Hopsworlks – Self-Service Spark/Flink/Kafka/Hadoop

Jim Dowling
Associate Prof @ KTH
Senior Researcher @ SICS

Apache BigData Europe, 2016
Hadoop is not a cool kid anymore!
Where did it go wrong for Hadoop?

• Data Engineers/Scientists
  - Where is the User-Friendly tooling and Self-Service?
  - How do I install/operate anything other than a sandbox VM?

• Operations Folks
  - Security model has become incomprehensible (Sentry/Ranger)
  - Major distributions not open enough for patching
  - Sensitive data still requires its own cluster
  - Why not just use AWS EMR/GCE/Databricks/etc ?!?
MAKE HADOOP GREAT AGAIN!

NOBODY HAS BIGGER DATA THAN ME. MY DATA IS THIS BIG!
Is this Hadoop?

Processing
- Presto
- Hive
- MR
- Kafka
- Spark
- HBase
- Flink
- Tensorflow

Storage
- S3
- HDFS
- WFS
- GCS

Resource Manager
- Mesos
- YARN
- Kubernetes

Platform
- Azure
- On-Premise
- AWS
- GCE
How about this?

- **Processing**
  - Presto
  - Hive
  - MR
  - Kafka
  - Spark
  - HBase
  - Flink
  - Tensorflow

- **Storage**
  - S3
  - HDFS
  - WFS
  - GCS

- **Resource Manager**
  - Mesos
  - YARN
  - Kubernetes

- **Platform**
  - Azure
  - On-Premise
  - AWS
  - GCE
Here’s Hops Hadoop Distribution

- **Processing**
  - Logstash
  - Flink
  - Hopsworks
  - Elasticsearch
  - Kibana
  - Spark
  - Kafka
  - Grafana
  - Tensorflow

- **Storage**
  - HDFS

- **Resource Manager**
  - YARN

- **Platform**
  - On-Premise
  - AWS
  - GCE
Hadoop Distributions Simplify Things

Install / Upgrade

Ambari  Karamell/Chef  Cloudera Mgr

MR  Spark  Kafka  Flink  Tensorflow

HDFS

YARN

On-Premise
Cloud-Native means Object Stores, not HDFS
Object Stores and False Equivalence*

• Object Stores are inferior to hierarchical filesystems
  - False equivalence in the tech media

• Object stores can scale better, but at what cost?
  - Read-your-writes existing objs
  - Write, then list
  - Hierarchical namespace properties
    • Quotas, permissions
  - Other eventual consistency probs

*False Equivalence*: “unfairly equating a minor failing of Hillary Clinton’s to a major failing of Donald Trump’s.”
Implement your own eventually consistent metadata store to mirror the (unobservable) metadata

- Netflix for AWS
- Spotify for GCS

Can we open up the ObjectStore’s metadata?

Object Store metadata is a Pandora’s Box. Best keep it closed.
Eventual Consistency is Hard

```java
r1 = read();
if (r1 == null) {
    // Loop until read completes
} else if (r1 == W1){
    // do Y
} else if (r1 == W2) {
    // do Z
}
```

Domain = MyDomain, Item = StandardFez

S3 guarantees
NoSQL systems grew from the need to scale-out relational databases. NewSQL systems bring both scalability and strong consistency, and companies moving back to strong consistency.*

Object stores systems grew from the need to scale-out filesystems. A new generation of hierarchical filesystem will appear that bring both scalability and hierarchy, and companies will eventually move back to scalable hierarchical filesystems.

*http://www.cubrid.org/blog/dev-platform/spanner-globally-distributed-database-by-google/
HopsFS and Hops YARN

![Bar chart comparing HDFS 2.1, HDFS 2.7, and HopsFS in terms of performance. The chart shows that HopsFS has 16x better performance compared to HDFS 2.1 and HDFS 2.7.](chart.png)
Externalizing the NameNode/YARN State

• Problem: Metadata is preventing Hadoop from becoming the great platform we know she can be!

• Solution: Move the metadata off the JVM Heap

• Where? To an in-memory database that can be transactionally and efficiently queried and managed. The database should be Open-Source. We chose NDB – aka MySQL Cluster.
HopsFS – 1.2 million ops/sec (16X HDFS)

NDB Setup: Nodes using Xeon E5-2620 2.40GHz Processors and 10GbE. NameNodes: Xeon E5-2620 2.40GHz Processors machines and 10GbE.
HopsFS Metadata Scaleout

Assuming 256MB Block Size, 100 GB JVM Heap for Apache Hadoop
Hops YARN Architecture

YARN Client

NDB

ResourceMgrs
  Scheduler
  Resource Trackers

AM Reqs
(5-30%)

NodeManagers

Heartbeats
(70-95%)
Extending Metadata in Hops

• JSON API (with and without schema)

  public void attachMetadata(Json obj, String pathToFileorDir)
  • Row added with jsonObj & foreign key to the inode (on delete cascade)
  • Use to annotate files/directories for search in Elasticsearch

<table>
<thead>
<tr>
<th>id</th>
<th>json_obj</th>
<th>inode (fk)</th>
<th>inode_id</th>
<th>attrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{ trump: non}</td>
<td>10101</td>
<td>10101</td>
<td>/tmp...</td>
</tr>
</tbody>
</table>

• Add Tables in the database
  • Foreign keys to inode/applications ensure metadata integrity
  • Transactions involving both the inode(s) and metadata ensure metadata consistency
  • Enables modular extensions to Namenodes and ResourceManagers
Strongly Consistent Metadata

Metadata Integrity maintained using 2PC and Foreign Keys.

NDB

Provenance
Security
Quotas
Projects
Datasets

2-phase commit

Files
Directories
Containers
Eventually Consistent Metadata with Epipe

Metadata Integrity maintained by one-way Asynchronous Replication.

Kafka → async. replication → Epipe → NDB

[ RTP Tutorial, BOSS Workshop, VLDB 2016]
Metadata integrity maintained by custom recovery logic and polling.
Tinker-Friendly Metadata

- Erasure Coding in HopsFS
- Free-Text search of HopsFS namespace
  - Export changelog for NDB to Elasticsearch
- Dynamic User Roles in HopsFS
- **New abstractions:** Projects and Datasets

Extensible, tinker-friendly metadata is like 3-d printing for Hadoop
New Concepts in Hops Hadoop

Hops
- Projects
  - Datasets
  - Topics
  - Users
  - Jobs
  - Notebooks

Enabled by Extensible Metadata

Hadoop
- Users
- Applications
- Jobs
- Files
- Clusters
- ACLs
- Sys Admins
- Kerberos
Simplifying Hadoop with Projects

Project

Metadata in Hops
- App Logs
- Datasets
- HDFS files/dir
- Users
- Notebooks
- Access Control

Metadata not in Hops
- Kafka Topics
Simplifying Hadoop with Datasets

- Datasets are a directory subtree in HopsFS
  - Can be shared securely between projects
  - Indexed by Elasticsearch

- Datasets can be made public and downloaded from any Hopsworks cluster anywhere in the world
The Distributed Database is the Single Source-of-Truth for Metadata

Projects & Datasets
- Files
- Directories
- Containers
- Provenance
- Security
- Quotas

2-Phase Commit in NDB

Hopworks REST API

Epipe (DB Changelog)

Eventual Consistency

Elasticsearch
Kafka

mutation
Hopsworks
A project is a collection of:
- Users with Roles
- HDFS DataSets
- Kafka Topics
- Notebooks, Jobs

Per-Project quotas:
- Storage in HDFS
- CPU in YARN
  - Uber-style Pricing

Sharing across Projects:
- Datasets/Topics
Project Roles

- **Data Owner Privileges**
  - Import/Export data
  - Manage Membership
  - Share DataSets, Topics

- **Data Scientist Privileges**
  - Write and Run code

We delegate administration of privileges to users
Hopsworks – Dynamic Roles

Alice@gmail.com

Authenticate

Projects

Secure Impersonation

NSA_Alice

NSA_Alice

Users_Alice

HopsYARN

HopsFS

X.509 Certificates

Kafka
X.509 Certificates Everywhere

• User and service certs share same self-signed root CA

• Project-Specific User Certificates
  - Every user in every project is issued with a X.509 certificate, containing the project-specific userID.
    • Scales using intermediate CAs at each Hopsworks instance.
    • Inspired by Netflix’ BLESS system.

• Service Certificates
  - Services use a host-specific certificate that is signed by the root CA. Process managed by an agent program (kagent).
  - Services identify SSL clients by extracting the CommonName from client certificate in RPC calls. Kerberos keytab gone.
Users don’t see the certificates. Users authenticate using:
- LDAP,
- password,
- 2-Factor Authentication

Alice@gmail.com

Hopsworks

Project Mgr

Add/Del Users

Insert/Remove Certs

Intermediate Certificate Authority

Cert Signing Requests

Root CA

Distributed Database

Kagent

HDFS
Spark
Kafka
YARN

X.509 Certificate Generation
Alice@gmail.com

1. Launch Spark Job

Distributed Database

2. Get certs, service endpoints

Spark/Flink Streaming App

3. YARN Job, config

YARN Private LocalResources

4. Materialize certs

5. Read Certs

6. Get Schema

7. Consume Produce

8. Authenticate

KafkaUtil
1. **Discover**: Schema Registry and Kafka Broker Endpoints
2. **Create**: Kafka Properties file with certs and broker details
3. **Create**: producer using Kafka Properties
4. **Download**: the Schema for the Topic from the Schema Registry
5. **Distribute**: X.509 certs to all hosts on the cluster
6. **Cleanup securely**

All of these steps are now down automatically by Hopsworks’ KafkaUtil library
Hops simplifies Secure Flink/Kafka Producer

Properties props = new Properties();
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, brokerList);
props.put(SCHEMA_REGISTRY_URL, restApp.restConnect);
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
org.apache.kafka.common.serialization.StringSerializer.class);
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
io.confluent.kafka.serializers.KafkaAvroSerializer.class);
props.put("producer.type", "sync");
props.put("serializer.class","kafka.serializer.StringEncoder");
props.put("request.required.acks", "1");
props.put("ssl.keystore.location","/var/ssl/kafka.client.keystore.jks")
props.put("ssl.keystore.password","test1234 ")
props.put("ssl.key.password","test1234")
ProducerConfig config = new ProducerConfig(props);
String userSchema = "{"namespace": "example.avro",
 \"type\": \"record\", \"name\": \"User\"," +
 \"fields\": [{\"name\": \"name\", 
 \"type\": \"string\"}]"");
Schema schema = parser.parse(userSchema);
GenericRecord avroRecord = new GenericData.Record(schema);
avroRecord.put("name", "testUser");
Producer<String, String> producer = new Producer<String, String>(config);
ProducerRecord<String, Object> message = new
ProducerRecord<"topicName", avroRecord );
producer.send(data);

FlinkProducer prod =
KafkaUtil.getFlinkProducer (topicName);

DataStream<...> ms =
env.addSource(...);
ms.addSink(producer);
env.execute("Producing");

StreamExecutionEnvironment
env = ...

Massively Simplified
Code for Hops/Flink/Kafka

Losts of Hard-Coded Endpoints Here!
Zeppelin Support for Spark/Livy

HopsWorks

Create New Notebook

Goto Zeppelin

Create New Notebook

INTERPRETERS

fink Interpreter running

angular Interpreter stopped

livy Interpreter stopped

spark Interpreter stopped

md Interpreter stopped

Running a paragraph in a notebook will automatically start the necessary interpreters for that job.
Spark Jobs in YARN with Livy

MULTI USERS

Spark Client

Livy Server

Scalatra

Session Manager

Session

YARN Node

Spark Interpreter

Spark Context

YARN Node

Spark Interpreter

Spark Context

[Image from: http://gethue.com]
Debugging Spark Jobs with Dr Elephant

- Project-specific view of performance and correctness issues for completed Spark Jobs
- Customizable heuristics
- Doesn’t show killed jobs
SICS ICE: A datacenter research and test environment
Purpose: Increase knowledge, strengthen universities, companies and researchers
Karamel/Chef for Automated Installation

- Google Compute Engine
- OpenStack
- Amazon EC2
- BareMetal
Summary

• Hops is the only European distribution of Hadoop
  - More scalable, tinker-friendly, and open-source.

• Hopsworks provides first-class support for Flink-/Spark-Kafka-as-a-Service
  - Streaming or Batch Jobs
  - Zeppelin Notebooks

• Hopworks provides best-in-class support for secure streaming applications with Kafka
Hops Team

Active: Jim Dowling, Seif Haridi, Tor Björn Minde, Gautier Berthou, Salman Niazi, Mahmoud Ismail, Theofilos Kakantousis, Antonios Kouzoupis, Ermias Gebremeskel.
