IBM Platform Computing Group

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

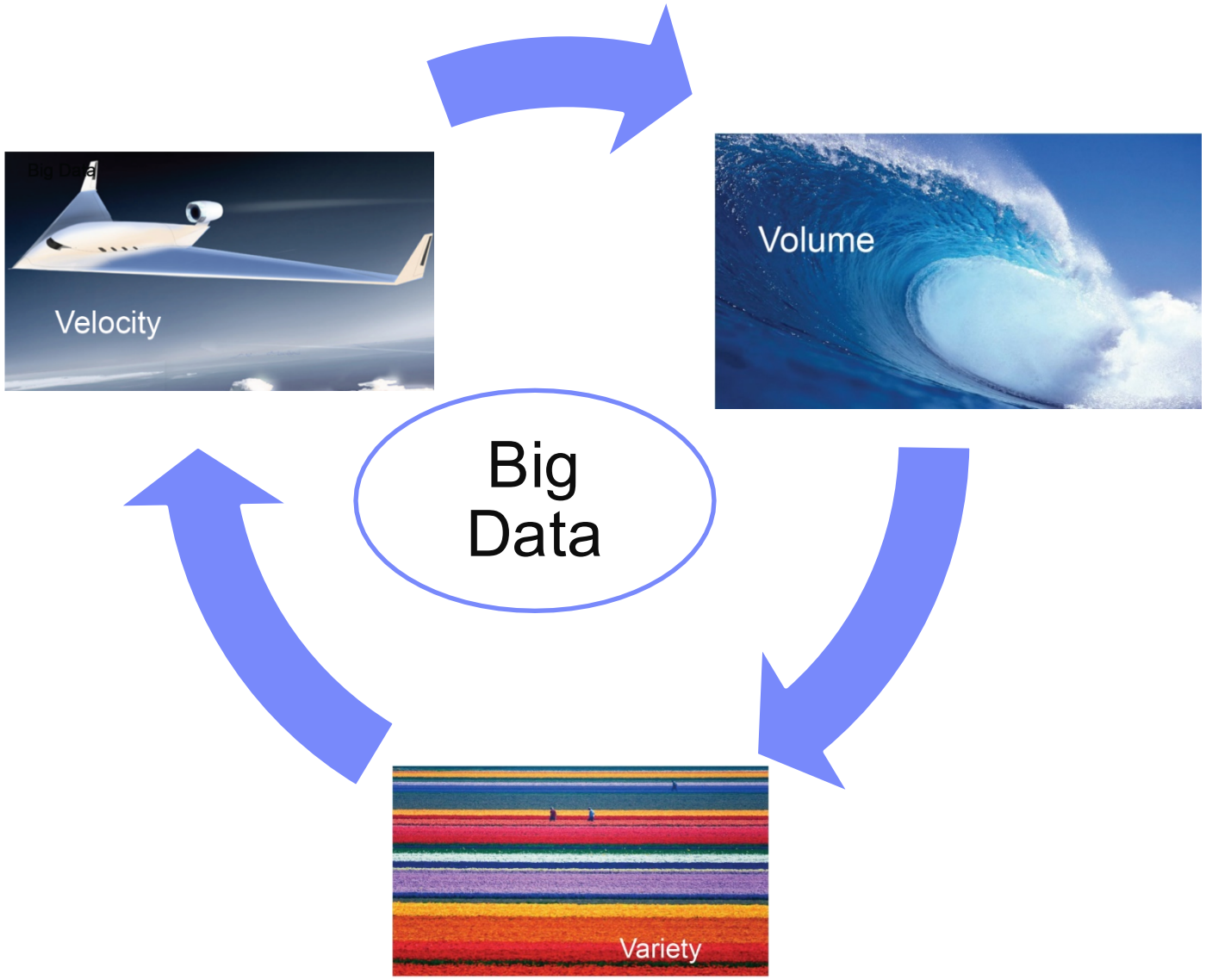
1,3 EXABYTES

DATA SENT AND RECEIVED BY MOBILE INTERNET USERS

398 ITEMS

PRODUCTS ORDERED ON AMAZON PER SECOND

What is Big Data?



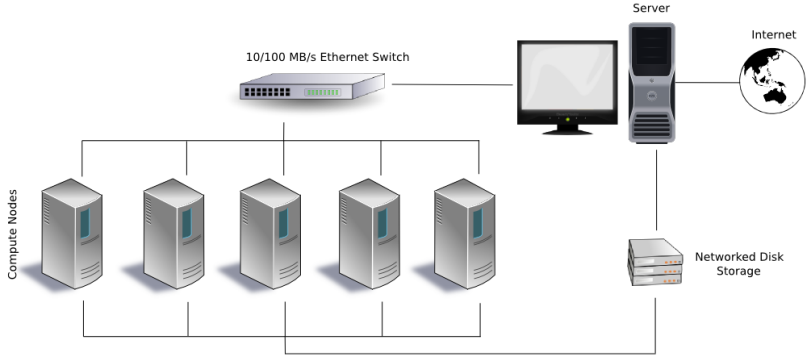
Challenges

- 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

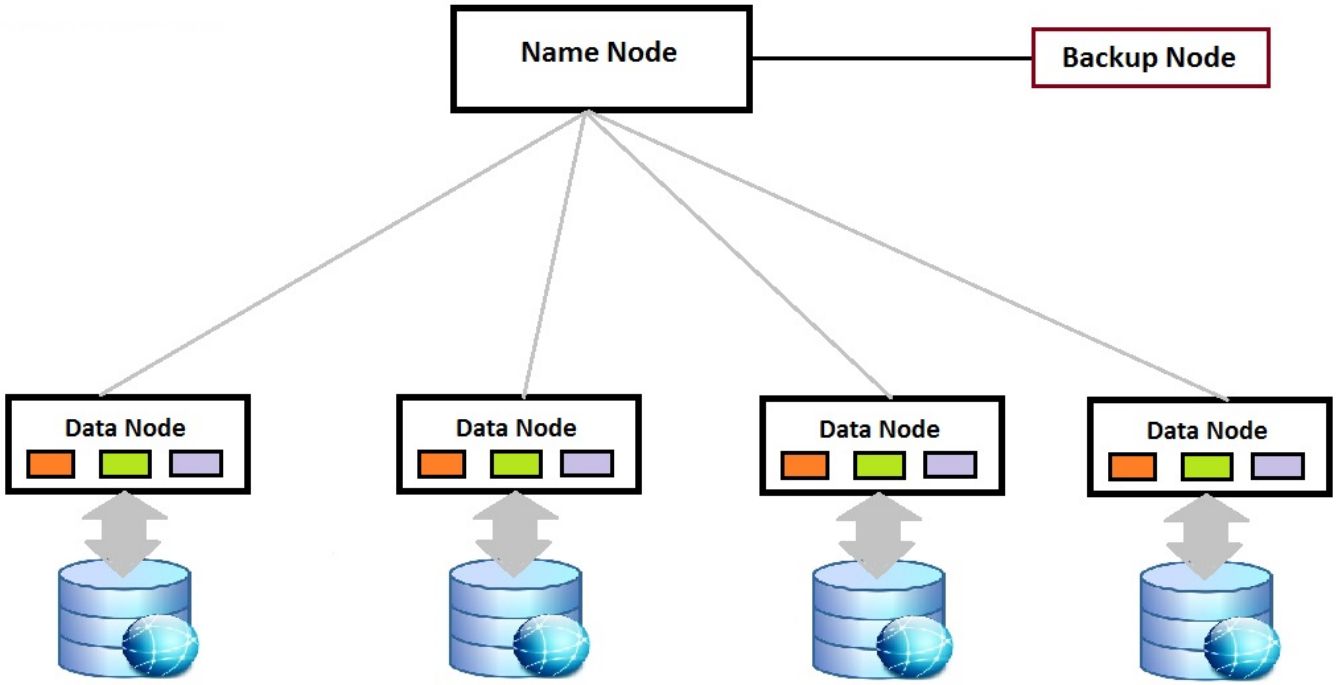


HDFS

- 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

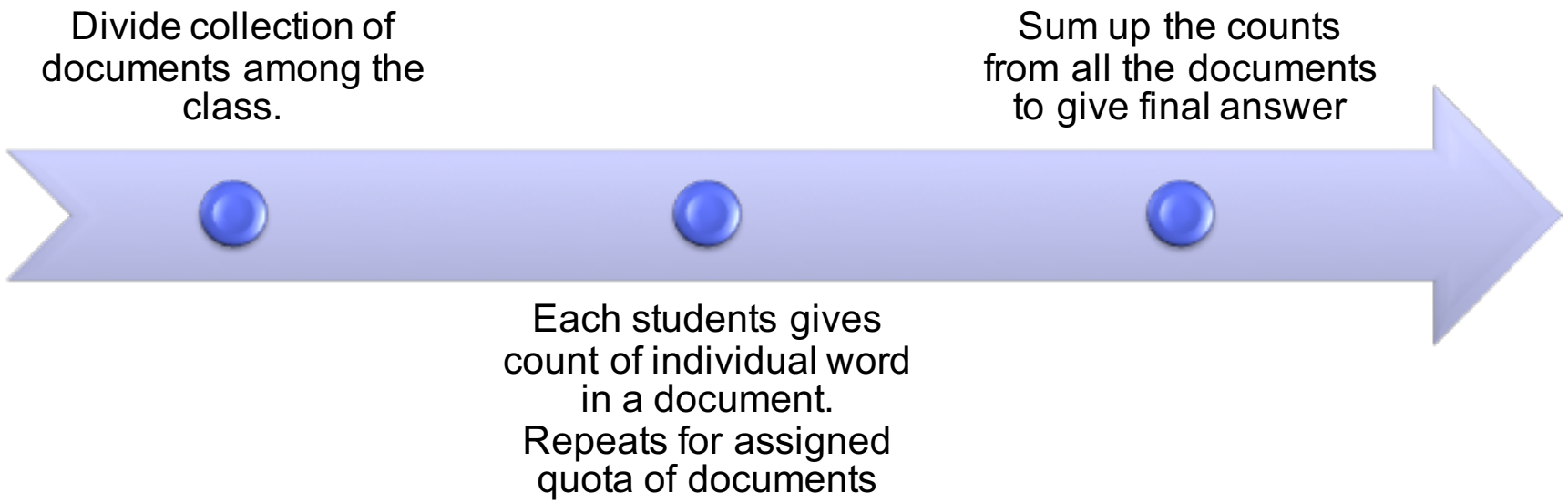


MapReduce

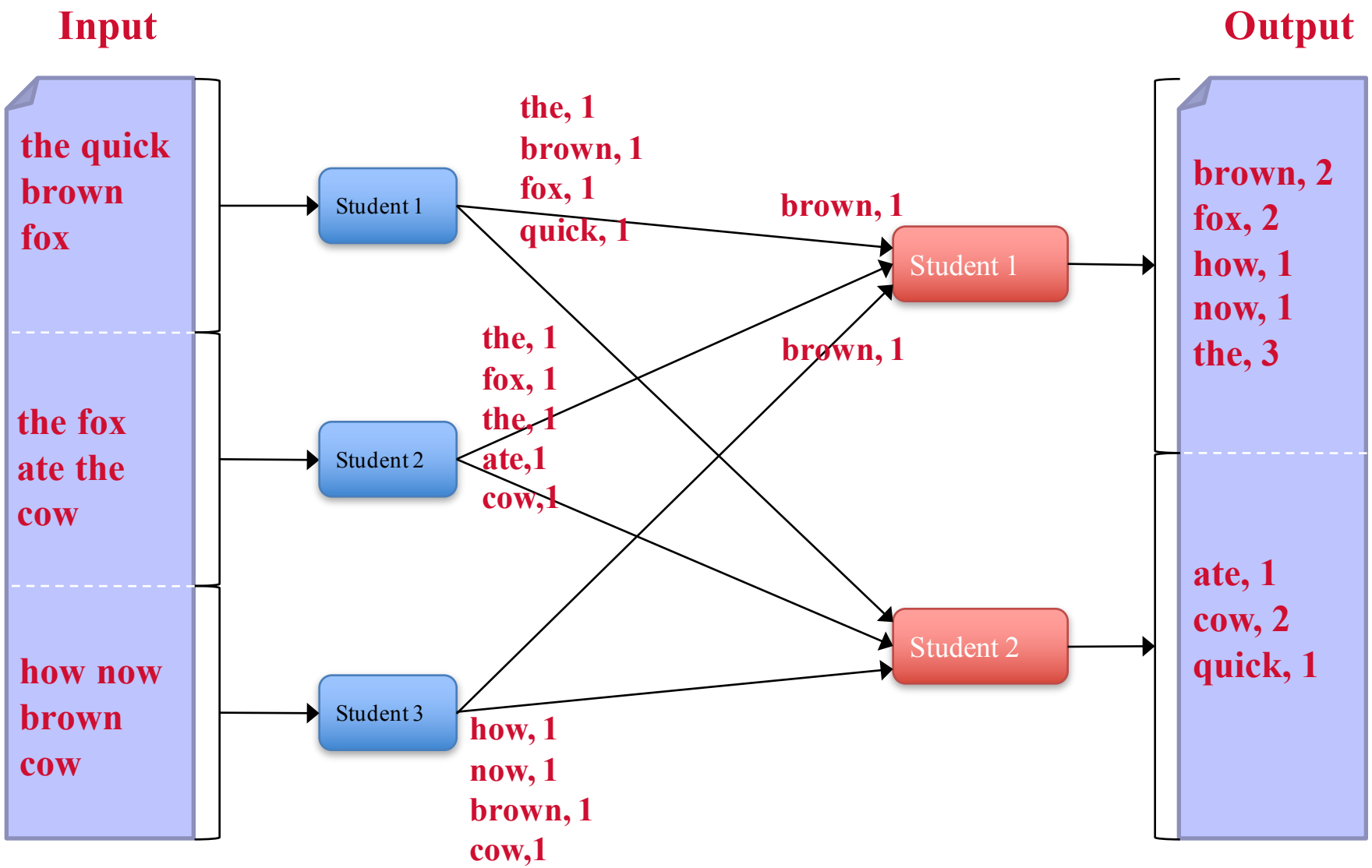
- MapReduce is a method for distributing a task across multiple nodes
- Consists of two phases
  - Map
  - Reduce
- The data is process in the form of <key,value>
- Each map task processes a discrete portion of the overall data
- After all Maps are complete, the system distributes the intermediate data to nodes which perform Reduce phase (aggregation)

Example: Word counting

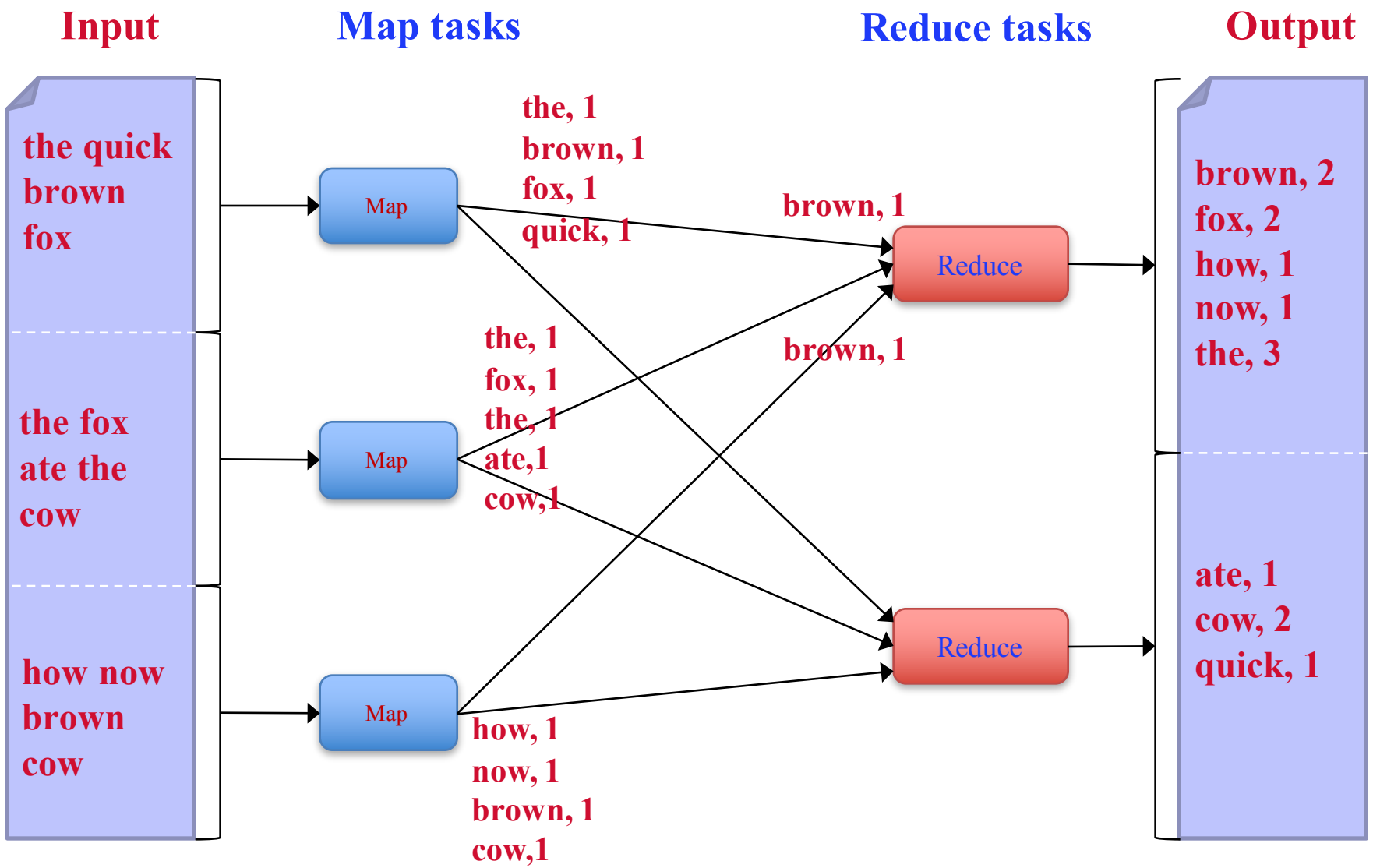
- Consider the problem of counting the number of occurrences of each word in a large collection of documents
  - Input: documents
  - Output: <word,frequency>



Word Count Execution



Word Count Execution



MapReduce: Execution overview

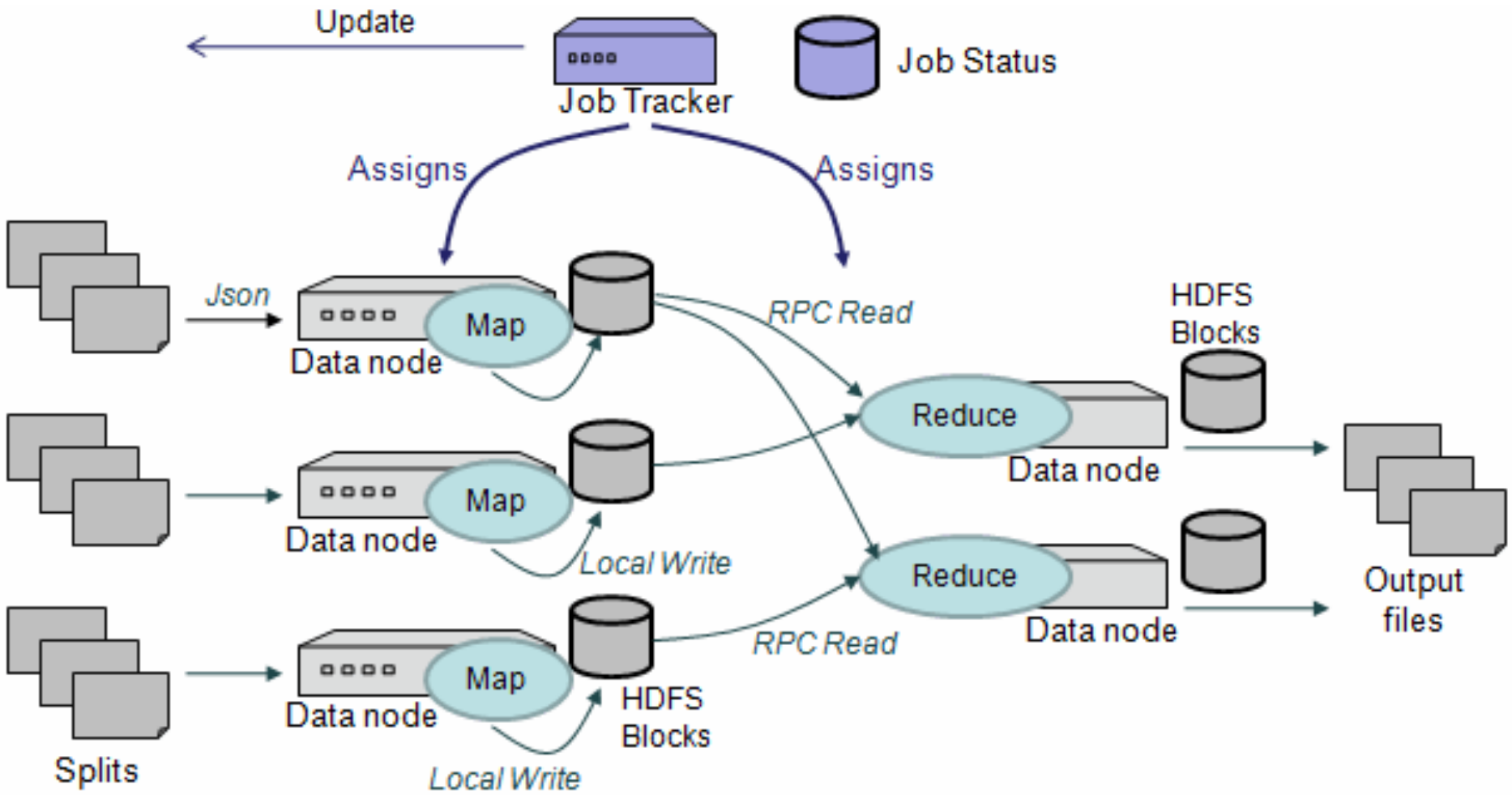
Master Server distributes M map tasks to machines and monitors their progress.

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.

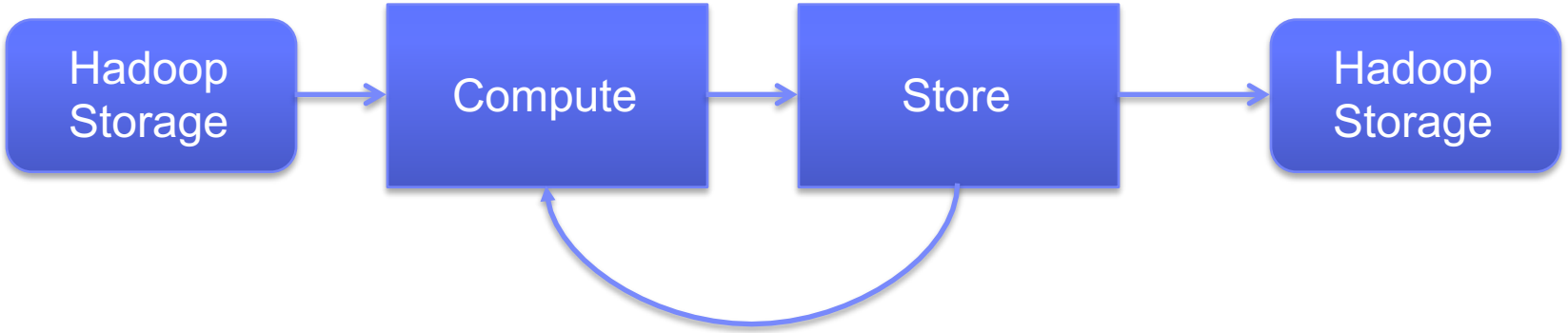
Reducers output the result on stable storage.

MapReduce on a Cluster with Hadoop HDFS



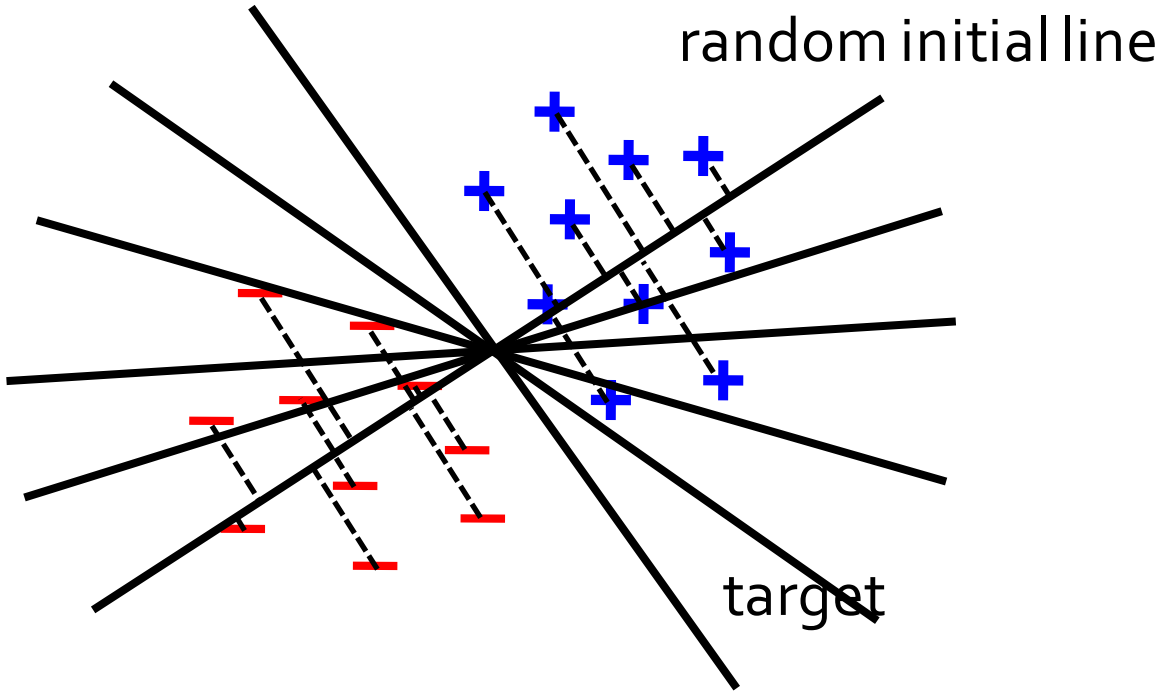
Problem #1

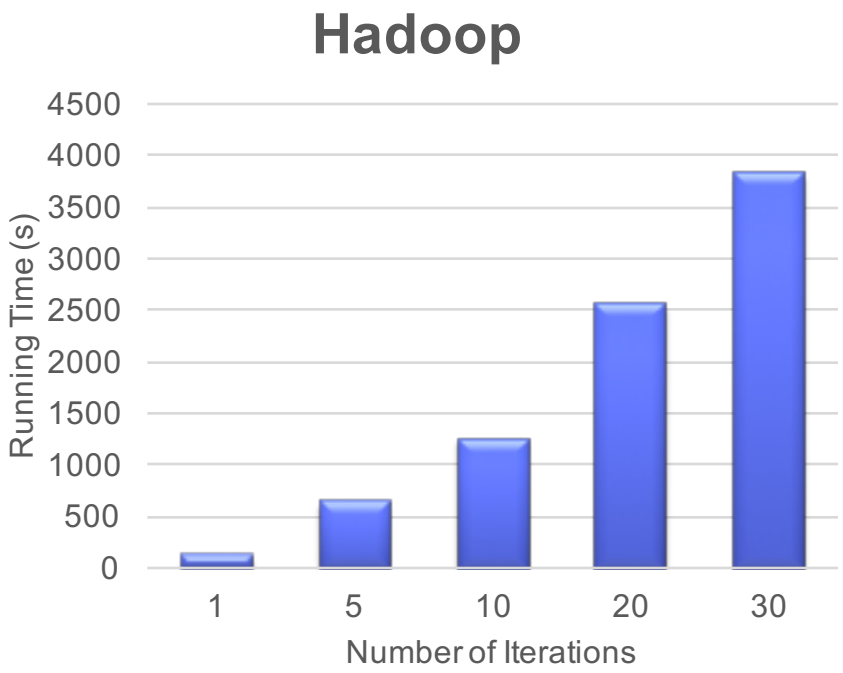
- MapReduce I/O sandbags runtime for advanced analytics.
  - Must persist results after each pass through data
  - Advanced analytics often requires multiple passes through data



Example: Logistic Regression

- Goal: find best line separating two sets of points



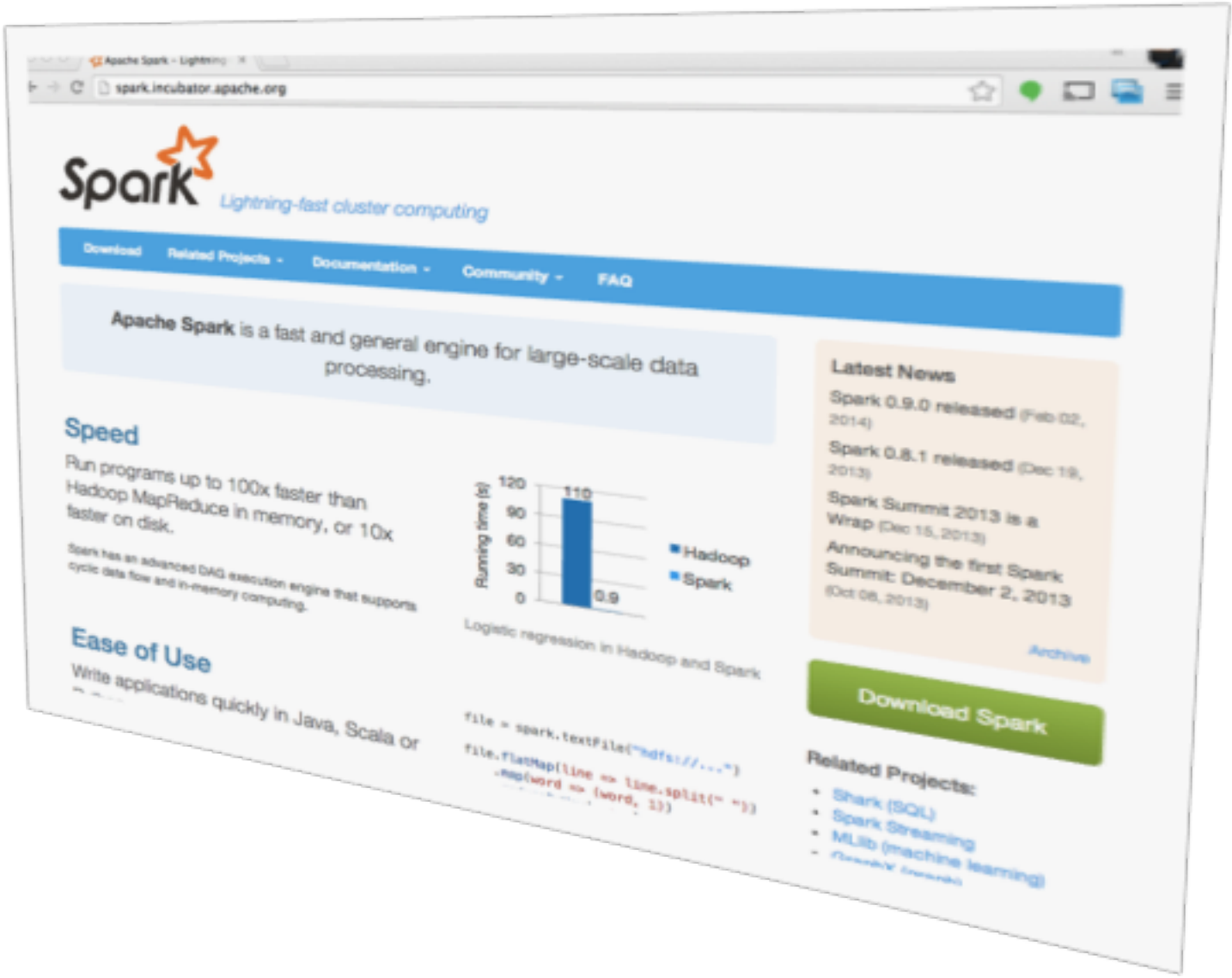


Problem #2

- 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

2-5× less code

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

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

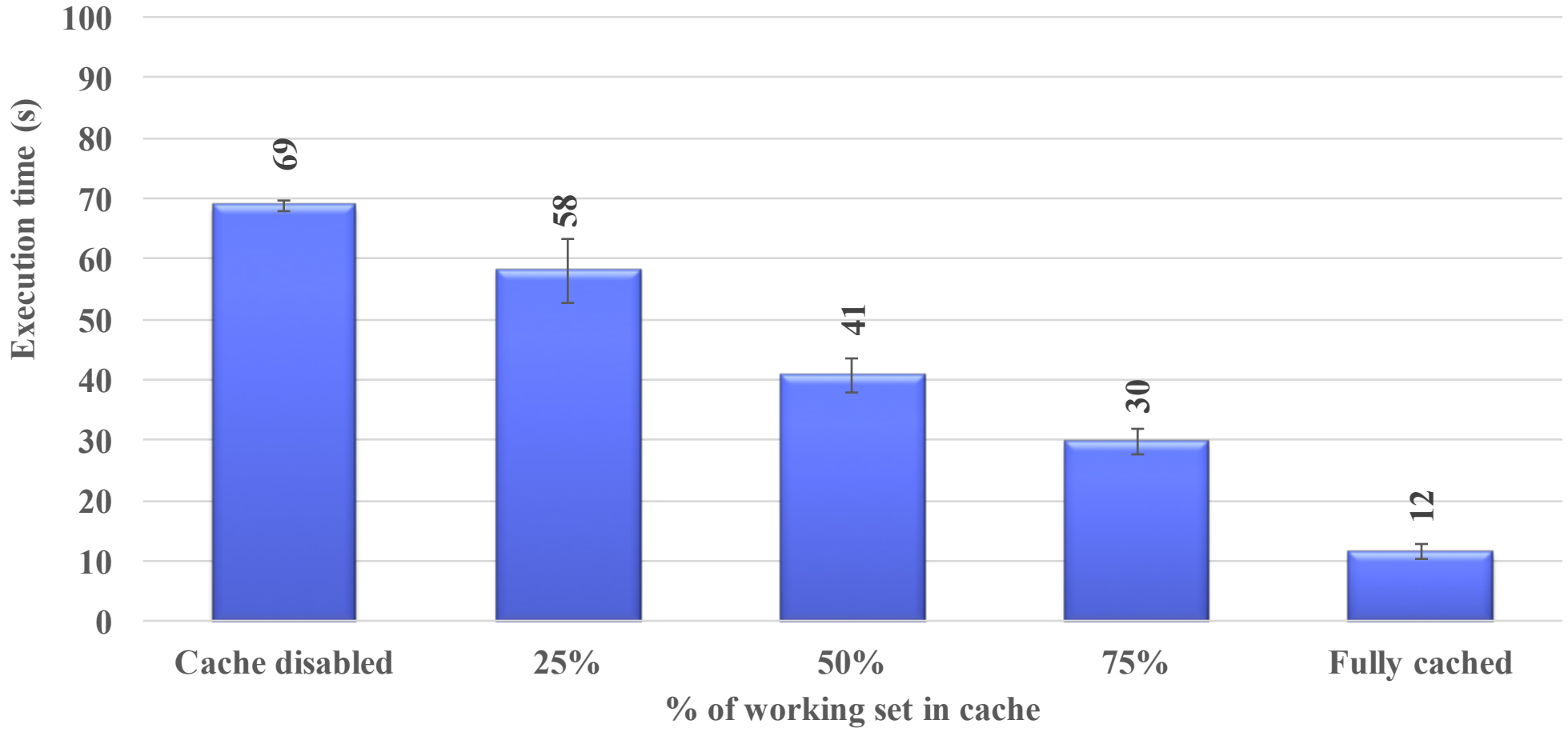
RDD Operations

- 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…

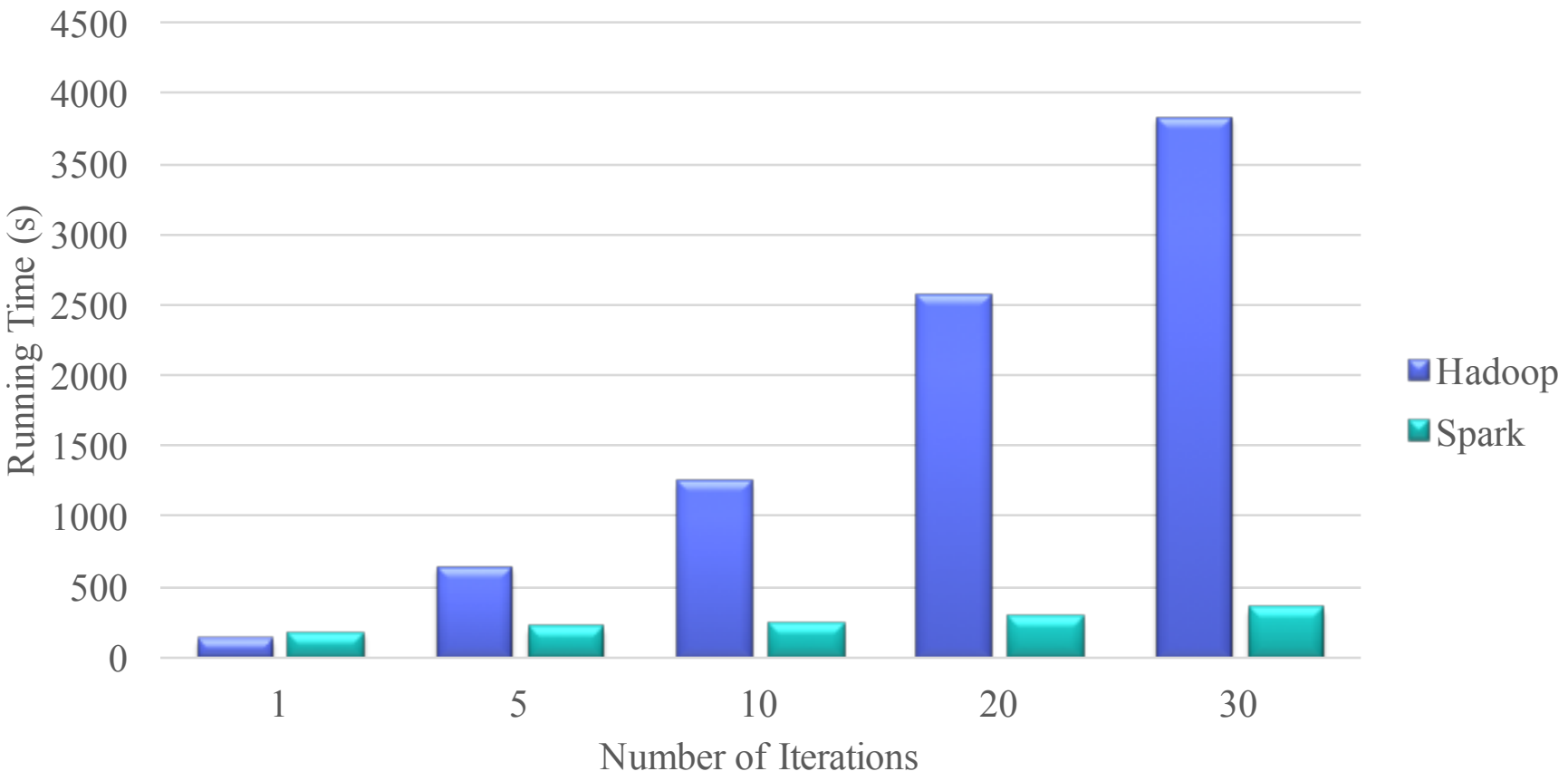
RDD Persistence / Caching

- 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

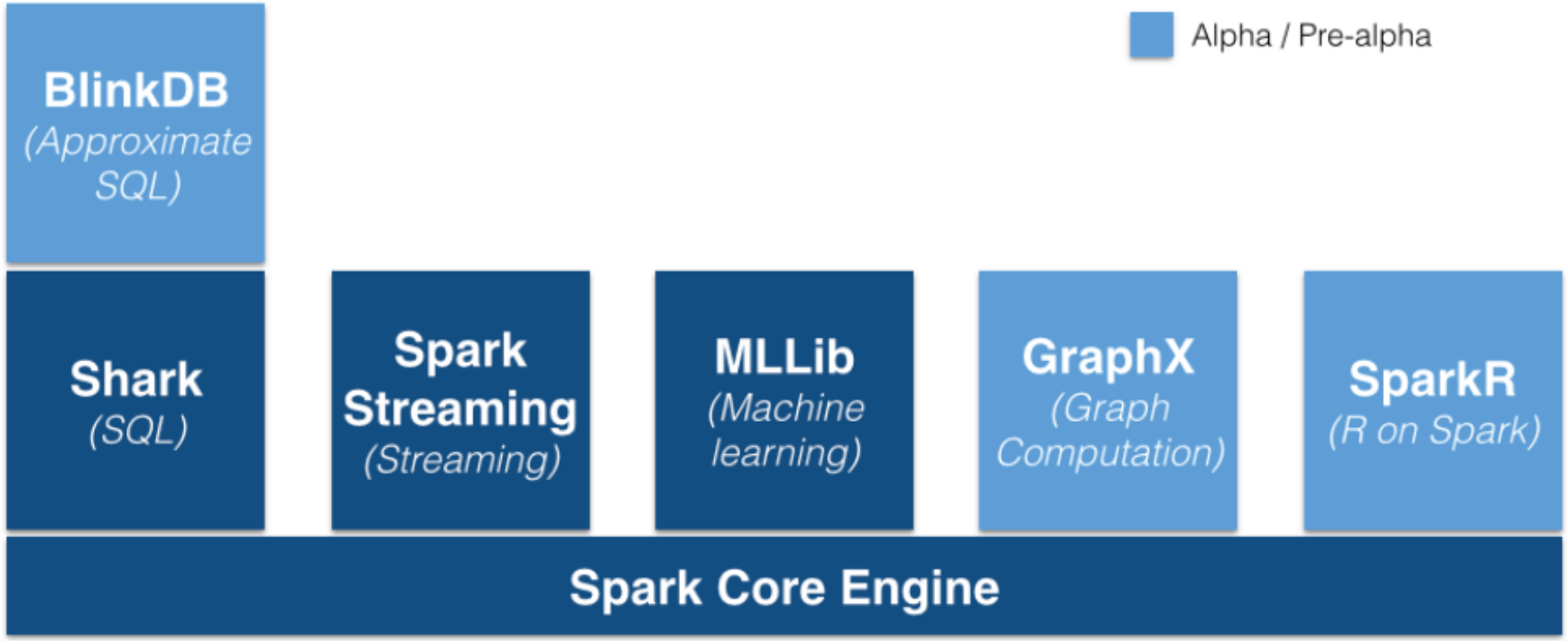
Cache Scaling Matters



Logistic Regression Performance

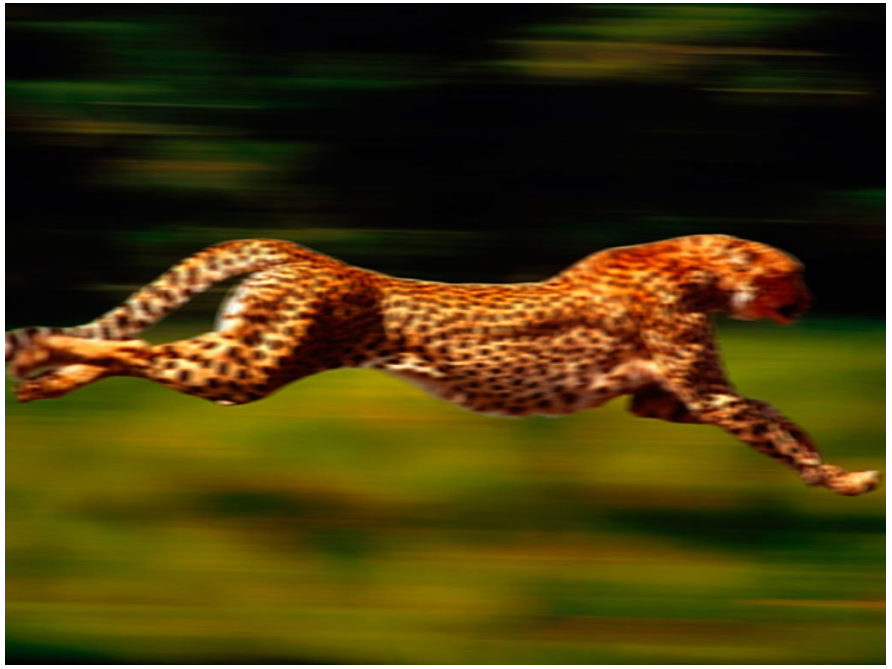


A single integrated platform for advanced analytics



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

TID	Transaction
T_1	{Bread, Milk}
T_2	{Bread, Milk}
T_3	{Bread, Milk, Diaper, Beer}
T_4	{Bread, Milk, Diaper, Beer}
T_5	{Diamond, Necklace}
T_6	{Diamond, Necklace}

Minimum support threshold: 60%

Sup({Bread, Milk}) = 4/6 = 66.6%

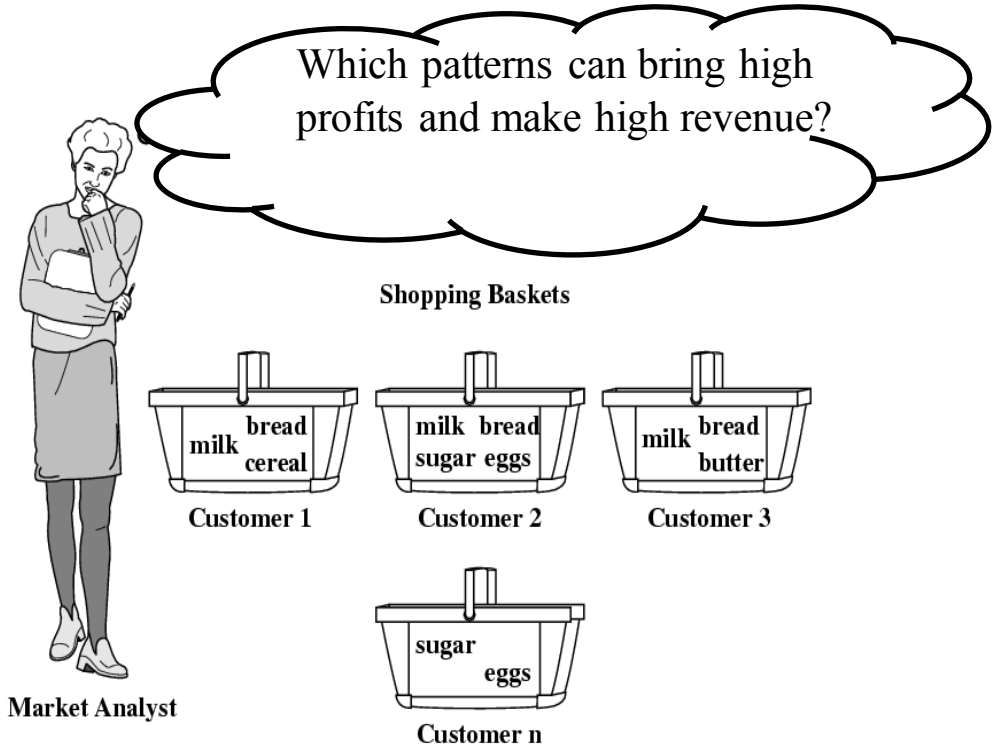
{Bread, Milk} is a frequent itemset

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

TID	Transaction
T_1	{Bread(1), Milk(1)}
T_2	{Bread(1), Milk(1)}
T_3	{Bread(1), Milk(1), Diaper(3), Beer(6)}
T_4	{Bread(1), Milk(1), Diaper(3), Beer(6)}
T_5	{Diamond(1), Necklace(1)}
T_6	{Diamond(1), Necklace(1)}

Item	Unit Profit
Bread	20
Milk	30
Diamond	1,000
Necklace	300
Diaper	300
Beer	70

{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)

Items	Profit
Milk	$3
Egg	$2
Birthday Cake	$20
Birthday Card	$10
Bread	$1

CID	TID	
C1	T1	{(Bread,2), (Milk,6)}
C1	T2	{(Birthday Card,2)}
C1	T3	{ (Birthday Cake,2), (egg,3)}
C2	T3	{(Bread,2), (Milk,4), (Yoghurt,3), (Tuna,5)}
C2	T4	{(egg,5), (Pizza,4), (Juice,2)}
C3	T5	{ (Bread,2),(Yoghurt,4), (Milk,3)}
C3	T6	{(Milk,1), (cheese,2)}

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(Milk,T1) = 3 × 6 = 18
- Utility of itemset in a sequence = sum of utilities of its items:
  - U({Bread, Milk}, C1)= 2 × 1 + 6 × 3 = 20
- Utility of sub-sequence in a sequence = sum of its itemsets' utilities
  - U(<{Milk}{egg}>,C1) = 3 × 6 + 3 × 2 = 24
  - If more than one occurrence, then maximum value among occurrences

Items	Profit
Milk	$3
Egg	$2
Birthday Cake	$20
Birthday Card	$10
Bread	$1

CID	TID	
C1	T1	{(Bread,2), (Milk,6)}
C1	T2	{(Birthday Card,2)}
C1	T3	{ (Birthday Cake,2), (egg,3)}
C2	T3	{(Bread,2), (Milk,4), (Yoghurt,3), (Tuna,5)}
C2	T4	{(egg,5), (Pizza,4), (Juice,2)}
C3	T5	{ (Bread,2),(Yoghurt,4), (Milk,3)}
C3	T6	{(Milk,1), (cheese,2)}

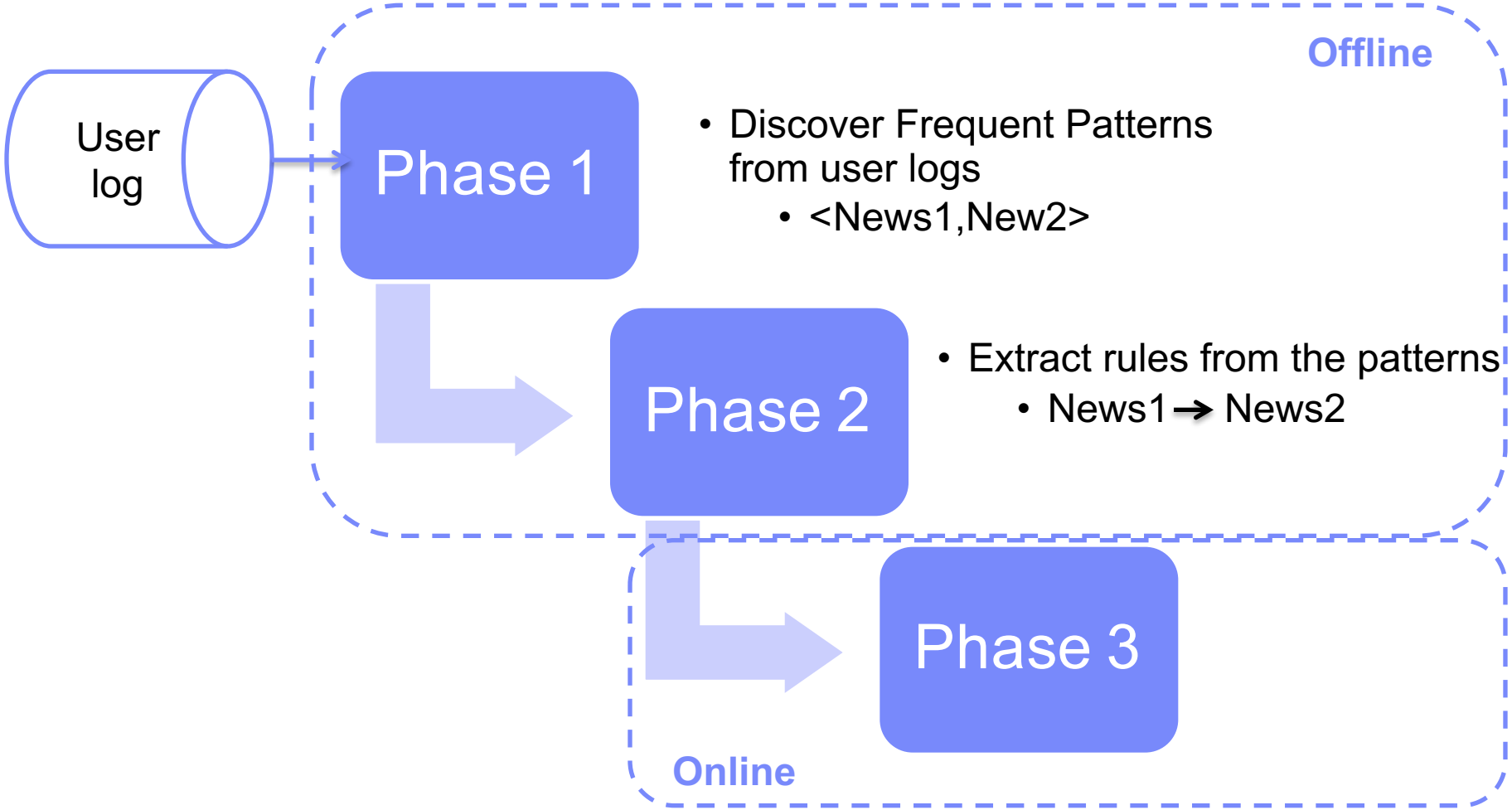
A Rule-Based News Recommendation System

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

```
65.55.212.161 - - [12/Dec/2006:03:27:24 -0600] "GET /java/guide/plugin/developer_guide/applet_tag.html HTTP/1.0" 200 7860
65.55.212.161 - - [12/Dec/2006:03:27:24 -0600] "GET /~rburke/images/Sep-01/ian-abby.jpg HTTP/1.0" 200 89205
65.55.212.161 - - [12/Dec/2006:03:27:25 -0600] "GET /~rburke/images/Sep-01/sylvesters+ian.jpg HTTP/1.0" 200 75805
65.55.212.161 - - [12/Dec/2006:03:27:27 -0600] "GET /java/api/javax/swing/event/class-use/DocumentEvent.html HTTP/1.0" 200 44986
65.55.212.161 - - [12/Dec/2006:03:27:27 -0600] "GET /~rburke/courses/s01/msis554/images/sales.jpg HTTP/1.0" 200 58245
65.55.212.161 - - [12/Dec/2006:03:27:28 -0600] "GET /~rburke/courses/s01/msis554/images/strategy.jpg HTTP/1.0" 200 78556
65.55.212.161 - - [12/Dec/2006:03:27:29 -0600] "GET /~rburke/courses/s01/msis554/images/supplier.jpg HTTP/1.0" 200 63930
65.55.212.161 - - [12/Dec/2006:03:27:30 -0600] "GET /~rburke/courses/s01/msis554/images/technical.jpg HTTP/1.0" 200 103955
65.55.212.161 - - [12/Dec/2006:03:27:31 -0600] "GET /~rburke/courses/s01/msis554/images/operations.jpg HTTP/1.0" 200 63459
65.55.212.161 - - [12/Dec/2006:03:27:32 -0600] "GET /~rburke/courses/s01/msis554/images/customer.jpg HTTP/1.0" 200 65737
65.55.212.161 - - [12/Dec/2006:03:27:32 -0600] "GET /~rburke/courses/s01/msis554/images/security.jpg HTTP/1.0" 200 75751
65.55.212.161 - - [12/Dec/2006:03:27:33 -0600] "GET /~rburke/courses/s01/msis554/images/management.jpg HTTP/1.0" 200 81863
65.55.212.161 - - [12/Dec/2006:03:27:34 -0600] "GET /~rburke/courses/s01/msis554/images/design.jpg HTTP/1.0" 200 86735
65.55.212.161 - - [12/Dec/2006:03:27:35 -0600] "GET /java/guide/rmi/logging.html HTTP/1.0" 200 5106
65.55.212.161 - - [12/Dec/2006:03:27:35 -0600] "GET /java/guide/intl/fontprop.html HTTP/1.0" 200 14900
65.55.212.161 - - [12/Dec/2006:03:27:41 -0600] "GET /java/guide/plugin/developer_guide/proxie_config.html HTTP/1.0" 200 9899
65.55.212.161 - - [12/Dec/2006:03:27:45 -0600] "GET /~rburke/courses/w05/gam224/hwk/leader-schedule.html HTTP/1.0" 200 12178
65.55.212.161 - - [12/Dec/2006:03:27:45 -0600] "GET /~rburke/research/cbrec/cfp.html HTTP/1.0" 200 6453
65.55.212.161 - - [12/Dec/2006:03:27:46 -0600] "GET /java/guide/jdbc/getstart/resultset.gif HTTP/1.0" 200 16765
65.55.212.161 - - [12/Dec/2006:03:27:46 -0600] "GET /~rburke/courses/s04/csc594/kb/people/akass.html HTTP/1.0" 200 1442
65.55.212.161 - - [12/Dec/2006:03:27:46 -0600] "GET /~rburke/courses/s04/csc594/kb/people/rburke.html HTTP/1.0" 200 1452
65.55.212.161 - - [12/Dec/2006:03:27:47 -0600] "GET /java/guide/awt/1.3/AWT_Native_Interface.html HTTP/1.0" 200 25273
65.55.212.161 - - [12/Dec/2006:03:27:47 -0600] "GET /~rburke/images/Sep-01/lily-maggiel.jpg HTTP/1.0" 200 72818
65.55.212.161 - - [12/Dec/2006:03:27:48 -0600] "GET /~rburke/images/Sep-01/david-lilyl.jpg HTTP/1.0" 200 68863
65.55.212.161 - - [12/Dec/2006:03:27:49 -0600] "GET /~rburke/courses/w05/gam224/schedule.html HTTP/1.0" 200 4577
65.55.212.161 - - [12/Dec/2006:03:27:49 -0600] "GET /java/api/javax/naming/event/class-use/NamingListener.html HTTP/1.0" 200 17322
65.55.212.161 - - [12/Dec/2006:03:27:50 -0600] "GET /java/api/javax/swing/event/class-use/AncestorListener.html HTTP/1.0" 200 9365
65.55.212.161 - - [12/Dec/2006:03:27:50 -0600] "GET /~rburke/images/lily-ian-crib.jpg HTTP/1.0" 200 74637
65.55.212.161 - - [12/Dec/2006:03:27:51 -0600] "GET /~rburke/images/lily-ian-check.jpg HTTP/1.0" 200 56516
65.55.212.161 - - [12/Dec/2006:03:27:52 -0600] "GET /~rburke/personal/littleleague/games/game14.html HTTP/1.0" 200 2229
65.55.212.161 - - [12/Dec/2006:03:27:53 -0600] "GET /~rburke/courses/w05/gam224/hwk/ HTTP/1.0" 200 1140
65.55.212.161 - - [12/Dec/2006:03:27:53 -0600] "GET /java/guide/rmi/activation/activation.3.html HTTP/1.0" 200 24143
```

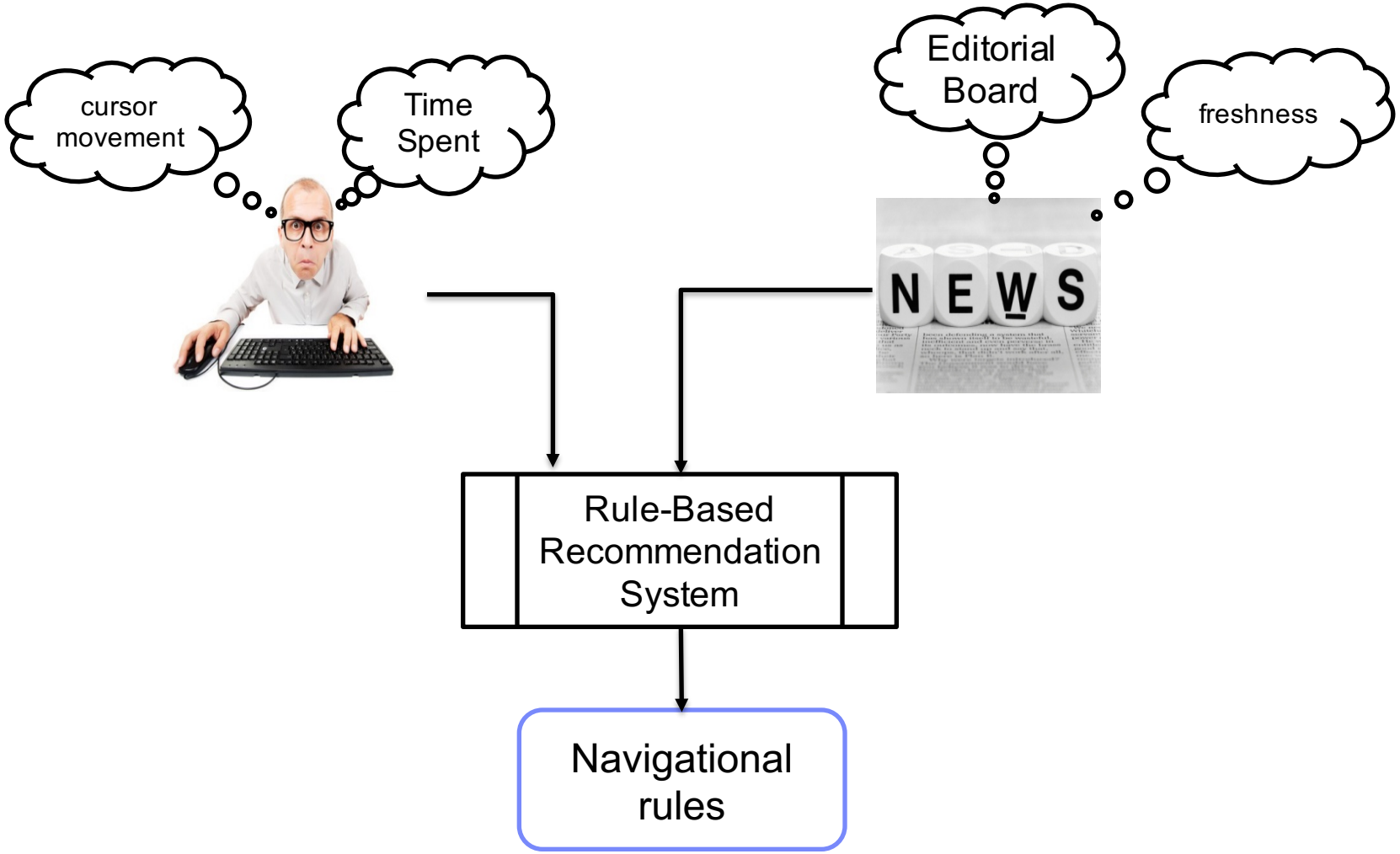
A Rule-Based Recommendation System



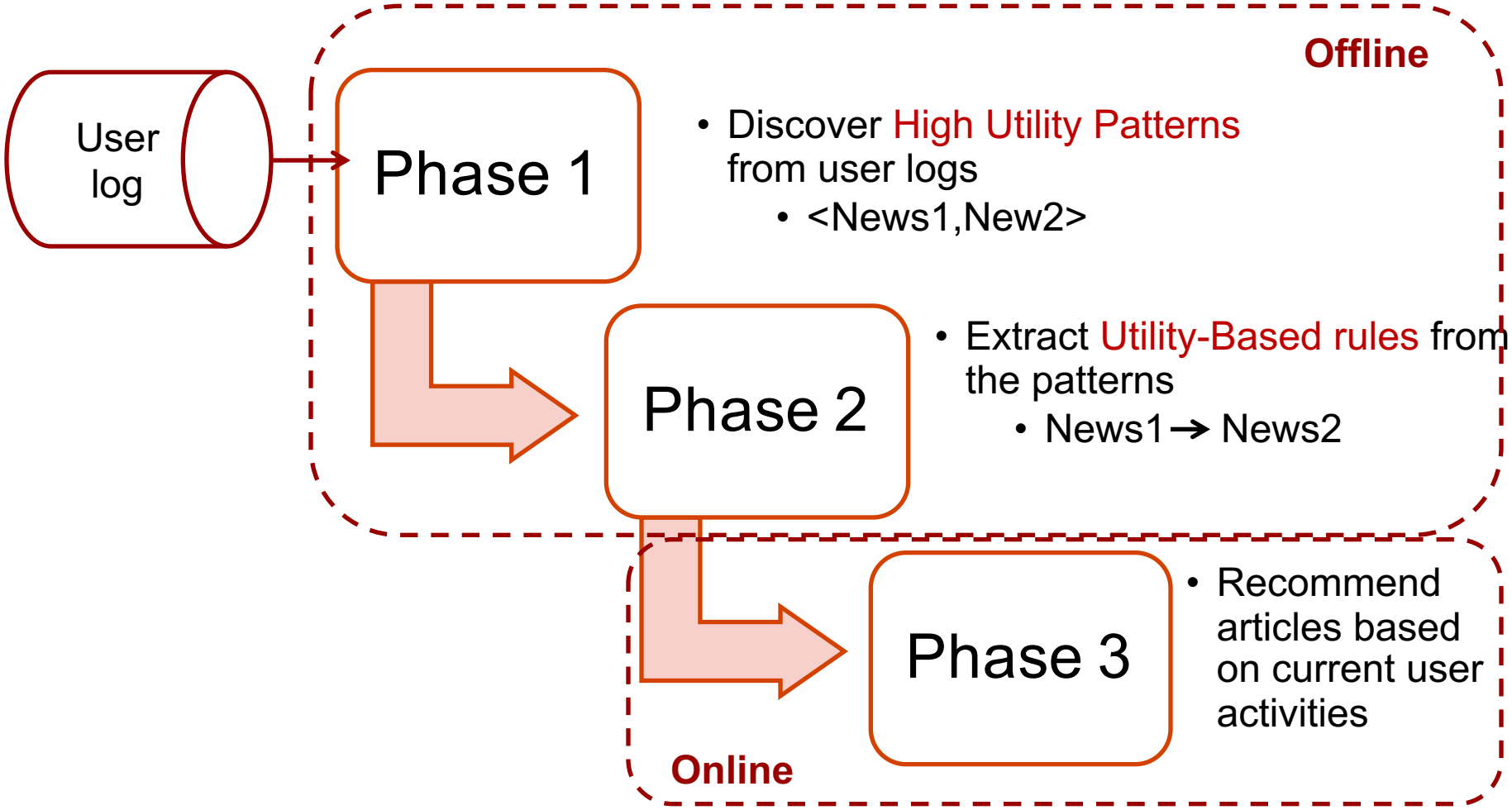
Goal

- 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

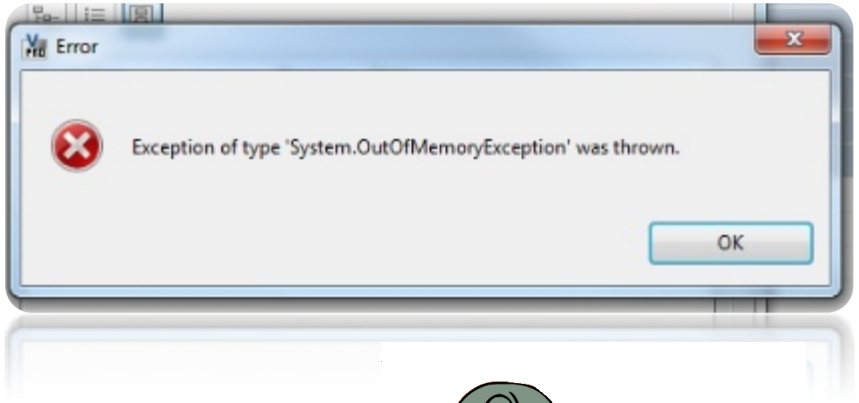


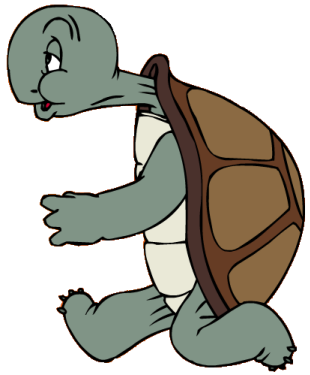
The Proposed Framework



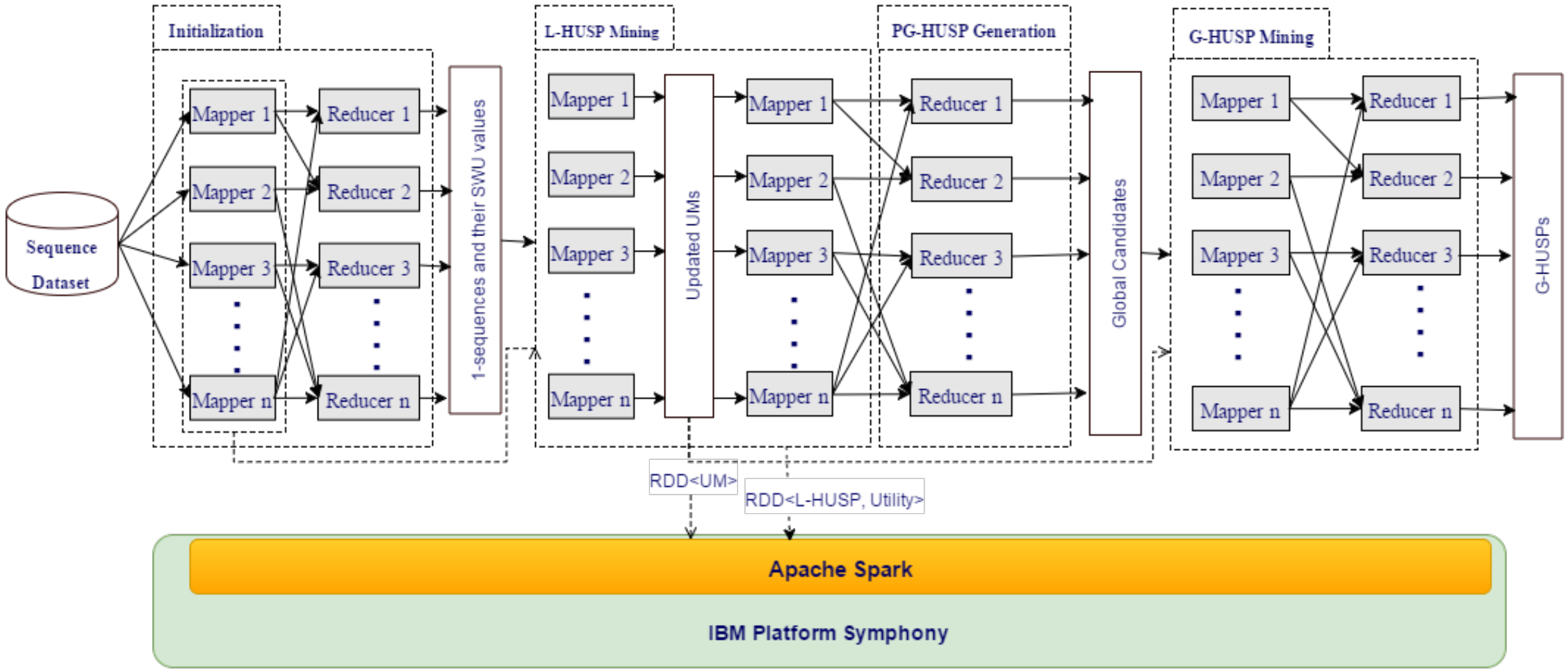
Preliminary Experiment

- 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

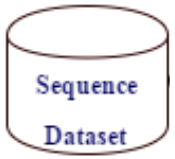




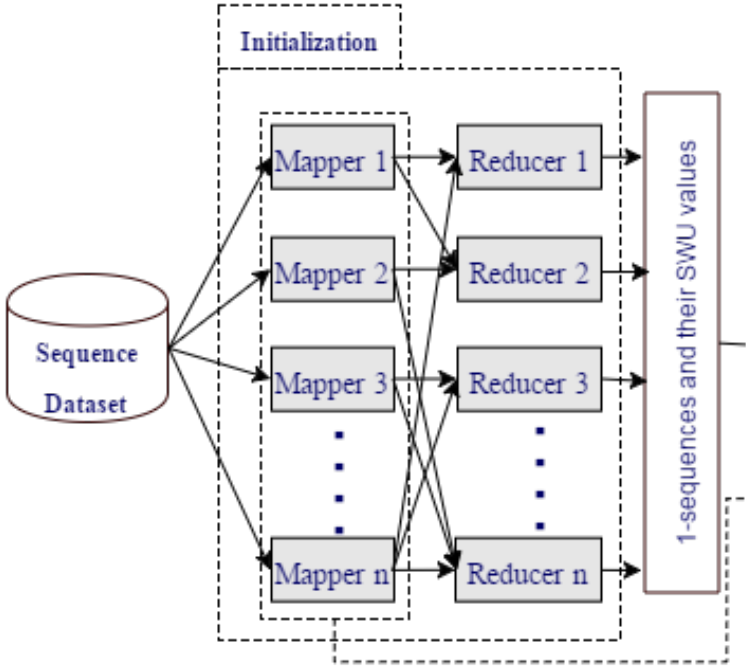
BigHUSP


Initialization
L-HUSP Mining
PG-HUSP Generation
G-HUSP Mining
Sequence Dataset
Mapper 1
Mapper 2
Mapper 3
Mapper n
Reducer 1
Reducer 2
Reducer 3
Reducer n
1-sequences and their SWU values
Updated UMs
Global Candidates
G-HUSPs
RDD<UM>
RDD<L-HUSP, Utility>
Apache Spark
IBM Platform Symphony

BigHUSP

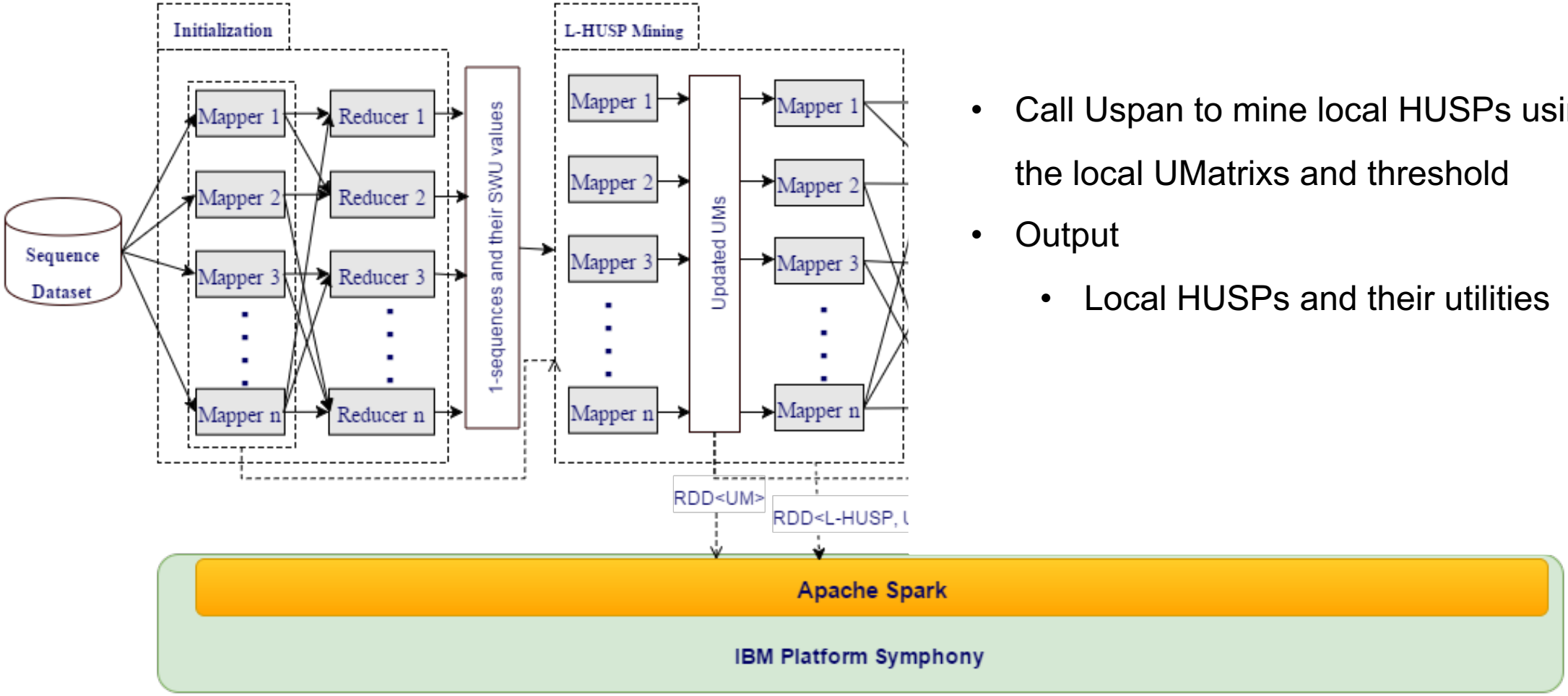


BigHUSP



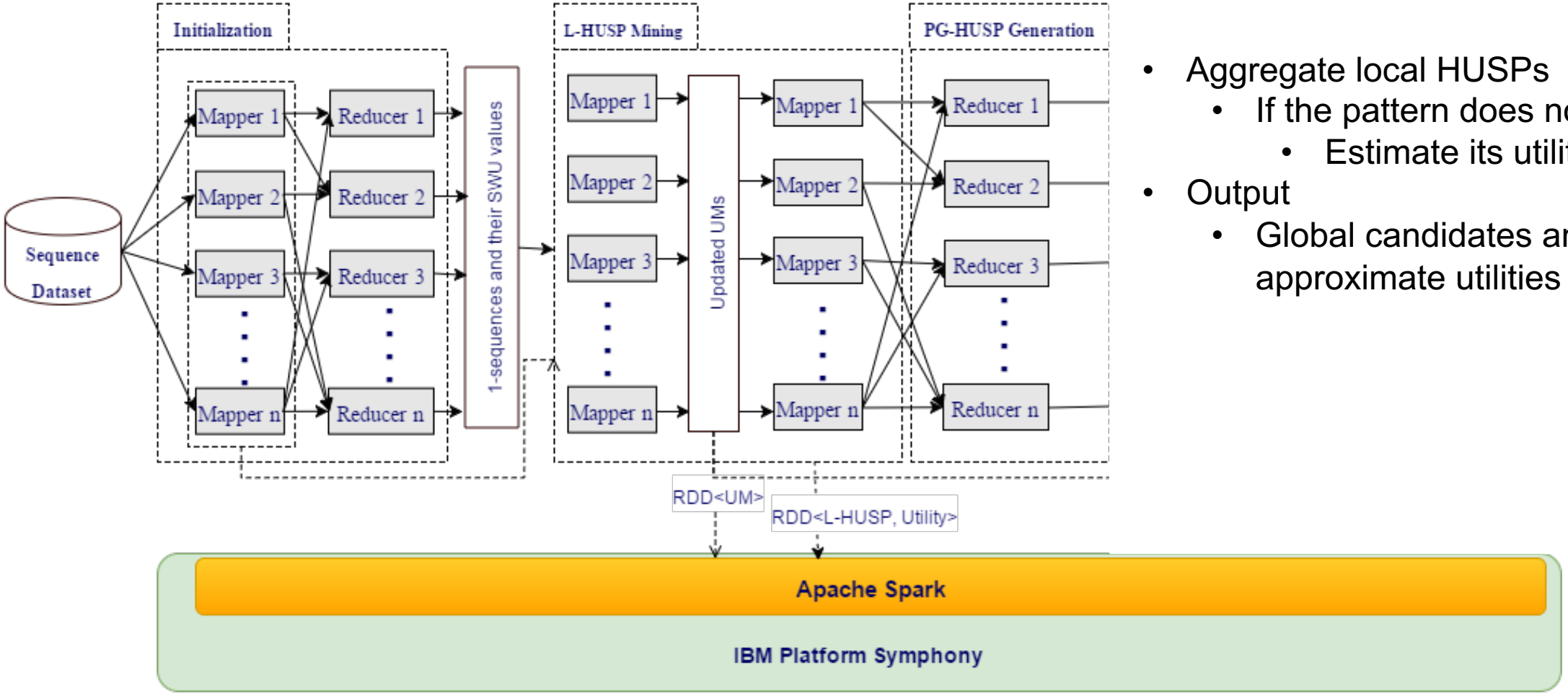
- Partition data
- In each mapper
  - Scan input sequences
  - Construct UMatrixs,
  - Calculate required parameters: threshold, …
  - Remove all blocks for sequences from memory
  - Store Umatrixs in Memory-Disk to re-use

BigHUSP



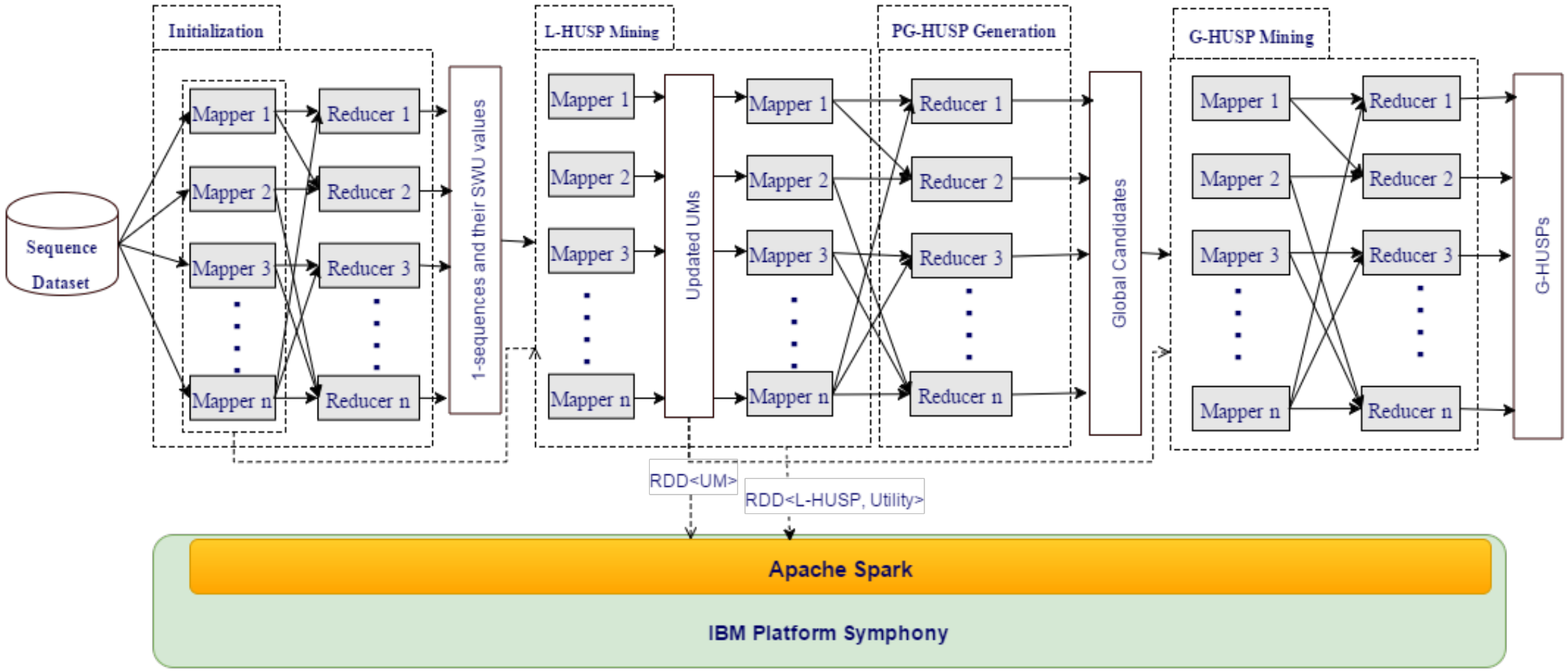
- Call Uspan to mine local HUSPs using the local UMatrixs and threshold
- Output
  - Local HUSPs and their utilities

BigHUSP



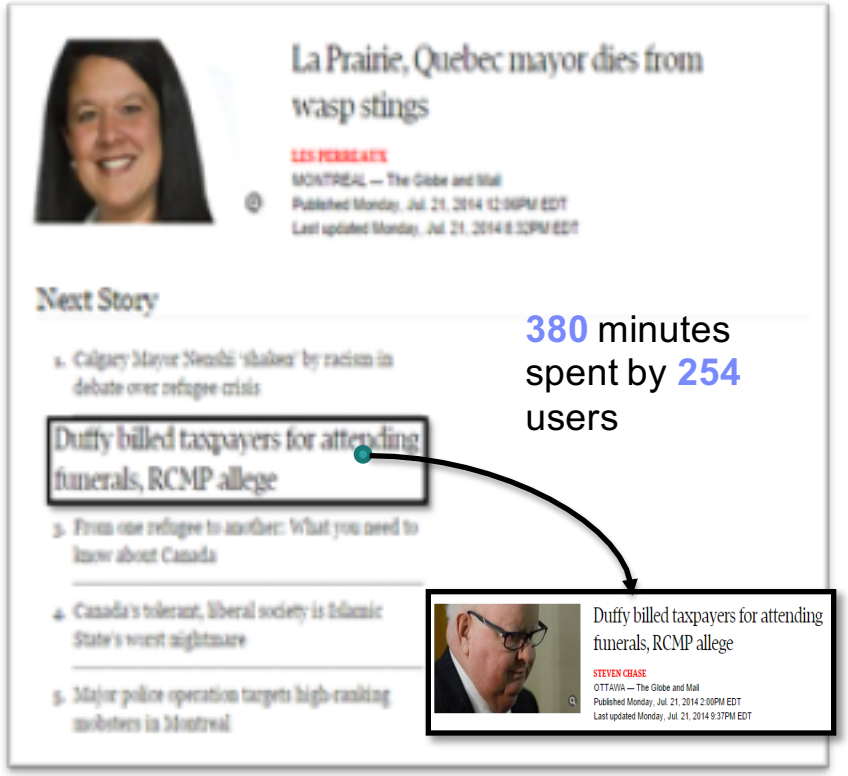
- Aggregate local HUSPs
  - If the pattern does not exist
    - Estimate its utility
- Output
  - Global candidates and their approximate utilities

BigHUSP


Initialization
L-HUSP Mining
PG-HUSP Generation
G-HUSP Mining
Sequence Dataset
Mapper 1
Mapper 2
Mapper 3
Mapper n
Reducer 1
Reducer 2
Reducer 3
Reducer n
1-sequences and their SWU values
Updated UMs
Global Candidates
G-HUSPs
RDD<UM>
RDD<L-HUSP, Utility>
Apache Spark
IBM Platform Symphony

Our recommendations keep user longer

2,537 minutes spent by 107 users



380 minutes spent by 254 users

Experimental Results

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

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

zihayatm@gmail.com

THANKS!

Q1

- 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

Q4

- 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