Warehouse-Scale Computers

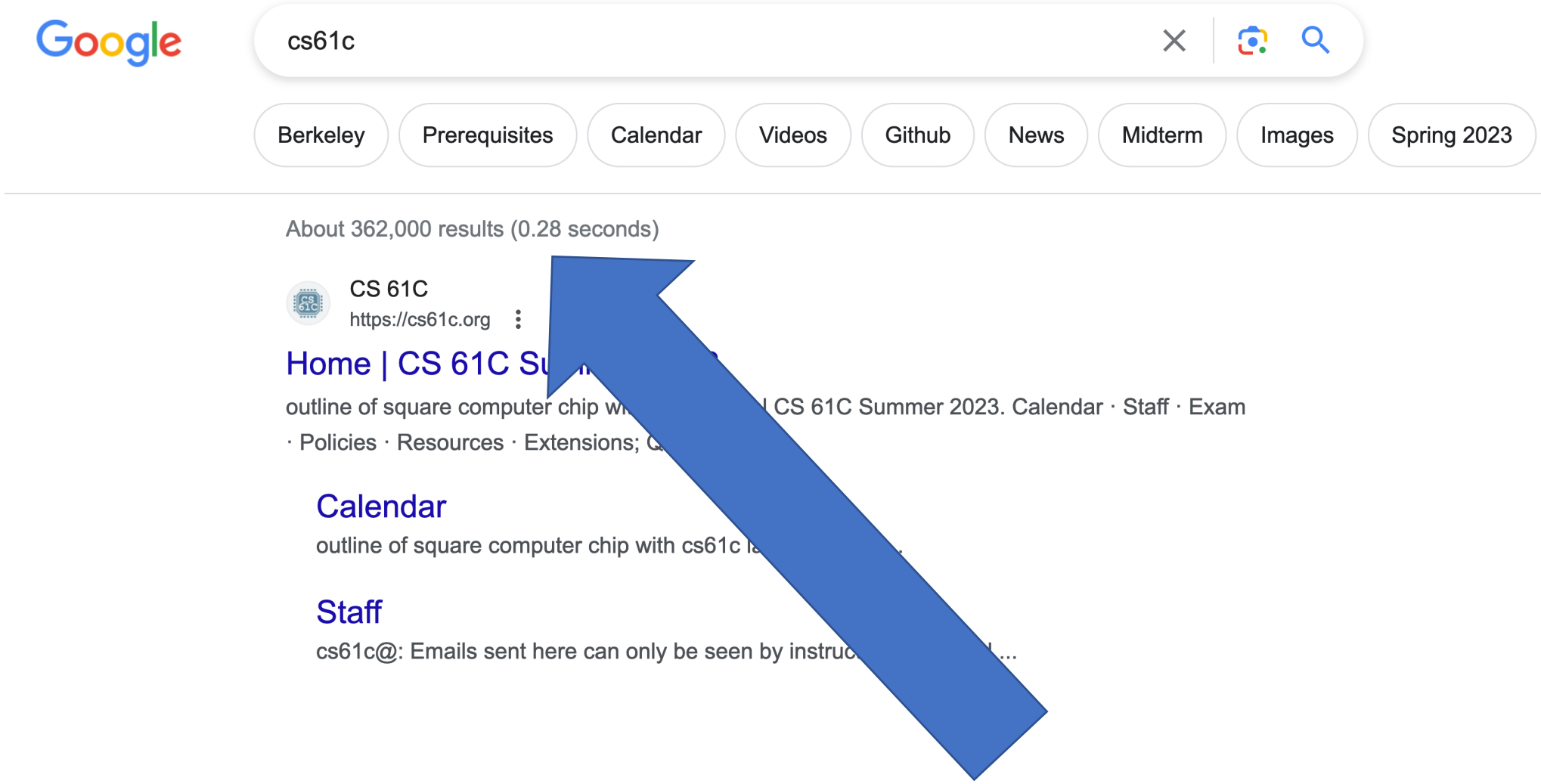
Sagar Karandikar

CS61C Guest Lecture

August 2, 2023

Image: A Google Datacenter in Eemshaven, Netherlands. https://www.google.com/about/datacenters/gallery/

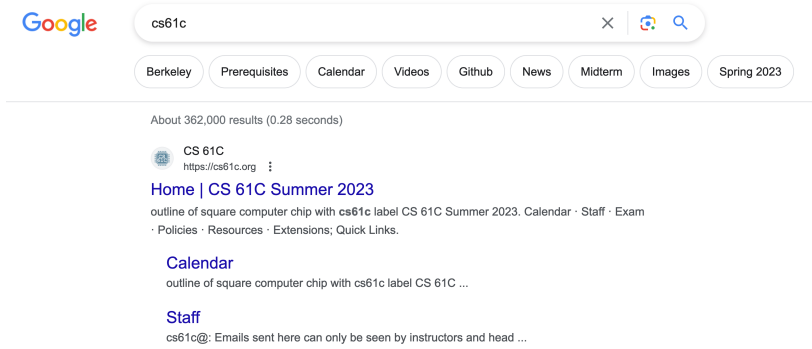
How is this possible? *WSCs*



WSCs power the apps you use every day

NETFLIX

Google Maps



Traditional Datacenters

- Co-location of servers and networking infrastructure
- *Traditional Datacenters* commonly consist of isolated clusters with varying HW components, owned by arbitrary third-parties
  - Usually do not communicate with each other
  - Usually don’t have a unifying management infrastructure
- DC-operator provides co-location in order to reduce overheads of:
  - Maintenance
  - Environmental requirements
  - Physical security

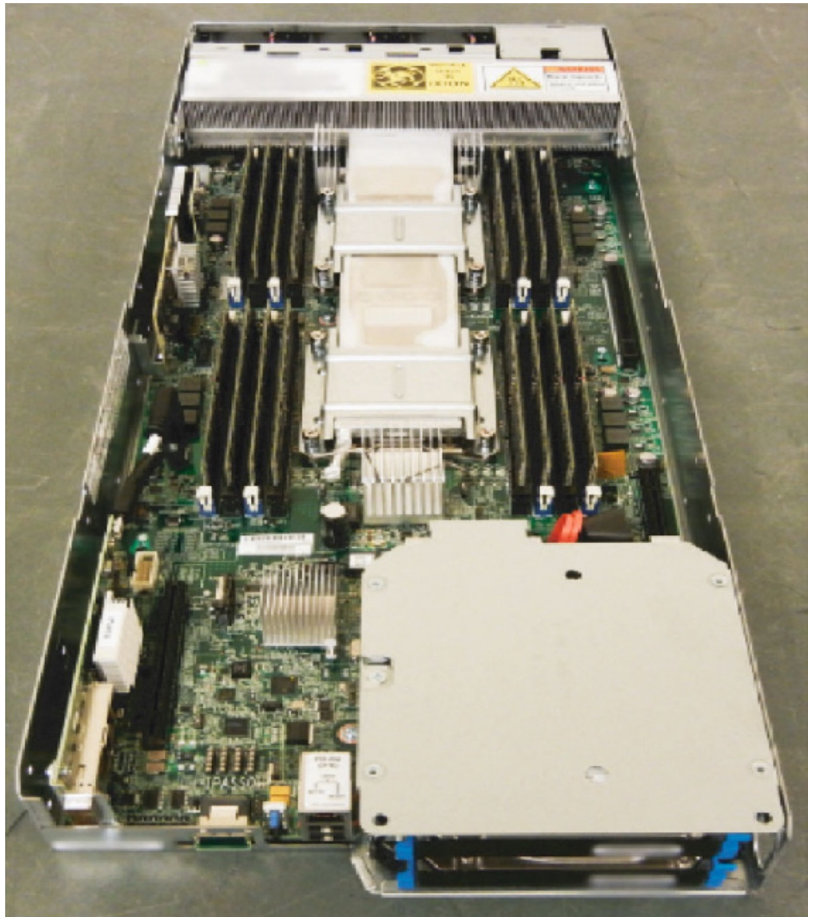
Warehouse-Scale Computers (WSCs)

- Also a building full of servers + networking equipment
- Belong to a single organization (called “Hyperscalers”)
- Use a homogeneous hardware platform*
  - *Or at least one designed by one entity
- Common systems-level software platforms
- Common management layers
- Run large, in-house services designed for the WSC
  - Usually a smaller number of very large internet services
- Many also offer *public cloud computing*: renting VMs w/compute, memory, storage, etc. to customers
- Heavy reliance on *Request-Level Parallelism*

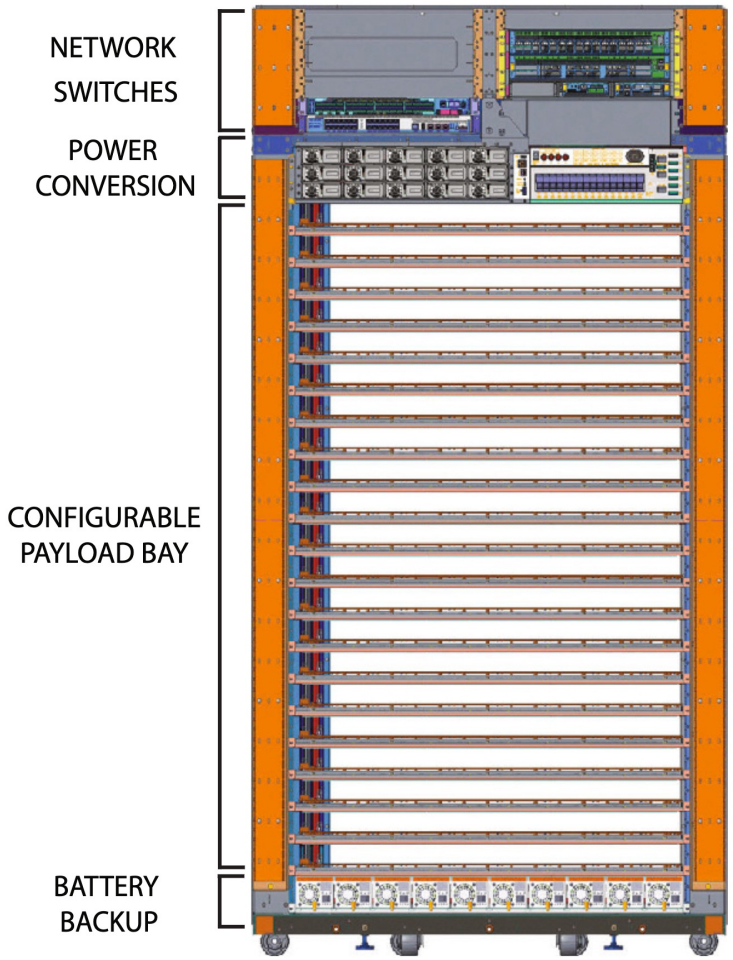
What’s in a WSC?

Servers

- Like, a desktop PC, but in “blade” enclosure
  - Dual-socket CPU, ~10 DIMM slots (memory), PCIe cards, power, cooling, etc.
- Prefer mid-range servers for cost-efficiency
- Share components with PC market



Racks



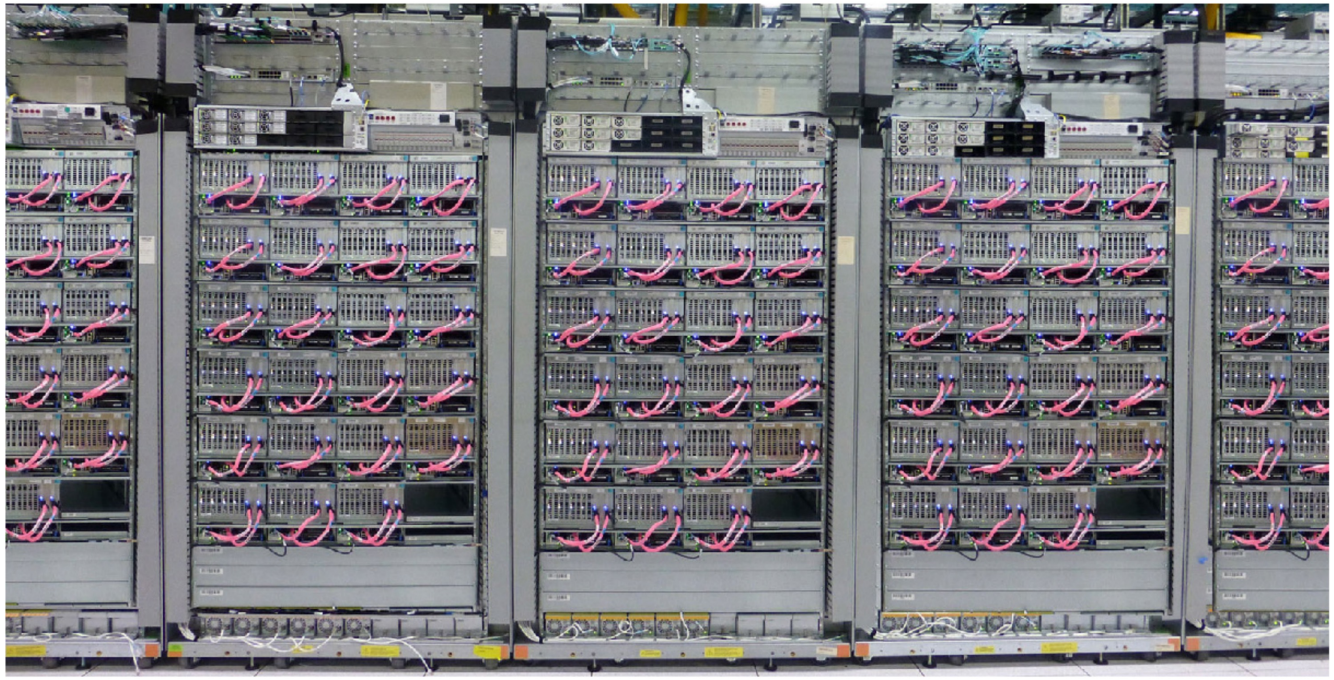
- Hold tens of servers together
- Provides:
  - Power delivery
  - Battery backup
  - Power conversion
- Commonly have a ToR (“top-of-rack”) networking switch
  - E.g., Google’s Jupiter network with 64 port, 40 Gbps ToR switches

Accelerators in WSCs

- Traditionally, little motivation to add acceleration in WSCs
- Few workloads could benefit (we could just use the same space/power/money to get more servers that can run anything)
- Servers got reasonably faster every few years w/traditional scaling techniques and microarchitectural improvements
- Large management/engineering overhead to introducing (the first) heterogeneity:
  - Cluster management now needs to be aware of different types of compute
  - WSC application developer needs to be (at least somewhat) aware of accelerator usage
- Today: the end of traditional scaling + demand for ML = huge proliferation of custom accelerators in WSCs

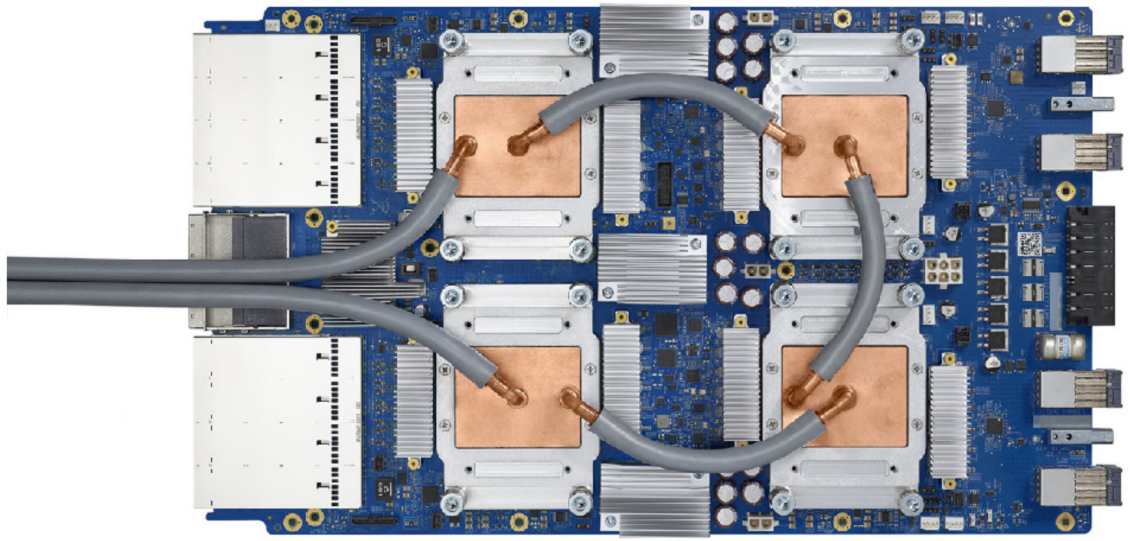
GPUs

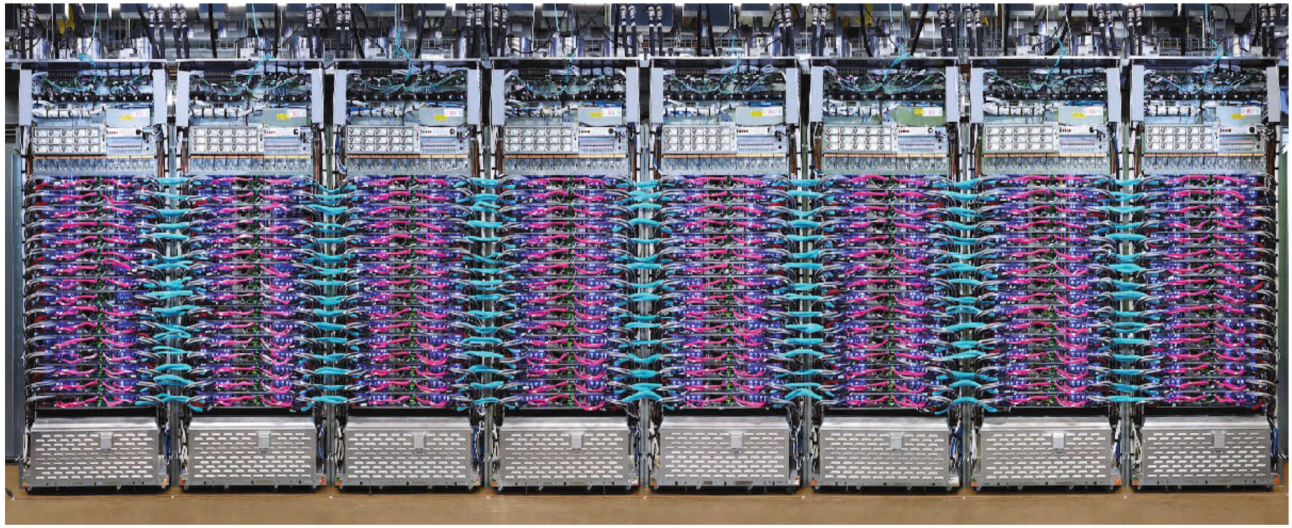
- Available in many WSCs for ML/AI and other acceleration
- Easier “first pass” at integrating heterogeneous compute (not custom built by WSC operator)
- Can be connected to host server over PCIe + high-BW custom interconnect like Nvlink
- Easy to expose to public cloud user or WSC service developer through standardized platform like CUDA



TPUs: Tensor Processing Units

- Google's custom ASIC for ML inference and training
- TPUv3 pod provides 100 petaflops of ML compute
- TPUv3 First liquid-cooled accelerator in Google's datacenter
- TPUv4 paper at ISCA: https://arxiv.org/abs/2304.01433
- A variety of ML ASICs also available from other providers





FPGAs

- Map novel hardware designs onto fabric of logic blocks, memories, interconnect (and some hard IP)
  - Learn about this in EECS151
- Tradeoff between flexibility and efficiency
- Microsoft Brainwave: real-time AI inference using a soft NPU running on FPGA
- Also available as “bare” FPGAs on several providers
  - A huge opportunity for computer architecture researchers

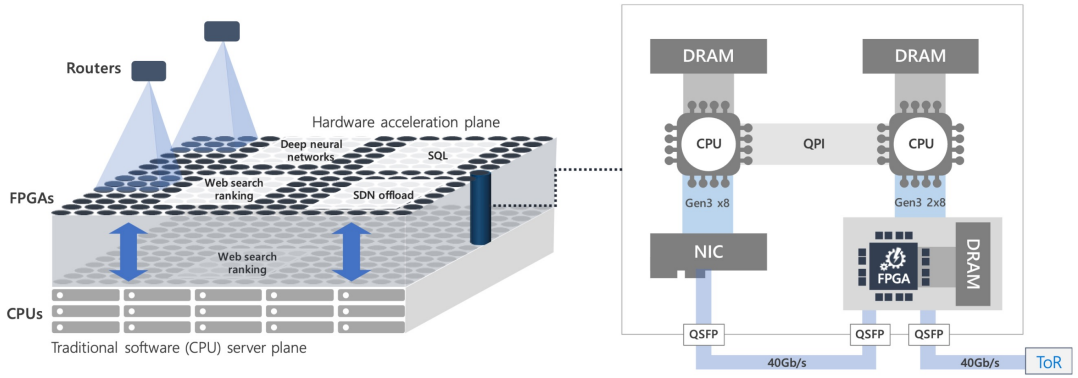
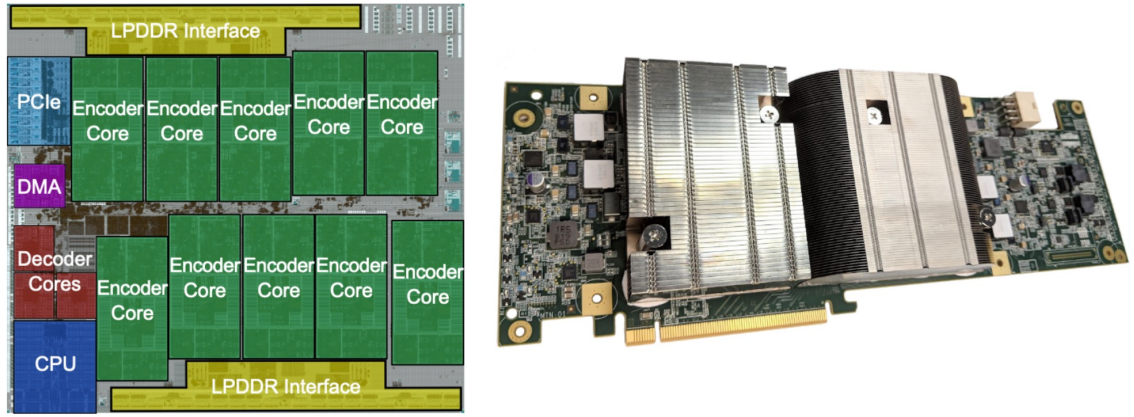


Image: “Serving DNNs in Real Time at Datacenter Scale with Project Brainwave”, IEEE Micro.

Image: https://aws.amazon.com/ec2/instance-types/f1/

VCU: Accelerating YouTube

- Video is > 60% of internet traffic
- Need to convert videos to/from multiple formats and resolutions
- Video Coding Unit (VCU) from Google/YouTube
- 7x perf/$ for H.264 and 33x for VP9 vs. software encoding



(a) Chip floorplan (b) Two chips on a PCBA

Figure 5: Pictures of the VCU

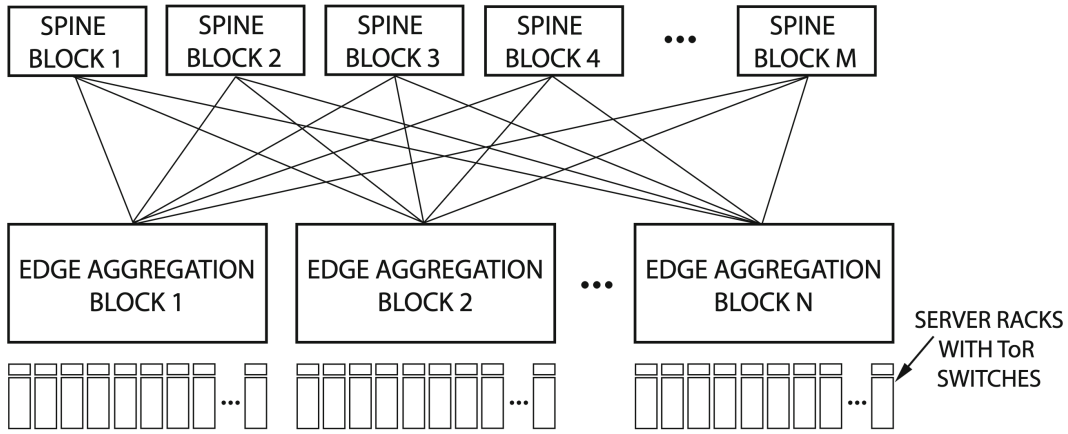
Image/Learn More: “Warehouse-Scale Video Acceleration: Co-design and Deployment in the Wild.” Parthasarathy Ranganathan, et. al. ASPLOS 2021.

WSC Networking

- Increasing aggregate storage or compute is easy: buy more servers and disks
- Networking has no straightforward horizontal scaling
- Scaling *leaf bandwidth* is easy; more servers = more network ports
- *bisection bandwidth* = bandwidth across the narrowest line that equally divides a cluster into two
- Scaling bisection bandwidth is hard
  - The “easy” way would be to build an arbitrarily large switch

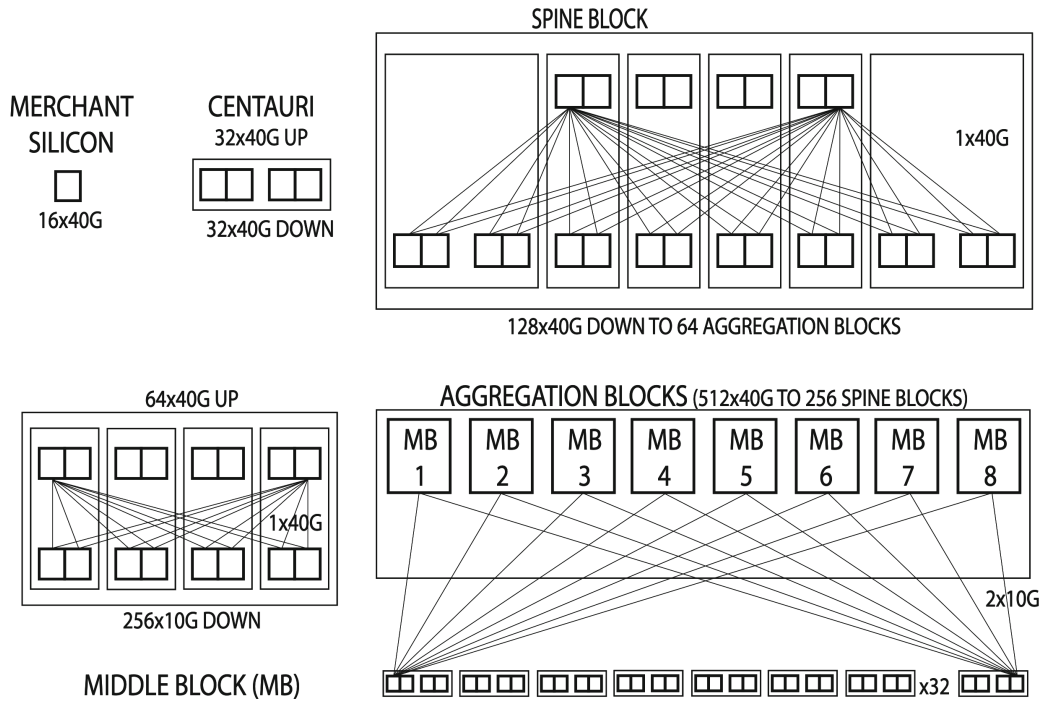
Building network switches

- Have N ports @ M Gbps, need to support full line-rate communication between all pairs of ports
  - E.g., N = 16 ports, M = 40 Gbps -> 640 Gbps bisection bandwidth
- Quickly become pin and power limited
- Can instead build larger switches by cascading switches in a more complicated topology, e.g. fat tree or Clos



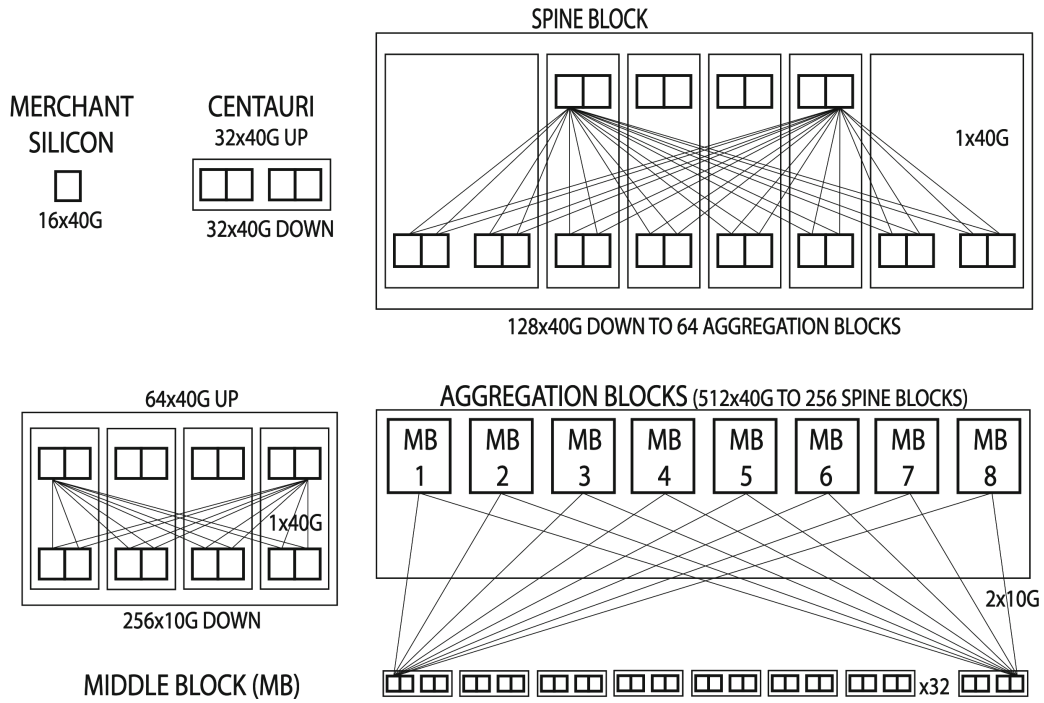
Modern WSC networks: Google Jupiter

- Multi-stage fabric built from low-radix switches built from merchant silicon
  - Each supporting 16 40Gbps ports
  - Configurable as 4x10G or 1x40G
- Server connects to ToR with 40 Gbps NIC
- Centauri = 2 x line card with 2 switch chips
- Allows, for example, 48x 10G to servers and 16x 10G to fabric: oversubscription of 3:1



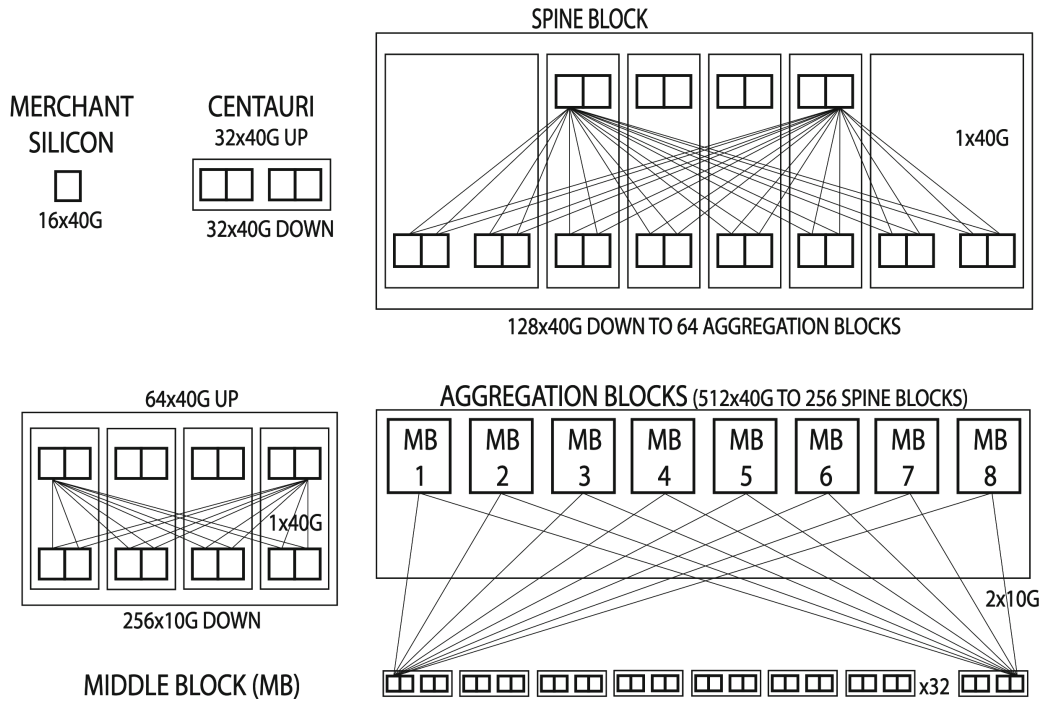
Modern WSC networks: Google Jupiter

- Centauri = 2 x line card with 2 switch chips
- ToRs connect to aggregation blocks, consisting of Middle Blocks
- Middle Block (MB) = 4 Centauris; 2-stage blocking network; 256x10G for ToR, 64x40G to spine
- Each ToR connects to 8 MBs



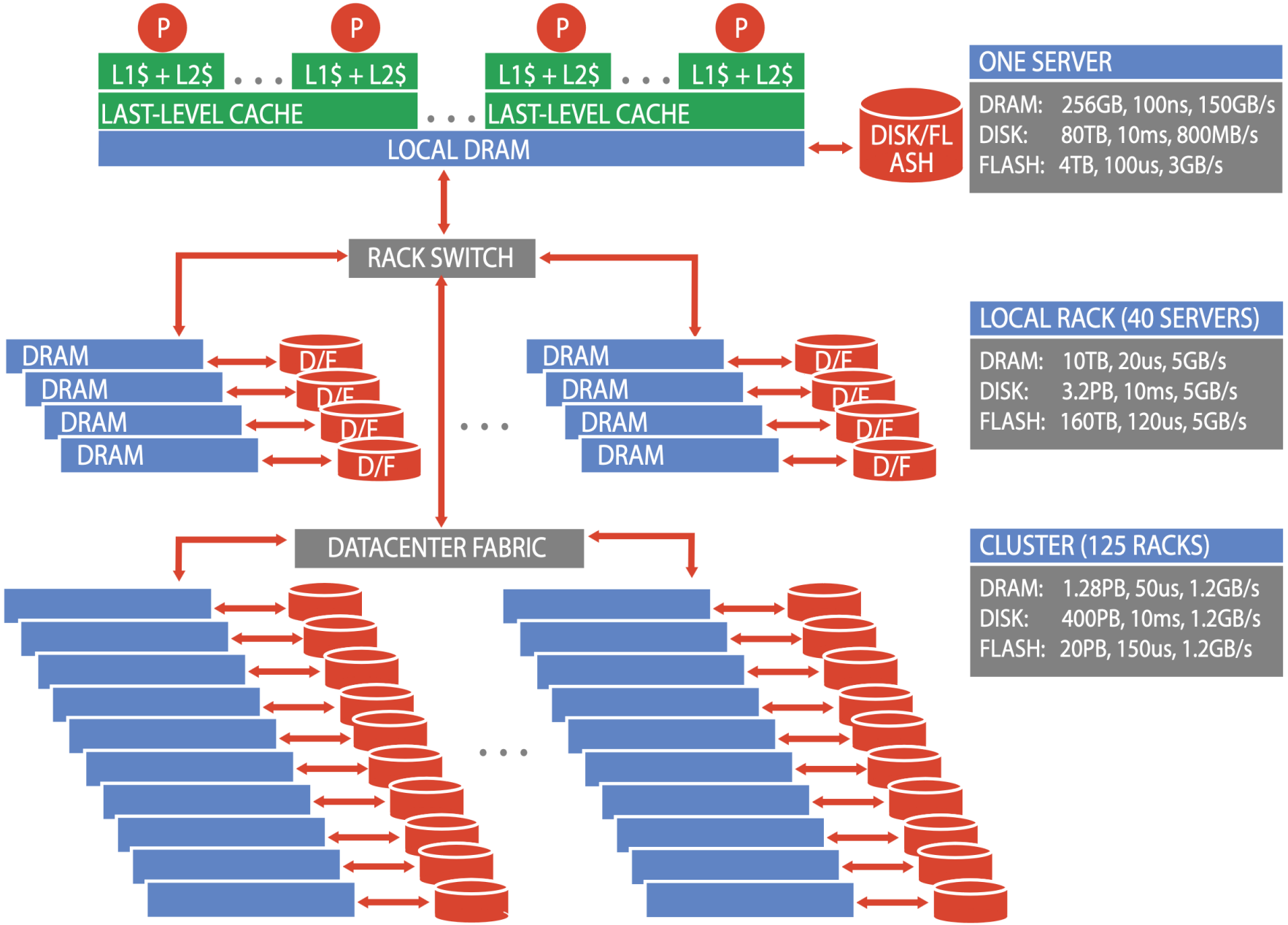
Modern WSC networks: Google Jupiter

- Centauri = 2 x line card with 2 switch chips
- ToRs connect to aggregation blocks, consisting of Middle Blocks
- Each aggregation block connects 512x40G or 256x40G to spine
- Spine = 6 Centauris w/ 128x40G to aggregation blocks
- At this maximum size, ***1.3 petabits per second*** bisection bandwidth



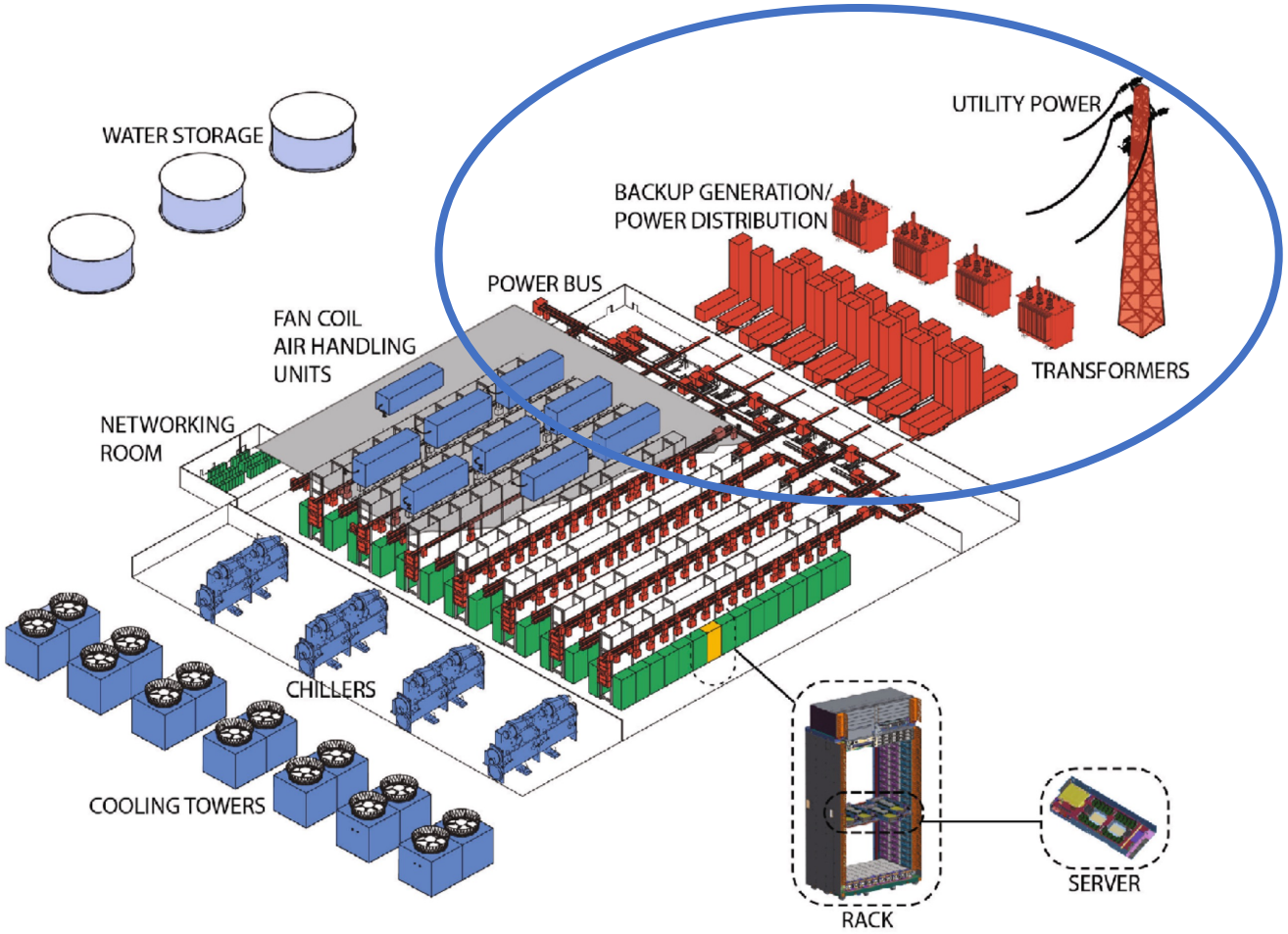
Learn more in CS168/CS268

WSC rules of thumb



What else is in a WSC?

WSC Power Delivery



Power enters at utility substation (110 kV+)

Transformed to medium voltage (~50 kV)

Primary distribution centers transform to < 1kV

Power enters building, fed to uninterruptible power systems (UPS)

UPS accepts power from utility supply plus second source from diesel generators

PDUs accept power from UPSes; route power to Racks/individual servers

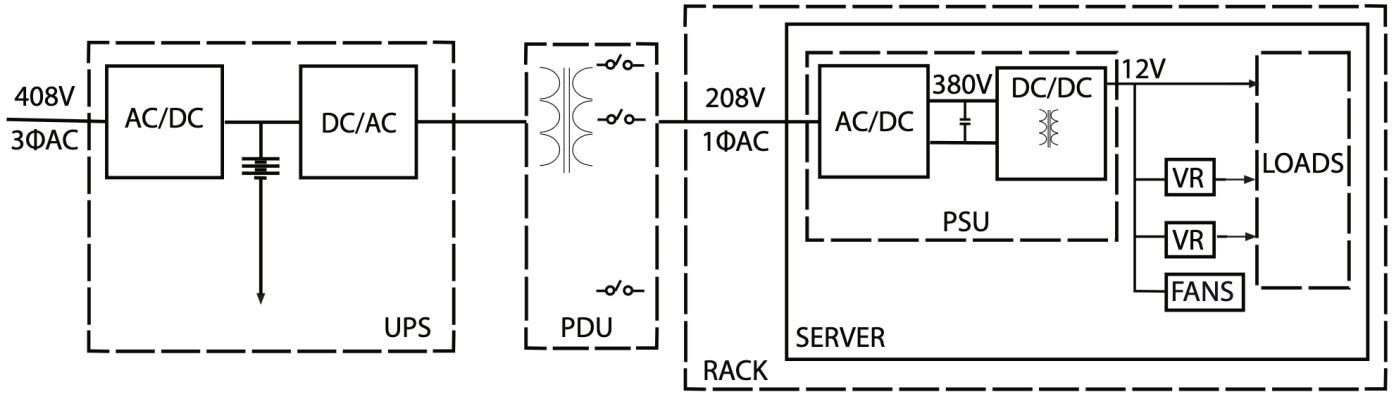
Uninterruptible Power Systems (UPS)

Three parts:

1. Transfer switch to choose the power input. When utility power fails, sense when the generator has started and can provide power (usually 10-15s)
2. Energy storage to bridge the gap between loss of utility power and generator ready
3. Power conditioning: remove voltage spikes/sags, distortion

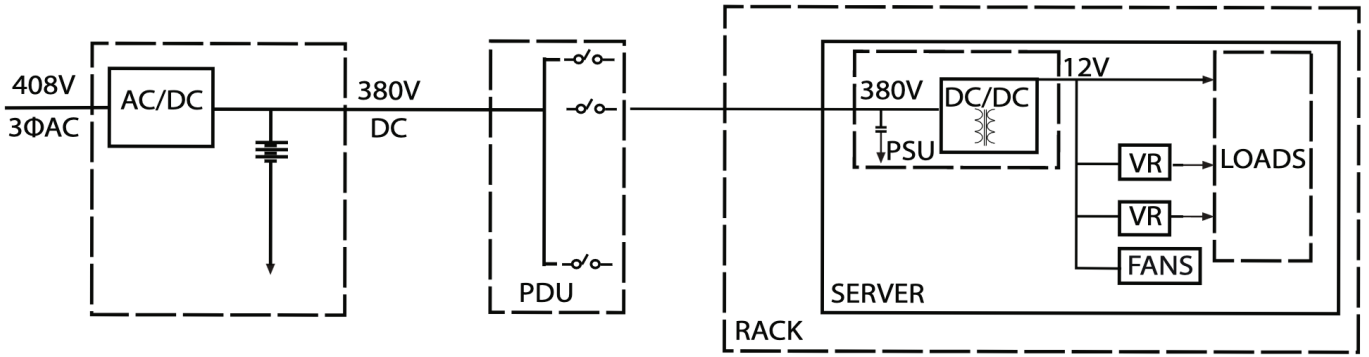
Power System Design Implications

CONVENTIONAL AC ARCHITECTURE



97% x 97% x 94% x 96% = 85%

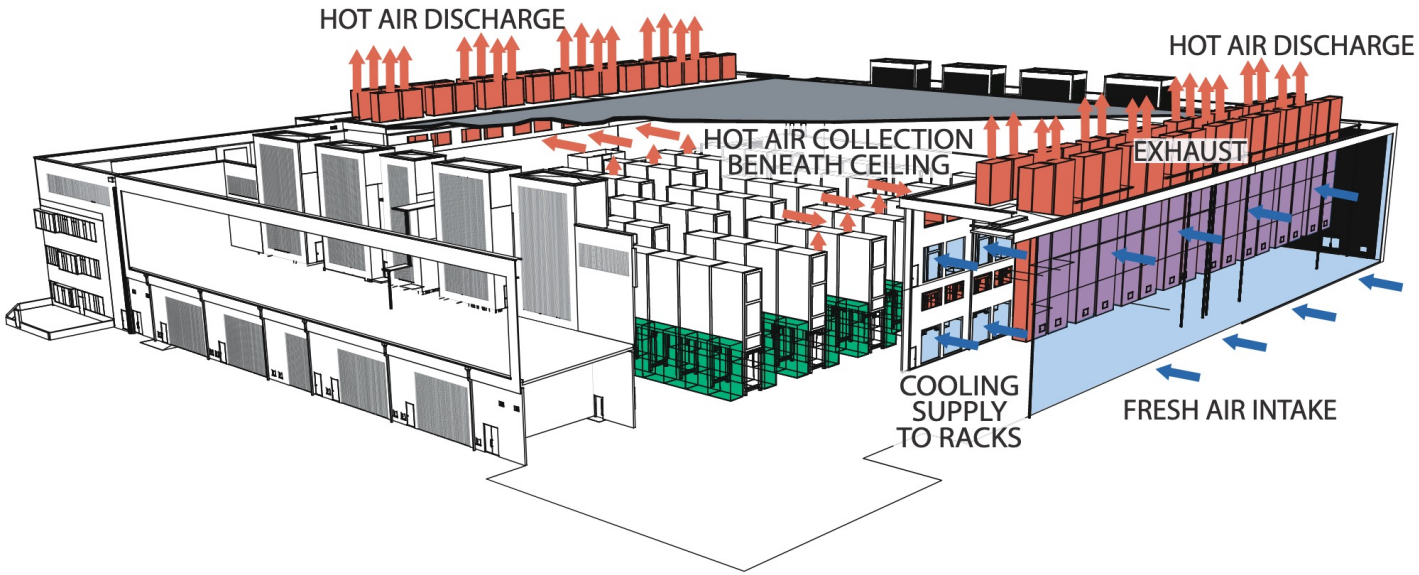
CONVENTIONAL DC ARCHITECTURE



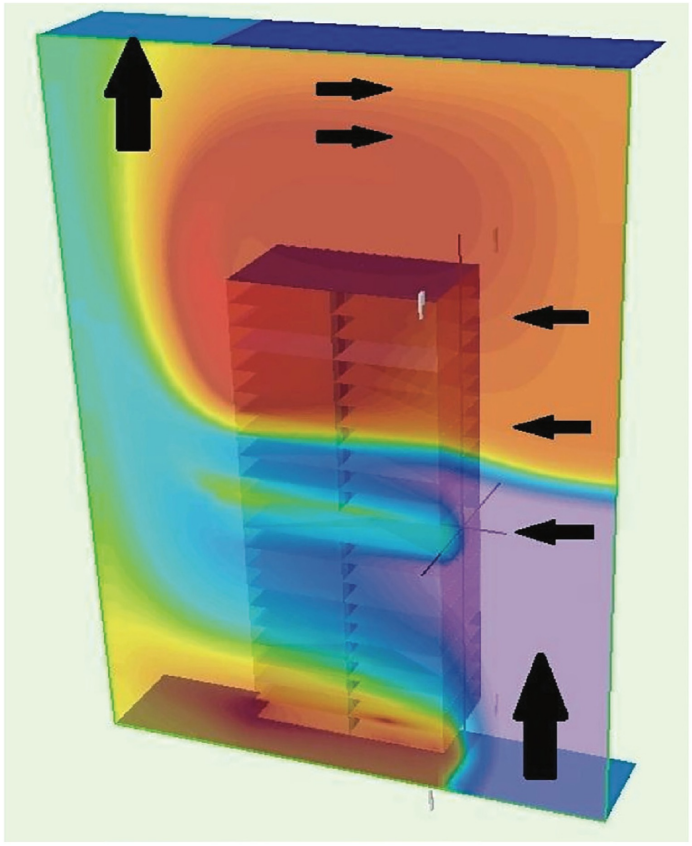
99% x 98% x 96% x 96% = 89%

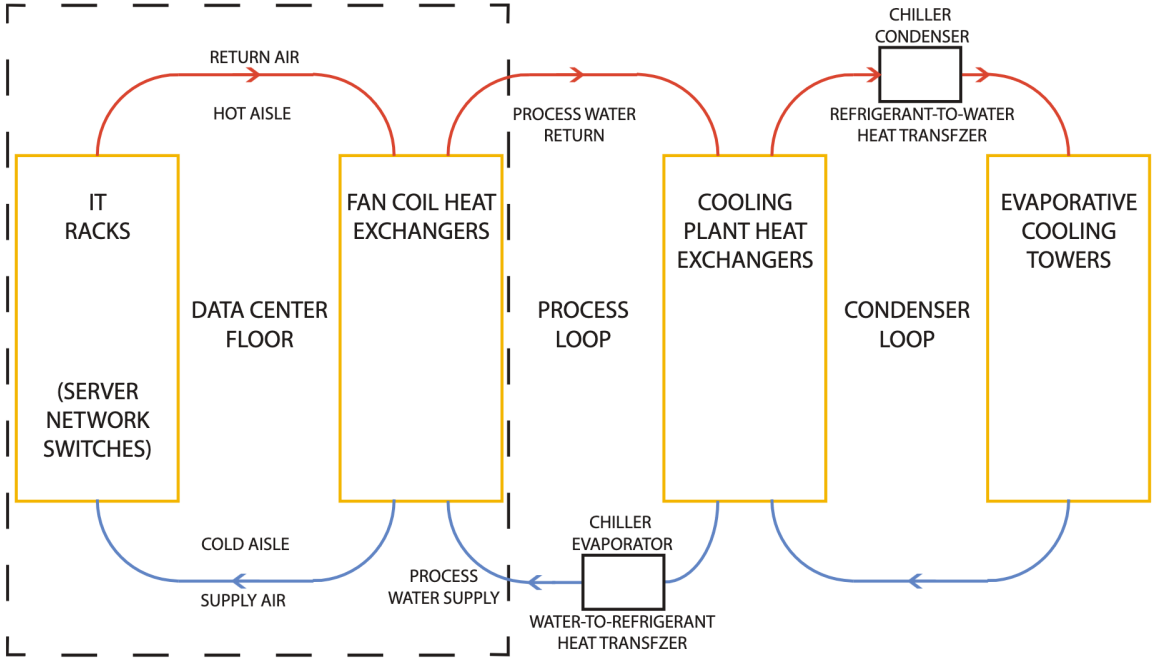
WSC Cooling

- Need to remove heat generated by equipment
- A hierarchy of loops that circulate a cold medium that warms up via heat exchange and is then cooled again
- Open loop = continuously replace medium with new, cool medium
- Closed loop = recirculate medium, transferring heat to some other medium before re-use
- Simplest: fresh-air cooling:

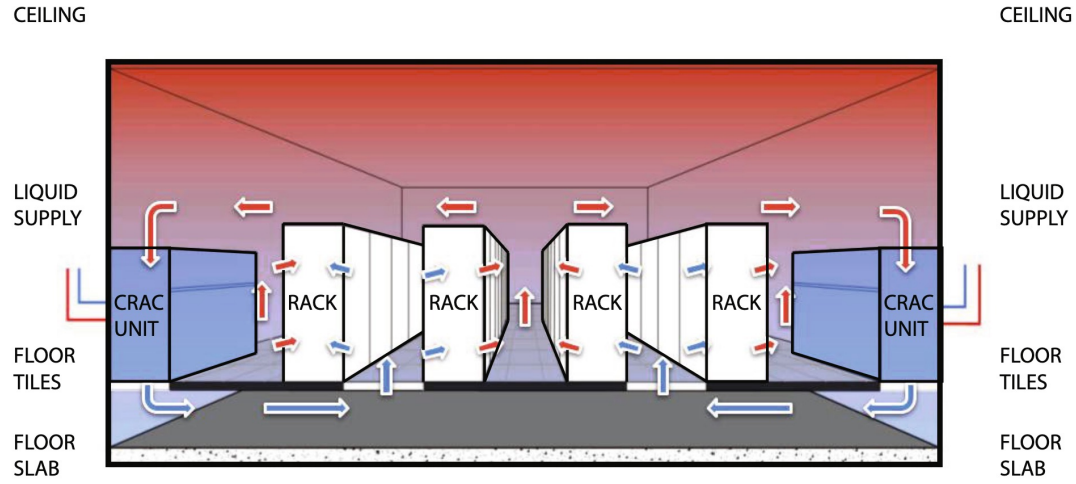


WSC Cooling




RETURN AIR
HOT AIRLE
IT RACKS
(SERVER NETWORK SWITCHES)
DATA CENTER FLOOR
FAN COIL HEAT EXCHANGERS
COLD AISLE
SUPPLY AIR
PROCESS WATER RETURN
PROCESS LOOP
COOLING PLANT HEAT EXCHANGERS
CHILLER CONDENSER
REFRIGERANT-TO-WATER HEAT TRANSFZER
CONDENSER LOOP
EVAPORATIVE COOLING TOWERS
CHILLER EVAPORATOR
PROCESS WATER SUPPLY
WATER-TO-REFRIGERANT HEAT TRANSFZER


VFD
MOTOR
COMPRESSOR
EVAPORATOR
CONDENSER


CEILING
CEILING
LIQUID SUPPLY
LIQUID SUPPLY
CRAC UNIT
RACK
RACK
RACK
RACK
CRAC UNIT
FLOOR TILES
FLOOR TILES
FLOOR SLAB
FLOOR SLAB

WSC Energy Efficiency

$$\text{Efficiency} = \frac{\text{Computation}}{\text{Total Energy}} = \underbrace{\left(\frac{1}{\text{PUE}}\right)}_{\text{(a)}} \times \underbrace{\left(\frac{1}{\text{SPUE}}\right)}_{\text{(b)}} \times \underbrace{\left(\frac{\text{Computation}}{\text{Total Energy to Electronic Components}}\right)}_{\text{(c)}}$$

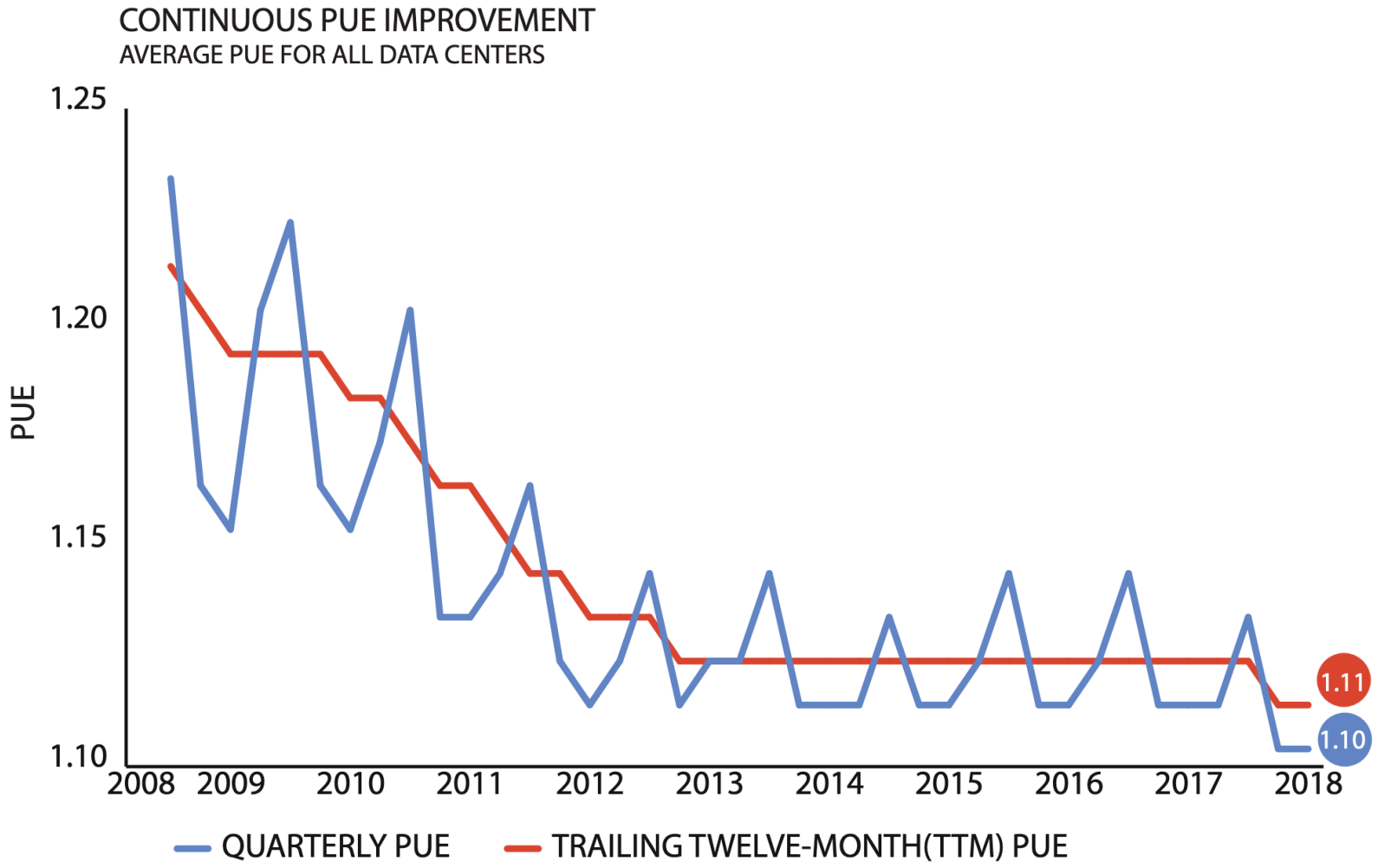
(a): Facility efficiency
(b): Server power conversion efficiency
(c): Server architectural efficiency

(a) Facility Efficiency: PUE

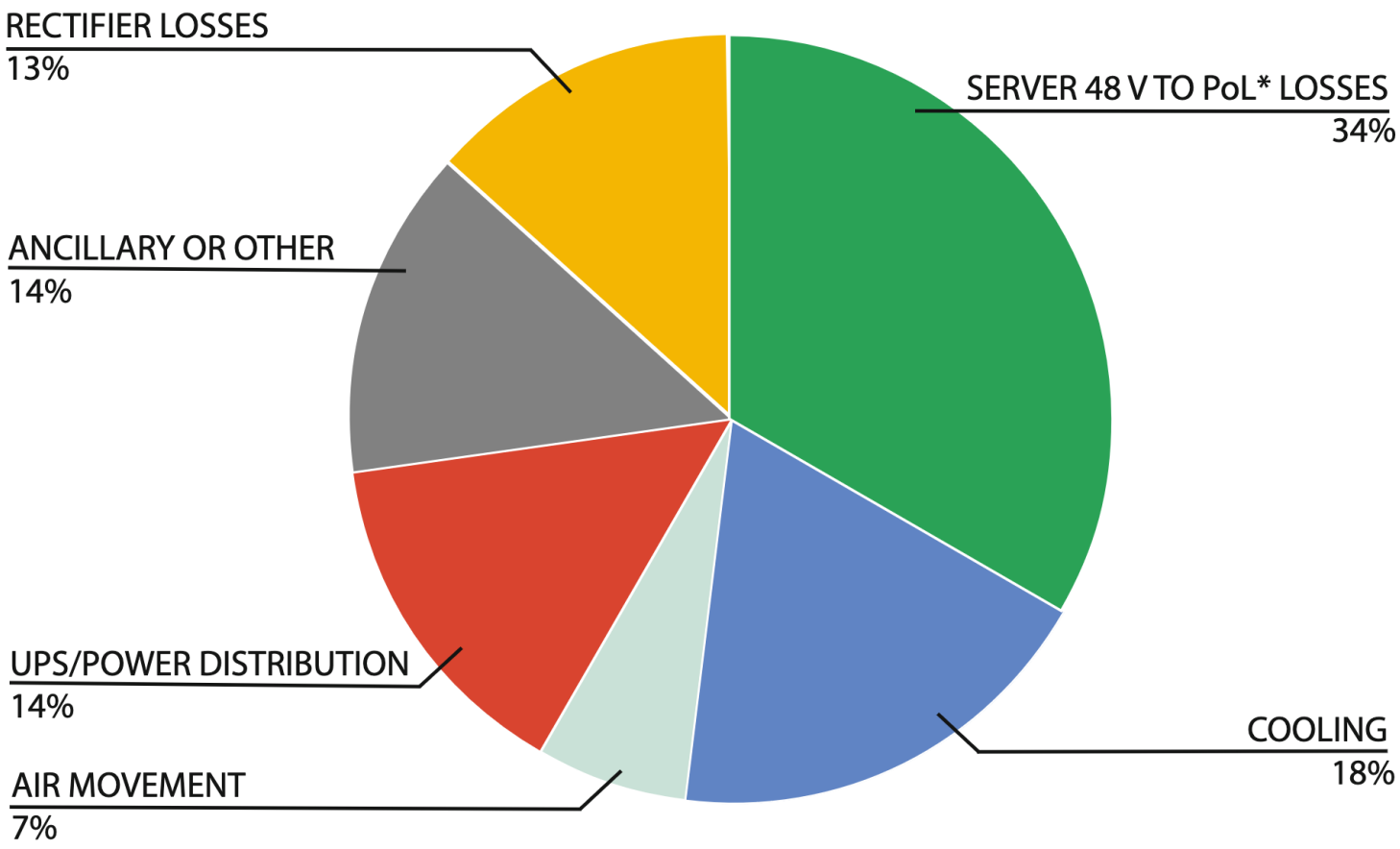
- Power Usage Effectiveness measures the quality of WSC building infrastructure
- PUE = (Facility power) / (IT Equipment power)
- Much work on improving PUE in the past ~15-20 years:
  - A 2006 study showed 85% of data centers with PUE greater than 3
  - A 2016 study showed PUEs of 1.13 for WSCs and 1.6-2.35 for traditional datacenters

(a) Facility Efficiency: PUE for Hyperscalers

- Hyperscaler (i.e., WSC operator) PUE has become very good
- E.g., Google's PUE evolution over time:



(a) PUE: Where are the losses?



(b) Server power conversion efficiency: sPUE

- Generally result from two steps:
  - Transform input voltage (110-220 VAC) to local DC (usually 12V)
  - Voltage regulator modules (VRM) transform down to actual voltages required by CPU, DRAM, etc.
- In 2009, ratios around 1.6-1.8, with 20% loss at server power supply and 25% loss in VRM
- In modern systems, 6% loss and 4% loss respectively, for sPUE = 1.1

(c) Server architectural efficiency

- Want *power proportionality*: performance to power ratio constant, and power = 0 at performance = 0
- Don't get this in practice:

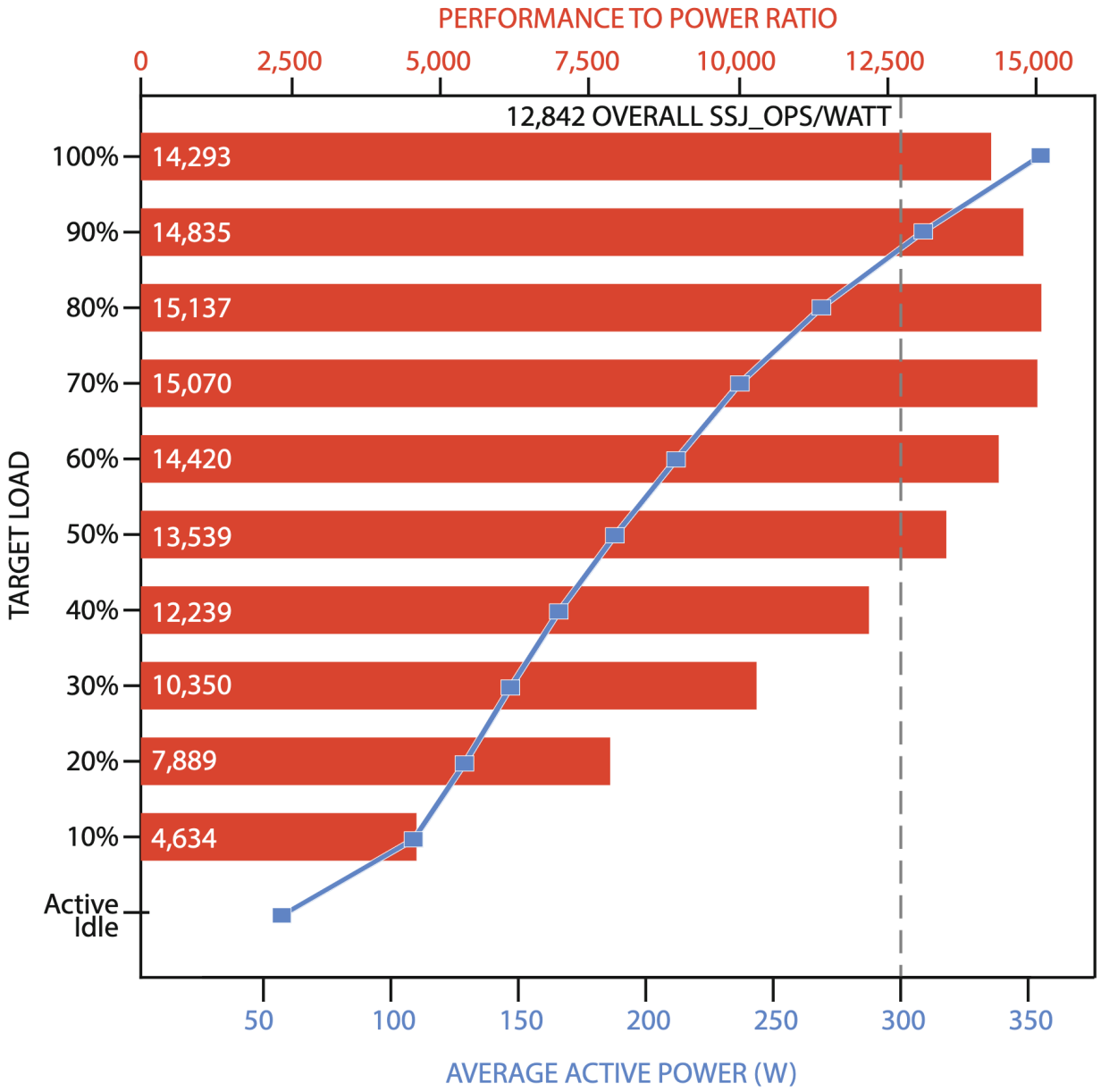


Figure 5.4: Example benchmark result for SPECpower_ssj2008; bars indicate energy efficiency and the line indicates power consumption. Both are plotted for a range of utilization levels, with the average energy efficiency metric corresponding to the vertical dark line. The system has two 2.1 GHz 28-core Intel Xeon processors, 192 GB of DRAM, and one M.2 SATA SSD.

(c) Server architectural efficiency

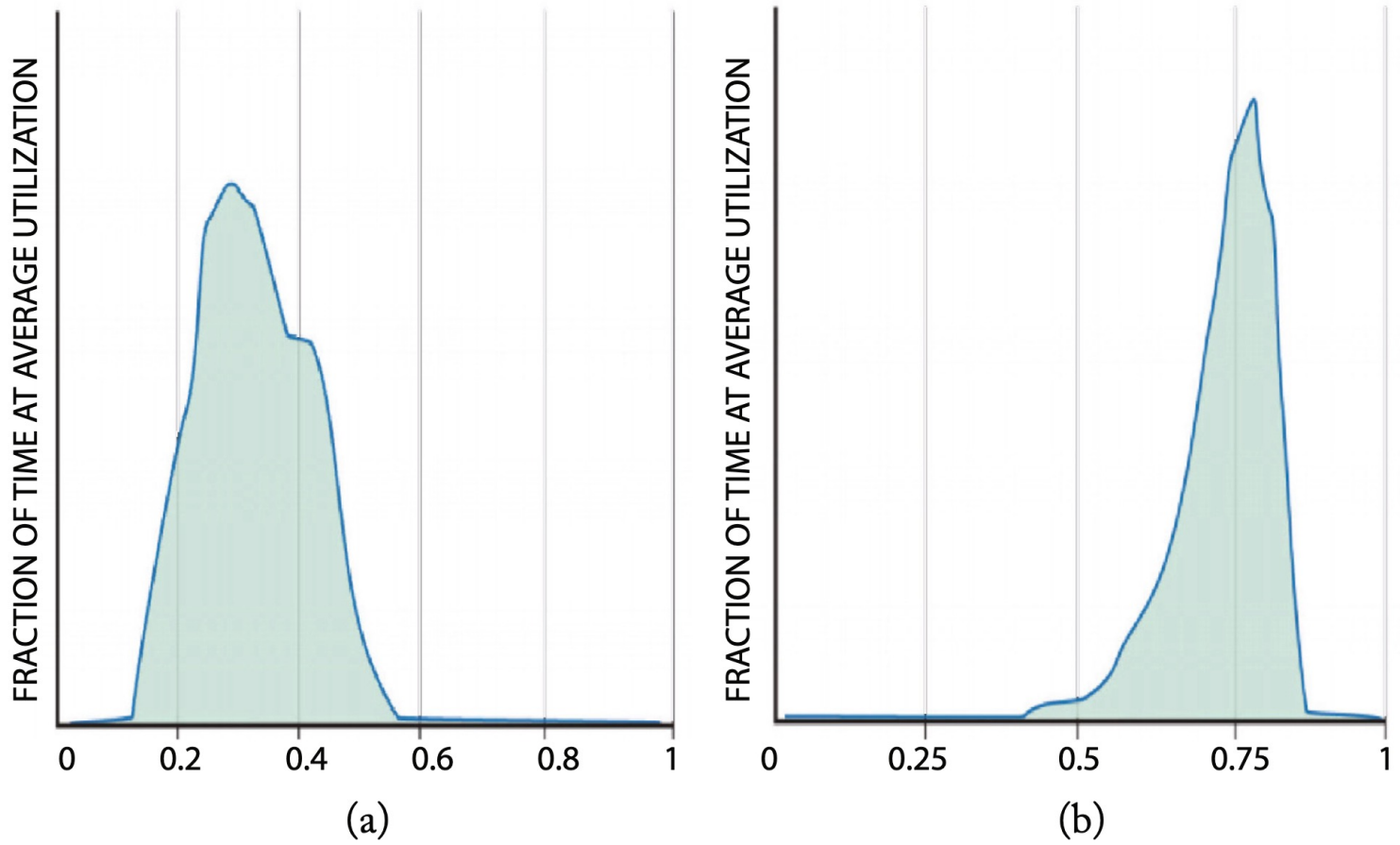


Figure 5.5: Average activity distribution of a sample of 2 Google clusters, each containing over 20,000 servers, over a period of 3 months.

Review: WSC Energy Efficiency

$$\text{Efficiency} = \frac{\text{Computation}}{\text{Total Energy}} = \underbrace{\left(\frac{1}{\text{PUE}}\right)}_{\text{(a)}} \times \underbrace{\left(\frac{1}{\text{SPUE}}\right)}_{\text{(b)}} \times \underbrace{\left(\frac{\text{Computation}}{\text{Total Energy to Electronic Components}}\right)}_{\text{(c)}}$$

(a): Facility efficiency
(b): Server power conversion efficiency
(c): Server architectural efficiency

Understanding WSC costs: TCO

- TCO = Total Cost of Ownership
- Capex = Capital expenses: upfront investments that depreciate over a timeframe
  - E.g., cost of constructing datacenter or buying a server
- Opex = Operational expenses: monthly costs like electricity, repairs, maintenance, salaries, etc.

Handling Faults in WSCs

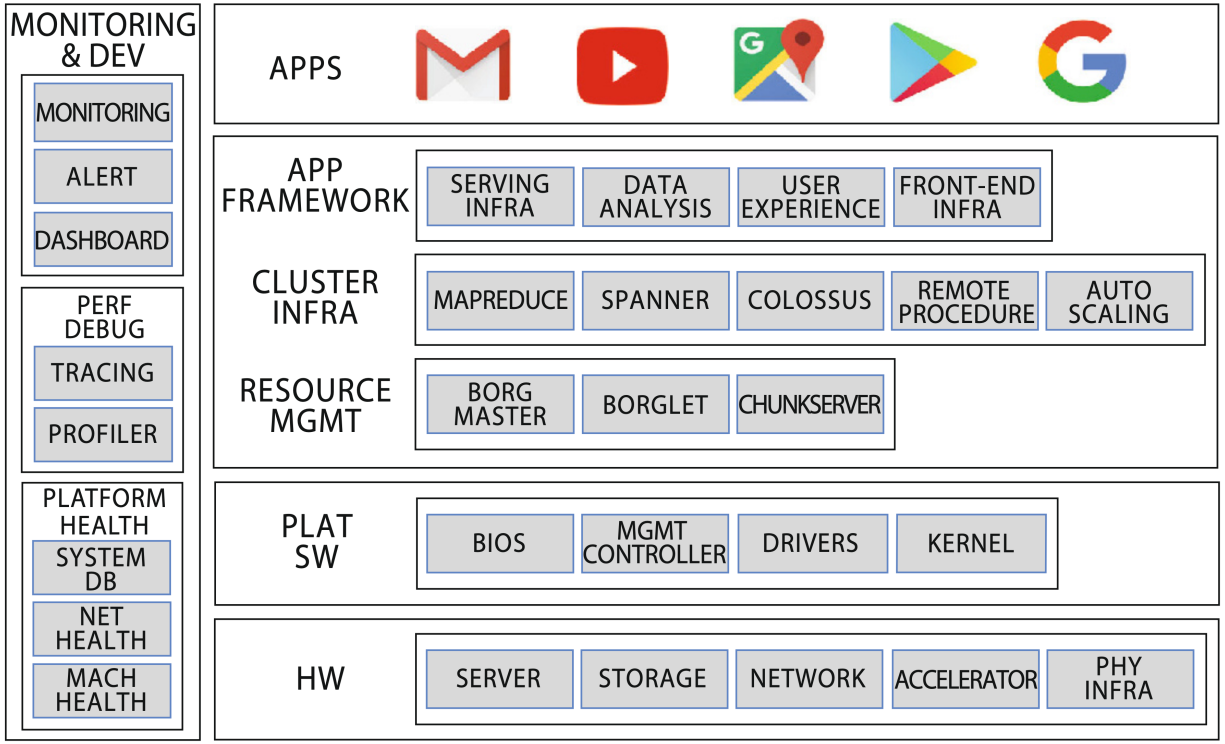
- A lot of work on reducing failures in enterprise-class servers:
  - Error correction coding (ECC), Redundant Arrays of Inexpensive Disks (RAID), redundancy in power supplies, fans, etc.
- At scale, hardware reliability simply isn't good enough:
  - Suppose unrealistically good servers with MTBF (mean time between failures) of 30 years
  - In a cluster of 10,000 servers, we will still see one failure per-day, on average
  - An application that relies on availability of the entire cluster will, on average, fail every day
- Also have large, complex services with many layers, each with their own bugs
- Must design with fault-tolerance in mind

Summary of fault tolerance mechanisms

- Power delivery and cooling: Distribute power hierarchically, scheduler can spread jobs across failure domains. Also spread storage hierarchy across failure domains. UPSes, generators, etc.
- Networking: Fabric redundancy, robust protocols, reliable out-of-band control plane. High bisection bandwidth supports other failure mitigation techniques (e.g., spreading jobs)
  - Learn more in CS168/CS268
- Scheduling: Applications expected to work well with replication, using distributed filesystems, etc. Automatic re-scheduling under failure.
  - Learn more in CS162/CS262
- Storage: fast recovery and replication/encoding

WSC Software

- Platform-level software: firmware, kernel, OS (for individual servers)
- Cluster-level software:
  - Distributed systems that provide services and manage resources
  - The “datacenter’s operating system”
  - Distributed file systems, schedulers, remote procedure call (RPC) frameworks, programming models like MapReduce
- Application-level software: Services you use every day
- Monitoring and dev software: e.g., Google-Wide Profiling (GWP)

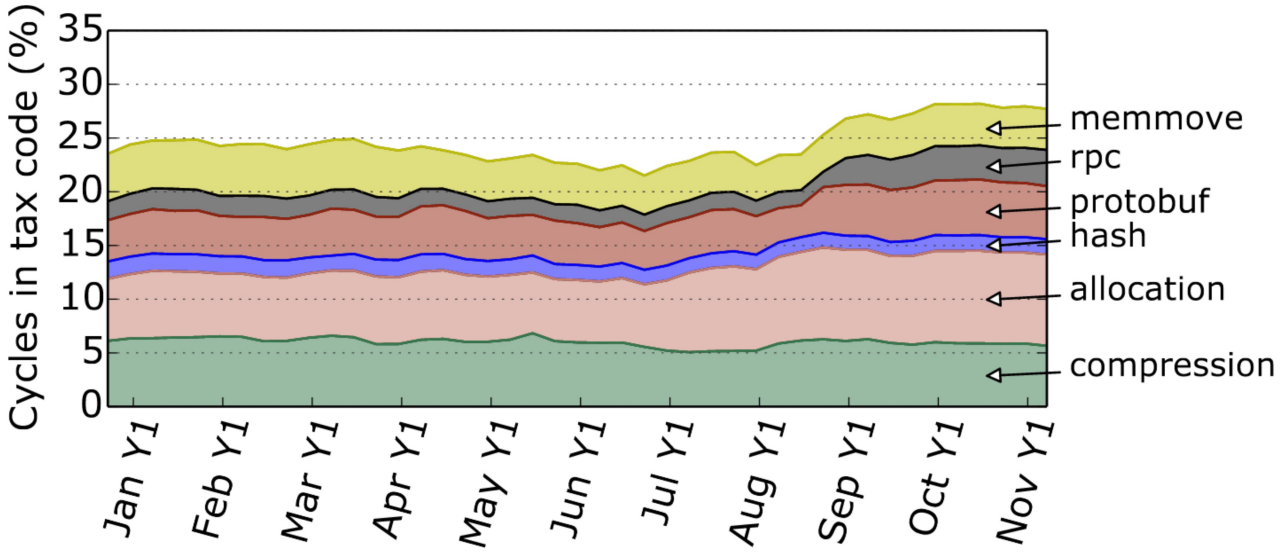


Learn more in CS162/CS262

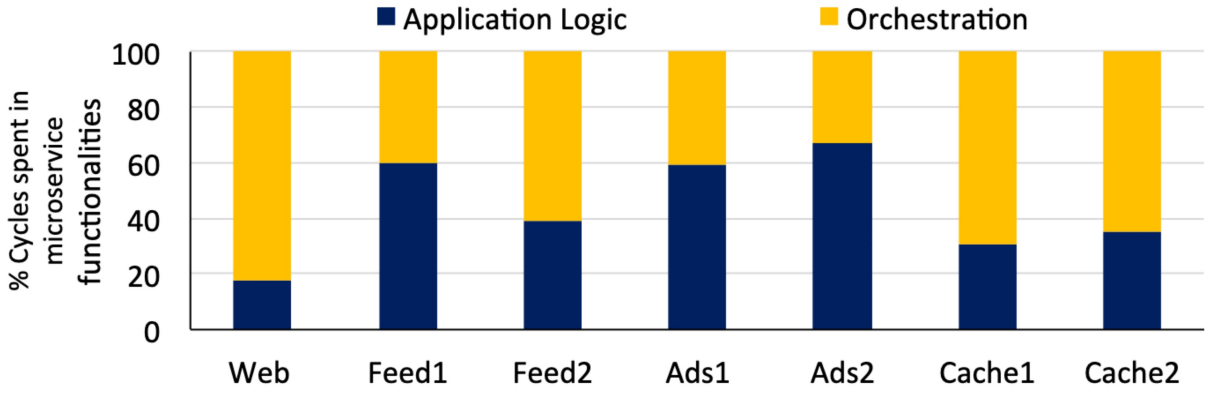
WSC research at Berkeley

(what follows, except summary slide, is optional material)

Eliminating *Datacenter Taxes*



Datacenter Tax Overheads, Google, 2015



Orchestration Overheads, Meta, 2020

EECS 151 / CS152 Preview: How is HW really built?

- RTL is the “code” for implementing a digital hardware design
  - Languages like Verilog, VHDL, Chisel, etc.
  - i.e., Logisim, but you write code instead of using drag-and-drop blocks
- RTL can be simulated in SW
  - At < 1 to 10 KHz
- RTL can run on an FPGA
  - At 1s to 100s of MHz
- RTL can be “taped-out” to produce a chip
  - i.e., the real systems you use, at 1s of GHz

Agile Hardware Methodology for Hyperscale Systems Development

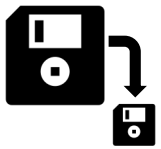
- How can we model our systems of interest?
- How can we improve our open-source application core to serve as a realistic host system?
- How do we build useful hardware accelerators for hyperscale?
  - Hyperscaler fleet-wide profiling
  - Build parameterized hardware generators for DC Tax accelerators that integrate into a complete SoC
    - Incorporate design feedback from ASIC flow
  - Build hyperscaler-representative benchmarks
  - Combine all of the above, make it all in open-source

S. Karandikar, et. al.
ISCA ‘18
IEEE Micro Top Picks 2018
Used by 20+ institutions

S. Karandikar, et. al.
ASPLOS ‘20

A Hardware Accelerator for Protocol Buffers

S. Karandikar, et. al. w/Google
MICRO ’21
Honorable Mention, IEEE Micro Top Picks 2021
MICRO ‘21 Distinguished Artifact Award Winner



CDPU: Compression / Decompression Processing Unit

S. Karandikar, et. al. w/Google
ISCA ‘23

Many more in progress, email me!
sagark@eecs.berkeley.edu

How can we do WSC research without building an entire WSC?

Build a fast simulator that is true to the WSC hardware: *FireSim*

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

The architect/chip-developer's design flow

1. High-level Simulation
2. Write RTL + Software, plug into your favorite ecosystem (e.g. Chipyard)
3. Co-design in software RTL sim (e.g. Verilator, VCS, etc.)
   - Run microbenchmarks
4. Co-design in FPGA-accelerated simulation
   - Boot an OS and run the complete software stack,
     obtain realistic performance measurements
5. Tapeout → Chip
   - Boot OS and run applications, but no more opportunity for co-design

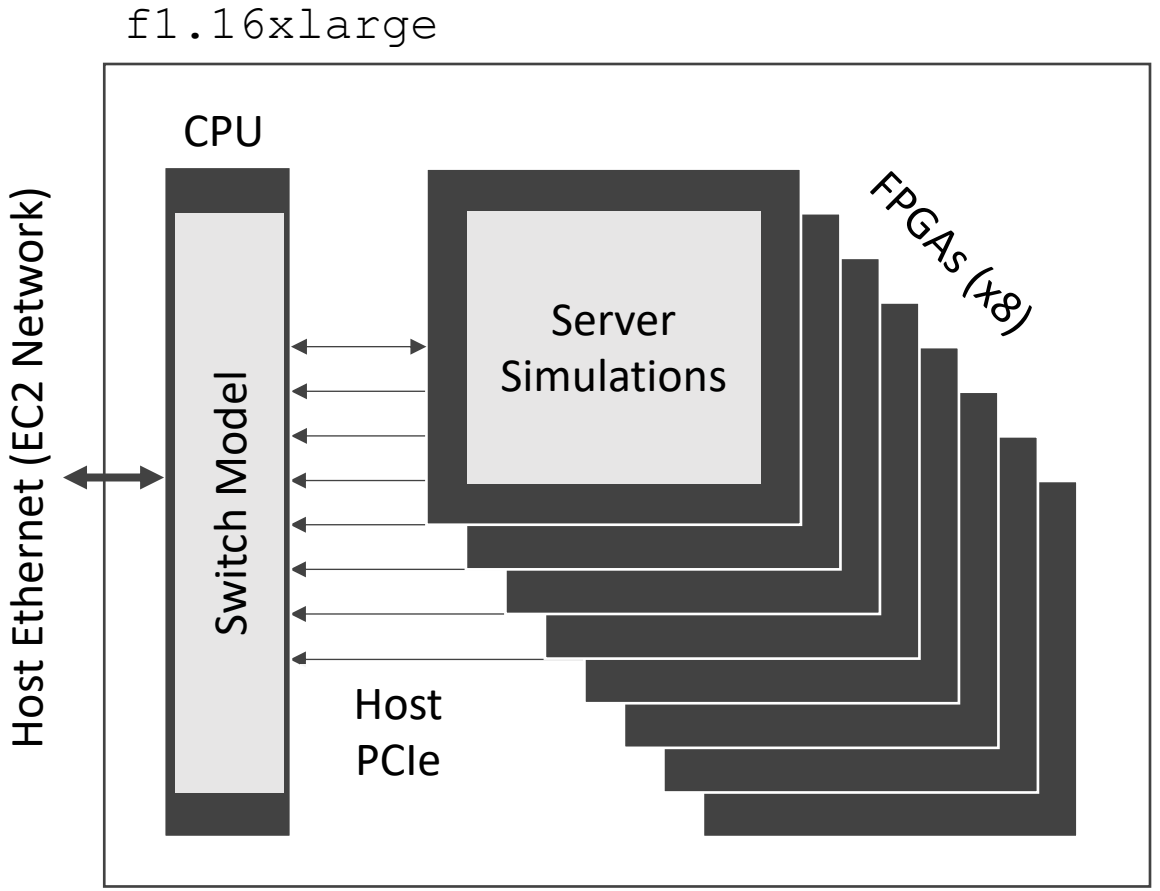
The architect/chip-developer's design flow

1. High-level Simulation
2. Write RTL + Software, plug into your favorite ecosystem (e.g. Chipyard)
3. Co-design in software RTL sim (e.g. Verilator, VCS, etc.)
   - Run microbenchmarks
4. **Co-design in FPGA-accelerated simulation**
   - **Boot an OS and run the complete software stack, obtain realistic performance measurements**

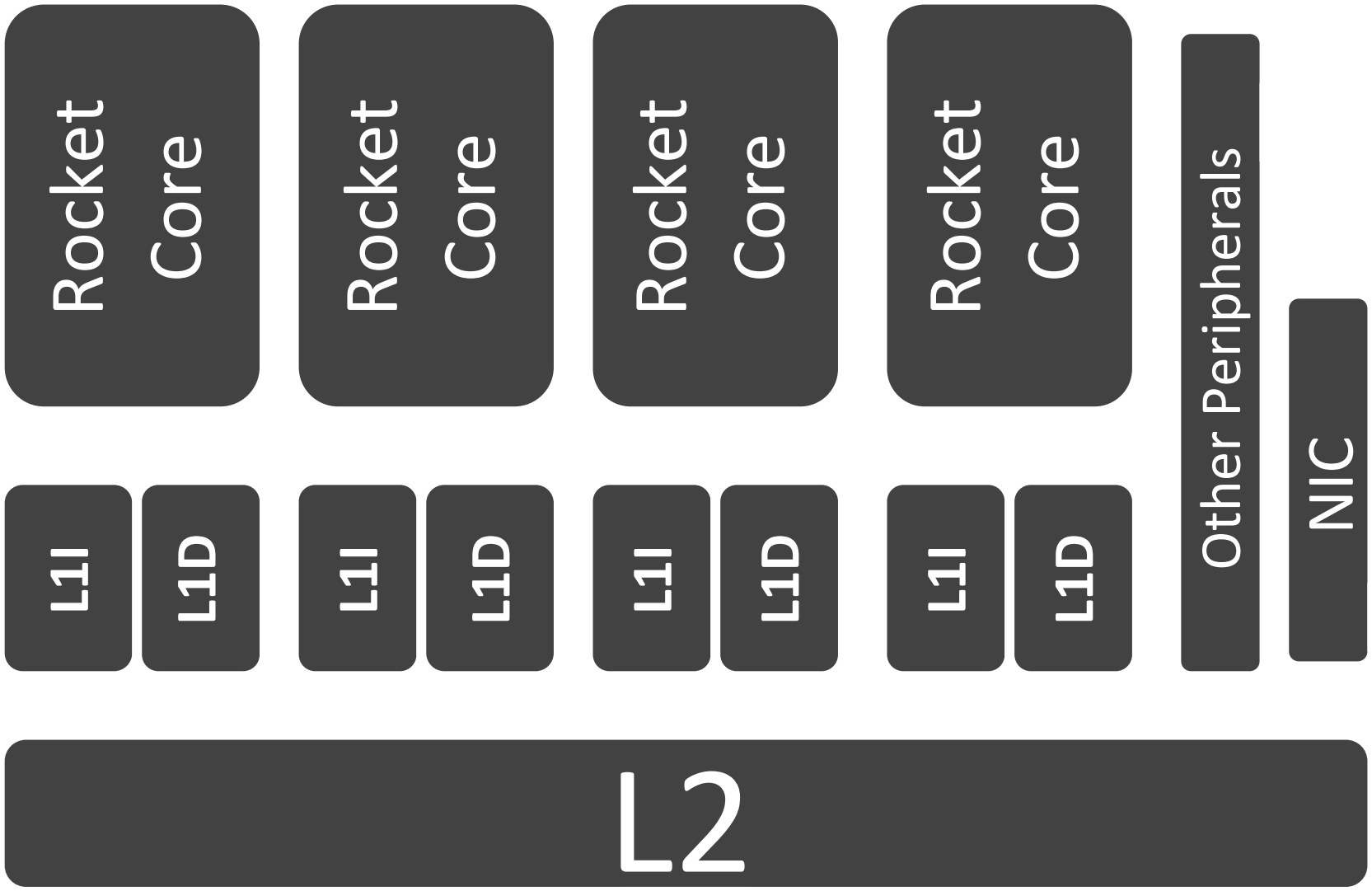
5. Tapeout → Chip
   - Boot OS and run applications, but no more opportunity for co-design

Simulating a datacenter using a datacenter

- DC simulation requires:
  - 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



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

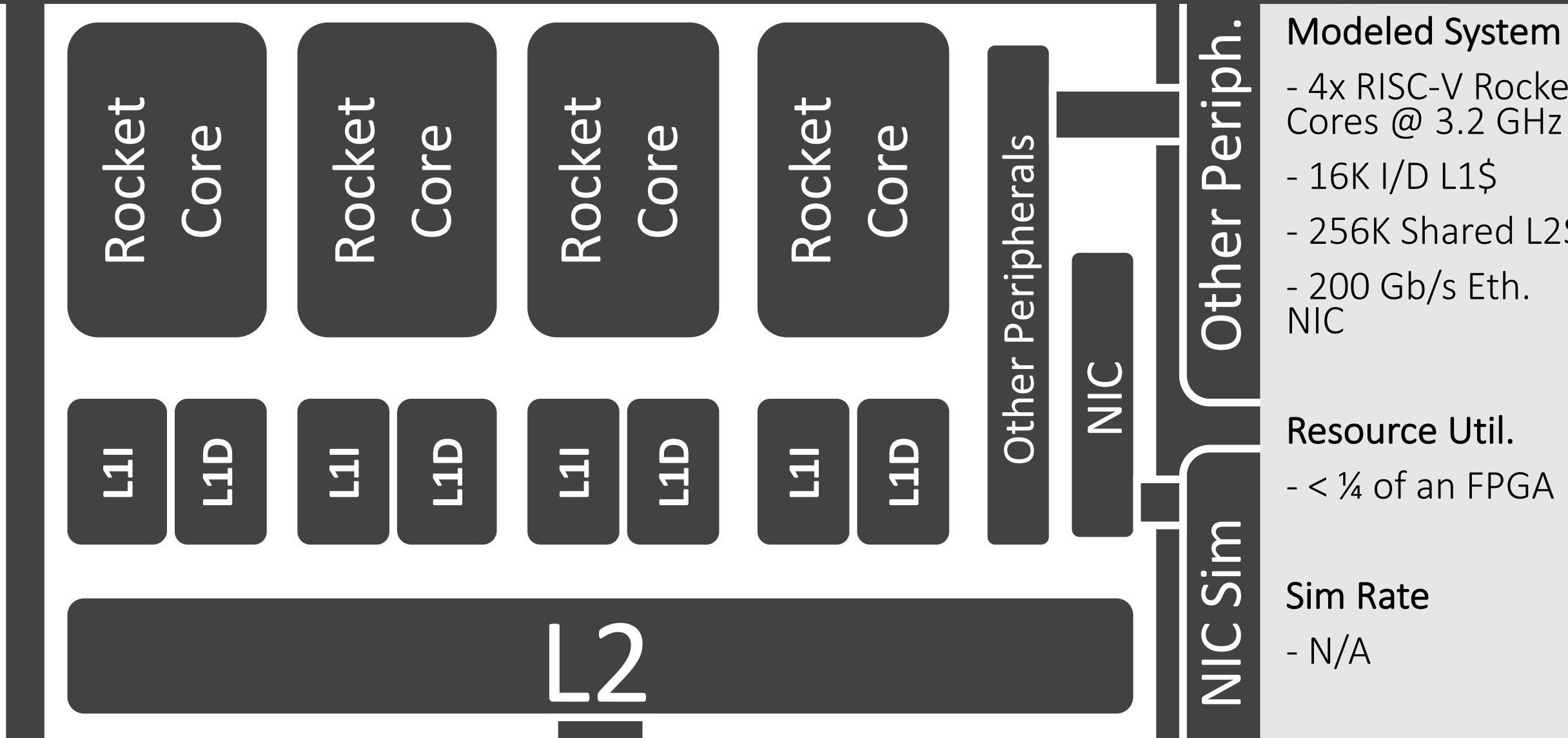
Resource Util.

- < 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

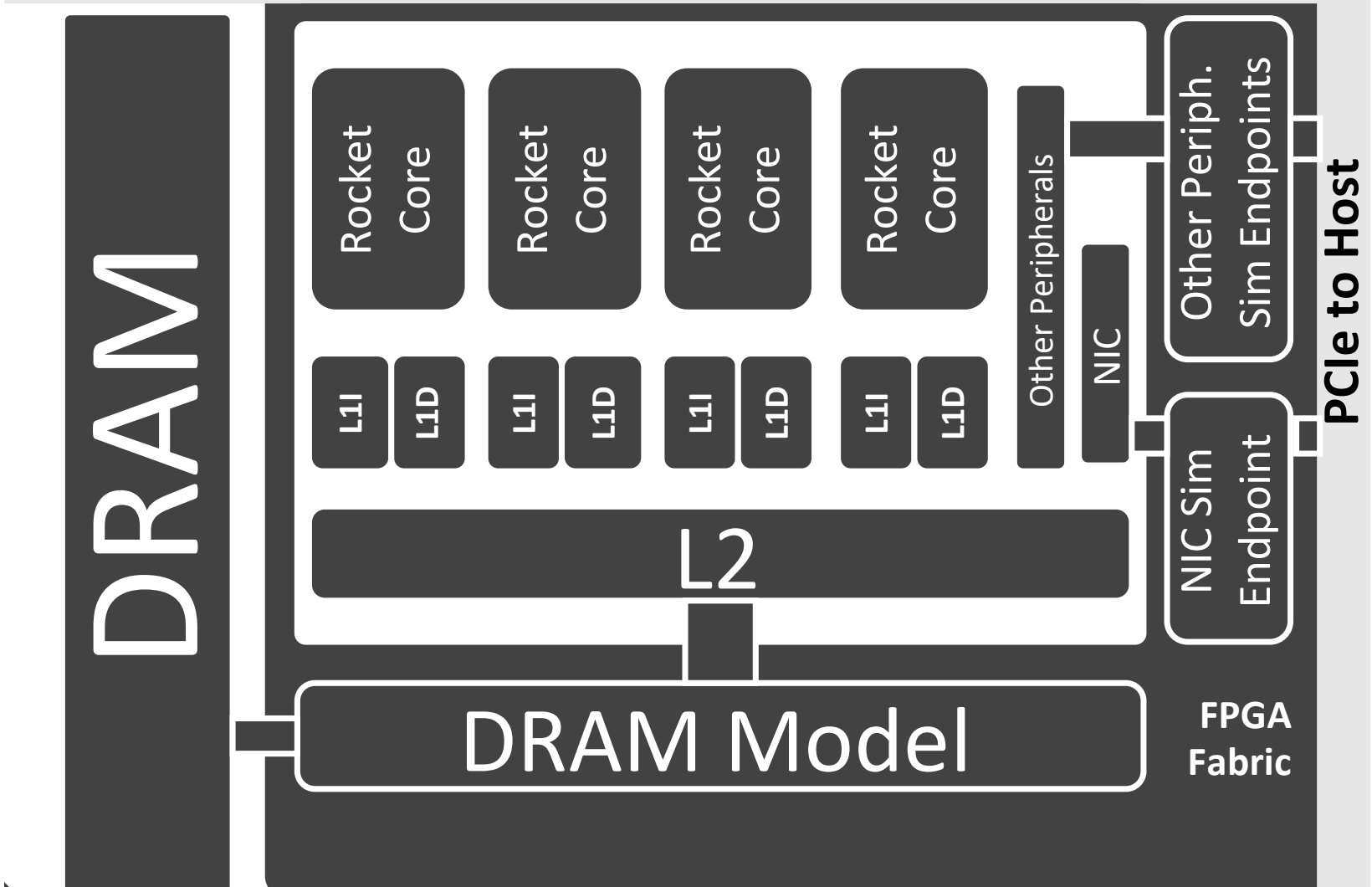
Resource Util.

- < 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

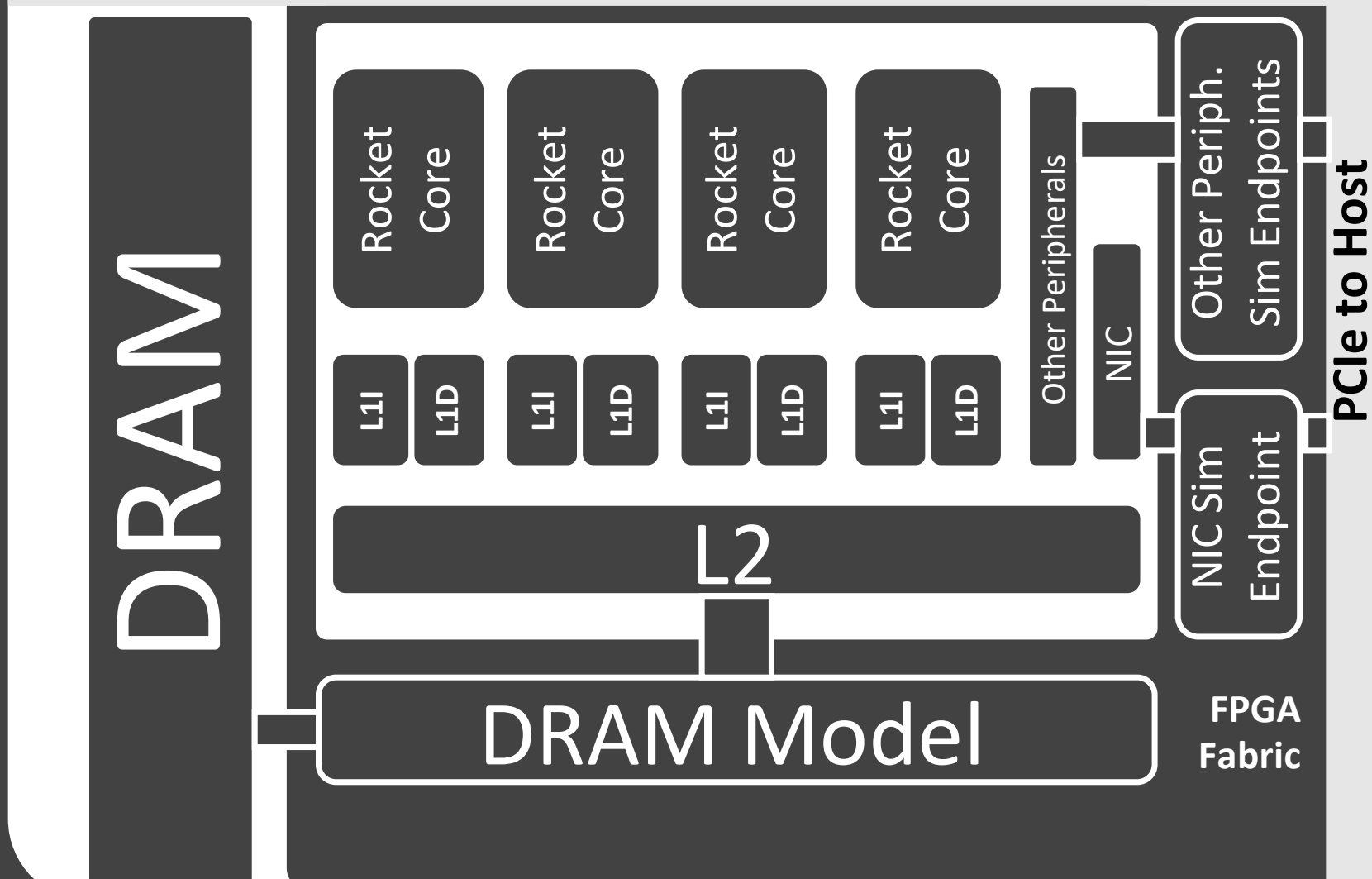
Resource Util.

- < ¼ of an FPGA
- ¼ Mem Chans

Sim Rate

- ~150 MHz
- ~40 MHz (netw)

Step 2: FPGA Simulation of one server blade



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

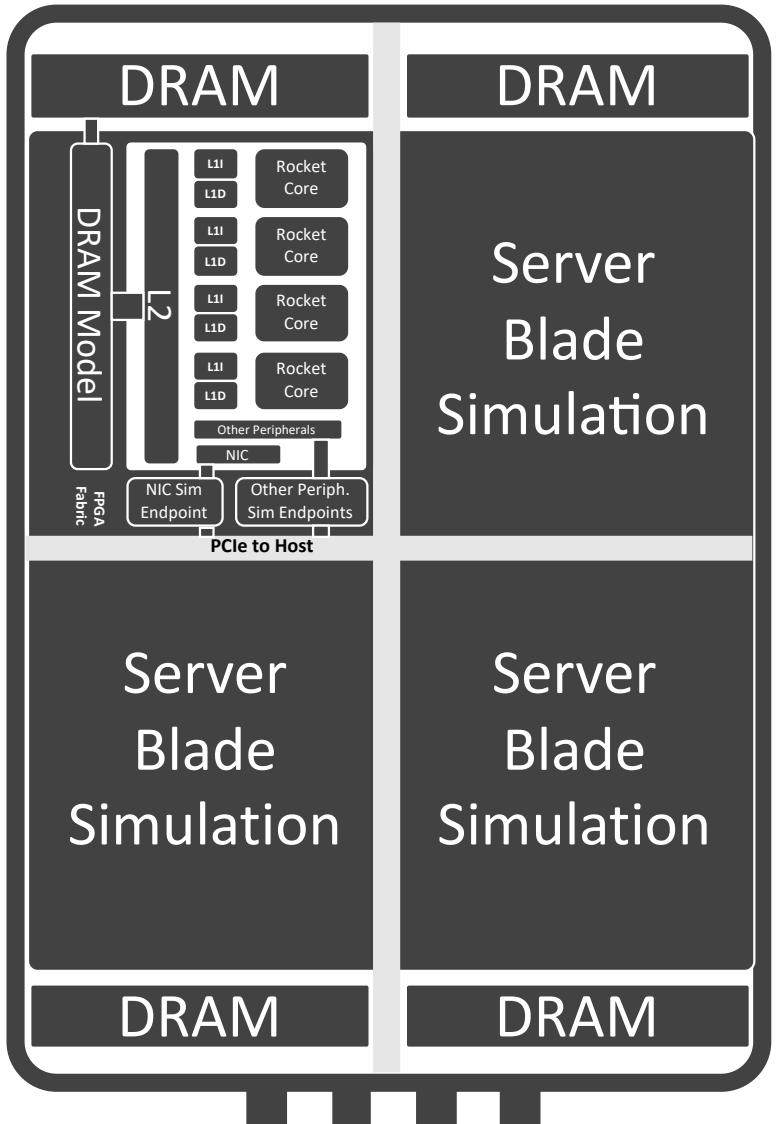
Sim Rate

- ~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)



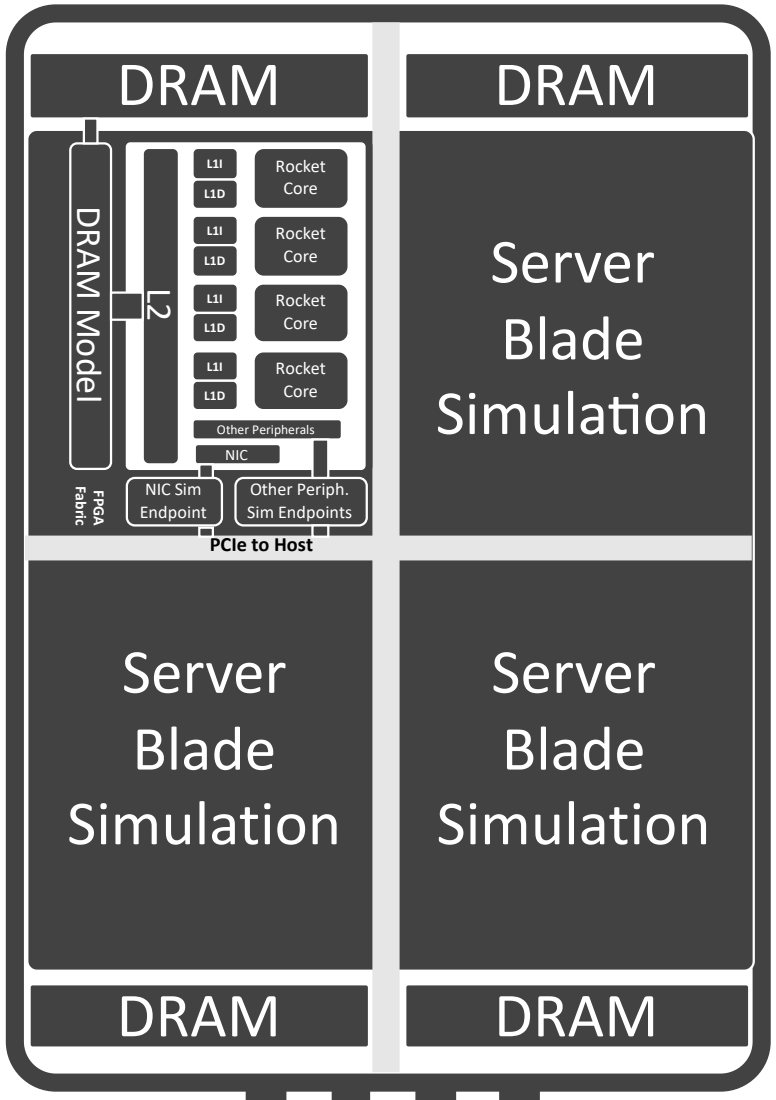
Modeled System
- 4 Server Blades
- 16 Cores
- 64 GB DDR3

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

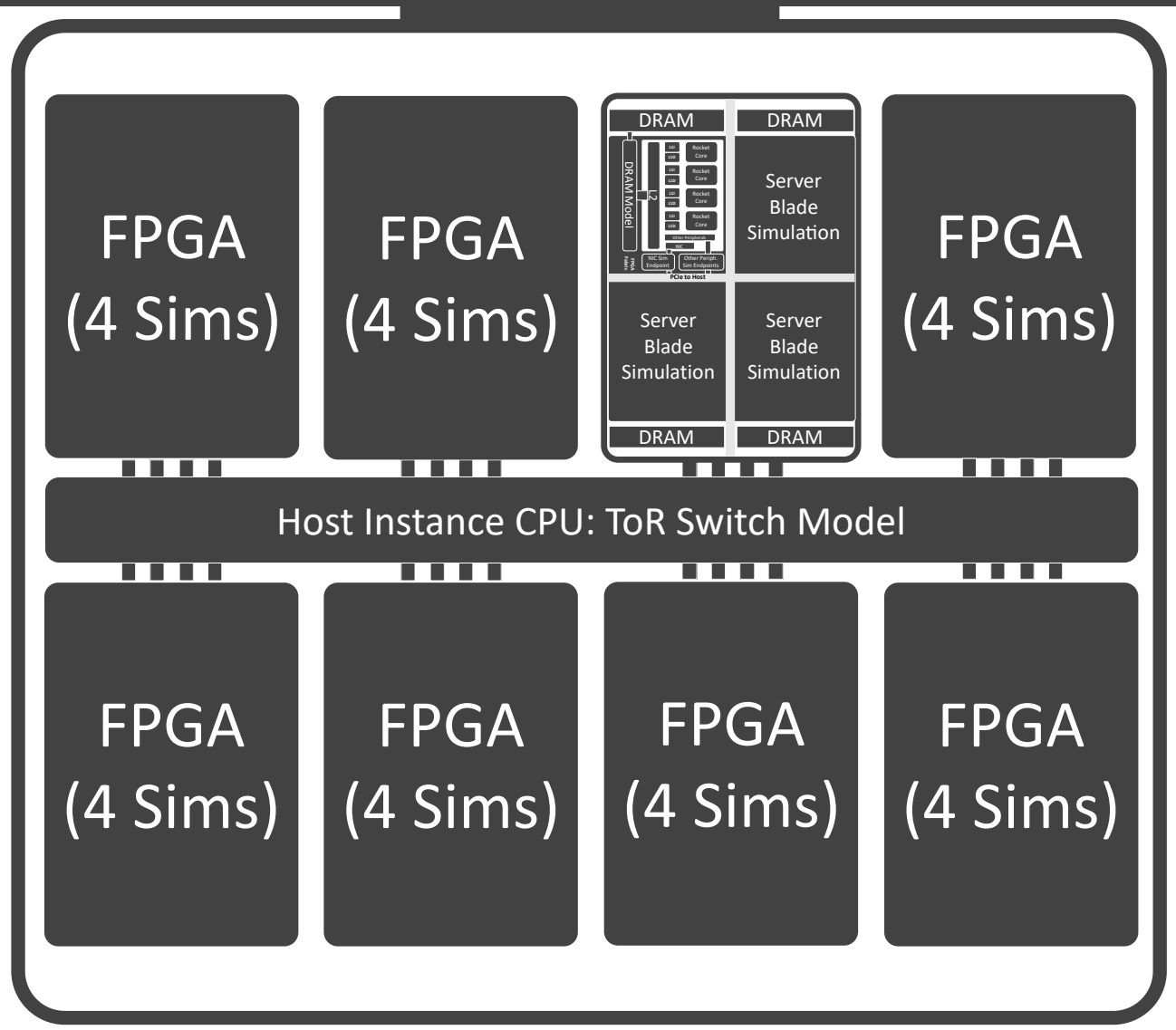
Sim Rate

- ~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

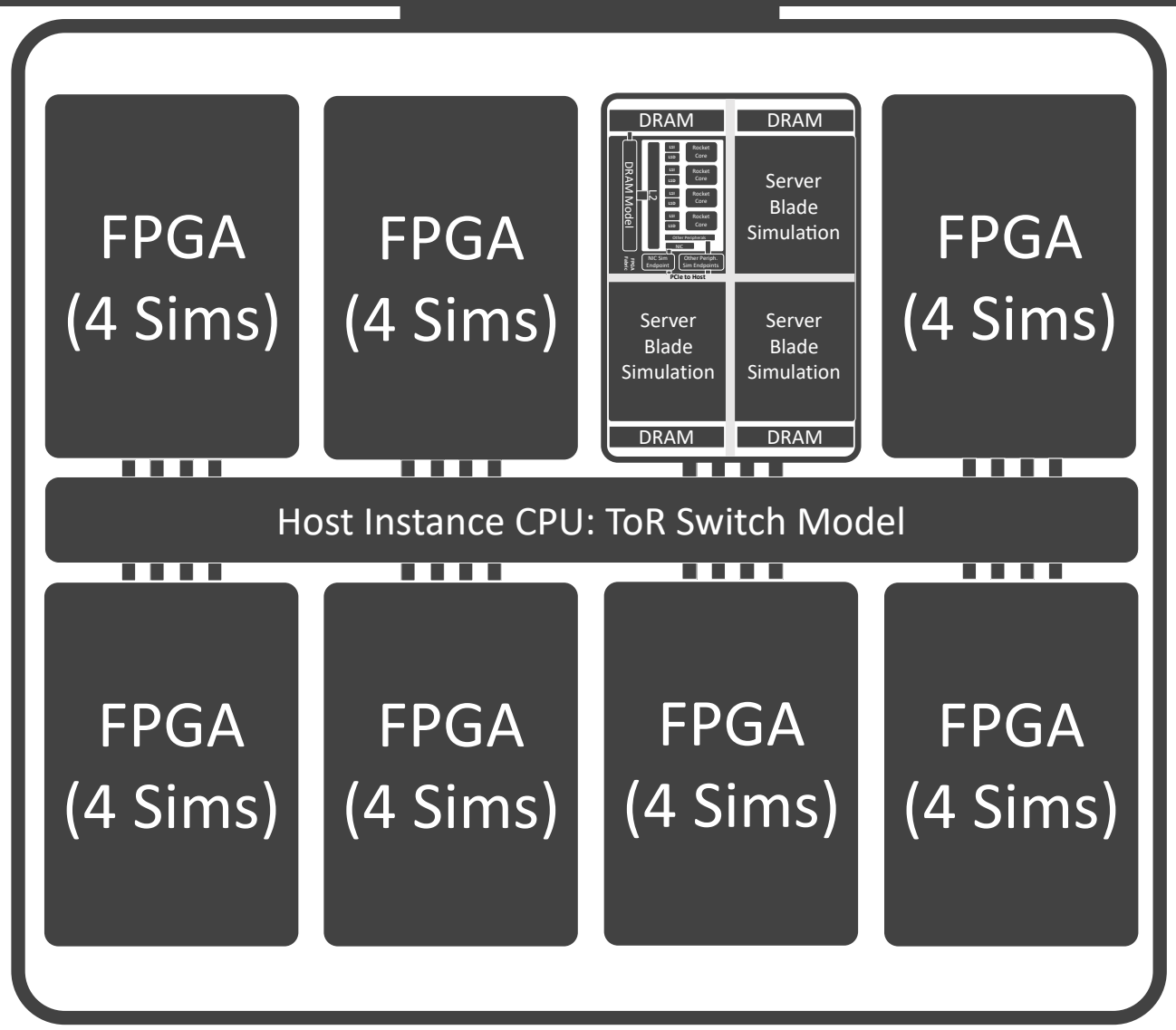
Sim Rate

- ~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

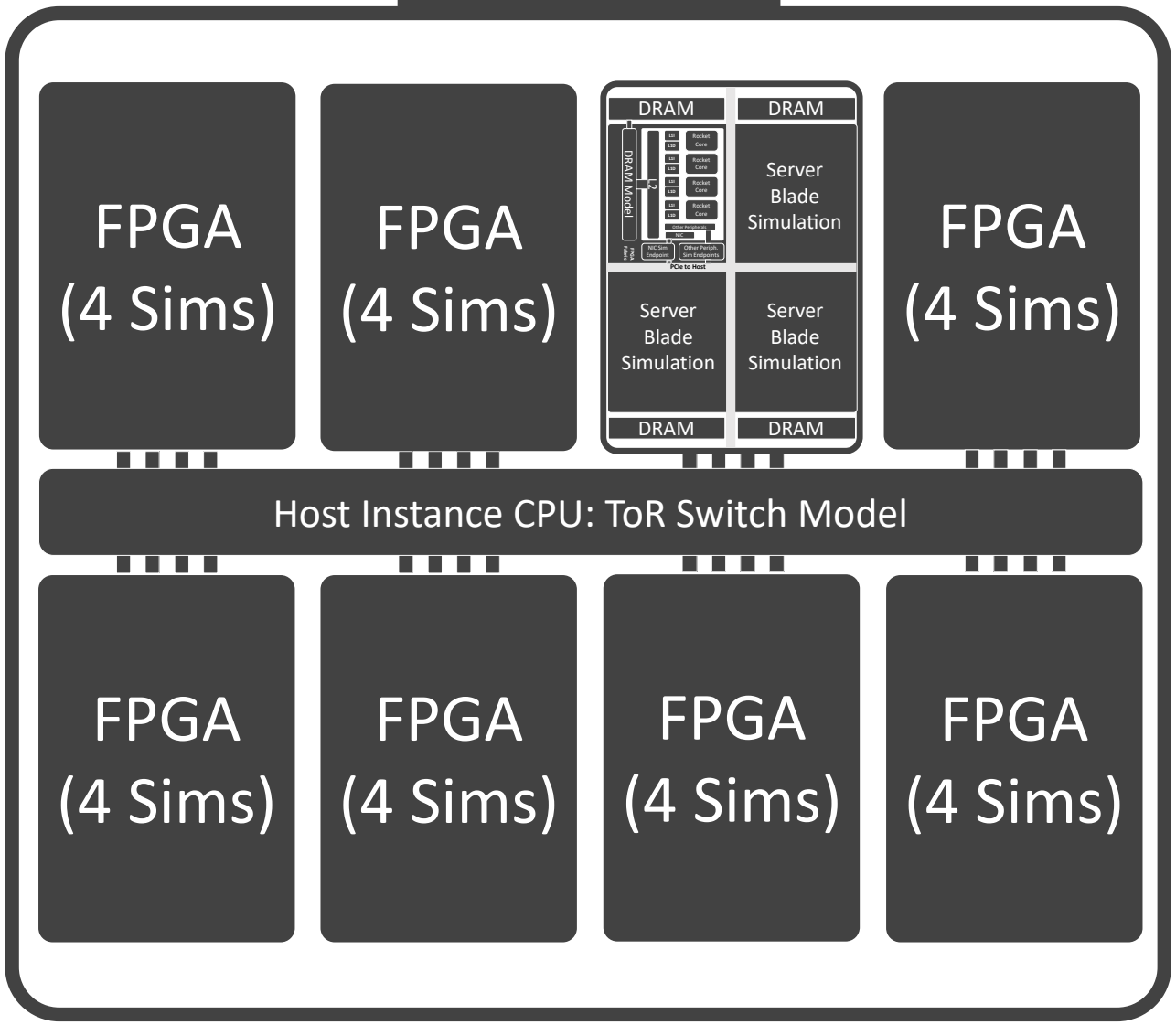
Resource Util.

- 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

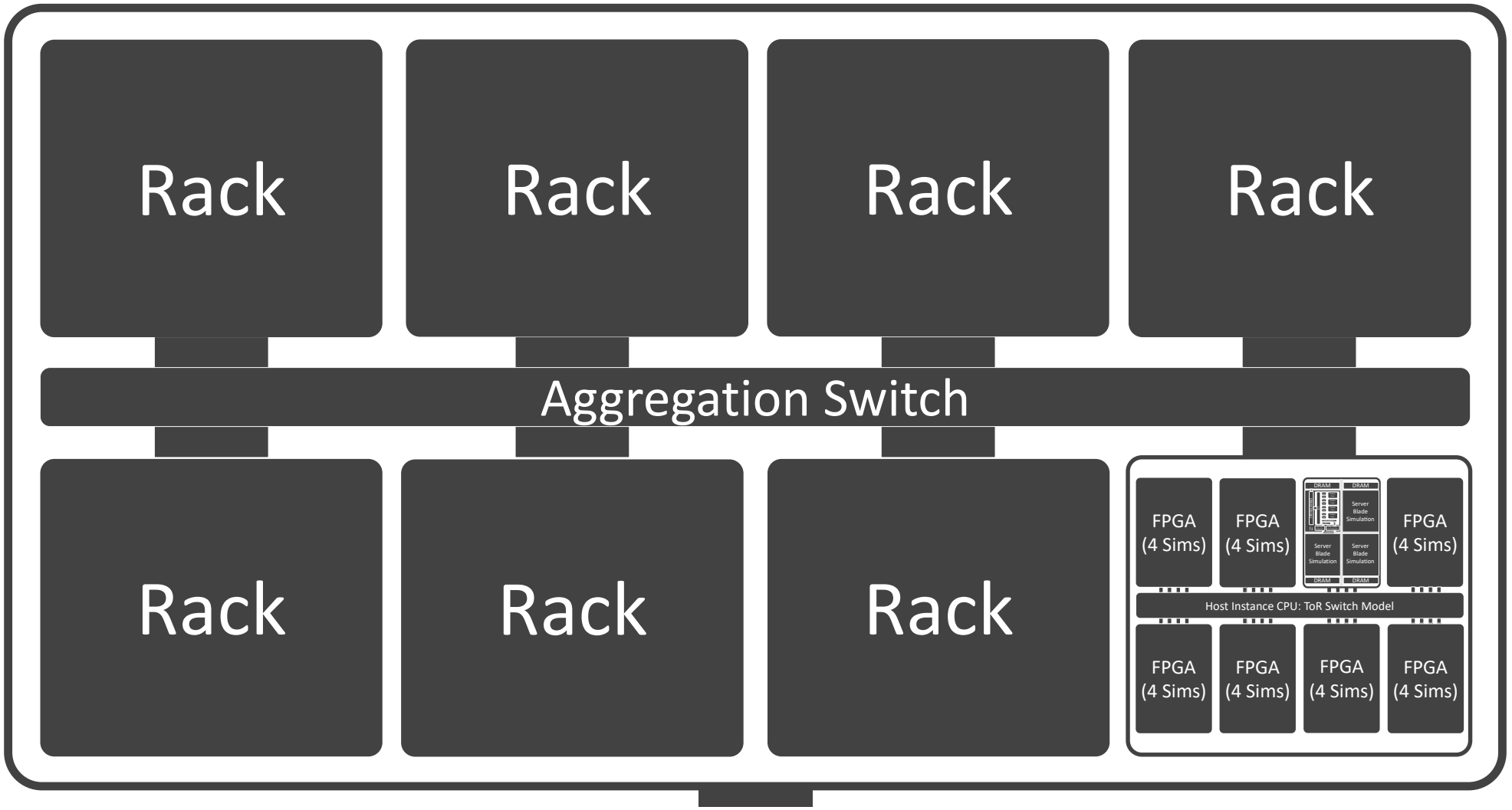
Resource Util.

- 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

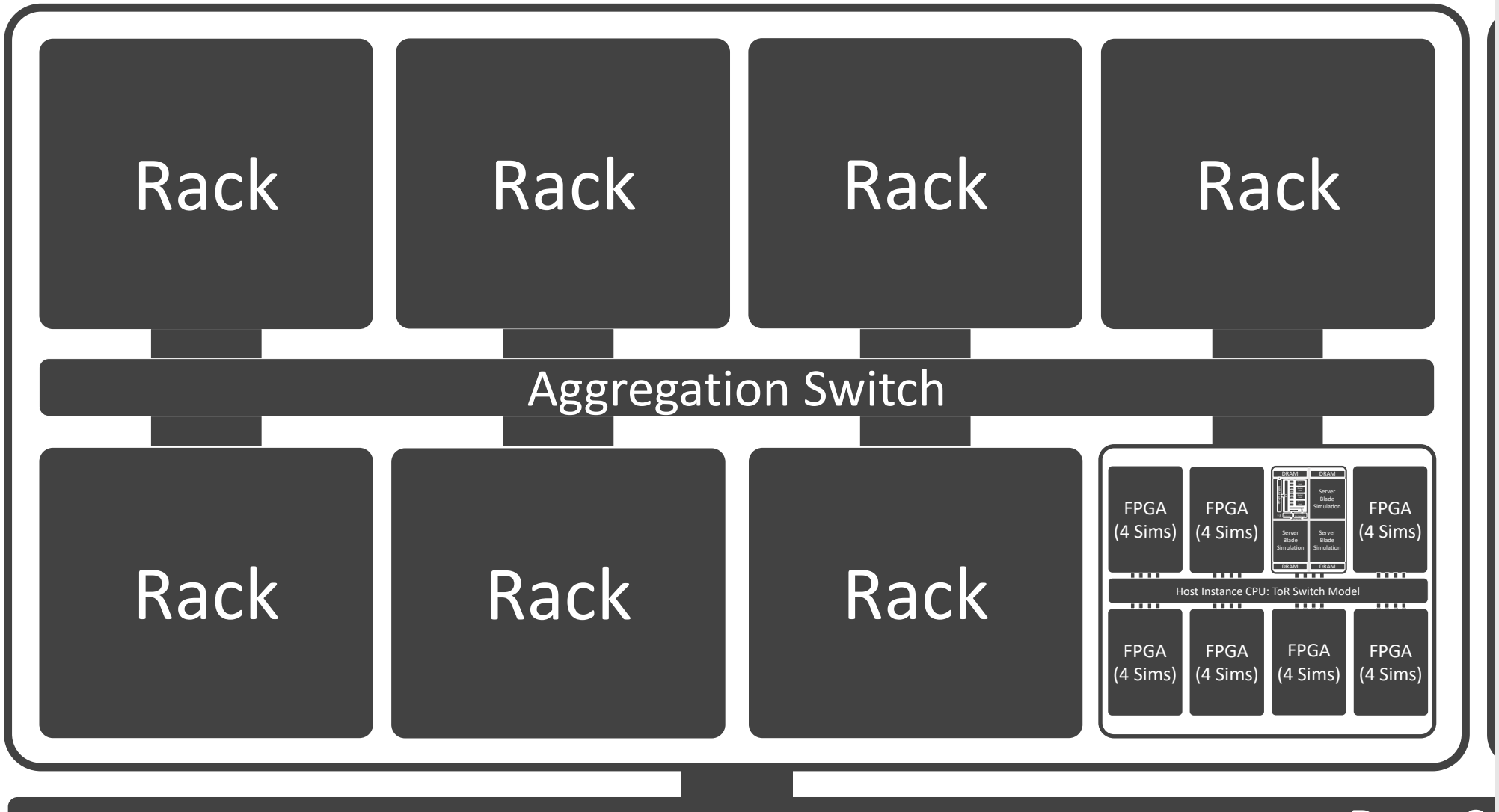
Resource Util.

- 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

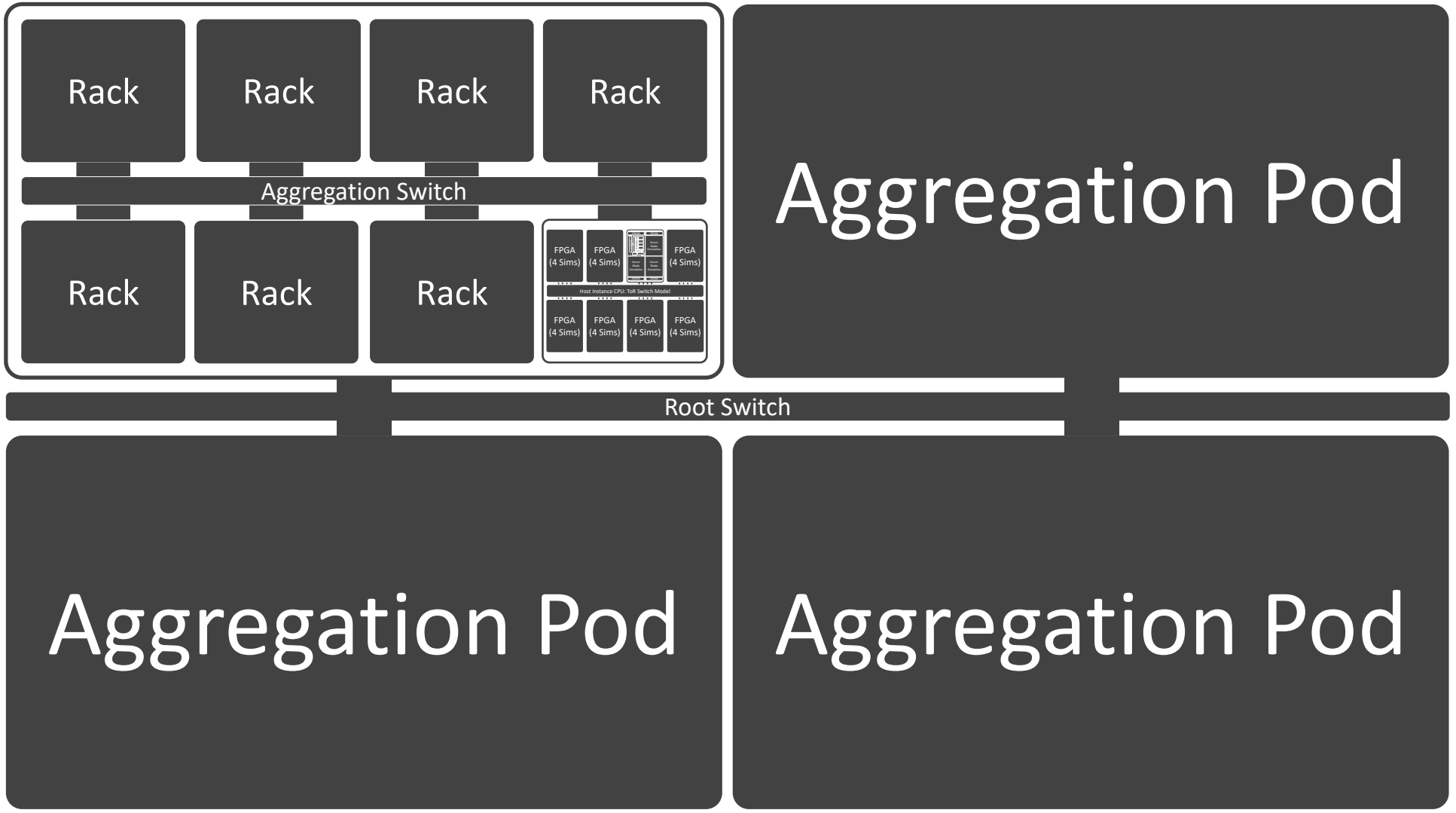
Resource Util.

- 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

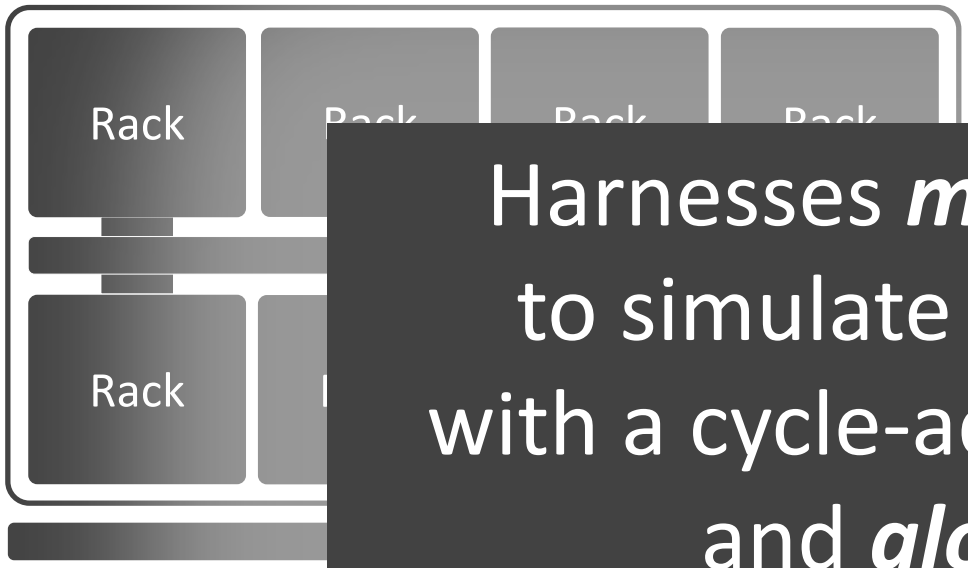
Resource Util.

- 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***

Aggregation Pod

Aggregation Pod

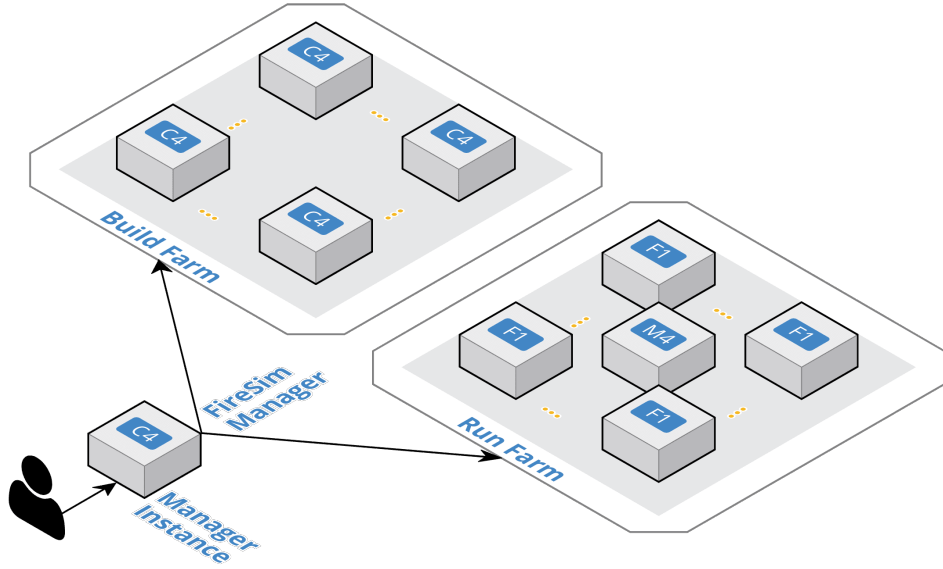
Modeled System
- 1024 Servers
- 6 Cores
- TB DDR3
- oRs, 4 Aggr, 1
- Gb/s, 2us

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

Sim Rate
- ~6.6 MHz (netw)

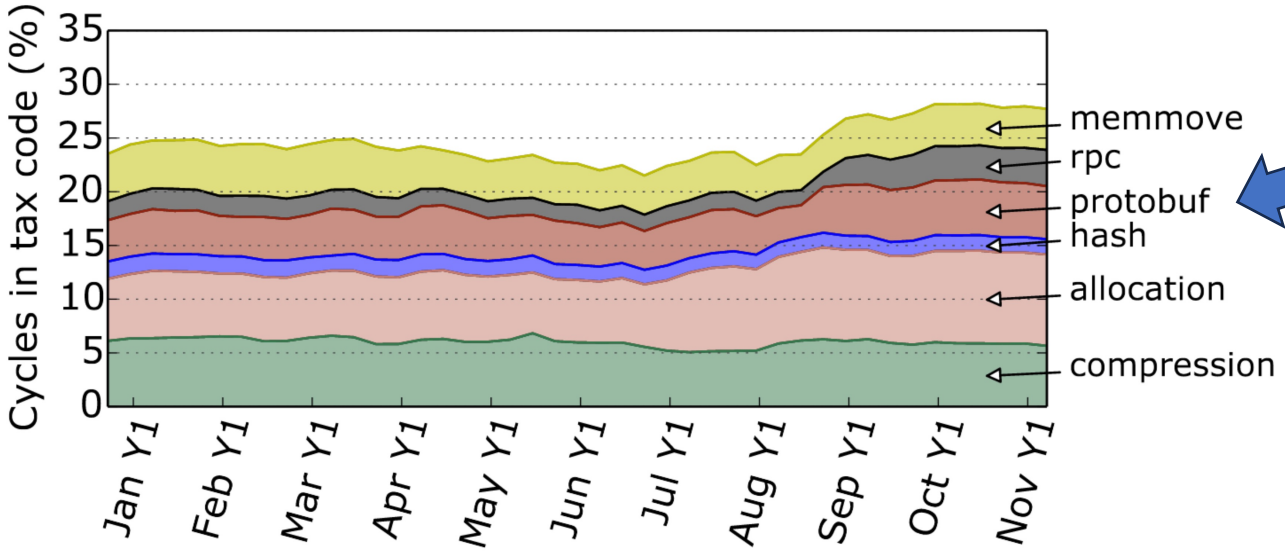
Productive Open-Source FPGA Simulation

- github.com/firesim/firesim, BSD Licensed
- An “easy” button for fast, FPGA-accelerated full-system simulation
  - 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

Eliminating *Datacenter Taxes*



Datacenter Tax Overheads, Google, 2015

A Hardware Accelerator for Protocol Buffers

S. Karandikar, et. al. w/Google
MICRO ’21
Honorable Mention, IEEE Micro Top Picks 2021
MICRO ‘21 Distinguished Artifact Award Winner

CDPU: Compression / Decompression Processing Unit

S. Karandikar, et. al. w/Google
ISCA ‘23

WSCs Summary

- WSCs power the services you use every day
  - Represent computer architecture on the biggest scale (along with supercomputer design)
- WSCs initially built from standard components like mid-range servers and network switches
  - Make HW reliable within reason, but software must be able to handle HW failures at this scale
- Designed to take advantage of ample *Request-Level Parallelism*
  - i.e., Handling unrelated user requests in parallel
- Today, large proliferation of custom hardware in WSCs
  - GPUs, TPUs, FPGAs, ASICs, etc.
- Improving hardware performance is important, but also focus on energy-efficiency, power-delivery, cooling
- Key overall metric for a hyperscaler: Total cost of operation (TCO)

Questions?

Learn more/Get involved in WSC research:

Email sagark@eecs.berkeley.edu

https://fires.im and https://sagark.org

References

- Unless otherwise noted, figures in slides prior to “WSC research at Berkeley” are taken from: “The Datacenter as a Computer”, 3rd. Ed. By Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan
  - https://link.springer.com/book/10.1007/978-3-031-01761-2