

Introduction à Apache Cassandra

Une exploration approfondie d'Apache Cassandra, un système de gestion de base de données NoSQL distribué pour les professionnels.

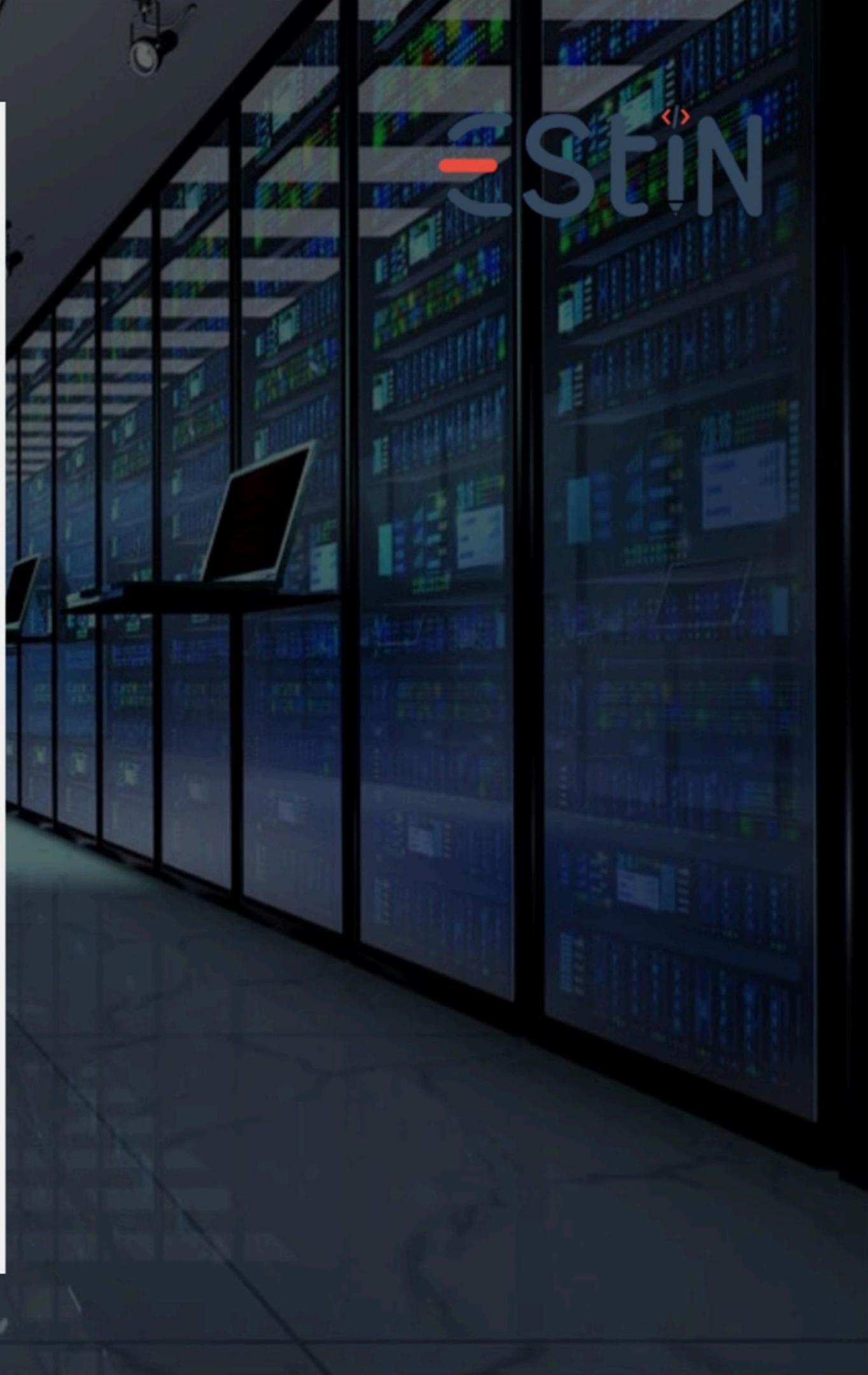
Membres :

- Slimani Wahid
- Benali Amine
- Lallouche Abdelghani
- Koudia yacine

Superviseurs :

- PR Sebaa abderrazak
- Mme Haroune asma

[Apache cassandra](#)





SGBD NoSQL

Apache Cassandra : Guide Complet

Un aperçu approfondi d'Apache Cassandra, un système de gestion de base de données distribué et évolutif

Sommaire

Un aperçu des sujets clés de la présentation

01 Introduction

Présentation des objectifs de la présentation et des thèmes abordés.

02 Architecture et Fonctionnement

Analyse de l'architecture des systèmes et de leur fonctionnement interne.

03 Modèle de Données

Exploration des modèles de données utilisés dans les systèmes de gestion de bases de données.

04 Caractéristiques Principales

Examen des principales caractéristiques qui définissent les systèmes de gestion de bases de données.

05 ACID vs BASE

Comparaison des propriétés ACID et BASE en matière de gestion de données.

06 Tolérance aux Pannes

Discussion sur les mécanismes de tolérance aux pannes dans les systèmes de gestion de bases de données.

07 Scalabilité et Performance

Analyse de la scalabilité et des performances des systèmes de gestion de bases de données.

08 Sécurité et API REST

Évaluation des aspects de sécurité et de l'utilisation des API REST dans la gestion des données.

09 Avantages et Inconvénients

Analyse des avantages et des inconvénients des différents systèmes de gestion de bases de données.

10 Comparaison avec d'autres SGBD

Comparaison des systèmes de gestion de bases de données abordés avec d'autres sur le marché.

11 Cas d'Usage dans le Monde Réel

Étude des cas d'utilisation des systèmes de gestion de bases de données dans divers secteurs.

12 Déploiement et Bonnes Pratiques

Conseils sur le déploiement et les meilleures pratiques pour les systèmes de gestion de bases de données.

13 Démonstration

Démonstration des concepts abordés à travers un exemple pratique.

14 Conclusion et Perspectives

Résumé des points clés abordés et discussion des perspectives d'avenir.

Introduction to Apache Cassandra

Une base de données NoSQL performante et fiable

● Qu'est-ce qu'Apache Cassandra ?

Apache Cassandra est un Système de Gestion de Base de Données (SGBD) NoSQL distribué, conçu pour gérer d'énormes volumes de données répartis sur plusieurs serveurs. Ce système garantit une haute disponibilité et une tolérance aux pannes exceptionnelle, ce qui en fait un choix idéal pour les applications critiques.

● Historique et Origine

Développée par Facebook en 2008 pour améliorer la recherche dans les messages, Apache Cassandra est devenue un projet open source sous la fondation Apache en 2009. Son nom est inspiré de la prophétesse de la mythologie grecque, symbolisant la prévision et la connaissance.

● Adoption Industrielle

Apache Cassandra est largement adoptée par des entreprises de renom telles que Netflix, qui utilise la base de données pour gérer les données de visionnage de plus de 200 millions d'utilisateurs, avec jusqu'à 1 million d'écritures par seconde. D'autres exemples incluent Apple pour la gestion des données iCloud, Uber pour le stockage des données de trajets, et Spotify pour la gestion des playlists et des historiques d'écoute.

Architecture et Fonctionnement de Cassandra

Exploration des caractéristiques fondamentales et de l'architecture décentralisée

Wide-Column Store

Système de gestion de base de données orienté colonnes, optimisé pour le stockage flexible et les charges distribuées massives.

Architecture Décentralisée

Tous les nœuds sont égaux dans une configuration peer-to-peer, éliminant ainsi le point unique de défaillance.

Distribution des Données

Utilisation d'un algorithme de hachage pour distribuer les données entre les nœuds de manière équilibrée.

Communication Inter-Nœuds

Les nœuds communiquent entre eux via le protocole Gossip, permettant une mise à jour et une synchronisation efficaces.

Cassandra Query Language (CQL)

Langage d'interrogation similaire à SQL, adapté à l'architecture distribuée, permettant des opérations sur les données.

Modèle de Données dans Cassandra

Hiérarchie et Structure

Cluster

Ensemble de noeuds Cassandra qui forment la base de l'architecture de données.

Partition

Ensemble de lignes identifiées par une clé de partition, optimisant l'accès aux données.

Keyspace

Équivalent d'une base de données en SQL, définit la stratégie de réPLICATION et le facteur de réPLICATION.

Colonne

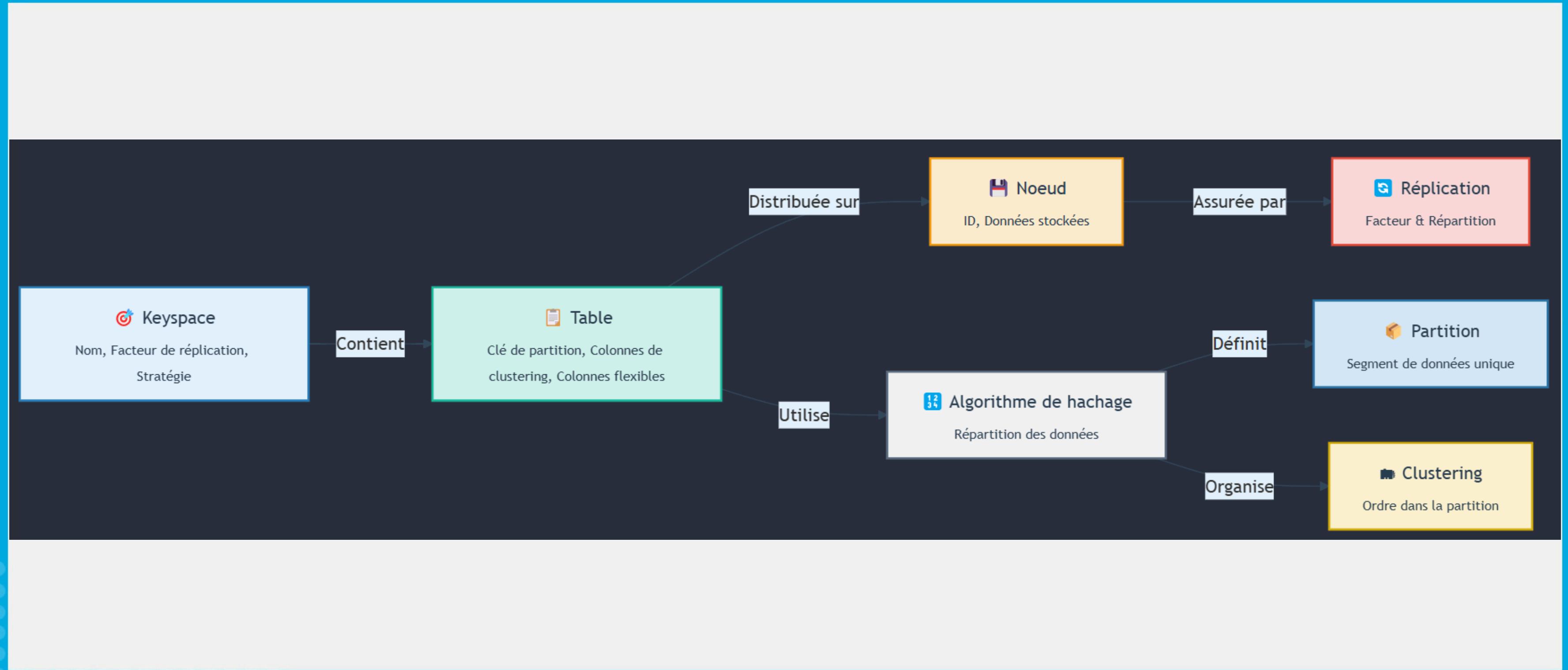
Unité fondamentale de stockage des données, regroupée en familles de colonnes pour un accès rapide.

Table

Collection de données organisées en lignes et colonnes, où chaque ligne peut contenir un nombre variable de colonnes.

Modèle de Données dans Cassandra

Hiérarchie et Structure



Caractéristiques Principales (1/2)

Exploration des principes clés du système de base de données Cassandra

01 Théorème CAP et Positionnement

Le théorème CAP stipule qu'il est impossible pour un système distribué de garantir simultanément la cohérence, la disponibilité et la tolérance au partitionnement. Cassandra se positionne sur le plan CAP en favorisant la disponibilité et la tolérance au partitionnement.

02 Cohérence, Availability (Disponibilité), Partition Tolerance (Tolérance au partitionnement)

Ces trois caractéristiques sont fondamentales pour évaluer les performances des systèmes de gestion de bases de données. Cassandra offre une disponibilité élevée et une tolérance au partitionnement au détriment d'une cohérence forte.

03 Cassandra priviliege AP (Disponibilité et Tolérance au partitionnement)

Cassandra est conçue pour être un système hautement disponible, même en cas de partitionnement du réseau. Cela signifie que le système peut continuer à fonctionner et à répondre aux demandes des utilisateurs malgré des interruptions.

04 Sacrifie la cohérence forte au profit de la cohérence éventuelle

Cassandra adopte un modèle de cohérence éventuelle, permettant aux données de se synchroniser sur plusieurs nœuds au fil du temps, plutôt que de nécessiter une cohérence immédiate.

05 Modèle de Cohérence

Le modèle de cohérence de Cassandra est basé sur le principe de BASE, qui signifie Basically Available, Soft State, Eventual Consistency. Cela permet une flexibilité dans la gestion des requêtes et des données.

06 BASE (Basically Available, Soft State, Eventual Consistency)

Le modèle BASE est une approche alternative au modèle ACID traditionnel, offrant une disponibilité accrue et une gestion plus souple des données, particulièrement adapté aux systèmes distribués comme Cassandra.

07 Niveaux de cohérence ajustables par requête

Cassandra permet d'ajuster les niveaux de cohérence lors des lectures et écritures via des paramètres tels que ONE, QUORUM, ALL, ainsi que LOCAL_QUORUM et EACH_QUORUM pour les configurations multi-datacenter.

08 Exemple de requête avec cohérence QUORUM

Un exemple de requête illustrant l'utilisation du niveau de cohérence QUORUM serait : `SELECT * FROM users WHERE user_id = 123e4567-e89b-12d3-a456-426614174000 CONSISTENCY QUORUM;` Cela démontre comment les utilisateurs peuvent spécifier le niveau de cohérence souhaité pour leurs requêtes.

Caractéristiques Principales (2/2)

01 Tolérance aux Pannes

La tolérance aux pannes est cruciale pour assurer la continuité des services. Grâce à la réPLICATION automatique des données, les systèmes peuvent continuer à fonctionner même en cas de défaillance d'un ou plusieurs nœuds. Cela est renforcé par la détection automatique des nœuds défaillants, qui permet d'identifier rapidement les problèmes et de prendre les mesures nécessaires pour minimiser les interruptions.

02 Auto-réparation

Les systèmes modernes intègrent des mécanismes d'auto-réparation tels que le hinted handoff et le read repair. Ces techniques garantissent que les données sont toujours à jour et disponibles, même en cas de panne, permettant ainsi une récupération rapide et efficace des informations sans intervention manuelle.

03 Multi-datacenter pour la résilience géographique

La capacité de déployer des services sur plusieurs datacenters renforce la résilience géographique. Cela signifie qu'une région peut continuer à fonctionner même si une autre est affectée par un désastre naturel ou une panne de service, assurant ainsi la disponibilité des données à l'échelle mondiale.

04 Scalabilité et Performance

La scalabilité est fondamentale pour s'adapter à la croissance des données et des utilisateurs. La scalabilité horizontale linéaire permet d'ajouter des nœuds sans compromettre les performances, garantissant ainsi que le système peut croître de manière fluide et efficace.

05 Élasticité du système

L'élasticité désigne la capacité d'ajouter ou de retirer des nœuds sans interruption de service. Cela permet aux entreprises de s'ajuster dynamiquement aux variations de la charge de travail, assurant ainsi une répartition optimale des ressources et une performance constante.

06 Performances prévisibles

Les systèmes doivent offrir des performances prévisibles, indépendamment de la taille du cluster. Cela signifie que même avec une augmentation du nombre de nœuds, les performances restent constantes, ce qui est essentiel pour maintenir la satisfaction des utilisateurs.

07 Distribution équilibrée via partitionnement par token

La distribution des données est optimisée grâce à des techniques de partitionnement par token, qui assurent que les données sont réparties équitablement sur tous les nœuds. Cela évite les goulets d'étranglement et améliore l'efficacité globale du système.

08 Optimisation pour l'écriture

Les systèmes modernes sont souvent optimisés pour l'écriture plutôt que pour la lecture, ce qui est crucial dans des environnements à forte charge d'écriture. Cela garantit que les opérations d'écriture sont réalisées rapidement et efficacement, améliorant ainsi la réactivité du système.

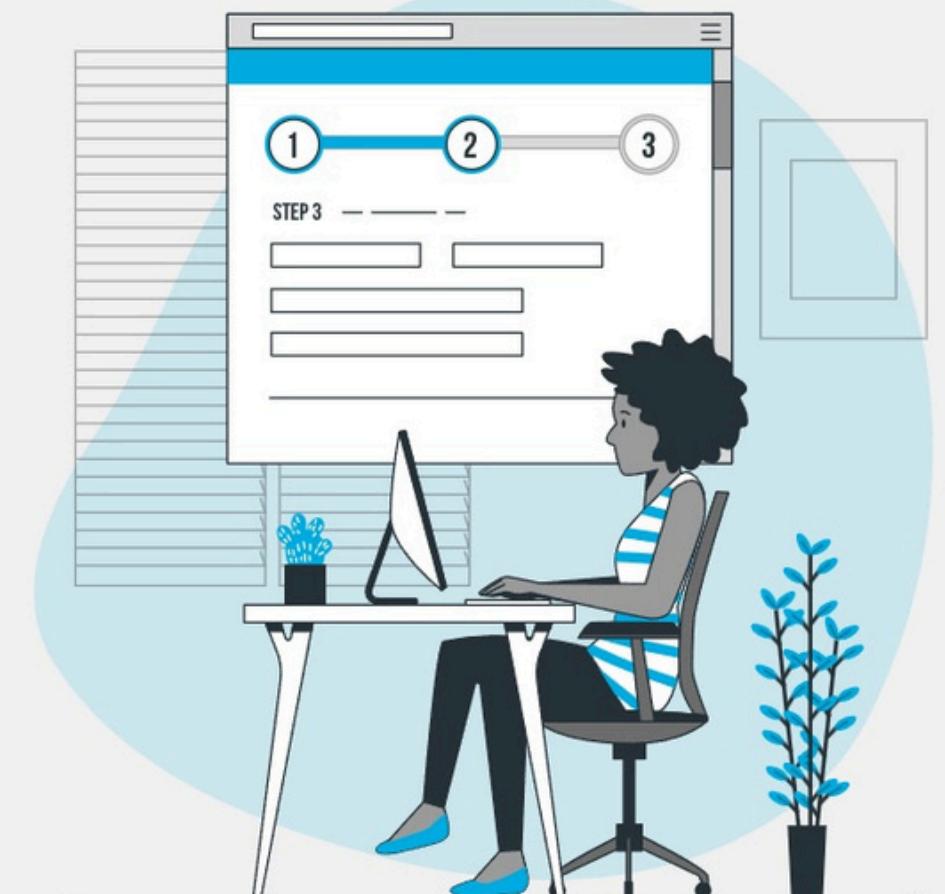
Modèles ACID vs BASE

Analyse comparative des modèles de transactions ACID et BASE

01

Modèle ACID

- Atomicité : Une transaction est entièrement exécutée ou annulée
- Cohérence : La base passe d'un état valide à un autre
- Isolation : Une transaction en cours n'affecte pas les autres
- Durabilité : Une transaction validée persiste même après une panne
- Cas d'utilisation : Virement bancaire (débit/crédit simultanés)



02

Modèle BASE

- Basically Available : Le système reste accessible même en cas de panne partielle
- Soft state : L'état des données peut évoluer sans intervention directe
- Eventual consistency : Les mises à jour se propagent progressivement
- Cas d'utilisation : Post sur réseau social (délai d'affichage acceptable)

Tolérance aux Pannes

Stratégies et Mécanismes pour Assurer la Continuité des Données



01

Mécanismes de RéPLICATION

La réPLICATION est essentielle pour garantir la disponibilité des données. Le facteur de réPLICATION détermine combien de copies de chaque donnée sont stockées dans le système, ce qui aide à prévenir la perte de données en cas de panne. Deux stratégies principales de réPLICATION existent : la SimpleStrategy, qui convient aux environnements à un seul datacenter, et la NetworkTopologyStrategy, idéale pour les configurations multi-datacenter.

02

Gestion des Nœuds Défaillants

La gestion des nœuds défaillants est cruciale pour maintenir l'intégrité des données. Le Hinted Handoff permet de stocker temporairement les écritures destinées à un nœud indisponible, garantissant ainsi que les données ne sont pas perdues. Le Read Repair, quant à lui, s'assure que les données incohérentes sont réparées lors des lectures. Enfin, l'Anti-Entropy Node Repair effectue une synchronisation périodique des répliques pour maintenir la cohérence des données.

03

Exemple Concret

Un exemple illustratif de la tolérance aux pannes est le service Apple iCloud, où les photos sont répliquées sur trois nœuds géographiquement distants. Cela garantit que même si un datacenter rencontre une panne, les données restent accessibles aux utilisateurs, démontrant l'importance de la réPLICATION et de la gestion des pannes.

Sécurité et API REST

01 Mécanismes de Sécurité

La sécurité des API REST repose sur plusieurs mécanismes fondamentaux. Cela inclut l'authentification et le chiffrement, qui garantissent que seules les parties autorisées peuvent accéder aux ressources de l'API et que les données échangées restent confidentielles et intègres.

02 Authentification

L'authentification est un processus essentiel qui vérifie l'identité des utilisateurs. Cela peut être réalisé via l'intégration de systèmes d'authentification tels que LDAP et Kerberos, ou par des méthodes internes spécifiques à l'application. Ces approches garantissent que seuls les utilisateurs légitimes peuvent accéder à l'API.

03 Chiffrement

Le chiffrement protège les données pendant leur transmission et leur stockage. L'utilisation de SSL/TLS pour les communications entre le client et le serveur, ainsi que JMX pour les communications entre nœuds, assure un échange sécurisé. Pour les données au repos, le chiffrement AES-256 est recommandé, garantissant ainsi leur sécurité même lorsqu'elles ne sont pas en transit.

04 Gestion des Accès

La gestion des accès définit qui peut faire quoi au sein de l'API. Le contrôle d'accès basé sur les rôles (RBAC) permet de gérer les permissions de manière structurée. Des permissions granulaires, telles que CREATE, ALTER, DROP, SELECT, et MODIFY, assurent que seuls les utilisateurs habilités peuvent effectuer des actions spécifiques.

05 Exemple de RBAC

Un exemple concret de gestion des rôles est la création d'un rôle d'analyste avec des permissions spécifiques. Par exemple, la commande 'CREATE ROLE analyst WITH PASSWORD = 'password' AND LOGIN = true;' permet de créer un rôle et d'accorder l'accès de sélection sur un espace de clés nommé 'metrics'. Ceci illustre comment les rôles peuvent être gérés de manière sécurisée.

06 API REST & Intégration

Les API REST facilitent l'intégration avec divers langages de programmation tels que Java, Python et Node.js. Elles permettent une communication fluide entre le client et le serveur, tout en respectant les normes de sécurité établies. L'utilisation de ces API dans des écosystèmes Big Data, comme Spark et Hadoop, renforce leur applicabilité dans des environnements complexes.

07 Connecteurs pour Langages Populaires

Des connecteurs sont disponibles pour des langages de programmation populaires, permettant aux développeurs d'intégrer facilement des API REST dans leurs applications. Cela favorise une adoption plus rapide et une meilleure interopérabilité entre différents systèmes.

08 Compatibilité avec Écosystèmes Big Data

Les API REST sont conçues pour être compatibles avec des écosystèmes Big Data tels que Spark et Hadoop, ce qui permet une meilleure exploitation des données à grande échelle tout en maintenant des standards de sécurité élevés.

ANALYSE SWOT DES CARACTÉRISTIQUES DU SYSTÈME

Forces et limites des systèmes modernes

S

SCALABILITÉ LINÉAIRE

Permet l'ajustement facile des ressources en fonction de la demande, garantissant une utilisation optimale des capacités.

W

HAUTE DISPONIBILITÉ

Assure un accès constant aux données et services, minimisant les temps d'arrêt et améliorant la satisfaction des utilisateurs.

O

PERFORMANCES ÉLEVÉES POUR LES OPÉRATIONS D'ÉCRITURE

Optimise les processus d'écriture, rendant le système adapté aux applications nécessitant des mises à jour fréquentes.

T

FLEXIBILITÉ POUR LES DONNÉES STRUCTURÉES ET NON STRUCTURÉES

Permet de gérer divers types de données sans compromettre l'intégrité ni la performance, offrant une grande adaptabilité.

COMPARAISON DES SGBD

Caractéristiques clés des systèmes de base de données

CARACTÉRISTIQUE	APACHE CASSANDRA	MONGODB (DOCUMENT STORE)	COCKROACHDB (SQL)	AMAZON DYNAMODB (KEY-VALUE)
Modèle de données	Wide-column	Document	Relational	Key-Value
Scalabilité	Horizontale	Horizontale	Horizontale	Horizontale
Cohérence	Tunable	Strong/Eventual	Strong	Eventual
Performance	Élevée	Élevée	Équilibrée	Élevée
Complexité	Modérée	Modérée	Élevée	Faible

Cas d'Usage dans le Monde Réel (1/2)

Analyse des solutions de stockage de données innovantes



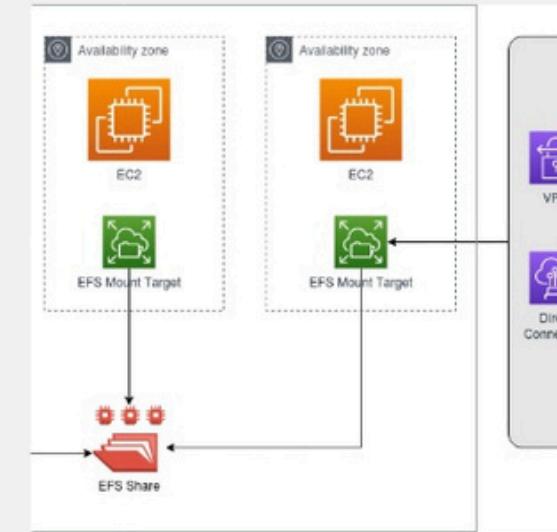
Utilisation de Netflix

Netflix utilise un système de stockage sophistiqué pour gérer les données de visionnage et fournir des recommandations personnalisées à ses utilisateurs. Cela



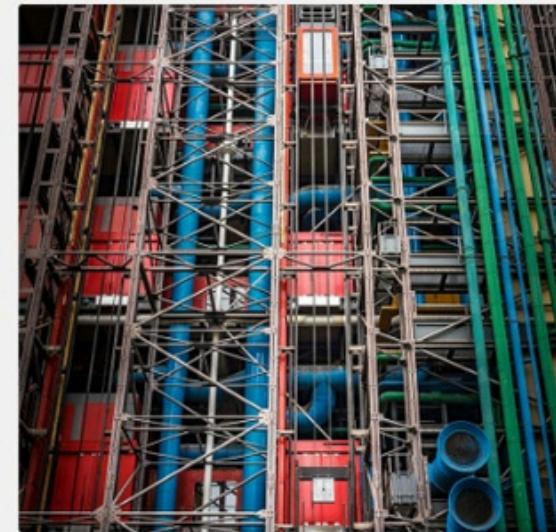
Volume d'écritures chez Netflix

Le volume d'écritures atteint un impressionnant 1 million par seconde, montrant l'échelle massive et l'efficacité de leur infrastructure de



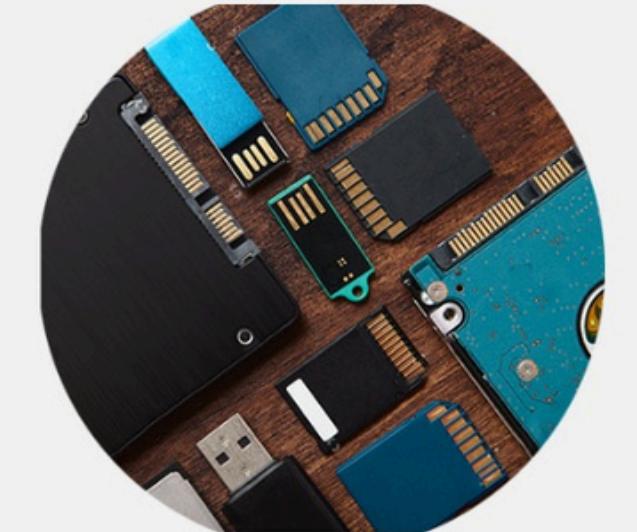
Configuration de cluster chez Netflix

Netflix dispose de plus de 2500 nœuds répartis sur 3 régions AWS, ce qui garantit une distribution efficace et une tolérance aux pannes.



Avantages pour Netflix

Avec une disponibilité de 99.999%, Netflix assure une expérience utilisateur continue, même en cas de panne régionale, grâce à sa conception robuste.



Utilisation d'Apple

Apple utilise une configuration de stockage pour iCloud, qui englobe les photos, documents et sauvegardes, assurant une gestion fluide des données pour ses

Déploiement et Bonnes Pratiques (2/2)

Optimisation et Surveillance

Modélisation des Données

La modélisation des données est cruciale pour optimiser les performances des requêtes. Elle comprend des techniques telles que la dénormalisation, qui privilégie une conception orientée requêtes afin d'améliorer l'accès aux données.

Dénormalisation : conception orientée requêtes

La dénormalisation consiste à réduire la complexité des requêtes en intégrant des données redondantes. Cela permet des accès plus rapides mais nécessite une gestion attentive lors des mises à jour.

Clés de partition : distribution équilibrée

Le choix des clés de partition est essentiel pour garantir une distribution équilibrée des données, minimisant ainsi les risques de 'hot spots' qui peuvent ralentir les performances.

Collections : utilisation judicieuse des listes, ensembles et maps

Utiliser des collections comme les listes, ensembles et maps de manière judicieuse permet de gérer efficacement des ensembles de données variés et d'optimiser les requêtes associées.

Tables Matérialisées : pour optimiser les requêtes

Les tables matérialisées sont des outils puissants pour améliorer les performances des requêtes, en permettant de stocker des résultats pré-calculés et d'accélérer l'accès aux données.

Pièges à Éviter

Il est important d'être conscient des pièges potentiels lors de la conception de la base de données, tels que les choix de clés de partition inappropriés qui peuvent entraîner des 'hot spots'.

Hot spots : mauvais choix de clés de partition

Un mauvais choix de clés de partition peut provoquer des 'hot spots', où certaines partitions sont surchargées, affectant la performance globale du système.

Tombstones excessifs : opérations de suppression massives

Les tombstones excessifs, résultant d'opérations de suppression massives, peuvent causer des problèmes de performance et compliquer la gestion des données.

Requêtes sans partition key : ALLOW FILTERING coûteux

Les requêtes qui n'utilisent pas de clés de partition et qui nécessitent le filtrage sont coûteuses en termes de performances et doivent être évitées.

Lecture avant écriture : anti-pattern à éviter

L'anti-pattern de la lecture avant l'écriture peut entraîner des incohérences dans les données et doit être évité pour maintenir l'intégrité des données.

Outils de Monitoring

L'utilisation d'outils de monitoring comme Prometheus et Grafana permet de suivre les métriques de performance, tandis que DataStax OpsCenter et nodetool aident à l'administration et au diagnostic.

Prometheus + Grafana pour métriques

Prometheus et Grafana offrent une solution complète pour la surveillance des métriques, permettant une visualisation claire des performances de la base de données.

DataStax OpsCenter pour administration

DataStax OpsCenter fournit une interface graphique pour la gestion et l'administration des bases de données, facilitant les tâches d'administration courantes.

nodetool pour diagnostic

L'outil nodetool est essentiel pour le diagnostic des bases de données, permettant aux administrateurs d'obtenir des informations détaillées sur l'état des nœuds et des performances.

CONCLUSION ET PERSPECTIVES SUR APACHE CASSANDRA

Analyse des performances et futur de la base de données

■ PUISSANCE ET SCALABILITÉ

Apache Cassandra est une base de données conçue pour offrir scalabilité et haute disponibilité.

■ UTILISATION DANS LE TEMPS RÉEL

Elle est largement utilisée dans des secteurs nécessitant un traitement des données en temps réel.

■ AMÉLIORATION DES MODÈLES DE COHÉRENCE

Les développements futurs se concentrent sur l'amélioration des modèles de cohérence.

■ INTÉGRATION DE L'APPRENTISSAGE AUTOMATIQUE

L'intégration des capacités d'apprentissage automatique est envisagée pour enrichir les fonctionnalités.

■ ÉVOLUTION GRÂCE À LA COMMUNAUTÉ

Cassandra évolue grâce aux contributions de la communauté open-source.

■ PERTINENCE À L'ÈRE DU BIG DATA

Cette évolution garantit la pertinence de Cassandra dans l'ère du big data.

Annexes : Ressources et Documentation

01 Ressources d'Apprentissage

Accédez à des ressources d'apprentissage essentielles pour maîtriser Cassandra. La documentation officielle d'Apache Cassandra offre une base solide pour comprendre les concepts et les fonctionnalités. Pour des ressources supplémentaires, Planet Cassandra et The Cassandra Cluster sont d'excellentes plateformes pour les articles, vidéos et tutoriels.

02 Documentation officielle

La documentation officielle d'Apache Cassandra, accessible à l'adresse <https://cassandra.apache.org/doc/latest/>, est une ressource incontournable pour les développeurs et les administrateurs cherchant à approfondir leur compréhension du système de gestion de base de données.

03 Planet Cassandra

Planet Cassandra, disponible à <https://www.datastax.com/learn>, est une plateforme complète qui regroupe une multitude de ressources et d'articles sur Cassandra, permettant aux utilisateurs de se tenir à jour avec les dernières nouvelles et meilleures pratiques.

04 The Cassandra Cluster

Le site The Cassandra Cluster, accessible à <https://thecassandracluster.com/>, propose des articles approfondis et des études de cas qui aident les utilisateurs à mieux comprendre les applications pratiques de Cassandra.

05 Outils et Utilitaires

Utilisez des outils dédiés pour optimiser vos performances et faciliter la gestion de vos bases de données. Des outils tels que Cassandra Stress Tool pour le benchmarking, cqlsh pour l'interaction CQL, et DataStax DevCenter comme IDE, sont essentiels.

06 Cassandra Stress Tool

Le Cassandra Stress Tool est un outil de benchmarking qui permet de mesurer les performances de votre cluster Cassandra, essentiel pour les tests de charge et l'optimisation des performances.

07 cqlsh

cqlsh est le shell interactif pour Cassandra Query Language, offrant une interface simple pour interagir avec votre base de données Cassandra.

08 DataStax DevCenter

DataStax DevCenter est un environnement de développement intégré (IDE) qui facilite le développement et le déploiement d'applications Cassandra, permettant aux développeurs d'écrire, tester et exécuter des requêtes CQL efficacement.

09 Cassandra Reaper

Cassandra Reaper est un outil de maintenance et de réparation pour Cassandra, permettant la gestion des opérations de nettoyage et d'optimisation de votre cluster.

10 Modèles de Données Spécifiques

Découvrez des modèles de données adaptés à différents secteurs. Des exemples de modélisation pour l'e-commerce, l'IoT et les applications sociales sont présentés pour aider les utilisateurs à concevoir des bases de données efficaces.

11 Exemples de modélisation pour e-commerce

Des modèles de données spécifiques à l'e-commerce peuvent inclure la gestion des produits, des commandes et des clients, assurant une structure de données efficace pour des transactions rapides.

12 Exemples de modélisation pour IoT

Pour l'IoT, la modélisation des données met l'accent sur la gestion des capteurs, des appareils et des flux de données en temps réel, offrant une base solide pour les applications IoT.