

# Computer Architecture

## Lecture 8: Processing near Memory

Prof. Onur Mutlu

ETH Zürich

Fall 2021

22 October 2021

# Sub-Agenda: In-Memory Computation

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - **Processing using Memory**
  - Processing near Memory
- How to Enable Adoption of Processing in Memory
- Conclusion



# Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near 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      Yoongu Kim      Chris Fallin\*      Donghyuk Lee  
vseshadr@cs.cmu.edu    yoongukim@cmu.edu    cfallin@c1f.net    donghyuk1@cmu.edu

Rachata Ausavarungnirun      Gennady Pekhimenko      Yixin Luo  
rachata@cmu.edu      gpekhime@cs.cmu.edu      yixinluo@andrew.cmu.edu

Onur Mutlu      Phillip B. Gibbons†      Michael A. Kozuch†      Todd C. Mowry  
onur@cmu.edu    phillip.b.gibbons@intel.com    michael.a.kozuch@intel.com    tcm@cs.cmu.edu

Carnegie Mellon University    †Intel Pittsburgh

# More on In-DRAM Bulk 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

# More on In-DRAM Bitwise Operations

---

- Vivek Seshadri et al., “**Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology**,” MICRO 2017.

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

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

<sup>1</sup>Microsoft Research India   <sup>2</sup>NVIDIA Research   <sup>3</sup>Intel   <sup>4</sup>ETH Zürich   <sup>5</sup>Carnegie Mellon University

# More on In-DRAM Bulk Bitwise Execution

---

- 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

# RowClone & Bitwise Ops 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



# Pinatubo: RowClone and Bitwise Ops in PCM

---

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

Shuangchen Li<sup>1\*</sup>, Cong Xu<sup>2</sup>, Qiaosha Zou<sup>1,5</sup>, Jishen Zhao<sup>3</sup>, Yu Lu<sup>4</sup>, and Yuan Xie<sup>1</sup>

University of California, Santa Barbara<sup>1</sup>, Hewlett Packard Labs<sup>2</sup>

University of California, Santa Cruz<sup>3</sup>, Qualcomm Inc.<sup>4</sup>, Huawei Technologies Inc.<sup>5</sup>  
{shuangchenli, yuanxie}@ece.ucsb.edu<sup>1</sup>

# Other Examples of “Why Change? It’s Working OK!”



# Mindset Issues Are Everywhere

---

- “Why Change? It’s Working OK!” mindset limits progress
- There are many such examples in real life
- Examples of Bandwidth Waste in Real Life
- Examples of Latency and Queueing Delays in Real Life
- Example of Where to Build a Bridge over a River

# Where to Build a Bridge?




# Another Example

# Initial RowHammer Reviews

## Disturbance Errors in DRAM: Demonstration, Characterization, and Prevention

**Rejected (R2)**



863kB

Friday 31 May 2013 2:00:53pm PDT

b9bf06021da54cddf4cd0b3565558a181868b972

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

+ ABSTRACT

+ AUTHORS

	OveMer	Nov	WriQua	RevExp
<a href="#">Review #66A</a>	1	4	4	4
<a href="#">Review #66B</a>	5	4	5	3
<a href="#">Review #66C</a>	2	3	5	4
<a href="#">Review #66D</a>	1	2	3	4
<a href="#">Review #66E</a>	4	4	4	3
<a href="#">Review #66F</a>	2	4	4	3



# Missing the Point **Reviews from Micro 2013**

## **PAPER WEAKNESSES**

This is an excellent test methodology paper, but there is no micro-architectural or architectural content.

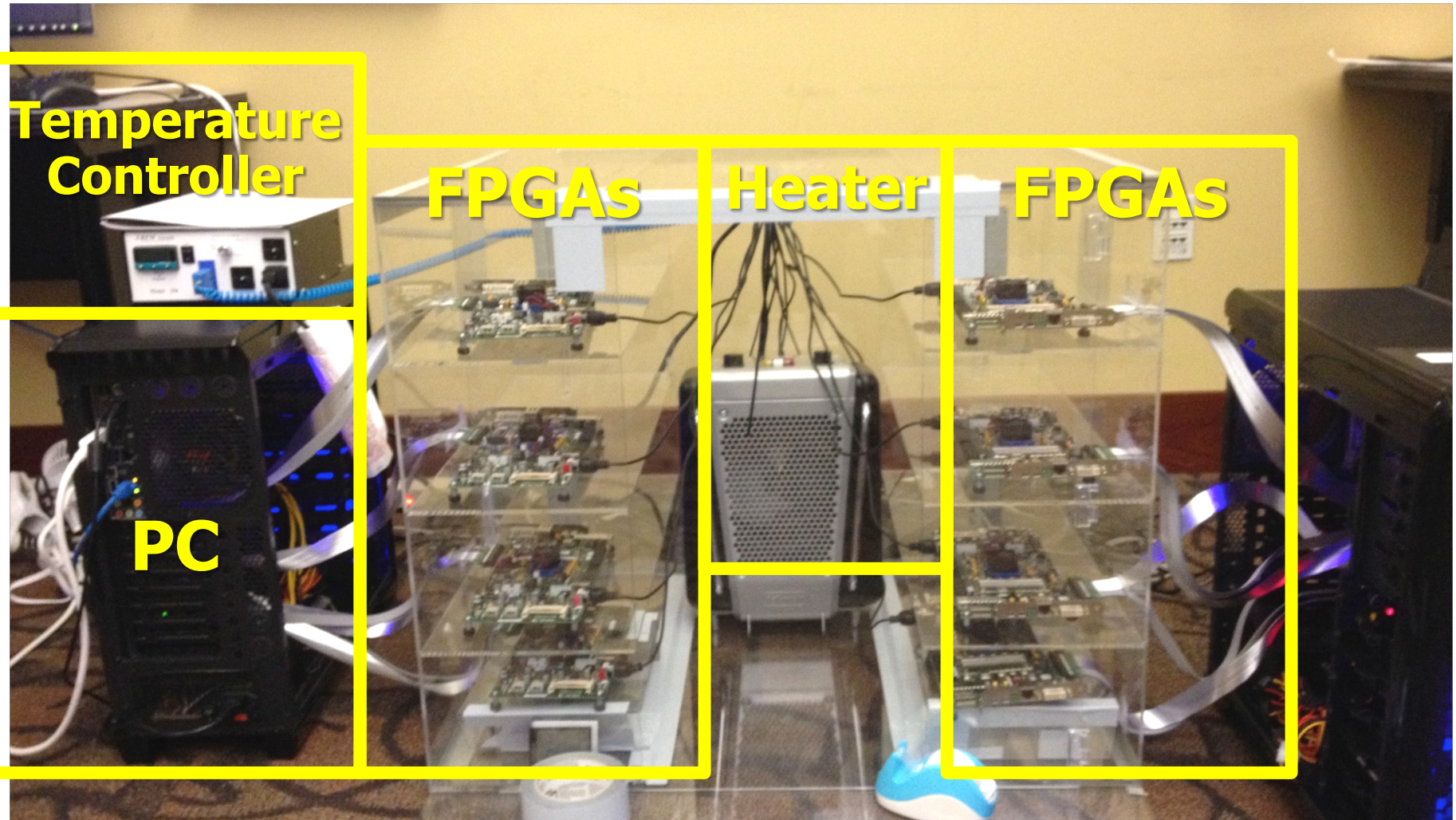
## **PAPER WEAKNESSES**

- Whereas they show disturbance may happen in DRAM array, authors don't show it can be an issue in realistic DRAM usage scenario
- Lacks architectural/microarchitectural impact on the DRAM disturbance analysis

## **PAPER WEAKNESSES**

The mechanism investigated by the authors is one of many well known disturb mechanisms. The paper does not discuss the root causes to sufficient depth and the importance of this mechanism compared to others. Overall the length of the sections restating known information is much too long in relation to new work.

# Experimental DRAM Testing Infrastructure





# Tested DRAM Modules (129 total)

Manufacturer	Module	Date*	Timing†		Organization		Chip			Victims-per-Module			RI <sub>th</sub> (ms)
		(yy-ww)	Freq (MT/s)	t <sub>RC</sub> (ns)	Size (GB)	Chips	Size (Gb)‡	Pins	DieVersion§	Average	Minimum	Maximum	Min
Total of 43 Modules	A <sub>1</sub>	10-08	1066	50.625	0.5	4	1 × 16	B	0	0	0	–	
	A <sub>2</sub>	10-20	1066	50.625	1	8	1 × 8	F	0	0	0	–	
	A <sub>3-5</sub>	10-20	1066	50.625	0.5	4	1 × 16	B	0	0	0	–	
	A <sub>6-7</sub>	11-24	1066	49.125	1	4	2 × 16	D	7.8 × 10 <sup>1</sup>	5.2 × 10 <sup>1</sup>	1.0 × 10 <sup>2</sup>	21.3	
	A <sub>8-12</sub>	11-26	1066	49.125	1	4	2 × 16	D	2.4 × 10 <sup>2</sup>	5.4 × 10 <sup>1</sup>	4.4 × 10 <sup>2</sup>	16.4	
	A <sub>13-14</sub>	11-50	1066	49.125	1	4	2 × 16	D	8.8 × 10 <sup>1</sup>	1.7 × 10 <sup>1</sup>	1.6 × 10 <sup>2</sup>	26.2	
	A <sub>15-16</sub>	12-22	1600	50.625	1	4	2 × 16	D	9.5	9	1.0 × 10 <sup>1</sup>	34.4	
	A <sub>17-18</sub>	12-26	1600	49.125	2	8	2 × 8	M	1.2 × 10 <sup>2</sup>	3.7 × 10 <sup>1</sup>	2.0 × 10 <sup>2</sup>	21.3	
	A <sub>19-30</sub>	12-40	1600	48.125	2	8	2 × 8	K	8.6 × 10 <sup>6</sup>	7.0 × 10 <sup>6</sup>	1.0 × 10 <sup>7</sup>	8.2	
	A <sub>31-34</sub>	13-02	1600	48.125	2	8	2 × 8	–	1.8 × 10 <sup>6</sup>	1.0 × 10 <sup>6</sup>	3.5 × 10 <sup>6</sup>	11.5	
	A <sub>35-36</sub>	13-14	1600	48.125	2	8	2 × 8	–	4.0 × 10 <sup>1</sup>	1.9 × 10 <sup>1</sup>	6.1 × 10 <sup>1</sup>	21.3	
	A <sub>37-38</sub>	13-20	1600	48.125	2	8	2 × 8	K	1.7 × 10 <sup>6</sup>	1.4 × 10 <sup>6</sup>	2.0 × 10 <sup>6</sup>	9.8	
	A <sub>39-40</sub>	13-28	1600	48.125	2	8	2 × 8	K	5.7 × 10 <sup>4</sup>	5.4 × 10 <sup>4</sup>	6.0 × 10 <sup>4</sup>	16.4	
	A <sub>41</sub>	14-04	1600	49.125	2	8	2 × 8	–	2.7 × 10 <sup>5</sup>	2.7 × 10 <sup>5</sup>	2.7 × 10 <sup>5</sup>	18.0	
	A <sub>42-43</sub>	14-04	1600	48.125	2	8	2 × 8	K	0.5	0	1	62.3	
Total of 54 Modules	B <sub>1</sub>	08-49	1066	50.625	1	8	1 × 8	D	0	0	0	–	
	B <sub>2</sub>	09-49	1066	50.625	1	8	1 × 8	E	0	0	0	–	
	B <sub>3</sub>	10-19	1066	50.625	1	8	1 × 8	F	0	0	0	–	
	B <sub>4</sub>	10-31	1333	49.125	2	8	2 × 8	C	0	0	0	–	
	B <sub>5</sub>	11-13	1333	49.125	2	8	2 × 8	C	0	0	0	–	
	B <sub>6</sub>	11-16	1066	50.625	1	8	1 × 8	F	0	0	0	–	
	B <sub>7</sub>	11-19	1066	50.625	1	8	1 × 8	F	0	0	0	–	
	B <sub>8</sub>	11-25	1333	49.125	2	8	2 × 8	C	0	0	0	–	
	B <sub>9</sub>	11-37	1333	49.125	2	8	2 × 8	D	1.9 × 10 <sup>6</sup>	1.9 × 10 <sup>6</sup>	1.9 × 10 <sup>6</sup>	11.5	
	B <sub>10-12</sub>	11-46	1333	49.125	2	8	2 × 8	D	2.2 × 10 <sup>6</sup>	1.5 × 10 <sup>6</sup>	2.7 × 10 <sup>6</sup>	11.5	
	B <sub>13</sub>	11-49	1333	49.125	2	8	2 × 8	C	0	0	0	–	
	B <sub>14</sub>	12-01	1866	47.125	2	8	2 × 8	D	9.1 × 10 <sup>5</sup>	9.1 × 10 <sup>5</sup>	9.1 × 10 <sup>5</sup>	9.8	
	B <sub>15-31</sub>	12-10	1866	47.125	2	8	2 × 8	D	9.8 × 10 <sup>5</sup>	7.8 × 10 <sup>5</sup>	1.2 × 10 <sup>6</sup>	11.5	
	B <sub>32</sub>	12-25	1600	48.125	2	8	2 × 8	E	7.4 × 10 <sup>5</sup>	7.4 × 10 <sup>5</sup>	7.4 × 10 <sup>5</sup>	11.5	
	B <sub>33-42</sub>	12-28	1600	48.125	2	8	2 × 8	E	5.2 × 10 <sup>5</sup>	1.9 × 10 <sup>5</sup>	7.3 × 10 <sup>5</sup>	11.5	
	B <sub>43-47</sub>	12-31	1600	48.125	2	8	2 × 8	E	4.0 × 10 <sup>5</sup>	2.9 × 10 <sup>5</sup>	5.5 × 10 <sup>5</sup>	13.1	
	B <sub>48-51</sub>	13-19	1600	48.125	2	8	2 × 8	E	1.1 × 10 <sup>5</sup>	7.4 × 10 <sup>4</sup>	1.4 × 10 <sup>5</sup>	14.7	
	B <sub>52-53</sub>	13-40	1333	49.125	2	8	2 × 8	D	2.6 × 10 <sup>4</sup>	2.3 × 10 <sup>4</sup>	2.9 × 10 <sup>4</sup>	21.3	
	B <sub>54</sub>	14-07	1333	49.125	2	8	2 × 8	D	7.5 × 10 <sup>3</sup>	7.5 × 10 <sup>3</sup>	7.5 × 10 <sup>3</sup>	26.2	
Total of 32 Modules	C <sub>1</sub>	10-18	1333	49.125	2	8	2 × 8	A	0	0	0	–	
	C <sub>2</sub>	10-20	1066	50.625	2	8	2 × 8	A	0	0	0	–	
	C <sub>3</sub>	10-22	1066	50.625	2	8	2 × 8	A	0	0	0	–	
	C <sub>4-5</sub>	10-26	1333	49.125	2	8	2 × 8	B	8.9 × 10 <sup>2</sup>	6.0 × 10 <sup>2</sup>	1.2 × 10 <sup>3</sup>	29.5	
	C <sub>6</sub>	10-43	1333	49.125	1	8	1 × 8	T	0	0	0	–	
	C <sub>7</sub>	10-51	1333	49.125	2	8	2 × 8	B	4.0 × 10 <sup>2</sup>	4.0 × 10 <sup>2</sup>	4.0 × 10 <sup>2</sup>	29.5	
	C <sub>8</sub>	11-12	1333	46.25	2	8	2 × 8	B	6.9 × 10 <sup>2</sup>	6.9 × 10 <sup>2</sup>	6.9 × 10 <sup>2</sup>	21.3	
	C <sub>9</sub>	11-19	1333	46.25	2	8	2 × 8	B	9.2 × 10 <sup>2</sup>	9.2 × 10 <sup>2</sup>	9.2 × 10 <sup>2</sup>	27.9	
	C <sub>10</sub>	11-31	1333	49.125	2	8	2 × 8	B	3	3	3	39.3	
	C <sub>11</sub>	11-42	1333	49.125	2	8	2 × 8	B	1.6 × 10 <sup>2</sup>	1.6 × 10 <sup>2</sup>	1.6 × 10 <sup>2</sup>	39.3	
	C <sub>12</sub>	11-48	1600	48.125	2	8	2 × 8	C	7.1 × 10 <sup>4</sup>	7.1 × 10 <sup>4</sup>	7.1 × 10 <sup>4</sup>	19.7	
	C <sub>13</sub>	12-08	1333	49.125	2	8	2 × 8	C	3.9 × 10 <sup>4</sup>	3.9 × 10 <sup>4</sup>	3.9 × 10 <sup>4</sup>	21.3	
	C <sub>14-15</sub>	12-12	1333	49.125	2	8	2 × 8	C	3.7 × 10 <sup>4</sup>	2.1 × 10 <sup>4</sup>	5.4 × 10 <sup>4</sup>	21.3	
	C <sub>16-18</sub>	12-20	1600	48.125	2	8	2 × 8	C	3.5 × 10 <sup>3</sup>	1.2 × 10 <sup>3</sup>	7.0 × 10 <sup>3</sup>	27.9	
	C <sub>19</sub>	12-23	1600	48.125	2	8	2 × 8	E	1.4 × 10 <sup>5</sup>	1.4 × 10 <sup>5</sup>	1.4 × 10 <sup>5</sup>	18.0	
	C <sub>20</sub>	12-24	1600	48.125	2	8	2 × 8	C	6.5 × 10 <sup>4</sup>	6.5 × 10 <sup>4</sup>	6.5 × 10 <sup>4</sup>	21.3	
	C <sub>21</sub>	12-26	1600	48.125	2	8	2 × 8	C	2.3 × 10 <sup>4</sup>	2.3 × 10 <sup>4</sup>	2.3 × 10 <sup>4</sup>	24.6	
	C <sub>22</sub>	12-32	1600	48.125	2	8	2 × 8	C	1.7 × 10 <sup>4</sup>	1.7 × 10 <sup>4</sup>	1.7 × 10 <sup>4</sup>	22.9	
	C <sub>23-24</sub>	12-37	1600	48.125	2	8	2 × 8	C	2.3 × 10 <sup>4</sup>	1.1 × 10 <sup>4</sup>	3.4 × 10 <sup>4</sup>	18.0	
	C <sub>25-30</sub>	12-41	1600	48.125	2	8	2 × 8	C	2.0 × 10 <sup>4</sup>	1.1 × 10 <sup>4</sup>	3.2 × 10 <sup>4</sup>	19.7	
	C <sub>31</sub>	13-11	1600	48.125	2	8	2 × 8	C	3.3 × 10 <sup>5</sup>	3.3 × 10 <sup>5</sup>	3.3 × 10 <sup>5</sup>	14.7	
	C <sub>32</sub>	13-35	1600	48.125	2	8	2 × 8	C	3.7 × 10 <sup>4</sup>	3.7 × 10 <sup>4</sup>	3.7 × 10 <sup>4</sup>	21.3	

\* We report the manufacture date marked on the chip packages, which is more accurate than other dates that can be gleaned from a module.

† We report timing constraints stored in the module's on-board ROM [33], which is read by the system BIOS to calibrate the memory controller.

‡ The maximum DRAM chip size supported by our testing platform is 2Gb.

§ We report DRAM die versions marked on the chip packages, which typically progress in the following manner:  $\mathcal{M} \rightarrow \mathcal{A} \rightarrow \mathcal{B} \rightarrow \mathcal{C} \rightarrow \dots$ .

Table 3. Sample population of 129 DDR3 DRAM modules, categorized by manufacturer and sorted by manufacture date

# Fast Forward 6 Months



# More Reviews... **Reviews from ISCA 2014**

---

## **PAPER WEAKNESSES**

1) The disturbance error (a.k.a coupling or cross-talk noise induced error) is a known problem to the DRAM circuit community.

2) What you demonstrated in this paper is so called DRAM row hammering issue - you can even find a Youtube video showing this! - <http://www.youtube.com/watch?v=i3-gQSnBcdo>

2) The architectural contribution of this study is too insignificant.

## **PAPER WEAKNESSES**

- Row Hammering appears to be well-known, and solutions have already been proposed by industry to address the issue.

- The paper only provides a qualitative analysis of solutions to the problem. A more robust evaluation is really needed to know whether the proposed solution is necessary.

# Final RowHammer Reviews

---

## Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Accepted



639kB

21 Nov 2013 10:53:11pm CST |

f039be2735313b39304ae1c6296523867a485610

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

	OveMer	Nov	WriQua	RevConAnd
<a href="#">Review #41A</a>	8	4	5	3
<a href="#">Review #41B</a>	7	4	4	3
<a href="#">Review #41C</a>	6	4	4	3
<a href="#">Review #41D</a>	2	2	5	4
<a href="#">Review #41E</a>	3	2	3	3
<a href="#">Review #41F</a>	7	4	4	3

# RowHammer: Hindsight & Impact (I)

---

## Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

*Abstract. Memory isolation is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology*

# Project Zero

Flipping Bits in Memory Without Accessing Them:  
An Experimental Study of DRAM Disturbance Errors  
(Kim et al., ISCA 2014)

News and updates from the Project Zero team at Google

Exploiting the DRAM rowhammer bug to  
gain kernel privileges (Seaborn, 2015)

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

# RowHammer: Hindsight & Impact (II)

---

- Onur Mutlu and Jeremie Kim,  
**"RowHammer: A Retrospective"**  
*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.  
[[Preliminary arXiv version](#)]

## RowHammer: A Retrospective

Onur Mutlu<sup>§‡</sup>      Jeremie S. Kim<sup>‡§</sup>  
§ETH Zürich      ‡Carnegie Mellon University

# RowHammer in 2020



# RowHammer in 2020 (I)

---

- Jeremie S. Kim, Minesh Patel, A. Giray Yaglikci, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu,  
**"Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"**  
*Proceedings of the 47th International Symposium on Computer Architecture (ISCA)*, Valencia, Spain, June 2020.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]  
[[Talk Video](#) (20 minutes)]  
[[Lightning Talk Video](#) (3 minutes)]

## Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques

Jeremie S. Kim<sup>§†</sup>      Minesh Patel<sup>§</sup>      A. Giray Yağlıkçı<sup>§</sup>  
Hasan Hassan<sup>§</sup>      Roknoddin Azizi<sup>§</sup>      Lois Orosa<sup>§</sup>      Onur Mutlu<sup>§†</sup>  
<sup>§</sup>*ETH Zürich*      <sup>†</sup>*Carnegie Mellon University*

# RowHammer in 2020 (II)

---

- Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi, **"TRRespass: Exploiting the Many Sides of Target Row Refresh"** *Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, USA, May 2020.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Talk Video](#) (17 minutes)]  
[[Source Code](#)]  
[[Web Article](#)]  
***Best paper award.***

## TRRespass: Exploiting the Many Sides of Target Row Refresh

Pietro Frigo<sup>\*†</sup> Emanuele Vannacci<sup>\*†</sup> Hasan Hassan<sup>§</sup> Victor van der Veen<sup>¶</sup>  
Onur Mutlu<sup>§</sup> Cristiano Giuffrida<sup>\*</sup> Herbert Bos<sup>\*</sup> Kaveh Razavi<sup>\*</sup>

# RowHammer in 2020 (III)

---

- Lucian Cojocar, Jeremie Kim, Minesh Patel, Lillian Tsai, Stefan Saroiu, Alec Wolman, and Onur Mutlu,  
**"Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers"**  
*Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, USA, May 2020.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Talk Video](#) (17 minutes)]

## Are We Susceptible to Rowhammer?

## An End-to-End Methodology for Cloud Providers

Lucian Cojocar, Jeremie Kim<sup>§†</sup>, Minesh Patel<sup>§</sup>, Lillian Tsai<sup>‡</sup>,  
Stefan Saroiu, Alec Wolman, and Onur Mutlu<sup>§†</sup>  
Microsoft Research, <sup>§</sup>ETH Zürich, <sup>†</sup>CMU, <sup>‡</sup>MIT



# RowHammer in 2020 (IV)

MICRO 2020

Submit Work ▾

Program ▾

Attend

## Session 1A: Security & Privacy I

5:00 PM CEST – 5:15 PM CEST

### **Graphene: Strong yet Lightweight Row Hammer Protection**

Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, Jae W. Lee (Seoul National University)

5:15 PM CEST – 5:30 PM CEST

### **Persist Level Parallelism: Streamlining Integrity Tree Updates for Secure Persistent Memory**

Alexander Freij, Shougang Yuan, Huiyang Zhou (NC State University); Yan Solihin (University of Central Florida)

5:30 PM CEST – 5:45 PM CEST

### **PThammer: Cross-User-Kernel-Boundary Rowhammer through Implicit Accesses**

Zhi Zhang (University of New South Wales and Data61, CSIRO, Australia); Yueqiang Cheng (Baidu Security); Dongxi Liu, Surya Nepal (Data61, CSIRO, Australia); Zhi Wang (Florida State University); Yuval Yarom (University of Adelaide and Data61, CSIRO, Australia)

# RowHammer in 2020 (V)

S & P

Home

Program ▼

Call For... ▼

Attend ▼

Workshops ▼

## Session #5: Rowhammer

Room 2

Session chair: Michael Franz (UC Irvine)

### **RAMBleed: Reading Bits in Memory Without Accessing Them**

Andrew Kwong (University of Michigan), Daniel Genkin (University of Michigan), Daniel Gruss (Data61)

### **Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers**

Lucian Cojocar (Microsoft Research), Jeremie Kim (ETH Zurich, CMU), Minesh Patel (ETH Zurich, Microsoft Research), Onur Mutlu (ETH Zurich, CMU)

### **Leveraging EM Side-Channel Information to Detect Rowhammer Attacks**

Zhenkai Zhang (Texas Tech University), Zihao Zhan (Vanderbilt University), Daniel Balasubramanian (Vanderbilt University), Peter Volgyesi (Vanderbilt University), Xenofon Koutsoukos (Vanderbilt University)

### **TRRespass: Exploiting the Many Sides of Target Row Refresh**

Pietro Frigo (Vrije Universiteit Amsterdam, The Netherlands), Emanuele Vannacci (Vrije Universiteit Amsterdam, The Netherlands), Onur Mutlu (ETH Zürich), Cristiano Giuffrida (Vrije Universiteit Amsterdam, The Netherlands), Kaveh Razavi (Vrije Universiteit Amsterdam, The Netherlands)

# RowHammer in 2020 (VI)

---

29<sup>TH</sup> USENIX  
SECURITY SYMPOSIUM

ATTEND

PROGRAM

PARTICIPATE

SPONSORS

ABOUT

DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips

Fan Yao, *University of Central Florida*; Adnan Siraj Rakin and Deliang Fan, *Arizona State University*

AVAILABLE MEDIA   

Show details ▶

More to Come...

Suggestion to Researchers: Principle: Passion

---

**Follow Your Passion**  
**(Do not get derailed  
by naysayers)**

---

Suggestion to Researchers: Principle: Resilience

---

**Be Resilient**

---

# Principle: Learning and Scholarship

---

Focus on  
learning and scholarship

# Principle: Learning and Scholarship

---

The quality of your work  
defines your impact



# An Interview on Research and Education

---

- Computing Research and Education (@ ISCA 2019)
  - [https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi\\_4oP9LdL3cc8G6NIjD2Ydz](https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz)
- Maurice Wilkes Award Speech (10 minutes)
  - [https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D\\_5MGV6EnXEJHnV2YFBJI&index=15](https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=15)

# More Thoughts and Suggestions

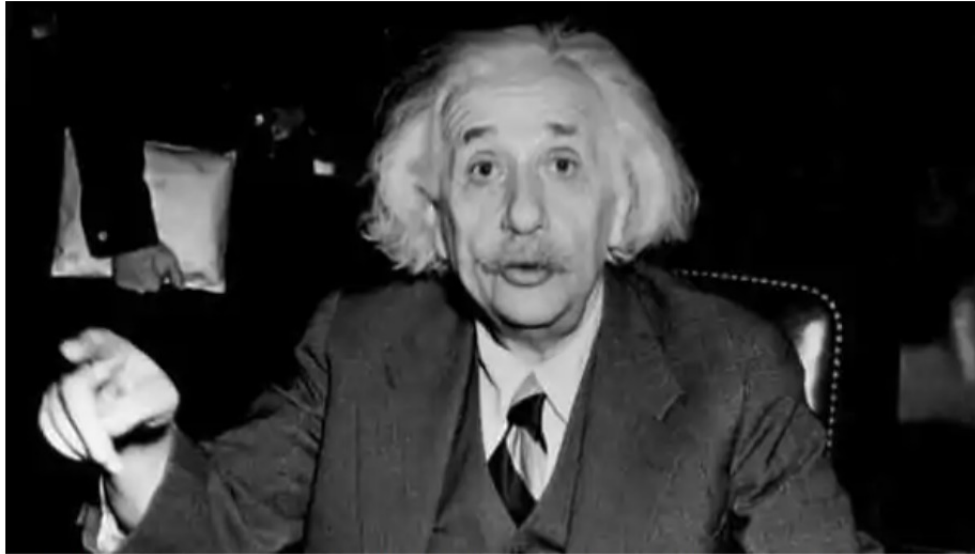
---

- Onur Mutlu,  
**"Some Reflections (on DRAM)"**  
*Award Speech for ACM SIGARCH Maurice Wilkes Award, at the **ISCA** Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.*  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Video of Award Acceptance Speech \(Youtube; 10 minutes\)](#)] [[Youku; 13 minutes](#)]  
[[Video of Interview after Award Acceptance \(Youtube; 1 hour 6 minutes\)](#)] [[Youku; 1 hour 6 minutes](#)]  
[[News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"](#)]
  
- Onur Mutlu,  
**"How to Build an Impactful Research Group"**  
*57th Design Automation Conference Early Career Workshop (**DAC**), Virtual, 19 July 2020.*  
[[Slides \(pptx\)](#)] [[pdf](#)]

# A Fun Reading: Food for Thought

---

- <https://www.livemint.com/science/news/could-einstein-get-published-today-11601014633853.html>



A similar process of professionalization has transformed other parts of the scientific landscape. (Central Press/Getty Images)

THE WALL STREET JOURNAL.

## Could Einstein get published today?

3 min read . Updated: 25 Sep 2020, 11:51 AM IST

The Wall Street Journal

Scientific journals and institutions have become more professionalized over the last century, leaving less room for individual style

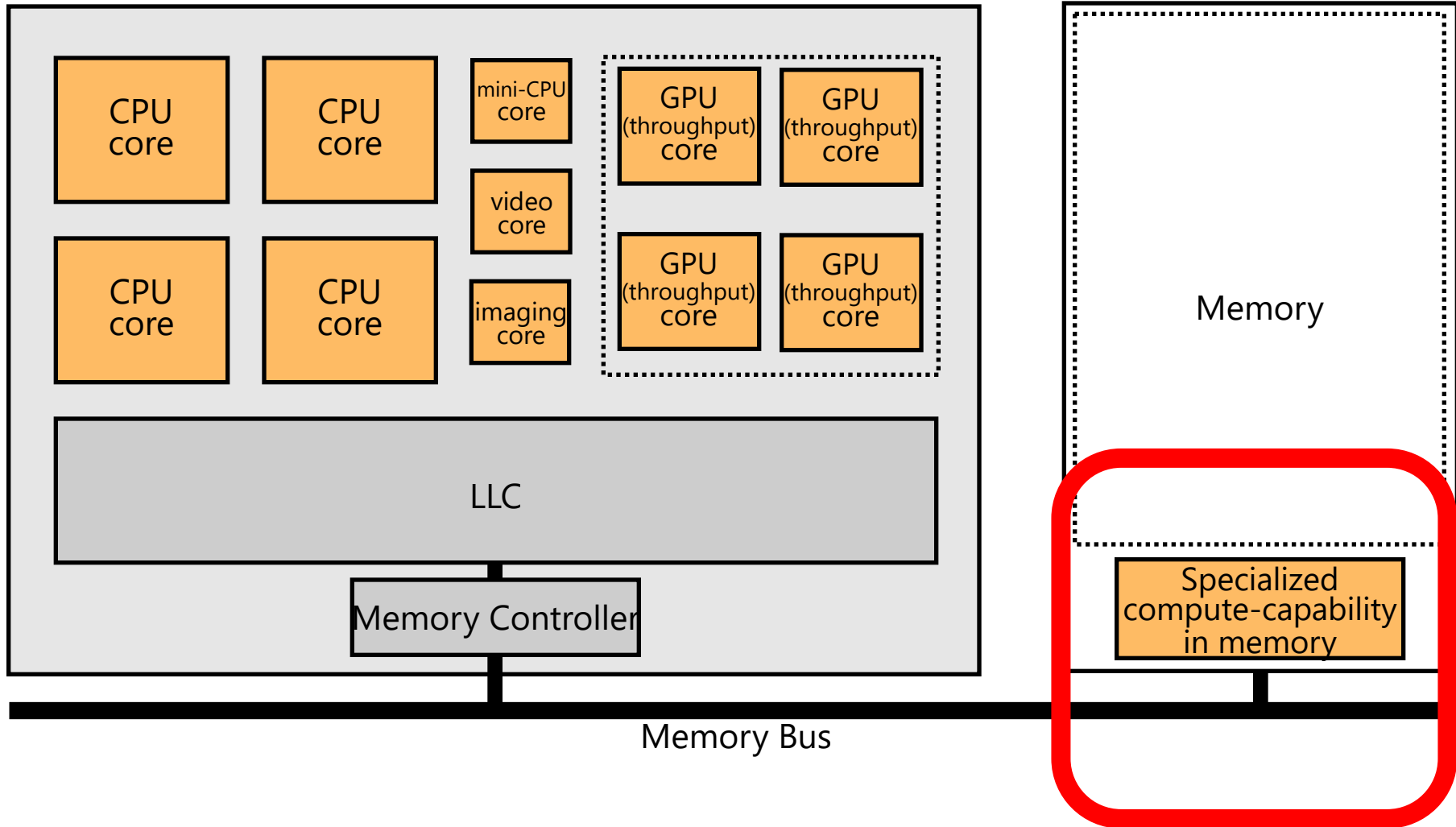
# Sub-Agenda: In-Memory Computation

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Processing using Memory
  - **Processing near Memory**
- How to Enable Adoption of Processing in Memory
- Conclusion

We Need to Think Differently  
from the Past Approaches

# Memory as an Accelerator



**Memory similar to a "conventional" accelerator**

# Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory

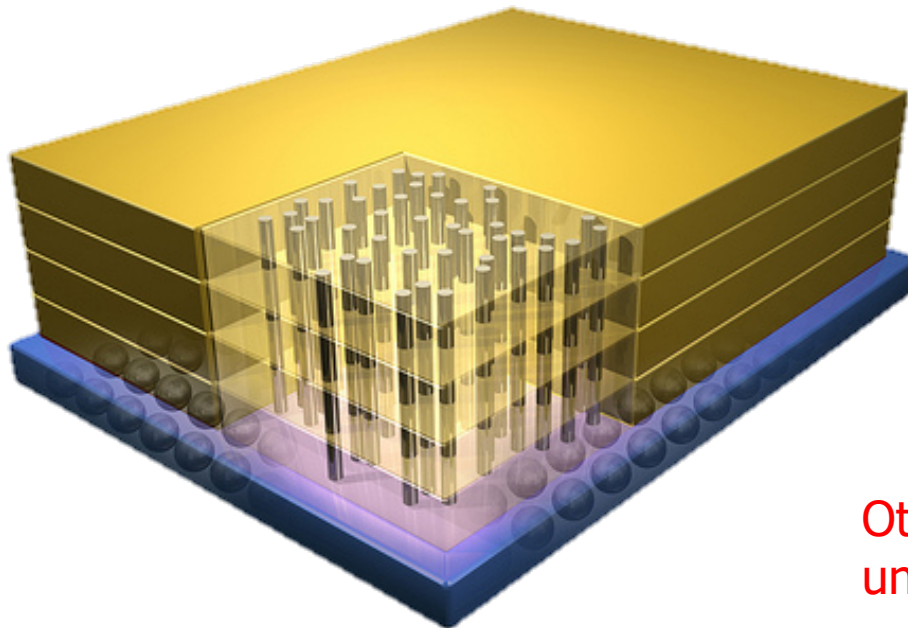


# Opportunity: 3D-Stacked Logic+Memory

---



Hybrid Memory Cube  
C O N S O R T I U M



Memory

Logic

Other "True 3D" technologies  
under development

# DRAM Landscape (circa 2015)

Segment	DRAM Standards & Architectures
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDram3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Kim+, “[Ramulator: A Flexible and Extensible DRAM Simulator](#)”, IEEE CAL 2015.

# Several Questions in 3D-Stacked PIM

---

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading
- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming

# Another Example: In-Memory Graph Processing

- Large graphs are everywhere (circa 2015)



36 Million  
Wikipedia Pages



1.4 Billion  
Facebook Users

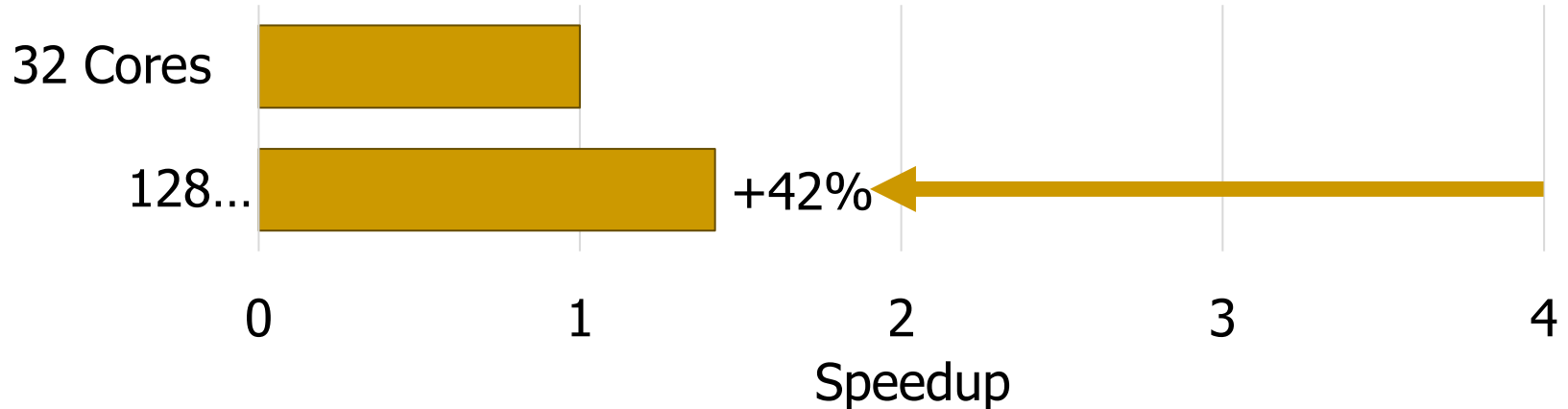


300 Million  
Twitter Users



30 Billion  
Instagram Photos

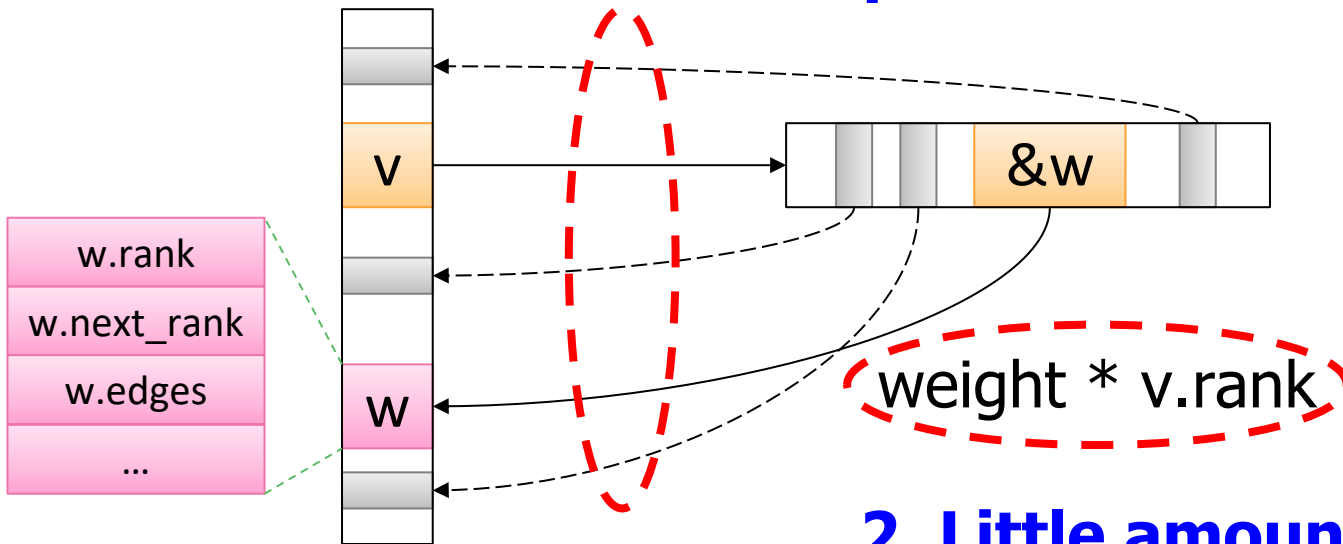
- Scalable large-scale graph processing is challenging



# Key Bottlenecks in Graph Processing

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```

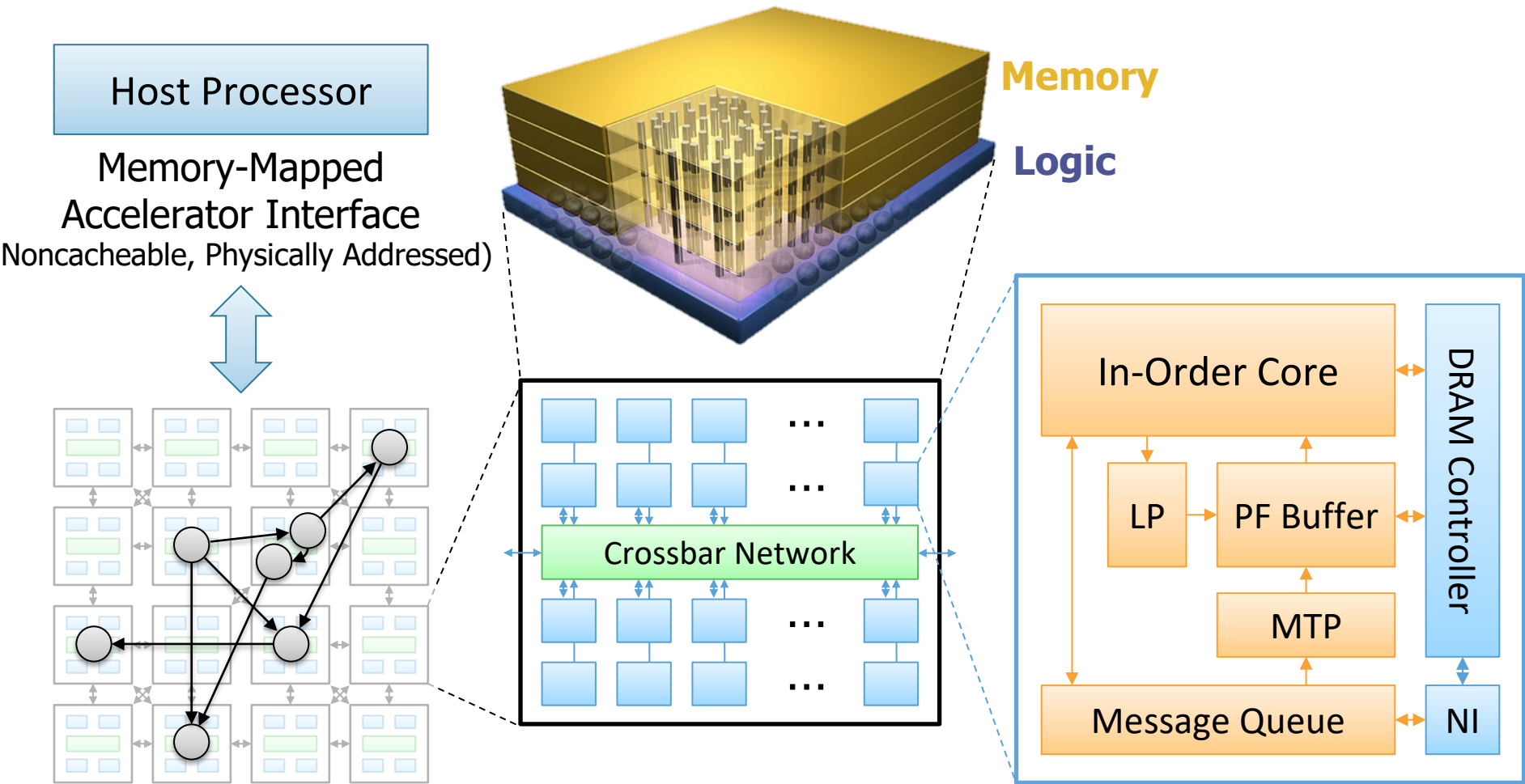
**1. Frequent random memory accesses**



**2. Little amount of computation**

# Tesseract System for Graph Processing

Interconnected set of 3D-stacked memory+logic chips with simple cores

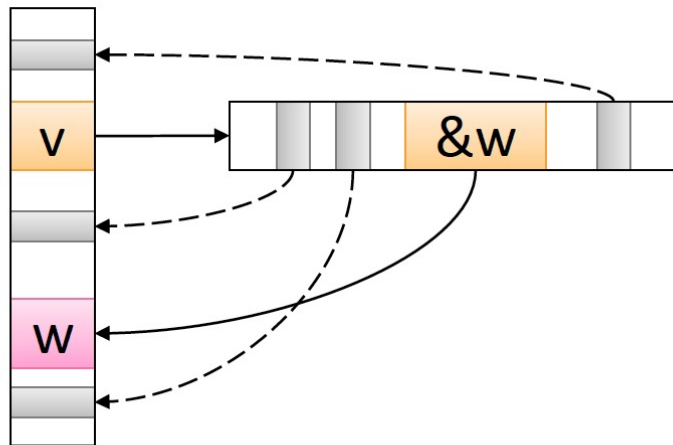






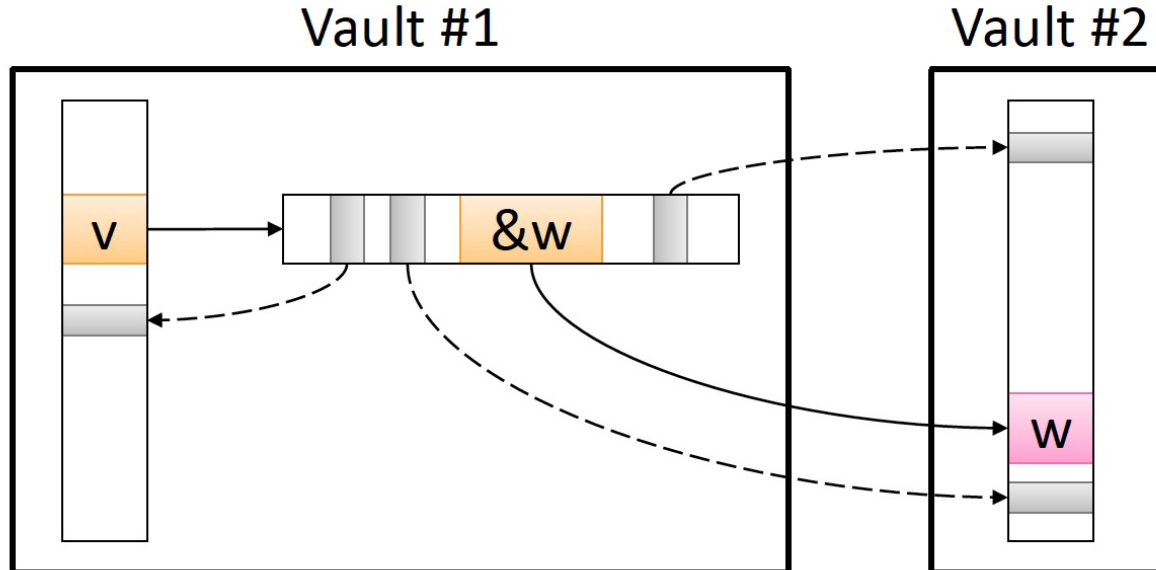
# Communications In Tesseract (I)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```



# Communications In Tesseract (II)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```

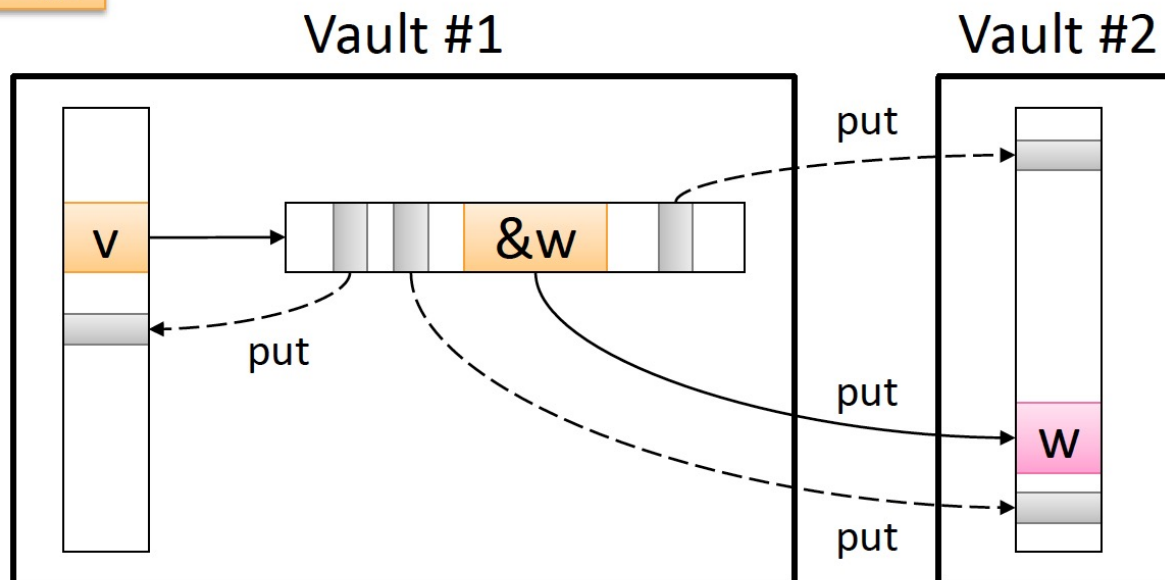


# Communications In Tesseract (III)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    put(w.id, function() { w.next_rank += weight * v.rank; });  
  }  
}  
barrier();
```

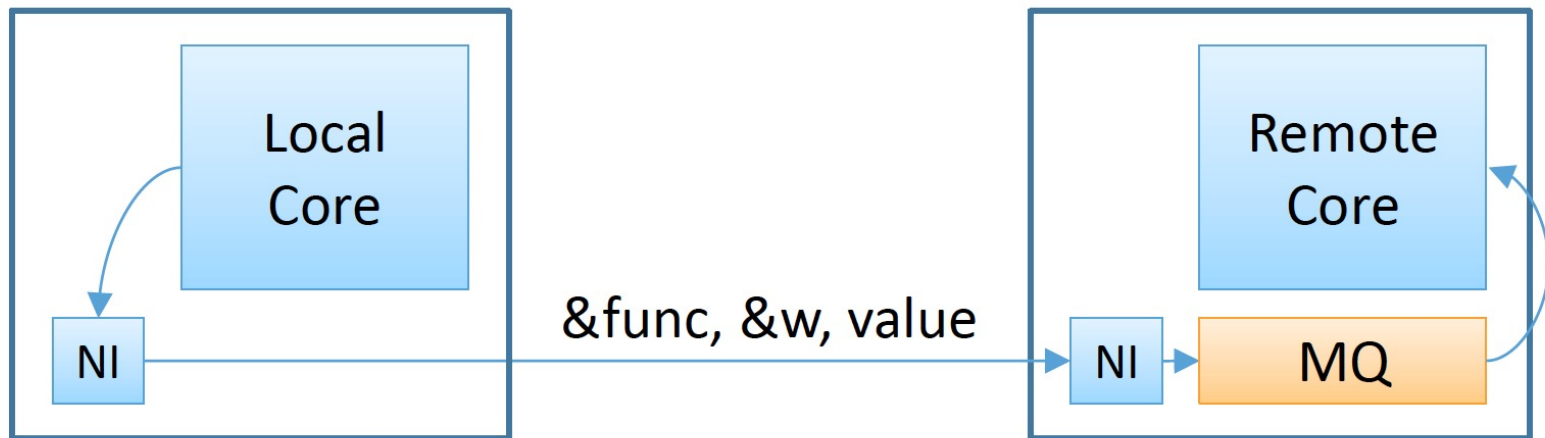
**Non-blocking Remote Function Call**

Can be **delayed**  
until the nearest barrier



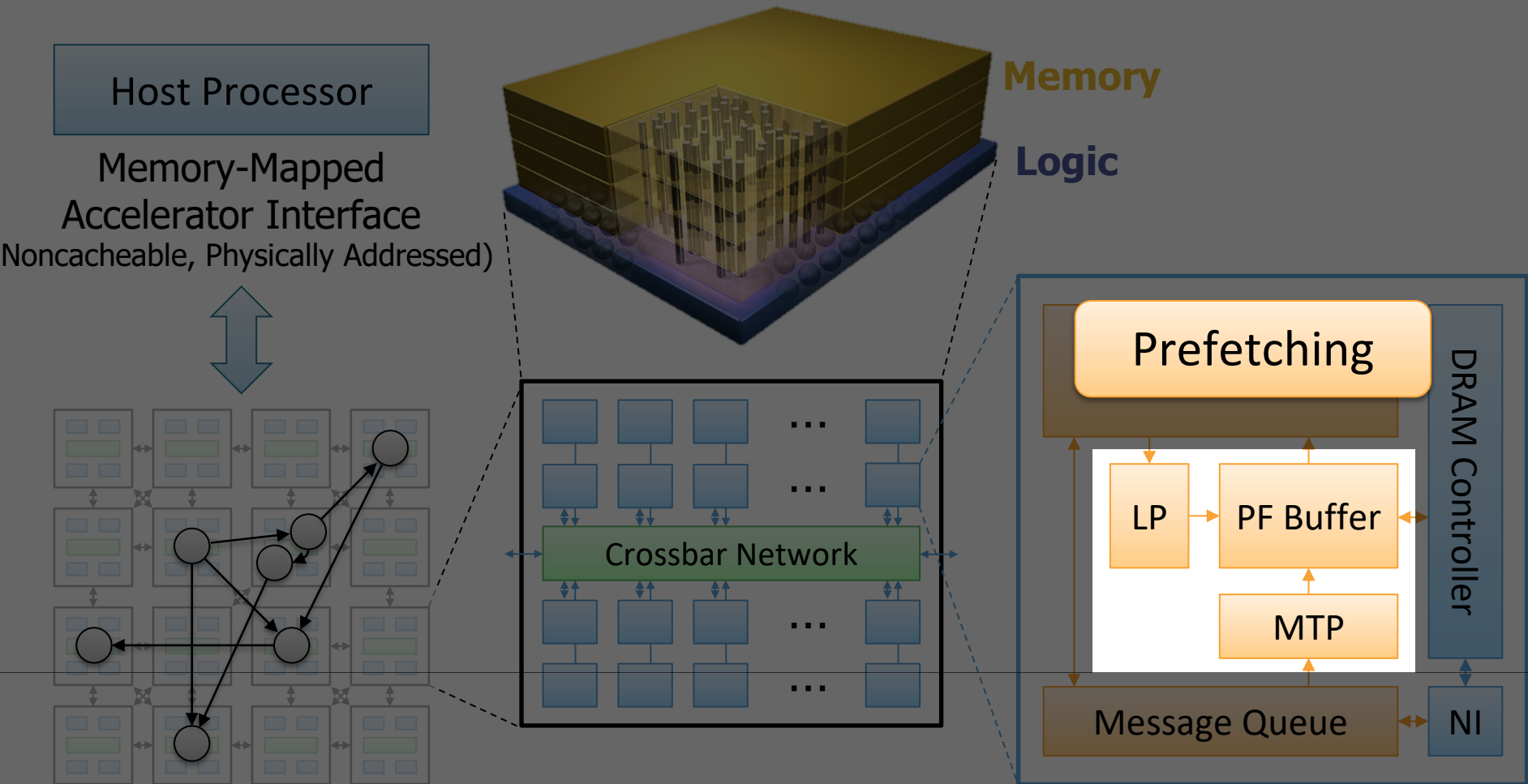
# Remote Function Call (Non-Blocking)

1. Send function address & args to the remote core
2. Store the incoming message to the message queue
3. Flush the message queue when it is full or a synchronization barrier is reached



```
put(w.id, function() { w.next_rank += value; })
```

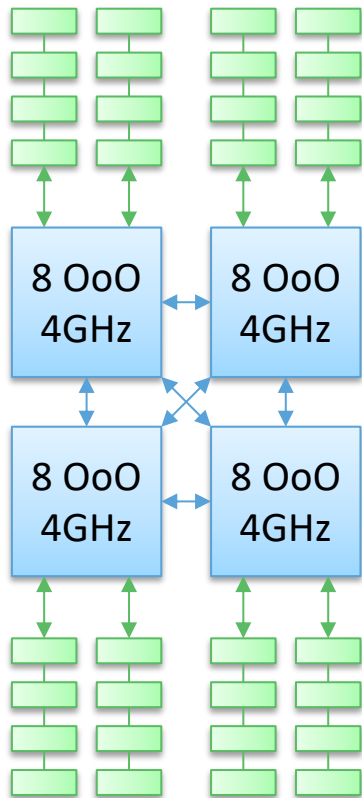
# Tesseract System for Graph Processing





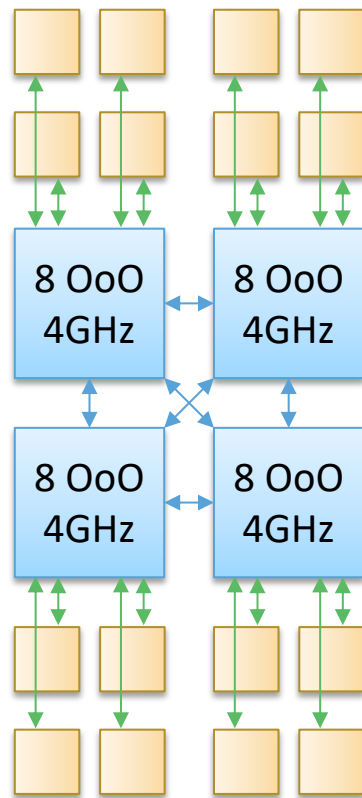
# Evaluated Systems

DDR3-OoO



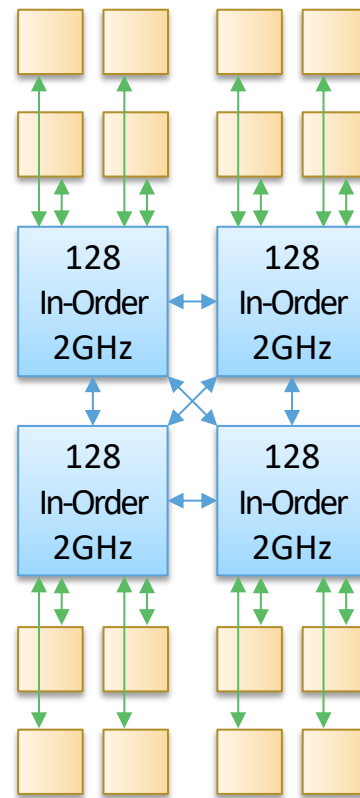
102.4GB/s

HMC-OoO



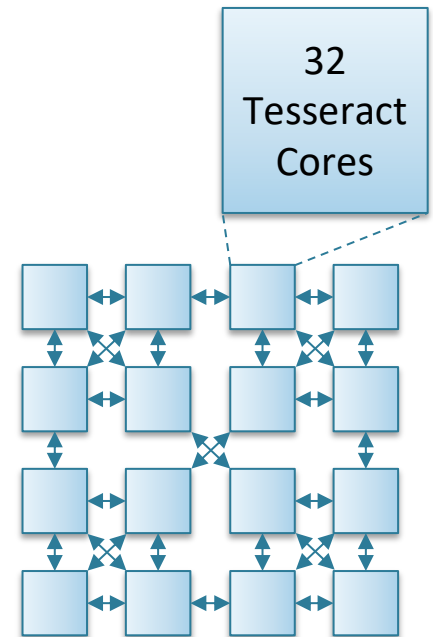
640GB/s

HMC-MC



640GB/s

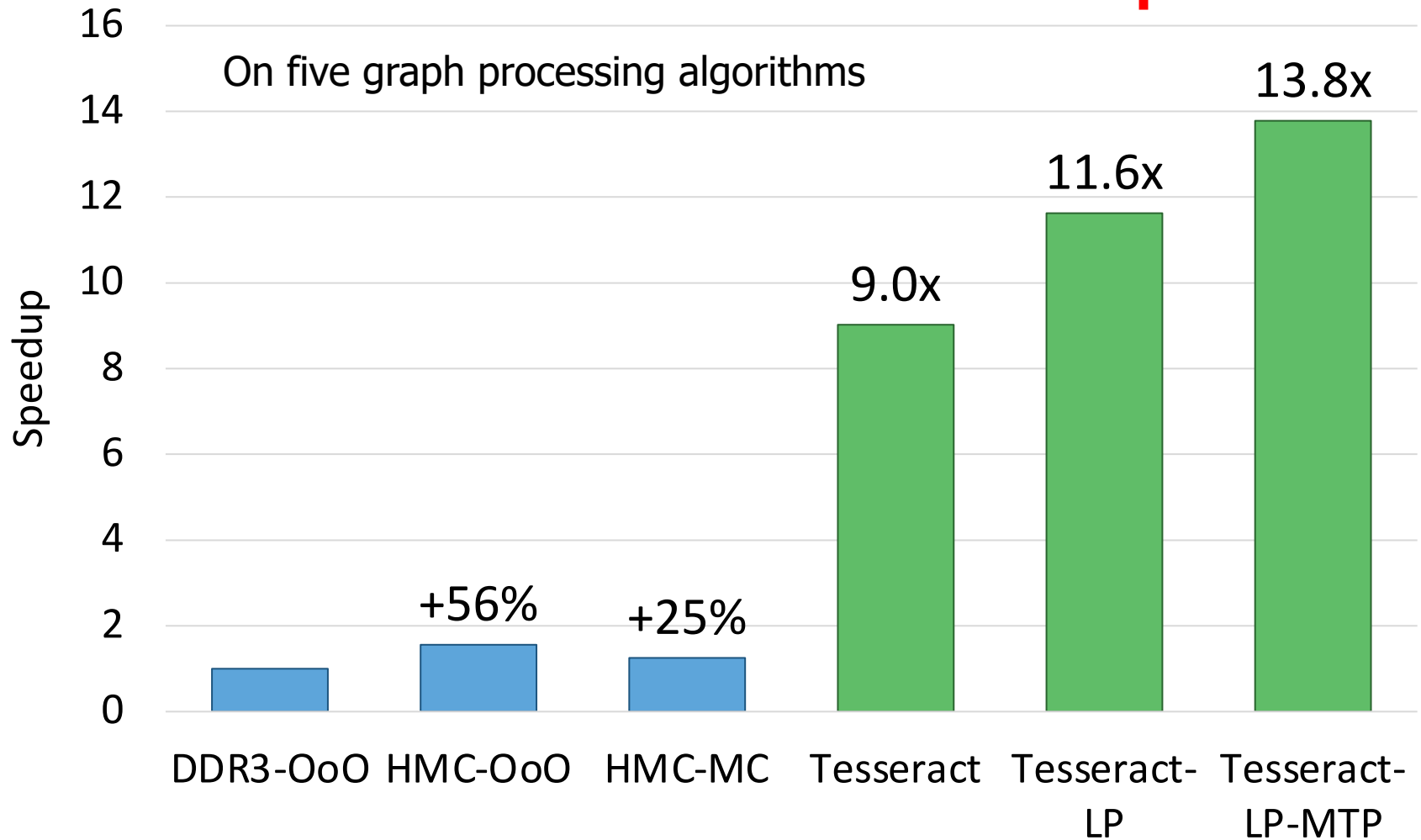
**Tesseract**



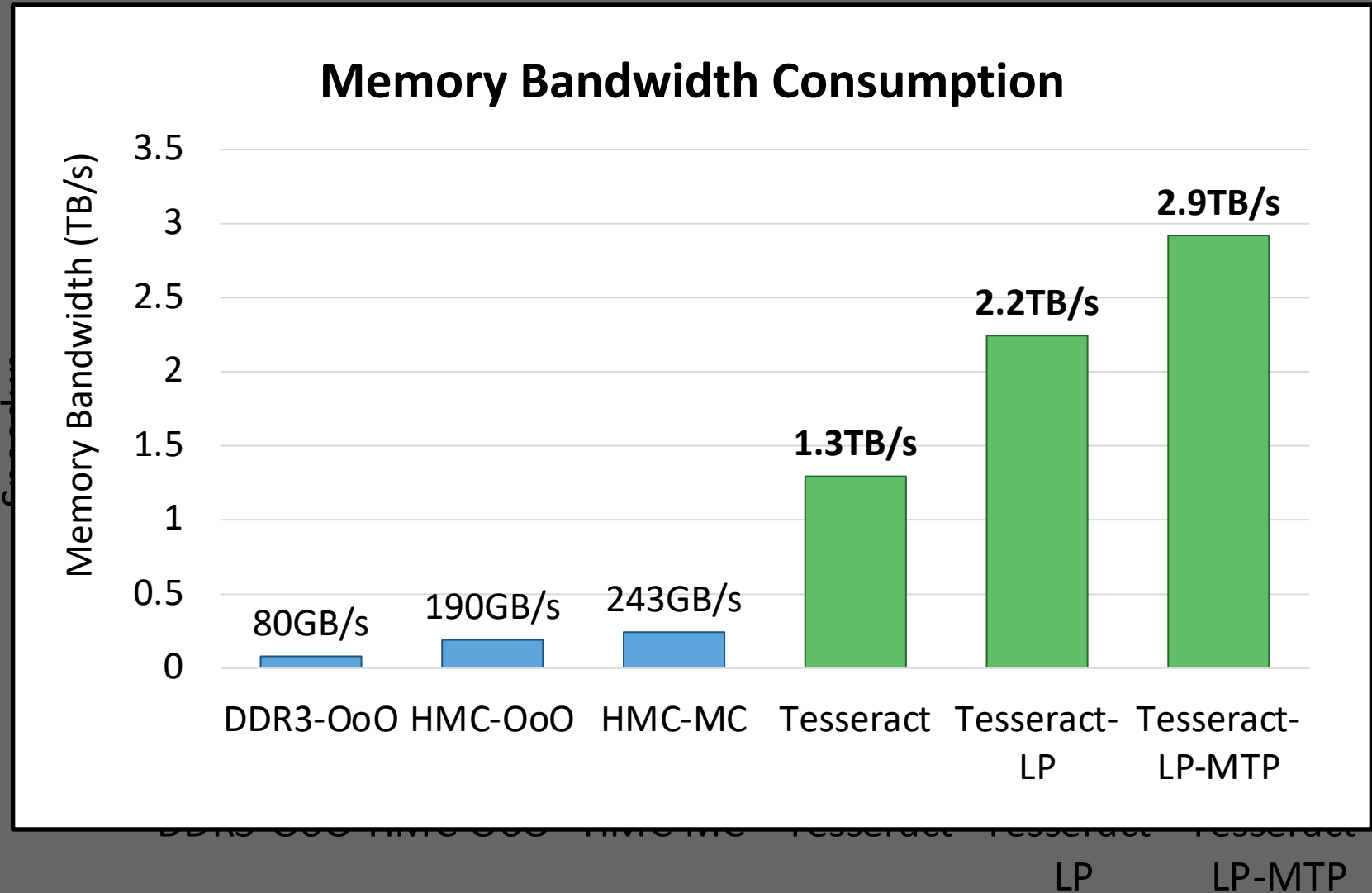
**8TB/s**

# Tesseract Graph Processing Performance

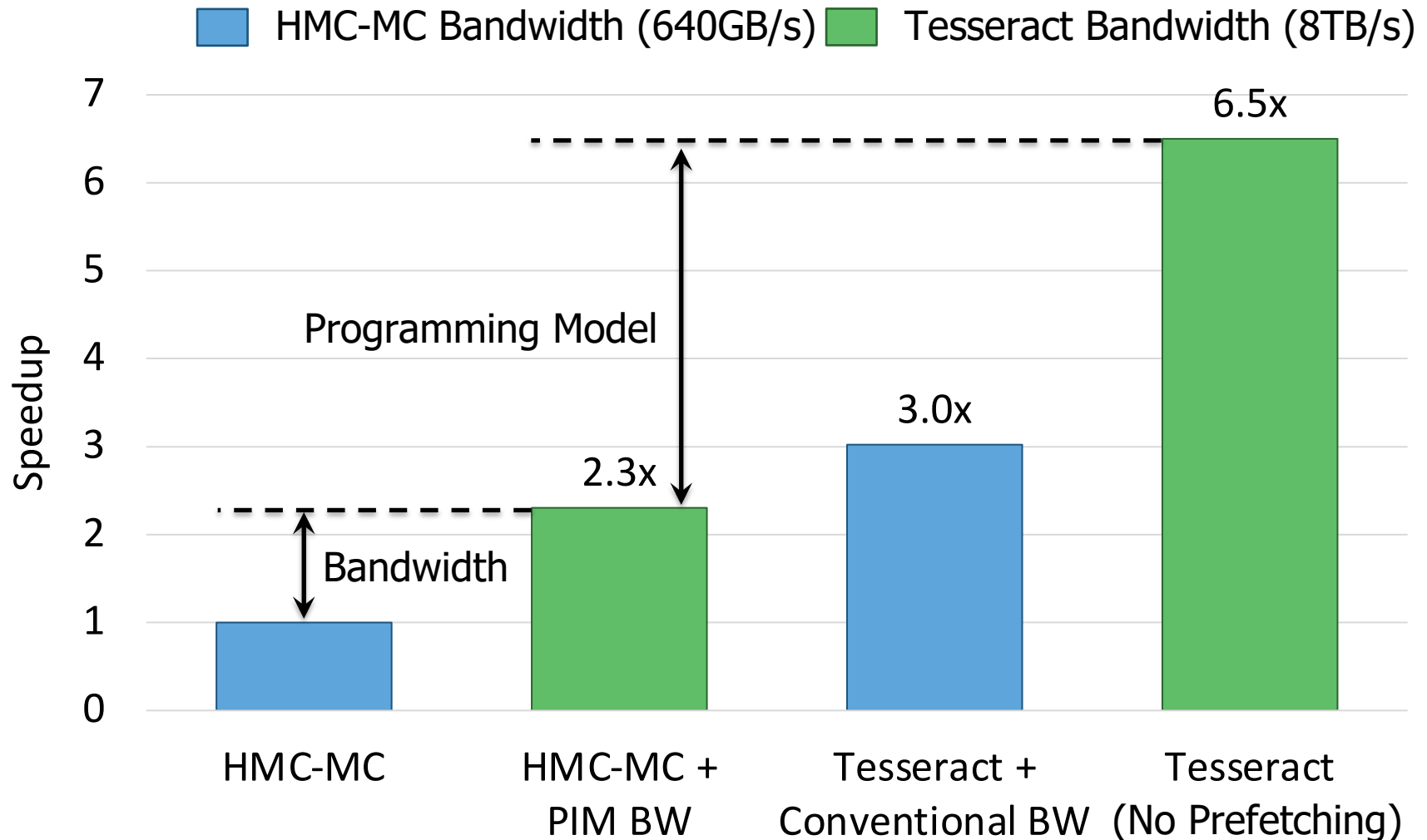
**>13X Performance Improvement**



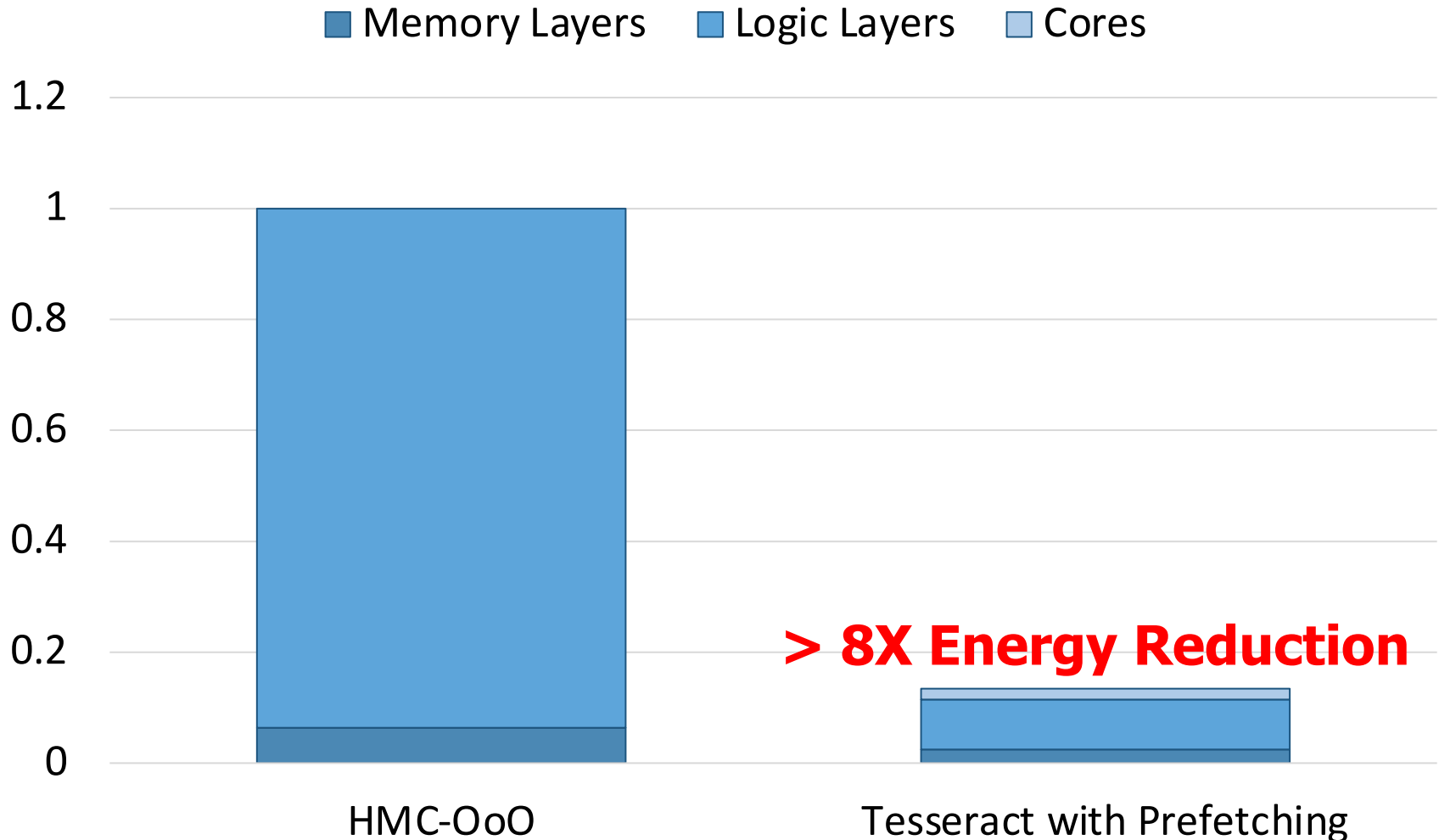
# Tesseract Graph Processing Performance



# Effect of Bandwidth & Programming Model



# Tesseract Graph Processing System Energy



# Tesseract: Advantages & Disadvantages

---

## ■ Advantages

- + Specialized graph processing accelerator using PIM
- + Large system performance and energy benefits
- + Takes advantage of 3D stacking for an important workload
- + More general than just graph processing

## ■ Disadvantages

- Changes a lot in the system
  - New programming model
  - Specialized Tesseract cores for graph processing
- Cost
- Scalability limited by off-chip links or graph partitioning

# More on Tesseract

---

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoungh Choi,  
**"A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"**  
*Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015.  
[\[Slides \(pdf\)\]](#) [\[Lightning Session Slides \(pdf\)\]](#)

## A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn   Sungpack Hong<sup>§</sup>   Sungjoo Yoo   Onur Mutlu<sup>†</sup>   Kiyoungh Choi

junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

<sup>§</sup>Oracle Labs

<sup>†</sup>Carnegie Mellon University



# Sub-Agenda: In-Memory Computation

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Processing using Memory
  - **Processing near Memory**
- How to Enable Adoption of Processing in Memory
- Conclusion

# Several Questions in 3D-Stacked PIM

---

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading
  
- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming

# 3D-Stacked PIM on Mobile Devices

---

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"**

*Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**), Williamsburg, VA, USA, March 2018.*

## Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand<sup>1</sup>

Saugata Ghose<sup>1</sup>

Youngsok Kim<sup>2</sup>

Rachata Ausavarungnirun<sup>1</sup>

Eric Shiu<sup>3</sup>

Rahul Thakur<sup>3</sup>

Daehyun Kim<sup>4,3</sup>

Aki Kuusela<sup>3</sup>

Allan Knies<sup>3</sup>

Parthasarathy Ranganathan<sup>3</sup>

Onur Mutlu<sup>5,1</sup>

# Consumer Devices



**Consumer devices are everywhere!**

**Energy consumption is  
a first-class concern in consumer devices**



# Four Important Workloads



**Chrome**

Google's web browser



**TensorFlow Mobile**

Google's machine learning  
framework

**VP9**



**Video Playback**

Google's **video codec**

**VP9**

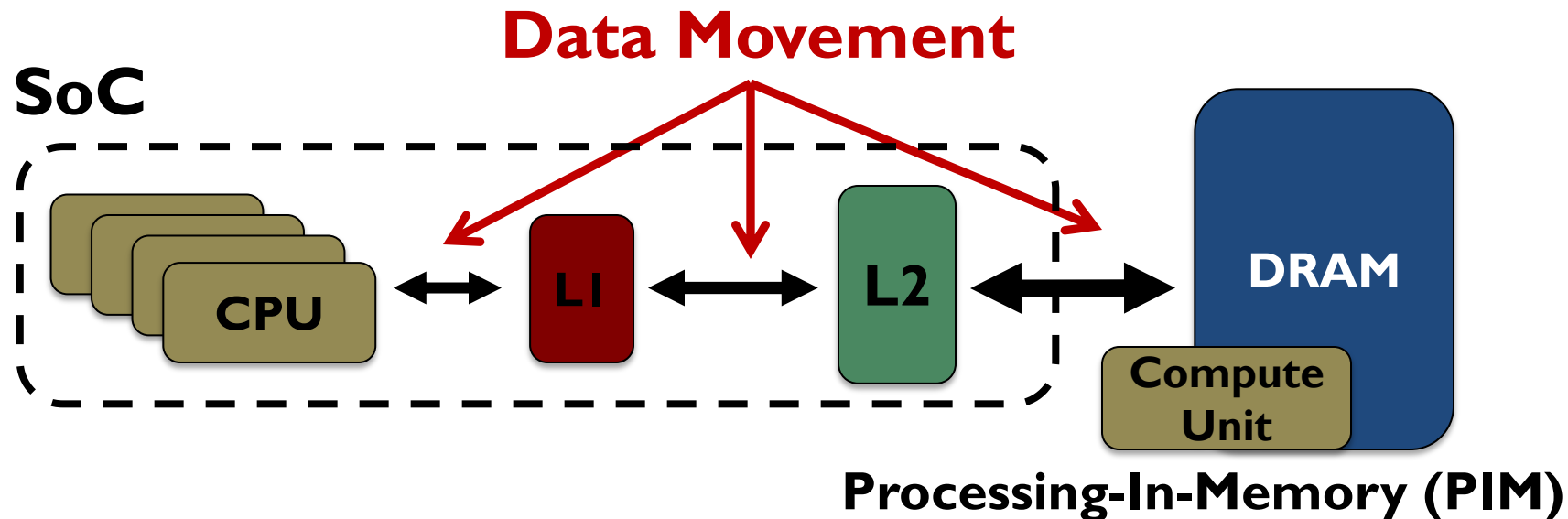


**Video Capture**

Google's **video codec**

# Energy Cost of Data Movement

**1<sup>st</sup> key observation:** **62.7%** of the total system energy is spent on **data movement**



**Potential solution:** move computation **close to data**

**Challenge:** limited area and energy budget

# Using PIM to Reduce Data Movement

**2<sup>nd</sup> key observation:** a significant fraction of the **data movement** often comes from **simple functions**

We can design lightweight logic to implement these simple functions in **memory**

Small embedded  
low-power core



Small fixed-function  
accelerators



Offloading to PIM logic reduces energy and improves performance, on average, by 55.4% and 54.2%



# Workload Analysis



**Chrome**

Google's web browser



**TensorFlow Mobile**

Google's machine learning  
framework

**VP9**



**Video Playback**

Google's **video codec**

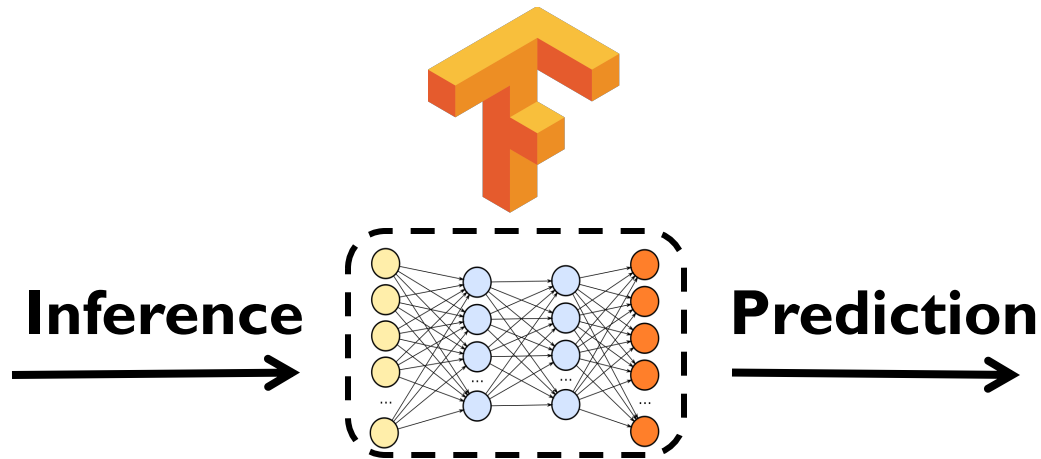
**VP9**



**Video Capture**

Google's **video codec**

# TensorFlow Mobile



**57.3%** of the inference energy is spent on data movement



**54.4%** of the **data movement** energy comes from packing/unpacking and quantization

# Packing



**Reorders** elements of matrices to minimize **cache misses** during **matrix multiplication**



Up to **40%** of the inference **energy** and **31%** of inference **execution time**



**Packing's data movement** accounts for up to **35.3%** of the inference **energy**

A simple **data reorganization** process that requires **simple arithmetic**

# Quantization



Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption



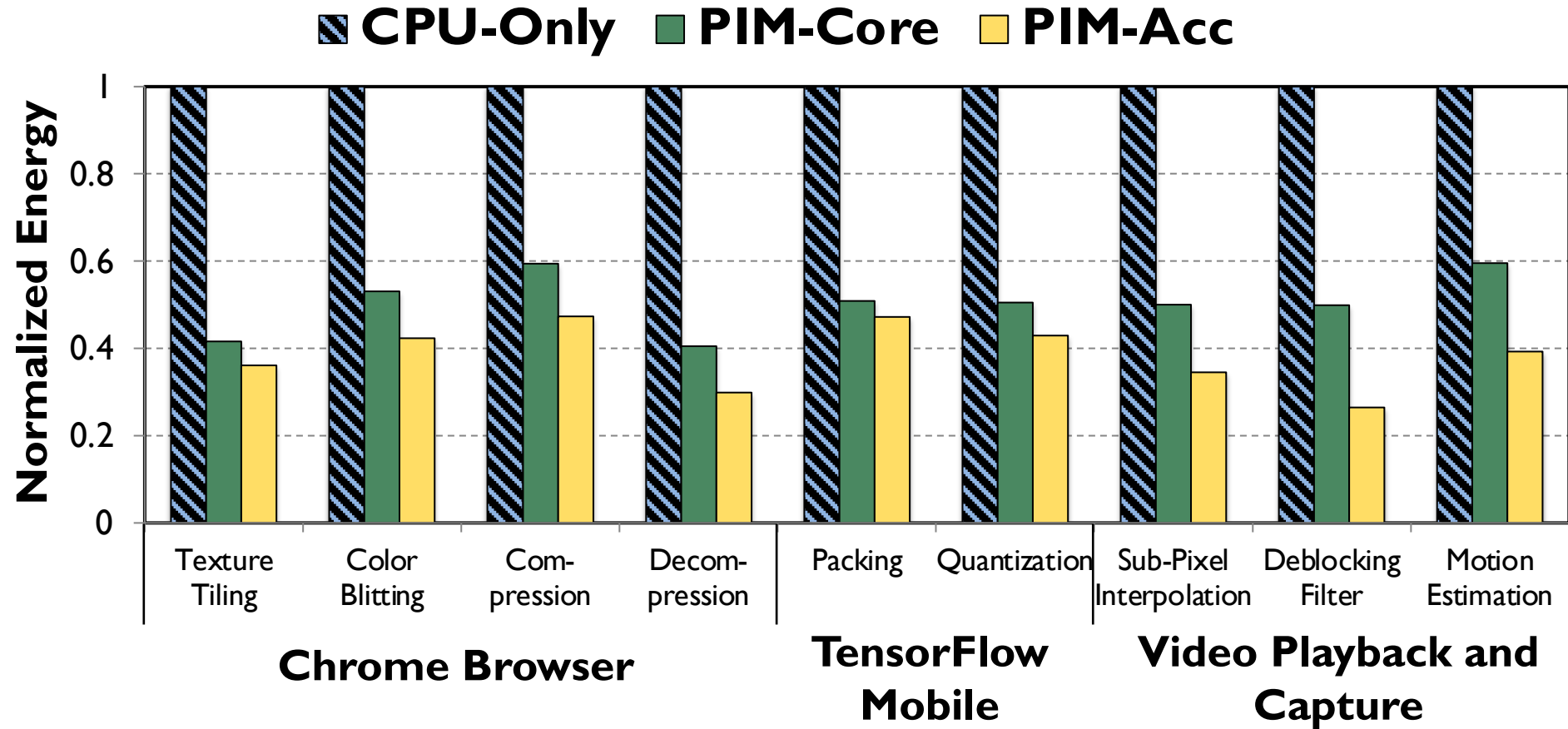
Up to **16.8%** of the inference **energy** and **16.1%** of inference **execution time**



Majority of **quantization** energy comes from **data movement**

A simple **data conversion** operation that requires **shift, addition, and multiplication** operations

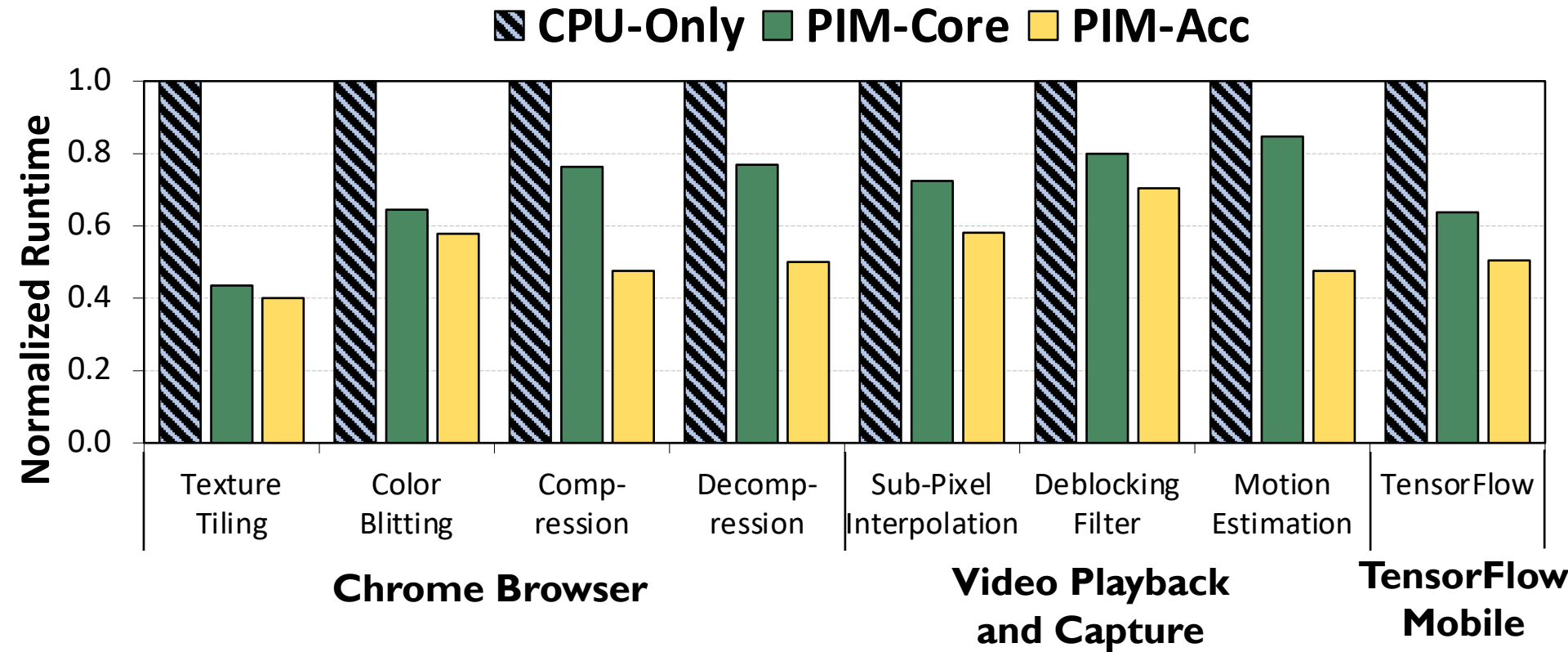
# Normalized Energy



**PIM core** and **PIM accelerator** reduce  
energy consumption on average by **49.1%** and **55.4%**

**SAFARI**

# Normalized Runtime



Offloading these kernels to **PIM core** and **PIM accelerator** improves **performance** on average by **44.6%** and **54.2%**

# Workload Analysis



**Chrome**

Google's web browser



**TensorFlow**

Google's machine learning  
framework

**VP9**



**Video Playback**

Google's video codec

**VP9**

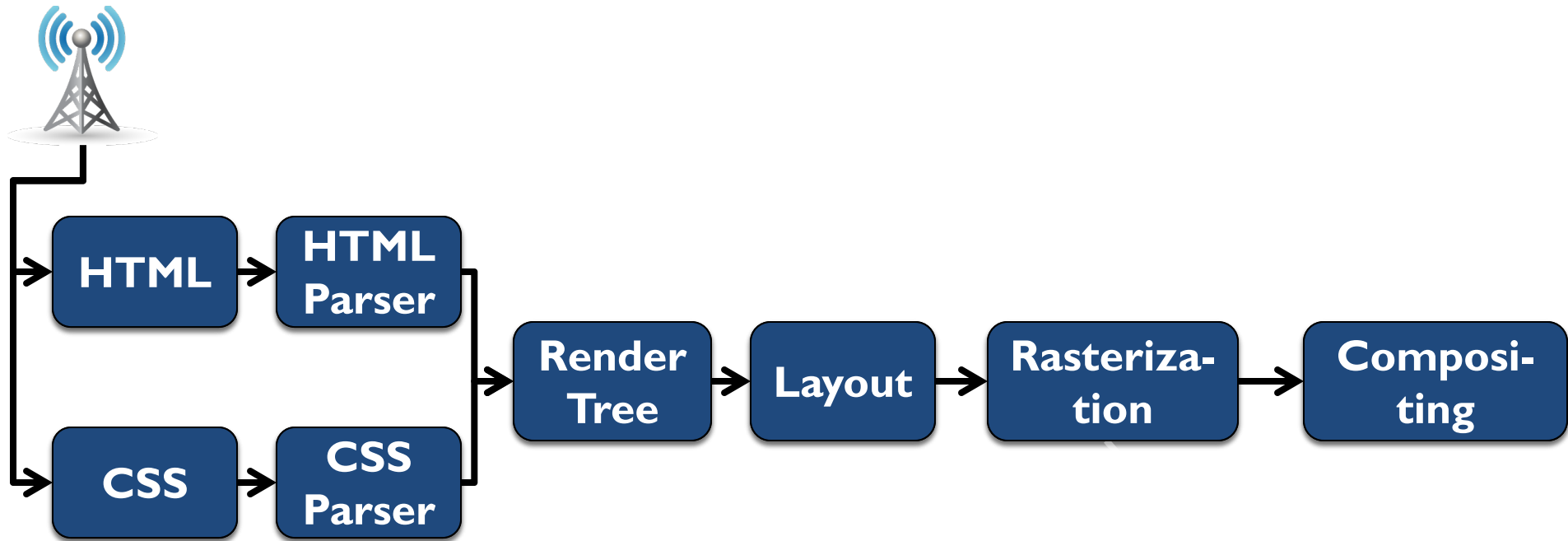


**Video Capture**

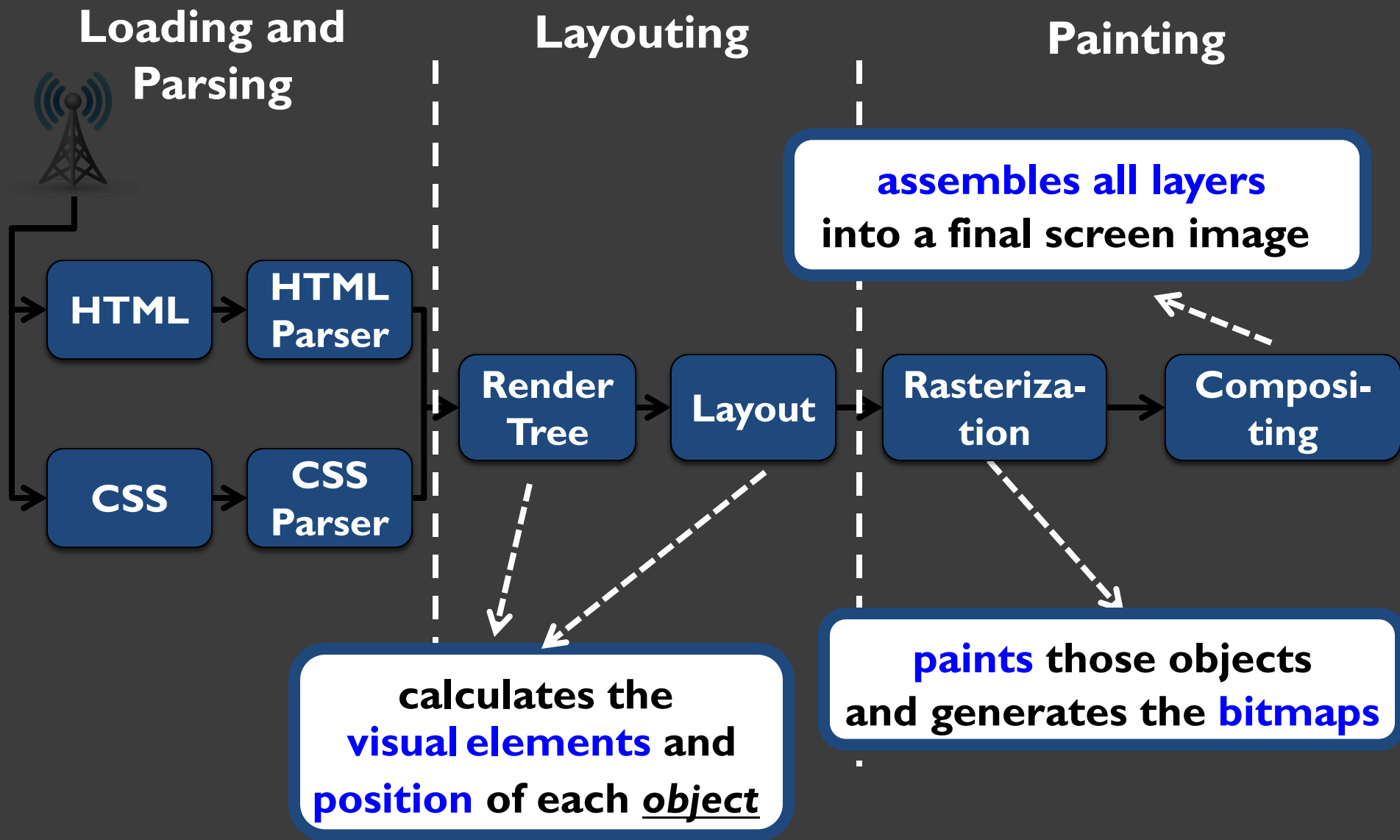
Google's video codec



# How Chrome Renders a Web Page



# How Chrome Renders a Web Page



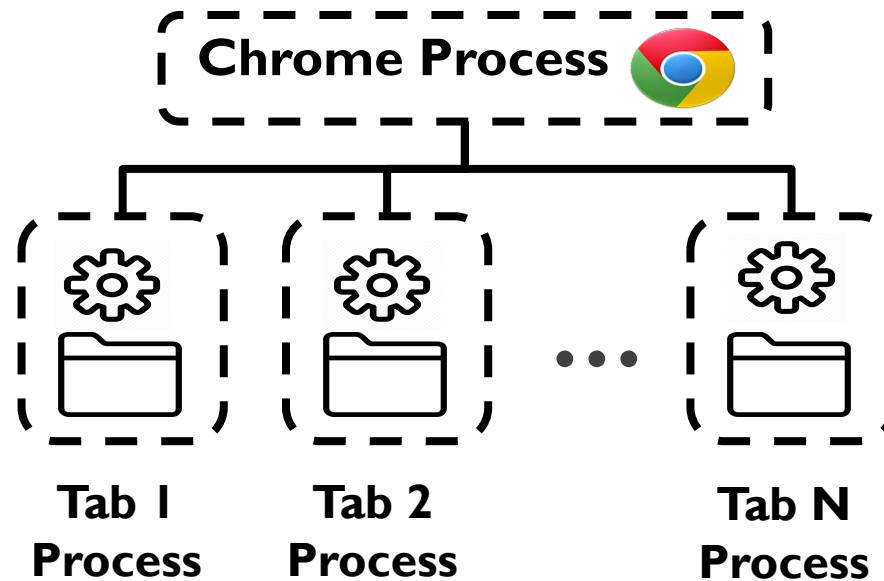
# Browser Analysis

- To satisfy user experience, the browser must provide:
  - Fast **loading** of webpages
  - Smooth **scrolling** of webpages
  - Quick **switching** between browser tabs
- We focus on two important user interactions:
  - 1) **Page Scrolling**
  - 2) **Tab Switching**
  - Both include page loading

# Tab Switching

# What Happens During Tab Switching?

- Chrome employs a **multi-process** architecture
  - Each tab is a separate process

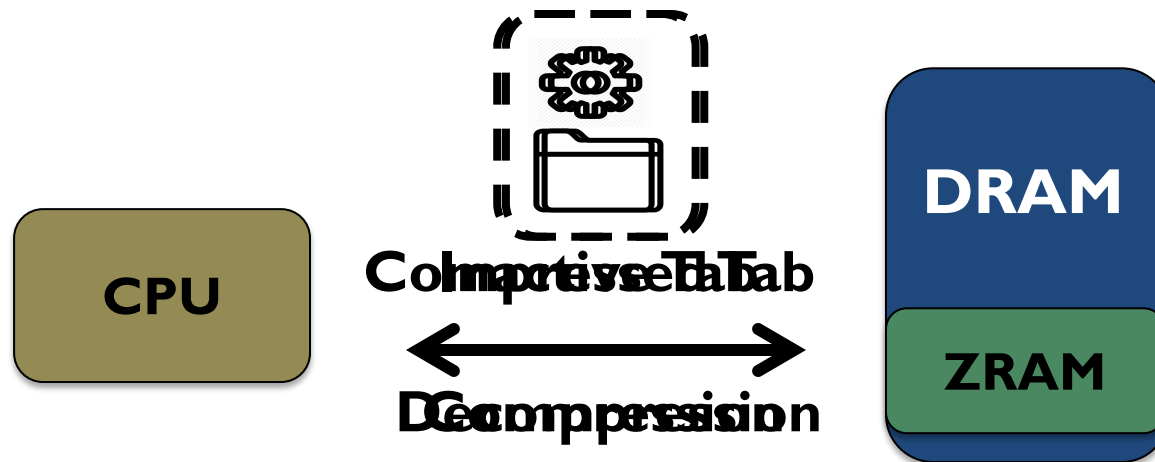


- Main operations during **tab switching**:
  - Context switch
  - Load the new page

# Memory Consumption

- **Primary concerns during tab switching:**
  - How fast a new tab **loads** and **becomes interactive**
  - **Memory consumption**

Chrome uses **compression** to reduce each tab's **memory footprint**



# Data Movement Study

- To study **data movement** during tab switching, we emulate a user switching through 50 tabs

We make two **key observations**:

1

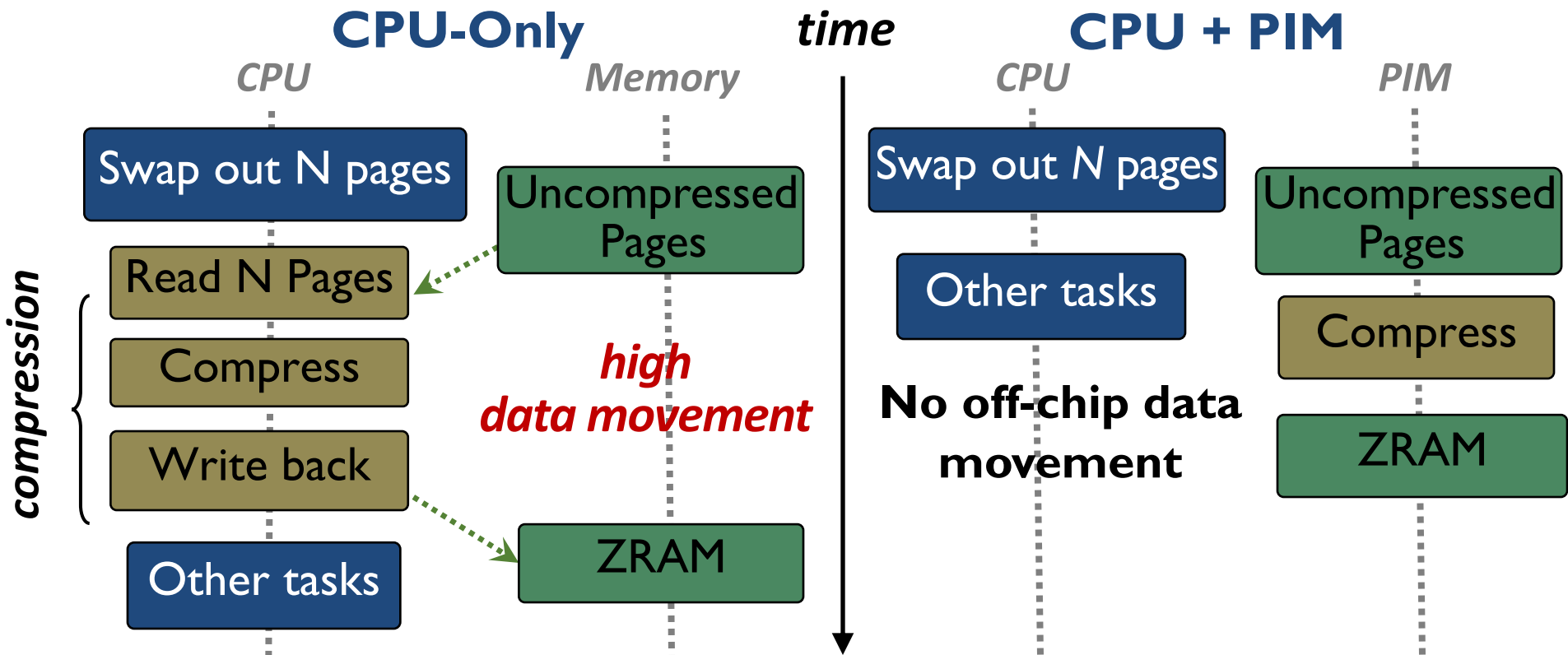
**Compression and decompression**  
contribute to **18.1%** of the total system energy

2

**19.6 GB** of data moves between  
**CPU** and **ZRAM**



# Can We Use PIM to Mitigate the Cost?



**PIM core and PIM accelerator are feasible to implement in-memory compression/decompression**

# Tab Switching Wrap Up

A large amount of **data movement** happens during **tab switching** as Chrome attempts to **compress** and **decompress** tabs

**Both functions can benefit from PIM execution and can be implemented as PIM logic**

# More on PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Williamsburg, VA, USA, March 2018.

**62.7% of the total system energy  
is spent on data movement**

## Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand<sup>1</sup>

Saugata Ghose<sup>1</sup>

Youngsok Kim<sup>2</sup>

Rachata Ausavarungnirun<sup>1</sup>

Eric Shiu<sup>3</sup>

Rahul Thakur<sup>3</sup>

Daehyun Kim<sup>4,3</sup>

Aki Kuusela<sup>3</sup>

Allan Knies<sup>3</sup>

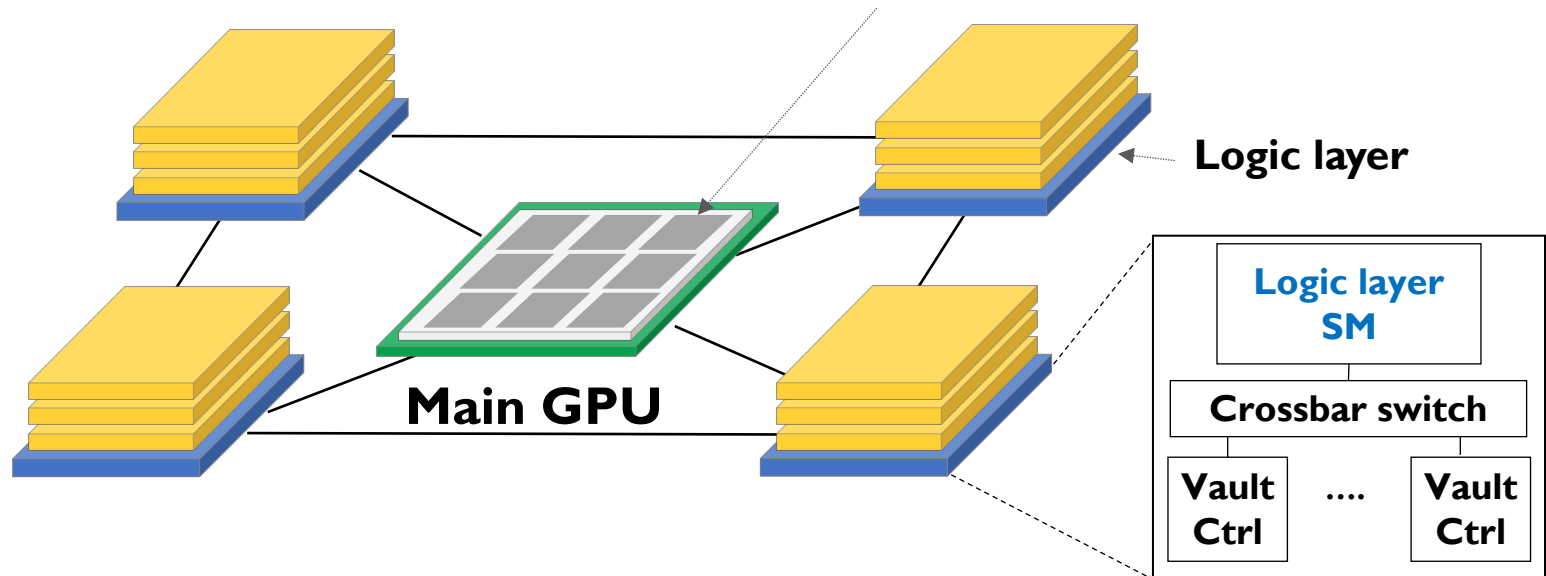
Parthasarathy Ranganathan<sup>3</sup>

Onur Mutlu<sup>5,1</sup>

# Truly Distributed GPU Processing with PIM?

**3D-stacked memory  
(memory stack)**

**SM (Streaming Multiprocessor)**



# Accelerating GPU Execution with PIM (I)

---

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**  
*Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea, June 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]

## Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh<sup>‡</sup> Eiman Ebrahimi<sup>†</sup> Gwangsun Kim\* Niladrish Chatterjee<sup>†</sup> Mike O'Connor<sup>†</sup>  
Nandita Vijaykumar<sup>‡</sup> Onur Mutlu<sup>§‡</sup> Stephen W. Keckler<sup>†</sup>

<sup>‡</sup>Carnegie Mellon University <sup>†</sup>NVIDIA <sup>\*</sup>KAIST <sup>§</sup>ETH Zürich

# Accelerating GPU Execution with PIM (II)

---

- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das,  
**"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"**  
*Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Haifa, Israel, September 2016.

## Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik<sup>1</sup>    Xulong Tang<sup>1</sup>    Adwait Jog<sup>2</sup>    Onur Kayiran<sup>3</sup>  
Asit K. Mishra<sup>4</sup>    Mahmut T. Kandemir<sup>1</sup>    Onur Mutlu<sup>5,6</sup>    Chita R. Das<sup>1</sup>

<sup>1</sup>Pennsylvania State University    <sup>2</sup>College of William and Mary  
<sup>3</sup>Advanced Micro Devices, Inc.    <sup>4</sup>Intel Labs    <sup>5</sup>ETH Zürich    <sup>6</sup>Carnegie Mellon University

# Accelerating Linked Data Structures

---

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,  
**"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"**  
*Proceedings of the 34th IEEE International Conference on Computer Design (ICCD)*, Phoenix, AZ, USA, October 2016.

## Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh<sup>†</sup>   Samira Khan<sup>‡</sup>   Nandita Vijaykumar<sup>†</sup>  
Kevin K. Chang<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Saugata Ghose<sup>†</sup>   Onur Mutlu<sup>§†</sup>  
<sup>†</sup>*Carnegie Mellon University*   <sup>‡</sup>*University of Virginia*   <sup>§</sup>*ETH Zürich*



# Accelerating Dependent Cache Misses

---

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**

*Proceedings of the 43rd International Symposium on Computer Architecture (ISCA), Seoul, South Korea, June 2016.*

[[Slides \(pptx\)](#) ([pdf](#))]

[[Lightning Session Slides \(pptx\)](#) ([pdf](#))]

## Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi\*, Khubaib<sup>†</sup>, Eiman Ebrahimi<sup>‡</sup>, Onur Mutlu<sup>§</sup>, Yale N. Patt\*

\*The University of Texas at Austin    <sup>†</sup>Apple    <sup>‡</sup>NVIDIA    <sup>§</sup>ETH Zürich & Carnegie Mellon University

# Accelerating Runahead Execution

---

- Milad Hashemi, Onur Mutlu, and Yale N. Patt,  
**"Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads"**  
*Proceedings of the 49th International Symposium on Microarchitecture (MICRO)*, Taipei, Taiwan, October 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pdf\)](#)] [[Poster \(pptx\)](#)] [[pdf](#)]

## Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

Milad Hashemi\*, Onur Mutlu<sup>§</sup>, Yale N. Patt\*

\*The University of Texas at Austin    <sup>§</sup>ETH Zürich

# Accelerating Climate Modeling

---

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,  
**"NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"**  
*Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, September 2020.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]  
[[Talk Video](#) (23 minutes)]  
***Nominated for the Stamatis Vassiliadis Memorial Award.***

## NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh<sup>a,b,c</sup>    Dionysios Diamantopoulos<sup>c</sup>    Christoph Hagleitner<sup>c</sup>    Juan Gómez-Luna<sup>b</sup>  
Sander Stuijk<sup>a</sup>    Onur Mutlu<sup>b</sup>    Henk Corporaal<sup>a</sup>  
<sup>a</sup>Eindhoven University of Technology    <sup>b</sup>ETH Zürich    <sup>c</sup>IBM Research Europe, Zurich

# Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zülal Bingöl, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu, **"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"**  
*Proceedings of the 53rd International Symposium on Microarchitecture (MICRO)*, Virtual, October 2020.  
[[Lighting Talk Video](#) (1.5 minutes)]  
[[Lightning Talk Slides \(pptx\)](#) ([pdf](#))]  
[[Talk Video](#) (18 minutes)]  
[[Slides \(pptx\)](#) ([pdf](#))]

## GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali<sup>†⋈</sup> Gurpreet S. Kalsi<sup>⋈</sup> Zülal Bingöl<sup>▽</sup> Can Firtina<sup>◇</sup> Lavanya Subramanian<sup>‡</sup> Jeremie S. Kim<sup>◇†</sup>  
Rachata Ausavarungnirun<sup>○</sup> Mohammed Alser<sup>◇</sup> Juan Gomez-Luna<sup>◇</sup> Amirali Boroumand<sup>†</sup> Anant Nori<sup>⋈</sup>  
Allison Scibisz<sup>†</sup> Sreenivas Subramoney<sup>⋈</sup> Can Alkan<sup>▽</sup> Saugata Ghose<sup>\*†</sup> Onur Mutlu<sup>◇†▽</sup>  
<sup>†</sup>Carnegie Mellon University   <sup>⋈</sup>Processor Architecture Research Lab, Intel Labs   <sup>▽</sup>Bilkent University   <sup>◇</sup>ETH Zürich  
<sup>‡</sup>Facebook   <sup>○</sup>King Mongkut's University of Technology North Bangkok   <sup>\*</sup>University of Illinois at Urbana-Champaign

# Accelerating Time Series Analysis

---

- Ivan Fernandez, Ricardo Quisiant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,  
**"NATSA: A Near-Data Processing Accelerator for Time Series Analysis"**  
*Proceedings of the 38th IEEE International Conference on Computer Design (ICCD)*, Virtual, October 2020.

## NATSA: A Near-Data Processing Accelerator for Time Series Analysis

Ivan Fernandez <sup>§</sup>	Ricardo Quisiant <sup>§</sup>	Christina Giannoula <sup>†</sup>	Mohammed Alser <sup>‡</sup>
Juan Gómez-Luna <sup>‡</sup>	Eladio Gutiérrez <sup>§</sup>	Oscar Plata <sup>§</sup>	Onur Mutlu <sup>‡</sup>
<sup>§</sup> <i>University of Malaga</i>	<sup>†</sup> <i>National Technical University of Athens</i>	<sup>‡</sup> <i>ETH Zürich</i>	



# Several Questions in 3D-Stacked PIM

---

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading
- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming

# PIM-Enabled Instructions

---

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoungh Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015. [[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

## **PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture**

Junwhan Ahn   Sungjoo Yoo   Onur Mutlu<sup>†</sup>   Kiyoungh Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

<sup>†</sup>Carnegie Mellon University

# PEI: PIM-Enabled Instructions (Ideas)

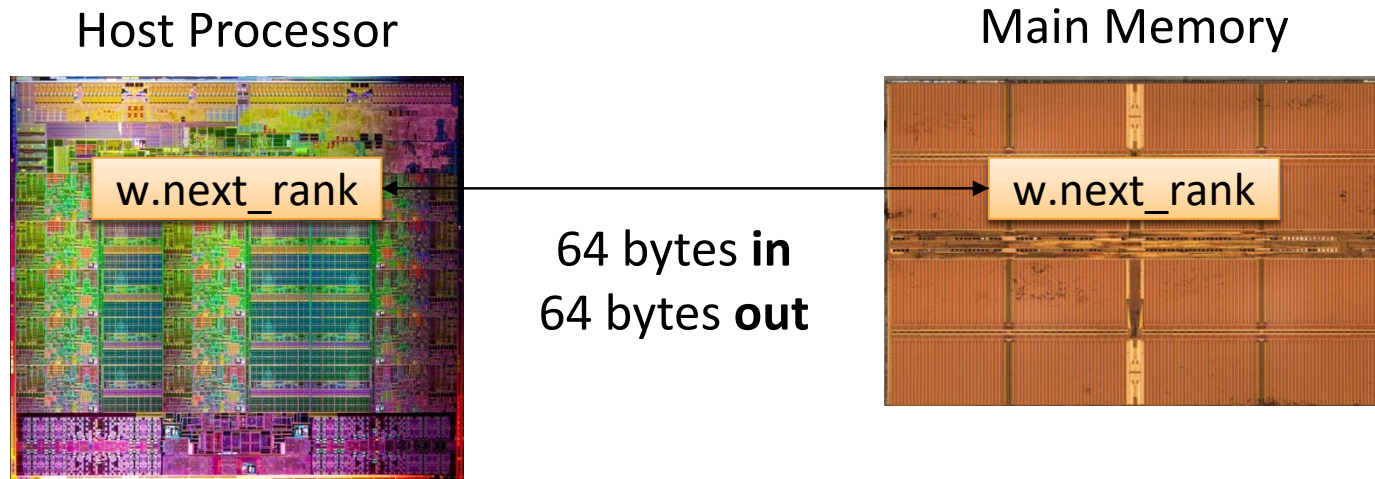
---

- **Goal:** Develop mechanisms to get the most out of near-data processing with **minimal cost, minimal changes to the system, no changes to the programming model**
- **Key Idea 1:** Expose each PIM operation as a **cache-coherent, virtually-addressed host processor instruction** (called PEI) that operates on **only a single cache block**
  - e.g., `__pim_add(&w.next_rank, value) → pim.add r1, (r2)`
  - No changes sequential execution/programming model
  - No changes to virtual memory
  - Minimal changes to cache coherence
  - No need for data mapping: Each PEI restricted to a single memory module
- **Key Idea 2:** **Dynamically decide where to execute a PEI** (i.e., the host processor or PIM accelerator) based on simple locality characteristics and simple hardware predictors
  - Execute each operation at the location that provides the best performance



# Simple PIM Operations as ISA Extensions (II)

```
for (v: graph.vertices) {  
    value = weight * v.rank;  
    for (w: v.successors) {  
        w.next_rank += value;  
    }  
}
```



Conventional Architecture

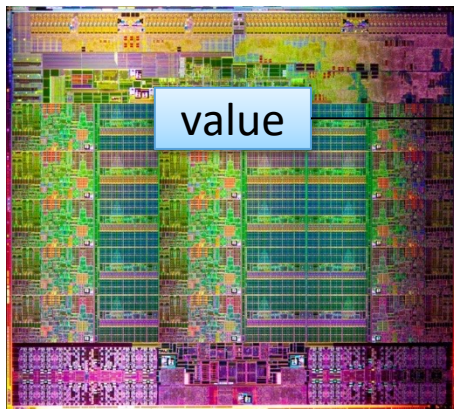
# Simple PIM Operations as ISA Extensions (III)

```
for (v: graph.vertices) {  
    value = weight * v.rank;  
    for (w: v.successors) {  
        __pim_add(&w.next_rank, value);  
    }  
}
```

pim.add r1, (r2)

\_\_pim\_add(&w.next\_rank, value);

Host Processor



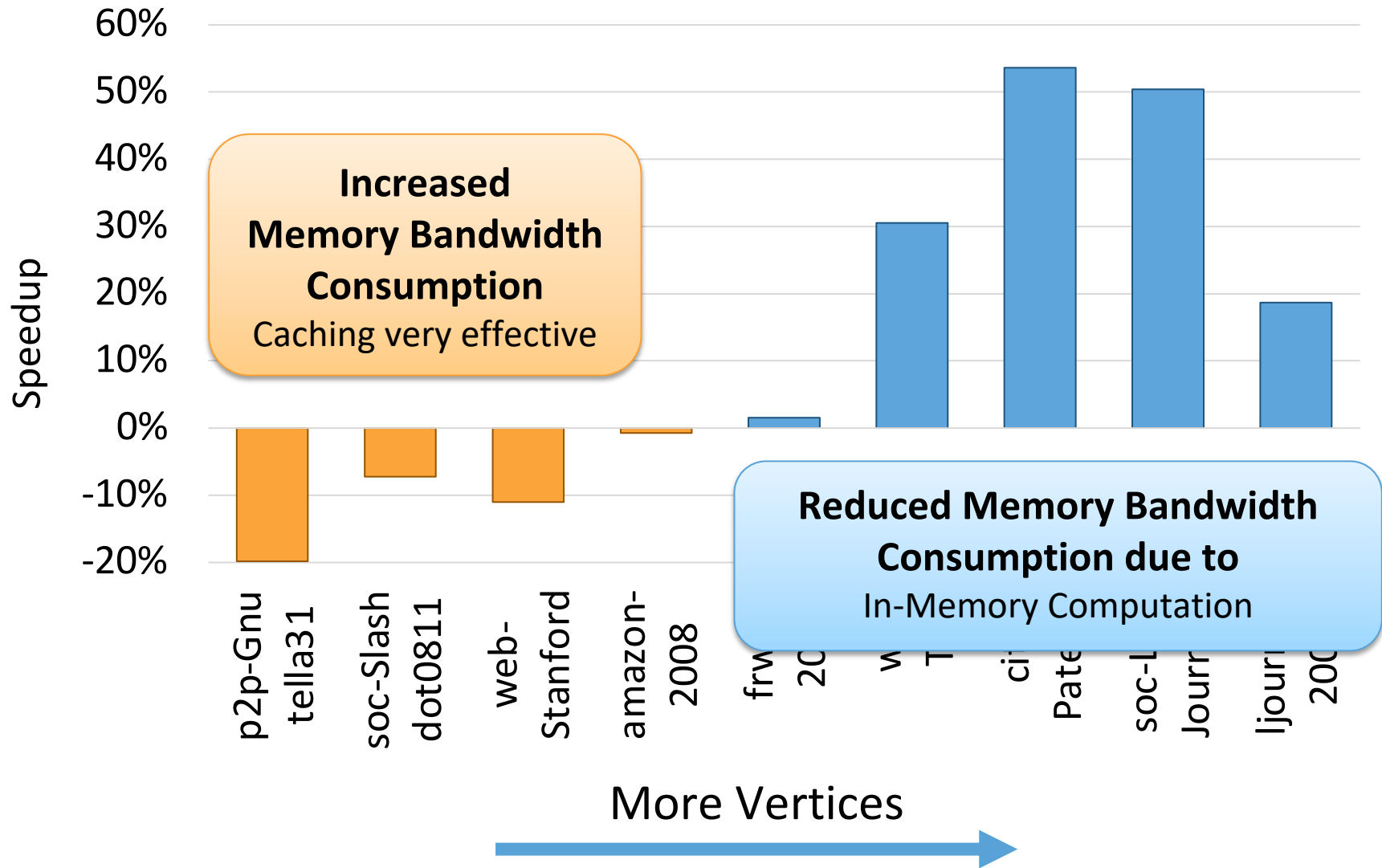
Main Memory



8 bytes in  
0 bytes out

**In-Memory Addition**

# Always Executing in Memory? Not A Good Idea



# PEI: PIM-Enabled Instructions (Example)

```
for (v: graph.vertices) {  
    value = weight * v.rank;  
    for (w: v.successors) {
```

pim.add r1, (r2)

```
        __pim_add(&w.next_rank, value);  
    }
```

pfence

```
pfence();  
}
```

Table 1: Summary of Supported PIM Operations

Operation	R	W	Input	Output	Applications
8-byte integer increment	O	O	0 bytes	0 bytes	AT
8-byte integer min	O	O	8 bytes	0 bytes	BFS, SP, WCC
Floating-point add	O	O	8 bytes	0 bytes	PR
Hash table probing	O	X	8 bytes	9 bytes	HJ
Histogram bin index	O	X	1 byte	16 bytes	HG, RP
Euclidean distance	O	X	64 bytes	4 bytes	SC
Dot product	O	X	32 bytes	8 bytes	SVM

- Executed either in memory or in the processor: dynamic decision
  - ❑ Low-cost locality monitoring for a single instruction
- Cache-coherent, virtually-addressed, single cache block only
- Atomic between different PEIs
- *Not* atomic with normal instructions (use *pfence* for ordering)

# PIM-Enabled Instructions

---

- Key to practicality: **single-cache-block restriction**
  - **Each PEI can access *at most one last-level cache block***
  - Similar restrictions exist in atomic instructions
- Benefits
  - **Localization**: each PEI is bounded to one memory module
  - **Interoperability**: easier support for cache coherence and virtual memory
  - **Simplified locality monitoring**: data locality of PEIs can be identified simply by the cache control logic

# PEI: Initial Evaluation Results

- Initial evaluations with **10 emerging data-intensive workloads**
  - ❑ Large-scale graph processing
  - ❑ In-memory data analytics
  - ❑ Machine learning and data mining
  - ❑ Three input sets (small, medium, large) for each workload to analyze the impact of data locality
- Pin-based cycle-level x86-64 simulation
- **Performance Improvement and Energy Reduction:**
  - 47% average speedup with large input data sets
  - 32% speedup with small input data sets
  - 25% avg. energy reduction in a single node with large input data sets

**Table 2: Baseline Simulation Configuration**

Component	Configuration
Core	16 out-of-order cores, 4 GHz, 4-issue
L1 I/D-Cache	Private, 32 KB, 4/8-way, 64 B blocks, 16 MSHRs
L2 Cache	Private, 256 KB, 8-way, 64 B blocks, 16 MSHRs
L3 Cache	Shared, 16 MB, 16-way, 64 B blocks, 64 MSHRs
On-Chip Network	Crossbar, 2 GHz, 144-bit links
Main Memory	32 GB, 8 HMCs, daisy-chain (80 GB/s full-duplex)
HMC	4 GB, 16 vaults, 256 DRAM banks [20]
– DRAM	FR-FCFS, tCL = tRCD = tRP = 13.75 ns [27]
– Vertical Links	64 TSVs per vault with 2 Gb/s signaling rate [23]

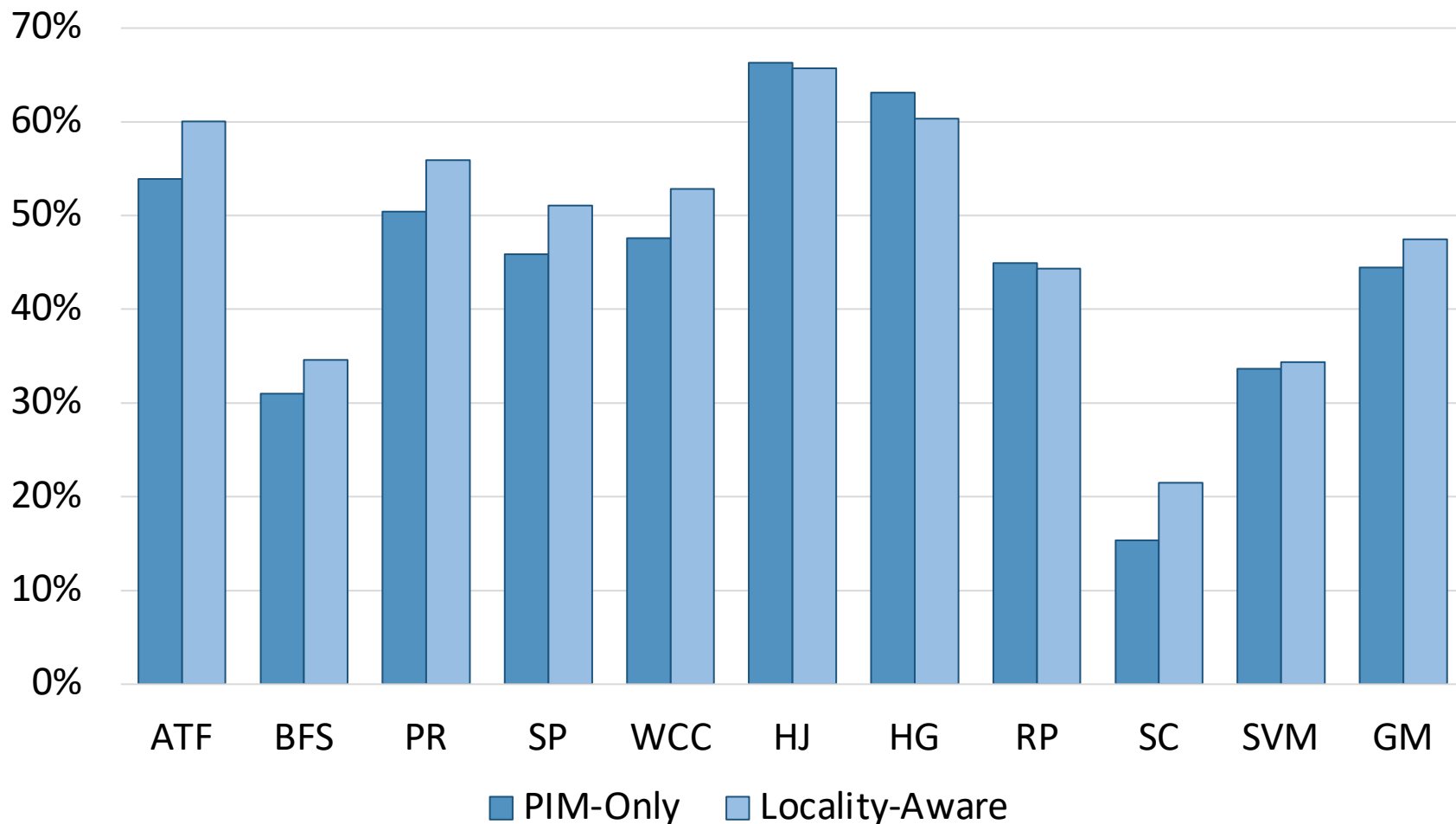
# Evaluated Data-Intensive Applications

---

- Ten emerging data-intensive workloads
  - Large-scale graph processing
    - Average teenage follower, BFS, PageRank, single-source shortest path, weakly connected components
  - In-memory data analytics
    - Hash join, histogram, radix partitioning
  - Machine learning and data mining
    - Streamcluster, SVM-RFE
- Three input sets (small, medium, large) for each workload to show the impact of data locality

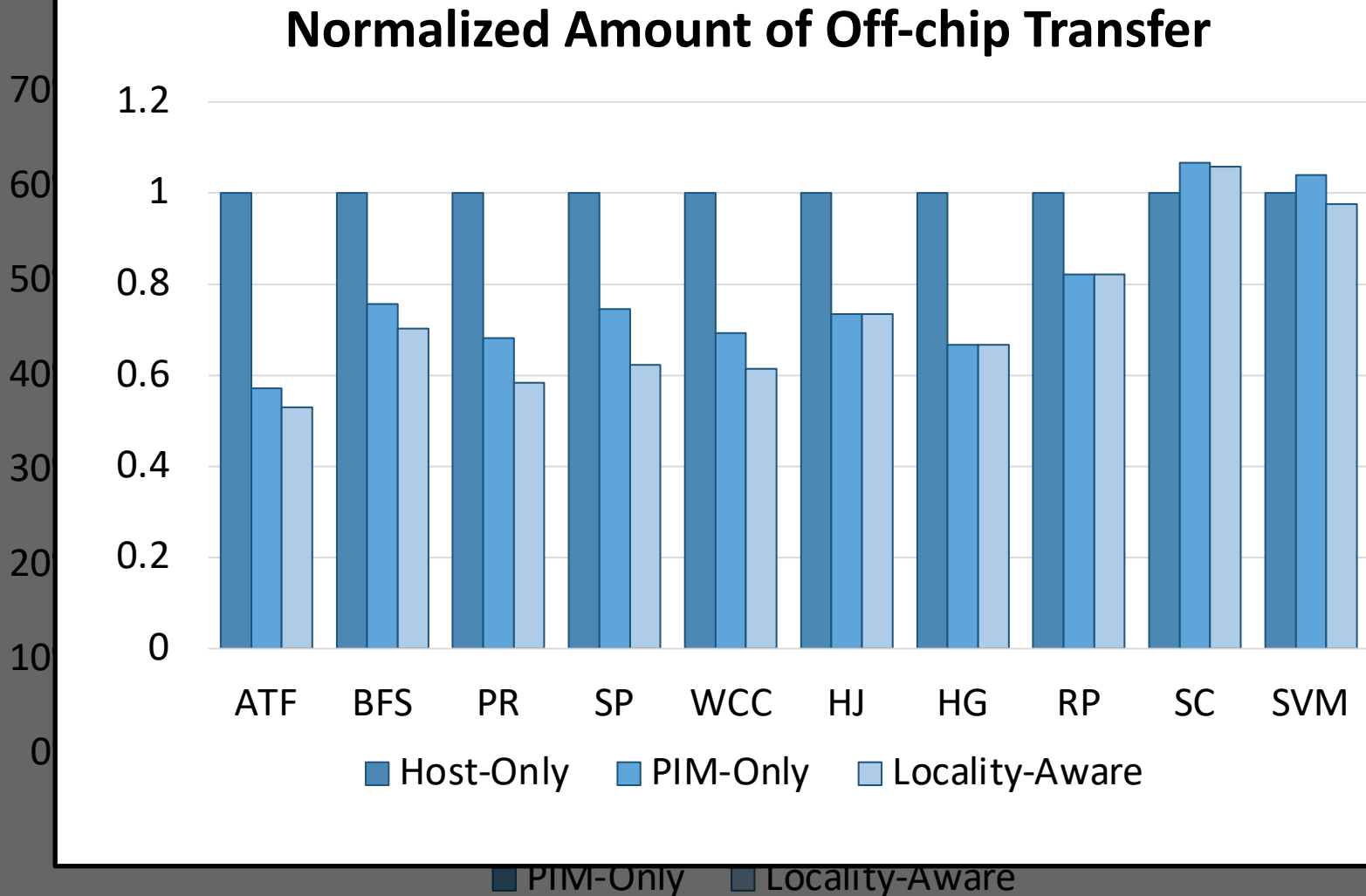
# PEI Performance Delta: Large Data Sets

(Large Inputs, Baseline: Host-Only)



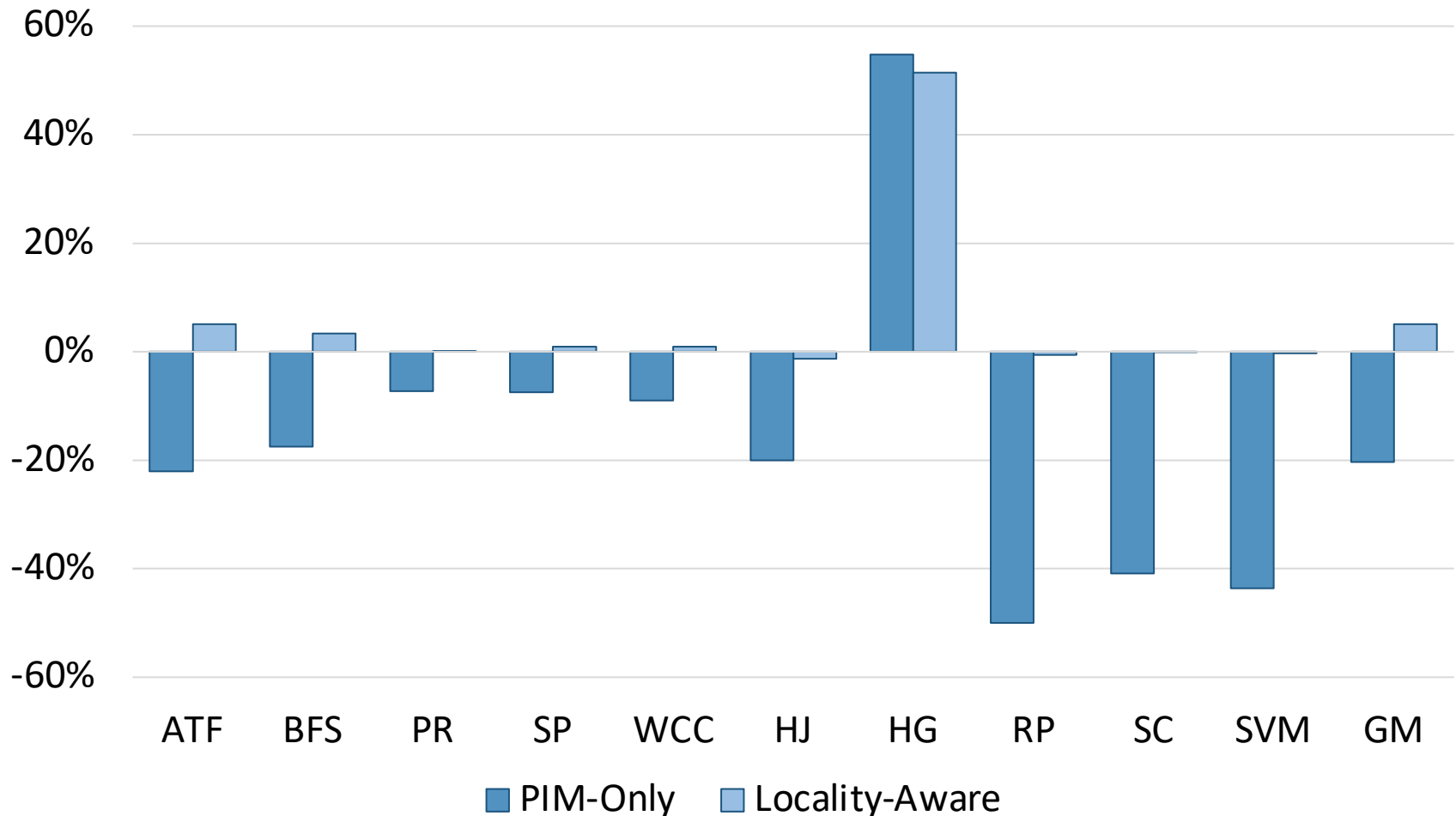


# PEI Performance: Large Data Sets

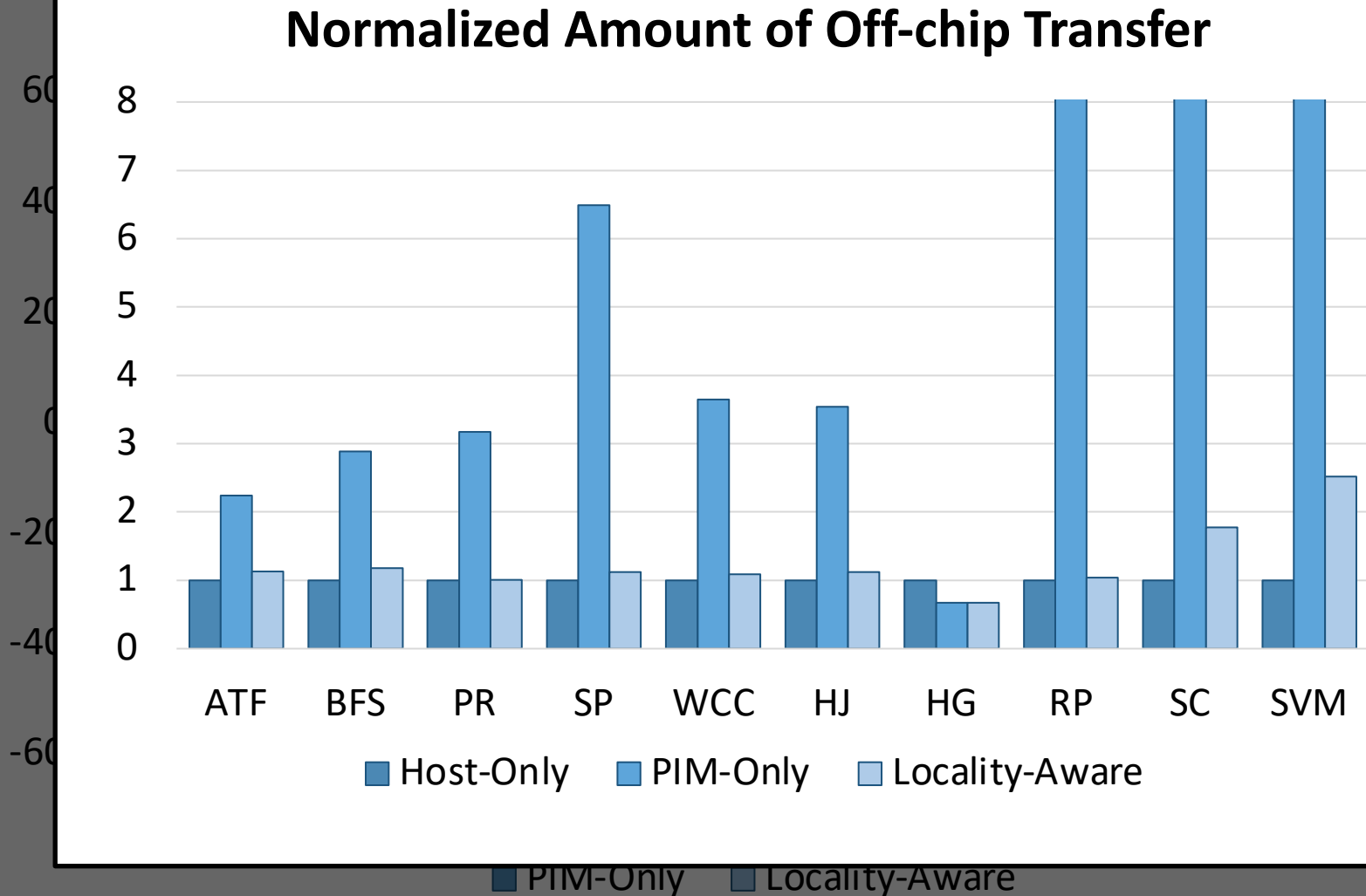


# PEI Performance Delta: Small Data Sets

(Small Inputs, Baseline: Host-Only)

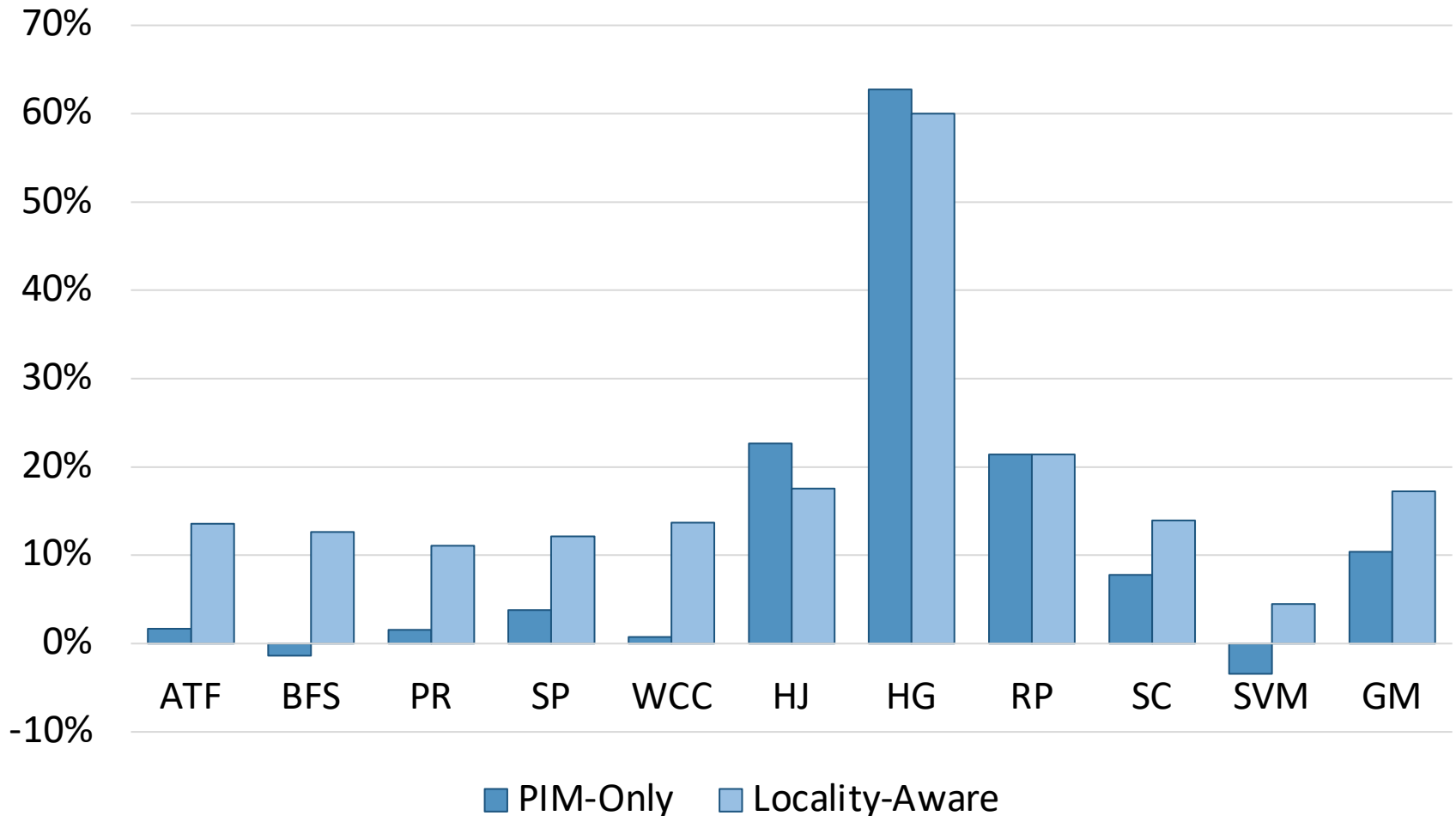


# PEI Performance: Small Data Sets

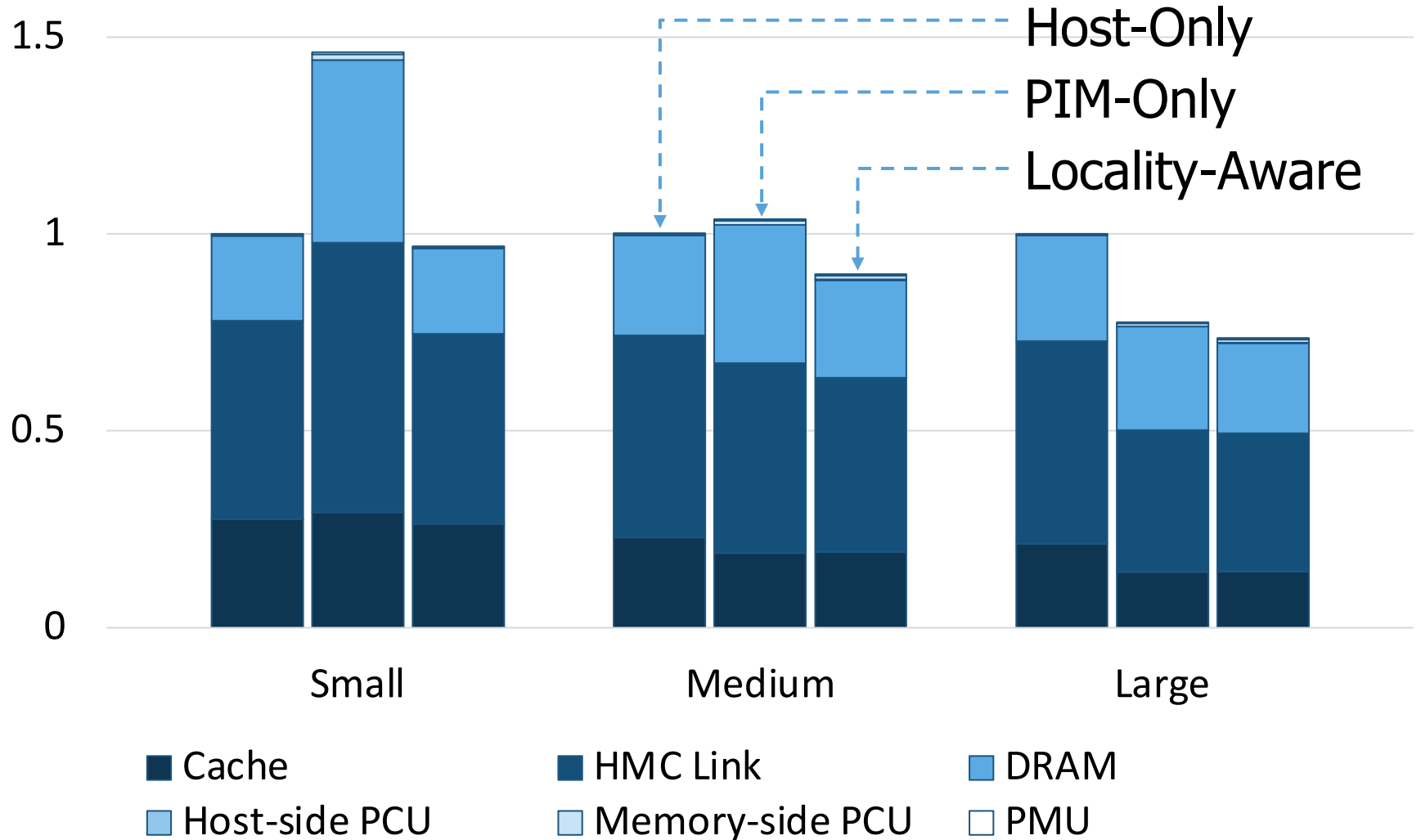


# PEI Performance Delta: Medium Data Sets

(Medium Inputs, Baseline: Host-Only)



# PEI Energy Consumption



# PEI: Advantages & Disadvantages

---

## ■ Advantages

- + Simple and low cost approach to PIM
- + No changes to programming model, virtual memory
- + Dynamically decides where to execute an instruction

## ■ Disadvantages

- Does not take full advantage of PIM potential
  - Single cache block restriction is limiting

# Simpler PIM: PIM-Enabled Instructions

---

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoungh Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015. [[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

## **PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture**

Junwhan Ahn   Sungjoo Yoo   Onur Mutlu<sup>†</sup>   Kiyoungh Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

<sup>†</sup>Carnegie Mellon University

# Automatic Code and Data Mapping

---

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**  
*Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea, June 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]

## Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh<sup>‡</sup> Eiman Ebrahimi<sup>†</sup> Gwangsun Kim\* Niladrish Chatterjee<sup>†</sup> Mike O'Connor<sup>†</sup>  
Nandita Vijaykumar<sup>‡</sup> Onur Mutlu<sup>§‡</sup> Stephen W. Keckler<sup>†</sup>

<sup>‡</sup>Carnegie Mellon University <sup>†</sup>NVIDIA \*KAIST <sup>§</sup>ETH Zürich



# Automatic Offloading of Critical Code

---

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**

*Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea, June 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Lightning Session Slides \(pptx\)](#) ([pdf](#))]

## Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi\*, Khubaib<sup>†</sup>, Eiman Ebrahimi<sup>‡</sup>, Onur Mutlu<sup>§</sup>, Yale N. Patt\*

\*The University of Texas at Austin    <sup>†</sup>Apple    <sup>‡</sup>NVIDIA    <sup>§</sup>ETH Zürich & Carnegie Mellon University

# Automatic Offloading of Prefetch Mechanisms

---

- Milad Hashemi, Onur Mutlu, and Yale N. Patt,  
**"Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads"**  
*Proceedings of the 49th International Symposium on Microarchitecture (MICRO)*, Taipei, Taiwan, October 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pdf\)](#)] [[Poster \(pptx\)](#)] [[pdf](#)]

## Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

Milad Hashemi\*, Onur Mutlu<sup>§</sup>, Yale N. Patt\*

\**The University of Texas at Austin*    <sup>§</sup>*ETH Zürich*

# Efficient Automatic Data Coherence Support

---

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,  
**"LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory"**  
***IEEE Computer Architecture Letters* (**CAL**), June 2016.**

## LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

Amirali Boroumand<sup>†</sup>, Saugata Ghose<sup>†</sup>, Minesh Patel<sup>†</sup>, Hasan Hassan<sup>†§</sup>, Brandon Lucia<sup>†</sup>,  
Kevin Hsieh<sup>†</sup>, Krishna T. Malladi<sup>\*</sup>, Hongzhong Zheng<sup>\*</sup>, and Onur Mutlu<sup>††</sup>

<sup>†</sup>Carnegie Mellon University   <sup>\*</sup>Samsung Semiconductor, Inc.   <sup>§</sup>TOBB ETÜ   <sup>‡</sup>ETH Zürich

# Efficient Automatic Data Coherence Support

---

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,  
**"CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators"**

*Proceedings of the 46th International Symposium on Computer Architecture (ISCA), Phoenix, AZ, USA, June 2019.*

## CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand<sup>†</sup>

Saugata Ghose<sup>†</sup>

Minesh Patel<sup>★</sup>

Hasan Hassan<sup>★</sup>

Brandon Lucia<sup>†</sup>

Rachata Ausavarungnirun<sup>†‡</sup>

Kevin Hsieh<sup>†</sup>

Nastaran Hajinazar<sup>◇†</sup>

Krishna T. Malladi<sup>§</sup>

Hongzhong Zheng<sup>§</sup>

Onur Mutlu<sup>★†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>★</sup>ETH Zürich

<sup>‡</sup>KMUTNB

<sup>◇</sup>Simon Fraser University

<sup>§</sup>Samsung Semiconductor, Inc.

# Fundamentally Energy-Efficient (Data-Centric) Computing Architectures

# Fundamentally High-Performance (Data-Centric) Computing Architectures

# Computing Architectures with Minimal Data Movement

# Sub-Agenda: In-Memory Computation

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Processing using Memory
  - Processing near Memory
- How to Enable Adoption of Processing in Memory
- Conclusion



## How to Enable Adoption of Processing in Memory

# Barriers to Adoption of PIM

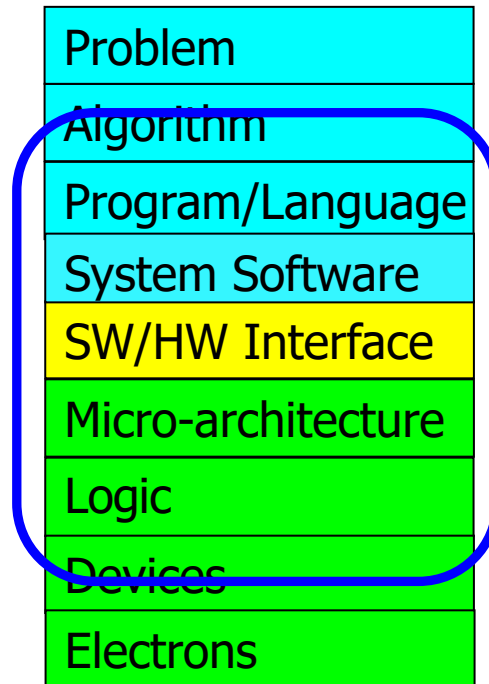
---

1. Functionality of and applications & software for PIM
2. Ease of programming (interfaces and compiler/HW support)
3. System support: coherence & virtual memory
4. Runtime and compilation systems for adaptive scheduling, data mapping, access/sharing control
5. Infrastructures to assess benefits and feasibility

**All can be solved with change of mindset**

# We Need to Revisit the Entire Stack

---



**We can get there step by step**

# PIM Review and Open Problems

---

## Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>b,c</sup>

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,  
**"Processing Data Where It Makes Sense: Enabling In-Memory  
Computation"**

*Invited paper in Microprocessors and Microsystems (**MICPRO**), June 2019.  
[arXiv version]*

# PIM Review and Open Problems (II)

---

## A Workload and Programming Ease Driven Perspective of Processing-in-Memory

Saugata Ghose<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Jeremie S. Kim<sup>†§</sup>   Juan Gómez-Luna<sup>§</sup>   Onur Mutlu<sup>§†</sup>

<sup>†</sup>*Carnegie Mellon University*

<sup>§</sup>*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

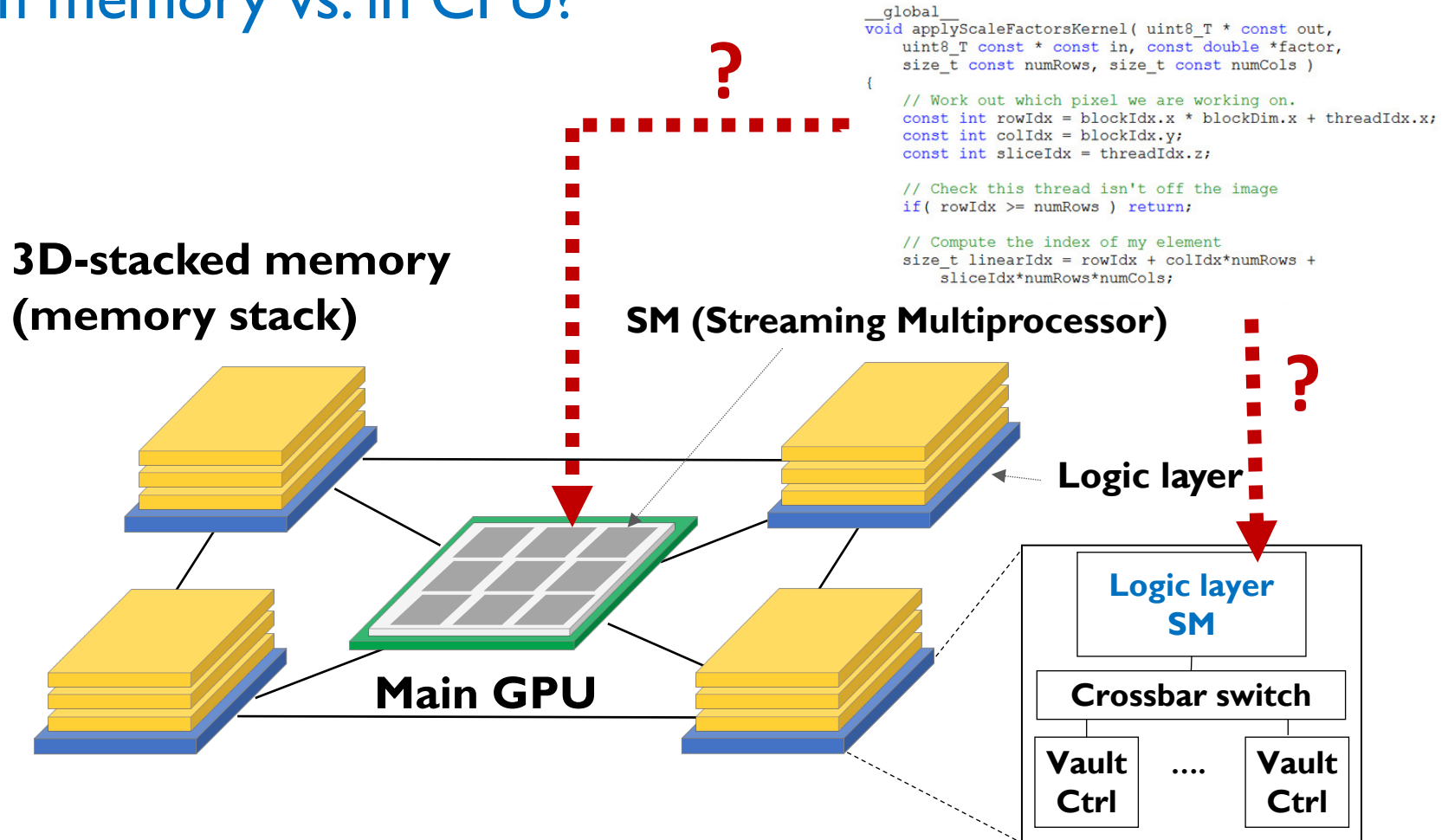
**"Processing-in-Memory: A Workload-Driven Perspective"**

*Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.*

[Preliminary arXiv version]

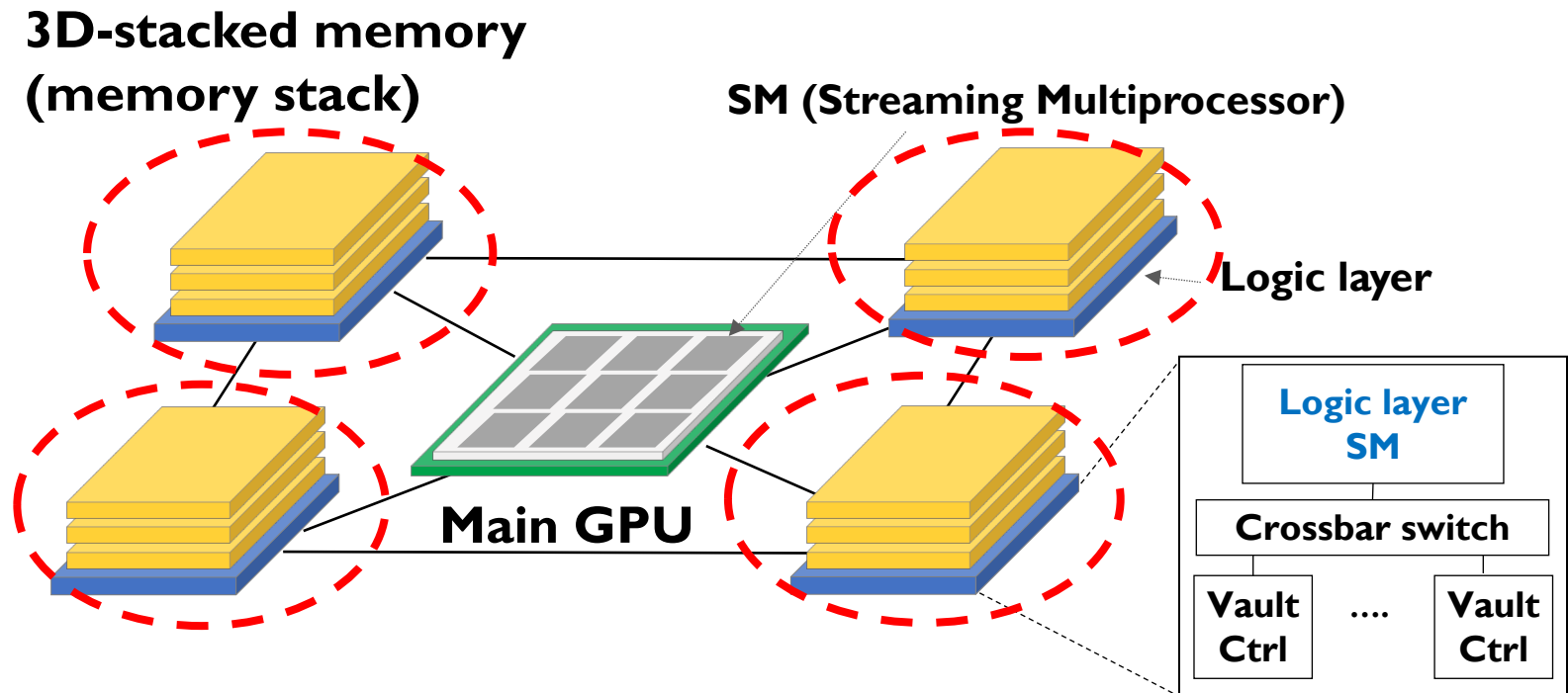
# Key Challenge 1: Code Mapping

- **Challenge 1: Which operations should be executed in memory vs. in CPU?**



# Key Challenge 2: Data Mapping

- **Challenge 2:** How should data be mapped to different 3D memory stacks?



# How to Do the Code and Data Mapping?

---

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**  
*Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea, June 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]

## Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh<sup>‡</sup> Eiman Ebrahimi<sup>†</sup> Gwangsun Kim\* Niladrish Chatterjee<sup>†</sup> Mike O'Connor<sup>†</sup>  
Nandita Vijaykumar<sup>‡</sup> Onur Mutlu<sup>§‡</sup> Stephen W. Keckler<sup>†</sup>

<sup>‡</sup>Carnegie Mellon University <sup>†</sup>NVIDIA <sup>\*</sup>KAIST <sup>§</sup>ETH Zürich



# How to Schedule Code? (I)

---

- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das, **"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"**  
*Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Haifa, Israel, September 2016.

## Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik<sup>1</sup>    Xulong Tang<sup>1</sup>    Adwait Jog<sup>2</sup>    Onur Kayiran<sup>3</sup>  
Asit K. Mishra<sup>4</sup>    Mahmut T. Kandemir<sup>1</sup>    Onur Mutlu<sup>5,6</sup>    Chita R. Das<sup>1</sup>  
<sup>1</sup>Pennsylvania State University    <sup>2</sup>College of William and Mary  
<sup>3</sup>Advanced Micro Devices, Inc.    <sup>4</sup>Intel Labs    <sup>5</sup>ETH Zürich    <sup>6</sup>Carnegie Mellon University

# How to Schedule Code? (II)

---

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**

*Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea, June 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Lightning Session Slides \(pptx\)](#) ([pdf](#))]

## Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi\*, Khubaib<sup>†</sup>, Eiman Ebrahimi<sup>‡</sup>, Onur Mutlu<sup>§</sup>, Yale N. Patt\*

\*The University of Texas at Austin    <sup>†</sup>Apple    <sup>‡</sup>NVIDIA    <sup>§</sup>ETH Zürich & Carnegie Mellon University

# How to Schedule Code? (III)

---

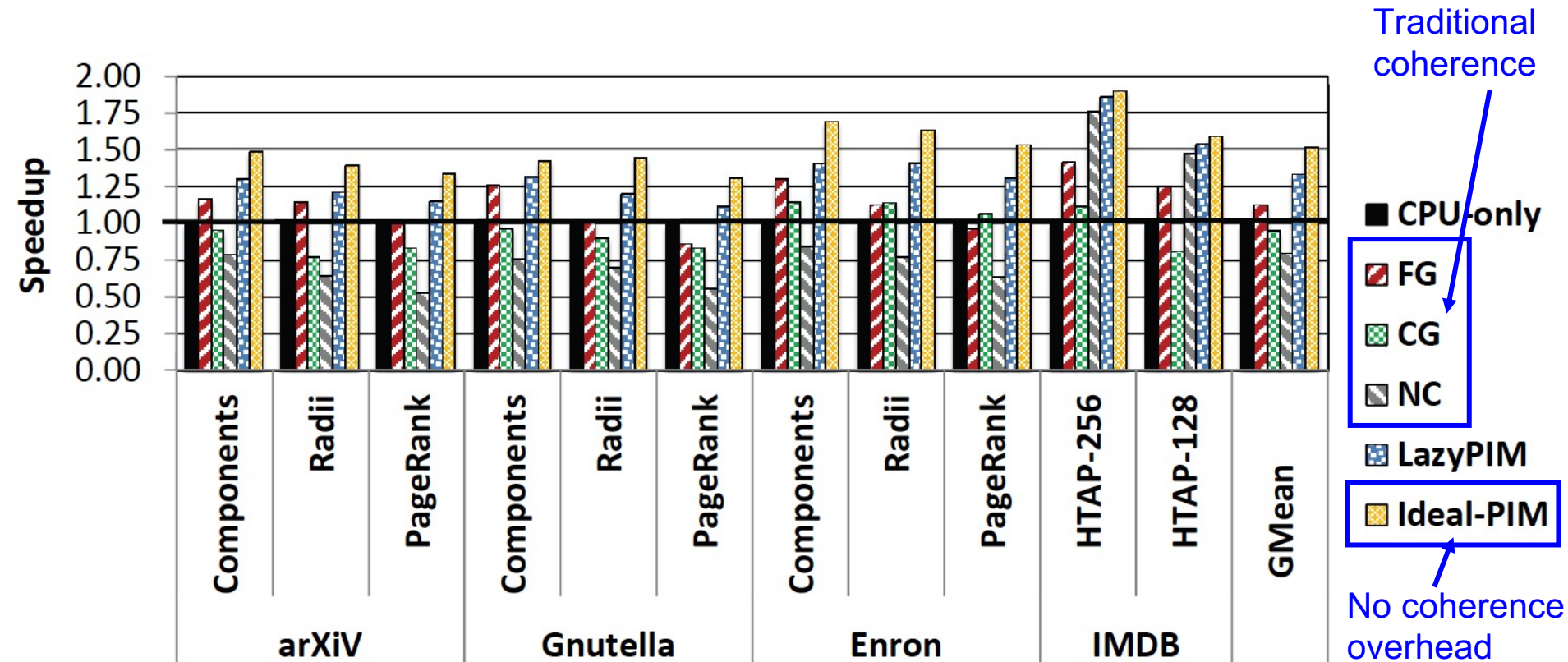
- Milad Hashemi, Onur Mutlu, and Yale N. Patt,  
**"Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads"**  
*Proceedings of the 49th International Symposium on Microarchitecture (MICRO)*, Taipei, Taiwan, October 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pdf\)](#)] [[Poster \(pptx\)](#)] [[pdf](#)]

## Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

Milad Hashemi\*, Onur Mutlu<sup>§</sup>, Yale N. Patt\*

\**The University of Texas at Austin*    <sup>§</sup>*ETH Zürich*

# Challenge: Coherence for Hybrid CPU-PIM Apps



# How to Maintain Coherence? (I)

---

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,  
**"LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory"**  
***IEEE Computer Architecture Letters* (**CAL**), June 2016.**

## LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

Amirali Boroumand<sup>†</sup>, Saugata Ghose<sup>†</sup>, Minesh Patel<sup>†</sup>, Hasan Hassan<sup>†§</sup>, Brandon Lucia<sup>†</sup>,  
Kevin Hsieh<sup>†</sup>, Krishna T. Malladi<sup>\*</sup>, Hongzhong Zheng<sup>\*</sup>, and Onur Mutlu<sup>††</sup>

<sup>†</sup>Carnegie Mellon University   <sup>\*</sup>Samsung Semiconductor, Inc.   <sup>§</sup>TOBB ETÜ   <sup>‡</sup>ETH Zürich



# How to Maintain Coherence? (II)

---

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,  
**"CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators"**  
*Proceedings of the 46th International Symposium on Computer Architecture (ISCA)*, Phoenix, AZ, USA, June 2019.

## CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand<sup>†</sup>

Saugata Ghose<sup>†</sup>

Minesh Patel<sup>★</sup>

Hasan Hassan<sup>★</sup>

Brandon Lucia<sup>†</sup>

Rachata Ausavarungnirun<sup>†‡</sup>

Kevin Hsieh<sup>†</sup>

Nastaran Hajinazar<sup>◇†</sup>

Krishna T. Malladi<sup>§</sup>

Hongzhong Zheng<sup>§</sup>

Onur Mutlu<sup>★†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>★</sup>ETH Zürich

<sup>‡</sup>KMUTNB

<sup>◇</sup>Simon Fraser University

<sup>§</sup>Samsung Semiconductor, Inc.

# CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

**Amirali Boroumand**

Saugata Ghose, Minesh Patel, Hasan Hassan,  
Brandon Lucia, Rachata Ausavarungnirun, Kevin Hsieh,  
Nastaran Hajinazar, Krishna Malladi, Hongzhong Zheng,  
Onur Mutlu

**SAFARI**



**Carnegie Mellon**



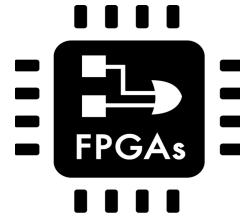
**ETH** zürich

# Specialized Accelerators

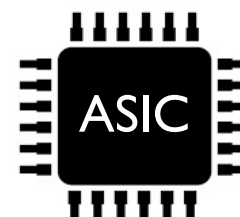
Specialized accelerators are now everywhere!



**GPU**

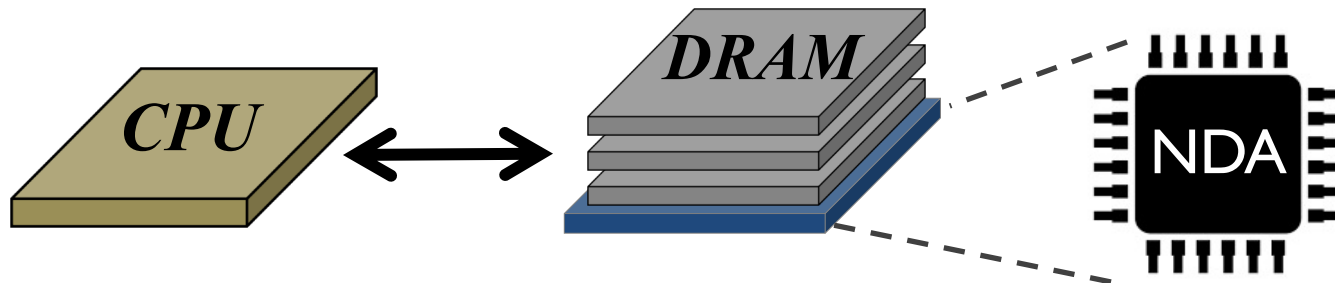


**FPGA**



**ASIC**

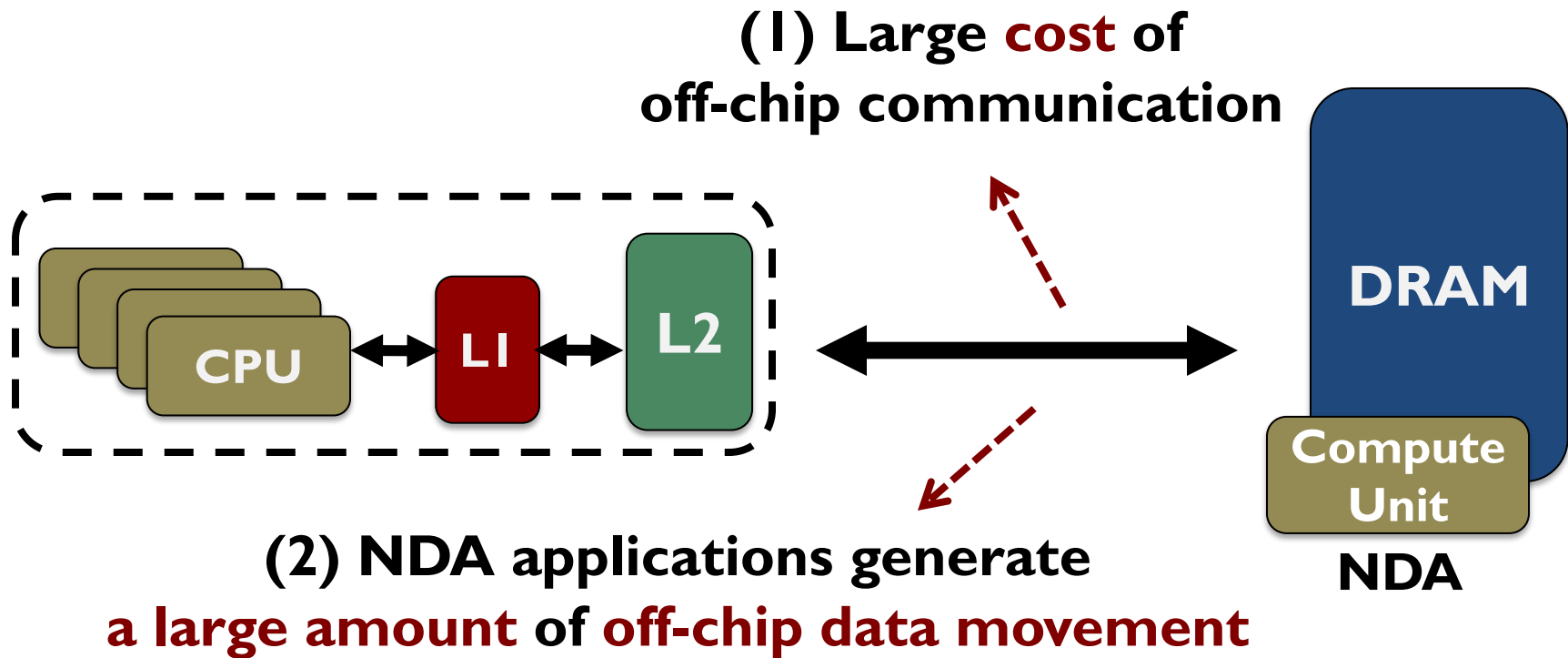
Recent advancement in 3D-stacked technology enabled **Near-Data Accelerators (NDA)**





# Coherence For NDAs

## Challenge: Coherence between NDAs and CPUs



It is **impractical** to use traditional coherence protocols

# Existing Coherence Mechanisms

We extensively study existing **NDA coherence mechanisms** and make **three key observations**:

1

These mechanisms **eliminate** a significant portion of **NDA's benefits**

2

The **majority of off-chip coherence traffic** generated by these mechanisms is **unnecessary**

3

Much of the **off-chip traffic** can be eliminated if the coherence mechanism has **insight** into the **memory accesses**

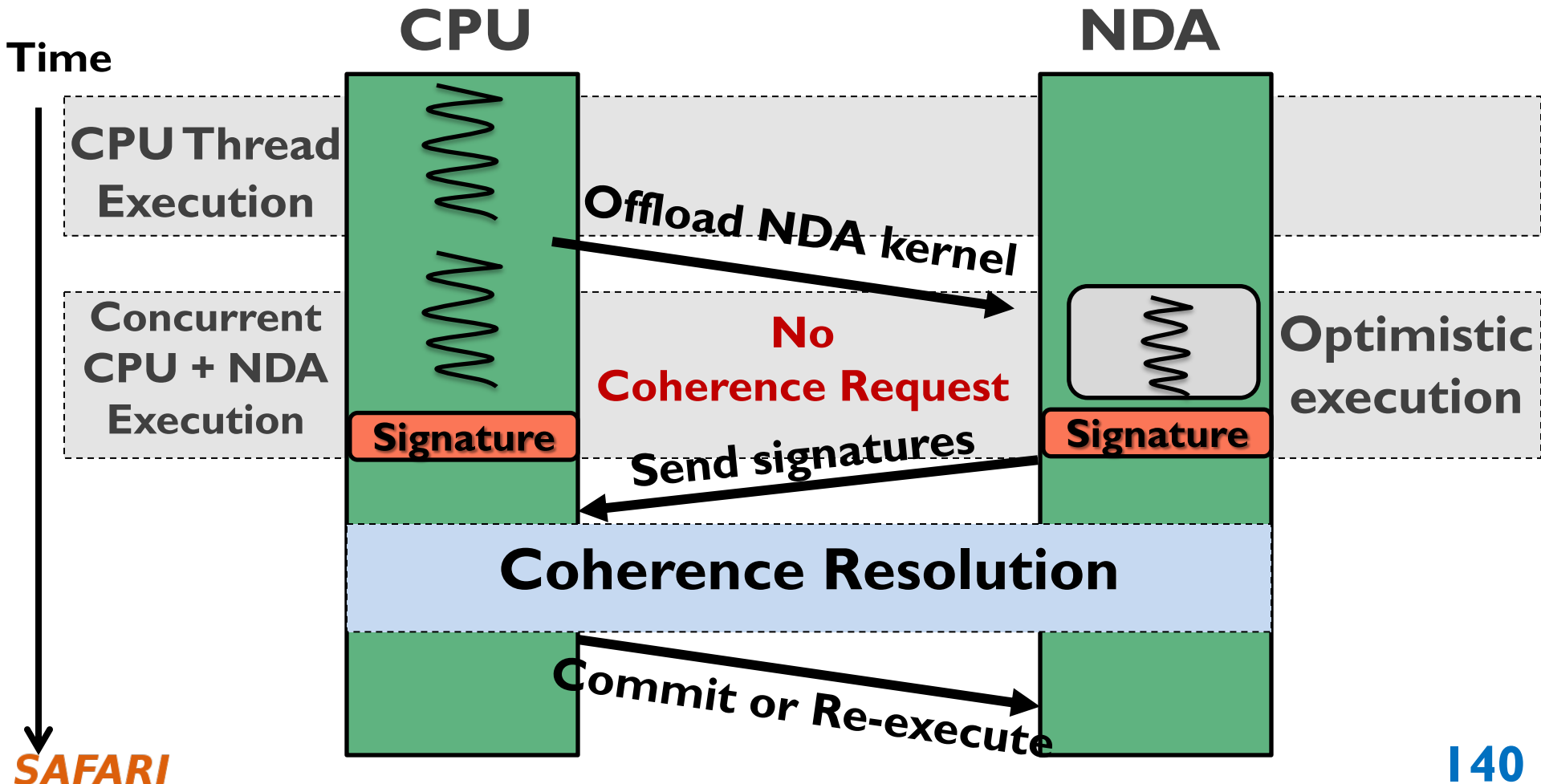
# An Optimistic Approach

We find that **an optimistic approach** to coherence can address the **challenges** related to NDA coherence

- 1 Gain insights **before** any coherence checks happens
- 2 Perform **only the necessary** coherence requests

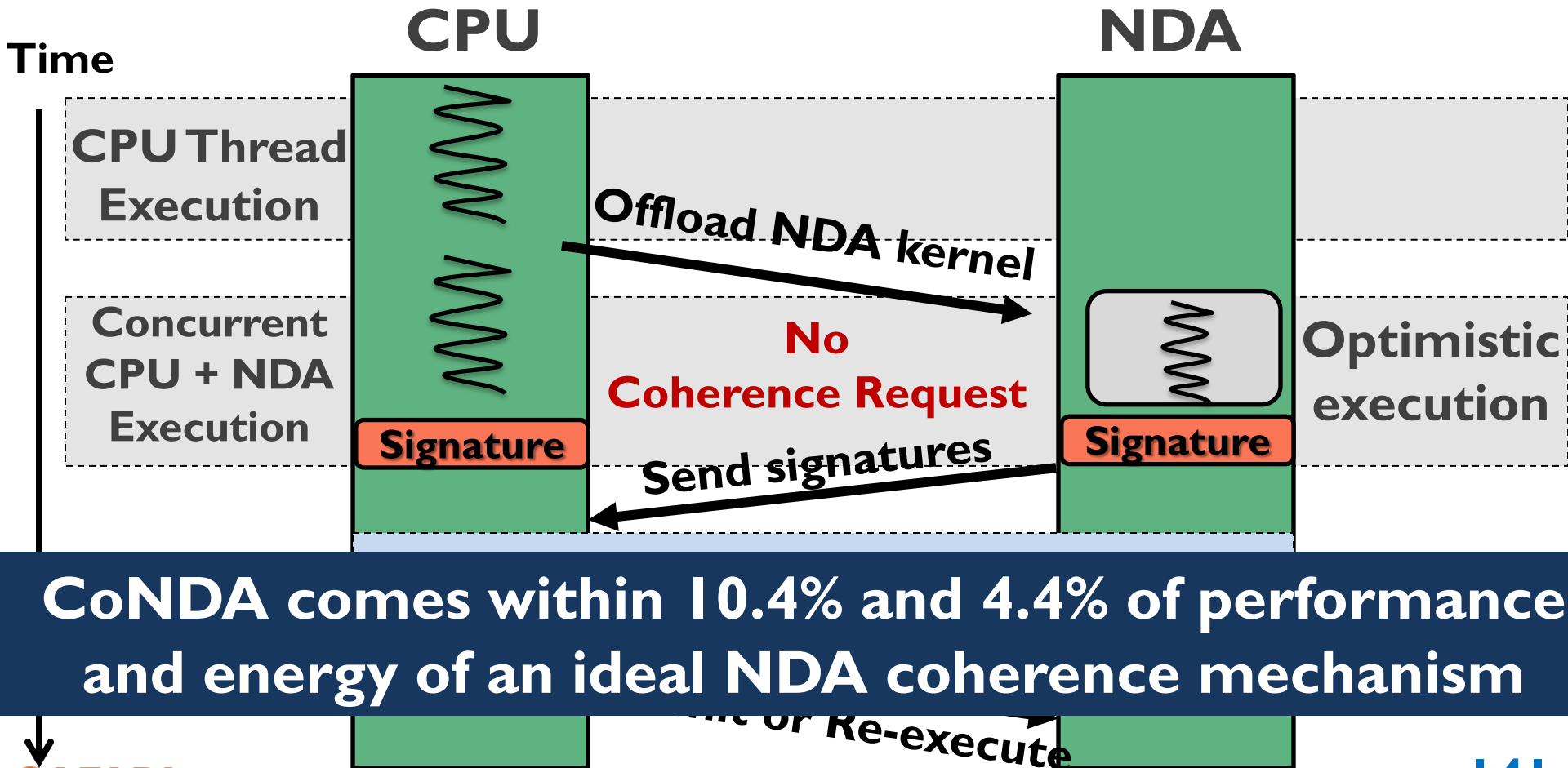
# CoNDA

We propose **CoNDA**, a mechanism that uses **optimistic NDA execution** to avoid **unnecessary coherence traffic**



# CoNDA

We propose **CoNDA**, a mechanism that uses **optimistic NDA execution** to avoid **unnecessary coherence traffic**



# CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

**Amirali Boroumand**

Saugata Ghose, Minesh Patel, Hasan Hassan,  
Brandon Lucia, Rachata Ausavarungnirun, Kevin Hsieh,  
Nastaran Hajinazar, Krishna Malladi, Hongzhong Zheng,  
Onur Mutlu

**SAFARI**



**Carnegie Mellon**



**ETH** zürich

# How to Maintain Coherence? (II)

---

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,  
**"CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators"**  
*Proceedings of the 46th International Symposium on Computer Architecture (ISCA)*, Phoenix, AZ, USA, June 2019.

## CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand<sup>†</sup>

Saugata Ghose<sup>†</sup>

Minesh Patel<sup>★</sup>

Hasan Hassan<sup>★</sup>

Brandon Lucia<sup>†</sup>

Rachata Ausavarungnirun<sup>†‡</sup>

Kevin Hsieh<sup>†</sup>

Nastaran Hajinazar<sup>◇†</sup>

Krishna T. Malladi<sup>§</sup>

Hongzhong Zheng<sup>§</sup>

Onur Mutlu<sup>★†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>★</sup>ETH Zürich

<sup>‡</sup>KMUTNB

<sup>◇</sup>Simon Fraser University

<sup>§</sup>Samsung Semiconductor, Inc.

# How to Support Virtual Memory?

---

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,  
**"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"**  
*Proceedings of the 34th IEEE International Conference on Computer Design (ICCD)*, Phoenix, AZ, USA, October 2016.

## Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh<sup>†</sup>   Samira Khan<sup>‡</sup>   Nandita Vijaykumar<sup>†</sup>  
Kevin K. Chang<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Saugata Ghose<sup>†</sup>   Onur Mutlu<sup>§†</sup>  
<sup>†</sup>*Carnegie Mellon University*   <sup>‡</sup>*University of Virginia*   <sup>§</sup>*ETH Zürich*



# How to Design Data Structures for PIM?

---

- Zhiyu Liu, Irina Calciu, Maurice Herlihy, and Onur Mutlu,  
**"Concurrent Data Structures for Near-Memory Computing"**  
*Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (**SPAA**), Washington, DC, USA, July 2017.*  
[[Slides \(pptx\)](#) ([pdf](#))]

## Concurrent Data Structures for Near-Memory Computing

Zhiyu Liu

Computer Science Department  
Brown University  
zhiyu.liu@brown.edu

Irina Calciu

VMware Research Group  
icalciu@vmware.com

Maurice Herlihy

Computer Science Department  
Brown University  
mph@cs.brown.edu

Onur Mutlu

Computer Science Department  
ETH Zürich  
onur.mutlu@inf.ethz.ch

# Simulation Infrastructures for PIM

---

- **Ramulator** extended for PIM
  - Flexible and extensible DRAM simulator
  - Can model many different memory standards and proposals
  - Kim+, “**Ramulator: A Flexible and Extensible DRAM Simulator**”, IEEE CAL 2015.
  - <https://github.com/CMU-SAFARI/ramulator-pim>
  - <https://github.com/CMU-SAFARI/ramulator>
  - [[Source Code for Ramulator-PIM](#)]

## Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim<sup>1</sup>    Weikun Yang<sup>1,2</sup>    Onur Mutlu<sup>1</sup>  
<sup>1</sup>Carnegie Mellon University    <sup>2</sup>Peking University

# Performance & Energy Models for PIM

---

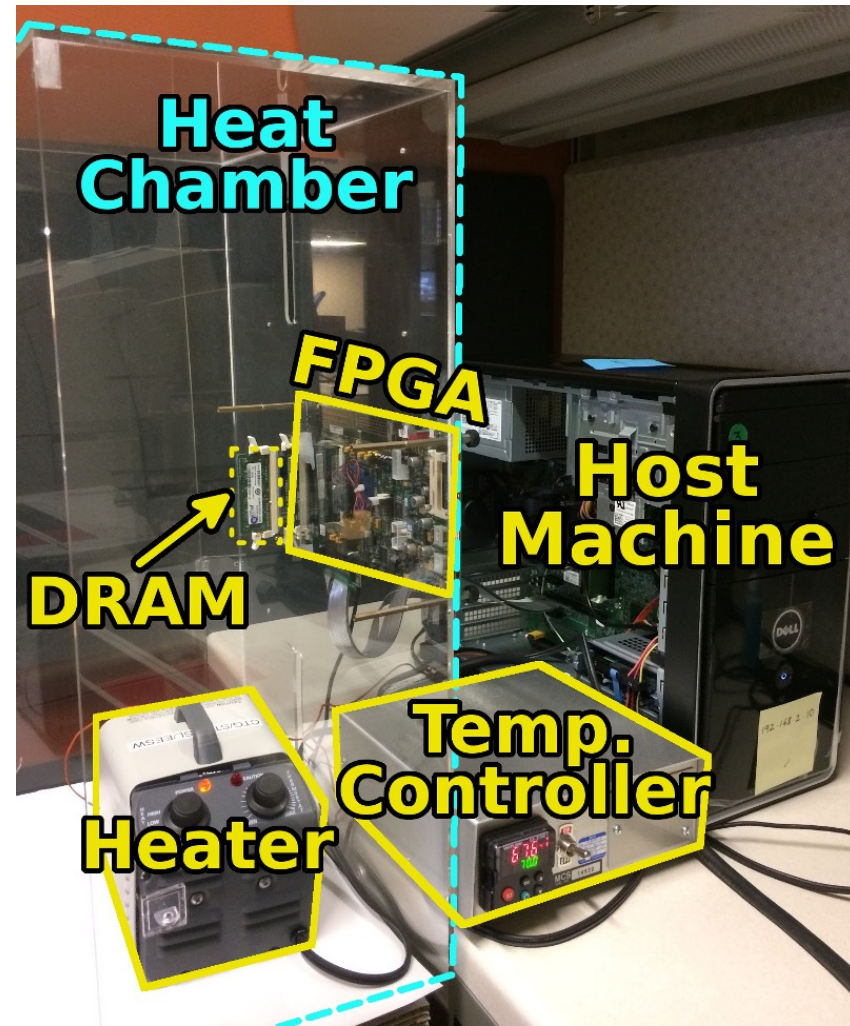
- Gagandeep Singh, Juan Gomez-Luna, Giovanni Mariani, Geraldo F. Oliveira, Stefano Corda, Sander Stujik, Onur Mutlu, and Henk Corporaal, **"NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning"**  
*Proceedings of the 56th Design Automation Conference (DAC)*, Las Vegas, NV, USA, June 2019.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Poster \(pptx\)](#)] [[pdf](#)]  
[[Source Code for Ramulator-PIM](#)]

## NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning

Gagandeep Singh <sup>a,c</sup>	Juan Gómez-Luna <sup>b</sup>	Giovanni Mariani <sup>c</sup>	Geraldo F. Oliveira <sup>b</sup>
Stefano Corda <sup>a,c</sup>	Sander Stuijk <sup>a</sup>	Onur Mutlu <sup>b</sup>	Henk Corporaal <sup>a</sup>
<sup>a</sup> Eindhoven University of Technology		<sup>b</sup> ETH Zürich	<sup>c</sup> IBM Research - Zurich

# An FPGA-based Test-bed for PIM?

- Hasan Hassan et al., **SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies** HPCA 2017.
- Flexible
- Easy to Use (C++ API)
- Open-source  
[github.com/CMU-SAFARI/SoftMC](https://github.com/CMU-SAFARI/SoftMC)



# Simulation Infrastructures for PIM (in SSDs)

---

- Arash Tavakkol, Juan Gomez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu,  
**"MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices"**  
*Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST)*, Oakland, CA, USA, February 2018.  
[Slides (pptx)] [pdf]  
[Source Code]

## MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices

Arash Tavakkol<sup>†</sup>, Juan Gómez-Luna<sup>†</sup>, Mohammad Sadrosadati<sup>†</sup>, Saugata Ghose<sup>‡</sup>, Onur Mutlu<sup>†‡</sup>  
<sup>†</sup>*ETH Zürich*                      <sup>‡</sup>*Carnegie Mellon University*



# New Applications and Use Cases for PIM

---

- Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu, **"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"** ***BMC Genomics***, 2018.  
*Proceedings of the 16th Asia Pacific Bioinformatics Conference (APBC)*, Yokohama, Japan, January 2018.  
[arxiv.org Version \(pdf\)](https://arxiv.org/abs/1801.00000)

## GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies

Jeremie S. Kim<sup>1,6\*</sup>, Damla Senol Cali<sup>1</sup>, Hongyi Xin<sup>2</sup>, Donghyuk Lee<sup>3</sup>, Saugata Ghose<sup>1</sup>, Mohammed Alser<sup>4</sup>, Hasan Hassan<sup>6</sup>, Oguz Ergin<sup>5</sup>, Can Alkan<sup>4\*</sup> and Onur Mutlu<sup>6,1\*</sup>

From The Sixteenth Asia Pacific Bioinformatics Conference 2018  
Yokohama, Japan. 15-17 January 2018

# Genome Read In-Memory (GRIM) Filter:

Fast Seed Location Filtering in DNA Read Mapping  
using Processing-in-Memory Technologies

**Jeremie Kim,**

Damla Senol, Hongyi Xin, Donghyuk Lee,  
Saugata Ghose, Mohammed Alser, Hasan Hassan,  
Oguz Ergin, Can Alkan, and Onur Mutlu

**Carnegie Mellon**



**ETH** zürich

# Executive Summary

---

- **Genome Read Mapping** is a very important problem and is the first step in many types of genomic analysis
  - Could lead to improved health care, medicine, quality of life
- Read mapping is an **approximate string matching** problem
  - Find the best fit of 100 character strings into a 3 billion character dictionary
  - **Alignment** is currently the best method for determining the similarity between two strings, but is **very expensive**
- We propose an in-memory processing algorithm **GRIM-Filter** for accelerating read mapping, by reducing the number of required alignments
- We implement GRIM-Filter using **in-memory processing** within **3D-stacked memory** and show up to **3.7x speedup**.



# Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

**Amirali Boroumand**

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun,  
Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela,  
Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

**SAFARI**

**Carnegie Mellon**

**Google**



SEOUL  
NATIONAL  
UNIVERSITY

**ETH** zürich

# Accelerating Climate Modeling

---

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,  
**"NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"**  
*Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, September 2020.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]  
[[Talk Video](#) (23 minutes)]  
***Nominated for the Stamatis Vassiliadis Memorial Award.***

## NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh<sup>a,b,c</sup>    Dionysios Diamantopoulos<sup>c</sup>    Christoph Hagleitner<sup>c</sup>    Juan Gómez-Luna<sup>b</sup>  
Sander Stuijk<sup>a</sup>    Onur Mutlu<sup>b</sup>    Henk Corporaal<sup>a</sup>  
<sup>a</sup>Eindhoven University of Technology    <sup>b</sup>ETH Zürich    <sup>c</sup>IBM Research Europe, Zurich

# Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zülal Bingöl, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu, **"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"**  
*Proceedings of the 53rd International Symposium on Microarchitecture (MICRO)*, Virtual, October 2020.  
[[Lighting Talk Video](#) (1.5 minutes)]  
[[Lightning Talk Slides \(pptx\)](#) ([pdf](#))]  
[[Talk Video](#) (18 minutes)]  
[[Slides \(pptx\)](#) ([pdf](#))]

## GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali<sup>†⌘</sup> Gurpreet S. Kalsi<sup>⌘</sup> Zülal Bingöl<sup>▽</sup> Can Firtina<sup>◇</sup> Lavanya Subramanian<sup>‡</sup> Jeremie S. Kim<sup>◇†</sup>  
Rachata Ausavarungnirun<sup>○</sup> Mohammed Alser<sup>◇</sup> Juan Gomez-Luna<sup>◇</sup> Amirali Boroumand<sup>†</sup> Anant Nori<sup>⌘</sup>  
Allison Scibisz<sup>†</sup> Sreenivas Subramoney<sup>⌘</sup> Can Alkan<sup>▽</sup> Saugata Ghose<sup>\*†</sup> Onur Mutlu<sup>◇†▽</sup>  
<sup>†</sup>Carnegie Mellon University   <sup>⌘</sup>Processor Architecture Research Lab, Intel Labs   <sup>▽</sup>Bilkent University   <sup>◇</sup>ETH Zürich  
<sup>‡</sup>Facebook   <sup>○</sup>King Mongkut's University of Technology North Bangkok   <sup>\*</sup>University of Illinois at Urbana-Champaign

# Accelerating Time Series Analysis

---

- Ivan Fernandez, Ricardo Quisiant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,  
**"NATSA: A Near-Data Processing Accelerator for Time Series Analysis"**  
*Proceedings of the 38th IEEE International Conference on Computer Design (ICCD)*, Virtual, October 2020.

## NATSA: A Near-Data Processing Accelerator for Time Series Analysis

Ivan Fernandez <sup>§</sup>	Ricardo Quisiant <sup>§</sup>	Christina Giannoula <sup>†</sup>	Mohammed Alser <sup>‡</sup>
Juan Gómez-Luna <sup>‡</sup>	Eladio Gutiérrez <sup>§</sup>	Oscar Plata <sup>§</sup>	Onur Mutlu <sup>‡</sup>
<sup>§</sup> <i>University of Malaga</i>	<sup>†</sup> <i>National Technical University of Athens</i>	<sup>‡</sup> <i>ETH Zürich</i>	

# PIM Review and Open Problems

---

## A Modern Primer on Processing in Memory

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b,c</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>d</sup>

*SAFARI Research Group*

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*University of Illinois at Urbana-Champaign*

<sup>d</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,

**"A Modern Primer on Processing in Memory"**

*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*



# PIM Review and Open Problems (II)

---

## Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>b,c</sup>

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,  
**"Processing Data Where It Makes Sense: Enabling In-Memory  
Computation"**

*Invited paper in Microprocessors and Microsystems (**MICPRO**), June 2019.  
[arXiv version]*

# PIM Review and Open Problems (III)

---

## A Workload and Programming Ease Driven Perspective of Processing-in-Memory

Saugata Ghose<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Jeremie S. Kim<sup>†§</sup>   Juan Gómez-Luna<sup>§</sup>   Onur Mutlu<sup>§†</sup>

<sup>†</sup>*Carnegie Mellon University*

<sup>§</sup>*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

**"Processing-in-Memory: A Workload-Driven Perspective"**

*Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.*

[Preliminary arXiv version]

# Fundamentally Energy-Efficient (Data-Centric) Computing Architectures



# Fundamentally High-Performance (Data-Centric) Computing Architectures

# Computing Architectures with Minimal Data Movement

# One Important Takeaway

---

Main Memory Needs  
Intelligent Controllers

# Enabling the Paradigm Shift

# Recall: Computer Architecture Today

---

- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)
- You can invent new paradigms for computation, communication, and storage
- Recommended book: Thomas Kuhn, “[The Structure of Scientific Revolutions](#)” (1962)
  - Pre-paradigm science: no clear consensus in the field
  - Normal science: dominant theory used to explain/improve things (business as usual); exceptions considered anomalies
  - Revolutionary science: underlying assumptions re-examined

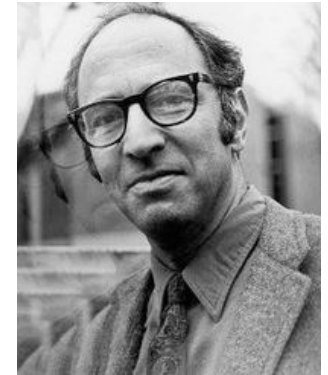
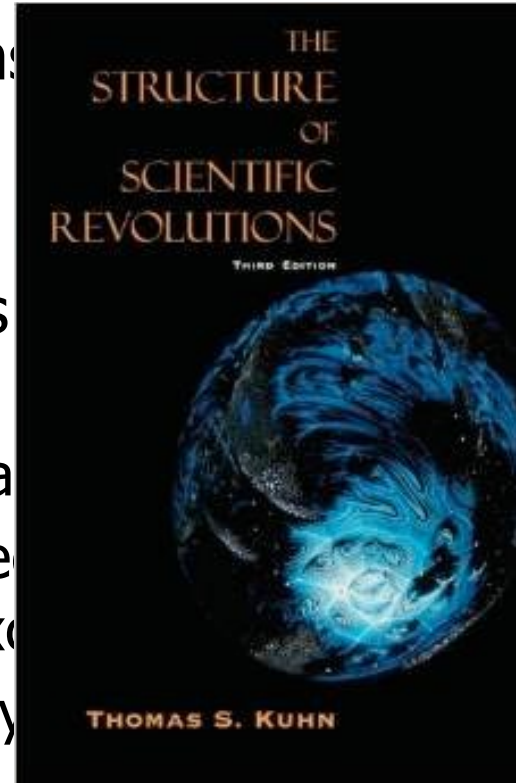
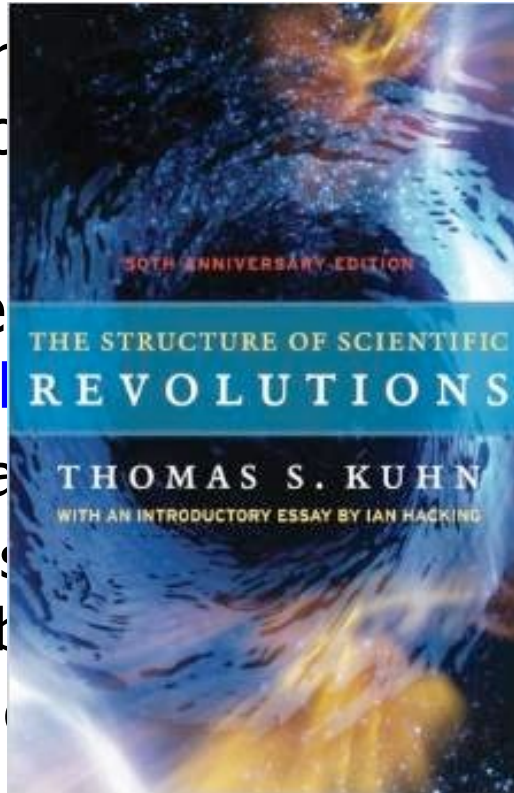
# Recall: Computer Architecture Today

- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)

- You can improve communication

- Recommend **Scientific Revolutions**

- Pre-prepare
- Normal scientific things (the existing theory)
- Revolutionary

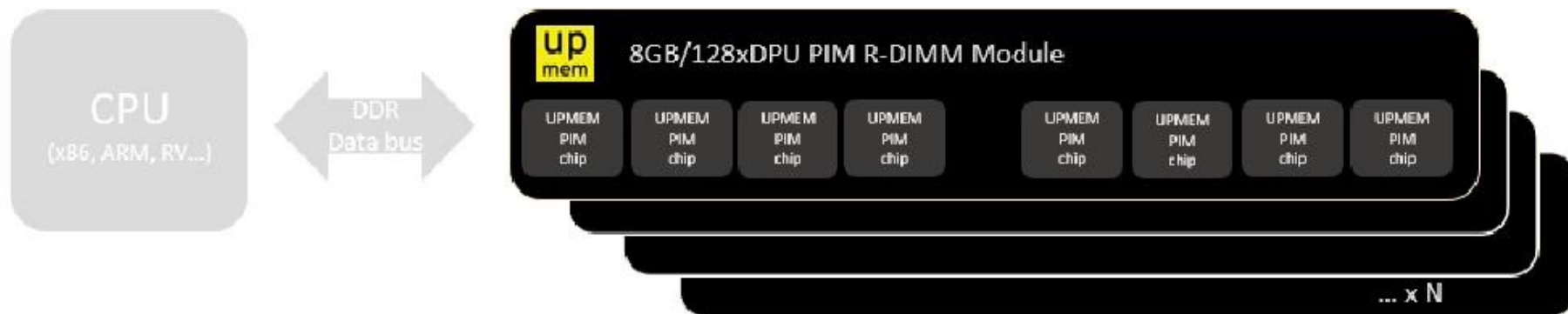
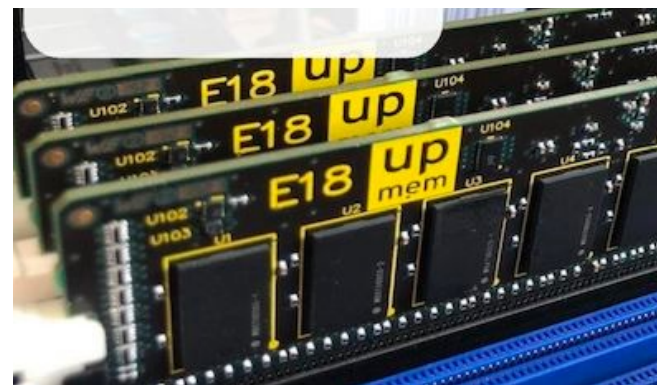


ure of

eld  
improve  
anomalies  
examined

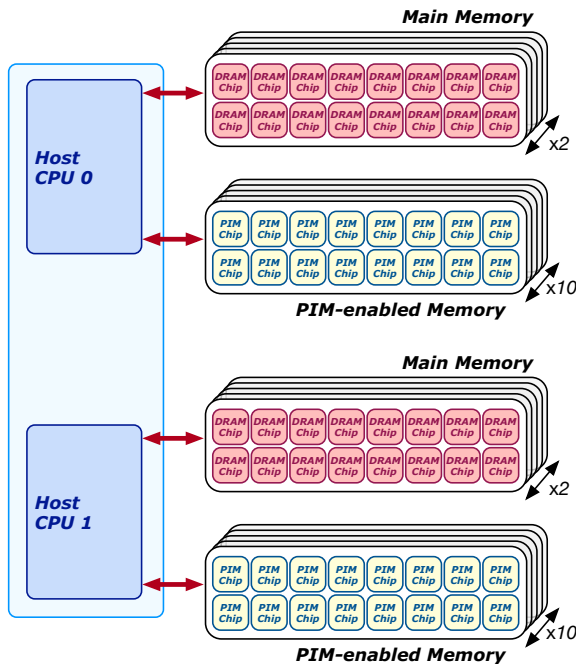
# UPMEM Processing-in-DRAM Engine (2019)

- **Processing in DRAM Engine**
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.
- Replaces **standard DIMMs**
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth





# 2,560-DPU Processing-in-Memory System



## Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland  
 IZZAT EL HAJJ, American University of Beirut, Lebanon  
 IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain  
 CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece  
 GERALDO F. OLIVEIRA, ETH Zürich, Switzerland  
 ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this *data movement bottleneck* requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as *processing-in-memory (PIM)*.

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called *DRAM Processing Units (DPUs)*, integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present *PrIM (Processing-In-Memory benchmarks)*, a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,560 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

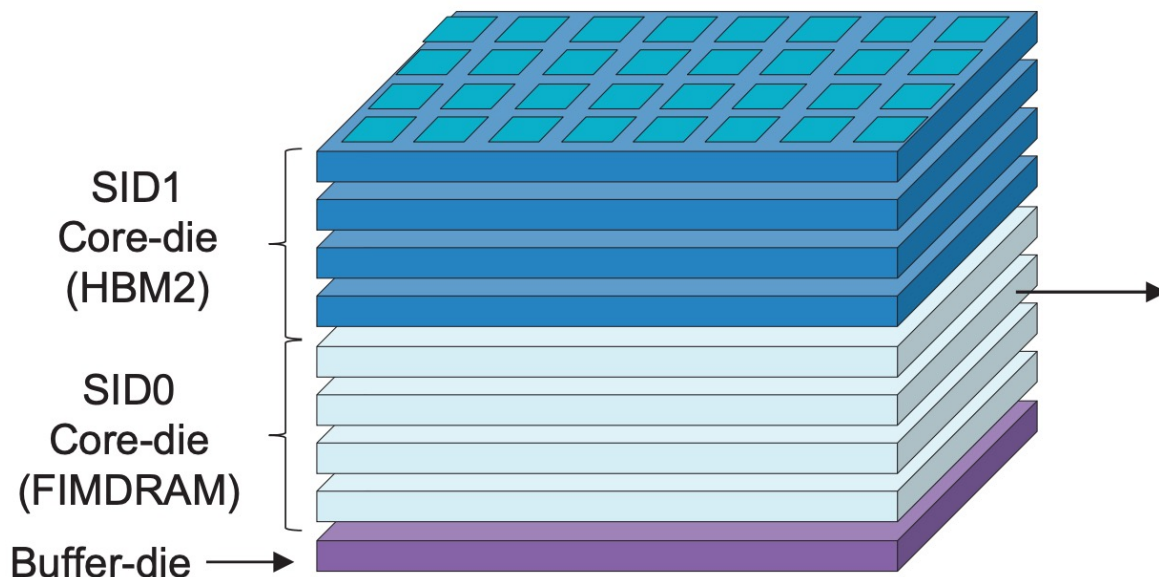


<https://arxiv.org/pdf/2105.03814.pdf>



# Samsung Function-in-Memory DRAM (2021)

## ■ FIMDRAM based on HBM2



[3D Chip Structure of HBM with FIMDRAM]

### Chip Specification

128DQ / 8CH / 16 banks / BL4

32 PCU blocks (1 FIM block/2 banks)

1.2 TFLOPS (4H)

**FP16 ADD /  
Multiply (MUL) /  
Multiply-Accumulate (MAC) /  
Multiply-and- Add (MAD)**

ISSCC 2021 / SESSION 25 / DRAM / 25.4

**25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications**

Young-Cheon Kwon<sup>1</sup>, Suk Han Lee<sup>1</sup>, Jaehoon Lee<sup>1</sup>, Sang-Hyuk Kwon<sup>1</sup>, Je Min Ryu<sup>1</sup>, Jong-Pil Son<sup>1</sup>, Seongil O<sup>1</sup>, Hak-Soo Yu<sup>1</sup>, Haesuk Lee<sup>1</sup>, Soo Young Kim<sup>1</sup>, Youngmin Cho<sup>1</sup>, Jin Guk Kim<sup>1</sup>, Jongyoon Choi<sup>1</sup>, Hyun-Sung Shin<sup>1</sup>, Jin Kim<sup>1</sup>, BengSeng Phuah<sup>1</sup>, HyoungMin Kim<sup>1</sup>, Myeong Jun Song<sup>1</sup>, Ahn Choi<sup>1</sup>, Daeho Kim<sup>1</sup>, SooYoung Kim<sup>1</sup>, Eun-Bong Kim<sup>1</sup>, David Wang<sup>2</sup>, Shinhaeng Kang<sup>1</sup>, Yuhwan Ro<sup>3</sup>, Seungwoo Seo<sup>3</sup>, JoonHo Song<sup>3</sup>, Jaeyoun Youn<sup>1</sup>, Kyomin Sohn<sup>1</sup>, Nam Sung Kim<sup>1</sup>

<sup>1</sup>Samsung Electronics, Hwaseong, Korea

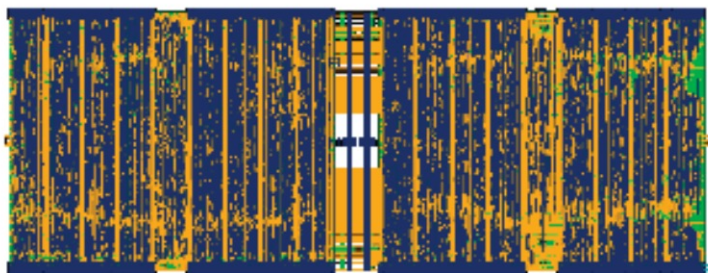
<sup>2</sup>Samsung Electronics, San Jose, CA

<sup>3</sup>Samsung Electronics, Suwon, Korea

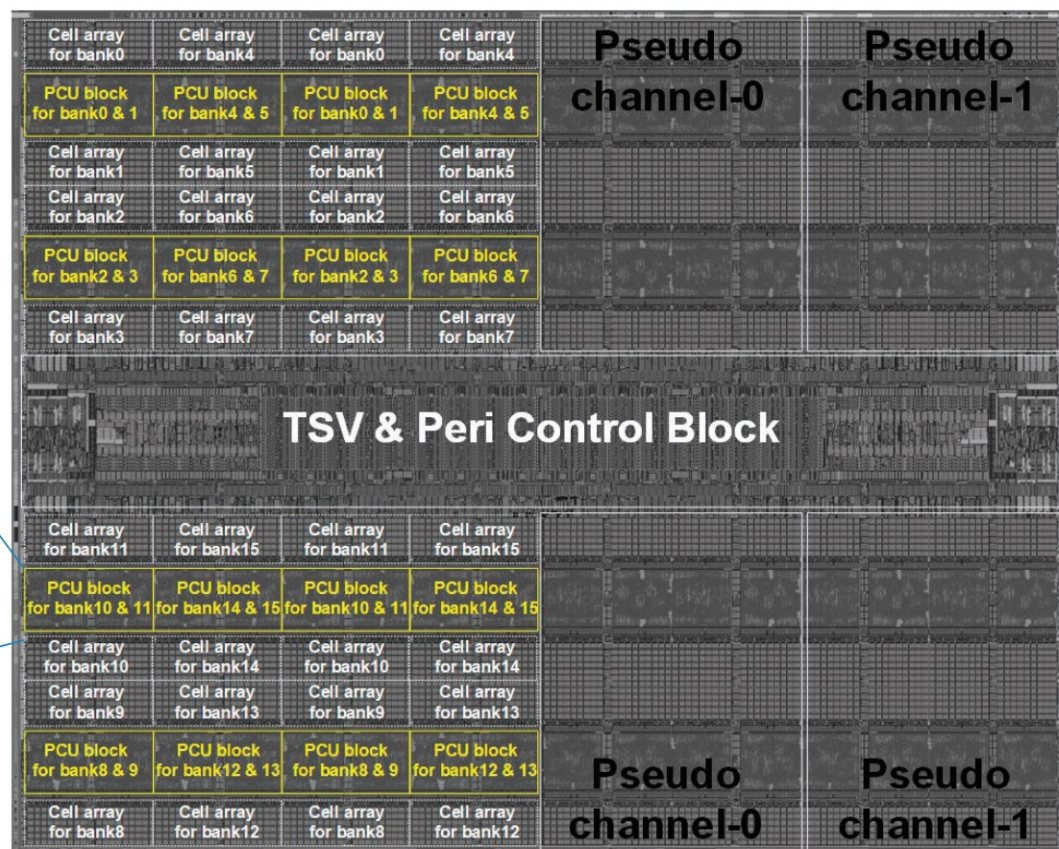
# Samsung Function-in-Memory DRAM (2021)

## Chip Implementation

- Mixed design methodology to implement FIMDRAM
  - Full-custom + Digital RTL



[Digital RTL design for PCU block]



ISSCC 2021 / SESSION 25 / DRAM / 25.4

25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Cheon Kwon<sup>1</sup>, Suk Han Lee<sup>1</sup>, Jaehoon Lee<sup>1</sup>, Sang-Hyuk Kwon<sup>1</sup>, Je Min Ryu<sup>1</sup>, Jong-Pil Son<sup>1</sup>, Seongil O<sup>1</sup>, Hak-Soo Yu<sup>1</sup>, Haesuk Lee<sup>1</sup>, Soo Young Kim<sup>1</sup>, Youngmin Cho<sup>1</sup>, Jin Guk Kim<sup>1</sup>, Jongyeon Choi<sup>1</sup>, Hyun-Sung Shim<sup>1</sup>, Jin Kim<sup>1</sup>, BengSeng Phuah<sup>1</sup>, HyounMin Kim<sup>1</sup>, Myeong Jun Song<sup>1</sup>, Ahn Chai<sup>1</sup>, Daeho Kim<sup>1</sup>, SooYoung Kim<sup>1</sup>, Eun-Bong Kim<sup>1</sup>, David Wang<sup>2</sup>, Shintae Kang<sup>3</sup>, Yulwan Ro<sup>3</sup>, Seungwoo Seo<sup>3</sup>, JoonHo Song<sup>3</sup>, Jaeyoun Yoon<sup>1</sup>, Kyomin Sohn<sup>1</sup>, Nam Sung Kim<sup>1</sup>

<sup>1</sup>Samsung Electronics, Hwaseong, Korea

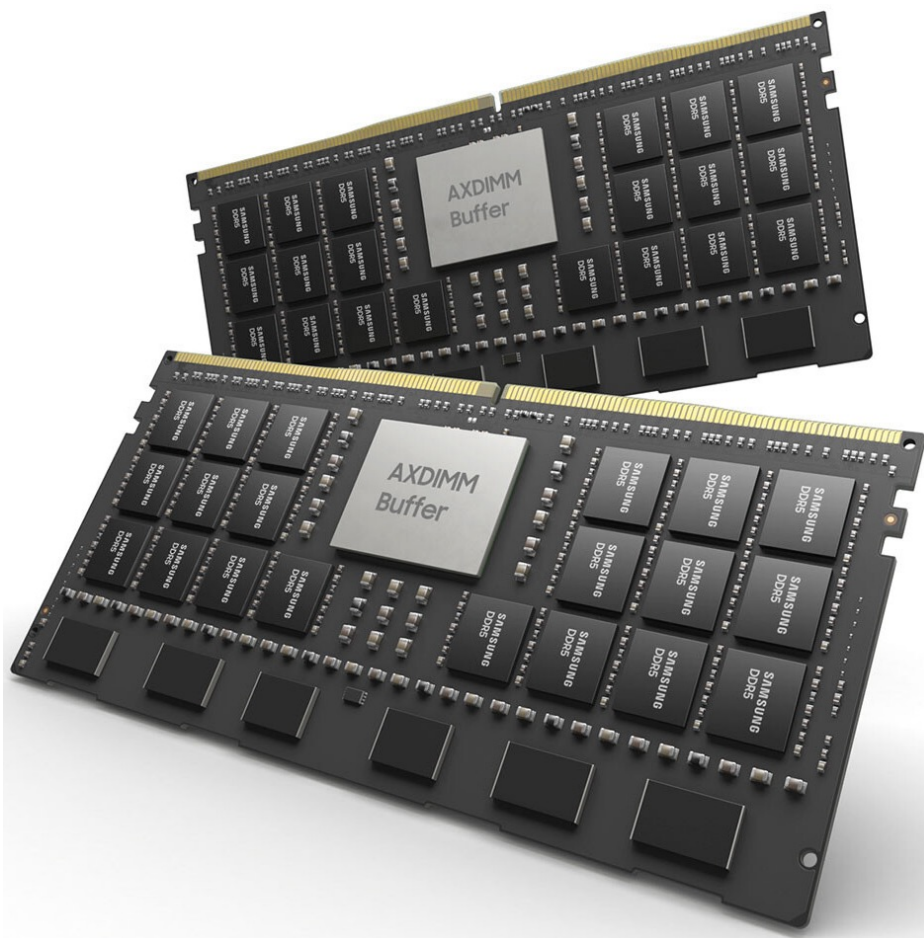
<sup>2</sup>Samsung Electronics, San Jose, CA

<sup>3</sup>Samsung Electronics, Suwon, Korea

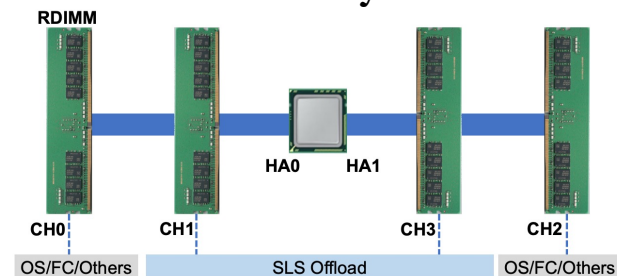


# Samsung AxDIMM (2021)

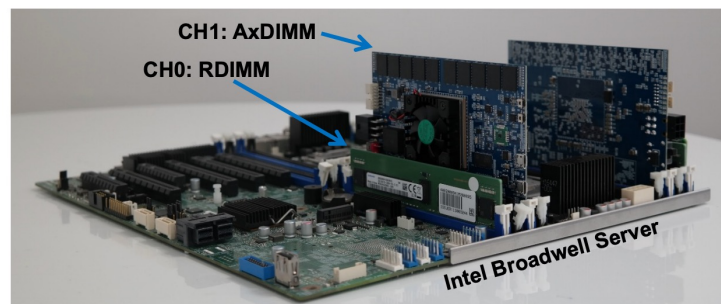
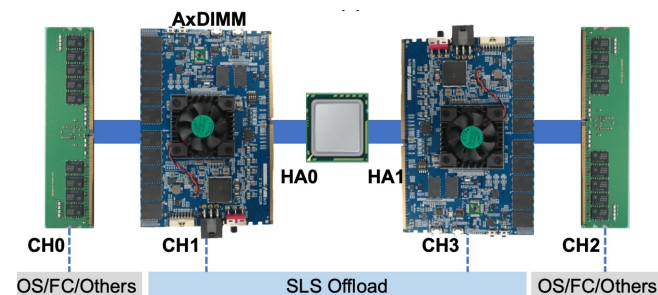
- DDR5-PIM
  - DLRM recommendation system



Baseline System



AxDIMM System



# Sub-Agenda: In-Memory Computation

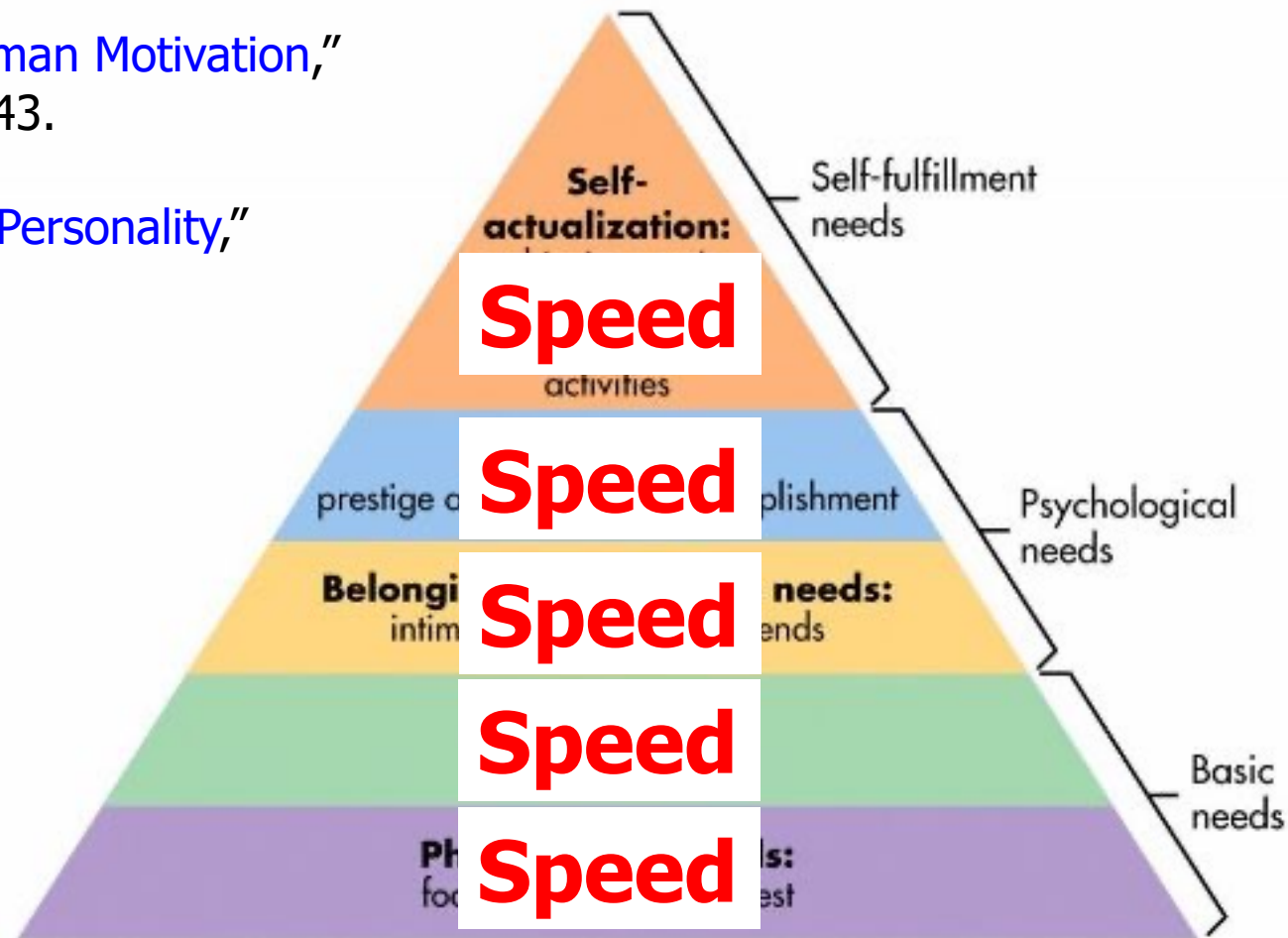
---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Processing using Memory
  - Processing near Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

# Maslow's Hierarchy of Needs, A Third Time

Maslow, "A Theory of Human Motivation,"  
Psychological Review, 1943.

Maslow, "Motivation and Personality,"  
Book, 1954-1970.



# Fundamentally High-Performance (Data-Centric) Computing Architectures

# Fundamentally Energy-Efficient (Data-Centric) Computing Architectures

# Fundamentally Low-Latency (Data-Centric) Computing Architectures



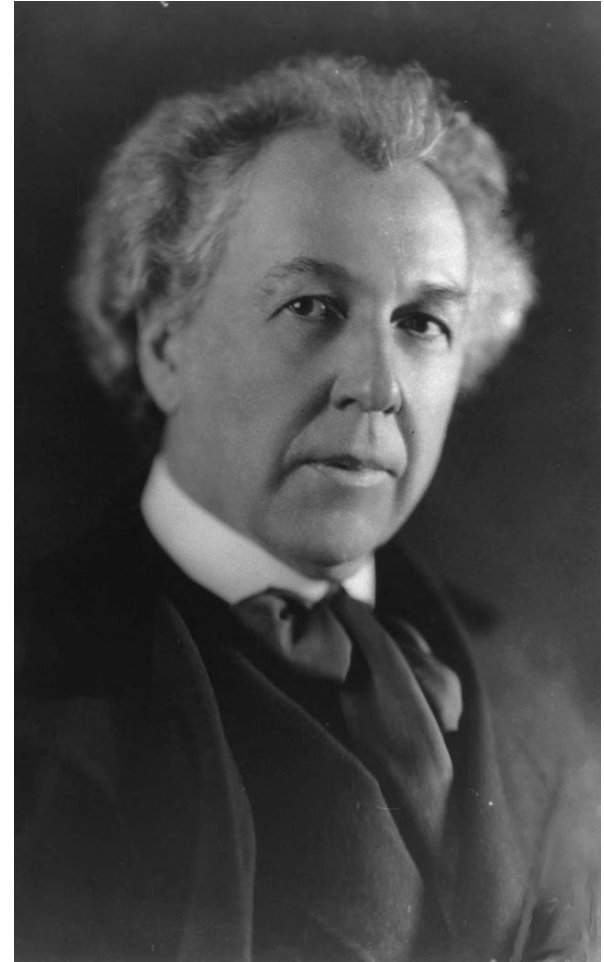
# Computing Architectures with Minimal Data Movement

# PIM: Concluding Remarks

# A Quote from A Famous Architect

---

- “architecture [...] based upon **principle**, and not upon **precedent**”



# Precedent-Based Design?

---

- “architecture [...] based upon **principle**, and not upon **precedent**”





# Principled Design

---

- “architecture [...] based upon **principle**, and not upon **precedent**”







# The Overarching Principle

---

## Organic architecture

---

From Wikipedia, the free encyclopedia

**Organic architecture** is a [philosophy](#) of [architecture](#) which promotes harmony between human habitation and the natural world through design approaches so sympathetic and well integrated with its site, that buildings, furnishings, and surroundings become part of a unified, interrelated composition.

A well-known example of organic architecture is [Fallingwater](#), the residence Frank Lloyd Wright designed for the Kaufmann family in rural Pennsylvania. Wright had many choices to locate a home on this large site, but chose to place the home directly over the waterfall and creek creating a close, yet noisy dialog with the rushing water and the steep site. The horizontal striations of stone masonry with daring [cantilevers](#) of colored beige concrete blend with native rock outcroppings and the wooded environment.



# Another Example: Precedent-Based Design

---





# Principled Design





# Another Principled Design



Source: By Martín Gómez Tagle - Lisbon, Portugal, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=13764903>

Source: <http://www.arcspace.com/exhibitions/unsorted/santiago-calatrava/>

# Another Principled Design

---





# Principle Applied to Another Structure



Source: By 準建築人手札網站 Forgemind ArchiMedia - Flickr: IMG\_2489.JPG, CC BY 2.0

Source: <https://www.dezeen.com/2016/08/29/santiago-calatrava-oculus-world-trade-center-transportation-hub-new-york-photographs-hufton-crow/>  
<https://commons.wikimedia.org/wiki/index.php?curid=91498396>, [https://en.wikipedia.org/wiki/Santiago\\_Calatrava](https://en.wikipedia.org/wiki/Santiago_Calatrava)



# The Overarching Principle

---

## Zoomorphic architecture

---

From Wikipedia, the free encyclopedia

**Zoomorphic architecture** is the practice of using animal forms as the inspirational basis and blueprint for architectural design. "While animal forms have always played a role adding some of the deepest layers of meaning in architecture, it is now becoming evident that a new strand of [biomorphism](#) is emerging where the meaning derives not from any specific representation but from a more general allusion to biological processes."<sup>[1]</sup>

Some well-known examples of Zoomorphic architecture can be found in the [TWA Flight Center](#) building in [New York City](#), by [Eero Saarinen](#), or the [Milwaukee Art Museum](#) by [Santiago Calatrava](#), both inspired by the form of a bird's wings.<sup>[3]</sup>

# Overarching Principle for Computing?

---



# Concluding Remarks

---

- It is time to design **principled system architectures** to solve the **memory problem**
- Design complete systems to be balanced, high-performance, and energy-efficient, i.e., **data-centric (or memory-centric)**
- Enable computation capability inside and close to memory
- **This** can
  - ❑ Lead to **orders-of-magnitude** improvements
  - ❑ **Enable new applications & computing platforms**
  - ❑ **Enable better understanding of nature**
  - ❑ ...



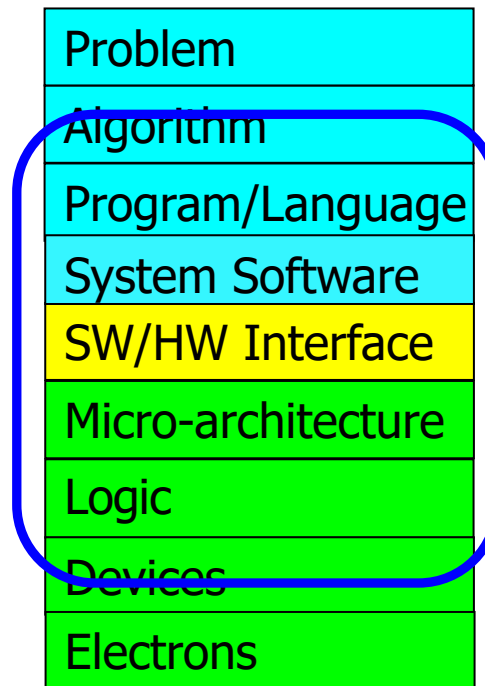
# The Future of Processing in Memory is Bright

---

- Regardless of challenges
  - in underlying technology and overlying problems/requirements

Can enable:

- Orders of magnitude improvements
- New applications and computing systems

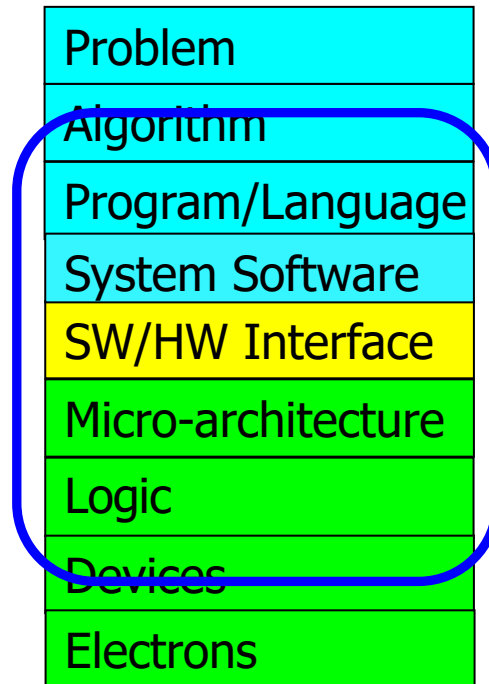


Yet, we have to

- Think across the stack
- Design enabling systems

# We Need to Revisit the Entire Stack

---



**We can get there step by step**

# We Need to Exploit Good Principles

---

- Data-centric system design
- All components intelligent
- Better cross-layer communication, better interfaces
- Better-than-worst-case design
- Heterogeneity
- Flexibility, adaptability

**Open minds**

# If In Doubt, See Other Doubtful Technologies

---

- A very “doubtful” emerging technology
  - for at least two decades



*Proceedings of the IEEE, Sept. 2017*

## Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

*This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.*

By YU CAI, SAUGATA GHOSE, ERICH F. HARATSCH, YIXIN LUO, AND ONUR MUTLU

# Flash Memory Timeline

## Flash Memory Timeline

1967

Dawon Kahng and Simon M. Sze invent the Non-Volatile Memory Floating Gate at Bell Labs; this is published as "A Floating Gate and Its Application to Memory Devices" (Bell System Technical Journal). Simon M. Sze went on to receive the 2014 FMS Lifetime Achievement Award

1970

Dov Frohman-Bentchkowsky invents the Erasable Programmable Read-Only Memory (EPROM) at Intel; this is published as "Memory Behavior in a Floating-Gate Avalanche-Injection MOS (FAMOS) Structure" in April 1971 (Applied Physics Letters), which cited the 1967 Kahng/Sze Bell Labs Floating Gate publication

1976

Hughes Microelectronics files Eli Haran patent for first practical floating gate EEPROM using thin SiO<sub>2</sub> and Fowler Nordheim tunneling for program and erase. Eli Haran went on to receive the 2012 FMS Lifetime Achievement Award

1977

Eli Haran of Hughes Microelectronics publishes "Conduction and Trapping of Electrons in Highly Stressed Thin Films of Thermal SiO<sub>2</sub>" (Applied Physics Letters)

1978

Eli Haran of Hughes Microelectronics publishes "Dielectric Breakdown in Electrically Stressed Thin Films of Thermal SiO<sub>2</sub>" (Journal of Applied Physics)

1979

Hughes Microelectronics introduces first CMOS NOR-1T1R 256-bit chip (non-volatile SRAM) employing Fowler Nordheim Floating gate EEPROM at IEEE ISSCC

1980

IEEE Solid State Circuits publishes paper titled "An Electrically Alterable Non-Volatile Memory Cell Using Floating Gate Structure" by Gulterman, Rinawi, Chieu, Holvorson, and McEvoy of Texas Instruments

Intel introduces the 2816, 16Kb HMOS EEPROM employing Fowler Nordheim tunneling

1981

British scientist and inventor Kane Kramer designs first digital audio player (IXI) based on magnetic bubble memory chips

1982

SEEO Technology introduces the 5213, first EEPROM with on-chip charge pump for in-system write and erase, an invention used in all flash memory devices

1983

Intel introduces 2817A 16Kb EEPROM

1984

First paper describing flash EEPROM presented by Fujio Masuoka of Toshiba at IEEE International Electron Devices Meeting (IEDM) in San Francisco. Fujio Masuoka went on to receive the 2013 FMS Lifetime Achievement Award

Intel begins flash process development. AT&T (Advanced Technology for Memory and Logic) is founded by George Perlegos, who went on to receive the 2017 Lifetime Achievement Award



Flash Memory Summit



# Flash Memory Timeline





# PIM Review and Open Problems

---

## A Modern Primer on Processing in Memory

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b,c</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>d</sup>

*SAFARI Research Group*

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*University of Illinois at Urbana-Champaign*

<sup>d</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,

**"A Modern Primer on Processing in Memory"**

*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*

# PIM Review and Open Problems (II)

---

## Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>b,c</sup>

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,  
**"Processing Data Where It Makes Sense: Enabling In-Memory  
Computation"**

*Invited paper in Microprocessors and Microsystems (**MICPRO**), June 2019.  
[arXiv version]*

# PIM Review and Open Problems (III)

---

## **A Workload and Programming Ease Driven Perspective of Processing-in-Memory**

Saugata Ghose<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Jeremie S. Kim<sup>†§</sup>   Juan Gómez-Luna<sup>§</sup>   Onur Mutlu<sup>§†</sup>

<sup>†</sup>*Carnegie Mellon University*

<sup>§</sup>*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

**"Processing-in-Memory: A Workload-Driven Perspective"**

*Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.*

[Preliminary arXiv version]

# Computer Architecture

## Lecture 8: Processing near Memory

Prof. Onur Mutlu

ETH Zürich

Fall 2021

22 October 2021

We Did Not Cover The Later Slides.  
They Are For Your Benefit.

# Accelerating Linked Data Structures

---

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,  
**"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"**  
*Proceedings of the 34th IEEE International Conference on Computer Design (ICCD)*, Phoenix, AZ, USA, October 2016.

## Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh<sup>†</sup> Samira Khan<sup>‡</sup> Nandita Vijaykumar<sup>†</sup>  
Kevin K. Chang<sup>†</sup> Amirali Boroumand<sup>†</sup> Saugata Ghose<sup>†</sup> Onur Mutlu<sup>§†</sup>  
<sup>†</sup>*Carnegie Mellon University*   <sup>‡</sup>*University of Virginia*   <sup>§</sup>*ETH Zürich*



# Executive Summary

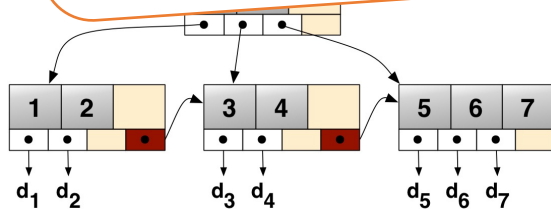
- **Our Goal:** Accelerating pointer chasing inside main memory
- **Challenges:** Parallelism challenge and Address translation challenge
- **Our Solution:** In-Memory Pointer Chasing Accelerator (IMPICA)
  - Address-access decoupling: enabling parallelism in the accelerator with low cost
  - IMPICA page table: low cost page table in logic layer
- **Key Results:**
  - 1.2X – 1.9X speedup for pointer chasing operations, +16% database throughput
  - 6% - 41% reduction in energy consumption

# Linked Data Structures

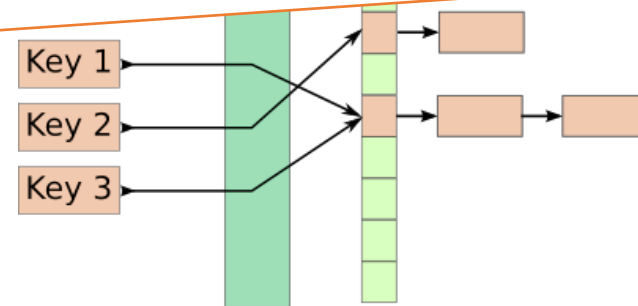
- Linked data structures are widely used in many important applications



**Linked data structures are connected by pointers**



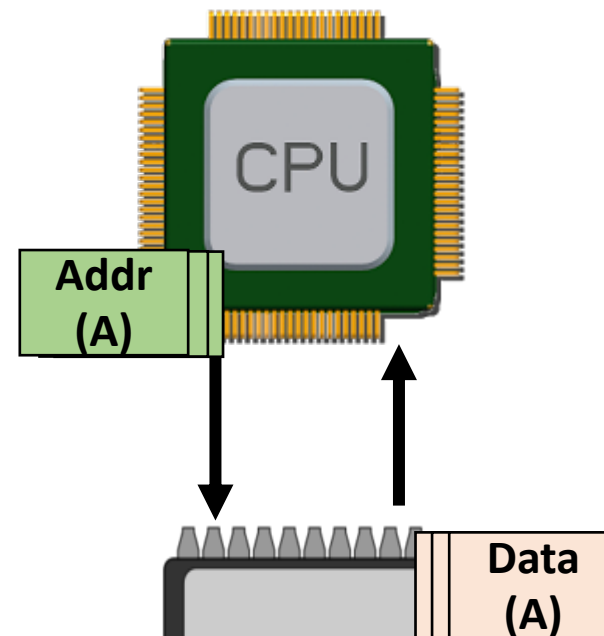
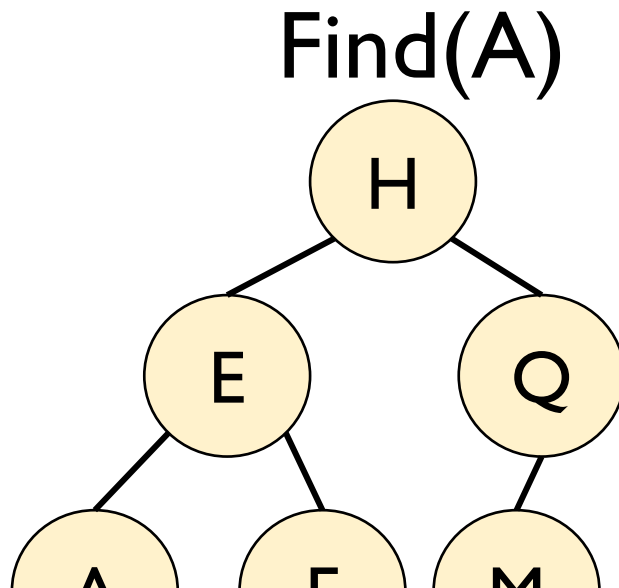
**B-Tree**



**Hash Table**

# The Problem: Pointer Chasing

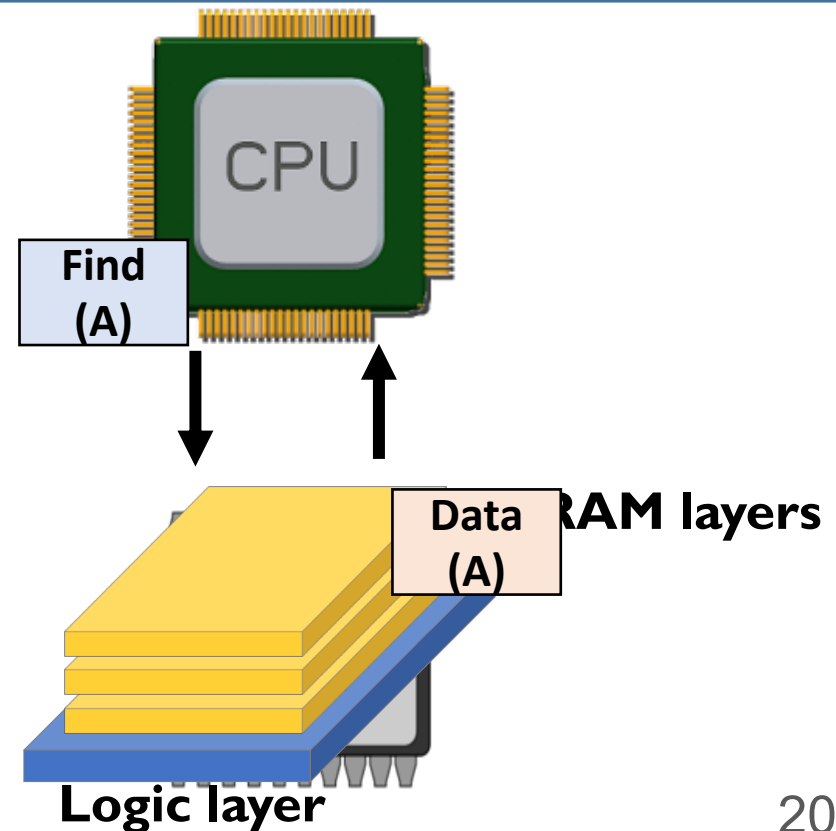
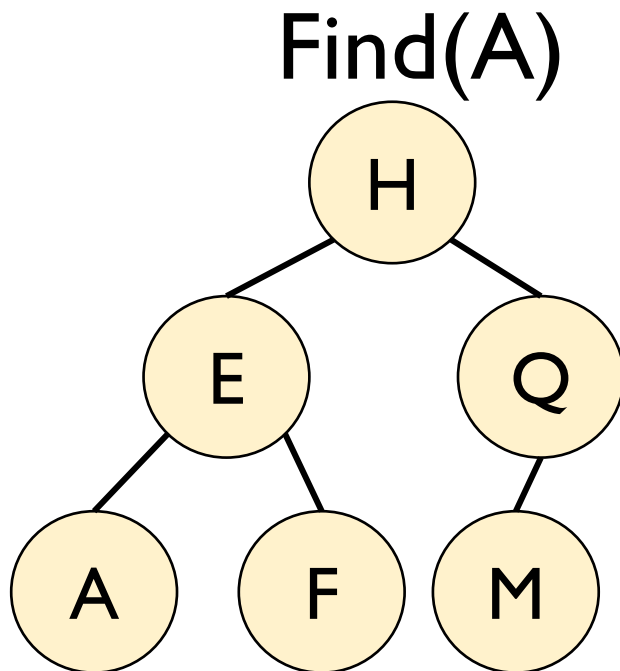
- Traversing linked data structures requires chasing pointers



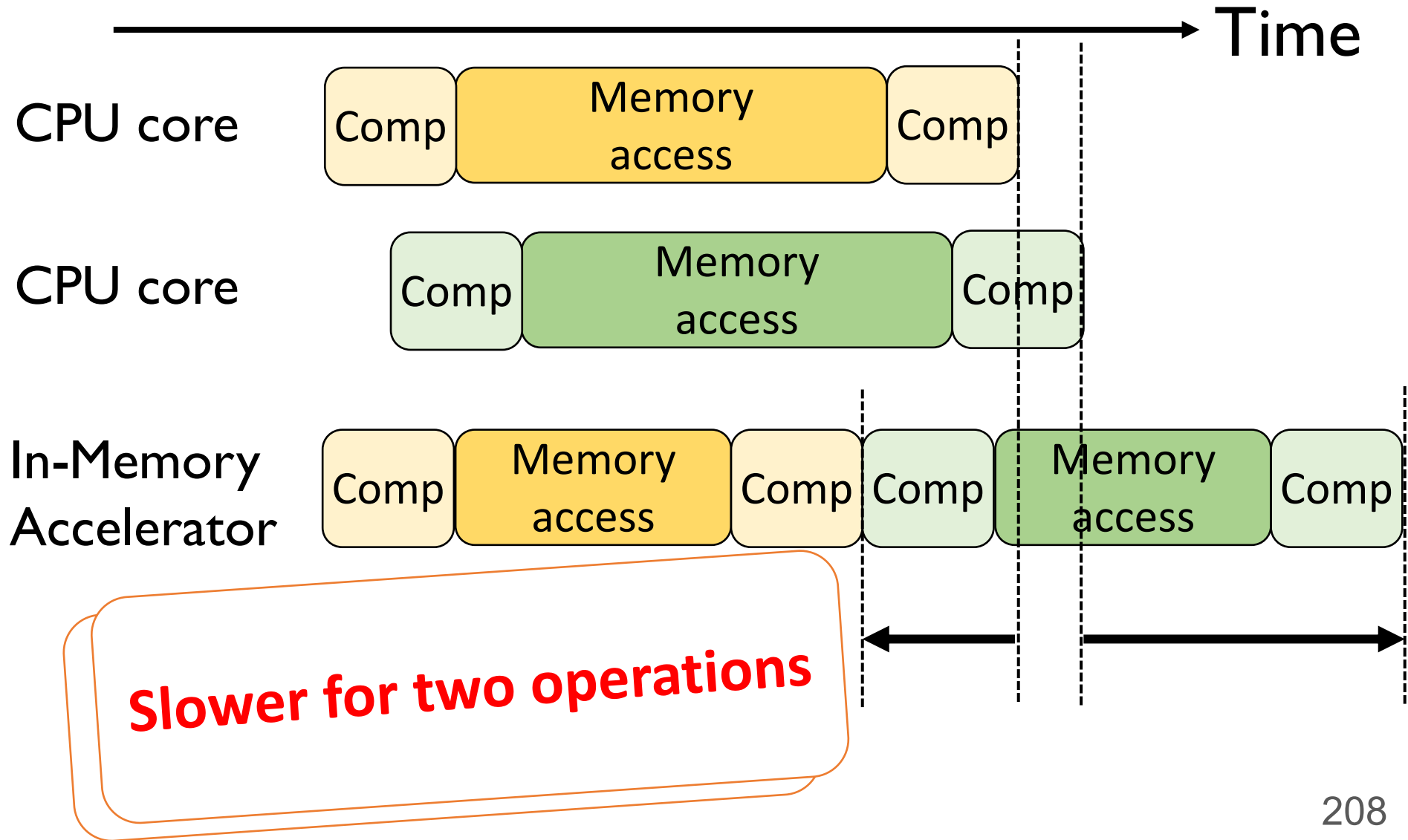
**Serialized and irregular access pattern  
6X cycles per instruction in real workloads**

# Our Goal

## Accelerating pointer chasing inside main memory

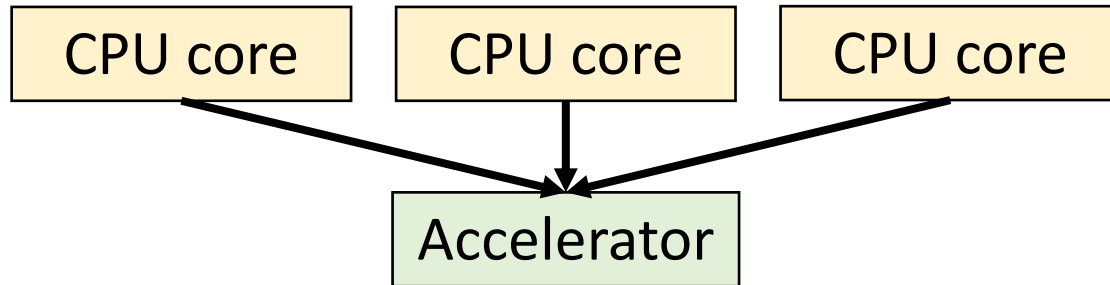


# Parallelism Challenge

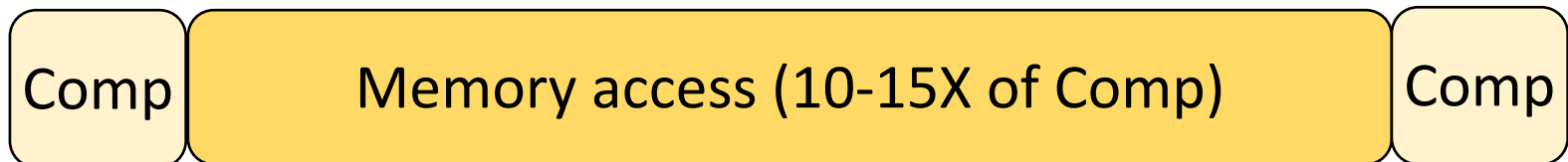


# Parallelism Challenge and Opportunity

- A simple in-memory accelerator can still be **slower** than multiple CPU cores

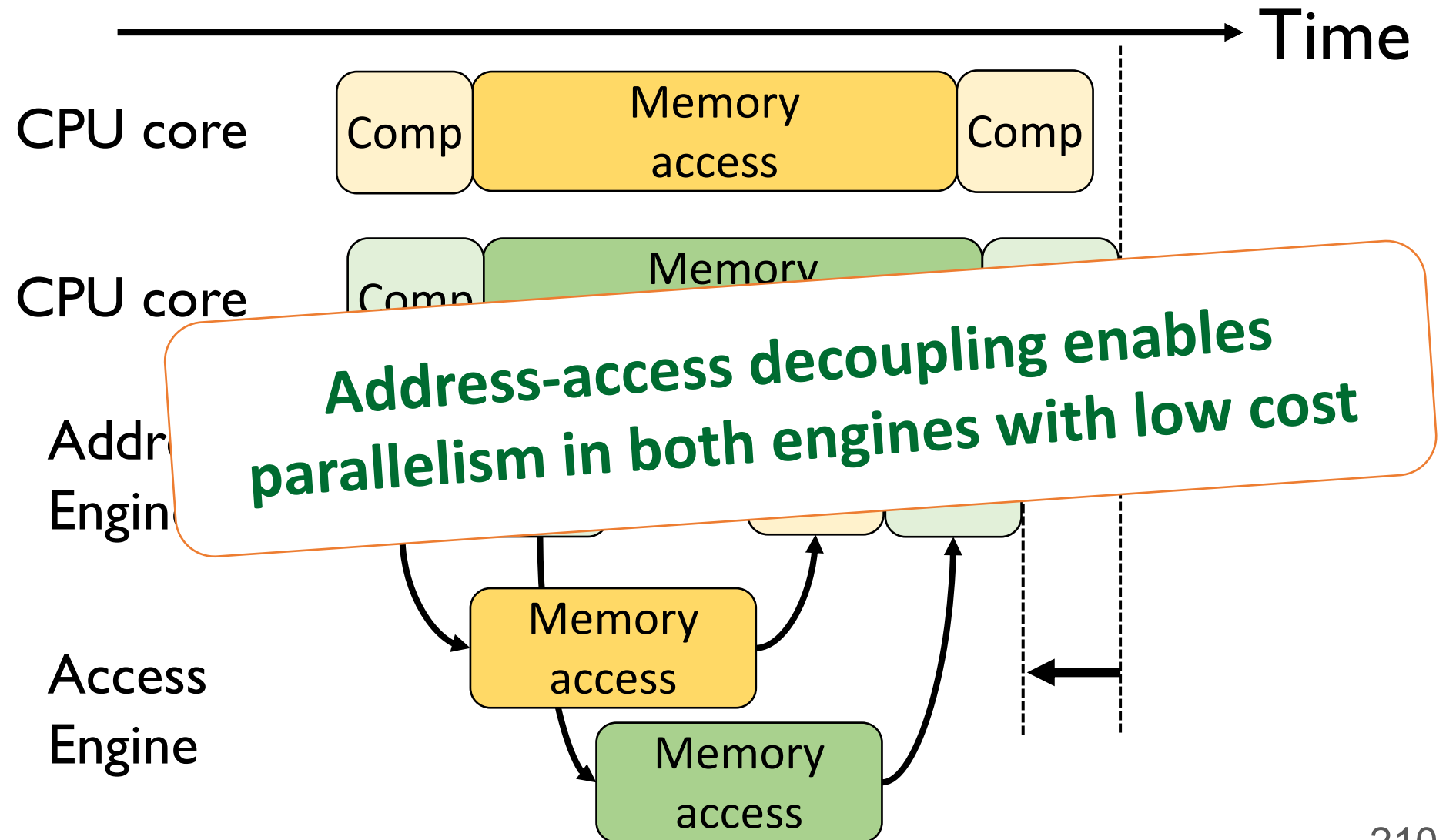


- **Opportunity:** a pointer-chasing accelerator spends a long time **waiting for memory**

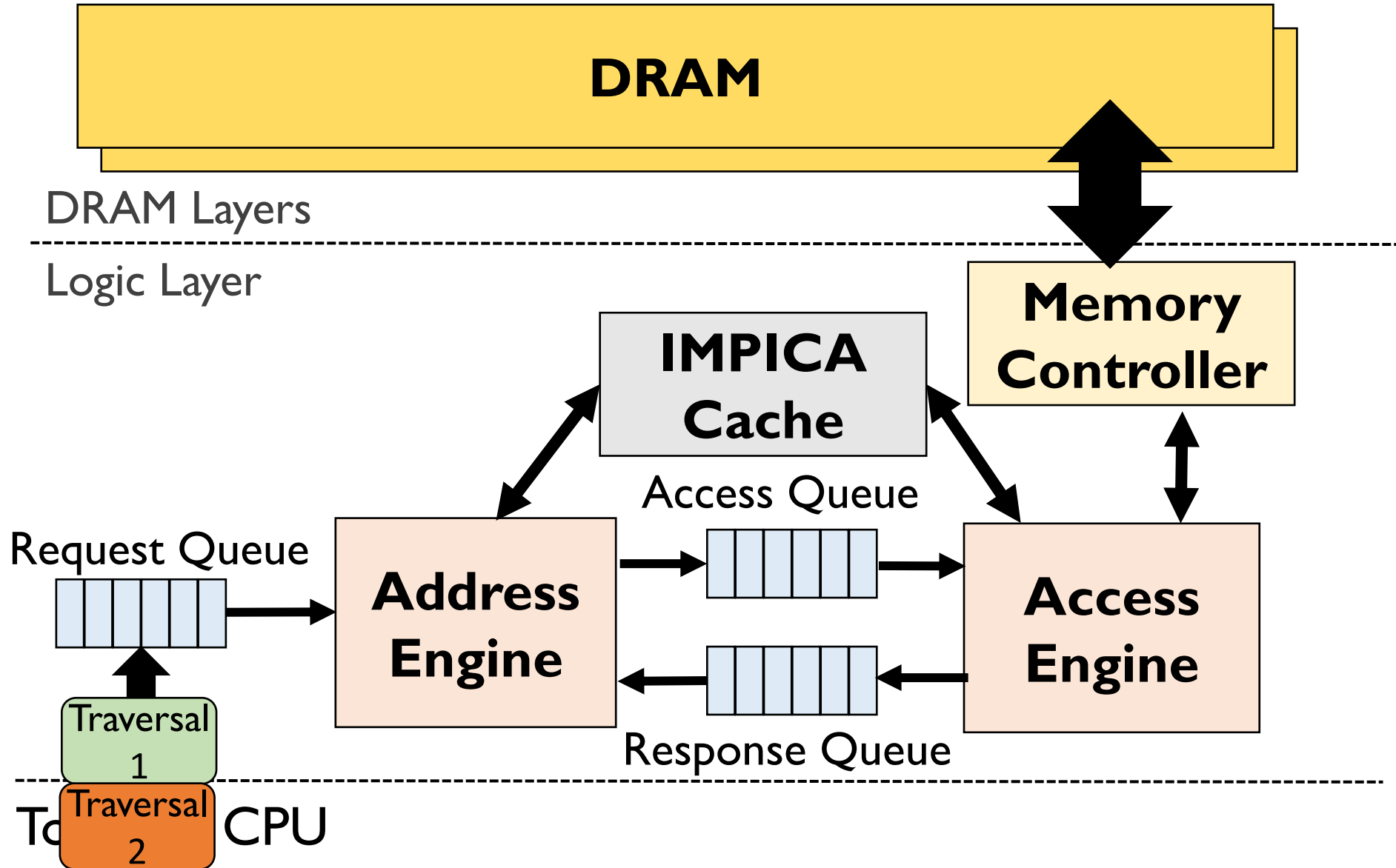




# Our Solution: Address-Access Decoupling

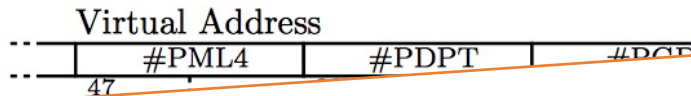


# IMPICA Core Architecture

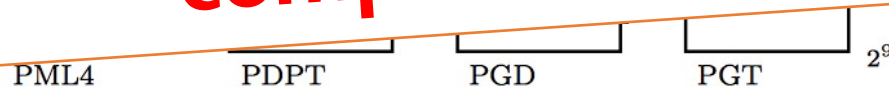


# Address Translation Challenge

**The page table walk requires multiple memory accesses**



**No TLB/MMU on the memory side**  
**Duplicating it is costly and creates compatibility issue**



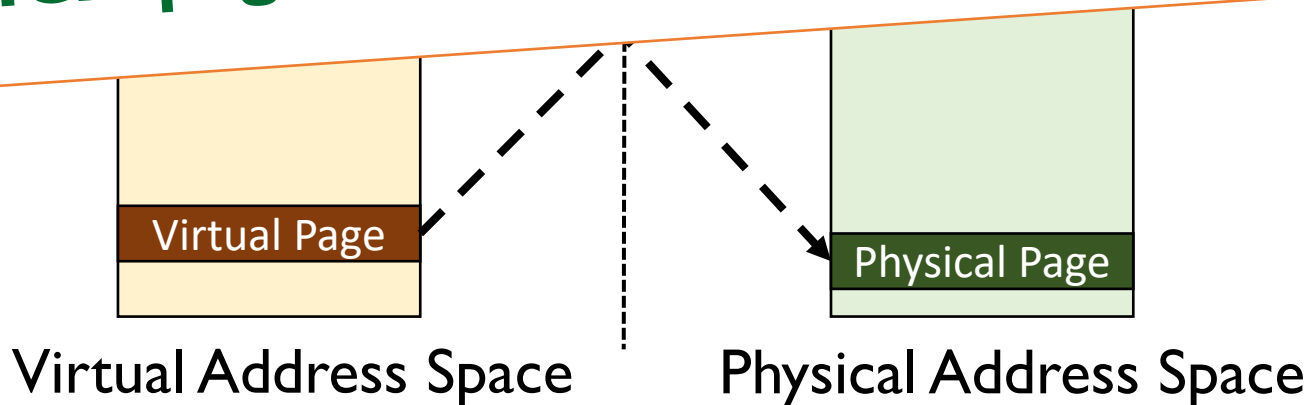
Page table walk

# Our Solution: IMPICA Page Table

- Completely decouple the page table of IMPICA from the page table of the CPUs

IMPICA Page Table

Map linked data structure into IMPICA regions  
IMPICA page table is a partial-to-any mapping



# IMPICA Page Table: Mechanism

Virtual Address

Bit [47:4]

Bit [11:0]

**Flat page table  
saves one memory access**

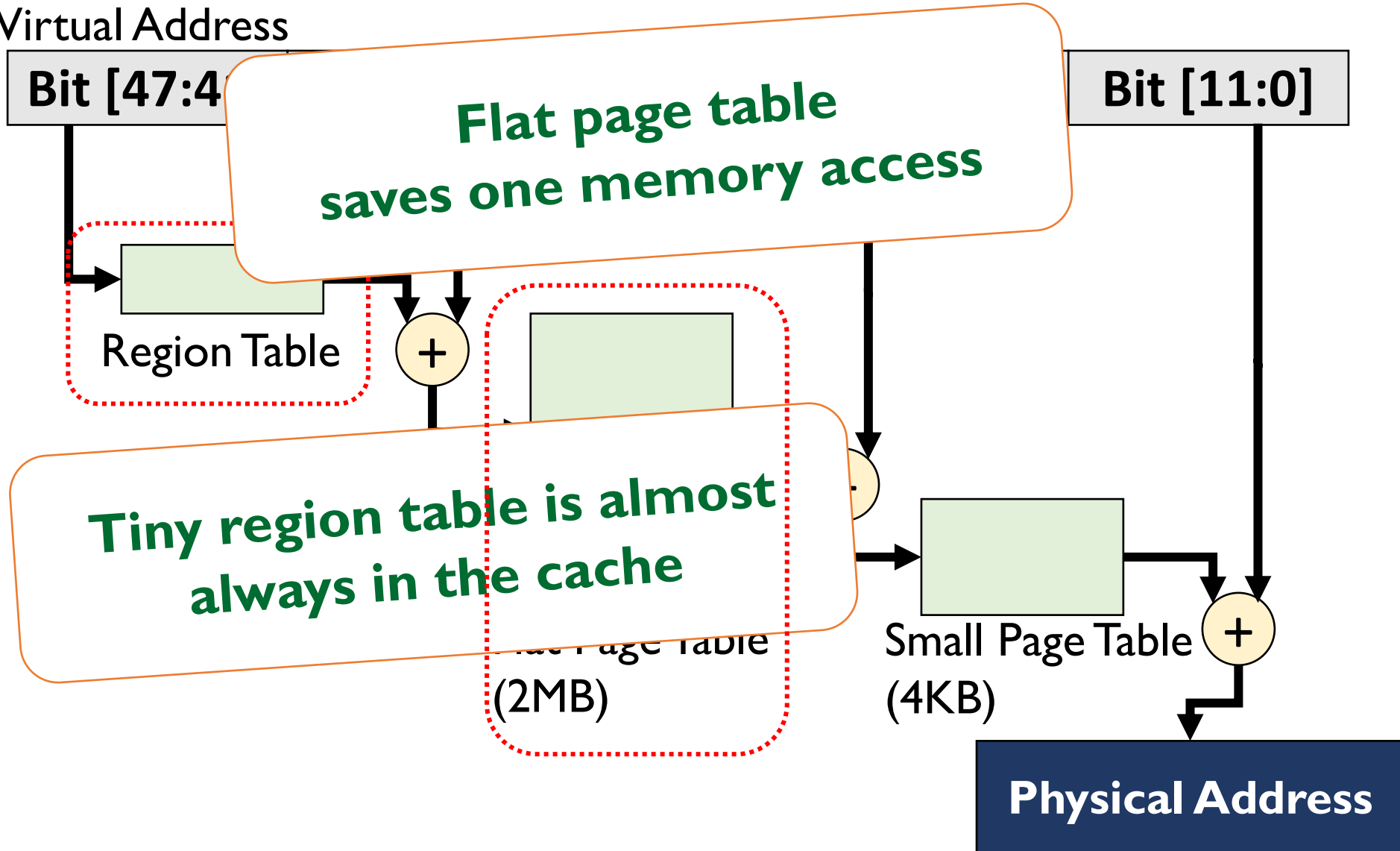
Region Table

**Tiny region table is almost  
always in the cache**

Large Page Table  
(2MB)

Small Page Table  
(4KB)

**Physical Address**

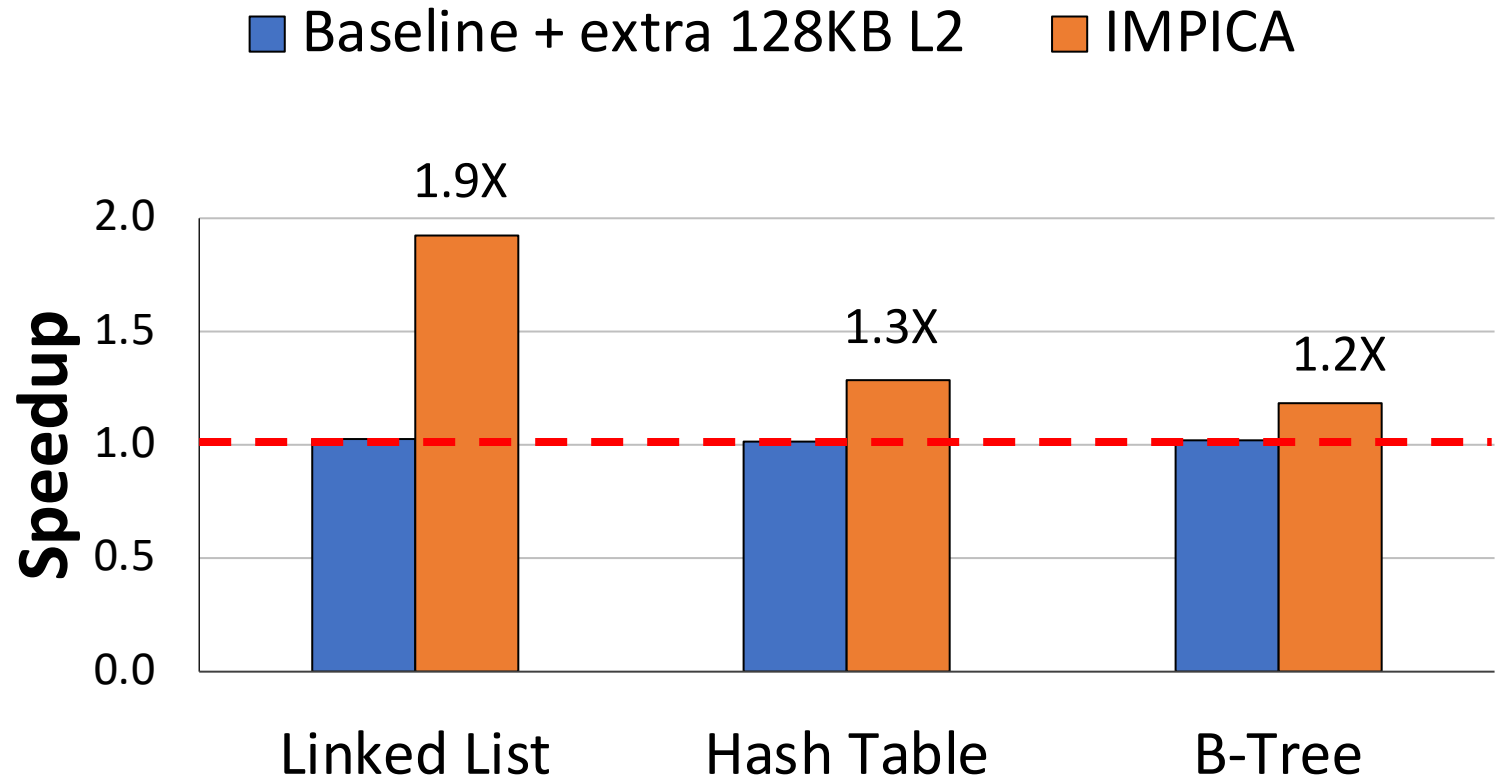


# Evaluation Methodology

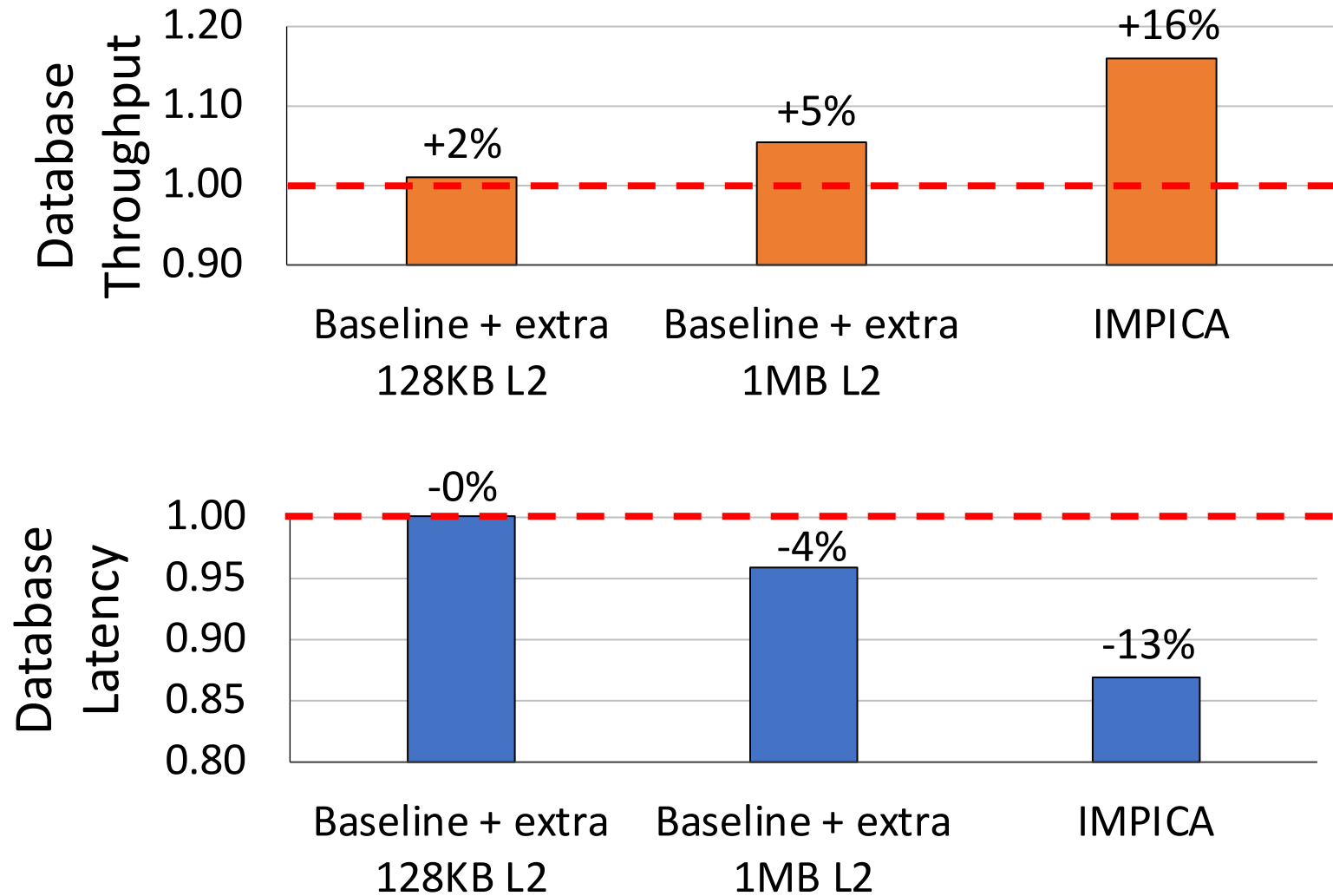
- Simulator: [gem5](#)
- System Configuration
  - CPU
    - 4 OoO cores, 2GHz
    - Cache: 32KB L1, 1MB L2
  - IMPICA
    - 1 core, 500MHz, 32KB Cache
  - Memory Bandwidth
    - 12.8 GB/s for CPU, 51.2 GB/s for IMPICA
- Our simulator code is open source
  - <https://github.com/CMU-SAFARI/IMPICA>



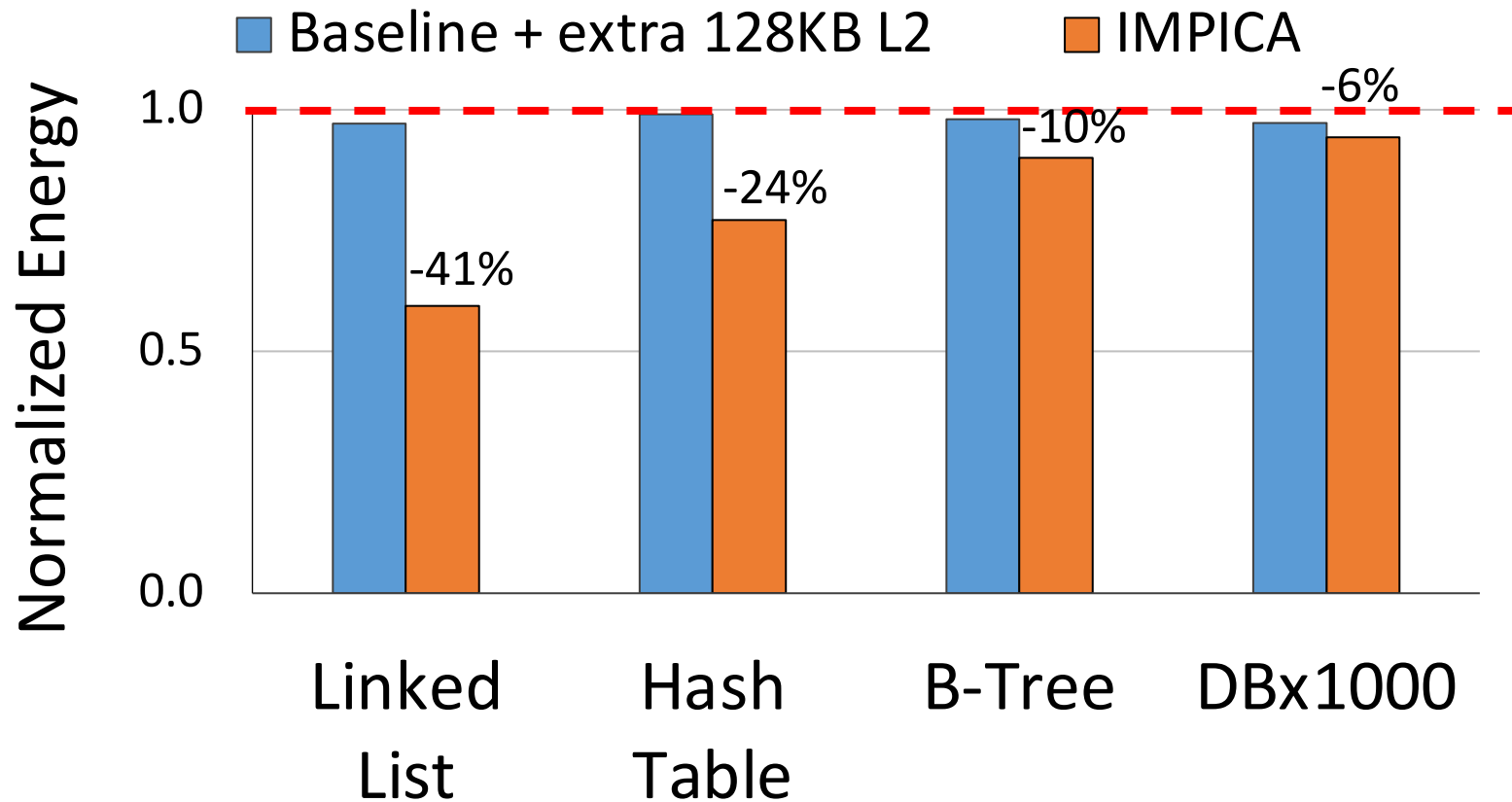
# Result – Microbenchmark Performance



# Result – Database Performance



# System Energy Consumption



# Area and Power Overhead

CPU (Cortex-A57)	5.85 mm <sup>2</sup> per core
L2 Cache	5 mm <sup>2</sup> per MB
Memory Controller	10 mm <sup>2</sup>
IMPICA (+32KB cache)	0.45 mm <sup>2</sup>

- Power overhead: average power increases by 5.6%