why are distributed systems so hard?

@deniseyu21  deniseyu.io
Land acknowledgment:

Portland is built on the unceded, traditional Tribal lands of the Multnomah, Kathlamet, Clackamas, other Chinook bands, Oregon City Tumwater, Tualatin Kalapuya, Molalla, and many other Tribes who make their homes along the Columbia and Willamette rivers.

In using this land, it is also important to acknowledge the policies of genocide, relocation, and assimilation that still impact many Indigenous and Native American families today.

Adapted from pcc.edu/about/diversity/cascade
Software Engineer @ Pivotal Toronto-based

Cloud Foundry
Concourse

Tech-doodling enthusiast

Hello, #LISA19!

@deniseyu21
deniseyu.io/art
AGENDA

- why is distsys a thing
- re CAP
- networks are hard
- how to make life better?

Follow Along! denseyu.io/lisa
A long time ago,
in a data center not too far away...

All business applications talked to one database,
usually hosted on a company's own hardware.

This worked for a while! Until IT stopped being cost centers & started being business enablers.

@denise_yu21
Data storage & retrieval needs evolved as software became business differentiators.

Business analysis & data warehouses

ML & Natural language processing

Faster, bigger queries!

I want all transactions where rabbits under 40 bought 3 or more carrots...

Summer in Norway.
No way.
So we scaled vertically...

until unit economics (or physics) caught up.
REASONS TO HORIZONTALLY DISTRIBUTE:

**Scalability:**
One machine cannot handle request or data size

**Availability:**
If one machine goes down, others keep working

**Latency:**
go faster when data is stored geographically closer to users

@deniseyu21
what does it actually mean to run a distributed system?
LOCAL
all entities in one ADDRESS SPACE

Caller & Receiver known

DISTRIBUTED
multiple address spaces, maybe multiple machines

Recipient details might be unknown

VS.

@deniseyu21
A note on distributed computing 1994

Jim Waldo
Geoff Wyant
Ann Wolfram
Sam Kendall
Sun Microsystems
THE HARD PARTS!

LATENCY
Processor speed vs. network speed

MEMORY ACCESS
Pointers are only valid in their own memory space

PARTIAL FAILURES
Inevitable!
You may have heard the term "shared nothing" architecture:

Machines do not share access to any resources.
A note on distributed computing

Jim Waldo, Geoff White, Andra Walrath, Sam Kendall, and Michael Bay

1994

Distributed computing has many facets that make it complex. Understanding and implementing distributed systems requires not only knowledge of computer science principles but also an understanding of the practical challenges that arise in real-world applications.

**Local**

- All entities in one address space
- vs.

**Distributed**

- Multiple entities, maybe across machines
- Remote procedures call (RPC)

**How to build a distributed app**

1. Make some interfaces
2. Test with great bullet service
3. Concrete idea to make local code perform
4. RPCs
5. Distributed memory

**Latency**

Processor speed is not the same as network speed. Cost of invoking remote procedures is not negligible.

**Partial failures**

Inconsistent failures: Somewhere in the middle of a communication link, we will always happen. We should also be aware of local and remote failures. We can't always expect the same behavior.

**Memory access**

Points are only valid in their own memory space. Therefore, all memory access must be either looped through, managed, or manually synchronized.

**TIP**

- Splitting apart remote and local interactions into separate objects
- More modular cleaner interfaces

deniseyu.io/art  @deniseyu21 🌟
Many machines, running many processes, and "only message-passing via unreliable networks with variable delays, and the system may suffer from partial failures."

Martin Kleppmann, "Designing Data Intensive Applications"
The network is reliable
Latency is ZERO
Bandwidth is infinite
The network is homogeneous
The network is Secure
Transport costs $0
There is only one administrator
Topology doesn’t change

8 Fallacies of Distributed Computing

Originally formulated by L. Peter Deutsch & Colleagues at Sun Microsystems in 1994; #8 added in 1997 by James Gosling
The Byzantine Generals Problem

Tell Bob we attack at dawn.

Ok!

But what if I don't?

That Numpty Tom never tells me anything.

I know lol.

Bob's Map.
So much unreliability!

How can we even know what is true about the state of the world?
MONITOR & observe

@deniseyu21
8 TRAPS OF CHAOS ENGINEERING

1. "More vulnerabilities = more success!"
   Error discovery is not linearly related to resilience outcomes.

2. You can prep an exercise with a subset of the team
   "we were just trying to save time."

3. There is a prescriptive way to "do" chaos eng.
   Chaos Eng is inherently unpredictable.

4. "Real" Chaos Eng goes beyond Game Theory

---

Apollo 11 Launch Test:
During a test, safety precautions were not taken = "an experiment that went wrong" changed space travel forever.

HUMAN FACTORS
SYSTEM RELIABILITY

Chaos Eng borrows from many other industries & incidents.
There is a 15% chance we are already in a partition.

There are lots we can't know. But in distributed computing, we can know one thing: Shit's gonna fail.
CAT THEOREM
2000

Dr. Eric Brewer

"Towards Robust Distributed Systems"

@ Principles of Computing Conf

@deniseyu21
Choose Two
C is for Linearizability

to: cat.state = 'hungry'
t: cat.state = 'full'

All nodes must have t, if anyone showed t.
This is really hard! Instant & universal replication.
Replication lag can’t ever be 0, but engineers spend a lot of time trying to get as close as possible.
BTW, eventual consistency doesn't count.
A is for availability

We tend to think of availability as a binary state. BUT—reality is much messier!

Because latency

@deniseyu21 😊
How can we know if a node is unresponsive... or just slow?

Network latency wasn’t part of the original CAP formulation.

@deniseyu21
Determining a timeout limit is a very scientific process.
P is for PARTITION TOLERANCE

Network partitions occur when network connectivity between two datacenters (running your nodes!) is interrupted!
P is for PARTITION TOLERANCE

During a partition, your nodes might as well be on opposite sides of a wormhole: there is no way to know what's happening on the other side.

@deniseyu21
CAP Theorem: Consistency, Availability, Partition Tolerance
Proof of CAP Theorem

Option 1
Let clients keep R/W in both sides of split
	✖️Linearizability

Option 2
Stop writing in one side until partition ends
	✖️Availability

@deniseyu21
Network partitions are inevitable.

How inevitable?
In the first year of a Google cluster’s life, it will experience:

5 rack failures
3 router failures
8 network maintenances

(Jeffrey Dean)
Hardware will fail

@deniseyu21
Hardware will fail

Network Cables fail
Hardware will fail

Hungry Shark

(Doo Doo Doo Doo Doo)

@deniseyu21️
POLICY —

It’s official: Sharks no longer a threat to subsea Internet cables

First known cable shark attacks were in 1985.

DAVID KRAVETS - 7/10/2015, 5:16 PM
Software will behave weirdly

“Busty” VMs borrow resources from each other—the Noisy Neighbor Problem

~denise yu 21~
Software will behave weirdly

“Stop the World” garbage collection

@deniseyu21
Network glitches
why does any of this matter?
Some part of every system is always at risk of failing.

Susan stopped texting back. Let's get food without her.
Fischer Lynch Paterson
Correctness Result
“Impossibility of Distributed Consensus with One Faulty Process, 1985.”
to manage uncertainty, we have mitigation strategies
**What is Automatic Leader Failover?**

**Leader-Follower**

A common pattern for replicating data across nodes in a cluster. If a follower node fails, it's NO BIG DEAL usually, because other nodes can continue to serve READ requests!

**Step One**

Detect that a node failed! Most databases have a default **TIMEOUT** (ex. 30s). If a node has no activity for this time, it is considered to be **offline**.

**Step Two**

From the remaining followers, elect a new leader.

Becky's data is most up-to-date. I vote Becky.

*Jen OK I'll do it*

Some databases have a CONTROLLER NODE who chooses. He has ONE job!

**Step Three**

Tell the world about the new LEADER.

* You may have encountered "Master-slave replication." I prefer not to use this term, because as technologists, our language matters. We should choose terms that are inclusive & don't cause harm.

* @ denise yu 21
  sketchops. tumblr.com
How does RAFT work?

RAFT is a consensus algorithm.

WHY?
Distributed systems come with uncertainty. Achieving consensus is important so data can be replicated!

WHAT ABOUT NETWORK PARTITIONS?!
In a partition, if the leader falls in the smaller side, a new leader is elected within the majority side. When the partition ends, messages received by the minority side are discarded, and those nodes converge their state to match the majority’s.

When a new write happens:

- RAFT doesn’t stand for anything. It’s a name of logs that goes together.
- Message is queued.
- Copy me!
- Message is forwarded to leader node.
- Leader “commits” the new data & followers increment their state counter.
- Ack!
- Followers copy leader.
- Any node can write!

Used in many projects in the real world & etc.

@deniseq421 with @mt165
this distsys stuff is hard!

... BUT WHY??
another highlight:

Me: so... what is the hard thing about distributed systems? if you had to pick one word...

Student 1: uncertainty?

Me: *beaming* GOOD.  *writes it on the board*

Student 2: Docker

Student 1: actually that's better, take mine off
Woods' theorem:

"as the complexity of a system increases, the accuracy of any single agent's own model of that system decreases rapidly."

BUILDING MENTAL MODELS

in theory: in practice:

@deniseyu21
Mental models are hard to compare, which makes them hard to calibrate.
Mental models are hard to compare, which makes them hard to calibrate.
INCIDENT ANALYSIS

This is the work of John Alls & his team

is particularly great for mental model calibration

@deniseyu21
Blameless discussions optimized for learning

what do you think went wrong?

well that angelfish owed money to a lot of bad in the prod env

oh
Don't accept HUMAN ERROR as the root cause. Dig deeper!
Unintuitive design?

☐ check if you do not wish to receive emails

Alert fatigue?

Not understanding the users' assumptions and needs?
design your systems for humans, not machines
tools and processes should promote learning & sustainable pace
be kind to each other
#HugOps
our superpower, as humans:

empathy
We owe it to our end users & our teams to understand & design for the whole system including the fleshy human parts.
Slides & references

deniseyu.io/lisa