Benchmarking Spark ML using BigBench

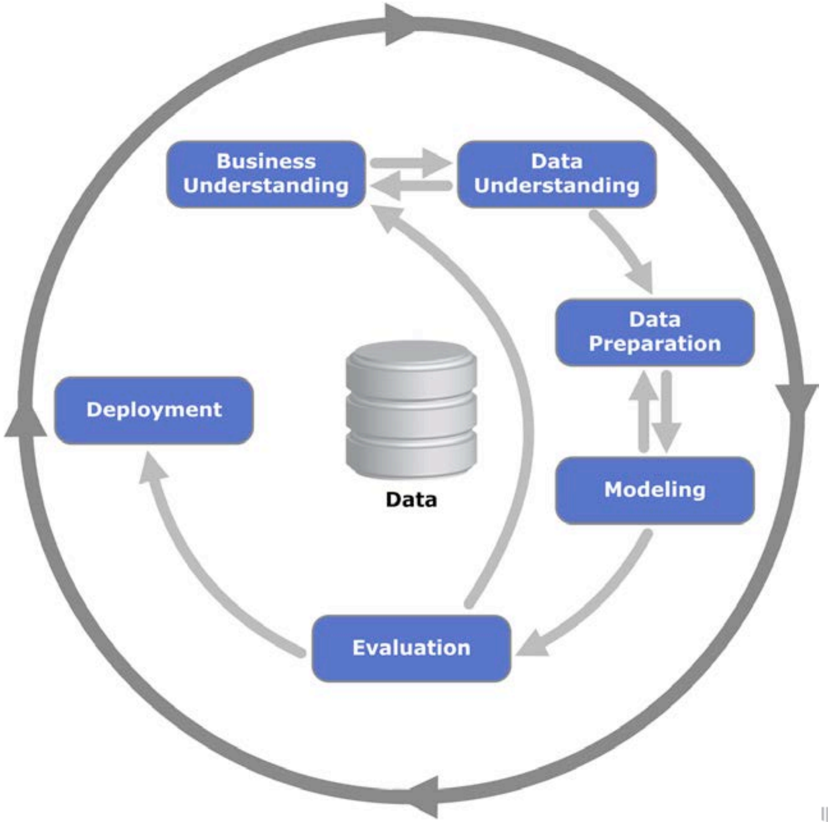
Sweta Singh
singhswe@us.ibm.com
TPCTC 2016

Motivation

Study the performance of Machine Learning use cases on *large* data warehouses in context of assessing

- Alternate approaches to connect from data warehouse to analytics engine
- Different machine learning frameworks

Data preparation and Modeling are the most time consuming phases in a ML cycle



CRISP-DM

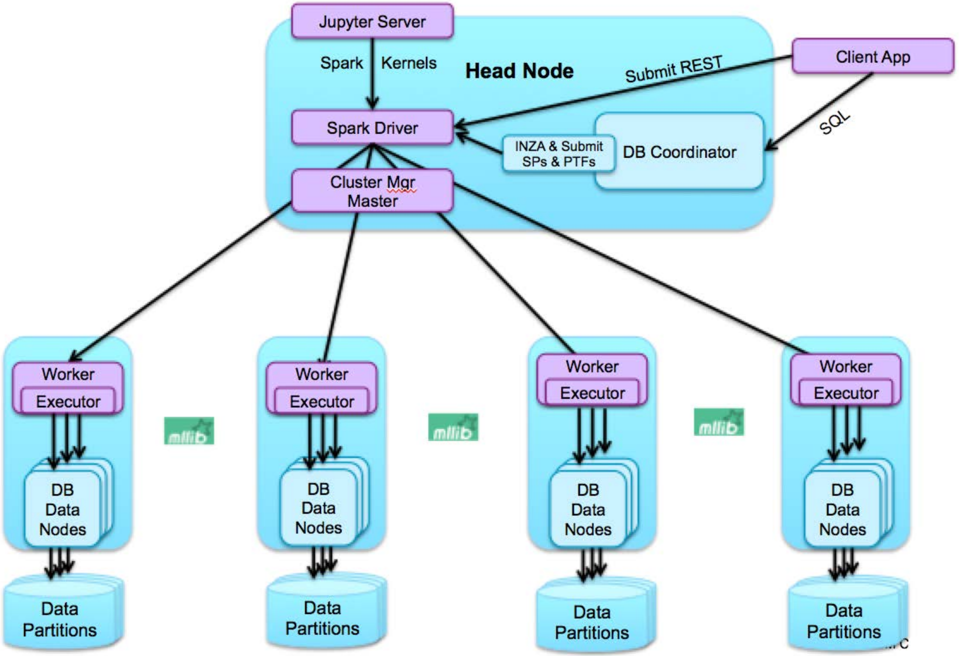
High Speed Data Connectors for Spark

Highly **optimized** and parallel **data transfer between** dashDB and Spark

- **Colocation** of Spark executors and DB2 data nodes
- Optimized exchange of data

Connectors between analytics engine and database can speed up

- ETL during data preparation phase
- Reading from data store during the Model Creation phase
  - Assessing alternate models
  - Tuning the model parameters
  - During model execution
- Writing back the scoring results to the database



dashDB Spark integration Layout

Why BigBench?

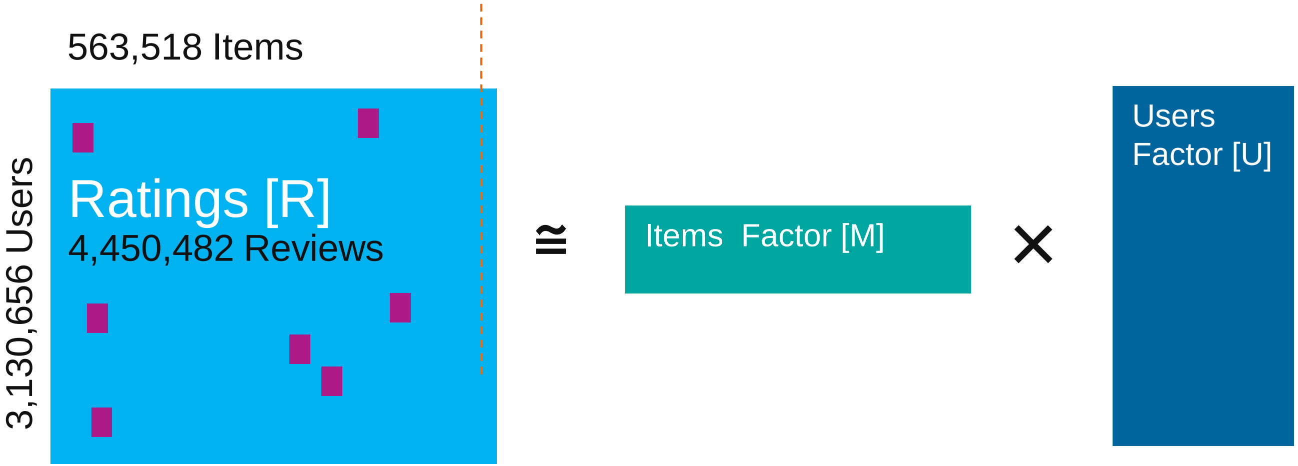
Requirements for benchmarking high speed data connectors

- Representative of a realistic use case for performing ML on data warehouse
- Ability to scale to large data volumes
- Supports read and write to data source
- Invoke Machine Learning algorithms via SQL interface (Stored Procedure) or via Spark jobs (using customized RDD to connect to data source)
- Ability to execute multiple streams to test scalability and resource management in an integrated solution where Spark and database co-exist on the same cluster
- Compare efficiency and *accuracy* of Spark MLlib versus IBM ML algorithms

BigBench met most of our requirements

Collaborative Filtering using Matrix Factorization (MF)

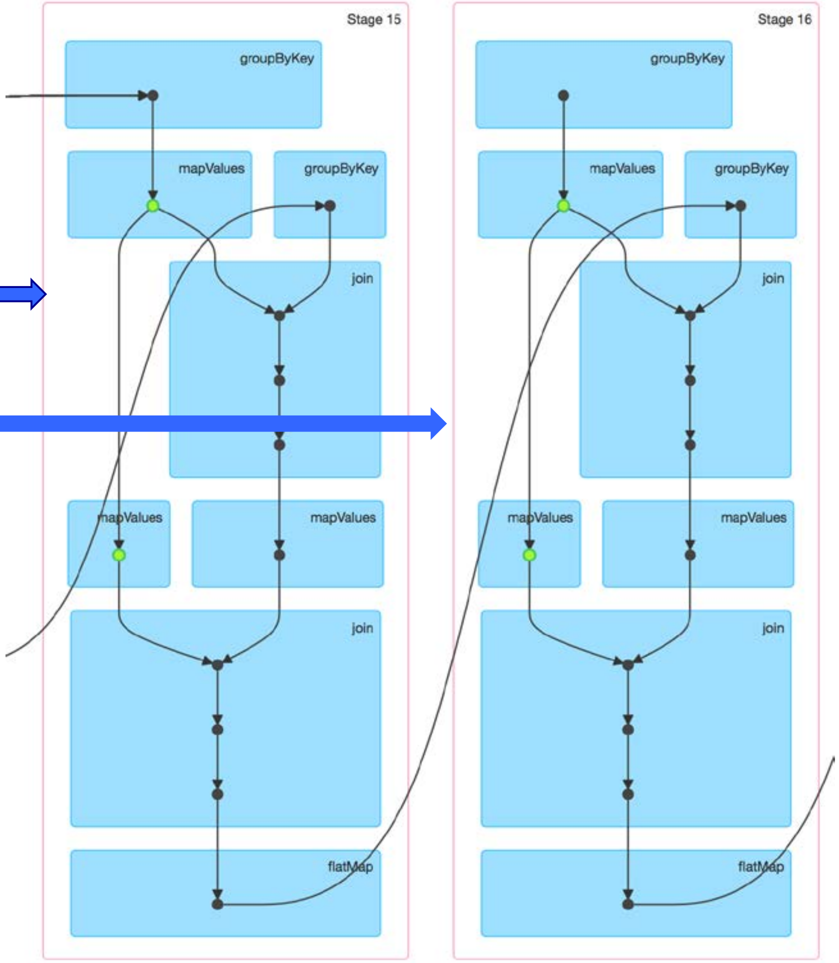
Known for unique challenges
- Data Sparsity: Very few customers rate items
- Scalability: Computational complexity in filling the sparse user item association matrix grows quickly on large data sets



BigBench Sparsity level = 0.00025%

Alternating Least Squares in Spark MLlib

- Step 1: Initialize with random factor
- Step 2: Hold the item factor constant and find the best value for user
- Step 3: Hold the user factor constant and find the best value for item
- Repeat Step 2 & Step 3 for convergence



ALS DAG Visualization

Reference: Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems

Why include Matrix Factorization in BigBench?

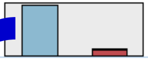
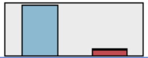
- Unique Performance characteristics
- Trade-off between efficiency and accuracy. Accuracy improves with high number of latent factors with a corresponding drop in performance
- Facilitates creation of real time analytics scenario: Saved Matrix Factorization model can be used to predict ratings on trickling *web_clickstreams* data during the workload run
- Good test bench for comparing implementation and optimizations of different ML frameworks

Q05: Through the SPSS Lens

- Predict if a visitor will be interested in a given item category, based on demographics and existing users online activities (interest in items of different categories)
- Label is 1 if “Clicks in Specified category” > Average Clicks in that category
- Modeler selection & Accuracy varies depending on the specified item category
  - If CLICKS_IN column of the item category is in the input vector, models are able to predict the outcome with 100% accuracy. Models selected are Logistic Regression & models of decision tree family
  - If CLICKS_IN column of the item category is NOT in the input vector, more complex models are chosen and accuracy < 100%

Scenario #1:

- Feature Vector
  – [CLICKS_IN_1, CLICKS_IN_2, CLICKS_IN_3, CLICKS_IN_4, CLICKS_IN_5, CLICKS_IN_6, CLICKS_IN_7, COLLEGE_EDUCATION, MALE]
- Specified category = 3

Graph	Model	Build Time (mins)	Max Profit	Max Profit Occurs in (%)	Lift{Top 30%}	Overall Accuracy (%)	No. Fields Used	Area Under Curve
	C5 1	15	1,564,925.0	9	3.333	100.0	1	1.0
	Logistic regression 1	15	1,564,925.0	9	3.333	100.0	9	1.0
	C&R Tree 1	15	1,564,925.0	9	3.333	100.0	6	1.0

CLICKS_IN_3 <= 756 [Mode: 0] ⇨ 0
CLICKS_IN_3 > 756 [Mode: 1] ⇨ 1

Tree Depth =1

Equation For 0
Equation For 1
0.07511 * COLLEGE_EDUCATION +
-0.01905 * MALE +
0.0001811 * CLICKS_IN_1 +
0.000004033 * CLICKS_IN_2 +
11.52 * CLICKS_IN_3 +
0.0008079 * CLICKS_IN_4 +
-0.001071 * CLICKS_IN_5 +
0.0003695 * CLICKS_IN_6 +
0.00254 * CLICKS_IN_7 +
+ -8715.9

Image: IBM SPSS Modeler Output

Scenario #2:

- Feature Vector
  – [CLICKS_IN_1, CLICKS_IN_2, CLICKS_IN_3, CLICKS_IN_4, CLICKS_IN_5, CLICKS_IN_6, CLICKS_IN_7, COLLEGE_EDUCATION, MALE]

- Specified category = 9

Model | Graph | Summary | Settings | Annotations

Sort by: Area under curve ◉ Ascending ○ Descending ✕ Delete Unused Models View: Training set

...	Graph	Model	Build Time	Max Profit	Max Profit Occurs in (%)	Lift{Top 30%}	Overall Accuracy (%)	No. Fields Used	Area Under Curve
☑		C5 1	4	579,106.558	8	2.896	91.156	9	0.903
☑		Neural Net 1	6	540,565	7	3.018	90.885	9	0.927
☑		Logistic regression 1	4	543,065	7	3.022	90.901	9	0.928

Model | Viewer | Summary | Settings | Annotations

1 2 3 4 5 6 7 8 All

```
⊟ CLICKS_IN_1 <= 233 [ Mode: 0 ]
    CLICKS_IN_5 <= 226 [ Mode: 0 ] ⇨ 0
  ⊟ CLICKS_IN_5 > 226 [ Mode: 0 ]
      CLICKS_IN_6 <= 218 [ Mode: 0 ] ⇨ 0
    ⊞ CLICKS_IN_6 > 218 [ Mode: 0 ]
⊟ CLICKS_IN_1 > 233 [ Mode: 0 ]
  ⊟ CLICKS_IN_5 <= 253 [ Mode: 0 ]
    ⊞ CLICKS_IN_6 <= 225 [ Mode: 0 ]
    ⊞ CLICKS_IN_6 > 225 [ Mode: 0 ]
  ⊟ CLICKS_IN_5 > 253 [ Mode: 1 ]
    ⊞ CLICKS_IN_1 <= 267 [ Mode: 1 ]
    ⊞ CLICKS_IN_1 > 267 [ Mode: 1 ]
```

Tree Depth = 25

```
⊞ Equation For 0
⊟ Equation For 1
    0.01647 * CLICKS_IN_1 +
    0.01627 * CLICKS_IN_2 +
    0.01658 * CLICKS_IN_3 +
    0.01667 * CLICKS_IN_4 +
    0.01651 * CLICKS_IN_5 +
    0.01646 * CLICKS_IN_6 +
    0.01629 * CLICKS_IN_7 +
    -0.003007 * [COLLEGE_EDUCATION=0] +
    0.003655 * [MALE=0] +
    + -25.8
```


Image: IBM SPSS Modeler Output

Scenario #3:

- Feature Vector
  – [CLICKS_IN_1, CLICKS_IN_2, ~~CLICKS_IN_3~~, CLICKS_IN_4, CLICKS_IN_5, CLICKS_IN_6, CLICKS_IN_7, COLLEGE_EDUCATION, MALE]
- Specified category = 3

Graph	Model	Build Time (mins)	Max Profit	Max Profit Occurs in	Lift{Top 3...	Overall Accuracy	No. Fields	Area Under
	Logistic regression...	11	706,920.0	7	3.282	94.145	8	0.967
	Neural Net 1	11	699,860.0	7	3.279	94.088	8	0.966
	CHAID 1	11	641240.571	9	3.266	93.722	6	0.96

1 2 3 4 All

CLICKS_IN_2 <= 786 [Mode: 0] ⇨ 0
CLICKS_IN_2 > 786 and CLICKS_IN_2 <= 818 [Mode: 0]
CLICKS_IN_2 > 818 and CLICKS_IN_2 <= 846 [Mode: 0]
CLICKS_IN_2 > 846 and CLICKS_IN_2 <= 873 [Mode: 0]
CLICKS_IN_2 > 873 and CLICKS_IN_2 <= 901 [Mode: 0]
CLICKS_IN_2 > 901 and CLICKS_IN_2 <= 930 [Mode: 0]
CLICKS_IN_2 > 930 and CLICKS_IN_2 <= 965 [Mode: 0]
CLICKS_IN_2 > 965 and CLICKS_IN_2 <= 1,013 [Mode: 0]

Tree Depth = 8

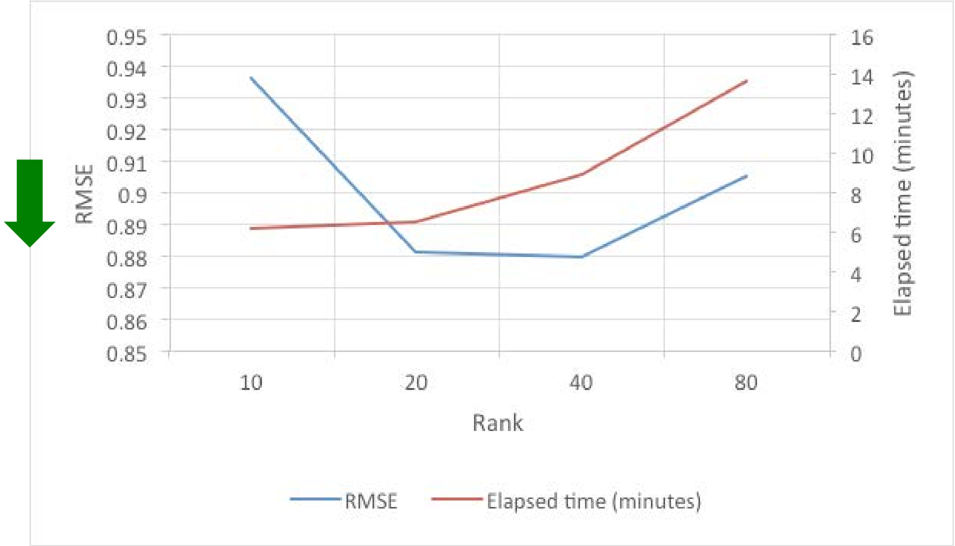
Image: IBM SPSS Modeler Output

Key Learning

- Not including the deterministic clicks in the input feature vector will exercise and stress the machine learning algorithms in a more realistic way. This clearly reflects in the tree depth
- Another benefit is the ability to introduce more complex algorithms such as Neural Networks to the BigBench ML mix

Tuning ML Pipeline

- Model Evaluation phase involves assessing alternate models or tuning the optimization parameters of an algorithm. Tuning is assessed by accuracy on test data sets using cross validation
- Example: Tuning regularization parameter for Logistic Model/ALS, Tuning “rank” for ALS
- Tuning can have interesting side effects on performance



Test Environment
BigBench Scale Factor = 1TB
dashDB Local cluster, CentOS7.0-64 and Spark 1.6.2
4 nodes with the following configuration:

- 24 cores (2.6GHz Intel Xeon-Haswell)
- 512 GB memory
- 10000 Mbps full duplex N/W card

Conclusion & Next Steps

- K-Means use case in BigBench has been very effective in proving the benefits of a high speed connector between data warehouse and Spark

- Our recommendations
  - Broaden the scope of BigBench to more Machine Learning algorithms since performance characteristics of ML algorithms vary
    - Achievable via addition of new use case like Recommender and tweaking existing scenarios like Q05
  - Simulate more ML usecases
    - Real time analytics for Collaborative Filtering
    - Tuning Machine Learning pipeline

- Continued work
  - Investigate ways to incorporate data transformations in the analytic engine layer in BigBench
  - Study the performance characteristics of other ML algorithms on BigBench use case – Trees and Neural Network

Thank you!

Sweta Singh
singhSwe@us.ibm.com