

# Dancing Elephants: Working with Object Storage in Apache Spark and Hive

Steve Loughran  
Sanjay Radia

April 2017



The screenshot shows the Microsoft Azure portal interface for creating a new Data Lake Store. The page title is 'New Data Lake Store'. The configuration fields are as follows:

- Name:** petastore (with a green checkmark icon)
- Subscription:** Visual Studio Enterprise (dropdown menu)
- Resource Group:** Create new (selected) / Use existing (radio buttons)
- Location:** North Europe (dropdown menu)
- Pricing:** 1 PB for 24,880 USD (dropdown menu, highlighted with a red circle)
- Encryption Settings:** Enabled (with a right arrow icon)
- Pin to dashboard:** ☐
- Buttons:** Create (blue button), Automation options (text link)

## Why?

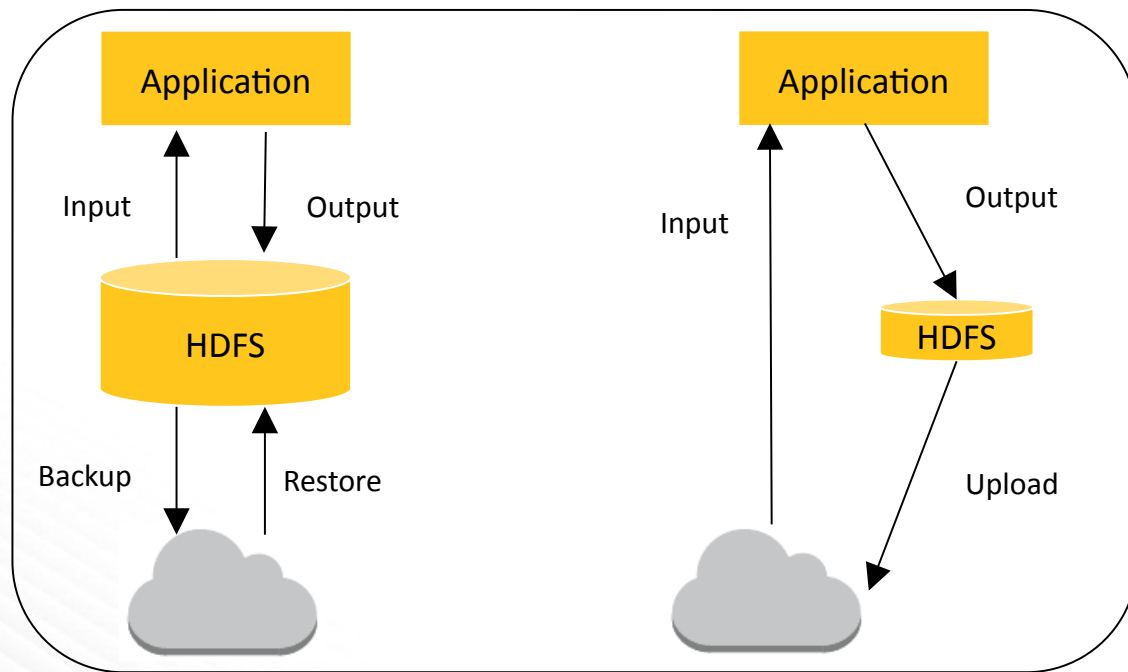
- No upfront hardware costs
- Data ingress for IoT, mobile apps
- Cost effective *if sufficiently agile*

# Object Stores are a key part of agile cloud applications

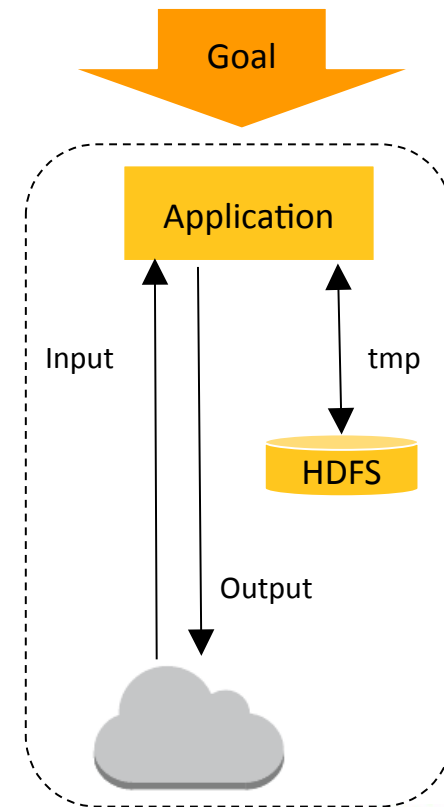
- ◆ It's the only persistent store for in-cloud clusters
  - ◆ Object stores are the source and final destination of work
  - ◆ Cross-application data storage
  - ◆ Asynchronous data exchange
  - ◆ External data sources
- } Also useful in physical clusters!

# Cloud Storage Integration: Evolution for Agility

Evolution towards cloud storage as the persistent Data Lake

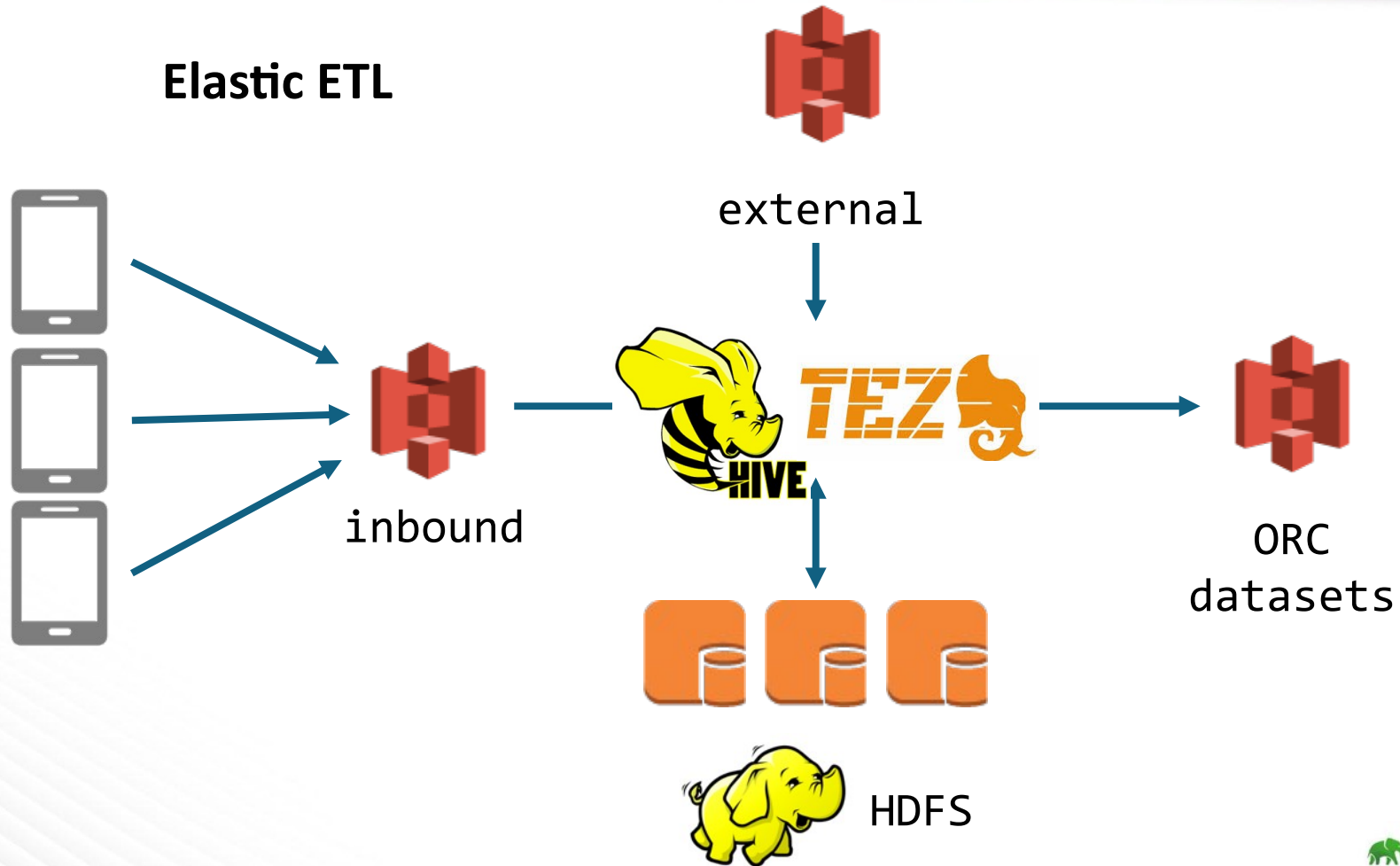


AWS -today

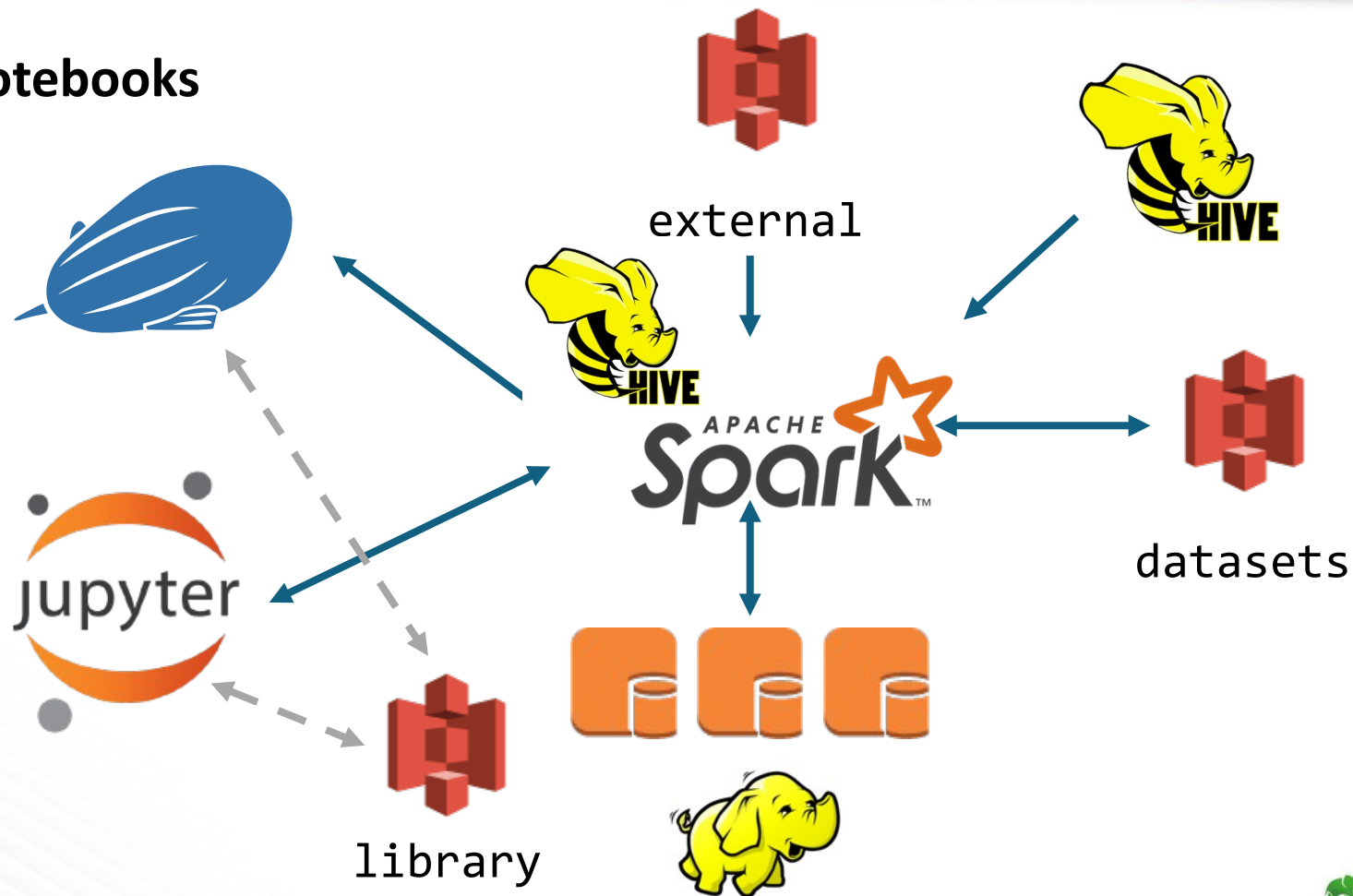


Azure

# Elastic ETL



# Notebooks





Streaming





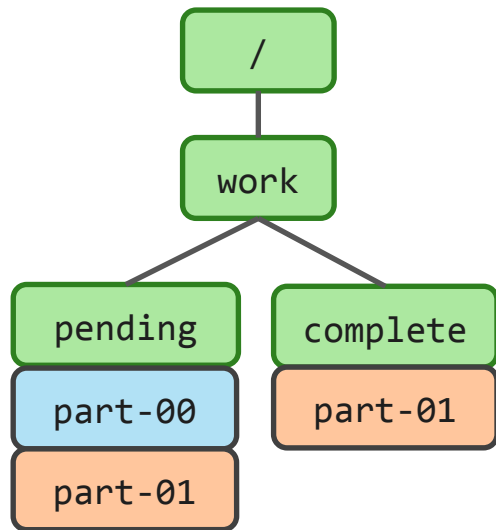
# Danger: Object stores are not filesystems\*

Cost & Geo-distribution over  
Consistency and Performance

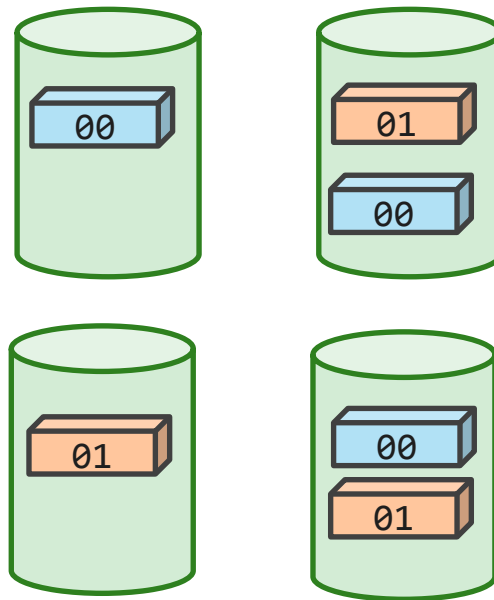
\*for more information, please re-read



# A Filesystem: Directories, Files → Data



`rename("/work/pending/part-01", "/work/complete")`



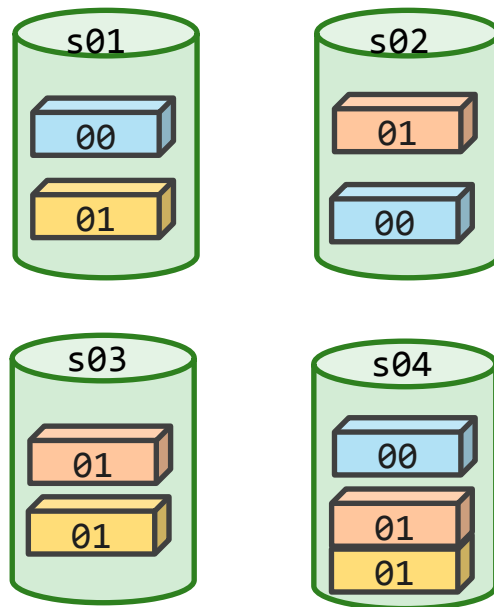
# Object Store: hash(name)⇒data

```
hash("/work/pending/part-00")  
["s01", "s02", "s04"]
```

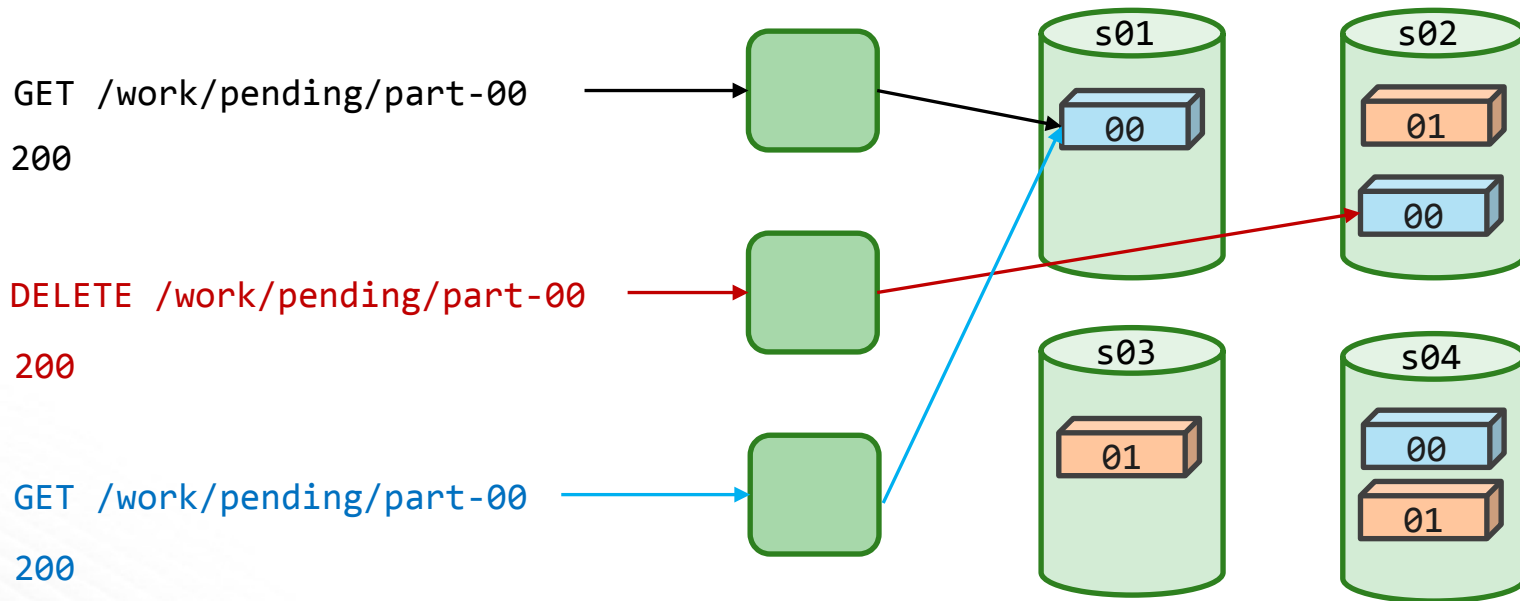
```
hash("/work/pending/part-01")  
["s02", "s03", "s04"]
```

```
copy("/work/pending/part-01",  
     "/work/complete/part01")
```

```
delete("/work/pending/part-01")
```



# Often: Eventually Consistent



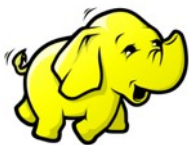
# The dangers of Eventual Consistency

- ◆ Temp Data leftovers
- ◆ List inconsistency means new data may not be visible
- ◆ Lack of atomic rename( ) can leave output directories inconsistent

*You can get bad data and not even notice*



`org.apache.hadoop.fs.FileSystem`



hdfs



wasb



s3a



swift

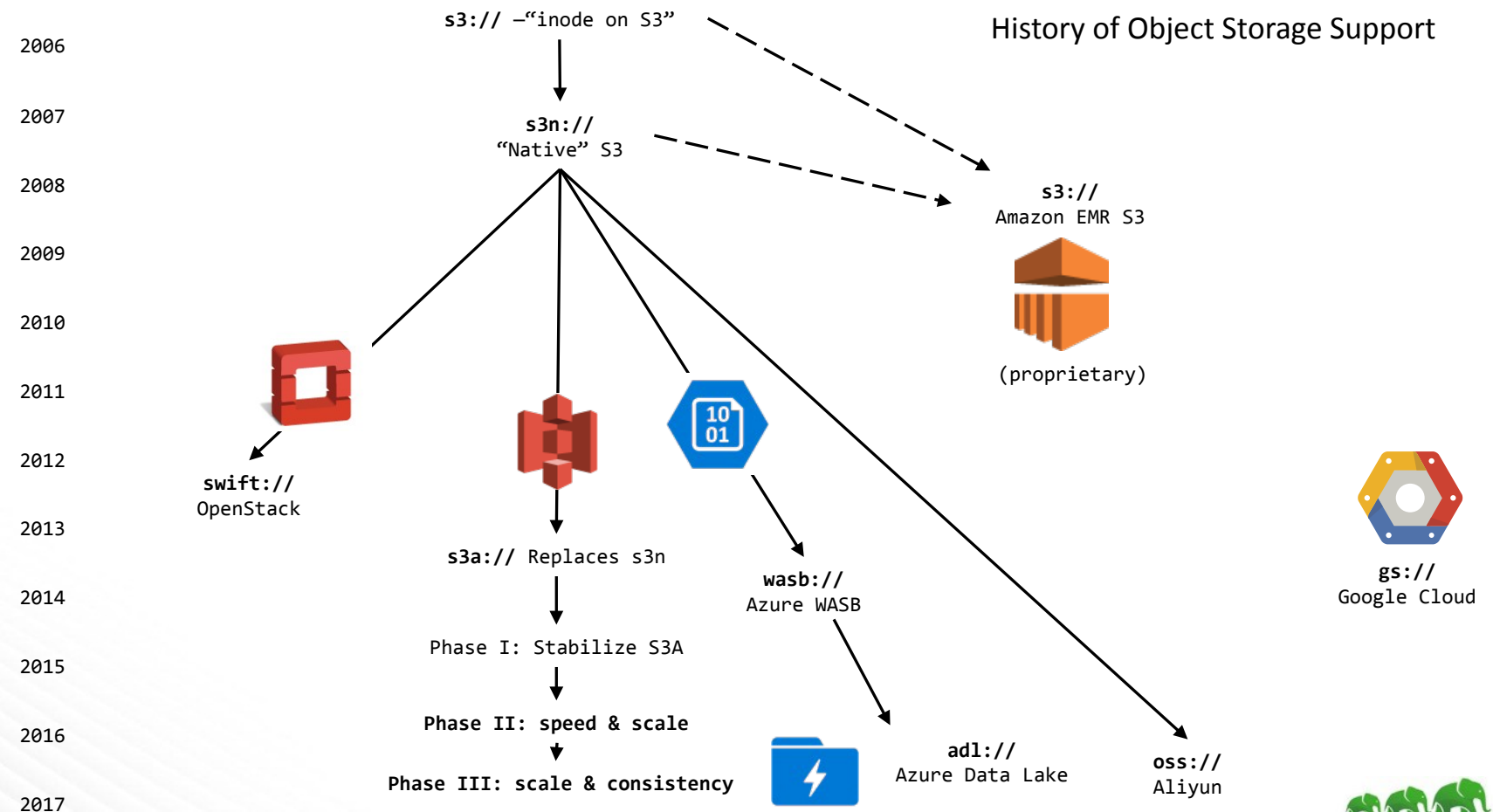


adl



gs

# History of Object Storage Support





# Make Apache Hadoop at home in the cloud

Step 1: Hadoop runs great on Azure ✓✓  
Step 2: Beat EMR on EC2 ✓

# Problem: S3 Analytics is too slow/broken

1. Analyze benchmarks and bug-reports
2. Fix Read path for Columnar Data
3. Fix Write path
4. Improve query partitioning
5. The Commitment Problem

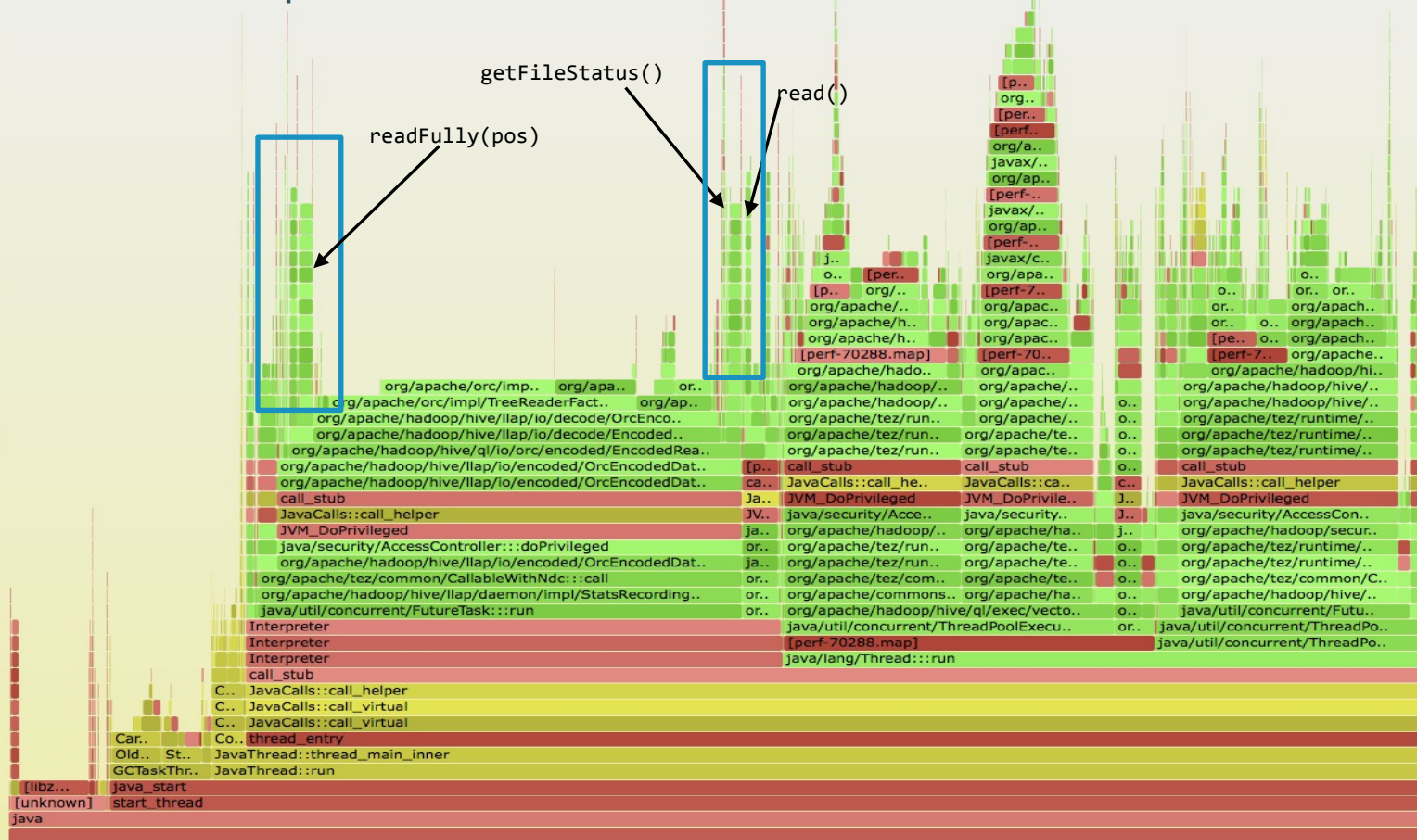


# LLAP (single node) on AWS

## TPC-DS queries at 200 GB scale

Flame Graph

Search



## HDP 2.6/Hadoop 2.8 transforms I/O performance!

// forward seek by skipping stream

`fs.s3a.readahead.range=256K`

// faster backward seek for Columnar Storage

`fs.s3a.experimental.input.fadvise=random`

// enhanced data upload

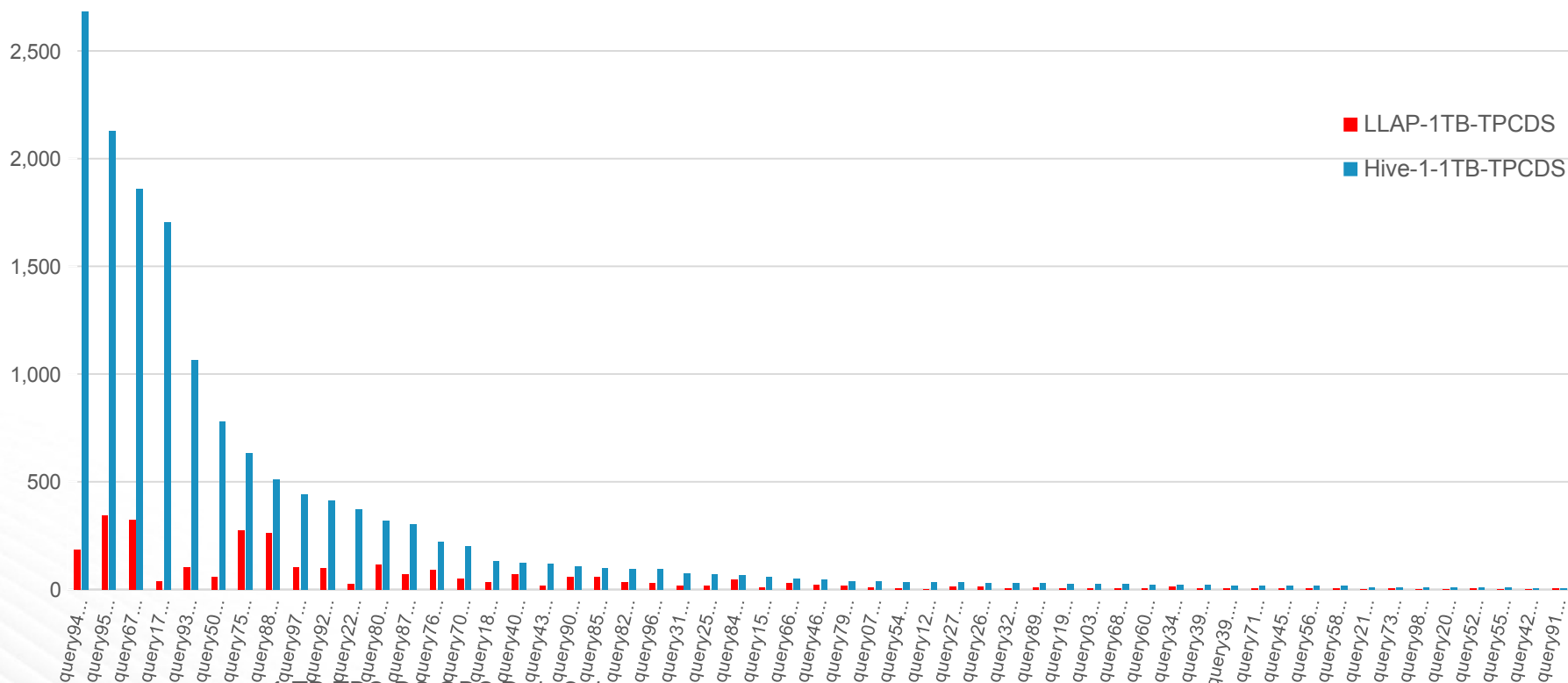
`fs.s3a.fast.output.enabled=true`

—see *HADOOP-11694* for lots more!

**benchmarks !=  
your queries  
your data  
your VMs  
your directory tree**

**...but we think we've made a good start**

## S3 Data Source 1TB TPCDS LLAP- vs Hive 1.x:



1 TB TPC-DS ORC DataSet

3 x i2x4x Large (16 CPU x 122 GB RAM x 4 SSD)



# Apache Spark

Object store work applies  
Needs tuning  
Commits to S3 "trouble"



# spark-default.conf

```
spark.hadoop.fs.s3a.readahead.range 256K
spark.hadoop.fs.s3a.block.size 32M
spark.hadoop.fs.s3a.fast.output.enabled true
spark.hadoop.fs.s3a.experimental.input.fadvise random
spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version 2
spark.hadoop.parquet.enable.summary-metadata false
```

```
spark.sql.orc.filterPushdown true
spark.sql.orc.splits.include.file.footer true
spark.sql.orc.cache.stripe.details.size 10000
spark.sql.hive.metastorePartitionPruning true
```

```
spark.sql.parquet.filterPushdown true
spark.sql.parquet.mergeSchema false
```

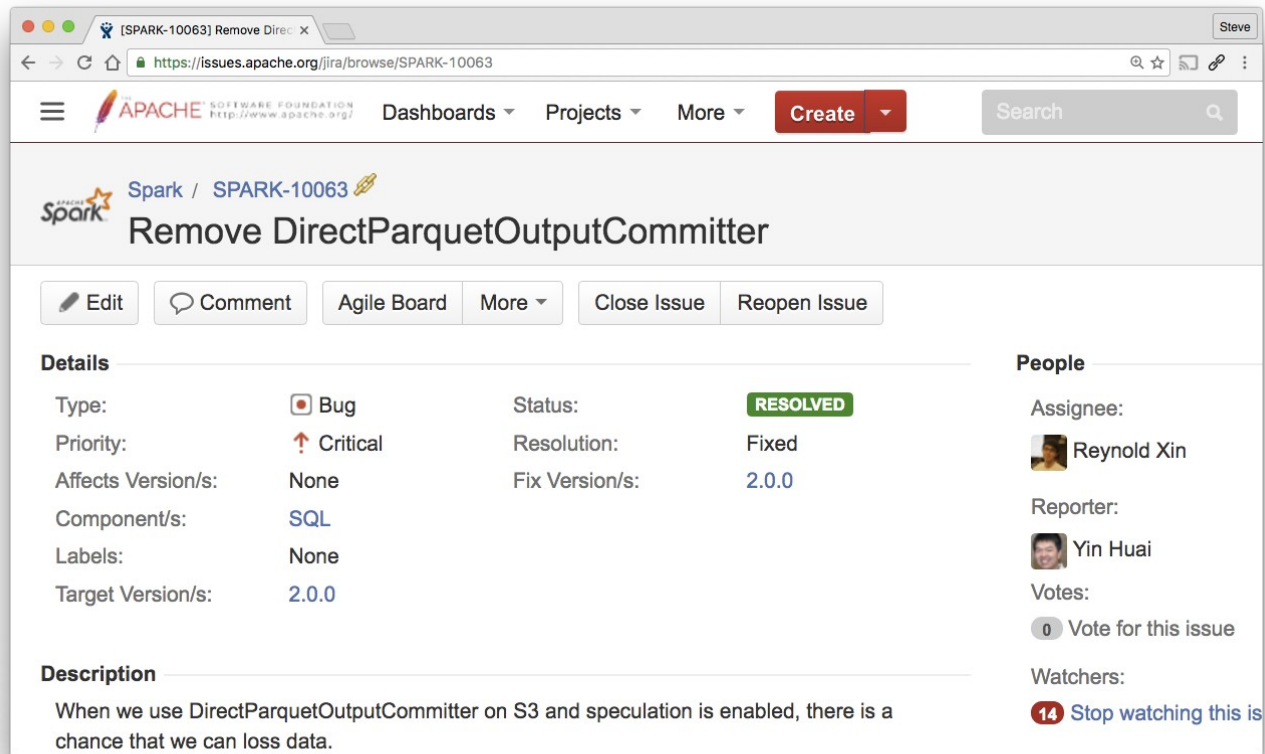
# Hive settings

```
fs.s3a.readahead.range 256K  
fs.s3a.block.size 32M  
fs.s3a.fast.output.enabled true  
fs.s3a.experimental.input.fadvise random  
mapreduce.fileoutputcommitter.algorithm.version 2  
parquet.enable.summary-metadata false
```

# The S3 Commitment Problem

- `rename()` depended upon for atomic transaction
- Time to `copy()` + `delete()` proportional to data \* files
- Compared to Azure Storage, S3 is slow (6-10+ MB/s)
- Intermediate data may be visible
- Failures leave storage in unknown state

# Spark's Direct Output Committer? Risk of Corruption of data



The screenshot shows the Apache JIRA issue page for SPARK-10063. The browser address bar shows the URL <https://issues.apache.org/jira/browse/SPARK-10063>. The page header includes the Apache Software Foundation logo and navigation links: Dashboards, Projects, More, Create, and a Search bar. The issue title is "Remove DirectParquetOutputCommitter". Below the title are buttons for Edit, Comment, Agile Board, More, Close Issue, and Reopen Issue. The "Details" section lists the following information:

Field	Value
Type:	Bug
Priority:	Critical
Affects Version/s:	None
Component/s:	SQL
Labels:	None
Target Version/s:	2.0.0
Status:	RESOLVED
Resolution:	Fixed
Fix Version/s:	2.0.0

The "Description" section contains the text: "When we use DirectParquetOutputCommitter on S3 and speculation is enabled, there is a chance that we can loss data."

The "People" section lists the Assignee: Reynold Xin, Reporter: Yin Huai, and Votes: 0. It also shows a "Watchers" section with 14 people watching the issue.

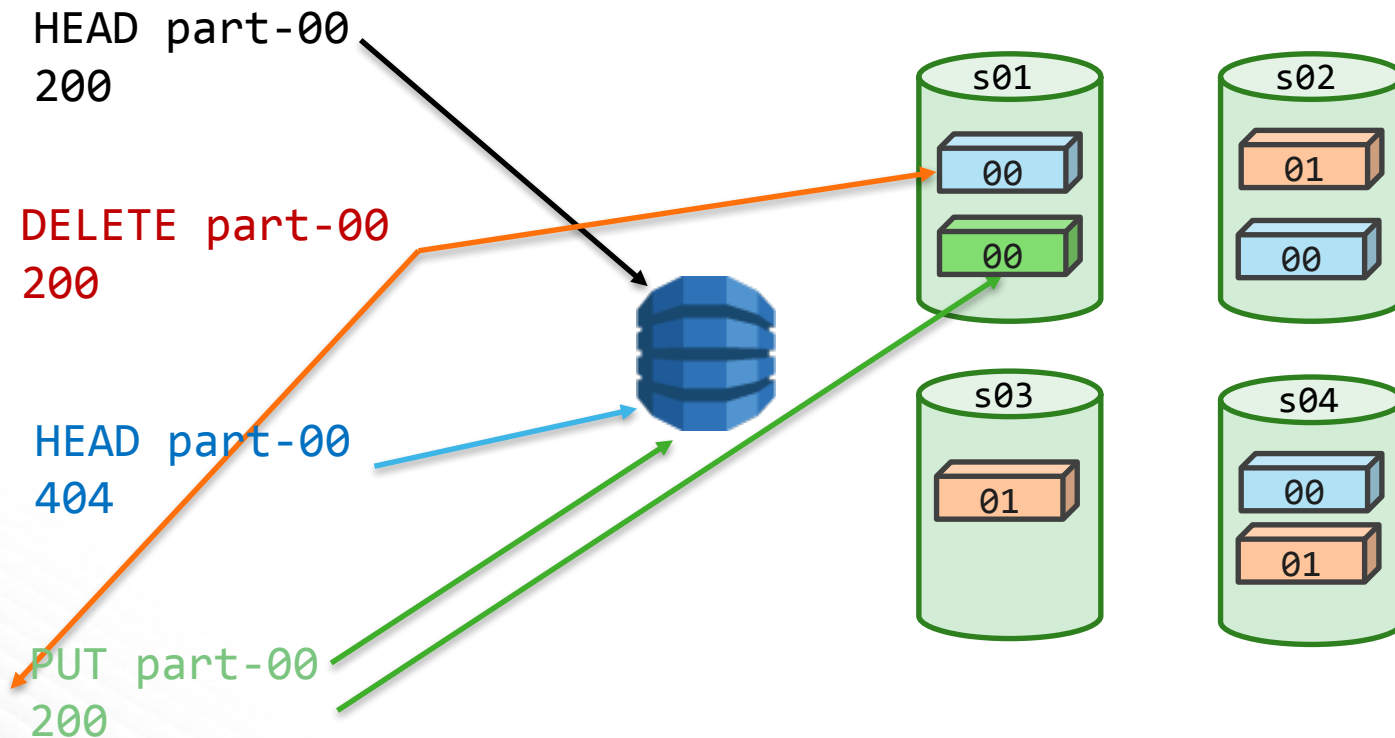
# S3guard

## Fast, consistent S3 metadata

**HADOOP-13445**



# DynamoDB as fast, consistent metadata store



# Netflix Staging Committer

1. Saves output to local files `file://`
2. Task commit: upload to S3A as multipart PUT —*but do not complete it*
3. Job committer completes all uploads from successful tasks; cancels others.

Outcome:

- ◆ No work visible until job is committed
- ◆ Task commit time = data/bandwidth
- ◆ Job commit time = POST \* #files

# Availability

- Read + Write in HDP 2.6 and Apache Hadoop 2.8
- S3Guard: preview of DDB integration *soon*
- Zero-rename commit: *work in progress*

*Look to HDCloud for the latest work!*

# Big thanks to:

Rajesh Balamohan

Mingliang Liu

Chris Nauroth

Dominik Bialek

Ram Venkatesh

Everyone in QE, RE

+ everyone who reviewed/tested, patches and  
added their own, filed bug reports and measured  
performance

# Questions?

stevel@hortonworks.com  
sanjay@hortonworks.com

@steveloughran  
@srr

