Transcript name: Flume – Part 1

English
Welcome to the unit of Hadoop Fundamentals on Flume. 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. 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. Controlling all of these agent and collector processes is a process called the master. The master is capable of reconfiguring routes on the fly to, say, decommission a collector node that is performing poorly and commission a new one. The master can communicate with any and all nodes. Flume was designed with four criteria in mind: reliability, scalability, manageability, and extensibility. We will examine each of these criteria in turn. Because there is tradeoff between reliability and performance, Flume supports multiple levels of reliability to suite the user's needs. End-to-end reliability guarantees that any event, such as an update to a log file, that is accepted into Flume makes it to the endpoint, such as the HDFS. "Store on failure" reliability guarantees

that an accepted event makes it to the next node in the chain, but doesn't guarantee reaching the endpoint. "Best effort" reliability makes no guarantees but has the least overhead. These reliability levels can be specified for each individual flow.

Flume's three tier architecture helps it to achieve a high level of scalability. Ihe individual layers within that architecture are also designed with scalability in mind. The collector tier lets you have multiple collectors to parallelize the workload and Flume can randomly assign flows to connectors to balance the workload among them. Flume also buffers data so that the storage tier avoids spikes from high-volume bursts of data. Using a scalable storage system like HDFS also wins you scalability at the storage layer. Even Flume's control mechanism is designed to scale through the use of multiple masters replicating their state.

In large distributed systems like the kinds you can build with Flume, centralized management and monitoring is critical. Flume can automatically make configuration changes in response to changes in the underlying system. For example, Flume can handle imbalances in the workload, failures of individual nodes, and Flume can automatically reconfigure itself to take advantage of newly provisioned hardware. You can build node topologies in Flume of arbitrary shape, size and complexity - whatever suits your particular data movement needs. Flume provides a data flow specification language that lets you easily build up these topologies.

Finally, Flume is extensible. You can add new sources as source extensions, add new kinds of sinks using sink extensions, and even apply processing to data as moves through a flow through the use of decorators, which can also be extended.

This lesson is continued in the next video.