



El libro verde del BigData

Procesamiento de datos con Open Source

stratebi
open business intelligence

CONTENIDO (SEPTIEMBRE 2014)

| | |
|---|-----------|
| Big Data... ¿y esto qué es? | 4 |
| Los problemas actuales | 5 |
| Las soluciones y Bases de datos propuestas | 7 |
| Bases de datos Big Data | 7 |
| Sistema de archivos distribuido para garantizar escalabilidad | 7 |
| Big Data... ¿y esto qué es?: Conclusión mundo BI | 8 |
| Creando flujos de datos BigData con Kettle | 11 |
| Imaginando una situación que se puede dar | 11 |
| Flujos de datos BigData: Hadoop, Hive y Kettle | 12 |
| Alcance | 13 |
| Trabajando con Kettle | 14 |
| Conclusiones..... | 18 |
| Bases de datos para proyectos BigData | 20 |
| ¿Por qué NoSQL? | 20 |
| BD Key-Value. | 20 |
| BD orientados a documentos | 21 |
| BD orientadas a grafos. | 21 |
| BD orientadas a objetos. | 21 |
| Not only noSQL: BD Columnares. | 21 |
| Introducción a MongoDB | 22 |
| Desde el punto de vista de la analítica de datos | 24 |
| Trabajando con Kettle y MongoDb | 25 |
| Alcance | 25 |
| Trabajando | 25 |
| ETL para insertar datos en MongoDB..... | 26 |
| ETL para extraer datos de MongoDB..... | 28 |
| Amazon S3 y Analítica de datos..... | 30 |
| Cómo funciona S3..... | 30 |
| Casos de analítica de datos | 31 |

| | |
|---|-----------|
| Introducción a Amazon Elastic MapReduce | 32 |
| Amazon Elastic Map Reduce | 32 |
| Cómo Funciona..... | 32 |
| Conclusión | 34 |
| Kettle BigData: Amazon EMR y S3 | 35 |
| Alcance | 35 |
| Trabajando con Amazon S3 | 35 |
| Trabajando con Amazon EMR | 36 |
| Conclusión | 37 |
| 9.000.000.000 de elementos en una tabla de hechos..... | 39 |
| Introducción | 39 |
| Arquitectura..... | 39 |
| Resultado..... | 40 |
| Sobre Stratebi..... | 42 |
| Más información..... | 43 |

BIG DATA... ¿Y ESTO QUÉ ES?

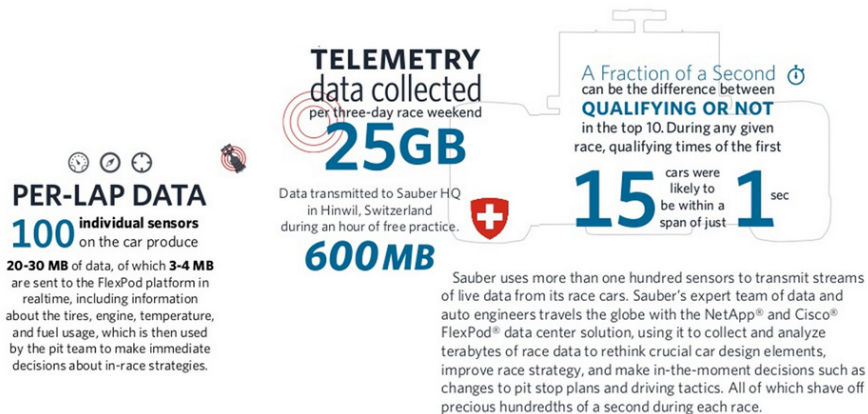
Hoy en día hay mucha confusión y demasiado populismo con este nuevo término llamado BIG DATA. A cualquier proyecto de análisis de datos se le está poniendo la etiqueta de **BigData simplemente por que se tratan muchos datos**. Entre los consultores BI está siempre la conversación y la gran mayoría, por no decir todos, creen que se está sustituyendo BI por BigData.

Surgen muchos *Masters*, cursos y charlas que en su contenido el 70%-80% es teoría de BI y el otro 20% es cómo usar tecnologías Big Data... Desde este documento, se busca dar un punto de vista de qué es BigData y cómo se asocia al Business Intelligence de forma natural.



Sauber F1 Team

Análisis de datos para mejorar el rendimiento

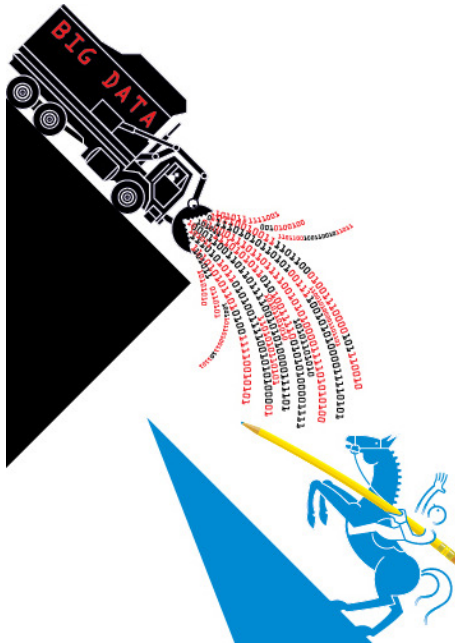


Desde antaño, **las empresas vienen sufriendo transformaciones debido a la tecnología**. Los ejecutivos empezaron a gestionar sus empresas sin guardar datos pues no existía ningún medio para ello, después surgió la tecnología y comenzaron a usar BD con pocos datos que se fueron transformando y creciendo hasta un punto de tener que surgir nuevas formas de "análisis" y "digestión" de esa información. Es ahí que surgieron las tecnologías y procesos de Business Intelligence que buscaban analizar esos datos que no podían analizar a simple vista. Pero ahora, **con la evolución de la tecnología han surgido nuevos tipos de datos que no hay como tratar con las tecnologías de siempre y también se generan millones de datos en muy poco tiempo que no se pueden almacenar pero sí se quiere analizar**. Es por ello que surgen estas tecnologías y procesos BigData que buscan proveer a aplicaciones

empresariales de las carencias que las actuales no consiguen proveer. Veamos un poco los retos y qué propuestas hay actualmente en el ecosistema BigData y BI.

BIG DATA... ¿Y ESTO QUÉ ES?

Actualmente las empresas están viendo cómo el mundo de la tecnología está creciendo y transformándose. **Surgen nuevos tipos de datos y necesidades que actualmente los sistemas no son suficientemente buenos o adecuados para poder atacar estos problemas** pues las empresas son más exigentes y buscan exprimir al máximo sus recursos para obtener el mayor beneficio. Sería semejante a escuderías de F1 que buscan superar al rival buscando la diferencia hasta en los grados de regulación de un alerón, analizando y optimizando al mayor detalle. A continuación, se expondrán algunos de los muchos problemas, pero estos que se destacan quizás son unos de los principales motivos que ha hecho que surja todo ese ecosistema de procesos y herramientas "BigData".



LOS PROBLEMAS ACTUALES

La teoría que nos enseñan en la carrera de informática es que el modelo tradicional de BD es el relacional que con ello podemos hacer todo. Hasta hace relativamente poco, incluso los hay que aún solventan cualquier problema con relacionales. Actualmente hay una serie de problemáticas con este tipo de BD que se resumen en estos 3 puntos:

Tipos de datos. Variedad. Han surgido nuevos tipos de datos que se quieren almacenar: datos no estructurados. Las BD Relacionales no pueden almacenar este tipo de datos.

Escalabilidad. En búsqueda de la rapidez y rendimiento en consultas o procesamiento de datos se busca escalar siempre en horizontal. Es decir, si necesitamos más

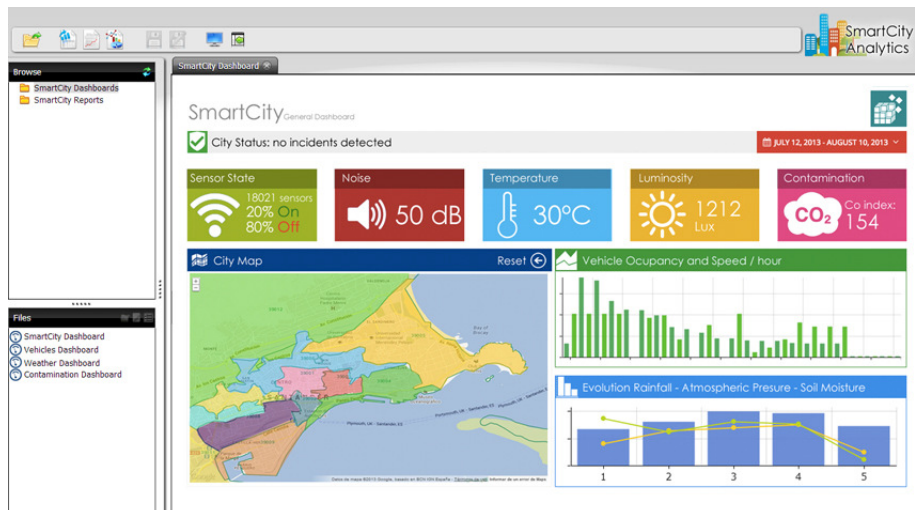
rendimiento añadimos una CPU a nuestro conjunto de trabajo para poder aumentar nuestras prestaciones en conjunto y aumentar el rendimiento reduciendo el tiempo de búsqueda o almacenamiento. El problema es que actualmente las BDRelacionales no pueden estar distribuidas en nodos diferentes de forma sencilla transparente al usuario. Por ello la única forma de conseguir estos dos objetivos en las BD Relacionales es añadiendo CPU y Memoria, haciendo escalabilidad vertical. Pero esto no es suficiente, buscamos escalabilidad

horizontal para tener todos los servidores que queramos trabajando en paralelo y no tener un límite como es el caso del escalado vertical.

Modelo relacional. El modelo relacional no da soporte para todos los problemas. No podemos atacar todos los problemas con el mismo enfoque, queremos optimizar al 100% nuestro sistema y no podemos ajustar nuestros sistemas a estas BD. Por ejemplo, en el modelo relacional no podemos tener herencia de objetos o no podemos tener columnas variables según las filas...

Velocidad. Esta es una de las "3 V's" del Big Data (velocidad, variedad, volumetría). La velocidad de generación de datos hoy en día es muy elevada, simplemente hay que verlo con las redes sociales actuales, aunque las empresas medias y muchas de las grandes no se ven afectadas por ello. Donde sí influye la velocidad es en el procesamiento de todo este conjunto ingente de datos, pues cuantos más datos tengamos más tiempo requieren. Por ello, se necesita un ecosistema que sea capaz de escalar en horizontal para trabajar en paralelo y ahorrar tiempo, siguiendo la técnica del "divide y vencerás".

Por ello, teniendo en cuenta estos principales problemas, se han creado nuevas herramientas y sistemas para poder tener alternativas, que no sustituyen a las actuales, sino que aportan una forma alternativa de atacar problemas y/o mejorar nuestra solución de procesamiento y análisis de datos.



Ejemplo de procesamiento de datos de sensores en Smart Cities

BIG DATA... ¿Y ESTO QUÉ ES?: SOLUCIONES A RETOS

Actualmente las empresas se ven con nuevos problemas que anteriormente no tenían. Nuevos tipos de datos, nuevas exigencias de rendimiento a la hora de procesamiento, nuevos enfoques de soluciones BI... por ello, **han surgido diferentes propuestas para atacar problemas concretos comentados en el post anterior**. No sustituyen a las BD Relacionales ni a los procesos de creación de proyectos BI, sino que son "**herramientas**" con las que **podemos trabajar para resolver con mayor eficacia esos problemas**.

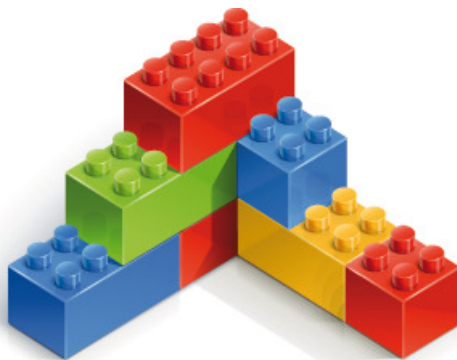
LAS SOLUCIONES Y BASES DE DATOS PROPUESTAS

BASES DE DATOS BIG DATA

Son nuevos datos y nuevas formas de almacenar estos datos que surgen en la actualidad. Por ello, surgen nuevos gestores como los NoSQL que buscan dar solución a los 3 problemas anteriores (escalabilidad, heterogeneidad y rendimiento). Por ejemplo, las BD clave-valor consiguen almacenar de forma sencilla diferentes tipos de datos no estructurados, sería un símil a tener una fila con 2 columnas y después una fila con columnas dinámicas. Esto en las relacionales es impensable. En un punto posterior se analizarán estas bases de datos propuestas.

SISTEMA DE ARCHIVOS DISTRIBUIDO PARA GARANTIZAR ESCALABILIDAD

El problema de la reducción de datos ha sido uno de los grandes problemas de Google al inicio de su actividad. Necesitaban procesar millones de millones de páginas para poder obtener el resultado de su PageRank de forma resumida, por lo tanto crearon un algoritmo el cual conseguía resumir de forma sencilla todos esos datos. **Crearon el corazón del BigData, el concepto del algoritmo MapReduce**. Después **llegó Yahoo y creó la cáscara que envuelve el MapReduce, comúnmente llamado Hadoop**. La gran base que garantiza ejecutar programas MapReduce hechos por usuarios en nodos distribuidos. Esta herramienta tiene un sistema de archivos HDFS el cual provee la distribución de trabajos a diferentes nodos que ejecutarán en



paralelo este algoritmo de reducción. Una de las cosas más importantes es que el HDFS consigue hacer transparente o simplificar la creación de clusters de nodos que trabajan en paralelo como un nodo solo. Si necesitas más potencia es simplemente añadir una nueva IP de un nodo en un fichero. Fácil y sencillo.

Este sistema HDFS es la base del BigData pues están surgiendo y se desarrollan herramientas que son ejecutadas encima de este sistema HDFS pudiendo obtener aplicaciones distribuidas. Las más importantes son las BD como **HBASE** o **HIVE** (que realmente no es una BD sino un intérprete SQL que traduce MapReduce). Son dos herramientas que se ejecutan encima del HDFS de forma sencilla pero que actúan sobre un cluster de N nodos con transparencia. Si necesitamos más potencia o espacio añadimos un nuevo nodo como anteriormente. **Las consultas irán más rápido conforme añadamos nodos a nuestro cluster. La famosa teoría del DIVIDE Y VENCERÁS**



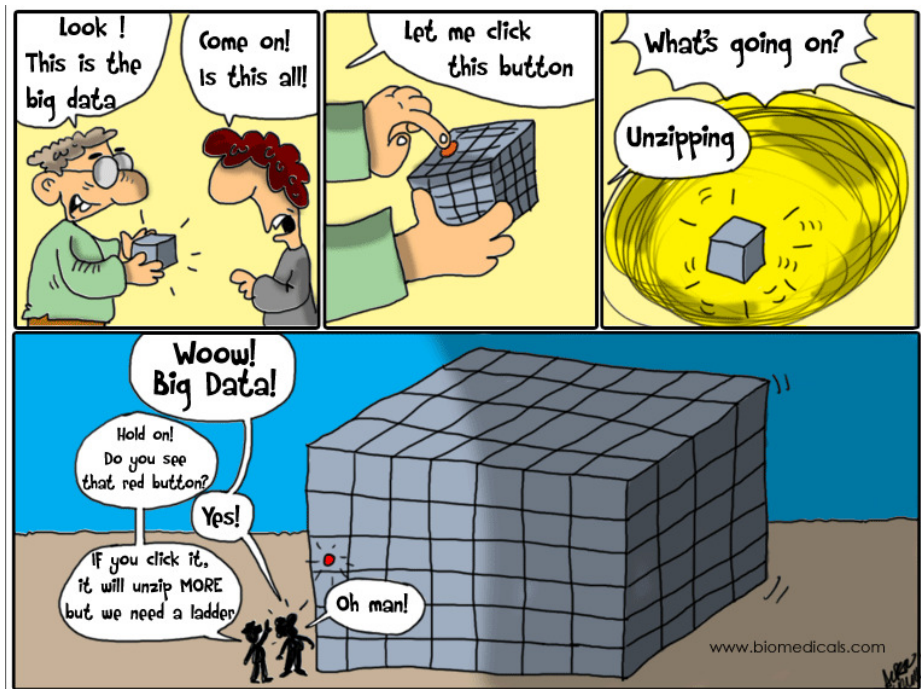
BIG DATA... ¿Y ESTO QUÉ ES?: CONCLUSIÓN MUNDO BI

Se puede ver que se disponen de nuevos recursos para poder atacar a toda esta ingente cantidad de datos y a los diferentes problemas vistos. **Pudiendo elegir nuevos tipos de BD que se ajusten a nuestro problema o configurando el sistema base que permite crear trabajar con transparencia los datos sobre un cluster** y tener procesamiento y almacenamiento de datos en paralelo de forma sencilla. Esto no sustituye a las BD Relacionales, sino que son herramientas que sirven para poder optimizar procesos. **En el campo del BI lo podemos situar en 3 partes: como fuente de datos, como Stage de nuestro sistema o como DW final al cual atacaremos**

1) Fuente de datos. Que salgan conectores a BD NoSQL desde herramientas ETLs o BI es vital en muchos proyectos BI pues ahora se empiezan a observar en empresas que prescinden de las BD Relacionales en sus sistemas y solamente tienen estos nuevos tipos de BD. Por ejemplo, **hay muchas empresas que están usando MongoDB para guardar datos no estructurados**, por ello, uno de los grandes retos para un consultor BI es poder obtener esa información.

2) Stage. El ecosistema Hadoop consiste en crear una base para poder escalar aplicaciones o aplicar MapReduce a nuestras fuentes de datos. Estas aplicaciones pueden ser BD las cuales funcionen de forma distribuida en el sistema de archivos HDFS. Por ello, para los consultores es una opción el tener un **stage donde se vuelquen todos los datos históricos y después se hagan agregaciones (MapReduce)** de forma distribuida en nodos y obteniendo un resultado final el cual, como opción, podemos cruzar con otra fuente de datos estructurada y dejando el resultado final en un DW.

3) Datawarehouse La finalidad de todos los proyectos BI es poder analizar los datos. Además de las anteriores opciones, podremos tener un ecosistema Hadoop el cual sirva de soporte para una BD distribuida que funcione como **almacén de datos analítico al cual atacaremos desde nuestra herramienta BI con cubos, informes o dashboards**. Con esto, se ganaría tener todos los datos de forma distribuida y que a la hora de realizar consultas se pueda tener mayor velocidad, si necesitamos más, se añaden más nodos que queramos al cluster.



Todos los enfoques son válidos, podemos tener un informe que se genere directamente atacando a un MongoDB, podríamos tener informes que ataquen a un DW en Hadoop o informes que ataquen a una estrella en una BD Columnar que recogiese datos que provienen de un ecosistema Hadoop y cruzando esta información con *smalldata*.

CREANDO FLUJOS DE DATOS BIGDATA CON KETTLE

Kettle es una herramienta de integración de datos Open Source que antiguamente estaba como herramienta independiente de procesos de ETLs hasta que comenzó a formar parte de la suite BI Open Source "Pentaho".



En los últimos años, han surgido nuevas necesidades en el procesamiento de datos masivo, y es por ello que ha surgido toda la corriente "Big Data. Los de Pentaho se dieron cuenta de que había un gran problema a la hora de tratar con todas las herramientas BigData como Hadoop, Hive, Hbase, MongoDB o Cassandra pues había que tener conocimientos muy técnicos para ello. Además, antiguamente estas herramientas al ser tan nuevas no había forma de integrarlas en procesos ETLs. Es por ello que Pentaho decidió apostar y aprovechar que la herramienta Kettle (PDI) es una herramienta Java y modular e desarrollar e incorporar los componentes "Big Data" para operar con estas herramientas.

Añadiendo estos componentes y conectores, el usuario técnico o no técnico sin conocimientos (porque **no podemos conocer todos los lenguajes de programación**) conseguirá crear flujos de datos los cuales consistan en extraer de algún origen de Big Data como puede ser MongoDB ,Hadoop (o cualquier herramienta BigData soportada por Kettle), operar con estos datos y después dejarlos en algún lugar, ya sea un informe, mandar un email con los datos o depositarlos en otra BD relacional o NoSQL como Cassandra.

IMAGINANDO UNA SITUACIÓN QUE SE PUEDE DAR

Un cliente tiene una BD como Cassandra. Imaginemos que este cliente ha desplegado 1000 sensores por toda una ciudad y necesita inserts y updates rápidos. Además, busca escalabilidad en una BD, pues mañana puede tener que desplegar 100.000 sensores, y esta BD podría escalar en número de nodos y así mantener su rendimiento.

Actualmente, este usuario tiene otras fuentes de datos como BD relacionales (un SQL Server), un montón de Excels que se generan a mano sus empleados, y algún fichero en txt con un formato standard autogenerado por los servidores (logs)

El problema viene a la hora de poder analizar los datos, el cliente tiene un montón de orígenes de datos que actualmente para su equipo técnico es un calvario pues tienen diferentes orígenes de datos y para extraer la información es necesario saber diferentes formas de programación: cómo conectarse a Cassandra, SQL Server con un lenguaje de programación, cómo abrir y parsear un Excel o un log en txt...etc.

Es en este punto que Kettle ayudará mucho a un equipo debido a su sencillez en el desarrollo de flujos de datos. Kettle permite a través de pasos (dedicados a realizar acciones específicas) extraer la información. Por ejemplo:

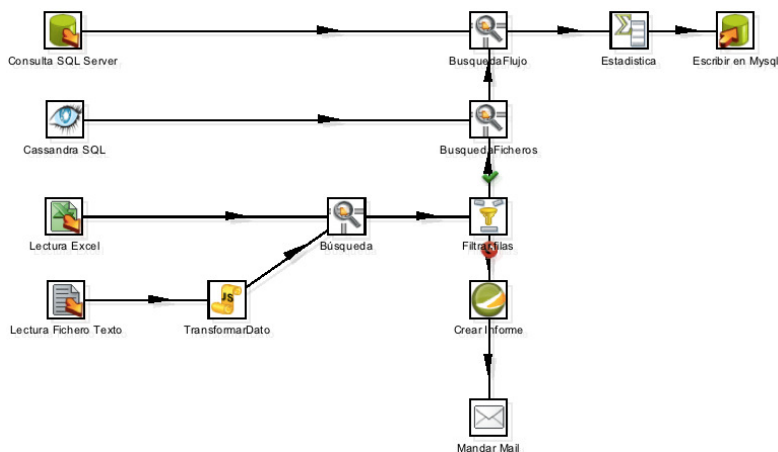
Se podría usar un paso para extraer de cada fuente de datos y configurar unos parámetros como la dirección del servidor, el nombre de BD y el puerto (eso para Cassandra y SQL Server).

También existen pasos para extraer datos de Excel y TXT, de forma sencilla y configurable con una interfaz gráfica. Y se podría crear automáticamente un informe y mandarlo por email configurando 2 pasos.

Con Kettle, los usuarios de negocio podrían hacer un flujo de datos que extraiga los datos de todas esas fuentes y los mezcle para crear automáticamente un informe con los indicadores deseados.

Ejemplo:

A continuación se muestra una ETL compuesta por 13 *steps* los cuales 4 de ellos extraen la información, los del medio los "mezclan" y por último hay 2 salidas, una de ellas es mandar un email con un informe creado en un paso previo y otro es escribir en una BD la información obtenida de SQL Server, Cassandra y Excel. ¿Sencillo verdad?



FLUJOS DE DATOS BIGDATA: HADOOP, HIVE Y KETTLE

Al hilo del punto anterior introductorio de ETLs Big Data anterior, se podrá ver lo fácil que es trabajar con **Hadoop y Hive** con la herramienta **Kettle** pues se podrá evitar entrar a lenguajes muy técnicos sabiendo usar los pasitos que nos ofrece Kettle para trabajar.



Hadoop es un framework para desplegar aplicaciones de forma distribuida. Es decir,

que si necesitamos potencia para procesar archivos de forma masiva, con Hadoop se podrá crear un proceso Java (hace falta conocimientos técnicos avanzados) que opere con estos archivos de forma paralela en nodos distribuidos.

Por otra parte tenemos a Hive, **que es una herramienta que crea esos procesos MapReduce de forma automática a partir de HQL (parecido a SQL).** Hive ha surgido debido a la necesidad de los desarrolladores de consultar "flat tables" con lenguaje SQL. Por lo tanto, si se le manda una query en HQL (Hive query language) él creará un proceso MapReduce y dará un resultado. Se podrá hacer por ejemplo "Select avg(columna) from tabla where columna2="valor" sin tener que picar nada de Java, solamente HQL.

Como se puede entender, por regla general, necesitamos los siguientes conocimientos para trabajar con BigData:

Sistemas Linux para trabajar con Hadoop

Programación Java para crear los procesos

Lenguaje HQL para consultar a Hive

Esta es la forma típica. El tiempo de desarrollo podrá ser de horas dependiendo de lo que se busque hacer.

Por otra parte, tenemos **Kettle**, una herramienta OpenSource la cual **ayudará al consultor sin tener "ni idea" de todo lo anterior y podrá hacer procesos ETL de forma más sencilla y mucho más rápido.**

ALCANCE

El alcance de lo que vamos a hacer consiste en:

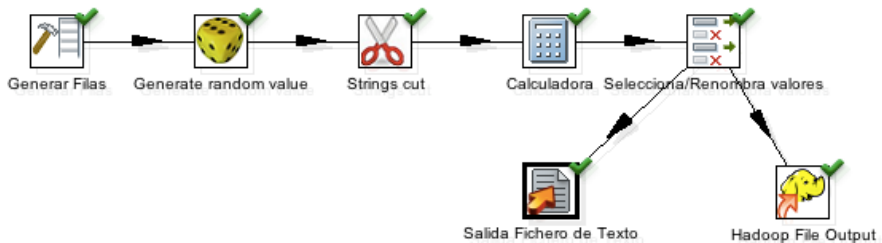
- generar un archivo con 2 columnas (columna de texto y columna de indicador que será un float)
- procesar este archivo con hadoop con muchos datos (en el ejemplo solamente 1000)

- atacar este archivo con Hive
- exportar datos de Hive a un archivo txt en un servidor

TRABAJANDO CON KETTLE

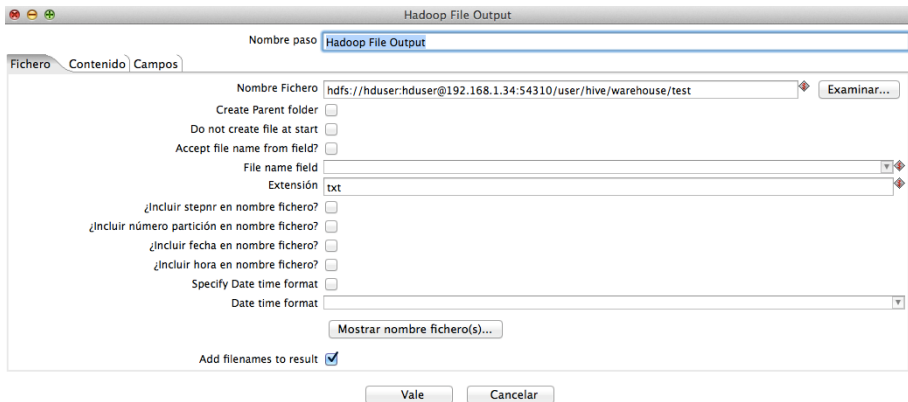
Lo primero es generar el archivo con 2 columnas y 1000 valores. Para ello, Kettle tiene una serie de pasos que nos ayudará a crear estos flujos de datos. Los primeros pasos (Generar Filas, Generate random value, Strings Cut, Caladora) permiten generar estos datos. Se generarán 1000 valores (filas).

Después lo que se hará en paralelo será crear un archivo en local (paso "Salida Fichero de Texto") y en paralelo se guardará en el sistema de archivos HDFS de Hadoop. Se disponen de 2 formas, o directamente a la tabla de Hive (para ello tendremos que crear una tabla y saber dónde está localizada en el HDFS) o en una carpeta en el HDFS y se mueve ese archivo.

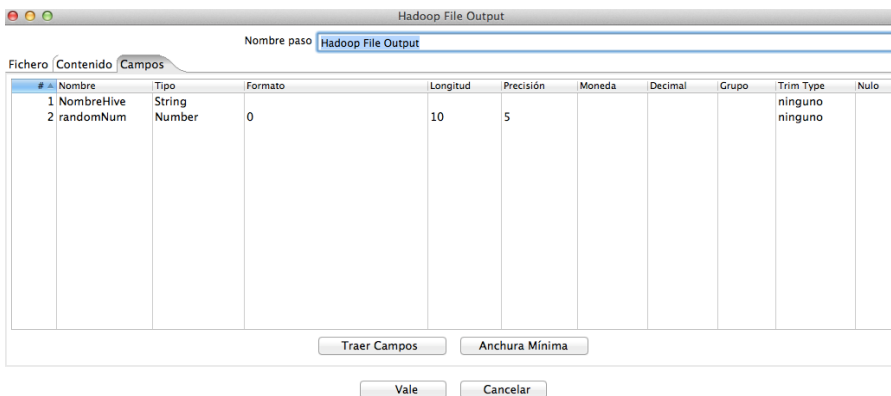


Ojo, en este caso se ha guardado el fichero de texto en local, pero se podría haber mandado por ejemplo ese fichero por Email a nuestro DBA o SysAdmin para que sepa que se ha cargado un fichero.

En el caso anterior, el paso "Hadoop File Output" se ha configurado de la forma más rápida para depositar los datos en "Hive". Para ello simplemente hay que decirle en qué carpeta de la estructura HDFS de Hadoop queremos dejarlo. En este caso en ***/hive/warehouse/test*** que es donde están los datos de la tabla "test" para atacarla con HQL



Otra de las configuraciones de este paso es decir qué tipo de datos vamos a insertar en el fichero en el HDFS. Para ello, nosotros podemos hacer **conversiones automáticas o establecer el formato de los campos** teniendo en cuenta el flujo de datos de entrada a ese paso. Muy simple y muy útil. **Facilita el no tener que estar usando funciones Java de conversión de objetos...** Nos fijamos en la pestaña "Campos" del paso "Hadoop File Output" donde he puesto que este archivo tenga 2 columnas.



En caso de que se haya dejado en otro lugar del HDFS y no directamente en las carpetas de Hive del HDFS, tienes otro paso que lo que hace es **"copiar archivos a Hadoop"**. Este paso es muy usado caso tengas ya un archivo generado en tu servidor y lo necesites mover al HDFS de Hadoop. O también puedes usarlo en caso que quieras mover archivos de una carpeta del HDFS a otra carpeta (como este ejemplo podría ser el warehouse de Hive)

Ojo, antes de ejecutar la ETL anterior, lo que se ha hecho ha sido crear una tabla por la consola de Hive con la siguiente estructura:

```
hive>
> create table test (
> NombreHive string,
> CodigoHive float)
> row format delimited
> fields terminated by '\t';
OK
Time taken: 0.06 seconds
```

Fijaros que he puesto que los campos están delimitados por '\t', y es que en el paso de "Hadoop File Output" lo he configurado así en la pestaña (que no se ve) de "Contenido".

Una vez ejecutada la ETL ya se podría hacer consultas como por ejemplo **"select * from test limit 10"**, pues se tendrá un fichero cargado en el repositorio de HDFS y concretamente en el warehouse de Hive.

```
OK
Codigo_2      0.1
Codigo_e      0.4
Codigo_5      0.9
Codigo_9      0.6
Codigo_0      1.0
Codigo_2      0.2
Codigo_9      0.1
Codigo_b      0.1
Codigo_7      0.8
Codigo_7      0.3
Time taken: 0.103 seconds
```

Si se hace algo más complejo a listar elementos, entra en funcionamiento la conversión de HQL a MapReduce. **Hive lo que hace es crear un programa MapReduce y ejecutarlo sobre Hadoop y su HDFS.** En este ejemplo, en vez de listar los valores de una tabla, lo que hacemos es la media de la 2a columna (el número decimal): **"Select avg(CodigoHive) from test"**. Se obtendrá la siguiente ejecución de MapReduce:


```

hive> select avg(CodigoHive) from test;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201311232007_0001, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201311232007_0001
Kill Command = /HDP20/hadoop/bin/../bin/hadoop job -Dmapred.job.tracker=localhost:54311 -kill job_201311232007_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2013-11-23 20:17:58,057 Stage-1 map = 0%, reduce = 0%
2013-11-23 20:18:01,097 Stage-1 map = 100%, reduce = 0%
2013-11-23 20:18:10,195 Stage-1 map = 100%, reduce = 100%
Ended Job = job_201311232007_0001
MapReduce Jobs Launched:
Job 0: Map: 1 Reduce: 1 HDFS Read: 12000 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
0.467
Time taken: 22.652 seconds

```

Se ha visto qué fácil es cargar datos desde Kettle y consultarlos desde la consola de usuario de Hive (\$CarpetaHive/bin/hive). Pero ¿qué pasaría si se necesita hacer otra ETL que opere con los datos de Hive? es decir... si se imagina que se quiere explorar los datos con un informe, se tendría que hacer un proceso Java para que se conecte a Hive y ejecutar consultas, pero **Kettle se podrá hacer una ETL como si fuera cualquier motor de BD y que se extraiga datos con consultas Hive. Parametrizando un paso de "Entrada tabla" que es muy común usar para consultar en un Mysql o Oracle...**

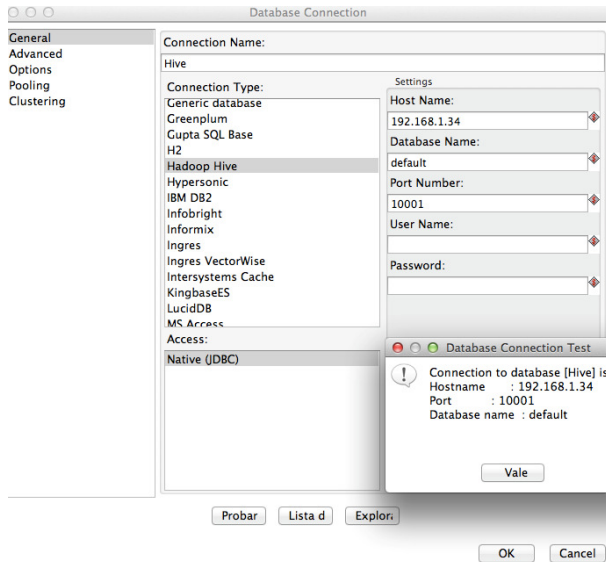
Para ello, primero se tendrá que poner Hive en modo servidor para crear una conexión desde Kettle:

```

hduser@ubuntu:/HDP20/hive/bin$ ./hive --service hiveserver -p 10001
Starting Hive Thrift Server

```

Una vez que se haga esto, se podrá configurar una conexión con Kettle de forma sencilla. Kettle ya tiene un driver para conectar a Hive, se configura HostName (IP), DatabaseName (default), PortNumber. Es la misma forma de configurar una conexión como si se fuera a atacar a una BD Relacional.



Y ahora se puede hacer una ETL (muy simple este ejemplo de abajo) que extraiga datos de una tabla Hive y lo vuelque a un archivo txt en un servidor. De forma sencilla sin tener que picar nada de código.

Este es un ejemplo muy simple, pero si se imagina que se dispone de una tabla de hechos de las ventas en modo "flat table" y se tiene que analizar las ventas por región, y dependiendo de la región y su estado mandar un email de aviso al gerente del país y al responsable de la región afectada, este proceso con Kettle es muy sencillo. No se tendrá que andar perdiendo mucho tiempo. Solamente habría que añadir un paso de "IF" y bifurcar el flujo y con el paso de "Mail" mandar ese email.



CONCLUSIONES

El tiempo dedicado a hacer estas 2 ETLs como mucho habrá sido de 20min, teniendo toda la infraestructura ya desplegada en un cluster o singlenode (Hadoop y Hive).

En Pentaho defienden que Pentaho Big Data aligera los procesos de desarrollo, y en ello se está de acuerdo, en este caso no se ha tardado más de 20min. Pero otra persona que lo haga sin Kettle y lo haga con lenguaje de programación, puede que le tome unas horas y si se le añade complejidad como mandar mails, generar reporting o volcarlo a otro tipo de BD columnar o NoSQL...pues quizás la persona se desespere al tener que aprender mil y una formas de procesar los datos y conectividad con tecnologías. En Kettle simplemente es saber usar y configurar los "steps" o pasos que ofrece Kettle.

BASES DE DATOS PARA PROYECTOS BIGDATA

Cada vez la tecnología evoluciona más rápido y las necesidades son diferentes. No solamente las necesidades sino también las soluciones desarrolladas. Es por ello donde se está viendo una **evolución en las Bases de Datos que han surgido en los últimos años**. Se pasa de dejar el modelo relacional de siempre creado hace décadas para ir a **nuevos gestores de datos**, llamados NoSQL por "Not Only SQL" pues en resumen, **no todo se puede representar/almacenar de a misma forma de siempre (relacional)**, sino que cada problema tiene una forma de ser enfocado y atacado por una misma BD. Estos tipos de BD son muy utilizados en proyectos BigData por que puede facilitar el almacenamiento o relacionamiento como son las NoSql o análisis de los datos como son las columnares que se repasan al final del post.

¿POR QUÉ NOSQL?

- 1) Leer datos es costoso.** Hacer múltiples Joins a la hora de analizar grandes volúmenes de datos penaliza tanto en rendimiento como en tiempo.
- 2) Integridad en los datos.** El modelo relacional permite tener integridad, pero no es necesario en la actualidad
- 3) Escalabilidad.** Las BD relacionales no escalan fácilmente. Si se hace en vertical será sencillo pues simplemente será añadir más RAM o CPU, pero si es en horizontal surgirán problemas pues no es posible tener relaciones entre tablas en diferentes servidores
- 4) No se puede representar todo con el modelo relacional.**

Como respuesta a estos problemas surgió NoSQL, que es un movimiento que propone alternativas a las Bases de datos Relacionales dependiendo de las necesidades. Los tipos son:

BD KEY-VALUE.

Estas son BD que buscan almacenar cualquier cosa en tablas con 2 columnas. La primera es la clave que identifica la fila y la segunda columna el valor. En esta columna valor guardaremos cualquier cosa en formato binario, y es así como conseguiremos guardar tanto imágenes, vídeos, texto o la necesidad que tengamos. Lo malo es que no será interpretado por la BD pues al ser guardado en formato binario no se puede, para ello han surgido las orientadas a documentos que posibilitan esto mismo:

BD ORIENTADOS A DOCUMENTOS

Estas son la evolución a las BD Key-Value pero con la diferencia de que la parte que se almacena el "todo" del valor, no estará guardado en bytes como es en la BD clave-valor sino que esta vez se guardará en un formato que los gestores de BD puedan interpretar y operar con ellos. Como gran ejemplo es MongoDB que lo que hace es guardar el valor en formato JSON.

BD ORIENTADAS A GRAFOS.

Son BD muy diferentes y que difícilmente serán usadas en proyectos pues se almacena en esquema de grafo. Esto es muy bueno pues como sugiere la ciencia de la teoría de grafos, se busca optimizar los caminos entre nodos, por lo que si se imaginan las BD distribuidas entre nodos y tablas, habrá ciertos caminos de relaciones que serán más eficientes en búsquedas.

BD ORIENTADAS A OBJETOS.

Estas BD surgen debido a la necesidad de llevar el paradigma de programación orientada a objetos (POO) al mundo de las BD. Se buscaba evitar ajustar todos los modelos al típico modelo relacional-transaccional. Por lo tanto, aquí en estas BD se pueden tener conceptos como "herencia" entre objetos.

Además, entre los tipos de BD para grandes volúmenes de datos, en los proyectos BI se aconsejan utilizar las BD Columnares.

NOT ONLY NOSQL: BD COLUMNARES.

Son BD que surgieron en la época de los 80 y que actualmente están renaciendo con otras compañías. Son BD las cuales cambian la forma de guardar los elementos físicamente, en vez de guardar por filas esta vez se guarda por columnas, por lo que cuando se accedan a agregaciones por una dimensión (operación mítica en BI) se traerá solamente los datos en necesarios para operar, es decir, solamente la columna seleccionada y no todas. Mucho más rápido que estar leyendo varias y varias columnas de cada filas de todo el conjunto.

INTRODUCCIÓN A MONGODB

Se habla mucho de la famosa base de datos NoSQL Mongodb, la verdad es que su departamento de Marketing están haciendo una muy buena labor en temas de diseño y campañas de posicionamiento.



Mongodb es un almacén de datos no relacional para documentos JSON. Por no relacional, decimos que no almacena sus datos en tablas, como lo hace una base de datos relacional como Oracle, SQL Server o la archiconocida Mysql. **Esta BD NoSQL lo almacena en documentos JSON.**

¿Y qué es JSON? JSON se entiende por JavaScript Object Notation. Básicamente como un XML pero que es interpretado por Javascript (para que nos entendamos). ¿Y cómo luce esto? pues con una clave y un valor. En el lenguaje relacional de las BD una clave sería una columna y un valor sería el dato almacenado en esa columna.

Por ejemplo:

```
1.{nombre:"Santiago", apellido: "Bernabeu"}
```

En Relacional vendría a ser, imaginarnos una tabla:

| Nombre | Apellido |
|----------|----------|
| Santiago | Bernabéu |

Pero claro, si pensamos así, ¿qué ventajas se tienen? pues ninguna...pero es que en un JSON se podría hacer esto:

```
1.{nombre:"Santiago", hobbies:["fútbol", "baloncesto", "natación"]}
```

En Mongo, todo el potencial es que podemos tener filas con "columnas dinámicas"... lo que viene ser **SCHEMALESS, que no respeta un esquema, cada fila tiene una estructura diferente**. Si se imaginan que en la primera fila se tienen 2 columnas (2 claves JSON) y en la segunda fila se tienen 10 columnas (10 claves JSON), esto en una BD Relacional es imposible, se tienen que crear tablas con columnas predefinidas, aquí simplemente sería insertar un documento con una estructura cualquiera.

Esto es muy útil por ejemplo en un caso que me encontré en un cliente. Este cliente tenía 20 columnas para especificar tallas de productos, y claro, había columnas que no existían en los zapatos, como la talla XL pero sí la 40-41 de zapatos... pero las columnas estaban ahí. Usando Mongo DB podría evitarse tener columnas con valores vacíos y simplemente insertar filas así:

1.{producto: "zapatilla", tallas: ["40-41", "42-43", "43-44"]}

2.{producto: "camiseta", tallas: ["L"]}

Pero claro, ¿dónde está MongoDB en el mundo de las Bases de datos?



En los dos ejes de la gráfica: Escalabilidad vs rendimiento/funcionalidad, se podrá encontrar herramientas como "Memcached" o BD "Key/Value". Son perfectas para la escalabilidad y ofrecen rendimiento pero no ofrecen mucha funcionalidad. Por el otro lado están las relacionales, que son BD pensadas para dar soporte a la funcionalidad, hacer agregaciones, multitud de queries complejas con muchos joins, etc... pero el problema es que estas BD no pueden escalar en horizontal, pero si en vertical pues se puede añadir más CPU o Memoria.

Entonces, viendo esta gráfica, MongoDB intenta dar las 2 opciones... pero obviamente siempre va a carecer de algo al estar en medio.

MongoDB carece de joins por ejemplo, no puedes hacer joins entre "documentos". Para hacer joins lo tienes que hacer de forma externa a la BD. **Otra de las cosas que falta en "MongoDB" son las transacciones**, esta BD no está hecha para transacciones en múltiples colecciones pues los documentos son jerárquicos y hay que acceder de forma atómica, por lo que alguna operación transaccional podría requerir múltiples actualizaciones que en una BD Relacional equivale a una sola operación.

MongoDB tiene índices e índices secundarios pues obviamente buscan darle potencial a las búsquedas y es así como lo consiguen, como cualquier otra BD.

DESDE EL PUNTO DE VISTA DE LA ANALÍTICA DE DATOS

Son muchos los ambientes operacionales que están usando MongoDB como mejor alternativa a los relacionales. Pero claro, no están optimizados para analizar la información (agregaciones y etc). Entonces, ¿dónde se puede encontrar, los que se dedican al análisis de la información esta magnífica BD, MongoDB? Pues sencillamente en los orígenes de datos.

Por ello, un consultor BI tendrá que estar preparado para atacar a estos orígenes para hacer ETLs que extraigan la información, la manipule y las deposite en modelos de análisis. Como una BD relacional cualquiera...con la peculiaridad que en este caso se tendrá que interpretar el JSON en los procesos ETLs...

TRABAJANDO CON KETTLE Y MONGODB

Kettle ofrece muchos conectores a diferentes Bases de Datos. No iba a ser menos y el equipo de Pentaho, ante la demanda que tiene esta nueva BD NoSQL han decidido desarrollar 2 pasos para que se pueda leer y escribir datos en esta BD.

Recordando un poco, MongoDB es una BD NoSQL la cual funciona bajo documentos JSON. Es decir, que siempre que se queira insertar en la BD o extraer de la BD se tendrá que trabajar con JSON. Si se requiere operar con esta BD harbá que saber algún lenguaje de programación para poder insertar/consultar datos y se tendrá que disponer alguna librería para interpretar el documento JSON que devuelve la BD.

Con Kettle no es así, no habrá que escribir ninguna línea de código y en este apartado se va a mostrar 2 ejemplos.

ALCANCE

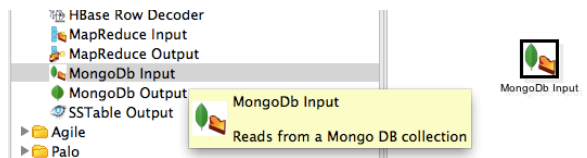
A continuación se generará 1000 datos de personas de forma aleatoria. Se busca tener una muestra de datos con el nombre, altura, peso y edad de personas. Todo ello lo se generará con funciones random() de Kettle.

Disponiendo de los datos se crearán unos procesos para:

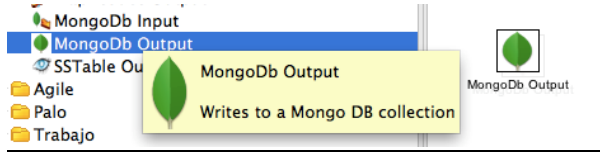
- Insertar 1000 datos en una "tabla" de MongoDB
- Leer 1000 datos de una tabla MongoDB.

TRABAJANDO

Si se abre Kettle y se crea una transformación (ktr), se podrá ver en la carpeta de "steps" de BigData que un usuario dispone de dos componentes de MongoDB. Tal que así:



MongoDb Input. Este paso lee de una colección de MongoDb



MongoDb Output. Paso el cual escribe en una colección de Mongoddb

ETL PARA INSERTAR DATOS EN MONGODB

Estos pasos de MongoDB se usarán, pero hay que generar los datos con una ETL. En este caso, con el primer paso de generar filas lo que se busca es generar 1000 filas, se le añade una secuencia para componer el nombre y después con el paso de Java se usan las funciones de Random() de java para poder obtener el peso, la altura y la Edad. Sencillo.

| # | Nombre | Peso | Altura | Edad |
|---|-----------|------|--------|------|
| 1 | Persona_1 | 94 | 101 | 33 |
| 2 | Persona_2 | 82 | 100 | 27 |
| 3 | Persona_3 | 72 | 195 | 76 |
| 4 | Persona_4 | 57 | 142 | 108 |
| 5 | Persona_5 | 82 | 199 | 25 |
| 6 | Persona_6 | 81 | 137 | 41 |
| 7 | Persona_7 | 90 | 149 | 82 |
| 8 | Persona_8 | 85 | 182 | 20 |

Una vez que se disponen las 1000 filas, lo que se busca es "unir" la salida del step "selección" a un "MongoDB Output". Si se configura el paso de Mongoddb, se tendrça que rellenar los siguientes campos: Host, Puerto, y después la BD y la colección donde se va a escribir. Si no existe la colección MongoDB por defecto la creará, no habrá que preocuparse en este aspecto.

A continuación, se realiza una consulta a MongoDB para obtener los datos de personas, en este caso es vacío el resultado:

```
MacBook-Air-de-IgnacioBustillo:MongoDB ignaciobustillo$ bin/mongo
MongoDB shell version: 2.4.8
connecting to: test
Server has startup warnings:
Tue Nov 26 19:38:05.830 [initandlisten]
Tue Nov 26 19:38:05.830 [initandlisten] ** WARNING: soft rlimits too low. Number
of files is 256, should be at least 1000
> db.personas.find();
>
```

Una vez visto que no se disponen de datos, se ejecuta la ETL para generar e insertar datos en MongoDB. Se disponen de estadísticas de ejecución:

| Nombre paso | Numero Copia | Leído | Escrito | Entrada | Salida | Actualizado | Rejected | Errores/Activo | Tiempo |
|------------------|--------------|-------|---------|---------|--------|-------------|----------|----------------|--------|
| Generar Filas | 0 | 0 | 1000 | 0 | 0 | 0 | 0 | 0 Finalizado | 0.0s |
| Añadir secuencia | 0 | 1000 | 1000 | 0 | 0 | 0 | 0 | 0 Finalizado | 0.0s |
| Calculadora | 0 | 1000 | 1000 | 0 | 0 | 0 | 0 | 0 Finalizado | 0.0s |
| Randoms | 0 | 1000 | 1000 | 0 | 0 | 0 | 0 | 0 Finalizado | 0.1s |
| Selección | 0 | 1000 | 1000 | 0 | 0 | 0 | 0 | 0 Finalizado | 0.1s |
| MongoDb Output | 0 | 1000 | 0 | 0 | 0 | 0 | 0 | 0 Finalizado | 0.2s |

A continuación se vuelve a consultar la base de datos, y se comprueba el contenido de Personas.

```

> db.personas.find();
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd8a"), "Nombre" : "Persona_1", "Peso" : NumberLong(61), "Altura" : NumberLong(161), "Edad" : NumberLong(92) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd8b"), "Nombre" : "Persona_2", "Peso" : NumberLong(54), "Altura" : NumberLong(177), "Edad" : NumberLong(50) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd8c"), "Nombre" : "Persona_3", "Peso" : NumberLong(59), "Altura" : NumberLong(155), "Edad" : NumberLong(97) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd8d"), "Nombre" : "Persona_4", "Peso" : NumberLong(51), "Altura" : NumberLong(168), "Edad" : NumberLong(96) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd8e"), "Nombre" : "Persona_5", "Peso" : NumberLong(76), "Altura" : NumberLong(131), "Edad" : NumberLong(41) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd8f"), "Nombre" : "Persona_6", "Peso" : NumberLong(76), "Altura" : NumberLong(195), "Edad" : NumberLong(31) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd90"), "Nombre" : "Persona_7", "Peso" : NumberLong(59), "Altura" : NumberLong(109), "Edad" : NumberLong(100) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd91"), "Nombre" : "Persona_8", "Peso" : NumberLong(52), "Altura" : NumberLong(187), "Edad" : NumberLong(108) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd92"), "Nombre" : "Persona_9", "Peso" : NumberLong(79), "Altura" : NumberLong(158), "Edad" : NumberLong(114) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd93"), "Nombre" : "Persona_10", "Peso" : NumberLong(59), "Altura" : NumberLong(182), "Edad" : NumberLong(49) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd94"), "Nombre" : "Persona_11", "Peso" : NumberLong(99), "Altura" : NumberLong(111), "Edad" : NumberLong(112) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd95"), "Nombre" : "Persona_12", "Peso" : NumberLong(69), "Altura" : NumberLong(174), "Edad" : NumberLong(37) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd96"), "Nombre" : "Persona_13", "Peso" : NumberLong(65), "Altura" : NumberLong(163), "Edad" : NumberLong(63) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd97"), "Nombre" : "Persona_14", "Peso" : NumberLong(72), "Altura" : NumberLong(181), "Edad" : NumberLong(49) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd98"), "Nombre" : "Persona_15", "Peso" : NumberLong(79), "Altura" : NumberLong(122), "Edad" : NumberLong(104) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd99"), "Nombre" : "Persona_16", "Peso" : NumberLong(65), "Altura" : NumberLong(103), "Edad" : NumberLong(63) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd9a"), "Nombre" : "Persona_17", "Peso" : NumberLong(98), "Altura" : NumberLong(172), "Edad" : NumberLong(93) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd9b"), "Nombre" : "Persona_18", "Peso" : NumberLong(63), "Altura" : NumberLong(162), "Edad" : NumberLong(95) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd9c"), "Nombre" : "Persona_19", "Peso" : NumberLong(62), "Altura" : NumberLong(165), "Edad" : NumberLong(20) }
{ "_id" : ObjectId("5294f0b330043cc9d8a3dd9d"), "Nombre" : "Persona_20", "Peso" : NumberLong(97), "Altura" : NumberLong(178), "Edad" : NumberLong(97) }
Type "it" for more

```

ETL PARA EXTRAER DATOS DE MONGODB

Una vez que se han insertado datos en la base de datos, se busca realizar la operación al revés, una lectura de datos

Primero lo que se busca es disponer de un paso de "Mongodb Input" en la ETL pues es así que se conseguirá sacar datos de la BD.

Si se realiza un "preview" y devolverá un formato JSON con los valores de personas.

The screenshot shows the 'MongoDB Input' configuration window with the following settings:

- Step name: MongoDB Input
- Host name or IP address: localhost
- Port: 27017
- Database: test
- Collection: personas
- Name of JSON output field: personasJSON
- Query expression (JSON):
- Fields expression (JSON):
- Authentication user:
- Authentication password:

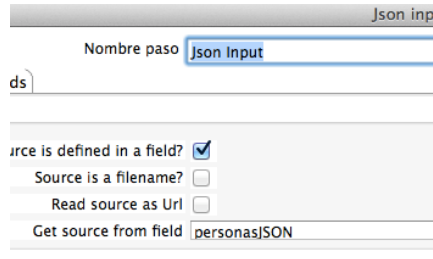
Below the configuration, the 'Examine preview data' section shows the following JSON output:

```

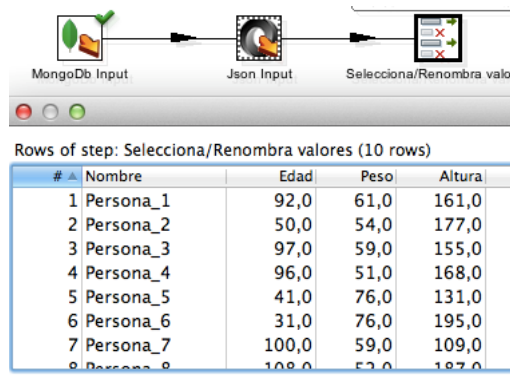
Rows of step: MongoDB Input (10 rows)
# :# personasJSON
1 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd8a" }, "Nombre" : "Persona_1", "Peso" : 61, "Altura" : 161, "Edad" : 92 }
2 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd8b" }, "Nombre" : "Persona_2", "Peso" : 54, "Altura" : 177, "Edad" : 50 }
3 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd8c" }, "Nombre" : "Persona_3", "Peso" : 59, "Altura" : 155, "Edad" : 97 }
4 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd8d" }, "Nombre" : "Persona_4", "Peso" : 51, "Altura" : 168, "Edad" : 96 }
5 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd8e" }, "Nombre" : "Persona_5", "Peso" : 76, "Altura" : 131, "Edad" : 41 }
6 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd8f" }, "Nombre" : "Persona_6", "Peso" : 76, "Altura" : 195, "Edad" : 31 }
7 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd90" }, "Nombre" : "Persona_7", "Peso" : 59, "Altura" : 109, "Edad" : 100 }
8 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd91" }, "Nombre" : "Persona_8", "Peso" : 52, "Altura" : 187, "Edad" : 108 }
9 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd92" }, "Nombre" : "Persona_9", "Peso" : 79, "Altura" : 158, "Edad" : 114 }
10 { "_id" : { "$oid" : "5294f0b330043cc9d8a3dd93" }, "Nombre" : "Persona_10", "Peso" : 59, "Altura" : 182, "Edad" : 49 }

```

Aun así, aún no se ha acabado pues no se disponen los datos por columnas, simplemente se disponen de documentos JSON con datos de personas. A continuación habría que añadir pasos para interpretar el formato JSON y obtener los datos para manipularlos de alguna forma.

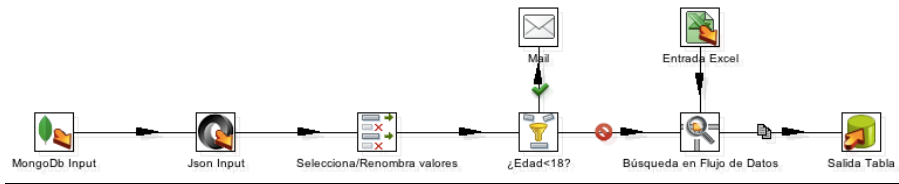


Una vez que se ha configurado el paso de JSON para que extraiga los datos de la colección, hacemos un preview y devuelve lo siguiente:



Con la unión de esos dos pasos se ha conseguido extraer los datos de MongoDB y se ha interpretado los documentos que se han obtenido de la colección de Personas.

A continuación se muestra un ejemplo de una ETL más rebuscada la cual busca obtener los datos de las personas y en caso de que la edad sea menor de 18 se mandará un Email para notificar a un administrador. En caso contrario se cruzará con un Excel y se depositará en una tabla.



AMAZON S3 Y ANALÍTICA DE DATOS

Amazon S3 es, de forma sencilla, un disco duro en internet. No se necesita tener un sistema operativo para tener tus ficheros de hasta 5 terabytes de datos en la nube, pudiendo dar acceso a innumerables personas e interactuar con estos archivos a través de servicios web que nos proporciona Amazon.



Para poder crear un espacio S3, primero hay que registrar una cuenta en Amazon, obviamente habrá que dar una tarjeta de crédito porque en caso de pasarse del programa gratuito empezarán a cobrar por lo que se use.

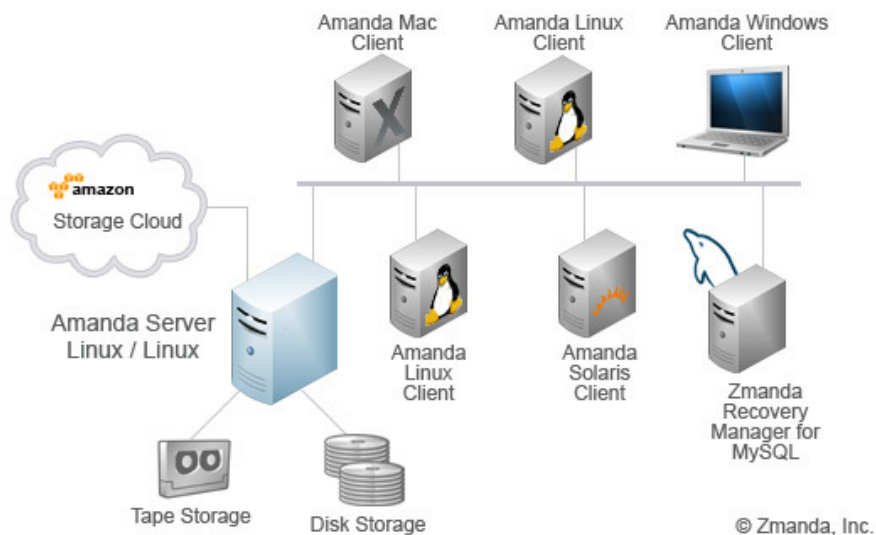
Según la política de la empresa:

"Como parte de la Capa de uso gratuito de AWS podrá empezar gratis con Amazon S3. Al registrarse, los clientes nuevos de AWS reciben 5 GB de almacenamiento estándar en Amazon S3, 20 000 solicitudes GET, 2 000 solicitudes PUT y 15 GB de transferencia de datos saliente al mes durante un año."

CÓMO FUNCIONA S3

S3 es como un disco duro en la nube. Para ello, hay que crear un "Bucket" (cubo) que permita meter datos. Este "Bucket" es como una caja contenedora de carpetas y archivos. La forma de acceder es `s3://BUCKET/CARPETA/ARCHIVO`. Se puede acceder desde un navegador o lenguaje de programación a archivos siempre y cuando estos tengan los permisos de acceso, que se pueden configurar.

Ejemplo de esto son los datasets que tienen en Amazon para jugar con Big Data: <http://aws.amazon.com/datasets>



CASOS DE ANALÍTICA DE DATOS

Este servicio de S3 que provee Amazon, viene genial a los consultores BI para poder dejar datos de histórico acumulados por ficheros. ¿Y por qué dejarlos aquí? Sencillamente porque es gratuito y lo se tendrá a mano. Además, se podrá de forma sencilla importar estos datos a otros servicios como Amazon EC2 para operar con ellos con instancias gratuitas o para hacer BigData. A continuación se explican dos casos que podrían darse y vendría bien Amazon S3:

Se dispone de histórico de 2013 de las ventas de una compañía en formato flat table particionados por ficheros por días.

- 1) Se busca hacer datamining de este dataset, por ejemplo un algoritmo de Clustering. Al tenerlos en la nube, lo que se puede hacer es crear una instancia EC2 de Amazon, desplegar R, importar el dataset con R y trabajar con el dataset. Sería tan sencillo trabajar con estos datos que solamente habrá que poner un path de referencia al S3: `S3://bucket/carpeta/fichero.txt` Al final de todo, se cierra la instancia BigData e interpretamos la conclusión obtenida de R
- 2) Se quiere obtener el promedio por mes de ventas y queremos sacar los datos en un archivo. En vez de importar todo esto a una BD que quizás no soporte este volumen de datos, lo que se podría hacer es levantar un sistema BigData Amazon EMR que nos monte un cluster según unas preferencias con Hadoop y Hive, y después lo único que se tendría que hacer es crear una tabla externa Hive apuntando a los datos. Solamente se tendría que hacer consultas desde, por ejemplo, Kettle.

INTRODUCCIÓN A AMAZON ELASTIC MAPREDUCE

¿Alguna vez el lector ha desplegado un sistema BigData? Sabrá que es "sencillo" pero también un poco tedioso. Puede ser que tenga unos scripts de configuración automáticos y facilite el despliegue.



Otra cosa que sabrá, es que para desplegar un cluster es necesario máquinas físicas, y al final eso se traduce dinero. No podrá tener 3 máquinas para hacer un test, y mañana comprar 10 más que quizás se queden obsoletas en 1 año. Amazon ha pensado en los consultores con necesidades temporales de BigData y ha inventado Amazon EMR.

En caso de que el lector nunca haya desplegado un sistema BigData, no tiene porqué preocuparse, Amazon ya ha pensado en estas personas y han facilitado las cosas.

AMAZON ELASTIC MAP REDUCE

Amazon EMR, es un servicio web que permite desplegar de forma sencilla un sistema Big Data con una configuración a medida de forma sencilla mediante un formulario. Con EMR se puede contar al instante con capacidad extra para realizar tareas de uso intensivo de datos en aplicaciones como indexación web, extracción de datos, análisis de archivos de registro, aprendizaje automático, análisis de datos...etc.


Amazon Elastic MapReduce permite centrar los esfuerzos en el procesamiento o analítica de datos sin que un cliente tenga que preocuparse en crear y configurar esa estructura BigData. Ya lo hacen por el cliente.

CÓMO FUNCIONA

Amazon ha creado un formulario el cual permite configurar un cluster BigData. EMR utiliza Apache Hadoop como motor de procesamiento distribuido. Hadoop es un framework de software Java de código abierto que permite utilizar aplicaciones de uso intensivo de datos que se ejecutan en agrupaciones de equipos sencillos de gran tamaño. Hadoop implementa un modelo informático denominado "MapReduce" que divide el trabajo en pequeños fragmentos, cada uno de los cuales puede ejecutarse en cualquiera de los nodos que forman la agrupación de equipos.

Por ello, permiten elegir la configuración de Hadoop en tu cluster, permitiéndote elegir las versiones de Hadoop estables y otras aplicaciones como Hive, Pig o HBase (no en la captura)

Software Configuration







Hadoop distribution  Amazon

- 3.0.2 (Hadoop 2.2.0)
- 3.0.1 (Hadoop 2.2.0)
- ✓ 2.4.2 (Hadoop 1.0.3) - latest
- 2.4.1 (Hadoop 1.0.3)
- 2.3.6 (Hadoop 1.0.3)
- 2.2.4 (Hadoop 1.0.3)
- 2.1.4 (Hadoop 0.20.205)

Use Amazon's Hadoop distribution. [Learn more](#)

Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

Use MapR's Hadoop distribution. [Learn more](#)

| Applications to be installed | Version | | | |
|------------------------------|----------|---|---|---|
| Hive | 0.11.0.1 |  |  |  |
| Pig | 0.11.1.1 |  |  |  |

Además, otra de las cosas a destacar, es que se puede lanzar un cluster bajo demanda de máquinas. Es decir, se puede configurar la tipología según la potencia que se quiera tener, no es lo mismo 20 máquinas de 2gb de ram que 10 máquinas de 10gb de ram. Tampoco el precio. Por ello, permiten elegir la tipología de la máquina Master y los esclavos que pueden ser de tipo Core (alojan datos y ejecutan tareas) y las de solamente Tareas.

EC2 availability zone Launch the cluster in a specific EC2 Availability Zone.

| | EC2 instance type | Count | Request spot | |
|---------------|---------------------------------------|--------------------------------|--------------------------|---|
| Master | <input type="text" value="m1.small"/> | <input type="text" value="1"/> | <input type="checkbox"/> | The Master instance assigns Hadoop tasks to core and task nodes, and monitors their status. |
| Core | <input type="text" value="m1.small"/> | <input type="text" value="2"/> | <input type="checkbox"/> | Core instances run Hadoop tasks and store data using the Hadoop Distributed File System (HDFS). |
| Task | <input type="text" value="m1.small"/> | <input type="text" value="0"/> | <input type="checkbox"/> | Task instances run Hadoop tasks. |

Los precios en Marzo 2014 son los siguientes:

Precios de Amazon EC2 (bajo demanda) y Amazon Elastic MapReduce

| Región: EE.UU. Este (Norte de Virginia) | | |
|--|----------------------|------------------------------------|
| | Precio de Amazon EC2 | Precio de Amazon Elastic MapReduce |
| Instancias según demanda estándar | | |
| Pequeño (predeterminado) | \$0,06 por hora | \$0,015 por hora |
| Mediano | \$0,12 por hora | \$0,03 por hora |
| Grande | \$0,24 por hora | \$0,06 por hora |
| Extragrande | \$0,48 por hora | \$0,12 por hora |

CONCLUSIÓN

Como se puede ver de forma introductoria, lo que permite Amazon Elastic MapReduce es evitar preocupaciones y directamente, con pocos clicks y 5 min de espera, se podrá contar con un cluster de X máquinas de Y potencia funcionando sin preocupaciones y bajo demanda.

¿Cómo se podría trabajar en el mundo real con esto? Muy sencillo. Ejemplo:

Se disponen de las ventas de los supermercados de una importante marca en España, en formato "flat table" en ficheros txt en [Amazon S3](#). En base a este histórico, se quiere calcular el promedio de ventas por mes para ver una evolución de nuestra actividad. Se disponen de miles de millones de datos divididos en ficheros de 100mb. La estrategia sería:

- Crear un cluster de 50 máquinas xlarge con Hadoop 1.0 y Hive 0.11,
- Crear una tabla en Hive externa que apunte a esos ficheros y ejecutamos una consulta sobre ellos.
- Ejecutar consulta HQL. Hive se encarga de traducir el HQL a MapReduce y obtendríamos el resultado. Una vez obtenido el resultado eliminamos nuestro cluster.

Tiempo estimado en creación de este cluster rondará los **10 min**. El tiempo de la consulta quizás podría tomar 1 hora, por lo que el precio del cluster de 50 máquinas de tipo "grande" durante 1 hora es 3\$. El **beneficio que se obtiene sería ahorro en inversión y mantenimiento de 50 servidores** en una empresa y tiempo de configuración de los 50 clusters de un especialista BigData.

KETTLE BIGDATA: AMAZON EMR Y S3

Como he repasado en puntos anteriores, Kettle ofrece gran conectividad con herramientas BigData como por ejemplo Hadoop, Hive o BD como Cassandra o MongoDB.



No solamente se dispone de esta conectividad a herramientas tan importantes en el ecosistema Hadoop, sino que además se dispone de conectividad a servicios cloud tan importantísimos en el mundo BigData como por ejemplo [Amazon S3](#) o [Amazon Elastic MapReduce](#) que se vio anteriormente.

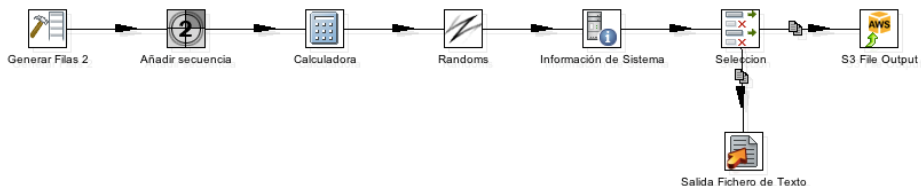
Es por ello que a continuación se van a repasar cómo trabajar con estos pasos.

ALCANCE

Se busca explicar qué pasos se tienen en Kettle para ejecutar ETLs con Amazon WebServices, concretamente con el servicio de Amazon S3 (un almacenamiento en la nube) y con Amazon Elastic MapReduce, con el que se podrá crear un sistema BigData y ejecutar trabajos MapReduce o Scripts en Hive al vuelo.

TRABAJANDO CON AMAZON S3

Un ejemplo muy sencillo que permite trabajar con Amazon S3 es el "S3 File Output". Básicamente lo que hace es subir un fichero a la nube y dejarlo en un "bucket" de Amazon S3. En el ejemplo a continuación es una ETL que genera filas con una serie de propiedades y exporta a la nube y a local el fichero de forma sencilla.



El paso "SE File Output", ofrece un formulario muy similar al de "Salida Fichero de Texto", simplemente habría que decirle dónde se quiere guardar la información, qué formato, la separación del contenido y los campos a escribir. Pero cuidado, la novedad aquí es el "Access Key" y "Secret Key" que Amazon proporciona, pues es así que Kettle se conectará a un "bucket" de Amazon S3 y escribirá este fichero en la nube.

Fichero | Contenido | Campos

Access Key: [.....]

Secret Key: [.....]

Nombre Fichero: s3://@s3/bucket/file [Examinar...]

¿Ejecutar como comando?

Do not create file at start

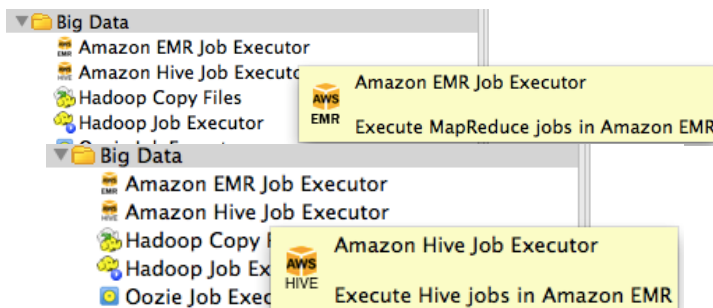
Accept file name from field?

File name field: [.....]

Extensión: csv

TRABAJANDO CON AMAZON EMR

Para trabajar con Elastic MapReduce de Amazon se tienen 2 opciones. El primer paso (izq) es el Amazon EMR Job Executor y el segundo (drcha) es el Amazon Hive Job Executor:



Amazon EMR Job Executor

Como se vio en el anterior punto de Introducción a Amazon MapReduce, un especialista BigData podría querer ahorrarse el tiempo de creación de un cluster y automatizar la creación gracias a este servicio que nos proporciona Amazon. Pentaho no ha querido perder esta oportunidad y ha creado un "step" o paso en Pentaho Data integration, donde permite a través de unas credenciales de Amazon, elegir un set de datos en S3 y ejecutar una tarea MapReduce sobre estos datos y en un cluster creado al vuelo. Para ello, el step tiene un formulario en el cual podemos elegir el número de instancias, el tipo de instancias, la tarea MapReduce que ejecutaremos y el set de datos donde reside en S3.

Amazon Hive Job Executor

Al igual que el anterior, PDI permite levantar un cluster al momento y en vez de ejecutar una tarea MapReduce, se podrá configurar para que ejecute un script de Hive, por ejemplo. En este script podemos crear una tabla que apunte a un set de datos S3 y ejecutar una query/consulta en HQL. Todo en el proceso de ejecución de la ETL.

CONCLUSIÓN

Como se puede ver, esto ofrece unas posibilidades muy grandes en cuanto a "orquestación" de tareas pues utilizamos todo el potencial de los servicios cloud pero orquestado en un servidor.

Por ejemplo, si todos los días se necesita crear una ETL la cual exporte a un almacenamiento en la nube todos los nuevos registros que se han producido hoy y que se ejecute después un algoritmo de minería de datos para clasificación (clustering) sobre el set masivo de datos actualizado. Para ello se necesitaría un cluster de máquinas y una tarea MapReduce que aplique la ejecución en distribuido del algoritmo de minería de datos. Se tendría que hacer lo siguiente:

- Crear una transformación para extraer los datos del origen y usar el paso de exportación a S3.
- Crear un trabajo que cree un cluster BigData y ejecute un MapReduce que clasifique los elementos con un algoritmo de minería de datos de Clustering sobre todo el set. Se usaría Amazon EMR Job Executor y se elegiría el directorio S3 que se ha usado en el punto 1 y un jar con la tarea MapReduce creada a medida con el algoritmo en cuestión.
- Se tendría una ETL programada de forma sencilla con un cronjob que llame por consola a Kitchen (herramienta para ejecución de trabajos). Kitchen se encargaría de ejecutar los puntos 1 y 2.

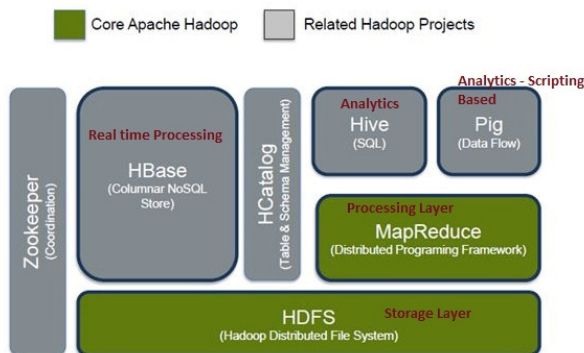
Por lo tanto, haciendo estos pasos se tendría un sistema de procesamiento masivo de datos de fácil mantenimiento y con seguridad ahorraría bastante dinero pues **no habría que mantener una infraestructura BigData las 24horas al día**, sino solamente **pagar por el uso de la misma**, en este caso, el tiempo que lleve crear y lanzar la tarea de Clustering.

9.000.000.000 DE ELEMENTOS EN UNA TABLA DE HECHOS

¿Es posible de almacenar 9 mil millones de elementos en una tabla de hechos? Posible es, pero las consultas serán eficientes de cara al usuario? Depende. Está claro que si utilizamos una BD Relacional tendríamos que tunear muy mucho este gestor para obtener buen rendimiento, utilizar ciertos métodos de particionado y de mejoras en consultas. ¿Es la mejor forma? Lo más seguro es que la BD no responda en una consulta. En cambio, si se utilizan tecnologías Big Data, esto ya empieza a cambiar. **Se podrá escalar la BD y utilizar operaciones de forma distribuida en un cluster que de potencia.**

INTRODUCCIÓN

Lo que se va a hacer es bien simple, es desplegar todos los datos que se disponen de histórico en el sistema distribuido de archivos en diferentes máquinas, esto se hará para que varias máquinas trabajen a la vez y procesen más archivos y datos en paralelo en menor tiempo. MapReduce es la capa de procesamiento que se usará en la distribución de tareas en los nodos (que tienen un conjunto de datos limitados) y sacar estadísticas de toda la colección de datos que se disponen con la filosofía "divide y vencerás". Pero no se busca programar en Java estos algoritmos, se busca usar la pieza del puzzle que hará transformar pseudo-consultas SQL a MapReduce sobre esos archivos. Hive será el encargado de ello y se interactuará con esta herramienta usando HQL o lo que viene siendo un pseudo-SQL.

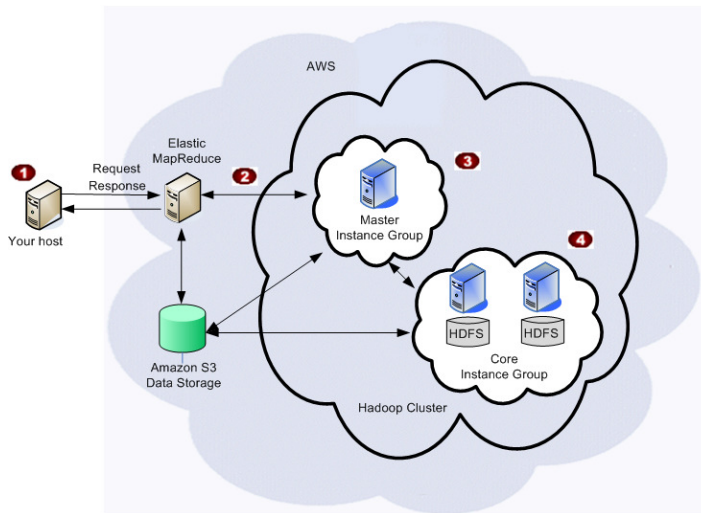


ARCHITECTURA

En este punto habría muchas opciones. En este caso se usará una que da la posibilidad de tener X nodos cuando sea necesario durante el tiempo que se quiera. No se requiere hacer ninguna compra de servidores, simplemente se busca alquilarlos por uso, y eso es Cloud Computing. Quien provee esa posibilidad es Amazon, como se ha visto en puntos anteriores.

En el próximo gráfico, se puede ver la arquitectura elegida. Se dispone de lo siguiente:

- Servidor que interactuará con la estructura BigData.
- Cluster Big Data Amazon EMR. Son X servidores que hemos contratado a Amazon. Hay 2 tipos: Master (que es un nodo) y Slaves (que son 20 máquinas).
- Servidor de almacenamiento Amazon S3. Es quien contendrá todos los datos.



El servidor que se dispone para orquestar con la arquitectura interactúa de 2 formas con ella:

1) **Generación de datos.** Se busca tener los datos en Amazon S3 y para ello se generarán datos en local y se subirán a la nube, todo ello supervisado y orquestado por Kettle.

2) **Consultas analíticas.** En este caso, como se busca hacer preguntas sobre los datos, se tienen dos opciones, utilizar Kettle para realizar consultas periódicas a través de un paso "Extraer de tabla" o con la consola que provee Hive. Se utilizará la segunda opción

RESULTADO

El resultado ha sido sorprendente a la par que sencillo. Obviamente, se cuenta con un cluster muy pequeñito de nodos: 20 nodos. Se pueden detallar las siguientes conclusiones:

Dificultad: Nula. NO es nada difícil el usar Amazon EMR para crear al momento los clusters y atacar un set de datos en Amazon S3.

Tiempo de puesta en marcha: 15min en el peor de los tiempos (creación manual de cluster e interacción manual con hive). Con Kettle se puede hacer una ETL automática que orqueste todas estas operaciones: levantar el cluster, creación de tabla apuntando a los datos en S3 y ejecutar la consulta.

NO es para tiempo real. Obviamente, como se ha visto anteriormente, no está hecho para "real time" o "near real time", existen soluciones que seguro que son mejores.

Ideal para batch processing. Este tipo de operaciones deberían ser por la noche y ejecutando ETLs que dejen datos agregados por la noche en un DW.

Coste: el coste ha sido el siguiente de 1 hora de cluster (20 máquinas): -> \$0.240 per M1 Standard Large (m1.large) Linux/UNIX instance-hour (or partial hour) -> 21 Hors -> 5\$.

SOBRE STRATEBI

Stratebi es una empresa española, con oficinas en Madrid y Barcelona, especializada en Business Intelligence, Big Data y Visualización con tecnologías Open Source

Esta experiencia y conocimientos, adquirida durante la participación en proyectos estratégicos en compañías de reconocido prestigio a nivel internacional, se ha puesto a disposición de nuestros clientes a través de Stratebi.

En Stratebi nos planteamos como objetivo dotar a las compañías de herramientas y servicios profesionales, escalables y adaptados a sus necesidades, que conformen una estrategia Business Intelligence capaz de rentabilizar la información disponible en la empresa.

"Dar valor a la información y convertirla así en un activo para la empresa son sus principales objetivos"

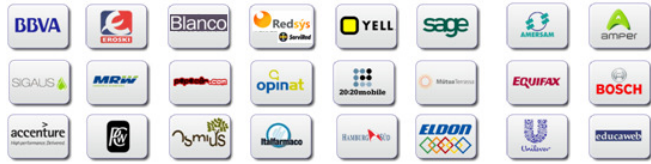
Dadas las características de complejidad técnica y de ser crítico para el negocio, los sistemas Business Intelligence demandan de personal altamente experimentado y que puedan garantizar el éxito de un proyecto.

Stratebi organiza junto a Medialab Prado los **Eventos y Talleres OpenAnalytics** (<http://www.openanalytics.es/>)



Nuestra experiencia con empresas de todos los sectores, tamaños y complejidades nos ha llevado a comprender la problemática en este área y a plantear soluciones y herramientas que solventen las carencias que se detectan.

Sector Privado



Sector Público



Más información
www.stratebi.com
info@stratebi.com

91.788.34.10 (Madrid)

93.425.50.10 (Barcelona)