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

CIENCIA DE DATOS:

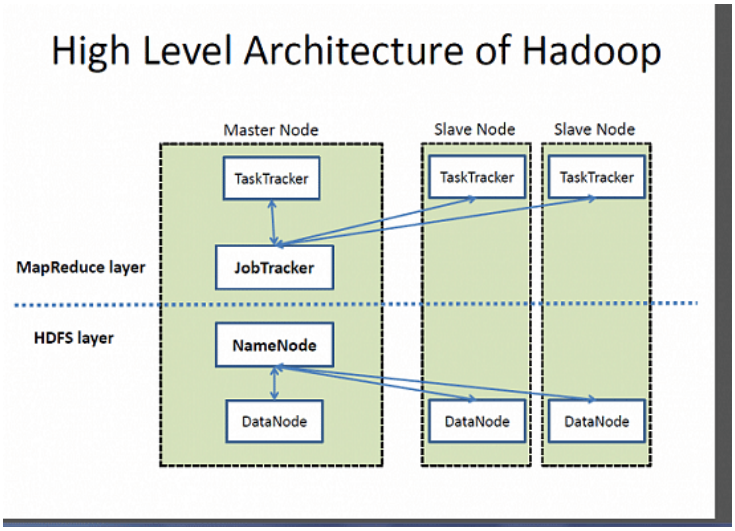
APRENDE LOS FUNDAMENTOS DE MANERA PRÁCTICA

SESION 02
HADOOP

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

ÍNDICE

OBJETIVO



El curso de capacitación Big Data y Hadoop está diseñado para proporcionar conocimientos y habilidades basicas para convertirse en un exitoso desarrollador de Hadoop. En este curso se cubrirá un conocimiento profundo de conceptos tales como Hadoop Distributed File System, Hadoop Cluster: nodo único y múltiple, Map-Reduce, etc.

Después de completar el curso, los estudiantes podrán analizar y trabajar con datos voluminosos de cualquier organización desde varias perspectivas y podrán desarrollar informes y se podrán ver tendencias y tomar decisiones con respecto a las actividades comerciales que se ejecutan en las organizaciones.

INTRODUCCION – HADOOP

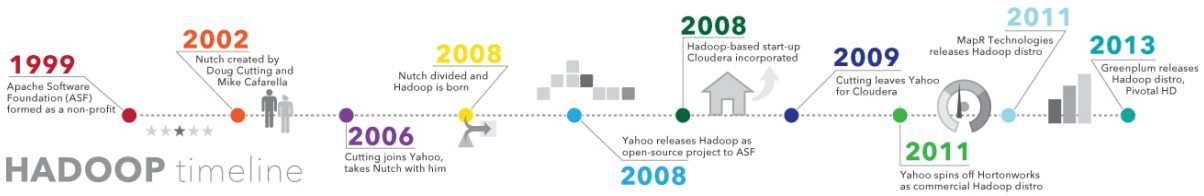
¿Qué es Hadoop?

La biblioteca de software Apache Hadoop es un framework que permite el procesamiento distribuido de grandes conjuntos de datos en clusters de computadoras utilizando modelos de programación simples. Está diseñado para escalar desde servidores individuales a miles de máquinas, cada una de las cuales ofrece computación y almacenamiento local. En lugar de depender del hardware para brindar alta disponibilidad, la biblioteca en sí está diseñada para detectar y manejar fallas en la capa de la aplicación, por lo que brinda un servicio de alta disponibilidad sobre un grupo de computadoras, cada una de las cuales puede ser propensa a fallas.

Asi mismo proporciona un framework para el almacenamiento distribuido y el procesamiento de grandes datos utilizando el modelo de programación MapReduce. Hadoop se diseñó originalmente para clústeres de computadoras creados a partir de hardware básico, que sigue siendo el uso común. Desde entonces, también ha encontrado uso en clústeres de hardware de gama alta. Todos los módulos en Hadoop están diseñados con la suposición fundamental de que las fallas de hardware son ocurrencias comunes y deben ser manejadas automáticamente por el framework.

El núcleo de Apache Hadoop consta de una parte de almacenamiento, conocida como Sistema de archivos distribuidos de Hadoop (HDFS), y una parte de procesamiento que es un modelo de programación MapReduce. Hadoop divide los archivos en grandes bloques y los distribuye entre los nodos de un clúster. Luego transfiere el código empaquetado a los nodos para procesar los datos en paralelo. Este enfoque aprovecha la localidad de los datos, donde los nodos manipulan los datos a los que tienen acceso. Esto permite que el conjunto de datos se procese de manera más rápida y eficiente de lo que sería en una arquitectura de supercomputadora más convencional que se basa en un sistema de archivos paralelo donde la computación y los datos se distribuyen a través de redes de alta velocidad.

Historia



A medida que la World Wide Web creció a finales del siglo XX y principios del 2000, se crearon motores de búsqueda e índices para ayudar a localizar información relevante en medio del contenido basado en texto. En los primeros años, los resultados de búsqueda eran devueltos por humanos. Pero a medida que la web creció de docenas a millones de páginas, se necesitó automatización. Se crearon web crawlers, muchos como proyectos de investigación dirigidos por universidades, y despegaron las nuevas empresas de motores de búsqueda (Yahoo, AltaVista, etc.).

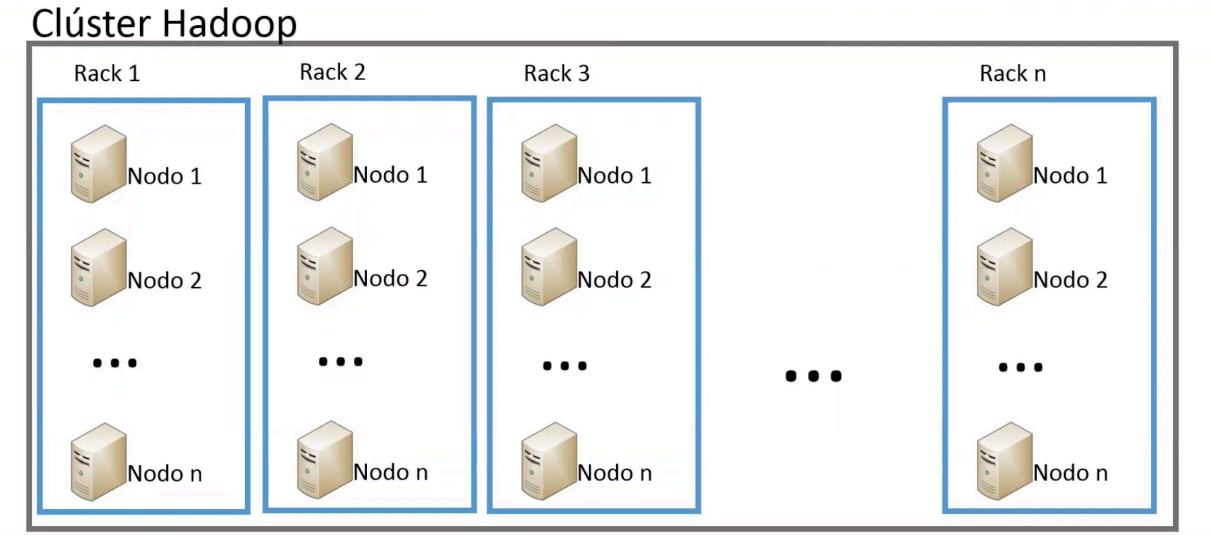
Uno de esos proyectos fue un motor de búsqueda web de código abierto llamado Nutch, una creación de Doug Cutting y Mike Cafarella. Querían obtener resultados de búsqueda web más rápido mediante la distribución de datos y cálculos en diferentes computadoras para poder realizar múltiples tareas simultáneamente. Durante este tiempo, estaba en marcha otro proyecto de motor de búsqueda llamado Google. Se basaba en el mismo concepto: almacenar y procesar datos de forma distribuida y automatizada para que los resultados de búsqueda web relevantes pudieran obtenerse más rápido.

En 2006, Cutting se unió a Yahoo y se llevó consigo el proyecto Nutch, así como ideas basadas en los primeros trabajos de Google con la automatización del almacenamiento y procesamiento de datos distribuidos. El proyecto Nutch se dividió: la parte del web crawler permaneció como Nutch y la parte de computación y procesamiento distribuidos se convirtió en Hadoop (llamado así por el elefante de juguete del hijo de Cutting). En 2008, Yahoo lanzó Hadoop como un proyecto de código abierto. Hoy en día, el framework y el ecosistema de tecnologías de Hadoop son administrados y mantenidos por Apache Software Foundation (ASF), una comunidad global de desarrolladores y colaboradores de software sin fines de lucro.

¿Por qué es importante Hadoop?

- Capacidad para almacenar y procesar grandes cantidades de cualquier tipo de datos rápidamente. Con volúmenes y variedades de datos en constante aumento, especialmente de las redes sociales y el Internet de las cosas (IoT), esa es una consideración clave.
- Poder computacional. El modelo de computación distribuida de Hadoop procesa big data rápidamente. Cuantos más nodos informáticos utilice, más potencia de procesamiento tendrá.
- Tolerancia a fallos. El procesamiento de datos y aplicaciones está protegido contra fallas de hardware. Si un nodo deja de funcionar, los trabajos se redirigen automáticamente a otros nodos para asegurarse de que la computación distribuida no falle. Múltiples copias de todos los datos se almacenan automáticamente.
- Flexibilidad. A diferencia de las bases de datos relacionales tradicionales, no es necesario preprocesar los datos antes de almacenarlos. Puede almacenar tantos datos como desee y decidir cómo usarlos más adelante. Eso incluye datos no estructurados como texto, imágenes y videos.

- Bajo costo. El marco de código abierto es gratuito y utiliza hardware básico para almacenar grandes cantidades de datos.
- Escalabilidad. Puede hacer crecer fácilmente su sistema para manejar más datos simplemente agregando nodos. Se requiere poca administración.



¿Cuáles son los desafíos de usar Hadoop?

La programación de MapReduce no es una buena combinación para todos los problemas. Es bueno para solicitudes de información simples y problemas que se pueden dividir en unidades independientes, pero no es eficiente para tareas analíticas iterativas e interactivas. MapReduce utiliza muchos archivos. Debido a que los nodos no se comunican entre sí, excepto a través de ordenaciones y reorganizaciones, los algoritmos iterativos requieren múltiples fases de asignación aleatoria/ordenación y reducción para completarse. Esto crea múltiples archivos entre las fases de MapReduce y es ineficiente para la computación analítica avanzada.

Hay una brecha de talento ampliamente reconocida. Puede ser difícil encontrar programadores de nivel de entrada que tengan suficientes conocimientos de Java para ser productivos con MapReduce. Esa es una de las razones por las que los proveedores de distribución están compitiendo para poner la tecnología relacional (SQL) por encima de Hadoop. Es mucho más fácil encontrar programadores con conocimientos de SQL que con conocimientos de MapReduce. Y, la administración de Hadoop parece en parte arte y en parte ciencia, lo que requiere un conocimiento de bajo nivel de los sistemas operativos, el hardware y la configuración del kernel de Hadoop.

Seguridad de datos. Otro desafío se centra en los problemas de seguridad de datos fragmentados, aunque están surgiendo nuevas herramientas y tecnologías. El protocolo de autenticación Kerberos es un gran paso para hacer que los entornos de Hadoop sean seguros.

Gestión y gobierno de datos completos. Hadoop no tiene herramientas completas y fáciles de usar para la gestión de datos, la limpieza de datos, el gobierno y los metadatos. Se carece especialmente de herramientas para la calidad y la estandarización de los datos.

¿Cómo se utiliza Hadoop?

Al ir más allá de su objetivo original de buscar en millones (o miles de millones) de páginas web y obtener resultados relevantes, muchas organizaciones buscan a Hadoop como su próxima plataforma de big data. Los usos populares hoy en día incluyen:

Almacenamiento y archivo de datos de bajo costo:

El costo modesto del hardware básico hace que Hadoop sea útil para almacenar y combinar datos tales como transaccionales, redes sociales, sensores, máquinas, datos científicos, secuencias de clics, etc. El almacenamiento de bajo costo le permite conservar información que actualmente no se considera crítica pero que podría querer analizar más adelante.

Sandbox para descubrimiento y análisis:

Debido a que Hadoop fue diseñado para manejar volúmenes de datos en una variedad de formas, puede ejecutar algoritmos analíticos. El análisis de big data en Hadoop puede ayudar a su organización a operar de manera más eficiente, descubrir nuevas oportunidades y obtener una ventaja competitiva del siguiente nivel. El enfoque de sandbox brinda la oportunidad de innovar con una inversión mínima.

Data Lakes:

Data Lakes admiten el almacenamiento de datos en su formato original o exacto. El objetivo es ofrecer una vista sin procesar o sin refinar de los datos a los científicos y analistas de datos para el descubrimiento y el análisis. Les ayuda a hacer preguntas nuevas o difíciles sin restricciones. Los lagos de datos no reemplazan a los almacenes de datos. De hecho, cómo proteger y gobernar los lagos de datos es un tema muy importante para TI. Pueden basarse en técnicas de federación de datos para crear estructuras de datos lógicas.

Complemente su almacén de datos (datawarehouse):

Ahora vemos que Hadoop comienza a ubicarse junto a los entornos de almacenamiento de datos, así como ciertos conjuntos de datos que se descargan del almacenamiento de datos a Hadoop o nuevos tipos de datos que van directamente a Hadoop. El objetivo final de cada organización es tener una plataforma adecuada para almacenar y procesar datos de diferentes esquemas, formatos, etc. para admitir diferentes casos de uso que se pueden integrar en diferentes niveles.

IoT y Hadoop:

Las cosas en el IoT necesitan saber qué comunicar y cuándo actuar. En el núcleo de IoT se encuentra una transmisión, siempre en forma de torrente de datos. Hadoop se usa a menudo como almacén de datos para millones o miles de millones de transacciones. Las capacidades masivas de almacenamiento y procesamiento también le permiten usar Hadoop como un espacio aislado para el descubrimiento y la definición de patrones que se monitorearán para la instrucción prescriptiva. Luego puede mejorar continuamente estas instrucciones, porque Hadoop se actualiza constantemente con nuevos datos que no coinciden con los patrones definidos anteriormente.

¿Como esta compuesto Hadoop?

El framework base de Apache Hadoop se compone de los siguientes módulos:

Hadoop Common: contiene bibliotecas y utilidades que necesitan otros módulos de Hadoop;
Sistema de archivos distribuidos de Hadoop (HDFS): un sistema de archivos distribuido que almacena datos en máquinas básicas, proporcionando un ancho de banda agregado muy alto en todo el clúster;

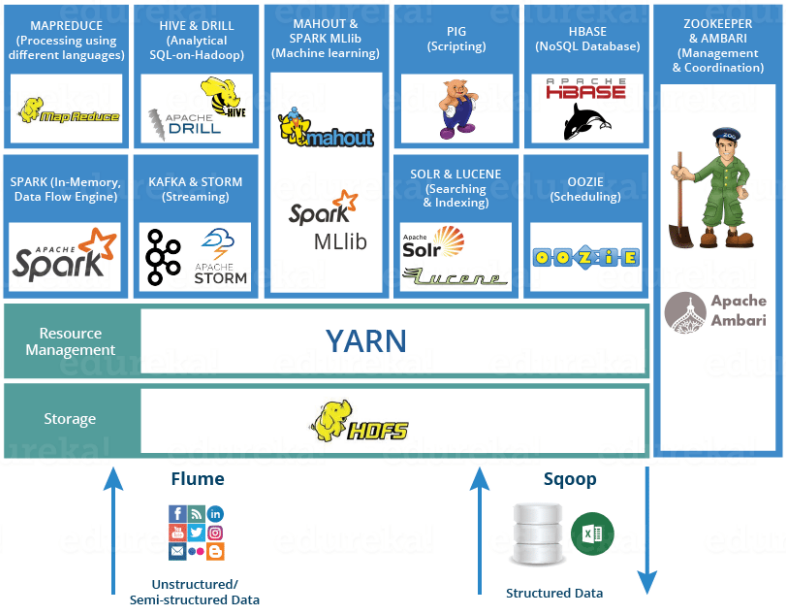
Hadoop YARN: (introducido en 2012) una plataforma responsable de administrar los recursos informáticos en clústeres y usarlos para programar las aplicaciones de los usuarios.

Hadoop MapReduce: una implementación del modelo de programación MapReduce para el procesamiento de datos a gran escala.

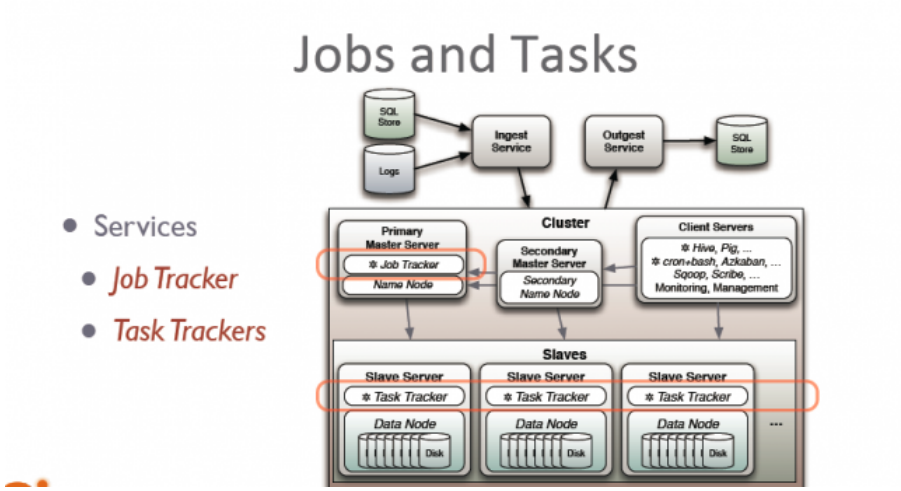
Hadoop Ozone: (presentado en 2020) Una tienda de objetos para Hadoop
El término Hadoop se usa a menudo tanto para los módulos base como para los submódulos y también para el ecosistema o la colección de paquetes de software adicionales que se pueden instalar encima o junto con Hadoop, como Apache Pig, Apache Hive, Apache HBase, Apache Phoenix, Apache Spark, Apache ZooKeeper, Apache Impala, Apache Flume, Apache Sqoop, Apache Oozie y Apache Storm.

Los componentes MapReduce y HDFS de Apache Hadoop se inspiraron en los documentos de Google sobre MapReduce y Google File System.

El marco Hadoop en sí está escrito principalmente en el lenguaje de programación Java, con algo de código nativo en C y utilidades de línea de comandos escritas como scripts de shell. Aunque el código MapReduce Java es común, se puede usar cualquier lenguaje de programación con Hadoop Streaming para implementar el mapa y reducir partes del programa del usuario. Otros proyectos en el ecosistema de Hadoop exponen interfaces de usuario más ricas.



JOB TRACKER



Por encima de los sistemas de archivos se encuentra el motor MapReduce, que consta de un JobTracker, al que las aplicaciones cliente envían trabajos de MapReduce. El JobTracker empuja el trabajo a los nodos TaskTracker disponibles en el clúster, esforzándose por mantener el trabajo lo más cerca posible de los datos.

Con un sistema de archivos compatible con racks, JobTracker sabe qué nodo contiene los datos y qué otras máquinas están cerca. Si el trabajo no se puede alojar en el nodo real donde residen los datos, se da prioridad a los nodos en el mismo rack. Esto reduce el tráfico de red en la red troncal principal.

Si un TaskTracker falla o se agota, esa parte del trabajo se reprograma. El TaskTracker en cada nodo genera un proceso de máquina virtual Java independiente para evitar que el propio TaskTracker falle si el trabajo en ejecución bloquea la JVM. Se envía un latido (heartbeat) desde TaskTracker a JobTracker cada pocos minutos para verificar su estado. Jetty (Java web server) expone el estado y la información de Job Tracker y TaskTracker y se puede ver desde un navegador web.

¿Entonces qué es un JobTracker en Hadoop?

JobTracker es el servicio daemon para enviar y rastrear trabajos de MapReduce en Hadoop. Job Tracker se ejecuta en su propio proceso JVM. En un clúster de producción típico, se ejecuta en una máquina separada. Cada nodo esclavo está configurado con la ubicación del nodo de seguimiento de trabajos. JobTracker es un punto único de falla para el servicio Hadoop MapReduce. Si se cae, todos los trabajos en ejecución se detienen.

¿Qué es una JVM?

Tradicionalmente, el compilador de un lenguaje de alto nivel se encargaba de traducir ese lenguaje "sencillo" en lenguaje máquina, directamente utilizable por el computador a través del sistema operativo. Es decir, cuando compilamos un programa en C++ lo que obtenemos es un

programa ejecutable, por ejemplo para Windows, que este sistema operativo es capaz de ejecutar directamente contra el procesador, en un lenguaje "entendible" por este.

Sin embargo muchos lenguajes modernos como Java o C# (y otros lenguajes de la plataforma .NET), lo que hacen es utilizar un paso intermedio entre estos dos estados: entre el código de alto nivel en el que escribimos las aplicaciones y el de bajo nivel que sale del proceso de compilación.

Cuando compilas una aplicación escrita en lenguaje Java, en realidad éste no se compila a lenguaje máquina, directamente entendible por el sistema operativo, sino a un lenguaje intermedio denominado Byte Code. Lo mismo ocurre con las aplicaciones .NET que se compilan también a un lenguaje intermedio llamado MSIL (MicroSoft Intermediate Language).

Entre el Byte Code (o el MSIL en el caso de .NET) y el sistema operativo se coloca un componente especial llamado Máquina virtual que es el que realmente va a ejecutar el código. Esta idea, por cierto, no tiene nada que ver con las máquinas virtuales a las que estamos acostumbrados hoy en día que ejecutan sistemas operativos completos, sino que es un concepto mucho más antiguo.

En el caso de Java, La **Java Virtual Machine o JVM** toma el código *Byte Code* resultante de compilar tu aplicación Java y lo compila a su vez a código nativo de la plataforma en la que se está ejecutando. La ventaja principal de este esquema es que es muy fácil crear un programa en Java y que luego éste se pueda ejecutar en cualquier sistema operativo para el cual exista una implementación de la JVM (hoy en día, casi literalmente todos).

¿Cuántas instancias de JobTracker se ejecutan en un Hadoop Cluster?

Solo se ejecuta un proceso de seguimiento de trabajos en cualquier clúster de Hadoop.

¿Qué acciones realiza JobTracker en hadoop?

Las aplicaciones cliente envían trabajos al rastreador de trabajos. JobTracker habla con NameNode para determinar la ubicación de los datos. JobTracker ubica los nodos de TaskTracker con slots disponibles en los datos o cerca de ellos. JobTracker envía el trabajo a los nodos de TaskTracker elegidos.

Los nodos TaskTracker son monitoreados. Si no envían señales de latido con la frecuencia suficiente, se considera que han fallado y el trabajo se programa en un TaskTracker diferente. Un TaskTracker notificará al JobTracker cuando una tarea falle. El JobTracker decide qué hacer entonces: puede volver a enviar el trabajo a otro lugar, puede marcar ese registro específico como algo que debe evitarse e incluso puede incluir en la lista negra al TaskTracker como poco confiable. Cuando se completa el trabajo, JobTracker actualiza su estado. Las aplicaciones cliente pueden sondear JobTracker para obtener información.

¿Cómo programa JobTracker una tarea?

Los TaskTrackers envían mensajes de latido al JobTracker, generalmente cada pocos minutos, para asegurarle a JobTracker que aún está activo. Estos mensajes también informan a JobTracker sobre la cantidad de slots disponibles, por lo que JobTracker puede mantenerse

actualizado con respecto a dónde se puede delegar el trabajo en el clúster. Cuando JobTracker intenta encontrar un lugar para programar una tarea dentro de las operaciones de MapReduce, primero busca un slot vacío en el mismo servidor que aloja el DataNode que contiene los datos y, si no, busca un slot vacío en una máquina en el mismo rack.

El tipo de slot indica qué tipo de tareas (map o reduce) pueden servir. En un momento dado, solo se puede ejecutar una tarea por slot. Si bien la configuración de los slots es fundamental para el rendimiento, Hadoop utiliza de manera predeterminada números fijos de map slots y reduce slots en cada nodo durante la vida útil de un clúster.

¿Qué es un rastreador de tareas en Hadoop?

Un TaskTracker es un demonio de nodo esclavo en el clúster que acepta tareas (operaciones Map, Reduce y Shuffle) de un JobTracker. Task Tracker se ejecuta en su propio proceso JVM. Cada TaskTracker está configurado con un conjunto de ranuras, estas indican la cantidad de tareas que puede aceptar. TaskTracker inicia un proceso JVM separado para hacer el trabajo real (llamado Instancia de tarea), esto es para garantizar que la falla del proceso no elimine el rastreador de tareas. TaskTracker monitorea estas instancias de tareas, capturando los códigos de salida. Cuando las instancias de Tarea finalizan, con éxito o no, el rastreador de tareas notifica al JobTracker. Los TaskTrackers también envían mensajes de latido al JobTracker, generalmente cada pocos minutos, para asegurarle a JobTracker que aún está activo. Estos mensajes también informan a JobTracker sobre la cantidad de slots disponibles, por lo que JobTracker puede mantenerse actualizado con respecto a dónde se puede delegar el trabajo en el clúster.

¿Cuántas instancias de TaskTracker se ejecutan en un Hadoop Cluster?

Solo se ejecuta un proceso de seguimiento de tareas en cualquier nodo esclavo de Hadoop.

¿Qué es una instancia de tarea en Hadoop?

Las instancias de tareas son los trabajos reales de MapReduce que se ejecutan en cada nodo esclavo. TaskTracker inicia un proceso JVM separado para hacer el trabajo real (llamado Instancia de tarea), esto es para garantizar que la falla del proceso no elimine el rastreador de tareas. Puede haber múltiples procesos de instancia de tarea ejecutándose en un nodo esclavo. Esto se basa en la cantidad de slots configuradas en el rastreador de tareas. De forma predeterminada, se genera un nuevo proceso JVM de instancia de tarea para una tarea.

¿Dónde se ejecuta una instancia de tarea?

Cada instancia de tarea se ejecuta en su propio proceso JVM.

¿Cuántos procesos Daemon se ejecutan en un sistema Hadoop?

Hadoop se compone de cinco demonios separados. Cada uno de estos demonios se ejecuta en su propia JVM.

Los siguientes 3 demonios se ejecutan en nodos maestros:

- NameNode: este demonio almacena y mantiene los metadatos para HDFS.
- NameNode secundario: realiza funciones de limpieza para NameNode.
- JobTracker: administra trabajos de MapReduce, distribuye tareas individuales a máquinas que ejecutan Task Tracker.

Los siguientes 2 demonios se ejecutan en cada nodo esclavo.

- Nodo de datos: almacena bloques de datos HDFS reales.
- TaskTracker: responsable de instanciar y monitorear tareas individuales de Map y Reduce.

¿Cuál es la configuración de un nodo esclavo típico en un clúster de Hadoop? ¿Cuántas JVM se ejecutan en un nodo esclavo?

Se ejecuta una única instancia de un rastreador de tareas en cada nodo esclavo. El rastreador de tareas se ejecuta como un proceso JVM separado.

Se ejecuta una única instancia de un daemon DataNode en cada nodo esclavo. El demonio DataNode se ejecuta como un proceso JVM separado.

Una o varias instancias de instancia de tarea se ejecutan en cada nodo esclavo.
Cada instancia de tarea se ejecuta como un proceso JVM independiente.
La cantidad de instancias de tareas se puede controlar mediante la configuración.
Por lo general, una máquina de gama alta está configurada para ejecutar más instancias de tareas.

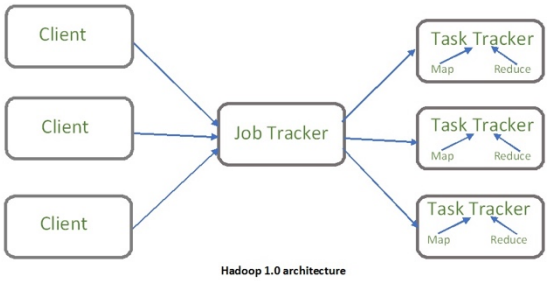
LOAD BALANCING

Hadoop divide la entrada de un trabajo de MapReduce en partes de tamaño fijo denominadas divisiones de entrada, o simplemente divisiones. Hadoop crea una tarea de mapa para cada división, que ejecuta la función de mapa definida por el usuario para cada registro en la división.

Tener muchas divisiones significa que el tiempo necesario para procesar cada división es pequeño en comparación con el tiempo para procesar toda la entrada. Por lo tanto, si estamos procesando las divisiones en paralelo, el procesamiento tiene mejor balanceo de carga si las divisiones son pequeñas, ya que una máquina más rápida podrá procesar proporcionalmente más divisiones en el transcurso del trabajo que una máquina más lenta. Incluso si las máquinas son idénticas, los procesos fallidos u otros trabajos que se ejecutan simultáneamente hacen deseable el balanceo de carga, y la calidad del balanceo de carga aumenta a medida que las divisiones se vuelven más finas. Por otro lado, si las divisiones son demasiado pequeñas, la sobrecarga de administrar las divisiones y la creación de tareas de mapas comienza a dominar el tiempo total de ejecución del trabajo. Para la mayoría de los trabajos, un buen tamaño de división tiende a ser el tamaño de un bloque HDFS, 64 MB de forma predeterminada, aunque esto se puede cambiar para el clúster (para todos los archivos recién creados) o se puede especificar cuando se crea cada archivo.

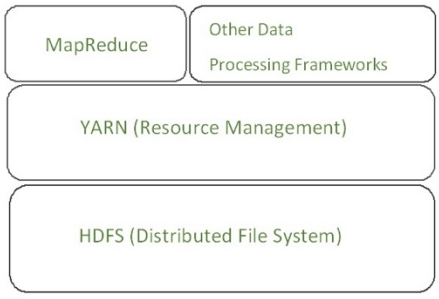
YARN (MapReduce 2.0)

YARN significa "*Yet Another Resource Negotiator*". Se introdujo en Hadoop 2.0 para eliminar el cuello de botella en Job Tracker que estaba presente en Hadoop 1.0. YARN se describió como un "Administrador de recursos rediseñado" en el momento de su lanzamiento, pero ahora ha evolucionado para ser conocido como un sistema operativo distribuido a gran escala utilizado para el procesamiento de Big Data.



Hadoop 1.0 architecture

La arquitectura YARN básicamente separa la capa de gestión de recursos de la capa de procesamiento. En la versión Hadoop 1.0, la responsabilidad del rastreador de trabajos se divide entre el administrador de recursos y el administrador de aplicaciones.



Hadoop 2.0

YARN también permite que diferentes motores de procesamiento de datos, como el procesamiento de gráficos, el procesamiento interactivo, el procesamiento de secuencias y el procesamiento por lotes, ejecuten y procesen datos almacenados en HDFS (Sistema de archivos distribuidos de Hadoop), lo que hace que el sistema sea mucho más eficiente. A través de sus diversos componentes, puede asignar dinámicamente varios recursos y programar el procesamiento de la aplicación. Para el procesamiento de datos de gran volumen, es muy necesario administrar los recursos disponibles adecuadamente para que cada aplicación pueda aprovecharlos.

Características de YARN: YARN ganó popularidad debido a las siguientes características:

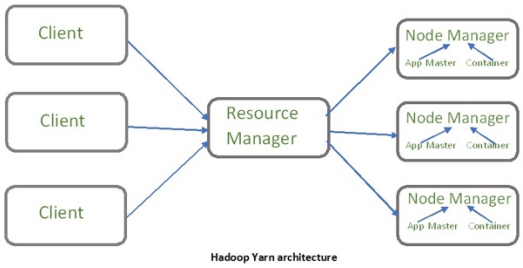
Escalabilidad: el programador en el administrador de recursos de la arquitectura YARN permite que Hadoop amplíe y administre miles de nodos y clústeres.

Compatibilidad: YARN es compatible con las aplicaciones de map reduce existentes sin interrupciones, por lo que también es compatible con Hadoop 1.0.

Utilización de clústeres: dado que YARN admite la utilización dinámica de clústeres en Hadoop, lo que permite una utilización de clústeres optimizada.

Multiusuario: permite el acceso a múltiples motores, lo que brinda a las organizaciones un beneficio de multiusuario.

Arquitectura de Hadoop Yarn:



Hadoop Yarn architecture

Los componentes principales de la arquitectura YARN incluyen:

Cliente: Envía trabajos map-reduce.

Administrador de recursos: es el demonio maestro de YARN y es responsable de la asignación y administración de recursos entre todas las aplicaciones. Cada vez que recibe una solicitud de procesamiento, la reenvía al administrador de nodo correspondiente y asigna recursos para completar la solicitud en consecuencia. Tiene dos componentes principales:

Programador: realiza la programación en función de la aplicación asignada y los recursos disponibles. Es un planificador puro, lo que significa que no realiza otras tareas, como la supervisión o el seguimiento, y no garantiza un reinicio si falla una tarea. El programador de YARN admite complementos como Capacity Scheduler y Fair Scheduler para particionar los recursos del clúster.

Gestor de aplicaciones: Es el encargado de aceptar la aplicación y negociar el primer contenedor del gestor de recursos. También reinicia el contenedor maestro de aplicaciones si falla una tarea.

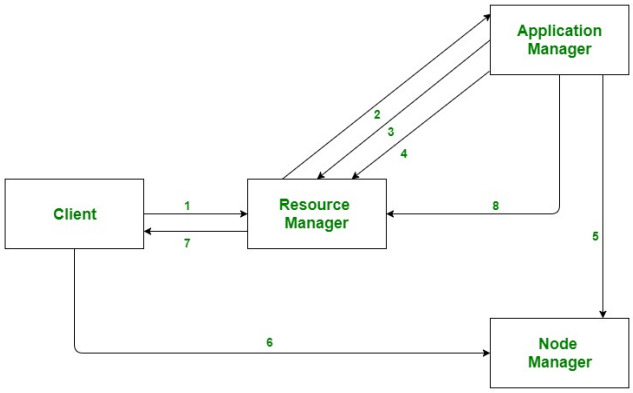
Administrador de nodos: se encarga de los nodos individuales en el clúster de Hadoop y administra la aplicación y el flujo de trabajo y ese nodo en particular. Su trabajo principal es mantenerse al día con el Administrador de recursos. Se registra con el Resource Manager y envía heartbeats con el estado de salud del nodo. Supervisa el uso

de recursos, realiza la gestión de registros y también elimina un contenedor según las instrucciones del administrador de recursos. También es responsable de crear el proceso contenedor e iniciarlo a pedido del maestro de la aplicación.

Maestro de aplicaciones: una aplicación es un solo trabajo enviado a un framework. El maestro de la aplicación es responsable de negociar los recursos con el administrador de recursos, rastrear el estado y monitorear el progreso de una sola aplicación. El maestro de aplicaciones solicita el contenedor al administrador de nodos mediante el envío de un Contexto de inicio de contenedor (CLC) que incluye todo lo que una aplicación necesita para ejecutarse. Una vez que se inicia la aplicación, envía el informe de estado al administrador de recursos de vez en cuando.

Contenedor: es una colección de recursos físicos como RAM, núcleos de CPU y disco en un solo nodo. Los contenedores son invocados por Container Launch Context (CLC), que es un registro que contiene información como variables de entorno, tokens de seguridad, dependencias, etc.

Flujo de trabajo de la aplicación en Hadoop YARN:



- El cliente presenta una solicitud
- El Administrador de recursos asigna un contenedor para iniciar el Administrador de aplicaciones
- El Administrador de aplicaciones se registra con el Administrador de recursos
- El Administrador de aplicaciones negocia contenedores desde el administrador de recursos
- El Administrador de aplicaciones notifica al administrador de nodos para lanzar contenedores
- El código de la aplicación se ejecuta en el contenedor.
- El cliente contacta al administrador de recursos/administrador de aplicaciones para monitorear el estado de la aplicación
- Una vez que se completa el procesamiento, el Administrador de aplicaciones cancela el registro con el Administrador de recursos