Transcript name: Working with Pig

English
Let us examine Pig in detail. Pig consists of a language and an execution environment. The language is called PigLatin. There are two choices of execution environment: a local environment and distributed environment. A local environment is good for testing when you do not have a full distributed Hadoop environment deployed. You tell Pig to run in the local environment when you start Pig's command line interpreter by passing it the -x local option. You tell Pig to run in a distributed environment by passing -x mapreduce instead. Alternatively, you can start the Pig command line interpreter without any arguments and it will start it in the distributed environment. There are three different ways to run Pig. You can run your PigLatin code as a script, just by passing the name of your script file to the pig command. You can run it interactively through the grunt command line launched using pig with no script argument. Finally, you can call into Pig from within Java using Pig's embedded form. Now let us look into the details of the PigLatin language. A PigLatin progam is a collection of statements. A statement can either be an operation or a command. For example, to load data from a file, issue the LOAD operation with a file name as an argument. A command could be an HDFS command used directly within PigLatin such as the ls command to list, say, all files with an extension of txt in the current directory. The execution of a statement does not necessarily immediately result in a job running on the Hadoop cluster. All commands and certain operators, such as DUMP will start up a job, but other operations simply get added to a logical plan. This plan gets compiled to a physical plan and executed once a data flow has been fully constructed, such as when a DUMP or STORE statement is issued. Here are some of the kinds of statements in PigLatin. There are UDF statements that you can use to REGISTER a user-defined function into PigLatin and DEFINE a short form for referring to it. I mentioned that HDFS commands are a form of PigLatin statement. Other commands include MapReduce commands and Utility commands. There are also diagnostic operators such as DESCRIBE that works much like an SQL DESCRIBE to show you the schema of the data. The largest number of operators fall under the relational operators category. Here we have the operators to LOAD data into the program, to DUMP data to the screen, and to STORE data to disk. We also have operators for filtering, grouping, sorting, combining and splitting data. The relational operators produce relations as their output. A relation is a collection of tuples. Relations can be named using an alias. For example, the LOAD operator produces a relation based on what it reads from the file you pass it. You can assign

this relation to an alias, say, x. If you DUMP the relation aliased by x by issuing "DUMP x", you get the collection of tuples, one for each row of input data. In this example, there is just the one tuple. When we ran LOAD on the file, we specified a schema with the AS argument. We can see this schema by running the DESCRIBE operator on the relation aliased by x.

Statements that contain relational operators may also contain expressions. There are many different kinds of expressions. For example, a constant expression is simply a literal such as a number. A field expression lets you reference a field by its position. You specify it with a dollar sign and a number.

PigLatin supports the standard set of simple data types like integers and character arrays. It also supports three complex types: the tuple as discussed previously, the bag, which is simply an unordered collection of tuples, and a map which is a set of key-value pairs with the requirement that the keys have a type of chararray.

PigLatin also supports functions. These include eval functions, which take in multiple expressions and produce a single expression. For example, you can compute a maximum using the MAX function. A filter function takes in a bag or map and returns a boolean. For example, the isEmpty function can tell you if the bag or map is empty.

A load function can be used with the LOAD operator to create a relation by reading from an input file. A store function does the reverse. It reads in a relation and writes it out to a file.

You can write your own eval, filter, load, or store functions using PigLatin's UDF mechanism. You write the function in Java, package it into a jar, and register the jar using the REGISTER statement. You also have the option to give the function a shorter alias for referring to it by using the DEFINE statement.

Let us now take a look at Apache Pig in action.

First, we need to start Hadoop using the start.sh script.

We will be using a data set on U.S. foreign aid available at data.gov.

Let us examine the data in an Open Office spreadsheet first.

We will import it as comma separated values.

We see two columns: the country and the amount of foreign aid. Countries repeat because the spending program column has been removed.

We need to copy this file into HDFS which we do with the hadoop fs -put command.

We will keep the same name.

Now let's launch the pig executable. It is not in the PATH, so we fully qualify it.

This gives us the grunt prompt.

We start by loading our data from the comma-separate value file, pass a comma to PigStorage, and apply a schema.

Next we group by country, assigning the result to an alias called grouped.

For each group, we want to generate a sum of the values in the sum column.

We assign this to an alias called thesum.

Then we DUMP thesum.

This runs Hadoop and gives us our answer.

This lesson is continued in the next video.