**Experiment-9**

**Install and Running Pig then write Pig Latin scripts to sort, group your data.**

**Experiment 9.1**

**Aim: To install and run Pig**

Apache Pig raises the level of abstraction for processing large datasets. MapReduce allows us, as the programmer, to specify a map function followed by a reduce function, but working out how to fit your data processing into this pattern, which often requires multiple MapReduce stages, can be a challenge. With Pig, the data structures are much richer, typically being multivalued and nested, and the transformations we can apply to the data are much more powerful.

Pig is made up of two pieces:

* The language used to express data flows, called Pig Latin.
* The execution environment to run Pig Latin programs. There are currently two environments: local execution in a single JVM and distributed execution on a Hadoopcluster.

A Pig Latin program is made up of a series of operations, or transformations, that are applied to the input data to produce output. Taken as a whole, the operations describe a data flow, which the Pig execution environment translates into an executable representation and then runs. Under the covers, Pig turns the transformations into a series of MapReduce jobs, but as a programmer we are mostly unaware of this, which allows us to focus on the data rather than the nature of the execution.

Pig is a scripting language for exploring large datasets. Pig is very supportive of a programmer writing a query, since it provides several commands for introspecting the data structures in our program as it is written. Even more useful, it can perform a sample run on a representative subset of our input data, so we can see whether there are errors in the processing before unleashing it on the full dataset.

1. **Installing and Running Pig**

Pig runs as a client-side application. Even if you want to run Pig on a Hadoop cluster, there is nothing extra to install on the cluster: Pig launches jobs and interacts with HDFS (or other Hadoop filesystems) from your workstation.

**Prerequisites**

You must install the Hadoop cluster as per the previously given guidelines.

Download Apache Pig

Go to the link

<https://dlcdn.apache.org/pig/latest/>

and then download the following tarball

Pig 0.17.0.tar.gz

Extract the Pig folder at a suitable place in your c drive.

Setting Environment Variables

PIG\_HOME= path to the pig folder

Now, we should edit the Path user variable to add the following paths:

%PIG\_HOME%\bin

**Starting Apache Pig**

**Note**: Hadoop Services must be running

Edit the pig.cmd file located in the “pig-0.17.0\bin” directory by changing the HADOOP\_BIN\_PATH value from “%HADOOP\_HOME%\bin” to “%HADOOP\_HOME%\libexec”.

Now, let's try to run the “pig -version” command:

The simplest way to write PigLatin statements is using Grunt shell which is an interactive tool where we write a statement and get the desired output. There are two modes to involve Grunt Shell:

1. Local: All scripts are executed on a single machine without requiring Hadoop.

pig -x local

1. MapReduce: Scripts are executed on a Hadoop cluster

pig -x MapReduce

**Execution Types**

Pig has two execution types or modes: local mode and MapReduce mode.

**Local mode**

In local mode, Pig runs in a single JVM and accesses the local file system. This mode is suitable only for small datasets and when trying out Pig.

The execution type is set using the -x or -exectype option. To run in local mode, set the option to local:

% pig -x local

grunt>

This starts Grunt, the Pig interactive shell, which is discussed in more detail shortly.

**MapReduce mode**

In MapReduce mode, Pig translates queries into MapReduce jobs and runs them on a Hadoop cluster. The cluster may be a pseudo- or fully distributed cluster. MapReduce mode (with a fully distributed cluster) is what we use when you want to run Pig on large datasets.

To use MapReduce mode, we first need to check that the version of Pig we downloaded is compatible with the version of Hadoop we are using. Pig releases will only work against particular versions of Hadoop; this is documented in the release notes.

It is required to set the HADOOP\_HOME environment variable as it is needed to find which Hadoop client to run.

Next, you need to point Pig at the cluster’s namenode and resource manager.

We need to set the properties in the pig.properties file in Pig’s conf directory (or the directory specified by PIG\_CONF\_DIR). Here’s an example for a pseudodistributed setup:

fs.defaultFS=hdfs://localhost/

mapreduce.framework.name=yarn

yarn.resourcemanager.address=localhost:8032

Once we have configured Pig to connect to a Hadoop cluster, we can launch Pig, setting the -x option to mapreduce or omitting it entirely, as MapReduce mode is the default.

We’ve used the -brief option to stop timestamps from being logged:

% pig -brief

Logging error messages to: /Users/tom/pig\_1414246949680.log

Default bootup file /Users/tom/.pigbootup not found

Connecting to hadoop file system at: hdfs://localhost/

grunt>

As we can see from the output, Pig reports the filesystem (but not the YARN resource manager) that it has connected to.

**Running Pig Programs**

There are three ways of executing Pig programs, all of which work in both local and MapReduce mode:

1. **Grunt**

Grunt is an interactive shell for running Pig commands. Grunt is started when no file is specified for Pig to run and the -e option is not used. It is also possible to run Pig scripts from within Grunt using **run** and **exec**.

1. **Script**

Pig can run a script file that contains Pig commands. For example, pig script.pig runs the commands in the local file script.pig. Alternatively, for very short scripts, you can use the -e option to run a script specified as a string on the command line.

1. **Embedded**

You can run Pig programs from Java using the PigServer class, much like you can use JDBC to run SQL programs from Java. For programmatic access to Grunt, use PigRunner.

1. **Grunt Shell**

Grunt has line-editing facilities for instance, the Ctrl-E key combination will move the cursor to the end of the line. Grunt remembers command history, too, and we can recall lines in the history buffer using Ctrl-P or Ctrl-N (for previous and next), or equivalently, the up or down cursor keys.

Example

The Grunt shell is useful to run small Pig Latin code for testing in development and testing environments.

Sample code in the Grunt shell:

Grunt>emp = load '/data/employee' using PigStorage() as (eno:int,ename:chararray,salary:chararray,deptno:int);

Another handy feature is Grunt’s completion mechanism, which will try to complete Pig Latin keywords and functions when you press the Tab key. For example, consider the following incomplete line:

grunt> a = foreach b ge

If we press the Tab key at this point, ge will expand to generate, a Pig Latin keyword:

grunt> a = foreach b generate

1. **Running Pig scripts**

Pig -e

When you have to run a list of Pig Latin statements, you can use the pig -e command.

Here’s the syntax:

Pig -e " <pig latin statement>"

The following code uses the Pig –e command, and the semicolon represents the end of a statement:

Pig -e "emp = load '/data/employee' using PigStorage(',');dump emp;"

Pig -f

The Pig –e command is not readable if it has multiple lines of code. Pig –f allows you to embed Pig Latin code in a file.

Syntax:

Pig -f /path/to/piglatin/file

Although the file can be saved with any extension, it is better to save the file with a .pig extension to distinguish it from other script files. This is the standard way to write Pig Latin code in a production environment.

Example

Cat /home/hdfs/dumpemp.pig

emp = load '/data/employee' using PigStorage(',') as (eno:int,ename:chararr

ay,salary:int,deptno:int) ;

dump emp;

pig -f /home/hdfs/dumpemp.pig