Zebra and Streaming

Table of contents

1. Overview

Streaming allows you to write application logic in any langugage and to process large amounts of data using the Hadoop framework. Streaming, which traditionally works with text files, can now be used to process data stored as Zebra tables.

2. Configuration Variables

To use Zebra tables with your streaming applications, used the `mapred.lib.table.input.projection` variable to specify Zebra columns (fields).

```
bin/hadoop jar $streamingJar -D mapred.lib.table.input.projection="word,
count"
```

3. Zebra Streaming Examples

In the following examples, TableInputFormat is used for the inputclass and the default TextOutputFormat is used for the outputclass.

3.1. Creating a Zebra Table

Suppose a data file, testfile, contains four fields.

```
en bbb1 1 1880
en bbb2 1 2000
```

You can use a simple Pig script to create a Zebra table, testfile-table. The table consists of one column group with four columns.

```
$ cat table-creator.pig

REGISTER $LOCATION/zebra-$version.jar;

testfile = LOAD 'testfile'
       USING PigStorage(' ') AS (language:chararray, page:chararray,
count:int, size:long);

STORE testfile INTO 'testfile-table'
       USING org.apache.hadoop.zebra.pig.TableStorer('[language, page,
count, size]');
```

3.2. Checking Serialization

This example is a map-only job that checks the serializtion. Note that each line starts with a tab since the key is an empty string for tables created by PIG (this changes with sorted tables).

```
$ bin/hadoop jar hadoop-0.20.2-dev-streaming.jar -D mapred.reduce.tasks=0 \
        -input testfile-table -output output -mapper 'cat' \
        -inputformat org.apache.hadoop.zebra.mapred.TableInputFormat

$ grep 'en' output/part-00000 | head

(en,bbb1,1,1880)
(en,bbb2,1,2000)
(en,bbb3,1,1950)
(en,bbb4,1,48900
```

3.3. Locating Frequently Visited Pages

This Perl script sorts the pages on number of page view counts. The script outputs space padded count so that string sorting results in correct output. The first TAB separates the key and value for Hadoop streaming.

```
while (<>) {

    chomp;

    s/.?\t(.*)$/$1/ or next;  # ignore the key (if any) and remove braces

    split ','; #comma seperated list.

    # key is space padded 3rd column.

    printf("%8d\t%s\n", $_[2], "@_") if @_ == 4; # without a projection

    # printf("%8d\t%s\n", shift @_, join(',', @_)); # with
projection="count, page"
}
```

Streaming command:

```
$ bin/hadoop jar hadoop-0.20.2-dev-streaming.jar
        -input testfile-table -output output -mapper table-mapper.pl
-reducer cat \
        -inputformat org.apache.hadoop.zebra.mapred.TableInputFormat
```

Pages are printed in increasing order of page view counts.

```
$ tail output/part-00000
     10      fr bbb1 10 5883
     14      de bbb2 14 2120
     20      it bbb3 20 229
     45      ja bbb4 45 75
     47      de bbb5 47 43488
     63      en bbb6 63 2404
     73       de bbb7 73 1090
    129       en bbb8 129 31
    188       en bbb9 188 37
    222       en bbb10 222 469
```

3.4. Projecting Columns

Use projection to view only a few columns (fields) of a very large table. Modify the output line in the table-mapper.pl script as shown below and run the following streaming command:

```

$ bin/hadoop jar hadoop-0.20.2-dev-streaming.jar -D
mapred.lib.table.input.projection="count,page" \
        -input testfile-table -output output -mapper table-mapper.pl
-reducer cat \
        -inputformat org.apache.hadoop.zebra.mapred.TableInputFormat

$ tail output/part-00000
     10       bbb1
     14       bbb2
     20       bbb3
     45       bbb4
     47       bbb5
     63       bbb6
     73       bbb7
    129       bbb8
    188       bbb9
    222       bbb10
```