MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS -

MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

RAPPORT DE MINI-PROJET

Option : **Ingénierie des Systèmes d'Information**

THEME :

Intégration des Méthodes de Partitionnement pour L'interrogation des Données Sémantiques dans les Environnements Big Data

Etudiant(e) : « **SAHRAOUI AFIF** »

Etudiant(e) : « **SMAINE Mohamed amine** »

Encadrant(e) : « BENHAMED Siham »

Année Universitaire 2020-2021

Résumé

La représentation des données sémantiques dans le web à travers le RDF (Resource Description Framework) est en forte augmentation ces dernières années. Le traitement des ensembles de données RDF par les solutions classiques des systèmes centralisés engendre différents problèmes au niveau de l'accès à la donnée et sa récupération. Pour ce faire, il existe des solutions qui ont été proposées récemment afin de gérer de grandes quantités de données sémantiques à grandes échelle. Afin d'améliorer le traitement des données RDF, des méthodes de partitionnement sont intégré au niveau du stockage et / ou le requêtage de ces données. Le partitionnement des données permet de réduire la quantité de données à traiter et le partitionnement de la requête permet de réduire le nombre de calculs à déclencher dans les systèmes de traitement de requêtes SPARQL à l'échelle Big Data.

Mots-clés : RDF, SPARQL, traitement parallèle, web sémantique, méthode de partitionnement, Big Data

Abstract

The representation of semantic data in the web through the RDF (Resource Description Framework) has increased sharply in recent years. The processing of RDF datasets by traditional solutions in centralized systems creates various problems in terms of data access and retrieval. To do this, there are solutions that have been proposed recently in order to manage large amounts of semantic data on a large scale. In order to improve the processing of RDF data, partitioning methods are integrated at the storage level and / or the query of this data. Data partitioning reduces the amount of data to be processed and query partitioning reduces the number of calculations to be triggered in SPARQL big data-scale query processing systems.

Keywords: RDF, SPARQL, parallel processing, semantic web, partitioning method, Big Data

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	Page
NoSQL	Not only SQL	13
W3C	World Wide Web Consortium	17
XML	eXtensible Markup Language	18
URI	Uniform Resource Identifier	18
URL	Uniform Resource Locator	18
IRI	International Resource Identifier	18
URIs	Schéma URI	18
DTD	Document Type Definition	18
RDF	Ressource Description Framework	19
OWL	Web Ontology Language	21
FOAF	Friend of a friend	21
SPARQL	Simple Protocol and RDF Query Language	24
TP	Table des propriété	25
RDD	Resilient Distributed Dataset	32

Table des matières

Introduction Générale

De nos jours, des très grandes quantités de données circulent de manière massive, à partir de diverses sources telles que les réseaux sociaux, internet, Google, appareils mobiles, système GPS… etc. Ces données circulent d'une manière rapide et en temps réel et à grande échelle. Elles sont de différentes types et structures, on trouve des textes, des audio, des images et des vidéos. Par conséquent, avec l'augmentation de ces données, le terme Big data est apparu.

Le défi confronté dans ce contexte par les systèmes classiques et les entrepôts de données est le problème de la dégradation des performances face à une quantité de données aussi importante en termes d'analyse et de traitement.

Cependant, ce problème a été résolu par le Big data. Par conséquent, il existe plusieurs domaines qui utilisent le Big data comme une solution optimale pour la gestion des données, parmi ces domaines, il y a le domaine de sélection des candidats qu'il profite de Big Data pour améliorer la sélection dans les entreprises.

La sémantique est un aspect essentiel qui permet d'aboutir à une interaction automatique et intelligente.

Dans le traitement des données sémantique à l'échelle Big data, plusieurs modèles de programmation ont été utilisés, et l'une des techniques les plus puissantes de traitement et l'analyse des données est le Framework Hadoop et Spark. Ces derniers ont un environnement d'exécution distribuée, performant et scalable. Par exemple, Hadoop propose un système de stockage distribué via son système de fichier HDFS (Hadoop Distributed File System) et un système d'analyse et de traitement de données basé sur le modèle de programmation MapReduce pour réaliser des traitements parallèles et distribués sur des gros volumes de données.

MPI a été écrite pour obtenir de bonnes performances aussi bien sur des machines massivement parallèles à mémoire partagée que sur des clusters d'ordinateurs hétérogènes à mémoire distribuée.

Le but de notre travail est d'étudier l'impact de l'utilisation des techniques du partitionnement de traitement et de données sur les données sémantiques. En particulier, nous nous intéresserons à la technologie Spark avec (ajouter le modèle big data qu'on va utiliser

dans l'implémentation, c'est une partie qu'on va ajouter après qu'on termine l'implémentation).

Notre mémoire est organisé comme suit :

Le premier chapitre : nous allons présenter deux parties. La première partie présente une introduction du concept du Big Data avec ses définitions et ses caractéristiques. Ensuite, nous présenterons les bases de données NoSQL, avec leurs caractéristiques, leurs avantages et inconvénients et leurs divers types. Dans la deuxième partie, nous proposant une introduction sur le concept des web sémantique. Pour ce faire, on présente les ressources du web sémantique avec les définitions des différents composants tels que : RDF, et schéma RDF, ontologie web, et donnée liée, et SPARQL, Ensuite, nous présenterons les différents types de partitionnement des donnes comme le hachage, le partitionnement vertical et horizontal, l'algorithme du clustering K-MEANS, et METIS. En plus, nous présenterons les outils de partitionnement du traitement, MPI, Hadoop, et Spark.

Le deuxième chapitre : ce chapitre est consacré à la présentation des systèmes de traitement de données sémantiques qui utilisent les outils de partitionnement du traitement (MPI, Hadoop, ou Spark). Pour ce faire, nous présenterons pour chaque système, l'objectif du système, le type de données traitées, le système de stockage utilisé, le partitionnement des donné utilisé, le type de system de partitionnement de traitement, et la contribution (**il faut mettre ces points suivants le contenu du chapitre à vous de vérifier**). En plus de cela nous allons présenter aussi les points faibles et les points fort du système. Et on terminera ce chapitre par une petite synthèse sur ces systèmes.

Le troisième chapitre : Nous présenterons la conception du travail, implémentation de l'application et les résultats obtenus (**Dans cette partie nous allons mettre le modèle et ça structure après implémentation)**

Le quatrième chapitre :

Enfin, Nous concluons par une conclusion générale. **(Dans cette partie nous allons mettre nos résultats après implémentation).**

Chapitre1 : les techniques du Big Data et le Web sémantique

1.1 Introduction

La création de RDF (Resource Description Framework) augmente à un rythme sans précédent, sous l'impulsion de la communauté du Web sémantique et des données ouvertes associées initiatives. D'une part, il y a l'explosion continue des données RDF ouvre la porte à de nouvelles innovations en matière de Big data et d'initiatives Web sémantique, et d'autre part, le traitement de ces données dépasse la mémoire et les ressources informatiques sur les serveurs primaires ce qui provoque de mauvaise performance due aux nombreux dépôts de stockage des données RDF et des interfaces de requête. De ce fait, les développeurs sont confrontés au défi d'exploiter les techniques proposées par le Big data afin d'être capables de traiter d'énormes quantités de données différentes dans un court laps de temps.

Dans ce chapitre, nous étudierons la signification et l'importance du big data, en présentant certaines techniques, et en identifiant les différents modèles de bases de données NoSQL qui existe actuellement sur le marché en mettant l'accent sur les solutions les plus populaires. Ensuite, nous présenterons les concepts majeurs du web sémantique avec une explication sur la façon dont ces données peuvent être stockées, comment les données sont divisées et comment ces données volumineuses sont traitées et interrogées.

1.2 Big Data

1.2.1 Définition du Big Data

Le terme de Big Data a été apparu avant quelques années englobant des nouvelles technologies pour résoudre les problèmes de la gestion traditionnelle des données. En 2001, Meta Group remonte le terme Big Data et en 2008 le cabinet d'études Gartner donne une définition pour le terme Big Data.

Le Big Data consiste à déterminer le stockage et l'analyse de volume de données complexes et massives. Elles ne peuvent être stocker par une base de données relationnelle traditionnel dans la mesure ou ces données sont beaucoup trop importante en termes de capacité. En plus ces données doivent se déplacer

rapidement et par conséquent, elles ne nécessitent pas les structures d'une base de données relationnelles.

Big data, littérairement les grosses données, est une expression anglophone utilisée pour désigner des ensembles de données qui deviennent tellement volumineux qu'elles en deviennent difficiles à travailler avec des outils classiques de gestion de base de données. Il s'agit donc d'un ensemble de technologies, d'architecture, d'outils et de procédures permettant à une organisation très rapidement de capter, traiter et analyser de larges quantités et contenus hétérogènes et changeants, et d'en extraire les informations pertinentes à un coût accessible [1].

1.2.2 Les caractéristiques

La caractérisation du Big Data est généralement faite selon le principe des « 3V », les V représentent respectivement le Volume, la Variété et la Vélocité [Référence].

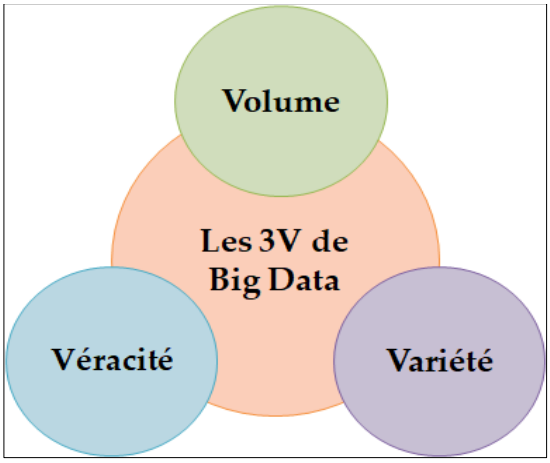


Figure 1 – Les 3V de Big Data.

1.2.2.1. Volume :

Le Big Data vous oblige à traiter d'énormes volumes de données non structurées et de faible densité. Il peut s'agir de données de valeur inconnue, comme les flux de données de Twitter, des flux de clics sur une page internet ou une application mobile, ou d'un appareil équipé d'un capteur. Pour certaines organisations, cela peut correspondre à des dizaines de téraoctets de données. Pour d'autre, il peut s'agir de centaines de pétaoctets

1.2.2.2. Vélocité :

Cela représente la vitesse à laquelle les données sont traitées ou reçues. Les données haute vitesse sont transmises directement à la mémoire au lieu d'être écrites sur le disque. Certains produits intelligents accessibles via Internet opèrent en temps réel ou quasi réel et nécessitent une évaluation et une action en temps réel.

1.2.2.3. Variété :

Les types de données traditionnels sont des données ayant une structure et elles sont stockées dans une base de données relationnelle, mais avec l'apparence du Big Data, les données ne sont pas nécessairement structurées telles que les données texte, audio et vidéo qui requièrent un prétraitement supplémentaire pour dégager du sens et prendre en charge les métadonnées [2].

1.2.3 Gestion de Big Data

La gestion de Big Data ou bien Big Data Management est une discipline dans laquelle les techniques, outils et plates-formes de gestion de données y comprend le stockage, le prétraitement, le traitement, et la sécurité peuvent être appliqués [3]. Le rôle de la gestion des données est assuré un haut niveau de qualité des données afin d'aider les entreprises à faire face à la quantité des données qui grandit [Référence].

1.2.3.1. Stockage des données

Afin de stocker les données on pétaoctets de façon distribuée, le stockage dans le Big data consiste trois opérations principales (Clusting, Réplication, Indexing) [Référence].

1.2.3.2. Prétraitement

Avant l'analyse de Big Data, il faut vérifier la qualité des données et réparer les données avant le traitement par l'application des étapes suivantes :le nettoyage des données, la transformation, l'intégration, la transmission, et la réduction, discrétisation.

1.2.3.3. **Traitement**

C’est l’aptitude de traiter un grand volume de données quel que soit le type ou bien la structure et l’emplacement de ces données, ce traitement peut être une classification ou une prédiction.

1.2.3.4. **Sécurité**

Pour sécuriser un grand volume des données, plusieurs algorithmes de sécurité sont apparus pour la confidentialité, intégrité, disponibilité **(il faut citer quelque algorithme**) [référence].

1.2.4 De nouveaux besoins en gestion de données

Avec l’apparition des applications Web (Google, Facebook, Twitter, Linkeln, Amazone etc.), le volume des données a augmenté considérablement ce qui nécessite une distribution de ces données et leur traitement sur de nouveaux serveurs qu’on appelle les « Data Center ». En plus ces données sont associées à des objets complexes et hétérogènes. Par conséquence, il est important d’utiliser de nouvelles approches de stockage et de gestion des données pour :

- Permettre une meilleure scalabilité dans des contextes fortement distribués.

- Permettre une gestion d’objets complexes et hétérogènes sans avoir à déclarer au préalable l’ensemble des champs représentant un objet.

- Améliorer les SGBD traditionnels par de nouveaux moteurs appelés NoSql [4].

1.2.5 Les bases de données NoSQL

Le NoSQL est un type de base de données, c'est une manière de stockage qui permet de récupérer des données de façon rapide, un peu comme une base de données relationnelle, sauf qu'il n'est pas basé sur des relations mathématiques entre les tables comme dans une base de données relationnelle traditionnelle. Le terme NoSQL est un terme apparu récemment dans le monde de traitement des données, il est très utilisé dans le stockage et le traitement de Big Data et les

applications web en temps réel, les bases de données NoSQL sont des bases de données extensible qui permettent le traitement distribué des données.[5]

1.2.5.1. Les types des bases de données NoSQL

Selon le modèle de données il existe 4 types de classification des bases de données NoSQL [6]:

1.2.5.1.1. Les bases de données orientées colonne

Dans ce type, les données sont stockées se forme par colonnes, l'orientation colonne permet l'ajoute des colonnes plus facilement aux tables ce qui permet aussi la compression par colonne. Parmi les bases de données orientées colonnes, il y a Hbase et Cassandra [6]. Les bases de données orientées colonne sont hybrides entre les bases de données relationnels et les clés-valeur [6], tel que :

- Les valeurs sont stockées dans des groupes de plusieurs colonnes mais ordonnés selon les colonnes.
- Les valeurs sont interrogées par correspondance de clé.
- Parmi les bases des données orientées colonne : Hbase (Facebook), Vertice. .[31]

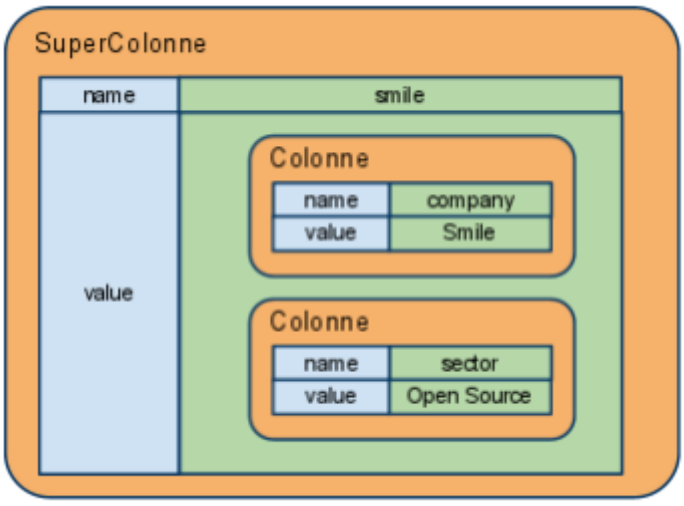


Figure 2 – Les bases de données orientées colonne.

Les données dans HBase sont stockées sous forme de HFiles, par colonnes, dans HDFS. Chaque HFile se charge de stocker des données correspondantes à une column family particulière.

1.2.5.1.2. **Les bases de données orientées document**

Le stockage des données dans ce type au format des fichiers JSON ou XML, et puisque les base de données NoSQL sont des bases non structurées, les fichiers ont des structures différentes. Parmi les bases de données orientées documents on a MongoDB et CouchDB [référence]. La base de données orientée documents est une évolution de la base de données clé-valeur, tel que :

- La clé n'est plus associée à une valeur sous forme de bloc binaire.
- Rendre la base de données consciente de la valeur qu'elle stocke.

Un document peut être définit comme un ensemble de couples propriété/valeur dont la seule contrainte est respectée le format de représentation.

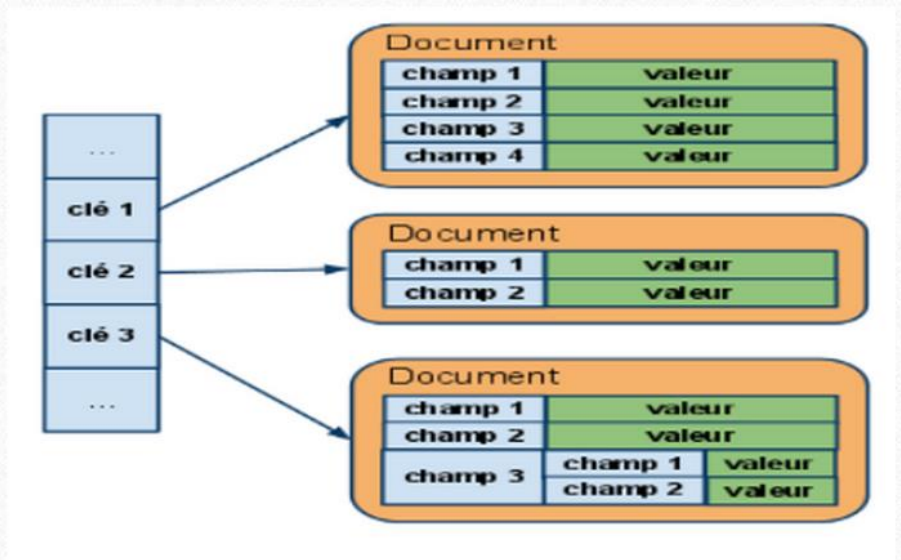


Figure 3 – Des bases des données orientées document.

Ce modèle ce base sur les documents de type XML. L'avantage est de pouvoir récupérer, via une seule clé, un ensemble d'informations structurées de manière hiérarchique. La même opération dans le monde relationnel impliquerait plusieurs jointures.

1.2.5.1.3. **Les bases de données orientées graphe**

Dans ce type les données sont stockées à la forme de (Node-Relationship) nœud-association, parmi les bases de données orientées graph Neo4j et FlockDB [**référence**]. Ce type de base de données permet de :

- Traiter des données fortement connectées.
- Gérer facilement un modèle complexe et flexible.

- Offrir des performances exceptionnelles pour les lectures locales par le parcours du graphe.

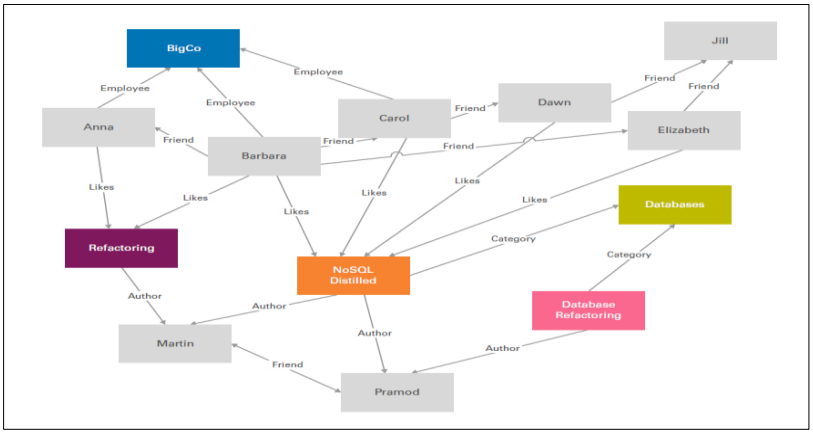


Figure 4 – Des bases des données orientées graph.

1.2.5.2. **Les bases de données clé-valeur:**

Dans ce type, les données sont stockées au format de clé a une valeur dont la clé n'est pas répétée et la valeur non structurée (image, vidéo, texte, nombre…), la puissance des base de données clé-valeur est la rapidité de la lecture [référence]. Ce type de base de données est caractérisé par :

- La plus simple et flexibles des BD NoSQL.
- Associe des clés à des valeurs.
- Solution aux limitations des BD relationnelles.
- Les valeurs sont identifiées et accédées via la clé, Ex: Dans les réseaux sociaux à partir d'un utilisateur (clé) on peut accéder aux ses amis (la valeur).
- Pas de schéma
- Parmi les bases de données clé-valeur : DynamoDB (amazon), Azure Table Storage (Microsoft), Riak, Redis

- La valeur stockée peut être: Entier, chaine de caractères, Document (XML, JSON, HTML etc.), image, vidéo etc.

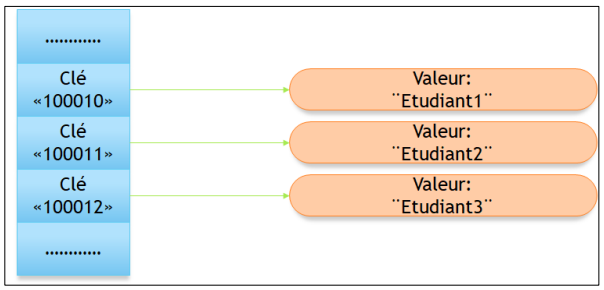


Figure 5 – Des bases des données clé-valeur.

1.2.5.3. **Avantages de NoSQL:**

- Supporte le semi-structuré (XML, JSon).
- Indexation de chaque colonne.
- Passage à l'échelle horizontal.
- Compression possible si les données d'une colonne proches.
- Possibilité de regrouper des colonnes en super-colonnes.

1.2.5.4. **Inconvénients de NoSQL:**

- Lecture de données complexes difficile.
- Difficulté de relier les données (distance, trajectoires, temps).
- Requêtes pré-définies.

Partie 02 : Les ressources sémantiques dans le web

1.3 Ressources sémantiques dans le web

1.3.1 Web sémantique

Le terme Web sémantique fait référence à la vision du W3C liée au réseau de données liées [référence]. Les technologies du Web sémantiques permettent aux gens de créer des systèmes de gestion de données sur le Web, de construire des vocabulaires, et d´écrire des règles pour le traitement de ces données. Il existe de nombreuses applications pratiques utilisant les technologies du Web sémantique. Aujourd'hui, la British Broadcasting Corporation (BBC) utilise le Web sémantique sur son service Web et fournit des ontologies pour le sport, l´éducation, la musique, etc. Certains grands moteurs de recherche tels que Google et Yahoo

utilisent également le Web sémantique pour améliorera les résultats de la recherche. Dans cette figure 6 nous présentons le Web sémantique comme une extension du Web actuel dans lequel l'information a un sens bien défini, permettant aux ordinateurs et aux gens de mieux travailler en coopération.

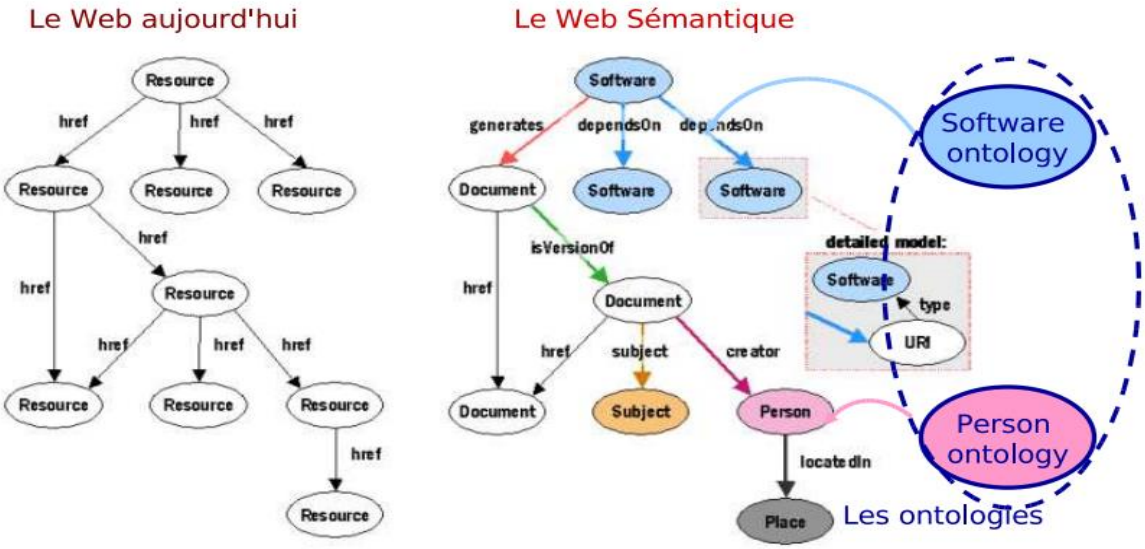


Figure 6 – web semantic et web actual [7]

la figure 7 est presenté des couches de cette pile technologique implique que les ´éléments d´écrits `a une couche donnée sont conformes aux normes définies aux couches inferieures. Chaque couche est moins générale que les couches ci-dessous.

- La couche la plus basse de cette pile fournit une solution d'identification globale pour les ressources trouvées sur le Web. Dans la Figure 7, ils sont appelés identificateurs de ressources uniformes/internationalisés (URI / IRIs).
- La deuxième couche prend en charge la définition d'une syntaxe basée sur XML, un métalangage basé sur la notion de tags.
- Dans la troisième couche, nous commençons notre voyage dans le Web Sémantique avec le langage RDF
- La quatrième couche permet d'étendre RDF pour décrire, en plus des faits, des métadonnées avec rfd schema .

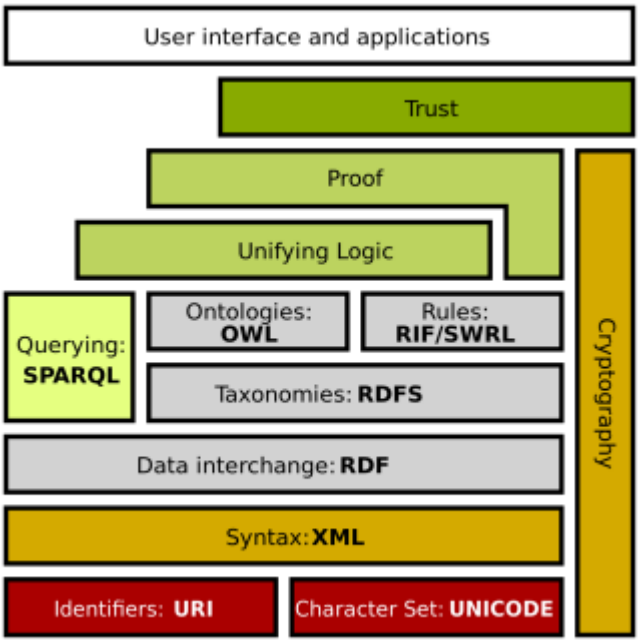


Figure 7 – Le ”layer cake” du Web sémantique [8]

1.3.2 RDF

RDF est le modèle standard recommandé par le W3C pour l'échange de données sur le Web. [30] Il aide les moteurs de recherche à comprendre la relation de ces ressources. La structure sous-jacente du modèle de données est simple et flexible. Toute expression en RDF est une collection de triplets [référence]. L'unité de base d'information dans RDF est donnée comme un triplet (s, p, o), composé d'un sujet (s), d'un prédicat (p), et un objet (o).

- Le sujet : La ressource sur laquelle une affirmation est faite. Seuls les URI et les nœuds vides sont autorisés à être utilisés comme sujet d'un triplet.
- Le prédicat : Un attribut d'une ressource ou d'une relation binaire qui relie cette ressource `a une autre. Seuls les URI sont valides pour être utilisés comme prédicat d'un triplet.
- L'objet : Généralement, il représente la valeur de l'attribut ou une autre ressource. Les objets valides sont des URI et des nœuds vides, mais aussi des chaînes de caractères. Ces chaînes, sont également appelées littéraux.

Chaque triplet représente un fait sur une chose qu'on veut décrire (c-`a-d., le sujet, qui est également appelé la ressource), sur une propriété spécifique (c.-`a-d., le prédicat), et avec une valeur donnée (c.-`a-d., l'objet).

Tableau 1: table de données RDF

sujet	prédicat	L'objet
resource:Bob	rdf:type	Person
resource:Bob	schema:interestedIn	resource:SemWeb
resource:Bob	schema:interestedIn	resource:DB
resource:Bob	schema:belongsTo	"USA"
resource:Alice	rdf:type	Person
resource:Alice	schema:interestedIn	resource:SemWeb
resource:SemWeb	rdf:type	Course
resource:SemWeb	schema:fieldOf	"Computer Science"
resource:DB	rdf:type	Course
resource:DB	schema:fieldOf	"Computer Science"

L'ensemble de données RDF représente les informations du web sémantique sous la forme d'un graphe orienté. Le graphe RDF est un ensemble fini de triplets RDF. RDF. montre un exemple de graphe RDF. Les nœuds représentent la ressource, les arêtes représentent la relation et les nœuds rectangulaires représentent les valeurs littérales.

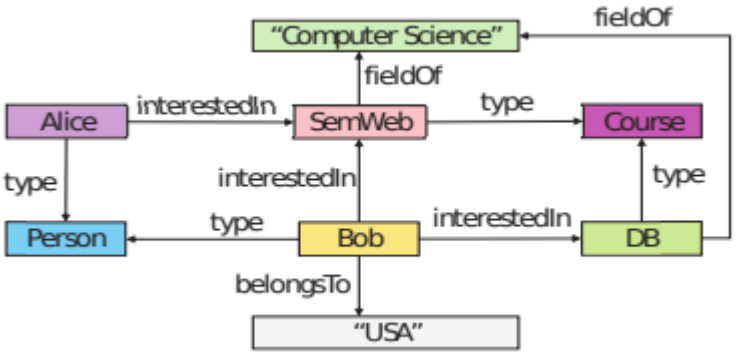


Figure 8– Représentations du graph RDF

1.3.3 Schéma RDF

Le RDF Schéma est un vocabulaire permettant de décrire des vocabulaires. C'est un des piliers du Web sémantique puisqu'il permet de bâtir des concepts, définis par rapport à d'autres concepts, ayant la particularité d'être partagés à travers le Web. Par extension, un schéma RDF désigne un vocabulaire définit avec la norme RDF Schéma, on parle aussi de "vocabulaire RDF". Par exemple, FOAF [11] possède un schéma RDF.

Le RDF Schéma permet de décrire des vocabulaires simples ; pour des vocabulaires plus expressifs, on se tournera vers OWL [référence] qui enrichit le modèle RDF Schéma.

RDF Schéma est doté du nombre minimum de constructeurs nécessaires à la définition d'un vocabulaire comme la classe, la sous classe pour définir une ressource, littéral. On trouve aussi la propriété s'applique à la classe, et la propriété est l'objet de la propriété.

Voici un exemple qui définit une classe « **herbivore »**, et qui instancie de cette classe une ressource **" http://www.ontoknowledge.org/Definedclass " ,** et la définie comme sous classe de la classe "**animal**":[10]

<rdfs:Class rdf:ID="herbivore**">**

<rdf:typerdf:resource=" http://www.ontoknowledge.org/#DefinedClass"/>

<rdfs:subClassOf rdf:resource="#animal**"/></rdfs:Class>**

Voici un autre exemple qui utilise les attributs **rdfs:domain** et **rdfs:range** pour désigner que la propriété "**frère de**" est une fonction qui a pour domaine les individus de la classe "**homme**" et pour image les individus de la classe "**humain**" **:**

<rdf:Property rdf:ID="frère_de**" > <rdfs:domain rdf:resource="#**homme**"/> <rdfs:range rdf:resource="#**humain**"/> </rdf:Property>** [12].

1.3.4 Ontologie Web

De nombreux langages informatiques sont apparus pour construire et manipuler des ontologies. Dans le but de mettre au point un langage standardisé, leW3C a créé le groupe Web Ont qui a mis en place le langage OWL [référence].

Le Web Ontology Language (OWL) est un vocabulaire XML basé sur RDF, et permet de spécifier ce qui peut être compris : il fournit un langage pour définir des ontologies Web structurées.

OWL définit donc une syntaxe RDF pour décrire et construire des vocabulaires pour créer des ontologies. Cependant, le langage OWL offre trois sous langages d'expression croissante conçus pour des communautés de développeurs et d'utilisateurs spécifiques qui sont :

- **OWL Lite :** Est le sous langage de OWL le plus simple, il est destiné à représenter des hiérarchies de concepts simples.

• **OWL DL :** Est plus complexe que le précédent, il est fondé sur la logique descriptive d'où son nom (*OWL Description Logics*). Il est adapté pour faire des raisonnements, et il garantit la complétude des raisonnements et leurs décidabilité.

• **OWL Full :** Est la version la plus complexe du OWL, destiné aux situations où il est <8important d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie. [12]

1.3.5 Linked data (données liées)

Le linked data est un ensemble de principes de conception pour le partage de données lisibles par les machines sur le Web pour une utilisation par les administrations publiques, les entreprises et les citoyens [référence].

Les 4 principes de conception des données liées (linked data) d'après Tim Berners Lee sont :

- Utiliser des URI (identificateurs de ressources uniformes) pour les noms des choses.
- Utiliser des URIs http, de sorte que les humains puissent consulter ces adresses.
- Fournir sur les URI (ressources) des informations utiles, en utilisant les standards du Web Sémantique (RDF *, SPARQL).
- Inclure des liens vers d'autres URIs afin qu'ils puissent découvrir plus de choses.

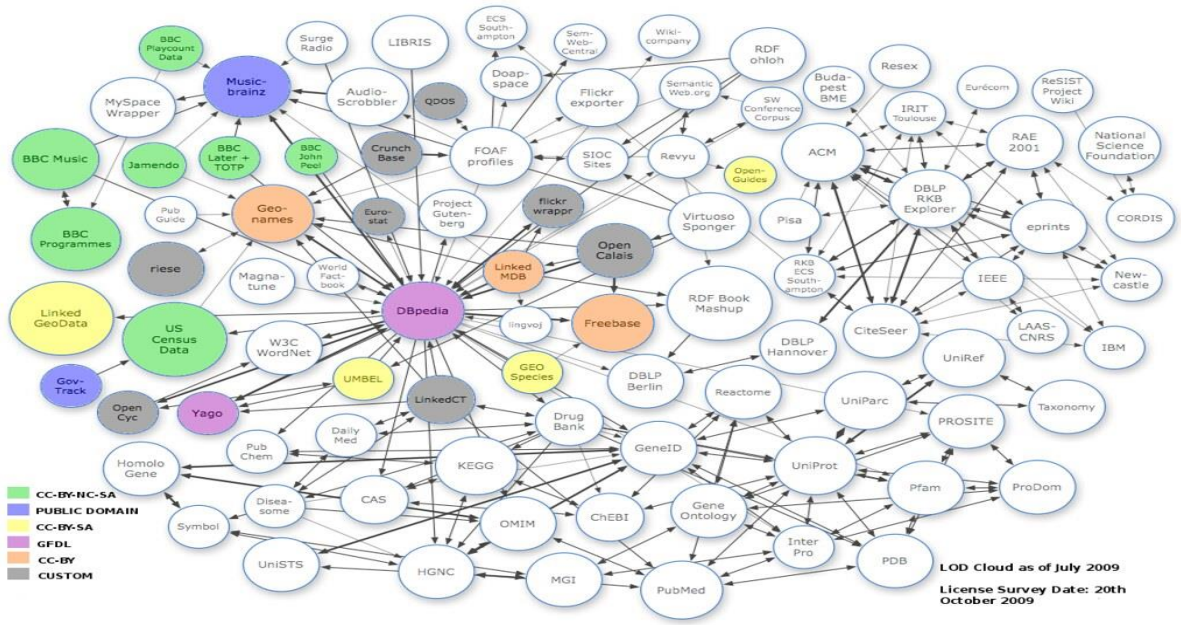


Figure 9 – Linked data [29]

1.3.6 SPARQL :

SPARQL est le langage de requête recommandé par le W3C pour RDF [30]. Sa syntaxe est similaire à la syntaxe d'une requête relationnelle. La requête SPARQL contient généralement plusieurs modèles triples. Un ensemble de motifs triples forme un motif graphique de base . Chaque tuple contient des variables représentées par? V, puis un groupe de requêtes correspond à un groupe de n sous-requêtes, chaque sous-requête contient un triplet, selon la variable pertinente du triplet qui peut interroger des informations.

Nous résumons le processus de requête comme suit: calculez les valeurs de liaison pour chaque TP, implémentez le résultat intermédiaire des jointures et construisez le résultat final de la requête. Par exemple, l'instruction de requête SPARQL est illustrée dans la figure 9.

Figure 10 – Instruction de requête SPARQL

```
SELECT ?name
WHERE {
   ?m <bornIn> ? city .   ?m <hasName> ?name .
   ?m <bornOnDate> ?bd .
   ? city <foundingYear> "1718" .
  }
```

Correspondant à la requête SPARQL ci-dessus, le graphe de requête SPARQL est illustré dans la figure 11.

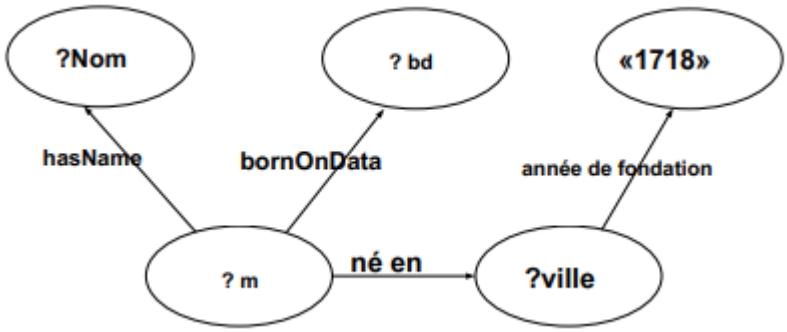


Figure 11 – Graphique de requête SPARQL

1.4 Méthode de partitionnement

1.4.1 Partitionnement des données

La disposition des données joue un rôle important pour les requêtes SPARQL dans un environnement distribué. Généralement pour une requête efficace, elle sera accompagnée de plusieurs index sur six triples permutations car l'évaluation de la requête se résume essentiellement à une série de jointures sur cette grande table.

1.4.1.1 Partitionnement vertical

Il s'agit de la distribution des données par colonne dans différentes partitions , cela implique la création de tables avec moins de colonnes avec l'utilisation de tables supplémentaires pour stocker les colonnes restantes. Dans le contexte de l'ensemble de données RDF, chaque triplet représente une ligne avec trois colonnes à savoir sujet, prédicat et objet. Par conséquent, la distribution par l'une de ces colonnes est considérée comme un partitionnement RDF vertical.

Le partitionnement vertical (VP) a été proposé dans [19]. Contrairement à PT(L'approche des tables de propriétés vise à réduire le nombre de jointure nécessaire évaluation de pour SPARQL Basic Graph Pattern) et TT (triple table pour enregistrer les données RDF dans un style relationnel), ,un VP stocke les données RDF sous forme de tableaux à deux colonnes. Sujet et objet nommés par la propriété. Le nombre de tables est égal au nombre de prédicats distincts utilisés dans l'ensemble de données RDF. Puisqu'il existe quatre prédicats distincts dans l'échantillon de données RDF présenté dans le tableau 1, les tableaux verticaux correspondants sont présentés dans le tableau 2.

Tableau 2: Partitionnement vertical de l'exemple de jeu de données RDF présenté dans le tableau 1. Les préfixes sont ignorés par souci de simplicité.

par belongsTo de predicate

sujet	L'objet
Bob	"USA"

Par interestedIn de predicate

sujet	L'objet
Bob	SemWeb
Bob	DB
Alice	SemWeb

Par type de predicate

sujet	L'objet
Bob	Person
Alice	Person
SemWeb	Course
DB	Course

par fieldOf de predicate

sujet	L'objet
SemWeb	"Computer Science"
DB	"Computer Science"

1.4.1.2 Partitionnement horizontal.

Il s'agit d'une distribution de données par ligne dans différentes partitions. Il est également appelé partitionnement de base de données. En RDF, chaque ligne d'un TT représente un triple, la distribution triple de l'ensemble de données complet est considérée comme un partitionnement horizontal.

Contrairement à PT et TT, un HP stocke les données RDF sous forme de tableaux à trois colonnes. Sujet et objet et prédicat. Le nombre de tables est égal au nombre de prédicats distincts utilisés dans l'ensemble de données RDF. Puisqu'il existe quatre prédicats distincts dans l'échantillon de données RDF présenté dans le tableau 1, les tableaux horizontaux correspondants sont présentés dans le tableau 3.

Tableau 3: partitionnement horizontal de l'échantillon de données RDF présenté dans

suje	prédicat	L'objet
Bob	**type**	**Person**
Bob	**interestedIn**	**SemWeb**
Bob	**interestedIn**	**DB**

sujet	prédicat	L'objet
Bob	**belongsTo**	**"USA"**
Alice	**type**	**Person**
Alice	interestedIn	SemWeb

sujet	prédicat	L'objet
DB	**fieldOf**	**Computer Science"**

sujet	prédicat	L'objet
SemWeb	**type**	**Course**
SemWeb	**fieldOf**	**Computer Science"**
DB	**type**	**Course**

1.4.1.3 Partitionnement basé sur le hachage:

Une fonction de hachage identifie la machine sur laquelle le fragment doit être stocké. La fonction de hachage fonctionne sur une clé de partage, qui est dans de nombreux cas l'objet

du triplet RDF; cela garantit que tous les triplets avec un sujet donné sont stockés sur la même machine. Cette approche permet la recherche directe vers les machines où les triplets ou quadruplets sont stockés avec une fonction de hachage connue. Il est reconnu comme efficace pour les requêtes en étoiles mais pas pour les requêtes complexes, car trop de communications réseau sont nécessaires. On a Trois techniques sont utilisées dans cette catégorie :

- **Sujet-haché:**

Dans cette technique, la fonction de hachage est appliquée sur le sujet du triplet et en fonction de la valeur de sortie, le sujet est assigné à une partition [16]. Cela provoque un déséquilibre entre les partitions. Ce déséquilibre est illustré dans l'exemple donné à la Figure 12. En utilisant cette technique, notre exemple d'ensemble de données est divisé de telle sorte que les triplets 3,10 et 11 sont attribués en partition rouge, le triple 7 est assigné en partition bleue et les triplets restants sont attribués en partition verte. C'est un déséquilibre évident de partitionnement.

Haché par prédicat :

Cette technique applique la fonction est appliquée sur le prédicat du triple et en fonction de la valeur de sortie, le prédicat est attribué à une partition. Cela provoque tous les triplets avec le même prédicat affecté à une partition. Dans l'exemple donné à la figure 12, il y a quatre prédicats distincts tandis que le nombre requis de partitions est 3. Ainsi, en utilisant la stratégie du premier arrivé pour servir, tous les triplets avec le prédicat p1 sont affectés à la première partition (rouge), les triplets p2 sont affectés à une seconde partition (verte), les triplets p3 sont affectés à la troisième partition (bleu) et les triplets p4 sont à nouveau affectés à la première partition.

- **Hierarchy-hashed URI:**

Cette technique de partitionnement est basée sur l'hypothèse que les IRI ont une hiérarchie de chemin et que ceux avec un même préfixe de hiérarchie sont souvent interrogés ensemble [16]. Les mêmes hypothèses fonctionnent dans cette technique, qui extrait la hiérarchie des chemins des IRI et ceux qui ont les mêmes sont affectés à la même partition. Dans l'exemple donné à la figure 12, montre que tous les triplets ayant une hiérarchie1 dans les sujets sont affectés à la partition verte, les triplets ayant une hiérarchie2 dans les sujets sont affectés à la partition rouge, et les triplets ayant une hiérarchie3 dans les sujets sont affectés à la partition bleue. Ce partitionnement peut ne pas produire les meilleurs temps d'exécution des requêtes car les hypothèses sous-jacentes sur les IRI peuvent ne pas être vraies en pratique [18].

```
@prefix hierarchy1: <http://first/r/> . @prefix hierarchy2: <http://second/r/> .
@prefix hierarchy3: <http://third/r/> . @prefix schema: <http://schema/> .
hierarchy1:s1          schema:p1          hierarchy2:s11 .  #Triple 1
hierarchy1:s1          schema:p2          hierarchy2:s2 .   #Triple 2
hierarchy2:s2          schema:p2          hierarchy2:s4 .   #Triple 3
hierarchy1:s1          schema:p3          hierarchy3:s3 .   #Triple 4
hierarchy3:s3          schema:p2          hierarchy1:s5 .   #Triple 5
hierarchy3:s3          schema:p3          hierarchy2:s13 .  #Triple 6
hierarchy2:s13         schema:p1          hierarchy2:s8 .   #Triple 7
hierarchy1:s1          schema:p4          hierarchy3:s9 .   #Triple 8
hierarchy3:s9          schema:p1          hierarchy2:s4 .   #Triple 9
hierarchy2:s4          schema:p4          hierarchy2:s13 .  #Triple 10
hierarchy2:s11         schema:p2          hierarchy1:s10 .  #Triple 11
```

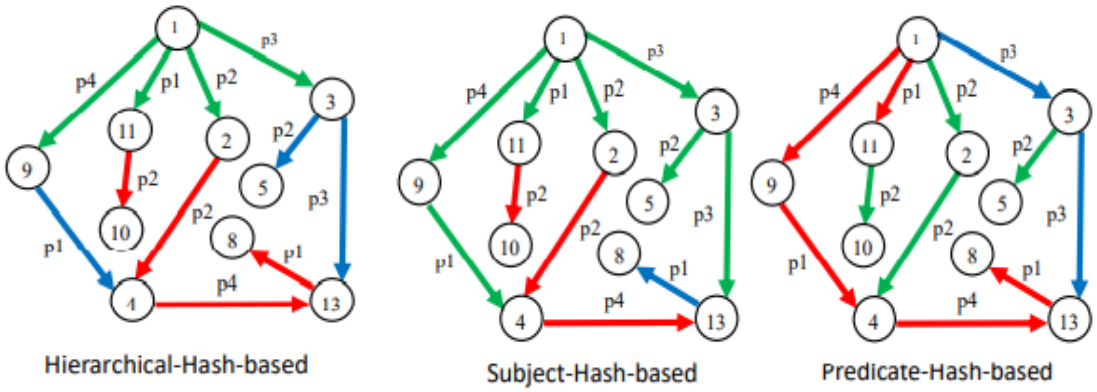


Figure 12 - Partitionnement d'un exemple de RDF en trois partitions en utilisant un partitionnement différent basé sur le hachage [17].

1.4.1.4 K-Means

K-means est l'un des algorithmes de clustering bien connus [référence de k-means]. L'algorithme k-means prend le paramètre d'entrée k et partitionne un ensemble de n objets en k clusters. La caractéristique des grappes résultantes est la forte similitude intracluster mais avec une faible similitude intercluster. Il se déroule comme suit. Tout d'abord, il choisit k d'objets au hasard comme centres de gravité, centre de chaque cluster. Pour chacun des autres objets, un objet est affecté au cluster auquel il est le plus similaire, en fonction de la distance entre l'objet et le centre de gravité. Il calcule ensuite la nouvelle moyenne de chaque cluster pour devenir un nouveau centre de gravité. Ce processus itère jusqu'à ce que la fonction critère converge ou qu'elle atteigne l'itération maximale [14].

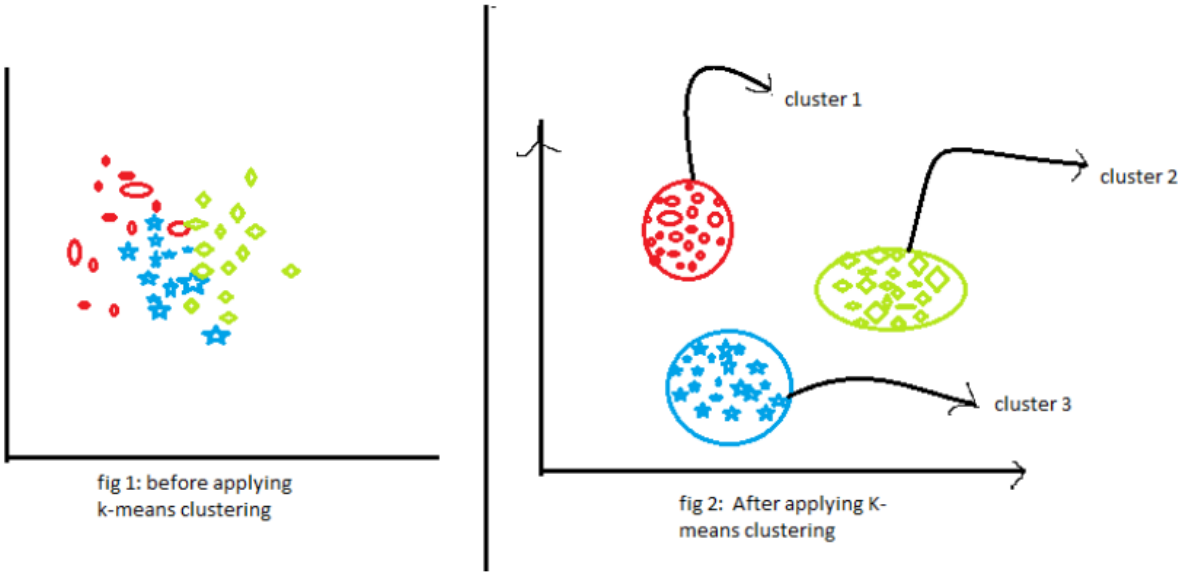


Figure 13: k-means clustering

La première figure(fig1) montre les données avant d'appliquer l'algorithme de clustering k-means. Ici, les trois catégories différentes sont gâchées. Lorsque vous verrez de telles données dans le monde réel, vous ne pourrez pas comprendre les différentes catégories.

La deuxième figure (fig 2). Cela montre les données après l'application de l'algorithme de clustering K-means. vous pouvez voir que les trois éléments différents sont classés en trois catégories différentes appelées clusters.

1.4.1.5 MÉTIS

METIS est un ensemble de programmes série permettant de partitionner des graphes et de partitionner des maillages d'éléments finis, et de produire des ordres de réduction de remplissage pour des matrices clairsemées. Les algorithmes implémentés dans METIS sont basés sur les schémas de partitionnement récursif à plusieurs niveaux, k-way et multi-contraintes développés dans notre laboratoire [référence].

Les principales caractéristiques de METIS sont les suivantes :

- Fournit des partitions de haute qualité !
- Les partitions produites par METIS sont constamment 10% à 50% meilleures que celles produites par les algorithmes de partitionnement spectral.
- C'est extrêmement rapide !
- Les graphes de plusieurs millions de sommets peuvent être partitionnés en 256 parties en quelques secondes sur les postes de travail et PC de la génération actuelle.
- Produit des commandes à faible remplissage !

METIS est capable de réduire les exigences de stockage et de calcul de la factorisation matricielle éparse, jusqu'à un ordre de grandeur. De plus, contrairement au degré minimum multiple, les arbres d'élimination produits par METIS sont adaptés à une factorisation directe parallèle. De plus, METIS est capable de calculer ces commandes très rapidement.

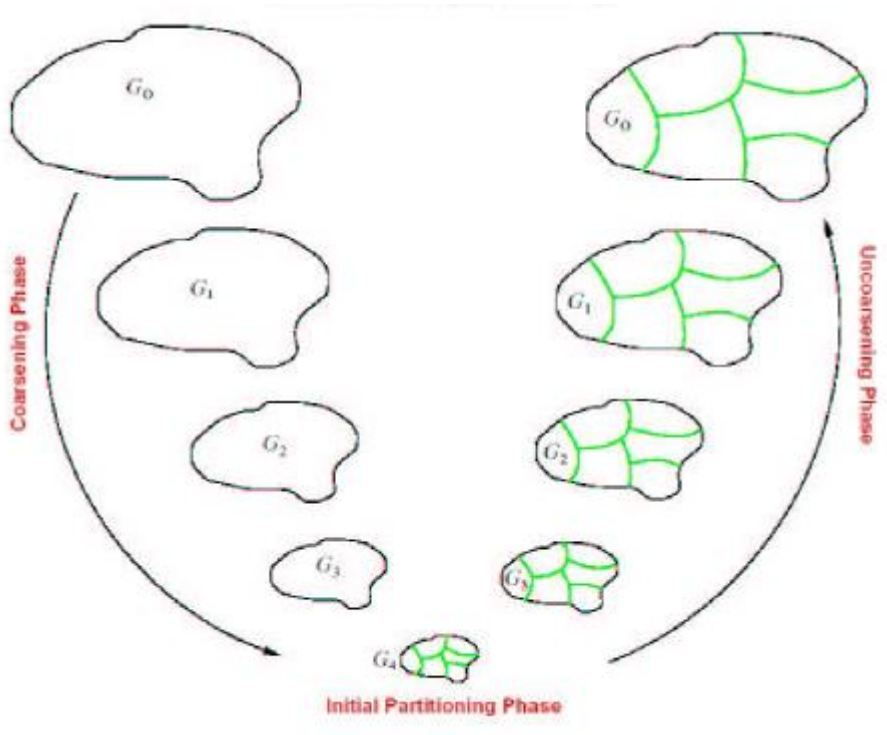


Figure 14 : Différentes phases de l'algorithme de partitionnement de METIS

L'idée derrière METIS est de créer successivement des graphes G1,G2,...,Gk de plus en plus petits à partir de G0, de partitionner Gk en très peu de temps, et de projeter la partition sur le G0, en affinant à chaque étape. La figure 13 illustre la méthode.

1.4.2 Partitionnement du traitement

Dans notre système, les requêtes sont exécutées dans les magasins RDF et / ou Hadoop. Puisque le traitement des requêtes est beaucoup plus e RD cace dans les RDFstores que dans Hadoop, nous poussons autant de traitement que possible dans les magasins RDF et laissons le reste à Hadoop ou mpi ou spark [9].

1.4.2.1 MPI

L'interface de passage de messages (MPI) [13] est une norme pour le calcul parallèle à grande échelle. MPI définit une interface qui spécifie la synchronisation et l'échange de données des messages entre les nœuds d'un cluster. Les appels de fonction MPI permettent au

programmeur de spécifier le contenu du message à échanger. Les fonctions sont explicitement énoncées et sont implémentées via des bibliothèques qui gèrent la communication inter-nœuds.

- **Objective MPI**
  - Concevoir une interface de programmation d'application.
  - Permettre des implémentations qui peuvent être utilisées dans un environnement hétérogène.
  - Autoriser les liaisons pratiques C et Fortran 77 pour l'interface.
  - Supposer une interface de communication fiable.
  - Définir une interface qui n'est pas trop différente de la pratique actuelle.
  - Définir une interface qui peut être mise en œuvre sur de nombreuses plates-formes de fournisseurs.
  - La sémantique de l'interface doit être indépendante de la langue.
  - L'interface doit être conçue pour permettre la sécurité des threads.

1.4.3 Types de partitionnent de donnée utilise

1.4.3.1MapReduce :

Le principe de MapReduce est simple, il s'agit de découper une tâche manipulant un gros volume de données en plusieurs tâches traitant chacune un sous-ensemble de ces données. MapReduce a deux étapes Map et Reduce. Dans Map les tâches sont donc dispatchées sur l'ensemble des nœuds. Chaque nœud traite un ensemble des données concernés. Dans Reduce, les résultats sont fusionnées pour former le résultat final du traitement. Nous pouvons aussi distinguer des autres étapes intermédiaires comme suivant:[26]

- **Map**

La fonction mapper est pour lire les données stockées et de les découper et générer un autre ensemble de données sous forme de tuples (paire de clé/valeur).

Dans Hadoop, cela se traduit par plusieurs exécutions de la fonction Map, sur les machines esclaves qui contient les données.

- **Combiner**

Une étape intermédiaire gérée directement par Hadoop, son rôle est de trier regrouper les paires avec des clés identiques. Elle sert donc d'une part à réduire le résultat à la sortie du mapper et d'autre part à faciliter la vie du Reduce.

- **Shuffle**

Une étape intermédiaire aussi permet de regrouper les tuples ayant la même clé dans une seul tuple mais contient la fusion des autres résultats.

- **Reduce**

La fonction Reduce prend la sortie de la phase Shuffle pour agréger les données. Chaque tâche de Reduce produit un fichier de sortie qui sera stocké dans le système de fichiers HDFS.

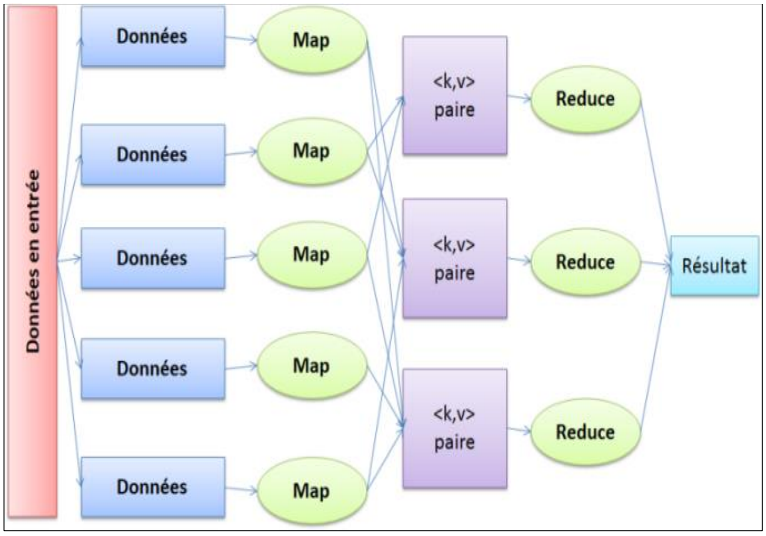


Figure -15 Exemple des étapes de MapReduce.

1.4.3.2 L'architecture de MapReduce.

Le MapReduce possède une architecture maître-esclave

- **Le maître MapReduce :** le JobTracker.
- **Les esclaves MapReduce :** les TaskTracker.

Le JobTracker

- Gérer l'ensemble des ressources du système.
- Recevoir les jobs des clients.
- Ordonnancer les différentes tâches des jobs.
- Assigner les tâches aux TaskTrackers.
- Réaffecter les tâches défaillantes.
- Sauvegarder des informations sur l'état d'avancement des jobs.

Le TaskTracker

- Exécute les tâches données par le JobTracker ;

– Exécution des tâches dans une autre JVM (Child) ;
– A une capacité en termes de nombres de tâches qu'il peut exécuter ;
– Heartbeat avec le JobTracker. [26]

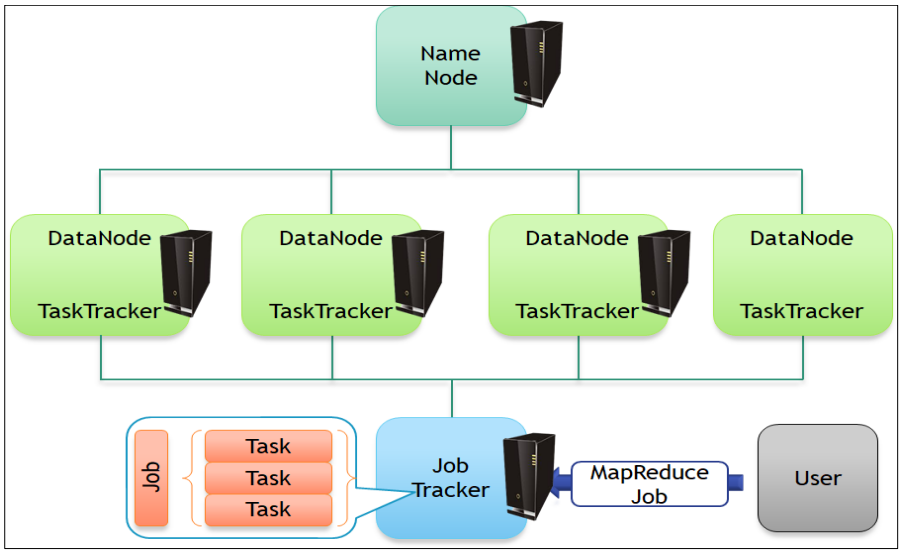


Figure -16 L'architecture de MapReduce.

1.4.4 Type de système de partitionnent utlisé

Hadoop

1.4.4.1 Présentation de Hadoop:

Hadoop est une plateforme open source de la fondation Apache, ayant une capacité de gérer des données volumineuses, qui sont structurées et non structurées. Elle est conçue pour trouver une solution aux problèmes liés à la volumétrie et la variété des données en les traitants sur des différents serveurs simultanément. Cette architecture va offrir une puissance, et un stockage importants, les données vont être par la suite répliquées et réparties sur les machines du cluster, grâce à un système de réplication de façon à garantir une très haute disponibilité des données en cas de défaillance d'un ou de plusieurs serveurs.[27].

1.4.4.2 Caractéristiques de Hadoop:

Hadoop a plusieurs caractéristiques, nous citons les suivants:

- **Résilience :**

Pour assurer la tolérance aux incidents de Hadoop. Quand un des nœuds tombe en panne les données stockées dans un nœud du cluster sont répliquées dans les autres nœuds du cluster qui disposent toujours d'une copie et sauvegardent des données.

- **Évolutivité** :

Hadoop fonctionne dans un environnement distribué. En cas de besoin de extensibilité la configuration peut être facilement étendue en installant d'autres serveurs, et la capacité de stockage peut ainsi atteindre plusieurs péta-octets.

- **Coût modéré** :

Hadoop étant un Framework open source n'exigeant aucune licence, les coûts de cette solution sont nettement inférieurs à ceux des bases de données relationnelles classiques. Par ailleurs, l'utilisation d'un matériel standard peu coûteux joue explique le coût modéré de cette solution.

- **Vitesse** :

Le système de fichiers distribué de Hadoop et les traitements concurrents de modèle MapReduce permettent d'exécuter les requêtes les plus complexes en temps très court.

- **Diversité des données**:

Le HDFS peut stocker différents formats de données structurées, non structurées (par exemple, des vidéos) ou semi-structurées (par exemple, des fichiers XML). Lors du stockage des données, il n'est pas nécessaire de valider celles-ci par rapport à un schéma prédéfini : les données peuvent être téléchargées sous n'importe quel format. Lors de leur récupération, les données sont analysées et utilisées en appliquant le ou les schémas requis. Cette souplesse permet de dériver des connaissances différentes à partir des mêmes données.[28]

1.4.4.3 L’architecture Hadoop:

Dans ce partie nous allons présente l’architecture de cluster dans Hadoop:

- **JobTracker:** Le responsable de lancer des tâches distribuées à les autres machines ou bien les esclaves, ensuite, il contrôle l’état de ces esclaves et fait agréger les résultats de calculs.

- **NameNode:** Est la machine qui fait la réplication et la répartition des données dans le cluster, possède toutes les informations des données et leurs emplacement, elle facilite donc l'accès des client aux fichiers stockés dans le cluster.

- **Secondary NameNode :** Durant le traitement d'une opération: Chaque couple de minutes, Secondary NameNode va copier les nouveaux informations stockés dans de NameNode, Normalement, le Secondary NameNode doit être assure par une autre machine physique autre que le master afin d'assurer la continuité du fonctionnement du cluster.

Pour les machines esclaves, nous pouvons leur attribuer ces différents rôles :

- **TaskTracker:** Permettre à l'esclave d'exécuter une tâche MapReduce sur les données qu'elle contient. JobTracker va envoyer les tâches à exécuter aux TaskTrackers. Nous pouvons remarquer qu'avec les solutions Big Data, les instructions sont amenées vers les données et non pas les données sont amenées aux instructions comme les programmes classiques.

- **DataNode:** C'est la machine qui contient un bloc des données, en effet, les données sont généralement partitionnées et répliquées sur les différents nœuds du cluster pour garantir la disponibilité des données. Cette machine doit périodiquement informer NameNode par un rapport d'état.[26]

1.4.5 Inconvénients de Système Hadoop

1.4.5.1 Problème avec les petits fichiers

Hadoop convient à un petit nombre de gros fichiers, mais lorsqu'il s'agit de l'application qui traite un grand nombre de petits fichiers, Hadoop échoue ici. Un petit fichier n'est rien d'autre qu'un fichier nettement plus petit que la taille de bloc de Hadoop, qui peut être de 128 Mo ou 256 Mo par défaut. Ce grand nombre de petits fichiers surchargent le Namenode car il stocke l'espace de noms pour le système et rend le fonctionnement d'Hadoop difficile.

1.4.5.2 Vulnérable par nature

- Hadoop est écrit en Java, un langage de programmation largement utilisé, il est donc facilement exploité par les cybercriminels, ce qui rend Hadoop vulnérable aux failles de sécurité.

1.4.5.3 Frais généraux de traitement

Dans Hadoop, les données sont lues à partir du disque et écrites sur le disque, ce qui rend les opérations de lecture / écriture très coûteuses lorsqu'il s'agit de téra et de pétaoctets de données. Hadoop ne peut pas effectuer de calculs en mémoire, ce qui entraîne une surcharge de traitement.

1.4.5.4 Prend en charge uniquement le traitement par lots

À la base, Hadoop dispose d'un moteur de traitement par lots qui n'est pas efficace dans le traitement de flux. Il ne peut pas produire de sortie en temps réel avec une faible latence. Il ne fonctionne que sur les données que nous collectons et stockons dans un fichier à l'avance avant le traitement.

1.4.5.4.1 Traitement itératif

Hadoop ne peut pas effectuer de traitement itératif par lui-même. L'apprentissage automatique ou le traitement itératif a un flux de données cyclique tandis que Hadoop a des données circulant dans une chaîne d'étapes où la sortie sur une étape devient l'entrée d'une autre étape.

1.4.5.4.2 Sécurité

Pour la sécurité, Hadoop utilise l'authentification Kerberos qui est difficile à gérer. Il manque un cryptage au niveau du stockage et du réseau, ce qui est un sujet de préoccupation majeur.

1.4.6 Présentation et concept de spark

Spark constitue la nouvelle brique In-Memory des distributions Hadoop. Grâce à la richesse de ses bibliothèques, Spark répond à vos besoins Big Data ou nécessitant des temps de réponse rapides ou encore de réaliser des calculs avancés. La solution Spark s'interface avec Yarn pour bénéficier des ressources allouées.

1.4.6.1 Type de produit et caractéristiques de spark

- **Spark travaille en mémoire**

La méthode de traitement des données utilisée par Spark est beaucoup plus rapide que le MapReduce de Hadoop. Spark analyse toutes les données en mémoire presque en temps réel

tandis que MapReduce va exécuter les opérations en plusieurs étapes. Spark est détenteur du record Daytona Graysort en 2014. [22]

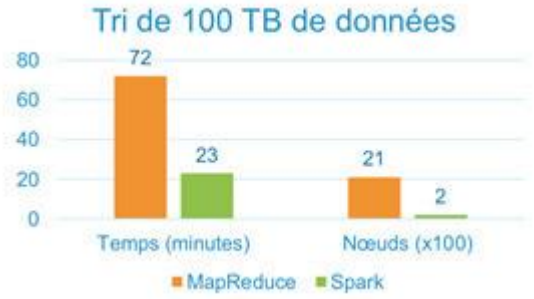


Figure -17 tri des données par MapReduce et Spark on années 2014

- **Spark intègre des outils d'analyse de données et de Data Science**

Spark Streaming permet d'accéder à des données en temps réel via les outils suivants : **Spark SQL** pour interroger et modifier les données comme avec des requêtes classiques, **Spark MLib** pour des modèles de Machine Learning et **GraphX** pour le calcul et la création de graphes. [22]

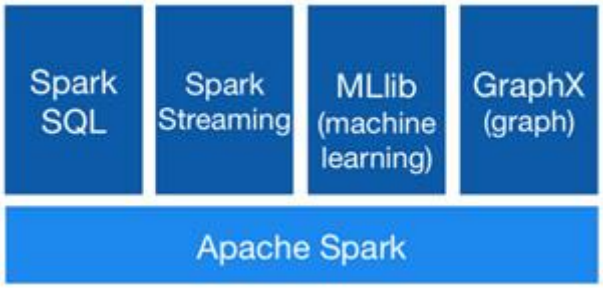


Figure -18 les outils de Spark

- **Spark s'intègre à de nombreux environnements**

Spark peut être exécuté en **mode standalone** ou sur un **cluster**. Sur Hadoop, il est possible d'utiliser les ressources avec Yarn et de stocker des fichiers sous HDFS. Sur Apache Mesos ou Kubernetes. Il est également possible de l'utiliser dans le cloud. Les principales sources de données utilisées sont Cassandra, HBase et Hive, mais il en existe des centaines d'autres. [22]

- **Spark est basé sur un langage rapide, Scala**

Scala est un langage objet et utilise la **programmation fonctionnelle**. Ce dernier étant régulièrement mis à jour, Spark bénéficie ainsi des améliorations et corrections apportées au

langage.Spark peut également être utilisé avec les langages Python, R et Java qui sont plus répandus.

Spark et le principe de "Lazy Evaluation" :Spark évalue les traitements si et seulement s'ils sont nécessaires lors de l'exécution. C'est le principe de **Lazy Evaluation**. Cela permet d'avoir un temps d'exécution plus rapide [22].

1.4.6.2 Ensembles de données distribués résilients

L'ensemble de données distribuées résilient (basé sur le document de recherche de Matei) ou RDD est le concept de base du framework Spark. Considérez RDD comme une table dans une base de données. Il peut contenir tout type de données. Spark stocke les données dans RDD sur différentes partitions. [22]

Ils aident à réorganiser les calculs et à optimiser le traitement des données.

Ils sont également tolérance aux pannes car un RDD sait recréer et recalculer les ensembles de données.

Les RDD sont immuables. Vous pouvez modifier un RDD avec une transformation mais la transformation vous renvoie un nouveau RDD alors que le RDD d'origine reste le même.

RDD prend en charge deux types d'opérations:

- Transformation
- action

- **Transformation:** les transformations ne renvoient pas une seule valeur, elles renvoient un nouveau RDD. Rien n'est évalué lorsque vous appelez une fonction de transformation, cela prend juste un RDD et retourne un nouveau RDD.

Certaines des fonctions de transformation sont **map, filter, flatMap, groupByKey, reductionByKey, aggregateByKey, pipe et coalesce.**

- **Action:** l'opération d'action évalue et renvoie une nouvelle valeur. Lorsqu'une fonction Action est appelée sur un objet RDD, toutes les requêtes de traitement de données sont calculées à ce moment et la valeur du résultat est renvoyée. [22]

Certaines des opérations Action sont réduire, collecter, compter, d'abord, prendre, countByKey et foreach. [22]

1.4.6.3 Écosystème Spark

Outre l'API Spark Core, il existe des bibliothèques supplémentaires qui font partie de l'écosystème Spark et fournissent des fonctionnalités supplémentaires dans les domaines d'analyse Big Data et d'apprentissage automatique. [23]

Ces bibliothèques comprennent:

- **Spark Streaming:**

Spark Streaming peut être utilisé pour traiter les données de streaming en temps réel. Ceci est basé sur un style de calcul et de traitement par micro-lots. Il utilise le DStream, qui est essentiellement une série de RDD, pour traiter les données en temps réel.

- **Spark SQL:**

Spark SQL offre la possibilité d'exposer les ensembles de données Spark via l'API JDBC et permet d'exécuter des requêtes de type SQL sur des données Spark à l'aide d'outils de BI et de visualisation traditionnels. Spark SQL permet aux utilisateurs d'ETL leurs données à partir de différents formats dans lesquels elles sont actuellement (comme JSON, Parquet, une base de données), de les transformer et de les exposer pour des requêtes ad hoc.

- **Spark MLlib:**

MLlib est la bibliothèque d'apprentissage automatique évolutive de Spark, composée d'algorithmes et d'utilitaires d'apprentissage courants, notamment la classification, la régression, le clustering, le filtrage collaboratif, la réduction de la dimensionnalité, ainsi que des primitives d'optimisation sous-jacentes.

- **Spark GraphX:**

GraphX est la nouvelle API Spark (alpha) pour les graphiques et le calcul parallèle de graphes. À un niveau élevé, GraphX étend le Spark RDD en introduisant le Resilient Distributed Property Graph: un multi-graphe dirigé avec des propriétés attachées à chaque sommet et arête. Pour prendre en charge le calcul de graphes, GraphX expose un ensemble d'opérateurs fondamentaux (par exemple, subgraph, joinVertices et aggregateMessages) ainsi qu'une variante optimisée de l'API Pregel. De plus, GraphX comprend une collection croissante d'algorithmes et de générateurs de graphes pour simplifier les tâches d'analyse de graphes.

En dehors de ces bibliothèques, il y en a d'autres comme BlinkDB et Tachyon.

1.4.5.3 Architecture Spark

L'architecture Spark comprend les trois composants principaux suivants:[23]

- Stockage de données
- API
- Cadre de gestion
- Examinons chacun de ces composants plus en détail.

Stockage de données: Spark utilise le système de fichiers HDFS à des fins de stockage de données. Il fonctionne avec n'importe quelle source de données compatible Hadoop, y compris HDFS, HBase, Cassandra, etc.

API: L'API permet aux développeurs d'applications de créer des applications basées sur Spark à l'aide d'une interface API standard. Spark fournit une API pour les langages de programmation Scala, Java et Python.

Vous trouverez ci-dessous les liens vers les sites Web de l'API Spark pour chacune de ces langues.

- API Scala
- Java
- Python

La gestion des ressources: Spark peut être déployé en tant que serveur autonome ou sur une infrastructure informatique distribuée comme Mesos ou YARN.

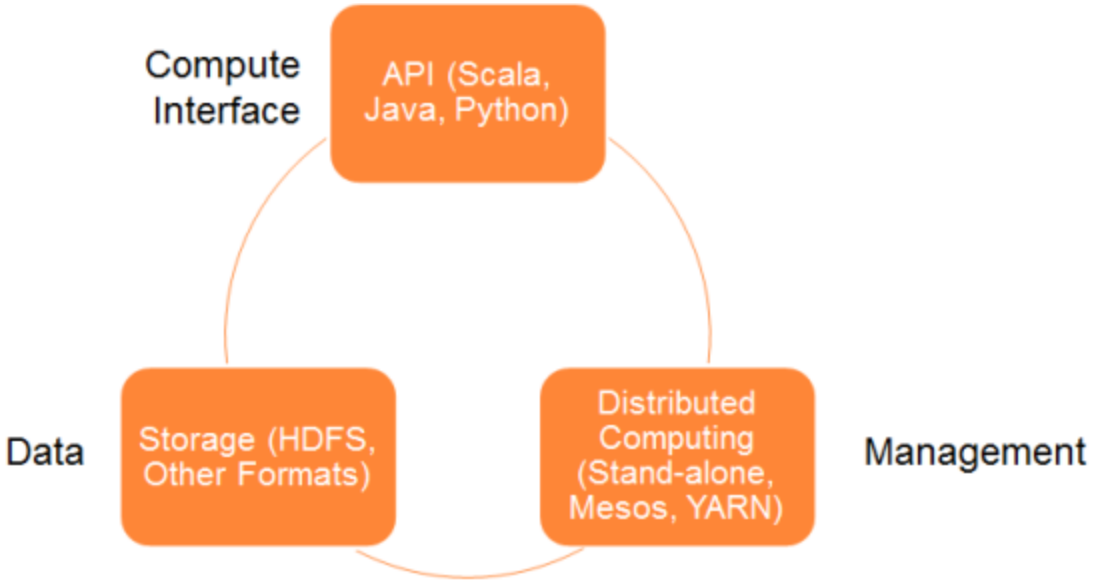


Figure -19 **Spark Architecture**

1.4.5.4 Avantages Spark

- **Calcul en mémoire dans Spark :**

Avec le traitement en mémoire, nous pouvons augmenter la vitesse de traitement. Ici, les données sont mises en cache, nous n'avons donc pas besoin d'extraire les données du disque, le temps est donc sauvegardé. Spark dispose d'un moteur d'exécution DAG qui facilite le calcul en mémoire et le flux de données acyclique se traduisant par une vitesse élevée.[21]

- **Rapidité de Traitement**

Avec Apache Spark, on atteint une vitesse de traitement de données élevée. Environ 100 fois plus rapide en mémoire et 10 fois plus rapide sur le disque. Ceci est rendu possible en réduisant le nombre de lecture-écriture sur le disque.

- **Dynamique dans la nature**

Il est facilement possible de développer une application parallèle, car Spark fournit 80 opérateurs de haut niveau.

- **Tolérance aux pannes dans l'étincelle**

Apache Spark offre une tolérance aux pannes via Spark abstraction-RDD. Les RDD Spark sont conçus pour gérer l'échec de tout nœud de travail du cluster. Ainsi, cela garantit une perte de données nulle.

- **Traitement de flux en temps réel**

Spark a une disposition pour le traitement de flux en temps réel. Auparavant, le problème avec Hadoop MapReduce était qu'il pouvait gérer et traiter des données déjà présentes, mais pas les données en temps réel. mais avec Spark Streaming, nous pouvons résoudre ce problème.

1.4.6.5 Inconvénients Spark

- **L'Absence de support pour le traitement en temps réel**

Dans Spark Streaming, le flux de données en direct qui arrive est divisé en lots de l'intervalle prédéfini, et chaque lot de données est traité comme une base de données distribuée résiliente Spark (RDD). Ensuite, ces RDD sont traités à l'aide d'opérations telles que mapper, réduire, joindre, etc. Le résultat de ces opérations est envoyé par lots. Ainsi, ce n'est pas un

traitement en temps réel, mais Spark gère presque en temps réel des données temps réel. Le traitement par micro-lots a lieu dans Spark Streaming.[21]

- **Problème avec les petits fichiers**

Si nous utilisons Spark avec Hadoop, nous rencontrons un problème pour la gestion de petits fichiers. HDFS fournit un nombre limité de gros fichiers plutôt qu'un grand nombre de petits fichiers. Un autre endroit où Spark Legs est derrière est que nous stockons les données compressées dans S3. Ce modèle est très bien sauf quand il y a beaucoup de petits fichiers compressés. La mission de Spark consiste maintenant à conserver ces fichiers sur le réseau et à les décompresser. Les fichiers compressés ne peuvent être décompressés que si l'intégralité du fichier se trouve sur un noyau. Il faudra donc beaucoup de temps pour graver leurs fichiers de décompression principaux en séquence. .[21]

Dans la RDD résultante, chaque fichier deviendra une partition; du coup, il y aura une grande quantité de petites partition dans une RDD. Maintenant, si nous voulons que notre traitement soit efficace, les RDD doivent être repartitionnés dans un format gérable. .[21]

Aucun système de gestion de fichiers

Apache Spark n'a pas son propre système de gestion de fichiers. Il repose donc sur une autre plate-forme comme Hadoop ou une autre plate-forme sur le cloud. C'est l'un des problèmes connus de Spark. La capacité en mémoire peut être un goulot d'étranglement lorsque nous voulons un traitement rentable des mégadonnées, car conserver des données en mémoire coûte assez cher. En effet, la consommation de mémoire est très élevée et elle n'est pas gérée de manière conviviale. Apache Spark nécessite beaucoup de RAM pour s'exécuter en mémoire, le coût de Spark est donc assez élevé. .[21]

- **Manque d'algorithmes**

Spark MLlib est à la ramasse en ce qui concerne un certain nombre d'algorithmes disponibles, tels que la distance de Tanimoto.

- **Optimisation manuelle**

Le travail Spark doit être optimisé manuellement et convient à des jeux de données spécifiques. Si nous voulons que la partition et le cache dans Spark soient corrects, ils doivent être contrôlés manuellement. .[21]

- **Traitement itératif**

Dans Spark, les données sont itérées par lots et chaque itération est planifiée et exécutée séparément.

- **Temps de Latence**

Apache Spark a une latence plus élevée par rapport à Apache Flink.

Spark ne prend pas en charge les critères de fenêtre basés sur les enregistrements. Il n'a que des critères de fenêtre temporels.

1.5 Conclusion :

Dans ce chapitre, nous avons présenté les principes des Big Data, ces caractéristiques, son fonctionnement ainsi que les différents domaines dans lesquels elles sont utilisées. On a aussi precensé les différents modèles de bases de données NoSQL qui existent actuellement dans le marché en accentuant sur les solutions les plus populaires. après nous avons présenté le web sémantique, puis son outil principal, qui sont les ontologies, et on a parlé du partitionnement de traitement, on a présenté l'outil Hadoop et Spark ses composants principaux tels que HDFS et MapReduce, l'architecture de cluster dans Hadoop et Spark. Ensuite nous avons montré la façon de distribuer les données (dans plusieurs machines (DataNode), les machines qui gèrent les métadonnées (NameNode), et la façon du traitement et de la gestion des données dans Hadoop et dans le RDD de Spark.

Chapitre2 : étude des systèmes de traitement de données RDF

2.1 Introduction

Aujourd'hui, de nombreuses recherches se concentrent sur l'utilisation des technologies Big Data pour gérer de gros volumes de données du Web sémantique[21,22]. l'utilisation de technologies Big Data telles que les systèmes de gestion de bases de données NoSQL garantit l'évolutivité et la haute disponibilité des données Web. RDF le Resource Description Framework [20] est un modèle de graphe dont l'unité d'information est le triplet, utilisé pour décrire tout type de ressource sur le Web. Il est à la base du Web sémantique. La syntaxe RDF standardise les descriptions pour permettre aux machines de trier et d'échanger plus efficacement les métadonnées propres à chaque ressource numérique (article, tableau, graphique, livre numérique, photo, animation, fichier son, vidéo, logiciel...). Un système d'information peut être perçu selon trois axes : les données, les traitements et les communications. Depuis cinquante ans, l'ensemble de ces axes fait l'objet de recherches intensives pour améliorer les temps de traitement, la gestion de l'espace physique, la recherche et la transmission de l'information. Depuis 20 ans, certains ont identifié un quatrième axe de représentation du système d'information qui est l'axe sémantique. Cet axe, indépendant des contraintes physiques de la gestion de l'information tente de résoudre les problèmes d'hétérogénéité sémantique On parle alors de graphe sémantique, d'ontologie et de triplestore.

Maintenant, nous passons en revue certains des systèmes RDF qui sont basés sur des bases de données NoSQL pour stocker des données RDF massives, notamment :SHARD ,TRIAD ,S2RDF ,HRDF3X ,SHARPE ,SYSTEM XU.La comparaison est basée sur certains critères de logiciel de base de données tels que : base de données NoSQL, modèle de base de données NoSQL, licence de base de données, structure d'index, interrogation, triples d'index, méthode d'optimisation de jointure, traduction SPARQL et temps d'exécution.

2.2 RDF Querying with SPARQL on Spark

Les objectifs:

1-La réduction de la taille des données d'entrée et donc des E / S en général, et la sauvegarde des opérations de jointure.

2-la réduction de la taille des entrées de données est beaucoup plus efficace que la sauvegarde des opérations de jointure.

Type de données traitées :

RDF

RDF-3X

Système de stockage utilisé :

HDFS

MapReduce

Type de partitionnement de données utilisées :

Partitionnement vertical.

METIS.

Type de système de partitionnement utilisé :

Spark

Contribution :

1-Nous définissons un nouveau schéma de partitionnement relationnel pour les données RDF appelé ExtVP qui peut réduire de manière significative la taille d'entrée d'une requête.

2-Nous fournissons en outre un compilateur de requêtes de SPARQL vers Spark SQL basé sur ExtVP qui utilise des statistiques de table pour sélectionner les tables avec la sélectivité la plus élevée pour une requête donnée.

3-Nous présentons une évaluation complète à l'aide de la récente suite de tests de diversité WatDiv SPARQL [3] où nous comparons S2RDF avec d'autres moteurs de requêtes SPARQL de pointe pour Hadoop et démontrons ses performances supérieures sur des charges de travail de requêtes très diverses.

Point faible du système :

La faiblesse de cette systeme est l'existence de prédicats à valeurs multiples dans RDF

2.3 Scaling Queries over Big RDF Graphs with Semantic Hash Partitioning

Les objectifs:

1- Créer un ensemble de partitions de hachage sémantique de sorte que le nombre de requêtes pouvant être évaluées par le traitement intra-partition soit augmenté et maximisé.

Type de données traitées :

RDF RDF-3X

Système de stockage utilisé :

HDFS

Type de partitionnement de données utilisées :

PARTITIONNEMENT DE HASH SEMANTIQUE

L'algorithme de partitionnement de hachage sémantique effectue le partitionnement des données en trois étapes principales:

1-Construction d'un ensemble de triples groupes qui sont des blocs de construction de base pour le partitionnement de hachage sémantique.

2-Regroupement des blocs de construction de base pour générer des partitions de hachage de base. Pour augmenter encore la localité d'accès des blocs de construction de base, nous développons également une technique d'optimisation spécifique à RDF qui applique l'URI.

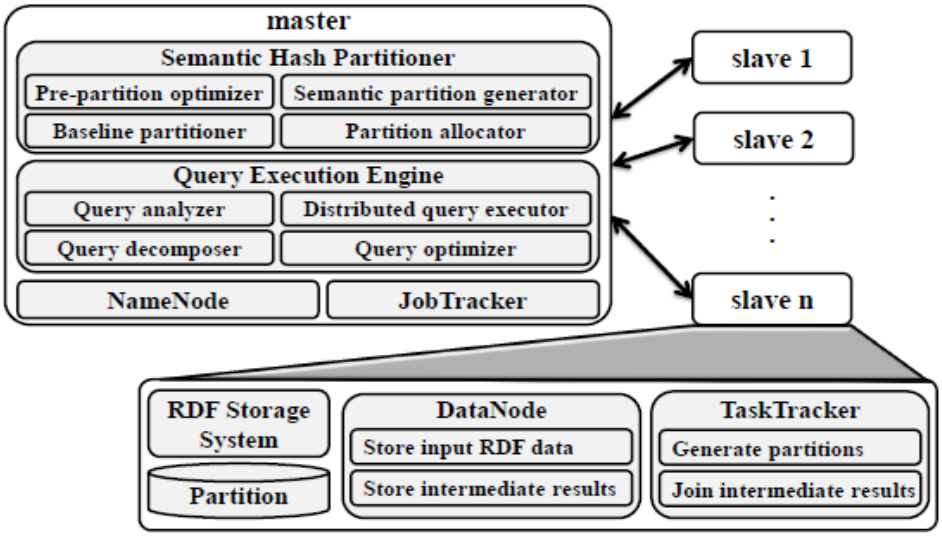
regroupement basé sur la hiérarchie pour fusionner les triples groupes dont les sommets d'ancrage partagent le même URI pre x avant de générer les partitions de hachage de ligne de base.

3-Génération de partitions de hachage sémantique à k-hop, qui étend chaque partition de hachage de base via une triple réplication contrôlée. Pour équilibrer davantage la quantité de triple réplication et l'efficacité du traitement des requêtes, nous développons également le triple ltre basé sur le type rdf pendant la triple réplication k-hop. Pour faciliter la lisibilité, nous décrivons d'abord les trois tâches principales, puis discutons des deux optimisations à la fin de cette section.

Type de système de partitionnement utilisé :

Hadoop

Structure de système :



Fiqure20 Architecture de système

La partition des graphes a été étudiée de manière approfondie dans plusieurs communautés pendant des décennies [10, 13]. Un partitionneur de graphe typique divise un graphe en partitions plus petites qui ont un minimum de connexions entre elles, comme adopté par METIS [13, 4] ou Chaco [10]. Divers efforts sur le partitionnement de graphe ont été consacrés au partitionnement d'un graphe en partitions de taille similaire afin que la charge de travail des serveurs hébergeant ces partitions soit mieux équilibrée. [11] promeut l'utilisation d'un partitionneur de graphes basé sur une coupe minimale pour distribuer de grandes données RDF sur un cluster de machines. Il montre expérimentalement que le partitionnement de graphe basé sur une coupe minimale surpasse l'approche de partitionnement par hachage simple.

Contribution :1-L'approche de partitionnement de hachage sémantique que nous proposons étend la méthode de partitionnement de hachage simple à travers des groupes triples basés sur la direction et des réplications triples basées sur la direction.

2- Nous générons des plans d'exécution de requêtes optimisés pour la localité qui sont plus efficaces que les systèmes de gestion de données RDF multi-nœuds populaires en minimisant efficacement le coût de communication inter-machine pour le traitement des requêtes.

3-Réduire davantage la taille de la partition et réduire le coût de communication entre les machines pendant le traitement des requêtes distribuées.

Point faible du système :les partitionneurs de graphes sont la surcharge élevée du chargement des données RDF dans le format de données des partitionneurs de graphes et la faible évolutivité vers de grands ensembles de données

artitionnement hachage est la mauvaise performance pour les requêtes complexes non étoiles.

2.4 Massive RDF Query Processing Efficiently in Spark Environment Based on Semantic Connection Set

Les objectifs:

1-Réduisez le temps nécessaire pour convertir les données brutes en données cibles tout en garantissant un schéma de partitionnement plus fin.

2-Réduisez la taille des données d'entrée pour éviter la surcharge de mémoire.

3-Récupération rapide des fichiers d'index associés.

Type de données traitées :

RDF

Système de stockage utilisé :

HDFS

Type de partitionnement de données utilisées :

1- Triple Matching Algorithm :

L'algorithme de triple correspondance est montré dans l'algorithme 1. En résumé, les lignes 1 à 8 représentent le cas où le prédicat est constant, et 9 à 10 représentent le cas où le prédicat est variable. Les lignes 2 à 6 considèrent le cas particulier où le prédicat est type. Les lignes 11 à 16 représentent les triplets qui sont filtrés par le sujet ou l'objet donné.

Optimisation des requêtes

Dans l'estimation des coûts, nous avons mentionné la réduction du coût de la jointure en réduisant la taille des résultats dans le processus et la communication des données. En raison de la requête BGP contient plusieurs triplets, nous les joignons en fonction de variables partagées. Mais dans le processus de requête, un ordre de connexion différent a une efficacité différente sur le résultat de la requête. Par conséquent, nous proposons une stratégie d'optimisation SCS pour générer l'ordre de jointure dans la requête RDF.

Algorithm 1: Triple Matching Algorithm

```
Input: tp:(s,p,o)
Output: IR
1  if p is constant then
2      if p is rdf:type then
3          if o is constant then
4              tq = spark.textFile(o.txt);
5          else
6              tq = spark.textFile(p.txt);
7      else
8          tq = spark.textFile(p.txt);
9  else
10     tq = spark.textFile(D.txt);
11 if s is constant then
12     tq = tq.Filtercase(s, o) => s.equals(constant);
13     size = tq.count;
14 if o is constant then
15     tq = tq.Filtercase(s, o) => o.equals(constant);
16     size = tq.count;
17 new IR(tq, size);
18 return IR
```

2- Semantic Connection Set :

L'ordre de jointure des sous-requêtes SPARQL a un impact significatif sur les performances des requêtes, c'est pourquoi la méthode d'optimisation de l'ensemble de connexions sémantiques (SCS) doit être créée. Le SCS contient plusieurs résultats intermédiaires (IR) obtenus après plusieurs correspondances TP, puis triés par ordre croissant en fonction de la taille de l'IR. La taille des IR dans l'ensemble initial est statistique dans la sous-section 6.2, puis les deux résultats intermédiaires plus petits qui contiennent des variables communes sont sélectionnés pour se joindre en premier, et les résultats générés sont ajoutés à l'ensemble. Supprimez les IR précédemment connectés et triez par taille. Répéter joignez les résultats intermédiaires jusqu'à ce qu'un seul résultat reste dans l'ensemble, qui est le résultat final de la requête SPARQL. Enfin, nous renvoyons les colonnes d'intérêt à l'utilisateur. Nous utilisons la méthode SCS pour générer un plan de requête optimisé afin d'améliorer les performances des requêtes RDF. Cette approche réduira la taille du résultat intermédiaire pour économiser les E / S et réduira le nombre total de comparaisons de connexion pour économiser le processeur. Comme le montre l'algorithme 2, la ligne 2 représente le plus petit IR de l'ensemble de la connexion sémantique.

De la ligne 3 à la ligne 6, indiquez que les résultats correspondants dans l'ensemble contenant la même variable et le plus petit IR sont extraits. Les lignes 8 à 10 représentent les deux ensembles de

résultats qui implémentent l'opération de jointure et la suppriment de l'ensemble, après quoi le résultat de la jointure est chargé dans l'ensemble de connexions.

Algorithm 2: SCS Algorithm

Input: List(IRs)
Output: Query Result

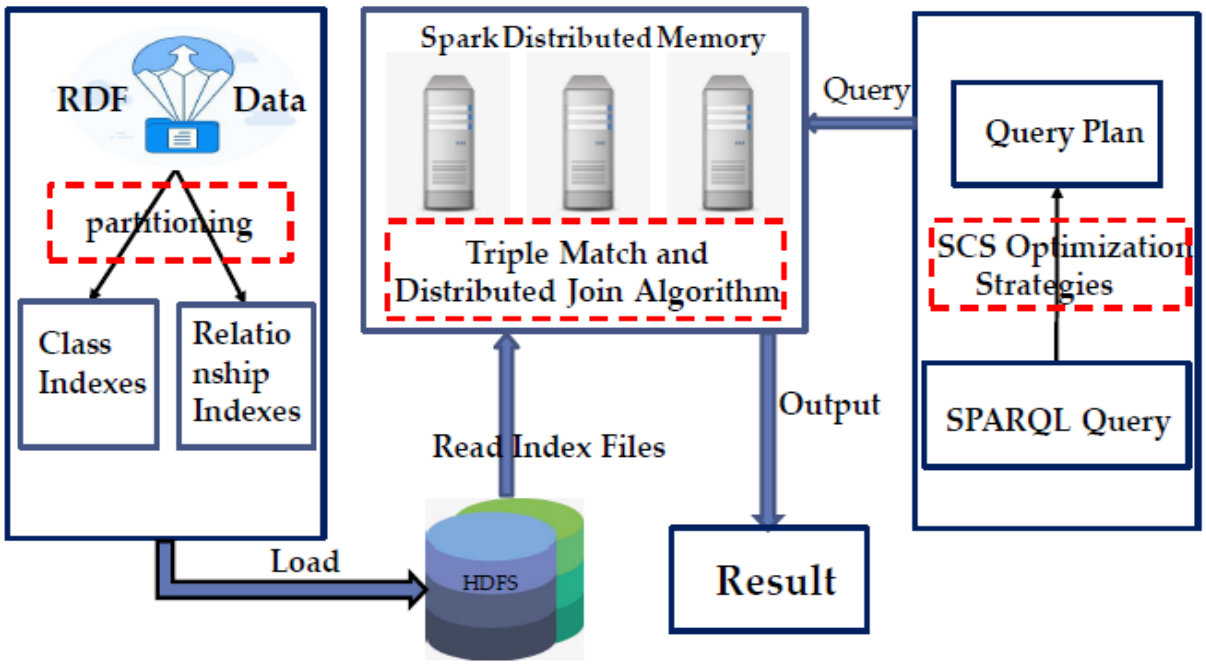
```
while list.size > 1 do
    IR1 = list.minby(.getSize);
    canJoin = list.filter(rdd =>
     rdd.ne(IR1)&&rdd.getVarSet.intersect(IR1.getVarSet).nonEmpty);
    if canjoin.isEmpty then
        print("can not join into one");
    IR2 = canjoin.minby(.getSize);
    broadcast(IR2);
    join = IR1.join(IR2);
    list.delete(IR1, IR2);
    list.add(join);
QueryResult = list.head.getTriple;
return Query Result
```

Type de système de partitionnement utilisé :

Spark

Structure de système :

Dans cette architecture système, le module de préparation de données est conçu pour convertir les données RDF sous forme de XML au format n-triples, diviser davantage les classes et les relations



basées sur une partition verticale, et générer des fichiers d'index relationnel et des fichiers d'index de classe. Le module de données persistantes est responsable du chargement des fichiers d'index divisés par les modules ci-dessus dans HDFS. Le module d'analyseur de requête est utilisé pour générer un plan de requête basé sur la stratégie d'optimisation SCS, y compris l'ordre de jointure des trois modèles et les informations de variable de diffusion. Sur la base des informations d'analyse, nous avons chargé les fichiers d'index correspondants de HDFS dans la mémoire distribuée Spark et les avons persistés. Le module de traitement distribué effectue une correspondance locale et une opération de jointure itérative selon le plan de requête et génère finalement le résultat de la requête.

Contribution :

1-Nous introduisons un schéma de stockage basé sur VP pour la gestion de données RDF massives en partitionnant davantage le prédicat rdf: type basé sur le patitionnement vertical (VP) [7].

2-L'ensemble de connexions sémantiques génère un plan de requête et utilise la méthode des variables de diffusion pour éviter de nombreux coûts de communication.

3-Nous effectuons une évaluation expérimentale en comparant ce système et le SPARQLGX de pointe actuel qui interroge les données RDF sur la plate-forme Spark.

4-Nous testons les performances des travaux actuels sur les ensembles de données LUBM [8] et les ensembles de données Watdiv [9] via des requêtes de référence standard.

Point faible du système :

Nous démontrons que le système considère principalement l'interrogation de données RDF à grande échelle de manière efficace dans l'environnement distribué. Le problème principal est considéré sur les deux points suivants:

- Partie stockage : comment réduire la surcharge de mémoire grâce à la partition et aux données d'indexation et atteindre un équilibre entre le prétraitement des données et l'indexation rapide. Différentes méthodes de stockage et d'accès affectent directement l'efficacité de la requête.
- Partie requête : comment réduire les coûts de traitement des requêtes SPARQL et les coûts de communication. Le traitement des requêtes SPARQL peut être transformé en problème de correspondance itérative et de jonction de sous-requêtes.

2.5 TRIADE a distributed shared-nothing RDF engine based on asynchronous message passing

• **Système centralisé fédéré**

- Extension du RDF-3X centralisé à l'environnement distribué

- Basé sur le passage de message asynchrone

• **Principales caractéristiques de TriAD**

- Construction d'un graphe récapitulatif

- Partitionnement de graphes avec METIS

- Le graphique récapitulatif définit la distribution des données

- Exécution de requêtes sur graphique récapitulatif (sur le site maître)

-élimine les partitions inutiles

- élagage de la partition

- Optimiseur de requêtes compatible avec la distribution

• **Optimisation des requêtes**

- Algorithme d'optimisation en deux étapes déterimining

-Meilleur ordre d'exploration pour le graphique récapitulatif

-Meilleur ordre de jointure pour le graphe de données RDF

Le but de triad

Exécutions de jointures parallèles et asynchrones. TriAD en principe suit une architecture maître-esclave classique.

Index RDF distribués avec élagage Join-Ahead Chaque liste de permutation SPO est la première

Presentation de Triad

-en combinaison avec un protocole asynchrone Message Passing [6]. Le moteur, baptisé TriAD (pour «Triple-Asynchronous-Distributed»), vise à combler l'écart entre les moteurs Hadoop actuels sans partage [8, 17, 31], sur le d'une part, et des stratégies d'exploration de graphes pures basées sur Message En passant [21, 30], d'autre part.

-TriAD est conçu pour atteindre parallélisme plus élevé et moins de surcharge de synchronisation pendant la requête exécutions que les moteurs Hadoop en ajoutant une couche supplémentaire de multithread pour des chemins entiers d'un plan de requête qui peuvent être exécutés en parallèle.

TriAD est le premier moteur RDF qui utilise des exécutions de jointure asynchrone (utilisant un protocole MPI personnalisé) qui sont couplé à une technique d'élagage à jointure légère pour le traitement distribué des requêtes SPARQL. Plus précisément, TriAD s'appuie sur les principes suivants.

-Le multi-thread et distribué L'exécution des jointures dans TriAD est facilitée par un protocole de passage de messages asynchrone qui nous permet d'exécuter plusieurs opérateurs de jointure le long d'un plan de requête de manière entièrement parallèle et asynchrone.

-Essayé par rapport aux benchmarks LUBM, BTC et WSDTS(dataset), il montre que TriAD surpasse constamment les moteurs RDF, signalé pour un serveur de milieu de gamme et une configuration Ethernet régulière.

-triad combine l'élagage par jointure via une nouvelle forme de résumé de graphe RDF avec un partitionnement horizontal

Partitionnement Des Données

partitionnent horizontalement

algorithme DP METIS , hashing-based partitioning

Traitememnt Des Données

Hexastore , RDF-3X, H-RDF-3X [8] et EAGRE

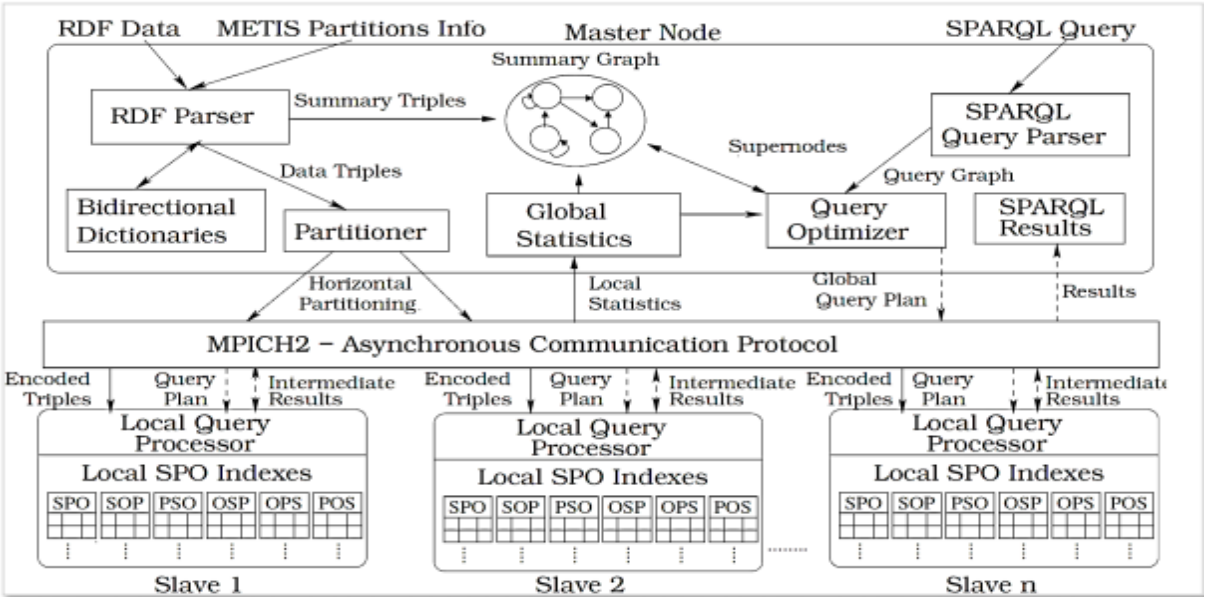
Mpi , Hadoop ,SIP ,MapReduce , C ++,Trinity.RDF,

Stockage System

RDF-3X ,hdfs

stockant des triplets directement dans des arbres B +

ARCHITECTURE DU SYSTÈME



Un aperçu de l'architecture du système TriAD est présenté dans Figure 2. TriAD ressemble à un maître-esclave typique, sans partage model, dans lequel chaque nœud de calcul gère sa propre zone de mémoire principale et stocke des partitions disjointes des structures d'index RDF.Un nœud de calcul désigné, le nœud maître, stocke toutes les métadonnées sur les faits RDF indexés et sert de point initial de

contact pour toutes les tâches d'indexation et de traitement des requêtes. Le reste les nœuds esclaves contiennent les structures d'index locales et échangent les résultats de la requête intermédiaire via un protocole de communication asynchrone directe entre eux. Toute communication est basée sur le message Passing Interface (**MPI**) utilisant l'API MPICH24.

Analyseur et partitionneur RDF. Ce composant s'occupe de l'analyse Fichiers RDF (fournis au format TTL / N3) et partitionnement de l'ensemble complet des triplets RDF entrants dans le graphe récapitulatif et le structures d'index local SPO (section 5).

Analyseur SPARQL. L'analyseur SPARQL est responsable du prétraitement des requêtes entrantes. Les requêtes sont transformées en une représentation graphique,avant que l'optimiseur de requêtes ne compile la requête en une join plan qui est ensuite envoyé à tous les esclaves (section 6).

Graphique récapitulatif. Le traitement initial d'un motif de requête SPARQL par rapport au graphe récapitulatif facilite l'élagage par jointure (en utilisant une stratégie de récapitulation basée sur la localité) au niveau des esclaves en déplaçant les partitions de graphe qui ne contiennent pas de triplets correspondants pour le modèle de graphe indiqué par la requête (section 5.1).

Dictionnaires bidirectionnels. L'étape d'analyse RDF implique la création de mappages bidirectionnels pour les triplets RDF entrants dans l'ordre pour convertir rapidement des chaînes en identifiants entiers et vice versa. Pour s'adapter à notre schéma de partitionnement de graphe pour le graphe récapitulatif, le dictionnaire de transfert conserve la combinaison d'identifiant de partition (un nœud dans le graphe récapitulatif) et l'identifiant du composant (section 5.2).

Statistiques mondiales. Une fois l'indexation terminée, le maître reçoit le les statistiques d'index local des esclaves et les fusionne dans ses propres statistiques globales à utiliser pour l'optimisation des requêtes (section 5.5).

Optimiseur de requêtes. Dans une deuxième étape de traitement, l'optimiseur de requêtes (Section 6.3) construit le plan de requête global basé sur les statistiques globales, la localité des index SPO et les réestimations de cardinalité après avoir traité la requête par rapport au graphique récapitulatif.

Nœuds esclaves

Index SPO locaux. A chaque esclave, un indexeur local reçoit le triplets au format id et construit ses structures d'index locales pour chacun des les six permutations SPO principales (sections 5.3 et 5.4).

Processeurs de requêtes locaux. Chaque esclave reçoit une copie du global plan de requête du maître, après quoi les processeurs de requête locaux initialiser leurs propres instances des opérateurs de requête physiques dans le plan. Les esclaves commencent simultanément à exécuter le même plan mais analyser différentes partitions de leurs index SPO locaux. Avec le plan global, le maître communique également l'élagage à tête jointe informations du graphe récapitulatif aux esclaves (section 6.4).

Aventage inconveniant :

tout en gagnant un facteur de plus de trois par rapport aux moteurs distribués actuellement les plus rapides.

TriAD à pas moins de neuf moteurs RDF, SGBD et Hadoop de pointe.

Problème 1: Exécutions de jointure synchrone ou asynchrone.

Problème 2: Exploration de graphes vs jointures relationnelles.

2.6 SHARD (Scalable, High-Performance, Robust and Distributed)

Objective et But:

-Matériel de base UNIQUEMENT

-Hautement évolutif

-Informatique décentralisée

- Résistant aux pannes de nœuds Considérations de conception pour SPARQL

- Requêtes complexes

- Réponses aux requêtes volumineuses

- Optimisation des requêtes distribuées

A partir de l'article SHARD a discuté de l'utilisation du logiciel MapReduce cadre pour relever le défi de la construction de haute performance

Dans cet article, nous nous concentrons sur les aspects de conception inhérents à l'utilisation du Framework logiciel MapReduce pour construire de manière hautement parallèle, des systèmes de gestion de l'information performants et évolutifs.

Nous suggérons également des Framework logiciels alternatifs pour les conception et développement de systèmes de gestion de l'information hautement évolutifs.

SHARD conserve les données du graphique sous forme de triplets RDF et répond aux requêtes sur ces données dans le langage de requête SPARQL.

SHARD a traite des dataset lorsqu'il est déployé sur un cloud Amazon AWS avec des nombre des nœuds de calcul

conception de shard permet une recherche efficace dans les données pour trouver des correspondances qui satisfont les requêtes.

Cette **conception** de shard est facilement répartie sur de nombreux nœuds de calcul pour un fonctionnement hautement parallèle et hautement évolutif.

pour améliorer encore les performances et l'applicabilité du point de vue de la conception et du cadre logiciel.

a développé première version de SHARD a utilise la version Cloudera de l'implémentation Hadoop que nous avons déployée sur un environnement cloud Amazon EC2 de 20 nœuds de calcul XL [1] exécutant RedHat Linux et Cloudera Hadoop.

La version de SHARD pour l'évaluation prend en charge la fonctionnalité de requête SPARQL de base (sans prise en charge des préfixes, des clauses facultatives ou de l'ordre des résultats) sur des données RDF complètes.

Bien que possible à implémenter, la version déployée de SHARD n'effectue aucune manipulation / réorganisation des requêtes / etc… normalement effectuée pour augmenter les performances des terminaux SPARQL dans les triple-stores matures.

System de Stockage Utilisé :

DAMLDB: la machine individuelle industrielle

Hdfs , triple store (forma rdf)

Type e Partitionnement des Données :

Triple store ,N3 , requet sparql

Autre article a définition de triple store

• Triple-store distribution

– Hash horizontal partitioning – Locality

-based horizontal partitioning – N-hop guarantee horizontal partitioning

– Semantic hash partitioning – Semantic-aware partitioning

Type de System de Partitionnement

MAPREDUCE , HADOOP , cloud Amazon EC2, Java(jena 2) ,sesame2

Version Cloudera de l'implémentation Hadoop

Point faible de systeme

Hadoop fournissent peu de support natif pour accéder aux données stockées dans les fichiers HDFS. Inversement, DAMLDB propose des optimisations d'indexation spéciales pour les requêtes simples, qui ne sont pas encore implémentées dans SHARD (hadoop vs BDMLDB)

De plus, la version déployée de SHARD ne profite pas encore de la mise en cache des requêtes rendue possible par nos choix de conception

Les inconvénients de l'implémentation Hadoop du framework MapReduce incluent que seuls les programmes Java peuvent être utilisés nativement pour des applications plus complexes, il est difficile d'exécuter du code Java sur des nœuds de calcul nécessitant une personnalisation d'exécution, NameNode crée un goulot d'étranglement pour l'accès HDFS et les échecs de NameNode pouvez être catastrophique.

le contexte problématique qui anime la conception de notre système d'information est le besoin de systèmes d'information à l'échelle du Web.(Par exemple, l'une des avancées singulières au cours des dernières années dans le domaine du Web sémantique a été l'explosion des données graphiques disponibles dans des formats sémantiques.) Malheureusement, les technologies de traitement de données du Web sémantique, sont conçus pour être déployés sur une seule (ou un petit nombre de) machine (s)

Contribution

meilleure liaison des données. Au lieu d'avoir à stocker des listes de données dans des fichiers plats dans une construction de type HDFS, un cadre logiciel pourrait fournir une construction de données liées native qui associe des éléments de données avec des pointeurs vers des données associées. Cette structure de données liées permettrait un traitement plus rapide des requêtes localisées sans nécessiter une recherche exhaustive de l'ensemble de données à chaque requête de requête.

est également peu coûteux car il peut fonctionner sur du matériel de base

ARCHITECTURE DU SYSTÈME

Nous utilisons notre contexte SHARD de données graphiques et de requêtes SPARQL pour discuter concrètement d'une conception de ce traitement itératif de requêtes à l'aide du framework logiciel MapReduce. Un aperçu schématique de ce processus de liaison de requête

itérative pour les données du graphe et le contexte SPARQL peut être vu dans la figure 1. Ce schéma se compose de plusieurs opérations MapReduce

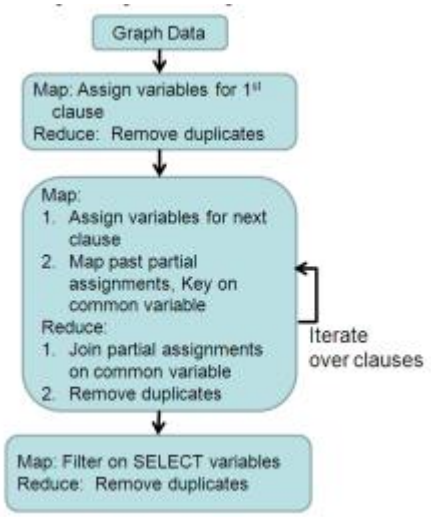


Figure 1 A Schematic Overview of the Iterative Algorithm to process SPARQL queries with Triple Data.

Explication :

La première étape MapReduce de mappage mappe les données triples à une liste de liaisons de variables qui satisfont la première clause de la requête. La clé de l'étape Map est la liste des liaisons de variables. Le réduire step supprime les résultats en double et les enregistre sur le disque avec les liaisons de variables comme clé.

Les étapes intermédiaires exécutent une opération MapReduce sur les données triples et les variables précédemment liées qui ont été enregistrées sur le disque.

La valeur de étape ième de mappage intermédiaire identifie toutes les variables dans les données triples est les liaisons d'autres variables non vues auparavant dans les clauses de requête, le cas échéant.

La ième étape de réduction exécute une opération de jointure sur les résultats intermédiaires de l'étape de mappage en itérant sur toutes les paires de résultats de la clause précédente et de la nouvelle clause avec la même affectation de clé.

Cette itération de ième étape de réduction map-reduction-join se poursuit jusqu'à ce que toutes les clauses soient traitées et que des variables soient affectées qui satisfont les clauses de la requête.

SHARD est conçu pour enregistrer les résultats intermédiaires du traitement des requêtes afin d'accélérer le traitement des requêtes ultérieures similaire

L'étape finale de MapReduce consiste à filtrer les affectations de variables liées pour satisfaire la clause SELECT de la requête SPARQL.

En generale En particulier, l'étape de mappage filtre chacune des liaisons et l'étape de réduction supprime les doublons où la valeur de clé pour mapper et réduire sont les variables liées dans la clause SELECT

CONCLUSION

Aujourd'hui, de nombreuses recherches se concentrent sur l'utilisation des technologies Big Data pour gérer de gros volumes de données du Web sémantique[21,22]. L'utilisation de technologies Big Data telles que les systèmes de gestion de bases de données NoSQL garantit l'évolutivité et la haute disponibilité des données Web. Nous avons présenté ce chapitre avec une comparaison et une évaluation de magasins de données RDF distribués basés sur des systèmes NoSQL. La comparaison révélée La comparaison a d'abord révélé que les systèmes NoSQL sont une très bonne solution pour la gestion de gros volumes de données RDF, et d'autre part cette variété de systèmes donne d'autres avantages techniques au niveau applicatif et diploïde de ces systèmes.

Chapitre 3 conception et implimentation

3.1 Introduction:

Dans ce chapitre nous allons présenter la solution au problème posé. Nous présentons l'architecture d'un système de sélection des candidats pour continuer leurs études de master dans les spécialités compatibles avec leurs compétences, à partir des données existants déjà dans l'université. Après nous allons expliquer de manière détaillée l'installation de tous les outils utilisés et les étapes de la réalisation de ce projet.

3.2. Présentation d'UML et des diagrammes:

3.2.1 Présentation « UML » :

3.2.1.1UML « UnifiedMethod Langage » : UML est un support de communication performant ou un outil qui permet de dessiner et facilite la représentation et la compréhension de solutions objet .Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation de solutions. L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions [01].

3.2.1.2 Méthode « UP » : UP ou « processus unifié » est un processus de développement logiciel itératif centré sur l'architecteur, piloté par des cas d'utilisation et orienté vers la diminution des risques. C'est un patron de processus pouvant être adaptée à une large classe de systèmes logiciels, à différents domaines d'application, différents types d'entreprises , différents niveaux de compétences et à différentes tailles d'entreprise.[02]

3.2.2 Présentation des diagrammes utilisés:

3.2.2.1 Le diagramme de cas d'utilisation:

C'est une manière spécifique d'utiliser un système. Les acteurs sont à l'extérieur du système ; ils modélisent tout ce qui interagit avec lui .Un cas d'utilisation réalise un service de bout en bout ,avec un déclenchement ,un déroulement et une fin pour l'acteur qui l'initie.[04]

Les acteurs de système:

Dans ce système on a deux acteurs: utilisateur normal consulte la résultat individuel de sélection afin de faire la sélection par le deuxième acteur qu'il a le rôle d'administrateur qu'il peut charger les données dans le système et lancer la sélection.

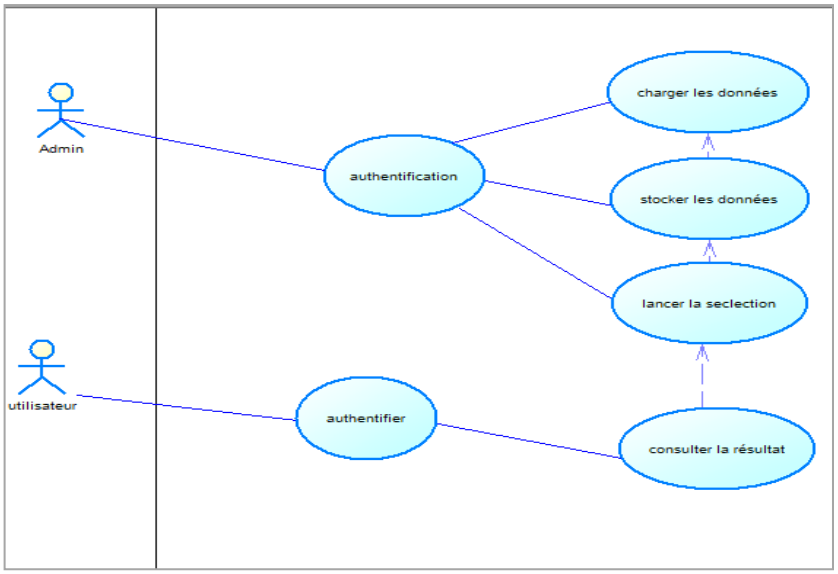


Figure 21 Diagramme de cas d'utilisation.

3.2.2.2 Le Diagramme de séquence :

sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation [06]

Scénario d'utilisateur:

Dans ce partie, l'utilisateur authentifie au système par un nom d'utilisateur et une mot de passe valide, le système vérifie si l'utilisateur est enregistré dans la base de données si l'utilisateur n'a pas un compte dans le système il peut s'inscrire, après l'authentification l'utilisateur peut consulter le système pour tester si il est accepté dans la sélection.

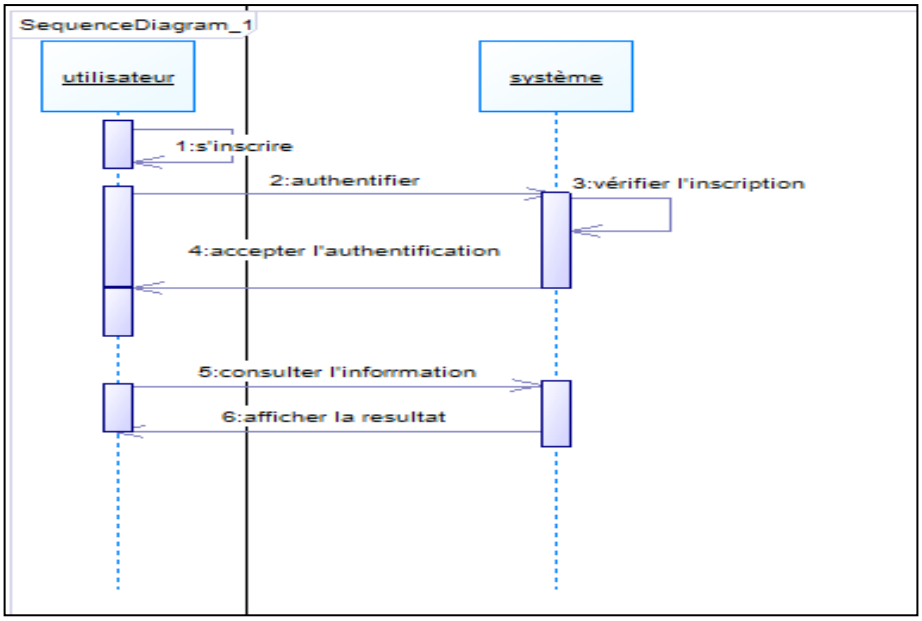


Figure 22.Le scénario d'utilisateur.

Scenario d'administrateurs:

- ✓ Administrateur connecte au système par l'authentification expliqué ci-dessus.
- ✓ L'administrateurs peut après l'authentification charger les données dans le système, ce dernier fait l'extraction et la distribution de ces données.
- ✓ Si les données est mise en place l'administrateur lance la sélection sur les données stockées.
- ✓ Le système retourne la résultat de sélection à l'administrateur.

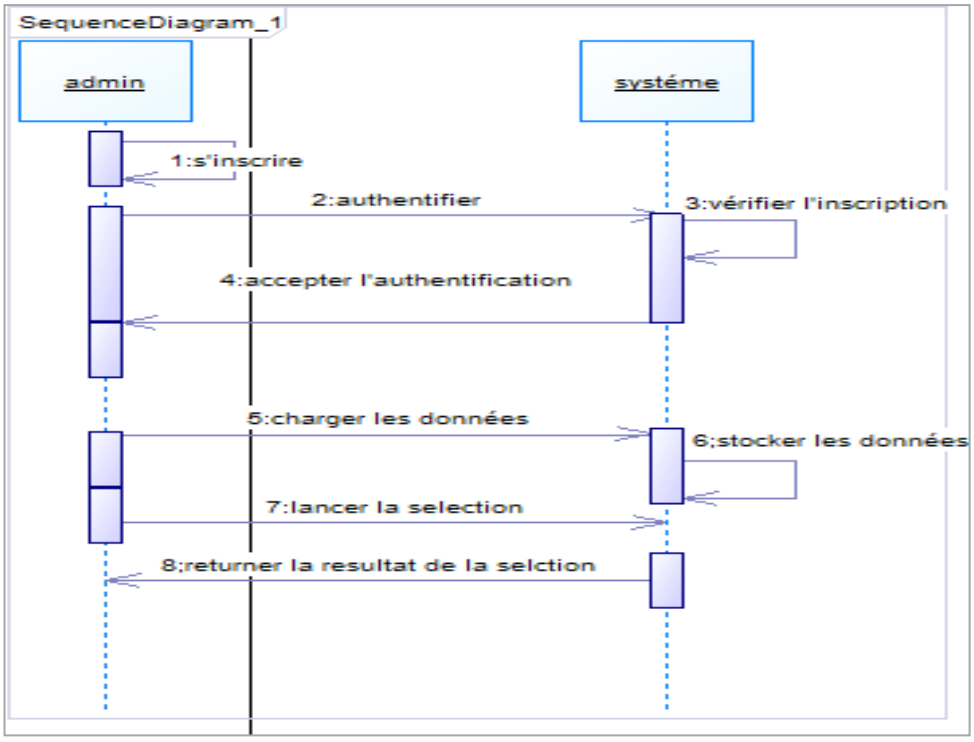


Figure 23. Le scénario d'administrateur.

III.3.1 L'architecture De Systeme

D'abord notre système est un système de sélection des candidats dans le Big Data, qui fonctionne généralement selon les principes indiqués dans les illustrations suivantes.

On a proposé une architecture qui permet :

3.3.1.1 Architecture de rdf par sparql sans pyspark :

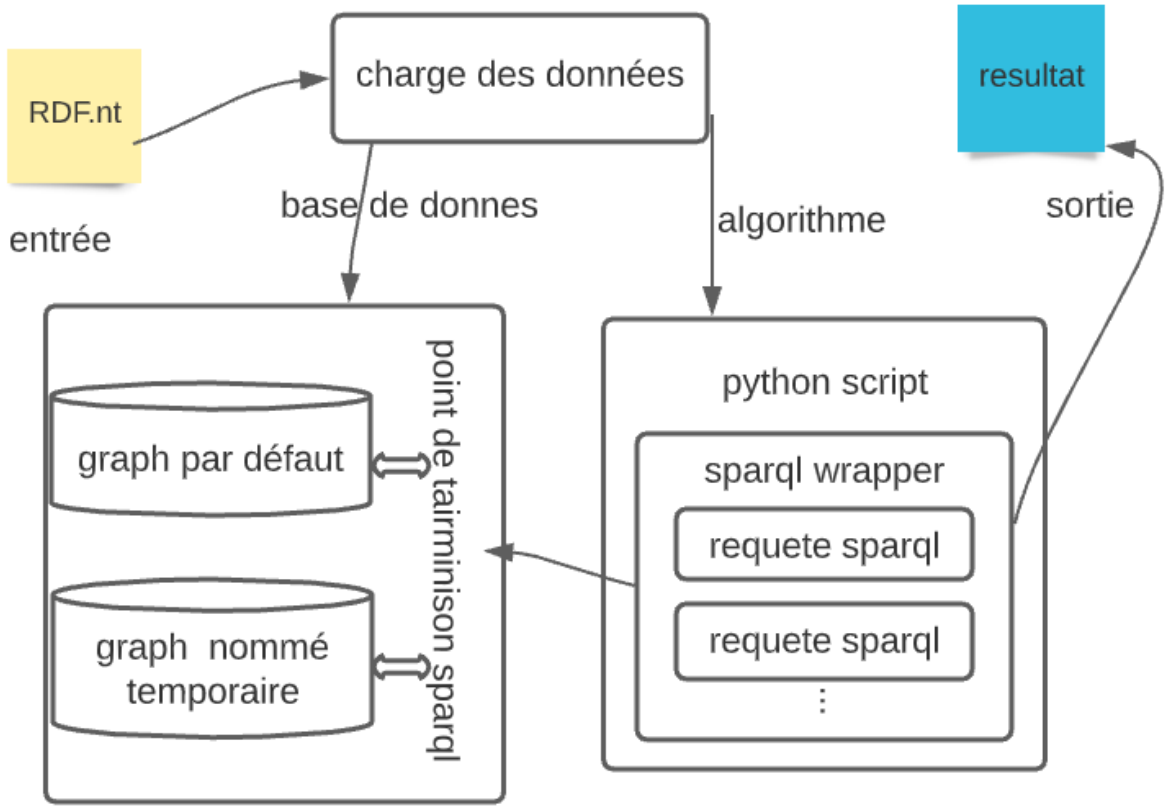


Figure 24 .Architecture de rdf par sparql sans pyspark

1. Conception de logiciels La figure 2 montre la conception du programme conceptuel pour la mise en œuvre des algorithmes d'exploration de graphes sur le trio RDF.

La première tâche de l'analyse graphique consiste à charger les données dans les systèmes.

1.Administrateur extrait sur les fonctions triplestores pour charger différents formats de séquence RDF (par exemple, N-triple, Turtle, RDF/XML, etc.).

Les ensembles de données RDF n'ont pas besoin d'une étape de transformation des données

Le graphique RDF d'intérêt est d'abord affiché dans un graphique triangulaire comme graphique par défaut.

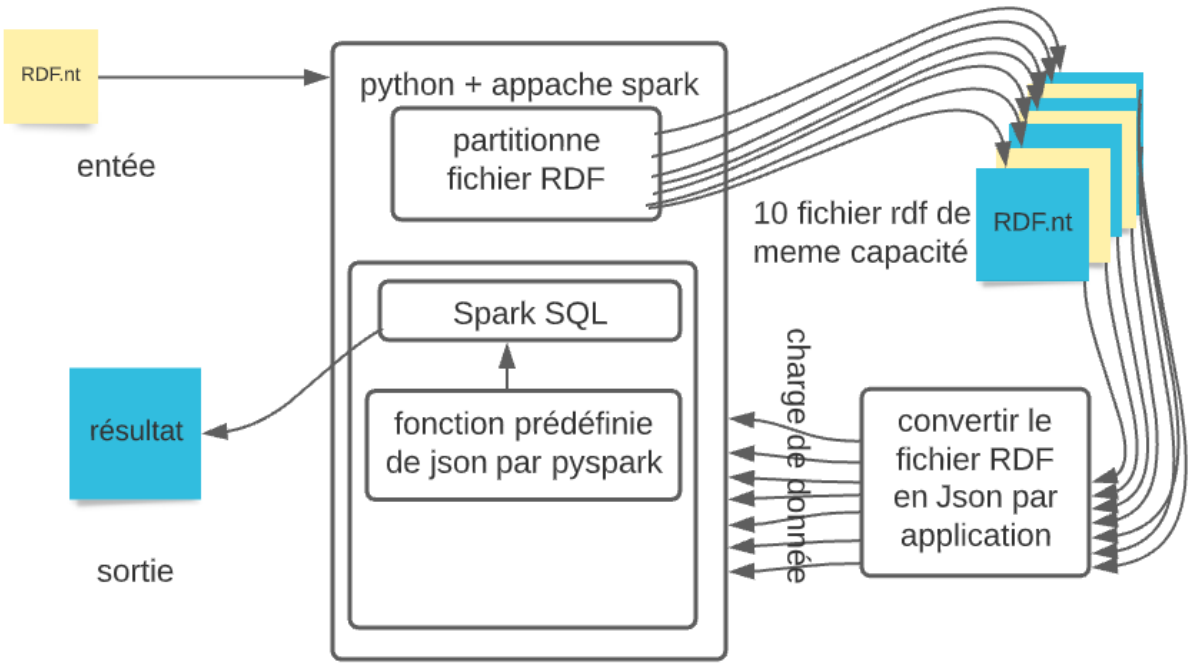
Nous analysons ensuite l'algorithme d'intérêt en une série d'étapes de traitement compatibles Sparql.

Comme le montre la figure 2, notre approche utilise à la fois Python et SPARQL pour traiter les données du graphique. Les requêtes SPARQL permettent le traitement à l'intérieur du triplestore, tandis que les scripts Python interagissent avec le points de terminaison SPARQL et contrôlent le flux logique de l'algorithme exprimé sous forme de séquence Demandes d'étincelles. Nous avons utilisé la bibliothèque SPARQLWrapper 3 pour les interactions entre un script Python et triplestore

Les Problèmes qui Résoudre :

1) **Interprétation des nœuds** : nous traitons les sujets et les objets avec des URI comme des nœuds, mais nous n'interprétons pas les littéraux comme des nœuds. En effet, les littéraux ne sont pas identifiés par leurs valeurs et, dans la plupart des cas, ils ne représentent pas des entités. Notez qu'un littéral est toujours connecté à un seul URI, ils n'affectent donc pas beaucoup le résultat de la plupart des algorithmes qui exploitent les connexions de nœuds.

2) **Interprétation des arêtes :** Les directions des arêtes doivent également être soigneusement prises en compte. Un jeu de données RDF forme un graphe étiqueté orienté. Ainsi, il est naturel que chaque triplet <s> <p> <o> soit interprété comme une arête <s> → <o>, où s, p et o sont des URI. Cependant, dans la pratique, nous devons parfois aussi considérer un ensemble de données RDF comme un graphe non orienté. Pour ce faire, il existe deux approches. Une approche consiste à stocker en plus un <o> <p> <s> supplémentaire pour chaque <s> <p> <o> d'origine, et ils sont interprétés comme un bord non dirigé <s> - <o>. Cette approche est simple et directe, mais elle nécessite un espace de stockage supplémentaire. Ainsi, nous avons choisi l'autre option consistant à



décider si un seul triple <s> <p> <o> est interprété comme une arête dirigée ou non dirigée lors de l'exécution de traversées dans les implémentations d'algorithmes. Par exemple, lors d'une traversée à 1 saut à partir d'un URI identifié par un nœud <a>, le triple <a> <p> <b> et <c> <p> <a> doit être pris en compte. Avec cette approche, nous pouvons économiser de l'espace de stockage par rapport à la première approche. Cependant, les algorithmes de graphes devraient être plus soigneusement conçus pour ses itinéraires.

3.3.1.2 Architecture partitionnée rdf et convert a JSON traité Pyspark

Figure 25 Architecture partitionnée rdf et convert a JSON traité Pyspark

1 etape inséré le fichier RDF dans pyspark et des fonctions lui sont appliquées afin de le rendre entièrement lisible. Des fonctions de division spéciales sont implémentées et placées dans RDD et placées dans des fichiers RDF tels qu'ils leur sont donnés afin de les diviser uniformément

Lorsqu'un fichier RDF est divisé, sa taille est réduite et la machine doit travailler sur le fichier facilement et rapidement, tous les fichiers sont traités et joints afin de terminer le traitement du fichier d'origine.

Lorsque a partitionnée d'un fichier en fichiers, ici chaque fichier est converti en JSON car JSON a des fonctions spéciales dans Pyspark et Pyspark peut lire plusieurs fichiers en même temps

Spark SQL peut déduire automatiquement le schéma d'un ensemble de données JSON et le charger en tant que Dataset[Row]. Cette conversion peut être effectuée à l'aide de SparkSession.read.json() sur un Dataset[String] ou un fichier JSON.

Cela facilite la lecture et traitement , avec conversion en JSON ou XML et donne un résultat direct via SQL spark

Il est facile d'extraire une colonne d'un DataFrame :

SparkSQL rajoute une couche simili-SQL au dessus des RDD de Spark. Ça s'appuie sur deux concepts :

DataFrames Ce sont des tables SparkSQL : des données sous forme de colonnes nommées. On peut les construire à partir de fichiers JSON, de RDD ou de tables Hive (voir le dernier cours). RDDSchema C'est la définition de la structure d'un DataFrame. C'est la liste des colonnes et de leurs types. Un RDDSchema peut être défini à l'aide d'un fichier JSON.

Il y a des liens entre DataFrame et RDD. Les RDD ne sont que des données, des n-uplets bruts. Les DataFrames sont accompagnées d'un schéma.

3.3.1.3 Architecture de partitionne rdf et lire via RDD in pyspark

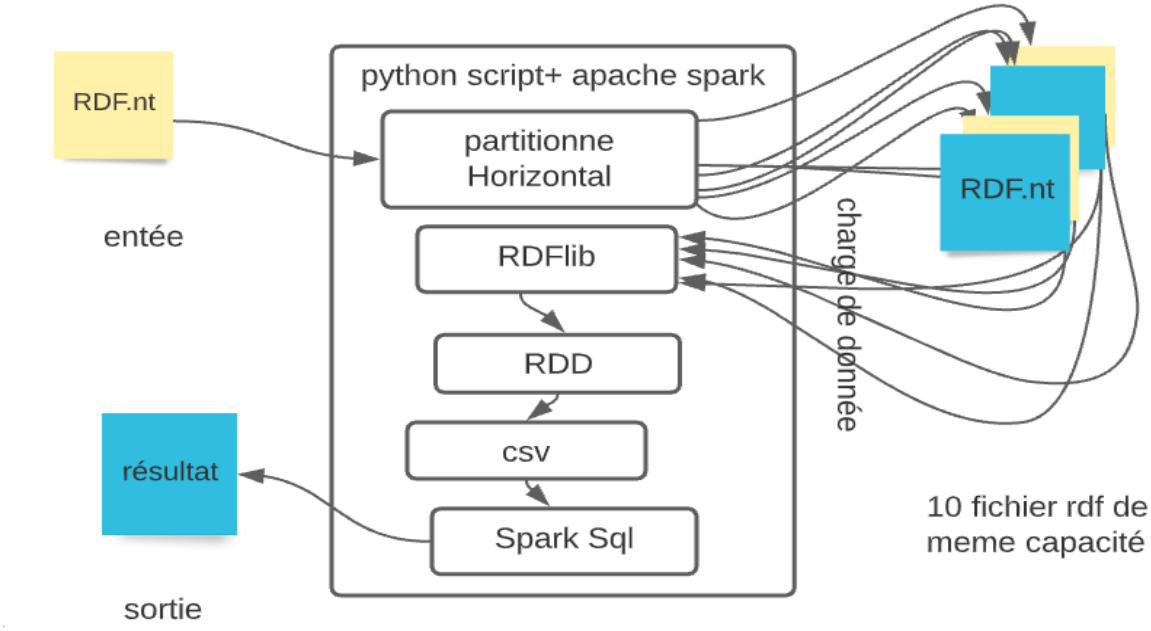


Figure 26 Architecture de partitionne rdf et lire via RDD in pyspark

Je suggère un système d'interrogation de données RDF à grande échelle basé sur Spark. La figure 4 montre l'architecture de notre système. Dans l'ensemble, la structure du système comprend cinq aspects : un module de partitionnement horizontal ,un module de préparation de données , un module de traitement distribué., un module de données persistantes, un module d'analyseur de requêtes

RDFLib est un package Python pur pour travailler avec RDF. RDFLib contient la plupart des éléments dont vous avez besoin pour travailler avec RDF, notamment :

analyseurs et sérialiseurs pour RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, Trig et JSON-LD

RDFLib représente les deux instructions dans le fichier : (URIRef ,URIRef) les instructions elles-mêmes ne sont que des groupes de longueur 3 ("triples"), et les sujets, prédicats et objets des triplets sont tous de type rdflib.

Il n'y a pas de fonctions spéciale de RDF dans pyspark, nous devons donc lire un fichier RDF qui inconnu dans pyspark et le faire via les fonctions pyspark pour changer le fichier et stocker le fichier dans rdd, et changer ce qu'il y a dans le rdd avec des fonctions spark par exemple(un map ou un filter..).et les fonction au sens mathématique du type : RDD ← transformation(RDD)

jusqu'à ce que nous commencions à travailler comme une liste dans laquelle sujet prédicat objet

permet d’écrire des traitements complexes composés de plusieurs phases map-reduce

le programme contient des données itérables liste, elles peuvent devenir un RDD par cette commend (donnees = list(sss) ,rdd = sc.parallelize(donnees)) donnees liste des donnee qui a été lire par rdflib (Jeux de données externs)

Ensuite, nous convertissons les données sur RDD en CSV afin de leur appliquer des fonctions CSV pour nous permettre de travailler plus facilement pour définir une table, et utilisons également SQL, et également pour travailler avec RDFLIB , car les données sont définies par(URIRef ,URIRef)

méthodes qui s’appliquent à un RDD pour retourner une valeur ou une collection :

Un RDD est une collection de données abstraite, résultant de la transformation d’un autre RDD ou d’une création à partir de données

Un RDD est distribué, c’est à dire réparti sur plusieurs machines afin de paralléliser les traitements.

liste = RDD.collect() retourne le RDD sous forme d’une liste Python. Attention à la taille si c’est du BigData.

nombre = RDD.count() retourne le nombre d’éléments

premier = RDD.first() retourne le premier élément

premiers = RDD.take(n) retourne les n premiers éléments. Note: il n’y a pas de méthode last pour retourner le ou les derniers éléments.

resultat = RDD.reduce(fonction) applique une fonction d’agrégation (associative) du type fn(a,b)→c

III.4 Implémentation

III.4.1 Outils et langages de programmation utilises

Pour la résiliation d’une application de sélection dans le Big data, nous avons utilisé quelques environnements de développement et langages de programmation, Citons :

4.1.1 Windows 10 :

est une version du système d'exploitation informatique Windows, produit par Microsoft, a été dévoilé le 30 septembre 2014, et sorti le 29 juillet 2015. Ce qui n'était pas prévu que le nom de cette nouvelle version soit « Windows 10 » et non "Windows 9". La

société déclare que le nom "Windows 9" ne correspondra pas au grand saut que la société a fait dans le nouveau système d'exploitation.

Désormais, le processus de navigation entre les fenêtres ouvertes et les applications utilisées est plus rapide et plus facile que jamais, ainsi que la division de l'écran avec facilité.

Vous pouvez maintenant utiliser directement les raccourcis copier-coller

La fonction de recherche de manière indépendante dans le menu Démarrer et elle présente de nombreux avantages qui recherchent le nom du fichier que vous souhaitez rechercher sur tout votre appareil, et s'il ne le trouve pas, vous pouvez rechercher sur Internet, et il conserve également les commandes de recherche que vous avez effectuées récemment afin qu'il vous soit plus facile de rechercher à l'avenir

4.1.2 Spark

Apache Spark est un moteur d'analyse unifié open source pour le traitement de données à grande échelle. Spark fournit à l'ensemble de l'interface de programmation de blocs un parallélisme de données implicite et une tolérance aux pannes. Développée à l'origine à l'AMPLab de l'Université de Californie, Berkeley, la base de code Spark a ensuite été donnée à la Apache Software Foundation, qui la maintient depuis.

Spark et ses RDD ont été développés en 2012 en réponse aux limitations du modèle de calcul en cluster MapReduce.

Spark peut interagir avec une grande variété, y compris Alluxio, Hadoop, Distributed File System (HDFS),[12] MapR File System (MapR-FS), [13] Cassandra, [14] OpenStack Swift, Amazon S3, Kudu, Lustre . des dossiers

4.1.2 Python

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau efficaces et d'une approche simple mais efficace de la programmation orientée objet. La syntaxe élégante et le typage dynamique de Python, ainsi que sa nature interprétée, en font un langage idéal pour les scripts et le développement rapide d'applications dans de nombreux domaines sur la plupart des plates-formes.

L'interpréteur Python et la vaste bibliothèque standard sont disponibles gratuitement sous forme source ou binaire pour toutes les principales plates-formes à partir du site Web Python, [20] et peuvent être distribués gratuitement. Le même site contient également des distributions et des pointeurs vers de nombreux modules, programmes et outils Python tiers gratuits, ainsi que de la documentation supplémentaire.

3.4.2 Environnement de travail:

Pour installation et la mise en œuvre de Hadoop plateforme, On a choisi le système d'exploitation Linux avec la distribution ubuntu 18.10. installé :

- **sur une machine Dell**
- Processeur Intel® Core™ i5-4210u 1.70GHz.
- RAM 4.00 Go.
- Disque 500 Go
- Système d exploitation : windows 10
- Type de systeme :64 bits.
- **sur une machine HP**
- Processeur intel® Core™ i3-6005 @ 2.30GHz.
- RAM 8.00 Go.
- Disque 1000 Go
- Système d exploitation : windows 10
- Type de systeme :64 bits.

3.4.2.1 La plateforme logicielle

Afin de réaliser un algorithme performant, il est indispensable d'utiliser des plateformes appropriées. Pour mettre en marche notre algorithme, notre choix s'est porté sur la plateforme pyspark (installe apache spark 3.1.1 en python 3.9.4) qui est installé sur le windows 10 .Les étapes d'installation de python 3.9.4 et apache spark 3.1.1 hadoop2.7 les suivants :

3.4.3 Les étapes de notre solution:

Pour l'implémentation de notre solution on va suivre les étapes suivantes

1. L'installation de JAVA.
2. L'installation de Python
3. L'installation de Spark

4. Introduction d'un fichier de donnée par l'administrateur.
5. Partitionner les données selon les mesures de la sélection.
6. Appliquer les requêtes nécessaires pour la sélection.
7. Afficher le résultat de sélection de l'utilisateur.

3.4.4 Instalation :

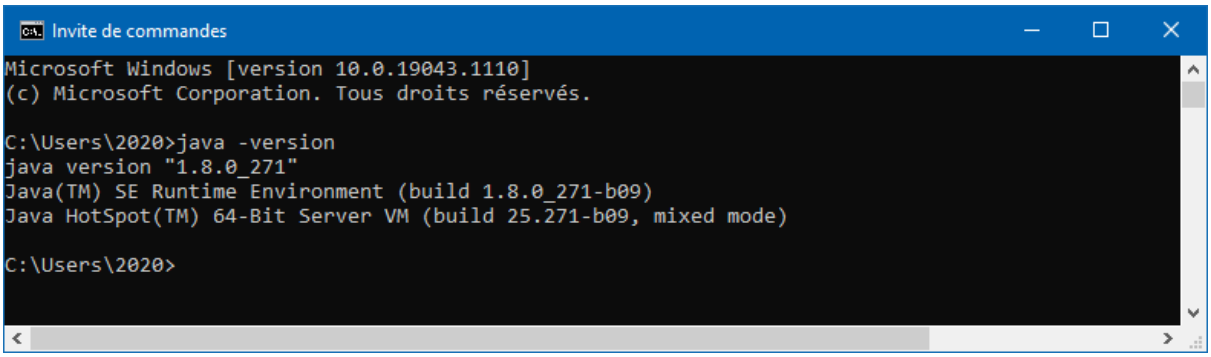
Je vais montrer comment installer Apache Spark sur Windows 10 et tester l'installation

Étape 1 : Installez Java 8

Apache Spark nécessite Java 8. Vous pouvez vérifier si Java est installé à l'aide de l'invite de commande.

Ouvrez la ligne de commande en cliquant sur Démarrer > tapez cmd > cliquez sur Invite de commandes.

Tapez la commande suivante dans l'invite de commande :

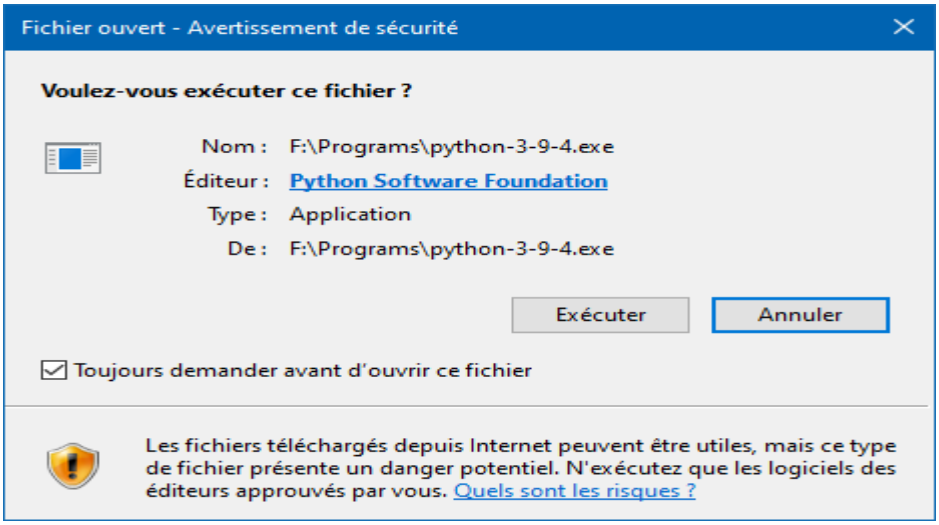


Étape 2 : Installez Python

1. Pour installer le gestionnaire de packages Python, accédez à https://www.python.org/ dans votre navigateur Web.

2. Passez la souris sur l'option de menu Télécharger et cliquez sur Python 3.9.4est la dernière version au moment de la rédaction de l'article.

3. Une fois le téléchargement terminé, exécutez le fichier.



4. Près du bas de la première boîte de dialogue de configuration, cochez Ajouter Python 3.9.4 à PATH. Laissez l'autre case cochée.

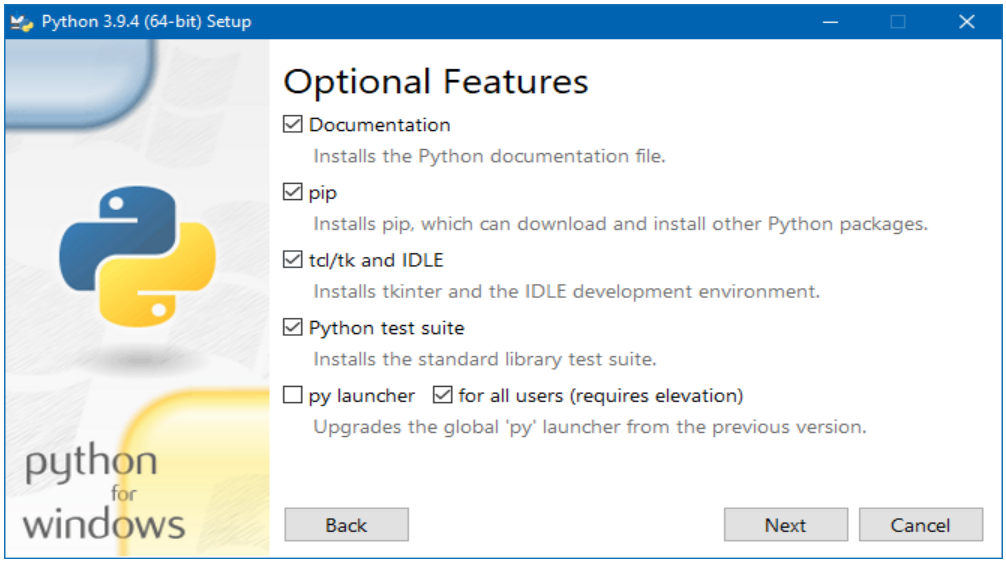
5. Ensuite, cliquez sur **Customize installation**.

6. Vous pouvez laisser toutes les cases cochées à cette étape, ou vous pouvez décocher les options que vous ne voulez pas.

7. Cliquez sur **Next**.

8. Cochez la case **for all users** et laissez les autres cases telles quelles.

9. Sous Personnaliser l'emplacement d'installation, cliquez sur **Browse** et accédez au lecteur C. Ajoutez un nouveau dossier et nommez-le Python.



10. Sélectionnez ce dossier et cliquez sur OK.

Étape 3 : Téléchargez Apache Spark

1. Ouvrez un navigateur et accédez à https://spark.apache.org/downloads.html.

2. Sous l'en-tête Télécharger Apache Spark, il y a deux menus déroulants. Utilisez la version actuelle sans prévisualisation.

Dans notre cas, dans le menu déroulant Choisir une version Spark, sélectionnez 3.1.2 (5 juin 2021).

Dans la deuxième liste déroulante Choisissez un type de package, laissez la sélection Pré-construit pour Apache Hadoop 3.2.

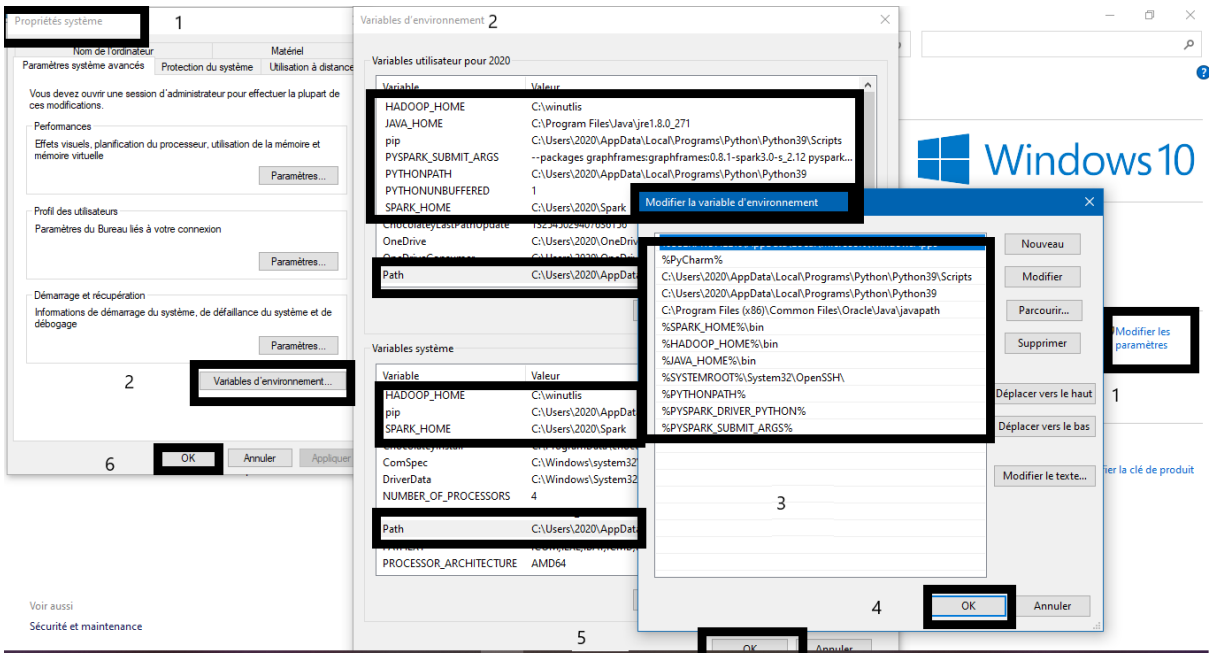
1. 3. Cliquez sur le lien : spark-3.1.2-bin-hadoop3.2.tgz

4. Une page avec une liste de miroirs se charge où vous pouvez voir différents serveurs à partir desquels télécharger. Choisissez-en un dans la liste et enregistrez le fichier dans votre dossier Téléchargements.

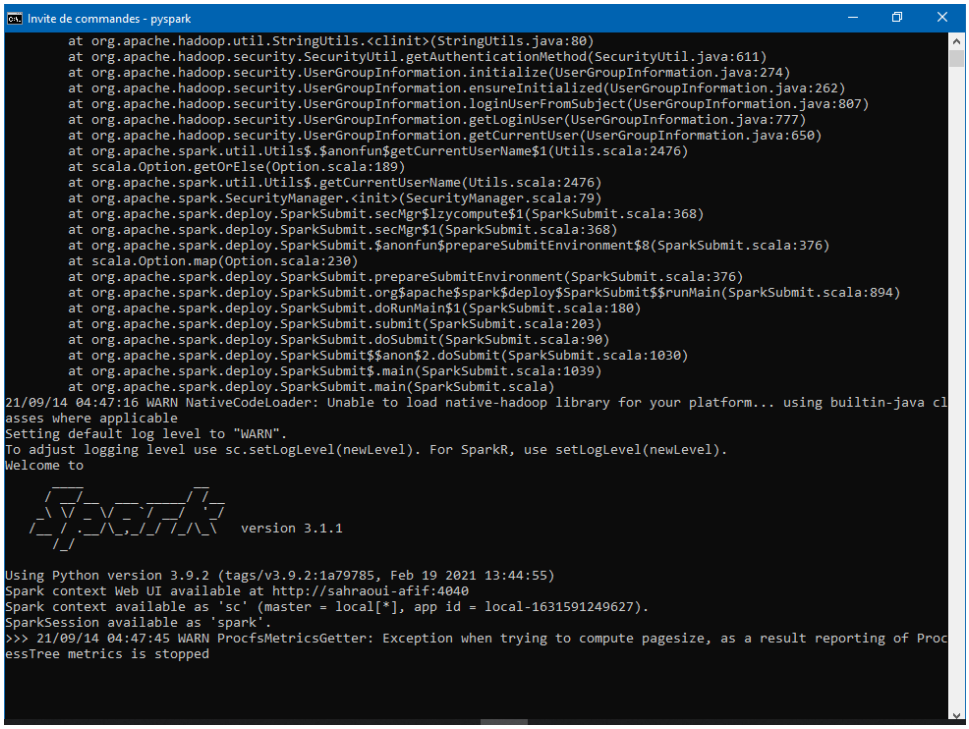
5. Pour installer Apache Spark, il n'est pas nécessaire d'exécuter un programme d'installation. Vous pouvez extraire les fichiers de l'archive téléchargée dans n'importe quel dossier de votre choix à l'aide de l'outil 7Zip.

6. Assurez-vous que le chemin du dossier et le nom du dossier contenant les fichiers Spark ne contiennent aucun espace.

7. Dans mon cas, j'ai créé un dossier appelé spark sur mon lecteur C et extrait l'archive zippée dans un dossier appelé spark-3.1.2-bin-hadoop3.2. Ainsi, tous les fichiers Spark se trouvent dans un dossier appelé C:\spark\spark-3.1.2-bin-hadoop3.2. À partir de maintenant, je ferai référence à ce dossier sous le nom de SPARK_HOME dans cet article.



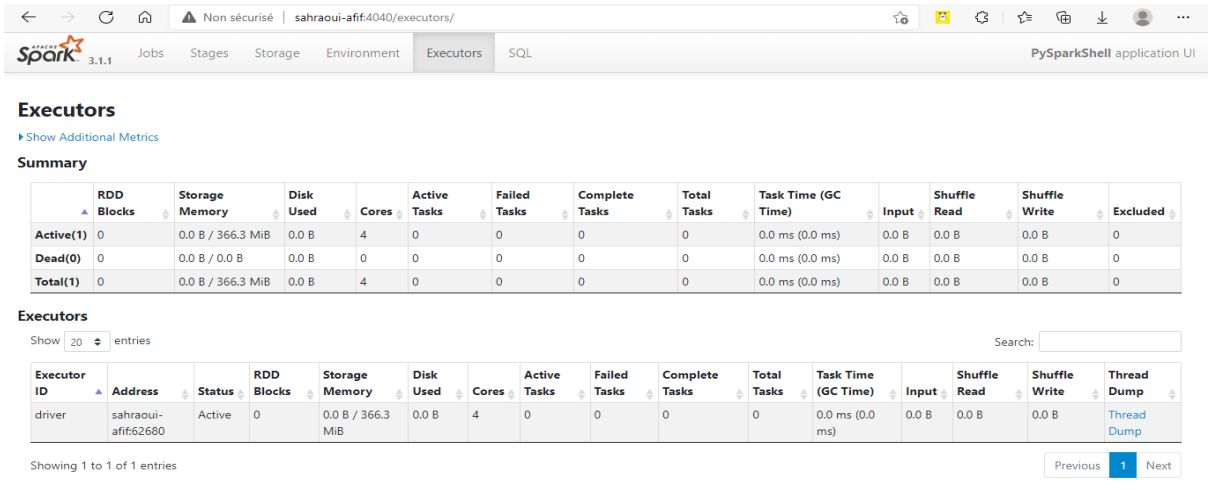
8. Pour tester si votre installation a réussi, ouvrez une invite de commande, accédez au répertoire SPARK_HOME et tapez bin\pyspark. Cela devrait démarrer le shell PySpark qui peut être utilisé pour travailler de manière interactive avec Spark. J'ai reçu les messages suivants dans la console après avoir exécuté la commande bin\pyspark.



9. Ouvrez un navigateur Web et accédez à **http://localhost:4040/**.

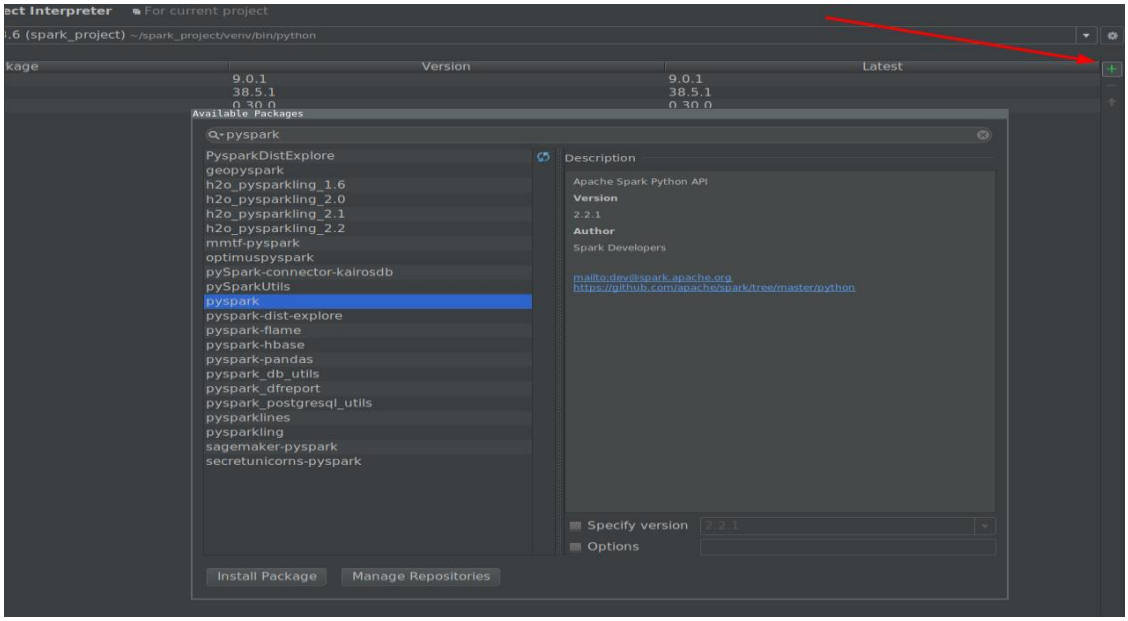
10. Vous pouvez remplacer **localhost par** le nom de votre système.

11. Vous devriez voir une interface utilisateur Web shell Apache Spark. L'exemple ci-dessous montre la page *Exécuteurs.*



Avec le package PySpark (Spark 3.1.2 et versions ultérieures)

1. Accédez à **Fichier -Paramètres** > -> **Interpréteur de projet**
2. Cliquez sur le bouton d'installation et recherchez PySpark



3. Click on install package button.

3.4.5 Traitement de la base de données:

Dans cette partie et à partir de ce que nous avons fait précédent, nous allons mettre en œuvre l'interface graphique de notre système pour interaction avec les utilisateurs.

Cette interface permet à l'administrateur ou l'utilisateur d'entrer dans notre système après la vitrification de leur compte.

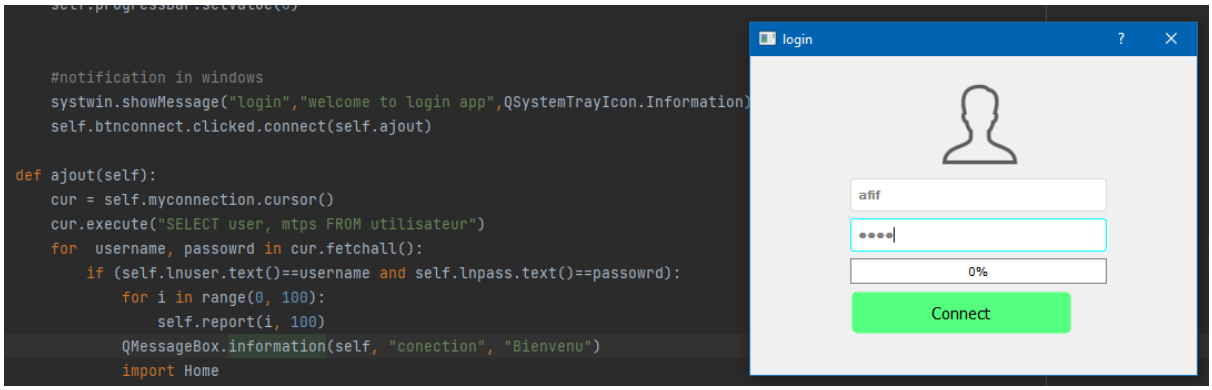
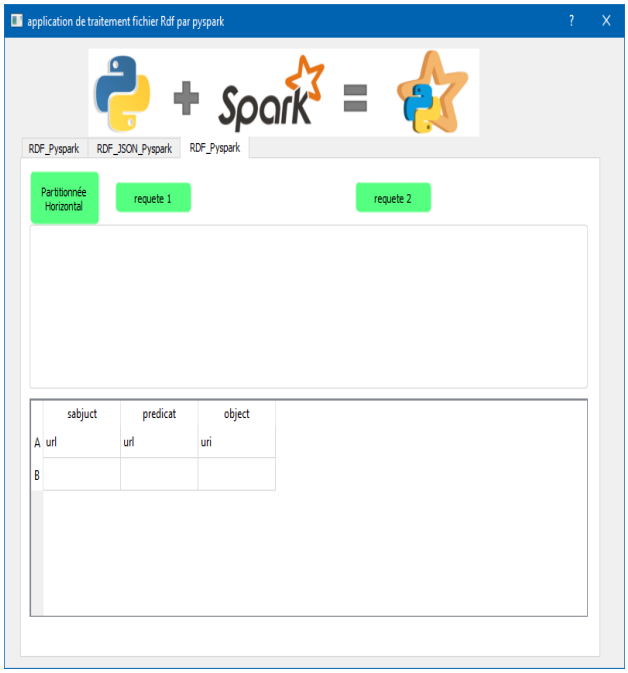


Figure27 .Interface d'authentification.

Cette interface est une interface administrateur lui permettant d'accéder à la base des données et extraire les données d'un emplacement et les stockées dans le système.



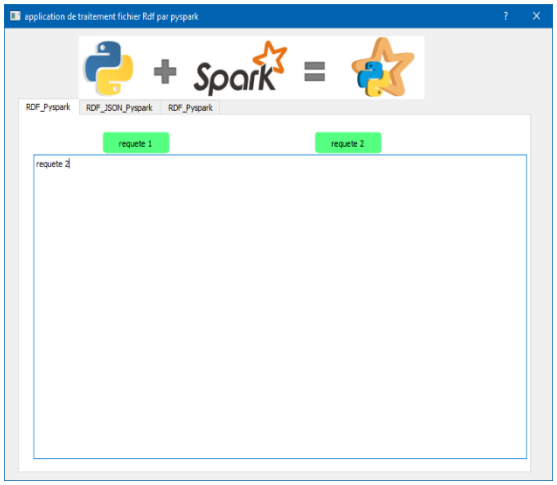
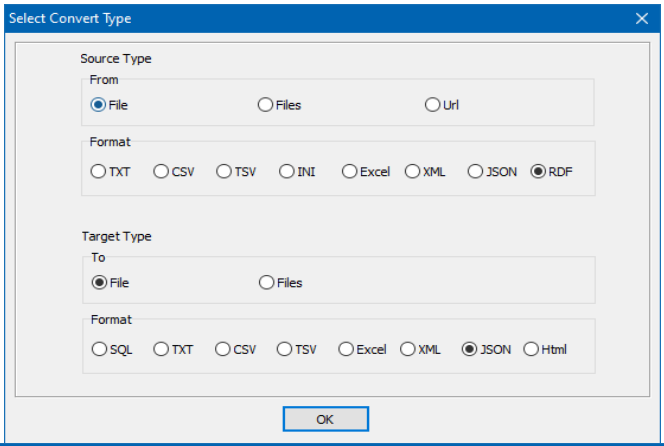
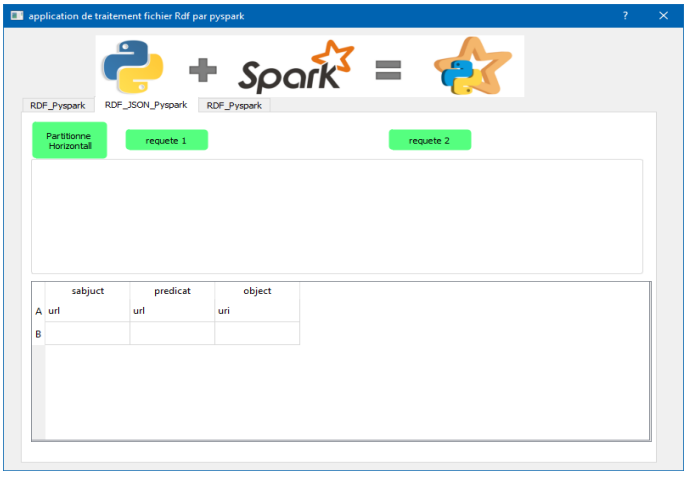


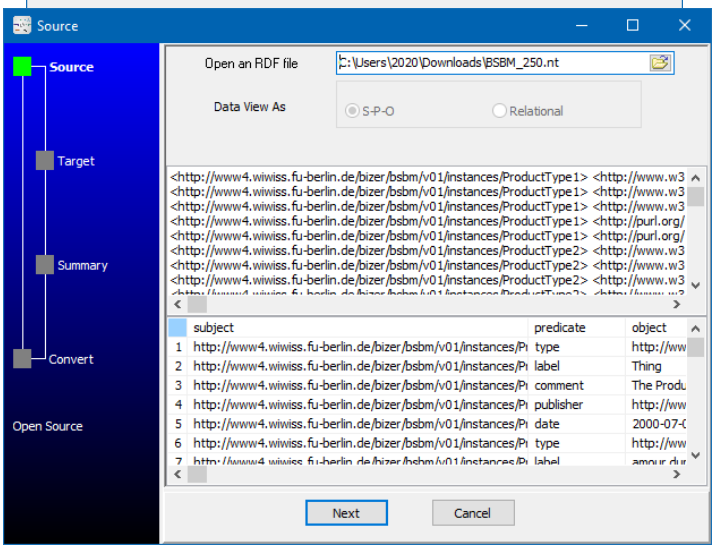
Figure 27. Interface d'administrateur.

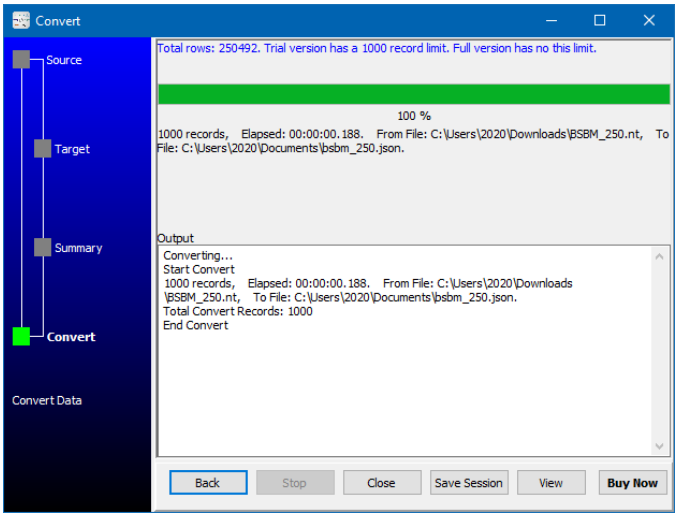
Figure 28. Interface converture RDF to json

Figure 27. Interface d'administrateur









Pour traiter un fichier RDF via Spark, une application n'a pas été implémentée en lisant des exemples tels que 3 difficultés de langue, RAM 4, win 10, espace libre pur, 1 Go, et pycharme prend environ 1 Go de RAM ou plus, ce qui rend le application incapable de travailler seule lorsque le mode optimal est devenu Une application par elle-même disparaît, et j'ai décidé de vous expliquer ces exemples sous la forme suivante

Traité rdf par sparql sans pyspark :sparql rdf par api ,Ici via Sparql nous avons traité un lien avec les informations RDF et c'était un fichier pour le laboratoire médical où dans la requête 1 définissez l'url db rousource w omim et récupérez toutes les maladies affectées par l'attribut disease contenant le mélanome par api et ici nous avons besoin d'une présence en ligne et cela via jsn les données ont été traitées afin que nous puissions proposer un plan

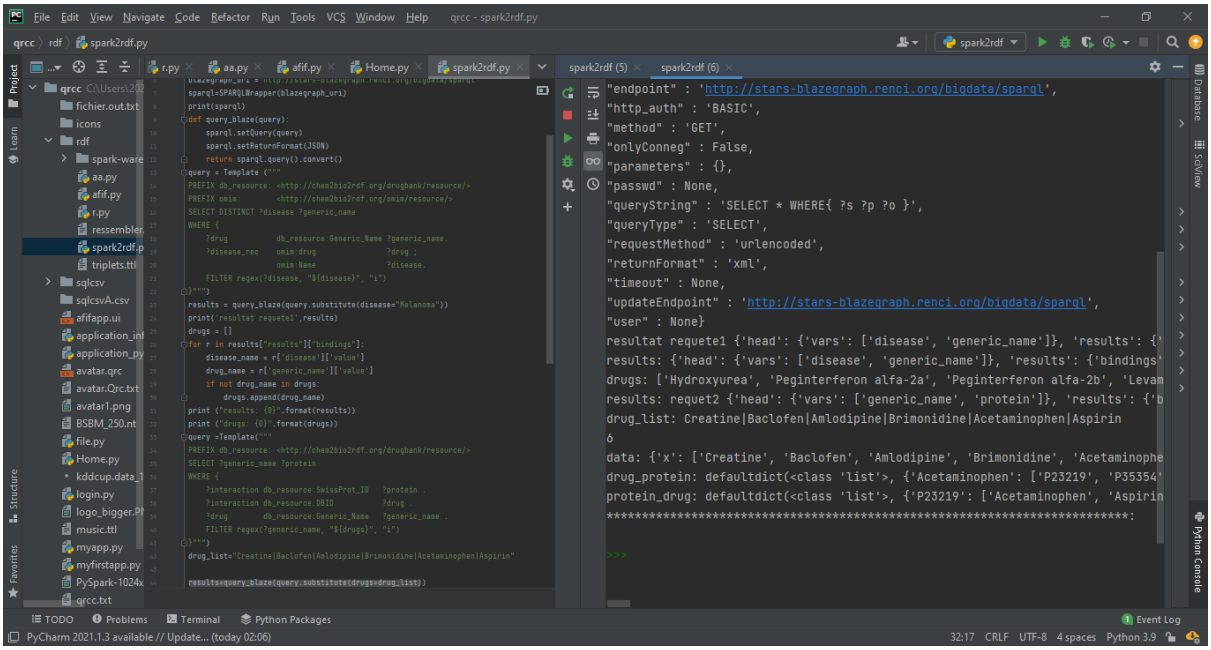


Figure 29. traitement Spraql par lien de Rdf

Voici le processus de partitionnee horizontale Nous divisons un fichier en 10 fichiers, et chaque fichier a la même taille que le deuxième fichier, et nous traitons chaque fichier comme indiqué dans le formulaire correspondant

partitionnée rdf et convert a JSON traité Pyspark :

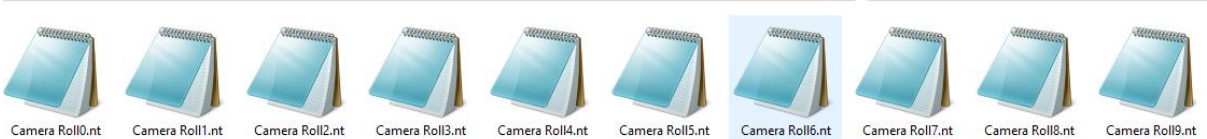


Figure 30. Fichier qui a été partitionne horizontal dans spark

Par a pour le fichier a de capacite 78 mo a partitiennée a 12 seq moyen

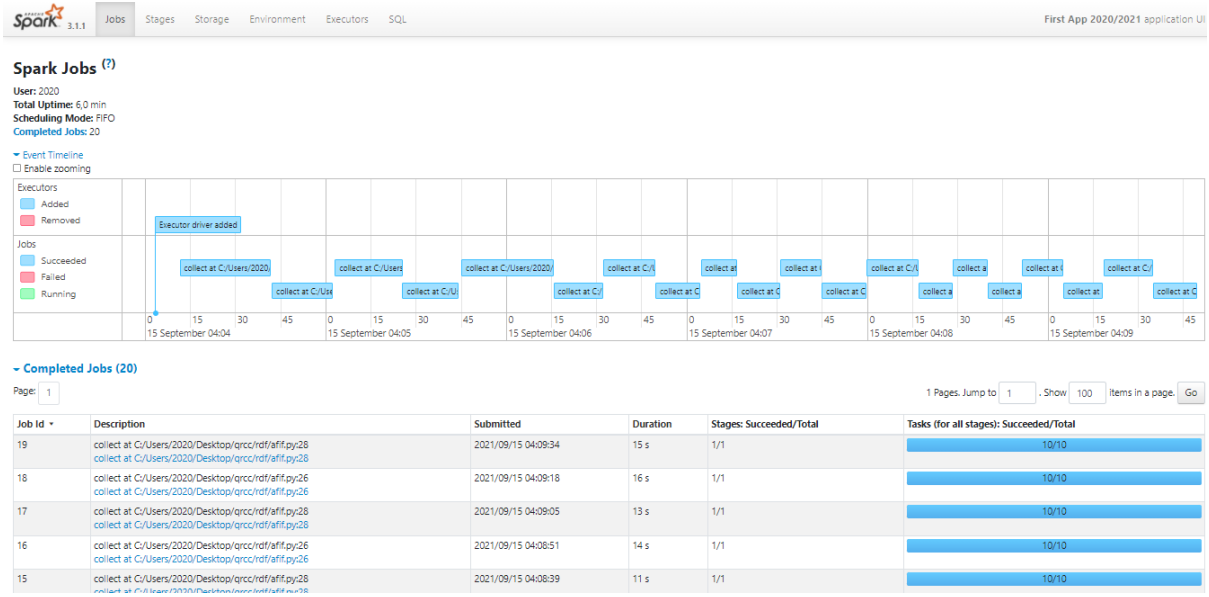


Figure 31. Fichier qui a été partitionne horizontal

Ici, l'opération JSON a été effectuée avec succès avec la conversion d'un fichier RDF et la création d'une table en SQL et l'extraction d'une table et l'affichage du Requete par un tabeu que ne contient pa in predicate label

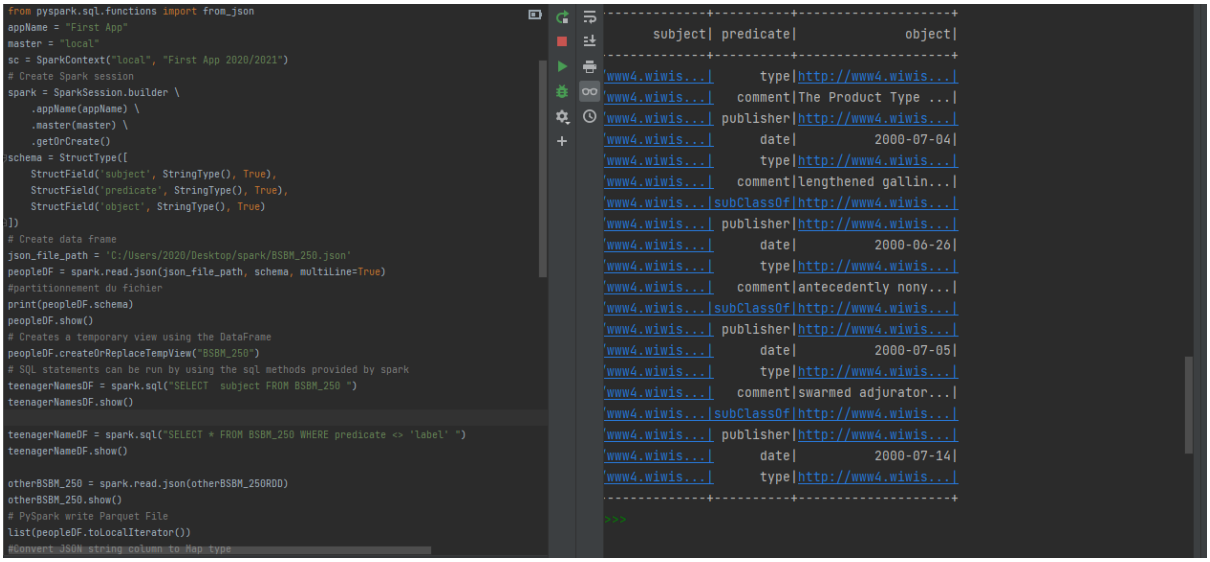


Figure 32 traitement de fichier json on pyspark

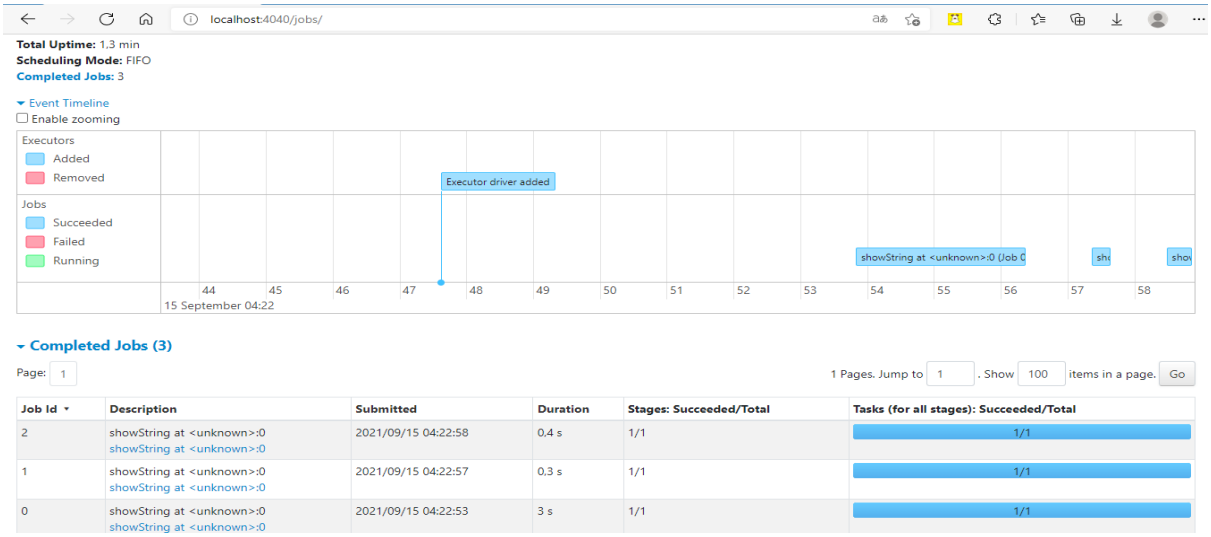


Figure 33. traitement de fichier json par pyspark in server spark

Partitionne RDF et lire via RDD in pyspark :Dans le troisième exemple, via un fichier d'origine, ce code n'a pas pu fonctionner.Maintenant, lors d'un processus de partitionnement, je vais travailler sur l'un des fichiers partitionnés

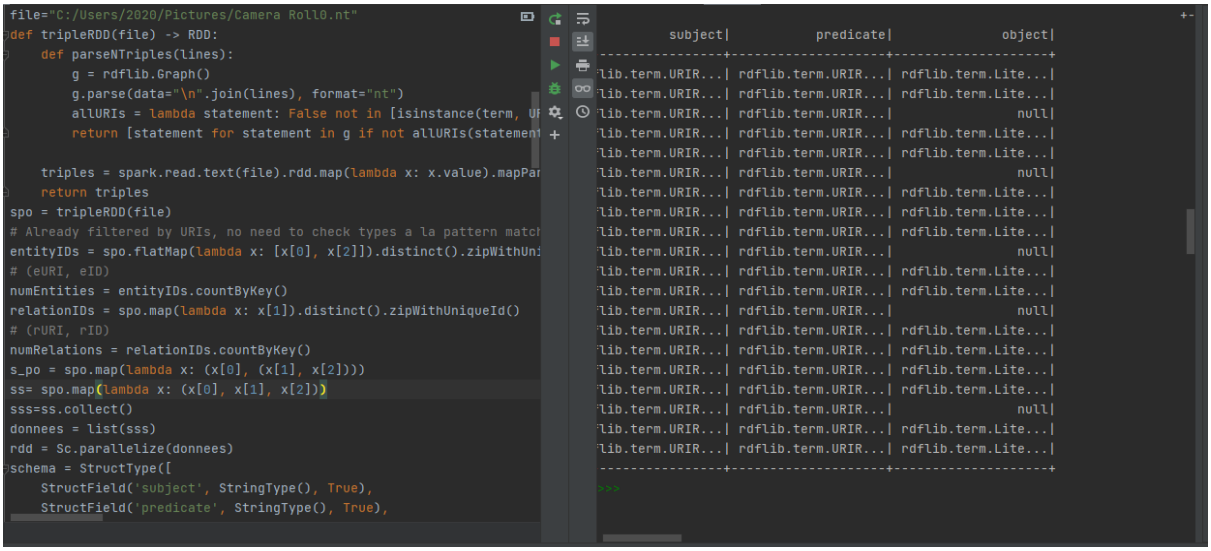


Figure 34. traitement de fichier rdf on pyspark

Ici les fichiers sont rangés, leurs dossiers sont stockés, ils sont stockés dans rdflib puis les informations sous la forme d'une petite url sont remplacées, les informations sont placées sur le RDD pour être organisées en tableau][puis nous les collectons via le fonctions de spark map et filter et groupby ... et insérez-le dans csv avec rdd par shema l'objet subject predicat nous montrer la table que je veux et ici nous appliquons SQL pour récupérer ce que nous voulons de la table ,Processus très compliqué.

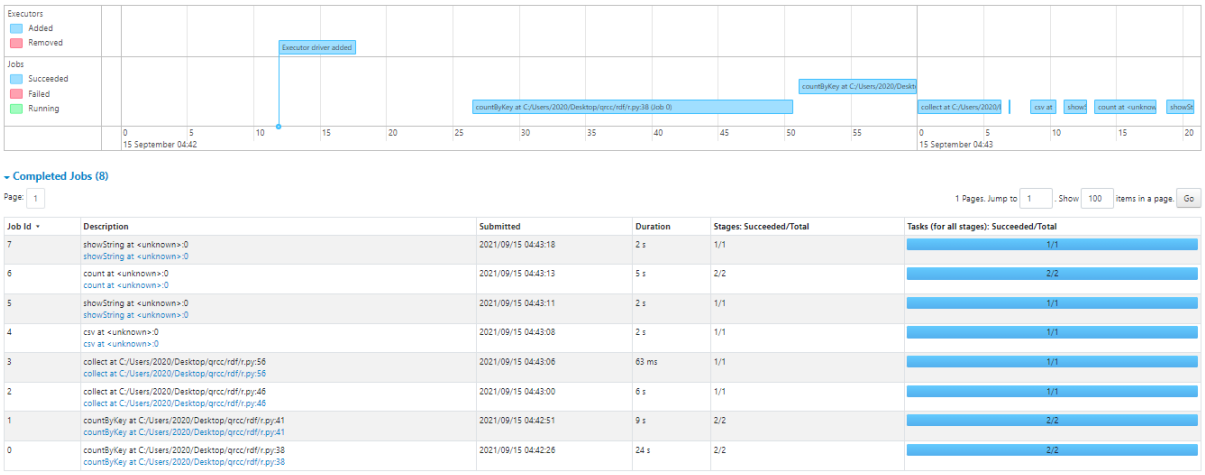


Figure 35 traitement de fichier rdf on pyspark server de spark

CONCLUSION Générale

Dans ce travail, nous avons mené une étude sur le big data et ses techniques dans le domaine de la sélection des candidats. Nous avons créé une application qui permet aux universités d'étudier et de faciliter le fonctionnement du Web sémantique, en appliquant un ensemble de normes aux données de stockage des bases de données. Pour résoudre le problème de l'augmentation du volume de données et de son impact pratique sur les ordinateurs en termes de non lecture de données énormes et de traitement difficile, nous avons fait une demande de big data. Nous avons utilisé SPARK comme outil de stockage et de traitement grâce à ces composants et RDD comme écosystème pour gérer ces données Nous avons eu des difficultés et des défis concernant la configuration de l'environnement système et l'installation des outils (apacher Spark Python) : installez d'abord Spark sur une machine et créez un cluster virtuel Nous avons installé et tout s'est bien passé mais nos machines ne peuvent pas gérer de gros fichiers car Spark traite avec la mémoire vive et c'était une chose difficile pour nos machines. Nous n'avons pas le choix, nous avons donc installé les outils dans un appareil et créé un groupe virtuel comme solution temporaire. Bref, on peut dire qu'à travers ce projet, on a beaucoup appris sur un nouveau domaine du big data. Après avoir parfaitement compris le domaine, nous nous sommes concentrés sur l'apprentissage d'un outil pour bien comprendre le principe de stockage et de traitement du big data. Nous avons choisi le framework spark qui est un outil très populaire. Nous avons pu l'installer avec les composants nécessaires malgré beaucoup de temps. C'était un défi pour nous au début. En développant une application, Destinations Désormais, nous espérons avoir l'opportunité de tester notre application dans des bâtiments très distribués. Spark est donc installé dans un vrai bloc de l'université pour permettre de résoudre les problèmes de stockage, d'augmentation des données et de profiter au maximum de l'amélioration des performances de l'université. Une autre perspective consiste à appliquer notre application à des données plus volumineuses et plus diversifiées.

Bibliographie

Livre, monographie

[1] http://www.redsen-consulting.com/2013/06/big-data/, consulté le 10/03/ 2021.

[2] https://www.oracle.com/fr/big-data/guide/what-is-big-data.html,consulté le 10/03/2021

[3] Big Data for Dummies, par Wiley Brand.

[4] BIG DATA Abdelouhab F Z Master2, RSD, 2018/2019 , consulté le 5/03/ 2021

[5] Big data et NoSQL,http://administrastration systeme.blog.com consulté le 5/03/ 2021

[6] KOUEDI Emmanuel, Approche de migration d'une base de données relationnelle versunebase de donnéesNoSQL orientée colonne, Mémoire master informatique,UNIVERSITE DE YAOUNDE I, mai 2012. consulté le 5/3/ 2021

[7] 1-IntroWebSem-BE-4P.pdf , Koivunen and Miller, 2001

[8] bdallah,Khelil.,Gestion et optimisation des données massives issues du Web. Autre [cs.OH]. ISAEENSMA,Ecole Nationale Supérieure de Mécanique et d'Aérotechique - Poitiers; Université d'Oran, 2020

[9] Interrogation SPARQL évolutive de grands graphes RDF Jiewen Huang huang , Daniel J. Abadi Université de Yale ,Kun Ren Polytechnique du Nord-Ouest Université, Chine renkun

[10] Memoire Online - Conception d'un système auteur pour la création et la manipulation d'une base de ressources pédagogiques - Manel et Hanane Bouhemila et Kahoul

[11] http://websemantique.orf consulté le 5/3/ 2021

[12] Memoire Online - Conception d'une ontologie pour une plate forme d'enseignement à distance - Saloua , Amina Chettibi , Rouibah consulté le 5/3/ 2021

[13] Big Data in metagenomics: Apache Spark vs MPIJosé M. Abuín, Nuno Lopes, Luís Ferreira,Tomás F. Pena, Bertil Schmidt ,Publié: 6 octobre 2020

[14] (PDF) Performance of K-means in Hadoop Using MapReduce Programming Model (researchgate.net) Department of Informatics, Faculty of Science and Technology, Sanata Dharma University, Mrican, Tromol Pos 29

[15] Linked Open Data Rights Survey | Results of a survey of the … | Flickr ,EC ISA Case Study: How Linked Data is transforming eGovernment, Leigh Doddsc

[16] D. Janke, S. Staab, and M. Thimm. On data placement strategies in distributed rdf stores. In 30 Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 August 2020 doi:10.20944/preprints202005.0360.v3Proceedings of The International Workshop on Semantic Big Data, SBD '17, New York, NY, USA, 2017. Association for Computing Machinery

[17]. Storage, Indexing, Query Processing, and Benchmarking in Centralized and Distributed RDF Engines: A Survey , Waqas Ali,Shanghai Jiao Tong University, Shanghai, China, Muhammad Saleem, Muhammad Saleem, Bin Yao, Shanghai Jiao Tong University, Shanghai, China , Publié: 28 août 2020

[18] A. Akhter, A.-C. N. Ngonga, and M. Saleem. An empirical evaluation of rdf graph partitioning techniques. In European Knowledge Acquisition Workshop, pages 3–18. Springer, 2018.

[19] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In 33rd International Conference on Very Large Data Bases, VLDB 2007 - Conference Proceedings, VLDB '07, pages 411–422. VLDB Endowment, 2007.

[20] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj5uqzdp8jvAhUCExoKHdWqA3MQFjAAegQIAxAD&url=https%3A%2F%2Fwww.alphorm.com%2Ftutoriel%2Fformation-en-ligne-big-data-avec-apache-spark-initiation%2Ftuto-video-l-ecosysteme-d-apache-spark&usg=AOvVaw34Mu9BGchZqP__uN7gsdK1 consulté le 24/03/ 2021

[21] Apache Spark: Avantages et Inconvénients - Analytics & Insights consulté le 24/03/ 2021

[22] Apache Spark Big Data - Next Decision (next-decision.fr) consulté le 24/03/ 2021

[23] Big Data Processing with Apache Spark – Part 1: Introduction (infoq.com) consulté le 24/03/ 2021

[24] afif sahraoui Conception et implémentation d'un moteur de recherche à base d'une architecture Hadoop (Big Data), Avril 2015.

[25] Introduction à Apache Hadoop : généralités sur HDFS et MapReduce (developpez.com)24-03-2021

[26] École des Mines de Nantes, 83p, Janvier 2015.

[27] L. R. JDN, « Hbase : le nosql au service du big data », 23-Avril-2013, [Accès le 24 /03/2021] adresse:http://www.journaldunet.com/developpeur/outils/comparatif-des-bases-nosql/hbase.shtml

[28] https://www.python.org/, consulté le 15/09/2021

[29]Linked Open Data Rights Survey | Results of a survey of the … | Flickr ,consulté le 15/09/2021

[30] Ling Liu a Georgia Institute of Technology , Kisung Lee a Georgia Institute of Technology ,Scaling Queries over Big RDF Graphs with Semantic Hash Partitioning ,a consulte 14/09/2021

[31] KOUEDI Emmanuel, Approche de migration d'une base de données relationnelle versunebase de donnéesNoSQL orientée colonne, Mémoire master informatique,UNIVERSITE DE YAOUNDE I, mai 2012.