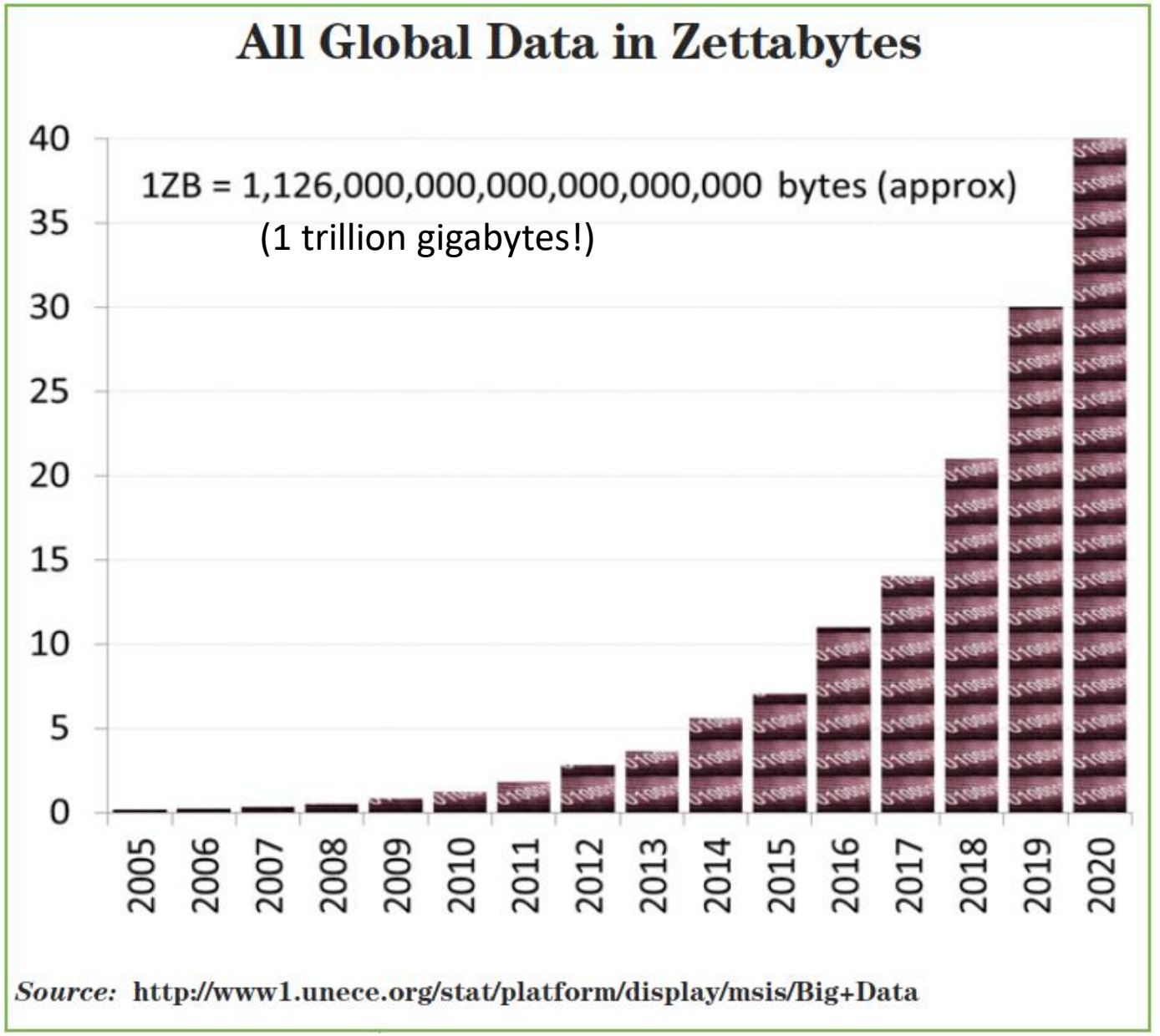
Atlanta Liu
Maruthi Mutnuri
Edwin Aguirre
Greg Cameron

November 2019

Exponential Growth in Data

The 4 V's of Big data

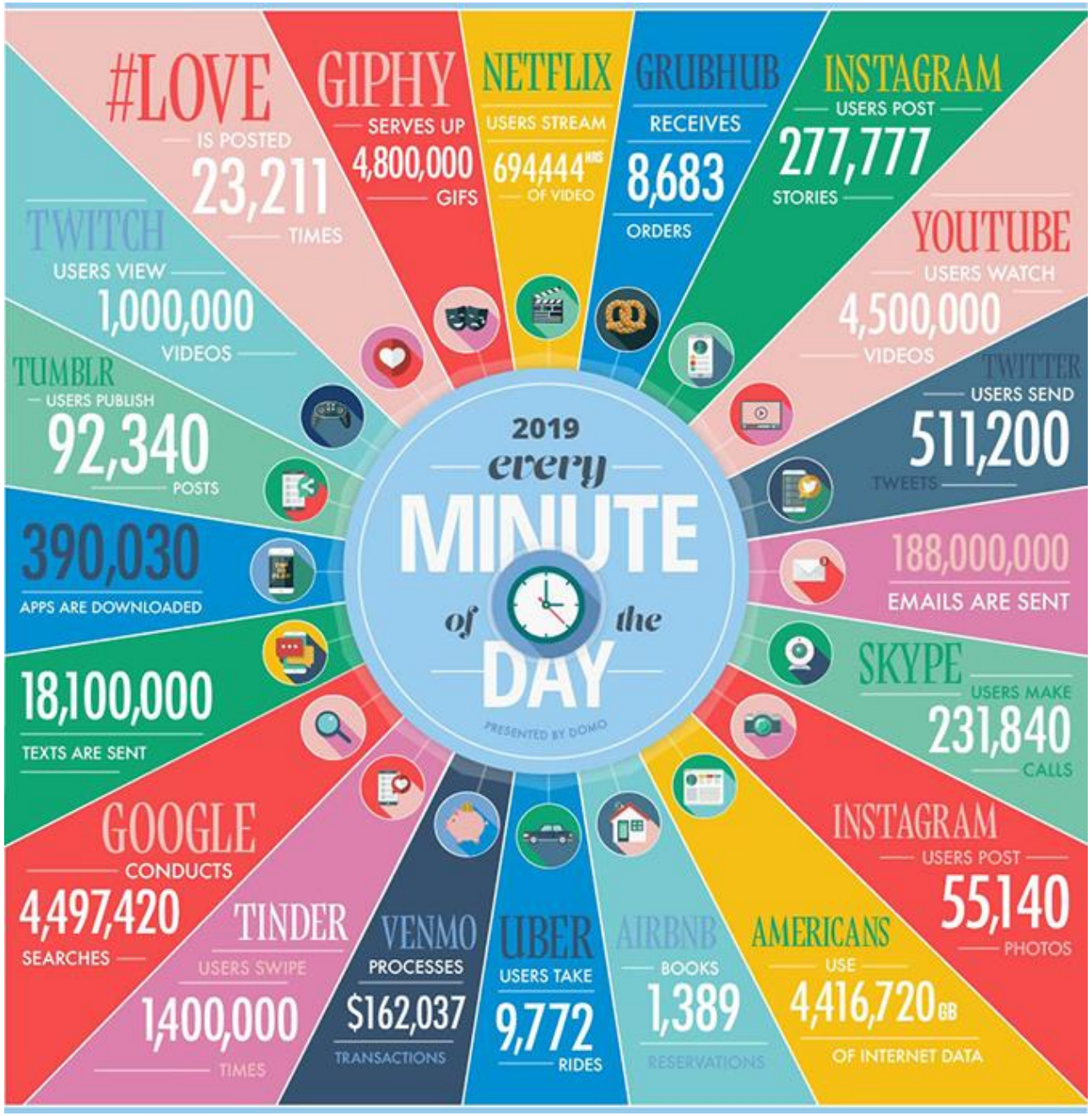
- Volume
- Velocity
- Variety
- Veracity



Exponential Growth in Data

The 4 V's of Big data

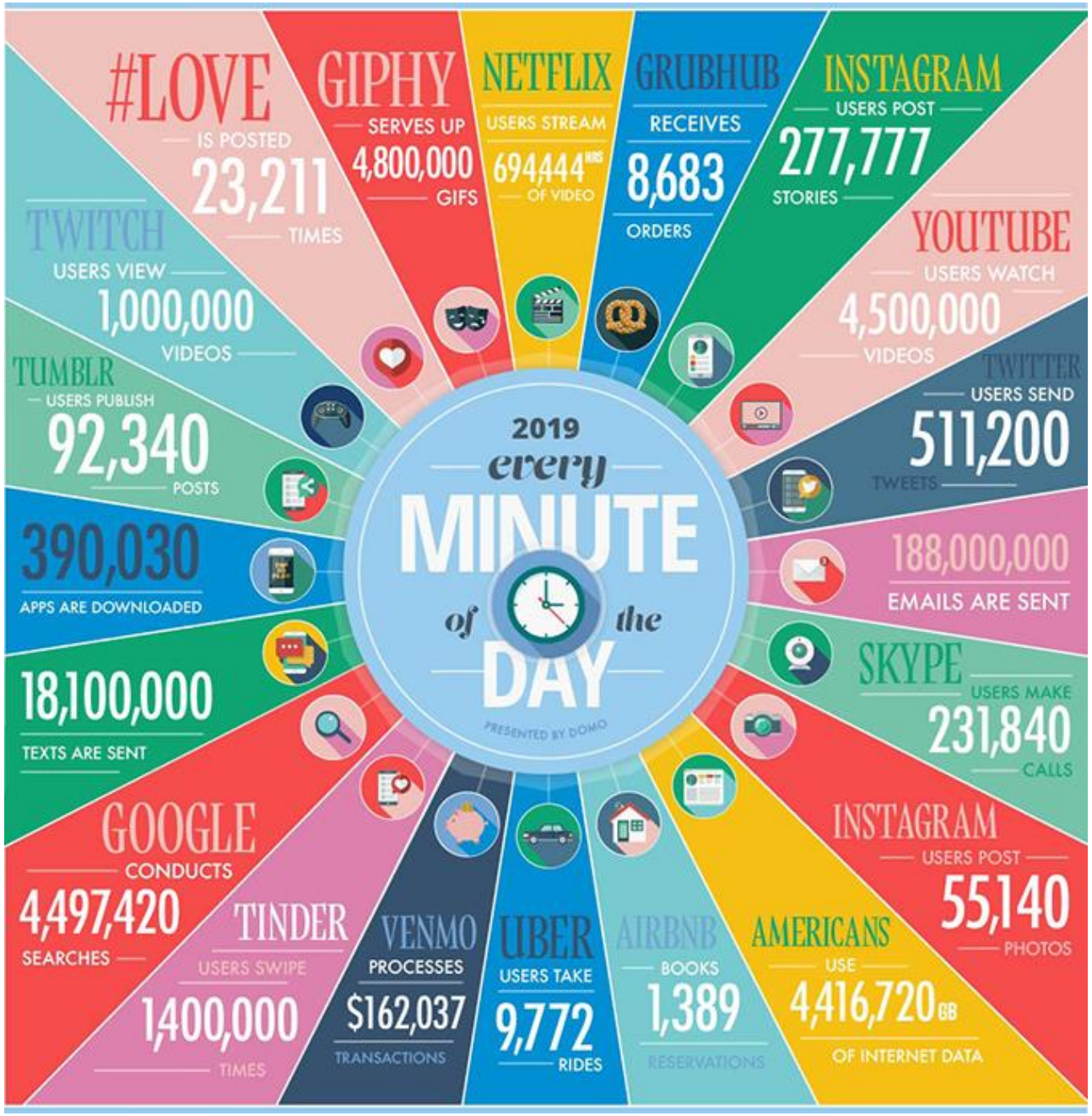
- Volume
- Velocity
- Variety
- Veracity



https://www.domo.com/learn/data-never-sleeps-7

How do we add value with all this data?

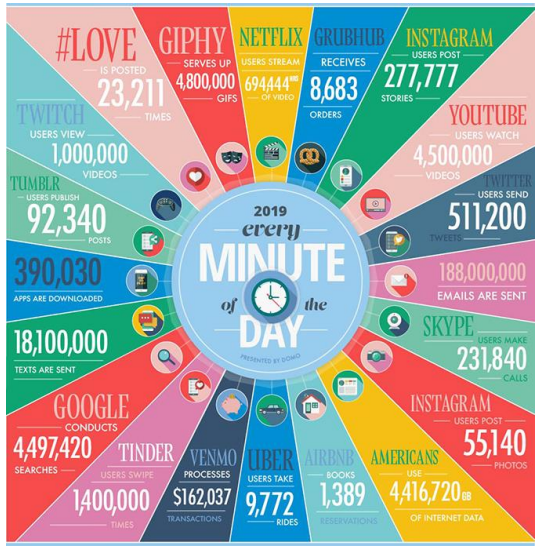
- Real-time, reliable results
  - streaming data
  - efficient parallel processing
  - fault handling
- Straightforward coding
  - good libraries on several platforms
- Actionable insights!
  - machine learning
  - useful visualizations



https://www.domo.com/learn/data-never-sleeps-7

Overview

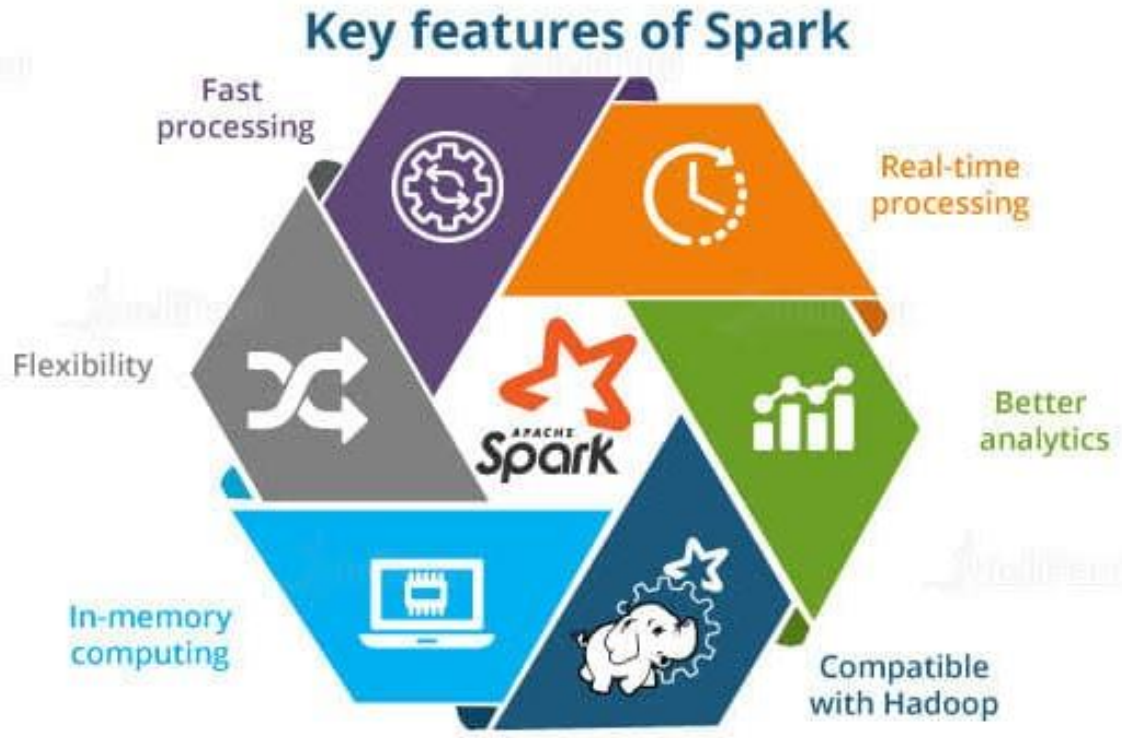
- Open-source cluster-computing framework
- Developed in UC Bekeley's AmpLab in 2009
- Donated to Apache Software Foundation in 2013, first release in 2014
- Developed to enable real-time processing of large/streaming datasets



Real-time results!

Features

- Distributes data to memory
  - Up to 100x faster than reading from disk
- Cross-platform
  - Hadoop, Apache Mesos, Kubernenetes, stand-alone or in the cloud
- Multiple languages
  - Java, R, Scala, and Python
- Rich set of libraries
  - Enable large variety of processing

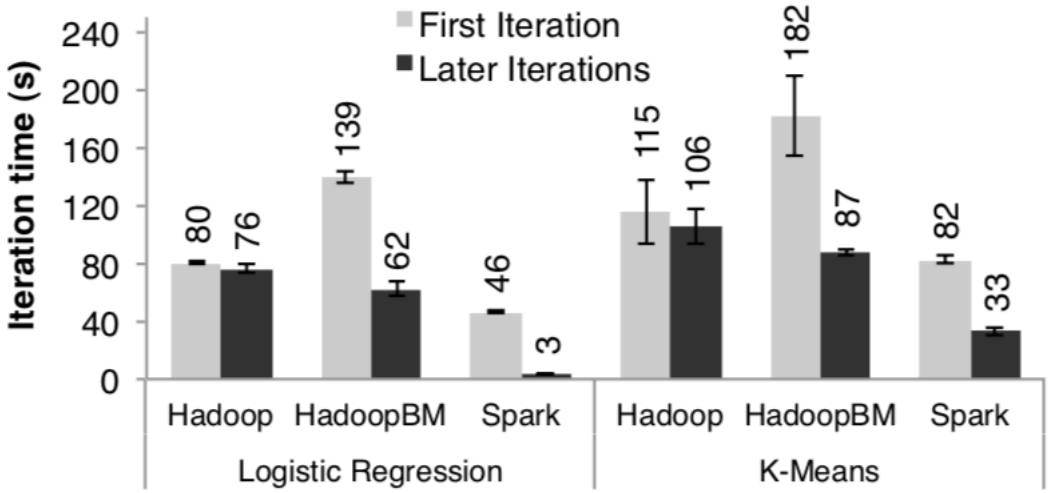


From https://intellipaat.com/blog/tutorial/spark-tutorial/spark-features/

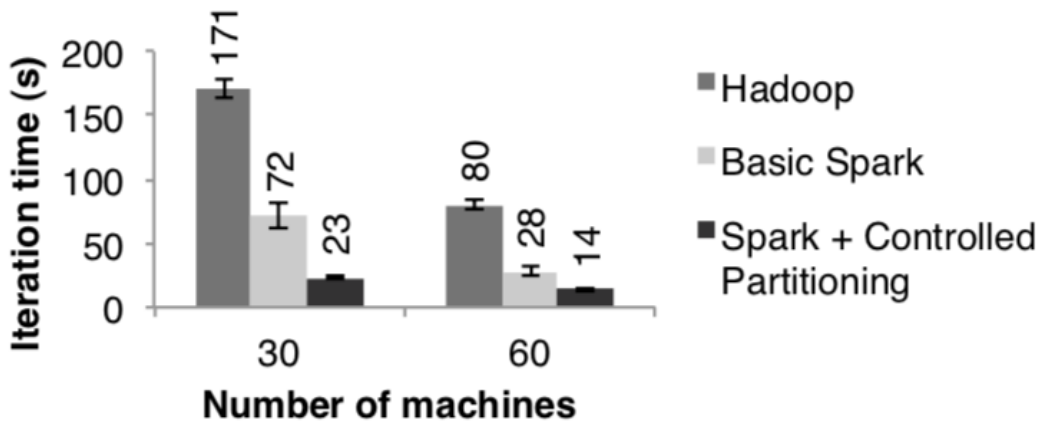
Performance

- Spark outperforms Hadoop by up to 20x in iterative machine learning applications.
  - avoids I/O and deserialization costs by storing data in memory as Java objects
- Spark can be used to query a 1 TB dataset interactively with latencies of 5–7 seconds

Machine Learning Example



Page Rank Example



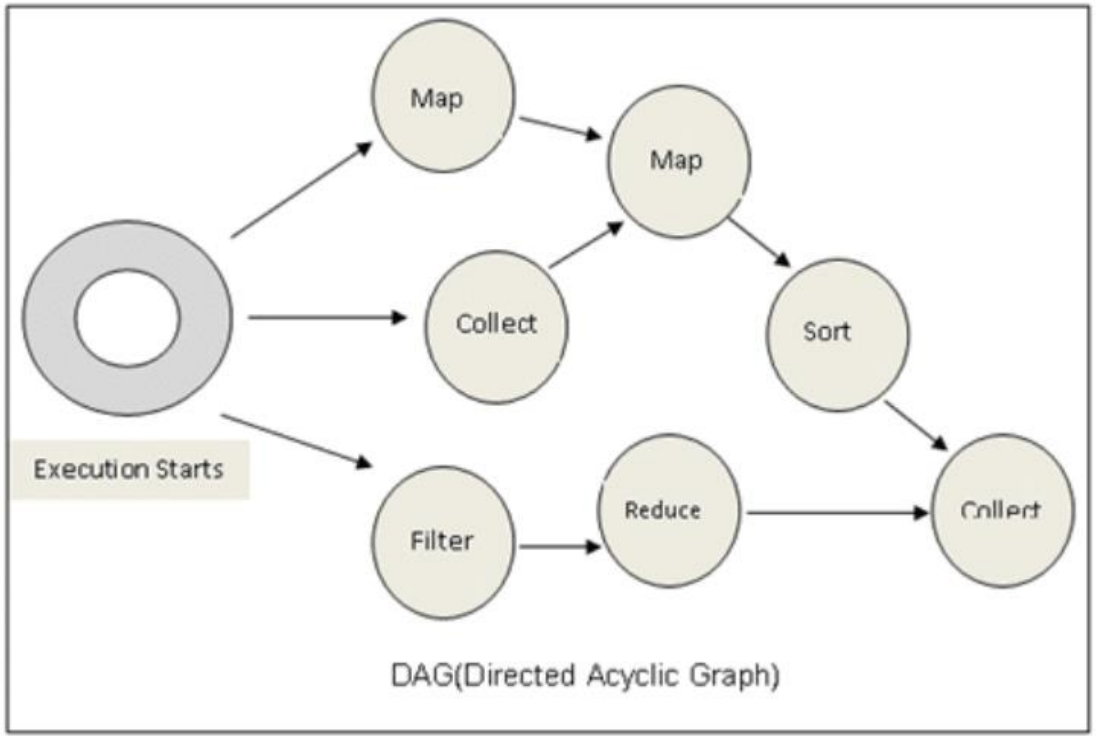
http://people.csail.mit.edu/matei/papers/2012/nsdi_spark.pdf

RDDs (Resilient Distributed Datasets)

- Fundamental Data Structure of Apache Spark
- Immutable
- Parallel Data Structures
- Are represented by coarse-based transformation (i.e. map,filter and join) that are applied to datasets
- RDDs can only be created through deterministic operations (transformations) on Data or from other RDDs. Hence, the RDDs are not an actual dataset but a transformation representation of a dataset.
- Fault tolerant

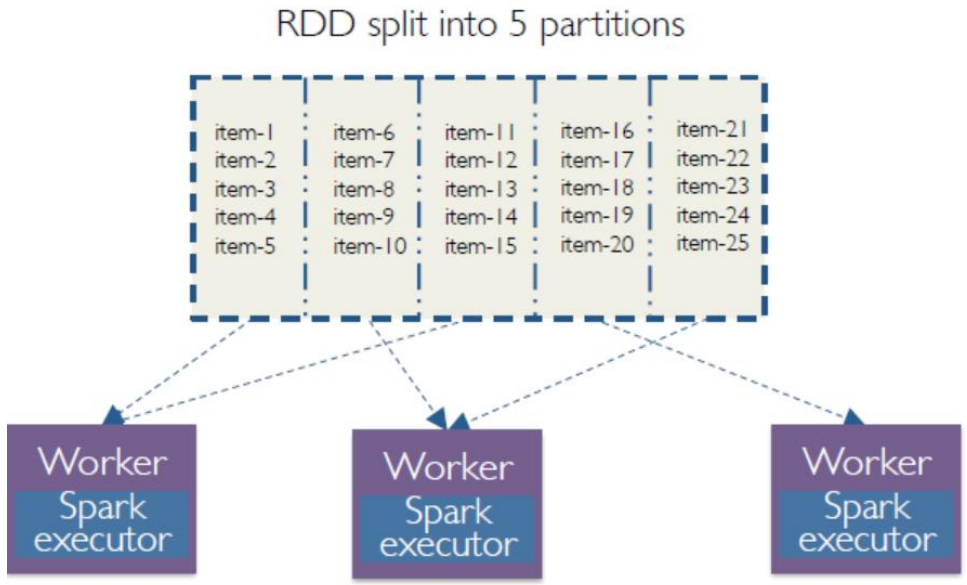
DAGs (Directed acyclic graphs)

- DAGs provides workflow to RDDs
- Structure that is used to model pairwise relations between objects
- Components of graphs are vertices(nodes or points) which are connected by edges (links or lines)
- DAGs have edges with directions and connect their vertices sequentially.
- Apache Spark uses DAGs to represent vertices as RDDs, and edges as transformations performed to RDDs.



Partitions

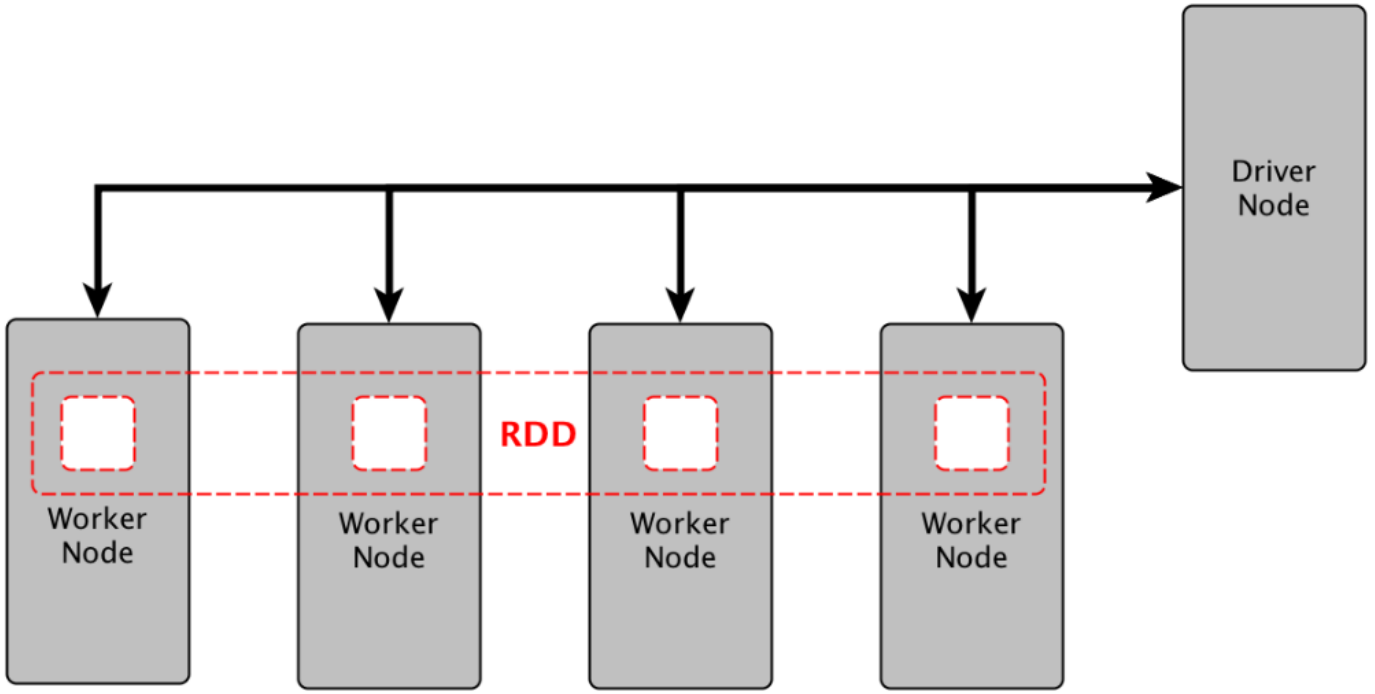
- Data size that is big in size, needs to be “partitioned” or sliced across different nodes/machines
- Number of partitions is chosen by the user



https://medium.com/@thejasbabu/spark-under-the-hood-partition-d386aaaa26b7

Parallelized

- Functions applied to the dataset can be run simultaneously to each “slice” or “partition”.



https://medium.com/@lavishj77/spark-fundamentals-part-2-a2d1a78eff73

Fault Tolerance/Failure Recovery

- The previously mentioned coarse grained transformation allows for easy “fault tolerance” as they are logged (called a lineage).
- If a node fails, the task manager then runs the lineage to restore information across all nodes
- Partitioned RDDs have enough information about how other partitions were derived, so lost data can be easily recovered

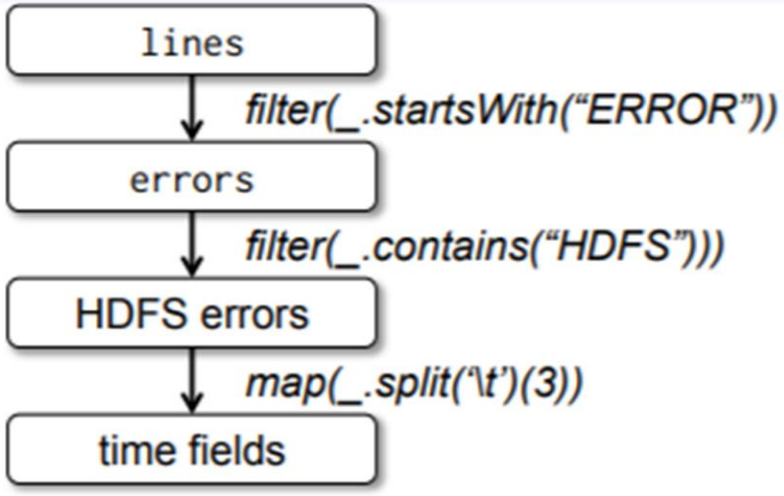
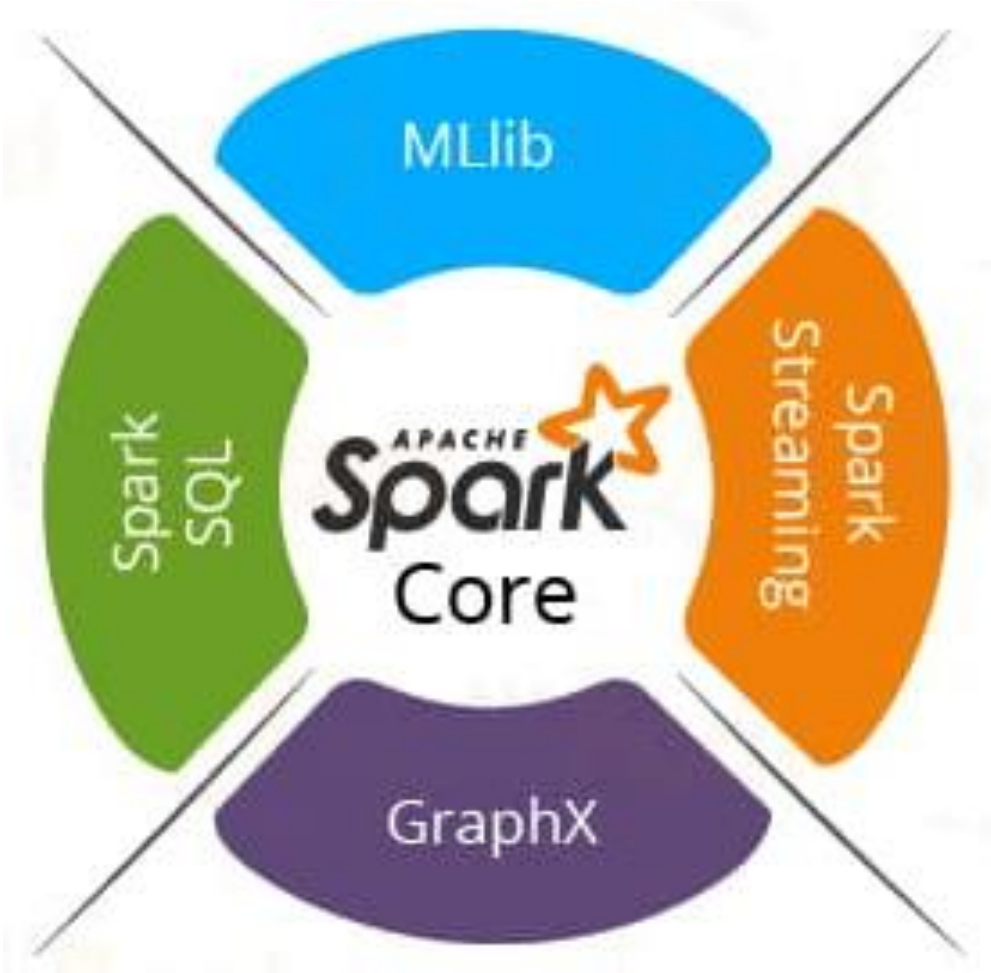


Figure 1: Lineage graph for the third query in our example. Boxes represent RDDs and arrows represent transformations.

http://people.csail.mit.edu/matei/papers/2012/nsdi_spark.pdf

Spark's Libraries



https://intellipaat.com/blog/what-is-apache-spark/

SQL and DataFrames

- Uses two main concepts:
  - DataFrame API
  - Catalyst optimizer
- The DataFrame API uses a collection of objects from Java and Python
- The Catalyst optimizer contains a general library, which represents trees and applies different rules to transform them
- The library runs on top of Spark as shown in picture

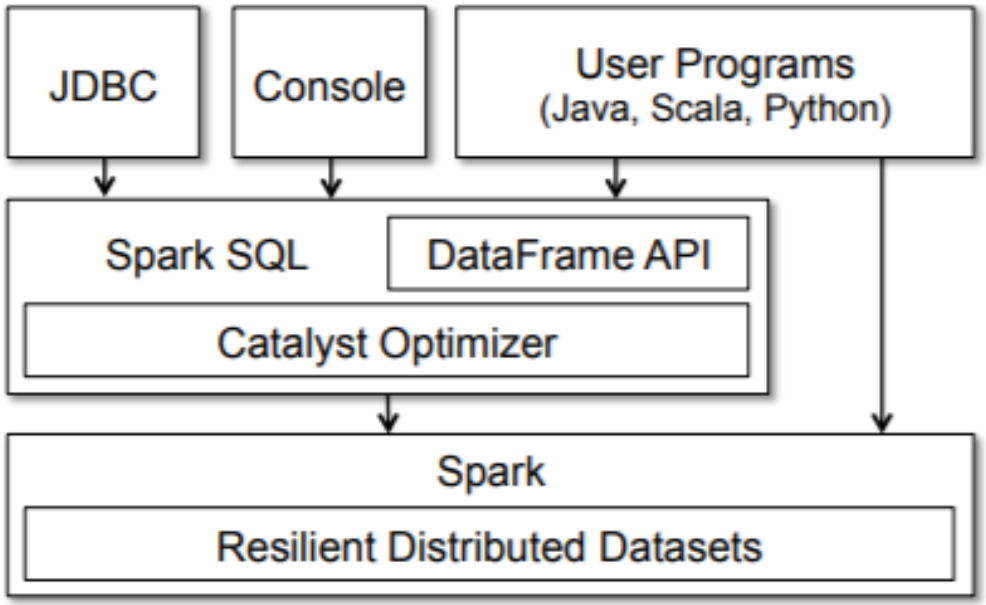


Figure 1: Interfaces to Spark SQL, and interaction with Spark.

http://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf

SQL and DataFrames

- The Catalyst's main Data type is a tree, which is composed of a node type objects
- Trees can be modified by functional transformations, and are able to communicate from one tree to another
- Tree Objects are Immutable
- Spark SQL uses the Pattern matching function/rule that allows for the extraction of values from nested structures of algebraic types.

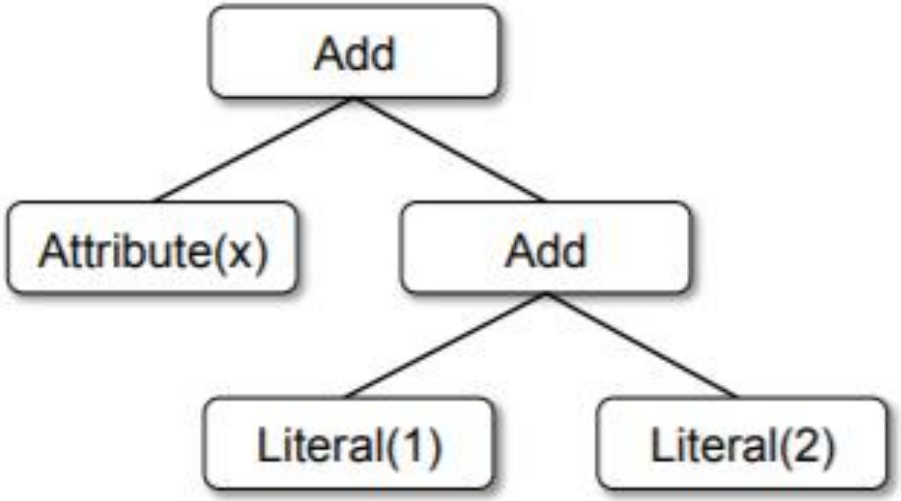


Figure 2: Catalyst tree for the expression x+(1+2).

http://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf

SQL and DataFrames

- Catalyst's transformation tree is used in four phases.
- Analysis: rule-based optimization performed when there are any unresolved relations or references
- Logical Optimization: rule-based optimization, to the already build "Logical Plan"
- Physical Planning: cost-based optimization, that chooses the best model based on a cost model
- Code Generation: Uses "quasiquotes", to produce code efficiently in Java bytecode

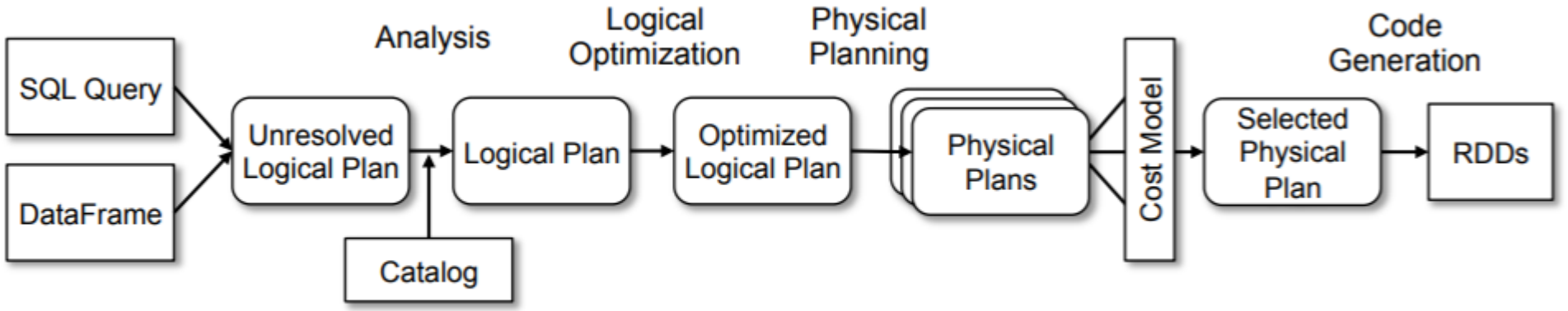


Figure 3: Phases of query planning in Spark SQL. Rounded rectangles represent Catalyst trees.

http://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf

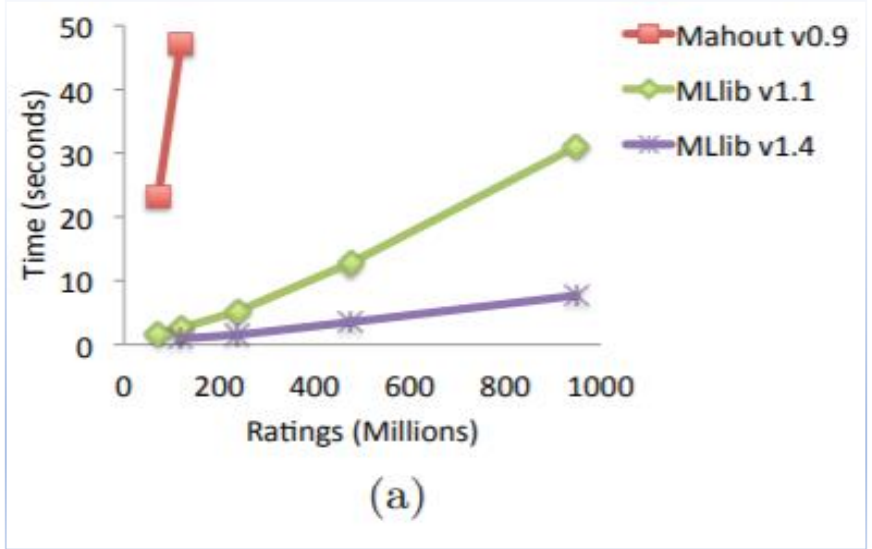
Machine Learning Library(MLlib)

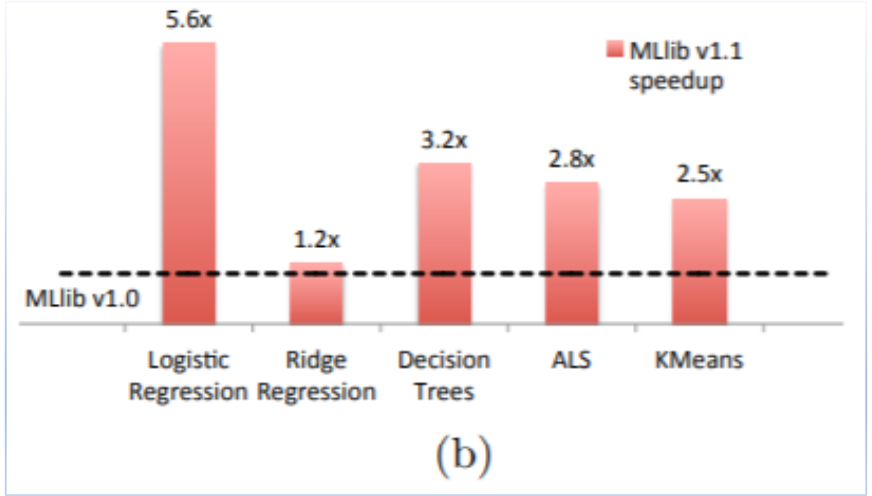
- Provides fast distributed implementations of common learning algorithms, which are: linear models, naïve Bayes, decision trees, and k-clustering
- Provide many algorithms optimizations, such as the ALS algorithm, to improve on linear algebra operations
- Provides with Machine learning pipelines , which can be often described as workflows that involve sequentially data preparation, feature extraction, model fitting and validation stages.
- Provides with robust documentation, it also provides with a list of codes dependencies.

Machine Learning Library(MLlib)

- MLlib's 1.4 version is faster than MLlib's version 1.1. Also, it can be observed that when using Apache's Mahout v0.9 (which runs Hadoop MapReduce), the computational time is substantially slower than any of MLlib's versions. The benchmark used for this graph was the ALS algorithm.

- MLlib v1.1 relative improvement over MLlib 1.0 , it can be observed that there was on average 3.0x improvement on different algorithms

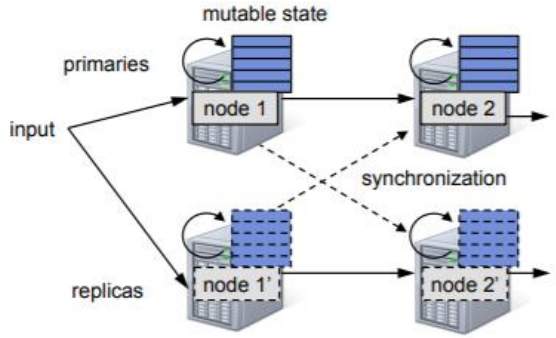




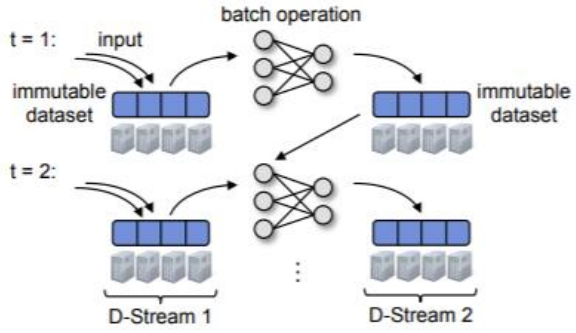
http://www.jmlr.org/papers/volume17/15-237/15-237.pdf

Spark Streaming

- Proposes new streaming model than the well known “continuous operator” model
- New model avoids regular problems by structuring computations as a set of short, stateless, deterministic tasks, rather than continuous operators. These objects are called discretized streams.



(a) Continuous operator processing model. Each node continuously receives records, updates internal state, and emits new records. Fault tolerance is typically achieved through replication, using a synchronization protocol like Flux or DPC [34, 5] to ensure that replicas of each node see records in the same order (*e.g.*, when they have multiple parent nodes).



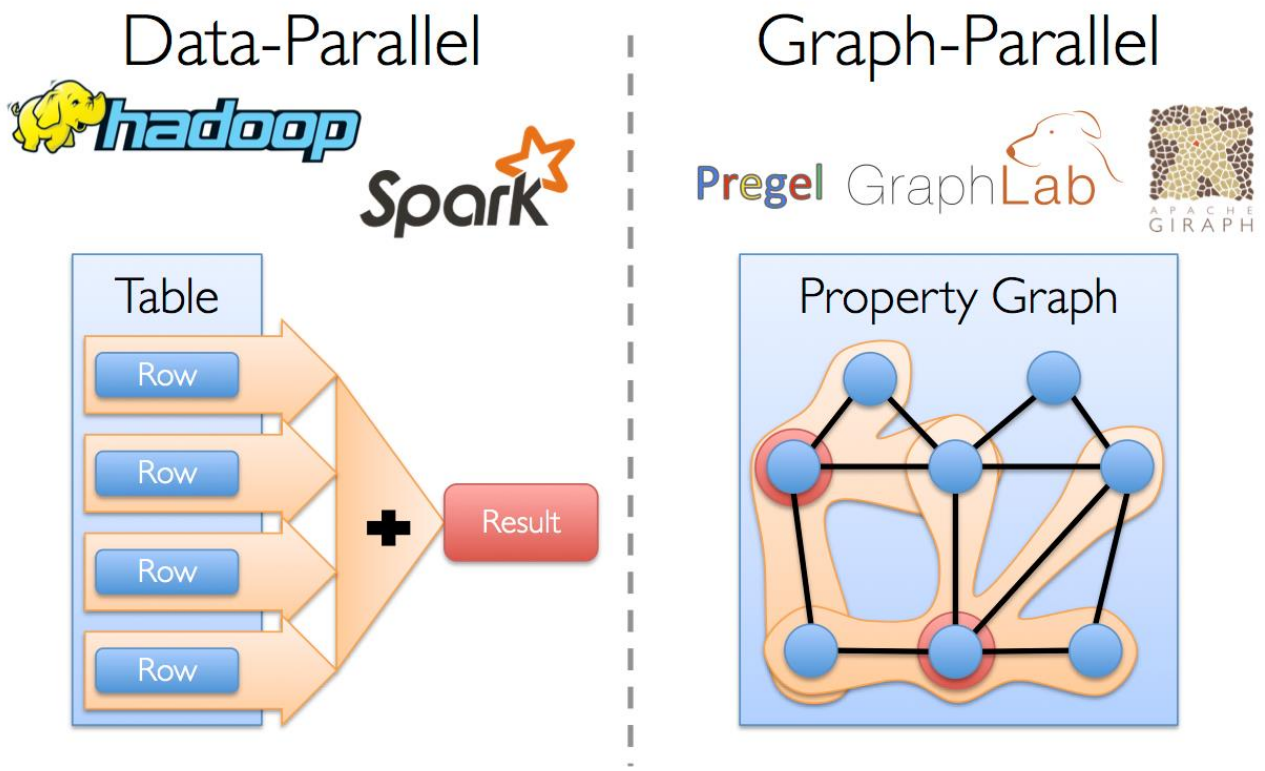
(b) D-Stream processing model. In each time interval, the records that arrive are stored reliably across the cluster to form an immutable, partitioned dataset. This is then processed via deterministic parallel operations to compute other distributed datasets that represent program output or state to pass to the next interval. Each series of datasets forms one D-Stream.

Figure 1: Comparison of traditional record-at-a-time stream processing (a) with discretized streams (b).

http://people.csail.mit.edu/matei/papers/2013/sosp_spark_streaming.pdf

GraphX – Intro to Graph Data

- Graph Data
  - Graphs – consist of nodes and edges
    - Nodes contain entities
    - Edges contain relationships
  - Most social media data are graph data
- Graph-parallel systems
  - Optimized for graph-based data (NoSQL)

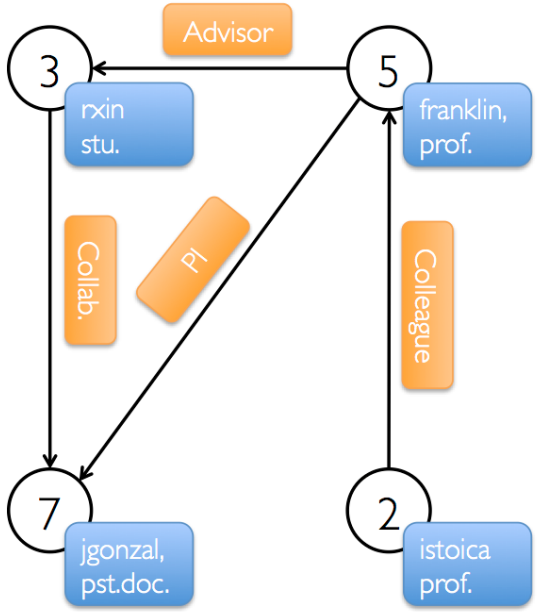


http://ampcamp.berkeley.edu/big-data-mini-course/graph-analytics-with-graphx.html

Library: GraphX

- Distributed graph framework that unifies graph-parallel and data-parallel computation
  - Data can be viewed as both graphs and tables
  - Uses two RDDs - edge collection, vertex collection
- Allows end-to-end pipeline of graph and table operations without moving the data
  - Less efficient on pure graph-parallel workflows
  - More efficient on combined graph-parallel and data-parallel workflows

Property Graph



Vertex Table

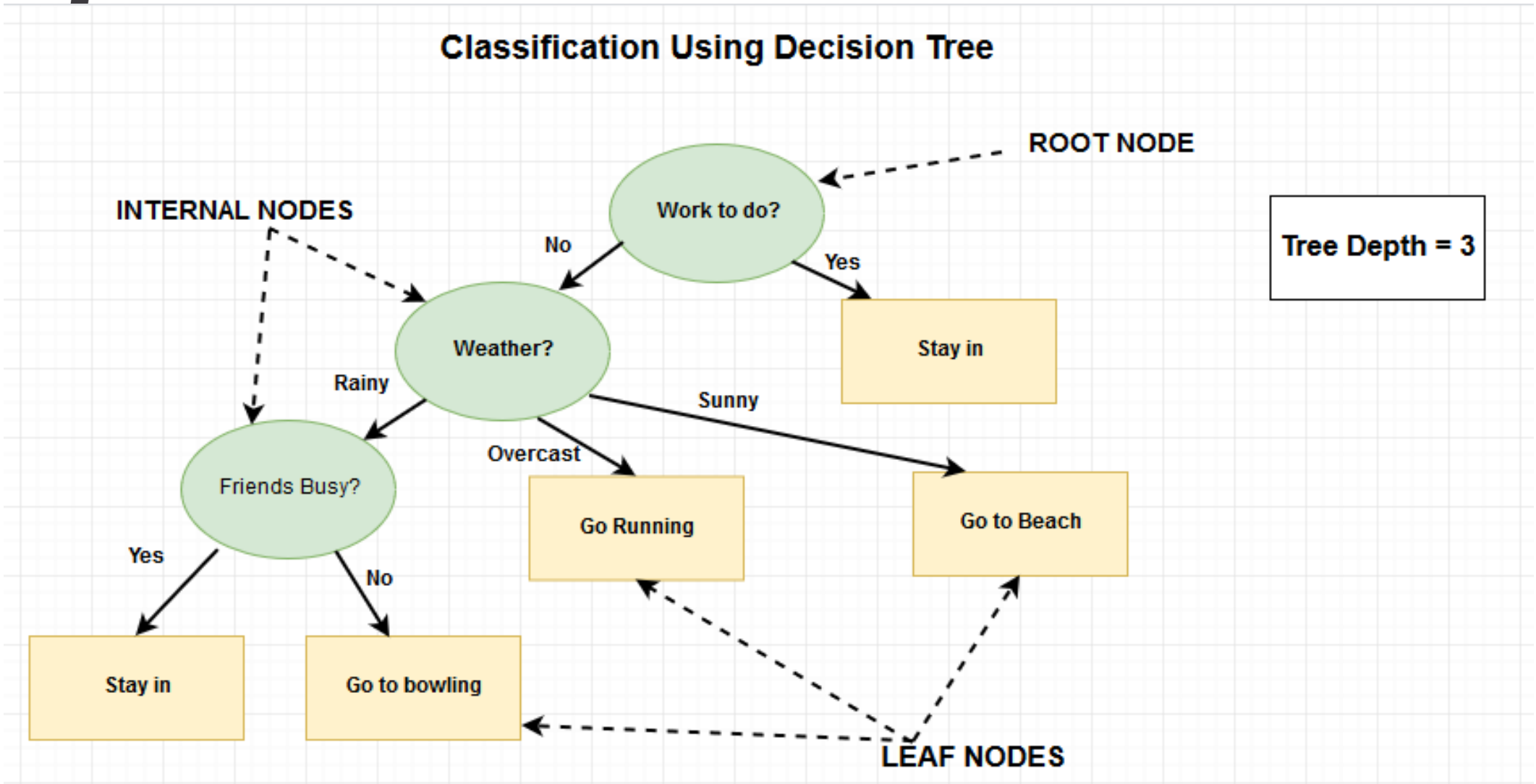
Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

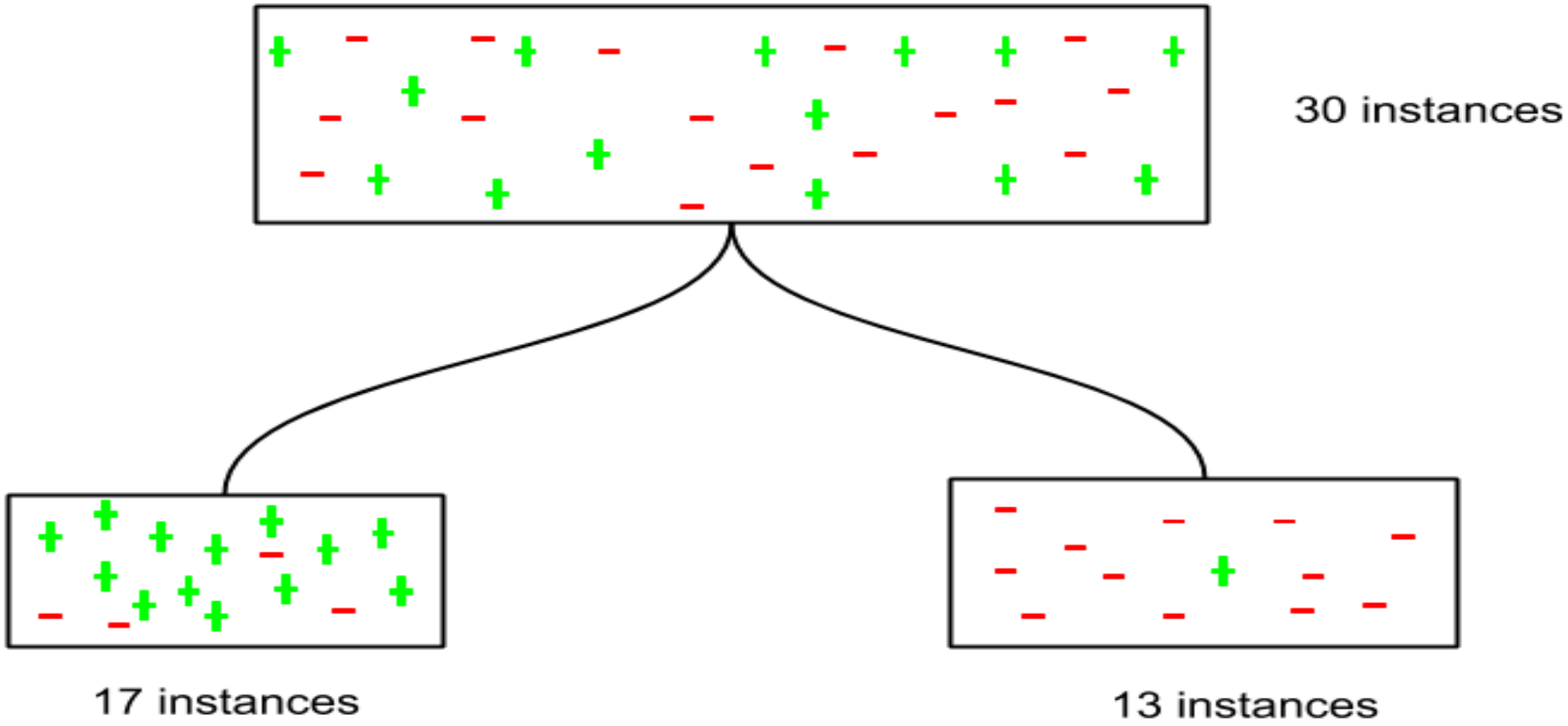
SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

https://spark.apache.org/docs/0.9.0/graphx-programming-guide.html

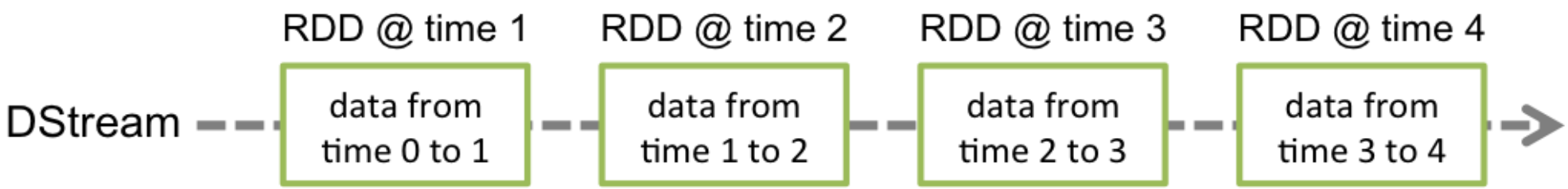
Technical Demonstration: ML Lib - Decision Tree



Decision Tree - Purity

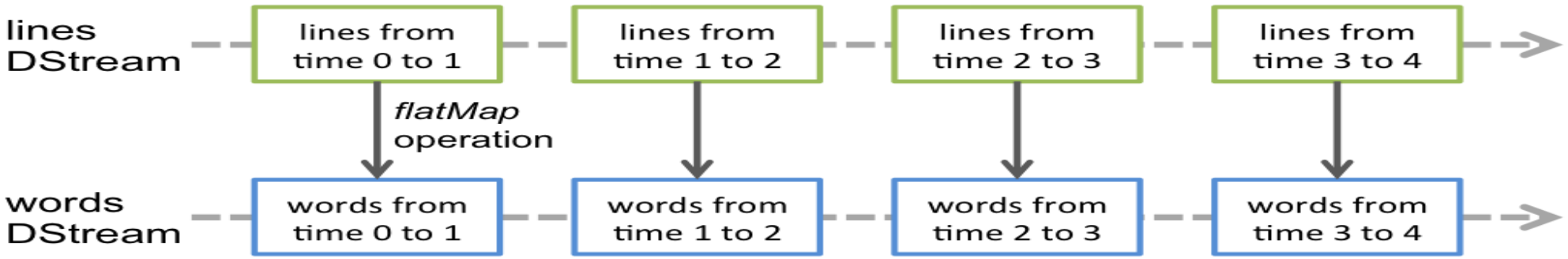


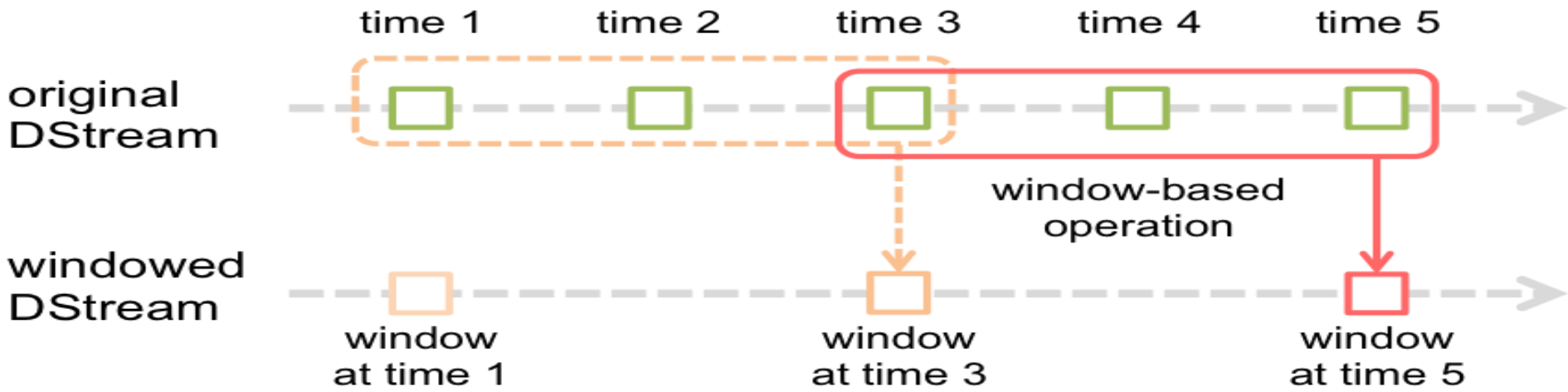
Technical Demonstration: Spark Streaming



https://spark.apache.org/docs/latest/streaming-programming-guide.html

Technical Demonstration: Spark Streaming





https://spark.apache.org/docs/latest/streaming-programming-guide.html

Need for Streaming Applications

- Real-time Sentiment Analysis – based on social media posts (facebook, twitter, etc.)
- Real-time Fraud Detection of bank transactions

Use Cases

- Netflix
  - captures all member activities to personalize recommendations
  - processes **450 billion events per day**
- MyFitnessPal
  - used to build their data pipeline, create a list of ‘Verified Foods’
  - provided a **ten-fold speed improvement** over their previous data pipeline

Use cases

Apache Spark at MyFitnessPal

The largest health and fitness community MyFitnessPal helps people achieve a healthy lifestyle through better diet and exercise. MyFitnessPal uses apache spark to clean the data entered by users with the end goal of identifying high quality food items. Using Spark, MyFitnessPal has been able to scan through food calorie data of about 80 million users. **Earlier, MyFitnessPal used Hadoop to process 2.5TB of data and that took several days to identify any errors or missing information in it**.

Netflix

Netflix uses Apache Spark for real-time stream processing to provide online recommendations to its customers. Streaming devices at Netflix send events which capture all member activities and play a vital role in personalization. It processes **450 billion events per day** which flow to server side applications and are directed to Apache Kafka.

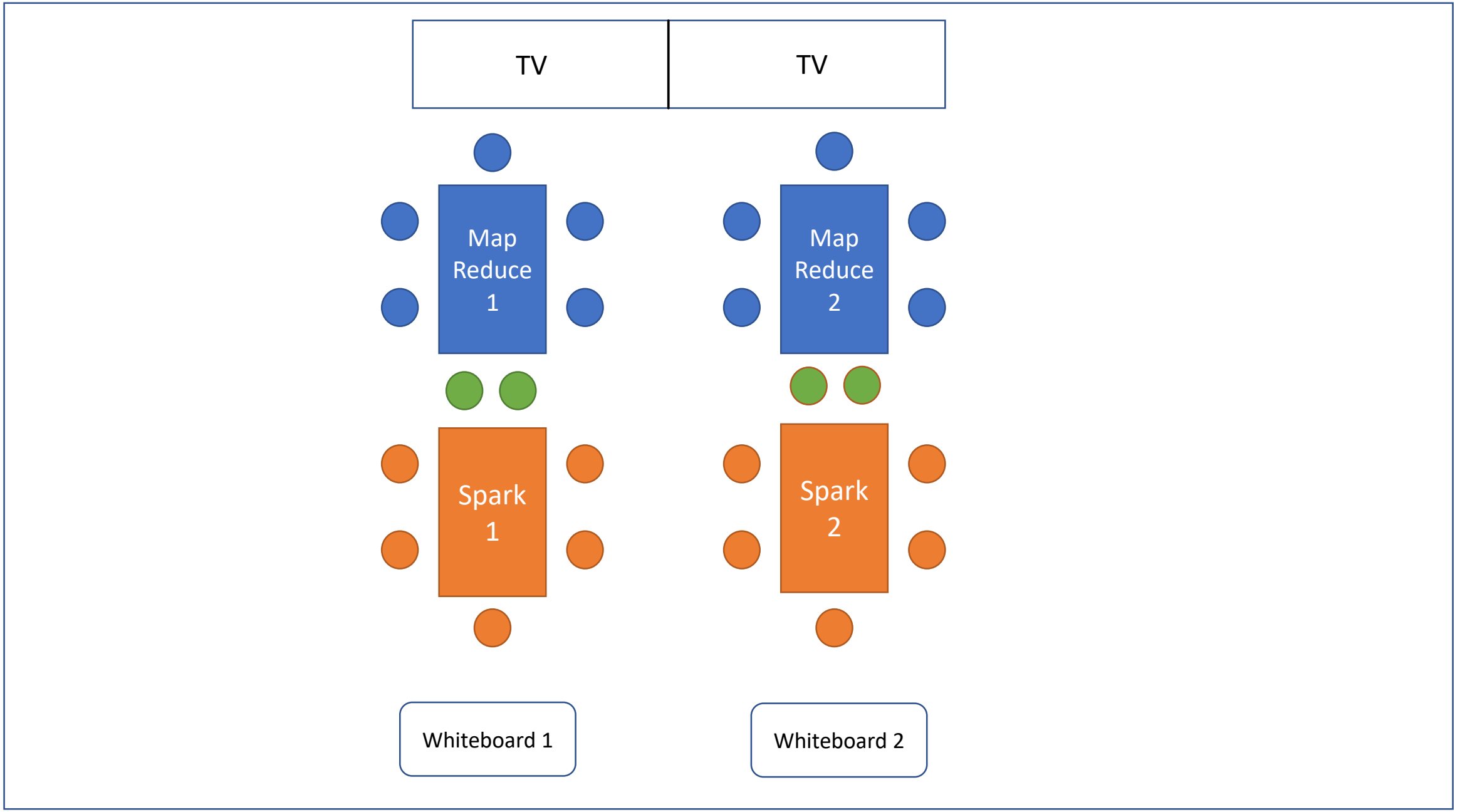
https://www.dezyre.com/article/top-5-apache-spark-use-cases/271

Key Take-Aways

- With more data being created all the time and users requiring real-time results, we need methods that can process big data quickly
- Spark distributes data to memory – up to 100x faster than reading from disk
- Rich set of libraries enable large variety of processing
- Enabled by RDDs (Resilient Distributed Arrays)
  - Read only, partitioned collection of records that are fault-tolerant

Interactive Component

Speed Comparison: MapReduce vs Spark


TV
TV
Map Reduce 1
Map Reduce 2
Spark 1
Spark 2
Whiteboard 1
Whiteboard 2

Instructions

Each worker will take 20 candies from the bin in the middle

Task 1: Group candies by category

Task 2: Count candies

Task 3: For each category, perform the following actions:

- If candy count in Lollipop is **even**, discard all Lollipop candies
- If candy count in Chocolate Minibar is **odd**, add an extra Chocolate Minibar candy to your pile
- If candy count in Kerr's Toffee is **even**, remove one Kerr's Toffee candy from your pile

Task 4: Sum Total Candy (Regardless of Category)

Category	Lollipop	Chocolate Minibar	Kerr's Toffee
Count	8	9	3
Count (after action)	0	10	2
Sum Total	12 (0 + 10 + 2)		

Debrief

Speed Comparison: MapReduce vs Spark

References

- PySpark Installation Guide and Jupyter notebooks – available on D2L
- Spark website
  - https://spark.apache.org
- Key technical papers

- M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing", *NSDI '12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation,* pp. 2-2, April 2012. Available: http://people.csail.mit.edu/matei/papers/2012/nsdi_spark.pdf
- Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, Ion Stoica. "*Discretized Streams: Fault-Tolerant Streaming Computation at Scale.*" SOSP 2013. November 2013. Available: http://people.csail.mit.edu/matei/papers/2013/sosp_spark_streaming.pdf
- X. Meng,J. Bradley, B.Yavuz,E.Sparks,S.Venkataramn, D.Liu,J.Freeman,DB Tai, M. Made, S.Owen, D. Xin, R.Xin,M. Franklin, R.Zadeh, M.Zaharia, A. Talwalkar, "MLlib: Machine Learning in Apache Spark",Journal of Machine Learning Research (2016), pp.1-7, May, 2015. Available: http://www.jmlr.org/papers/volume17/15-237/15-237.pdf
- M. Armburst, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M.J. Franklin, A. Ghodsi, M. Zaharia, "Spark SQL: Relational data processing in Spark", *SIGMOD 2015,* June 2015. Available: http://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf
- R.S. Xin, D. Crankshaw, A. Dave, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "GraphX: Unifying data-parallel and graph-parallel analytics". *Arxiv,* Feb. 2014. Available: https://amplab.cs.berkeley.edu/wp-content/uploads/2014/02/graphx.pdf