Transcript name: Pig, Hive and Jaql overview

English

Welcome to the unit of Hadoop Fundamentals on Pig, Hive and Jaql.

In this unit, we will examine three different technologies that make it easier to write MapReduce programs in Hadoop. We will begin with an overview of all three technologies, comparing and contrasting them with each other, then we will examine each technology in detail and see how to write MapReduce programs using them.

Pig, Hive, and Jaql have much in common. They all translate high-level languages into MapReduce jobs so that the programmer can work at a higher level than he or she would when writing MapReduce jobs in Java or other lower-level languages supported by Hadoop using Hadoop streaming. The high level languages offered by Pig, Hive and Jaql let you write programs that are much smaller than the equivalent Java code. When you find that you need to work at a lower level to accomplish something these high-level languages do not support themselves, you have the option to extend these languages, often by writing user-defined functions in Java. Interoperability can work both ways since programs written in these high-level languages can be imbedded inside other languages as well. Finally, since all these technologies run on top of Hadoop, when they do so, they have the same limitations with respect to random reads and writes and low-latency queries as Hadoop does.

Now, let us examine what is unique about each technology, starting with Pig. Pig was developed at Yahoo Research around 2006 and moved into the Apache Software Foundation in 2007. Pig's language, called PigLatin, is a data flow language - this is the kind of language in which you program by connecting things together. Pig can operate on complex data structures, even those that can have levels of nesting. Unlike SQL, Pig does not require that the data have a schema, so it is well suited to processing unstructured data. However, Pig can still leverage the value of a schema if you choose to supply one. Like SQL, PigLatin is relationally complete, which means it is at least as powerful as relational algebra. Turing completeness requires looping constructs, an infinite memory model, and conditional constructs. PigLatin is not Turing complete on its own, but is Turing complete when extended with User-Defined Functions.

Hive is a technology developed at Facebook that turns Hadoop into a data warehouse complete with a dialect of SQL for querying. Being a SQL dialect, HiveQL is a declarative language. Unlike in PigLatin, you do not specify the data flow, but instead describe the result you want and Hive figures out how to build a data flow to achieve it. Also unlike Pig, a schema is required, but you are not limited to one schema. Like PigLatin and SQL, HiveQL on its own is a relationally complete language but not a Turing complete language. It can be extended through UDFs just like Pig to be Turing complete.

The final technology is Jaql. Jaql was developed at IBM. It is a data flow language like PigLatin but its native data structure format is JavaScript Object Notation, or JSON. Schemas are optional and the Jaql language itself is Turing complete on its own without the need for extension through UDFs.

This lesson is continued in the next video.