Simulation: The Field of Dreams
Some General Issues in Architectural Simulation
An Example Simulator
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, TLDGRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

<table>
<thead>
<tr>
<th>Simulator</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>NVMMain</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

![Performance comparison of DRAM standards](Figure 2)
Ramulator Paper and Source Code


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

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim¹, Weikun Yang¹,², Onur Mutlu¹
¹Carnegie Mellon University, ²Peking University
Ramulator is a fast and cycle-accurate DRAM simulator [1, 2] that supports a wide array of commercial, as well as academic, DRAM standards:

- LPDDR3 (2012), LPDDR4 (2014)
- GDDR5 (2009)
- HBM (2013)
- SALP [3]
- TL-DRAM [4]
- RowClone [5]
- DSARP [6]

The initial release of Ramulator is described in the following paper:


For information on new features, along with an extensive memory characterization using Ramulator, please read:


Usage

Ramulator supports three different usage modes:

1. Memory Trace Driven: Ramulator directly reads memory traces from a file, and simulates only the DRAM subsystem. Each line in the trace file represents a memory request, with the hexadecimal address followed by 'R' or 'W' for read or write.

   - 0x12345680 R
   - 0x4cbb56c0 W
   - ...
Usage

Ramulator supports three different usage modes.

1. Memory Trace Driver: Ramulator directly reads memory traces from a file, and simulates only the DRAM subsystem. Each line in the trace file represents a memory request, with the hexadecimal address followed by ‘R’ or ‘W’ for read or write.
   - Ox12345670 R
   - Ox4cb56c0 W
   - ...

2. CPU Trace Driver: Ramulator directly reads instruction traces from a file, and simulates a simplified model of a “core” that generates memory requests to the DRAM subsystem. Each line in the trace file represents a memory request, and can have one of the following two formats.
   - <num-pcinst> <addr-read>: For a line with two tokens, the first token represents the number of CPU (i.e., non-memory) instructions before the memory request, and the second token is the decimal address of a read.
   - <num-pcinst> <addr-read> <addr-writeback>: For a line with three tokens, the third token is the decimal address of the writeback request, which is the dirty cache line evicted caused by the read request before it.

3. gem5 Driver: Ramulator runs as part of a full-system simulator (gem5 [7]), from which it receives memory request as they are generated.

For some of the DRAM standards, Ramulator is also capable of reporting power consumption by relying on either VAMPIRE [8] or DRAMPower [9] as the backend.


Getting Started

Ramulator requires a C++11 compiler (e.g., clang++, g++).

1. Memory Trace Driver

   $ cd ramulator
   $ make -j
   $ ./ramulator configs/DDR3-config.cfg --mode=dram dram.trace
   Simulation done. Statistics written to DDR3.stats
   # NOTE: dram.trace is a very short trace file provided only as an example.
   $ ./ramulator configs/DDR3-config.cfg --mode=dram --stats my_output.txt dram.trace
   Simulation done. Statistics written to my_output.txt
   # NOTE: optional --stats flag changes the statistics output filename

2. CPU Trace Driver
Ramulator: Reproducibility

Reproducing Results from Paper (Kim et al. [1])

Debugging & Verification (Section 4.1)
For debugging and verification purposes, Ramulator can print the trace of every DRAM command it issues along with their address and timing information. To do so, please turn on the `print_cmd_trace` variable in the configuration file.

Comparison Against Other Simulators (Section 4.2)
For comparing Ramulator against other DRAM simulators, we provide a script that automates the process: `test_ddr3.py`. Before you run this script, however, you must specify the location of their executables and configuration files at designated lines in the script's source code:

- Ramulator
- DRAMSim2 [https://wiki.umd.edu/DRAMSim2]: test_ddr3.py lines 39-40
- USIMM [http://www.cs.utah.edu/~rajev/wac12]: test_ddr3.py lines 54-55
- DrSim [http://ph.ece.utexas.edu/public/Main/DrSim]: test_ddr3.py lines 66-67
- NVMain [http://wiki.nvmain.org]: test_ddr3.py lines 78-79

Please refer to their respective websites to download, build, and set-up the other simulators. The simulators must be executed in saturation mode (always filling up the request queues when possible).

All five simulators were configured using the same parameters:
- DOR3-1600X (11-11-11), 1 Channel, 1 Rank, 2Gb x8 chips
- FR-FCFS Scheduling
- Open-Row Policy
- 32/32 Entry Read/Write Queues
- High/Low Watermarks for Write Queue: 28/16

Finally, execute `test_ddr3.py <num-requests>` to start off the simulation. Please make sure that there are no other active processes during simulation to yield accurate measurements of memory usage and CPU time.

Cross-Sectional Study of DRAM Standards (Section 4.3)
Please use the CPU traces (SPEC 2006) provided in the `cpu traces` folder to run CPU trace driven simulations.

Other Tips

Power Estimation
For estimating power consumption, Ramulator can record the trace of every DRAM command it issues to a file in DRAMPower [8] format. To do so, please turn on the `record_cmd_trace` variable in the configuration file. The resulting DRAM command trace (e.g., `cmd-trace-chan-4-rank-1.cmdtrace`) should be fed into a compatible DRAM energy simulator such as VAMPIRE [8] or DRAMPower [9] with the correct configuration (standard/speed/organization) to estimate energy/power usage for a single rank (a current limitation of both VAMPIRE and DRAMPower).


[https://github.com/CMU-SAFARI/ramulator](https://github.com/CMU-SAFARI/ramulator)
Ramulator Project Course
Exploration of Emerging Memory Systems (Spring/Fall 2022)

**Fall 2022 Edition:**
- https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=ramulator

**Spring 2022 Edition:**
- https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=ramulator

**Youtube Livestream (Spring 2022):**
- https://www.youtube.com/watch?v=aMLlXROq3s&list=PL5Q2soXY2ZiTLmLGwZ8hBo2925ZApqV

**Bachelor’s course**
- Elective at ETH Zurich
- Introduction to memory system simulation
- Tutorial on using Ramulator
- C++
- Potential research exploration

https://www.youtube.com/onurmutlulectures
Bonus Assignment as Part of Next HW

- Review the Ramulator paper
  - Same points as any other BONUS review in the next HW
Example Studies using Ramulator
An Example Study using Ramulator (I)

- Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu,
  "Demystifying Workload–DRAM Interactions: An Experimental Study"
  [Preliminary arXiv Version]
  [Abstract]
  [Slides (pptx) (pdf)]
  [MemBen Benchmark Suite]
  [Source Code for GPGPUSim-Ramulator]

Demystifying Complex Workload–DRAM Interactions: An Experimental Study

Saugata Ghose† Tianshi Li† Nastaran Hajinazar†‡
Damla Senol Cali† Onur Mutlu§†

†Carnegie Mellon University ‡Simon Fraser University §ETH Zürich
Why Study Workload–DRAM Interactions?

- Manufacturers are developing many new types of DRAM
  - DRAM limits performance, energy improvements: new types may overcome some limitations
  - Memory systems now serve a very diverse set of applications: can no longer take a one-size-fits-all approach

- So which DRAM type works best with which application?
  - Difficult to understand intuitively due to the complexity of the interaction
  - Can’t be tested methodically on real systems: new type needs a new CPU

- We perform a wide-ranging experimental study to uncover the combined behavior of workloads and DRAM types
  - 115 prevalent/emerging applications and multiprogrammed workloads
  - 9 modern DRAM types: DDR3, DDR4, GDDR5, HBM, HMC, LPDDR3, LPDDR4, Wide I/O, Wide I/O 2
Modern DRAM Types: Comparison to DDR3

<table>
<thead>
<tr>
<th>DRAM Type</th>
<th>Banks per Rank</th>
<th>Bank Groups</th>
<th>3D-Stacked</th>
<th>Low-Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDR4</td>
<td>16</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDDR5</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>HBM High-Bandwidth Memory</td>
<td>16</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>HMC Hybrid Memory Cube</td>
<td>256</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Wide I/O</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wide I/O 2</td>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LPDDR3</td>
<td>8</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>LPDDR4</td>
<td>16</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

- **Bank groups**
  - Bank Group 1: Increased latency
  - Bank Group 2: Increased area/power

- **3D-stacked DRAM**
  - High bandwidth with Through-Silicon Vias (TSVs)

- **Memory Layers**
  - Dedicated Logic Layer
4. Need for Lower Access Latency: Performance

- New DRAM types often increase access latency in order to provide more banks, higher throughput
- Many applications can’t make up for the increased latency
  - Especially true of common OS routines (e.g., file I/O, process forking)

- A variety of desktop/scientific, server/cloud, GPGPU applications

Several applications don’t benefit from more parallelism
Key Takeaways

1. DRAM latency remains a critical bottleneck for many applications

2. Bank parallelism is not fully utilized by a wide variety of our applications

3. Spatial locality continues to provide significant performance benefits if it is exploited by the memory subsystem

4. For some classes of applications, low-power memory can provide energy savings without sacrificing significant performance
Conclusion

- Manufacturers are developing many new types of DRAM
  - DRAM limits performance, energy improvements: new types may overcome some limitations
  - Memory systems now serve a very diverse set of applications: can no longer take a one-size-fits-all approach
  - Difficult to intuitively determine which DRAM–workload pair works best

- We perform a wide-ranging experimental study to uncover the combined behavior of workloads, DRAM types
  - 115 prevalent/emerging applications and multiprogrammed workloads
  - 9 modern DRAM types

- 12 key observations on DRAM–workload behavior

Open-source tools: https://github.com/CMU-SAFARI/ramulator
Full paper: https://arxiv.org/pdf/1902.07609
For More Information…

  - [Preliminary arXiv Version]
  - [Abstract]
  - [Slides (pptx) (pdf)]
  - [MemBen Benchmark Suite]
  - [Source Code for GPGPUSim-Ramulator]

Demystifying Complex Workload–DRAM Interactions: An Experimental Study

Saugata Ghose†
Tianshi Li†
Nastaran Hajinazar‡†
Damla Senol Cali†
Onur Mutlu§†

†Carnegie Mellon University
‡Simon Fraser University
§ETH Zürich
BlockHammer Study in 2021

- A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Ogun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,

"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Intel Hardware Security Academic Awards Short Talk Slides (pptx) (pdf)]
[Talk Video (22 minutes)]
[Short Talk Video (7 minutes)]
[Intel Hardware Security Academic Awards Short Talk Video (2 minutes)]
[BlockHammer Source Code]

Intel Hardware Security Academic Award Finalist (one of 4 finalists out of 34 nominations)
Summary: BlockHammer

- BlockHammer is the first work to practically enable throttling-based RowHammer mitigation.

- BlockHammer is implemented in the memory controller (no proprietary information of / no modifications to DRAM chips).

- BlockHammer is both scalable with worsening RowHammer and compatible with commodity DRAM chips.

- BlockHammer is open-source along with six state-of-the-art mechanisms: [https://github.com/CMU-SAFARI/BlockHammer](https://github.com/CMU-SAFARI/BlockHammer)

Source
BlockHammer

Aggressive memory density scaling causes modern DRAM devices to suffer from RowHammer, a phenomenon where rapidly activating a DRAM row can cause bit-flips in physically-nearby rows. Recent studies [1, 2, 3] demonstrate that modern DRAM chips, including chips previously marketed as RowHammer-safe, are even more vulnerable to RowHammer than older chips. Many works show that attackers can exploit RowHammer bit-flips to reliably mount system-level attacks to escalate privilege and leak private data. Therefore, it is critical to ensure RowHammer-safe operation on all DRAM-based systems.

Unfortunately, state-of-the-art RowHammer mitigation mechanisms face two major challenges. First, they incur increasingly higher performance and/or area overheads when applied to more vulnerable DRAM chips. Second, they require either proprietary information about or modifications to the DRAM chip design. In this paper, we show that it is possible to efficiently and scalably prevent RowHammer bit-flips without knowledge of or modification to DRAM internals.

We introduce BlockHammer, a low-cost, effective, and easy-to-adopt RowHammer mitigation mechanism that overcomes the two key challenges by selectively throttling memory accesses that could otherwise cause RowHammer bit-flips. The key idea of BlockHammer is to (1) track row activation rates using area-efficient Bloom filters and (2) use the tracking data to ensure that no row is ever activated rapidly enough to induce RowHammer bit-flips. By doing so, BlockHammer (1) makes it impossible for a RowHammer bit-flip to occur and (2) greatly reduces a RowHammer attack’s impact on the performance of co-running benign applications.

Compared to state-of-the-art RowHammer mitigation mechanisms, BlockHammer provides competitive performance and energy when the system is not under a RowHammer attack and significantly better performance and energy when the system is under attack.

The full paper is published in HPCA 2021 and the pdf is available online: [arXiv:2102.05981](https://arxiv.org/abs/2102.05981)

Citation

Please cite our full HPCA 2021 paper if you find this repository useful.


Getting Started

This repository has two subdirectories. Please refer to each subdirectory on reproducing results.

- **Ranulator Model**: This subdirectory includes an extended version of [Ranulator](https://github.com/CMU-SAFARI/BlockHammer) with a RowHammerDefense class, which implements BlockHammer along with six state-of-the-art RowHammer mitigation mechanisms:

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Reference</th>
</tr>
</thead>
</table>

- **RTL Model**: This subdirectory includes RTL implementation of the counters and buffers used in BlockHammer.

[https://github.com/CMU-SAFARI/BlockHammer](https://github.com/CMU-SAFARI/BlockHammer)
Many Other Ideas Evaluated w/ Ramulator

Key Idea and Applications

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

- **LISA is a versatile substrate** → new applications
  - Fast bulk data copy: Copy latency 1.363ms → 0.148ms (9.2x)
    → 66% speedup, -55% DRAM energy
  - In-DRAM caching: Hot data access latency 48.7ns → 21.5ns (2.2x)
    → 5% speedup
  - Fast precharge: Precharge latency 13.1ns → 5.0ns (2.6x)
    → 8% speedup
Ramulator for Processing in Memory
Simulation Infrastructure 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-pim)
  - [https://github.com/CMU-SAFARI/ramulator](https://github.com/CMU-SAFARI/ramulator)
  - [Source Code for Ramulator-PIM](https://github.com/CMU-SAFARI/ramulator-pim)
Ramulator for PIM


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

Gagandeep Singh$^{a,c}$, Juan Gómez-Luna$^b$, Stefano Corda$^{a,c}$, Sander Stujik$^a$, Giovanni Mariani$^c$, Onur Mutlu$^b$, Geraldo F. Oliveira$^b$, Henk Corporaal$^a$

$^a$Eindhoven University of Technology
$^b$ETH Zürich
$^c$IBM Research - Zurich
Ramulator Project Course
Exploration of Emerging Memory Systems (Spring/Fall 2022)

- **Fall 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=ramulator

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=ramulator

- **Youtube Livestream (Spring 2022):**
  - https://www.youtube.com/watch?v=aM-llXRQd3s&list=PL5Q2soXY2Zi_TImLGw_Z8hBo29Z5ZApqV

- **Bachelor’s course**
  - Elective at ETH Zurich
  - Introduction to memory system simulation
  - Tutorial on using Ramulator
  - C++
  - Potential research exploration

https://www.youtube.com/onurmutlulectures
Some Other Useful Simulators
Many Simulators for Many Things

- **gem5** full-system multi-core simulation
- **MQSim** for SSD simulation
- **DiskSim** for Hard Disk simulation
- **DAMOV-Sim** for Processing-near-Memory simulation
- **Sniper** for fast Processor Simulation
- **Scarab** for detailed Microarchitectural Simulation
- **Simics, Bochs, QEMU** for full-system functional simulation
- ...

Or, develop your own simulator for your purpose...
DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland
DAMOV is Open Source

• We open-source our benchmark suite and our toolchain.
DAMOV is Open Source

- We open-source our benchmark suite and our toolchain

Get DAMOV at:

https://github.com/CMU-SAFARI/DAMOV

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including BWA, Chai, Darknet, GASE, Hardware Effects, Hashjoin, HPCC, HPCG, Ligra, PARSEC, Parboil, PolyBench, Phoenix, Rodinia, SPLASH-2, STREAM.
More on DAMOV Analysis Methodology & Workloads

**Step 3: Memory Bottleneck Classification (2/2)**

- **Goal:** identify the specific sources of data movement bottlenecks

**SAFARI Live Seminar: DAMOV: A New Methodology & Benchmark Suite for Data Movement Bottlenecks**

352 views • Streamed live on Jul 22, 2021

https://www.youtube.com/watch?v=GWideVyo0nM&list=PL5Q2soXY2ZI_tOTAYm--dYByNPL7JhwR9&index=3
PIM Course (Spring 2022)

- **Spring 2022 Edition:**
  - [https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory](https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory)

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi841fUYYUK9EsXKhQKRPyX](https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi841fUYYUK9EsXKhQKRPyX)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - Processing-in-Memory lectures
  - Hands-on research exploration
  - Many research readings

[https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices

Arash Tavakkol†, Juan Gómez-Luna†, Mohammad Sadrosadati†, Saugata Ghose‡, Onur Mutlu‡‡

†ETH Zürich
‡Carnegie Mellon University


[Slides (pptx) (pdf)]
[Source Code]
Solid-State Drives Course (Spring 2022)

- **Spring 2022 Edition:**

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=_q4r71DsY4&list=PL5Q2soXY2Zi8vabcse1kL22DEcgMI2RAq](https://www.youtube.com/watch?v=_q4r71DsY4&list=PL5Q2soXY2Zi8vabcse1kL22DEcgMI2RAq)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - SSD Basics and Advanced Topics
  - Hands-on research exploration
  - Many research readings

[https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
Many More Simulators …

SAFARI Research Group at ETH Zurich and Carnegie Mellon University
Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

Overview  Repositories 71  Projects  Packages  People 13

Pinned

- **ramulator** (Public)
  A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WIOx, HBMx, and various academic proposals. Described in the...
  - C++ 356  167

- **MQSim** (Public)
  MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implement...
  - C++ 161  97

- **prim-benchmarks** (Public)
  PrIM (Processing-In-Memory benchmarks) is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publi...
  - C 64  24

- **rowhammer** (Public)
  - C 195  41

- **SparseP** (Public)
  SparseP is the first open-source Sparse Matrix Vector Multiplication (SpMV) software package for real-world Processing-In-Memory (PIM) architectures. SparseP is developed to evaluate and characteri...
  - C 37  7

- **SoftMC** (Public)
  SoftMC is an experimental FPGA-based memory controller design that can be used to develop tests for DDR3 SODIMMs using a C++ based API. The design, the interface, and its capabilities and limitatio...
  - Verilog 88  27

https://github.com/CMU-SAFA
What We Discussed Is Applicable to Simulation in Other Domains
Case Study:
COVID-19 Spread
Modeling and Prediction
COVIDHunter: COVID-19 Pandemic Wave Prediction and Mitigation via Seasonality Aware Modeling

Mohammed Alser*, Jeremie S. Kim, Nour Almadhoun Alser, Stefan W. Tell and Onur Mutlu

Department of Information Technology and Electrical Engineering (I-TEC), ETH Zurich, Zurich, Switzerland

Early detection and isolation of COVID-19 patients are essential for successful implementation of mitigation strategies and eventually curbing the disease spread. With a limited number of daily COVID-19 tests performed in every country, simulating the COVID-19 spread along with the potential effect of each mitigation strategy currently remains one of the most effective ways in managing the healthcare system and guiding policy-makers. We introduce COVIDHunter, a flexible and accurate COVID-19 outbreak simulation model that evaluates the current mitigation measures that are applied to a region, predicts COVID-19 statistics (the daily number of cases, hospitalizations, and deaths), and provides suggestions on what strength the upcoming mitigation measure should be. The key idea of COVIDHunter is to quantify the spread of COVID-19 in a geographical region by simulating the average number of new infections caused by an infected person considering the effect of external factors, such as environmental conditions (e.g., climate, temperature, humidity), different variants of concern, vaccination rate, and mitigation measures. Using Switzerland as a case study, COVIDHunter estimates that we are experiencing a deadly new wave that will peak on 26 January 2022, which is very similar in numbers to the wave we had in February 2020. The policy-makers have only one choice that is to increase the strength of the currently applied mitigation measures for 30 days. Unlike existing models, the COVIDHunter model accurately monitors and predicts the daily number of cases, hospitalizations, and deaths due to COVID-19. Our model is flexible to configure and simple to modify for modeling different scenarios under different environmental conditions and mitigation measures. We release the source code of the COVIDHunter implementation at https://github.com/CMU-SAFARI/COVIDHunter and show how to flexibly configure our model for any scenario and easily extend it for different measures and conditions than we account for.

Keywords: epidemiological modeling, COVID-19 outbreak simulation, seasonal epidemic, outbreak prevention and control, vaccination

INTRODUCTION

Coronavirus disease 2019 (COVID-19) is caused by SARS-CoV-2 virus, which has rapidly spread to nearly every corner of the globe and has been declared a pandemic in March 2020 by the World Health Organization (WHO) (1). As of November 2021, only about 40% of the entire world population is fully vaccinated and their protection wanes after a few months (2). Until an effective drug or vaccination is made widely available to everyone, early detection and isolation of COVID-19 patients are essential for successful implementation of mitigation strategies and eventually curbing the disease spread.
COVID-19 Measures: Evaluation Methods

How do we assess how an idea will affect a target metric X?

A variety of evaluation methods are available:

- Theoretical proof
- Analytical modeling/estimation
- Simulation (at varying degrees of abstraction and accuracy)
- Prototyping with a real system (e.g., FPGAs)
- Real implementation
Simulating & Predicting COVID-19 Spread

- An architect is in part a dreamer, a creator

- Simulation is a key tool of the architect
  - Allows the evaluation & understanding of non-existent systems

- Simulation enables
  - The exploration of many dreams
  - A reality check of the dreams
  - Deciding which dream is better

- Simulation also enables
  - The ability to fool yourself with false dreams
Simulating & Predicting COVID-19 Spread

To our knowledge, there is currently no model capable of accurately monitoring the current epidemiological situation and predicting future scenarios while considering a reasonably low number of parameters and accounting for the effects of environmental conditions (Table 1).

Our goal in this work is to develop and validate such a COVID-19 outbreak simulation model. To this end, we introduce COVIDHunter, a simulation model that evaluates the current mitigation measures (i.e., non-pharmaceutical intervention or NPI) that are applied to a region and provides insight into what strength the upcoming mitigation measure should be and for how long it should be applied, while considering the potential effect of environmental conditions. Our model accurately forecasts the

<table>
<thead>
<tr>
<th>Model</th>
<th>Open source</th>
<th>Well documented*</th>
<th>Accounting for seasonality</th>
<th>Low number of parameters</th>
<th>Reported COVID-19 statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>COVIDHunter (this work)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓ (R, cases, hospitalizations, and deaths)</td>
</tr>
<tr>
<td>IBZ (11)</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓ (only R)</td>
</tr>
<tr>
<td>LSHTM (7)</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓ (only cases)</td>
</tr>
<tr>
<td>ICL (9)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓ (R, cases, hospitalizations, and deaths)</td>
</tr>
<tr>
<td>IHME (10)</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓ (cases, hospitalizations, and deaths)</td>
</tr>
</tbody>
</table>

*The available packages are configured only for the IHME infrastructure. # Based on the documentation available on each model’s GitHub page (all models are available on GitHub).
Simulating & Predicting COVID-19 Spread

**FIGURE 4** Observed (officially reported) and expected number of COVID-19 cases in Switzerland during the years 2020 and 2021. We calculate the expected number of cases based on both the hospitalizations-to-cases and deaths-to-cases ratios for the second wave. We assume two certainty rate levels of 50 and 100%.

Epidemiological Situation in Switzerland Using COVIDHunter

We use Switzerland as a use-case for all the experiments. However, our model is not limited to any specific region as the parameters of COVIDHunter are completely configurable. Using COVIDHunter on 20 November 2021, we make four key observations:

**Prediction**

1. The spread of COVID-19 in Switzerland is still active as the reproduction number (R) is still greater than 1.0. The R number value will remain greater than 1 until at least February 2022.

2. COVIDHunter estimates that we are experiencing a deadly new wave that will peak on the last week of January 2022, which is very similar in numbers to the wave we had in February 2020.

3. The policy-makers have only one choice that is increasing the strength of the currently applied mitigation measures for 30 days. Relaxing the mitigation measures should not be an option before at least February 2022 as it would increase exponentially the number of cases, hospitalizations, and deaths by 5x.

4. COVIDHunter forecasts the effect of relaxing the current mitigation measures on November 20, 2021 on the daily maximum number of COVID-19 cases, hospitalizations, and deaths as follows:

<table>
<thead>
<tr>
<th>Strengths of the mitigation measures during November-December 2021</th>
<th>0.2</th>
<th>0.3</th>
<th>0.40</th>
<th>0.50</th>
<th>0.60</th>
<th>0.70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mitigation measures with similar strength were applied on</td>
<td>29 February - 11 March 2020</td>
<td>1 October - 20 October 2020</td>
<td>28 February - 15 March 2020</td>
<td>24 June - 20 August 2020</td>
<td>12-29 October 2020</td>
<td>28 November - 22 December 2020</td>
</tr>
<tr>
<td>Predicted daily number of cases</td>
<td>7822-43518</td>
<td>5706-22413</td>
<td>3401-10714</td>
<td>542-1'310</td>
<td>70-410</td>
<td>3-104</td>
</tr>
<tr>
<td>Predicted daily number of hospitalizations</td>
<td>124-693</td>
<td>90-357</td>
<td>54-170</td>
<td>9-21</td>
<td>1-7</td>
<td>1-2</td>
</tr>
<tr>
<td>Predicted daily number of deaths</td>
<td>38-216</td>
<td>28-112</td>
<td>16-53</td>
<td>3-7</td>
<td>1-3</td>
<td>1-2</td>
</tr>
</tbody>
</table>

**Reality**

![Graph showing the predicted and actual cases and deaths in Switzerland](https://mealser.github.io/COVIDHunter/index.html)
Recall: Goals in Simulation

- **Explore the design space quickly** and see what you want to
  - potentially implement in a next-generation platform
  - propose as the next big idea to advance the state of the art
  - the goal is mainly to see relative effects of design decisions

- **Match the behavior of an existing system** so that you can
  - debug and verify it at high accuracy
  - propose small tweaks to the design that can make a difference
  - the goal is very high accuracy

- Other goals in-between:
  - **Refine the explored design space** without going into full detailed modeling
  - **Gain confidence in your design decisions** made by higher-level design space exploration
Recall: Tradeoffs in Simulation

- Three metrics to evaluate a simulator
  - Speed
  - Flexibility
  - Accuracy

- Speed: How fast the simulator runs (xIPS, xCPS, slowdown)
- Flexibility: How quickly one can modify the simulator to evaluate different algorithms and design choices?
- Accuracy: How accurate the performance (energy) numbers the simulator generates are vs. a real design (Simulation error)

- The relative importance of these metrics varies depending on where you are in the design process (what your goal is)
Recall: Trading Off Speed, Flexibility, Accuracy

- **Speed & flexibility affect:**
  - How quickly you can make design tradeoffs

- **Accuracy affects:**
  - How good your design tradeoffs *may* end up being
  - How fast you can build your simulator (simulator design time)

- **Flexibility also affects:**
  - How much human effort you need to spend modifying the simulator

- You can **trade off between the three to achieve design exploration and decision goals**
Recall: High-Level Simulation

- **Key Idea:** Raise the abstraction level of modeling to **give up some accuracy to enable speed & flexibility** (and quick simulator design)

- **Advantage**
  + Can still make the right tradeoffs, and can do it quickly
    + All you need is modeling the key high-level factors, you can omit corner case conditions
    + All you need is to get the “relative trends” accurately, not exact performance numbers

- **Disadvantage**
  -- Opens up the possibility of potentially wrong decisions
    -- How do you ensure you get the “relative trends” accurately?
Recall: Simulation as Progressive Refinement

- High-level models (Abstract, C)
- ...
- Medium-level models (Less abstract)
- ...
- Low-level models (RTL with everything modeled)
- ...
- Real design

As you refine (go down the above list)
- Abstraction level reduces
- Accuracy (hopefully) increases (not necessarily, if not careful)
- Flexibility reduces; Speed likely reduces except for real design
- You can loop back and fix higher-level models
Recall: Making The Best of Architecture

- A good architect is comfortable at all levels of refinement
  - Including the extremes

- A good architect knows when to use what type of simulation
  - And, more generally, what type of evaluation method

Recall: A variety of evaluation methods are available:
- Theoretical proof
- Analytical modeling
- Simulation (at varying degrees of abstraction and accuracy)
- Prototyping with a real system (e.g., FPGAs)
- Real implementation