CS 152 Computer Architecture and Engineering
CS252 Graduate Computer Architecture

Lecture 23 I/O & Warehouse-Scale Computing

Krste Asanovic
Electrical Engineering and Computer Sciences
University of California at Berkeley

http://www.eecs.berkeley.edu/~krste
http://inst.eecs.berkeley.edu/~cs152

Last Time in Lecture 22

- Virtual machines date back to sharing mainframes for OS development
- Two main types of virtual machine:
  - User-level, runs a single application
  - OS-level, emulates a whole machine for an OS
- ABI – application binary interface
  - How an application is encoded in a file – ISA + system calls
- Emulating ISAs
  - Software interpreter
  - Ahead-of-time binary translation
  - Dynamic translation
  - Hardware emulation
- Hypervisors
  - Type-1 present separate machines to guest OS (e.g., Xen)
  - Type-2 built as recursive virtual machine within OS (e.g., KVM)
- Increasing hardware support for virtualization
  - Some ISAs (including x86) not designed with virtualization in mind

I/O (Input/Output)

Computers useless without I/O

- Over time, literally thousands of forms of computer I/O: punch cards to brain interfaces

Broad categories:

- Secondary/Tertiary storage (flash/disk/tape)
- Network (Ethernet, WiFi, Bluetooth, LTE)
- Human-machine interfaces (keyboard, mouse, touchscreen, graphics, audio, video, neural,…)
- Printers (line, laser, inkjet, photo, 3D, …)
- Sensors (process control, GPS, heartrate, …)
- Actuators (valves, robots, car, …)

Mix of I/O devices is highly application-dependent

Interfacing to I/O Devices

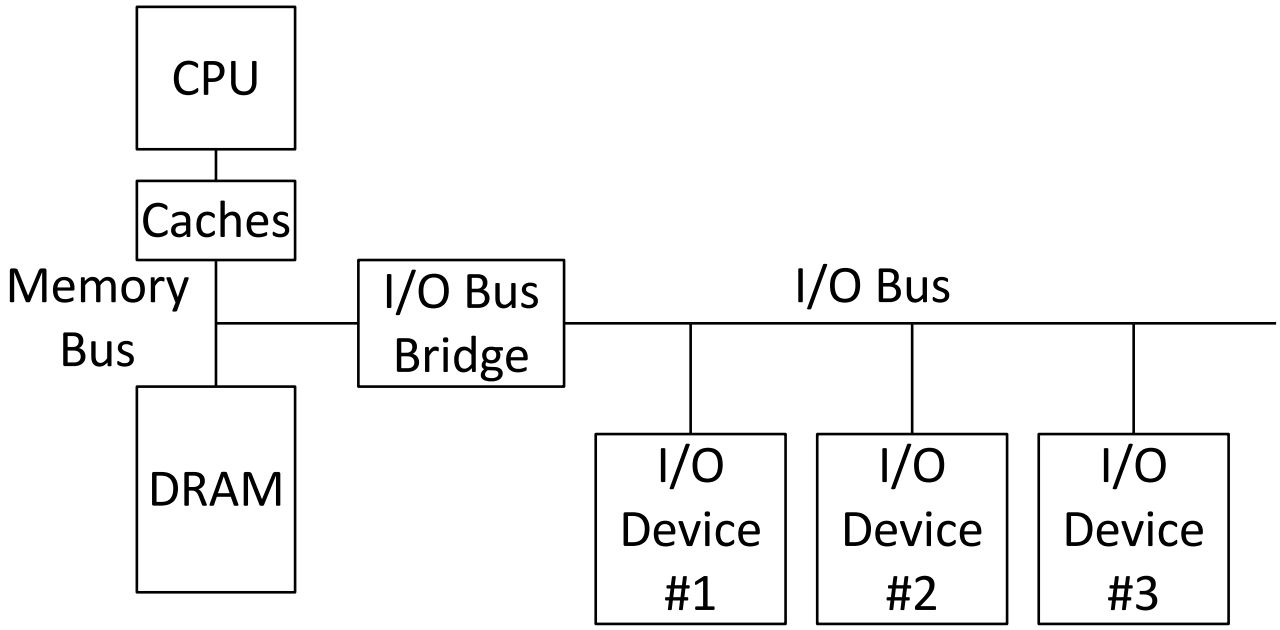
Two general strategies:

- Memory-mapped
  - I/O devices appear as memory locations to processor
  - Reads and writes to I/O device locations configure I/O and transfer data (using either programmed I/O or DMA)
- I/O channels
  - Architecture specifies commands to execute I/O commands over defined channels
  - I/O channel structure can be layered over memory-mapped device structure
- In addition to data transfer, need synchronization method
  - Polling: CPU checks status bits
  - Interrupts: Device interrupts CPU on event

Memory-Mapped I/O

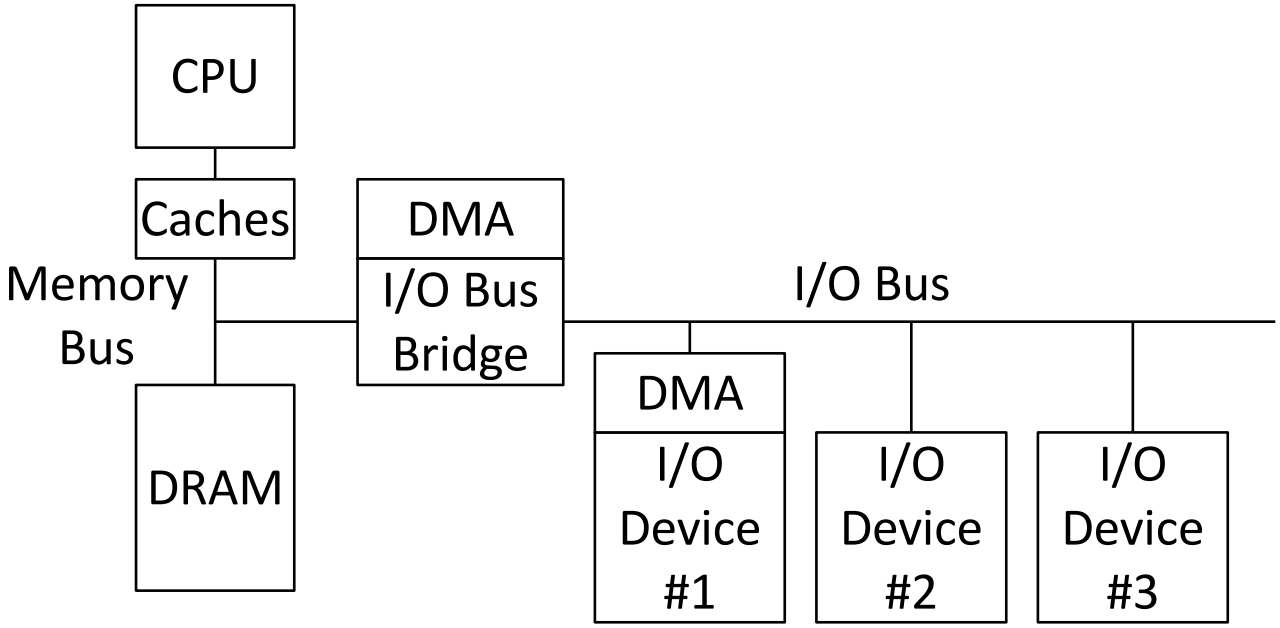
- Programmed I/O uses CPU to control I/O device using load and store instructions, with address specifying device register to access
  - Load and store can have side effect on device, e.g., writing a byte causes that byte to be output over a serial-port interface
- Usually, only privileged code can access I/O devices directly, to provide secure multiprogramming
  - System calls sometimes provided for application to open and reserve a device for exclusive access
- Processors provide “uncached” loads and stores to prevent caching of device registers
  - Usually by statically assigning portions of the physical address space as uncached, or by using configuration bits in page table entries (if system has virtual memory support)

Simple I/O Bus Structure



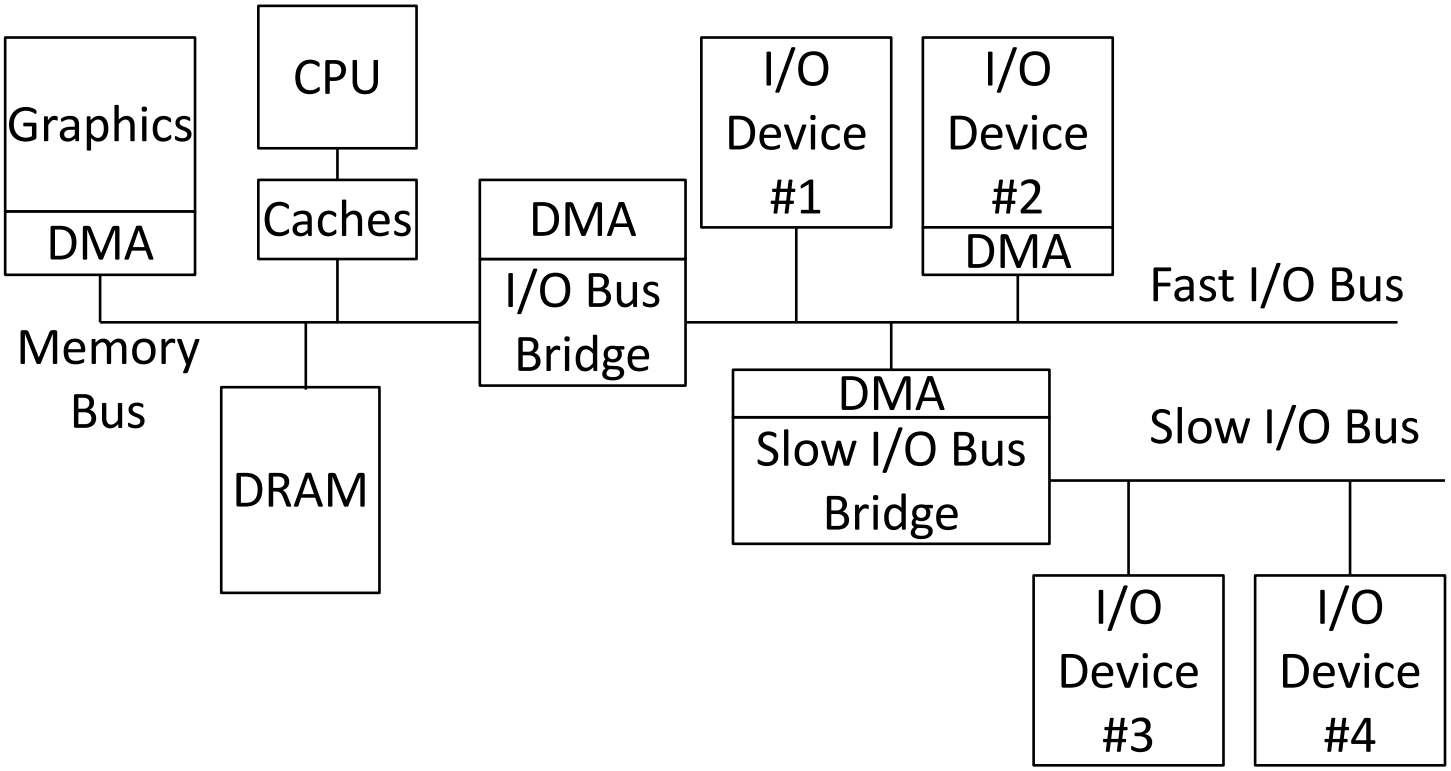
- Some range of physical addresses map to I/O bus devices
- I/O bridge reduces electrical loading on critical CPU-DRAM bus
- Devices can be “initiators”, initiating I/O bus transfers
- Devices can be “responders”, only responding to I/O bus transactions begun by initiators

DMA (Direct Memory Access)



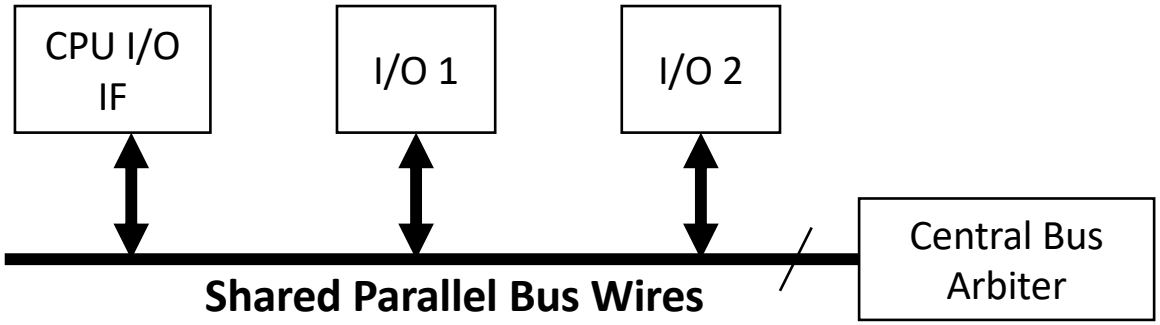
- DMA engines offload CPU by autonomously transferring data between I/O device and main memory. CPU can poll device or be interrupted to determine when DMA is done with transaction.
  - DMA programmed through memory-mapped registers
  - Some systems use dedicated processors inside DMA engines
- Often, many separate DMA engines in modern systems:
  - Centralized in I/O bridge (usually supporting multiple concurrent channels to different devices), works on responder-only I/O busses
  - Directly attached to each peripheral (if device is initiator)

More Complex Bus Structures



- Match speed of I/O connection to device demands
  - Special direct connection for graphics
  - Fast I/O bus for disk drives, ethernet
  - Slow I/O bus for keyboard, mouse, touchscreen
    - Reduces load on fast I/O bus + less bus logic needed on device

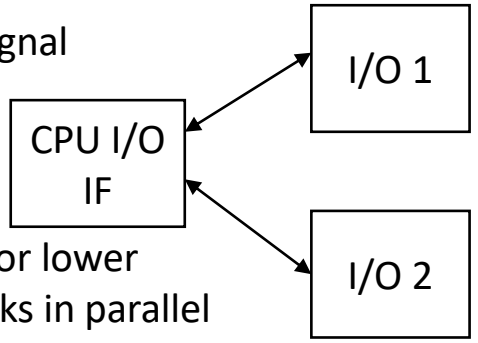
Move from Parallel to Serial for Off-Chip I/O



- Parallel bus clock rate limited by signal propagation across long bus (~100MHz)
- High power to drive large number of loaded bus lines
- Central bus arbiter adds latency to each transaction, sharing limits throughput
- Expensive parallel connectors and backplanes/cables (all devices pay costs)
- Examples: VMEbus, Sbus, ISA bus, PCI, SCSI, IDE

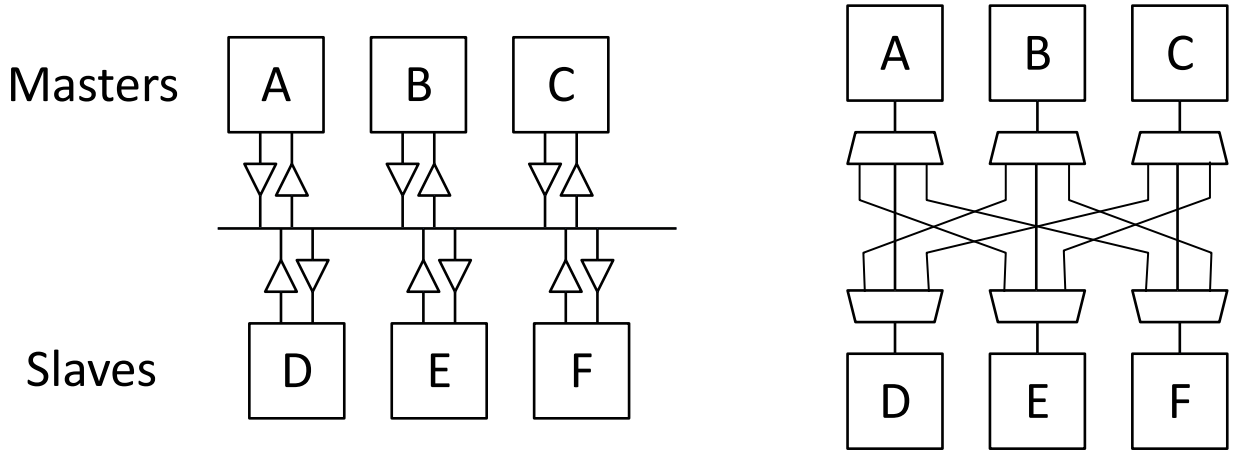
Dedicated Point-to-point Serial Links

- Point-to-point links run at multi-gigabit speed using advanced clock/signal encoding (requires lots of circuitry at each end)
- Lower power since only one well-behaved load
- Multiple simultaneous transfers
- Cheap cables and connectors (trade greater endpoint transistor cost for lower physical wiring cost), customize bandwidth per device using multiple links in parallel
- Examples: Ethernet, Infiniband, PCI Express, SATA, USB, Firewire, etc.



Move from Bus to Crossbar On-Chip

- Busses evolved in era where wires were expensive and had to be shared
- Bus tristate drivers problematic in VLSI standard cell flows, so replaced with combinational muxes
- Crossbar exploits density of on-chip wiring, allows multiple simultaneous transactions



Tristated Bus Crossbar

I/O and Memory Mapping

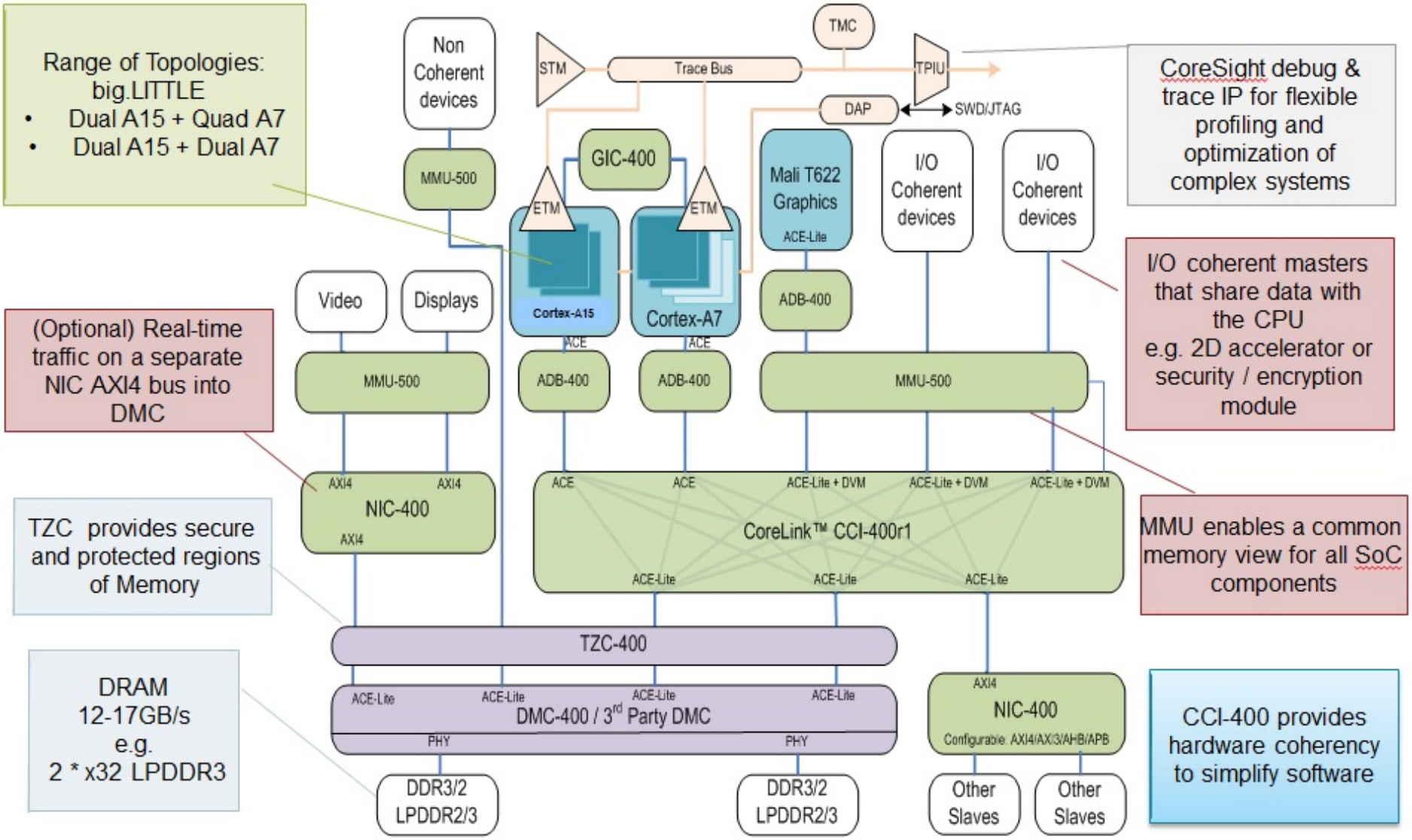
- I/O busses can be coherent or not
  - Non-coherent simpler, but might require flushing caches or only uncached accesses (much slower on modern processors)
  - Some I/O systems can cache coherently also (SGI Origin, TileLink, newer x86 systems)
- I/O can use virtual addresses and an IOMMU
  - Simplifies DMA into user address space, otherwise contiguous user segment needs scatter/gather by DMA engine
  - Provides protection from bad device drivers
  - Adds complexity to I/O device

Interrupts versus Polling

Two ways to detect I/O device status:

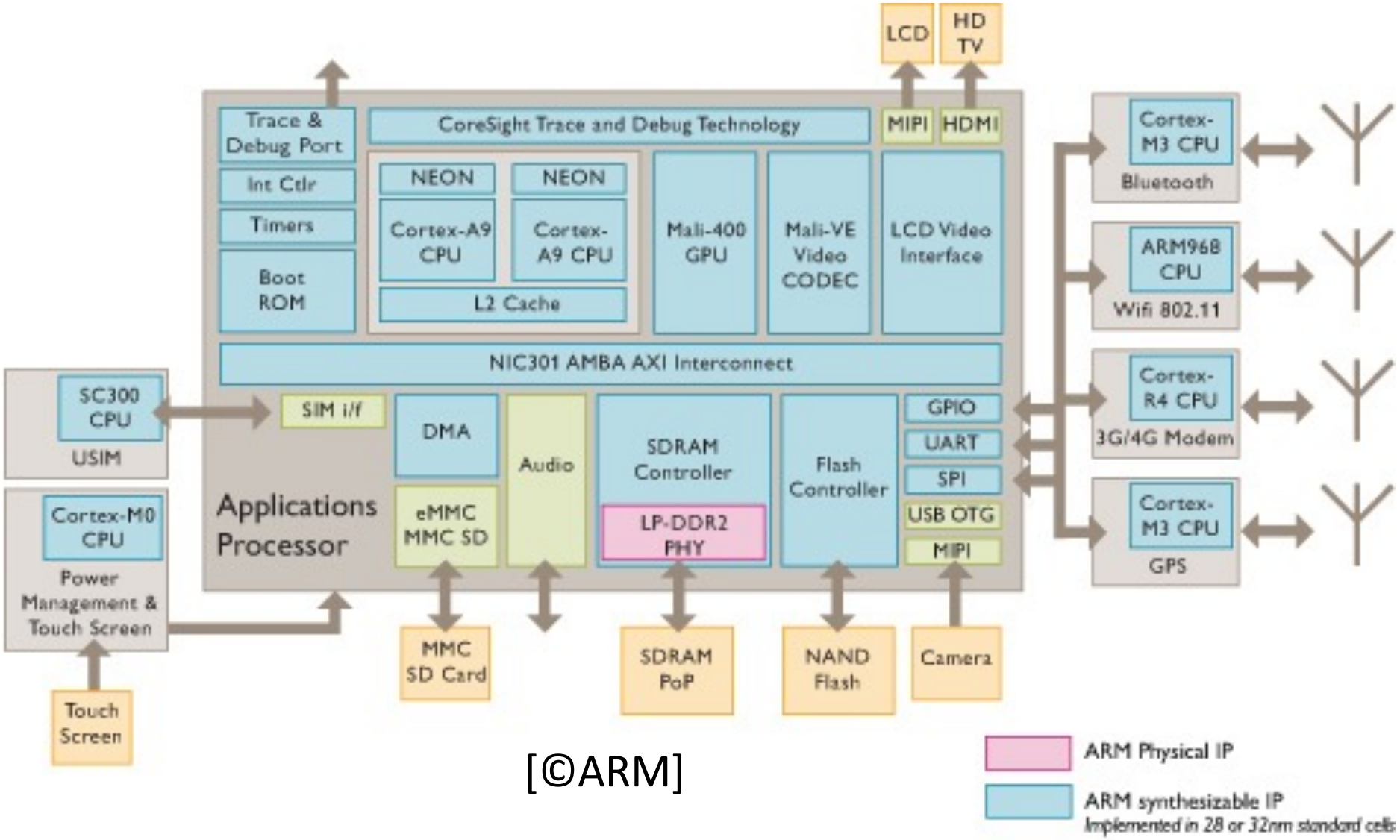
- Interrupts
  - \+ No CPU overhead until event
  - – Can happen at awkward time
  - – Large context-switch overhead on each event (trap flushes pipeline, disturbs current working set in cache/TLB)
- Polling
  - – CPU overhead on every poll
  - – Difficult to insert in all code
  - \+ Can control when handler occurs, reduce working set hit
- Hybrid approach:
  - – Interrupt on first event, keep polling in kernel until sure no more events, then back to interrupts

Example ARM SoC Structure



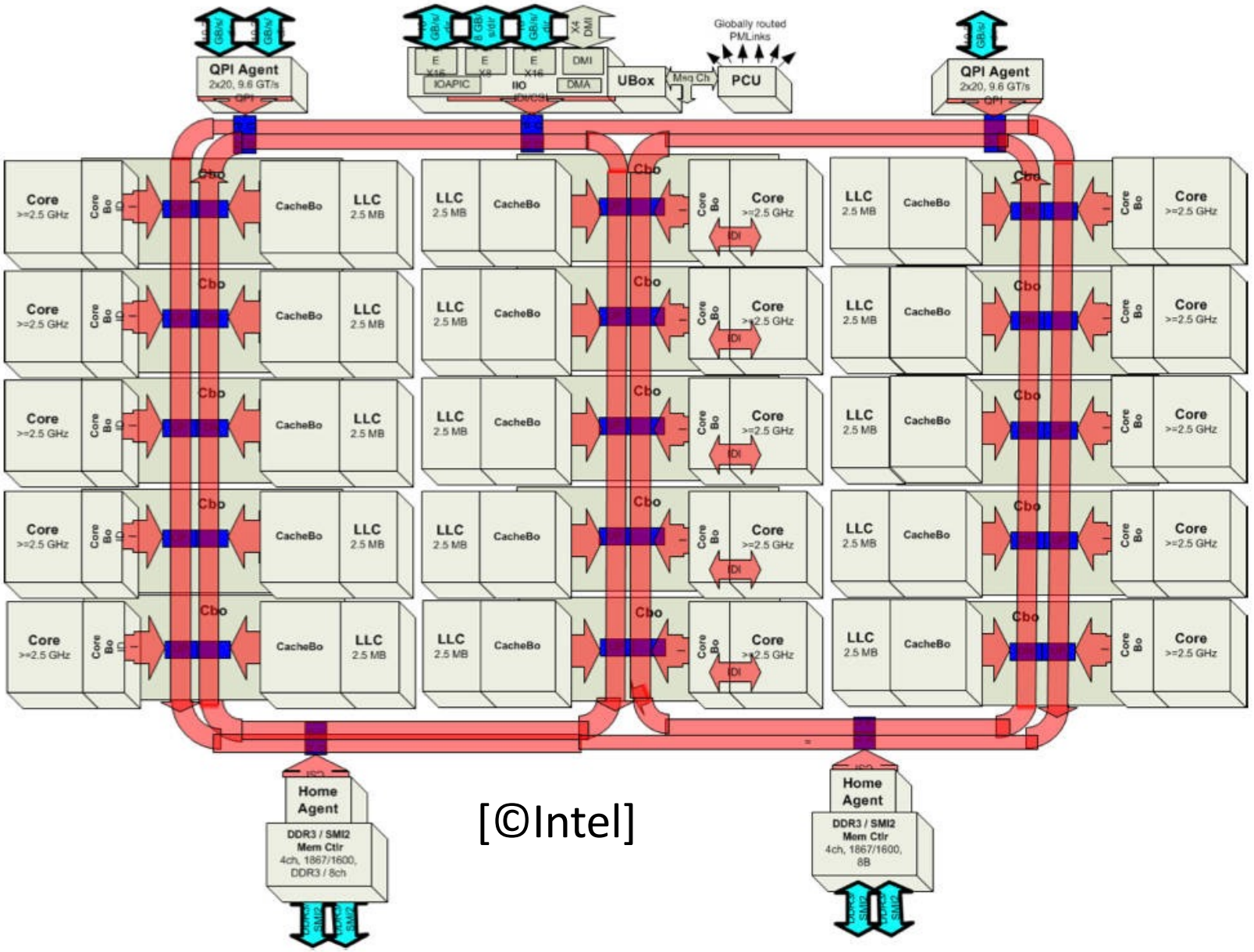
[©ARM]

ARM Sample Smartphone Diagram



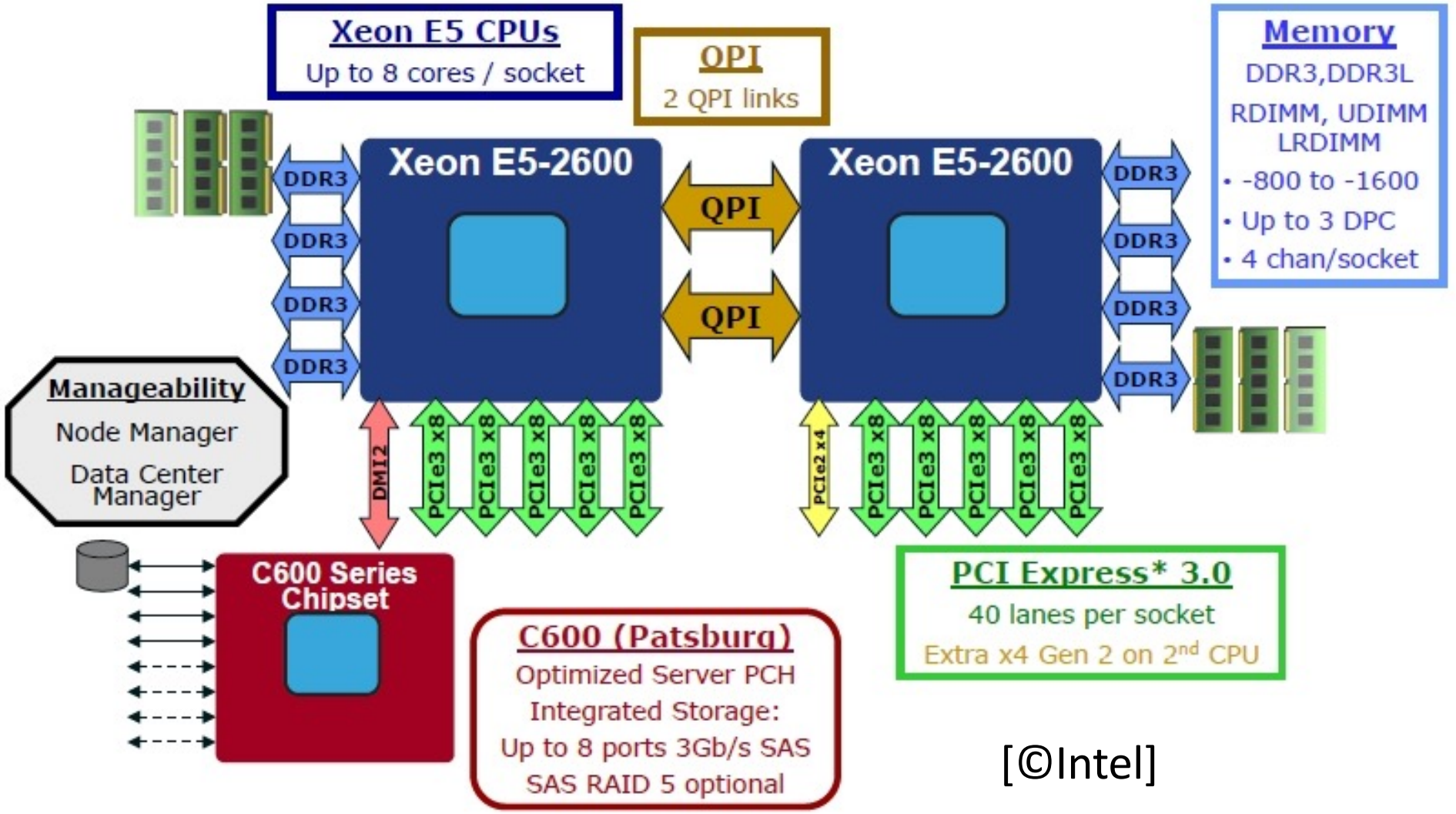
[©ARM]

Intel Ivy Bridge Server Chip I/O



[©Intel]

Intel Romley Server Platform



[©Intel]

CS152 Administrivia

- Lab 5 due on May 3rd
- Final Monday May 10th, 7PM-10PM
  - Covers entire class

CS252 Administrivia

- Thursday April 29th Project Checkpoint
  - Schedule 8-minute Zoom calls during discussion period
    - Same schedule as last time
  - Please have status slides, what's completed, what's left to do
- Final project presentations, Thursday May 6th, 4PM-6:30PM
  - All invited, including CS152 class
  - 20-minute presentation, plus Q&A time
  - Same order as project checkpoint meetings
- Final project paper due 11:59PM Friday May 14th
  - Send PDF file to all instructors
  - 10-page max in conference format

Warehouse-scale computers (WSCs)

- Provides Internet services
  - Search, social networking, online maps, video sharing, online shopping, email, cloud computing, etc.
- Differences with high-performance computing (HPC) "clusters":
  - Clusters have higher performance processors and network
  - Clusters emphasize thread-level parallelism, WSCs emphasize request-level parallelism
- Differences with datacenters:
  - Datacenters consolidate different machines and software into one location
  - Datacenters emphasize virtual machines and hardware heterogeneity in order to serve varied customers

WSC Characteristics

- Ample computational parallelism is not important
  - Most jobs are totally independent
  - “Request-level parallelism”
- Operational costs count
  - Power consumption is a primary, not secondary, constraint when designing system
- Scale and its opportunities and problems
  - Can afford to build customized systems since WSC require volume purchase
- Location counts
  - Real estate, power cost; Internet, end-user, and workforce availability
- Computing efficiently at low utilization
- Scale and the opportunities/problems associated with scale
  - Unique challenges: custom hardware, failures
  - Unique opportunities: bulk discounts

Efficiency and Cost of WSC

- Location of WSC
  - Proximity to Internet backbones, electricity cost, property tax rates, low risk from earthquakes, floods, and hurricanes

Amazon Sites

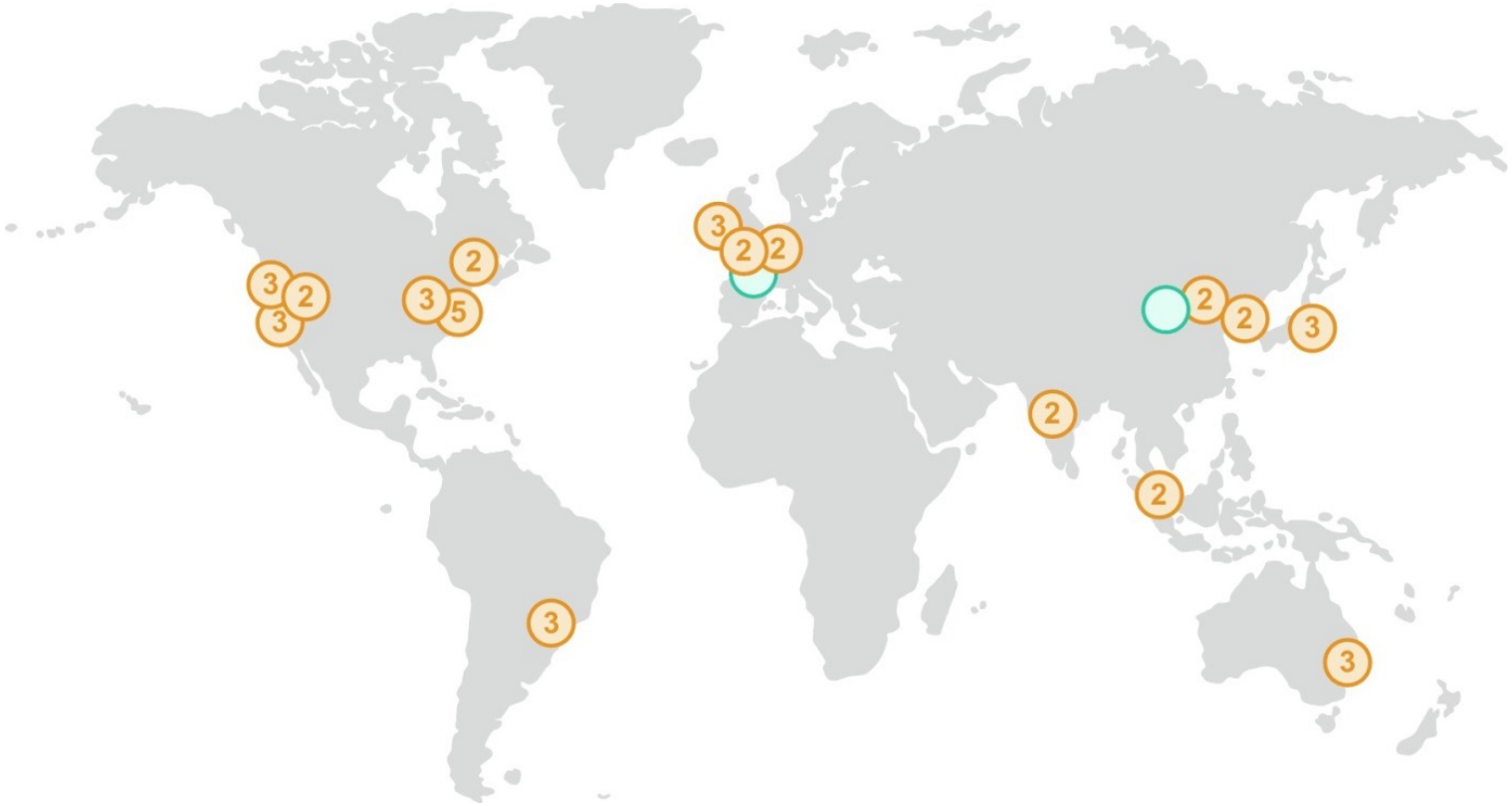


Figure 6.18 In 2017 AWS had 16 sites ("regions"), with two more opening soon. Most sites have two to three *availability zones*, which are located nearby but are unlikely to be affected by the same natural disaster or power outage, if one were to occur. (The number of availability zones are listed inside each circle on the map.) These 16 sites or regions collectively have 42 availability zones. Each availability zone has one or more WSCs. https://aws.amazon.com/about-aws/global-infrastructure/.

Google Sites

Figure 6.19 In 2017 Google had 15 sites. In the Americas: Berkeley County, South Carolina; Council Bluffs, Iowa; Douglas County, Georgia; Jackson County, Alabama; Lenoir, North Carolina; Mayes County, Oklahoma; Montgomery County, Tennessee; Quilicura, Chile; and The Dalles, Oregon. In Asia: Changhua County, Taiwan; Singapore. In Europe: Dublin, Ireland; Eemshaven, Netherlands; Hamina, Finland; St. Ghislain, Belgium. https://www.google.com/about/datacenters/inside/locations/.

Microsoft Sites

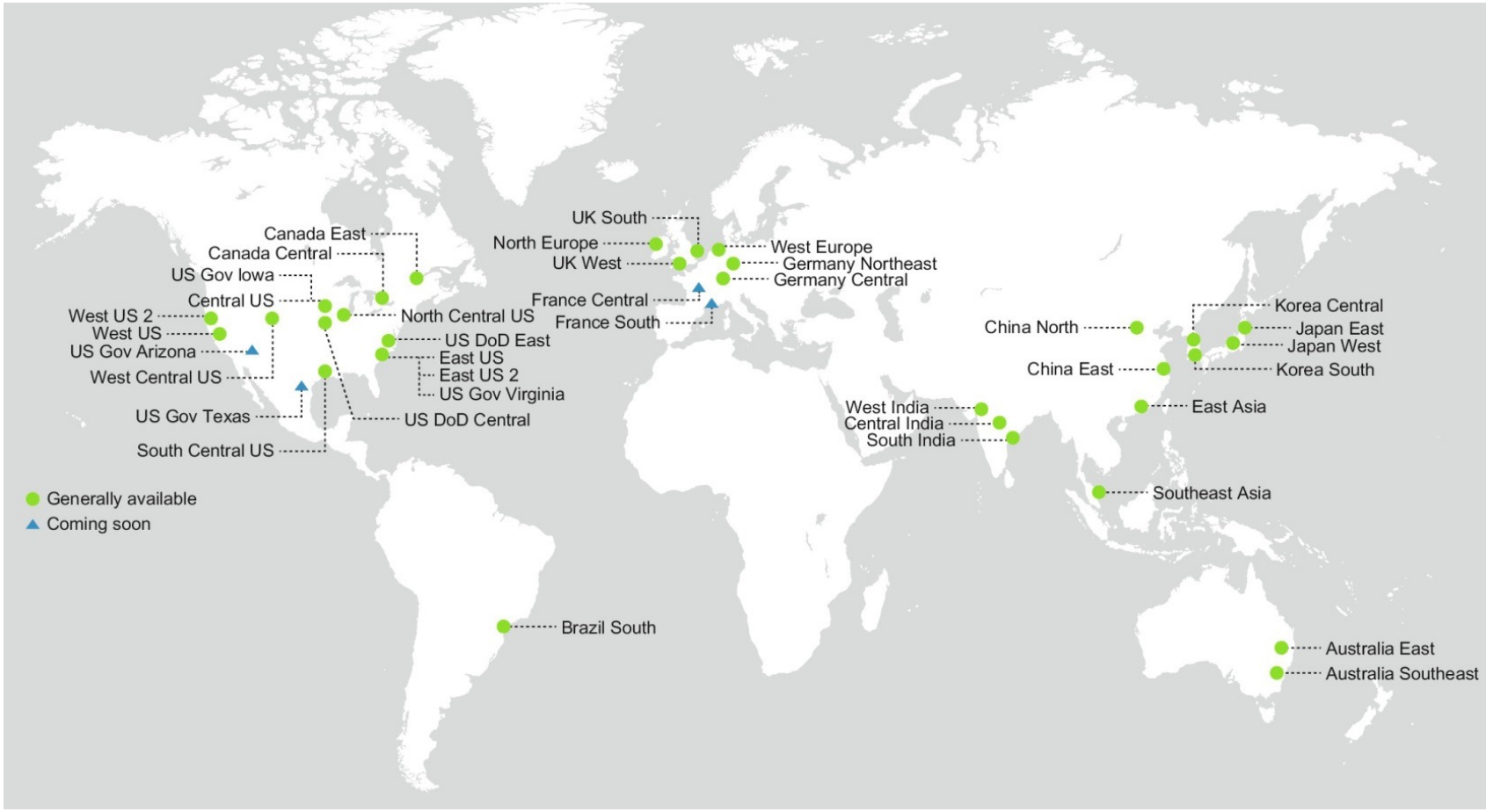
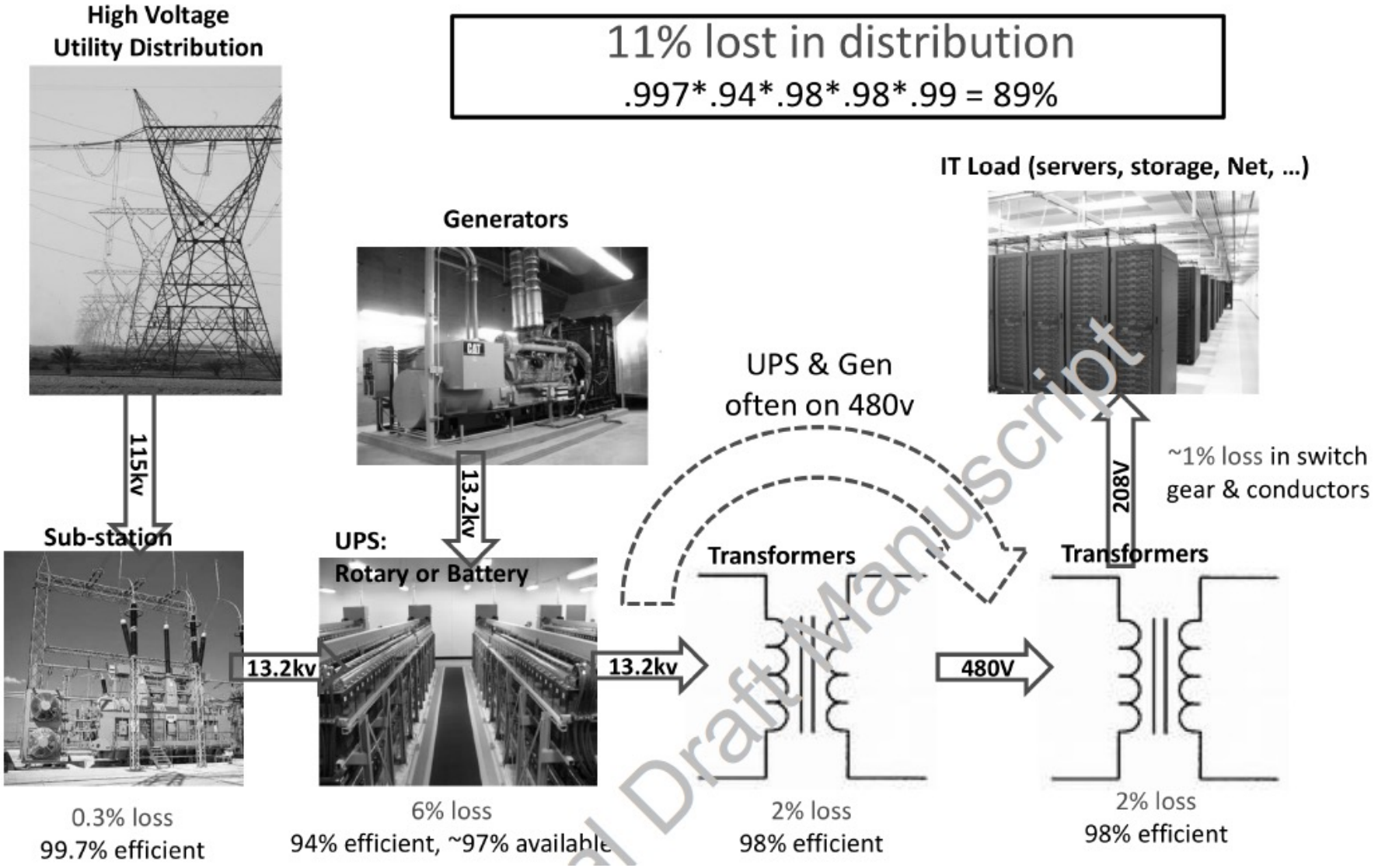


Figure 6.20 In 2017 Microsoft had 34 sites, with four more opening soon. https://azure.microsoft.com/en-us/regions/.

Power Distribution



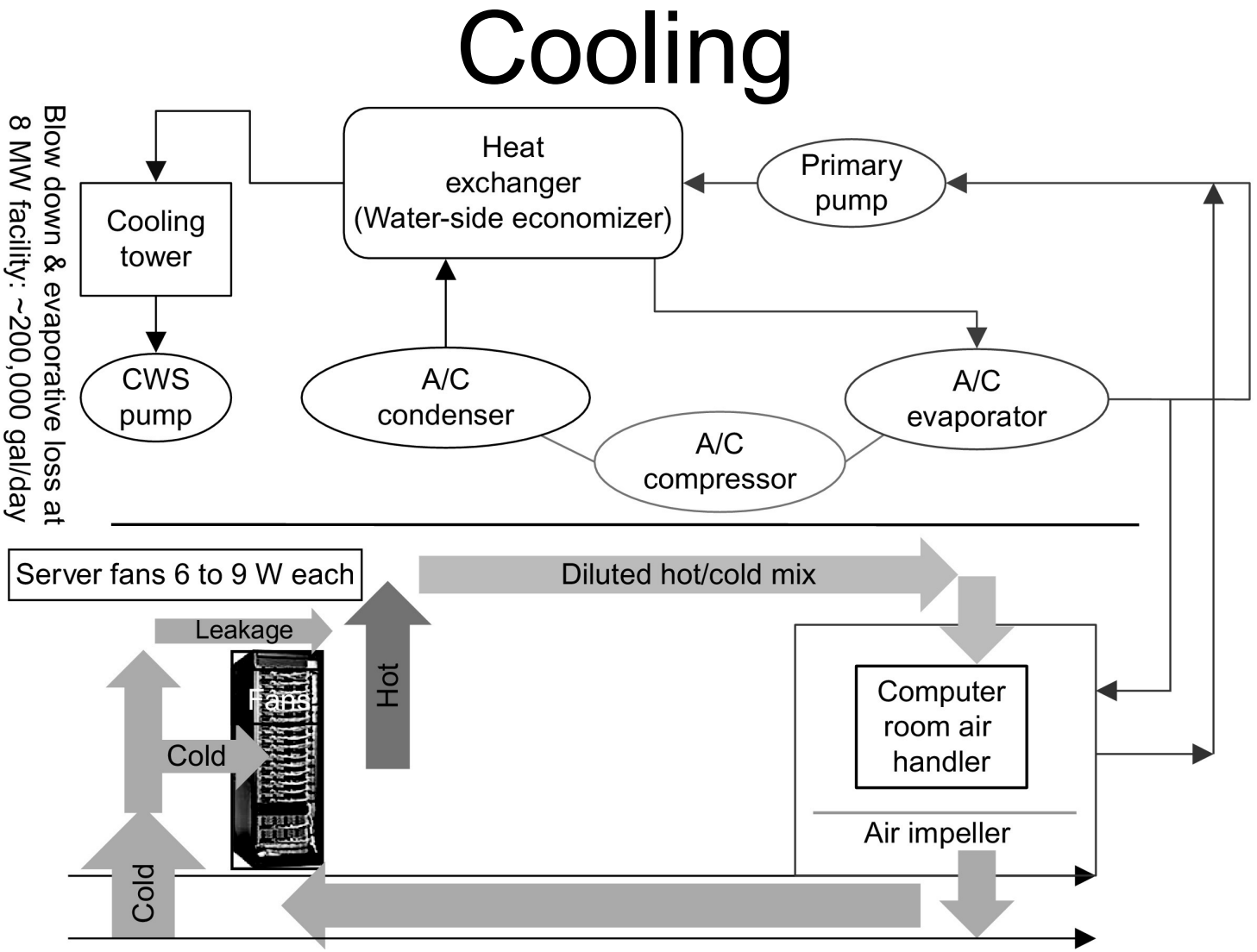


Figure 6.9 Mechanical design for cooling systems. CWS stands for circulating water system. From Hamilton, J., 2010. Cloud computing economies of scale. In: Paper Presented at the AWS Workshop on Genomics and Cloud Computing, June 8, 2010, Seattle, WA. http://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_GenomicsCloud20100608.pdf.

Infrastructure and Costs of WSC

- Cooling system also uses water (evaporation and spills)
  - E.g. 70,000 to 200,000 gallons per day for an 8 MW facility
- Power cost breakdown:
  - Chillers: 30-50% of the power used by the IT equipment
  - Air conditioning: 10-20% of the IT power, mostly due to fans
- How many servers can a WSC support?
  - Each server:
    - "Nameplate power rating" gives maximum power consumption
    - To get actual, measure power under actual workloads
  - Oversubscribe cumulative server power by 40%, but monitor power closely

Infrastructure and Costs of WSC

- Determining the maximum server capacity
  - Nameplate power rating: maximum power that a server can draw
  - Better approach: measure under various workloads
  - Oversubscribe by 40%
- Typical power usage by component:
  - Processors: 42%
  - DRAM: 12%
  - Disks: 14%
  - Networking: 5%
  - Cooling: 15%
  - Power overhead: 8%
  - Miscellaneous: 4%

Power Utilization Effectiveness (PUE)

- = Total facility power / IT equipment power

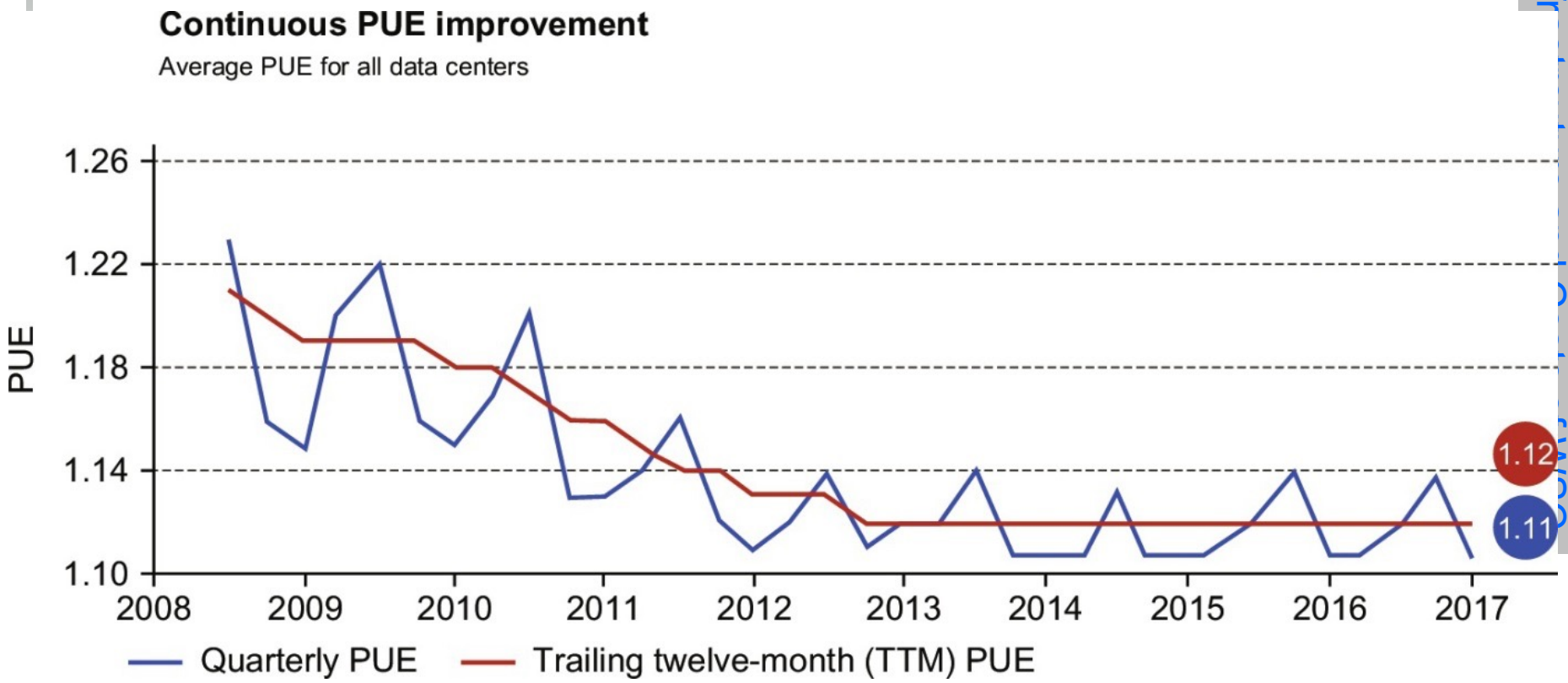


Figure 6.11 Average power utilization efficiency (PUE) of the 15 Google WSCs between 2008 and 2017. The spiking line is the quarterly average PUE, and the straighter line is the trailing 12-month average PUE. For Q4 2016, the averages were 1.11 and 1.12, respectively.

Performance, Latency

- Latency is important metric because it is seen by users
- Bing study: users will use search less as response time increases
- Service Level Objectives (SLOs)/Service Level Agreements (SLAs)
  - E.g. 99% of requests be below 100 ms

Server delay (ms)	Increased time to next click (ms)	Queries/ user	Any clicks/ user	User satisfaction	Revenue/ user
50	–	–	–	–	–
200	500	–	–0.3%	–0.4%	–
500	1200	–	–1.0%	–0.9%	–1.2%
1000	1900	–0.7%	–1.9%	–1.6%	–2.8%
2000	3100	–1.8%	–4.4%	–3.8%	–4.3%

Outages and Anomalies

Approx. number events in 1st year	Cause	Consequence
1 or 2	Power utility failures	Lose power to whole WSC; doesn't bring down WSC if UPS and generators work (generators work about 99% of time).
4	Cluster upgrades	Planned outage to upgrade infrastructure, many times for evolving networking needs such as recabling, to switch firmware upgrades, and so on. There are about nine planned cluster outages for every unplanned outage.
1000s	Hard-drive failures	2%–10% annual disk failure rate (Pinheiro et al., 2007)
	Slow disks	Still operate, but run 10× to 20× more slowly
	Bad memories	One uncorrectable DRAM error per year (Schroeder et al., 2009)
	Misconfigured machines	Configuration led to ∼30% of service disruptions (Barroso and HÖlzle, 2009)
	Flaky machines	1% of servers reboot more than once a week (Barroso and HÖlzle, 2009)
5000	Individual server crashes	Machine reboot; typically takes about 5 min (caused by problems in software or hardware).

Figure 6.1 List of outages and anomalies with the approximate frequencies of occurrences in the first year of a new cluster of 2400 servers. We label what Google calls a cluster an *array*; see Figure 6.5. Based on Barroso, L.A., 2010. Warehouse Scale Computing [keynote address]. In: Proceedings of ACM SIGMOD, June 8–10, 2010, Indianapolis, IN.

CPU Utilization is Usually Low

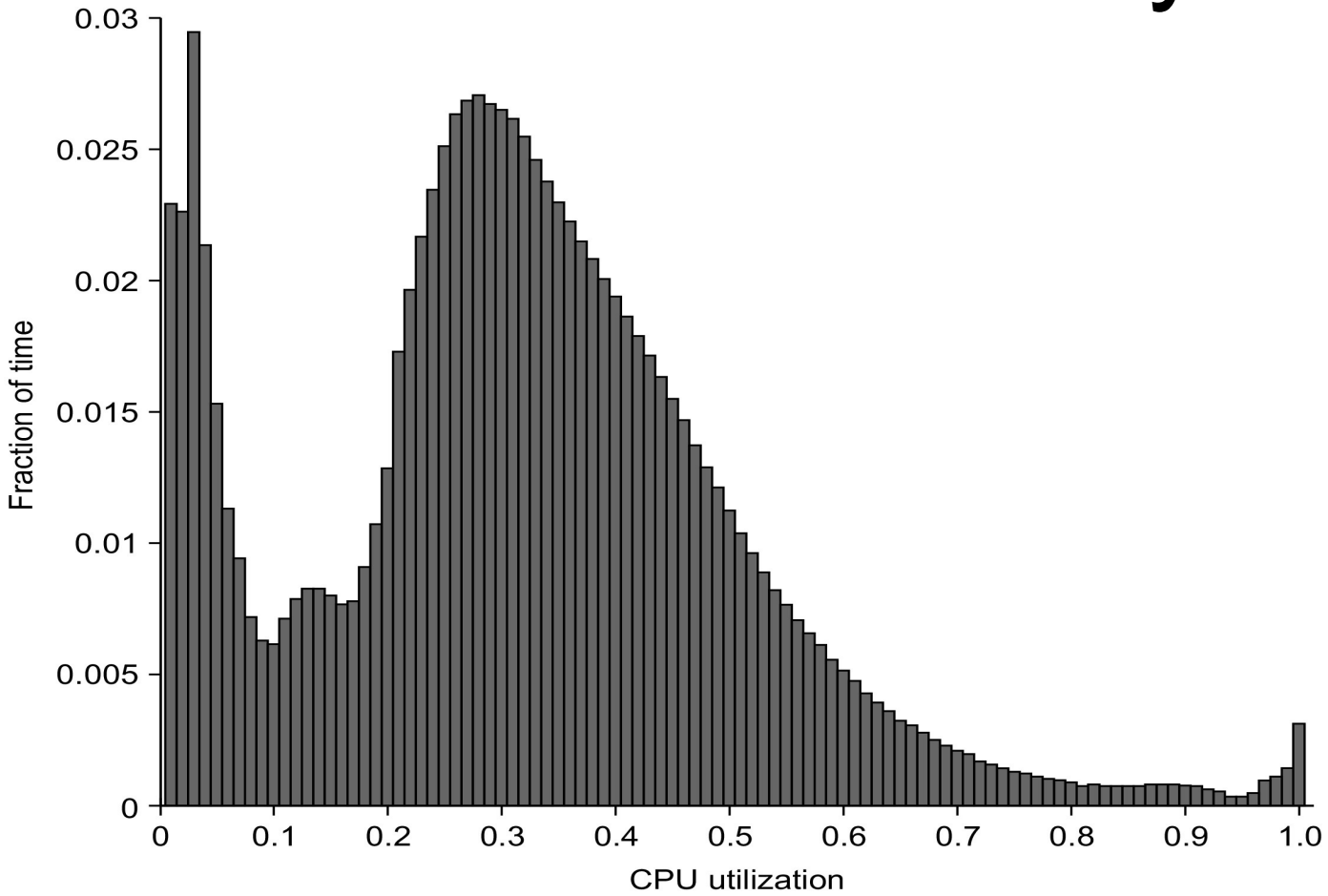


Figure 6.3 Average CPU utilization of more than 5000 servers during a 6-month period at Google. Servers are rarely completely idle or fully utilized, instead operating most of the time at between 10% and 50% of their maximum utilization. The third column from the right in Figure 6.4 calculates percentages plus or minus 5% to come up with the weightings; thus 1.2% for the 90% row means that 1.2% of servers were between 85% and 95% utilized. From Figure 1 in Barroso, L.A., Hölzle, U., 2007. The case for energy-proportional computing. IEEE Comput. 40 (12), 33–37.

Google WSC Rack

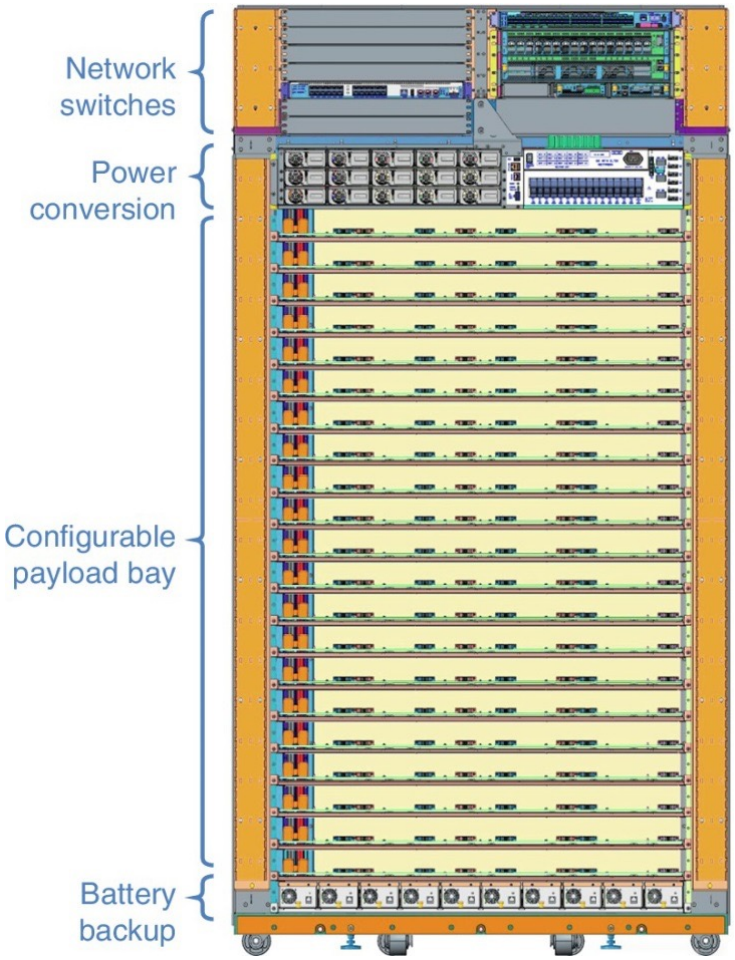


Figure 6.30 A Google rack for its WSC. Its dimensions are about 7 ft high, 4 ft wide, and 2 ft deep (2 m × 1.2 m × 0.5 m). The Top of Rack switches are indeed at the top of this rack. Next comes the power converter that converts from 240 V AC to 48 V DC for the servers in the rack using a bus bar at the back of the rack. Next is the 20 slots (depending on the height of the server) that can be configured for the various types of servers that can be placed in the rack. Up to four servers can be placed per tray. At the bottom of the rack are high-efficiency distributed modular DC uninterruptible power supply (UPS) batteries.

Array of Racks

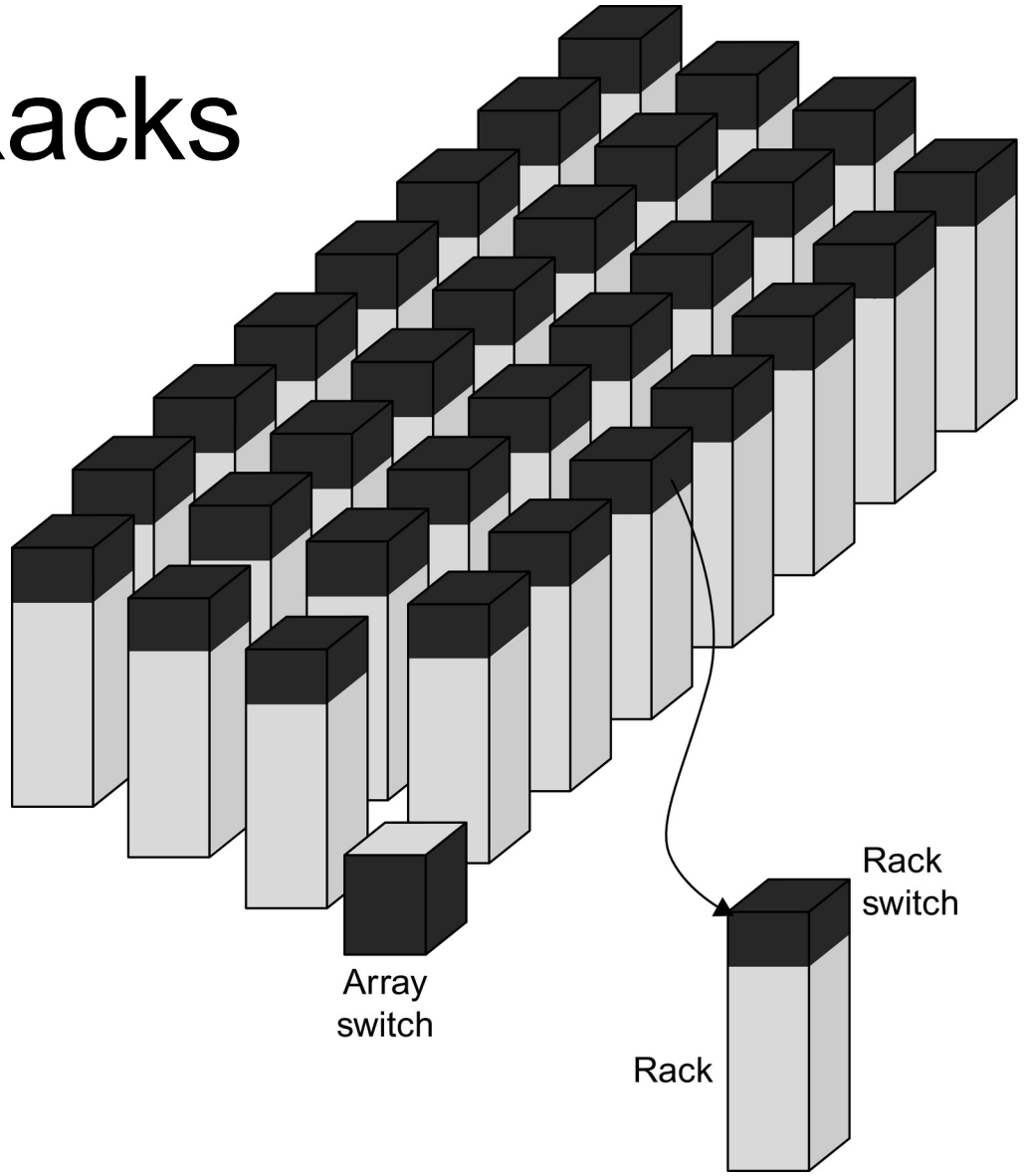


Figure 6.5 Hierarchy of switches in a WSC. Based on Figure 1.1 in Barroso, L.A., Clidaras, J., Hölzle, U., 2013. The datacenter as a computer: an introduction to the design of warehouse-scale machines. Synth. Lect. Comput. Architect. 8 (3), 1–154.

Older WSC Network Structure

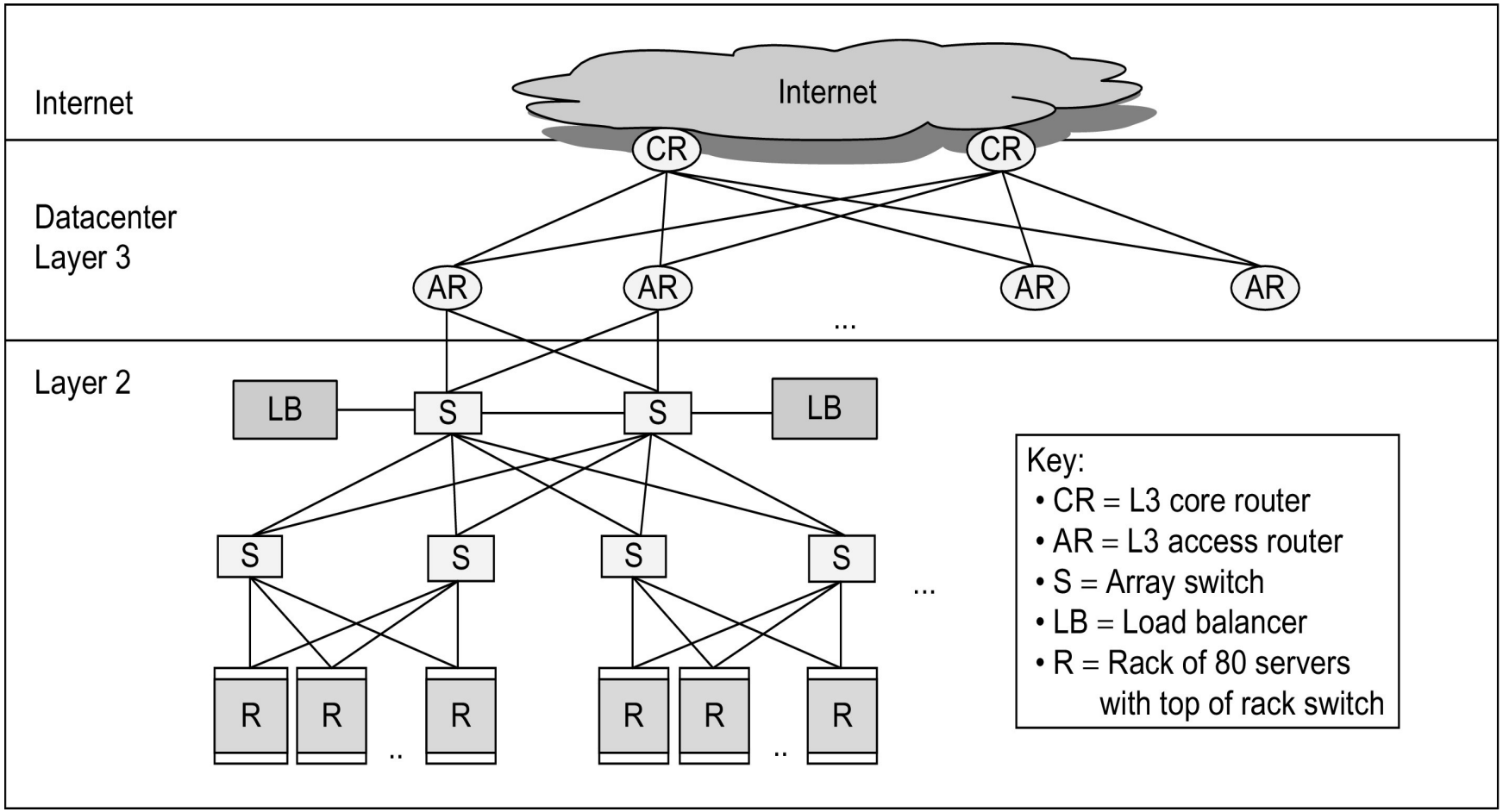


Figure 6.8 A Layer 3 network used to link arrays together and to the Internet (Greenberg et al., 2009). A load balancer monitors how busy a set of servers is and directs traffic to the less loaded ones to try to keep the servers approximately equally utilized. Another option is to use a separate *border router* to connect the Internet to the data center Layer 3 switches. As we will see in Section 6.6, many modern WSCs have abandoned the conventional layered networking stack of traditional switches.

Array Switch

- Switch that connects an array of racks
  - Array switch should have 10 X the bisection bandwidth of rack switch
  - Cost of *n*-port switch grows as n^2
  - Often utilize content-addressable memory chips and FPGAs

Newer Clos Network Structure

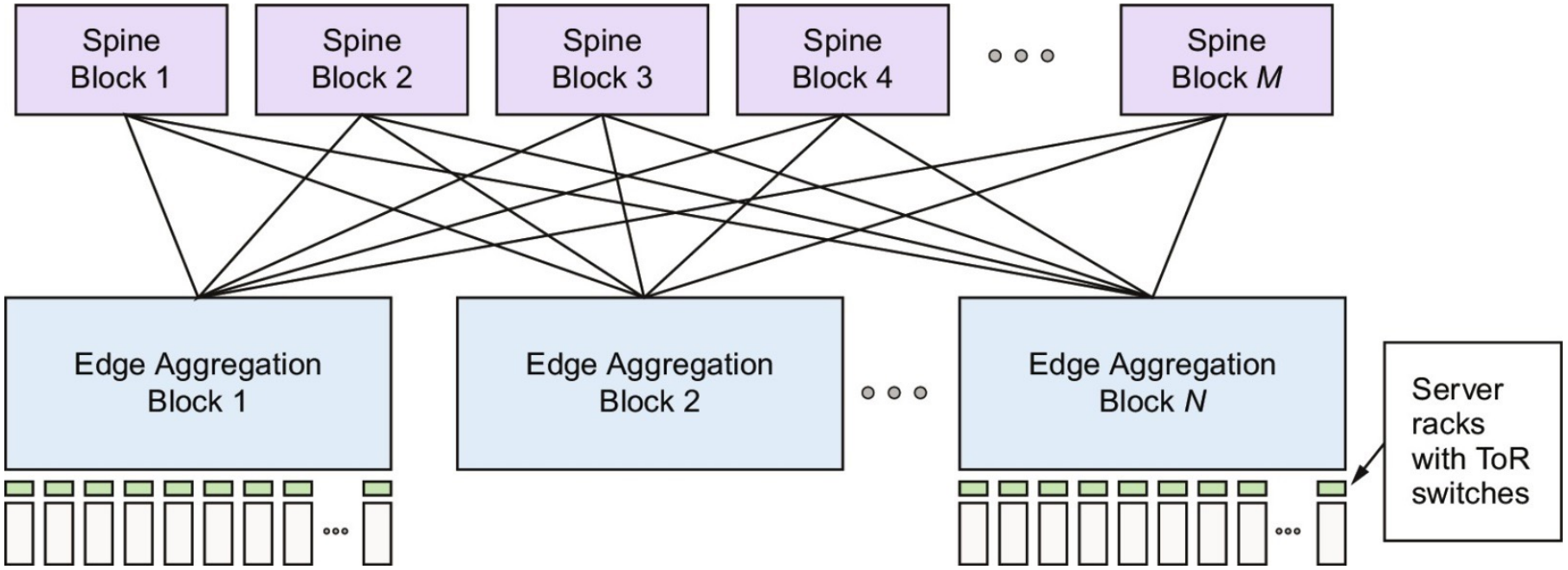


Figure 6.31 A Clos network has three logical stages containing crossbar switches: ingress, middle, and egress. Each input to the ingress stage can go through any of the middle stages to be routed to any output of the egress stage. In this figure, the middle stages are the *M* Spine Blocks, and the ingress and egress stages are in the *N* Edge Activation Blocks. Figure 6.22 shows the changes in the Spine Blocks and the Edge Aggregation Blocks over many generations of Clos networks in Google WSCs.

Google Jupiter Clos Network

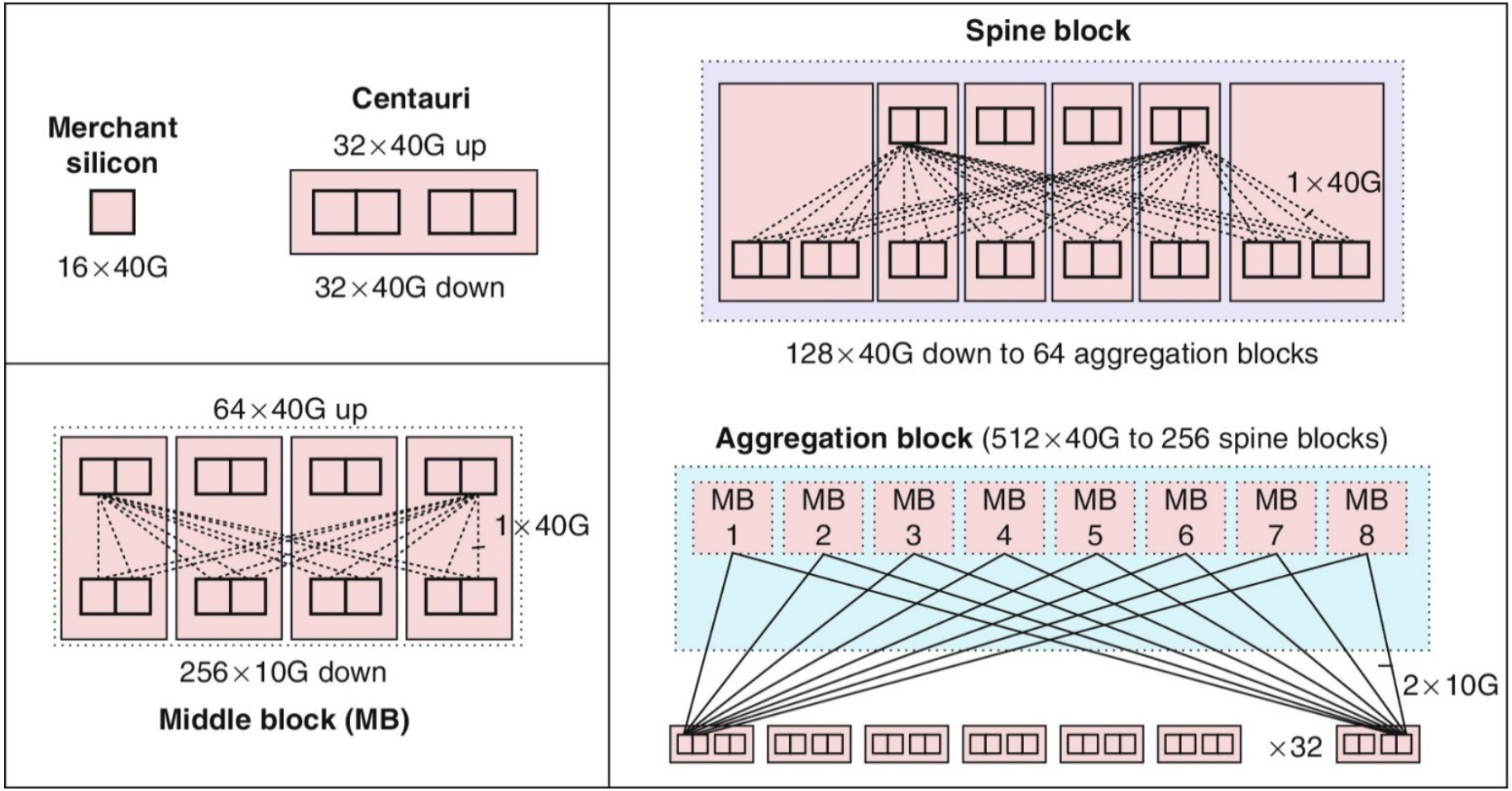


Figure 6.32 Building blocks of the Jupiter Clos network.

WSC Memory Hierarchy

- Servers can access DRAM and disks on other servers using a NUMA-style interface

	Local	Rack	Array
DRAM latency (μs)	0.1	300	500
Flash latency (μs)	100	400	600
Disk latency (μs)	10,000	11,000	12,000
DRAM bandwidth (MB/s)	20,000	100	10
Flash bandwidth (MB/s)	1000	100	10
Disk bandwidth (MB/s)	200	100	10
DRAM capacity (GB)	16	1024	31,200
Flash capacity (GB)	128	20,000	600,000
Disk capacity (GB)	2000	160,000	4,800,000

Storage options

- Use disks inside the servers, or
- Network attached storage through Infiniband

- WSCs generally rely on local disks
- Google File System (GFS) uses local disks and maintains at least three replicas

Cost of a WSC

- Capital expenditures (CAPEX)
  - Cost to build a WSC
  - $9 to 13/watt
- Operational expenditures (OPEX)
  - Cost to operate a WSC

Prgrm'g Models and Workloads

- Batch processing framework: MapReduce
  - **Map:** applies a programmer-supplied function to each logical input record
    - Runs on thousands of computers
    - Provides new set of key-value pairs as intermediate values
  - **Reduce:** collapses values using another programmer-supplied function

Prgrm'g Models and Workloads

- Example:
  - **map (String key, String value):**
    - **// key: document name**
    - **// value: document contents**
    - **for each word w in value**
      - **EmitIntermediate(w,"1"); // Produce list of all words**
  - **reduce (String key, Iterator values):**
    - **// key: a word**
    - **// value: a list of counts**
    - **int result = 0;**
    - **for each v in values:**
      - **result += ParseInt(v); // get integer from key-value pair**
    - **Emit(AsString(result));**

Prgrm'g Models and Workloads

- Availability:
  - Use replicas of data across different servers
  - Use relaxed consistency:
    - No need for all replicas to always agree
- File systems: GFS and Colossus
- Databases: Dynamo and BigTable

Prgrm'g Models and Workloads

- MapReduce runtime environment schedules map and reduce task to WSC nodes
  - Workload demands often vary considerably
  - Scheduler assigns tasks based on completion of prior tasks
  - Tail latency/execution time variability: single slow task can hold up large MapReduce job
  - Runtime libraries replicate tasks near end of job
    - Don't wait for stragglers, repeat task somewhere else