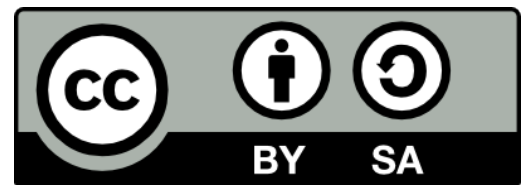
Big Data I:

Ingeniería de datos

Felipe Ortega
Dpto. de Estadística e Investigación Operativa
Universidad Rey Juan Carlos

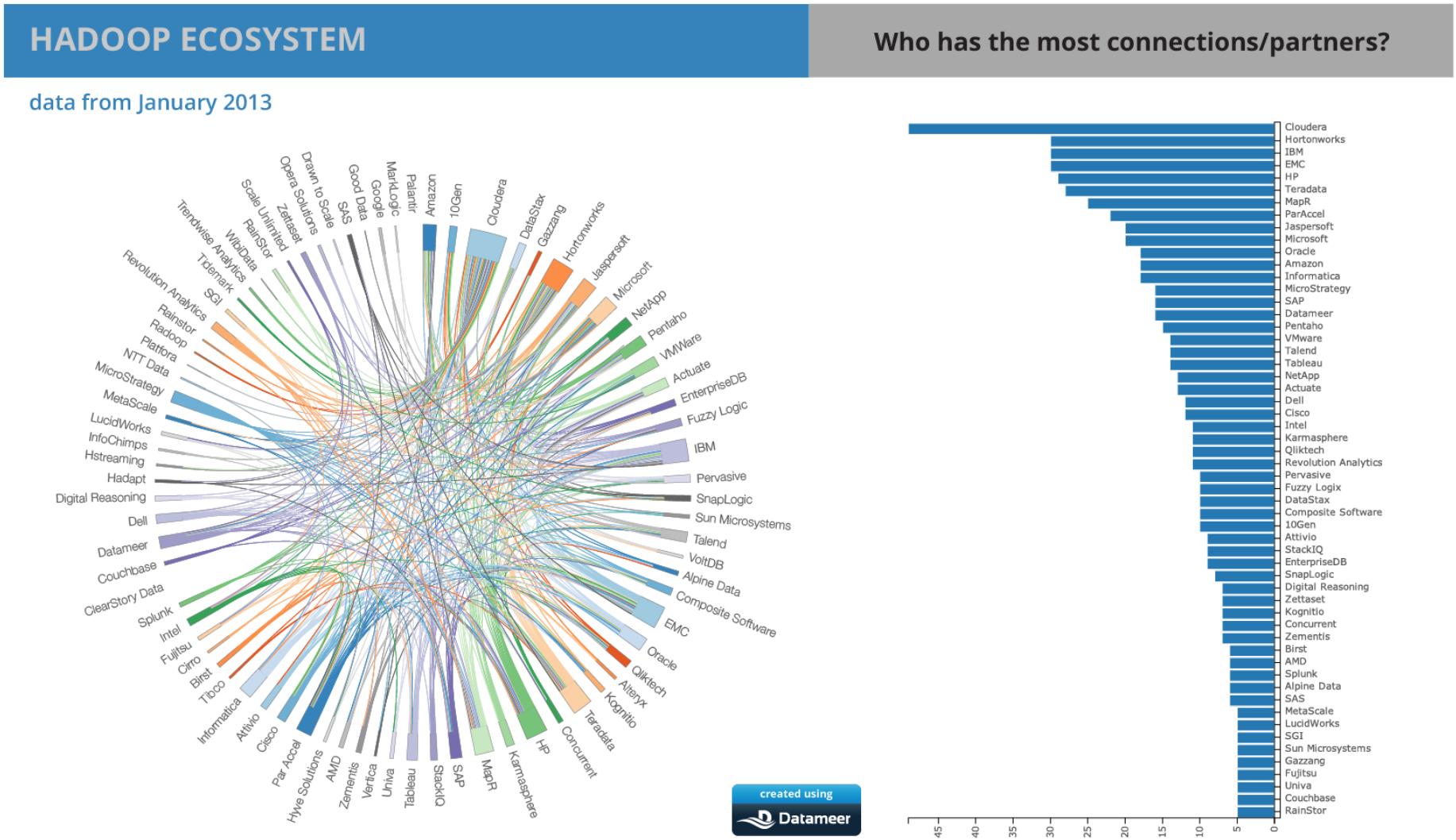
March 26, 2015

Universidad
Rey Juan Carlos

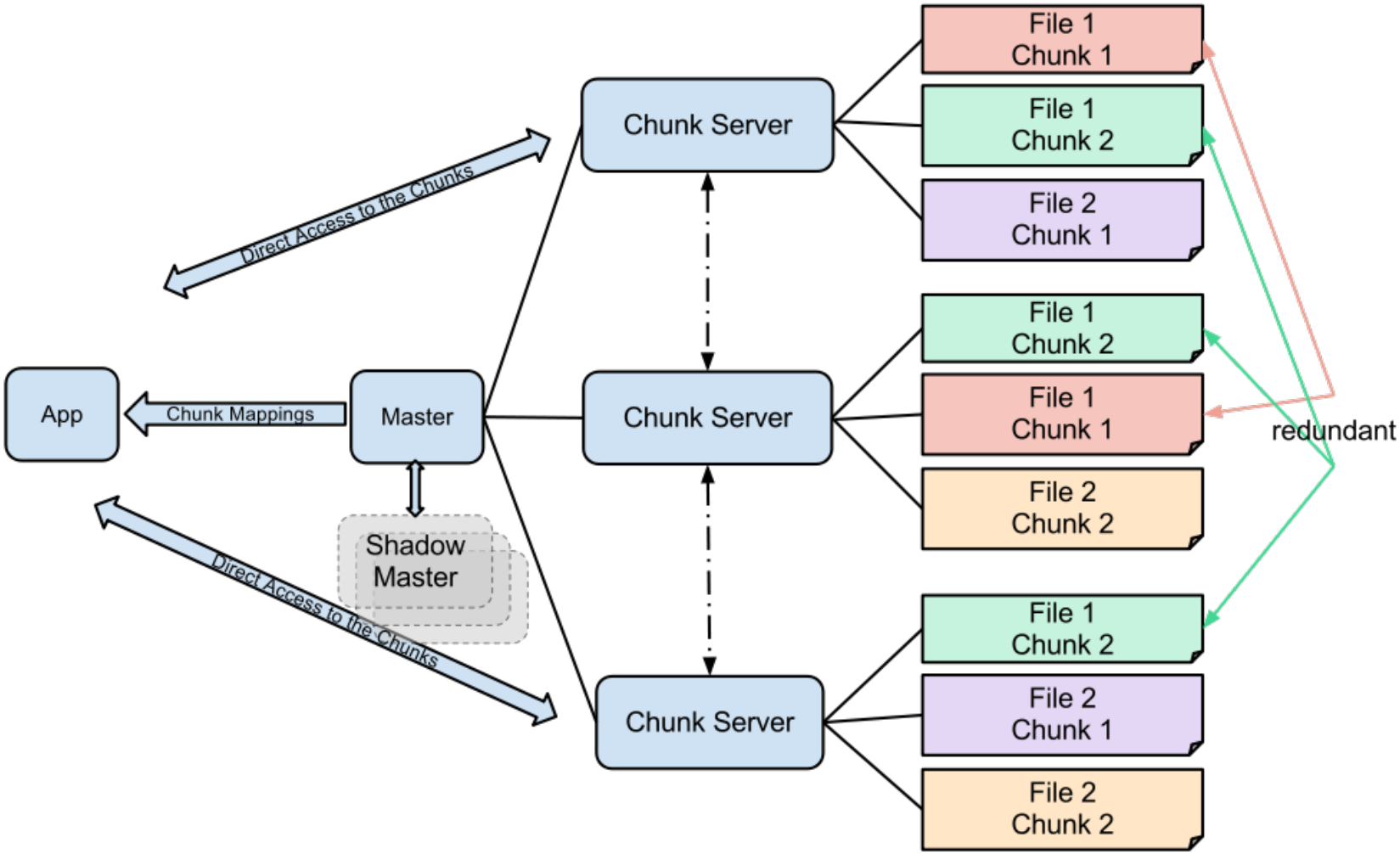


Ecosistema Apache Hadoop

Ecosistema Apache Hadoop



Google File System



HDFS

- Modelado a partir del Google File System [2].
- Optimizado para maximizar el throughput, mejor cuanto más grandes sean los archivos.
- Tamaño de bloques grande, intenta optimizar distribución de datos en nodos siguiendo principios de localidad espacial y temporal (similar a jerarquía de memoria).
- Las primeras versiones utilizaban un nodo para metadatos (*NameNode*) y varios nodos para almacenar y trabajar con los datos (*DataNodes*). En las versiones más modernas, datos y metadatos están distribuidos.

Almacenamiento en la nube

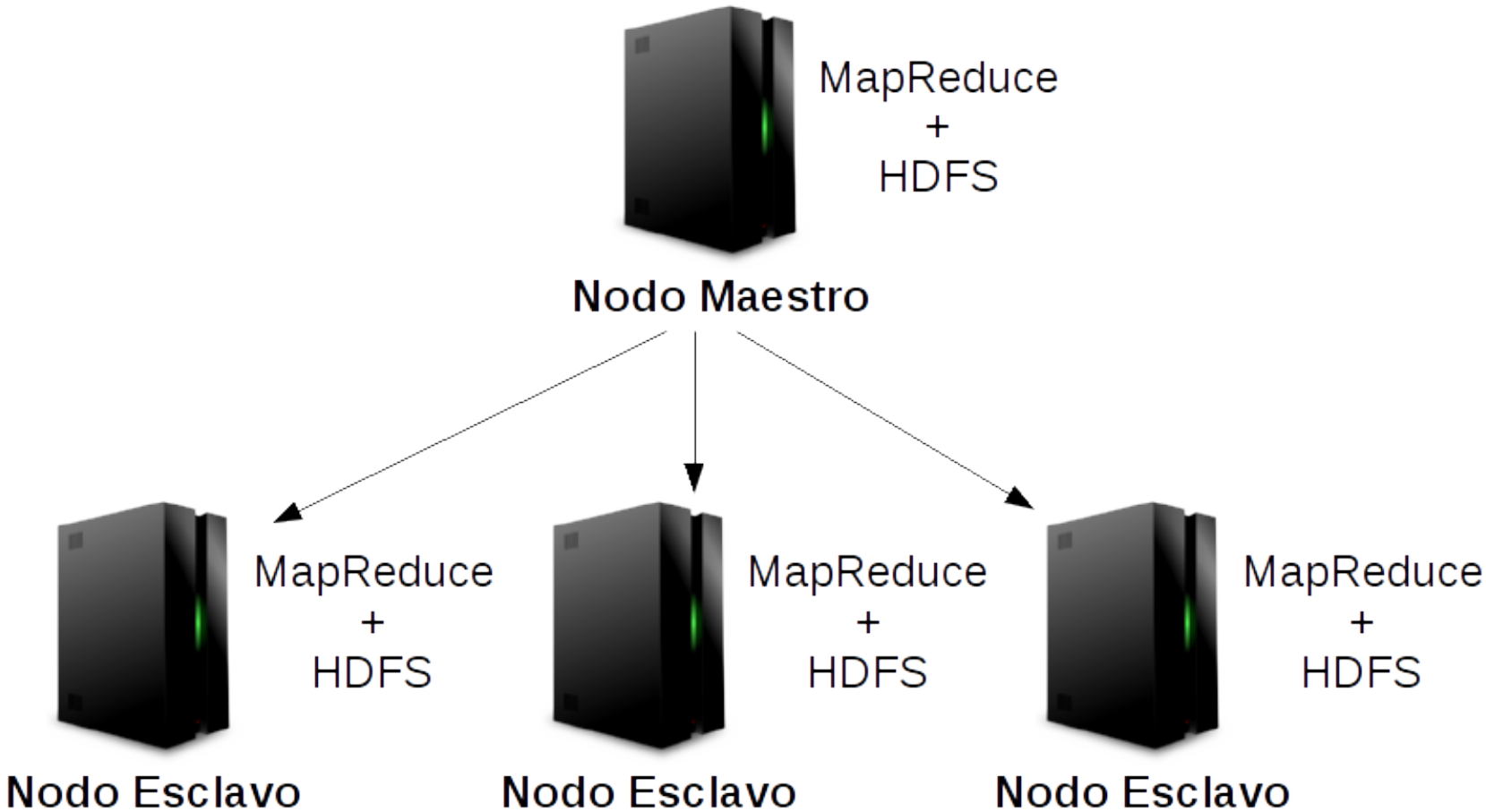
- Es posible poder ejecutar instancias Hadoop sobre servicios de computación en la nube.
- Ejemplo: servicios web de Amazon.
  - Podemos utilizar instancias de cómputo y almacenamiento de datos en Amazon EC2 para ejecutar nuestras tareas desde Hadoop (importación de recursos).
  - Adicionalmente, Amazon también proporciona Amazon Elastic MapReduce (EMR), un entorno de gestión alternativo similar a Hadoop, que permite gestionar las tareas y recursos de cluster.
  - EMR también puede albergar otros entornos para procesado de flujos de datos, como Apache Spark o Presto.
- Otros proveedores: Microsoft Azure, etc.

Proyectos Hadoop: Hadoop

- El núcleo de todo el ecosistema de aplicaciones.
- Hadoop Common Package (abstracciones) + HDFS (sistema de ficheros distribuido) + YARN (MapReduce engine).
- Se aplica el paradigma MapReduce a datos almacenados en múltiples nodos.
- Arquitectura Maestro-Esclavo (en su primera versión).

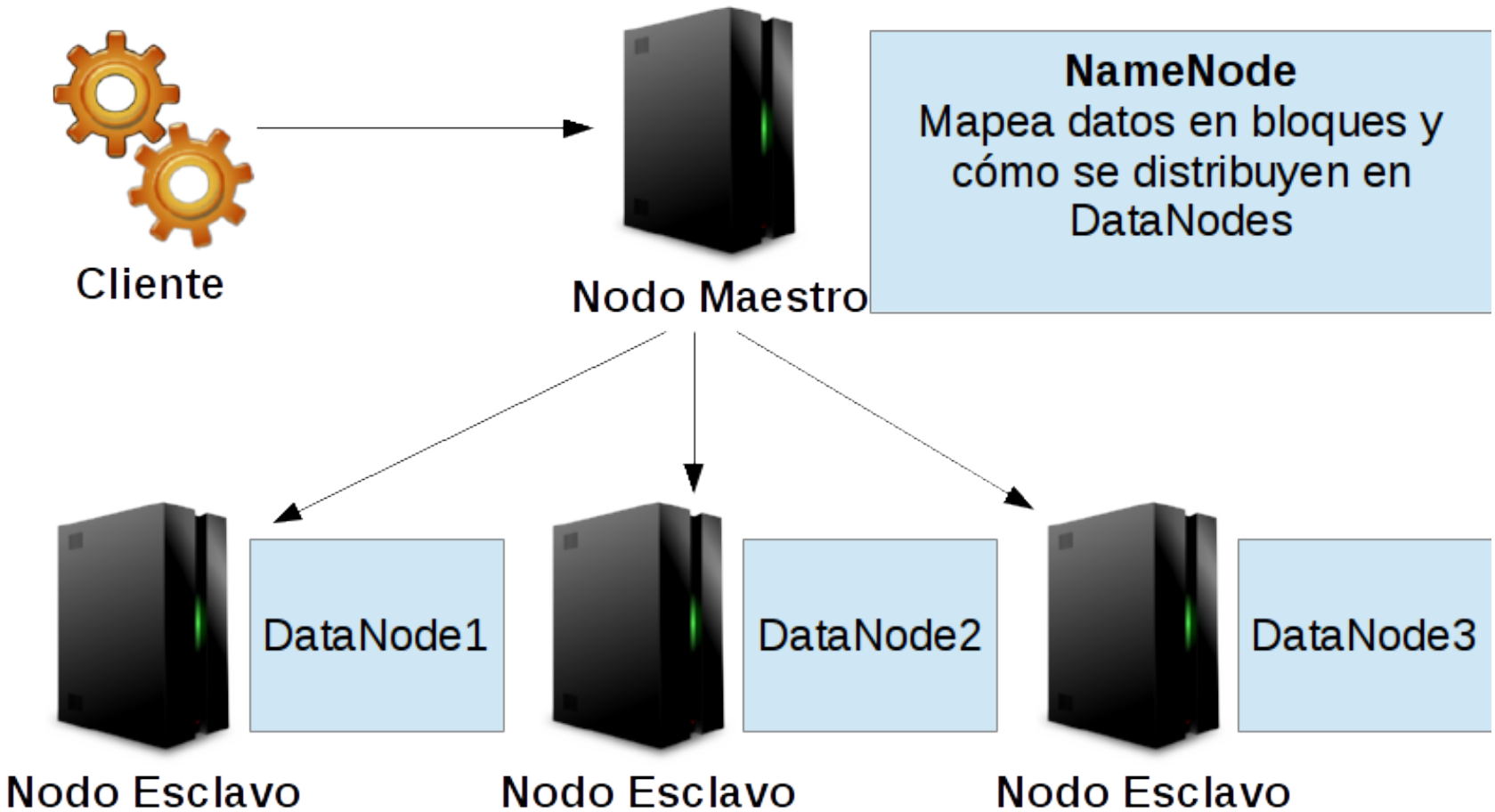
Proyectos Hadoop: Hadoop

- Infraestructura para Hadoop (MR1).



Proyectos Hadoop: Hadoop

- Infraestructura para Hadoop (MR1).



Proyectos Hadoop: Hadoop

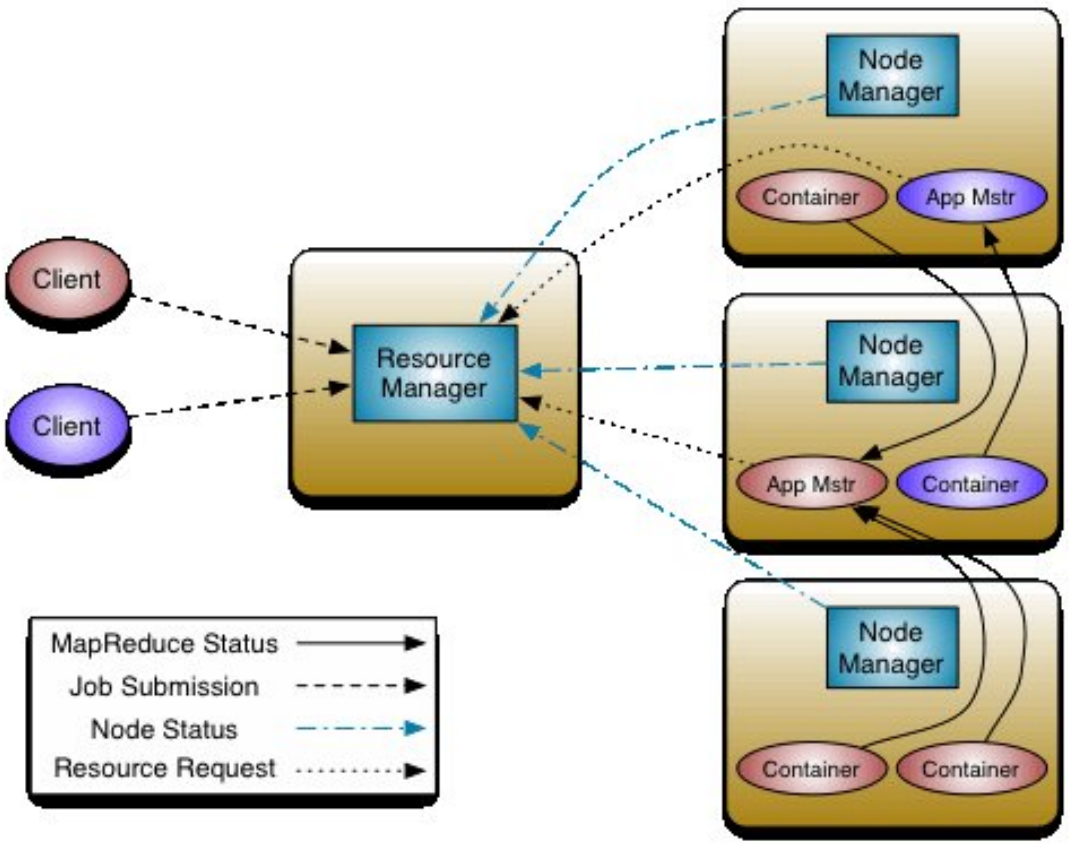
- Gestión de tareas MapReduce (MR1).
- El JobTracker mantiene en memoria del nodo maestro información sobre todas las tareas planificadas y en ejecución.
- Se gestionan tanto las tareas de tipo `map` como las de tipo `reduce`, asociadas a trabajos de alto nivel enviados por el cliente.
- Límites versión 1 (MapReduce/MR1).
  - ~5.000 NODOS, 40.000 tareas concurrentes.
  - Distribución fija de recursos entre procesos *map* y `reduce`.
  - Fallos acaban con trabajos en ejecución y encolados (catastrófico).

Proyectos Hadoop: Hadoop

- Principales diferencias de YARN (MR2) respecto a MapReduce (MR1).
  - Soporte para mútiples estrategias (batch, flujo de datos, interactivo).
  - Data Operating System (un solo conjunto de datos, múltiples instancias).
  - Gestión de metadatos y trabajos distribuido.
  - Mejor aprovechamiento de los recursos de computación y almacenamiento.
  - Introducción de aspectos de seguridad y autenticación en el diseño.
  - Compatibilidad hacia atrás (evitar grandes cambios para Hive, Pig, etc.).

Proyectos Hadoop: Hadoop

- YARN/MR2.



Proyectos Hadoop: Hadoop

- Principales elementos de YARN (MR2).
- **Aplicación**.
  - Representación a alto nivel de un trabajo de procesamiento de datos.
  - Puede ser un trabajo MapReduce o un script de shell.
- **Contenedor**.
  - Unidades de particionado de los recursos hardware subyacentes.
  - Permiten la distribución de los recursos computacionales entre diferentes aplicaciones en ejecución, intentando maximizar el aprovechamiento de los mismos.

Proyectos Hadoop: Hadoop

- Principales componentes de YARN (MR2).
- **Resource Manager**.
  - Gestión de tareas de alto nivel.
  - Gestión de colas jerárquicas de trabajos.
- **Application Master**.
  - Uno por cada instancia a nivel de aplicación.
  - Gestiona recursos, progreso de tareas, planificación, etc.
- **Node Manager**.
  - Agentes encargados de la gestión y monitorización de contenedores en cada nodo del cluster.

Proyectos Hadoop: Hadoop

- Últimos avances en YARN (MR2).
- El Resource Manager continúa siendo un punto de fallo singular. Genera graves trastornos en caso de pérdida de información.
- Solución: intentar reconstruir la información que contiene en caso de fallo.
- Propuesta: Zookeper (u otro meta-gestor de recursos) debe monitorizar las instancias de Resource Managers y actuar en caso de fallo.

Proyectos Hadoop: Apache Hbase

- Base de datos NoSQL, diseñada a imagen de Google BigTable, escrita en Java.
- Orientada a partición por columnas.
- Tolerancia a fallos.
- Compresión de datos, operaciones en memoria, filtros Bloom.
- Funciona sobre HDFS.

Proyectos Hadoop: Apache Cassandra

- Base de datos NoSQL liberada por Facebook en 2009.
- Especialmente pensada para requisitos de alta disponibilidad.
- Replicación en múltiples nodos (incluso alejados geográficamente).
- Diferentes niveles de consistencia de datos entre réplicas (configurable).
- No admite operaciones como JOIN ni subconsultas.

Proyectos Hadoop: Apache Hive

- Sistema *datawarehouse* que ejecuta sobre Hadoop.
- Programar operaciones para análisis de datos directamente sobre Hadoop puede llegar a ser muy tedioso.
- Hive proporciona un lenguaje de abstracción similar a las consultas SQL.
- Procesado de logs (tráfico web, sistemas), minería de texto, indexación de documentos, inteligencia de negocio, predicciones y contraste de hipótesis.

Proyectos Hadoop: Apache Pig

- Creado por Yahoo.
- Resuelve el problema de evitar escribir flujos de análisis de datos en Java para Hadoop.
- Pig-Latin: Lenguaje declarativo para trabajar con flujos de datos.
- Estrategia diferente a Hive, que está más orientado a consultas tipo SQL [7].

Proyectos Hadoop: Hive vs. Pig

- Cómo lo haríamos en Apache Pig [7].

```
INSERT INTO ValuableClicksPerDMA
SELECT dma, COUNT(*)
FROM geoinf JOIN (
    SELECT name, ipaddr
    FROM users join clicks
    ON (users.name = clicks.user)
    WHERE value > 0;) USING ipaddr
GROUP BY dma;
```

Proyectos Hadoop: Hive vs. Pig

- Cómo lo haríamos en Apache Pig [7].

```
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 ValueClicksPerDMA into 'ValuableClicksPerDMA';
```

Proyectos Hadoop: Apache Mahout

- Construcción de bibliotecas de machine learning sobre Hadoop.
- Clustering (K-means, K-means con lógica difusa).
- Sistemas de recomendación.
- Múltiples clasificadores:
  - Regresión logística.
  - Naive Bayes.
  - Árboles de decisión.
  - Random forest, etc.

Proyectos Hadoop: Flujos de datos

- Storm, Flume

Proyectos Hadoop: Gestión de flujos de trabajo

- **Zookeper**. Gestión de procesos distribuidos.
  - Registro de nombres.
  - Monitorización y alta disponibilidad.
  - Coordinación: información de estado, ejecución de tareas, etc.
- **Oozie**. Planificador de tareas Hadoop.
  - Funciona principalmente con flujos de tareas Hadoop y Pig.
  - Proporciona un sistema para definir el grafo de tareas y dependencias entre ellas, de manera que podamos organizar la ejecución de actividades complejas.

Otros proyectos: Procesamiento *streaming*

- Apache **Samza**: Framework para procesamiento de flujos de datos, concebido para su utilización junto a Apache Kafka y YARN.
  - Combina un sistema de streaming de datos (Apache Kafka) con un gestor de recursos (que puede ser YARN) y una API que ofrece primitivas y rutinas de procesamiento de flujos de datos.

- Apache **Kafka**. Sistema *message broker* para gestionar flujos de datos de alta velocidad y elevado throughput. Fue creado originalmente para tratamiento de logs de eventos en la plataforma LinkedIn, pero su uso se puede generalizar a otros tipos de flujos de datos.
  - Puede funcionar en combinación con Samza, YARN, Spark, etc.
  - Optimizado para procesamiento de grandes volúmenes de mensajes minimizando las probabilidades de pérdida de información.

- Apache Spark.
  - Lo veremos en detalle en la próxima sesión.

Referencias

1. Holmes, A. *Hadoop in Practice*. Manning Publications, 2012.
2. Google File System
   `http://research.google.com/archive/gfs.html`
3. Murthy, C. A., Vavilapalli, V. K., Eadline, D., Niemiec, J. Markham, J. *Apache Hadoop YARN*. Addison-Wesley Professional, Mar. 2014.
4. Fasale, A., Kumar, N. *YARN Essentials*. Packt Publishing. Feb. 2015.