COMS E6998
Cloud Computing

Fall 2024, Friday 10:10am-12pm, 411 Kent

cloud-cu.github.io

Who are we?

Kostis Kaffes kkaffes@cs.columbia.edu

New faculty in the CS department working on systems and cloud computing! Before: NTUA, Stanford, Google

Vahab Jabrayilov vj2267@columbia.edu

2nd-year PhD student in the CS department working with Kostis

Who are you?

Class Format

Lectures will be paper discussions

– Read the papers ahead of time and submit short 1-paragraph summaries (for each paper) in Canvas

– 2 students present the papers & lead the discussion

– **We all participate in the discussion**

Grading

Project: 50%

Participation: 25%

Paper summaries/Discussion lead: 25%

What to understand in a paper?

Challenge addressed by the paper

The key insight

Strengths and weaknesses

Future work: Potential extensions and improvements

How to read the paper?

Read the abstract and introduction first

Read the rest of the paper twice

Use the motivation to understand the challenge

Keep notes/questions on the margins

Seek related work that is important or missing

How to lead the discussion?

Start with a 20-25 min. summary using slides

Prepare questions to keep the discussion flowing

- Question assumptions made by the authors, links to other work seen in the class or elsewhere, alternative approaches, discuss whether the experimental methodology and evaluation is sound and sufficient

Moderate student-to-student discussion

Research Project

Groups of 2-3 students

Topic

- Address an open question in cloud computing (extend, build, benchmark)
- Proposed by us or your own idea

Guidelines

- Select something that YOU find interesting
- Set realistic goals (semesters are short and you are busy…)
- Continuously revisit your approach and goals
- Leave time for the presentation and the report

Research Project Timeline

Form a group and submit a proposal (~ week 3)

Midterm review (~ week 7)

Presentation (last week of the quarter)

Paper due (after finals week)

Immediate TODOs

1. Fill in form with interests for discussion topics that is posted on the class website

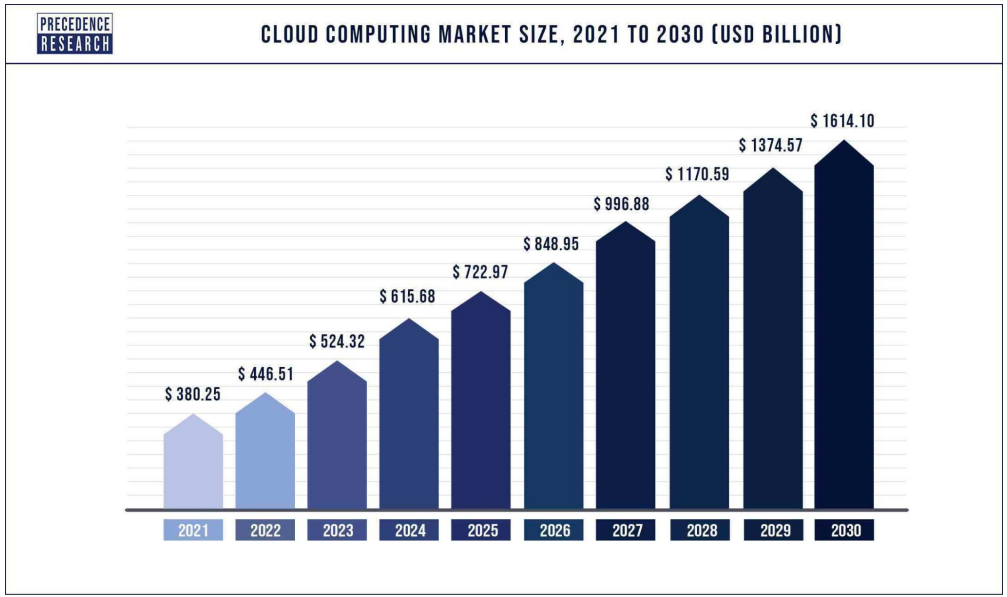
2. Volunteers for next week?

3. Start talking about projects and form groups (we will post list of projects next week)

Cloud 101

What is cloud computing?

Computing taking place in large data centers



What is a data center?

50-200K servers

10-100MW

High-speed network

Cooling systems

Power systems

Example: MS Quincy Datacenter

470k sq feet

(10 football fields)

40 MW power

(enough for 30,000)

Next to a hydro power plant

($0.02/kWh < $0.15/kWh)

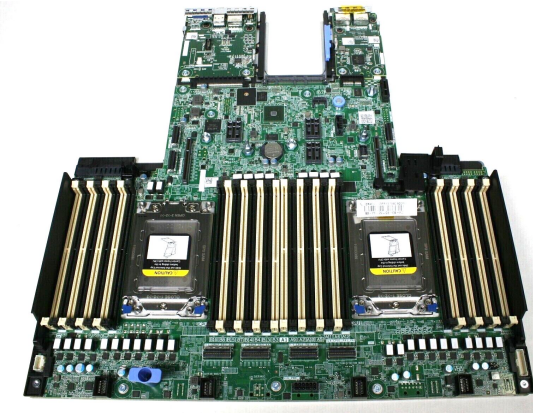
Data center HW: Servers

Usually dual socket

100s cores

100GBs of memory

Storage (SSD/HDs) local or remote

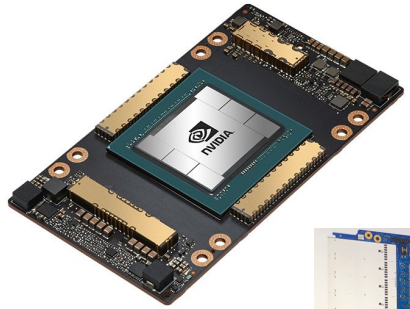


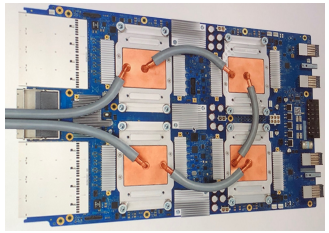
Data center HW: Accelerators

GPUs

Custom AI Accelerators

FPGAs

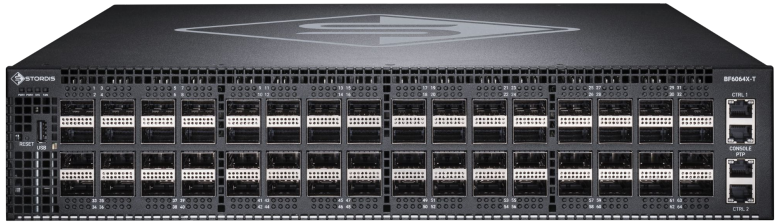




Data center HW: Networking

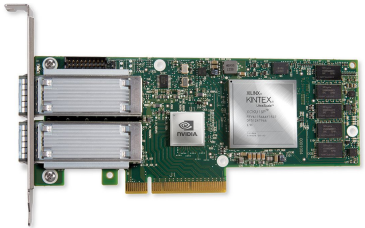
100-200 Gbps switches

Potentially programmable



40,100,200 Gbps NICs

SmartNICs/DPUs



What is cloud computing?

~~Computing taking place in large data centers~~

Our focus: **computing as a utility**

Outsourced to a third party or internal org that manages these data centers

Cloud Services

Infrastructure-as-a-Service (IaaS): **VMs, storage**

Platform-as-a-Service (PaaS): **K8s, Heroku, LLM Serving**

Software-as-a-Service (SaaS): **Email, Github**

Private vs. Public Cloud:

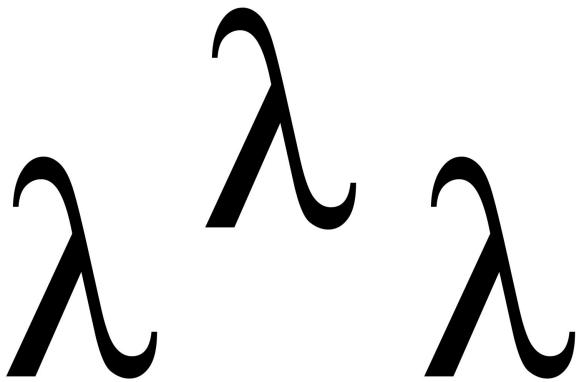
Shared across arbitrary users vs. within an organization

Serverless: Function-as-a-Service

New paradigm between PaaS and SaaS

Users only specify code in a high level language

Providers manage everything (placement, scaling, runtime environment, etc.)



Question

Why use and offer cloud computing services?

Users: Money, money, money

Pay as you go

– Services charge per minute/byte/etc.

– No minimum/upfront fees

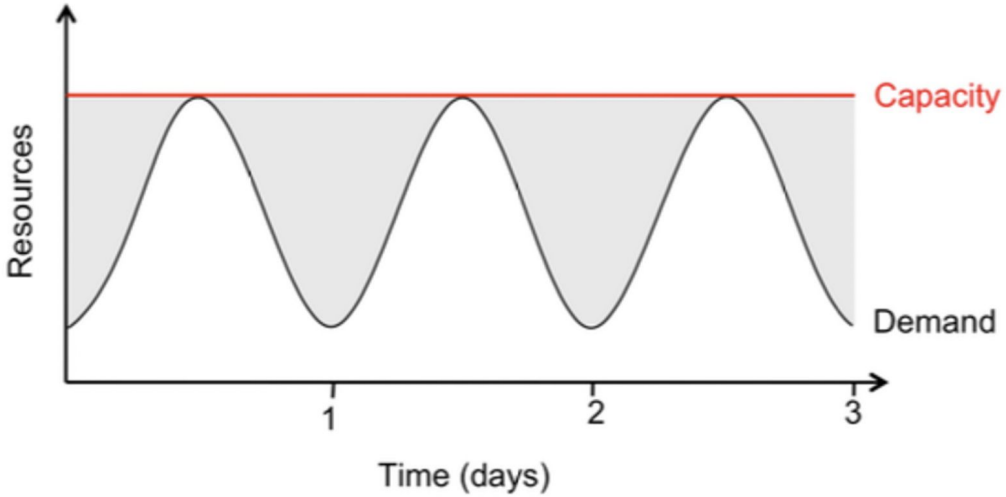
Question

When is pay-as-you-go pricing helpful?

Question

When is pay-as-you-go pricing helpful?

When apps have variable utilization



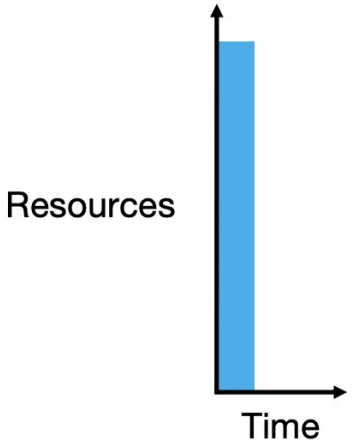
Users: Money **and performance**

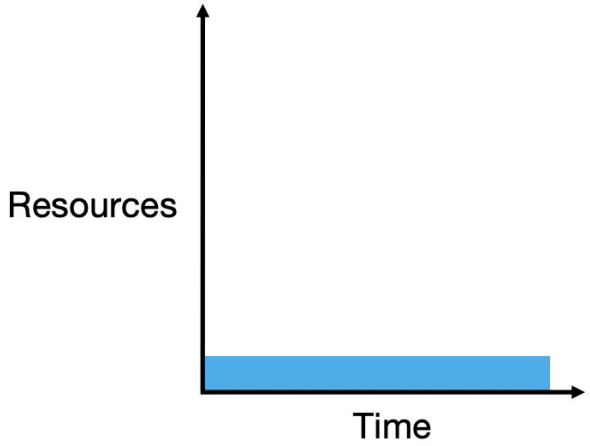
Elasticity

– 100 servers for 1h costs the same as 1 server for 100h

– Same price for

faster results





More user benefits

- Wide geographic access for disaster recovery and high availability
- Ability to try new/exotic hardware (?)
- Easy A/B testing

Question

When not to use the cloud?

Providers: Economy of Scale

– Scale lowers costs (hardware, power, cooling)

Providers: Speed of iteration

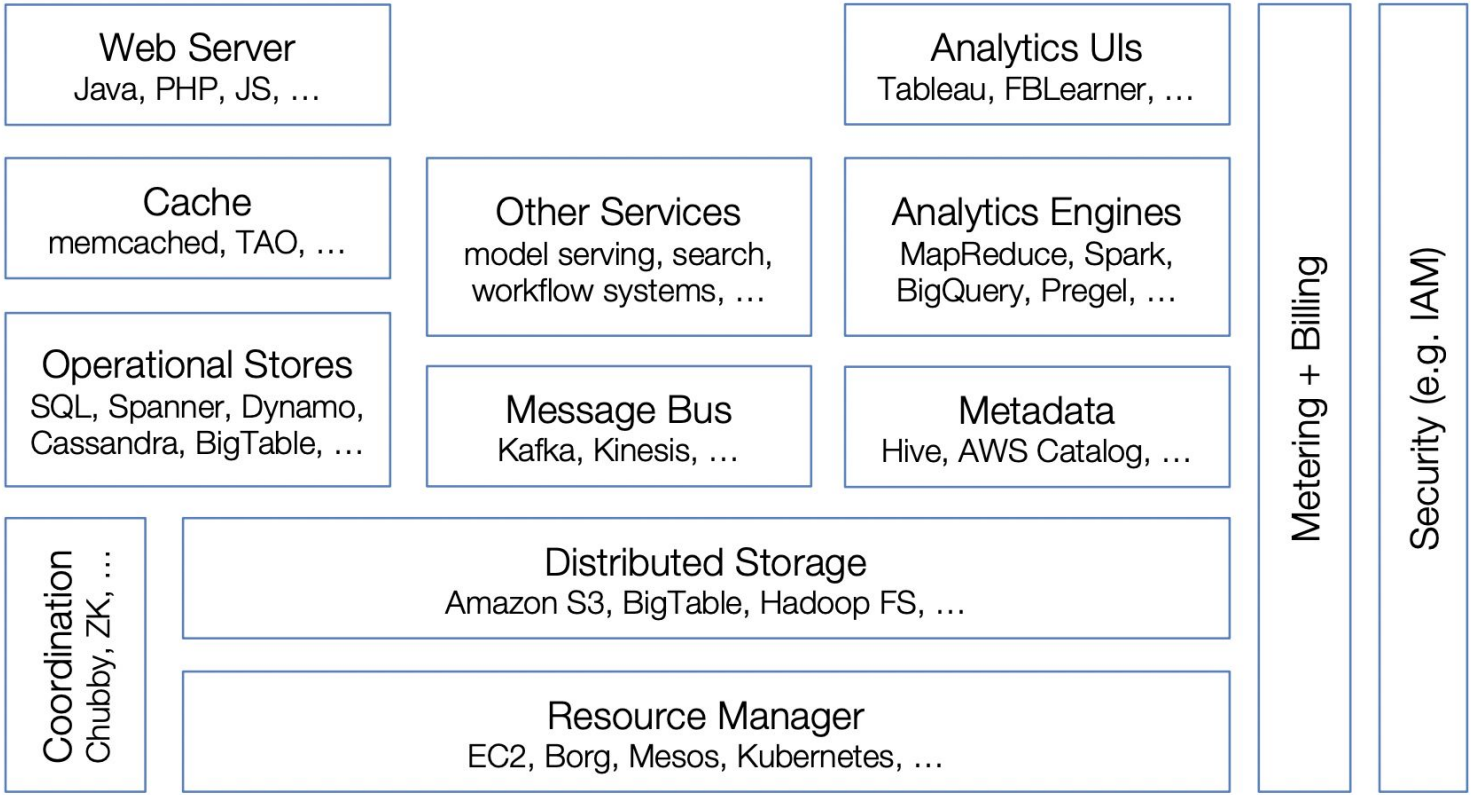
– Software as a service means fast time-to-market, updates, and detailed monitoring/feedback

– Compare to speed of iteration with ordinary software distribution

Question

How can providers avoid having usage spike at the same time?

Cloud Software Stack



Cloud Applications I

- Batch applications
  - Analytics like Spark, ML Training
  - E.g., create Netflix recommendations, search index, ...
  - Typically operate in the background (not interactive)
  - They care about throughput

Cloud Applications II

- Interactive Services
  - Search, mail, ML inference, messaging
  - Typically multi-tiered
  - They care about (tail) latency
- Customer VMs or Containers
  - E.g., Amazon EC2, Azure, etc.

Example: Web Search

- 1st Tier: Frontend

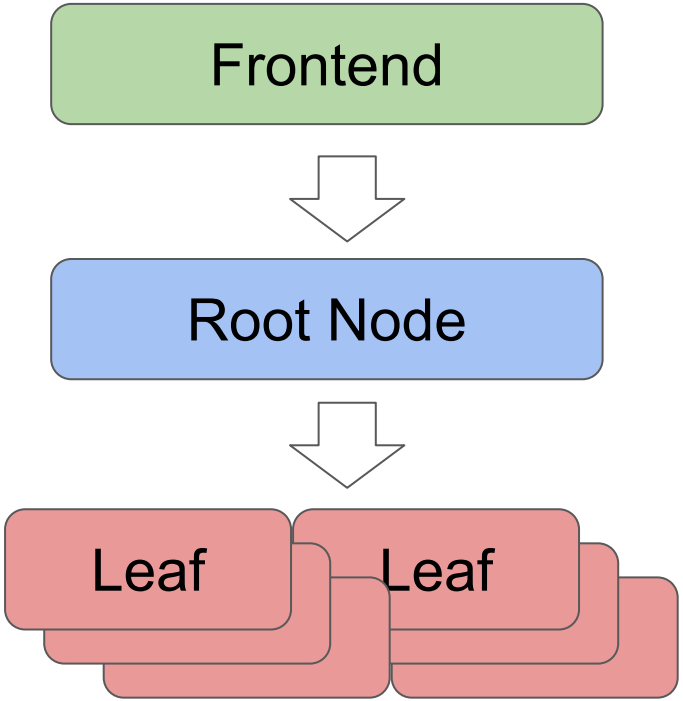
  HTTP Processing

- 2nd Tier: Logic

  Aggregation, Ranking, Caching

- 3rd Tier: Storage

  Index/document storage



Question

Why (not) multi-tiered services?

Performance Metrics

Throughput

- Requests Per Second (RPS)
- GBps processed

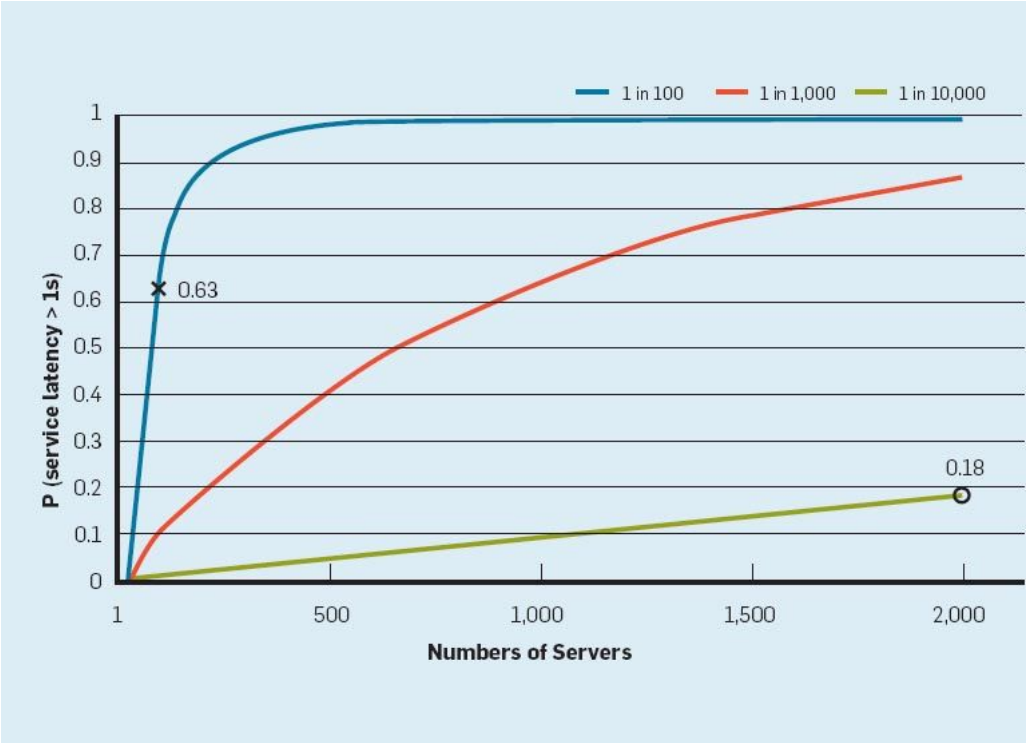
Latency

- End-to-end time experience by users
- Latency of individual requests
- Distribution matters (50%, 95%, 99% percentile) – Tail Latency

Question

Why do we care about the tail latency?

Tail at Scale



Probability of one-second service-level response time as the system scales and frequency of server-level high-latency outliers varies

Why server-level outliers exist?

Interference on shared resources

Spikes on input load

Queueing and head-of-line blocking

Maintenance/Background activities, e.g., garbage collection

Question

How to reduce this variability?

How to reduce variability?

Overprovisioning: allocate more resources than needed

Differentiate service classes

Reduce head-of-line blocking using time slicing

Defer events such as as log compaction or garbage collection

Synchronize background tasks across machines

Question

Variability always exists, how can we reduce tail latency?

How to reduce tail latency?

Replicate requests

Partition more

Detect “bad” servers

Question

What else do we care about in a cloud setting?

Reliability

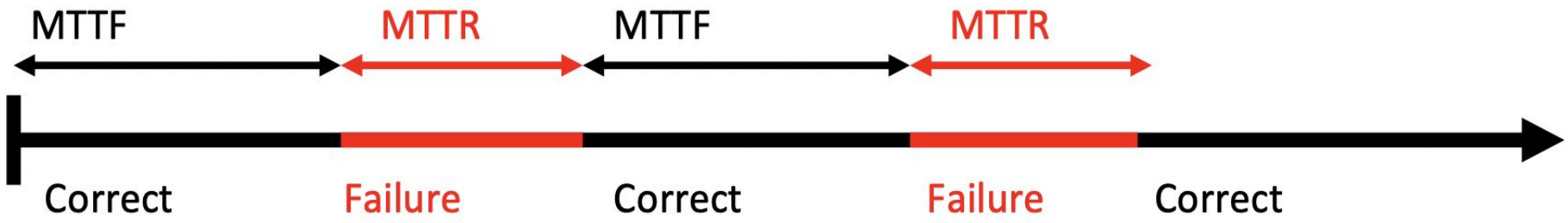
Failure in Time (FIT)

Failures per billion hours of operation

Mean Time to Failure (MTTF)

Mean Time to Repair (MTTR)

Availability



Average Availability = MTTF / (MTTF + MTTR)

Reliability Events

~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, ~50-100 machines powered down, ~6 hrs)

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

~20 rack failures (800-1600 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (200-400 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)

Question

What are some techniques that improve availability?

Availability Techniques

Replication

Partitioning

Watchdog Timers

Integrity Checks

Canaries

Eventual Consistency

Total Cost of Ownership (TCO)

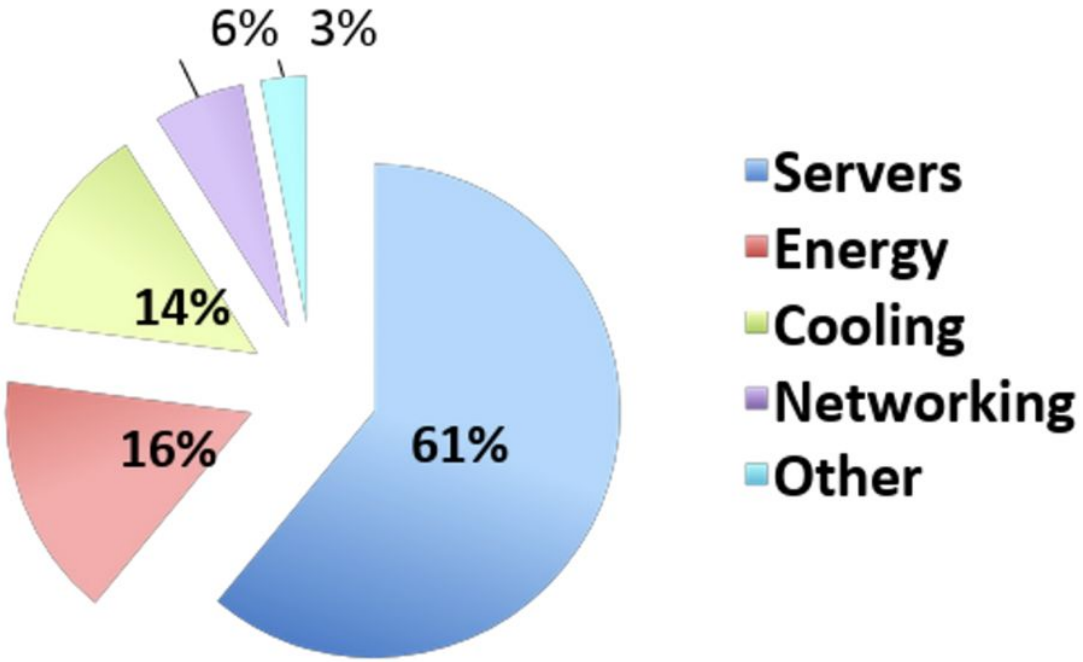
TCO = capital (**CapEx**) + operational (**OpEx**) expenses

CapEx: building, generators, A/C, compute/storage/net HW, amortized over a few years

OpEx: electricity (5-7c/KWh), repairs, people, WAN

TCO Breakdown

Hardware dominates



[Source: James Hamilton]

What’s (very) new?

ML and LLMs

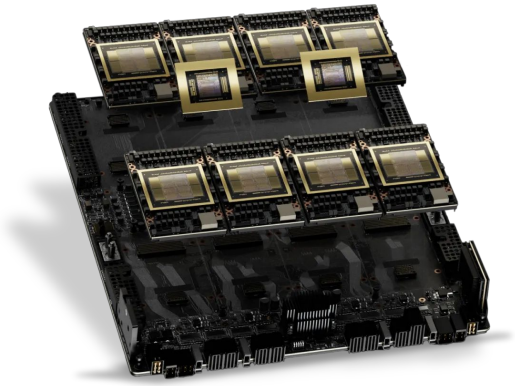
Or maybe not?

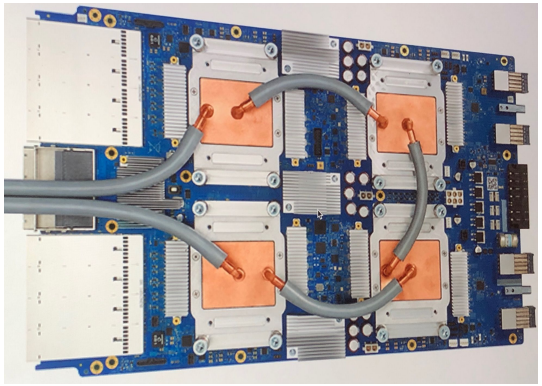
ML and LLMs workloads consist of:

1. Doing computations on some data
2. Moving data between servers

Or maybe yes?

But a lot of computation and a lot of data to move around:





But the goals are the same

Maximize accelerator and interconnect utilization

Minimize latency (defined differently!)

Maximize throughput

Minimize power consumption

System management of large accelerator clusters

Thank you!

(+syllabus)

Further optional reading: The datacenter as a computer