What is Cloud Computing?

Cloud computing is a model for enabling ubiquitous, convenient, **on-demand** network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1].

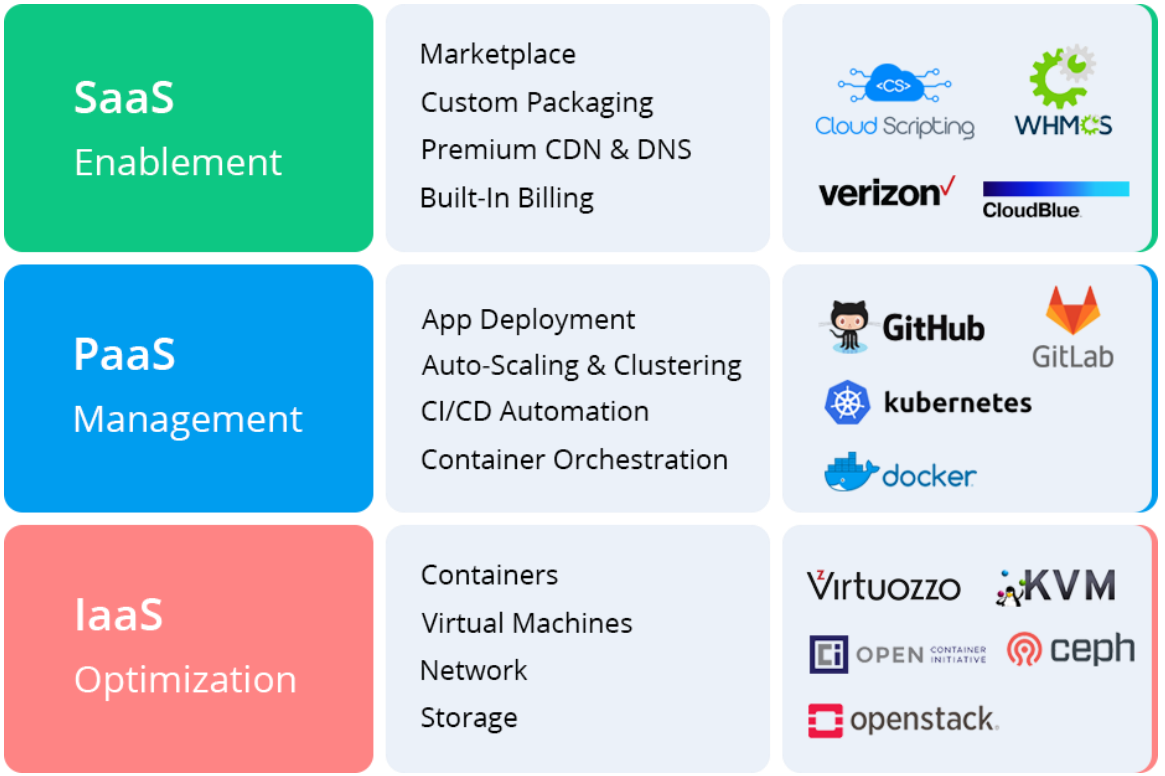
Informal: Cloud Computing = computing with large datacenters.

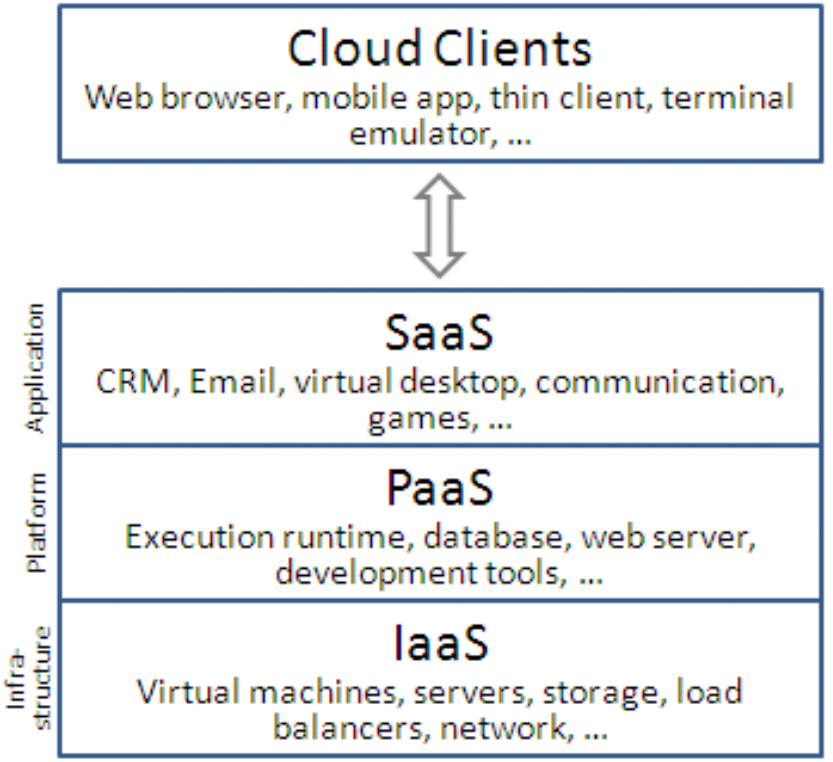
Our focus: Cloud Computing = computing as a utility outsourced to a third party or internal org

[1] Peter Mell (NIST), Tim Grance (NIST),
The National Institute of Standards and Technology (NIST) Definition of Cloud Computing
https://csrc.nist.gov/publications/detail/sp/800-145/final

Service Models

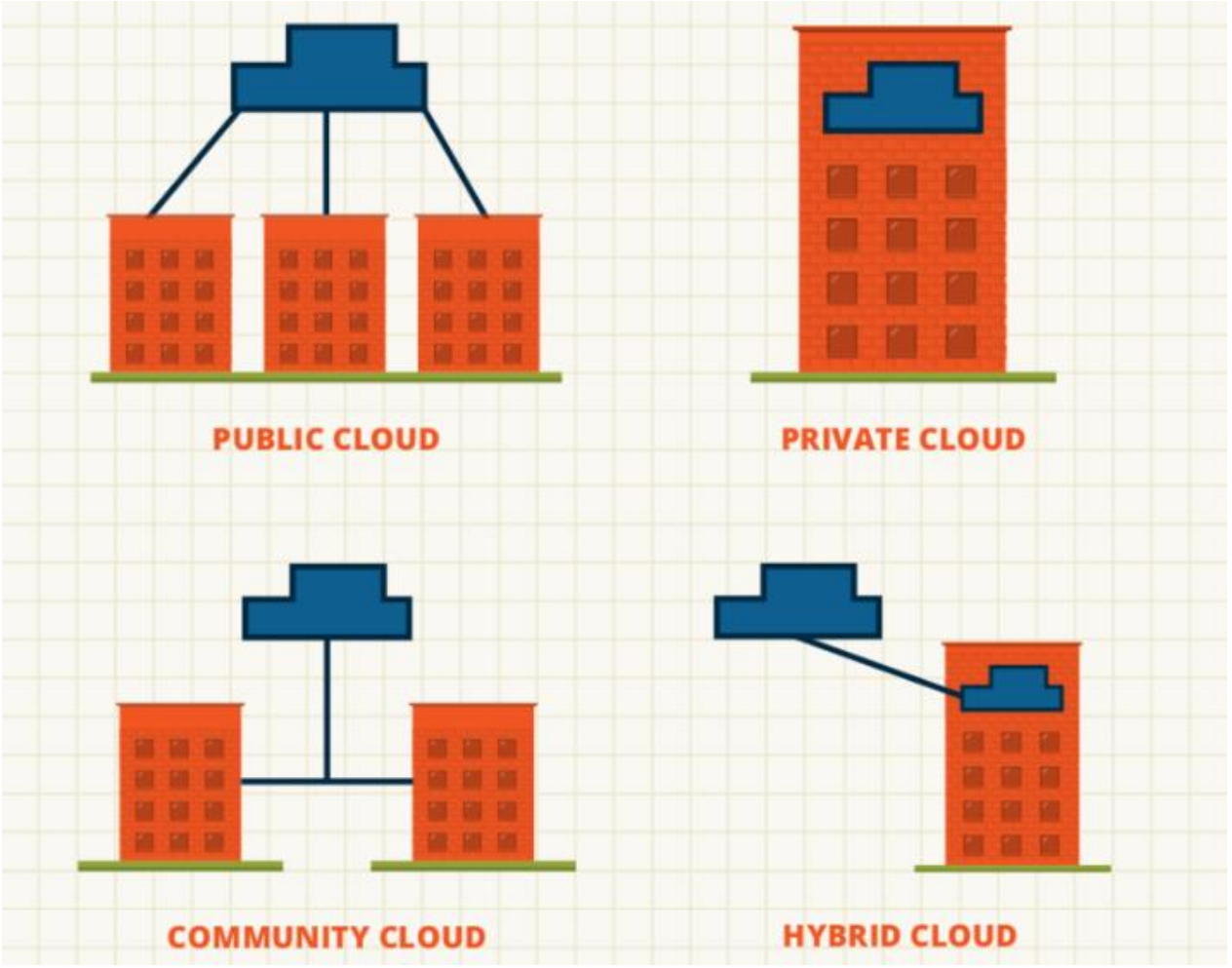
Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS)





Deployment Models

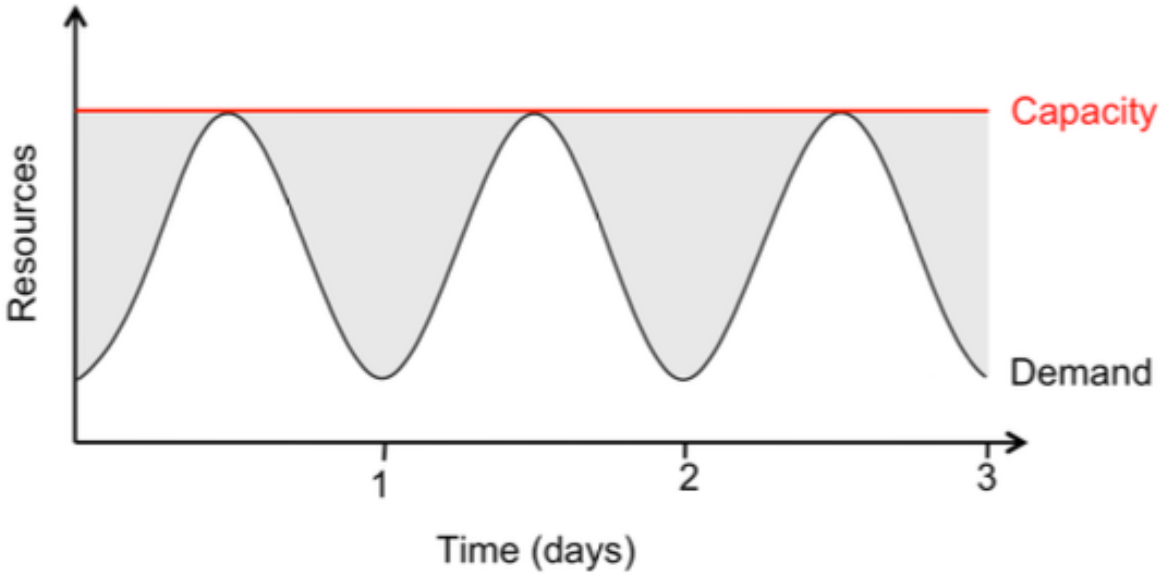
Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS)



Cloud Economics: For Users

Pay-as-you-go (usage-based) pricing:

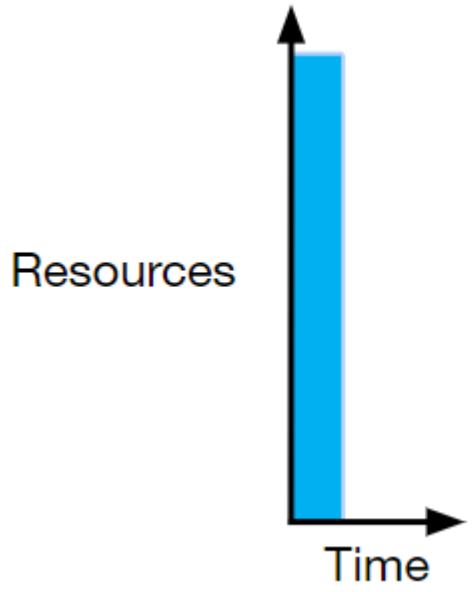
- Most services charge per minute, per byte, etc
- No minimum or up-front fee
- Helpful when apps have variable utilization

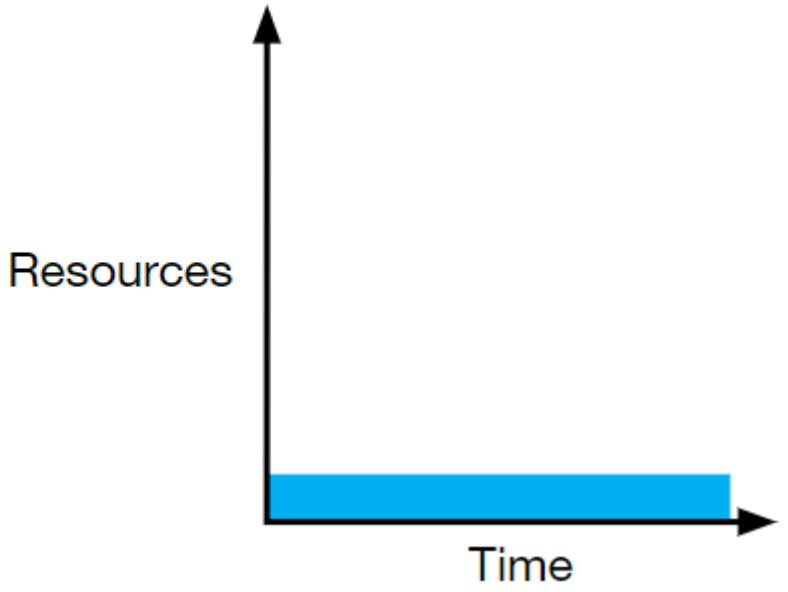


Cloud Economics: For Users

Elasticity:

- Using 1000 servers for 1 hour costs the same as 1 server for 1000 hours
- Same price to get a result faster!





Cloud Economics: For Providers

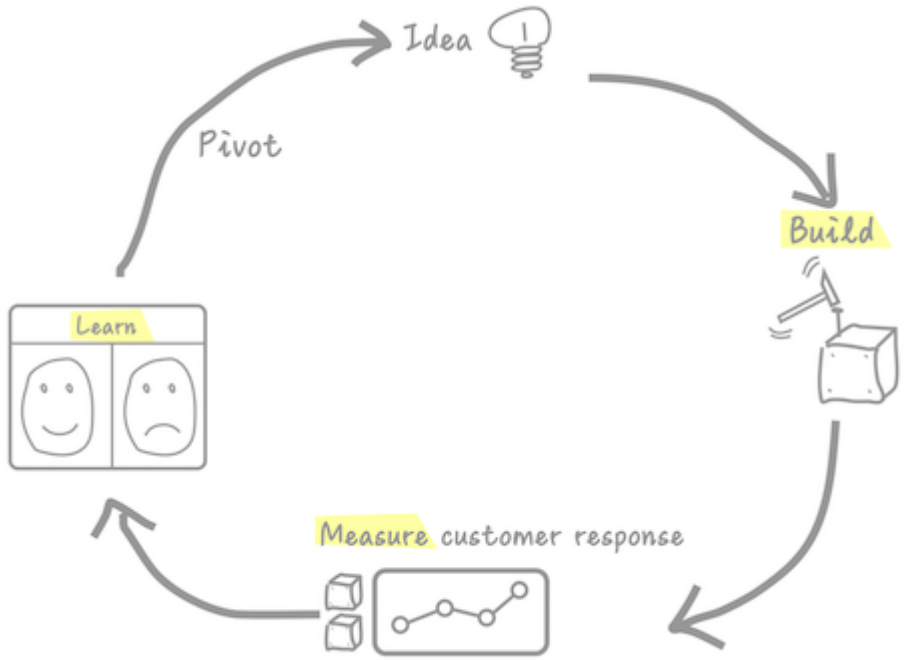
Economies of scale:

- Purchasing, powering & managing machines at scale gives lower per-unit costs than customers'
- To find Tradeoff: fast growth vs efficiency
- To find Tradeoff: flexibility vs cost

Cloud Economics: For Providers

Speed of iteration:

- Software as a service means fast time-to-market, updates, and detailed monitoring/feedback
- Compare to speed of iteration with ordinary software distribution



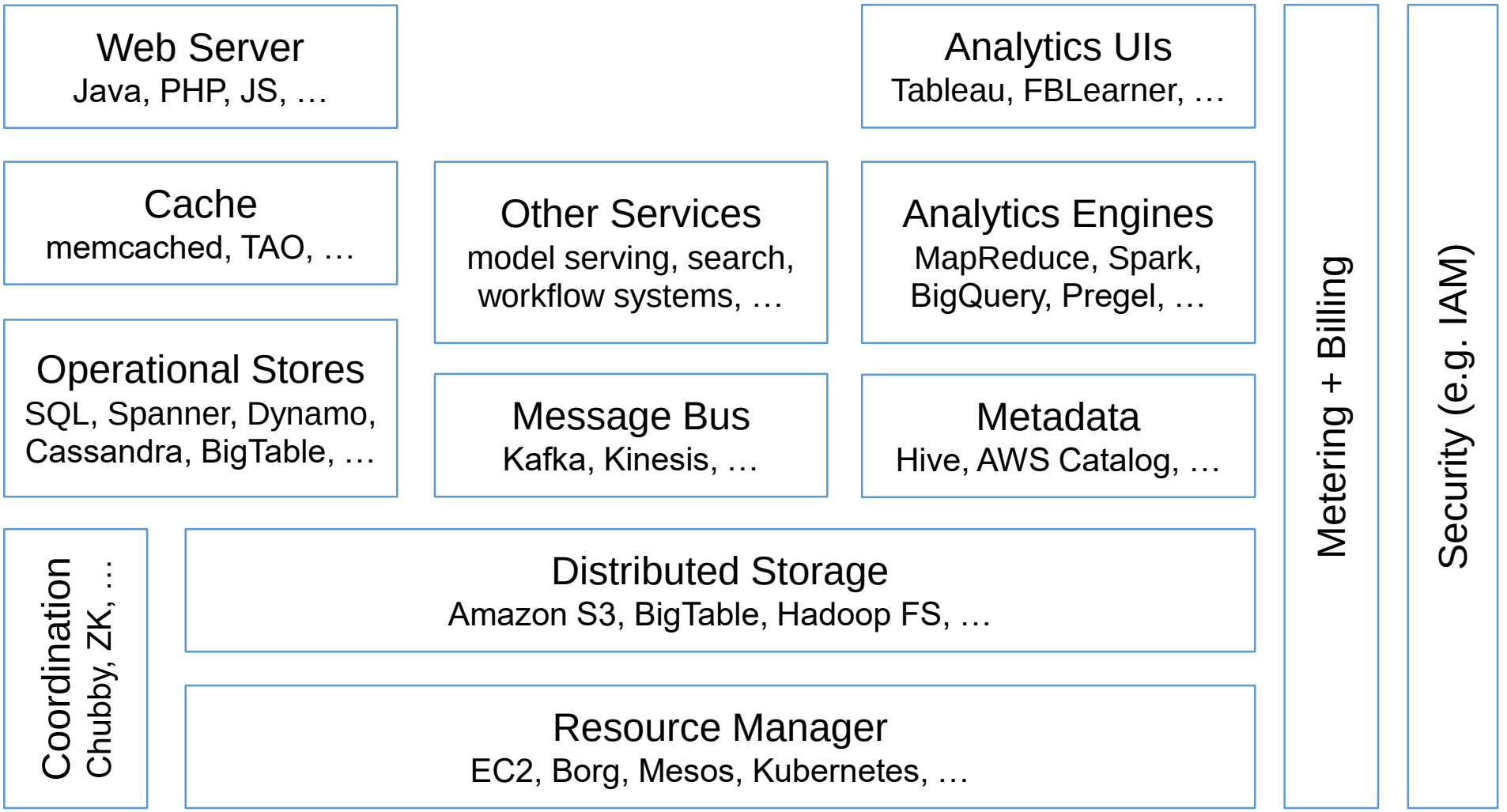
Other Interesting Features

- Spot market for preemptible machines, the ability to purchase a productive old server
- Wide geographic access for disaster recovery and speed of access
- Ability to quickly try exotic hardware
- Ability to A/B test anything

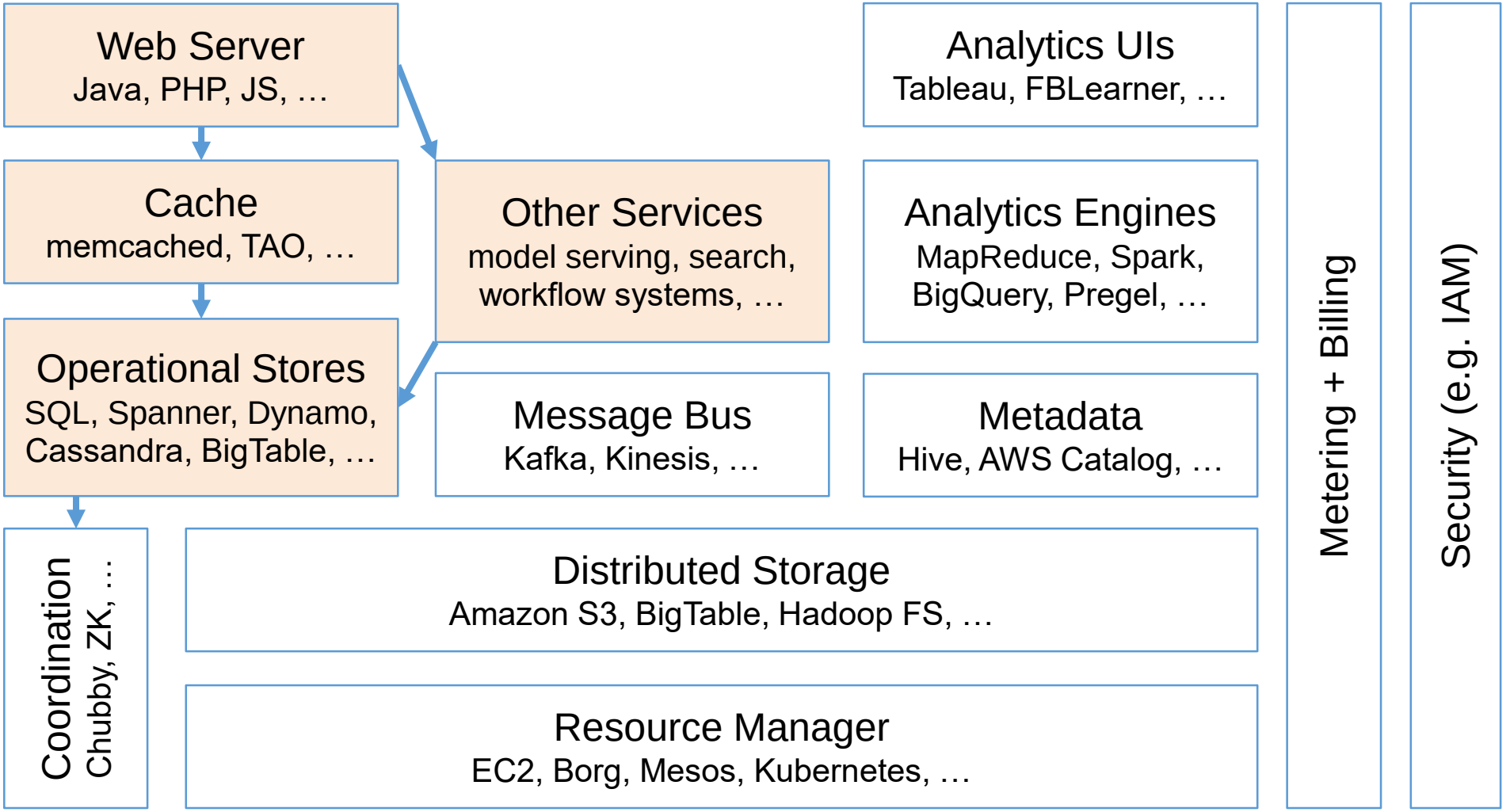
Common Cloud Applications

- Web and mobile applications
- Data analytics (MapReduce, SQL, ML, etc)
- Stream processing
- Batch computation (HPC, video, etc)

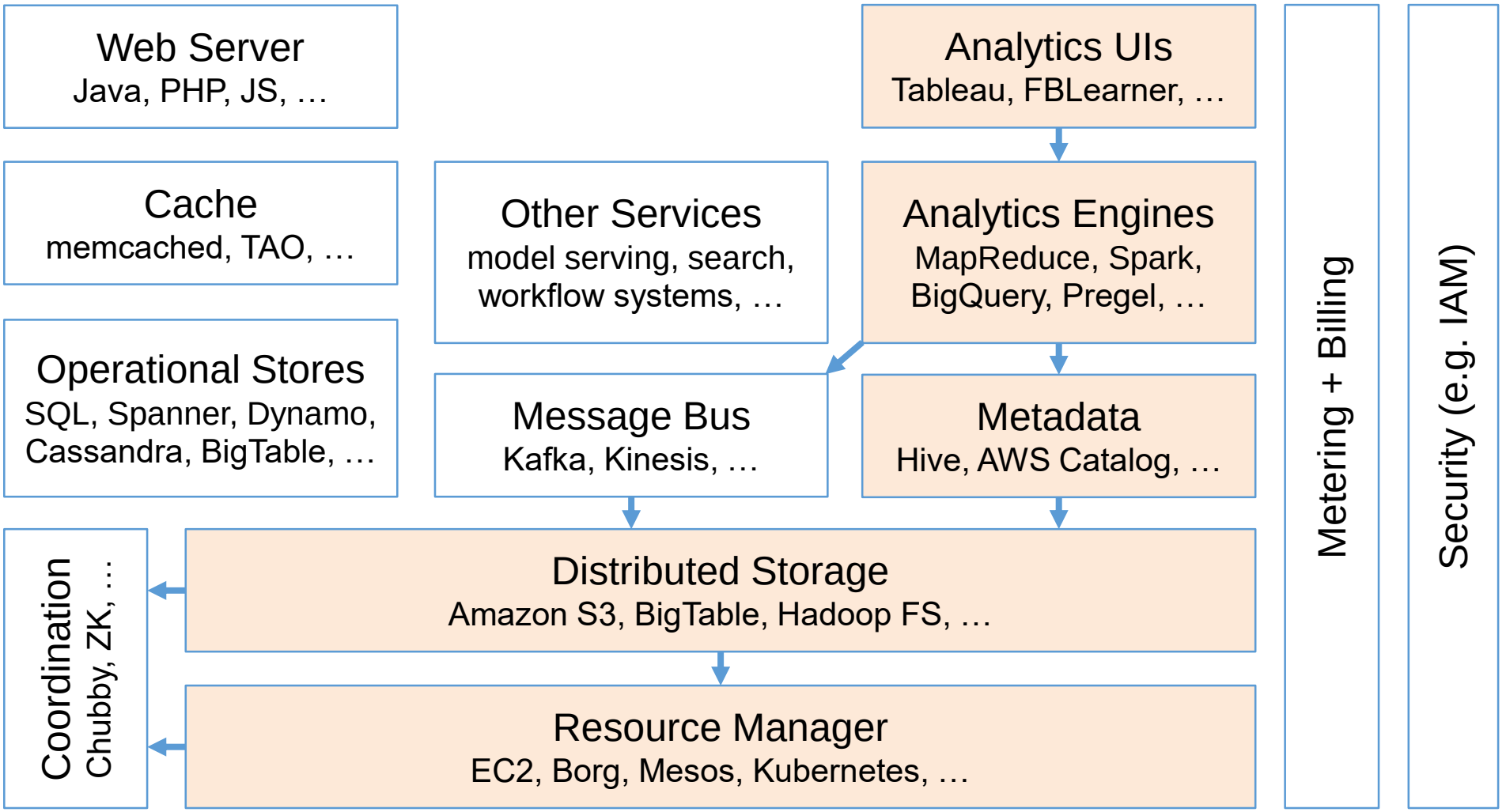
Cloud Software Stack



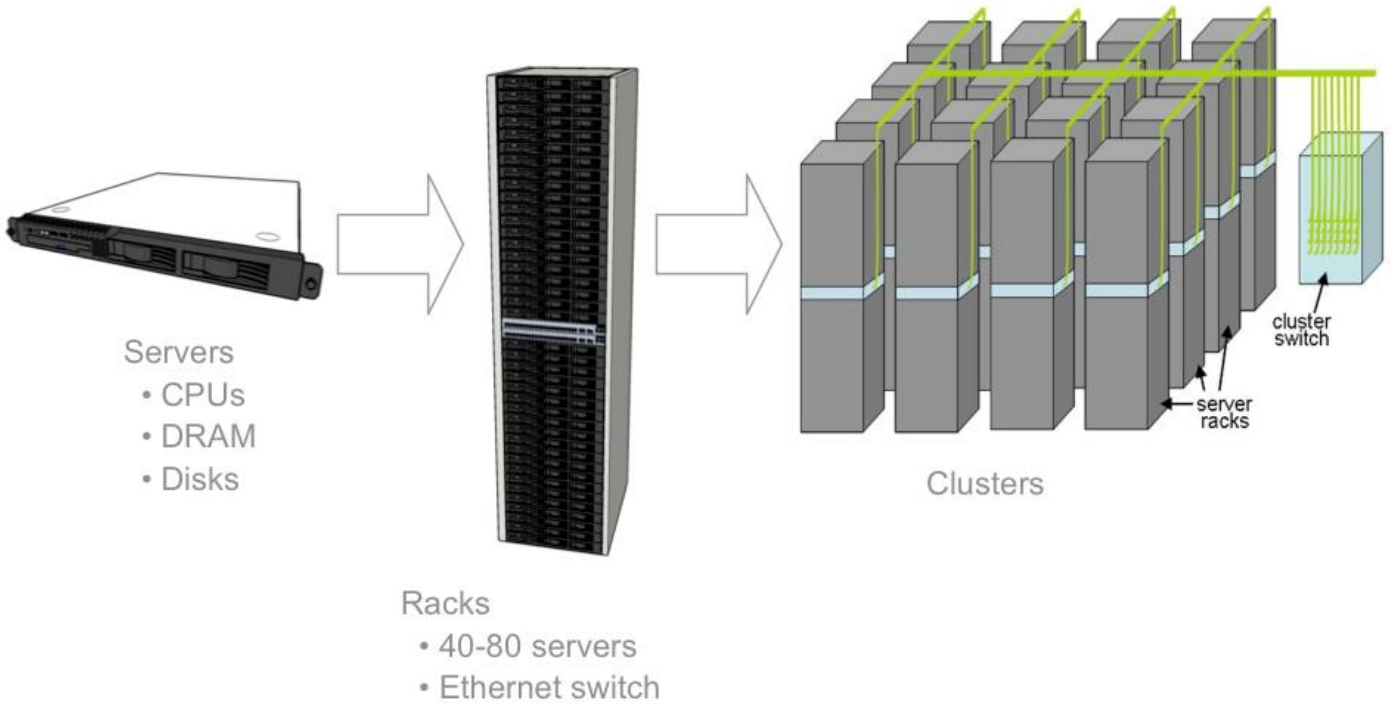
Example: Web Application



Example: Analytics Warehouse



Datacenter Hardware

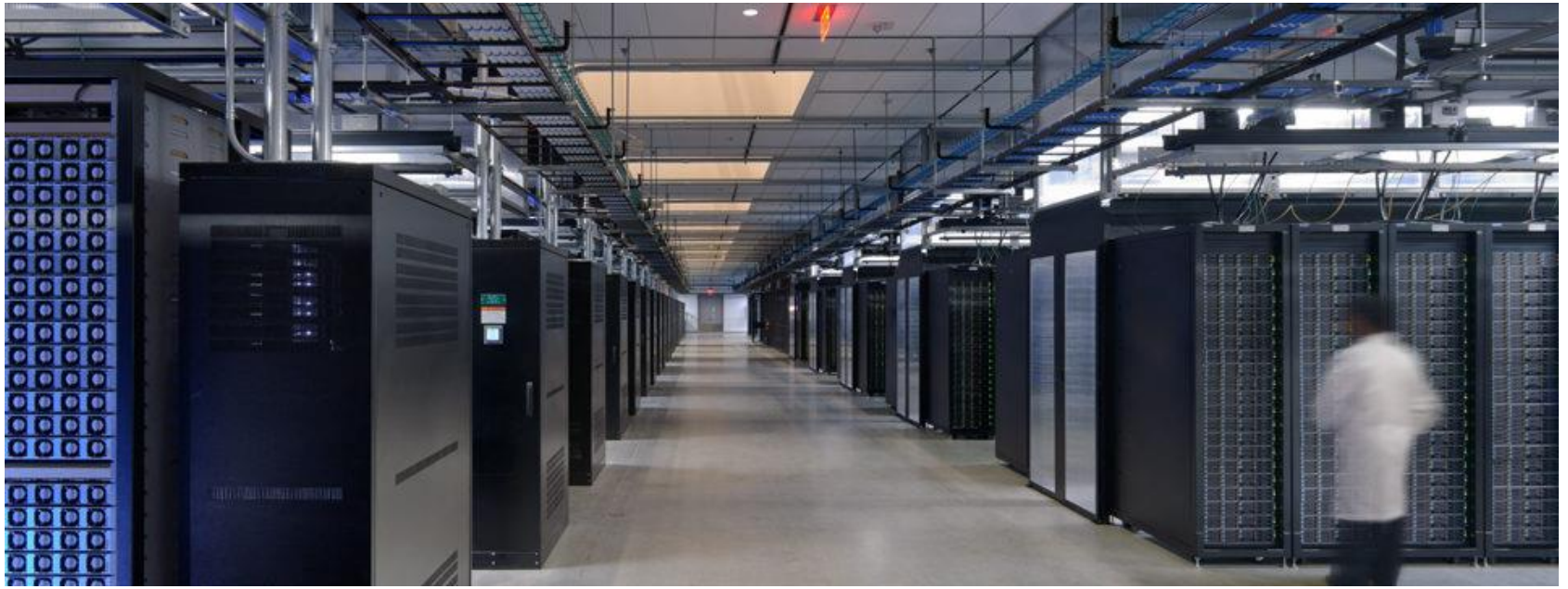


Rows of rack-mounted servers

Datacenter: 50 – 200K of servers, 10 – 100MW

- Often organized as few and mostly independent clusters

Datacenter Example

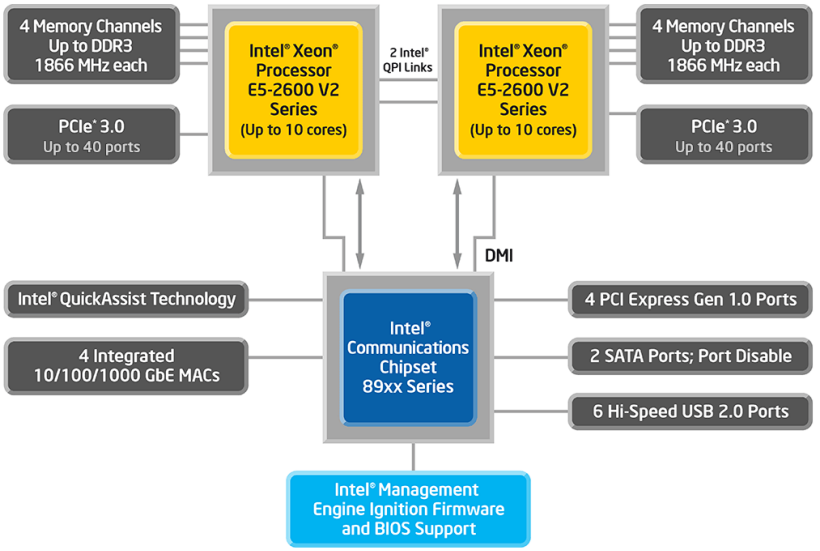


Datacenter HW: Compute

The basics

- Multi-core CPU servers
- 1 & 2 sockets

2-socket server



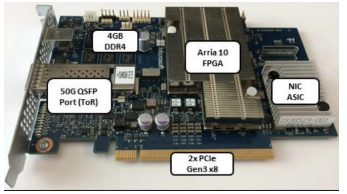
What's new

- GPUs
- FPGAs
- Custom accelerators (AI)

Google TPU²

Nvidia Tesla

Microsoft Catapult



Hardware Heterogeneity

Standard Systems	I Web	III Database	IV Hadoop	V Haystack	VI Feed
CPU	High 2 x E5-2670	High 2 x E5-2660	High 2 x E5-2660	Low 1 x E5-2660	High 2 x E5-2660
Memory	Low 16GB	High 144GB	Medium 64GB	Med-Hi 96GB	High 144GB
Disk	Low 250GB	High IOPS 3.2 TB Flash	High 15 x 4TB SATA	High 30 x 4TB SATA	Medium 2TB SATA + 1.6TB Flash
Services	Web, Chat	Database	Hadoop	Photos, Video	Multifeed, Search, Ads

[Facebook server configurations]

Custom-design servers

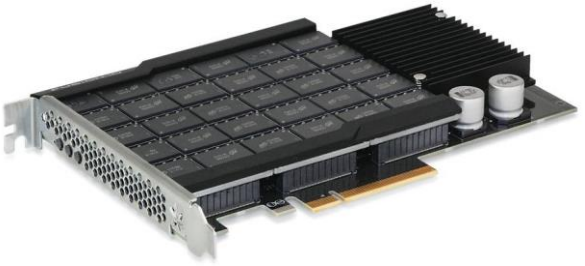
- Configurations optimized for major app classes
- Few configurations to allow reuse across many apps
- Roughly constant power budget per volume

Datacenter HW: Storage

The basics

- Disk trays
- SSD & NVM Flash

NVMe Flash

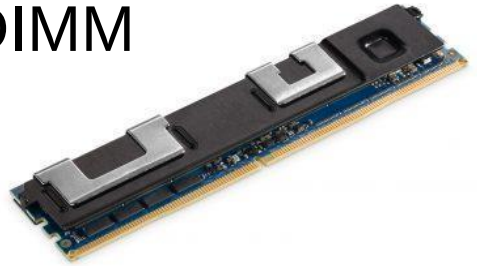


JBOD disk array

What's new

- Non-volatile memories
- New archival storage (e.g., glass)

NVM DIMM



Distributed with compute or NAS systems

- Remote storage access for many use cases (why?)

Datacenter HW: Networking

The basics

- 10, 25, and 40GbE NICs
- 40 to 100GbE switches
- Clos topologies

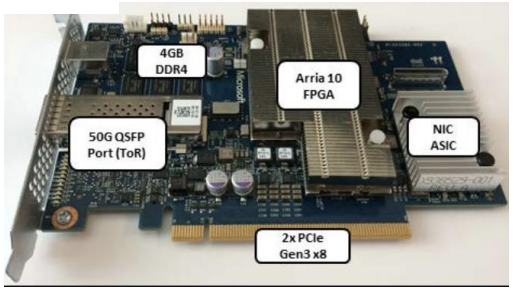
40GbE Switch

What's new

- Software defined networking
- Smart NICs
- FPGAs

Smart NIC

Microsoft Catapult



Performance Metrics

- Throughput
  - Requests per second
  - Concurrent users
  - Gbytes/sec processed
  - ...
- Latency
  - Execution time
  - Per request latency

Useful Throughput Numbers

DDR4 channel bandwidth (DDR4-2400 PC4-19200)	~20 GB/sec
PCIe gen3 (2010) x16 channel	15.8 GB/sec
PCIe gen6 (2021) x16 channel	126.03 GB/sec
NVMe Flash bandwidth	~32 GB/sec
GbE link bandwidth	10 – 100 Gbps
HDD (6Gb/s SAS)	230 MB/s
SSD (6Gb/s SAS)	550 MB/s
6 Gb/s SAS x4	2200 MB/s
12 Gb/s SAS x4	4400 MB/s

Useful Latency Numbers

Initial list from Jeff Dean, Google

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L3 cache reference	20 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Snappy	3,000 ns
Send 2K bytes over 10Ge	2,000 ns
Read 1 MB sequentially from memory	100,000 ns
Read 4KB from NVMe Flash	50,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA → Europe → CA	150,000,000 ns

Total Cost of Ownership (TCO)

TCO = capital (CapEx) + operational (OpEx) expenses

Operators perspective

CapEx: building, generators, A/C, compute/storage/net HW

Including spare parts, amortized over 3 – 15 years

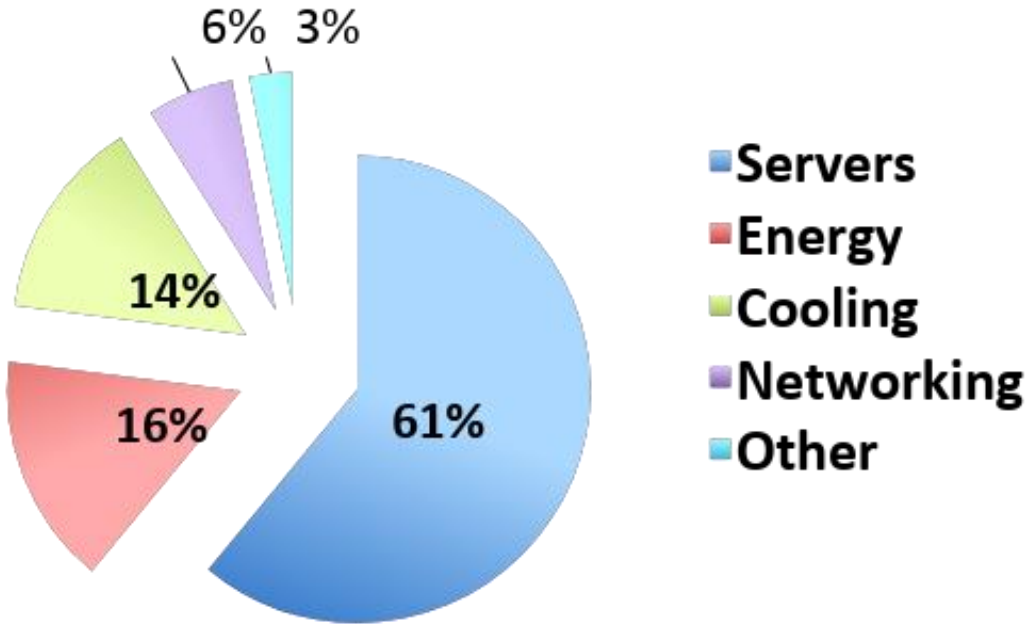
OpEx: electricity (5-7cents/KWh), repairs, people, WAN, insurance, …

Users perspective

CapEx: cost of long term leases on HW and services

OpeEx: pay per use cost on HW and services, people

Operator's TCO Example



[Source: James Hamilton]

Hardware dominates TCO, so we have to make it cheap

Must use it as well as possible

Reliability

Failure in time (FIT)

Failures per billion hours of operation = 10^9/MTTF

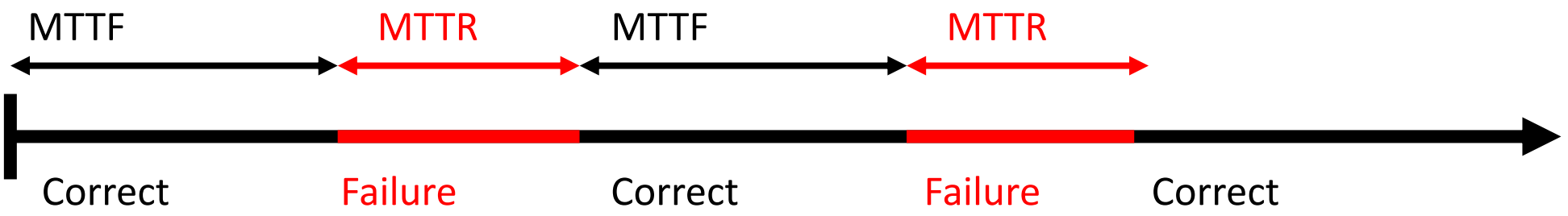
Mean time to failure (MTTF)

Time to produce first incorrect output

Mean time to repair (MTTR)

Time to detect and repair a failure

Availability



Steady state availability = MTTF / (MTTF + MTTR)

Yearly Datacenter Flakiness

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hrs to come back)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hrs)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packet loss)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~12 router reloads (takes out DNS and external vIPs for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

~1000 individual machine failures (2-4% failure rate, machines crash at least twice)

~thousands of hard drive failures (1-5% of all disks will die)

Add to these SW bugs, config errors, human errors, …

Key Availability Techniques

Technique	Performance	Availability
Replication	✓	✓
Partitioning (db tables sharding)	✓	✓
Load-balancing	✓	
Watchdog timers (hang prevention)		✓
Integrity checks (check sum etc)		✓
Canaries (Canary release)		✓
Eventual consistency (inconsistency tolerance)	✓	✓

Make apps do something reasonable when not all is right

- Better to give users limited functionality than an error page

Aggressive load balancing or request dropping

- Better to satisfy 80% of the users rather than none

The CAP Theorem

In distributed systems, choose 2 out of 3

Consistency

Every read returns data from most recent write

Availability

Every request executes & receives a (non-error) response

Partition-tolerance

The system continues to function when network partitions occur (messages dropped or delayed)

Useful Tips

Check for single points of failure

Keep it simple stupid (KISS)

- The reason many systems use centralized control

If it’s not tested, do no rely on it

Question: how do you test availability techniques with hundreds of loosely coupled services running on thousands of machines?