Next let us look at moving data into Hadoop.

We will begin by looking at Flume's architecture, then examine the three modes it can run in followed by a look at the event data model.

Flume is an open source software program developed by Cloudera that acts as a service for aggregating and moving large amounts of data around a Hadoop cluster as the data is produced or shortly thereafter. Its primary use case is the gathering of log files from all the machines in a cluster to persist them in a centralized store such as HDFS.

This topic is not intended to cover all aspects of Sqoop but to give you an idea of the capabilities of Sqoop.

Sqoop is an open source product designed to transfer data between relational database systems and Hadoop. It uses JDBC to access the relational systems.

Sqoop accesses the database in order to understand the schema of the data. It then generates a MapReduce application to import or export the data. When you use Sqoop to import data into Hadoop, Sqoop generates a Java class that encapsulates one row of the imported table. You have access to the source code for the generate class. This can allow you to quickly develop any other MapReduce applications that use the records that Sqoop stored into HDFS.

In Flume, you create data flows by building up chains of logical nodes and connecting them to sources and sinks. For example, say you wish to move data from an Apache access log into HDFS. You create a source by tailing access.log and use a logical node to route this to an HDFS sink.

Most production Flume deployments have a three tier design. The agent tier consists of Flume agents colocated with the source of the data that is to be moved. The collector tier consists of perhaps multiple collectors each of which collects data coming in from multiple agents and forwards it on to the storage tier which may consist of a file system such as HDFS.

Here is an example of such a design. Say we have four http server nodes producing log files labelled httpd_logx where x is a number between 1 and 4. Each of these

http server nodes has a Flume agent process running on it. There are two collector nodes. Collector1 is configured to take data from Agent1 and Agent2 and route it to HDFS. Collector2 is configured to take data from Agent3 and Agent4 and route it to HDFS. Having multiple collector processes allows one to increase the parallelism in which data flows into HDFS from the web servers.

Oozie is an open source job control component used to manage Hadoop jobs.

Oozie workflows are collections of actions arranged in a Direct Acyclic Graph. There is a control dependency between actions in that a second action cannot run until the proceeding action completes. For example, you have the capability of having one job execute only when a previous job completed successfully. You can specify that several jobs are permitted to execute in parallel but a final job must wait to begin executing until all parallel jobs complete. Workflows are written in hPDL an XML process definition language, and are stored in a file called workflow.xml.

Each workflow action starts jobs in some remote system and that remote system performs a callback to Oozie to notify that the action completed.

The coordinator component can invoke workflows based upon a time interval, that is for example, once every 15 minutes, or based upon the availability of specific data. It might also be necessary to connect workflow jobs that run regularly but at different time intervals. For example, you may want the output of the last four jobs that run every 15 minutes to be the input to a job that runs every hour.

A single workflow can invoke a single task or multiple tasks, either in sequence or based upon some control logic.

The is the end of this unit and the course. As I said earlier, the components discussed in this unit have Big Data University courses that cover them in more detail. Thank you for attending.

Here is a list of trademarks that may have been referenced in this unit.