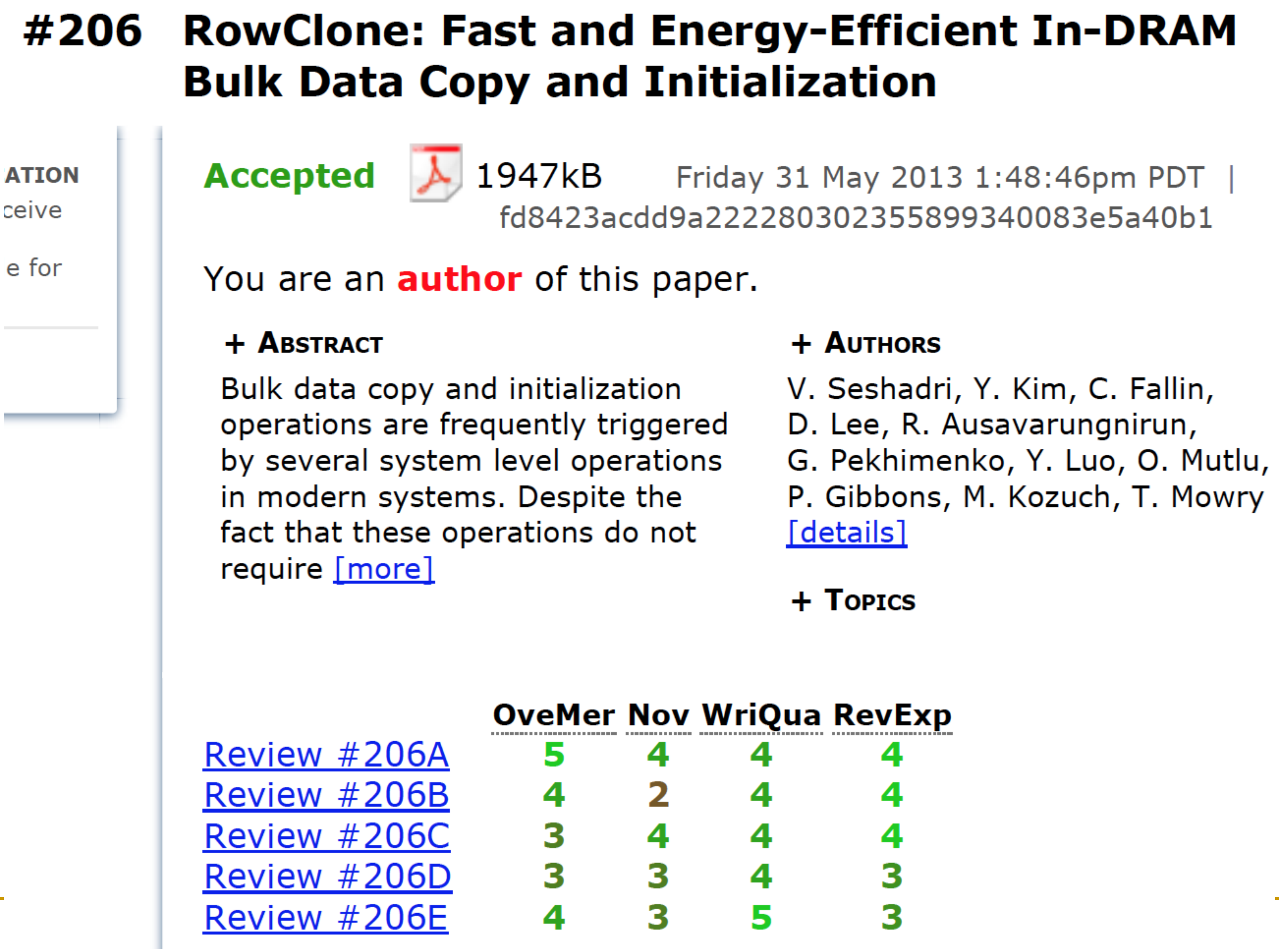
Figure 4: 22-27% of WSC cycles are spent in different co nents of "datacenter tax".

we see common building blocks once we aggregate sampled profile data across many applications running in a datacenter. In this section, we quantify the performance impact of the **datacenter tax**, and argue that its components are prime candidates for hardware acceleration in future datacenter SoCs.

Data movement In fact, RPCs are by far not the only code portions that do data movement. We also tracked all calls to the `memcpy()` and `memmove()` library functions to estimate the amount of time spent on explicit data movement (i.e., exposed through a simple API). This is a conservative estimate because it does not track inlined or explicit copies. Just the variants of these two library functions represent 4-5% of datacenter cycles.

Recent work in performing data movement in DRAM [45] could optimize away this piece of tax.

MICRO 2013 Submission



#206 RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

ATION
ceive
e for

Accepted 1947kB Friday 31 May 2013 1:48:46pm PDT | fd8423acdd9a2222803023558993400 83e5a40b1

You are an **author** of this paper.

+ ABSTRACT

Bulk data copy and initialization operations are frequently triggered by several system level operations in modern systems. Despite the fact that these operations do not require [more]

+ AUTHORS

V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. Gibbons, M. Kozuch, T. Mowry [details]

+ TOPICS

	OveMer	Nov	WriQua	RevExp
Review #206A	5	4	4	4
Review #206B	4	2	4	4
Review #206C	3	4	4	4
Review #206D	3	3	4	3
Review #206E	4	3	5	3

Suggestions to Reviewers

- Be fair; you do not know it all
- Be open-minded; you do not know it all
- Be accepting of diverse research methods: there is no single way of doing research
- Be constructive, not destructive
- Do not have double standards…

Do not block or delay scientific progress for non-reasons

Seminar in Computer Architecture

Meeting 2: Example Review: RowClone

Prof. Onur Mutlu

ETH Zürich

Fall 2020

24 September 2020