

Big Data

2023-2024

ECOLE SUPÉRIEUR POLYTECHNIQUE D'ORAN
MAURICE AUDIN

Certifications Big Data

- ▶ Cloudera Data Platform Certification Program.
 - ▶ Levels CCS, CCP, CCA.
 - ▶ Data Engineer, Data Scientist, Admin, Spark Hadoop Dev.

- ▶ Databricks Certification and Badging.
 - ▶ Levels pro and associate.
 - ▶ Data analyst, Data Engineer, and Hadoop Migration.



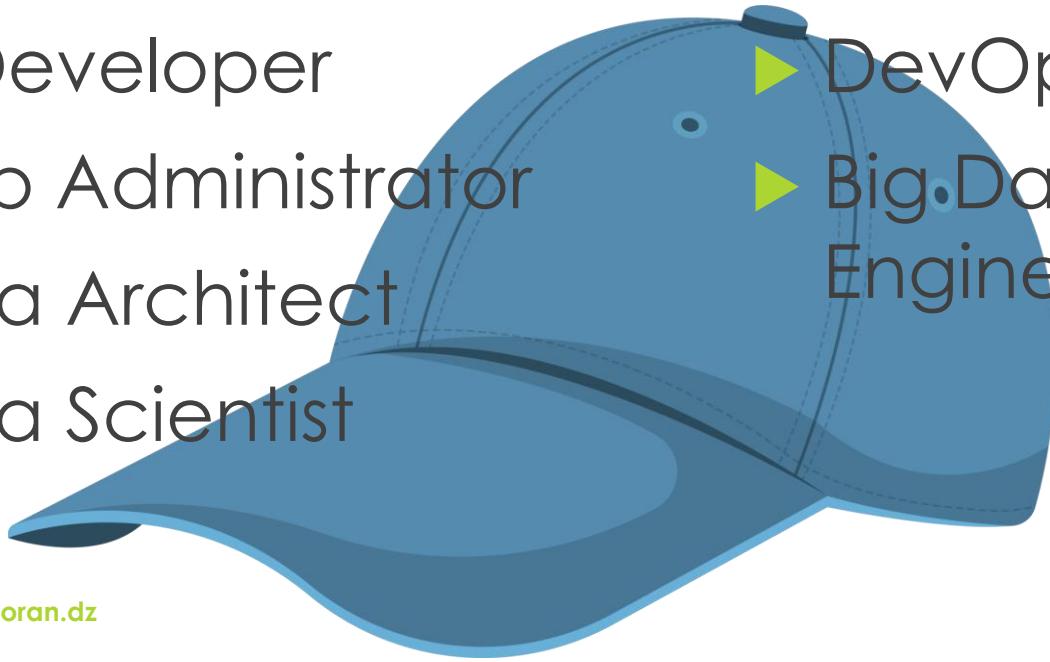
databricks

Certifications Big Data

- ▶ Hortonworks Data Platform (HDP).
 - ▶ HDP CERTIFIED ADMINISTRATOR (HDPCA).
 - ▶ HDP CERTIFIED APACHE SPARK DEVELOPER (HDPCD-Spark).
 - ▶ HDP CERTIFIED DEVELOPER (HDPCD)
- ▶ Amazon : AWS.
- ▶ Google : GFS.
- ▶ Microsoft Azure.
- ▶ IBM.



Casquettes en entreprise

- 
- ▶ Big Data Engineer
 - ▶ Data Processing Engineer
 - ▶ Spark Developer
 - ▶ Hadoop Administrator
 - ▶ Big Data Architect
 - ▶ Big Data Scientist
 - ▶ Big Data Analyst
 - ▶ Big Data Consultant
 - ▶ DevOps Engineer (Big Data)
 - ▶ Big Data Machine Learning Engineer

Evolution et Explosion de la donnée

- ▶ Les données ont évolué au cours des 5 dernières années comme jamais auparavant.



- ▶ De nombreuses données sont générées chaque jour dans tous les secteurs d'activité.

Evolution et Explosion de la donnée

- Diversité de l'intérêt, d'extraction et objectifs : le **trésor caché**.



Evolution de la donnée

Every minute, users send 31.25 million messages and watch 2.77 million videos on Facebook



Walmart handles more than 1 million customer transactions every hour



300 hours of video are uploaded every minute on YouTube



40,000 search queries are performed on Google per second, i.e. 3.46 million searches a day

55 billion messages and 4.5 billion photos are sent each day on WhatsApp



IDC reports that by 2025, real time data will be more than a quarter of all the data



By 2025, the volume of digital data will increase to 163 zettabytes

DOMO Statistics

- ▶ Plus de détails :
 - ▶ DOMO
- ▶ Statistiques des pays asiatiques en occurrence la Chine ainsi que la Russie.

Pourquoi les Big Data ?



Pourquoi les Big Data ?



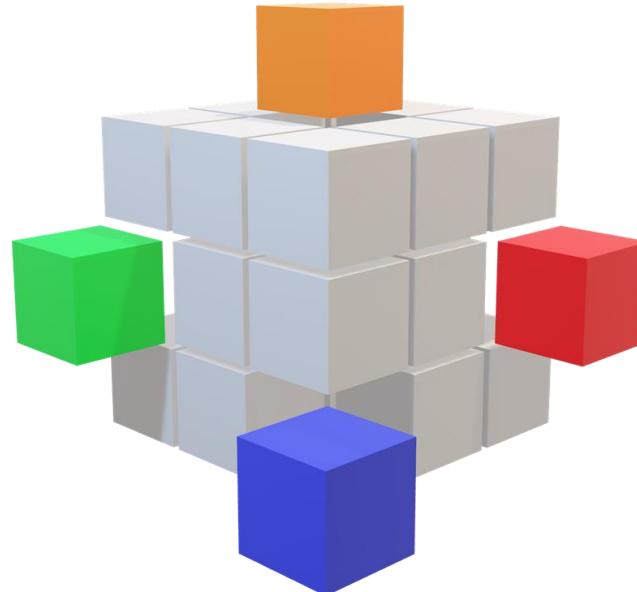
Personalized News Feed

Pourquoi les Big Data ?



Pourquoi les Big Data ?

Si on torture suffisamment les données, elles finiront bien par confesser



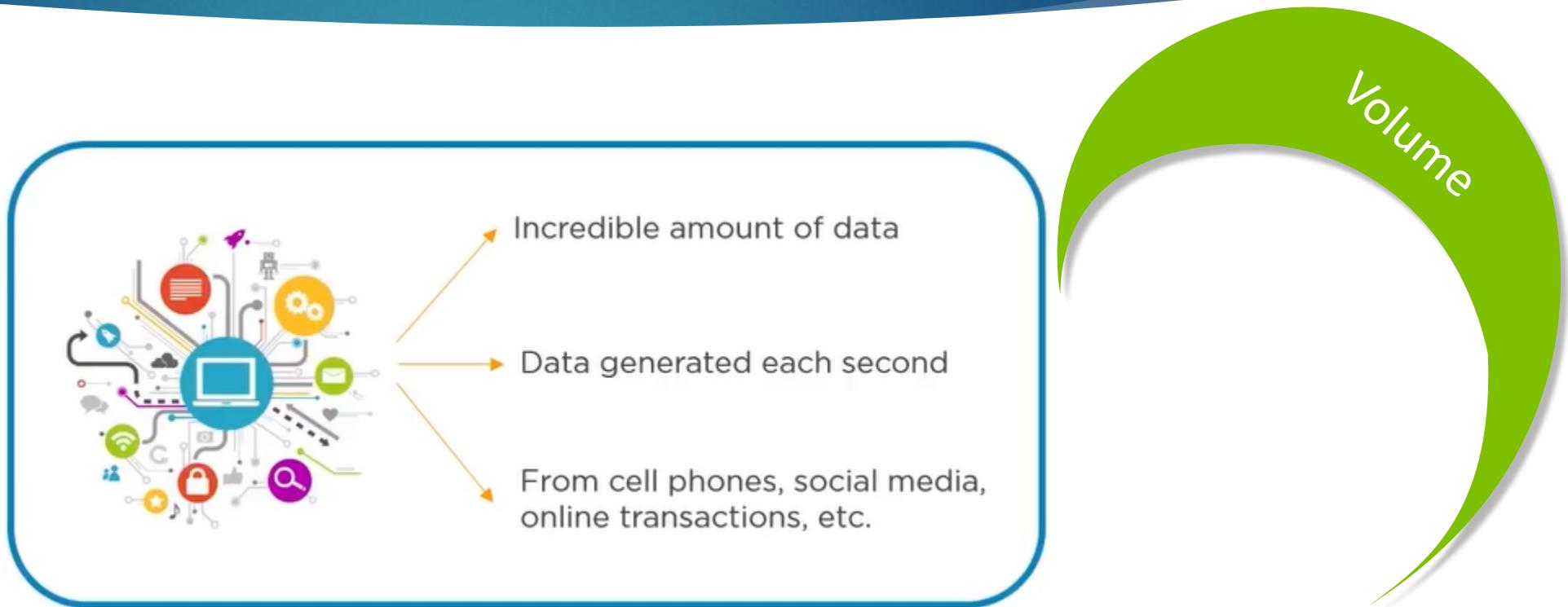
Définition

- ▶ Le terme Big Data se réfère aux technologies qui permettent aux entreprises d'analyser rapidement un volume de données très important, d'obtenir une vue générale, extraire une certaine valeur ajoutée.
- ▶ En mixant intégration et stockage de données, analyse prédictive et applications, le Big Data permet de gagner en temps, en efficacité, en qualité d'interprétation de données.

Problématiques des 5-Vs



Problématiques des 5-Vs



Problématiques des 5-Vs



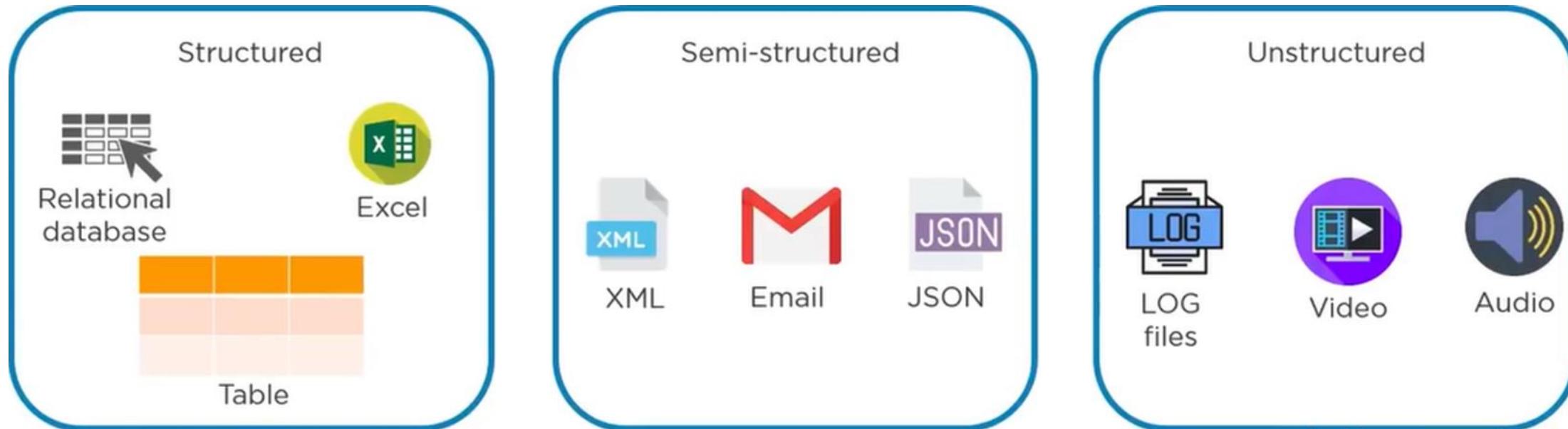
Speed at which data is:

- Generated
- Collected
- Analyzed



Velocity
Vélocité

Problématiques des 5-Vs



Problématiques des 5-Vs

The diagram features a large yellow circle on the left containing the word "Veracité" above "Veracity". To its right is a white rectangular box with a blue border. Inside the box, the text reads: "Extracting loads of data is not useful if the data is messy and poor in quality" and "Twitter posts with abbreviations, spelling mistakes, etc.". Below this text is a small illustration showing a laptop screen with a bar chart, an arrow pointing to a magnifying glass over a pie chart, and a sad face emoji, all enclosed in a blue rounded rectangle.

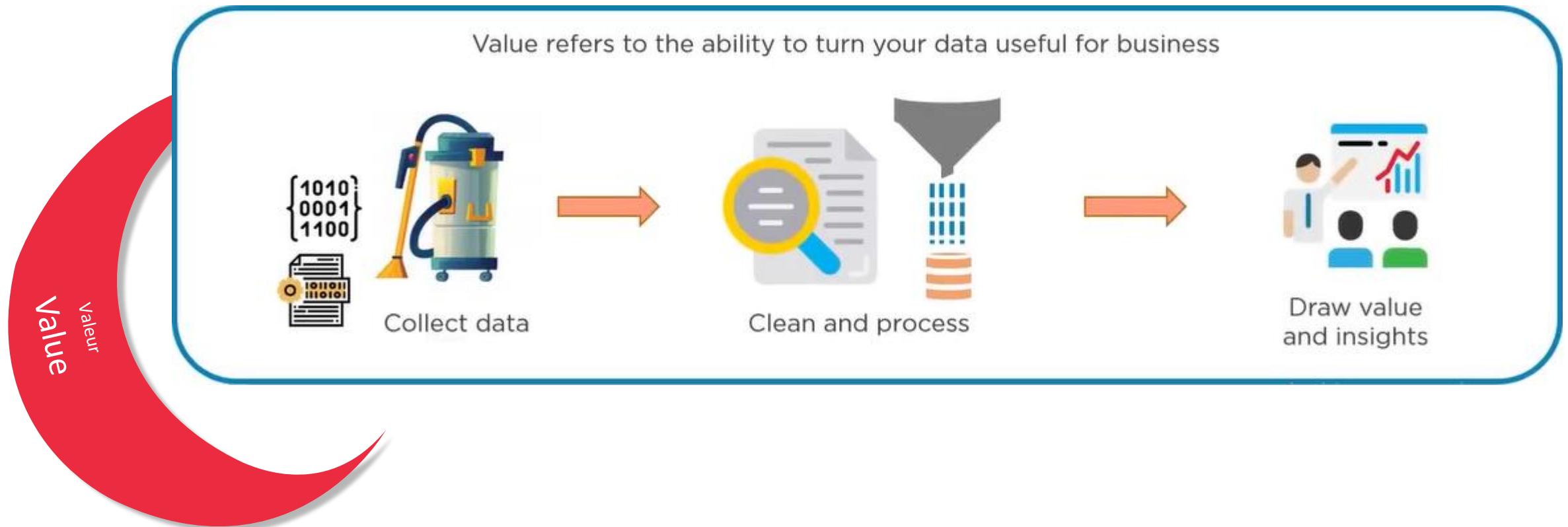
Vérité
Veracity

Extracting loads of data is not useful if the data is messy and poor in quality

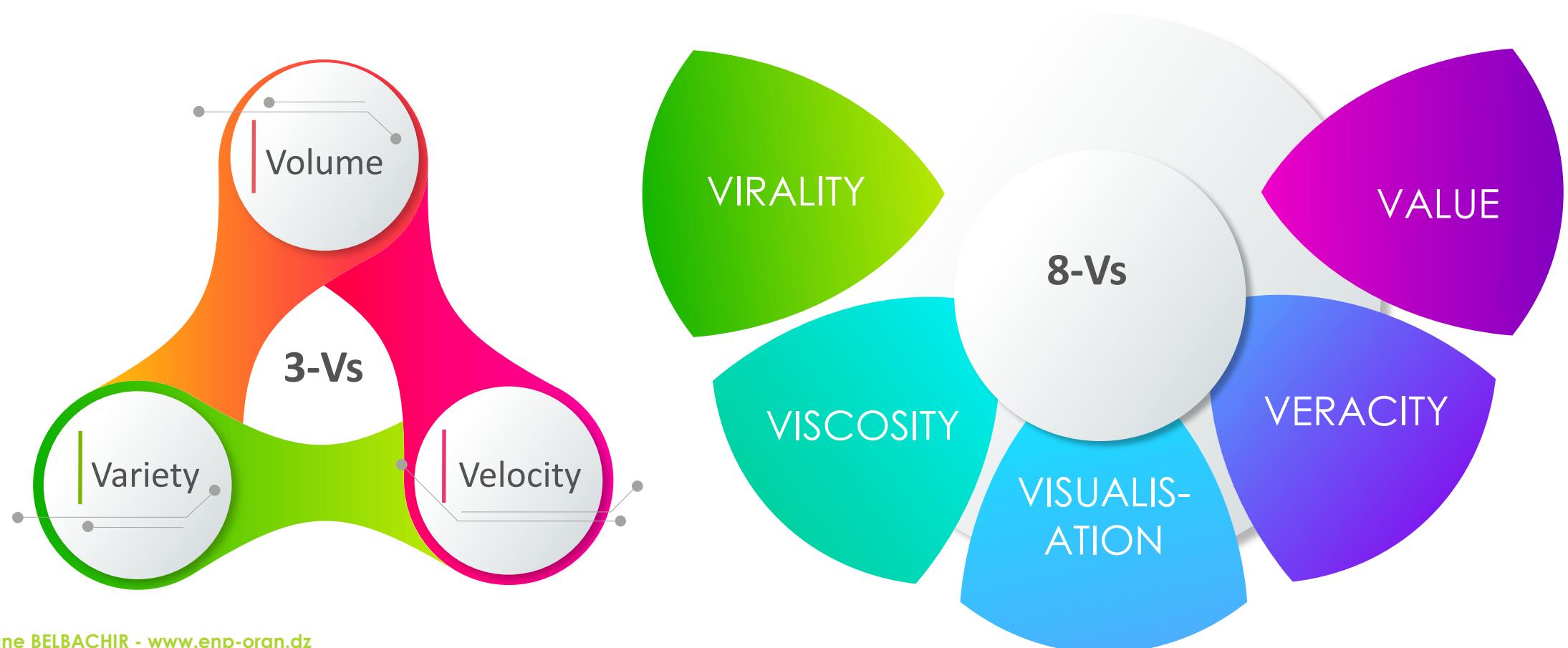
Twitter posts with abbreviations, spelling mistakes, etc.

Difficult to analyze messy data

Problématiques des 5-Vs



Problématiques de Big Data.



3V de base

System
Should
Provide

Scalability



Consistency



Availability

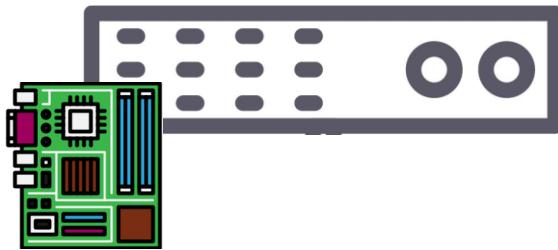


Volume

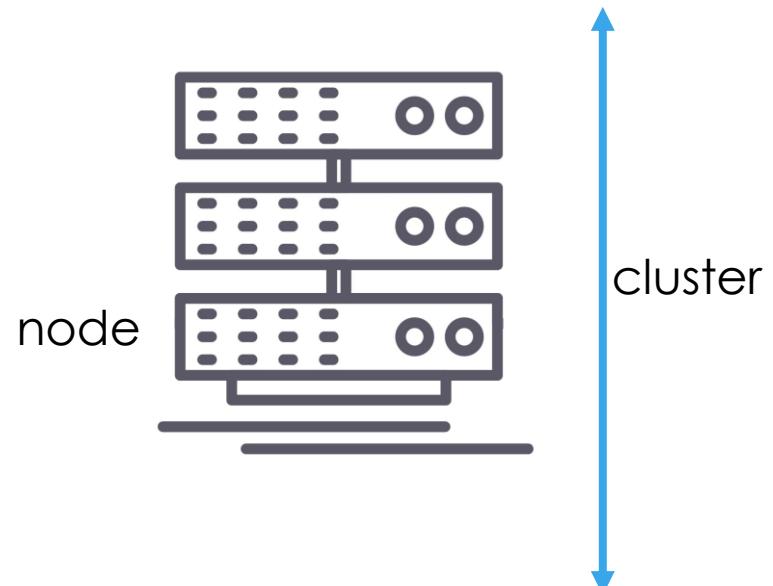
- ▶ Scaling (Scalabilité)

- ▶ Scale-Up

(Verticale : adding physical resources)



- ▶ Scale-Out

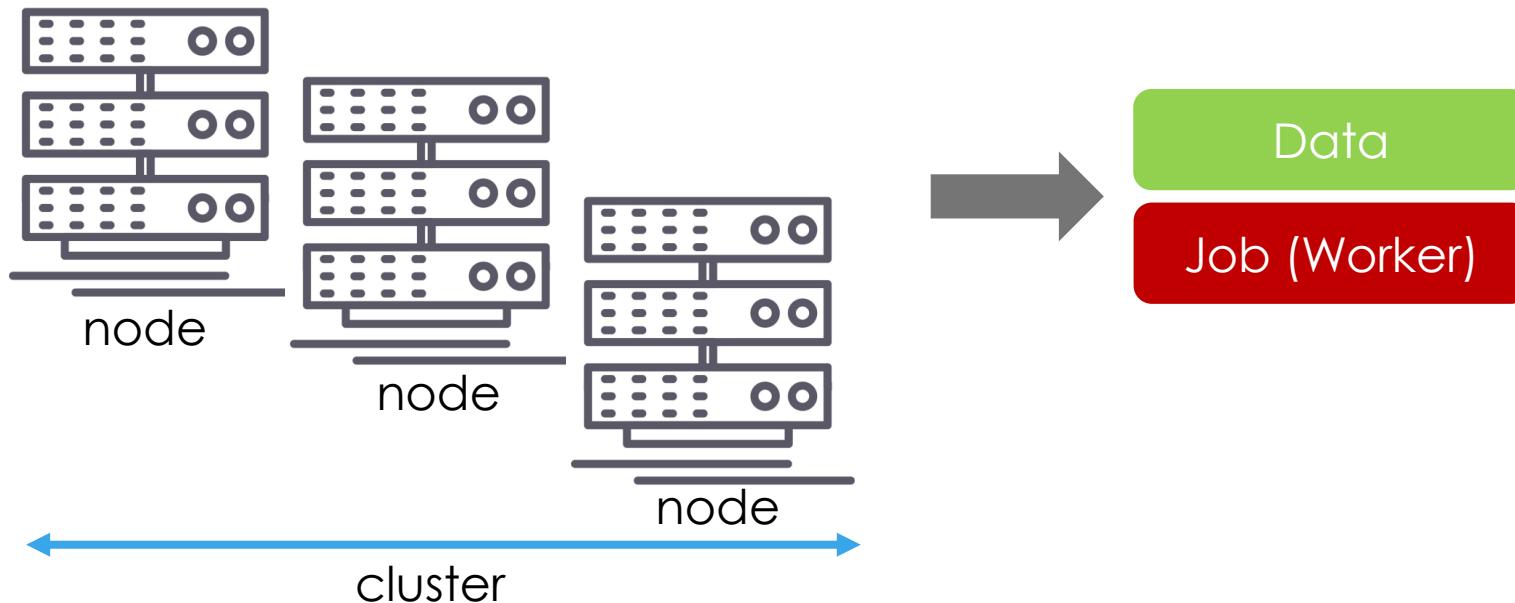


Cluster

- ▶ Deux approches ont vu le jour :
- ▶ Cluster local
- ▶ Cluster via Internet

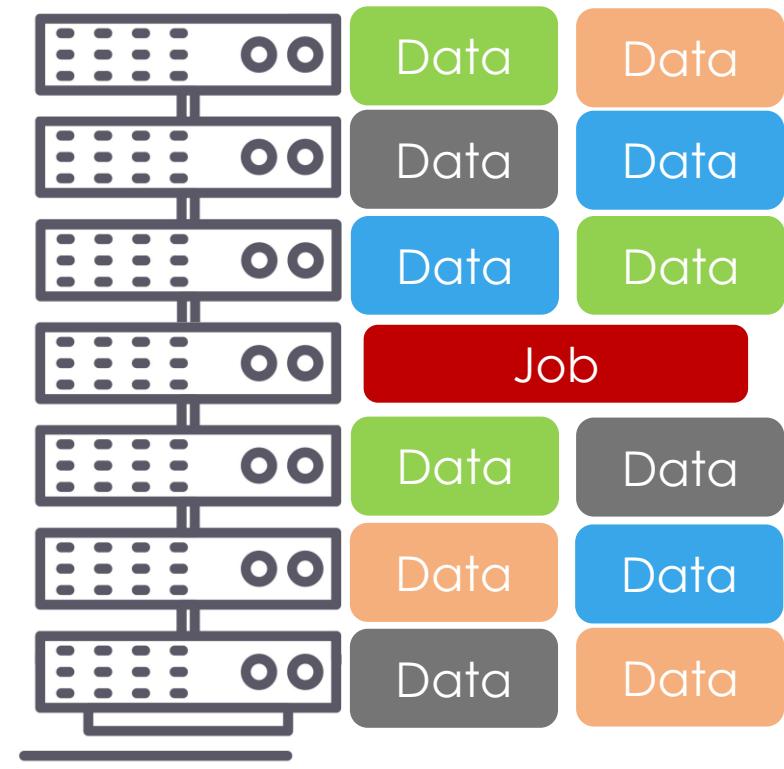
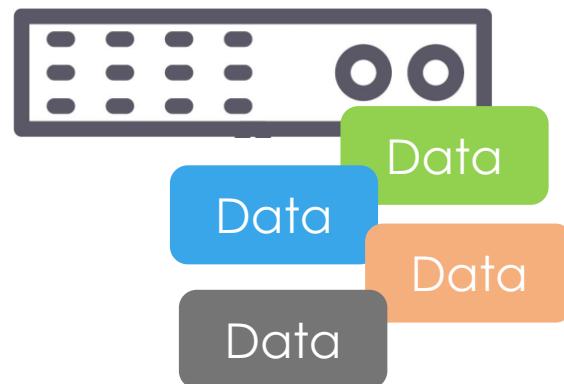
Scaling in Big Data

- ▶ Data and Data Handling (Colocalité des données et des traitements)



Répliques

- ▶ Tolérance au partitionnement



Calcul distribué

- ▶ Le calcul distribué désigne l'exécution d'un traitement informatique sur une multitude de machines différentes (un cluster de machines) de manière transparente.
- ▶ Problèmes soulevés
 - ▶ Accès et partage des ressources pour toutes les machines.
 - ▶ Extensibilité : on doit pouvoir ajouter de nouvelles machines pour le calcul si nécessaire.
 - ▶ **Tolérance aux pannes** : une machine en panne faisant partie du cluster ne doit pas produire d'erreur pour le calcul dans son ensemble.
 - ▶ Transparence : le cluster dans son ensemble doit être utilisé comme une seule et même machine traditionnelle .
 - ▶ **Hétérogénéité** : les machines doivent pouvoir avoir différentes architectures et caractéristiques

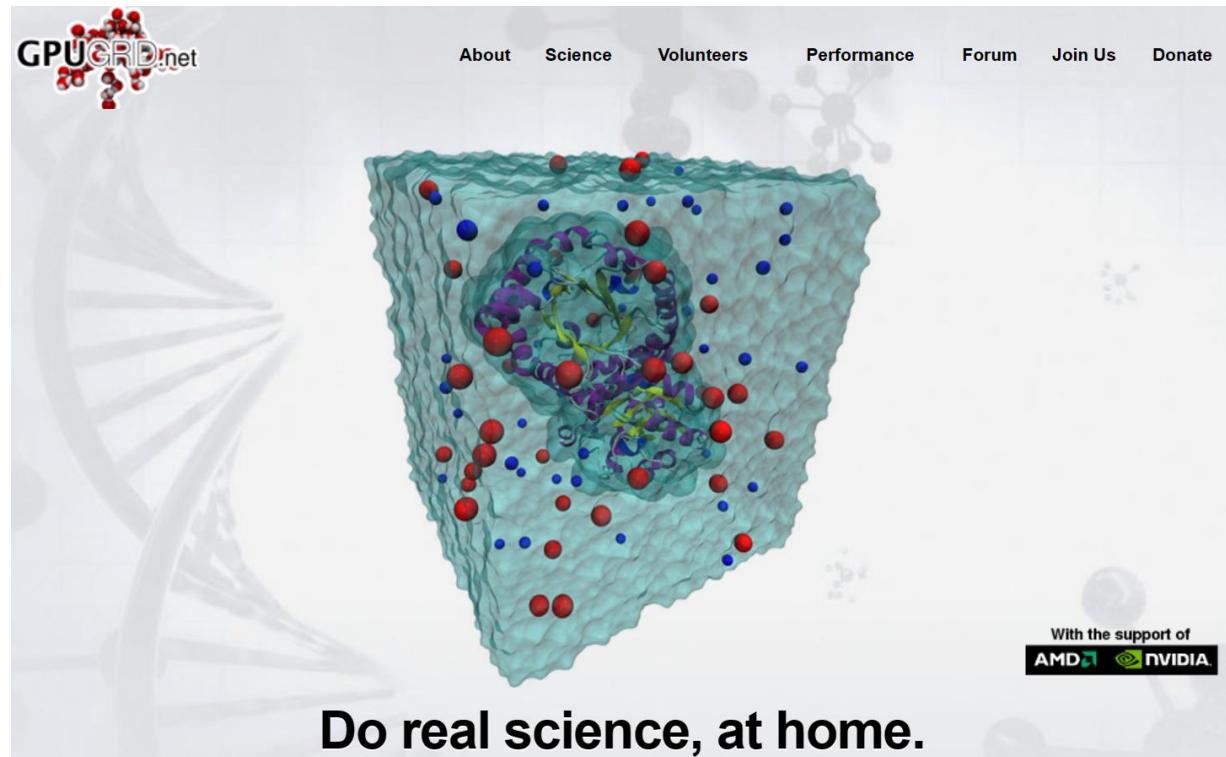
Distributed Systems

- ▶ SETI@Home/BOINC

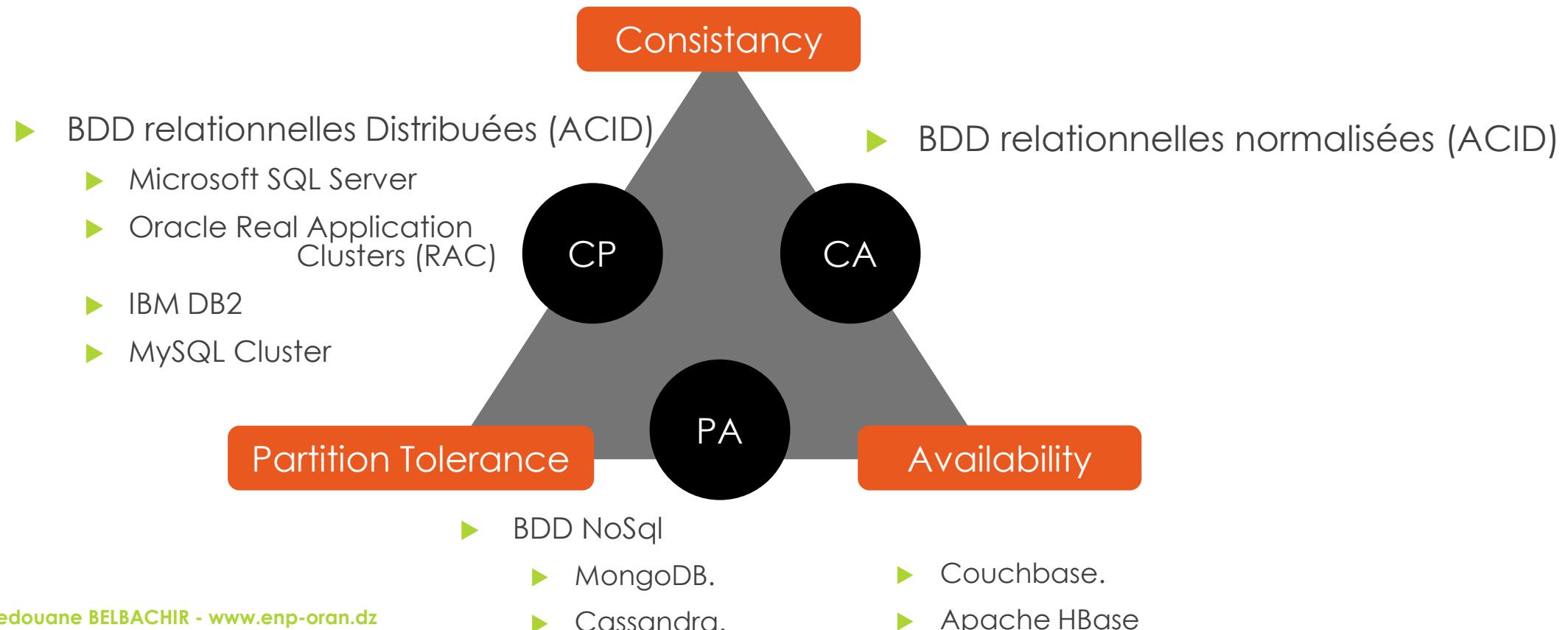


Distributed Systems

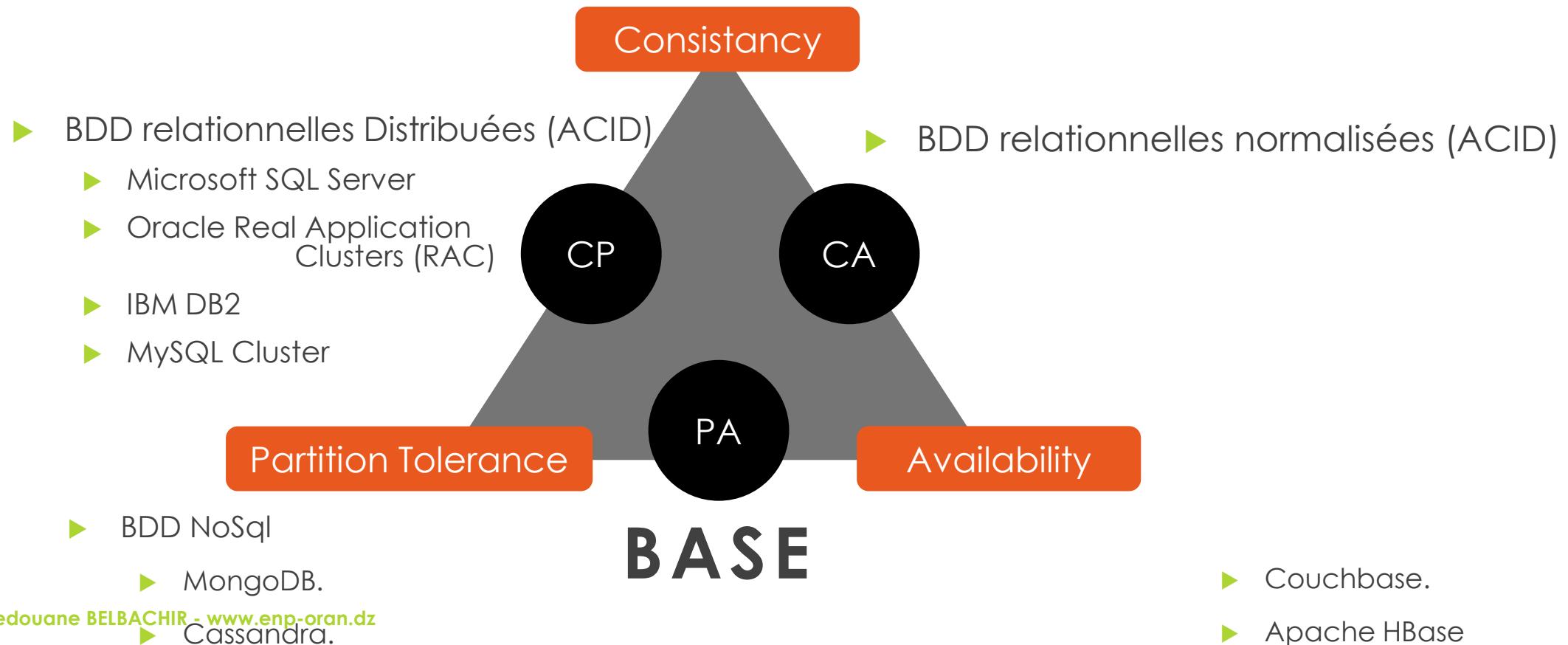
- ▶ GPUGRID.net 2007



Rappel sur le CAP



Rappel sur le CAP



Historique : 2002



Doug Cutting
(directeur archive.org)



Web-scale search engine toolkit



Mike Cafarella
(étudiant)

Historique : 2003/2004

- ▶ Le département « R and D » de recherche de Google publie deux whitepapers :

GFS (un système de fichier distribué)

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the compo-

Map/Reduce

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the pro-

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

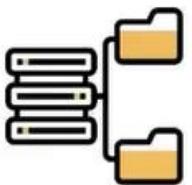
As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is in-

Etude de cas : Google

Google



Thousands of search queries were raised per second



Every query read 100's of MB of data and consumed
10's of billions of CPU cycles



Need for large, distributed, highly fault tolerant file
system to store and process the queries



Traditional
data server

Solution

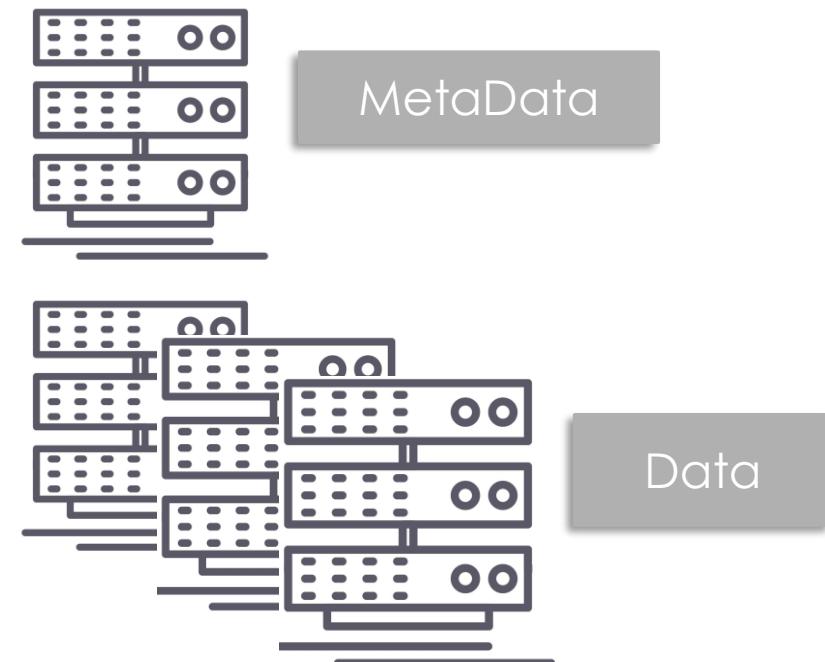
Google File System
(GFS)

Google Solution : GFS

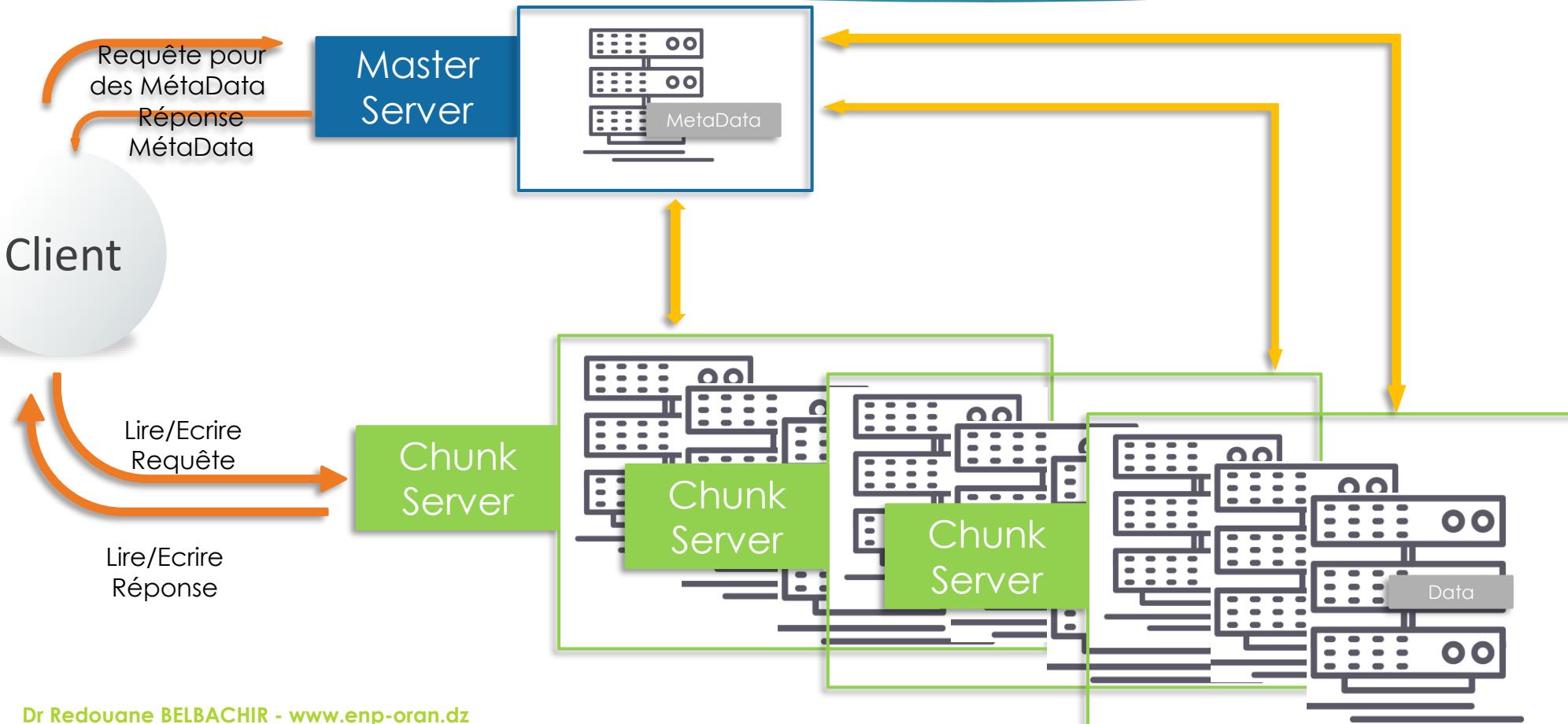
► Nœuds :

Master
Server

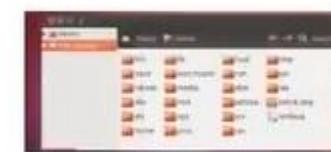
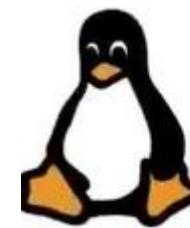
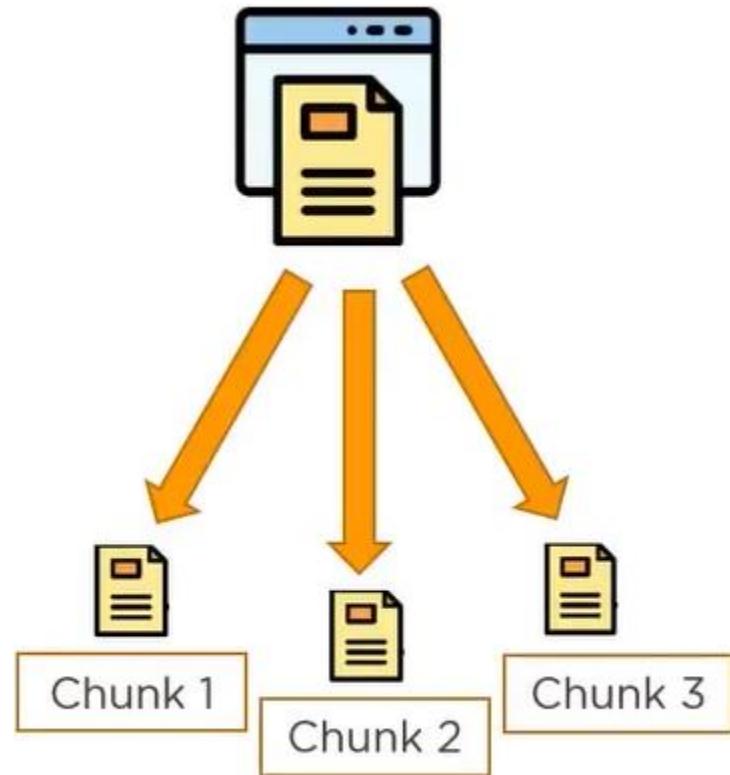
Slave
(Chunk)
Servers



Google Solution : GFS

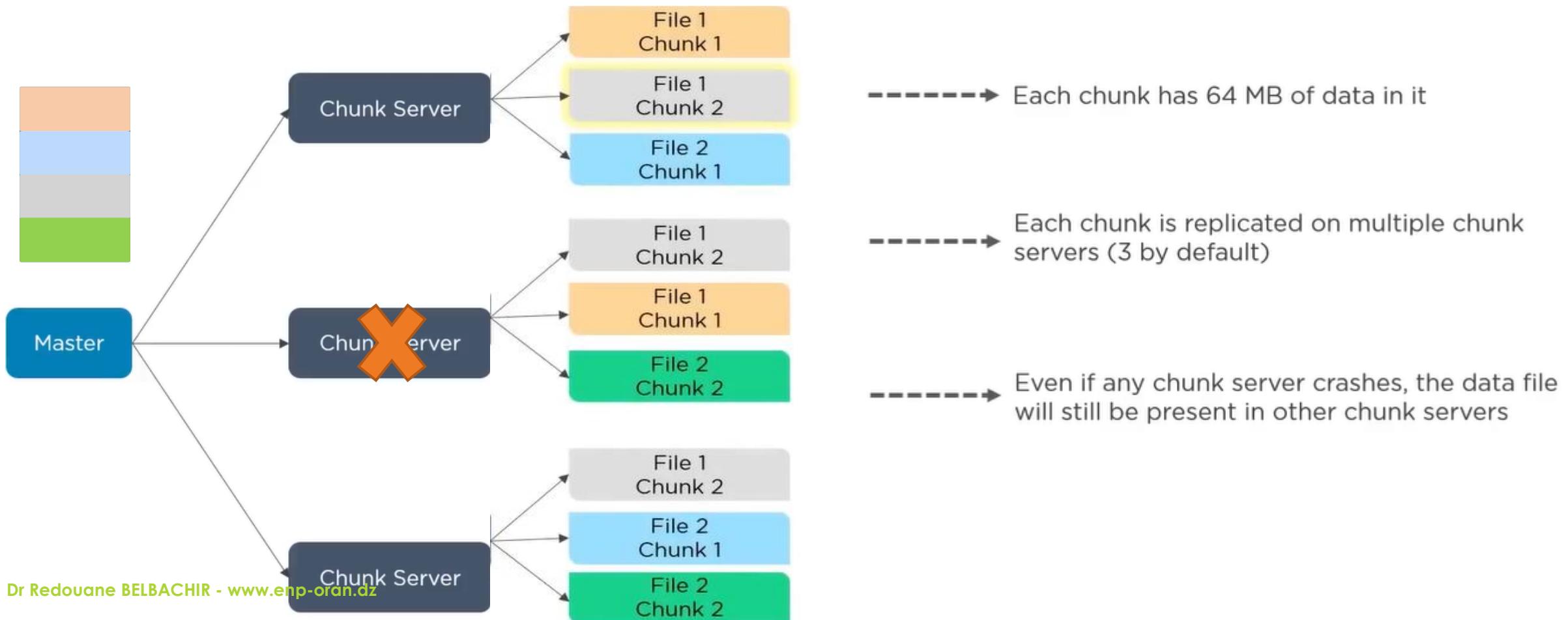


Chunk Server



Ext4, XFS

GFS : détails

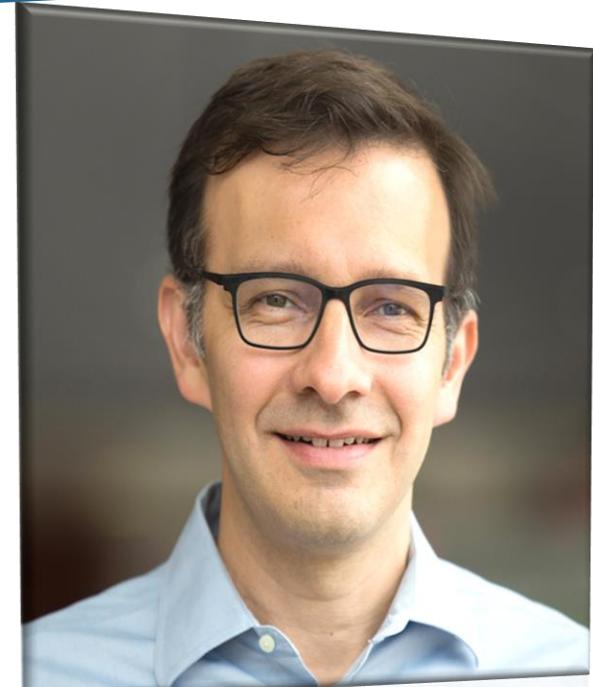


Séquence d'événements : 2004



Doug Cutting
(directeur archive.org)

Développent un framework
(encore assez primitif)
inspiré des travaux de Google
et portent leur projet Nutch
sur ce framework.



Mike Cafarella

Séquence d'événements : 2006



Doug Cutting

- ▶ En charge d'améliorer l'indexation du moteur de recherche de Yahoo.
 - ▶ Il exploite son framework en créant une version améliorée
 - ▶ il le nomme Hadoop (le nom d'un éléphant en peluche de son fils)
 - ▶ Le cluster du projet contenait au maximum 20 machines.

YAHOO!

Séquence d'événements : 2008 et 2011

- ▶ **2008**
- ▶ Hadoop est exploité par le moteur de recherche de Yahoo.
- ▶ **2011**
- ▶ Hadoop est désormais utilisé par de nombreuses autres entreprises et universités
 - ▶ le cluster Yahoo comporte 42000 machines et des centaines de
 - ▶ petaoctets d'espace de stockage



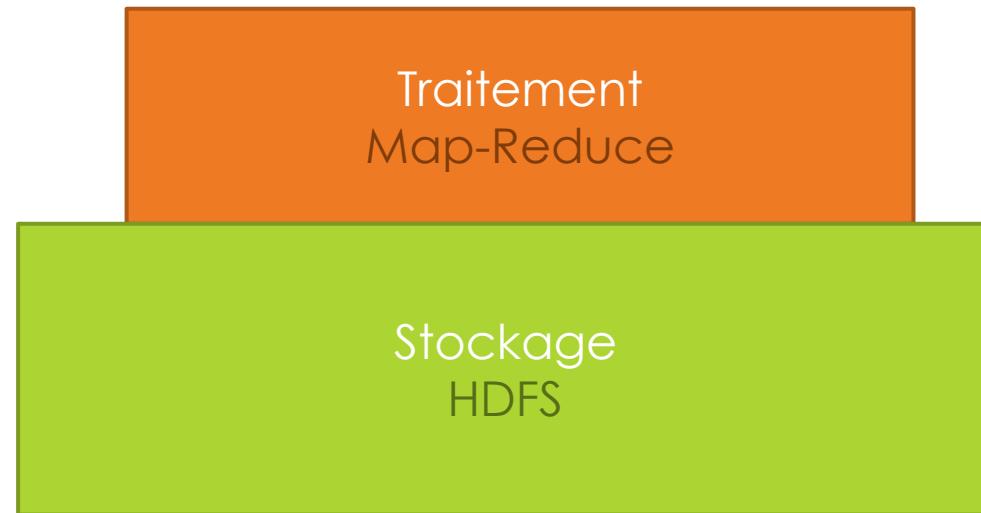
Hadoop FrameWork



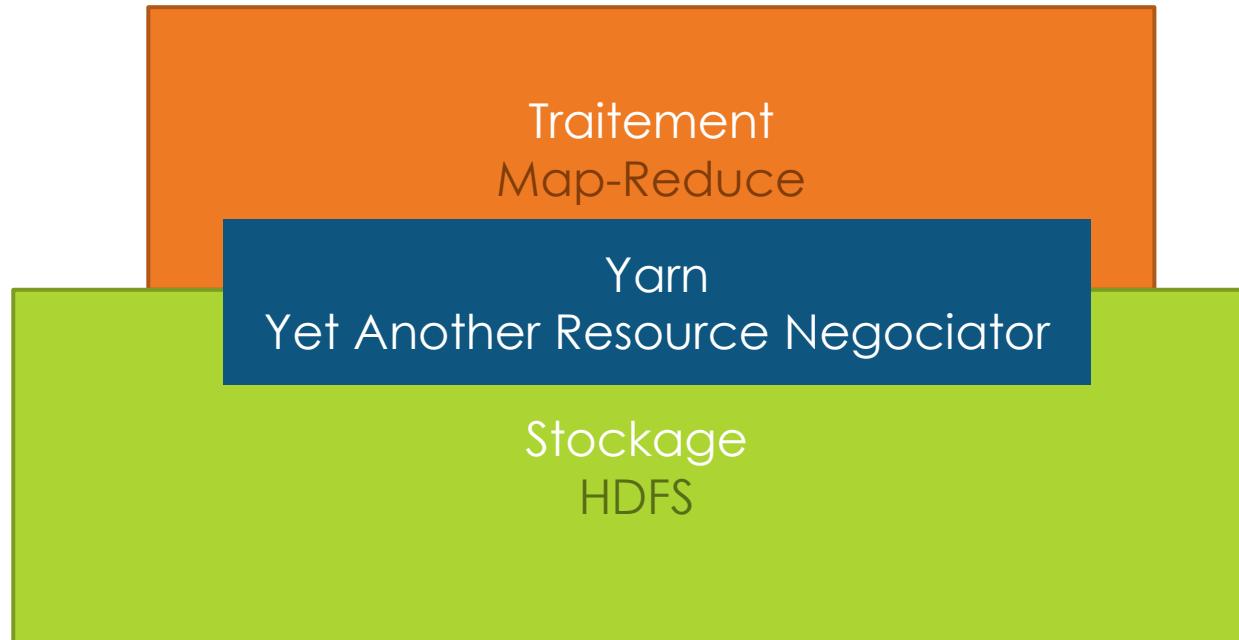
Constitué de deux composants clés



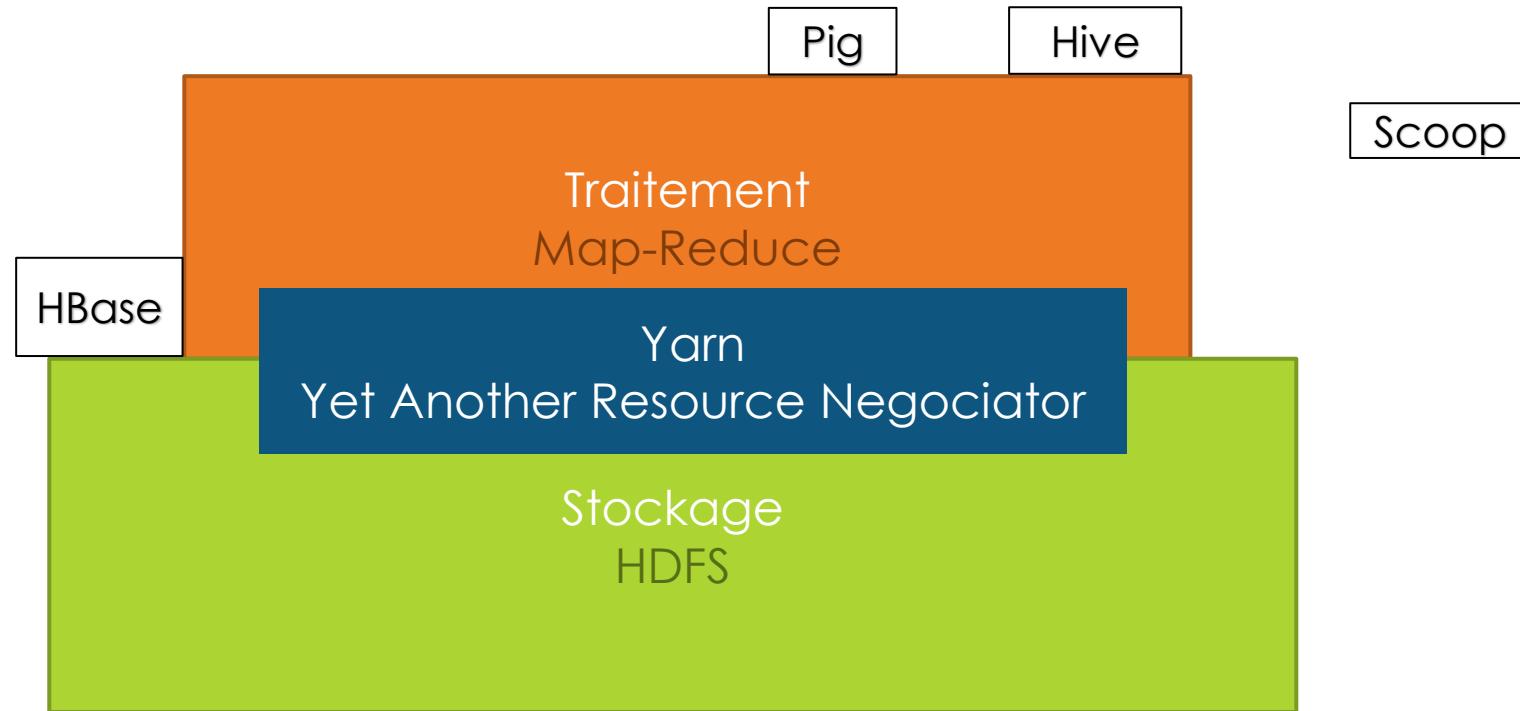
Hadoop FrameWork et son écosystème



Hadoop FrameWork et son écosystème



Hadoop FrameWork et son écosystème



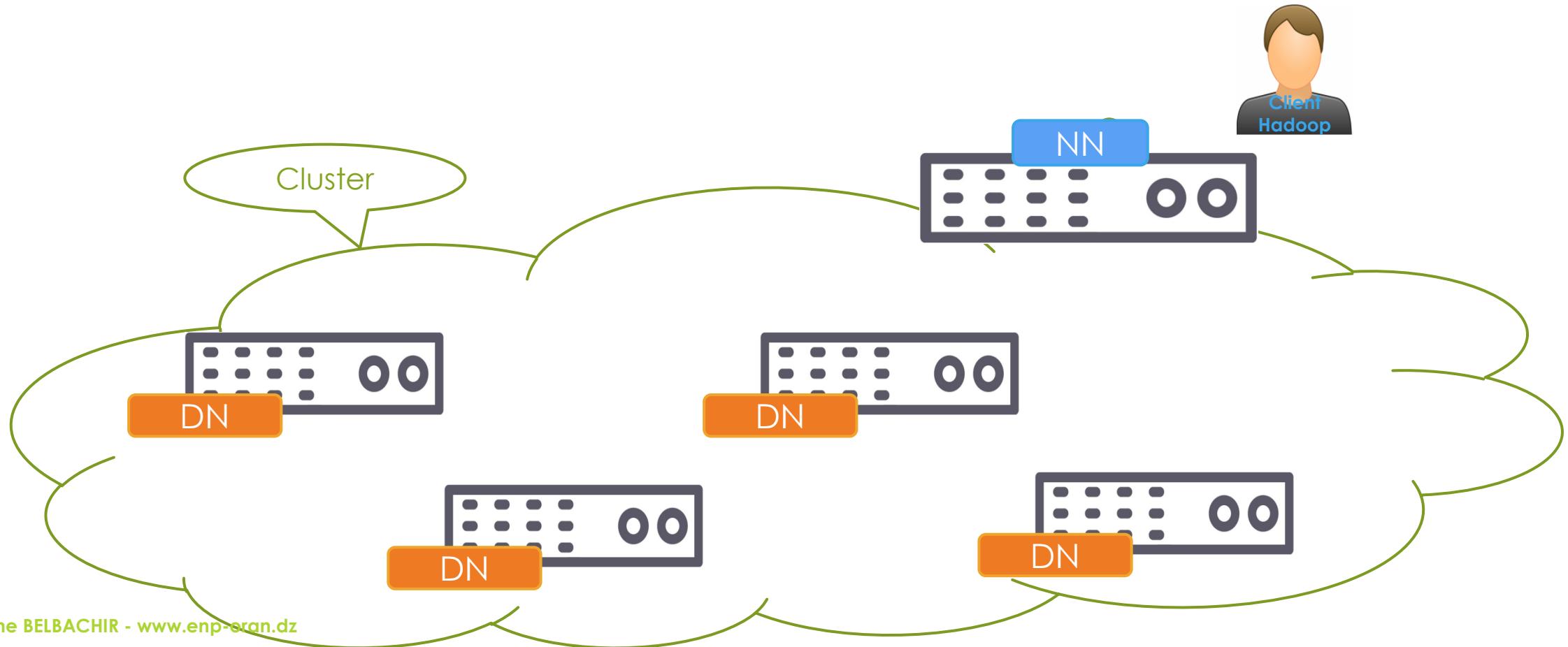
Principe

- ▶ Diviser les données.
- ▶ Les sauvegarder sur une collection de machines, appelées cluster.
- ▶ **Traiter les données directement là où elles sont stockées**, plutôt que de les copier à partir d'un serveur distribué.
- ▶ Il est possible d'ajouter des machines à votre cluster, au fur et à mesure que les données augmentent

HDFS : Hadoop Distributed File System

- ▶ Un système de Fichier standard comme les systèmes de fichiers FAT32, NTFS... sauf qu'il est :
 - ▶ Distribué : Les données sont réparties sur toutes les machines du cluster.
 - ▶ Répliqué : Si une des machines du cluster tombe en panne, aucune donnée n'est perdue.
- ▶ Les données sont répliquées de telle sorte que si une panne affecte un rack de serveur entier (exemple : un incident d'alimentation), cette dernière ne provoque aucune perte de données

HDFS : Architecture, Exemple

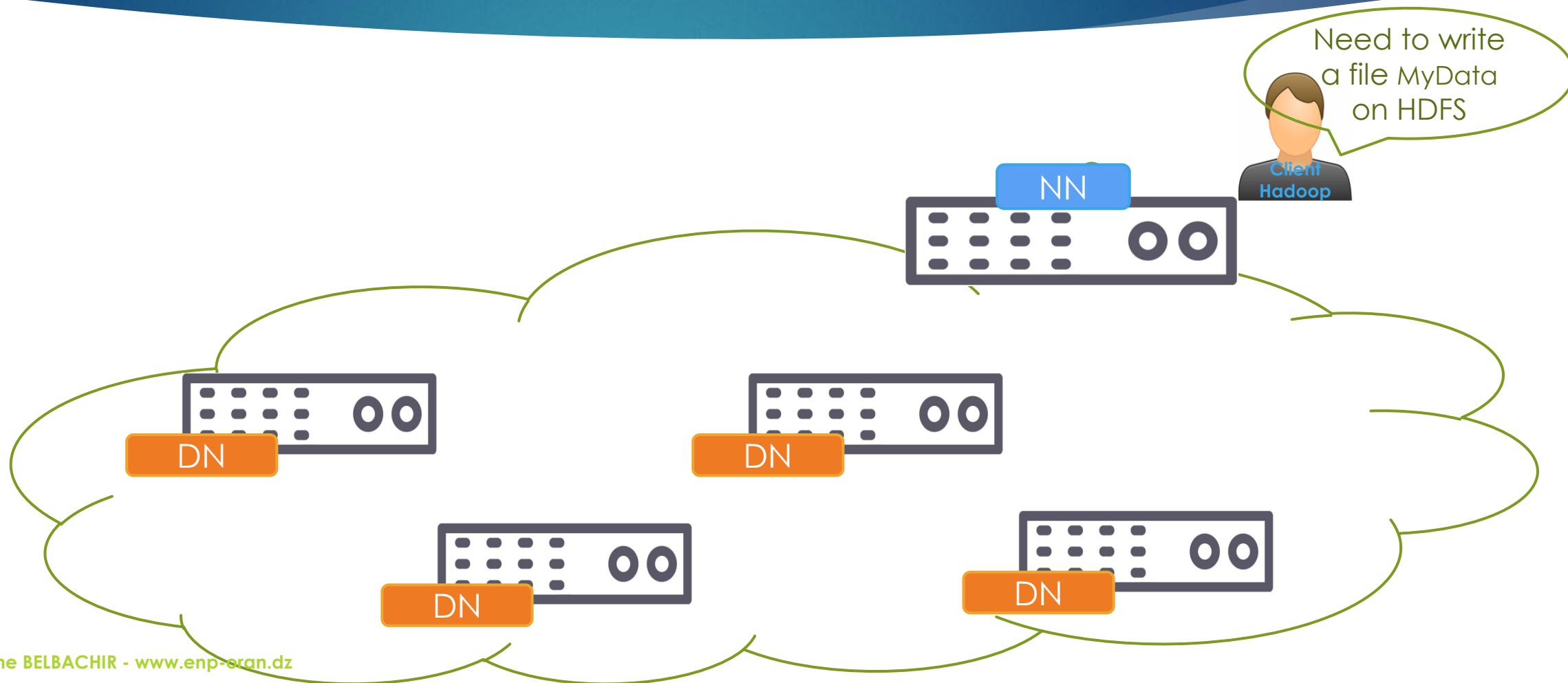


Hadoop : Architecture, NameNode

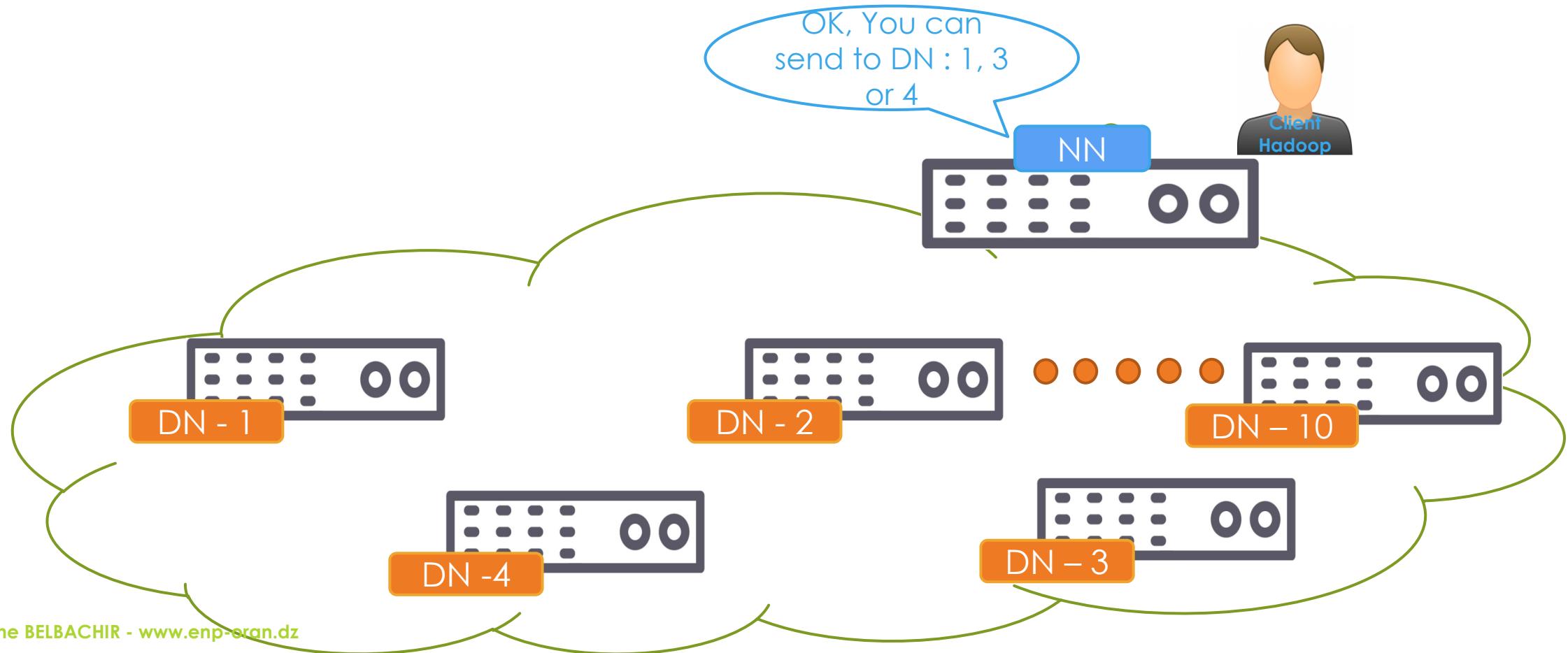
- ▶ Un seul NameNode dans tout le cluster Hadoop.
- ▶ Stocke les informations relatives aux noms de fichiers et l'emplacement des blocs de données (dans le cluster) correspondant à chaque fichier.
 - ▶ Contient les Métadonnées.
 - ▶ Contient l'arborescence.
- ▶ Une visibilité sur ce qui se passe aux DN.
- ▶ Le seul disponible afin de recevoir les requêtes des Clients Hadoop.
- ▶ Si un client veut faire une requête sur les données, le NN l'oriente vers la bonne machine **(Next Slide Example)**
 - ▶ Le NN permet, l'organisation et la synchronisation du travail.



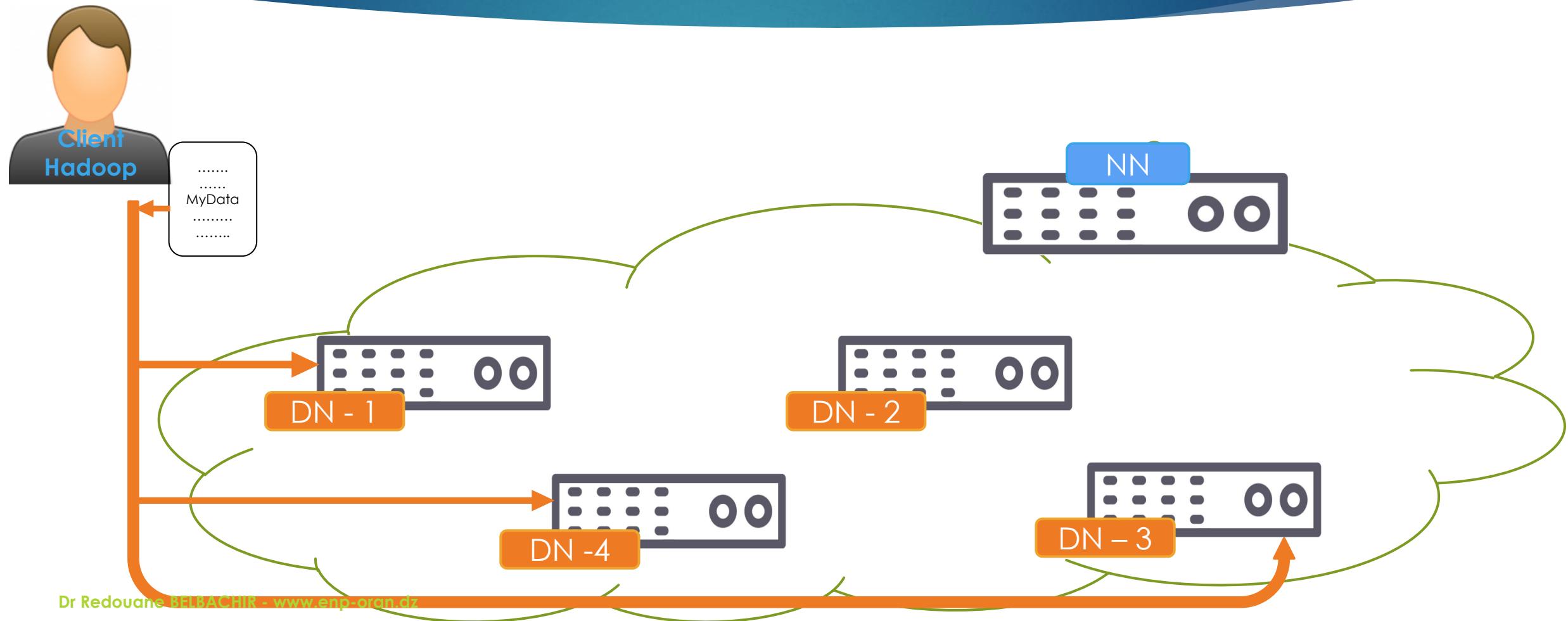
HDFS : Architecture, Exemple



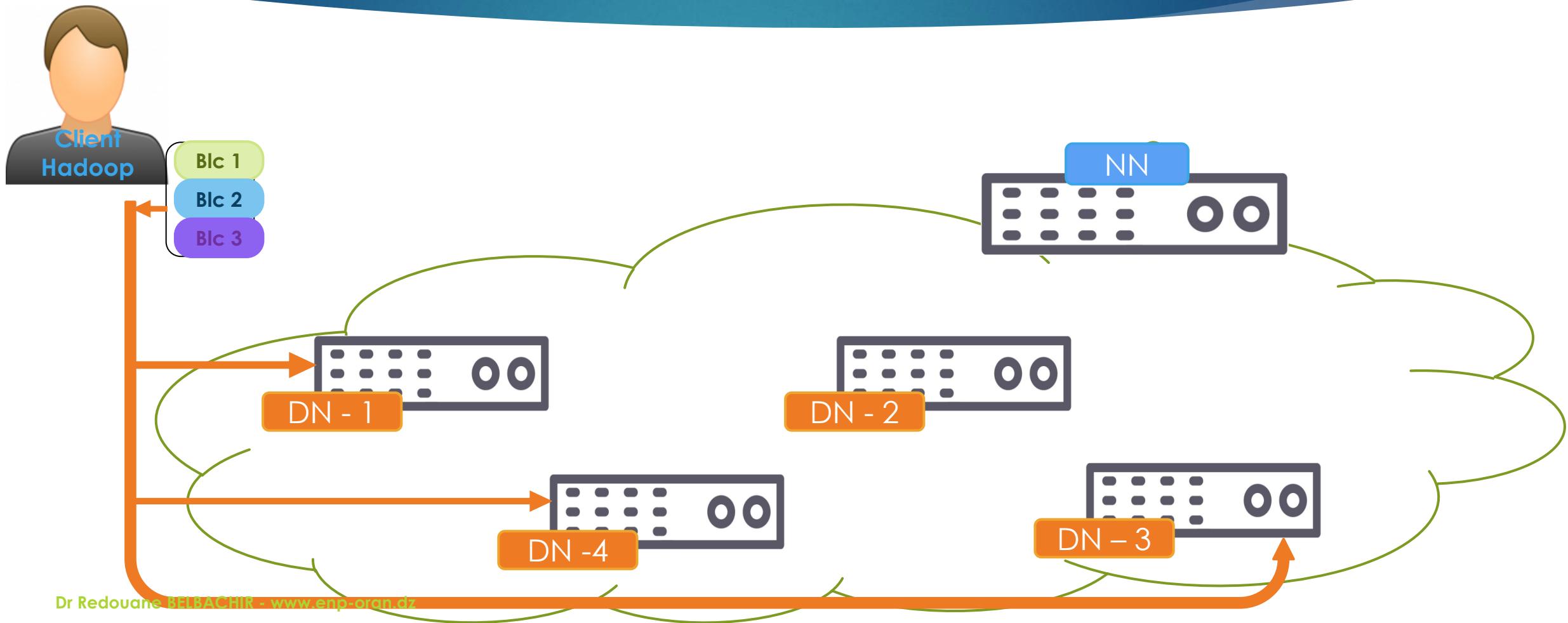
HDFS : Architecture, Exemple



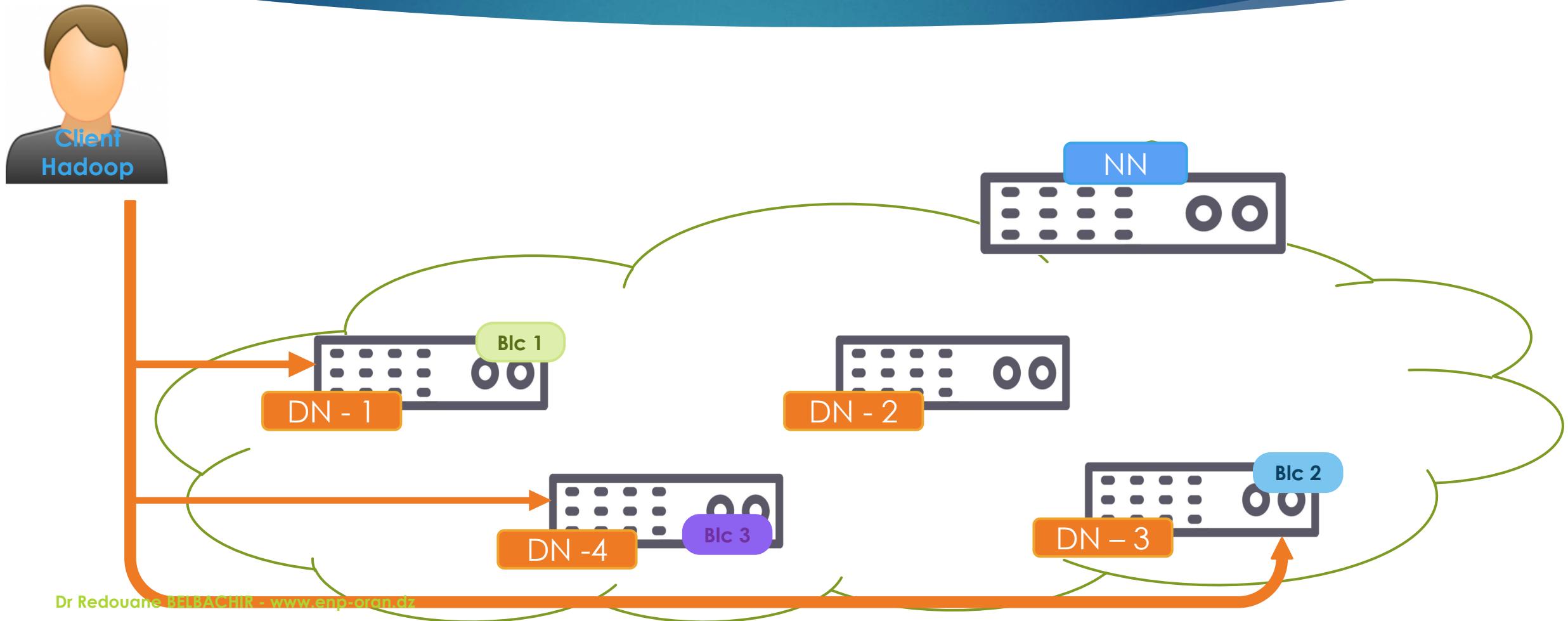
HDFS : Architecture, Exemple



HDFS : Architecture, Exemple



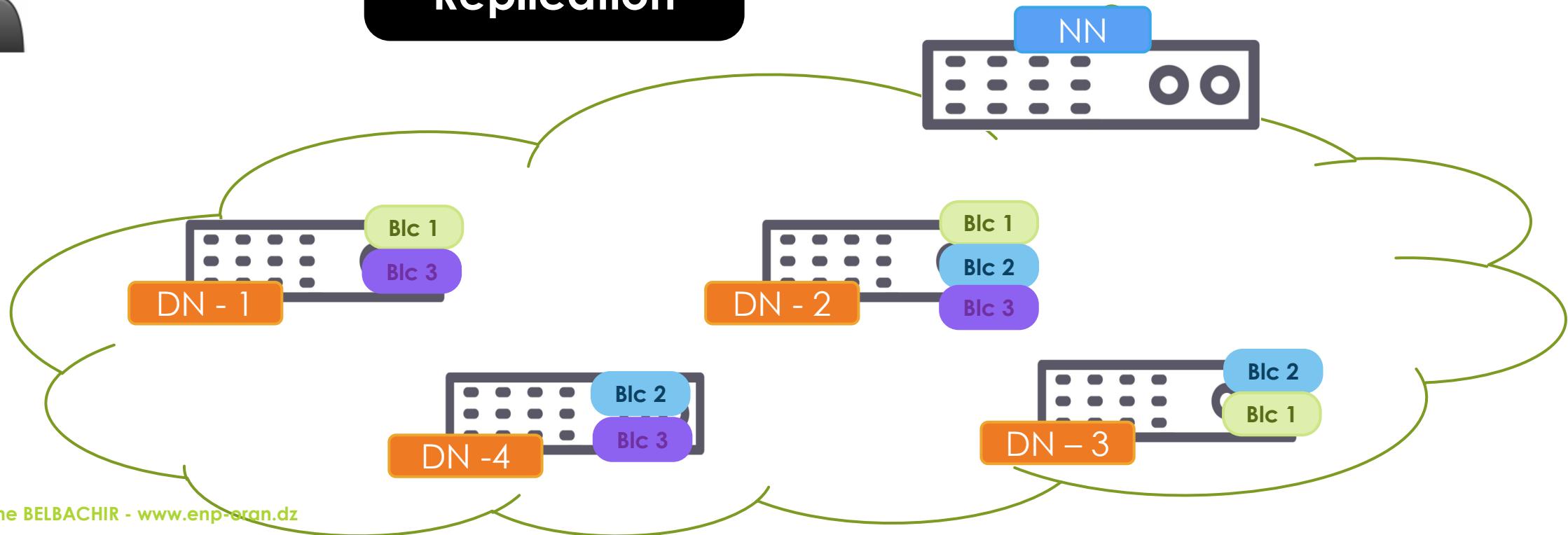
HDFS : Architecture, Exemple



HDFS : Architecture, Exemple



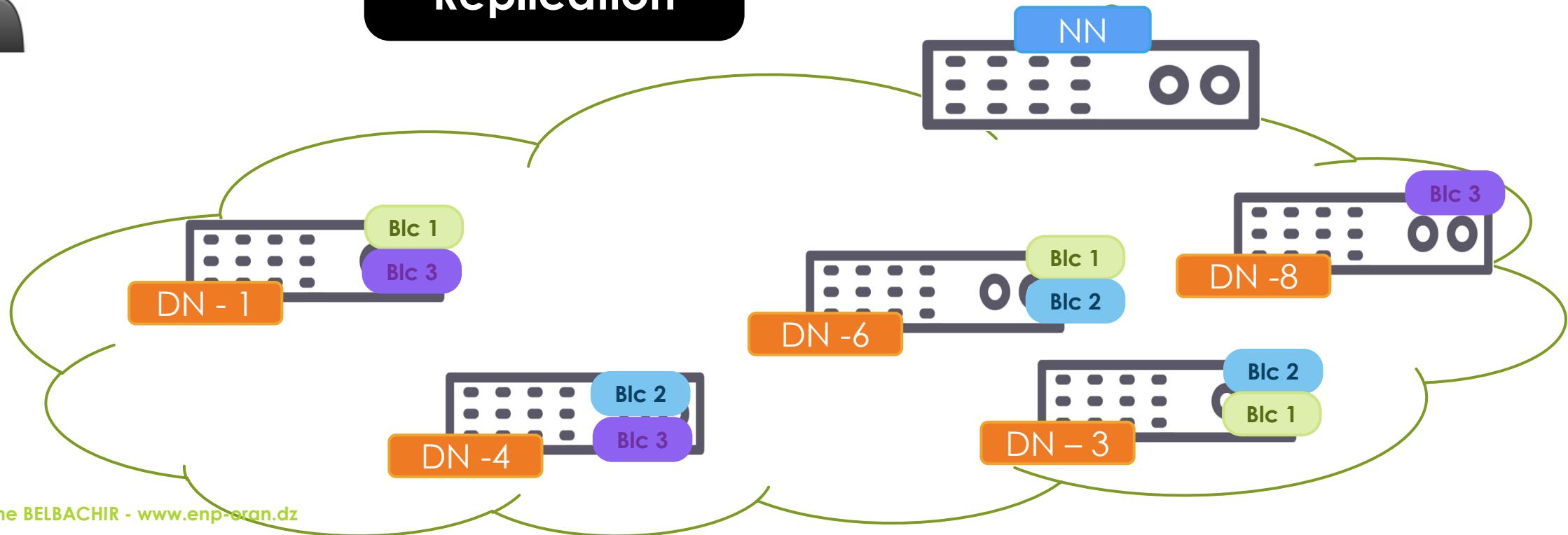
Facteur de RéPLICATION



HDFS : Architecture, Exemple



Facteur de RéPLICATION



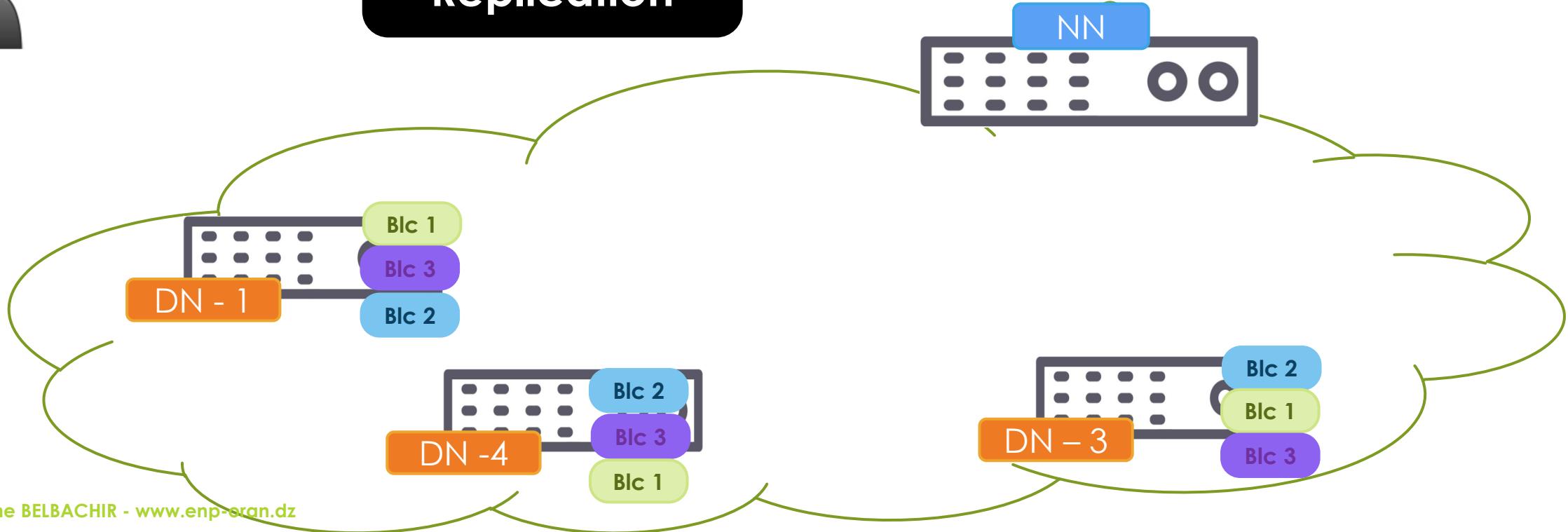
Hadoop : Architecture, DataNode

- ▶ Envoie régulièrement des HeartBeat au NN.
- ▶ Dès que le Master détecte qu'un DN n'est plus visible:
 - ▶ Réplique une nouvelle fois les données, considérées comme perdues, DN concerné.
- ▶ Le DN Stocke les blocs de données lui même.
- ▶ Communique constamment avec le NameNode pour :
 - ▶ Recevoir de nouveaux blocs
 - ▶ Indiquer les blocs présents dans le DataNode
 - ▶ Signaler des erreurs
- ▶ Les premières versions de Hadoop, les DataNodes ne peuvent communiquer entre elles.

HDFS : Architecture, Exemple

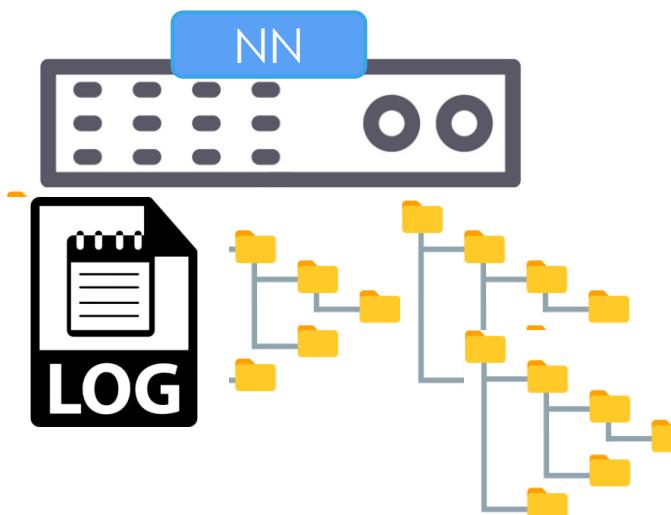


Facteur de RéPLICATION



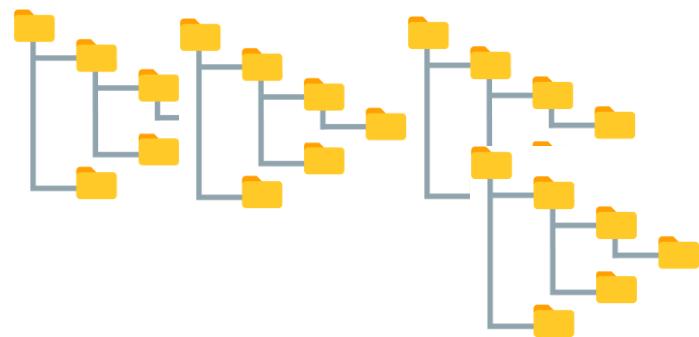
NameNode

- ▶ Dans le NN, on a 2 structures de données qu'il utilise :
(MétaData)
 - ▶ FS_image : Parmi ses contenus, un snapshot des arborescences.
 - ▶ Edit_Log : (Availability).



FS-Image

- ▶ Conserve l'arborescence HDFS en mémoire vive.
- ▶ Rapide en lecture : hdfs ls ...
- ▶ En Ecriture, **très lourde**.
 - ▶ Nécessite les modifications en mémoire.
 - ▶ Et, sur l'ensemble des DN contenant les blocs concernés.
 - ▶ A noter, le FS-Image est de très grande taille.
- ▶ **Availability**
 - ▶ Solution : Edit_Log.



Edit_Log

- ▶ To Do list on FS-Image.
 - ▶ Journal stocker sur Disque.
- ▶ Contenant les opérations qui sont sensées se faire sur FS-Image.
 - ▶ les informations de modifications.
 - ▶ Créer par le NN et mis-a-jour a chaque réception de requête.
- ▶ De façon régulière, le NN effectue les modifications sur le FS-Image a partir du Edit_Log.
 - ▶ **CheckingPoint.**
 - ▶ Mais en même temps il peut prendre quelques minutes.



Edit_Log

- ▶ Problématique : les modifications régulières a partir de l>Edit_Log sont consommatrices de temps.

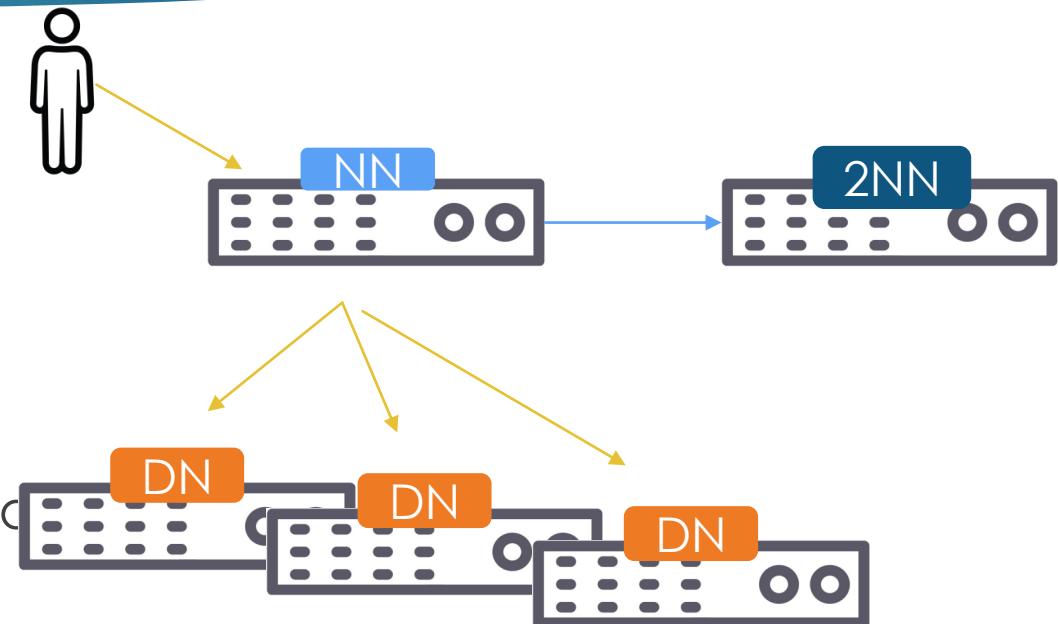
- ▶ Availability issue.

- ▶ Solution : Secondary NN.



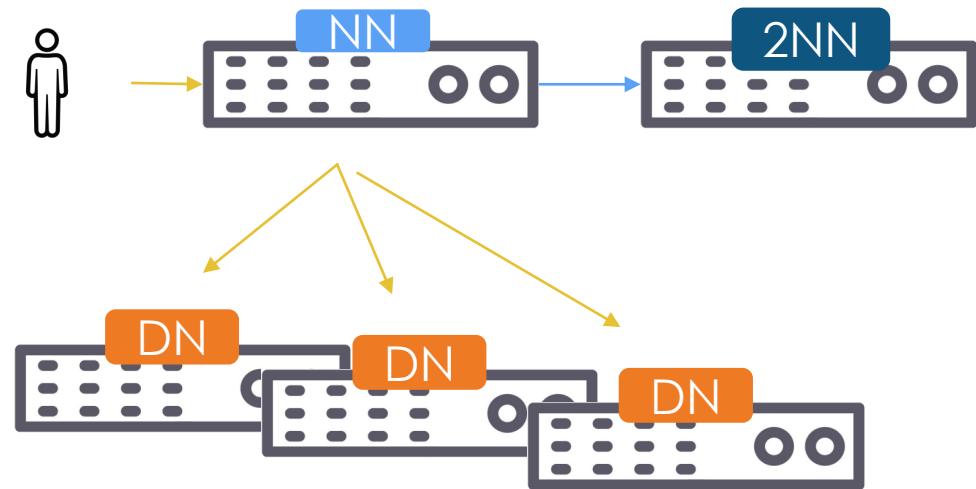
Secondary NameNode

- ▶ Travaille d'une façon passive.
- ▶ Pas de distribution de charge avec le NN.
- ▶ Le NN reste toujours le seul qui reçoit les requêtes et le seul qui communique avec les DN.



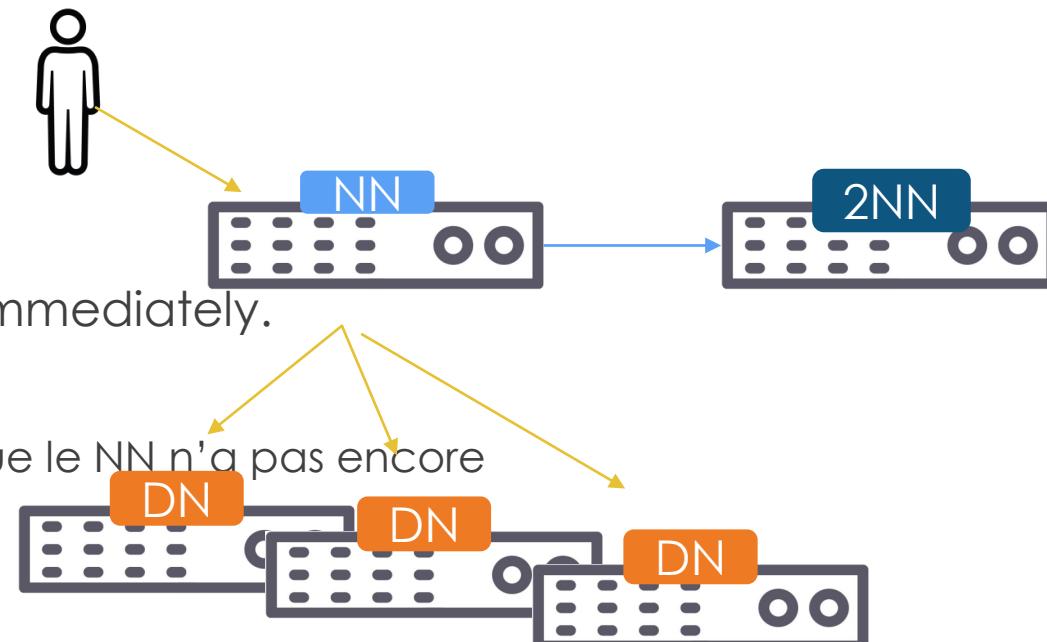
Secondary NameNode

- ▶ Contient une copie du FS-Image.
- ▶ NN crée l>Edit_Log.
 - ▶ L'envoie régulièrement au 2NN.
 - ▶ A chaque remplissage (taille spécifique) ou après un certain temps.
- ▶ 2NN update the FS_Image according to the received Edit_Log.
 - ▶ 2NN envoie le FS_Image updaté au NN.



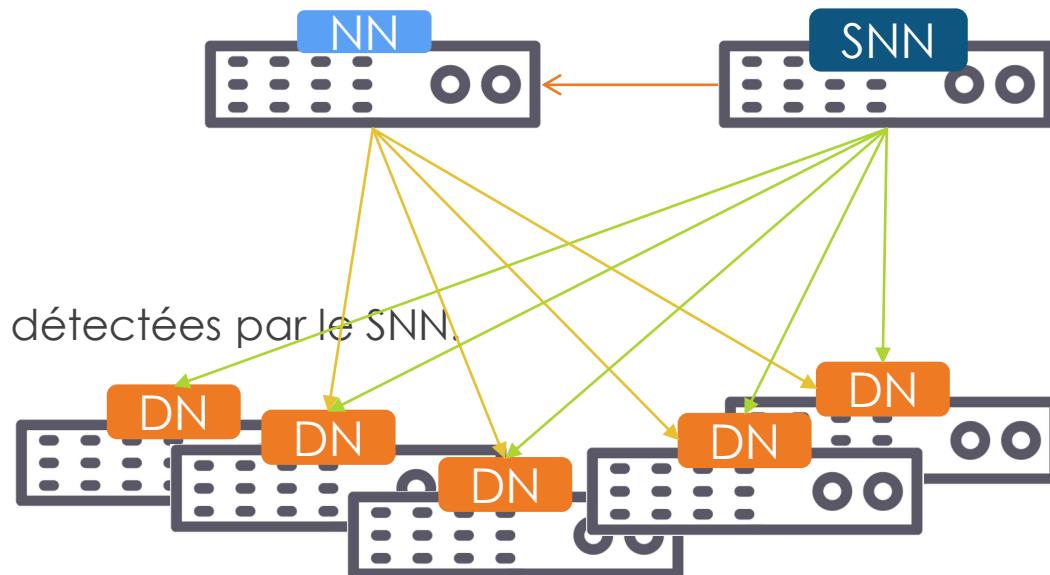
Outcome of 2NN solution

- ▶ Décharger le NN de l'update du FS_Image.
- ▶ Le NN peut continuer ses activités durant Le checking point au 2NN.
- ▶ Les données de la FS_Image que le NN procure
Ne sont pas très fraîches.
- ▶ At the NN failure, le 2NN is not able to take-over immediately.
 - ▶ Etablir la communication avec les DN.
 - ▶ Augmenter l'incohérence en raison de possibilité que le NN n'a pas encore
communiquer certaines requêtes au 2NN



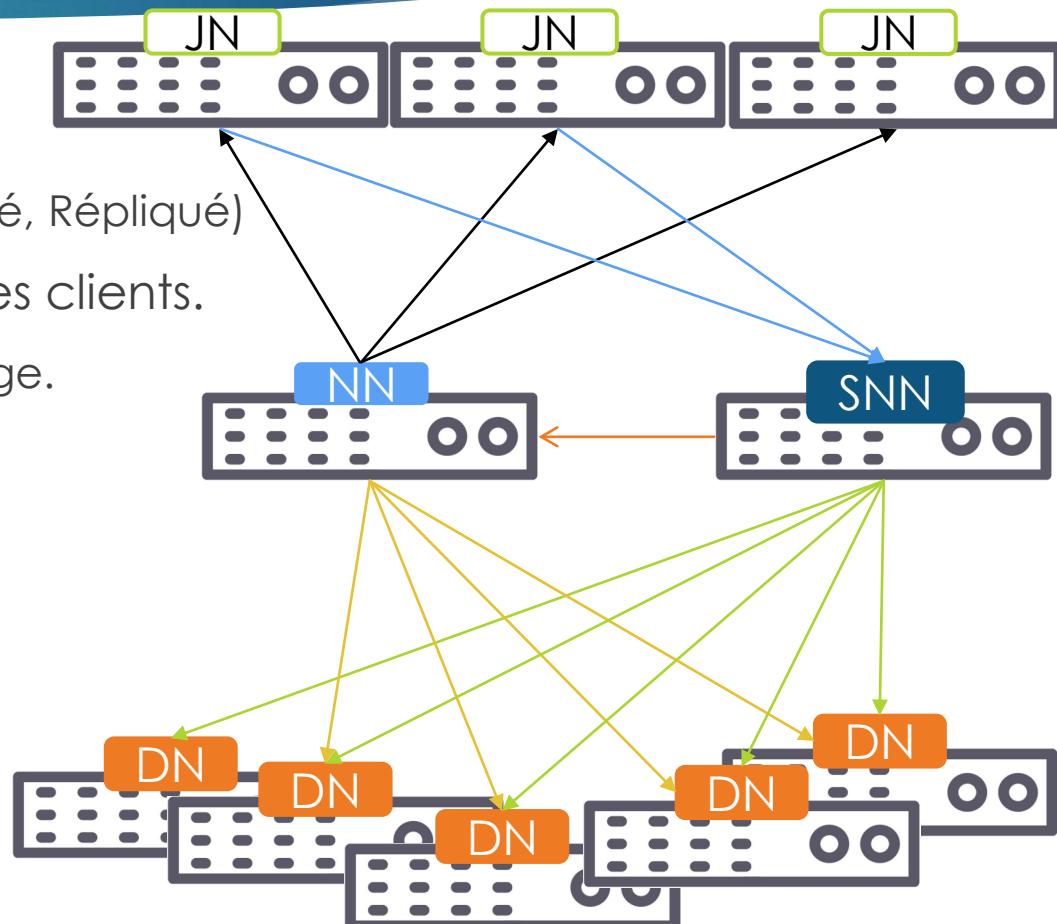
Stand-by Name Node : StdNN or SNN

- ▶ 2NN + Distribution de charge.
 - ▶ Very high Availability.
- ▶ Le NN contient la FS_Image.
 - ▶ SNN contient une copie du FS_Image.
- ▶ Ce n'est pas le NN qui envoie l>Edit_Log au SNN.
 - ▶ Le SNN a une visibilité sur les DN.
 - ▶ Toutes les modifications dans les DN les HB sont aussi détectées par le SNN.



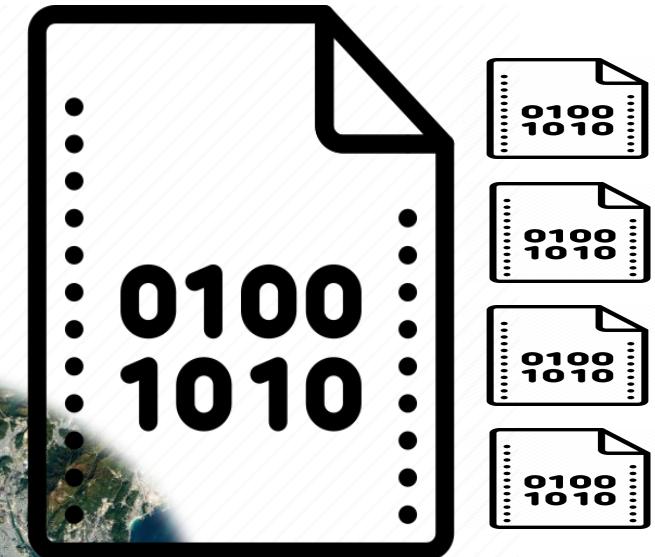
Stand-by Name Node : StdNN or SNN

- ▶ Journal Nodes (JN - par défaut 3) sont définies.
 - ▶ Servent à stocker, **exclusivement**, l>Edit_Log (Distribué, Répliqué)
- ▶ Toujours, le NN est le seul qui reçoit les requêtes des clients.
 - ▶ Le SNN assure l'update et l'envoi régulier du FS_Image.



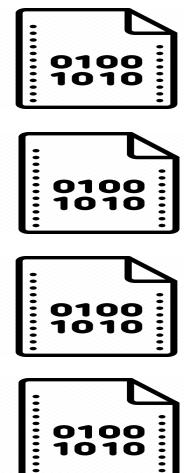
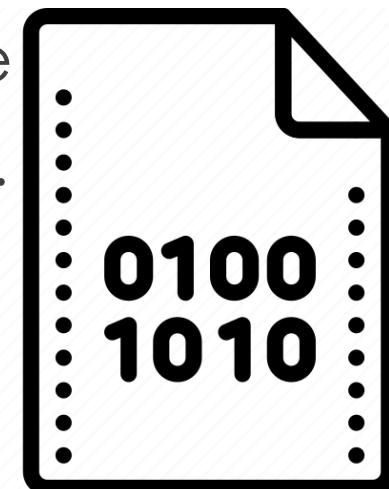
Map-Reduce

- ▶ Architecture de développement permettant de traiter des données volumineuses
 - ▶ De manière parallèle et distribuée
- ▶ Hadoop Streaming
 - ▶ Other languages than Java, could be used.

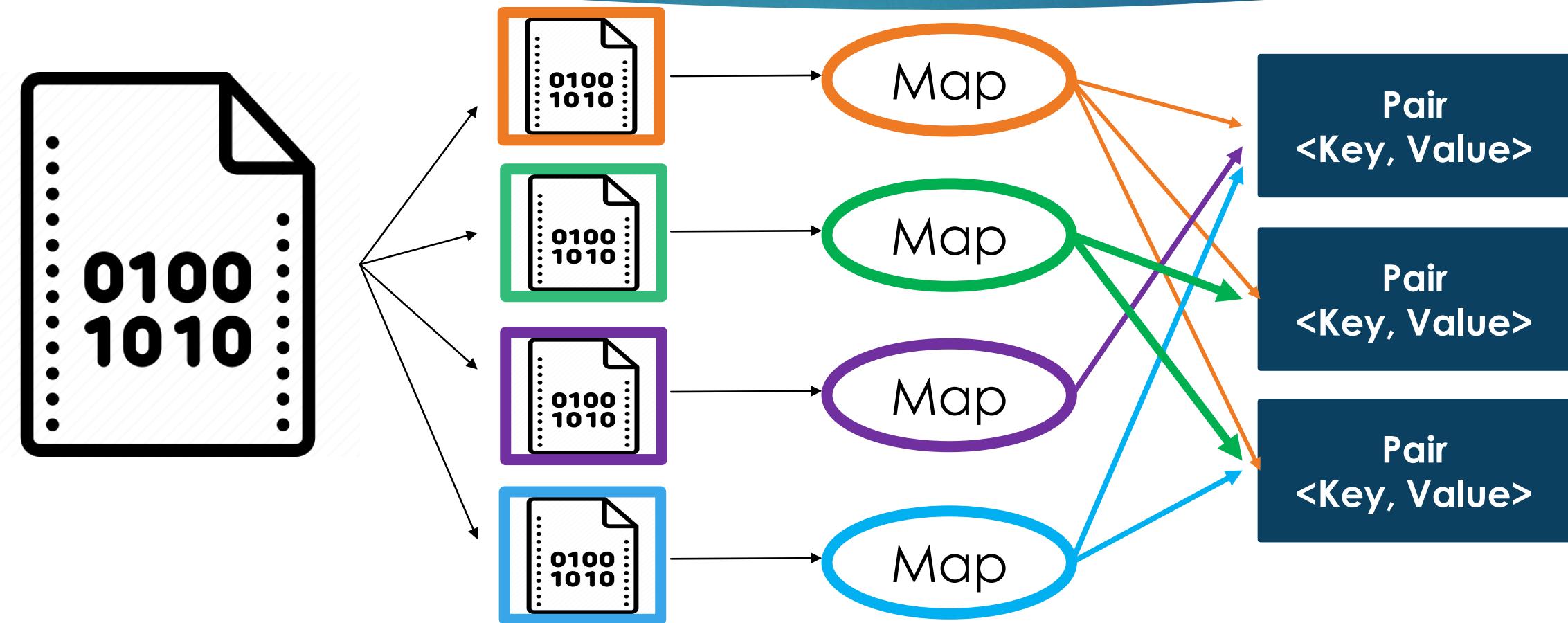


Map-Reduce

- ▶ Transforme les données d'entrée en série de couples clé/valeur.
- ▶ Les couples clés/valeurs doivent avoir un sens par rapport au problème à résoudre.
- ▶ Les données doivent être découpées en plusieurs fragments.
- ▶ l'opération Map est exécutée sur chaque machine du cluster sur un fragment distinct.



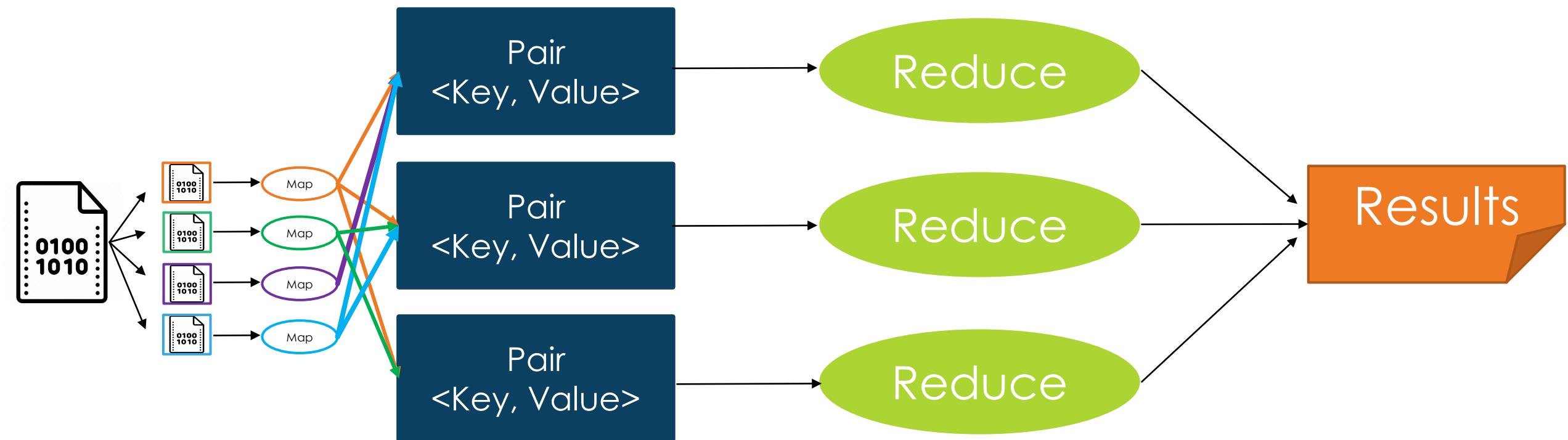
Map-Reduce



Reduce

- ▶ Applique un traitement à toutes les valeurs de chacune des clés distinctes produites par l'opération Map.
- ▶ Pour chaque machine du cluster est attribué une des clés uniques produites par l'opération Map, en lui donnant la liste des valeurs associées à cette clé.
- ▶ Chacune des machine effectue l'opération Reduce pour cette clé.

Reduce



Méthodologie

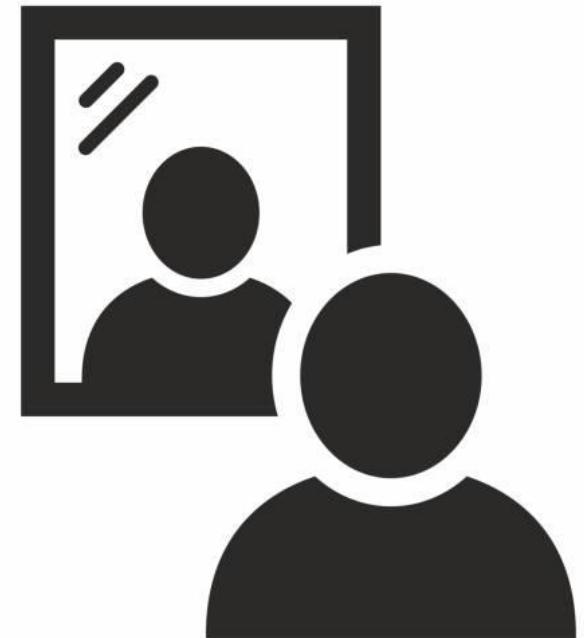
- ▶ Choisir une manière de découper les données d'entrée pour pouvoir paralléliser l'opération Map.
- ▶ Définir quelle clé utiliser pour le problème à résoudre.
- ▶ Écrire le programme pour l'opération Map.
- ▶ Écrire le programme pour l'opération Reduce.

Étapes de traitements

- ▶ Un traitement MapReduce est caractérisé par 4 étapes distinctes :
 - ▶ **Split** : Découper les données d'entrée en plusieurs fragments.
 - ▶ **Map** : Mapper chacun de ces fragments pour obtenir des couples (clé/valeur).
 - ▶ **Shuffle and Sort** : Grouper ces couples par clé et les trier.
 - ▶ **Reduce** : Réduire les groupes indexés par clé en une forme finale, avec un traitement pour chacune des clés distinctes.
- ▶ Chacune des tâches (à l'exception de la première) seront effectuées de manière distribuée.

Map-Reduce

- ▶ Le calcul distribué, groupement par Hadoop de manière transparente.



Exemple de WordCount

- ▶ Purpose
 - ▶ Dans le cadre d'une étude linguistique, nous désirons décompter le nombre d'occurrence des mots dans un script d'une conversation télévisée.

```
cat dog elephant dog bird  
catelephant dog cat bird bird bird  
cat dog cat dog elephant elephant  
elephant dog cat cat bird dog  
elephant elephant cat dog bird  
birdcat dog bird elephant elephant
```

WordCount : Etape 1.

- ▶ Étape 1 : Comment découper les données ?
- ▶ Séparation des données en blocs traitables séparément.
- ▶ Exemple de découpage : en lignes ou en n-uplets.

Input

cat dog elephant
dog bird
catelephant dog cat
bird bird bird bird cat
dog cat dog
elephant elephant
elephant dog cat
cat bird dog
elephant elephant
cat dog bird birdcat
dog bird elephant
elephant

Splitting

cat dog elephant
dog bird cat

elephant dog cat bird
bird bird bird cat

dog cat dog elephant
elephant elephant dog cat

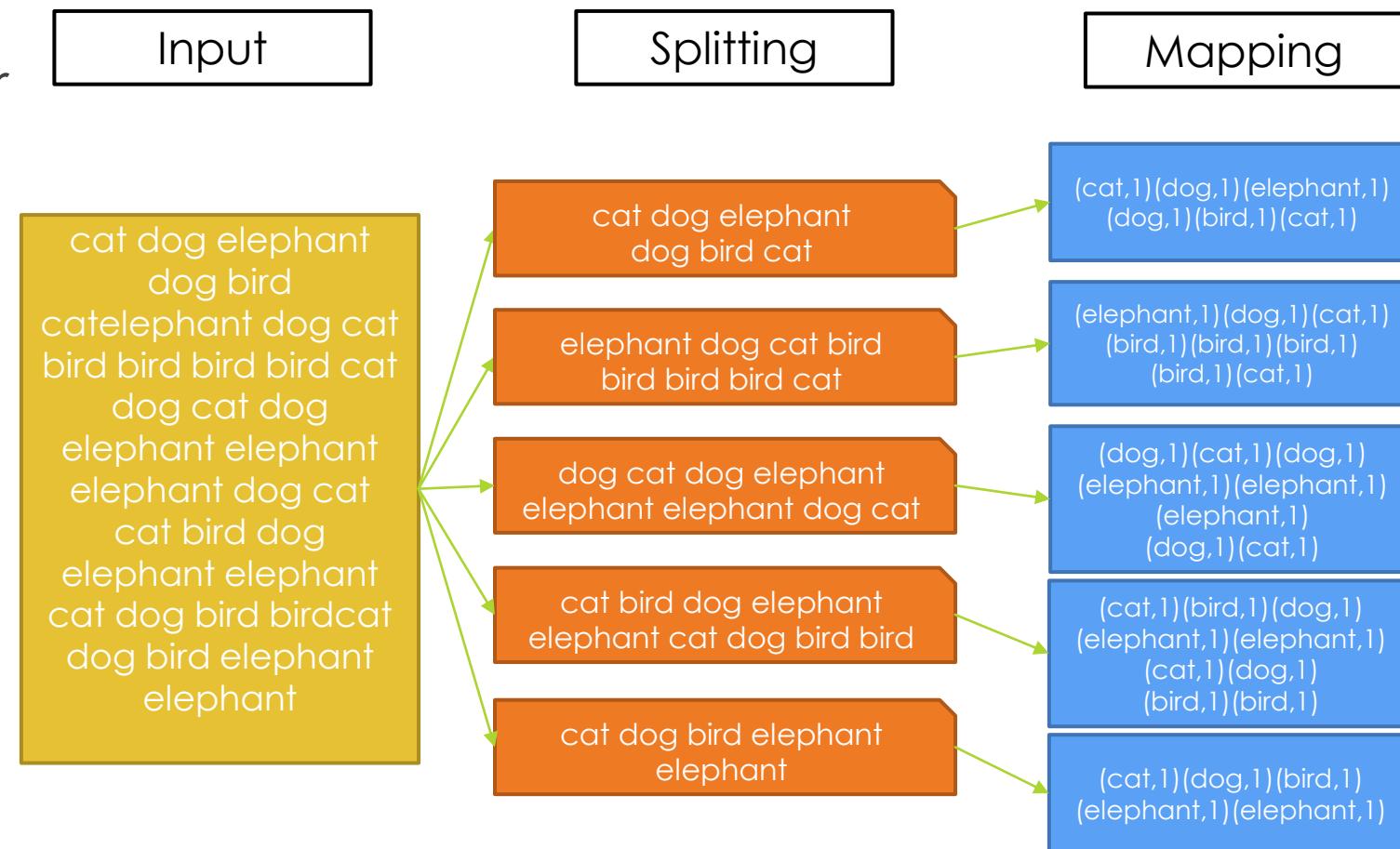
cat bird dog elephant
elephant cat dog bird bird

cat dog bird elephant elephant

WordCount : Etape 2.

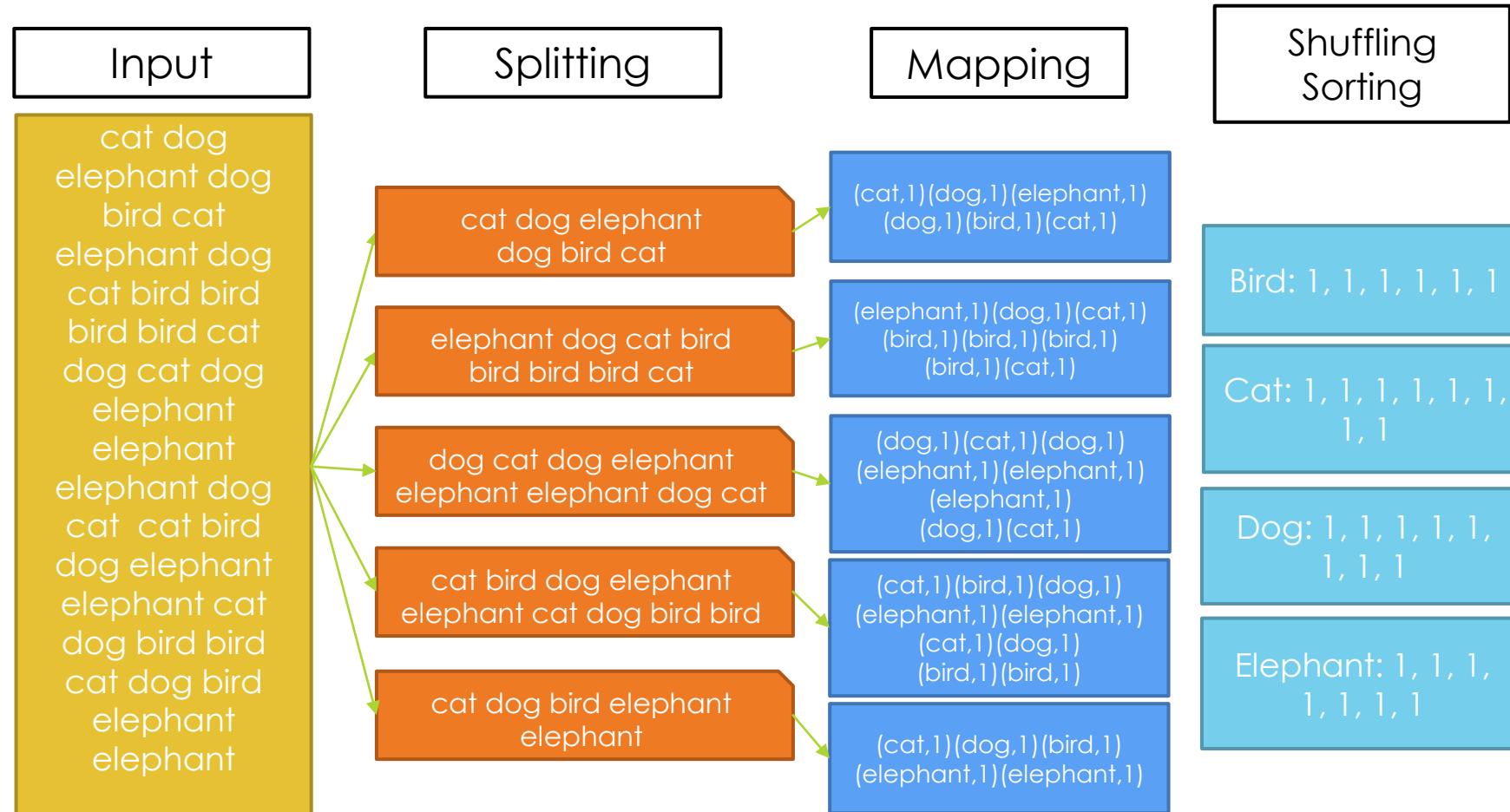
- Déterminer la clé et définir l'opération Map

- Clé : le mot lui-même.
- Opération Map : Générer le couple clé/valeur : (mot, 1)



WordCount : Etape 3.

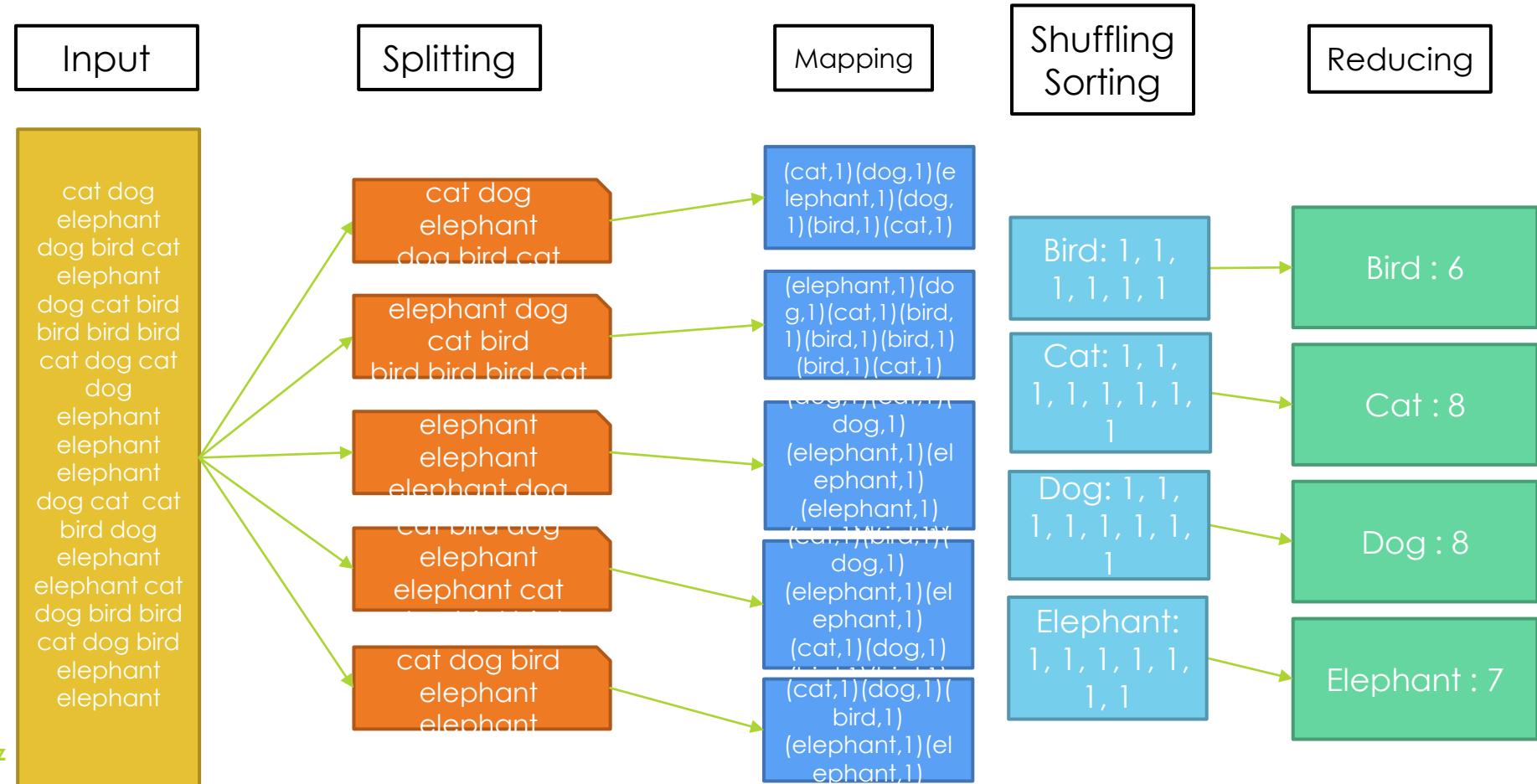
- ▶ Grouper tous les couples par clé commune
(Assurée par Hadoop)
- ▶ Cette opération est effectuée automatiquement et de manière distribuée par Hadoop.



WordCount : Etape 4.

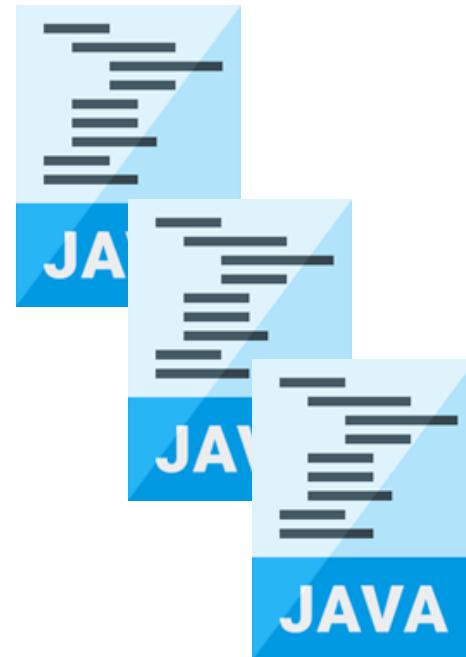
► Définir l'opération Reduce

- Elle sera appliquée sur chacun des groupes par clé distincte.
- Additionner toutes les valeurs liées à la clé spécifiée.



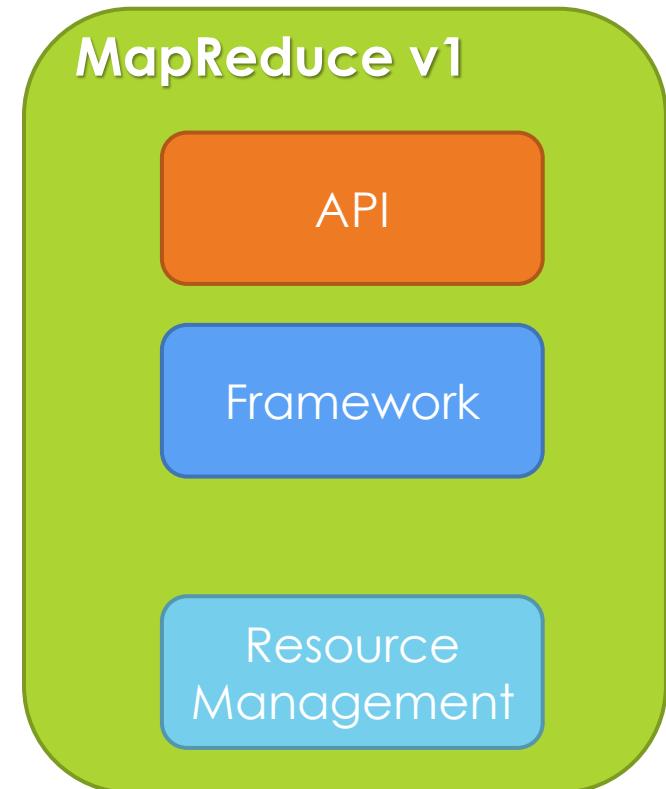
Map-Reduce, le paradigme

- ▶ Mapper Class
 - ▶ Fonctions de Mapping
 - ▶ Input : Fichier à traiter
 - ▶ Output : Liste de <Clé, Valeur>
- ▶ Reducer Class
 - ▶ Fonctions de Réduction
 - ▶ Input : le Output du Mapper : Liste de <Clé, Valeur>
 - ▶ Output : Une nouvelle Liste de <Clé, Valeur>, Réduite.
- ▶ Main Class
 - ▶ Configuration et Définition du Job.
 - ▶ Instanciation des classes de Raffinement du Job.
 - ▶ Définition des Input/Output du Job et leurs emplacements.



MapReduce V1 (MR v1)

- ▶ MapReduce V1 intègre trois composants :
 - ▶ API
 - ▶ Pour permettre au développeur l'écriture d'application MapReduce
 - ▶ Framework
 - ▶ Services permettant l'exécution des Jobs MapReduce, le Shuffle, Sort, ...
 - ▶ **Resource Management**
 - ▶ Infrastructure pour gérer les nœuds du cluster, allouer des ressources et ordonner les jobs.



MapReduce V1 (MR v1)

► JobTracker

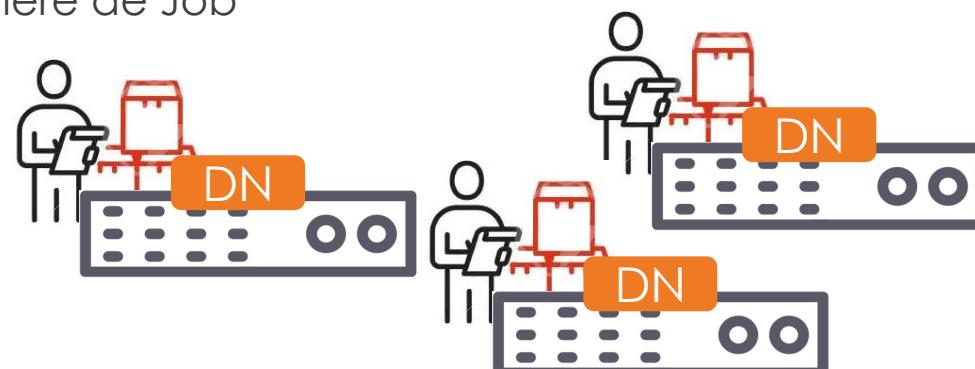


- Distribuer l'activité entre les différents DN
- Les DN représentent les mappers et les reducers en matière de Job

► TaskTracker



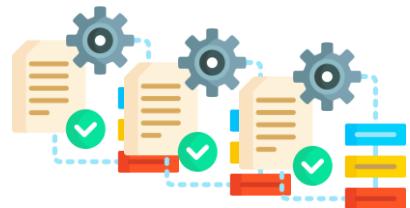
- Processus qui s'exécute sur chacun des DN.
- Colocalisé avec les DN.
- Effectuer le PayloadData Handling (calcul et parcours des fichiers de données).



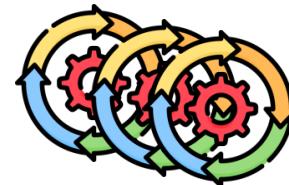
MapReduce V1 (MR v1)

- ▶ Job Map-Reduce
 - ▶ Application Map-Reduce
 - ▶ Fractionné en plusieurs tâches :

- ▶ Map



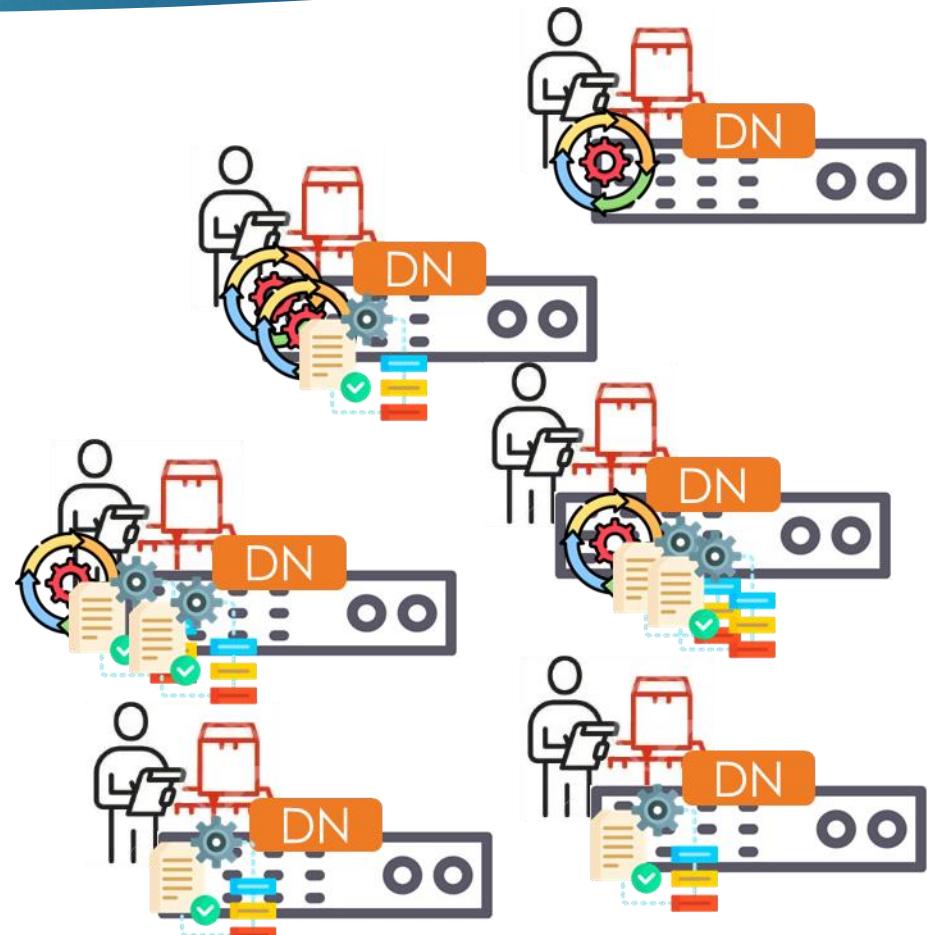
- ▶ Reduce



- ▶ Chaque tâche est exécuté sur un nœud du cluster.
 - ▶ Là où se trouvent les blocs des données concernées.

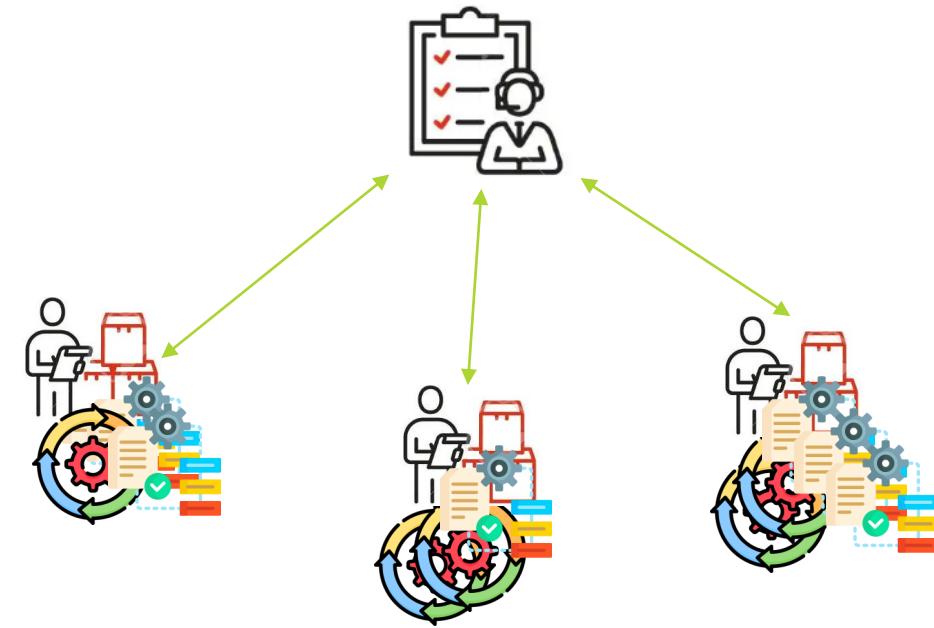
Principe des Slots dans MR-V1

- ▶ Un nombre de slots prédéfini sur le cluster.
- ▶ Slot = Unité de ressources que le nœud dispose.
 - ▶ Capacité du TaskTracker à exécuter une tâche.
- ▶ Slots prédéfinis dans le fichier de conf.
- ▶ Chaque nœud a un nombre de slots prédéfinis
 - ▶ Map Slots
 - ▶ Reduce Slots



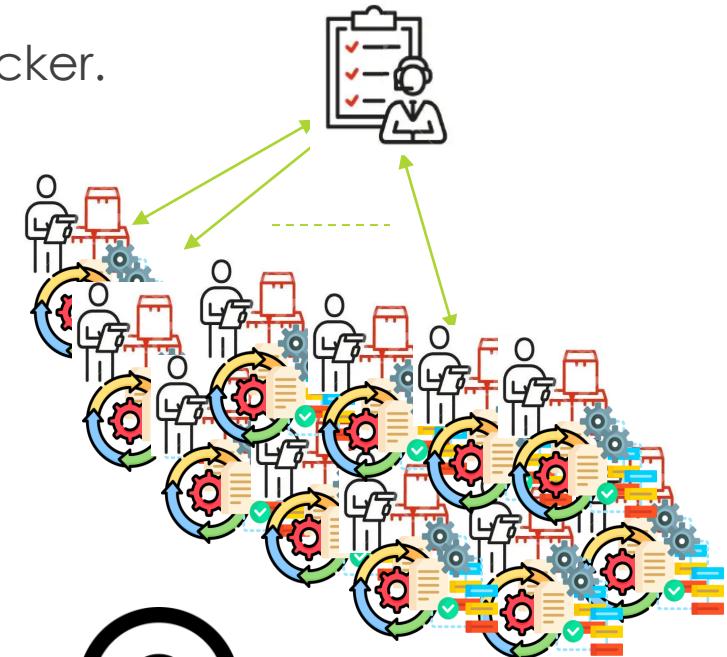
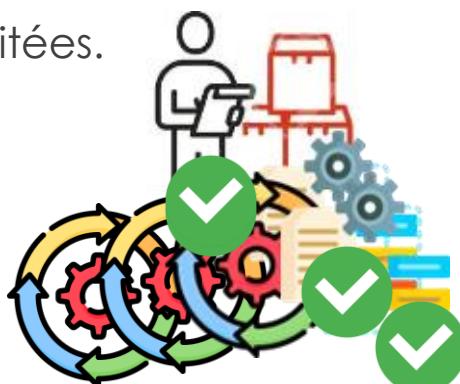
Coordination en MRv1

- ▶ JobTracker
 - ▶ A travers les TaskTracker se charge de :
 - ▶ L'allocations des Slots.
 - ▶ Gérer les ré-allocation suite aux échecs.
 - ▶ Coordonner l'achèvement des Jobs Map-Reduce.

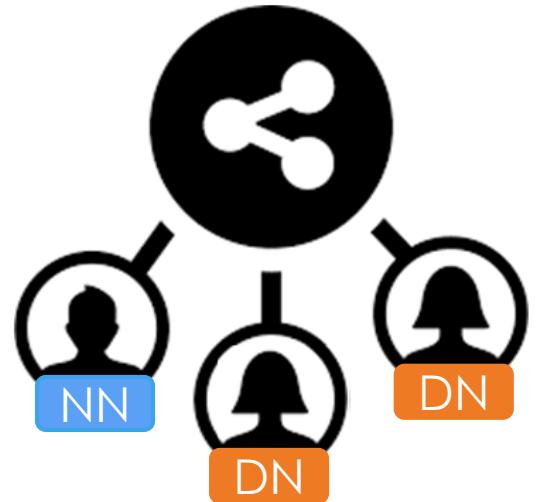
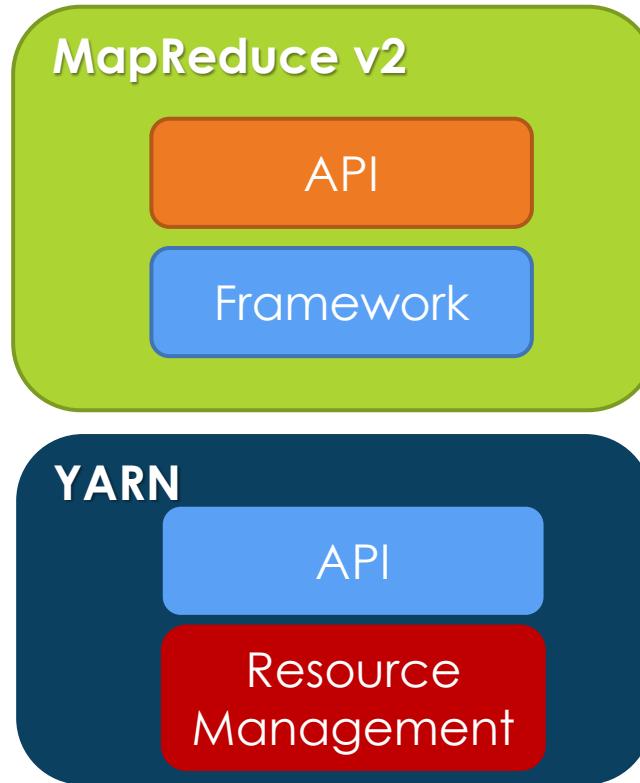
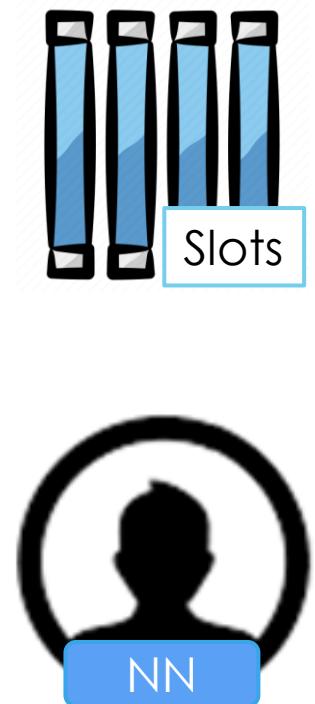
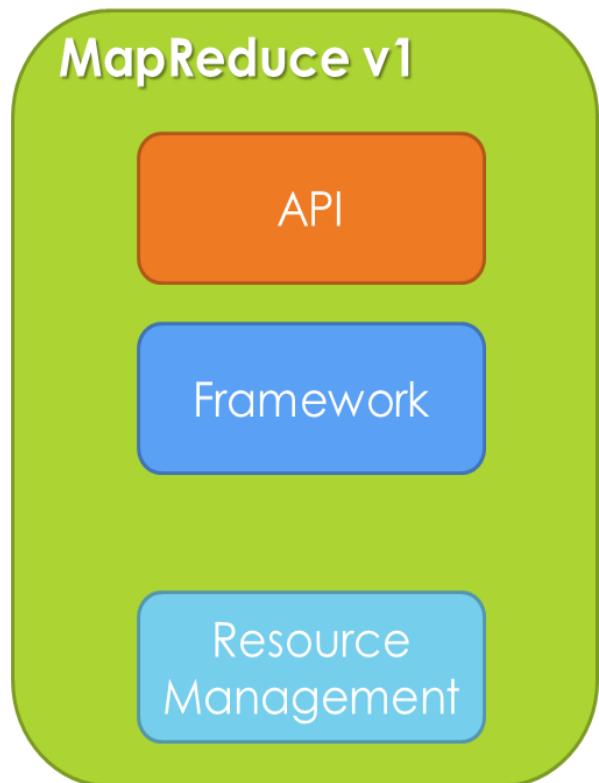


Problématiques

- ▶ Uniquement le NN qui héberge et assure l'exécution du JobTracker.
 - ▶ La gestion de ressources, une activité de plus.
 - ▶ Activité très lourde par le temps.
 - ▶ **Single point of failure.**
 - ▶ Soucis d'Interopérabilité d'HDFS.
- ▶ Les machines du cluster en occurrence les DN
 - ▶ N'assurent que les tâches Map et Reduce.
 - ▶ Nombreuses d'entre elles sont inexploitées.
 - ▶ Up to 4000 machines.
- ▶ Soucis de gestion des slots prédéfinis.



MapReduce V2 (MR v2)



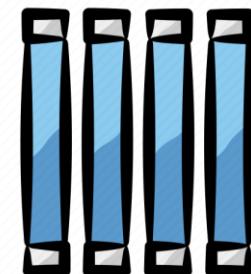
MapReduce V2 (MR v2)

- ▶ Resource Manager



- ▶ Node Manager

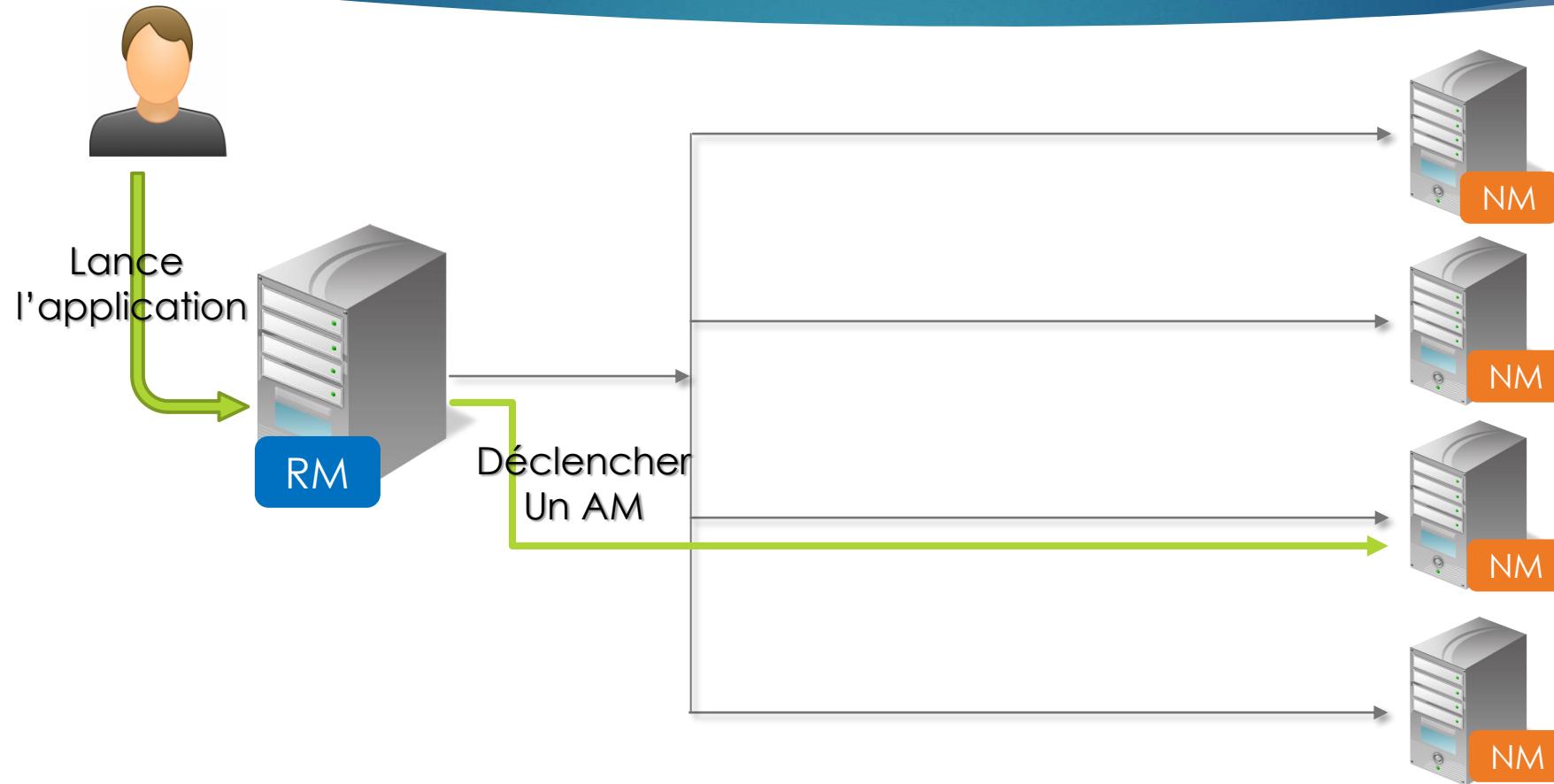
- ▶ Containers



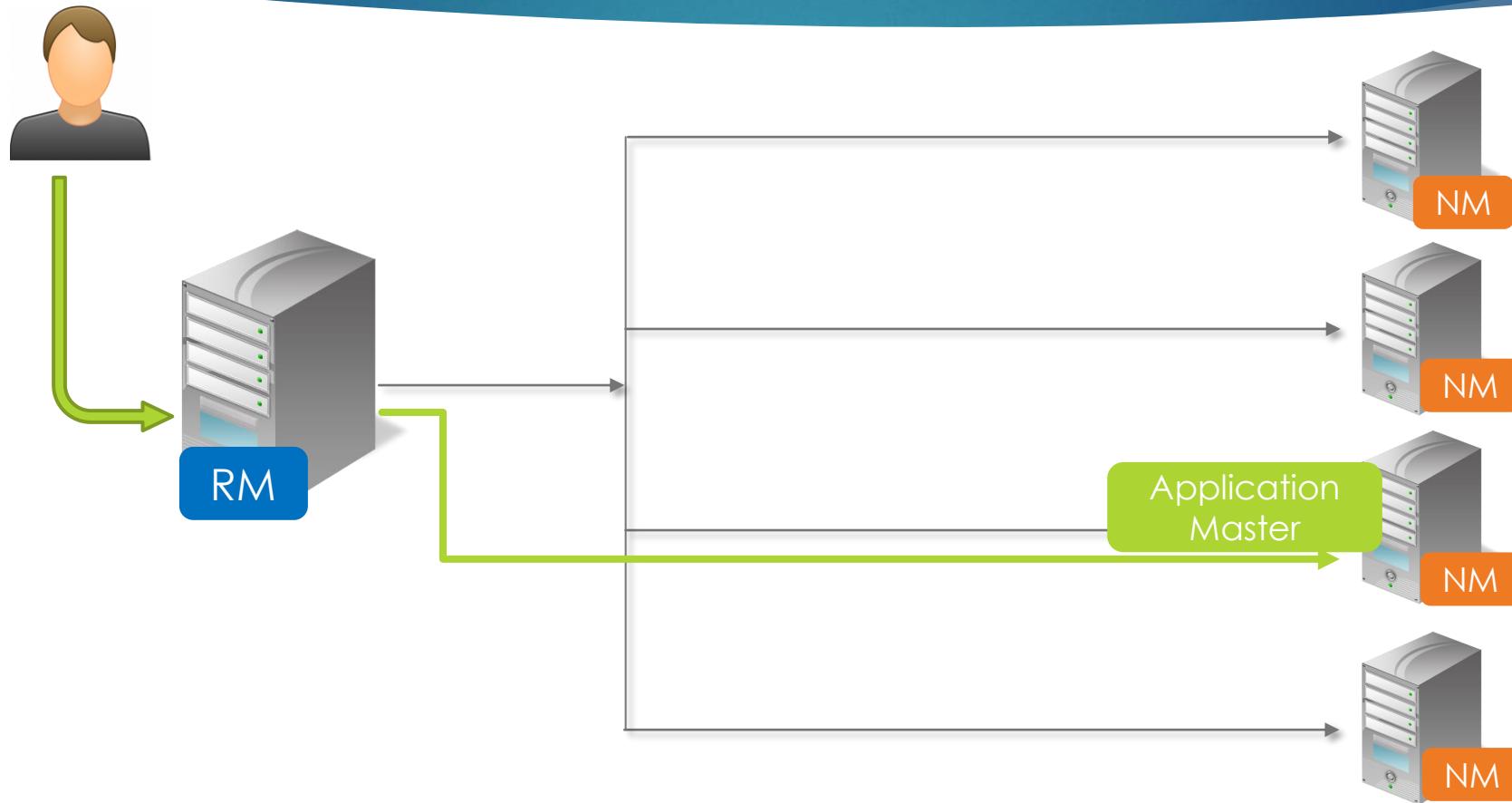
- ▶ Application Master



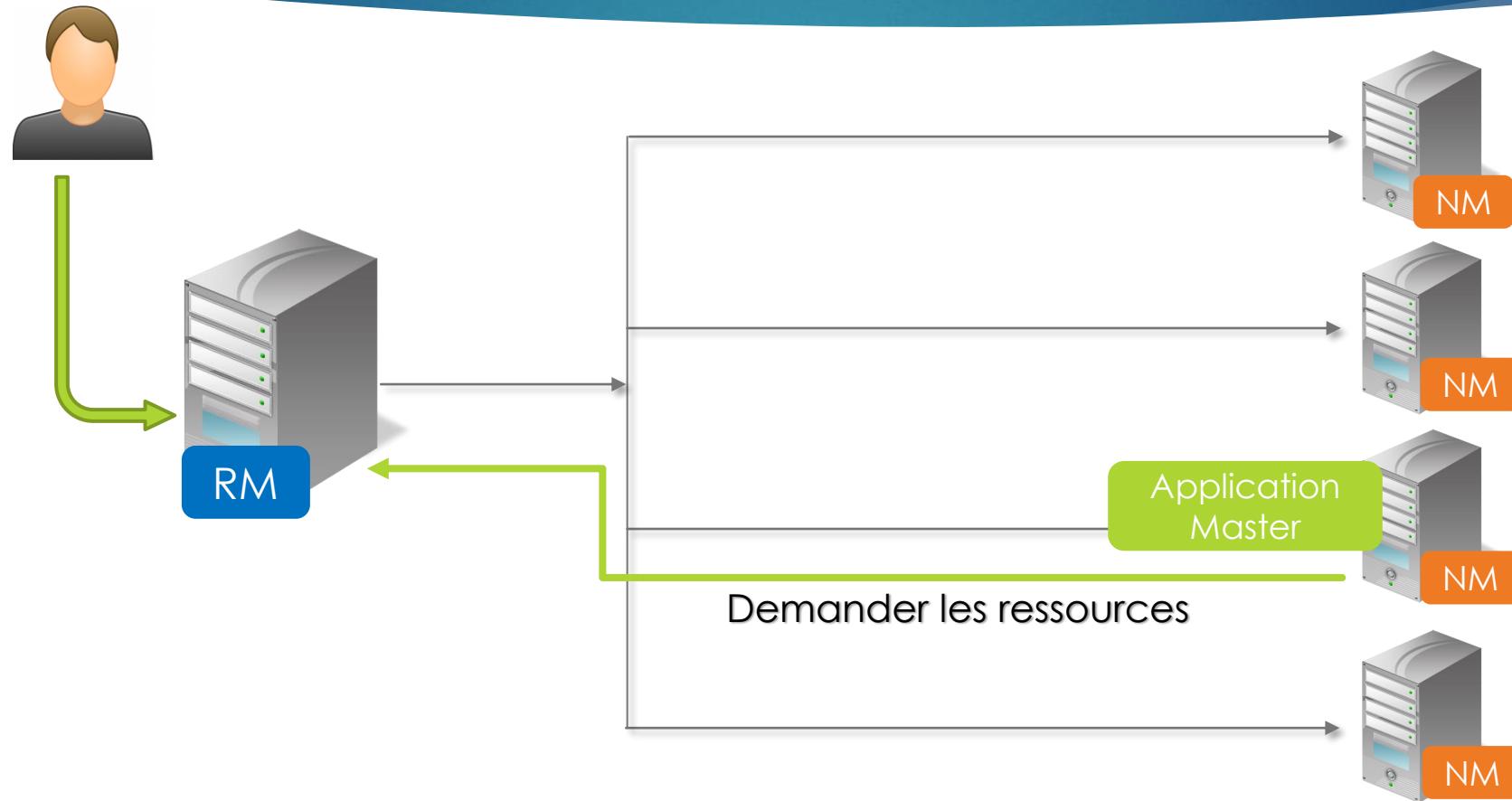
Fonctionnement en MRv2



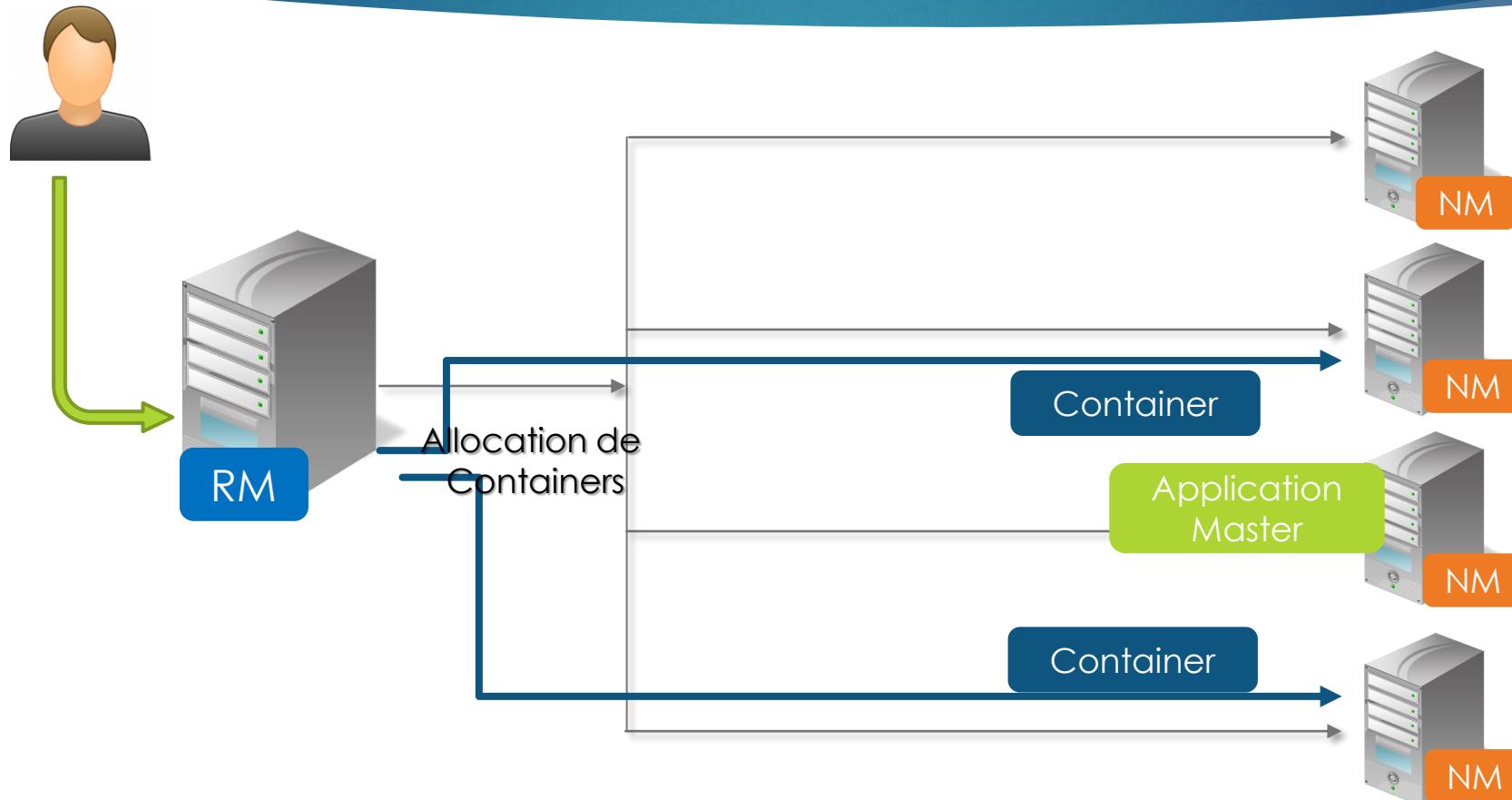
Fonctionnement en MRv2



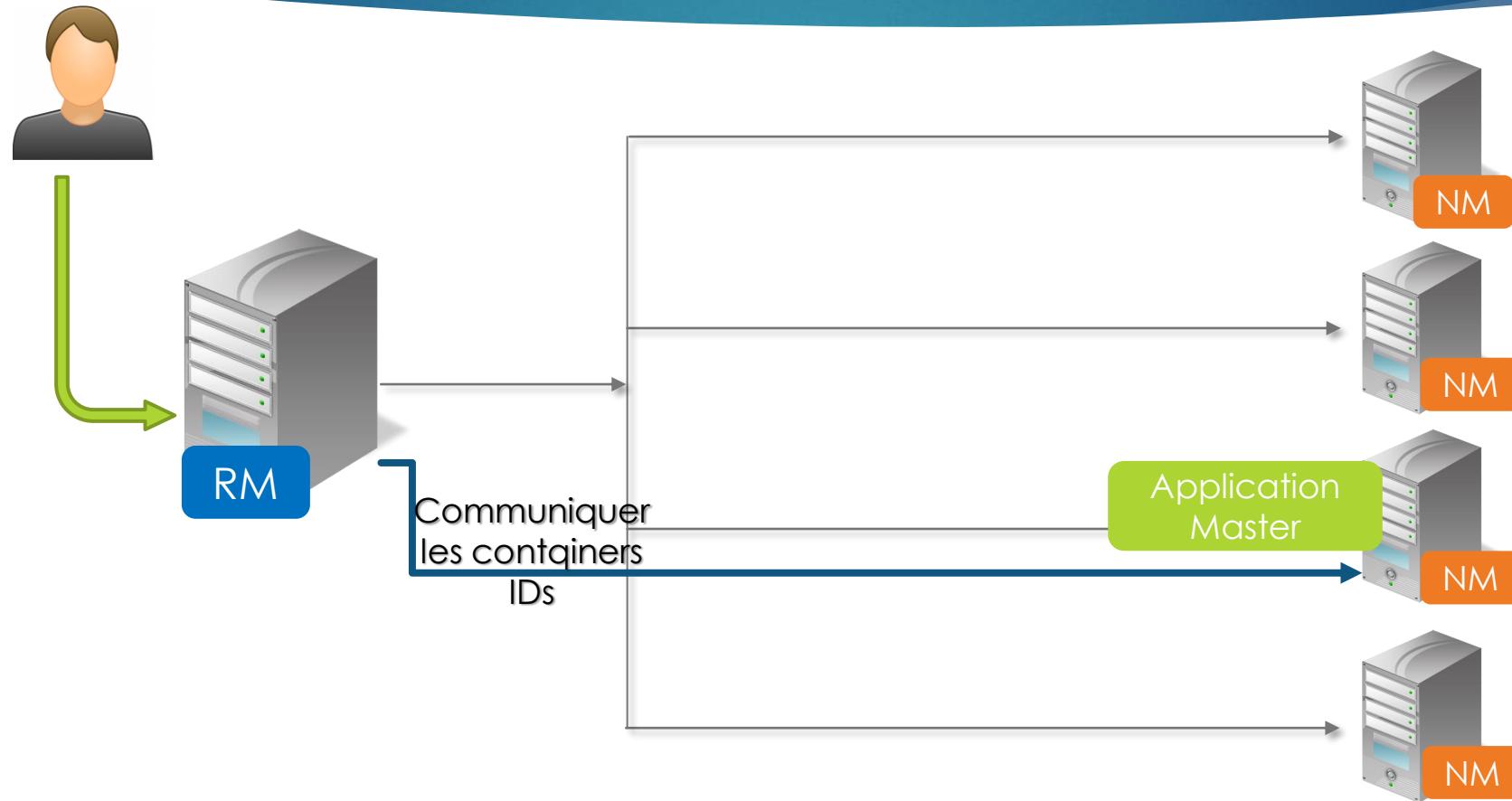
Fonctionnement en MRv2



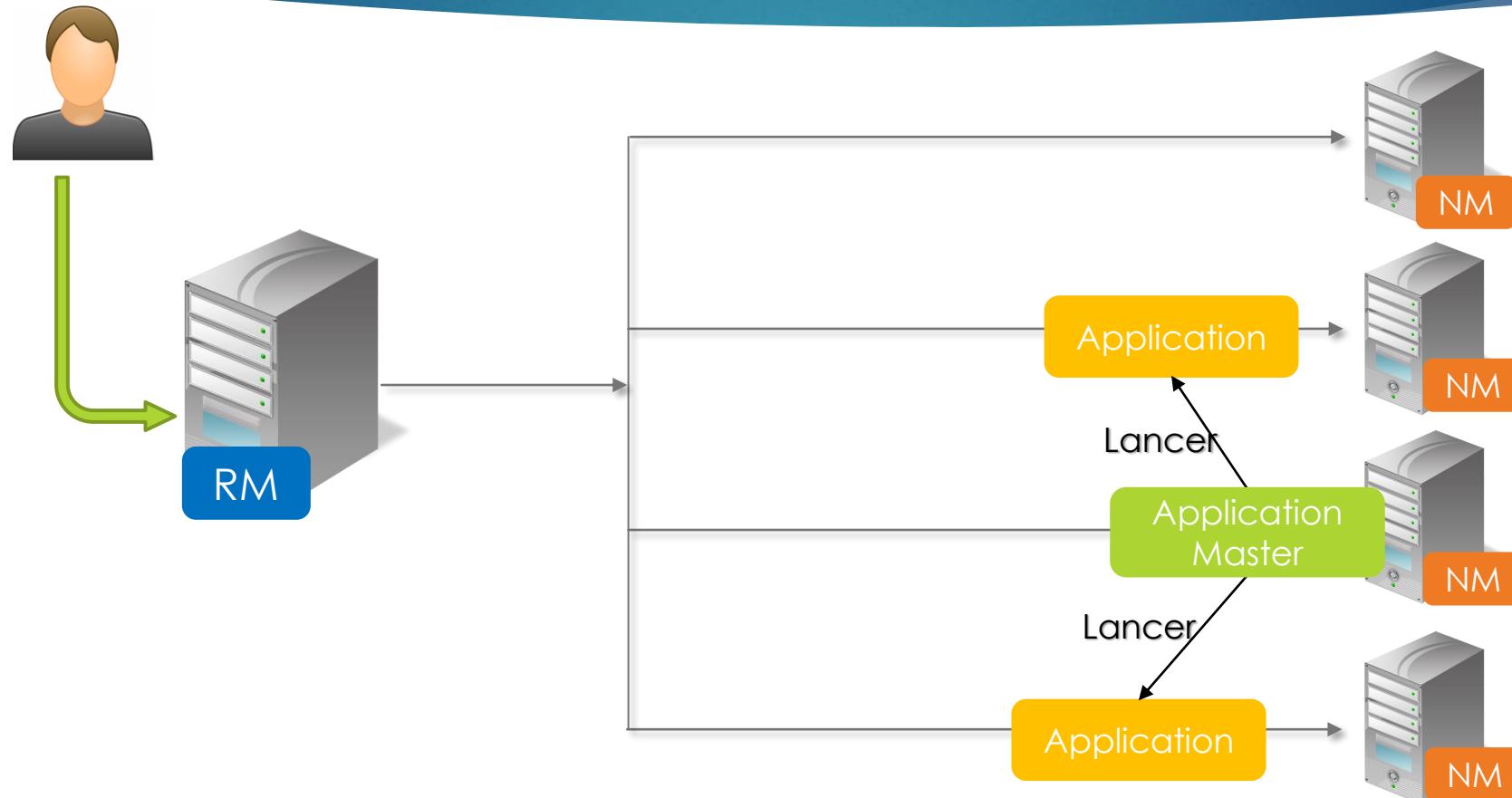
Fonctionnement en MRv2



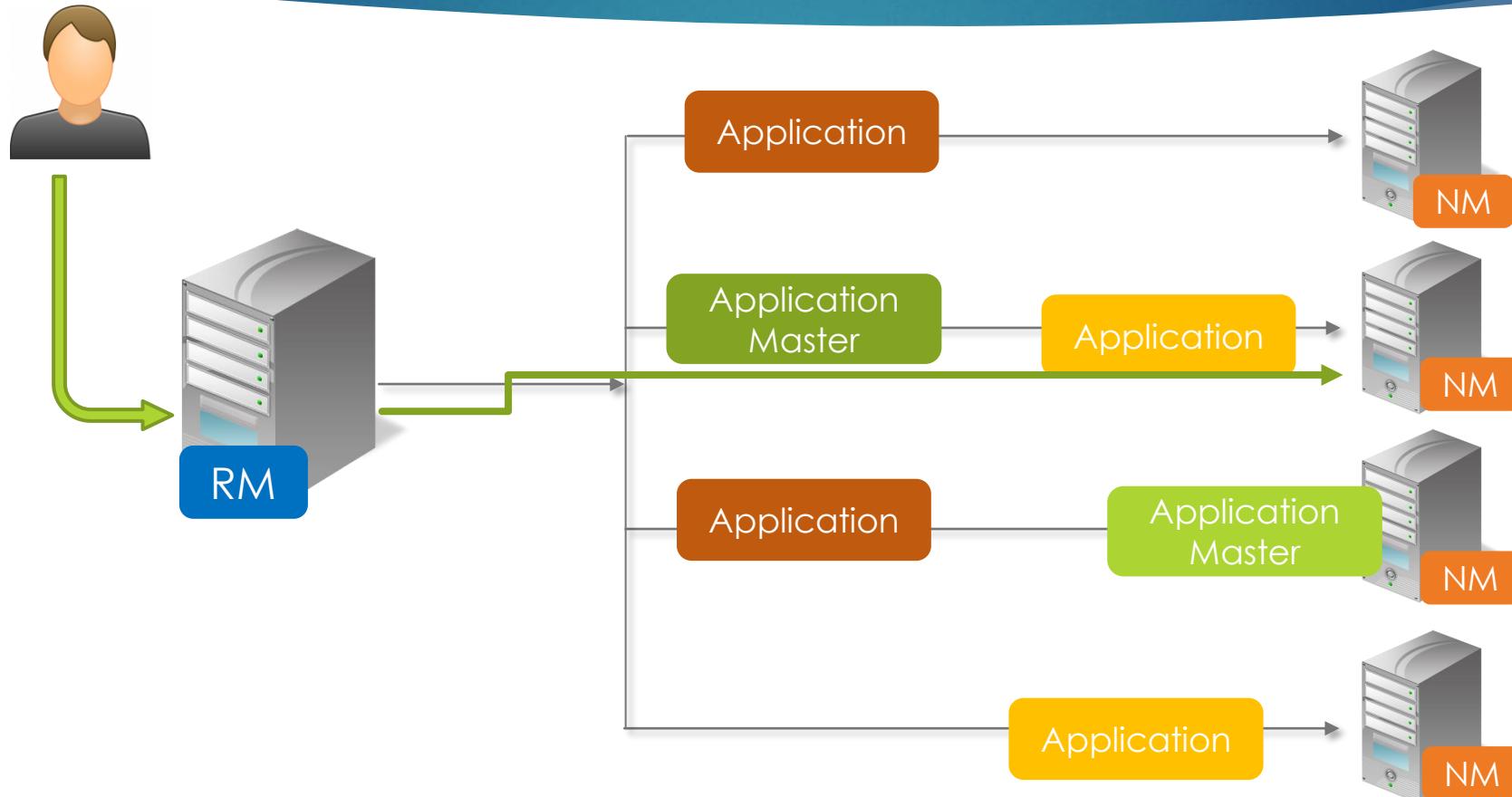
Fonctionnement en MRv2



Fonctionnement en MRv2

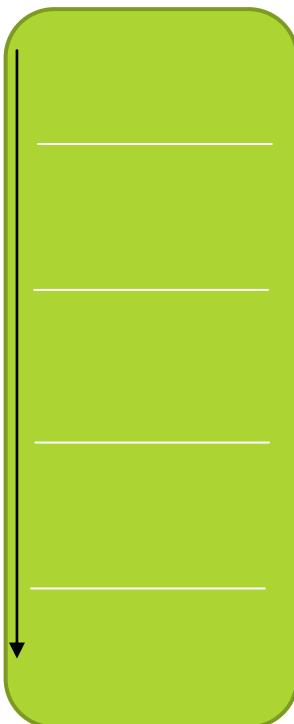


Fonctionnement en MRv2



Types de Traitement

- ▶ Batch Processing
 - ▶ Traitement par Lot
 - ▶ Parcourir l'ensemble de la données afin de réaliser le traitement.
 - ▶ Même en parallèle comme en Map.
- ▶ Difficilement Interactif (Latence très élevée).



Types de Traitement

► Streaming

- Traitement à l'arrivée.
- Ne pas attendre l'achèvement de la données complète.



- Le cas des données ne sont disponibles immédiatement.
- La vitesse de traitement doit être plus importante qu'au débit d'arrivée du flux de données.
- Batch Processing or Streaming : Dépendamment du type de la source de données.

Types de Traitement

- ▶ La vitesse de traitement doit être plus importante qu'au débit d'arrivée du flux de données.

Solutions:

- ▶ **Systèmes d'ingestion de données** : Plusieurs files d'attente et de traitement.
 - ▶ Solution Kafka (limiter les dégâts).



- ▶ **Micro-Batching** (Spark Streaming)

- ▶ Regrouper des données et les traiter d'une façon agrégée.
- ▶ Regroupement according a defined time (généralement 2 secondes).



Architectures Big Data

- ▶ Architecture Lambda
 - ▶ Batch Layer
 - ▶ Speed Layer
 - ▶ Serving Layer
- ▶ Crédit de lien entre Batch et Speed Layers (Training Layer)

Architectures Big Data

- ▶ Architecture Kappa (Implémentation plus difficile)
 - ▶ Conserver uniquement la Speed Layer avec le Speed View.
 - ▶ Accès à un Data Lake (avoir un traitement en Batch) a partir de la Speed Layer (Stream)
 - ▶ Sachant que la Speed Layer qui alimente le DataLake pour un éventuelle traitement par lot.
 - ▶ Sans que le EndUser puisse avoir les résultats du Batch processing.
 - ▶ Le EndUser à un accès uniquement sur les résultats de Streaming.
 - ▶ Les opération de streaming ont accès aux résultat du Batch Processing