Google

Fallacies of Distributed computing

and their effect on Cloud Solutions

Amo Abeyaratne (5/3/19)

What is this session about?

- What to think about when building data pipelines in Cloud platforms
- Managed is a big deal - but can we take it for granted?
- Best practices for architecting Cloud Data platforms (and how some of the veteran cloud customers deal with it)

This is the opposite of a cloud sales pitch: But don't forget the following..

Edy S. Liongosari
@edyliong

Follow

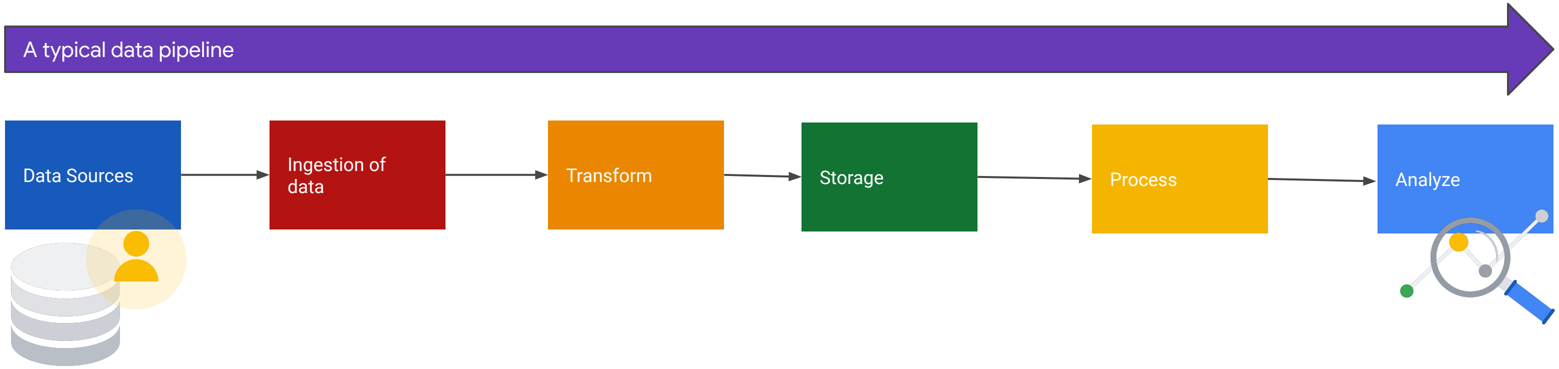
"@joeweinman: @simoncrosby at #structureconf Cloud outages are like plane crashes: lots of publicity but still safer than driving yourself"

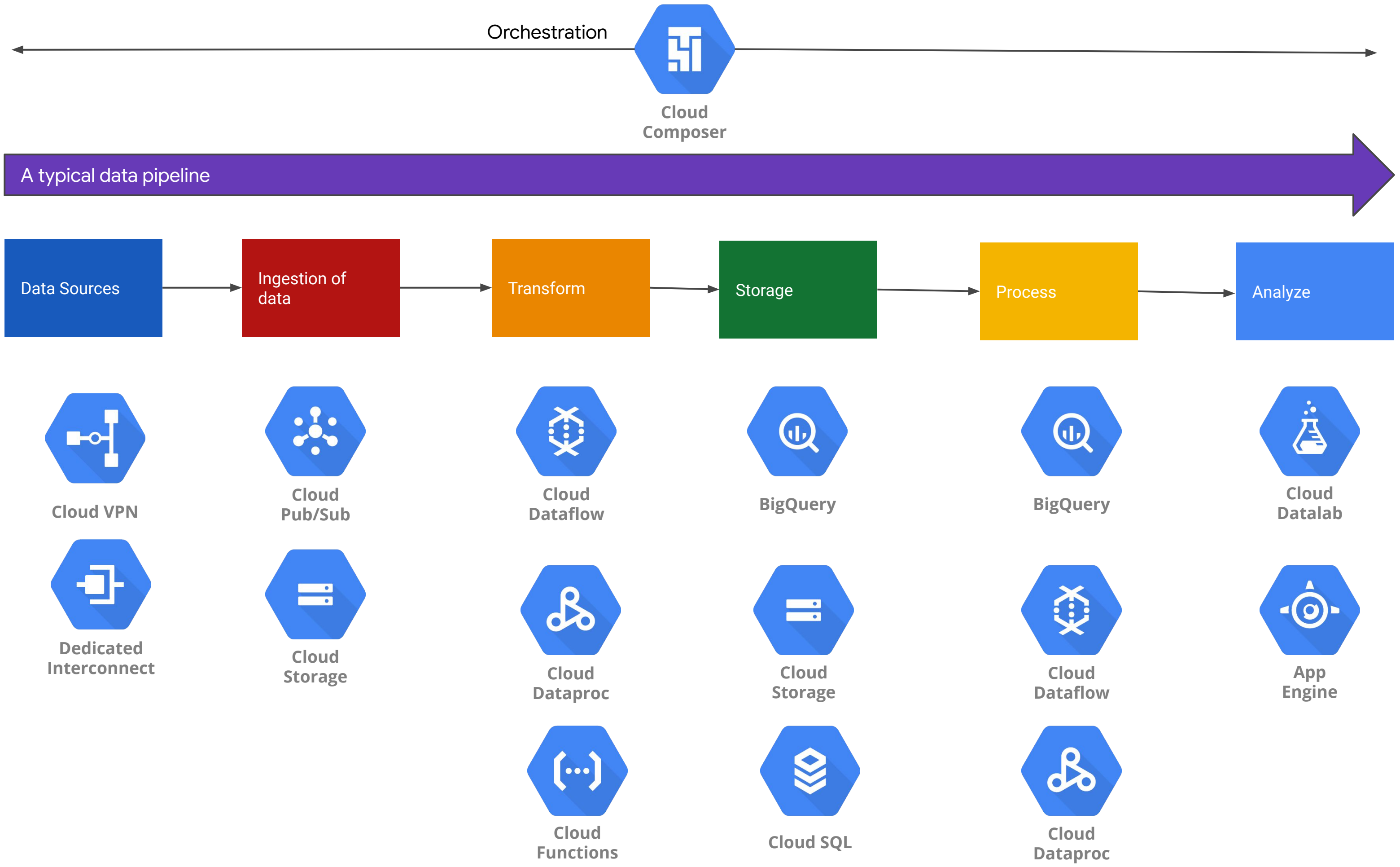
12:08 PM - 22 Jun 2011

"just as flying is still the safest form of travel in terms of number of fatalities per miles travelled, cloud computing services provide as high—or higher—levels of availability and reliability as enterprise data centers."

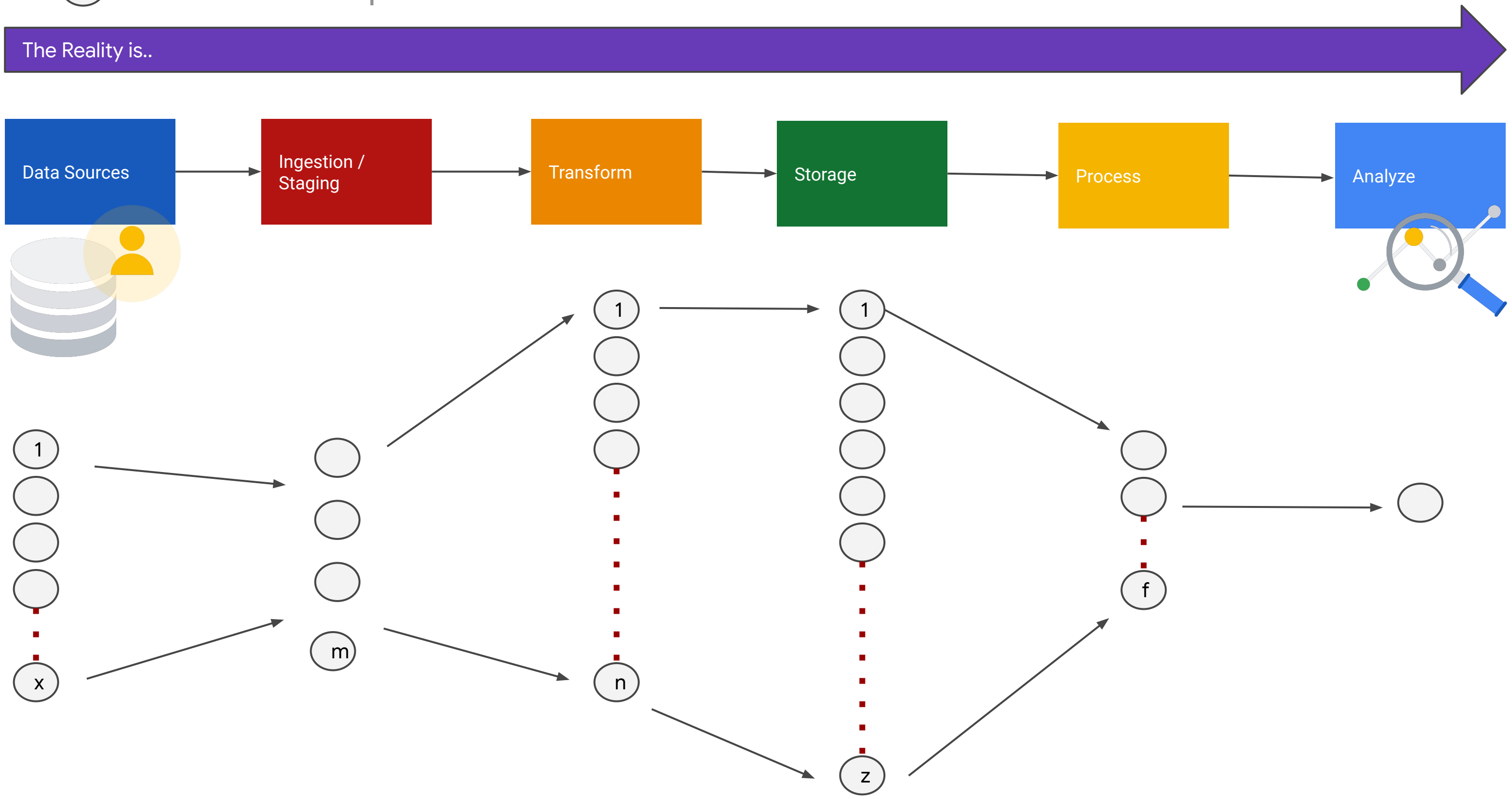
Let's look at a typical data pipeline

How would you build it in the cloud?


A typical data pipeline
Data Sources
Ingestion of data
Transform
Storage
Process
Analyze


Orchestration
Cloud Composer
A typical data pipeline
Data Sources
Ingestion of data
Transform
Storage
Process
Analyze
Cloud VPN
Dedicated Interconnect
Cloud Pub/Sub
Cloud Storage
Cloud Dataflow
Cloud Dataproc
Cloud Functions
BigQuery
Cloud Storage
Cloud SQL
BigQuery
Cloud Dataflow
Cloud Dataproc
Cloud Datalab
App Engine

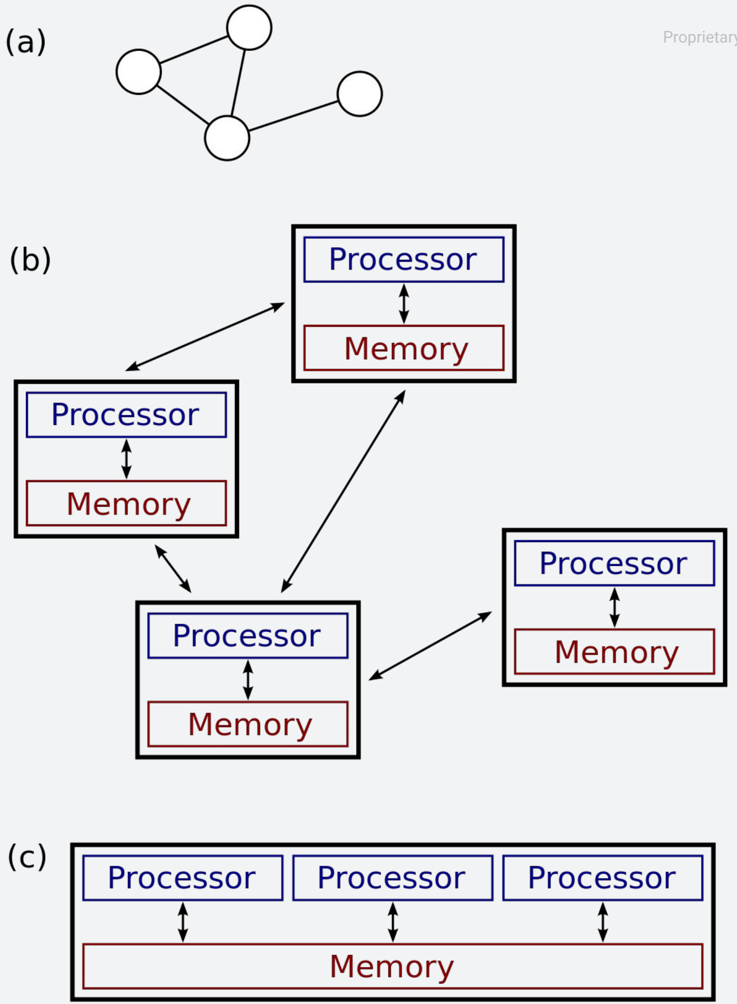
If = Network component / Node



What is a Distributed System?

What is a Distributed System?

A distributed system is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another.

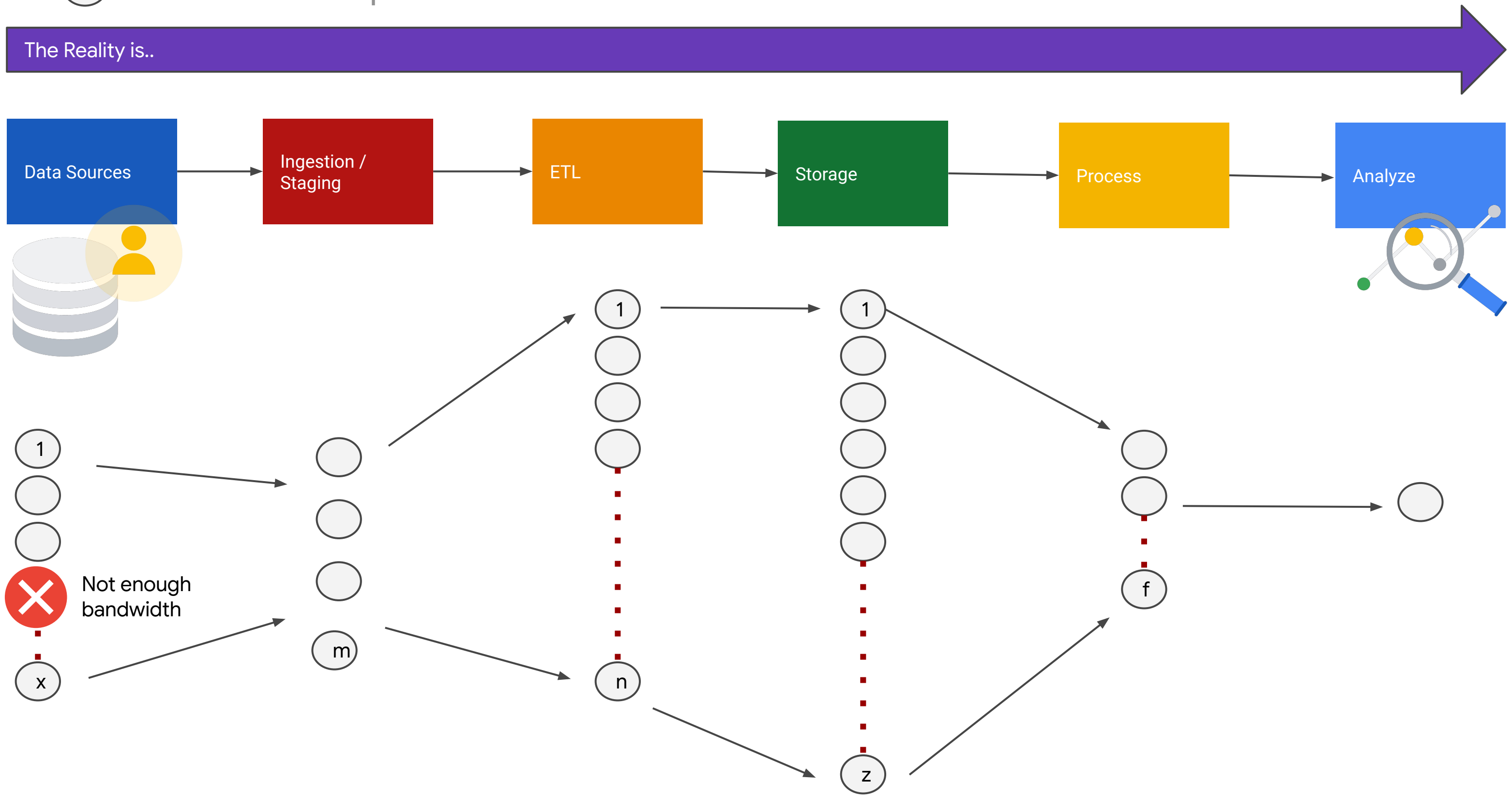


Source: https://en.wikipedia.org/wiki/Distributed_computing

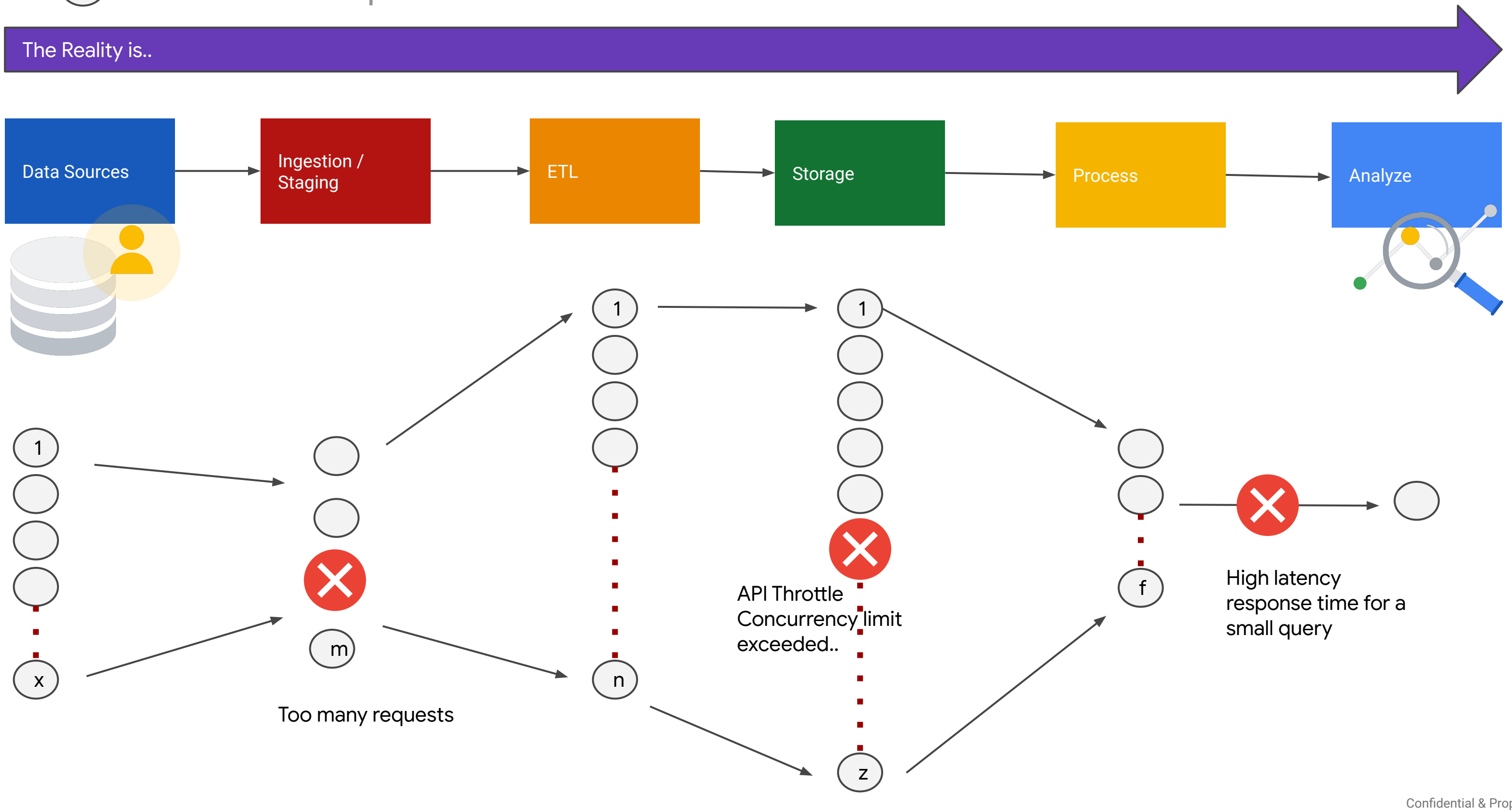
What can fail?

Anything that can fail, will.

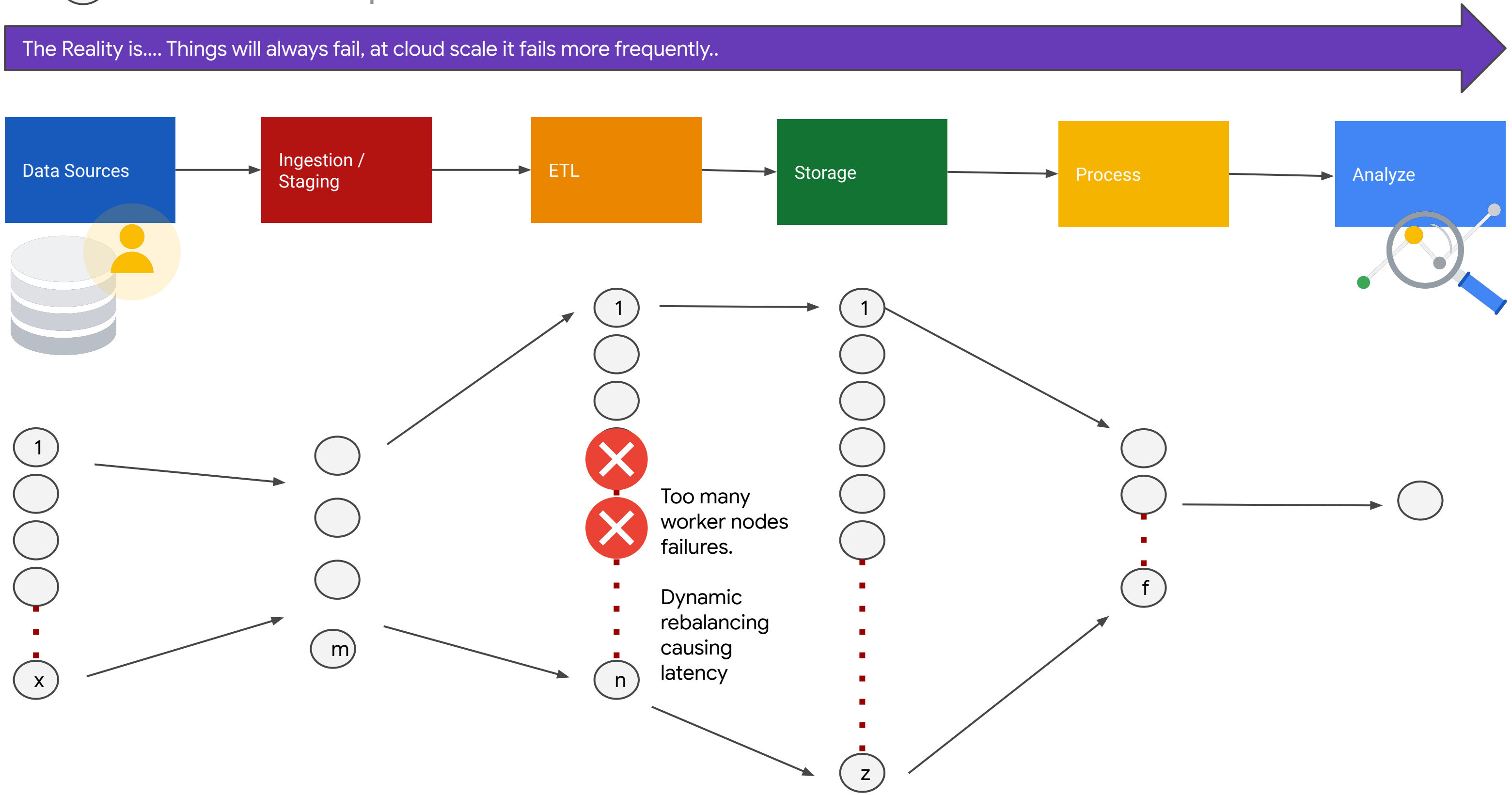
If ◯ = Network component / Node



If = Network component / Node



If ◯ = Network component / Node



What do we take for granted here?

“The fallacies of Distributed Computing”

The fallacies of Distributed Computing

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

Peter Deutsch - Sun Microsystems, 1994

How much time you would spend uploading data?

Close — Far

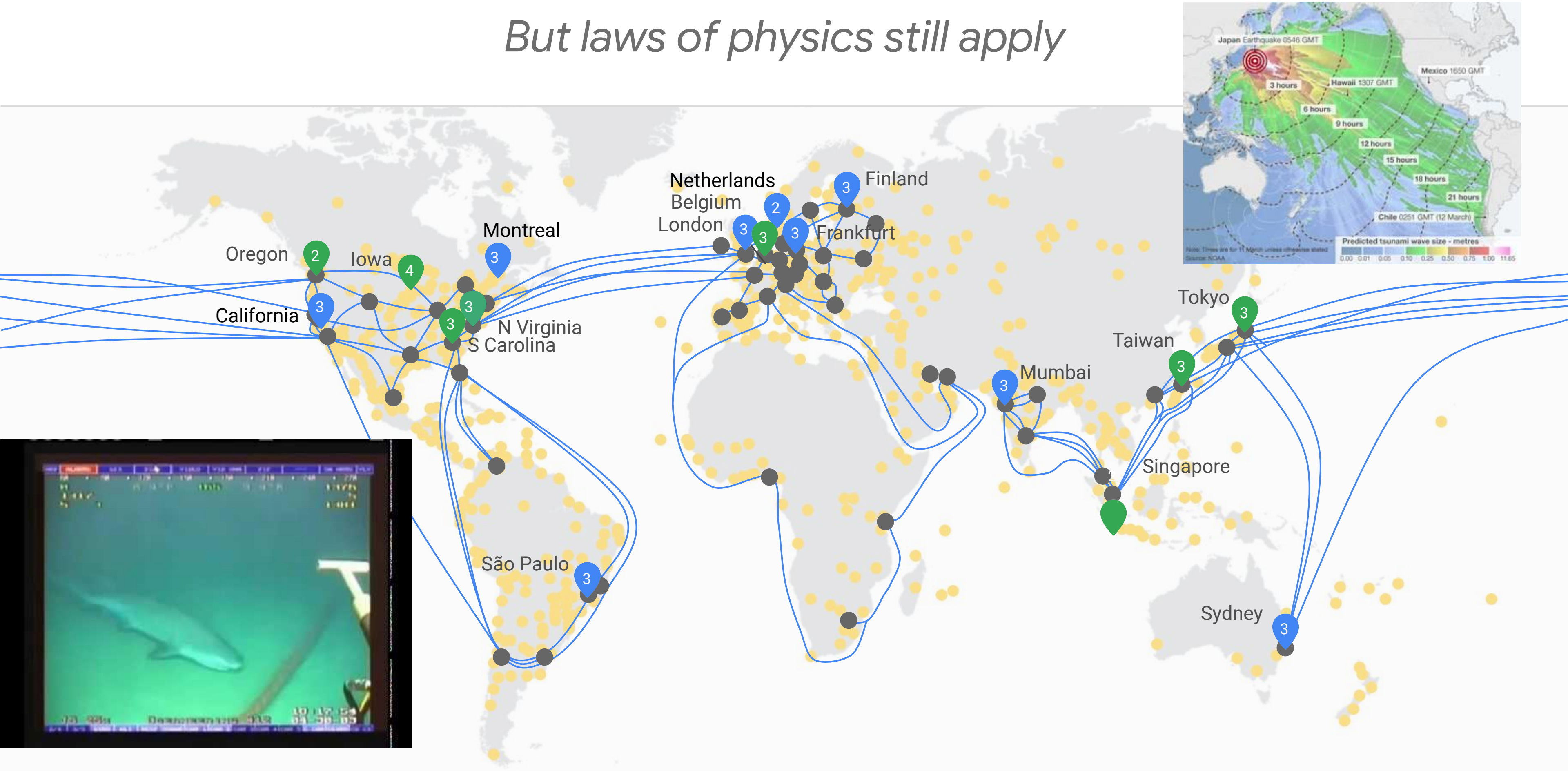
Data Size						
100 PB	124 days	3 years	34 years	340 years	3,404 years	34,048 years
10 PB	12 days	124 days	3 years	34 years	340 years	3,404 years
1 PB	30 hours	12 days	124 days	3 years	34 years	340 years
100 TB	3 hours	30 hours	12 days	124 days	3 years	34 years
10 TB	18 minutes	3 hours	30 hours	12 days	124 days	3 years
1 TB	2 minutes	18 minutes	3 hours	30 hours	12 days	124 days
100 GB	11 seconds	2 minutes	18 minutes	3 hours	30 hours	12 days
10 GB	1 second	11 seconds	2 minutes	18 minutes	3 hours	30 hours
1 GB	0.1 seconds	1 second	11 seconds	2 minutes	18 minutes	3 hours
	100 Gbps	10 Gbps	1 Gbps	100 Mbps	10 Mbps	1 Mbps

Network Bandwidth

Ref: https://cloud.google.com/solutions/transferring-big-data-sets-to-gcp

Hundreds of thousands of miles of fiber optic cable
connecting all of our datacenter regions and 100+ points of presence.

But laws of physics still apply



So, what should we do?

Design for failure :)

Start your solution design with the end user in mind..

*https://landing.google.com/sre/books/ (if interested to learn more about Google SRE philosophy)

The fallacies of Distributed Computing

Meets

Design principles for Cloud Solutions

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

1. Design for failure
2. Loose coupling
3. Implement “Elasticity”
4. Build Security in every layer
5. Don't fear constraints
6. Think Parallel
7. Leverage different storage options

This doesn’t matter anymore, we have achieved a good level of abstraction

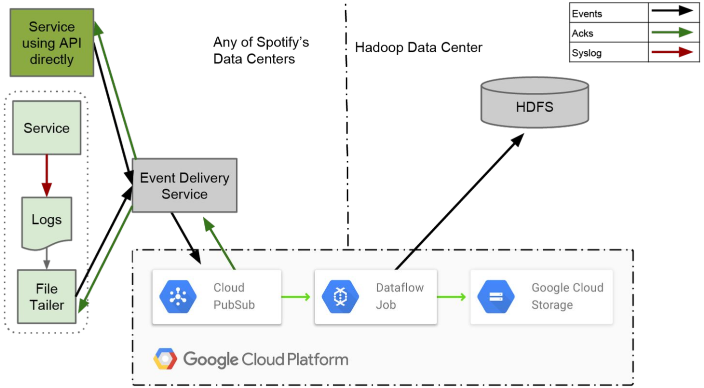
** These best practices for Architecting in Cloud was introduced by AWS in 2011.*
https://aws.amazon.com/whitepapers/architecting-for-the-aws-cloud-best-practices/

Design principles for Cloud Solutions

1. Design for failure
2. Loose coupling
3. Implement “Elasticity”
4. Build Security in every layer
5. Don't fear constraints
6. Think Parallel
7. Leverage different storage options

- **Avoid single points of failure**
- Use **queues to decouple** : *Job A passes the output to a queue for Job B to pull from*
- ***Idempotent*** *and* ***Metadata driven*** *scaling - new jobs/actors should be bootstrapped to ask “who am I, what am I supposed to do?”*
- In a network, location/perimeter security is useless - so wrap all the components with **ownership, audit.**
- Memory, IOPS distribution by **sharding**, **Caching** for performance
- **Divide, Parallelize and conquer**. Serial/Sequential days are gone.
- **One database does not fit all** anymore. Think different stores for the outcome (Graph, Key-value, OLAP)

Learn from the industry leaders..



"To achieve good end-to-end latency, we wrote our ETL as a streaming job. By having it constantly running, we're able to incrementally fill discrete hourly buckets as the data arrives. This gives us better latency compared to a batch job that exported data once at the end of every hour."

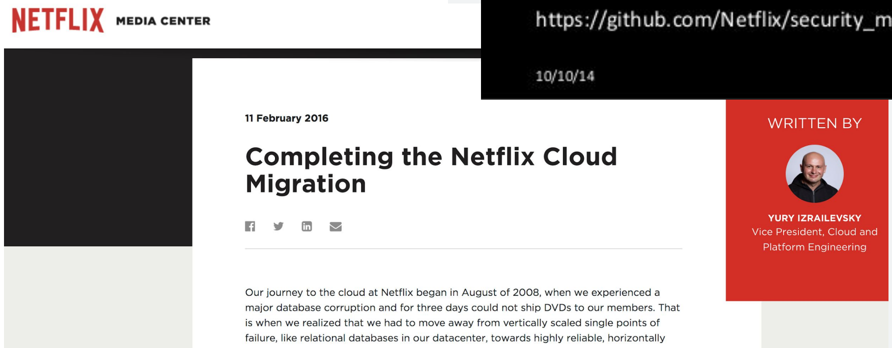
- Igor Maravić, Spotify

Spotify has a great published story about their migration to cloud.

BigQuery	**10 million queries** per month on scanning over **500PBs**
Pub/Sub	**Over 1 Trillion** messages/day - under **400ms latency**
Dataflow	**5000+ dataflow jobs** a day
Dataproc	Lots of dynamically spun up job scoped clusters

*Watch on youtube our Cloud Next presentation they did talking about how they shutdown the largest Hadoop cluster in Europe

“The best way to avoid failure is to fail constantly”



NETFLIX MEDIA CENTER

11 February 2016

Completing the Netflix Cloud Migration

Our journey to the cloud at Netflix began in August of 2008, when we experienced a major database corruption and for three days could not ship DVDs to our members. That is when we realized that we had to move away from vertically scaled single points of failure, like relational databases in our datacenter, towards highly reliable, horizontally

WRITTEN BY

YURY IZRAILEVSKY
Vice President, Cloud and Platform Engineering

Thank You

END