Computer Architecture

Lecture 22a: Simulation

Dr. Mohammad Sadrosadati
Prof. Onur Mutlu
ETH Zürich

Fall 2023

8 December 2023

Simulating (Memory) Systems

Evaluating New Ideas for New (Memory) Architectures

Potential 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

An Example Prototyping Platform

Real Processing Using Memory Prototype

- End-to-end RowClone & TRNG using off-the-shelf DRAM chips
- Idea: Violate DRAM timing parameters to mimic RowClone

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

Ataberk Olgun[§†] Juan Gómez Luna[§] Konstantinos Kanellopoulos[§] Behzad Salami[§*]
Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

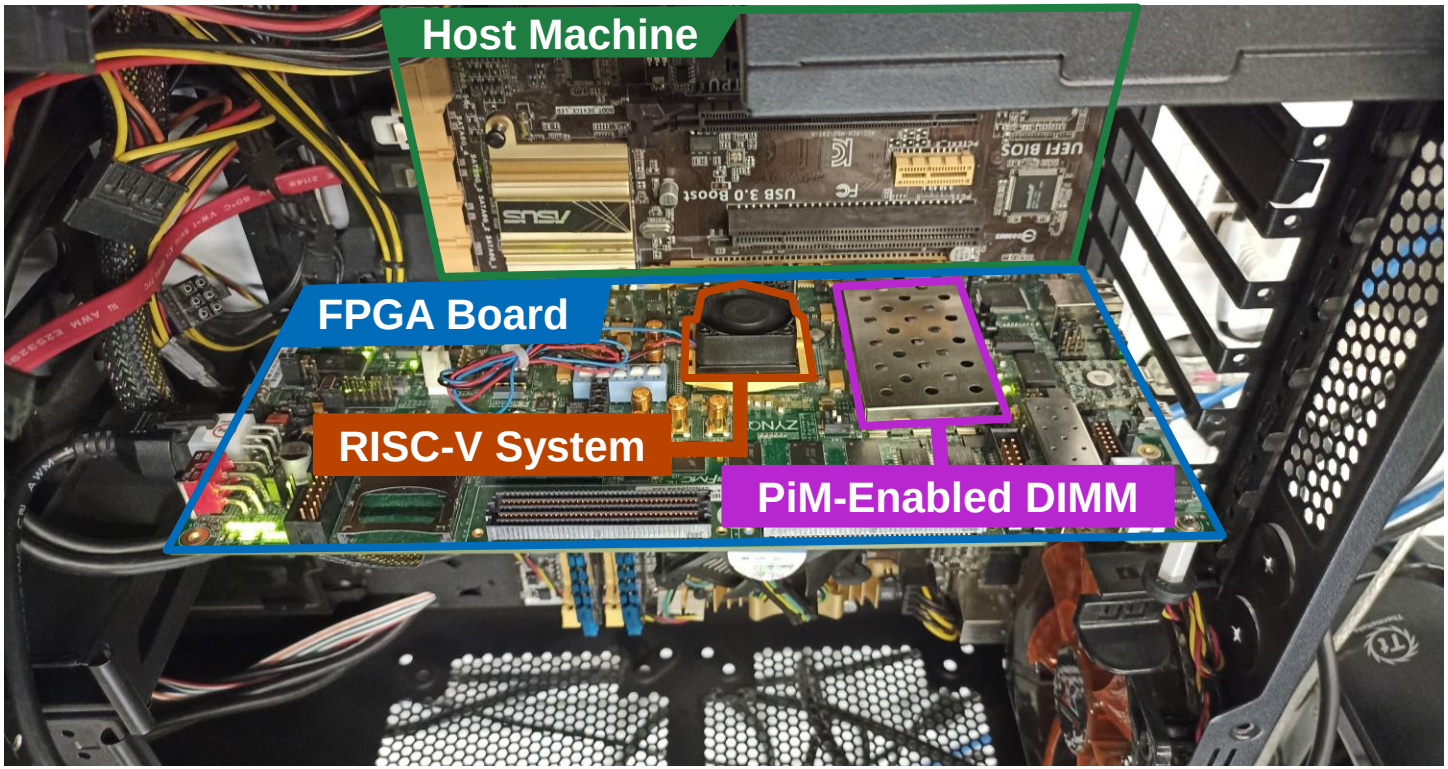
[§]ETH Zürich [†]TOBB ETÜ [*]BSC

https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing Using Memory Prototype



https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing Using Memory Prototype

README.md

Building a PiDRAM Prototype

To build PiDRAM's prototype on Xilinx ZC706 boards, developers need to use the two sub-projects in this directory. `fpga-zynq` is a repository branched off of UCB-BAR's fpga-zynq repository. We use `fpga-zynq` to generate rocket chip designs that support end-to-end DRAM PuM execution. `controller-hardware` is where we keep the main Vivado project and Verilog sources for PiDRAM's memory controller and the top level system design.

Rebuilding Steps

1. Navigate into `fpga-zynq` and read the README file to understand the overall workflow of the repository
   - Follow the readme in `fpga-zynq/rocket-chip/riscv-tools` to install dependencies
2. Create the Verilog source of the rocket chip design using the `ZynqCopyFPGAConfig`
   - Navigate into zc706, then run `make rocket CONFIG=ZynqCopyFPGAConfig -j<number of cores>`
3. Copy the generated Verilog file (should be under zc706/src) and overwrite the same file in `controller-hardware/source/hdl/impl/rocket-chip`
4. Open the Vivado project in `controller-hardware/Vivado_Project` using Vivado 2016.2
5. Generate a bitstream
6. Copy the bitstream (system_top.bit) to `fpga-zynq/zc706`
7. Use the `./build_script.sh` to generate the new `boot.bin` under `fpga-images-zc706`, you can use this file to program the FPGA using the SD-Card
   - For details, follow the relevant instructions in `fpga-zynq/README.md`

You can run programs compiled with the RISC-V Toolchain supplied within the `fpga-zynq` repository. To install the toolchain, follow the instructions under `fpga-zynq/rocket-chip/riscv-tools`.

Generating DDR3 Controller IP sources

We cannot provide the sources for the Xilinx PHY IP we use in PiDRAM's memory controller due to licensing issues. We describe here how to regenerate them using Vivado 2016.2. First, you need to generate the IP RTL files:

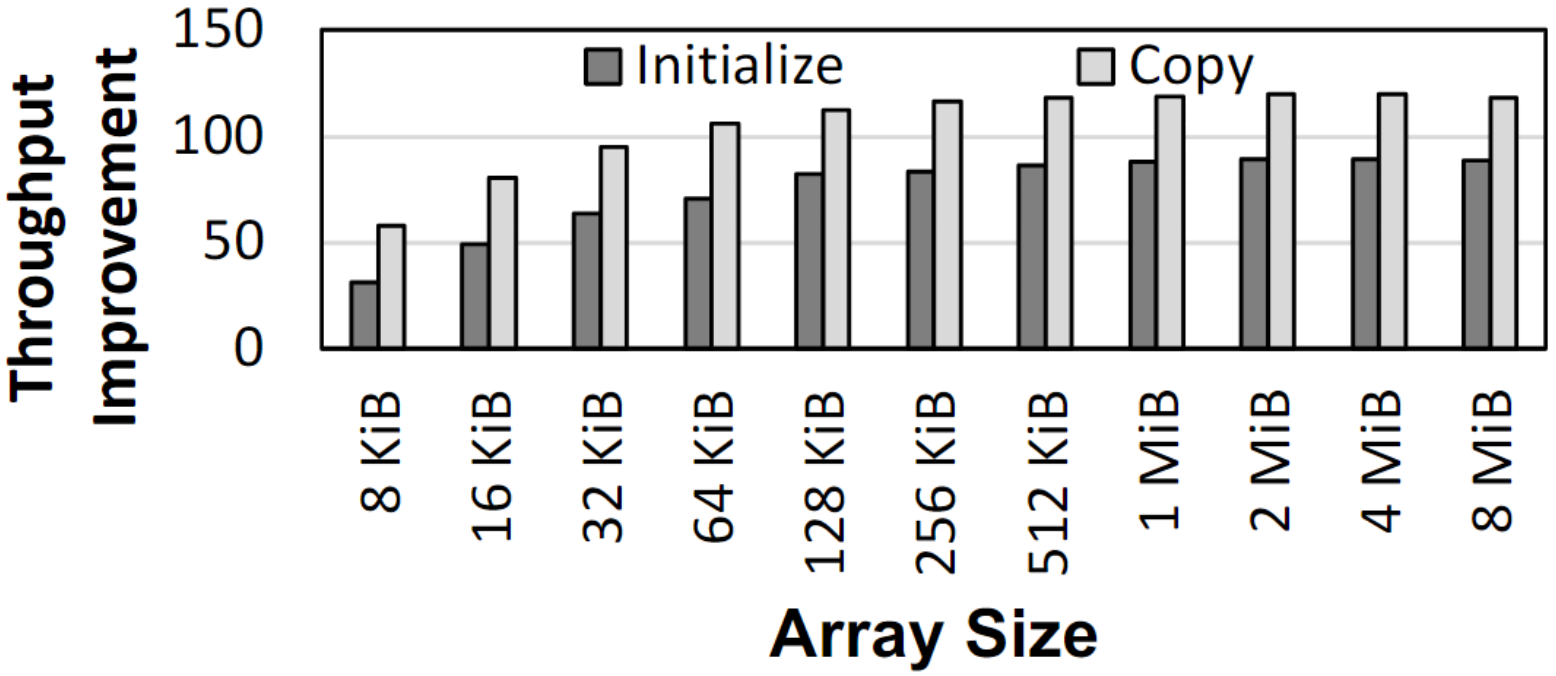
1- Open IP Catalog
2- Find "Memory Interface Generator (MIG 7 Series)" IP and double click

https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Microbenchmark Copy/Initialization Throughput



In-DRAM Copy and Initialization improve throughput by 119x and 89x

The Difficulty in Architectural Evaluation

- The answer is usually workload dependent
  - E.g., think caching
  - E.g., think pipelining
  - E.g., think any idea we talked about (RAIDR, SALP, TL-DRAM, …)
- Workloads change
- System has many design choices and parameters
  - Architect needs to decide many ideas and many parameters for a design
  - Not easy to evaluate all possible combinations!
- System parameters may change

Simulation: The Field of Dreams

Dreaming and Reality

- 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

Why High-Level Simulation?

- Problem: RTL simulation is intractable for design space exploration → too time consuming to design and evaluate
  - Especially over a large number of workloads
  - Especially if you want to predict the performance of a good chunk of a workload on a particular design
  - Especially if you want to consider many design choices
    - Cache size, associativity, block size, algorithms
    - Memory control and scheduling algorithms
    - In-order vs. out-of-order execution
    - Reservation station sizes, ld/st queue size, register file size, …
    - …
- Goal: Explore design choices quickly to see their impact on the workloads we are designing the platform for

Different 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 cycle-level accuracy
  - propose small tweaks to the design that can make a difference in performance or energy
  - the goal is very high accuracy
- Other goals in-between:
  - Refine the explored design space without going into a full detailed, cycle-accurate design
  - Gain confidence in your design decisions made by higher-level design space exploration

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)

Trading Off Speed, Flexibility, Accuracy

- Speed & flexibility affect:
  - How quickly you can make design tradeoffs/decisions
- Accuracy affects:
  - How good your design tradeoffs/decisions might 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

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?

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

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

Some General Issues in Architectural Simulation

Goal of Simulation Dictates Many Things

- Drives many design choices and what simulator to build/use
- Accuracy, flexibility, speed
- Many possible goals (as discussed earlier)
  - Entire system performance estimation
  - Component performance estimation (cache, SSD, memory, …)
  - Profiling for statistics
  - ...

What, How, Where, When

- What do you simulate?
  - Component(s) or full system
  - Program(s) or trace(s)
- How do you simulate it?
  - Many choices: Functional vs. timing, event-driven vs. cycle-by-cycle, state maintenance & recovery, …
- Where do you simulate (each thing you want to simulate)?
  - Software vs. hardware-accelerated
- When do you do things in simulation?
  - Oracle information vs. execute-like-a-real-machine

Simulator Inputs and Outputs: Generalized

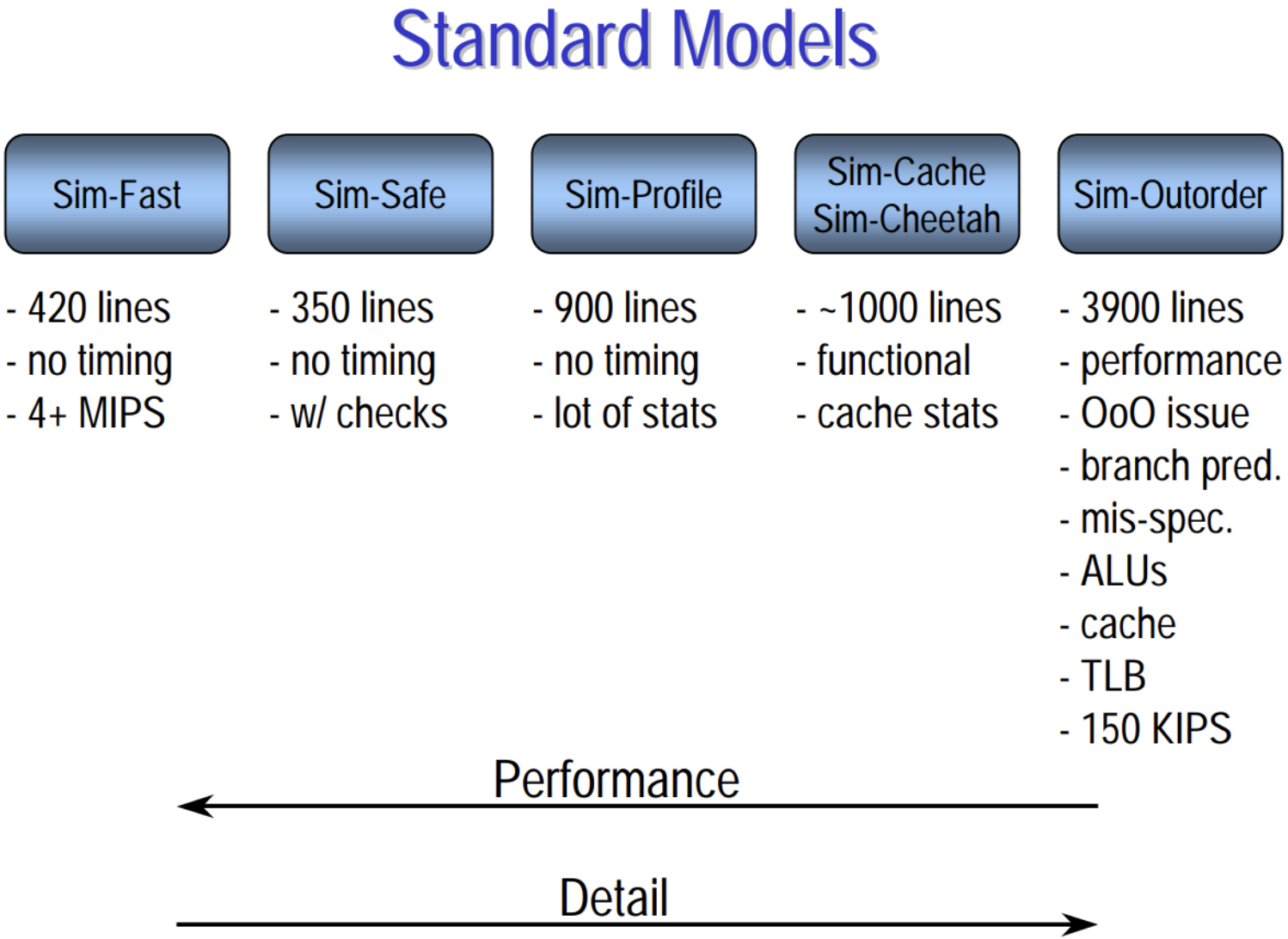
- Inputs
  - Program binary (can be multiple program binaries)
  - System state/checkpoint (e.g., including OS, memory, devices, storage, network, …)

- Outputs
  - Functional
    - Program results
    - Statistics about functional execution
  - Timing related
    - Execution time of each program
    - System throughput of all programs
    - Statistics about timing and performance events

Some General Issues in Architecture Simulation

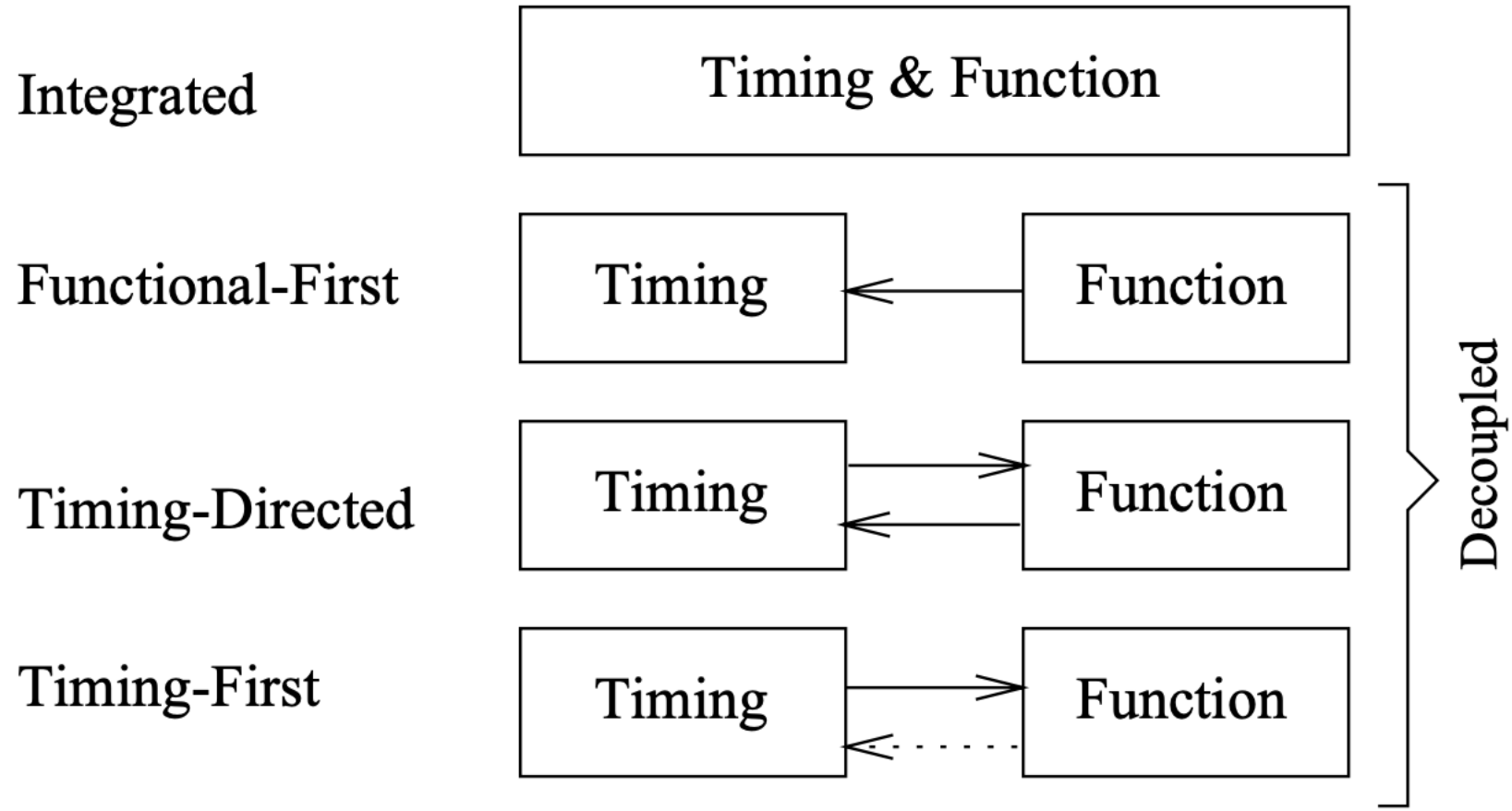
- Functional vs. Timing Simulation
  - Purpose: When/why functional vs. timing
  - Integration of functional and timing simulation
- Full-System vs. Component Simulation
  - Purpose: When/why full system vs. component(s)
  - What should be modeled? OS, VMs, memory allocator?
- Complete Workload vs. Sampling Based Simulation
  - How do you form workloads to simulate; what parts to simulate?
- Warm-Up of Simulated Structures
  - Steady-state vs. cold-start (e.g., caches, branch predictor tables)

An Old Example: Simplescalar Func. vs Timing



https://web.eecs.umich.edu/~taustin/papers/simple-tutorial-2up.pdf

Functional vs. Timing: Many Choices



Arrows indicate inter-simulator interactions per committed instruction.

Figure 1. Simulator Organizations

Mauer et al., "Full-system timing-first simulation," SIGMETRICS 2002.

An Example Functional-First Model

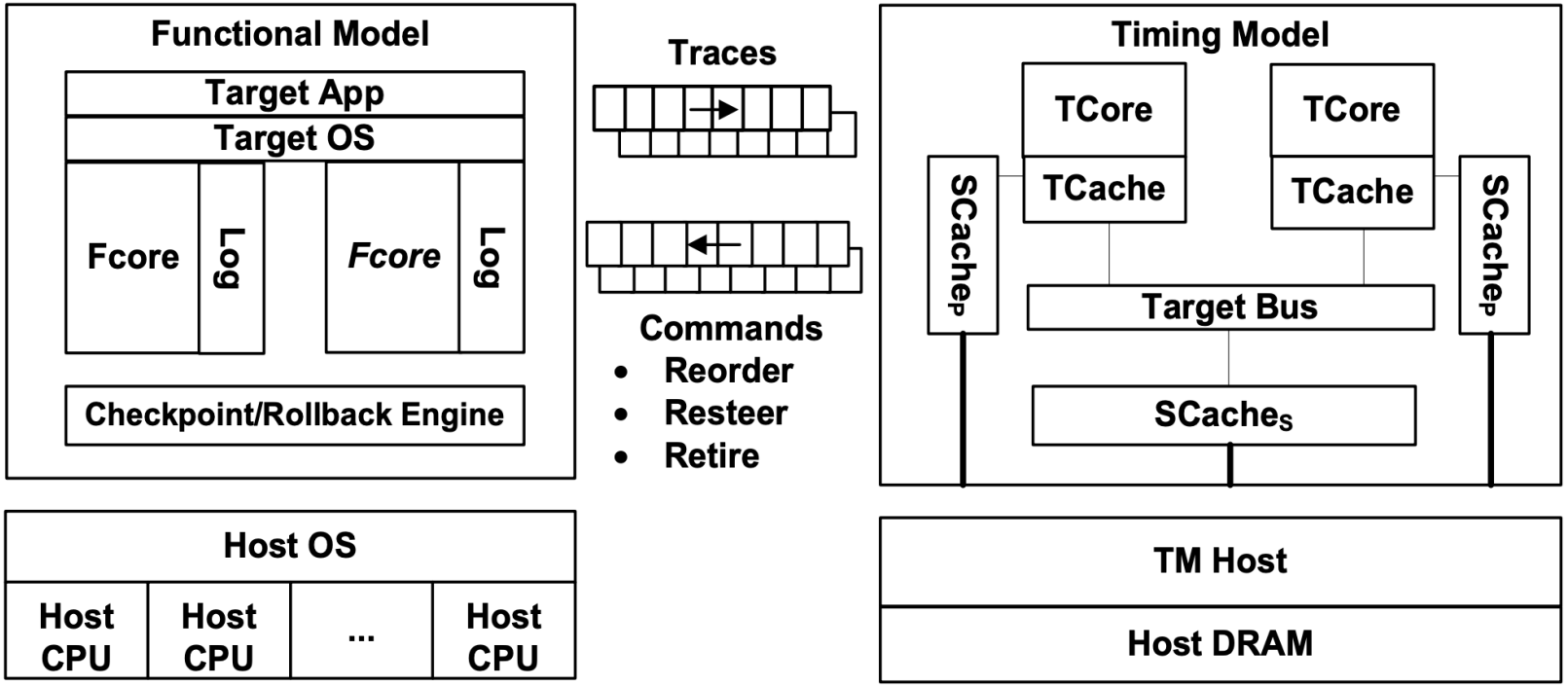


Fig. 1. FAST Simulator of a Parallel Computer System

Some General Issues in Architecture Simulation

- Validation of Accuracy
  - Functional accuracy vs. timing accuracy
  - Online vs. offline validation
- Simulation Acceleration
  - via Hardware Support (e.g., FPGAs)
  - via Software Methods (e.g., Memoization)
  - via Better Software Engineering
- Trace-driven vs. Execution-driven Simulation
  - Captured trace drives what is modeled in simulation (and timing)
  - Simulator itself emulates program execution and determines what is executed (and timing)
  - Affects what can be modeled (easily): e.g., wrong path

Some General Issues in Architecture Simulation

- Execute-at-Frontend vs. Execute-at-Execute
  - Is execution done only in functional simulator or in the timing simulator as well?
  - Timing-dependent execution becomes harder if execution is done only in the frontend, in the timing simulator
  - Examples: value prediction of L2 misses
- State Maintenance and Recovery
  - Modeling of mispredicted execution (wrong path, wrong values, …)
- Event-driven vs. Cycle-by-cycle Polling
  - One of many simulator design choices

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

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

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

Simulator (clang -O3)	*Cycles (10^6)*		*Runtime (sec.)*		*Req/sec (10^3)*		*Memory (MB)*
	Random	*Stream*	*Random*	*Stream*	*Random*	*Stream*	
Ramulator	652	411	752	249	133	402	2.1
DRAMSim2	645	413	2,030	876	49	114	1.2
USIMM	661	409	1,880	750	53	133	4.5
DrSim	647	406	18,109	12,984	6	8	1.6
NVMain	666	413	6,881	5,023	15	20	4,230.0

Table 3. Comparison of five simulators using two traces

Case Study: Comparison of DRAM Standards

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

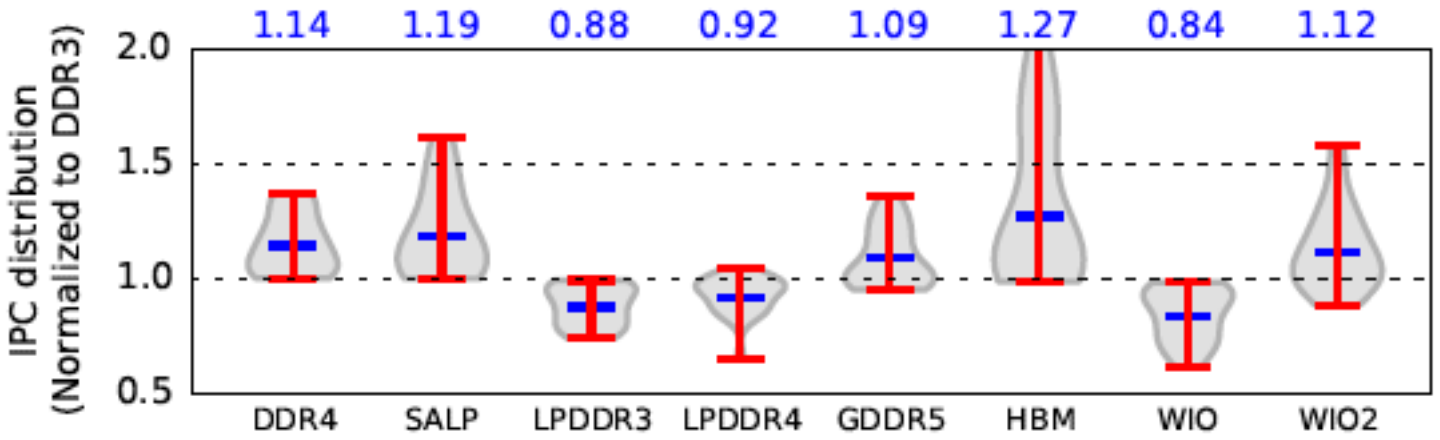


Figure 2. Performance comparison of DRAM standards

Across 22 workloads, simple CPU model

Ramulator Paper and Source Code

- Yoongu Kim, Weikun Yang, and Onur Mutlu,
  "Ramulator: A Fast and Extensible DRAM Simulator"
  IEEE Computer Architecture Letters (***CAL***), March 2015.
  [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 Kim[1] Weikun Yang[1,2] Onur Mutlu[1]
[1]Carnegie Mellon University [2]Peking University

Ramulator: Free & Open Source

README.md

Ramulator: A DRAM Simulator

Ramulator is a fast and cycle-accurate DRAM simulator [1, 2] that supports a wide array of commercial, as well as academic, DRAM standards:

- DDR3 (2007), DDR4 (2012)
- LPDDR3 (2012), LPDDR4 (2014)
- GDDR5 (2009)
- WIO (2011), WIO2 (2014)
- HBM (2013)
- SALP [3]
- TL-DRAM [4]
- RowClone [5]
- DSARP [6]

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

> Y. Kim, W. Yang, O. Mutlu. "**Ramulator: A Fast and Extensible DRAM Simulator**". In *IEEE Computer Architecture Letters*, March 2015.

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

> S. Ghose, T. Li, N. Hajinazar, D. Senol Cali, O. Mutlu. "**Demystifying Complex Workload–DRAM Interactions: An Experimental Study**". In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, June 2019 (slides). In *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, 2019.

[1] Kim et al. *Ramulator: A Fast and Extensible DRAM Simulator.* IEEE CAL 2015.
[2] Ghose et al. *Demystifying Complex Workload–DRAM Interactions: An Experimental Study.* SIGMETRICS 2019.
[3] Kim et al. *A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM.* ISCA 2012.
[4] Lee et al. *Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture.* HPCA 2013.
[5] Seshadri et al. *RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization.* MICRO 2013.
[6] Chang et al. *Improving DRAM Performance by Parallelizing Refreshes with Accesses.* HPCA 2014.

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
- 0x4cbd56c0 W
- ...

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

Ramulator: Integration with Other Simulators

README.md

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
- 0x4cbd56c0 W
- ...

2. **CPU Trace Driven:** 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-cpuinst> <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-cpuinst> <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 eviction caused by the read request before it.

3. **gem5 Driven:** 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.

[7] The gem5 Simulator System.
[8] Ghose et al. *What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study.* SIGMETRICS 2018.
[9] Chandrasekar et al. *DRAMPower: Open-Source DRAM Power & Energy Estimation Tool.* IEEE CAL 2015.

Getting Started

Ramulator requires a C++11 compiler (e.g., `clang++`, `g++-5`).

1. **Memory Trace Driven**

```
$ 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 Driven**

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

Ramulator: Reproducibility

README.md

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/~rajeev/jwac12): `test_ddr3.py` lines 54-55
- DrSim (http://lph.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 to be executed in saturation mode (always filling up the request queues when possible).

All five simulators were configured using the same parameters:

- DDR3-1600K (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 `cputraces` 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-N-rank-M.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

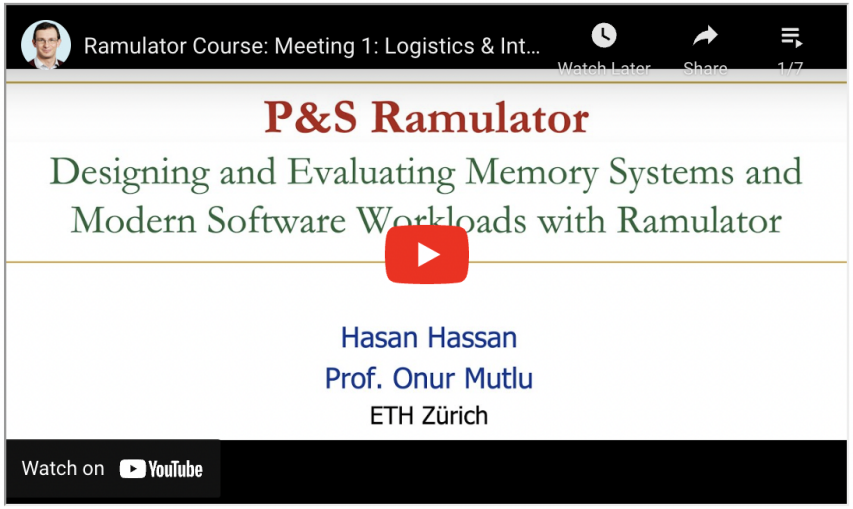
Ramulator Project Course

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_TlmLGw_Z8hBo2925ZApqV
- Bachelor's course
  - Elective at ETH Zurich
  - Introduction to memory system simulation
  - Tutorial on using Ramulator
  - C++
  - Potential research exploration

Lecture Video Playlist on YouTube

Lecture Playlist



2022 Meetings/Schedule (Tentative)

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	09.03 Wed.	Video	M1: Logistics & Intro to Simulating Memory Systems Using Ramulator (PDF) (PPT)		HW0
W2	16.03 Fri.	Video	M2: Tutorial on Using Ramulator (PDF) (PPT)		
W3	25.02 Fri.	Video	M3: BlockHammer (PDF) (PPT)		
W4	01.04 Fri.	Video	M4: CLR-DRAM (PDF) (PPT)		
W5	08.04 Fri.	Video	M5: SIMDRAM (PDF) (PPT)		
W6	29.04 Fri.	Video	M6: DAMOV (PDF) (PPT)		
W7	06.05 Fri.	Video	M7: Syncron (PDF) (PPT)		

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"** *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems* (***SIGMETRICS***), Phoenix, AZ, USA, June 2019.
  [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

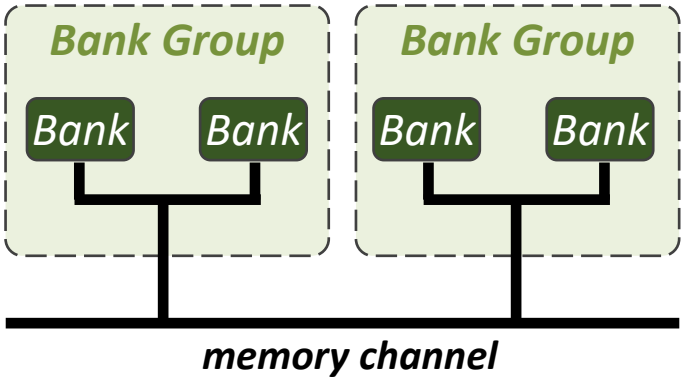
DRAM Type	Banks per Rank	Bank Groups	3D-Stacked	Low-Power
DDR3	8			
DDR4	16	✓		
GDDR5	16	✓		
HBM High-Bandwidth Memory	16		✓	
HMC Hybrid Memory Cube	256		✓	
Wide I/O	4		✓	✓
Wide I/O 2	8		✓	✓
LPDDR3	8			✓
LPDDR4	16			✓

increased latency

increased area/power

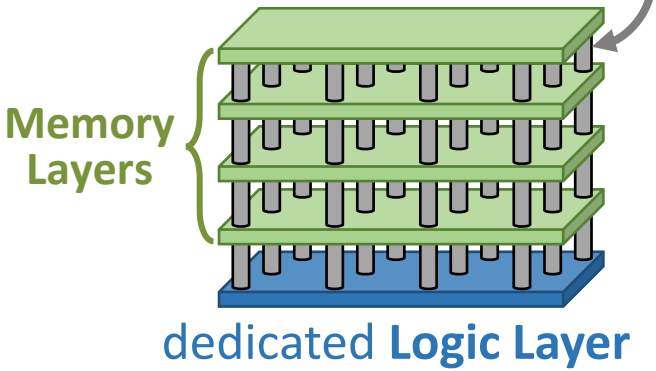
narrower rows, higher latency

- Bank groups



- 3D-stacked DRAM

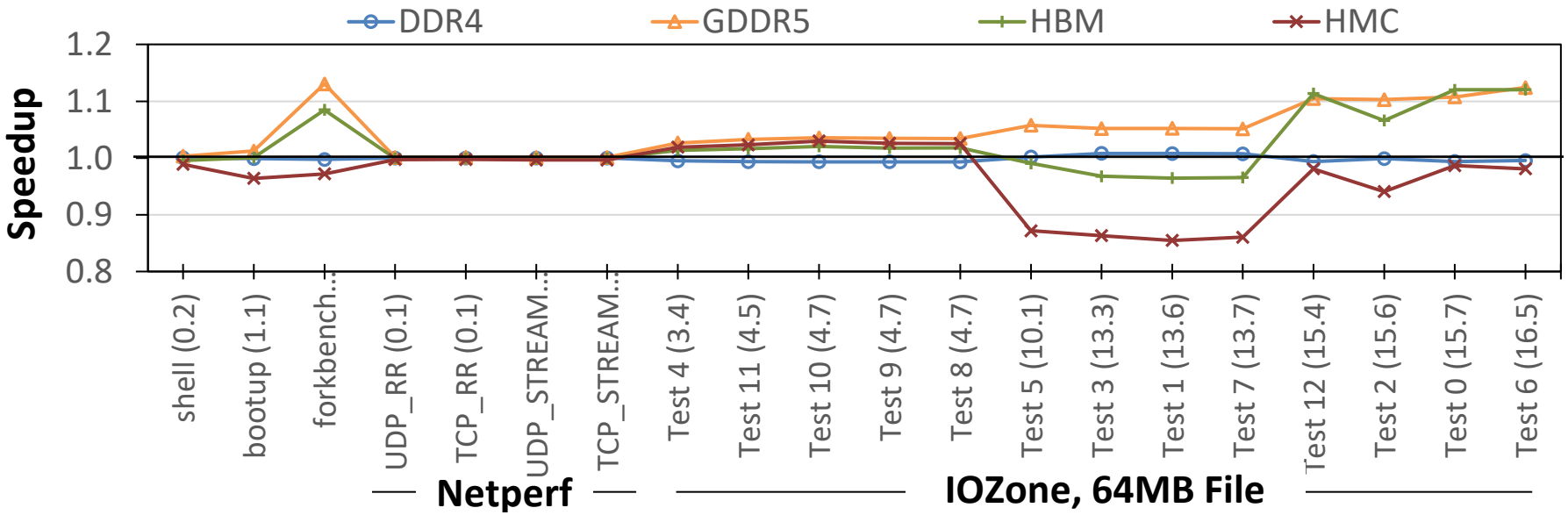
high bandwidth with Through-Silicon Vias (TSVs)



4. Need for Lower Access Latency: Performance

SAFARI

- 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…

- Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu, **"Demystifying Workload–DRAM Interactions: An Experimental Study"** *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems* (***SIGMETRICS***), Phoenix, AZ, USA, June 2019.
  [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 Olgun, 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"
  *Proceedings of the 27th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Virtual, February-March 2021.
  [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)

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

A. Giray Yağlıkçı[1] Minesh Patel[1] Jeremie S. Kim[1] Roknoddin Azizi[1] Ataberk Olgun[1] Lois Orosa[1]
Hasan Hassan[1] Jisung Park[1] Konstantinos Kanellopoulos[1] Taha Shahroodi[1] Saugata Ghose[2] Onur Mutlu[1]
[1]*ETH Zürich* [2]*University of Illinois at Urbana–Champaign*

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

Source

BlockHammer: Free & Open Source

README.md

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 [cs.CR]`

Citation

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

> A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, 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"` Proceedings of the 27th International Symposium on High-Performance Computer Architecture (HPCA), Virtual, February-March 2021.

Getting Started

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

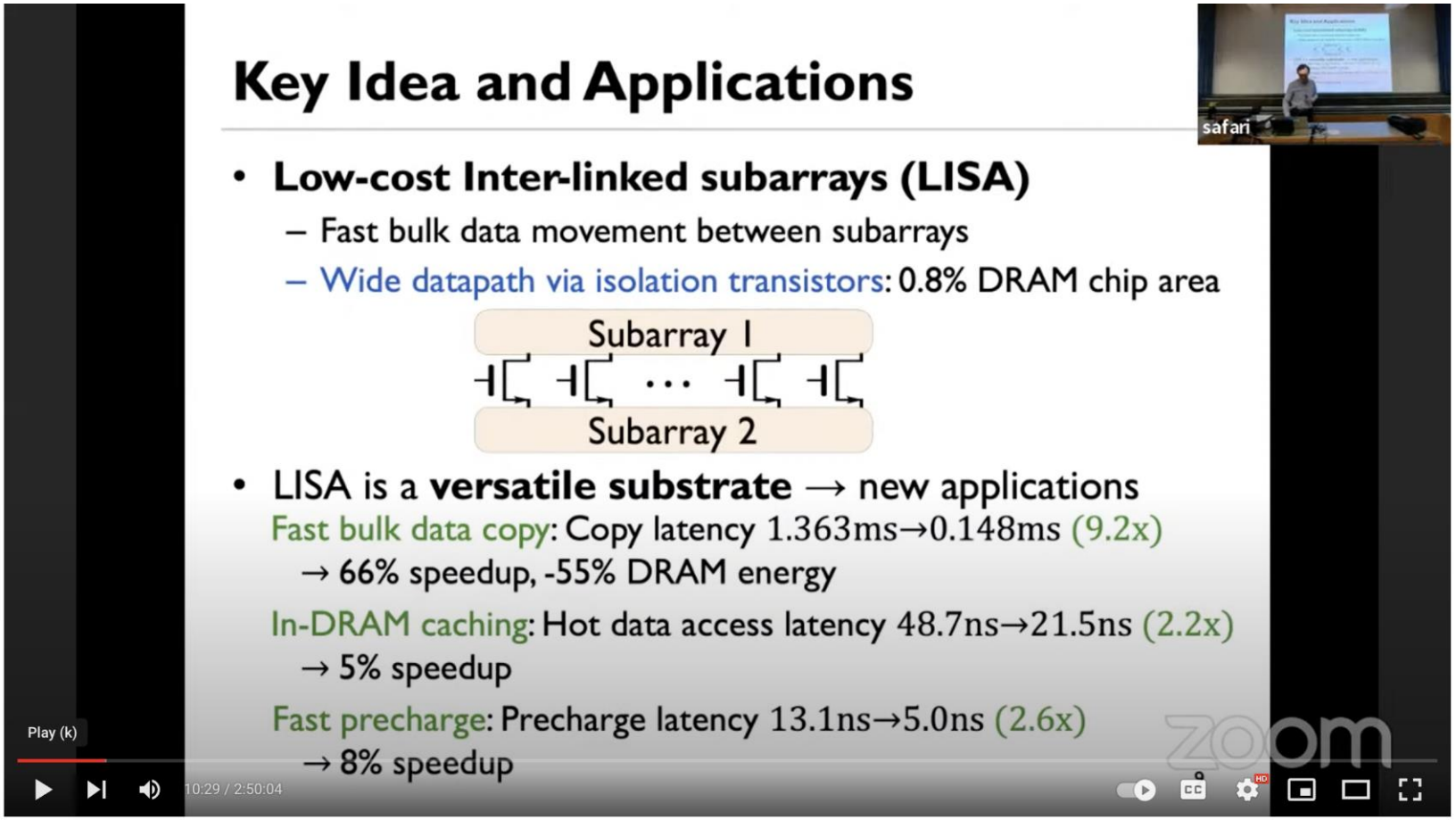
- `Ramulator Model`: This subdirectory includes an extended version of `Ramulator` with a RowHammerDefense class, which implements BlockHammer along with six state-of-the-art RowHammer mitigaiton mechanisms:

Mechanism	Reference
PARA	Y. Kim, et al., **"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors,"** in ISCA, 2014.
CBT	S. M. Seyedzadeh et al., **"Mitigating Wordline Crosstalk Using Adaptive Trees of Counters,"** in ISCA, 2018.
ProHIT	M. Son et al., **"Making DRAM Stronger Against Row Hammering,"** in DAC, 2017.
MrLoc	J. M. You et al., **"MRLoc: Mitigating Row-Hammering Based on Memory Locality,"** in DAC, 2019.
TWiCe	E. Lee et al., **"TWiCe: Preventing Row-hammering by Exploiting Time Window Counters"** in ISCA, 2019.
Graphene	Y. Park et al., **"Graphene: Strong yet Lightweight Row Hammer Protection,"** in MICRO, 2020.

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

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

Many Other Ideas Evaluated w/ Ramulator



Computer Architecture - Lecture 9: Memory Latency & Memory Controllers (Fall 2022)

Onur Mutlu Lectures
28.8K subscribers
Analytics | Edit video | 29 | Share | Download | Clip | Save | ...

711 views Streamed live on Oct 27, 2022
Computer Architecture, ETH Zürich, Fall 2022 (https://safari.ethz.ch/architecture/f...)

https://www.youtube.com/watch?v=pJ3mdIvMOY4&list=PL5Q2soXY2Zi-cAls3cyauNzM7-74Eq31O&index=9

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
  - [Source Code for Ramulator-PIM]

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim[1] Weikun Yang[1,2] Onur Mutlu[1]
[1]Carnegie Mellon University [2]Peking University

Ramulator 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[a,c] Juan Gómez-Luna[b] Giovanni Mariani[c] Geraldo F. Oliveira[b]
Stefano Corda[a,c] Sander Stuijk[a] Onur Mutlu[b] Henk Corporaal[a]
[a]Eindhoven University of Technology [b]ETH Zürich [c]IBM Research - Zurich

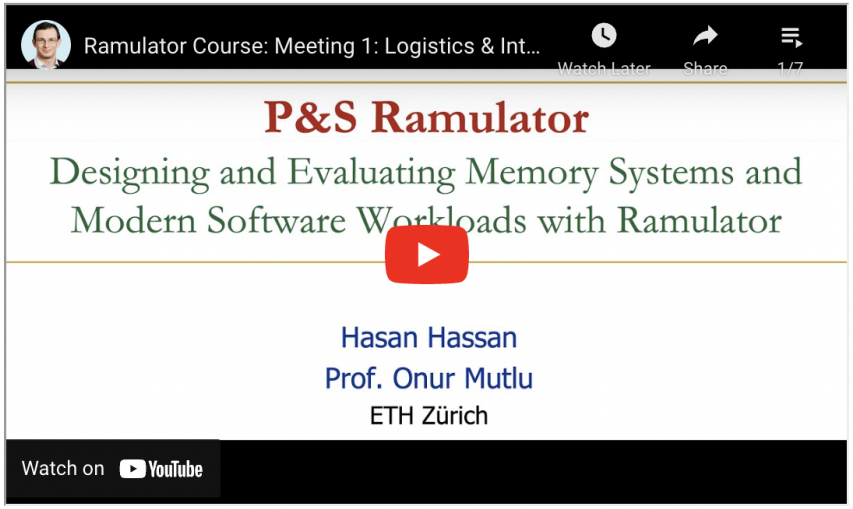
Ramulator Project Course

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_TlmLGw_Z8hBo2925ZApqV
- Bachelor's course
  - Elective at ETH Zurich
  - Introduction to memory system simulation
  - Tutorial on using Ramulator
  - C++
  - Potential research exploration

Lecture Video Playlist on YouTube

Lecture Playlist



2022 Meetings/Schedule (Tentative)

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	09.03 Wed.	Video	M1: Logistics & Intro to Simulating Memory Systems Using Ramulator (PDF) (PPT)		HW0
W2	16.03 Fri.	Video	M2: Tutorial on Using Ramulator (PDF) (PPT)		
W3	25.02 Fri.	Video	M3: BlockHammer (PDF) (PPT)		
W4	01.04 Fri.	Video	M4: CLR-DRAM (PDF) (PPT)		
W5	08.04 Fri.	Video	M5: SIMDRAM (PDF) (PPT)		
W6	29.04 Fri.	Video	M6: DAMOV (PDF) (PPT)		
W7	06.05 Fri.	Video	M7: Syncron (PDF) (PPT)		

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 Simulator, Methods & Benchmarks

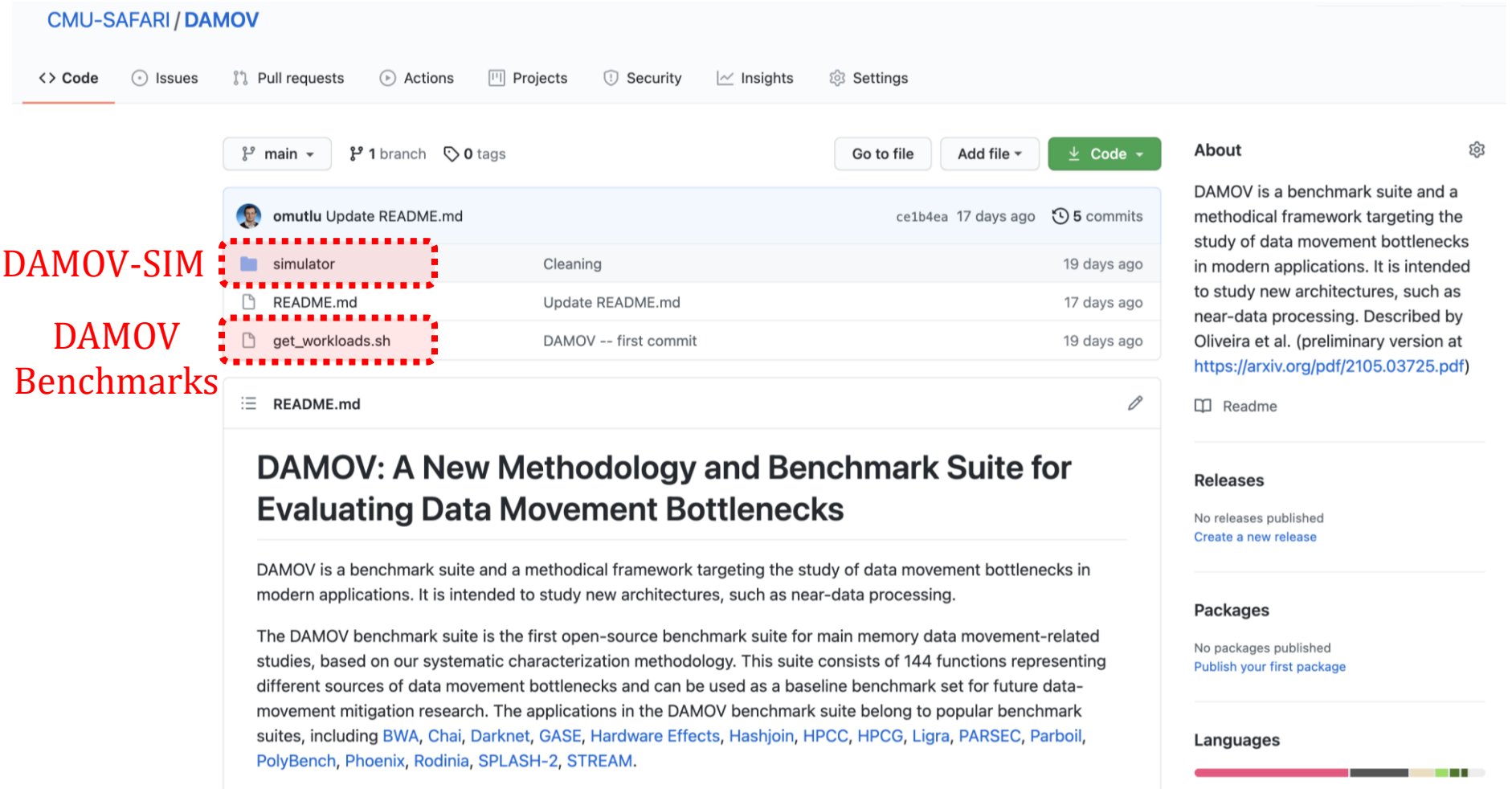
- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan fernandez, Mohammad Sadrosadati, and Onur Mutlu, **"DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"**
  IEEE Access, 8 September 2021.
  Preprint in ***arXiv***, 8 May 2021.
  [arXiv preprint]
  [IEEE Access version]
  [DAMOV Suite and Simulator Source Code]
  [SAFARI Live Seminar Video (2 hrs 40 mins)]
  [Short Talk Video (21 minutes)]

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

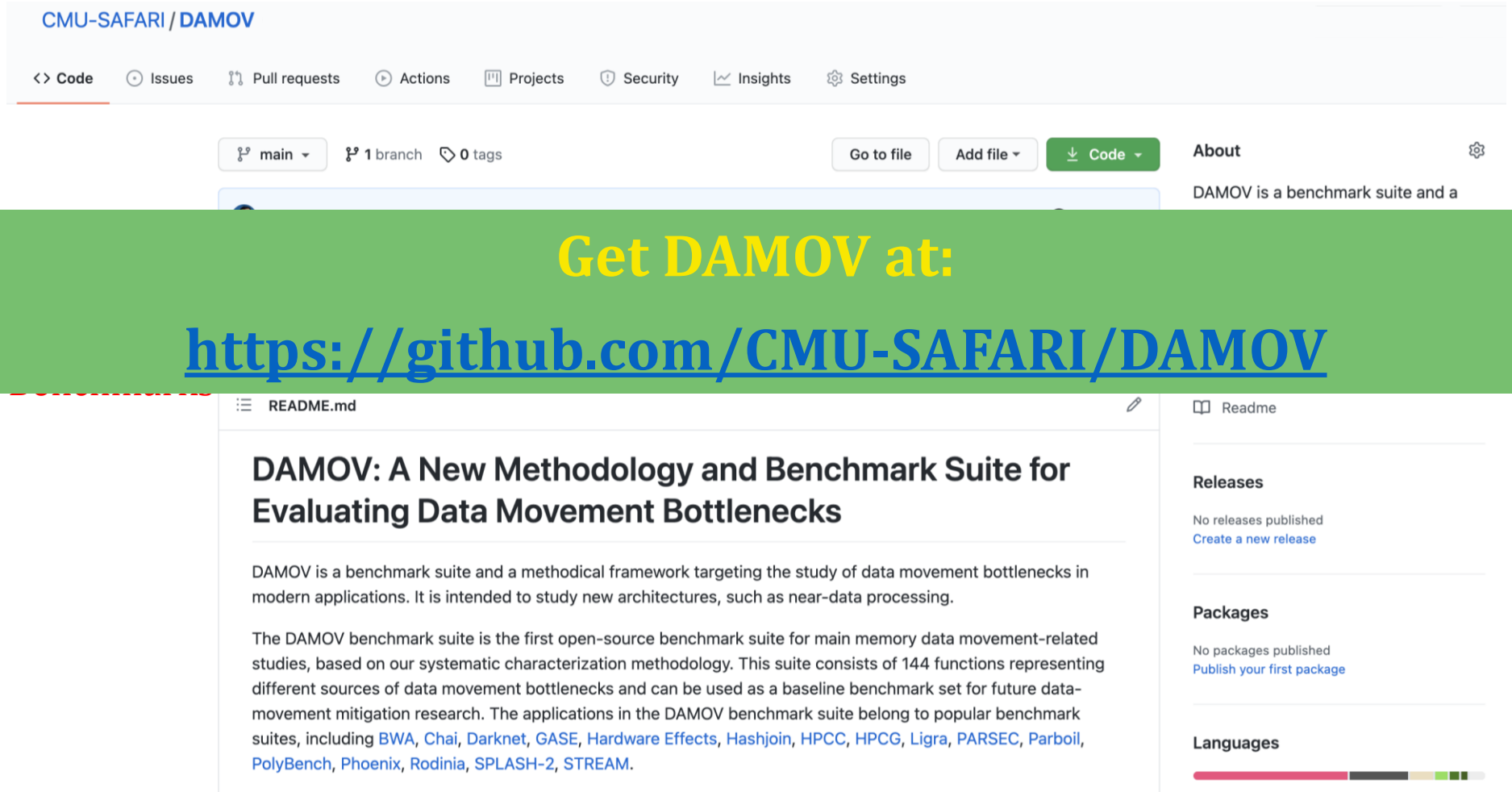
DAMOV is Open Source

- We open-source our benchmark suite and our toolchain

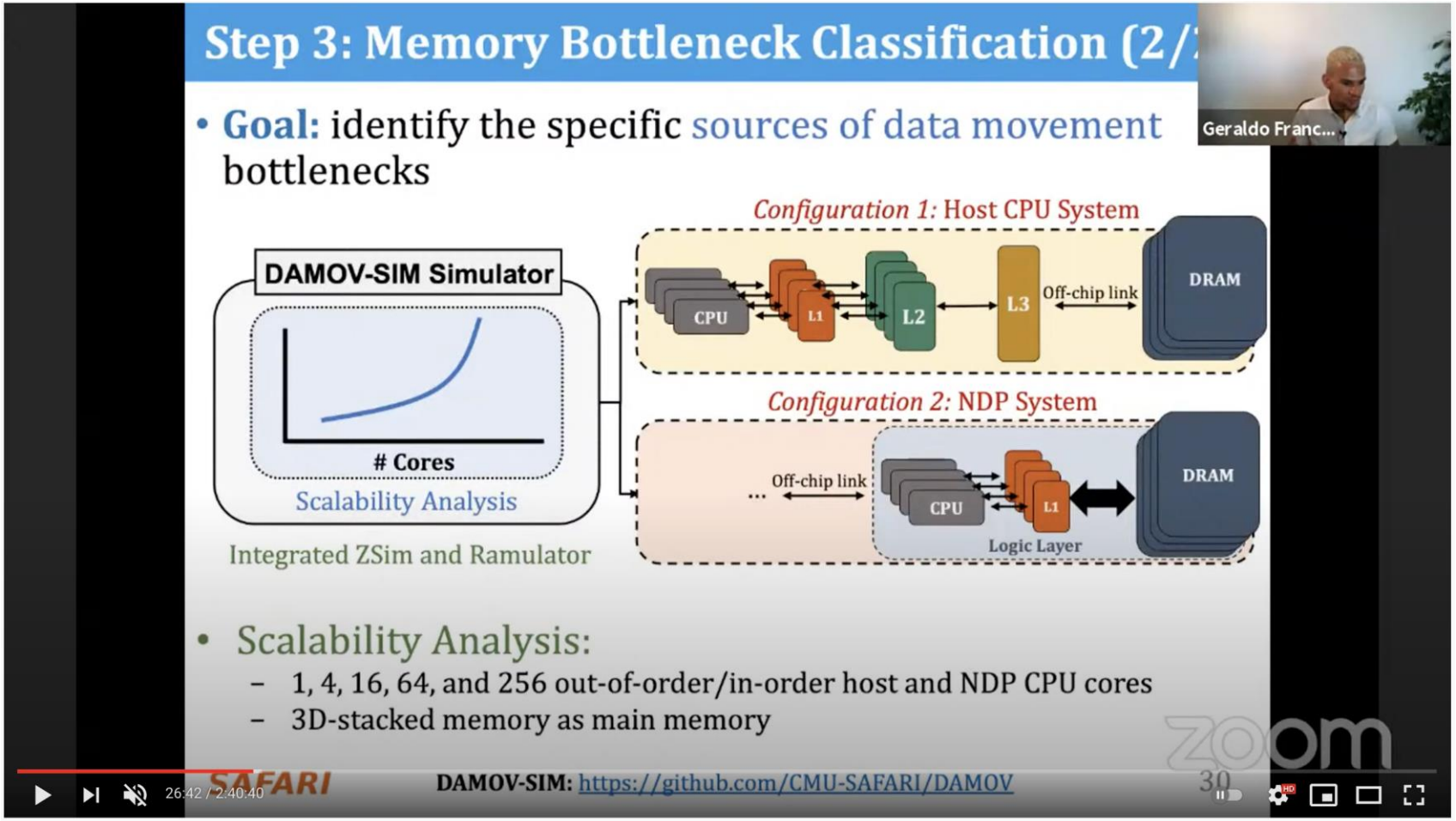


DAMOV is Open Source

- We open-source our benchmark suite and our toolchain



More on DAMOV Analysis Methodology & Workloads



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

352 views • Streamed live on Jul 22, 2021

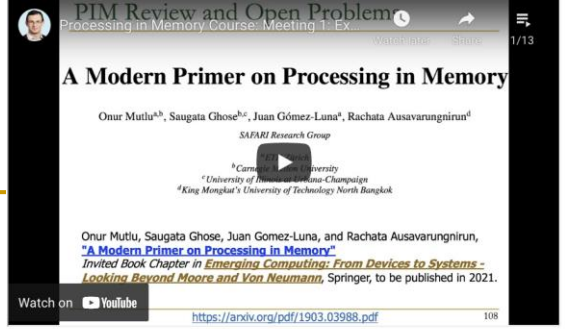
18 0 SHARE SAVE ...

Onur Mutlu Lectures
17.7K subscribers

ANALYTICS EDIT VIDEO

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
- **Youtube Livestream:**
  - https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi-841fUYYUK9EsXKhQKRPyX
- Project course
  - Taken by Bachelor's/Master's students
  - Processing-in-Memory lectures
  - Hands-on research exploration
  - Many research readings

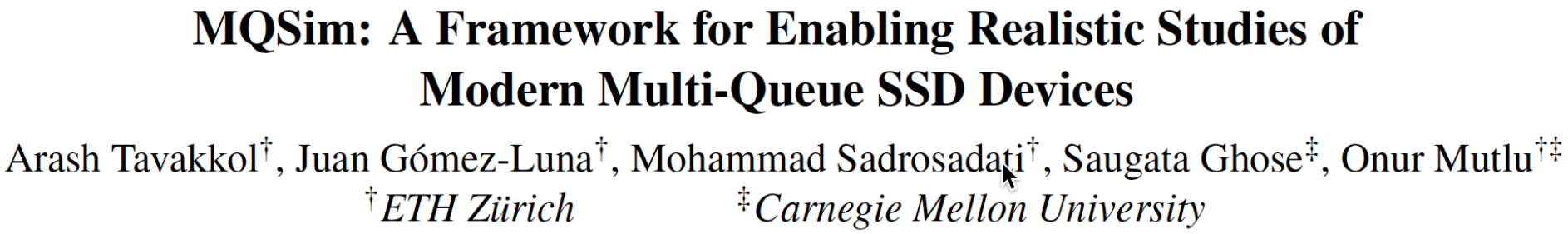
Spring 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	10.03 Thu.	Live	M1: P&S PIM Course Presentation (PDF) (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	15.03 Tue.		Hands-on Project Proposals		
	17.03 Thu.	Premiere	M2: Real-world PIM: UPMEM PIM (PDF) (PPT)		
W3	24.03 Thu.	Live	M3: Real-world PIM: Microbenchmarking of UPMEM PIM (PDF) (PPT)		
W4	31.03 Thu.	Live	M4: Real-world PIM: Samsung HBM-PIM (PDF) (PPT)		
W5	07.04 Thu.	Live	M5: How to Evaluate Data Movement Bottlenecks (PDF) (PPT)		
W6	14.04 Thu.	Live	M6: Real-world PIM: SK Hynix AiM (PDF) (PPT)		
W7	21.04 Thu.	Premiere	M7: Programming PIM Architectures (PDF) (PPT)		
W8	28.04 Thu.	Premiere	M8: Benchmarking and Workload Suitability on PIM (PDF) (PPT)		
W9	05.05 Thu.	Premiere	M9: Real-world PIM: Samsung AxDIMM (PDF) (PPT)		
W10	12.05 Thu.	Premiere	M10: Real-world PIM: Alibaba HB-PNM (PDF) (PPT)		
W11	19.05 Thu.	Live	M11: SpMV on a Real PIM Architecture (PDF) (PPT)		
W12	26.05 Thu.	Live	M12: End-to-End Framework for Processing-using-Memory (PDF) (PPT)		
W13	02.06 Thu.	Live	M13: Bit-Serial SIMD Processing using DRAM (PDF) (PPT)		
W14	09.06 Thu.	Live	M14: Analyzing and Mitigating ML Inference Bottlenecks (PDF) (PPT)		
W15	15.06 Thu.	Live	M15: In-Memory HTAP Databases with HW/SW Co-design (PDF) (PPT)		
W16	23.06 Thu.	Live	M16: In-Storage Processing for Genome Analysis (PDF) (PPT)		
W17	18.07 Mon.	Premiere	M17: How to Enable the Adoption of PIM? (PDF) (PPT)		
W18	09.08 Tue.	Premiere	SS1: ISVLSI 2022 Special Session on PIM (PDF & PPT)		

https://www.youtube.com/onurmutlulectures

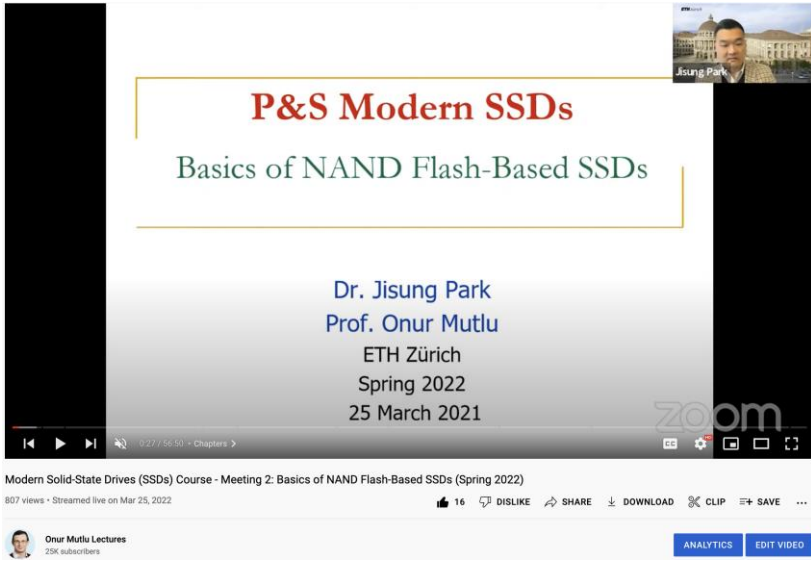
MQSim for Modern SSD Simulation

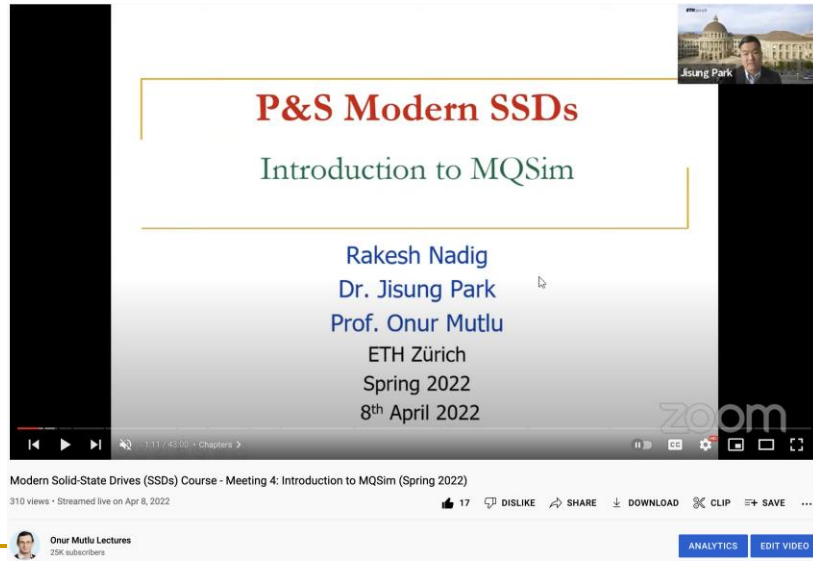
- Arash Tavakkol, Juan Gomez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu,
  "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices"
  *Proceedings of the 16th USENIX Conference on File and Storage Technologies (**FAST**)*, Oakland, CA, USA, February 2018.
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



Solid-State Drives Course (Spring 2022)

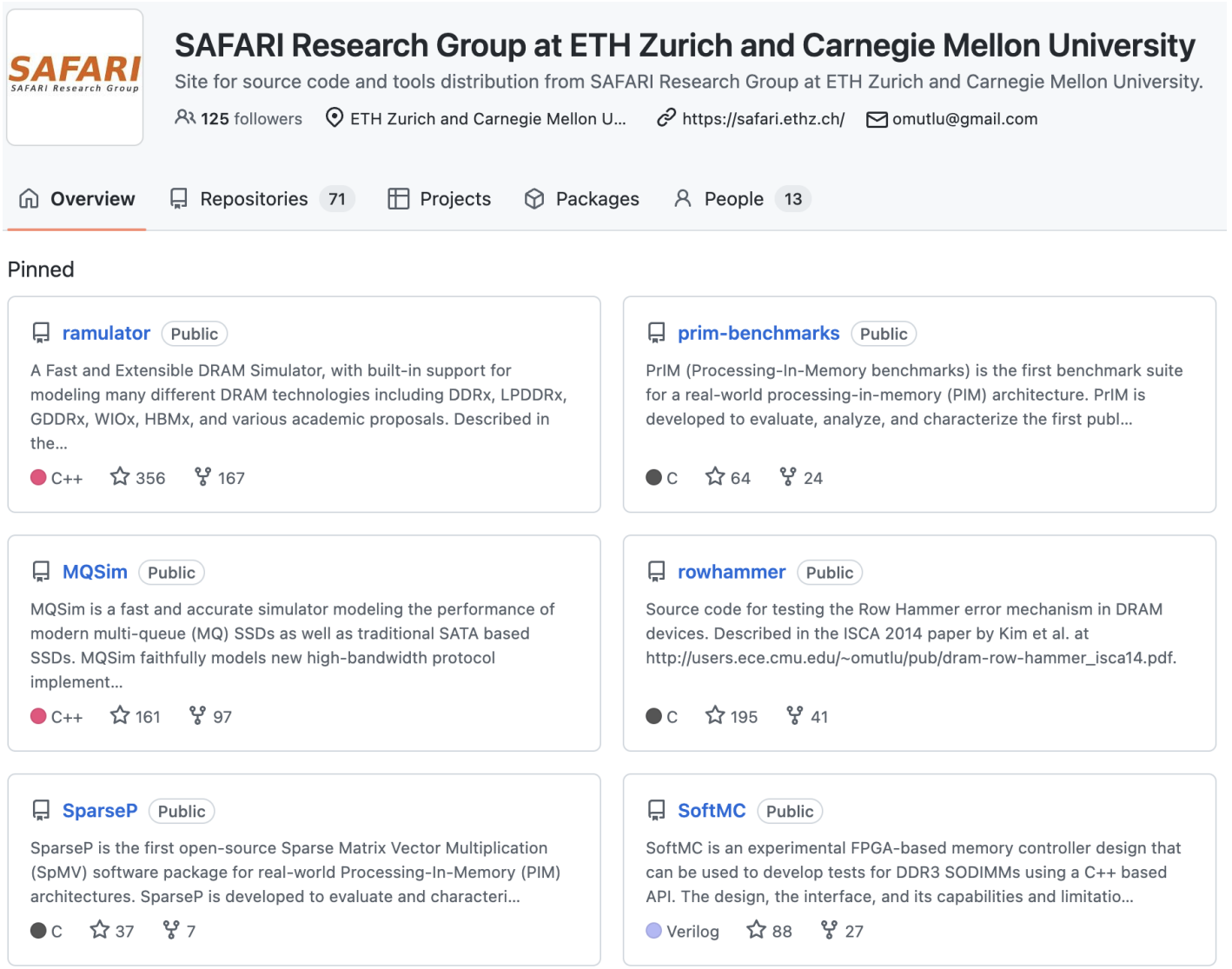
- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=modern_ssds
- **Youtube Livestream:**
  - https://www.youtube.com/watch?v=_q4rm71DsY4&list=PL5Q2soXY2Zi8vabcse1kL22DEcgMl2RAq
- Project course
  - Taken by Bachelor's/Master's students
  - SSD Basics and Advanced Topics
  - Hands-on research exploration
  - Many research readings





https://www.youtube.com/onurmutlulectures

Many More Simulators …



https://github.com/CMU-SAFARI

What We Discussed Is Applicable to Simulation in Other Domains

Case Study:
COVID-19 Spread
Modeling and Prediction

frontiers | Frontiers in Public Health

ORIGINAL RESEARCH
published: 17 June 2022
doi: 10.3389/fpubh.2022.877621

COVIDHunter: COVID-19 Pandemic Wave Prediction and Mitigation via Seasonality Aware Modeling

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

Department of Information Technology and Electrical Engineering (D-ITET), 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

OPEN ACCESS

Edited by:
Zisis Kozlakidis, International Agency For Research On Cancer (IARC), France

Reviewed by:
Shudi Li, University of Texas Health Science Center at Houston, United States
Vivek Bora, Nirma University, India

****Correspondence:***
Mohammed Alser
alserm@ethz.ch

Specialty section:
This article was submitted to Infectious Diseases – Surveillance, Prevention and Treatment, a section of the journal Frontiers in Public Health

Received: *16 February 2022*
Accepted: *20 May 2022*
Published: *17 June 2022*

Citation:
Alser M, Kim JS, Almadhoun Alserr N, Tell SW and Mutlu O (2022) COVIDHunter: COVID-19 Pandemic Wave Prediction and Mitigation via Seasonality Aware Modeling. Front. Public Health 10:877621. doi: 10.3389/fpubh.2022.877621

https://arxiv.org/pdf/2102.03667.pdf

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 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 1 | Comparison to other models used to inform government policymakers, as of January 2021.

Model	Open source	Well documented[#]	Accounting for seasonality	Low number of parameters	Reported COVID-19 statistics
COVIDHunter (this work)	✓	✓	✓	✓	✓ (*R*, cases, hospitalizations, and deaths)
IBZ (11)	✓	χ	χ	✓	χ (only *R*)
LSHTM (7)	✓	χ	χ	✓	χ (only cases)
ICL (9)	✓	✓	χ	χ	✓ (*R*, cases, hospitalizations, and deaths)
IHME (10)	✓*	χ	χ	χ	χ (cases, hospitalizations, and deaths)

*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).

https://arxiv.org/pdf/2102.03667.pdf

Simulating & Predicting COVID-19 Spread

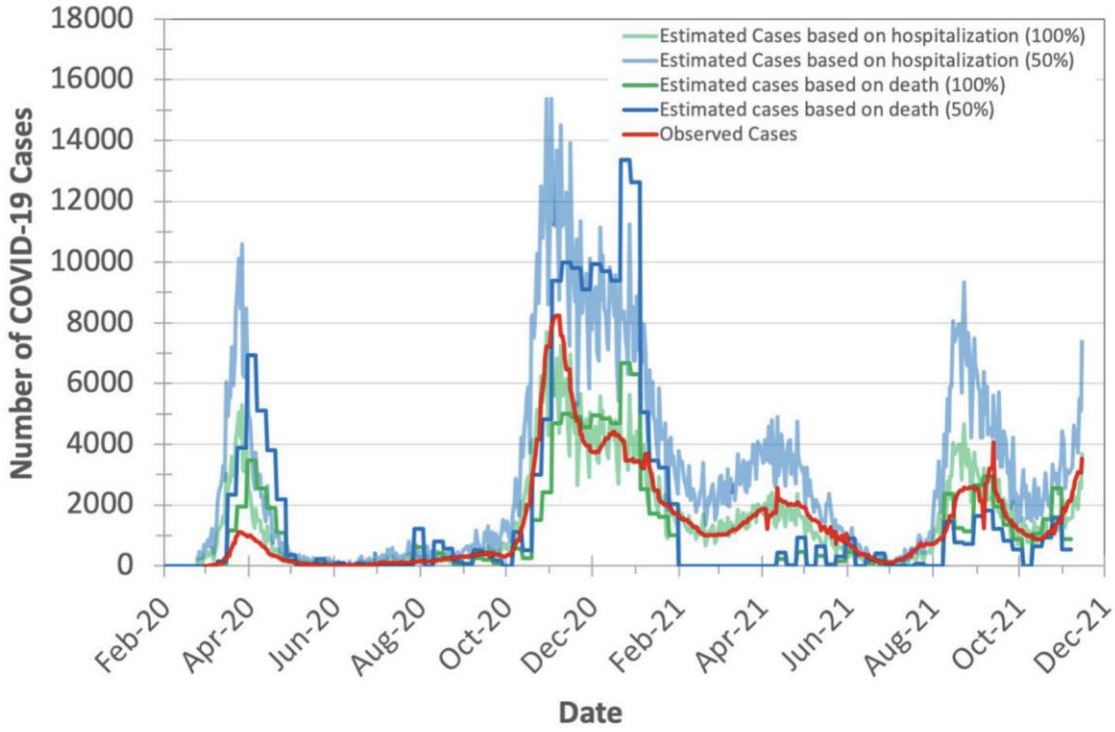


FIGURE 4 | Observed (officially reported) and expected number of COVID-19 cases in Switzerland during the years 2020 and 2021. We calculate the expected number of cases based on both the hospitalizations-to-cases and deaths-to-cases ratios for the second wave. We assume two certainty rate levels of 50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

https://arxiv.org/pdf/2102.03667.pdf

Predicting Effectiveness of Measures

Epidemiological Situation in Switzerland Using COVIDHunter

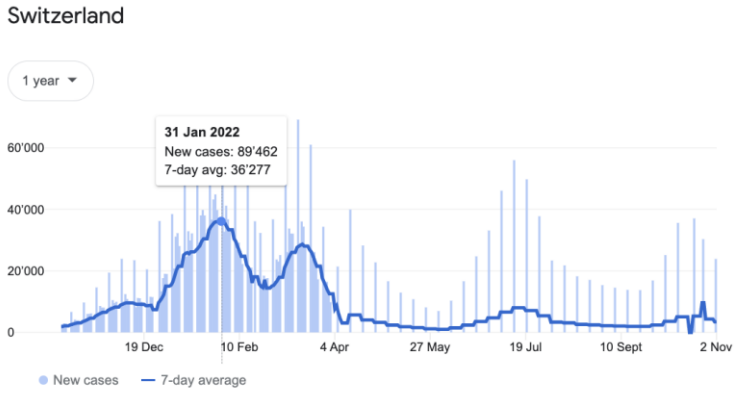
We use Switzerland as a use-case for all the experiments. However, our model is not limited to any specific region as the parameters of COVIDHunter are completely configurable. Using COVIDHunter on **20 November 2021**, we make four key observations:

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 5.5x.
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:

Strengths of the mitigation measures during November-December 2021	0.2	0.3	0.40	0.50	0.60	0.70
Mitigation measures with similar strength were applied on	29 February - 11 March 2020	1 October - 20 October 2020	28 February - 15 March 2020	24 June - 20 August 2020	12-29 October 2020	28 November - 22 December 2020
Predicted daily number of cases	7'822-43'518	5'706-22'413	3'401-10'714	542-1'310	70-410	3-104
Predicted daily number of hospitalizations	124-693	90-357	54-170	9-21	1-7	1-2
Predicted daily number of deaths	38-216	28-112	16-53	3-7	1-3	1-2

Real Outcome



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)

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

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?

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

Computer Architecture

Lecture 22a: Simulation

Dr. Mohammad Sadrosadati
Prof. Onur Mutlu
ETH Zürich

Fall 2023

8 December 2023