Trends, Challenges, and Opportunities in Main Memory
How Do We Solve The Memory Problem?

- **Fix it**: Make memory and controllers more intelligent
  - New interfaces, functions, architectures: system-mem codesign

- **Eliminate or minimize it**: Replace or (more likely) augment DRAM with a different technology
  - New technologies and system-wide rethinking of memory & storage

- **Embrace it**: Design heterogeneous memories (none of which are perfect) and map data intelligently across them
  - New models for data management and maybe usage

- ...
How Do We Solve The Memory Problem?

- **Fix it**: Make memory and controllers more intelligent
  - New interfaces, functions, architectures: system-mem codesign

- **Eliminate or minimize it**: Replace or (more likely) augment DRAM with a different technology
  - New technologies and system-wide rethinking of memory & storage

- **Embrace it**: Design heterogeneous memories (none of which are perfect) and map data intelligently across them
  - New models for data management and maybe usage

Solutions (to memory scaling) require software/hardware/device cooperation
How Do We Solve The Memory Problem?

- **Fix it**: Make memory and controllers more intelligent
  - New interfaces, architectures: system-mem codesign

- **Eliminate or minimize it**: Replace or (more likely) augment DRAM with a different technology
  - New technologies and system-wide rethinking of memory & storage

- **Embrace it**: Design heterogeneous memories (none of which are perfect) and map data intelligently across them
  - New models for data management and maybe usage

Solutions (to memory scaling) require software/hardware/device cooperation
Solution 1: New Memory Architectures

- Overcome memory shortcomings with
  - Memory-centric system design
  - Novel memory architectures, interfaces, functions
  - Better waste management (efficient utilization)

- Key issues to tackle
  - Enable reliability at low cost → high capacity
  - Reduce energy
  - Reduce latency
  - Improve bandwidth
  - Reduce waste (capacity, bandwidth, latency)
  - Enable computation close to data
Solution 1: New Memory Architectures

- Seshadri+, “RowDrain: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data,” MICRO 2013.
- Kim+, “‘Nipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors,” ISCA 2014.
- Seshadri+, “Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-uniform Strided Accesses,” MICRO 2015.
- Khan+, “A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM,” IEEE CAL 2016.
- Mutlu+, “The RowHammer Problem and Other Issues We May Face as Memory Becomes Dense,” DATE 2017.
Solution 2: Emerging Memory Technologies

- Some emerging **resistive** memory technologies seem more scalable than DRAM (and they are non-volatile)

Example: Phase Change Memory
- Data stored by changing phase of material
- Data read by detecting material’s resistance
- Expected to scale to 9nm (2022 [ITRS 2009])
- Prototyped at 20nm (Raoux+, IBM JRD 2008)
- Expected to be denser than DRAM: can store multiple bits/cell

- But, emerging technologies have (many) shortcomings
  - Can they be enabled to replace/augment/surpass DRAM?
Solution 2: Emerging Memory Technologies

- Zhao+, “FIRM: Fair and High-Performance Memory Control for Persistent Memory Systems,” MICRO 2014.
Combination: Hybrid Memory Systems

Hardware/software manage data allocation and movement to achieve the best of multiple technologies.

Exploiting Memory Error Tolerance with Hybrid Memory Systems

- **Vulnerable data**
- **Tolerant data**
- **Reliable memory**
- **Low-cost memory**

On Microsoft’s Web Search workload,

- Reduces server hardware cost by 4.7%.
- Achieves single server availability target of 99.90%.

**Heterogeneous-Reliability Memory** [DSN 2014]
Heterogeneous-Reliability Memory

**Step 1:** Characterize and classify application memory error tolerance

**Step 2:** Map application data to the HRM system enabled by *SW/HW cooperative solutions*

- **Reliable memory**
  - Reliable memory
  - Parity memory + software recovery (Par+R)

- **Unreliable memory**
  - Low-cost memory
Evaluation Results

- **Typical Server**
- **Consumer PC**
- **HRM**
- **Less-Tested (L)**
- **HRM/L**

Bigger area means better tradeoff

Outer is better

Inner is worse

Server HW cost savings (%)

Crashes/server/month

Memory cost savings (%)

13
More on Heterogeneous Reliability Memory

- Yixin Luo, Sriram Govindan, Bikash Sharma, Mark Santaniello, Justin Meza, Aman Kansal, Jie Liu, Badriddine Khessib, Kushagra Vaid, and Onur Mutlu,
"Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory"
Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Atlanta, GA, June 2014. [Summary] [Slides (pptx) (pdf)] [Coverage on ZDNet]

Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory

Yixin Luo  Sriram Govindan*  Bikash Sharma*  Mark Santaniello*  Justin Meza  Aman Kansal*  Jie Liu*  Badriddine Khessib*  Kushagra Vaid*  Onur Mutlu
Carnegie Mellon University, yixinluo@cs.cmu.edu, {meza, onur}@cmu.edu
*Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bkhessib, kvaid}@microsoft.com
An Orthogonal Issue: Memory Interference

- **Problem:** Memory interference between cores is uncontrolled
  - unfairness, starvation, low performance
  - uncontrollable, unpredictable, vulnerable system

- **Solution:** QoS-Aware Memory Systems
  - Hardware designed to provide a configurable fairness substrate
    - Application-aware memory scheduling, partitioning, throttling
  - Software designed to configure the resources to satisfy different QoS goals

- QoS-aware memory systems can provide predictable performance and higher efficiency
Strong Memory Service Guarantees

- **Goal:** Satisfy performance/SLA requirements in the presence of shared main memory, heterogeneous agents, and hybrid memory/storage

- **Approach:**
  - Develop techniques/models to accurately estimate the performance loss of an application/agent in the presence of resource sharing
  - Develop mechanisms (hardware and software) to enable the resource partitioning/prioritization needed to achieve the required performance levels for all applications
  - All the while providing high system performance

DRAM Controllers
Method and apparatus to control memory accesses

Abstract
A method and apparatus for accessing memory comprising monitoring memory accesses from a hardware prefetcher and determining whether the memory accesses from the hardware prefetcher are used by an out-of-order core. A front side bus controller switches memory access modes from a minimize memory access latency mode to a maximize memory bus bandwidth mode if a percentage of the memory accesses generated by the hardware prefetcher are used by the out-of-order core.

Images (6)

Classifications

G06F12/0215  Addressing or allocation; Relocation with look ahead addressing means

SAFARI
Memory Performance Attacks [USENIX SEC’07]

- Thomas Moscibroda and Onur Mutlu, "Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems"
  

Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems

Thomas Moscibroda    Onur Mutlu
Microsoft Research
{moscitho, onur}@microsoft.com
Onur Mutlu and Thomas Moscibroda,
"Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors"
*Proceedings of the 40th International Symposium on Microarchitecture (MICRO)*, pages 146-158, Chicago, IL, December 2007. [Summary] [Slides (ppt)]

[Summary] [Slides (ppt)]
On PAR-BS

- Variants implemented in Samsung SoC memory controllers

Effective platform level approach and DRAM accesses are crucial to system performance. This paper touches this topics and suggest a superior approach to current known techniques.

Review from ISCA 2008
ATLAS Memory Scheduler [HPCA’10]

- Yoongu Kim, Dongsu Han, Onur Mutlu, and Mor Harchol-Balter, "ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers" Proceedings of the 16th International Symposium on High-Performance Computer Architecture (HPCA), Bangalore, India, January 2010. Slides (pptx)
Yoongu Kim, Michael Papamichael, Onur Mutlu, and Mor Harchol-Balter,
"Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior"
BLISS [ICCD’14, TPDS’16]

- Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, and Onur Mutlu,
  "The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost"

Proceedings of the 32nd IEEE International Conference on Computer Design (ICCD), Seoul, South Korea, October 2014.
[Slides (pptx) (pdf)]

The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost

Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, Onur Mutlu
Carnegie Mellon University
{lsubrama,donghyu1,visesh,harshar,onur}@cmu.edu
Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems

Rachata Ausavarungnirun, Kevin Chang, Lavanya Subramanian, Gabriel Loh, and Onur Mutlu,

"Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems"

Proceedings of the 39th International Symposium on Computer Architecture (ISCA), Portland, OR, June 2012. Slides (pptx)
Hiroyuki Usui, Lavanya Subramanian, Kevin Kai-Wei Chang, and Onur Mutlu,
"DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators"
Presented at the 11th HiPEAC Conference, Prague, Czech Republic, January 2016.
[Slides (pptx) (pdf)]
[Source Code]
MISE: Predictable Performance [HPCA’13]

- Lavanya Subramanian, Vivek Seshadri, Yoongu Kim, Ben Jaiyen, and Onur Mutlu,

"MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems"

Proceedings of the 19th International Symposium on High-Performance Computer Architecture (HPCA), Shenzhen, China, February 2013. Slides (pptx)
ASM: Predictable Performance [MICRO’15]

- Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu,

"The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory"

Proceedings of the 48th International Symposium on Microarchitecture (MICRO), Waikiki, Hawaii, USA, December 2015.

[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
[Source Code]
The Future

Memory Controllers are critical to research

They will become even more important
Memory Control is Getting More Complex

- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs

Many goals, many constraints, many metrics …
Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana, "Self Optimizing Memory Controllers: A Reinforcement Learning Approach" 
The Future

Memory Controllers: Many New Problems
Takeaway

Main Memory Needs
Intelligent Controllers
What We Will Cover
In The Next Few Lectures
Agenda for The Next Few Lectures

- Memory Importance and Trends
- RowHammer: Memory Reliability and Security
- In-Memory Computation
- Low-Latency Memory
- Data-Driven and Data-Aware Architectures
- Guiding Principles & Conclusion
An “Early” Position Paper [IMW’13]

- Onur Mutlu,
  "Memory Scaling: A Systems Architecture Perspective"
  Proceedings of the 5th International Memory Workshop (IMW), Monterey, CA, May 2013. Slides (pptx) (pdf)
  EETimes Reprint

Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

Challenges in DRAM Scaling

- Refresh
- Latency
- Bank conflicts/parallelism
- Reliability and vulnerabilities
- Energy & power
- Memory’s inability to do more than store data
A Recent Retrospective Paper [TCAD’19]

- Onur Mutlu and Jeremie Kim, "RowHammer: A Retrospective"
  [Preliminary arXiv version]

RowHammer: A Retrospective

Onur Mutlu$^\S\S$
§ETH Zürich

Jeremie S. Kim$^{\S\S}$
‡Carnegie Mellon University
Computer Architecture
Lecture 4a: Memory Solution Ideas

Prof. Onur Mutlu
ETH Zürich
Fall 2019
27 September 2019
Backup Slides
Readings, Videos, Reference Materials
Accelerated Memory Course (~6.5 hours)

- **ACACES 2018**
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu July 9-13, 2018
  - ~6.5 hours of lectures

- **Website for the Course including Videos, Slides, Papers**
  - [https://safari.ethz.ch/memory_systems/ACACES2018/](https://safari.ethz.ch/memory_systems/ACACES2018/)
  - [https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x](https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x)

- **All Papers are at:**
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)
  - Final lecture notes and readings (for all topics)
Longer Memory Course (~18 hours)

- **Tu Wien 2019**
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu June 12-19, 2019
  - ~18 hours of lectures

- **Website for the Course including Videos, Slides, Papers**
  - [https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_gntM55VoMIKlw7YrXOhbl](https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_gntM55VoMIKlw7YrXOhbl)

- **All Papers are at:**
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)
  - Final lecture notes and readings (for all topics)
Some Overview Talks

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZlSOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI

- Enabling In-Memory Computation
  - https://www.youtube.com/watch?v=oHqsNbxgdzM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=7

- Accelerating Genome Analysis
  - https://www.youtube.com/watch?v=hPnSmfwu2-A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=9

- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=3
Reference Overview Paper I

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

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{b,c}

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}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]  

Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND, RACHATA AUSAVARUNGNIRUN
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University

Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu, "Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions"
[Preliminary arxiv.org version]

Onur Mutlu and Lavanya Subramanian, "Research Problems and Opportunities in Memory Systems"

Invited Article in *Supercomputing Frontiers and Innovations (SUPERFRI)*, 2014/2015.

Onur Mutlu,
"The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
[Slides (pptx) (pdf)]
Onur Mutlu,
"Memory Scaling: A Systems Architecture Perspective"
Technical talk at MemCon 2013 (MEMCON), Santa Clara, CA, August 2013. [Slides (pptx) (pdf)] [Video] [Coverage on StorageSearch]
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
Onur Mutlu and Jeremie Kim,
"RowHammer: A Retrospective"
[Preliminary arXiv version]
Related Videos and Course Materials (I)


- **Parallel Computer Architecture Course Materials (Lecture Videos)**
Related Videos and Course Materials (II)

- Memory Systems Short Course Materials (Lecture Video on Main Memory and DRAM Basics)
Some Open Source Tools (I)

- **Rowhammer – Program to Induce RowHammer Errors**
  - [https://github.com/CMU-SAFA/rowhammer](https://github.com/CMU-SAFA/rowhammer)
- **Ramulator – Fast and Extensible DRAM Simulator**
  - [https://github.com/CMU-SAFA/ramulator](https://github.com/CMU-SAFA/ramulator)
- **MemSim – Simple Memory Simulator**
  - [https://github.com/CMU-SAFA/memsim](https://github.com/CMU-SAFA/memsim)
- **NOCulator – Flexible Network-on-Chip Simulator**
  - [https://github.com/CMU-SAFA/NOCulator](https://github.com/CMU-SAFA/NOCulator)
- **SoftMC – FPGA-Based DRAM Testing Infrastructure**
  - [https://github.com/CMU-SAFA/SoftMC](https://github.com/CMU-SAFA/SoftMC)

- **Other open-source software from my group**
  - [https://github.com/CMU-SAFA/](https://github.com/CMU-SAFA/)
  - [http://www.ece.cmu.edu/~safari/tools.html](http://www.ece.cmu.edu/~safari/tools.html)
Some Open Source Tools (II)

- MQSim – A Fast Modern SSD Simulator
  - https://github.com/CMU-SAFARI/MQSim

- Mosaic – GPU Simulator Supporting Concurrent Applications
  - https://github.com/CMU-SAFARI/Mosaic

- IMPICA – Processing in 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/IMPICA

- SMLA – Detailed 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/SMLA

- HWASim – Simulator for Heterogeneous CPU-HWA Systems
  - https://github.com/CMU-SAFARI/HWASim

- Other open-source software from my group
  - https://github.com/CMU-SAFARI/
  - http://www.ece.cmu.edu/~safari/tools.html
More Open Source Tools (III)

- A lot more open-source software from my group
  - [https://github.com/CMU-SAFARI/](https://github.com/CMU-SAFARI/)
  - [http://www.ece.cmu.edu/~safari/tools.html](http://www.ece.cmu.edu/~safari/tools.html)
Referenced Papers

- All are available at

  https://people.inf.ethz.ch/omutlu/projects.htm

  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

Ramulator: A Fast and Extensible DRAM Simulator

[IEEE Comp Arch Letters’15]
Ramulator Motivation

- DRAM and Memory Controller landscape is changing
- Many new and upcoming standards
- Many new controller designs
- A fast and easy-to-extend simulator is very much needed

<table>
<thead>
<tr>
<th>Segment</th>
<th>DRAM Standards &amp; Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commodity</td>
<td>DDR3 (2007) [14]; DDR4 (2012) [18]</td>
</tr>
<tr>
<td>Performance</td>
<td>eDRAM [28], [32]; RLDRAM3 (2011) [29]</td>
</tr>
</tbody>
</table>

Table 1. Landscape of DRAM-based memory
Ramulator

- Provides out-of-the-box support for many DRAM standards:
  - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAm, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

<table>
<thead>
<tr>
<th>Simulator (clang -O3)</th>
<th>Cycles (10^6)</th>
<th>Runtime (sec.)</th>
<th>Req/sec (10^3)</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Random</td>
<td>Stream</td>
<td>Random</td>
<td>Stream</td>
</tr>
<tr>
<td>Ramulator</td>
<td>652</td>
<td>411</td>
<td>752</td>
<td>249</td>
</tr>
<tr>
<td>DRAMSim2</td>
<td>645</td>
<td>413</td>
<td>2,030</td>
<td>876</td>
</tr>
<tr>
<td>USIMM</td>
<td>661</td>
<td>409</td>
<td>1,880</td>
<td>750</td>
</tr>
<tr>
<td>DrSim</td>
<td>647</td>
<td>406</td>
<td>18,109</td>
<td>12,984</td>
</tr>
<tr>
<td>NVMain</td>
<td>666</td>
<td>413</td>
<td>6,881</td>
<td>5,023</td>
</tr>
</tbody>
</table>

Table 3. Comparison of five simulators using two traces
Case Study: Comparison of DRAM Standards

<table>
<thead>
<tr>
<th>Standard</th>
<th>Rate (MT/s)</th>
<th>Timing (CL-RCD-RP)</th>
<th>Data-Bus (Width × Chan.)</th>
<th>Rank-per-Chan</th>
<th>BW (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3</td>
<td>1,600</td>
<td>11-11-11</td>
<td>64-bit × 1</td>
<td>1</td>
<td>11.9</td>
</tr>
<tr>
<td>DDR4</td>
<td>2,400</td>
<td>16-16-16</td>
<td>64-bit × 1</td>
<td>1</td>
<td>17.9</td>
</tr>
<tr>
<td>SALP†</td>
<td>1,600</td>
<td>11-11-11</td>
<td>64-bit × 1</td>
<td>1</td>
<td>11.9</td>
</tr>
<tr>
<td>LPDDR3</td>
<td>1,600</td>
<td>12-15-15</td>
<td>64-bit × 1</td>
<td>1</td>
<td>11.9</td>
</tr>
<tr>
<td>LPDDR4</td>
<td>2,400</td>
<td>22-22-22</td>
<td>32-bit × 2*</td>
<td>1</td>
<td>17.9</td>
</tr>
<tr>
<td>GDDR5 [12]</td>
<td>6,000</td>
<td>18-18-18</td>
<td>64-bit × 1</td>
<td>1</td>
<td>44.7</td>
</tr>
<tr>
<td>HBM</td>
<td>1,000</td>
<td>7-7-7</td>
<td>128-bit × 8*</td>
<td>1</td>
<td>119.2</td>
</tr>
<tr>
<td>WIO</td>
<td>266</td>
<td>7-7-7</td>
<td>128-bit × 4*</td>
<td>1</td>
<td>15.9</td>
</tr>
<tr>
<td>WIO2</td>
<td>1,066</td>
<td>9-10-10</td>
<td>128-bit × 8*</td>
<td>1</td>
<td>127.2</td>
</tr>
</tbody>
</table>

Across 22 workloads, simple CPU model
Ramulator Paper and Source Code


- Source code is released under the liberal MIT License
  - [https://github.com/CMU-SAFARI/ramulator](https://github.com/CMU-SAFARI/ramulator)

---

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim\(^1\)  Weikun Yang\(^1,2\)  Onur Mutlu\(^1\)

\(^1\)Carnegie Mellon University  \(^2\)Peking University
Optional Assignment

- Review the Ramulator paper
  - Email me your review (omutlu@gmail.com)

- Download and run Ramulator
  - Compare DDR3, DDR4, SALP, HBM for the libquantum benchmark (provided in Ramulator repository)
  - Email me your report (omutlu@gmail.com)

- This will help you get into memory systems research
End of Backup Slides