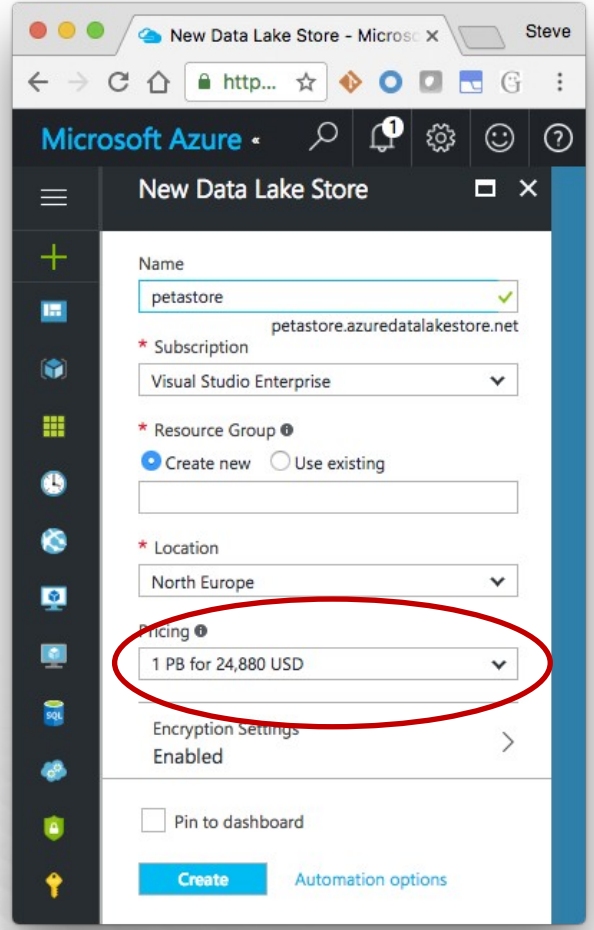
Dancing Elephants: Working with Object Storage in Apache Spark and Hive

Steve Loughran
Sanjay Radia

April 2017



Why?

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

HORTONWORKS

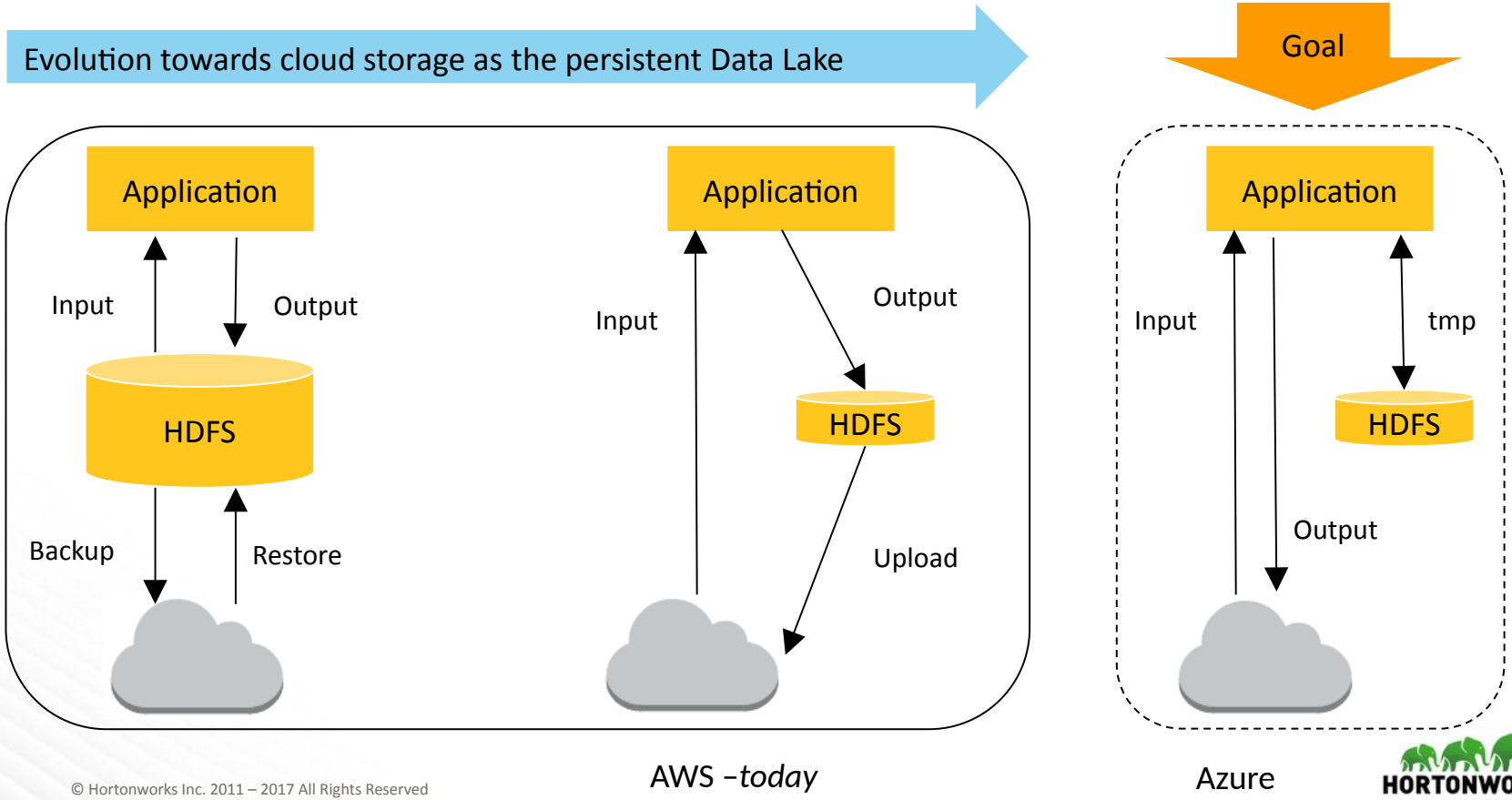
Object Stores are a key part of agile cloud applications

- 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!

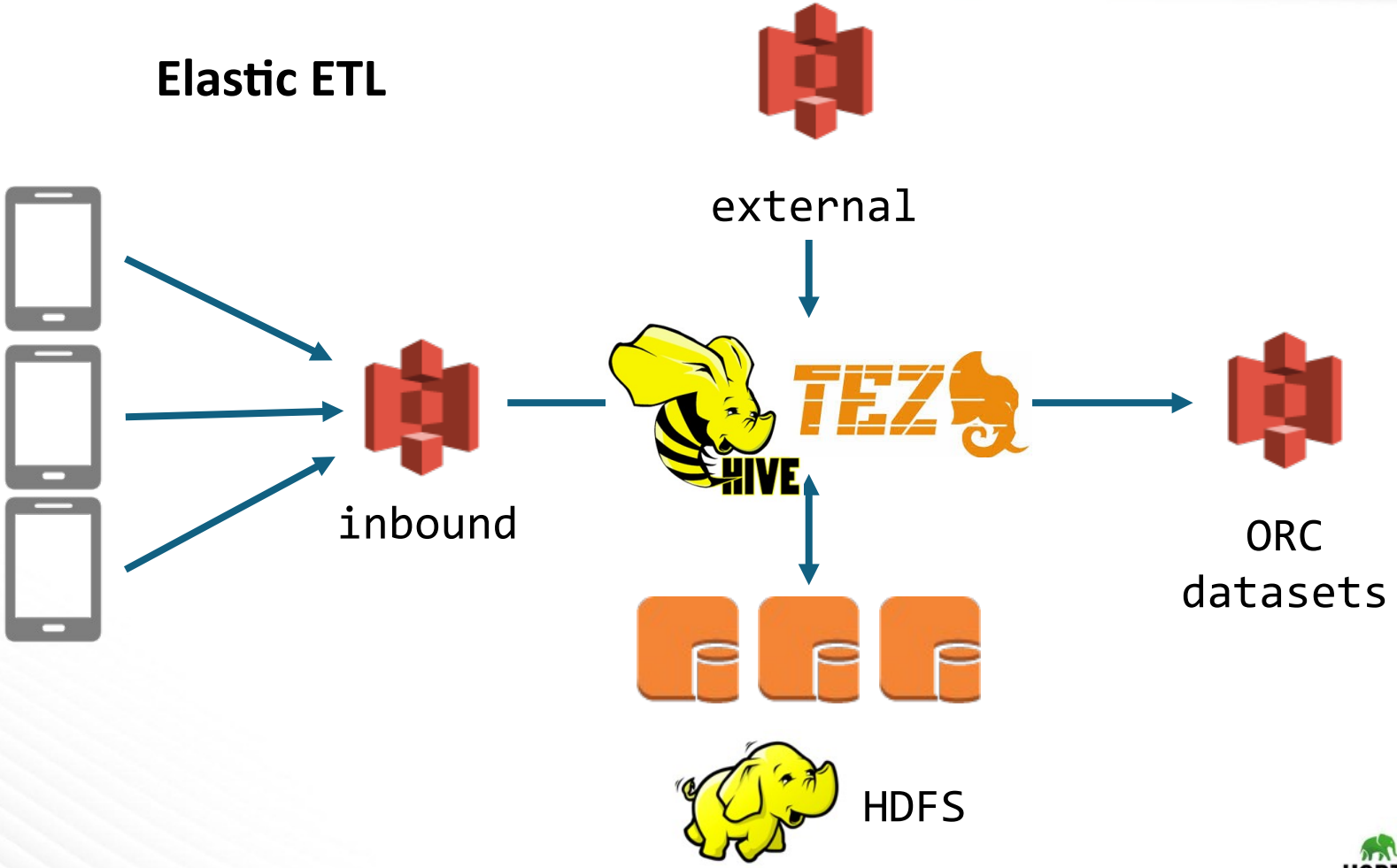
HORTONWORKS

Cloud Storage Integration: Evolution for Agility



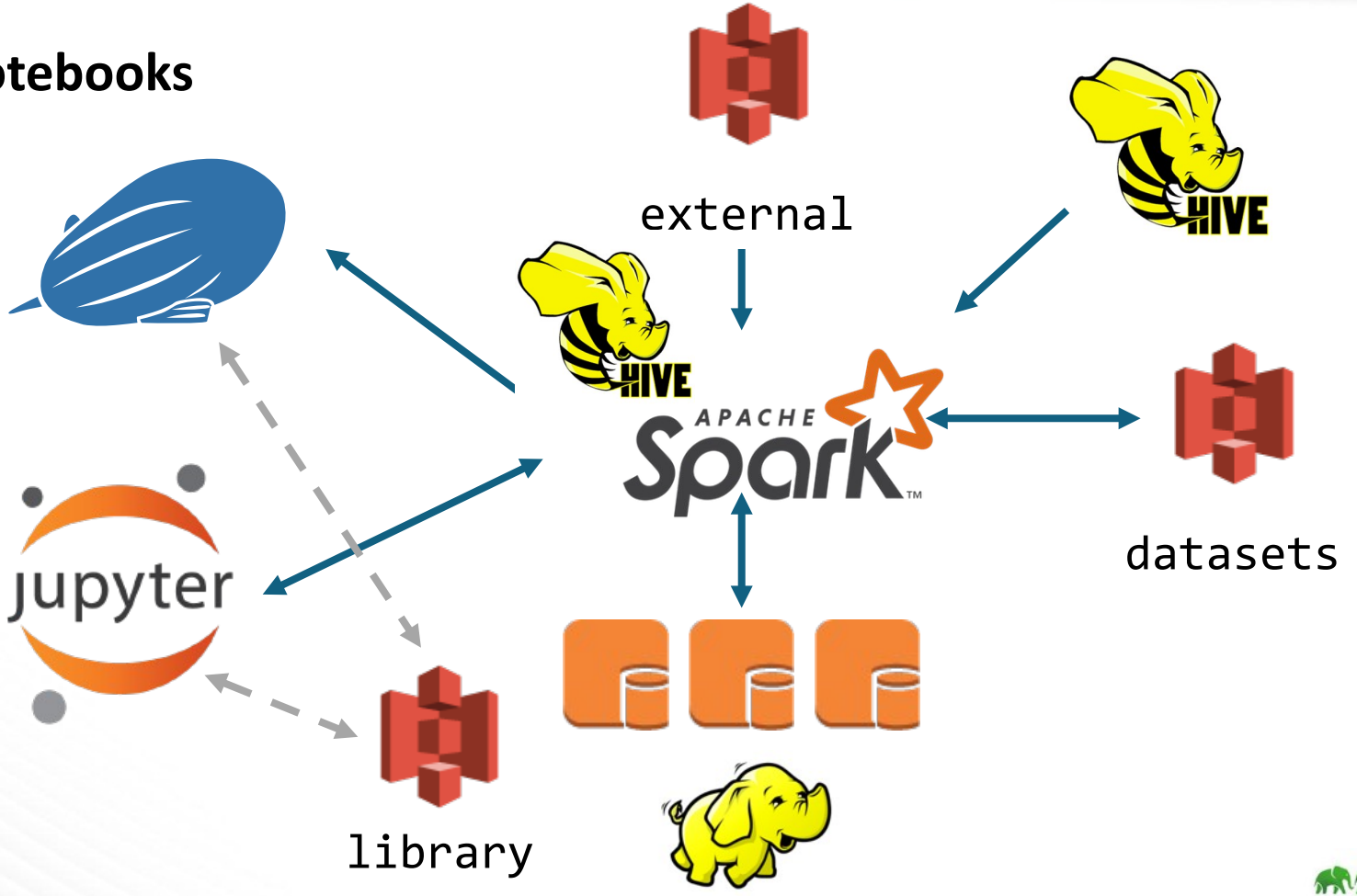
HORTONWORKS

Elastic ETL



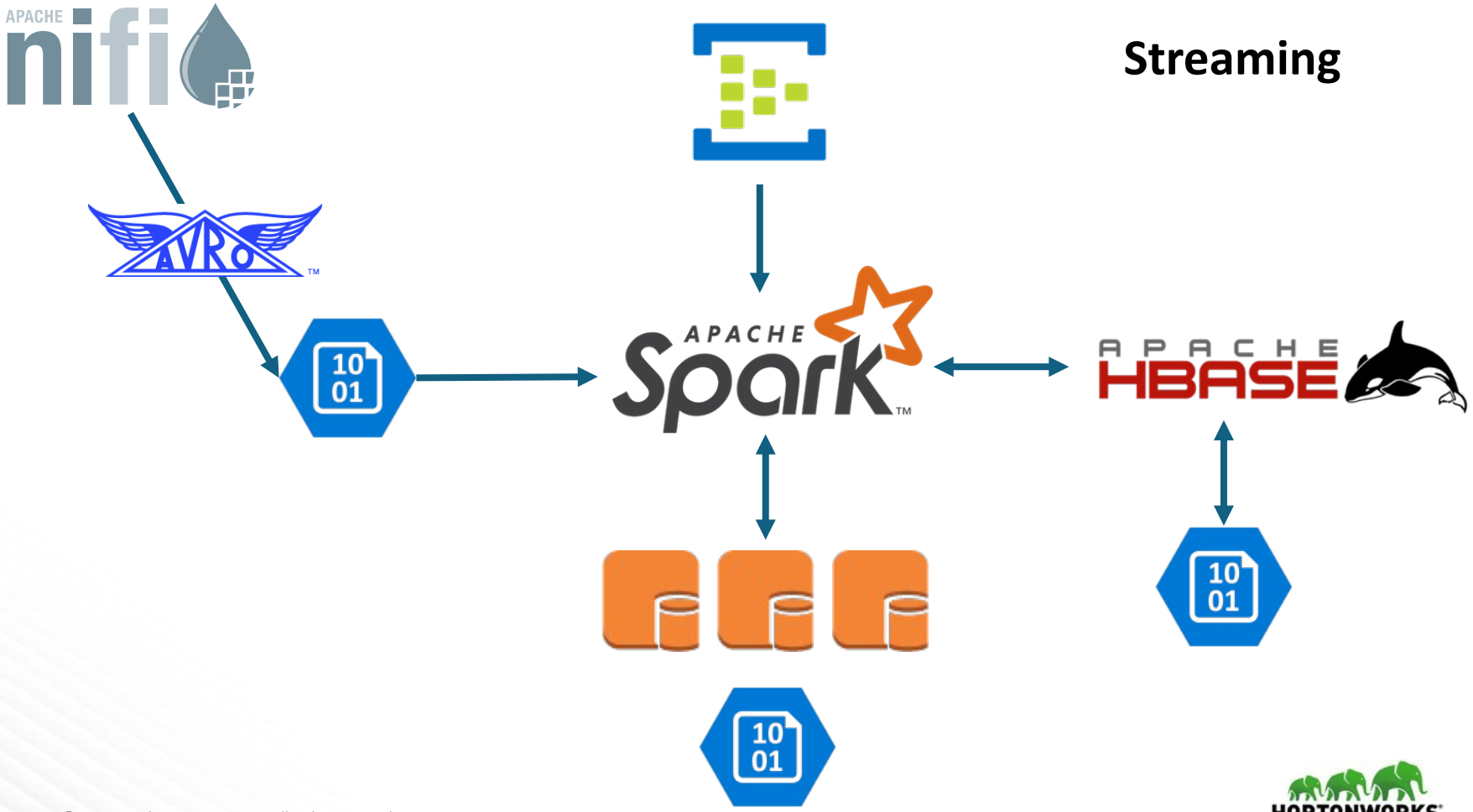
HORTONWORKS

Notebooks



HORTONWORKS

Streaming



HORTONWORKS

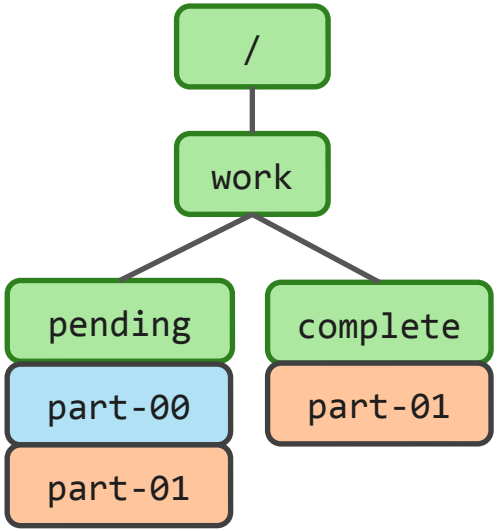
Danger:
Object stores are not filesystems*

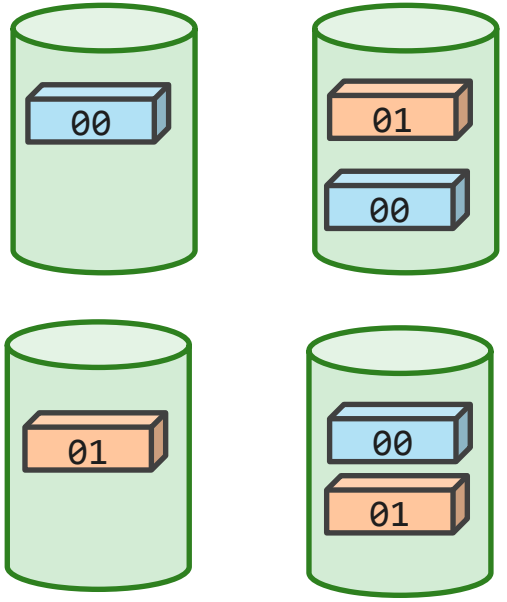
Cost & Geo-distribution over Consistency and Performance

*for more information, please re-read

HORTONWORKS®
POWERING THE FUTURE OF DATA™

A Filesystem: Directories, Files → Data



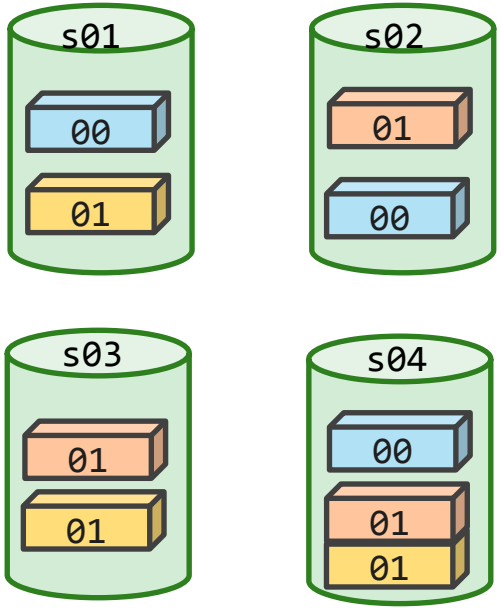



```
rename("/work/pending/part-01", "/work/complete")
```


HORTONWORKS

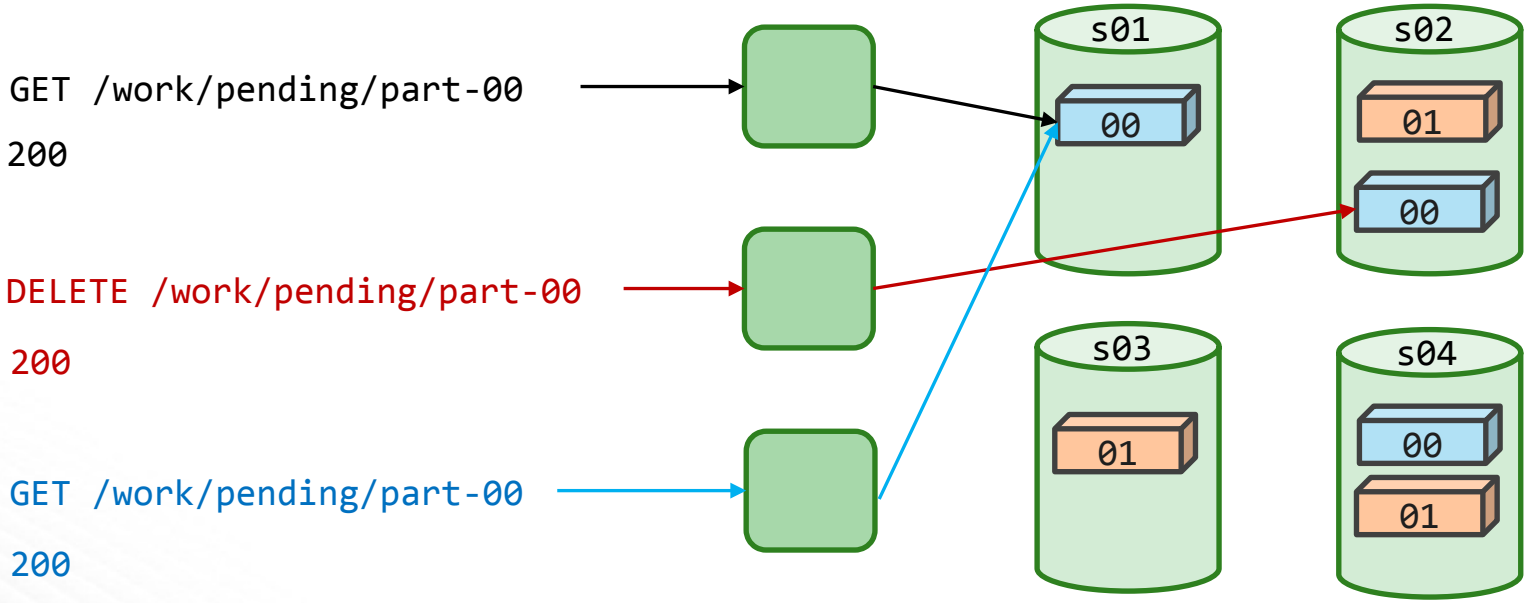
Object Store: hash(name)⇒data

```
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")
```



HORTONWORKS

Often: Eventually Consistent



HORTONWORKS

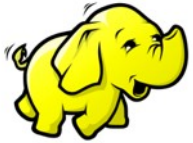
The dangers of Eventual Consistency

- 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

HORTONWORKS

`org.apache.hadoop.fs.FileSystem`



`hdfs`

`wasb`

`s3a`

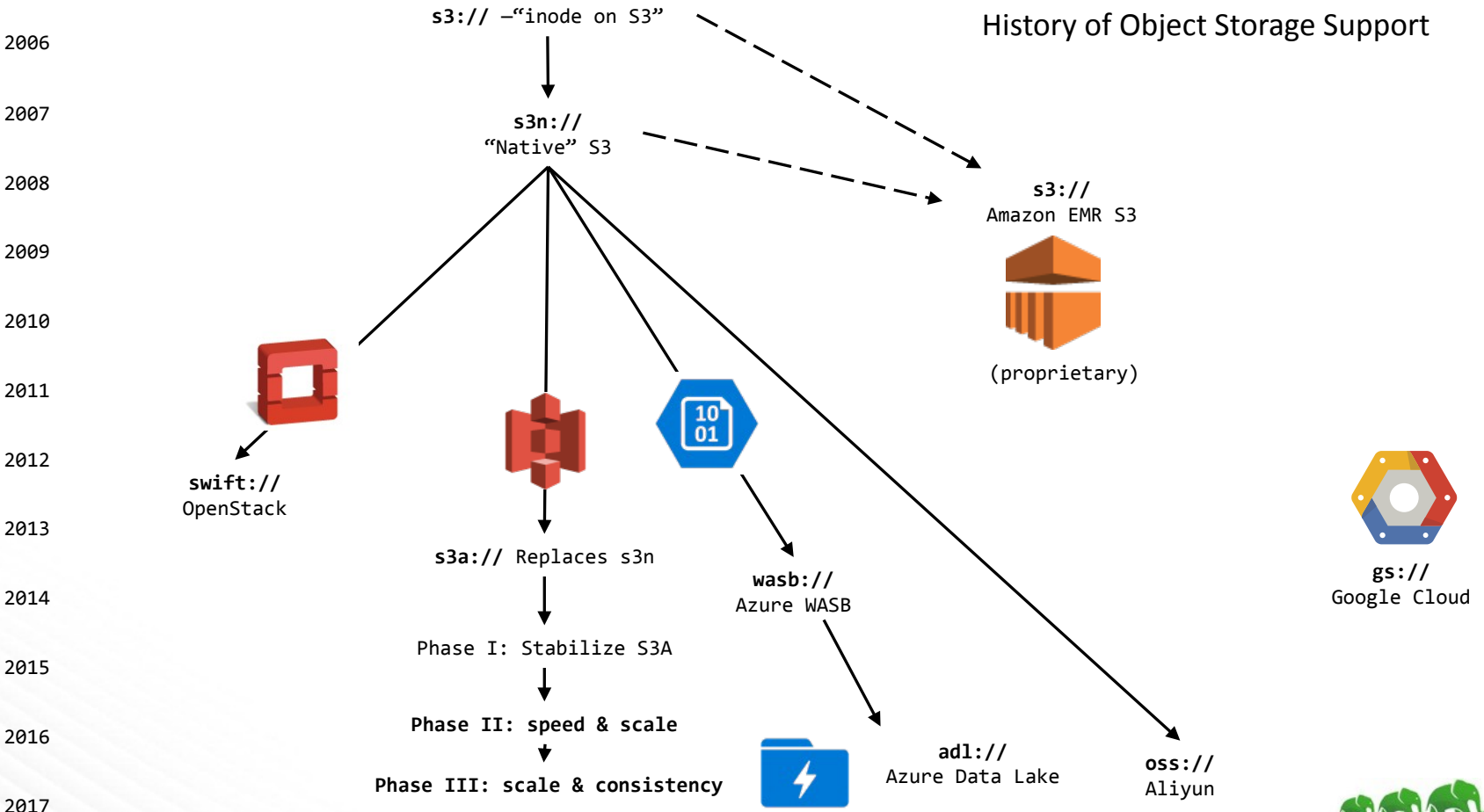
`swift`

`adl`

`gs`

HORTONWORKS

History of Object Storage Support



HORTONWORKS

Make Apache Hadoop at home in the cloud

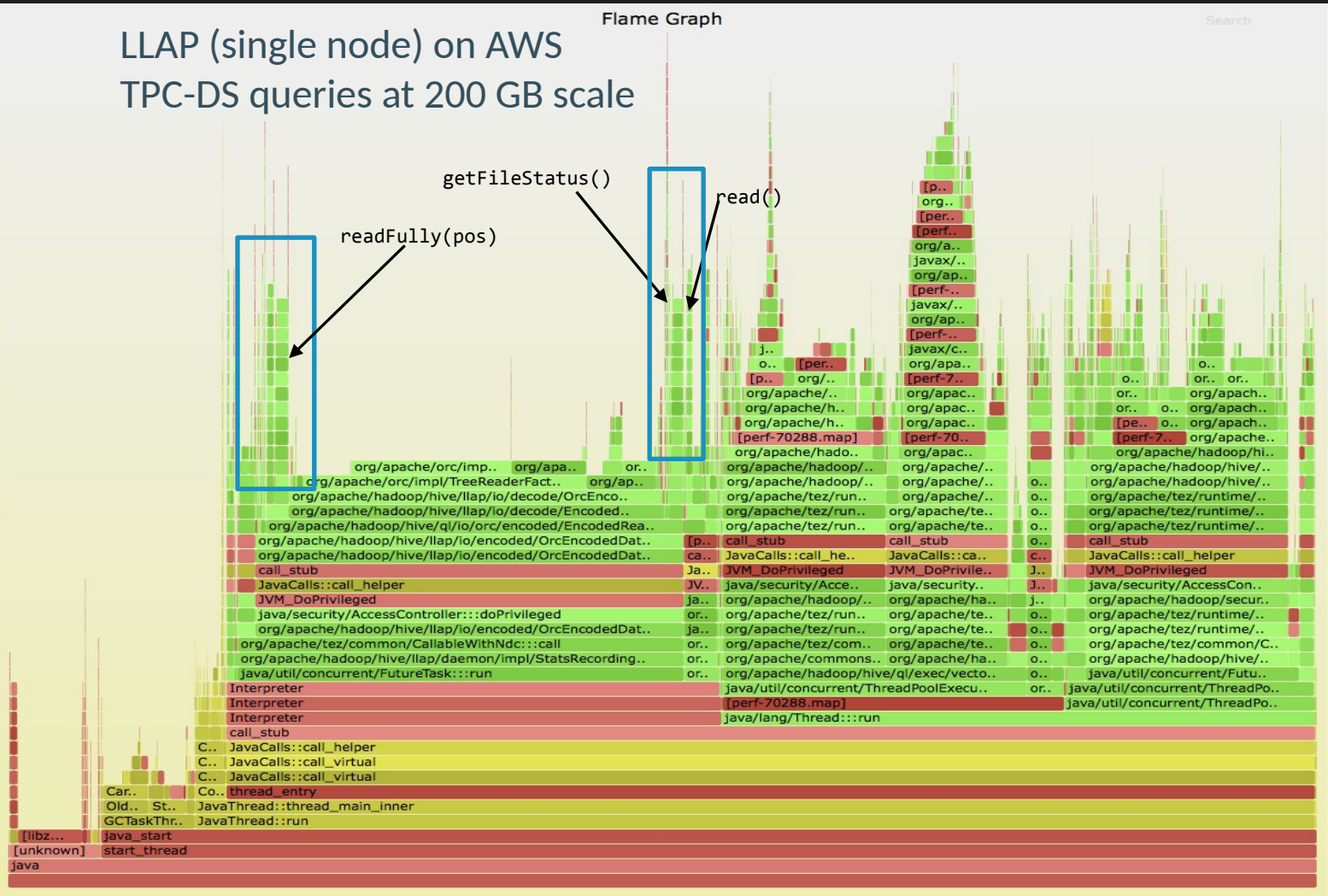
Step 1: Hadoop runs great on Azure ✔✔
Step 2: Beat EMR on EC2 ✔

HORTONWORKS®
POWERING THE FUTURE OF DATA™

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



HDP 2.6/Hadoop 2.8 transforms I/O performance!

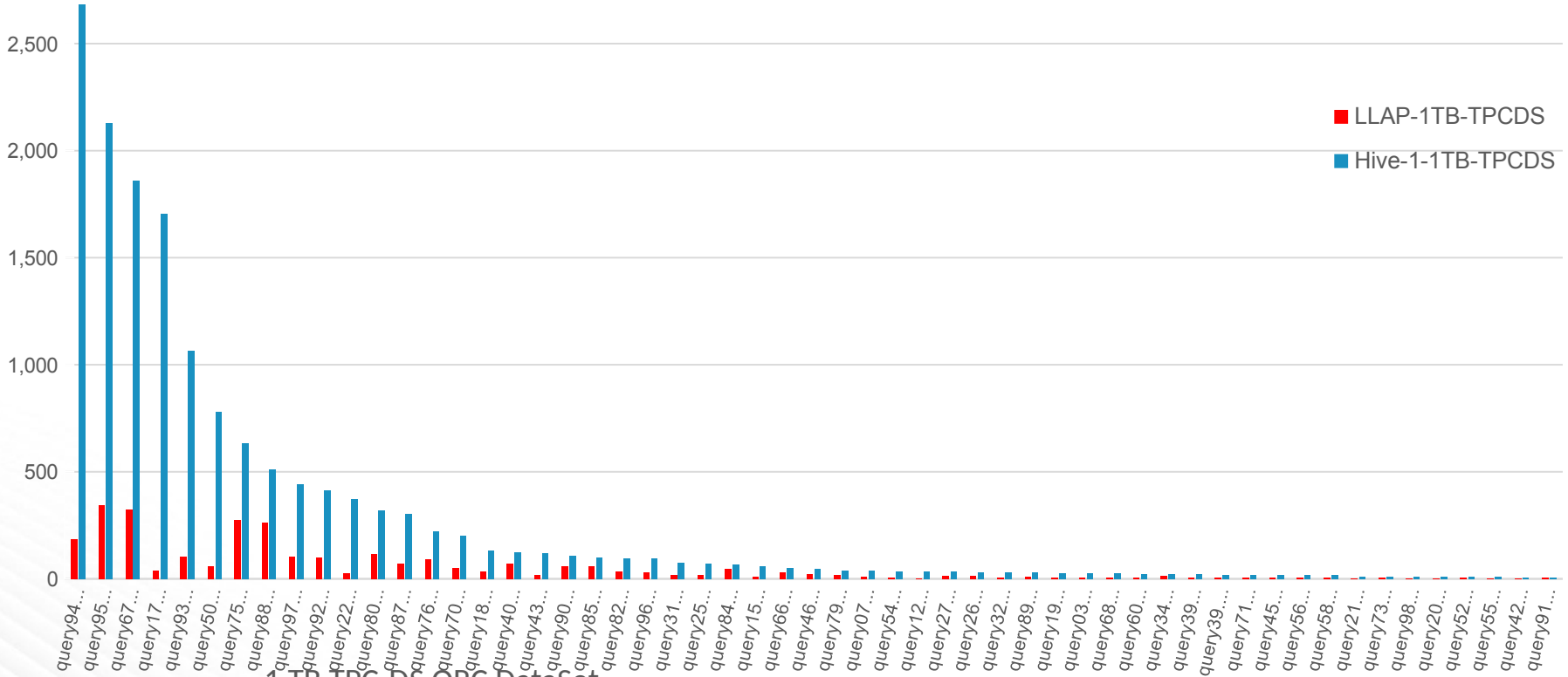
```
// forward seek by skipping stream
fs.s3a.readahead.range=256K

// 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!

HORTONWORKS


benchmarks !=
your queries
your data
your VMs
your directory tree
…but we think we've made a good start
HORTONWORKS
POWERING THE FUTURE OF DATA

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)

HORTONWORKS

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
```


HORTONWORKS

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
```

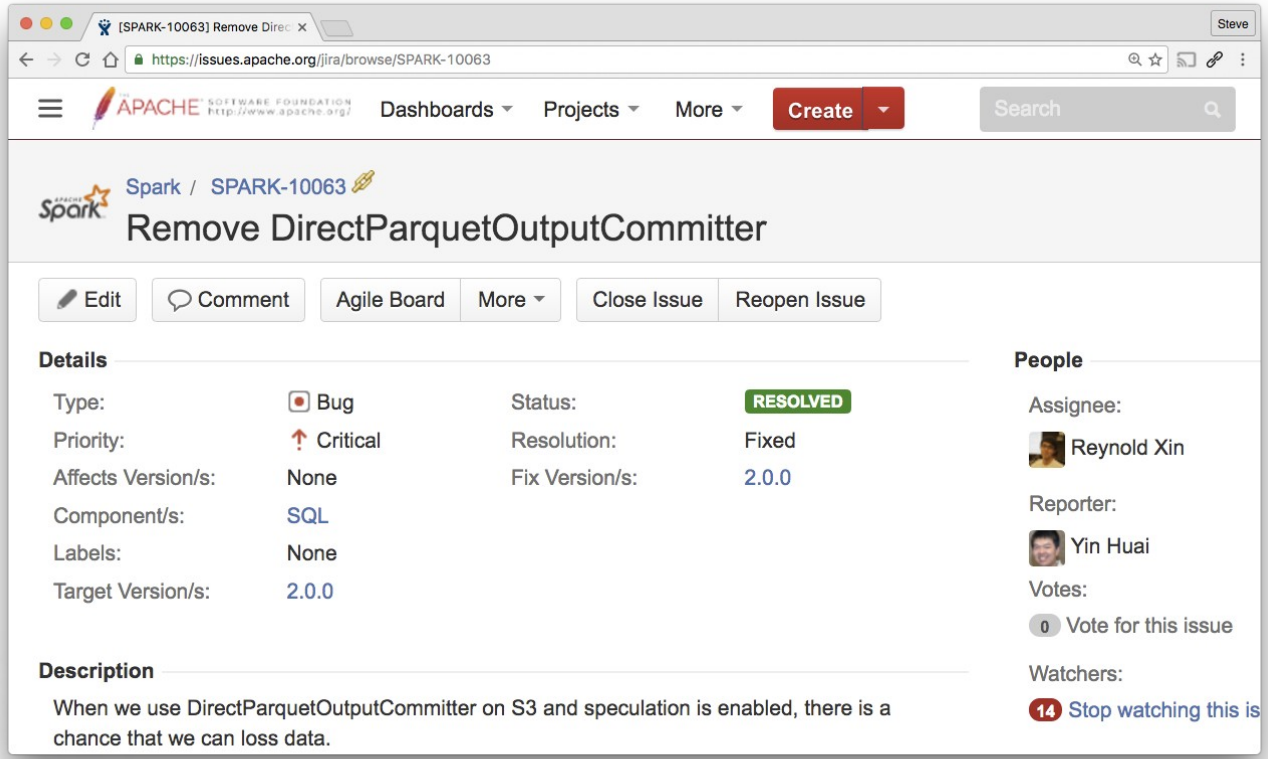

HORTONWORKS

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

HORTONWORKS

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



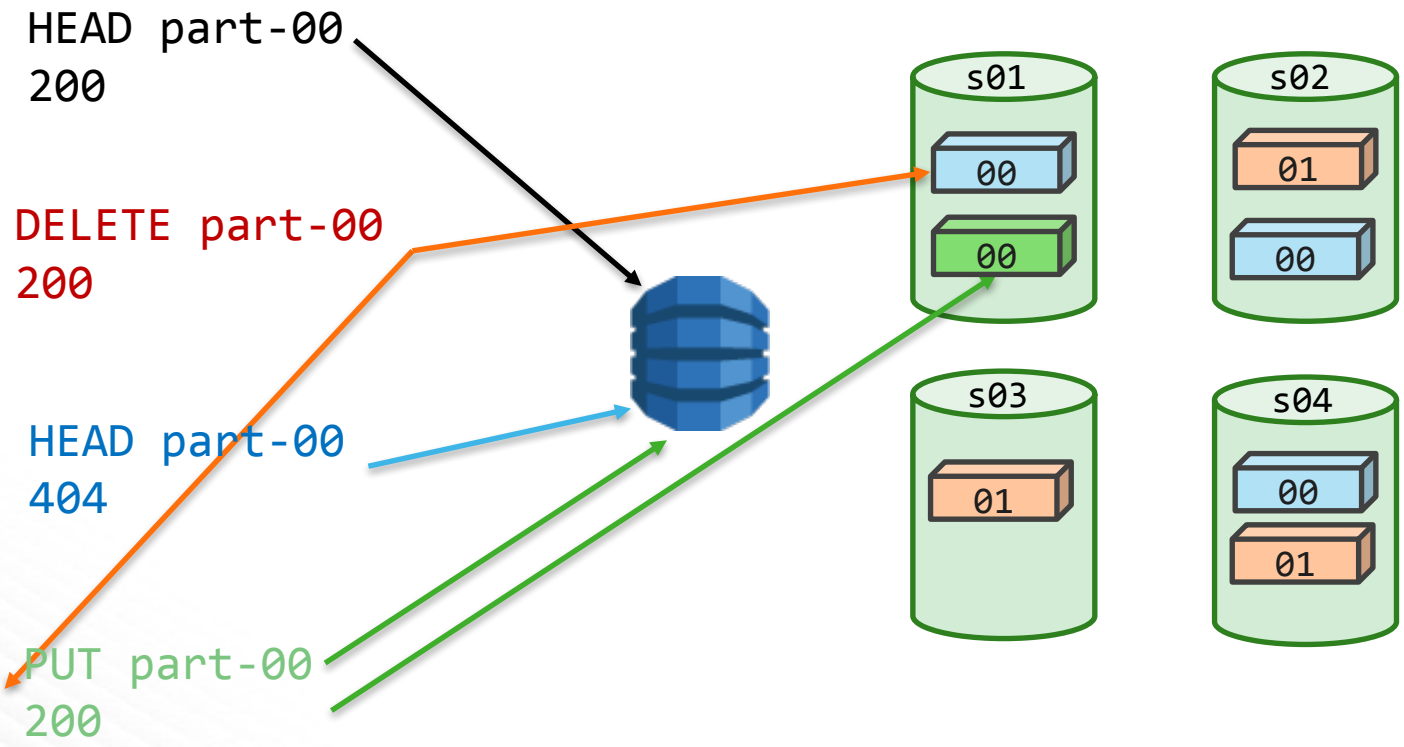
HORTONWORKS

S3guard
Fast, consistent S3 metadata

HADOOP-13445

HORTONWORKS
POWERING THE FUTURE OF DATA

DynamoDB as fast, consistent metadata store



HORTONWORKS

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 commited
- Task commit time = `data/bandwidth`
- Job commit time = POST * `#files`

HORTONWORKS

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

HORTONWORKS®
POWERING THE FUTURE OF DATA™

Questions?

stevel@hortonworks.com @steveloughran
sanjay@hortonworks.com @srr

HORTONWORKS
POWERING THE FUTURE OF DATA