BIG DATA ANALYTICS

Apache Pig

Pig in Practice

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

Pig in Practice: *Outline*

- Parallelism
- Anonymous Relations
- Parameter Substitution

Pig in Practice

- There are some practical techniques that are worth knowing about when we are developing and running Pig programs.
  - Parallelism
  - Anonymous Relations
  - Parameter Substitution

Pig in Practice: Parallelism

- When running in MapReduce mode, it's important that the degree of parallelism matches the size of the dataset.
- By default, Pig sets the number of reducers by looking at the size of the input and using one reducer per 1 GB of input, up to a maximum of 999 reducers.
- We can override these parameters by setting **`pig.exec.reducers.bytes.per.reducer`** (the default is `1,000,000,000` bytes) and **`pig.exec.reducers.max`** (the default is `999`).

Pig in Practice: Parallelism

- To explicitly set the number of reducers we want for each job, we can use a **PARALLEL** clause for operators that run in the reduce phase.
- These include all the grouping and joining operators (**GROUP**, **COGROUP**, **JOIN**, **CROSS**), as well as **DISTINCT** and **ORDER**.
- The following line sets the number of reducers to **30** for the **GROUP**:

```
grouped_records = GROUP records BY year PARALLEL 30;
```

- Alternatively, we can set the `default_parallel` option, and it will take effect for all subsequent jobs:

```
grunt> set default_parallel 30
```

- The number of map tasks is set by the size of the input (with one map per `HDFS block`) and is not affected by the **PARALLEL** clause.

Pig in Practice: Anonymous Relations

- We usually apply a diagnostic operator like **`DUMP`** or **`DESCRIBE`** to the most recently defined relation.
- Since this is so common, Pig has a shortcut to refer to the previous relation: `@`.
- Similarly, it can be tiresome to have to come up with a name for each relation when using the interpreter.
- Pig allows us to use the special syntax `=>` to create a relation with no alias, which can only be referred to with `@`.

Pig in Practice: Anonymous Relations

Example:

```
grunt> => LOAD 'input/ncdc/micro-tab/sample.txt';
grunt> DUMP @
(1950,0,1)
(1950,22,1)
(1950,-11,1)
(1949,111,1)
(1949,78,1)
```

Pig in Practice: Parameter Substitution

- If we have a Pig script that we run on a regular basis, it's quite common to want to be able to run the same script with different parameters.
- For example, a script that runs daily may use the date to determine which input files it runs over.
- Pig supports parameter substitution, where parameters in the script are substituted with values supplied at runtime.

Pig in Practice: Parameter Substitution

- Parameters are denoted by identifiers prefixed with a `$` character; for example, `$input` and `$output` are used in the following script to specify the input and output paths:

```
-- max_temp_param.pig
records = LOAD '$input' AS (year:chararray, temperature:int, quality:int);
filtered_records = FILTER records BY temperature != 9999 AND
  quality IN (0, 1, 4, 5, 9);
grouped_records = GROUP filtered_records BY year;
max_temp = FOREACH grouped_records GENERATE group,
  MAX(filtered_records.temperature);
STORE max_temp into '$output';
```

Pig in Practice: Parameter Substitution

- Parameters can be specified when launching `Pig` using the `-param` option, once for each parameter:

```
% pig -param input = /user/tom/input/ncdc/micro-tab/sample.txt \
> -param output = /tmp/out \
> ch16-pig/src/main/pig/max_temp_param.pig
```

- We can also put parameters in a file and pass them to `Pig` using the `-param_file` option.
- For example, we can achieve the same result as the previous command by placing the parameter definitions in a file:

Pig in Practice: Parameter Substitution

- For example, we can achieve the same result as the previous command by placing the parameter definitions in a file:

```
# Input file
input = /user/tom/input/ncdc/micro-tab/sample.txt
# Output file
output = /tmp/out
```

The pig invocation then becomes:

```
% pig -param_file ch16-pig/src/main/pig/max_temp_param.param \
          > ch16-pig/src/main/pig/max_temp_param.pig
```

- We can specify multiple parameter files by using **`-param_file`** repeatedly.
- We can also use a combination of **`-param`** and **`-param_file`** options; if any parameter is defined both in a parameter file and on the command line, the last value on the command line takes precedence.

Pig in Practice: Parameter Substitution

Dynamic parameters

- For parameters that are supplied using the `-param` option, it is easy to make the value dynamic by running a command or script.
- Many Unix shells support command substitution for a command enclosed in backticks, and we can use this to make the output directory date-based:

```
% pig -param input=/user/tom/input/ncdc/micro-tab/sample.txt \
> -param output=/tmp/`date "+%Y-%m-%d"`/out \
> ch16-pig/src/main/pig/max_temp_param.pig
```

- Pig also supports backticks in parameter files by executing the enclosed command in a shell and using the shell output as the substituted value.
- If the command or script exits with a nonzero exit status, then the error message is reported and execution halts.
- Backtick support in parameter files is a useful feature; it means that parameters can be defined in the same way in a file or on the command line.

Pig in Practice: Parameter Substitution

Parameter substitution processing

- Parameter substitution occurs as a preprocessing step before the script is run.
- We can see the substitutions that the preprocessor made by executing Pig with the `-dryrun` option.
- In dry run mode, Pig performs parameter substitution (and macro expansion) and generates a copy of the original script with substituted values, but does not execute the script.
- We can inspect the generated script and check that the substitutions look well-balanced (because they are dynamically generated, for example) before running it in normal mode.

Pig in Practice: Parameter Substitution

Apache Pig Reference:

- Tom White, "Hadoop: The Definitive Guide", 4th Edition, O'Reilly Media Inc, 2015. (Chapter-16)

https://www.oreilly.com/library/view/hadoop-the-definitive/9781491901687/