

Overview of Tools in the HDFS Ecosystem

Lecture BigData Analytics

Julian M. Kunkel

julian.kunkel@googlemail.com

University of Hamburg / German Climate Computing Center (DKRZ)

29-01-2016



Outline

- 1 Hadoop Ecosystem
 - 2 Supporting Tools
 - 3 More Frameworks
 - 4 Summary

1 Hadoop Ecosystem

- Hortonworks
- Cloudera
- Supporting Tools

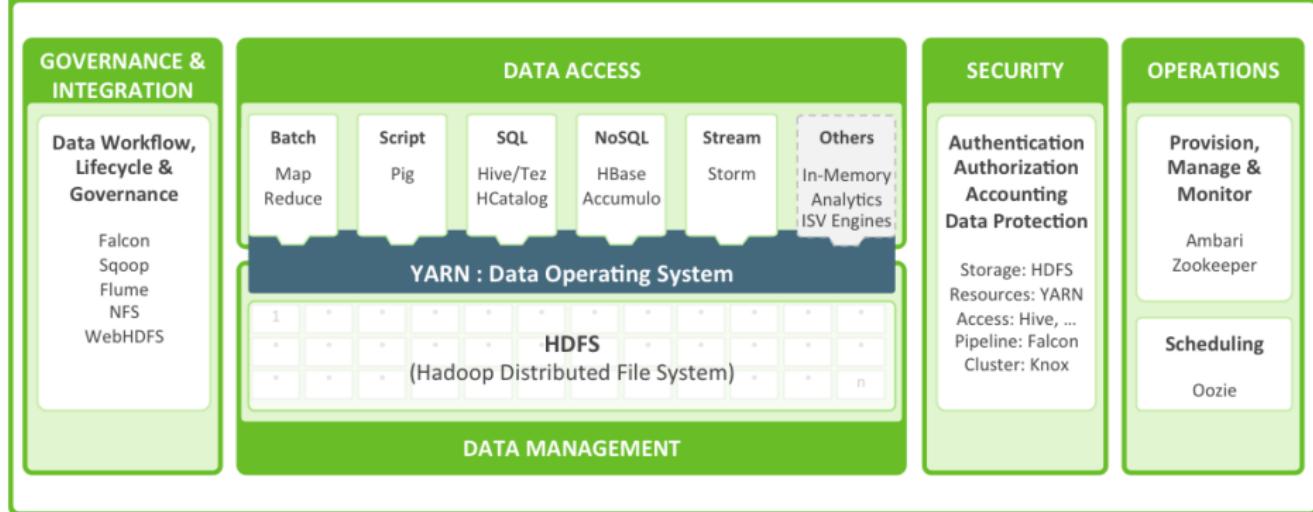
2 Supporting Tools

3 More Frameworks

4 Summary

Hortonworks

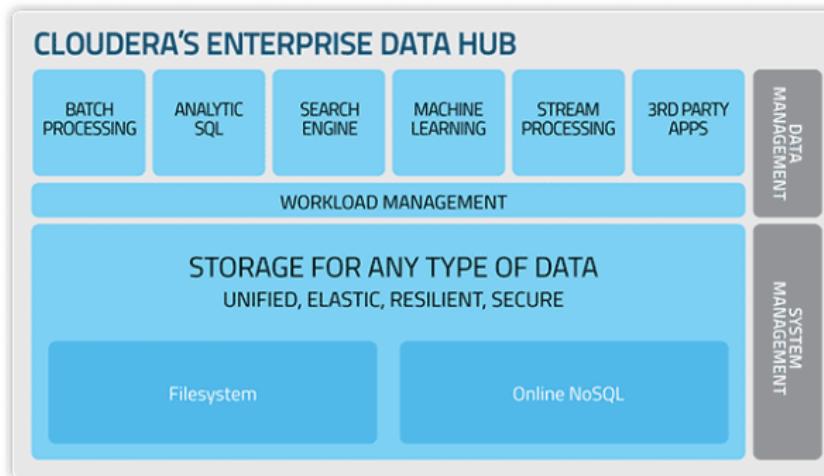
Hortonworks Data Platform



Source: Defining Enterprise Hadoop. Hortonworks [12]

Cloudera Enterprise Hadoop Ecosystem [25]

- Cloudera offers support, services and tools around Hadoop
 - Unified architecture: common infrastructure and data pool for tools
 - Build with open-source



Source: [26]

Supporting Tools¹

- Ambari: A Tool for Managing Hadoop Clusters
 - Hue: Manage „BigData“ projects in a browser
 - ZooKeeper: coordination/configuration service for services
 - Sqoop: ETL between HDFS and structured data stores
 - Oozie: Workflow scheduler (schedules/triggers workflows)
 - Falcon: Data governance engine for data pipelines
 - Flume: collecting, aggregating and moving large streaming event data
 - Kafka: publish-subscribe distributed messaging system
 - knox: REST API gateway (for all services)
 - Ranger: Integrate ACL permissions into Hadoop (ecosystem)
 - Slider: YARN application supporting monitoring and dynamic scaling of non-YARN apps

¹<https://hadoop.apache.org/>

1 Hadoop Ecosystem

2 Supporting Tools

- Ambari
 - Hue
 - Zeppelin
 - Oozie
 - Falcon
 - Sqoop
 - Slider
 - Knox
 - Atlas

3 More Frameworks

4 Summary

Ambari: A Tool for Managing Hadoop Clusters

- Convenient tool managing 10+ Apache tools
 - Supports installation and management
 - Dealing with data dependencies
 - Service startup
 - Monitoring of health and performance
 - (Re)configuration of services

Management with Ambari: Dashboard

The screenshot shows the Ambari Metrics dashboard with a sidebar on the left listing various services: Ambari, wr, HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Ambari Metrics, Kafka, Knox, Slider, and Spark. The main area displays a grid of 16 cards with metrics:

- HDFS Disk Usage: 7% (Circular gauge)
- DataNodes Live: 5/5
- HDFS Links: NameNode, Secondary NameNode, 5 DataNodes (More...)
- Memory Usage: (Line chart)
- CPU Usage: 2% (Bar chart)
- Cluster Load: (Horizontal bar chart)
- NameNode Heap: 17% (Circular gauge)
- NameNode RPC: 0 ms
- NameNode CPU WIO: 0.0% (Circular gauge)
- NameNode Uptime: 1.2 hr
- HBase Master Heap: 2% (Circular gauge)
- HBase Links: HBase Master, 5 RegionServers, Master Web UI (More...)
- HBase Ave Load: 1
- HBase Master Uptime: 32.9 min
- ResourceManager Heap: 9% (Circular gauge)
- ResourceManager Uptime: 25.9 min
- NodeManagers Live: 5/5
- YARN Memory: 0% (Circular gauge)
- YARN Links: ResourceManager, 5 NodeManagers (More...)

At the bottom left is an "Actions" button, and at the top right are links for Dashboard, Services, Hosts (with a red alert), Alerts, Admin, and a user dropdown for "admin".

Screenshot from the WR-cluster Ambari

Management with Ambari: Configuration

Summary **Configs** **Quick Links** **Service Actions**

Restart Required: 1 Component on 1 Host **Restart**

Group: **HDFS Default (5)** Manage Config Groups Filter...

V2 admin 2 months ago Current V1 admin 2 months ago

V2 Current admin authored on Tue, Jul 07, 2015 19:05 Discard Save

NameNode

NameNode hosts	abu1.cluster
NameNode directories	/tmp/hadoop/hdfs/namenode,/bigdata/hdfs/namenode
NameNode Java heap size	95744 MB
NameNode new generation size	23936 MB
NameNode maximum new generation size	23936 MB
NameNode permanent generation size	128 MB

Hue [12]: Lightweight Web Server for Hadoop

- Manage BigData projects in a browser
 - Supports: Hadoop ecosystem
 - HDFS, Pig, Sqoop, Hive, Impala, MapReduce, Spark, ...

Features

- Data upload/download
 - Management of HCatalog tables
 - Query editor (Hive, Pig, Impala)
 - Starting and monitoring of jobs

Hue: Lightweight Web Server for Hadoop

HUE  Query Editors  Data Browsers  Workflows  Suche  Security  Datei-Browser  Job-Browser  51emffu  ? 

Oozie-Dashboard Workflows **Coordinators** Bundles SLA Oozie

Nach Benutzernamen, Namen usw. suchen  Fortsetzen  Unterbrechen  Beenden  Nur Folgende anzeigen  1 7 15 30 Tage mit Status  Erfolgreich  Aktiv  Beendet 

Aktiv

<input type="checkbox"/> Nächste Übermittlung	Status	Name	Fortschritt	Sender	Häufigkeit	Startzeit	ID
Keine Daten verfügbar							

0 bis 0 von 0 Einträgen werden angezeigt  Zurück  Weiter 

Abgeschlossen

Fertigstellung	Status	Name	Dauer	Sender	Häufigkeit	Startzeit	ID
Wed, 30 Sep 2015 22:41:00	KILLED	My_Coordinator	93d:14h:56m:0s	dttqo8j	*/1 * * * *	Mon, 29 Jun 2015 07:45:00	0000304-150621143055208-oozie-oozi-C
Mon, 07 Sep 2015 17:05:00	KILLED	My_Coordinator	7d:0h:0m:0s	a309ve7	30 1 * * *	Mon, 31 Aug 2015 17:05:00	0000094-150828163545629-oozie-oozi-C
Tue, 25 Aug 2015 13:15:00	KILLED	My_Coordinator	7d:0h:0m:0s	4k0susv	17 0 * * *	Tue, 18 Aug 2015 13:15:00	0000507-150730175918991-oozie-oozi-C
Tue, 25 Aug 2015 13:13:00	SUCCEEDED	My_Coordinator	7d:0h:0m:0s	9jalpv9	1 0,6 * * *	Tue, 18 Aug 2015 13:13:00	0000504-150730175918991-oozie-oozi-C

Monitoring Oozie Workflows (Live system on gethue.com)

Hue: Lightweight Web Server for Hadoop

HUE Datei-Browser Data Browsers Workflows Suche Security

Nach Dateinamen suchen Aktionen In Papierkorb verschieben Hochladen Neu

Startseite / user / 51emffu Verlauf Papierkorb

	Name	Größe	Benutzer	Gruppe	Berechtigungen	Datum
<input type="checkbox"/>	..		hdfs	supergroup	drwxr-xr-x	September 17, 2015 02:46 AM
<input type="checkbox"/>	.		51emffu	51emffu	drwxr-xr-x	September 17, 2015 02:39 AM
<input type="checkbox"/>	.staging		51emffu	51emffu	drwx----	September 17, 2015 02:46 AM
<input type="checkbox"/>	oozie-oozi		51emffu	51emffu	drwxr-xr-x	September 17, 2015 02:40 AM

Anzeigen 45 von 2 Elemente Seite 1 of 1 << <> >>

File browser (Live system on gethue.com)

Hue: Lightweight Web Server for Hadoop

The screenshot shows the Hue Hive Editor interface. The top navigation bar includes links for 'Query Editors', 'Data Browsers', 'Workflows', 'Suche', 'Security', 'Datei-Browser', 'Job-Browser', and '51emffu'. The left sidebar has sections for 'Unterstützung' and 'Einstellungen', and a 'DATENBANK' dropdown set to 'default'. Below it is a 'Tabellenname...' input field and a list of tables: sample_07, sample_06, code (string), description (string), total_emp (int), and salary (int). The main area features a query editor with the following code:

```
1 SELECT * FROM sample_08
2 WHERE salary < 100000
```

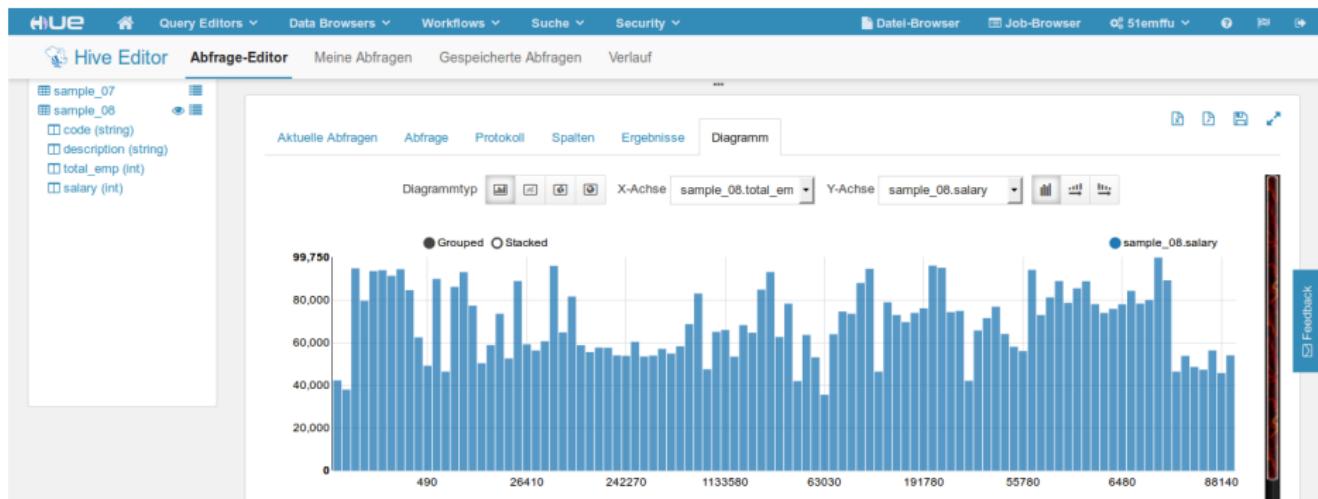
Below the editor are buttons for 'Ausführen', 'Speichern unter...', 'Erklären', 'oder erstellen Sie eine', and 'Neue Abfrage'. The results section shows a table with columns: sample_08.code, sample_08.description, sample_08.total_emp, and sample_08.salary. The data is as follows:

sample_08.code	sample_08.description	sample_08.total_emp	sample_08.salary
0	All Occupations	135185230	42270
1	Legislators	64650	37980
2	Advertising and promotions managers	36100	94720
3	Administrative services managers	246930	79500
4	Compensation and benefits managers	38810	93410
5	Training and development managers	29350	93830
6	Industrial production managers	154030	91200

Query editor (Live system on gethue.com)



Hue: Lightweight Web Server for Hadoop



Visualizing query results in diagrams (Live system on gethue.com)

Zeppelin [39]

- Web-based notebook for interactive data analytics
 - Add code snippets
 - Arrange them
 - Execute them
 - Visualizes results
 - Supports Spark, Scala, psql, R
 - Collaborative environment
 - Can be embedded into a webpage
 - A bit premature (currently incubating)

Zeppelin



Notebook ▾ Interpreter

Zeppelin Tutorial



```
%nd
## Welcome to Zeppelin.
##### This is a live tutorial, you can run the code yourself. (Shift-Enter to Run)

Took 1 seconds (outdated)
```

FINISHED ▶ 🔍 🌐

Load Data Into Table

```

import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset

// Zeppelin creates and injects sc (SparkContext) and sqlContext (HiveContext or SQLContext)
// So you don't need create them manually

// load bank data
val bankText = sc.parallelize(
    IOUtils.toString(
        new URL("https://s3.amazonaws.com/apache-zeppelin/tutorial/bank/bank.csv"),
        Charset.forName("utf8")).split("\n"))

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bank = bankText.map(s => s.split(",")).filter(s => s(0) != "\"age\"").map(
    s => Bank(s(0).toInt,
              s(1).replaceAll("\"", ""),
              s(2).replaceAll("\"", ""),
              s(3).replaceAll("\"", ""),
              s(5).replaceAll("\"", "")).toInt
    )
).toDF()
bank.registerTempTable("bank")

```

Too many open files

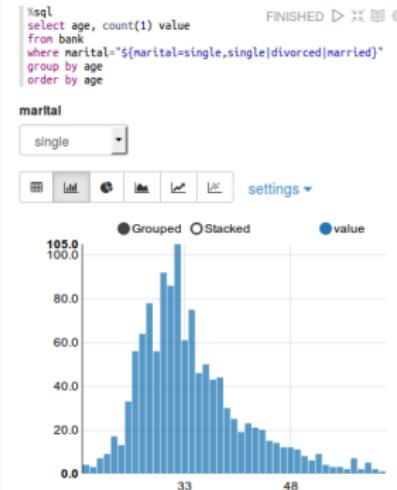
Took 0 seconds (outdated)

```
%sql  
select age, count(1) value  
from bank  
where age < 30  
group by age  
order by age
```

ERROR ▶ ✖️ 🗑️⚙️

```
%sql  
select age, count(1) value  
from bank  
where age < ${maxAge=30}  
group by age  
order by age
```

ERROR ▶ ✎ ☐ ⚙



Took 1 seconds (outdated)

Oozie [15, 16]

- Scalable, reliable and extensible workflow scheduler
 - Jobs are DAGs of actions specified in XML workflows
 - Actions: Map-reduce, Pig, Hive, Sqoop, Spark, Shell actions
 - Workflows can be parameterized
 - Triggers notifications via HTTP GET upon start/end of a node/job
 - Automatic user-retry to repeat actions when fixable errors occur
 - Monitors a few runtime metrics upon execution
 - Interfaces: command line tools, web-service and Java APIs
 - Integrates with HCatalog
 - Coordinator jobs trigger jobs
 - By time schedules
 - When data becomes available
 - Requires polling of HDFS (1-10 min intervals)
 - With HCatalog's publish-subscribe, jobs can be started immediately
 - Can record events for service level agreement

Workflows [16]

- A workflow application is a ZIP file to be uploaded
 - Includes workflow definition and coordinator job
 - Bundles scripts, JARs, libraries needed for execution
 - Workflow definition is a DAG with control flow and action nodes
 - Control flow: start, end, decision, fork, join
 - Action nodes: whatever to execute
 - Variables/Parameters²
 - Default values can be defined in a config-default.xml in the ZIP
 - Expression language functions help in parameterization¹
 - Basic functions: timestamp(), trim(), concat(s1, s2)
 - Workflow functions: wf:errorCode(< action node >)
 - Action specific functions:
`hadoop:counters("mr-node")["FileSystemCounters"]["FILE_BYTES_READ"]`
 - Coordinator job is also an XML file

²They are used with with \${NAME/FUNCTION}, e.g. \${timestamp()}

Coordinator Jobs [17]

App which periodically starts a workflow (every 60 min)

```
1 <coordinator-app name="MY_APP" frequency="60" start="2009-01-01T05:00Z" end="2009-01-01T06:00Z" timezone="UTC">
2   <!-- xmlns="uri:oozie:coordinator:0.1" -->
3   <action>
4     <workflow>
5       <app-path>hdfs://localhost:9000/tmp/workflows</app-path>
6     </workflow>
7   </action>
8 </coordinator-app>
```

Every 24h check if dependencies for a workflow are met, then run it

```
1 <coordinator-app name="MY_APP" frequency="1440" start="2009-02-01T00:00Z" end="2009-02-07T00:00Z" ...>
2   <datasets> <-- check for existence of this URI -->
3     <dataset name="input1" frequency="60" initial-instance="2009-01-01T00:00Z" timezone="UTC">
4       <uri-template>hdfs://localhost:9000/tmp/revenue_feed/${YEAR}/${MONTH}/${DAY}/${HOUR}</uri-template>
5     </dataset>
6   </datasets>
7   <input-events> <-- we depend on the last 24 hours input data -->
8     <data-in name="coordInput1" dataset="input1">
9       <start-instance>${coord:current(-23)}</start-instance>
10      <end-instance>${coord:current(0)}</end-instance>
11    </data-in>
12  </input-events>
13  <action>
14    <workflow>
15      <app-path>hdfs://localhost:9000/tmp/workflows</app-path>
16    </workflow>
17  </action>
18 </coordinator-app>
```

Example Oozie Workflow [13]

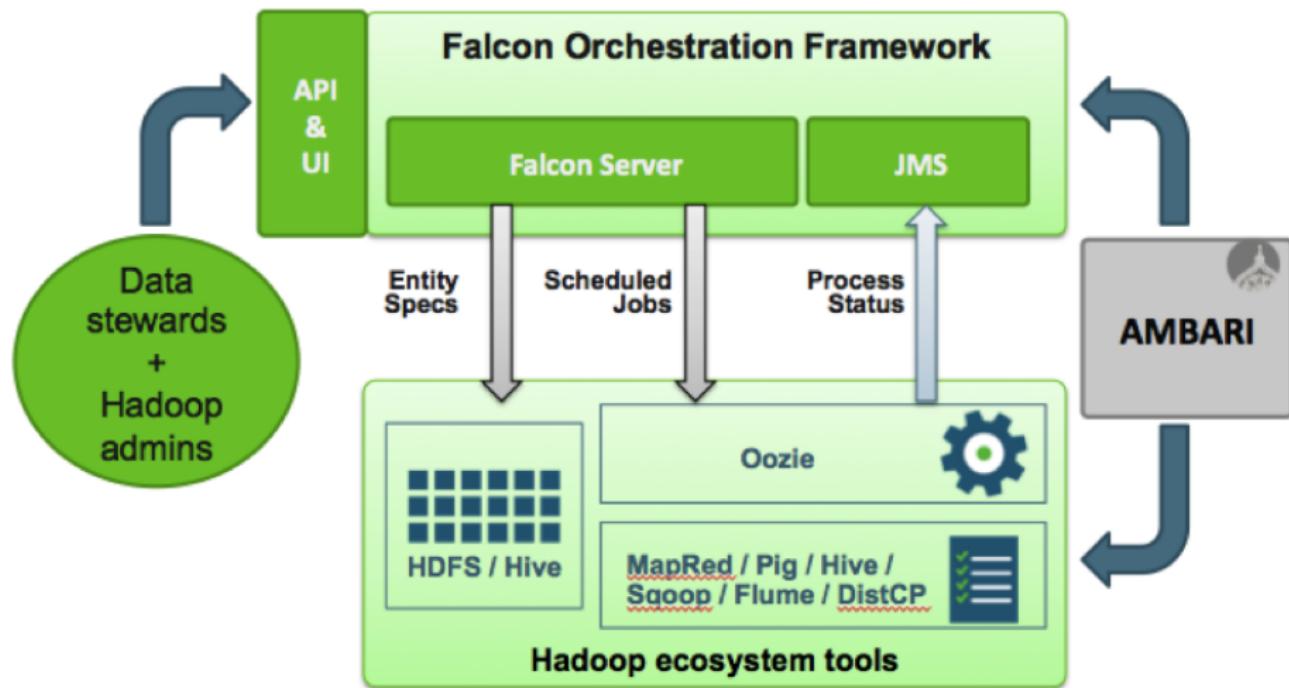
Three actions: Execute pig script, concatenate reducer files, upload files to a remote via ssh

```
1 <workflow-app xmlns="uri:oozie:workflow:0.2" name="sample-wf">
2   <start to="pig" />
3   <action name="pig">
4     <pig><job-tracker>${jobTracker}</job-tracker>
5       <name-node>${nameNode}</name-node>
6       <prepare><delete path="${output}" /></prepare>
7       <configuration>
8         <property> <name>mapred.job.queue.name</name><value>${queueName}</value></property>
9         <property> <name>mapreduce.fileoutputcommitter.marksuccessfuljobs</name><value>true</value></property>
10      </configuration>
11      <script>${nameNode}/projects/bootcamp/workflow/script.pig</script>
12      <param>input=${input}</param>
13      <param>output=${output}</param>
14      <file>lib/dependent.jar</file>
15    </pig><ok to="concatenator" /><error to="fail" /> <-- the concatenator action is not shown here -->
16  </action>
17
18  <action name="fileupload">
19    <ssh><host>localhost</host>
20    <command>/tmp/fileupload.sh</command>
21    <args>${nameNode}/projects/bootcamp/concat/data-${fileTimestamp}.csv</args><args>${wf:conf("ssh.host")}</args>
22    <capture-output/></ssh>
23  <ok to="fileUploadDecision" /><error to="fail"/>
24  </action>
25
26  <decision name="fileUploadDecision" > <-- check the exit status of the file upload -->
27    <switch><case to="end">${wf:actionData('fileupload')['output']} == '0'</case><default to="fail"/> </switch>
28  </decision>
29
30  <kill name="fail"><message>Workflow failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message></kill>
31  <end name="end" />
32 </workflow-app>
```

Falcon [11,13]

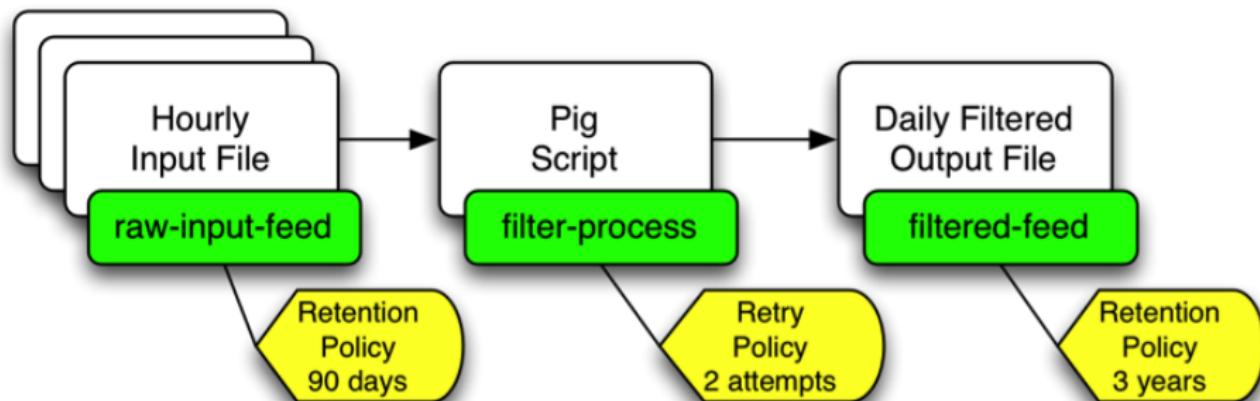
- Feed (data set) management and processing system
- Simplifies dealing with many Oozie jobs
- Provides data governance
 - Define and run data pipelines (management policies)
 - Monitor data pipelines
 - Trace pipelines to identify dependencies and perform audits
- Data model defines entities describing policies and pipelines
 - Clusters define resources and interfaces to use
 - Feeds define frequency, data retention, input, outputs, retry and use clusters (multiple for replication)
 - Process: processing task, i.e. Oozie workflow, Hive or Pig script
- Features
 - Supports reuse of entities for different workflows
 - Enables replication across clusters and data archival
 - Supports HCatalog
 - Notification of users upon availability of feed groups

Falcon: High-level Architecture



Source: [11]

Falcon: Example Pipeline



Source: [11]

Falcon: Example Process Definition [11, 14]

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- Sample process. Runs at 6th hour every day. Input: last day hourly data. Output: for yesterday -->
3 <process name="SampleProcess">
4   <cluster name="wr" />
5   <frequency>days(1)</frequency>
6
7   <validity start="2015-04-03T06:00Z" end="2022-12-30T00:00Z" timezone="UTC" />
8
9   <inputs>
10    <input name="input" feed="SampleInput" start="yesterday(0,0)" end="today(-1,0)" />
11  </inputs>
12
13  <outputs>
14    <output name="output" feed="SampleOutput" instance="yesterday(0,0)" />
15  </outputs>
16
17  <properties>
18    <property name="queueName" value="reports" />
19    <property name="ssh.host" value="host.com" />
20    <property name="fileTimestamp" value="${coord:formatTime(coord:nominalTime(), 'yyyy-MM-dd')}" />
21  </properties>
22
23  <workflow engine="oozie" path="/projects/bootcamp/workflow" />
24
25  <retry policy="backoff" delay="minutes(5)" attempts="3" />
26
27  <!-- How to check and handle late arrival of input data-->
28  <late-process policy="exp-backoff" delay="hours(1)">
29    <late-input input="input" workflow-path="/projects/bootcamp/workflow/lateinput" />
30  </late-process>
31 </process>
```

Sqoop [18, 19]

- Transfers bulk data between Hadoop and RDBMS, either
 - One/multiple tables (preserving their schema)
 - Results of a free-form SQL query
- Uses MapReduce to execute import/export jobs
 - Parallelism is based on splitting one column's value
- Validate data transfer (comparing row counts) for full tables
- Save jobs for repeated invocation
- Main command line tool `sqoop`, more specific tools `sqoop*`

Features [19]

Import Features

- Incremental import (scan and add only newer rows)
- File formats: CSV, SequenceFiles, Avro, Parquet
 - Compression support
- Outsource large BLOBS/TEXT into additional files
- Import into Hive (and HBase)
- Can create the table schema in HCatalog automatically
 - With HCatalog, only CSV can be imported

Export Features

- Bulk insert: 100 records per statement
- Periodic commit after 100 statements

Import Process [19]

- Read the schema of the source table
- Create a Java class representing a row of the table
 - This class can be used later to work with the data
- Start MapReduce to load data parallel into multiple files
 - The number of mappers can be configured
 - Mappers work on different values of the splitting column
 - The default splitting column is the primary key
 - Determines min and max value of the key
 - Distributes fixed chunks to mappers
- Output status information to the MapReduce job tracker

Example Imports [19]

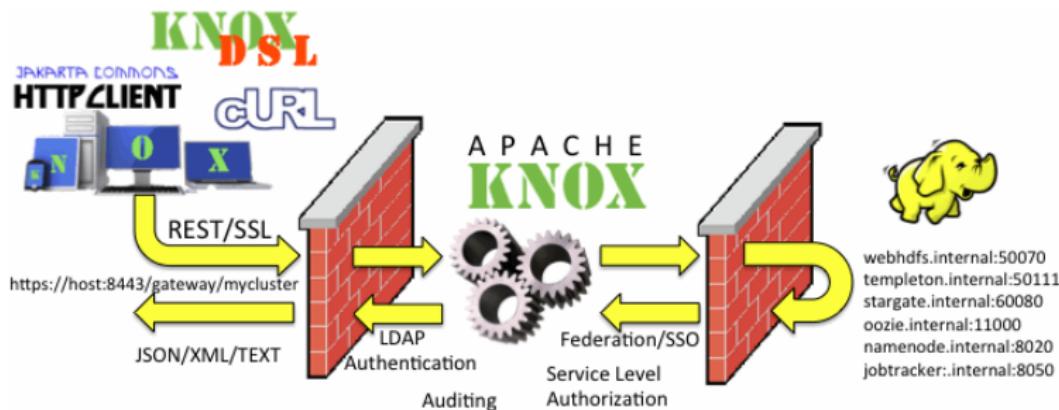
```
1 # Import columns from "foo" into HDFS to /home/x/foo (table name is appended)
2 # When not specifying any columns, all columns will be imported.
3 $ sqoop import --connect jdbc:mysql://localhost/db --username foo --table TEST --columns
   ↪ "matrikel,name" --warehouse-dir /home/x --validate
4
5 # We'll use a free-form query, it is parallelized on the split-by column
6 # The value is set into the magic $CONDITIONS variable
7 $ sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id) WHERE
   ↪ $CONDITIONS' --split-by a.id --target-dir /user/foo/joinresults
8
9 # To create the HCatalog table use --hcatalog-table or --hive-import
10 # See [19] for details of the available options
```

Slider [20]

- Is a YARN application that manages non-YARN apps on a cluster
- ⇒ Utilize YARN for resource management
- Enables installation, execution, monitoring and dynamic scaling
- Command line tool slider
- Apps are installed and run from a package
 - Tarball with well-defined structure [21]
 - Scripts for installing, starting, status, ...
- Example packages: jmemcached, HBase
- Slider is currently extended to deploy Docker images

Knox: Security for Hadoop [22]

- REST API Gateway for Hadoop ecosystem services
 - Supports: HDFS, Hcatalog, HBase, Oozie, Hive, Yarn, Storm
 - Supports multiple clusters
- Provides authentication, federation/SSO, authorization, auditing
- Enhances security providing central control and protection
 - SSL encryption
 - Authentication: LDAP, Active Directory, Kerberos
 - Authorization: ACL's (user, group, IP) on service level³



Source: [22]

Example Accesses via the REST API [22]

List a HDFS directory

```
1 curl -i -k -u guest:guest-password -X GET  
    ↳ 'https://localhost:8443/gateway/sandbox/webhdfs/v1/?op=LISTSTATUS'
```

Example response

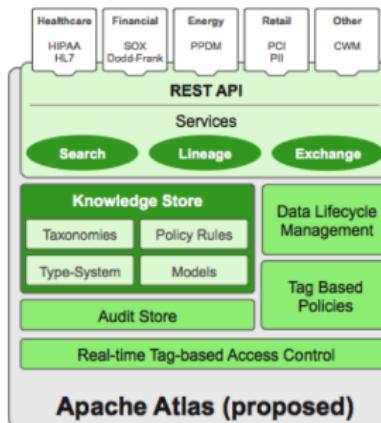
```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3 Content-Length: 450
4 Server: Jetty(6.1.26)
5
6 {"FileStatuses": {"FileStatus": [
7     {"accessTime": 0, "blockSize": 0, "group": "hdfs", "length": 0,
8         "modificationTime": 1350595859762, "owner": "hdfs", "pathSuffix": "apps",
9         "permission": "755", "replication": 0, "type": "DIRECTORY"},

10    {"accessTime": 0, "blockSize": 0, "group": "mapred", "length": 0,
11        "modificationTime": 1350595874024, "owner": "mapred", "pathSuffix": "mapred",
12        "permission": "755", "replication": 0, "type": "DIRECTORY"},

13 ]}}
```

Atlas [23]

- A proposed⁴ framework for platform-agnostic data governance
- Exchange metadata with other tools
- Audit operations, explore history of data and metadata
- Support lifecycle management workflows built with Falcon
- Support Ranger access control (ACL's)



Source: [23]

⁴It is already shipped with Ambari

1 Hadoop Ecosystem

2 Supporting Tools

3 More Frameworks

- Drill
 - Impala
 - Solr
 - Mahout

4 Summary

Drill [10, 29, 30]

- Software framework for data-intensive distributed applications
 - Data model: relational (ANSI SQL !) + schema-free JSON
 - Analyse data in-situ without data movement
 - Execute one query against multiple NoSQL datastores
 - Datastores: HBase, MongoDB, HDFS, S3, Swift, local files
 - Features
 - REST APIs
 - Columnar execution engine supporting complex data
 - Locality-aware execution
 - Cost-based optimizer pushing processing into datastore
 - Runtime compilation of queries

```
1 # Different datastores, localstorage, mongodb and s3
2 SELECT * FROM dfs.root.'/web/logs';
3 SELECT country, count(*) FROM mongodb.web.users GROUP BY country;
4 SELECT timestamp FROM s3.root.'clicks.json' WHERE user_id = 'jdoe';
5
6 # Query JSON: access the first student's age from private data (a map)
7 SELECT student[0].private.AGE, FROM dfs.'students.json';
```

Cloudera Impala [25, 26]

- Enterprise analytic database
 - Utilizes HDFS, HBase and Amazon S3
 - Based on Google Dremel like Apache Drill
 - Written in C++, Java
 - Massively-parallel SQL engine
 - Supports HiveQL and subset of ANSI-92 SQL
 - Uses LLVM to generate efficient code for queries

Solr [10, 31]

- Full-text search and indexing platform
 - REST API: index documents and query via HTTP
 - Query response in JSON, XML, CSV, binary
 - Features
 - Data can be stored on HDFS
 - High-availability, scalable and fault tolerant
 - Distributed search
 - Faceted classification: organize knowledge into a systematic order using (general or subject-specific) semantic categories that can be combined for a full classification entry [10]
 - Geo-spatial search
 - Caching of queries, filters and documents
 - Uses lucene library for search
 - Similar tools: Elasticsearch [33]

Example Query [32]

Identifying available facets terms and number of docs for each

```
1 curl http://localhost:8983/solr/gettingstarted/select?wt=json&indent=true&q=*&rows=0&facet=true& facet.field=manu_id_s
```

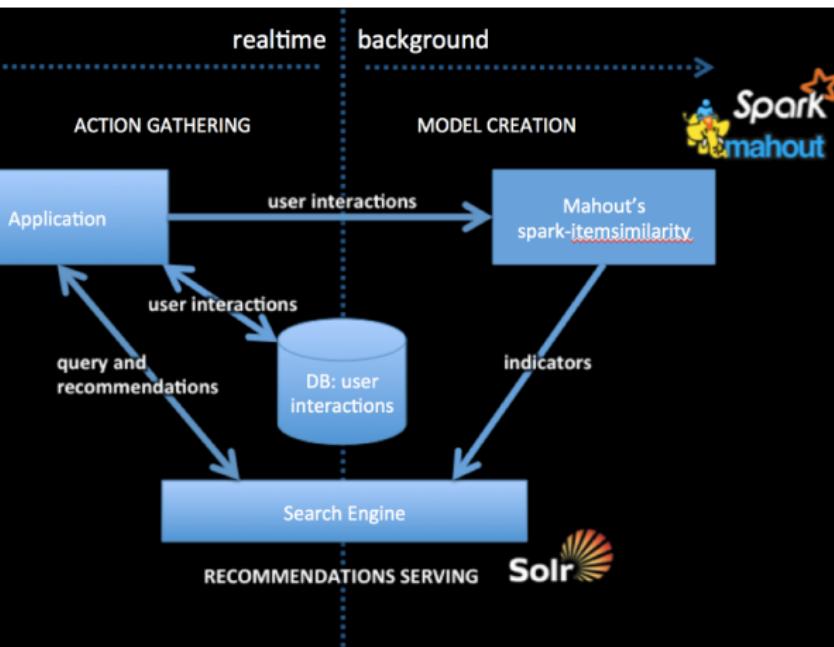
Response

```
1 {
2   "responseHeader": {
3     "status":0,
4     "QTime":3,
5     "params": { /* Parameters of the query */
6       "facet":"true", "indent":"true", "q":":*", "facet.field":"manu_id_s", "wt":"json",
7       "rows":"0"}},
8   "response":{ "numFound":2990, "start":0, "docs":[]}, /* number of documents found */
9   "facet_counts": {
10     "facet_queries":{},
11     "facet_fields": { /* the available facets and number of documents */
12       "manu_id_s": ["corsair", 3, "belkin", 2, "canon", 2, "apple", 1, "asus", 1, "ati", 1, "boa", 1, "dell", 1, "eu", 1, "maxtor", 1,
13         "nor", 1, "uk", 1, "viewsonic", 1, "samsung", 0]},
14     "facet_dates":{},
15     "facet_ranges":{},
16     "facet_intervals":{}}
17 }
```

Mahout [34]

- Framework for scalable machine learning
 - Collaborative filtering
 - Classification
 - Clustering
 - Dimensionality reduction
 - Recommender
 - history: user purchases + all purchases \Rightarrow recommendations (user)
 - Computation on Spark, MapReduce, H2O engines [36]
 - Can also use a single machine without Hadoop
 - Algorithm availability depends on the backend
 - Bindings for Scala language [35]
 - Provide distributed BLAS, Row Matrix (DRM)
 - R-like DSL embedded in Scala
 - Algebraic optimizer

Recommender Architecture



- 1 Collect user interactions
 $n \times (\text{user-id}, \text{item-id})$
- 2 Learning:
 1. Itemsimilarity creates item, list-of-similar-items
 2. Store those tuples in the search engine
- 3 Query search engine with n latest user interactions
- 4 If they occur in the list-of-similar-items, recommend item

Source: [36]

Summary

- The (Apache) Hadoop community is active
 - Software responsibilities:
 - Hadoop deployment and cluster management
 - Data management and provenance
 - Security
 - Analysis
 - Automation (scheduling, data ingestion)
 - Many software packages are used but still in Apache incubator (beta)

Bibliography

- 10 Wikipedia
 - 11 <http://hortonworks.com/blog/introduction-apache-falcon-hadoop/>
 - 12 <http://gethue.com/>
 - 13 <http://falcon.apache.org/>
 - 14 <http://falcon.apache.org/0.7/EntitySpecification.html>
 - 15 <http://oozie.apache.org/>
 - 16 <http://oozie.apache.org/docs/4.2.0/index.html>
 - 17 <https://github.com/yahoo/oozie/wiki/Oozie-Coord-Use-Cases>
 - 18 <http://sqoop.apache.org/>
 - 19 <http://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html>
 - 20 <http://slider.incubator.apache.org/>
 - 21 http://slider.incubator.apache.org/docs/slider_specs/application_package.html
 - 22 <https://knox.apache.org/>
 - 23 <http://hortonworks.com/blog/apache-atlas-project-proposed-for-hadoop-governance/>
 - 24 <http://ranger.incubator.apache.org/>
 - 25 <http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html>
 - 26 <http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise.htm>
 - 27 <https://github.com/cloudera/impala>
 - 28 <http://incubator.apache.org/drill/>
 - 29 <https://drill.apache.org/docs/json-data-model/>
 - 30 <http://lucene.apache.org/solr/features.html>
 - 31 <http://lucene.apache.org/solr/quickstart.html>
 - 32 <http://solr-vs-elasticsearch.com/>
 - 33 <http://mahout.apache.org/>
 - 34 <http://www.scala-lang.org/>
 - 35 <http://h2o.ai/>
 - 36 <https://mahout.apache.org/users/recommender/quickstart.html>
 - 37 Free Ebook: <https://www.mapr.com/practical-machine-learning>
 - 38 <https://zeppelin.incubator.apache.org/>