

Apache Spark and Object Stores —What you need to know

Steve Loughran

stevel@hortonworks.com

@steveloughran

February 2017



Steve Loughran,
Hadoop committer, PMC member, ...



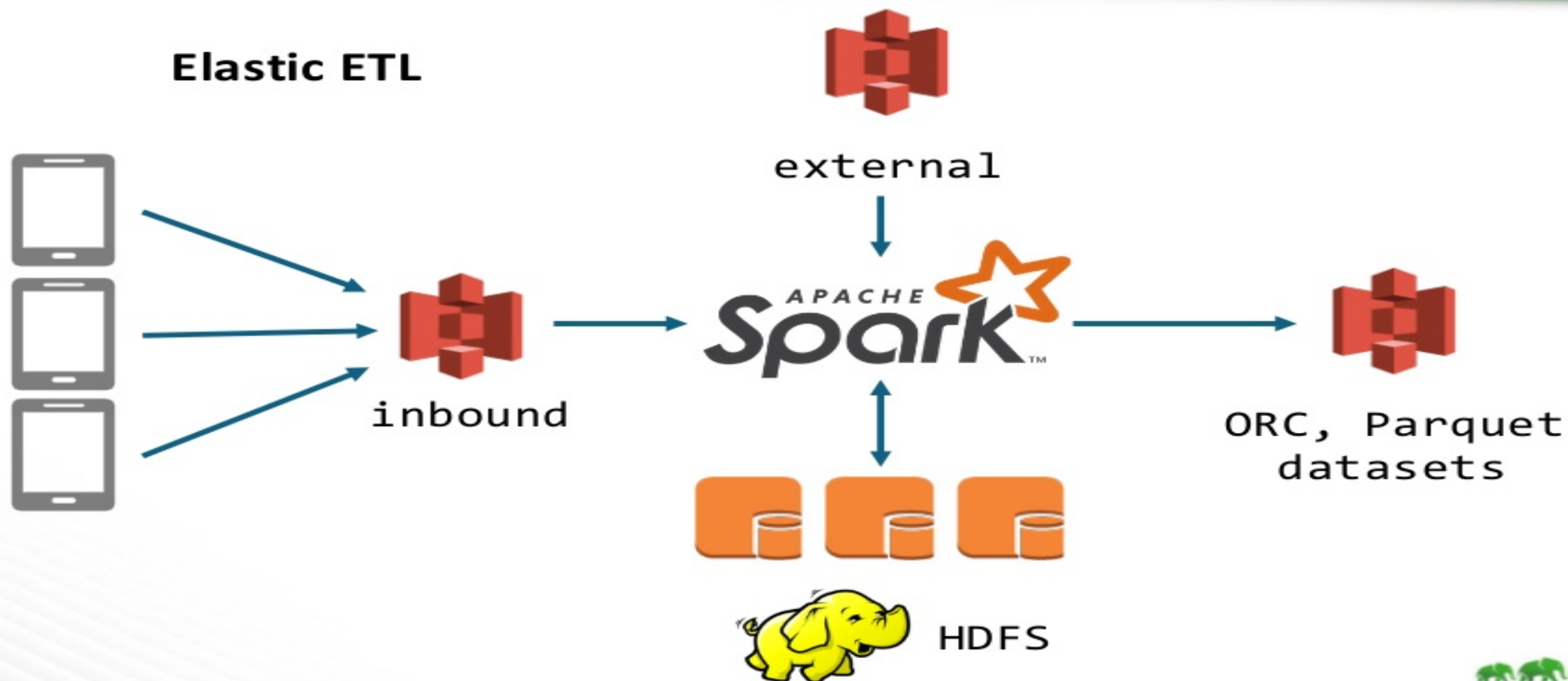
Rajesh Balamohan
Tez Committer, PMC Member



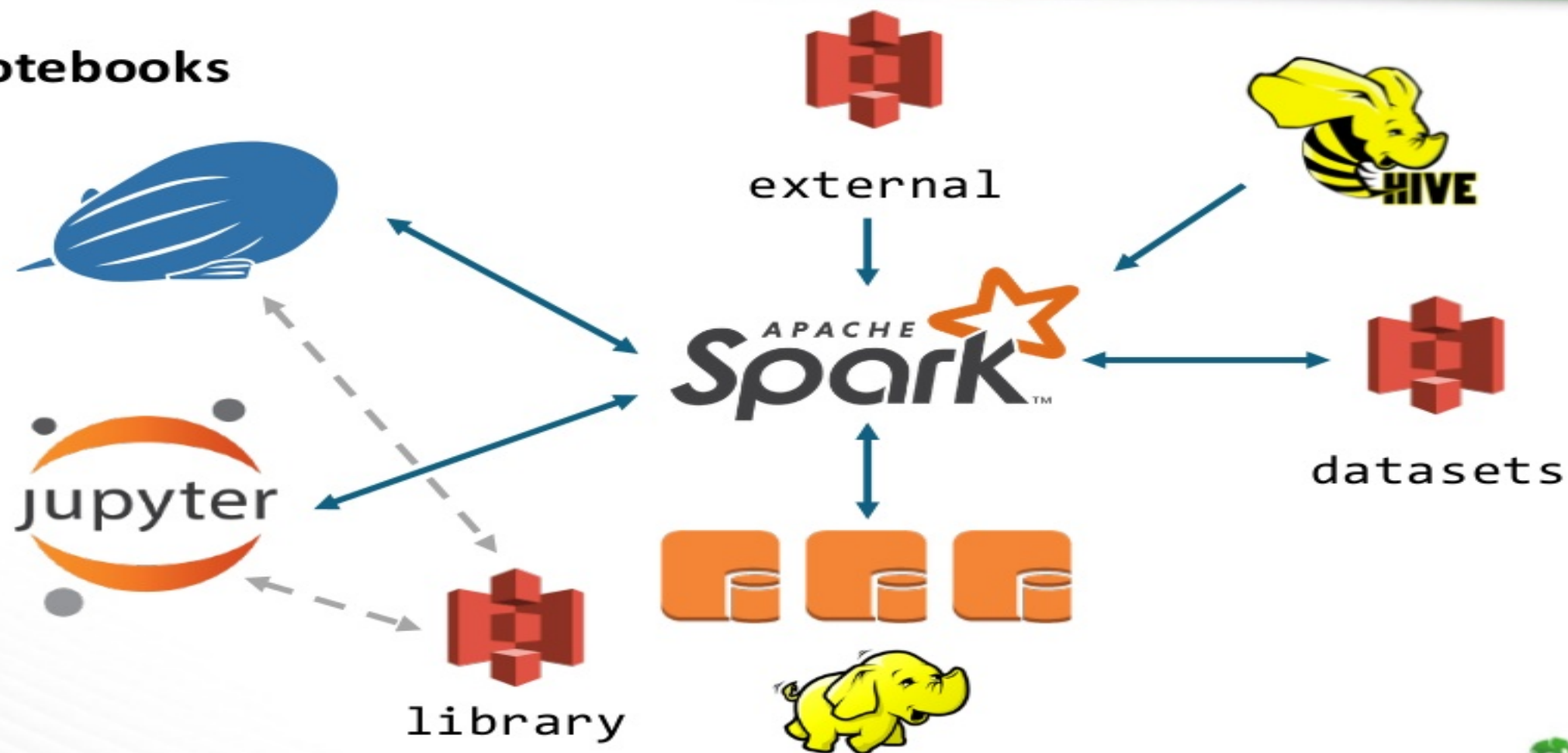
Chris Nauroth,
Apache Hadoop committer & PMC
ASF member



Elastic ETL



Notebooks



APACHE
nifi



APACHE
Spark



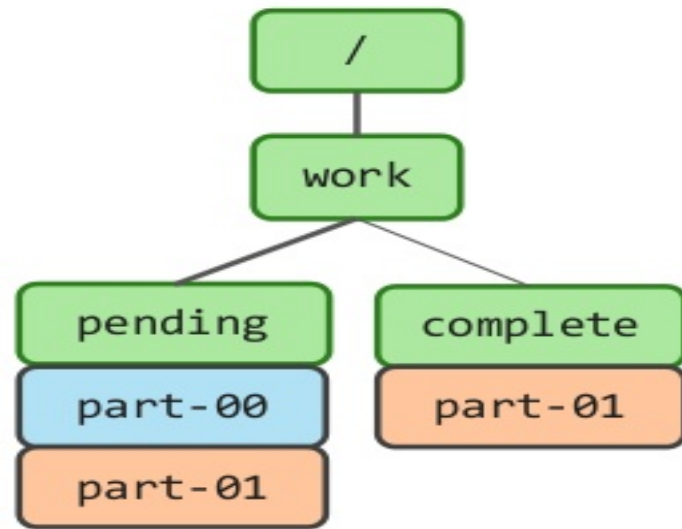
Streaming

APACHE
HBASE

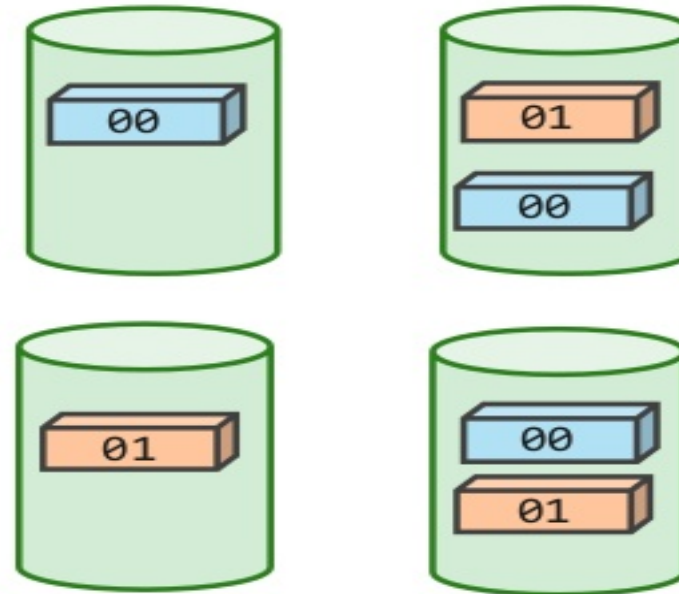


The Hortonworks logo, featuring three green elephants of increasing size.
HORTONWORKS

A Filesystem: Directories, Files → Data



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



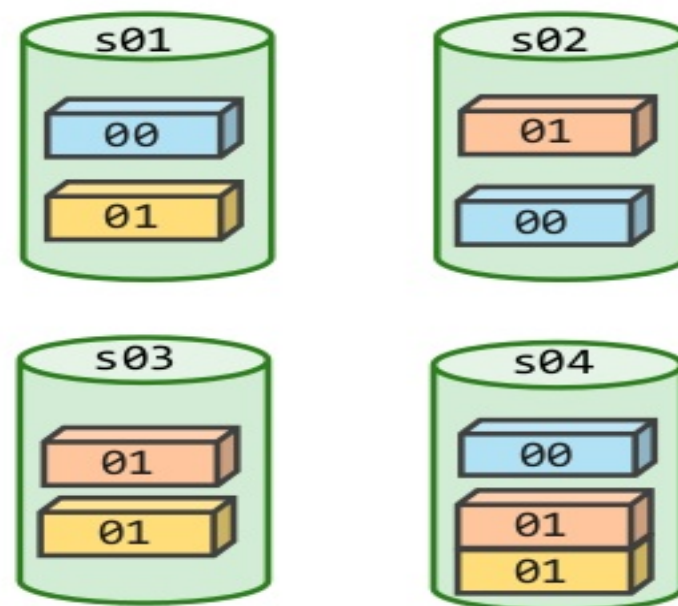
Object Store: hash(name) -> blob

```
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")
```



REST APIs

```
PUT /work/pending/part-01  
... DATA ...
```

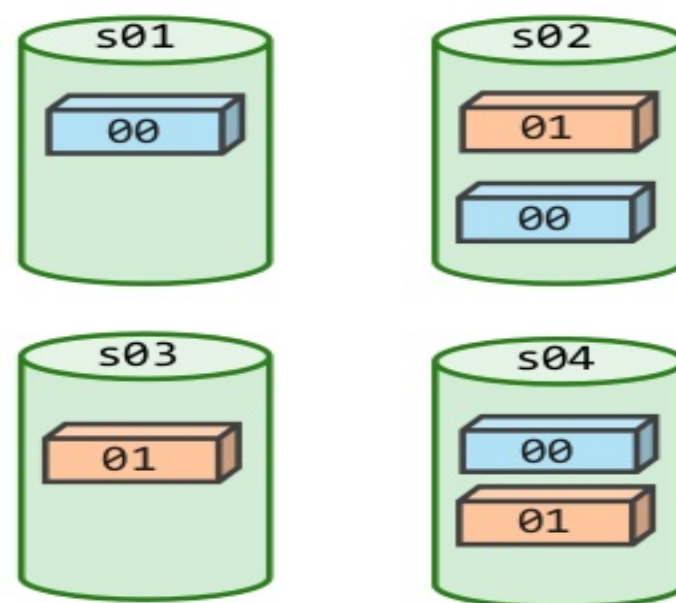
```
GET /work/pending/part-01  
Content-Length: 1-8192
```

```
PUT /work/complete/part01  
x-amz-copy-source: /work/pending/part-01
```

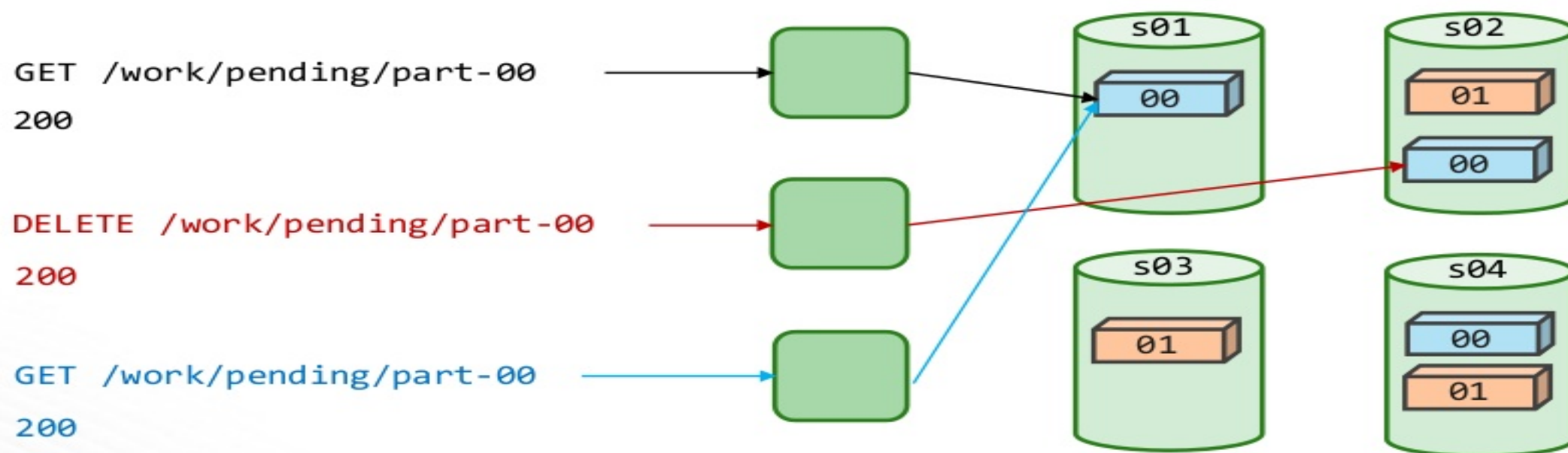
```
DELETE /work/pending/part-01
```

```
HEAD /work/complete/part-01
```

```
GET /?prefix=/work&delimiter=/
```



Often: Eventually Consistent





`org.apache.hadoop.fs.FileSystem`



hdfs



wasb



s3a



swift

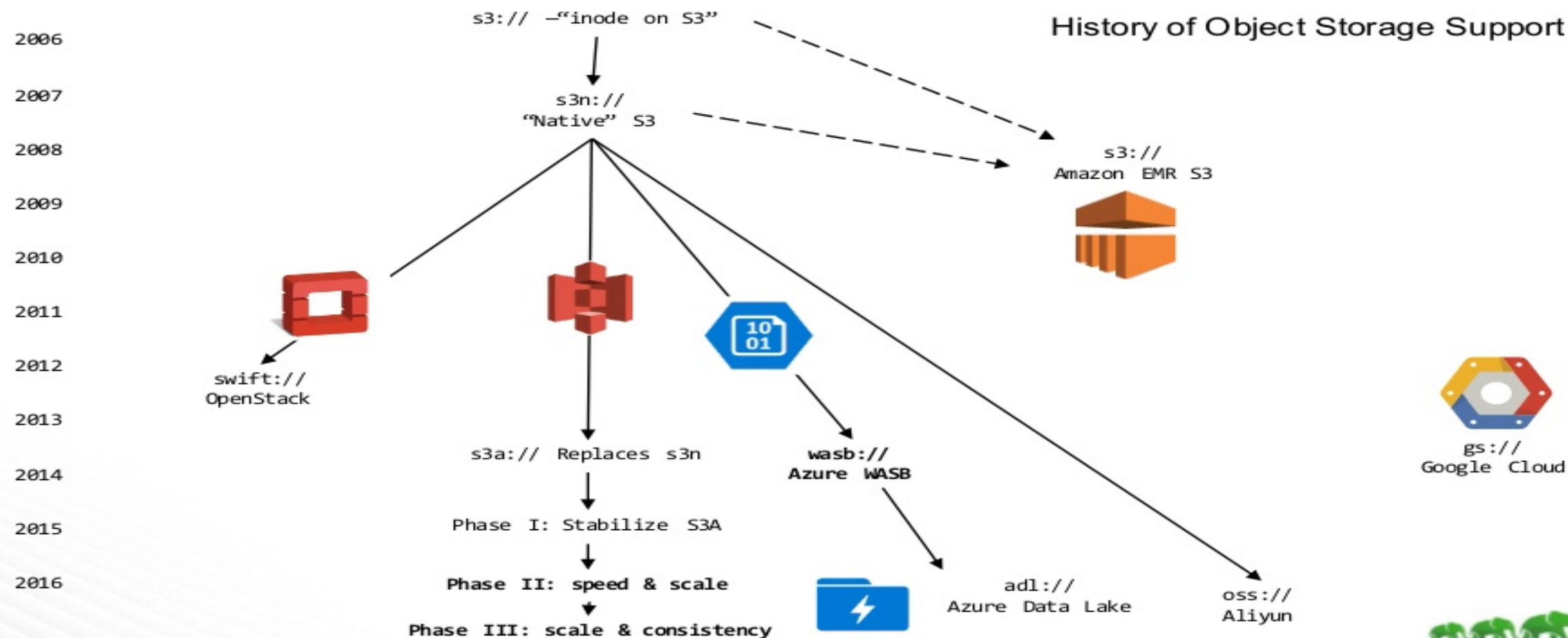


adl



gs

History of Object Storage Support



Cloud Storage Connectors

| | | |
|------------------------------|-------|---|
| Azure | WASB | <ul style="list-style-type: none">• Strongly consistent• Good performance• Well-tested on applications (incl. HBase) |
| | ADL | <ul style="list-style-type: none">• Strongly consistent• Tuned for big data analytics workloads |
| Amazon Web Services | S3A | <ul style="list-style-type: none">• Eventually consistent - consistency work in progress by Hortonworks• Performance improvements in progress• Active development in Apache |
| | EMRFS | <ul style="list-style-type: none">• Proprietary connector used in EMR• Optional strong consistency for a cost |
| Google Cloud Platform | GCS | <ul style="list-style-type: none">• Multiple configurable consistency policies• Currently Google open source• Good performance• Could improve test coverage |

Four Challenges

1. Classpath
2. Credentials
3. Code
4. Commitment

Use S3A to work with S3

(EMR: use Amazon's s3://)



Classpath: fix “No FileSystem for scheme: s3a”

hadoop-aws-2.7.x.jar

aws-java-sdk-1.7.4.jar
joda-time-2.9.3.jar
(jackson-*-2.6.5.jar)

*Get Spark with
Hadoop 2.7+ JARs*

See SPARK-7481

Credentials

`core-site.xml` or `spark-default.conf`

`spark.hadoop.fs.s3a.access.key MY_ACCESS_KEY`

`spark.hadoop.fs.s3a.secret.key MY_SECRET_KEY`

`spark-submit` propagates Environment Variables

`export AWS_ACCESS_KEY=MY_ACCESS_KEY`

`export AWS_SECRET_KEY=MY_SECRET_KEY`

NEVER: share, check in to SCM, paste in bug reports...

Authentication Failure: 403

```
com.amazonaws.services.s3.model.AmazonS3Exception:  
The request signature we calculated does not match  
the signature you provided.  
Check your key and signing method.
```

1. Check joda-time.jar & JVM version
2. Credentials wrong
3. Credentials not propagating
4. Local system clock (more likely on VMs)

Code: Basic IO

```
// Read in public dataset  
val lines = sc.textFile("s3a://landsat-pds/scene_list.gz")  
val lineCount = lines.count()
```

```
// generate and write data  
val numbers = sc.parallelize(1 to 10000)  
numbers.saveAsTextFile("s3a://hwdev-steve1-demo/counts")
```

All you need is the URL

Code: just use the URL of the object store

```
val csvdata = spark.read.options(Map(  
  "header" -> "true",  
  "inferSchema" -> "true",  
  "mode" -> "FAILFAST"))  
  .csv("s3a://landsat-pds/scene_list.gz")
```

...read time $O(\text{distance})$

DataFrames

```
val landsat = "s3a://stevel-demo/landsat"  
csvData.write.parquet(landsat)
```

```
val landsatOrc = "s3a://stevel-demo/landsatOrc"  
csvData.write.orc(landsatOrc)
```

```
val df = spark.read.parquet(landsat)  
val orcDf = spark.read.parquet(landsatOrc)
```

...list inconsistency
...commit time $O(\text{data})$

Finding dirty data with Spark SQL

```
val sqlDF = spark.sql(  
  "SELECT id, acquisitionDate, cloudCover"  
  + s" FROM parquet.`${landsat}`")
```

```
val negativeClouds = sqlDF.filter("cloudCover < 0")  
negativeClouds.show()
```

- * filter columns and data early
- * whether/when to cache()?
- * copy popular data to HDFS

spark-default.conf

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

Recent S3A Performance (Hadoop 2.8, HDP 2.5, CDH 6 (??))

// forward seek by skipping stream

`spark.hadoop.fs.s3a.readahead.range 256K`

// faster backward seek for ORC and Parquet input

`spark.hadoop.fs.s3a.experimental.input.fadvise random`

// PUT blocks in separate threads

`spark.hadoop.fs.s3a.fast.output.enabled true`

The Commitment Problem

- `rename()` used for atomic commitment transaction
- time to `copy()` + `delete()` proportional to data * files
- S3: 6+ MB/s
- Azure: a lot faster —*usually*

`spark.speculation false`

`spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version 2`

`spark.hadoop.mapreduce.fileoutputcommitter.cleanup.skipped true`

The "Direct Committer"?

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, and a Create button. A search bar is also present.

The issue title is "Spark / SPARK-10063 Remove DirectParquetOutputCommitter". Below the title are buttons for Edit, Comment, Agile Board, More, Close Issue, and Reopen Issue.

Details

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

Description

When we use DirectParquetOutputCommitter on S3 and speculation is enabled, there is a chance that we can loss data.

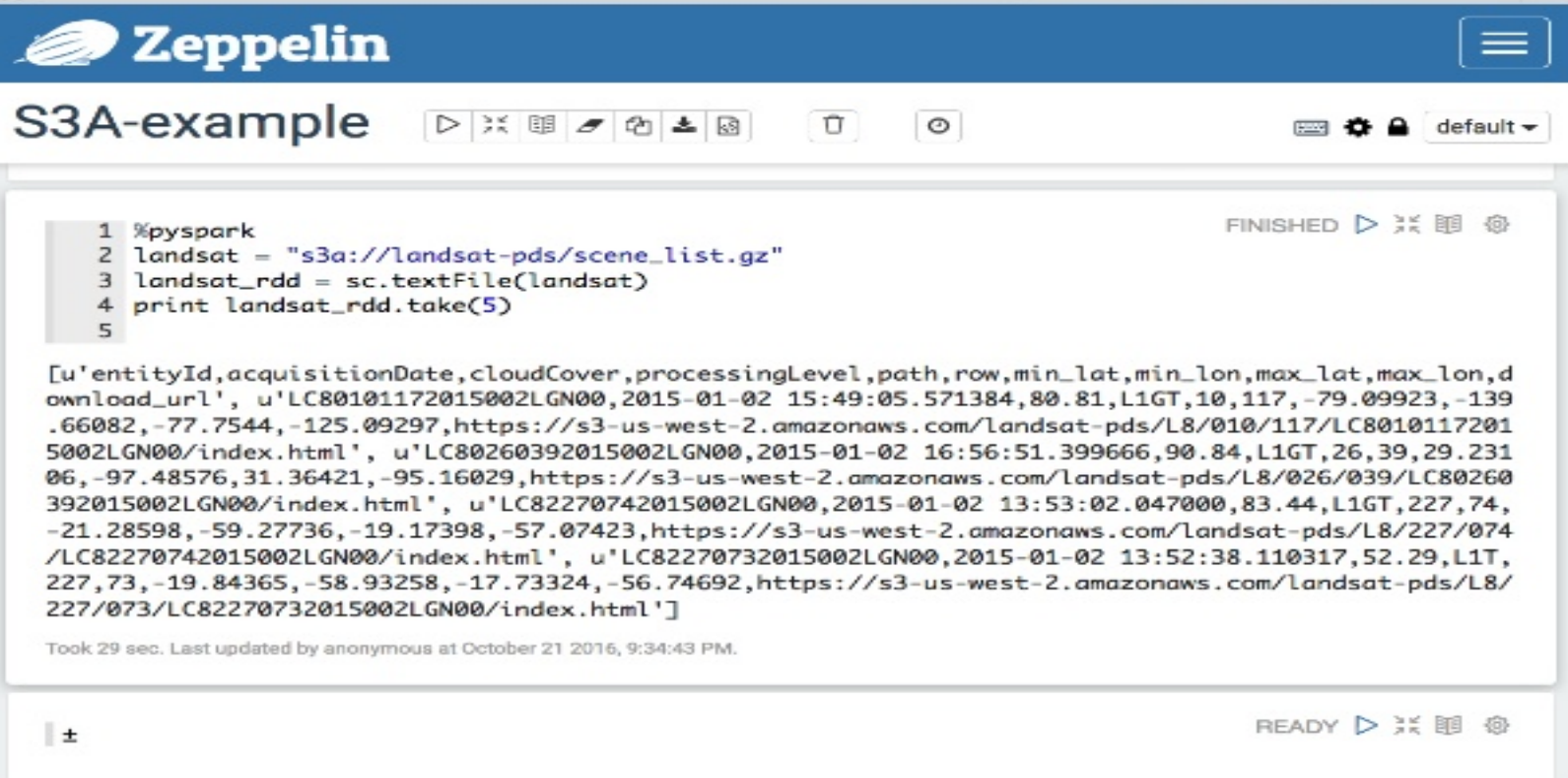
People

Assignee: Reynold Xin

Reporter: Yin Huai

Votes: 0 Vote for this issue

Watchers: 14 Stop watching this is



The image shows a Zeppelin notebook interface with a blue header bar containing the Zeppelin logo and a hamburger menu. Below the header, the notebook title "S3A-example" is displayed. A toolbar with various icons (play, undo, redo, copy, paste, download, upload, delete, refresh) is located below the title. The main content area shows a code cell with the following Python code:

```
1 %pyspark
2 landsat = "s3a://landsat-pds/scene_list.gz"
3 landsat_rdd = sc.textFile(landsat)
4 print landsat_rdd.take(5)
5
```

The code cell status is "FINISHED". Below the code, the output is displayed as a long string of JSON-like data representing satellite scene information. At the bottom of the code cell, it says "Took 29 sec. Last updated by anonymous at October 21 2016, 9:34:43 PM." Below the code cell, there is a "READY" status and a toolbar with icons for zooming, copying, and refreshing.

Notebooks? Classpath & Credentials

Azure Storage: wasb://

A full substitute for HDFS



Classpath: fix “No FileSystem for scheme: wasb”

`wasb://` : Consistent, with very fast rename (hence: commits)

`hadoop-azure-2.7.x.jar`

`azure-storage-2.2.0.jar`

+ (jackson-core; http-components, hadoop-common)

Credentials: core-site.xml / spark-default.conf

```
<property>  
  <name>fs.azure.account.key.example.blob.core.windows.net</name>  
  <value>0c0d44ac83ad7f94b0997b36e6e9a25b49a1394c</value>  
</property>
```

```
spark.hadoop.fs.azure.account.key.example.blob.core.windows.net  
0c0d44ac83ad7f94b0997b36e6e9a25b49a1394c
```

```
wasb://demo@example.blob.core.windows.net
```

Example: Azure Storage and Streaming

```
val streaming = new StreamingContext(sparkConf, Seconds(10))
val azure = "wasb://demo@example.blob.core.windows.net/in"
val lines = streaming.textFileStream(azure)
val matches = lines.map(line => {
    println(line)
    line
})
matches.print()
streaming.start()
```

- * PUT into the streaming directory
- * keep the dir clean
- * size window for slow scans
- * checkpoints slow —reduce frequency

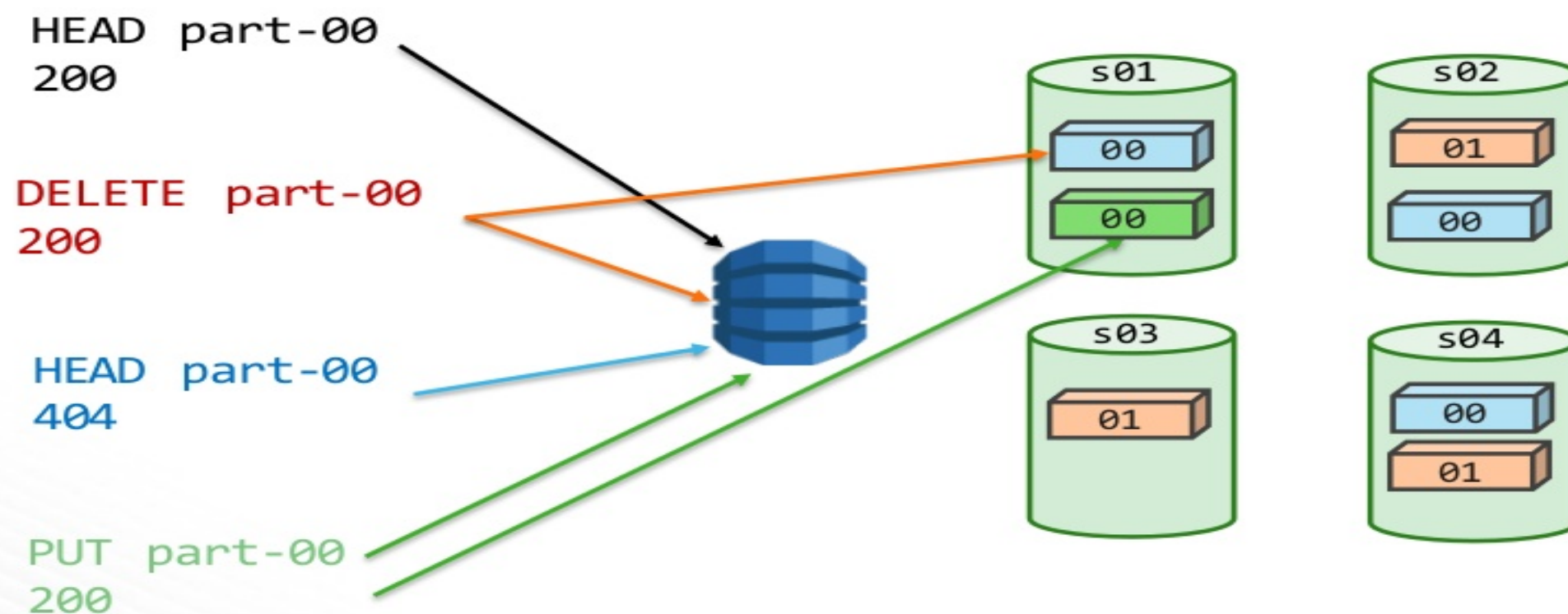
s3guard: fast, consistent S3 metadata

HADOOP-13445

Hortonworks + Cloudera + Western Digital



DynamoDB as consistent metadata store



[HADOOP-13786] Add S3Guard committer

Secure | https://issues.apache.org/jira/browse/HADOOP-13786

APACHE HADOOP

Dashboards Projects Issues Agile Service Desk Create

Search

Hadoop Common / HADOOP-13786

Add S3Guard committer for zero-rename commits to consistent S3 endpoints

Edit Comment Assign More Submit Patch Start Progress Resolve Issue Export

Details

| | | | |
|--------------------|--------------|----------------|------------|
| Type: | New Feature | Status: | OPEN |
| Priority: | Major | Resolution: | Unresolved |
| Affects Version/s: | HADOOP-13345 | Fix Version/s: | None |
| Component/s: | fs/s3 | | |
| Labels: | None | | |
| Target Version/s: | HADOOP-13345 | | |

Description

A goal of this code is "support O(1) commits to S3 repositories in the presence of failures" (inherent it, and to the user is needed to demonstrate the correctness of the algorithm. (that is, assuming that s3guard provides a consistent view of the presence/absence of blobs, show that we can commit directly).

I consider ourselves free to expose the blobstore-ness of the s3 output stream (not visible under the close()), if we need to use that to allow us to abort commit operations.

Attachments

| | | |
|-------------------------------------|--------|-----------------|
| HADOOP-13786-HADOOP-13345-001.patch | 126 kB | 16/Dec/16 18:28 |
|-------------------------------------|--------|-----------------|

Issue Links

| | | |
|---------------|--|----------|
| depends upon | HADOOP-13449 S3Guard implement DynamoDBMetadataStore | RESOLVED |
| is related to | HADOOP-13449 S3Guard implement OutputFormat to support configurable FileOut... | OPEN |
| relates to | HADOOP-13449 S3Guard implement DirectParquetOutputCommitter | RESOLVED |
| relates to | HADOOP-13449 S3Guard implement S3a Multipart Committer (avoid rename) | OPEN |
| relates to | HADOOP-13449 S3Guard implement rename(final Path src, final Path ds... | OPEN |

Show 1 more links (1 relates to)

Sub-Tasks

| | | |
|--|------|----------------|
| 1. S3ABlockOutputStream to support plugin point for different multipart strategies | OPEN | Steve Loughran |
|--|------|----------------|

People

| | |
|-----------|--------------------------|
| Assignee: | Steve Loughran |
| Reporter: | Steve Loughran |
| Votes: | 0 |
| Watchers: | Stop watching this issue |

Dates

| | |
|----------|-----------------|
| Created: | 02/Nov/16 18:16 |
| Updated: | 2 minutes ago |

Development

JIRA is having difficulty contacting fisheye. If this condition persists, please contact your JIRA administrators.

Agile

View on Board

Summary

- ◆ Object Stores look just like any other Filesystem URL
- ◆ ...but do need classpath and configuration
- ◆ Issues: performance, commitment
- ◆ *Tune to reduce I/O*
- ◆ *Keep those credentials secret!*

Finally: keep an eye out for s3guard!

Questions?



Backup Slides



Not Covered

- Partitioning/directory layout
- Infrastructure Throttling
- Optimal path names
- Error handling
- Metrics

Dependencies in Hadoop 2.8

hadoop-aws-2.8.x.jar

aws-java-sdk-core-1.10.6.jar
aws-java-sdk-kms-1.10.6.jar
aws-java-sdk-s3-1.10.6.jar
joda-time-2.9.3.jar
(jackson-*-2.6.5.jar)

hadoop-aws-2.8.x.jar

azure-storage-4.2.0.jar

S3 Server-Side Encryption

- Encryption of data at rest at S3
- Supports the SSE-S3 option: each object encrypted by a unique key using AES-256 cipher
- Now covered in S3A automated test suites
- Support for additional options under development (SSE-KMS and SSE-C)

Advanced authentication

```
<property>  
  <name>fs.s3a.aws.credentials.provider</name>  
  <value>  
    org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider,  
    org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider,  
    com.amazonaws.auth.EnvironmentVariableCredentialsProvider,  
    com.amazonaws.auth.InstanceProfileCredentialsProvider,  
    org.apache.hadoop.fs.s3a.AnonymousAWSCredentialsProvider  
  </value>  
</property>
```

+encrypted credentials in JECKS files on
HDFS

Optimize uses of FS operations in the ASF analysis frameworks and libraries

Details

| | | | |
|--------------------|-----------|----------------|------------|
| Type: | Sub-task | Status: | OPEN |
| Priority: | Major | Resolution: | Unresolved |
| Affects Version/s: | 2.7.2 | Fix Version/s: | None |
| Component/s: | fs, fs/s3 | | |
| Labels: | None | | |

Description

Review uses of the FS APIs in applications using the Hadoop FS API to access filesystems; identify suboptimal uses and tune them for better performance against HDFS and object stores

- Assume arbitrary Hadoop 2.x releases: make no changes which are known to make operations on older versions of Hadoop slower
- Do propose those changes which deliver speedups in later versions of Hadoop, while not impacting older versions, or risk of causing scalability problems.
- Add more tests, especially scalable ones which also display metrics.
- Use standard benchmarks and optimization tools to identify hotspots.
- Use FS behaviour as verified in the FS contract tests as evidence that filesystems correctly implement the Hadoop FS APIs. If a use of an API call is made which hints at the expectation of different/untested behaviours, leave alone and add new tests to the Hadoop FS contract to determine cross-FS semantics.
- Focus on the startup, split calculation and directory scanning operations: the ones which slow down entire queries.
- Eliminate use of `isDirectory()`, `getLength()`, `exists()` if a follow-on operation (`getStatus()`, `delete()`, ...) makes the use redundant.
- Assume that `FileStatus` entries are not cached; the cost of creating them is 1 RPC call against HDFS, 1+ HTTPS call against object stores.
- Locate calls to the listing operations, identify speedups, especially on recursive directory scans.
- Identify suboptimal seek patterns (backwards as well as forwards) and attempt to reduce/eliminate through reordering and result caching.
- Try to reuse the results of previous operations (e.g. `FileStatus` instances) in follow-on calls.
- Commonly used file formats (e.g. ORC) will have transitive benefits.
- Frameworks to use predicate pushdown where this delivers speedups
- Document best practises identified and implemented.

People

Assignee: Steve Loughran
 Reporter: Steve Loughran
 Votes: 0
 Watchers: Stop watching this issue

Dates

Created: 20/Aug/16 12:33
 Updated: 20/Aug/16 12:43

Agile

[View on Board](#)