Overview of Tools in the HDFS Ecosystem

Lecture BigData Analytics

Julian M. Kunkel

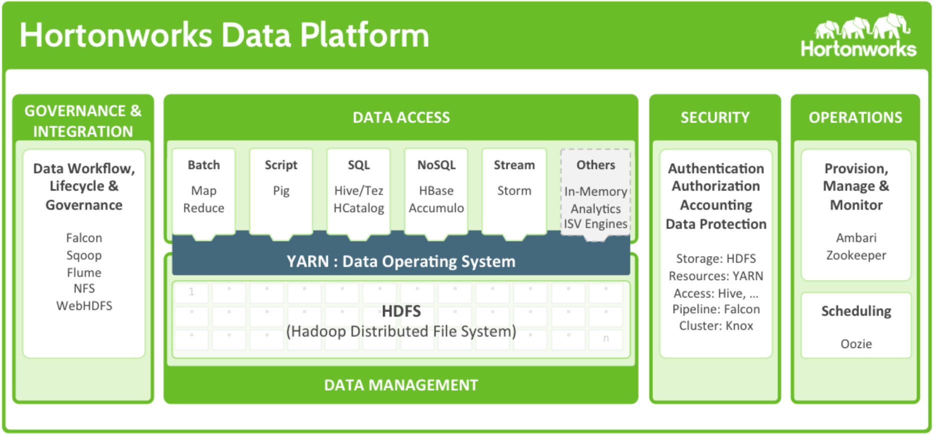
julian.kunkel@googlemail.com

University of Hamburg / German Climate Computing Center (DKRZ)

29-01-2016

Outline

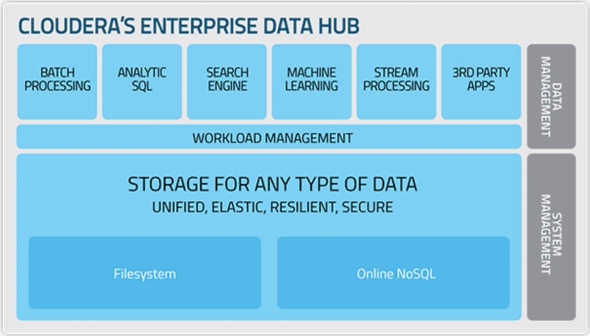
Hortonworks



Source: Defining Enterprise Hadoop. Hortonworks [12]

Cloudera Enterpise 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 Tools[1]

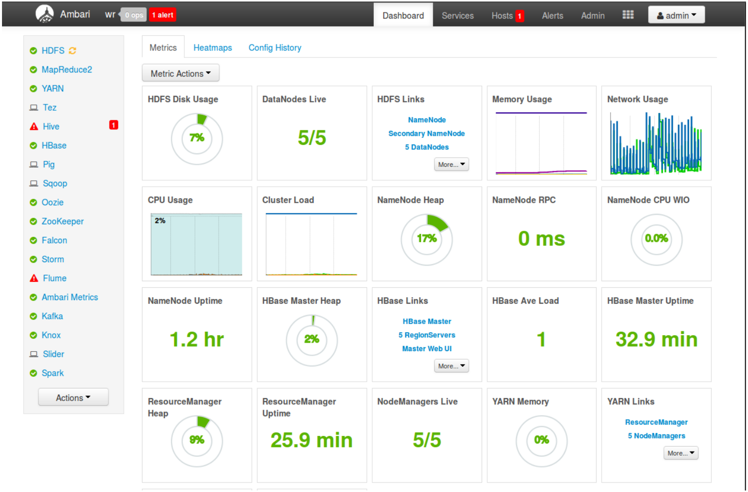
- Ambari: A Tool for Managing Hadoop Clusters
- Hue: Manage „BigData“ projects in a browser
- ZooKeeper: coordination/configuration service for services
- Sqoop: ETL between HDFS and structured data stores
- Oozie: Workflow scheduler (schedules/triggers workflows)
- Falcon: Data governance engine for 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/

Ambari: A Tool for Managing Hadoop Clusters

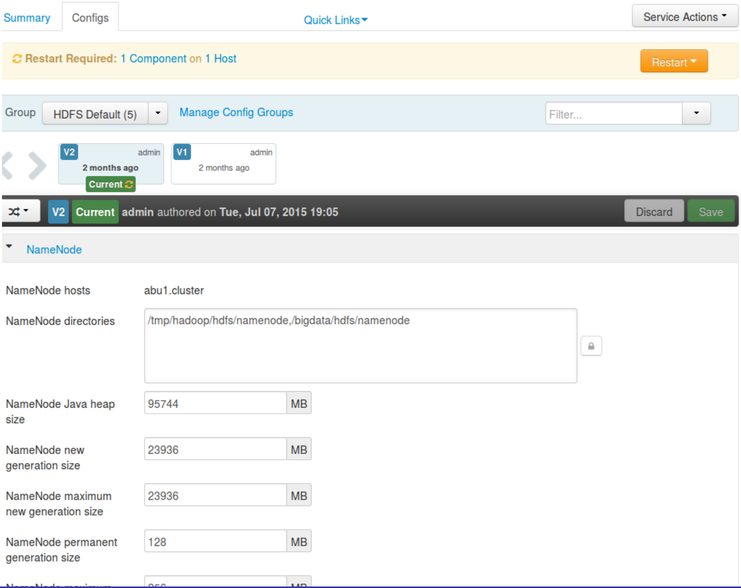
- Convenient tool managing 10+ Apache tools
- Supports installation and management
  - Dealing with data dependencies
  - Service startup
  - Monitoring of health and performance
  - (Re)configuration of services

Management with Ambari: Dashboard



Screenshot from the WR-cluster Ambari

Management with Ambari: Configuration



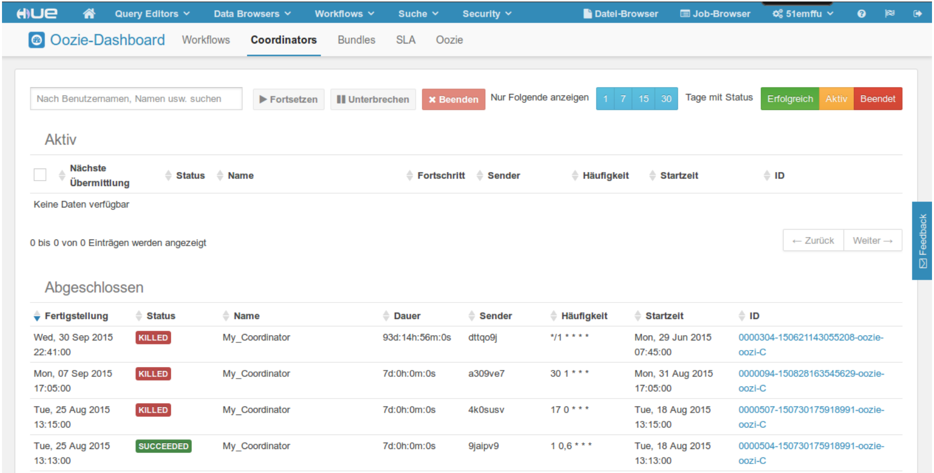
Hue [12]: Lightweight Web Server for Hadoop

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

Features

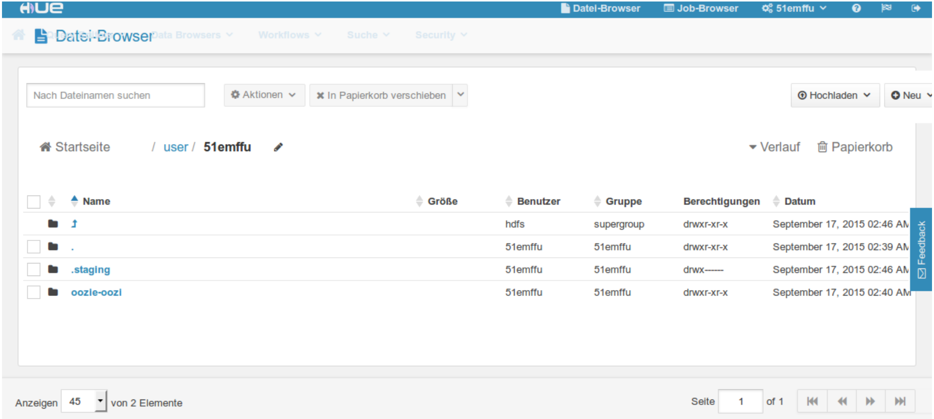
- Data upload/download
- Management of HCatalog tables
- Query editor (Hive, Pig, Impala)
- Starting and monitoring of jobs

Hue: Lightweight Web Server for Hadoop



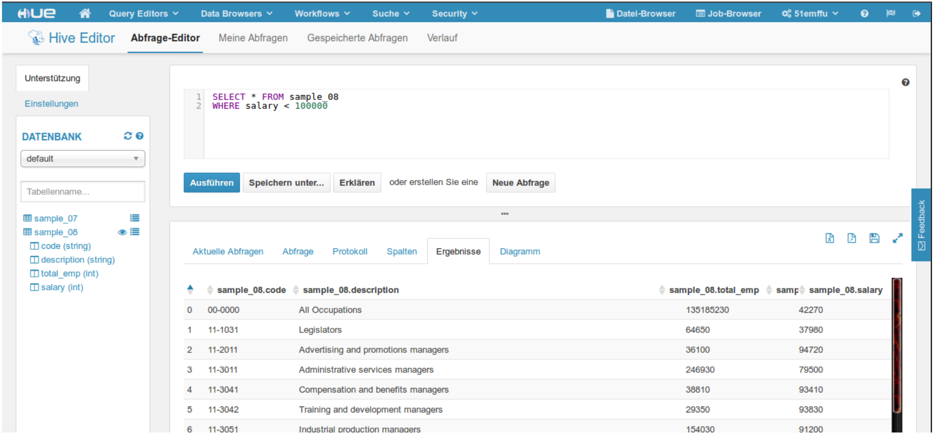
Monitoring Oozie Workflows (Live system on gethue.com)

Hue: Lightweight Web Server for Hadoop



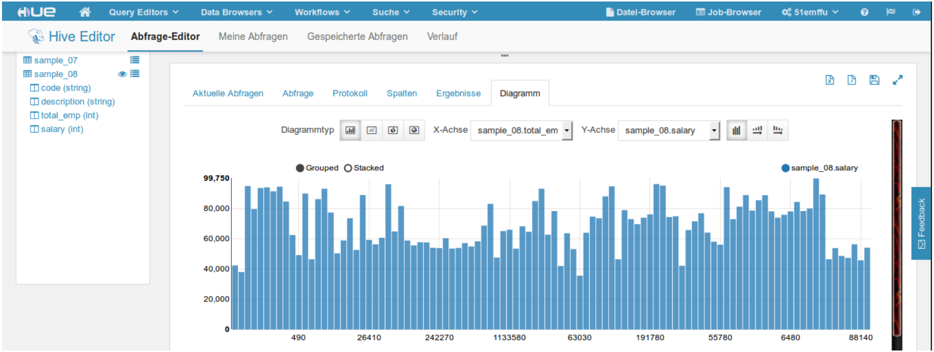
File browser (Live system on gethue.com)

Hue: Lightweight Web Server for Hadoop



Query editor (Live system on gethue.com)

Hue: Lightweight Web Server for Hadoop

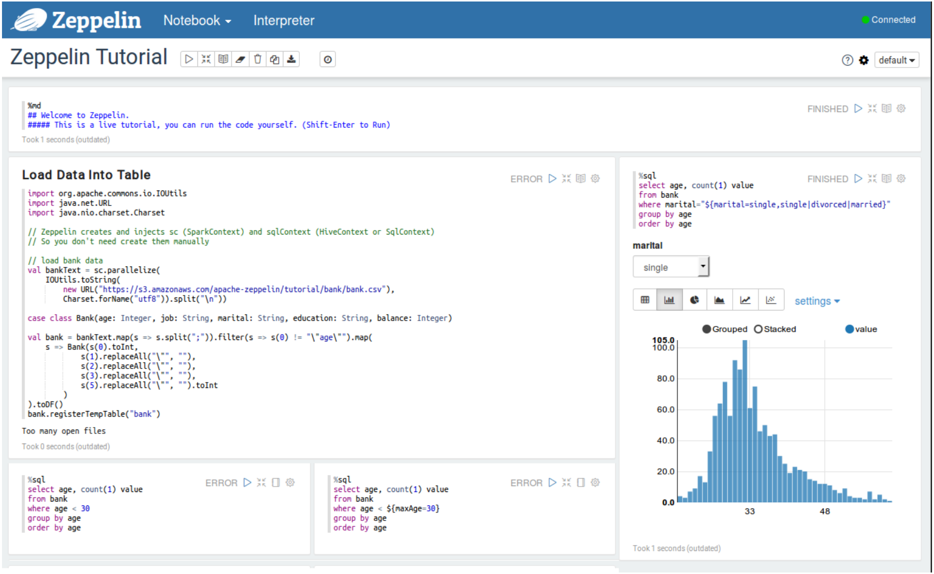


Visualizing query results in diagrams (Live system on gethue.com)

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


Zeppelin
Notebook
Interpreter
Connected
Zeppelin Tutorial
default
%md
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
ERROR
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 marital="${marital=single,single|divorced|married}"
group by age
order by age
FINISHED
marital
single
settings
Grouped
Stacked
value
105.0
100.0
80.0
60.0
40.0
20.0
0.0
33
48
Took 1 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

Oozie [15, 16]

- Scalable, reliable and extensible workfow 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 invervalls)
    - 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/Parameters [2]
  - Default values can be defined in a config-default.xml in the ZIP
- Expression language functions help in parameterization[1]
  - 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

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

Coordinator Jobs [17]

App which periodically starts a workflow (every 60 min)

```
<coordinator-app name="MY_APP" frequency="60" start="2009-01-01T05:00Z" end="2009-01-01T06:00Z" timezone="UTC"
      ↪ xmlns="uri:oozie:coordinator:0.1">
   <action>
      <workflow>
         <app-path>hdfs://localhost:9000/tmp/workflows</app-path>
      </workflow>
   </action>
</coordinator-app>
```

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

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

Example Oozie Workflow [13]

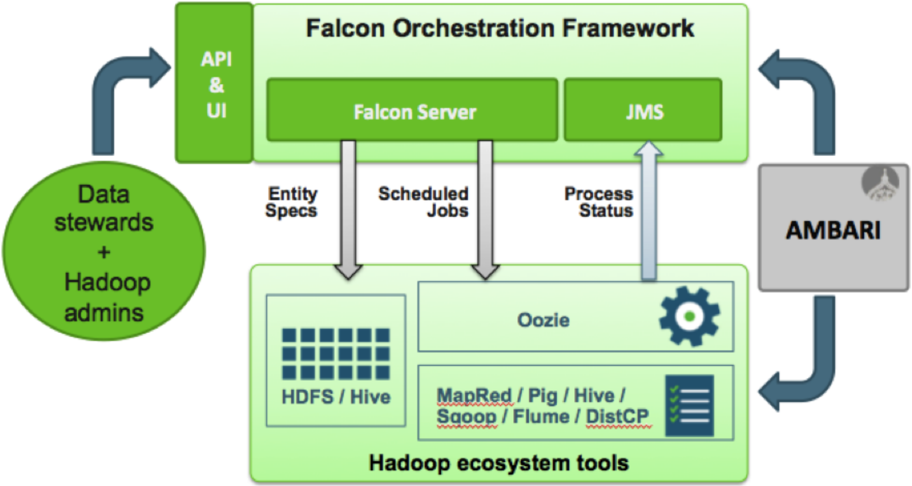
Three actions: Execute pig script, concatenate reducer files, upload files to a remote via ssh

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="sample-wf">
  <start to="pig" />
  <action name="pig">
    <pig><job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <prepare><delete path="${output}"/></prepare>
      <configuration>
        <property> <name>mapred.job.queue.name</name><value>${queueName}</value></property>
        <property> <name>mapreduce.fileoutputcommitter.marksuccessfuljobs</name><value>true</value></property>
      </configuration>
      <script>${nameNode}/projects/bootcamp/workflow/script.pig</script>
      <param>input=${input}</param>
      <param>output=${output}</param>
      <file>lib/dependent.jar</file>
    </pig><ok to="concatenator" /><error to="fail" /> <-- the concatenator action is not shown here -->
  </action>

  <action name="fileupload">
  <ssh><host>localhost</host>
    <command>/tmp/fileupload.sh</command>
    <args>${nameNode}/projects/bootcamp/concat/data-${fileTimestamp}.csv</args><args>${wf:conf("ssh.host")}</args>
    <capture-output/></ssh>
  <ok to="fileUploadDecision" /><error to="fail"/>
  </action>

  <decision name="fileUploadDecision"> <-- check the exit status of the file upload -->
    <switch><case to="end">${wf:actionData('fileupload')['output'] == '0'}</case><default to="fail"/> </switch>
  </decision>

  <kill name="fail"><message>Workflow failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message></kill>
  <end name="end" />
</workflow-app>
```

Falcon [11,13]

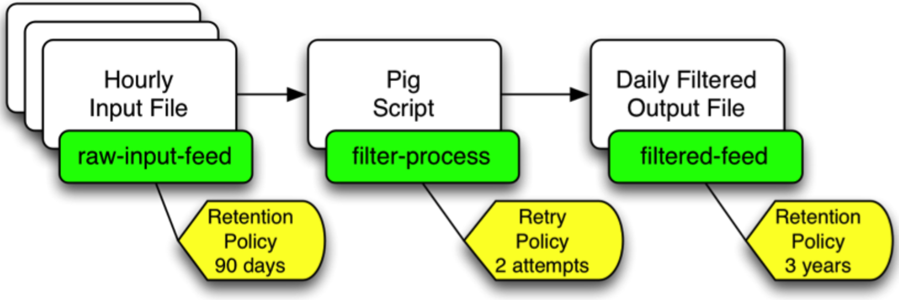
- Feed (data set) management and processing system
- Simplifies dealing with many Oozie jobs
- Provides data governance
  - Define and run data pipelines (management policies)
  - Monitor data pipelines
  - Trace pipelines to identify dependencies and perform audits
- Data model defines entities describing 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]

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sample process. Runs at 6th hour every day. Input: last day hourly data. Output: for yesterday  -->
<process name="SampleProcess">
    <cluster name="wr" />
    <frequency>days(1)</frequency>

    <validity start="2015-04-03T06:00Z" end="2022-12-30T00:00Z" timezone="UTC" />

    <inputs>
        <input name="input" feed="SampleInput" start="yesterday(0,0)" end="today(-1,0)" />
    </inputs>

    <outputs>
            <output name="output" feed="SampleOutput" instance="yesterday(0,0)" />
    </outputs>

    <properties>
        <property name="queueName" value="reports" />
        <property name="ssh.host" value="host.com" />
        <property name="fileTimestamp" value="${coord:formatTime(coord:nominalTime(), 'yyyy-MM-dd')}" />
    </properties>

    <workflow engine="oozie" path="/projects/bootcamp/workflow" />

    <retry policy="backoff" delay="minutes(5)" attempts="3" />

    <-- How to check and handle late arrival of input data-->
    <late-process policy="exp-backoff" delay="hours(1)">
        <late-input input="input" workflow-path="/projects/bootcamp/workflow/lateinput" />
    </late-process>
</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]

```
# Import columns from "foo" into HDFS to /home/x/foo (table name is appended)
# When not specifying any columns, all columns will be imported.
$ sqoop import --connect jdbc:mysql://localhost/db --username foo --table TEST --columns
    ↪ "matrikel,name" --warehouse-dir /home/x --validate

# We'll use a free-form query, it is parallelized on the split-by column
# The value is set into the magic $CONDITIONS variable
$ 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

# To create the HCatalog table use --hcatalog-table or --hive-import
# See [19] for details of the available options
```

Slider [20]

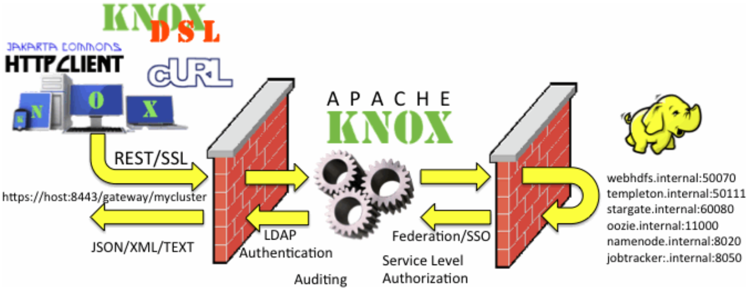
- Is a YARN application that manages non-YARN apps on a cluster

⇒ 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 level[3]



Source: [22]

Example Accesses via the REST API [22]

List a HDFS directory

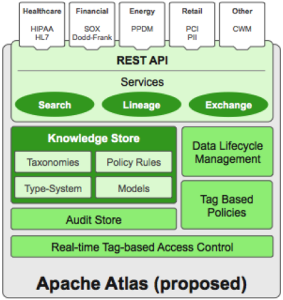
```
curl -i -k -u guest:guest-password -X GET
     ↪ 'https://localhost:8443/gateway/sandbox/webhdfs/v1/?op=LISTSTATUS'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 450
Server: Jetty(6.1.26)

{"FileStatuses":{"FileStatus":[
{"accessTime":0,"blockSize":0,"group":"hdfs","length":0,
     ↪ "modificationTime":1350595859762, "owner":"hdfs", "pathSuffix":"apps",
     ↪ "permission":"755", "replication":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0, "group":"mapred","length":0,
     ↪ "modificationTime":1350595874024, "owner":"mapred","pathSuffix":"mapred",
     ↪ "permission":"755", "replication":0,"type":"DIRECTORY"},
]}}
```

Atlas [23]

- A proposed[4] 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]

[4] It 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

```
# Different datastores, localstorage, mongodb and s3
SELECT * FROM dfs.root.‘/web/logs‘;
SELECT country, count(*) FROM mongodb.web.users GROUP BY country;
SELECT timestamp FROM s3.root.‘clicks.json‘  WHERE user_id = ’jdoe’;

# Query JSON: access the first students age from private data (a map)
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

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

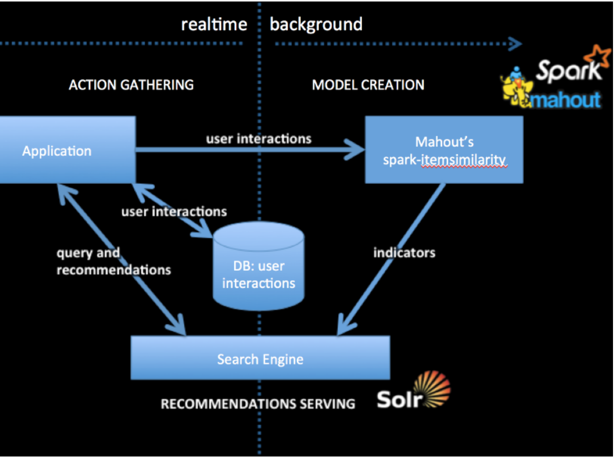
Response

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

Mahout [34]

- Framework for scalable machine learning
  - Collbarorative filtering
  - Classification
  - Clustering
  - Dimensionality reduction
  - Recommender
    - history: user purchases + all purchases ⇒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



Source: [36]

1. Collect user interactions n x (user-id, 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

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.html
27 https://github.com/cloudera/impala
29 http://incubator.apache.org/drill/
30 https://drill.apache.org/docs/json-data-model/
31 http://lucene.apache.org/solr/features.html
32 http://lucene.apache.org/solr/quickstart.html
33 http://solr-vs-elasticsearch.com/
34 http://mahout.apache.org/
35 http://www.scala-lang.org/
36 http://h2o.ai/
37 https://mahout.apache.org/users/recommender/quickstart.html
38 Free Ebook: https://www.mapr.com/practical-machine-learning
39 https://zeppelin.incubator.apache.org/