HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

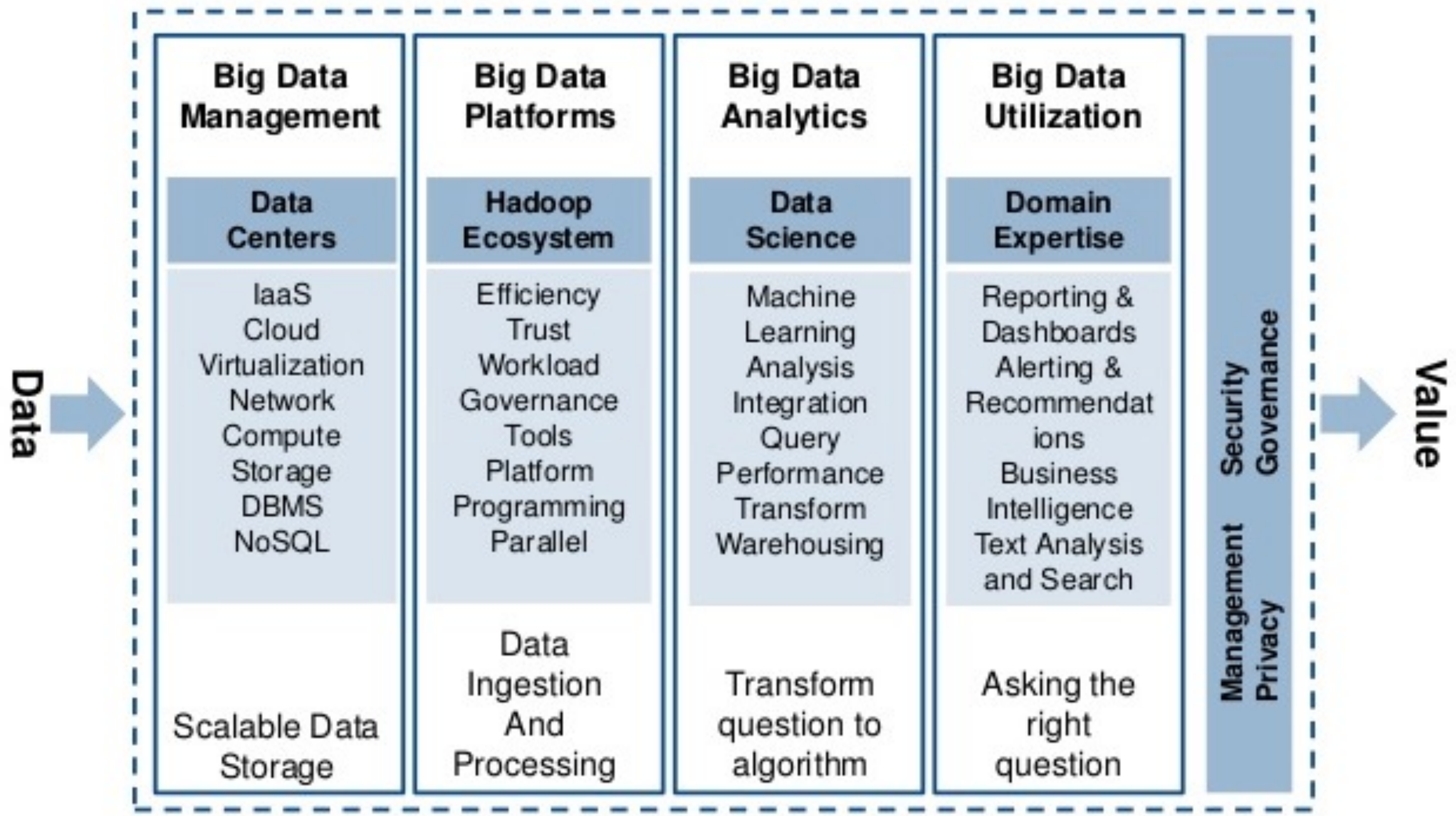
Introduction to
Data Science
(IT4142E)

Contents

Big data 5'V

Big data is a term for data sets that are so large or complex that traditional data processing application software is inadequate to deal with them (wikipedia)

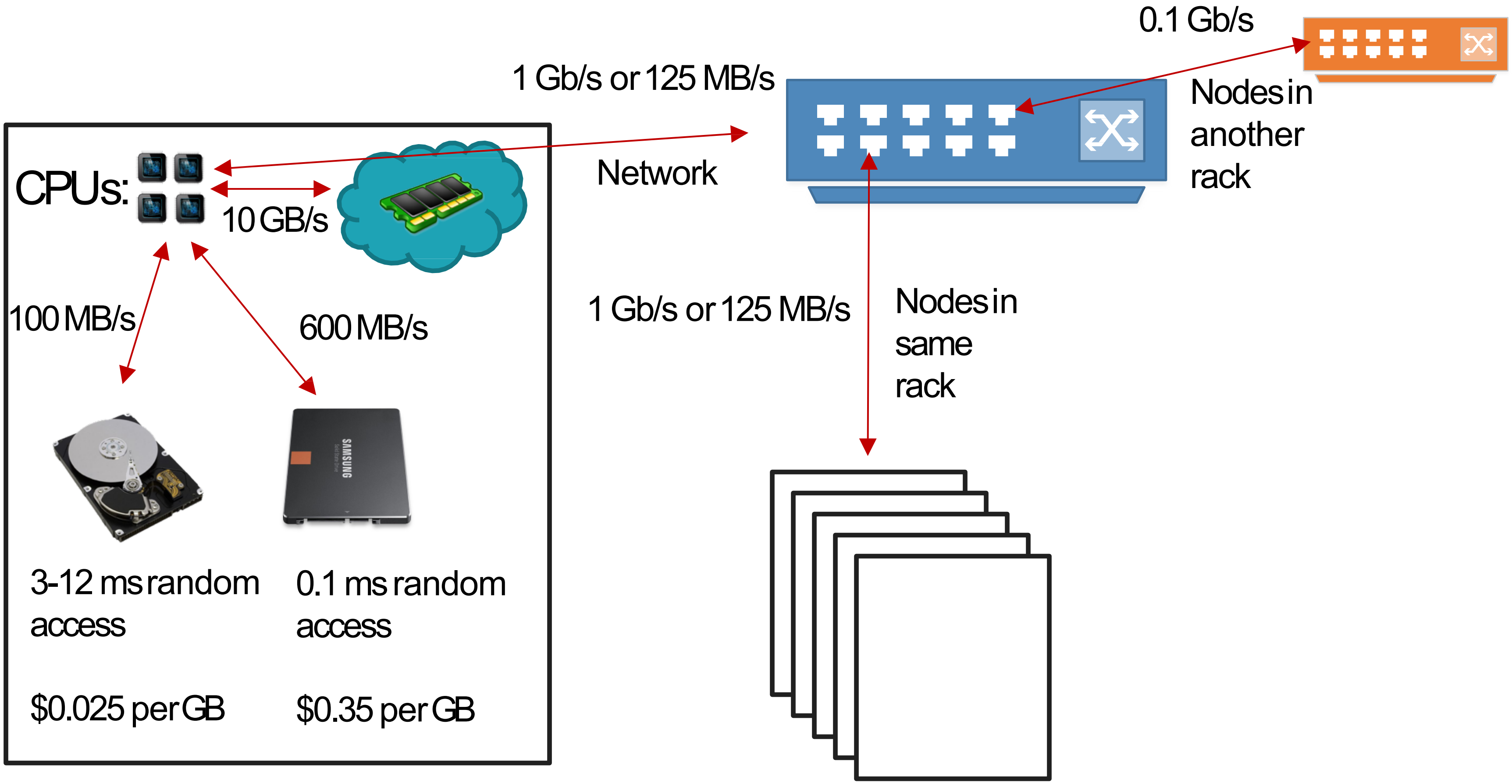
Big data technology stack



Scalable data management

- Scalability
  - Able to manage incresingly big volume of data
- Accessibility
  - Able to maintain efficiciency in reading and writing data (I/O) into data storage systems
- Transparency
  - In distributed environment, users should be able to access data over the network as easily as if the data were stored locally.
  - Users should not have to know the physical location of data to access it.
- Availability
  - Fault tolerance
  - The number of users, system failures, or other consequences of distribution shouldn't compromise the availability.

Data I/O landscape



Scalable data ingestion and processing

- Data ingestion
  - Data from different complementing information systems is to be combined to gain a more comprehensive basis to satisfy the need
  - How to ingest data efficiently from various, distributed heterogeneous sources?
    - Different data formats
    - Different data models and schemas
    - Security and privacy
- Data processing
  - How to process massive volume of data in a timely fashion?
  - How to process massive stream of data in a real-time fashion?
  - Traditional parallel, distributed processing (OpenMP, MPI)
    - Big learning curve
    - Scalability is limited
    - Fault tolerence is hard to achive
    - Expensive, high performance computing infrastructure

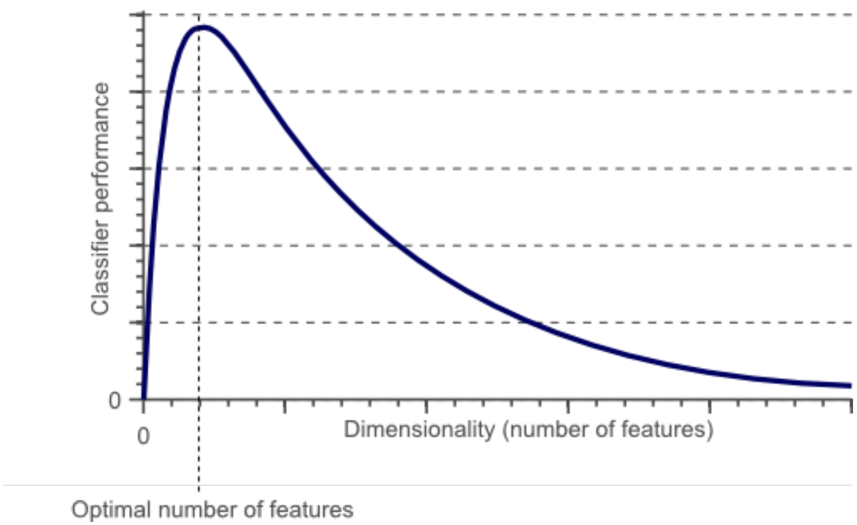
Scalable analytic algorithms

- Challenges
  - Big volume
  - Big dimensionality
  - Realtime processing
- Scaling-up Machine Learning algorithms
  - Adapting the algorithm to handle Big Data in a single machine.
    - Eg. Sub-sampling
    - Eg. Principal component analysis
    - Eg. feature extraction and feature selection
  - Scaling-up algorithms by parallelism
    - Eg. k-nn classification based on MapReduce
    - Eg. scaling-up support vector machines (SVM) by a divide and-conquer approach
  - Novel realtime processing architecture
    - Eg. Mini-batch in Spark streaming
    - Eg. Complex event processing in Apache Flink

Eg. Curse of dimensionality

- The required number of samples (to achieve the same accuracy) grows exponentionally with the number of variables!
- In practice: number of training examples is fixed!

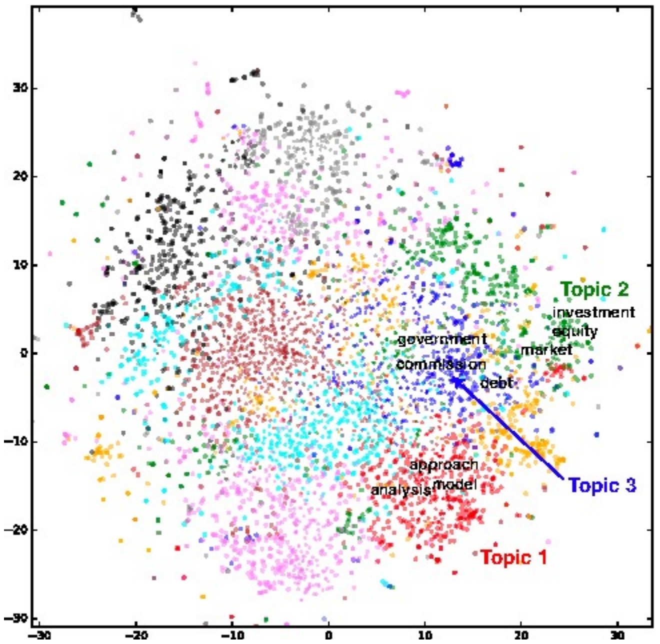
  => the classifier's performance usually will degrade for a large number of features!

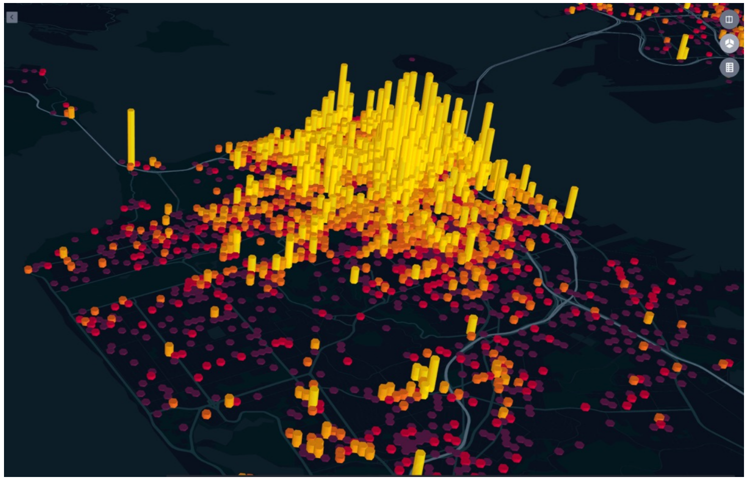


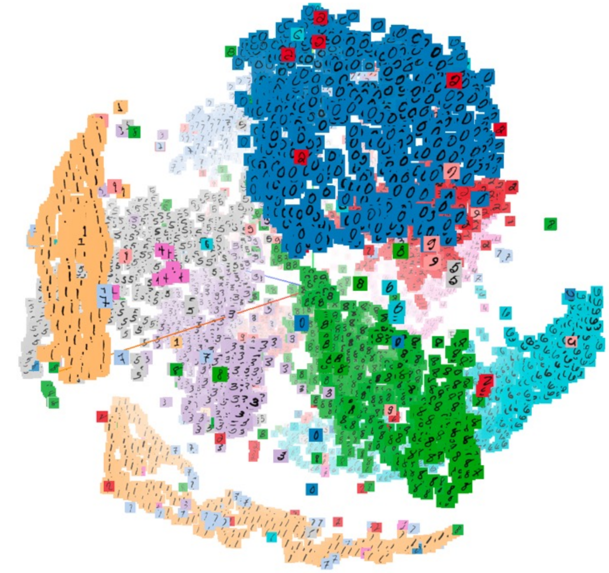
In fact, after a certain point, increasing the dimensionality of the problem by adding new features would actually degrade the performance of classifier.

Utilization and interpretability of big data

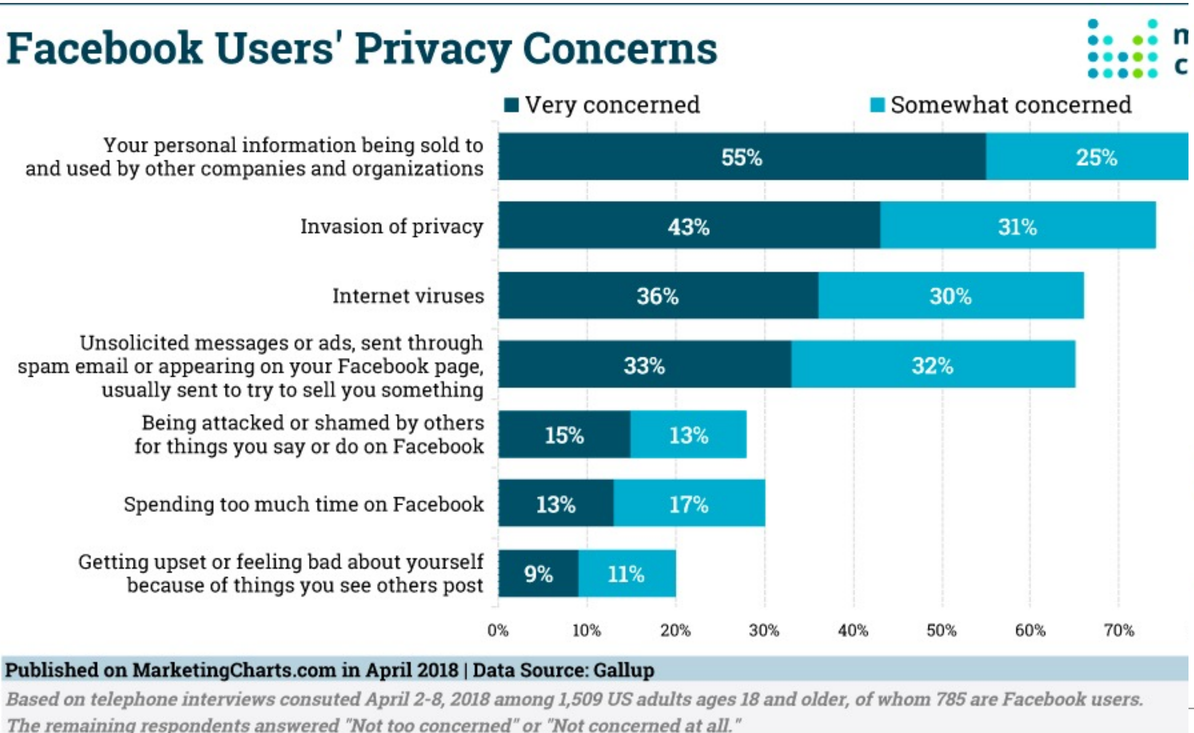
- Domain expertise to findout problems and interprete analytics results
- Scalable visualization and interpretability of million data points
  - to facilitate their interpretability and understanding

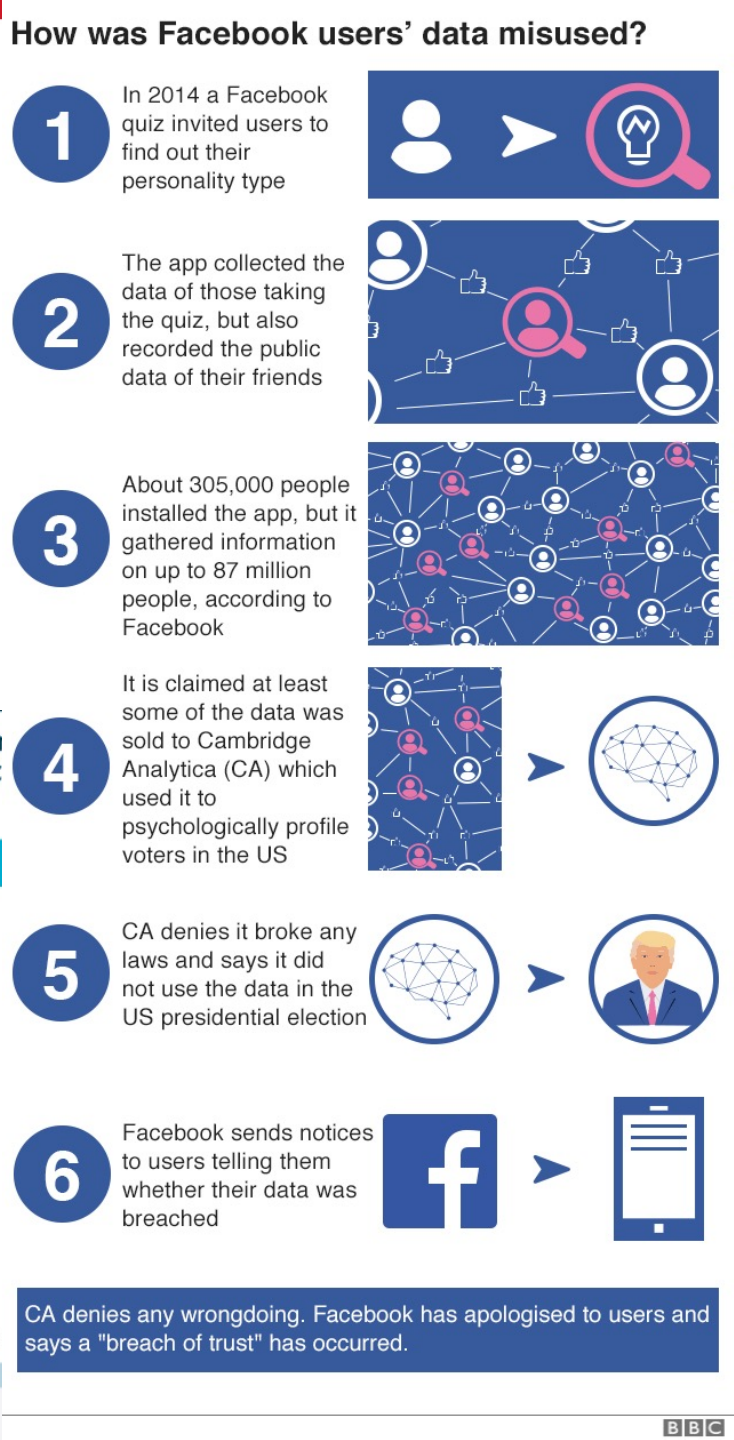






Privacy and security





Big data job trends

Talent shortage in big data

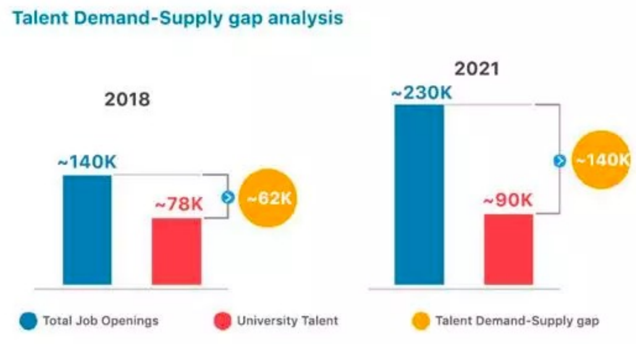
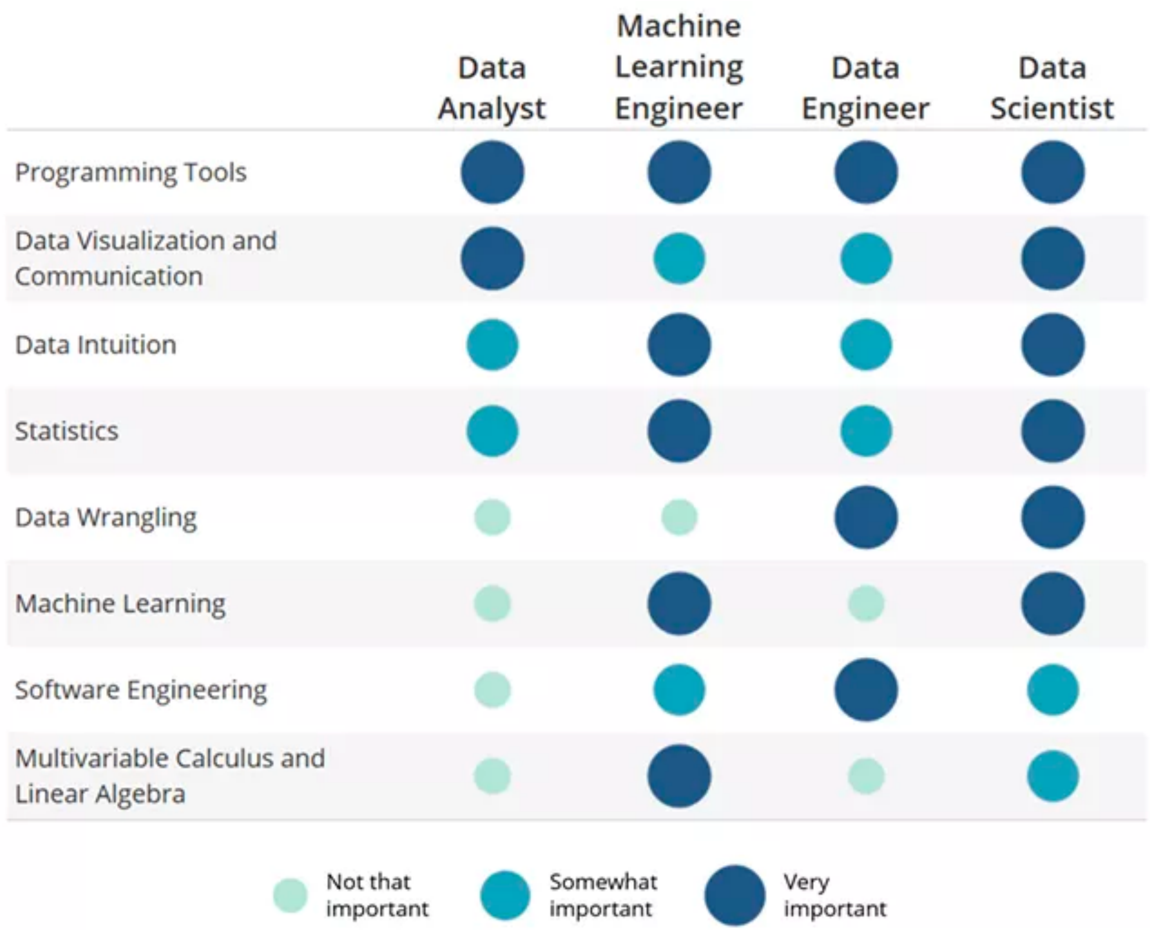


Table 2. Summary Demand Statistics

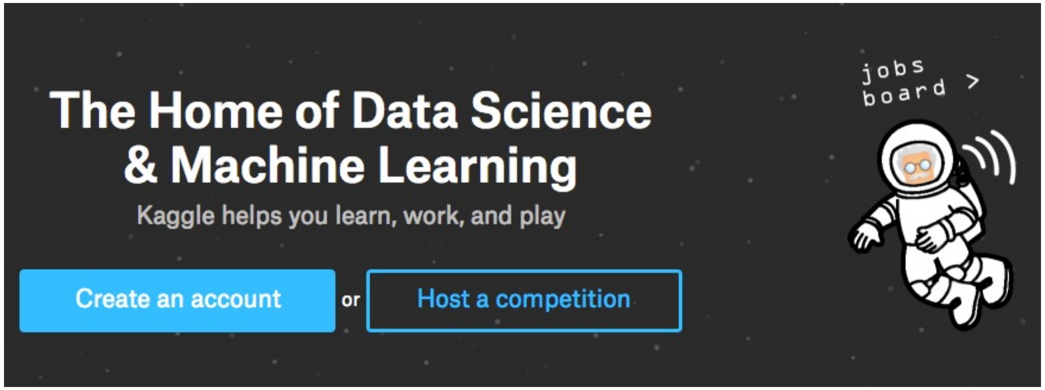
DSA Framework Category	Number of Postings in 2015	Projected 5-Year Growth	Estimated Postings for 2020	Average Time to Fill (Days)	Average Annual Salary
All	2,352,681	15%	2,716,425	45	$80,265
Data-Driven Decision Makers	812,099	14%	922,428	48	$91,467
Functional Analysts	770,441	17%	901,743	40	$69,162
Data Systems Developers	558,326	15%	641,635	50	$78,553
Data Analysts	124,325	16%	143,926	38	$69,949
Data Scientists & Advanced Analysts	48,347	28%	61,799	46	$94,576
Analytics Managers	39,143	15%	44,894	43	$105,909

Big data skill set

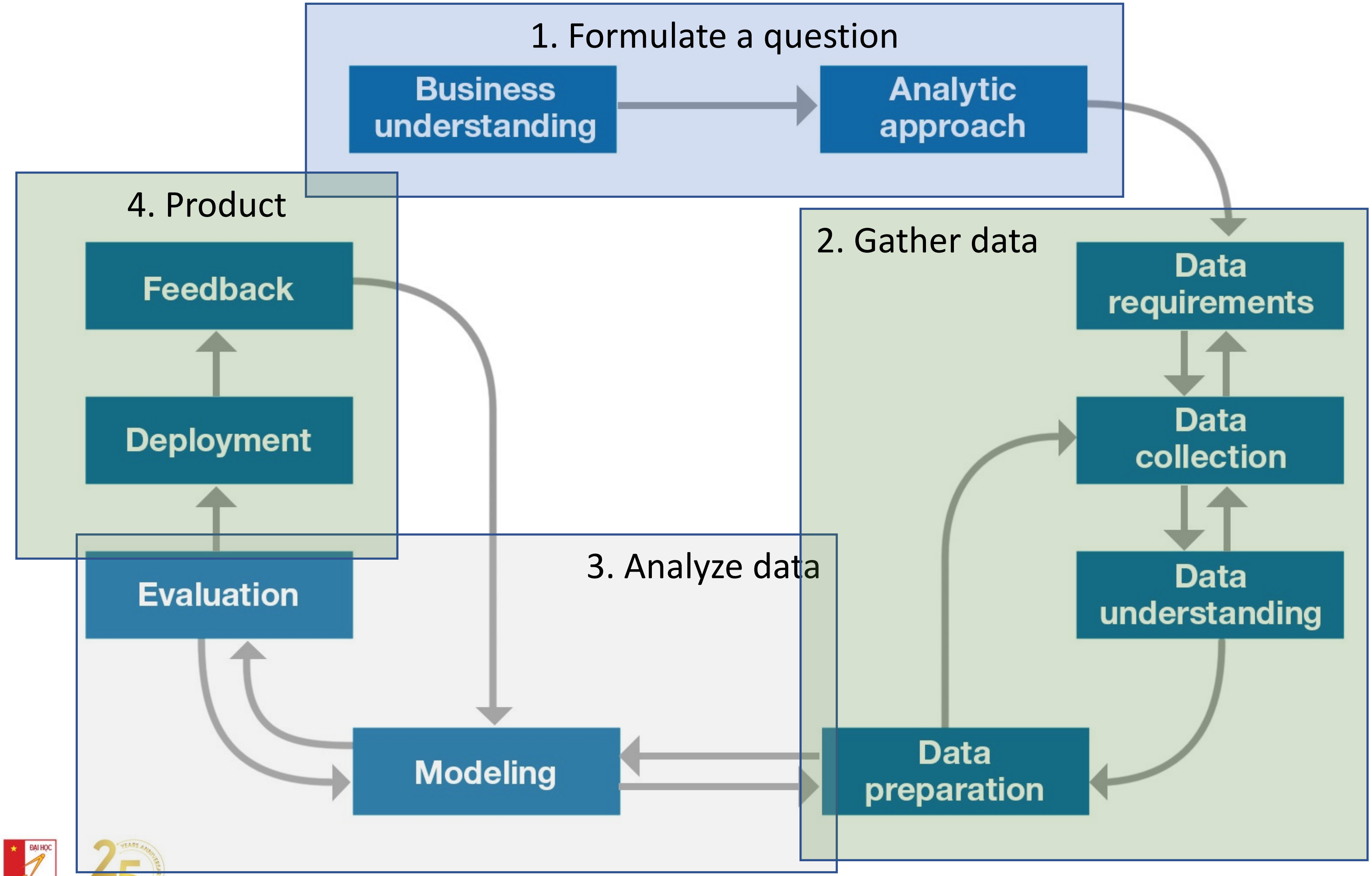


How to land big data related jobs

- Learn to code
  - Coursera
  - Udacity
  - Freecodecamp
  - Codecademy
- Math, Stats and machine learning
  - Kaggle
- Hadoop, NoSQL, Spark
- Visualization and Reporting
  - Tableau
  - Pentahoo
- Meetup & Share
- Find a mentor
- Internships, projects

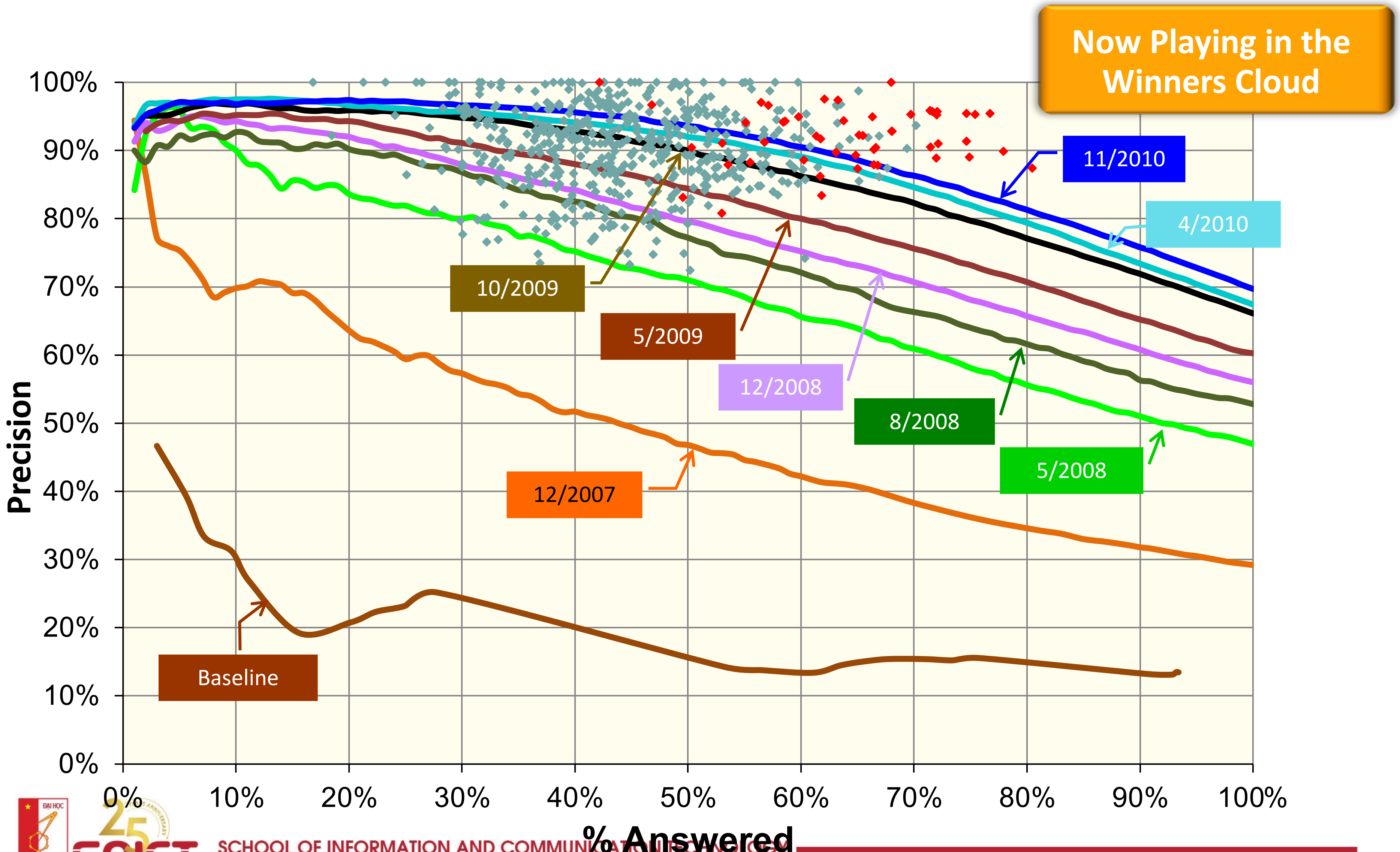


Data science method



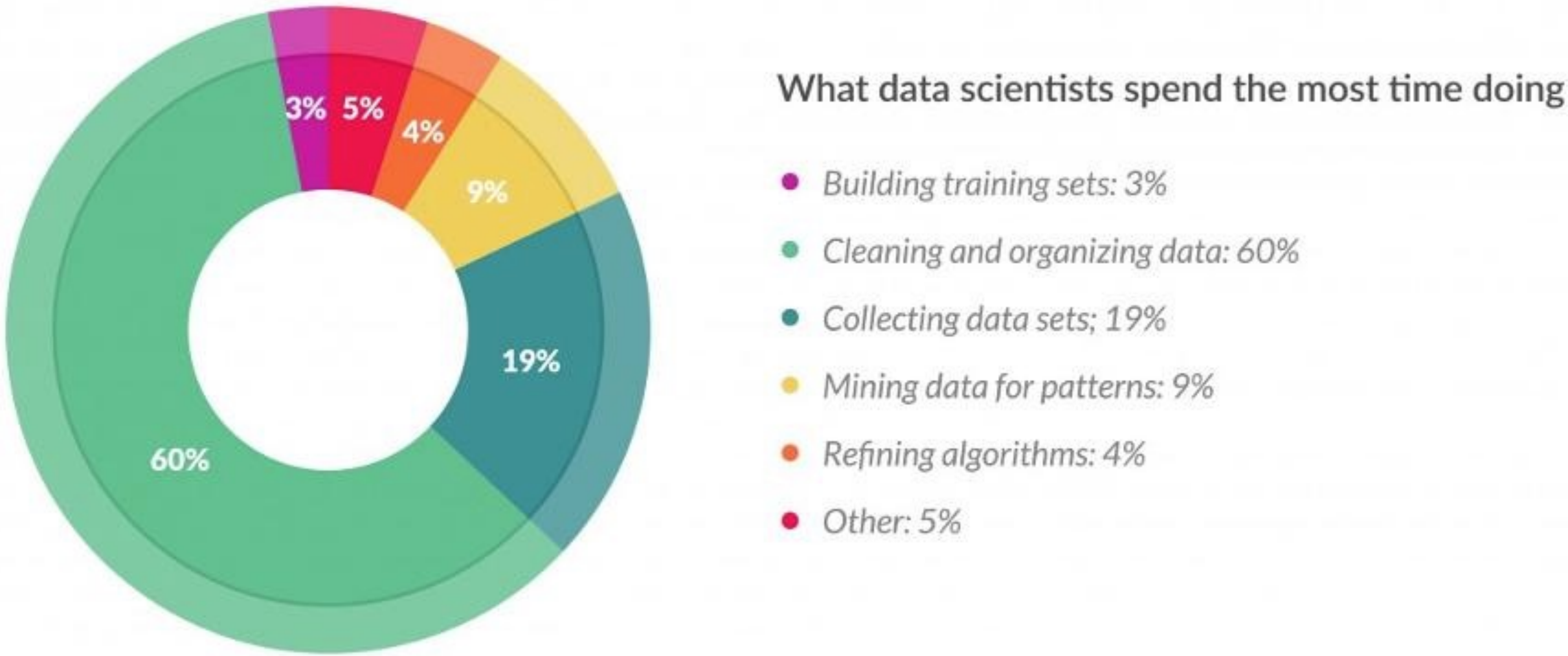
Source: *Foundational Methodology for Data Science*, IBM, 2015

DeepQA: Incremental Progress in Precision and Confidence 6/2007-11/2010



Cleaning big data: most time-consuming, least enjoyable data science task

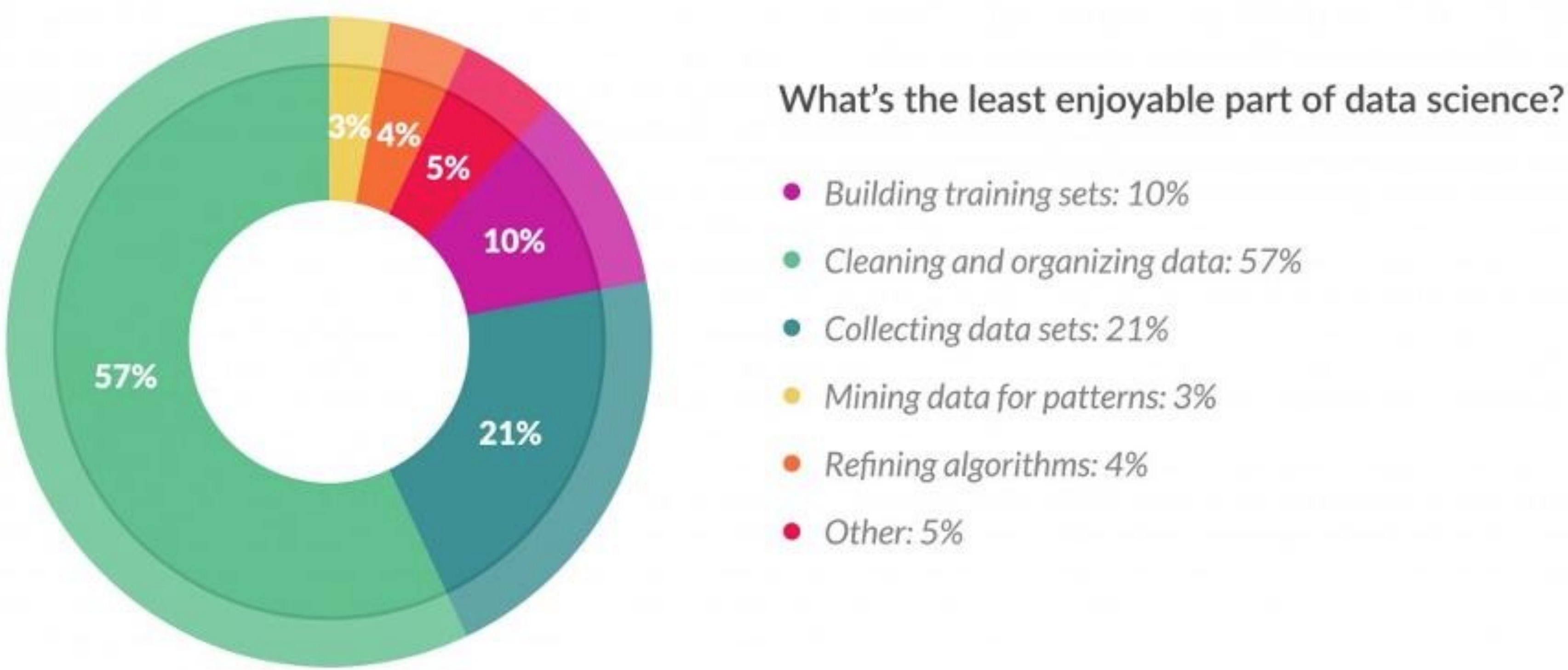
- Data preparation accounts for about 80% of the work of data scientists



source: https://www.forbes.com/

Cleaning big data: most time-consuming, least enjoyable data science task

- 57% of data scientists regard cleaning and organizing data as the least enjoyable part of their work and 19% say this about collecting data sets.



References

[1] Tiwari, Shashank. Professional NoSQL. John Wiley & Sons, 2011.

[2] Lam, Chuck. Hadoop in action. Manning Publications Co., 2010.

[3] Miner, Donald, and Adam Shook. MapReduce design patterns: building effective algorithms and analytics for Hadoop and other systems. " O'Reilly Media, Inc.", 2012.

[4] Karau, Holden. Fast Data Processing with Spark. Packt Publishing Ltd, 2013.

[5] Penchikala, Srini. Big data processing with apache spark. Lulu. com, 2018.

[6] White, Tom. Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.

[7] Gandomi, Amir, and Murtaza Haider. "Beyond the hype: Big data concepts, methods, and analytics." International Journal of Information Management 35.2 (2015): 137-144.

[8] Cattell, Rick. "Scalable SQL and NoSQL data stores." Acm Sigmod Record 39.4 (2011): 12-27.

[9] Gessert, Felix, et al. "NoSQL database systems: a survey and decision guidance." Computer Science-Research and Development 32.3-4 (2017): 353-365.

[10] George, Lars. HBase: the definitive guide: random access to your planet-size data. " O'Reilly Media, Inc.", 2011.

[11] Sivasubramanian, Swaminathan. "Amazon dynamoDB: a seamlessly scalable non-relational database service." Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012.

[12] Chan, L. "Presto: Interacting with petabytes of data at Facebook." (2013).

[13] Garg, Nishant. Apache Kafka. Packt Publishing Ltd, 2013.

[14] Karau, Holden, et al. Learning spark: lightning-fast big data analysis. " O'Reilly Media, Inc.", 2015.

[15] Iqbal, Muhammad Hussain, and Tariq Rahim Soomro. "Big data analysis: Apache storm perspective." International journal of computer trends and technology 19.1 (2015): 9-14.

[16] Toshniwal, Ankit, et al. "Storm@ twitter." Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 2014.

[17] Lin, Jimmy. "The lambda and the kappa." IEEE Internet Computing 21.5 (2017): 60-66.

Online courses

- https://www.coursera.org/learn/nosql-database-systems
- https://who.rocq.inria.fr/Vassilis.Christophides/Big/index.htm
- https://www.coursera.org/learn/big-data-introduction?specialization=big-data
- https://www.coursera.org/learn/big-data-integration-processing?specialization=big-data
- https://www.coursera.org/learn/big-data-management?specialization=big-data
- https://www.coursera.org/learn/hadoop
- https://www.coursera.org/learn/scala-spark-big-data

Hadoop ecosystem

We need a system that scales

- Traditional tools are overwhelmed
  - Slow disks, unreliable machines, parallelism is not easy
- 3 challenges
  - **Reliable storage**
  - **Powerful data processing**
  - Efficient visualization

What is Apache Hadoop?

- **Scalable** and **economical** data **storage** and **processing**
  - The Apache Hadoop software library is a framework that allows for the **distributed processing** of large data sets across clusters of computers using **simple programming models**. It is designed to **scale out** from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and **handle failures** at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures **(commodity hardware)**.
- Heavily inspired by Google data architecture

Hadoop main components

- Storage: Hadoop distributed file system (HDFS)
- Processing: MapReduce framework
- System utilities:
  - Hadoop Common: The common utilities that support the other Hadoop modules.
  - Hadoop YARN: A framework for job scheduling and cluster resource management.

Scalability

- Distributed by design
  - Hadoop can run on cluster
- Individual servers within a cluster are called nodes
  - each node may both store and process data
- Scale out by adding more nodes to increase scalability
  - Up to several thousand nodes

Fault tolerance

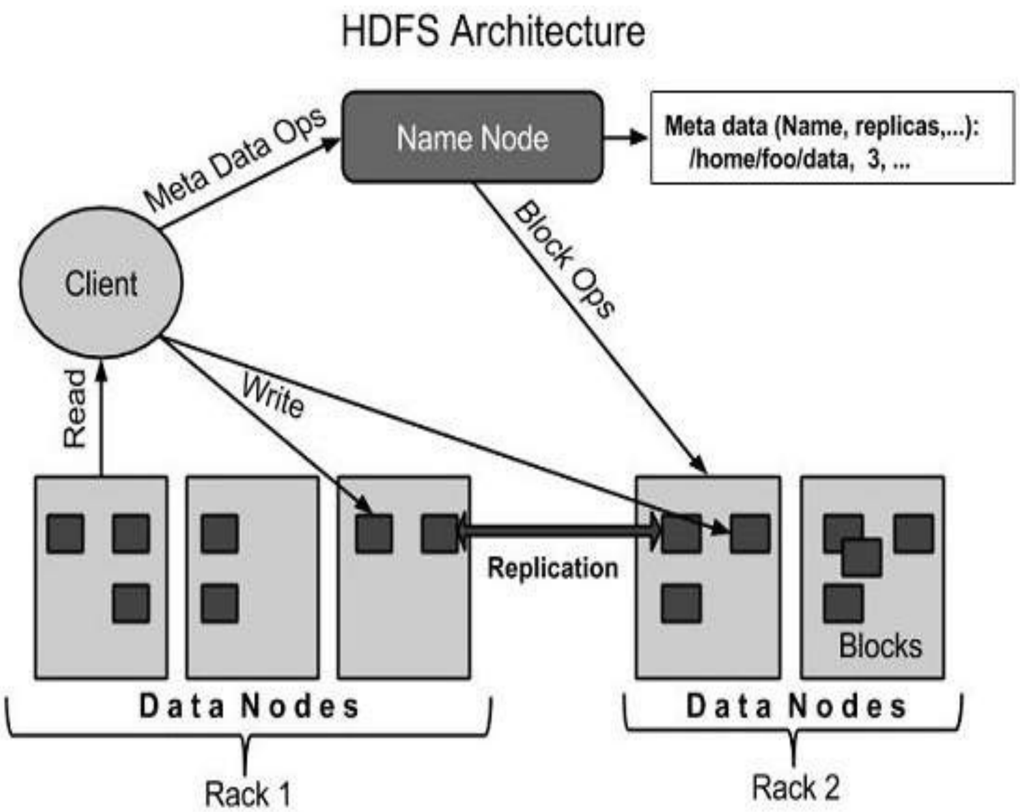
- Cluster of commodity servers
  - Hardware failure is the norm rather than the exception
  - Built with redundancy
- File loaded into HDFS are replicated across nodes in the cluster
  - If a node failed, its data is re-replicated using one of the copies
- Data processing jobs are broken into individual tasks
  - Each task takes a small amount of data as input
  - Parallel tasks execution
  - Failed tasks also get rescheduled elsewhere
- Routine failures are handled automatically without any loss of data

Hadoop distributed file system

- Provides inexpensive and reliable storage for massive amounts of data
- Optimized for big files (100 MB to several TBs file sizes)
- Hierarchical UNIX style file system
  - (e.g., /hust/soict/hello.txt)
  - UNIX style file ownership and permissions
- There are also some major deviations from UNIX
  - Append only
  - Write once read many times

HDFS Architecture

- Master/slave architecture
- HDFS master: namenode
  - Manage namespace and metadata
  - Monitor datanode
- HDFS slave: datanode
  - Handle read/write the actual data



HDFS main design principles

- I/O pattern
  - Append only → reduce synchronization
- Data distribution
  - File is splitted in big chunks (64 MB)

  → reduce metadata size

  → reduce network communication
- Data replication
  - Each chunk is usually replicated in 3 different nodes
- Fault tolerance
  - Data node: re-replication
  - Name node
    - Secondary namenode
    - Enqury data nodes instead of complex checkpointing scheme

Data processing: MapReduce

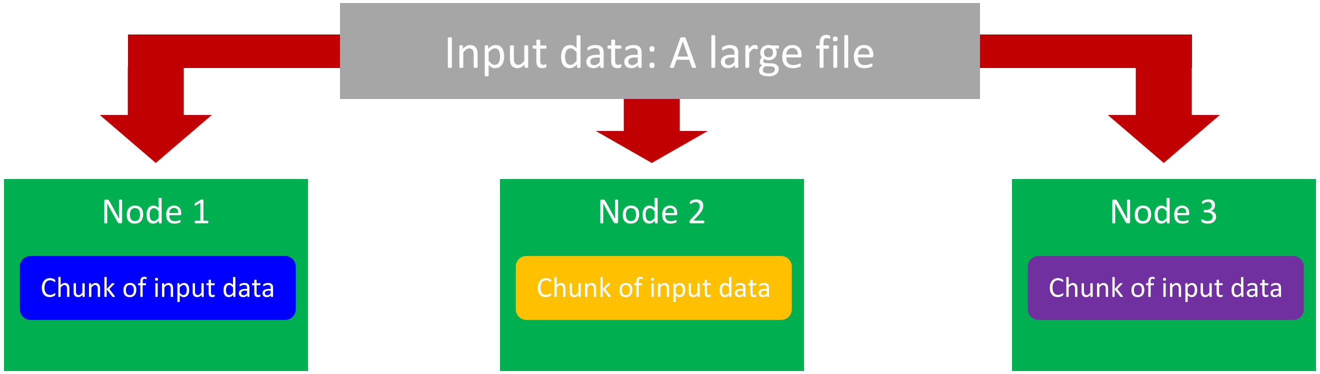
- MapReduce framework is the Hadoop default data processing engine
- MapReduce is a programming model for data processing
  - it is not a language, a style of processing data created by Google
- The beauty of MapReduce
  - Simplicity
  - Flexibility
  - Scalability

a MR job = {Isolated Tasks}n

- MapReduce divides the workload into multiple *independent tasks* and schedule them across cluster nodes
- A work performed by each task is done *in isolation* from one another for scalability reasons
  - The communication overhead required to keep the data on the nodes synchronized at all times would prevent the model from performing reliably and efficiently at large scale

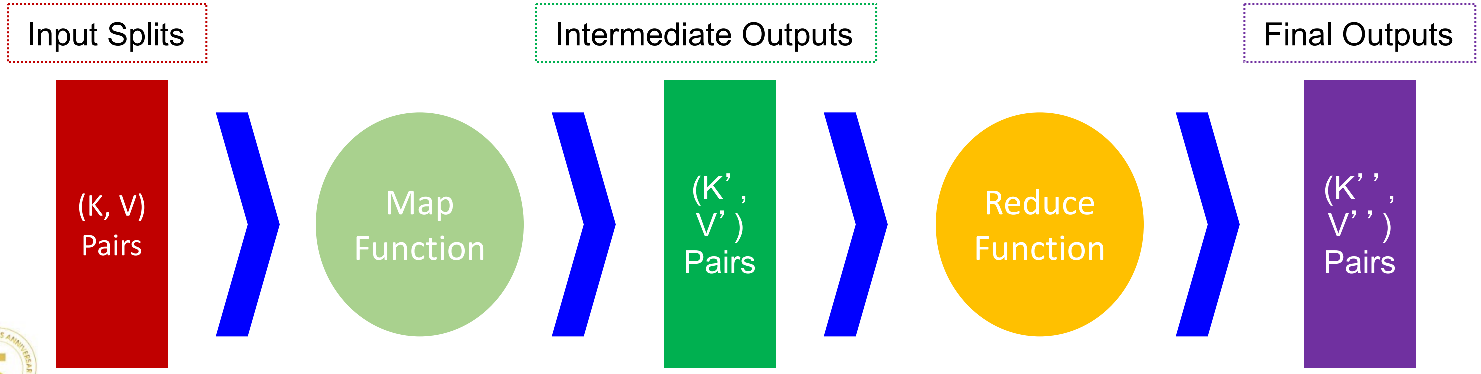
Data Distribution

- In a MapReduce cluster, data is usually managed by a distributed file systems (e.g., HDFS)
- Move code to data and not data to code



Keys and Values

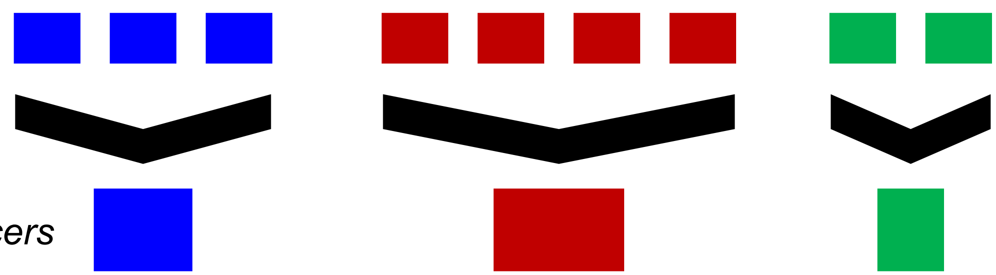
- The programmer in MapReduce has to specify two functions, the *map function* and the *reduce function* that implement the Mapper and the Reducer in a MapReduce program
- In MapReduce data elements are always structured as key-value (i.e., (K, V)) pairs
- The map and reduce functions receive and *emit* (K, V) pairs



Partitions

- A different subset of intermediate key space is assigned to each Reducer
- These subsets are known as *partitions*

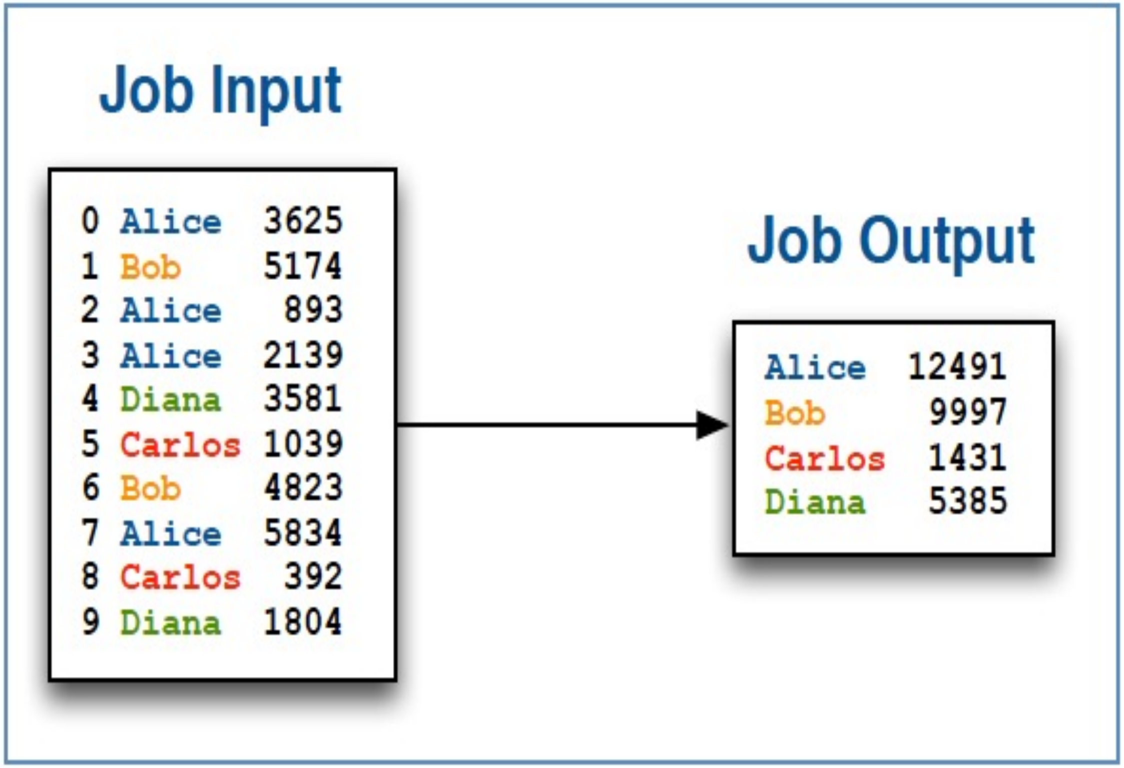
Different colors represent different keys (potentially) from different Mappers



Partitions are the input to Reducers

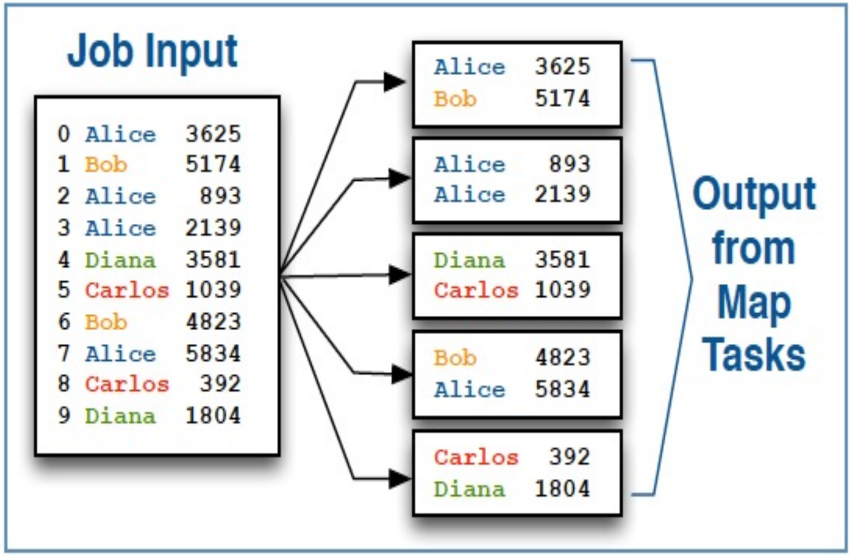
MapReduce example

- Input: text file containing order ID, employee name, and sale amount
- Output: sum of all sales per employee



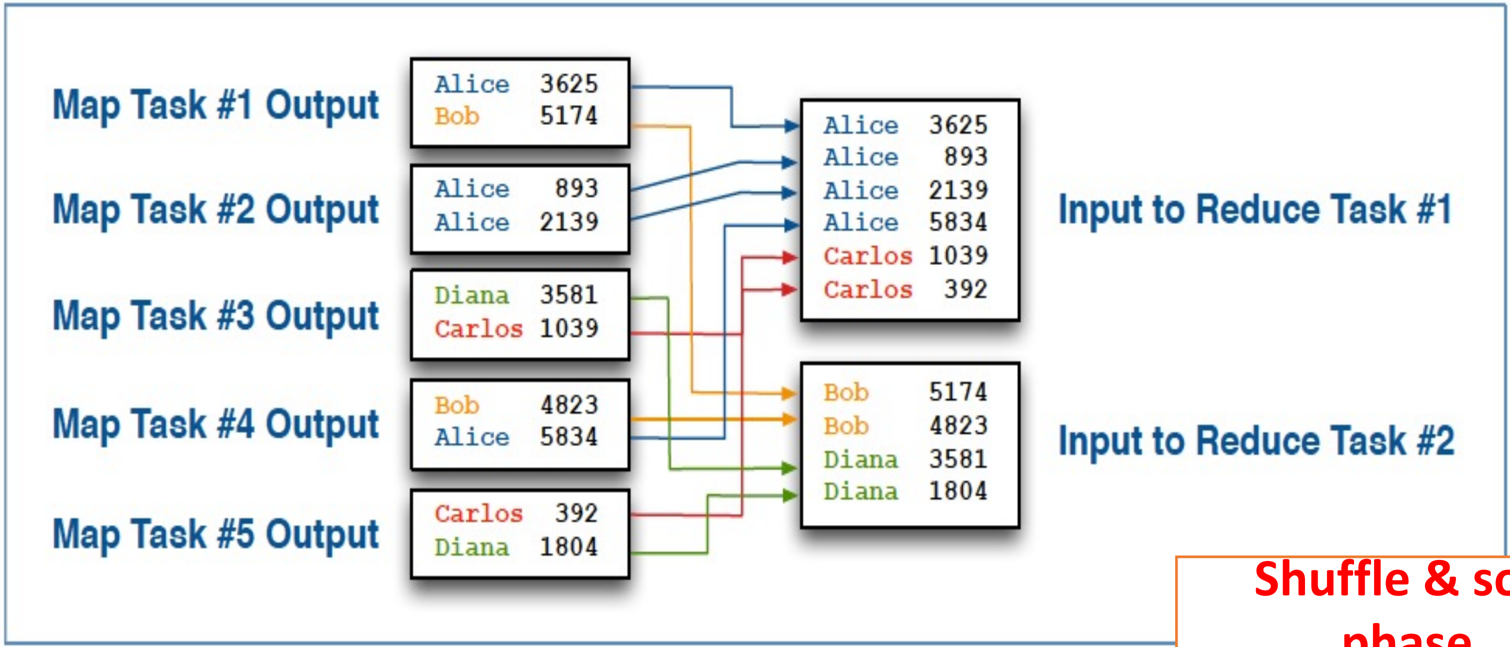
Map phase

- Hadoop splits job into many individual map tasks
  - Number of map tasks is determined by the amount of input data
  - Each map task receives a portion of the overall job input to process
  - Mappers process one input record at a time
  - For each input record, they emit zero or more records as output
- In this case, the map task simply parses the input record
  - And then emits the name and price fields for each as output



Map phase

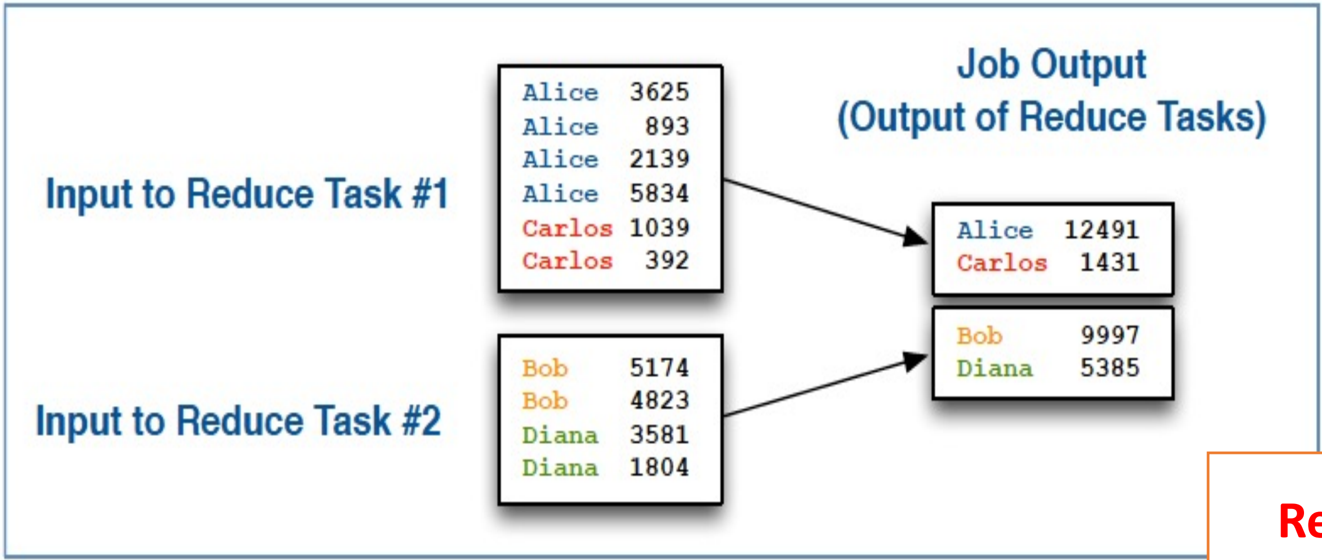
- Hadoop automatically sorts and merges output from all map tasks
  - This intermediate process is known as the **shuffle and sort**
  - The result is supplied to reduce tasks



Shuffle & sort phase

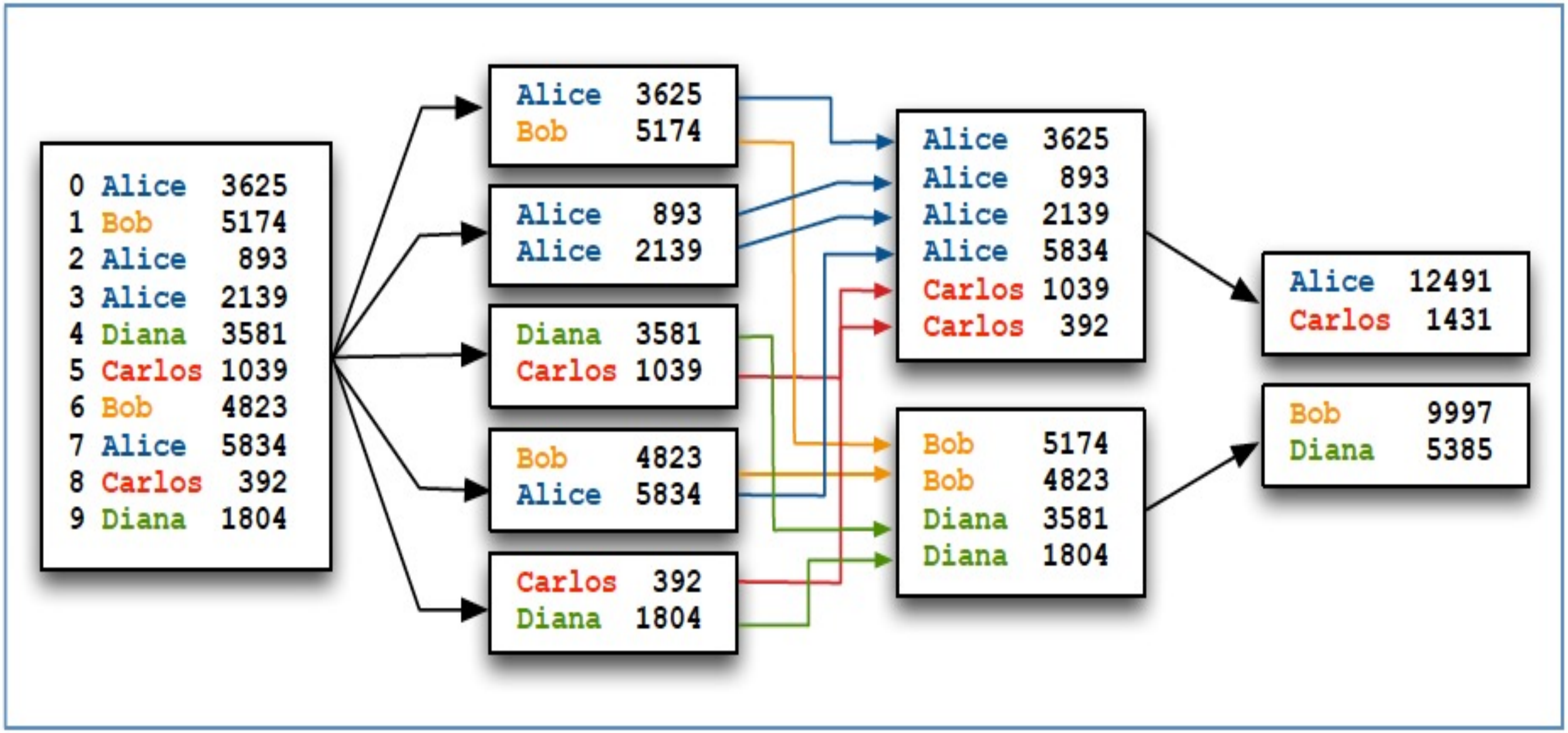
Reduce phase

- Reducer input comes from the shuffle and sort process
  - As with map, the reduce function receives one record at a time
  - A given reducer receives all records for a given key
  - For each input record, reduce can emit zero or more output records
- Our reduce function simply sums total per person
  - And emits employee name (key) and total (value) as output

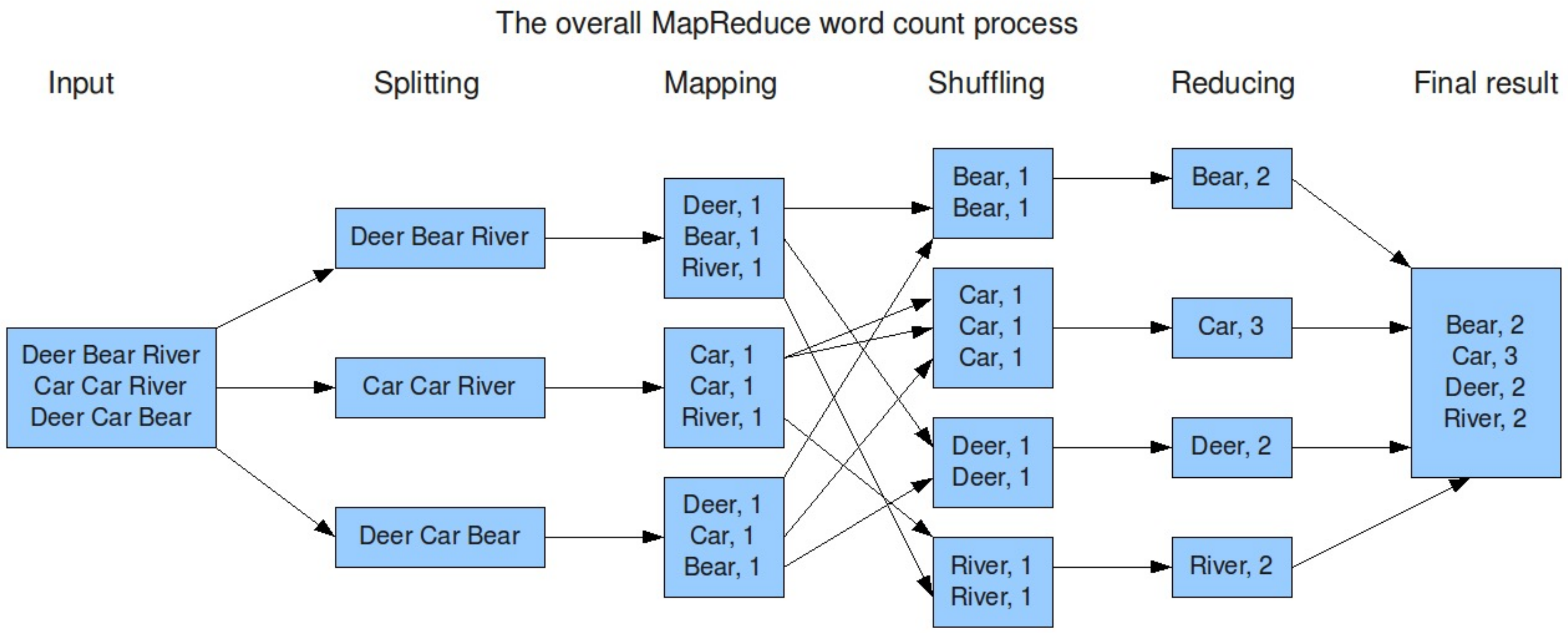


Reduce phase

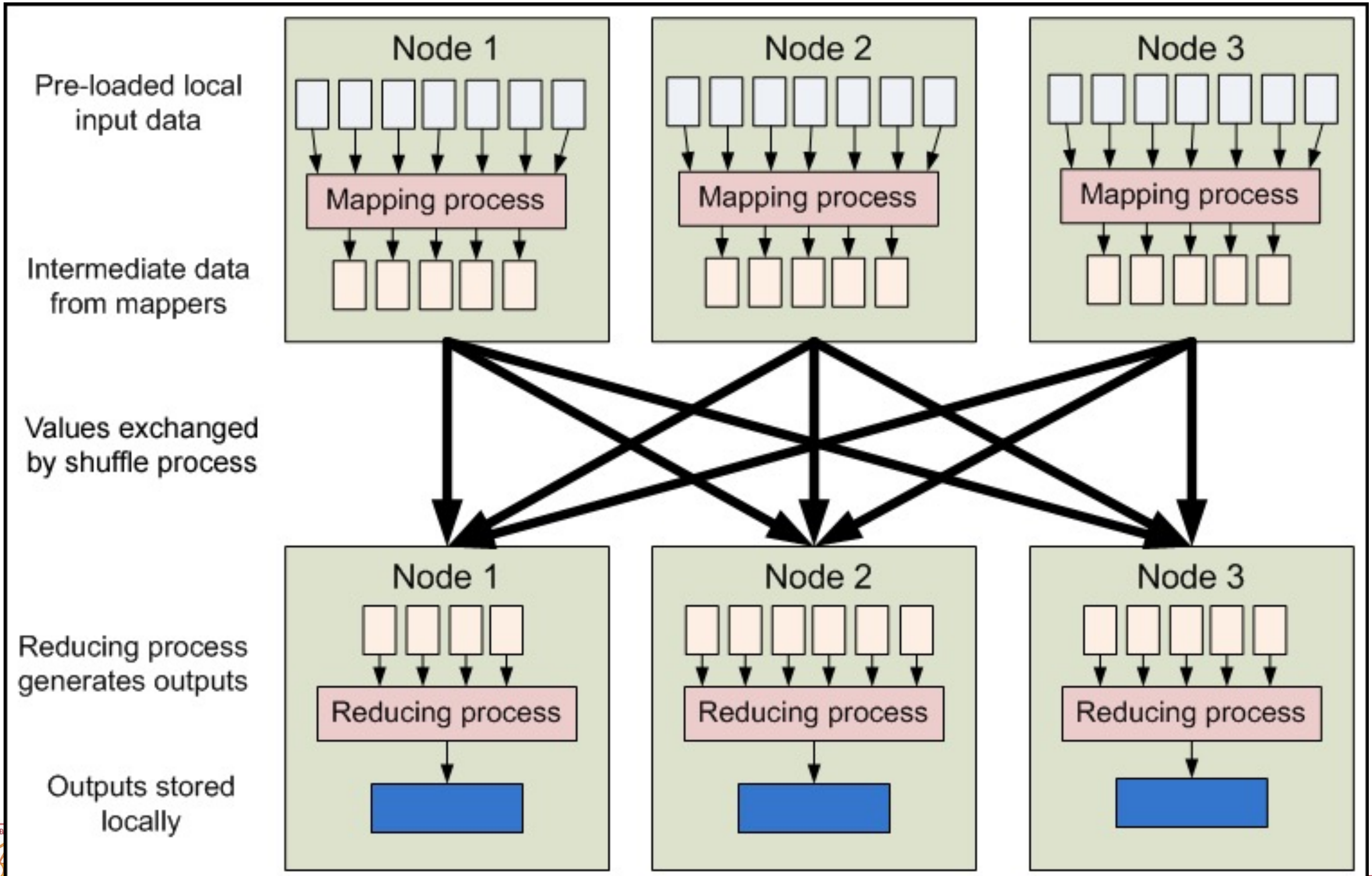
Data flow for the entire MapReduce job



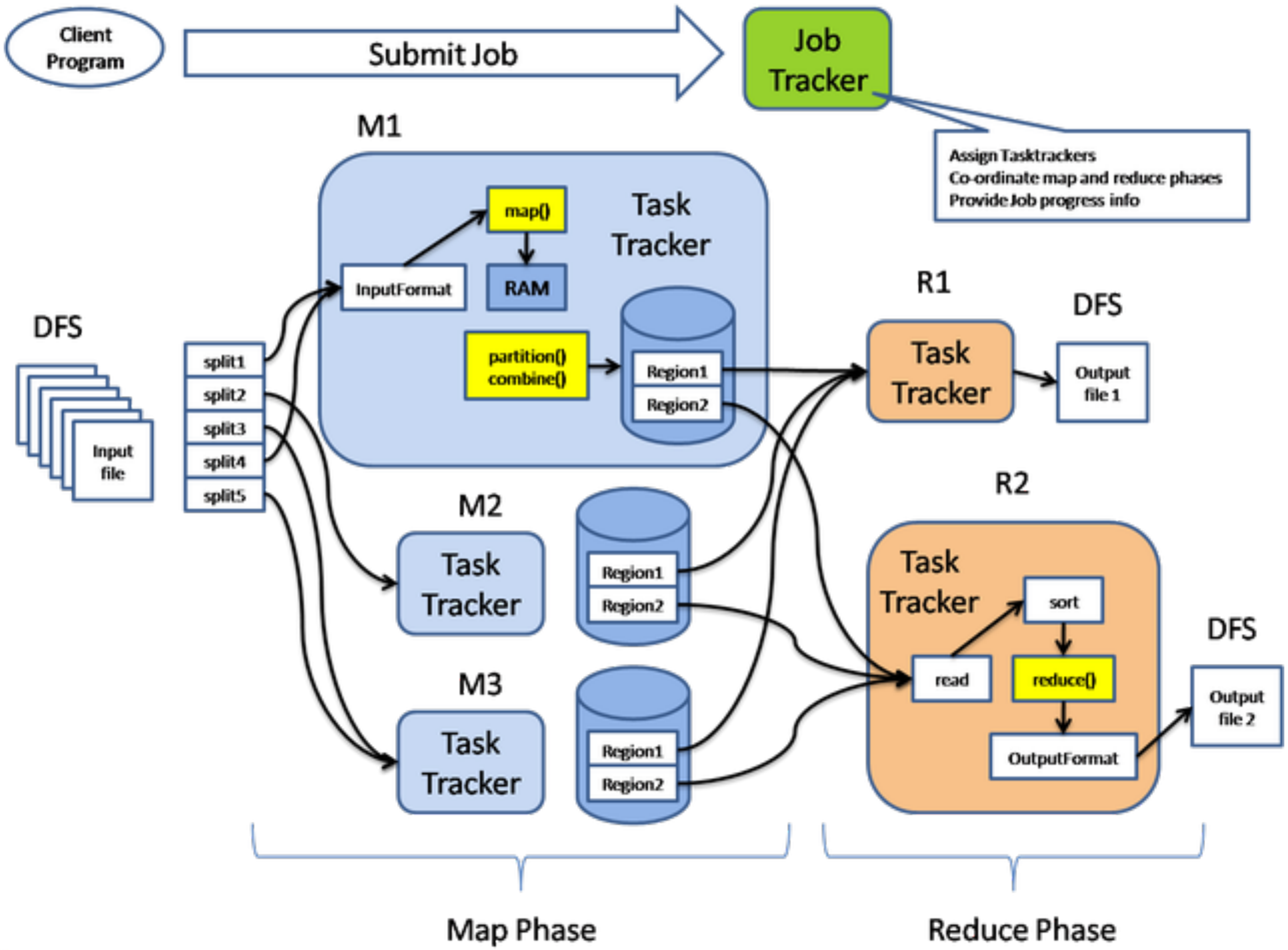
Word Count Dataflow



MapReduce - Dataflow



Map reduce life cycle



Hadoop ecosystem

- Many related tools integrate with Hadoop
  - Data analysis
  - Database integration
  - Workflow management
- These are not considered ‘core Hadoop’
  - Rather, they are part of the ‘Hadoop ecosystem’
  - Many are also open source Apache projects

Apache Pig

- Apache Pig builds on Hadoop to offer high level data processin
  - Pig is especially good at joining and transforming data
- The Pig interpreter runs on the client machine
  - Turns PigLatin scripts into MapReduce jobs
  - Submits those jobs to the cluster

```
people = LOAD '/user/training/customers' AS (cust_id, name);
orders = LOAD '/user/training/orders' AS (ord_id, cust_id, cost);
groups = GROUP orders BY cust_id;
totals = FOREACH groups GENERATE group, SUM(orders.cost) AS t;
result = JOIN totals BY group, people BY cust_id;
DUMP result;
```

Apache Hive

- Another abstraction on top of MapReduce
  - Reduce development time
  - HiveQL: SQL-like language
- The Hive interpreter runs on the client machine
  - Turns HiveQL scripts into MapReduce jobs
  - Submits those jobs to the cluster

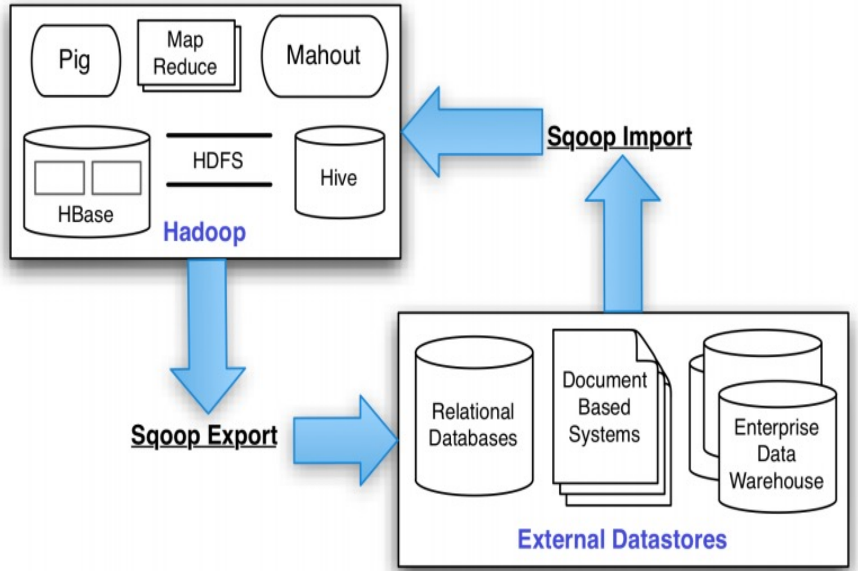
```
  SELECT customers.cust_id, SUM(cost) AS total
    FROM customers
    JOIN orders
      ON customers.cust_id = orders.cust_id
GROUP BY customers.cust_id
ORDER BY total DESC;
```

Apache Hbase

- HBase is a distributed column-oriented data store built on top of HDFS
  - Is considered as the Hadoop database
- Data is logically organized into tables, rows and columns
  - terabytes, and even petabytes of data in a table
  - Tables can have many thousands of columns
- Scales to provide very high write throughput
  - Hundreds of thousands of inserts per second
- Fairly primitive when compared to RDBMS
  - NoSQL : There is no high/level query language
  - Use API to scan / get / put values based on keys

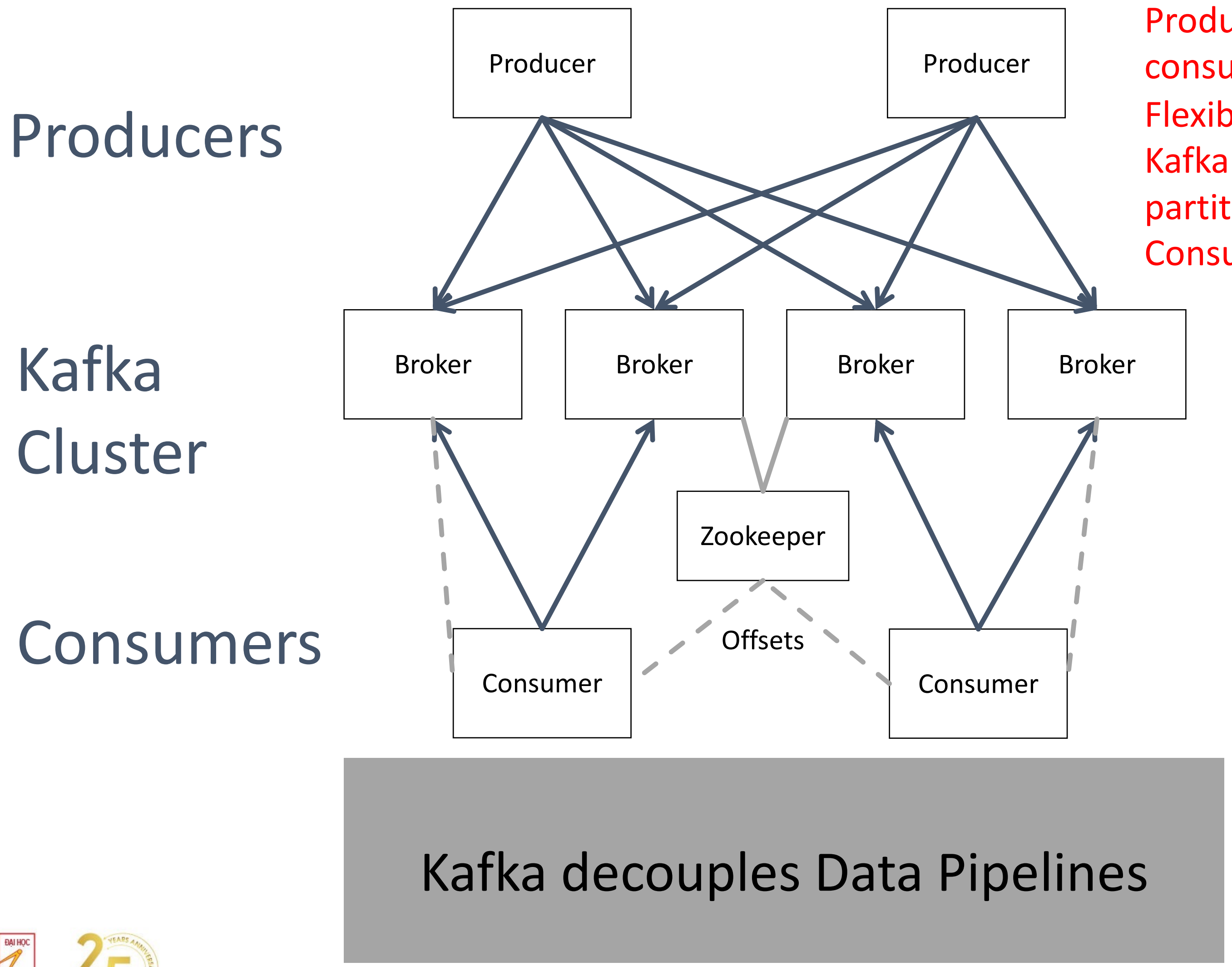
Apache sqoop

- Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.
- It can import all tables, a single table, or a portion of a table into HDFS
  - Via a Map/only MapReduce job
  - Result is a directory in HDFS containing comma/delimited text files
- Sqoop can also export data from HDFS back to the database



Apache Kafka

Kafka decouple data streams
Producers don’t know about consumers
Flexible message consumption
Kafka broker delegates log partition offset (location) to Consumers (clients)



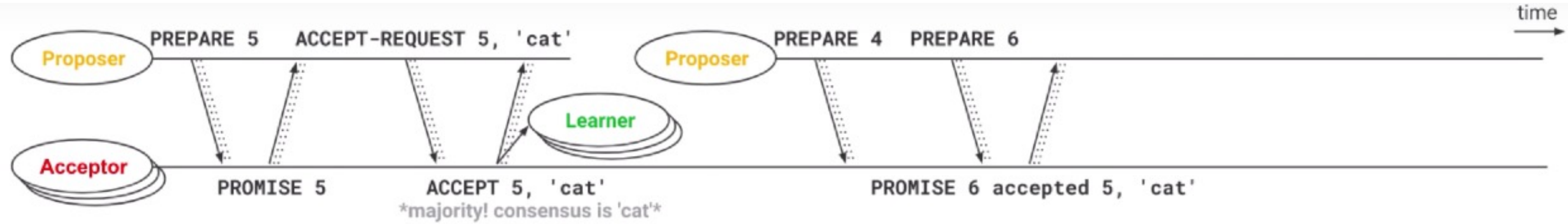
Apache Oozie

- Oozie is a workflow scheduler system to manage Apache Hadoop jobs.
- Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions.
- Oozie supports many workflow actions, including
  - Executing MapReduce jobs
  - Running Pig or Hive scripts
  - Executing standard Java or shell programs
  - Manipulating data via HDFS commands
  - Running remote commands with SSH
  - Sending e/mail messages

Apache Zookeeper

- Apache ZooKeeper is a highly reliable distributed coordination service
  - Group membership
  - Leader election
  - Dynamic Configuration
  - Status monitoring
- All of these kinds of services are used in some form or another by distributed applications

PAXOS algorithm



⇨ Proposer wants to propose a certain value:
It sends **PREPARE** <u>**IDp**</u> to a majority (or all) of **Acceptors**.
IDp must be unique, e.g. slotted timestamp in nanoseconds.
e.g. Proposer 1 chooses IDs 1, 3, 5...
Proposer 2 chooses IDs 2, 4, 6..., etc.
Timeout? retry with a new (higher) IDp.

⇨ **Acceptor** receives a **PREPARE** message for IDp:
Did it promise to ignore requests with this IDp?
Yes -> then ignore
No -> Will promise to ignore any request lower than IDp.
Has it ever accepted anything? (assume accepted ID=**IDa**)
Yes ->Reply with **PROMISE** <u>**IDp**</u> **accepted** <u>**IDa**</u>, <u>**value**</u>.
No -> Reply with **PROMISE** <u>**IDp**</u>.

1 **If a majority of acceptors promise, no ID<IDp can make it through.**

⇨ Proposer gets majority of **PROMISE** messages for a specific IDp:
It sends **ACCEPT-REQUEST** <u>**IDp**</u>, <u>**VALUE**</u> to a majority (or all) of **Acceptors**.
Has it got any already accepted value from promises?
Yes - > It picks the value with the highest **IDa** that it got.
No -> It picks any value it wants.

⇨ **Acceptor** receives an **ACCEPT-REQUEST** message for IDp, value:
Did it promise to ignore requests with this IDp?
Yes -> then ignore
No -> Reply with **ACCEPT** <u>**IDp**</u>, <u>**value**</u>. Also send it to all **Learners**.

2 **If a majority of acceptors accept <u>IDp</u>, <u>value</u>, consensus is reached. Consensus is and will always be on <u>value</u> (not necessarily IDp).**

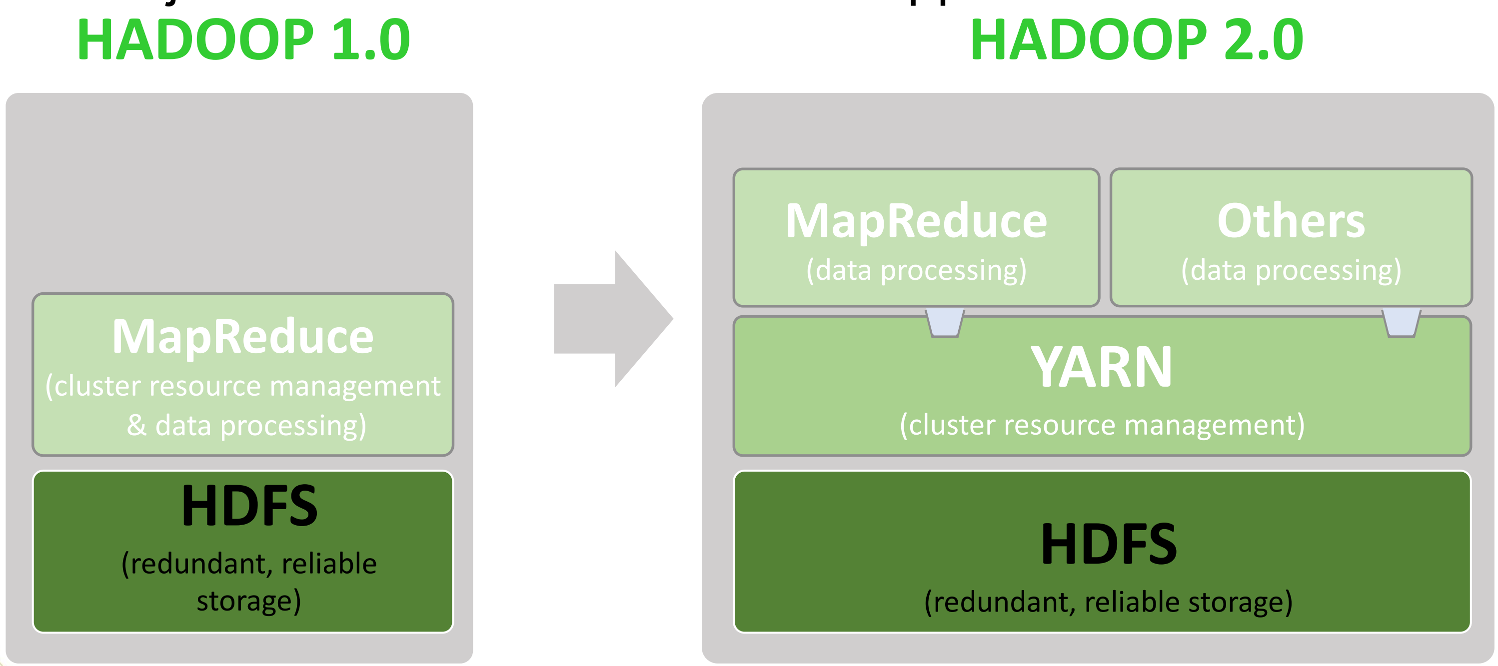
⇨ Proposer or **Learner** get **ACCEPT** messages for IDp, value:

3 **If a proposer/learner gets majority of accept for a specific <u>IDp</u>, they know that consensus has been reached on <u>value</u> (not IDp).**

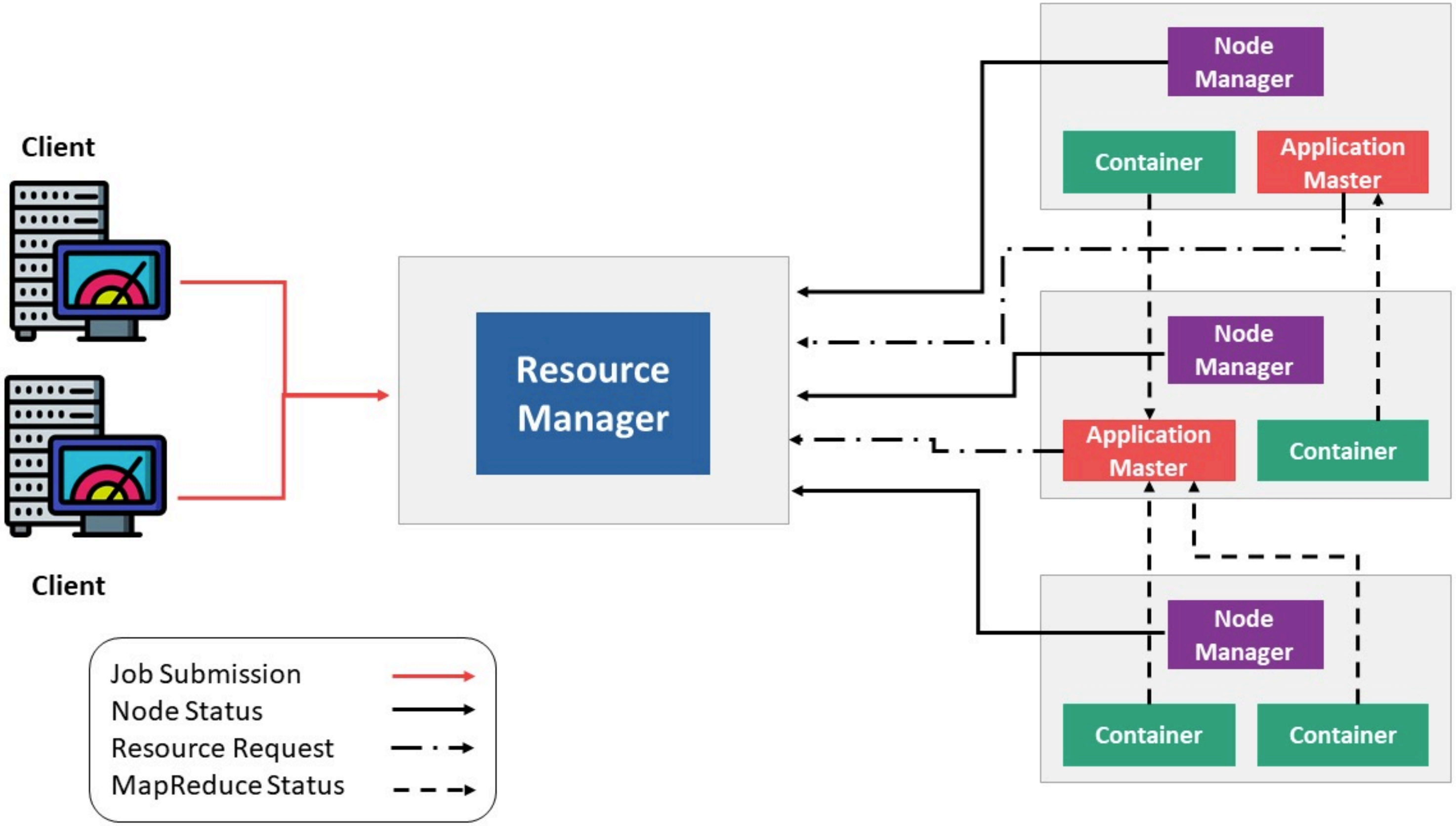
https://www.youtube.com/watch?v=d7nAGI_NZPk

YARN – Yet Another Resource Negotiator

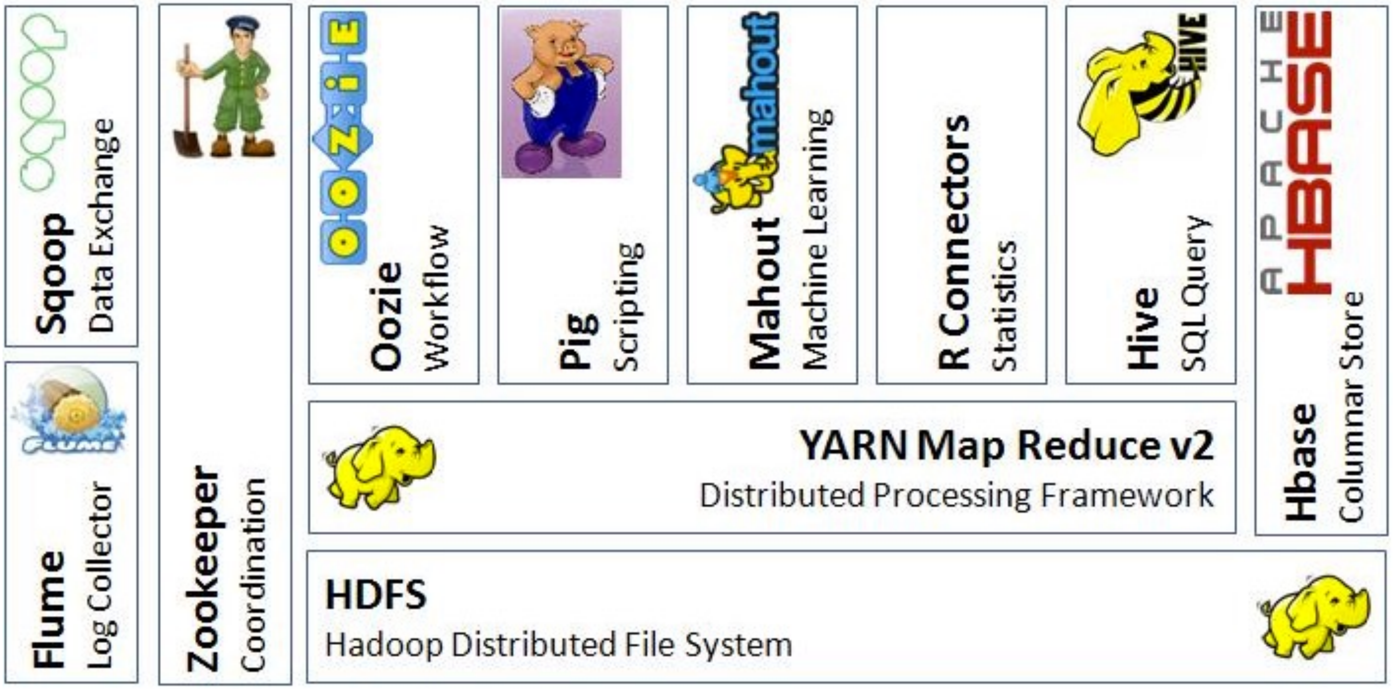
- Nodes have "resources" – memory and CPU cores – which are allocated to application when requested
- Moving beyond Map Reduce
  - MR and non-MR running on the same cluster
  - Most jobtracker functions moved to application masters



YARN execution



Big data platform: Hadoop ecosystem



Big data management

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!!!

soict.hust.edu.vn/ fb.com/groups/soict