**Gestion des Big Data dans les bases de données Oracle : Défis et Solutions**  
**Introduction**  
De nos jours, des volumes massifs de données sont stockés et traités dans des systèmes axés sur les données. L’un des défis les plus fréquents dans ce domaine est la gestion et l’analyse des données volumineuses dans des bases de données comme Oracle. Cet article examine un problème pratique impliquant des tables extrêmement grandes dans une architecture d’entrepôt de données (data warehouse).

Dans ce scénario, des tables contenant plus de 40 milliards d’enregistrements occupent plus de 2 téraoctets d’espace, avec environ 300 millions de nouveaux enregistrements ajoutés quotidiennement. Ce volume colossal de données engendre plusieurs défis, notamment :

* La mise à jour des données
* La production de rapports rapides et efficaces
* La présentation des données sous forme de graphiques et de tableaux de bord analytiques
* La réalisation d’opérations de fouille de données

Les requêtes échouent souvent à s’exécuter complètement ou prennent un temps extrêmement long en raison de la taille des données. Cet article propose un ensemble de solutions pratiques et optimisées pour relever ces défis.

**1. Optimisation de la Structure et du Stockage des Données**  
**a) Utilisation du Partitionnement**

* Partitionnez les tables volumineuses selon des critères adaptés (par exemple, date, identifiants spécifiques ou plages numériques).
  + Cela permet aux requêtes de scanner uniquement une partie des données au lieu de la table entière.
  + Oracle prend en charge diverses méthodes de partitionnement, telles que le partitionnement par plage (Range Partitioning), par hachage (Hash Partitioning) et par liste (List Partitioning).

**b) Compression des Données**

* Utilisez la fonctionnalité Advanced Compression d’Oracle.
  + Cela réduit l’espace disque et accélère le temps d’exécution des requêtes.

**c) Indexation Intelligente**

* Employez des Bitmap Indexes pour les tables impliquées dans des requêtes analytiques.

**• Utilisation des Index Partition-Wise**

* Dans les cas où une grande quantité de données est ajoutée aux tables, utilisez les *Partition-Wise Indexes* pour éviter une surcharge (overhead).

**2. Optimisation des Requêtes**  
**a) Query Rewrite et Vues Matérialisées**

* Exploitez les Vues Matérialisées (*Materialized Views*) pour effectuer des pré-calculs et mettre en cache les résultats.
  + Cette méthode permet de stocker les résultats complexes et de les réutiliser directement.
* Activez la fonction *Query Rewrite* pour que les requêtes utilisent automatiquement les Vues Matérialisées.

**b) Exécution Parallèle des Requêtes**

* Lancez les requêtes lourdes avec la fonction *Parallel Execution*.
  + Configurez correctement les paramètres de traitement parallèle (comme PARALLEL\_DEGREE\_POLICY) pour améliorer significativement les performances.

**c) Optimisation du Plan de Requête**

* Analysez et optimisez les plans d’exécution des requêtes en utilisant la commande EXPLAIN PLAN et des outils comme SQL Developer.

**3. Utilisation d’Outils et de Technologies Complémentaires**  
**a) Archivage des Données Anciennes**

* Transférez les données qui ne sont plus nécessaires quotidiennement (par exemple, les données de plus d’un an) vers une archive.
  + Des outils comme Oracle Data Pump ou les *External Tables* peuvent être utilisés à cet effet.

**b) Exploitation d’Oracle Exadata**

* Si des mises à niveau d’infrastructure sont possibles, l’utilisation d’Oracle Exadata peut considérablement améliorer les performances pour le chargement et l’analyse de grands ensembles de données.
  + Ce système est optimisé pour les volumes de données extrêmement élevés.

**c) Intégration avec des Technologies Big Data**

**• Pour l'analyse des données, transférez les grands ensembles de données vers des systèmes Big Data comme Apache Spark ou Hadoop.**

* Ces systèmes sont conçus pour le traitement et l’analyse de données massives et peuvent être utilisés en complément d’Oracle.

**4. Visualisation des Données et Reporting**  
**a) Utilisation de Tableaux de Bord Optimisés**

* Connectez des outils de BI comme Power BI, Tableau ou Oracle Analytics Cloud aux Vues Matérialisées ou aux données résumées.
* Exploitez les fonctionnalités de mise en cache (*Caching*) de ces outils.

**b) Résumé des Données**

* Stockez les données résumées dans des *Aggregation Tables*.
  + Par exemple, sauvegardez les données quotidiennes sous forme résumée (*Sum*, *Avg*, *Count*).

**5. Prévision et Fouille de Données (Data Mining)**  
**a) Utilisation d’Oracle Data Mining**

* Exploitez les fonctionnalités d’Oracle Advanced Analytics pour la fouille de données directement dans la base de données (*In-Database Analytics*).

**b) Transfert des Données vers des Outils Spécialisés**

* Pour la fouille de données, exportez les ensembles de données nécessaires vers des outils comme Python (avec des bibliothèques telles que pandas, scikit-learn et PySpark) ou des outils de BI spécialisés.

**6. Gestion des Charges Quotidiennes**  
**a) Utilisation du Traitement par Lots pour les Insertions**

* Chargez les nouvelles données dans les tables en utilisant les méthodes *Bulk Insert* ou *Direct Path Insert*.
  + Des outils comme Oracle SQL Loader peuvent considérablement accélérer le chargement des données.

**b) Conception d’un Processus ETL Optimisé**

* Utilisez des outils ETL tels qu’Oracle Data Integrator (ODI) pour concevoir des processus de chargement de données efficaces.

**Conclusion**

* **Priorisation des Tâches :** Commencez par le partitionnement des tables, la création de Vues Matérialisées et la configuration de l'optimisation des requêtes.
* **Amélioration de l’Infrastructure :** Si les problèmes persistent, envisagez la migration vers Oracle Exadata ou des systèmes Big Data.
* **Outils Complémentaires :** Exploitez des outils analytiques et de BI appropriés pour réduire les charges directes sur la base de données.

**Augmentation de l’Espace Disque avec les Vues Matérialisées**  
L’utilisation des Vues Matérialisées augmente l’espace disque utilisé, car elles stockent une copie physique des résultats des requêtes. Le niveau de cette augmentation dépend de plusieurs facteurs :

1. **Volume des Données Stockées dans la Vue Matérialisée :**
   * Si la Vue Matérialisée contient des données résumées ou agrégées (par exemple, SUM, AVG, COUNT), elle occupe beaucoup moins d’espace que l’ensemble des données.
   * Cependant, si elle stocke la totalité ou une grande partie des données (par exemple, à partir d’une requête JOIN lourde), l’espace disque utilisé peut être conséquent.
2. **Fréquence de Rafraîchissement :**
   * L’utilisation du **Rafraîchissement Incrémental** réduit l’augmentation de l’espace, car seules les modifications sont appliquées.
   * Avec un **Rafraîchissement Complet**, toutes les données sont reconstruites à chaque fois, ce qui peut entraîner une augmentation significative de l’espace disque.
3. **Compression :**
   * Dans Oracle, vous pouvez compresser les Vues Matérialisées pour réduire l’utilisation de l’espace disque. Cela est particulièrement efficace pour les vues contenant de grands ensembles de données.

**Comment Gérer l'Impact du Volume ?**  
Si l'augmentation du volume vous inquiète, envisagez les stratégies suivantes :

1. **Créer des Vues Matérialisées Légères et Ciblées :**
   * Stockez uniquement les données essentielles ou agrégées.
   * Évitez de créer des Vues Matérialisées qui stockent l’ensemble des données.
2. **Partitionnement des Vues Matérialisées :**
   * Comme pour les tables, les Vues Matérialisées peuvent être partitionnées pour réduire le volume et améliorer les performances.
3. **Politique de Rétention des Données :**
   * Ne conservez pas les données dans la Vue Matérialisée au-delà d’une plage de temps définie.
4. **Planification Appropriée du Rafraîchissement :**
   * Dans la mesure du possible, utilisez le Rafraîchissement à la Demande (*On-Demand Refresh*) ou le Rafraîchissement Incrémental Planifié pour réduire les coûts de mise à jour.

Bien que les Vues Matérialisées augmentent l’utilisation de l’espace disque, une conception adéquate axée sur la synthèse et les données analytiques critiques, associée à des fonctionnalités telles que la **Compression** et le **Rafraîchissement Incrémental**, peut considérablement réduire ces coûts tout en améliorant les performances du système.

**Utilisation du Big Data pour le Reporting et les Tableaux de Bord**  
De nos jours, les organisations génèrent des données à un rythme sans précédent. Ce volume massif de données pose des défis en matière de traitement, de stockage et de reporting. Les bases de données traditionnelles, en particulier lorsqu'elles traitent des volumes de données importants, peuvent rencontrer des limites en termes de vitesse et d'efficacité. Dans de tels cas, les **technologies Big Data** apparaissent comme une solution efficace.

Les technologies Big Data permettent non seulement un traitement plus rapide des ensembles de données massifs, mais offrent également des outils puissants pour le reporting et la visualisation des données. Ces capacités permettent aux organisations d’exploiter leurs données de manière plus efficace et de prendre de meilleures décisions.

1. **Pourquoi Utiliser les Technologies Big Data ?**

**- Évolutivité :** Les systèmes Big Data peuvent facilement s’adapter à l’augmentation des volumes de données.  
**- Vitesse Élevée de Traitement :** Le traitement des données de manière distribuée permet une exécution plus rapide des requêtes.  
**- Analyses et Reporting Rapides :** Des outils comme Apache Spark et Presto permettent des requêtes interactives sur de grands ensembles de données.  
**- Intégration avec les Outils BI :** De nombreux outils Big Data s’intègrent facilement aux tableaux de bord comme Tableau, Power BI ou Superset.

**2. Recommandations Technologiques**

**a) Apache Hadoop et HDFS :**

* Les données archivées ou anciennes peuvent être transférées vers HDFS pour le reporting à l’aide d’outils tels que Hive ou Impala.
* Hive peut servir de couche SQL sur vos données.

**b) Apache Spark :**

* Spark est particulièrement adapté au traitement en mémoire (In-Memory) des grands ensembles de données.
* Grâce à Spark SQL, vous pouvez exécuter des requêtes complexes sur vos données et transmettre directement les résultats aux outils BI.

**c) Presto (ou Trino) :**

* Presto est un moteur de requêtes distribué qui peut interroger rapidement des données provenant de différentes sources (par exemple, HDFS, Amazon S3, Oracle).
* Cet outil est très efficace pour construire des tableaux de bord rapides et des rapports en temps réel.

**d) Apache Kafka pour les Données en Streaming :**

* Si certaines de vos données sont en temps réel (par exemple, des enregistrements ajoutés quotidiennement), Kafka peut gérer et traiter ces données.
* En combinant Kafka avec des outils comme Apache Flink ou Apache Storm, vous pouvez activer le reporting en temps réel.

**e) Data Lake (par exemple, Amazon S3 ou Azure Data Lake) :**

* Transférez vos données vers un Data Lake et utilisez des outils Big Data pour les interroger.
* Des outils comme Athena (pour AWS) ou Databricks sont excellents pour cet objectif.

**3. Connexion aux outils de tableaux de bord**

Les technologies Big Data peuvent facilement s’intégrer avec des outils de visualisation de données :

* **Tableau :** Connexion directe à Apache Hive, Spark SQL, Presto ou BigQuery.
* **Power BI :** Prend en charge Spark et d'autres sources Big Data.
* **Apache Superset :** Un outil gratuit et open-source pour créer des tableaux de bord interactifs, compatible avec Spark et Presto.

**4. Processus proposé pour le reporting avec Big Data**

**Étape 1 :** Transférer les données analytiques vers Big Data

* Déplacez les données massives et historiques d'Oracle vers un Data Lake ou un cluster Big Data (par exemple, Hadoop).
* Utilisez des outils comme Sqoop pour migrer les données entre Oracle et Hadoop.

**Étape 2 :** Traiter les données avec Spark ou Hive

* Traitez les données à l'aide de Spark ou Hive.
* Extrayez les données agrégées ou résumées nécessaires pour le reporting.

**Étape 3 :** Stocker les résultats dans un format optimal

* Enregistrez les résultats des requêtes analytiques dans des formats légers et compatibles BI (par exemple, Parquet ou ORC).

**Étape 4 :** Connecter aux outils BI

* Connectez votre outil BI préféré au moteur Big Data et créez des tableaux de bord.

**5. Outils et services Big Data prêts à l’emploi**  
Si vous souhaitez obtenir des résultats plus rapidement, vous pouvez utiliser des services cloud :

* **Google BigQuery :** Un service d’analyse de données dans le cloud, rapide et idéal pour le reporting.
* **Amazon Redshift :** Une base de données analytique distribuée.
* **Azure Synapse Analytics :** Pour analyser des données volumineuses et créer des tableaux de bord BI.

**6. Points clés**

* **Partitionnement :** Assurez-vous que vos données sont partitionnées en fonction de critères appropriés dans toutes les technologies Big Data.
* **Mise en cache des données :** Utilisez la mise en cache pour améliorer la vitesse.
* **Sécurité et contrôle d’accès :** Gérez les accès aux données et outils pour protéger les informations sensibles.

**Comment les technologies Big Data accélèrent les requêtes et le traitement**

Les technologies Big Data sont conçues pour gérer des volumes massifs de données avec une vitesse remarquable. Cette efficacité repose sur des architectures et des techniques adaptées aux grandes échelles de données. Voici comment les outils Big Data accélèrent le traitement et les requêtes :

**1. Traitement distribué**

* Les données sont stockées et traitées sur plusieurs serveurs (nœuds) de manière distribuée.
* Chaque nœud traite une partie des données, et les tâches sont exécutées en parallèle.
* Des outils comme Hadoop et Spark utilisent ce modèle.  
  **Avantage :**  
  Le temps de traitement est réduit, car les opérations sont réparties entre plusieurs machines au lieu d’être effectuées sur une seule.

**2. Traitement en mémoire**

* Des outils comme Apache Spark traitent les données en mémoire (RAM) au lieu de les lire sur disque à chaque fois.
* Cette technique est particulièrement efficace pour les tâches de traitement en temps réel et répétitif.

**Avantages :**  
Lire et écrire des données depuis la mémoire est beaucoup plus rapide que depuis le disque, ce qui réduit considérablement le temps de traitement.

**3. Architecture de stockage en colonnes (Columnar Storage)**

* Des outils comme **Parquet** et **ORC** stockent les données dans un format en colonnes.
* Dans ce modèle de stockage, seules les colonnes nécessaires pour une requête sont lues, au lieu de la table entière.  
  **Avantage :**  
  Cette approche réduit le volume des données lues et augmente les performances des requêtes, notamment pour les analyses de grandes données.

**4. Exécution parallèle des requêtes (Parallel Query Execution)**

* Les moteurs de requête comme **Presto**, **Hive** et **Spark SQL** utilisent l'exécution parallèle des requêtes.
* Chaque requête est divisée en petites tâches et exécutée en parallèle.  
  **Avantage :**  
  Cette méthode utilise de manière optimale les ressources de calcul et réduit le temps d'exécution des requêtes.

**5. Utilisation des index et du partitionnement**

* Les outils Big Data partitionnent souvent les données.
  + Par exemple, les données peuvent être divisées selon la date, la localisation géographique ou d'autres champs.
* Les index sont également utilisés pour une récupération rapide des données.  
  **Avantage :**  
  Les requêtes ne scannent que les données nécessaires, pas l'ensemble du jeu de données.

**6. Compression**

* Les données sont stockées dans des formats compressés (ex. **Parquet** et **ORC**).
* Cela réduit la taille des données stockées et améliore la vitesse de lecture depuis le disque.  
  **Avantage :**  
  Lire des données compressées depuis le disque est plus rapide et permet également d'économiser de l'espace de stockage.

**7. Optimisation des requêtes et Catalyst Optimizer**

* Des outils comme **Apache Spark** utilisent des optimisateurs de requêtes pour analyser et choisir le chemin d'exécution le plus rapide pour une requête.  
  **Avantage :**  
  L'exécution optimisée des requêtes réduit l'utilisation des ressources et le temps d'exécution.

**8. Structuration en couches dans un Data Lake ou Data Warehouse**

* Les outils Big Data organisent les données en couches dans un Data Lake ou Data Warehouse :
  + **Données brutes** : Données non traitées.
  + **Données traitées** : Données préparées pour des analyses complexes.
  + **Données agrégées** : Données résumées pour des requêtes plus rapides.  
    **Avantage :**  
    Cette structure optimise l'accès aux données nécessaires et évite l'analyse de données inutiles.

**9. Traitement en flux pour les données en temps réel**

* Des outils comme **Apache Kafka**, **Flink**, et **Storm** traitent les données en temps réel plutôt qu'en lots.  
  **Avantage :**  
  Cette méthode est très rapide et efficace pour les analyses en temps réel et les tableaux de bord interactifs.

**10. Optimisation du cache (Caching)**

* De nombreux outils Big Data mettent en cache les résultats des requêtes fréquemment utilisées.
* Par exemple, **Presto** ou **Spark** peuvent conserver en mémoire les données récurrentes.  
  **Avantage :**  
  Il n'est pas nécessaire de réexécuter la requête, et les résultats sont livrés plus rapidement.

**Pourquoi ces techniques sont-elles efficaces ?**

* La distribution des tâches entre plusieurs machines garantit une utilisation optimale des ressources matérielles.
* Le traitement en mémoire et le stockage optimisé réduisent les coûts de lecture et d'écriture sur le disque.
* Les techniques d'optimisation des requêtes et la mise en cache des données réduisent considérablement le temps de réponse.

L'utilisation des outils Big Data ne nécessite pas nécessairement de grandes quantités de RAM. L'une des caractéristiques clés de ces outils est le traitement distribué, qui vous permet de traiter de grands volumes de données sans avoir besoin d'un matériel extrêmement puissant. Voici comment les outils mentionnés aident à améliorer l'utilisation de la RAM :

1. **Traitement Distribué**
   * Les données et les tâches de traitement sont réparties sur plusieurs machines (nœuds).
   * Chaque nœud ne traite qu'une partie des données, ce qui signifie que l'ensemble du jeu de données ne doit pas tenir dans la mémoire d'une seule machine.
   * Cela signifie qu'avec une RAM limitée (par exemple, 16 ou 32 Go par nœud), vous pouvez traiter plusieurs téraoctets de données.
   * Exemple : Dans un cluster Hadoop ou Spark avec 10 machines, si chaque machine dispose de 32 Go de RAM, un total de 320 Go de RAM est disponible et utilisé de manière distribuée.
2. **Stockage Basé sur Disque**
   * De nombreux outils Big Data comme Hadoop et Spark ont la capacité de traiter des données à partir du disque.
   * Si les données sont trop volumineuses pour tenir en mémoire, ces outils utilisent le traitement basé sur le disque.
   * Dans Spark, cette fonctionnalité est appelée Spill to Disk.
   * Avantage : Cela permet de traiter de très grands ensembles de données même avec du matériel moyen.
3. **Traitement par Lots (Batch Processing)**

• **Les outils comme Hadoop et Hive traitent les données par lots, ce qui signifie que les données sont divisées en morceaux plus petits et il n'est pas nécessaire de charger toutes les données en mémoire.**  
**Avantage :**  
Cette méthode est adaptée au traitement de grandes quantités de données lorsque la mémoire est limitée.

1. **Utilisation de formats optimisés et compressés**  
   • Les données sont généralement stockées dans des formats compressés comme Parquet ou ORC.  
   • Ces formats ne chargent que les colonnes nécessaires depuis le disque, ce qui réduit la pression sur la RAM.  
   **Avantage :**  
   Il n'est pas nécessaire de charger toute la table en mémoire, seules les parties nécessaires sont lues.
2. **Traitement en mémoire pour des données spécifiques**  
   • Des outils comme Apache Spark permettent le traitement en mémoire, mais vous pouvez également traiter des données lourdes par lots ou les diviser en morceaux plus petits.  
   • Si vous avez peu de RAM, Spark déplace automatiquement les données sur le disque.  
   **Résultat :**  
   Le traitement en mémoire est optimisé et effectué uniquement lorsque cela est nécessaire.
3. **Clusters plus grands au lieu de plus de RAM**  
   • Au lieu d'utiliser un serveur avec une très grande RAM (par exemple, plusieurs téraoctets), vous pouvez créer un cluster avec plusieurs serveurs plus petits.  
   • Chaque serveur (nœud) peut avoir une RAM modérée (par exemple, 16 ou 32 Go).  
   **Avantage :**  
   Cette méthode réduit les coûts et rend votre système évolutif.
4. **Exemples pratiques pour gérer de grandes données**

**a) Spark avec de grandes données**  
• Spark stocke les données dans des RDDs ou des DataFrames.  
• Si la RAM est pleine, les données sont transférées sur le disque (Spill to Disk).  
• Vous pouvez augmenter le nombre de partitions pour réduire la charge sur chaque nœud.  
**b) Hive et Hadoop**  
• Hadoop divise les données en petits blocs (généralement 128 Mo ou 256 Mo).  
• Chaque bloc est traité par un nœud, et il n'est pas nécessaire d'avoir une grande RAM sur une seule machine.

1. **Points clés pour réduire les besoins en RAM**
2. **Partitionnement des données :**
   * Stockez les données de manière partitionnée pour que seules les sections nécessaires soient traitées.
3. **Utilisation de formats compressés :**
   * Stocker les données dans les formats Parquet ou ORC aide à réduire la charge sur la RAM.
4. **Configuration correcte des partitions dans Spark ou Hadoop :**
   * Augmenter le nombre de partitions réduit la charge sur la RAM de chaque nœud.
5. **Distribution en cluster :**
   * Au lieu d'augmenter la RAM sur une seule machine, utilisez plusieurs machines avec une RAM modérée.
6. **Réglage des ressources système :**
   * Optimisez les paramètres de mémoire et de disque pour des outils comme Spark.

Pour implémenter Apache Spark dans un environnement d'entreprise multi-utilisateurs et le configurer pour différents niveaux d'accès, vous devez procéder à une installation et une configuration précises. Voici les étapes d'installation, de configuration et de mise en place de Spark dans les environnements Linux et Windows.

1. **Choisir l'architecture de Spark**  
   Tout d'abord, vous devez décider du type d'architecture pour déployer Spark :  
   • **Standalone Cluster :** Pour des besoins simples et à petite échelle.  
   • **YARN Cluster :** Si vous utilisez Hadoop.  
   • **Kubernetes Cluster :** Adapté aux environnements cloud et à haute évolutivité.  
   • **Spark sur Databricks ou EMR :** Pour un déploiement plus rapide et une gestion plus facile dans les environnements cloud.
2. **Prérequis**  
   a) **Prérequis communs (Linux et Windows) :**

* **Java :**  
  o Spark nécessite Java 8 ou Java 11. Installez-le et assurez-vous que la variable d'environnement JAVA\_HOME est définie.  
  java -version
* **Python (facultatif) :**  
  Si vous souhaitez utiliser PySpark, installez Python 3.x.
* **Scala (facultatif) :**  
  Si vous utilisez Scala, installez-le.
* **Hadoop (facultatif) :**  
  Si vous utilisez YARN ou HDFS, installez et configurez Hadoop.
* **Binaries Spark :**  
  o Téléchargez Spark depuis son site officiel : [Apache Spark Downloads].  
  o Choisissez la version pré-compilée avec Hadoop.

b) **Prérequis supplémentaires pour Linux :**  
• **SSH :** Pour la communication entre les nœuds du cluster Spark.  
Installation sur Ubuntu :

sudo apt-get install openssh-server

• **Configurations :**  
Configurez l'accès SSH sans mot de passe pour l'utilisateur spark.

1. **Installation et Configuration**  
   **Création d’un utilisateur sur Hadoop**  
   Au départ, un utilisateur est créé pour Hadoop, dédié aux services Hadoop et aux tâches administratives.  
   Pour créer un utilisateur, utilisez la commande ci-dessous, qui emploie la commande adduser :

$ sudo adduser hadoop

De plus, pour accorder un accès administratif à cet utilisateur, ajoutez-le au groupe sudo :

$ sudo usermod -aG sudo hadoop

Enfin, passez à ce nouvel utilisateur :

$ sudo su - hadoop

**Configuration de l’accès SSH sans mot de passe**  
Pour permettre une communication fluide entre les nœuds dans un cluster Hadoop, un accès SSH sans mot de passe est configuré. Cela permet aux services Hadoop de communiquer en toute sécurité sans nécessiter d'authentification manuelle.  
Tout d’abord, installez le serveur et le client OpenSSH :

**Installation d’OpenSSH**  
La commande suivante installe OpenSSH sur Linux, vérifie la disponibilité des composants du serveur, et démarre le service par défaut :

$ apt install openssh-server openssh-client -y

Pour vérifier si l’installation a réussi, utilisez la commande suivante :

$ systemctl status ssh

Sortie attendue :

● sshd.service - OpenSSH server daemon

Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: disabled)

Active: (running) since Mon 2024-03-18 15:04 WAT; 1h 1min ago

Docs: man:sshd(8) man:sshd\_config(5)

Main PID: 4308 (sshd)

Tasks: 1 (limit: 18957)

Memory: 2.2M CPU: 38ms

CGroup: /system.slice/sshd.service

└─4308 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Ce résultat indique qu’OpenSSH fonctionne correctement.

**Création d’une clé SSH**  
Pour générer une clé SSH, utilisez la commande ssh-keygen comme suit :

$ ssh-keygen -t rsa -P "" -f ~/.ssh/id\_rsa

Exemple de sortie :

Pour générer une clé SSH, nous utilisons la commande ssh-keygen :

$ ssh-keygen -t rsa -P "" -f ~/.ssh/id\_rsa

Generating public/private rsa key pair.

/home/hdoop/.ssh/id\_rsa already exists.

Overwrite (y/n)? y

Your identification has been saved in /home/hdoop/.ssh/id\_rsa

Your public key has been saved in /home/hdoop/.ssh/id\_rsa.pub

The key fingerprint is:

SHA256:09K2smH7S2nmCAG8PpYM7jkNVCvYJ8HbDtWbO7U+Hes hdoop@kali

The key's randomart image is:

+---[RSA 3072]----+

| . . |

| o.o . |

| o \*o. o |

|. B +oo .o |

| ..\*. .oS.+ |

| ..+..o..+.o |

| .o\* .o+.\*o |

| ..o.. oo@o |

| o. ++E. |

+----[SHA256]-----+

Ce code génère une clé RSA sans nécessiter de phrase de passe.

Cette clé sera sauvegardée dans le chemin ~/.ssh/id\_rsa, et la clé publique sera stockée dans ~/.ssh/id\_rsa.pub.  
Ensuite, la clé publique est copiée dans le fichier authorized\_keys :

$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

Pour garantir un accès correct, utilisez les commandes suivantes :

$ chmod 700 ~/.ssh

$ chmod 600 ~/.ssh/authorized\_keys

**Configuration du pare-feu pour l'accès**

Dans certains cas, les paramètres du pare-feu doivent être configurés. Les commandes suivantes peuvent être utilisées pour appliquer les paramètres souhaités :  
Par exemple, si vous utilisez **Uncomplicated Firewall (UFW)**, vous pouvez facilement ouvrir le port 22 :

$ sudo ufw allow ssh

Alternativement, si vous utilisez un autre outil de gestion de pare-feu ou si vous modifiez directement les règles du pare-feu, vous devez vous assurer que les connexions entrantes au port 22 sont autorisées.

**Tester la connexion SSH**

Enfin, utilisez la commande SSH :

$ ssh localhost

La sortie ressemblera à ce qui suit :

Linux kali 6.6.9-amd64 1 SMP PREEMPT\_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08) x86\_64

The programs included with the Kali GNU/Linux system are free software;

the exact distribution terms for each program are described in the

individual files in /usr/share/doc/\*/copyright.

...

**Télécharger Hadoop**

Ensuite, téléchargez la dernière version stable de Hadoop depuis la page de téléchargement officielle d'Apache Hadoop à l'aide de la commande wget :

$ wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz

Après avoir terminé le téléchargement, nous extrayons le fichier téléchargé à l'aide de la commande tar :

$ tar -xvzf hadoop-3.3.6.tar.gz

Cette commande extrait le contenu du fichier téléchargé.

**Déplacement de Hadoop**

Ensuite, nous déplaçons le contenu extrait du package Hadoop vers le répertoire d'installation /usr/local/hadoop :

$ sudo mv hadoop-3.3.6 /usr/local/hadoop

Ensuite, nous créons un répertoire pour stocker les journaux système de Hadoop :

$ sudo mkdir /usr/local/hadoop/logs

Enfin, nous modifions la propriété du répertoire Hadoop :

$ sudo chown -R hadoop:hadoop /usr/local/hadoop

**Configuration des variables d'environnement**

Pour configurer les variables d'environnement de Hadoop, nous ouvrons le fichier ~/.bashrc avec un éditeur de texte :

$ sudo nano ~/.bashrc

Ensuite, nous ajoutons plusieurs lignes à la fin du fichier :

export HADOOP\_HOME=/usr/local/hadoop

export HADOOP\_INSTALL=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export PATH=$PATH:$HADOOP\_HOME/sbin:$HADOOP\_HOME/bin

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/native"

Enfin, nous activons les variables d'environnement mises à jour :

$ source ~/.bashrc

**Configuration des variables d'environnement Java**

Pour permettre à Hadoop d'utiliser ses différents composants tels que YARN, HDFS, MapReduce et les paramètres liés aux projets, les variables d'environnement Java sont définies dans le fichier de configuration hadoop-env.sh.

Tout d'abord, nous devons trouver le chemin vers le compilateur Java (javac) :

$ which javac

/usr/bin/javac

De plus, nous utilisons la commande readlink pour déterminer le répertoire où se trouve OpenJDK :

$ readlink -f /usr/bin/javac

/usr/lib/jvm/jdk-21-oracle-x64/bin/javac

Ensuite, nous éditons le fichier hadoop-env.sh pour définir les variables d'environnement Java :

$ sudo nano $HADOOP\_HOME/etc/hadoop/hadoop-env.sh

Nous ajoutons les lignes suivantes au fichier :

export JAVA\_HOME=/usr/lib/jvm/jdk-21-oracle-x64/bin/javac

export HADOOP\_CLASSPATH+=" $HADOOP\_HOME/lib/\*.jar"

Enfin, nous sauvegardons le fichier et fermons l'éditeur.

**Configuration de l'environnement Hadoop**

Avant de démarrer le cluster Hadoop, il est essentiel de configurer l'environnement Hadoop pour définir divers paramètres.

**Téléchargement des bibliothèques nécessaires**

Nous commençons par naviguer vers le répertoire lib :

$ cd /usr/local/hadoop/lib

Ensuite, nous utilisons la commande wget pour télécharger le fichier Javax Activation :

$ sudo wget https://jcenter.bintray.com/javax/activation/javax.activation-api/1.2.0/javax.activation-api-1.2.0.jar

Après avoir terminé le téléchargement, vérifions la version installée de Hadoop :

$ hadoop version

Résultat :

Hadoop version: 3.3.6

Subversion revision: rXXXXXXX

Compiled by: username on date at time

Compiled with flags: ...

Le résultat montre que la version installée de Hadoop est la 3.3.6.

**Modifier la configuration de core-site.xml**

Ensuite, nous éditons le fichier de configuration core-site.xml pour spécifier l’URL du NameNode :

$ sudo nano $HADOOP\_HOME/etc/hadoop/core-site.xml

Nous ajoutons les lignes suivantes au fichier et le sauvegardons :

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://0.0.0.0:9000</value>

<description>The default file system URI</description>

</property>

</configuration>

De plus, nous créons un répertoire pour stocker les métadonnées des nœuds et modifions sa propriété pour Hadoop :

$ sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}

$ sudo chown -R hadoop:hadoop /home/hadoop/hdfs

Cette commande crée avec succès un répertoire pour stocker les métadonnées des nœuds, puis en modifie la propriété.

**Modifier la configuration de hdfs-site.xml**

Nous éditons le fichier de configuration hdfs-site.xml :

$ sudo nano $HADOOP\_HOME/etc/hadoop/hdfs-site.xml

Ensuite, nous ajoutons plusieurs lignes au fichier et le sauvegardons :

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.name.dir</name>

<value>file:///home/hadoop/hdfs/namenode</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>file:///home/hadoop/hdfs/datanode</value>

</property>

</configuration>

Ainsi, nous utilisons les chemins que nous avons créés et les configurons de manière appropriée.

**Modifier la configuration de yarn-site.xml**

Enfin, nous modifions le fichier de configuration yarn-site.xml et définissons les paramètres liés à YARN :

$ sudo nano $HADOOP\_HOME/etc/hadoop/yarn-site.xml

Nous ajoutons les lignes suivantes pour configurer les services auxiliaires de YARN :

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

</configuration>

**Valider la configuration de Hadoop**

Toujours connecté en tant qu'utilisateur Hadoop, nous validons la configuration de Hadoop et formatons le NameNode HDFS :

$ hdfs namenode -format

Cette commande efface toutes les données existantes dans le NameNode et initialise efficacement un nouveau système de fichiers HDFS. Comme prévu, elle supprime également les métadonnées du système de fichiers ainsi que les fichiers, répertoires et emplacements des blocs.

**4) Travailler avec le cluster Apache Hadoop**

Pour utiliser **Apache Hadoop**, nous démarrons tous les clusters.

**Démarrer le NameNode et le DataNode**

Nous utilisons le script start-dfs.sh pour démarrer le NameNode et le DataNode :

$ start-dfs.sh

Starting namenode on [namenode\_host]... started

Starting secondarynamenode on [secondarynamenode\_host]... started

Starting datanode on [datanode\_host1]... started

... (similar messages for all datanodes)

**Démarrer les gestionnaires de ressources et de nœuds Yarn**

La commande ci-dessus démarre le gestionnaire de ressources YARN et les gestionnaires de nœuds, permettant au cluster de gérer l'allocation des ressources et l'exécution des tâches :

$ start-yarn.sh

starting yarn daemons

starting resourcemanager, logging to /path/to/yarn-resourcemanager.log ... started

starting nodemanager on [nodemanager\_host1]... started

... (similar messages for all nodemanagers)

**Vérification de tous les composants en cours d'exécution**

Après avoir démarré les différents composants du cluster Apache Hadoop, il est important de vérifier s'ils fonctionnent correctement.

La commande **Java Virtual Machine Process Status Tool (jps)** répertorie tous les processus Java en cours d'exécution sur le système. Dans le contexte de Hadoop, exécuter la commande jps est un moyen pratique de vérifier si les composants Hadoop sont actifs et en cours d'exécution.

$ jps

3214 SecondaryNameNode

4320 Jps

3854 Resourcemanager

3456 DataNode

4084 NodeManager

3274 NameNode

Comme prévu, la sortie de cette commande fournit une liste de tous les processus Java en cours d'exécution sur le système, y compris les composants Hadoop.

Enfin, nous pouvons ouvrir un navigateur web et accéder aux interfaces pour le NameNode (http://localhost:9870) et le ResourceManager (http://localhost:8088).

**Installation de Spark**

Tout d'abord, mettez à jour les paquets du système :

$ sudo apt update

Ensuite, installez Java :

$ sudo apt install default-jdk -y

Enfin, vérifiez la version :

$ java -version

Installez les paquets nécessaires :

$ sudo apt install curl mlocate git scala -y

$ curl -O https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz

Extrayez l'archive Spark tarball :

$ sudo tar xvf spark-3.2.0-bin-hadoop3.2.tgz

Créez un répertoire nommé /opt/spark. Ensuite, déplacez les fichiers extraits dans le répertoire d'installation et modifiez les permissions :

$ sudo mkdir /opt/spark

$ sudo mv spark-3.2.0-bin-hadoop3.2/\* /opt/spark

$ sudo chmod -R 777 /opt/spark

Modifiez le fichier de configuration .bashrc pour ajouter l'installation d'Apache Spark au chemin système (PATH) :

$ sudo vim ~/.bashrc

Ajoutez les lignes suivantes à la fin du fichier, enregistrez et quittez :

export SPARK\_HOME=/opt/spark

export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

Enregistrez les modifications pour qu'elles soient appliquées :

$ source ~/.bashrc

Démarrez le serveur maître autonome :

$ start-master.sh

Démarrez le processus de travail d'Apache Spark :

$ start-slave.sh

Pour tester l'installation, créez un RDD depuis le CLI.

Enfin, vous pouvez commencer à travailler en utilisant l'URL suivante :

http://server\_public\_IP:8080

**Conseils importants**  
• **Surveillance :**  
Utilisez des outils comme Ganglia ou Spark UI pour la surveillance du cluster.  
• **Scalabilité :**  
Augmentez le nombre de nœuds du cluster si nécessaire.

• **Sauvegarde :**  
Sauvegardez régulièrement les données et les configurations de Spark.

Étant donné les conditions, qui incluent des données volumineuses (plus de 2 To), un taux de croissance élevé (300 millions d'enregistrements par jour) et la nécessité d'un environnement multi-utilisateurs pour des tableaux de bord rapides et optimisés, l'architecture Apache Spark sur YARN ou Kubernetes est recommandée. Ci-dessous, les raisons de ce choix et ses avantages sont expliquées.

**Pourquoi Spark sur YARN ?**  
Si vous utilisez déjà Hadoop ou HDFS (ou prévoyez de l'utiliser), Spark sur YARN est la meilleure option. Cette architecture vous aide à :

1. Traiter directement les données volumineuses stockées dans HDFS.
2. Utiliser la plateforme distribuée Hadoop pour la gestion des ressources et la distribution des tâches.
3. Profiter des autres fonctionnalités de Hadoop, comme Hive, pour les requêtes basées sur SQL.

**Avantages pour vous :**

1. **Traitement des données volumineuses :**  
   YARN vous permet de collecter et de traiter les données provenant de plusieurs sources volumineuses (par exemple, Oracle).
2. **Support multi-utilisateurs :**  
   YARN prend en charge nativement plusieurs utilisateurs, et vous pouvez gérer les accès avec des configurations simples.
3. **Scalabilité :**  
   Vous pouvez facilement ajouter plus de nœuds au cluster Hadoop pour évoluer avec l'augmentation des données et des utilisateurs.
4. **Support des données provenant de plusieurs sources :**  
   YARN peut intégrer des données provenant de diverses sources (Oracle, fichiers CSV, JSON, etc.).

**Pourquoi Spark sur Kubernetes ?**  
Si vous souhaitez un environnement plus flexible, évolutif et moderne, l'architecture Spark sur Kubernetes est la meilleure option. Kubernetes vous aide à :

1. Exécuter le cluster Spark sur des environnements cloud ou sur des serveurs sur site.
2. **Allouer dynamiquement des ressources (CPU, mémoire).**
3. **Gérer différents utilisateurs avec des niveaux d'accès variés.**

**Avantages pour vous :**

**Haute flexibilité :**  
Kubernetes vous permet de configurer plusieurs clusters dans différents environnements (cloud ou sur site).

**Gestion intelligente des ressources :**  
Kubernetes gère automatiquement les ressources en fonction des besoins de Spark. Cette fonctionnalité permet de réduire les coûts et d'optimiser l'utilisation des ressources.

**Intégration avec différents systèmes :**  
Kubernetes s'intègre facilement avec des outils cloud tels qu'AWS, Azure, ou Google Cloud, ainsi qu'avec des bases de données comme Oracle.

**Support multi-utilisateurs et sécurité :**  
Kubernetes permet de définir des rôles (Roles) et des politiques de sécurité (RBAC), permettant aux différents utilisateurs de se connecter au cluster avec des niveaux d'accès différents.

**Comparaison des deux architectures pour vos conditions**

|  |  |  |
| --- | --- | --- |
| Caractéristique | Spark sur YARN | Spark sur Kubernetes |
| Support des données volumineuses | Très fort (HDFS et Hadoop) | Fort (avec Object Storage) |
| Gestion des ressources | Forte, mais légèrement manuelle | Très intelligente et dynamique |
| Flexibilité | Limitée à l'environnement Hadoop | Très flexible (local et cloud) |
| Support multi-utilisateurs | Bon, mais nécessite une configuration | Très fort et simple avec RBAC |
| Scalabilité | Excellente (ajout de nœuds à YARN) | Excellente et dynamique (scalabilité des nœuds dans Kubernetes) |
| Sécurité et rôles | Limité à YARN | Avancée et flexible |
| Facilité d'installation | Plus facile (pour ceux qui ont Hadoop) | Plus complexe mais plus moderne |

**Choix approprié :**

1. **Si vous utilisez déjà Hadoop ou HDFS :**  
   Le meilleur choix est Spark sur YARN. Cette architecture est plus adaptée à vos conditions et nécessite peu de modifications de l'infrastructure.
2. **Si vous souhaitez une infrastructure plus moderne et dynamique :**  
   Choisissez Spark sur Kubernetes. Cette option est idéale pour les environnements cloud ou hybrides et offre une plus grande flexibilité.

**Recommandations pratiques**

1. **Pour transférer des données Oracle vers Spark :**  
   o Utilisez des outils comme Apache Sqoop pour transférer des données volumineuses d'Oracle vers HDFS.  
   o Alternativement, lisez les données directement depuis Oracle en utilisant le Spark JDBC Connector.
2. **Pour des tableaux de bord rapides et optimisés :**  
   o Préparez les données avec Spark SQL ou Hive.  
   o Utilisez des outils comme Apache Superset ou Tableau pour créer des tableaux de bord.
3. **Pour un environnement multi-utilisateurs :**  
   o Sur Spark sur YARN : Gérez l'accès au niveau du YARN Resource Manager.  
   o Sur Spark sur Kubernetes : Utilisez RBAC pour définir l'accès des utilisateurs.