BIG DATA

Facebook Social Network Data Processing using FacePager to Hive

Instructor: Dr. Sunnie Chung

Presented by: Abhay Raval
CSU ID : 2636049
Outline

• Data Source
• Tools for Processing Data
• Data Processing System: Hive
• Data Transformation: Hive
• Data Mining Queries
• References
• Research Paper
Data Source

• Downloaded data from Facebook Graph Api

• Approach:
  • RestFb Api
    • Not able to transform data into proper format.

  • Python script
    • Using FQL (Facebook Query Language)
    • FQL is not supported for Graph API versions 2.0 or above.

  • FacePager Application
    • Tool to fetch Facebook data
    • Received authorize ‘Access Token’ through own Facebook application.
FacePager

- Fetching public available data from Facebook, Twitter and JSON based APIs.
- Downloaded from http://www.ls1.ifkw.uni-muenchen.de/personen/wiss_ma/keyling_till/software.html
FacePager
FacePager
FacePager

<table>
<thead>
<tr>
<th>Object ID</th>
<th>Object Type</th>
<th>Query Status</th>
<th>Query Time</th>
<th>C^</th>
</tr>
</thead>
<tbody>
<tr>
<td>thisinsider</td>
<td>seed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
<tr>
<td>1413475...</td>
<td>data</td>
<td>fetched (200)</td>
<td>2015-12-01 23:34:...</td>
<td>F</td>
</tr>
</tbody>
</table>

### Key
- **picture**: https://scontent.xx.fbcdn.net/hvthumb-xp1/v/t15.0-10/...
- **from**: [3]
- **id**: 141347369659824_1403622557611794
- **object_id**: 1403622557611794
- **privacy**: [5]
- **type**: video
- **message**: Tasmania is the perfect place to see the stars without ...
- **comments**: [2]
- **shares**: [1]
- **source**: https://video.xx.fbcdn.net/hvideo-xfa/v/142.1790-3/1230...
- **link**: https://www.facebook.com/thisisvideo/videos/1493622...

### Custom Table Columns (one key per line)
- **name**
- **message**
- **created_time**
- **updated_time**

### Settings
- **Node level**: 2
- **Object types**: a,unpacked
- **Parallel Threads**: 1
- **Requests per minute**: 50000

### Status Log
- [Log entries]
FacePager

<table>
<thead>
<tr>
<th>Object ID</th>
<th>Object Type</th>
<th>Query Status</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
<tr>
<td>1413475_1</td>
<td>data</td>
<td>fetched (20)</td>
<td>2015-12-01 23:33:58</td>
</tr>
</tbody>
</table>

Custom Table Columns (one key per line)

- name
- message
- created_time
- updated_time

Settings

- Node level: 1
- Object types: b.unpacked
- Parallel Threads: 1
- Requests per minute: 00:0300

Status Log

Fetch Data

Login to Facebook
Possible Tools for Processing Data

- Hive
- Spark
- Cassandra
Data Processing System: Hive

- Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage.
- Built on top of Apache Hadoop.
- Provides query language like SQL, called HiveQL.
- Provides easy data extract/transform/load (ETL).
Data Transformation: Hive
Data Transformation: Hive

<table>
<thead>
<tr>
<th>Permission</th>
<th>Owner</th>
<th>Group</th>
<th>Size</th>
<th>Last Modified</th>
<th>Replication</th>
<th>Block Size</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rw-r--r-</td>
<td>abhay</td>
<td>supergroup</td>
<td>1.09 MB</td>
<td>12/2/2015, 12:28:36 AM</td>
<td>1</td>
<td>128 MB</td>
<td>comments.csv</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>abhay</td>
<td>supergroup</td>
<td>174.69 KB</td>
<td>12/2/2015, 12:53:09 AM</td>
<td>1</td>
<td>128 MB</td>
<td>fb_post.csv</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>abhay</td>
<td>supergroup</td>
<td>2.77 MB</td>
<td>12/2/2015, 12:28:19 AM</td>
<td>1</td>
<td>128 MB</td>
<td>likes.csv</td>
</tr>
</tbody>
</table>

Hadoop, 2015.
Data Transformation: Hive

- Create table to store page posts data.

```
hive> create table fbpost(id String,parentId String,level String,objectId String,objectType String,queryStatus String,queryTime String,queryType String,name String,message String,createdTime String,updatedTime String) row format delimited fields terminated by',' stored as textfile;
OK
Time taken: 2.522 seconds
hive>
```

- Load fb_post.csv data into table.

```
hive> LOAD DATA INPATH '/user/abhay/facebook/fb_post.csv' OVERWRITE INTO TABLE fb_post;
Loading data to table default.fb_post
Table default.fb_post stats: [numFiles=1, numRows=0, totalSize=198201, rawDataSize=0]
OK
Time taken: 1.207 seconds
hive>
```
Data Transformation: Hive

- Select * from fb_post;
Data Transformation: Hive

• Create table to store likes data.

```
hive> create table likes(id String, parentid String, level String, objectid String, objecttype String, querystatus String, querytime String, querytype String, name String, message String, createdtime String, updatedtime String) row format delimited fields terminated by ',' stored as textfile;
```

• Load likes.csv data into table.

```
hive> LOAD DATA INPATH '/user/abhay/facebook/likes.csv' OVERWRITE INTO TABLE likes;
```
Data Transformation: Hive

- Select * from likes;
Data Transformation: Hive

- Create table to store comments data and load data into it.

```
hive> create table comments(id String, parentid String, level String, objectid String, objecttype String, querystatus String, querytime String, querytype String, name String, message String, createdtime String, updatedtime String) row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 0.57 seconds
hive> LOAD DATA INPATH '/user/abhay/facebook/comments.csv' OVERWRITE INTO TABLE comments;
Loading data to table default.comments
Table default.comments stats: [numFiles=1, numRows=9, totalSize=1145394, rawDataSize=0]
OK
Time taken: 0.399 seconds
hive> 
```
Data Transformation: Hive

- Filtering posts messages from fb_post table.
Data Transformation: Hive

- Create table to store filtered posts data

```bash
abhay@abhay:~/Documents/hadoop-2.7.1S /bin/hdfs dfs -put /home/abhay/Music/output/postmessages.csv /user/abhay/facebook

hive> create table postmessages(messages String) row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 0.116 seconds
```
Data Transformation: Hive

• Select * from postmessages;
Data Transformation: Hive

- Temporary table (wordcount) created to store words from postmessages table and run python script to perform wordcount.

```
hive> create table wordcount(words String);
```

- Transform messages data by adding wordcount python script.

```
hive> insert overwrite table wordcount select transform(messages) using 'python /home/abhay/wordcount.py' as word from postmessages;
```

```
Query ID = abhay_2015120213221_1f62328b-5716-4470-9567-97cc56a5673b
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2015-12-02 01:52:23,774 Stage-1 map = 100%, reduce = 0%
Ended Job = job_local.24710192.0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/user/hive/warehouse/wordcount/hive-staging_hive_2015-12-02_01-32-21_444_371014980730691017-1/-ext-10000
Loading data to table default.wordcount
Table default.wordcount stats: [numFiles=1, numRows=7709, totalSize=44995, rowDataSize=37280]
MapReduce Jobs Launched:
Stage-Stage-1: HDFS Read: 90091 HDFS Write: 45873 SUCCESS
Total MapReduce CPU Time Spent: 0 Msec
Ok
Time taken: 2.768 seconds
hive>
```
Data Transformation: Hive

• Wordcount query

```
hive> select words, count(*) as count from wordcount group by words;
```

• Result:
Data Transformation: Hive

• Wordcount query

```sql
hive> select words, count(*) as count from wordcount group by words;
```

• Result:
Data Transformation: Hive

• Structure of data:
Data Transformation: Hive

• Filtering names and parent id from likes
Data Transformation: Hive

- Created new pidlikes table and inserted filtered data.

```
abhay@abhay:~/Documents/hadoop-2.7.1S bin/hdfs dfs -put /home/abhay/Music/pidlikes.csv /user/abhay/facebook
abhay@abhay:~/Documents/hadoop-2.7.1S hive

Logging initialized using configuration in jar:file:/usr/lib/hive/apache-hive-1.2.1-bin/lib/hive-common-1.2.1.jar!/hive-log4j.properties
hive> create table pidlikes(pid String,name String) row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 1.936 seconds
hive> LOAD DATA INPATH '/user/abhay/facebook/pidlikes.csv' OVERWRITE INTO TABLE pidlikes;
Loading data to table default.pidlikes
Table default.pidlikes stats: [numFiles=1, numRows=0, totalSize=370640, rawDataSize=0]
OK
Time taken: 0.9 seconds
hive>
```
Data Transformation: Hive

• Pidlikes table data:
Data Transformation: Hive

- Created new pidcomments table and inserted filtered data
Data Transformation: Hive

• Pidcomments table data:
Data Transformation: Hive

- Performing joins between `fb_post` and `pidlikes` tables which gives postid, totallikes of that post and postmessage.

```sql
hive> select f1.id, count(l1.name), f1.message from fb_post f1 right join pidlikes l1 on f1.id = l1.pid group by l1.pid, f1.id, f1.message;
```
Data Transformation: Hive

- Performing joins between `fb_post` and `pidcomments` tables which gives `postid`, total comments of that post and `postmessage`.
Data Mining Queries

• Find the posts which are liked by the persons with the names starts with David

```
hive> select f1.id, count(c1.name), f1.message from fb_post f1, pidlikes c1 where f1.id = c1.pid and c1.name like 'David%' group by c1.pid, f1.id, f1.message;
```

Query ID = abhay_2015120221531_b704eea5-96a0-4fbd-afe5-591755737470
Total jobs = 1
Data Mining Queries

• Find the posts which are liked by the persons with the name starts with David
Data Mining Queries

- Find the posts which are liked by the person with name ‘David Geary’
Data Mining Queries

• Find the top 5 posts with minimum likes.

```sql
hive> select f1.id, count(c1.name) as likes, f1.message from fb_post f1 right join pidlikes c1 on f1.id = c1.pid group by c1.pid, f1.id, f1.message
order by likes limit 5;
Query ID = abhay_2015120221433_32ef6907-d0ff-4f91-9360-08419345fc7b
Total jobs = 2.
```

```
Stage-Stage-3: HDFS Read: 30784616 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
163 1 Sticks have been used as weapons for thousands of years. So why not selfle sticks?
158 4 One of the biggest waves you can surf.
160 6 Zlon is the youngest person ever to receive a hand transplant.
161 16 The USS Harry Truman is a beast.
164 17 Duchess Kate spoke to a bunch of teachers about her life's work.
Time taken: 9.397 seconds, Fetched: 5 row(s)
hive>
```
Data Mining Queries

• Find the posts which have likes $\geq 50$ but $\leq 100$.  

```
hive> select f1.id, count(c1.name) as likes, f1.message from fb_post f1 right join pidlikes c1 on f1.id = c1.pid group by c1.pid, f1.id, f1.message having (likes <= 100 and likes >= 50);
Query ID = abhay_20151202233525_7a424e5e-de3c-4bb6-b47b-ab98b457e003
Total jobs = 1
Total MapReduce CPU time spent: 0.00 sec
OK
101 85  Patti LaBelle spent Thanksgiving with the guy who made her Walmart pies go viral.
104 50  China banned a Canadien beauty queen because she criticized its government.
109 62  These floating drones can help you understand virtual reality (via Vocativ).
114 68  Breaking down barriers between the hearing and deaf worlds.
125 74  The Rock opened up about his childhood.
130 71  New York's cheese experts explain how to make the perfect cheese plate.
141 99  A Harvard University scientist who's studied coffee for 20 years explains why it's amazing.
159 62  This guy interviewed ... himself!
16 66  Congress just reached a deal to spend $305 billion on highways and mass-transit.
25 75  MIA's new music video takes on the refugee crisis.
3 65  Hundreds of limestone pillars.
34 52  Being evicted as a teenager motivated The Rock to become one of the most successful wrestlers...ever.
40 57  Six things that trans women want you to know (via Bustle).
41 80  "For nine years
42 62  "PSY is out with a new single. Here's why it took him so long to follow up his mega-hit ""Gangnam Style."
45 84  Amy Schumer got this giant bruise while hosting Saturday Night Live.
52 78  "IKEA USA designed these simple
80 97  Get ready for Godzilla (the weather event).
87 99  Trump booted this guy out of his rally.
91 77  Hundreds of protesters blocked Black Friday shoppers in Chicago. The boycott came days after authorities released a video of a white cop shooting and killing a black teenager.
99 72  Hundreds of protesters blocked Black Friday shoppers in Chicago. The boycott came days after authorities released a video of a white cop shooting and killing a black teenager.
```

Time taken: 8.175 seconds, Fetched: 21 row(s)
References

- http://www.ls1.ifkw.uni-muenchen.de/personen/wiss_ma/keyling_till/software.html
- https://hive.apache.org/
- https://developers.facebook.com/
- https://developers.facebook.com/tools/explorer/145634995501895/?method=GET&path=me&version=v2.5
- https://docs.treasuredata.com/articles/hive
Research Paper

Hive - A Warehousing Solution Over a Map-Reduce Framework
Facebook Data Infrastructure Team
Introduction

• In Business Intelligence era, Hadoop is a open-source map-reduce implementation to store and process large amount of data.
• Hive is an open-source data warehousing solution built on top of Hadoop.
• Hive supports HiveQL like SQL which are compiled into map-reduce jobs executed on Hadoop.
• Supports custom map-reduce scripts to be plugged into queries.
Introduction

- HiveQL includes a type system with tables and collections like arrays and maps.
- Hive includes a system catalog, Hive-Metastore, containing schemas and statistics, which is very helpful in data exploration and query optimization.
Hive Database

- Data organized into:
  - Tables:
    - Each table has HDFS directory respectively
    - Data is serialized and stored in files within directory
    - Serialization format is inbuilt and used by hive for query compilation and execution.
    - Supports external tables on data
  - Partitions:
    - Determining the distribution of data within sub-directories of respective table directory.
  - Buckets:
    - In partition directory each bucket is stored as a file.
Query Language: HiveQL

- Supports select, project, join, aggregate, union all and sub-queries.
- Also supports DDL statements with specific formats such as serialization, partitioning and bucketing columns
- Insert and load external data
- Supports User defined column transformation and aggregation functions implemented in java
- Supports custom map-reduce scripts
Example: StatusMeme

• Users updates their status, updates logged into flat files in directory which is rotated everyday. We load that data into hive table on daily basis.

• Table: status_update(userid int, Status String, ds String)

• Load statement: LOAD DATA LOCAL INPATH ‘/logs/status_updates’ INTO TABLE status_updates PARTITION (ds=’2009-03-20’)
Example: StatusMeme

- Daily statistics of status updates based on gender and school which the user attends.

```sql
FROM (SELECT a.status, b.school, b.gender FROM status_updates a JOIN profiles b ON (a.userid = b.userid and a.ds='2009-03-20' ) ) subq1
  INSERT OVERWRITE TABLE gender_summary PARTITION(ds='2009-03-20') SELECT subq1.gender, COUNT(1) GROUP BY subq1.gender
  INSERT OVERWRITE TABLE school_summary PARTITION(ds='2009-03-20') SELECT subq1.school, COUNT(1) GROUP BY subq1.school
```
Example: StatusMeme

- Top ten memes per school as determined by status updates by users who attend that school using map-reduce constructs.

```
REDUCE subq2.school, subq2.meme, subq2.cnt USING ‘top10.py’ AS (school,meme,cnt)
FROM (SELECT subq1.school, subq1.meme, COUNT(1) AS cnt FROM (MAP
b.school, a.status USING ‘meme-extractor.py’ AS (school,meme) FROM
status_updates a JOIN profiles b ON (a.userid = b.userid) ) subq1 GROUP BY
subq1.school, subq1.meme DISTRIBUTUE BY school, meme SORT BY school,
meme, cnt desc ) subq2;
```
Hive Architecture
Metastore

• System catalog which contains metadata about stored table.
• Specified during table creation and reused every time the table is referenced in HiveQL.
• Maintains consistency between metadata and data.
Metastore

• Metastore contains:
  • Database:
    • Namespace for tables
  
  • Table:
    • List of columns and their types, owner, storage and SerDe information
    • Also contain user supplied key and value.
    • Provided during the creation of table
  
  • Partition:
    • Own column and SerDe & storage information.
Compiler

- DML, DDL or query statement as input and converts string into plan.
- Plan consists of metadata operations for load, DDL and HDFS statement.
- For Insert and queries it generates DAG of map-reduce job.
- Parser transforms query to parse tree representation.
- Semantic analyzer verifies and type-checking.
- Logical plan generator transform into logical plan representation.
- Optimizer rewrites the query and choose the best logical plan.
- Physical plan generator converts logical plan into a physical plan consisting of DAG map-reduce jobs.
Demonstration Description

• Functionality
  • HiveQL construction using Statusmeme example.

• Tuning
  • Query plan viewer in which queries translated into physical plans to map-reduce jobs

• User Interface
  • Explore Hive database, HiveQL, queries.

• Scalability
  • By increasing size of data and the complexity of queries.
Future Work

• Currently accepts only subset SQL as valid query. Try to make HiveQL subsume SQL syntax
• Build Cost based optimizer and adaptive optimization techniques to come up with more efficient plans
• Improve scan performance
• Exploring multi query optimizer techniques and performing joins in a single map-reduce job.
Thank you.