Building Recommendation Platform with Hadoop

Jayant Shekhar
Agenda

• Why Big Data Recommendation Platform?
• Architecture
• Design & Implementation
• Common Recommendation Patterns
Recommendations is one of the commonly used use cases of Hadoop

<table>
<thead>
<tr>
<th>Recommendations Broader Use Cases</th>
<th>Recommendations can be</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Product Recommendation</td>
<td>• Realtime</td>
</tr>
<tr>
<td>• People/Social Recommendation</td>
<td>• News Recommendations</td>
</tr>
<tr>
<td>• Merchant Recommendation</td>
<td>• Merchant/Offer</td>
</tr>
<tr>
<td>• Content Recommendation</td>
<td>• Recommendations on</td>
</tr>
<tr>
<td>• Query Recommendation</td>
<td>• mobile</td>
</tr>
<tr>
<td>• Sponsored Search Advertising</td>
<td>• Offline</td>
</tr>
<tr>
<td></td>
<td>• Similar Profiles/Resumes</td>
</tr>
<tr>
<td></td>
<td>• In Between</td>
</tr>
</tbody>
</table>

**Recommendation Patterns**

- People who viewed this also viewed
- Content/Product Recommendation
- Frequently Bought Together
- Related Searches or Query Recommendation
- Related Articles/News
- People/Social Recommendation
Recommendations are delivered through

- Web
- Mobile
- Email
- Postal mail
- Newspaper/Magazine ads

Data Sets Involved

- Items/Products/Content
- Transaction Data
- User Data
- Logs & User Activity
- Additional 3rd Party Data
  - Geo
  - Social
  - Reviews
  - …

Different Time to Action Targeted

- User would view the content now/Buy the Product Now
- User would buy the product in a week
  - Next when he/she goes grocery shopping
- User would buy the product in the next 3 months
  - TV/Dishwasher etc.
  - Vacation

Also need to be able to determine/differentiate between the users in a household
In Summary

• Users have lots of options for most things
• Their attention span is very limited
• Highly targeted Recommendations stand a much better chance of conversions/better user experience
• Larger datasets of items, user behavior, related data (from 3rd party etc.) are needed for targeting
• Recommendations Use Cases are Many, Fast Changing & Complex
• Having an agile, flexible, scalable, end-to-end Platform is the way to go
• Hence we need a Big Data Platform for Recommendations
Architecture
Components

A Hadoop based Recommendation Platform has to provide the following:

- ETL
- Feature Generation
- Recommendation Algorithms/Model Generation
- Workflows & Scheduling
- Serving Systems
- Support Testing of the Models
- Measure Performance
Feature Generation & Model Building

Hadoop enables easy iteration over the process of Model Generation and testing it out offline.
Serving Systems

Real-time Content

HBase

HDFS

Docs
Pre-calculated Results
Features
Models

Real-time Content

Recommendation Server

Solr

Web Server

Web Server

Web Server

Web Server
Components/Design
ETL/Flume

- Has concept of Source/Channel/Sink

*Flume User Guide*
agent_foo.sources = s1
agent_foo.sinks = k1
agent_foo.channels = c1

# properties of s1
agent_foo.sources.s1.type = avro
agent_foo.sources.s1.bind = localhost
agent_foo.sources.s1.port = 10000
agent_foo.sources.s1.channels = c1

# properties of c1
agent_foo.channels.c1.type = memory
agent_foo.channels.c1.capacity = 1000
agent_foo.channels.c1.transactionCapacity = 100

# properties of k1
agent_foo.sinks.k1.type = hdfs
agent_foo.sinks.k1.hdfs.path = hdfs://analytics-1.cloudera.com/user/jayant/flume1
agent_foo.sinks.k1.hdfs.rollSize = 3000000
agent_foo.sinks.k1.hdfs.rollInterval = 60
agent_foo.sinks.k1.hdfs.rollCount = 0
agent_foo.sinks.k1.hdfs.fileType = DataStream
agent_foo.sinks.k1.hdfs.filePrefix = logs-
agent_foo.sinks.k1.hdfs.writeFormat = Text
agent_foo.sinks.k1.serializer = avro_event
agent_foo.sinks.k1.channel = c1

• flume-ng agent --conf-file avro.conf --name agent_foo -Dflume.root.logger=INFO,console
• flume-ng avro-client -H localhost -p 10000 -F ./error_log
ETL

**Transform**

- Flume – Interceptors, Serializers etc.
- Map Reduce
- HIVE
- PIG
- Streaming
- Sessionization
- Field Extraction
- Match/Dedup…..

**Load**

- Data loaded onto HDFS/HIVE
- Data loaded into HBase
- Data loaded into external systems – RDBMS/No SQL
CREATE EXTERNAL TABLE apachelog (  
  remote_ip STRING,  
  request_date STRING,  
  method STRING,  
  request STRING,  
  protocol STRING  
)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'  
WITH SERDEPROPERTIES (  
  "input.regex" = "([ ] . . [[[^]]]+) \"([ ] ) ([ ] ) ([^ \"])\" *",  
  "output.format.string" = "%1$s %2$s %3$s %4$s %5$s"  
)  
STORED AS TEXTFILE  
location '/user/jayant/apachelog';

select remote_ip from apachelog;

66.249.68.6 - - [14/Jan/2012:06:25:03 -0800]  
"GET /example.com HTTP/1.1" 200 708 "-"  
"Mozilla/5.0 (compatible; Googlebot/2.1;  
+http://www.google.com/bot.html)"
add file t2.pl;

INSERT OVERWRITE TABLE processed_logs
SELECT t.x, t.y, t.z from (  
  SELECT transform( body)  
  using 't2.pl'  
  as (x, y, z) from mylogs
) t;
Feature Generation

- Generates the features needed for the Recommendation Algorithms
- Various groups can share the features / work done by others on the Platform

- Daily Query Volume
- Number of transaction for a user at a Merchant in the last X months
- Sentiment of a user for a given merchant
- Distance of a User from a Merchant
- List of merchants visited by a user in a trip
- Number of times the user viewed an item in the last week
- Items bought together
Feature Generation

- Hadoop and HBase are ideal for Building and Sharing of features offline on HDFS
- MR/HIVE/PIG/Impala/Mahout/Streaming provide extreme flexibility in writing code for Feature Generation
- *Makes it tremendously easy to share features, results, iterate and innovate etc.*
Workflows with Oozie

- Start -> Map Reduce -> PIG -> Java/Mahout
- End -> FS -> HIVE
- Decision
- Fork -> Join
- Sqoop

- oozie job -oozie http://xyz.com:11000/oozie -config job.properties -run
- Can also be triggered by time frequency and data availability
ML on Hadoop – Different users prefer different set of tools for Data Science/ML on Hadoop

Mahout  Impala/HIVE/PIG  RHIPE  Rhadoop
Model Generation

HDFS

Mahout/HIVE/PIG/R

Data

Data

Data

Data

Clustering / Collaborative Filering

Model Generation

Model

Model

Model

Model

Model
ML Tools on Hadoop

Mahout

- ML Algorithms in Java/MR
- Input as text or SequenceFiles.
- Can write your own embedded Driver Programs
- Can control the output format with your own embedded code

PIG

- RANK/sample functions
- Supports UDFs
  - Has built in math UDFs
- Streaming
  - foreach/group by/order by

HIVE

- Sampling
  - Supports sampling bucketized tables
  - Block sampling
- Hook into streaming with ‘Transform’
  - UDFs
  - Group by/Order by

CREATE TABLE mymodel AS SELECT fit_logis(actual, features) model FROM obs1;
CREATE TABLE test_model AS SELECT obs1.actual, predict(features, model) as predicted FROM obs1 JOIN model_out;
ML Tools on Hadoop

Streaming with PIG

trans= load 'my_transactions' as (id, item_id, dt, amt);
highdivs = stream trans through `calculate.pl` as (id, score);

calculate.pl is invoked once on every map or reduce task
ML Tools

R

- Popular tool with Data Scientists & Analysts
- Has numerical and visualization methods for statistics & machine learning
- Strong support for statistical analysis, predictive analytics, data mining, etc.
- Use Reduce Script to invoke R script on a list of records

RHadoop

- rmr – MapReduce
- rhdfs – R interface to HDFS
- rhbase – R interface to HBase

RHIPE

- Provides access to HDFS and MR from within the R environment
- Requires R, RHIPE and protobuf installed on all the data nodes.
ML Algorithms & Mahout

- Collaborative Filtering
- Clustering
- Classification
- Pattern Mining
- MR with Aggregation
- Decision Tree
- Content Analysis

Collaborative Filtering
- Item Based
- User Based
- Slope One Recommender

Clustering
- K-means Clustering
- Canopy Clustering
- Fuzzy K-Means

Classification
- Logistic Regression
- Naïve Bayes
- Complementary Naïve Bayes
- Random Forests

Pattern Mining
- Parallel FP Growth
Mahout - Clustering

K-Means

- hadoop jar my.jar com.cloudera.clustering.featuregen.KMeansInputGenerator input output
- hadoop jar org.apache.mahout.vectorizer.SparseVectorsFromSequenceFiles -i kmeans_input
- mahout seq2sparse -i kmeans_input -o tfidf-vectors
- hadoop jar my.jar com.cloudera.clustering.kmeans.KMeansDriver -i ../tfidf-vectors -k 8000 -dm org.apache.mahout.common.distance.CosineDistanceMeasure -x 10 -ow
Item Based Collaborative Filtering

- Build an item-item matrix determining relationships between pairs of items
- Using the matrix, and the data on the current user, infer his/her taste

### UserID, ItemID, Rating

<table>
<thead>
<tr>
<th>UserID</th>
<th>ItemID</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1000</td>
<td>5.0</td>
</tr>
<tr>
<td>10</td>
<td>1001</td>
<td>3.0</td>
</tr>
<tr>
<td>10</td>
<td>1004</td>
<td>2.5</td>
</tr>
<tr>
<td>13</td>
<td>1001</td>
<td>3.5</td>
</tr>
<tr>
<td>13</td>
<td>1002</td>
<td>4.5</td>
</tr>
<tr>
<td>13</td>
<td>1003</td>
<td>1.0</td>
</tr>
<tr>
<td>13</td>
<td>1004</td>
<td>3.5</td>
</tr>
<tr>
<td>15</td>
<td>1000</td>
<td>4.5</td>
</tr>
<tr>
<td>15</td>
<td>1001</td>
<td>3.5</td>
</tr>
<tr>
<td>15</td>
<td>1002</td>
<td>2.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1000</th>
<th>1001</th>
<th>1002</th>
<th>1003</th>
<th>1004</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1001</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1002</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1003</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1004</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Collaborative Filtering

<table>
<thead>
<tr>
<th></th>
<th>1000</th>
<th>1001</th>
<th>1002</th>
<th>1003</th>
<th>1004</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1001</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1002</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1003</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1004</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

R value for item **1002**:

$$1 \times 5.0 + 2 \times 3.0 + 2 \times 0.0 + 1 \times 0.0 + 1 \times 2.5 = 13.5$$
Mahout – Collaborative Filtering

Item Based Collaborative Filtering

- hadoop jar mahout.jar org.apache.mahout.cf.taste.hadoop.similarity.item.ItemSimilarityJob --input /user/xyz/ratings --output preference -m 10
- Can also be used to generate features for say Logistic Regression
Trends, Aggregation & Counters

- Most Popular Searches/Downloads/News Articles/Movies/Products
- Load results into HBase
- Use HBase where we need NRT count of things (categories/products etc.)
- Impala is very useful here for faster SLAs

HBase Counters
- Has concept of Incrementing column values
  - Avoids lock row/read value/increment it/write it back/unlock rows
  - Great for counting specific metrics over time
  - Example - count per URL/Product
- Can disable write to WAL on puts
Graph Processing with Giraph on Hadoop

- Provides a more natural way to model graph problems
- One implements a Vertex, which has a value and edges and is able to send and receive messages to other vertices in the graph as the computation iterates.
- Bulk Synchronous Processing computing model.
  - Sent messages are available at the next superstep during compute
  - Vertex to Vertex messages – delivered in the next superstep
  - Computation completes when all components complete
- Runs as a single map only job.
- Executes the compute method for every Vertex it is assigned once per Superstep
Graph Processing with Giraph on Hadoop

Use Cases
- Recommend Missing Links in a Social Network
- How are users connected
- Clustering – find related people in groups
- Iterative Graph Ranking
Offline Training & Testing of Models

Hadoop provides an excellent platform to train and test out the Models and various Algorithms
A/B testing is used to test the performance of the Models online

A/B testing involves:

• Partitioning real traffic to two models and then measuring the performance to the desired result (maximize CTR, revenue, page views etc.).
• The partitioning logic can get complicated. In such cases they can be pre-computed on Hadoop offline and pushed to an online store.
Serving Systems

- Load Pre-computed stuff into HBase
- Models generated offline are loaded into the serving systems to compute in real-time
- Cache recommendation for certain time periods and re-calculate on expiry
Bulk Loading into HBase

• Prepare via a MR job using HFileOutputFormat
• Or, use importtsv to prepare the StoreFiles from tsv files
• Import the prepared data using completebulkload tool
• If the target table does not already exist in HBase, it would be created automatically.

```
$ bin/hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=a,b,c -Dimporttsv.bulk.output=/user/xyz/myoutput <tablename> <hdfs-inputdir>

hadoop jar hbase-VERSION.jar completebulkload /user/xyz/myoutput mytable
  Takes the same output path where the importtsv put its results
```
Measuring Performance

Metrics
- CTR
- Revenue
- Engagement
- ...

Why use Hadoop?
- Store all tracking events on HDFS
- Easier to scale to joining high number of tracking streams
- Look back far in time

• Track Behavior & Performance over time
• Retrain a Model if performance drops
• Measure the performance of any newly deployed model
• Try out new stuff
Common Recommendation Patterns

Content/Product Recommendation
People who viewed this Product/Profile also viewed
Frequently bought together
Related Searches or Query Recommendation
Related Articles/News
People/Social Recommendations
Content/Product Recommendation

Use Cases
- Recommend Product
- Recommend Movies/Videos

Design
- Collaborative Filtering
- Logistic Regression

Frequently bought together

Use Cases
- Find items that are frequently bought together
- Related Searches

Design
- Parallel FP Growth
Parallel FP Growth

mahout fpg -i core/src/test/resources/retail.dat -o patterns \ 
-k 50 \ 
-method mapreduce \ 
-regex '\['\]'

32 41 59 60 61 62
3 39 48
63 64 65 66 67 68
32 69
48 70 71 72
39 73 74 75 76 77 78 79
36 38 39 41 48 79 80 81
82 83 84
41 85 86 87 88
Use Cases
• View Item Page
• View Content (Movies, Videos, News)

Design
• This data does not change often
• Record the views in a session – co-view
• Collaborative Filtering
Related Searches or Query Recommendation

Design
• Use Query Log Data
• Cluster similar queries
• Use Parallel FP Growth to find the related searches

Query Distance
• Based on keywords or phrases
• Based on searches in the same session
• Based on common clicked URLs
• Based on the distance of the clicked documents

Related Articles/News
• Batch clustering with K-Means
• NRT clustering using the centroids
• Perform canopy on left over articles
Social/People Recommendations

Use Case
• Recommend Missing Links in a Social Network
• Bipartite Matching – Recommend Men/Women

Design
• Take existing edges and friend of friends
• Build Regression Models based on latest activity
• Scale easily offline with Hadoop as number of friends of friends and activities could be very high.
• Giraph
• June 13 at SF Marriott Marquis

• Call for Speakers is open until April 1

• Early Bird reg open as of Monday night, and Strata attendees get a $25 discount (use code promo-strata)

• hbasecon.com
Thank you!

Jayant Shekhar, Solutions Architect, Cloudera