
plain concepts

ABOUT US

XAMARIN PREMIUM CONSULTING PARTNER

13 MICROSOFT MOST VALUABLE PROFESSIONAL

Xbox PREMIER DEVELOPER PARTNER LIVE Apps

ALM PARTNER OF THE YEAR FOR 7 CONSECUTIVE YEARS

AGILE ALLIANCE CORPORATE MEMEBER

MICROSOFT GOLD CLOUD PLATFORM

MICROSOFT GOLD APP DEVELOPMENT

MICROSOFT GOLD LIFECYCLE MANAGEMENT APPLICAITON

MICROSOFT SILVER APP INTEGRATION

MICROSOFT SILVER COLLABORATION AND CONTENT

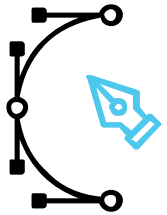
WINDOWS 8 APPLICATIONS PARTNER OF THE YEAR

MICROSOFT BEST CLOUD APPLICATIONS 2016

plain concepts

OUR SERVICES

UI/UX Design



Web & App development

Demos & Whitepapers

Marketing Campaigns

Custom CMS

LEARNING PATH

Sesiones On-Line

25 Mayo - Procesado de Streams sobre Hadoop y Azure Stream Analytics. https://goo.gl/eqx1v0	1 Junio - Machine Learning sobre Hadoop y Azure ML. https://goo.gl/OoGJuA	8 Junio - Visualización en Hadoop IaaS y Power BI. https://goo.gl/ucsnQU

BIG DATA

IMPLEMENTACION DE HADOOP EN AZURE

Francisco Martínez

Data Engineer at Plain Concepts
fmartinez@plainconcepts.com
@pacommiranda

SQL SOBRE HADOOP: HIVE

- Motor de ejecución: MR y Tez
- Creación de esquema de datos
- Consultas utilizando HiveQL
- Optimización: particionado, compresión, vectorización

HIVE - ¿QUE ES?

- Subproyecto de Apache Hadoop para construir un Data Warehouse sobre el clúster
- Estructura los datos mediante conceptos clásicos de bases de datos
  - Tablas, particiones, filas, columnas
- Usando HiveQL, un lenguaje ANSI SQL
- Schema on Read
- Se encarga de ejecutar los jobs de MapReduce (o Tez) de forma automática
- Muchísimo mas sencillo que utilizar MapReduce (o Tez) directamente

HIVE - ¿QUE NO ES?

- Un RDBMS (Relational DataBase Management System)
  - Hay una base de datos donde almacena metadatos, pero los datos se almacenan en ficheros, como hasta ahora
- Diseñado para sistemas OLTP (Online Transaction Processing)
  - No hay consultas en tiempo real (esto es una verdad a medias)
  - No hay actualización de filas (esto también es una verdad a medias)

CREACION DE UN ESQUEMA DE DATOS

- Montar un DW con Hive es sencillo
- Definimos
  - Campos
  - Formato de almacenamiento
  - Localización de los datos
  - Otros (particionado, compresión, etc)

ESQUEMA DE DATOS

CREATE TABLE / ALTER TABLE

EXTERNAL / INTERNAL

INSERT INTO / INSERT OVERWRITE / LOAD DATA LOCAL INPATH

SHOW CREATE TABLE / DESCRIBE

EXPLAIN ☺

plain concepts

HIVE DATAWAREHOUSE

plain concepts

METASTORE

HIVEQL - ¿QUE ES?

- Hive Query Language
- Subconjunto de ANSI SQL
- Algunas limitaciones en comparaciones de igualdad y joins
- No permite updates (hasta la version 0.14)
  - Pero si INSERT/INSERT OVERWRITE
- Nos permite escribir queries SQL y que Hive las traduzca a un job de MapReduce (o Tez)

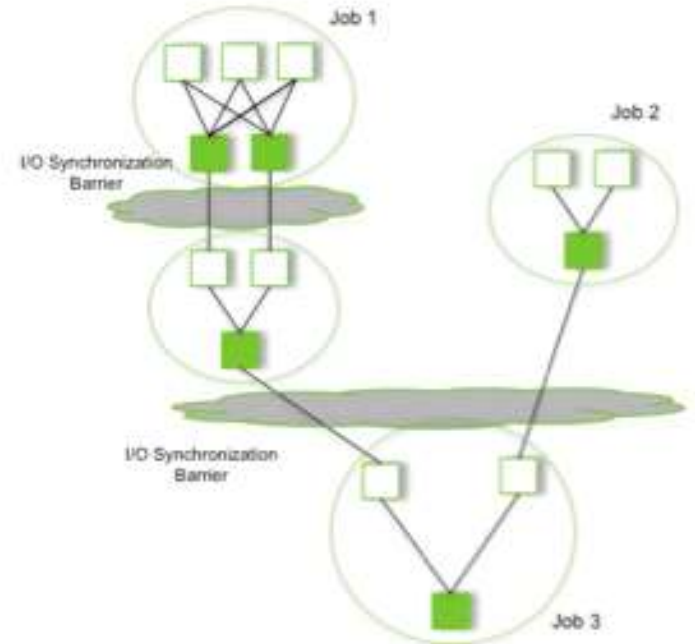
plain concepts

HIVEQL

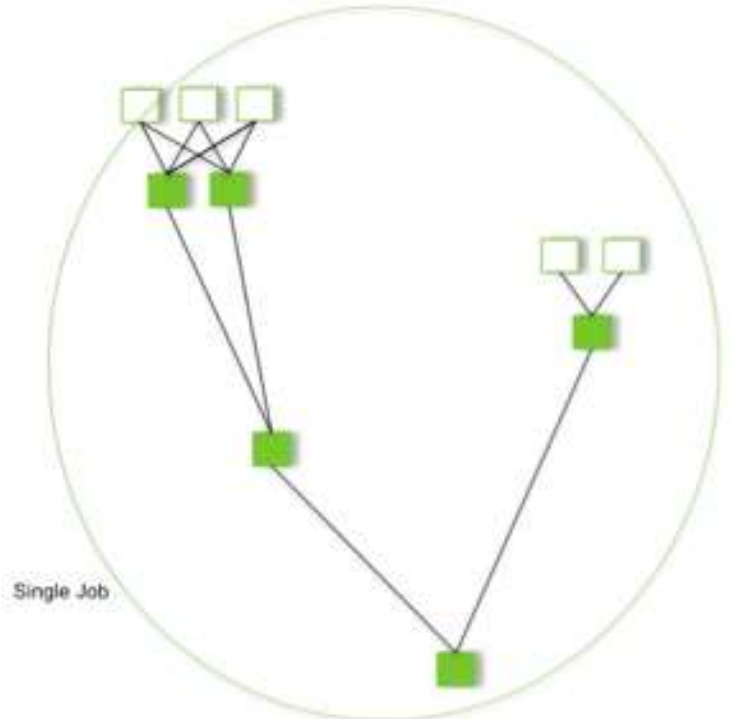
TEZ

- Proyecto Apache para tener un sistema de computación distribuida de proposito general
- Generaliza el paradigma de MapReduce expresando el flujo como un DAG (Directed Acyclic Graph)
  - Elimina procesos de IO, replicaciones, lanzamiento de jobs innecesarios...
- Reemplaza la parte de procesamiento de datos de MapReduce
- Ampliamente personalizable
- Construido sobre YARN como sistema de gestión de recursos

TEZ



Pig/Hive - MR



Pig/Hive - Tez

plain concepts

MR, TEZ E IMPALA

PARTICIONES EN HIVE

- PARTITIONED BY (BirthYear INT, BirthState STRING)
- Particiones físicas
- Optimización de consultas
- Optimización de escrituras
- Necesario especificar la particion en INSERT

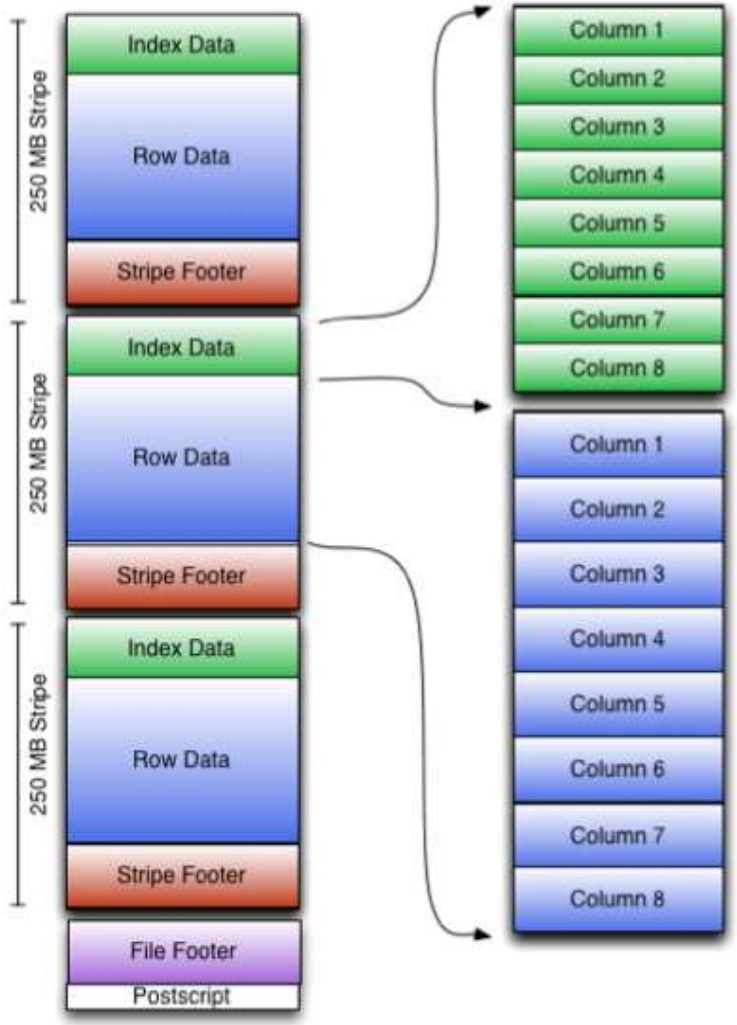
plain concepts

PARTICIONADO

ORC

- Optimized Row Columnar File Format
- Mejor rendimiento al leer, escribir y procesar
- Menor espacio de almacenamiento
- Almacena álgunos valores utiles en el footer
  - COUNT, SUM, MAX, MIN
- Un único fichero de output
- Indexado intra-fichero

ORC


250 MB Stripe
Index Data
Row Data
Stripe Footer
250 MB Stripe
Index Data
Row Data
Stripe Footer
250 MB Stripe
Index Data
Row Data
Stripe Footer
File Footer
Postscript
Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8
Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8

plain concepts

ORC

VECTORIZACION

- Require datos en format ORC
- set hive.vectorized.execution.enabled = true;
- Aumenta el rendimiento procesando bloques de 1024 filas
  - Dentro del bloque, cada columna es un vector (array)
  - Las operaciones simples pueden iterar sobre el vector con rapidez

CONFIGURACIONES INTERESANTES

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

set tez.session.client.timeout.secs = -1;

set mapreduce.input.fileinputformat.split.maxsize = 32000000;

set hive.mapred.supports.subdirectories = true;

set mapred.input.dir.recursive = true;

set hive.stats.autogather = true;

set hive.cli.print.header = true;
```

LENGUAJE DE FLUJO DE DATOS: PIG

- Que es Pig
- Creación y ejecución de scripts Pig

PIG - ¿QUE ES?

- Apache Pig es una plataforma para analizar datos consistente en un lenguaje de alto nivel y un interprete de ese lenguaje
- Una de sus caracteristicas principales es que la estructura de las consultas permite un alto nivel de paralelismo, mejorando el rendimiento

PIG LATIN

- Las consultas Pig se escriben en Pig Latin

Procedimental

Extensible

Sencillo de programar

PIG VS SQL

Pig es procedimental	SQL es declarativo
El esquema es opcional	Requiere un esquema
Pensado para trabajos analiticos de tipo scan (sin lecturas o escrituras aleatorias)	Pensado para trabajos OLTP y OLAP
Optimización limitada	Queries faciles de optimizar

PIG VS SQL

```
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';
```

```
insert into ValuableClicksPerDMA
select dma, count(*) from  geoinfo
 join
( select name, ipaddr from users join
    clicks on (users.name = clicks.user)
    where value > 0; )
using ipaddr group by dma;
```

plain concepts

PIG

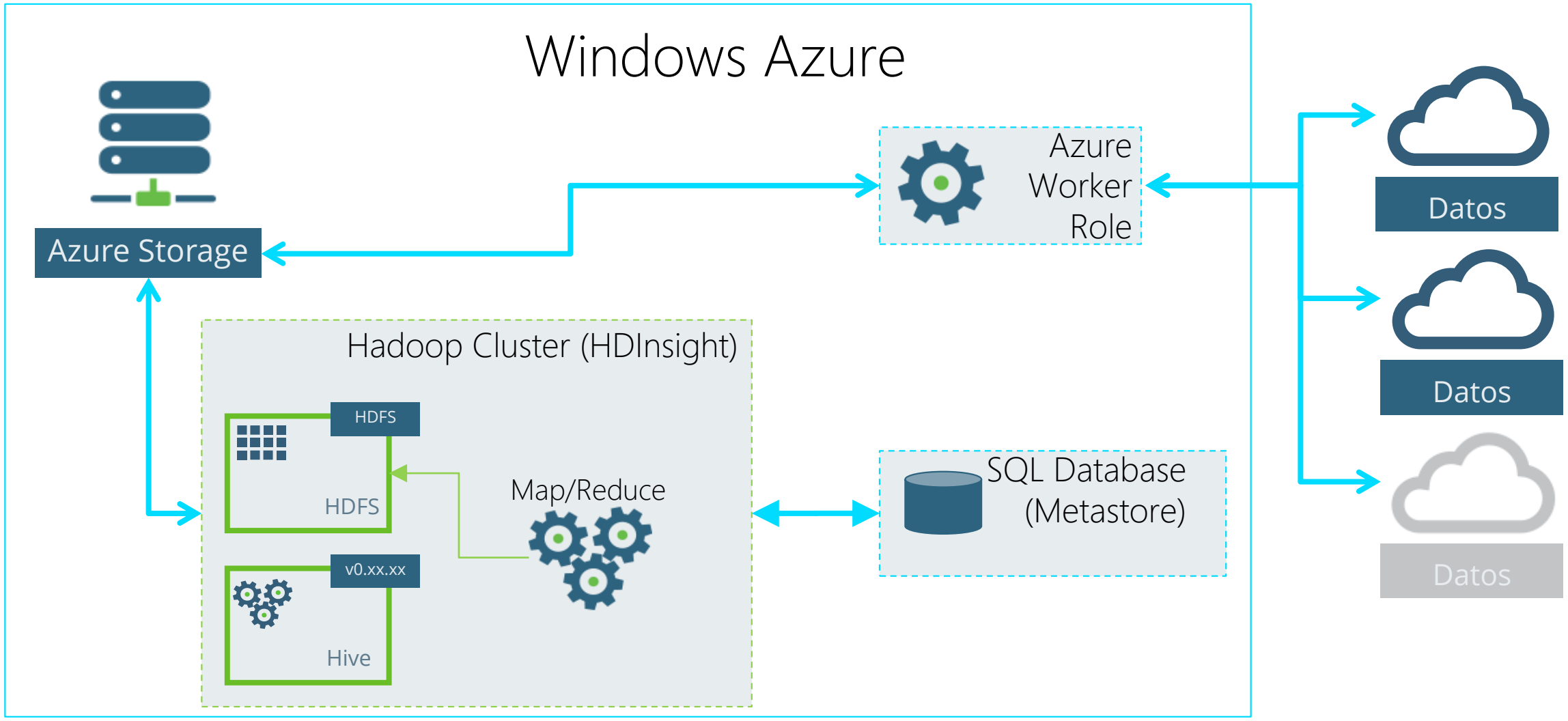
INGESTA DE DATOS EN HADOOP

- Estrategias de carga
- SerDes y formatos
- Utilidades: Sqoop, AzCopy, Flume, etc

ESTRATEGIAS DE CARGA

- Utilizando Ambari para gestionar los ficheros
  - Para pequeñas pruebas de concepto
- Creando un Worker Role para almacenar datos en Blob Storage
- Cargando datos desde un EdgeNode

INGESTA DESDE WORKER ROLE



SERDES Y FORMATOS

- SerDe es la abreviación de Serializador/Deserializador
- Son las interfaces que se encargan de interpretar el formato de los ficheros donde se almacenan los datos del clúster

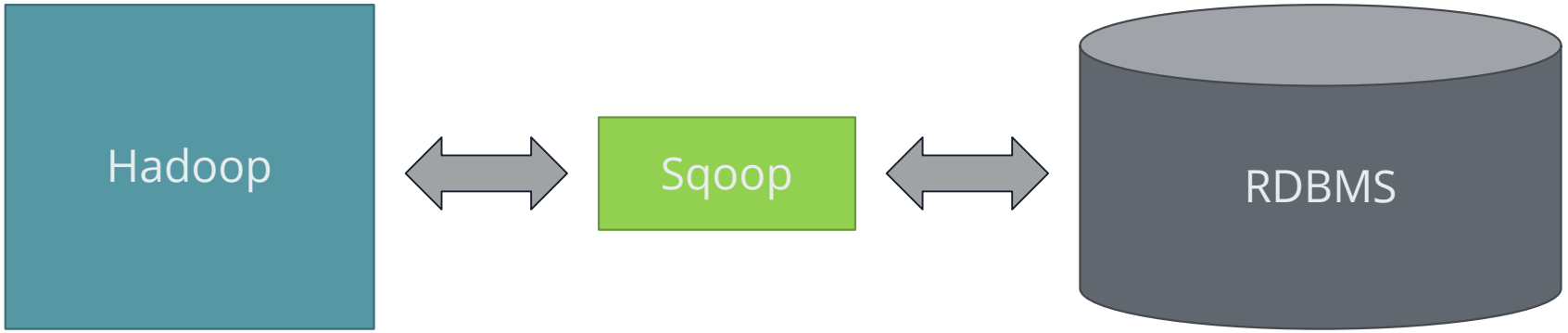
Avro	ORC	RegEx	Thrift
Parquet	CSV	Json	Custom

SERDES Y FORMATOS

- STORED AS ORC
  - ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
  - STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
  - OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat‘
- STORED AS TEXTFILE
  - STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
  - OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'

SQOOP

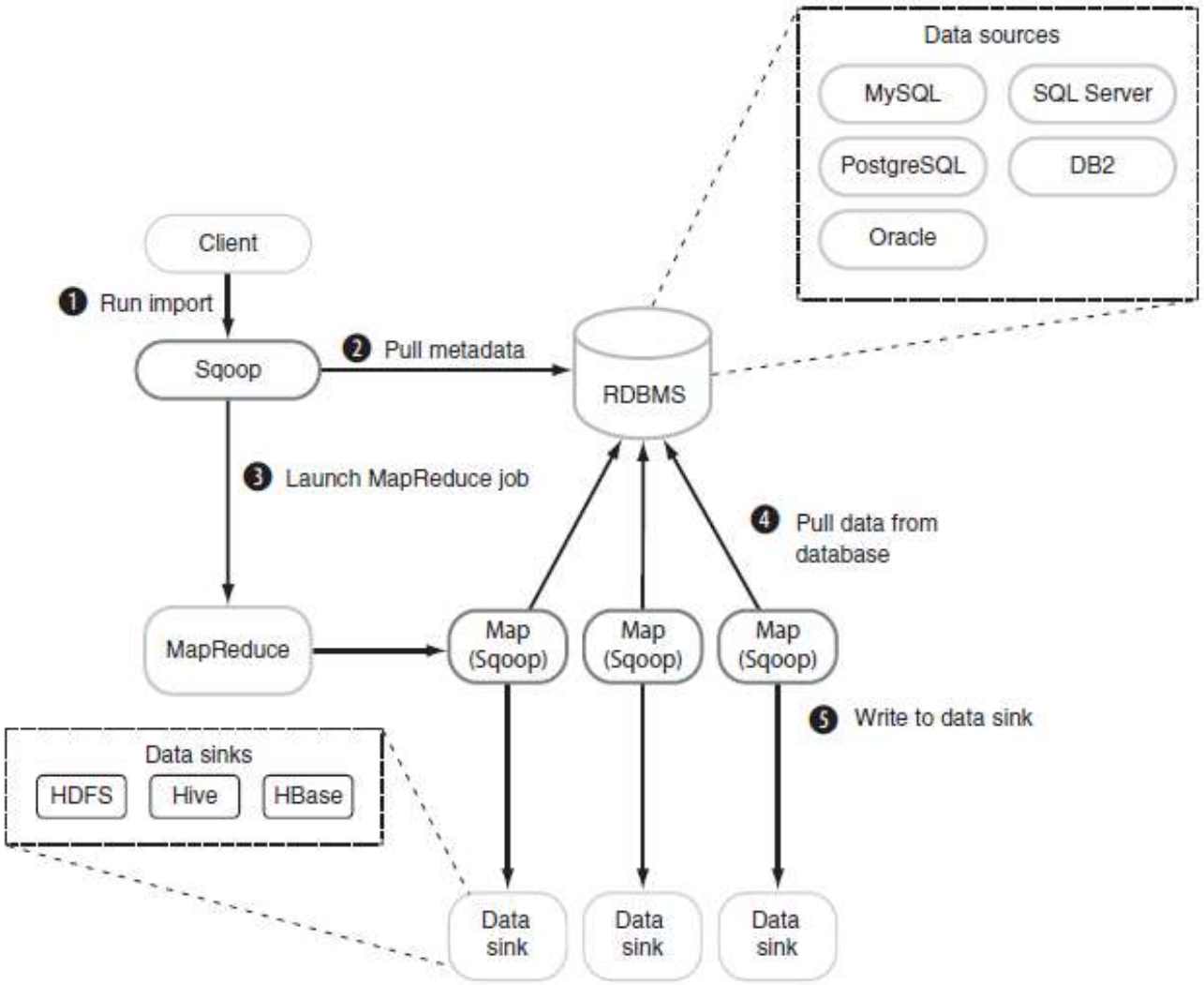
- Utilidad de linea de commandos para transformer y transferir datos entre una base de datos relacional y Hadoop
- Soporta importaciones incrementales
- La funcion Import mueve datos de una base de datos relacional hacia Hadoop
- La funcion Export mueve datos desde Hadoop a una base de datos relacional



SQOOP


El dataset a transferir se divide en bloques pequeños
Sqoop lanza un job que contiene solamente mappers
Cada mapper es responsable de transferir un bloque del dataset

SQOOP


Data sources
MySQL
SQL Server
PostgreSQL
DB2
Oracle
Client
1 Run import
Sqoop
2 Pull metadata
RDBMS
3 Launch MapReduce job
4 Pull data from database
MapReduce
Map (Sqoop)
Map (Sqoop)
Map (Sqoop)
5 Write to data sink
Data sinks
HDFS
Hive
HBase
Data sink
Data sink
Data sink

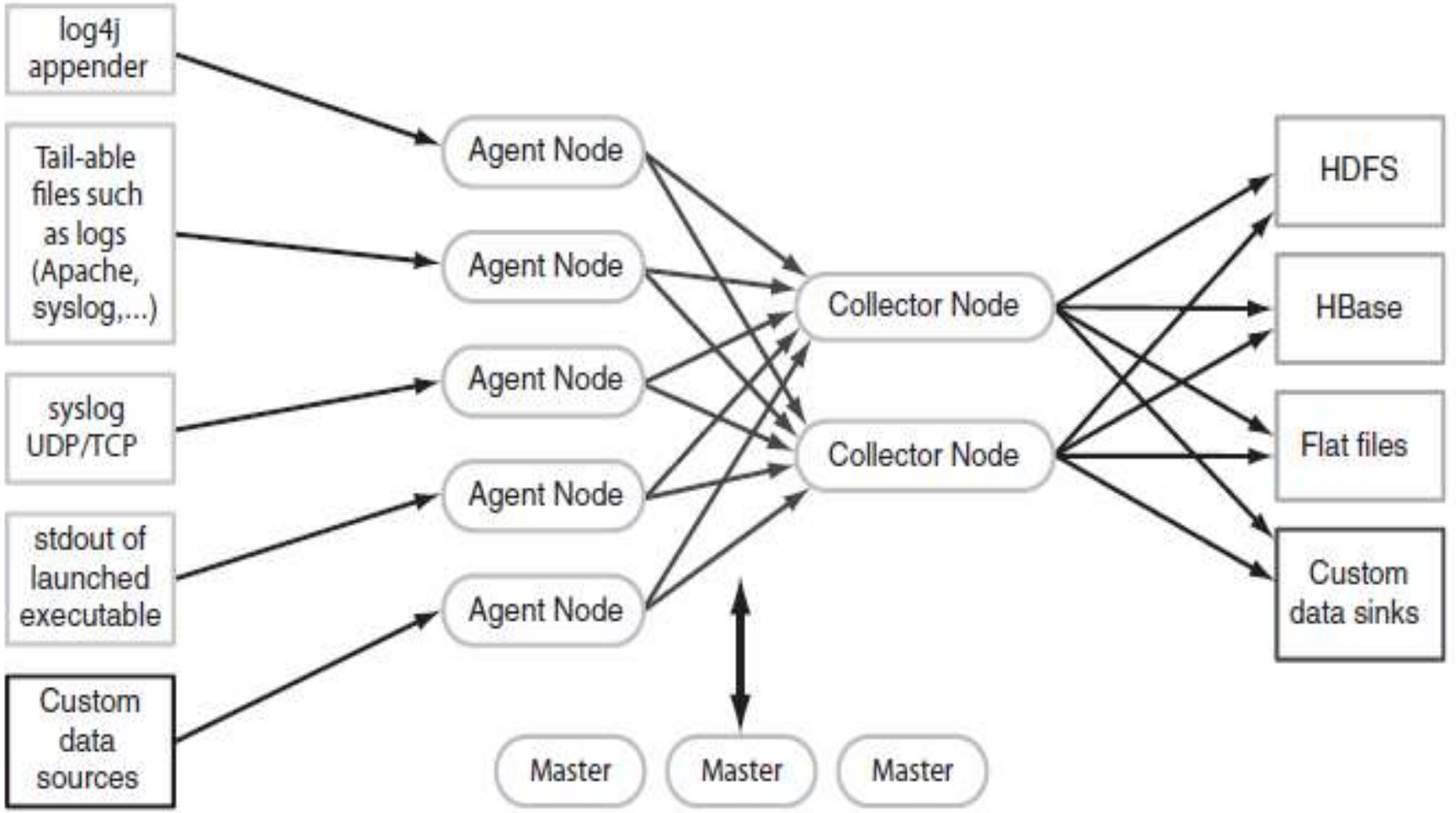
plain concepts

SQOOP

FLUME

- Apache Flume es un servicio distribuido para recolectar, agrupar y mover gran cantidad de datos streaming hacia HDFS
- Los nodos agente se instalan en las maquinas que generan los datos
- Se encargan de enviar la informacion a los collector nodes que la agregan y la envian al storage

FLUME


log4j appender
Tail-able files such as logs (Apache, syslog,...)
syslog UDP/TCP
stdout of launched executable
Custom data sources
Agent Node
Agent Node
Agent Node
Agent Node
Agent Node
Collector Node
Collector Node
HDFS
HBase
Flat files
Custom data sinks
Master
Master
Master

OLAP SOBRE HADOOP: KYLIN

- Construcción de cubos
- Lenguaje de consultas
- Conectividad con herramientas de cliente

¿QUE ES KYLIN?

- Kylin = Motor de análisis distribuido open source
- Proporciona:
  - Capacidad de análisis en grandes datasets
  - ANSI SQL query interface
  - Análisis multidimensional
- Proyecto de primer nivel de Apache
- Originario de eBay

OBJETIVOS INICIALES

- Baja latencia (<1sec) con billones de registros
- Uso de estándar ANSI SQL
- Ofrecer funcionalidades OLAP avanzadas
- Integración con herramientas BI
- Soporte para alta cardinalidad en dimensiones
- Alta concurrencia
- Arquitectura distribuida para proceso de grandes volúmenes de datos

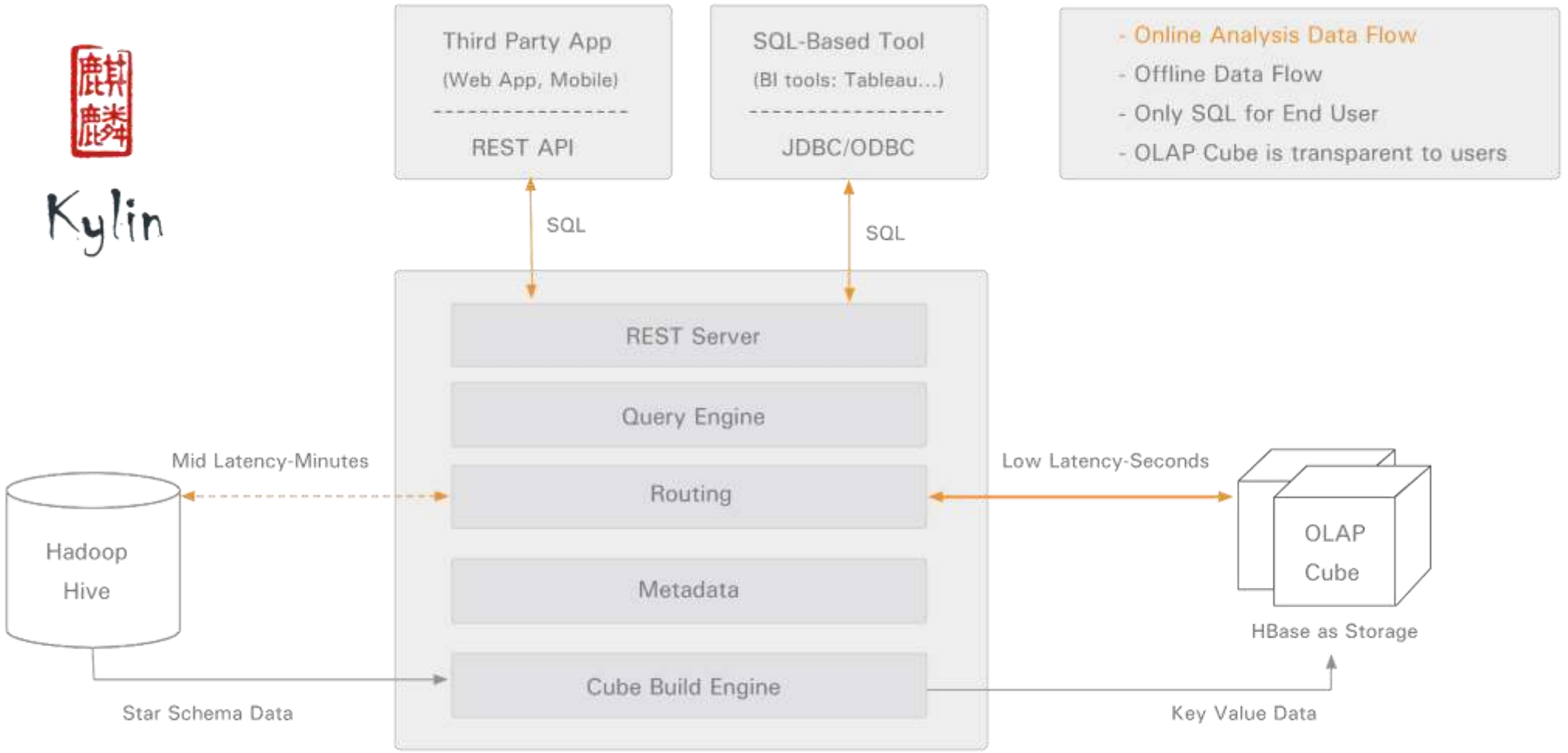
ESTRATEGIA

- Usar Calcite como interprete SQL
  - SQL
  - Optimizador CBO (Cost-Based Optimizer)
  - Enlaza Kylin con Apache Drill y con versiones futuras de Hive
- Construir los cubos independientes del origen, ahora Hive y en un futuro Spark
- Las agregaciones generadas se almacenan en una única maquina

RETOS TECNICOS

- Gran volumen de datos
- Joins de grandes tablas
- Análisis de diferentes niveles de agregación
- Map Reduce jobs

ARQUITECTURA



ECOSISTEMA

- Hive
  - Fuente de datos, pre join star schema en proceso del cubo
- MapReduce
  - Agrega las medidas en el proceso del cubo
- HDFS
  - Almacena los archivos temporales en el proceso
- HBase
  - Almacena los cubos y es la Fuente de datos de las consultas
- Calcite
  - SQL parsing, generación y optimización de código

CARACTERISTICAS DESTACADAS

ANSI SQL Interface	Integración con herramientas de BI	Acceso a datos en Hadoop con latencias < 1 segundo	MOLAP Cube
Soporta origenes >10 billones de filas	Soporta compresión	Proceso incremental de cubos	Gestión y monitorización de jobs de carga
Interfaz web	Seguridad a nivel de Proyecto y cubo	Soporta LDAP	REST API

plain concepts

KYLIN

HADOOP IN-MEMORY: SPARK

- Spark sobre YARN
- Análisis exploratorio en tiempo real

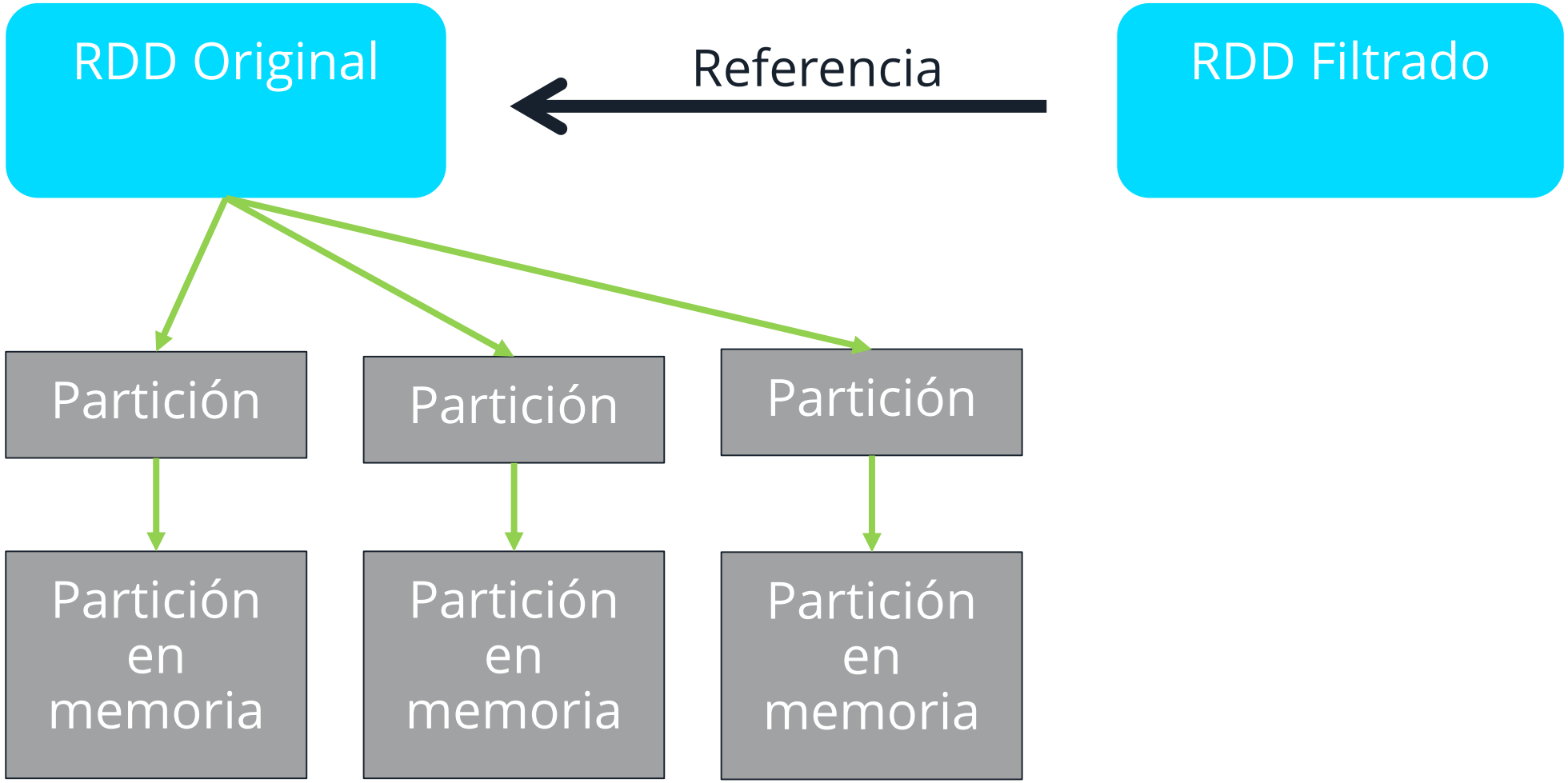
¿QUE ES SPARK?

- Spark es un motor de computación de propósito general que soporta operaciones en memoria
- Con Tez, MR y demás nos basamos en un DAG (Grafo Acíclico Dirigido), trabajando de storage a storage
- Esto es ineficiente en casos en los que necesitamos reutilizar un conjunto de datos (working set)
  - Algoritmos iterativos (machine learning)
  - Análisis exploratorio en tiempo real

¿QUE ES SPARK?

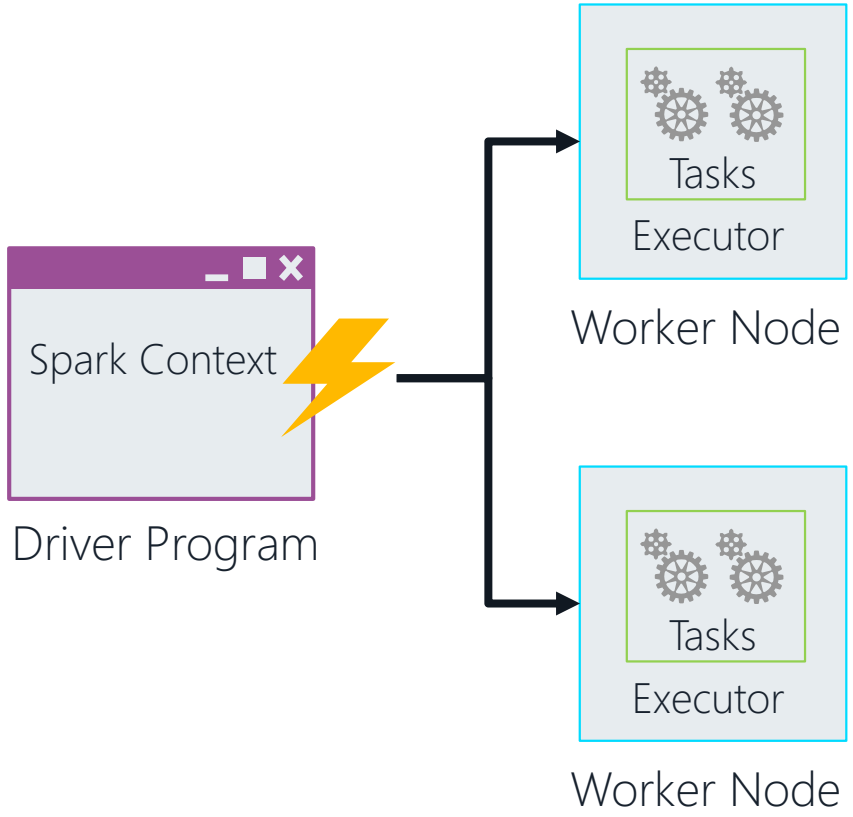
- Spark trabaja mediante operaciones sobre datasets distribuidos
- Dataset distribuido (RDD, Resilient Distributed Datasets)
  - Colección de objetos repartidos en un clúster, bien en memoria o en disco
  - Construidos mediante transformaciones paralelas
  - Reconstruidos de forma automática si hay un fallo
- Operaciones
  - Transformaciones (map, filter, group by...)
  - Acciones (count, save...)

RDD


RDD Original
Referencia
RDD Filtrado
Partición
Partición
Partición
Partición en memoria
Partición en memoria
Partición en memoria

ARQUITECTURA SPARK

- La arquitectura de procesado distribuida se compone de
  - Un Driver Program
  - Uno o mas Worker Nodes
- El Driver Program utiliza un context de Spark para conectarse al cluster…
- …y utiliza los Worker Nodes para realizar operaciones sobre los RDDs



OPERACIONES

- Transformaciones
  - Crean un nuevo RDD al transformar uno existente
- Acciones
  - Devuelven resultados al Driver Program o a un fichero de salida
- Spark utiliza evaluación perezosa
  - Nada se ejecuta hasta llegar a una acción
  - Los RDDs se recomputan en cada acción

OPERACIONES

- La mayoria de operaciones consisten en pasar una función a una transormación o una acción
- Las funciones pueden ser
  - Declaradas de forma explicita
  - Pasadas inline
    - Python usa la keyword lambda
    - Scala usa la sintaxis =>
    - Java usa function classes o lambdas (Java 8)

```
RDD.filter(function)
```

```
def containsMSTag(txt):
    return "#ms" in txt

msTwts = txtRDD.filter(containsMSTag)
```

```
#Python
msTwts = txtRDD.filter(lambda txt: "#ms" in txt)
```

```
//Scala
msTwts = txtRDD.filter(txt => txt.contains("#ms")
```

TRANSFORMACIONES COMUNES

- filter: Crea un RDD filtrado
- flatMap: Aplica una function a cada elemento, retornando multiples elementos a un nuevo RDD
- map: Aplica una function a cada element, retornan un elemento a un nuevo RDD
- reduceByKey: agrega valores por cada clave en un RDD clave-valor

```
txt = sc.parallelize(["the owl and the
pussycat","went to sea"])
```

{["the owl and the pussycat"], ["went to sea"]}

```
owlTxt = txt.filter(lambda t: "owl" in t)
```

{["the owl and the pussycat"]}

```
words = owlTxt.flatMap(lambda t: t.split(" "))
```

{["the"], ["owl"], ["and"], ["the"], ["pussycat"]}

```
kv = words.map(lambda key: (key, 1))
```

{["the",1], ["owl",1], ["and",1], ["the",1], ["pussycat",1]}

```
counts = kv.reduceByKey(lambda a, b: a + b)
```

{["the",2], ["owl",1], ["and",1], ["pussycat",1]}

ACCIONES COMUNES

- reduce: Agrega los elementos de un RDD utilizando una función con dos argumentos
- count: Devuelve el numero de elementos del RDD
- first: Devuelve el primer element del RDD
- collect: Devuelve el RDD como un array
- saveAsTextFile: Almacena el RDD como un fichero de texto en el path proporcionado

```
nums = sc.parallelize([1, 2, 3, 4])
```

{[1], [2], [3], [4]}

```
nums.reduce(lambda x, y: x + y)
```

9

```
nums.count()
```

4

```
nums.first()
```

1

```
nums.collect()
```

[1, 2, 3, 4]

```
nums.saveAsTextFile("/results")
```

/results/part-00000

plain concepts

SPARK

¿PREGUNTAS?

GRACIAS

plain concepts

Barcelona

Bilbao

Madrid

Sevilla

Dubai

London

Seattle

plain concepts