Data Platforms and Pattern Mining

Morteza Zihayat
About Myself

- Big Data Scientist
  - Platform Computing, IBM (2014 – Now)

- PhD Candidate (2011 – Now)
  - Lassonde School of Engineering, York University

Main research interests

- Big Data Mining and Engineering
  - Finding Meaningful Patterns from Structured and Unstructured Big Data
  - Parallel and Distributed Data Mining
- Mining Dynamic and Stream Data
- Social Network Analysis
Agenda

- How Big is this Big Data?
- Big Data and Apache Hadoop
- Apache Spark: Lightning-fast cluster computing
- IBM Platform Conductor for Spark
- A Case Study
  - Mining High Utility Sequential Patterns
  - User behavioral analysis
    - Globe and Mail Newspaper
How Big is this Big Data?

- **2,9 MILLION**
  - Number of mails sent every second

- **375 MEGABYTES**
  - Data consumed by households each day

- **20 HOURS**
  - Video uploaded to YouTube every minute

- **24 PETABYTES**
  - Data per day processed by Google

- **50 MILLION**
  - Tweets per day

- **700 BILLION**
  - Total minutes spent on Facebook each month

- **1,3 EXABYTES**
  - Data sent and received by mobile Internet users

- **398 ITEMS**
  - Products ordered on Amazon per second
What is Big Data?
Challenges

- Read/Write to disk is slow
  - Use multiple disks for parallel read

- Hardware failure
  - Single machine/disk failure
  - Keep multiple copies of data

- How do we merge data from different reads
  - Distributed processing or Hadoop MapReduce
Apache Hadoop

- **Hadoop** is an open-source software framework written in Java
  - Distributed storage
  - Distributed processing of very large data sets
  - Computer clusters

- Two main components
  - **HDFS** (Hadoop Distributed File System)
    - Provides Distributed Storage
  - **MapReduce** (Distributed Data Processing Model)
    - Provides Distributed Processing
HDFS

- Based on Google’s GFS (Google File System)
- Data is distributed across all nodes
- Build around the idea of “write-once, read-many-times” (WORM)
- File are stored as “Blocks”
  - 64MB
  - Different blocks of the same file are stored on different nodes
  - Same block is replicated across several nodes for redundancy
HDFS Architecture

- **NameNode**
  - It stores **metadata** (No of blocks, On which rack which DataNode the data is stored and other details)

- **DataNode**
  - It stores the **actual** data.

- There is **only one** NameNode in a cluster and **many** DataNodes
MapReduce

- MapReduce is a method for distributing a task across multiple nodes

- Consists of two phases
  - Map
  - Reduce

- The data is processed in the form of `<key, value>`

- Each map task processes a discrete portion of the overall data

- After all Maps are complete, the system distributes the intermediate data to nodes which perform Reduce phase (aggregation)
Example: Word counting

• Consider the problem of counting the number of occurrences of each word in a large collection of documents
  • Input: documents
  • Output: <word,frequency>

Divide collection of documents among the class.

Each student gives count of individual word in a document. Repeats for assigned quota of documents.

Sum up the counts from all the documents to give final answer.
Word Count Execution

Input

the quick brown fox
the fox ate the cow
how now brown cow

Output

brown, 2 fox, 2 how, 1 now, 1 the, 3
ate, 1 cow, 2 quick, 1

the, 1 brown, 1 fox, 1 quick, 1
brown, 1
the, 1 fox, 1
the, 1
ate, 1
cow, 1
brown, 1
the, 1 fox, 1
det, 1 cow, 1
how, 1
now, 1
brown, 1
cow, 1
Word Count Execution

Input

the quick brown fox
the fox ate the cow
how now brown cow

Map tasks

Map
the, 1
brown, 1
fox, 1
quick, 1

Map
the, 1
fox, 1
the, 1
ate, 1
cow, 1

Map
how, 1
now, 1
brown, 1
cow, 1

Reduce
brown, 1

Reduce
brown, 1

Output

brown, 2
fox, 2
how, 1
now, 1
the, 3

ate, 1
cow, 2
quick, 1
MapReduce: Execution overview

Master Server distributes M map tasks to machines and monitors their progress.

Map task reads the allocated data, saves the map results in local buffer.

Shuffle phase assigns reducers to these buffers, which are remotely read and processed by reducers.

Reducers output the result on stable storage.
MapReduce on a Cluster with Hadoop HDFS
Problem #1

- MapReduce I/O sandbags runtime for advanced analytics.
  - Must persist results after each pass through data
  - Advanced analytics often requires multiple passes through data
Example: Logistic Regression

- Goal: find best line separating two sets of points

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```

```
Example: Logistic Regression

- Goal: find best line separating two sets of points

Hadoop

```
Problem #2

- Many “point” solutions for advanced analytics in Hadoop
  - Queries
    - Apache drill, …
  - Machine Learning
    - Mahout, H2o, …
  - Graph Analytics
    - Graphlab, …
  - Streaming Analytics
    - S4, …
Apache Spark

- It was developed in response to limitations in the MapReduce
- Originally developed in 2009 in UC Berkeley’s AMP Lab
- Fully open sourced in 2010 – now a Top Level Project at the Apache Software Foundation
Easy and Fast Big Data

- **Easy to Develop**
  - Rich APIs in Java, Scala, Python
  - Interactive shell

- **Fast to Run**
  - In-memory storage

2-5× less code

Up to 10× faster on disk, 100× in memory
Resilient Distributed Datasets (RDD)

- Spark revolves around RDDs
- Fault-tolerant collection of elements that can be operated on in parallel
  - Parallelized Collection: Scala collection which is run in parallel
  - Hadoop Dataset: records of files supported by Hadoop
  - The data flow from one iteration to another happens through memory
    - Doesn't touch the disk.
  - When the memory is not sufficient enough for the data to fit it
    - It can be either spilled to the drive or is just left to be recreated upon request for the same
 RDD Operations

- Transformations
  - Creation of a new dataset from an existing
    - map, filter, distinct, union, sample, groupByKey, join, etc…

- Actions
  - Return a value after running a computation
    - collect, count, first, takeSample, foreach, etc…
RDD Persistence / Caching

- Variety of storage levels
  - memory_only (default), memory_and_disk, etc…

- API Calls
  - `persist(StorageLevel)`
  - `cache()` – shorthand for `persist(StorageLevel.MEMORY_ONLY)`

- Considerations
  - Read from disk vs. recompute (memory_and_disk)
  - Total memory storage size (memory_only_ser)
  - Replicate to second node for faster fault recovery (memory_only_2)
    - Think about this option if supporting a web application
Cache Scaling Matters

Execution time (s) vs. % of working set in cache

Cache disabled: 69
25%: 58
50%: 41
75%: 30
Fully cached: 12
Logistic Regression Performance

![Graph showing running time across iterations for different platforms: Hadoop and Spark.](image-url)
A single integrated platform for advanced analytics

- BlinkDB (Approximate SQL)
- Shark (SQL)
- Spark Streaming (Streaming)
- MLLib (Machine learning)
- GraphX (Graph Computation)
- SparkR (R on Spark)
Spark Performance

- **Machine Learning**
  - 100x faster than MapReduce

- **Queries (Shark)**
  - 100x faster than Hive

- **Streaming**
  - 2X throughput of Storm

- **Graph (GraphX)**
  - 10X faster than MapReduce
IBM Platform Conductor for Spark

- Addresses the following challenges
  - Integrating Spark into existing environments
  - Managing Spark lifecycles in the face of frequent updates to open-source Spark distributions
  - Managing numerous Spark deployments within the organization

- Benefits
  - Cuts costs by using a service orchestration framework to maximize resource utilization
  - Improves performance and efficiency
  - Simplifies application management
A Case Study

*High Utility Sequential Pattern Mining in Big Data*
Frequent Pattern Mining

- **Frequent Pattern Mining (FPM)**
  - FPM is a fundamental research topic in data mining
  - Example application
    - Discover sets of items (i.e., itemsets) that are frequently purchased together by customers

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>{Bread, Milk}</td>
</tr>
<tr>
<td>$T_2$</td>
<td>{Bread, Milk}</td>
</tr>
<tr>
<td>$T_3$</td>
<td>{Bread, Milk, Diaper, Beer}</td>
</tr>
<tr>
<td>$T_4$</td>
<td>{Bread, Milk, Diaper, Beer}</td>
</tr>
<tr>
<td>$T_5$</td>
<td>{Diamond, Necklace}</td>
</tr>
<tr>
<td>$T_6$</td>
<td>{Diamond, Necklace}</td>
</tr>
</tbody>
</table>

Minimum support threshold: 60%
Sup({Bread, Milk}) = 4/6 = 66.6%
{Bread, Milk} is a frequent itemset
Domain Driven Actionable Knowledge Discovery

- The identified patterns are handed over to business people
- They cannot interpret the patterns for business use
  - There are many patterns/not informative
  - Not interested to business needs.
  - How to interpret the patterns to business actions
Insufficiency of Frequent Pattern Mining

- In Market Analysis
  - Business objective: Increase Revenue
  - May lose infrequent but valuable patterns
  - May present too many frequent but unprofitable patterns
  - Cannot find patterns having high profits
A Motivation Example

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
<th>Item</th>
<th>Unit Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>{Bread(1), Milk(1)}</td>
<td>Bread</td>
<td>20</td>
</tr>
<tr>
<td>$T_2$</td>
<td>{Bread(1), Milk(1)}</td>
<td>Milk</td>
<td>30</td>
</tr>
<tr>
<td>$T_3$</td>
<td>{Bread(1), Milk(1), Diaper(3), Beer(6)}</td>
<td>Diamond</td>
<td>1,000</td>
</tr>
<tr>
<td>$T_4$</td>
<td>{Bread(1), Milk(1), Diaper(3), Beer(6)}</td>
<td>Necklace</td>
<td>300</td>
</tr>
<tr>
<td>$T_5$</td>
<td>{Diamond(1), Necklace(1)}</td>
<td>Diaper</td>
<td>300</td>
</tr>
<tr>
<td>$T_6$</td>
<td>{Diamond(1), Necklace(1)}</td>
<td>Beer</td>
<td>70</td>
</tr>
</tbody>
</table>

\{Bread, Milk\}: $200

\{Diamond, Necklace\}: $2600

\{Diaper, Beer\}: $2640
High Utility Sequential Pattern Mining

- Given a set of sequences: find all sequences whose utility is > a user-specified minimum threshold
  - Each item has quantity in a transaction
  - Each item has a value (e.g., price)

<table>
<thead>
<tr>
<th>Items</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk</td>
<td>$3</td>
</tr>
<tr>
<td>Egg</td>
<td>$2</td>
</tr>
<tr>
<td>Birthday Cake</td>
<td>$20</td>
</tr>
<tr>
<td>Birthday Card</td>
<td>$10</td>
</tr>
<tr>
<td>Bread</td>
<td>$1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CID</th>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>T1</td>
<td>{(Bread,2), (Milk,6)}</td>
</tr>
<tr>
<td>C1</td>
<td>T2</td>
<td>{(Birthday Card,2)}</td>
</tr>
<tr>
<td>C1</td>
<td>T3</td>
<td>{(Birthday Cake,2), (Egg,3)}</td>
</tr>
<tr>
<td>C2</td>
<td>T3</td>
<td>{(Bread,2), (Milk,4), (Yoghurt,3), (Tuna,5)}</td>
</tr>
<tr>
<td>C2</td>
<td>T4</td>
<td>{(Egg,5), (Pizza,4), (Juice,2)}</td>
</tr>
<tr>
<td>C3</td>
<td>T5</td>
<td>{(Bread,2), (Yoghurt,4), (Milk,3)}</td>
</tr>
<tr>
<td>C3</td>
<td>T6</td>
<td>{(Milk,1), (cheese,2)}</td>
</tr>
</tbody>
</table>
What is utility?

- Utility of item in a transaction = internal utility (quantity of items in the transaction) x external utility (profit of the item).
  - \( U(\text{Milk}, T1) = 3 \times 6 = 18 \)
- Utility of itemset in a sequence = sum of utilities of its items:
  - \( U(\{\text{Bread, Milk}\}, C1) = 2 \times 1 + 6 \times 3 = 20 \)
- Utility of sub-sequence in a sequence = sum of its itemsets’ utilities
  - \( U(\{\text{Milk}\}, C1) = 3 \times 6 + 3 \times 2 = 24 \)
  - If more than one occurrence, then maximum value among occurrences

<table>
<thead>
<tr>
<th>Items</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk</td>
<td>$3</td>
</tr>
<tr>
<td>Egg</td>
<td>$2</td>
</tr>
<tr>
<td>Birthday Cake</td>
<td>$20</td>
</tr>
<tr>
<td>Birthday Card</td>
<td>$10</td>
</tr>
<tr>
<td>Bread</td>
<td>$1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CID</th>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>T1</td>
<td>{(Bread, 2), (Milk, 6)}</td>
</tr>
<tr>
<td>C1</td>
<td>T2</td>
<td>{(Birthday Card, 2)}</td>
</tr>
<tr>
<td>C1</td>
<td>T3</td>
<td>{(Birthday Cake, 2), (egg, 3)}</td>
</tr>
<tr>
<td>C2</td>
<td>T3</td>
<td>{(Bread, 2), (Milk, 4), (Yoghurt, 3), (Tuna, 5)}</td>
</tr>
<tr>
<td>C2</td>
<td>T4</td>
<td>{(egg, 5), (Pizza, 4), (Juice, 2)}</td>
</tr>
<tr>
<td>C3</td>
<td>T5</td>
<td>{(Bread, 2), (Yoghurt, 4), (Milk, 3)}</td>
</tr>
<tr>
<td>C3</td>
<td>T6</td>
<td>{(Milk, 1), (cheese, 2)}</td>
</tr>
</tbody>
</table>
A Rule-Based News Recommendation System
Globe and Mail Dataset

- Globe and Mail Inc.
  - Corpus of news (142,163,909 articles)
- Web clickstream
  - 2 billion records
  - Each record
    - A sequence of visited news
    - 245 attributes
  - 4 TB
A Rule-Based Recommendation System

**Phase 1**
- Discover Frequent Patterns from user logs
  - \(<\text{News1}, \text{New2}>\)

**Phase 2**
- Extract rules from the patterns
  - \(\text{News1} \rightarrow \text{News2}\)

**Phase 3**

User log

Offline

Online

**Notes:**
- Phase 1: Discover frequent patterns from user logs.
  - Example pattern: \(<\text{News1}, \text{New2}>\).

**Phase 2:**
- Extract rules from the patterns.
  - Rule: \(\text{News1} \rightarrow \text{News2}\).

**Phase 3:**

- Further processing or implementation based on the extracted rules.
Goal

- A rule based news recommendation engine
  - News1 ➔ News2

- It is quite probable that not all the pages visited by the user are of interest to him/her

- The value or importance of a news changes over time
A Utility Based News Recommendation System

Rule-Based Recommendation System

Navigational rules

Editorial Board

freshness

cursor movement

Time Spent
The Proposed Framework

Phase 1
- Discover High Utility Patterns from user logs
  - <News1, News2>

Phase 2
- Extract Utility-Based rules from the patterns
  - News1 → News2

Phase 3
- Recommend articles based on current user activities
Preliminary Experiment

- Only 120000 records
  - Over 32 GB memory usage in average
  - Around 55 mins run time

- With the exponential growth of data
  - It is impossible to execute algorithms on a single machine
BigHUSP

Sequence Dataset

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
Mapper n → Reducer n

1-sequences and their SWU values

L-HUSP Mining

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
Mapper n → Reducer n

Updated Util

PG-HUSP Generation

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
Mapper n → Reducer n

Global Candidates

G-HUSP Mining

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
Mapper n → Reducer n

G-HUSPs

Apache Spark

IBM Platform Symphony
BigHUSP

- Partition data
- In each mapper
  - Scan input sequences
  - Construct UMatrixs,
  - Calculate required parameters: threshold, ...
  - Remove all blocks for sequences from memory
  - Store UMatrixs in Memory-Disk to re-use
BigHUSP

- Call Uspan to mine local HUSPs using the local UMatrixs and threshold
- Output
  - Local HUSPs and their utilities
BigHUSP

- Aggregate local HUSPs
  - If the pattern does not exist
    - Estimate its utility
- Output
  - Global candidates and their approximate utilities
BigHUSP

Initialization

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
... → ...
Mapper n → Reducer n

L-HUSP Mining

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
... → ...
Mapper n → Reducer n

PG-HUSP Generation

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
... → ...
Mapper n → Reducer n

Global Candidates

G-HUSP Mining

Mapper 1 → Reducer 1
Mapper 2 → Reducer 2
Mapper 3 → Reducer 3
... → ...
Mapper n → Reducer n

ROD<UM>

ROD<L-HUSP, Utility>

Apache Spark

IBM Platform Symphoni
Our recommendations keep user longer

2,537 minutes spent by 107 users

380 minutes spent by 254 users
Experimental Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\delta$ (%)</th>
<th>BigHUSP</th>
<th>BigHUSP$_{Basic}$</th>
<th>BigHUSP$_{SA}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Globe</td>
<td>0.09</td>
<td>1.6 m</td>
<td>3.6 m</td>
<td>0.99 h</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>2.3 m</td>
<td>4.4 m</td>
<td>1.4 h</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>3.1 m</td>
<td>6.6 m</td>
<td>2.2 h</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>5.0 m</td>
<td>11.0 m</td>
<td>3.3 h</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>9.2 m</td>
<td>20.7 m</td>
<td>4.5 h</td>
</tr>
<tr>
<td>synthDS1</td>
<td>0.05</td>
<td>3.0 m</td>
<td>10.0 m</td>
<td>1.1 h</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>4.26 m</td>
<td>14.4 m</td>
<td>1.2 h</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>6.23 m</td>
<td>17.9 m</td>
<td>1.6 h</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>9.9 m</td>
<td>27.2 m</td>
<td>1.9 h</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>14.3 m</td>
<td>29.4 m</td>
<td>3.2 h</td>
</tr>
<tr>
<td></td>
<td>0.009</td>
<td>37.6 m</td>
<td>76.5 m</td>
<td>7.8 h</td>
</tr>
<tr>
<td>ChainStore</td>
<td>0.09</td>
<td>15.0 m</td>
<td>33.4 m</td>
<td>6.4 h</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>19.9 m</td>
<td>56.2 m</td>
<td>12.0 h</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>25.0 m</td>
<td>77.0 m</td>
<td>13.4 h</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>34.8 m</td>
<td>107.7 m</td>
<td>14.6 h</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>38.7 m</td>
<td>159.8 m</td>
<td>17.4 h</td>
</tr>
<tr>
<td>synthDS2</td>
<td>0.09</td>
<td>13.1 m</td>
<td>26.3 m</td>
<td>7.7 h</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>16.3 m</td>
<td>34.5 m</td>
<td>9.3 h</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>20.6 m</td>
<td>47.2 m</td>
<td>15.7 h</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>23.8 m</td>
<td>51.8 m</td>
<td>17.8 h</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>32.2 m</td>
<td>85.3 m</td>
<td>21.4 h</td>
</tr>
</tbody>
</table>
Summary

- What Big Data is
- What Hadoop is and why it is important
- Hadoop main components
- What Spark is
- Why Spark is popular
- High Utility Pattern Mining
- Rule-Based News Recommendation System
- Big Data Analysis Using IBM Conductor for Spark
THANKS!

zihayatm@gmail.com
According to Spark advocates, how much faster can Apache Spark potentially run batch-processing programs when processed in memory than MapReduce can?

- 10 times faster
- 20 times faster
- 100 times faster
- 200 times faster
Q2

- What is the name of the programming framework originally developed by Google that supports the development of applications for processing large data sets in a distributed computing environment?
  - MapReduce
  - Hive
  - Spark
Q3

- True or false? A big data analytics strategy is often defined by the three V's – volume, variety and velocity -- which is helpful but ignores other commonly cited characteristics, such as complexity and variability.
  - True
  - False
Just collecting and storing information isn't enough to produce real business value. Big data analytics technologies are necessary to:

- Formulate eye-catching charts and graphs
- Extract valuable insights from the data
- Integrate data from internal and external sources
Q5

Which of the following application types can Spark run in addition to batch-processing jobs?

- Stream processing
- Machine learning
- Graph processing
- All of the above
Q6

Which of the following is NOT a characteristic shared by Hadoop and Spark?

- Both are data processing platforms
- Both are cluster computing environments
- Both have their own file system
- Both use open source APIs to link between different tools