Introduction To Hive

How to use Hive in Amazon EC2

CS 341: Project in Mining Massive Data Sets
Hyung Jin(Evion) Kim
Stanford University

References:
Cloudera Tutorials,
CS345a session slides,
“Hadoop - The Definitive Guide”
Roshan Sumbaly, LinkedIn
Todays Session

• Framework: Hadoop/Hive
• Computing Power: Amazon Web Service
• Demo
• LinkedIn’s frameworks & project ideas
Hadoop

- Collection of related sub projects for distributed computing
- Open source
- Core, Avro, **MapReduce**, HDFS, Pig, HBase, ZooKeeper, **Hive**, Chukwa ...
Hive

• Data warehousing tool on top of Hadoop
• Built at Facebook
• 3 Parts
  • Metastore over Hadoop
  • Libraries for (De)Serialization
  • Query Engine(HQL)
AWS - Amazon Web Service

- S3 - Data Storage
- EC2 - Computing Power
- Elastic Map Reduce
Step by step

• Prepare Security Keys
• Upload your input files to S3
• Turn on elastic Map-Reduce job flow
• Log in to job flow
• HiveQL with custom mapper/reducer
0. Prepare Security Key

- AWS: Access Key / Private Key

- EC2: Key Pair - Key name and Key file (.pem)
1. Upload files to S3

- Data stored in buckets (folders)
- This is your only permanent storage in AWS - save input, output here
- Use Firefox Add-on S3Fox Organizer (http://www.s3fox.net)
2. Turn Elastic MapReduce On
3. Connect to Job Flow (I)

- Using Amazon Elastic MapReduce Client
- Need Ruby installed on your computer
3. Connect to Job flow (2) - security

- Place credentials.json and .pem file in Amazon Elastic MapReduce Client folder, to avoid type things again and again

- 
  
  ```json
  {
    "access-id": "<access-id>",
    "private-key": "<private-key>",
    "key-pair": "new-key",
    "key-pair-file": ".\new-key.pem",
    "region": "us-west-1"
  }
  ```
3. Connect to Job Flow

(3)

- list jobflows:
  elastic-mapreduce --list

- terminate job flow:
  elastic-mapreduce --terminate --jobflow <id>

- SSH to master:
  elastic-mapreduce --ssh <id>
4. HiveQL(1)

- SQL like language
- Hive WIKI
  http://wiki.apache.org/hadoop/Hive/GettingStarted
- Cloudera Hive Tutorial
  http://www.cloudera.com/hadoop-training-hiveintroduction
4. HiveQL (2)

- SQL like Queries
- SHOW TABLES, DESCRIBE, DROP TABLE
- CREATE TABLE, ALTER TABLE
- SELECT, INSERT
4. HiveQL(3) - usage

- Create a schema around data: CREATE EXTERNAL TABLE
- Use like regular SQL: Hive automatically change SQL query to map/reduce
- Use with custom mapper/reducer: Any executable program with stdin/stdout.
Example - problem

- Basic map reduce example - count frequencies of each word!

  ‘I’ - 3
  ‘data’ - 2
  ‘mining’ - 2
  ‘awesome’ - 1

  ...
Example - Input

- Input: 270 twitter tweets
- sample_tweets.txt

T 2009-06-08 21:49:37
U http://twitter.com/evion
W I think data mining is awesome!

T 2009-06-08 21:49:37
U http://twitter.com/hyungjin
W I don’t think so. I don’t like data mining
Example - How?

• **Create table from raw data file**
  
  table raw_tweets

• **Parse data file to match our format, and save to new table**
  
  parser.py
  
  table tweets_test_parsed

• **Run map/reduce**
  
  mappr.py, reducer.py

• **Save result to new table**
  
  table word_count

• **Find top 10 most frequent words from word_count table.**
Example - Create Input Table

Create Schema around raw data file

```
CREATE EXTERNAL TABLE raw_tweets(line string)
ROW FORMAT DELIMITED
LOCATION 's3://cs341/test-tweets';
```

With this command, ‘\t’ will be separator among columns, and ‘\n’ will be separator among rows.
Example - Create Output Table

CREATE EXTERNAL TABLE tweets_parsed
(time string, id string, tweet string)
ROW FORMAT DELIMITED
LOCATION 's3://cs341/tweets_parsed';

CREATE EXTERNAL TABLE word_count
(word string, count int)
ROW FORMAT DELIMITED
LOCATION 's3://cs341/word_count';
Example -
TRANSFORM

TRANSFORM - given python script will transform the input columns
Let’s parse original file to <time>, <id>, <tweet>

ADD FILE parser.py;

INSERT OVERWRITE TABLE tweets_parsed
SELECT TRANSFORM(line)
USING 'python parser.py' AS (time, id, tweet)
FROM raw_tweets;

Add whatever the script file you want to use to hive first.

Write out result of this select to tweets_parsed table
Example - Map/Reduce

Use command MAP and REDUCE: Basically, same as TRANSFORM
tweets_parsed -> map_output -> word_count

ADD FILE mapper.py;
ADD FILE reducer.py;

FROM (  
    FROM tweets_parsed  
    MAP tweets_parsed.time, tweets_parsed.id, tweets_parsed.tweet  
    USING 'python mapper.py'  
    AS word, count  
    CLUSTER BY word) map_output  
INSERT OVERWRITE TABLE word_count  
    REDUCE map_output.word, map_output.count  
    USING 'python reducer.py'  
    AS word, count;

Use word as key
Example - Finding Top 10 Words

Using similar syntax as SQL

```
SELECT word, count FROM word_count
ORDER BY count DESC limit 10;
```
Example -JOIN

Finding pairs of words that have same count, and count bigger than 5

```
SELECT wc1.word, wc2.word, wc2.count
FROM word_count wc1 JOIN word_count wc2
ON(wc1.count = wc2.count)
WHERE wc1.count > 5  AND wc1.word < wc2.word;
```
Frameworks from LinkedIn

- Complete “data stack” from LinkedIn open source @ http://sna-projects.com
- Any questions - rsumbaly@linkedin.com
- Introduce “Kafka” and “Azkaban” today.
Kafka

- Distributed publish/subscribe system
- Used at LinkedIn for tracking activity events
Kafka(2)

- Parsing data in files every time you want to run an algorithm is tedious.

- What would be ideal? An iterator over your data (hiding all the underneath semantics).

- Kafka helps you publish data once (or continuously) to this system and then consume it as a “stream”.

Kafka(3)

- Example: Easy for implementing stream algorithms on top of Twitter stream
Azkaban(1)

- A simple Hadoop workflow system
- Used at LinkedIn to generate workflows for recommendation features
- Last year many students wanted to iterate on their algorithms multiple times. This required them to build a chain of Hadoop jobs which they ran manually every day.
Azkaban(2)

• Example workflow

• Generate n-grams as a Java program
  -> Feed n-grams to MR Algorithms $X$ run on Hadoop
  -> Fork n parallel MR jobs to feed this to
    Algorithm $X_1$ to $X_n$
  -> Compare the results at the end

• [http://sna-projects.com/azkaban](http://sna-projects.com/azkaban)