Introduction au Big Data avec Spark et Hadoop

Module 6 Glossaire : Surveillance et réglage

Bienvenue ! Ce glossaire alphabétique contient de nombreux termes utilisés dans ce cours. Ce glossaire complet comprend également des termes supplémentaires reconnus par l'industrie qui ne sont pas utilisés dans les vidéos de cours. Ces termes sont essentiels pour que vous puissiez les reconnaître lorsque vous travaillez dans l'industrie, participez à des groupes d'utilisateurs et à d'autres programmes de certification professionnelle.

Temps de lecture estimé : 5 minutes

Terme	Définition
Interface utilisateur d'Apache Spark	Fournit des informations précieuses, organisées sur plusieurs onglets, sur l'application en cours d'exécution. L' onglet **Tâches** affiche toutes les tâches de l'application, y compris leur état. L' onglet **Étapes** indique l'état des tâches au sein d'une étape. L' onglet **Stockage** indique la taille de tous les RDD ou DataFrames qui ont persisté dans la mémoire ou le disque. L' onglet **Environnement** inclut toutes les variables d'environnement et les propriétés système pour Spark ou la JVM. L' onglet **Exécuteur** affiche un résumé qui indique l'utilisation de la mémoire et du disque pour tous les exécuteurs utilisés pour l'application. Des onglets supplémentaires s'affichent en fonction du type d'application utilisé.
Problèmes de dépendance des applications	Les applications Spark peuvent avoir de nombreuses dépendances, notamment des fichiers d'application tels que des fichiers de script Python, des fichiers JAR Java et même des fichiers de données requis. Les applications dépendent des bibliothèques utilisées et de leurs dépendances. Les dépendances doivent être rendues disponibles sur tous les nœuds du cluster, soit par préinstallation, y compris les dépendances dans le script Spark-submit fourni avec l'application, soit sous forme d'arguments supplémentaires.
Problèmes de ressources d'application	Les cœurs de processeur et la mémoire peuvent devenir un problème si une tâche se trouve dans la file d'attente de planification et que les travailleurs disponibles n'ont pas suffisamment de ressources pour exécuter les tâches. Lorsqu'un travailleur termine une tâche, le processeur et la mémoire sont libérés, ce qui permet la planification d'une autre tâche. Cependant, si l'application demande plus de ressources qui peuvent devenir disponibles, les tâches peuvent ne jamais être exécutées et finir par expirer. De même, supposons que les exécuteurs exécutent de longues tâches qui ne se terminent jamais. Dans ce cas, leurs ressources ne deviennent jamais disponibles, ce qui entraîne également l'impossibilité d'exécuter les tâches futures, ce qui entraîne une erreur de dépassement de délai. Les utilisateurs peuvent facilement accéder à ces erreurs lorsqu'ils consultent l'interface utilisateur ou les journaux d'événements.
ID d'application	Un identifiant unique que Spark attribue à chaque application. Ces fichiers journaux apparaissent pour chaque processus d'exécution et de pilote exécuté par l'application.
Validation des données	La pratique consistant à vérifier l'intégrité, la qualité et l'exactitude des données utilisées dans les applications Spark ou les flux de travail de traitement des données. Ce processus de validation comprend la vérification des données pour détecter des problèmes tels que des valeurs manquantes, des valeurs aberrantes ou des erreurs de format de données. La validation des données est essentielle pour garantir que les données traitées dans les applications Spark sont fiables et adaptées à l'analyse ou au traitement ultérieur. Diverses techniques et bibliothèques, telles que l'API DataFrame d'Apache Spark ou des outils externes, peuvent être utilisées pour effectuer des tâches de validation des données dans les environnements Spark.
Graphe acyclique dirigé (DAG)	Représentation conceptuelle d'une série d'activités. Un graphique illustre l'ordre des activités. Il se présente visuellement comme un ensemble de cercles, chacun représentant une activité, certains reliés par des lignes, représentant le flux d'une activité à une autre.
Mémoire du conducteur	Désigne l'allocation de mémoire désignée pour le programme pilote d'une application Spark. Le programme pilote sert de coordinateur central des tâches, gérant la distribution et l'exécution des tâches Spark sur les nœuds du cluster. Il contient le flux de contrôle de l'application, les métadonnées et les résultats des transformations et actions Spark. La capacité de la mémoire du pilote est un facteur critique qui a un impact sur la faisabilité et les performances des applications Spark. Elle doit être configurée avec soin pour garantir une exécution efficace des tâches sans problèmes liés à la mémoire.
Onglet Environnement	Contient plusieurs listes pour décrire l'environnement de l'application en cours d'exécution. Ces listes incluent les propriétés de configuration Spark, les profils de ressources, les propriétés pour Hadoop et les propriétés système actuelles.
Mémoire de l'exécuteur	Utilisé pour le traitement. Si la mise en cache est activée, une mémoire supplémentaire est utilisée. Une mise en cache excessive entraîne des erreurs de manque de mémoire.
Onglet Exécuteurs	Composant de certains outils et infrastructures informatiques distribués utilisés pour gérer et surveiller l'exécution des tâches au sein d'un cluster. Il présente généralement un tableau récapitulatif en haut qui affiche les mesures pertinentes pour les exécuteurs actifs ou terminés. Ces mesures peuvent inclure des statistiques liées aux tâches, l'entrée et la sortie de données, l'utilisation du disque et l'utilisation de la mémoire. Sous le tableau récapitulatif, l'onglet répertorie tous les exécuteurs individuels qui ont participé à l'application ou au travail, ce qui peut inclure le pilote principal. Cette liste fournit souvent des liens pour accéder aux messages de journal de sortie standard (stdout) et d'erreur standard (stderr) associés à chaque processus d'exécuteur. L'onglet Exécuteurs constitue une ressource précieuse pour les administrateurs et les opérateurs afin d'obtenir des informations sur les performances et le comportement des exécuteurs de cluster pendant l'exécution des tâches.
Détails du poste	Provides information about the different stages of a specific job. The timeline displays each stage, where the user can quickly see the job's timing and duration. Below the timeline, completed stages are displayed. In the parentheses beside the heading, users will see a quick view that displays the number of completed stages. Then, view the list of stages within the job and job metrics, including when the job was submitted, input or output sizes, the number of attempted tasks, the number of succeeded tasks, and how much data was read or written because of a shuffle.
Jobs tab	Commonly found in Spark user interfaces and monitoring tools, it offers an event timeline that provides key insights into the execution flow of Spark applications. This timeline includes crucial timestamps such as the initiation times of driver and executor processes, along with the creation timestamps of individual jobs within the application. The Jobs tab serves as a valuable resource for monitoring the chronological sequence of events during Spark job execution.
Multiple related jobs	Spark application can consist of many parallel and often related jobs, including multiple jobs resulting from multiple data sources, multiple DataFrames, and the actions applied to the DataFrames.
Parallelization	Parallel regions of program code executed by multiple threads, possibly running on multiple processors. Environment variables determine the number of threads created and calls to library functions.
Parquet	A columnar format that is supported by multiple data processing systems. Spark SQL allows reading and writing data from Parquet files, and Spark SQL preserves the data schema.

Terme	Définition
Serialization	Required to coordinate access to resources that are used by more than one program. An example of why resource serialization is needed occurs when one program is reading from a data set and another program needs to write to the data set.
Spark data persistence	Also known as caching data in Spark. Ability to store intermediate calculations for reuse. This is achieved by setting persistence in either memory or both memory and disk. Once intermediate data is computed to generate a fresh DataFrame and cached in memory, subsequent operations on the DataFrame can utilize the cached data instead of reloading it from the source and redoing previous computations. This feature is crucial for accelerating machine learning tasks that involve multiple iterations on the same data set during model training.
Spark History server	Web UI where the status of running and completed Spark jobs on a provisioned instance of Analytics Engine powered by Apache Spark is displayed. If users want to analyze how different stages of the Spark job are performed, they can view the details in the Spark history server UI.
Spark memory management	Spark memory stores the intermediate state while executing tasks such as joining or storing broadcast variables. All the cached and persisted data will be stored in this segment, specifically in the storage memory.
Spark RDD persistence	Optimization technique that saves the result of RDD evaluation in cache memory. Using this technique, the intermediate result can be saved for future use. It reduces the computation overhead.
Spark standalone	Here, the resource allocation is typically based on the number of available CPU cores. By default, a Spark application can occupy all the cores within the cluster, which might lead to resource contention if multiple applications run simultaneously. In the context of the standalone cluster manager, a ZooKeeper quorum is employed to facilitate master recovery through standby master nodes. This redundancy ensures high availability and fault tolerance in the cluster management layer. Additionally, manual recovery of the master node can be achieved using file system operations in the event of master failure, allowing system administrators to intervene and restore cluster stability when necessary.
SparkContext	When a Spark application is being run, as the driver program creates a SparkContext, Spark starts a web server that serves as the application user interface. Users can connect to the UI web server by entering the hostname of the driver followed by port 4040 in a browser once that application is running. The web server runs for the duration of the Spark application, so once the SparkContext stops, the server shuts down, and the application UI is no longer accessible.
Stages tab	Displays a list of all stages in the application, grouped by the current state of either completed, active, or pending. This example displays three completed stages. Click the Stage ID Description hyperlinks to view task details for that stage.
Storage tab	Displays details about RDDs that have been cached or persisted to memory and written to disk.
Unified memory	Unified regions in Spark shared by executor memory and storage memory. If executor memory is not used, storage can acquire all the available memory and vice versa. If the total storage memory usage falls under a certain threshold, executor memory can discard storage memory. Due to complexities in implementation, storage cannot evict executor memory.
User code	Made up of the driver program, which runs in the driver process, and the functions and variables serialized that the executor runs in parallel. The driver and executor processes run the application user code of an application passed to the Spark-submit script. The user code in the driver creates the SparkContext and creates jobs based on operations for the DataFrames. These DataFrame operations become serialized closures sent throughout the cluster and run on executor processes as tasks. The serialized closures contain the necessary functions, classes, and variables to run each task.
Workflows	Inclure les tâches créées par SparkContext dans le programme du pilote. Les tâches en cours s'exécutent en tant que tâches dans les exécuteurs et les tâches terminées transfèrent les résultats vers le pilote ou écrivent sur le disque.

Auteur(s)

- Niha Ayaz Sultan