Adriel Araya Vargas 2019312845 Instituto Tecnológico de Costa Rica Bases de Datos 2 IC-4302

Apache Spark: A Unified Engine for Big Data Processing

El crecimiento exponencial de datos generados por aplicaciones modernas ha creado la necesidad de motores de procesamiento más potentes y eficientes. Apache Spark es un motor unificado para el procesamiento de big data que proporciona una plataforma escalable y tolerante a fallos para procesar grandes conjuntos de datos. En este artículo, se exploran las diversas características y funcionalidades del motor Spark, destacando su modelo de programación unificado y de propósito general, sus bibliotecas de alto nivel y sus aplicaciones.

Spark es un marco de propósito general, lo que significa que es capaz de manejar una amplia variedad de cargas de trabajo de procesamiento de datos tales como el procesamiento de lotes, el procesamiento de streaming, el machine learning y el procesamiento de grafos. Esto facilita el manejo de Big Data y la creación de aplicaciones.

Modelo de Programación

El modelo de programación de Apache Spark se basa en una colección distribuida de objetos particionadas a través de un clúster que pueda manipular en paralelo, esta se denomina RDD (Resilient Distributed Datasets). Spark expone los RDD a través de programación funcional de interfaz de aplicación donde los usuarios pueden pasar funciones locales a correr en el clúster. Los RDD se crean a través de operaciones llamadas "transformaciones" en los datos de entrada. Las transformaciones en los RDD se evalúan de forma perezosa, lo que significa que Spark solo calcula el resultado de una transformación cuando es requerido por una acción. Este enfoque de evaluación perezosa ayuda a optimizar el rendimiento del marco al minimizar el movimiento de datos entre nodos en el clúster. Los RDDs también proporcionan soporte para compartir datos entre computaciones, permitiendo a los usuarios persistir ciertos RDDs en memoria para su reutilización rápida. El compartir datos provee de aumentos en velocidad sumamente significativo que pone a Spark encima de la competencia.

RDDs también se recuperan automáticamente de los fallos. Spark utiliza un enfoque diferente llamado "lineage". Cada RDD sigue el gráfico de transformaciones que se usó para construirlo y vuelve a ejecutar estas operaciones en los datos base para reconstruir cualquier partición perdida. Este método es más eficiente que la replicación de datos, ya que ahorra tiempo y espacio de almacenamiento en memoria.

Librerías de Alto Nivel.

A través del modelo de programación RDD, se han construido bibliotecas de nivel superior en Spark para abordar casos de uso específicos de motores de cómputo especializados. A continuación se abordan las diferentes librerías y sus funciones.

Spark SQL es una biblioteca de Apache Spark que implementa consultas relacionales utilizando técnicas similares a las de las bases de datos analíticas, como el almacenamiento columnar y la optimización basada en costos. Spark SQL también proporciona el marco de trabajo de DataFrames que permite trabajar con datos tabulares con una estructura de conocida de Python y R.

Spark Streaming implementa procesamiento incremental de flujos usando un modelo llamado "flujos discretizados". El sistema divide los datos de entrada en pequeños lotes que se combinan regularmente con el estado almacenado dentro de RDD para producir nuevos resultados. Esta técnica tiene varias ventajas como una recuperación de fallos menos costosa y la capacidad de combinar la transmisión de datos con consultas por lotes e interactivas.

GraphX proporciona una interfaz de cómputo de gráficos similar a Pregel y GraphLab, con optimizaciones de colocación implementadas iguales a las de estos sistemas.

MLib es la librería de machine learning para Spark con más de 50 algoritmos para modelos distribuídos de aprendizaje.

Las librerías de Spark facilitan la combinación de aplicaciones debido a que pasan su información entre ellas rápidamente. Además, por la optimización a base de los RDDs se corren todas estas librerías en el mismo motor sin perder rendimiento que proveen los motores especializados.

Aplicaciones.

Spark es usado en un largo rango de aplicaciones, Existen varios casos de uso de Spark que se presentan en la siguiente sección.

Spark es utilizado comúnmente para procesamiento en lotes en grandes conjuntos de datos, incluyendo la transformación de datos de formato crudo a estructurado y el entrenamiento de modelos de aprendizaje automático fuera de línea. También es usado el procesamiento de streaming para analíticas en tiempo real.

Spark hace uso de queries de manera interactiva para consultas relacionales con Spark SQL, a menudo a través de herramientas de inteligencia empresarial como Tableau. Además, los desarrolladores y científicos de datos pueden utilizar las interfaces de Scala, Python y R de Spark de forma interactiva.

Este también es usado en dominios científicos para procesamiento de datos genomicos o imagenes. En resumen, el despliegue de aplicaciones compatibles con Spark se ha diversificado y crecido substancialmente desde su creación.

¿Porqué es el modelo de Spark general?

En esta sección se discute la generalidad y limitaciones de los RRDS. Se entienden RDDs desde dos perspectivas. Desde una perspectiva de expresividad, los autores argumentan que los RDD pueden emular cualquier computación distribuida de manera eficiente en muchos casos, excepto cuando la computación es sensible a la latencia de la red. Se compara los RDD con el modelo MapReduce, en el que se basan los RDD, y señalan que MapReduce puede emular cualquier cálculo distribuido al dividir su trabajo en intervalos de tiempo.

Desde una perspectiva de sistemas, los autores muestran que los RDD brindan a las aplicaciones control sobre los recursos de cuello de botella más comunes en los clústeres, como la red y la E/S de almacenamiento. Los autores señalan que Spark puede ejecutar los mismos algoritmos y bibliotecas que se usan en sistemas especializados en cada nodo para solucionar los problemas de cuello de botella en el tiempo de la CPU.

Spark puede incurrir en costos adicionales en comparación con algunos sistemas especializados debido a la tolerancia a fallas, pero se argumenta que muchas

aplicaciones están limitadas por operaciones de E/S y, más allá de esta operación, las optimizaciones agregan solo un beneficio sencillo.