



BIG DATA: Diseño y Arquitectura de Soluciones con Hadoop, Spark y R

Antonio Soto
CEO
asoto@solidq.com



Organización

- Objetivos
- Prerrequisitos
- Materiales
- Laboratorios
- Logística

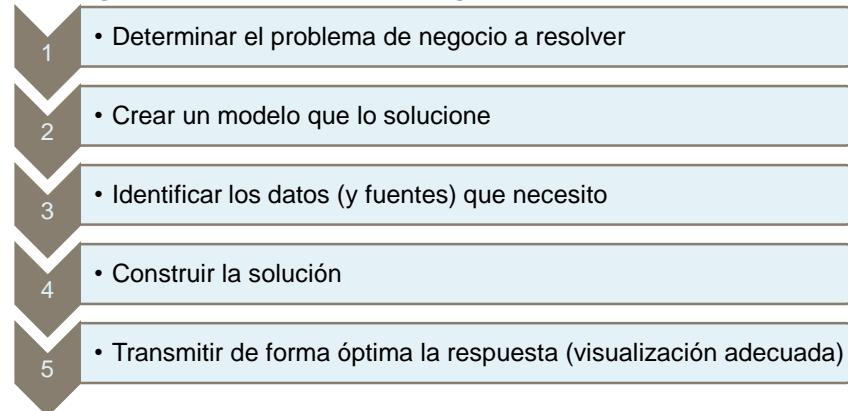
Objetivos

- Conocer las diferentes Arquitecturas posibles para definir una Arquitectura Big Data.
- Diferenciar los distintos tipos de requisitos de análisis de información.
- Introducirse en la Analítica Avanzada a través de Spark y R.

3

Objetivos

- Responder determinadas preguntas de negocio implica los siguientes pasos:



4

Prerrequisitos

- Conocimientos de Bases de Datos en general
- Análisis de Información
- Conocimientos básicos de Administración de Sistemas

5

Materiales

- Material teórico-práctico en PDF
 - Presentaciones PowerPoint
 - Laboratorios

6

Laboratorios

- Se realizarán de forma individual
- Son guías no detalladas que permiten llegar a las soluciones deseadas si has entendido las demostraciones
- Intenta revisarlas hasta al final, entender el objetivo y hacerlas sin consultar

7

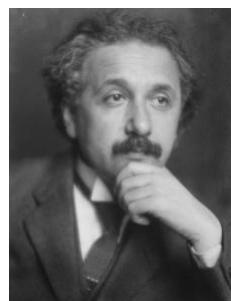
Agenda

- **Introducción a Hadoop**
- **Fundamentos de Almacenamiento**
- **Configuración de Servicios Hadoop**
 - Comprendiendo del Framework de Hadoop
 - Ficheros de Configuración
 - YARN
- **Uso de Ambari para Monitorización y Administración**
- **Fundamentos de Hive**
 - Trabajos Hive
 - Estrategias de Join
- **Arquitectura de Apache Spark**
 - Casos de Uso
 - Modos de Despliegue
 - Ejecución Física
- **R-Server en Spark**
 - Distribuciones de R
 - Modelos de ejecución

8



Introducción a Big Data



“La información NO es conocimiento”

Para que **sea conocimiento** tiene que estar **organizada, procesada y dirigida** a las personas adecuadas

Agenda

- **Introducción a Big Data**
- Hadoop / HDInsight
- Ecosistema Hadoop
- Datos estructurados y no estructurados
- Casos de Estudio

11

Definiciones de Big Data

- Un conjunto de tecnologías relacionadas y no relacionadas para analítica a gran escala
- Gran volumen, alta velocidad y gran variedad de información que demanda un procesado poco costoso para obtener conocimiento y tomar decisiones.

12

Las V's de Big Data

- **Volumen:** Terabytes, Petabytes, Exabytes
- **Velocidad:** hora, segundos, milisegundos
- **Variedad:** 5 formatos, 10 formatos, 20+ formatos
- **Variabilidad:** formatos cambian en el tiempo

No todas las V's tienen que estar presentes

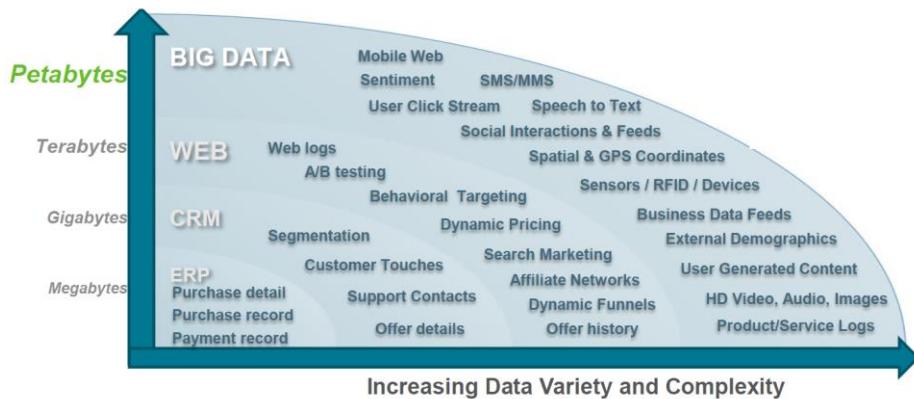
13

ROI probado

- Google AdWords: Predicción de click through rates (CTR)
- Netflix: 75% del streaming de video viene de recomendaciones
- Amazon: 35% de las ventas de producto vienen de recomendaciones de producto

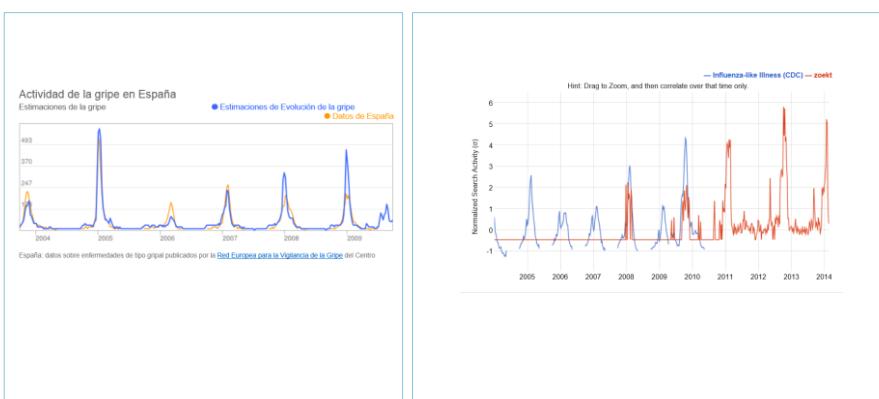
14

Nada nuevo, excepto las V's



15

Correlación de Búsquedas

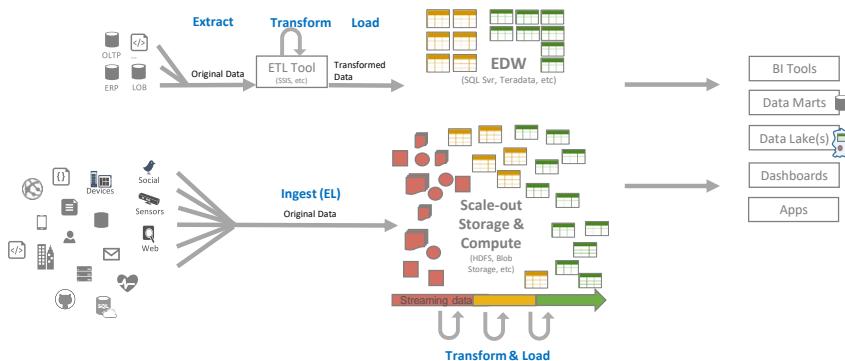


Big Data ≠ BI Tradicional con más datos

- Big Data está redefiniendo los procesos de gestión de datos maestros, calidad de datos y gestión del ciclo de vida de la información
- Big Data NO reemplaza EDW y OLAP, suplementa esas inversiones
- El ecosistema Big Data incluye una gran variedad de tecnologías analíticas
- Bases de datos columnares, JSON y almacenes de ficheros no estructurados

17

Enfoque evolutivo



18

Cambios en patrones DW

El almacenamiento Big Data (hoy en día Data Lake) se caracteriza por tres atributos clave:

- **Recoge todo**
- **Explotalo desde cualquier sitio**
- **Acceso Flexible**

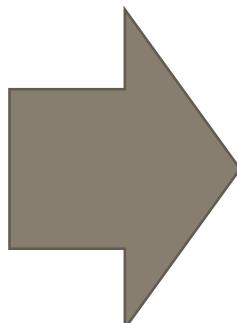
19

Cambios en patrones DW

- Cambiar de Esquema primero a Esquema más tarde

1. Llegan los datos
2. Se deriva el esquema
3. Limpieza de Datos
4. Transformación
5. Carga en EDW
6. Análisis

Valor de
los datos
LENTO



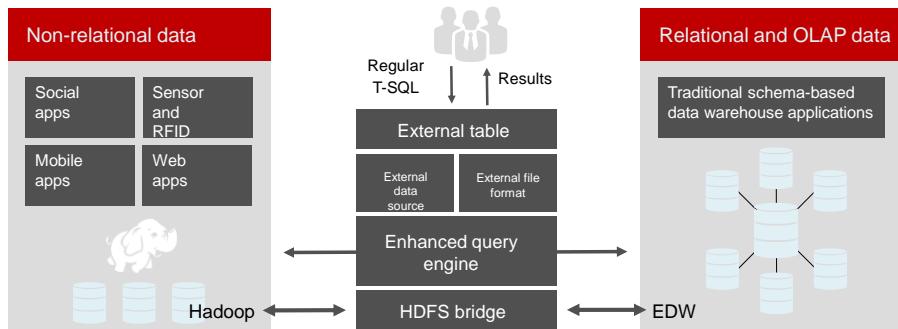
1. Llegan los datos
2. Se cargan en Hadoop
3. Análisis
4. Subconjuntos cargados en EDW

Rápido valor de
los datos

20

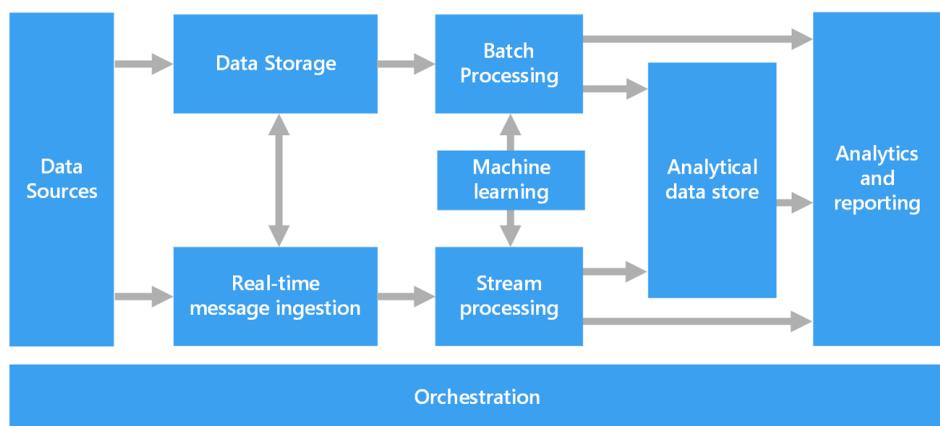
Cambios en Patrones

Basically adding a “bridge” to Big Data



21

Componentes de una Arquitectura Big Data



22

Tipos de Carga de Trabajo

- Procesado batch de grandes orígenes de datos
- Procesado de grandes cantidades de datos en tiempo real
- Exploración interactiva de grandes cantidades de datos
- Analítica predictiva y Machine Learning
- Considerar big data cuando:
 - Se almacenan y procesan grandes cantidades de datos demasiado grandes para una base de datos tradicional
 - Transformar datos no estructurados para análisis y Reporting
 - Capturar, procesar y analizar grandes cantidades de datos en streaming en tiempo real o con muy baja latencia

23

Agenda

- Introducción a Big Data
- **Hadoop / HDInsight**
- Ecosistema Hadoop
- Datos estructurados y no estructurados
- Casos de Estudio

24

Hadoop

- Open Source ☺
- Plataforma de almacenamiento **y** procesado para **Big Data**
- Optimizado para manejar
 - Datos de forma masiva utilizando paralelismo
 - Variedad de datos(Estructurado, No estructurado, Menos estructurado)
 - Uso de hardware barato
- No para OLTP / OLAP
- Mover el cómputo hacia el dato



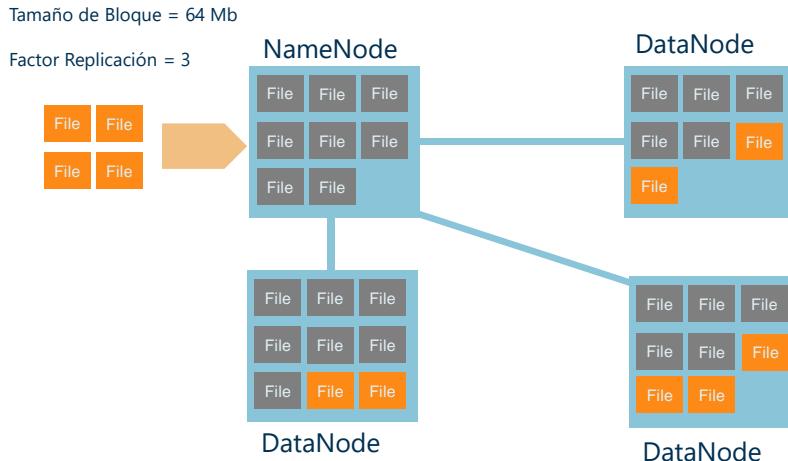
25

Hadoop

- Componentes core de Hadoop: HDFS & MapReduce
- Hadoop Distribution File System
 - Distribuido, tolerante a fallos, redundante, autorecuperable
- Map Reduce
 - Procesamiento distribuido, tolerante a fallos, procesa donde está el dato. Lectura y procesado distribuido.

26

Un cliente escribiendo datos en HDFS



Hadoop

- Escalable
 - Escala linealmente en capacidad de almacenamiento y procesado.
- Tolerante a Fallos
 - Matrimonio entre un Sistema de ficheros distribuido y un framework tolerante a fallos utilizado para leer datos
- Procesamiento distribuido
 - Sigue la estrategia de divide y vencerás.

HDInsight

- HDInsight es la distribución de Microsoft de Apache Hadoop
- On premise. Instalación servidor desarrollo
- En Azure: Despliegue en la nube

29

RDBMS vs Hadoop

Característica	RDBMS	Hadoop
Tamaño de Datos	Gigabytes (Terabytes)	Petabytes (Hexabytes)
Acceso	Interactivo y Batch	Batch – NO Interactivo
Actualizaciones	Leer/ Escribir varias veces	Escribir una vez, leer varias veces
Estructura	Esquema estático	Esquema dinámico
Integridad	Alta (ACID)	Baja
Escalado	No lineal	Lineal
Tiempo de respuesta consultas	Puede ser casi inmediato	Tiene latencia (debido a procesamiento batch)

30

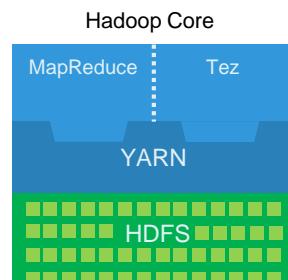
Historia Hadoop

- 2002: **Nutch** open source motor de búsqueda por **Doug Cutting**
- 2003: Google publica un documento sobre **GFS** (Google Distribute File System)
- 2004: **Nutch** Distributed Files System (**NDFS**)
- 2004: Google publica un documento sobre **MapReduce**
- 2005: **MapReduce** se implementa en **NDFS**
- 2006: Doug Cutting se une a Yahoo! & inician Apache Hadoop Subproject
- 2008: Hadoop se convierte en un Proyecto top de Apache
 - El índice de Yahoo's se ejecuta en un cluster de 10.000 nodos
 - Hadoop rompe el record de 1TB en ordenación: 209s en 910 nodos
 - New York Times convierte 4TB de archivos en PDF en 24h en 100 nodos
- 2011: Yahoo crea **HortonWorks**, una compañía dedicada a Hadoop
- 2011: **HortonWorks** y **Microsoft** anuncian un acuerdo
- 2011: Microsoft libera la primera preview de Isotop/**HDInsight**

31

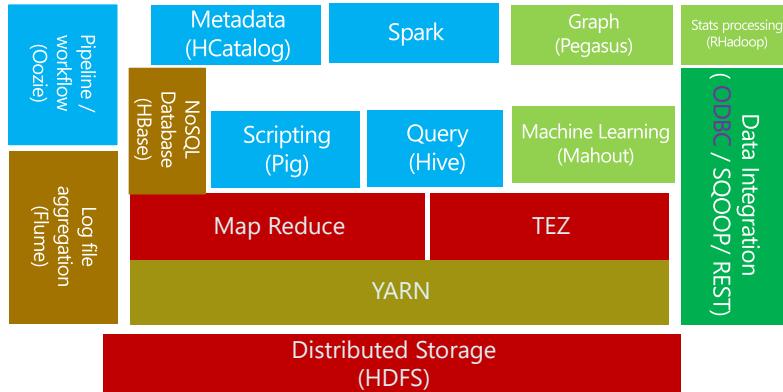
El Zoo de Big Data

- El objetivo de Hadoop es crear un framework unificado para procesar big data
- Tres requisitos principales:
 - Escalabilidad
 - Fiabilidad
 - Eficiencia

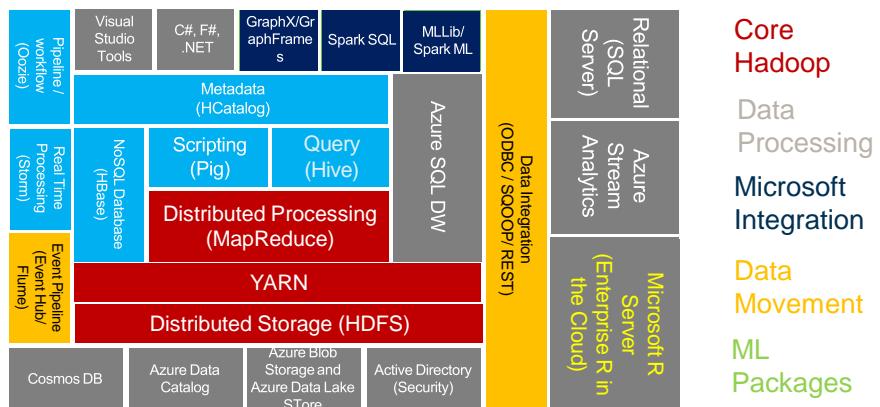


32

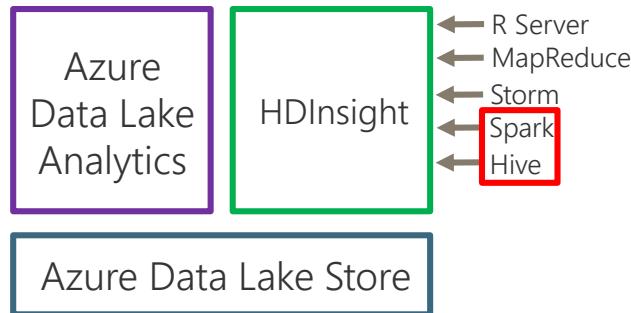
Ecosistema Hadoop



HDInsight



La pila Microsoft simplificada



35

Tipos de cluster HDInsight

- Hadoop
- Hbase
- Spark
- Interactive Query
- R Server
- Storm
- Kafka

36

Hive

- Sistema Data Warehouse para Hadoop
- Facilita las summarizaciones de datos
- Consultas Ad-hoc
- Lenguaje consulta similar SQL: **HiveQL**
- Análisis de grandes conjuntos de datos almacenados en Hadoop
- Por detrás ejecuta
 - Trabajos **MapReduce**
 - **Stinger / Tez**
 - **LLAP** (Long Live and Process)

37

Pig

- Lenguaje scripting de Alto nivel
- Capa de procesamiento de Alto Nivel que se ejecuta en Hadoop
 - Usa ambos **HDFS** y **Map Reduce**
- Facilidad de programación
 - El Usuario se enfoca en semántica en lugar de eficiencia. Map Reduce es como lenguaje de ensamblador
- Extensibilidad

38

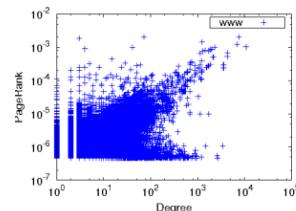
Flume & Sqoop

- Flume
 - Recolectar y mover grandes cantidades de datos
 - Ejecución distribuida
- Sqoop
 - Import y Export: RDBMS \leftrightarrow HDFS, Hive..
 - SQL Server, MySQL, Oracle
 - Ejecución distribuida

39

Mahout & Pegasus

- Mahout
 - Machine learning y data mining a gran escala
 - clusterización, recomendaciones, clasificación, y más.
- Pegasus
 - Page Rank y Graph Mining
 - Network Analysis.
- Por detrás se ejecutan Trabajos **MapReduce**



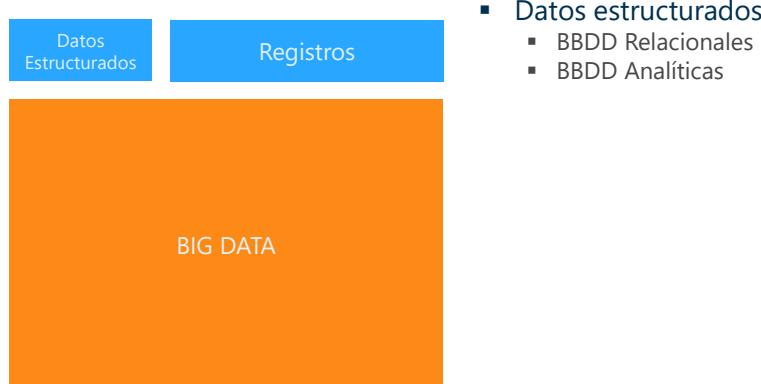
40

Agenda

- Introducción a Big Data
- Hadoop / HDInsight
- Ecosistema Hadoop
- **Datos estructurados y no estructurados**
- Casos de Estudio

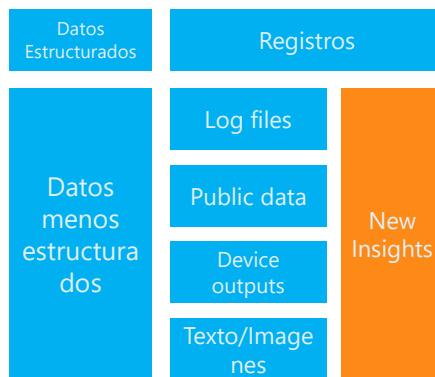
41

Datos no estructurados no se están analizando



42

Datos no estructurados no se están analizando

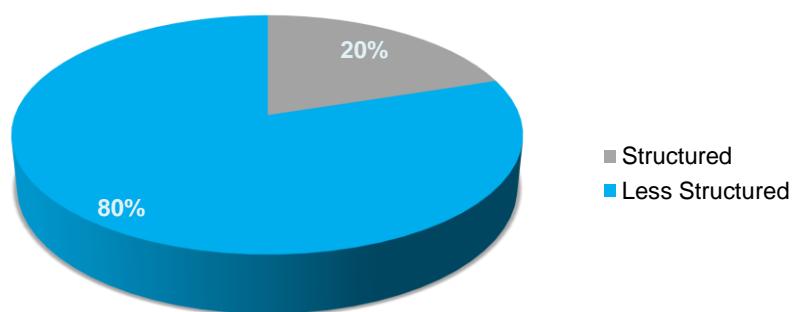


- Datos estructurados
 - BBDD Relacionales
 - BBDD Analíticas
- Datos Menos estructurados
 - Crear ETL para transformar en Relacional
 - Mucho tiempo desarrollo
 - Susceptible cambio estructura
 - Archivado o borrado
 - Acceso caro

43

Datos en las organizaciones

Tipos de datos



44

Ejemplos de datos no estructurados

	12 Tb day	21 Pb Hadoop cluster
	7 Pb Month (search queries info)	
	1 Tb tweets/day	7 Tb data/day
	75 Million scores/day	4 Billion Graph edg/day
FINANCIAL TIMES THE WALL STREET JOURNAL	USA TODAY	Millions of opinions
BBC		

45

Almacena ahora, averigua más tarde

- Hadoop facilita almacenamiento y procesado
- Fácil de desarrollar programas que obtienen conocimiento de datos no estructurados
- Framework para almacenar y procesar subconjunto de los datos a demanda

46

Datos en Tiempo Real

- Utilizar almacenes de datos operacionales en tiempo real (RT ODSs)
- Utilizar DW en Tiempo real
- Implementar CDC
- Presentar datos en tiempo real y datos históricos
- Definir umbrales aceptables y reglas de negocio para todas las entidades del tiempo real

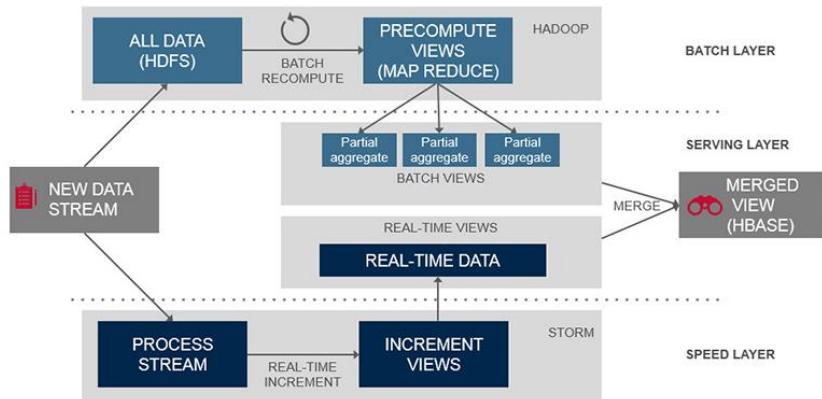
47

Datos en Tiempo Real

- Flujo de datos continuo
- Manejar el stream como si fuese una cola
- Ventanas de tiempo
- Arquitectura de datos Hadoop y Lambda
- Enriquecer datos de streaming con datos de la organización
- Almacenar los datos de stream para construir la historia

48

Arquitectura Lambda



49

Agenda

- Introducción a Big Data
- Hadoop / HDInsight
- Ecosistema Hadoop
- Microsoft BI & Big Data Vision
- Datos estructurados y no estructurados
- **Casos de Estudio**

50

Casos de estudio Hadoop (I)

- Modelado de Riesgos
 - Banca y seguros,
- Análisis de rotación
 - Consultar logs y datos complejos desde multiples orígenes
- Motor de recomendaciones

51

Casos de Estudio Hadoop (II)

- Ad Targeting
 - CTR, placement, auction
- Análisis de transacción en punto de venta
 - Análisis cesta de la compra, mejora de márgenes
- Datos de Redes
 - Predicción de fallos, ratios de transmission, protocolos de transmisión

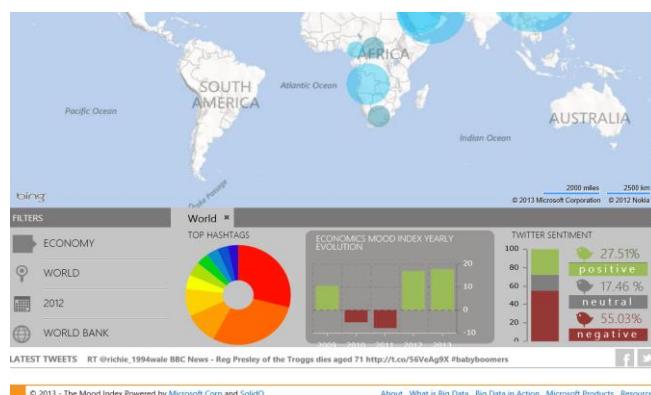
52

Casos de Estudio Hadoop (III)

- Detección de fraude
 - En transacciones
- Calidad búsquedas
 - Resultados relevantes, utilidad de búsquedas
- Data sandbox
 - Almacenado ahora y analizado después
- Análisis de Sentimiento
 - Twitter

53

Caso de Éxito: Mood Index



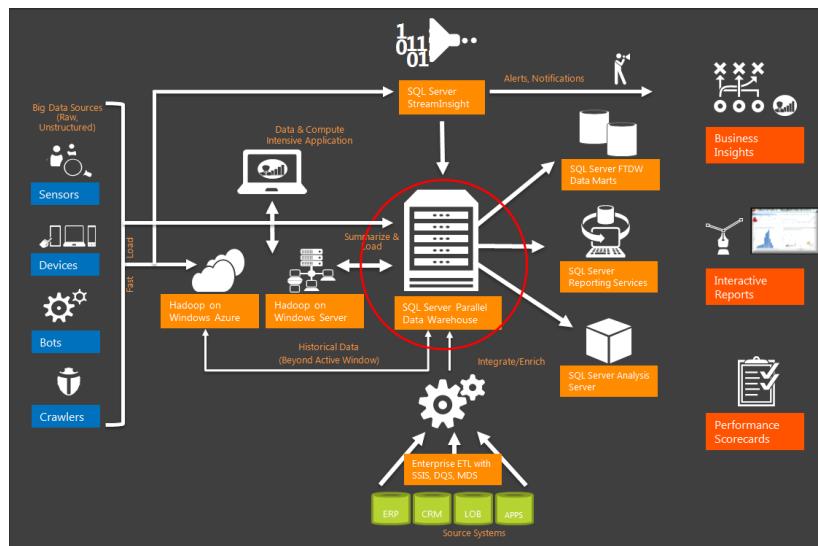
MIDAMO



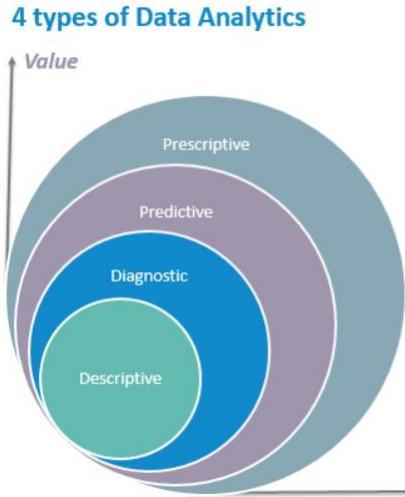
The screenshot shows a dashboard with the following sections:

- Temas:** A list of topics with their respective scores and percentages of moderated posts.
- Configuración:** A section for configuration with a "Aplicar cambios" button.
- Noticias:** A news article titled "Florentino sí que es un galáctico". It includes a thumbnail of Florentino Pérez, a summary, and a comment section.
- Comentarios:** A list of comments from users like "javi86" and "madrid1".
- Normas:** Guidelines for posting.
- Publicar:** A button to publish a comment.

Opciones



Tipos de Análisis



What is the data telling you?

Descriptive: What's happening in my business?

- Comprehensive, accurate and live data
- Effective visualisation

Diagnostic: Why is it happening?

- Ability to drill down to the root-cause
- Ability to isolate all confounding information

Predictive: What's likely to happen?

- Business strategies have remained fairly consistent over time
- Historical patterns being used to predict specific outcomes using algorithms
- Decisions are automated using algorithms and technology

Prescriptive: What do I need to do?

- Recommended actions and strategies based on champion / challenger testing strategy outcomes
- Applying advanced analytical techniques to make specific recommendations

57

Demo



Creando un cluster HDInsight

58

Laboratorio 1



Crear vuestro cluster SPARK:

- Nombre: <A vuestra elección>
- Dejad usuario admin
- Contraseña: Puk02020#!
- Crear cuenta de almacenamiento
 - Mismo Grupo de Recursos
 - Misma región
- Editar tamaño de cluster
 - 2 nodos

59

Fundamentos de Almacenamiento

Agenda

- **Introducción a HDFS**
- Línea de Comandos
- Direccionamiento de bloques y Red
- HDInsight

61

Introducción

- Sistema de Ficheros para almacenar ficheros muy grandes
- Patrón de Escribe una vez, lee muchas veces
- Commodity Hardware
- Gran trasiego de datos
- Self-Healing High-Bandwidth Clustered Storage

62

Introducción

- HDFS no ofrece buen rendimiento para:
 - Accesos de baja latencia
 - Ficheros pequeños (a menos que se agrupen)
 - Múltiples “escritores”
 - Modificaciones arbitrarias de ficheros

63

Bloque HDFS

- Cantidad mínima de datos que puede ser leída o escrita.
- Bloques de Sistema de ficheros tienen habitualmente unos pocos kilobytes, los de disco son habitualmente de 512 bytes.
- El tamaño predeterminado de HDFS son 64 MB.
- Las tareas Map en MapReduce operan habitualmente con un bloque.
- Gran tamaño para optimizar búsquedas.

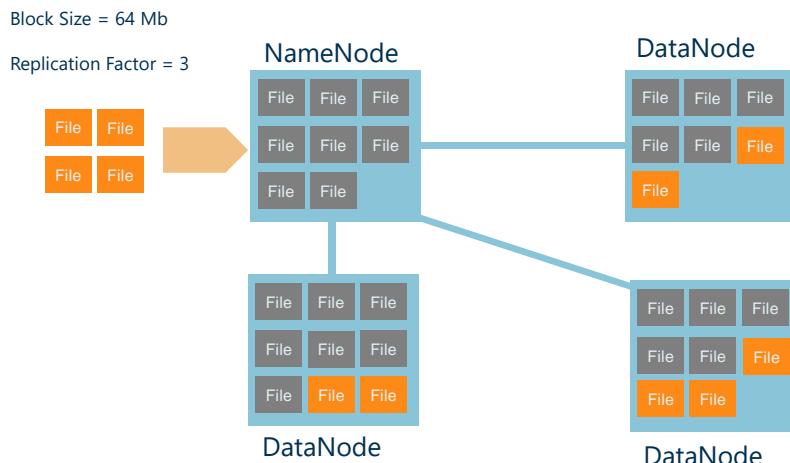
64

Namenodes y Datanodes

- 1 Namenode
 - El maestro
- Multiples Datanodes
 - Los “trabajadores”

65

Un cliente escribiendo datos en HDFS



Namenode

- Solo uno.
- Gestiona el espacio del Sistema de ficheros
- Mantiene el árbol del Sistema de ficheros y los metadatos para todos los ficheros y directorios en el árbol.
- Es posible ejecutar un NameNode secundario

67

DataNodes

- Más de uno
- Almacena y lee bloques.
- Recuperado por Namenode o clientes
- Reportan al Namenode la lista de bloques que están almacenando.

68

Factor Replicación

- No para directorios
- Ficheros y Bloques
- Configuración para todo el cluster
- Se lista con ls

69

Formatos de fichero

- Texto / CSV
- JSON
- Avro
- Sequence – Formato nativo Hadoop
- Almacenamiento Columnar
 - RC
 - ORC
 - Parquet (Cloudera / Impala)

Agenda

- Introducción a HDFS
- **Línea de Comandos**
- Direccionamiento de Bloque y Red

71

Interface Linea de comandos

- HDFS por defecto puerto 8020.
Hdfs//localhost/
- Interface POSIX
hadoop fs -help

72

cmd hadoop fs

-ls	-lsr	-du	-dus
-count	-mv	-cp	-rm
-rmr	-expunge	-put	-copyFromLocal
-moveFromLocal	-get	-getmerge	-cat
-text	-copyToLocal	-moveToLocal	-mkdir
-setrep	-touch	-test	-stat
-tail	-chmod	-chown	-chgrp

73

Permisos de ficheros

- Cómo POSIX
 - (r)ead
 - (w)rite
 - e(x)ecute
- Se pueden aplicar a ficheros o directorios
 - rw-r—r- or drwxr-xr-x
- Primer grupo → Owner
- Segundo grupo → Group
- Tercer grupo → Mode

74

Patrones de Ficheros / Caracteres Globales

*	asterisk	matches zero or more characters
?	question mark	matches a single character
[ab]	character class	matches a single character in the set {a,b}
[^ab]	negated character class	Matches a single character that is not in the set {a,b}
[a-b]	character range	matches single character in the (closed) range [a,b], where a is lexicographically less than or equal to b
[^a-b]	negated character range	matches single character that is not in the (closed) range [a,b], where a is lexicographically less than or equal to b
{a,b}	alternation	matches either expression a or b
\c	escaped character	maches character c when it is a metacharacter

75

cmd: hadoop

- namenode –format
 - Formatea el name node
- Secondarynamenode
 - Habilita un nodo secundario

76

cmd: hadoop

- **dfsadmin**
 - Comando de Administración para la configuración DFS.
- **fsck**
 - Chequeo de Sistema de Ficheros

77

cmd Hadoop distcp

- Copia paralela
- Para copiar gran cantidad de datos hacia o desde Hadoop
- Implementado como MapReduce
 - La copia hecha por los mappers no los reducers.

78

cmd hadoop archive

- Hadoop archive (HAR) utilizando la herramienta de archive, se crea a partir de una colección de pequeños ficheros
- Ficheros pequeños no usan el tamaño de bloque en los datanodes, pero no son óptimos para namenodes
- Ficheros HAR pueden ser entrada de MapReduce
- HAR crea una copia de los originales
- HARs son inmutables

79

Agenda

- Introducción a HDFS
- Línea de Comandos
- **Direccionamiento de Bloque & Red**

80

Topología de Ancho de Banda de Red

- Hadoop representa la red como un árbol.
- Distancia entre 2 nodos es la suma de sus distancias al antecesor más cercano en común.
- Hadoop necesita ayuda para configurar la topología de red
- Sirve bloques de los nodos más cercanos

81

Ubicación de Réplicas

- Fiabilidad
- Ancho de banda escritura
- Ancho de banda lectura
- Ejemplo para factor de replicación 3
 - Primera Replica: mismo nodo (sino aleatorio) como cliente
 - Segunda Replica: rack diferente al primero de forma aleatoria
 - Tercera Replica: mismo rack que el Segundo pero diferente nodo

82

Demo



HDFS

83

Laboratorio 2



Hadoop Distribution File System

84

Agenda

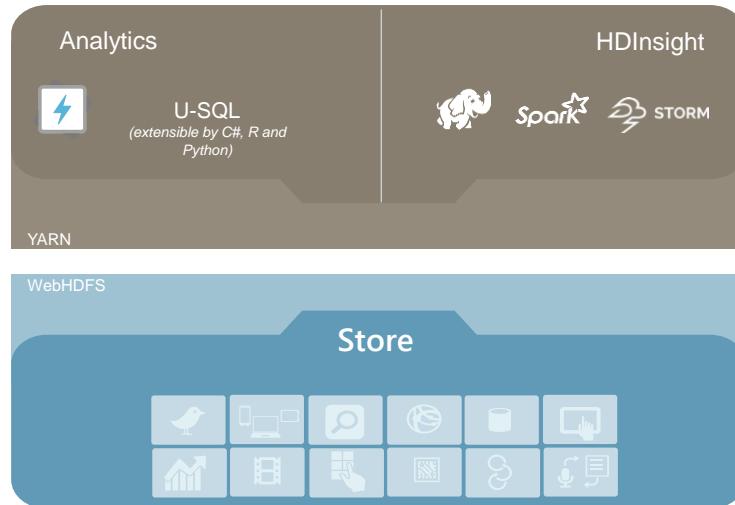
- Introducción a HDFS
- Línea de Comandos
- Direccionamiento de bloques y Red
- **HDInsight**

85

Almacenamiento en HDInsight

- Capa de HDFS sobre:
 - Azure Data Lake
 - Azure Blob Storage

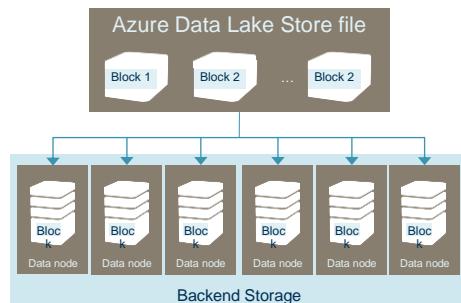
Azure Data Lake



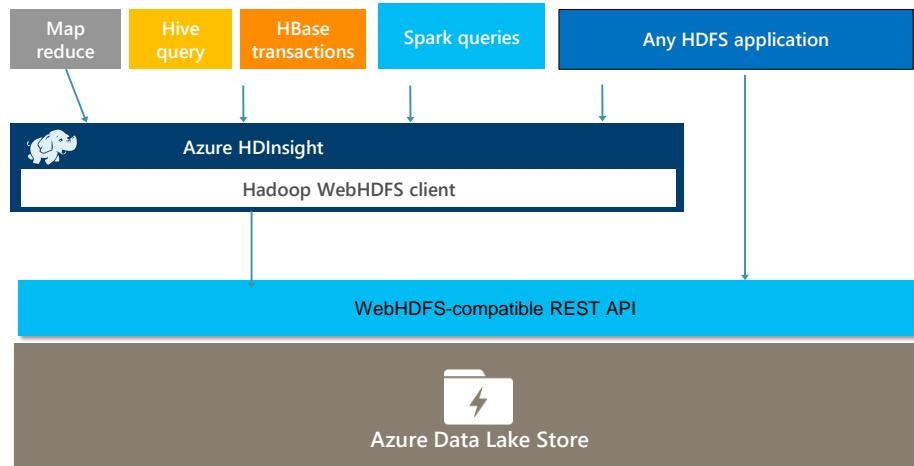
87

Almacenamiento ADL

- Cada fichero se divide en bloques
- Los bloques se distribuyen entre los nodos
- Sin límite tamaño de ficheros
- Recursos ilimitados
- Se almacenan metadatos sin límite
- Paralelismo en lectura
- Azure mantiene tres réplica de cada objeto por región



Compatibilidad HDFS



Data Lake vs Blob Storage

	Azure Data Lake Store	Azure Blob Storage
Purpose	Optimized for Analytics	General purpose bulk storage
Scenarios	Batch, Interactive, Streaming, ML	App backend, backup data, media storage for streaming
Units of Storage	Accounts / Folders / Files	Accounts / Containers / Blobs
Structure	Hierarchical File System	Flat namespace
Supports WebHDFS	Yes	No
Billing	Pay for data stored and for I/O	Pay for data stored and for I/O
Region Availability	US (Other regions coming)	All Azure Regions
Authentication	Azure Active Directory	Access keys
Authorization	POSIX ACLs on Files and Folders	Access Keys
Server-side Encryption	Yes	Yes

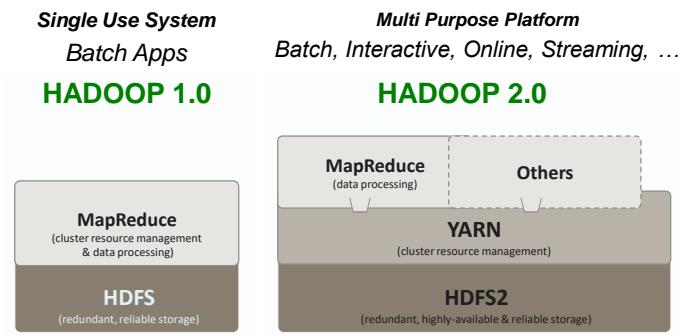


Configuración de Servicios Hadoop

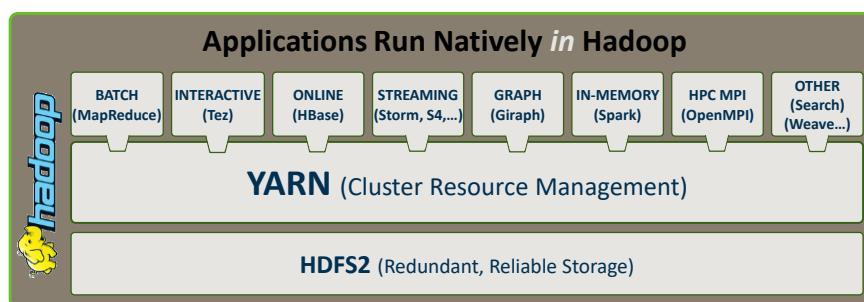
Agenda

- Framework de Hadoop 2.0
- YARN
- Ficheros de Configuración

Hadoop 2.0



Arquitectura Hadoop 2.0



Servicios Hadoop

- Servicios HDFS
 - Gestiona el almacenamiento
 - Las unidades son NameNodes y DataNodes
 - NameNode – mantiene metadatos, en memoria, sobre la estructura hdfs y nombres
 - DataNodes – nodos que comunican el NameNode cambios en hdfs o actualizaciones durante las computaciones locales
- Servicios YARN
 - ResourceManager – el servicio “maestro” para el cluster que se ejecuta en uno de los headnodes
 - Responsable de dirigir los recursos del cluster y la planificación de trabajos en los nodos de trabajo
 - ApplicationMaster – Un servicio maestro único por aplicación.
 - Coordinada la ejecución de una aplicación en el cluster y negocia con el ResourceManager los recursos para la aplicación

95

Introducción

- Basado en el framework de Google Map Reduce y en el Sistema de ficheros de Google
- Procesamiento de datos distribuidos a gran escala
- Pensado para hardware “commodity”
- Auto recuperable
- Escrito en Java

96

Introducción

- Map Reduce forma parte del Core de Hadoop junto con HDFS
- Nos proporciona un modelo de programación paralelo
- Divide una tarea entre procesadores “cerca” de los datos y ensambla los resultados
- Se encarga de programar y tolerancia a fallos
- Monitorización y reporte de Estado

97

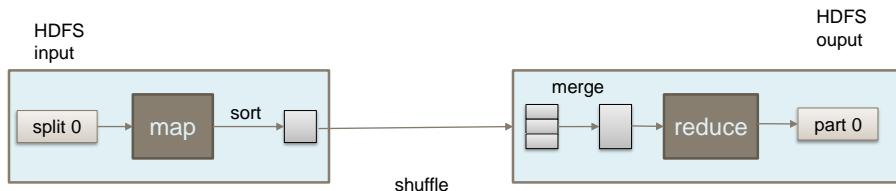
¿Por qué MapReduce?

- Aplicaciones de procesado de gran cantidad de datos
- Divide los datos y procesa en varios nodos
- Cada aplicación maneja
 - Comunicación entre los nodos
 - División y programación del trabajo
 - Tolerancia a Fallos
 - Monitorización y reporting

98

Map Reduce – Single Reducer

► 1 Mapper 1 Reducer



99

Map Reduce

► Map

- Basado en lenguajes funcionales
- Ejecuta una función cerca de la partición de datos
- Uno o más mappers por host
- Múltiples hosts ejecutando mappers
- Tiene como salida un par de valores Clave
- **Map f** listas: aplica una función f a cada elemento de una lista y devuelve una nueva lista
 - Map square [1 2 3 4 5]=[1 4 9 16 25]

100

Map Reduce

► Reduce

- Basado en lenguajes funcionales
- Se ejecuta como una fase posterior después de la fase mapper
- Uno o más reducers por host
- Múltiples hosts ejecutando reducers
- **Reduce g** lista: combina elementos de una lista utilizando la función g para generar un nuevo valor
 - Reduce sum[1 2 3 4 5]=[15]

101

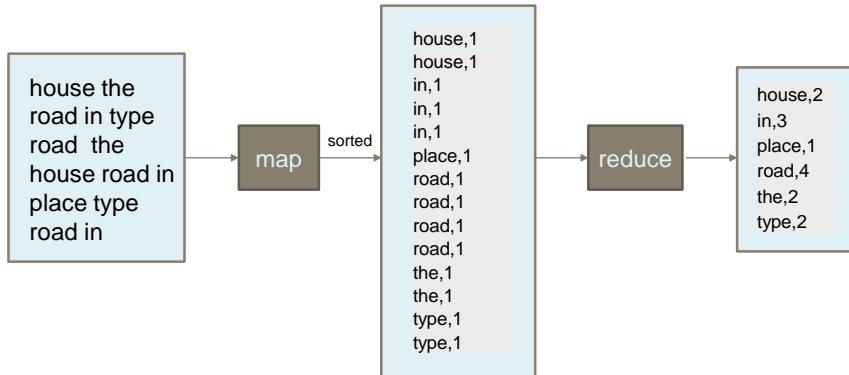
Jobtracker y Tasktracker

- JobTracker -- Maestro
 - Divide las tareas según la ubicación de los datos
 - Programa y monitoriza varias tareas map reduce
- Task Tracker -- Esclavos
 - Ejecuta tareas map y reduce

102

Ejemplo Cuenta Palabras

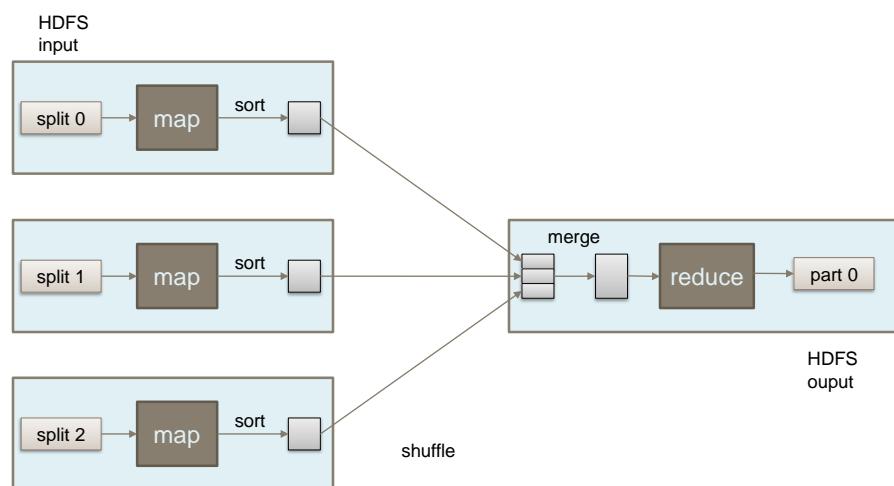
► 1 Mapper, 1 Reducer



103

Map Reduce

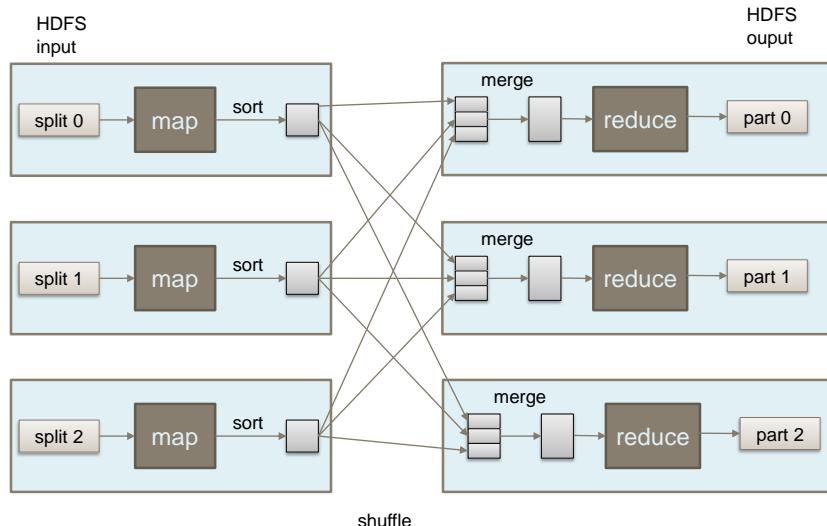
► 3 Mappers 1 Reducer



104

Map Reduce

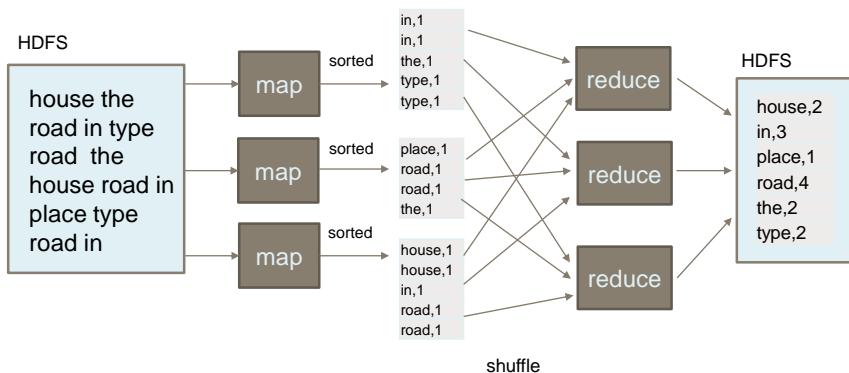
► 3 Mappers, 3 Reducers



105

Ejemplo cuenta palabras

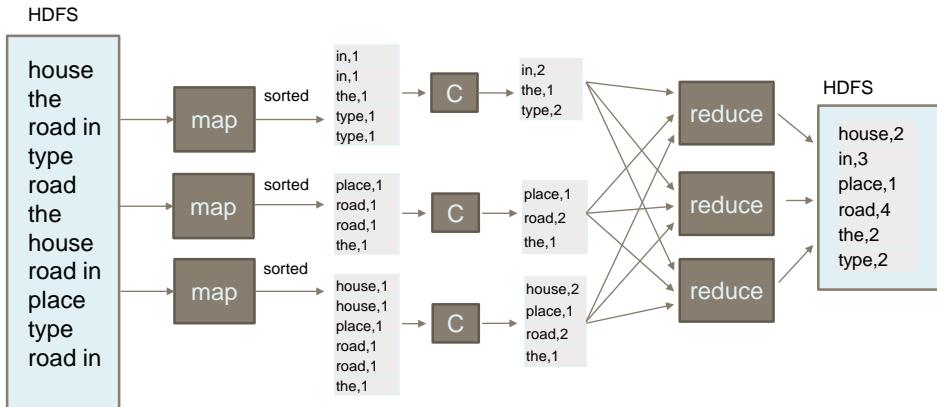
► 3 Mappers, 3 Reducers



106

Combiner

► 3 Mappers, 3 Combiners, 3 Reducers



107

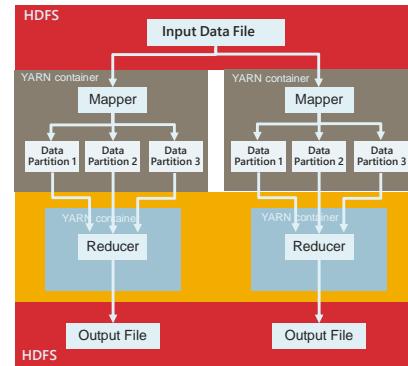
Combiner

- La función Combiner después y sobre la salida de la función map
- Reduce ancho de banda de red
- La función debe de ser **acumulativa** y **asociativa**
 - No se puede utilizar para todo tipo de cálculos, como medias o significados
- Funciona con Cuenta palabras, Máximos, etc

108

En Resumen....

- ❖ Hadoop divide el fichero de entrada y asigna cada trozo a un mapper diferente.
- ❖ Hadoop lee localmente el trozo línea por línea y llama a map() para cada línea, pasándolo como parámetros clave / valor
- ❖ El mapper genera otro par de clave / valor intermedio
- ❖ Todos los valores intermedios para una clave intermedia se combinan juntos en una lista.
- ❖ La lista se le da a uno o varios Reducers
 - Todos los valores asociados con una clave intermedia van al mismo Reducer
 - Se envían ordenadas por clave 'shuffle and sort'
 - Hadoop llama reduce() por cada línea de entrada
 - El Reducer genera los pares clave / valor finales que se escriben en HDFS

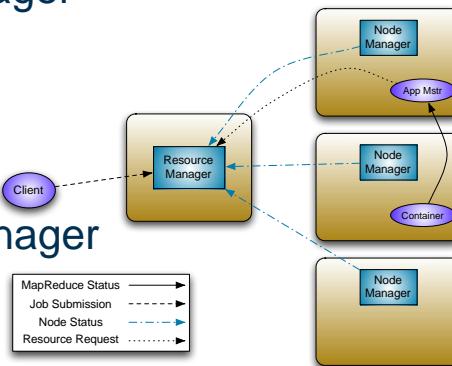


Agenda

- Framework de Hadoop 2.0
- **YARN**
- Ficheros de Configuración

Arquitectura YARN

- Resource Manager
- Node Manager
- Application Manager



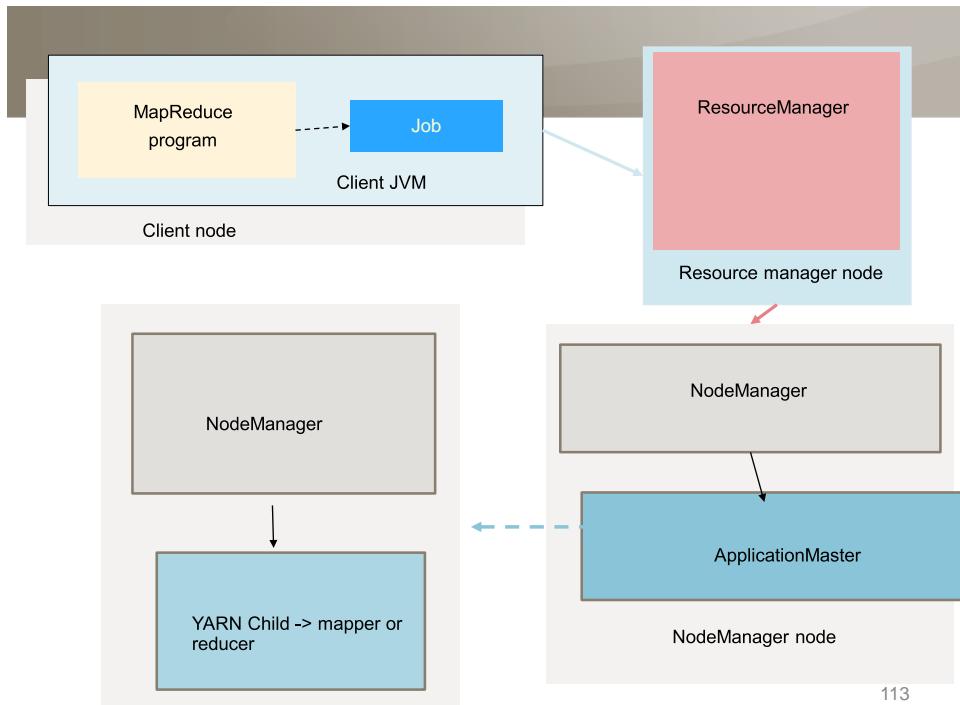
YARN Daemons

- **ResourceManager (RM)**
 - Se ejecuta en el nodo maestro
 - Planificador de recursos Global
 - Arbitra los recursos del Sistema entre aplicaciones
 - Tiene un planificador externo para soportar diferentes algoritmos
- **NodeManager (NM)**
 - Se ejecuta en un nodo de trabajo (en un contenedor)
 - Se comunica con RM para asegurar que tiene recursos y que está vivo
- **Application Master (AM)**
 - Uno por aplicación
 - Se ejecuta en un contenedor

ResourceManager

NodeManager

ApplicationMaster



113

Mejoras clave con YARN

- Framework que soporta múltiples aplicaciones
- Utilización del cluster
- Escalabilidad
- Agilidad
- Servicios Compartidos

Cargas de Trabajo en YARN



Variantes de Cargas de Trabajo



Características
• Computaciones “duras”
Ejemplos
• Machine Learning
• Natural Language Processing

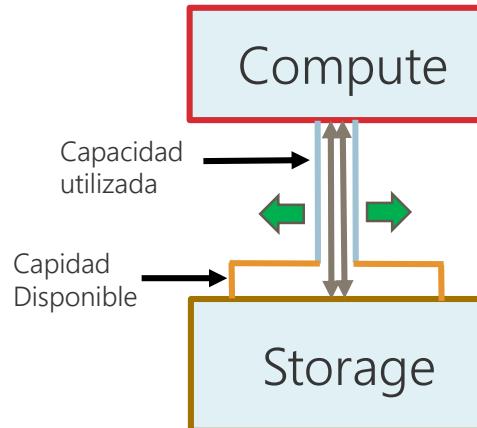


Características
• Gran uso memoria
Ejemplos
• PageRank
• Real-time analytics

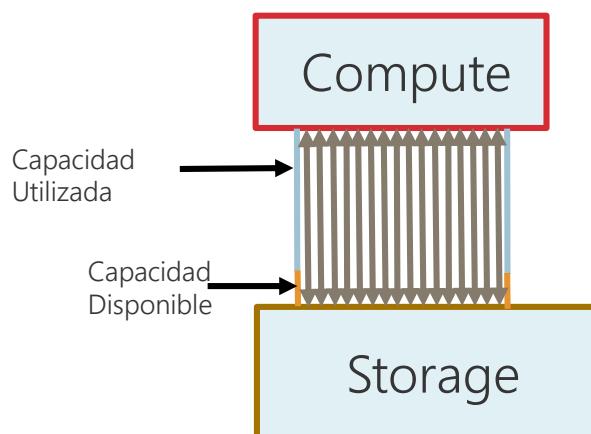


Características
• Lectura / Escritura
Ejemplos
• Copy
• Data preparation

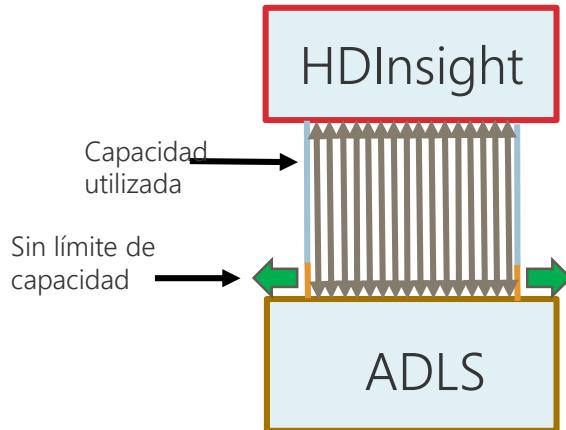
Uso de la capacidad disponible



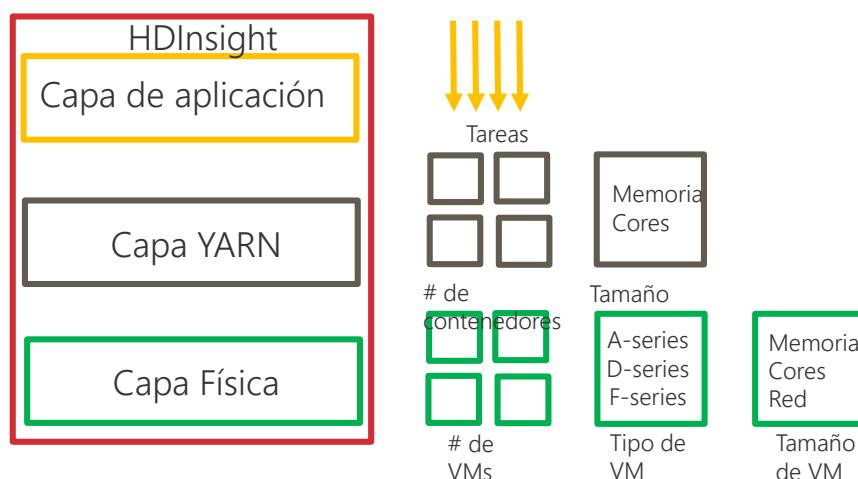
Consumo de la Capacidad Disponible



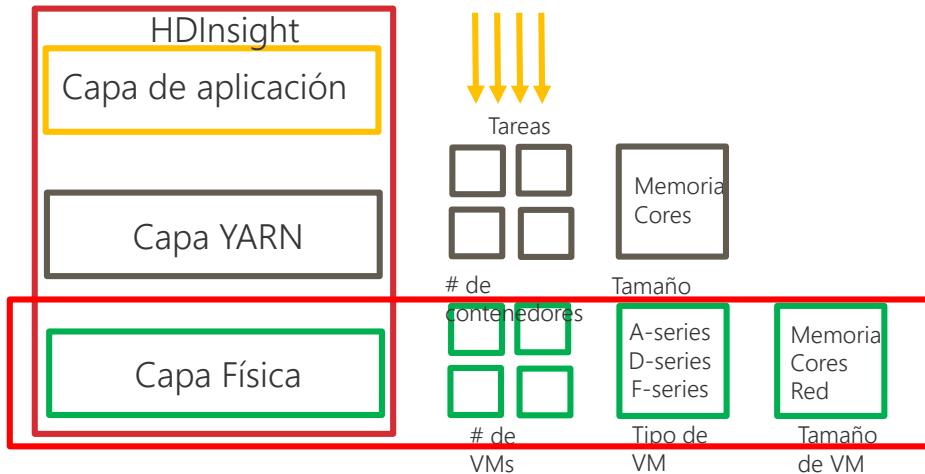
ADLS proporciona capacidad máxima



Anatomía de un cluster

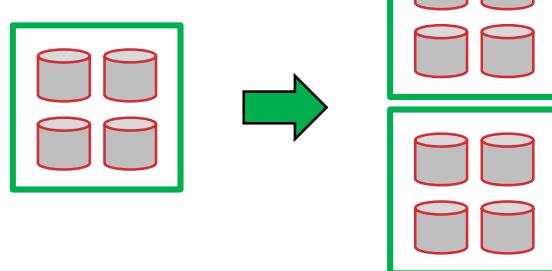


Anatomía de un cluster



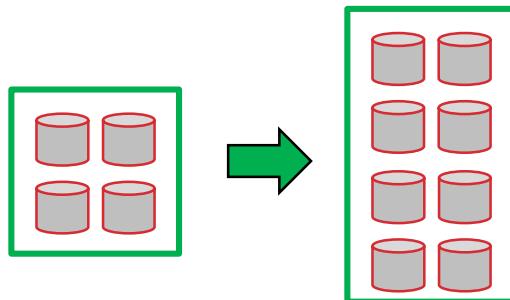
Capa Física

Incremento # de VMs



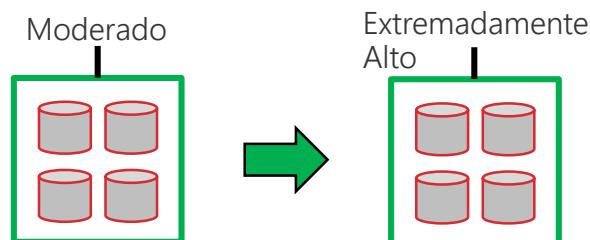
Capa Física

Usar VMs con más memoria

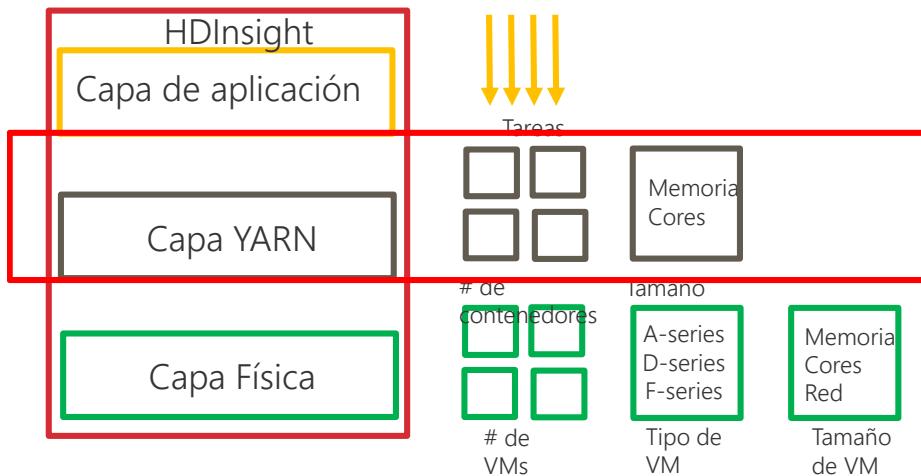


Capa Física

Usar VMs con más ancho de banda de red

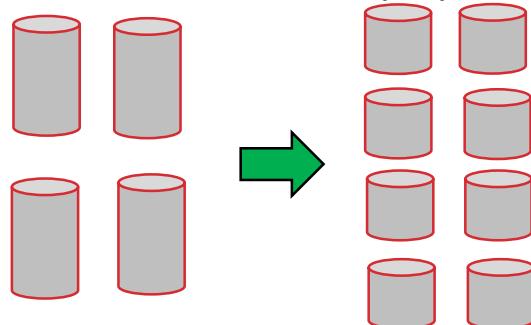


Anatomía de un cluster



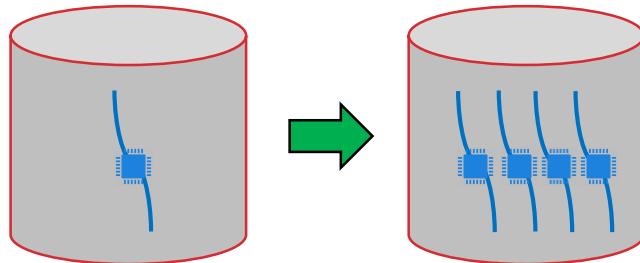
Capa YARN

Usar contenedores YARN más pequeños

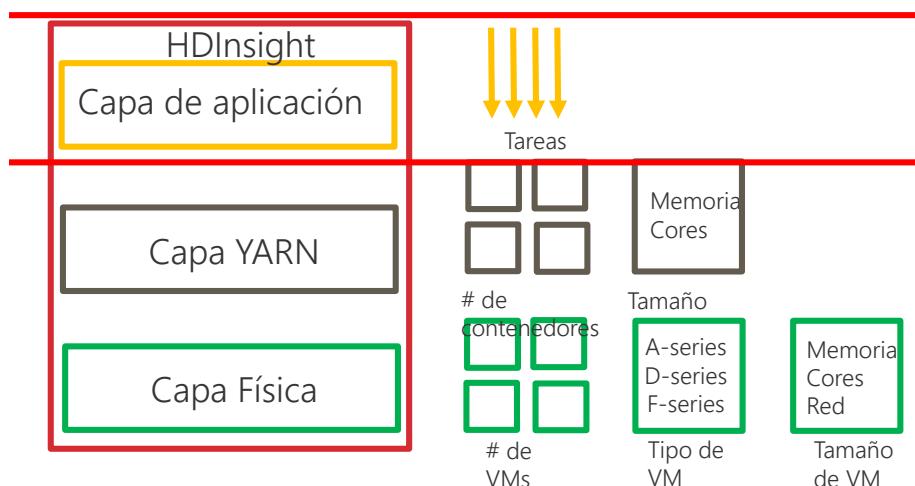


Capa YARN

Incrementar Cores por contenedor

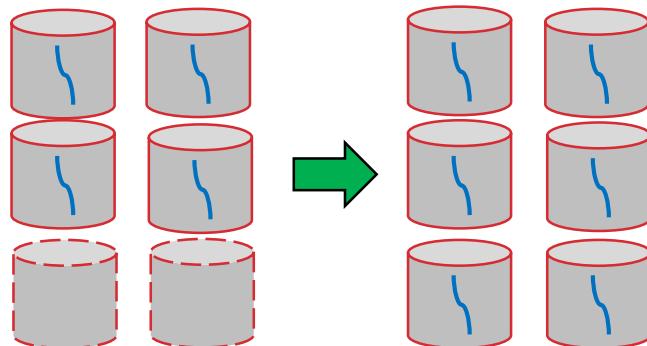


Anatomía de un cluster



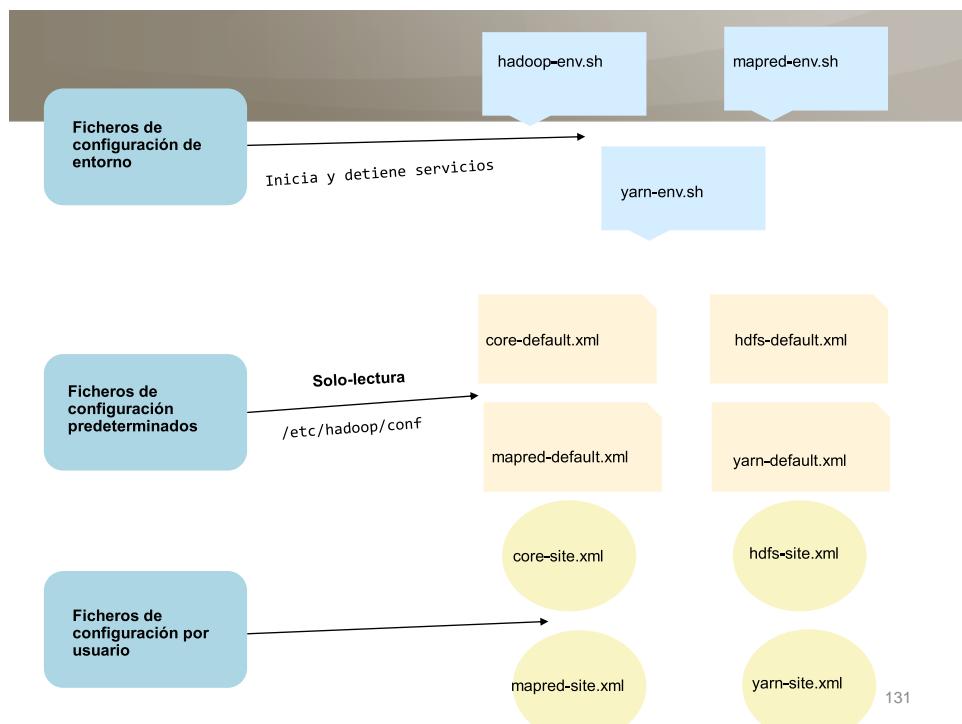
Capa de Aplicación

Utilizar todos los contenedores posibles



Agenda

- Framework de Hadoop 2.0
- YARN
- **Ficheros de Configuración**

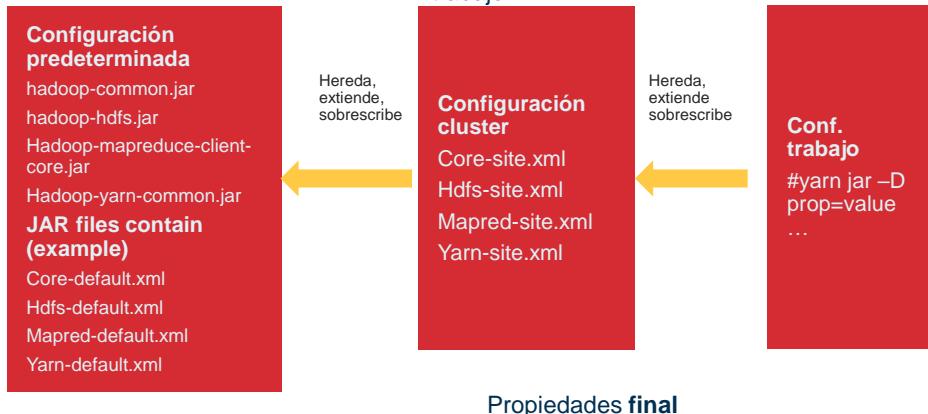


Ficheros de configuración

Nombre Fichero	Formato	Propósito
core-site.xml	Hadoop configuration XML	Configuraciones base de Hadoop, HDFS, YARN, MapReduce
hdfs-site.xml	Hadoop configuration XML	Configuraciones específicas HDFS (NameNode y DataNode)
yarn-site.xml	Hadoop configuration XML	Configuraciones YARN
Mapred-site.xml	Hadoop configuration XML	MapReduce
Hadoop-env.sh	Bash script	Variables de entorno
log4j.properties	Java properties	Configuraciones del log de sistema
Hadoop-metrics2.properties	Java properties	Configuración de métricas

Precedencia de Configuración

La configuración actual para cualquier trabajo en ejecución en un cluster se deriva de una combinación de orígenes, incluyendo configuración predeterminada, configuración del cluster o del nodo y la configuración del trabajo



133

Demo



Configuración del cluster Hadoop

134



Uso de Ambari para Monitorización y Administración

Agenda

- ¿Qué es Ambari?
- Administrando Hadoop
- Alertas, Servicios y Hosts
- Vistas de Usuario

¿Qué es Ambari?

- Framework 100% Software Libre para aprovisionar, administrar y monitorizar clusters Apache Hadoop
- Aprovisionamiento
 - Proporciona asistentes paso a paso para instalar servicios Hadoop a través de hosts
 - Maneja la configuración de servicios
- Administración
 - Proporciona un punto de administración central para iniciar, detener y reconfigurar servicios a través de todo el cluster
- Monitorización
 - Proporciona cuadros de mando de monitorización de estado
 - Ambari Metrics System
 - Ambari Alert Framework

137

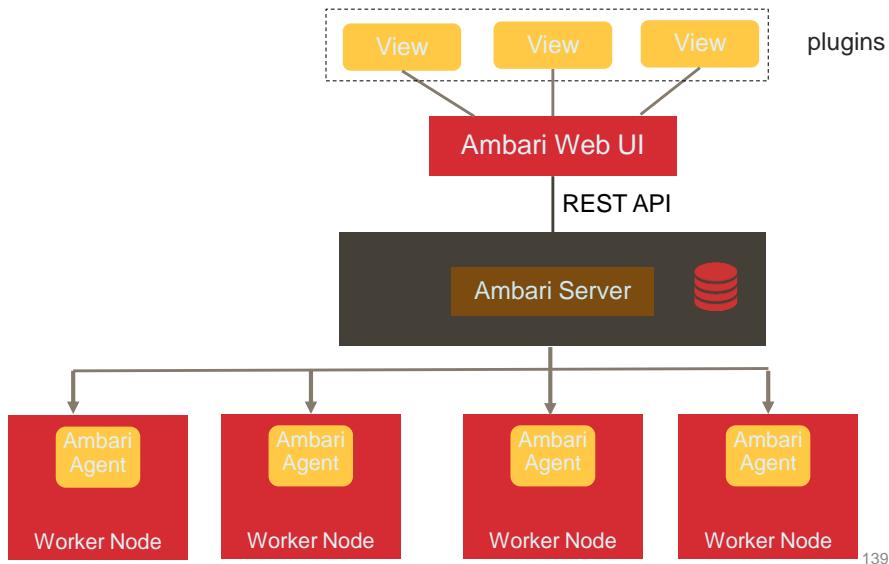
Características



- ❖ Instalación por asistentes
- ❖ Instalación de cluster a través de API no interactiva
- ❖ Control Granular de servicios
- ❖ Configuración de Servicio de Cluster
- ❖ Monitorización y Alertas
- ❖ REST API para integración
- ❖ Ambari Views para plug-ins

138

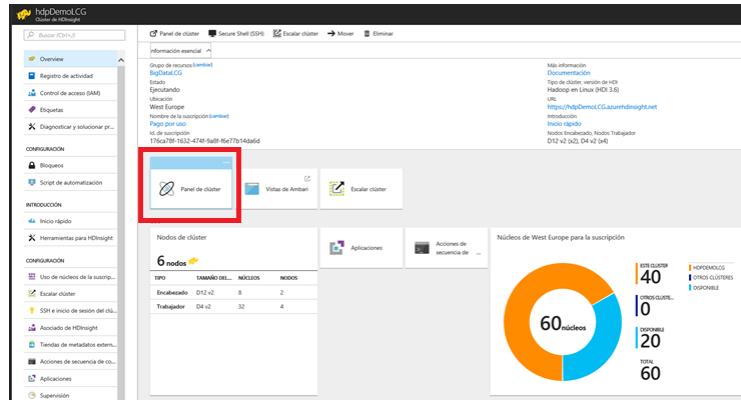
Arquitectura



Agenda

- ¿Qué es Ambari?
- **Administrando Hadoop**
- Alertas, Servicios y Hosts
- Vistas de Usuario

El interfaz

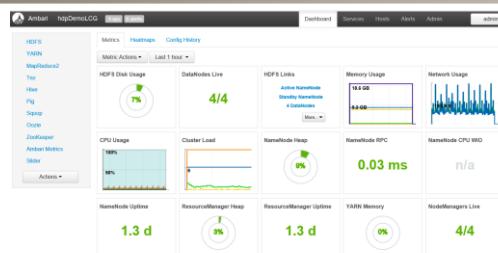


Acceso desde el portal Azure

141

Cuadro de Mando de métricas

- La pestaña de métricas del Cuadro de Mando muestra métricas a nivel de cluster:
 - Uso CPU
 - Uso disco HDFS
 - Uso Memoria
 - Uso Red
 -
- Nos permite conocer el estado de un vistazo
- Puede personalizarse agregando o quitando widgets

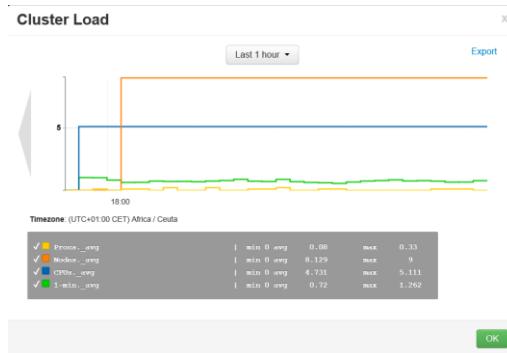


- La lista de servicios instalados siempre aparece en el panel de la izquierda.
- El botón de Acciones permite iniciar, detener y reiniciar el cluster

142

Detalle

- Podemos obtener más detalle de:
 - Uso CPU
 - Carga del Cluster
 - Uso Red
 - Uso de Memoria
- Las estadísticas de uso pueden verse por períodos de tiempo
- Para otras métricas podemos ver información adicional pasando el ratón



143

Personalización

- Podemos personalizar
 - YARN Memory
 - Node Managers
 - Resource Managers
 - NameNode CPU
 - NameNode RPC
 - NameNode Heap



144

HeatMap

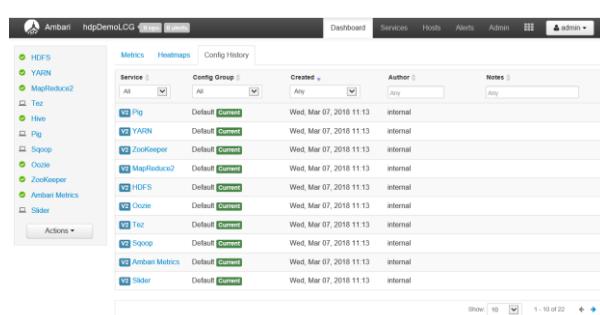
- Vista en color de cada nodo del cluster
- Podemos seleccionar métricas



145

Historia de Configuración

- Lista del histórico de cambios
- Podemos bajar al detalle de cada servicio



The screenshot shows the Ambari interface for the hdpDemolCG cluster. The 'Config History' tab is selected. It displays a table of configuration changes for various services: Pig, YARN, ZooKeeper, MapReduce2, HDFS, Tez, Oozie, Sqoop, and Amban Metrics. Each row includes details like the service, config group, creation date, author, and notes. At the bottom, there are navigation controls for page size and search.

146

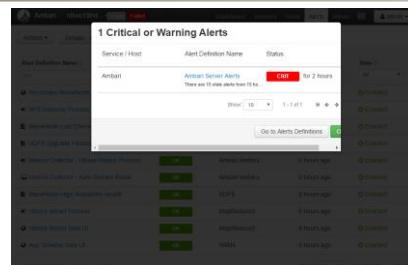
Agenda

- ¿Qué es Ambari?
- Administrando Hadoop
- **Alertas, Servicios y Hosts**
- Vistas de Usuario

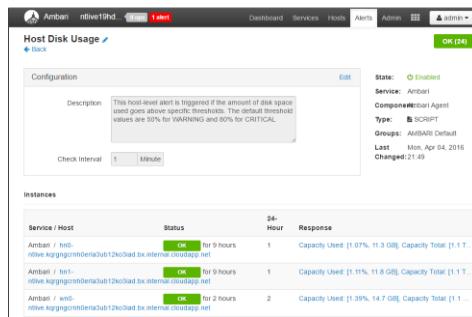
147

Alertas

- Nos muestra alertas en la parte superior de la página



The screenshot shows a modal window titled '1 Critical or Warning Alerts'. It displays a single alert for 'Ambari Server Alert' with the message 'There are 11 disk alerts from 15 K.' The status is 'CRITICAL' for 2 hours. Below the modal, the main dashboard lists several other alerts, each with a status (OK, OK, OK, OK, OK, OK, OK) and a timestamp (1 hour ago, 1 hour ago).

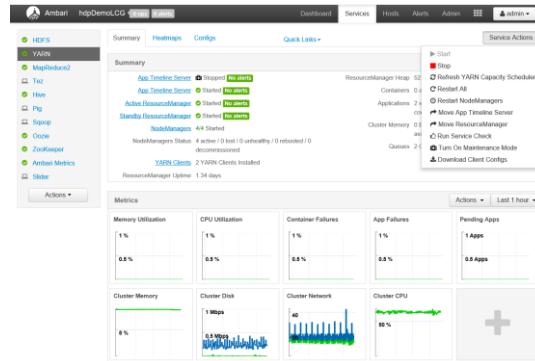


The screenshot shows the 'Host Disk Usage' configuration page. It includes a 'Description' section with a note about host-level alerts being triggered if disk space goes above specific thresholds (50% for WARNING and 80% for CRITICAL). A 'Check Interval' of 1 minute is selected. Below this, a table lists three instances of 'Ambari / host' with their status (OK), last checked time (9 hours ago), and response details (Capacity Used: 1.07%, Capacity Total: 11.3 GB for instance 1; Capacity Used: 1.11%, Capacity Total: 11.8 GB for instance 2; Capacity Used: 1.39%, Capacity Total: 14.7 GB for instance 3).

148

Servicios

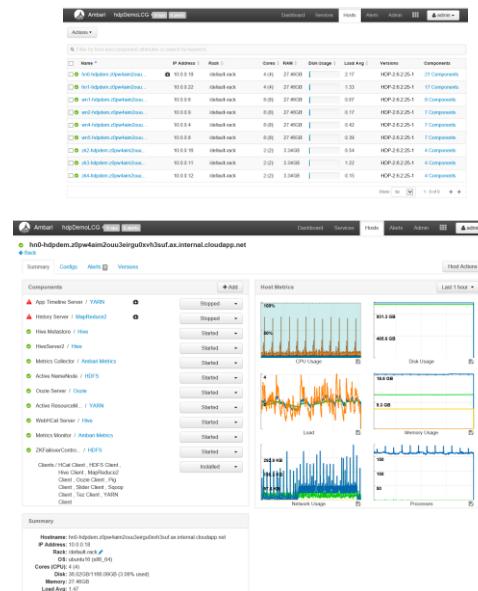
- La página de servicios nos proporciona información sobre el estado de los servicios en ejecución
- Los iconos indican el estado o acciones que debemos de tomar
- Para cada servicio tenemos una lista de acciones



149

Hosts

- Métricas a nivel de sistema para cada nodo
- Podemos ver la lista de componentes
- Detalle del host

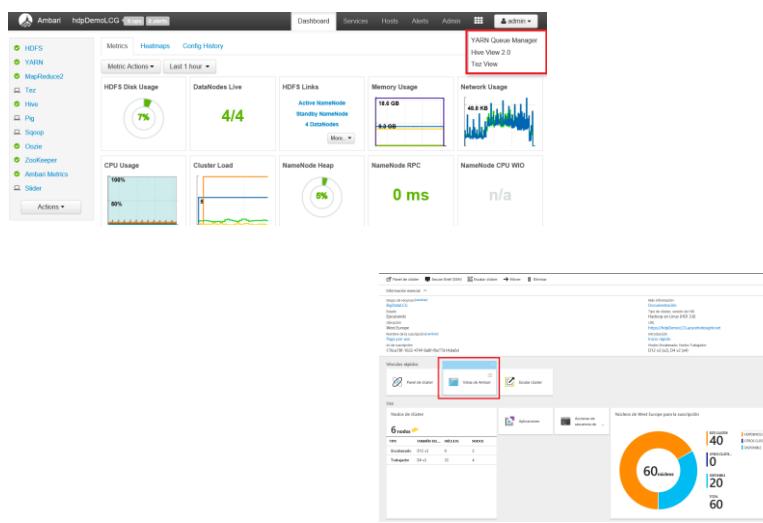


Agenda

- ¿Qué es Ambari?
- Administrando Hadoop
- Alertas, Servicios y Hosts
- **Vistas de Usuario**

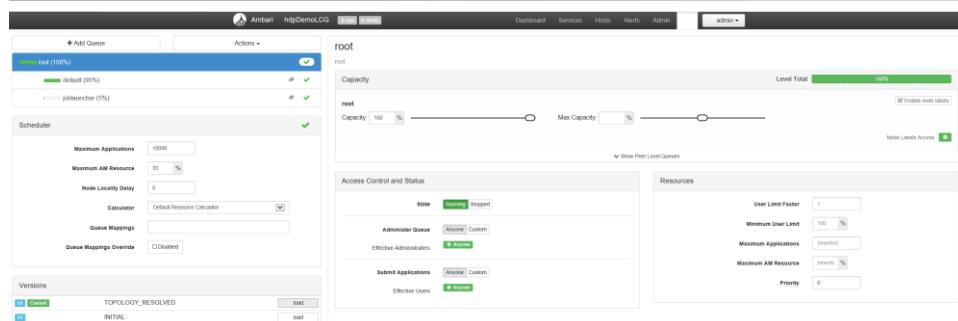
151

Vistas de Usuario



152

Vistas de Usuario: YARN



- Uso de las colas y configuración
- Podemos definir nuevas colas

153

Vistas de Usuario: TEZ

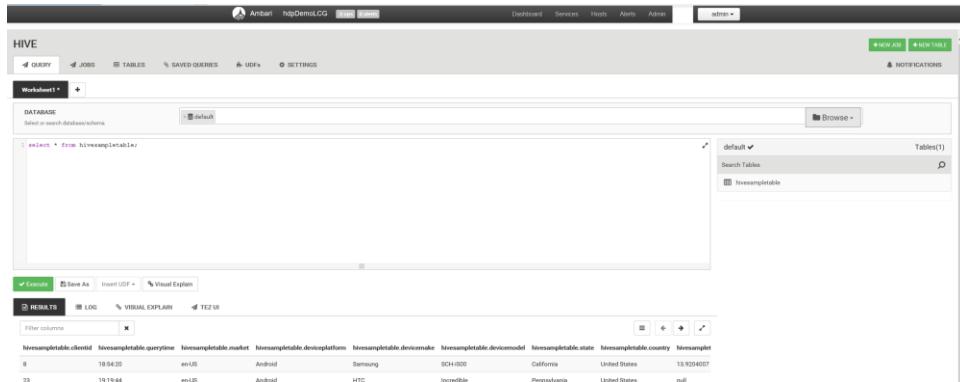


- Muestra todos los DAG en un periodo de tiempo
- Podemos acceder al detalle



154

Vistas de Usuario: Hive



The screenshot shows the Ambari Hive interface. At the top, there are tabs for QUERY, JOBS, TABLES, SAVED QUERIES, UDFs, and SETTINGS. Below the tabs, a query window displays the command: `select * from hivesampletable;`. To the right of the query window is a sidebar titled "default" with a "Tables(1)" section containing a single entry: "hivesampletable". Below the sidebar is a results table with one row of data:

hivesampletable.clientid	hivesampletable.querytime	hivesampletable.market	hivesampletable.devicetype	hivesampletable.devicebrand	hivesampletable.devicemodel	hivesampletable.state	hivesampletable.country	hivesampletable
8	18:54:20	en-US	Android	Samsung	SCH-I800	California	United States	139204007
33	10:19:44	en-US	Android	HTC	Incredible	Pennsylvania	United States	null

155

Demo / Lab



Navegando por interfaz Ambari

156



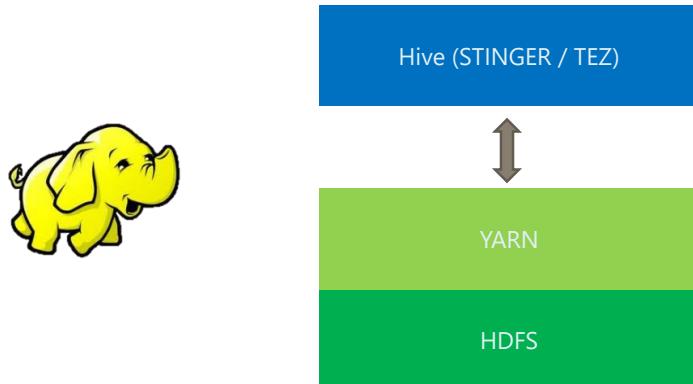
Fundamentos de Hive

Agenda

- **Introducción a Hive**
- Arquitectura Hive
- HiveQL y Almacenamiento
- UDF y MapReduce
- Particiones y cubos
- Navegando Hive con Excel
- Estrategias de Join
- Evolución Hive: Tez y LLAP

Introducción

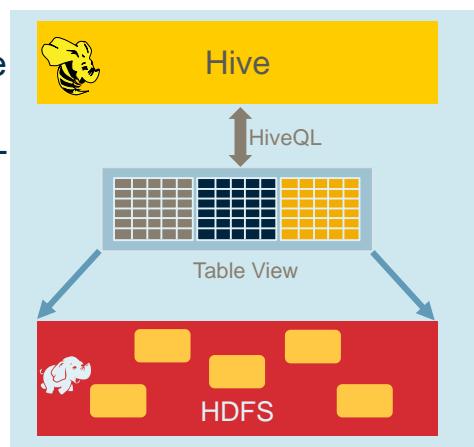
- Sistema Data Warehouse para Hadoop



159

Introducción

- Permite exponer como “tablas” el contenido de HDFS
- Lenguaje similar a SQL llamado HiveQL
- Permite extender el lenguaje HiveQL
- Estructuras de datos con tipos de datos “ricos” (structs, listas y mapas)
- Consultas de datos de diferentes formatos: texto / binario



160

Introducción

- Facilita summarizaciones de datos
- Consultas Ad-hoc
- Análisis de grandes cantidades de datos almacenados en Hadoop

161

¿Por qué Hive?

- Map Reduce duro de programar
- Carecen de expresividad
- SQL es un lenguaje familiar
- Permite consultar de forma estructurada datos no estructurados
- Bueno para Análisis
- Rendimiento

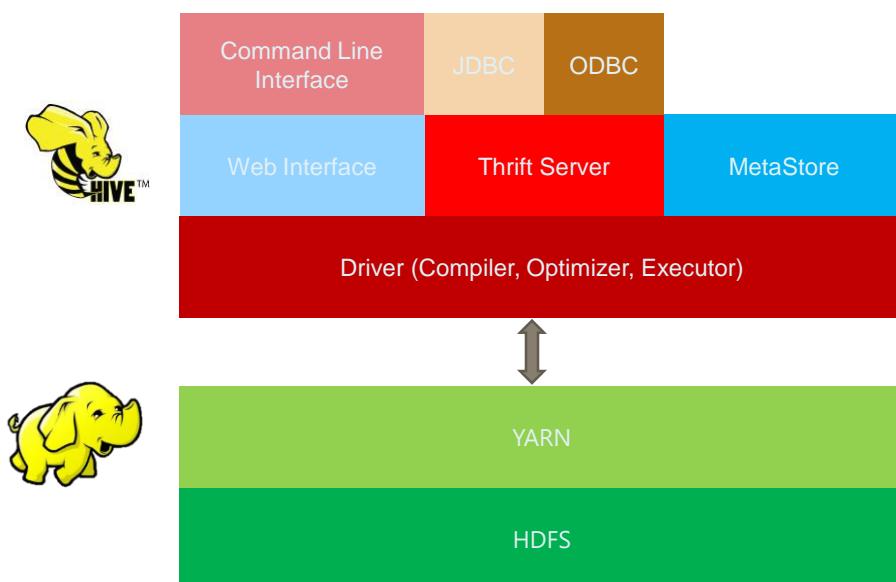
162

Agenda

- Introducción a Hive
- **Arquitectura Hive**
- HiveQL y Almacenamiento
- UDF y MapReduce
- Particiones y cubos
- Navegando Hive con Excel
- Estrategias de Join
- Evolución Hive: Tez y LLAP

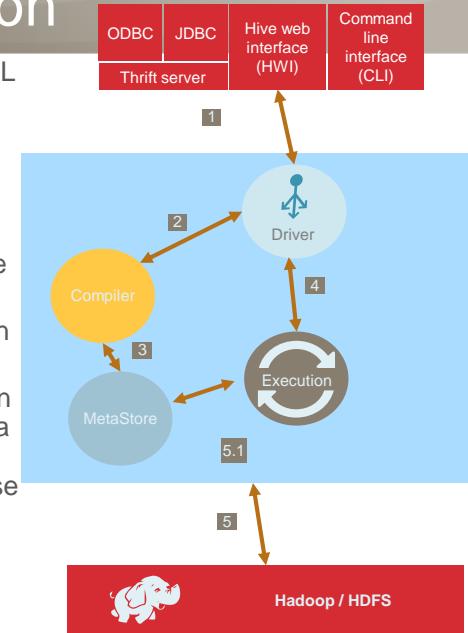
163

Arquitectura Hive



Flujo de Ejecución

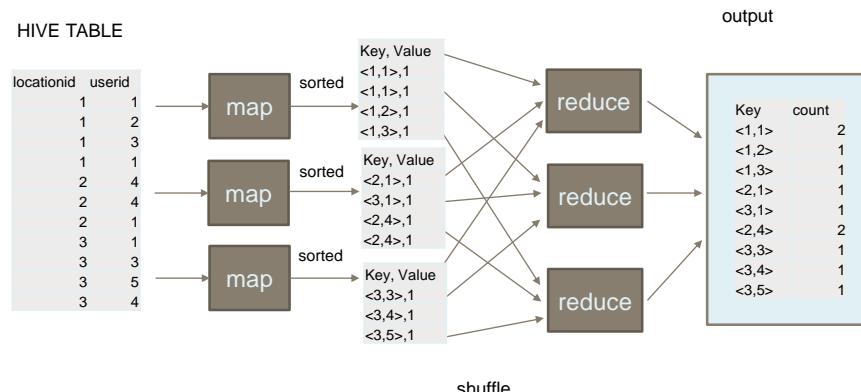
- 1 Cliente Hive envía consulta HiveQL al driver
- 2 El driver envía la consulta al compilador para comprobación de sintaxis y parsing.
- 3 El compilador obtiene metadatos del almacen y devuelve un plan de ejecución al driver.
- 4 El driver envía el plan de ejecución al motor.
- 5 El motor de ejecución envía el plan como un trabajo Map Reduce. Una vez completada la ejecución el motor obtiene los resultados que se devuelven al driver que los devuelven al cliente Hive.
- 5.1 El motor de ejecución puede ejecutar operaciones de metadatos contra el Metastore



Agrupar por ubicación, userid

► 3 Mappers, 3 Reducers

- Trabajo para una consulta Hive



Demostración



Consultas HIVE simples

167

Agenda

- Introducción a Hive
- Arquitectura Hive
- **HiveQL y Almacenamiento**
- UDF y MapReduce
- Particiones y cubos
- Navegando Hive con Excel
- Estrategias de Join
- Evolución Hive: Tez y LLAP

168

HiveQL vs SQL-92

Caract.	SQL	HiveQL
Updates	UPDATE, INSERT, DELETE	INSERT OVERWRITE, NO DELETE NO UPDATE
Transacciones	Soportado	No Soportado
Índices	Soportado	No Soportado
Latencia	Sub-segundo	Minutos
Funciones	Cientos de funciones	Docenas de funciones
Inserts Multitable	No Soportado	Soportado
Create table as select	No válido en SQL-92 pero disponible en varios RDBS	Soportado
Select	SQL-92	Solo un objeto en e IFROM. SORT BY para ordenación parcial. LIMIT para limitar filas
Joins	Soportado	Soportado
Subqueries	Cualquier cláusula. Correlacionada o no	Solo cláusula FROM. Subconsultas correlacionadas no soportadas
Vistas	Actualizable. Materializada para la no materializada	Solo lectura. Materialización no soportada
Puntos extensión	Funciones de usuario. Procedimientos almacenados	Funciones de usuario. Scripts MapReduce.

169

Tipos de Datos

Type	Description
TINYINT	1 byte signed integer
SMALLINT	2 byte signed integer
INT	4 byte signed integer
BIGINT	8 byte signed integer
FLOAT	4 byte single precision floating point number
DOUBLE	8 byte double precision floating point number
BOOLEAN	true/false
STRING	Character string 'a', "a"
ARRAY	An ordered collection of fields, must all be of the same type
MAP	An unordered connection of Key-values pairs. Keys must be primitives. Keys must be the same type, values must be the same type. map('a',1,'b',2)
STRUCT	A collection name fields. The fields may be of different types. struct ('a',1,1.0)

170

Almacenamiento de Tablas(I)

- Datos almacenados en HDFS
- La operación de carga es una copia de fichero
- La tabla especifica como parsear el fichero para devolver los datos.
- Se denomina esquema o lectura
- No todos los datos de los ficheros tienen que utilizarse en la definición de la tabla
- Rendimiento en la carga (no parse), flexibilidad en consultas

171

Almacenamiento de tablas(II)

- Tablas Administradas
 - Los ficheros se copian el directorio hive
- Tablas externas
 - Referencias a ficheros en su ubicación original

172

Almacenamiento de Tablas(III)

- Formato de fila
 - Filas y campos en una fila
 - Por defecto: texto delimitado
- Formato Fichero
 - Formato contenedor
 - Por defecto: fichero de texto
 - Existen también formatos binarios orientados a columnas (RCFiles) y filas

173

Cómo ejecutar una consulta

- Hive CLI
 - El cliente pesado HIVE
 - \$hive
 - Hive>
- Beeline
 - Nueva consola de comandos que conecta a una instancia HiveServer2
 - \$ beeline –u url –n username –p password
beeline>
- Vistas de Ambari / Herramientas como Visual Studio

174

HiveQL CREATE TABLE

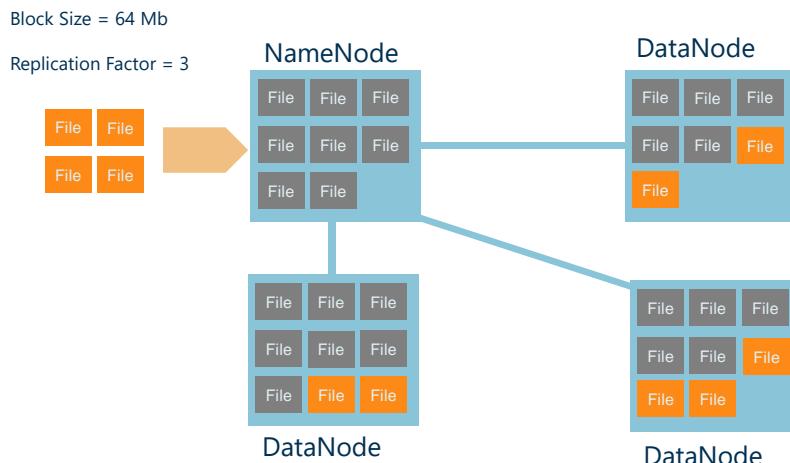
```
CREATE [EXTERNAL] TABLE table_name
[(col_name data_type [COMMENT col_comment], ...)]
[ [ROW FORMAT row_format] [STORED AS file_format] ]
[LOCATION hdfs_path]
```

```
CREATE EXTERNAL TABLE log (gid INT, name STRING,
pubname STRING, releaseDate STRING, rating DOUBLE)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LOCATION '/user/paco/log';
```

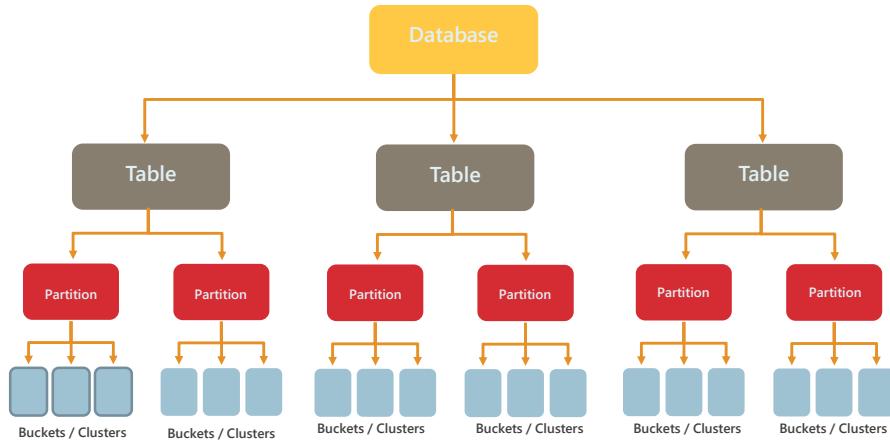
- Si no se especifica External, los datos se mueven a la carpeta hive
- Tab como delimitador de campo

175

Recordemos HDFS



Organización Jerárquica datos



177

Demostración



CREATE TABLE

178

HiveQL ALTER TABLE

--Rename Table

```
ALTER TABLE tableName RENAME TO newTableName
```

--Change column name, type, and position

```
ALTER TABLE tableName CHANGE col_old_name col_new_name  
column_type [COMMENT comment] [FIRST|AFTER colName]
```

--Add or replace column

```
ALTER TABLE tableName ADD|REPLACE  
COLUMNS (col_name data_type [COMMENT col_comment], ...)
```

179

HiveQL DROP TABLE

```
DROP TABLE tableName
```

- Borra los ficheros si la tabla es administrada, si es externa elimina la definición de tabla

180

HiveQL Otros comandos DDL

```

CREATE/DROP DATABASE dbName

CREATE/DROP VIEW viewName AS SELECT ...

CREATE/DROP FUNCTION funcName AS className

CREATE/DROP INDEX idxName ON TABLE tblName (colName)

SHOW DATABASES/TABLES/COLUMNS

```

181

Creación de una Tabla

Soporta tipos complejos como “map”

Internal Tables

Se gestiona / controla los datos. Si una tabla interna se borra, los datos asociados se eliminan. Predeterminada

External Tables

No controla / gestiona los datos. Si se elimina los datos asociados no se borran

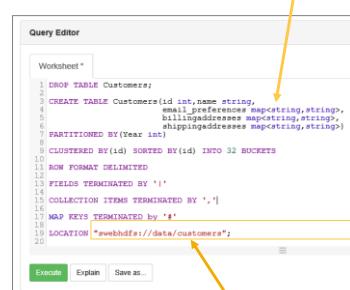
LOCATION

Define la ubicación de los datos

- Puede ser una carpeta HDFS que ya contiene datos. O podemos agregar datos más tarde a una carpeta vacía
- Si no se especifica el predeterminado es: /hive/warehouse/

Varias vistas

Podemos tener varias tablas apuntando a los mismos datos



```

Query Editor

Worksheet *
1 DROP TABLE Customers;
2 CREATE TABLE Customers(id int, name string,
3                         preferences map<string, string>,
4                         billingaddresses map<string, string>,
5                         shippingaddresses map<string, string>)
6 PARTITIONED BY(year int)
7 CLUSTERED BY(id) SORTED BY(id) INTO 32 BUCKETS
8 ROW FORMAT DELIMITED
9 FIELDS TERMINATED BY ','
10 COLLECTION ITEMS TERMINATED BY '['
11 MAP KEYS TERMINATED BY '#'
12 LOCATION "wasb://data/customers"
13
14 Execute Explain Save as...

```

HDInsight puede apuntar directamente a directorios de ADLS

182

HiveQL INSERT

-- INSERT

```
INSERT INTO tableName  
SELECT ...
```

-- LOAD

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE]  
INTO TABLE tablename
```

183

Carga de Datos

Opciones

Cargar datos locales

```
LOAD DATA LOCAL INPATH ... INTO TABLE ...  
hadoop dfs -put 'local source' 'destination'
```

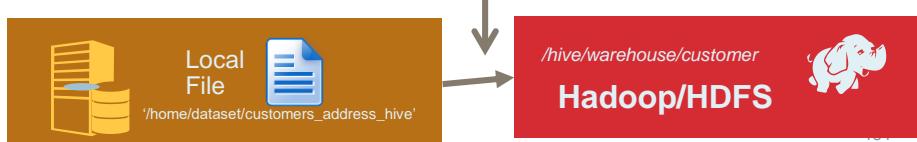
Cargar datos de otras tablas Hive

```
INSERT INTO TABLE 'destination-hive-table'  
SELECT 'select-statement' FROM 'source-hive-table'
```

Insertar valores directamente

```
INSERT INTO TABLE 'tablename'  
VALUES (row1_values), (row2_values), ....
```

```
LOAD DATA LOCAL INPATH '/home/dataset/customers_address_hive'  
OVERWRITE INTO table Customers PARTITION (year=2015);
```



HiveQL SELECT

```
SELECT [DISTINCT] <attrList>
FROM <tableExpr>
[WHERE <condition>]
[GROUP BY <attrList>]
[HAVING <condition>]
[ORDER BY <attr> [ASC|DESC], ... ]
[LIMIT <number>]
```

185

HiveQL Joins

- FULL OUTER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN, LEFT SEMI JOIN, JOIN (INNER JOIN)

```
SELECT sales.* , customers.* FROM
sales JOIN customers
ON sales.customerid=sales.customerid
```

186

HiveQL GROUP BY

- Soportado sobre cualquier número de columnas o expresiones.
- Funciones de agregado: MIN, MAX, COUNT, SUM, AVG
- Cláusula HAVING aplicada DESPUES del GROUP BY, el resultado es un filtro de grupo agregado

```
SELECT sales.*, customers.* FROM
sales JOIN customers
ON sales.customerid=sales.customerid
```

187

Sub Consultas

- Solo en la cláusula FROM

```
SELECT col1, col2
FROM (SELECT col1, col2 ... FROM Table_1 ...) t
GROUP BY col1, col2
```

188

Demostración



Hive Queries

189

Agenda

- Introducción a Hive
- Arquitectura Hive
- HiveQL y Almacenamiento
- **UDF y MapReduce**
- Particiones y cubos
- Navegando Hive con Excel
- Estrategias de Join
- Evolución Hive: Tez y LLAP

190

User Defined Functions

- UDF, consume una fila, genera una fila, por ejemplo funciones matemáticas y de cadena. En c# usando TRANSFORM
- UDA(gregate)F: consumen múltiples filas, produce una única fila. Por ejemplo COUNT y MIN. Solo Java.
- UDT(able)F: consume una fila, genera múltiples filas. Solo JAVA.

191

HIVE Map Reduce .NET

- Similar a Hadoop Streaming.
- Cláusulas TRANSFORM, MAP, y REDUCE invocan un programa externo a HIVE.
- El programa necesita registrarse en Hive

```
FROM records
SELECT TRANSFORM(stuff)
USING 'program'
AS thing1, thing2
```

192

HIVE Map Reduce .NET

```
FROM (
    FROM pv_users
    MAP pv_users.userid, pv_users.date
    USING 'map_program'
    AS dt, uid
    CLUSTER BY dt) map_output
INSERT OVERWRITE TABLE pv_users_reduced
    REDUCE map_output.dt, map_output.uid
    USING 'reduce_program'
    AS date, count;
```

- Si utilizamos anidamiento, podemos utilizar una función MAP REDUCE. SELECT TRANSFORM tendría los mismos resultados

193

Agenda

- Introducción a Hive
- Arquitectura Hive
- HiveQL y Almacenamiento
- UDF y MapReduce
- **Particiones y cubos**
- Navegando Hive con Excel
- Estrategias de Join
- Evolución Hive: Tez y LLAP

194

Particiones

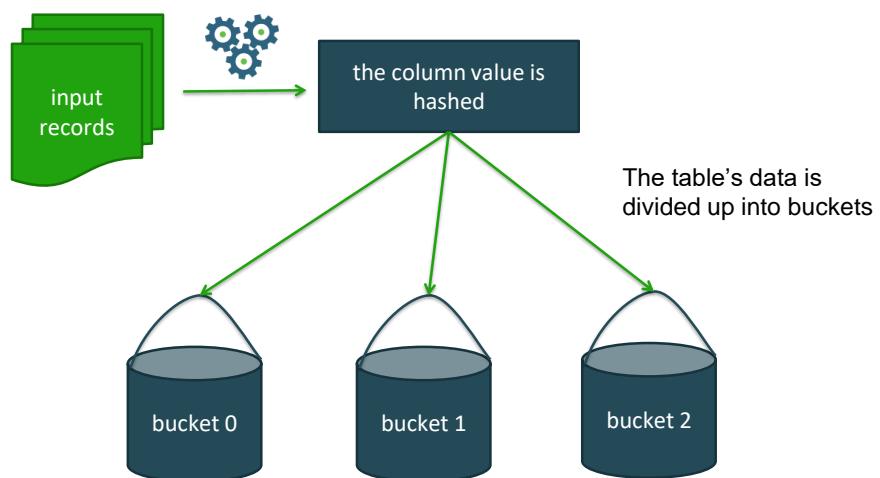
- Definidas en tiempo de creación de la tabla utilizando PARTITIONED BY
- Subdirectorios anidados a nivel de Sistema de ficheros

```
CREATE TABLE logs (ts BIGINT, line STRING)
PARTITIONED BY (dt STRING, country STRING);

LOAD DATA LOCAL INPATH 'input/hive/partitions/file1'
INTO TABLE logs
PARTITION (dt='2001-01-01', country='GB');
```

195

Hive buckets



196

Buckets

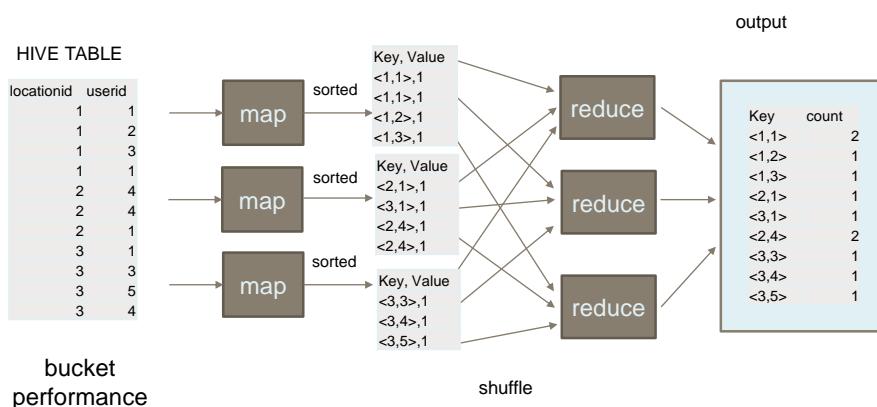
- Posibilita consultas más eficientes, porque se ejecuta Map Reduce cuando se consulta
- Ejemplo de rendimiento MAP JOINS

```
CREATE TABLE bucketed users (id INT, name STRING)
CLUSTERED BY (id) INTO 4 BUCKETS;
```

197

Group by location, userid

► 3 Mappers, 3 Reducers



198

Skewed tables

```
CREATE TABLE Customers (
    id int,
    username string,
    zip int
)
SKEWED BY (zip) ON (57701, 57702)
STORED as DIRECTORIES;
```

199

Distribute By

```
insert overwrite table mytable
select gender,age,salary
from salaries
distribute by age;
```

```
insert overwrite table mytable
select gender,age,salary
from salaries
distribute by age
sort by age;
```

200

Almacenando resultados

```
INSERT OVERWRITE DIRECTORY
```

```
'/user/train/ca_or_sd/'  
from names  
select name, state  
where state = 'CA'  
or state = 'SD';
```

```
INSERT OVERWRITE LOCAL DIRECTORY
```

```
'/tmp/myresults/'  
SELECT * FROM bucketnames  
ORDER BY age;
```

201

Propiedades Map Reduce

```
SET mapreduce.job.reduces = 12
```

```
hive -f myscript.hive  
-hiveconf mapreduce.job.reduces=12
```

```
SELECT * FROM names
```

```
WHERE age = ${age}
```

```
hive -f myscript.hive -hivevar age=33
```

202

Demostración



Particiones y Buckets

203

Agenda

- Introducción a Hive
- Arquitectura Hive
- HiveQL y Almacenamiento
- UDF y MapReduce
- Particiones y cubos
- **Navegando Hive con Excel**
- Estrategias de Join
- Evolución Hive: Tez y LLAP

204

Conecotor ODBC HDInsight

- Conecotor para HDInsight Apache Hadoop Hive
- Soporte de 64/32 Bits
- Nos posibilita Business Intelligence
 - Excel
 - Analysis Services
 - PowerPivot
 - Reporting Services
 -

205

Demostración



Demo, Consultar tablas Hive desde Power BI

206

Agenda

- Introducción a Hive
- Arquitectura Hive
- HiveQL y Almacenamiento
- UDF y MapReduce
- Particiones y cubos
- Navegando Hive con Excel
- **Estrategias de Join**
- Evolución Hive: Tez y LLAP

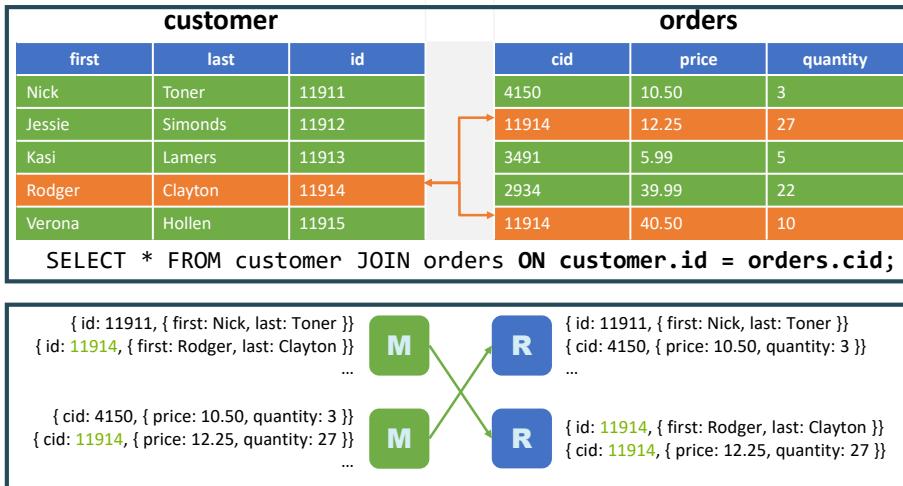
207

Estrategias de Join

Tipo	Aproximación	Pros	Cons
Shuffle Join	Las claves de Join se “barajan” usando MapReduce y los joins se hacen en el Reduce	Funciona independientemente del tamaño o estructura de datos	Más recursos consume y más lento
Map (Broadcast) Join	Se cargan las tablas pequeñas en memoria en todos los nodos, mapper escanea a través de la tabla más grande y hace el join	Muy rápido, un solo escaneo de la tabla más grande	Todas las tablas menos una deben de ser lo suficientemente pequeñas como para caber en RAM
Sort-Merge-Bucket Join	Mappers se benefician de la co-localización de claves para hacer más eficiente el join	Muy rápido para tablas de cualquier tamaño	Datos deben de estar ordenados y clusterizados anteriormente

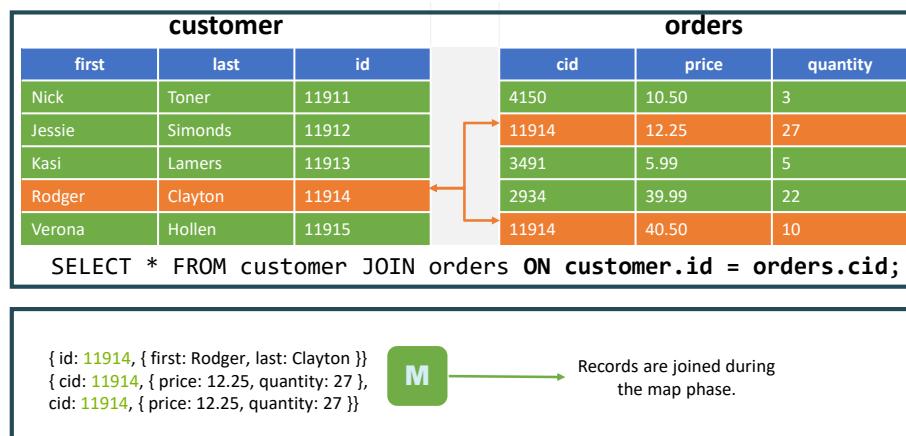
208

Shuffle Joins



209

Map (Broadcast) Join



210

Sort-Merge-Bucket Joins

customer			orders		
first	last	id	cid	price	quantity
Nick	Toner	11911	4150	10.50	3
Jessie	Simonds	11912	11914	12.25	27
Kasi	Lamers	11913	11914	40.50	10
Rodger	Clayton	11914	12337	39.99	22
Verona	Hollen	11915	15912	40.50	10

```
SELECT * FROM customer join orders ON customer.id = orders.cid;
```

Distribute and sort by the most common join key.

```
CREATE TABLE orders (cid int, price float, quantity int)
CLUSTERED BY(cid) SORTED BY(cid) INTO 32 BUCKETS;
```

```
CREATE TABLE customer (id int, first string, last string)
CLUSTERED BY(id) SORTED BY(cid) INTO 32 BUCKETS;
```

211

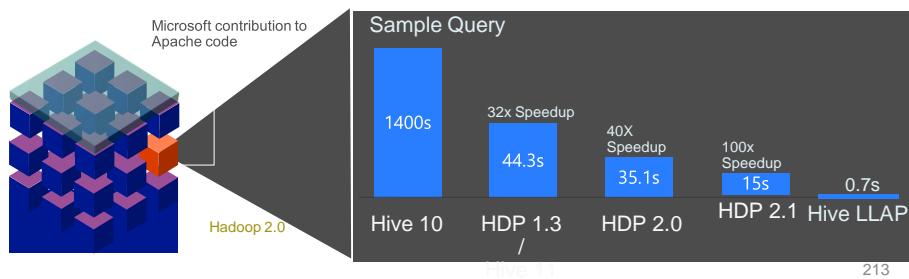
Agenda

- Introducción a Hive
- Arquitectura Hive
- HiveQL y Almacenamiento
- UDF y MapReduce
- Particiones y cubos
- Navegando Hive con Excel
- Estrategias de Join
- **Evolución Hive: Tez y LLAP**

212

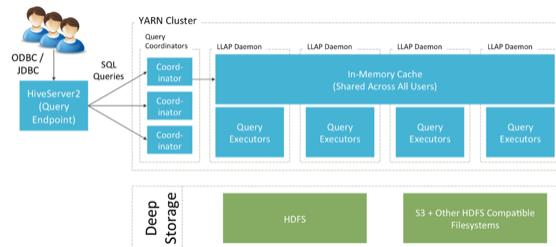
La iniciativa Stinger / TEZ

- Construido sobre YARN
- Fast Interactive Query
 - Mejoras de 100x de rendimiento para HIVE
 - Mejora HIVEQL parar proporcionar compatibilidad SQL



Hive LLAP

- Caché en Memoria



LAB



Trabajando con Hive

215

Arquitectura de Apache SPARK



Agenda

- **La motivación y nacimiento de SPARK**
- Casos de Uso
- Modos de Despliegue
- Ejecución Física

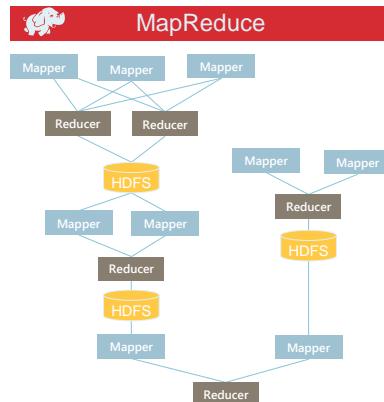
217

Motores de procesamiento
distribuido de datos para
propósito general

Map Reduce

Ventajas

- Framework simple para computación distribuida
- Toda computación debe expresarse como una serie de pasos definidos por dos Operaciones simples:
 - *Mappers*
 - *Reducers*
- Trabajos en batch que pueden expresarse como un data flow simple
- Solidez y Tolerancia a fallos se consiguen por la naturaleza del gráfico acíclico del flujo:
 - Cada ronda de map-reduce es independiente de otras y puede reconstruirse si se pierde
 - La comunicación se gestiona a través de los extremos del flujo



219

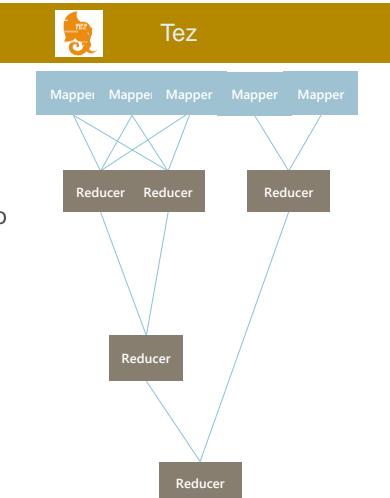
Map Reduce

Desventajas

- Cada paso reduce require escribir a disco
- Para trabajos iterativos, como modelos de Machine Learning, una implementación MapReduce tendría multiples reducers
- No tiene vision global para optimizar el flujo

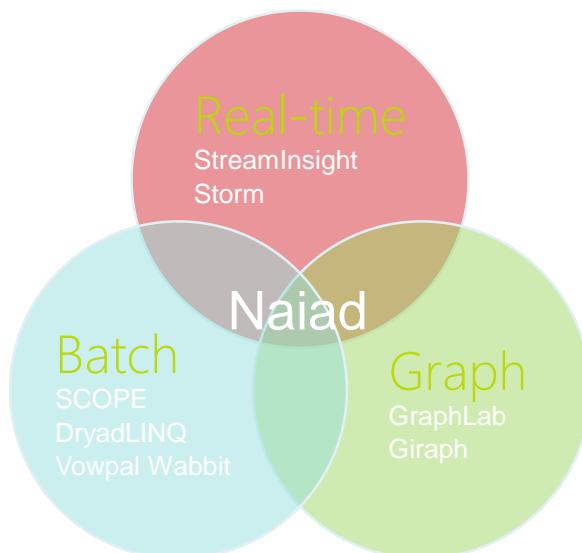
Mejoras propuestas

- Stinger/Tez
- Dryad
- Naiad:
 - *Cyclical differential dataflow model*
- Apache Spark



220

Resultado....



221

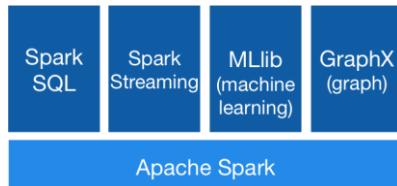
El nacimiento de Apache SPARK

- Inspirado directamente por el modelo de flujo de datos cíclico de Naiad y Dryad
- Se inició como un Proyecto de investigación en el AMPLab en UC Berkeley en 2009
- Open Source License (Apache 2.0)
- Es el Proyecto Apache más activo (800+ developers)



222

Framework Unificado



- **Unifica:**
 - Procesado Batch
 - Procesado Tiempo Real
 - Stream Analytics
 - Machine Learning
 - SQL Interactivo

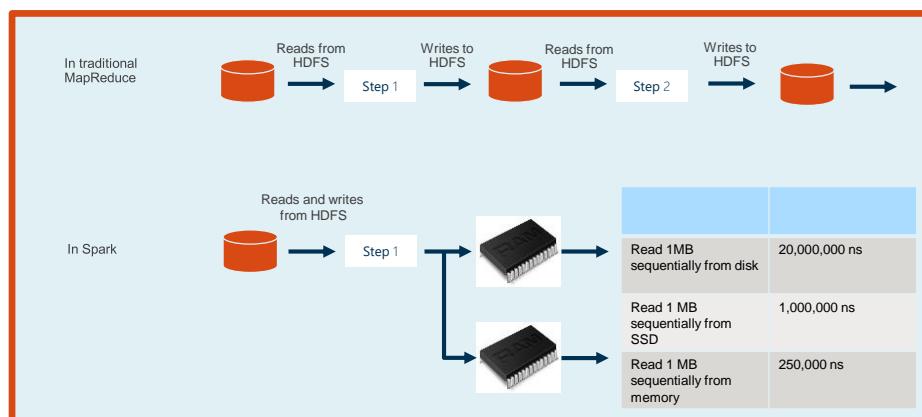
Objetivo: motor unificado entre orígenes de datos, cargas de trabajo y entornos

APIs de desarrollo simplificadas: Scala, Java, Python, R

223

Spark vs. Map Reduce

Datos en memoria compartidos entre los pasos del trabajo



224

Proceso de Datos

- Al contrario que MapReduce el motor de procesado de datos de Spark no está limitado a mappers y reducers
- Un trabajo Spark puede tener múltiples estados, cada uno de ellos compuesto por varias tareas (“mappers” y “reducers”)
- Los datos se meten en caché y se reutilizan entre iteraciones, reduciendo I/O
- Desarrollado como un motor de procesado general para varias cargas de trabajo

225

RDD(Resilient Distributed Dataset)

- Distribuido entre los “workers” de Spark
- Inmutables
 - Si se transforma se genera una nuevo
- Evaluación “perezosa”
 - No se ejecuta en tiempo de definición, sino cuando se evalúe aplicando una acción sobre el RDD
- Para crear un RDD
 - Paralelizando.
 - val newRDD= sc.parallelize(List(1, 2, 3, 4))
 - A partir de una fuente de almacenamiento
 - val newRDD: RDD[Int] = sc.textFile("myValues.txt")
 - Transformando un RDD
 - val newRDD: RDD[String] = intValues.map(_.toString)

226

Transformaciones sobre RDD

- **map:** aplica una función a cada elemento de la colección:
 - `intValues.map(_.toString) // RDD[String]`
- **filter:** selecciona el subconjunto de elementos que cumplen una determinada expresión booleana:
 - `intValues.filter(_.isOdd)// RDD[Int]`
- **flatMap:** además de realizar una función map, aplica un método flatten:
 - `textFile.map(_.split(" ")) //RDD[Array[String]]`
 - `textFile.flatMap(_.split(" ")) //RDD[String]`

227

Acciones sobre RDD

- **count:** nos devuelve el número total de elementos:
 - `sc.parallelize(List(1, 2, 3, 4)).count //4`
- **collect:** nos vuelca toda la colección distribuida en un array en memoria:
 - `sc.parallelize(List(1, 2, 3, 4)).collect // Array(1, 2, 3, 4)`
 - Ojo, cuidao. Si el RDD es muy grande, podemos tener problemas al volcar toda la colección en memoria.
- **saveAsTextFile:** nos vuelca la información en un fichero de texto:
 - `intValues.saveAsTextFile("results.txt")`

228

Agenda

- La motivación y nacimiento de SPARK
- **Casos de Uso**
- Modos de Despliegue
- Ejecución Física

229

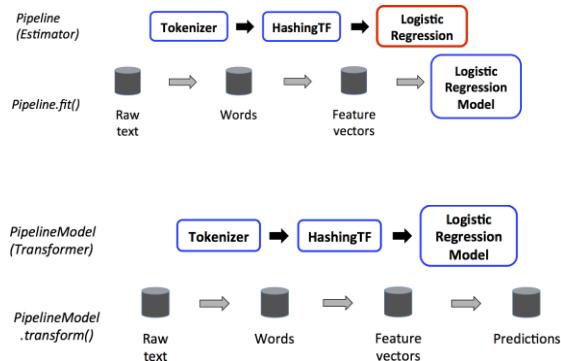
Casos de Uso

- Análisis interactivo de datos y BI
- Machine Learning con Spark
- Streaming y Análisis de datos en tiempo real con Spark
- Deep Learning

230

Spark Machine Learning

- MLlib



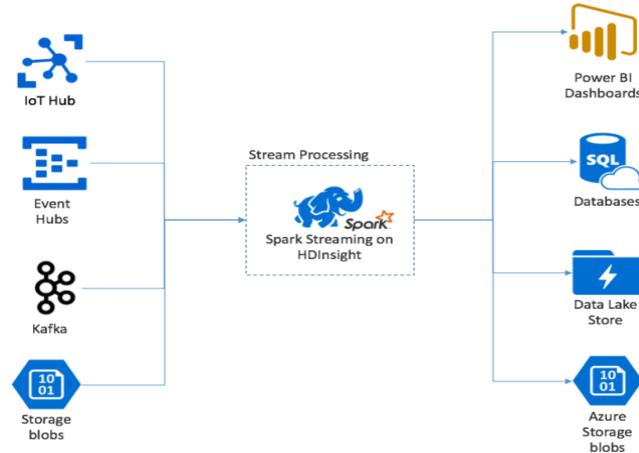
231

Apache Spark Streaming



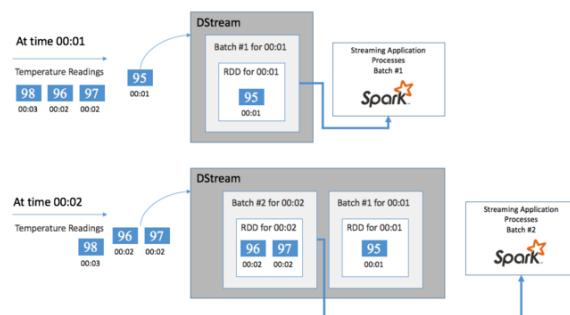
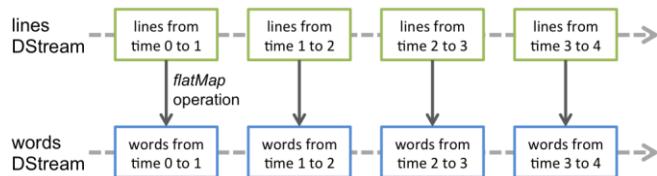
232

Spark Streaming en HDInsight



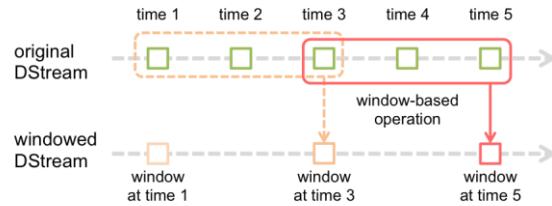
233

Dstreams (Discretized Streams)



234

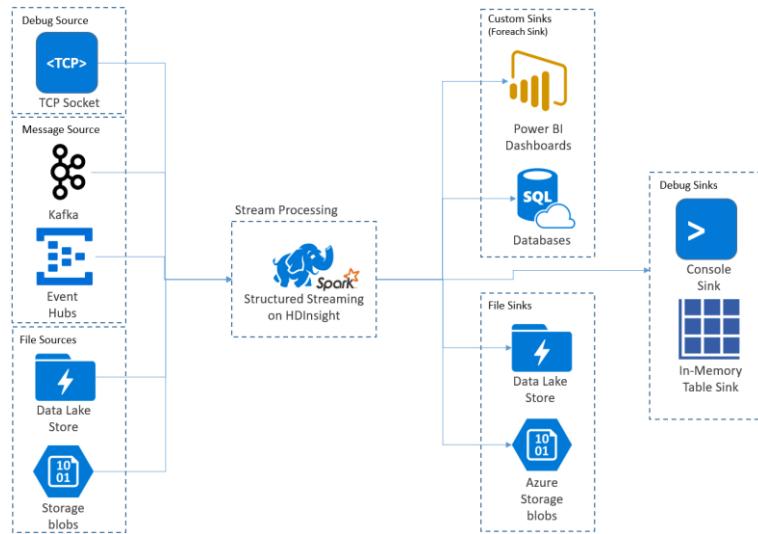
Window



- *window length* – La duración de la Ventana (3 en la figura)
- *sliding interval* – El intervalo en el que la operación window se realiza (2 en la figura)

235

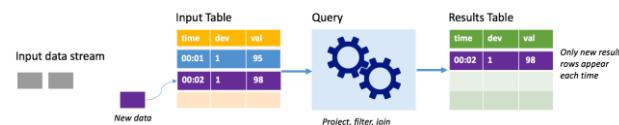
Structured Streaming



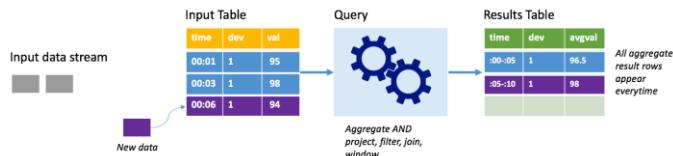
236

Streams como tablas

- Modo Append



- Modo Completo



237

Comparativa

	Apache Spark Streaming	Apache Apex	Apache Beam	Apache Flink	Apache Kafka Streams	Akka Streams
Project	 Spark Streaming	 APEx	 beam	 Flink	 kafka	 akka
Version Tested	2.10	3.6.0	2.0.0	1.3.1	0.10.0.0	0.17
Easy to find examples?	Easy	Medium/Hard	Easy	Easy	Medium/Hard	
Reference Documentation	Streaming Programming Guide	Apex Documentation - Maher Documentation	Apache Beam Documentation	Apache Flink Documentation	Apache Kafka Stream Documentation	Akka Streams documentation
Implementation Difficulty	Easy	Easy	Easy	Easy	Easy (with high level API)	Medium (some Akka knowledge is helpful)
Paradigm	RDD	DAG of Operators	Pipelines	DAG of Operators	Topology (DAG) High level API KStream API Low level API: Source/Stream/Sink Processors	Actor model
Graphical Designer	No (Seahorse for Spark, not Streaming)	Yes (dAssemble)	No	No (only visualise)	No	No
Optimization	Optimization of DAG & SQL queries	DAG Enhancement	DAG Enhancement	DAG Enhancement	DAG Enhancement	Future materialization model allows it
Execution	Batch with memory pinning	Event-driven	Runner-dependent	Event-driven	Commitlog (Kafka Cluster)	Event-driven
Parallelization	Container-based	Container-based	Runner-dependent	Container-based	Kafka Partition / Consumer Groups	Actor-based
Documentation	Good	Apex - Medium Mahar - Medium/Weak	Good	Good	Good	Good / Heavy
Best Practices	Available	Not much	Some	Some	Some	Not much
Main backer	Databricks	DataTorrent	Google	Apache	Confluent	Lightbend (Ra Typesafe)

238

Comparativa

Apache Spark Streaming <p>Es una extensión de la API central de Spark que permite el procesamiento escalable, de alto rendimiento y tolerante a fallos de las transmisiones de datos en directo.</p>	Apache Apex <p>Es una plataforma nativa de Hadoop YARN que unifica el procesamiento streaming y por lotes en un mismo framework. Procesa big data en movimiento de una manera que es altamente escalable, de alto rendimiento, tolerante a fallos, de firewall con estado, seguro, distribuido y fácilmente operable. Añadiendo a esto, que mantiene Apache Mathur, una librería extensible de operadores para reutilizar en Apex.</p>
Apache Beam <p>Proporciona un avanzado modelo unificado de programación, que permite implementar trabajos tanto de procesamiento de datos por lotes como de streaming, con la particularidad de poder elegir el motor de ejecución: Beam es un SDK que requiere un runner por debajo (por ejemplo, Apex, Flink, Spark o Google Cloud DataFlow).</p>	Apache Flink <p>Es un framework de procesamiento de streams de código abierto para aplicaciones de transmisión de datos distribuidas, de alto rendimiento, siempre disponibles y precisas.</p>
Apache Kafka Streams <p>Es un componente de código abierto de Apache Kafka. Es una biblioteca potente y fácil de usar para construir aplicaciones de procesamiento de flujo distribuido altamente escalables y tolerantes a fallos de Apache Kafka.</p>	Akka Streams <p>Proporciona una abstracción de alto nivel sobre el modelo de actores ya existente de Akka. El modelo de actores ofrece una excelente oportunidad para escribir software simultáneo y escalable, pero sigue estando a bajo nivel. Entonces, ¿es posible hacer ambas cosas a la vez? ¡Podemos simplificar la funcionalidad que queremos lograr con actores en un conjunto de llamadas a funciones?</p> <p>¡Podemos tratar los mensajes de actores como entradas y salidas de funciones con type-safe! Akka-Streams nos lo permite.</p>

239

Comparativa

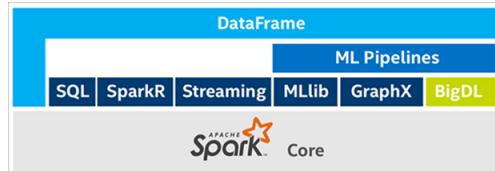
Data Practice – Streaming Technologies Cheat Sheet

Apache Spark Streaming <pre>Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.</pre> <p>Read from File</p> <pre>SparkConf conf = new SparkConf().setAppName("StreamingWordCount").setMaster("local[2]"); JavaDStream<String> lines = JavaSparkContext.textFile("hdfs://localhost:9000/README.txt"); lines.foreachRDD(rdd => rdd.mapPartitions(partition => partition.filter(_.nonEmpty).map(_.toLowerCase().split(" ")).flatMap(_.map(word => (word, 1))).groupByKey().mapValues(_.sum)))</pre> <p>Process Metrics</p> <pre>lines.foreachRDD(rdd => rdd.mapPartitions(partition => partition.filter(_.nonEmpty).map(_.toLowerCase().split(" ")).flatMap(_.map(word => (word, 1))).groupByKey().mapValues(_.sum)))</pre> <p>Save to HDFS/S3</p> <pre>val wordCounts = lines.mapPartitions(partition => partition.filter(_.nonEmpty).map(_.toLowerCase().split(" ")).flatMap(_.map(word => (word, 1))).groupByKey().mapValues(_.sum))</pre>	Apache Apex <p>Apex is a Hadoop YARN native platform that unifies stream and batch processing. It provides a simple API that enables developers to write or re-use generic java code. The Apex platform comes with Mathur, a library of operators (units of functionality).</p> <p>Read from File</p> <pre>FileInputFormat.setInputPath(job, "/user/hadoop/input"); job.setInputFormat(new TextInputFormat()); job.setOutputFormat(new TextOutputFormat()); job.setMapperClass(TextMapper.class); job.setReducerClass(TextReducer.class); job.setMapOutputKeyClass(Text.class); job.setMapOutputValueClass(Text.class); job.setPartitionerClass(HashPartitioner.class); job.setNumReduceTasks(1); job.setJarByClass(TextJob.class);</pre> <p>Process Metrics</p> <pre>FileInputFormat.setInputPath(job, "/user/hadoop/input"); job.setInputFormat(new TextInputFormat()); job.setOutputFormat(new TextOutputFormat()); job.setMapperClass(TextMapper.class); job.setReducerClass(TextReducer.class); job.setMapOutputKeyClass(Text.class); job.setMapOutputValueClass(Text.class); job.setPartitionerClass(HashPartitioner.class); job.setNumReduceTasks(1); job.setJarByClass(TextJob.class);</pre> <p>Write to HDFS</p> <pre>FileOutputFormat.setOutputPath(job, "/user/hadoop/output"); job.setOutputFormat(new TextOutputFormat()); job.setMapperClass(TextMapper.class); job.setReducerClass(TextReducer.class); job.setMapOutputKeyClass(Text.class); job.setMapOutputValueClass(Text.class); job.setPartitionerClass(HashPartitioner.class); job.setNumReduceTasks(1); job.setJarByClass(TextJob.class);</pre>
Apache Beam <p>Apache Beam provides an advanced unified programming model, allowing you to implement batch and streaming data processing jobs that can run on any execution engine. Beam is an SDK that requires a runner (e.g. Apex, Flink, Spark, or Google Cloud DataFlow).</p> <p>Read from File</p> <pre>Pipeline pipeline = Pipeline.create(); pipeline.readTextFile("file:///tmp/words.txt") .apply("Map", ParDo.of(new MapFn<String, String>() { @Override public Tuple2<String, String> process(String value) throws Exception { return new Tuple2<String, String>(value, "1"); } })) .apply("Combine", Combine.perKey()); .apply("Write", TextOutputFormat.writeText("file:///tmp/counts.txt")); .run();</pre> <p>Process Metrics</p> <pre>Pipeline pipeline = Pipeline.create(); pipeline.readTextFile("file:///tmp/words.txt") .apply("Map", ParDo.of(new MapFn<String, String>() { @Override public Tuple2<String, String> process(String value) throws Exception { return new Tuple2<String, String>(value, "1"); } })) .apply("Combine", Combine.perKey()); .apply("Write", TextOutputFormat.writeText("file:///tmp/counts.txt")); .run();</pre>	Flink and accurate data streaming applications <p>Read from File</p> <pre>FileInputFormat.setInputPath(job, "/user/hadoop/input"); job.setInputFormat(new TextInputFormat()); job.setOutputFormat(new TextOutputFormat()); job.setMapperClass(TextMapper.class); job.setReducerClass(TextReducer.class); job.setMapOutputKeyClass(Text.class); job.setMapOutputValueClass(Text.class); job.setPartitionerClass(HashPartitioner.class); job.setNumReduceTasks(1); job.setJarByClass(TextJob.class);</pre> <p>Process Metrics</p> <pre>FileInputFormat.setInputPath(job, "/user/hadoop/input"); job.setInputFormat(new TextInputFormat()); job.setOutputFormat(new TextOutputFormat()); job.setMapperClass(TextMapper.class); job.setReducerClass(TextReducer.class); job.setMapOutputKeyClass(Text.class); job.setMapOutputValueClass(Text.class); job.setPartitionerClass(HashPartitioner.class); job.setNumReduceTasks(1); job.setJarByClass(TextJob.class);</pre>
Apache Kafka <p>Kafka Streams is a component of open source Apache Kafka. It is a powerful, easy-to-use library for building highly scalable, fault-tolerant, distributed stream processing applications on top of Apache Kafka</p> <p>Read from Kafka</p> <pre>Properties props = new Properties(); props.put("bootstrap.servers", "localhost:9092"); props.put("group.id", "my-group"); props.put("enable.auto.commit", "true"); props.put("auto.commit.interval.ms", "1000"); KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(props); consumer.subscribe(Arrays.asList("my-topic")); while(true) { ConsumerRecords<String, String> records = consumer.poll(100); for(ConsumerRecord<String, String> record : records) { System.out.printf("offset = %d, key = %s, value = %s\n", record.offset(), record.key(), record.value()); } }</pre> <p>Process Metrics</p> <pre>Properties props = new Properties(); props.put("bootstrap.servers", "localhost:9092"); props.put("group.id", "my-group"); props.put("enable.auto.commit", "true"); props.put("auto.commit.interval.ms", "1000"); KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(props); consumer.subscribe(Arrays.asList("my-topic")); while(true) { ConsumerRecords<String, String> records = consumer.poll(100); for(ConsumerRecord<String, String> record : records) { System.out.printf("offset = %d, key = %s, value = %s\n", record.offset(), record.key(), record.value()); } }</pre>	Akka Streams <p>Akka Streams provide a higher-level abstraction over Akka's existing actor model. An implementation of Reactive Streams, it comes to solve some of Akka's model problems with streams, like backpressure or semantic guarantees.</p> <p>Read from File</p> <pre>ReadFromKafka.readFrom("file:///tmp/words.txt", ActorMaterializer.create(system)) .map { words => words.map(_.toLowerCase().split(" ")).flatMap(_.map(word => (word, 1))).groupByKey().mapValues(_.sum) } .to(Sink.ignore) .run();</pre> <p>Process Metrics</p> <pre>ReadFromKafka.readFrom("file:///tmp/words.txt", ActorMaterializer.create(system)) .map { words => words.map(_.toLowerCase().split(" ")).flatMap(_.map(word => (word, 1))).groupByKey().mapValues(_.sum) } .to(Sink.ignore) .run();</pre>

240

Deep Learning

- Soporte Enriquecido de DL
 - Tensor
 - Redes Neuronales
 - Carga de Modelos Caffe, Torch, Keras
- Muy Alto Rendimiento -> Intel MKL
- Escalado



241

Demo Spark



Casos de Uso

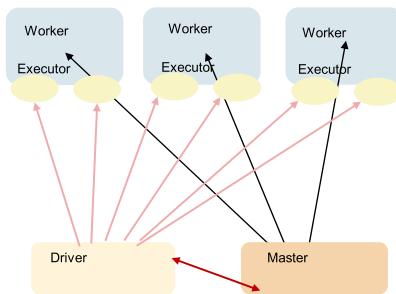
242

Agenda

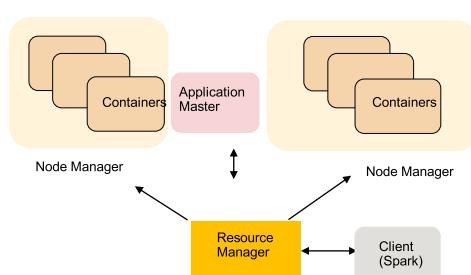
- La motivación y nacimiento de SPARK
- Casos de Uso
- **Modos de Despliegue**
- Ejecución Física

243

Modos de Despliegue



Spark Standalone

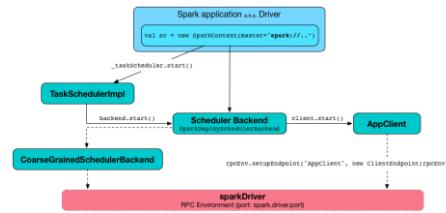


Spark en YARN

244

Standalone

- En este modo Spark usa un *daemon* Master para coordinar con los *workers* que ejecutan los *executors*
- Es el modo predeterminado
- Al lanzar la aplicación podemos decidir cuando memoria utilizarán los *executors*, así como el número total de cores



245

Modo YARN

- El ResourceManager de YARN juega el rol de Spark Master
- Los NodeManagers de YARN actúan como *executors*
- Más complejo pero más seguro
- Podemos ejecutarlo en dos modos
 - Cliente - interactivo
 - Cluster - background

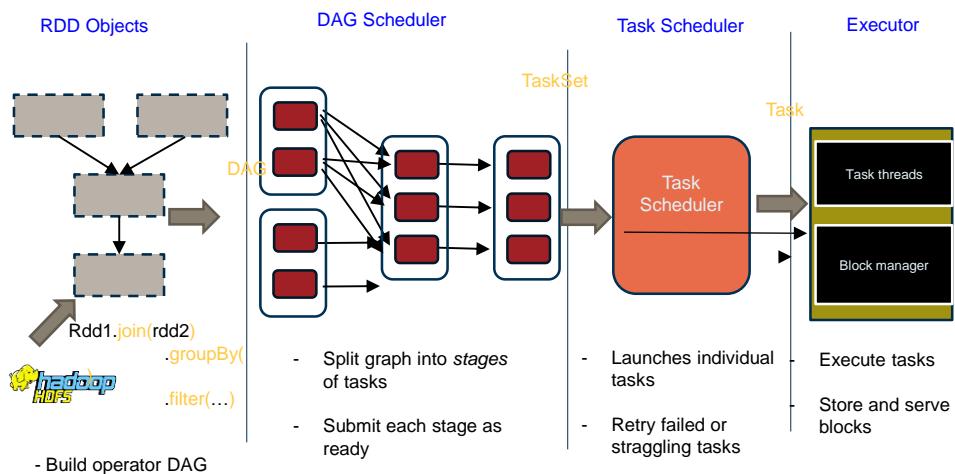
246

Agenda

- La motivación y nacimiento de SPARK
- Casos de Uso
- Modos de Despliegue
- **Ejecución Física**

247

Proceso de Schedule



248

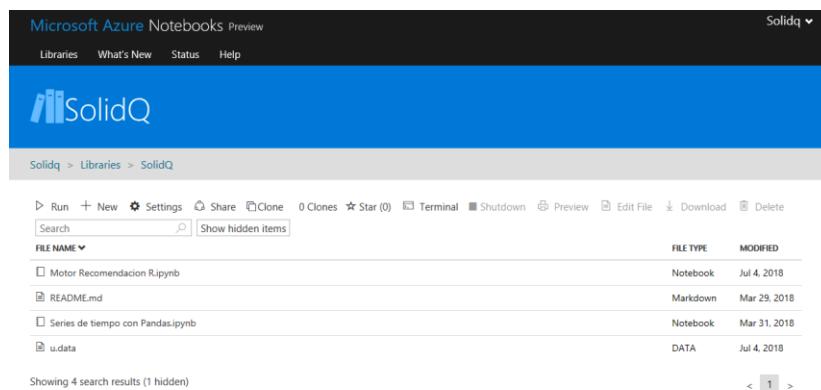
Notebooks



- REPL – Read-Evaluate-Print-Loop
- Prototipo, desarrollo rápido, exploración,...
- Colaboración en equipos

249

Azure Notebooks



The screenshot shows the Microsoft Azure Notebooks interface. At the top, there's a navigation bar with 'Microsoft Azure Notebooks Preview' and a 'SolidQ' dropdown. Below it is a blue header bar with the 'SolidQ' logo. The main area displays a search result for 'SolidQ'. The search bar at the top has 'Search' and 'Show hidden items' buttons. Below the search bar is a table with four rows of results:

FILE NAME	FILE TYPE	MODIFIED
Motor Recomendacion R.ipynb	Notebook	Jul 4, 2018
README.md	Markdown	Mar 29, 2018
Series de tiempo con Pandas.ipynb	Notebook	Mar 31, 2018
u.data	DATA	Jul 4, 2018

At the bottom of the table, it says 'Showing 4 search results (1 hidden)'. There are navigation arrows for the search results.

250

Laboratorio



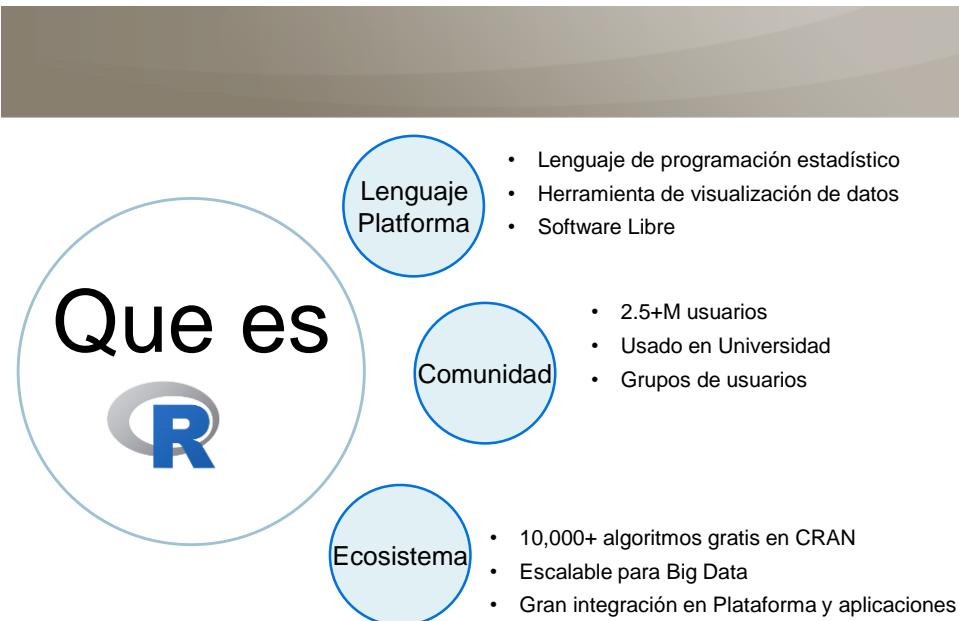
Creando un cluster SPARK
Consultas interactivas
Machine Learning

251

Agenda

- Distribuciones de R
- Modelos de ejecución

253



254

Distribuciones de R


[CRAN-R](#)

[MRO](#)

[R Client](#)

- a.k.a., GNU-R
- Single-threaded
- In-memory
- 10K+ packages
- Interfaces to C++, C, and Fortran for speed
- Cross-platform

- GNU-R + Intel MKL = multi-threaded linear algebra
- 100% CRAN-R compatible
- Reproducible R toolkit with the checkpoint package
- Open-source

- High-performance analytics with the RevoScaleR package
- Two-threads
- Deployment capabilities through the `mrsdeploy` package
- MicrosoftML
- Free, windows

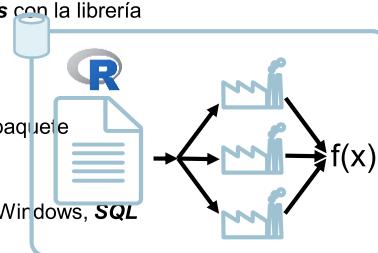
Distribuciones en memoria

255

Distribuciones de R

[Microsoft R Server](#)

- Distribución empresarial multicore, multi-node, algoritmos paralelos
- **Algoritmos fuera de memoria paralelos** con la librería RevoScaleR
 - Sin límite de hilos
- **Formato de fichero XDF**
- Capacidades de despliegue a través del paquete `mrsdeploy`
- *Soporte comercial*
- Disponible en Linux (Red Hat, CentOS), Windows, **SQL Server, Spark, Hadoop y Teradata**
- Librería MicrosoftML



256

Entornos

- Microsoft R Client
- R Studio
- Visual Studio R Tools



257

Demo



Entornos R

258

Agenda

- Distribuciones de R
- Modelos de ejecución

259

Contextos

- Estación de trabajo
 - Todos los cores se utilizan para operaciones matemáticas y procesos paralelos
 - El tamaño de datos viene limitado por la capacidad del disco no la memoria
- En cluster
 - Utilización paralela de todos los nodos disponibles
 - Sistemas de archivo distribuidos como HDFS aumenta los posibles tamaños de datos
- Reutilización del código con un simple cambio de función de llamada

```
# equipo local:  
rxSetComputeContext( LocalParallel() )  
  
# cluster:  
rxSetComputeContext( RxSpark(...) )
```

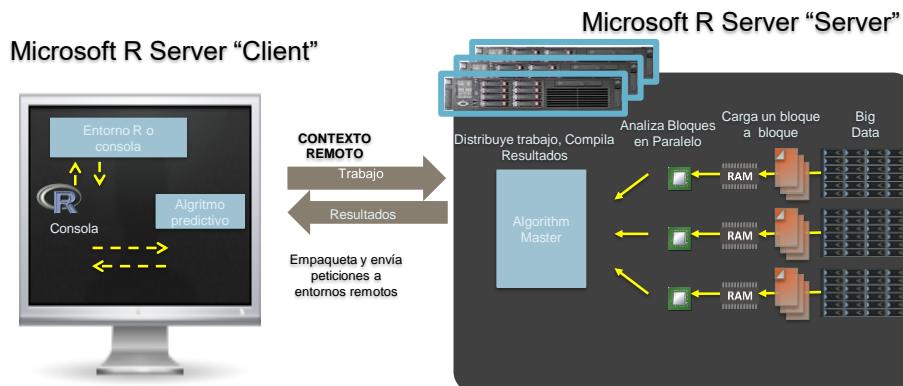
260

Funcionamiento de R Server

- Parallel External Memory Algorithms (PEMAs)
 1. Se extrae un subconjunto de datos del dataset original
 2. Se calcula un resultado intermedio de ese subconjunto
 3. Se combinan los resultados intermedios en un dataset final

261

Contexto de ejecución remoto



262

XDF: Formato de datos nativo

- eXternal Data Frame file (**XDF**)
- Orientado a “trozos”
 - Fácil de distribuir a nodos
 - Rápido para anexar
- Orientado a columnas
 - Rápida obtención de variables
- Metadatos pre-calculados

263

Importación a XDF

- **rxImport**
 - **inData**
 - **outFile**
 - **varsToKeep, varsToDelete**
 - **numRows**
 - **rowSelection**
 - **overwrite**
 - **append**

264

R Server en Hadoop

- R Server soporta Hadoop MapReduce como un contexto de computación Hadoop
- Esto significa que las funciones de R Server se ejecutan en Hadoop como un trabajo MapReduce en YARN
- Lento por el overhead de arranque y las comunicaciones basadas en ficheros

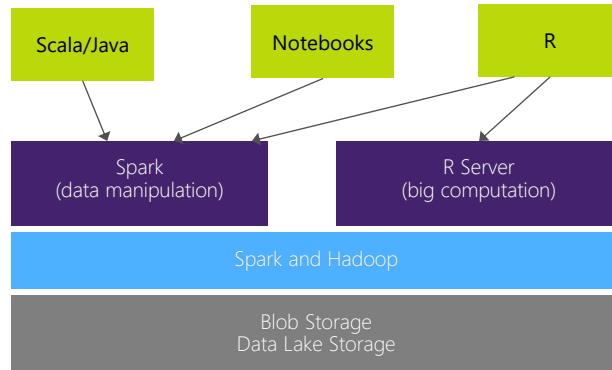
265

R Server en Spark

- Spark permite retener los datos en memoria entre iteraciones o pipelines de ML a través de Resilient Distributed Datasets (RDD) y Spark Dataframes
- R Server en Spark no tiene la sobrecarga de MapReduce
- Como funciona:
 - Cada función rx invoca una nueva aplicación spark que persiste durante la vida de la rx
 - XDFs se pasan a los nodos de trabajo como RDDs para cargar y retener entre iteraciones
 - Los datos en RDD van a Cache entre iteraciones
 - Los nodos de trabajo y el Maestro se comunican con el API Scala
 - Los archivos temporales se guardan en tmpfs (Sistema de ficheros en memoria de Linux)
 - Soporte modo persistente en el Contexto Spark para reducir la sobrecarga de lanzar el proceso

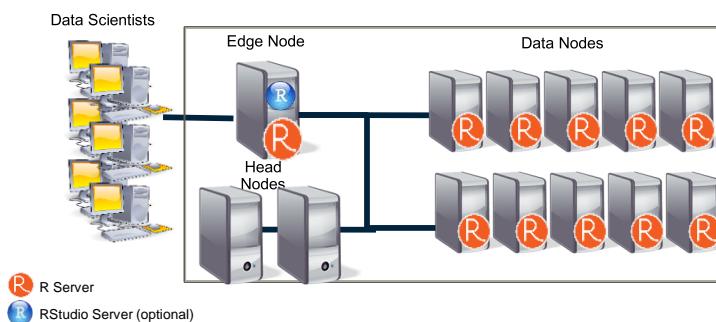
266

HDInsight + R Server



267

Arquitectura R Server en HDInsight



Procesado distribuido R Server:

- Proceso Master R en Edge node
- Apache YARN y Spark
- Procesos Worker R en nodos de Datos

268

APIs R para Spark

- *SparkR*
 - ASF, Apache-licensed
 - Ships with Apache-Spark since 1.5x
 - SparkSQL and SparkML support through RPC
 - UDF support through `gapply`, `dapply`, `spark.lapply`
- *sparklyr*
 - On CRAN/github
 - Apache-licensed
 - SparkSQL and SparkML support through `DBI`, `dplyr` and `RPC`
 - Limited UDF support through `do`
 - Remote execution with `Liyy`
- *RxSpark*
 - Commercially licensed
 - Deploy RevoScaleR in Spark clusters
 - PEMA support
 - Consume Spark DataFrames, Parquet, and Hive Tables
 - Robust UDF support with `rxDataStep`, `rxExec`, and `foreach`

269

RxSpark

- `RxSpark(`
`hdfsShareDir, localShare, clientShare, ssh,`
`ssh..., spark_config...,)`
- Puede desplegarse en remoto utilizando parámetros de conexión SSH
- Puede desplegarse usando el paquete *mrsdeploy*

270

Demo



Cluster R

271

Laboratorio



Desplegando un cluster R
Trabajando con R sobre Spark

272



www.solidq.com

info@solidq.com