ADEN

PROCESANDO GRANDES VOLÚMENES DE DATOS

AUTOR: GASTÓN ADDATI

CONTENIDO

INTRODUCCIÓN

En el siguiente libro se presentan, en primera instancia, las definiciones de Small Data, Big Data y Huge Data, para, desde este punto, pensar cuáles herramientas de procesamiento de datos serán las pertinentes conforme al flujo de datos que se deba gestionar en una empresa.

Para el procesamiento de grandes volúmenes de datos, se presentará Hadoop, que es la tecnología que corre detrás de todos los sistemas de Big Data. Luego, se introducirá y ahondará en la arquitectura de Hadoop, compuesta por Apache HDFS, Apache Map Reduce, Apache HIVE y Apache HBASE.

De esta manera, se espera profundizar en las herramientas de procesamiento de Big Data, en su composición y funcionamiento.

01

BIGDATA – SMALL DATA Y HUGE DATA

Cuando nos referimos al procesamiento de grandes volúmenes de datos es frecuente encontrarnos con distintos conceptos que de alguna forma dan un orden de magnitud en lo que refiere a ese gran volumen de datos. Y, justamente, dependiendo del concepto al que nos estemos refiriendo, las tecnologías de procesamiento de los datos serán diferentes.

De alguna manera suena bastante lógico si lo contrastamos con el uso que le damos, por ejemplo, a nuestra computadora portátil o al uso que se le da un servidor de procesamiento de información en una empresa determinada. Tanto la computadora portátil como el servidor procesan datos, pero la magnitud y el procesamiento son en parte diferentes, por lo tanto, las particularidades son, en cada caso, únicas. Mientras que con nuestro ordenador portátil (o notebook) podemos procesar planillas de cálculos, usar software para programar algoritmos y, hasta incluso, utilizar una base de datos del tipo SQL, lo cierto es que las capacidades de ese hardware que viene en la notebook, son limitadas para ciertos tipos de trabajos y de procesamiento. Podríamos decir de manera muy simple, que existe una relación entre el problema que tengo que resolver y el hardware y las tecnologías que utilizaré para resolverlo.

Mientras para algunos trabajos con una computadora portátil o PC alcanzaría de sobra, para otro tipo de problemas, o para otro tipo de procesamiento de datos, sin dudas, no y, por eso, existen tecnologías de servidores más potentes (hardware de alta prestaciones) que son fabricados para dar respuesta a esos problemas.

Cuando nos referimos al big data, sabemos que nos referimos al procesamiento de grandes volúmenes de información. Pero, de alguna forma, algunas preguntas que el lector podría hacerse son: ¿Cuánto es grande? ¿Cómo podemos medir ese volumen para saber, efectivamente, que estamos trabajando en big data?

También, podríamos preguntarnos si el hardware de un servidor es suficiente para procesar ese gran volumen de datos que requiere el big data. ¿Cuánto hardware necesito? ¿Cuánto procesamiento, cuanta memoria y cuanto de almacenamiento en disco?

Las respuestas son simples y complejas a la vez. Las respuestas las iremos dando a lo largo de este curso y veremos, sobre todo, la necesidad de contar con tecnología específica para el manejo de los grandes volúmenes de datos, porque sin dudas, con un solo servidor sería imposible gestionar efectivamente un proyecto de Big Data de grandes dimensiones.

Big Data se refiere a los grandes volúmenes de datos. Sabemos que no existe una única y consensuada definición acerca de lo que es, pero diversos autores concuerdan en definirla en relación a la necesidad de tener que capturar, almacenar, procesar y analizar grandes volúmenes de datos. Tal vez el término grandes volúmenes no sea muy específico para determinar a partir de qué valor cuantitativo (por ejemplo, en Gigabytes) hablamos de big data. Y, es por eso, que existen diversas clasificaciones como Big Data, Small Data y Hudge Data.

Regresando a la definición de Big data, podemos dar una breve definición que nos proporciona la empresa de consultoría mundial llamada McKinsey, en la que afirma: "El Big Data se refiere a los conjuntos de datos cuyo tamaño está más allá de las capacidades de las herramientas típicas de software de bases de datos para capturar, almacenar, gestionar y analiza". Esta definición se encuentra bastante acorde a lo que veremos en esta asignatura.

Volumen, Variedad y Velocidad: fueron las 3 (tres) V's con las que se caracterizó inicialmente al big data. Esta caracterización la realizó Douglas Laney[1]:

- **Volumen:** término que se encuentra en relación con el tamaño de los datos que pueden ser procesados, generados y/o capturados por una organización para su posterior tratamiento y análisis.
- **Variedad:** término que se refiere a la variedad de los datos que se pueden utilizar y procesar con Big Data. Sabemos que los datos provienen de diferentes fuentes, ya sean estructurados, no estructurados o semiestructurados. Esta variedad es la que afecta directamente a la complejidad de almacenamiento y análisis de estos.

[1] Douglas, Laney. «3D Data Management: Controlling Data Volume, Velocity and Variety». Gartner, Doug Laney de Gartner: analista Doug Laney del grupo Gartner, es una empresa consultora y de investigación de las tecnologías de la información.

Big Data se refiere a los conjuntos de datos cuyo tamaño está más allá de las capacidades de las herramientas típicas de software de bases de datos para capturar, almacenar, gestionar y analizar.

- **Velocidad:** hace referencia a la rapidez en la que circulan los datos, así como al tiempo de procesamiento de los mismos.

Posterior a esta caracterización inicial, se adicionaron algunas V's que, con mucho tino, tienen en los sistemas de Big Data una importancia fundamental:

- **Veracidad:** dado el gran volumen de datos que se genera, es necesario analizarlos para garantizar así la autenticidad y fiabilidad para la posterior toma de decisiones.
- **Valor:** este contexto menciona la necesidad de seleccionar aquellos datos que sean útiles para poder rentabilizarlos y generar ventajas competitivas.

Si bien el big data por sí sólo no aporta mucho a una organización (porque se encuentra orientado hacia el procesamiento y gestión de los datos) la combinación del Big Data con diversos algoritmos (como, por ejemplo, de Inteligencia Artificial) y tecnologías específicas, permiten extraer valor agregado y podríamos decir que este es el eje central donde las empresas ponen foco... ¿Cómo extraer valor mediante el análisis de los grandes volúmenes de datos?[2]

Recordemos que con el Big Data podremos realizar una serie de análisis que nos permitirán conocer, predecir y hasta adaptar nuestro negocio a los objetivos que establezcamos.

Básicamente, el análisis de los datos puede ser de 3 (tres) grandes tipos:

- **Análisis Descriptivo:** la cual utiliza una serie de técnicas sencillas como medias, desviaciones típicas, gráficos, etc. Y es una etapa preliminar de procesamiento de datos que trata de contestar a preguntas sobre hechos que ya se han producido. Analiza lo sucedido con información estadística principalmente.
- **Análisis Predictivo:** pretende predecir resultados basándose en tendencias, patrones y anomalías dentro del histórico de datos. Su núcleo se basa en la relación de variables explicativas y sucesos pasados, que permiten explotarlos para predecir un resultado futuro.
- **Análisis prescriptivo:** trata de determinar las acciones que deben llevarse a cabo ante una situación concreta basándose en modelos predictivos. Este sistema analítico aporta recomendaciones óptimas sobre dichas acciones.

[2] Esta asignatura pondrá foco en cómo se hace para procesar datos masivos y en cuáles son las tecnologías más frecuentes a la hora de implementar un big data, pero recuerde que una de las cuestiones fundamentales en todo proyecto es el Business Analytics.

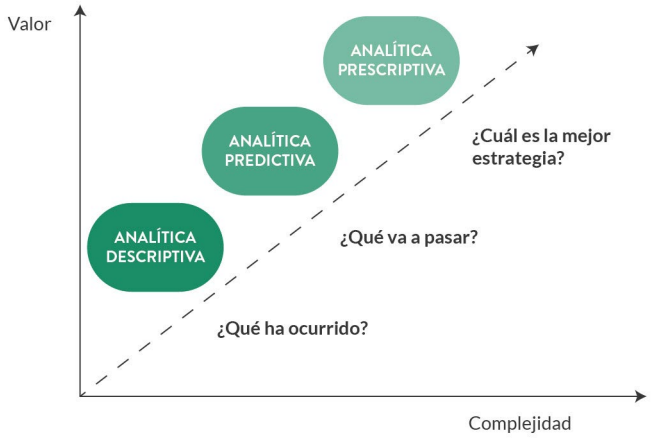


Figura 1. Tipos de análisis de datos. (Elaboración propia, 2022)

Con esta breve descripción, podríamos sintetizar que el Big Data hace alusión a grandes cantidades de datos que deben ser procesados a una velocidad adecuada para ser tratados y, a partir de ese tratamiento, extraer algún valor agregado. Pero aún no hemos respondido al interrogante de ¿cuán grande debe ser un sistema para ser considerado como Big Data? Lo cierto es que no hay un consenso, una clara marca que indique a partir de X cantidad de datos, es Big Data.

En mi experiencia personal, he visto proyectos de Big Data del orden de los varios cientos de Terabytes e, incluso, de PetaBytes. El lector debe saber que, en general, los datos se van incorporando al sistema de Big Data, en algunos proyectos que comienzan desde cero, los datos se comienzan a capturar (desde diversas fuentes de información) y el mismo Big Data va creciendo con el tiempo (con el uso y con la incorporación o ingesta diaria de los datos).

Recuerde que los sistemas de Big Data en general recopilan (o ingestan) datos de diversas fuentes, como ser: redes sociales, emails, formularios de contacto, sistemas de CRM, sensores de IoT (por ejemplo, sensores de temperatura y humedad, etc), sólo por mencionar algunos.

Sucede muy frecuentemente que, en diversas organizaciones, se hace alusión a un sistema de big data cuando, en realidad, si uno se pone a analizar el sistema, se da cuenta de que no es un big data, sino que se trata de un simple sistema de información, como puede ser un sistema de información CRM, ERP o incluso un sistema de facturación y Stock.

Suele existir mucha confusión respecto de lo que implica un Big Data y lo que es un Small Data.

Podemos afirmar que, en líneas generales, ambos llegan al objetivo de satisfacer una necesidad, de resolver un problema. El punto es el orden de magnitud de la información y el procesamiento.

En general, las empresas cuentan con sistemas del tipo CRM (para la gestión de clientes) o de un ERP (para la gestión de la empresa). Y esos sistemas, típicamente, son sistemas donde existe una Base de Datos (que generalmente es una base de datos estructurada – del tipo SQL -) y allí residen todos los datos necesarios, mediante los cuales, un gerente obtiene indicadores, por ejemplo, de ventas, de clientes que más compraron, de productos que faltan en el stock, etc.

Claramente, estos sistemas son de apoyo a la gestión cotidiana de las empresas, y son sumamente valiosos para poder mantener la operación del día a día, pero tal como se están describiendo, no son sistemas de Big Data.

Tampoco lo son las páginas web de comercio electrónico, ni tampoco lo son los formularios de Google que recopilan información de prospectos de clientes. En todos estos casos, nos referimos a proyectos de tecnología que utilizan sistemas de información basados en small data. Y es que antes del big data, todo era "Small" y no era necesaria la aclaración. Todos eran sistemas informáticos.

Small Data se refiere entonces a los procedimientos que realizaremos dentro de nuestra empresa, donde utilizaremos software informático, para la toma de decisiones cotidianas, y para la gestión de la información.

Small data se refiere a micro datos, a base de datos de tamaño reducido, y que no por ello se debe inferir que los datos no tienen valor o bien que no puede extraerse valor de los mismos.

¿Cómo sabemos si debemos reforzar una sucursal porque el día anterior faltaron 5 empleados? ¿Cómo sabemos a qué tienda enviar más stock si no tenemos indicadores de control? ¿Cómo sabemos a qué cliente debemos pagarle hoy si no sabemos las fechas de vencimiento? En fin, son muchas las ventajas que tiene utilizar los sistemas de información que terminan concluyendo en la disposición de un SMALL DATA.

¿Es mejor Big que Small? Es una pregunta que el lector seguramente se puede estar haciendo a sí mismo. La respuesta es que no hay uno mejor o uno peor. No son conceptos opuestos. Son conceptos diferentes. Con ambos se pueden realizar grandes cosas. Tienen perspectivas distintas y, por supuesto, desde lo tecnológico, tienen implementaciones y complejidades totalmente diferentes. Mientras con Small data, las organizaciones

están acostumbradas a instalar servidores y bases de datos de una manera natural, el despliegue de una solución de Big Data puede resultar mucho más costosa y no tan simple.

Hasta este punto hemos realizado un recorrido y una síntesis del Big Data y del Small Data, pero nos queda un término más para develar y se refiere a ¿qué es el Huge Data?

Huge Data: Es un término relativamente nuevo, que sugiere ser la "Evolución" de lo que conocemos como Big Data. Claro que muchas empresas todavía no han implementado un Big y ahora estamos hablando de un Huge, pero no hay que preocuparse. No son cuestiones desconocidas ni muy diferentes a las que definimos y conocemos hasta el momento.

El crecimiento exponencial que están teniendo las tecnologías de internet de las cosas (IoT) y el crecimiento exponencial de los datos, confluyen en que, dentro de muy poco tiempo, será necesario procesar mucha más información, sobre todo en tiempo real, utilizando machine learning y Deep learning, para poder obtener valor de cada dato y proveer información al instante a los diversos usuarios que lo requieran.

Es decir, el crecimiento de los datos y la necesidad de procesamiento en tiempo real, llevan a que la integración de diversas tecnologías permita tomar decisiones al instante, tal como ocurre hoy con los sistemas de big data, pero a una escala mucho más rápida y sobre todo considerando fuentes de datos como lo son los sensores de IoT.

Para visualizarlo, podría imaginarse una ciudad inteligente (Smart City) donde millones de sensores envían datos en tiempo real y donde un conjunto de tecnologías interviene para tomar decisiones en el instante, sea de prender, apagar luces, de abrir o cerrar vías alternas de tráfico, de disminuir el consumo eléctrico porque no hay actividad, etc.

También, cuando hablamos de Huge Data, no podemos dejar afuera la tecnología de BlockChain, que sin dudas es una de las tecnologías que más está evolucionando y que sin dudas, revolucionará el futuro, junto con la Inteligencia artificial[3].

[3] Recomiendo al lector leer la revista Harvard Deusto Business Nro 285 que se encuentra como material complementario.

02

PROCESANDO GRANDES VOLÚMENES DE DATOS

Cuando hacemos referencia al procesamiento de los datos en Big Data, automáticamente nos viene a la mente el término de Hadoop. Y es que hadoop es la tecnología que corre detrás de todos los sistemas de Big Data. También lo es Spark, pero Hadoop ha sido el referente desde los inicios.

De hecho, cuando una empresa adquiere una solución *"empaquetada"* de big data, a uno de los grandes proveedores de tecnología, como pueden Ser IBM, o Cloudera, no resultará llamativo saber que detrás del paquete, en verdad, funciona Hadoop y/o Spark y que estas grandes empresas, realizan algunas mejoras, colocan sus logos, lo empaquetan y le brindan soporte especializado ¿Por qué ocurre esto? Simplemente, porque estamos hablando de software open source.

Hadoop tiene como objetivo principal almacenar y procesar grandes cantidades de datos. Fue concebido, justamente, para ser eficiente en cuanto al procesamiento y al almacenamiento de grandes cantidades de información, a una alta velocidad (características del big data).

Nació en la empresa Yahoo! y fue creado por Doug Cutting[1], inspirado en tecnologías de Google (donde Doug Cutting trabajaba anteriormente). Hadoop nace en el año 2004, cuando su creador describe en un documento técnicas para manejar grandes volúmenes de datos, desgranándolos en problemas cada vez más pequeños para hacerlos abordables. Recién en el año 2008 se termina el desarrollo de Hadoop y comienza a utilizarse en empresas muy grandes como Yahoo, Facebook, Twitter, entre otras.

[1] https://en.wikipedia.org/wiki/Doug_Cutting

La importancia de Hadoop radica en que puede desarrollar tareas muy intensivas de computación masiva, dividiéndolas en pequeñas piezas que se distribuyen en un conjunto de máquinas. Conceptualmente, podemos hablar de una gran cantidad de máquinas (computadoras simples y comunes) que utilizan el poder de la computación distribuida. Es decir, el procesamiento y el almacenamiento se reparte entre varias máquinas o computadoras para ser más eficiente.

Hadoop está compuesto por cuatro grandes componentes:

- **Hadoop HDFS:** Sistema de archivos distribuido que proporciona acceso de alto rendimiento a datos.
- **Hadoop MapReduce:** Sistema basado en YARN para el procesamiento en paralelo de grandes conjuntos de datos.
- **Hadoop Common:** Utilidades comunes en las que se apoyan los otros módulos
- **Hadoop YARN:** Marco de trabajo para la planificación de tareas y gestión de recursos de clúster.

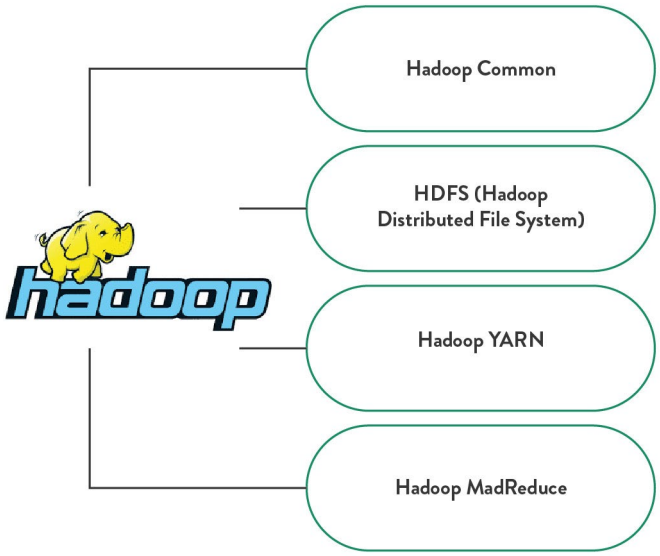
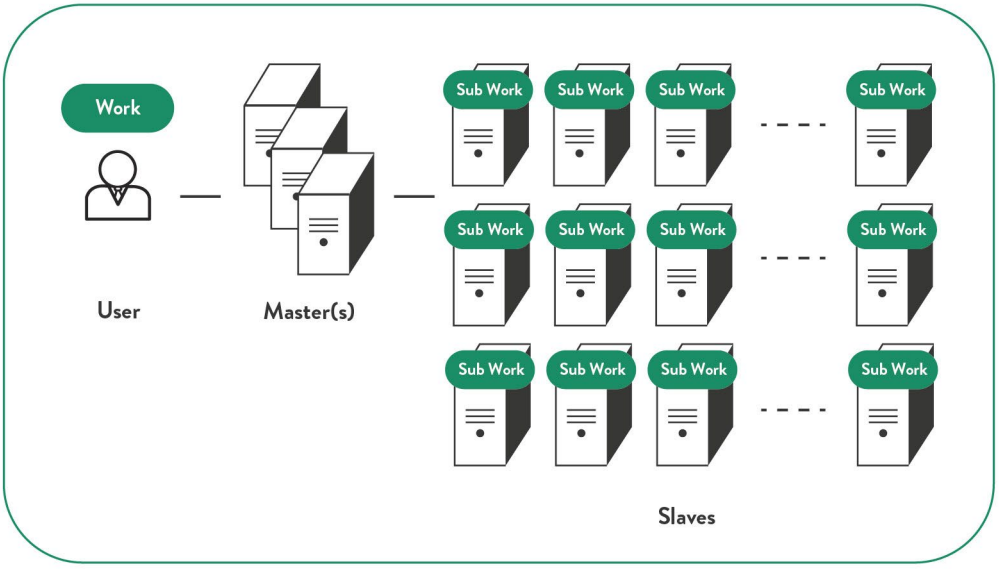


Figura 2. Arquitectura de Hadoop.
(https://elentornodehadoop.wordpress.com/tag/arquitectura-hadoop/2022)

Hadoop basa su funcionamiento a partir de los CLUSTERS. Un cluster es un conjunto de computadoras que funcionan colaborativamente para procesar la información. Como veremos más adelante, esas computadoras se clasificarán en Masters y Slaves (maestro / esclavo).



HADOOP CLUSTER

Figura 3. Hadoop Cluster.

Algunas de las características de Hadoop:

- **Escalabilidad:** esta herramienta permite almacenar y distribuir conjuntos de datos inmensos en sus cientos de servidores que operan en paralelo, permitiendo olvidarse de los límites que otras alternativas imponen.
- **Velocidad:** garantiza una eficiencia de procesamiento que nadie puede igualar ¿De qué otra forma se pueden procesar terabytes de información en pocos minutos?
- **Efectividad en costos:** disminuye los costos por el almacenamiento (lo hace más accesible).
- **Flexibilidad:** Apache Hadoop se adapta a las necesidades del negocio y lo acompaña en su expansión, aportando soluciones reales para cualquier iniciativa que surja.
- **Resistencia al fracaso:** su tolerancia a errores es uno de sus atributos mejor valorados por los usuarios ya que toda la información contenida en cada nodo tiene su réplica en otros nodos del cluster. En caso de producirse un fallo siempre existirá una copia lista para ser usada.

CONCEPTO

Un cluster es un conjunto de computadoras que funcionan colaborativamente para procesar la información.

Hadoop cuenta también con un gran número de aplicaciones que permiten resolver diferentes cuestiones relacionadas a la operación y a la gestión de la información y de los datos. Algunas de las aplicaciones del ecosistema de hadoop son:

Sqoop. Nos permite conectarnos a cualquier base de datos relacional e intercambiar datos con nuestro sistema de ficheros HDFS. Es muy importante poder incorporar fácilmente datos de nuestras bases de datos (datawarehouse, ERPs, etc.) y, del mismo modo, poder llevar fácilmente el resultado de un cálculo (scoring, segmentación) a nuestras bases de datos.

Flume. Nos permite recuperar información de sitios remotos. Mediante un agente que se ejecuta en el lugar que se producen los datos (fichero de log), recoge los datos y los importa en HDFS. Es unidireccional, no permite exportar datos de HDFS a otras ubicaciones. Resulta una herramienta francamente útil para recuperar información en tiempo real.

Hive. Actúa como la base de datos de Hadoop. Es un intérprete SQL – MapReduce. Traduce la query a programas Java que realicen los MapReduce. Esto permite utilizar herramientas de Business Intelligence convencionales (que admitan conexión ODBC) con los datos de HDFS.

Pig. Para trabajar con MapReduce es necesario programar, tener sólidos conocimientos de Java, saber cómo funciona MapReduce, conocer el problema a resolver, escribir, probar y mantener el código, etc. Para ello, es muy beneficioso disponer de un sistema más sencillo, que nos abstraiga de la complejidad del MapReduce.

Hbase. Es una base de datos columnar que se ejecuta sobre HDFS. Puede almacenar grandes cantidades de datos, accediendo a los mismos de una manera rápida, pudiendo procesarlos sin problemas incluso si hay datos dispersos.

Zookeeper. Cumple el rol de coordinador del ecosistema Hadoop, guardando la configuración de los metadatos, bloqueando un proceso cuando accede al mismo fichero, al mismo tiempo que otro proceso, guardando los usuarios y las contraseñas de acceso a los distintos lugares, etc.

Mahout. Es una librería de algoritmos de Machine Learning, codificados en Java. Es un programa que aprende por sí mismo.

03

APACHE HDFS

El HDFS (Hadoop distributed file system) es, tal vez, el principal componente de hadoop, debido a que permite crear sistemas de archivos utilizando servidores o computadoras simples y comunes (commodity hardware). Además, permite disponer de replicación y redundancia de los datos almacenados y, por supuesto, provee alto rendimiento.

El sistema de archivos HDFS de hadoop es útil para archivos realmente grandes. Digamos que no es aplicable para manejar archivos pequeños (de los que estamos acostumbrados a trabajar en nuestro día a día).

Como mencionamos anteriormente, hadoop basa su arquitectura HDFS en una arquitectura Master / Slave o lo que en español se denomina arquitectura maestro/esclavo.

Debemos de saber, que un cluster de hadoop HDFS consiste en un NAMENODE. Este NameNode es quien cumple el rol de ser el servidor Maestro y, además, es quien se ocupa de administrar y regular el acceso a los archivos que son solicitados por los clientes. Los clientes aquí serían los usuarios que operan el sistema.

Adicionalmente, recordemos que un cluster es un conjunto de máquinas y cada máquina dentro de un cluster se denomina nodo. Veamos el siguiente diagrama para comprenderlo mejor.

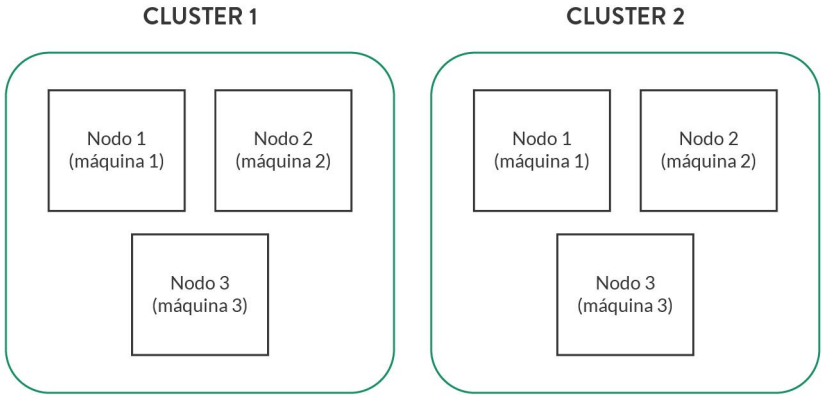


Figura 4. Cluster. (Elaboración propia, 2022)

Aquí observamos que tenemos 2 (dos) Clusters. Cada Cluster tiene 3 (tres) máquinas (servidores o Pc 's de hardware estándar), también llamados DataNodes.

Decíamos anteriormente que Hadoop HDFS funciona bajo el esquema Master/Slave. Veamos un poco más en detalle dicha explicación, por medio de la siguiente imagen de creación propia.

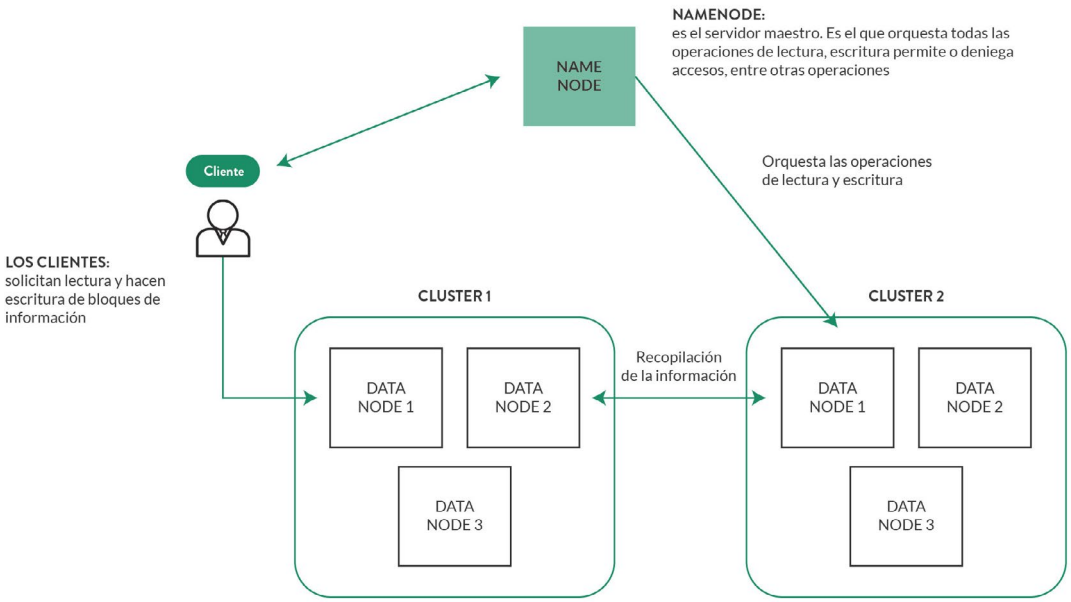


Figura 5. Esquema master/slave. (Elaboración propia, 2022)

Al servidor que llamamos Maestro (o master Server), es el único servidor que estará encargado de administrar y gestionar todas aquellas operaciones que requieran acceso a los datos, los cuales se encuentran almacenados en los diferentes DataNodes (o máquinas esclavas).

La característica fundamental del funcionamiento de hadoop HDFS es que el almacenamiento de los datos no se realiza en un único DataNode, sino que se replica entre varios, logrando así contar con dos grandes ventajas. La primera, la redundancia, y la segunda, el procesamiento de pequeños bloques de información, para que los nodos funcionen de manera más eficiente. **Esto es lógico:** no es lo mismo procesar un único archivo de 1TB de información, que procesar un único archivo de 1MB de información. Cuanto más divida la información en trozos más manejables, será más eficiente el procesamiento, y si al mismo tiempo replico la información en diversos nodos, lograré más eficiencia.

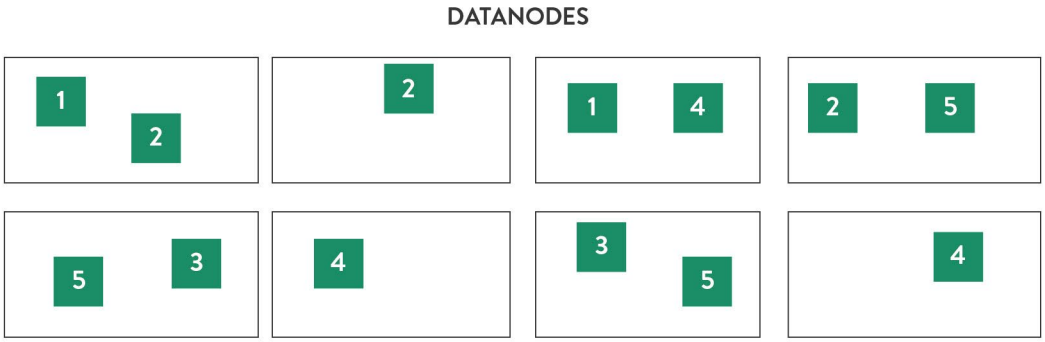


Figura 6. Replicación de bloques de información.
(https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Introduction 2022)

En la figura podemos observar cómo los bloques (numerados) son almacenados en diversos nodos (datanodes) lo que permite tener redundancia ante fallos.

Hadoop HDFS es un sistema de archivo realmente muy eficiente para trabajar en Big Data y, sobre todo, considerando que cualquiera puede implementar un sistema HDFS con hardware tradicional, pues no se requiere de un hardware extremadamente potente o difícil de adquirir. Adicionalmente, Hadoop HDFS fue concebido, justamente, para aquellos casos donde el hardware puede fallar.

LINK

https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Para más información al respecto de Hadoop HDFS, donde en la web oficial del proyecto encontrarán toda la documentación e incluso el código fuente de HDFS.

EXTRA

La explicación técnica de cómo hace Hadoop HDFS para elegir donde replicar la información (en qué datanode) o bien, cómo hace hadoop HDFS para asegurar la integridad de los datos, es decir, todo lo relacionado el funcionamiento a "bajo nivel" de hadoop, es información libre que se puede acceder en la página oficial y que no se explica en este apartado por exceder el alcance de la materia y la complejidad técnica que conlleva.

04

APACHE MAP REDUCE

MapReduce es otro de los componentes principales de Hadoop. Es un gran distribuidor de tareas que se adapta perfectamente con Apache HDFS.

MapReduce facilita la capacidad de proceso de grandes volúmenes de datos. Su innovación importante es la capacidad para realizar una consulta en un conjunto de datos, dividirla y ejecutarla en paralelo en múltiples nodos. La computación distribuida o distribución de la computación resuelve el problema que se plantea cuando los datos son demasiado grandes para caber en una sola máquina. En pocas palabras, MapReduce es un paradigma de programación pensado para el análisis de gran cantidad de datos en paralelo. Su modelo se basa en el concepto de "divide y vencerás", separando el procesado de la información en dos fases: MAP y REDUCE. Los beneficios principales de MapReduce son su escalabilidad y la variedad de datos que puede procesar, tales como archivos, tablas de bases de datos, sitios Web. Con MapReduce el procesamiento computacional puede ocurrir en datos almacenados en un sistema de archivos sin necesidad de cargarlos primero en una base de datos, una idea importante.

Una característica grande del entorno MapReduce es la capacidad específica para manejar datos no estructurados. En una base de datos relacional todo está en tablas, filas y columnas. Los datos ya tienen relaciones bien definidas. Esto no sucede siempre con los flujos de datos en bruto, y aquí es donde MapReduce tiene fortaleza. La carga de las partes (chunks) de texto en un campo con una base de datos es posible, pero realmente no es el mejor uso de una base de datos o el mejor camino para manejar tales datos, y MapReduce ayuda en esta tarea.

Anteriormente mencionamos que MapReduce se caracteriza por dividirse en 2 (dos) fases o etapas: la fase de MAP y la fase de REDUCE. Vamos a adentrarnos en estas fases para comprender mejor su funcionamiento.

Observemos la siguiente figura:

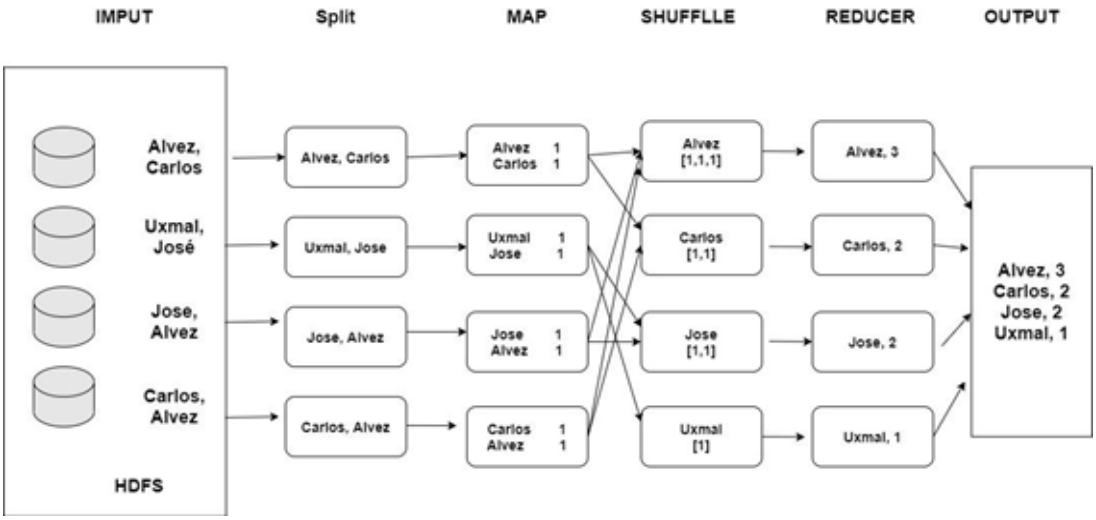


Figura 7. MapReduce. (Elaboración propia, 2022)

Considerando que tenemos información almacenada previamente en nuestro HDFS, y a sabiendas de que, además, esa información (por lo visto anteriormente) se encuentra dispersa entre varios nodos de un cluster, el funcionamiento de MapReduce tiene mucho sentido cuando alguien necesita ir a recopilar la información que se encuentra dispersa para presentarla o mostrarla como una salida (Output). Los datos se toman de la entrada y se ordenan en la llamada fase de MAP.

La fase de REDUCE toma la información procesada en la etapa del MAP y la presenta de una manera ordenada y ajustada a la salida que se requiere. Nótese que antes de la fase de reduce existe una etapa previa de consolidación, llamada **Shuffle**.

Dado que cada vez que se pide información se deben ejecutar "Jobs" de MapReduce y que estos Jobs generalmente tienen siempre la lógica parecida a la que mencionamos, es importante notar que, entonces, MapReduce tiene una gran habilidad, que es la de permitir la ejecución de Jobs en paralelo para ser más eficiente.

Tal vez la dificultad que tiene MapReduce es la complejidad que acarrea la programación de estos Jobs. Se requieren conocimientos avanzados de programación en Java y mucha experiencia en el terreno del big data.

Seguramente se estará preguntando si Hadoop HDFS y Hadoop MapReduce son componentes excluyentes para realizar proyectos de big data, y la respuesta es contundente: SI.

Sin MapReduce el HDFS no tendría sentido, porque tendríamos la información almacenada y dispersa en varios nodos, pero no habrá ningún orquestador que permita buscar, consolidar y presentar la información que necesitamos. Sin MapReduce el procesamiento de los datos sería imposible.

No obstante, también hay que mencionar que MapReduce es muy importante, pero no es perfecto. Para ciertos casos de big data (como veremos luego) MapReduce tiene algunas dificultades técnicas que son para destacar, y es allí donde otro framework como Apache Spark toma relevancia, porque logra incrementar notablemente la velocidad del procesamiento. Luego veremos cómo.

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
Aquí encontrará toda la documentación sobre MapReduce.

05

APACHE HIVE

Apache HIVE es una Solución de almacenamiento masivo de datos, el cuál integra funcionalidades de búsqueda a través de Apache Hadoop, las cuales son similares a los RDBMS (*Relational Database Management System*), así como, también, usa especificaciones HiveQL, las cuales son similares a SQL. Es decir, cuenta con un lenguaje propio, llamado HIVEQL, y se destaca por la optimización de consultas, debido a su estabilidad y rendimiento, más aún cuando el rendimiento es medido por el tiempo de respuesta y cantidad de procesos ejecutados.

HIVE, inicialmente, fue desarrollado por la empresa Facebook hasta que se pasó a la apache foundation. Es una tecnología ampliamente utilizada por grandes empresas que utilizan big data, como Netflix, Facebook, entre otras.

La característica fundamental de Apache Hive es que soporta el análisis de grandes conjuntos de datos almacenados bajo hadoop HDFS, es decir, fue desarrollado para trabajar con datos masivos (en proyectos de big data) y, como dijimos antes, ofrece un lenguaje de consulta, muy similar al lenguaje SQL que se denomina HIVEQL.

Como mencionamos anteriormente, los JOBs que se programaban en Hadoop MapReduce, que eran Jobs difíciles de programar, pueden encontrar cierto alivio al realizarse con Hive. Básicamente, lo que hace HIVE es transformar las consultas SQL en Jobs de MapReduce y, dado que armar una consulta con HIVEQL es mucho más amigable que hacerlo en MapReduce nativo, esto aporta sin dudas una gran ventaja para quienes deben hacer este tipo de trabajos.

Si bien trabajar con HIVE es similar a trabajar con bases de datos estructuradas como pueden ser bases de datos SQL u Oracle, no hay que perder de vista que HIVE no es un motor de base de datos, sino que es un lenguaje de consulta.

Veamos las diferencias para poder entender mejor los conceptos:

El término de motor de base de datos es frecuentemente llamado servidor de base de datos o sistema de administración de la base de datos. Se utiliza para crear, leer,

actualizar y eliminar (CRUD) datos de una base de datos, garantizando accesibilidad, consistencia, seguridad, respaldo y recuperación, control de concurrencia, robustez, entre otros grandes factores.

Algunos sistemas de base de datos son: MySQL, PostgreSQL, MSSQL, DB2, SQLite, MS Access, entre otros.

Figura 8. HIVE. *(https://blog.dataprius.com/index.php/2019/11/22/una-intranet-que-es-y-para-que-sirve-en-la-empresa/2022)*

Los Motores de Bases de Datos, nacen como alternativa para optimizar el procedimiento de acceso, consulta y extracción o inyección de información de las Bases de Datos, creando así un entorno más sencillo, agradable y eficaz a la hora de utilizarlas.

Un lenguaje de consulta, en cambio, es aquel programa informático que permite mediante un conjunto de instrucciones, acceder a una base de datos determinada, para poder extraer información, agregar información o manipularla. El lenguaje de consulta más popular es el lenguaje SQL.

Tabla comparativa. Sistema de BD Tradicionales VS. APACHE HIVE

SISTEMAS DE BASES DE DATOS TRADICIONALES	
Es utilizado para análisis en tiempo real de la información, como por ejemplo, registrar las ventas realizadas a cada instante	Se utilizará hacer análisis de datos estáticos como información en un archivos de texto
Es utilizado para realizar lecturas y escrituras tantas veces como se necesite	Es utilizado para hacer muchas lecturas y una única escritura.
Está preparado para manejar en el orden de los Terabytes de información	Permite manejar información en el orden de los PETAbytes.
Típicamente los sistemas de bases de datos tradicionales, no están preparados para trabajar o soportar datos particionados	Permite trabajar con datos particionados

Figura 9. Tabla comparativa. (Elaboración propia, 2022)

Cuando hablamos, por ejemplo, de un sistema CRM, podemos pensar como una gran base de datos donde tenemos todos nuestros clientes, contactos, oportunidades registradas en un único punto, permitiendo acceder a la información, agregar nueva, o modificar la existente. [1]”

[1]” Cuando trabajamos con sistemas de Big Data, las bases de datos no son las típicas que mencionamos anteriormente. Se requiere tecnología específica para almacenar mucha información y sobre todo estructurada y no estructurada.

SOBRE EL FUNCIONAMIENTO DE APACHE HIVE

Hemos dicho que Hive fue desarrollado para funcionar sobre hadoop. Básicamente, es una infraestructura de data warehouse que facilita administrar grandes conjuntos de datos almacenados en un ambiente distribuido.

Hive tiene 3 (tres) funciones principales:

1. Resumen de datos
2. Consulta
3. Análisis

Hive organiza los datos en tablas para el sistema HDFS y puede ejecutar las consultas en varios DataNodes en paralelo (convirtiéndolos a trabajos de MapReduce).

Las tablas de Hive son similares a las tablas de una base de datos del tipo relacional. Desde Hive debemos estructurar los datos agrupándolos en tablas, con sus columnas y tipos de datos asociados.

A su vez, Hive tiene una particularidad con el llamado schema-on-read. Esta propiedad le permite a Hive ser más flexible en la lectura de los datos; por ejemplo, un mismo dato se puede ajustar a varios esquemas, uno en cada lectura. Los sistemas RDBMS tienen una política schema-on-write, que obliga a las escrituras a cumplir un esquema. En este caso acelera las lecturas, por eso es más eficiente.

ARQUITECTURA APACHE HIVE

La arquitectura de Apache Hive se describe de la siguiente forma:

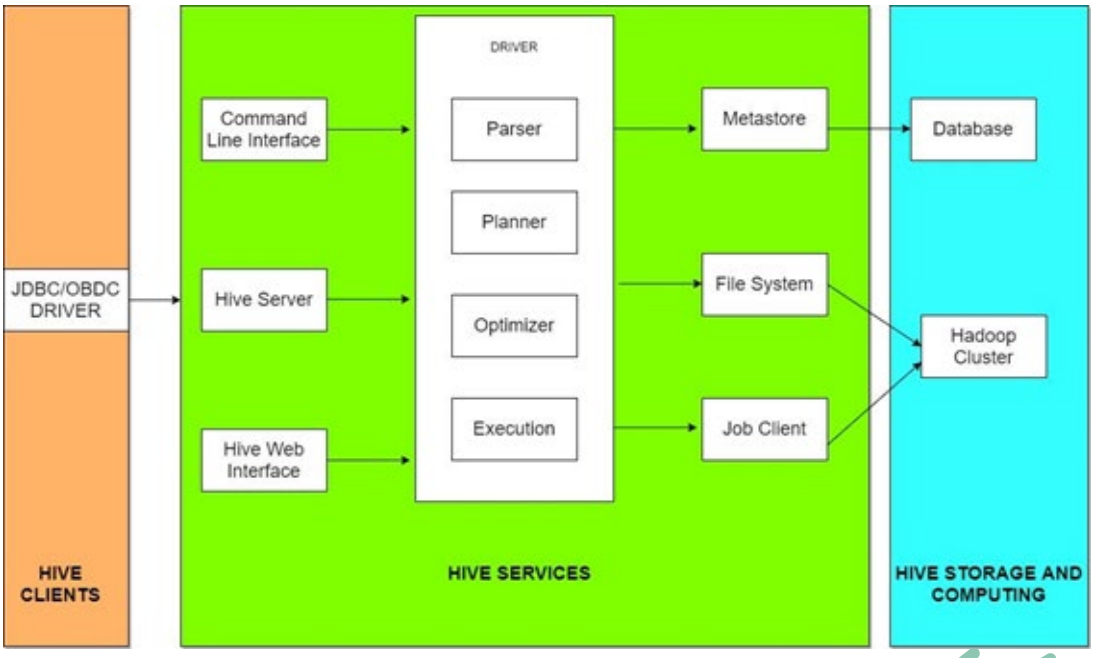


Figura 10. Arquitectura de Apache HIVE. (Elaboración propia, 2022)

Hive Server: se compone de una interfaz que permite a clientes externos ejecutar consultas contra Apache Hive y obtener los resultados.

Hive Client: componente que provee los drivers que permitirán efectuar la comunicación. Esos drivers, típicamente, son los denominados OBDC/JDBC. Estos dependen de la aplicación que se utilice. Por ejemplo, aplicaciones basadas en JAVA utilizan el driver de comunicación JDBC. Estos drivers permiten la comunicación entre el cliente y los servidores.

Hive Services: todas las interacciones de los clientes se realizan a través de este módulo de servicios. Cualquier consulta se realiza aquí dentro.

Hive MetaStore: reside en una base de datos relacional como MySQL, PostgreSQL o Apache Derby (base de datos interna), donde persiste la información. Mantiene un seguimiento de los metadatos, las tablas y sus tipos mediante Hive DDL (Data Definition Language). Además, el sistema se puede configurar para que también almacene estadísticas de las operaciones y registros de autorización para optimizar las consultas.

Ventajas de Apache Hive:

- Reduce la complejidad de la programación MapReduce al usar HQL como lenguaje de consulta (dialecto de SQL).
- Está orientado a aplicaciones de tipo Data Warehouse, con datos estáticos, poco cambiantes y sin requisitos de tiempos de respuesta rápidos.
- Permite a los usuarios despreocuparse de en qué formato y dónde se almacenan los datos.

Desventaja:

- Hive no es la mejor opción para consultas en tiempo real o de tipo OLTP (Online Transaction Processing). Es por eso que Hive es particularmente útil para aplicaciones de data warehouse, donde los datos relativamente estáticos son analizados y donde no se requieren respuestas rápidas o donde los datos no cambian rápidamente.

06

APACHE HBASE

Es una base de datos open source, distribuida, columnar (column-oriented database) que se ejecuta en HDFS. Hbase no soporta SQL, de hecho, HBase no es una base de datos relacional.

Veremos a continuación que en Hbase cada tabla contiene filas y columnas como una base de datos relacional. También, veremos que HBase permite que muchos atributos sean agrupados llamándolos familias de columnas, de tal manera que los elementos de una familia de columnas son almacenados en un solo conjunto. Eso es distinto de las bases de datos relacionales orientadas a filas, donde todas las columnas de una fila dada son almacenadas en conjunto.

Hbase forma parte la Fundación de Software Apache (Apache Foundation) y se ejecuta sobre HDFS (el sistema de archivos distribuidos de Hadoop que vimos con anterioridad), proporcionando tolerancia a fallos al momento de almacenar grandes cantidades de datos distribuidos, es decir, dispersos entre varios nodos de un cluster.

Por otro lado, HBASE, es adecuado para acelerar operaciones de lectura y escritura en los grandes conjuntos de datos con un alto rendimiento y una baja latencia de entrada/salida.

SOBRE EL BIG TABLE DE GOOGLE

Decíamos que HBASE deriva de lo que se conoce como BigTable de Google. BigTable es un sistema de gestión de base de datos creado por Google con las características de ser distribuido de alta eficiencia. Está construido sobre lo que se conoce como **GFS** (*Google File System*) que sería algo similar al hadoop HDFS.

BigTable comenzó a ser desarrollado a principios de 2004 y es, actualmente, una apuesta muy fuerte de la empresa Google[2'].

[2'] https://www.bbvaapimarket.com/es/mundo-api/bigtable-el-servicio-de-base-de-datos-nosql-con-el-que-google-quiere-dominar-los-big-data/

Bigtable es un mapa distribuido ordenado y multidimensional con tres dimensiones: filas, columnas y marca temporal. Pero ¿cómo se relacionan entre sí? Es un sistema que divide los datos en columnas para almacenar toda la información en tablas compuestas por celdas. Cada una de esas celdas dispone de una marca temporal que permite visualizar la evolución de ese dato. A lo largo del tiempo, Google creó BigTable porque los sistemas de bases de datos tradicionales no tenían ni tienen la capacidad de crear sistemas lo suficientemente grandes. Además, estos sistemas de bases de datos relacionales, como **SQL Server, Oracle o MySQL** fueron pensados y diseñados para que ejecutarse en un solo servidor con mucha potencia. Por ello, no encajarían en las estructuras distribuidas en miles de servidores.

SOBRE EL FUNCIONAMIENTO DE HBASE

HBASE se utiliza fundamentalmente en aplicaciones o en proyectos de BigData que requieren grandes volúmenes de datos y que, sobre todo, requieren la utilización o implementación de analíticas en tiempo real.

HBASE almacena tablas muy grandes. Cuando decimos muy grandes, nos referimos tal vez a millones de registros que tienen miles de columnas. Y dado que, fundamentalmente HBASE tiene como propiedad principal que provee almacenamiento orientado a columnas, donde estas columnas, como veremos luego, son organizadas en familias y a su vez estas familias serán almacenadas físicamente juntas en un cluster distribuido, lo que hace HBASE es garantizar que las lecturas y las escrituras sean mucho más rápidas comparado con los sistemas de bases de datos tradicionales, como podría ser el caso de SQL, por citar alguno. Por eso es que, anteriormente, mencionamos que era idea para analíticas en tiempo real.

Seguramente, el lector puede estar preguntándose qué relación podría existir entre HBASE y Apache HIVE, descrito con anterioridad. Pues bien, recordemos que HIVE en principio no es un sistema de base de datos, sino más bien, un lenguaje de consulta que utiliza un lenguaje propio similar al SQL (llamado HIVEQL); lo que hace a grandes rasgos es evitar el problema de programar en MapReduce, porque HIVE convierte automáticamente las consultas a un JOB de MapReduce.

HBASE, en cambio, es un sistema de base datos que además de todas las características que estamos mencionando, también se “integra” con HIVE, lo que termina generando una dupla de aplicaciones realmente interesante por el poder de la potenciación que ambos pueden generar, sobre todo cuando estamos en proyectos de las características que tienen los Big Data’s.

Recomiendo para quien le interese, leer el paper de Google, sobre bigtable desde aquí:
http://static.googleusercontent.com/media/research.google.com/es//archive/bigtable-osdi06.pdf
o bien desde el portal del curso. Es un paper sumamente técnico, no obligatorio de lectura, y sólo para aquellos tecnólogos que son curiosos.

HBASE tiene algunos aspectos técnicos que también son importantes mencionar. Por ejemplo, tiene capacidad de tolerancia a fallos, es decir, que tiene la capacidad de seguir funcionando aun cuando un servidor deje de funcionar. Esto se logra gracias a que soporta la replicación de datos entre clusters, tal como lo hemos visto con HDFS, por ejemplo.

A continuación, describiremos un ejemplo muy simple y resumido de HBASE.

En HBASE, una columna representa lo que llamaremos un atributo de un objeto.

La tabla que se muestra a continuación, contiene a los empleados de una empresa, la edad de cada uno de ellos, un parámetro que mide el esfuerzo en una hora laboral y la hora en la que se midió el parámetro.

Nombre del empleado	Edad del empleado	Medición de esfuerzo	Hora de la medición
Empleado 1	28	120/hora	7:35 a. m.
Empleado 2	35	100/hora	8:45 a. m.
Empleado 3	41	90/hora	9:55 a. m.
Empleado 4	24	105/hora	8:24 p. m.
....			

Figura 11. Tabla ejemplo de HBASE 1. (Elaboración propia, 2022)

La característica fundamental de HBASE es que permite que varios atributos puedan ser agrupados en una única **familia de columnas.**

Por ejemplo: Nombre del empleado y Edad del empleado, pueden almacenarse bajo el rótulo de "Detalle del empleado", lo cual podrá representarse de la siguiente forma:

Detalle del empleado		Medición de esfuerzo	Hora de la medición
Nombre del empleado	Edad del empleado		
Empleado 1	28	120/hora	7:35 a. m.
Empleado 2	35	100/hora	8:45 a. m.
Empleado 3	41	90/hora	9:55 a. m.
Empleado 4	24	105/hora	8:24 p. m.
....			

Figura 12. Tabla ejemplo de HBASE 2. (Elaboración propia, 2022)

La ventaja de HBASE es que esta familia de columnas se puede modificar en cualquier momento, tanto como se necesite. Se pueden agrupar, desagrupar, e incluso se pueden agregar nuevas columnas, en cualquier momento que el usuario (quien manipula los datos), lo desee.

Recordemos que lo aquí mencionado es sólo un ejemplo para comprender el criterio de cómo se almacena la información y cómo se puede manipular. En proyectos realmente grandes, la cantidad de columnas y filas que se operan son realmente muchas y de una magnitud significativa para los proyectos de big data. El punto es que con HBASE se logra eficiencia, redundancia y escalabilidad por la forma en la funciona.

Al igual que el resto de las tecnologías de Hadoop, HBASE también tiene su propia arquitectura, la cual describiremos a continuación:

La arquitectura HBASE consiste en 4 (cuatro) grandes componentes:

1. HMaster
2. HRegionServer
3. HRegions
4. ZooKeeper

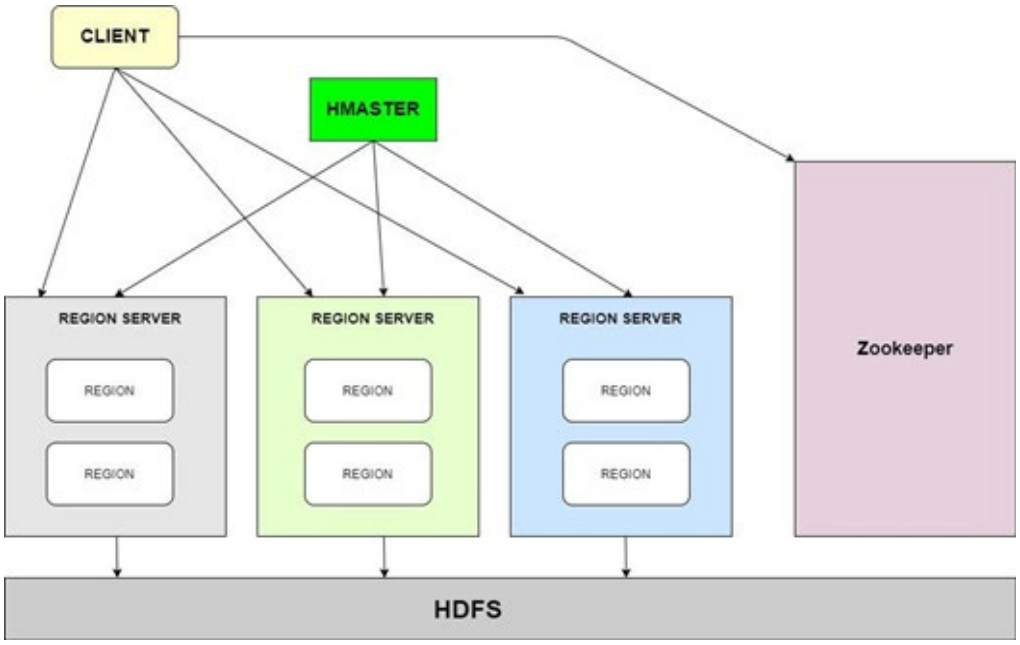


Figura 13. Arquitectura de HBASE. (Elaboración propia, 2022)

Vamos a describir el funcionamiento de la arquitectura HBASE a alto nivel. Lo primero que podemos observar, que se desprende de la figura, es que HBASE se ejecuta y corre perfectamente sobre el sistema de hadoop HDFS. De hecho, esto es así, porque el HDFS provee el entorno de almacenamiento distribuido que se necesita para trabajar.

Uno de los elementos primordiales de la arquitectura HBASE es el denominado HMASTER, el cual se refiere al servidor que oficia el rol de MASTER o Servidor Maestro. Su rol principal será monitorear las diferentes instancias de los denominados “Region Servers” o servidores Regionales. Esto es importante porque en una

implementación de HBASE pueden existir numerosos REGIONS Servers.

El “HMaster”, también cumple el rol de asignar los REGIONs (son los equipos que se encuentran dentro de una región) a los REGIONS Servers e, incluso, intercambiarlos, si fuera necesario. Esta tarea, desde el punto de vista de la arquitectura e infraestructura tecnológica, es primordial para quienes se ocupan de la salud de los equipos.

Los “Region Servers” son los que reciben las peticiones de lectura y de escritura por parte del cliente y se encargan de asignar el requerimiento a una región específica donde las columnas residen. Es decir, son los encargados de buscar las columnas, las filas, la información, estén donde estén. Recordemos que la información se almacena de manera distribuida. Por ejemplo, las columnas pueden estar almacenadas en diferentes servidores, y, por eso, la importancia de enviar peticiones para buscarlas y consolidarlas.

Por último, dentro de la arquitectura HBASE, nos encontramos con Zookeeper. Éste es un servicio que básicamente cumplirá la función de centralizar y mantener la salud y la configuración de los diferentes nodos. Este módulo nos permitirá sincronizar e identificar posibles fallas en los equipos (servidores), al mismo tiempo que nos podrá alertar sobre cualquier otro error que se produzca en los nodos. Recordemos que la cantidad de nodos es muy grande y, por lo tanto, es muy recomendable contar con alguna aplicación que simplifique la gestión operativa de cada uno.

Con Zookeeper, también lograremos que los clientes se puedan comunicar con los Region Servers, para obtener respuestas o consultas.

BIBLIOGRAFÍA

ANGUS, A. Las 10 principales tendencias globales del consumidor para 2018. Factores que impulsan cambios en el comportamiento de los consumidores.

AGUILAR, L. J. Big Data, análisis de grandes volúmenes de datos en organizaciones. 1era edición. Ed. AlfaOmega.

LANEY, D. (2001) Gestión de datos 3D: control del volumen, la velocidad y la variedad de datos. Nota de investigación del grupo META, 6.

QUÉ ES HUGE DATA Y POR QUÉ IMPLICA IOT, IA Y BLOCKCHAIN. (7 DE FEBRERO DE 2019). Obtenido de https://www.kanlli.com/estrategia-marketing-digital/que-es-huge-data/ el 17/01/2022

REVISTA HARVARD DEUSTO BUSINESS (2019) Nro 285.

REFERENCIAS

1. https://hadoop.apache.org/
2. https://hive.apache.org/
3. https://hbase.apache.org/index.html
4. http://static.googleusercontent.com/media/research.google.com/es//archive/bigtable-osdi06.pdf
5. https://zookeeper.apache.org/
6. https://intellipaat.com/blog/tutorial/hbase-tutorial/
7. https://www.bbvaapimarket.com/es/mundo-api/bigtable-el-servicio-de-base-de-datos-nosql-con-el-que-google-quiere-dominar-los-big-data/

ADEN