SHAKE THE FUTURE

CENTRALE NANTES

Bases de Données

Les bases de données noSQL

JY Martin

Plan

1 Le contexte

2 Fondements de NoSQL

3 Classes de serveurs

4 Bilan

Introduction à NoSQL

Constat :

- De plus en plus d'information collectées (Google, Facebook, ...)
- Données distribuées
- Mal structurées, hétérogènes => Le relationnel ne suffit pas

Besoin :

- Contextes fortement distribués
- Objets complexes, hétérogènes
- Gros volumes de données => Not Only SQL (noSQL)

Les SGBD Relationnels ne suffisent pas

SGBD Relationnel => Contraintes ACID

- Atomicity = tout ou pas du tout
- Consistancy = D'un état valide à un autre
- Isolation = S'exécute comme si elle était unique
- Durability = Une fois effectuée, la modification est enregistrée

Propriétés difficiles à respecter en milieu fortement distribué en gardant des performances comparables

Vers les systèmes distribués

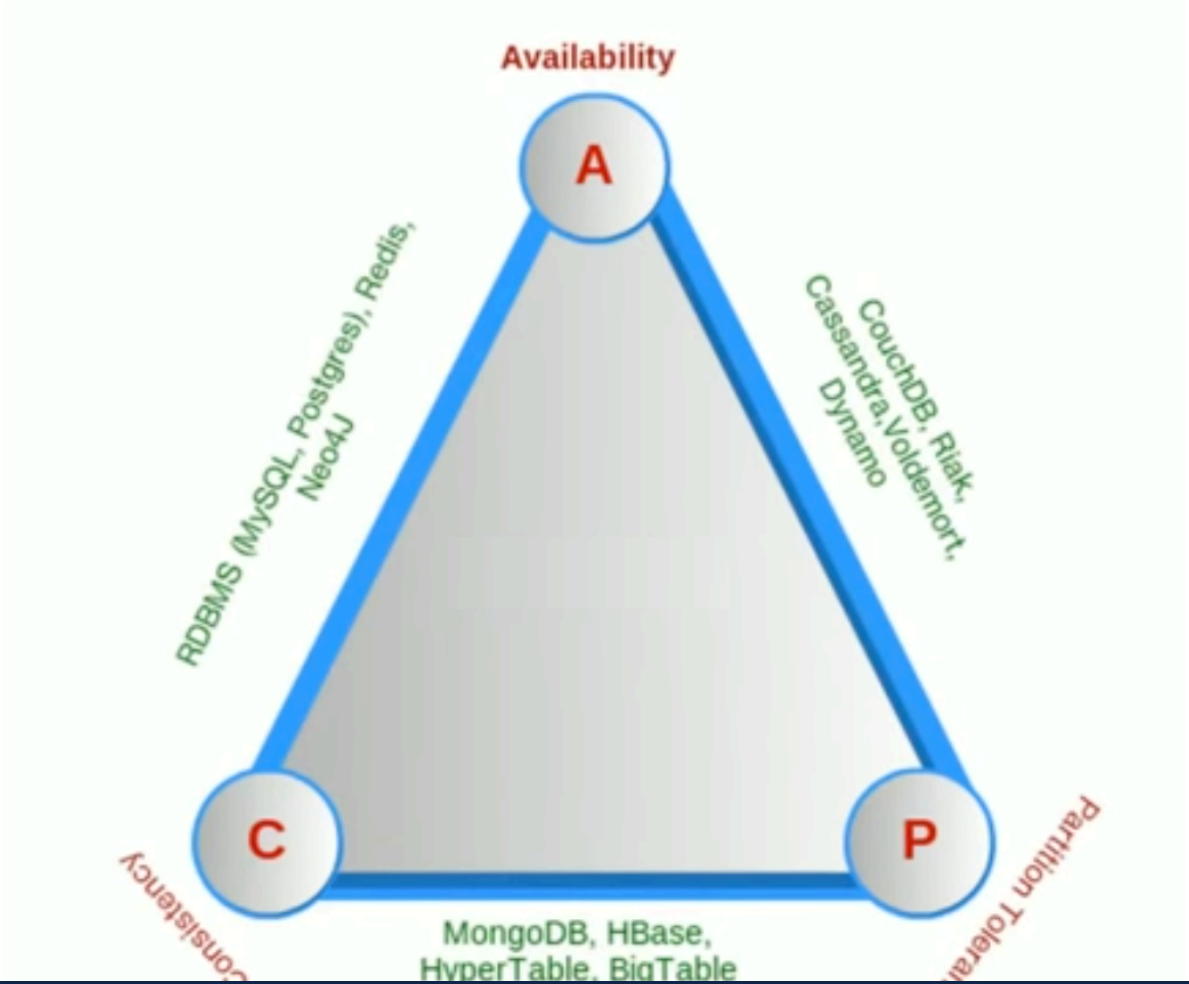
Système distribués : 3 propriétés = CAP

- Coherence = tous les nœuds voient les même données au même moment
- Availability = Le système répond toujours aux requêtes
- Partition Tolerence = Une fois partitionné, seule une panne totale du réseau peut l'empêcher de fonctionner

Théorème de CAP (Brewer, 2000)

Dans un système distribué, il est impossible d'obtenir ces 3 propriétés CAP en même temps, il faut en choisir 2 parmi les 3

Vers les systèmes distribués



Vers les systèmes distribués

A-Z		Consistant, Non distribué, Résistant
A-M	N-Z	Consistant, Non disponible, Non Résistant
A-Z	A-Z	Non Consistant, Disponible, Résistant

NoSQL

En résumé sur les BD noSQL

- Représentation des données non relationnelle
- Gestion de gros volumes de données
- Forte distribution des données
- Performantes
- Pas de schéma
- Données complexes ou imbriquées
- Réplication éventuelle des données sur plusieurs nœuds
- Ne remplissent pas complètement les conditions ACID
- Privilégient la Disponibilité de la donnée à sa Cohérence
- Peu d'écriture, Beaucoup de lectures

Plan

1 Le contexte

2 Fondements de NoSQL

3 Classes de serveurs

4 Bilan

Principaux éléments

- Sharding : partitionnement des données sur plusieurs serveurs
- Consistent hashing : partitionnement des données sur plusieurs serveurs eux même partitionnés sur un segment
- Map Reduce : modèle de programmation parallèle
- MVCC : Contrôle de cohérence Multi-Versions
- Vector-Clock : permet la mise à jour concurrentes en datant les données

Le Sharding

Sharding = Partitionnement des données sur plusieurs serveurs

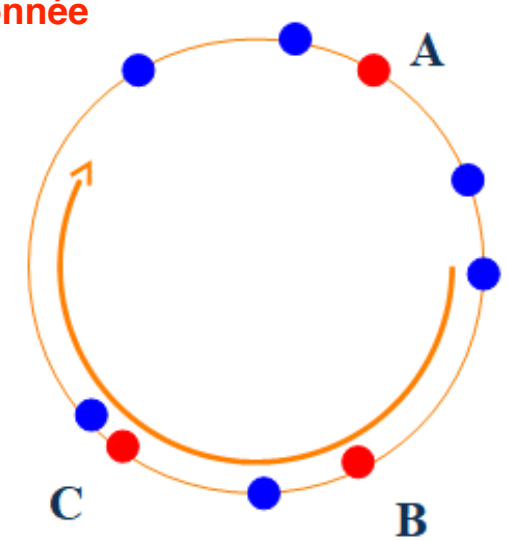
- Partitionnement horizontal = on découpe la population
  - Les données liées sont sur le même serveur
  - Répartition de la charge entre les serveurs
    => Duplication de certains nœuds sur des serveurs
- Partitionnement vertical = on découpe par colonne

Consistent Hashing

Pour le Partitionnement horizontal uniquement

- Données et serveurs sont “hashés” par une même fonction
- Les données sont associées au serveur qui les précède sur un anneau représentant les valeurs de hashage

chercher le serveur puis trouver le donnée



Map-Reduce

- Modèle de programmation parallèle pour le traitement de grands ensembles de données en environnement distribué
- Utilisé par les grands acteurs du web
  - Construire des index (Google)
  - Détection de spam (Yahoo)
  - Data Mining (Facebook)
  - ...

Map-Reduce

Mais pas que :

- Analyse d'images astronomiques
- BioInformatique
- Simulation météo
- Machine Learning
- ...

Map-Reduce

- Répartit la charge entre les serveurs
- Gestion transparente de la distribution des données, de la répartition de charge, de la tolérance aux pannes, ...
- Permet d'améliorer les performances en cas d'ajout de machines
- API existant dans de nombreux langages de programmation

Map Reduce

Utilise 2 opérateurs

- Map : Traite une paire clé valeur et génère une paire de clés intermédiaires / valeurs
- Reduce :fusionne toutes les valeurs intermédiaires associées à la même clé intermédiaire

Map Reduce

Principe de fonctionnement :

- Itérer un grand nombre d'enregistrements
- Extraire quelque chose ayant un intérêt de la part de chacun des enregistrements
- Regrouper et trier les résultats intermédiaires
- Agréger les résultats
- Générer un résultat final

Map Reduce en pratique

On implémente 2 fonctions

- Map(data) -> List(cléInterm, valeurInterm)
- Reduce(cléInter, List(valeursInterm)) -> valeur

L'algorithme comporte 5 phases

- Initialisation
- Map
- Regroupement
- Tri
- Reduce

Exemple Map Reduce

Compter le nombre de mots contenus dans un fichier

Approche classique :

- Parcourir les lignes
  - Pour chaque ligne, compter les mots et faire la somme

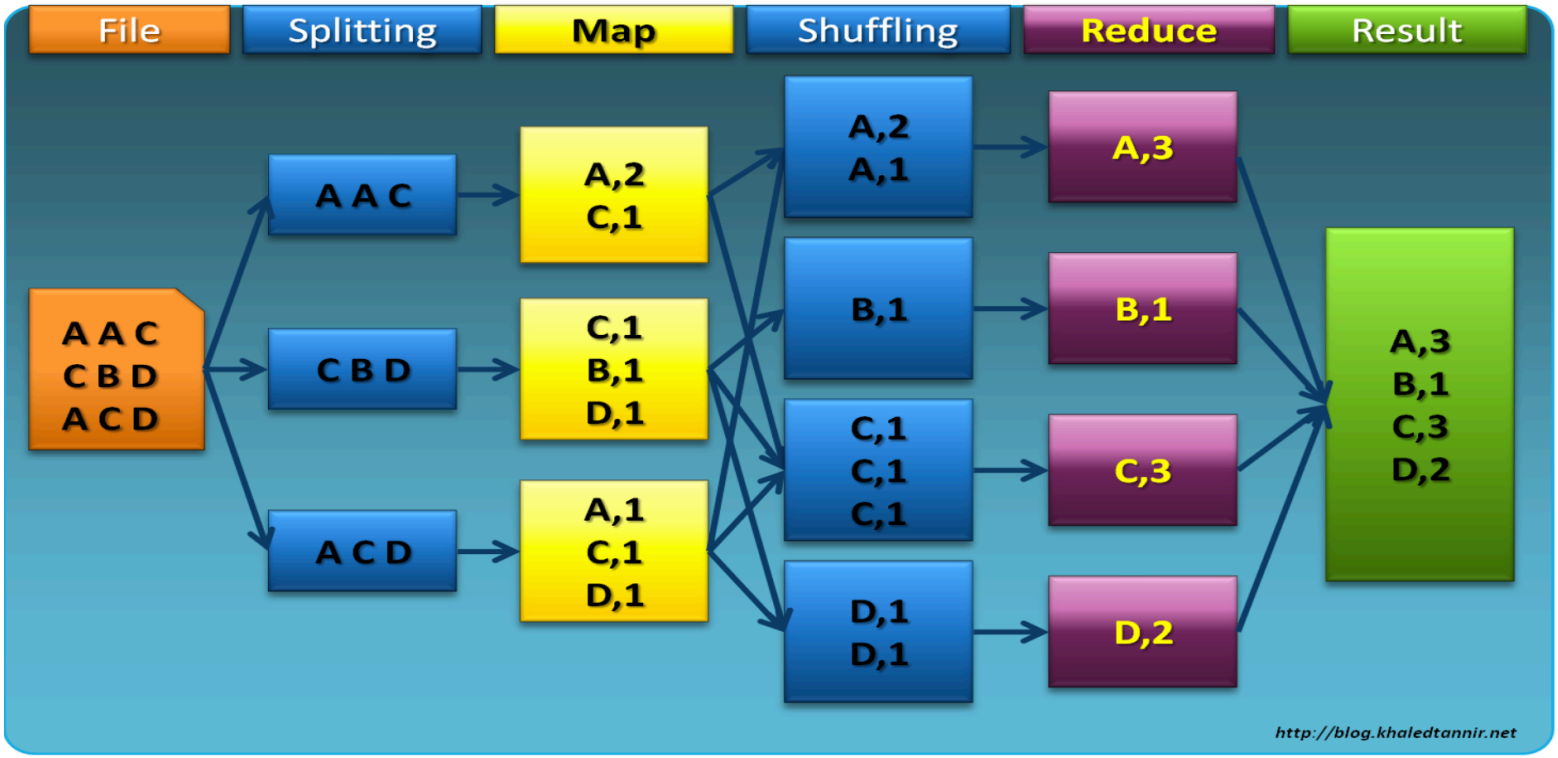
Parallelisation => synchronisation du processus de somme de nombre d'apparitions

Exemple Map Reduce

Approche Map-Reduce :

- Initialisation : découpage des lignes pour la répartition entre les procesus
- Map : traitement d'une ligne = établir une association mot-nombre d'apparitions
- Regroupement : regrouper les nombre d'apparitions pour un même mot clé
- Tri : chaque processus traite un mot clé : il fait la somme des nombres d'apparitions
- Reduce : regroupement des couples mot-clé / nombre d'apparitions

Exemple Map Reduce



Contrôle de Concurrence Multi-Versions (MVCC)

Gère les accès simultanés aux données, en particulier pour les mises-à-jour

- Marque les anciennes données
- Ajoute une nouvelle version des données et les marque avec une horloge logique
- Effectue un balayage régulier des données pour éliminer celles qui sont obsolètes

Plan

1 Le contexte

2 Fondements de NoSQL

3 Classes de serveurs

4 Bilan

Les grandes classes de serveurs

- Modèle Clé Valeur :
  Chaque objet est identifié par une clé unique
- Modèle Colonne :
  Chaque ligne comporte un grand nombre de valeurs
- Modèle Document :
  Gestion de collections de documents
- Modèle Graphe :
  Gestion de relations multiples entre objets

Modèle Clé-Valeur

- Fonctionne un peu à la manière d'une table de hashage
- Les données sont représentées par un couple clé / valeur
- Pour accéder à une donnée on passe par sa clé

K0

K1 -> valeurs

K2

K3 -> valeurs

K4

Modèle Clé-Valeur

- Avantages :
  - Modèle simple
  - Partitionnement horizontal facilité
  - Modification du nombre de serveur
  - Disponibilité
  - Pas de maintenance particulière lors de l'ajout / supression de colonnes
- Inconvénients
  - Trop simple
  - Pauvre pour les données complexes
  - Interrogation uniquement sur la clé
  - Déporte la complexité vers l'application

Modèle Clé-Valeur

Exemples de serveurs :

- Voldemort,
- Redis,
- Riak,
- DynamoDB,
- ...

Modèle Colonne

- Les données sont stockées en colonne plutôt qu'en lignes
- Modèle proche des SGBDR mais avec un nombre de colonne dynamique et pas forcément identique d'une ligne à l'autre
- Possibilité de compresser les données d'une colonne quand elles se ressemblent
- On distingue
  - Colonne :Champ de données, définie par un couple clé-valeur
  - Famille de colonnes : Permet de regrouper plusieurs colonnes ou super-colonnes
  - Super-Colonne : utilisées comme les lignes d'une table de jointure d'un modèle relationnel

Modèle Colonne

- Avantages :
  - Modèle supportant les données semi-structurées
  - Naturellement indexé (colonnes)
  - Bonne mise à l'échelle horizontale
- Inconvénients
  - A éviter si les données sont trop interconnectées
  - A éviter pour les données complexes
  - Maintenance nécessaire en cas d'ajout / suppression de colonnes
  - Il faut avoir une idées des requêtes à effectuer avant de concevoir le modèle

Modèle Colonne

Exemples de serveurs :

- Hbase,
- Cassandra,
- Hypertable,
- ...

Modèle Document

- Stockent une collection de « document »
- Basée sur le modèle clé-valeur, la valeur étant un document
- Les documents n'ont pas de schéma mais une structure arborescente. Ils contiennent une liste de champs et les valeurs associées
- Les champs d'un document ne sont pas obligatoirement prédéfinis
- Les documents peuvent être hétérogènes
- Permettent de faire des requêtes sur le contenu des documents
- Principalement utilisées pour les CMS

Modèle Document

- Avantages :
  - Modèle de données simple mais puissant
  - Bonne mise à l'échelle
  - Pas de maintenance pour ajouter / supprimer des colonnes
  - Possibilité de faire des requêtes complexes
- Inconvénients
  - Ne pas utiliser pour des données interconnectées
  - Modèle de requête limité à des clés
  - Lent pour les grandes requêtes

Modèle Document

Exemples de serveurs :

- MongoDB,
- Couchbase,
- ...

Modèle Graphe

- Modélisation stockage et manipulation de données complexes liées par des relations non triviales ou variables
- Basé sur la théorie des graphes
- S'appuient sur la notion de nœuds, de relations et de propriétés
- Utilisent
  - Un moteur de stockage pour les objets
  - Un mécanisme de description d'arcs pour les relations entre objets
- Plus efficaces que les BD Relationnelles pour les problématiques de réseaux

Modèle Graphe

- Avantages :
  - Modèle de données puissant
  - Rapide pour les données liées
  - Modèle d'interrogation performants
- Inconvénients
  - Fragmentation des données

Modèle Graphe

Exemples de serveurs

- Neo4J,
- InfiniteGraph,
- DEX,
- OrientDB,
- ...

Plan

1 Le contexte

2 Fondements de NoSQL

3 Classes de serveurs

4 Bilan

Pourquoi choisir un serveur noSQL

SGBDR et NoSQL ne fournissent pas les mêmes services

SGBDR

- Données structurées
- Interconnectées
- Propriétés ACID

NoSQL

- Grand nombre de données
- Données distribuées
- Peu ou pas interconnectées


Success
NEXT EXIT