Compléments sur le moteur NoSQL MongoDB et d'un Moteur NoSQL de votre Choix

Cette section présente une analyse théorique et pratique de plusieurs concepts liés aux moteurs NoSQL. Nous nous focalisons principalement sur **MongoDB** et, à titre d'exemple, sur **Cassandra**, un moteur NoSQL orienté colonnes reconnu pour sa scalabilité et sa tolérance aux pannes.

---

1. Modèles de données supportés

MongoDB

- **Type de modèle :**
  MongoDB est une base de données orientée documents.
- **Format des données :**
  Utilise le format BSON (Binary JSON), une extension binaire de JSON, qui permet de stocker des données complexes (documents, tableaux, sous-documents, etc.).
- **Structure :**
  Les données sont organisées en collections (similaires aux tables en SQL) qui contiennent des documents (similaires aux enregistrements), où chaque document possède des champs pouvant être de types variés et possiblement imbriqués.
- **Application pratique :**
  Dans notre système, chaque entité telle que **Member**, **Publisher**, **Category** ou **Book** est stockée sous forme de document, ce qui offre une grande flexibilité dans l'évolution du schéma des données sans nécessiter de migrations lourdes.

Cassandra

- **Type de modèle :**
  Cassandra est une base de données orientée colonnes (Wide Column Store).
- **Format des données :**
  Les données sont organisées en tables, mais avec un schéma très flexible qui permet d'avoir des colonnes dynamiques.
- **Structure :**
  Les données sont stockées dans des lignes identifiées par une clé primaire et organisées en familles de colonnes. Ce modèle supporte des structures hiérarchiques de colonnes.
- **Application pratique :**
  Si nous choisissions Cassandra comme moteur alternatif, chaque entité serait modélisée sous forme de ligne dans une table. Par exemple, la table des **Members** pourrait être conçue de manière à optimiser les requêtes de lecture par email ou statut, avec des colonnes dynamiques pour stocker des informations supplémentaires si nécessaire.

---

2. Réévaluer la procédure d'installation du moteur et des utilitaires

MongoDB

- **Procédure d'installation :**
  L'installation de MongoDB repose sur des packages précompilés disponibles pour divers systèmes d'exploitation ou via des gestionnaires de packages comme `apt` ou `yum`.
- **Utilitaires :**
  - `mongod` : Le serveur MongoDB.
  - `mongo` : Le shell de commande pour interagir avec la base.
  - Des outils comme `MongoDB Compass` permettent d'effectuer des analyses visuelles et administratives.
- **Pratique dans notre système :**
  Une procédure d'installation bien documentée et automatisée (via des scripts ou une configuration Docker) garantit que tous les développeurs et systèmes de production déploient une configuration identique et optimisée.

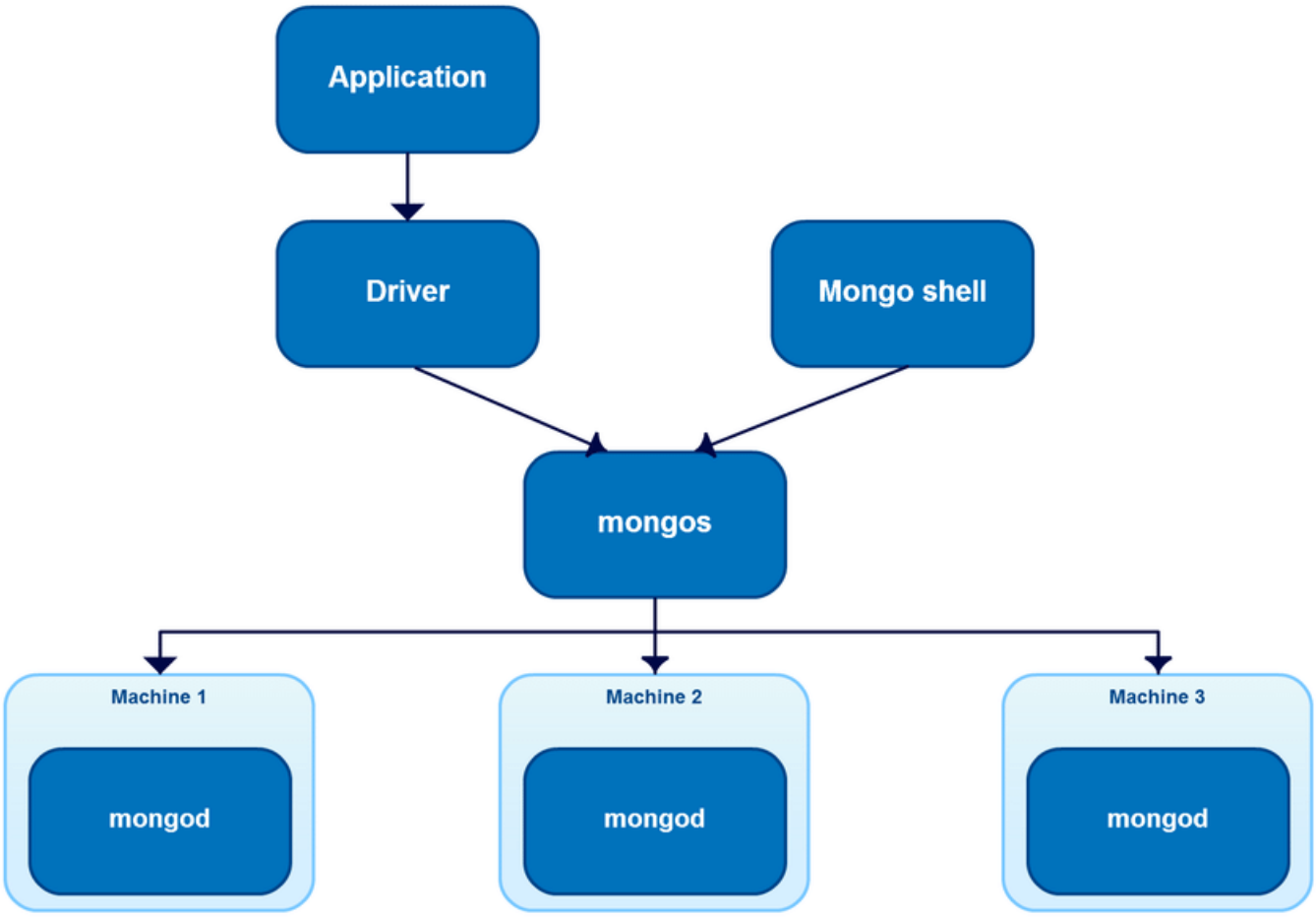
Cassandra

- **Procédure d'installation :**
  Cassandra s'installe via des packages tar.gz ou via des gestionnaires de packages. Il est également courant d'utiliser des clusters Docker pour les environnements de test et de production.
- **Utilitaires :**
  - `cassandra` : Le service principal.
  - `cqlsh` : L'interface en ligne de commande basée sur le langage CQL (Cassandra Query Language).
  - Des outils de monitoring comme `nodetool` pour la gestion du cluster.
- **Pratique dans notre système :**
  La procédure doit inclure des vérifications de compatibilité, la configuration des nœuds et la mise en place d'un environnement de test permettant de simuler des charges pour valider la scalabilité.

---

3. Architecture du moteur NoSQL (schémas expliqués)

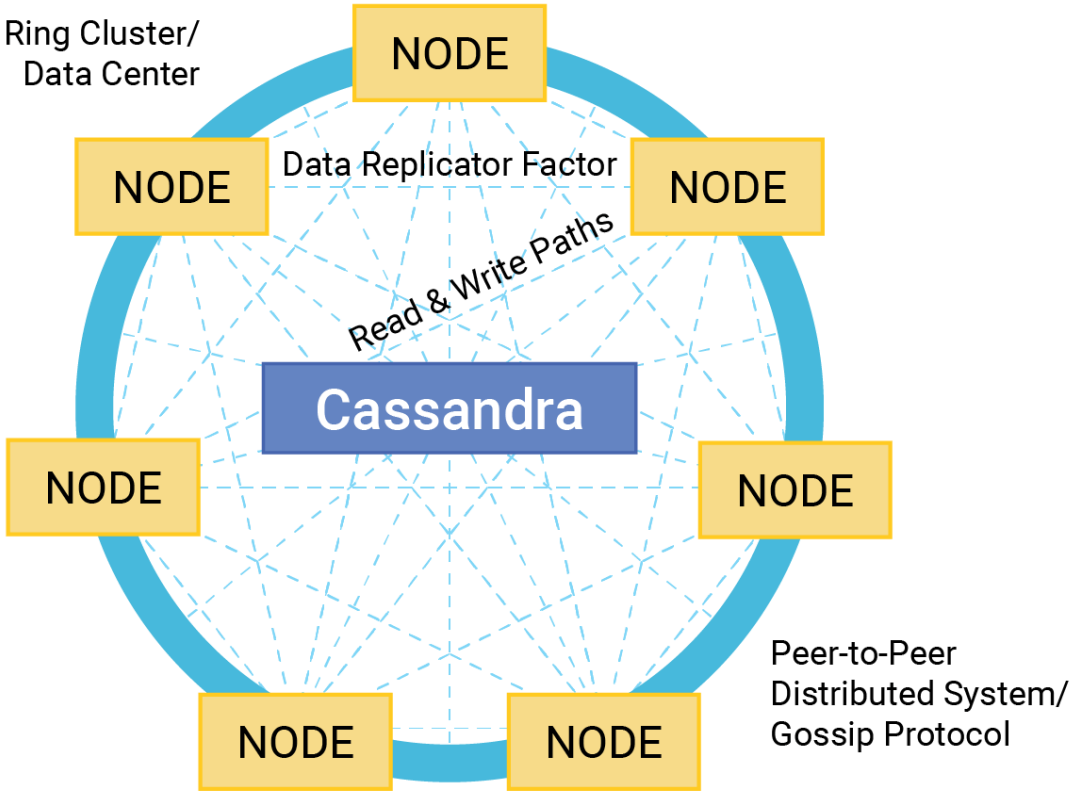
MongoDB



Cette architecture montre une application connectée à un mongos (routeur MongoDB), qui redirige les requêtes vers différents mongod (instances de données) répartis sur plusieurs machines. Le mongos agit comme une interface unique pour le client, simplifiant l'accès aux données distribuées sur plusieurs nœuds du cluster.

- **Architecture générale :**
  MongoDB utilise une architecture maître-esclave (ou réplication multi-maître) dans des configurations de réplication, et supporte le sharding pour la partition des données.
- **Schéma explicatif (conceptuel) :**
  - **Instance de MongoDB (mongod) :** Chaque instance gère des collections et des documents.
  - **Replica Set :** Un ensemble d'instances répliquant les mêmes données pour la haute disponibilité.
  - **Shard Cluster :** Les données sont partitionnées (shardées) sur plusieurs nœuds, chaque shard pouvant être un replica set.
- **Application pratique dans notre système :**
  En production, notre système pourrait déployer un cluster shardé pour répartir la charge, avec des replica sets assurant la redondance et la tolérance aux pannes.

Cassandra



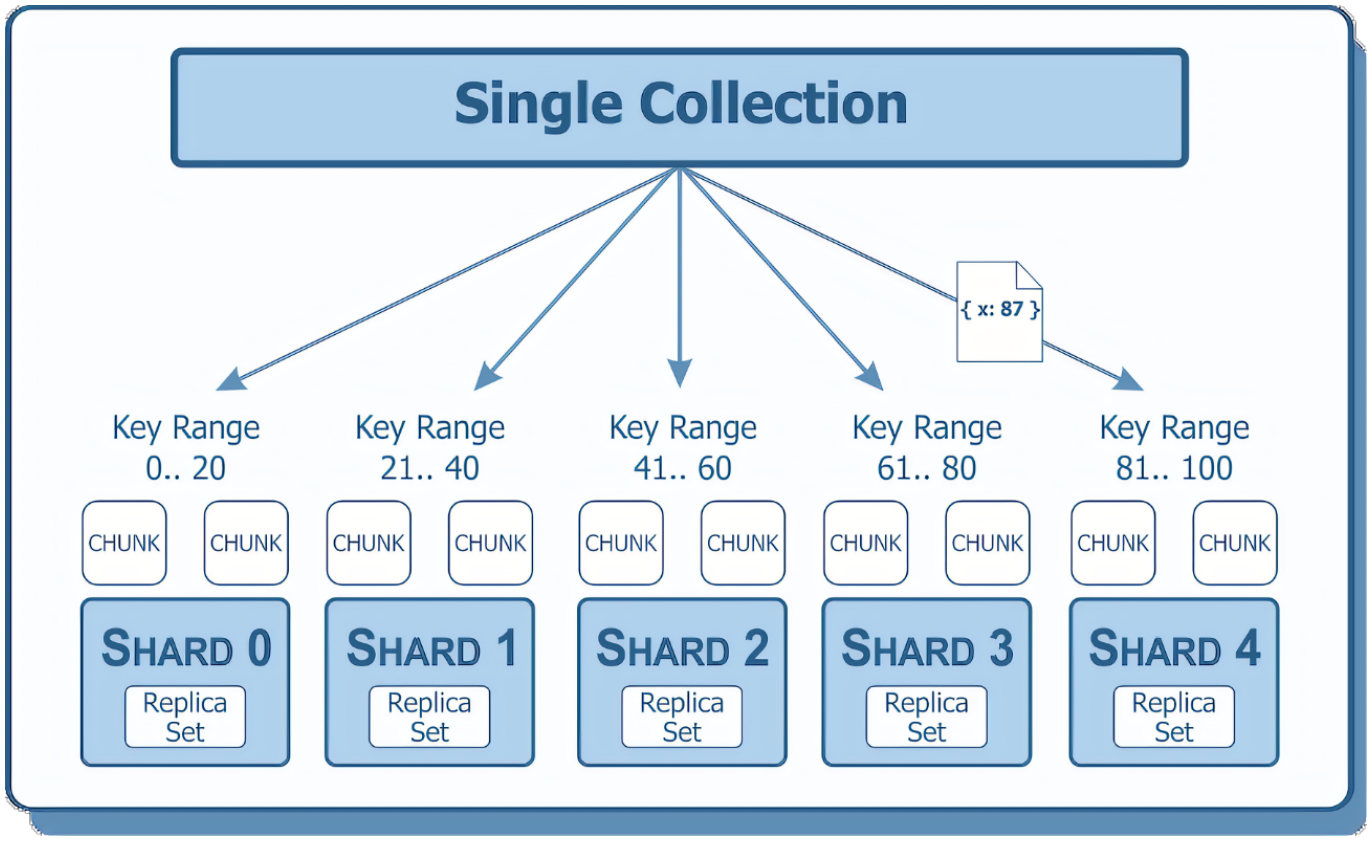
L'architecture de Cassandra repose sur un modèle en anneau (ring) sans maître. Chaque nœud possède les mêmes responsabilités (pas de nœud central), ce qui permet une haute disponibilité. Les nœuds communiquent entre eux en pair-à-pair et les données sont automatiquement réparties via un partitionnement basé sur les clés.

- **Architecture générale :**
  Cassandra adopte une architecture en peer-to-peer dans laquelle chaque nœud est égal et capable de gérer les lectures et écritures.
- **Schéma explicatif (conceptuel) :**
  - **Cluster :** Composé de multiples nœuds répartis sur différentes data centers.
  - **Partitionnement :** Les données sont réparties selon une clé de partition à l'aide d'un algorithme de hachage.
- **Application pratique dans notre système :**
  Un cluster Cassandra permettrait de distribuer les données de manière homogène et de bénéficier d'une très haute disponibilité, en particulier utile en cas de montée en charge.

---

4. Méthode de partitionnement (schémas expliqués)

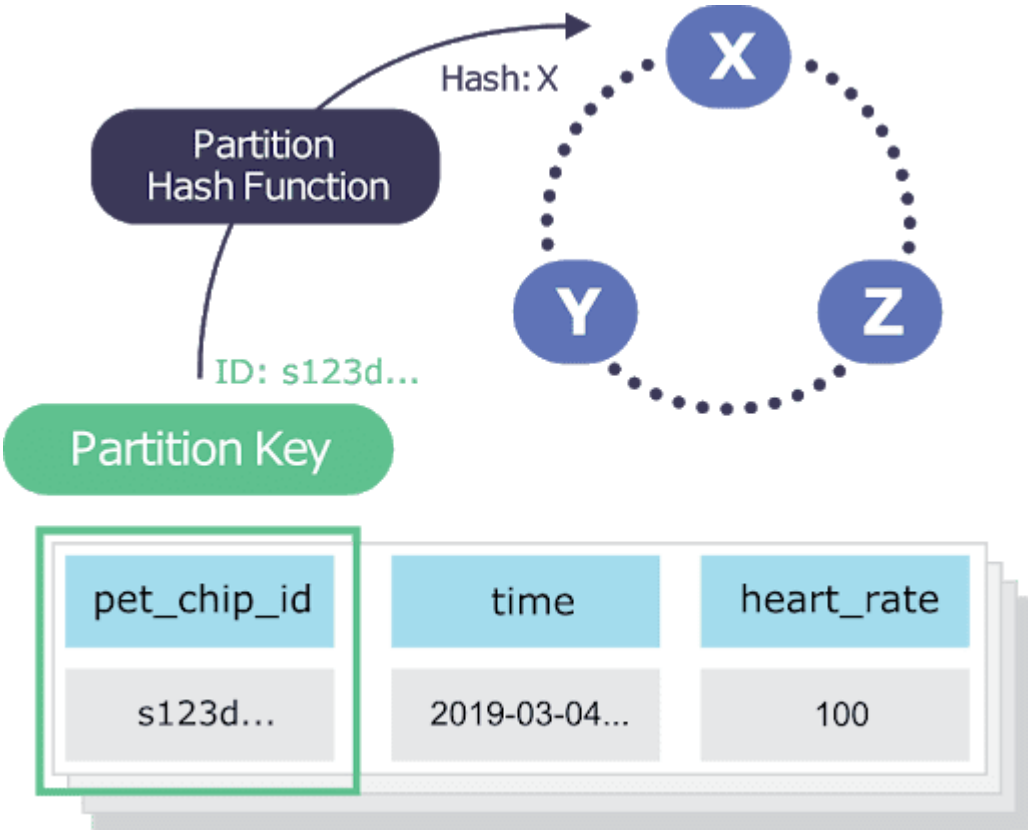
MongoDB



MongoDB utilise un partitionnement appelé sharding où chaque fragment (shard) contient une portion des données. Le mongos est responsable de router les requêtes vers le bon shard. Les documents sont distribués en fonction d'une clé de partition définie par l'utilisateur (shard key).

- **Partitionnement (Sharding) :**
  - **Principe :** Les données sont réparties sur plusieurs shards en utilisant une clé de sharding.
  - **Schéma conceptuel :**
    - **Shard Key :** Un champ ou une combinaison de champs sélectionné pour partitionner les documents.
    - **Mongos :** Le routeur qui distribue les requêtes vers les shards correspondants.
- **Application pratique dans notre système :**
  En choisissant une shard key judicieuse (par exemple, par région géographique pour des **Members** ou par catégories pour des **Books**), nous pouvons garantir une distribution équilibrée des données et améliorer les performances des requêtes.

Cassandra



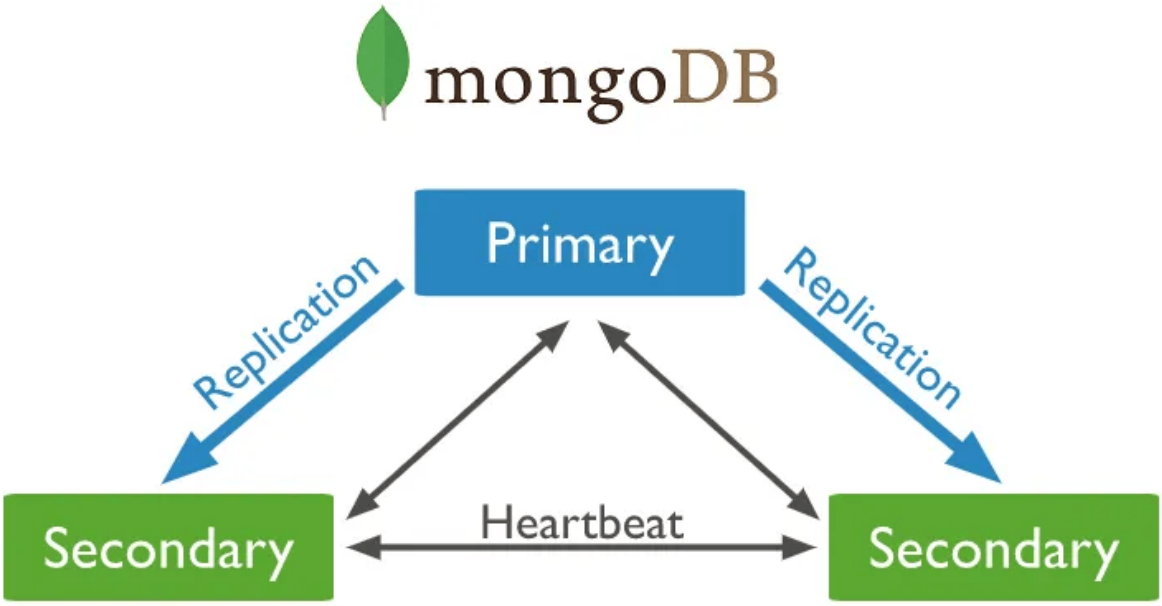
Cassandra applique un partitionnement automatique via une fonction de hachage sur la clé primaire. Les données sont divisées en token ranges et réparties sur les nœuds du ring. Cela garantit une distribution équilibrée des données et permet une extension horizontale fluide.

- **Partitionnement :**
  - **Principe :** Le partitionnement se fait à l'aide d'une clé de partition, sur laquelle Cassandra applique une fonction de hachage pour distribuer les données sur tous les nœuds.
  - **Schéma conceptuel :**
    - **Partition Key :** Détermine la distribution de chaque ligne dans le cluster.
    - **Token Ring :** Chaque nœud est responsable d'un segment du "ring" des tokens.
- **Application pratique dans notre système :**
  Pour des tables telles que la table des **Members**, la clé de partition peut être choisie de manière à optimiser l'accès et l'écriture, par exemple, en utilisant l'email comme clé de partition pour une recherche rapide.

---

5. Méthode de réplication (schémas expliqués)

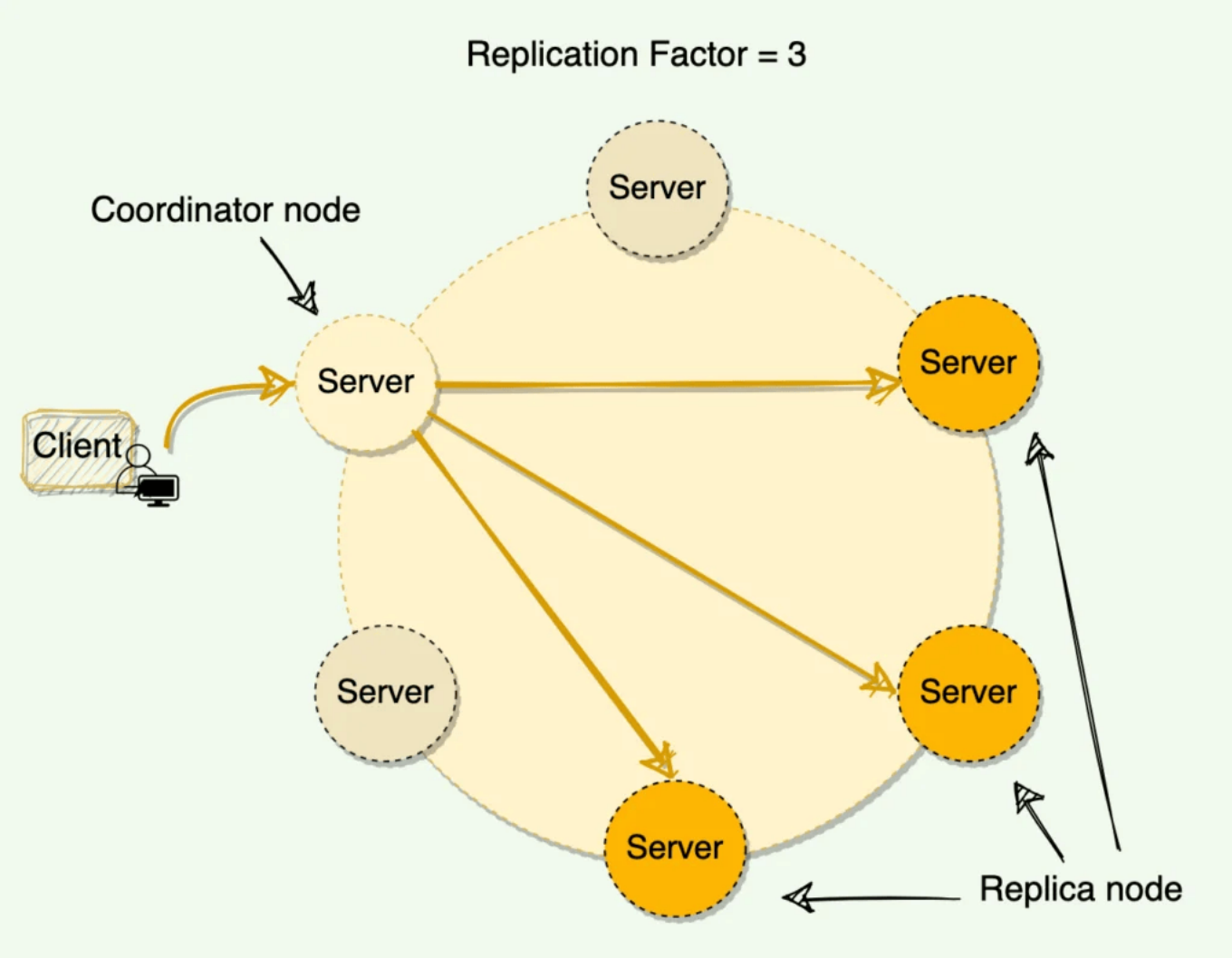
MongoDB



Chaque shard dans MongoDB peut être un replica set : un groupe de nœuds où un nœud est primaire (écriture) et les autres secondaires (lecture / failover). En cas de panne du primaire, un secondaire est élu automatiquement pour assurer la continuité de service.

- **Réplication :**
  - **Principe :** Utilisation de Replica Sets pour répliquer les données sur plusieurs serveurs.
  - **Schéma conceptuel :**
    - **Primary Node :** Responsable de toutes les écritures.
    - **Secondary Nodes :** Copient les opérations du primary pour assurer la redondance.
- **Application pratique dans notre système :**
  La réplication assure la haute disponibilité. En cas de panne du nœud primaire, l'un des secondaires peut être promu pour minimiser l'interruption du service.

Cassandra

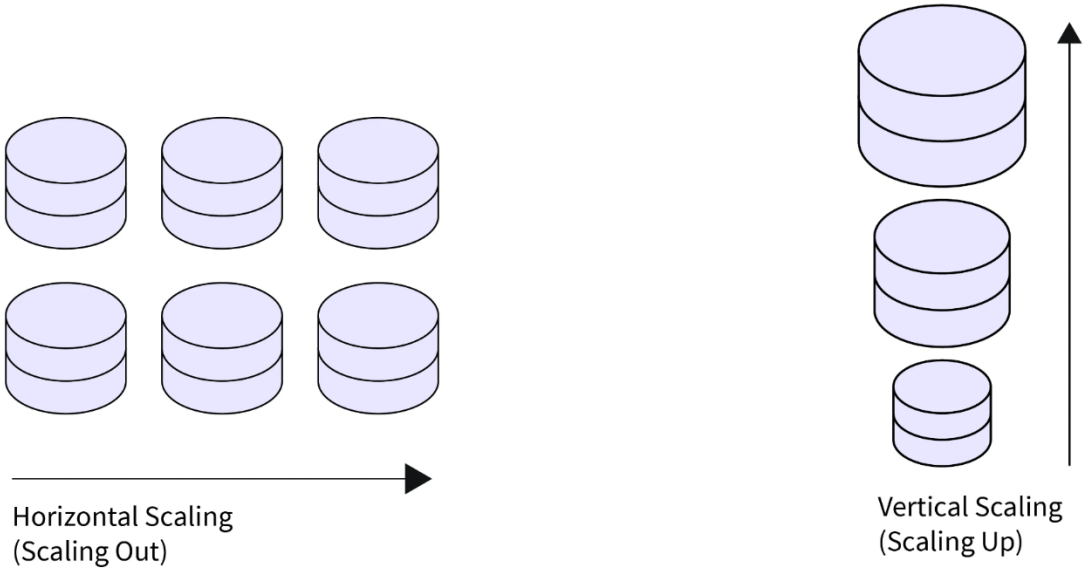


Cassandra assure une réplication uniforme des données sur plusieurs nœuds grâce à sa topologie en anneau. Chaque donnée est copiée sur plusieurs nœuds définis par le replication factor. Tous les nœuds sont égaux, ce qui permet un modèle sans point de défaillance unique.

- **Réplication :**
  - **Principe :** Chaque donnée est répliquée sur plusieurs nœuds selon une stratégie de réplication (ex. SimpleStrategy, NetworkTopologyStrategy).
  - **Schéma conceptuel :**
    - **Réplication factor :** Le nombre de nœuds sur lesquels chaque ligne de données est copiée.
- **Application pratique dans notre système :**
  En définissant un facteur de réplication approprié, Cassandra assure la disponibilité et la tolérance aux pannes, garantissant que la perte d'un ou plusieurs nœuds n'affecte pas l'accès aux données.

---

6. Montée en charge (schémas expliqués)



Ces figure illustre que MongoDB comme Cassandra sont capables de monter en charge horizontalement. Il suffit d'ajouter de nouveaux nœuds au cluster, et les données (ou leurs partitions) sont redistribuées automatiquement. Cela permet de traiter plus de requêtes, stocker plus de données, et améliorer la tolérance aux pannes.

MongoDB

- **Scalabilité horizontale (sharding) :**
  - **Principe :** Ajout de nouveaux nœuds pour distribuer la charge et le volume des données.
  - **Schéma conceptuel :**
    - **Shard Cluster :** Chaque shard contient une partie des données et fonctionne en replica set.
- **Application pratique dans notre système :**
  Lorsque le volume de données et le nombre de requêtes augmentent, nous pouvons ajouter des shards pour étendre les capacités de traitement sans interruption.

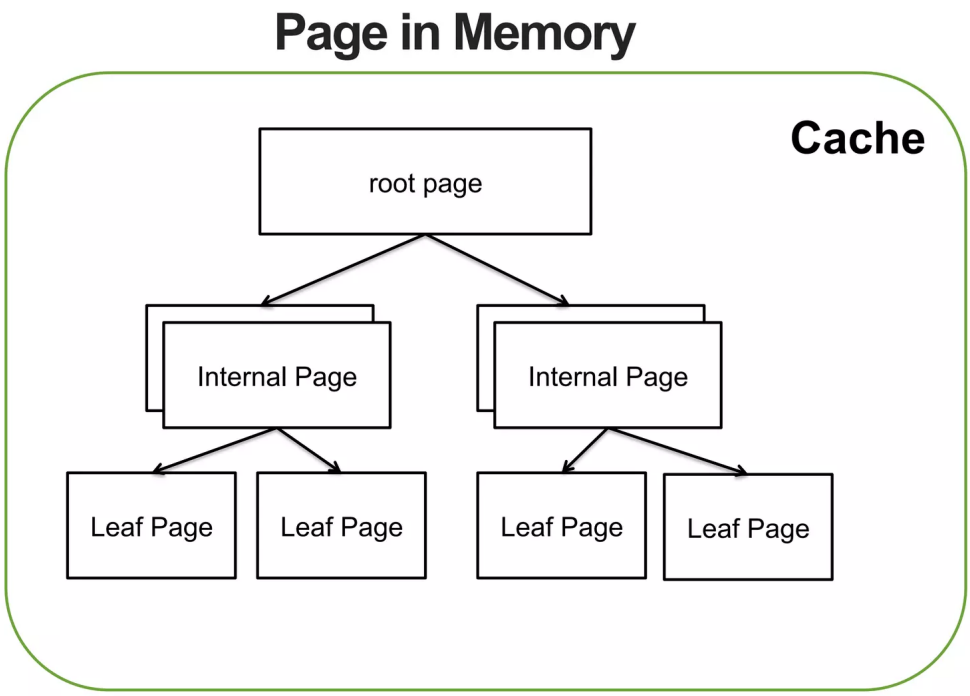
Cassandra

- **Scalabilité horizontale :**
  - **Principe :** Chaque nouveau nœud ajouté au cluster partage la charge de travail de manière égale grâce à un modèle peer-to-peer.
  - **Schéma conceptuel :**
    - **Cluster Expansion :** Ajout de nœuds, rééquilibrage automatique des données via la fonction de hachage.
- **Application pratique dans notre système :**
  Cassandra est conçu pour supporter l'ajout de nœuds de manière transparente, garantissant une montée en charge linéaire, essentielle pour des applications nécessitant une haute disponibilité et des écritures/lectures intensives.

---

7. Gestion du ou des caches mémoire (schémas expliqués)

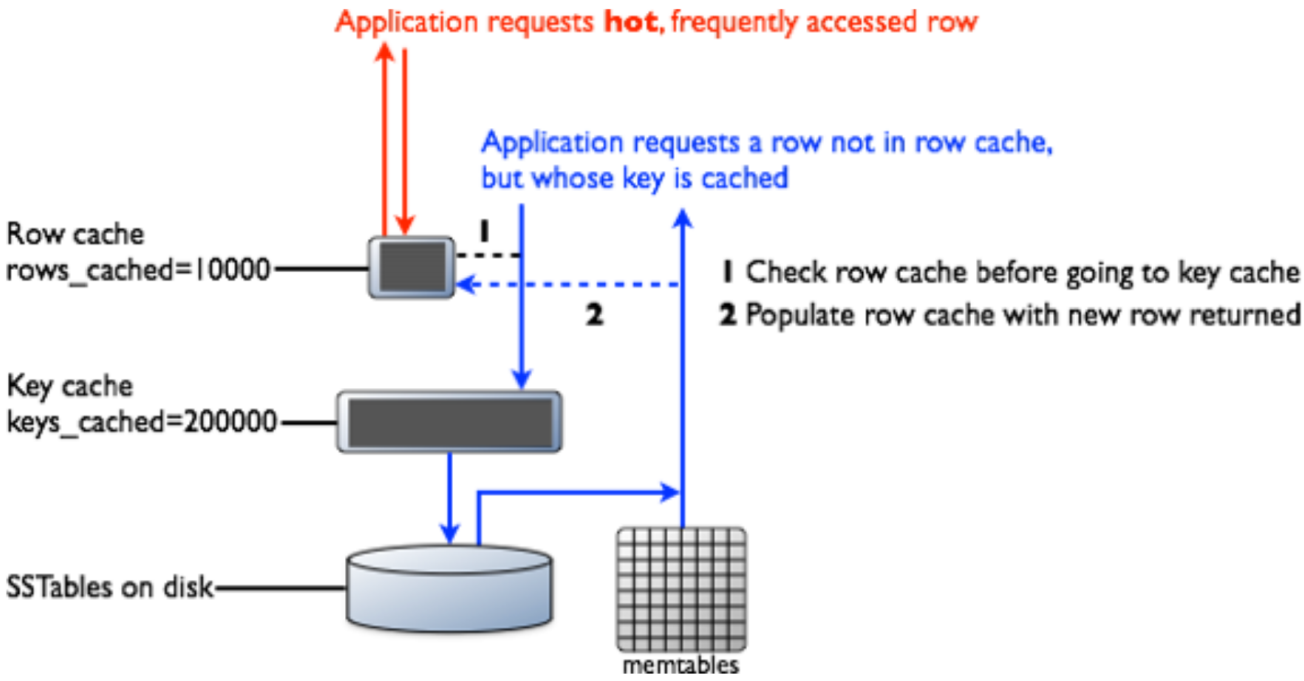
MongoDB



MongoDB utilise un moteur de stockage basé sur la mémoire (WiredTiger), qui garde les données récemment utilisées dans le cache mémoire (RAM). Cela accélère les lectures en évitant d'accéder au disque à chaque fois. La gestion du cache est automatique et adaptative.

- **Gestion du cache :**
  - **Principes :**
    - **WiredTiger Cache :** MongoDB utilise WiredTiger (son moteur de stockage par défaut) qui intègre un cache mémoire pour accélérer les opérations de lecture/écriture.
    - **OS Cache :** MongoDB exploite également la mémoire disponible du système d'exploitation pour stocker les pages de données.
  - **Schéma conceptuel :**
    - **Cache Memory Layer :** Combine le cache spécifique à WiredTiger et la mémoire de l'OS pour réduire l'accès disque.
- **Application pratique dans notre système :**
  Une configuration adéquate (taille du cache, paramètres de WiredTiger) peut améliorer significativement la performance des requêtes sur les collections telles que **Members** ou **Publishers**.

Cassandra



Cassandra gère plusieurs types de caches : key cache (index des clés) et row cache (lignes complètes), pour accélérer les lectures fréquentes. Ces caches sont maintenus en mémoire pour éviter les accès disques répétitifs, avec des stratégies de nettoyage en fonction de la pression mémoire.

- **Gestion du cache :**
  - **Principes :**
    - **Key Cache :** Mémorise les clés des partitions pour accélérer la recherche.
    - **Row Cache :** (Optionnel) Cache de lignes entières pour des lectures ultra-rapides sur des données fréquemment consultées.
  - **Schéma conceptuel :**
    - **Caches spécifiques :** Chaque nœud comporte plusieurs couches de cache pour minimiser l'accès au disque.
- **Application pratique dans notre système :**
  L'optimisation des caches dans Cassandra permet de répondre efficacement aux requêtes de lecture lourdes, tout en réduisant la latence, ce qui est crucial dans un environnement distribué.

Ces concepts, bien compris et correctement mis en œuvre, permettent de concevoir une architecture NoSQL robuste et évolutive pour notre système. Chaque moteur possède ses propres avantages et mécanismes internes pour assurer la haute disponibilité, la scalabilité et la performance des opérations de lecture et d'écriture.