Distributed and Parallel Databases

Distributed and Parallel Databases

- **What is a Distributed Database?**
- A Distributed Database is a collection of logically interrelated databases distributed across a computer network. A Distributed Database Management System (DDBMS) manages these databases to provide transparency and consistency.

Distributed and Parallel Databases

- **Key Features:**
- **Distribution**: Data is stored across multiple physical locations.
- **Transparency**:
  - **Location Transparency**: Users don’t need to know where data is stored.
  - **Fragmentation Transparency**: Data may be split (fragmented) across sites.
  - **Replication Transparency**: Data may be duplicated.
- **Scalability**: Systems can grow by adding more sites.
- **Autonomy**: Each site may manage its own data.

Distributed and Parallel Databases

- **Types of Distributed Databases:**

1. **Homogeneous**: Same DBMS at each site.
2. **Heterogeneous**: Different DBMSs, schemas, or data models.

Distributed and Parallel Databases

- What is a Parallel Database?
- A Parallel Database System uses multiple processors and storage devices to execute queries simultaneously, providing high performance and scalability.

- Goals:
- Increase query throughput and response time
- Handle large volumes of data efficiently

Distributed and Parallel Databases

Types of Parallelism:

Type	Description
Intra-query	One query is divided and processed in parallel.
Intra-operation	A single operation (e.g., join) is parallelized.
Inter-query	Multiple queries processed simultaneously.
Pipeline	Different operations of the same query are parallelized.

Distributed and Parallel Databases

Distributed Database Platforms

Platform	Description
Google Spanner	A globally distributed SQL database with strong consistency and ACID support.
Amazon Aurora	A MySQL/PostgreSQL-compatible cloud database with multi-region replication.
Cassandra	A distributed NoSQL database designed for high availability and scalability.
CockroachDB	A distributed SQL database inspired by Google Spanner (strong consistency).
MongoDB (Sharded)	MongoDB supports horizontal partitioning (sharding) for distributed deployments.

Distributed and Parallel Databases

Parallel Database Platforms

Platform	Description
Greenplum	An MPP (Massively Parallel Processing) relational database built on PostgreSQL.
Amazon Redshift	A fully managed cloud data warehouse using parallel query execution.
IBM Netezza	High-performance parallel DB appliance for big data analytics.
Teradata	Enterprise-grade parallel RDBMS designed for OLAP workloads.
Vertica	Columnar storage and massively parallel query engine for big data.

Distributed and Parallel Databases

Hybrid Platforms (Distributed + Parallel)

Platform	Description
Snowflake	Cloud-native data warehouse with automatic scaling, distributed storage + compute.
Apache Hive (on Hadoop)	SQL engine on top of Hadoop — distributes data and runs parallel MapReduce or Spark jobs.
Google BigQuery	Serverless, distributed, and parallel execution engine for real-time analytics.

Distributed and Parallel Databases
Introduction to Apache Hadoop

- **Apache Hadoop**
- Apache Hadoop is an open-source framework designed for:

1. Storing massive volumes of data (structured or unstructured)
2. Processing that data in parallel across a distributed cluster of computers

Distributed and Parallel Databases
Introduction to Apache Hadoop

- **Advantages of Hadoop**

Scalable: Handles petabytes of data
Cost-effective: Runs on commodity hardware
Fault-tolerant: Automatic recovery from node failure
Flexible: Works with structured, semi-structured, unstructured data
Open-source and widely supported

Distributed and Parallel Databases
Introduction to Apache Hadoop

- **Hadoop Use Cases**

1. **Log processing** (e.g. web server logs)
2. **Big data analytics** (e.g. social media analysis)
3. **ETL pipelines**
4. **Machine learning preprocessing**
5. **Bioinformatics** (e.g. genome processing)

Distributed and Parallel Databases
Introduction to Apache Hadoop

Hadoop Ecosystem Overview

Component	Role
HDFS	Hadoop Distributed File System (stores data)
YARN	Resource manager for job scheduling/execution
MapReduce	Programming model for batch processing
Hive	SQL-like queries on Hadoop
Pig	High-level scripting for data flows
HBase	NoSQL distributed database on top of HDFS
Spark	In-memory distributed computing engine
Oozie	Workflow scheduler
Flume / Sqoop	Data ingestion tools

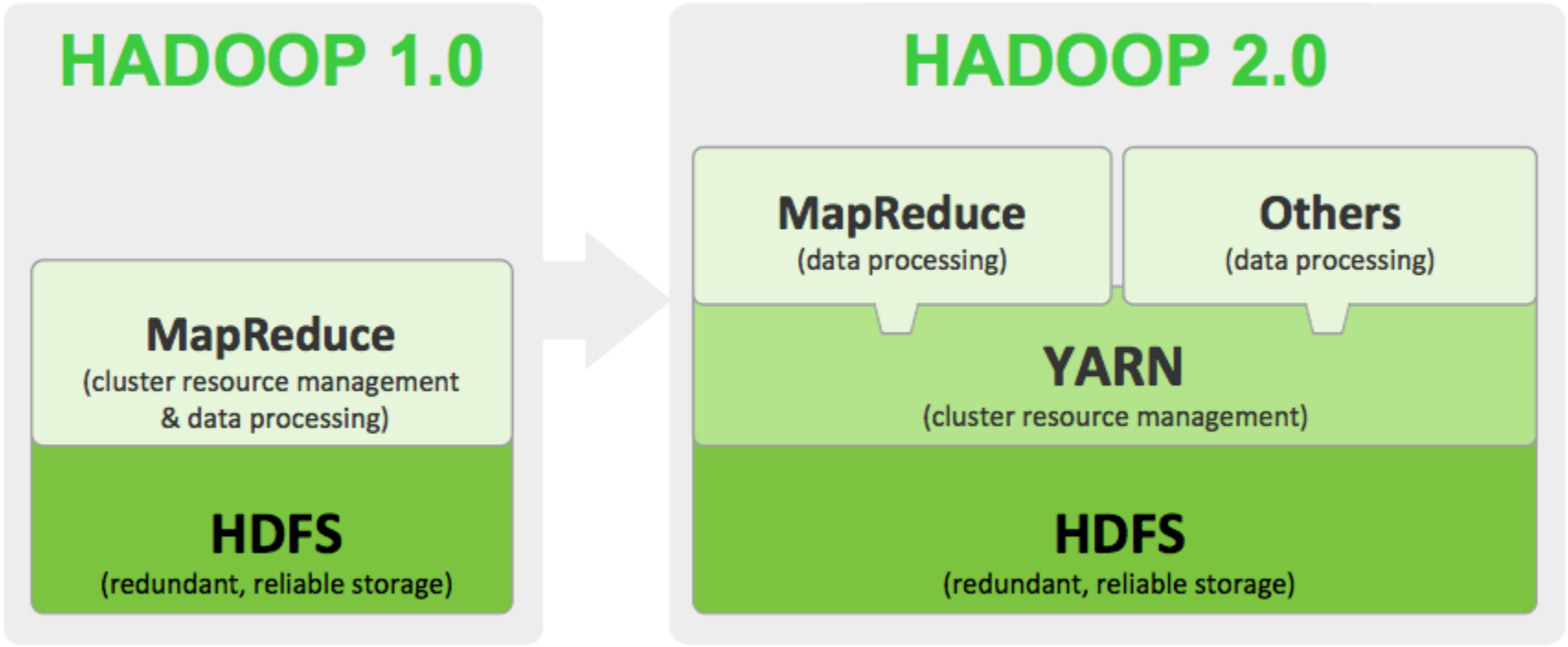
Distributed and Parallel Databases
Introduction to Apache Hadoop

- **Hadoop Architecture**

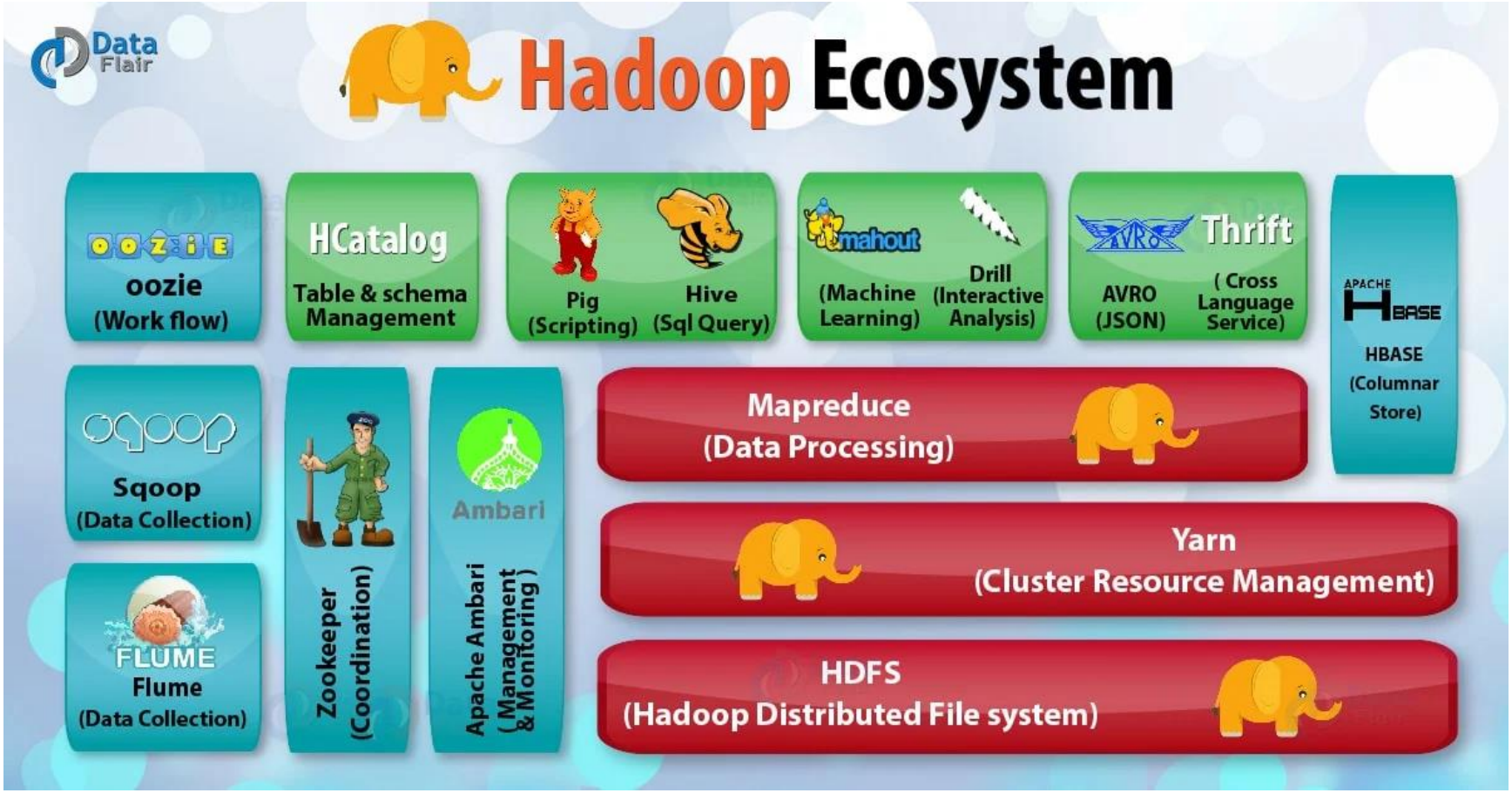
➢ **HDFS (Hadoop Distributed File System)**
  - **NameNode**: Master node that manages metadata (file names, block locations)
  - **DataNodes**: Store actual data blocks
- Data is **split into blocks** (default 128MB) and **replicated** (typically 3 copies)

➢ **MapReduce Processing Model**
  - **Map Phase**: Data is split into chunks and processed in parallel to produce key-value pairs.
  - **Shuffle & Sort Phase**: Intermediate keys are sorted and grouped.
  - **Reduce Phase**: Aggregates or processes each group of keys.

Distributed and Parallel Databases
Introduction to Apache Hadoop

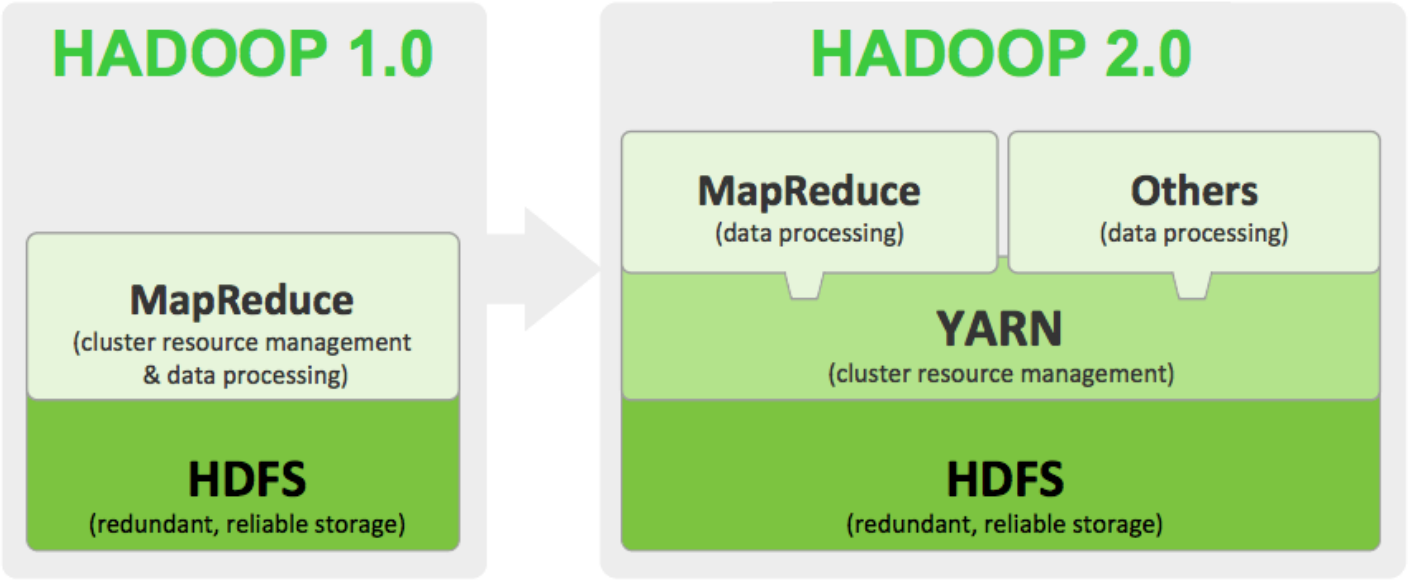


Distributed and Parallel Databases
Introduction to Apache Hadoop



Distributed and Parallel Databases
Introduction to Apache Hadoop

- MapReduce est un modèle de programmation disponible dans les environnements Hadoop
- Utilisé pour accéder aux big data stockées dans le Hadoop File System (HDFS)

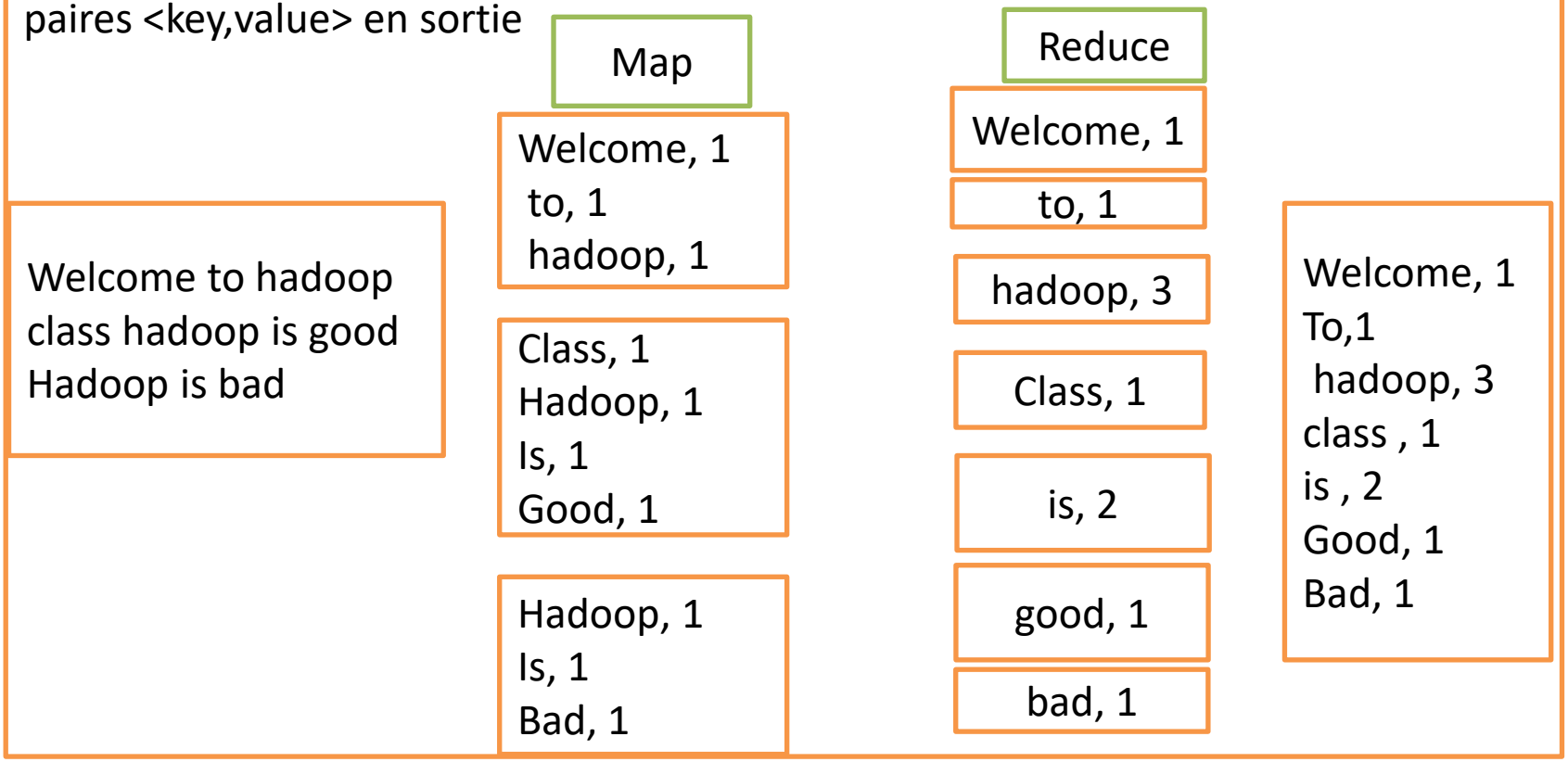


Distributed and Parallel Databases
Introduction to Apache Hadoop

MapReduce – Principes de base (ex: WORD COUNT)

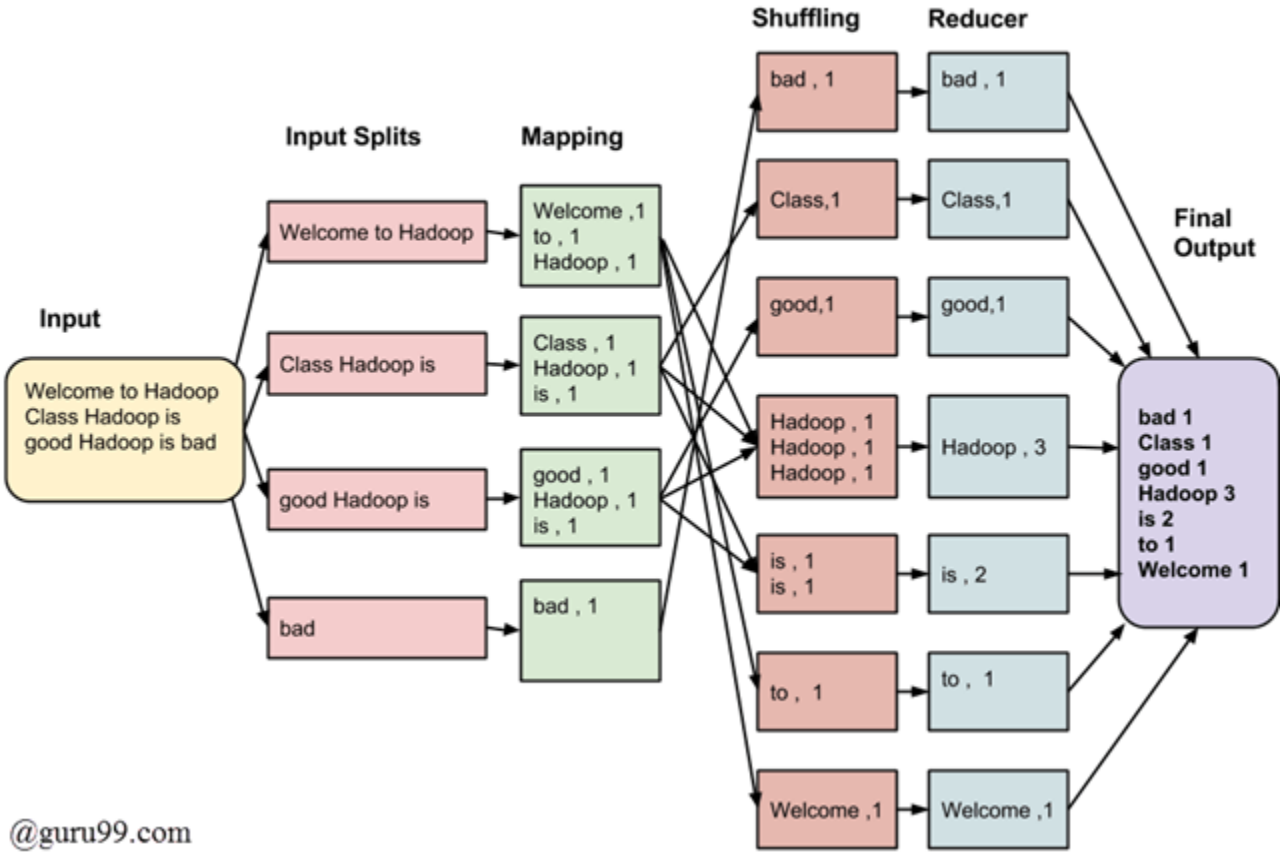
Map transforme les entrées du disque en paires <key,value>

Reduce transforme également les entrées en paires <key,value> et génère une des paires <key,value> en sortie



Distributed and Parallel Databases
Introduction to Apache Hadoop

MapReduce :**WORD COUNT**



Distributed and Parallel Databases
Introduction to Apache Hadoop

- **MapReduce :WORD COUNT**
- **programme: MAP**

public class WordCountMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

private final IntWritable one = new IntWritable(1); private Text word = new Text();

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {

String line = value.toString();

StringTokenizer itr = new StringTokenizer(line.toLowerCase());

while(itr.hasMoreTokens())

{ word.set(itr.nextToken()); output.collect(word, **one**); } } }

Distributed and Parallel Databases
Introduction to Apache Hadoop

REDUCER Code

```
public class WordCountReducer extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {


   public void reduce(Text key, Iterator<IntWritable> values,  OutputCollector
<Text,  IntWritable> output, Reporter reporter) throws IOException {


 int sum = 0;
      while (values.hasNext()) {
         // replace ValueType with the real type of your value
         IntWritable value = (IntWritable) values.next();
         sum += value.get(); // process value
      }
      output.collect(key, new IntWritable(sum));
   }}
```

Distributed and Parallel Databases
Introduction to Apache Hadoop

```
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
```

Distributed and Parallel Databases
Introduction to Apache Hadoop

Calculer le total des ventes par magasin pour l'année en cours ?

2012-01-01	09:00	San Jose	Men's Clothing	214.05	Amex
2012-01-01	09:00	Fort Worth	Women's Clothing	153.57	Visa
2012-01-01	09:00	San Diego	Music	66.08	Cash
2012-01-01	09:00	Pittsburgh	Pet Supplies	493.51	Discover
2012-01-01	09:00	Omaha	Children's Clothing	235.63	MasterCard
2012-01-01	09:00	Stockton	Men's Clothing	247.18	MasterCard
2012-01-01	09:00	Austin	Cameras	379.6	Visa
2012-01-01	09:00	New York	Electronics	296.8	Cash
2012-01-02	15:20	Lincoln	Cameras	242.2	Discover
2012-01-02	15:20	Madison	Baby	254.15	MasterCard
2012-01-02	15:20	Wichita	Cameras	446.66	Amex
2012-01-02	15:20	Irvine	Computers	9.23	Discover
2012-01-02	15:20	Anaheim	Cameras	3.64	Visa
2012-01-02	15:21	Birmingham	Music	156.68	Cash

Distributed and Parallel Databases
Introduction to Apache Hadoop

- Hadoop + Hive Example (SQL on Hadoop)

```
CREATE TABLE words (line STRING);
LOAD DATA INPATH '/user/data/words.txt' INTO TABLE words;

-- Count word occurrences
SELECT word, COUNT(*)
FROM (
  SELECT explode(split(line, ' ')) AS word
  FROM words
) t
GROUP BY word;
```

Distributed and Parallel Databases
Introduction to Apache Hadoop

- Hadoop Distributed File System (HDFS),
- Amazon S3,
- Google Cloud Storage

Distributed and Parallel Databases
Introduction to Apache Hadoop

Hadoop Distributed File System (HDFS)

- est un système de fichiers distribué qui gère de grands ensembles de données
- HDFS est l'un des principaux composants de Apache Hadoop, (avec MapReduce et YARN).
- s'exécutant sur du matériel de base

Les objectifs de HDFS

1. Récupération rapide des pannes de matériel
   - HDFS peut comprendre des milliers de serveurs➔ la panne d'au moins un serveur est inévitable
   - HDFS a été conçu pour détecter les défauts et se restaurer automatiquement rapidement

Distributed and Parallel Databases
Introduction to Apache Hadoop

Objectif de HDFS

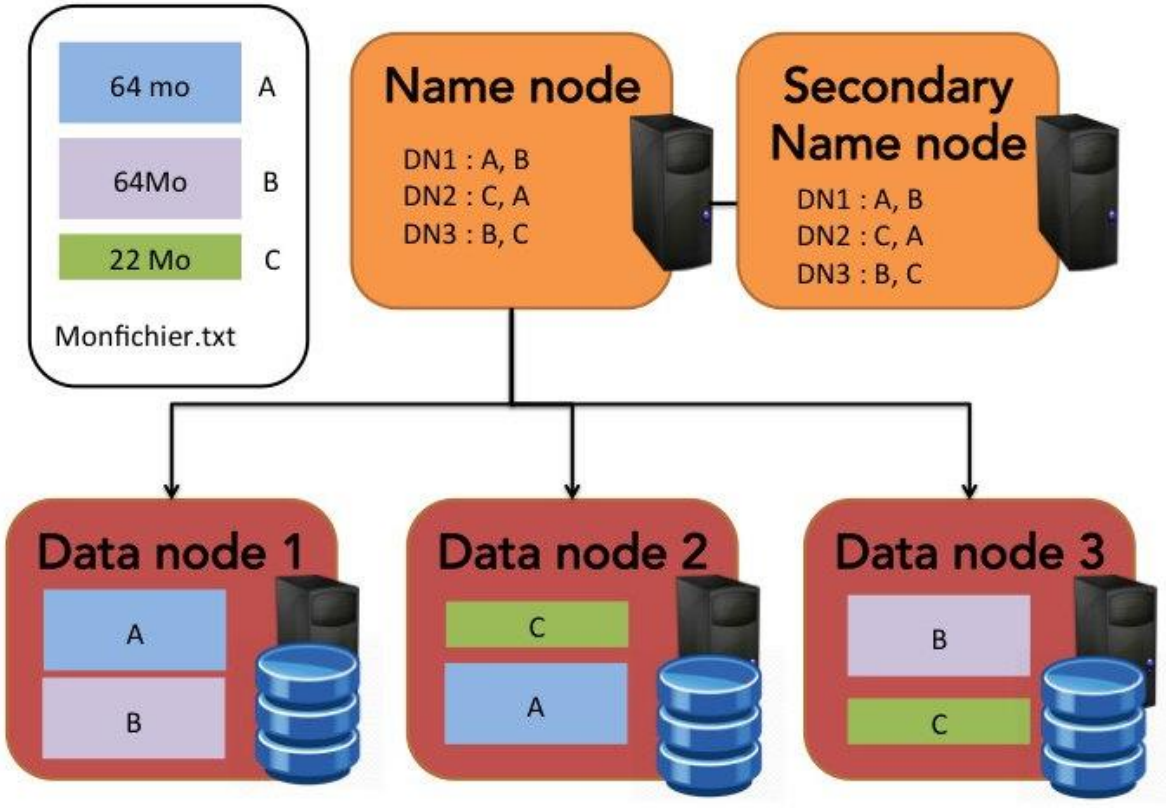
1. Accès aux flux de données en continu
2. Hébergement de grands ensembles de données.
3. Portabilité

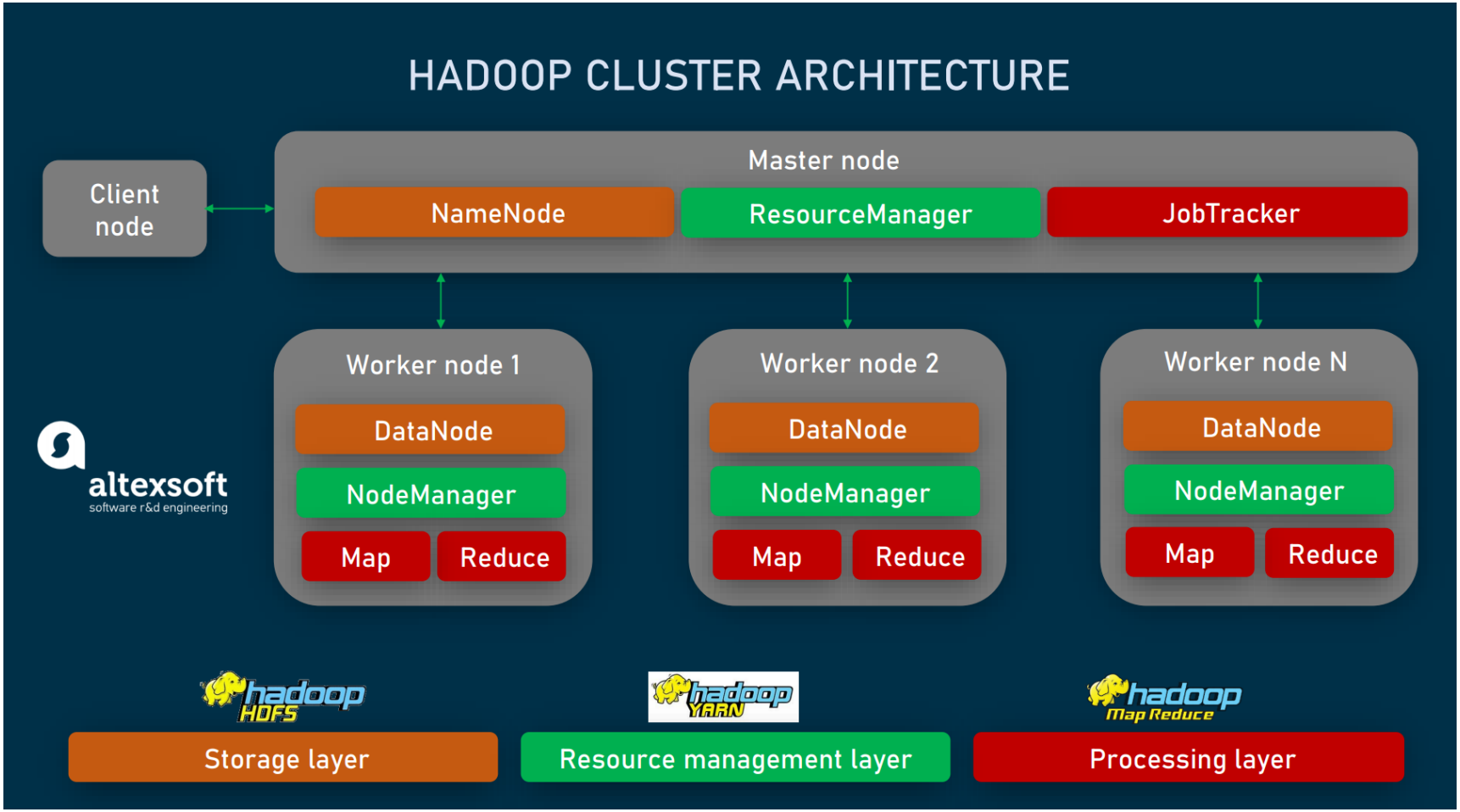
Architecture de HDFS

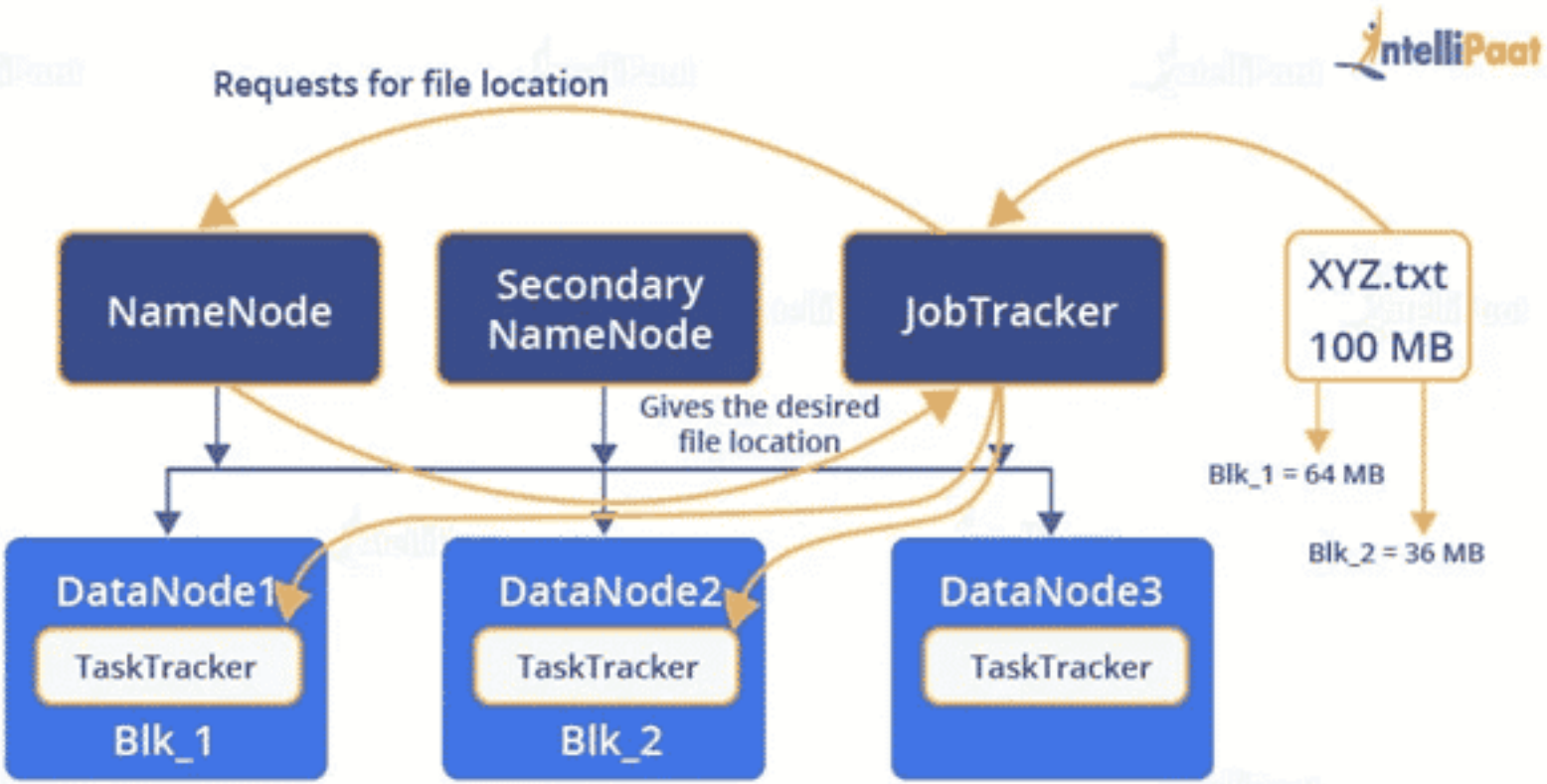
- Un cluster de plusieurs machines
- Principe de Maitre/ esclave
- les données sont stockées sur les **datanodes** (esclaves)
- les Métadonnées sur les blocs de données sont géré par le **namenode** (maître)

Architecture de HDFS

- Chaque fichier de données est décomposé en blocs. Par défaut 64 Mo de taille
- Avec principe de **réplication**




HADOOP CLUSTER ARCHITECTURE
Client node
Master node
NameNode
ResourceManager
JobTracker
Worker node 1
DataNode
NodeManager
Map
Reduce
Worker node 2
DataNode
NodeManager
Map
Reduce
Worker node N
DataNode
NodeManager
Map
Reduce
altexsoft
software r&d engineering
hadoop HDFS
Storage layer
hadoop YARN
Resource management layer
hadoop Map Reduce
Processing layer


IntelliPaat
Requests for file location
NameNode
Secondary NameNode
JobTracker
XYZ.txt
100 MB
Gives the desired file location
Blk_1 = 64 MB
Blk_2 = 36 MB
DataNode1
TaskTracker
Blk_1
DataNode2
TaskTracker
Blk_2
DataNode3
TaskTracker