République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaia

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

L'ultime guide pour commencer un projet Big Data

-Polycopié de cours-

Etabli par Dr Houda EL BOUHISSI Epouse BRAHAMI

Année académique 2024-2025

Préface

Bienvenue dans ce polycopié de cours et travaux pratiques consacré à l'univers du Big Data. Nous allons explorer les concepts fondamentaux, les outils et les techniques qui permettent de traiter, analyser et exploiter des ensembles de données massives. Actuellement, le Big Data est aujourd'hui l'une des pierres angulaires de l'innovation technologique dans de nombreux secteurs, de la santé à l'économie, en passant par l'industrie et les sciences sociales. Il offre des opportunités sans précédent, mais aussi des défis importants en termes de gestion, de traitement et de sécurité des données.

Ce polycopié a été conçu pour accompagner les étudiants et les professionnels du Big Data afin qu'ils puissent acquérir une maîtrise concrète des outils et des méthodologies utilisées dans le traitement du Big Data. Ainsi, ce cours, vous permettra de :

- Comprendre les concepts fondamentaux du Big Data, à savoir c'est quoi le Big Data, de données massives, les différents Vs et les défis associés à leur gestion.
- Découvrir les bases de données NoSQL et les modèles existants.
- Faire du calcul distribué avec MapReduce.
- Faire du calcul distribué avec SPARK.

Les travaux pratiques permettront de travailler sur des jeux de données réels et complexes, et vous donneront un aperçu des défis que rencontrent les entreprises lorsqu'elles manipulent de vastes quantités d'informations.

Ce parcours est conçu pour être progressif et interactif, avec des explications détaillées, des exemples concrets et des exercices à réaliser sur le terrain.

J'espère que ce polycopié sera aussi enrichissant pour nos étudiants.

Pour toute correction ou proposition en vue d'améliorer, merci d'envoyer un mail à l'adresse :

Houda.elbouhissi@univ-bejaia.dz

Dr. Houda EL BOUHISSI Epse BRAHAMI

Table des matières

Liste des abréviations

API	Application Programming Interface
CAP	Consistency, Availability et Partition tolerance
HDFS	Hadoop File System
NoSQL	Not only SQL
SGBD	Système de gestion des bases de données
SGBDR	Système de gestion des bases de données Relationnel
SQL	Structured Query Language
YARN	Yet Another Resource Negotiator

Introduction

Le reste du document est organisé en 5 différents chapitres :

- Chapitre 1 : « Introduction **aux Big Data** » met l'accent sur les notions fondamentales à acquérir pour démarrer un projet Big Data.
- Chapitre 2 : « **Les bases de données NoSQL »** qui présente les différents types de bases de données NoSQL avec des exemples explicatifs et des TPs.
- Chapitre 3 : « **Calcul distribué avec MapReduce »** permettant aux lecteurs de découvrir le monde du calcul distribué à travers des exemples détaillés et des exercices.
- Chapitre 4 : « **L'écosystème Hadoop »,** une vue générale de Framework le plus utilisé pour la mise en place d'un projet Big Data.
- Chapitre 5 : « **Apache SPARK »,** ce dernier chapitre présente des notions basiques de l'util Apache Spark qui est actuellement le concurrent de MapReduce pour le calcyl distribué.

J'espère que ce document servira les étudiants et sera d'un grand support pédagogique.

Chapitre I : Introduction aux Big Data

I.1. Introduction

Les Big Data (données massives en français) désignent des ensembles de données tellement volumineux, complexes et variés qu'ils ne peuvent pas être traités et analysés avec les outils traditionnels. L'émergence de ces données a été rendue possible grâce à l'évolution des technologies de collecte, de stockage, de traitement et d'analyse des informations.

Les Big Data sont devenues une ressource précieuse dans de nombreux domaines, car elles permettent d'obtenir des insights pertinents qui peuvent transformer les processus décisionnels, améliorer la performance des entreprises et offrir de nouvelles perspectives dans divers secteurs.

I.2. Les Vs du Big Data

Le concept « Big Data » regroupe une famille d'outils qui répondent à une triple problématique dite règle des 3Vs : le Volume, la Vitesse et la Variété, auxquels s'ajoutent d'autres "V" complémentaires, comme ceux de Valeur et de Validité (figure I.1).

Ces sept caractéristiques, également appelées les 7Vs, permettent de définir les Big Data et d'illustrer leur potentiel et leurs défis.

Il est à noter qu'il existe d'autres Vs concernant l'analyse et le traitement des Big Data.

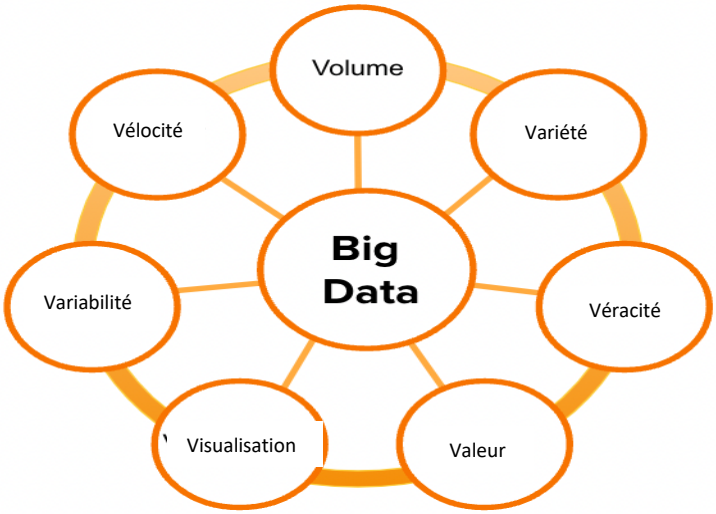


Figure I.1 : Les 7 Vs du Big Data

- **Volume** : Il s'agit de la quantité massive de données générées par les utilisateurs, les objets connectés, les transactions commerciales, les réseaux sociaux, les blocs, …etc. Nous comptons actuellement des tailles de données dépassant les milliards
- **Vitesse** : Les données sont générées et doivent être traitées à une grande vitesse, souvent en temps réel, pour permettre une prise de décision rapide. Un exemple intéressant est celui de la santé du citoyen où les données sont importantes pour la prise de décision de l'état de santé des patients.
- **Variété** : Les données proviennent généralement de sources multiples comme les objets connectés, les smartphones, etc. sous différentes formes : données structurées (tables, bases de données relationnelles), semi-structurées (fichiers JSON, XML) ou non structurées (textes, vidéos, images).
- **Véracité** : Il s'agit de la fiabilité et de la qualité des données. Certaines données peuvent être inexactes ou contradictoires, ce qui complique leur analyse. La véracité des données analysées constitue un atout majeur pour les décideurs.
- **Valeur** : Cela se réfère à l'importance et à l'utilité des données dans un contexte particulier. Certaines données ont une valeur dans un domaine et pas dans un autre, par exemple, le matricule d'un étudiant est utile pour une application de gestion de scolarité et non dans une application de gestion des patients.
- **Variabilité** : Ce terme fait référence aux fluctuations dans la manière dont les données sont générées et collectées, rendant difficile leur gestion et leur traitement.
- **Visibilité** : Ce concept désigne la capacité à identifier, comprendre et extraire les informations significatives des vastes ensembles de données.

I.3. Domaines d'application des Big Data

L'utilisation des technologies des Big Data concerne actuellement plusieurs domaines et diverses applications, je cite ici quelques exemples (figure I.2) :

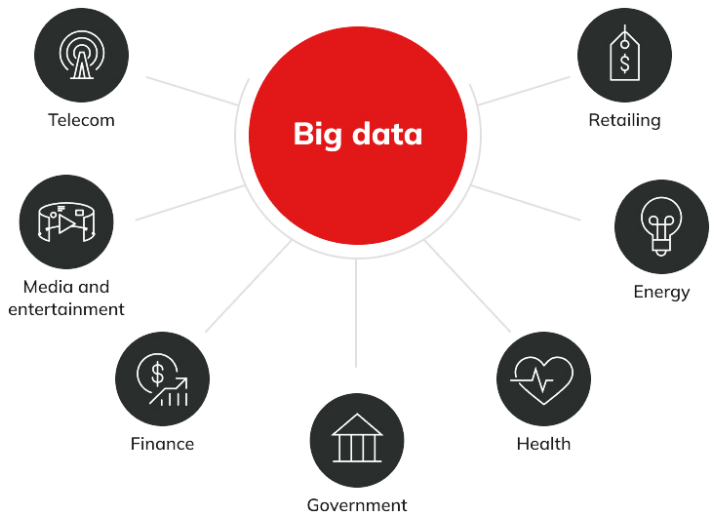


Figure I.2 : Domaines d'application du Big Data

- **Marketing** : Les entreprises utilisent les Big Data pour comprendre les comportements des consommateurs, cibler les publicités de manière plus précise et personnaliser les offres en fonction des besoins des clients.
- **Healthcare** : L'analyse des Big Data dans le domaine de la santé permet de mieux comprendre les maladies, d'améliorer les soins aux patients, de prédire des épidémies, ou encore de personnaliser les traitements médicaux grâce aux données génomiques et cliniques et de recommander des traitements et des suivis efficaces aux patients.
- **Finance et Banque** : Les banques et les finances utilisent les Big Data pour améliorer les prestations, détecter les fraudes, analyser les risques, optimiser les investissements et améliorer la prise de décision en matière de crédits et de placements.
- **Transport** : Les entreprises du secteur du transport utilisent les Big Data pour optimiser les itinéraires, prévoir la demande, gérer les stocks et améliorer la gestion des chaînes d'approvisionnement et offrir aux clients des services en fonction de leurs besoins.
- **Enseignement et Éducation** : Les données collectées auprès des étudiants et des enseignants permettent d'améliorer les méthodes d'enseignement, de personnaliser les parcours éducatifs et de prédire les performances des élèves.

- **Agriculture** : L'agriculture de précision utilise les Big Data pour analyser les conditions météorologiques, la qualité du sol et la croissance des cultures, permettant ainsi d'optimiser les rendements et d'assurer une gestion durable des ressources.

I.4. Technologie des Big Data

Pour traiter et analyser les Big Data, diverses technologies ont été développées. Ces technologies permettent de gérer la quantité de données, de les stocker, de les traiter et d'en extraire des informations utiles :

- Les bases de données NoSQL (MongoDB, Cassandra) sont couramment utilisées pour stocker des données volumineuses.
- L'écosystème Hadoop qui est Framework regroupant divers outils pour le traitement et l'analyse, la visualisions es données volumineuses.

I.5. Conclusion

En conclusion, les Big Data représentent un changement de paradigme dans la gestion et l'analyse des données. Grâce à leur capacité à offrir des informations pertinentes et en temps réel, elles permettent d'améliorer les performances, d'optimiser les processus et de créer de nouvelles opportunités dans de nombreux secteurs.

Cependant, le traitement des Big Data implique de surmonter des défis techniques importants, notamment la gestion du volume, de la vitesse, de la variété, de la véracité, de la valeur, de la variabilité et de la visibilité des données. L'évolution continue des technologies de Big Data promet de faciliter cette gestion et d'ouvrir encore plus de possibilités d'innovation.

I.6. Quiz

Voilà quelques questions pour tester vos connaissances, il est demandé de cocher la bonne réponse :

1. Lequel des points suivants ne décrivent pas le phénomène Big Data :
   - ☐ Le volume des données
   - ☐ La structure des données

- ☐ L’origine des données

2. Le Framework Hadoop a été développé en langage :
   - ☐ C++
   - ☐ Java
   - ☐ SQL
3. La simple collecte d'informations ne suffit pas pour produire une réelle valeur commerciale. L'analyse par le Big Data est nécessaire pour :
   - ☐ Formuler des diagrammes et des graphiques accrocheurs
   - ☐ Extraire des indications précieuses à partir des données
   - ☐ Intégrer les données provenant de sources internes et externes
4. Quelle entreprise informatique est le plus grand fournisseur de Big Data dans le monde ?
   - ☐ DELL
   - ☐ HP
   - ☐ IBM
5. Le volume des données d'entreprise à travers le monde double environ tous les :
   - ☐ 1 an
   - ☐ 1 an et demi
   - ☐ 2 ans
6. Combien d'heures de vidéo sont téléchargées sur YouTube chaque minute ?
   - ☐ 100
   - ☐ 200
   - ☐ 400
7. Le type de données qu'Hadoop peut traiter est :
   - ☐ Structurée
   - ☐ Nom structurée
   - ☐ Semi-structurée
   - ☐ Toutes ces réponses

8. L'architecture d'un cluster Hadoop est dite :
   - ☐ Monolithique
   - ☐ Hybride
   - ☐ Distribuée
9. Qui a créé le Framework logiciel Hadoop, très populaire dans le monde du Big Data ?
   - ☐ Doug Cutting
   - ☐ Richard Stallman
   - ☐ Alan Cox
10. Combien de giga-octets y a-t-il dans un exaoctet ?
   - ☐ Un millier
   - ☐ Un million
   - ☐ Un milliard

I.7. Exercice

Vous êtes le responsable des ventes d'un site de vente de voitures et vous souhaitez améliorer votre moteur de recommandations afin d'augmenter les ventes du site.
Vous souhaitez bénéficier des atouts du Big Data, mais vous ne savez pas par où commencer.

Question 1 : Présentez la démarche que vous allez suivre pour parvenir au résultat voulu.

Question 2 : Décrivez le processus à suivre pour un déploiement progressif du Big Data à l'échelle de votre unité.

I.8. Solutions

Question 1 :

Nous commencerons par analyser le parcours du prospect sur le site afin d'identifier ses centres d'intérêt. En fonction des interactions ou des recherches effectuées (modèles ou marques de voitures), nous pouvons dégager un modèle comportemental que nous intégrerons dans un groupe d'utilisateurs ayant les mêmes critères de recherche.

Pour inciter le visiteur à revenir sur le site, nous lui demandons son email en contrepartie d'informations sur des modalités de paiement avantageuses ou des remises exceptionnelles.

Cette technique peut être couplée à du retargeting. Dans les communications email, nous lui suggérons des voitures qui correspondent aux critères du segment de clients auquel il appartient, avec à la clé une offre avantageuse. Pour affiner les recommandations, une fois de retour sur le site, on peut lui proposer de configurer sa voiture favorite en échange d'une estimation de prix. On obtient ainsi des détails techniques supplémentaires par rapport à ses préférences.

Question 2 :

Il est important de débuter par la définition d'un cas d'usage justifiant le recours au Big Data (personnalisation des prix, meilleure connaissance client, etc.). Une fois l'objectif identifié, on fait le tour des données internes et externes dont nous disposons. Si nous visons une meilleure connaissance client, les réseaux sociaux constituent des sources de premier choix. Pour l'interconnexion des informations provenant de sources internes et externes, nous utilisons une API.

Enfin, il nous faut un outil capable d'exploiter les métadonnées à notre disposition. Puisque notre budget de départ est restreint et que nous débutons avec un volume de données relativement faible, nous opterons pour une solution NoSQL cloud. Nous faisons ainsi l'économie d'infrastructures Big Data physiques, tout en bénéficiant de services d'analyse de données à un coût abordable.

Chapitre II : Les bases de données NoSQL

II.1. Introduction

Les bases de données NoSQL (Not Only SQL) sont un type de base de données qui diffère des systèmes de gestion de bases de données relationnelles (SGBDR) traditionnels. Contrairement aux SGBDR qui utilisent un modèle de données tabulaire (relations), les bases de données NoSQL permettent une plus grande flexibilité dans la gestion et la structure des données.

Elles sont souvent utilisées dans des environnements où la scalabilité, la flexibilité des schémas, et la performance sont des priorités. Ce type de base de données est particulièrement adapté aux applications modernes, telles que les applications web à grande échelle, les systèmes distribués et le Big Data.

Les bases de données NoSQL ne se limitent pas à une seule technologie ou à une seule méthode de stockage, mais elles incluent une variété de types différents qui répondent à des besoins variés.

II.2. Types de bases de données NoSQL

Les bases de données NoSQL se divisent en quatre modèles différents, chacun adapté à des besoins spécifiques. Ces modèles sont (Figure II.1) : Clé-Valeur, Colonne, Document et graphe). Dans ce qui suit, nous allons présenter en détail ces modèles et les avantages et inconvénients de chaque modèle.

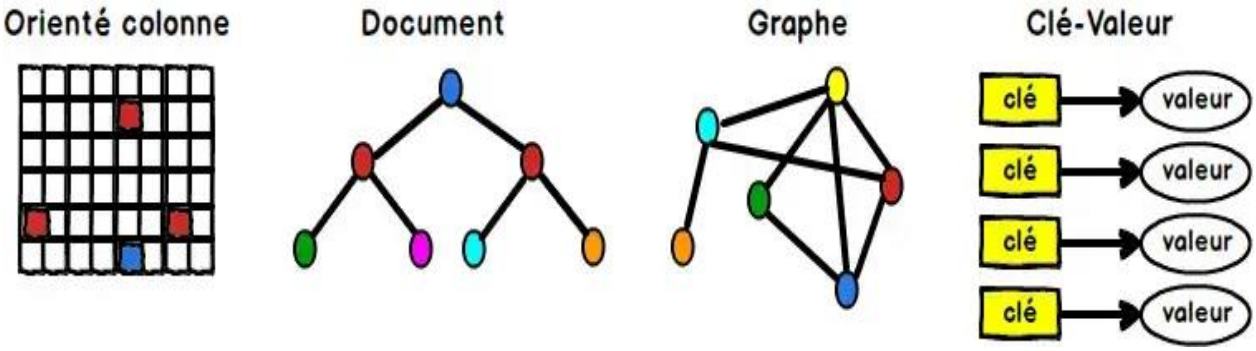


Figure II.1 : Modèles de bases de données NoSQL [2]

- **Bases de données clé-valeur** : C'est le modèle de bases de données NoSQL le plus simple. Les données sont stockées sous forme de paires clé-valeur, où chaque clé unique est associée à une valeur (un nombre, une chaîne de caractères, ou même un objet complexe).

  Ces bases de données sont très efficaces pour les applications où l'accès rapide à des valeurs par clé est nécessaire. Elles sont caractérisées par Performances très élevées pour des lectures et écritures simples.

  Cependant, ce type de bases de données manque de complexité dans les requêtes et l'agrégation de données c'est pour cela, ce modèle est rarement utilisé pour les données complexes.

Exemples : Redis, Amazon DynamoDB.

- **Bases de données de type colonne** : Ces bases de données stockent les données sous forme de colonnes plutôt que de lignes. Elles sont particulièrement adaptées aux applications de Big Data et aux systèmes où les données sont fortement dispersées sur de nombreux nœuds.

  Ces bases de données ont de bonnes performances pour les lectures et écritures et elles sont optimisées. Cependant, ce modèle est moins efficace pour les petites applications et souffre de la complexité accrue durant le traitement des données et des requêtes

  Exemples : Apache Cassandra, HBase.

- **Bases de données orientées documents** : Ces bases de données stockent les données sous forme de documents, généralement en format JSON ou BSON. Chaque document peut être un objet autonome avec une structure flexible. Ce modèle est très populaire pour les applications web, où les données peuvent avoir des structures variées. Ce modèle est performant et flexible dans la structuration des données facilitant ainsi une évolution facile.

  Cependant, ce modèle peut poser des problèmes cohérence dans certains cas ce qui le rend moins adapté aux relations complexes existantes entre les données.

  Exemples : MongoDB, CouchDB.

- **Bases de données orientées graphes** : Ces bases de données sont utilisées pour stocker et manipuler des données fortement interconnectées. Elles modélisent les données sous forme de graphes, où les nœuds représentent des entités et les arêtes représentent les relations entre elles.

  Ce type de base de données est intéressant et performant pour les applications qui nécessitent une gestion de données interconnectées et les requêtes complexes, telles que les réseaux sociaux, les recommandations ou les systèmes d'authentification.

  Exemples : Neo4j.

II.3. Bases de données NoSQL VS Bases de données relationnelles

Les bases de données NoSQL se différencient des SGBDR sur plusieurs aspects. Le tableau suivant représente les principales différences :

Critère	SGBDR	SGBD NoSQL
Modèle de données	Utilisent un modèle relationnel strict, basé sur des tables et des clés primaires/étrangères.	Offrent des modèles de données flexibles, comme des paires clé-valeur, des documents ou des graphes.
Scalabilité	Souvent limitées en termes de scalabilité horizontale.	Conçues pour une scalabilité horizontale, permettant de gérer de grandes quantités de données réparties sur plusieurs serveurs.

Requêtes	Utilisent le langage SQL pour interroger les données, avec une syntaxe et des règles strictes.	N'ont pas de langage standardisé, ce qui peut rendre les requêtes moins cohérentes mais plus flexibles.
Transactions	Garantissent des propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) pour les transactions.	Peuvent offrir une consistance plus faible pour améliorer la disponibilité et la partition des données (selon le théorème CAP), ce qui peut entraîner des compromis en termes de fiabilité pour certaines applications.

Table II.1 : Comparaison bases de données relationnelles et NoSQL

II.3. Conclusion

Les bases de données NoSQL sont devenues un choix privilégié pour de nombreuses applications modernes en raison de leur capacité à gérer des données massives, leur flexibilité et leur évolutivité.

En fonction des besoins spécifiques d'une application (performance, flexibilité du schéma, gestion de données non structurées, etc.), différents types de bases de données NoSQL peuvent être utilisés pour répondre efficacement aux exigences des utilisateurs.
Cependant, comme pour toute technologie, leur adoption nécessite une évaluation précise des cas d'utilisation, des compromis en termes de cohérence, et de la complexité du système à mettre en place.

II.4. Exemple

L’exercice suivant illustre la modélisation d’un modèle SGBD NoSQL orienté document et je justifier le choix.

Il est demandé de concevoir un modèle de données NoSQL orienté document pour un système de gestion de bibliothèque en ligne où des livres peuvent être empruntés par des utilisateurs. Les informations à prendre sont les suivantes :

- Chaque livre a un titre, un auteur, une catégorie (par exemple, fiction, non-fiction, science, etc.), une liste d'évaluations et une liste d'emprunts.
- Chaque utilisateur peut emprunter plusieurs livres et peut laisser des évaluations sur ces livres.
- L'emprunt d'un livre est une opération liée à un utilisateur, et chaque emprunt a une date d'emprunt et une date de retour prévue.

Dans un modèle orienté document (comme MongoDB), les données sont généralement stockées sous forme de documents JSON dans des collections. Chaque document représente une entité et peut contenir des sous-documents et des tableaux. Les relations sont souvent intégrées directement dans les documents.

Pour ce système, nous considérons les collections principales :

- Livres : Cette collection stocke des informations sur les livres.
- Utilisateurs : Cette collection contient les informations des utilisateurs.

Collection Livres Chaque document dans la collection Livres représente un livre avec les informations suivantes :

```
{
    "_id": ObjectId("..."),  // ID unique du livre
    "titre": "Le Seigneur des Anneaux",
    "auteur": "J.R.R. Tolkien",
    "categorie": "Fantasy",
    "evaluations": [
        {
        "utilisateur_id": ObjectId("..."),
        "note": 5,
   "commentaire": "Un livre fantastique.",
            "date": ISODate("2024-12-01")
        },
        {
            "utilisateur_id": ObjectId("..."),
            "note": 4,
            "commentaire": "Très bon livre",
            "date": ISODate("2024-12-02")
```

```
        }
    ],
    "emprunts": [
        {
            "utilisateur_id": ObjectId("..."),
            "date_emprunt": ISODate("2024-12-10"),
            "date_retour": ISODate("2024-12-17")
        },
        {
            "utilisateur_id": ObjectId("..."),
            "date_emprunt": ISODate("2024-12-15"),
            "date_retour": ISODate("2024-12-22")
        }
    ]
}
```

Collection Utilisateurs : Chaque document dans la collection Utilisateurs représente un utilisateur avec les informations suivantes :

```
{
    "_id": ObjectId("..."),  // ID unique de l'utilisateur
    "nom": "John Doe",
    "email": "johndoe@example.com",
    "emprunts": [
        {
            "livre_id": ObjectId("..."),
            "date_emprunt": ISODate("2024-12-10"),
            "date_retour": ISODate("2024-12-17")
        },
        {
            "livre_id": ObjectId("..."),
            "date_emprunt": ISODate("2024-12-15"),
            "date_retour": ISODate("2024-12-22")
        }
    ],
    "evaluations": [
```

```
        {
            "livre_id": ObjectId("..."),
            "note": 5,
            "commentaire": "Excellent livre",
            "date": ISODate("2024-12-01")
        }
    ]
}
```

Pourquoi choisir une modélisation orientée document :

- **Performance** : La modélisation orientée document est rapide pour les requêtes simples, car toutes les informations d'un livre ou d'un utilisateur sont stockées ensemble. Cela évite les jointures complexes.
- **Scalabilité** : Le modèle est facilement scalable, car les données peuvent être réparties sur plusieurs serveurs sans perte de performance.
- **Flexibilité** : Si le schéma de données change (par exemple, ajout de nouveaux champs dans les emprunts ou les évaluations), il est facile de modifier les documents sans perturber les autres utilisateurs ou livres dans la base de données.

Voici quelques exemples de requêtes que l'on pourrait exécuter sur ce modèle de données :

- **Liste des livres empruntés par un utilisateur :**

```
db.utilisateurs.find({ "_id": ObjectId("utilisateur_id") }, { "emprunts": 1 })
```

- **Liste des évaluations d'un livre :**

```
db.livres.find({"_id": ObjectId("livre_id")},{"evaluations": 1 })
```

- **Ajouter une évaluation pour un livre :**

db.livres.update(

```
    { "_id": ObjectId("livre_id") },
    {
      $push: {
        "evaluations": {
          "utilisateur_id": ObjectId("utilisateur_id"),
          "note": 5,
          "commentaire": "Super livre !",
          "date": new Date()
        }
      }
    }
)
```

La modélisation NoSQL orientée document permet une gestion flexible et efficace des données dans le contexte d'une bibliothèque en ligne, en minimisant les jointures complexes et en optimisant l'accès aux données liées à chaque entité.

Dans ce qui suit, je vais présenter quelques travaux pratiques de manipulation de bases de données NoSQL, j'ai choisis comme modèles les plus utilisés : MangoDB et Neo4j.

- **TP1** : Manipulation du SGBD MangoDB
- **TP2** : Manipulation du SGBD Neo4J

TP1 : Manipulation du SGBD MangoDB

Activité 1

MongoDB est un système de gestion de base de données ou SGBD, comme Mysql ou PostgreSQL, mais dont le mécanisme est complètement différent..

Grâce à MongoDB vous allez pouvoir stocker vos données un peu comme vous le feriez dans un fichier JSON. C'est à dire, une sorte de dictionnaire géant composé de clés et de valeurs.

Ces données peuvent ensuite être exploitées par du JavaScript, directement intégré dans MongoDB, mais peuvent également être exploitées par d'autres langages comme le python.

Terminologies SQL	Terminologies MangoDB
Base de données	Base de données
Table	Collection
Ligne	Document
Colonne	Champ
Index	Index
Jointure	Imbrication ou Référence
Clef Primaire (peut être multiple)	Clef Primaire (une seule, et représentée par le champ _id)

Table II.2 : Comparaison entre la terminologie SQL et MangoDB

Pour comprendre le fonctionnement de MangoDB, il est indispensable de commencer par des commandes usuelles. Voilà quelques commandes basiques pour manipuler une base de données NoSQL MangoBD :

- Après avoir installé MangoDB, pour lancer le moteur de base de données : `mongod`
- Afficher toutes les bases de données : `show dbs`

Normalement vous devriez avoir une base **local** propre à mongo et une base **test** :

- Création d'une base de données : `use <nom de la base de données>`

Vous pouvez faire `db` pour voir la base de données courante. Attention, si vous faites **show dbs**, vous ne verrez pas encore votre base. En effet, MangoDB attend d'avoir du contenu pour créer votre base.

- Pour créer une collection, il suffit simplement d'ajouter un patient. Par exemple pour :

```
{
    "nom":"Dupond",
    "prenom":"Jean
    Claude",
    "ddn": new Date('May 18, 1984')
}
```

- Avec la commande :

```
db.patients.insert({"nom":"jay gould", "prenom":"stephen", new Date('May 18, 1984')})
```

La collection ***patients*** se crée automatiquement lors de la première utilisation.

- Pour voir le document : `db.patients.find()`

Notez que MongoDB ajoute automatiquement un **index** si rien n'est spécifié

- Par exemple, nous allons remplir notre collection en répétant cette procédure 50 fois.

```
for ( var i = 0 ; i&lt;50; i++){
db.patients.insert({"nom":"jay gould" , "prenom":"stephen", "age": i})}
```

- Vérifier le nombre de patients : `db.patients.count()`
- Retourner toute la liste de la collection patients : `db.patients.find()`
- Récupérer les patients dont l'âge = 5 : `db.patients.find({age:5})`

- Afficher tous les prénoms commençant par "*j*" :

```
db.patients.find({prenom: /^j*/})
```

- Récupérer les patients dont l'âge est supérieur à 40 :

```
db.patients.find({age:{$gt:40}})
```

$gt est un mot clef de mongo qui veut dire *greater than* (*supérieur à*).

- Récupérer les patients dont l'âge est 5 ou 10 :

```
db.patients.find({age:{$in:[5,10]}})
```

- Récupérer uniquement les noms des patients dont l'âge est supérieur à 40 :

```
db.patients.find({age:{$gt:40}},{"nom":true})
```

- Pour limiter le nombre de résultat à 3 : `db.patients.find().limit(3)`

- Pour ordonner la liste par âge décroissant. -1 pour décroissant et 1 pour croissant :

```
db.patients.find().sort(age:-1)
```

- Modifier la collection : `update(query, update, options)`

- Remplacer tous les prénoms *Ali* par *Mohamed :*

```
db.patients.update({"prenom":"Ali"},{$set:{"prenom":"Mohamed"}}
,{multi:true})
```

- Ajoute une clé *sexe* à tous les patients Mohamed:

```
db.patients.update({prenom:"Mohamed"},{$set:{sexe:"male"}},
{multi:true})
```

- Ajouter un patient *Farid* s'il n'existe pas :

```
db.patients.update({prenom:"Farid"},{$set:{sexe:"male"}},
{upsert:true})
save(document, writeConcern)
```

La différence avec `insert` est que `save`, fait un `update` du document s'il existe déjà:

```
db.patient.save({"prenom":"jean
claude",nom":"Van Damme"})
```

- Supprimer tous les patients qui s'appellent Farid ***:***

```
db.patients.remove({prenom:"Farid"})
```

- Supprimer la collection : `db.patients.drop()`

- Supprimer la base de données :

```
use medical
db.runCommand({dropDatabas : 1});
```

Consignes pour exécuter les requêtes de la série :

1. Se connecter à WINDOWS (ligne de commande)
2. Lancer la commande MongoDB
3. Copiez votre base de données (vous pouvez créer votre propre base de données)

Activités :

Réaliser sur machine les requêtes MongoDB suivantes :

1. Afficher toutes les collections de la base :
2. Afficher tous les documents de la base
3. Compter le nombre de documents de la collection employés
4. Insérer de deux manières différentes deux employés avec les champs nom, prénom et soit prime soit ancienneté
5. Afficher la liste des employés dont le prénom est David
6. Afficher la liste des employés dont le prénom commence ou se termine par D
7. Afficher la liste des personnes dont le prénom commence par D et contient exactement 5 lettres
8. Afficher la liste des personnes dont le prénom commence et se termine par une voyelle
9. Afficher la liste des personnes dont le prénom commence et se termine par une même lettre
10. Afficher le noms et prénom de chaque employé ayant une ancienneté > 10

11. Afficher les nom et adresse complète des employés ayant un attribut rue dans l'objet adresse
12. Incrémenter de 200 la prime des employés ayant déjà le champ prime
13. Afficher les trois premières personnes ayant la plus grande valeur d'ancienneté
14. Regrouper les personnes dont la ville de résidence est Toulouse (afficher nom, prénom et ancienneté)
15. Afficher les personnes dont le prénom commence par M et la ville de résidence est soit Foix soit Bordeaux
16. Mettre à jour l'adresse de Dominique Mani : nouvelle adresse ({numéro : 20, ville : 'Marseille',codepostal : '13015' }). Attention, il n'y aura plus d'attribut rue dans adresse
17. Attribuer une prime de 1 500 à tous les employés n'ayant pas de prime et dont la ville de résidence est différente de Toulouse, Bordeaux et Paris.
18. Remplacer le champ tel, pour les documents ayant un champ tel), par un tableau nommé téléphone contenant la valeur du champ tel (le champ tel est à supprimer)
19. Créer un champ prime pour les documents qui n'en disposent pas et de l'affecter à 100 * nombre de caractère du nom de la ville
20. Créer un champ mail dont la valeur est égale soit à nom.prenom@formation.fr pour les employés ne disposant pas d'un champ téléphone, soit à prenom.nom@formation.fr (nom et prénom sont à remplacer par les vraies valeurs de chaque employé)
21. Calculer et afficher la somme de l'ancienneté pour les employés disposant du même prénom

Activité 2 :

Vous trouverez ci-dessous un script SQL d'une base de données à créer sur machine (utiliser l'outil de votre choix).

```
DROP TABLE IF EXISTS joueurs, matchs, terrains, creneaux ;
```

```
CREATE TABLE joueurs ( id_joueur INT PRIMARY KEY, nom_joueur VAR-
CHAR(255), prenom_joueur VARCHAR(255), login VARCHAR(255) UNIQUE, mdp
VARCHAR(255) );
INSERT INTO joueurs VALUES (1,'Dupont','Alice','alice','1234'),
(2,'Durand','Belina','belina','5694'), (3,'Caron','Camilia','cami-
lia','9478'), (4,'Dupont','Dorine','dorine','1347');
CREATE TABLE matchs ( id_match INT PRIMARY KEY, date_match DATE,
id_creneau INT, id_terrain INT, id_joueur1 INT, id_joueur2 INT, FOR-
EIGN KEY (id_joueur1) REFERENCES joueurs(id_joueur), FOREIGN KEY
(id_joueur2) REFERENCES joueurs(id_joueur), FOREIGN KEY (id_creneau)
REFERENCES creneaux(id_creneau), FOREIGN KEY (id_terrain) REFERENCES
terrains(id_terrain) );
INSERT INTO matchs VALUES (1 ,'2020-08-01', 2, 1, 1, 4), (2 ,'2020-08-
01', 3 ,1 ,2 ,3), (3 ,'2020-08-02', 6 ,2 ,1 ,3), (4 ,'2020-08-02', 7
,2 ,2 ,4), (5 ,'2020-08-08', 3 ,3 ,1 ,2), (6 ,'2020-08-08', 5 ,2 ,3
,4);
CREATE TABLE terrains ( id_terrain INT PRIMARY KEY,
nom_terrain VARCHAR(255), surface VARCHAR(255) );
INSERT INTO terrains VALUES (1,'stade','terre battue'), (2,'gym-
nase','synthétique'), (3,'hangar','terre battue');
CREATE TABLE creneaux ( id_creneau INT PRIMARY KEY, plage_horaire VAR-
CHAR(255) );
INSERT INTO creneaux (id_creneau,plage_horaire) values (1,'8h-9h'),
(2,'9h-10h'), (3,'10h-11h'), (4,'11h-12h'), (5,'12h-13h'), (6,'13h-
14h'), (7,'14h-15h'), (8,'15h-16h'), (9,'16h-17h'), (10,'17h-18h'),
(11,'18h-19h'), (12,'19h-20h');
```

Travail à faire

Exercice 1 :

Proposer une modélisation de cette base de données relationnelle sous la forme d'une base de données MongoDB (base de données basée sur des documents).

- Créer une base de données.
- Pensez au nombre de collections dont vous avez besoin pour mieux représenter cette base de données. Avons-nous besoin d'une ou de plusieurs collections ?

- Créez la ou les collections et insérez les documents.

Exercice 2 :

Lorsque votre base de données est prête, répondez aux mêmes questions posées par M. Millet en utilisant les opérations MongoDB CRUD.

- Ecrire une requête qui renvoie les prénoms des joueurs dont le nom est 'Dupont'.
- Ecrire une requête qui modifie le mot de passe de Dorine Dupont, son nouveau mot de passe étant 1976.
- Ecrire une requête permettant d'ajouter le nouveau membre « Zora MAGID » dont le login est « zora » et le mot de passe 2021.
- Déterminer le jour et la plage horaire du match entre Durand Belina et Caron Camilia.
- Déterminer le nom des deux joueurs qui sont les seuls à avoir joué dans le "hangar".

Exercice 3 :

- Compter le nombre de matchs joués sur chaque terrain.
- Trouver la plage horaire la plus fréquemment utilisée pour les matchs.
- Trouver le joueur qui a joué le moins de matchs.
- Trouver le terrain le plus utilisé pour les matchs

TP2 : Manipulation de Neo4J

L'objectif de ce TP est de vous familiariser avec le système de gestion de bases de données orientées graphes Neo4j. Vous apprendrez à créer des graphes, insérer des données, interroger des graphes via le langage Cypher, et réaliser des analyses de graphes simples.

Au cours de ce TP, vous allez :

- Créer une base de données de graphes simple.
- Ajouter des nœuds et des relations.
- Interroger cette base de données en utilisant Cypher.
- Analyser le graphe pour en extraire des informations utiles.

Pourquoi Neo4j ?

Neo4j est un SGBD NoSQL orienté document des données sous forme de graphes. Contrairement aux bases de données relationnelles classiques, qui organisent les données sous forme de tables, les bases de données de graphes organisent les données sous forme de « nœuds » (représentant des entités) et de relations (représentant des connexions entre ces entités). Ces graphes permettent de naviguer efficacement dans les données, surtout lorsqu'il existe des relations complexes entre les entités.

Les bases de données orientées graphes, comme Neo4j, sont particulièrement efficaces pour gérer des relations complexes entre les données. Elles sont largement utilisées dans des domaines tels que : l'analyse de réseaux sociaux, la gestion de recommandations (par exemple, pour recommander des films ou des produits), la détection de fraudes et la gestion de connaissances complexes dans les systèmes d'information.

Principes de Neo4j :

Dans un graphe Neo4j :

- Nœuds : Ce sont des entités (par exemple, des personnes, des articles, des lieux, etc.). Chaque nœud peut avoir des propriétés (attributs).
- Relations : Ce sont les liens entre les nœuds. Chaque relation a un type et peut également contenir des propriétés.

- Propriétés : Ce sont des informations attachées aux nœuds ou aux relations. Par exemple, une personne peut avoir un nom, une date de naissance, etc.
- Cypher : C'est le langage de requête utilisé par Neo4j, similaire au SQL mais conçu pour interroger des graphes.

Prérequis

- Installation préalable de Neo4j.
- Connaissance de base en SQL et en programmation.

Création d'un Graphe Simple

Nous allons créer un graphe représentant un réseau social simple, où chaque personne est un nœud et les relations entre les personnes sont des "AMIS".

1. Lancer Neo4j. Si vous utilisez Neo4j Desktop, ouvrez l'application et créez un nouveau projet.
2. Créer des nœuds représentant des personnes, ouvrez l'interface de requête et entrez les commandes suivantes pour créer trois personnes (langage Cypher):

```
CREATE (alice:Person {name: 'Alice', age: 30})
CREATE (bob:Person {name: 'Bob', age: 25})
CREATE (charlie:Person {name: 'Charlie', age: 35})
```

Ici, nous avons créé trois nœuds de type `Person` avec des propriétés `name` et `age`.

3. Créer des relations entre les personnes. Pour créer des relations "AMIS" entre ces personnes, entrez la commande suivante :

```
MATCH(alice:Person {name: 'Alice'}), (bob:Person {name: 'Bob'})
CREATE (alice)-[:AMIS]->(bob)
MATCH (bob:Person {name: 'Bob'}), (charlie:Person {name: 'Char-
lie'})
CREATE (bob)-[:AMIS]->(charlie)
```

Cette requête crée une relation `AMIS` entre Alice et Bob, et une relation `AMIS` entre Bob et Charlie.

4. Interroger le Graphe :

- Utilisez cette requête pour afficher tous les nœuds et relations :

```
MATCH (n) RETURN n
```

- Utilisez cette requête pour trouver tous les amis de Bob :

```
MATCH (bob:Person {name: 'Bob'})-[:AMIS]->(ami)
RETURN ami.name
```

Cela retournera les amis de Bob dans la base de données (Alice et Charlie).

- Pour explorer la structure des relations entre les personnes, vous pouvez utiliser cette requête pour afficher les relations avec les types et les propriétés associées :

```
MATCH (a)-[r:AMIS]->(b)
RETURN a.name, r, b.name
```

- Imaginons que chaque relation ait une date de connexion. Vous pouvez modifier la requête pour ajouter une propriété à la relation :

```
MATCH (alice:Person {name: 'Alice'}), (bob:Person {name: 'Bob'})
CREATE (alice)-[:AMIS {since: '2022-01-01'}]->(bob)
```

Cela ajoutera une propriété `since` à la relation `AMIS` entre Alice et Bob.

- Trouver des amis de personnes de plus de 30 ans :

Cette requête trouve tous les amis de personnes dont l'âge est supérieur à 30 ans :

```
MATCH (p:Person)-[:AMIS]->(ami:Person)
WHERE p.age > 30
RETURN ami.name
```

- Calculer le nombre d'amis pour chaque personne :

Cette requête renvoie le nombre d'amis de chaque personne dans le réseau :

```
MATCH (p:Person)-[:AMIS]->(ami)
RETURN p.name, COUNT(ami) AS amis_count
```

Chapitre III : Calcul distribué avec MapReduce

III.1 Introduction

MapReduce est un patron d'architecture de programmation utilisé pour paralléliser des calculs dans un contexte Big Data. Imaginé et développé par Jeff Dean chez Google, il a été repris dans plusieurs projets, dont le très populaire Framework Hadoop qui utilise MapReduce.

Au fil des ans, ce modèle est devenu incontournable pour traiter de très grandes quantités de données. Son modèle, particulièrement robuste, permet d'adapter les calculs à n'importe quelle volumétrie de données, de quelques giga-octets jusqu'à des dizaines de téraoctets de données. En bref, il n'y a pas de limite (si ce n'est le temps de calcul) aux applications de MapReduce.

III.2 Comment fonctionne MapReduce ?

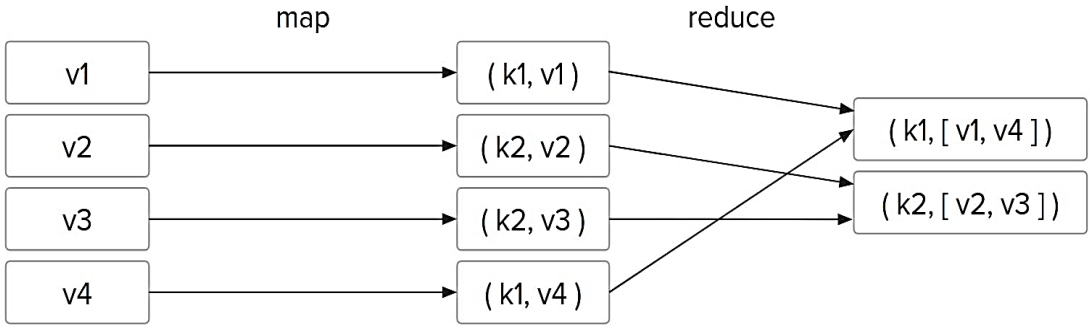
Comme son nom l'indique, MapReduce est constituée de deux étapes : une première appelée Map, et une seconde appelée Reduce. Ces deux étapes sont importantes, car elles permettent de transposer de nombreuses opérations dans un cadre distribué.

- **Étape Map :** L'étape Map va projeter les éléments d'une liste dans une nouvelle liste par l'intermédiaire d'une fonction f que l'on définit nous-même. Les données en entrée (*input*) vont être décomposées sous forme de plusieurs blocs de même tailles et envoyés vers différents **workers** (nœud du cluster qui exécute les calculs).

  Chaque donnée des blocs est « mappé » à partir de la fonction f : son objectif est d'ajouter des informations supplémentaires de sorte à pouvoir créer des groupes de données, bien que celles-ci soient dispersées sur plusieurs serveurs.

  Supposons que l'on ait des valeurs **v1, v2, v3, v4** et que l'on souhaite regrouper des valeurs entre-elles. Par exemple, on peut supposer que v1, v4 appartiennent à un premier groupe, et **v2, v3** à un deuxième groupe.

Nous allons alors mapper chaque valeur initiale des blocs vers une **paire clé-valeur** : la clé indique le groupe auquel la valeur appartient. On obtient **(k1, v1), (k1, v4) et (k2, v2), (k2, v3).**



Le principal avantage de cette représentation est de pouvoir dire que des valeurs appartiennent au même groupe, même si elles se retrouvent dans des blocs différents, et donc sur des serveurs différents.

- **Étape Reduce :** Maintenant que l'on dispose de paires clé-valeur, l'étape Reduce cherche à agréger les valeurs qui possèdent la même clé.

Dans cette étape, nous définissons une stratégie d'agrégation, c'est-à-dire une manière dont nous allons agréger toutes les valeurs pour une même clé. MapReduce va se charger lui-même de déterminer quelles sont les clés identiques et quelles sont les valeurs qui appartiennent au même groupe.

Si l'on reprend les paires précédentes, alors le regroupement des paires donnerait (k1, [v1, v4]) **et** (k2, [v2, v3])**.** À partir de là, il est très facile d'agréger chaque liste : on peut faire une somme, une opération mathématique ou tout autre fonctions, libre à nous de la définir.

Ce procédé permet de créer un lien entre des données situées sur des serveurs différents, alors même que ces derniers sont indépendants. Nous sommes ainsi capables de regrouper des données en utilisant les clés des paires.

Ce qui est crucial dans MapReduce, c'est de choisir les bonnes fonctions Mapper et Reducer à utiliser pour répondre précisément au besoin.

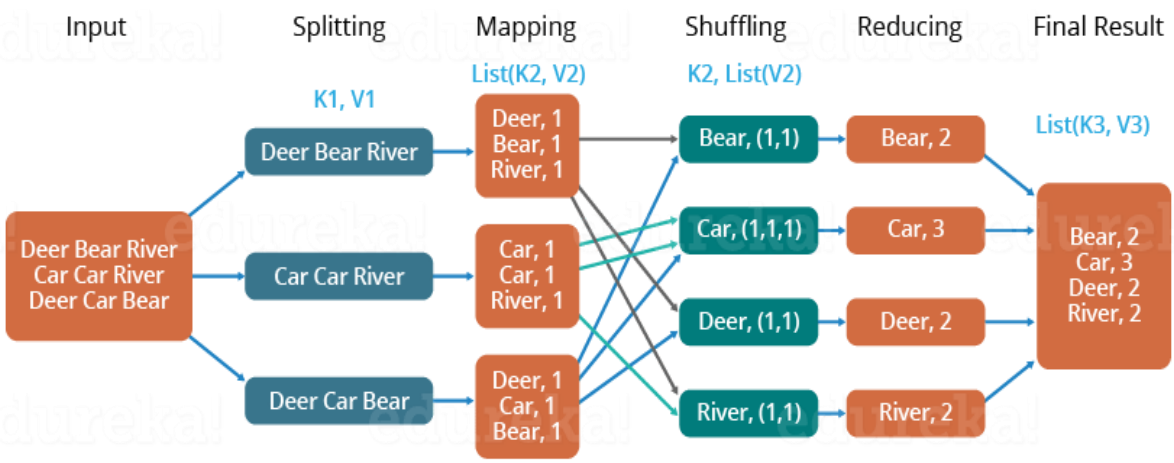


Figure III.1 : Le processus entier du MapReduce à travers un exemple ‘avec les étapes intermédiaires)

Le processus en entier est défini comme suit : le fichier est divisé en bloc (Splitting), chaque bloc est traité par un mapper. Ensuite, les couples générés par l’étape du mapping sont organisés selon la clé (Sfuffling) pour préparer l’étape suivante. L’étape de reduce enfin, prend les couples de même clés et réalise des calculs sur la valeur.

III.3 Exemple

Voici un exemple simple de traitement MapReduce, que l'on peut utiliser pour comprendre comment fonctionne cette approche. L'exemple que je vais expliquer est très courant en Big Data, il consiste à calculer l’occurrence de chaque mot dans un fichier texte de taille importante.

Imaginons un fichier texte contenant des phrases ou des mots séparés par des espaces, et l'objectif est de compter combien de fois chaque mot apparaît.

Voilà un extrait du fichier en entrée : **input.txt**

…….
chat chien chat cheval
chien cheval chat

- **Map :** La fonction Map prend chaque ligne du fichier et la transforme en un couple clé-valeur. La clé est le mot, et la valeur est le nombre `1` (indiquant une occurrence de ce mot).

Etape de Map	chat chien chat cheval	**devient**	(`"chat"`, `1`) (`"chien"`, `1`) (`"chat"`, `1`) (`"cheval"`, `1`)
	chien cheval chat	**devient**	(`"chien"`, `1`) (`"cheval"`, `1`) (`"chat"`, `1`)

Chaque ligne est traitée indépendamment, et chaque mot dans la ligne est extrait pour générer une paire clé-valeur. Cette étape se réalise d'une manière distribuée. La taille du bloc est définie par le système.

- **Shuffling and Sorting** (opération interne) :

  Après le Map, les paires sont regroupées par clé. Tous les mots identiques seront envoyés à la même machine (ou à la même tâche dans un système distribué).

Shuffle and Sort	(`"chat"`, `1`) (`"chien"`, `1`) (`"chat"`, `1`) (`"cheval"`, `1`)	**devient**	(`"chat"`, `1`) (`"chat"`, `1`) (`"chat"`, `1`)
			(`"chien"`, `1`) (`"chien"`, `1`)
	(`"chien"`, `1`) (`"cheval"`, `1`) (`"chat"`, `1`)		(`"cheval"`, `1`) (`"cheval"`, `1`)

- **Reduce** :

  La fonction Reduce reçoit les paires regroupées par clé (c'est-à-dire, pour chaque mot, une liste de `1`), puis elle calcule la somme des valeurs (ce qui donne le nombre d'occurrences de chaque mot).

Cette étape est parallèle, plusieurs reducers travaillent en même temps pour trouver le résultat attendu. En général le nombre de reducers dépasse le nombre de mapper.

(`"chat"`, `1`) (`"chat"`, `1`) (`"chat"`, `1`)	**devient**	(`"chat"`, `3`)
(`"chien"`, `1`) (`"chien"`, `1`)		(`"chien"`, `2`)
(`"cheval"`, `1`) (`"cheval"`, `1`)		(`"cheval"`, `2`)

Après le traitement MapReduce, le résultat final sera :

chat -> 3

chien -> 2

cheval -> 2

III.4 Exercices

Dans ce qui suit, je vous propose quelques exercices à réaliser pour comprendre le fonctionnement du MapReduce à travers des exemples de la vie quotidienne.

Pour vérifier vos résultats, merci de m'envoyer un mail.

Exercice 1

Soit un grand document de type texte. Nous souhaitons réaliser quelques analyses sur son contenu. Plus particulièrement, catégoriser les mots en 4 catégories :

- Minuscule : 1 lettre ;
- Petit : 2 à 4 lettres ;
- Moyen : 5 à 9 lettres ;
- Grand : plus de 10 lettres

Travail à faire :

1. Donner le schéma détaillé des opérations Map-Reduce

2. Ecrire en pseudo code les fonctions Map et Reduce permettant de retourner le nombre de mots dans chaque catégorie.
3. Traduire le pseudo code en langage java

Exercice 2

On reçoit d'un fournisseur une livraison de fruits (pommes, ananas, oranges, etc.). On décide d'effectuer un test qualité en leur donnant 0 ou 1 pour les critères suivants (trop mûr, tâché, déformé, etc). Pour chaque espèce de fruit, on veut calculer la proportion de fruits trop mûrs. Comment faire avec MapReduce ?

- Présentez avec explication) les taches Map-Reduce permettant de retourner le résultat souhaité à l'aide d'un schéma détaillé.
- Ecrivez en pseudo-code (algorithmes) les fonctions Map et Reduce.
- On veut faire le même calcul pour tous les critères. Comment faire ?

Exercice 3

Nous disposons de l'historique de vente sur plusieurs années d'un magasin. Chaque enregistrement de la liste correspond à une vente de plusieurs articles. Les seules données qui nous intéressent en l'occurrence sont les noms des articles, leur prix ainsi que la quanti té vendue.

Notre objectif est d'utiliser MapReduce pour calculer le prix de vente total et la quantité vendue par produit.

Question : Ecrire les algorithmes des fonctions Map et Reduce.

Dans ce qui suit, je vais présenter deux TPs de traitements MapReduce sans utiliser l'environnement Hadoop pour bien comprendre le principe.

TP1 : Job MapReduce en Python sans Hadoop

L'objectif de ce TP est d'implémenter une version simple de MapReduce avec le langage Python sans utiliser l'écosystème Hadoop en utilisant des bibliothèques Python de base comme multiprocessing ou concurrent.futures pour la parallélisassions.

Je vise dans ce premier temps d'expliquer le calcul distribué offert par la technologie Big Data d'une façon assez simple avant d'introduire l'environnement Hadoop.

Je rappelle que l'idée principale de MapReduce est de diviser le travail en deux étapes :

Etape 1 : Map : Appliquer une fonction à chaque élément de l'entrée et créer des couples clés/valeur.

Etape 2 : Reduce : Combiner les résultats partiels du Map pour produire le résultat final.

Je vais utiliser pour ce TP, l'exemple le plus courant des tâches MapReduce qui consiste en le calcul des occurrences des mots dans un texte, le fichier réellement compte une taille très importante. Je suppose que vous avez déjà installé Python sur vos machines.

```
import string
from collections import defaultdict
from concurrent.futures import ProcessPoolExecutor
```

Fonction map : Cette fonction va découper le texte et associer chaque mot à la valeur 1 :

```
def mapper(text_chunk):
    word_count = defaultdict(int)
```

Découper chaque chunk de texte en mots

```
    words=text_chunk.translate(str.maketrans('',                    '',
string.punctuation)).lower().split()
    for word in words:
        word_count[word] += 1
    return word_count
```

Fonction reduce : Cette fonction combine les résultats des mappers

```
def reducer(mapped_data):
    final_count = defaultdict(int)
    for data in mapped_data:
        for word, count in data.items():
            final_count[word] += count
    return final_count
```

Fonction principale qui simule MapReduce

```
def mapreduce(input_data, num_chunks=4):
```

Diviser les données d'entrée en morceaux (chunks)

```
    chunk_size = len(input_data) // num_chunks
chunks = [input_data[i:i + chunk_size] for i in range(0, len(input_data), chunk_size)]
```

Étape Map : Appliquer la fonction mapper à chaque chunk en parallèle

```
    with ProcessPoolExecutor() as executor:
        mapped_results = list(executor.map(mapper, chunks))
```

Étape Reduce : Fusionner les résultats du Map

```
    final_result = reducer(mapped_results)
    return final_result
```

Exemple de texte d'entrée

text = """

MapReduce is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. It is a powerful tool for big data processing.

mapper : La fonction mapper prend un morceau de texte, découpe ce texte en mots, supprime la ponctuation et met les mots en minuscules. Ensuite, elle compte l'occurrence de chaque mot dans le morceau de texte.

reducer : La fonction reducer prend les résultats de tous les mappers et agrège les résultats pour chaque mot en additionnant les comptages.

mapreduce : La fonction principale divise les données d'entrée en morceaux égaux, applique le mappage en parallèle en utilisant un ProcessPoolExecutor (qui permet l'exécution parallèle) et combine les résultats à l'aide du reducer.

Résultat attendu :

Le programme retournera le nombre d'occurrences de chaque mot dans le texte d'entrée :

```
{
 'mapreduce': 2,
 'is': 2,
 'a': 2,
 'programming': 1,
 'model': 1,
 'for': 2,
 'processing': 2,
 'large': 1,
 'data': 2,
 'sets': 1,
 'with': 1,
 'parallel': 1,
 'distributed': 1,
 'algorithm': 1,
 'on': 1,
 'cluster': 1,
 'it': 1,
 'powerful': 1,
 'tool': 1,
 'big': 1
}
```

TP2 : Job MapReduce avec MangoDB sans Hadoop

Objectifs :

Le but de ce TP est d'écrire des fonctions MapReduce pour une base de données Mangodb.

Consigne :

Le langage utilisé est le JavaScript. Il faut donc des notions dans celui-ci pour pouvoir programmer cet algorithme dans MongoDB. Vous pourrez trouver des ressources sur le net.

Au-delà de la syntaxe de JS, il faut noter que les deux fonctions map() et reduce() à passer en paramètre doivent respecter les contraintes suivantes :

- **map()** : la fonction ne prend aucun paramètre, on accède à l'objet analysé via l'opérateur *this*, la fonction peut émettre des couples via la fonction *emit(key, value)* autant de fois que nécessaire dans la fonction

- **reduce()** : la fonction prendre deux paramètres key et values (tableau des valeurs de la clé) elle peut être appeler plusieurs fois pour la même clé, elle doit donc renvoyer une valeur de même type que celles dans le tableau.

On doit passer un troisième paramètre, un littéral JSON, qui représente les options de la fonction. La principale option (out) est la collection dans laquelle le résultat sera placé. Si l'on veut voir le résultat, sans le stocker, il est possible d'indiquer out : { inline: 1 }.

Exemple 1

Dans l'exemple qui suit, nous allons chercher à calculer le nombre d'hommes et de femmes. Tout d'abord, on définit la fonction map(), qui émettra un couple (sexe, 1), pour chaque sportif.

```
var map1 = function() { emit(this.Sexe, 1) };
```

Ensuite, nous définissons la fonction reduce(), qui fera la somme des valeurs (qui seront tous des 1 donc).

```
var red1 = function(key, values) {
    return Array.sum(values);
    };
```

Ensuite, nous faisons appel à la fonction mapReduce() sur notre collection.

```
db.Sportifs.mapReduce( map1,
   red1,
  { out: { inline: 1 } }
)
```

On remarque qu'on a m et M. On peut résoudre ce problème en faisant une transformation (en majuscule ou en minuscule) dans la fonction map().

```
var map1 = function() { emit(this.Sexe.toUpperCase(), 1)
}
```

Au final, on peut aussi tout déclarer dans la fonction mapReduce(), en utilisant le principe des fonctions anonymes.

```
db.Sportifs.mapReduce(
  function() { emit(this.Sexe.toUpperCase(), 1)
  },
  function(key, values) {
    return Array.sum(values);
```

```
  },
  { out: { inline: 1 } }
)
```

Exemple 2

Si on cherche à calculer le nombre de sportifs jouant pour chaque sport, il faut émettre des couples (sport, 1) pour chaque sport joué, pour chaque sportif. Mais il faut d'une part prendre en compte qu'il existe des sportifs dans la base, qui ne joue aucun sport. Et d'autre part que pour certains sportifs, il n'y a qu'un sport, et que dans ce cas, nous avons juste une chaîne et non un tableau.

```
db.Sportifs.mapReduce(
  function() {
    if (this.Sports) { // le sportif joue
      if (typeof this.Sports.Jouer != "string") {
        for (sp
          of
          this.S
          ports.
          Jouer)
          {
          emit(s
          p, 1)
        }
      } else {
        emit(this.Sports.Jouer, 1)
      }
    }
```

```
  },
  function(key, values) {
    return Array.sum(values);
  },
  { out: { inline: 1 } }
)
```

Exercices : Répondez en utilisant le paradigme MapReduce

1. Calculer le nombre de gymnases pour chaque ville
2. Calculer le nombre de séances pour chaque jour de la semaine
3. De même pour chaque sport
4. Calculer la superficie moyenne des gymnases, pour chaque ville, pour cela, vous devez calculer la somme des superficies ET le nombre de gymnase (à émettre dans un même objet et à réduire en tenant compte que ce double aspect)
5. Calculer pour chaque sport, le nombre de séance pour chaque jour de la semaine
    - Il faudra émettre, pour chaque sport, une valeur complexe (littéral JSON pour le jour)
    - Il faudra réfléchir à l'étape de réduction

Chapitre IV : L'écosystème Hadoop

IV. 1 Introduction

Hadoop est actuellement l'écosystème le plus utilisé pour mettre en place un projet Big Data, il est même utilisé par les entreprises de grande envergure comme Google et Yahoo.

Hadoop est un outil open-source conçu pour stocker et traiter des données massives de manière distribuée. Il repose sur trois modules principaux : HDFS pour le stockage, MapReduce pour le traitement et YARN pour la gestion des ressources.
Hadoop est regroupés en deux principaux composants :

- Les nœuds maîtres (master nodes) : Ce sont les nœuds principaux de Hadoop (figure IV.1). Ces nœuds sont des serveurs hébergeant les différents services tels que : la gestion du stockage ou le traitement (NameNode et Secondary NameNode). On y trouve aussi, JobTracker est un processus essentiel impliqué dans l'exécution de MapReduce. Il regroupe les tâches de ce dernier vers des nœuds spécifiques du cluster.

- les nœuds esclaves (slave nodes) : Ce sont des machines, que l'on peut appeler également «nœuds travailleurs» (figure IV.1), qui exécutent les tâches envoyées par les nœuds maîtres. C'est dans ces nœuds que le stockage et le traitement des données de Hadoop a lieu (DataNode).

  On y trouve aussi, **TaskTracker** est un nœud du cluster qui accepte les tâches suivantes : mapper, réduire et mélanger. Il est configuré avec un ensemble d'emplacements, ceux-ci indiquent le nombre de tâches qu'il peut accepter.

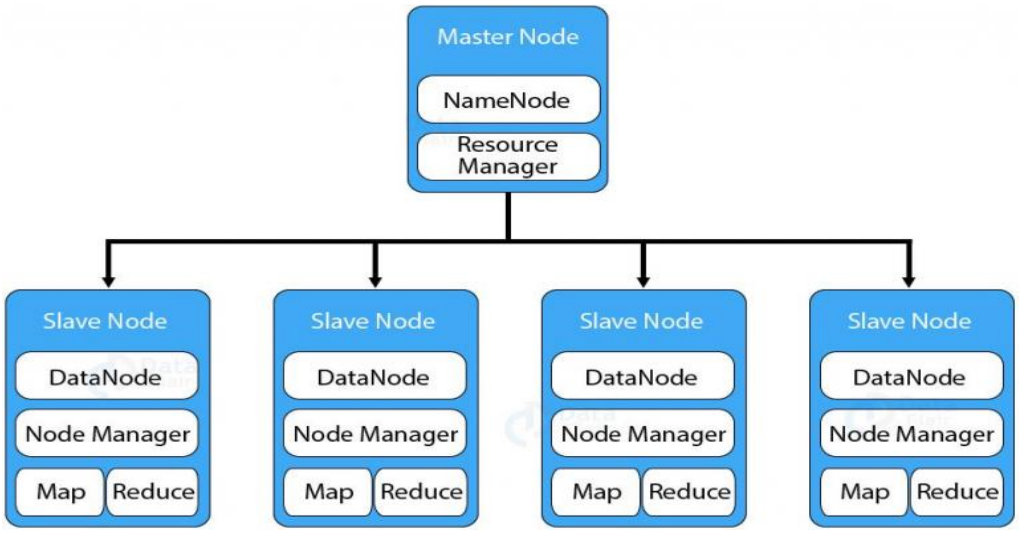


Figure IV.1 : Les Démons de Hadoop

Avec le temps, Hadoop a évolué pour répondre à des besoins croissants en matière de scalabilité, de flexibilité et de performances.

Le tableau suivant représente une comparaison entre les deux principales versions, Hadoop 1.0 et Hadoop 2.0, mettant en lumière les améliorations clés. Il est à noter que Hadoop a beaucoup évolué de la version 1.0 vers la version 2.0, cependant les versions suivantes, on y trouve de légères évolutions par rapport aux outils utilisés.

Critère	**Hadoop 1.0**	**Hadoop 2.0**
Architecture	Modèle classique MapReduce	YARN (Yet Another Resource Negotiator)
Gestion des ressources	Job Tracker gère les Ressources	YARN sépare la gestion des ressources et le Traitement
Scalabilité	Limité auframework MapReduce	Hautement scalable, prend en charge divers modèles de traitement

Cadre de traitement	MapReduce uniquement	Prend en charge plusieurs frameworks (MapReduce, Spark, Tez, etc.)
Planification Des tâches	Capacités de planification Basiques	Amélioration de la planification et de l'allocation des ressources
Multi-tenance	Support limité pour la multi-tenance	Meilleur support pour la multi-tenance
Localité des données	Optimisation basique de la localité	Améliore la localité des données avec des algorithmes sophistiqués
Utilisation du cluster	Utilisation inefficace des ressources	Amélioration de l'utilisation des ressources du cluster
Compatibilité	Limitations de compatibilité	Meilleure compatibilité avec des outils tiers

Table IV.1 : Différence entre Hadoop 1.0 et Hadoop 2.0

Maintenant, je présente une vue générale des deux versions Hadoop 1.0 et Hadoop 2.0 (figure IV.2)

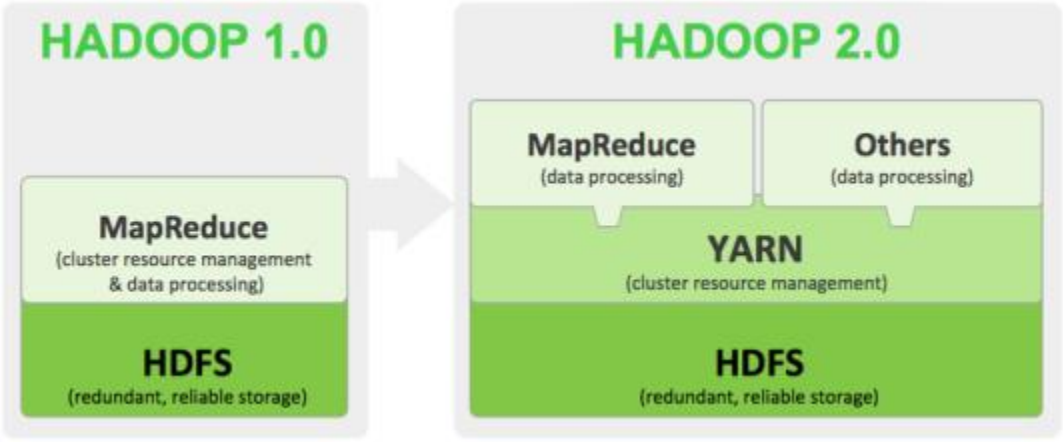


Figure IV.2 : Architecture de Hadoop 1.0 et Hadoop 2.0

IV.2 Principaux éléments de Hadoop :

- **HDFS (Hadoop Distributed File System) :** Un système de fichiers distribué qui permet de stocker de grandes quantités de données de manière fiable (Figure IV.3). Il divise les données en blocs et les réplique pour éviter la perte de données.

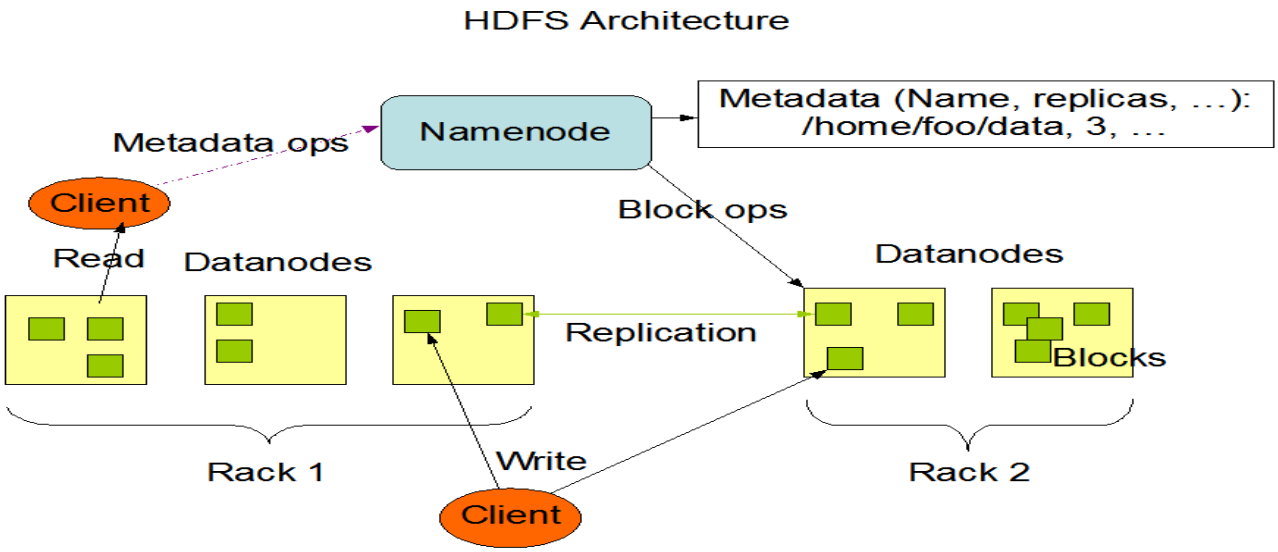


Figure IV.3 : Architecture de HDFS

HDFS utilise un NameNode et un DataNode. Le DataNode est un serveur standard sur lequel les données sont stockées. Le NameNode contient des métadonnées (informations sur les données stockées dans les différents nœuds). L'application interagit uniquement avec le NameNode, et celui-ci communique avec les nœuds de données selon besoin.

Ce qui est important à comprendre, c'est que HDFS divise les données en plusieurs blocs de même taille. Par défaut, la taille d'un bloc est de 128 Mo (Hadoop 2.0 et plus) et 64 Mo (Hadoop 1.0).

Le système de fichiers HDFS utilise une donc cette architecture de réplication pour garantir la disponibilité et la tolérance aux pannes des données. Les blocs de données sont répliqués sur plusieurs DataNodes pour réduire les risques de perte de données en

cas de panne de nœud ou de disque. Le nombre de réplications peut être configuré selon les besoins de tolérance aux pannes et de performances.

Par exemple, supposons que l'on ait un cluster avec 1 NameNode et 3 DataNodes. Si l'on y ajoute un fichier qui pèse 500 Mo, alors il y aura 3 blocs de 128 Mo et 1 bloc de 116 Mo (500 Mo - 3 x 128 Mo). Ainsi, 1 DataNode stockera 2 blocs, et chacun des deux autres stockera un seul bloc.

Et ce découpage par blocs est l'une des fonctionnalités clés qui permet de stocker et de gérer de grands fichiers de manière efficace et distribuée.

La structure de répertoire de HDFS est similaire à celle des systèmes de fichiers locaux sur les systèmes Unix/Linux. La structure de répertoire de HDFS est hiérarchique et est représentée par une arborescence de dossiers et de fichiers.

Le répertoire racine de HDFS est représenté par le caractère /. Tous les autres répertoires et fichiers sont créés sous ce répertoire racine. Les utilisateurs peuvent créer leurs propres dossiers personnalisés pour stocker leurs données et leurs applications dans HDFS.

```
/
|-- user
|   |-- blent
|   |   |-- hello.txt
|   |   `-- world.txt
|   |-- maxime
|   |   |-- data.csv
|   |   `-- code.sql
|-- tmp
|-- hadoop
|   `-- mapred
|       `-- system
```

```
|-- hbase
|-- hive
|-- user
|   `-- history
`-- var
```

Les applications de l'écosystème Hadoop, comme Apache HBase ou Apache Hive, vont eux aussi avoir leur dossier spécifique. C'est notamment à l'intérieur de ces dossiers que sont situées les tables de données.

Ainsi, chaque fichier peut donc être référencé par son chemin sur HDFS**.** Par exemple, si le nom de domaine du nœud principal HDFS est mon-adresse, alors le fichier data.csv représentée plus haut aura pour chemin HDFS :

hdfs://mon-adresse/user/maxime/data.csv.

Ainsi, toute application capable de lire sur un système HDFS pourra lire le contenu du fichier.

Cependant, malgré ses avantages, l'installation et la configuration de HDFS peuvent être coûteuses pour les entreprises en raison du coût des serveurs, des logiciels et des licences nécessaires pour exécuter et gérer le système.

De plus, HDFS peut avoir une latence élevée lorsqu'il s'agit d'accéder aux fichiers stockés dans le système de fichiers, en particulier pour les fichiers de petite taille, ce qui peut rendre le traitement de données en temps réel plus difficile.

Par ailleurs, HDFS peut avoir une faible efficacité pour les petits fichiers, car chaque fichier est divisé en blocs de taille fixe. Si un petit fichier est stocké sur HDFS, il peut occuper un espace disque important par rapport à sa taille réelle, ce qui peut entraîner une utilisation inefficace des ressources de stockage.

- **MapReduce :** C’est un modèle de programmation pour le traitement de données massives. MapReduce Fonctionne en deux étapes : "Map" pour transformer les données et "Reduce" pour les agréger. Cet élément a été présenté en détail au chapitre précédent.

- **YARN :** C'est est l'abréviation de « Yet Another Resource Negotiator » (plus simplement, un négociateur de ressources). Cet élément assure la gestion et planification des ressources (clusters) Hadoop et décide de ce qui doit se passer dans chaque nœud de données. Le nœud principal central qui gère toutes les demandes de traitement est le « Resource Manager ». Le Resource Manager interagit avec les différents Node Managers : chaque DataNode enfant possède son propre Node Manager pour l'exécution des tâches.

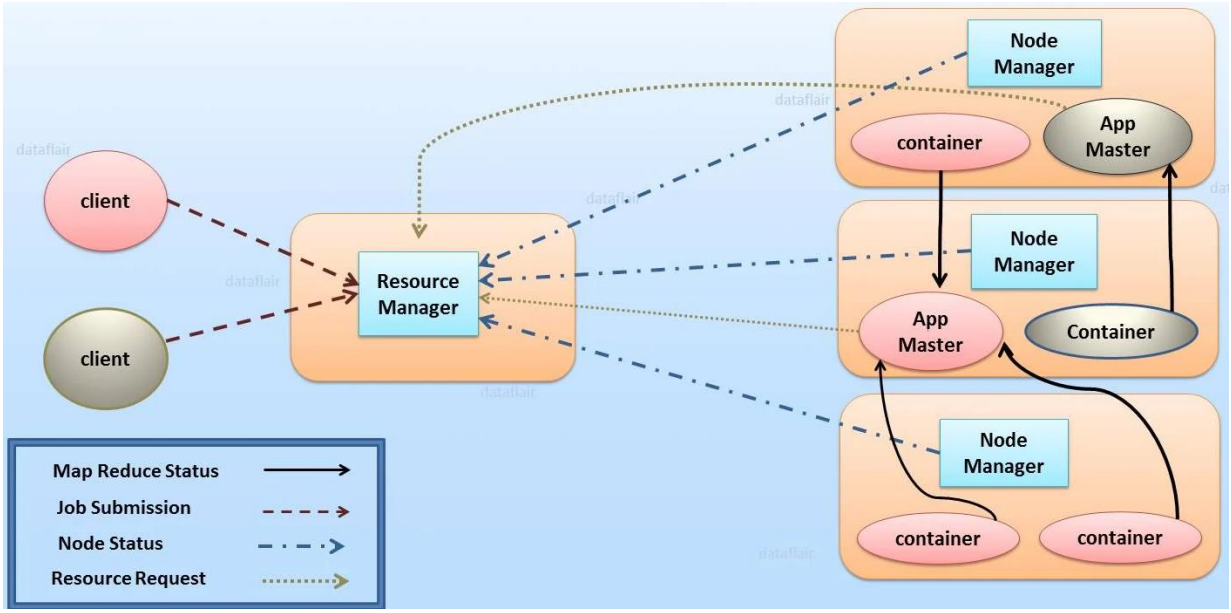


Figure IV.4 : Architecture et fonctionnement de YARN Hadoop

Il existe de nombreux outils basé sur un ou plusieurs composants de Hadoop. Voilà quelques exemples :

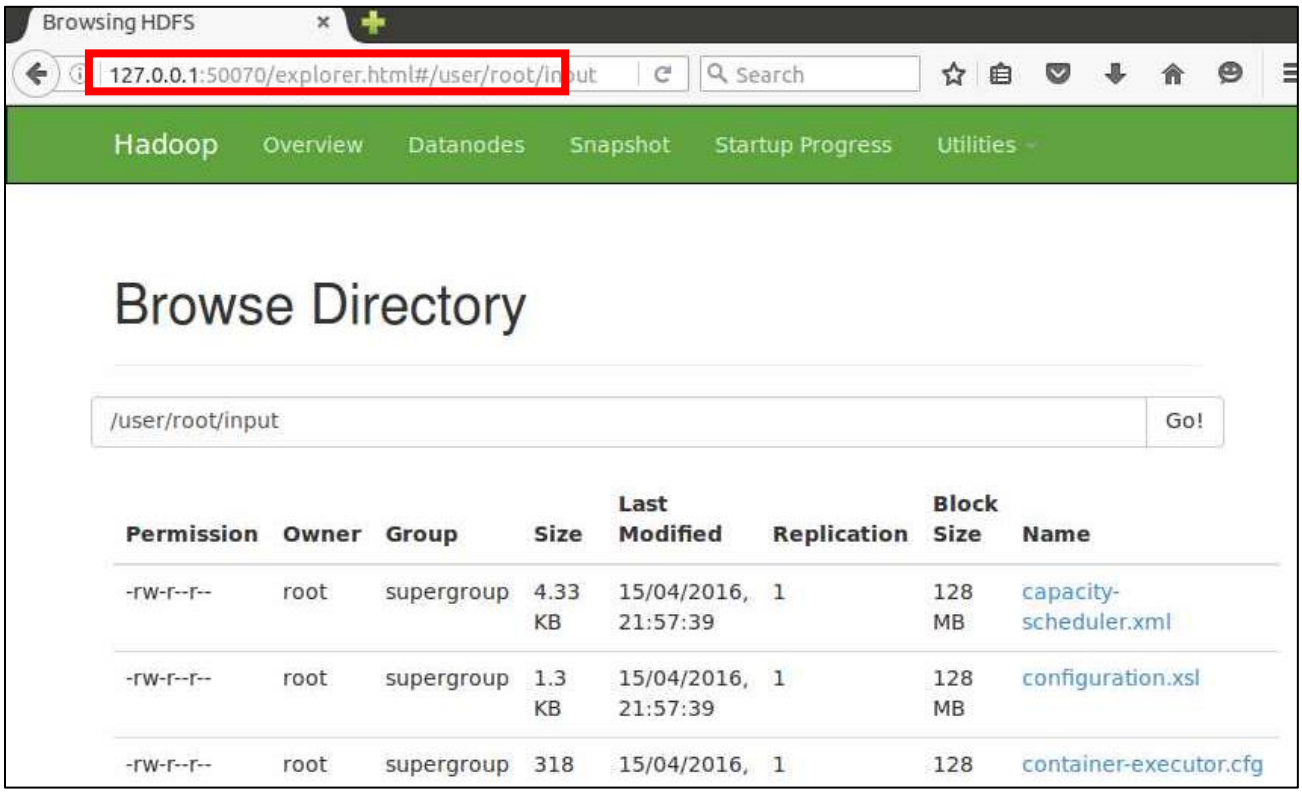
- **HBase**, une base de données NoSQL basée sur HDFS ;
- **Hive**, une base de données relationnelle basée sur Hadoop, utilisable en SQL et accessible avec JDBC ;
- **Mahout**, un outil logiciel basé sur Hadoop fournissant un framework et de nombreux algorithmes déjà implémentés pour effectuer du machine learning en se basant sur HDFS et MapReducs ;

- **Pig**, un outil de scripting basé sur Hadoop permettant de manipuler aisément de grandes quantités de données avec un langage proche du Python ou Bash ;
- **Oozie**, une interface Web de gestion des jobs Hadoop pour les lancer et les les planifier aisément en incluant les notions de dépendances de jobs à d'autres jobs ;

IV. 3 Accéder aux éléments de Hadoop :

Il existe différentes manières pour accéder aux éléments de Hadoop, en lignes commande ou avec des liens web. Seulement, l'utilisation des liens web semble la méthode la plus facile et rapide, je vais présenter dans ce qui suit, quelques liens web de Hadoop.

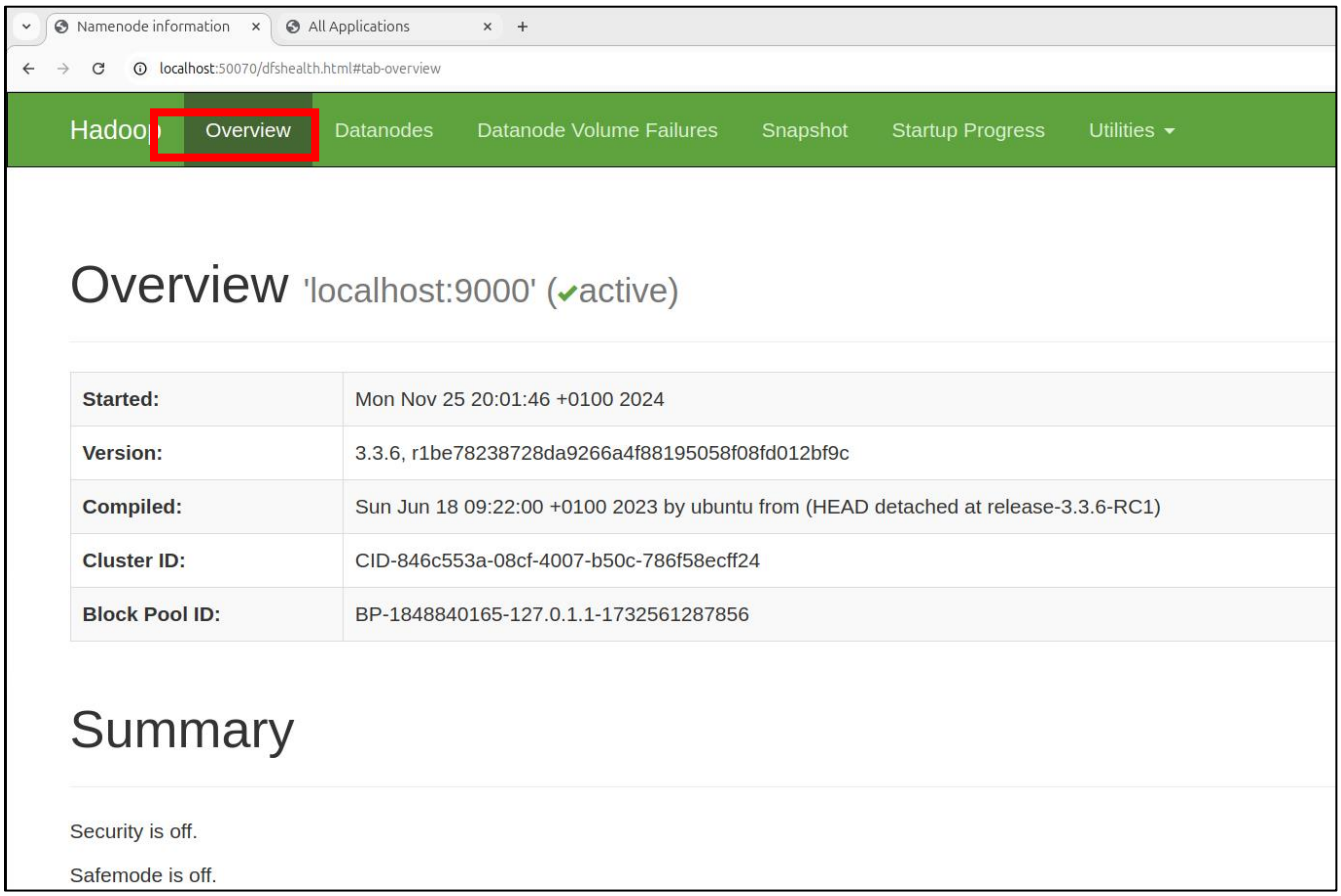
- **Accéder à HDFS :** via http://localhost:50070 pour visualiser l'état du HDFS



Depuis le navigateur, vous pouvez naviguer dans les répertoires de HDFS, voir les informations sur les blocs des fichiers et télécharger des fichiers entiers. C'est un outil

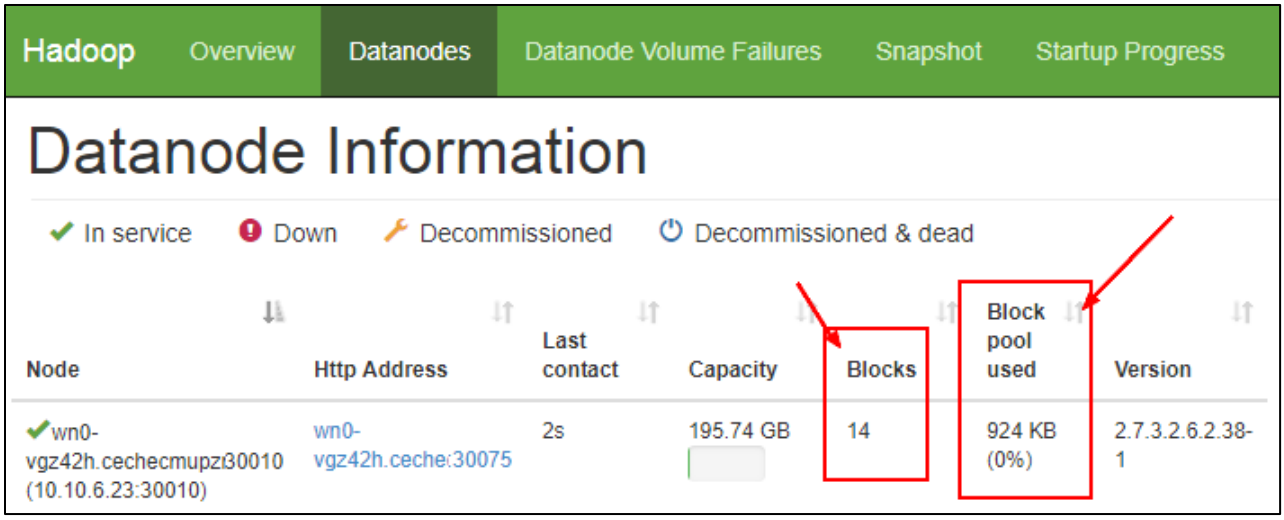
utile pour naviguer dans HDFS, d'autant plus qu'il est en lecture seule, de sorte que l'interface web peut être donnée aux utilisateurs qui n'ont pas accès à Hadoop.

Dans la première ligne de la page Vue d'ensemble, la valeur à droite de Vue d'ensemble indique le nom d'hôte du nœud sur lequel le NameNode est déployé. La valeur active ou standby entre parenthèses () indique si le NameNode est en état actif ou standby dans un cluster de haute disponibilité (HA). Le tableau suivant décrit les paramètres.



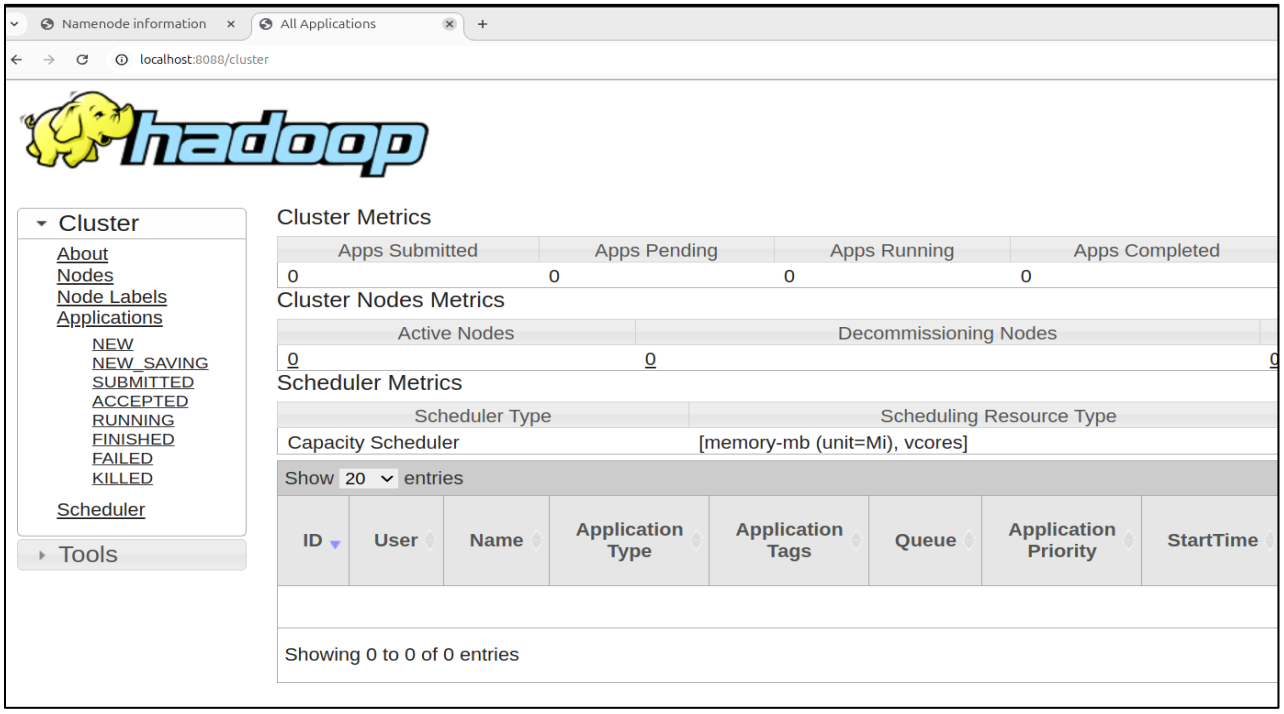
Cet écran permet de visualiser :

- Les fichiers dans HDFS.
- L'état des DataNodes et des blocs.
- La capacité de stockage utilisée.



La taille des blocs peut être modifiée par fichier. Les blocs sont stockés sur un datanode et sont regroupés dans un pool de blocs.

- **Accéder à YARN :** via **http://localhost:8088** pour suivre les ressources allouées aux jobs MapReduce et autres applications.

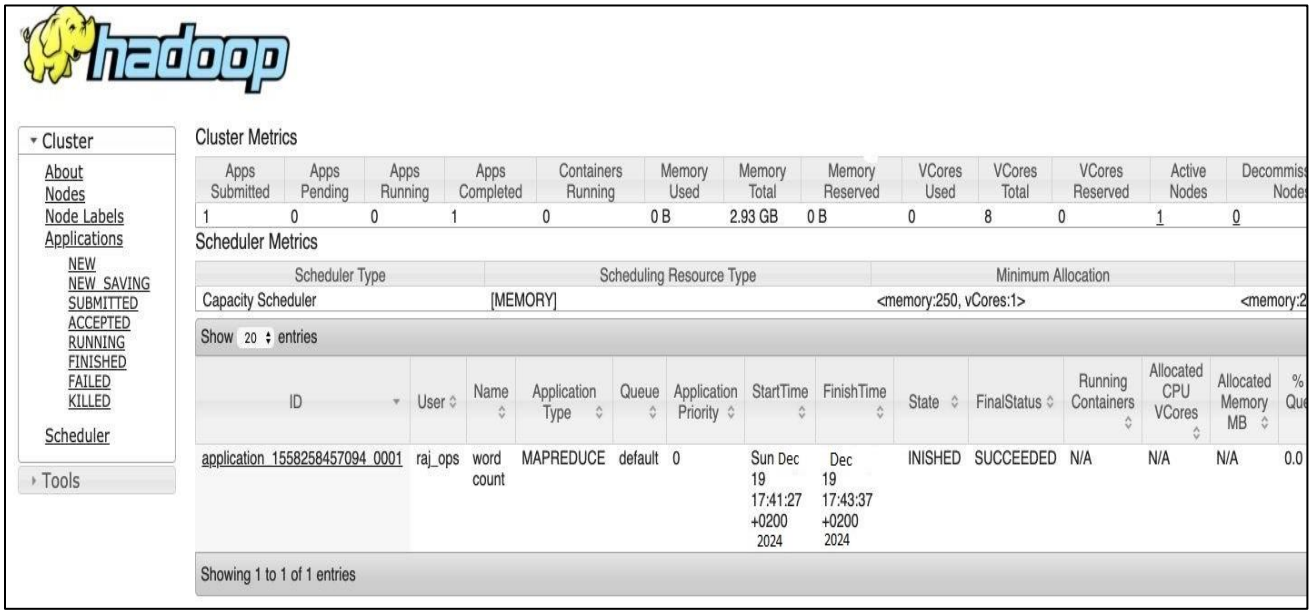


Comme HDFS, les différents services YARN incluent des serveurs web intégrés que vous pouvez utiliser pour vérifier l'état du travail sur le cluster.
Cet écran permet de suivre :

- Les tâches en cours (jobs).
- L'utilisation des ressources (mémoire, CPU).
- Les statuts des applications MapReduce.

Le type d'application indique le moteur qui a soumis l'application YARN. L'identifiant de l'application est l'identifiant unique de l'application. L'ID de l'application est un lien vers le résumé de l'application. L'URL est la même que l'URL de surveillance dans l'outil Administrateur.
Cliquez sur le lien Logs dans le résumé de l'application pour afficher les journaux de l'application sur le cluster Hadoop.
La quantité d'informations contenues dans les journaux d'application dépend du niveau de suivi que vous configurez pour un mappage dans l'outil Développeur.

- **Accéder à MapReduce :** Le Web UI de MapReduce est intégré à l'interface de YARN, accessible via le ResourceManager. Par défaut, l'URL est :http://localhost:8088



Cet écran permet de :

- Suivre l'exécution des jobs MapReduce : surveiller leur état (en cours, terminés, échoués).
- Analyser les performances : afficher les temps d'exécution et les statistiques sur les tâches "Map" et "Reduce".
- Diagnostiquer les erreurs : accéder aux logs des jobs pour résoudre les problèmes.
- Visualiser les étapes du job : détailler les phases de traitement comme le "Map", le "Shuffle", et le "Reduce".

IV. 4 Fichiers de configuration de Hadoop :

Les fichiers de configuration Hadoop sont localisés dans le dossier **HADOOP_HOME/etc/hadoop**. Les plus importants : hadoop-env.sh, core-site.xml, hdfs-site.xml, mapred-site.xml et yarn-site.xml.

- **Fichier core-site.xml :** Configure les propriétés générales comme le nom du serveur HDFS (par défaut : fs.defaultFS). Le fichier core-site.xml permet d'indiquer l'host et le port du système de fichier HDFS.

```xml
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

- **Fichier hdfs-site.xml :** Configure les paramètres spécifiques au HDFS, comme les répertoires pour le stockage des données (dfs.namenode.name.dir).

Ce fichier configure où le NameNode va stocker l'historique des transactions et où les DataNode vont stocker leurs blocks. C'est également ici où le coefficient de réplication est configuré.

```xml
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>~/bigdata/hadoop-3.3.0/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>~/bigdata/hadoop-3.3.0/data/datanode</value>
  </property>
</configuration>
```

Par défaut replication = 3

- **Fichier mapred-site.xml :** Configure les propriétés liées à MapReduce, comme l'emplacement du jobtracker.

```
<configuration>
  <property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
  </property>
  <property>
      <name>mapreduce.application.classpath</name>
      <value>~/bigdata/Hadoop-
3.3.0/share/hadoop/mapreduce/*: ~/bigdata/Hadoop-3.3.0
/share/hadoop/mapreduce/lib/*</value>
  </property>
</configuration>
```

- **Fichier yarn-site.xml :** Configure YARN pour la gestion des ressources.

Les options de configuration YARN sont stockées dans le fichier **/opt/mapr/hadoop/hadoop-2.x.x/etc/hadoop/yarn-site.xml** et sont modifiables par l'utilisateur root. Ce fichier contient des informations de configuration qui remplacent les valeurs par défaut des paramètres YARN. Les valeurs par défaut des propriétés de configuration de base sont stockées dans le fichier Default YARN **parameters**.

```
<configuration>
  <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
  </property>
  <property>
      <name>yarn.nodemanager.env-whitelist</name>
      <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DI
R,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
  </property>
</configuration>
```

Pour remplacer la valeur par défaut d'une propriété, spécifiez la nouvelle valeur dans les balises <configuration>, en utilisant le format suivant :

```
<property>
 <name> </name>
   <value> </value>
```

```
 <description> </description>
</property>
```

IV. 5 Principaux avantages et inconvénients d'Hadoop

Malgré sa facilité et sa simplicité et sa célébrité, beaucoup de géants du Web utilisent Hadoop, seulement, il présente quelques inconvénients. Le tableau suivant met l'accent sur les avantages et les inconvénients de Hadoop.

Avantages	Inconvénients
Traitement de grandes quantités de données : Gère efficacement des volumes massifs de données.	**Courbe d'apprentissage élevée** : Nécessite une expertise technique pour la configuration et l'optimisation.
Évolutivité horizontale : Ajout facile de nouveaux nœuds pour augmenter la capacité.	**Coût opérationnel** : Demande des ressources humaines spécialisées pour la gestion et la maintenance.
Coût réduit : Utilise du matériel standard et est open-source.	**Pas adapté aux petites données** : Peu efficace pour les petits ensembles de données.
Tolérance aux pannes **:** Réplication des données pour éviter les pertes**.**	**Latence élevée** : Fonctionnement par lots, inadapté pour le traitement en temps réel.
Flexibilité : Compatible avec divers types de données (structurées, non structurées).	**Consommation de ressources** : Gourmand en CPU, mémoire et disque pour les tâches complexes
Écosystème riche : Intègre des outils variés (Hive, Pig, Spark, etc.).	**Sécurité limitée** : Nécessite des outils supplémentaires pour une sécurité avancée.
Open-source : Gratuit et soutenu par une communauté active.	**Complexité de gestion** : Difficulté à gérer des clusters avec de nombreux nœuds.

Table IV.2 : Principaux avantages et inconvénients d'Hadoop

IV. 6 Conclusion :

L'installation d'Hadoop dans un environnement virtualisé, comme décrit dans ce projet, représente une étape cruciale pour comprendre les fondements du Big Data. Grâce à cette configuration, nous avons pu explorer les principaux composants d'Hadoop, notamment HDFS pour le stockage distribué, MapReduce pour le traitement des données, et YARN pour la gestion des ressources.

Dans ce qui suit, je vais présenter des travaux pratiques sur comment installer, configurer Hadoop. Ces TPs permettront de démontrer les avantages d'Hadoop dans la gestion et le traitement efficace de grandes quantités de données.

De plus, l'utilisation d'outils comme VirtualBox et Ubuntu fournira un environnement flexible et sécurisé pour tester et comprendre ces technologies sans affecter le système hôte.

En conclusion, cette expérience pratique offrira une base solide pour aborder des projets plus complexes et illustre le potentiel d'Hadoop dans des contextes variés, tels que l'analyse de données en temps réel, l'optimisation de l'utilisation des ressources, et la mise en place de systèmes scalables adaptés aux besoins croissants des entreprises modernes.

TP1 : Installation de Hadoop avec simple nœud

L'installation d'Hadoop dans un environnement virtualisé est une étape essentielle pour comprendre et exploiter les technologies du Big Data. Ce TP détaille les étapes pour configurer un système Hadoop simple nœud à l'aide de VirtualBox et Ubuntu, offrant ainsi un environnement isolé et pratique pour expérimenter ses fonctionnalités. À travers cette installation, nous explorons les différents composants d'Hadoop, notamment HDFS, MapReduce et YARN, tout en apprenant à interagir avec ses interfaces web et à exécuter des scripts de traitement distribué.

Technologies utilisées :

- **VirtualBox** : Logiciel qui permet d'exécuter plusieurs systèmes d'exploitation sur une seule machine. Il facilite la création d'un environnement isolé pour installer Hadoop.
- **Combinaison VirtualBox et Ubuntu** : Permet de tester Hadoop dans un environnement sécurisé, sans modifier la machine principale, tout en offrant la possibilité de sauvegarder ou cloner les configurations facilement.

Dans ce qui suit, je suppose que vous avez préparé votre machine, vous avez installé la virtualisation et vous avez installé le système d'exploitation Ubuntu. J'ai fait appel à la virtualisation avec VirtualBox pour deux raisons, le matériel utilisé pour l'installation de Hadoop n'est pas aussi performant, et d'un autre cité pour sécuriser les données des ordinateurs des étudiants en cas de mauvaise manœuvres

Installation de Hadoop :

- Mettre à jour les fichiers de dépôt avec sudo apt-get update :

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

alaa-salima@alaa-salima-VirtualBox:~$ sudo apt-get update
[sudo] password for alaa-salima:
Hit:1 http://security.ubuntu.com/ubuntu oracular-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu oracular InRelease
Hit:3 http://archive.ubuntu.com/ubuntu oracular-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu oracular-backports InRelease
Reading package lists... Done
alaa-salima@alaa-salima-VirtualBox:~$
```

- Installer java avec sudo apt-get install default-jdk :

```
alaa-salima@alaa-salima-VirtualBox:~$ sudo apt-get install default-jdk.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'default-jdk-doc' for regex 'default-jdk.'
Note, selecting 'default-jdk-headless' for regex 'default-jdk.'
The following packages were automatically installed and are no longer required:
  linux-headers-6.11.0-8 linux-headers-6.11.0-8-generic
  linux-modules-6.11.0-8-generic linux-modules-extra-6.11.0-8-generic
  linux-tools-6.11.0-8 linux-tools-6.11.0-8-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ca-certificates-java default-jre-headless java-common javascript-common
  libjs-jquery libjs-jquery-ui libjs-jquery-ui-theme-base openjdk-21-doc
  openjdk-21-jdk-headless openjdk-21-jre-headless
Suggested packages:
  default-jdk default-jre apache2 | lighttpd | httpd libjs-jquery-ui-docs
  openjdk-21-jdk openjdk-21-demo openjdk-21-source fonts-dejavu-extra
  fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
  | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  ca-certificates-java default-jdk-doc default-jdk-headless
  default-jre-headless java-common javascript-common libjs-jquery
  libjs-jquery-ui libjs-jquery-ui-theme-base openjdk-21-doc
  openjdk-21-jdk-headless openjdk-21-jre-headless
0 upgraded, 12 newly installed, 0 to remove and 61 not upgraded.
Need to get 141 MB/142 MB of archives.
After this operation, 593 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu oracular-updates/universe amd64 openjdk-21-doc all 21.0.5+11-1ubuntu1~24.10 [12.0 MB]
Get:2 http://archive.ubuntu.com/ubuntu oracular/universe amd64 default-jdk-doc amd64 2:1.21-76 [3,244 B]
Get:3 http://archive.ubuntu.com/ubuntu oracular/main amd64 java-common all 0.76 [6,852 B]
Get:4 http://archive.ubuntu.com/ubuntu oracular-updates/main amd64 openjdk-21-jre-headless amd64 21.0.5+11-1ubuntu1~24.10 [46.4 MB]
Get:5 http://archive.ubuntu.com/ubuntu oracular/main amd64 default-jre-headless amd64 2:1.21-76 [3,178 B]
Get:6 http://archive.ubuntu.com/ubuntu oracular-updates/main amd64 openjdk-21-jdk-headless amd64 21.0.5+11-1ubuntu1~24.10 [82.8 MB]
Get:7 http://archive.ubuntu.com/ubuntu oracular/main amd64 default-jdk-headless amd64 2:1.21-76 [966 B]
Get:8 http://archive.ubuntu.com/ubuntu oracular/main amd64 javascript-common all 11+nmu1 [5,936 B]
Fetched 141 MB in 9min 0s (262 kB/s)
Selecting previously unselected package ca-certificates-java.
```

- Vérifier l'installation de java avec java –version

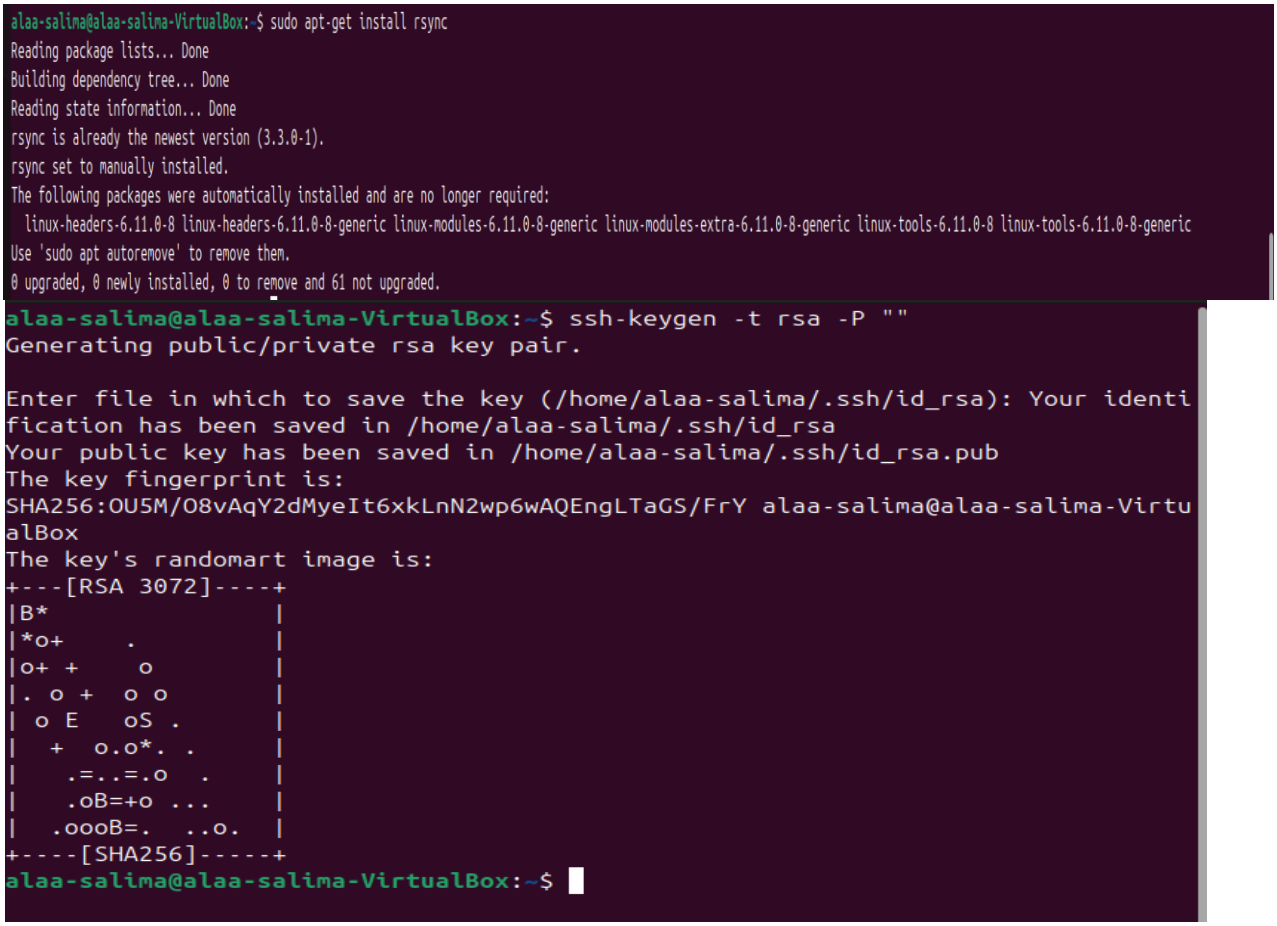
```
alaa-salima@alaa-salima-VirtualBox:~$ java -version
openjdk version "21.0.5" 2024-10-15
OpenJDK Runtime Environment (build 21.0.5+11-Ubuntu-1ubuntu124.10)
OpenJDK 64-Bit Server VM (build 21.0.5+11-Ubuntu-1ubuntu124.10, mixed mode, sharing)
alaa-salima@alaa-salima-VirtualBox:~$ sudo apt-get install ssh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-6.11.0-8 linux-headers-6.11.0-8-generic linux-modules-6.11.0-8-generic linux-modules-extra-6.11.0-8-generic linux-tools-6.11.0-8 linux-tools-6.11.0-8-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh ssh-import-id
0 upgraded, 5 newly installed, 0 to remove and 61 not upgraded.
Need to get 841 kB of archives.
After this operation, 6,838 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu oracular/main amd64 openssh-sftp-server amd64 1:9.7p1-7ubuntu4 [37.1 kB]
Get:2 http://archive.ubuntu.com/ubuntu oracular/main amd64 openssh-server amd64 1:9.7p1-7ubuntu4 [514 kB]
Get:3 http://archive.ubuntu.com/ubuntu oracular/main amd64 ncurses-term all 6.5-2 [275 kB]
Get:4 http://archive.ubuntu.com/ubuntu oracular/main amd64 ssh all 1:9.7p1-7ubuntu4 [4,656 B]
Get:5 http://archive.ubuntu.com/ubuntu oracular/main amd64 ssh-import-id all 5.11-0ubuntu2 [10.0 kB]
Fetched 841 kB in 9s (90.1 kB/s)
Preconfiguring packages ...
Selecting previously unselected package openssh-sftp-server.
(Reading database ... 202776 files and directories currently installed.)
Preparing to unpack .../openssh-sftp-server_1%3a9.7p1-7ubuntu4_amd64.deb ...
Unpacking openssh-sftp-server (1:9.7p1-7ubuntu4) ...
Selecting previously unselected package openssh-server.
Preparing to unpack .../openssh-server_1%3a9.7p1-7ubuntu4_amd64.deb ...
Unpacking openssh-server (1:9.7p1-7ubuntu4) ...
Selecting previously unselected package ncurses-term.
Preparing to unpack .../ncurses-term_6.5-2_all.deb ...
Unpacking ncurses-term (6.5-2) ...
Selecting previously unselected package ssh.
```

- Installer ssh avec sudo apt-get install ssh

```
alaa-salima@alaa-salima-VirtualBox:~$ ls ~/.ssh/
authorized_keys  id_rsa  id_rsa.pub
alaa-salima@alaa-salima-VirtualBox:~$
```

- Installer rsync avec sudo apt-get install rsync

- Générer une clé de cryptage avec ssh-keygen -t rsa -P ""



- Copier cette clé dans « authorized_keys » avec cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys et Vérifier l'installation avec ssh localhost.

```
alaa-salima@alaa-salima-VirtualBox:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized
_keys
alaa-salima@alaa-salima-VirtualBox:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:BAwrowVT2QAkKqX+54+tBnLbO8E6z1x05FDkLH78xyc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.10 (GNU/Linux 6.11.0-9-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

52 updates can be applied immediately.
```

- Télécharger une version de Hadoop et Accéder au répertoire Téléchargement et extraire le dossier zippé avec sudo tar -zxvf hadoop

```
alaa-salima@alaa-salima-VirtualBox:~$ wget https://downloads.apache.org/hadoop/c
ommon/hadoop-3.3.6/hadoop-3.3.6.tar.gz
--2024-11-21 14:55:03--  https://downloads.apache.org/hadoop/common/hadoop-3.3.6
/hadoop-3.3.6.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.208.237, 135.181.
214.104, 2a01:4f8:10a:39da::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|88.99.208.237|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 730107476 (696M) [application/x-gzip]
Saving to: ‘hadoop-3.3.6.tar.gz’

hadoop-3.3.6.tar.gz 100%[===================>] 696.28M  1.11MB/s    in 24m 15s

2024-11-21 15:19:20 (490 KB/s) - ‘hadoop-3.3.6.tar.gz’ saved [730107476/73010747
6]

alaa-salima@alaa-salima-VirtualBox:~$
```

```
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Null.java
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.8.3.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.3.5.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.8.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.3.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/hadoop-hdfs_0.22.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.9.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.1.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/hadoop-hdfs_0.20.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.0-alpha4.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.9.2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.0-alpha2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.10.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.1.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.4.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/hadoop-hdfs_0.21.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.1.3.xml
hadoop-3.3.6/share/hadoop/hdfs/hadoop-hdfs-client-3.3.6-tests.jar
hadoop-3.3.6/share/hadoop/hdfs/hadoop-hdfs-httpfs-3.3.6.jar
alaa-salima@alaa-salima-VirtualBox:~$
```

- Créer le dossier hadoop dans /usr/local avec sudo mkdir /usr/local/hadoop :

```
alaa-salima@alaa-salima-VirtualBox:~$ sudo mkdir /usr/local/hadoop
alaa-salima@alaa-salima-VirtualBox:~$ ls /usr/local/
bin  etc  games  hadoop  include  lib  libexec  man  sbin  share  src
alaa-salima@alaa-salima-VirtualBox:~$ ls -l /usr/local/
total 40
drwxr-xr-x 2 root root 4096 Oct  9 14:16 bin
drwxr-xr-x 2 root root 4096 Oct  9 14:16 etc
drwxr-xr-x 2 root root 4096 Oct  9 14:16 games
drwxr-xr-x 2 root root 4096 Nov 21 15:26 hadoop
drwxr-xr-x 2 root root 4096 Oct  9 14:16 include
drwxr-xr-x 3 root root 4096 Oct  9 14:16 lib
drwxr-xr-x 2 root root 4096 Oct  9 14:16 libexec
lrwxrwxrwx 1 root root    9 Oct  9 14:16 man -> share/man
drwxr-xr-x 2 root root 4096 Oct  9 14:16 sbin
drwxr-xr-x 7 root root 4096 Oct  9 14:19 share
drwxr-xr-x 2 root root 4096 Oct  9 14:16 src
alaa-salima@alaa-salima-VirtualBox:~$
```

- Ajouter les alias ci-dessous au fichier .bashrc :

```
  GNU nano 8.1                  /home/alaa-salima/.bashrc
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
# HADOOP VARIABLES END

^G Help      ^O Write Out ^F Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Pour que les alias puissent entrer en vigueur, il faut redémarrer le terminal ou taper la commande source .bashrc.

```
alaa-salima@alaa-salima-VirtualBox:~$ echo $JAVA_HOME
/usr/lib/jvm/java-21-openjdk-amd64
alaa-salima@alaa-salima-VirtualBox:~$ echo $HADOOP_INSTALL
/usr/local/hadoop/hadoop-3.3.6
alaa-salima@alaa-salima-VirtualBox:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/snap/bin:/snap/bin:/usr/local/hadoop/hadoop-3.3.6/bin:/usr/local/hadoo
p/hadoop-3.3.6/sbin
alaa-salima@alaa-salima-VirtualBox:~$ nano ~/.bashrc
alaa-salima@alaa-salima-VirtualBox:~$ source ~/.bashrc
alaa-salima@alaa-salima-VirtualBox:~$
```

- Modifier le fichier de démarrage de hadoop « hadoop-env.sh » en ajoutant le chemin du JAVA_HOME :

```
  GNU nano 8.1 /usr/local/hadoop/hadoop-3.3.6/etc/hadoop/hadoop-env.sh
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64

# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=
export HADOOP_HOME=/usr/local/hadoop/hadoop-3.3.6

# Location of Hadoop's configuration information.  i.e., where this
# file is living. If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
#
# NOTE: It is recommend that this variable not be set here but in
# /etc/profile.d or equivalent.  Some options (such as
# --config) may react strangely otherwise.

^G Help      ^O Write Out ^F Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Modifier le fichier core-site.xml :

```
  GNU nano 8.1 /usr/local/hadoop/hadoop-3.3.6/etc/hadoop/core-site.xml
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
</property>
</configuration>

                            [ Wrote 24 lines ]
^G Help      ^O Write Out ^F Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Modifier le fichier hdfs-site.xml :

```
  GNU nano 8.1 /usr/local/hadoop/hadoop-3.3.6/etc/hadoop/hdfs-site.xml *
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
 <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.name.dir</name>
        <value>/home/alaa-salima/hadoop/dfs/name</value>
    </property>
    <property>
        <name>dfs.data.dir</name>
        <value>/home/alaa-salima/hadoop/dfs/data</value>
    </property>
</configuration>


^G Help      ^O Write Out ^F Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Faire une copie du fichier mapred-site.xml.template sous le nom mapred-site.xml :

```
alaa-salima@alaa-salima-VirtualBox:~$ sudo cp $HADOOP_INSTALL/etc/hadoop/mapred-site.xml.template $HADOO
P_INSTALL/etc/hadoop/mapred-site.xml
cp: cannot stat '/usr/local/hadoop/hadoop-3.3.6/etc/hadoop/mapred-site.xml.template': No such file or di
rectory
alaa-salima@alaa-salima-VirtualBox:~$ ls /usr/local/hadoop/hadoop-3.3.6/etc/hadoop/
capacity-scheduler.xml            httpfs-env.sh               mapred-site.xml
configuration.xsl                 httpfs-log4j.properties     shellprofile.d
container-executor.cfg            httpfs-site.xml             ssl-client.xml.example
core-site.xml                     kms-acls.xml                ssl-server.xml.example
hadoop-env.cmd                    kms-env.sh                  user_ec_policies.xml.template
hadoop-env.sh                     kms-log4j.properties        workers
hadoop-metrics2.properties        kms-site.xml                yarn-env.cmd
hadoop-policy.xml                 log4j.properties            yarn-env.sh
hadoop-user-functions.sh.example  mapred-env.cmd              yarnservice-log4j.properties
hdfs-rbf-site.xml                 mapred-env.sh               yarn-site.xml
hdfs-site.xml                     mapred-queues.xml.template
alaa-salima@alaa-salima-VirtualBox:~$ nano /usr/local/hadoop/hadoop-3.3.6/etc/hadoop/mapred-site.xml
alaa-salima@alaa-salima-VirtualBox:~$ nano /usr/local/hadoop/hadoop-3.3.6/etc/hadoop/mapred-site.xml
alaa-salima@alaa-salima-VirtualBox:~$
```

- Formater le système hdfs avec : hdfs namenode –format :

```
alaa-salima@alaa-salima-VirtualBox:~$ hdfs namenode -format
2024-11-23 14:36:12,890 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = alaa-salima-VirtualBox/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.3.6
STARTUP_MSG:   classpath = /usr/local/hadoop/hadoop-3.3.6/etc/
op/hadoop-3.3.6/share/hadoop/common/lib/commons-configuration2
hadoop/hadoop-3.3.6/share/hadoop/common/lib/slf4j-reload4j-1.7
oop/hadoop-3.3.6/share/hadoop/common/lib/jetty-http-9.4.51.v20
hadoop/hadoop-3.3.6/share/hadoop/common/lib/commons-math3-3.1.
p/hadoop-3.3.6/share/hadoop/common/lib/zookeeper-3.6.3.jar:/us
3.3.6/share/hadoop/common/lib/commons-logging-1.1.3.jar:/usr/l
```

- Démarrer le système hadoop (hdfs aussi) avec start-all.sh :

```
alaa-salima@alaa-salima-VirtualBox:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as alaa-salima in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [alaa-salima-VirtualBox]
2024-11-25 11:43:01,441 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN_OPTS.
Starting resourcemanager
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN_OPTS.
Starting nodemanagers
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN_OPTS.
localhost: WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN_OPTS.
```

- Vérifier les services actifs après le démarrage :

```
alaa-salima@alaa-salima-VirtualBox:~$ jps
12736 DataNode
12577 NameNode
12978 SecondaryNameNode
13186 ResourceManager
13637 Jps
alaa-salima@alaa-salima-VirtualBox:~$
```

Maintenant, je vais présenter un exemple d'exécution d'un job avec MapReduce. Je vais utiliser un exemple très connu est celui du comptage du nombre d'occurrences de chaque mot dans un texte

- Créer un répertoire datainput dans votre dossier personnel avec mkdir ~/datainput. Ce dossier va servir comme un bassin ou on va mettre les fichiers à analyser :

```
laa-salima@alaa-salima-VirtualBox:~$ mkdir ~/datainput
laa-salima@alaa-salima-VirtualBox:~$ ls ~/datainput
```

- Créer un fichier texte.txt (à remplir avec plusieurs mots) dans ce dossier datainput avec cat > ~/datainput/texte.txt :

```
alaa-salima@alaa-salima-VirtualBox:~$ cat > ~/datainput/texte.txt
Bonjour Hadoop
Ceci est un exemple
alaa-salima@alaa-salima-VirtualBox:~$
```

- Transférer les fichiers à traiter avec hdfs dfs -put /home/<nom>/datainput /user/input :

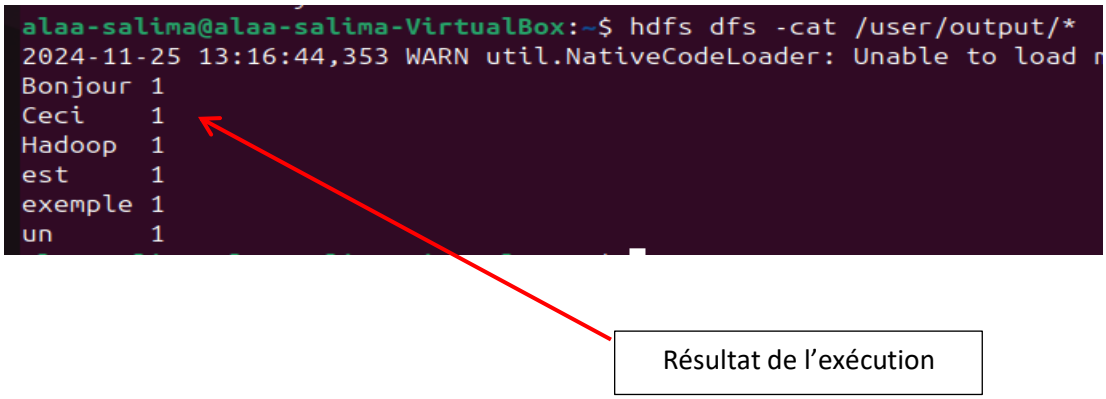
```
alaa-salima@alaa-salima-VirtualBox:~$ hdfs dfs -put ~/datainput /user/input
2024-11-25 12:28:13,265 WARN util.NativeCodeLoader: Unable to load native-hadoop
alaa-salima@alaa-salima-VirtualBox:~$ hdfs dfs -ls /user/input
2024-11-25 12:28:30,062 WARN util.NativeCodeLoader: Unable to load native-hadoop
Found 1 items
drwxr-xr-x   - alaa-salima supergroup          0 2024-11-25 12:28 /user/input/da
alaa-salima@alaa-salima-VirtualBox:~$ hdfs dfs -ls /user/input/datainput
2024-11-25 12:29:36,318 WARN util.NativeCodeLoader: Unable to load native-hadoop
Found 1 items
-rw-r--r--   3 alaa-salima supergroup         35 2024-11-25 12:28 /user/input/da
alaa-salima@alaa-salima-VirtualBox:~$ hdfs dfs -cat /user/input/datainput/texte.

2024-11-25 12:30:20,756 WARN util.NativeCodeLoader: Unable to load native-hadoop
Bonjour Hadoop
Ceci est un exemple
```

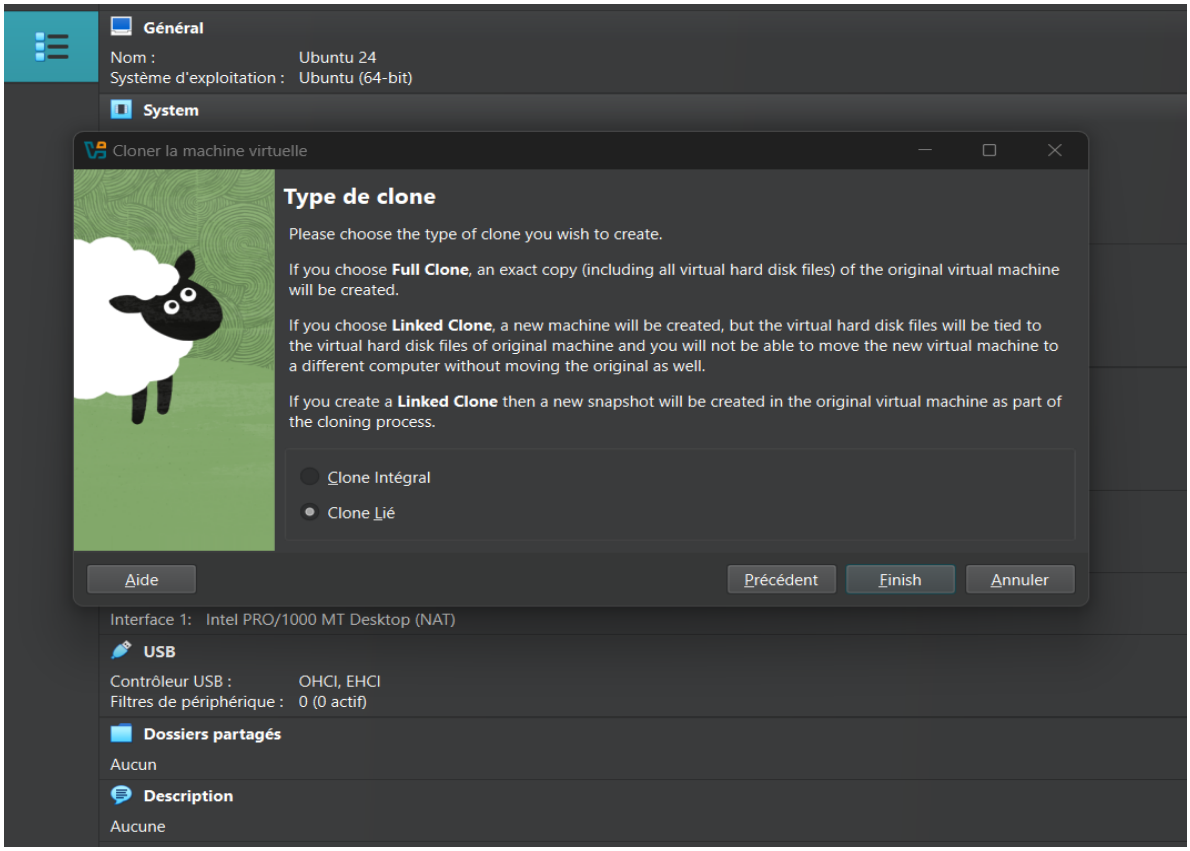
- Lancez un jar exemple (wordcount) avec hadoop jar /usr/local/hadoop/hadoop-…/share/hadoop/mapreduce/hadoop-mapreduce-examples-….jar wordcount /user/input /user/output :

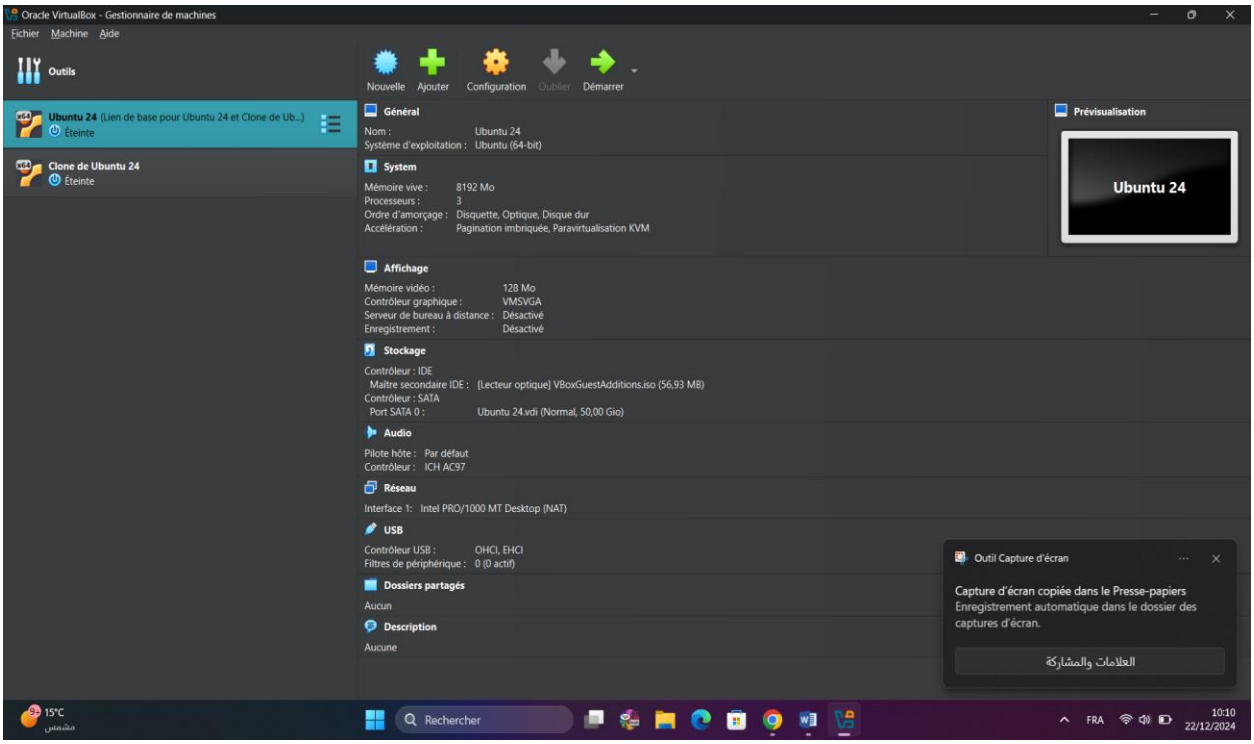
```
        at org.apache.hadoop.util.RunJar.run(RunJar.java:328)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:241)
alaa-salima@alaa-salima-VirtualBox:~$ hdfs dfs -rm -r /user/output
2024-11-25 13:12:58,695 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
Deleted /user/output
alaa-salima@alaa-salima-VirtualBox:~$ hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount /user/input/
2024-11-25 13:13:19,497 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
2024-11-25 13:13:20,254 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-11-25 13:13:20,368 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-11-25 13:13:20,369 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-11-25 13:13:20,937 INFO input.FileInputFormat: Total input files to process : 1
2024-11-25 13:13:21,236 INFO mapreduce.JobSubmitter: number of splits:1
2024-11-25 13:13:21,486 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local805325119_0001
2024-11-25 13:13:21,486 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-11-25 13:13:21,712 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-11-25 13:13:21,730 INFO mapreduce.Job: Running job: job_local805325119_0001
2024-11-25 13:13:21,746 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-11-25 13:13:21,782 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2024-11-25 13:13:21,784 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-11-25 13:13:21,784 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cl
2024-11-25 13:13:21,788 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2024-11-25 13:13:21,931 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-11-25 13:13:21,934 INFO mapred.LocalJobRunner: Starting task: attempt_local805325119_0001_m_000000_0
2024-11-25 13:13:22,014 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2024-11-25 13:13:22,020 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-11-25 13:13:22,040 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cl
2024-11-25 13:13:22,116 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2024-11-25 13:13:22,129 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/user/input/datainput/texte.txt:0+35
2024-11-25 13:13:22,355 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2024-11-25 13:13:22,356 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2024-11-25 13:13:22,356 INFO mapred.MapTask: soft limit at 83886080
2024-11-25 13:13:22,357 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2024-11-25 13:13:22,357 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2024-11-25 13:13:22,374 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2024-11-25 13:13:22,647 INFO mapred.LocalJobRunner:
2024-11-25 13:13:22,660 INFO mapred.MapTask: Starting flush of map output
```

- Récupérer le résultat avec hdfs dfs -cat /user/output/* :



Il est recommandé de cloner cette machine pour avoir une machine prête au cas où celle d'origine aurait des problèmes suite à l'installation de nouveaux logiciels.




Oracle VirtualBox - Gestionnaire de machines
Fichier Machine Aide
Outils
Ubuntu 24 (Lien de base pour Ubuntu 24 et Clone de Ub...)
Éteinte
Clone de Ubuntu 24
Éteinte
Nouvelle Ajouter Configuration Oublier Démarrer
Général
Nom : Ubuntu 24
Système d'exploitation : Ubuntu (64-bit)
System
Mémoire vive : 8192 Mo
Processeurs : 3
Ordre d'amorçage : Disquette, Optique, Disque dur
Accélération : Pagination imbriquée, Paravirtualisation KVM
Prévisualisation
Ubuntu 24
Affichage
Mémoire vidéo : 128 Mo
Contrôleur graphique : VMSVGA
Serveur de bureau à distance : Désactivé
Enregistrement : Désactivé
Stockage
Contrôleur : IDE
Maître secondaire IDE : [Lecteur optique] VBoxGuestAdditions.iso (56,93 MB)
Contrôleur : SATA
Port SATA 0 : Ubuntu 24.vdi (Normal, 50,00 Gio)
Audio
Pilote hôte : Par défaut
Contrôleur : ICH AC97
Réseau
Interface 1: Intel PRO/1000 MT Desktop (NAT)
USB
Contrôleur USB : OHCI, EHCI
Filtres de périphérique : 0 (0 actif)
Dossiers partagés
Aucun
Description
Aucune
Outil Capture d'écran
Capture d'écran copiée dans le Presse-papiers
Enregistrement automatique dans le dossier des captures d'écran.
العلامات والمشاركة
15°C
Rechercher
FRA
10:10
22/12/2024

Chapitre V : Apache SPARK

V.1 Introduction

Dans un monde de plus en plus numérique, la gestion et l'analyse de vastes volumes de données deviennent des enjeux cruciaux pour les entreprises. La quantité de données générées par les utilisateurs, les entreprises, ainsi que les objets connectés croît de manière exponentielle, posant ainsi des défis majeurs en termes de traitement rapide et efficace.

Les systèmes de gestion de bases de données relationnelles traditionnels peinent à répondre à ces besoins, principalement en raison de la volumétrie, de la vitesse et de la diversité des données. C'est dans ce contexte que les solutions Big Data ont émergé, permettant de traiter et d'analyser d'énormes quantités de données de manière distribuée et parallèle.

Ce chapitre se propose d'explorer Apache Spark, l'un des Framework les plus performants de l'écosystème Big Data. Nous nous concentrerons sur la compréhension de son architecture, sa configuration, ses principales fonctionnalités et ses cas d'usage dans des applications réelles. L'objectif est de mettre en lumière les avantages de Spark en comparaison avec des technologies concurrentes comme Hadoop, tout en explorant ses applications pratiques dans le monde de l'industrie.

Je rappelle que le Big Data inclut un ensemble de technologies conçues pour traiter et analyser d'immenses volumes de données de manière distribuée. Parmi ces technologies, Hadoop se distingue comme l'un des Framework open-source les plus utilisés. Il repose sur des systèmes de fichiers distribués (HDFS) et utilise le modèle de programmation MapReduce pour traiter les données de manière séquentielle. Cependant, bien que robuste, Hadoop présente certaines limitations en termes de performance, notamment en raison des écritures et lectures sur disque entre chaque étape du traitement.

Dans ce contexte, Apache Spark se positionne comme une alternative plus rapide et flexible. En traitant les données directement en mémoire (RAM), Spark réduit

considérablement les délais de traitement par rapport à Hadoop, qui repose principalement sur le disque dur pour la gestion des données intermédiaires.

V.2 Pourquoi Spark ?

Apache Spark se distingue par sa capacité à effectuer des traitements rapides en mémoire, ce qui améliore nettement les performances par rapport à Hadoop, notamment dans le cadre des analyses interactives ou des traitements en temps réel.

Il prend en charge plusieurs types de traitements, dont le traitement par lots, le traitement en temps réel (avec Spark Streaming), l'apprentissage automatique (via MLlib), et les requêtes SQL (avec Spark SQL), offrant ainsi une grande polyvalence pour divers cas d'usage dans l'écosystème Big Data.

En outre, Spark est compatible avec des outils populaires comme Hadoop, Kafka, et Hive, et il peut être déployé dans des environnements cloud. Cela en fait une solution de choix pour les entreprises souhaitant traiter efficacement de grands volumes de données. Il est devenu un outil incontournable pour les data engineers, data scientists et développeurs travaillant avec des ensembles de données massives.

L'objectif de ce projet est de comprendre comment Spark s'intègre dans l'écosystème Big Data, d'explorer ses forces et ses limites, et de démontrer comment il peut résoudre des problématiques concrètes de gestion et d'analyse de données.

Apache Spark a été développé en 2009 par Matei Zaharia à l'Université de Californie, Berkeley, dans le but d'améliorer les performances de Hadoop MapReduce en utilisant la mémoire vive (RAM) pour le traitement des données. Après sa présentation à la communauté académique en 2010 et son incubation par la fondation Apache en 2012, Spark est devenu un projet Apache Top-Level en 2014.

Depuis, Spark a évolué pour inclure des fonctionnalités comme Spark SQL, MLlib pour l'apprentissage automatique et GraphX pour le traitement de graphes. Aujourd'hui, Spark est largement adopté par des entreprises comme Uber et Netflix pour le traitement de données massives.

V.3 Architecture de SPARK

L'architecture de Spark repose sur une approche distribuée. Elle se compose de plusieurs composants clés, permettant un traitement parallèle et résilient des données à grande échelle (figure V.1) :

- **Driver** : C'est le programme principal qui coordonne l'exécution des applications Spark. Il gère le **SparkContext**, qui permet d'interagir avec le cluster.
- **Cluster Manager** : Ce composant gère les ressources du cluster et attribue des ressources aux applications Spark. Il peut être Standalone, YARN, Mesos, ou Kubernetes.
- **Worker Nodes** : Ce sont les nœuds qui exécutent les tâches. Chaque nœud contient plusieurs **Executors**.
- **Executors** : Ce sont des processus qui effectuent les calculs et stockent les données en mémoire pour un accès rapide.
- **RDD (Resilient Distributed Dataset)** : C'est la structure de données fondamentale de Spark, permettant de stocker et traiter les données de manière distribuée et résiliente.
- **DAG (Directed Acyclic Graph)** : Il représente l'ensemble des transformations effectuées sur les RDD, permettant une planification optimale des tâches.

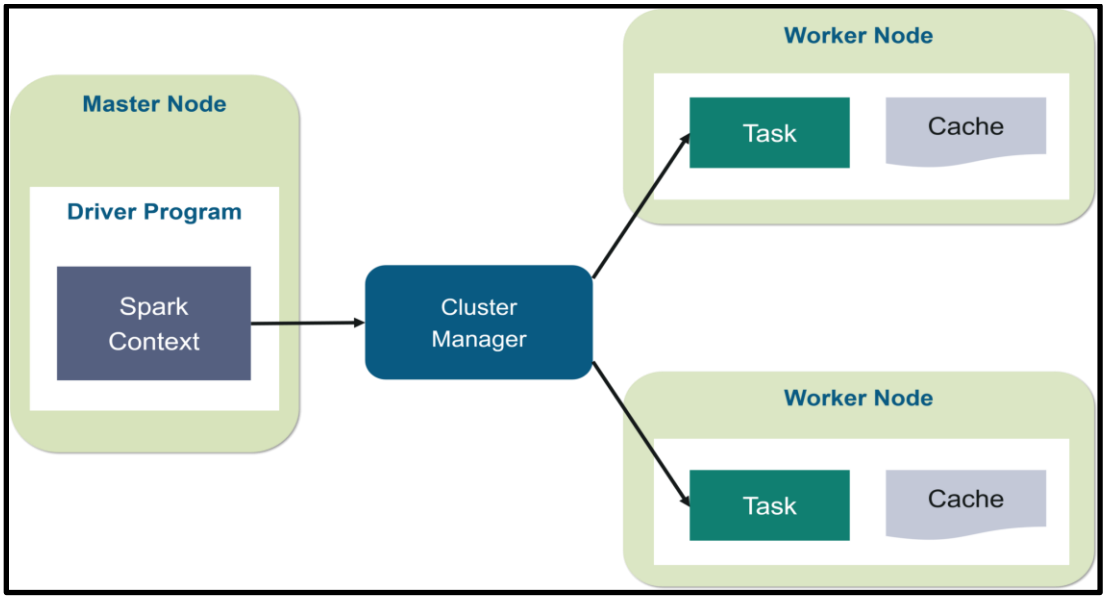


Figure V.1 : Architecture de Spark

V.4 MapReduce vs Apache Spark

Actuellement, SPARK est le concurrent de MapReduce comme il offre des possibilités d'amélioration du traitement MapReduce comme le traitement parallèle et le temps réduis d'exécution.

Le tableau suivant présente une comparaison entre SPARK et MapReduce selon plusieurs critères :

Critère	MapReduce	Apache Spark
Vitesse	Moins rapide, nécessite des lectures/écritures disque.	Plus rapide grâce au traitement en mémoire.
Modèle de Programmation	Basé sur "Map" et "Reduce", complexe à gérer.	API simple avec abstractions comme RDD et DataFrame.
Tolérance aux pannes	Répétition des tâches échouées à partir des disques.	Tolérance aux pannes grâce à la réplication des données en mémoire.
Traitement en temps réel	Non adapté.	Support du traitement en temps réel (Spark Streaming).
Facilité d'utilisation	Programmation complexe.	Interface intuitive avec des API modernes.
Scalabilité	Très scalable mais dépend des clusters Hadoop.	Très scalable, fonctionne avec Hadoop ou en mode stand-alone.
Flexibilité	Moins flexible pour des traitements complexes.	Très flexible, adapté pour le machine Learning, SQL, et plus.

Table V.1 : Comparaison MapReduce et Apache Spark

V.5 Utilisation de Spark dans l'industrie

Apache Spark est largement utilisé dans de nombreux secteurs industriels pour le traitement de données massives en temps réel et pour les analyses complexes.

Voici quelques exemples d'applications concrètes de Spark dans des secteurs comme la finance, et bien d'autres.

- **Finance :** Dans le secteur financier, Spark est utilisé pour traiter et analyser de grandes quantités de données transactionnelles en temps réel, permettant ainsi de détecter des fraudes, de réaliser des analyses prédictives et d'améliorer la gestion des risques.

- **Détection de fraudes :** Grâce à la capacité de Spark à traiter des flux de données en temps réel avec Spark Streaming, les institutions financières peuvent analyser les transactions instantanément. Par exemple, lors de chaque transaction, Spark peut appliquer des modèles de machine Learning (via MLlib) pour détecter des comportements suspects et alerter les analystes ou automatiquement bloquer les transactions potentiellement frauduleuses.

- **Secteur du retaille (Commerce de détail) :** Dans le secteur du commerce de détail, Spark est souvent utilisé pour personnaliser l'expérience client, recommander des produits et optimiser les processus logistiques.

- **Systèmes de recommandation :** Spark est utilisé pour développer des systèmes de recommandation qui analysent les comportements d'achat des consommateurs afin de proposer des produits adaptés.

  Par exemple, des entreprises comme Amazon utilisent des algorithmes de filtrage collaboratif et de filtrage basé sur le contenu, qui sont déployés efficacement sur Spark grâce à son traitement rapide en mémoire et à sa capacité à traiter de grands ensembles de données (ex. : historique des achats, produits similaires).

V.6 Intégration de Spark avec d'autres outils Big Data

L'une des forces d'Apache Spark est sa capacité à s'intégrer de manière fluide avec d'autres outils et technologies de l'écosystème Big Data, ce qui permet de tirer parti des points forts de chaque outil pour créer des solutions complètes.

- **Intégration avec Kafka (Gestion des flux de données en temps réel) :**

Apache Kafka est un système de messagerie distribué qui permet de traiter des flux de données en temps réel. Spark peut être utilisé en conjonction avec Kafka pour traiter ces flux de données de manière distribuée.

Kafka est souvent utilisé pour collecter des données provenant de diverses sources en temps réel (par exemple, logs d'application, événements utilisateur, etc.). Ces données peuvent ensuite être envoyées à Spark Streaming pour un traitement en temps réel, comme l'analyse de ces événements pour détecter des anomalies ou des comportements suspects.

Par exemple, un site de e-commerce pourrait utiliser Kafka pour récupérer les actions des utilisateurs en temps réel et les transmettre à Spark pour analyser les tendances de consommation et adapter les offres instantanément.

Cette intégration permet de traiter des données volumineuses et des événements en temps réel de manière très efficace. Kafka gère la collecte et la transmission des données, tandis que Spark assure le traitement et l'analyse rapide de ces données à l'échelle.

- **Intégration avec Hadoop (Stockage et gestion des données)**

Spark et Hadoop sont souvent utilisés ensemble dans de nombreuses entreprises, où Hadoop est utilisé pour le stockage et la gestion de données, tandis que Spark est utilisé pour les calculs distribués rapides.

Hadoop, via son système de fichiers distribué HDFS (Hadoop Distributed File System), est utilisé pour stocker d'énormes volumes de données non structurées provenant de diverses sources (logs, images, vidéos, etc.).

Spark peut alors accéder à ces données stockées dans HDFS pour les traiter rapidement en mémoire, grâce à son architecture optimisée pour les calculs en mémoire.

Par exemple, une entreprise qui possède une grande quantité de données historiques sur ses clients pourrait utiliser Hadoop pour stocker ces données et Spark pour effectuer une analyse en temps réel de l'activité des clients et améliorer ses stratégies marketing.

Cette combinaison permet de stocker et de gérer de grands ensembles de données de manière économique tout en offrant des performances de traitement beaucoup plus rapides que celles offertes par Hadoop seul, grâce à Spark.

V.7 Limitations et défis du déploiement de Spark à grande échelle

Bien qu'Apache Spark soit une solution très puissante et flexible, son déploiement à grande échelle peut entraîner certains défis et limitations.

- **Scalabilité et gestion des ressources**

L'un des principaux défis dans le déploiement de Spark à grande échelle est la gestion des ressources, notamment en termes de mémoire. Spark est un moteur en mémoire, ce qui signifie qu'il nécessite une quantité significative de mémoire pour traiter de grands volumes de données.

Si la mémoire n'est pas suffisante, cela peut entraîner des erreurs ou un ralentissement des performances. La scalabilité des clusters Spark doit être soigneusement gérée pour éviter des goulets d'étranglement, en particulier lorsque le nombre de nœuds augmente.

Il est possible de gérer ces défis en optimisant la configuration des ressources (par exemple, la gestion des partitions de données) et en utilisant des systèmes de gestion de clusters comme Kubernetes ou YARN pour répartir les tâches de manière plus équilibrée.

- **Complexité de la gestion de la configuration et du déploiement**

Le déploiement de Spark dans de grandes infrastructures peut être complexe. La configuration des clusters, la gestion des dépendances logicielles, et la surveillance des performances sont des tâches nécessitant une expertise technique avancée.

De plus, le processus de déploiement d'applications Spark sur des clusters distribués peut entraîner des erreurs difficiles à diagnostiquer.

L'utilisation de systèmes d'orchestration comme Kubernetes peut simplifier la gestion des ressources et la mise à l'échelle dynamique. Par ailleurs, des outils comme Apache Mesos ou YARN peuvent être utilisés pour gérer les ressources et assurer la gestion des tâches dans un environnement de traitement distribué.

- **Coût et gestion de la mémoire**

Le traitement en mémoire implique que Spark utilise une grande quantité de RAM, ce qui peut rendre le déploiement coûteux en termes d'infrastructure, surtout lorsqu'on travaille avec de très grands ensembles de données.

Si les données dépassent la capacité de la mémoire, Spark peut rencontrer des problèmes de performance, voire échoué.

Pour éviter ces problèmes, il est possible de recourir à des techniques comme la persistance des RDDs (Resilient Distributed Datasets) qui permettent de stocker les données sur disque lorsque la mémoire est saturée, ou d'optimiser les algorithmes pour réduire la quantité de mémoire nécessaire.

V.8 Conclusion

MapReduce reste une technologie robuste pour les traitements par lots et les tâches nécessitant une tolérance élevée aux pannes. Cependant, avec l'essor des données massives et la nécessité de traitements plus rapides et interactifs, Apache Spark a émergé comme une solution plus performante et flexible.

Grâce à son traitement en mémoire, sa prise en charge de divers types de traitements, et sa facilité d'intégration avec d'autres outils Big Data, Spark est aujourd'hui une référence dans le domaine du traitement des données massives.

TP : Installer et configurer Apache Spark sur Windows

Prérequis

- Un système exécutant Windows 10 ou 11.
- Un compte utilisateur avec des privilèges d'administrateur.
- Accès à Windows Invite de commande ou PowerShell.
- Un outil pour extraire les fichiers .tar, tels que 7-Zip ou WinRAR.

Pour configurer Apache Spark, vous devez installer Java, téléchargez le package Spark et configurez les variables d'environnement. Python est également requis pour utiliser l'API Python de Spark appelée PySpark.

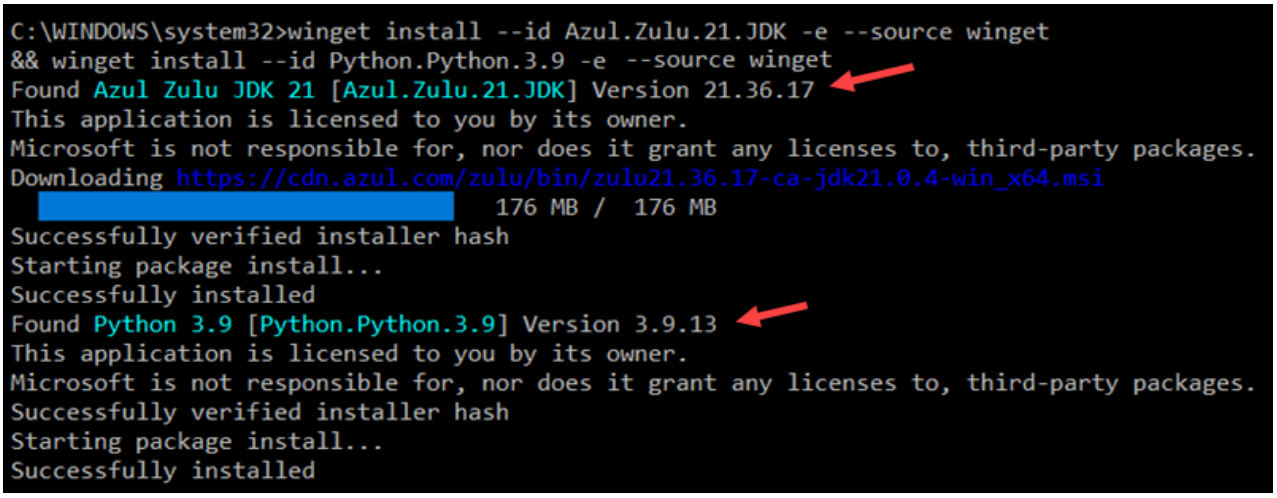
Si vous possédez déjà Java 8 (ou version ultérieure) et Python 3 (ou version ultérieure) installée, vous pouvez ignorer la première étape de ce guide.

Étape 1 : Installer les dépendances Spark

Utilisation de Windows aileron L'utilitaire est un moyen pratique d'installer les dépendances nécessaires pour Apache Spark :

- Ouvrir Invite de commandes ou PowerShell en tant qu'administrateur.
- Entrez la commande suivante pour installer le OpenJDK 21 bleu zoulou (Kit de développement Java) et Python 3.9 :

winget install --id Azul.Zulu.21.JDK -e --source winget && winget install --id Python.Python.3.9 -e --source winget



Ajustez la commande si vous avez besoin d'un Java ou Version Python ou un autre fournisseur OpenJDK.

L'installation d'OpenJDK se trouve dans C:\Program Files\Zulu\zulu-21 dossier par défaut. L'espace dans le chemin peut entraîner des problèmes lors du lancement d'Apache Spark. Évitez cela en déplaçant l'installation vers un dossier sans espaces. Utilisez la commande suivante pour créer un nouveau dossier Zulu dans le répertoire racine et déplacer l'installation :

mkdir C:\Zulu && robocopy "C:\Program Files\Zulu\zulu-21" "C:\Zulu\zulu-21" /E /MOVE

```
               Total    Copié   IgnoréDiscordance    ÉCHEC    Extras
    Rép :        118      118         0         0         0         0
Fichiers :       577      577         0         0         0         0
  Octets :   333.88 m  333.88 m       0         0         0         0
   Heures:   0:00:02   0:00:01                       0:00:00   0:00:01


   Débit :           282 117 697 Octets/sec.
   Débit :            16142,904 Méga-octets/min.
   Fin : mardi 10 décembre 2024 09:27:38
```

Vérifiez l'installation en vérifiant les versions Java et Python :

java -version && python --version

```
C:\WINDOWS\system32>java -version && python --version
openjdk version "21.0.4" 2024-07-16 LTS
OpenJDK Runtime Environment Zulu21.36+17-CA (build 21.0.4+7-LTS)
OpenJDK 64-Bit Server VM Zulu21.36+17-CA (build 21.0.4+7-LTS, mixed mode, sharing)
Python 3.9.13
```

La sortie confirme que votre système utilise OpenJDK 21 et Python 3.9.

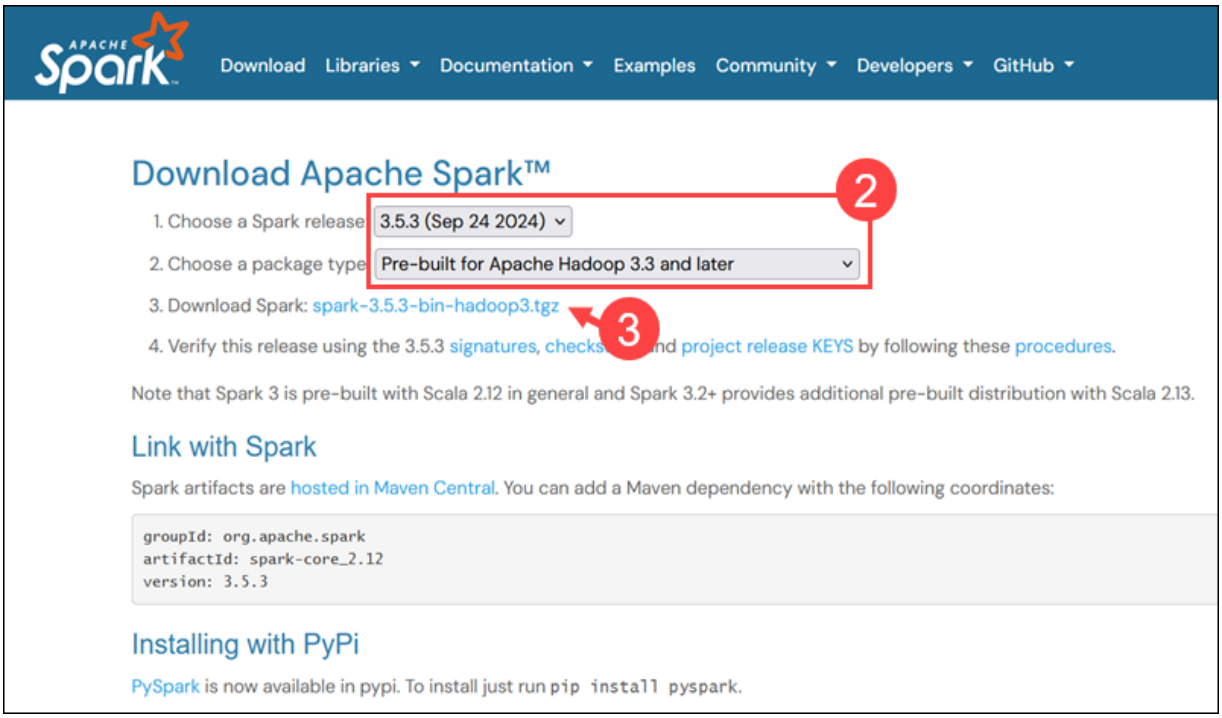
Étape 2 : Télécharger Apache Spark

Pour télécharger la dernière version d'Apache Spark :

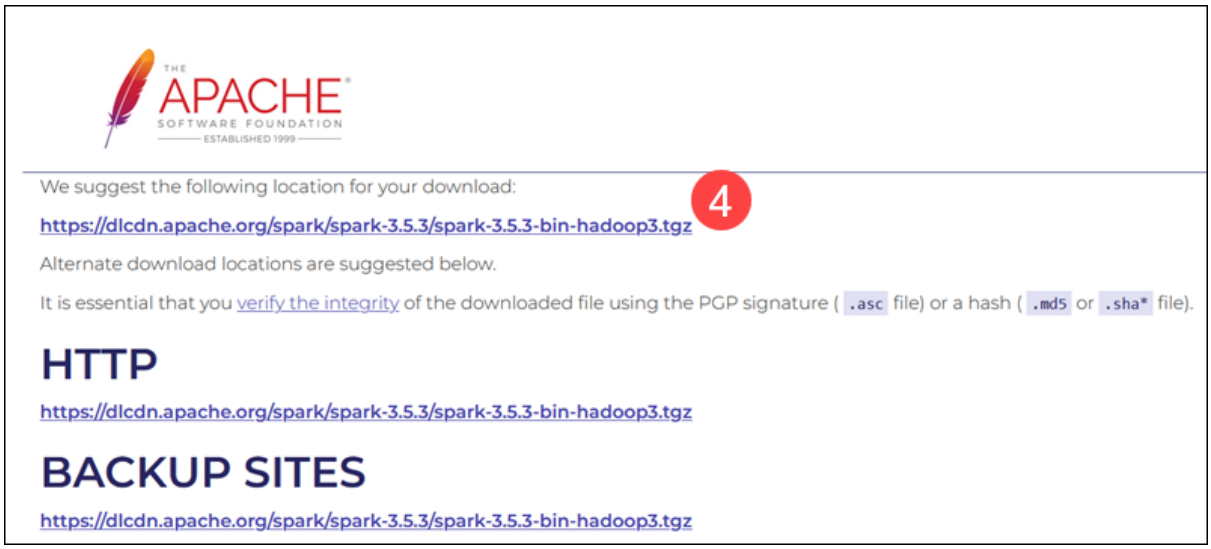
1. Ouvrez un navigateur et accédez au site officiel Page de téléchargement d'Apache Spark.

2. La dernière version de Spark est sélectionnée par défaut. Au moment de la rédaction de cet article, la dernière version est Spark 3.5.3 pour Hadoop 3.3.

3. Cliquez sur le lien de téléchargement spark-3.5.3-bin-hadoop3.tgz.



4. Sélectionnez un emplacement dans une liste de serveurs miroirs pour commencer le téléchargement.

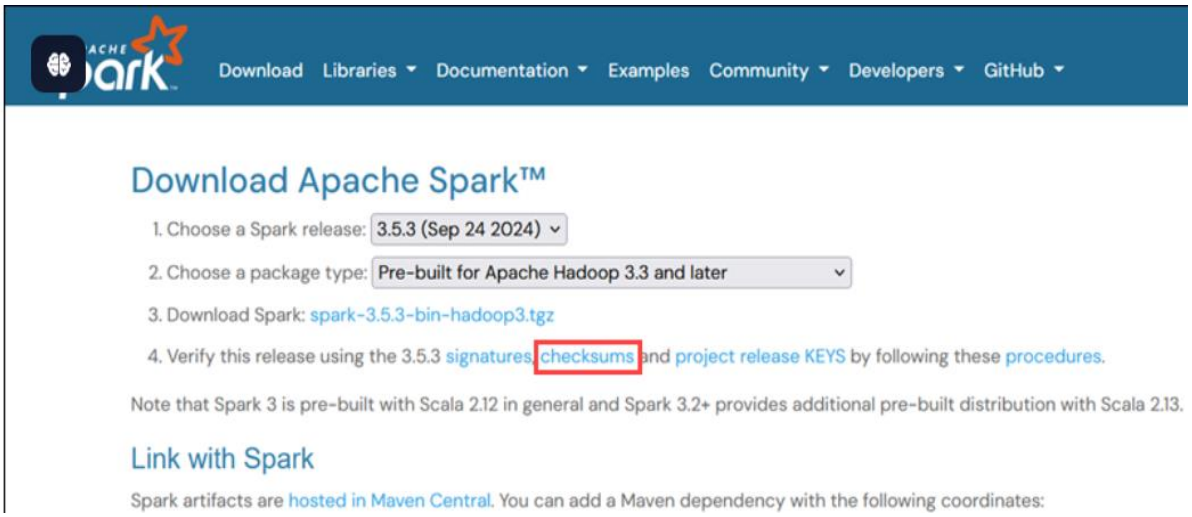


5. Enregistrez le fichier, par exemple, dans le dossier Téléchargements.

<u>Étape 3</u> : Vérifier le fichier logiciel Spark

Pour vérifier que vous travaillez avec un fichier non modifié et non corrompu, vérifiez son somme de contrôle :

1. Accédez à la page de téléchargement de Spark et ouvrez le lien des sommes de contrôle, de préférence dans un nouvel onglet.



2. Ouvrez l'invite de commande et utilisez le CD pour accéder au dossier dans lequel vous avez téléchargé Apache Spark. Par exemple, si le fichier se trouve dans le dossier Téléchargements, saisissez :

cd C:\Utilisateurs\nom d'utilisateur\Téléchargements

Remplacer nom d'utilisateur avec votre nom d'utilisateur Windows actuel.

3. Utilisez la commande suivante pour calculer la somme de contrôle du fichier téléchargé :

certutil -hashfile spark-3.5.3-bin-hadoop3.tgz SHA512

```
cd C:\Users\username\Downloads
```

Le système affiche un long code alphanumérique suivi du message Certutil :
-hashfile terminé avec succès.

4. Comparez manuellement la sortie de la somme de contrôle avec celle du site Web Apache Spark. S'ils correspondent, le fichier est légitime.

<u>Étape 4</u> : Installer Apache Spark

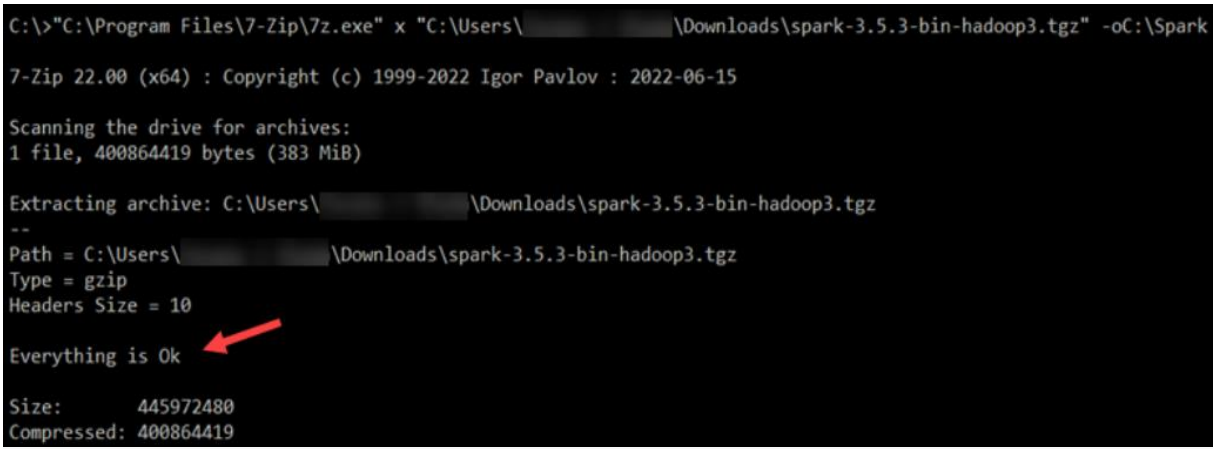
Pour installer Apache Spark, extrayez le fichier téléchargé à l'emplacement souhaité :

1. Par exemple, créez un nouveau dossier Spark à la racine du lecteur C: à l'aide de la commande suivante :

`cd \ && mkdir Spark`

2. Utilisez 7-Zip pour extraire le fichier Spark .tar du fichier compressé .tgz et placez-le dans le dossier Spark :

`"C:\Program Files\7-Zip\7z.exe" x "C:\Users\nom d'utilisateur\Téléchargements\spark-3.5.3-bin-hadoop3.tgz" -oC:\Spark`



Remplacer `nom d'utilisateur` dans le chemin du fichier avec votre nom d'utilisateur.

> Remarque : les chemins d'accès aux fichiers sont entourés de guillemets doubles, car le nom d'utilisateur et les noms de dossier contiennent des espaces.

3. Extraire Spark binaires à partir du fichier Spark .tar :

`"C:\Program Files\7-Zip\7z.exe" x "C:\Spark\spark-3.5.3-bin-hadoop3.tar" -oC:\Spark`

```
C:\>"C:\Program Files\7-Zip\7z.exe" x "C:\Spark\spark-3.5.3-bin-hadoop3.tar" -oC:\Spark

7-Zip 22.00 (x64) : Copyright (c) 1999-2022 Igor Pavlov : 2022-06-15

Scanning the drive for archives:
1 file, 445972480 bytes (426 MiB)

Extracting archive: C:\Spark\spark-3.5.3-bin-hadoop3.tar
--
Path = C:\Spark\spark-3.5.3-bin-hadoop3.tar
Type = tar
Physical Size = 445972480
Headers Size = 1328128
Code Page = UTF-8
Characteristics = GNU LongName ASCII

Everything is Ok

Folders: 196
Files: 1825
Size:       444177357
Compressed: 445972480
```

4. Utilisez le dir commande pour lister le contenu du dossier Spark :

```
cd C:\Spark && dir
```

```
C:\>cd C:\Spark && dir
 Volume in drive C has no label.
 Volume Serial Number is 323E-E686

 Directory of C:\Spark

10/08/2024  04:11 PM    <DIR>          .
10/08/2024  04:11 PM    <DIR>          ..
09/09/2024  07:33 AM    <DIR>          spark-3.5.3-bin-hadoop3
10/08/2024  02:24 PM       445,972,480 spark-3.5.3-bin-hadoop3.tar
               1 File(s)    445,972,480 bytes
               3 Dir(s)  31,055,310,848 bytes free
```

Le spark-3.5.3-bin-hadoop3 le dossier contient les fichiers nécessaires pour exécuter Spark.

Étape 5 : Ajouter le fichier winutils.exe

L'utilitaire winutils permet à Apache Spark et à d'autres Basé sur Hadoop outils à exécuter sous Windows. Vous devez télécharger le fichier winutils.exe qui correspond à la version Hadoop utilisée par votre installation Spark :

1. Créez un dossier hadoop\bin sur le lecteur C: pour stocker le fichier winutils.exe :

`cd \ &&` 2. Utilisez le `curl` commande pour télécharger le fichier depuis le Dépôt GitHub winutils dans le dossier nouvellement créé :

`curl --ssl-no-revoke -L -o C:\hadoop\bin\winutils.exe https://github.com/cdarlint/winutils/raw/master/hadoop-3.3.5/bin/winutils.exe`

```
C:\>curl --ssl-no-revoke -L -o C:\hadoop\bin\winutils.exe
https://github.com/cdarlint/winutils/raw/master/hadoop-3.3.5/bin/winutils.exe
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  110k  100  110k    0     0  90063      0  0:00:01  0:00:01 --:--:--  156k
```

Dans cet exemple, la version Hadoop est 3.3.5. Si nécessaire, modifiez la commande pour qu'elle corresponde à la version Hadoop utilisée par votre installation Spark.

3. Vérifiez que le fichier a été téléchargé :

`cd C:\hadoop\bin && dir`

```
C:\WINDOWS\system32>cd C:\hadoop\bin && dir
 Volume in drive C has no label.
 Volume Serial Number is 323E-E686

 Directory of C:\hadoop\bin

10/08/2024  03:44 PM    <DIR>          .
10/08/2024  03:44 PM    <DIR>          ..
10/08/2024  03:44 PM           112,640 winutils.exe
               1 File(s)        112,640 bytes
               2 Dir(s)  30,871,511,040 bytes free
```

Le fichier winutils.exe est répertorié dans le dossier.

Étape 6 : Configurer les variables d'environnement

Ajoutez les emplacements Spark, Java et Hadoop à votre système `Chem cd C:\hadoop\bin in` variable d'environnement pour exécuter le shell Spark directement depuis la CLI.

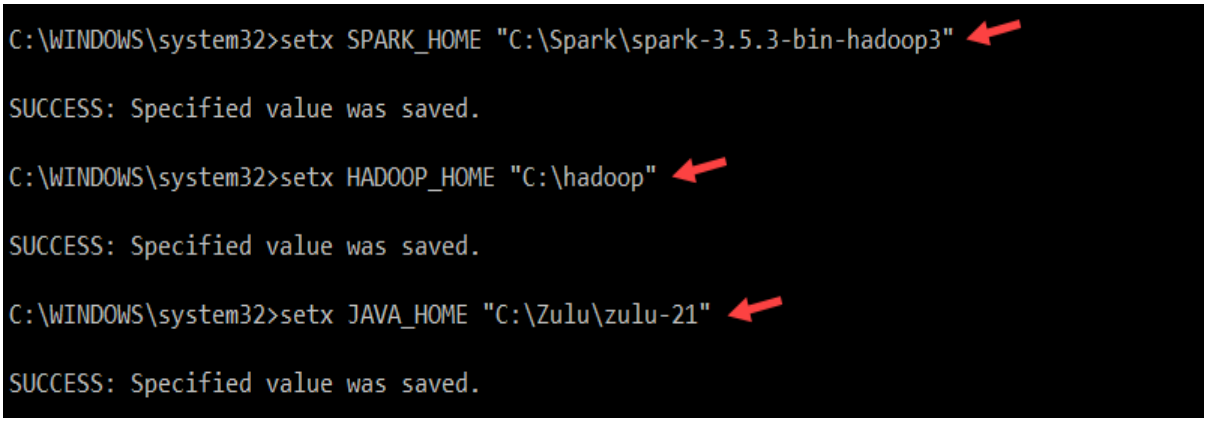
On peut modifier les variables d'environnement manuellement à l'aide de l'interface utilisateur Windows. Cependant, la définition des variables utilisateur à partir de l'invite de commande peut être plus rapide et plus efficace.

Utilisez le ensemblex commande pour définir le SPARK_ACCUEIL Variable utilisateur :

setx SPARK_HOME "C:\Spark\spark-3.5.5-bin-hadoop3"

Pour régler HADOOP_ACCUEIL, entrer: setx HADOOP_HOME "C:\hadoop"

Pour régler JAVA_ACCUEIL, taper: setx JAVA_HOME "C:\Zulu\zulu-21"



Les chemins d'accès dans les commandes sont définis en fonction des étapes d'installation précédentes. Si vous avez utilisé des versions et des chemins d'accès de fichiers différents, ajustez les commandes en conséquence.

Entrez la commande suivante pour récupérer le chemin actuel du registre et le stocker dans le OldPath variable:

for /f "tokens=2*" %A dans ('reg query "HKCU\Environment" /v Path') doSet "oldPath=%B"

Ajoutez les chemins Spark, Hadoop et Java au Path variable sans écraser les entrées existantes :

setx path "%oldPath%;%SPARK_HOME%\bin;%HADOOP_HOME%\bin;%JAVA_HOME%\bin"

```
C:\WINDOWS\system32>for /f "tokens=2*" %A in ('reg query "HKCU\Environment" /v Path') do set "oldPath=%B"

C:\WINDOWS\system32>set "oldPath=C:\Program Files\Zulu\zulu-21\bin\;C:\Program Files\Common Files\Oracle\Ja
:\Program Files (x86)\Common Files\Oracle\Java\javapath;c:\Program Files (x86)\NVIDIA Corporation\PhysX\Com
ot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;%SYSTEMR
\OpenSSH\;C:\Program Files (x86)\Calibre2\;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\I
Common\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Python27;C:\Cassandra\apache-cassandra-3.11.6
\php;C:\ProgramData\ComposerSetup\bin;%NVM_HOME%;%NVM_SYMLINK%;C:\Program Files\nodejs\;C:\Program Files\Po
in;C:\Program Files (x86)\Bitvise SSH Client;C:\Program Files\PuTTY\;C:\Program Files (x86)\NetSarang\Xshel
\[redacted]\AppData\Local\Programs\Python\Python39;C:\Users\[redacted]\AppData\Local\Programs\Pytho
ripts;"

C:\WINDOWS\system32>setx Path "%oldPath%;%SPARK_HOME%\bin;%HADOOP_HOME%\bin;%JAVA_HOME%\bin"

SUCCESS: Specified value was saved.
```

Redémarrez l'invite de commande pour appliquer les modifications. Dans de rares cas, un redémarrage complet du système peut être nécessaire.

<u>Étape 7</u> : Lancer Spark

Pour démarrer le shell Spark Scala, saisissez ce qui suit : `spark-shell`

Ceci ouvre une interface shell interactive avec des notifications et des informations Spark.

La sortie inclut des détails sur la version Spark, la configuration et les ressources disponibles.

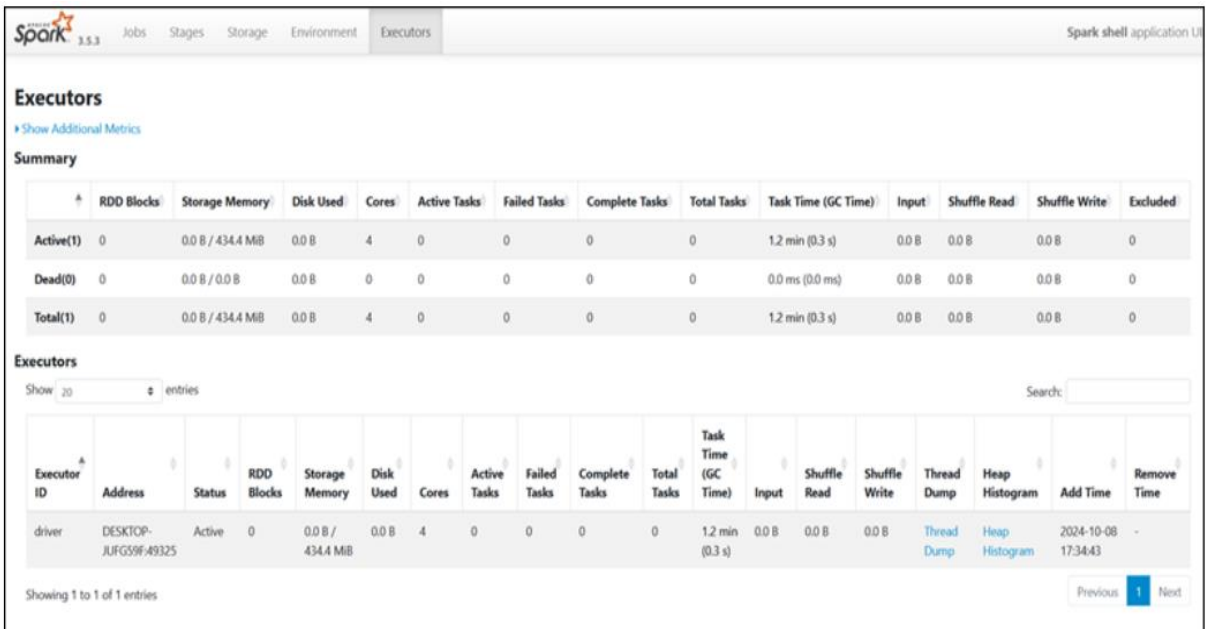
```
C:\WINDOWS\system32>spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/08 17:29:27 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform
ble
Spark context Web UI available at http://DESKTOP-JUFG59F:4040
Spark context available as 'sc' (master = local[*], app id = local-1728401369663).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.3
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 21.0.4)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Pour accéder à l'interface utilisateur Web de Spark, ouvrez un navigateur Web et accédez à l'URL suivante : <u>http://localhost:4040/</u>

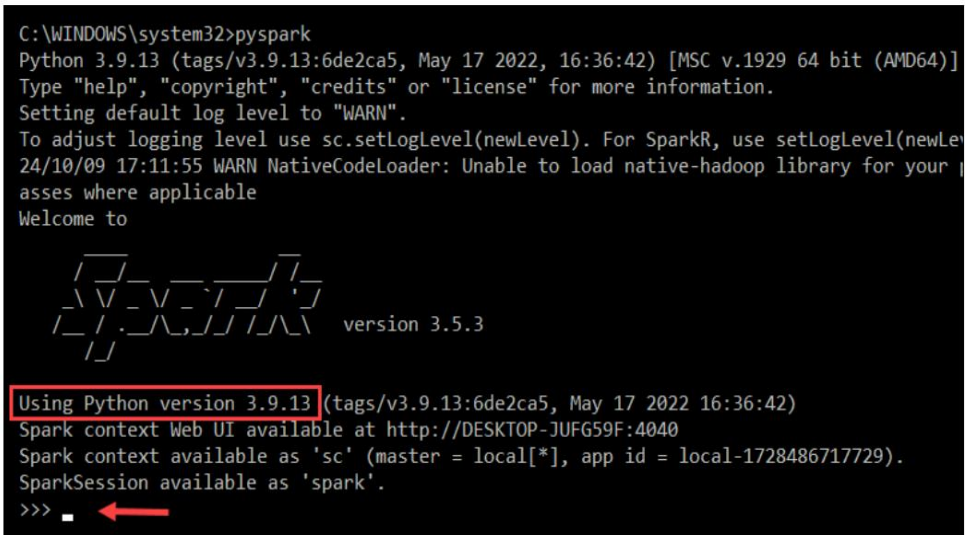
Vous pouvez remplacer `hôte local` avec le nom ou l'adresse IP de votre système. L'interface utilisateur Web Spark fournit des détails sur l'exécution des tâches, les étapes et l'utilisation des ressources.



Utilisez la commande suivante pour quitter Spark et fermer le shell Scala : `:quitter`

Vous pouvez également appuyer sur CTRL+D tout en utilisant l'invite de commande. Saisissez la commande suivante pour exécuter Spark à l'aide de Python : `pyspark`

Ceci lance le shell Spark avec une interface Python.



Pour quitter pyspark, tapez : `quitter()`

Quelques commandes usuelles sous Python et Scala

Action	Scala	Python (PySpark)
Initialisation de SparkSession	`scala   val spark = SparkSession.builder().appName("MonApplicationSpark").getOrCreate()`	`python   spark = SparkSession.builder.appName("MonApplicationSpark").getOrCreate()`
Lecture d'un fichier CSV	`scala   val df = spark.read.option("header", "true").csv("chemin/vers/fichier.csv")`	`python   df = spark.read.option("header", "true").csv("chemin/vers/fichier.csv")`
Lecture d'un fichier JSON	`scala   val jsonDf = spark.read.json("chemin/vers/fichier.json")`	`python   json_df = spark.read.json("chemin/vers/fichier.json")`
Afficher les premières lignes	`scala   df.show(5)`	`python   df.show(5)`
Sélectionner des colonnes	`scala   val selectedDf = df.select("colonne1", "colonne2")`	`python   selected_df = df.select("colonne1", "colonne2")`
Filtrer des données	`scala   val filteredDf = df.filter(df("colonne1") > 50)`	`python   filtered_df = df.filter(df["colonne1"] > 50)`
Ajouter une colonne calculée	`scala   val newDf = df.withColumn("nouvelle_colonne", df("colonne1") * 2)`	`python   new_df = df.withColumn("nouvelle_colonne", df["colonne1"] * 2)`
Calculer une agrégation	`scala   val avgDf = df.groupBy("colonne1").avg("colonne2")`	`python   avg_df = df.groupBy("colonne1").avg("colonne2")`
Agrégation multiple	`scala   val aggDf = df.groupBy("colonne1").agg(avg("colonne2"), count("colonne3"))`	`python   agg_df = df.groupBy("colonne1").agg({"colonne2": "avg", "colonne3": "count"})`
Écrire un DataFrame au format CSV	`scala   df.write.option("header", "true").csv("chemin/vers/sortie.csv")`	`python   df.write.option("header", "true").csv("chemin/vers/sortie.csv")`
Écrire un DataFrame au format Parquet	`scala   df.write.parquet("chemin/vers/sortie.parquet")`	`python   df.write.parquet("chemin/vers/sortie.parquet")`
Créer un RDD	`scala   val rdd = spark.sparkContext.parallelize(List(1, 2, 3, 4, 5))`	`python   rdd = spark.sparkContext.parallelize([1, 2, 3, 4, 5])`
Opération sur un RDD (map)	`scala   val mappedRdd = rdd.map(x => x * 2)`	`python   mapped_rdd = rdd.map(lambda x: x * 2)`
Collecter les résultats d'un RDD	`scala   val result = mappedRdd.collect()`	`python   result = mapped_rdd.collect()`
Créer une vue temporaire pour SQL	`scala   df.createOrReplaceTempView("table_temp")`	`python   df.createOrReplaceTempView("table_temp")`
Exécuter une requête SQL	`scala   val sqlDf = spark.sql("SELECT * FROM table_temp WHERE colonne1 > 50")`	`python   sql_df = spark.sql("SELECT * FROM table_temp WHERE colonne1 > 50")`
Cache des données	`scala   val cachedDf = df.cache()`	`python   cached_df = df.cache()`

Résolution d'un problème sous MapReduce et Spark

Je vais maintenant présenter un exemple de programme implémenté de deux différentes façons : sous MapReduce et SPARK.

L'exemple consiste à compter le nombre d'occurrences de chaque mot dans un fichier texte de **500 Mo** comme point de référence.

Implémentation sous MapReduce

Mapper (Python) :

```
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print(f"{word}\t1")
```

Reducer (Python) :

```
current_word = None
current_count = 0

for line in sys.stdin:
    word, count = line.strip().split('\t')
    count = int(count)
    if current_word == word:
        current_count += count
else:
 if current_word:
 print(f"{current_word}\t{current_count}")
 current_word = word
 current_count = count

if current_word:
 print(f"{current_word}\t{current_count}")
```

Pour exécuter avec Hadoop :

```
hadoop jar /path/to/hadoop-streaming.jar \
    -input /input_dir \
    -output /output_dir \
    -mapper mapper.py \
    -reducer reducer.py
```

Implémentation sous Spark en Python

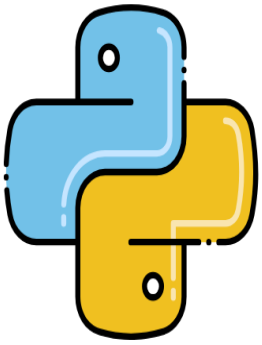
Code Spark en Python :

```
from pyspark import SparkContext

sc = SparkContext("local", "WordCount")
text_file = sc.textFile("file:///path/to/input.txt")

word_counts = (
    text_file
    .flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a + b)
)

word_counts.saveAsTextFile("file:///path/to/output")
```



Implémentation sous Spark en Scala

Code Spark en Scala :

```
import org.apache.spark.{SparkConf, SparkContext}

object WordCount {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("WordCount").setMaster("local")
    val sc = new SparkContext(conf)

    val textFile = sc.textFile("file:///path/to/input.txt")
    val wordCounts = textFile
      .flatMap(line => line.split(" "))
      .map(word => (word, 1))
      .reduceByKey(_ + _)

    wordCounts.saveAsTextFile("file:///path/to/output")
  }
}
```

Comparaison du code sous MapReduce, Spark Python et Spark Scala

Critère	MapReduce	Spark Python	Spark Scala
Langage	Python (Mapper et Reducer séparés)	Python (script unique)	Scala (script unique)
Complexité du code	Élevée (code en 2 parties)	Moyenne	Moyenne
Facilité d'exécution	Nécessite Hadoop Streaming	Simple avec Spark	Simple avec Spark
Modèle de calcul	Basé sur étapes explicites (Map et Reduce)	Transformations et actions sur RDD	Transformations et actions sur RDD
Complexité temporelle	**O(n + k log k)**	**O(n)**	**O(n)**
Performance	Moyenne (écriture/lecture disque fréquente)	Élevée (mémoire optimisée)	Élevée (mémoire optimisée)
Flexibilité	Faible, statique	Très flexible avec API Python	Très flexible avec API native Scala
Compatibilité Hadoop	Native	Compatible	Compatible
Interopérabilité	Limité à Hadoop	Intégration facile avec autres outils	Intégration facile avec autres outils
Temps d'exécution	15-20 min	2-5 min	1-3 min

Table V.2 : Comparaison des structures algorithmiques MapReduce, Spark (Python & Scala)

Voilà un autre exemple d'un traitement de données avec PySpark SPARK expliqué de détail. Imaginons que nous avons un fichier CSV contenant des informations sur des ventes, et nous voulons effectuer un traitement simple :

1. Charger les données
2. Filtrer les ventes d'un produit spécifique
3. Calculer la somme des ventes pour ce produit
4. Afficher les résultats

Installation de PySpark (si ce n'est pas déjà fait) : pip install pyspark

Code avec PySpark :

```
from pyspark.sql import SparkSession
```

Créer une session Spark

```
spark = SparkSession.builder.appName("TraitementVentes").getOrCreate()
```

Charger un fichier CSV dans un DataFrame Spark

```
df = spark.read.csv("ventes.csv", header=True, inferSchema=True)
```

Afficher les premières lignes du DataFrame pour vérifier le contenu : `df.show()`

Supposons que le CSV ait les colonnes : 'Produit', 'Quantite', 'Montant'

Filtrer les données pour un produit spécifique, par exemple "ProduitA"

```
df_produitA = df.filter(df['Produit'] == 'ProduitA')
```

Calculer la somme des montants des ventes pour "ProduitA"

```
somme_ventes_produitA = df_produitA.groupBy().sum('Montant').collect()
```

Afficher le résultat

```
print(f"Somme des ventes pour 'ProduitA': {somme_ventes_pro-
duitA[0]['sum(Montant)']}")
```

Fermer la session Spark : `spark.stop()`

Maintenant, je vais expliquer le code

- **SparkSession** : Nous créons d'abord une session Spark, qui est nécessaire pour manipuler des DataFrames.

- **Chargement du fichier CSV** : Nous chargeons un fichier CSV avec spark.read.csv(). L'option header=True indique que le premier en-tête du fichier contient les noms de colonnes. L'option inferSchema=True permet à Spark de deviner automatiquement les types de données.
- **Filtrage** : Nous filtrons les ventes correspondant à un produit particulier (ici "ProduitA") à l'aide de la méthode filter().
- **Agrégation** : Nous utilisons groupBy() pour regrouper les lignes et calculer la somme de la colonne Montant.
- **Affichage du résultat** : Nous récupérons le résultat sous forme de liste via collect() et affichons la somme des ventes pour le produit spécifique.
- **Fermeture de la session** : Nous fermons la session Spark après le traitement avec spark.stop().

L'exécution de ce code va afficher le résultat suivant :

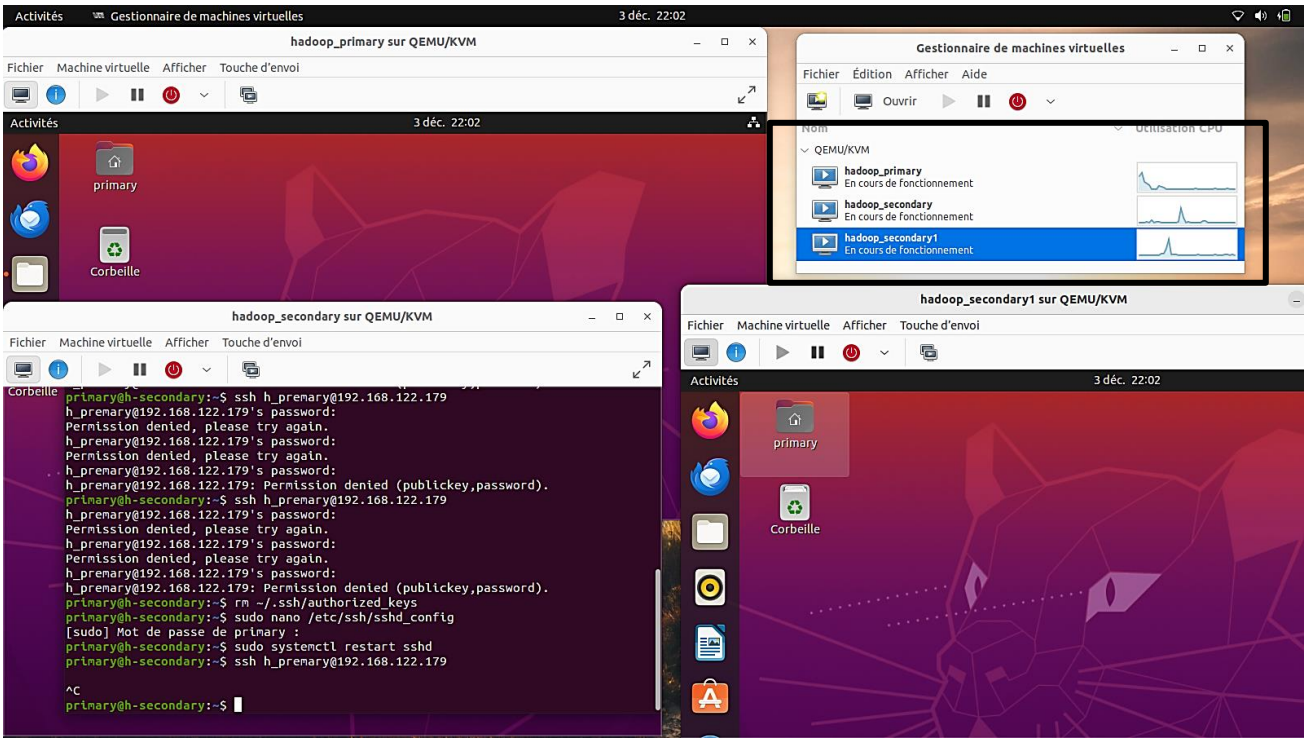
```
+---------+--------+-------+
| Produit | Quantite| Montant|
+---------+--------+-------+
| ProduitA|   5     | 100.0 |
| ProduitA|   10    | 200.0 |
+---------+--------+-------+
Somme des ventes pour 'ProduitA': 300.0
```

Cet exemple montre comment Spark peut être utilisé pour effectuer un traitement de données de manière simple et rapide sur de grands ensembles de données. Vous pouvez facilement étendre ce processus pour effectuer des traitements plus complexes comme des jointures, des calculs statistiques, etc.

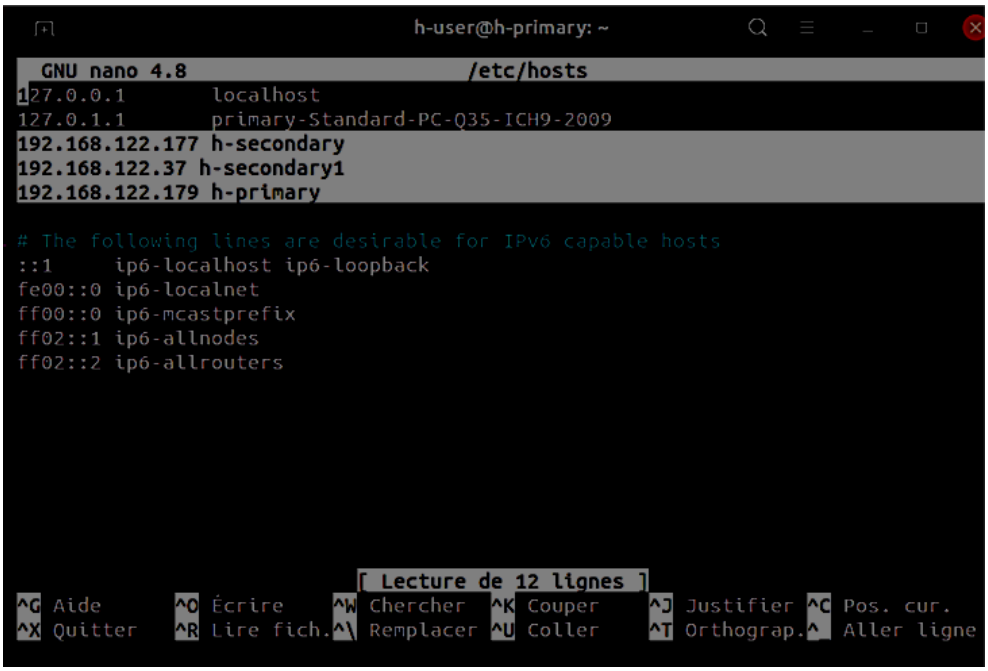
Annexe I : Installation de Hadoop en environnement multi-nœud

Ce TP présente les étapes d'installation et la configuration de Hadoop dans un environnement multi-nœud (en mode pseudo-distribué).

- Nous allons utiliser KVM pour créer les machines virtuelles. Pour cela, nous avons créé 3 machines virtuelles dotées du système d'exploitation Linus Ubuntu avec 1.5 Go de RAM chacune et un core.



- Nous avons nommé chaque machine afin de l'identifier.



- Configurez SSH pour permettre une connexion sans mot de passe entre le nœud maître (primary) et les nœuds esclaves (secondary).

  SSH ou Secure Shell, est un protocole réseau sécurisé qui permet d'établir une connexion à distance entre deux ordinateurs, de manière cryptée, sur un réseau non sécurisé. Il est couramment utilisé pour administrer des serveurs et échanger des données de manière sécurisée.

- Configuration des fichiers principaux :

core-site.xml

hdfs-site.xml

mapred-site.xml

yarn-site.xml

workers

Le fichier hdfs-site.xml contient la configuration du NameNode et des DataNodes avec notamment l'emplacement du stockage de l'historique des transactions et des blocks.

On y définit aussi le coefficient de réplication :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>3</value>
  <description>The actual number of replications can be specified when the file is created.</des
cription>
 </property>
<property>
    <name>dfs.data.dir</name>
    <value>/srv/hadoop/datanode</value>
</property>
<property>
    <name>dfs.name.dir</name>
    <value>/srv/hadoop/namenode</value>
</property>
</configuration>
```

- Maintenant, pour exécuter un job MapReduce, ce job consiste à compter le nombre d'occurrences de chaque mot dans un fichier texte. Pour cela, nous allons créer un fichier nommé WordCount.java. Le programme est réalisé avec le langage java.

La structure générale du programme est comme suit :

Classe « *Mapper* » : Analyse les lignes d'entrée, divise en mots, et émet des paires clé-valeur (mot, 1).

Classe « *Reducer* » : Regroupe toutes les valeurs associées à chaque clé (mot) et calcule leur somme.

Classe « *Main* » : Configure et exécute le job MapReduce.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
public class WordCount {

// Mapper Class
```

```
    public static class TokenizerMapper extends Mapper<Object,
Text, Text, IntWritable> {
        private final static IntWritable one = new
IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context
context) throws IOException, InterruptedException {
            String[] tokens = value.toString().split("\\s+");
            for (String token : tokens) {
                word.set(token);
                context.write(word, one);
            }
        }
    }
// Reducer Class
    public static class IntSumReducer extends Reducer<Text,
IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable>
values,        Context        context)        throws        IOException,
InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    // Main Method
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

```
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Compiler le fichier Java : *javac -classpath $(hadoop classpath) -d . WordCount.java*

Emballez le fichier compilé en un JAR exécutable : *jar -cvf WordCount.jar -C . .*

On doit créer un fichier texte (input.txt) qui servira d'entrée au programme et le mettre sur le HDFS.

Voilà un exemple de fichier texte :

Install Hadoop
Run Hadoop Wordcount Mapreduce Example

Cette commande permet de créer un répertoire input dans hdfs pour stocker les données : *hadoop fs -mkdir -p /user/input*

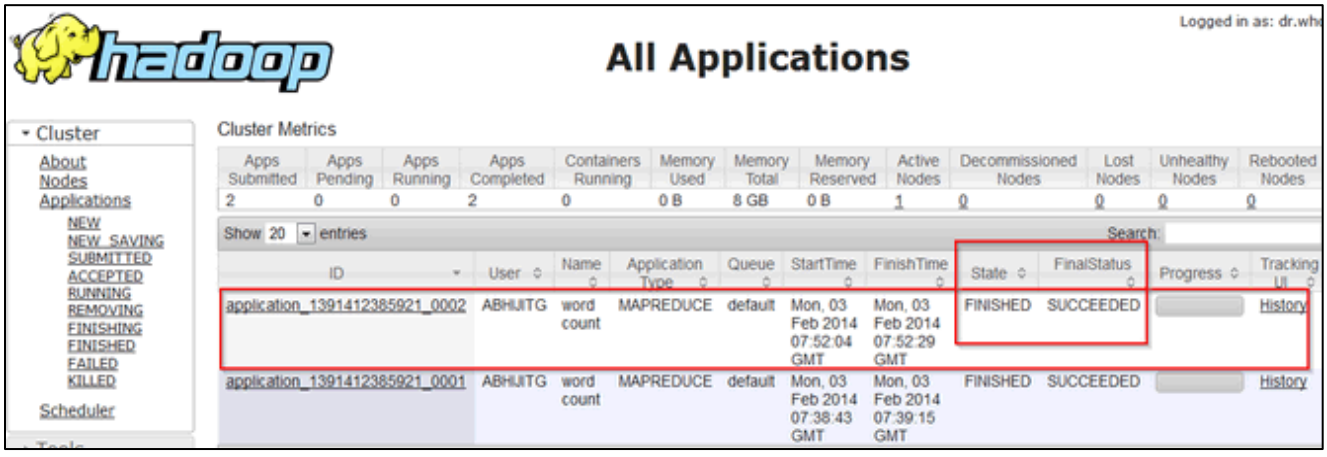
Cette commande permet de d'ajouter les données sur le HDFS, dans ce cas nous avons ajouté le fichier input.txt dans le répertoire input de HDFS créé au préalable :

hadoop fs -put input.txt /user/input

Pour exécuter le programme :

hadoop jar WordCount.jar WordCount /user/input /user/output, output est le répertoire où on vas stocker les résultat.

Hadoop exécute le job et affiche son état comme un job accompli :



Pour vérifier les résultats : *hadoop fs -cat /user/output/part-r-00000*

```
C:\hadoop>bin\hdfs dfs -cat output/*
Example 1
Hadoop  2
Install 1
Mapreduce       1
Run     1
Wordcount       1
```

Comment le programme distribue-t-il les tâches ?

- ✓ Hadoop divise le fichier texte en blocs et les attribue aux DataNodes.
- ✓ Chaque bloc est traité par un nœud différent.
- ✓ Le **Mapper** traite les blocs individuellement, compte les occurrences de chaque mot et génère des paires (mot, 1).
- ✓ Le Reducer combine les résultats de tous les nœuds pour calculer les totaux finaux.

Annexe II : Installation de VirtualBox sous Windows

Le logiciel **VirtualBox** est un logiciel de virtualisation. Ce logiciel permet de créer une machine virtuelle (Pc ou ordinateur) et de le faire vivre comme un vrai ordinateur. Ainsi sur cette machine on pourra installer un système d'exploitation et l'utiliser comme une véritable machine physique bien qu'elle n'en soit pas une.

En plus simple vous allez démarrer votre véritable Pc tournant sous Windows 10 par exemple et sans quitter votre session vous pourrez lancer dans une fenêtre un autre système d'exploitation comme un **Linux Ubuntu**.

C'est dans le domaine des serveurs que l'on emploie le plus la virtualisation. En règle générale un serveur est conçu pour exécuter un seul système d'exploitation et une application à la fois. Avec ce principe à peine 10% des capacités du serveur sont utilisés. Avec la puissance d'un seul serveur on arrive à virtualiser l'existence de plusieurs machines qui font tourner autant de systèmes d'exploitations (mêmes différents) et autant d'applications. Ainsi la totalité de la puissance du serveur peut être utilisé et les départements informatiques peuvent réaliser de grosse économie dans l'achat de matériel.

Dans le contexte du module Big Data, vous pouvez utiliser la virtualisation et simuler votre machine à une machine de bonne configuration matérielle, vous pouvez aussi installer les outils nécessaires sans nuire au fonctionnement de votre machine.

Voilà comment télécharger l'outil Virtualbox le plus approprié à nos travaux pratiques, bien qu'il existe bien d'autres outils gratuits.

- Assurez-vous d'abord que votre machine accepte la virtualisation, en général en se référant à votre BIOS.
- Rendez-vous sur le site de VirtualBox *: https://www.virtualbox.org/*. Vous arrivez sur la page d'accueil de *VirtualBox,* où vous allez pouvoir télécharger le fichier d'installation.

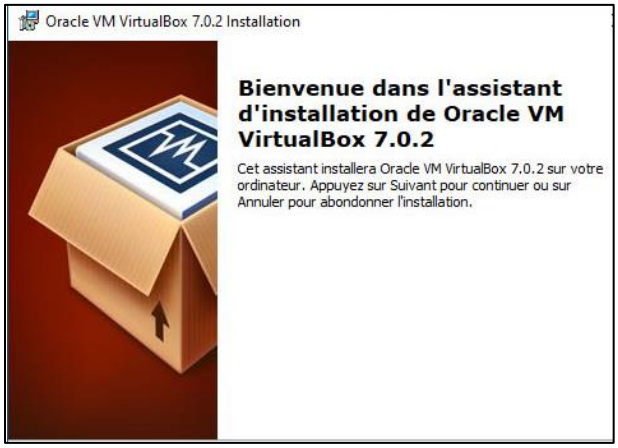
- Cliquez sur Windows hosts. Sous le titre *VirtualBox 5.2.8 platform packages*, vous trouverez ce lien en bleu. Le fichier exécutable d'installation (EXE) de *VirtualBox* est alors en voie de transfert sur votre disque dur.

- Ouvrez le fichier exécutable *VirtualBox EXE*. Retrouvez le fichier exécutable (par exemple, dans le dossier *Téléchargements*), puis double cliquez dessus. L'installation de *VirtualBox* peut alors commencer.

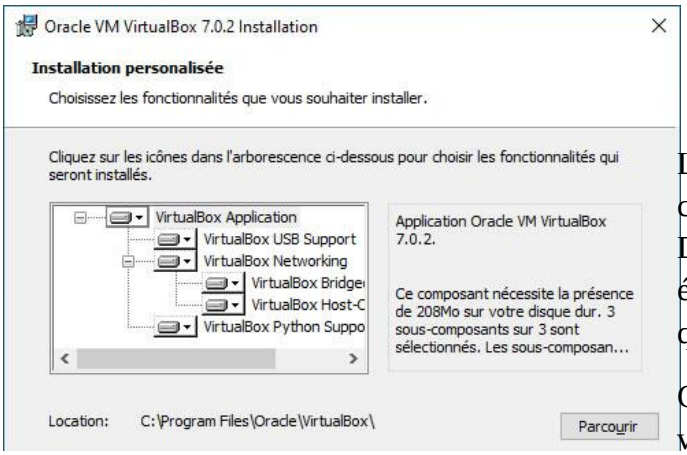
- Lisez et suivez les instructions. Opérez exactement comme suit :
  - cliquez sur *Suivant* sur les trois premières pages,
  - cliquez sur *Oui* au moment voulu,
  - cliquez sur *Installer*,
  - cliquez sur *Oui* au moment voulu.

- Cliquez finalement sur Installer. VirtualBox commence alors son installation sur votre disque dur.

Maintenant, nous allons installer et configurer Virtual Box sur notre machine. Après décompression, cliquer sur le fichier exécutable.



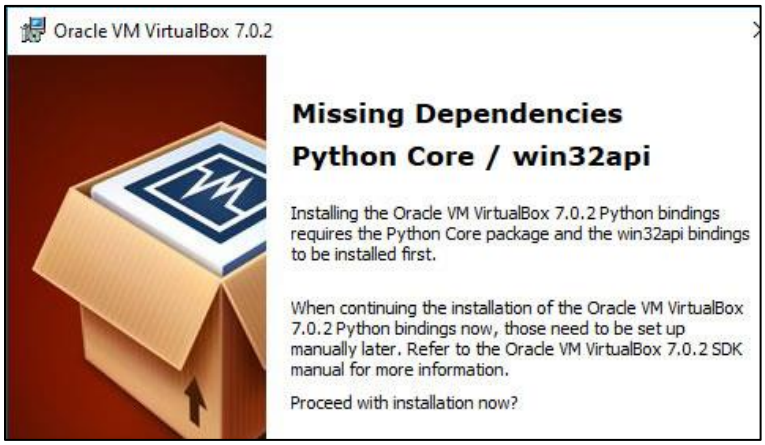
Sous VirtualBox 7, les composants utilisés sont toujours les mêmes.
Cliquez sur Suivant.



Lors de l'installation de VirtualBox, votre connexion réseau sera réinitialisée.
D'où l'apparition de cet avertissement pour éviter de couper un téléchargement important qui serait en cours (par exemple).

Cela est dû à l'installation de cartes réseau virtuelles sur le PC physique qui serviront à fournir l'accès au réseau et/ou à Internet à vos machines virtuelles VirtualBox.

Cliquer sur Oui



Depuis VirtualBox 7, une étape "Missing Dependencies Python Core / win32api" apparaitra.
Cliquez sur Oui pour installer automatiquement les dépendances nécessaires

Cliquez ensuite sur Installer

VirtualBox 7 a été installé. Cliquez sur Terminer.

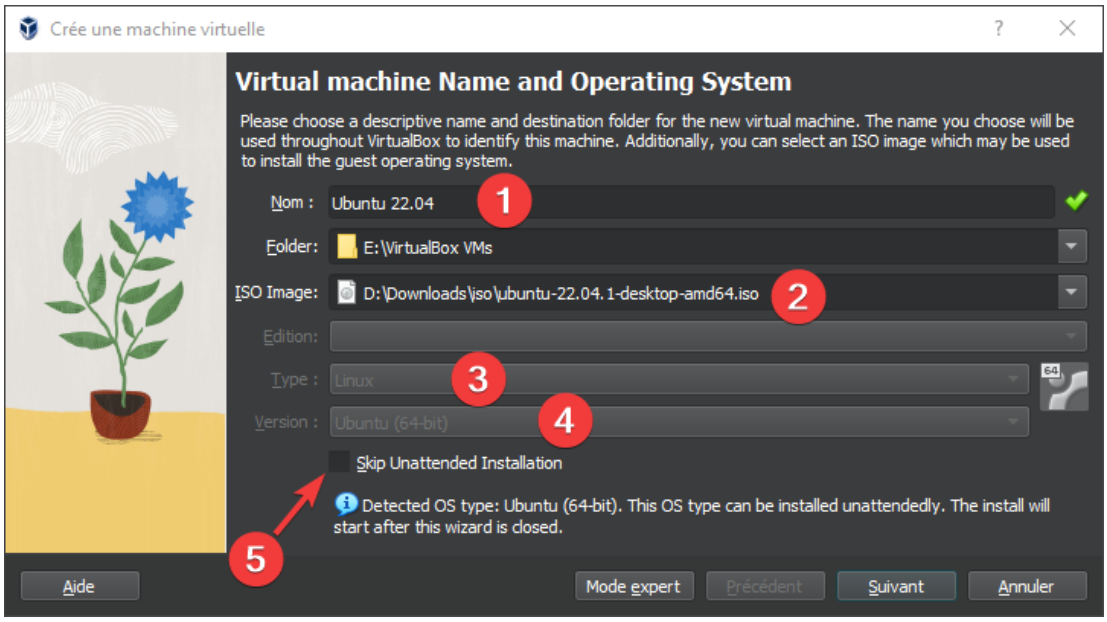
Voilà enfin l'interface de VirtualBox apparait comme suit :

L'assistant "Créer une machine virtuelle" apparait et vous permet de :

- Nom : indiquer un nom pour votre machine virtuelle.
  Le nom sous lequel celle-ci apparaitra dans VirtualBox.
- Folder : choisir l'emplacement où vous souhaitez stocker votre nouvelle machine virtuelle. Par défaut, le dossier "C:\Users\[nom de l'utilisateur actuel]\VirtualBox VMs".
- ISO Image : sélectionner le fichier ISO du DVD d'installation du système d'exploitation (Windows, Linux, ...) que vous souhaitez installer dans votre machine virtuelle.
- Edition : l'édition de Windows que vous souhaitez installer (si plusieurs éditions se trouvent sur le DVD d'installation sélectionné ci-dessus).
- Type : le type de système d'exploitation que vous souhaitez installer (Windows, Linux, ...).
  Cette valeur sera sélectionnée automatiquement en fonction du nom que vous indiquerez pour votre machine virtuelle.
- Version : la version du système d'exploitation que vous souhaitez installer (Windows 10, Windows 8.1, version du noyau Linux, ...).
  A nouveau, cette valeur sera généralement sélectionnée automatiquement en fonction du nom de votre machine virtuelle.
- Skip Unattended Installation : cocher cette case permet de désactiver la configuration automatique du système d'exploitation invité.
  Ce qui vous permettra de l'installer manuellement.
- Pour commencer, indiquez un nom pour votre machine virtuelle, changez l'emplacement où la stocker si vous le souhaitez, puis sélectionnez l'option "Autre" dans la liste "ISO Image".

Maintenant, nous allons présenter les étapes de configuration de notre machine virtuelle.

Étape 1 : Nommer la machine virtuelle et choisir le système installé



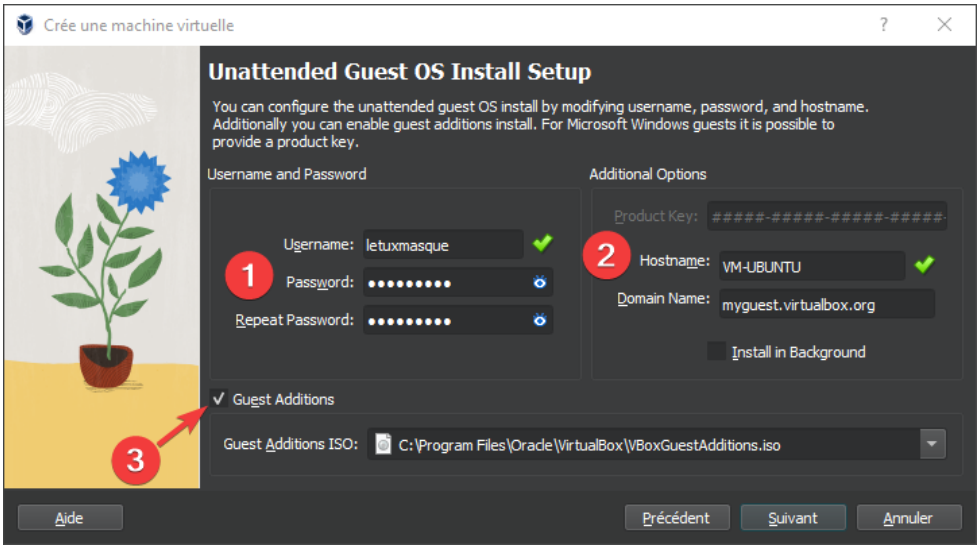
Commencez par donner un nom à votre machine virtuelle (1). Si **le nom choisi comprend le nom du système d'exploitation** (OS) à installer, par exemple dans notre cas *Ubuntu*, alors **VirtualBox sélectionnera automatiquement**, pour vous, le bon **Type** (3) et la bonne **Version** (4).

La sélection de l'image ISO d'Ubuntu (2) est facultative. Elle peut se faire plus tard, par exemple lors du premier démarrage de la VM.

Si vous la choisissez à cette étape, elle permet aussi de déterminer automatiquement pour vous le type et version. De plus, elle permet, une fois les différentes étapes de l'assistant passées et parce que c'est Ubuntu (ce n'est pas le cas avec tous les OS), de débuter directement l'installation automatique du système.

Puis, faites Suivant.

Étape 2 : *Configuration pour l'installation automatique*



Si vous avez choisi précédemment de faire une installation automatique, vous entrez ici les éléments nécessaires pour cette dernière :

- le nom de l'utilisateur et son mot de passe (**1**). Il sera le compte administrateur du système
- le nom de machine (**2**). Vous pouvez laisser le nom de domaine par défaut si vous n'en avez pas.

Puis, si vous voulez installer automatiquement les *Additions Invité…* après l'installation du système, cochez l'option (**3**) et sélectionnez l'image ISO de ces derniers.

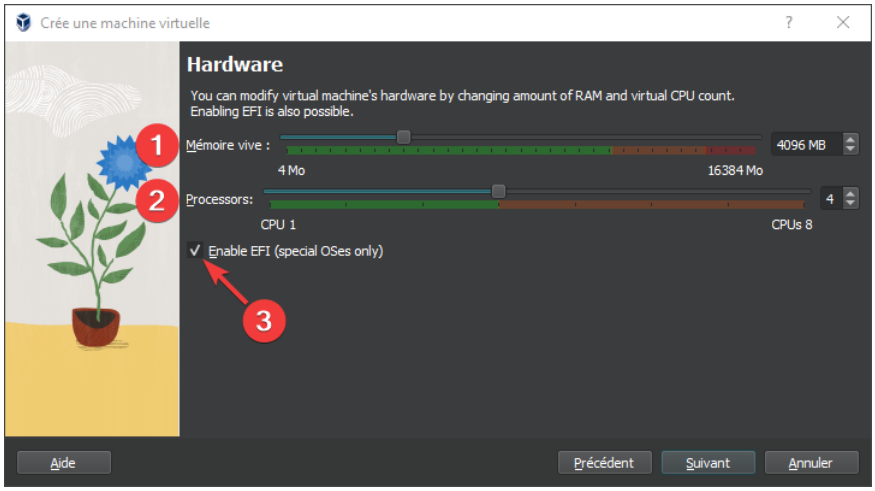
Par défaut, sous Windows, vous les trouverez dans :

`C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso`

Et sous Linux dans : `/usr/share/virtualbox/VBoxGuestAdditions.iso`

Cliquer ensuite sur suivant :

Étape 3 : Configuration du matériel virtuel



Comme pour un ordinateur physique, la machine virtuelle a besoins de mémoire vive (RAM) et d'un CPU (ou plus) pour fonctionner.

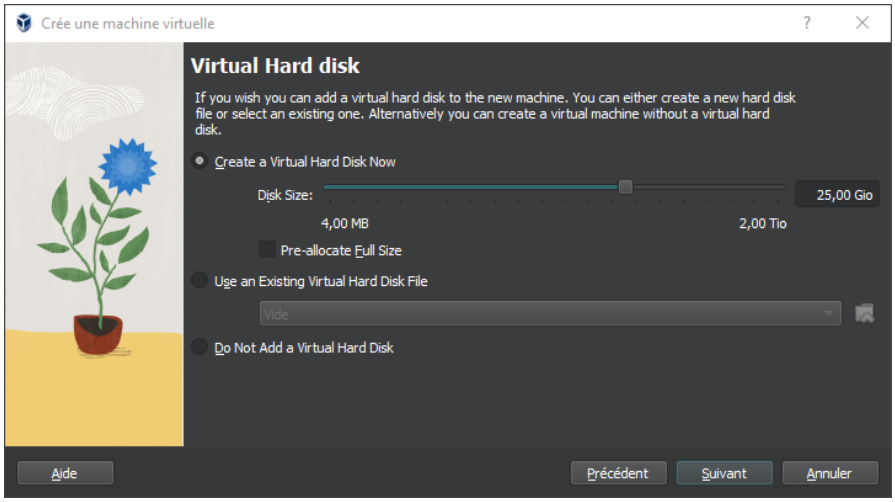
Par défaut, pour Ubuntu, VirtualBox propose :

- un minimum **2 Go** (2048 Mo) de RAM (**1**). Si votre machine physique le permet, vous pouvez décider d'en affecter plus à la VM, toutefois sans mettre le curseur dans le rouge.
- **1 CPU** (**2**). Je recommande d'allouer le plus de CPU disponibles à la machine virtuelle, mais sans aller dans la zone rouge…

Aussi, vous pouvez choisir d'émuler un BIOS UEFI (**3**), comme sur la plupart des machines vendues aujourd'hui.

Cliquer sur suivant :

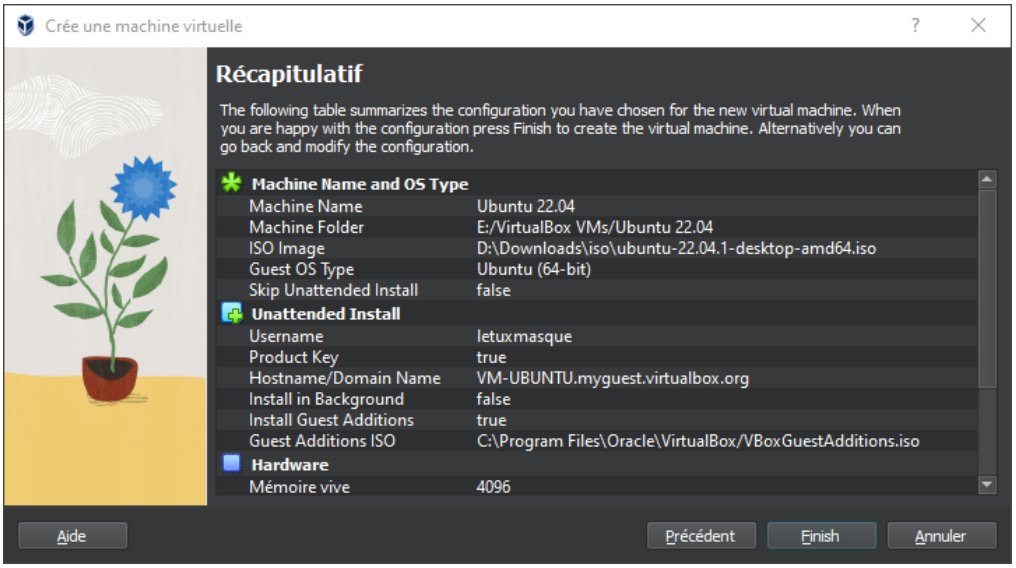
Étape 4 : Création d'un disque dur virtuel



Pour installer le système d'exploitation, il faut aussi un disque dur virtuel. La taille minimum proposée par VirtualBox pour Ubuntu est de 25 Go. *C'est un minimum*.

Enfin, par la suite, si vous manquez de place sur le disque virtuel, vous pourrez toujours modifier cette taille. Pour cela, n'hésitez pas à consulter le tutoriel.

Étape 5 : Récapitulatif



Vous avez dans cette fenêtre un résumé de la configuration précédemment faite.

Si vous avez choisi de faire l'installation automatique, en cliquant sur le bouton **Finish**, vous la lancerez.

Sinon, vous reviendrez au gestionnaire. Alors, vous pourrez modifier les paramètres de votre VM et la démarrer pour installer le système d'exploitation.

Annexe III : Exercices corrigés

Exercice 1

Considérons un dossier HDFS nommé patchesFolder qui contient deux fichiers : patches2018.txt de taille 513MB et patches2019.txt de taille 515MB. Supposons que le cluster Hadoop utilisé peut supporter jusqu'à 5 instances mapper en parallèle. Le nombre d'instances reducer est 2 et la taille d'un bloc HDFS est 512MB.

Quel est le nombre d'instances du mapper utilisées lorsqu'on exécute une application MapReduce sur les deux fichiers du dossier patchesFolder ?

Solution

Dossier patchesFolder :

Fichier1 : patches2018.txt, taille= 513MB

Fichier2 : patches2019.txt, taille= 515MB

Nombre maximum d'instances Mapper : 5 Bloc HDFS = 512MB.

Fichier1 : on utilise 2 blocs

Fichier2 : on utilise 2 blocs

Le nombre d'instances Mapper utilisées lorsqu'on exécute une application MapReduce sur les deux fichiers du dossier patchesFolder doit supporter les quatre blocs des données en entrée. On utilise donc 04 instances de Mapper.

Exercice 2

On veut exécuter un programme MapReduce qui permet de sélectionner les lignes du fichier logs.txt qui contiennent les mots « ERROR » ou « WARNING ».

Sachant que la taille du fichier logs.txt est 5000 MB, quelle est la taille du block HDFS qui sera choisi si on veut forcer Hadoop à exécuter 10 mappers en parallèle pour le programme MapReduce sur le fichier logs.txt

1. Block size: 5000MB
2. Block size: 2048MB

3. Block size: 1024MB
4. Block size: 512MB

Solution

Taille du fichier logs.txt : 5000 MB

Pour forcer Hadoop à exécuter 10 mappers en parallèle, on doit diviser le fichier en 10 blocks, la taille de chaque block sera : 5000 / 10= 500 MB, la réponse choisie est donc : **4) Block size=512 MB**

Exercice 3

Considérons deux fichiers HDFS, logs1.txt et logs2.txt de tailles 1036MB et 500MB respectivement. Supposons que le facteur de réplication est 4 (c.-à-d. nombre de copies de chaque block) et que la taille du block HDFS est 512MB.

Quel est le nombre total de blocks sont utilisés pour stocker les deux fichiers logs1.txt et logs2.txt (Attention : considérer aussi les copies).

1. 3 blocks
2. 4 blocks
3. 12 blocks
4. 16 blocks

Solution

Taille du fichier logs1.txt: 1036MB

Taille du fichier logs2.txt : 500MB

Facteur de réplication : 4

Taille du block : 512 MB

Nombre de blocks nécessaires pour stocker le fichier logs1.txt : 1036 / 512= 2.023 soient 3 blocks

Nombre de blocks après réplication : 3*4 =12 blocks

Nombre de blocks nécessaires pour stocker le fichier logs2.txt : 500 / 512= 0.97 soit 1 block

Nombre de blocks après réplication : 1*4 =4 blocks Nombre total de blocks =16 blocks **La réponse 4) est choisie.**

Annexe IV : QCM

Voici un **QCM de 20 questions** sur le **Big Data**. Ce questionnaire couvre divers aspects de l'écosystème Big Data, des outils utilisés, ainsi que des concepts et des technologies associés.

Ce QCM couvre une large gamme de concepts et d'outils associés au Big Data et à l'écosystème Hadoop. Vous pouvez l'utiliser pour évaluer vos connaissances de base en Big Data ou pour vous préparer à un examen sur ce sujet.

Question 1 : Qu'est-ce que le Big Data ?

a) Des données d'une taille inférieure à 1 To.

b) Des données structurées uniquement.

c) Des ensembles de données volumineux, complexes et variés.

d) Des données accessibles uniquement via des bases de données relationnelles.

Réponse correcte : c) Des ensembles de données volumineux, complexes et variés.

Question 2 : Quel est le système de fichiers utilisé dans Hadoop pour stocker des données massives ?

a) NTFS

b) HDFS (Hadoop Distributed File System)

c) FAT32

d) ZFS

Réponse correcte : b) HDFS (Hadoop Distributed File System)

Question 3 : Quel modèle de programmation est utilisé dans Hadoop pour le traitement des données ?

a) MapReduce

b) SQL

c) OOP (Programmation Orientée Objet)

d) Lambda

Réponse correcte : a) MapReduce

Question 4 : Quel est le rôle de YARN dans l'écosystème Hadoop ?

a) Gérer les utilisateurs et la sécurité.

b) Gérer les ressources du cluster et l'exécution des tâches.

c) Gérer le stockage des données sur HDFS.

d) Orchestrer les requêtes SQL.

Réponse correcte : b) Gérer les ressources du cluster et l'exécution des tâches.

Question 5 : Quelle technologie est utilisée pour effectuer des requêtes SQL sur les données stockées dans HDFS ?

a) Spark

b) Pig

c) Hive

d) Mahout

Réponse correcte : c) Hive

Question 6 : Quel outil Hadoop est utilisé pour importer et exporter des données entre Hadoop et une base de données relationnelle ?

a) HBase

b) Sqoop

c) Flume

d) Oozie

Réponse correcte : b) Sqoop

Question 7 : Quel outil Hadoop permet de traiter des données en temps réel en mémoire ?

a) Tez

b) Spark

c) HBase

d) Pig

Réponse correcte : b) Spark

Question 8 : Quel composant Hadoop permet de traiter des données non structurées à l'aide de scripts ?

a) HBase

b) Hive

c) Pig

d) Sqoop

Réponse correcte : c) Pig

Question 9 : Quelle base de données est utilisée dans Hadoop pour stocker des données structurées en temps réel ?

a) HDFS

b) MongoDB

c) HBase

d) Cassandra

Réponse correcte : c) HBase

Question 10 : Quel est le rôle de Flume dans l'écosystème Hadoop ?

a) Importer et exporter des données entre Hadoop et une base de données relationnelle.

b) Collecter, agréger et transférer des flux de données en temps réel dans HDFS.

c) Exécuter des calculs de machine learning sur Hadoop.

d) Orchestrer les flux de données dans des applications Java.

Réponse correcte : b) Collecter, agréger et transférer des flux de données en temps réel dans HDFS.

Question 11 : Quelle est la principale différence entre Hadoop et un système de gestion de bases de données relationnelles (SGBDR) ?

a) Hadoop est plus rapide pour des petites quantités de données.

b) Hadoop est conçu pour traiter des données non structurées et massives, tandis que les SGBDR gèrent des données structurées.

c) Hadoop ne nécessite pas de machine physique pour fonctionner.

d) Hadoop stocke toutes les données dans des bases relationnelles.

Réponse correcte : b) Hadoop est conçu pour traiter des données non structurées et massives, tandis que les SGBDR gèrent des données structurées.

Question 12 : Quel composant Hadoop est utilisé pour planifier et automatiser les workflows ?

a) Oozie

b) Tez

c) Sqoop

d) Hive

Réponse correcte : a) Oozie

Question 13 : Qu'est-ce que le terme "3V" dans le contexte du Big Data ?

a) Volume, Vitesse, Valeur

b) Vitesse, Variété, Vérification

c) Volume, Variété, Vérification

d) Volume, Variété, Vérité

Réponse correcte : a) Volume, Vitesse, Valeur

Question 14 : Quel est le principal avantage de l'utilisation de HDFS dans Hadoop ?

a) Les données sont stockées sur un seul serveur centralisé.

b) Il permet d'effectuer des traitements de données en temps réel.

c) Il permet de stocker des données massives de manière distribuée et tolérante aux pannes.

d) Il permet d'exécuter des requêtes SQL sur des données non structurées.

Réponse correcte : c) Il permet de stocker des données massives de manière distribuée et tolérante aux pannes.

Question 15 : Quel outil Hadoop est spécifiquement conçu pour le machine learning ?

a) Mahout

b) HBase

c) Pig

d) Spark

Réponse correcte : a) Mahout

Question 16 : Quelle est la principale fonction de Zookeeper dans l'écosystème Hadoop ?

a) Gérer le stockage des données dans HDFS.

b) Orchestrer et coordonner les services dans un cluster Hadoop distribué.

c) Analyser les données stockées dans HDFS.

d) Planifier l'exécution de tâches de traitement.

Réponse correcte : b) Orchestrer et coordonner les services dans un cluster Hadoop distribué.

Question 17 : Quelle est la fonction principale de MapReduce dans Hadoop ?

a) Effectuer des calculs interactifs en temps réel.

b) Gérer les erreurs dans les systèmes distribués.

c) Diviser un processus en petites tâches parallèles pour un traitement massif de données.

d) Importer des données depuis des bases de données relationnelles.

Réponse correcte : c) Diviser un processus en petites tâches parallèles pour un traitement massif de données.

Question 18 : Dans l'écosystème Hadoop, quel composant est utilisé pour effectuer un traitement interactif des données avec des performances accrues par rapport à MapReduce ?

a) Hive

b) Pig

c) Spark

d) Oozie

Réponse correcte : c) Spark

Question 19 : Quel est le principal avantage d'utiliser des bases de données NoSQL comme HBase avec Hadoop ?

a) Une meilleure capacité de stockage pour les données structurées.

b) La possibilité de gérer des données relationnelles complexes.

c) La capacité de traiter des données massives et non structurées avec un faible temps de réponse.

d) L'intégration automatique avec des systèmes de fichiers relationnels.

Réponse correcte : c) La capacité de traiter des données massives et non structurées avec un faible temps de réponse.

Question 20 : Quel est le rôle principal de la technologie Tez dans Hadoop ?

a) Remplacer MapReduce pour des traitements plus rapides et plus interactifs.

b) Gérer le stockage des données dans HDFS.

c) Réaliser des calculs de machine learning sur des données massives.

d) Orchestrer des workflows ETL.

Réponse correcte : a) Remplacer MapReduce pour des traitements plus rapides et plus interactifs.

Conclusion

En conclusion, le Big Data représente une révolution technologique qui transforme les secteurs économiques, scientifiques et sociaux. L'analyse de grandes quantités de données permet d'obtenir des insights précieux pour la prise de décisions stratégiques, l'optimisation des processus et la personnalisation des services. Toutefois, l'exploitation efficace du Big Data nécessite des outils et des compétences spécialisées, ainsi qu'une gestion rigoureuse de la confidentialité et de la sécurité des données.

Les technologies de traitement de données massives, telles que le machine Learning, l'intelligence artificielle, et les systèmes de stockage distribués, ont ouvert la voie à des applications variées dans des domaines tels que la santé, les finances, le marketing, et l'industrie. Cependant, cette explosion des données soulève aussi des questions éthiques, en particulier en matière de vie privée et de gouvernance des données, qui doivent être traitées de manière responsable.

Le Big Data est donc à la fois un défi et une opportunité. Pour en tirer parti pleinement, les entreprises et organisations doivent non seulement investir dans des technologies et des infrastructures adaptées, mais aussi développer une culture de l'innovation et de la gestion des données. À l'avenir, le Big Data continuera à évoluer, avec des avancées dans la qualité des données, l'automatisation de l'analyse, et l'intégration de l'IA pour encore plus d'efficacité et de précision.

Ainsi, en adoptant les bonnes pratiques et en surmontant les défis associés, le Big Data sera un levier majeur pour créer de la valeur ajoutée et construire un avenir plus connecté et plus intelligent.

J'espère que ce document sera utile à mes étudiants, à mes collègues et à tout professionnel.

Références

[1] **Tom White (2015).** *Hadoop: The Definitive Guide, Fourth Edition.* O'Reilly Media.

[2] https://waytolearnx.com/ dernier accès 01/01/2025.

[3] https://hadoop.apache.org/ dernier accès 01/01/2025.

[4] V. K. Jain (2017). *Big Data and Hadoop.* Khanna Publishing House.

[5] https://cedric.cnam.fr/vertigo/Cours/RCP216/tpSparkScala.html dernier accès 01/01/2025.

[6] Mes supports de cours sur la plateforme e-learning de l'université : https://elearning.univ-bejaia.dz/course/view.php?id=13072.

[7] *Michel Adiba, Jean-Pierre Giraudin Du Big Data à l'IA, 2024, Collection Prométhée, Edition Eyrolles.*

[8] Nastasia Saby, Sébastien Chazallet , Apache Spark et Python, 2024 Edition Eyrolles

[1] Tom White (2015). *Hadoop: The Definitive Guide, Fourth Edition.* O'Reilly Media.

[2] https://waytolearnx.com/ dernier accès 01/01/2025.

[3] https://hadoop.apache.org/ dernier accès 01/01/2025.

[4] V. K. Jain (2017). *Big Data and Hadoop.* Khanna Publishing House.

[5] https://cedric.cnam.fr/vertigo/Cours/RCP216/tpSparkScala.html dernier accès 01/01/2025.

[6] Mes supports de cours sur la plateforme e-learning de l'université : https://elearning.univ-bejaia.dz/course/view.php?id=13072.

[7] *Michel Adiba, Jean-Pierre Giraudin Du Big Data à l'IA, 2024, Collection Prométhée, Edition Eyrolles*

[8] Nastasia Saby, Sébastien Chazallet , Apache Spark et Python, 2024 Edition Eyrolles