CTIC UNI
CENTRO DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES
UNIVERSIDAD NACIONAL DE INGENIERÍA

CIENCIA DE DATOS:

APRENDE LOS FUNDAMENTOS DE MANERA PRÁCTICA

SESION 01

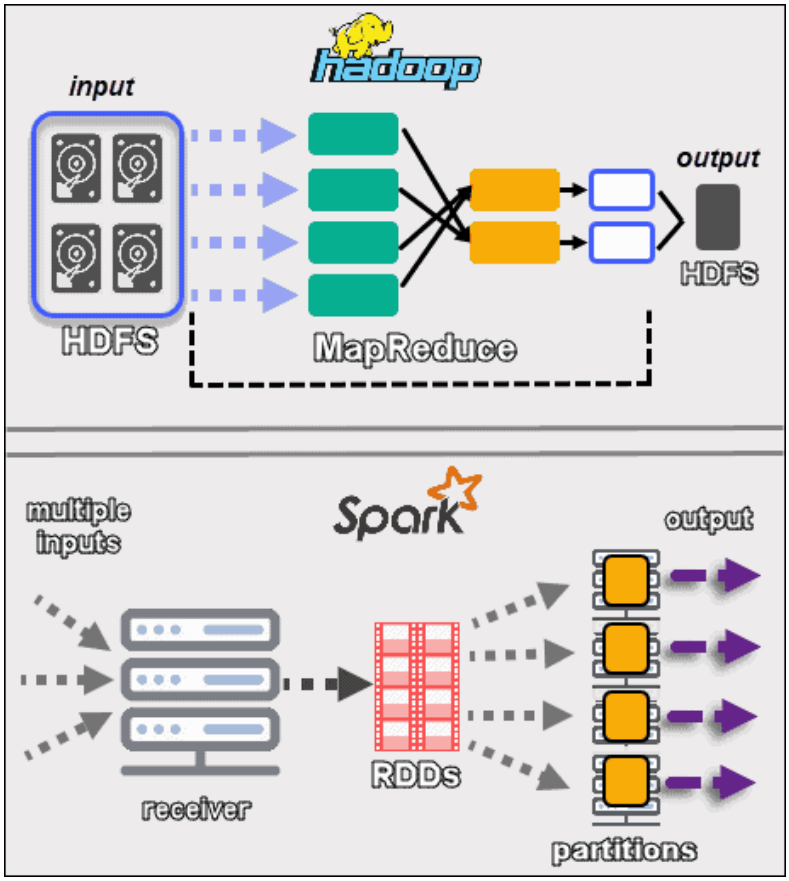
BIG DATA

Juan Chipoco
mindquasar@gmail.com

*Las imágenes y graficos no me pertenecen y son usados solo con fines educativos

ÍNDICE

OBJETIVO



La disrupción tecnológica y los modelos derivados, han creado una especialidad que cada vez cobra más relevancia en el mundo moderno: La Ciencia de Datos. El Data Scientist es un profesional capaz de analizar realidades complejas y resolver problemas de negocio a través de la explotación de grandes volúmenes de datos, siendo capaz de ser el puente y traductor entre las áreas técnicas de la empresa y la alta gerencia.

La definición de Big Data es que son datos que contienen una mayor variedad, que llegan en volúmenes crecientes y con más velocidad. Esto también se conoce como las tres Vs. En pocas palabras, los grandes datos son conjuntos de datos más grandes y complejos, especialmente de nuevas fuentes de datos.

Los datos masivos tienen varias características además de ser muy abundantes, entre ellas, heterogeneidad, complejidad, desestructuración, falta de completitud, y tener potencial para ser erróneos. Por esta razón, al diseñar procesos para gestionar datos, debemos tener en cuenta todos estos aspectos con el objetivo de garantizar y preservar su calidad, así como la extracción de información útil y sin errores, lo cual garantizará la fiabilidad de los datos y, por ende del análisis resultante.

Hadoop y Spark, ambos desarrollados por Apache Software Foundation, son frameworks de código abierto ampliamente utilizados para arquitecturas de big data. Cada marco contiene un extenso ecosistema de tecnologías de código abierto que preparan, procesan, administran y analizan grandes conjuntos de datos.

Apache Hadoop es una utilidad de software de código abierto que permite a los usuarios administrar grandes conjuntos de datos (desde gigabytes hasta petabytes) al habilitar una red de computadoras (o "nodos") para resolver problemas de datos extensos e intrincados. Es una solución altamente escalable y rentable que almacena y procesa datos estructurados, semiestructurados y no estructurados (por ejemplo, registros de flujo de clics de Internet, registros de servidores web, datos de sensores de IoT, etc.).

Apache Spark, que también es de código abierto, es un motor de procesamiento de datos para grandes conjuntos de datos. Al igual que Hadoop, Spark divide tareas grandes en diferentes nodos. Sin embargo, tiende a funcionar más rápido que Hadoop y utiliza memoria de acceso aleatorio (RAM) para almacenar en caché y procesar datos en lugar de un sistema de archivos. Esto permite que Spark maneje casos de uso que Hadoop no puede.

INTRODUCCION – BIG DATA

¿Qué son los datos?

Las cantidades, caracteres o símbolos sobre los que una computadora realiza operaciones, que pueden almacenarse y transmitirse en forma de señales eléctricas y grabarse en medios de grabación magnéticos, ópticos o mecánicos.

¿Qué es Big Data?

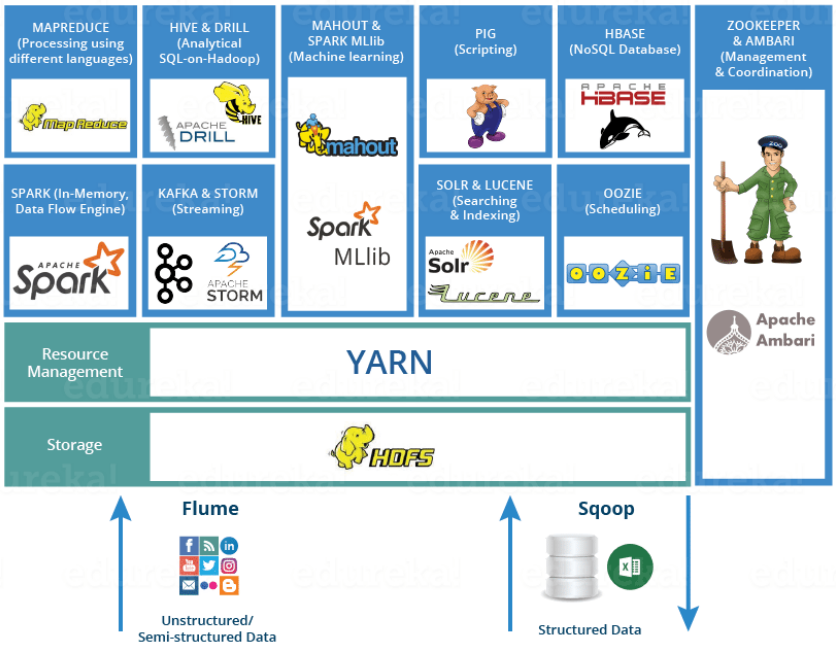
Big Data es una colección de datos que es enorme en volumen, pero que crece exponencialmente con el tiempo. Se trata de datos con un tamaño y una complejidad tan grandes que ninguna de las herramientas tradicionales de gestión de datos puede almacenarlos o procesarlos de manera eficiente. Big data también es un dato pero con un tamaño enorme.

¿Cuales son ejemplos de Big Data?

Los siguientes son algunos de los ejemplos de Big Data:

- La Bolsa de Valores de Nueva York es un ejemplo de Big Data que genera alrededor de un terabyte de nuevos datos comerciales por día.
- La estadística muestra que más de 500 terabytes de datos nuevos se introducen en las bases de datos del sitio de redes sociales Facebook todos los días. Estos datos se generan principalmente en términos de carga de fotos y videos, intercambio de mensajes, comentarios, etc.
- Un solo motor Jet puede generar más de 10 terabytes de datos en 30 minutos de tiempo de vuelo. Con muchos miles de vuelos por día, la generación de datos alcanza muchos Petabytes.

INTRODUCCIÓN – HADOOP



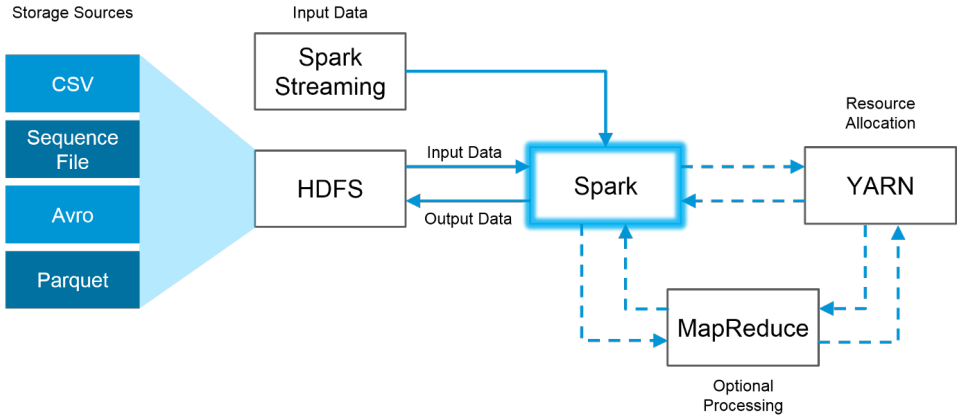
¿Qué es Hadoop?

Apache Hadoop es un framework de código abierto que se utiliza para desarrollar aplicaciones de procesamiento de datos que se ejecutan en un entorno informático distribuido.

Las aplicaciones creadas con HADOOP se ejecutan en grandes conjuntos de datos distribuidos en grupos de computadoras básicas. Las computadoras comerciales son baratas y ampliamente disponibles. Estos son principalmente útiles para lograr una mayor potencia computacional a bajo costo.

Al igual que los datos que residen en un sistema de archivos local de un sistema informático personal, en Hadoop, los datos residen en un sistema de archivos distribuido que se denomina sistema de archivos distribuidos de Hadoop. El modelo de procesamiento se basa en el concepto de "localidad de datos" en el que la lógica computacional se envía a los nodos del clúster (servidor) que contienen datos. Esta lógica computacional no es más que una versión compilada de un programa escrito en un lenguaje de alto nivel como Java. Dicho programa procesa los datos almacenados en Hadoop HDFS.

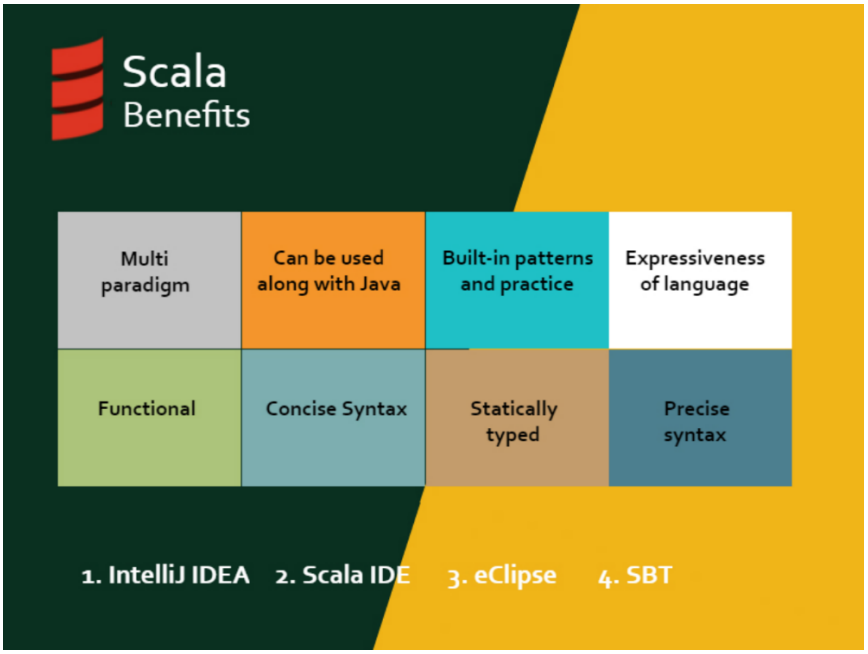
INTRODUCCIÓN – SPARK



Apache Spark es un motor de computación unificado y un conjunto de bibliotecas para el procesamiento de datos en paralelo en clústeres de computadoras. Al momento de escribir este manual, Spark es el motor de código abierto más activamente desarrollado para esta tarea; convirtiéndolo en la herramienta de facto para cualquier desarrollador o científico de datos interesado en Big Data. Spark admite múltiples lenguajes de programación ampliamente utilizados (Python, Java, Scala y R), incluye bibliotecas para diversas tareas que van desde SQL hasta transmisión y aprendizaje automático, y se ejecuta en cualquier lugar, desde una computadora portátil hasta un clúster de miles de servidores. Esto lo convierte en un sistema fácil para comenzar y escalar hasta el procesamiento de Big Data a una escala increíblemente grande.

Spark proporciona API sencillas para trabajar con una gran cantidad de datos, mientras que los usuarios finales apenas necesitan saber sobre la gestión de tareas y recursos en todas las máquinas, todo esto y mas hace a Spark una buena eleccion.

INTRODUCCIÓN – SCALA



Scala es un lenguaje de programación multiparadigma, de alto nivel y de propósito general. Es un lenguaje de programación puro orientado a objetos que también brinda soporte al enfoque de programación funcional.

No existe el concepto de datos primitivos ya que todo es un objeto en Scala. Está diseñado para expresar los patrones generales de programación de una manera refinada, sucinta y segura. Los programas de Scala se pueden convertir a bytecodes y se pueden ejecutar en la JVM (Java Virtual Machine).

Scala significa lenguaje escalable. También proporciona los tiempos de ejecución de Javascript. Scala está muy influenciado por Java y algunos otros lenguajes de programación como Lisp, Haskell, Pizza, etc.

¿Por qué Scala?

Scala tiene muchas razones para ser popular entre los programadores. Algunas de las razones son:

- *Fácil para comenzar:* Scala es un lenguaje de alto nivel, por lo que está más cerca de otros lenguajes de programación populares como Java, C, C++. Por lo tanto, se vuelve muy fácil aprender Scala para cualquier persona. Para los programadores de Java, Scala es más fácil de aprender.

- *Contiene las mejores funciones:* Scala contiene las funciones de diferentes lenguajes como C, C ++, Java, etc., lo que lo hace más útil, escalable y productivo.

- *Estrecha integración con Java:* El código fuente de Scala está diseñado de tal manera que su compilador puede interpretar las clases de Java. Además, su compilador puede utilizar los marcos, las bibliotecas Java y las herramientas, etc. Después de la compilación, los programas Scala pueden ejecutarse en JVM.

- *Desarrollo de aplicaciones de escritorio y basadas en la web:* para las aplicaciones web, proporciona soporte mediante la compilación en JavaScript. De manera similar, para las aplicaciones de escritorio, se puede compilar en el código de bytes JVM.

- *Utilizado por grandes empresas:* la mayoría de las empresas populares como Apple, Twitter, Walmart, Google, etc. mueven la mayoría de los códigos a Scala desde otros idiomas. La razón es que es altamente escalable y se puede usar en operaciones de back-end.

HADOOP

Apache Hadoop es el framework más importante para trabajar con Big Data. La mayor fortaleza de Hadoop es la escalabilidad. Se actualiza de trabajar en un solo nodo a miles de nodos sin ningún problema de manera transparente.

Los diferentes dominios de Big Data significan que podemos administrar los datos de videos, medios de texto, datos transaccionales, información de sensores, datos estadísticos, conversaciones de redes sociales, consultas de motores de búsqueda, datos de comercio electrónico, información financiera, datos meteorológicos, actualizaciones de noticias, debates en foros, informes ejecutivos, etc.

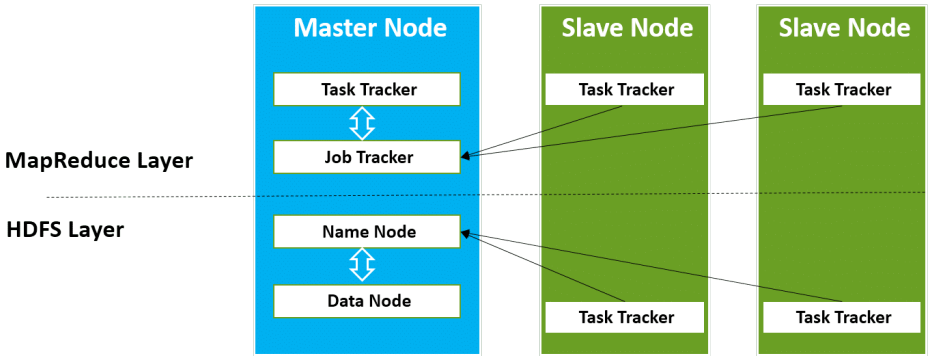
Doug Cutting de Google y los miembros de su equipo desarrollaron un proyecto de código abierto conocido como HADOOP que le permite manejar una gran cantidad de datos. Hadoop ejecuta las aplicaciones sobre la base de MapReduce, donde los datos se procesan en paralelo y realizan el análisis estadístico completo sobre una gran cantidad de datos.

Es un framework que se basa en la programación en Java. Está destinado a funcionar desde un solo servidor hasta miles de máquinas, cada una de las cuales ofrece computación y almacenamiento local. Admite la gran colección de conjuntos de datos en un entorno informático distribuido.

El framework basado en la biblioteca de software Apache Hadoop que otorga permisos para distribuir una gran cantidad de procesamiento de conjuntos de datos en grupos de computadoras utilizando modelos de programación sencillos.

La arquitectura de alto nivel de Hadoop

Arquitectura Hadoop basada en los dos componentes principales, a saber, MapReduce y HDFS



El módulo Apache Hadoop

- Hadoop Common: incluye las utilidades comunes que admiten los otros módulos de Hadoop

- HDFS: el sistema de archivos distribuidos de Hadoop proporciona acceso sin restricciones y de alta velocidad a la aplicación de datos.

- Hadoop YARN: esta tecnología se utiliza básicamente para la programación de trabajos y la gestión eficiente de los recursos del clúster.

- MapReduce: Esta es una metodología altamente eficiente para el procesamiento paralelo de grandes volúmenes de datos.

Hay otros proyectos incluidos en el módulo Hadoop que son menos utilizados:

- Apache Ambari: Es una herramienta para la gestión, monitorización y aprovisionamiento de los clústeres de Hadoop. Apache Ambari es compatible con los programas HDFS y MapReduce. Los aspectos más destacados de Ambari son:

  La gestión del marco Hadoop es altamente eficiente, segura y consistente.

  Gestión de operaciones de clúster con una interfaz de usuario web intuitiva y una API robusta.

  La instalación y configuración del clúster de Hadoop se simplifica de manera efectiva.

  Se utiliza para admitir la automatización, la configuración inteligente y las recomendaciones.

  La configuración avanzada de seguridad del clúster viene adicional con este kit de herramientas.

  Todo el clúster se puede controlar utilizando métricas, mapas de calor, análisis y solución de problemas.

  Los mayores niveles de personalización y extensión hacen que esto sea más valioso.

- Cassandra: Es un sistema distribuido para manejar una cantidad extremadamente grande de datos que se almacenan en varios servidores básicos. El sistema de administración de bases de datos (DBMS) tiene una alta disponibilidad sin ningún punto único de falla.

- HBase: es un sistema de administración de bases de datos distribuidas no relacionales que funciona de manera eficiente en conjuntos de datos dispersos y es altamente escalable.

- Apache Spark: este es un motor de cómputo de Big Data altamente ágil, escalable y seguro, versátil y suficiente trabajo en una amplia variedad de aplicaciones como procesamiento en tiempo real, aprendizaje automático, ETL, etc.

- Hive: es una herramienta de almacenamiento de datos que se utiliza básicamente para analizar, consultar y resumir los conceptos de datos analizados sobre el marco Hadoop.
- Pig: Pig es un marco de trabajo de alto nivel que nos garantiza trabajar en coordinación con Apache Spark o MapReduce para analizar los datos. El lenguaje utilizado para codificar los marcos se conoce como Pig Latin.
- Sqoop: este marco se utiliza para transferir los datos a Hadoop desde bases de datos relacionales. Esta aplicación se basa en una interfaz de línea de comandos.
- Oozie: este es un sistema de programación para la gestión del flujo de trabajo, ejecutando rutas de flujo de trabajo para completar con éxito la tarea en un Hadoop.
- Zookeeper: servicio centralizado de código abierto que se utiliza para proporcionar coordinación entre aplicaciones distribuidas de Hadoop. Ofrece el servicio de registro y sincronización a un alto nivel.
- Hadoop Mapreduce (capa de procesamiento/computación): MapReduce es un modelo de programación paralelo que se utiliza principalmente para escribir una gran cantidad de aplicaciones de distribución de datos diseñadas por Google para el procesamiento eficiente de grandes cantidades de conjuntos de datos, en un gran grupo de clústeres.
- Hadoop HDFS (capa de almacenamiento): el sistema de archivos distribuidos de Hadoop o HDFS se basa en el sistema de archivos de Google (GFS), que proporciona un sistema de archivos distribuido que está especialmente diseñado para ejecutarse en hardware básico. Reduce las fallas o errores y ayuda a incorporar hardware de bajo costo. Brinda acceso de rendimiento de procesamiento de alto nivel a los datos de la aplicación y es adecuado para aplicaciones con grandes conjuntos de datos.
- Hadoop YARN: Hadoop YARN es un marco utilizado para la programación de trabajos y la gestión de recursos de clúster.

- Hadoop Common: incluye bibliotecas y utilidades de Java que proporcionan los archivos Java que son esenciales para iniciar Hadoop.
- Rastreador de tareas: es un nodo que se utiliza para aceptar tareas como la reproducción aleatoria y el rastreador de trabajos de formulario Mapreduce.
- Rastreador de trabajos: es un proveedor de servicios que ejecuta trabajos de Mapreduce en un clúster.
- Nodo de nombre: es un nodo donde Hadoop almacena toda la información de ubicación de archivos (ubicación de datos almacenados) en el sistema de archivos distribuido de Hadoop.
- Nodo de datos: los datos se almacenan en el sistema de archivos distribuido de Hadoop.
- Nodo de datos: almacena datos en el sistema de archivos distribuido Hadoop.

¿Cómo funciona Hadoop?

Hadoop ayuda a ejecutar una gran cantidad de procesamiento en el que el usuario puede conectar varias computadoras a una sola CPU, como un único sistema funcional distribuido y asi tener un conjunto particular de máquinas agrupadas que leeran los datos en paralelo y asi obtener la salida deseada.

Hadoop ejecuta código en un grupo de computadoras y realiza las siguientes tareas:

- Los datos se dividen inicialmente en archivos y directorios. Los archivos se dividen en bloques de tamaño consistente que van desde 128M y 64M.
- Luego, los archivos se distribuyen en varios nodos de clúster para un mayor procesamiento de los datos.
- El Job Tracker inicia sus tareas de programación en nodos individuales.
- Una vez que todos los nodos han terminado con la programación, la salida es retornada.

Los desafíos que enfrentan los datos a escala y el alcance de Hadoop

Los Big Data se clasifican en:

- Estructurado: que almacena los datos en filas y columnas como conjuntos de datos relacionales
- No estructurado: aquí los datos no se pueden almacenar en filas y columnas como videos, imágenes, etc.
- Semiestructurado: los datos en formato XML son legibles por máquinas y humanos

Existe una metodología estandarizada que Big Data sigue, destacando la metodología de uso de ETL.

ETL: significa Extraer, Transformar y Cargar.

Extraer: obtener los datos de múltiples fuentes

Transformar: convertir los datos existentes para que se ajusten a las necesidades analíticas

Cargar: los sistemas correctos para obtener valor en ellos.

Comparación con tecnologías de bases de datos existentes

La mayoría de los sistemas de administración de bases de datos no están a la altura para operar a niveles tan elevados de exigencias de Big data debido a la pura ineficiencia técnica. Cuando los datos están totalmente desestructurados, el volumen de datos es enorme, donde los resultados son de alta velocidad, finalmente, la única plataforma que puede enfrentar el desafío de manera efectiva es Apache Hadoop.

Hadoop debe principalmente su éxito a un marco de procesamiento llamado MapReduce que es fundamental para su existencia. La tecnología MapReduce brinda la oportunidad a todos los programadores de contribuir con su parte donde se dividen grandes conjuntos de datos y se procesan de forma independiente en paralelo. Estos codificadores no necesitan conocer la computación de alto rendimiento y pueden trabajar de manera eficiente sin preocuparse por las complejidades dentro del clúster, el monitoreo de tareas, la administración de fallas de nodos, etc.

Hadoop también contribuye con su otra plataforma, conocida como Hadoop Distributed File System (HDFS). La principal fortaleza de HDFS es su capacidad para escalar rápidamente y funcionar sin problemas, independientemente de cualquier falla en los nodos. HDFS, en esencia, divide archivos grandes en bloques o unidades más pequeñas que van desde 64 a 128 MB y luego se copian en un par de nodos del clúster. A partir de esto, HDFS garantiza que no se detenga el trabajo, incluso cuando algunos nodos quedan fuera de servicio. HDFS posee API para garantizar que el programa MapReduce se utilice para leer y escribir datos (contenidos) simultáneamente a altas velocidades. Cuando existe la necesidad de acelerar el rendimiento, y luego agregar nodos adicionales en paralelo al clúster y la mayor demanda se puede satisfacer de inmediato.

Ventajas de Hadoop

Da acceso al usuario para escribir y probar rápidamente los sistemas distribuidos y luego distribuye automáticamente los datos y funciona entre las máquinas y, a su vez, utiliza el paralelismo primario de los núcleos de la CPU.

La biblioteca Hadoop está desarrollada para encontrar/buscar y manejar las fallas en la capa de aplicación.

Los servidores se pueden agregar o quitar del clúster dinámicamente en cualquier momento.

Es de código abierto basado en aplicaciones Java y, por lo tanto, compatible con todas las plataformas.

Funciones y características de Hadoop

Apache Hadoop es la herramienta de big data más popular y poderosa, que proporciona la capa de almacenamiento mas confiable del mundo: HDFS (Sistema de archivos distribuidos de Hadoop), un motor de procesamiento por lotes llamado MapReduce y una capa de administración de recursos como YARN.

Código abierto: Apache Hadoop es un proyecto de código abierto. Significa que su código se puede modificar de acuerdo con los requisitos comerciales.

Procesamiento distribuido: el almacenamiento de datos se mantiene de manera distribuida en HDFS en todo el clúster, los datos se procesan en paralelo en el clúster de nodos.

Tolerancia a fallas: de manera predeterminada, las tres réplicas de cada bloque se almacenan en el clúster en Hadoop y solo se modifican cuando es necesario. La tolerancia a fallas de Hadoop se puede examinar en tales casos, cuando cualquier nodo deja de funcionar, los datos en ese nodo se pueden recuperar fácilmente de otros nodos. El marco recupera automáticamente las fallas de un nodo o tarea en particular.

Confiabilidad: debido a la replicación de datos en el clúster, los datos pueden ser confiables y se almacenan en el clúster de la máquina a pesar de las fallas de la máquina. Incluso si su máquina falla, sus datos también se almacenarán de manera confiable.

Alta disponibilidad: los datos están disponibles y accesibles incluso si se produce una falla de hardware debido a múltiples copias de datos. Si ocurriera algún incidente, como si su máquina o algún hardware fallara, se accederá a los datos desde otra ruta.

Escalabilidad: Hadoop es altamente escalable y, de una manera única, se puede agregar fácilmente hardware a los nodos. También proporciona escalabilidad horizontal, lo que significa que se pueden agregar nuevos nodos en la parte superior sin ningún tiempo de inactividad.

Económico: Hadoop no es muy costoso, ya que se ejecuta en un grupo de hardware básico. No requerimos de ninguna máquina especializada para ello. Hadoop proporciona una gran reducción de costos ya que es muy fácil agregar más nodos en la parte superior aquí. Entonces, si el requisito aumenta, entonces hay un aumento de nodos, sin tiempo de inactividad y sin mucha planificación previa.

Fácil de usar: no es necesario que el cliente se ocupe de la informática distribuida, el marco se encarga de todo. Así que es fácil de usar.

Localidad de datos: Hadoop funciona según el principio de localidad de datos que establece que el movimiento de computación de datos en lugar de datos a computación. Cuando el cliente envía su algoritmo, el algoritmo se mueve a los datos en el clúster en lugar de llevar los datos a la ubicación donde se envía el algoritmo y luego procesarlo.

Suposiciones de Hadoop

Hadoop está escrito teniendo en cuenta una gran cantidad de clústeres de computadoras y se basa en las siguientes suposiciones:

- El hardware puede fallar debido a un mal funcionamiento externo o técnico donde, en su lugar, se puede usar hardware básico.
- El procesamiento se ejecutará en lotes y se hace hincapié en un alto rendimiento en lugar de una baja latencia.
- Las aplicaciones que se ejecutan en HDFS tienen grandes conjuntos de datos. Un archivo típico en HDFS puede tener un tamaño de gigabytes a terabytes.
- Las aplicaciones requieren un modelo de acceso de una sola escritura y varias lecturas.
- La computación en movimiento es más barata en comparación con los datos en movimiento.

Principios de diseño de Hadoop

Los siguientes son los principios de diseño sobre los que trabaja Hadoop:

- El sistema se administrará y reparará a sí mismo apenas ocurra un evento.
- Las tolerancias a fallas se enrutan de forma automática y transparente, se administran en torno a fallas, ejecutan especulativamente tareas redundantes si se detecta que ciertos nodos se ejecutan en una fase más lenta.
- El rendimiento se escala en función de la linealidad.
- Cambio proporcional en términos de capacidad con cambio de recursos (Escalabilidad)
- La localidad de datos se denomina latencia más baja, ancho de banda más bajo.
- Se basa en núcleo simple, modular y extensible (Económico).

APACHE SPARK

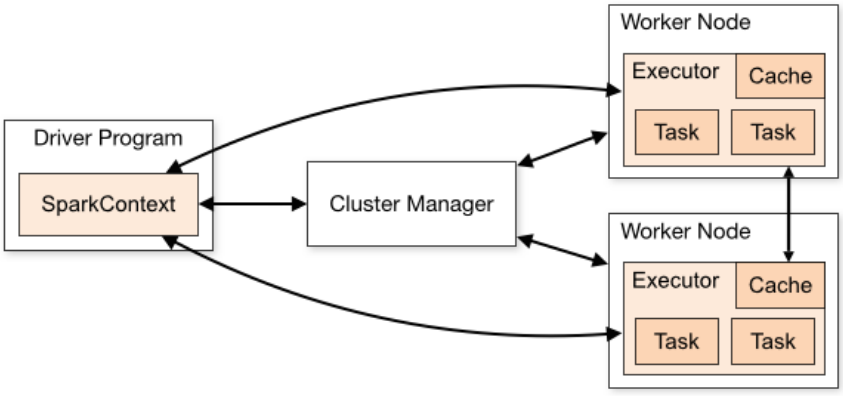
Apache Spark es un framework popular en el campo de Big Data. Viniendo de un background de codigo en Python y SQL, sin entender los mecanismos, confunde bastante al principio. El cambio de ejecutar código en una sola máquina hacia el uso de clústeres, junto con el cambio del tamaño de los datos procesados de MB a GB (incluso TB), nos lleva a comenzar a aprender Spark.

Computación distribuida

Para lograr la computación distribuida se requiere la gestión de recursos y tareas en un grupo de máquinas. La gestión de recursos implica adquirir las máquinas disponibles para la tarea actual, mientras que la gestión de tareas implica coordinar el código y los datos en todo el clúster.

Una aplicación Spark consta de un programa controlador y ejecuta computación paralela en un clúster. Para iniciar una aplicación Spark, el programa controlador que se ejecuta en una máquina host primero iniciará un objeto SparkContext. Este objeto SparkContext se comunicará con un administrador de clústeres, que puede ser el administrador de clústeres independiente de Spark, Mesos, YARN o Kubernetes, para adquirir recursos para esta aplicación. Luego, el objeto SparkContext enviará el código de la aplicación y las tareas a los nodos trabajadores.

Para una aplicación, un nodo worker puede tener varios executors, según la cantidad de CPU disponibles en este nodo worker. Durante el cómputo de una aplicación, cada executor mantiene los datos en la memoria o en el almacenamiento en disco y ejecuta tareas. De esta forma, los executors están aislados entre sí y las tareas de una misma aplicación se ejecutan en paralelo.



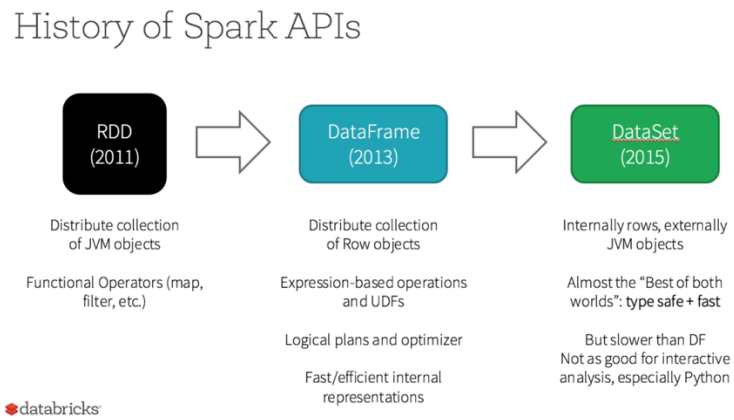
Conjunto de datos distribuido resistente (RDD)

RDD es una abstracción central en Spark, que significa Resilient Distributed Dataset. Permite la partición de grandes datos en datos más pequeños que se adaptan a cada máquina, de modo que el cálculo se puede realizar en paralelo en varias máquinas. Además, los RDD se recuperan automáticamente de las fallas de los nodos para garantizar la resiliencia del almacenamiento.

HDFS (Sistema de archivos distribuidos de Hadoop) es otro concepto importante con el que uno se encuentroa a menudo cuando usa Spark. Aunque tanto RDD como HDFS tienen que ver con el almacenamiento distribuido resiliente, están diseñados para manejar diferentes problemas. La perspectiva resiliente de RDD se refiere al manejo automático de fallas de cálculo. Si bien HDFS se trata de la administración del almacenamiento, está diseñado para manejar fallas de almacenamiento.

API de Spark

Spark proporciona tres API: RDD, DataFrames, Datasets. Estas tres API garantizan una computación de datos resiliente y distribuida, y son adecuadas para diferentes escenarios de aplicación.



RDD es la API fundamental de bajo nivel proporcionada por Spark y admite la manipulación de datos no estructurados o semiestructurados. Usar RDD es como decirle a Spark cómo realizar una tarea, en lugar de simplemente decirle a Spark qué tarea realizar. Como resultado, RDD ofrece la mejor flexibilidad de codificación y control de datos entre

las tres API. Sin embargo, al mismo tiempo, sin aprovechar la optimización interna de Spark, un buen maestro de RDD impone requisitos más altos en la experiencia de los programadores.

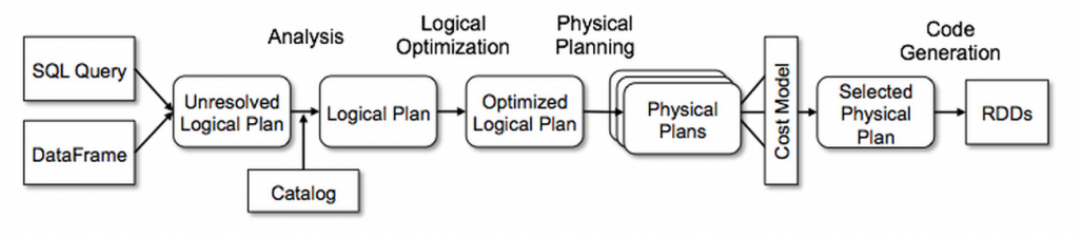
Además de la API RDD de bajo nivel, Spark también proporciona la API DataFrames de alto nivel. Los DataFrames enfatizan la estructura de datos. Por lo tanto, DataFrames es amigable para los programadores que vienen con experiencia en bases de datos relacionales. Al usar DataFrames, se siente bastante similar a usar Pandas DataFrame o Excel Spreadsheet, pero con Spark manejando la computación del clúster bajo tras bambalinas.

Si decimos que las API de RDD y DataFrames están en los dos lados de una inclinación, con RDD en el lado del control flexible de bajo nivel y DataFrames en el lado de la codificación fácil de alto nivel, entonces la API de conjuntos de datos se encuentra en el medio de los otros dos API. Además de la API de DataFrames, la API de conjuntos de datos impone seguridad de tipos para evitar errores en tiempo de ejecución.

Tanto las API de RDD como las de conjuntos de datos requieren seguridad de tipo y solo son compatibles con Java y Scala. Sin embargo, la API de DataFrame admite lenguajes de escritura dinámica, como Python y R.

Spark SQL

Spark SQL es el módulo de Spark para trabajar con datos estructurados.
Spark SQL funciona con las dos API estructuradas, a saber, conjuntos de datos y dataframes. Aprovechando la información del esquema que solo está disponible en Datasets o DataFrames, el código Spark SQL le dice a Spark qué hacer de manera declarativa, a diferencia de la forma en que le dice a Spark cómo hacerlo cuando se usa la API RDD de bajo nivel. De esta forma, el código escrito en Spark SQL se beneficia del catalizador de Spark, que optimiza el rendimiento. Por lo tanto, usar Spark SQL con las API estructuradas es más fácil de escribir código de alto rendimiento.



Usar Spark SQL con la API de DataFrames es equivalente a ejecutar una consulta SQL en la base de datos relacional. Las funciones SQL de uso común, como filtro, combinación, agregación y función de ventana, también están disponibles en Spark SQL. Spark SQL y la API de DataFrames admiten varios lenguajes de programación, incluidos Python, R, Scala y Java.

Spark SQL, Presto y Hive admiten consultas de datos a gran escala que residen en almacenamiento distribuido mediante la sintaxis SQL, pero se utilizan para diferentes escenarios.

Spark SQL es el módulo central de Spark, mientras que Presto está en el ecosistema de Hadoop. Spark SQL hace hincapié en el cálculo y, por lo general, se usa en ETA y canalización a gran escala. Sin embargo, Presto hace hincapié en la consulta y se utiliza más a menudo para análisis ad-hoc. Tanto Spark SQL como Presto calculan en memoria. Cuando se trata de escasez de memoria, Spark SQL permite que se haga spilling en el disco, mientras que Presto sufrirá problemas de OOM. La tolerancia a fallas también se considera en Spark SQL, pero no en Presto.

Hive es un software de almacenamiento de datos que gestiona datos estructurados a gran escala en el ecosistema Hadoop. Las consultas de Hive se pueden ejecutar a través de Spark o MapReduce. Hive tiene su propio motor SQL llamado HiveQL. Como hemos mencionado anteriormente, Spark SQL es el módulo de Spark para trabajar con datos estructurados. Del mismo modo, Hive es el módulo de Hadoop para trabajar con datos estructurados.

Hemos discutido algunos conceptos básicos de Spark. Aunque la API DataFrame de alto nivel junto con Spark SQL facilita la escritura de código de alto rendimiento, comprender cómo funciona Spark ayuda a mejorar aún más el rendimiento. En la próxima sesion, veremos YARN como ejemplo y analizaremos cómo comprender la administración de tareas y recursos de Spark mediante la interfaz de usuario web de YARN.

SCALA

Comenzando con la programación en Scala

Encontrar un compilador: hay varios IDE en línea, como GeeksforGeeks IDE, Scala Fiddle IDE, etc., que se pueden usar para ejecutar programas Scala sin instalar.

Programación en Scala: dado que Scala es muy similar sintácticamente a otros lenguajes ampliamente utilizados, es más fácil codificar y aprender en Scala. Los programas se pueden escribir en Scala en cualquiera de los editores de texto más utilizados como Notepad++, gedit, etc. o en cualquiera de los editores de texto. Después de escribir el programa, guarde el archivo con la extensión .sc o .scala.

Para Windows y Linux: Antes de instalar Scala en Windows o Linux, debe tener instalado el Kit de desarrollo de Java (JDK) 1.8 o superior en su sistema. Porque Scala siempre se ejecuta en Java 1.8 o superior.

La gente siempre piensa que Scala es una extensión de Java. Pero no es cierto. Es completamente interoperable con Java. Los programas de Scala se convierten en un archivo .class que contiene el código de bytes de Java después de la compilación exitosa y luego se pueden ejecutar en JVM (Java Virtual Machine).

En este sesion, discutiremos cómo ejecutar los programas Scala en IDE en línea.

Ejemplo: Un programa simple para imprimir Hello UNI! utilizando un enfoque orientado a objetos.

```
// Scala program to print Hello, UNI!
// by using object-oriented approach

// creating object
object Geeks {

        // Main method
        def main(args: Array[String])
        {

                // prints Hello, UNI!
                println("Hello, UNI!")
        }
}
```

Caracteristicas de scala

Hay muchas características que lo hacen diferente de otros idiomas.

- Orientado a objetos: cada valor en Scala es un objeto, por lo que es un lenguaje de programación puramente orientado a objetos. El comportamiento y el tipo de objetos están representados por las clases y rasgos en Scala.
- Funcional: también es un lenguaje de programación funcional ya que cada función es un valor y cada valor es un objeto. Proporciona soporte para funciones de alto orden, funciones anidadas, funciones anónimas, etc.
- Estáticamente tipificado: el proceso de verificar y hacer cumplir las restricciones de los tipos se realiza en tiempo de compilación en Scala. A diferencia de otros lenguajes de programación tipificados estáticamente como C++, C, etc., Scala no espera la información de tipo redundante del usuario. En la mayoría de los casos, el usuario no necesita especificar un tipo.
- Extensible: se pueden agregar nuevas construcciones de lenguaje a Scala en forma de bibliotecas. Scala está diseñado para interpolar con JRE (Java Runtime Environment).
- Procesamiento concurrente y sincronizado: Scala permite al usuario escribir los códigos de una manera inmutable que facilita la aplicación del paralelismo (sincronización) y la concurrencia.
- Ejecutar en JVM y puede ejecutar código Java: Java y Scala tienen un entorno de tiempo de ejecución común. Así, el usuario puede pasar fácilmente de Java a Scala. El compilador de Scala compila el programa en un archivo .class, que contiene el código de bytes que JVM puede ejecutar. Todas las clases de Java SDK pueden ser utilizadas por Scala. Con la ayuda de Scala, el usuario puede personalizar las clases de Java.

Ventajas

- Las características complejas de Scala proporcionaron una mejor codificación y eficiencia en el rendimiento.
- Tuplas, macros y funciones son los avances en Scala.
- Incorpora la programación funcional y orientada a objetos que a su vez lo convierten en un lenguaje poderoso.
- Es altamente escalable y, por lo tanto, brinda un mejor soporte para las operaciones de back-end.

- Reduce el riesgo asociado con la seguridad de subprocesos, que es mayor en Java.
- Debido al enfoque funcional, generalmente, un usuario termina con menos líneas de códigos y errores, lo que resulta en una mayor productividad y calidad.
- Debido a la lazy computation, Scala calcula las expresiones solo cuando son necesarias en el programa.
- No hay métodos estáticos ni variables en Scala. Utiliza el objeto singleton (clase con un objeto en el archivo fuente).
- También proporciona el concepto Traits. Los rasgos son la colección de métodos abstractos y no abstractos que se pueden compilar en las interfaces de Java.

Desventajas

- A veces, dos enfoques hacen que la Scala sea difícil de entender.
- Hay un número limitado de desarrolladores de Scala disponibles en comparación con los desarrolladores de Java.
- No tiene optimización recursiva de cola verdadera ya que se ejecuta en JVM.
- Siempre gira en torno al concepto orientado a objetos porque cada función es un valor y cada valor es un objeto en Scala.

Aplicaciones

- Se utiliza sobre todo en el análisis de datos con Spark.
- Se utiliza para desarrollar las aplicaciones web y la API.
- Proporciona la facilidad para desarrollar los frameworks y las bibliotecas.
- Se prefiere usar en operaciones de back-end para mejorar la productividad de los desarrolladores.
- El procesamiento por lotes en paralelo se puede realizar con Scala.