Impala

¿Qué es Impala?

Es un motor SQL de elevado performance pensada para operar sobre grandes volúmenes de datos

- El motor es MPP: procesamiento masivo en paralelo
- Con latencias de Milisegundos

Impala corre sobre clusters Hadoop

- Puede ejecutar Queries sobre HDFS o Hbase
- Lee y escribe datos sobre ficheros con tipos de datos típicos de Hadoop

Originalmente desarrollada por Cloudera

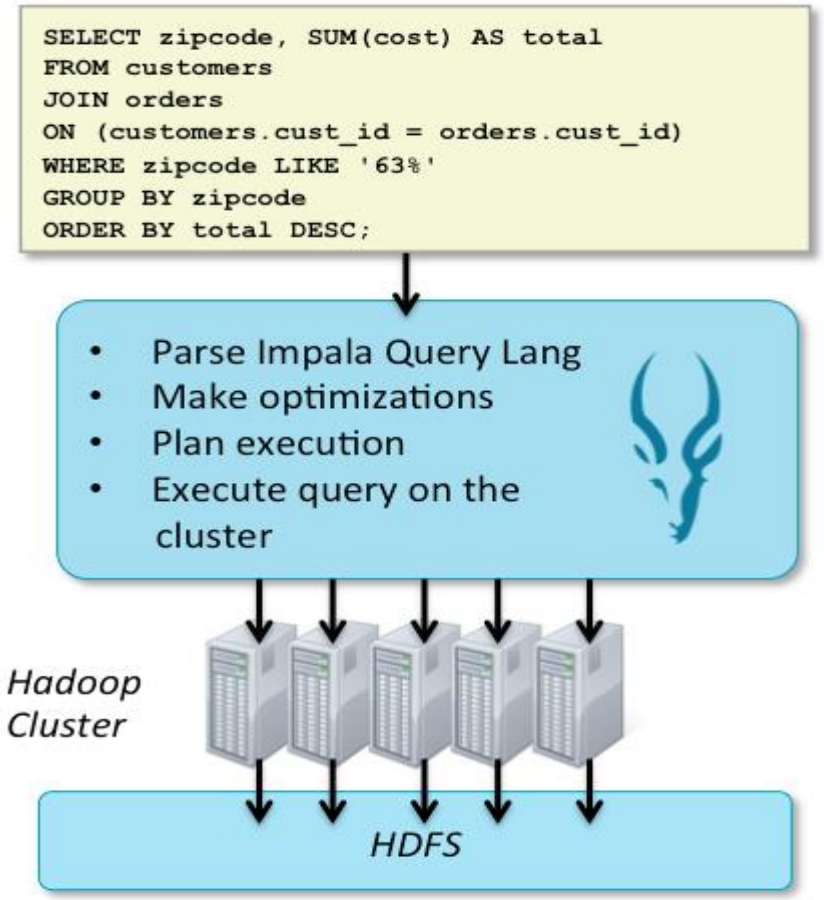
- Hoy en día es un proyecto perteneciente al ASF (Apache Software Foundation)
- Es 100% open source
- Todavía está en proceso de incubación

Impala: características

Impala ejecuta las queries directamente sobre el cluster en lugar de ejecutar MapReduce para procesar

Es en torno a unas **5 veces más rápido** que Hive o Pig, aunque a menudo puede ser hasta 20 veces más rápido.

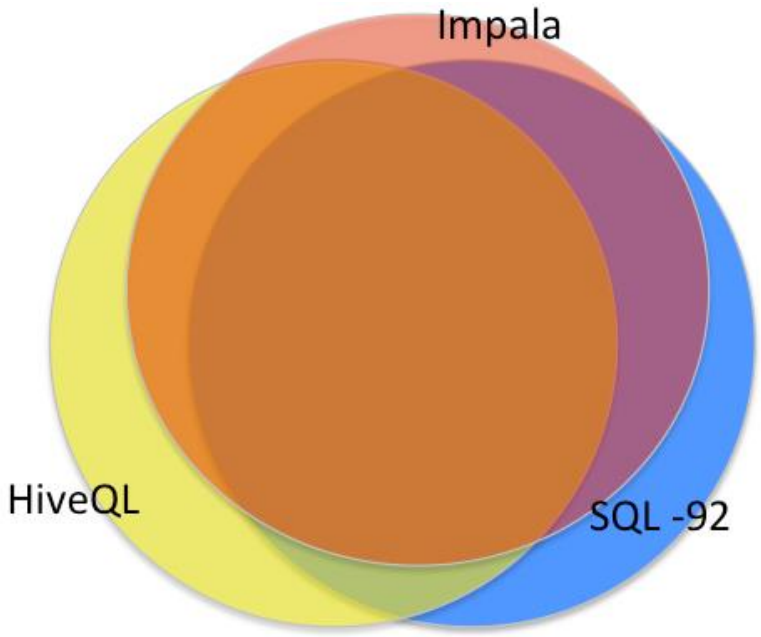
Está especialmente optimizado para ejecutar Queries.



Bosonit

Impala: características

Este es un gráfico de **similitudes** entre tres de las principales herramientas de procesado de datos en Hadoop.



Bosonit

Impala: por qué usarlo

Es más efectivo que programar Map Reduce

- Ya hemos visto que 10 líneas en SQL pueden equivaler a 70 líneas Java

Hay mucha más gente en el mundo capaz de escribir SQL en lugar de Java

- De esta manera se llega a mucha más gente
- No hace falta tener gran experiencia en programación
- Se puede aprovechar el Código SQL existente

Ofrece interoperabilidad con otros sistemas

- Al final es extensible mediante Java
- Muchas herramientas de BI soportan Hive o Impala

Bosonit

Impala: comparativa con Hive

Hive fue una de las primeras herramientas que formaron parte del entorno de producción en soluciones **basadas en Hadoop**, por lo que, a día de hoy, ofrece mucha más funcionalidad.

Hive e Impala son herramientas diferentes, pero ofrecen bastantes similitudes

- Ambas usan **variantes de SQL**
- Ambas **comparten** el mismo **metastore** y **data warehouse** en el cluster
- En proyectos grandes se suelen usar juntas

Las **queries** en Impala y Hive **operan sobre tablas**, igual que en RDBMS(Sist. De db relacional)

- Como sabemos, una tabla Hive no es más que una ruta en el cluster que contiene archivos

La **estructura** y **localización** de las tablas **se definen** cuando se **crean**

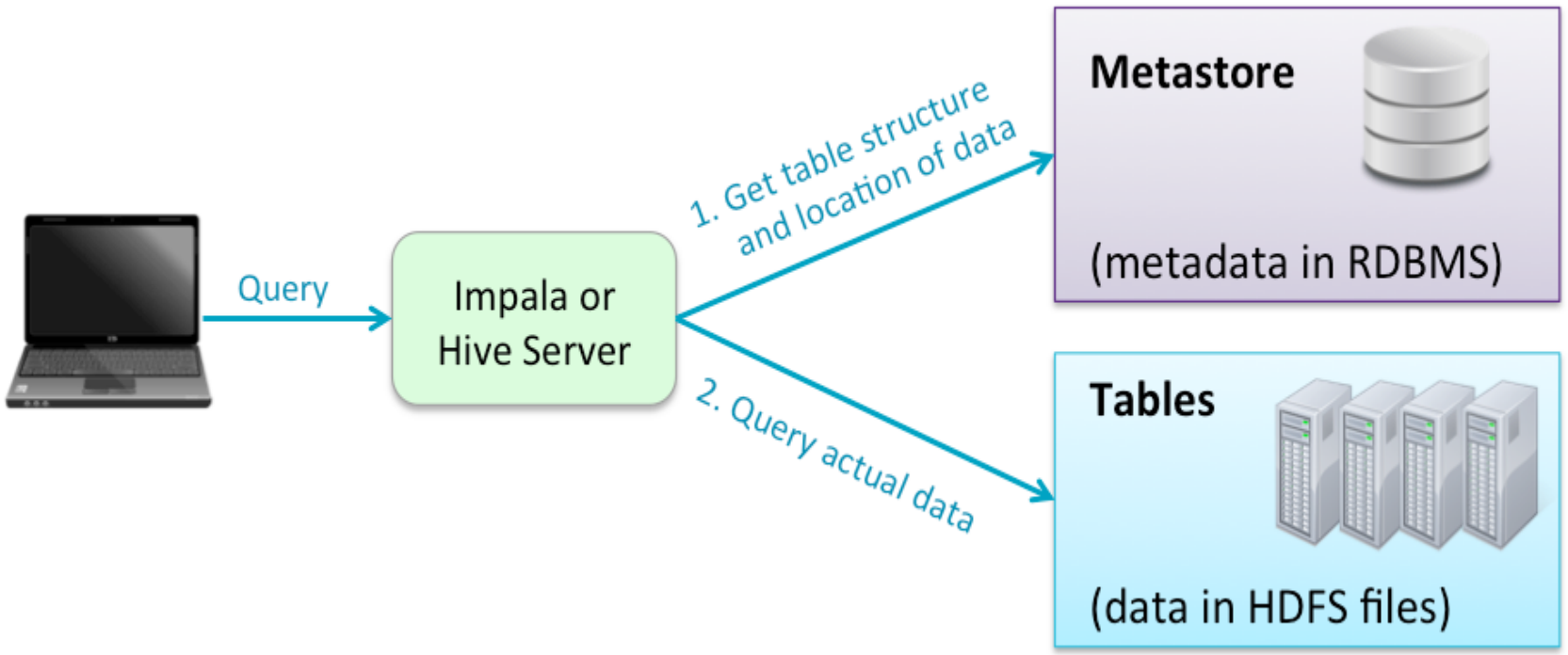
- Dicha información se almacena en el Metastore
  - Almacenado en una RDBMs, por ejemplo MySQL
  - Almacena metadatos de tablas de Hive/Impala

Impala, como Hive, operan sobre los mismos datos

- Tablas en HDFS, y metadatos en el Metastore

Impala: comparativa con Hive

El flujo sería el siguiente:



Bosonit

Impala: comparativa con Hive

Actualmente hay una serie de funcionalidades que Impala no soporta pero Hive sí

- Ficheros con tipos de datos a medida
- El tipo de datos DATE
- Funciones XML y JSON
- Algunas funciones de agregación como: “covar_pop, covar_samp, corr, percentile, percentile_approx, histogram_numeric, collect_set”
- Sampling (que es el ejecutar queries sobre un subset de una tabla en lugar de sobre toda la tabla)
- Vistas laterales (sobre una columna de una tabla, fila a fila, se le aplica una función (que crea un resultado en forma de vista en ejecución) y sobre ese resultado se aplica otra función, que es lo que se muestra.
- Multilples cláusulas DISTINCT por query
- UDFs (soportadas a partir de impala 1.2)
- Hay más diferencias que quedan fuera del alcance de este curso

Bosonit

Impala y Hive: comparativa con RDBMs

Como sabemos, el formato tradicional tiene muchas ventajas

- Tiempos de respuesta rápidos
- Soporta transaccionalidad
- Permiten modificaciones en registros
- Pueden servir a muchos clientes a la vez

Hadoop no es una RDBMs

- Al procesar datos con Hive o Impala tenemos las limitaciones de Hadoop (write once read many)
- Impala es más rápido que Hive, pero aun así, no es un sustitutivo válido para una base de datos OLTP (online transaction processing)
- No soporta transaccionalidad

Una de las grandes ventajas de estas dos herramientas frente a RDBMS es que te permiten definir el esquema de los datos después de que estén en el cluster, no antes, como ocurre con las otras.

Bosonit

Impala y Hive: comparativa con RDBMs

Tabla comparativa

	Relational Database	Hive	Impala
Query language	SQL (full)	SQL (subset)	SQL (subset)
Update individual records	Yes	No	No
Delete individual records	Yes	No	No
Transactions	Yes	No	No
Index support	Extensive	Limited	No
Latency	Very low	High	Low
Data size	Terabytes	Petabytes	Petabytes

Bosonit

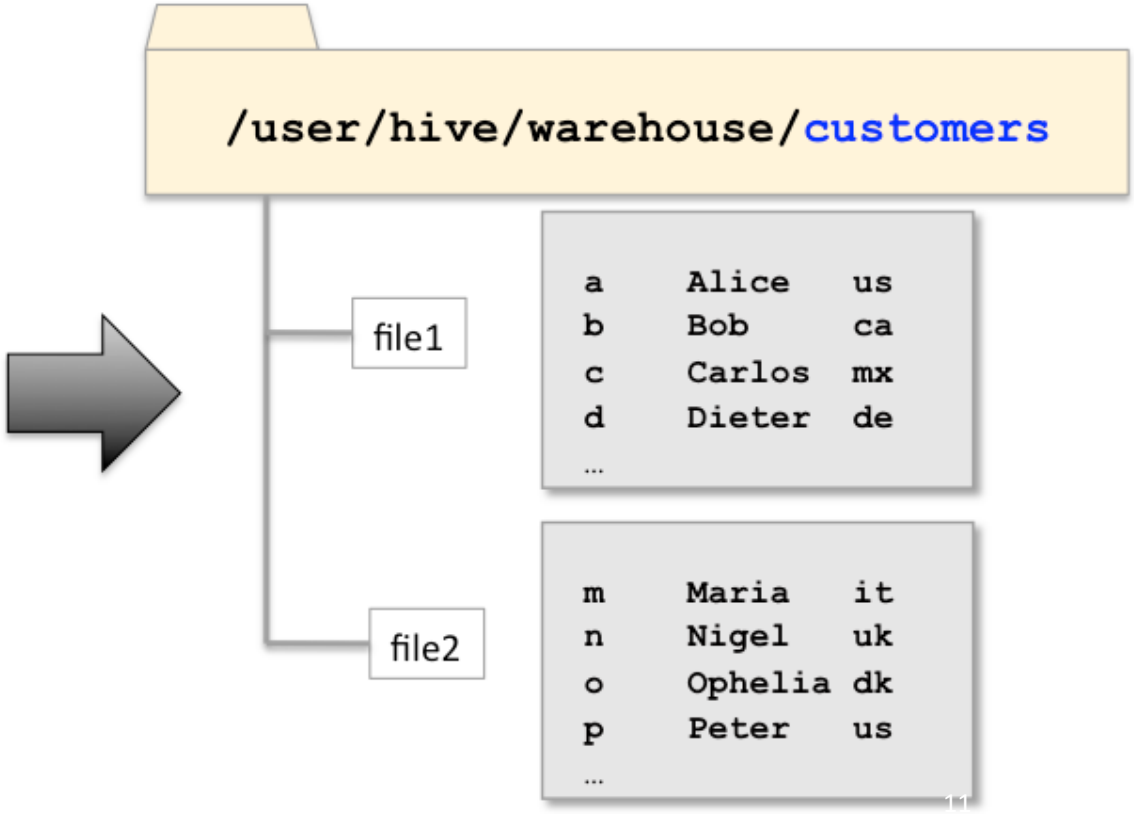
Impala y Hive: Data Management

Por defecto, los datos en hive e impala son almacenados en HDFS en la ruta /user/hive/warehouse

Cada tabla es un subdirectorio que contiene un número de ficheros

customers table

cust_id	name	country
001	Alice	us
002	Bob	ca
003	Carlos	mx
…	…	…
392	Maria	it
393	Nigel	uk
394	Ophelia	dk
…	…	…

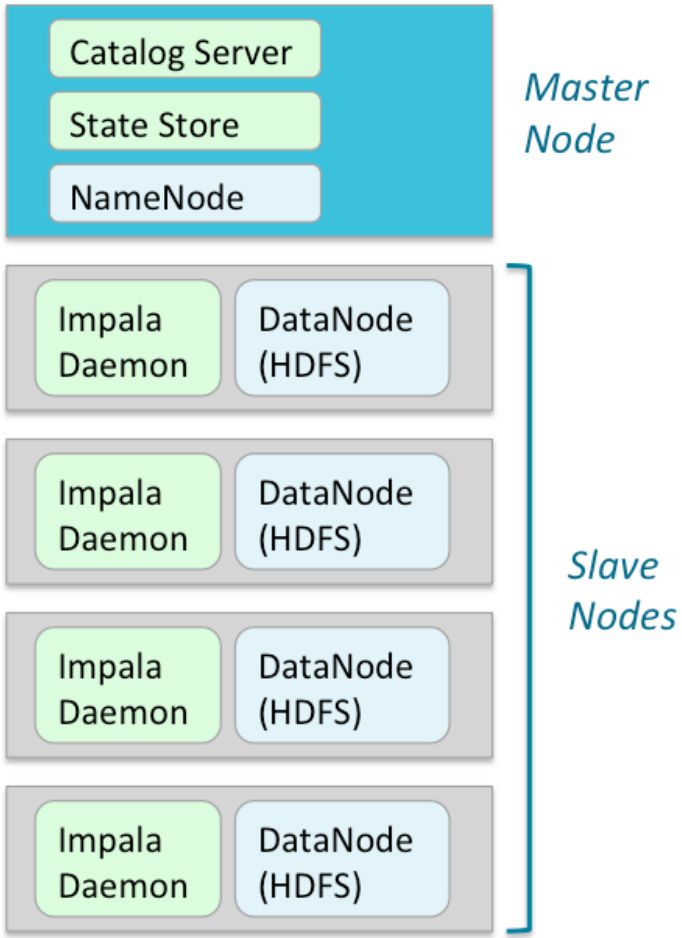


Impala: en el cluster

Cada nodo esclavo (slave), en el cluster, tiene un demonio Impala

Hay otros dos demonios en los nodos maestros que dan soporte a la ejecución de las queries

- **State Store**:
  - Es un servicio de look up para los demonios impala que corren en los nodos esclavos.
  - Periódicamente comprueban su estado
- **Catalog**
  - Transmiten los cambios en los metadatos a todos los demonios del resto del cluster

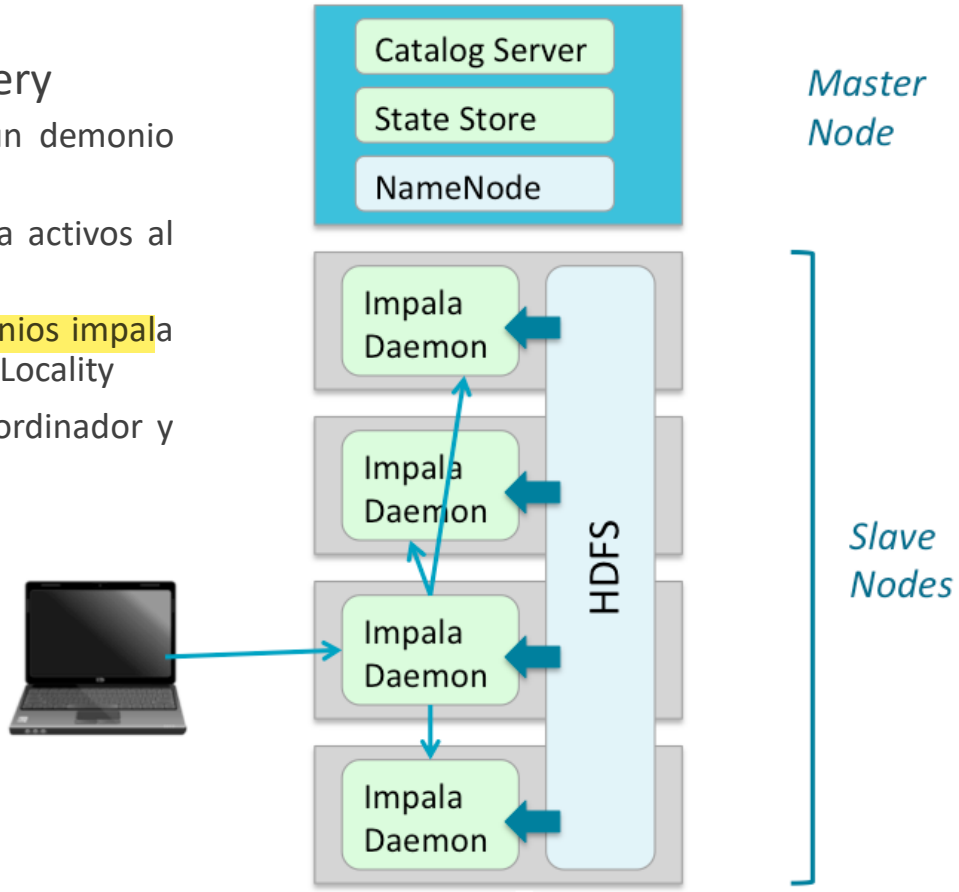


Bosonit

Impala: ejecución de una query

Es el demonio Impala quien planifica la query

- El cliente (impala-Shell o Hue) se conecta con un demonio local impala, llamado *coordinador*
- El *coordinador* pide una lista de demonios impala activos al State Store
- El *coordinador* distribuye la query entre los demonios impala disponibles, siempre intentando mantener el Data Locality
- Cada demonio va devolviendo el resultado al coordinador y este lo combina y lo muestra al cliente



Bosonit

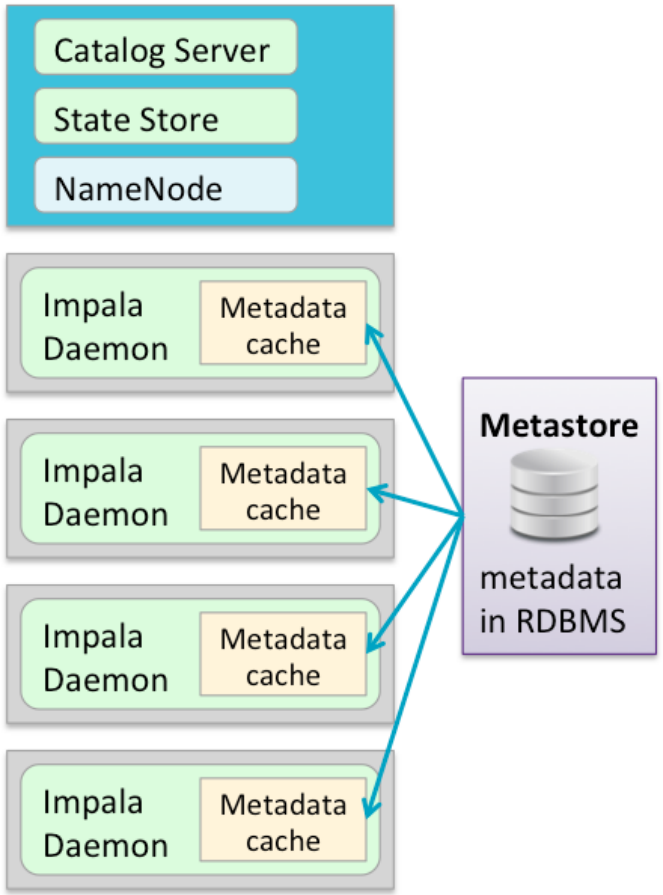
Impala: Cacheo de metadatos

Los demonios impala cachean los metadatos

- Tablas de definición del esquema
- Localización de donde están los bloques que componen una tabla en HDFS

Los metadatos son cacheados desde el metastore al inicio de la sesión.

Si los datos en el metastore cambian, hay que actualizar el metadata cache de cada nodo del cluster

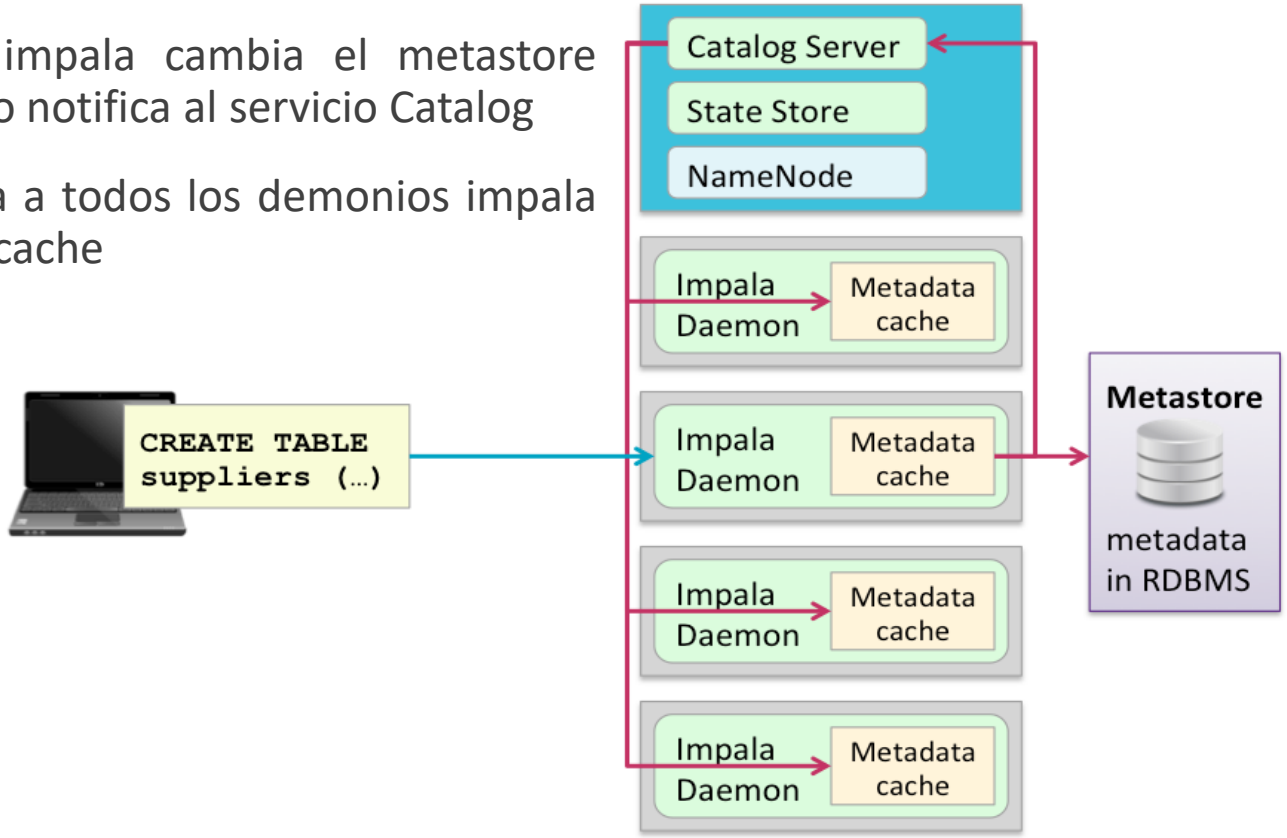


Bosonit

Impala: Cacheo de metadatos

Cuando un demonio impala cambia el metastore (Create table), este se lo notifica al servicio Catalog

El Catalog se lo notifica a todos los demonios impala para que actualicen su cache



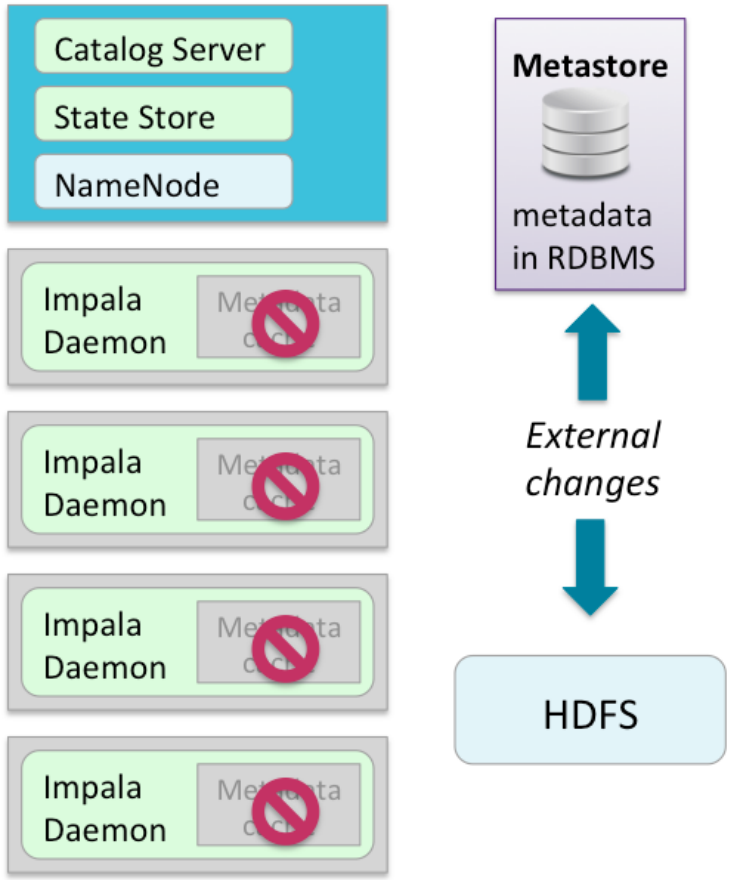
Impala: cambios externos y cacheo del metadata

Los cambios realizados en los metadatos desde fuera de impala no son conocidos para impala

- Cambios vía Hive
- Cambios vía Pig
- Hue Metadata Manager
- Datos añadidos directamente al directorio en HDFS

Por lo tanto, el metadata cacheado que tiene impala no será válido

Es necesario una refresco manual de los metadatos de la cache de impala o su invalidación



Bosonit

Impala: acciones cambios externos

External Metadata Change	Required Action	Effect on Local Caches
New table added	**`INVALIDATE METADATA`** (with no table name)	Marks the entire cache as stale; metadata cache is reloaded as needed
Table schema modified *or* New data added to a table	**`REFRESH <table>`**	Reloads the metadata for one table immediately. Reloads HDFS block locations for new data files only
Data in a table extensively altered, such as by HDFS balancing	**`INVALIDATE METADATA <table>`**	Marks the metadata for a single table as stale. When the metadata is needed, all HDFS block locations are retrieved

Bosonit

Impala: tolerancia a fallos

Sabemos que Hive dispone de tolerancia a fallos otorgada por su forma de procesamiento basada en Map Reduce

- Si un nodo falla otro se hace cargo

Impala no posee tolerancia a fallos

- Si un nodo falla durante la ejecución, toda la ejecución falla
- Las acciones a llevar a cabo son volver a lanzar la query

Bosonit

Impala: optimización

En Impala es posible optimizar la ejecución de las queries simplemente informando de las características de las tablas involucradas en dicha query.

Esto se hace mediante el comando COMPUTE STATS

Es recomendable ejecutarlo

- Justo después de cargar las tablas
- Cuando haya una modificación en una tabla del 20% o más

```
COMPUTE STATS orders;
COMPUTE STATS order_details;
SELECT COUNT(o.order_id)
  FROM orders o
  JOIN order_details d
  ON (o.order_id = d.order_id)
  WHERE YEAR(o.order_date) = 2008;
```

Bosonit

Impala: optimización

Esta información se puede mostrar al usuario con

- SHOW TABLE STATS
- SHOW COLUMN STATS

```
SHOW TABLE STATS orders;
```

#Rows	#Files	Size	Bytes cached	Format
1662951	4	60.26MB	NOT CACHED	TEXT

```
SHOW COLUMN STATS orders;
```

Column	Type	#Distinct Values	#Nulls	Max Size	Avg Size
order_id	INT	2224714	-1	NULL	4
cust_id	INT	257818	-1	NULL	4
order_date	INT	1570457	-1	NULL	16

Bosonit

Sentencias

- $ impala-Shell
- > CREATE DATABASE IF NOT EXISTS ejemplo;
- >SHOW DATABASES;
- USE ejemplo;
- CREATE TABLE IF NOT EXISTS ejemplo.estudiante (name STRING, age INT, contact INT);
- insert into estudiante (name,age,contact)VALUES ('Ramesh', 32, 941587841);
- Select * from estudiante;

Funciona al igual que hive, con sentencias SQL.

Bosonit

Impala: Aclaración

Impala: Consultas ligeras para buscar velocidad, tener en cuenta que si falla tenemos que relanzar la query (diapositiva 18).

Hive: Mejor para trabajos pesados de ETL, buscamos la robustez no tanto la velocidad, y tenemos tolerancia a fallos.

¿Cómo vais viendo el curso?