Master 1 informatique 2022/2023

Environnement et RD en informatique

Rapport

Stream processing : Traitement de données en flux

Membres du groupe

Smail Aghilas
Mouzni Lamara

Table des matières

Introduction 3
1. C'est quoi le stream processing ? 3
1.1. Comment cela fonctionne-il 4
2. Domaine d'application de stream processing 4
2.1. Détection de fraudes : 5
2.2. Analyse de périphérie : 5
2.3. Marketing en temps réel : 5
3. Principaux algorithmes méthodes de traitements 5
3.1. Échantillonnage et statistiques : 5
3.2. Méthodes d'accès à la marche : 6
3.3. Index et compression : 6
3.4. L'algorithme AMS : 7
3.5. L'algorithme BJKST : 7
4. Éléments d'architecture 8
4.1. Architecture Datalake: 8
4.2. Architecture Lambda : 8
4.3. Architecture Kappa: 9
5. Panorama des systèmes de stream processing 9
5.1. Apache Flink : 9
5.2. Spark : 10
5.3. Kafka : 11
5.4. Amazon Kinesis : 12
Conclusion 13
Bibliographie : 13

Résumé :

Avec l'émergence de l'internet et l'augmentations constante de nombre d'appareils connectés, des quantités massives de données sont constamment générées, ce qui complique leur traitement par les systèmes classiques de traitement de données.

Le stream processing semble être la meilleure réponse à cette problématique, il permet de traiter les données au fur et à mesure de leur arrivée en temps réel.

Spark, Flink et Kafka Streams sont les frameworks de traitement de flux open source les plus courants. En outre, tous les principaux services cloud disposent également de services natifs qui simplifient le développement du traitement de flux sur leurs plates-formes respectives, telles qu'Amazon Kinesis, Azure Stream Analytics et Google Cloud Dataflow.

Abstract :

With the emergence of the internet and the constant increase in the number of connected devices, massive amounts of data are constantly being generated, which complicates their processing by conventional data processing systems.

Stream processing seems to be the best answer to this problem, it allows data to be processed as it arrives in real time.

Spark, Flink and Kafka Streams are the most common open source stream processing frameworks. Additionally, all major cloud services also have native services that simplify the development of stream processing on their respective platforms, such as Amazon Kinesis, Azure Stream Analytics, and Google Cloud Dataflow.

Introduction

Avec la popularisation de l'Internet des objets (IoT), le nombre d'appareils intelligents utilisés pour la surveillance, la gestion et la maintenance augmente rapidement, et de nouvelles données sont constamment générées, formant des flux de données massifs, qui finiront par submerger les flux de données traditionnels et les systèmes de gestion des données.

Parallèlement, à mesure que le débit de données continue d'augmenter, le besoin de traitement des données à faible latence augmente également. Les applications urgentes telles que la détection des fraudes, le trading algorithmique et la surveillance de la santé gagnent en popularité et reposent toutes fortement sur la garantie d'une faible latence pour obtenir des résultats significatifs.

Le désir d'une analyse rapide des données a conduit au traitement par flux qui est le Stream processing en anglais, cette nouvelle technologie qui permet la collecte, l'analyse et la visualisation continues des données avec une latence de quelques secondes ou millisecondes qui est vraiment très rapide. Contrairement au paradigme ancien « stocker maintenant, traiter plus tard » du traitement par lots, le traitement par flux (Stream processing) utilise les données entrantes pour fournir des informations instantanées avant que leur valeur ne diminue rapidement avec le temps

Les données entrantes sont traitées au fur et à mesure de leur arrivée et les résultats sont mis à jour progressivement au fur et à mesure que les données circulent dans le système. En raison des ressources limitées pour traiter l'entrée continue, le traitement de flux ne peut pas accéder de manière aléatoire à l'intégralité du flux.

Le but de ce rapport est de présenter le Stream processing (Traitement des données en flux), c'est quoi cette nouvelle technologie, place des données en flux dans le traitement de la data, les principaux algorithmes, méthodes de traitements de ce flux de données …

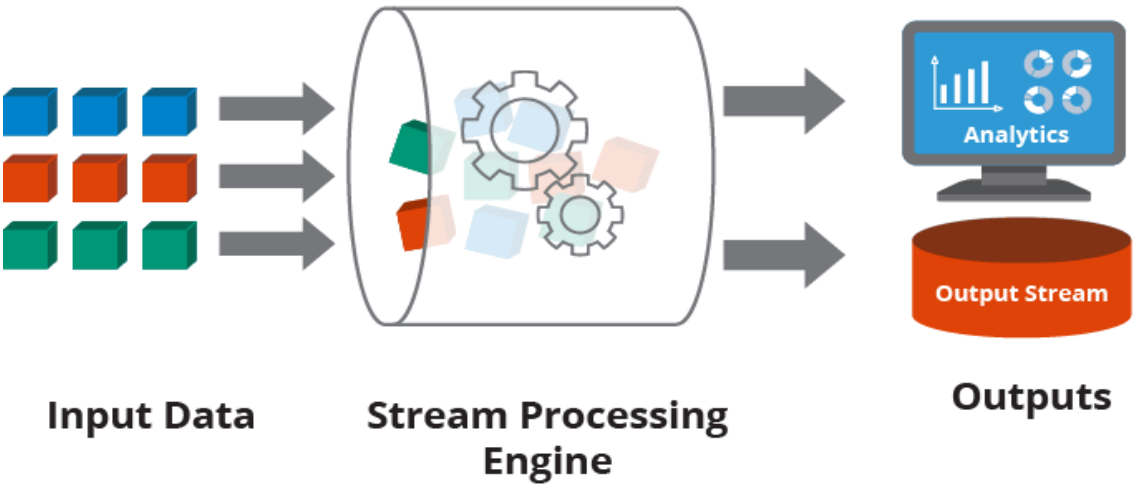
1. C'est quoi le stream processing ?

Le traitement de flux est une technologie big data qui se concentre sur le traitement en temps réel de flux continus de données en mouvement. Un cadre de traitement en continu simplifie le matériel et les logiciels parallèles en limitant les performances du calcul parallèle. Des fonctions de noyau en pipeline sont appliquées à chaque élément d'un flux de données, en utilisant la réutilisation de la mémoire sur puce pour minimiser la perte de bande passante. Les outils et les technologies de traitement de flux sont disponibles dans une variété de formats : systèmes de messagerie distribués de type publish-subscribe tels que Kafka, systèmes de calcul en temps réel distribués tels que Storm, et moteurs de flux de données en continu tels que Flink.

Les actions que le traitement de flux effectué sur les données comprennent les agrégations (par exemple, des calculs tels que la somme, la moyenne, l'écart type), l'analyse (par exemple, la prédiction d'un événement futur sur la base de modèles dans les données), les

transformations (par exemple, la conversion d'un nombre en un format de date), l'enrichissement (par exemple, la combinaison du point de données avec d'autres sources de données pour créer plus de contexte et de signification) et l'ingestion (par exemple, l'insertion des données dans une base de données).

Figure 1 : *Le traitement en flux permet aux applications de répondre aux nouveaux événements de données au moment où ils se produisent. Dans cet exemple simplifié, le pipeline de données d'entrée est traité par le moteur de traitement en continu en temps réel. Les données de sortie sont transmises à une application d'analyse en continu et ajoutées au flux de sortie.*



Il y a aussi des changements d'état, comme une transaction ou un prospect naviguant sur votre site Web. Les événements sont essentiellement des points de données capturés dans les systèmes d'entreprise. Un flux d'événements est une séquence chronologique d'événements commerciaux. Dans les activités quotidiennes de toute entreprise, les clients font constamment leurs courses, appellent le service d'assistance ou remplissent leurs paniers.

1.1. Comment cela fonctionne-il

Le traitement de flux est le plus souvent appliqué aux données générées sous la forme d'une série d'événements, telles que les données des capteurs IoT, les systèmes de traitement des paiements et les journaux des serveurs et des applications. Les paradigmes courants incluent l'éditeur/abonné (souvent appelé pub/sub) et là source/puits. Les données et les événements sont générés par des éditeurs ou des sources et livrés à des applications de traitement de flux, où les données peuvent être enrichies, testées avec des algorithmes de détection de fraude ou présentées avec Convert d'autres manières. Techniquement, les sources et les puits communs incluent Apache Kafka®, les référentiels de Big Data tels que Hadoop, les sockets TCP et les grilles de données en mémoire telles que Hazelcast IMDG.

2. Domaine d’application de stream processing

Les cas d'utilisation impliquent généralement des données d'événement qui sont générées par une action et sur lesquelles une action doit immédiatement se produire. Les cas d'utilisation courants du traitement des flux en temps réel sont les suivants :

2.1. Détection de fraudes :

Détection en temps réel des fraudes et anomalies. L'un des plus grands fournisseurs de cartes de crédit au monde a pu réduire les paiements frauduleux de 800 millions de dollars par an grâce à un traitement continu qui a facilité la détection des fraudes et des écarts. Les retards dans le traitement des cartes de crédit nuisent à l'expérience du client final et du magasin qui traite la carte de crédit (et des autres clients en ligne). Historiquement, les fournisseurs de cartes de crédit effectuent leurs longs processus de détection des fraudes par lots après la transaction. Avec un traitement continu, ils peuvent exécuter des algorithmes étendus immédiatement après le passage de la carte pour détecter et empêcher les paiements frauduleux et vous alerter des frais inhabituels qui nécessitent une vérification supplémentaire sans attendre les clients (non-fraude).

2.2. Analyse de périphérie :

Analyse de périphérie de l'Internet des objets (IoT). Les entreprises manufacturières, pétrolières, gazières et de transport, ainsi que les entreprises qui conçoivent des villes et des bâtiments intelligents, utilisent le traitement de flux pour suivre les données de milliards de "choses". Un exemple d'analyse de données IoT identifie des anomalies dans la fabrication qui indiquent des problèmes qui doivent être résolus pour améliorer les opérations et augmenter les revenus. Avec le traitement de flux en temps réel, un fabricant peut détecter qu'une ligne de production génère trop d'écarts au fur et à mesure qu'ils se produisent (au lieu de trouver l'ensemble du lot défectueux après un quart de travail quotidien). Il peut réaliser d'importantes économies et éviter d'énormes déchets en coupant la ligne pour des réparations immédiates.

2.3. Marketing en temps réel :

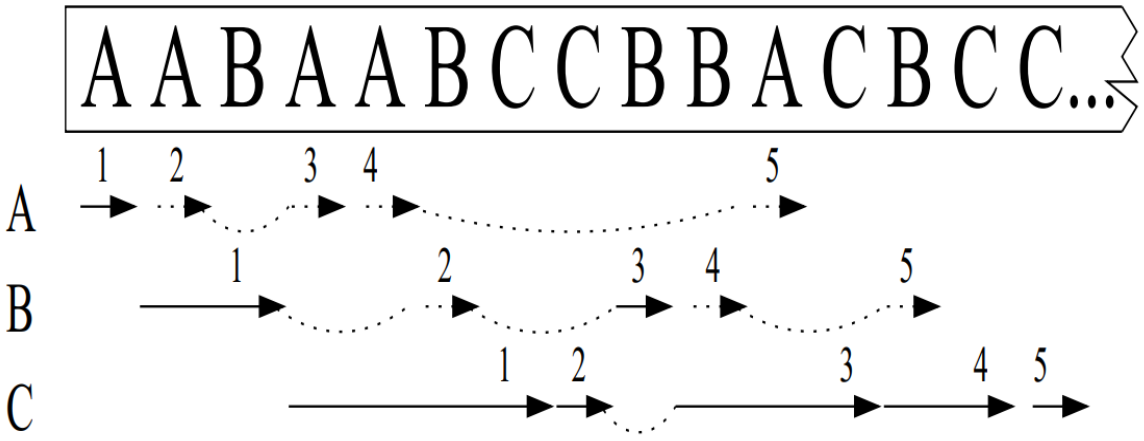
Personnalisation, marketing et publicité en temps réel. En traitant les flux en temps réel, les entreprises peuvent offrir des expériences personnalisées et contextuelles à leurs clients. Il peut s'agir d'une réduction sur un produit que vous avez ajouté à votre panier sur un site Web mais que vous n'avez pas acheté tout de suite, d'une recommandation de contacter un ami qui vient de s'inscrire sur un site de réseau social ou d'une publicité pour un produit similaire.

3. Principaux algorithmes méthodes de traitements

3.1. Échantillonnage et statistiques :

Pour qu'une estimation en ligne soit représentative, elle doit être construite sur un ensemble approprié d'échantillons d'entrées. Olken a proposé des méthodes d'accès par échantillonnage, qui sont tout à fait applicables aux applications qui exigent des garanties d'aléa pendant le traitement en ligne qui nécessitent des garanties d'aléa pendant le traitement en ligne. Dans de nombreux cas, des méthodes d'accès plus traditionnelles d'accès plus traditionnelles peuvent être utilisées, à condition que l'ordre d'accès ne soit pas corrélé avec l'estimation effectuée en cours.

Figure 2 : *Les grandes lettres représentent les attributs de groupe des tuples, les trois rangées de flèches représentent les accès par groupe, les lignes pleines indiquent les entrées/sorties initiales et les lignes en pointillés indiquant les tuples qui sont revisitées pour être livrées.*



3.2. Méthodes d'accès à la marche :

Étant donné un ensemble de données qui peut être décomposé en groupes, les méthodes d'accès strident fournissent des tuples de différents groupes à des taux relatifs contrôlables par l'utilisateur. Tuples des différents groupes à des taux relatifs contrôlables par l'utilisateur.

La méthode d'accès Index Stride a été présentée. Étant donné un index d'arbre B sur les colonnes du groupement de regroupement 1, à la première demande d'un tuple, nous ouvrons un balayage sur le bord le plus à gauche de l'index, où nous trouvons une valeur de clé k1. Trouvons une valeur de clé k1. Nous attribuons à cette analyse une clé de recherche de la forme [= k1]. Après avoir récupéré le premier tuple avec la valeur de clé k1, lors d'une requête ultérieure pour un tuple, nous ouvrons un deuxième balayage de l'index avec la clé de recherche [> k1], afin de trouver rapidement le groupe suivant dans la table. Lorsque nous trouvons cette valeur, k2, nous changeons la clé de recherche du second scan en la clé de recherche du deuxième balayage en [= k2], et nous retournons le tuple trouvé. Nous répétons cette procédure Pour les demandes suivantes jusqu'à ce que nous ayons une valeur kn telle qu'une clé de recherche [> kn] ne renvoie aucun tuples.

3.3. Index et compression :

Les index peuvent être exploités de diverses manières pour le traitement en ligne. Nous avons vu comment ils peuvent être utilisés pour l'accès strident. Ils peuvent également être utilisés plus directement pour fournir des estimations de renommée d'un ensemble de données.

Considérons la racine d'un index d'arbre B+. Elle contient un ensemble de clés k1;:::;kn qui partitionnent le jeu de données en n+1 compartiments de taille à peu près égale. Cela peut être considéré comme une représentation de la distribution des valeurs de la colonne indexée. Valeurs dans la colonne indexée. En fait, tout niveau de profondeur dans l'arbre est un histogramme grossièrement équi-profond de (n + 1) de cases. Ainsi, un arbre B+ standard peut être parcouru dans l'ordre de la largeur pour produire une estimation itérative de la distribution des valeurs. Estimation itérative d'un ensemble de données.

3.4. L'algorithme AMS :

Ce qui suit est l'algorithme d'Alon, Matias, Szegedy en 1996, pour estimer le nombre d'éléments distincts.

1. Choisissez une fonction de hachage aléatoire h : {1, 2, . . . , m} → {1, 2, . . . , m3} d'une famille de hachage indépendante par paires.
2. Initialisez z = 0.
3. Pour chaque élément x du flux, mettez à jour z comme z = M ax(zeroes(h(x)), z), Où zeroes(y) désigne le nombre de zéros à la fin de y dans la représentation binaire
4. Sortie 2 z+c pour c = 1/2.

Figure 3 : AMS Algorithme.

Algorithm 1 An Algorithm For Approximating F_0

Choose a random function $h : [n] \to [n]$ from a family of pairwise independent hash functions;
$z \leftarrow 0$;
while an item x arrives **do**
    if $\rho(h(x)) > z$ **then**
        $z \leftarrow \rho(h(x))$;
Return $2^{z+1/2}$

3.5. L'algorithme BJKST :

Voici l'algorithme de Bar-Yossef, Jayram, Kumar, Sivakumar et Trevisan, en 2002, pour estimer le nombre d'éléments distincts.

1. Utilisez un ensemble B pour conserver les éléments échantillonnés ;
2. Lorsque l'ensemble B est plein, réduisez B en enlevant environ la moitié des articles et de alors sur l'échantillon la probabilité devient plus petite.
3. En fin de compte, le nombre d'éléments de B peut être utilisé pour donner une bonne approximation du nombre d'éléments distincts dans S.

Figure 4 : BJKST Algorithme.

Algorithm 2 The BJKST Algorithm (Simplified Version)

Choose a random function $h : [n] \to [n]$ from a family of pairwise independent hash functions;
$z \leftarrow 0$;
$B \leftarrow \emptyset$
while an item x arrives **do**
    if $\rho(h(x)) \geq z$ **then**
        $B \leftarrow B \cup \{(x, \rho(h(x)))\}$;
        while $|B| \geq c/\varepsilon^2$ **do**
            $z \leftarrow z + 1$;
            shrink B by removing all $(x, \rho(h(x)))$ with $\rho(h(x)) < z$;
Return $|B| \cdot 2^z$

4. Éléments d'architecture

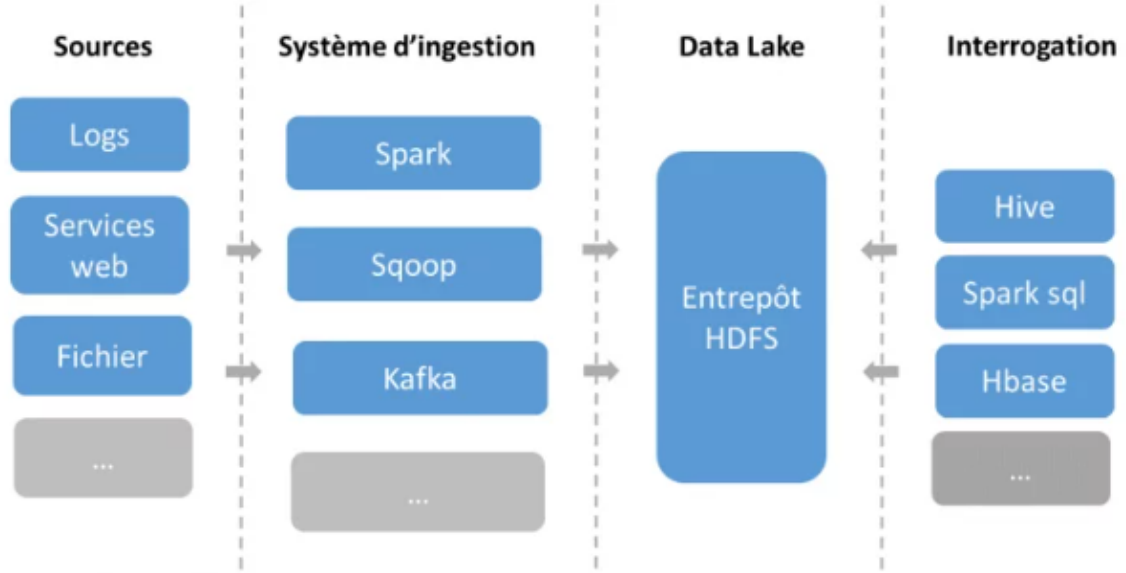
4.1. Architecture Datalake:

Datalake (ou datalake) est une architecture apparue avec les technologies Big Data et permet de stocker de gros volumes de données.

Datalake propose aux entreprises un système de stockage pouvant accueillir tout type de données (brutes ou non structurées), qu'elles soient structurées, semi-structurées et/ou non structurées. Cette architecture big data permet ainsi de transformer et d'affiner rapidement les données stockées, que le volume à traiter soit important ou non. Il peut être défini en un mot : adaptatif.

Bien que Hadoop ne soit pas le seul, il reste le framework le plus largement utilisé pour construire un Datalake.

Figure 5 : Architecture Datalake.



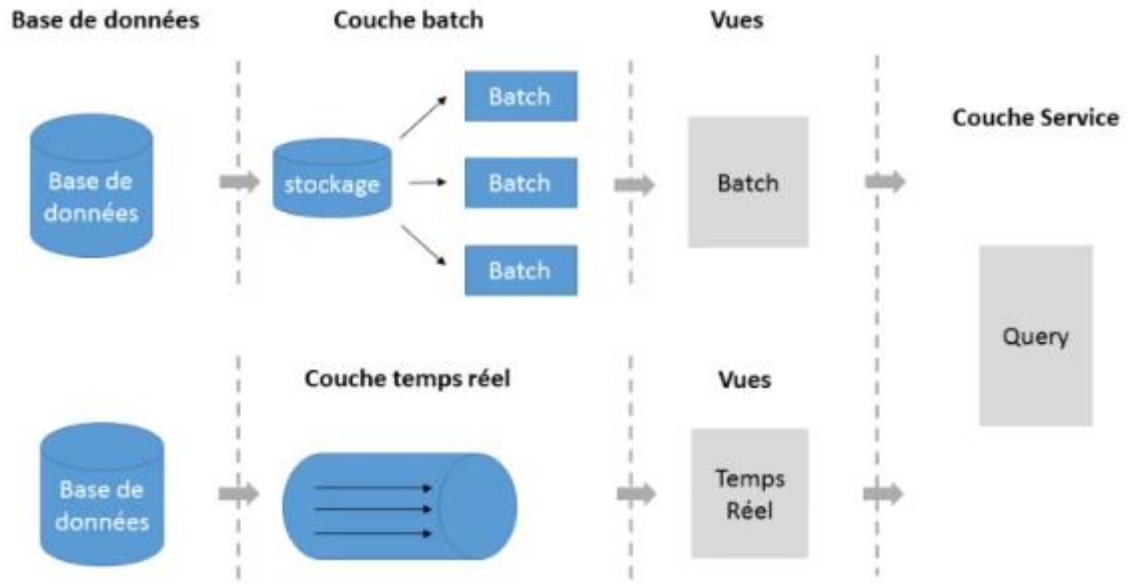
4.2. Architecture Lambda :

L'architecture Lambda est conçue pour effectuer simultanément un traitement de type batch (traitement des blocs de données) et un traitement en temps réel (en continu). Contrairement à Datalake, qui est principalement utilisé pour le stockage, l'architecture Lambda permet de traiter des blocs de données (ensembles) et de fusionner de nouvelles données d'entrée (temps réel).

L'architecture Lambda se découpe en 3 couches :

Couche batch (Batch Layer) .
Couche temps réel (Speed Layer) .
Couche de service (Serving Layer) .

Figure 6 : Architecture Lambda.



4.3. Architecture Kappa:

L'architecture KAPPA est conçue pour supprimer la complexité de l'architecture Lambda. Elle repose sur le principe de combiner des couches temps réel et batch, ce qui la rend moins complexe que l'architecture Lambda. Cette architecture ne permet pas le stockage persistant, mais est conçue pour le traitement des données.

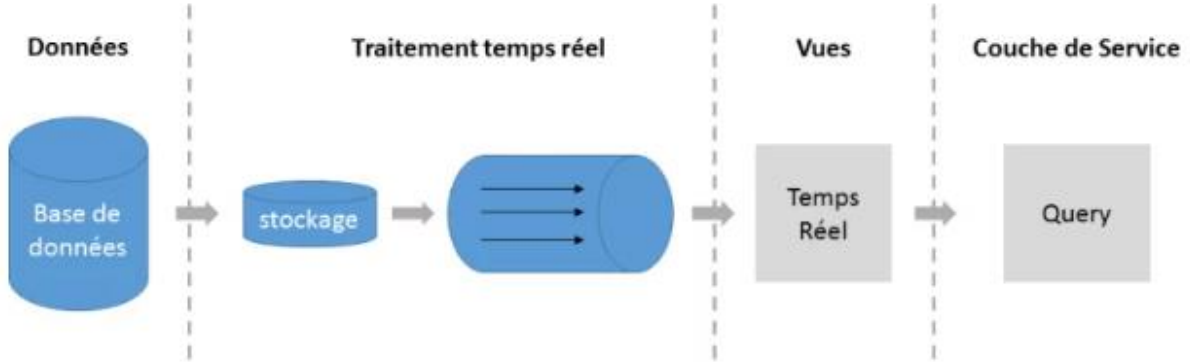
Kappa n'est pas non plus lié à une seule technologie, vous pouvez donc y connecter différents outils, comme le montre le schéma ci-dessous :

Stockage/Live : Kafka permet de stocker des messages à retraiter ultérieurement.

Traitement : Storm, Spark

Couche de service : Cassandra, Hive, HBase, Home Tool, etc...

Figure 7 : Architecture Kappa.



5. Panorama des systèmes de stream processing.

5.1. Apache Flink :

Apache Flink est une plate-forme open source qui fournit des capacités de traitement de flux évolutives, distribuées, tolérantes aux pannes et avec état. Flink est l'un des frameworks de traitement Big Data les plus récents et les plus innovants.

Apache Flink permet d'ingérer des données de streaming massives (jusqu'à plusieurs téraoctets) provenant de différentes sources et de les traiter de manière distribuée sur plusieurs nœuds, avant de pousser les flux dérivés vers d'autres services ou applications tels qu'Apache Kafka, les bases de données et la recherche élastique. Simplement, les éléments de base d'un pipeline Flink : entrée, traitement et sortie. Son environnement d'exécution prend en charge le traitement à faible latence à des débits extrêmement élevés d'une manière tolérante aux pannes. Les capacités de Flink permettent d'obtenir des informations en temps réel à partir de données en continu et de capacités basées sur des événements. Flink permet l'analyse de données en temps réel sur les données de streaming et s'adapte bien aux pipelines de chargement de transformation d'extraction (ETL) continus sur les données de streaming ainsi qu'aux applications pilotées par les événements.

Il donne des modèles de traitement pour les données en continu et par lots, où le modèle de traitement par lots est traité comme un cas particulier de celui en continu (c'est-à-dire un flux fini). La pile logicielle de Flink comprend les API DataStream et DataSet pour le traitement des données infinies et finies, respectivement. Flink offre plusieurs opérations sur des flux ou des ensembles de données tels que le mappage, le filtrage, le regroupement, la mise à jour de l'état, la jointure, la définition de fenêtres et l'agrégation.

Les deux principales abstractions de données de Flink sont DataStream et DataSet, elles représentent des collections d'éléments de données en lecture seule. La liste des éléments est bornée (c'est-à-dire finie) dans DataSet, alors qu'elle est illimitée (c'est-à-dire infinie) dans le cas de DataStream.

Les programmes Flink sont représentés par un graphe de flux de données (c'est-à-dire un graphe acyclique dirigé - DAG) qui est exécuté sur le cœur de Flink, qui est un moteur de flux de données en continu distribué. Les graphiques de flux de données sont composés d'opérateurs avec état et de partitions de flux de données intermédiaires. L'exécution de chaque opérateur est gérée par plusieurs instances parallèles dont le nombre est déterminé par le niveau de parallélisme. Chaque instance d'opérateur parallèle est exécutée dans un emplacement de tâche indépendant sur une machine au sein d'un groupe d'ordinateurs. La figure ci-dessous montre un exemple de graphique de flux de données pour l'application Flink.

5.2. Spark :

Spark Streaming est une extension de l'API principale de Spark qui permet un traitement de flux évolutif, à haut débit et tolérant aux pannes des flux de données en direct. Les données peuvent être ingérées à partir de nombreuses sources telles que Kafka, Kinesis ou les sockets TCP, et peuvent être traitées à l'aide d'algorithmes complexes exprimés avec des fonctions de haut niveau telles que map, reduce, join et window. Enfin, les données traitées peuvent être transférées vers des systèmes de fichiers, des bases de données et des tableaux de bord en direct. En fait, vous pouvez appliquer les algorithmes d'apprentissage automatique et de traitement de graphes de Spark sur des flux de données.

En interne, cela fonctionne comme suit. Spark Streaming reçoit des flux de données d'entrée en direct et divise les données en lots, qui sont ensuite traitées par le moteur Spark pour générer le flux final de résultats par lots.

Spark Streaming fournit une abstraction de haut niveau appelée flux discrétisé ou DStream, qui représente un flux continu de données. Les DStreams peuvent être créés soit à partir de flux de données d'entrée provenant de sources telles que Kafka et Kinesis, soit en appliquant des opérations de haut niveau sur d'autres DStreams. En interne, un DStream est représenté comme une séquence de RDD (Resilient Distributed Dataset).

5.3. Kafka :

Apache Kafka (Kafka) est une plate-forme de streaming distribuée open source qui permet (entre autres) le développement d'applications en temps réel, pilotées par des événements.

Aujourd'hui, des milliards de sources de données génèrent en continu des flux d'enregistrements de données, y compris des flux d'événements. Un événement est un enregistrement numérique d'une action qui s'est produite et de l'heure à laquelle elle s'est produite. En règle générale, un événement est une action qui entraîne une autre action dans le cadre d'un processus. Un client passant une commande, choisissant un siège sur un vol ou soumettant un formulaire d'inscription sont autant d'exemples d'événements. Un événement ne doit pas nécessairement impliquer une personne. Par exemple, le rapport d'un thermostat connecté sur la température à un moment donné est également un événement.

Ces flux offrent des opportunités aux applications qui répondent aux données ou aux événements en temps réel. Une plate-forme de streaming permet aux développeurs de créer des applications qui consomment et traitent ces flux en continu à des vitesses extrêmement élevées, avec un haut niveau de fidélité et de précision basé sur l'ordre correct de leur apparition.

LinkedIn a développé Kafka en 2011 en tant que courtier de messages à haut débit pour son propre usage, puis a ouvert et donné Kafka à Apache Software Foundation. Aujourd'hui, Kafka est devenue la plate-forme de streaming la plus utilisée, capable d'ingérer et de traiter des milliards d'enregistrements par jour sans aucun décalage de performance perceptible à mesure que les volumes évoluent. Des organisations du Fortune 500 telles que Target, Microsoft, AirBnB et Netflix s'appuient sur Kafka pour offrir à leurs clients des expériences en temps réel basées sur les données.

Kafka a trois capacités principales :

- Il permet aux applications de publier ou de s'abonner à des flux de données ou d'événements.
- Il stocke les enregistrements avec précision (c'est-à-dire dans l'ordre dans lequel ils se sont produits) d'une manière tolérante aux pannes et durable.
- Il traite les enregistrements en temps réel (au fur et à mesure qu'ils se produisent). Les développeurs peuvent tirer parti de ces fonctionnalités de Kafka via quatre API :

API Producteur : cela permet à une application de publier un flux dans un sujet Kafka. Une rubrique est un journal nommé qui stocke les enregistrements dans l'ordre dans lequel ils se sont produits les uns par rapport aux autres. Une fois qu'un enregistrement a été écrit dans un sujet, il ne peut pas être modifié ou supprimé ; au lieu de cela, il reste dans le sujet pendant une durée préconfigurée (par exemple, pendant deux jours) ou jusqu'à ce que l'espace de stockage soit épuisé.

API consommateur : cela permet à une application de s'abonner à un ou plusieurs sujets et d'ingérer et de traiter le flux stocké dans le sujet. Il peut travailler avec des enregistrements dans le sujet en temps réel, ou il peut ingérer et traiter des enregistrements antérieurs.

API Streams : elle s'appuie sur les API Producteur et Consommateur et ajoute des capacités de traitement complexes qui permettent à une application d'effectuer un traitement continu de flux d'avant en arrière, en particulier pour consommer des enregistrements d'un ou plusieurs sujets, pour les analyser, les agréger ou les transformer selon les besoins, et de publier les flux résultants sur les mêmes sujets ou d'autres sujets. Alors que les API Producer et Consumer peuvent être utilisées pour un traitement de flux simple, c'est l'API Streams qui permet le développement d'applications de streaming de données et d'événements plus sophistiqués.

API de connecteur : cela permet aux développeurs de créer des connecteurs, qui sont des producteurs ou des consommateurs réutilisables qui simplifient et automatisent l'intégration d'une source de données dans un cluster Kafka. Découvrez quelques connecteurs prêts à l'emploi pour les magasins de données populaires.

5.4. Amazon Kinesis :

Amazon Kinesis est un service fourni par Amazon Web Service qui permet aux utilisateurs de traiter une grande quantité de données (qui peuvent être de l'audio, de la vidéo, des journaux d'application, des flux de clics sur le site Web et de la télémétrie IoT) par seconde en temps réel. Dans le scénario d'aujourd'hui, la gestion d'une grande quantité de données devient très importante et pour cela, il existe un sujet complet connu sous le nom de Big Data qui travaille sur la façon de traiter ou de gérer les flux de grandes quantités de données. Amazon propose donc une solution connue sous le nom d'Amazon Kinesis qui est entièrement gérée et automatisée et peut gérer facilement les grands flux de données en temps réel. Il permet aux utilisateurs de collecter, stocker, capturer et traiter une grande quantité de journaux à partir des flux distribués tels que les flux de médias sociaux. Il permet aux utilisateurs de se concentrer sur le développement en prenant n'importe quelle quantité de données de n'importe quelle source pour la traiter. Et après avoir traité toutes les données, Kinesis distribue également toutes les données aux consommateurs simultanément.

Il y a trois composants clés sur lesquels kinesis fonctionne sont les suivants :

Kinesis Firehose :

Firehose permet aux utilisateurs de charger ou de transformer leurs flux de données en un dernier transfert de service Web Amazon pour les autres fonctionnalités telles que l'analyse ou le stockage. Il ne nécessite pas de gestion continue car il est entièrement automatisé et évolue automatiquement en fonction des données.

Kinesis Analytics :

Il permet aux flux de données fournis par les flux kinesis firehose et kinesis de les analyser et de les traiter avec le SQL standard. Il analyse le format des données et analyse

automatiquement les données et en utilisant un éditeur de schéma interactif standard pour le modifier dans le schéma recommandé. Il fournit également des modèles de processus de flux prédéfinis qui peuvent être utilisés pour sélectionner un modèle approprié pour leurs analyses de données.

Kinesis streams :

Il fournit une plate-forme pour le traitement en temps réel et continu des données. Il est également utilisé pour chiffrer les données sensibles en utilisant les clés principales KMS et le chiffrement côté serveur à des fins de sécurité.

Conclusion

Les architectures de traitement de données traditionnelles sont très complexes et nécessitent souvent différentes technologies, par exemple, la technologie ETL est nécessaire pour extraire des données de différents entrepôts de données et les transformer en un seul entrepôt de données ou lac de données, et différentes méthodes doivent utiliser des processeurs par lots tels que Map- reduce. Convertir les données dans un format utile lors de l'analyse.

Pour résoudre le problème en temps réel du traitement des paquets, divers cadres de traitement des files d'attente de messages ont été intégrés pour combiner les données par paquets avec un ensemble de données en temps réel en fonction d'exigences spécifiques en temps réel.

À mesure que les activités des organisations se développent, la gestion de gros volumes de données devient essentielle pour atteindre l'efficacité souhaitée. Le processus de streaming permet aux parties prenantes et à la direction de gérer leurs données de la meilleure façon possible.

Bibliographie :

X. Liu and R. Buyya, "Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy, Review, and Future Directions," ACM Computing Surveys, vol. 53, (3), pp. 1-41, 2020;2021;.

X. Wei et al, "Reliable stream data processing for elastic distributed stream processing systems," Cluster Computing, vol. 23, (2), pp. 555-574, 2020.

Y. Busnel, "Panorama des modèles de flux de données à large échelle," 2020.