Analyzing Big Data in Hadoop Spark

Kazi Aminul Islam

Department of Computer Science

Kennesaw State University

Oto von Bismarck

- Data applications are like sausages. It is better not to see them being made.

Data

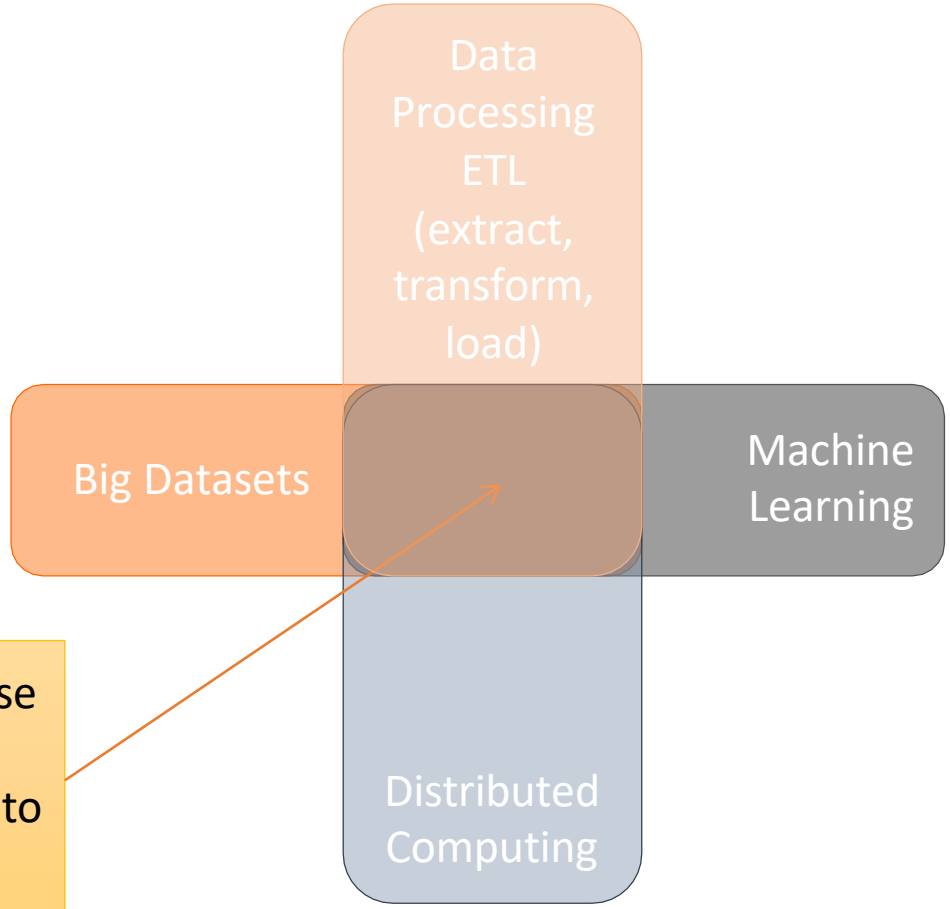
- The Large Hadron Collider produces about 30 petabytes of data per year
- Facebook’s data is growing at 8 petabytes per month
- The New York stock exchange generates about 4 terabyte of data per day
- YouTube had around 80 petabytes of storage in 2012
- Internet Archive stores around 19 petabytes of data

Cloud and Distributed Computing

- The second trend is pervasiveness of cloud based storage and computational resources
  - For processing of these big datasets
- Cloud characteristics
  - Provide a scalable standard environment
  - On-demand computing
  - Pay as you need
  - Dynamically scalable
  - Cheaper

Data Processing and Machine learning Methods

- Data processing (third trend)
  - Traditional ETL (extract, transform, load)
  - Data Stores (HBase, ……..)
  - Tools for processing of streaming, multimedia & batch data
- Machine Learning (fourth trend)
  - Classification
  - Regression
  - Clustering
  - Collaborative filtering



Working at the Intersection of these four trends is very exciting and challenging and require new ways to store and process **Big Data**

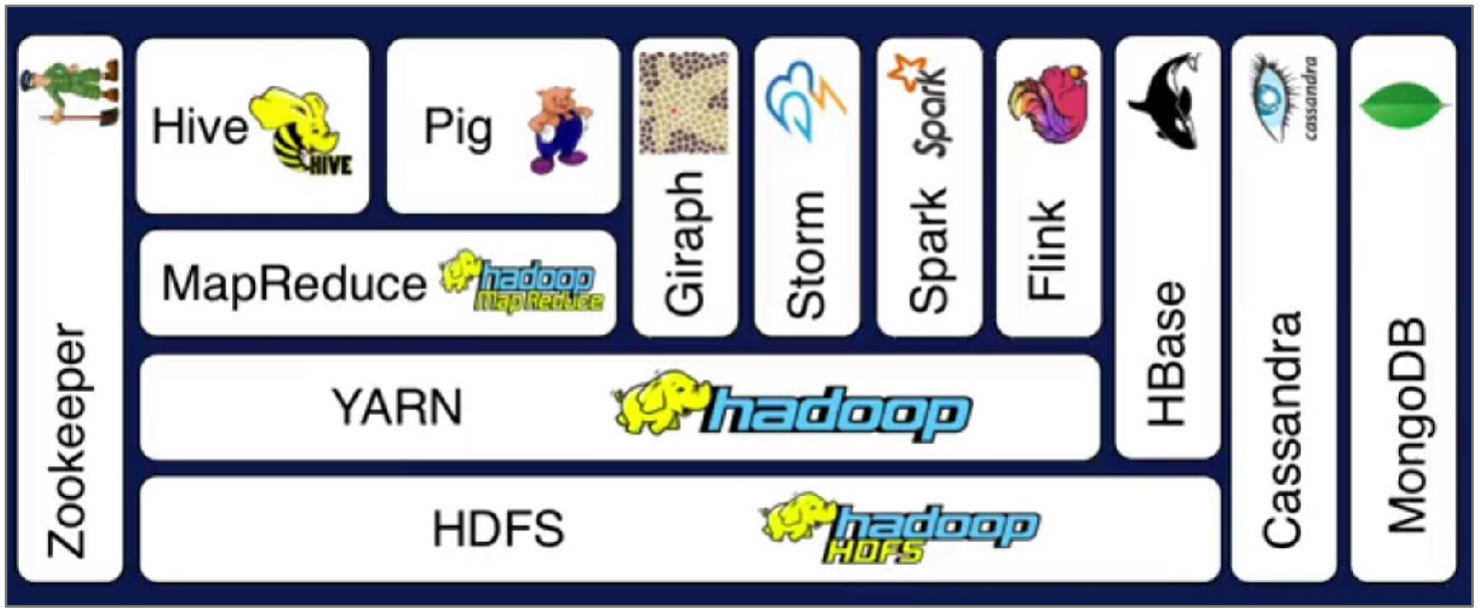
Tasks Not Easily Accomplished 5 Years Ago

- Build a model to detect credit card fraud using thousands of features and billions of transactions.
- Intelligently recommend millions of products to millions of users.
- Estimate financial risk through simulations of portfolios including millions of instruments.
- Easily manipulate data from thousands of human genomes to detect genetic associations with disease.

Apache Hadoop

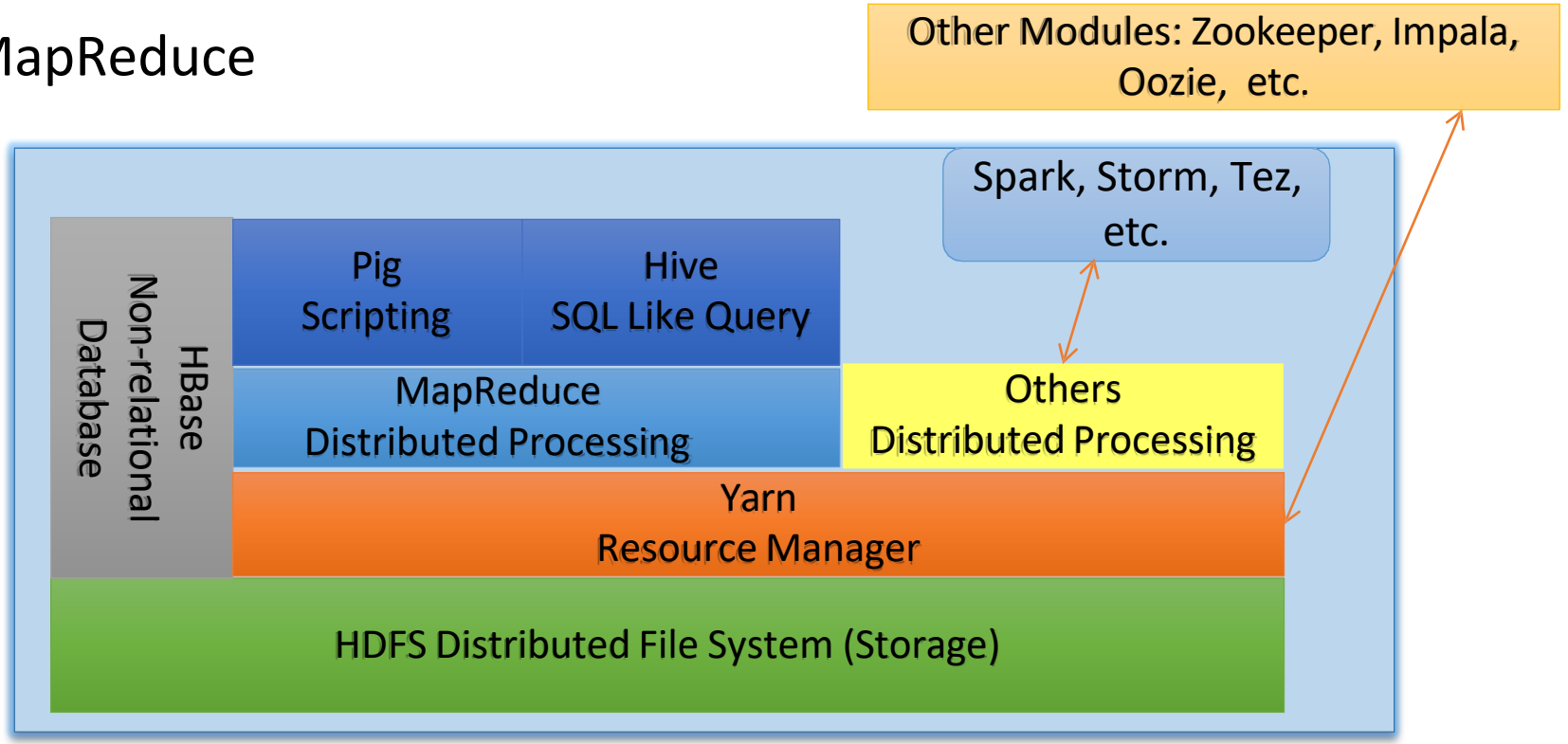
- Open source, enable scalability on commodity hardware
- Widespread deployment, affordable
- Distributed, fault tolerance
- Data Science
- Answer questions such as “of the gazillion users who made it to the third page in our registration process, how many are over 25?”
- Or even what is the largest ethnic group among those over 25?

Hadoop Ecosystem



Apache Hadoop Basic Modules

- Hadoop Common
- Hadoop Distributed File System (HDFS)
- Hadoop YARN
- Hadoop MapReduce



Hadoop HDFS

- Hadoop distributed File System (based on Google File System (GFS) paper, 2004)
  - Serves as the distributed file system for most tools in the Hadoop ecosystem
  - Scalability for large data sets
  - Reliability to cope with hardware failures
- HDFS good for:
  - Large files
  - Streaming data
- Not good for:
  - Lots of small files
  - Random access to files
  - Low latency access

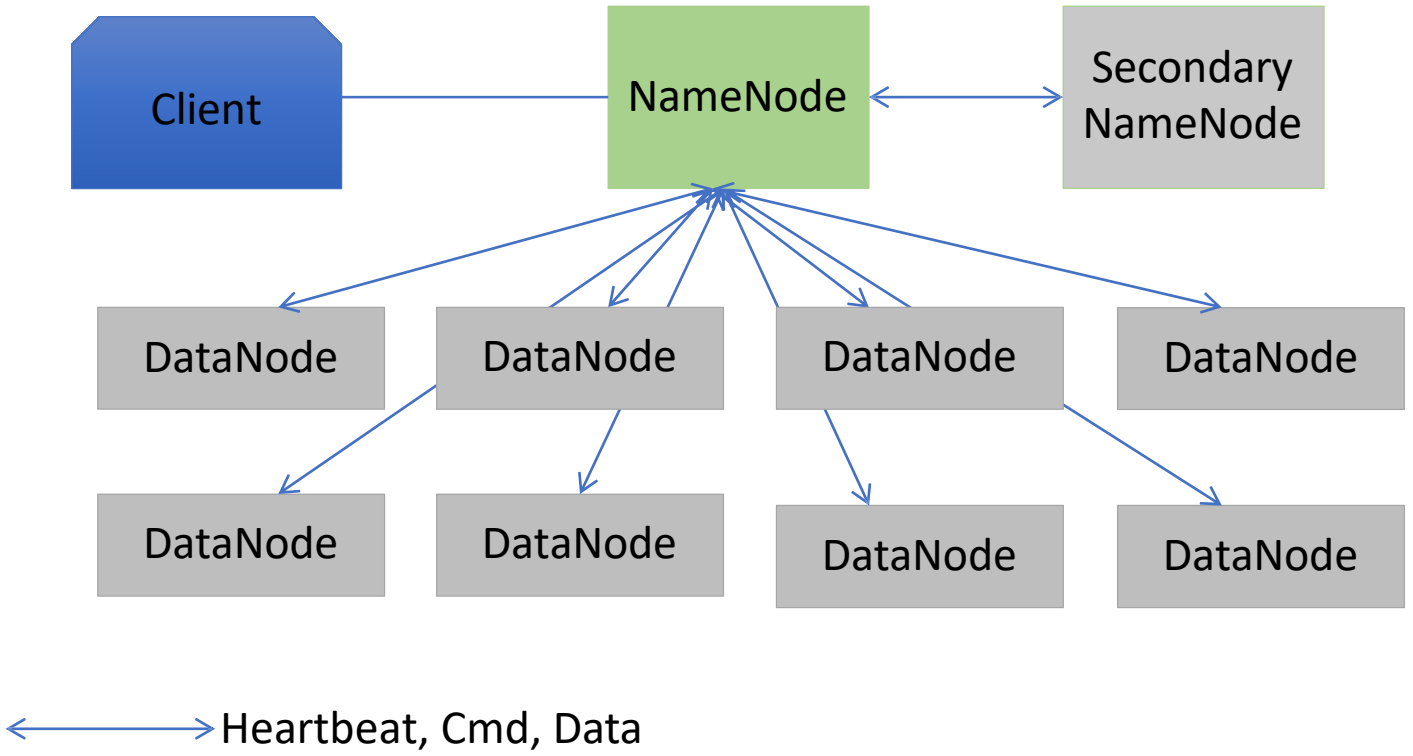
Single Hadoop cluster with 5000 servers and 250 petabytes of data

Design of Hadoop Distributed File System (HDFS)

- Master-Slave design
- Master Node
  - Single NameNode for managing metadata
- Slave Nodes
  - Multiple DataNodes for storing data
- Other
  - Secondary NameNode as a backup

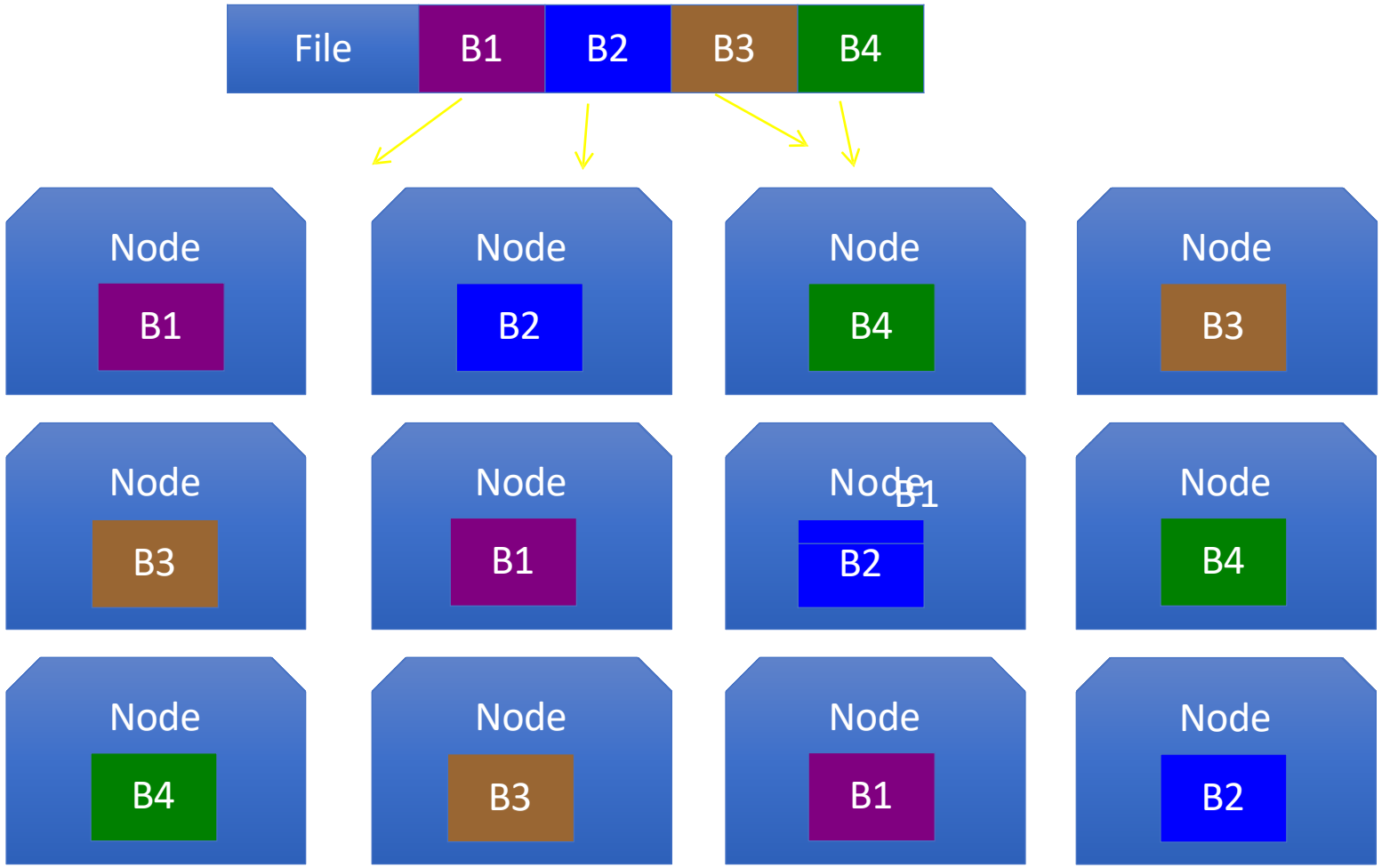
HDFS Architecture

NameNode keeps the metadata, the name, location and directory
DataNode provide storage for blocks of data



HDFS

What happens; if node(s) fail?
Replication of Blocks for fault tolerance



HDFS

- HDFS files are divided into blocks
  - It’s the basic unit of read/write
  - Default size is 64MB, could be larger (128MB)
  - Hence makes HDFS good for storing larger files
- HDFS blocks are replicated multiple times
  - One block stored at multiple location, also at different racks (usually 3 times)
  - This makes HDFS storage fault tolerant and faster to read

Few HDFS Shell commands

Create a directory in HDFS

- hadoop fs -mkdir /user/godil/dir1

List the content of a directory

- hadoop fs -ls /user/godil

Upload and download a file in HDFS

- hadoop fs -put /home/godil/file.txt /user/godil/datadir/
- hadoop fs -get /user/godil/datadir/file.txt /home/

Look at the content of a file

- Hadoop fs -cat /user/godil/datadir/book.txt

Many more commands, similar to Unix

Tools

- Need a programming paradigm that would be flexible, closer to the ground, and richer functionality in machine learning and statistics.
- Open source frameworks like R, the PyData stack, Octave: raid analysis and modeling over small datasets.
- HPC (high performance computing): low level abstraction, hard to use, and unable to read in-memory data.
- MPI: low-level distributed framework: difficult to program without C and distributed computing knowledge.
- Hadoop: cheaper than HPC and will get jobs done as if in single machine.

HBase

- NoSQL data store build on top of HDFS
- Based on the Google BigTable paper (2006)
- Can handle various types of data
- Stores large amount of data (TB,PB)
- Column-Oriented data store
- Big Data with random read and writes
- Horizontally scalable

HBase, not to use for

- Not good as a traditional RDBMs (Relational Database Model)
  - Transactional applications
  - Data Analytics

- Not efficient for text searching and processing

MapReduce: Simple Programming for Big Data

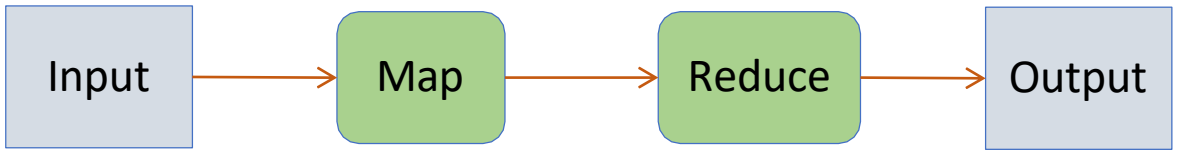
Based on Google's MR paper (2004)

- MapReduce is simple programming paradigm for the Hadoop ecosystem
- Traditional parallel programming requires expertise of different computing/systems concepts
  - examples: multithreads, synchronization mechanisms (locks, semaphores, and monitors)
  - incorrect use: can crash your program, get incorrect results, or severely impact performance
  - Usually not fault tolerant to hardware failure
- The MapReduce programming model greatly simplifies running code in parallel
  - you don't have to deal with any of above issues
  - only need to create, map and reduce functions

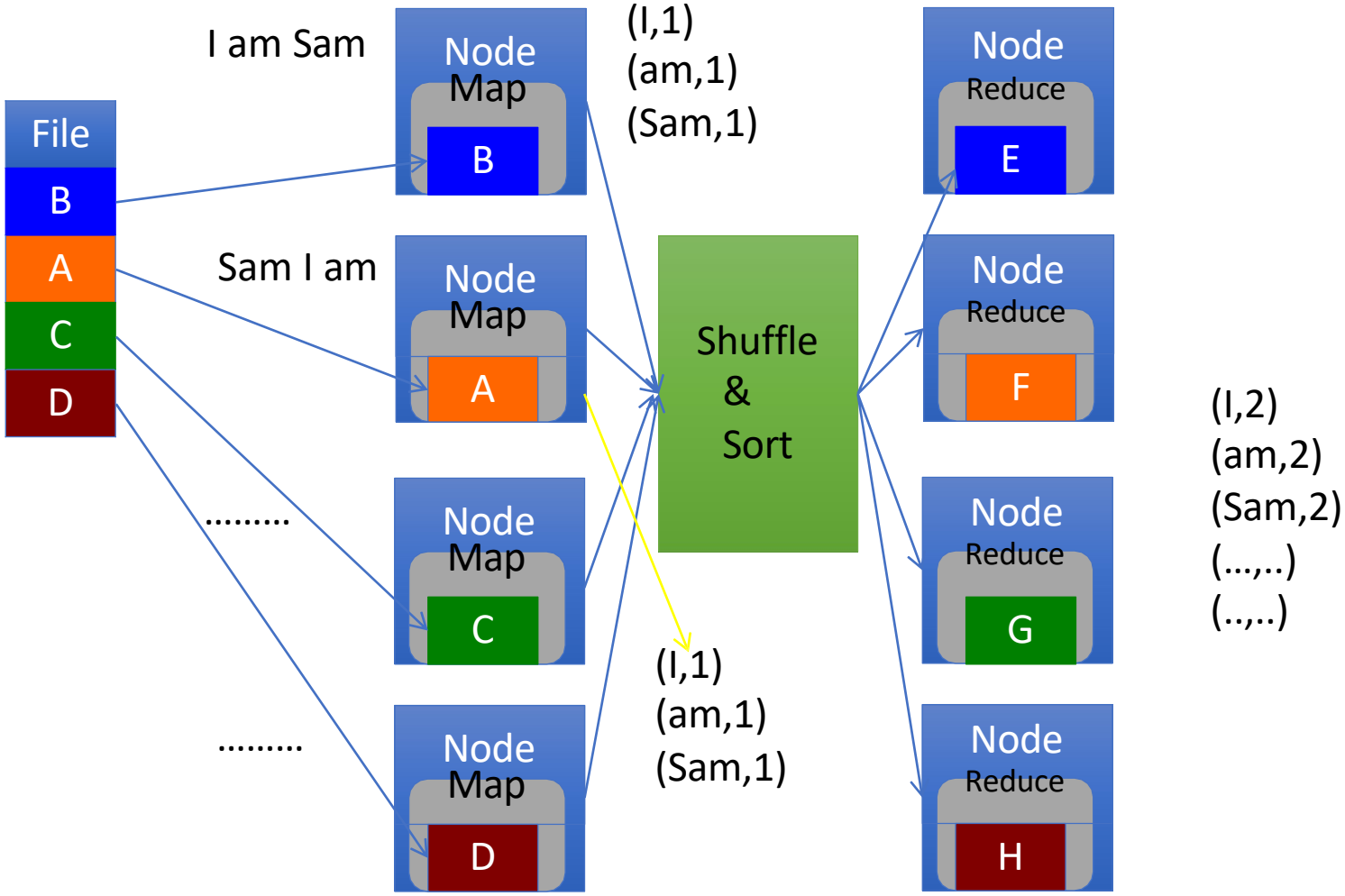
Map Reduce Paradigm

- Map and Reduce are based on functional programming

Map:	**Reduce:**
Apply a function to all the elements of List list1=[1,2,3,4,5]; square x = x * x list2=Map square(list1) print list2 -> [1,4,9,16,25]	Combine all the elements of list for a summary list1 = [1,2,3,4,5]; A = reduce (+) list1 Print A -> 15



MapReduce Word Count Example



Shortcoming of MapReduce

- Forces your data processing into Map and Reduce
  - Other workflows missing include join, filter, flatMap, groupByKey, union, intersection, …
- Based on “Acyclic Data Flow” from Disk to Disk (HDFS)
- Read and write to Disk before and after Map and Reduce (stateless machine)
  - Not efficient for iterative tasks, i.e. Machine Learning
- Only Java natively supported
  - Support for other languages needed
- Only for Batch processing
  - Interactivity, streaming data

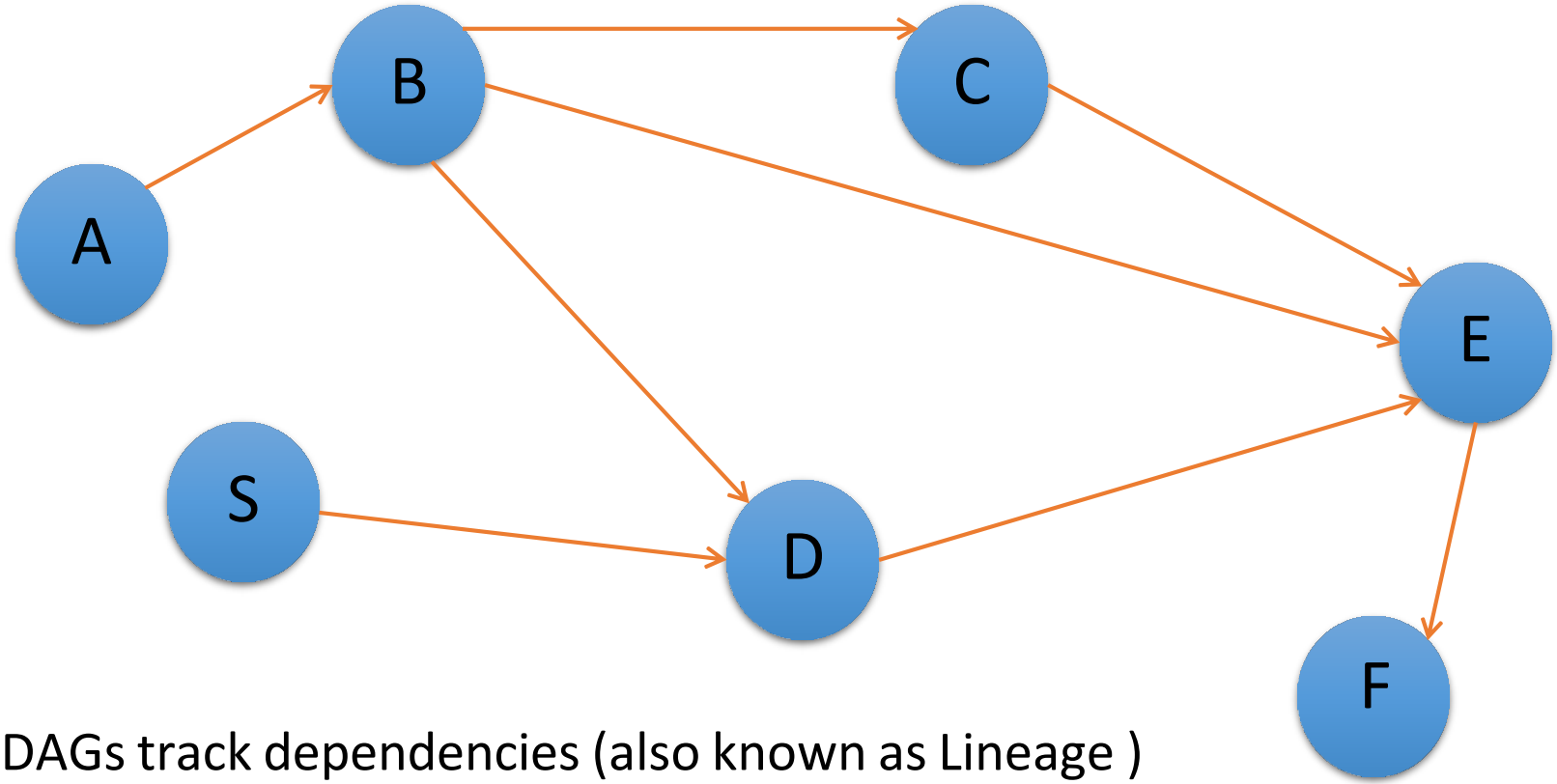
Challenges of Data Science

- Data preprocessing or data engineering (a majority of work)
- Iteration is a fundamental part of the data science
  - Stochastic gradient decent (SGE)
  - Maximum likely estimation (MLE)
  - Choose the right features, picking the right algorithms, running the right significance tests, finding the right hyperparameters, etc.
  - So data should be read once and stay in memory!
- Integration of models to real useful products
- Easy modeling but hard to work well in reality
  - R is slow and lack of integration capability
  - Java and C++ poor for exploratory analytics (lack of Read-Evaluate-Print-Loop)

Apache Spark

- Open source, originated from UC Berkeley AMPLab
- Distributed over a cluster of machines
- An elegant programming model
- Predecessor: MapReduce -> linear scalability and resilient to failures
- Improvement
  - Execution operations over a directed acyclic graph (DAG), unlike map-then-reduce, to keep data in memory rather than store in disks, like Microsoft's Dryad
  - A rich set of transformations and APIs to easy programming
  - In memory processing across server operations: Resilient Distributed Dataset (RDD)
  - Support Scala and Python APIs

Directed Acyclic Graphs (DAG)

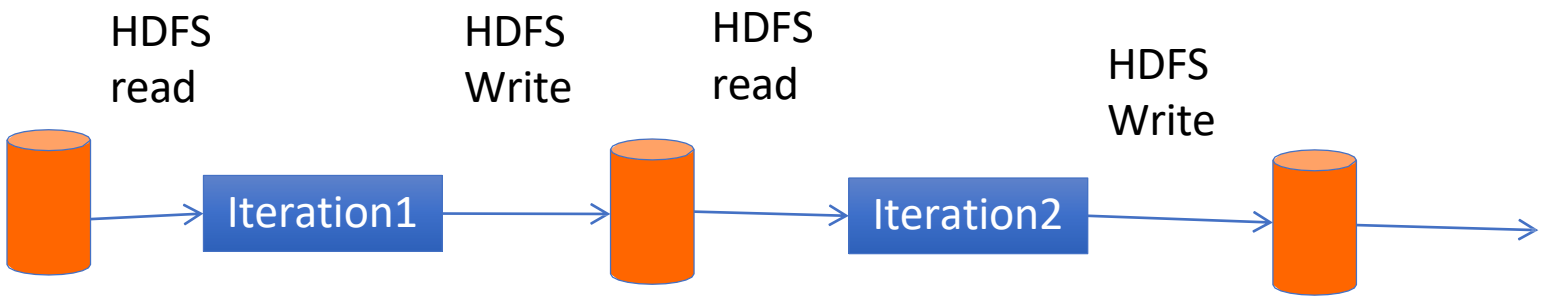


DAGs track dependencies (also known as Lineage)

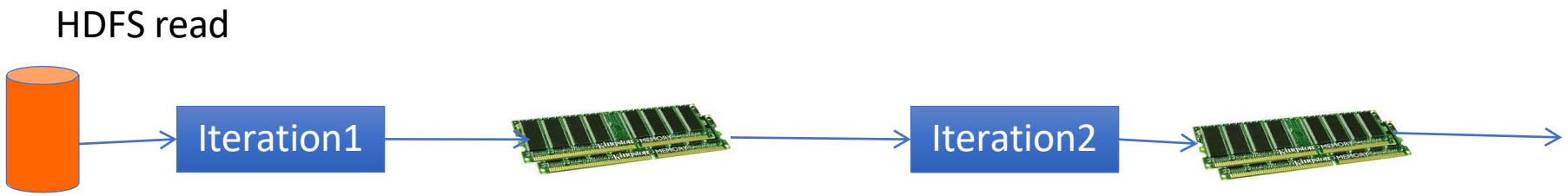
- nodes are RDDs
- arrows are Transformations

Spark Uses Memory instead of Disk

Hadoop: Use Disk for Data Sharing



Spark: In-Memory Data Sharing



Sort competition

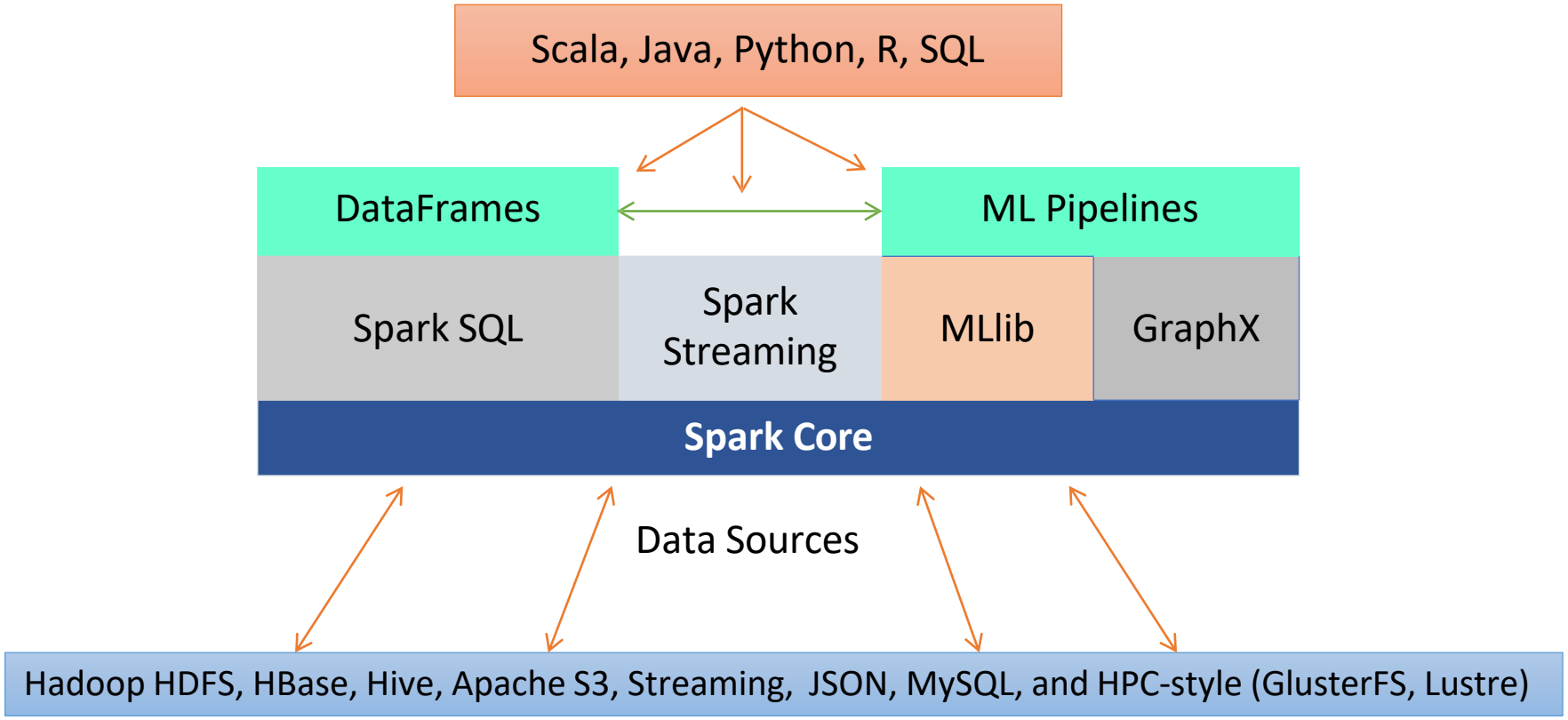
	Hadoop MR Record (2013)	Spark Record (2014)
Data Size	102.5 TB	100 TB
Elapsed Time	72 mins	23 mins
# Nodes	2100	206
# Cores	50400 physical	6592 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**

Sp ark, 3x faster with 1/10 the nodes

Sort benchmark, Daytona Gray: sort of 100 TB of data (1 trillion records)
http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html

Apache Spark

Apache Spark supports data analysis, machine learning, graphs, streaming data, etc. It can read/write from a range of data types and allows development in multiple languages.



Resilient Distributed Datasets (RDDs)

- RDDs (Resilient Distributed Datasets) is Data Containers
- All the different processing components in Spark share the same abstraction called RDD
- As applications share the RDD abstraction, you can mix different kind of transformations to create new RDDs
- Created by parallelizing a collection or reading a file
- Fault tolerant

So…

- Spark spans the gap between exploratory analytics and operational analytics systems.
- Data scientists are those who are better at engineering than most statisticians and those who are better at statistics than most engineers.
- Spark is built for performance and reliability from the ground up.
- It can read/write data formats supported by MapReduce, like Avro and Parquet. Works with NoSQL like Hbase and Cassandra. Spark streaming can ingest data from Flume and Kafka. SparkSQL interacts with the Hive Metastore.