Chapitre III

Stockage de données en nuage

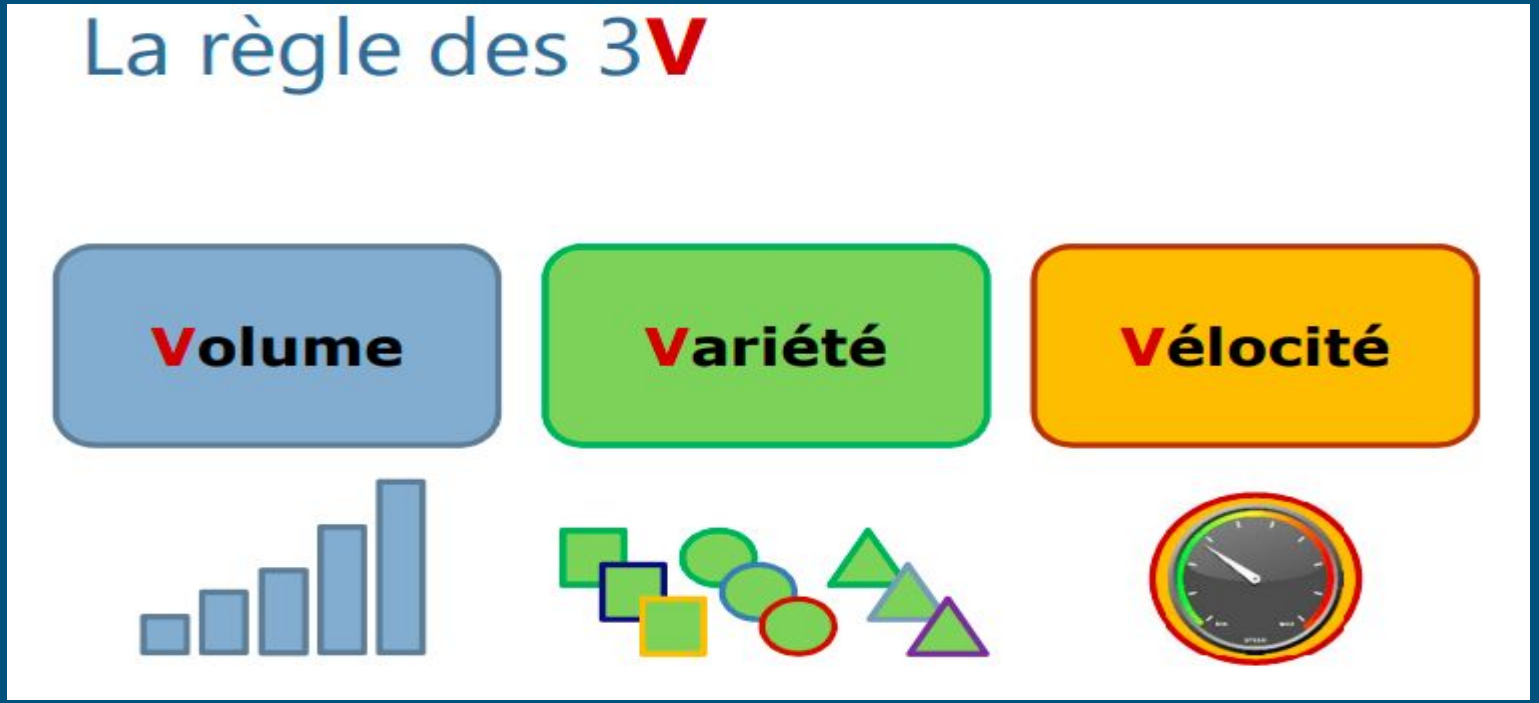
I. Data-intensive computing

- Le Data-intensive computing (ou informatique à haute performance orientée données) est une branche de l'informatique qui se concentre sur le traitement et l'analyse de grandes quantités de données. Cette approche vise à exploiter des ressources informatiques à grande échelle pour traiter et analyser des ensembles de données volumineux, variés et complexes.
- Le Big Data c'est quoi ? (Rappel)
  Le Big Data désigne **un très grand volume de données** souvent hétérogènes qui ont **plusieurs formes et formats** (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.), et comprenant des formats hétérogènes : **données structurées, non structurées et semi-structurées**

I. Data-intensive computing

- Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés pour son **traitement** et son **stockage**. Ainsi, il ne peut être traité en utilisant des outils tels que les SGBD traditionnels. La plupart des scientifiques et experts des données définissent le Big Data avec **le concept des 3V.**

I. Data-intensive computing



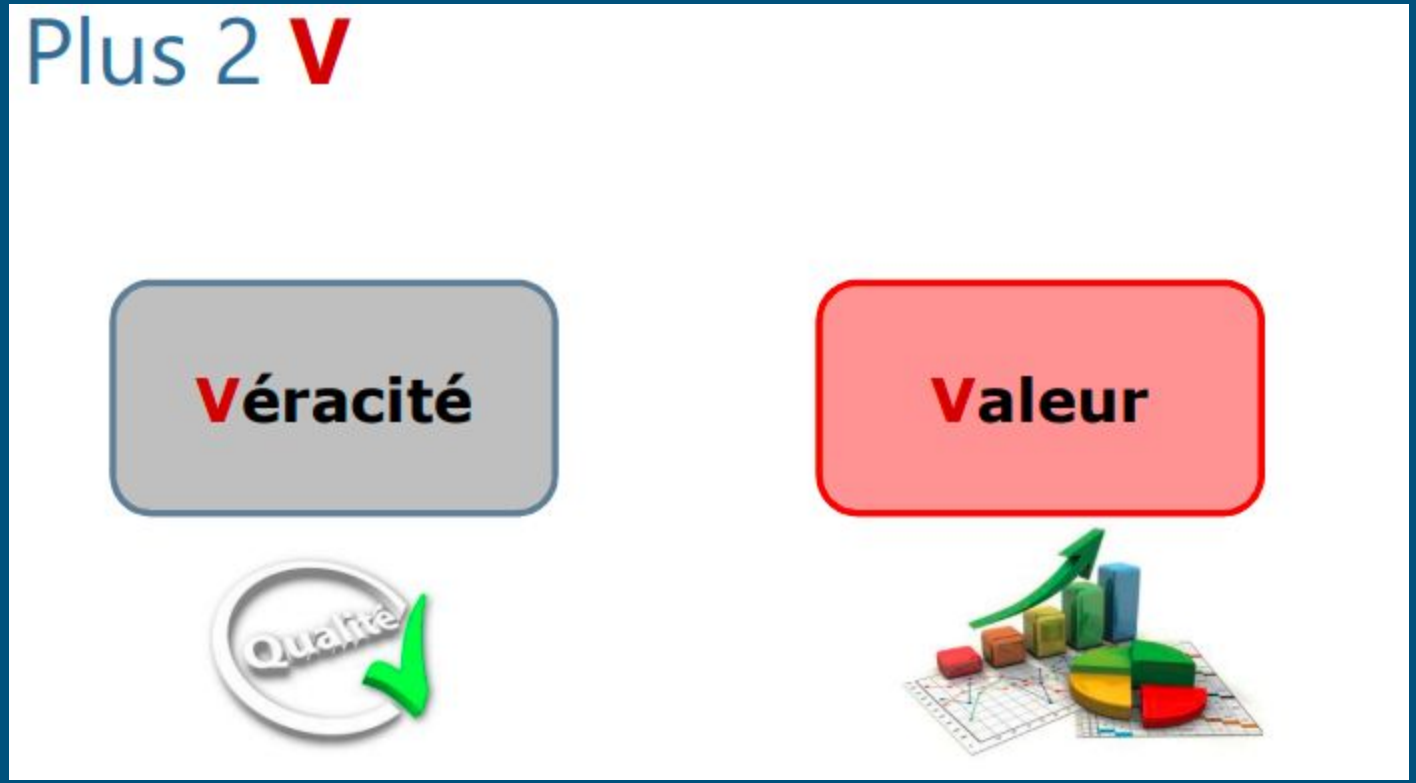
I. Data-intensive computing

- **Volume** : il représente l**a quantité de données générées**, stockées et exploitées.
- **Variété** : Les données volumineuses sont générées à partir de diverses sources distribuées dans **plusieurs formats** (vidéos, documents, commentaires, journaux,...). Les grands ensembles de données comprennent des données **structurées** et **non structurées,** publiques ou privées, locales ou distantes, partagées ou confidentielles, **complètes** ou **incomplètes**, etc

I. Data-intensive computing

- **Vélocité :** Les données sont **générées rapidement** et **doivent être traitées rapidement** pour extraire des informations utiles et des informations pertinentes. Par exemple, Wallmart (une chaîne internationale de détaillants à prix réduits) génère plus de 2,5 petabyte(PB) de données toutes les heures à partir des transactions de ses clients. YouTube est un autre bon exemple qui illustre la vitesse rapide du Big Data.

I. Data-intensive computing



I. Data-intensive computing

- **Véracité :** La véracité (ou validité) des données correspond à la fiabilité et l'exactitude des données, et la confiance que ces Big Data inspirent aux décideurs. Si les utilisateurs de ces données doutent de leur qualité ou de leur pertinence, il devient difficile d'y investir davantage.
- **Valeur** : Ce dernier V joue un rôle primordial dans les Big Data, la démarche Big Data n'a de sens que pour atteindre des objectifs stratégiques de création de valeur pour les clients et pour les entreprises dans tous les domaines.

I. Data-intensive computing

- Le Data-intensive computing est souvent utilisé pour résoudre des problèmes de traitement de données tels que l'analyse de données massives, la fouille de données, l'apprentissage automatique, la simulation numérique et la modélisation. Les applications courantes incluent la recherche scientifique, l'analyse financière, la surveillance de la santé publique, la publicité en ligne, la recherche Web et les réseaux sociaux
- Les technologies clés utilisées dans le Data-intensive computing incluent les systèmes distribués, les bases de données distribuées, les systèmes de fichiers distribués, les architectures de calcul haute performance et les frameworks de traitement de données tels que Hadoop, Spark, Flink, . . .

II. Map Reduce pour Big Data

- Maîtriser les données est un des grands **défis** sociétaux *(Un monde maîtrisant ses données de bout en bout, détiendrait un atout majeur pour relever les nombreux défis humains, environnementaux et économiques fixés par l'ONU pour l'horizon 2030)* avec l'objectif de disposer d'une donnée plus accessible, propre, intelligible, etc.. La donnée est considérée comme une matière première indispensable pour prendre de meilleures décisions à tous les niveaux de la société
- Le traitement des mégadonnées (à large échelle) est un ensemble de techniques ou de modèles de programmation permettant d'accéder à des données à grande échelle afin d'extraire des informations utiles pour soutenir et fournir des décisions.

II. Map Reduce pour Big Data

- Les contraintes **5V** rendent plus difficile la gestion des méga-données
- Par ailleurs, les infrastructures informatiques ont fortement évolué avec l'essor du cloud computing (Une grappe de machines appelée cluster de calcul).
- Il est devenu possible de louer rapidement un cluster servant d'infrastructure pour gérer des méga-données. Ainsi des solutions logicielles parallèles et distribuées sont conçues pour offrir des solutions de gestion de données

II. Map Reduce pour Big Data

La gestion de méga-données, au moyen d'une infrastructure distribuée de type cluster ou fédération de machines, soulève de nombreux défis:

1. Défi du passage à l'échelle

- Ce défi est tout d'abord posé par la contrainte de volume. Il s'agit de garantir que la solution de gestion de données continue de fonctionner lorsque la quantité devient très élevée.
- De plus, la contrainte de vélocité rend plus difficile le passage à l'échelle car l'adaptation doit être dynamique lorsque le volume des données fluctue.

II. Map Reduce pour Big Data

→ Cela nécessite de concevoir une solution dite élastique, il s'agit de concevoir des algorithmes pour contrôler la distribution des données et des traitements.

- Distribuer les données sur plusieurs machines et adapter cette distribution en fonction du volume des données et du nombre de machines, et de la capacité de stockage des machines.
- Distribuer les traitements entre les machines. Pour faciliter le passage à l' échelle, les algorithmes doivent être décentralisés ce qui permet à chaque machine de traiter des données indépendamment. Cela nécessite également de coordonner les traitements des différentes machines pour être capable de traiter des requêtes plus complexes car accédant aux données de plusieurs machines.

II. Map Reduce pour Big Data

2. Défi de la représentation des données

- Ce défi est posé par les contraintes de diversité et de complexité des données.
- Les données peuvent provenir de contextes très divers.
- Il n’y a pas de formalisme général pour « injecter » automatiquement les informations du contexte dans un système de gestion de données.
- Le défi est de définir ou compléter la représentation des données avec des informations spécifiques issues du contexte des données

II. Map Reduce pour Big Data

- Le Big Data est un environnement de collecte, de **stockage** et de **traitement** des données.
- **Ce terme cache** ainsi un **écosystème technologique riche** qui **comprend** tous les éléments nécessaires **(Distributed File System, Map Reduce, …)** à la constitution d'un socle de données complet et évolutif pour assurer la logistique des données à l'échelle de l'entreprise

II. Map Reduce pour Big Data

Qu’est-ce qu’un système de fichiers distribués ?

- Un système de fichiers distribué, ou DFS, est un système de stockage et de gestion des données qui permet aux utilisateurs ou aux applications d'accéder à des fichiers de données tels que des fichiers PDF, des documents Word, des images, des fichiers vidéo, des fichiers audio, etc., à partir d'un stockage partagé sur l'un des multiples serveurs en réseau. Avec des données partagées et stockées sur un cluster de serveurs, un DFS permet à de nombreux utilisateurs de partager des ressources de stockage et des fichiers de données sur de nombreuses machines.

II. Map Reduce pour Big Data

Les systèmes de fichier distribués les plus répandus sont :

- Ceph (https://docs.ceph.com/en/quincy/) est une plateforme libre de stockage distribué.
- GlusterFS (https://docs.gluster.org/en/latest/) est un système de fichiers réseau évolutif adapté aux tâches “data-intensive” telles que le stockage en nuage et le streaming multimédia.
- Google File System (GFS) est un système de fichiers distribué propriétaire. Il est développé par Google pour leurs propres applications
- Hadoop Distributed File System (HDFS) est un système de fichiers distribué, extensible et portable développé par Hadoop à partir du GoogleFS.

II. Map Reduce pour Big Data

Quand un système de fichiers distribués est-il essentiel ?

- Pour stocker des données de manière permanente.
- Pour partager facilement, efficacement et en toute sécurité des informations entre les utilisateurs et les applications.

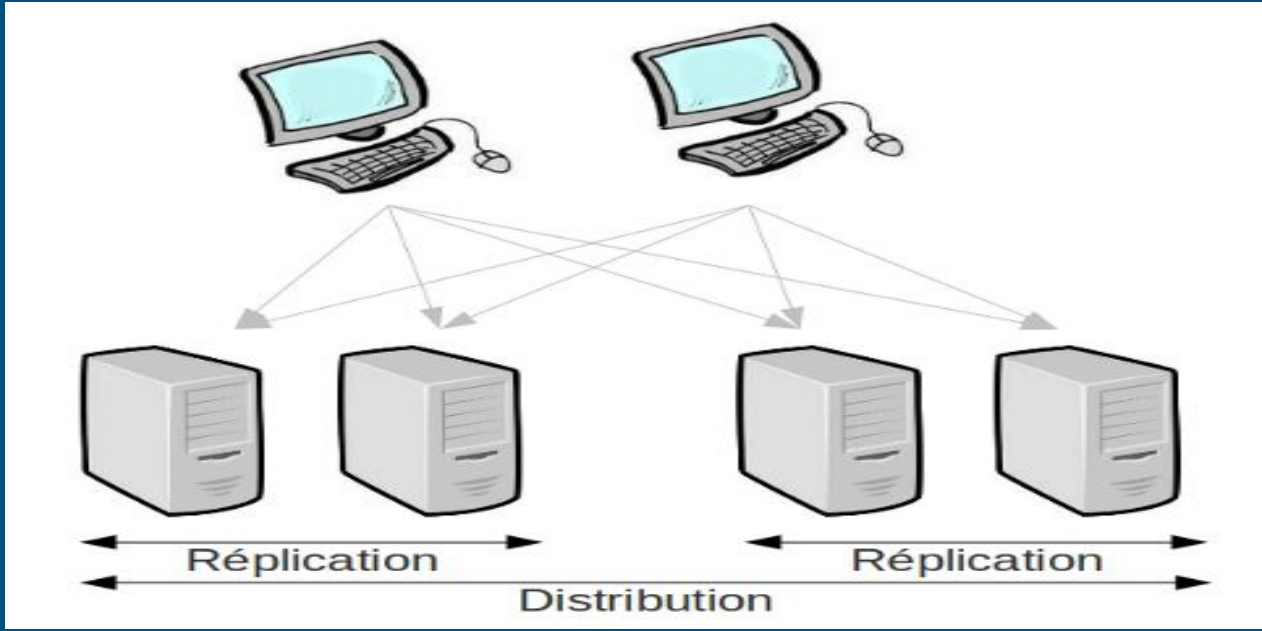
II. Map Reduce pour Big Data

Avantages d'un système de fichiers distribués :

- Tolérance aux pannes
- Un accès local transparent
- Une indépendance par rapport à l'emplacement
- De capacités scale-out : pouvoir monter en charge massivement de façon parallèle en ajoutant plus de machines. Les systèmes DFS peuvent évoluer vers des clusters très importants qui comptent des milliers de serveurs.

II. Map Reduce pour Big Data

Comment fonctionne un système de fichiers distribués ?



II. Map Reduce pour Big Data

Comment fonctionne un système de fichiers distribués ?

- Distribution : le DFS distribue d'abord les jeux de données sur plusieurs clusters ou nœuds. Chaque nœud fournit sa propre puissance de calcul, ce qui permet au DFS de traiter les jeux de données en parallèle.
- Réplication : le DFS répliquera également les jeux de données sur différents clusters en copiant les mêmes informations sur plusieurs clusters. Cela rend le système de fichiers distribués tolérant aux pannes

II. Map Reduce pour Big Data

Idée/Principe :

- Pour exécuter un problème large (Big Data) de manière distribué, il faut pouvoir découper le problème en plusieurs problèmes de taille réduite à exécuter sur chaque machine du cluster
- De multiples approches existent et ont existé pour cette division d'un problème en plusieurs

MapReduce **:**

- MapReduce inventé par Google, un cadre/modèle de traitement extrêmement parallèle adapté au traitement de très grandes quantités de données

II. Map Reduce pour Big Data

MapReduce définit deux opérations distinctes à effectuer sur les données d'entrée:

- La première, MAP, va transformer les données d'entrée en une série de couples clé/valeur. Elle va regrouper les données en les associant à des clefs, choisies de telle sorte que les couples clé/valeur aient un sens par rapport au problème à résoudre. Par ailleurs, cette opération doit être parallélisable: on doit pouvoir découper les données d'entrée en plusieurs fragments, et faire exécuter l'opération MAP à chaque machine du cluster sur un fragment distinct.

II. Map Reduce pour Big Data

MapReduce définit deux opérations distinctes à effectuer sur les données d'entrée:

- La seconde, REDUCE, va appliquer un traitement à toutes les valeurs de chacune des clés distinctes produite par l'opération MAP. Au terme de l'opération REDUCE, on aura un résultat pour chacune des clés distinctes. Ici, on attribuera à chacune des machines du cluster une des clefs uniques produites par MAP, en lui donnant la liste des valeurs associées à la clef. Chacune des machines effectuera alors l'opération REDUCE pour cette clef.

II. Map Reduce pour Big Data

On distingue donc 4 étapes distinctes dans un traitement MapReduce:

- Découper (split) les données d'entrée en plusieurs fragments.
- Mapper chacun de ces fragments pour obtenir des couples (clé; valeur).
- Grouper (shuffle) ces couples (clé; valeur) par clef.
- Réduire (reduce) les groupes indexés par clé en une forme finale, avec une valeur pour chacune des clefs distinctes.

II. Map Reduce pour Big Data

Exemple1 :

- **Objectif** : Imaginons qu'on nous donne un texte écrit en langue Française. On souhaite déterminer pour un travail de recherche quels sont les mots les plus utilisés au sein de ce texte
- **Texte**:

> “ Que la paix soit sur toi, que la paix soit sur nous,Que la paix vienne à toi, que la paix vienne à nous,Que la paix soit sur celui qui nous la souhaite “

II. Map Reduce pour Big Data

Exemple1 :

- **Première étape**: déterminer une manière de découper (split) les données d'entrée pour que chacune des machines puisse travailler sur une partie du texte.

  → On peut par exemple décider de découper les données d'entrée ligne par ligne

  que la paix soit sur toi

  que la paix soit sur nous

  que la paix vienne à toi

  que la paix vienne à nous

II. Map Reduce pour Big Data

Exemple1 :

- **Première étape**: on obtient 4 fragments depuis nos données d'entrée.

que la paix soit sur toi
que la paix soit sur nous
que la paix vienne à toi
que la paix vienne à nous

II. Map Reduce pour Big Data

Exemple1 :

- **Deuxième étape**: Mapper chacun de ces fragments pour obtenir des couples.

  → On doit désormais déterminer la clé à utiliser pour notre opération MAP, et écrire le code de l'opération MAP elle-même

  → Puisqu'on s'intéresse aux occurrences des mots dans le texte, et qu'à terme on aura après l'opération REDUCE un résultat pour chacune des clefs distinctes, la clé qui s'impose logiquement dans notre cas est: le mot-lui même.

  → Générer le couple clef/valeur: (MOT :1)

II. Map Reduce pour Big Data

Exemple1 : Deuxième étape:

POUR MOT dans LIGNE, FAIRE:
GENERER COUPLE (MOT: 1)

que la paix soit sur toi	**(que:1)(la:1)(paix:1)(soit:1)(sur:1)(toi:1)**
que la paix soit sur nous	**(que:1)(la:1)(paix:1)(soit:1)(sur:1)(nous:1)**
que la paix vienne à toi	**(que:1)(la:1)(paix:1)(vienne:1) (à:1)(toi:1)**
que la paix vienne à nous	**(que:1)(la:1)(paix:1)(vienne:1)(à:1)(nous:1)**

II. Map Reduce pour Big Data

Exemple1 : Troisième étape

→Grouper ces couples par clé; (COUPLE(s) par GROUPE)

(que:1)(que:1)(que:1)(que:1)	**(soit:1)(soit:1)**	**(à:1) (à:1)**
(la:1)(la:1)(la:1)(la:1)	**(vienne:1)(vienne:1)**	**(toi:1) (toi:1)**
(paix:1)(paix:1)(paix:1)(paix:1)	**(sur:1) (sur:1)**	**(nous:1) (nous:1)**

II. Map Reduce pour Big Data

Exemple1 : Quatrième étape

→Il nous reste à créer notre opération REDUCE, qui sera appelée pour chacun des groupes/clé distincte, elle va simplement consister à additionner toutes les valeurs liées à la clé spécifiée

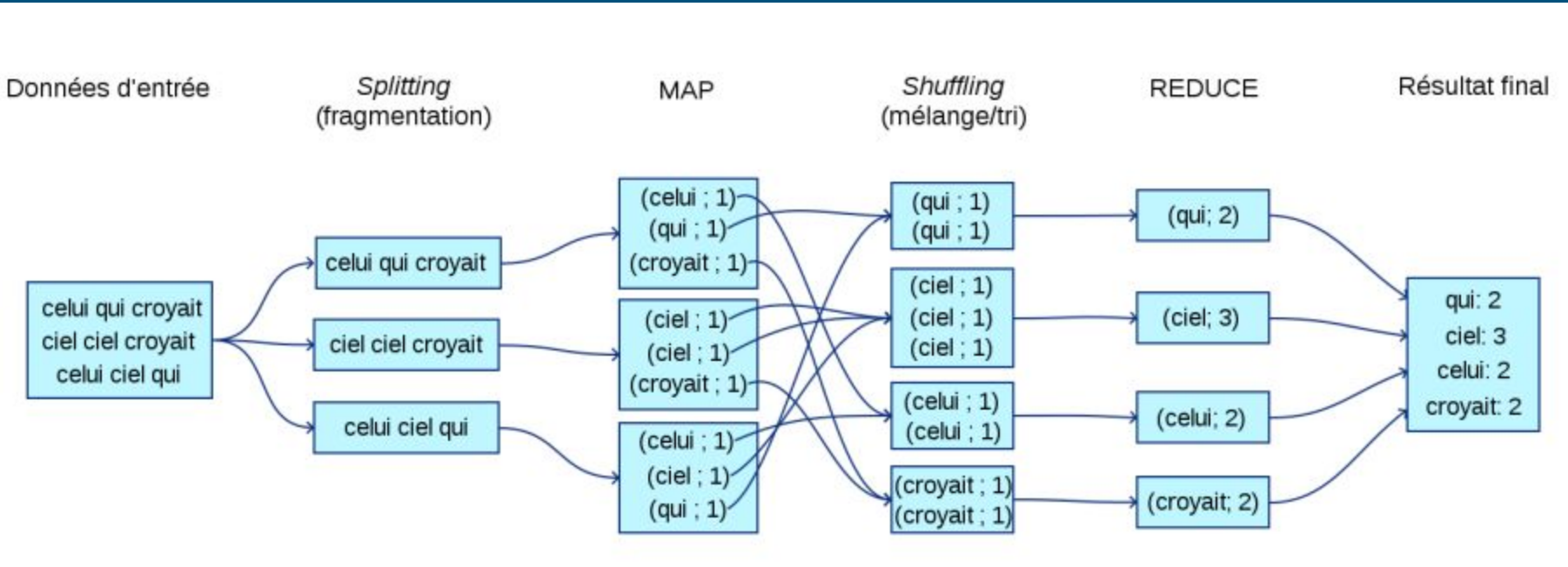
```
nbr = 0
POUR COUPLE dans GROUPE, FAIRE:
      nbr = nbr + 1
RENVOYER nbr
```

II. Map Reduce pour Big Data

Exemple1 : Quatrième étape

que → 4
la → 4
paix→ 4
soit → 2
…

II. Map Reduce pour Big Data



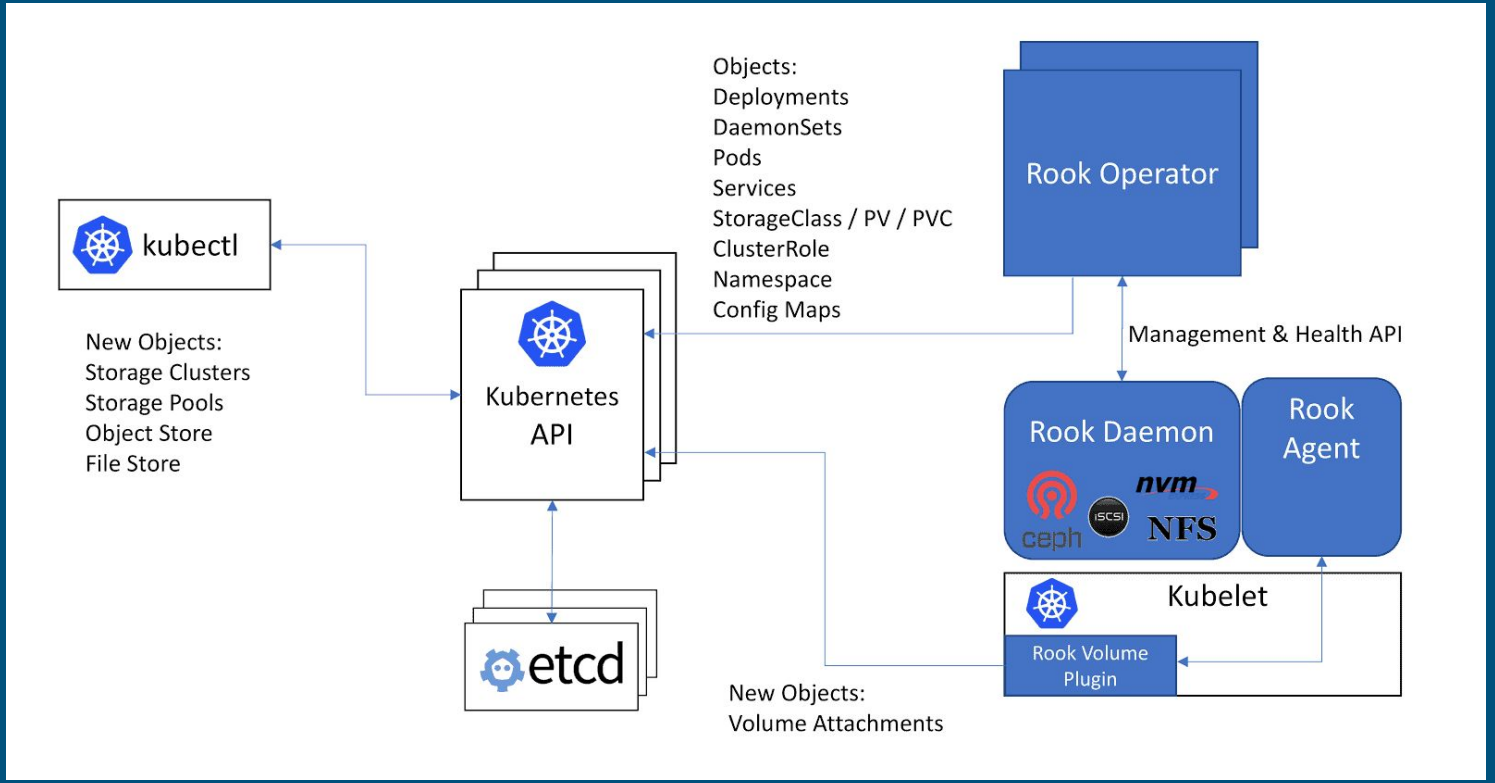
III. Le Cloud Native File System (CNFS)

- Le Cloud Native File System (CNFS) est un système de fichiers conçu pour les applications cloud natives, qui sont des applications qui sont développées et déployées dans des environnements cloud. Le CNFS est conçu pour fonctionner dans des environnements de conteneurs, tels que Kubernetes, et permet aux applications d'accéder aux données stockées dans le cloud de manière transparente.

III. Le Cloud Native File System (CNFS)

- Le CNFS utilise des API de stockage de cloud pour offrir une abstraction de stockage qui permet aux applications de stocker, d'accéder et de gérer des données de manière transparente, indépendamment de l'emplacement physique de ces données.
- Le CNFS est souvent utilisé pour stocker des données dans des environnements cloud, tels que des conteneurs, des clusters de conteneurs et des plateformes cloud hybrides

III. Le Cloud Native File System (CNFS)



IV. Modèles de déploiement de stockage

Il existe plusieurs modèles de déploiement de stockage dans le cloud pour le big data, voici les principaux :

1. Stockage objet : c'est un modèle de stockage dans lequel les données sont stockées en tant qu'objets avec des métadonnées associées, ce qui permet un accès efficace aux données. Les exemples de stockage objet dans le cloud incluent Amazon S3, Google Cloud Storage et Microsoft Azure Blob Storage.
2. Stockage en bloc : c'est un modèle de stockage dans lequel les données sont stockées en blocs, souvent utilisé pour les bases de données et les applications nécessitant des performances élevées en termes de lecture et d'écriture. Les exemples de stockage en bloc dans le cloud incluent Amazon Elastic Block Store (EBS) et Microsoft Azure Disks.

IV. Modèles de déploiement de stockage

3. Stockage de fichiers : c'est un modèle de stockage dans lequel les données sont stockées en tant que fichiers dans un système de fichiers distribué, permettant une gestion de fichiers traditionnelle. Les exemples de stockage de fichiers dans le cloud incluent Amazon Elastic File System (EFS) et Google Cloud Filestore.
4. Stockage en base de données : c'est un modèle de stockage dans lequel les données sont stockées dans une base de données distribuée pour permettre une récupération rapide et une analyse de données. Les exemples de stockage de bases de données dans le cloud incluent Amazon Relational Database Service (RDS), Google Cloud SQL et Microsoft Azure SQL Database

IV. Modèles de déploiement de stockage

5. Stockage en cache : c'est un modèle de stockage dans lequel les données sont stockées dans une mémoire cache pour permettre un accès rapide et une réduction des temps de latence. Les exemples de stockage en cache dans le cloud incluent Amazon ElastiCache, Google Cloud Memorystore et Microsoft Azure Cache for Redis.

→ Le choix du modèle de stockage dépendra de plusieurs facteurs, tels que la taille des données, la complexité de l'application, la fréquence d'accès aux données et les besoins en termes de sécurité.

V. Base de données dans le Cloud

- Les bases de données dans le cloud sont des systèmes de gestion de base de données (SGBD) qui sont hébergés et exécutés dans un environnement cloud. Les bases de données dans le cloud offrent de nombreux avantages par rapport aux SGBD traditionnels, tels que la flexibilité, la scalabilité, la disponibilité, la sécurité et la facilité de gestion.
- Les bases de données dans le cloud peuvent être classées en plusieurs catégories, selon leur fonctionnement :

V. Base de données dans le Cloud

1. Les bases de données relationnelles dans le cloud : ces SGBD stockent des données dans des tables reliées les unes aux autres par des clés. Les exemples de bases de données relationnelles dans le cloud incluent Amazon Relational Database Service (RDS), Google Cloud SQL et Microsoft Azure SQL Database.
2. Les bases de données NoSQL dans le cloud : ces SGBD stockent des données sous forme de documents, graphes, clés/valeurs ou colonnes. Ils sont utilisés pour des applications nécessitant une grande scalabilité, une disponibilité élevée et une faible latence. Les exemples de bases de données NoSQL dans le cloud incluent Amazon DynamoDB, Google Cloud Firestore et Microsoft Azure Cosmos DB

V. Base de données dans le Cloud

3. Les bases de données de traitement en mémoire : ces SGBD stockent des données en mémoire pour offrir des performances de lecture/écriture ultra-rapides et une faible latence. Ils sont utilisés pour les applications nécessitant une vitesse de traitement élevée, telles que l'analyse de données en temps réel. Les exemples de bases de données de traitement en mémoire dans le cloud incluent Amazon ElastiCache, Google Cloud Memorystore et Microsoft Azure Cache for Redis.

→ Les bases de données dans le cloud sont faciles à déployer et à gérer, car les fournisseurs de services cloud prennent en charge les tâches de maintenance et de mise à l'échelle.

VI. Base de données en tant que service

Une base de données en tant que service (DBaaS) est un service cloud qui fournit un accès à une base de données hébergée et gérée par un fournisseur de cloud. Avec un DBaaS, les utilisateurs n'ont pas à gérer l'infrastructure sous-jacente et les tâches de maintenance associées à la base de données. Au lieu de cela, le fournisseur de DBaaS gère la base de données, y compris la configuration, la sauvegarde, la récupération et la mise à l'échelle.

Les avantages d'une base de données en tant que service incluent :

- Facilité de déploiement : les DBaaS peuvent être facilement déployées à partir du portail de gestion du fournisseur cloud, ce qui permet un déploiement rapide et facile de la base de données.

VI. Base de données en tant que service

- Gestion simplifiée : le fournisseur de DBaaS s'occupe des tâches de maintenance de la base de données, telles que la mise à jour, la sauvegarde, la récupération et la surveillance, ce qui permet aux utilisateurs de se concentrer sur le développement d'applications.
- Coûts réduits : les DBaaS sont souvent facturées à l'utilisation, ce qui permet aux utilisateurs de payer uniquement pour les ressources qu'ils utilisent.
- Évolutivité : les DBaaS sont facilement évolutives, ce qui permet aux utilisateurs d'ajuster les ressources allouées à la base de données en fonction de la demande.

VI. Base de données en tant que service

- Sécurité : les fournisseurs de DBaaS offrent des mesures de sécurité pour protéger les données stockées dans la base de données, telles que la surveillance des menaces, la conformité réglementaire et la protection contre les intrusions.

→ Les exemples de fournisseurs de DBaaS incluent Amazon Relational Database Service (RDS), Google Cloud SQL, Microsoft Azure SQL Database, IBM Cloudant, MongoDB Atlas, Oracle Database Cloud Service, ...

VII. Bases de données commerciales NoSQL

Il existe plusieurs bases de données commerciales NoSQL disponibles sur le marché qui offrent des fonctionnalités et des capacités étendues pour gérer des données de manière non relationnelle:

- MongoDB : MongoDB est une base de données NoSQL **orientée document** qui offre une grande flexibilité de schéma et une évolutivité horizontale. Il prend en charge des fonctionnalités telles que l'indexation, l'agrégation, la réplication et la mise en cluster.

VII. Bases de données commerciales NoSQL

- Cassandra : Cassandra est une base de données NoSQL **orientée colonne** qui offre une grande évolutivité horizontale et une disponibilité élevée. Il prend en charge des fonctionnalités telles que la mise en cluster, la réplication, la tolérance aux pannes et la récupération après sinistre.
- DynamoDB : DynamoDB est un service de base de données NoSQL entièrement géré par AWS qui prend en charge les opérations de lecture et d'écriture à haute performance et les mises à l'échelle automatiques en fonction de la demande. Il est **orienté clé-valeur** et offre une évolutivité horizontale et une disponibilité élevée.

VII. Bases de données commerciales NoSQL

- Google Cloud Datastore : Google Cloud Datastore est une base de données NoSQL entièrement gérée par Google qui prend en charge les opérations de lecture et d'écriture à haute performance et une grande évolutivité horizontale. Il est orienté document et offre une API REST pour accéder aux données.
- …

→ Ces bases de données commerciales NoSQL offrent des fonctionnalités avancées pour répondre aux besoins spécifiques des applications modernes, telles que l'évolutivité, la performance et la disponibilité. Cependant, il est important de considérer les coûts associés à l'utilisation de ces bases de données, ainsi que leur adéquation pour les besoins de l'application avant de les choisir.

Fin

Merci Pour Votre Attention