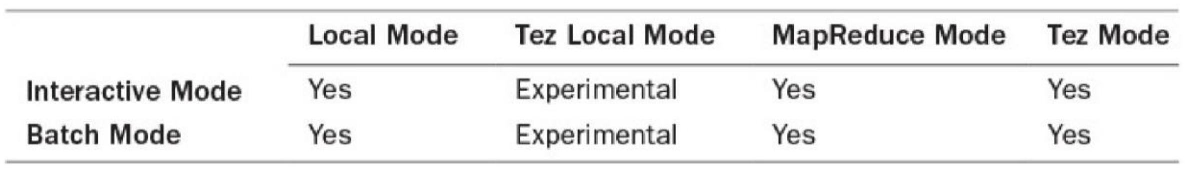
Module 2

Essential Hadoop Tools

1 Discuss the usage of Apache pig.

- **Apache Pig** is a high-level language that enables programmers to write complex Map Reduce transformations using a simple scripting language.
- Pig's simple SQL-like scripting language is called **Pig Latin**, and appeals to developers already familiar with scripting languages and SQL.
- Pig Latin (the actual language) defines a set of transformations on a data set such as aggregate, join, and sort.
- Pig is often used to extract, transform, and load (ETL) data pipelines, quick research on raw data.
- Apache Pig has several usage modes. The first is a **local mode** in which all processing is done on the local machine. The **non-local (cluster) modes** are Map Reduce and Tez.
- These modes execute the job on the cluster using either the Map Reduce engine or the optimized Tez engine.
- There are also interactive and batch modes available; they enable Pig applications to be developed locally in interactive modes, using small amounts of data, and then run at scale on the cluster in a production mode. The modes are in below fig.



	Local Mode	Tez Local Mode	MapReduce Mode	Tez Mode
Interactive Mode	Yes	Experimental	Yes	Yes
Batch Mode	Yes	Experimental	Yes	Yes

Table 7.1 **Apache Pig Usage Modes**

Pig Example Walk-Through:

- Working knowledge of Pig through the hand-on experience of creating pig scripts to carry out essential data operations and tasks.
- Apache Pig is also installed as part of the Horton works HDP Sandbox.
- In this simple example, Pig is used to extract user names from the **/etc/passwd file.**
- The following example assumes the user is hdfs , but any valid user with access to HDFScan run the example.
- To begin first, copy the passwd file to a working directory for local Pig operation: **$cp/etc/passwd.**
- Next, copy the data file into HDFS for Hadoop Map Reduceoperation: $
  - **hdfs dfs –put passwd passwd.**
- To confirm the file is in HDFS by entering the following command:
  - **hdfs dfs –ls passwd**
  - **-rw-r--r- 2 hdfs hdfs 2526 2015-03-17 11:08 passwd.**
- In local Pig operation, all processing is done on the local machine (Hadoop is not used). First, the interactive command line started: **$ pig -x local.**
- If Pig starts correctly, you will seea **grunt>** prompt.
- And also see a bunch of INFO messages. Next, enter the commands to load thepasswd file and then grab the user name and dump it to the terminal.
- Pig commands must end with a semicolon (;).
  - **grunt> A = load 'passwd' using Pig Storage(':') ;**
  - **grunt> B = foreach A generate $0 as id;**
  - **grunt> dumpB;**
- The processing will start and a list of user names will be printed to the screen.

- To exit the interactive session, enter the command quit.
  - **$ grunt> quit.**

2 Explain Apache Sqoop to Acquire Relational data with an example.

- Sqoop is a tool designed to transfer data between Hadoop and relational databases.
- Sqoop is used to

  -import data from a relational database management system (RDBMS) into the Hadoop Distributed File System(HDFS),

  - transform the data in Hadoop and

  - export the data back into an RDBMS.

Sqoop import method :

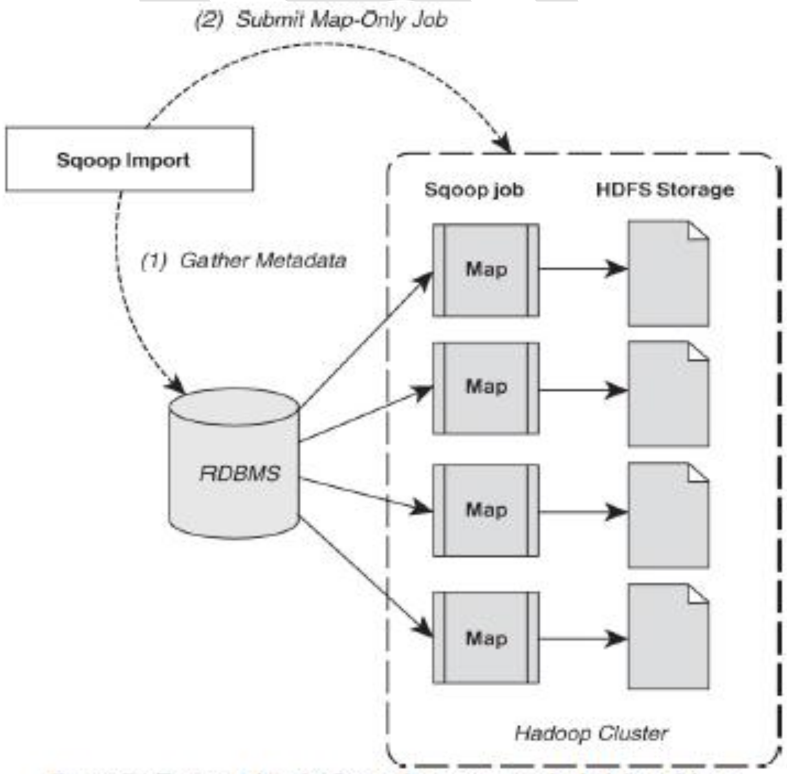


Figure 7.1 Two-step Apache Sqoop data import method (Adapted from Apache Sqoop Documentation)

The data import is done in two steps :

1) Sqoop examines the database to gather the necessary metadata for the data to be imported.
2) Map-only Hadoop job : Transfers the actual data using the metadata.

- The imported data are saved in an HDFS directory.
- Sqoop will use the database name for the directory, or the user can specify any alternative directory where the files should be populated. By default, these files contain comma delimited fields, with new lines separating different records.

Sqoop Export method :

Data export from the cluster works in a similar fashion. The export is done in two steps :

1) examine the database for metadata.
2) Map-only Hadoop job to write the data to the database.

Sqoop divides the input data set into splits, then uses individual map tasks to push the splits to the database.

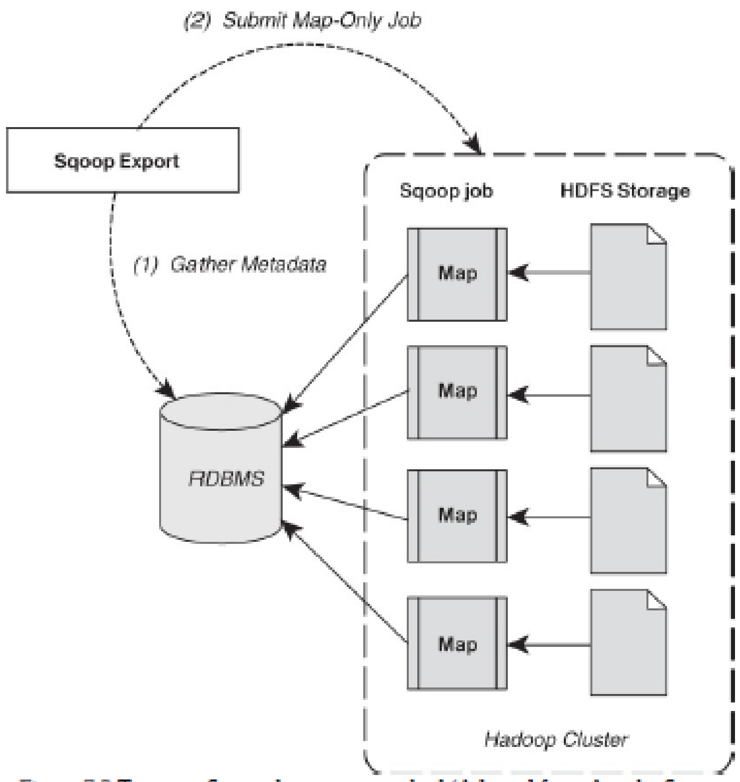


Figure 7.2 Two-step Sqoop data export method (Adapted from Apache Sqoop

Example: The following example shows the use of sqoop:

Steps:

1. Download Sqoop.

2. Download and load sample MySQL data.

3. Add Sqoop user permissions for the local machine and cluster.

4. Import data from MySQL to HDFS.

5. Export data from HDFS to MySQL.

Step 1: Download Sqoop and Load Sample MySQL Database

To install sqoop,

```
# yum install sqoop sqoop-metastore
```

To download database,

```
$ wget http : //downloads.mysql.com/docs/world_innodb.sql.gz
```

Step 2: Add Sqoop User Permissions for the Local Machine and Cluster.

In MySQL, add the following privileges for user sqoop to MySQL.

```
mysql> GRANT ALL PRIVILEGES ON world.* To 'sqoop'@'limulus'
IDENTIFIED BY 'sqoop';
mysql> GRANT ALL PRIVILEGES ON world.* To 'sqoop'@'10.0.0.%'
IDENTIFIED BY 'sqoop';
mysql> quit
```

Step 3: Import Data Using Sqoop

To import data, we need to make a directory in HDFS:

```
$ hdfs dfs -mkdir sqoop-mysql-import
```

The following command imports the Country table into HDFS. The option -table signifies the table to import, --target-dir is the directory created previously, and -m 1 tells Sqoop to use one map task to import the data.

```
$ sqoop import --connect jdbc:mysql://limulus/world --username sqoop
--password sqoop --table Country -m 1 --target-dir /user/hdfs/sqoopmysql-
import/country
```

The file can be viewed using the hdfs dfs -cat command:

Step 4: Export Data from HDFS to MySQL

Sqoop can also be used to export data from HDFS. The first step is to create tables for exported data.

Then use the following command to export the cities data into MySQL:

```
sqoop --options-file cities-export-options.txt --table CityExport --
```

```
staging-table CityExportStaging --clear-staging-table -m 4 --exportdir
/user/hdfs/sqoop-mysql-import/city
```

3.Discuss Apache Flume to acquire data streams

- ApacheFlume is an independent agent designed to collect, transport, and store data into HDFS.
- Data transport involves a number of Flume agents that may traverse a series of machines and locations.
- Flume is often used for log files, social media-generated data, email messages, and just about any continuous data source.

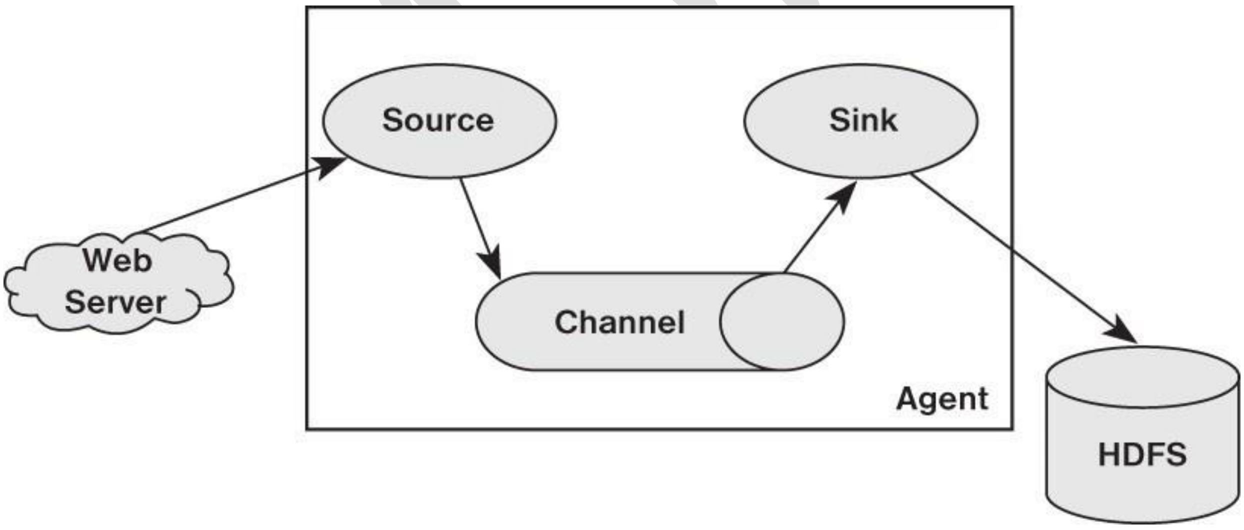


Figure 7.1Flume agent with source, channel, and sink

- Flame agent is composed of three components.
  - Source: The source component receives data and sends it to a channel. It can send the data tomore than one channel.

- Channel: A channel is a data queue that forwards the source data to the sink destination.
- Sink: The sink delivers data to destination such as HDFS, a local file, or another Flume agent.

- A Flume agent must have all three of these components defined. Flume agent can have several source, channels, and sinks.
- Source can write to multiple channels, but a sink can take data from only a single channel.
- Data written to a channel remain in the channel until a sink removes the data.
- By default, the data in a channel are kept in memory but may be optionally stored on disk to prevent data loss in the event of a network failure.

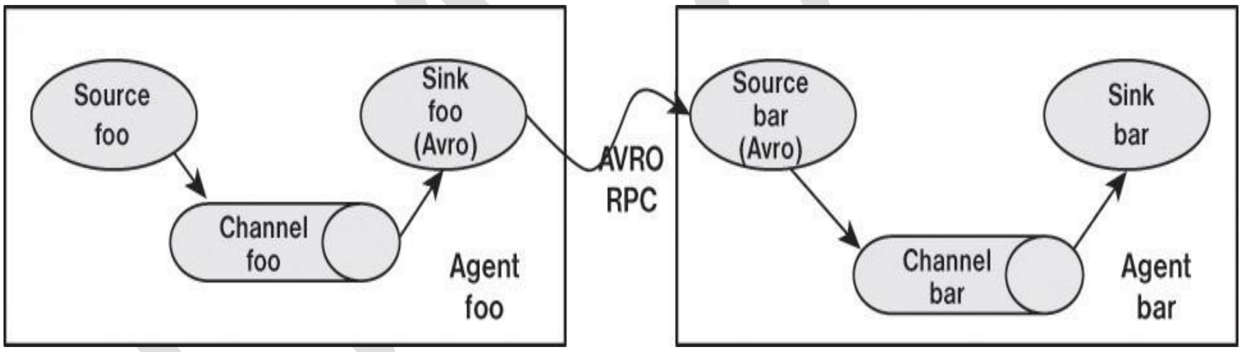


Figure 7.2 Pipeline created by connecting Flume agents

- As shown in the above figure, Sqoop agents may be placed in a pipeline, possibly to traverse several machines or domains.
- In this Flume pipeline, the sink from one agent is connected to the source of another.
- The data transfer normally used by Flume, which is called Apache Avro.

- Avro is a data serialization/deserialization system that uses a compact binary format.
- The scheme is sent as part of the data exchange and is defined using JSON.
- Avro also uses remote procedure calls (RPCs) to send data.

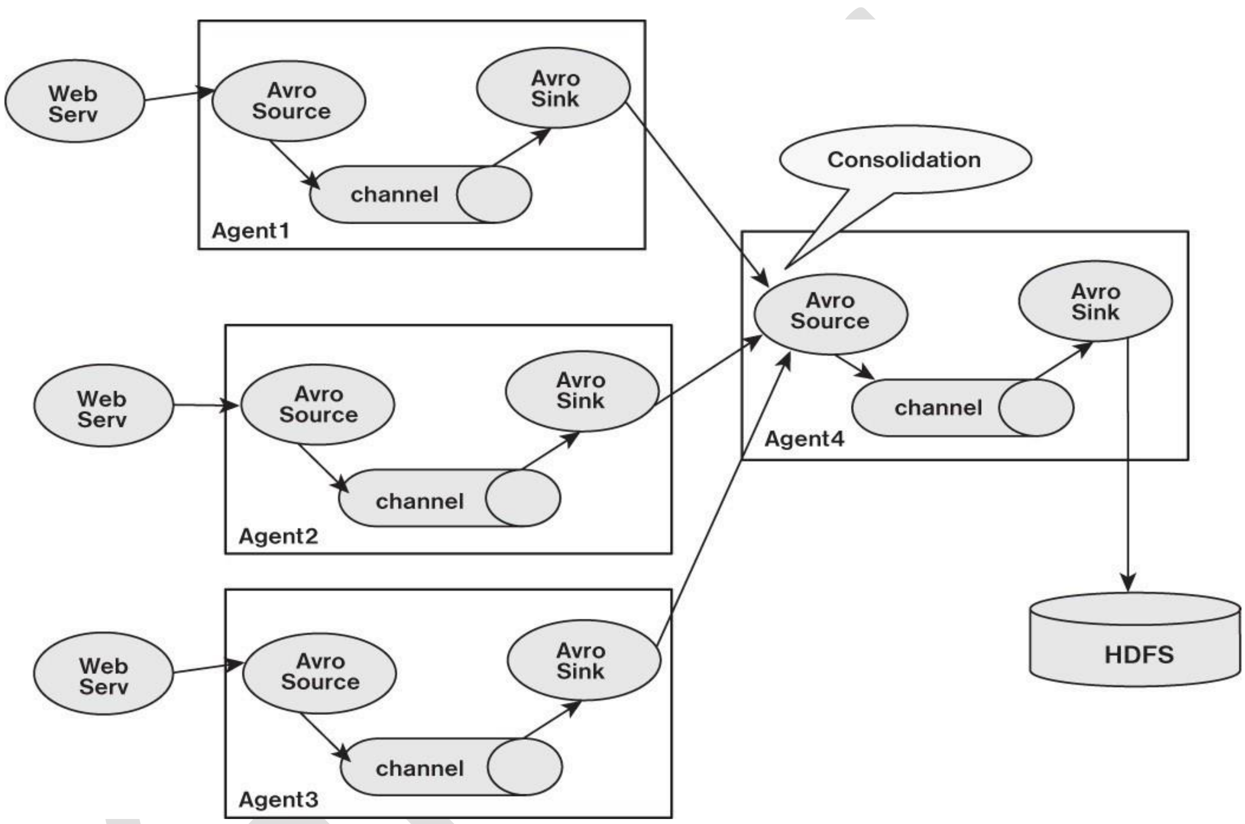


Figure 7.3 A Flume consolidation network.

4 Demonstrate the working of Hive with Hadoop

- Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, ad hoc queries, and the analysis of large data sets using a SQL-like language called HiveQL.
- Hive is considered the de facto standard for interactive SQL queries over petabytes of data using Hadoop.

- Some essential features:
- Tools to enable easy data extraction, transformation, and loading (ETL)
- A mechanism to impose structure on a variety of data formats
- Access to files stored either directly in HDFS or in other data storage systems such as HBase
- Query execution via MapReduce and Tez (optimized MapReduce)
- Hive is also installed as part of the Hortonworks HDP Sandbox.
- To work in Hive with Hadoop, user with access to HDFS can run the Hive queries.
- Simply enter the hive command. If Hive start correctly,it get a hive> prompt.

```
$ hive
(some messages may show up here)
hive>
```

- Hive command to create and drop the table. That Hive commands must end with a semicolon (;).

```
hive> CREATE TABLE pokes (foo INT, bar STRING);
OK
Time taken: 1.705 seconds
```

- To see the table is created,

```
hive> SHOW TABLES;
OK
pokes
```

Time taken: 0.174 seconds, Fetched: 1 row(s)

- To drop the table,

  hive> DROP TABLE pokes;

  OK

  Time taken: 4.038 seconds

- The first step is to Creation of table can be developed using a web server log file:

  hive> CREATE TABLE logs(t1 string, t2 string, t3 string, t4 string, t5 string, t6 string, t7 string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ';

- Next, to load the data from the sample.log file, the file is found in the local directory and not in HDFS.

  hive> LOAD DATA LOCAL INPATH 'sample.log' OVERWRITE INTO TABLE logs;

- Finally, the select step that this invokes a Hadoop MapReduce operation. The results appear at the end of the output.
  - hive> SELECT t4 AS sev, COUNT(*) AS cnt FROM logs WHERE t4 LIKE '[%'
  - GROUP BY t4;
  - Total jobs = 1
  - 2015-03-27 13:00:17,399 Stage-1 map = 0%, reduce = 0%
  - 2015-03-27 13:00:26,100 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.14 sec
  - 2015-03-27 13:00:34,979 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.07 sec

- Total MapReduce CPU Time Spent: 4 seconds 70 msec
- OK
- [DEBUG] 434
- [ERROR] 3
- [FATAL] 1
- [INFO] 96
- [TRACE] 816
- [WARN] 4
- Time taken: 32.624 seconds, Fetched: 6 row(s)

- To exit Hive, simply type exit;
  - hive> exit;

5. Explain yarn application framework with a neat diagram?

YARN presents a resource management platform, which provides services such as scheduling, fault monitoring, data locality, and more to MapReduce and other frameworks. Below figure illustrates some of the various frameworks that will run under YARN.

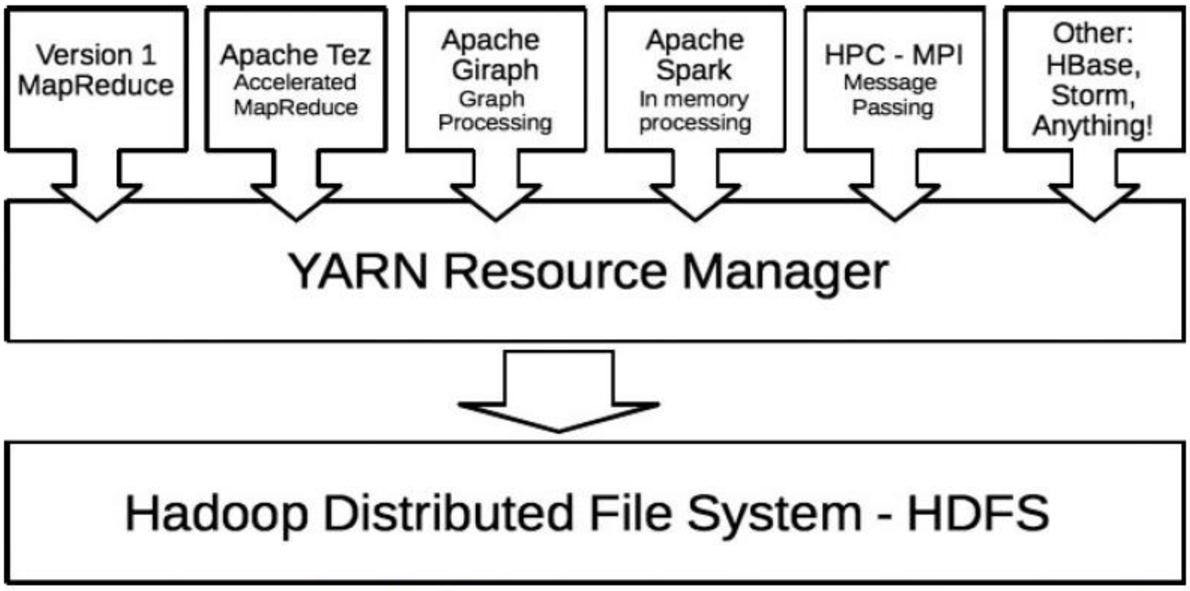


Figure 8.2 Example of the Hadoop version 2 ecosystem. Hadoop version 1 supports batch MapReduce applications only.

Distributed-Shell

- Distributed-Shell is an example application included with the Hadoop core components that demonstrates how to write applications on top of YARN.
- It provides a simple method for running shell commands and scripts in containers in parallel on a Hadoop YARN cluster.

Hadoop MapReduce

- MapReduce was the first YARN framework and drove many of YARN's requirements. It is integrated tightly with the rest of the Hadoop ecosystem projects, such as Apache Pig, Apache Hive, and Apache Oozie.

Apache Tez:

- Many Hadoopjobs involve the execution of a complex directed acyclic graph (DAG) of tasks using separate MapReduce stages. Apache Tez generalizes this process and enables these tasks to be spread across stages so that they can be run as a single,all-encompassing job.

- Tez can be used as a MapReduce replacement for projects such as Apache Hive and Apache Pig. No changes are needed to the Hive or Pig applications.

Apache Giraph

- Apache Giraph is an iterative graph processing system built for high scalability.
- Facebook, Twitter, and LinkedIn use it to create social graphs of users.
- Giraph was originally written to run on standard Hadoop V1 using the MapReduce framework, but that approach proved inefficient and totally unnatural for various reasons.
- The native Giraph implementation under YARN provides the user with an iterative processing model that is not directly available with MapReduce.
- In addition, using the flexibility of YARN, the Giraph developers plan on implementing their own web interface to monitor job progress.

Hoya: HBase on YARN

- The Hoya project creates dynamic and elastic Apache HBase clusters on top of YARN.
- A client application creates the persistent configuration files, sets up the HBase cluster XML files, and then asks YARN to create an ApplicationMaster.

- YARN copies all files listed in the client's application-launch request from HDFS into the local file system of the chosen server, and then executes the command to start the Hoya ApplicationMaster.
- Hoya also asks YARN for the number of containers matching the number of HBase region servers it needs.

Dryad on YARN

- Similar to Apache Tez, Microsoft's Dryad provides a DAG as the abstraction of execution flow. This framework is ported to run natively on YARN and is fully compatible with its non-YARN version.
- The code is written completely in native C++ and C# for worker nodes and uses a thin layer of Java within the application.

Apache Spark

- Spark was initially developed for applications in which keeping data in memory improves performance, such as iterative algorithms, which are common in machine learning, and interactive data mining.
- Spark differs from classic MapReduce in two important ways.
- First, Spark holds intermediate results in memory, rather than writing them to disk.
- Second, Spark supports more than just MapReduce functions; that is, it greatly expands the set of possible analyses that can be executed over HDFS data stores.
- It also provides APIs in Scala, Java, and Python.

Apache Storm

- This framework is designed to process unbounded streams of data in real time. It can be used in any programming language.
- The basic Storm use-cases include real-time analytics,online machine learning, continuous computation, distributed RPC (remote procedure calls), ETL (extract, transform, and load), and more.
- Storm provides fast performance, is scalable, is fault tolerant, and provides processing guarantees.
- It works directly under YARN and takes advantage of the common data and resource management substrate.

6. Explain Apache oozie workflow for Hadoop Architecture with a neat diagram.

- Oozie is a workflow director system designed to run and manage multiple related Apache Hadoop jobs.
- Oozie is designed to construct and manage these workflows.
- Oozie is not a substitute for the YARN scheduler.That is, YARN manages resources for individual Hadoop jobs, and Oozie provides a way to connect and control Hadoop jobs on the cluster.
- Oozie workflow jobs are represented as directed acyclic graphs (DAGs) of actions. (DAGs are basically graphs that cannot have directed loops.) Three types of Oozie jobs are permitted:
  1. Workflow—a specified sequence of Hadoop jobs with outcome-based decision points and control dependency. Progress from

one action to another cannot happen until the first action is complete.

2. Coordinator—a scheduled workflow job that can run at various time intervals or when data become available.
3. Bundle—a higher-level Oozie abstraction that will batch a set of coordinator jobs.

Oozie also provides a CLI and a web UI for monitoring jobs.

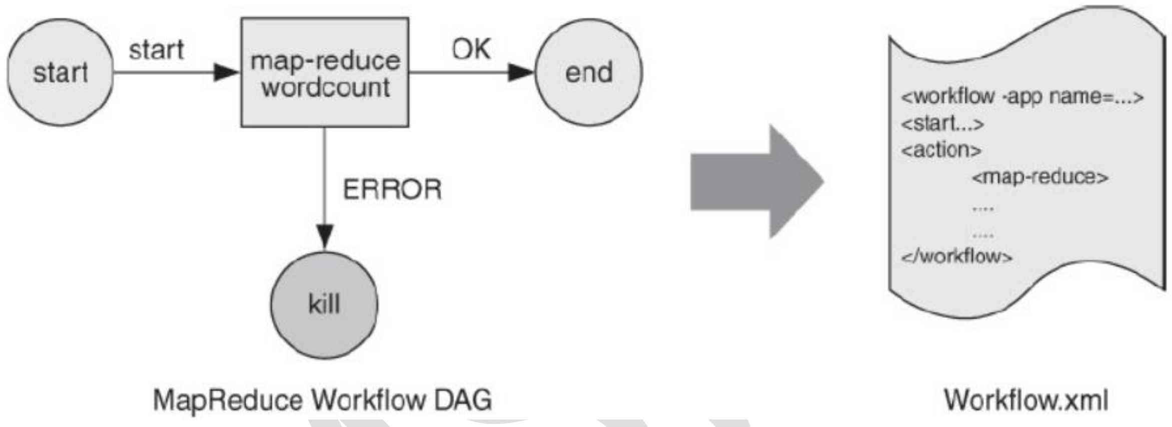


Figure : simple oozie DAG workflow

Oozie workflow definitions are written in hPDL. Such workflows contain several types of nodes:

a) **Control flow nodes** define the beginning and the end of a workflow. They include start, end, and optional fail nodes.
b) **Action nodes** are where the actual processing tasks are defined. When an action node finishes, the remote systems notify Oozie and the next node in the workflow is executed. Action nodes can also include HDFS commands.
c) **Fork/join nodes** enable parallel execution of tasks in the workflow. The fork node enables two or more tasks to run at the

same time. A join node represents a rendezvous point that must wait until all forked tasks complete.

d) **Control flow nodes** enable decisions to be made about the previous task. Control decisions are based on the results of the previous action (e.g., file size or file existence). Decision nodes are essentially switch-case statements that use JSP EL (Java Server Pages—Expression Language) that evaluate to either true or false.

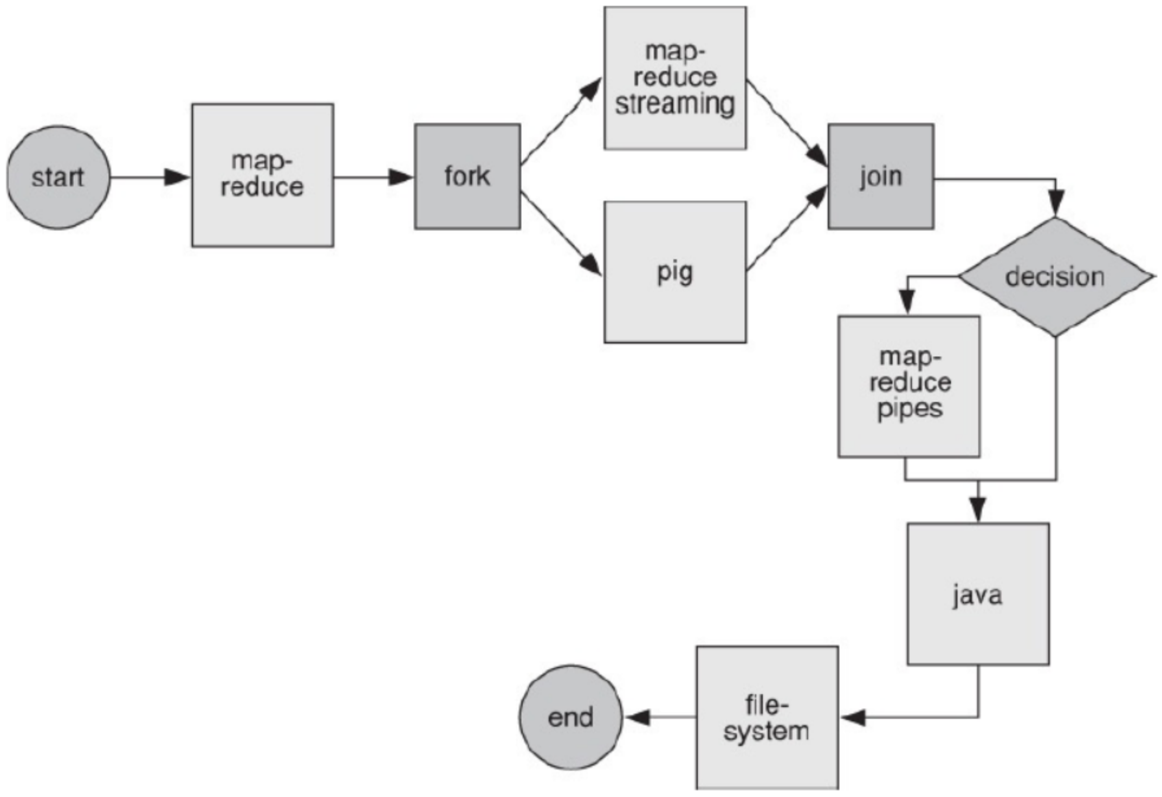


Figure: A more complex Oozie DAG workflow