BIG DATA ANALYTICS/MANAGEMENT CS 6350

Pig Latin

What is Pig?

- Pig is a platform for analyzing large data sets.
- Pig's language, Pig Latin, lets you specify a sequence of data transformations such as merging data sets, filtering them, and applying functions to records or groups of records.
- At Yahoo! 40% of all Hadoop jobs are run with Pig.

Various Modes

- You can run Pig (execute Pig Latin statements and Pig commands) using various modes.

	Local Mode	**Mapreduce Mode**
Interactive Mode	yes	yes
Batch Mode	yes	yes

Execution Modes using the pig command

- **Local Mode**
  - Local host and file system.
  - /* local mode */
  - $ pig -x local ...

- **Mapreduce Mode**
  - Access to a Hadoop cluster and HDFS installation.
  - /* mapreduce mode */
  - $ pig ...

    or
  - $ pig -x mapreduce ...

Interactive Mode Local Mode

- using the Grunt shell.
- Can be invoked using:
  - $ pig -x local

    ... - Connecting to ...
  - grunt>
  - grunt> A = load '/etc/passwd ' using PigStorage(':');
  - grunt> B = foreach A generate $0 as id;
  - grunt> dump B;

Interactive Mode
Mapreduce Mode

- using the Grunt shell.
- Can be invoked using:
  - $ pig -x mapreduce

    ... - Connecting to ...
  - grunt>

    or
  - $ pig

    ... - Connecting to ...
  - grunt>

Batch Mode

- /* id.pig */ ***HDFS path in case of map-reduce mode***
- A = load '/etc/passwd ' using PigStorage(':'); -- load the passwd file
- B = foreach A generate $0 as id; -- extract the user IDs
- store B into ‘id.out’; -- write the results to a file name id.out

Batch Mode

- **Local Mode**
  - $ pig -x local id.pig

- **Mapreduce Mode**
  - $ pig id.pig

    or
  - $ pig -x mapreduce id.pig

Pig Scripts

- To place Pig Latin statements and Pig commands in a single file.
- Using the *.pig extension is good (please do it for HW too).

Pig Latin Statements

- OutputRelation = InputRelation
- A relation is a bag.
- A bag is a collection of tuples.
- A tuple is an ordered set of fields.
- A field is a piece of data.

Debugging

- Use the DUMP operator to display results to your terminal screen.
- Use the DESCRIBE operator to review the schema of a relation.
- Use the EXPLAIN operator to view the logical, physical, or map reduce execution plans to compute a relation.
- Use the ILLUSTRATE operator to view the step-by-step execution of a series of statements.

Debug: dump

- `A = LOAD 'student' AS (name:chararray, age:int, gpa:float);`

- `DUMP A;`

```
(John,18,4.0F)
(Mary,19,3.7F)
(Bill,20,3.9F)
(Joe,22,3.8F)
(Jill,20,4.0F)
```

Debug: describe

- grunt> A = load '/home/kma041000/pig/input' as (line:chararray);
- grunt> describe A;

  A: {line: chararray}
- grunt>

Debug: explain

```
grunt> explain A
; #-----------------------------------------------
# New Logical Plan:
#-----------------------------------------------
A: (Name: LOStore Schema: line#3:chararray)
|
|---A: (Name: LOForEach Schema: line#3:chararray)
    |   |
    |   (Name: LOGenerate[false] Schema: line#3:chararray)ColumnPrune:InputUids=[3]ColumnPrune:OutputUids=[3]
    |   |   |
    |   |   (Name: Cast Type: chararray Uid: 3)
    |   |   |
    |   |   |---line:(Name: Project Type: bytearray Uid: 3 Input: 0 Column: (*))
    |   |
    |   |---(Name: LOInnerLoad[0] Schema: line#3:bytearray)
    |
    |---A: (Name: LOLoad Schema: line#3:bytearray)RequiredFields:null

#-----------------------------------------------
# Physical Plan:
#-----------------------------------------------
A: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-5
|
|---A: New For Each(false)[bag] - scope-4
    |   |
    |   Cast[chararray] - scope-2
    |   |
    |   |---Project[bytearray][0] - scope-1
    |
    |---A: Load(/home/kma041000/pig/input:org.apache.pig.builtin.PigStorage) - scope-0

2013-03-19 18:19:40,210 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2013-03-19 18:19:40,255 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2013-03-19 18:19:40,255 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
#--------------------------------------------------
# Map Reduce Plan
#--------------------------------------------------
MapReduce node scope-6
Map Plan
A: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-5
|
|---A: New For Each(false)[bag] - scope-4
    |   |
    |   Cast[chararray] - scope-2
    |   |
    |   |---Project[bytearray][0] - scope-1
    |
    |---A: Load(/home/kma041000/pig/input:org.apache.pig.builtin.PigStorage) - scope-0--------
Global sort: false
----------------

grunt>
```

Debug: illustrate

- grunt> illustrate A;

```
---------------------------------------
| A     | line:chararray              |
---------------------------------------
|       | word count in pig tutorial |
---------------------------------------
```

Pig WordCount – Batch mode

- Script file name: wordcount.pig
- Contains:

```
------------
A = load '/home/kma041000/pig/input';
B = foreach A generate flatten(TOKENIZE((chararray)$0)) as
    word;
C = group B by word;
D = foreach C generate COUNT(B), group;
/* rm '/home/kma041000/pig/output'; */
store D into '/home/kma041000/pig/output';
--------------
```

Pig WordCount – Batch mode

- To run wordcount.pig
- Batch mode:
- {cs6360:~/BigData/Pig} pig -x mapreduce wordcount.pig
- .
- . *Hadoop map/reduce is running…*
- .
- {cs6360:~/BigData/Pig} hadoop fs -cat /home/kma041000/pig/output/part-r-00000
- 4	in
- 2	for
- 4	pig
- 4	2012
- 2	word
- 2	count
- 4	school
- 4	summer
- 2	indiana
- 4	tutorial

Pig WordCount – Interactive mode

- {cs6360:~/BigData/Pig} pig
- grunt> A = load '/home/kma041000/pig/input';
- grunt> B = foreach A generate flatten(TOKENIZE((chararray) $0)) as word;
- grunt> C = group B by word;
- grunt> D = foreach C generate COUNT(B), group;
- grunt> rm '/home/kma041000/pig/output';
- grunt> dump D; /* to see output in terminal */
- grunt> store D into '/home/kma041000/pig/output'; /* to part-r-00000 file */

Pig WordCount – Interactive mode

- grunt> cat /home/kma041000/pig/output/part-r-00000
- 4	in
- 2	for
- 4	pig
- 4	2012
- 2	word
- 2	count
- 4	school
- 4	summer
- 2	indiana
- 4	tutorial
- Or
- grunt> quit;
- {cs6360:~/BigData/Pig} hadoop fs -cat /home/kma041000/pig/output/part-r-00000

Pig UDF

- Pig provides extensive support for user-defined functions (UDFs) as a way to specify custom processing.
- Functions can be a part of almost every operator in Pig.

Sample UDF Function

```
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.pig.EvalFunc;
import org.apache.pig.backend.executionengine.ExecException;
import org.apache.pig.data.Tuple;

public class ConvertToUpper extends EvalFunc <String> {

@Override
    public String exec(Tuple input) {
        try {
            if (input == null || input.size() == 0) {
                return null;
            }

            String str = (String) input.get(0);
            return str.toUpperCase();
        } catch (ExecException ex) {
            System.out.println("Error: " + ex.toString());
        }

        return null;
    }
}
```

Compile & Run Jar

Compiling java file:

```
{cs6360:~} mkdir PIG_UDF
{cs6360:~} cd PIG_UDF
```

Copy the ConvertToUpper.java file to this directory.

```
{cs6360:~/PIG_UDF} javac -cp /usr/local/pig-0.10.1/pig-0.10.1.jar
    ConvertToUpper.java
```

Create the jar file

```
{cs6360:~/PIG_UDF} jar -cf pig_udf.jar .
```

Create Pig Script

Create a script pig_script.pig in PIG_UDF folder. The script has the following:

```
REGISTER /people/cs/m/mxs121731/HW_TA/PIG_UDF/pig_udf.jar;

A = LOAD '/HW_3_Data/movies_new' using PigStorage(';') as (MOVIEID:
    chararray, TITLE: chararray, GENRE: chararray);

B = FOREACH A GENERATE MOVIEID,  ConvertToUpper(TITLE), GENRE;
C = LIMIT B 10;
DUMP C;
```

You can also do it in interactive mode (using grunt shell)

NB: The location of the pig_udf.jar file should be changed according to your location

Run Pig Script

Running pig_script.pig

```
{cs6360:~/PIG_UDF}  pig -x mapreduce pig_script.pig
```

Output

```
(1,TOY STORY (1995),Animation|Children's|Comedy)
(2,JUMANJI (1995),Adventure|Children's|Fantasy)
(3,GRUMPIER OLD MEN (1995),Comedy|Romance)
(4,WAITING TO EXHALE (1995),Comedy|Drama)
(5,FATHER OF THE BRIDE PART II (1995),Comedy)
(6,HEAT (1995),Action|Crime|Thriller)
(7,SABRINA (1995),Comedy|Romance)
(8,TOM AND HUCK (1995),Adventure|Children's)
(9,SUDDEN DEATH (1995),Action)
(10,GOLDENEYE (1995),Action|Adventure|Thriller)
```

Pig Example

Query: List the top 10 average rated movies in descending order.

```
A = load  '/HW_3_Data/ratings_new ' using PigStorage(';') as (USERID:int,
      MOVIEID:int, RATING:double, TIMESTAMP:chararray);
B = group A by MOVIEID;
C = foreach B generate group, AVG(A.RATING) as avgRating;
D = order C by avgRating desc;
E = limit D 10;
dump E;
```

Output:

```
(3280,5.0)
(989,5.0)
(3656,5.0)
(1830,5.0)
(3607,5.0)
(787,5.0)
(3382,5.0)
(3172,5.0)
(3881,5.0)
(3233,5.0)
```

References

http://pig.apache.org/

http://pig.apache.org/docs/r0.10.0/basic.html#comparison

http://pig.apache.org/docs/r0.7.0/udf.html

http://www.bidn.com/blogs/cprice1979/ssas/4218/mmm-more-bacon-pig-user-defined-functions-udfs

Thank You