Prof R. Madana Mohana

BIG DATA ANALYTICS

Apache Pig

Introduction | Installing & Running Pig

https://www.youtube.com/c/RASINENIMADANAMOHANA

Apache Pig: Introduction

Apache Pig: *Introduction*

- **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 our data processing into this pattern, which often requires multiple **MapReduce** stages, can be a challenge.

Apache Pig: *Introduction*

- 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. They include joins, for example, which are not for the faint (indefinite) of heart in **MapReduce**.

Apache Pig: *Introduction*

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 **Hadoop cluster**.

Apache Pig: *Introduction*

- 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.

Apache Pig: *Introduction*

- **Pig** is a scripting language for exploring large datasets.
- One criticism of **MapReduce** is that the development cycle is very long.
- Writing the **mappers** and **reducers**, compiling and packaging the code, submitting the job(s), and retrieving the results is a time consuming business, and even with Streaming, which removes the compile and package step, the experience is still involved.

Apache Pig: *Introduction*

- **Pig**'s ability is to process terabytes of data in response to a half-dozen lines of **Pig Latin** issued from the console.
- **Pig** was created at **Yahoo!** to make it easier for researchers and engineers to mine the huge datasets.
- **Pig** is very supportive of a programmer writing a query, since it provides several commands for self analysing the data structures in our program as it is written.

Apache Pig: *Introduction*

- **Pig** 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 (release) it on the full dataset.
- **Pig** was designed to be extensible.
- Virtually all parts of the processing path are customizable: **loading**, **storing**, **filtering**, **grouping**, and **joining** can all be altered by **user-defined functions** (**UDF**s).

Apache Pig: *Introduction*

- **user-defined functions** (**UDF**s) operate on **Pig**'s nested data model, so they can integrate very deeply with **Pig**'s operators.
- As another benefit, **UDF**s tend to be more reusable than the libraries developed for writing **MapReduce** programs.

Apache Pig: *Introduction*

- In some cases, **Pig** doesn't perform as well as programs written in **MapReduce.**
- However, the gap is narrowing with each release, as the **Pig** team implements sophisticated algorithms for applying **Pig**'s relational operators. It's fair to say that unless we are willing to invest a lot of effort optimizing **Java** MapReduce code, writing queries in **Pig Latin** will save our time.

Apache Pig: *Introduction*

- **Apache Pig** introduction is shown in below video:

https://youtu.be/5cIVtDFfaOM

Apache Pig: Installing & Running Pig

Apache Pig: *Installing and Running Pig*

- **Pig** runs as a client-side application.
- Even if we 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 our workstation.
- Installation is straightforward. Download a stable release from http://pig.apache.org/releases.html, and unpack the **tarball** in a suitable place on our workstation.

Apache Pig: *Installing and Running Pig*

- **Pig** Installation and running **Pig** on **Ubuntu** is shown in below video:

https://youtu.be/21WGMxl0ZAE

Apache Pig: *Installing and Running Pig*

Execution Types:

- **Pig** has *two* execution types or modes: **local mode** and **MapReduce mode**.
- Execution modes for **Apache Tez** and **Spark** were both under development at the time of writing.
- Both promise significant performance gains over **MapReduce** mode, so try them if they are available in the version of **Pig** we are using.

Apache Pig: *Installing and Running Pig*

Execution Types: *Local mode*

- In local mode, **Pig** runs in a single **JVM** and accesses the local filesystem.
- This mode is suitable only for small datasets and when trying out **Pig**.
- The execution type is set using the **-x** or **-exectype** option.

Apache Pig: *Installing and Running Pig*

Execution Types: *Local mode*

To run in local mode, set the option to local:

```
% pig -x local
grunt>
```

This starts **Grunt**, the **Pig interactive shell.**

Apache Pig: *Installing and Running Pig*

Execution Types: *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 we 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.

Apache Pig: *Installing and Running Pig*

Execution Types: *MapReduce mode*

- **Pig** releases will only work against particular versions of **Hadoop**; this is documented in the release notes.
- **Pig** honors the **HADOOP_HOME** environment variable for finding which **Hadoop client** to run. However, if it is not set, **Pig** will use a bundled copy of the **Hadoop libraries**.

Apache Pig: *Installing and Running Pig*

Execution Types: *MapReduce mode*

- Next, we need to point **Pig** at the cluster's **namenode** and **resource manager**.
- If the installation of Hadoop at **HADOOP_HOME** is already configured for this, then there is nothing more to do.
- Otherwise, we can set **HADOOP_CONF_DIR** to a directory containing the Hadoop site file (or files) that define **fs.defaultFS**, **yarn.resourcemanager.address**, and **mapreduce.framework.name** (the latter should be set to **yarn**).

Apache Pig: *Installing and Running Pig*

Execution Types: *MapReduce mode*

- Alternatively, we can set these properties in the ***pig.properties*** file in **Pig**'s ***conf*** directory (or the directory specified by **PIG_CONF_DIR**).

Example for a pseudo distributed setup:

```
fs.defaultFS=hdfs://localhost/
mapreduce.framework.name=yarn
yarn.resourcemanager.address=localhost:8032
```

Apache Pig: *Installing and Running Pig*

Execution Types: *MapReduce mode*

- 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
```

Apache Pig: *Installing and Running Pig*

Execution Types: *MapReduce mode*

In **MapReduce** mode, we can optionally enable ***auto-local mode*** (by setting **pig.auto.local.enabled** to **true**), which is an optimization that runs small jobs locally if the input is less than **100 MB** (set by **pig.auto.local.input.maxbytes**, default **100,000,000**) and no more than one reducer is being used.

Apache Pig: *Installing and Running Pig*

Running Pig Programs:

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

1. Script
2. Grunt
3. Embedded

Apache Pig: *Installing and Running Pig*

Running Pig Programs:

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,we can use the **-e** option to run a script specified as a string on the command line.

Apache Pig: *Installing and Running Pig*

Running Pig Programs:

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**.

Apache Pig: *Installing and Running Pig*

Running Pig Programs:

Embedded:

- We can run **Pig** programs from **Java** using the **PigServer** class, much like we can use **JDBC** to run **SQL** programs from **Java**.
- For programmatic access to **Grunt**, use **PigRunner**.

Apache Pig: *Installing and Running Pig*

Grunt:

- **Grunt** has line-editing facilities like those found in **GNU** Readline (used in the bash shell and many other command-line applications).
- 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.

Apache Pig: *Installing and Running Pig*

Grunt:

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

```
grunt> a = foreach b ge
```

Apache Pig: *Installing and Running Pig*

Grunt:

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

```
grunt> a = foreach b generate
```

Apache Pig: *Installing and Running Pig*

Grunt:

- We can customize the completion tokens by creating a file named ***autocomplete*** and placing it on **Pig**'s classpath (such as in the ***conf*** directory in Pig's ***install*** directory) or in the directory we invoked **Grunt** from.
- The file should have one token per line, and tokens must not contain any whitespace.

Apache Pig: *Installing and Running Pig*

Grunt:

- Matching is case sensitive. It can be very handy to add commonly used file paths (especially because **Pig** does not perform filename completion) or the names of any user-defined functions we have created.
- We can get a list of commands using the **help** command.
- When we've finished our **Grunt session**, we can exit with the **quit** command, or the equivalent shortcut **\q**.

Apache Pig: *Installing and Running Pig*

Pig Latin Editors:

- There are **Pig Latin** syntax highlighters available for a variety of editors, including **Eclipse**, **IntelliJ IDEA**, **Vim**, **Emacs**, and **TextMate**. Details are available on the Pig wiki.
- Many **Hadoop** distributions come with the Hue web interface, which has a Pig script editor and launcher.