Structures des données dans les systèmes NoSQL

Ètude de cas – M2 Data – Ynov

Mohammed El Malki

7 avril 2025

Objectifs

Ce Projet a pour objectif de choisir le moteur NoSQL adéquat en fonction d'un besoin métier précis. Précisément, nous allons aborder les points suivants :

1. Comparaison des tois moteurs : MongoDB, Neo4j et Cassandra
2. Argumentation et critères ayant permis le choix du moteur
3. Proposition d'une modélisation permettant de répondre au besoin de l'entreprise (requêtes en annexe)
4. Proposition d'un dimensionnement de votre cluster orienté Cassandra, Neo4j ou MongoDB. Justifier le dimensionnement proposé.

Ce travail :

—est à rendre pour le **8 avril 2025 à 23h59**

—doit être rendu sous forme de rapport couvrant les différents aspects de votre étude (choix du moteur, modélisation, comparaisons, requêtes, architecture, sauvegarde/restauration...)

—fera l'objet d'une soutenance le **8 vril 2025**

1. Étude de cas

1.1. Étude à réaliser

Vous êtes une entreprise spécialisée dans le management des données avec des moteurs NoSQL. Pour l'évolution de son système d'information, l'entreprise **IADATA** vous a sélectionnés pour l'accompagner dans cette transition.

Vous devez l'aider à :

1. Faire une étude comparative des trois solutions.
2. proposer deux modélisations basées sur deux moteurs différents de votre choix.
3. Proposer une structuration des données sous le moteur MongoDB ou Neo4J
4. Traduire les requêtes dans le langage MongoDB (ou Neo4j) et vérifier qu'elles sont passantes (apporter les transformations si nécessaire)
5. Proposer une structuration des données sous le moteur Cassandra
6. Traduire les requêtes dans le langage de Cassandra et les adapter si besoin
7. Choisir le moteur NoSQL et la modélisation adéquate selon vous, et justifier votre choix

8. Vérifier si votre modélisation peut répondre à l'évolution des traitements. Argumenter.
9. Proposer une architecture de cluster, en précisant les aspects de cohérence, de résilience, de réplication, etc.
10. Justifier votre architecture à la lumière du modèle CAP.
11. Sachant que le client exprime un contexte où la cohérence de données est un point très critique, proposer une politique de consistance (en lecture et écriture) pour répondre à cette exigence.

1.2. Traitements métiers

Voici les besoins/requêtes à considérer :

—Noms des logiciels UNIX

—Type du poste p8

—Nom, adresse IP, numéro de salle des postes de type UNIX ou PCWS

—Postes du segment 130.120.80 triés par numéro de salle décroissant

—Numéros des logiciels installés sur le poste p6

—Numéros des postes qui hébergent le logiciel log1

—Nom et adresse IP complète des postes de type TX

—Nombre de logiciels par poste

—Noms des salles hébergeant au moins un poste avec le logiciel 'Oracle 6'

2. Contexte d'exploitation, sauvegarde et restauration

L'entreprise IADATA souhaite mettre en place une **politique de sauvegarde et de restauration** adaptée à son environnement de production, basé sur un moteur NoSQL distribué (MongoDB, Cassandra ou Neo4j, selon votre choix).

L'entreprise effectue **chaque nuit des traitements critiques de facturation**. En cas de défaillance durant ces traitements, une **restauration rapide et fiable** est requise à partir de la sauvegarde de la veille. Avant la restauration, une **copie des données altérées** doit être conservée pour analyse.

Deux partenaires externes doivent recevoir quotidiennement une **copie cohérente des données** :

—Le premier utilise le même moteur que la production et exige une **copie complète**.

—Le second demande une **extraction au format CSV**.

L'entreprise dispose également de plusieurs **clusters de pré-production** pour tester des workflows métier. Ces environnements peuvent être restaurés à partir de sauvegardes de production.

Travail demandé

Vous devez :

1. Définir une politique de sauvegarde adaptée au moteur choisi (fréquence, format, cohérence, gestion des exports).
2. Proposer une politique de restauration pour :
   —La production en cas d'échec nocturne.
   —La préparation d'un cluster de pré-production à partir d'une sauvegarde.
3. Décrire comment gérer les cas de sauvegardes incomplètes (certaines instances non sauvegardées).
4. Préciser les conditions pour qu'une sauvegarde soit considérée comme **valide et exploitable**.

2.1. Sauvegarde localisée vs externalisée

Lors de la mise en place de la politique de sauvegarde, l'entreprise souhaite pouvoir externaliser les sauvegardes vers un backend distant (type `S3`, `MinIO`, stockage objet cloud...).

Le processus de sauvegarde doit gérer les deux cas suivants :

—Si le backend distant est disponible, la sauvegarde est directement poussée vers celui-ci.

—En cas d'indisponibilité du backend distant, une **sauvegarde locale** est réalisée sur le nœud. Une solution doit alors permettre de synchroniser les sauvegardes locales vers le backend distant une fois celui-ci de nouveau disponible.

Lors de la restauration, chaque nœud peut retrouver ses propres sauvegardes soit en local, soit depuis le backend distant, selon la disponibilité au moment de la sauvegarde.

Travail attendu :

—Décrire une solution de sauvegarde compatible avec ces contraintes (local fallback + push différé).

—Définir les implications de ce modèle sur la politique de restauration : comment garantir l'accès aux données si une partie est locale et une autre sur backend distant ?

—Proposer un mécanisme de suivi ou d'orchestration de l'état des sauvegardes et de leur synchronisation.

3. Démarrage consistant et automatique

Actuellement, dans le texte actuel, basé sur un moteur relationnel Oracle, en cas de perte de l'instance, une politique de démarrage automatique est en place pour redémarrer automatiquement l'instance Oracle.

Proposer une solution de **démarrage consistant et automatique** qui prend en considération les contraintes de cohérence de données (lecture écriture) pour que le démarrage automatique ne cause aucune corruption des données.

Annexe : Comparaison des moteurs

En conclusion de votre rapport, vous réaliserez une comparaison entre MongoDB et Cassandra à partir de votre expérience :

—Modèle de données

—Performance sur les requêtes métier

—Cohérence / disponibilité

—Scalabilité et tolérance aux pannes

—Facilité de sauvegarde/restauration

4. Outils de sauvegarde et restauration par moteur

Pour orienter votre réflexion, voici quelques outils typiques utilisés pour la sauvegarde et la restauration dans les trois moteurs NoSQL à l'étude :

—**MongoDB** : `mongodump`, `mongorestore`, `mongosync`, snapshots filesystem

—**Cassandra** : `nodetool snapshot`, `Medusa`, `sstableloader`

—**Neo4j** : `neo4j-admin backup/restore`, `apoc.export`, `dump` et `load`

Vous pouvez intégrer ces outils dans votre stratégie globale de sauvegarde/restauration, tout en les adaptant aux contraintes du sujet (sauvegarde partielle, synchronisation, cohérence, restauration multi-nœud, etc.).