Hadoop Scripting with Jaql & Pig

Konstantin Haase und Johan Uhle

Outline

- Introduction
- Markov Chain
- Jaql
- Pig
- Testing Scenario
- Conclusion
- Sources

Introduction

Goal:

Compare two high level scripting languages for Hadoop

Contestants:

- Pig
- Jaql
- Hive

Markov Chain

"Random" Text Generator

Scan Text for Succeeding Word Tuple

```
Apache Hadoop is a free Java software framework that

Apache Hadoop is -> a
Hadoop is a -> free
is a free -> Java
...
```

Generate a new Text

Markov Chain

Generated Senseless Text

Can anyone think of myself as a third sex. Yes, I am expected to have. People often get used to me knowing these things and then a cover is placed over all of them. Along the side of the $$ are spent by (or at least for) the girls. You can't settle the issue. It seems I've forgotten what it is, but I don't. I know about violence against women, and I really doubt they will ever join together into a large number of jokes. It showed Adam, just after being created. He has a modem and an autodial routine. He calls my number 1440 times a day. So I will conclude by saying that I can well understand that she might soon have the time, it makes sense, again, to get the gist of my argument, I was in that (though it's a Republican administration).

(by Mark V Shaney on http://en.wikipedia.org/wiki/Mark_V_Shaney)

Markov Chain

How to implement?

- Read
- **Transform**
- SQL
- Database
- Ruby Generator

Source ≈ Wikipedia

Markov Chain

Outline

- Introduction
- Markov Chain
- **Jaql**
- Pig
- Testing Scenario
- Conclusion
- Sources

Jaql

Based on JavaScript Object Notation
Superset of JSON, subset of YAML and JavaScript

{ *hash: ["with", "an", "array", 42]* }

Inspired by Unix Pipes

read -> transform -> count -> write;

Auto-Optimise Plan

Jaql

Different Input/Output Sources: hdfs, hbase, local, http

Cluster Usage depends on Input Source

Interactive Shell or prepacked JAR

Not turing complete

Jaql

Apache License 2.0
Developed mainly by IBM

Lack of documentation
Mailing list is dead

Latest Release missed lots of features
Used SVN head

Jaql

```
registerFunction("strSplit", "com.acme.extensions.expr.SplitIterExpr");

$markovEntry = fn($words, $pos) (
    $words -> slice($pos - 3, $pos)
           -> transform serialize($)
           -> strJoin(", "));

$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
[
   "Apache Hadoop is a free Java software framework that ...",
   ...
]

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

"Apache Hadoop is a free Java software framework that supports data intensive distributed applications..."

```
$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                   $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
["Apache", "Hadoop", "is", "a", "free", "Java", ...]

$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

[3, 4, 5, ...]

```
$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
$markovEntry = fn($words, $pos) (
    $words -> slice($pos - 3, $pos)
            -> transform serialize($)
            -> strJoin(", "));

$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                   $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

["Apache", "Hadoop", "is", "a", ...]

```
$markovEntry = fn($words, $pos) (
    $words -> slice($pos - 3, $pos)
           -> transform serialize($)
           -> strJoin(", "));

$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
["\"Apache\"", "\"Hadoop\"", "\"is\"", "\"a\""]

$markovEntry = fn($words, $pos) (
    $words -> slice($pos - 3, $pos)
           -> transform serialize($)
           -> strJoin(", "));

$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
"\"Apache\", \"Hadoop\", \"is\", \"a\""

$markovEntry = fn($words, $pos) (
    $words -> slice($pos - 3, $pos)
           -> transform serialize($)
           -> strJoin(", "));

$markovLine = fn($line) (
    $words = $line -> strSplit(" "),
    range(3, count($words) - 1) -> transform $markovEntry($words, $));

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
$w = "\"Apache\", \"Hadoop\", \"is\", \"a\""
$ = ["\"Apache\", \"Hadoop\", \"is\", \"a\""]

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                 $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
"INSERT INTO jaqltest.splitsuc VALUES
 (\"Apache\", \"Hadoop\", \"is\", \"a\", 1);"

$markov = fn($lines) (
  $lines
    -> expand each $line $markovLine($line)
    -> group by $w = ($) into "INSERT INTO jaqltest.splitsuc VALUES (" +
                    $w + ", " + serialize(count($)) + ");");

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Jaql

```
"INSERT INTO jaqltest.splitsuc VALUES
 (\"Apache\", \"Hadoop\", \"is\", \"a\", 1);"

read(file("input.dat")) -> $markov() -> write(file("output.dat"));
```

Pig

Hadoop Subproject in Apache Incubator since 2007

Mainly developed by Yahoo

Active Mailinglist and average documentation

High-level language for data transformation

Apache License

Pig

Key Aspects:

- Ease of Programming
- Optimisation Opportunities
- Extensibility

Similar to SQL but data flow programming language

Similar to the DBMS query planner

Auto Optimisation
Plan is also visible

Step by Step set of operations on an input relation, in which each step is a single transformation.

Pig

Text: chararray

Numeric: int, long, float, double

Tuple: *(1, 'element2', 300L)*

sequence of fields of any type

Bag: { *(1),(1, 'element2')* }

unordered collection of tuples

Pig

```
register splitsuc.jar;

wikipedia = LOAD 'corpera' USING PigStorage() AS (row:chararray);
tuples = FOREACH wikipedia GENERATE FLATTEN(splitsuc.SPLITSUC());
grouped = GROUP tuples by (keyTuple, successorTuple);
grouped_counted = FOREACH grouped GENERATE group, COUNT(tuples);
STORE final INTO 'wikipedia.sql' USING splitsuc.STORESQL();
```

Pig

```
register splitsuc.jar;

wikipedia = LOAD 'corpera' USING PigStorage() AS (row:chararray);
```

...The capital of Algeria is Algiers ...

wikipedia: row1, row2, row3, ...

Pig

```
register splitsuc.jar;

wikipedia = LOAD 'corpera' USING PigStorage() AS (row:chararray);
tuples = FOREACH wikipedia GENERATE FLATTEN(splitsuc.SPLITSUC());
```

(The,capital,of) (Algeria)
(capital,of,Algeria) (is)
(of,Algeria,is) (Algiers.)

```
tuples: ((word1,word2,word3),(successor))
        (       keyTuple,     successorTuple)
```

Pig

```
register splitsuc.jar;

wikipedia = LOAD 'corpera' USING PigStorage() AS (row:chararray);
tuples = FOREACH wikipedia GENERATE FLATTEN(splitsuc.SPLITSUC());
grouped = GROUP tuples by (keyTuple, successorTuple);
```

((of,Algeria,is),(Algiers.)) {((of,Algeria,is),(Algiers.)), ((of,Algeria,is),(Algiers.))}

```
grouped: {(keyTuple, successorTuple), { (keyTuple, successorTuple) , ... } }
         {         group ,             {           group ,               ... } }
```

Pig

```
register splitsuc.jar;

wikipedia = LOAD 'corpera' USING PigStorage() AS (row:chararray);
tuples = FOREACH wikipedia GENERATE FLATTEN(splitsuc.SPLITSUC());
grouped = GROUP tuples by (keyTuple, successorTuple);
grouped_counted = FOREACH grouped GENERATE group, COUNT(tuples);
```

((of,Algeria,is),(Algiers.)) 2

grouped_counted: (keyTuple, successorTuple), Count

Pig

```
register splitsuc.jar;

wikipedia = LOAD 'corpera' USING PigStorage() AS (row:chararray);
tuples = FOREACH wikipedia GENERATE FLATTEN(splitsuc.SPLITSUC());
grouped = GROUP tuples by (keyTuple, successorTuple);
grouped_counted = FOREACH grouped GENERATE group, COUNT(tuples);
STORE final INTO 'wikipedia.sql' USING splitsuc.STORESQL();

INSERT INTO table VALUES
...
('of', 'Algeria', 'is', 'Algiers.', '2'),
...
```

Show Output to File and Output to MySQL

Outline

- Introduction
- Markov Chain
- Jaql
- Pig
- **Testing Scenario**
- Conclusion
- Sources

Test Scenario

Both Need Hadoop 0.18
Will Run On Cluster Week After Exams

Building Markov Chain Index Of Four with JAQL and PIG

Benchmarking
Minimising error by multiple tests per setup
Input: Wikipedia Corpus Size 1 GB, 10 GB, 20 GB

Generating Random Text with a Ruby Script from DB

Conclusion

Rapid Development compared to native Java Hadoop

Less thinking in Map/Reduce tasks

Both offer tight Java integration

Pig	Jaql
declarative	functional / procedural
evolving ecosystem	small ecosystem

Conclusion

Both still immature but promising

- Bad Documentation
- Bad Debugging
- Lack of Built-In Functionality
- Lack of Ecosystem

No Community

„Inhouse Development“

We are looking forward to cluster tests

Sources

http://en.wikipedia.org/wiki/Markov_chain
http://www.jaql.org/
http://code.google.com/p/jaql/
http://hadoop.apache.org/pig/

"Hadoop: The Definitive Guide" by Tom White
O'Reilly Media, Inc. 2009

End

- Introduction
- Markov Chain
- Jaql
- Pig
- Testing Scenario
- Conclusion
- Sources