BDA – MOD 1 and MOD 2

Slides by author

Module 2

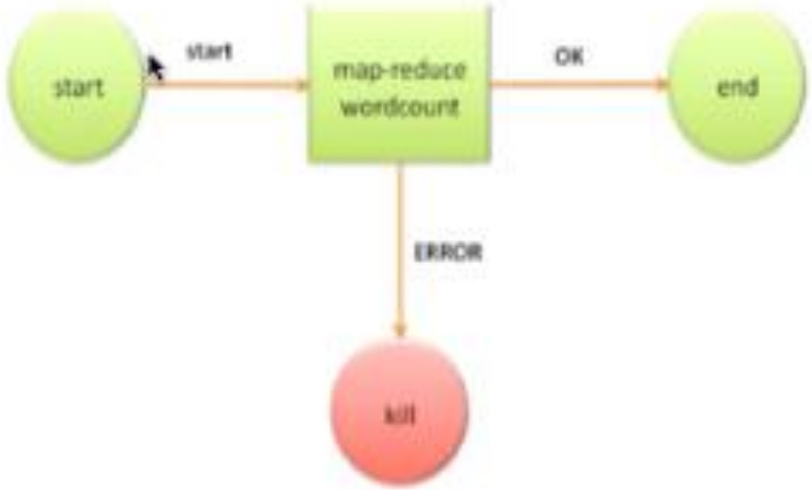
Apache Oozie

Oozie is a workflow scheduler system to run and manage Apache Hadoop jobs.

- Oozie **Workflow** jobs are represented as Directed Acyclical Graphs (DAGs) of actions. Jobs are run on submission. (DAG's cannot have loops.)
- Oozie **Coordinator** jobs are repetitive tasks and are often triggered by time (frequency) and data availability.
- Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, and Sqoop) as well as system specific jobs (such as Java programs and shell scripts).
- Oozie is a scalable, reliable, and an extensible system. Provides a CLI and a Web UI for monitoring.

Apache Oozie Workflow

An Oozie workflow is a collection of actions (i.e. Hadoop Map/Reduce jobs, Pig jobs) arranged in a control dependency DAG (Direct Acyclic Graph). The "control dependency" from one action to another means that the second action can't run until the first action has completed.



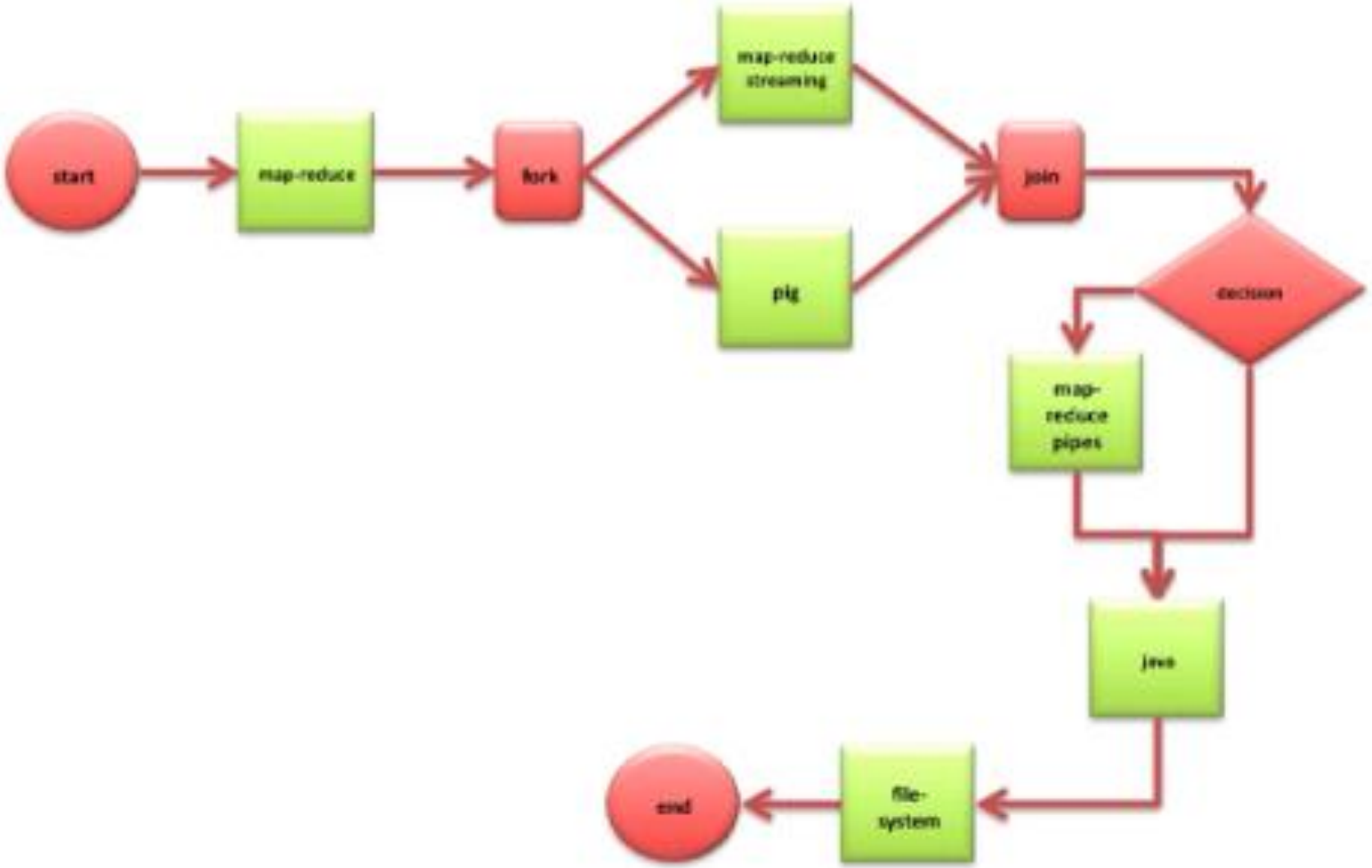
MapReduce Workflow DAG

Workflow.xml

Apache Oozie Features

- Oozie workflows definitions are written in hPDL (an XML Process Definition Language).
- Oozie workflow actions start jobs on a Hadoop cluster. Upon action completion, the remote systems callback Oozie to notify the action completion, at this point Oozie proceeds to the next action in the workflow.
- Oozie workflows contain **control flow** nodes and **action nodes**.
- Control flow nodes define the beginning and the end of a workflow (start, end, and fail nodes).
- Oozie provides support for different types of actions nodes (Hadoop MapReduce, HDFS, Pig, Hive, SSH, HTTP, eMail, etc.). Oozie can be extended to support additional type of actions.

Example Apache Oozie Workflow



```
Step 1: Set hadoop.proxyuser.oozie.groups in core-site.xml
=========================================================
# This step depends on the group that will be running
# Oozie. The default is "users." We are going to run
# Oozie as group "hadoop" so there is a change needed
# in the core-site.xml file.

# If using Ambari, go to Services/HDFS/Config tab and at the bottom
# click on  Custom core-site.xml. Change "hadoop.proxyuser.oozie.groups"
# from "users" to "hadoop."

# Note: You may wish to change this for faclon, hcat, and hive as well.
# If you plan on keeping placing users in group "users" this is not needed.

# If not using Ambari, set this directly in core-site.xml and restart Hadoop.

<property>
      <name>hadoop.proxyuser.oozie.groups</name>
      <value>hadoop</value>
</property>


Step 2: Download Oozie Examples
```

```
Step 2: Download Oozie Examples
================================
# The examples are part of the oozie-client-4.0.0.2.1.2.1-471.el6.noarch.rpm
# package. Make sure this is installed, then as user hdfs,

  tar xvzf /usr/share/doc/oozie-4.0.0.2.1.2.1/oozie-examples.tar.gz

# the examples must also be placed in HDFS

  hdfs dfs -put examples/ examples

# The the Oozie Share Library must be installed in HDFS.
# If you are using the Ambari install of HDP 2.1 from lesson 8.6, this is
# already in HDFS under: /user/oozie/share/lib. It is part of the
# oozie-4.0.0.2.1.2.1-471.el6.noarch.rpm and is installed on the host in
# /usr/lib/oozie/oozie-sharelib.tar.gz. If you install by hand, then
# make /user/oozie in HDFS and put the extracted oozie-sharelib files
# in this directory as user oozie and group hadoop.

# The example applications are under the examples/app directory, one directory per example.
# The directory contains the application XML file (workflow, or worklfow and coordinator),
# the job.properties file to submit the job and any JAR files the example may need.

# The inputs for all examples are in the examples/input-data/ directory.
```

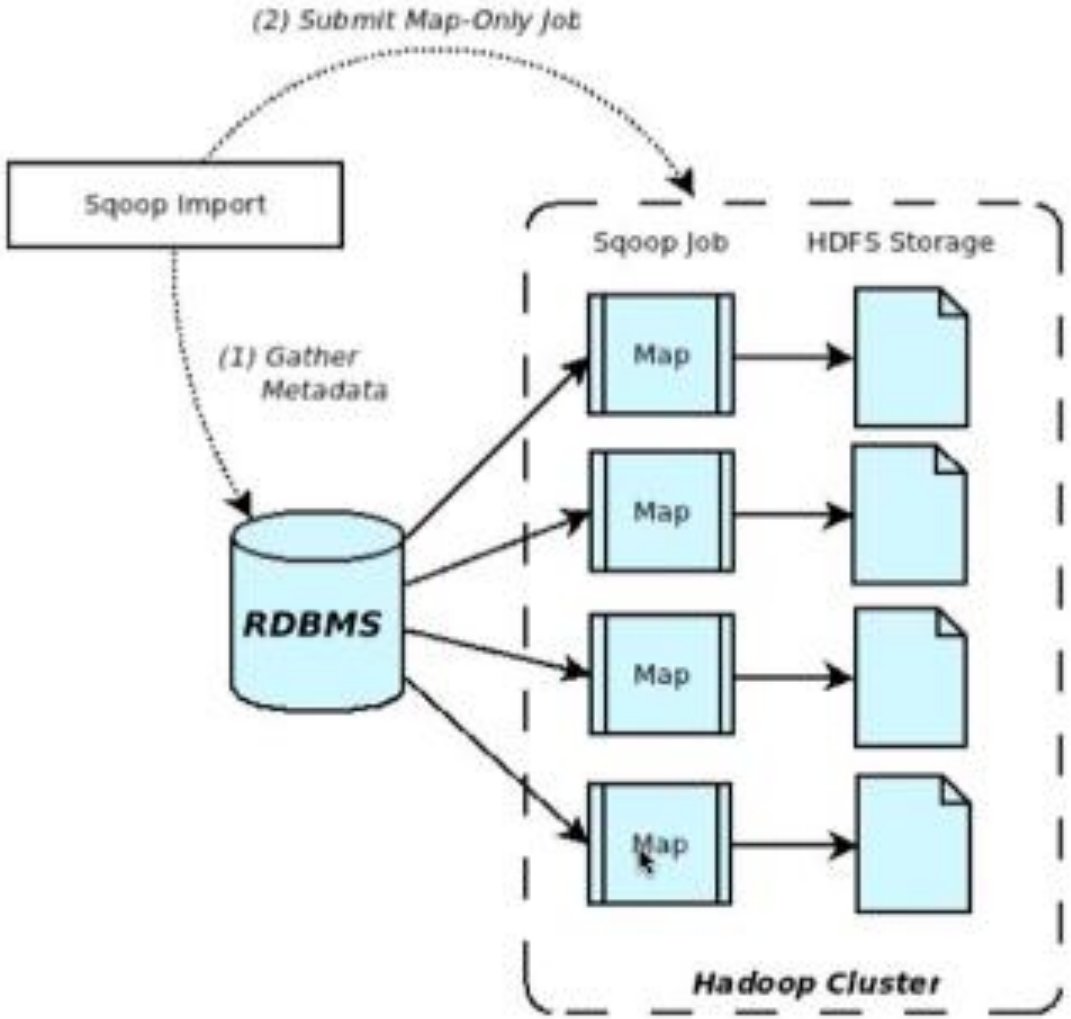
doug — root@limulus:~/Hadoop-Fundamentals-V2/Lesson-6/Lesson-6.5 — ssh — 96×26

```
Step 3: Run The Simple MapReduce Example
========================================
# job.properties - defined values (path names, ports, etc) for a jobs
# workflow.xml - workflow for the job. In this case, simple MR (pass/fail)

# view the workflow.xml

  vi examples/apps/map-reduce/workflow.xml

# update local job.properties to reflect current system.
# Important: This must be done for all the examples.

   vi examples/apps/map-reduce/job.properties

# change

   nameNode=hdfs://localhost:8020
   jobTracker=localhost:8032

# to

   nameNode=hdfs://limulus:8020
   jobTracker=limulus:8050

# Note the change in port number for the jobTracker line.
```

Apache Sqoop Version 2

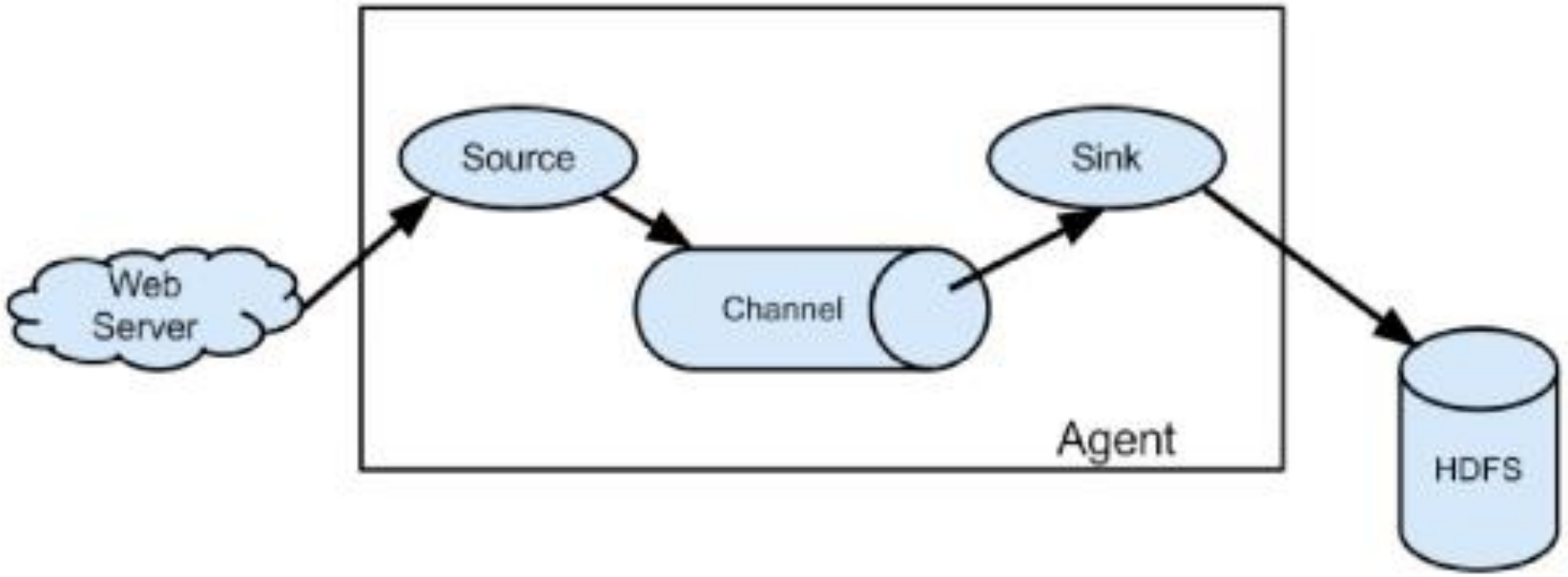
- Sqoop is a tool designed to transfer data between Hadoop and relational databases. You can use Sqoop to import data from a relational database management system (RDBMS) into the Hadoop Distributed File System (HDFS), transform the data in Hadoop, and then export the data back into an RDBMS.
- You can use Sqoop with any JDBC-compliant database. Sqoop has been tested on the following databases: Microsoft SQL Server, PostgreSQL, MySQL and Oracle.
- Version 2 (in beta) does not support "connectors" or data transfer from RDBMS directly to Hive or Hbase or data transfer from Hive or HBase to your RDBMS. (There are pathways to accomplish these tasks.)

Apache Sqoop Import

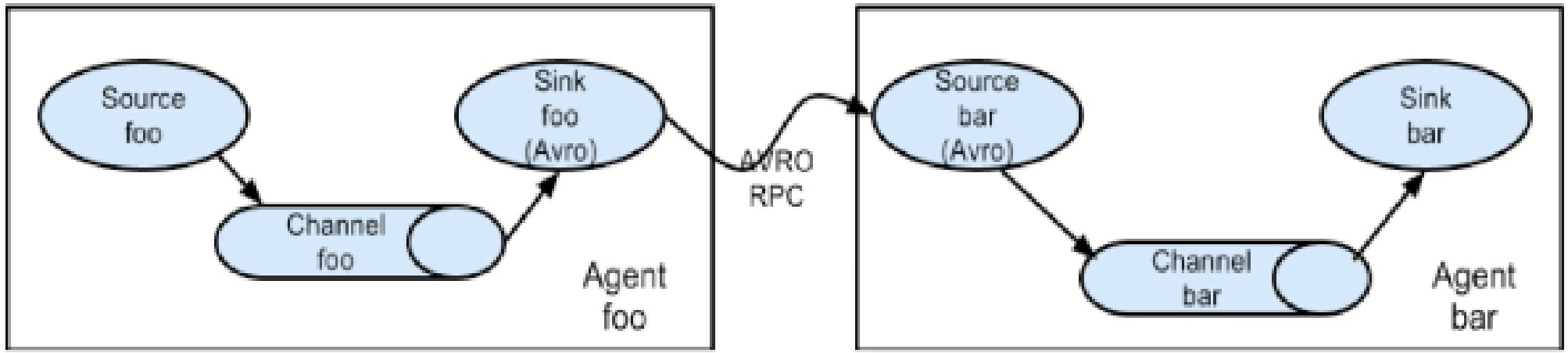


Apache Flume

- Apache Flume is a distributed, reliable, and available system for collecting, aggregating and moving large amounts of event data (e.g log files) from many different sources to a centralized data store (e.g HDFS).
- Can be used for log files, social-media-generated data, email messages and pretty much any continuous data source.

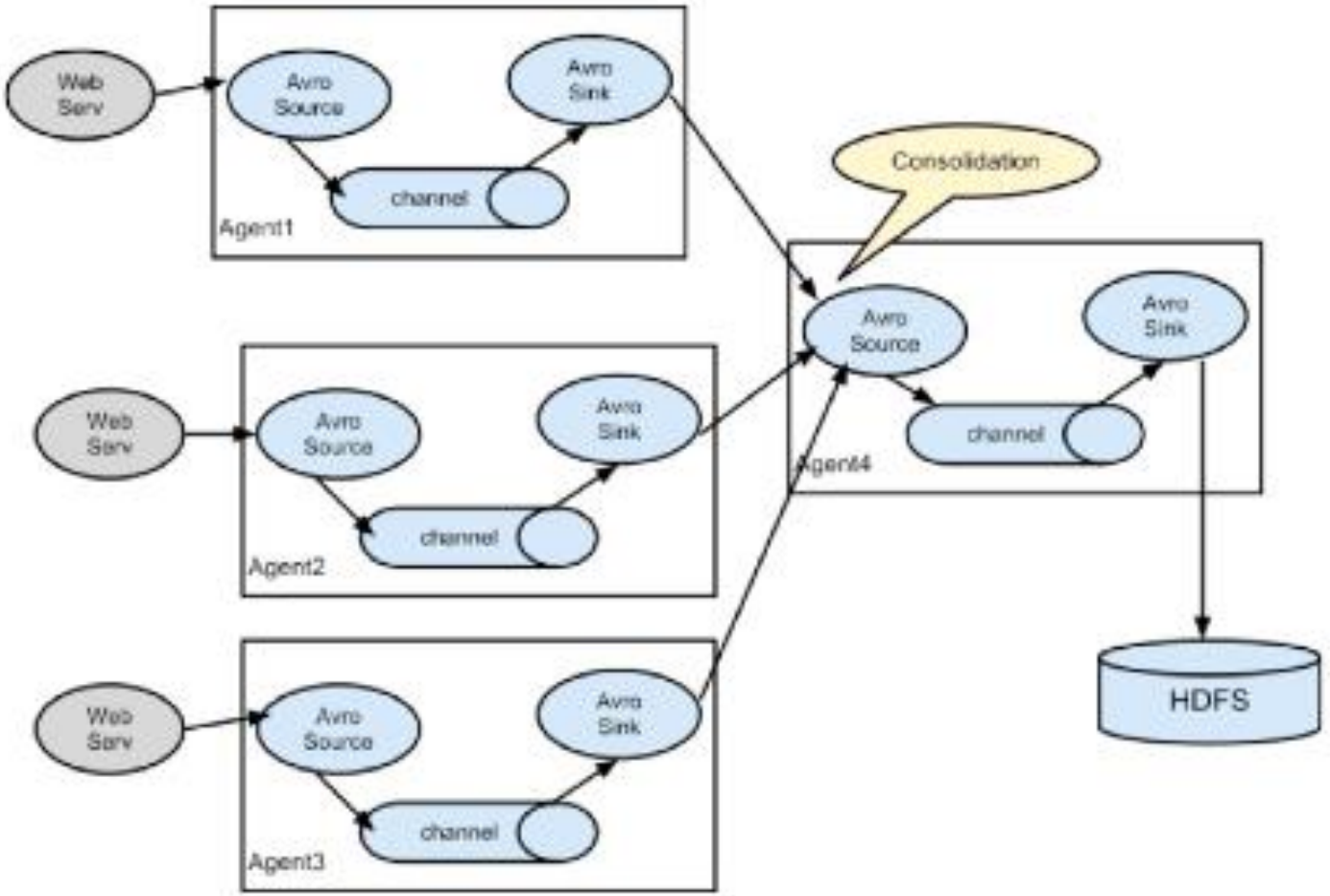


Apache Flume Deployment Examples



Pipeline Multiple Agents

Apache Flume Deployment Examples

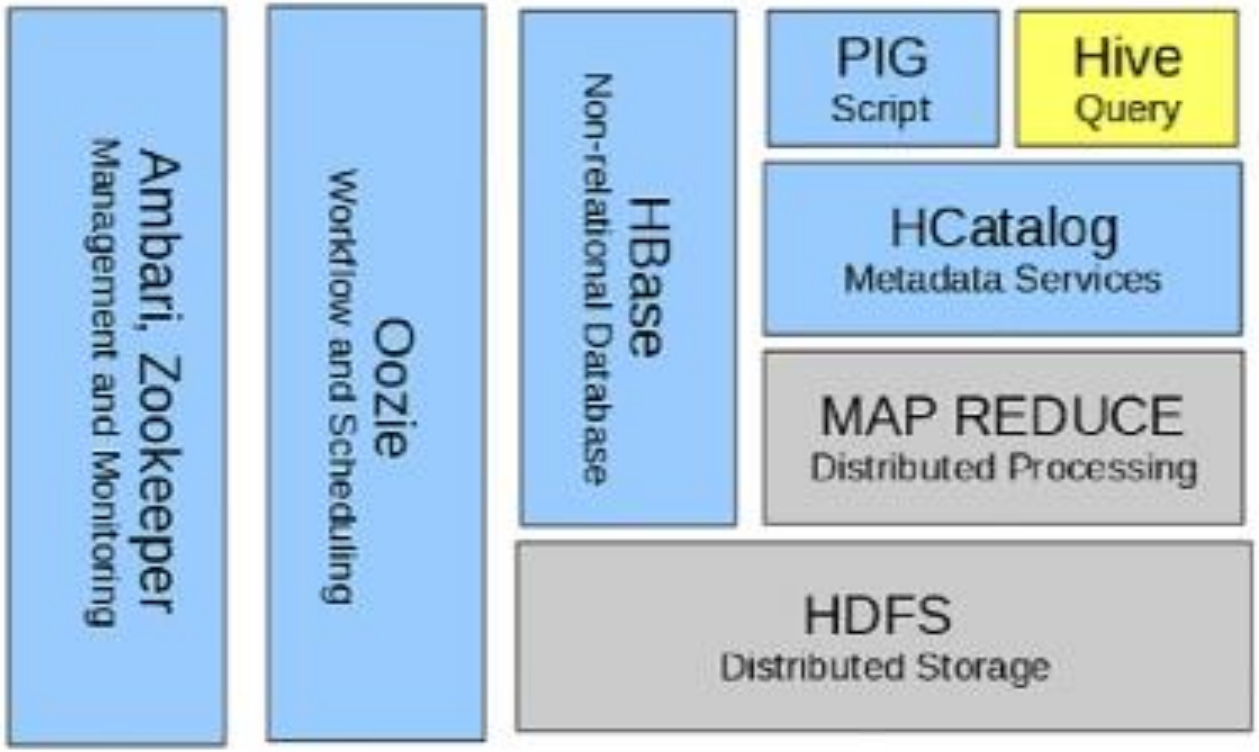


Consolidate Multiple Sources

Apache Hive

Apache Hive is a data warehouse infrastructure built on top of Hadoop. Hive provides data summarization, ad-hoc queries, and the analysis of large datasets using data stored in the Hadoop file systems. It provides a mechanism to query the data using a SQL-like language called HiveQL.

Apache Hive And Hadoop



Apache Pig

Apache Pig is a high-level language for running MapReduce applications with Hadoop. Users write application scripts using a higher level "database" like notation.

MapReduce Compatibility

- The primary goal with Hadoop version 2 was MapReduce compatibility with version 1.
- MapReduce functionality is now a YARN framework (not part of core).

MapReduce Compatibility

- The primary goal with Hadoop version 2 was MapReduce compatibility with version 1.
- MapReduce functionality is now a YARN framework (not part of core).
- The MapReduce Application Master is required to locate data blocks for processing and then request containers on those blocks.
- The Application Master handles failed tasks and requests new containers to re-run them.
- Map and Reduce tasks are now created as needed by the Application Master providing a much better cluster utilization.
- A parallel shuffle service was added as an auxiliary service.

Binary Compatibility of `org.apache.hadoop.mapred` APIs

For the vast majority of users who use the older `org.apache.hadoop.mapred` APIs, MapReduce on YARN ensures full binary compatibility. These existing applications can run on YARN directly without recompilation. You can use jars of your existing application that codes against `mapred` APIs and use `bin/hadoop` to submit them directly to YARN.

Source Compatibility of `org.apache.hadoop.mapreduce` APIs

It was difficult to ensure full binary compatibility with the newer `org.apache.hadoop.mapreduce` APIs. Existing applications using `mapreduce` APIs are source compatible and can run on YARN with no changes, with simple recompilation against MapReduce Version 2 jars that are shipped with Hadoop Version 2.

Compatibility of Command-line Scripts

- Most of the command line scripts from Hadoop 1.x should just work
- The `mradmin` functionality is now replaced with `rmadmin` because there is no version 1 JobTracker and TaskTracker
- If `mradmin` commands are executed they will produce warning messages and will not work
- There is no binary or source compatibility for `mradmin` in Hadoop version 2 with YARN

Module 1

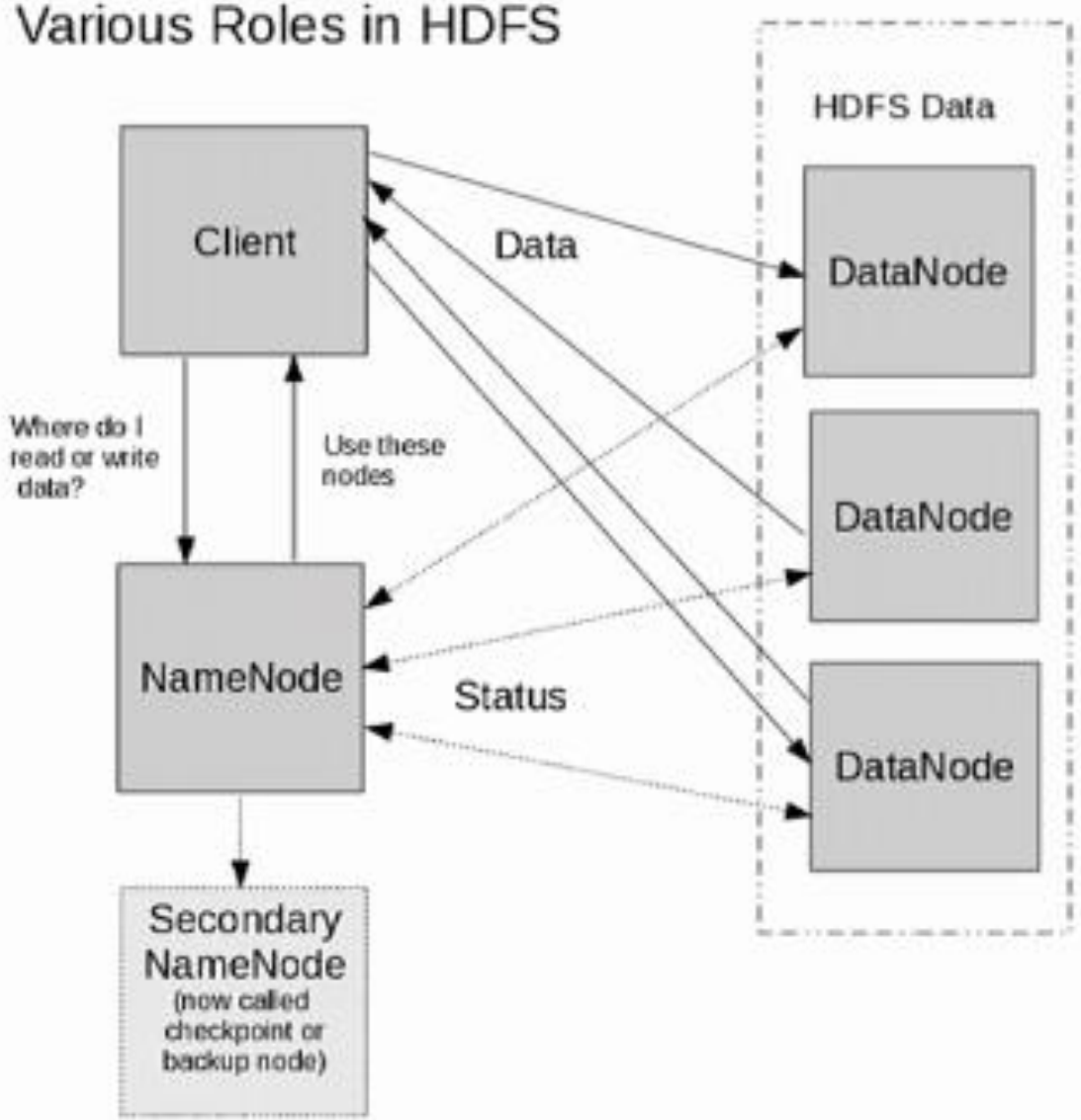
HDFS Trade-offs

- Optimized for streaming reads makes it poor for random seeks
- Write once file system
- No local cache needed
- With hardware failure comes reduced performance
- A specialized filesystem, not designed for general use

NameNode and DataNodes

- Master/slave model
- NameNode–meta data server or "data traffic cop"
- Single Namespace–managed by the NameNode
- DataNodes–where the data live
- Secondary NameNode–checkpoints NameNode, but not failover

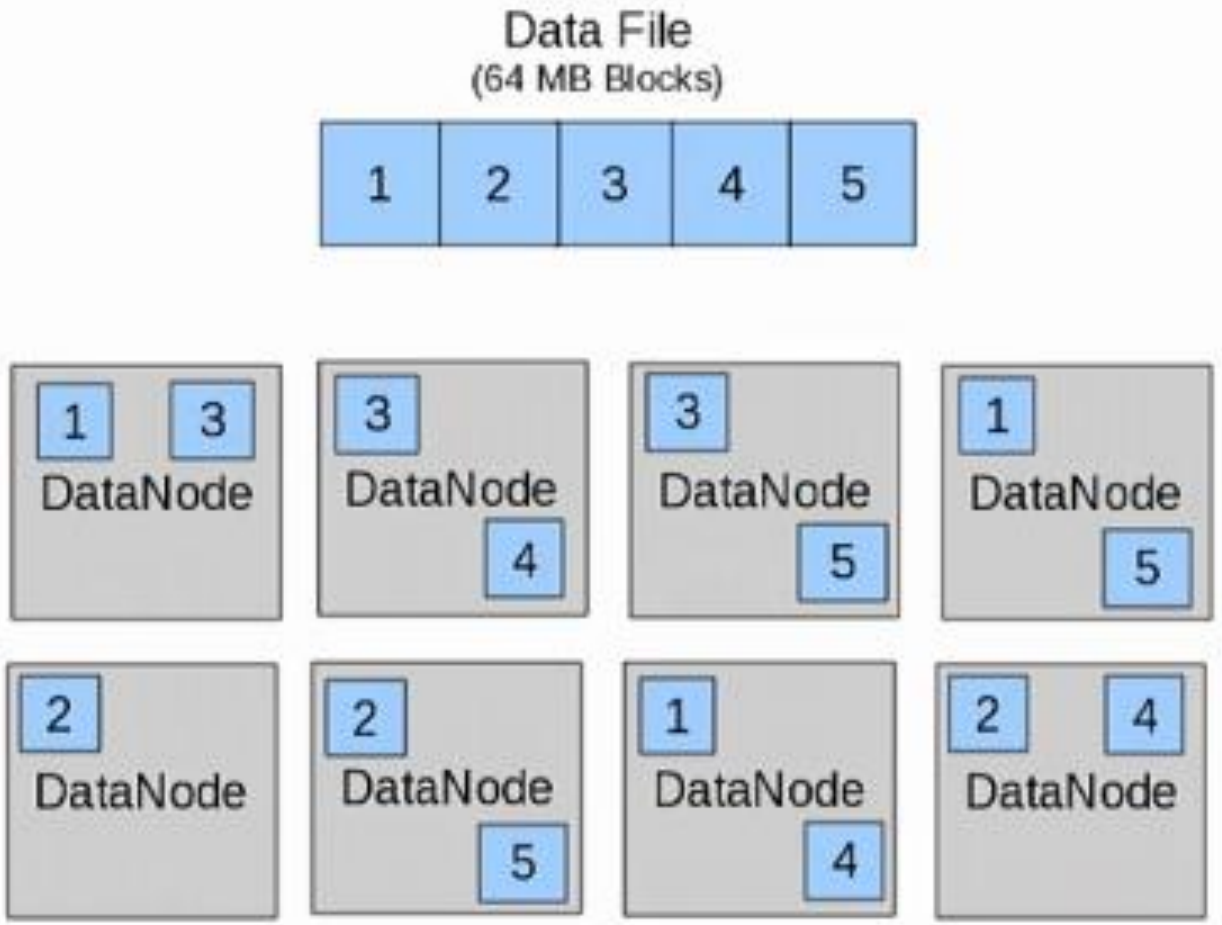
Various Roles in HDFS



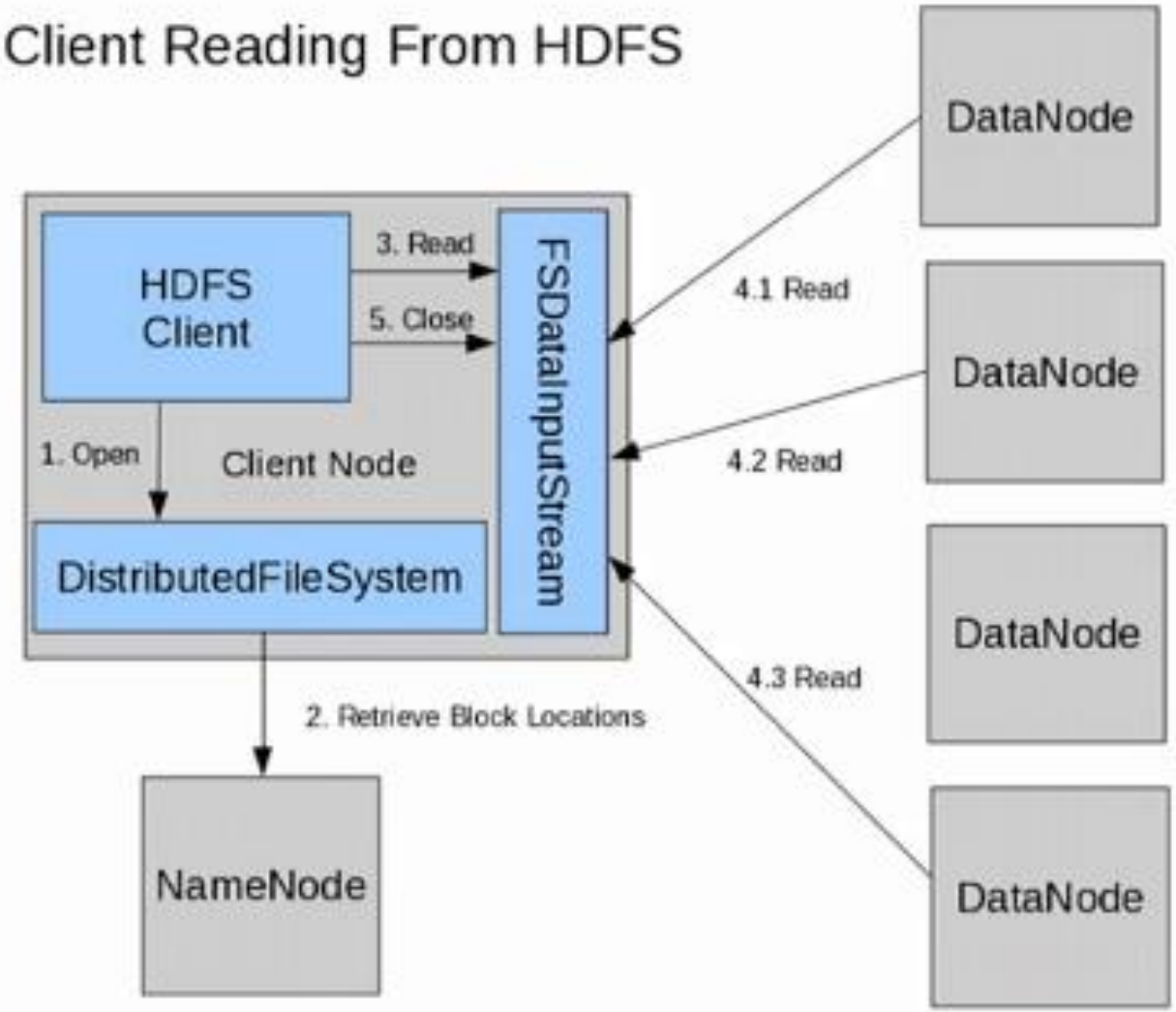
The File System Namespace

- Traditional hierarchical filesystem with files and directories
- Users can create, remove, move, rename, copy (ls, du, mv, cp, rm)
- Must use HDFS through Hadoop, not directly on DataNodes

Block Replication in HDFS



Client Reading From HDFS



New `hdfs` Command

The `hadoop dfs <args>` option has been depreciated (i.e., it will work for a while). The new command for using and administering HDFS is `hdfs`. For example instead of :

```
$ hadoop dfs -ls /
$ hadoop dfsadmin -report
```

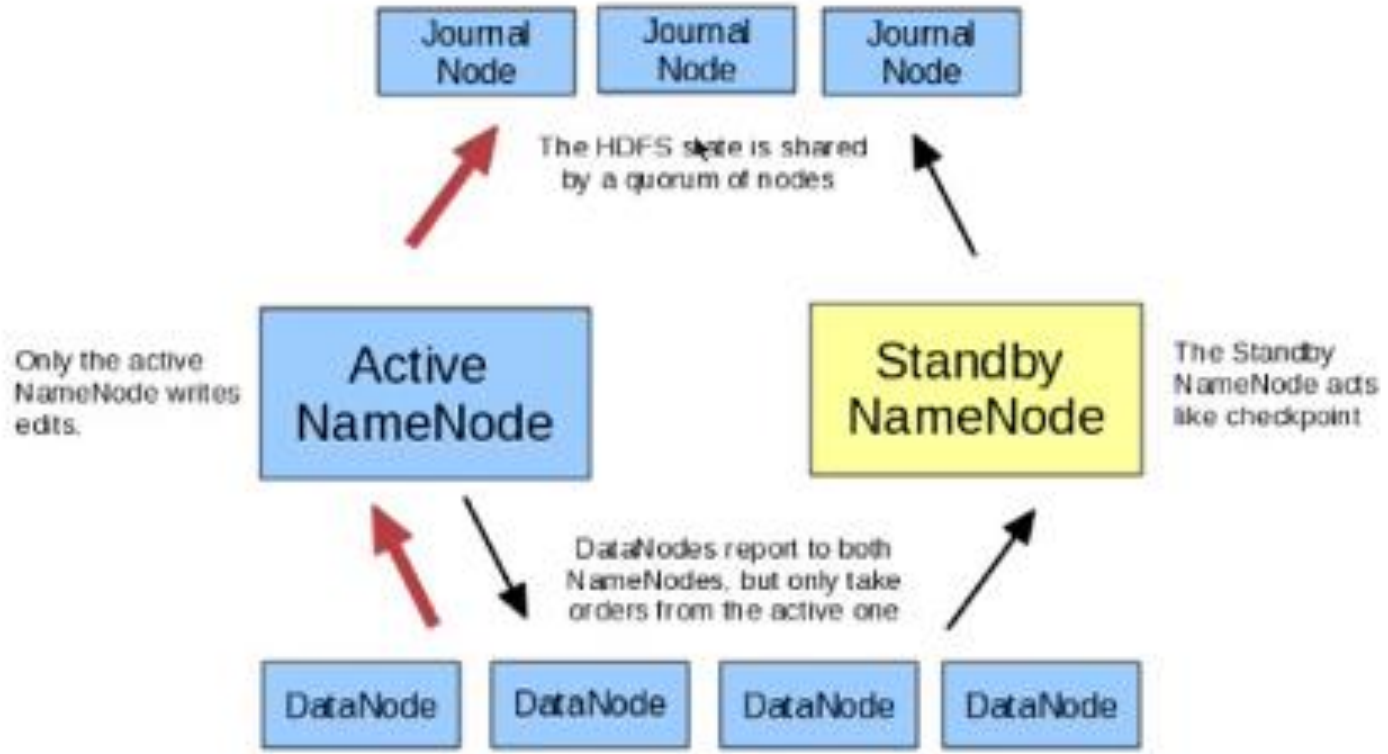
Use the following:

```
$ hdfs dfs -ls /
$ hdfs dfsadmin -report
```

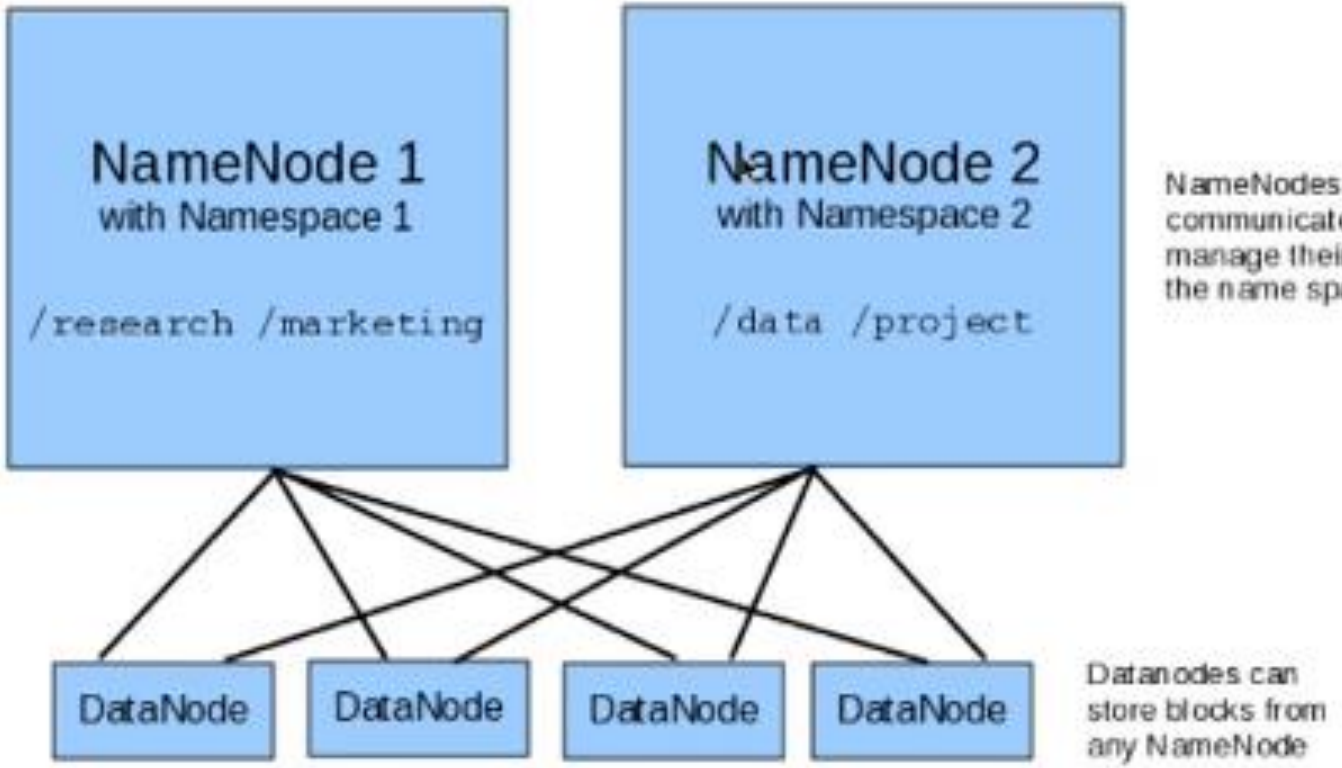
HDFS High Availability

- The **NameNode** stores the metadata of the HDFS in a file called fsimage at runtime. Modifications are written to a log file called edits. NameNode merges fsimage and edits on startup.
- The **Checkpoint Node** or **Secondary NameNode** periodically fetch fsimage and edits from the NameNode and merge them and return the update to the NameNode.
- The **Backup Node** provides the same functionality as the Checkpoint Node but is synchronized using a real time stream from the NameNode.

HDFS High Availability



HDFS Federation



NameNodes do not communicate, they only manage their slice of the name space

Datanodes can store blocks from any NameNode

HDFS Snapshots

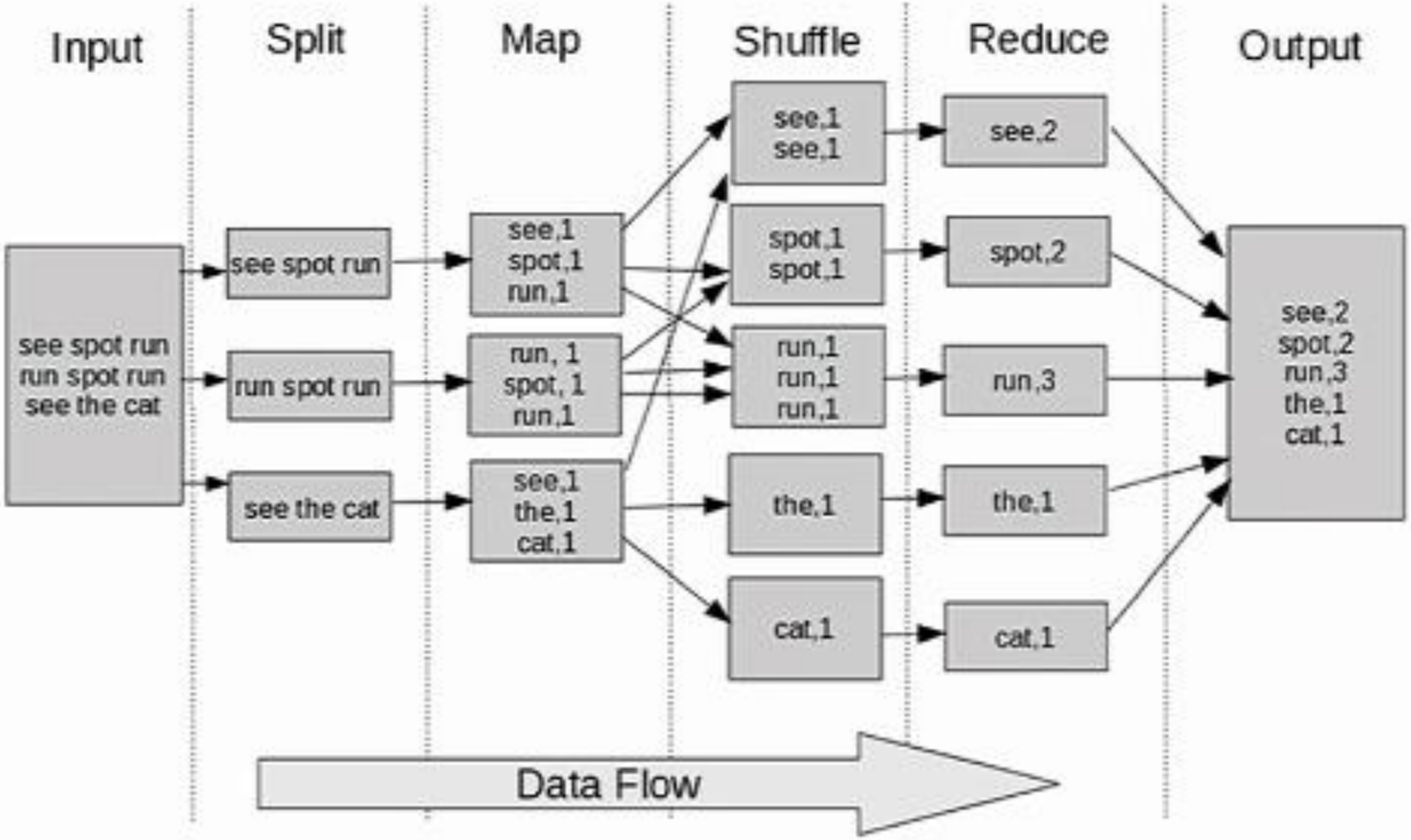
- HDFS Snapshots are read-only point-in-time copies of the file system. Snapshots can be taken on a subtree of the file system or the entire file system.
- They are used for data backup, protection against user errors and disaster recovery.
- Snapshot creation is instantaneous.
- Blocks in DataNodes are not copied—the snapshot files record the block list and the file size. There is no data copying.
- Snapshots do not adversely affect regular HDFS operations.

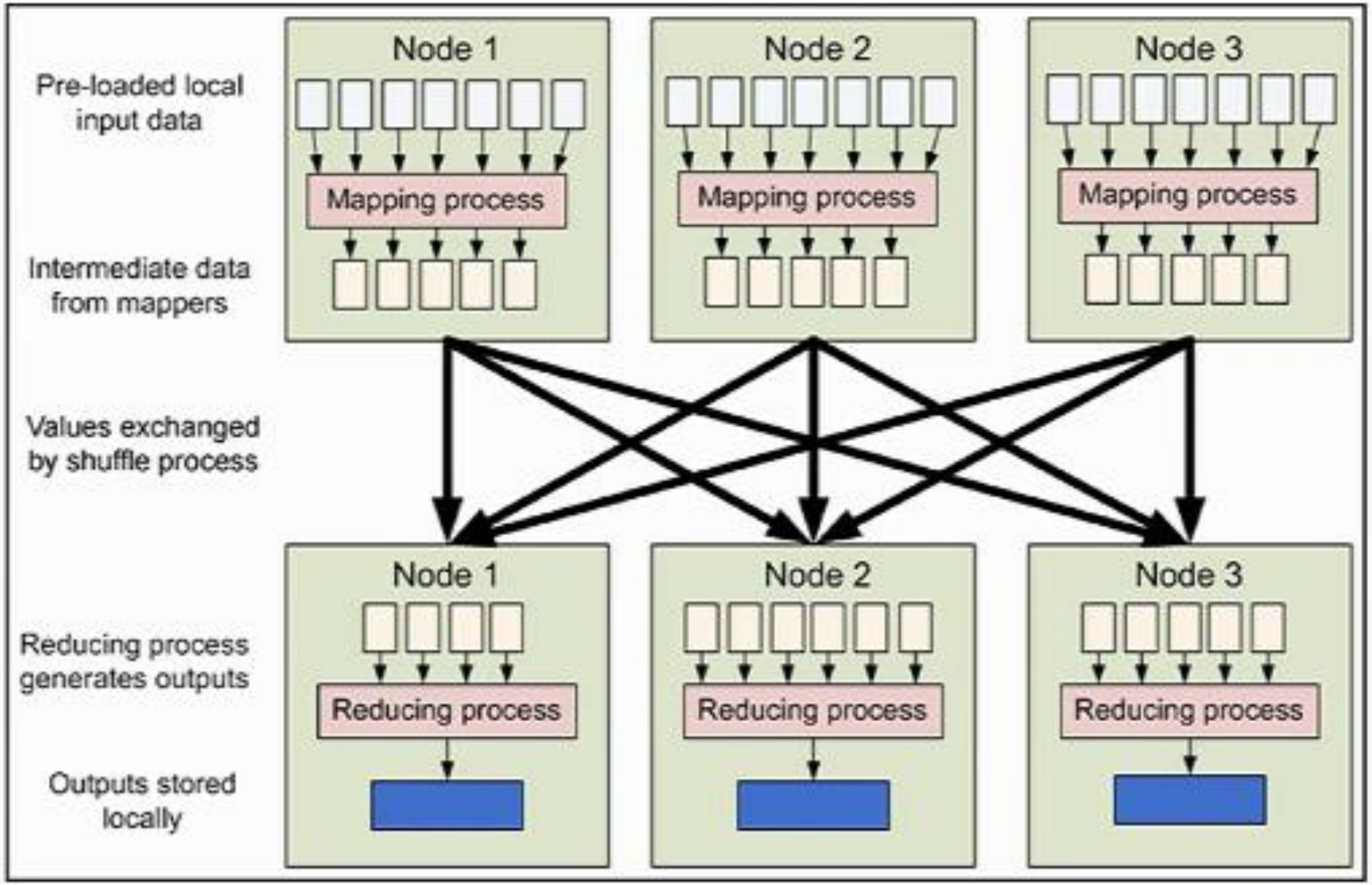
NFSv3 Access to HDFS

The HDFS NFS Gateway supports NFSv3 and allows HDFS to be mounted as part of the client's local file system. Users can browse the HDFS file system through their local file system on NFSv3 client compatible operating systems.

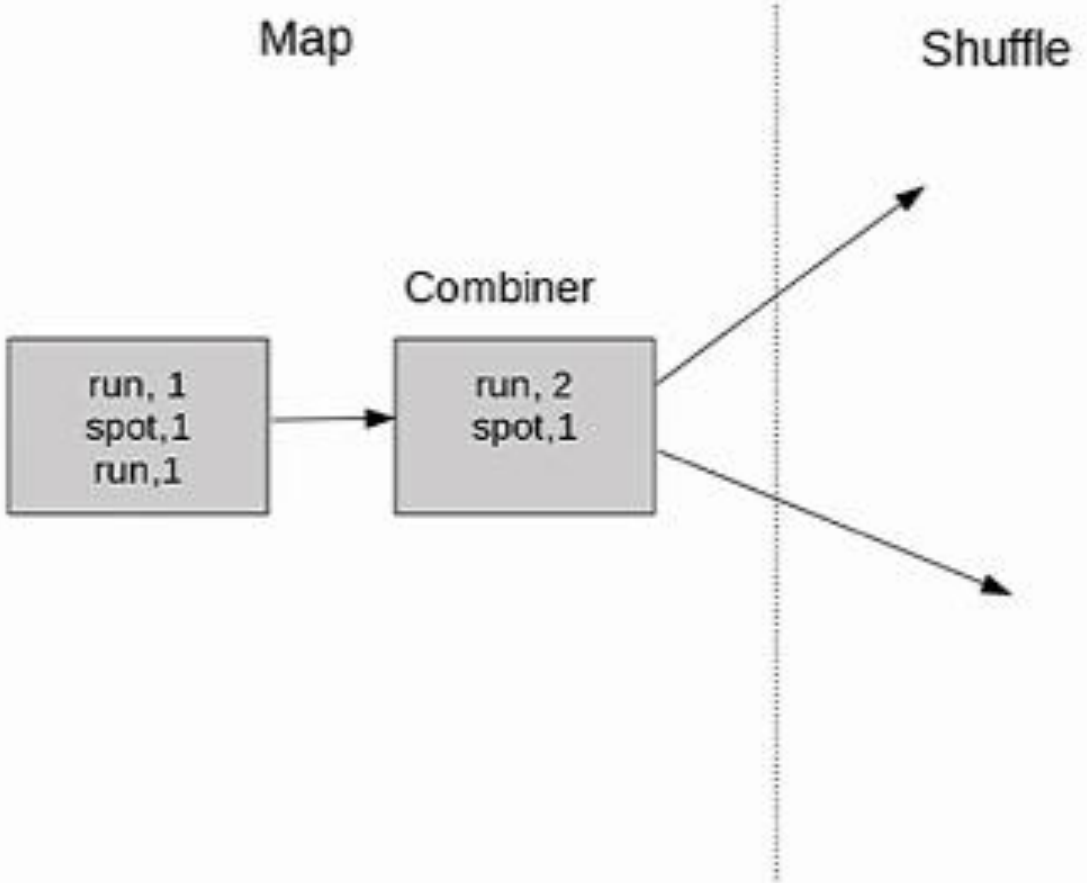
- Users can easily download/upload files from/to the HDFS file system to/from their local file system.
- Users can stream data directly to HDFS through the mount point. File append is supported but random write is not supported.

MapReduce Word Count Data Flow





Adding A Combiner To the Mapper



Fault Tolerance and Speculative Execution

- Faults are handled by restarting tasks
- All managed in the background by JobTracker and TaskTrackers.
- No need to manage side-effects or process state
- Speculative execution helps prevent bottlenecks