FireSim

An Open-Source Platform for Scalable FPGA-Accelerated Hardware Simulation in the Cloud

-

Afternoon Preview

https://fires.im

@firesimproject

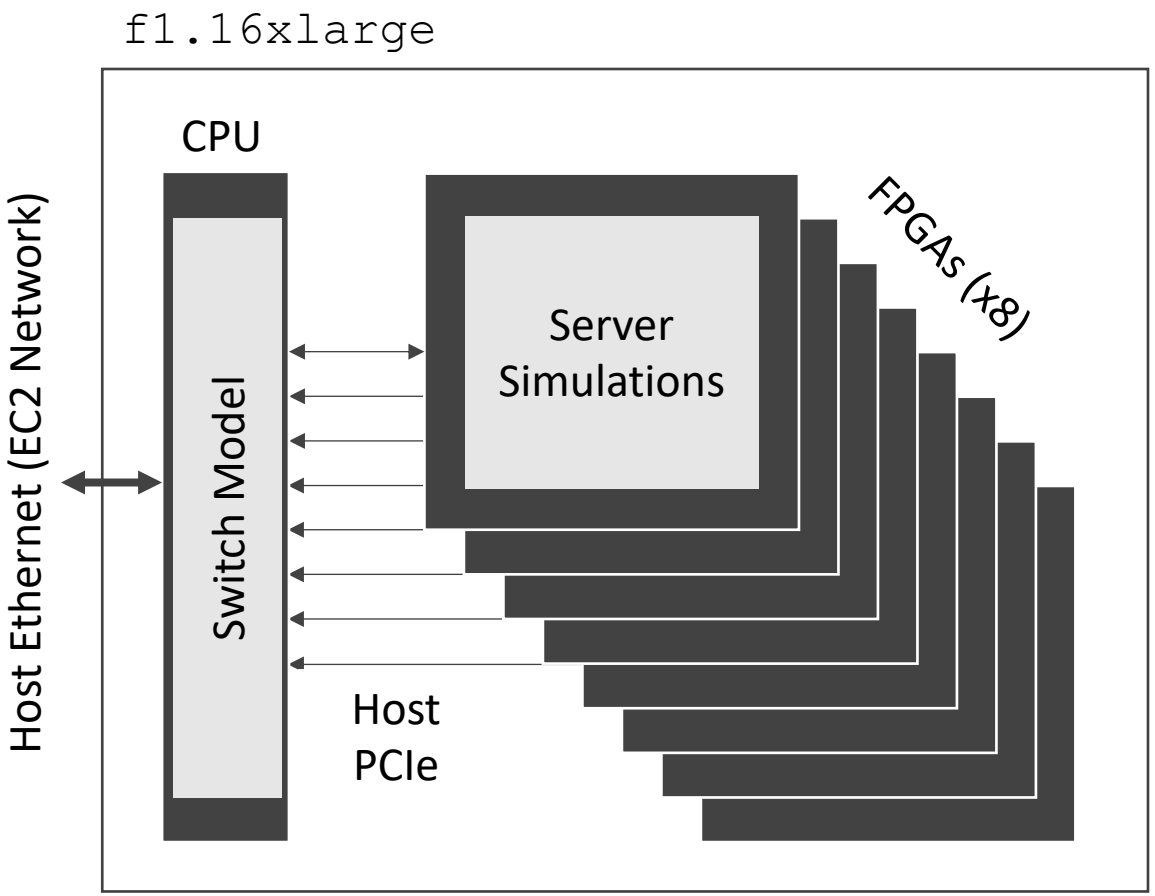
Sagar Karandikar, David Biancolin, Alon Amid, Albert Magyar, Howard Mao, Nathan Pemberton, Albert Ou, Randy Katz, Borivoje Nikolić, Jonathan Bachrach, Krste Asanović

Berkeley Architecture Research

What Can You Do With FireSim?

Simulating a datacenter-scale system

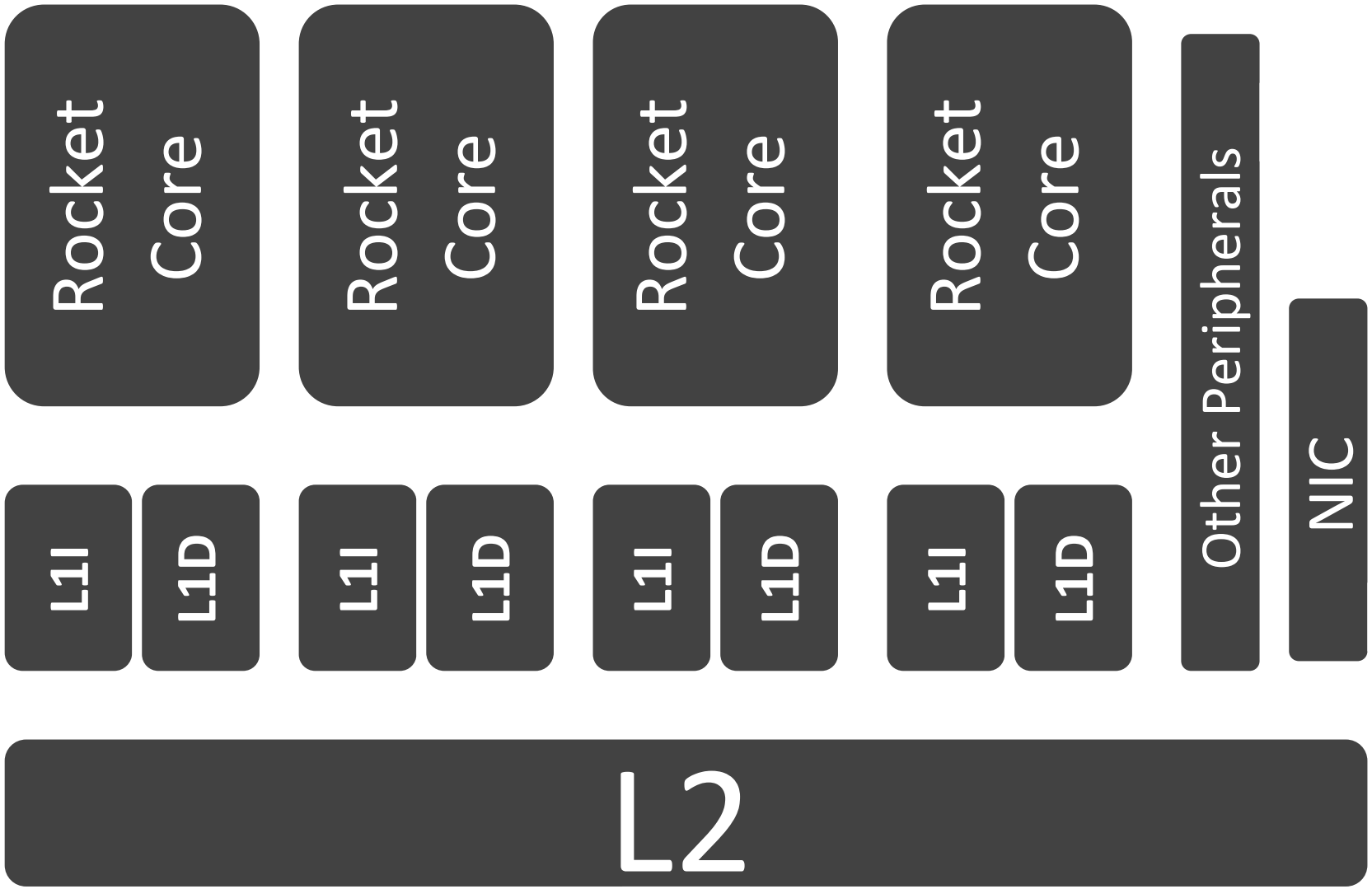
- Scale-out simulation
  - Model hardware at scale, cycle-accurately
  - Run real software
- RTL and abstract SW model co-simulation
- Server Simulations
  - Good fit for the FPGA
  - We have tapeout-proven RTL: FAME-1 transform w/Golden Gate
- Network simulation
  - Little parallelism in switch models (e.g. a thread per port)
  - Need to coordinate all the distributed server simulations
  - So use CPUs + host network



How-to-build a *datacenter-scale* FireSim simulation

[1] S. Karandikar et. al., “FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud.” *ISCA 2018*
[4] S. Karandikar et. al., “FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud.” *IEEE Micro Top Picks 2018*

Step 1: Server SoC in RTL



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1$
- 256K Shared L2$
- 200 Gb/s Eth. NIC

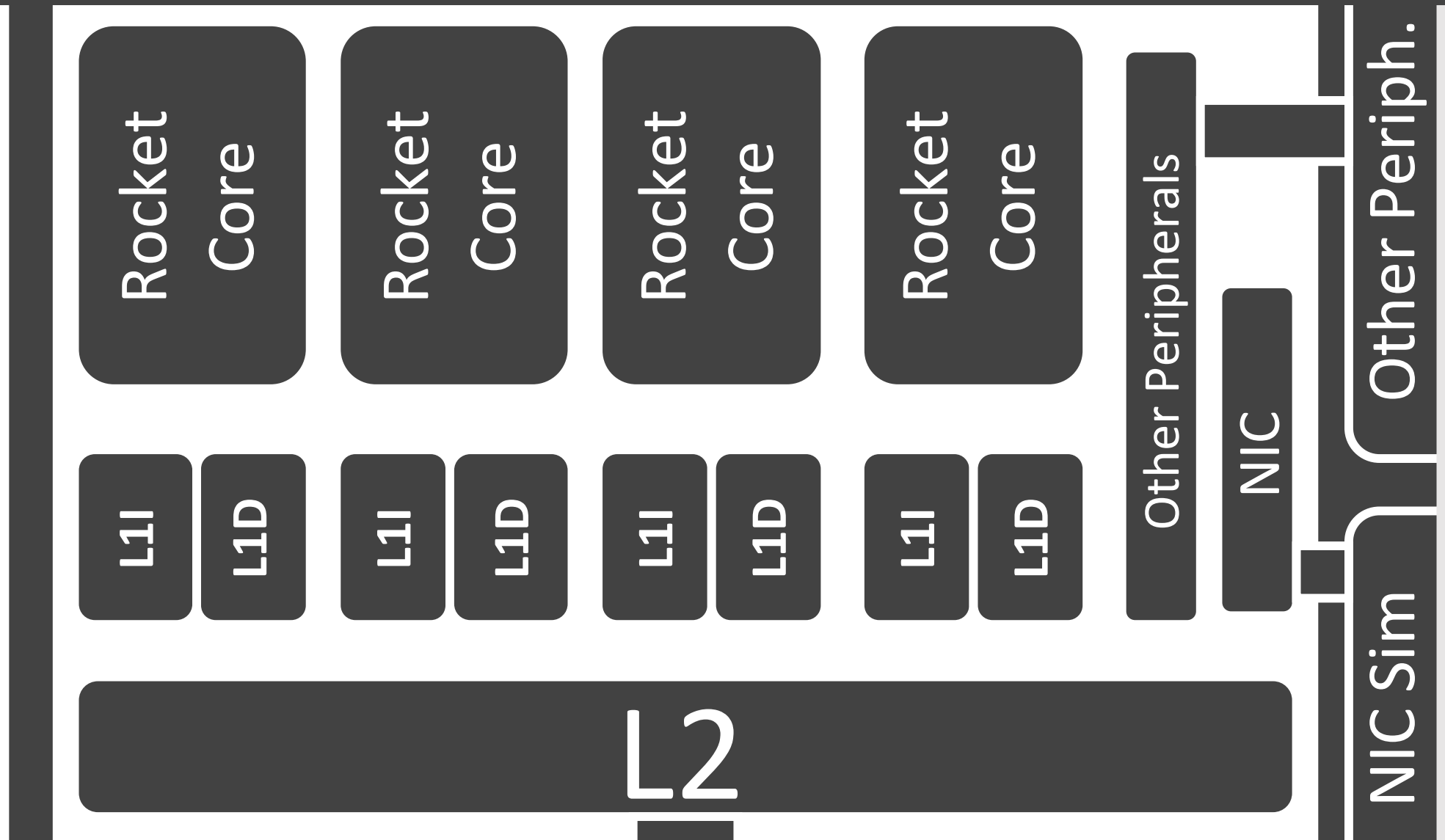
Resource Util.

- < ¼ of an FPGA

Sim Rate

- N/A

Step 1: Server SoC in RTL



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1$
- 256K Shared L2$
- 200 Gb/s Eth. NIC

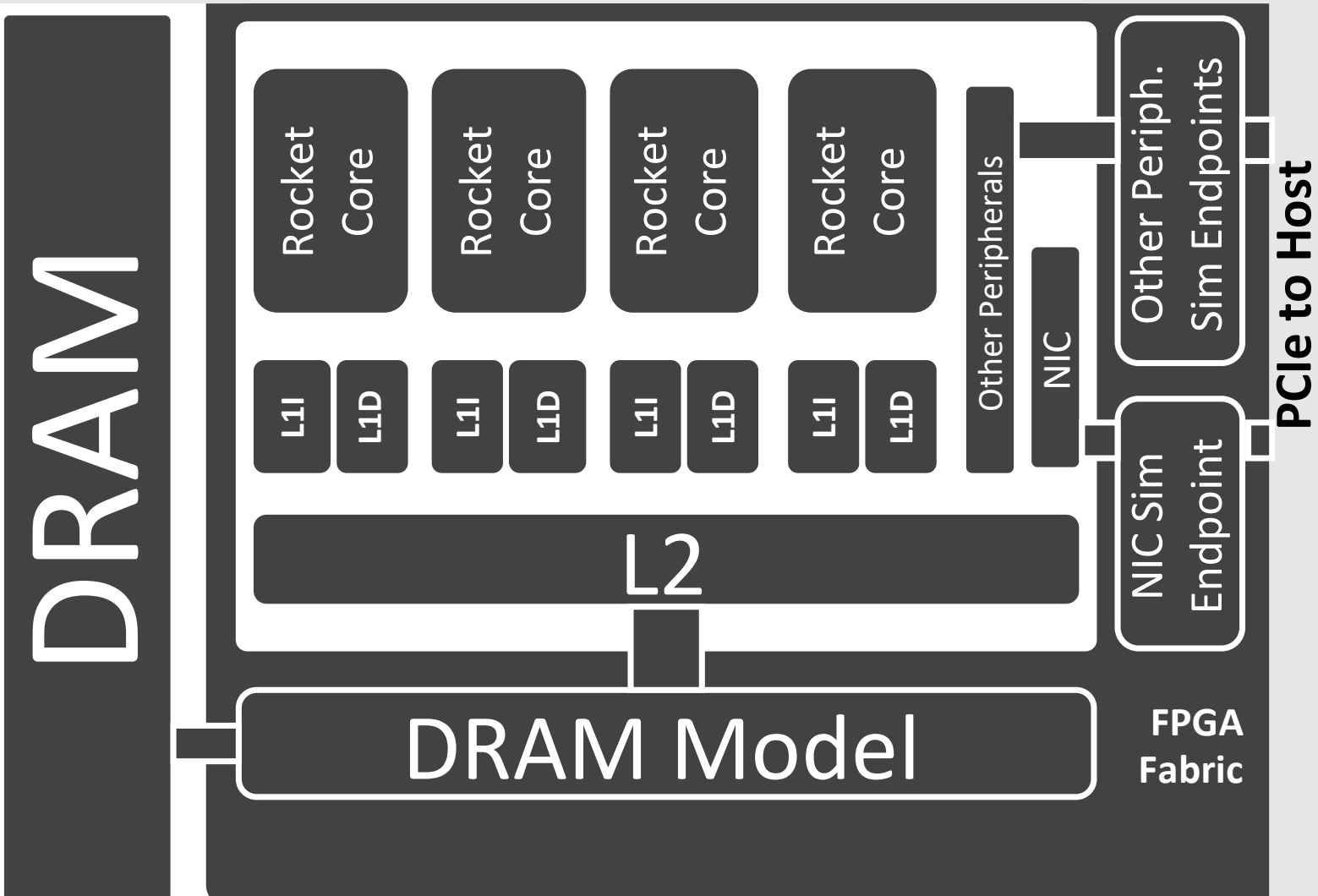
Resource Util.

- < ¼ of an FPGA

Sim Rate

- N/A

Step 2: FPGA Simulation of one server blade



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1$
- 256K Shared L2$
- 200 Gb/s Eth. NIC
- 16 GB DDR3

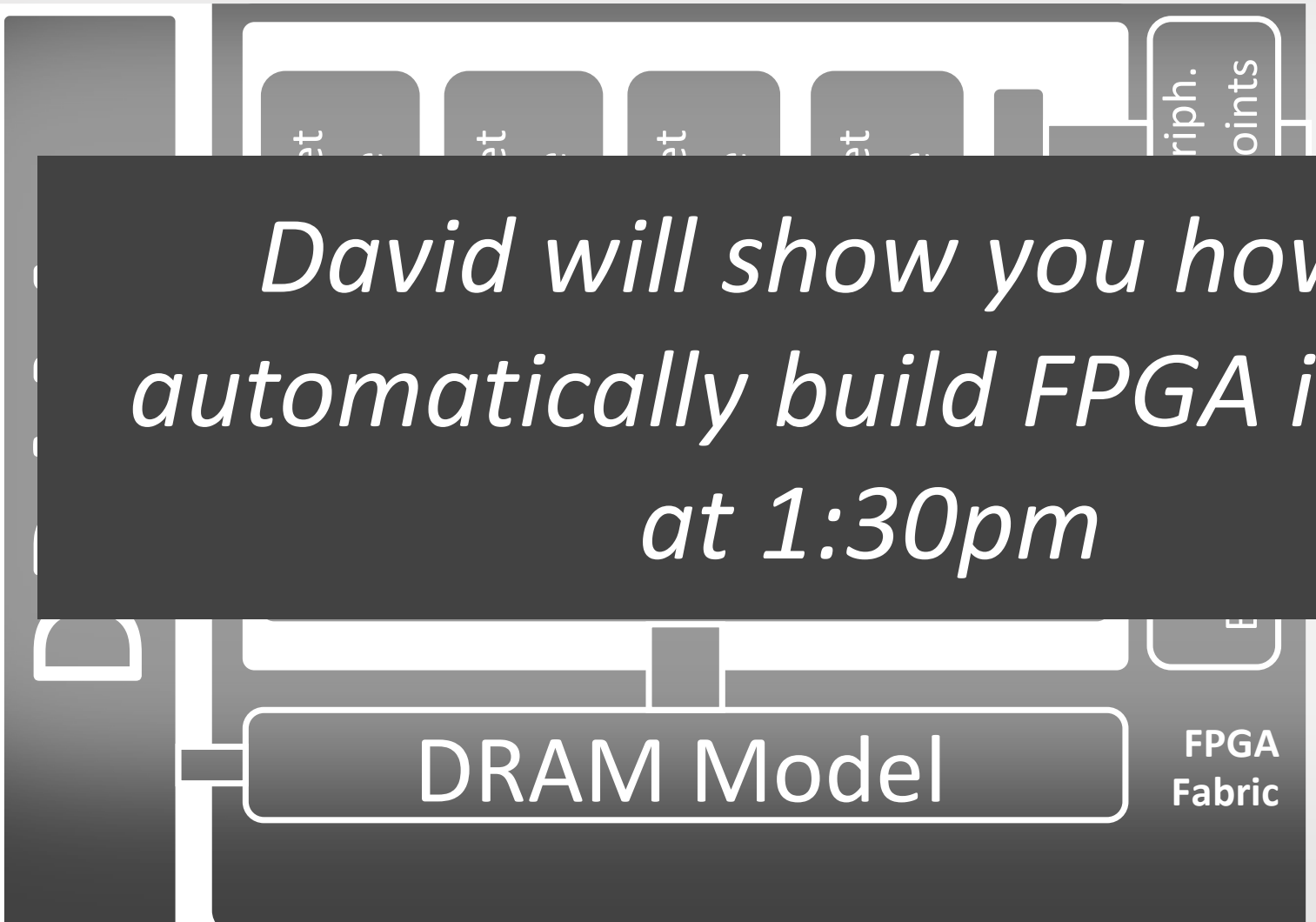
Resource Util.

- < ¼ of an FPGA
- ¼ Mem Chans

Sim Rate

- ~150 MHz
- ~40 MHz (netw)

Step 2: FPGA Simulation of one server blade



David will show you how to automatically build FPGA images at 1:30pm

Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1$
- Shared L2$
- b/s Eth.
- DDR3

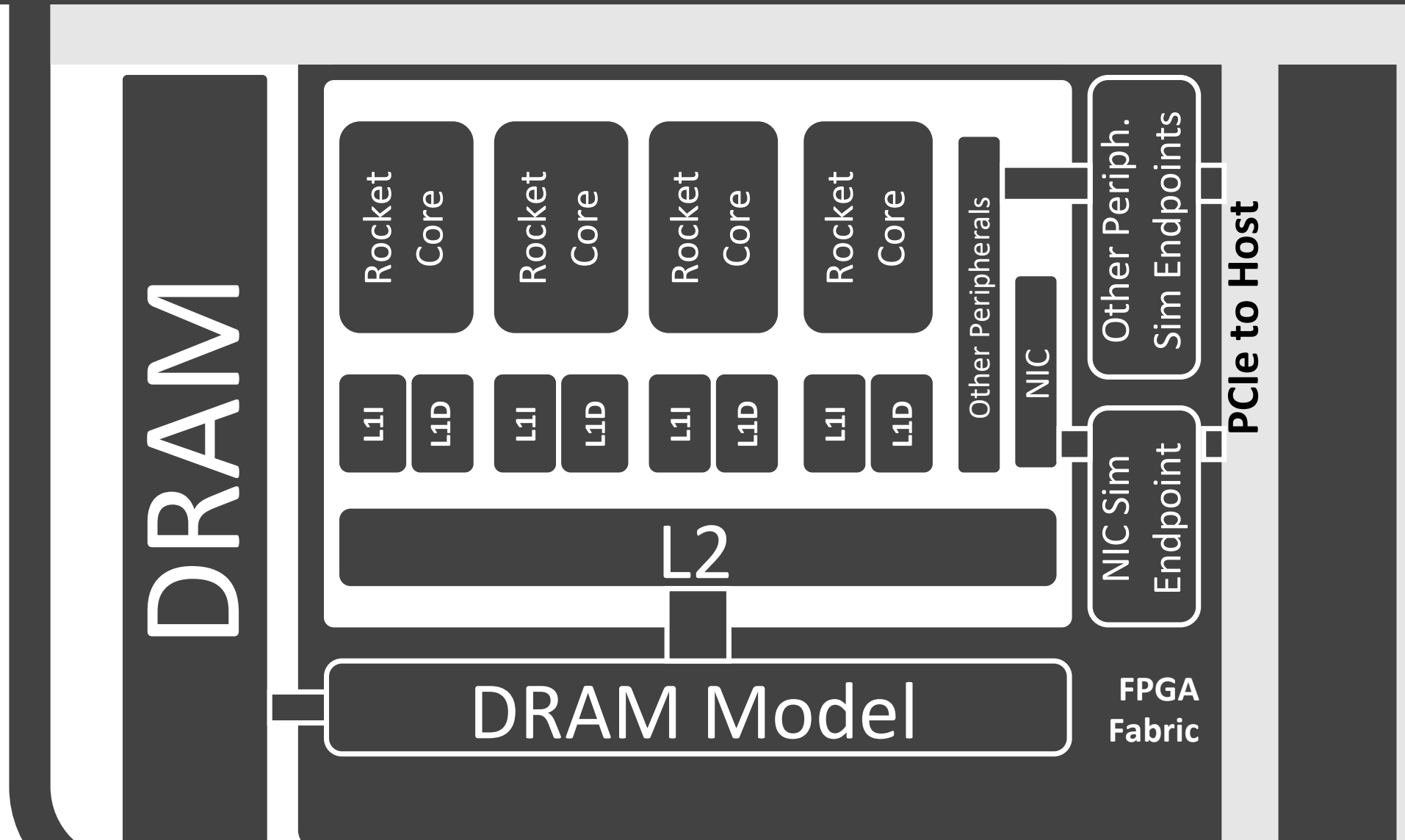
ce Util.

- an FPGA
- ¼ Mem Chans

Sim Rate

- ~150 MHz
- ~40 MHz (netw)

Step 2: FPGA Simulation of one server blade



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1$
- 256K Shared L2$
- 200 Gb/s Eth. NIC
- 16 GB DDR3

Resource Util.

- < ¼ of an FPGA
- ¼ Mem Chans

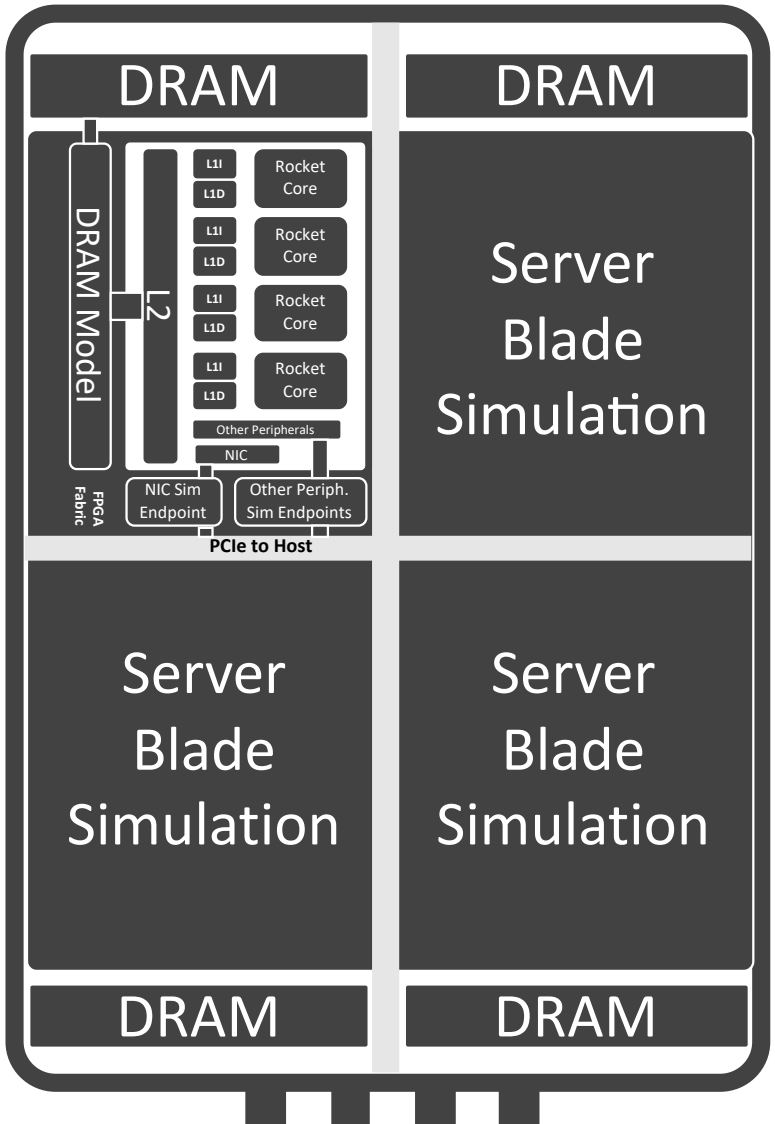
Sim Rate

- ~150 MHz
- ~40 MHz (netw)

Step 3: FPGA Simulation of 4 server blades

Cost:
$0.49 per hour (spot)

$1.65 per hour (on-demand)



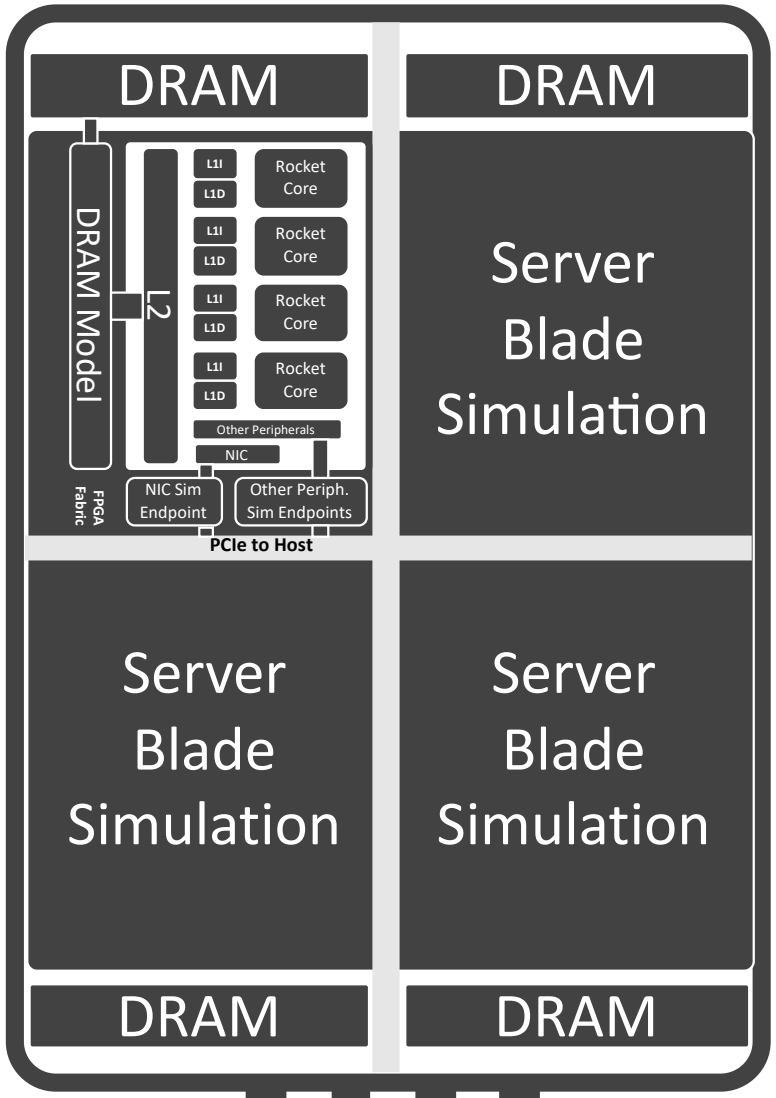
Modeled System
- 4 Server Blades
- 16 Cores
- 64 GB DDR3

Resource Util.
- < 1 FPGA
- 4/4 Mem Chans

Sim Rate
- ~14.3 MHz (netw)

Step 3: FPGA Simulation of 4 server blades

FPGA
(4 Sims)



FPGA
(4 Sim

Modeled System

- 4 Server Blades
- 16 Cores
- 64 GB DDR3

Resource Util.

- < 1 FPGA
- 4/4 Mem Chans

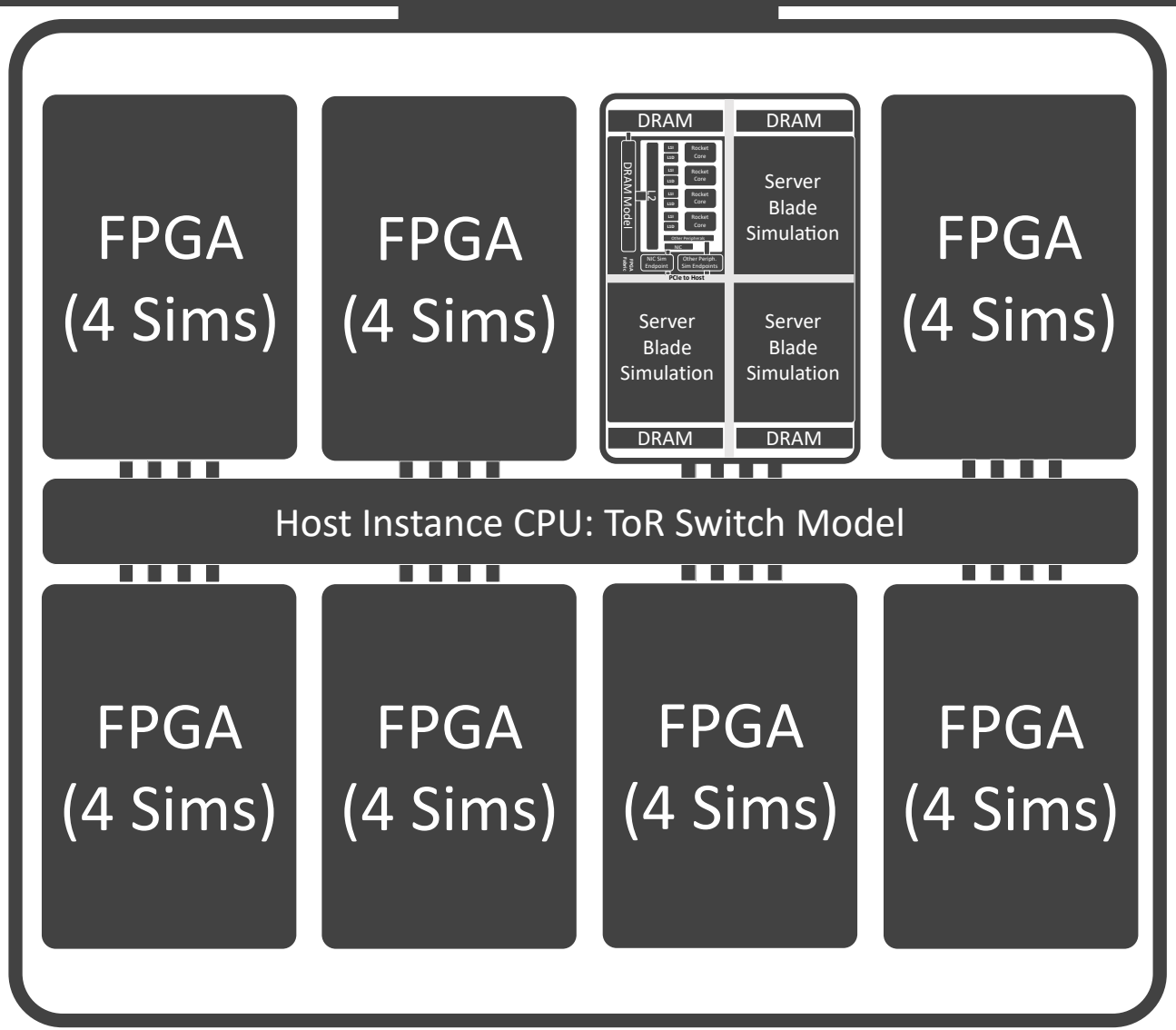
Sim Rate

- ~14.3 MHz (netw)

Step 4: Simulating a 32 node rack

Cost:
$2.60 per hour (spot)

$13.20 per hour (on-demand)



Modeled System

- 32 Server Blades
- 128 Cores
- 512 GB DDR3
- 32 Port ToR Switch
- 200 Gb/s, 2us links

Resource Util.

- 8 FPGAs =
- 1x f1.16xlarge

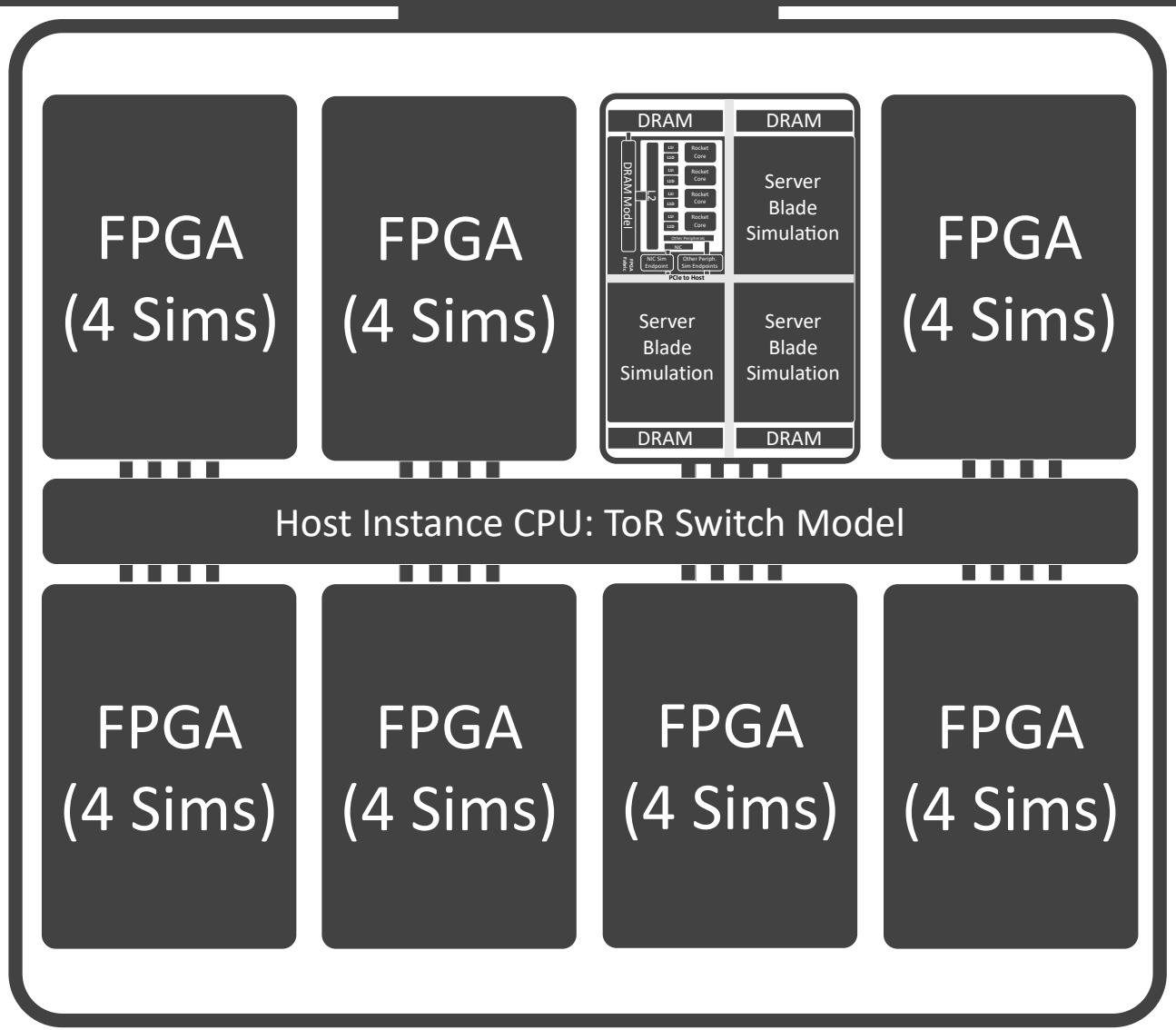
Sim Rate

- ~10.7 MHz (netw)

Step 4: Simulating a 32 node rack

Cost:
$2.60 per hour (spot)

$13.20 per hour (on-demand)



Modeled System

- 32 Server Blades
- 128 Cores
- 512 GB DDR3
- 32 Port ToR Switch
- 200 Gb/s, 2us links

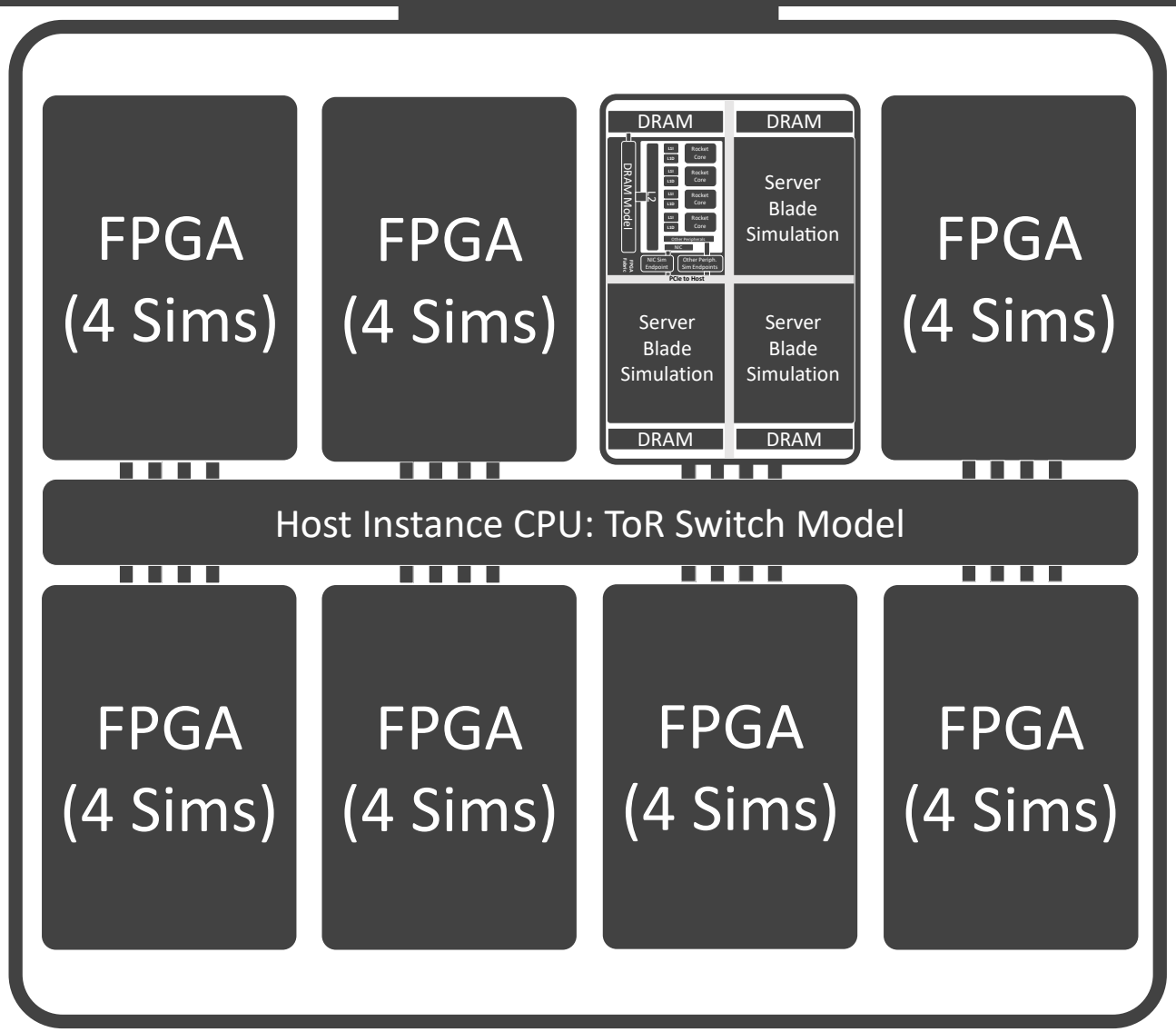
Resource Util.

- 8 FPGAs =
- 1x f1.16xlarge

Sim Rate

- ~10.7 MHz (netw)

Step 4: Simulating a 32 node rack



Modeled System

- 32 Server Blades
- 128 Cores
- 512 GB DDR3
- 32 Port ToR Switch
- 200 Gb/s, 2us links

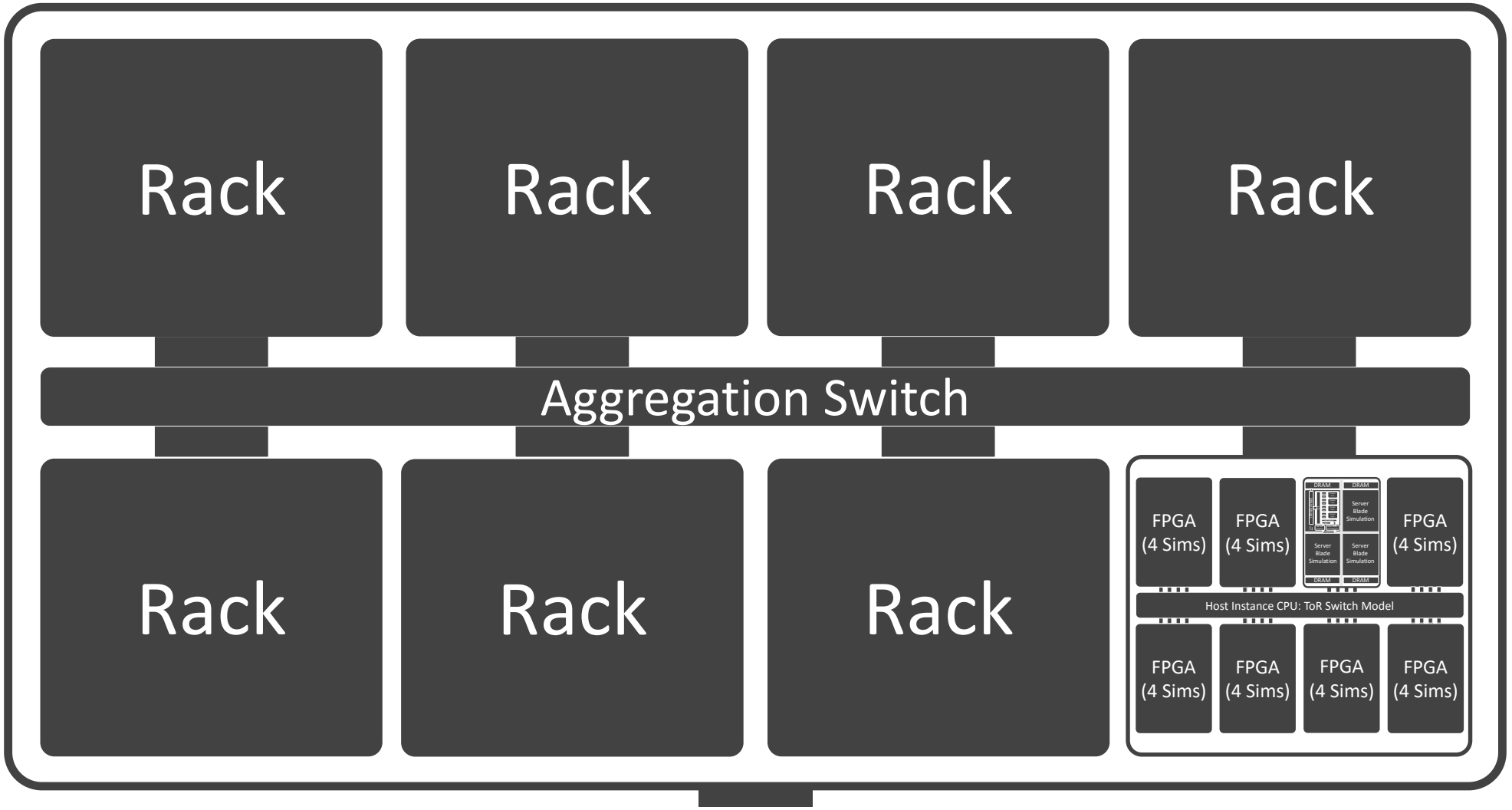
Resource Util.

- 8 FPGAs =
- 1x f1.16xlarge

Sim Rate

- ~10.7 MHz (netw)

Step 5: Simulating a 256 node “aggregation pod”



Modeled System

- 256 Server Blades
- 1024 Cores
- 4 TB DDR3
- 8 ToRs, 1 Aggr
- 200 Gb/s, 2us links

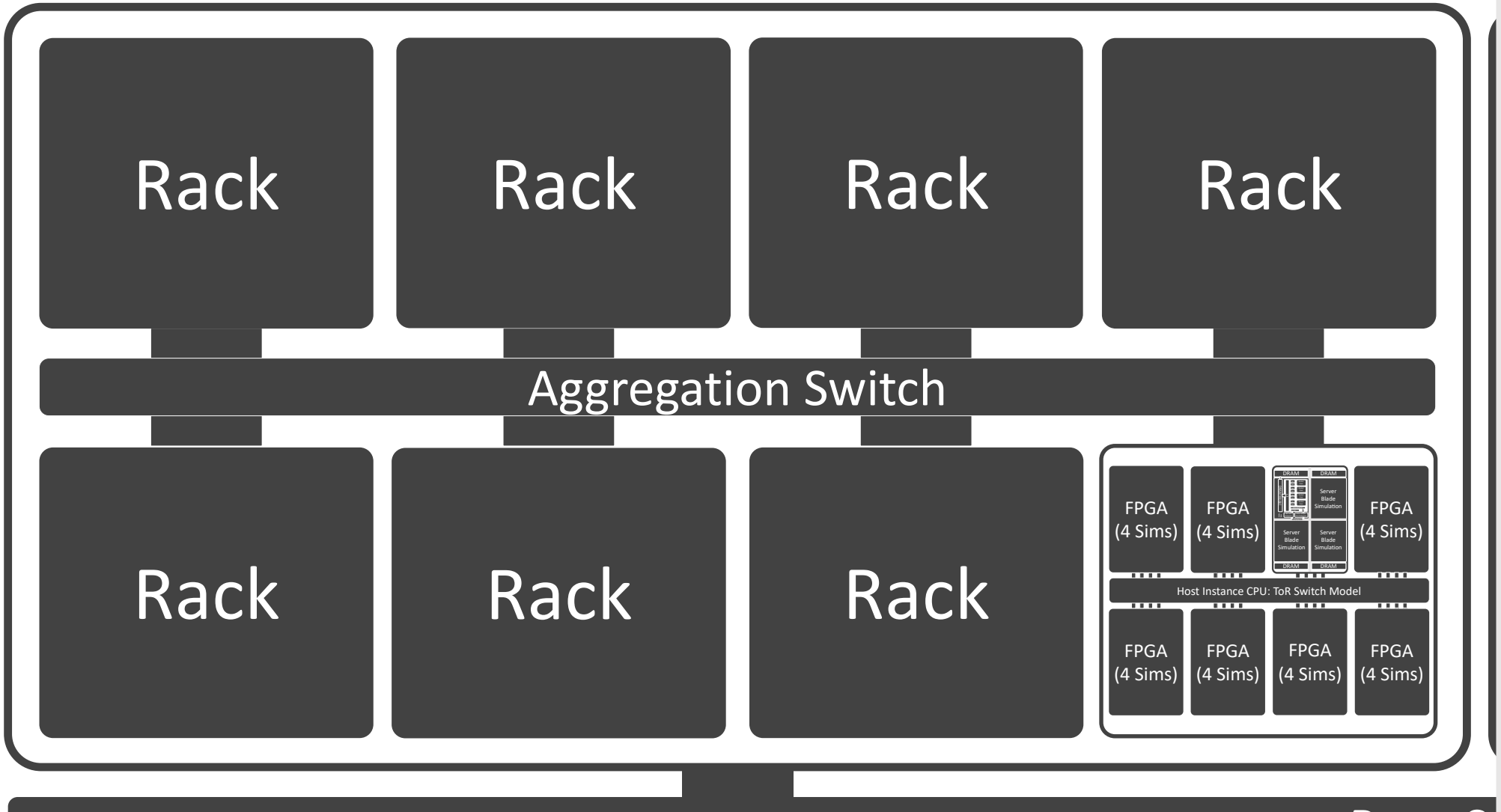
Resource Util.

- 64 FPGAs =
- 8x f1.16xlarge
- 1x m4.16xlarge

Sim Rate

- ~9 MHz (netw)

Step 5: Simulating a 256 node “aggregation pod”



Modeled System

- 256 Server Blades
- 1024 Cores
- 4 TB DDR3
- 8 ToRs, 1 Aggr
- 200 Gb/s, 2us links

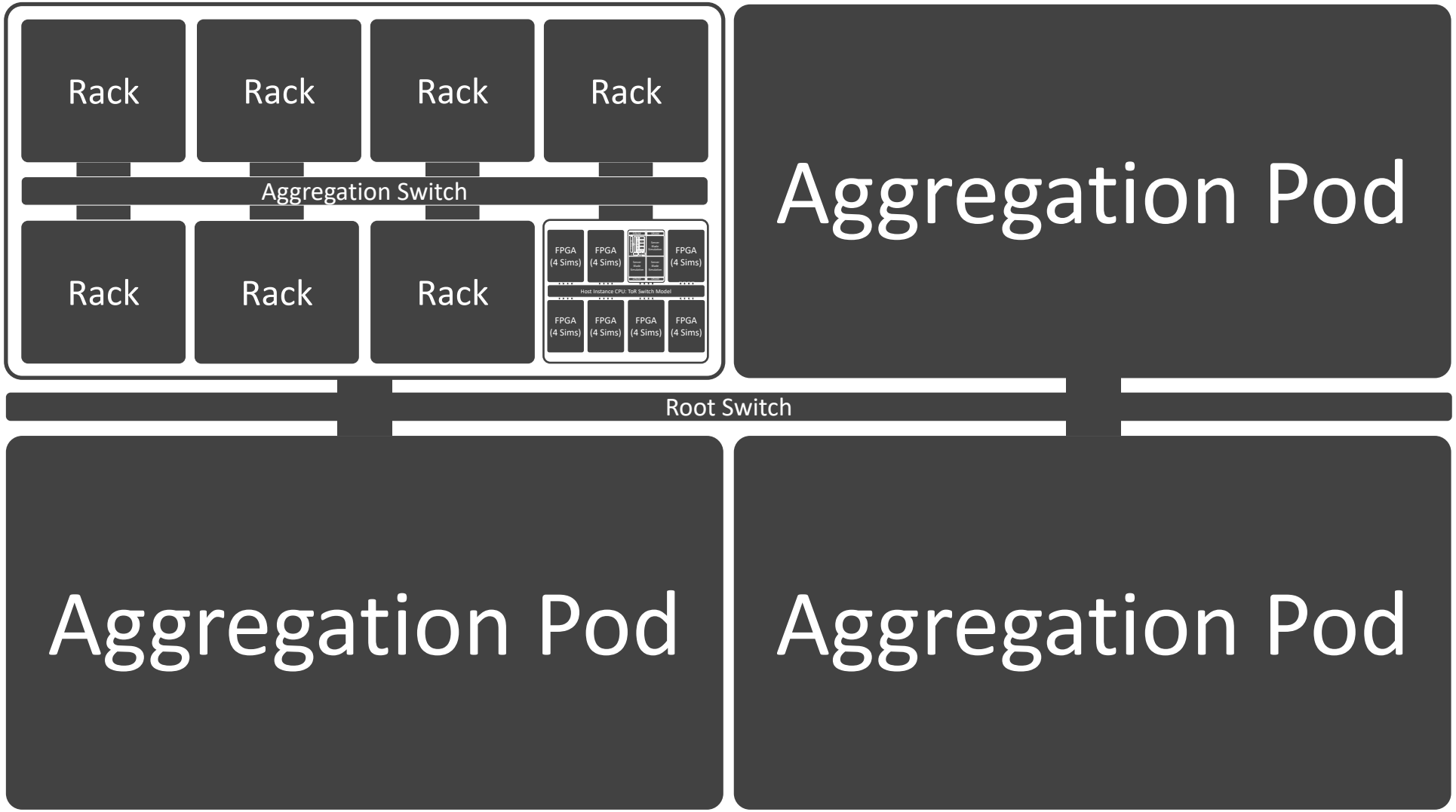
Resource Util.

- 64 FPGAs =
- 8x f1.16xlarge
- 1x m4.16xlarge

Sim Rate

- ~9 MHz (netw)

Step 6: Simulating a 1024 node datacenter



Modeled System

- 1024 Servers
- 4096 Cores
- 16 TB DDR3
- 32 ToRs, 4 Aggr, 1 Root
- 200 Gb/s, 2us links

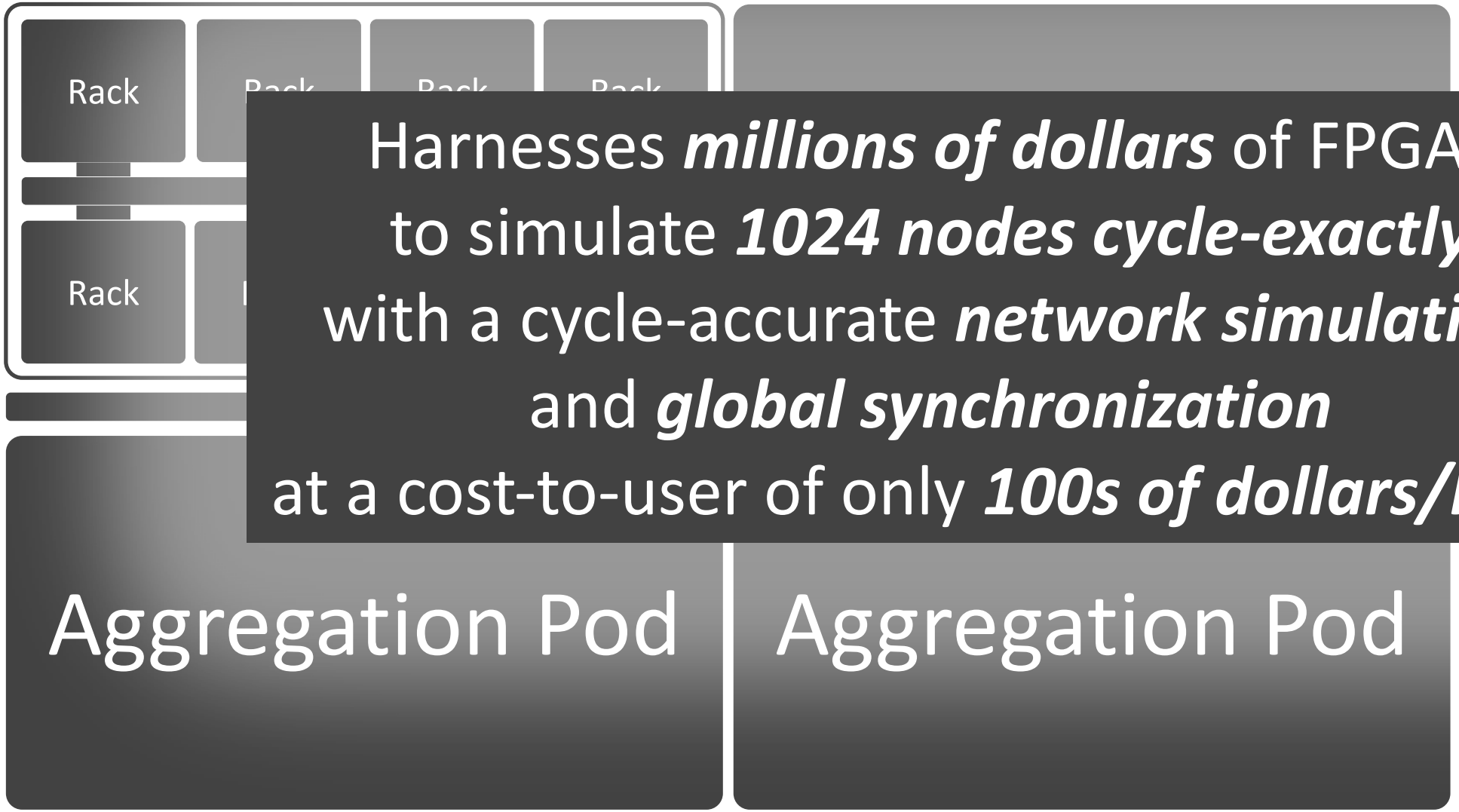
Resource Util.

- 256 FPGAs =
- 32x f1.16xlarge
- 5x m4.16xlarge

Sim Rate

- ~6.6 MHz (netw)

Step 6: Simulating a 1024 node datacenter



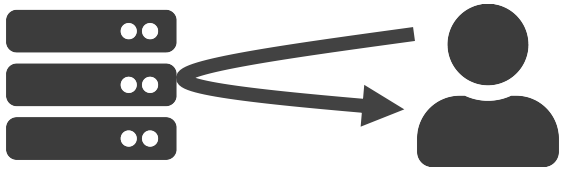
Harnesses ***millions of dollars*** of FPGAs
to simulate ***1024 nodes cycle-exactly***
with a cycle-accurate ***network simulation***
and ***global synchronization***
at a cost-to-user of only ***100s of dollars/hour***

Modeled System
- 1024 Servers
6 Cores
TB DDR3
ToRs, 4 Aggr, 1
Gb/s, 2us

urce Util.
FPGAs =
- 32x f1.16xlarge
- 5x m4.16xlarge

Sim Rate
- ~6.6 MHz (netw)

Experimenting on a 1024 Node Datacenter

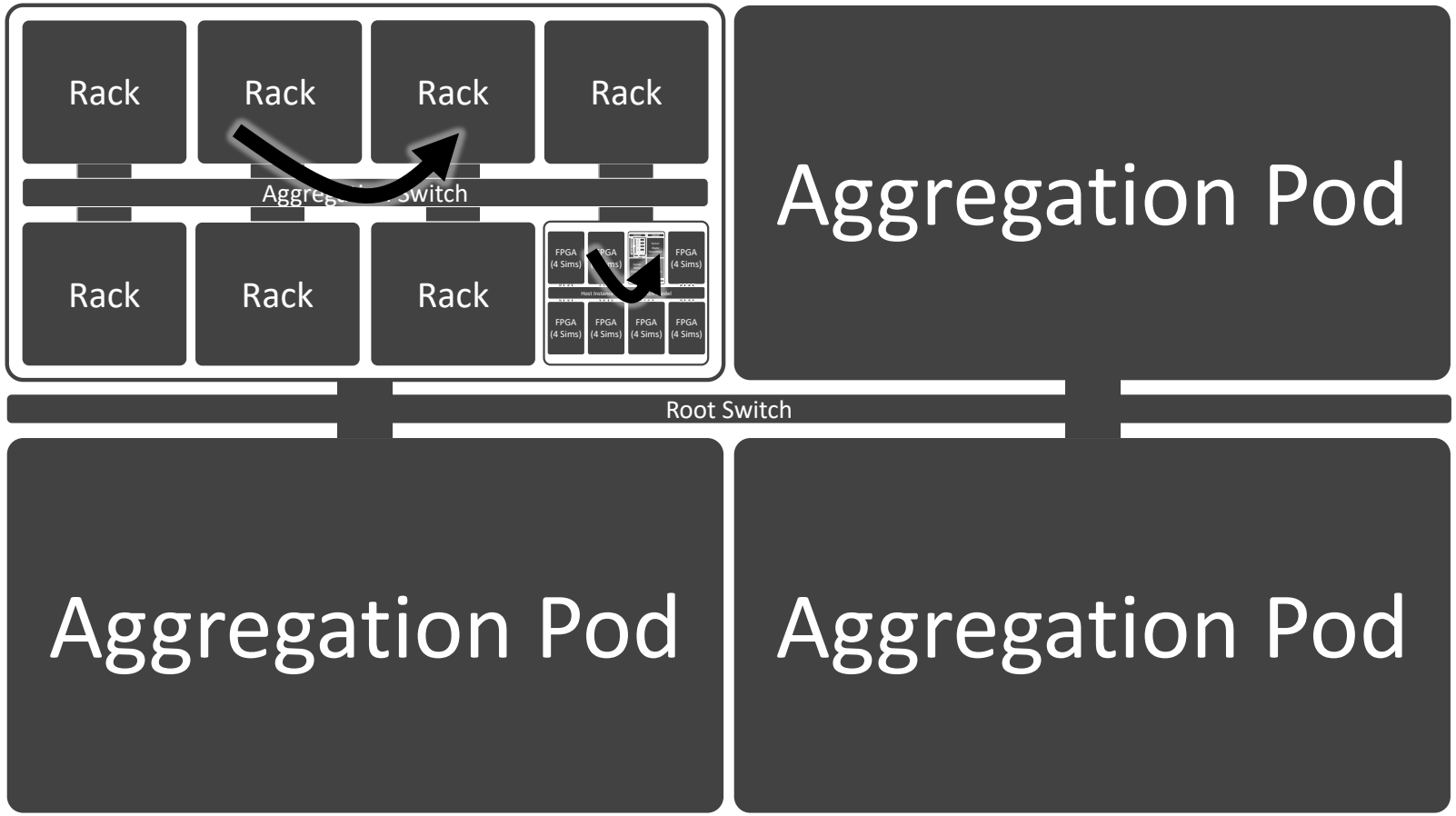


512 Memcached Servers

512 Mutilate Clients

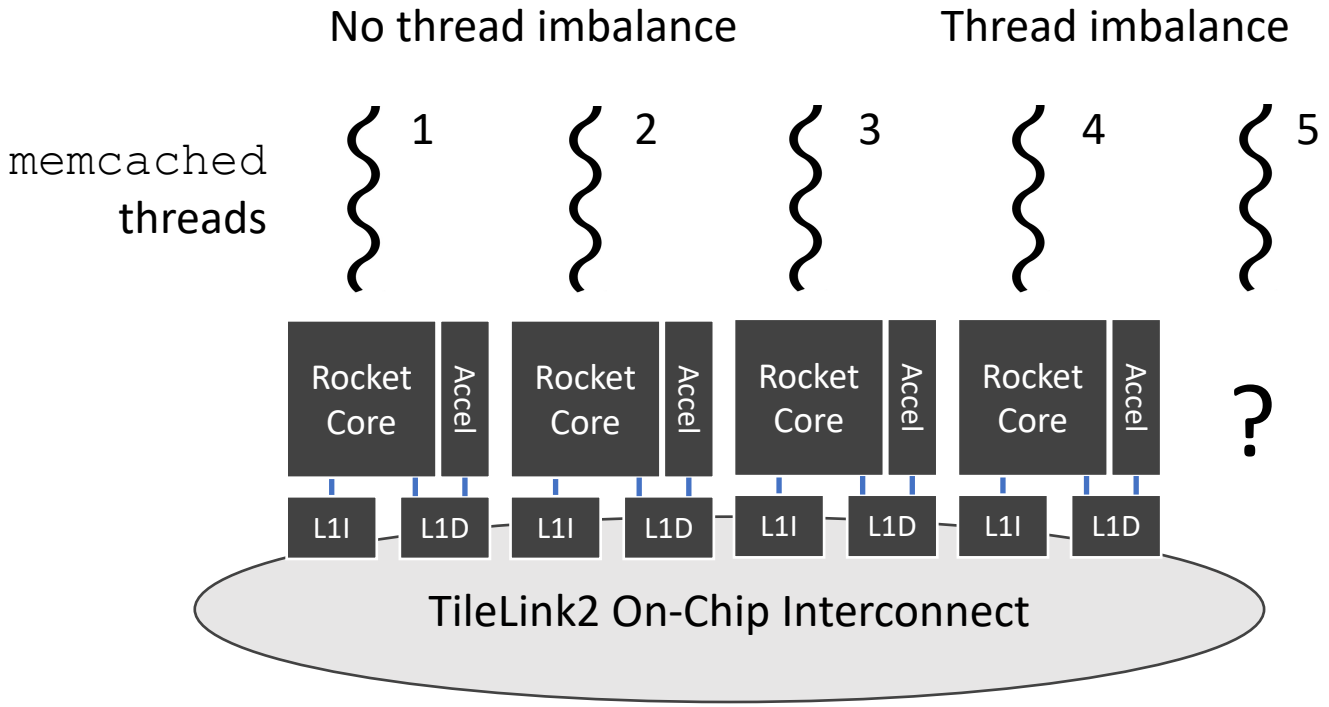
(All 1024 nodes boot Linux first)

	50th %-ile (us)
Cross-ToR	79.3



Reproducing tail latency effects from deployed clusters

- Leverich and Kozyrakis show effects of thread-imbalance in `memcached` in *EuroSys '14* [3]



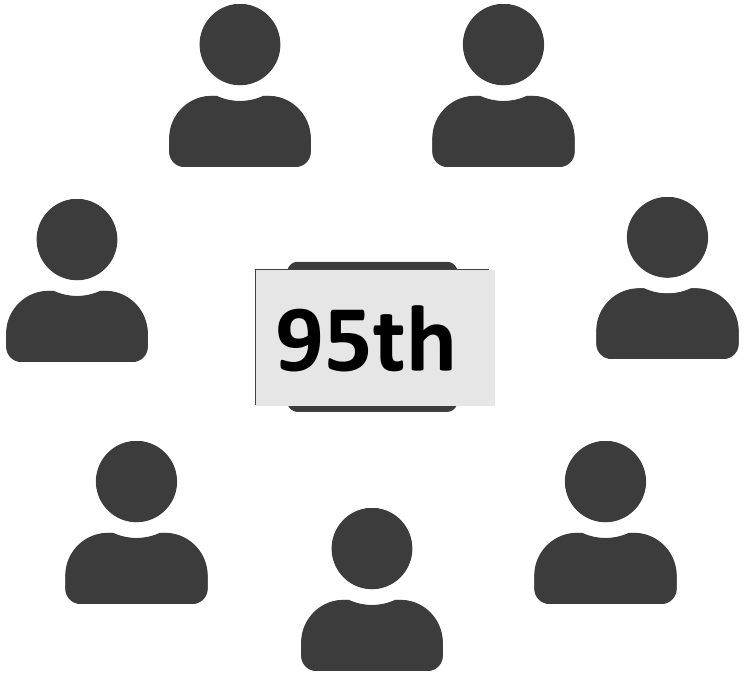
From [3], under thread imbalance, we expect:

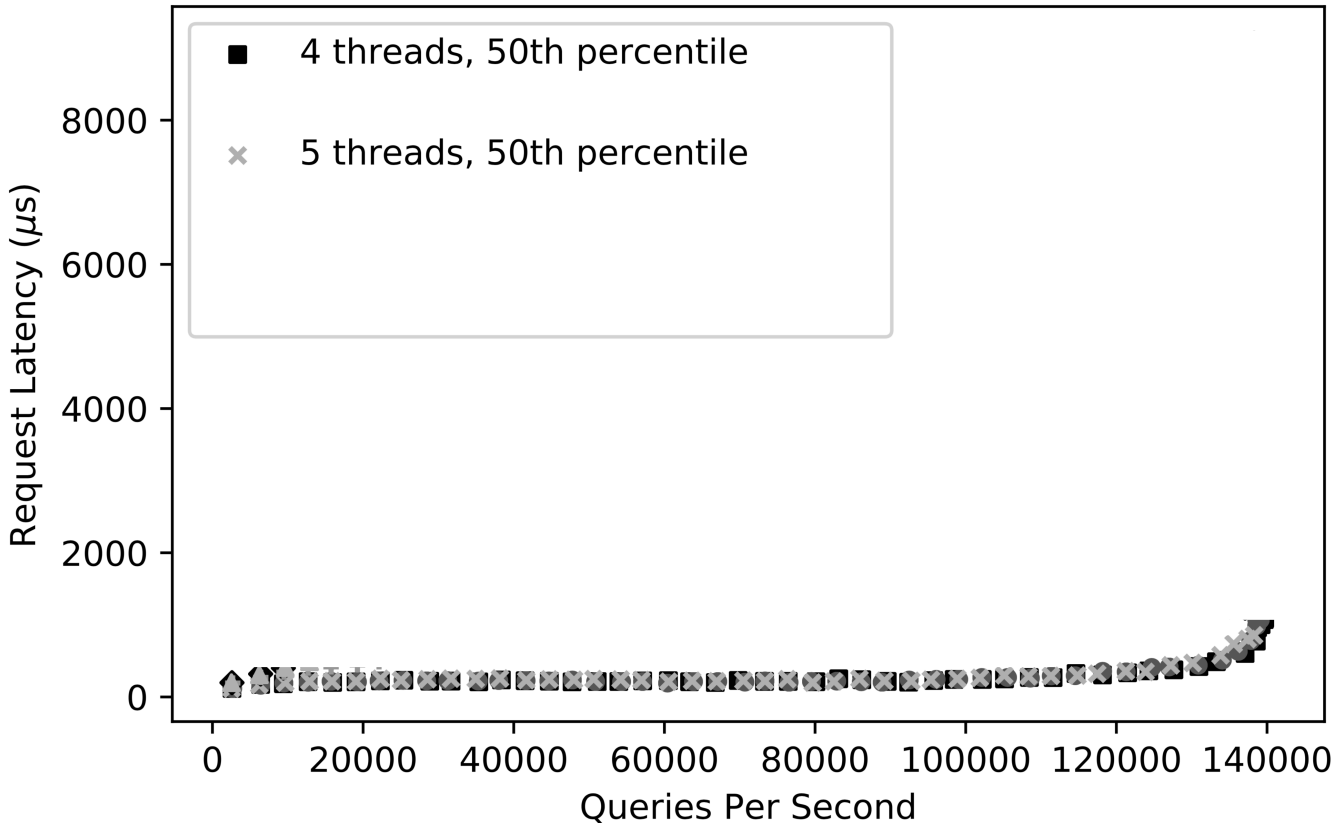
1) **Median latency** to be *unchanged*

2) **Tail latency** to *increase drastically*

Reproducing tail latency effects from deployed clusters

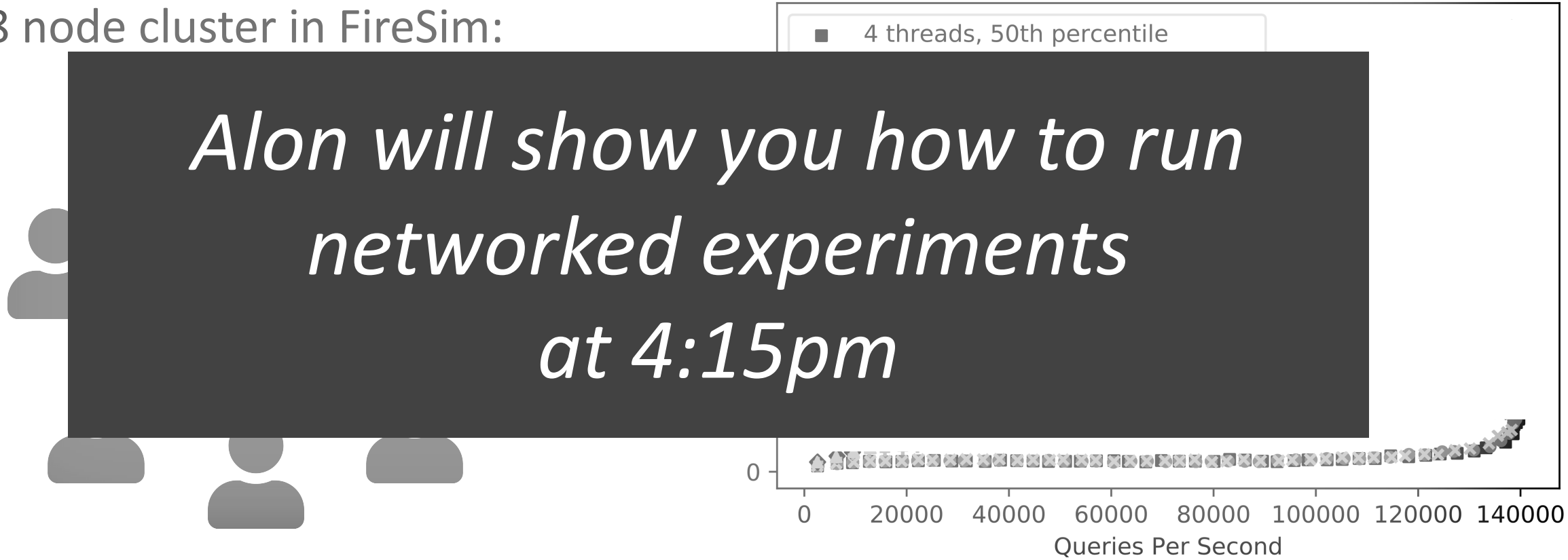
- Let's run a similar experiment on an 8 node cluster in FireSim:





Reproducing tail latency effects from deployed clusters

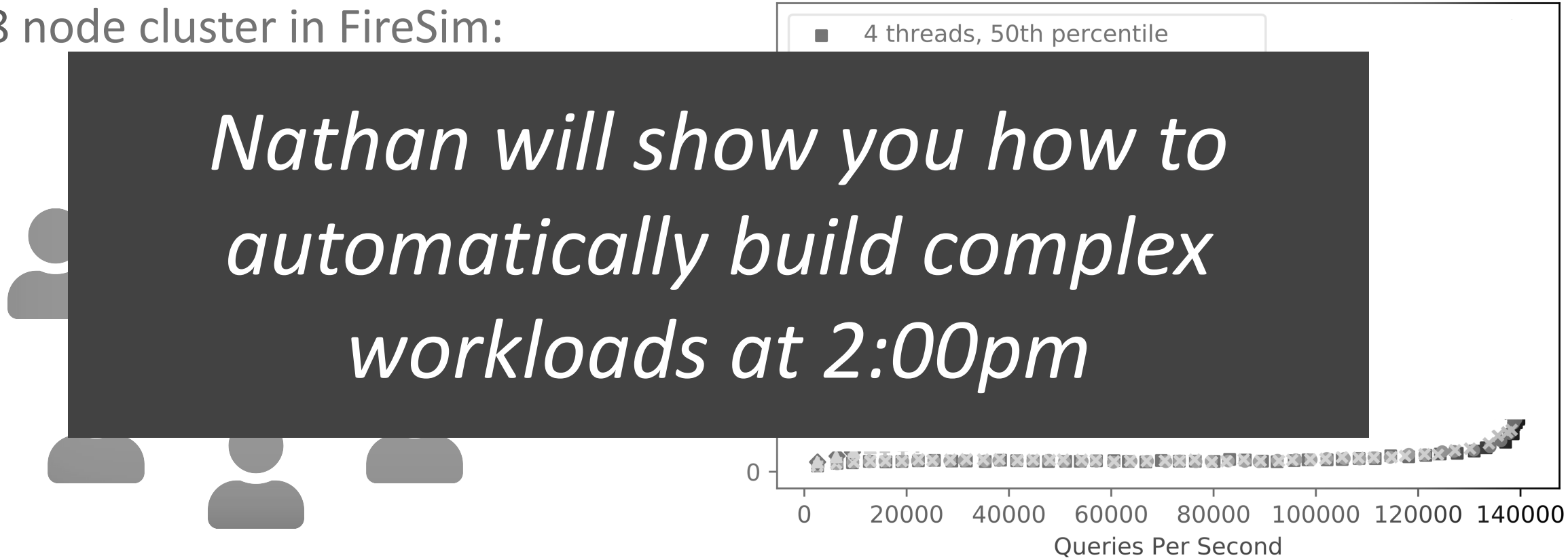
- Let's run a similar experiment on an 8 node cluster in FireSim:



Alon will show you how to run networked experiments at 4:15pm

Reproducing tail latency effects from deployed clusters

- Let's run a similar experiment on an 8 node cluster in FireSim:

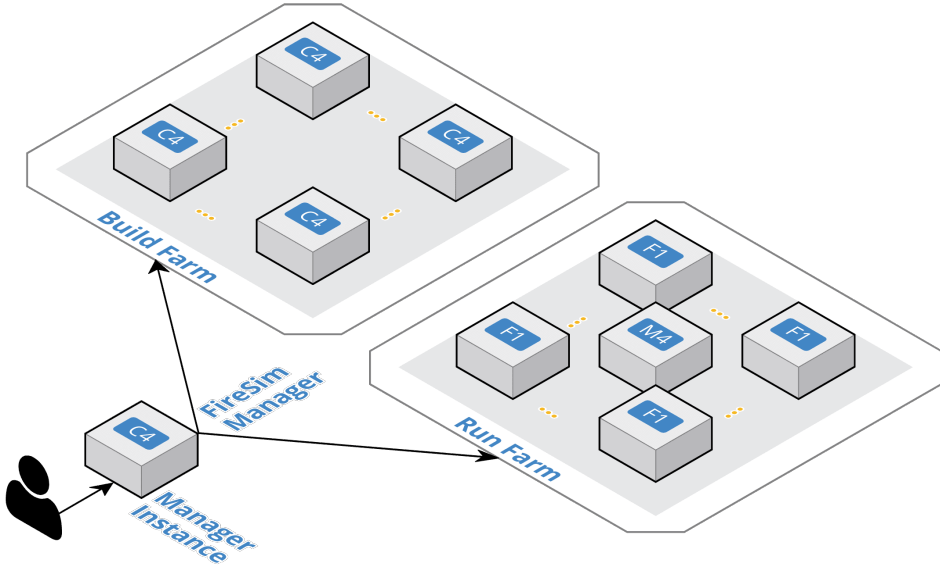


Nathan will show you how to automatically build complex workloads at 2:00pm

Open-Source

Productive Open-Source FPGA Simulation

- github.com/firesim/firesim, BSD Licensed
- An "easy" button for fast, FPGA-accelerated full-system simulation, *including single-node designs*
  - Plug in your own RTL designs, your own HW/SW models
  - One-click: Parallel FPGA builds, Simulation run/result collection, building target software
  - Scales to a variety of use cases:
    - Networked (performance depends on scale)
    - Non-networked (150+ MHz), limited by your budget
- `firesim` command line program
  - Like `docker` or `vagrant`, but for FPGA sims
  - User doesn't need to care about distributed magic happening behind the scenes

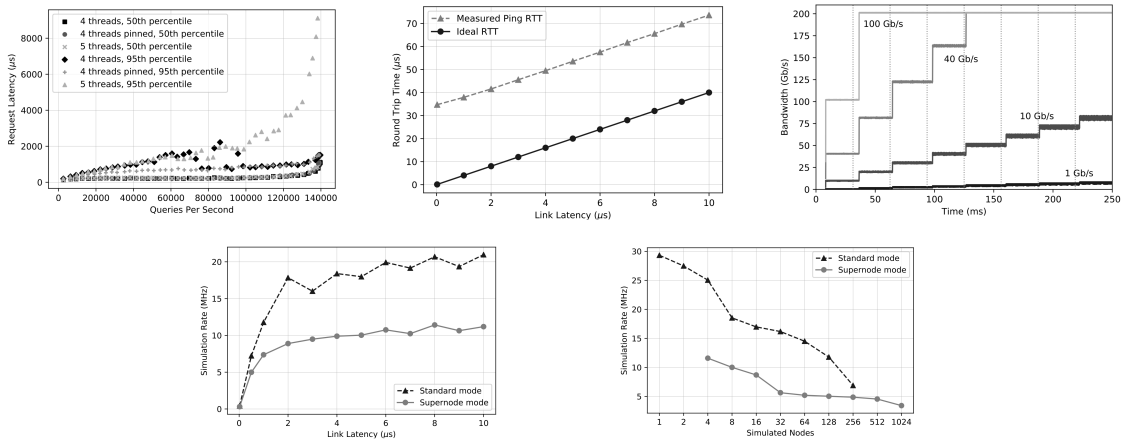


FireSim Developer Environment

Productive Open-Source FPGA Simulation

- Scripts can call `firesim` to fully automate distributed FPGA sim
  - **Reproducibility**: included scripts to reproduce ISCA 2018 results
  - e.g. scripts to automatically run SPECInt2017 **reference inputs** in ≈1 day
  - Many others
- 100+ pages of documentation: https://docs.fires.im
- AWS provides grants for researchers: https://aws.amazon.com/grants/

```
$ cd fsim/deploy/workloads
$ ./run-all.sh
```



What about *non-datacenter* targets?

Example use case: Evaluating SoC Designs w/SPEC

Get SPECInt2017 results w/reference inputs on Rocket Chip (or other CPU design) within a day:

- Automatically using 10 or 11 FPGAs in parallel for `intrate` or `intspeed`
- FireSim manager takes care of all the heavy lifting/tedious management

What does it take to run SPECInt2017 rate on 10 rocket chips with FireSim?

```
cd firesim/deploy/workloads
make spec17-intrate                    # build a disk image for each spec bmark
firesim -c spec17intrate.ini launchrunfarm        # launch 10 FPGA instances
firesim -c spec17intrate.ini infrasetup    # copy root FS, linux, flash FPGAs
firesim -c spec17intrate.ini runworkload   # run 10 bmarks on 10 FPGA
                                           # instances, collect all results
# when this completes, all run logs/performance results found in
# firesim/deploy/results-workload/TIMESTAMP-spec17intrate/
# This takes < 1 day, with reference inputs (many benchmarks take much less)
```

Example use case: Evaluating SoC Designs w/SPEC

Get SPECInt2017 results w/reference inputs on Rocket Chip (or other CPU design) within a day:

- Automatically using 10 or 11 FPGAs in parallel for `intrate` or `intspeed`
- FireSim manager takes care of all the heavy lifting/tedious management

Albert will show you how to run SPEC17 quickly at 2:30pm

What doe

```
cd firesi
make spec                                                    omark
firesim -                                                    ances
firesim -                                                    FPGAs
firesim -c spec17intrate.ini runworkload    # run 10 bmarks on 10 FPGA
                                            # instances, collect all results
# when this completes, all run logs/performance results found in
# firesim/deploy/results-workload/TIMESTAMP-spec17intrate/
# This takes < 1 day, with reference inputs (many benchmarks take much less)
```

Easy to quickly collect large amts. of data on real designs, even for a few grad students

- e.g. in our FASED [3] paper:

Benchmarks	Insns (T)		D$ MPKI		I$ MPKI	
perlbench	2.98	2.99	9.0	8.9	10.0	10.1
gcc	2.43	1.35	36.6	29.5	9.7	11.1
mcf	1.60	0.91	97.9	80.9	0.1	0.1
omnetpp	1.11	1.11	56.9	56.6	9.3	10.4
xalancbmk	1.21	1.21	62.9	62.9	7.9	7.6
x264	4.55	4.55	3.0	3.0	2.9	3.0
deepsjeng	2.51	2.14	8.7	8.2	15.4	15.3
leela	2.59	2.59	5.8	5.8	1.5	1.5
exchange2	3.24	3.24	0.0	0.0	0.1	0.1
xz	9.41	2.25	19.8	15.7	0.2	0.1

Table 5: Dynamic instruction counts and L1 MPKIs of SPEC2017int rate and speed (single threaded), respectively.

[3] Biancolin et. al., "FASED: FPGA-accelerated Simulation and Evaluation of DRAM", FPGA'19

Easy to quickly collect large amts. of data on real designs, even for a few grad students

- e.g. in our FASED [3] paper:

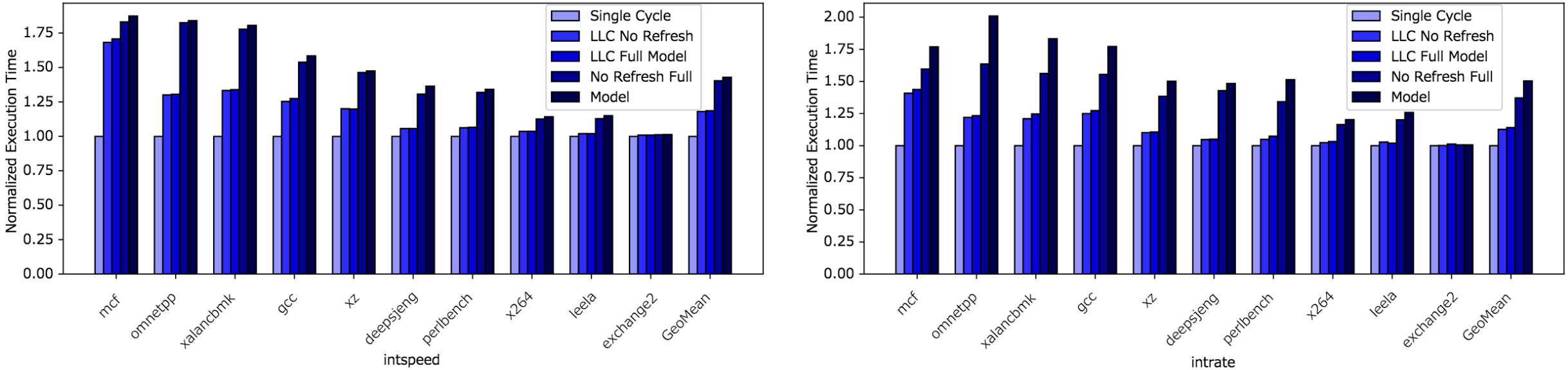


Figure 5: Target-execution time of SPEC2017 intspeed and intrate (4 copies) with reference inputs for DRAM models with and without refresh enabled. Runtime is normalized to that of a single-cycle memory system. LLCs, if present, are 256 KiB and 1 MiB large for intspeed and intrate respectively and are 8-way set associative.

[3] Biancolin et. al., "FASED: FPGA-accelerated Simulation and Evaluation of DRAM", FPGA'19

Easy to quickly collect large amts. of data on real designs, even for a few grad students

- e.g. in our FASED [3] paper:

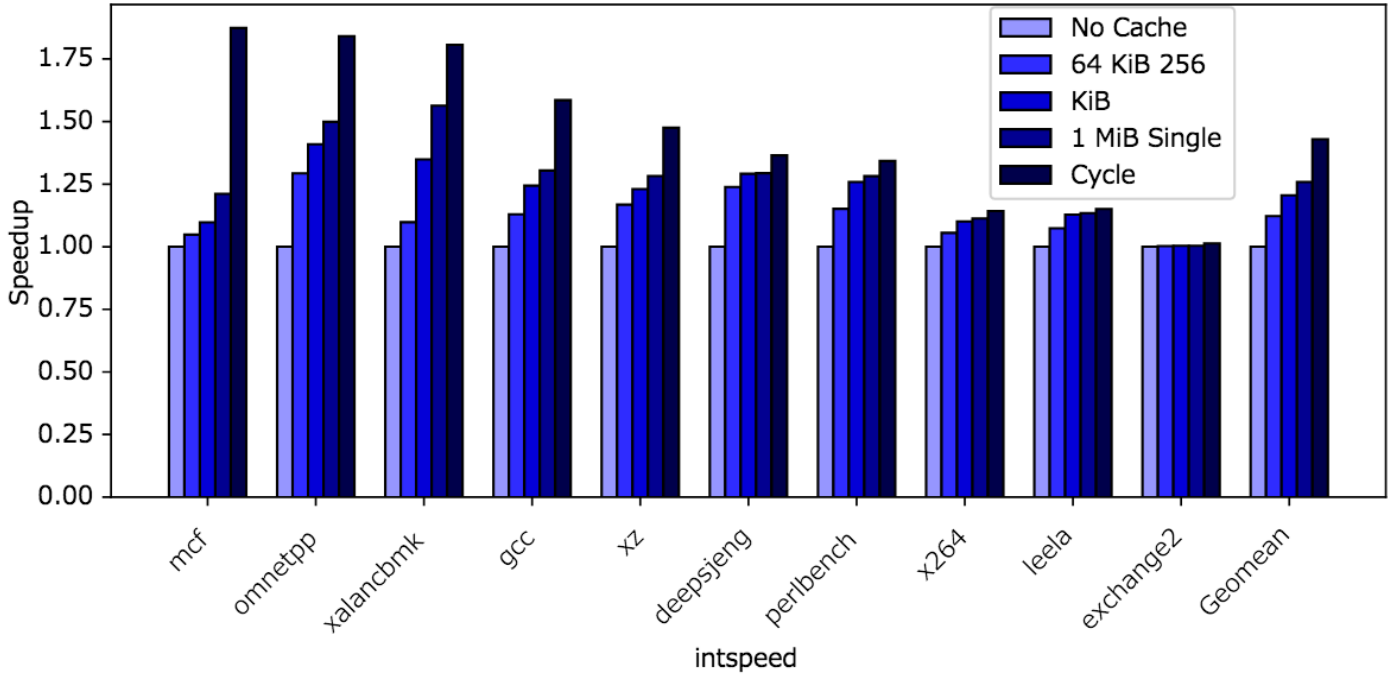


Figure 6: Speedup in SPEC2017 intspeed (reference inputs) vs LLC model size. All caches are 8-way set associative.

[3] Biancolin et. al., "FASED: FPGA-accelerated Simulation and Evaluation of DRAM", FPGA'19

Easy to quickly collect large amts. of data on real designs, even for a few grad students

- e.g. in our FASED [3] paper:

		perlbench		gcc		mcf		omnetpp		xalancbmk		x264		deepsjeng		leela		exchange2		xz	
	Model Type	hr	f	hr	f	hr	f	hr	f	hr	f	hr	f	hr	f	hr	f	hr	f	hr	f
Speed	Single Cycle	14.4	95	20.4	73	24.7	62	13.8	67	14.0	68	12.9	123	13.4	87	8.6	119	6.9	153	50.5	90
	FCFS-256KB	14.7	100	20.8	91	25.6	102	14.1	86	14.5	88	13.1	125	13.6	91	8.6	121	6.9	153	51.8	105
	FCFS	14.7	126	20.9	113	25.7	112	14.3	119	14.7	118	13.2	137	13.6	117	8.7	135	7.0	152	52.1	110
Rate	Single Cycle	31.6	50	28.5	38	33.3	36	37.1	35	36.6	37	20.8	79	25.8	44	14.9	73	7.0	151	22.6	51
	FRFCFS-1MB	31.2	54	24.4	57	23.9	72	32.5	49	31.4	54	20.4	84	24.8	48	14.3	78	7.1	150	20.7	62
	FRFCFS	21.6	111	19.6	98	20.2	104	27.3	96	22.8	110	15.9	125	15.4	110	11.4	120	7.0	152	16.4	107

Table 6: Simulation times (hours) and rates (f, MHz) for SPEC2017 intspeed and intrate (four copies) running on single and quad-core Rocket Chip targets. In all cases, the FPGA-host frequency is 160 MHz.

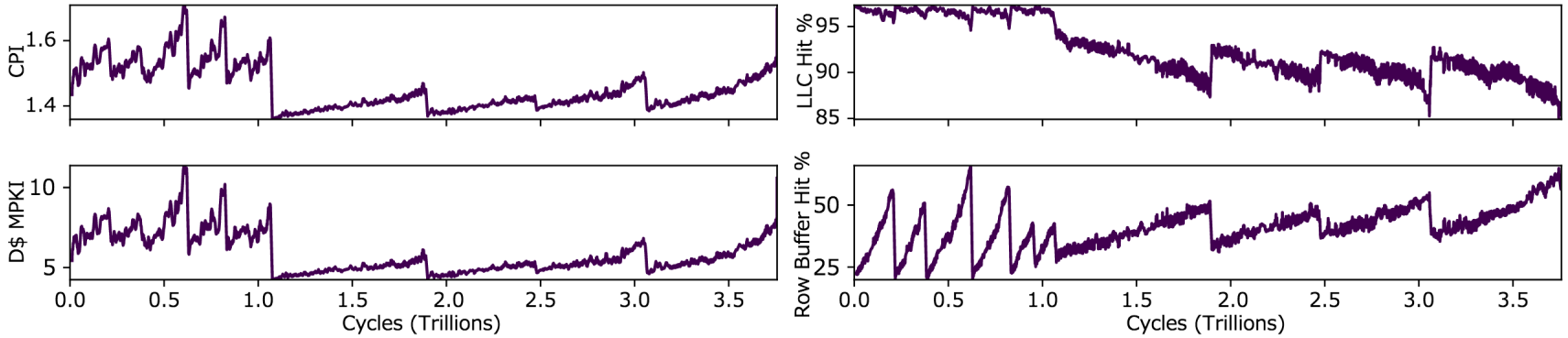


Figure 7: CPI, D$ MPKI, and row buffer and LLC hit rates running 641.leela_s with on Rocket with 256 KiB of LLC and a FCFS MAS model. These plots use a rolling average of 10 samples spaced a billion cycles apart.

[3] Biancolin et. al., "FASED: FPGA-accelerated Simulation and Evaluation of DRAM", FPGA'19

Post-lunch schedule

13:00 - 13:30: FireSim Introduction - Sagar

13:30 – 14:00: Building Hardware Designs in FireSim - David

14:00 – 14:30: Building Software Workloads in FireSim - Nathan

14:30 – 15:00: Running a FireSim Simulation: Password Cracking on a RISC-V SoC with SHA-3 Accelerators and Linux - Albert

15:00 – 15:30: Coffee break

15:30 – 16:15: Instrumenting and Debugging FireSim-Simulated Designs - Alon

16:15 – 16:55: FireSim Multi-FPGA Networked Simulation - Alon

16:55 – 17:00: Conclusion - Alon

FireSim

See you after lunch!

Berkeley Architecture Research

Learn More:

Web: https://fires.im

Docs: https://docs.fires.im

GitHub: https://github.com/firesim/firesim

Mailing List: https://groups.google.com/forum/#!forum/firesim

@firesimproject

Email: sagark@eecs.berkeley.edu

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000849, and by DARPA, Award Number HR0011-12-2-0016. Research was also partially funded by ADEPT Lab industrial sponsors and affiliates Intel, Apple, Futurewei, Google, and Seagate, and RISE Lab sponsor Amazon Web Services. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.