Hopsworks: Secure, UI-Driven Spark/Flink/Kafka-as-a-Service

Dr. Jim Dowling
Associate Prof @ KTH
Senior Researcher @ SICS
CEO @ Logical Clocks AB

Apache BigData North America, 2017
True Hoppiness is a state of mind where your users are at one with your Data Platform

NOT THIS
Love for Hadoop is in short Supply

- Limited self-service
- Data parallel JVM Programming
- Kerberos
- Distributed System!
- Integration
Hadoop is not a cool kid anymore!
The Evolution of Hadoop

2009

Huge Body (DataNodes)

2017

Tiny Brain (NameNode, ResourceMgr)

ApacheCon BigData, Hopsworks, J Dowling, May 2017
Evolutionary Dead-End – the NameNode

- Limited namespace/metadata
  - JVM Heap (~200 GB)

- Limited concurrency
  - Single global namespace lock (single-writer, multiple readers)
*Cladogenesis is the process by which a species splits into two distinct species, rather than one species gradually transforming into another.
1. Scale-out Metadata
   - Metadata in an in-memory distributed database
   - Multiple stateless NameNodes

2. Remove the Global Namespace Lock
   - Supports multiple concurrent read and write operations
HopsFS Architecture
HopsFS: Next Generation HDFS*

**Bigger**

**Faster**

Scale Challenge Winner (2017)

*https://www.usenix.org/conference/fast17/technical-sessions/presentation/niazi
Can HopsFS help us reimagine Hadoop? But, first..
EU General Data Protection Regulation (GDPR)

• Privacy by Design and by Default
• Roles: Data Controller, Data Processor
• Data Retention
• Data Protection Officer
• Consent, Portability, Right-to-be-forgotten
Can we just try and make Hadooping fun?
Hopsworks Data Platform

• Self-Service Check-List
  1. Manage Projects like Github
  2. Share data like Dropbox
  3. 1st Class Python Support
  4. Search support for Datasets
  5. GDPR compliant

I Sparked it up all by myself, Obama
Manage Projects Like GitHub

Data Owner

Datasets

Data Scientist

member
Share Data like Dropbox

Share any Data Source/Sink: HDFS Datasets, Kafka Topics, etc
Modern Data Parallel Platforms
(Anti Kitchen Sink Pattern)
Workflows/Jobs and Notebook Support

Data Scientist → Apache Zeppelin

Jobs (batch/streaming)
Custom Python Environments with Conda

Python libraries are usable by any framework (Spark/Tensorflow)
Download into HDFS Datasets and/or Kafka Topics
Projects, Data, Users
A **Project** is a Grouping of **Users** and **Data**

### Projects for Multi-tenancy

**CompanyDB**

**Proj-All**

**Proj-42**

**Proj-X**

**Topic**

**Shared Topic**

/Projs/My/Data
Multi-Cloud SaaS IoT Data Platform

User Apps control IoT Devices

ACME

DontBeEvil Corp

Evil-Corp

AWS

Google Cloud

Oracle Cloud

IoT Data Platform

Field Gateway

Field Gateway

Field Gateway
SaaS IoT Platform: Project per Customer

ACME manage membership

IoT Project

Generic Analytics

Kafka Topic

Shared

ACME HDFS Dataset

ACME Topic

Custom Analytics

ACME Project

Data Stream
Project Roles

- Authorization made Easy
  - ACLs are hard

- 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

ApacheCon BigData, Hopsworks, J Dowling, May 2017
Alice has a new HDFS user per Project (ProjectUser)

- projX__Alice
- proj42__Alice

Each ProjectUser has its own SSL/TLS cert.
Certificate Generation/Removal/Revocation

Add/Del Users → Project

Distributed Database

Insert/Remove Certs

Cert Signing Requests → Root CA

Intermediate Certificate Authority

Hopsworks

Alice@gmail.com

HDFS

Spark

Kafka

YARN

HDFS Spark Kafka YARN
Frameworks can help (if done right)

- Hopsworks knows the following:
  - Endpoints of all services (NameNode, Kafka, Zookeeper)
  - Project Users, Datasets and Topics
  - Quotas (HDFS, YARN), Job Histories

- Hopsworks has two main APIs:
  1. Native HopsUtil library
  2. REST API
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
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.key.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<String, Object>("topicName", avroRecord);
producer.send(data);

Lots of Hard-Coded Endpoints Here!

SparkProducer producer = 
HopsUtil.getSparkProducer();
Map<String, String> message = ...
sparkProducer.produce(message);

Massively Simplified Code with HopsUtil

https://github.com/hopshadoop/hops-avro-util

https://github.com/hopshadoop/hops-util
Securely Managing Logs in YARN

- Application Logs are private to your Project

- Apps are launched as the same OS user and there is a need to sandbox/isolate app logs.

- Apps write local logs to an app-private directory.
  - Remove directory listing privileges on the parent directory.

- Apps log to a sticky-bit write-only directory in HDFS.
  - On app completion, logs are moved to the owning project.
Hopsworks UI
Self-Service Kafka UI

Manage & Share
- Topics
- ACLs
- Avro Schemas
Real-Time Logs

Elasticsearch, Logstash, Kibana (ELK Stack)
Zeppelin - Visualization
Jupyter Notebooks

Jupyter Contents Manager for HDFS

Jupyter server per user/project

Spark REST server

Sparkmagic

HopsFS

Notebooks stored securely in HDFS per user/project

Local Storage

Storing user files locally for running Python/Scala programs

Cluster
Jupyter and Zeppelin use Livy/Spark

MULTI USERS

Spark Client

Spark Client

Livy Server

Scalatra

Session Manager

Session

YARN Node

Spark Interpreter

Spark Context

YARN Node

Spark Interpreter

Spark Context

YARN Node

Spark Interpreter

Spark Context

[Image from: http://gethue.com]
www.hops.site
  - Provided as a service since mid-2016

- **RISE SICS ICE**
  - 250 kW Datacenter, ~1000 servers*
  - Research and test environment
Demo Streaming-as-a-Service

1. Data Ingestion
   - Distributed, Scalable
     • Apache Kafka
2. Processing
   - Apache Spark / Flink
3. Sinks
   - Shareable Datasets
     • Hopsworks
4. Visualization and Debugging

Security
SSL/TLS
Certificates

Hopsworks
Hops Ongoing Work
Dela – A Global Ecosystem for Datasets

Search results

3 Results found for: milli

<table>
<thead>
<tr>
<th>DATASET</th>
<th>PUBLIC DATASET</th>
<th>PUBLIC DATASET</th>
</tr>
</thead>
<tbody>
<tr>
<td>millimetre</td>
<td>millionsong</td>
<td>millionthings</td>
</tr>
</tbody>
</table>

Legend:
- D: Dela Instance
- C: Dela Cache
- B: Dela Buffer
- So: Hops Source
- Si: Hops Sink
Distributed Tensorflow on YARN

https://github.com/hopshadoop/hops-tensorflow

- GPU-as-a-resource in Hops-YARN
- Two Tensorflow Options:
  1. Tensorflow-on-Spark
  2. Native Tensorflow-on-YARN with Infiniband Support
Small Files in HDFS

- In both Spotify and Yahoo 20% of the files are <= 4 KB
In HopsFS, we can store small files co-located with the metadata in MySQL Cluster as on-disk data.

<table>
<thead>
<tr>
<th>inode_id</th>
<th>varbinary (on-disk column)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32123432</td>
<td>[File contents go here]</td>
</tr>
</tbody>
</table>

*Niazi et al, Size Matters: Improving the Performance of Small Files in HDFS, Poster at Eurosys 2017*
30 namenodes/datanodes and 6 NDB nodes were used. Small file size was 4 KB. HopsFs files were stored on Intel 750 Series SSDs.
Multi-Data-Center HopsFS

Client

Asynchronous Multi-Master Replication of Metadata

Network Partition Identification Service

Synchronous Replication of Blocks

Hops-eu-west1

Hops-eu-west2
Hive Metastore is Moving in with HopsFS
Hive Metastore is Moving in with HopsFS
strongly consistent hive metadata

1. removing the hdfs backing directory removes the table from hive the metastore

```sql
0: jdbc:hive2://localhost:9084> show tables;
+----------------+
| tab_name        |
| web_sales       |
| web_site        |
+----------------+
2 rows selected (0.141 seconds)
0: jdbc:hive2://localhost:9084>
```

```bash
$ hdfs dfs -rm -r /user/glassfish/2/web_sales
```

```sql
0: jdbc:hive2://localhost:9084> show tables;
+----------------+
| tab_name        |
| web_site        |
+----------------+
1 row selected (0.155 seconds)
0: jdbc:hive2://localhost:9084>
```
• Hops supports larger/faster Hadoop Clusters
  - More scalable, tinker-friendly, and fully open-source.

• Hopsworks is a new Data Platform built on Hops
  - Hadoop for Humans

• Hopsworks is ready for prime time.
Hops Heads

Active:

Alumni: