Data Input

Jim Harner

1/6/2020

4.2 Data Import

Loading data into HDFS, or other persistent data stores such as HBase, generally is not done by copying data from the local filesystem to HDFS. If the data fits in the local filesystem, it probably does not need to be analyzed in Hadoop.

Within the framework of a batch architecture, the two most common systems for loading data are Sqoop and Flume. At this time, neither are implemented, but we will discuss their use cases.

4.2.1 Sqoop

Sqoop performs ETL processing by transferring data from a relational database, or other structured datastore, to HDFS. Sqoop can also be used to extract data from Hadoop and export it into external structured datastores. Sqoop works with the major databases, including PostgreSQL. Sqoop leverages Hadoop's execution engine, MapReduce, to perform data transfers.

Importing data into HDFS would look something like this:

```
sqoop import \
  --connect jdbc:postgresql://url/dataexpo \
  --username sqoop \
  --password sqoop \
  --table location_table
```

A directory will be created called `location_table` with a csv file containing the data. It is possible to specify the target directory rather than using your home directory. The password can hidden by a password file or by prompt.

The user can import subsets of the data (using the `-where` option, or incrementally import mutable data. It is also possible to import all tables or join tables (using a `--query` option with `SELECT`). Finally binary data can be imported.

Data can also be imported directly into Hive or HBase. The additional option `--hive-import` is used for Hive. For HBase, use the `--hbase-table location_table` and `--column-family` options.

The `sqoop export` command can be used to export data from HDFS to a relational database.

4.2.2 Flume

Flume is a distributed service for efficiently collecting, aggregating, and moving large amounts of streaming data into HDFS. It has a simple and flexible architecture based on streaming data flows; and is robust and fault tolerant with reliability mechanisms for failover and recovery.

Flume allows HDFS and HBase clusters to handle bursts of data without having the capacity to handle that rate of writes continuously. These systems act as a buffer between the data producers and the final destination (HDFS or HBase) allowing a steady state of flow.

Flume's HDFS and HBase sinks provide features that makes it possible to write data in any format that is supported by these systems in a MapReduce friendly way.

The simplest unit of Flume deployment is a Flume *agent*. By connecting multiple Flume agents to each other, a *flow* is established. Each Flume agent has three components:

- the source: gets events into the Flume agent;
- the channel: is a buffer that stores data that the source has received, until a sink has successfully written the data out to the next hop or the eventual destination;
- the sink: removes the events from the agent and forwarding them to the next agent in the topology, or to HDFS, HBase, etc.

Flume agents can be configured to send data from one agent to another to form a *pipeline* before the data is written out to the destination. Generally the workflow is written in Java, but command tools are available.