Emergency data sources

INFO319 - Research Topics in Big data

Session – 3

19.09.2019

Recap

- Introduction to EM
- Introduction to Bigdata
- Presentations by you
- Essay topics
- Practical session

  - Apache Spark installation

  - Exercises?

Today’s outline

- Individual essay and programming projects topics
- EM data sources

  - open data

  - linked open data

  - open government data
- Big data architecture and technologies
- Practical session

  - Running Hadoop and MapReduce

  - Querying and analyzing government open data source by using Spark cluster
- Presentations by you

Individual essays

- The essay shall present and discuss selected theory, technology and tools related to big data technologies and EM, backed by scholarly and other references
  - counts 30% of final grade
  - presentations: December 5th
  - deadline: December 4rth 1400
  - **deadline for selection of the essay topic is 27.09.2019!**
- Encouraged:
  - more than a report

Some possible Essay Themes

- Types of Big Data challenges and analytical methods in terms of disaster management: A systematic literature review
- Exploring different visualization methods for social media big data: A systematic literature review
- Exploring different machine learning approaches for natural disaster management: A systematic literature review
- Evaluating different machine learning approaches for man-made disaster management
- Social media analytics for natural disaster management
- The Rising Role of Big Data Analytics and IoT in Disaster Management: Recent Advances, Taxonomy and Prospects
- How social media enhances emergency situation awareness?
- Discovering Big data Technologies for natural disaster management: recent research and future directions
- Discovering Big data Technologies for man-made disaster management: recent research and future directions
- Internet of Things (IoT) Considerations, Requirements, and Architectures for Disaster Management System
- Exploring IoT Applications for Disaster Management: recent research and future directions

Student group programming project

- The project shall develop an application that can be used for emergency management. Development and run-time platform is free choice, as is programming language. The project should be carried out in groups of three and not more. Working individually is allowed only if you come up with your own proposal idea.
  - Counts 40% of final grade.
  - Final presentation: Friday December 6th
  - Submission deadline: December 13th, 1400
  - **Optional deadline:** Friday September 27th.
  - **Deadline for selection of the project and group: Oct 4rth, 2019, kl.14.00**

Project Topics

1. Detecting trending topics over the social media for emergency management.
2. Social media analytics for disaster management
3. Deep Learning for emergency management Using Social Media Information.
4. Classification of crisis–related data on social networks (Twitter data).
5. Develop an application to identify fake news from Twitter data during disasters.
6. Detecting Hurricane information through Twitter: The 2019 Hurricane Dorian disaster in USA.
7. Sentiment analysis during Hurricane Dorian in emergency response
8. Disaster early warning and damage assessment analysis using social media data and geo-location information.
9. Social media data and post-disaster recovery
10. Processing Social Media Images by using machine learning algorithms.
11. Classifying and Summarizing Information from Microblogs During Epidemics

BDEM workshop

- Annual BDEM meeting will be held in Sogndal on October 21-22, 2019
- Free hotel rooms and lunches and 1 dinner
- We need your itinerary.
- Register using this link **https://forms.gle/apD3f6yCT7sBb4EX6**
- Deadline: on or before 25 September.

Lecture outline

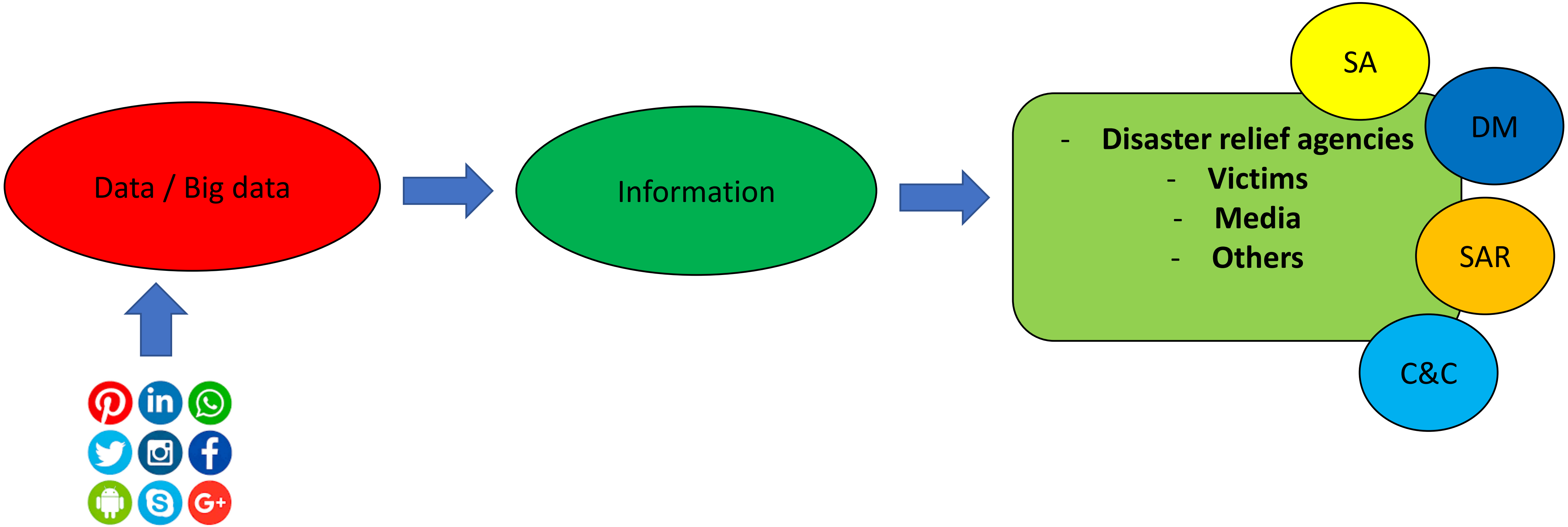
- Individual essay and programming projects
- Available data sources for EM
  - open data
  - linked open data
  - open government data
- Big data architecture and technologies
- Presentations by you
- Practical session
  - Running Hadoop and MapReduce
  - Querying and analyzing government open data source by using Spark cluster

Data vs Information

Data	Information
• Data is **raw.** • It simply exists and has no significance beyond its existence (in and of itself). • It can exist in any form, usable or not. It does not have meaning of itself. • In computer parlance, a spreadsheet generally starts out by holding data. • **Example:** Each student's test score is one piece of data	• It is data that has been **given meaning** by way of relational connection. • This "meaning" can be useful (but does not have to be). • In computer language, a relational database makes information from the data stored within it. • **Example:** The average score of a class or of the entire school is information that can be derived from the given data

Data for Disaster management

- SA: situational awareness
- DM: Decision Making
- SAR: Search and rescue operation
- C&C: Coordination and collaboration



Data / Big data

Information

- **Disaster relief agencies**
  - **Victims**
  - **Media**
  - **Others**

SA

DM

SAR

C&C

Data sources for Disaster management

- Open data
- Linked open data
- Open government data

Data sources for Disaster management

Open data	Linked open data	Open government data
• Data that can be **freely used**, re-used and shared by anyone. • Data to be open must be open **legally and technically**. • Legally means open data license which **allows anyone freely to access, reuse and redistribute.** • Technically open means that data **must be machine-readable**. • Open data examples: digitalglobe, https://data.world/datasets	• Linked data is also known as Web of Data. It is simply about using the Web to create typed links between data from different sources. • Technically, it refers to data published on the Web in such a way that it is **machine-readable**, its **meaning is explicitly defined**, it is **linked to other external data sets**, and can in turn be linked to external data sets. • Linked Open Data is a powerful blend of Linked Data and Open Data: it is both linked and uses open sources. • LOD examples: Dbpedia, GraphDB,	• Governments have taken stock of huge amount of data on environment, climate, construction permits, forest, geographic, flood risk assessment, land use planning, metrological, water level management for disaster risk reduction. • These data are accumulated and maintained. This data should be **made available to the public**. • Open legally and accessible and interoperable technically .

Linked open data

- It is about using web techniques to share and interconnect pieces of data.
- It builds on Semantic Web technologies
  - data is encoded in the form of <subject, predicate, object> RDF-triples.
  - Here, Resource Description Framework (RDF) is a framework for representing information in the Web.
- The World Wide Web has an **abundance of data resources** related to disaster management, either **authoritative data** possessed by a relief organization or crowdsourced data in the social web.
- The advantage of Linked Open Data is easy manipulation and loose integration make it a potential way to interlink the observed data from volunteers to existing systems.

Emergency Data sources

- data.norge.no (Open Public Data in Norway)
  - ...or other public data sources (EU, other...)
- There are lots of open data out there

  - data.gov

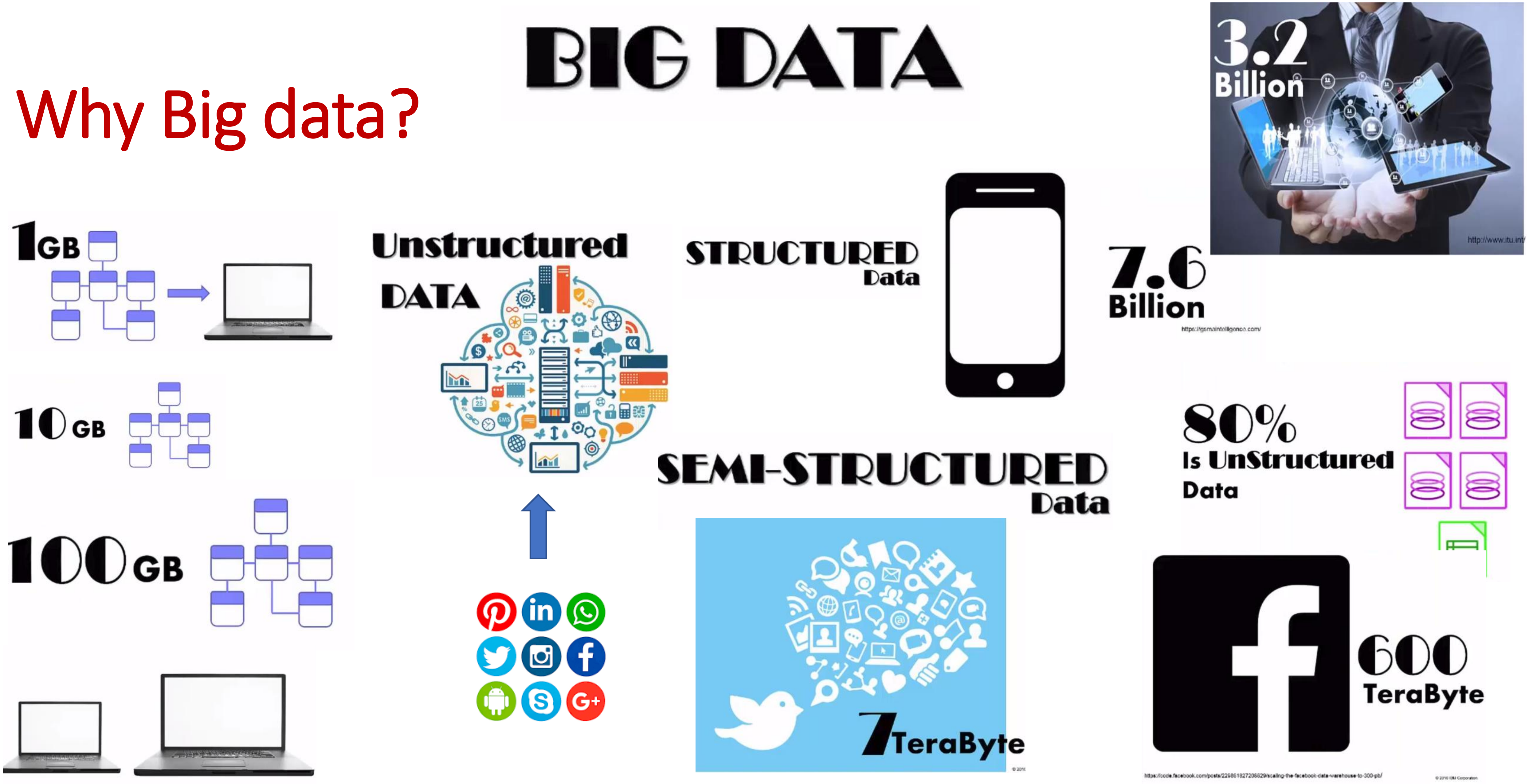
  - GIS datasets

  https://researchguides.dartmouth.edu/gisdata/disasterdata

  - Social media

  – but not so much of it is in semantic formats

Why Big data?


BIG DATA
1GB
10GB
100GB
Unstructured DATA
STRUCTURED Data
SEMI-STRUCTURED Data
7TeraByte
3.2 Billion
http://www.itu.int/
7.6 Billion
https://gsmaintelligence.com/
80% Is UnStructured Data
600 TeraByte
https://code.facebook.com/posts/229861827206629/scaling-the-facebook-data-warehouse-to-300-pb/
© 2010 IBM Corporation

Major challenges with Big Data

Storing the colossal amount of data:

Storing huge data in a traditional system is not possible.

- the storage will be limited to one system
- the data is increasing at a tremendous rate.

Storing heterogeneous data:

Data is in various formats i.e.,

- unstructured,
- semi-structured and
- Structured

Need a system to store different types of data that is generated from various sources

Processing speed:

Time taken to process the huge amount of data is quite high.

- data to be processed is too large.

To address the challenges, Big Data technologies are becoming more and more relevant!!

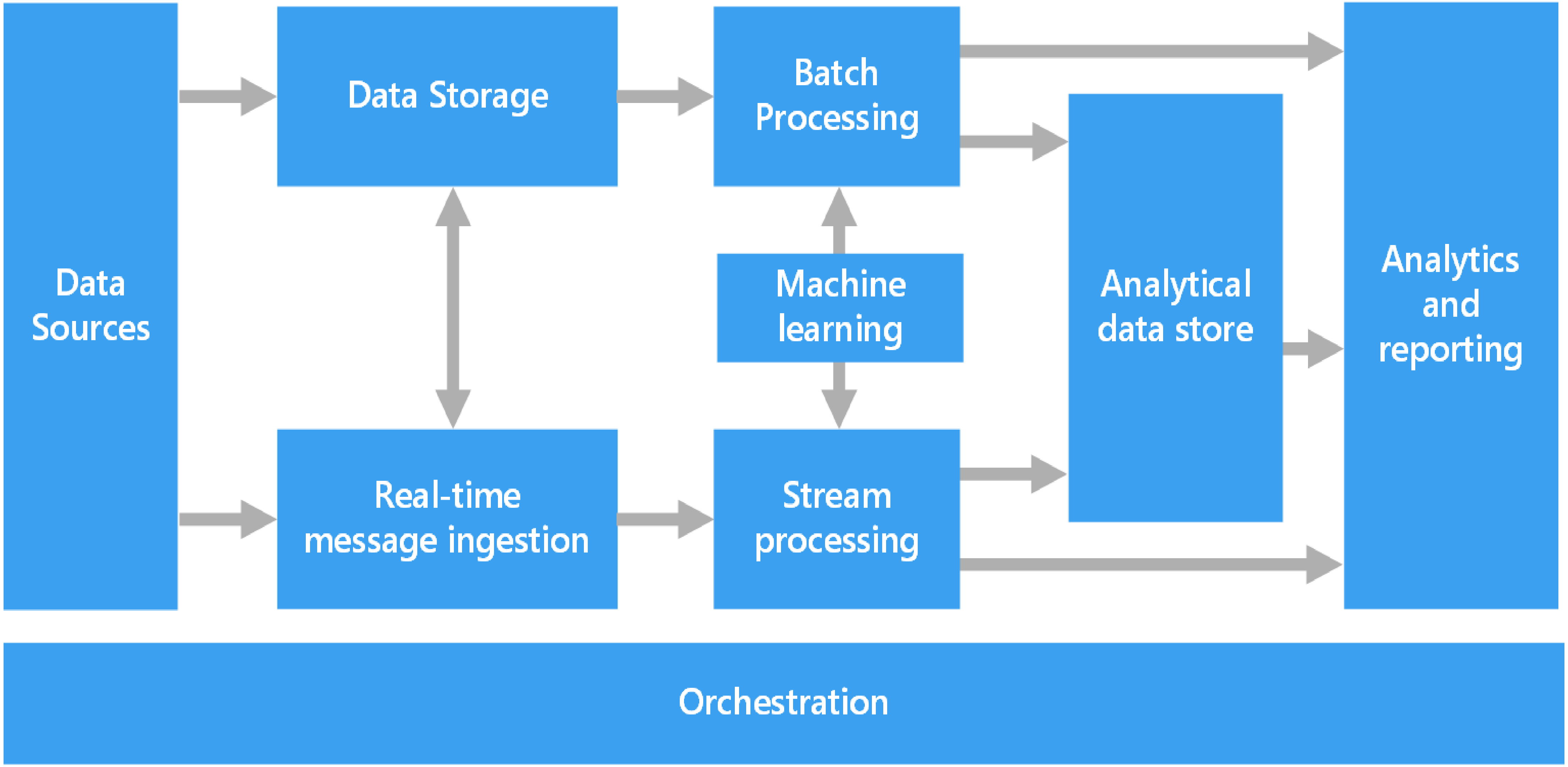
Big data architecture

- Handles the ingestion, processing, and analysis of data that is too large/ complex for traditional database systems.

Big data architectures used for solving the following challenges:	Big data solutions typically involve one or more of the following types of workload:	Big data architectures are needed when:
❖ Expectations with data has changed. ❖ The cost of storage has fallen dramatically ❖ Facing an advanced analytics problem	❖ Batch processing of big data sources at rest. ❖ Real-time processing of big data in motion. ❖ Interactive exploration of big data. ❖ Predictive analytics and machine learning.	❖ Store and process data in volumes too large for a traditional database. ❖ Transform unstructured data for analysis and reporting. ❖ Capture, process, and analyze unbounded streams of data in real time, or with low latency.

Source: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/

Components of a big data architecture



Data Sources

Data Storage

Batch Processing

Machine learning

Real-time message ingestion

Stream processing

Analytical data store

Analytics and reporting

Orchestration

Source: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/

Big Data Technologies

Ingestion **Ingestion Architecture**	**Storage/Retention** **Data Storage:**	**Processing** **Data processing:**	**Access** **Visualization and APIs:**
❖ Scalable, extensible to capture streaming and batch data. ❖ Provide capability to business logic, filters, validation, data quality, routing, etc. business requirements.	❖ Depending on the requirements, data is placed into Hadoop HDFS, Hive, HBase, Elastic search or in-memory. ❖ Metadata management. ❖ Policy-based Data retention is provided.	❖ Processing is provided for both **batch and near-real time** use cases. ❖ Provision workflows for repeatable data processing. ❖ Provide late data arrival handling.	❖ Dashboard and applications that provides valuable business insights. ❖ Data will be made available to consumers using API, MQ feed and DB access.
Technology Stack: **Apache Flume** Apache Kafka Apache Storm **Apache Sqoop** NFS Gateway	**Technology Stack:** **HDFS** Hive Tables **HBase / MapReduce DB** Elastic Search	**Technology Stack:** **Map Reduce** Hive **Spark** Storm Drill	**Technology Stack:** Qlik/Tableau/Spotfire REST APIs Apache Kafka JDBC
Management, Monitoring, Governance Ambari, Cloudera Manager, Cloudera Navigator, MapReduce MCS			

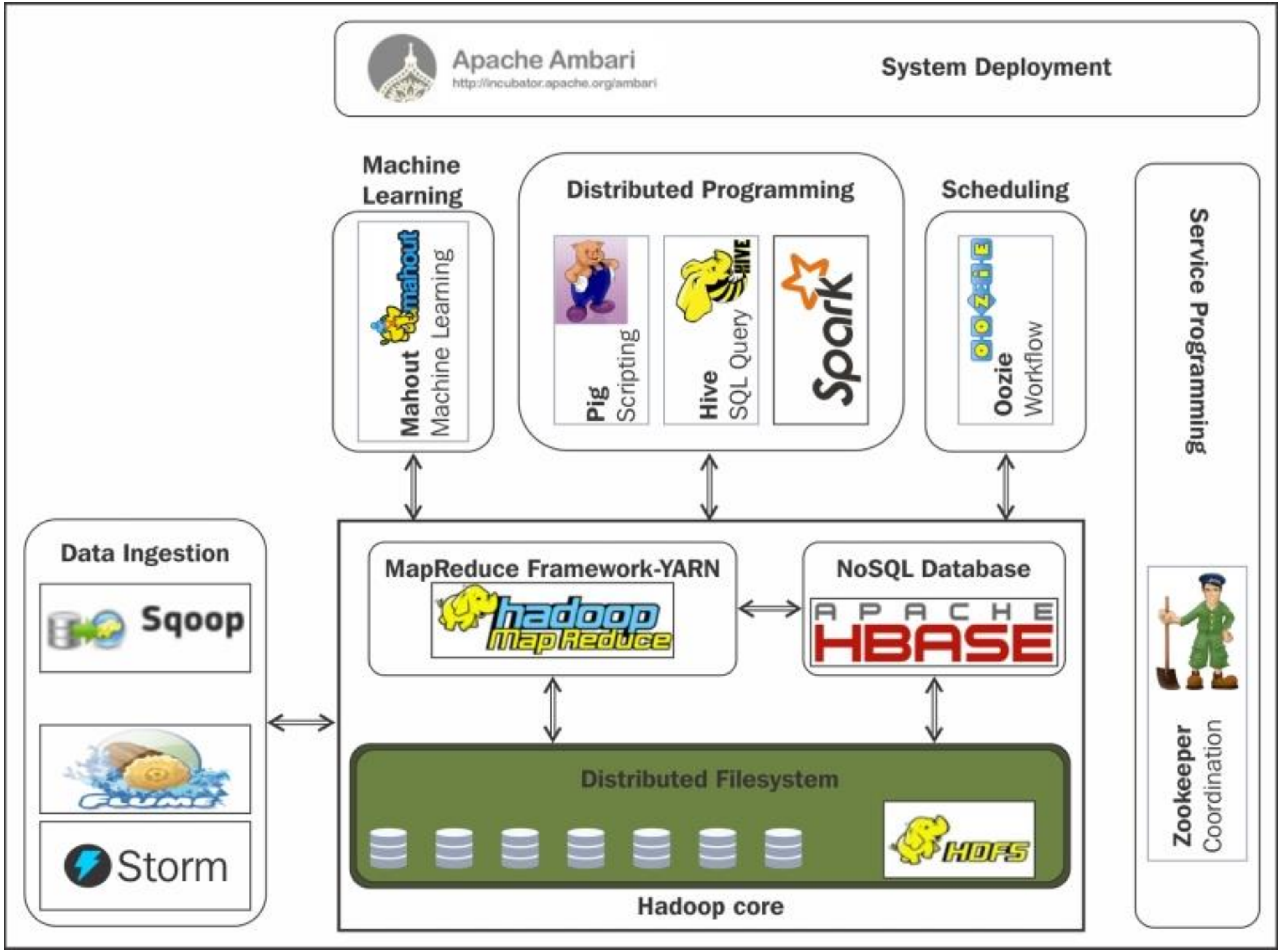
Batch vs Real-time processing

Batch Processing	Real-time processing
• Large amount of data/transactions are processed in a single run over a time period. • This associated jobs generally run without any mutual intervention. • The entire data is pre-selected and fed using command-line parameters and scripts • It is used to execute multiple operations, handle heavy data load, generate reports and manage data flow which is offline. **Example:** Regular reports requiring decision making	• Data processing takes place upon **data entry or command receipt** instantaneously • It must execute within stringent response time constraints. **Example:** Fraud detection

What is Apache Hadoop?

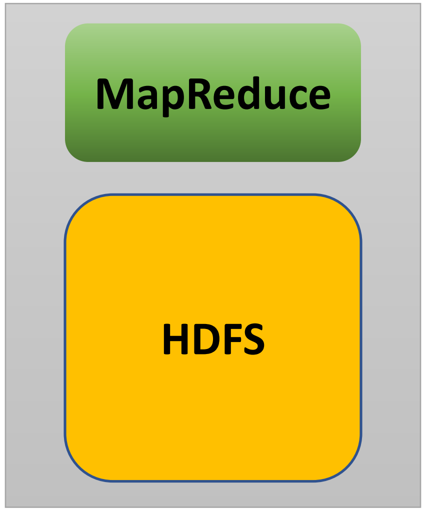
- Hadoop is a running software framework
- Solution for Big Data to deal with complexities of high volume, velocity and variety of data.
- Transforms commodity hardware into a service that:
  - Stores petabytes of data reliably
  - Allows huge distributed computations
- Key attributes:
  - Redundant and reliable (no data loss)
  - Extremely powerful
  - Batch processing centric
  - Easy to program distributed applications
  - Runs on commodity hardware

Hadoop ecosystem

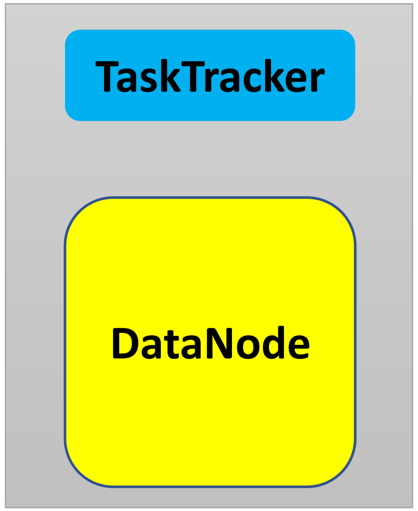


Apache Hadoop

- MapReduce is the processing part of Hadoop
- HDFS is the data part of Hadoop
- The MapReduce server on a typical machine called a **TaskTracker**
- The HDFS server on a typical machine is called a **DataNode**



Machine



Machine

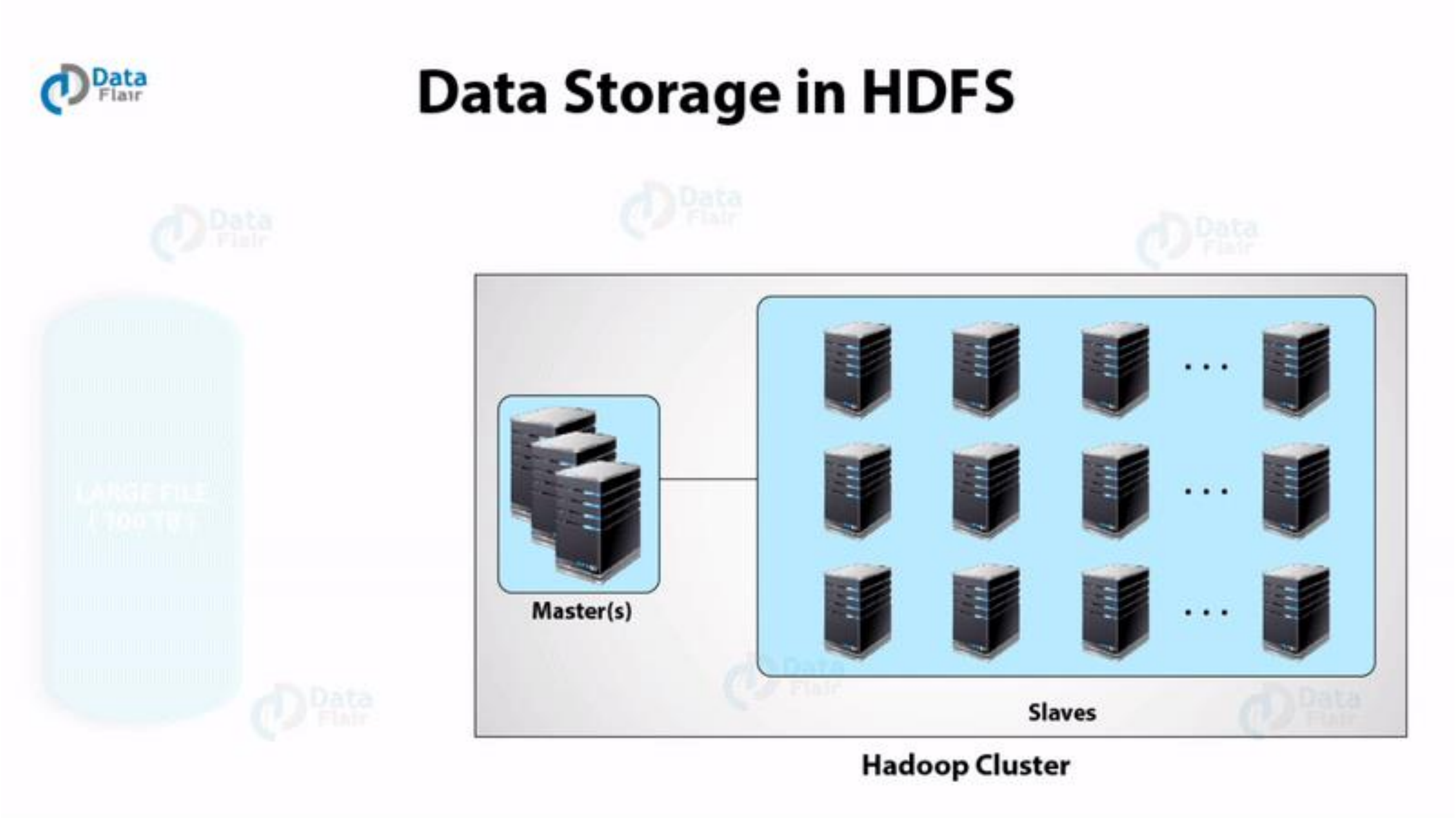
Hadoop Distributed File System (HDFS)

- It provides distributed storage for Hadoop.
- It has a **master-slave topology**.
- Master is a high-end machine.
- Slaves are inexpensive computers.
- It has features like fault tolerance, replication of data, reliability, high throughput access etc.
- The Big Data files get divided into the number of blocks. Hadoop stores these blocks in a distributed fashion on the cluster of slave nodes **(Data nodes**). On the master, we have metadata stored (**namenode**).

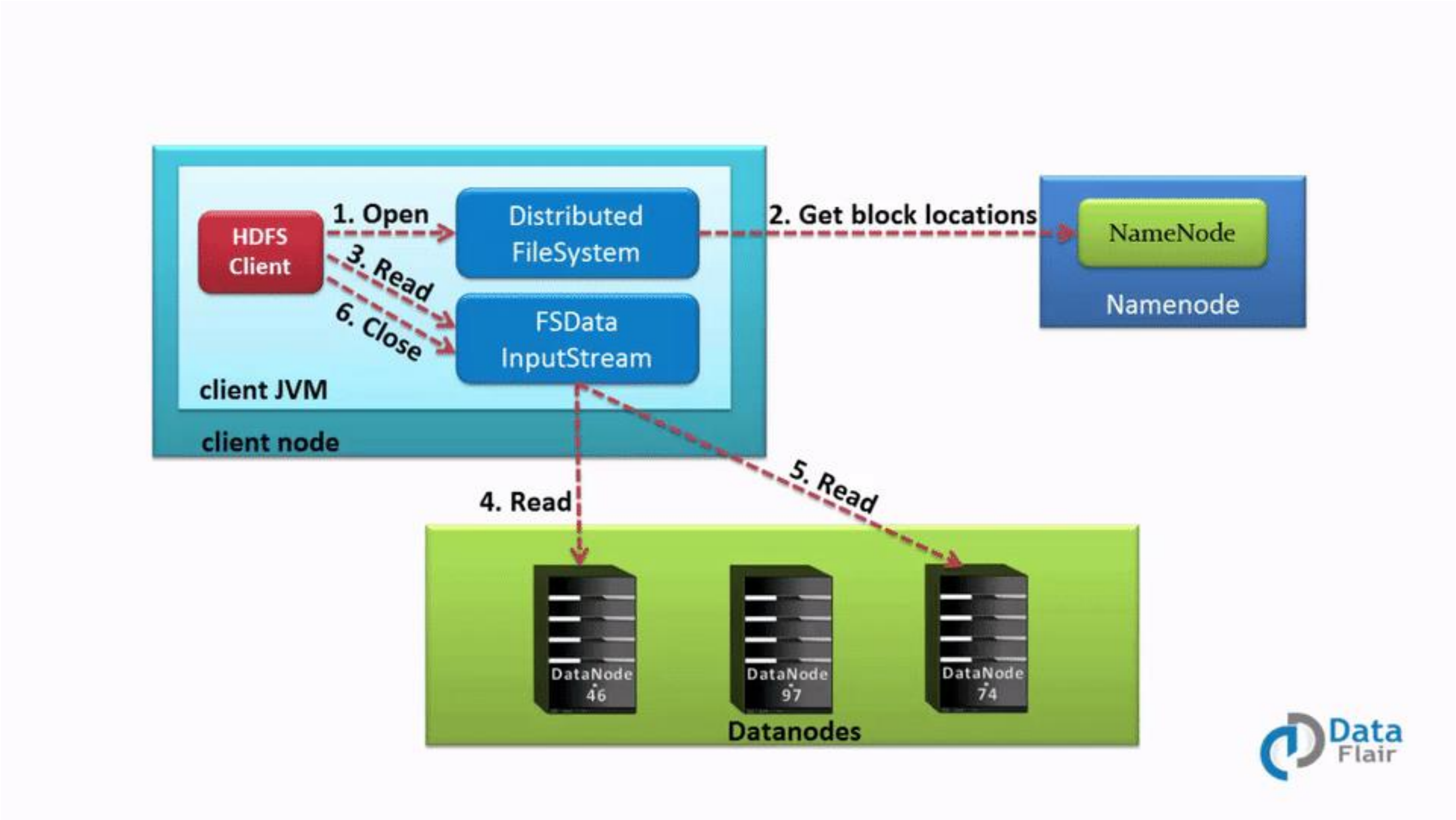
Namenode vs Datanode

Name node	Data node
❖ It **regulates file access** to the clients. ❖ It **maintains and manages the slave nodes** and assign tasks to them. ❖ It executes file system namespace operations like **opening, closing, and renaming files and directories.** ❖ It should be deployed on reliable hardware. ❖ It stores **metadata** like **filename, the number of blocks, number of replicas, a location of blocks, block IDs etc**. ❖ This metadata is **available in memory** in the master for **faster retrieval of data**. In the local disk, a copy of metadata is available for persistence. So name node memory should be high as per the requirement.	❖ It manages the data storage of the system. ❖ There can be 'n' number of slaves (where n can be up to 1000) or data nodes. ❖ It performs **read-write operations** on the file systems, as per client request. ❖ It also performs operations such as **block creation, deletion, and replication** according to the instructions of the namenode. ❖ Once a block is written on a datanode, it replicates it to other datanode and process continues until the number of replicas mentioned is created. ❖ Datanodes can be deployed on commodity Hardware.

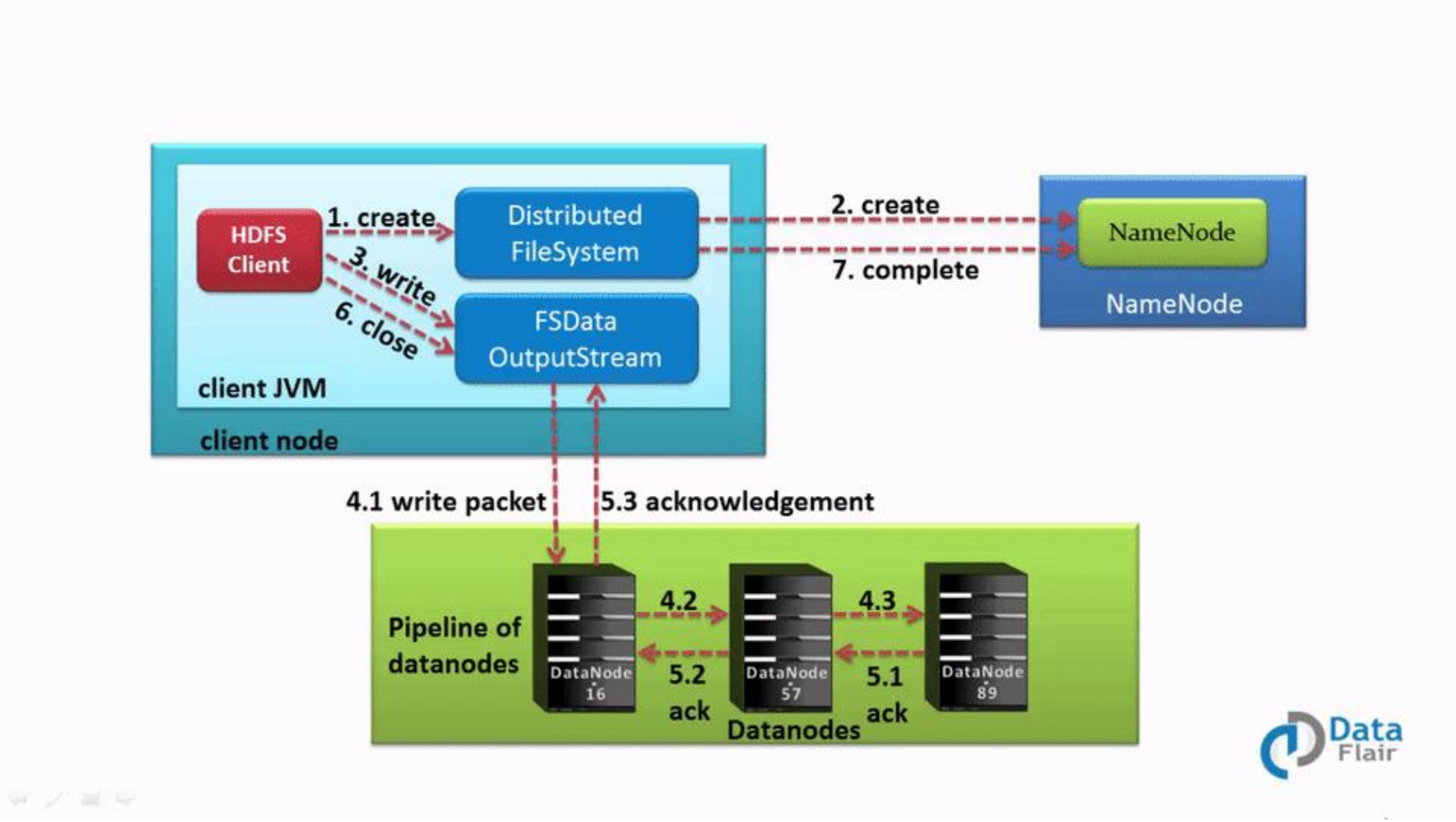
Data storage in HDFS



HDFS Read operation



HDFS Write operation



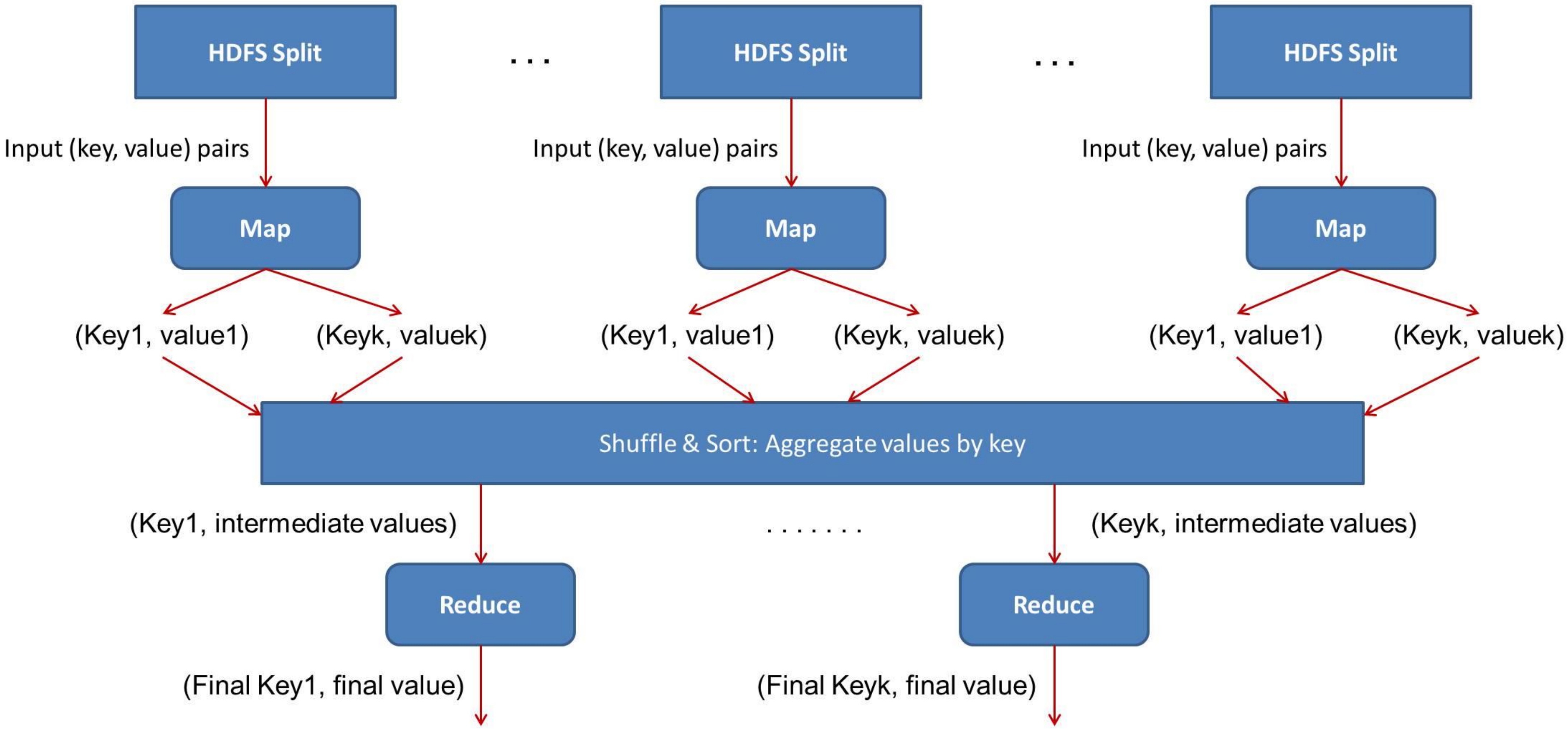
Limitations of HDFS

- Hadoop can perform **only batch processing**, and data will be accessed only in a sequential manner. That means one **has to search the entire dataset** even for the simplest of jobs.
- A huge dataset when processed **results in another huge data set**, which should also be processed sequentially. At this point, a new solution is needed to access any point of data in a single unit of time (random access).
  - Hadoop Random Access Databases: **HBase**, Cassandra, couchDB, Dynamo, and MongoDB

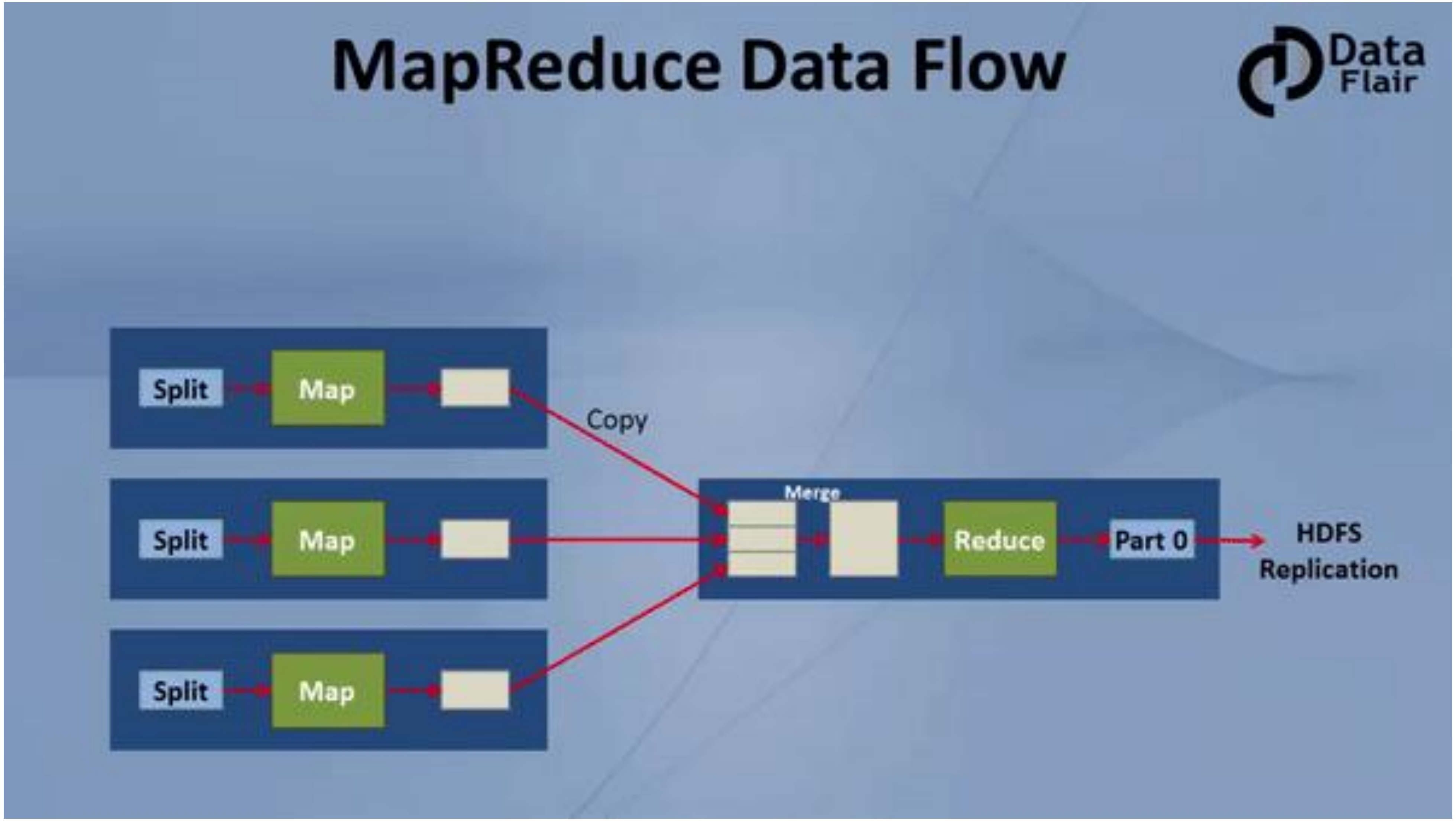
MapReduce

- It is the **processing layer of Hadoop** and designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks.
- It **transforms** lists of input data elements into lists of output data elements.
- A Map-Reduce program will do this twice, using two different list processing idioms:
  - Map
  - Reduce
- In between Map and Reduce, there is small phase called **Shuffle** and **Sort** .

MapReduce data flow



MapReduce Data Flow



Limitations of MapReduce

Unsuitable in real-time processing

Being batch oriented, it takes minutes to execute jobs depending upon the amount of data and number of nodes in the cluster.

Unsuitable for trivial operations

For operations like filters and joins, you might need to rewrite the jobs, which becomes complex because of the key-value pattern.

Unfit for large data on network

It works on the data locality principle, it cannot process a lot of data requiring shuffling over the network well.

Limitations of MapReduce

Unsuitable in OLTP (online Transaction Processing)

OLTP requires a large number of short transactions, as it works on the batch-oriented framework.

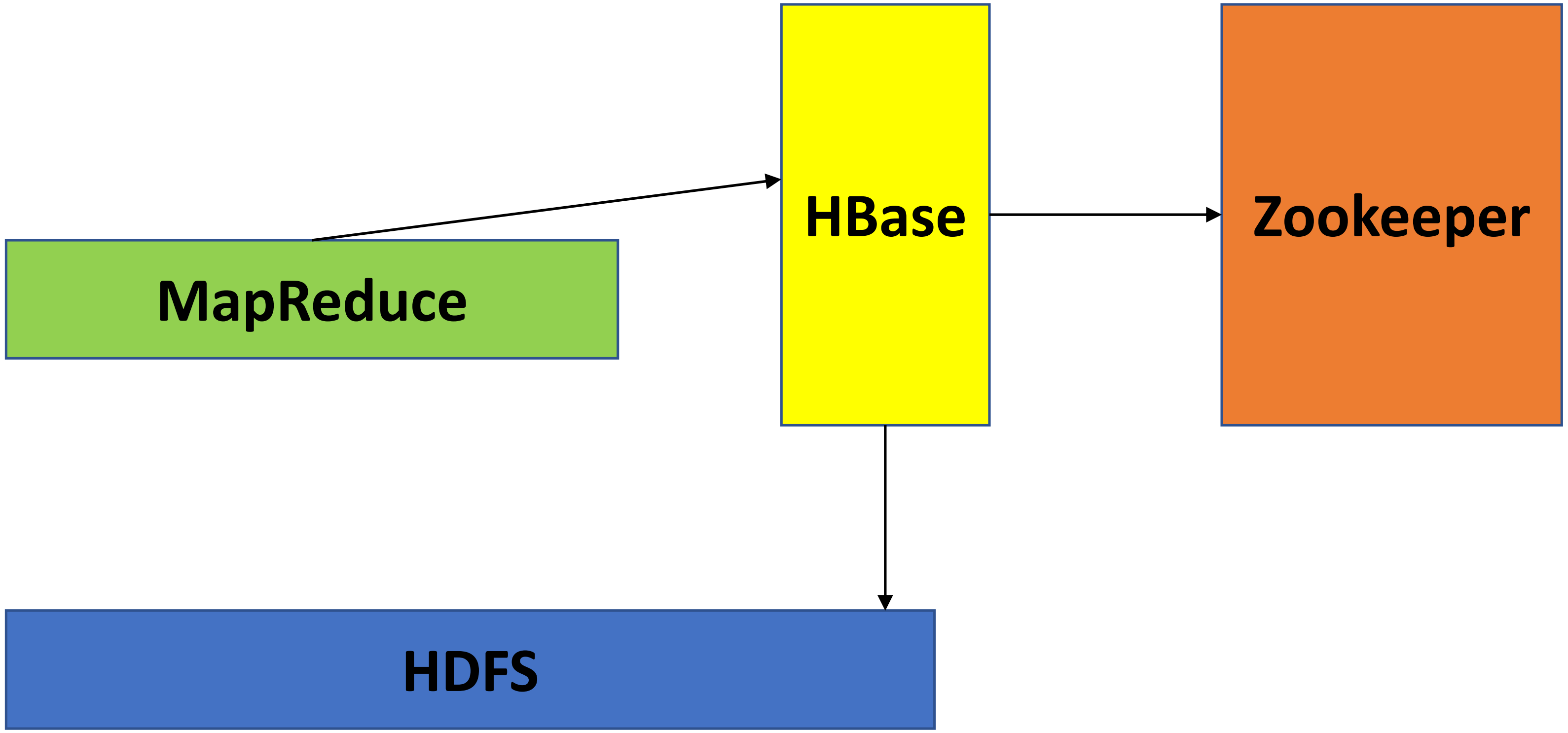
Unfit for processing graphs

The Apache Giraph library processes graphs, which adds additional complexity on top of MapReduce.

Unfit for iterative execution

Being a state-less execution, MapReduce does not fit with use cases like Kmeans that need iterative execution.

Hbase and ZooKeeper



HDFS and HBase

HDFS	HBase
• It is a distributed file system suitable for storing large files. • Does not support fast individual record lookups • It provides high latency batch processing • Only sequential access of data	• It is a nonrelational, distributed database runs on top of the Hadoop. • HBase table can serve as input and output for MapReduce jobs. • Provides fast lookups for larger tables. • Provides low-latency access to single rows from billions of records (random access) • Internally uses Hash tables and provides random access and stores the data in indexed HDFS files for faster lookups.

Hadoop Key Characteristics

- Reliable
  - ❖Data is typically held on multiple DataNodes
  - ❖Tasks that fail are redone
- Scalable
  - ❖Same program runs on 1, 1000 or 4000 machines
  - ❖Scales linearly
- Simple APIs
- Very powerful

Hadoop ecosystem vs Spark

Hadoop Ecosystem	Apache Spark
• Uses MapReduce for batch analytics • Supports third-party plugins/Tools such as Talend • Supports Pig or Hive Queries	• Supports both real-time and batch processing

Hadoop ecosystem vs Spark

Batch Processing:

Spark batch can be used over Hadoop MapReduce

Real-time Streaming Data Analysis:

Spark streaming can be used over specialized library like Storm.

Interactive SQL Analysis:

Spark batch can be used over Impala

Structured Data Analysis:

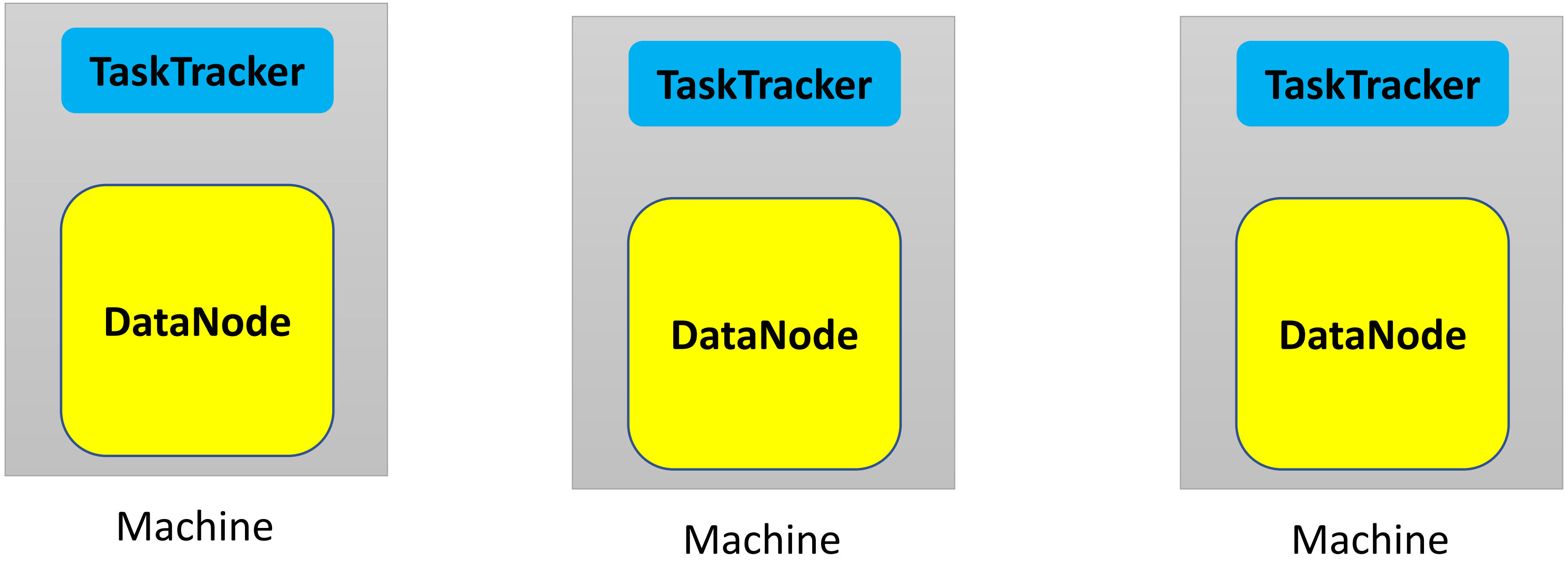
Spark SQL can be using SQL

Machine Learning Analysis:

MLLib can be used for clustering, recommendation, and classification

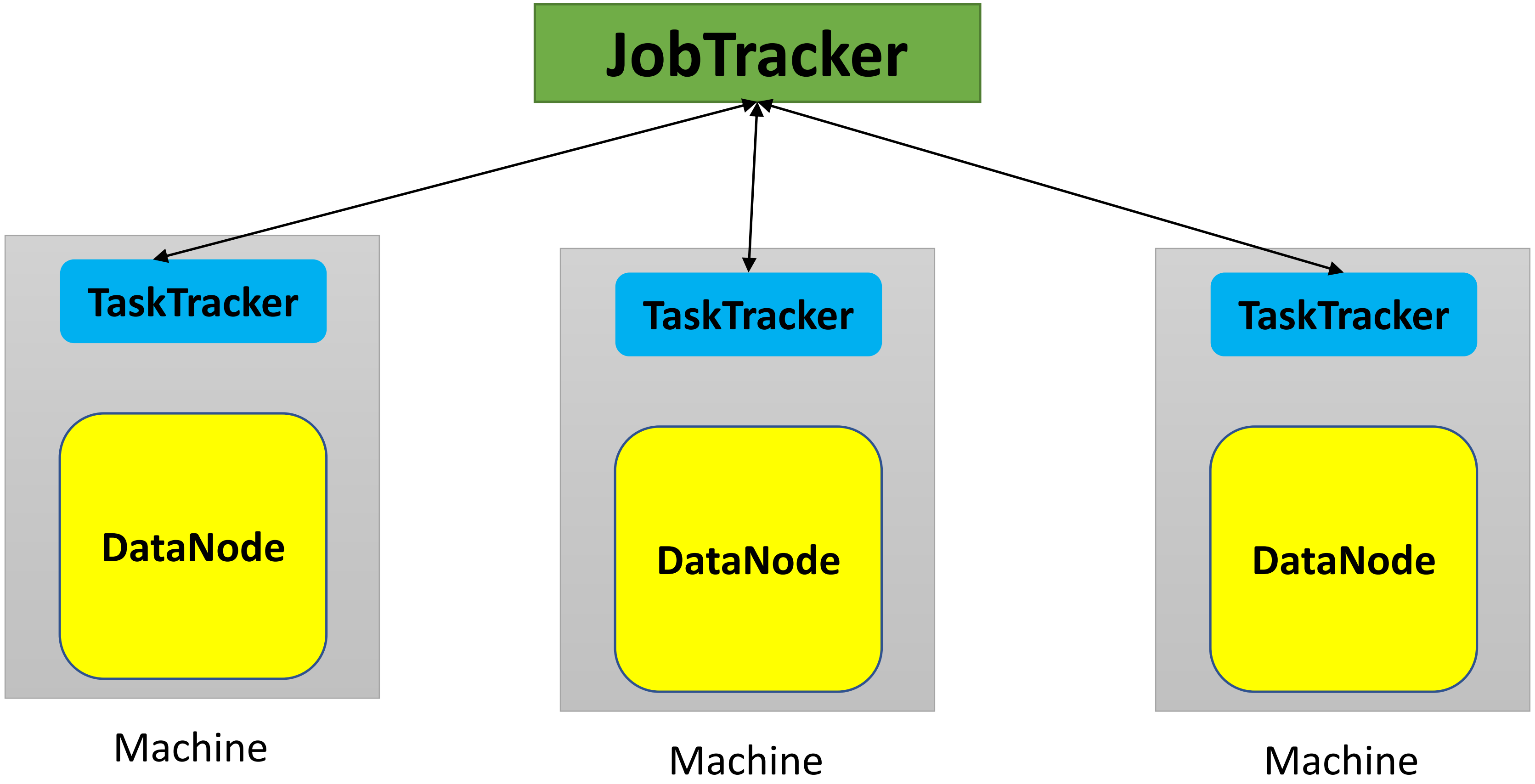
Hadoop Cluster

- Having multiple machines with Hadoop creates a cluster.



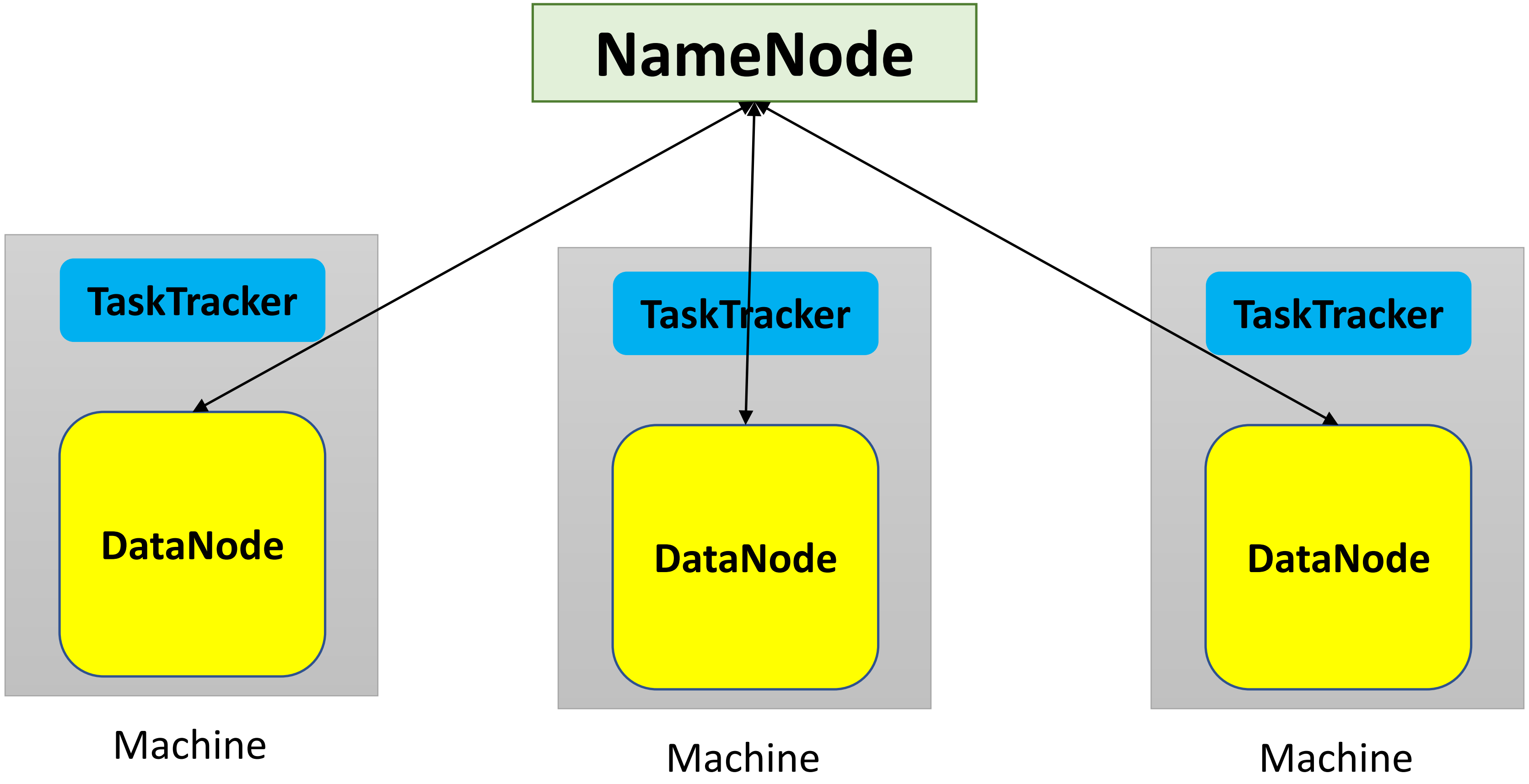
Hadoop Cluster

- JobTracker keeps track of jobs being run



Hadoop Cluster

- NameNode keeps information on data location.



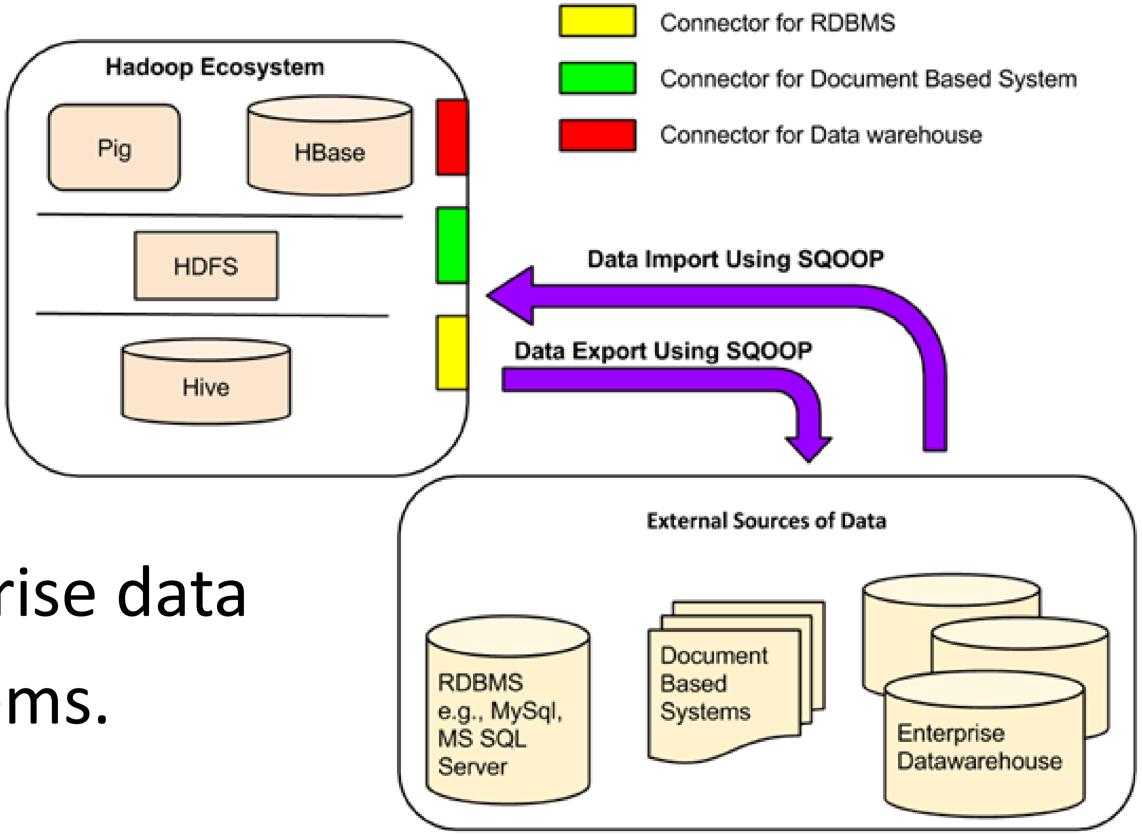
Data Loading in Hadoop

- Work with enormous data in several different forms and load such heterogeneous data into Hadoop, different tools have been developed.

  - Sqoop and **Flume** .

SQOOP

- Designed to support bulk import of data into HDFS
  - from structured data stores
  - relational databases, enterprise data warehouses, and NoSQL systems.
- Sqoop Connectors
  - Data transfer between Sqoop and external storage system



FLUME

- Used for moving massive quantities of streaming data into HDFS

  - E.g., . Collecting log data present in log files from web servers and aggregating it in HDFS for analysis

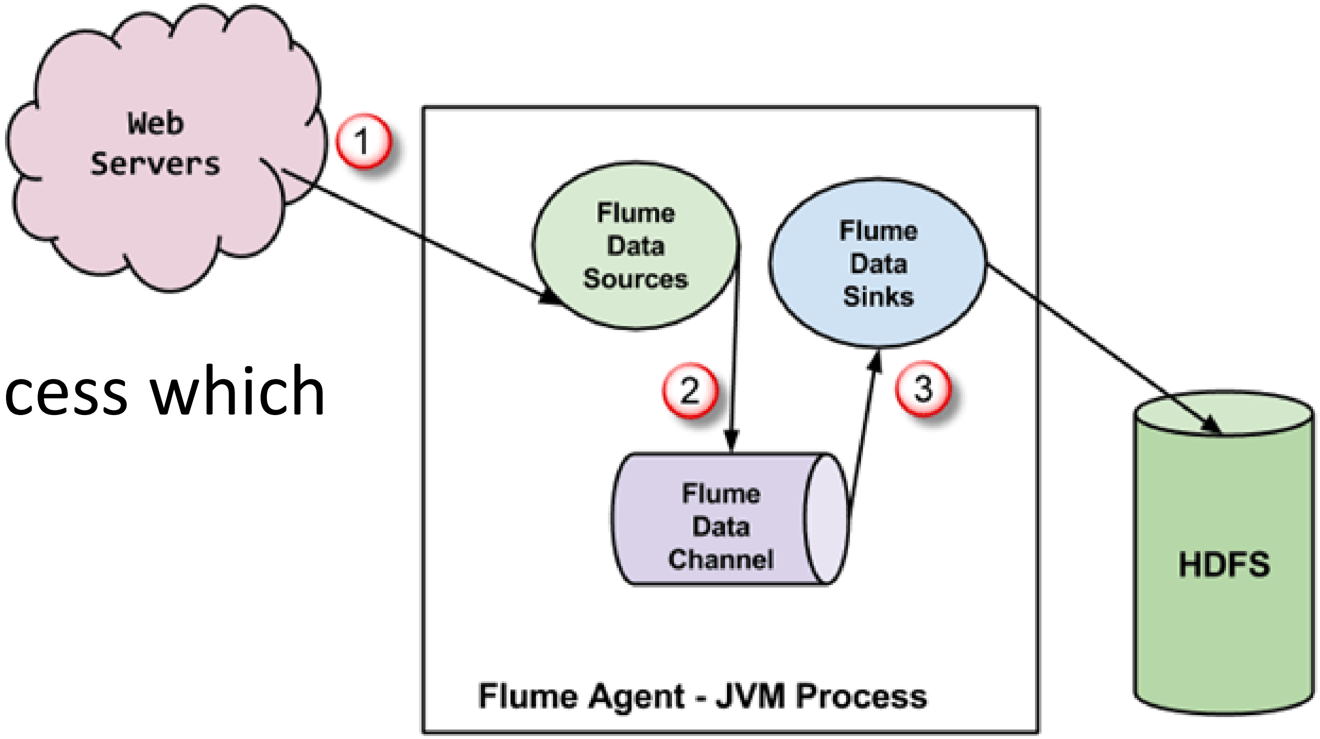
- Flume supports multiple sources like –

  - 'tail' (which pipes data from local file and write into HDFS via Flume, similar to Unix command 'tail')

  - System logs

  - Apache log4j (enable Java applications to write events to files in HDFS via Flume).

Data Flow in Flume



- A **Flume agent** is a **JVM** process which has 3 components
  - ❖ **Flume Source**,
  - ❖ **Flume Channel** and
  - ❖ **Flume Sink**-

  through which events propagate after initiated at an external source .

Comparison

Sqoop	Flume	HDFS
• Used for importing data from structured data sources such as RDBMS. • It has a connector-based architecture. • Connectors know how to connect to the respective data source and fetch the data. • HDFS is the destination for data import using Sqoop. • Sqoop data load is not event driven. • Example: To import data from structured data sources, Sqoop should be used as its connectors know how to interact with structured data sources and fetch data from them.	• It is used for moving bulk streaming data into HDFS. • It has an agent-based architecture i.e., code is written (called as agent) to take care of fetching the data. • Data flows to HDFS through zero or more channels. • Flume data load can be driven by event. • Example: to load streaming data such as tweets generated by Twitter or log files of a web server, Flume should be used as Flume agents are built for fetching the streaming data.	• It is a distributed file system used by Hadoop ecosystem to store the data. • It has a distributed architecture where data is distributed across multiple data nodes. • It just stores data provided to it by whatsoever means. • It has its own built-in shell commands to store data into it. HDFS can not import streaming data.

Run the Tutorial:

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Check out the JavaDoc:

http://hadoop.apache.org/docs/r2.7.4/api/index.html

Summary

- Individual essay and programming projects topics
- EM data sources
    - open data
    - linked open data
    - open government data
- Big data architecture and technologies
- Next: Practical session
    - Running Hadoop and MapReduce
    - Querying and analyzing government open data source by using Spark cluster
- Presentations by you