CS 4530 & CS 5500 Software Engineering

Lecture 9.1: Why Engineer Distributed Software?

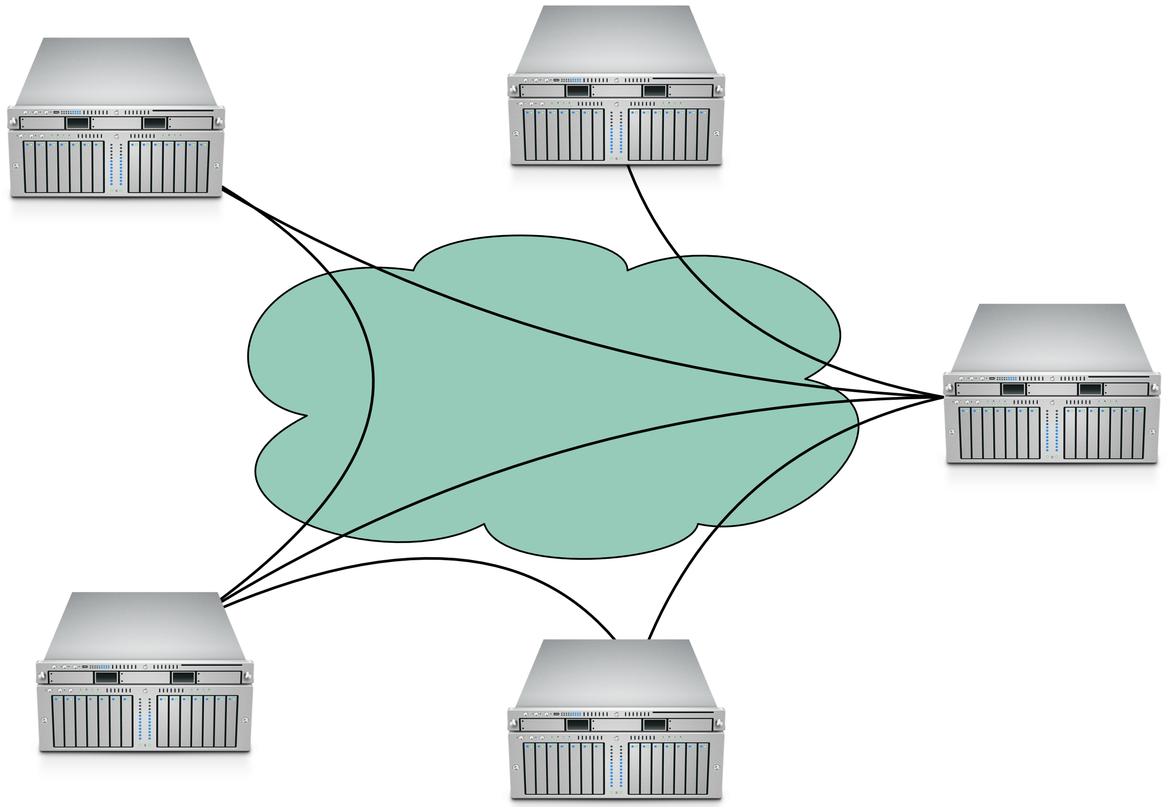
Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

Learning Objectives for this Lesson

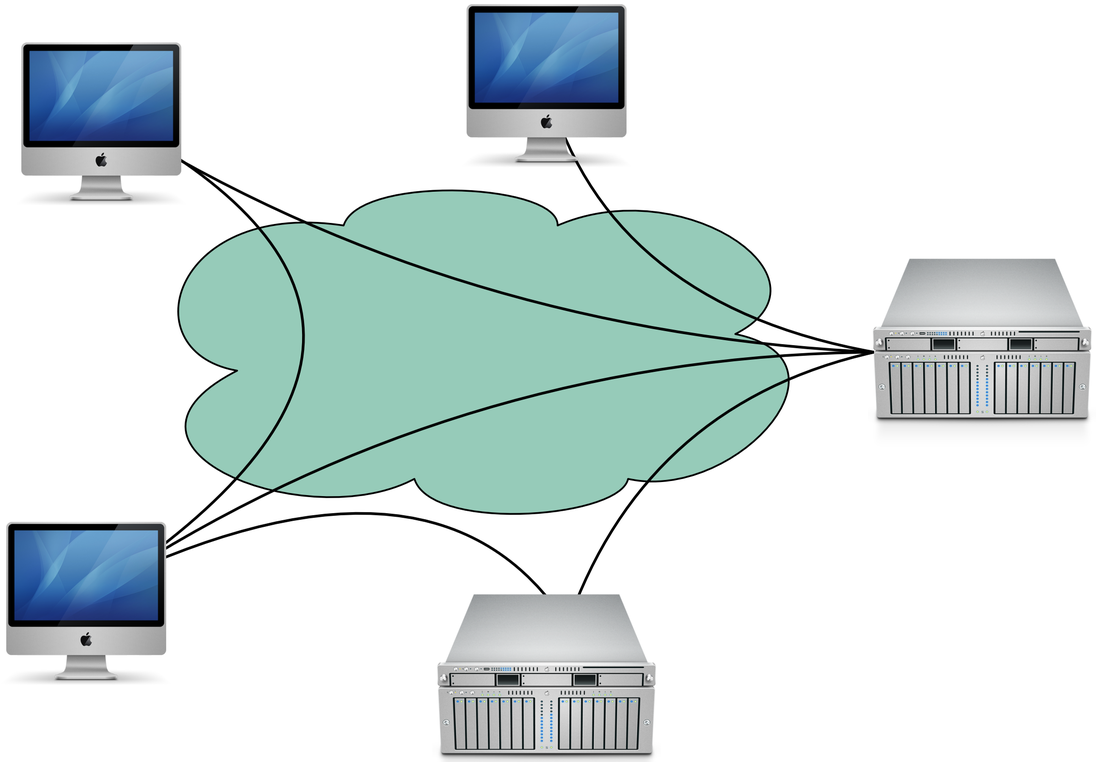
By the end of this lesson, you should be able to…

- Describe 5 key goals of distributed systems
- Describe 8 key common fallacies of distributed systems
- Analyze a system's requirements and determine if it should be implemented as a distributed system or not

What is a distributed system?



Model:
Many servers talking through a network



Model:
Many servers and clients talking through a network

Why expand to distributed systems?

- Scalability
- Performance
- Latency
- Availability
- Fault Tolerance

"Distributed Systems for Fun and Profit", Takada

Distributed Systems Goals

- **Scalability**
- Performance
- Latency
- Availability
- Fault Tolerance

"the ability of a system, network, or process, to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth."

"Distributed Systems for Fun and Profit", Takada

Distributed Systems Goals

- Scalability
- **Performance**
- Latency
- Availability
- Fault Tolerance

"is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used."

"Distributed Systems for Fun and Profit", Takada

Distributed Systems Goals

- Scalability
- Performance
- **Latency**
- Availability
- Fault Tolerance

"The state of being latent; delay, a period between the initiation of something and the it becoming visible."

"Distributed Systems for Fun and Profit", Takada

Distributed Systems Goals

- Scalability
- Performance
- Latency
- **Availability**
- Fault Tolerance

"the proportion of time a system is in a functioning condition. If a user cannot access the system, it is said to be unavailable."

Availability = uptime / (uptime + downtime).

Often measured in "nines"

Availability %	Downtime/year
90%	>1 month
99%	< 4 days
99.9%	< 9 hours
99.99%	<1 hour
99.999%	5 minutes
99.9999%	31 seconds

"Dis[illegible]ada

Distributed Systems Goals

- Scalability
- Performance
- Latency
- Availability
- **Fault Tolerance**

"ability of a system to behave in a well-defined manner once faults occur"

What kind of faults?

Disks fail

Power supplies fail

Networking fails

Security breached

Datacenter goes offline

Power goes out

"Distributed Systems for Fun and Profit", Takada

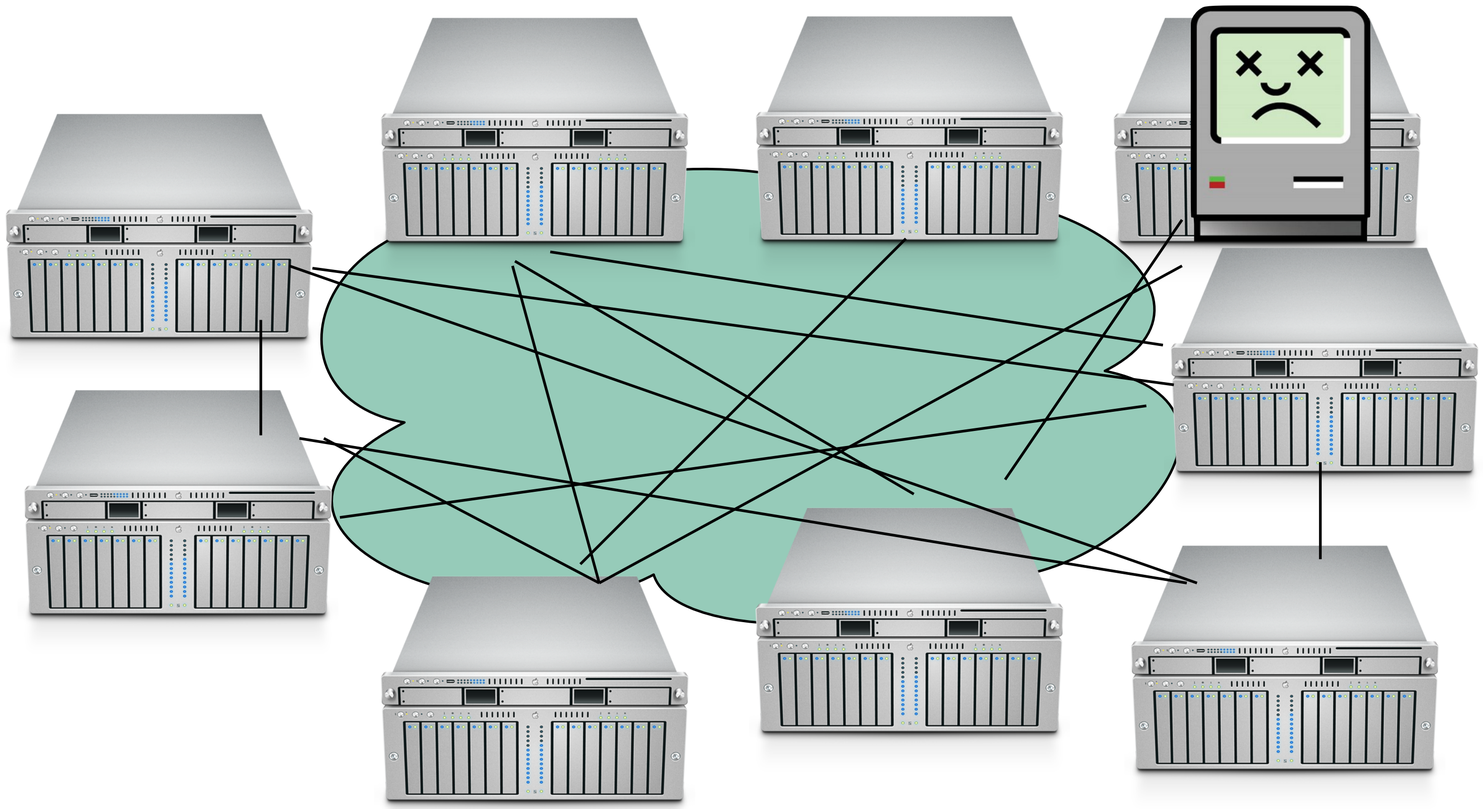
Distributed Systems Challenges

More machines, more problems

- Say there's a 1% chance of having some hardware failure occur to a machine (power supply burns out, hard disk crashes, etc)
- Now I have 10 machines
  - Probability(at least one fails) = 1 - Probability(no machine fails) = $1-(1-.01)^{10}$ = 10%
- 100 machines -> 63%
- 200 machines -> 87%

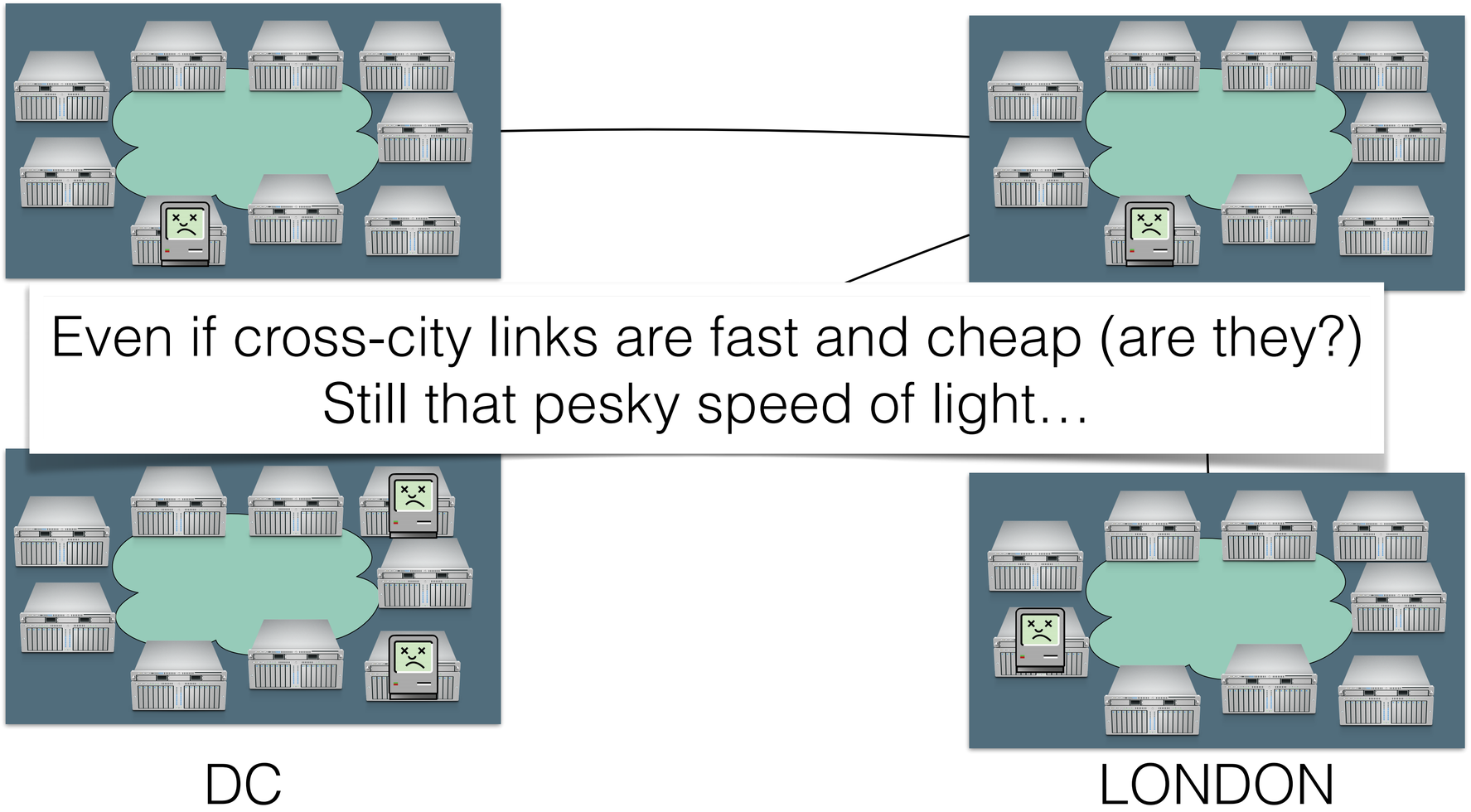
Distributed Systems Challenges

Number of nodes + distance between them



Distributed Systems Challenges

Number of nodes + distance between them

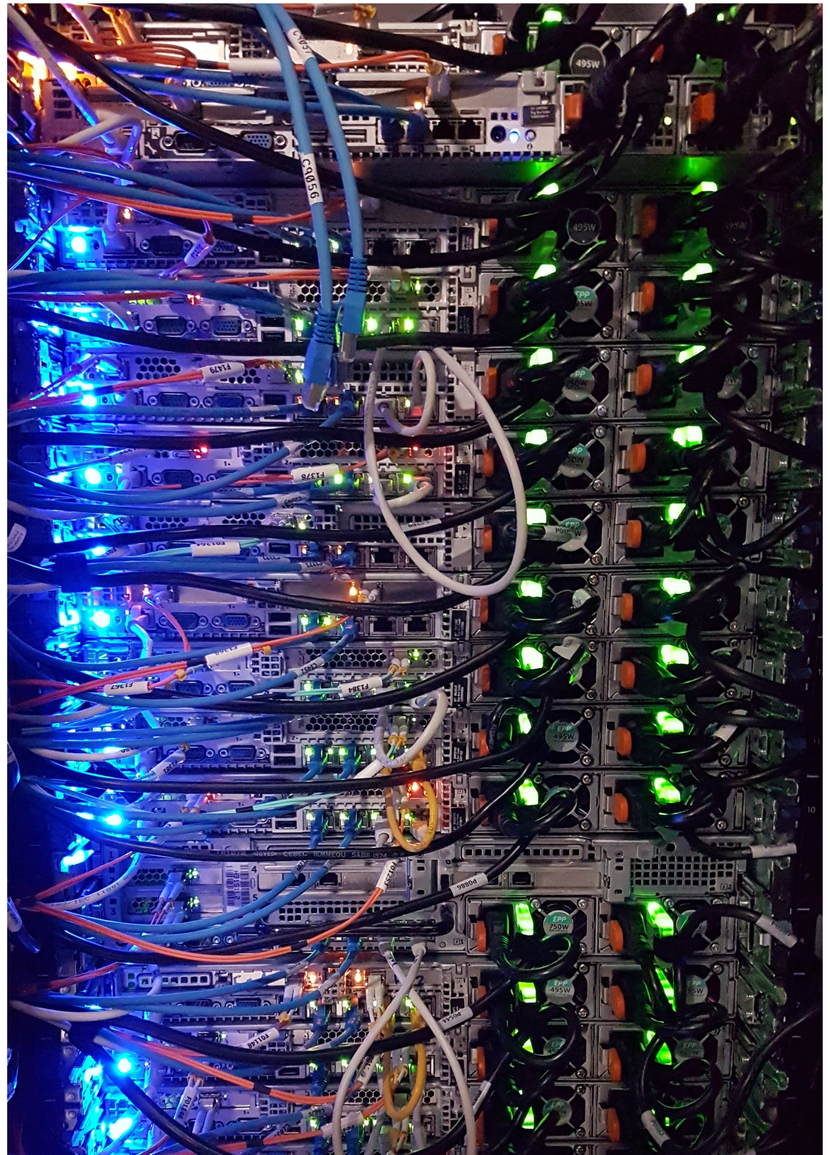


8 Fallacies of Distributed Computing

Sun Microsystems, early 1990's

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. Transport cost is zero
7. The network is homogeneous
8. There is one administrator

Physical Properties of networks



Do these fallacies still hold?

Networks still fail, intermittently and for prolonged periods

arstechnica.com

ars TECHNICA SUBSCRIBE SIGN IN

BIZ & IT —

The discovery of Apache ZooKeeper's poison packet

How PagerDuty found four different bugs.

EVAN GILMAN - 5/13/2015, 9:00 AM

Evan Gilman is an operations engineer at PagerDuty, responsible for designing and automating the company's resilient infrastructure. Prior to PagerDuty, he operated AS4511 at the University of Miami, and has a passion for all things network. This story originally appeared on PagerDuty.

ZooKeeper, for those who are unaware, is a well-known open source project that enables highly reliable distributed coordination. It is trusted by many around the world, including PagerDuty. It provides high availability and linearizability through the concept of a leader, which can be dynamically re-elected, and ensures consistency through a majority quorum.

The leader election and failure detection mechanisms are fairly mature, and typically just work... until they don't. How can this be? Well, after a lengthy investigation, we managed to uncover four different bugs coming together to conspire against us, resulting in random cluster-wide lockups. Two of those bugs lay in ZooKeeper, and the other two were lurking in the Linux kernel. This is our story.

Background: The use of ZooKeeper at PagerDuty

Here at PagerDuty, we have several disparate services that power our alerting pipeline. As events are received, they traverse these services as a series of tasks that get picked up off of various work queues. Each one of these services leverages a dedicated ZooKeeper cluster to coordinate which application host processes each task. As such, you can imagine that ZooKeeper operations are absolutely critical to the reliability of PagerDuty at large.

Part I: The ZooKeeper bugs

wired.com

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY SIGN IN | SUBSCRIBE

LILY HAY NEWMAN BUSINESS 06.29.2018 07:57 PM

Friday's Massive Comcast Outage Shows How Fragile the Internet Is

Comcast customers across the country experienced outages Friday, thanks to multiple cuts to fiber optic cables.

Blame cuts to fiber optic cables for Comcast's outage Friday. GETTY IMAGES

Do these fallacies still hold?

We still rely on other administrators, who are not infallible

Amazon Web Services outage takes a portion of the internet down with it

Zack Whittaker

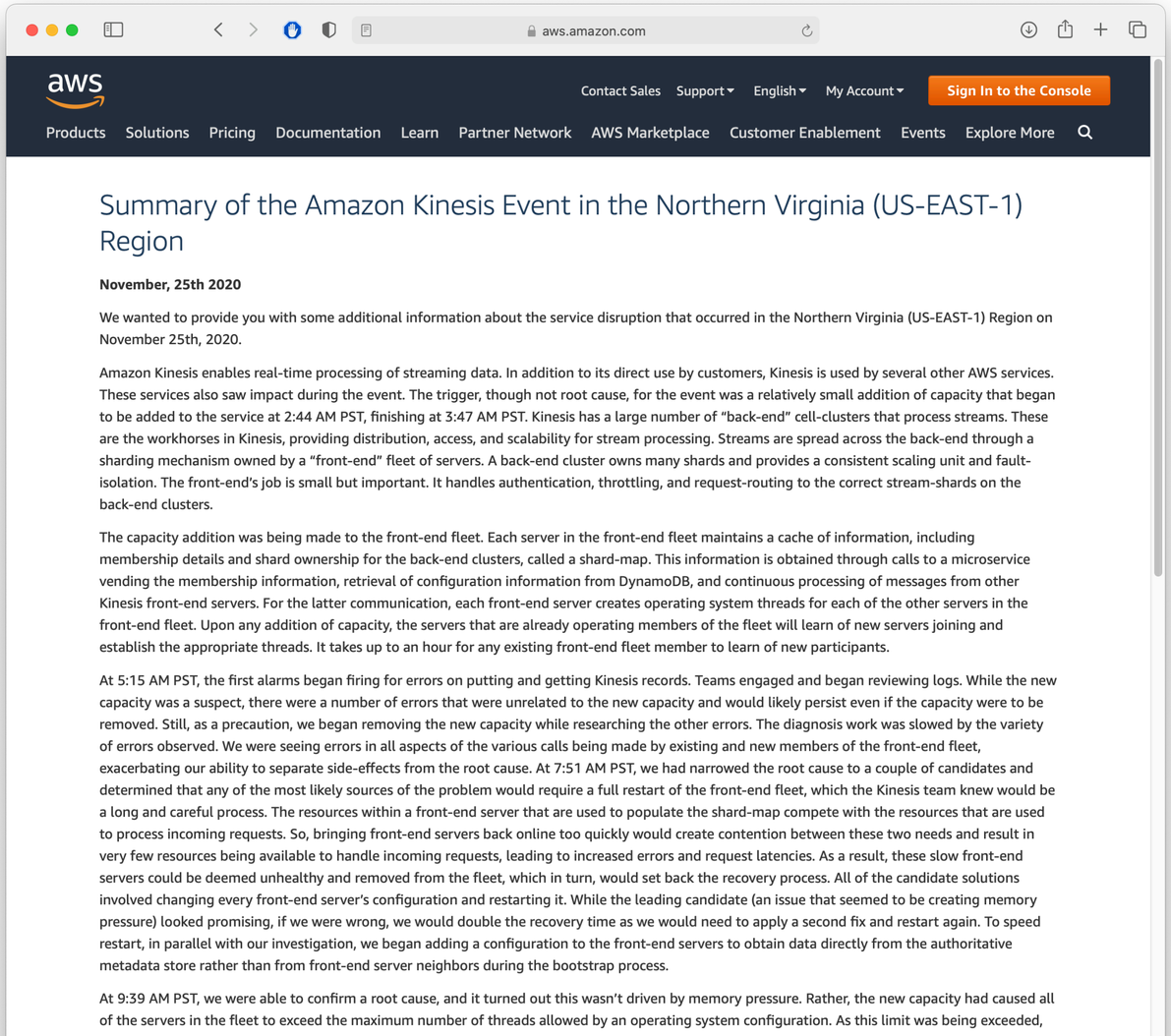
@zackwhittaker / 12:32 PM EST • November 25, 2020

Comment

Image Credits: David Becker / Getty Images

Amazon Web Services is currently having an outage, taking a chunk of the internet down with it.

Several AWS services were experiencing problems as of early Wednesday, according to its status page. That means any app, site or service that relies on AWS might also be down, too. (As I found out the hard way this morning when



Should we make {our software} distributed?

Reflecting on goals + challenges

- Do we need to store more data than one computer can store?
- Do we need to process requests faster than one computer can?
- Are we willing and able to take on these additional complications?
- Next lesson: what tools do we have at our disposal?

This work is licensed under a Creative Commons Attribution-ShareAlike license