CS349D Cloud Computing

Christos Kozyrakis

Spring 2023, MW 1.30– 2.50pm, Hewlett 101

cs349d.stanford.edu
Class Staff

Christos Kozyrakis
www.stanford.edu/~kozyraki

Michael Abbott
www.linkedin.com/in/michaelabbott

Mark Zhao
www.stanford.edu/~myzhao

Franky Romero
www.stanford.edu/~faromero
Student Intro
Class Topics in 2018

- Storage
- CAP theorem
- Cloud economics
- Databases
- Analytics systems
- Stream processing
- Resource managers
- Resource allocation
- Serverless computing
- Monitoring & debugging
- Programming models
- ML platforms
- ML serving
- Hardware
- Security
- Privacy
- SRE / DevOps
- Edge computing
Class Topics in 2023

Cloud basics

Tools for building infrastructure [management, observability]

Data infrastructure [databases and lakehouses]

Infrastructure for ML [training, serving, MLops]

Security infrastructure [Nitro, data-driven security]

Confidential computing [private analytics and ML]
Class Format

Half lectures will be a guest lecture
   No video, come in person
   Participate in the discussion

Half lectures will be paper discussions
   Read the papers ahead of time and submit summaries
   2-3 students summarize papers & lead discussion
   We all participate in the discussion
   1 student takes notes
What to Look for in a Paper

The challenge addressed by the paper

The key insights & original contributions
Real or claimed, you have to check

Critique: the major strengths & weaknesses
Look at the claims and assumptions, the methodology, the analysis/evaluation, and the presentation style

Future work: extensions & improvements
Can we apply the methodology to other problems?
What are the broader implications?
Tips for Reading Papers

Read the abstract, intro, & conclusion first

Read the rest of the paper twice
  First a quick pass to get rough idea then a detailed reading

Underline/highlight the important parts of the paper

Keep notes on the margins about issues/questions
  Key insights, questionable claims, relevance to other topics, etc.

Look up references that seem to important or missing
  You may also want to check who references this paper and how
Tips for Leading Discussion

Keep paper summary to 5min
   Assume everyone has read it recently

Prepare a few questions to keep discussion going
   Questions on basics, dig further into techniques, alternative approaches, draw links to recent discussions, …
   Be open to questions from the rest of the class

Moderate discussion
Research Project

Groups of 2-3 students

Topic

  Address an open question in cloud computing
  Suggested by staff or your own idea

Timeline (TBD)

  Project proposal – around week 3
  Mid-term checkpoint – around week 6
  Presentation/paper – week 10
Grading

Project 65%

Participation 20%

Paper summaries/presentation: 15%
Reminders

Make sure you are registered on Axess and EdStem
   Contact instructors if you need help

Fill in form with interests for discussion topics
   We will assign topics for leads and note taking

Start talking about projects
   Form a group
Next Meeting: Cloud Basics

Goal: get us all on the same page

Read the two white papers from AWS

AWS Overview, Well architected Framework
No summaries needed
Come prepared to discuss the state of cloud
Cloud Computing Overview

Christos Kozyrakis

cs349d.stanford.edu
What is Cloud Computing?

Informal: computing with large datacenters
What is Cloud Computing?

Informal: computing with large datacenters

Our focus: computing as a utility
  » Outsourced to a third party or internal org
Types of Cloud Services

Infrastructure as a Service (IaaS): VMs, disks
Platform as a Service (PaaS): K8S, MapReduce
Software as a Service (SaaS): Email, GitHub

Public vs private clouds:

Shared across arbitrary orgs/customers vs internal to one organization
Example

AWS Lambda functions-as-a-service

- Runs functions in a Linux container on events
- Used for web apps, IoT apps, stream processing, highly parallel MapReduce and video encoding
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’
» Tradeoff: fast growth vs efficiency
» 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
Questions

• Assume you are a cloud provider

   How do you avoid having many of your customers spike at the same time?
Other Interesting Features

Spot market for preemptible machines

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

1. Web and mobile applications
2. Data analytics (MapReduce, SQL, ML, etc)
3. Stream processing
4. Batch computation (HPC, video, etc)
Cloud Software Stack

Web Server
Java, PHP, JS, …

Cache
memcached, TAO, …

Other Services
model serving, search, workflow systems, …

Analytics Engines
MapReduce, Spark, BigQuery, Pregel, …

Analytics UIs
Tableau, FBLearner, …

Operational Stores
SQL, Spanner, Dynamo, Cassandra, BigTable, …

Message Bus
Kafka, Kinesis, …

Metadata
Hive, AWS Catalog, …

Distributed Storage
Amazon S3, BigTable, Hadoop FS, …

Resource Manager
EC2, Borg, Mesos, Kubernetes, …

Coordination
Chubby, ZK, …

Resource Manager
EC2, Borg, Mesos, Kubernetes, …

Security (e.g. IAM)

Metering + Billing
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

What’s new
- GPUs
- Custom accelerators (AI)
- FPGAs
Datacenter HW: Storage

The basics
- Disk trays
- SSD & NVM Flash

What’s new
- Non-volatile memories
- New archival storage (e.g., glass)

Distributed with compute or NAS systems
- Remote storage access for many use cases (why?)
Datacenter HW: Networking

The basics
- 40, 100, 200 GbE NICs
- 100GbE to 200 GbE switches
- Clos topologies

What’s new
- Software defined networking
- In network computation
- Smart NICs
- FPGAs
Performance Metrics

Throughput
- Requests per second
- Concurrent users
- Gbytes/sec processed

Latency
- Execution time
- Per request latency
Tail Latency

The 95\textsuperscript{th} or 99\textsuperscript{th} percentile request latency
End-to-end with all tiers included

Larger scale $\rightarrow$ more prone to high tail latency

[Dean & Barroso,’13]
Total Cost of Ownership (TCO)

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

Operators perspective

CapEx: building, generators, A/C, compute/storage/net HW
Including spares, amortized over 3 – 15 years
OpEx: electricity (5-7c/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

Hardware dominates TCO, make it cheap
Must utilize it as well as possible

[Source: James Hamilton]
Questions

How can both providers and users benefit financially from cloud computing?

When should users consider hybrid or on-premise computing?
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)

MTTF  MTTR  MTTF  MTTR
Correct  Failure  Correct  Failure  Correct

Steady state availability  = MTTF / (MTTF + MTTR)
## Key Availability Techniques

<table>
<thead>
<tr>
<th>Technique</th>
<th>Performance</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replication</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Partitioning (sharding)</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Load-balancing</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Watchdog timers</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Integrity checks</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Canaries</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Eventual consistency</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

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?
Questions

Other major advantages or disadvantages of cloud computing?