TRAINS Project and Google Cloud Platform
An Experiment in Big Data Analysis

Laila Daniel

November 28, 2018

Overview of the presentation

1 TRAINS project

2 Big data

3 Apache Spark

4 Google Cloud Platform

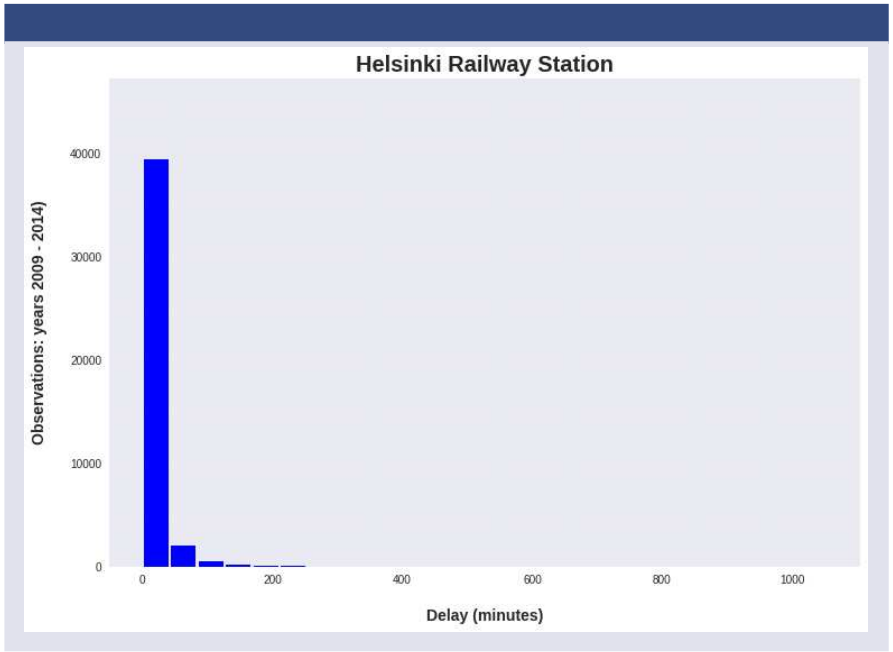
TRAINS Project

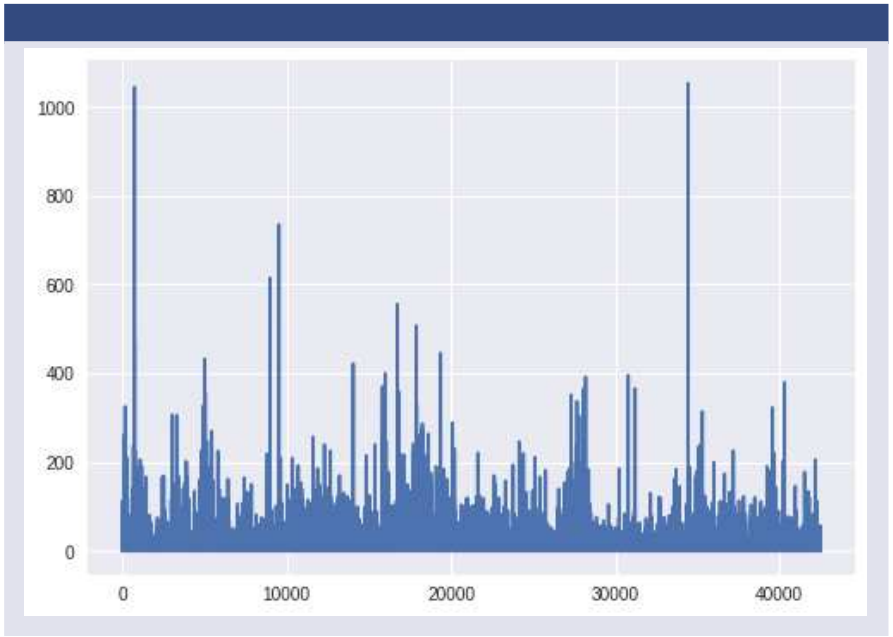
- Aim to predict disruption of rail traffic caused by weather
- Problem restricted to delays due to weather
  - delays of other trains are not considered in prediction
- Timeline: 01/2018-10/2018
- Partners: IL, LiVi, Trafi, VR
- Area: Finland
- Time range: 2 days ahead
- Time step: 1 hour
- Project leader: Roope Tervo

TRAINS Data

- Train and Weather data from 2010-2018
- 514 train stations
- 5 weather station's data within 100 kms of the train station is retrieved and the best weather station's data based on the least number of null values is taken
- Only passenger trains are considered
- Weather observation fetched for every train station for every hour when train has passed the station
- 27132 093 rows of data about 4.6 GB data
- 19 observation variables

TRAINS Data


Helsinki Railway Station
Observations: years 2009 - 2014)
40000
30000
20000
10000
0
0
200
400
600
800
1000
Delay (minutes)


1000
800
600
400
200
0
0
10000
20000
30000
40000

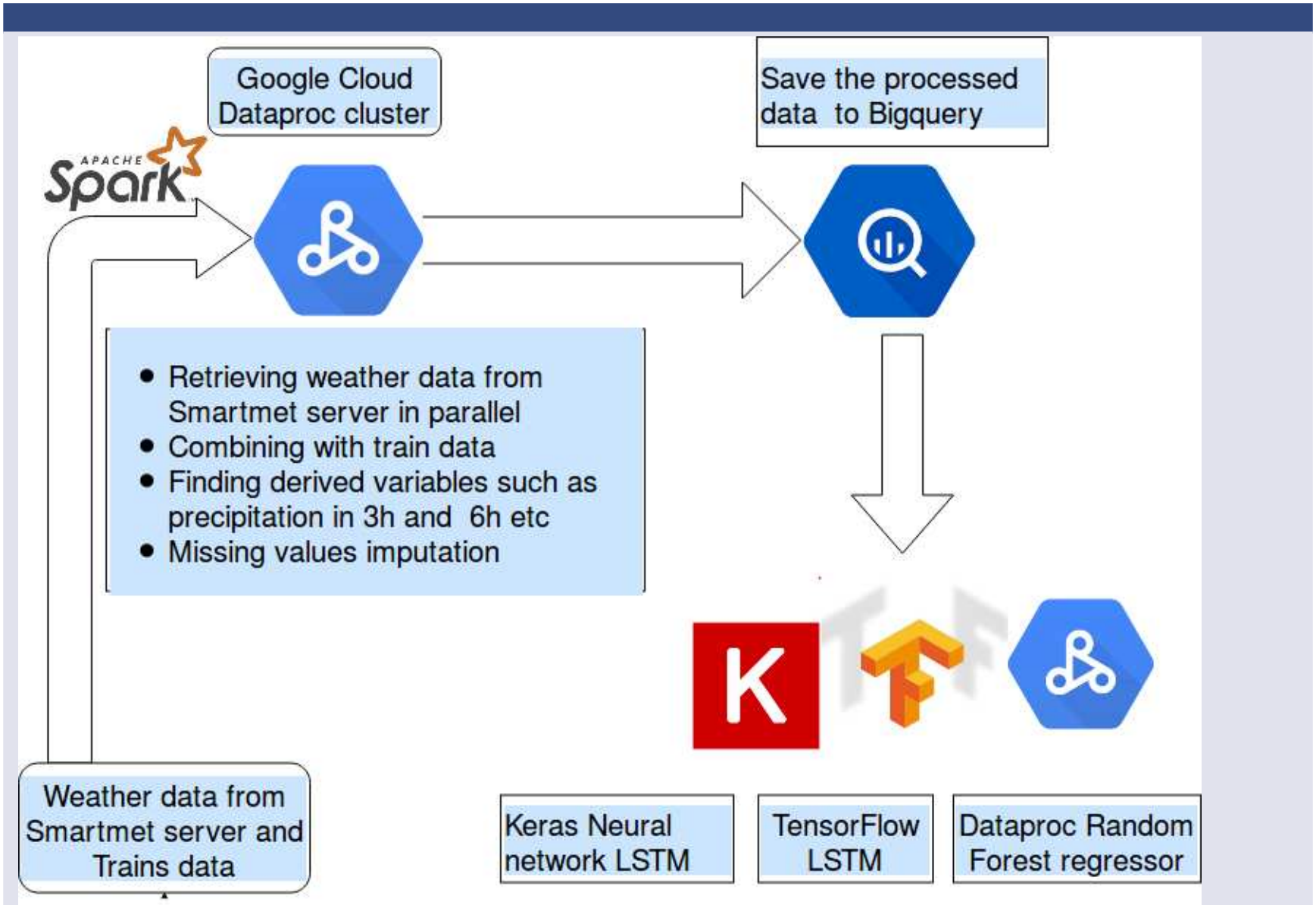
TRAINS Data Variables

- time, train station, train_type, train_count, delay,
- weather station, latitude, longitude, pressure,
- max_temperature, min_temperature, mean_temperature, mean_dewpoint,
- mean_humidity, mean_winddirection, mean_windspeedms, max_windgust,
- max_snowdepth, max_n, min_vis, min_clhb, max_precipitation1h,
- max_precipitation3h, max_precipitation6h, flashcount

TRAINS project: What kind of framework we need?

- Weather data including flash data retrieval
- Weather data and Train data to be combined and saved
- Prediction of delay using machine learning algorithms
  - Random Forest Regression
  - LSTM (Long Short Term Memory) Neural networks
- Integrated view of data which allows unified framework for
  - Data Retrieval
  - Data processing
  - Data analysis
  - Data visualization
  - Saving and sharing the data

TRAINS project and Google Cloud Platform



Big data

- Large volume of data
- Structured, semi-structured and unstructured data
- Structured data
  - clearly defined data types resides in relational databases
- Unstructured data
  - data has internal structure but is not structured via pre-defined data models or schema
  - Textual or non-textual
  - Human- or machine-generated
  - Eg: Text, emails, social media data, satellite images, sensor data
- Real-time and non real-time data
- Quality of the data captured varies widely

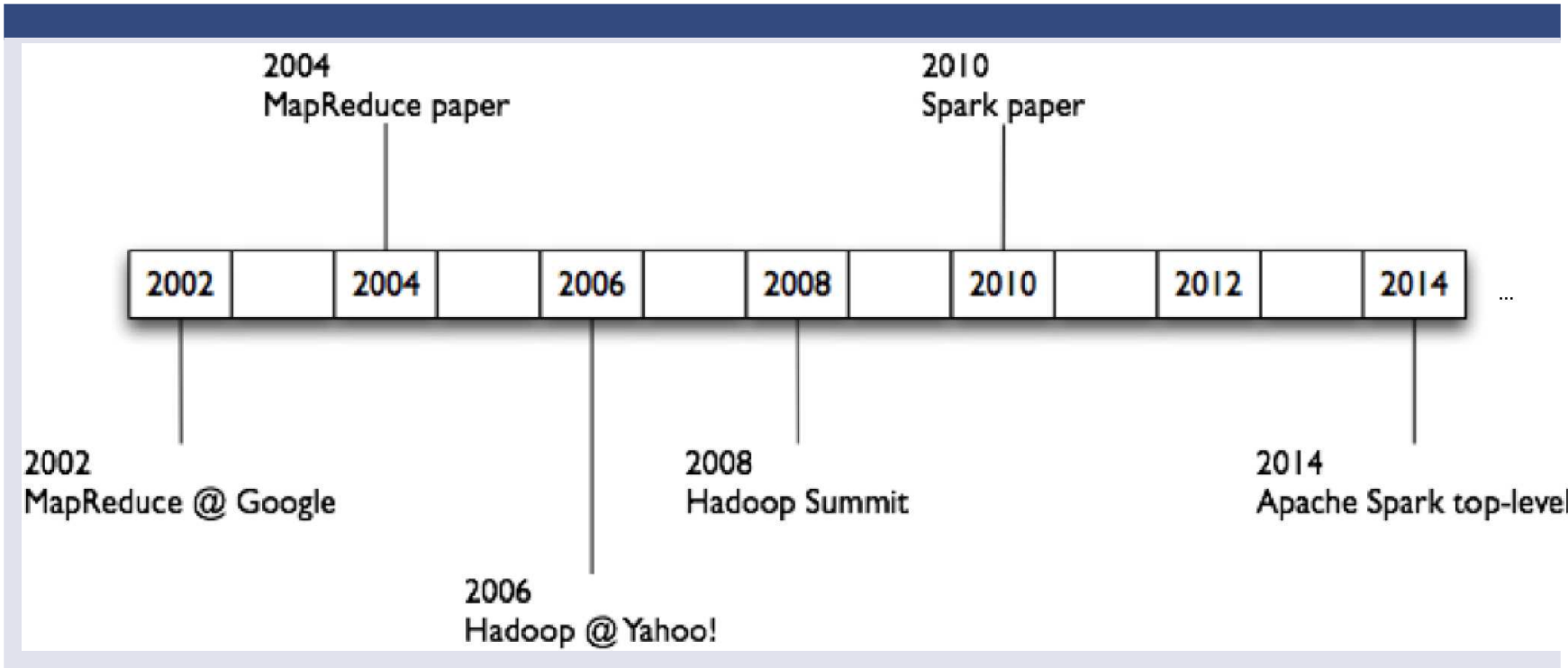
What are the problems with Big data?

- One machine cannot process large amounts of data
- Traditional analysis techniques
  - Shell scripts (grep, awk, sed), Python pandas or R
  - These tools run on a single machine
  - How to store the big data?
  - How distribute the work?
  - How to deal with failures and slow machines?
  - What kind of analysis tools are needed?

Modern cluster computing environment

- Based on less expensive, consumer grade hardware, desktop like hardware, which makes it easy to grow in capacity.
- Complex software is used to automatically handle
  - distributing the data
  - problems due to node failures and slow machines
  - analyzing the data
- Move computation to data
- Data-centric computation

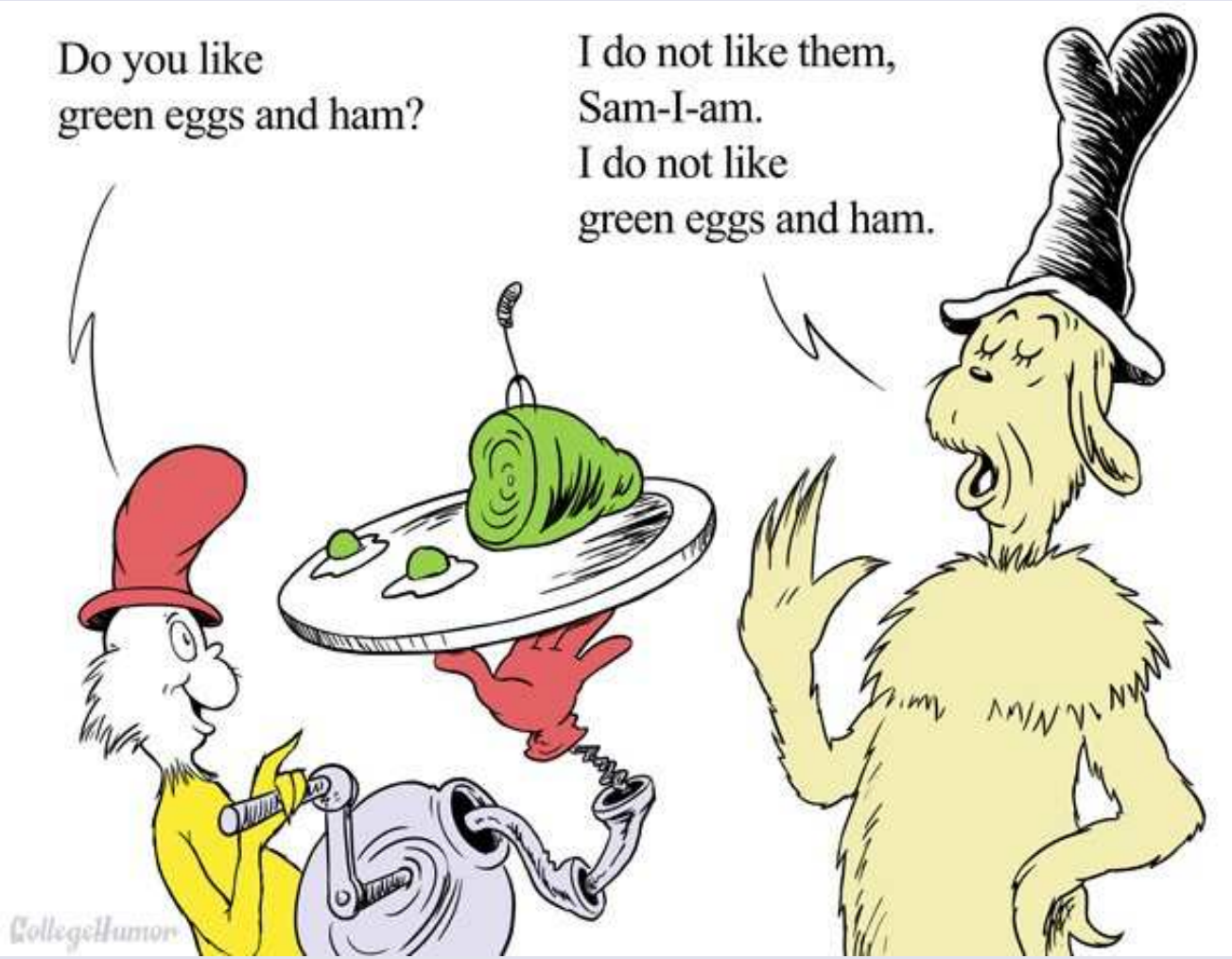
History of Modern cluster computing environment



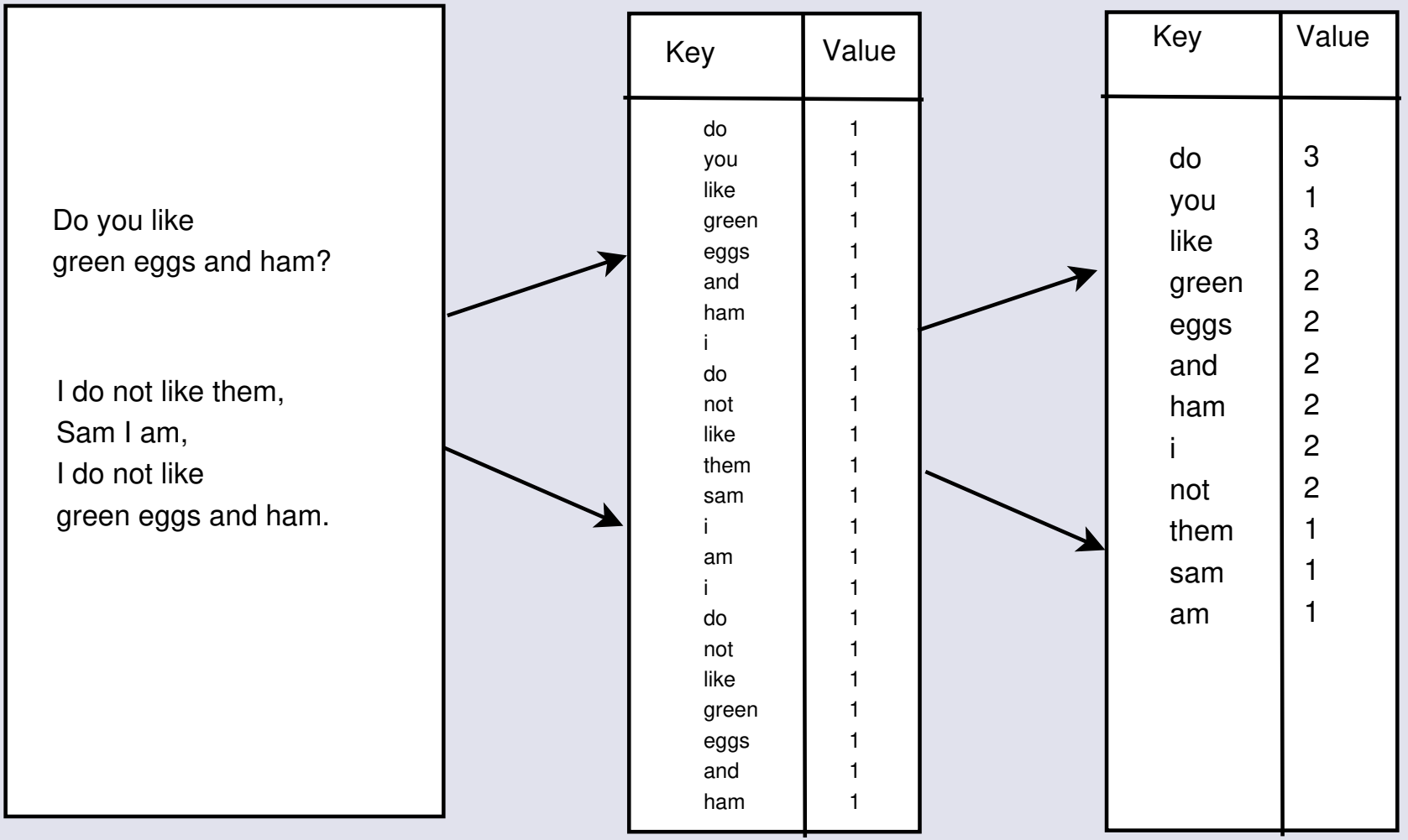
MapReduce

- A programming model developed by Google
- A framework for processing parallelizable problems across large datasets using a large number of computers
- Map: map function which transform a small unit data into some number of key/value pairs
- Reduce: reduce function to merge the values (of the same key) into a single result
- Map and Reduce reads from the disk and writes to the disk

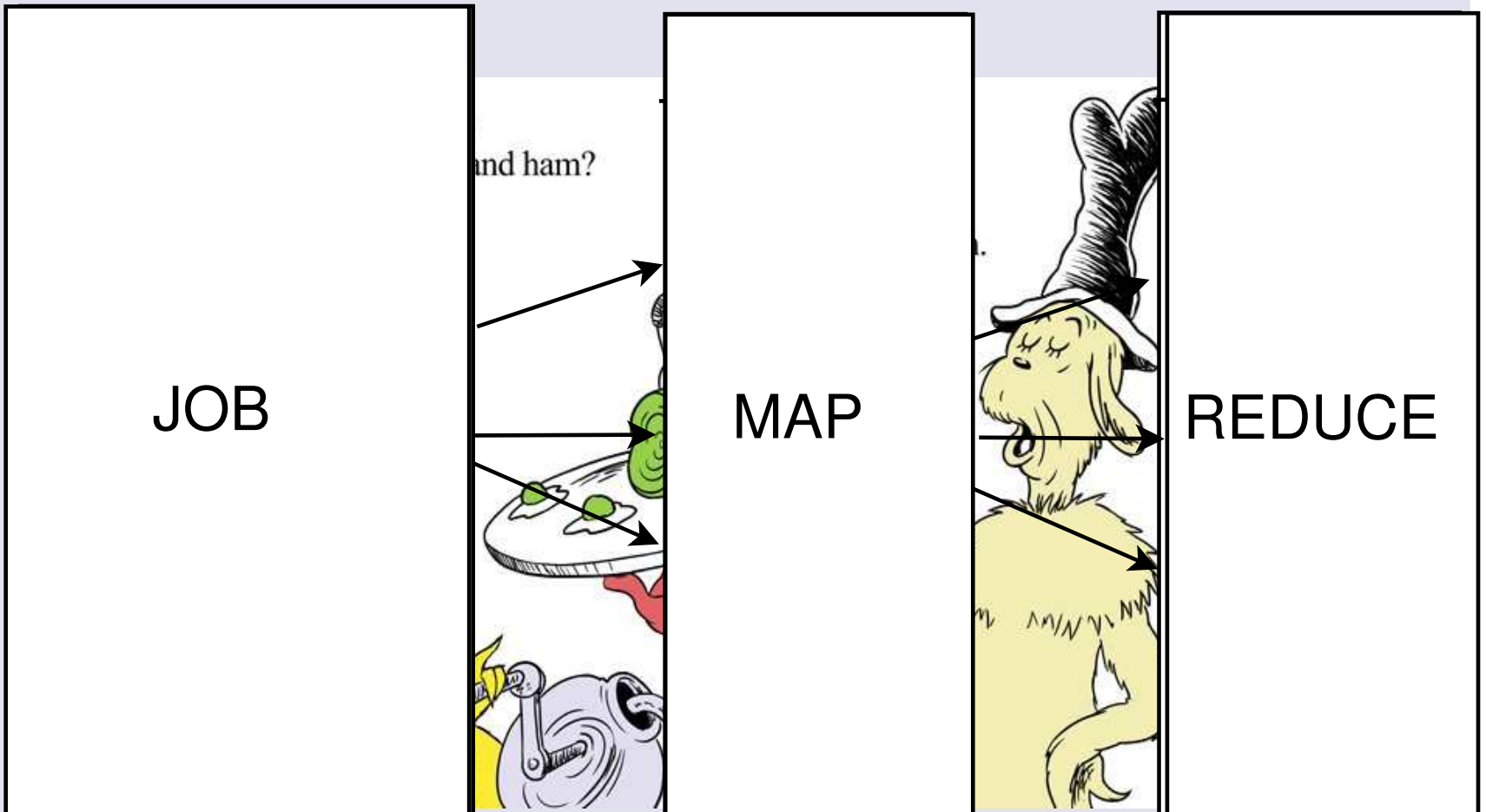
Word Count - Hello, World! program of Big Data



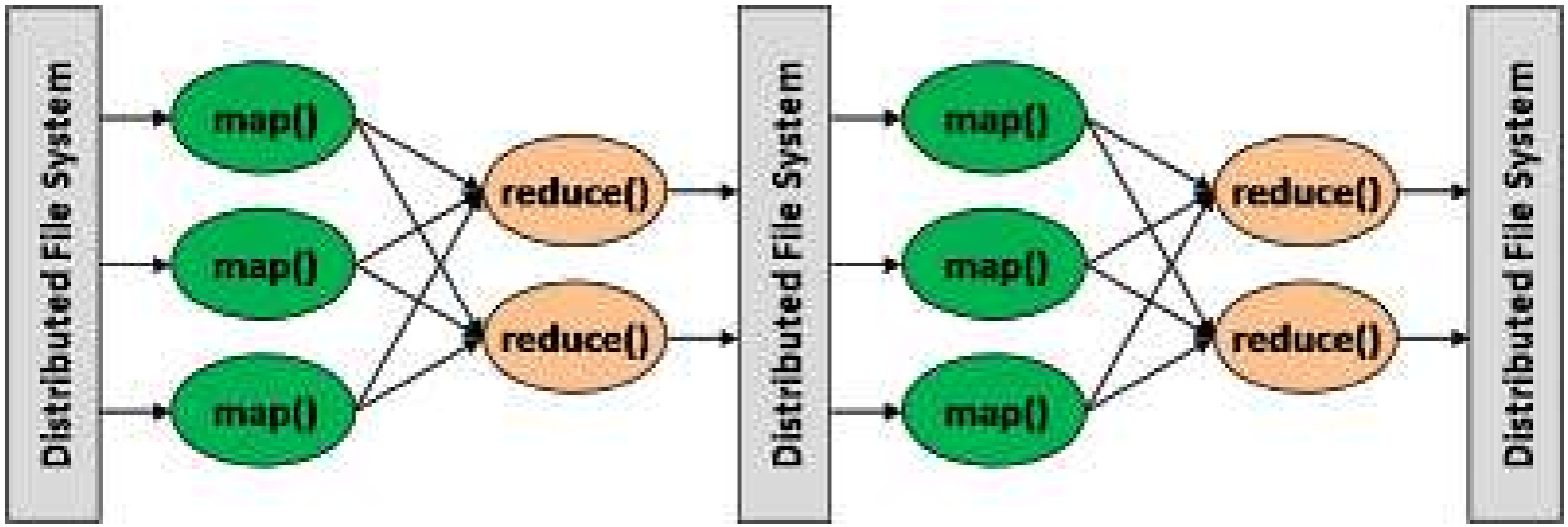
Word Count


Do you like
green eggs and ham?
I do not like them,
Sam I am,
I do not like
green eggs and ham.
Key
Value
do 1
you 1
like 1
green 1
eggs 1
and 1
ham 1
i 1
do 1
not 1
like 1
them 1
sam 1
i 1
am 1
i 1
do 1
not 1
like 1
green 1
eggs 1
and 1
ham 1
Key
Value
do 3
you 1
like 3
green 2
eggs 2
and 2
ham 2
i 2
not 2
them 1
sam 1
am 1

Word Count

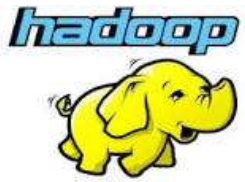


MapReduce



- https://dzone.com/articles/how-hadoop-mapreduce-works

Apache Hadoop



- Open source software framework for distributed storage and distributed processing of very large data sets in computer clusters built from commodity hardware
- Inexpensive alternative to big data analysis
- Hadoop is written in Java
- Hadoop's HDFS is a distributed file system is designed to handle large files with sequential read/write operation.
- Programming model is MapReduce

MapReduce and Hadoop

- Map initially reads from the disk and writes to the disk
- Reduce reads from the disk and writes to the disk
- Disk I/O is very slow
- MapReduce supports only Batch processing
- So for iterative jobs and online processing MapReduce performs poorly
- Difficulty in creating "map" and "reduce" functions

Nature of Big Data processing

- Big Data applications need to combine different processing types
- MapReduce-like jobs, SQL queries, Interactive machine learning
- Hadoop MapReduce framework created many specialized engines for different processes
- Specialized engines increase complexity and inefficiency
- Some applications cannot be expressed efficiently in any engine

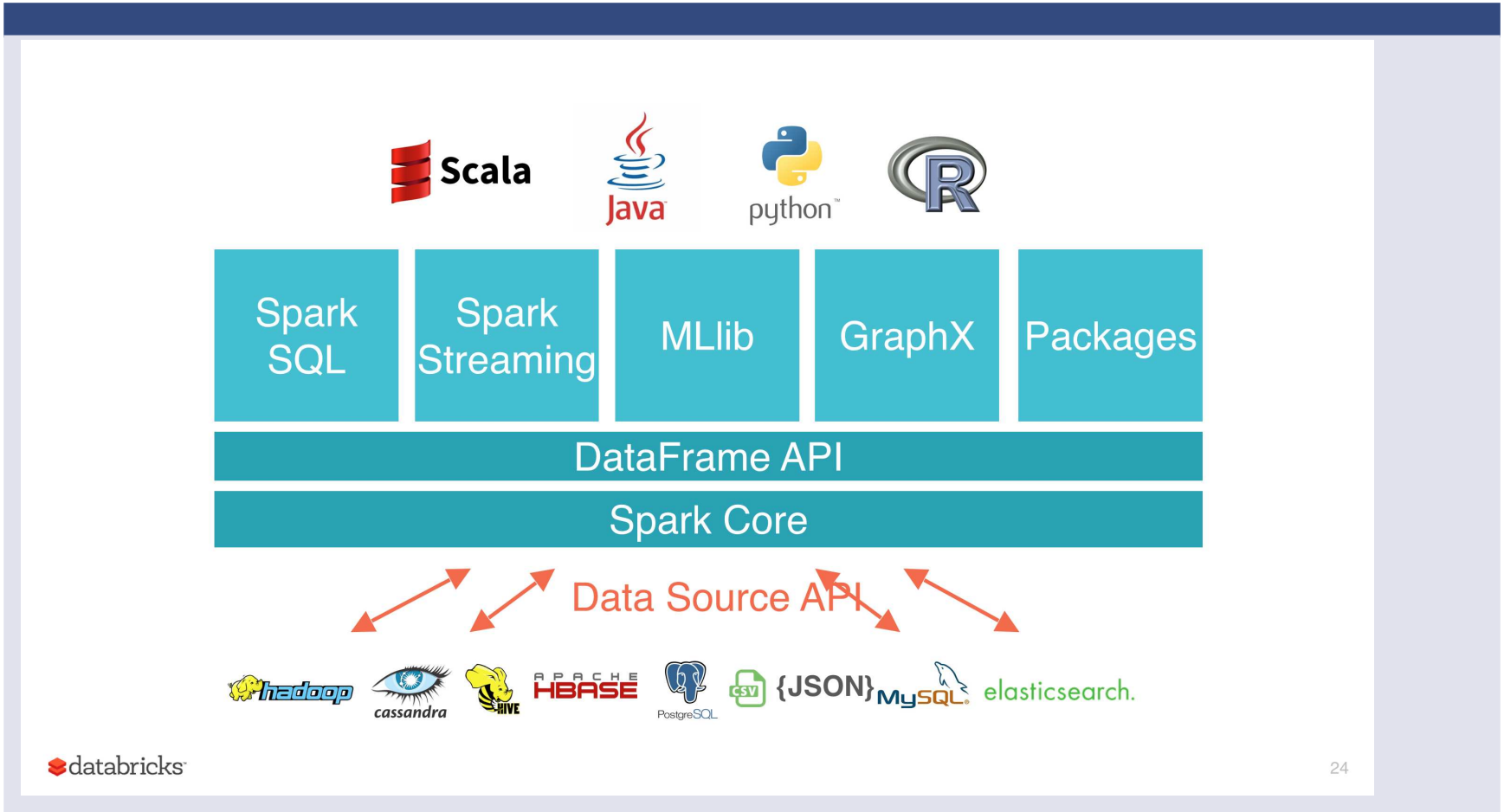
Apache Spark

- Distributed computing framework
- Written in Scala, built on JVM
- A unified engine for SQL, Machine learning, Streaming and Graph processing
- Extended MapReduce framework
- Originated from Matei Zaharia's PhD work University of California, Berkeley
- First deployed in 2010, grown to +1000 contributors, thousands of deployments

Apache Spark

- Supports batch, interactive and stream processing
- Processing done in memory and reduced Disk I/O
- 100x faster than Hadoop MapReduce
- Integration to many data sources, text, JSON, mySQL, Hadoop, Amazon EC2, Google cloud ...
- Scala, Java, Python and R interfaces
- Has an interactive Spark shell

Apache Spark



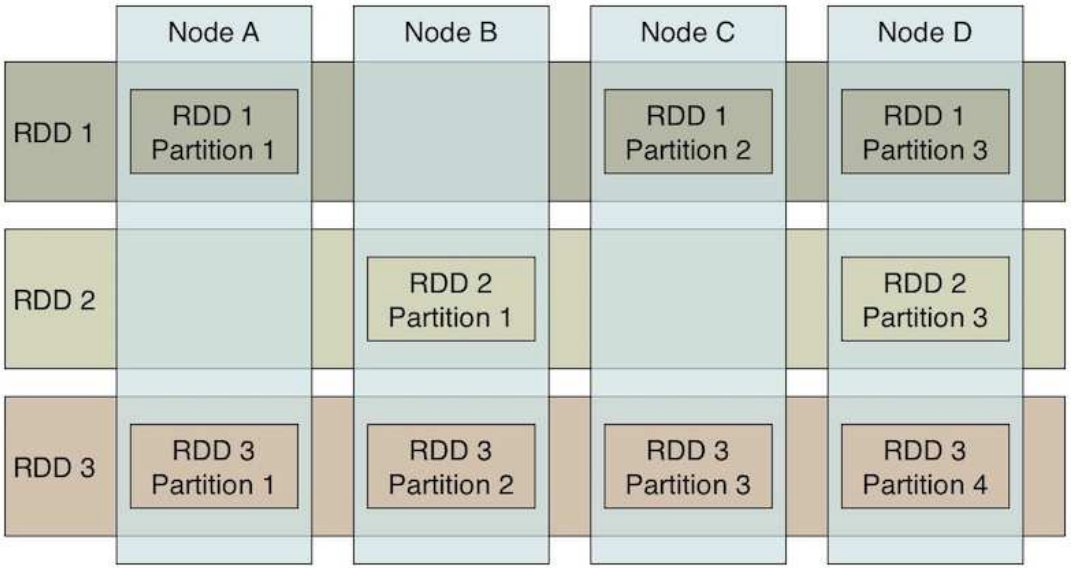
Why Spark so powerful?

- Resilient Distributed Dataset (RDD)

Resilient Distributed Dataset (RDD)

- Fundamental data sharing abstraction in Spark.
- RDDs provide data sharing among computations
- An immutable, fault tolerant collections of objects
- Partitioned across clusters and operated in parallel on different nodes
- With RDD, Spark captures a wide range of processing workloads, such as SQL, machine learning, streaming and graph processing

Resilient Distributed Dataset (RDD)



- Creating an RDD
  - parallelizing an existing collection in your driver program
  - referencing a dataset in an external storage system like HDFS or cloud
- Operations on RDD
  - Transformations: map, filter, groupBy, ...
  - Actions: reduce, count, collect, ...

Word Count in Spark

Word Count in Scala

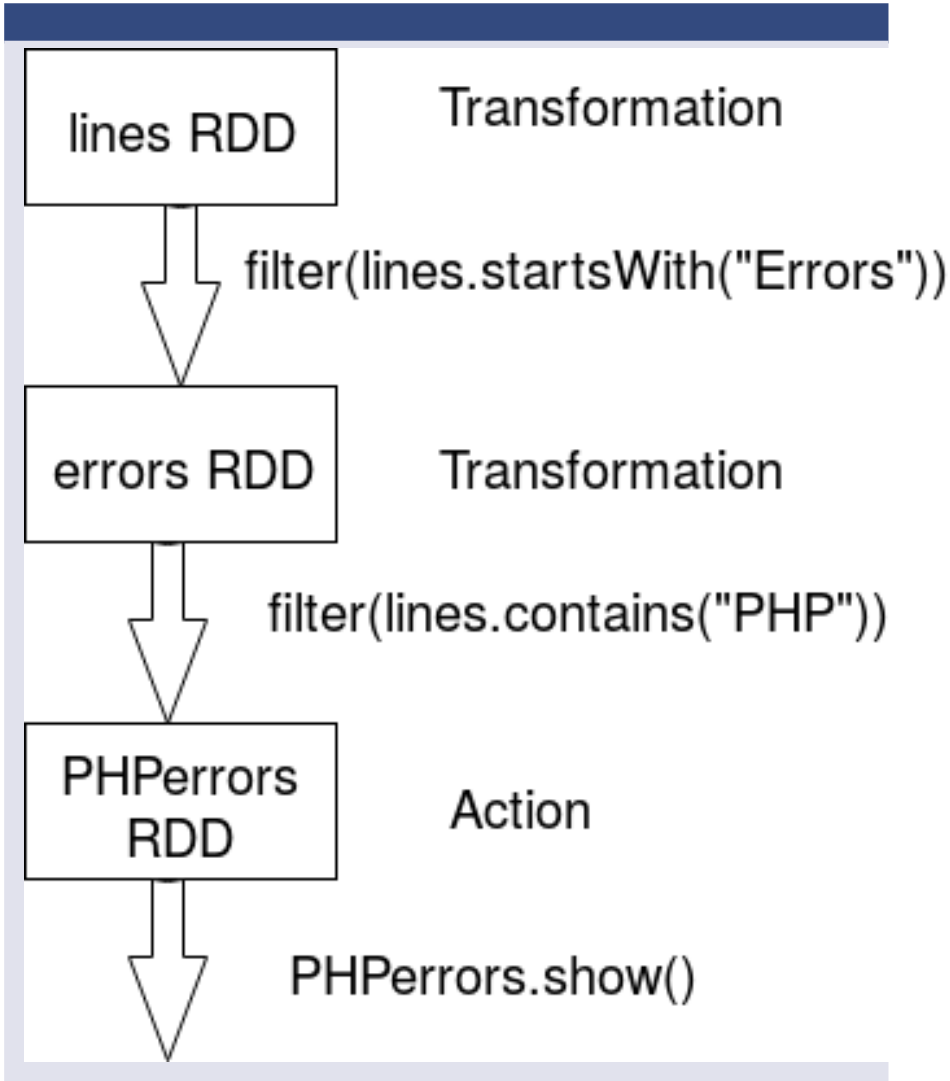
```
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

Word Count in Python

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

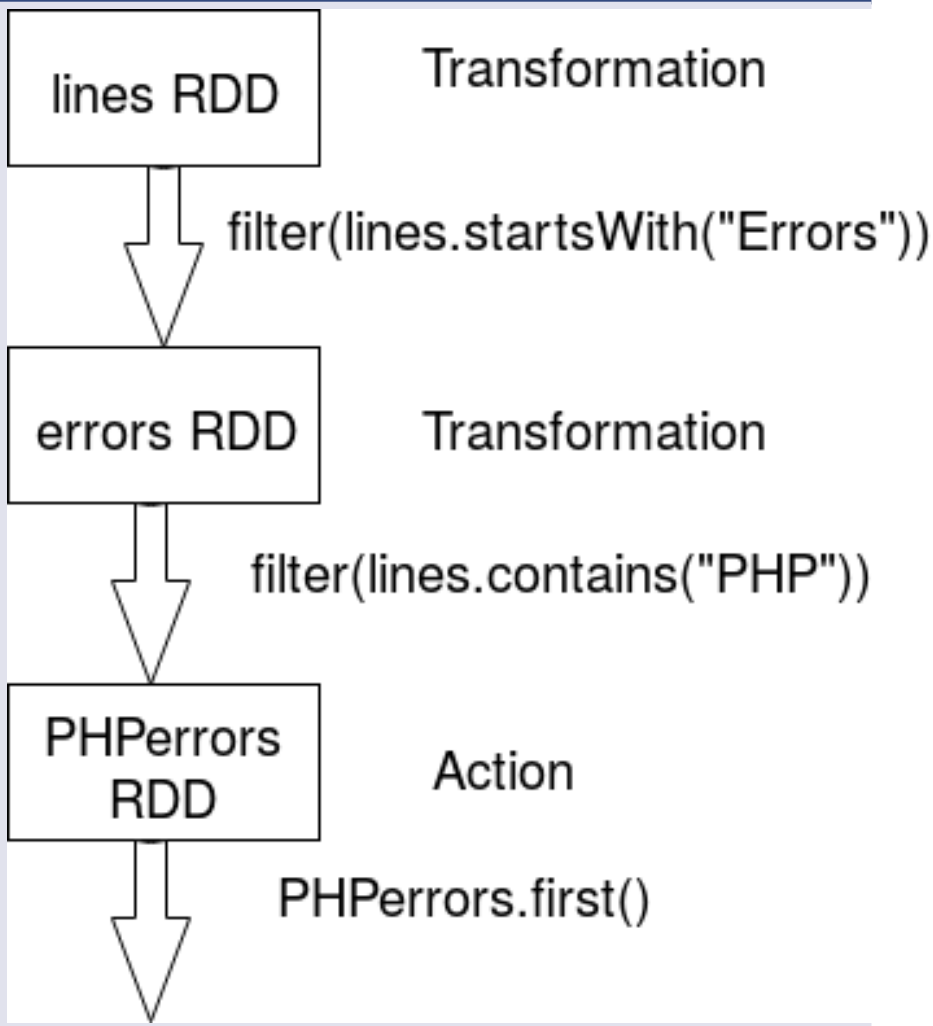
Lineage Graphs/ Direct Acyclic Graphs (DAG) for RDD

- RDD Lineage or RDD dependency graph is a graph of all the parent RDDs of a RDD
- Built as a result of applying transformations to the RDD and creates a logical execution plan
- A RDD lineage graph is a graph of all transformations need to be executed after an action has been called



Lazy evaluations

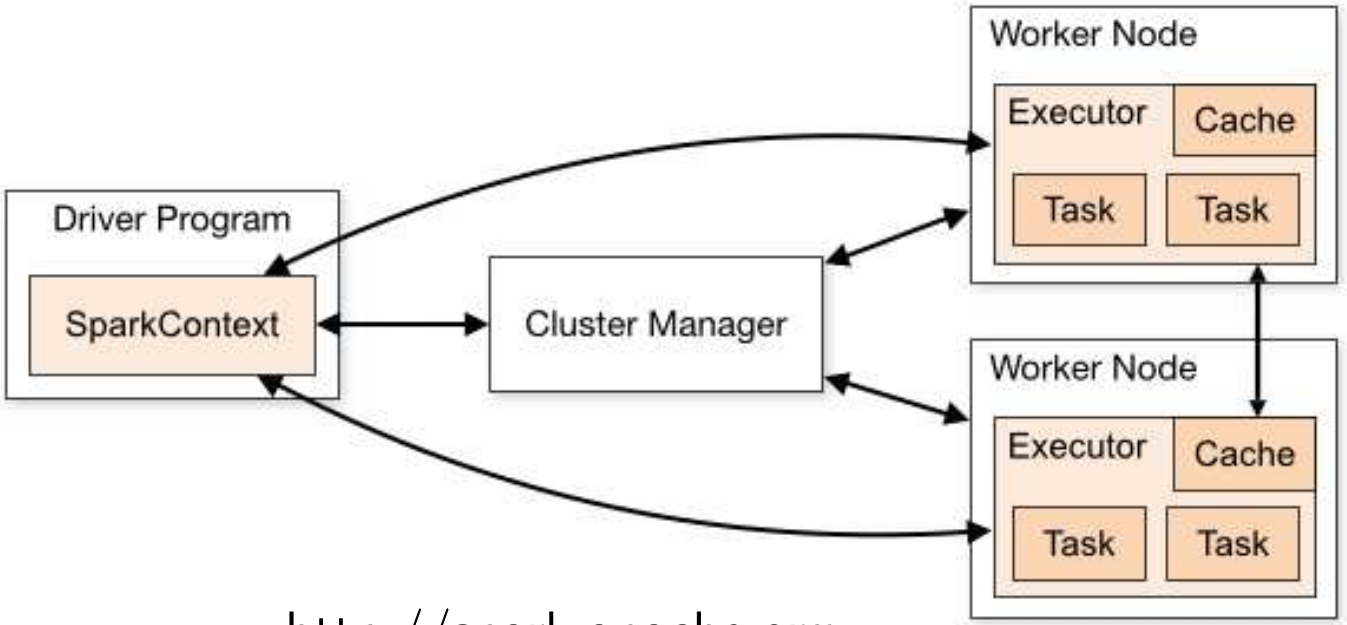
- Spark Transformations are lazily evaluated
  - Spark executes all the transformations based on lineage graph/DAG only when an action is called
  - Spark can make many optimization decisions after it had a chance to look at the DAG in entirety
  - No need to materialize intermediate datasets in memory



Apache Spark

- Data Sharing among computations
- Scalable
- Fault-tolerant
  - Lineage based recovery
- Optimized evaluation
  - Lazy evaluation
  - Spark SQLs Catalyst Optimizer
- High level libraries
  - SQL and Dataframes, Machine learning, Streaming, Graphicx
- Combining processes using pipelines
- Well documented Apache Spark

Spark Clusters



http://spark.apache.org

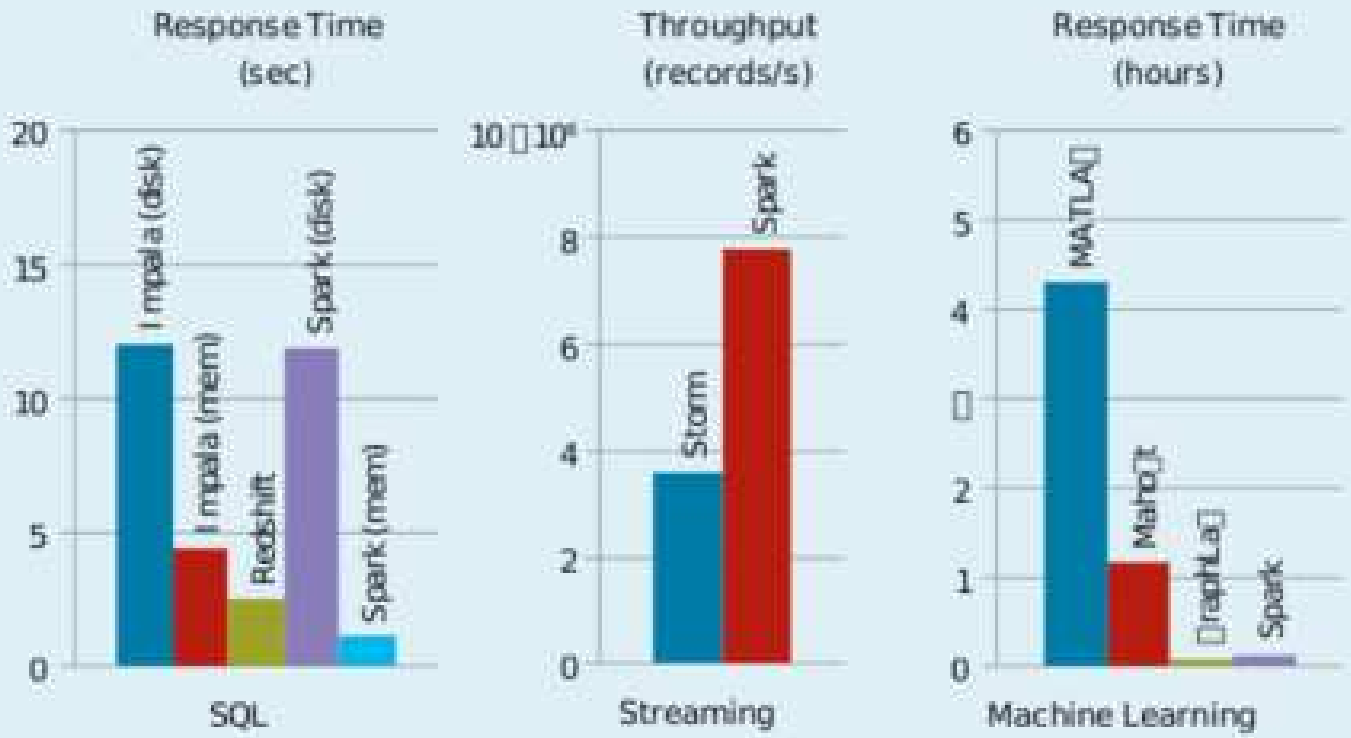
- Master-Worker architecture
- A central coordinator *Driver* coordinates with many distributed *Workers/Executors*
- Driver and each of the executors have their own Java processes

Spark Clusters

- Driver
  - Process where the main method runs
  - Converts the user program to tasks and schedules the task to executors
- SparkContext
  - Spark context sets up internal services and establishes a connection to a Spark execution environment
  - Driver uses SparkContext to communicate with the Cluster Manager
- Cluster Manager
  - Allocates resources
  - Hadoop YARN, Apache Mesos, Kubernetics

Spark Performance

Figure 6. Comparing Spark's performance with several widely used specialized systems for SQL, streaming, and machine learning. Data is from Zaharia[34] (SQL query and streaming word count) and Sparks et al.[17] (alternating least squares matrix factorization).



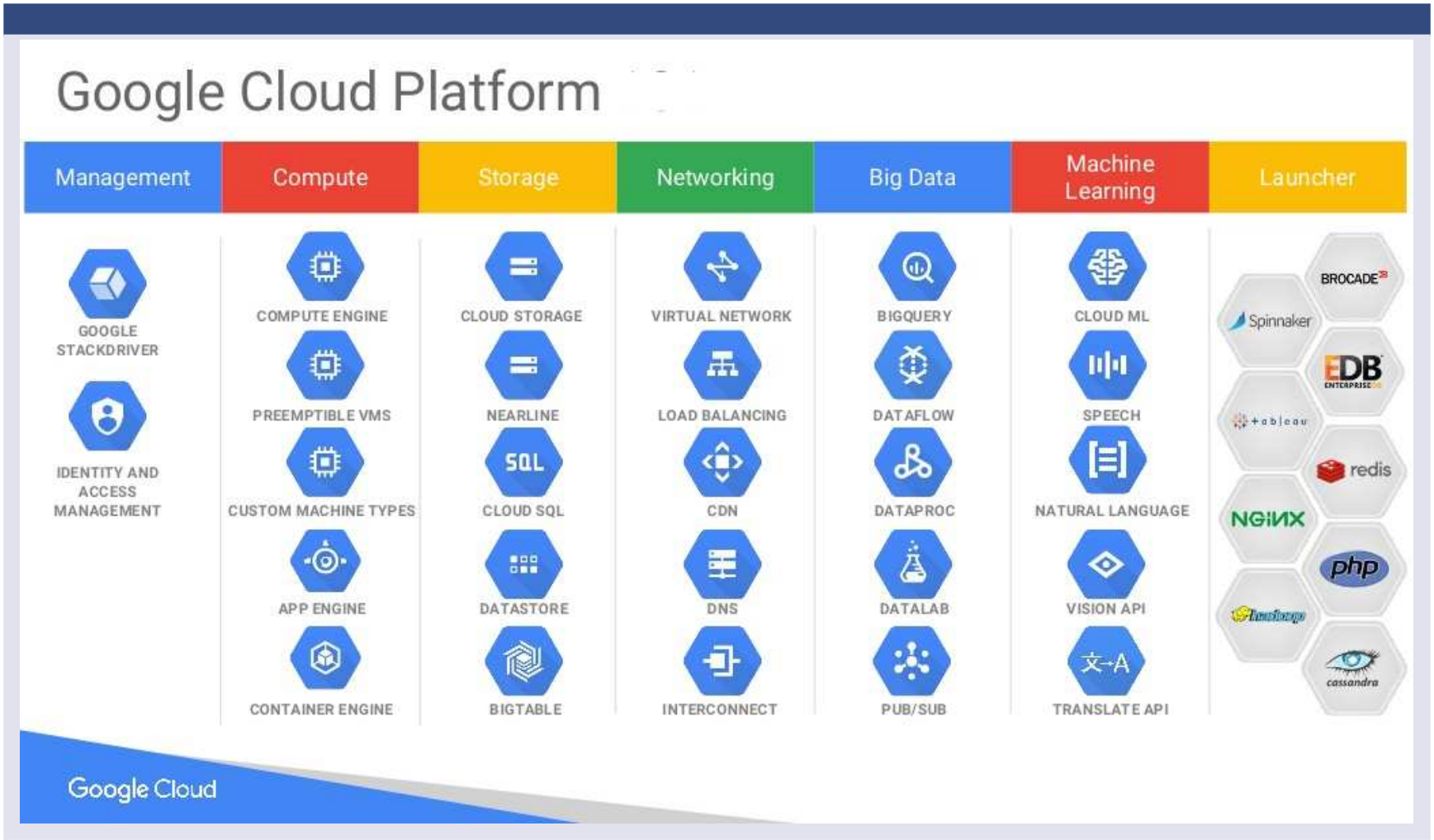
Ref: Apache Spark CACM paper

When to use Scala, Python or R

- Scala
  - Spark is implemented in Scala
  - can understand and modify what Spark does internally
  - allows to access the latest features
  - Scala is fast
- Python (PySpark)
  - general purpose, easy to understand
  - Data science libraries and Visualization tools
- R (SparkR, Sparklyr)
  - large number of packages on statistical analysis and data visualization.

Databrick's Apache Spark Survey 2016

Google Cloud Platform (GCP)



Components of our data processing engine in TRAINS

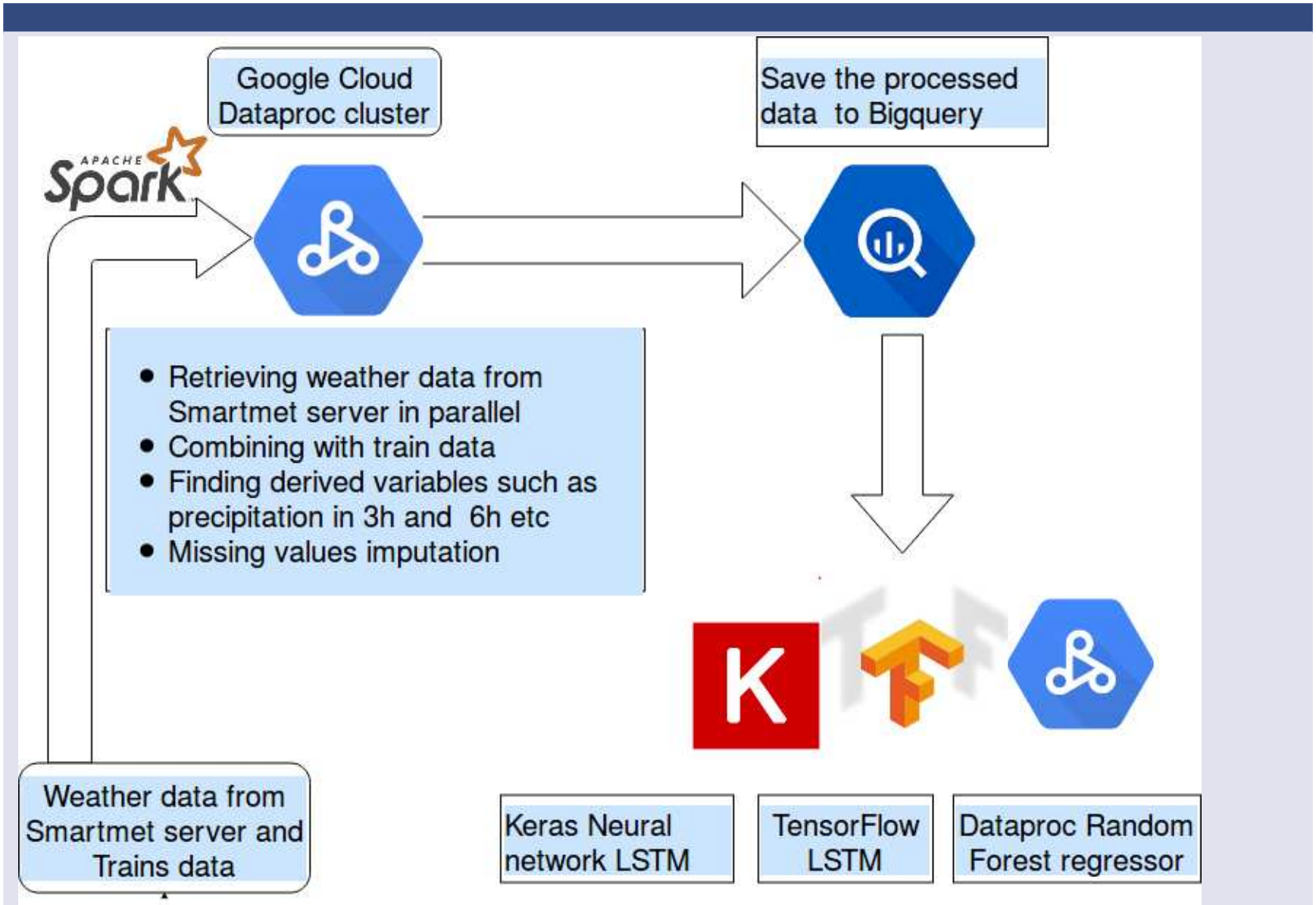
- Smartmet Server used as Weather and Flash data repository
- Google Cloud's Dataproc Apache Spark cluster for retrieving data in parallel
- Google BigQuery for saving the data
- Apache Spark to implement Random Forest regression
- TensorFlow and Keras to implement LSTM
- Google Colab notebook and Jupyter notebooks for intermediate code development and testing

- Account in Google cloud
  - $200 free credit or one year
- Dataproc
  - Apache Spark Cluster
  - Maximum of 8 virtual machines in free credit
- Storage
  - Buckets accessed as gs://...
  - Saving programs, data and notebooks
- BigQuery
  - RESTful web service
  - Interactive analysis of massively large datasets
- Datastudio
  - Visualization of data
  - Can generate reports
- Entry point to GCP

Google Cloud Platform - contd

- Deep learning libraries
  - TensorFlow
  - Keras
- Google Colab
  - Jupyter notebook environment
  - free cloud service that supports free GPU
- Tour to Colab

TRAINS project and Google Cloud Platform

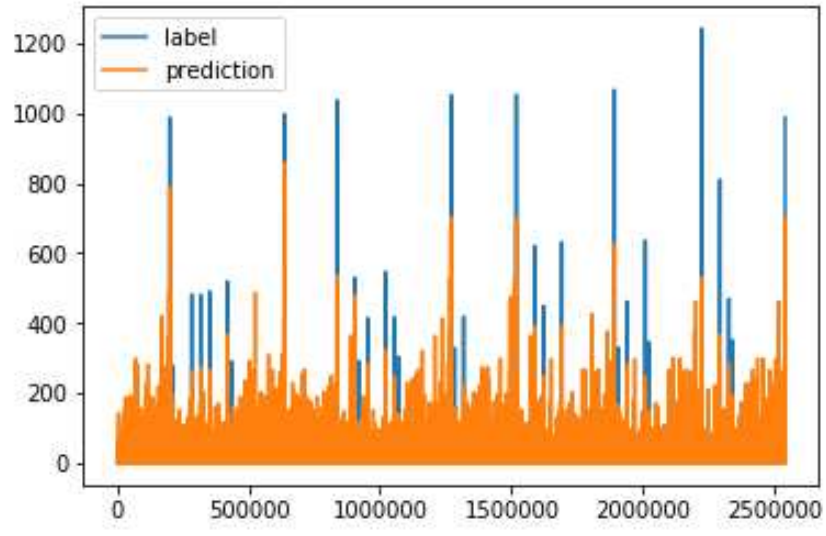


TRAINS project Results

- Data retrieval and preprocessing of data using Apache Spark
  - Weather data and flash data for 514 stations for the years 2009 - 2018
  - Dataproc cluster consists of 8 CPUs with 2 cores each
  - With producer Opendata took about 8 hours
  - With producer fmi took about 3 days
  - With producer fmi we could retrieve one month data in a request
  - With producer flash, we retrieve the flash data for June to August with a single worker

TRAINS project Results - contd

- Prediction for delay
  - Linear Regression (LR): Did not converge
  - Random Forest Regression (RFR): RMSE 13.55 and MAE 4.52
  - Long Short Term Memory (LSTM): RMSE 11.38, MAE 7.87
  - Results are trade off between MAE and RMSE
  - MAE tells about overall accuracy while RMSE gives more weight on large mistakes
  - Rail traffic operation center wants to emphasize large delays



Future work

- More Refined model for TRAINS data
  - Dependencies on derived variables
  - Handling of missing data
  - Distributed Keras in Dataproc
  - Use ML engine in GCP
- Spark for other Big data - computationally intensive problems
  - Road delays and weather
  - Air traffic and weather
  - Ensemble calibration
  - Nowcasting

Thank You

- To Roope Terve for
  - Introducing me to the TRAINS project
  - Discussions, clarifications
  - Implementation of the TRAINS project
  - Beautiful coding style and programs
- To Jussi Ylhäisi for
  - All ongoing discussions
- To you all for
  - Your Attention
- Questions?

References

- Apache Spark: A Unified Engine for Big Data Processing, Communications of the ACM (CACM), November 2016
- Apache Spark Overview
- Apache Spark - databricks
- Coursera, Edx, Udemy, ... courses on Apache Spark
- Google cloud platform, just google
- Coursera and Edx courses on Google cloud platform