Evaluating gem5's Memory Components

Mahyar Samani, Jason Lowe-Power

Department of Computer Science, UC Davis

ISCA, June 18, 2022

Abstract

- Simulation is an important tool in Computer Architecture research.

Abstract

- Simulation is an important tool in Computer Architecture research.
- Sufficiently large errors in simulations could mislead researchers.

Abstract

- Simulation is an important tool in Computer Architecture research.
- Sufficiently large errors in simulations could mislead researchers.
- Simulators need to be validated.

Abstract

- Simulation is an important tool in Computer Architecture research.
- Sufficiently large errors in simulations could mislead researchers.
- Simulators need to be validated.
- In this workshop we present validation of different components of the memory subsystem (i.e. DRAM and Caches).

What causes errors?

- Modeling errors: modeling functionality incorrectly.
  - example: gem5 memory models delivering wrong values for accesses.

What causes errors?

- Modeling errors: modeling functionality incorrectly.
- Specification errors: modeling using incorrect information.
  - example: incorrect timing parameters for DRAM models in gem5.

What causes errors?

- Modeling errors: modeling functionality incorrectly.
- Specification errors: modeling using incorrect information.
- Abstraction errors: not accounting for timing effects of abstracting components or forgoing implementing features.
  - example: Not accounting for interleaving cycles when scheduling events in gem5.

How to validate?

- Manual inspection: Inspect the code manually by reading through it.
  - Simulators are almost as complex as the hardware they are simulating.
  - Inspection is susceptible to human error.

How to validate?

- Manual inspection: Inspect the code manually by reading through it.
- Comparing against trusted references.
  - cycle-accurate simulators compare against real hardware.
  - programs are formally verified.

Why is it hard to validate gem5?

- gem5 code base is complex.
  - It's hard to manually inspect its code base.

Why is it hard to validate gem5?

- gem5 code base is complex.
- gem5 is not cycle accurate.
  - Information is not available for each cycle. It's not appropriate to take the same approach as cycle-accurate simulators.

Why is it hard to validate gem5?

- gem5 code base is complex.
- gem5 is not cycle accurate.
- gem5 models are diverse.
  - Need to use a general methodology for validation.

What to do with gem5?

- Use a bottom-up approach.
  - Allows for testing more complicated configurations with validated components.

What to do with gem5?

- Use a bottom-up approach.
- Isolate the component under test.
  - Isolates the observed errors to the component under test.
  - Already know CPU models in gem5 are not accurate.

What to do with gem5?

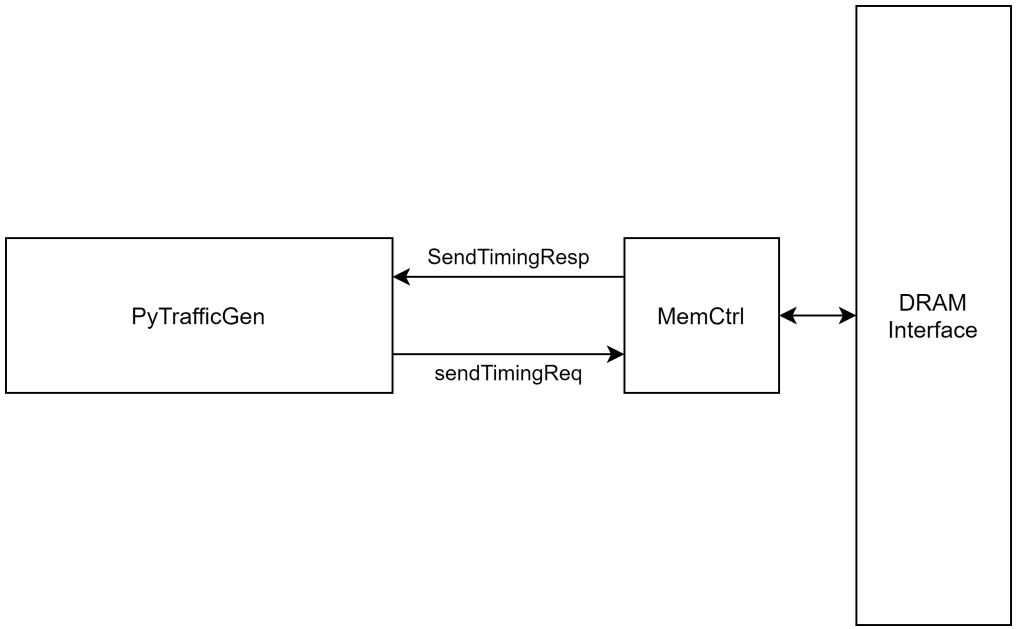
- Use a bottom-up approach.
- Isolate the component under test.
- Use tools that provide defined behavior to test the component.
  - PyTrafficGen, GUPSGen, SimpleMemory

What to do with gem5?

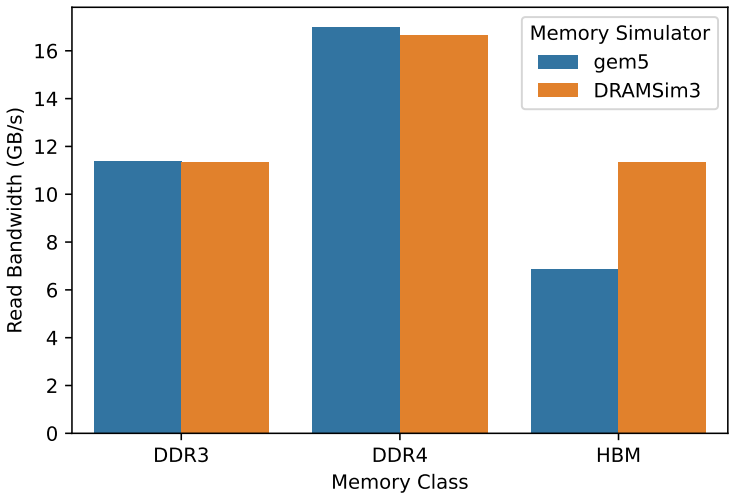
- Use a bottom-up approach.
- Isolate the component under test.
- Use tools that provide defined behavior to test the component.
- For each component use a trusted reference.
  - DRAMSim3, AMAT, GUPS
  - DRAMSim3 has been validated against Micron's Verilog bench.

DRAM Models

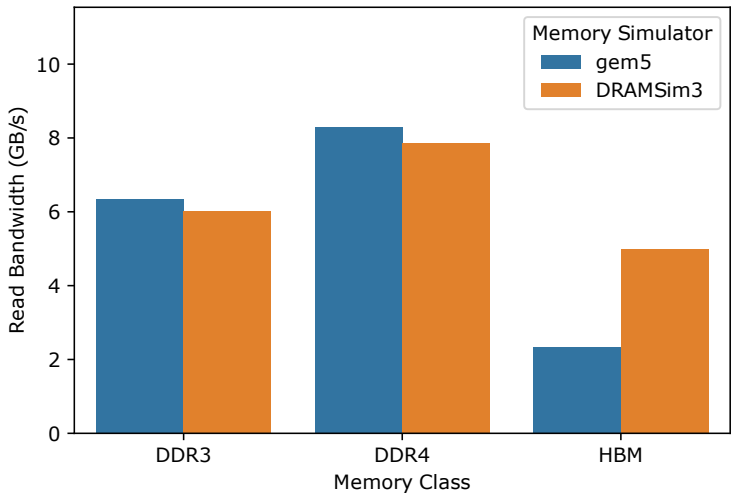
- List of models: DDR3, DDR4, HBM
- Configuration parameters:
  - Address Mapping: RoCoRaBaCh, RoRaBaCoCh, RoRaBaChCo.
  - Paging Policy: Open, Close.
- Reference: Counterpart model in DRAMSim3.
- Tools: PyTrafficGen.
- Metric: bandwidth and average latency.



Bandwidth: gem5 vs DRAMSim3



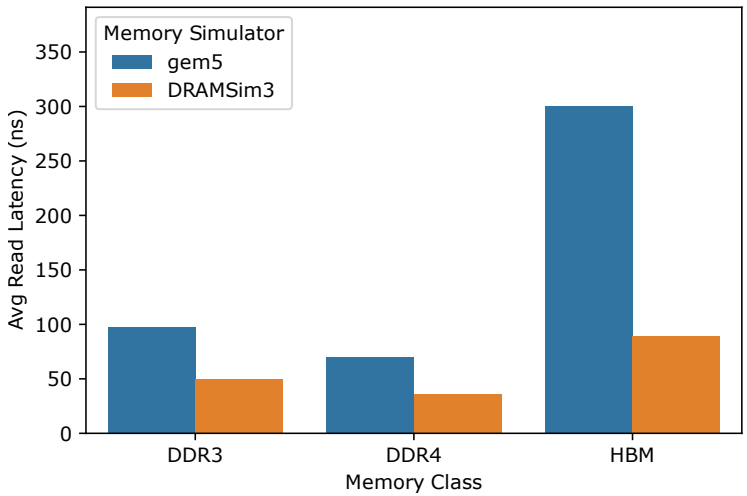
(a) Linear Traffic



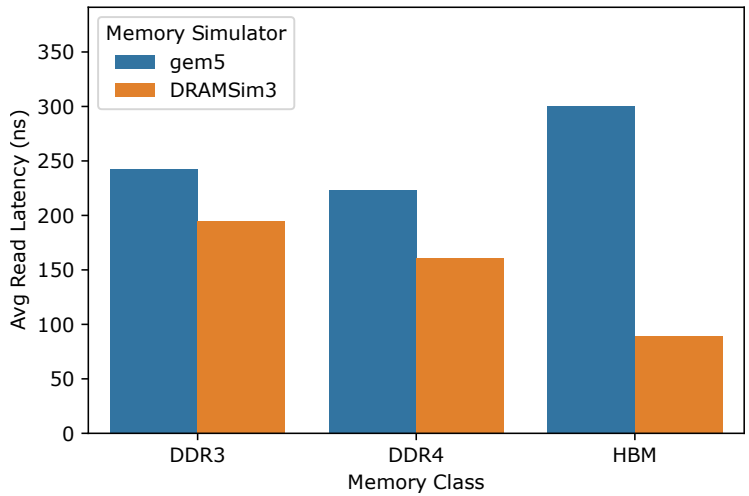
(b) Random Traffic

Figure: Comparing gem5 and DRAMSim3 DRAM models using bandwidth.

Latency: gem5 vs DRAMSim3



(a) Linear Traffic



(b) Random Traffic

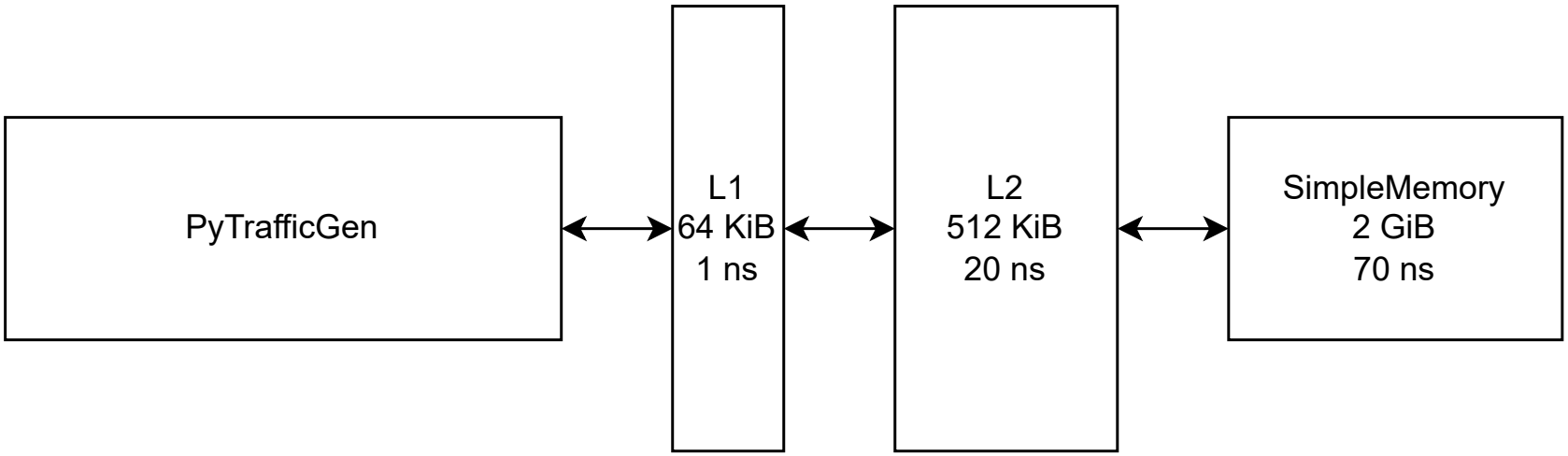
Figure: Comparing gem5 and DRAMSim3 DRAM models using latency.

Caches

- List of models: MESI Two Level cache hierarchy.
- Configuration parameter: Working set size (number of bytes accessed by PyTrafficGen).
- Reference: **A**verage **M**emory **A**ccess **T**ime (AMAT).
- Tools: PyTrafficGen (linear traffic), SimpleMemory
- Metric: average latency

Memory Name	Size	Latency	Notes
L1 Cache	64 KiB	1 ns	
L2 Cache	512 KiB	20 ns	inclusive of L1
Simple Memory	2 GiB	70 ns	

Caches: system diagram



AMAT

$$AMAT = \begin{cases} 1\ ns & 0 < range \leq 64\ KiB \\ 20\ ns - \frac{\alpha}{range} & 64\ KiB < range \leq 512 KiB \\ 70\ ns - \frac{\beta}{range} & 512\ KiB < range \end{cases}$$

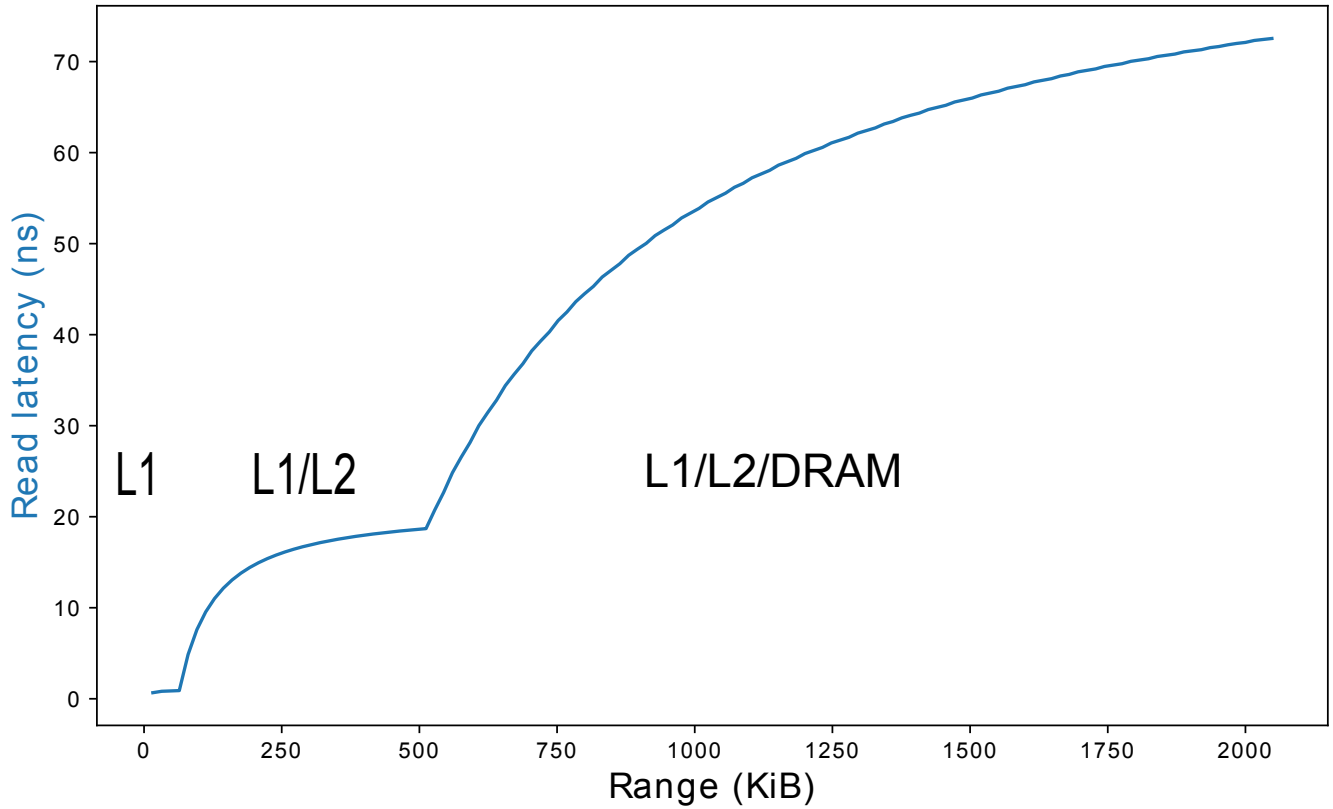


Figure: Effect of hierarchical memory on AMAT.

GUPS Test

- List of models: MESI Two Level cache, Dual channel DDR4.
  - Used available information on Intel Skylake to model cache.
  - Used L2 as L2/L3
- Reference: Intel Skylake.
- Tools: GUPSGen
  - Implemented a synthetic generator based on HPCC RandomAccess benchmark.
- Metric: **G**iga **U**pdates **P**er **S**econd (GUPS)

GUPS: gem5 vs Intel Skylake

Specification		Intel Skylake	gem5 model
L1 Cache	**Size**	32 KiB	32 KiB
	Associativity	8	8
	MSHRs	10	10
	Access Latency	4 cycles	4 cycles
L2 Cache	**Size**	256 KiB	16 MiB
	Associativity	4	16
	Access Latency	12 cycles	40 cycles
L3 Cache	**Size**	16 MiB	N/A
	Associativity	16	N/A
	Access Latency	42	N/A
	GUPS	0.39	0.43

Table: Comparison of real hardware with gem5 model for GUPS Test.

Conclusion

- Presented a validation methodology for the gem5's memory subsystem models.
- Of the three DRAM models tested HBM showed a significant difference between gem5 and DRAMSim3.
- We need to define references that capture the requirements the model is designed to satisfy: cycle-accurate validation of gem5 is not feasible.