Big Data Streaming

Définition et Enjeux

Big Streaming Data désigne le traitement continu de très grands volumes de données générées en temps réel. Parmi les applications typiques, on trouve :

- La détection de fraudes (ex. transactions bancaires),
- La détection d'anomalies (ex. données issues de capteurs),
- L'analyse de tendances (ex. réseaux sociaux).

Ces applications, souvent regroupées sous le terme *streaming analytics* (analyse de flux), reposent sur l'exécution de requêtes continues sur des données "en mouvement".

Streaming Analytics et Comparaison avec le Batch Processing

La **streaming analytics** consiste à :

- Mesurer des taux d'événements,
- Calculer des statistiques glissantes (moyennes, histogrammes, etc.),
- Détecter des tendances en comparant les statistiques actuelles aux historiques,
- Filtrer et nettoyer les données en temps réel.

Comparativement, le **batch processing** traite des ensembles de données statiques. Les principales différences sont :

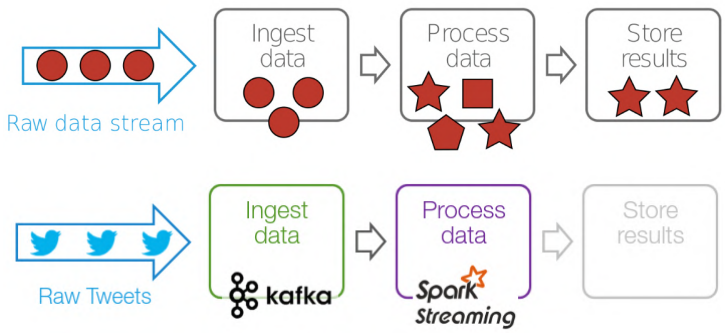
	Batch processing	**Stream processing**
Jeux de données statiques	✓	(✓)
Débits élevés	✓	×
Analyse complexe	✓	×
Itérations multiples	✓	×
Gestion des données en temps réel	×	✓
Mise à jour continue des résultats	×	✓
Faible latence (entre production et résultat)	×	✓

Architecture de Traitement des Flux de Données

Le traitement de Big Streaming Data se décompose en trois étapes clés :

1. **Ingestion** : Réception et tamponnement des flux de données à l'aide de systèmes spécialisés.
2. **Traitement** : Nettoyage, extraction et transformation des données.
3. **Stockage** : Sauvegarde des données transformées pour une utilisation ultérieure (analyses, dashboards, etc.).

Chaque étape nécessite une infrastructure adaptée et souvent distribuée.

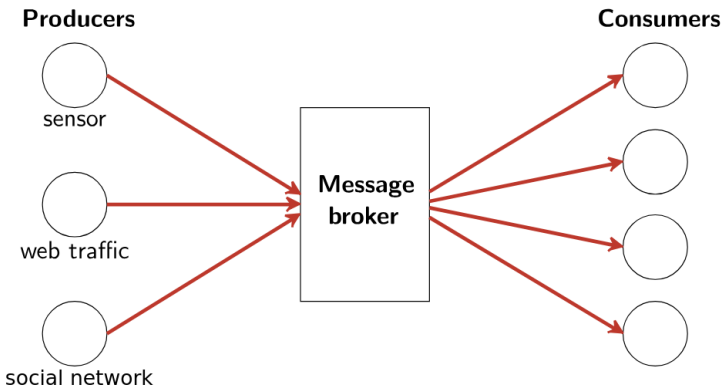


Message Brokers

Définition

Les **message brokers** sont des systèmes intermédiaires qui facilitent la communication entre producteurs et consommateurs de données. Dans ce paradigme :

- Un **Producteur** (publisher) génère et envoie des messages.
- Un **Consommateur** (subscriber) reçoit et traite ces messages.
- Le **broker** traduit et achemine les messages entre ces entités.



Défis des Message Brokers

Les principaux défis rencontrés sont :

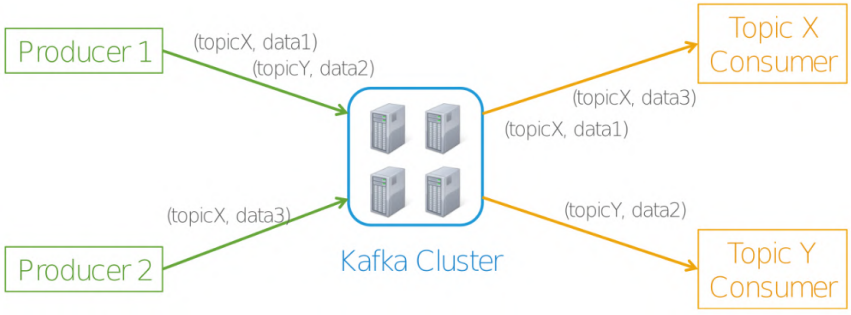
- **Routage** : Certains consommateurs n'ont besoin que d'un sous-ensemble de messages, et un message peut être destiné à plusieurs consommateurs.
- **Performance** : La production de données peut être très rapide et en grand volume, nécessitant une haute capacité de traitement.
- **Tolérance aux pannes** : Les clients se connectent et se déconnectent, et le système doit gérer ces variations sans perte de données.

Apache Kafka

Présentation Générale

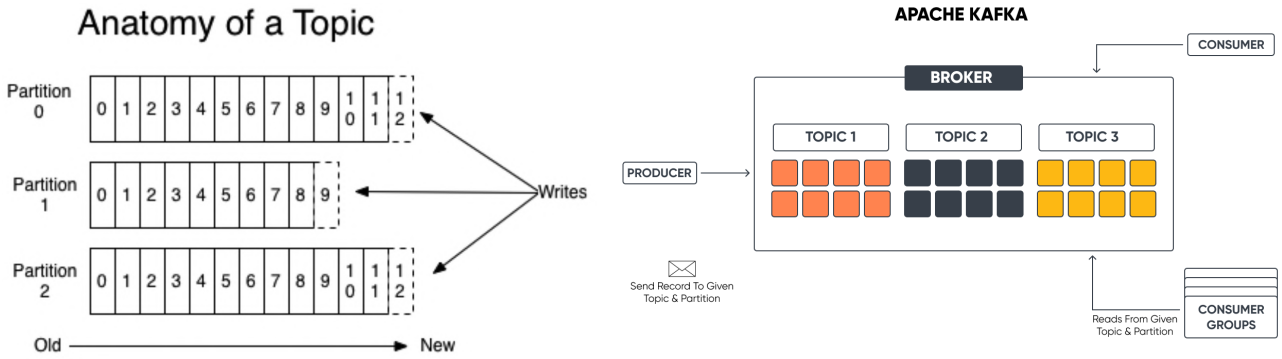
Apache Kafka est une plateforme de streaming d'événements distribuée, open-source, initialement développée chez LinkedIn. Ses caractéristiques incluent :

- La gestion d'un cluster de brokers basés sur un système de log.
- L'utilisation du modèle Publish/Subscribe.
- Des API dédiées pour les producteurs, consommateurs et pour le traitement de flux.
- Une adoption par des entreprises majeures (Spotify, Netflix, LinkedIn, Uber, etc.) [**?**].



Concepts Clés : Producteurs, Consommateurs, Thèmes et Partitions

- **Thème (Topic)** : Catégorie ou flux de messages. Chaque thème est divisé en partitions.
- **Partition** : Séquence ordonnée et numérotée de messages. Le producteur choisit la partition pour chaque message et le consommateur choisit celle depuis laquelle lire.
- **Producteurs et Consommateurs** : Les producteurs publient des messages sous un thème, tandis que les consommateurs s'y abonnent pour recevoir les données.



Intégration avec Spark Streaming

Kafka joue un rôle crucial lors de l'ingestion des données. Il reçoit et tamponne les messages qui seront ensuite traités par des systèmes comme Spark Streaming.

Spark Streaming

Définition et Fonctionnement

Spark Streaming est une extension de l'API Spark permettant le traitement de flux en temps réel. Ses points forts :

- Traitement scalable et tolérant aux pannes.
- Intégration facile avec les autres composants Spark (Spark SQL, MLlib, GraphX, etc.) [**?**].
- Traitement des données en **micro-batch** grâce à un découpage temporel.

Modèle de Programmation : RDDs et DStreams

- **RDD (Resilient Distributed Dataset)** : Collection distribuée et partitionnée d'objets. Les transformations sur les RDDs sont évaluées de manière paresseuse.
- **DStream (Discretized Stream)** : Représente un flux de données continu sous forme de séquence infinie de RDDs. Les DStreams permettent d'appliquer des transformations en continu sur les données issues de diverses sources (Kafka, Flume, etc.).

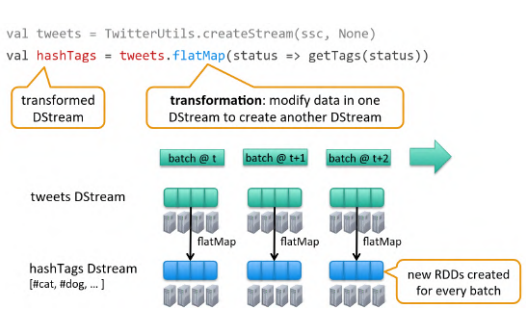
Exemple d'Application : Extraction des Hashtags sur Twitter

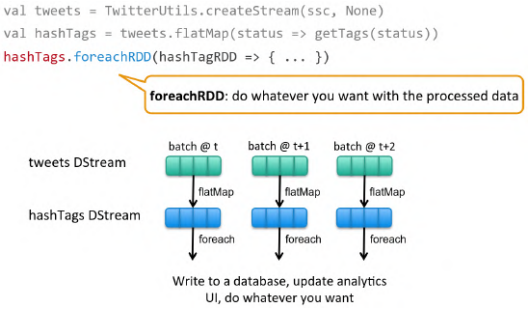
Un exemple classique d'utilisation de Spark Streaming consiste à extraire les hashtags d'un flux Twitter :

1. Création d'un `StreamingContext` avec un intervalle de batch (ex. 1 seconde).
2. Connexion à l'API Twitter via `TwitterUtils.createStream`.
3. Application d'une transformation `flatMap` pour extraire les hashtags.
4. Traitement des données avec `foreachRDD` ou en sauvegardant directement dans un système de stockage (ex. HDFS).

Ce processus permet de convertir un flux continu de tweets en micro-batchs traitables par le moteur Spark. Une fois la configuration terminée, on démarre le processus de streaming avec ssc.start()

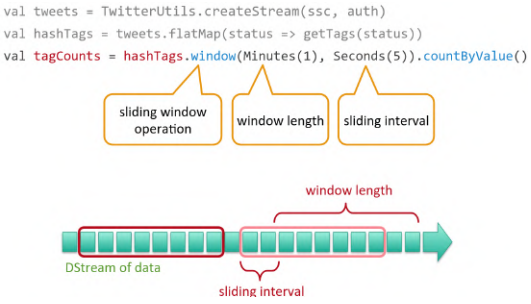
Hugo Tondenier





Transformations Basées sur des Fenêtres

Spark Streaming offre la possibilité d'appliquer des transformations sur des fenêtres temporelles glissantes. Par exemple, on peut compter le nombre d'occurrences d'un hashtag sur une fenêtre d'une minute avec un intervalle de glissement de 5 secondes. Ces opérations facilitent l'analyse de tendances sur des périodes courtes.



Architecture Interne

- **Receiver** : Bufferise les données reçues (ex. tweets) dans la mémoire des *executors*.
- **Driver** : Lance des tâches pour traiter les données contenues dans les RDDs issus des DStreams.
- **Executors** : Effectuent le traitement en parallèle sur les données réparties.