Arquitecturas BigData y Cloud Computing

‘Big Data & Apache Hadoop’

Autor: Pedro Pablo
Malagón Amor

Actualizado Diciembre 2022

¿Qué es Bigdata?

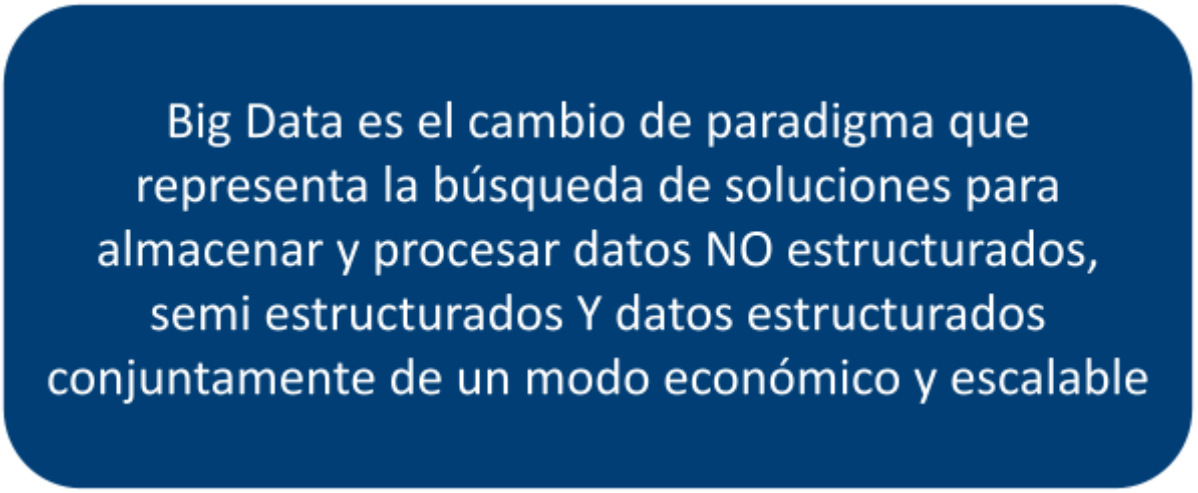
El término Bigdata se utiliza hoy en día muy ligado a la tecnología de Apache Hadoop, como pieza tecnológica core, para su implementación.

Es importante comprender que la tecnología de Hadoop es utilizada principalmente para las tareas de:

- Almacenamiento de cualquier tipo de datos, ya sean estructurados, semiestructurados o estructurados.

- Procesamiento distribuido de las operaciones sobre estos datos.

Por lo tanto, la utilización de la tecnología Apache Hadoop, nos va a permitir de forma económicamente viable el almacenamiento de grandes volúmenes de información, terabytes, petabytes…, independientemente del formato que tengan (documentos, texto, imágenes, video, etc.) así como la capacidad de realizar operaciones, consulta etc etc, sobre estos datos de forma escalable y económicamente viable.



BigData NO es…:

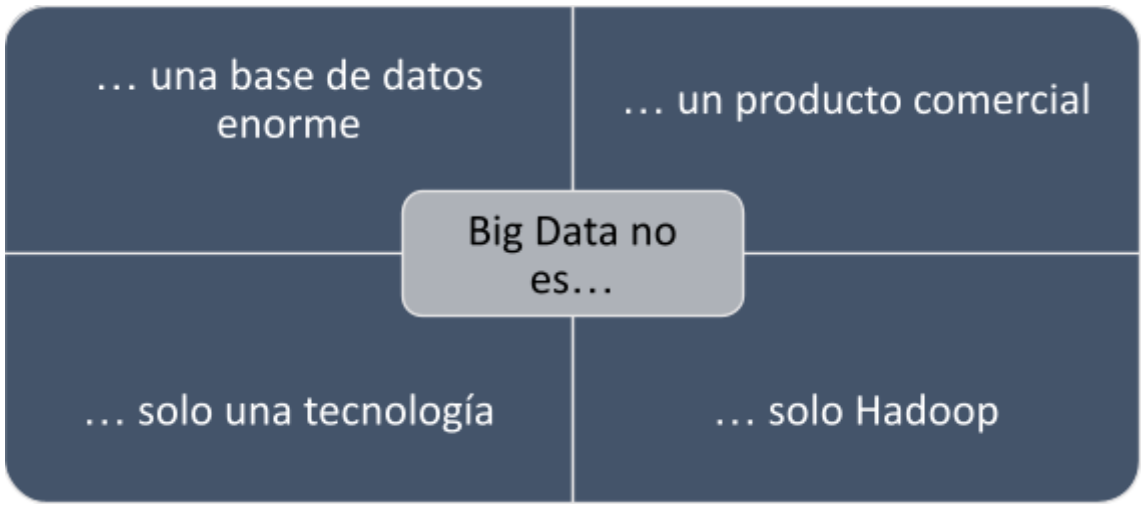
Es importante también saber QUE ES y QUE NO ES un proyecto de Big Data, dado que el término BigData se podría aplicar a casi cualquier problemática, en realidad no lo es.

Lo primero es que Big Data NO es, NO es SOLO Hadoop, dado que Apache Hadoop es una pieza tecnológica asociada a los proyectos de BigData, pero no es exclusivamente eso, dado que Hadoop, nos proporciona las soluciones de almacenamiento y procesamiento distribuido, pero no es la única solución ni es necesario que lo sea de forma exclusiva.

Lo segundo que Bigdata NO es, NO es una base de datos enorme, no es una evolución de los sistemas SQL u OLAP, sino que es una forma de separar la computación del almacenamiento de los datos, sin tener las restricciones de un motor de base de datos tradicional.

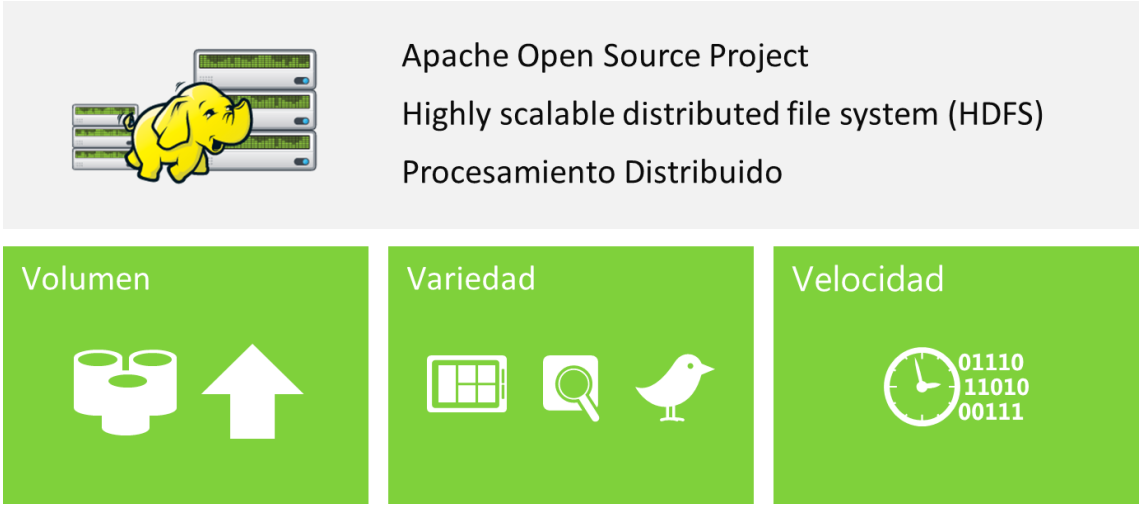
Lo tercero que bigdata NO es, NO es un producto comercial, la base tecnológica es open source, creado por la comunidad de la fundación Apache, donde el proyecto de Hadoop ha ido creciendo, aunque a partir de esa base tecnológica diferentes compañías han desarrollado productos, algunos open source otros comerciales, para ser utilizados.

Lo cuarto que BigData NO es, NO es solo una tecnología, sino que es un concepto que agrupa un compendio de tecnologías para dar solución a problemas de gran variedad de tipos de datos, con el volumen que sea necesario y de forma rápida y económicamente viable.

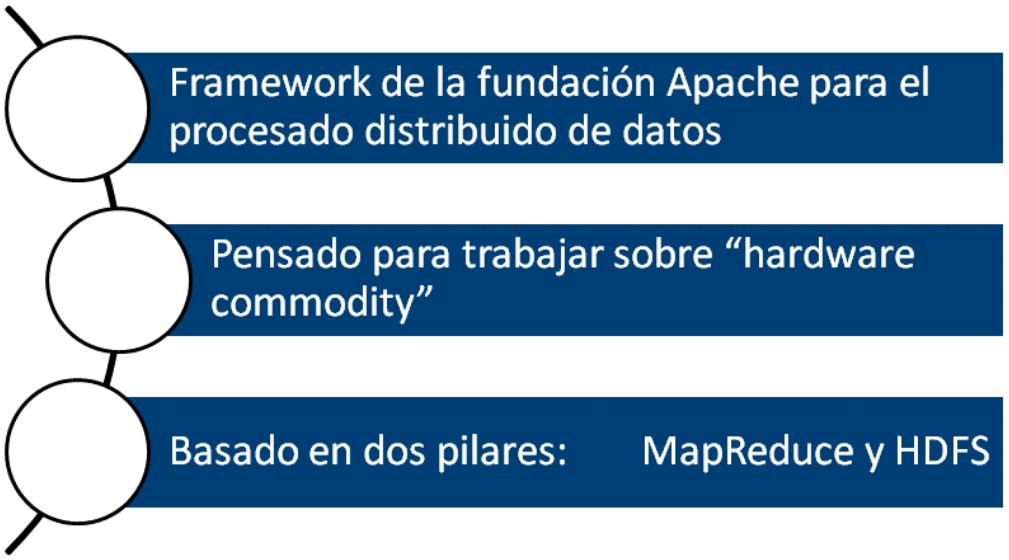


Apache Hadoop

De forma resumida, el proyecto de Apache Hadoop es el compendio de tecnologías open source de la comunidad que proporcionan la solución tecnológica para nuestros proyectos de Bigdata.



El proyecto de apache hadoop nos proporciona un framework pensado para poder ejecutarse sobre cualquier tipo de hardware, sin requerimientos particulares y que además pueda ser heterogéneo, además crea dos de los pilares principales de Hadoop, MapReduce y HDFS.

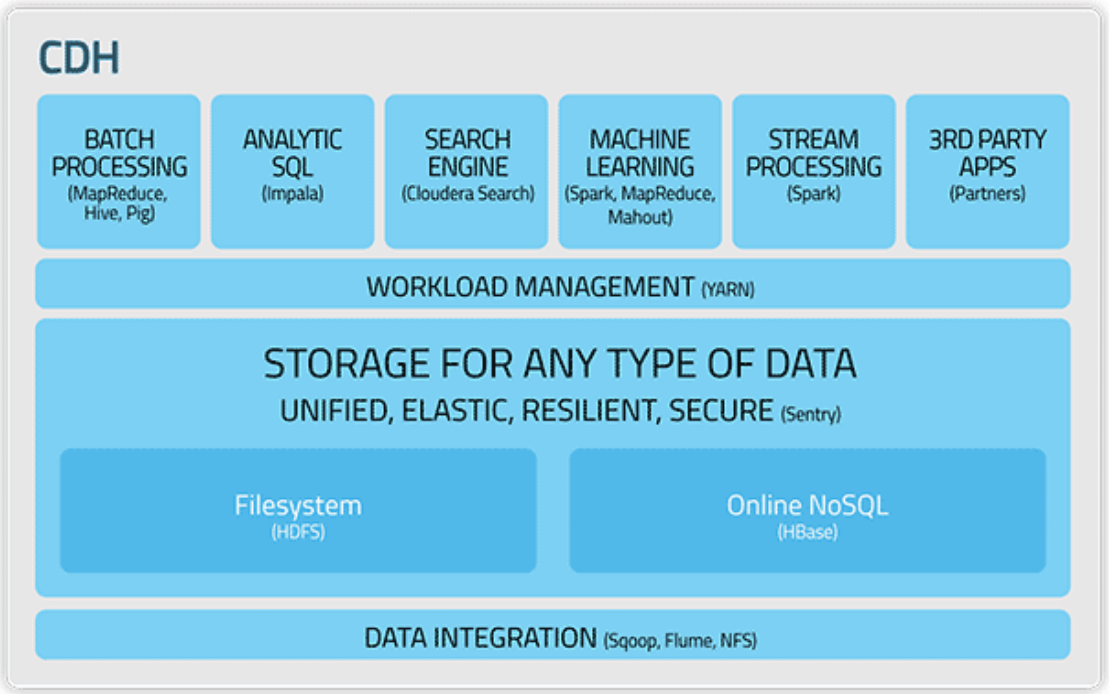


Distribuciones de Apache Hadoop

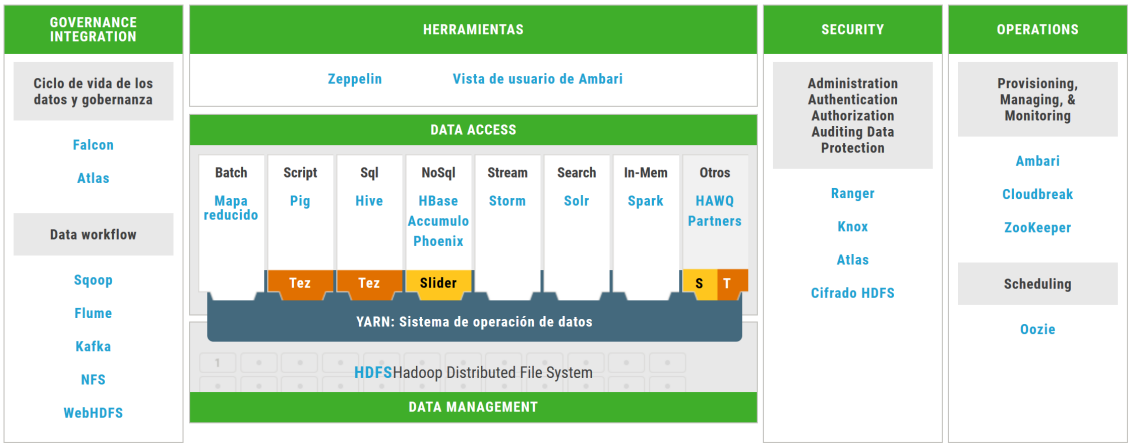
A día de hoy, hay varias empresas que han empaquetado el core de Hadoop de la fundación apache, le han añadido algunas piezas para su gestión instalación administración, gobierno y securización. Y lo distribuyen bajo diferentes nombres con modelos de comercialización diferentes.
Las más populares en este momento son tres de ellas:

cloudera

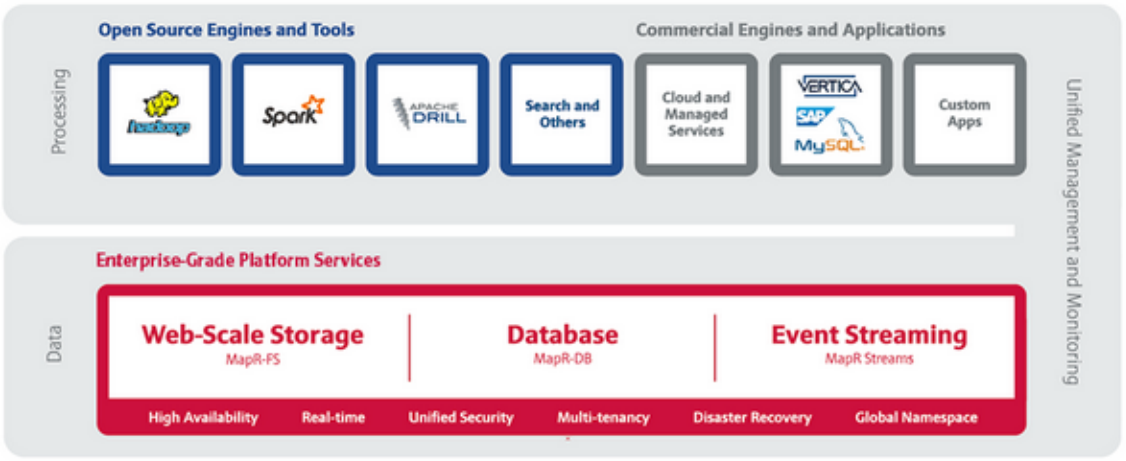
Cloudera Enterprise Data Hub, la cual fue una de las primeras y es de las más activas en el desarrollo de Hadoop, proporciona un entorno de administración, despliegue y funciones como Impala y los módulos de seguridad de Sentry que la hacen muy popular.



Hortonworks Data Platform, es una de las distribuciones más importantes y de las más activas en el desarrollo de Hadoop, su despliegue totalmente open source basado en Ambari, la implementación de Tez y el soporte de Ranger en los aspectos de seguridad la hacen muy popular.



MapR Converged Data Platform, es la siguiente distribución más utilizada actualmente, con componentes particulares sobre el core de hadoop como su base de datos MapR-DB, su enfoque al soporte a escenarios en tiempo real con MapR Streams y el soporte y contribución a proyectos como Apache drill para realizar consultas combinando datos de Hadoop, la hacen diferenciarse de las demás distribuciones.



Google DataProc

Google Cloud Dataproc es un servicio de Apache Hadoop, Apache Spark, Apache Pig y Apache Hive para procesar sin esfuerzo grandes conjuntos de datos a bajo coste. Puedes crear clústeres administrados de cualquier tamaño y desactivarlos cuando acabes para controlar los costes.

Google Cloud Dataproc se integra en todos los productos de Google Cloud Platform y ofrece una plataforma de procesamiento de datos potente y completa.

Puedes crear rápidamente clústeres de Cloud Dataproc y cambiar su tamaño en cualquier momento (desde tres nodos a cientos de ellos). Así te despreocupas de que las canalizaciones de tus datos sobrepasen los clústeres.
Como cada acción de clúster tarda menos de 90 segundos de media, dispones de más tiempo para centrarte en la información y pierdes menos tiempo supervisando la infraestructura.

El ecosistema de Spark y Hadoop proporciona herramientas, bibliotecas y documentación que puedes aprovechar con Cloud Dataproc. Como ofrece versiones actualizadas y nativas de Spark, Hadoop, Pig y Hive, no tienes que aprender herramientas ni API nuevas. Además, puedes mover proyectos o canalizaciones ETL sin necesidad de volver a desarrollarlas.

Cloud Platform analiza y procesa datos clave con mayor escala, eficiencia y simplicidad. Si usas Hive en Hadoop (o SparkSQL), te recomendamos Google BigQuery, un servicio de análisis de SQL según demanda con un gran rendimiento. Si programas canalizaciones de transformación de datos con Spark o MapReduce, entonces te aconsejamos Google Cloud Dataflow, un servicio totalmente administrado que elimina el trabajo que requieren otras herramientas y que ejecuta una gran variedad de patrones de procesamiento de datos, como el ETL y las operaciones informáticas, tanto continuas como por lotes.

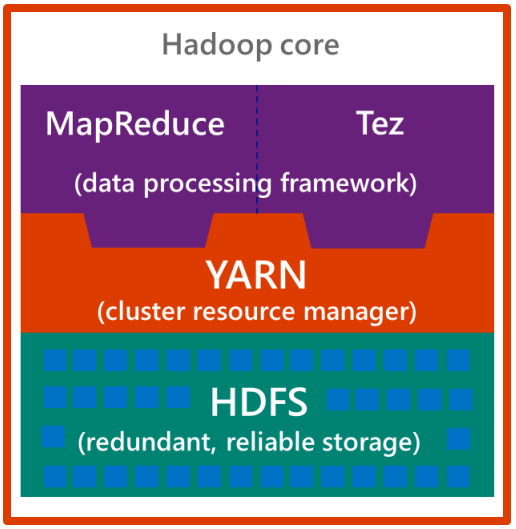
La semana próxima estaremos centrados en DataProc y BigQuery, por lo tanto entraremos en más detalles en unos dia.

Capacidades y Componentes de Hadoop

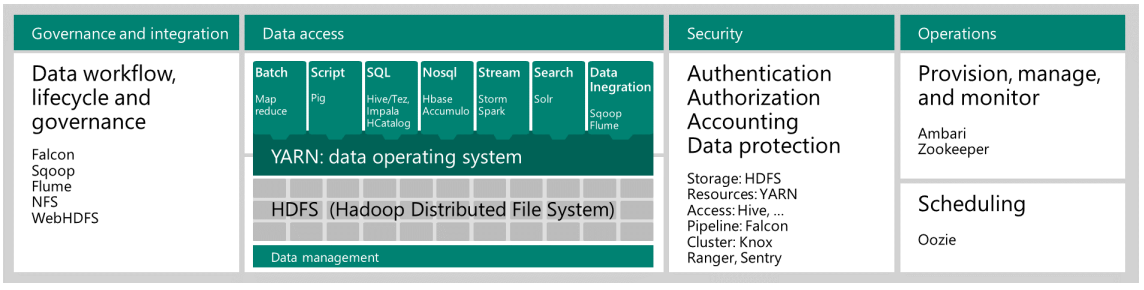
El proyecto de apache Hadoop contempla piezas tecnológicas para dar solución a diferentes escenarios.

Hadoop, desde el punto de vista técnico, lo definiremos como:
Un marco de programación altamente fiable, distribuido y paralelizable para el análisis de entornos BigData.

El core de Hadoop lo forman las piezas que se puede observar en la siguiente figura:



- Hadoop es un proyecto open source de la fundación apache, basado en el lenguaje de programación Java
- El Core de Hadoop incluye:
  - Un sistema de ficheros escalable y seguro (HDFS)
  - Un framework que permite el desarrollo de programas basados en MapReduce (MR)
  - YARN, un gestor de recursos distribuidos que proporciona y controla el acceso a los recursos del cluster de Hadoop
- Además del Core, Hadoop tiene un gran ecosistema de componentes que añaden soporte para SQL/NoSQL, streaming, real-time y aplicaciones interactivas.



Ambari

Apache Ambari sirve para el aprovisionamiento, la administración y la supervisión de clústeres de Hadoop de Apache. Incluye una recopilación intuitiva de herramientas de operador y un conjunto sólido de APIs que ocultan la complejidad de Hadoop y simplifican la operación de clústeres.

HDFS

El sistema de archivos distribuido de Hadoop (HDFS) es un sistema de archivos distribuido que, junto con MapReduce y YARN, conforma el núcleo del ecosistema Hadoop. HDFS es el sistema de archivos estándar para los clústeres de Hadoop.

Hive

Apache Hive es el software de data warehouse de datos basado en Hadoop que nos permite consultar y administrar grandes conjuntos de datos en HDFS, con un lenguaje de tipo SQL denominado HiveQL. Hive, al igual que Pig, es una abstracción situada en un nivel superior a MapReduce y, al ejecutarse, traduce las consultas a una serie de trabajos de MapReduce.
Hive se encuentra conceptualmente más cerca de un sistema de administración de bases de datos relacionales que Pig y, por lo tanto, es más adecuado para su uso con datos más estructurados.
Para los datos no estructurados, Pig es una mejor opción.

HCatalog

Apache HCatalog es una capa de administración de almacenamiento y tablas para Hadoop que presenta a los usuarios una vista relacional de los datos. En HCatalog, se pueden leer y escribir archivos en cualquier formato para el que se pueda escribir un SerDe de Hive (serializador-deserializador).

Mahout

Apache Mahout es una biblioteca escalable de algoritmos de Machine Learning, que se ejecuta en Hadoop. Mediante principios de estadísticas, las aplicaciones de Machine Learning enseñan a los sistemas a aprender de los datos y a usar los resultados obtenidos en el pasado para determinar el comportamiento en el futuro

MapReduce

MapReduce es el marco de software heredado para permitir a Hadoop la escritura de aplicaciones con el fin de procesar en lote conjuntos de datos en paralelo. Un trabajo de MapReduce divide conjuntos de datos de gran tamaño y organiza los datos en pares clave-valor para su procesamiento.

MapReduce 2 o YARN es el marco de aplicación y administrador de recursos de última generación, al que se conoce como MapReduce 2.0. Los trabajos de MapReduce se ejecutan en YARN.

Oozie

Apache Oozie es un sistema de coordinación de flujos de trabajo que administra trabajos de Hadoop. Se integra con la pila de Hadoop y es compatible con los trabajos de Hadoop para MapReduce, Pig, Hive y Sqoop. También puede usarse para programar trabajos específicos de un sistema, como scripts de shell o programas Java.

Phoenix

Apache Phoenix es una capa de base de datos relacional sobre HBase. Phoenix incluye un controlador JDBC que permite a los usuarios consultar y administrar tablas SQL directamente. Phoenix traduce consultas y otras instrucciones en llamadas nativas de la API NoSQL nativas en lugar de usar MapReduce, lo que permite aplicaciones más rápidas además de los almacenes NoSQL.

Pig

Apache Pig es una plataforma de alto nivel que permite realizar transformaciones complejas de MapReduce en conjunto de datos de gran tamaño con un sencillo lenguaje de scripting denominado Pig Latin. Pig traduce los scripts de Pig Latin para que se ejecuten en Hadoop. Se pueden crear funciones definidas por el usuario (UDF) para ampliar Pig Latin.

Sqoop

Apache Sqoop es una herramienta que transfiere grandes cantidades de datos entre Hadoop y bases de datos relacionales, como SQL, u otros almacenes de datos estructurados, de la manera más eficiente posible.

Tez

Apache Tez es un marco de aplicación basado en Hadoop YARN que ejecuta procesos complejos de procesamiento de datos. Es un sucesor más flexible y eficaz al marco de MapReduce que permite procesos intensivos de datos, como Hive, para ejecutar de manera más eficaz a escala.

YARN

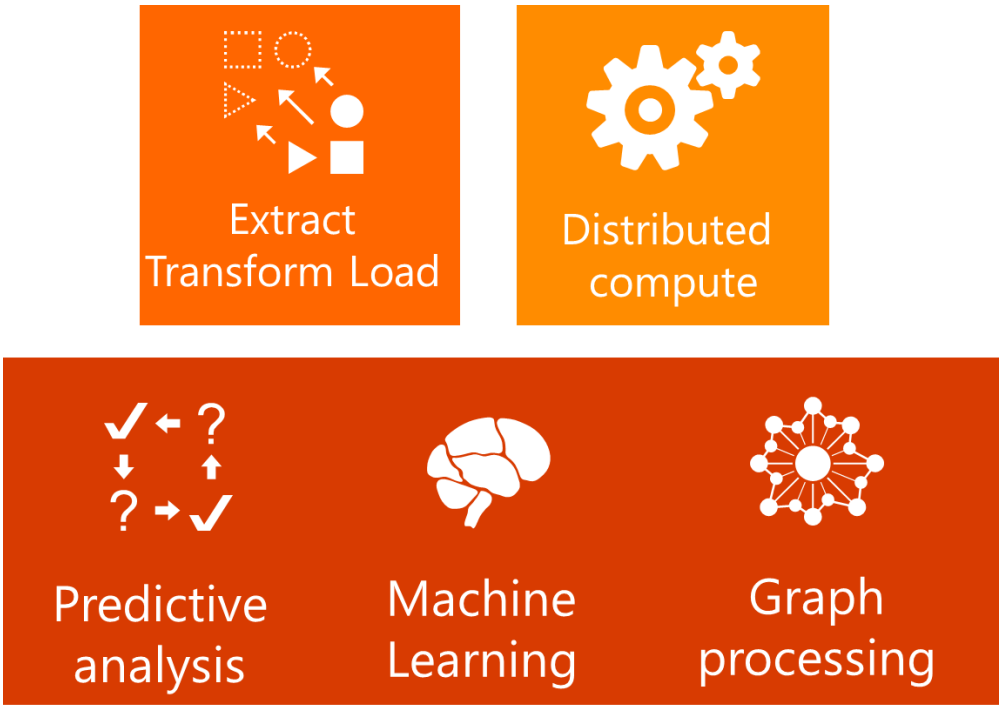
Apache YARN es la siguiente generación de MapReduce (MapReduce 2.0, o MRv2) y es compatible con escenarios de procesamiento de datos más allá del procesamiento por lotes de MapReduce con mayor escalabilidad y procesamiento en tiempo real. YARN proporciona un

marco de aplicación distribuida y administración de recursos. Los trabajos de MapReduce se ejecutan en YARN.

ZooKeeper

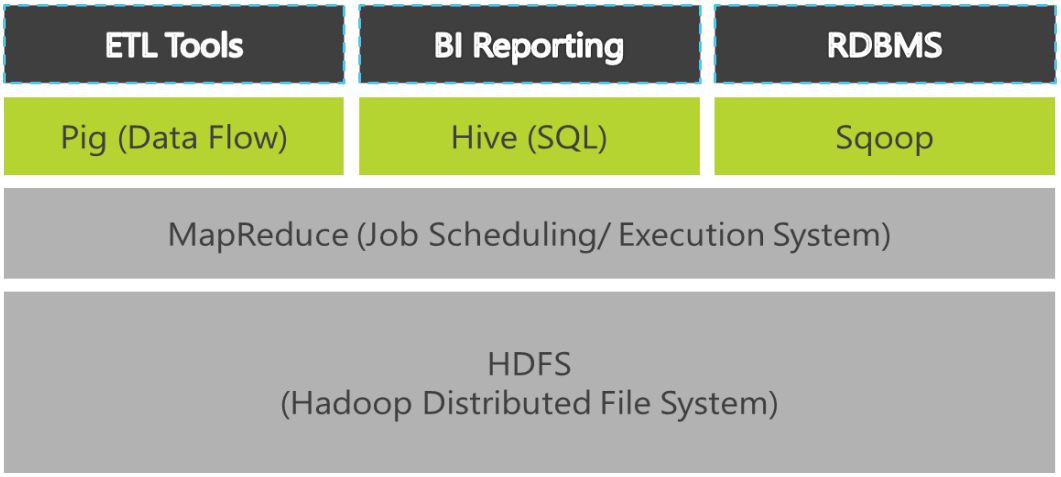
Apache ZooKeeper coordina procesos en grandes sistemas distribuidos por medio de un espacio de nombres jerárquico compartido de registros de datos (znodes). Znodes contiene pequeñas cantidades de metadatos necesarios para la coordinación de procesos: estado, ubicación, configuración, entre otros.

Por lo tanto, como podemos ver claramente, con hadoop podemos dar solución a escenarios de integración de datos, de diferentes formatos, tipos, en los cuales queramos poder cruzar y hacer consultas sobre esos datos, como por ejemplo los archivos de logs de un web site, trazas de los dispositivos móviles conectados y videos e imágenes de las cámaras de seguridad, utilizando el módulo de Hadoop conocido como PIG podemos abordar este tipo de proyectos.

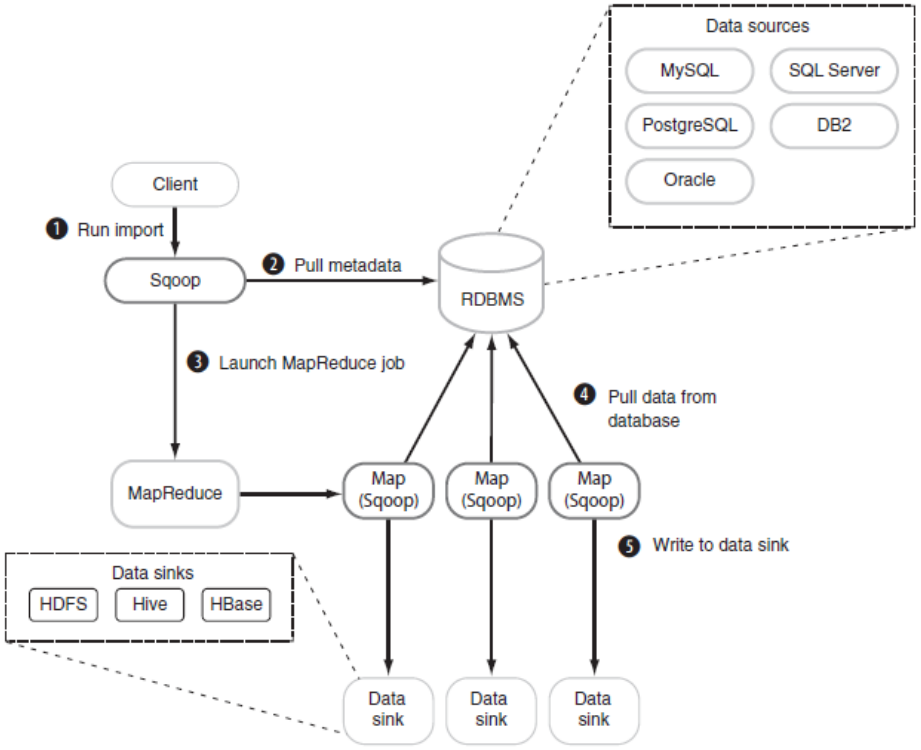


Además, por ejemplo, nos permite aplicar algoritmos estadísticos de análisis predictivo, sobre los datos anteriormente almacenados en sistema de archivos de hadoop, el cual se conoce como HDFS.

Un escenario muy típico es realizar consultas sobre los datos almacenados de forma similar a como haríamos sobre un motor de base de datos como SQL, en este caso hadoop proporciona módulos como HIVE para poder lanzar consultas tipo SELECT sobre los archivos almacenados en el sistema de archivos HDFS.



Otra de las áreas importantes a tener en cuenta dentro de un proyecto de BigData, es la integración con bases de datos, como SQL Server, Oracle, DB2, Mysql…
Este tipo de integraciones se pueden realizar mediante la utilización del componente de Hadoop, SQOOP, que es una utilidad sencilla de hadoop, que nos sirve para transformar y transferir datos entre una base de datos relacional y Hadoop.



Como capacidades principales de SQOOP :

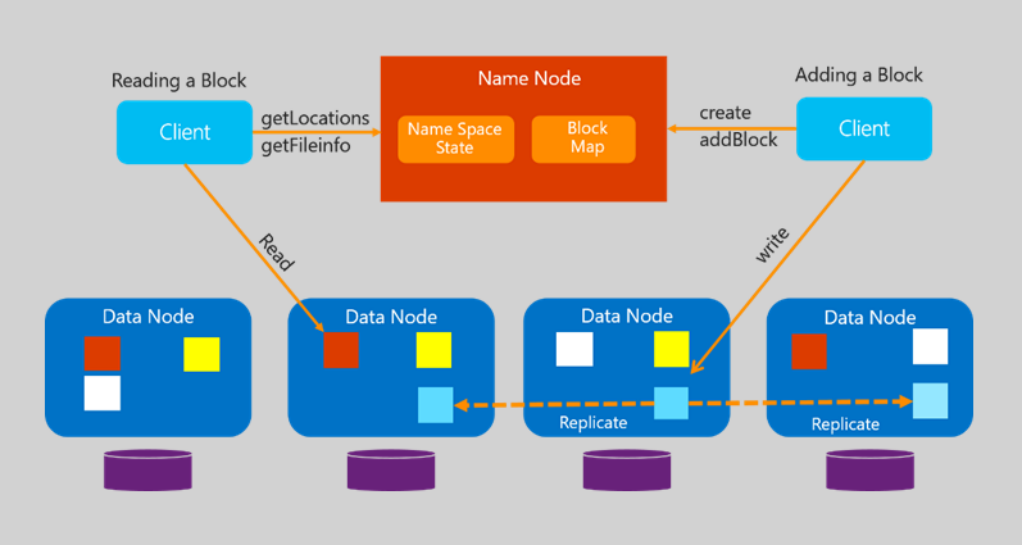
1. Soporta importaciones incrementales
2. La función Import mueve datos de una base de datos relacional hacia Hadoop
3. La función Export mueve datos desde Hadoop a una base de datos relacional

Sistema de Ficheros HDFS.

Uno de elementos principales a configurar y utilizar correctamente es el sistema de ficheros de Hadoop, llamado HDFS.
HDFS Nos proporciona las siguientes funcionalidades a modo de resumen:

- Sistema de Ficheros distribuido
- Construido sobre Hardware no específico
- Alta Resistencia a fallos
    - Replicación de ficheros
    - Detección y Recuperación automática
- Optimizado para procesos por lotes (batch)
    - Lista de ubicaciones expuesta para minimizar tráfico

Para realizar estas tareas los componentes de HDFS son los siguientes como se puede apreciar en la figura siguiente:



Como resumen para entender el porqué de estos componentes, las tareas principales que realiza cada uno de ellos son las siguientes:

Name Node

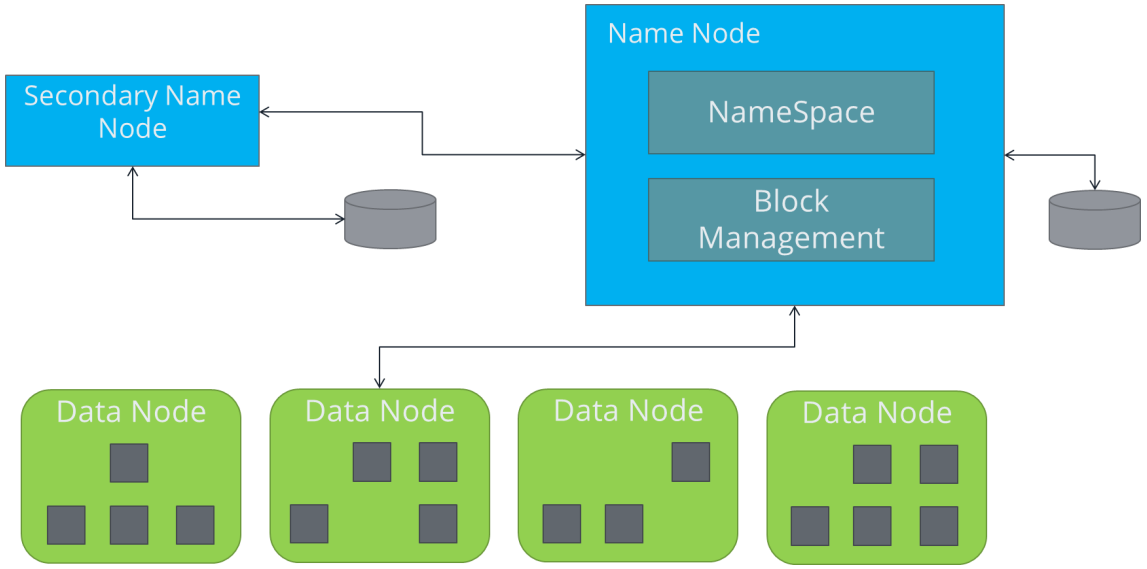
- Gestiona los Data Nodes
- Guarda metadatos para todos los ficheros y bloques
- Mantiene los namespaces en memoria
- Monitoriza el estado de salud de los data node
- Replica los bloques perdidos o dañados
- Mantiene un mapeo de la lista de bloques<-> localizaciones fisicas
- Matiene los datos de utienticaciòn
- Mantiene una imagen del sistema de ficheros en memoria
- Utiliza un log de transacciones (EditLog) para almacenar cambios en el Sistema de ficheros (nuevos ficheros, cambios en el numero de replicas, etc…)
  - Se almacena en el Sistema de ficheros local del Name Node
- El Sistema de ficheros completo, incluyendo el mapeo de bloques y demás metadatos, se almacena en un fichero FsImage
  - También se almacena en el Sistema de ficheros local del Name Node
- Utiliza un Sistema de checkpoints para poder recuperar el sistema en caso de fallo
  - En cada arranque, recupera FsImage, lo actualiza con la información de EditLog y almacena una copia de FsImage como checkpoint

Data Nodes

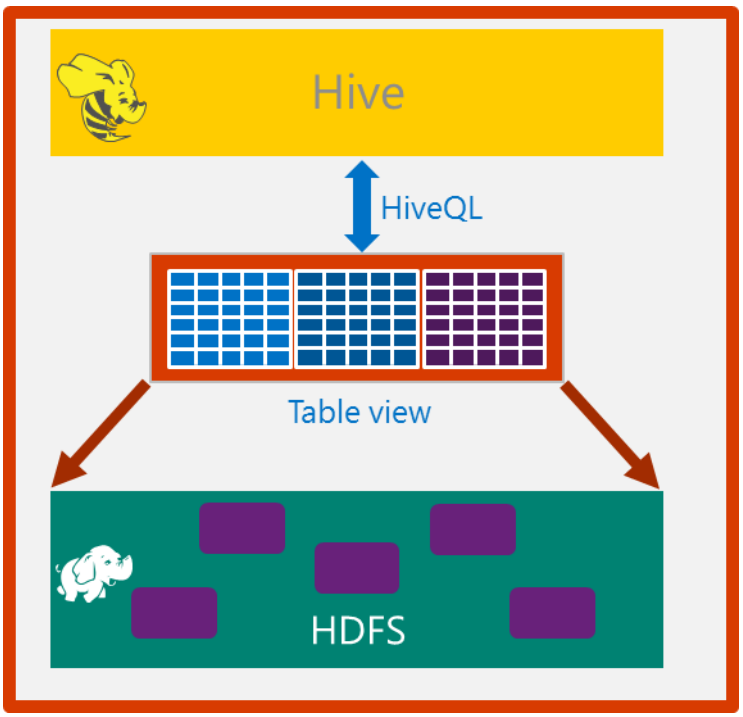
- Almacenan los bloques de datos
- Se distribuyen por la topología de racks
- Sirve los bloques de datos directamente a los clientes
- Mantiene la integridad de los cloques almacenados en multiples volumenes
- Periodicamente enviar señales de control e informes de cloques a los NameNode
- Almacena los quoques bajo el sistema de archivos del sistema operativo.
- No crea todos los ficheros en el mismo directorio
  - Utiliza un algoritmo para calcular en numero optimo de ficheros por directorio, creando directorios nuevos a medida que los necesita
- Cuando arranca, genera una lista de todos los bloques y se los envía al Name Node como BlockReport

Por lo tanto, HDFS, que recordamos, es un proyecto open source de la fundación apache, es quien organiza los datos en ficheros y directorios, de forma que esta optimizado para un acceso rápido a los datos. HDFS divide los ficheros en bloques del mismo tamaño, el cual puede ser configurado y divide los ficheros entre los diferentes nodos del cluster, pudiendo re-balancear los bloques de datos utilizando diferentes modelos.

Los bloques por defecto son replicados, en 3 réplicas, pudiendo modificar el número de réplicas, esto nos permite manejar de forma adecuada el rendimiento y los fallos.
Por lo tanto, HDFS nos permite implementar la filosofía de "mover la computación al dato".



HIVE.



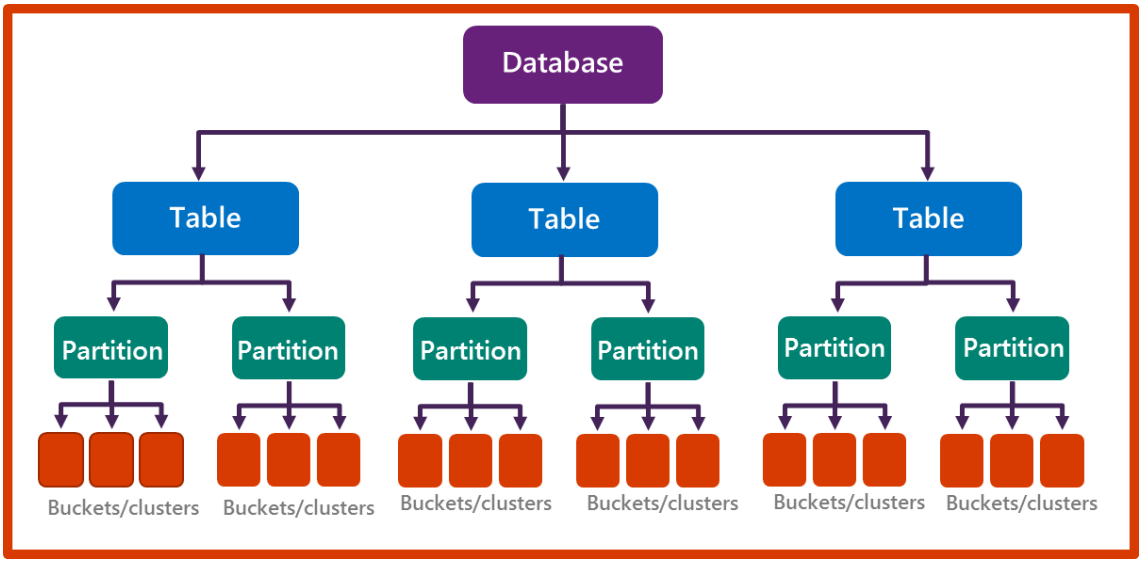
Hive, como resumen para nosotros, tiene como puntos principales:

- Es un sistema de data warehouse para Hadoop
- Nos permite creamos las definiciones de los esquemas y tablas para apuntar a los datos de Hadoop
- Nos permite tratar tus datos de Hadoop como tablas
- El lenguaje de programación es compatible SQL 92
- Nos proporciona una herramienta de Consultas Interactivas

Apache Hive, es una infraestructura de data warehouse sobre Apache Hadoop que nos proporciona acceso a los datos, consultas y análisis de grandes conjuntos de datos como es premisa de BigData.
Esto nos permite proyectar una serie de estructuras sobre los datos de Hadoop y lanzarles consultas a esos datos, utilizando un lenguaje tipo SQL llamado HiveQL (HQL).

Las tablas en Hive, son similares las tablas en una base de datos relacional, y los datos son organizados en una taxonomía de mayor a menor.
En Hive, se define una estructura, donde las bases de datos están formadas por tablas, las cuales están formadas por particiones.

A su vez, los datos serán accedidos vía el lenguaje de HiveQL el cual soporta añadir y sobrescribir datos.
En la figura siguiente podemos observar una representación de la estructuración de HIVE



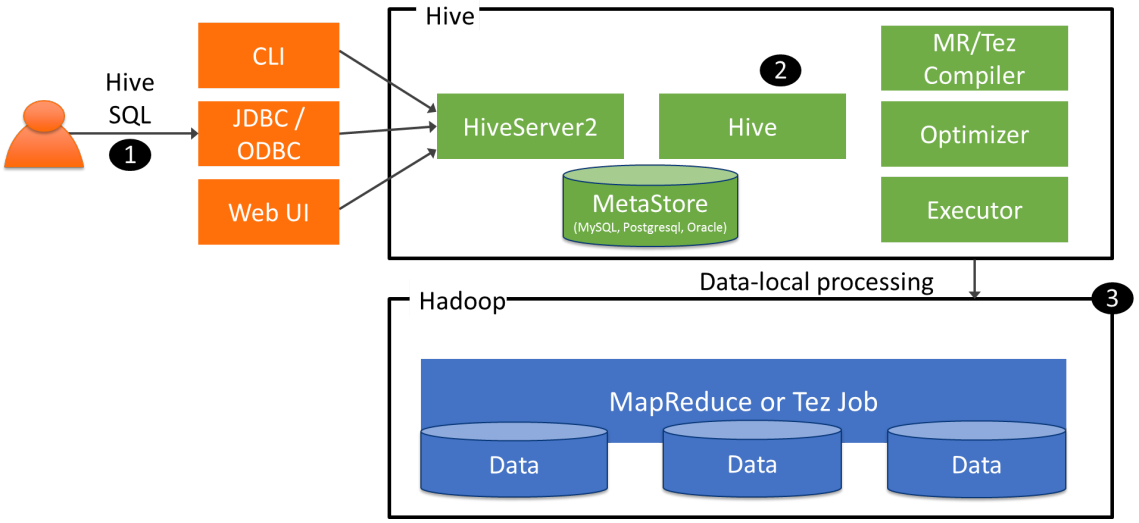
Los datos en las tablas son serializados y cada tabla tiene su propio directorio HDFS Hadoop Distributed File System (HDFS).
Cada tabla puede ser subdividido en particiones que determina cómo se distribuyen los datos en subdirectorios del directorio de la tabla.
Los datos con particiones pueden ser a su vez divididos en cubos.
Hive soporta todos los tipos de datos primitivos más comunes como BIGINT, BINARY, BOOLEAN, CHAR, DECIMAL, DOUBLE, FLOAT, INT, SMALLINT, STRING, TIMESTAMP y TINYINT.

SQL Datatypes	SQL Semantics
INT	SELECT, LOAD, INSERT from query
TINYINT/SMALLINT/BIGINT	Expressions in WHERE and HAVING
BOOLEAN	GROUP BY, ORDER BY, SORT BY
FLOAT	CLUSTER BY, DISTRIBUTE BY
DOUBLE	Sub-queries in FROM clause
STRING	GROUP BY, ORDER BY
BINARY	ROLLUP and CUBE
TIMESTAMP	UNION
ARRAY, MAP, STRUCT, UNION	LEFT, RIGHT and FULL INNER/OUTER JOIN
DECIMAL	CROSS JOIN, LEFT SEMI JOIN
CHAR	Windowing functions (OVER, RANK, etc.)
VARCHAR	Sub-queries for IN/NOT IN, HAVING
DATE	EXISTS / NOT EXISTS

Por supuesto además podemos combinar los tipos de datos primitivos para formar datos complejos como structs, maps y arrays.

¿Cómo funcionan las consultas dentro de Hive?

El proceso que sufre una consulta de Hive sobre tu sistema de Hadoop es el siguiente:



- El Usuario inicia una consulta HiveSQL
- Hive recoge y gestiona la consulta
- La consulta es convertida en MapReduce y ejecutada en Hadoop

Las llamadas de las consultas son recogidas y enviadas por HiveServer2 pero el procesamiento real ocurren en los worker nodes del cluster de Hadoop

Para iniciar y probar el flujo de una consulta en Hive, tenemos múltiples herramientas tipo GUI para acceder a programar con Hive, aunque Ambari es la que utilizaremos.

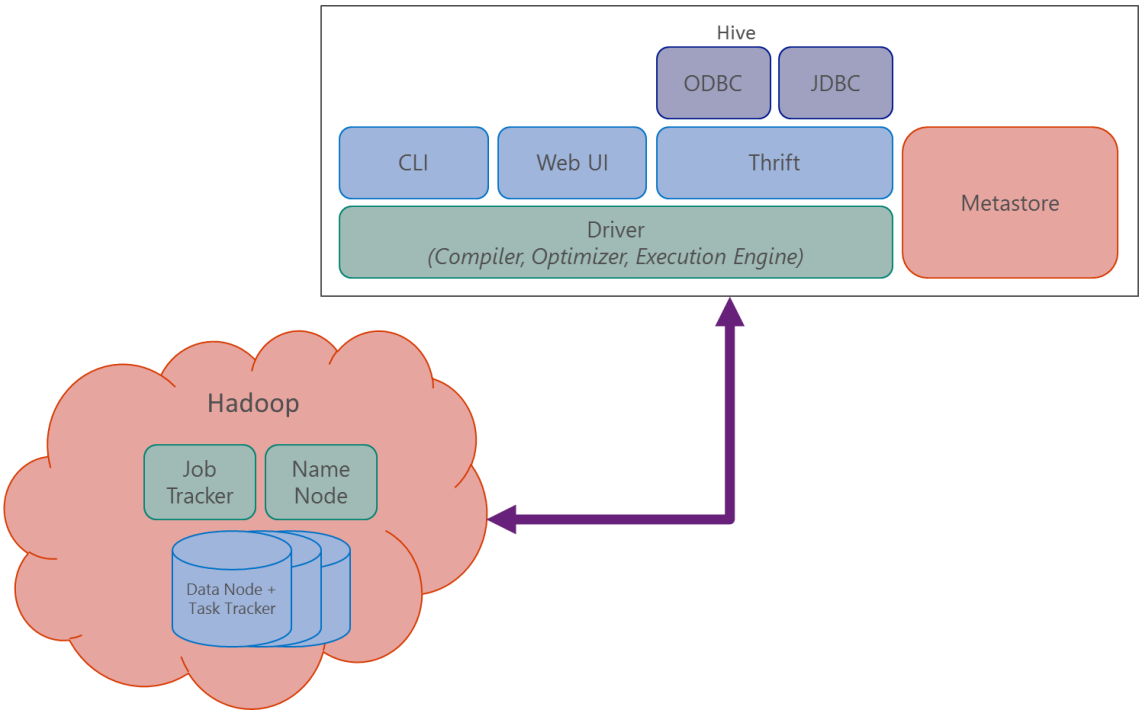
Open source SQL:

- Ambari
- Zeppelin (https://zeppelin.incubator.apache.org/)
- DBVisualizer (https://www.dbvis.com/download/)
- Dbeaver (http://dbeaver.jkiss.org/)
- SquirrelSQL (http://squirrel-sql.sourceforge.net/)
- SQLWorkBench (http://www.sql-workbench.net/)

¿Cuál es la Arquitectura de Hive?

Apache Hive, nos proporciona una infraestructura para la ejecución de las consultas HiveQL que nos abstraiga de Hadoop, tanto a nivel de HDFS como sobre todo de MapReduce o Tez, esto simplifica el desarrollo y acelera la capacidad de análisis sobre los datos almacenados en archivos de Hadoop

- Construido sobre Hadoop para proporcionar datos administración, consulta y análisis de los datos.
- Acceso y consulta de datos a través de declaraciones de SQL, llamadas consultas Hive.
- En Resumen, Hive, en concreto Hive Server versión 2, es quien consulta y Hadoop ejecuta



El proceso a seguir en un proyecto de Hive, consta de tres pasos al menos, que son:

Creación de un esquema de datos

- Los datos de las tablas Hive se almacenan en el warehouse
- Por defecto en HDFS: /user/hive/warehouse
- Las tablas se almacenan como subdirectorios del warehouse
- Las particiones se almacenan como subdirectorios de las tablas
- Soporte a tablas externas – LOCATION (afecta LOAD y DROP)
- NO olvides que los datos se almacenan en ficheros

Vamos a pasar a ver cómo serían los procesos de carga de datos en las tablas de Hive. Podemos cargar datos desde archivos locales o desde archivos de datos que ya se encuentren en HDFS

Ejemplo: Carga de datos desde archivos locales:
LOAD DATA LOCAL INPATH '/tmp/clientes.csv' OVERWRITE INTO TABLE clientes;

Ejemplo: Carga de datos desde archivos ya almacenados en HDFS:
LOAD DATA INPATH '/user/master/clientes.csv' OVERWRITE INTO TABLE clientes;

En este ejemplo insertamos los resultados de ejecución de una consulta con Select

```
INSERT INTO aniversarios
    SELECT nombre, apellidos, fechanacimiento
    FROM clientes
    WHERE fechanacimiento IS NOT NULL;
```

Podríamos incluso realizar la creación de una tabla a partir de una consulta “Create Table As Select (CTAS)” para definir y cargar la tabla en una sola operación

Vamos a pasar a analizar algunos ejemplos de consultas Hive para que os familiaricéis con el lenguaje de HiveQL:

Ejemplo: Crear una tabla de Hive

```
CREATE TABLE clientes (
                IDclientes INT,
                Nombre STRING,
                apellidos STRING,
                fechanacimiento TIMESTAMP,
 ) ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ',';
```

Ejemplo: Crear una tabla externa de Hive

Recordar que, en este tipo de tablas, no se borran los datos, aunque se borre la tabla

```
CREATE EXTERNAL TABLE sueldos(
  genero string,
  edad int,
  sueldo double,
) ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ',';
```

Ejemplo: Crear una tabla externa y definir donde su **LOCATION**

Si no se marca la propiedad LOCATION los datos de la tabla se almacenan en /apps/hive/warehouse/db_name/table_name

La cláusula LOCATION no está solo disponible para las tablas externas, pero es donde más se utiliza.

```
CREATE EXTERNAL TABLE sueldos(
  genero string,
  edad int,
  sueldo double,
) ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ',';
  LOCATION '/user/train/sueldos/';
```

Ejemplo: SELECT … FROM … WHERE … GROUP BY

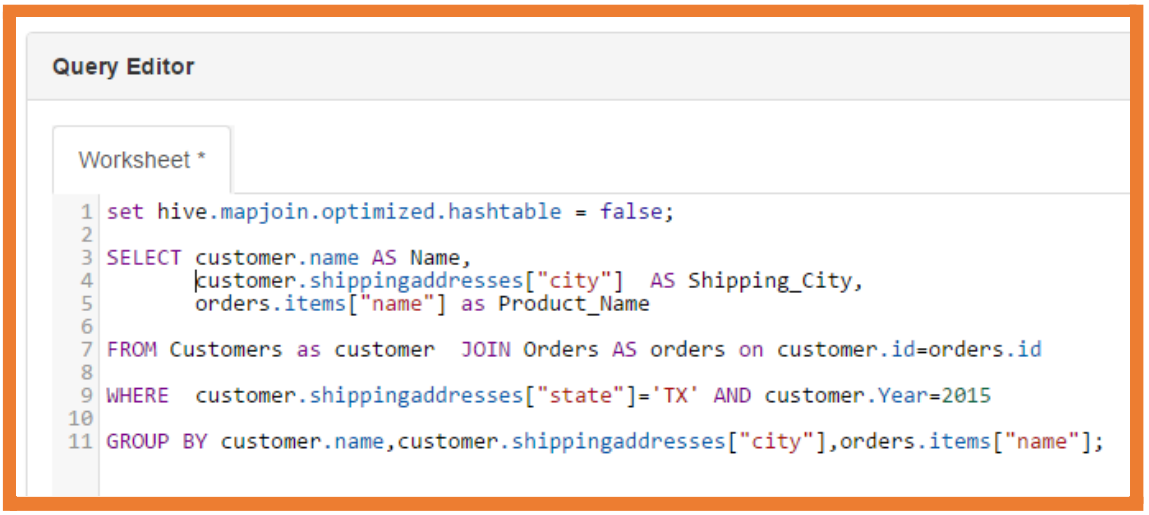
Esta Consulta lo que hace es:
"Encontrar el nombre y ciudad de facturación de los clientes con direcciones de envío en 'TX' o 'IL'. Agrupar por la ciudad"

Ejemplo: SET HIVE … SELECT … FROM … WHERE … GROUP BY

Esta Consulta lo que hace es:

Lo primero como veis es establecer a FALSE el parámetro que indica al servidor de HiveServer2, que no utilice una tabla optimizada para funcionar en memoria.

Y Encontrar los nombres de los productos comprados por los clientes cuya dirección de envío es en Texas (TX).

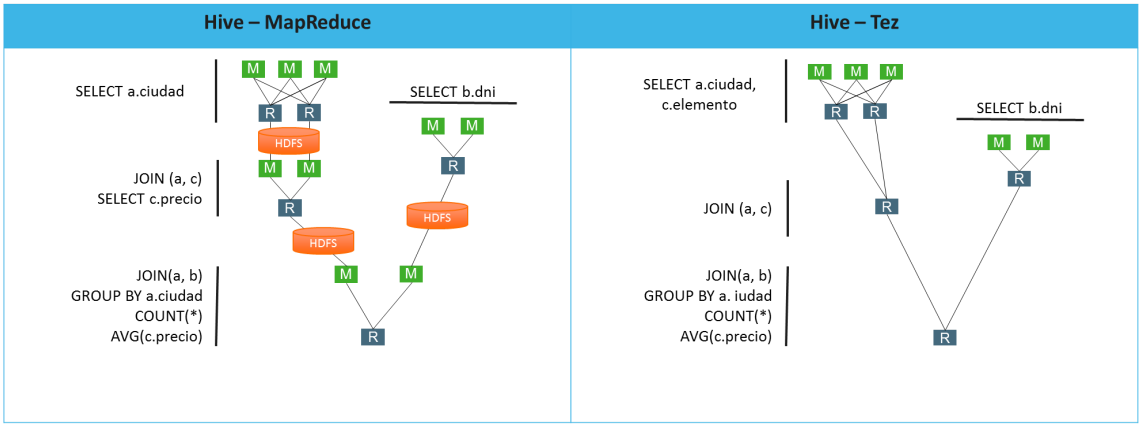


Un tema muy importante es la configuración del motor que queremos, que se ejecute por debajo de nuestras consultas de Hive, como hemos visto en la presentación podemos definir mediante los ficheros de configuración de Hadoop.

Por ejemplo, para la ejecución de la siguiente consulta de ejemplo podríamos modificar el parámetro de configuración.

set hive.execution.engine = mr; / set hive.execution.engine = tez;

```
SELECT a.ciudad, COUNT(*), AVG(c.precio)
FROM a
JOIN b ON (a.id = b.dni)
JOIN c ON (a.elemento = c.elemento)
GROUP BY a.ciudad
```



Es bueno pararse un momento y introduciros Apache™ Tez es un framework para el desarrollo de aplicaciones batch e interactivas, las cuales serán coordinadas por Hadoop. Tez mejora el paradigma de MapReduce dramáticamente mejorando su velocidad, manteniendo la capacidad de MapReduce de escalar hasta petabytes de datos.

Otros proyectos del ecosistema Hadoop, como Hive y Pig, usan Apache Tez.

Apache Tez proporciona un framework y un API para escribir aplicaciones YARN nativas. Esto nos permite que esas aplicaciones puedan trabajar con petabytes de datos y miles de nodos de Hadoop. La biblioteca de componentes de Apache Tez nos permite como desarrolladores crear aplicaciones de Hadoop que se integran nativamente con Apache Hadoop.

PIG.

Apache Pig es, Una plataforma de scripting basada en MapReduce o Tez, que sirve para analizar grandes conjuntos de datos almacenados en HDFS
PIG está formado por dos elementos principales:

- Un motor para ejecutar programas sobre Hadoop
- Proporciona un lenguaje, llamado Pig Latin, para estos programas.

Los principios de creación de Pig, haciendo una comparación con la palabra que define también al animal con el comparte nombre, “cerdo”, son:

- **(Los Cerdos) Pig comen de todo**
  Pig puede procesar cualquier tipo de datos, estructurados o no estructurados.

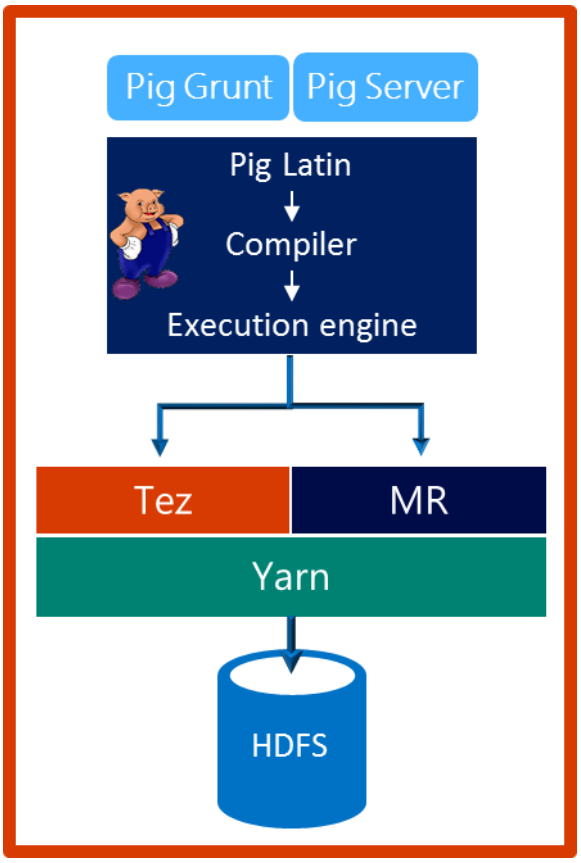
- **(Los Cerdos) Pigs vive en cualquier sitio**
  Pig puede ejecutarse sobre cualquier framework de procesamiento paralelo, por lo tanto los scripts de Pig no tiene que ejecutarse solo sobre Hadoop.

- **(Los Cerdos) Pigs son animales domésticos**
  Pig está diseñado para ser controlado y modificado de forma sencilla.

- **(Los Cerdos) Pigs vuelan.**
  Pig está diseñado para procesar los datos muy deprisa

La arquitectura de Pig, es la siguiente

Está formada por dos piezas a primer nivel que son el servidor de Pig y al menos una consola como Pig Grunt.
A continuación tenemos el leguaje de Pig Latín , con su compilador asociado para interpretar los comandos recibidos y finalmente el motor de ejecución.
Y todo esto se apoya como es lógico sobre los pilares de Hadoop, que son Tez, Mapreduce sobre Yarn y con los datos persistidos en ficheros sobre HDFS.



Como podemos deducir, Pig nos sirve de forma directa para conseguir:

- Hace los análisis de Big Data accesible a los NO programadores que vean codificar un programa MR como un obstáculo
- Apache Pig es una plataforma para analizar datos, consistente en un lenguaje de alto nivel y un intérprete de ese lenguaje
- Una de sus características principales es que la estructura de las consultas permite un alto nivel de paralelismo, mejorando el rendimiento
- Características:
    - Extensible: los usuarios pueden crear funciones a medida

- Sencillo de programar: Las tareas complejas que implican transformaciones de datos relacionados entre sí pueden ser simplificadas y codificadas como secuencias de flujo de datos
- Auto Optimizable: Automáticamente optimiza la ejecución de los trabajos de Pig

❖ Casos de Uso:
- ETL
- Procesamiento iterativo de datos
- Búsquedas sobre datos en bruto

Grunt shell

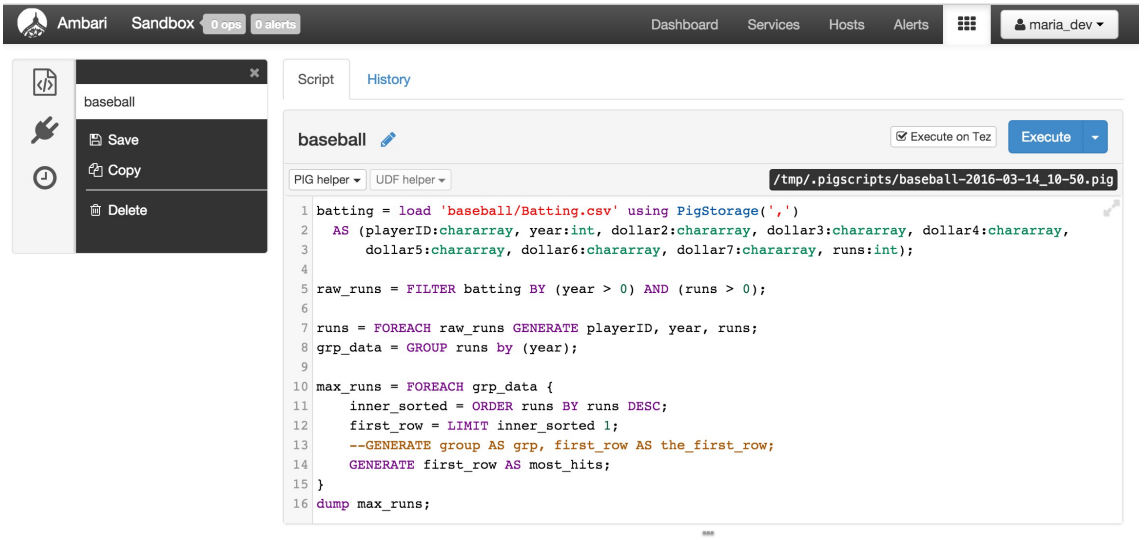
Grunt shell

```
grunt> employees = LOAD 'pigdemo.txt' AS (state, name);
grunt> describe employees;
employees: {state: bytearray,name: bytearray}
grunt> employees_grp = group employees by state;
grunt> dump employees;
```

Grunt es una consola interactiva que nos permite a los usuarios introducir comandos de Pig Latin e interactuar con los ficheros de HDFS.

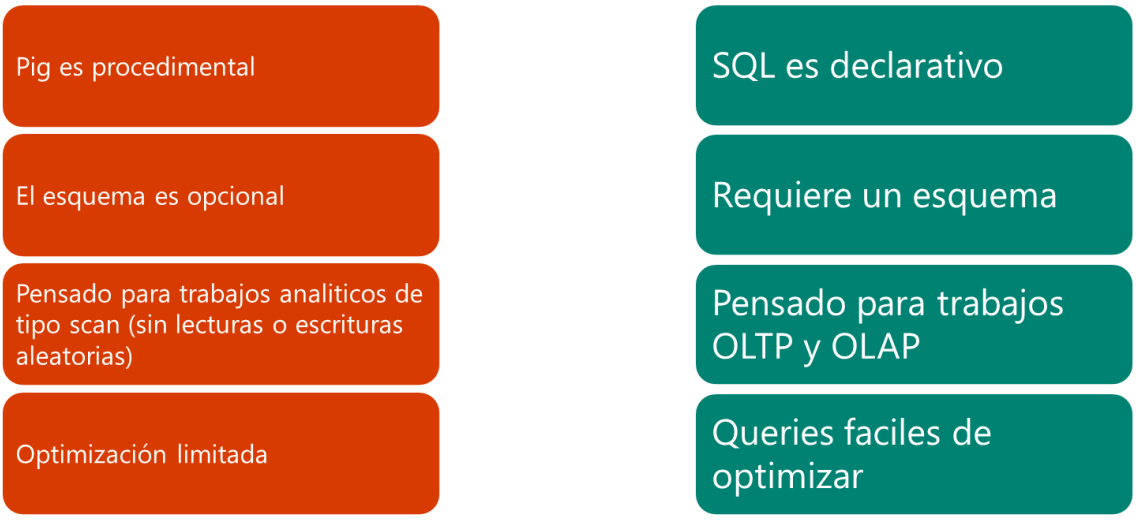
Ambari

Es una consola web para poder ejecutar los scripts de Pig, muchas veces es donde nos vamos a mover la mayor parte del tiempo dado que viene integrada por defecto en algunas de las distribuciones de hadoop.


Ambari
Sandbox
0 ops
0 alerts
Dashboard
Services
Hosts
Alerts
maria_dev
baseball
Save
Copy
Delete
Script
History
baseball
Execute on Tez
Execute
PIG helper
UDF helper
/tmp/.pigscripts/baseball-2016-03-14_10-50.pig
1 batting = load 'baseball/Batting.csv' using PigStorage(',')
2 AS (playerID:chararray, year:int, dollar2:chararray, dollar3:chararray, dollar4:chararray,
3 dollar5:chararray, dollar6:chararray, dollar7:chararray, runs:int);
4
5 raw_runs = FILTER batting BY (year > 0) AND (runs > 0);
6
7 runs = FOREACH raw_runs GENERATE playerID, year, runs;
8 grp_data = GROUP runs by (year);
9
10 max_runs = FOREACH grp_data {
11 inner_sorted = ORDER runs BY runs DESC;
12 first_row = LIMIT inner_sorted 1;
13 --GENERATE group AS grp, first_row AS the_first_row;
14 GENERATE first_row AS most_hits;
15 }
16 dump max_runs;

Pig versus SQL

Es sencillo ver las diferencias principales entre, PIG y un lenguaje tipo SQL, dado que como estamos aprendiendo, Pig no necesita esquemas para funcionar con los datos, y está orientado a realizar trabajos que no tengan tareas diversas, sino más simples como el escaneo de los datos.



Pig está formado por una serie de comandos básicos:

Muchos de ellos son sencillos para realizar la carga de datos, como:

LOAD

ventas = LOAD 'misventas.txt'
USING TextLoader();

LIMIT

empleados = LOAD 'pigdemo.txt' AS (provincia:chararray, nombre:chararray);

emp_group = GROUP empleados BY provincia;

L = LIMIT emp_group 3;

DUMP y STORE

Sin embargo, hay dos de ellos que lanza un plan de ejecución a nivel lógico al comenzar que son los comando DUMP y STORE, esto los diferencia de un simple ForEach o similares

Load	Read data from the file system
Store	Write data to the file system
Dump	Write output to stdout
Foreach	Apply expressions to each record and generate one or more record
Filter	Apply predicates to each record and remove record where false
Group Cogroup	Collect records with the same key from one or more inputs
Join	Join two or more inputs based on a key
Order	Sort records based on a key
Distinct	Remove duplicate records
Union	Merge two datasets
Limit	Limit the number of records
Split	Split data into 2 or more sets, based on filter conditions

Pig Latin

Vamos a pasar a analizar algunos ejemplos de programas Pig para que os familiaricéis con el lenguaje de Pig Latin:

Ejemplo script de Pig

En este ejemplo cargamos usuarios, clics de los usuarios en una dirección web y su localización geográfica por ip, para almacenar el número de clics agrupados

```
Users = load 'users' as (name, age, ipaddr);
Clicks = load 'clicks' as (user, url, value);
ValuableClicks = filter Clicks by value > 0;
UserClicks = join Users by name, ValuableClicks by user;
Geoinfo = load 'geoinfo' as (ipaddr, dma);
UserGeo = join UserClicks by ipaddr, Geoinfo by ipaddr; ByDMA = group UserGeo by dma;
ValuableClicksPerDMA = foreach ByDMA
   generate group, COUNT(UserGeo);
store ValuableClicksPerDMA into 'ValuableClicksPerDMA';
```

Ejemplo script de Pig

En este ejemplo cargamos lo datos de sueldo y luego filtramos por las personas que ganen más de 100.000.

```
sueldos = LOAD 'sueldos.data'
        USING PigStorage(',')
                        AS (genero, edad, ingresos, cp);
sueldazos = FILTER sueldos BY ingresos > 100000;
```

Ejemplo script de Pig

En este ejemplo cargamos

```
empleados = LOAD 'pigdemo.txt' AS (provincia:chararray, nombre:chararray);
grupo_ empleados = GROUP empleados BY provincia;
L = LIMIT grupo_ empleados 3;
```

Spark.

¿Qué es Spark?

Spark es una solución open-source para realizar cálculos rápidos sobre datos en memoria.

Spark fue creado para ser un motor de procesamiento de datos de propósito general, centrado en casos de uso alrededor del análisis en memoria, por lo tanto, Spark tomó muchos conceptos de MapReduce y los ha implementado de una manera mejorada.

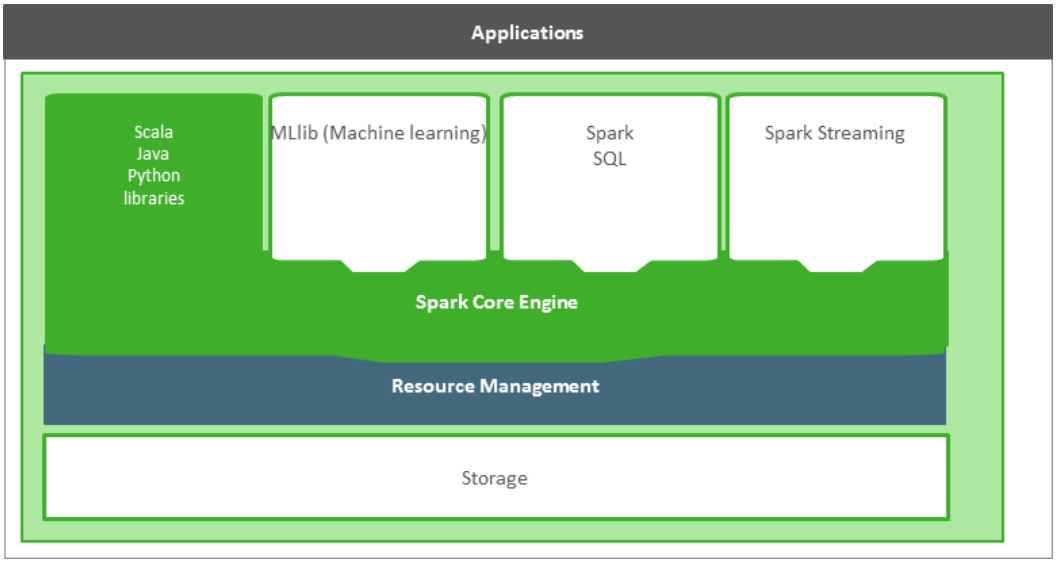
Dicho esto, Spark permite acceso a una gran variedad de tipos de usuario, para hacer cada uno de ellos tareas distintas sobre los datos.

- Mllib fue integrado para los científicos de los datos.
- Spark SQL fue creado para permitir a analistas sql tradicional una interfaz conocida para interactuar con grandes cantidades de datos.
- Spark streaming fue también introducido para cubrir los escenarios de micro batches.
- Spark permite a los usuarios cambiar entre estos frameworks con diferencias mínimas en la programación para aumentar la flexibilidad de las aplicaciones que los usuarios pueden desarrollar.
- Spark está totalmente integrado con Hadoop.
- Spark ahora mismo, soporta lenguajes como: Scala, Python, R, Java

En nuestro caso vamos a repasar la arquitectura y el módulo de Spark SQL dado que el master está centrado alrededor de los datos.

Arquitectura de Spark

La arquitectura de Spark está formada en su core, por los módulos reflejados en la siguiente imagen:

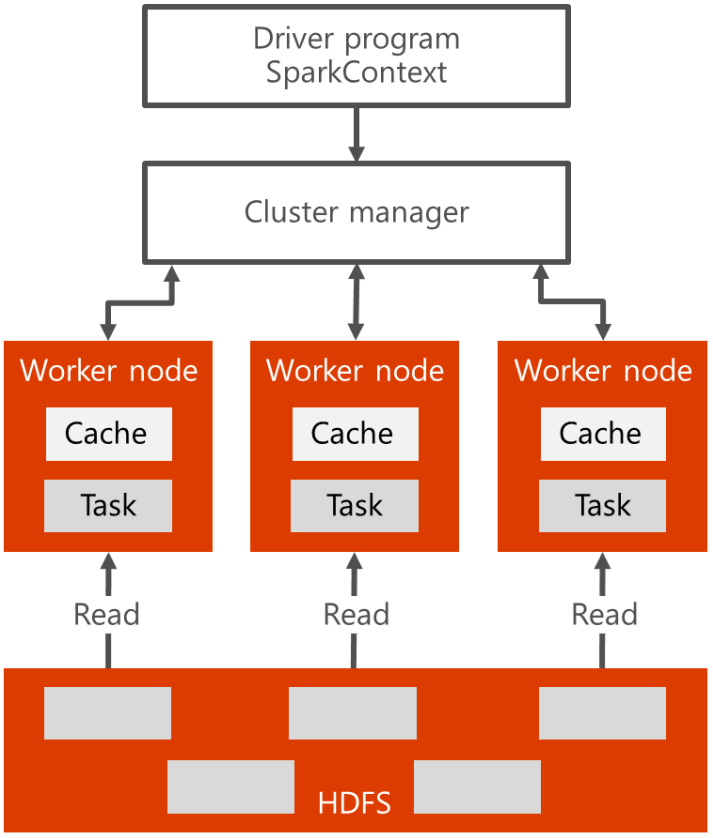


Además, Spark está dividido en 3 capas principalmente,

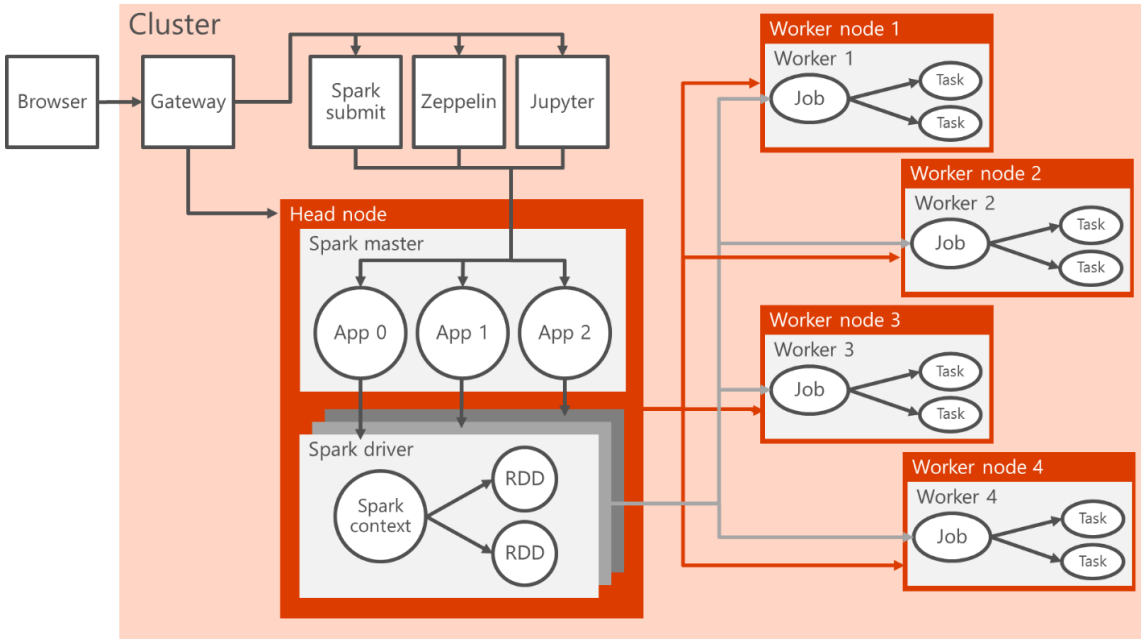
1. La capa de aplicación que es la que el desarrollador usa en función de sus requisitos
2. El core de Spark que se encarga de las tareas de ejecución de Jobs (tareas)
3. La capa de infraestructura, que depende de cómo ha sido aprovisionado el cluster

La arquitectura de un cluster de Spark funciona del modo siguiente:

En el nodo principal, tenemos el maestro de Spark que gestiona las aplicaciones que se asignan al driver de Spark, después cada aplicación es manejada por el Spark master, y Spark asigna al worker node los trabajos y asigna los recursos necesarios para la aplicación. Los recursos del clúster son administrados por el Spark master de Hadoop. Esto significa que el Spark master tiene conocimiento de qué recursos y que memoria está ocupada o disponible en cada Worker Node.



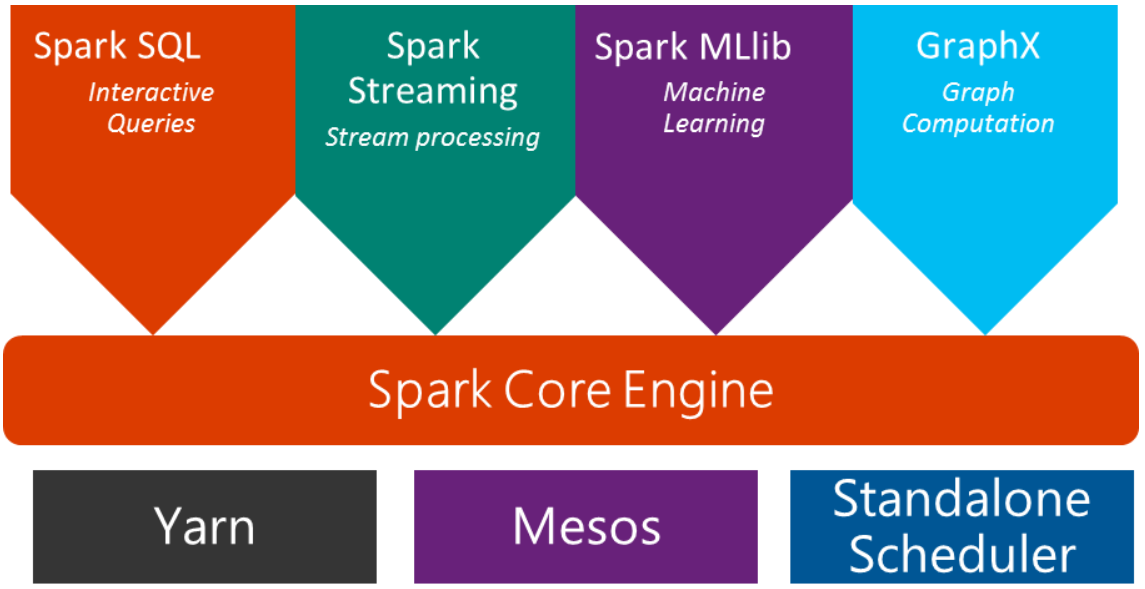
Una arquitectura básica de Spark, se corresponde con la siguiente imagen:



Spark & Hadoop

La estructuración en estas tres capas, nos habilitan que Spark cumpla las siguientes capacidades respecto a la tecnología de Hadoop:

- Spark puede utilizar Hadoop 1.0 o Hadoop YARN.
- Spark puede también utilizarse con otros gestores de recursos como MESOS
- Spark no tiene su propia capa de almacenamiento, utiliza directamente HDFS.
- Spark está integrado con Apache Hive, Apache HBase.



Spark proporciona en su convivencia con Hadoop ventas desde el punto de vista de 4 áreas:

Rendimiento:

- Dado que es computación en memoria, Spark es mucho más rápido que Hadoop.
- Spark puede ser utilizado en proyectos con procesos batch y tiempo real.

Productividad para los desarrolladores:

- Spark ofrece interfaces de programación (APIs) sencillas de utilizar para el procesamiento de grandes conjuntos de datos.
- Spark incluye más de 100 operadores para la transformación de datos.

Motor unificado:

Incluye un framework de alto nivel para realizar consultas SQL, streaming, machine Learning y procesamiento gráfico.

Spark ofrece una única aplicación que puede combinar todos los tipos de procesamiento.

Ecosistema:

- Spark tiene soporte incorporado para muchas fuentes de datos como HDFS, RDBMS,Google Cloud Storage, S3, Apache Hive, Cassandra, MongoDB.

- Spark se ejecuta sobre Apache YARN resource manager.

Spark SQL y RDD

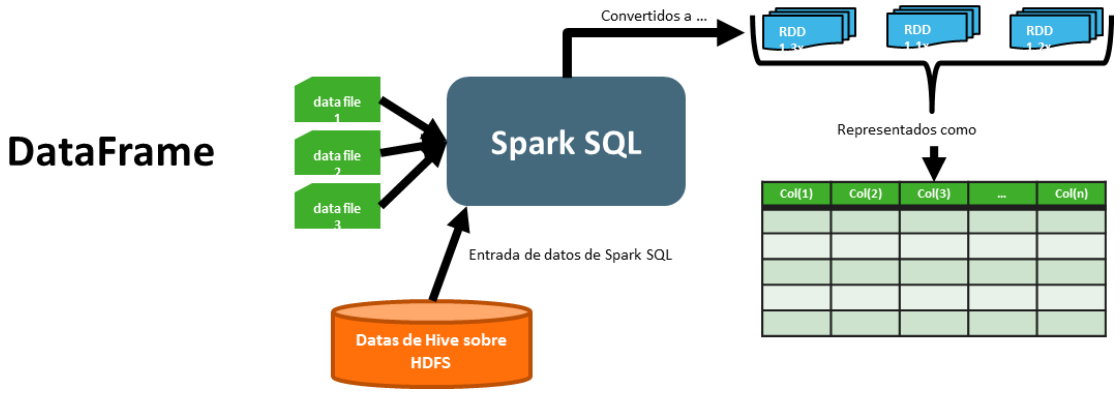
El core de las APIs de Spark funciona por pares clave/valor para manipular los datos.
Pero además del core de Spark, tenemos unos framework desarrollado sobre esa API.
Uno de los más importantes es el módulo de SparkSQL.
El módulo de SparkSQL permite a los desarrolladores mezclar consultas SQL dentro de sus aplicaciones de Spark.

Spark como vemos nos proporciona:

- Soporte de SQL y HiveQL como lenguajes de consulta
- Utilizar lenguajes como Python, Scala y Java
- Consultar datos almacenados en fuentes externas como, con ficheros estructurados (ejemplo: JSON) y tablas de Hive

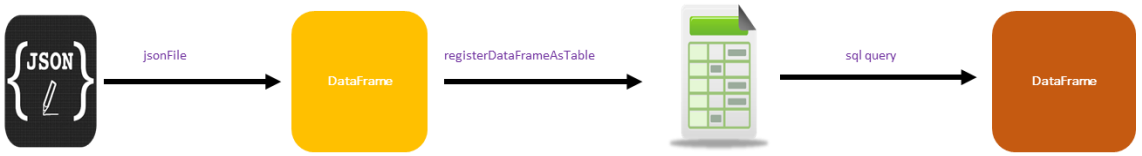
Spark está construido alrededor del concepto de RDD.
RDD significa Resilient Distributed Dataset.
Resistente en el sentido de que datos pueden crearse sobre la marcha en caso de fallos.
Distribuidos en el sentido de que operaciones pueden ocurrir en paralelo en varias máquinas trabajando en bloques de datos que permitan a Spark escalar muy fácilmente.

Como mencionamos antes, Spark SQL, es un módulo que está construido sobre el núcleo de Spark. Spark SQL proporciona otro nivel de abstracción para programación declarativa encima de Spark. En Spark SQL, los datos se describen como un DataFrame con filas, columnas y un esquema.
La manipulación de datos en Spark SQL está disponible a través de consultas SQL y las API de DataFrames.
Spark SQL se utiliza más a nivel de empresa dado que Spark SQL permite al desarrollador centrarse aún más en la lógica de negocio y mucho menos las optimizaciones, etc…



Un Dataframe se puede registrar como una tabla, la cual puede ser utilizada en las consultas tipo SQL

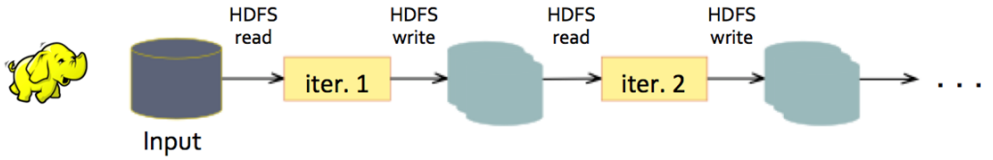
```
// first create  a DataFrame from JSON file
>>> val df = sqlContext.jsonFile(”Users.json“)

// register the Dataframe as a temporary table. Temp tables exist only during the lifetime of this instance of SQLContext
 >>> val usertable = sqlContext.registerDataFrameAsTable(df, “UserTable”)

//execute a SQL query on the table. The query returns a DataFrame.
// This is another way to create DataFrames
>>> val teenagers = sqlContext.sql(“select  Age as Years from UserTable where age > 13 and age <= 19”)
```



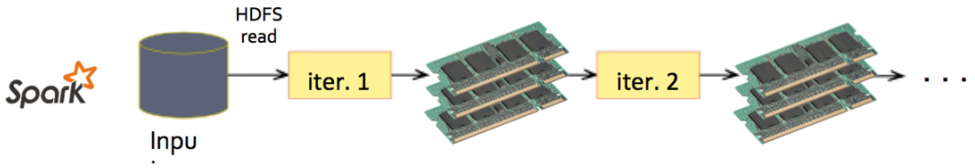
Resumen Spark

La motivación inicial de Spark fue que las aplicaciones iterativas no funcionan bien con MapReduce.
Las Aplicaciones iterativas en MapReduce requieren escribir datos intermedios en HDFS y luego volver a leer para cada iteración.
El objetivo de Spark era mantener más datos en memoria, para reducir al mínimo el leer/escribir de disco. Dado que leer de memoria se mide en nanosegundos, mientras que la lectura de disco se mide en milisegundos.

- **MapReduce** – implica *mucho acceso a disco, lo cual ralentiza todo*



- **Spark** – Mantiene muchos de los datos en memoria



Spark es más rápido que MapReduce por varias razones. Primero y principal, Spark puede cachear datos en memoria. Además de los beneficios de leer datos de memoria frente a leerlos de disco, la programación de tareas en Spark ha disminuido considerablemente frente a MapReduce.
Debido a esto, la programación ha pasado si tardaba 15-20 segundos a tardar 15-20ms.
En Spark, puedes tener múltiples map/reduce en una fila.
No necesita una fase de map para cada fase de reduce esto ahorra leer y escribir datos en disco.