BIG DATA

MSc. Felipe Meza

Objetivos

General

- Entender y aplicar técnicas de análisis de grandes cantidades de datos para la resolución de problemas concretos a través de tecnologías de manipulación, extracción y sintetización estadística.

Específicos

- Aplicar bibliotecas para la transformación de datos a gran escala para poder sintetizar el conocimiento para futuro análisis.
- Aplicar técnicas de análisis de datos para extraer patrones que mejoren el entendimiento de un problema concreto.
- Aplicar técnicas para aprendizaje automatizado de patrones, basado en datos existentes, para mejorar la certeza de la solución aplicada a problemas concretos.

Contenidos

- Introducción a procesamiento a gran escala
- Fuentes y repositorios de datos
- Procesamiento de fuentes (data frames)
- Procesamiento de atributos
- Organización de datos procesados
- Análisis de datos a gran escala
- Uso de modelos de aprendizaje automático a gran escala

Evaluación

TAREAS CORTAS 70%

PROYECTO 30%

REPASO ML…

Big Data Intro

Por qué Big Data?

Se están generando grandes cantidades de datos.

- Algunas organizaciones almacenan datos de 3-4 años.
- Hay nuevas fuentes de datos, social media, bussiness beahivor etc.
- Nuevos métodos de almacenamiento, cloud etc.

Hay una variedad de datos de distinta naturaleza.

- Alrededor del 80% de los datos son no-estructurados.

Nuevos tipos de aplicaciones hacen uso de esos datos.

- Hasta hace poco para grandes cantidades de datos se usaban aplicaciones que tenian ~15 años de existir, los negocios son un factor de cambio.

Por qué no usar los métodos convencionales?

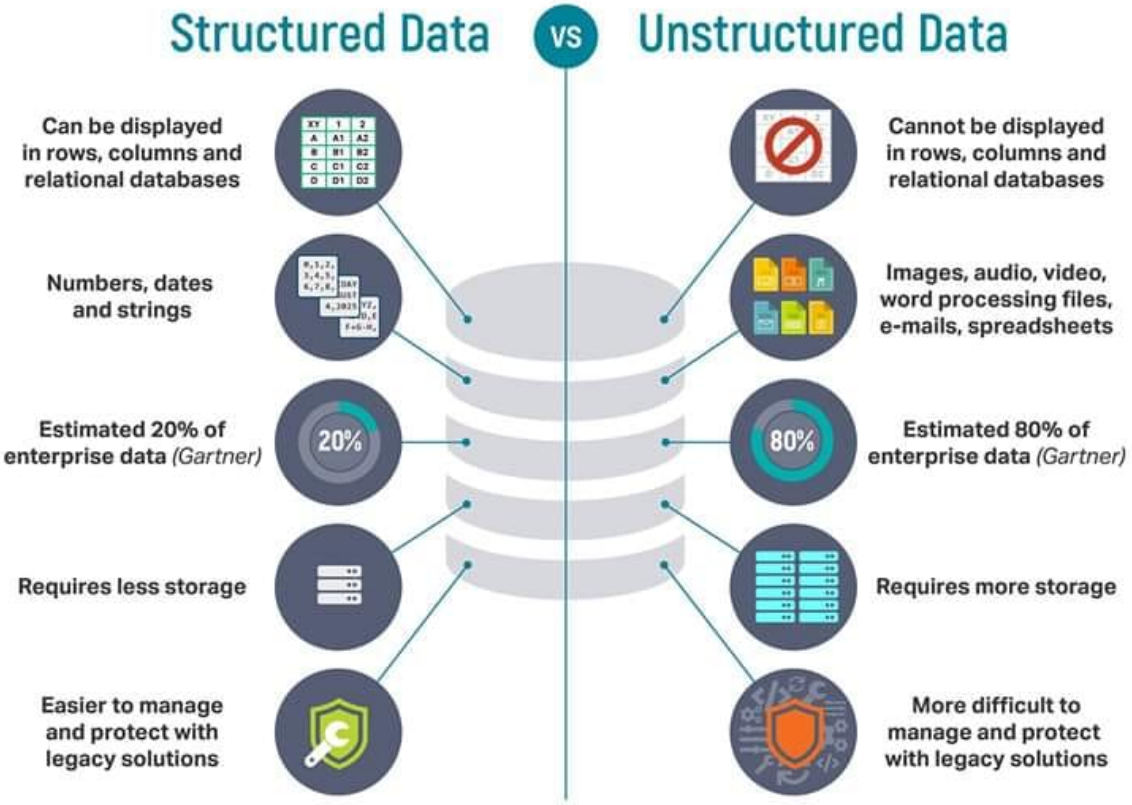
Los niveles de complejidad en las tareas de captura, administración y análisis de grandes cantidades de datos son muy **ALTOS**.

- Métodos actuales = MB, GB de información.
- Métodos BIG DATA = TB, PB de información.

Los datos no estructurados son complejos de manejar en grandes volúmenes con los métodos actuales.

Las grandes cantidades de datos conllevan a un almacenamiento distribuido.

Tipos de datos en Big Data?



Aspectos a tomar en cuenta con Big Data

Crecimientos desmedidos en la cantidad de datos.

Poder de procesamiento.

Almacenamiento físico.

Formato de datos e interoperabilidad.

Extracción, transformación y carga de datos (ETL) puede llegar a ser muy tiempo/costo consumo.

El problema…

- Imagine que tenemos una lista de palabras y necesitamos contar las palabras e identificar las de mayor frecuencia:
  - red, green, blue, bird, green, red, red.
- Será igual si tenemos 10 000, 100 000 o 10^9 palabras?
- Se puede pensar en un algoritmo de conteo de palabras, pero:
  - Problema a nivel de memoria.
  - Uso de streams? Problema cuando la palabras no son tan frecuentes.

El problema…

- Y que pasa si usamos múltiples maquinas para tratar el problema?
  - Por ejemplo, si por maquina se tiene 10 trillones de palabras, con 10 máquinas tendría 100 trillones de palabras.
- Pero…cómo se consolidan los datos provenientes de las 10 máquinas?
  - Por ejemplo, con “controlador central” (CC) donde se vayan dando los resultados y al final se sumen en este punto.
- Potenciales problemas con esta aproximación:
  - Ancho de banda, potencial solución: enviar la data al CC en grupos, e.g 10
  - Podrían fallar algunas máquinas? **SI**

El problema…

- Sea X=1 cuando una máquina no funciona.
- Sea X=0 cuando una máquina funciona.

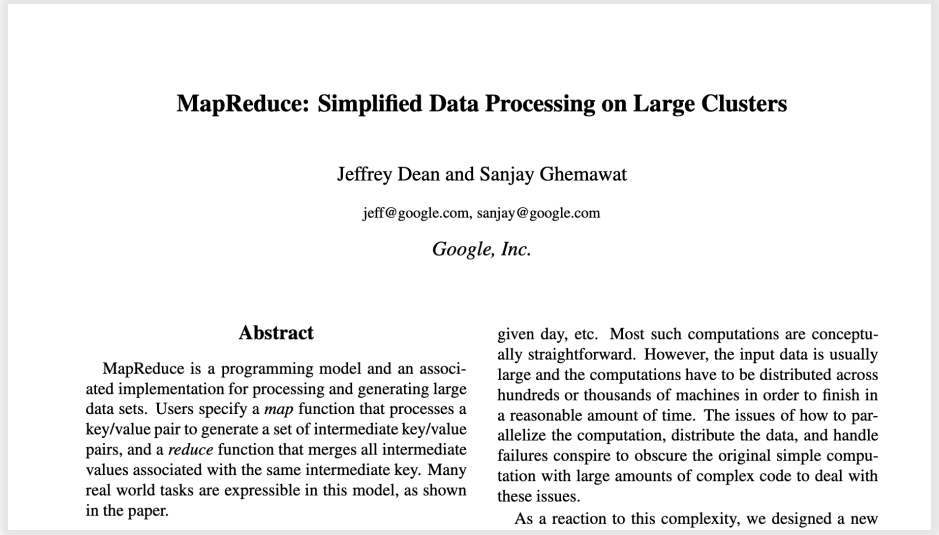
- Tenemos:

  - $P(X = 0) = (1 - \varepsilon)^n$, donde n es igual al numero de máquinas y ε la probabilidad de que una máquina falle e.g 0.001.
  - Para el caso de 10 máquinas, más del **99%** de las ejecuciones no debería existir errores.
  - Para el caso de 1000 máquinas, más del **37%** de las ejecuciones no debería existir errores.
  - Potencial solución? Haciendo copias de los datos (sistema tolerante a fallas), típicamente se usan 3 copias.

Pero Google usa 10K computadores?
Cómo lo hacen?

MapReduce

- Sistema tolerante de fallas, es automático.
- Usar 1000 máquinas es mas fácil que 100.



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

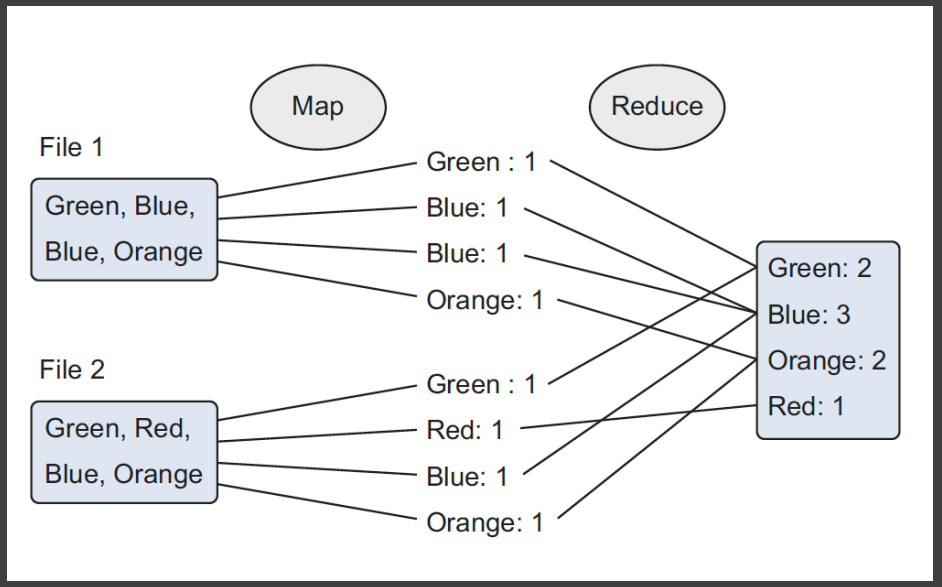
MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

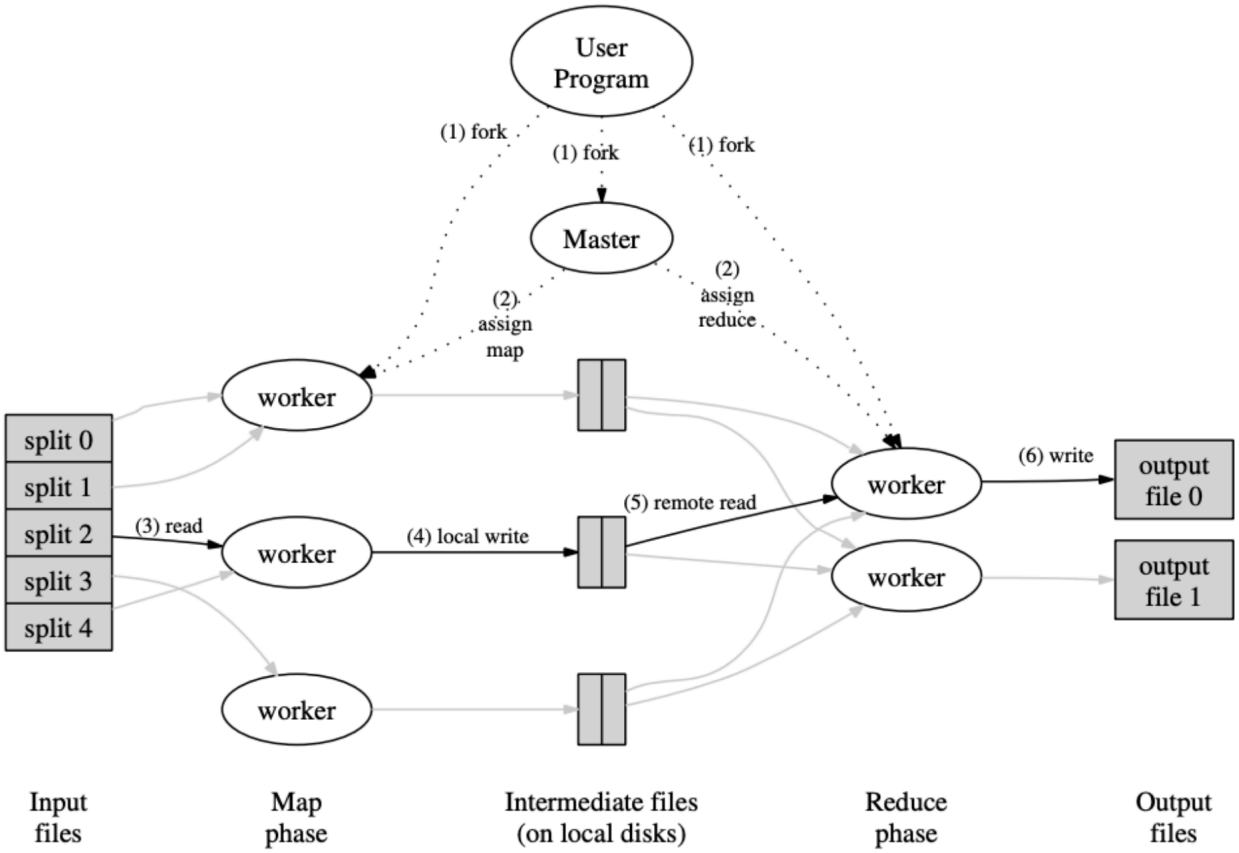
given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

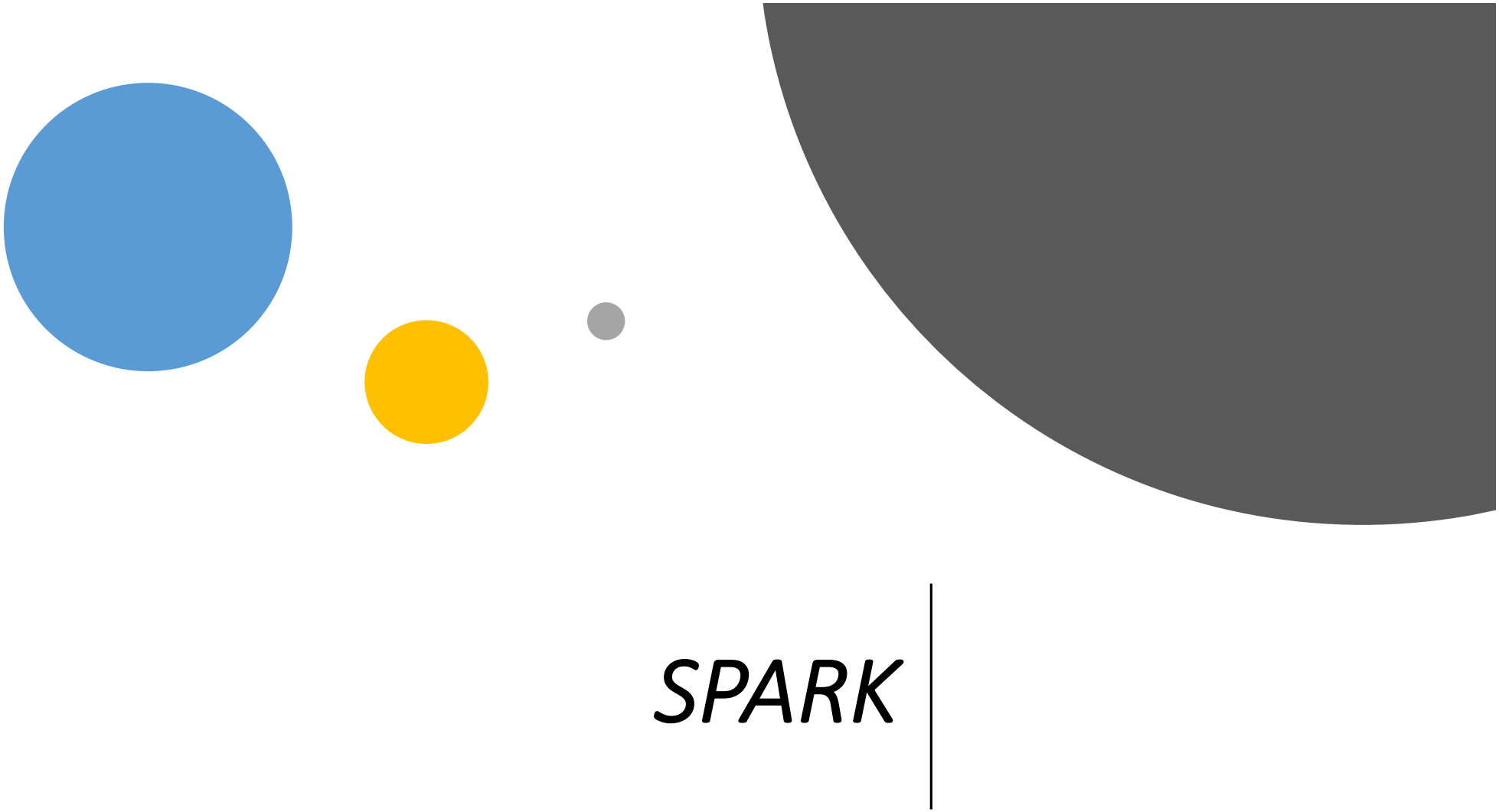
As a reaction to this complexity, we designed a new

MapReduce

- Mapper Function:
  - Función de mapeo de los pares (Key, Value) originales a una representación intermedia.
- Reducer Function:
  - Funcio de reduccion que une todas las llaves (Keys) en una sola entrada.



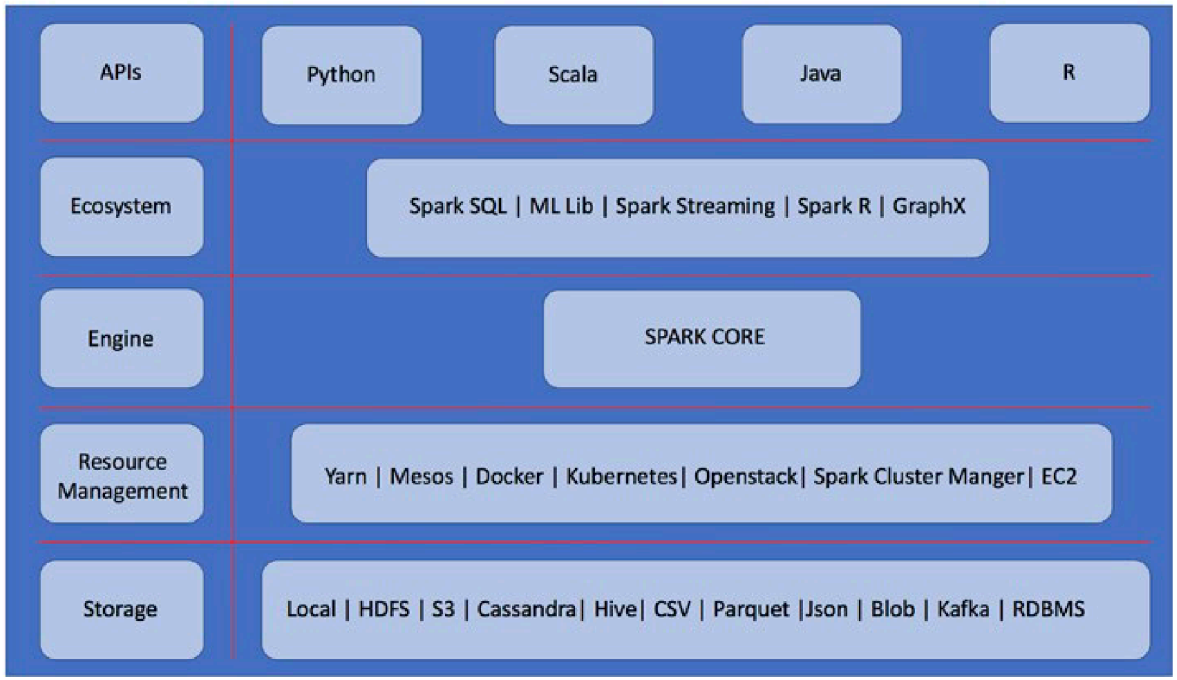

User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
split 0
split 1
split 2
split 3
split 4
(3) read
worker
(4) local write
worker
(5) remote read
worker
worker
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files


SPARK

Origen Spark

- Nace como una propuesta para mejorar algunas limitaciones de Hadoop MapReduce.
- En especial temas asociados a velocidad.
- x100 mas veloz que otros frameworks.
- Nativo SCALA.

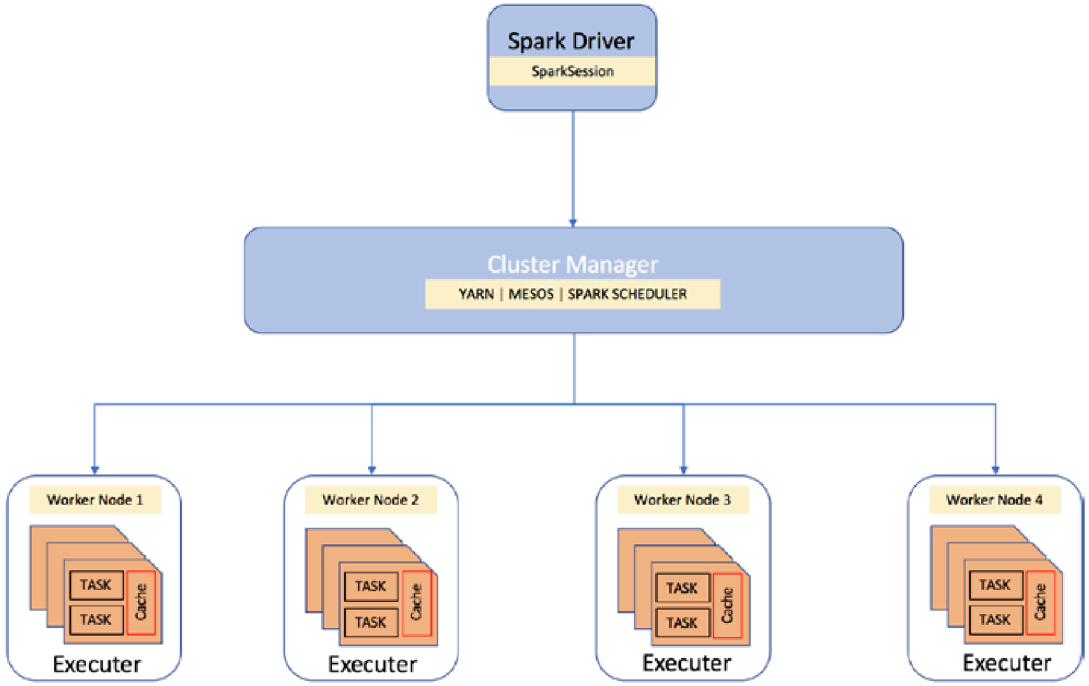
Arquitectura Spark



Arquitectura Spark

- ***STORAGE***: Datos típicamente almacenados en una DB.

- ***RESOURCE MANAGEMENT***: Administración del *Spark Cluster (conjunto de maquinas/cores).*
  - ***Cluster Manager****: Controla y asigna tareas al worker, guarda información de los nodos que operan los "workers" (memoria, CPU's, localización etc).*
  - ***Worker****: Ejecuta lo que le indica en CM.*

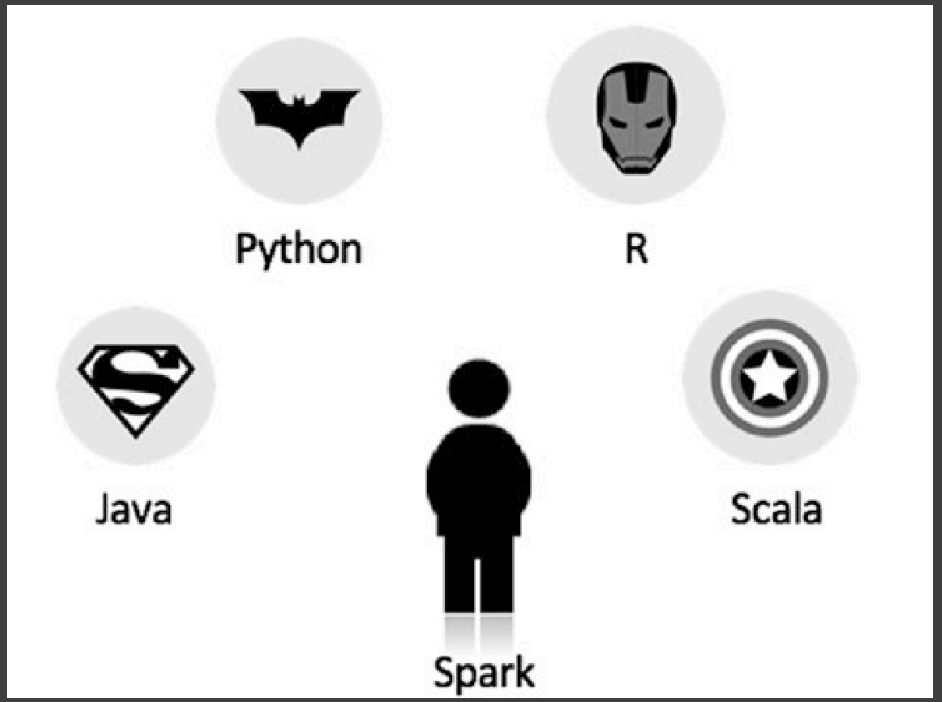

Spark Driver
SparkSession
Cluster Manager
YARN | MESOS | SPARK SCHEDULER
Worker Node 1
TASK
TASK
Cache
Executer
Worker Node 2
TASK
TASK
Cache
Executer
Worker Node 3
TASK
TASK
Cache
Executer
Worker Node 4
TASK
TASK
Cache
Executer

Arquitectura Spark

- ***ENGINE and ECOSYSTEM***:
  - Dos componentes; la infraestructura de computo distribuida y la programación RDD (Resilient Distributed Datasets) la cual es accedida mediante los APIs.
  - El ecosystem esta compuesto por módulos o librerías para diferentes objetivos:
    - **Spark SQL**: Acceso a SQL DB.
    - **Mllib**: ML Library.
    - **Structured Streaming**: Datos en tiempo real.
    - **Graph X**: Visualización.

Arquitectura Spark

- ***API***: Interface por la que se opera el Spark Core (Engine).



AMBIENTE TRABAJO


CONDA


FindSpark


Java


PySpark


PostgreSQL


Java
Development Kit


APACHE
Spark