Zebra and Pig

Table of contents

1. Overview

With Pig you can load and store data in Zebra format. You can also take advantage of sorted Zebra tables for map-side groups and merge joins. When working with Pig keep in mind that, unlike MapReduce, you do not need to declare Zebra schemas. Zebra automatically converts Zebra schemas to Pig schemas (and vice versa) for you.

2. Zebra Pig Examples

2.1. Loading Data

To load data stored in Zebra format (columns) use the TableLoader function.

```
register $LOCATION/zebra-$version.jar;
A = LOAD 'studenttab' USING org.apache.hadoop.zebra.pig.TableLoader();
B = FOREACH A GENERATE name, age, gpa;
```

There are a couple of things to note:

1. You need to register a Zebra jar file the same way you would do it for any other UDF.
2. You need to place the jar on your classpath.

Zebra data is self-described meaning that the name and type information is stored with the data; you don't need to provide an AS clause or perform type casting unless you actually need to change the data. To check column names and types, you can run the DESCRIBE statement right after the load:

```
A = LOAD 'studenttab' USING org.apache.hadoop.zebra.pig.TableLoader();
DESCRIBE A;
A: {name: chararray,age: int,gpa: float}
```

You can provide alternative names to the columns with the AS clause. You can also provide alternative types as long as the original type can be converted to the new type. (One exception to this rule are maps since you can't specify schema for a map. Zebra always creates map values as bytearrays which would require casting to real type in the script. Note that this is not different for treating maps in Pig for any other storage.) For more information see Schemas and Arithmetic Operators and More.

You can provide multiple, comma-separated files to the loader:

```
A = LOAD 'studenttab, votertab' USING
org.apache.hadoop.zebra.pig.TableLoader();
```

TableLoader supports efficient column selection; projections are automatically pushed down to the loader. This example tells the loader to only return two columns, name and age.

```
A = LOAD 'studenttab' USING org.apache.hadoop.zebra.pig.TableLoader('name,
age');
```

2.2. Map-Side Group and Merge Join

If the input data is globally sorted, merge join and map-side group can be used. Please note the “sorted” argument that is passed to the loader. This lets the loader know that the data is expected to be globally sorted and that a single key must be given to the same map.

Here is an example of the merge join. Note that the first argument to the loader is left empty to indicate that all columns are requested.

```
A = LOAD'studentsortedtab' USING
org.apache.hadoop.zebra.pig.TableLoader('', 'sorted');
B = LOAD 'votersortedtab' USING org.apache.hadoop.zebra.pig.TableLoader('',
'sorted');
G = JOIN A BY $0, B By $0 USING "merge";
```

Here is an example of a map-side group. Note that multiple sorted files are passed to the loader and that the loader will perform sort-preserving merge to make sure that the data is globally sorted.

```
A = LOAD 'studentsortedtab, studentnullsortedtab' using
org.apache.hadoop.zebra.pig.TableLoader('name, age, gpa, source_table',
'sorted');
B = GROUP A BY $0 USING "collected";
C = FOREACH B GENERATE group, MAX(a.$1);
```

2.3. Sorting Data

Pig allows you to sort data by ascending (ASC) or descending (DESC) order (for more information, see ORDER). Currently, Zebra supports tables that are sorted in ascending order. Zebra does not support tables that are sorted in descending order; if Zebra encounters a table to be stored that is sorted in descending order, Zebra will issue a warning and store the table as an unsorted table.

2.4. Storing Data

To store data in Zebra format use the TableStorer function. Since Zebra does not support anonymous columns, the relation that is stored must have a name (alias) assigned to every column in the relation.

```
A = LOAD 'studentsortedtab, studentnullsortedtab' USING
org.apache.hadoop.zebra.pig.TableLoader('name, age, gpa, source_table',
'sorted');
```

```
B = GROUP A BY $0 USING "collected";
C = FOREACH B GENERATE group, MAX(a.$1) AS max_val;
STORE C INTO 'output' USING org.apache.hadoop.zebra.pig.TableStorer('');
```

2.5. Simple Types

```
-- The following line is required if the Zebra jar is not part of Hadoop
Core

REGISTER $LOCATION/zebra-$version.jar;

-- Load an existing table (a projection string is not specified so all the
columns will be loaded)

A = LOAD '$PATH/tbl1' USING org.apache.hadoop.pig.zebra.pig.TableLoader();

-- Store in table format with storage hint '[c1];[c2]' which mean two
Column Groups with one column each (note that you do not specify a schema)

STORE A INTO '$PATH/tbl2' USING
org.apache.hadoop.zebra.pig.TableStorer('[c1] COMPRESS BY gz; SERIALIZE BY
pig;[c2]');

-- Load an existing table (two columns are projected)

B = LOAD '$PATH/tbl3' USING
org.apache.hadoop.zebra.pig.TableLoader('c1,c2');

-- Load an existing table (one column is projected)

C = LOAD '$PATH/tbl4' USING org.apache.hadoop.zebra.pig.TableLoader('c2');
```

2.6. Complex Types

```
-- The following line is required if the Zebra jar is not part of Hadoop
Core

register $LOCATION/zebra-$version.jar;

-- STR_SCHEMA =
-- "s1:bool, s2:int, s3:long, s4:float, s5:string, s6:bytes,
--  r1:record(f1:int, f2:long), r2:record(r3:record(f3:float, f4)),
--  m1:map(string),m2:map(map(int)),
--  c:collection (r1:record(f13:double, f14:float, f15:bytes))";

-- STR_STORAGE =
-- "[s1, s2]; [m1#{a}]; [r1.f1]; [s3, s4, r2.r3.f3];
--  [s5, s6, m2#{x|y}]; [r1.f2, m1#{b}]; [r2.r3.f4, m2#{z}]";


-- Load an existing table (two columns are projected)
```

```
A = LOAD '$PATH/tbl1' USING org.apache.hadoop.zebra.pig.TableLoader('r1.f2,
s1');

-- Store in table format (all columns are stored in one column group)

STORE A INTO '$PATH/tbl2' USING
org.apache.hadoop.zebra.pig.TableStorer('');

-- Load an existing table (two columns are projected)

B = LOAD '$PATH/tbl3' USING org.apache.hadoop.zebra.pig.TableLoader('s1,
r1');
```

2.7. HDFS File Globs

Pig supports HDFS file globs (for more information see GlobStatus).

In this example, all Zebra tables in the directory of /path/to/PIG/tables will be loaded as a union (table union).

```
 A = LOAD ‘/path/to/PIG/tables/*’ USING
org.apache.hadoop.zebra.pig.TableLoader(‘’);
```

In this example, three Zebra tables of t1, t2 and t3 in /path/to/PIG/tables will be loaded as a union (table union). Note that the ordering of the three tables in the union may not necessarily be t1 followed by t2 followed by t3 as you would expect if you specified ‘/path/to/PIG/tables/t1, /path/to/PIG/tables/t2, /path/to/PIG/tables/t3’. Instead, the ordering is determined by the ordering the HDFS glob expansion generates, namely, the *string ordering* of the expanded paths.

```
 A = LOAD ‘/path/to/PIG/tables/{t1, t2, t3}’ USING
org.apache.hadoop.zebra.pig.TableLoader(‘’);
```