Lessons from Large-Scale Cloud Software at Databricks

Matei Zaharia

@matei_zaharia

databricks

Outline

The cloud is eating software, but why?

About Databricks

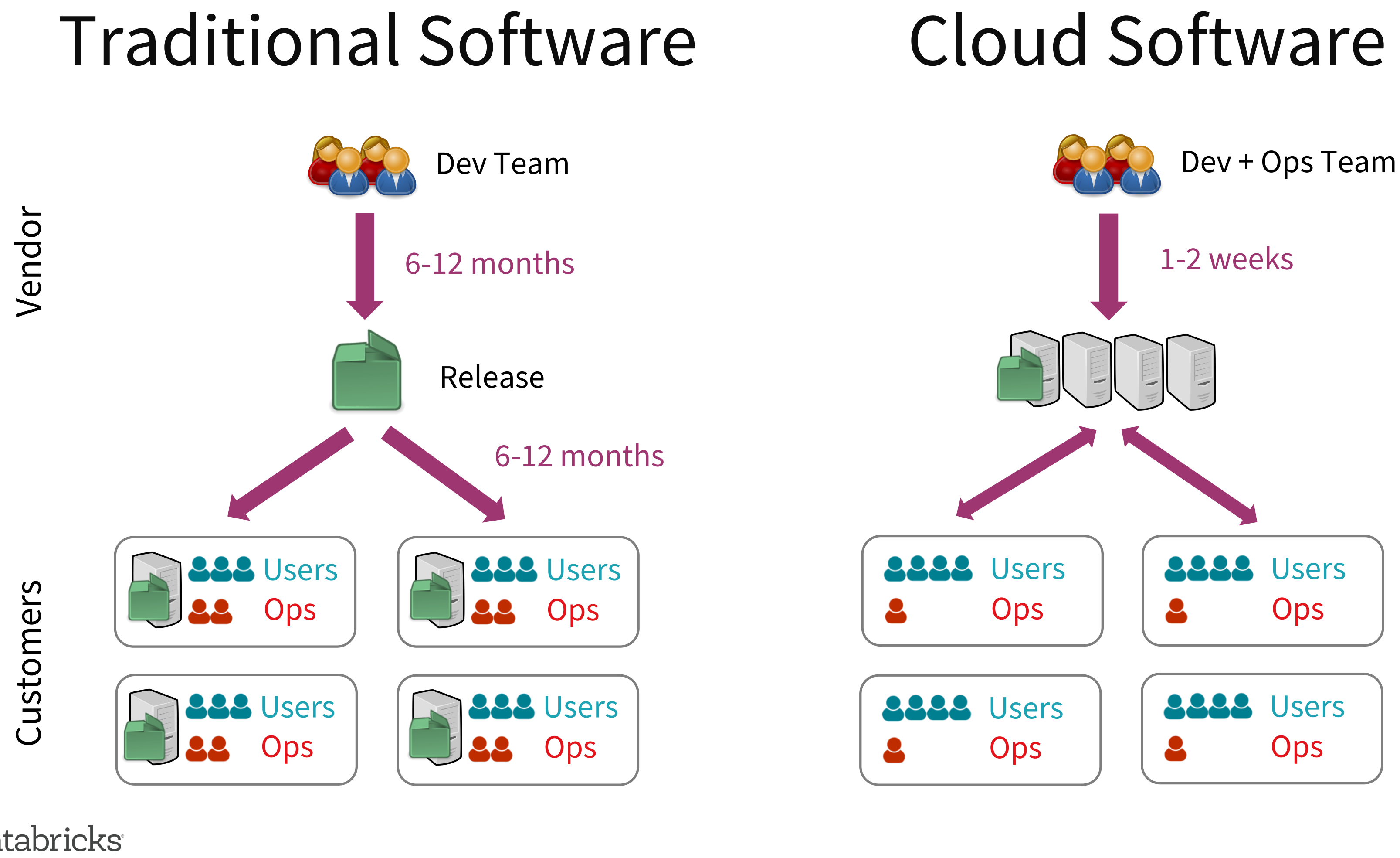
Challenges, solutions and research questions

Outline

The cloud is eating software, but why?

About Databricks

Challenges, solutions and research questions


Traditional Software
Cloud Software
Vendor
Customers
Dev Team
6-12 months
Release
6-12 months
Users
Ops
Users
Ops
Users
Ops
Users
Ops
Dev + Ops Team
1-2 weeks
Users
Ops
Users
Ops
Users
Ops
Users
Ops
databricks

Why Use Cloud Software?

1. Management built-in: much more value than the software bits alone (security, availability, etc)

2. Elasticity: pay-as-you-go, scale on demand

3. Better features released faster

databricks

Differences in Building Cloud Software

+ Release cycle: send to users faster, get feedback faster

+ Only need to maintain 2 software versions (current & next), in fewer configurations than you'd have on-prem

– Upgrading without regressions: very hard, but critical for users to trust your cloud (on-prem apps don't need this)
  - Includes API, semantics, and performance regressions

Differences in Building Cloud Software

– Building a multitenant service: significant scaling, security and performance isolation work that you won't need on-prem (customers install separate instances)

– Operating the service: security, availability, monitoring, etc (but customers would have to do it themselves on-prem)

+ Monitoring: see usage live for ops & product analytics

Many of these challenges aren't studied in research

About Databricks

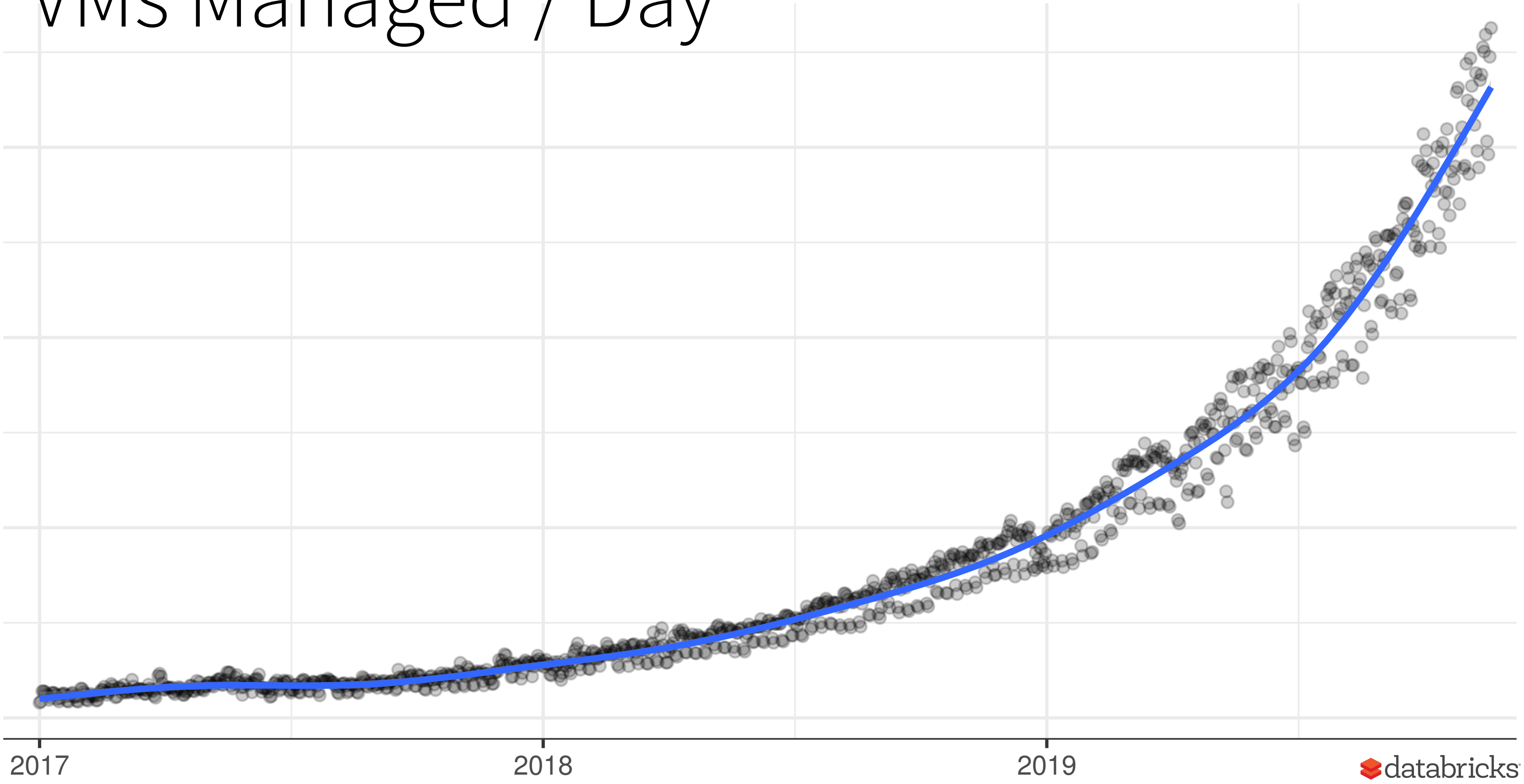
Founded in 2013 by the Apache Spark team at UC Berkeley

Data and ML platform on AWS and Azure for >5000 customers

- Millions of VMs launched/day, processing exabytes of data
- 100,000s of users

1000 employees, 200 engineers, >$200M ARR

VMs Managed / Day



Some of Our Customers

Financial Services
- FINRA
- BlackRock
- Fidelity Investments
- Nasdaq

Healthcare & Pharma
- Regeneron
- Human Longevity, Inc.
- Sanofi
- American Diabetes Association.
- amino

Media & Entertainment
- NBCUniversal
- Viacom
- Riot Games
- SiriusXM
- Bethesda Game Studios
- McGraw Hill Education
- zynga
- TiVo
- Live Nation

Data & Analytics Services
- Radius
- Dow Jones
- Thomson Reuters
- Elsevier
- edmunds

Technology
- Autodesk
- hp
- Samsung Electronics
- Adobe
- Cisco
- GoPro
- NetApp

Public Sector
- U.S. Citizenship and Immigration Services
- AARP
- BlackSky
- CMS — Centers for Medicare & Medicaid Services
- U.S. Securities and Exchange Commission
- noblis
- DigitalGlobe

Retail & CPG
- Red Bull
- ShopRite
- Groupon
- Dollar Shave Club
- flipp
- Chegg
- Westfield
- overstock.com

Consumer Services
- Expedia
- Flipboard
- Hotels.com
- mapquest
- OpenTable
- Blue Apron
- Medium
- glassdoor
- myfitnesspal

Marketing & AdTech
- MediaMath
- Social Code
- Tune
- LoyaltyOne
- eyeview
- AIMIA
- Radius
- Inneractive

Energy & Industrial IoT
- Shell
- DNV·GL
- Quby
- Siemens
- ConocoPhillips

Some of Our Customers

Financial Services | Healthcare & Pharma | Media & Entertainment | Data & Analytics Services | Technology

FINRA

Identify fraud using machine learning on 30 PB of trade data

Public Sector | Retail & CPG | Consumer Services | Marketing & AdTech | Energy & Industrial IoT

Some of Our Customers

Financial Services

Healthcare & Pharma

Media & Entertainment

Data & Analytics Services

Technology

Public Sector

Retail & CPG

Consumer Services

Marketing & AdTech

Energy & Industrial IoT

REGENERON

Correlate 500,000 patients' records with their DNA to design therapies

Some of Our Customers

Financial Services

Healthcare & Pharma

Media & Entertainment

Data & Analytics Services

Technology

Curb abusive behavior in the world's largest online game

Public Sector

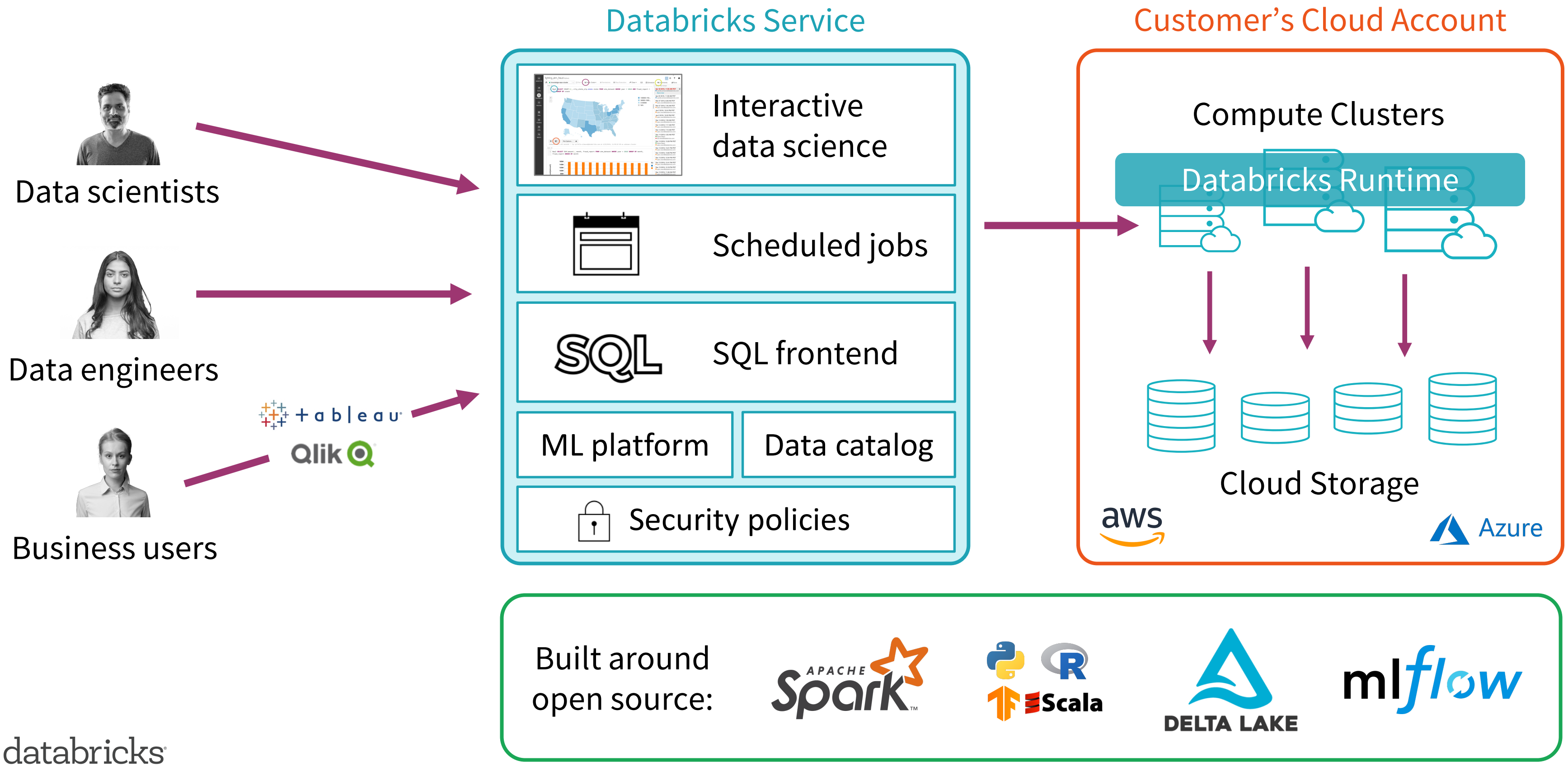
Retail & CPG

Consumer Services

Marketing & AdTech

Energy & Industrial IoT

Our Product


Databricks Service
Customer's Cloud Account
Data scientists
Data engineers
Business users
tableau
Qlik
Interactive data science
Scheduled jobs
SQL
SQL frontend
ML platform
Data catalog
Security policies
Compute Clusters
Databricks Runtime
Cloud Storage
aws
Azure
Built around open source:
APACHE Spark
R
Scala
DELTA LAKE
mlflow

Our Specific Challenges

All the usual challenges of SaaS:

- Availability, security, multitenancy, updates, etc

Plus, **the workloads themselves are large-scale!**

- One user job could easily overload control services
- Millions of VMs ⇒ many weird failures

Four Lessons

1. What goes wrong in cloud systems?
2. Testing for scalability & stability
3. Developing control planes
4. Evolving big data systems for the cloud

Four Lessons

1. What goes wrong in cloud systems?
2. Testing for scalability & stability
3. Developing control planes
4. Evolving big data systems for the cloud

What Goes Wrong in the Cloud?

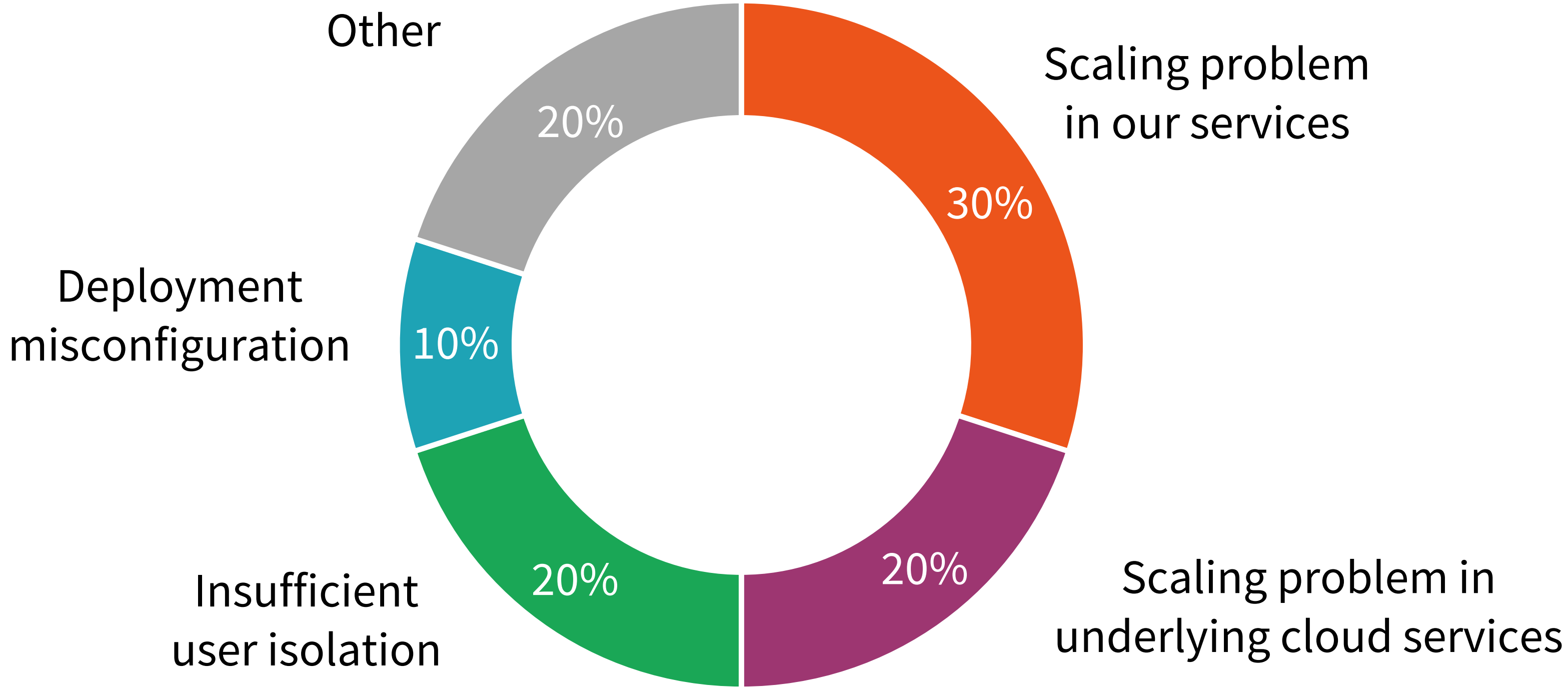
Academic research studies many kinds of failures:

- Software bugs, network config, crash failures, etc

These matter, but other problems often have larger impact:

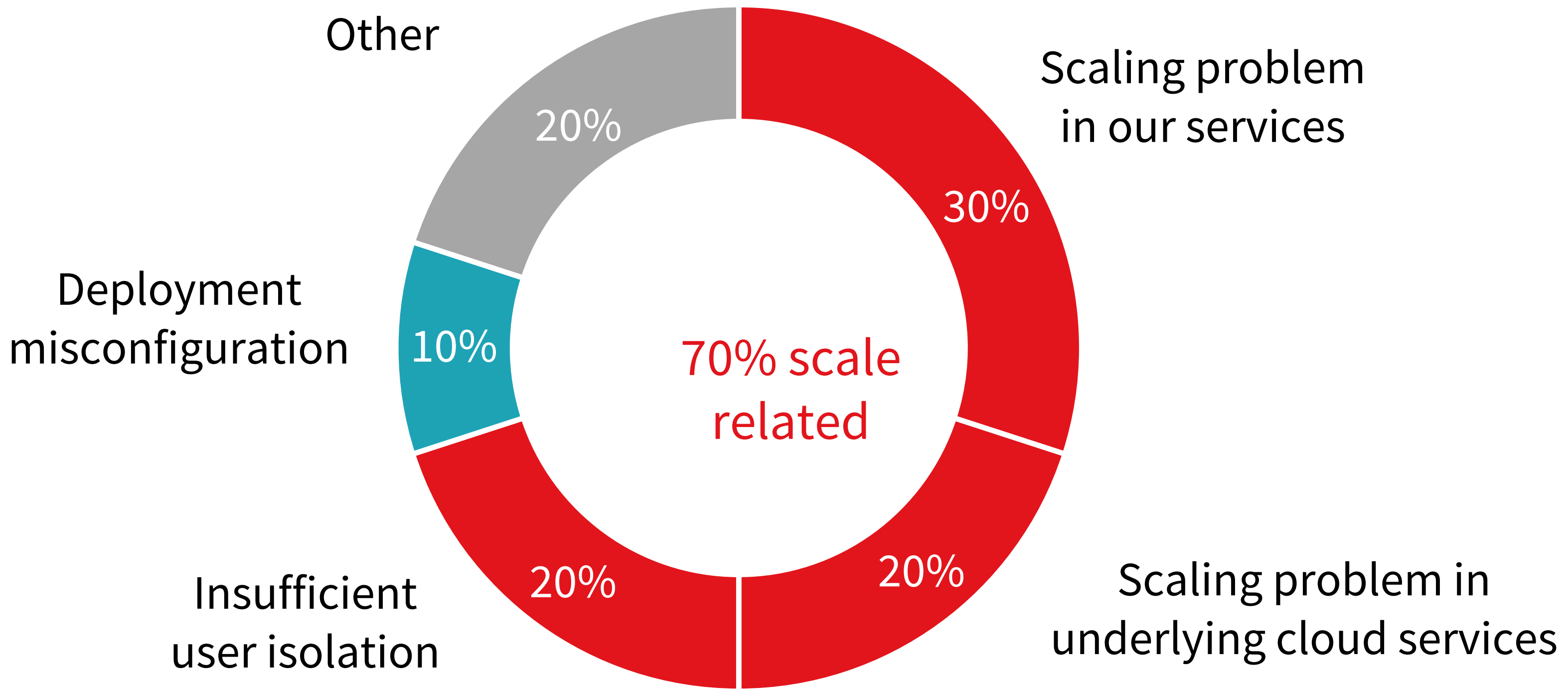
- Scaling and resource limits
- Workload isolation
- Updates & regressions

Causes of Significant Outages



databricks

Causes of Significant Outages



Some Issues We Experienced

Cloud networks: limits, partitions, slow DHCP, hung connections

Automated apps creating large load

Very large requests, results, etc

Slow VM launches/shutdowns, lack of VM capacity

Data corruption writing to cloud storage

Example Outage: Aborted Jobs

Jobs Service launches & tracks jobs on clusters

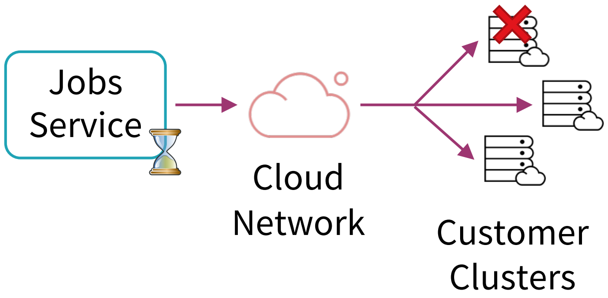
1 customer running many jobs/sec on same cluster

Cloud's network reaches a limit of 1000 connections/VM between Jobs Service & clusters

- After this limit, new connections hang in state SYN_SENT

Resource usage from hanging connections causes memory pressure and GC

Health checks to some jobs time out, so we abort them



Surprisingly Rare Issues

1 cloud-wide VM restart on AWS (Xen patch)

1 misreported security scan on customer VM

1 significant S3 outage

1 kernel bug (hung TCP connections due to SACK fix)

Lessons

Cloud services must handle load that varies on many dimensions, and rely on other services with varying limits & failure modes

- Problems likely to get worse in a “cloud service economy”

End-to-end issues remain hard to prevent

The usual factors of MTTR, monitoring, testing, etc help

Four Lessons

1. What goes wrong in cloud systems?
2. Testing for scalability & stability
3. Developing control planes
4. Evolving big data systems for the cloud

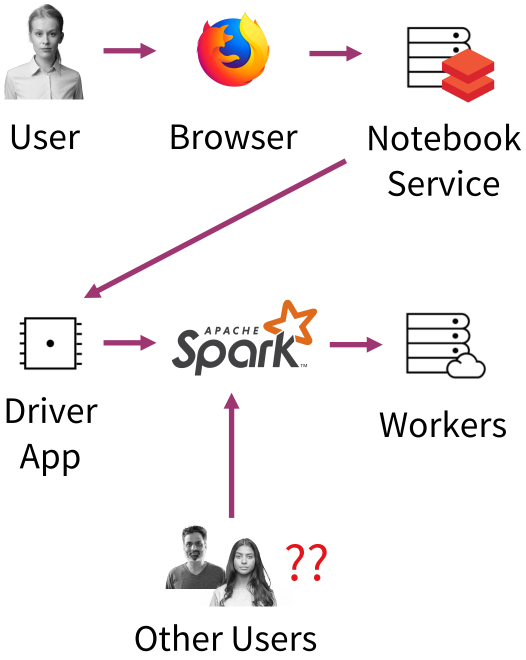
Testing for Scalability & Stability

Software correctness is a Boolean property: does your software give the right output on a given input?

Scalability and stability are a matter of degree

- What load will your system fail at? (any system with limited resources will)
- What failure behavior will you have? (crash all clients, drop some, etc)

Example Scalability Problems



Large result: can crash browser, notebook service, driver or Spark

Large record in file

Large # of tasks

Code that freezes a worker

+ All these affect other users!

Databricks Stress Test Infrastructure

1. Identify dimensions for a system to scale in (e.g. # of users, number of output rows, size of each output row, etc)

2. Grow load in each dimension until a failure occurs

3. Record failure type and impact on system
   - Error message, timeout, wrong result?
   - Are other clients affected?
   - Does the system auto-recover? How fast?

4. Compare over time and on changes

Example Output

Suite	Test	MaxValue	State	MaxStep	Flags	Message	Prev MaxValue	Prev State	Prev MaxStep	Prev Flags	Prev Message	MaxStep diff
ScalaClusterSuite	big broadcast	1000000000	FAILED	4		at sun.nio.ch.FileCha	1000000000	FAILED	4		at sun.nio.ch.FileChan	0
ScalaClusterSuite	big tasks	1000000000	FAILED	4		at java.io.ByteArray(	1000000000	FAILED	4		at java.util.Arrays.copy	0
ScalaClusterSuite	caching large objects	100000000	TIMED_OUT	3			100000000	TIMED_OUT	3			0
ScalaClusterSuite	caching small objects	1000000000	SUCCEEDED	4			1000000000	SUCCEEDED	4			0
ScalaClusterSuite	crashing executors	1000	SUCCEEDED	4			1000	SUCCEEDED	4			0
ScalaClusterSuite	crashing tasks	1000	SUCCEEDED	4			1000	SUCCEEDED	4			0
ScalaClusterSuite	display large rows	10000000	TIMED_OUT	4	CB SB LS	java.lang.Exception:	10000000	FAILED	4		org.apache.spark.Spark	0
ScalaClusterSuite	lots of shuffle tasks	100000	TIMED_OUT	3			1000000	FAILED	4		at org.apache.spark.sc	-1
ScalaClusterSuite	lots of tasks	1000000	TIMED_OUT	3			1000000	TIMED_OUT	3			0
ScalaClusterSuite	popular key in groupBy	10000000	TIMED_OUT	4			10000000	TIMED_OUT	4			0
ScalaDriverSuite	allocate big arrays	100	TIMED_OUT	3	CB		100	TIMED_OUT	3	CB		0
ScalaDriverSuite	allocate small arrays	100000	FAILED	3		at Notebook$$anonf	100000	FAILED	3		at Notebook$$anonfun	0
ScalaDriverSuite	infinite loop	0	TIMED_OUT	1	CB SB LS	java.lang.Exception:	0	TIMED_OUT	1	CB SB LS	java.lang.Exception: Co	0
ScalaDriverSuite	no such method error	0	SUCCEEDED	4			0	SUCCEEDED	4			0
ScalaDriverSuite	print a lot	1000000000	TIMED_OUT	4	CB SB LS	java.lang.Exception:	100000000	TIMED_OUT	3			1
ScalaDriverSuite	system exit	0	FAILED	1		at com.databricks.ba	0	FAILED	1		at com.databricks.back	0
ScalaDriverSuite	thread sleep	100	TIMED_OUT	2			100	TIMED_OUT	2			0
SQLClusterSuite	broadcast join	1000000000	FAILED	4		at com.databricks.ba	1000000000	SUCCEEDED	4			0
SQLClusterSuite	broadcast join on cached dat	1000000000	SUCCEEDED	4			1000000000	SUCCEEDED	4			0
SQLClusterSuite	count distinct	1000000000	TIMED_OUT	4			1000000000	TIMED_OUT	4			0
SQLClusterSuite	count distinct with common k	10000000	FAILED	2		at com.databricks.ba	1000000000	SUCCEEDED	4			-2
SQLClusterSuite	self join	10000000	FAILED	2		at com.databricks.ba	100000000	TIMED_OUT	3			-1
SQLClusterSuite	self join on cached data	1000000000	TIMED_OUT	4			1000000	TIMED_OUT	4			0
SQLClusterSuite	self join with common keys	10000000	TIMED_OUT	2			10000000	TIMED_OUT	2			0

Four Lessons

1. What goes wrong in cloud systems?
2. Testing for scalability & stability
3. Developing control planes
4. Evolving big data systems for the cloud

Developing Control Planes

Cloud software consists of interacting, independently updated services, many of which call other services

What should be the programming model for this software?

Examples

Cluster manager service:

- API: requests to launch, scale and shut down clusters
- Behavior: request VMs, set up clusters, reuse VMs in pools
- State: requests, running VMs, etc

Jobs service:

- API: scheduled or API-triggered jobs to execute
- Behavior: acquire a cluster, run job, monitor state, retry
- State: jobs to be run, what’s currently active, where is it, etc

Examples

Cloud VM Service

IAM Service

. . .

Cluster manager service:

- API: requests to launch, scale and shut down clusters
- Behavior: request VMs, set up clusters, reuse VMs in pools
- State: requests, running VMs, etc

Usage Service

Jobs service:

- API: scheduled or API-triggered jobs to execute
- Behavior: acquire a cluster, run job, monitor state, retry
- State: jobs to be run, what's currently active, where is it, etc

Notebook Service

. . .

Control Plane Infrastructure

Our Platform Team develops a service framework that handles:

- Deployment: AWS, Azure, local, special environments
- Storage: databases, schema updates, etc
- Security tokens & roles
- Monitoring
- API routing & limiting
- Feature flagging

Our service stack:

kubernetes
Scala
JSonnet
Prometheus
envoy
databricks

Best Practices

Isolate state: relational DB is usually enough with org sharding

Isolate components that scale differently: allows separate scaling

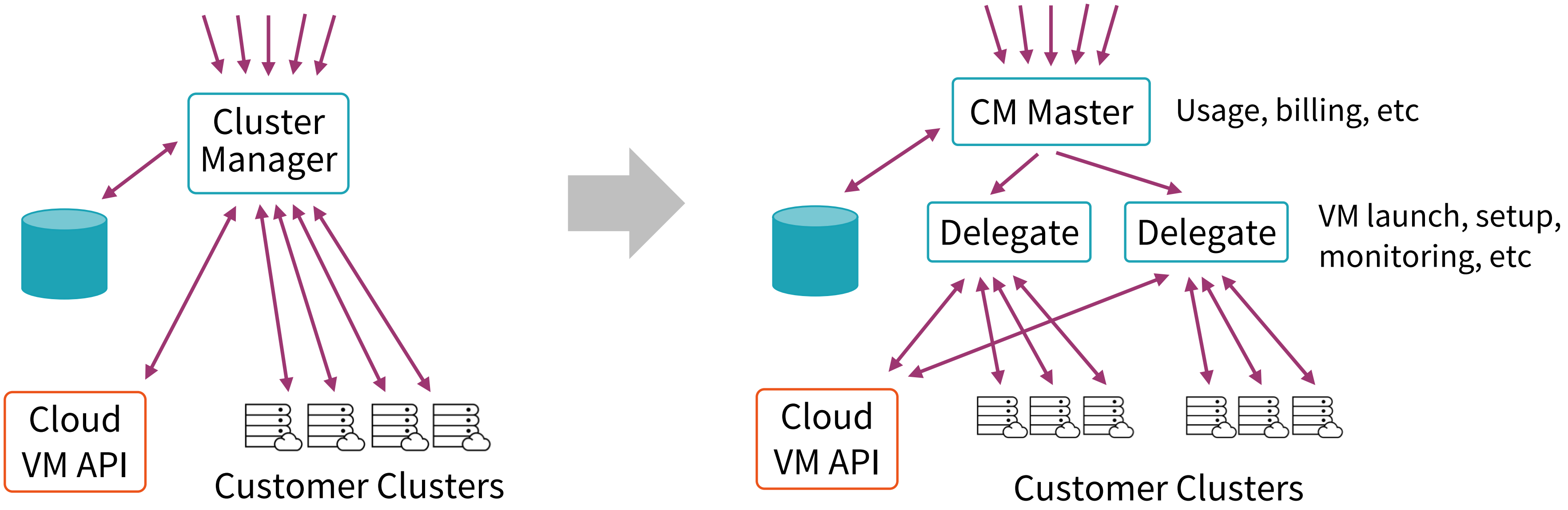
Manage changes through feature flags: fastest, safest way

Watch key metrics: most outages could be predicted from one of CPU load, memory load, DB load or thread pool exhaustion

Test pyramid: 70% unit tests, 20% integration, 10% end-to-end

Example: Cluster Manager

Cluster manager v1 / Cluster manager v2



Challenges in Control Planes

Fine-grained isolation *within* a service

Non-standard failure modes (e.g. network conn. exhaustion)

Transitioning between architectures

Four Lessons

1. What goes wrong in cloud systems?
2. Testing for scalability & stability
3. Developing control planes
4. Evolving big data systems for the cloud

Evolving Big Data Systems for the Cloud

MapReduce, Spark, etc were designed for on-premise datacenters

How can we evolve these leverage the benefits of the cloud?

- Availability, elasticity, scale, multitenancy, etc

Two examples from Databricks:

- Delta Lake: ACID on cloud object stores
- Cloudifying Apache Spark

Delta Lake Motivation

Cloud object stores (S3, Azure blob, etc) are the largest storage systems on the planet

- Unmatched availability, parallel I/O bandwidth, and cost-efficiency

Open source big data stack was designed for on-prem world

- Filesystem API for storage
- RDBMS for table metadata (Hive metastore)
- Other distributed systems, e.g. ZooKeeper

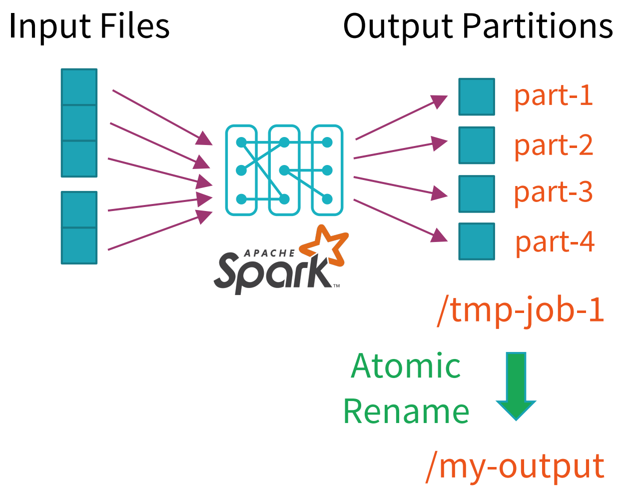
Stronger consistency model

Scale & management complexity

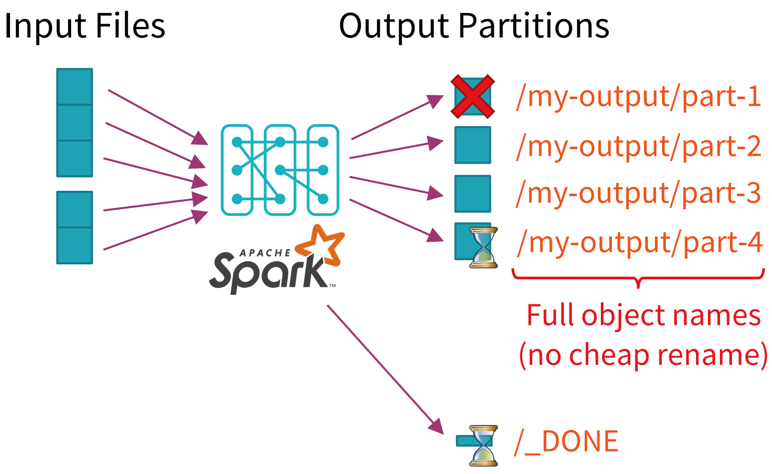
How can big data systems fully leverage cloud object stores?

Example: Atomic Parallel Writes

Spark on HDFS



Spark on S3 (Naïve)

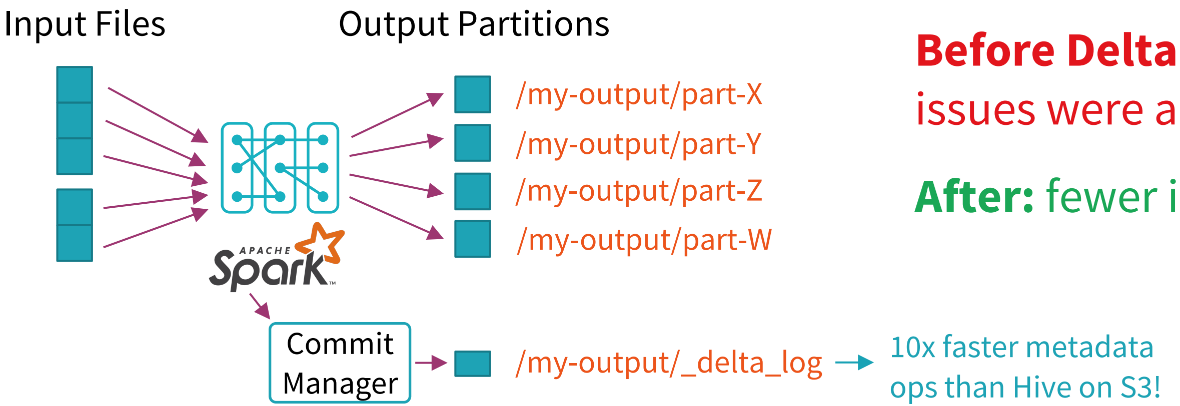


Real cases are harder (e.g. appending to a table)

Delta Lake Design

1. Track metadata that says *which* objects are part of a dataset

2. Store this metadata itself in a cloud object store
   - Write-ahead log in S3, compressed using Apache Parquet



Before Delta Lake: 50% of Spark support issues were about cloud storage

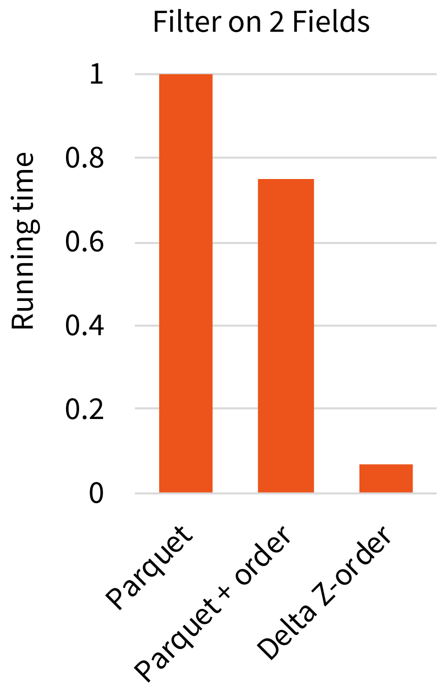
After: fewer issues, increased perf

https://delta.io

Major Benefits of Delta Lake

Once we had transactions over S3, we could build much more:

- UPSERT, DELETE, etc (GDPR)
- Caching
- Multidimensional indexing
- Audit logging
- Time travel
- Background optimization



Result: greatly simplified customers' data architectures

Other Cloud Features

Scheduler-integrated autoscaling for Apache Spark

Autoscaling local storage volumes

User isolation for high-concurrency Spark clusters

- Serverless experience for users inside an org
- Separate library envs, IAM roles, performance & fault isolation

Conclusion

The cloud is eating software by enabling much better products

- Self-managing, elastic, more reliable & scalable

But building cloud products is understudied and hard

- Come see what's involved in an internship!

Many opportunities, from service fabrics to cloud-native systems

We're hiring in SF, Amsterdam & Toronto: databricks.com/jobs