Spark – Part 2

Software Concurrent

Lluís Garrido – lluis.garrido@ub.edu

Grau d'Enginyeria Informàtica

Spark

Hem vist que

- Spark és un motor de computació unificat per a clústers d'ordinadors. A l'actualitat inclou una sèrie de llibreries i eines per a fer un munt de tasques paral·leles.
- Les aplicacions són escrites a alt nivell, sense haver de preocupar-se per temes de baix nivell com la xarxa, dependències, sincronitzacions...
- Spark es pot considerar una millora de Hadoop, un sistema que va ser desenvolupat per processar grans volumns de dades.

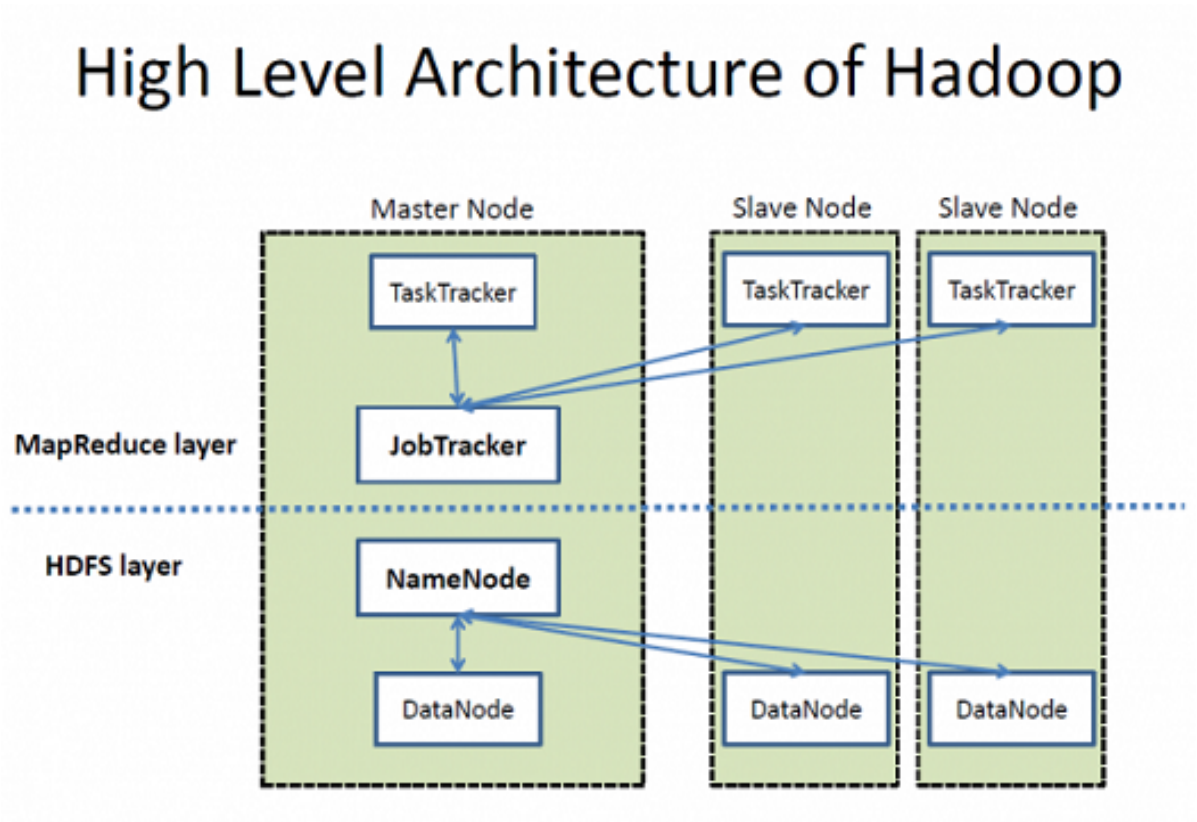
Hadoop

Una mica de l'evolució històrica...

- Hadoop es basa en el treball fet per Google als inicis dels 2000.
- Adopta una nova aproximació al problema de computació distribuïda, complint tots els requiriments desitjats de escalabilitat, robustesa, fiabilitat... de forma similar a com ho fa Spark actualment.

Hadoop: arquitectura

L'arquitectura de Hadoop es basa en el sistema de fitxers HDFS (Hadoop Distributed FileSystem) i una arquitectura màster-esclau escrita en Java.



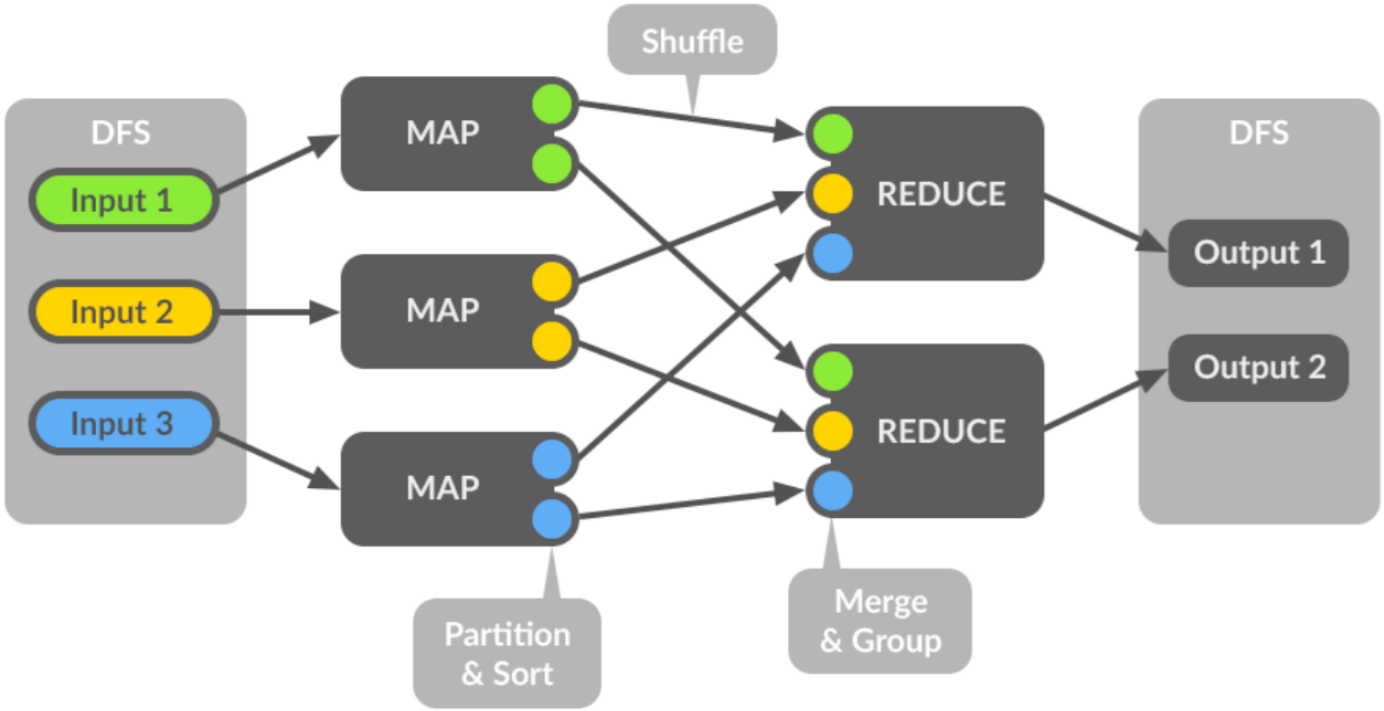
Hadoop: components

Els components de Hadoop són

- Hadoop Distributed File System (HFDS): el sistema de fitxers distribuït per emmagatzemar grans volums de dades.
- Hadoop Common: llibreries i utilitats.
- Hadoop YARN: entorn per a la planificació i gestió dels recursos del clúster.
- Hadoop MapReduce: implementació del paradigma MapReduce que permet processar grans volums de dades.

Map Reduce

Esquema del paradigma Map Reduce



Map Reduce

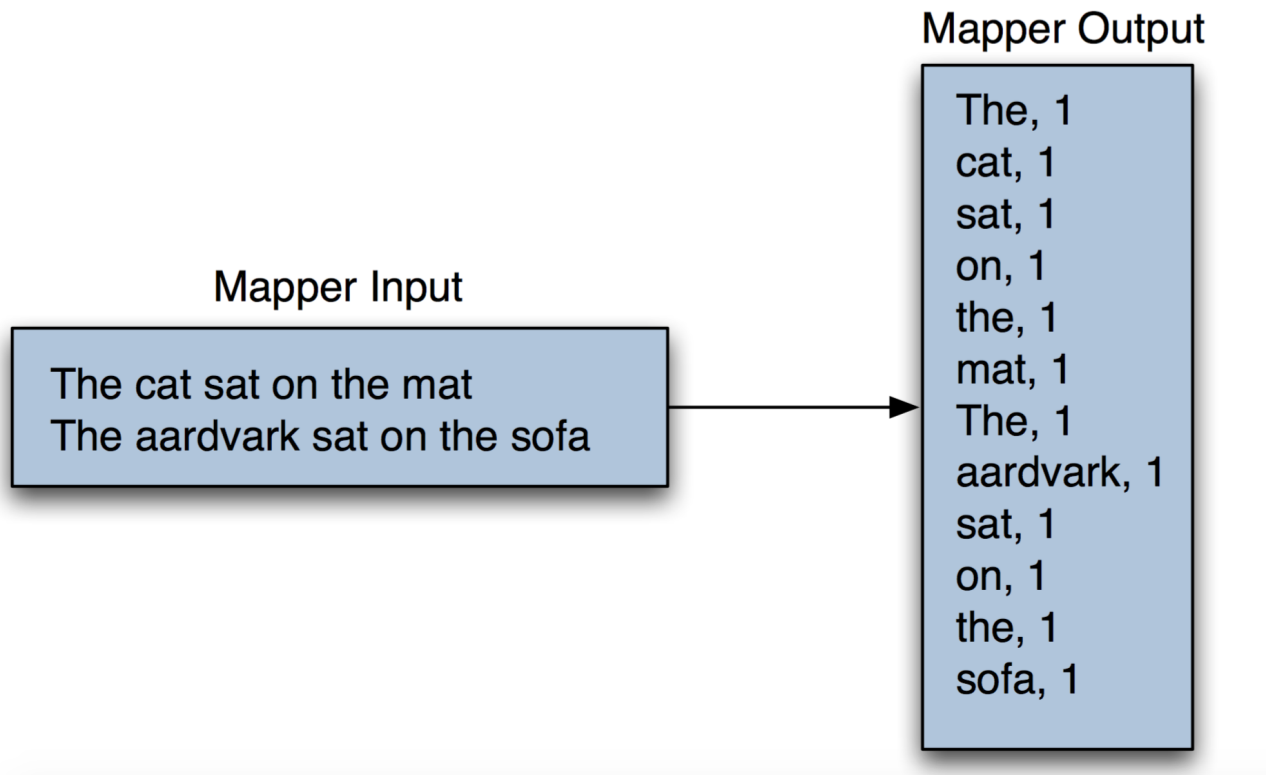
Paradigma map-reduce

1. Les dades d'entrada estan distribuïdes en un sistema de fitxers (DFS).
2. El map: Cada node esclau s'encarrega de processar el fitxer de la partició emmagatzemada al seu node (comptar, filtrar, aplicar una funció agregada tipus màxim, mínim, variància, ...)
3. El reducer: un cop aplicat el map, es combinen els resultats (intermedis) i s'envien al reducer. Hi pot haver un o múltiples reducers: es fa servir una "clau" per saber a quin reducer s'han d'enviar els resultats. Aquests resultats intermedis s'envien al reducer ordenats per la "clau" per facilitar el processament.

Map Reduce

Exemple: comptar paraules. El primer procés és el map

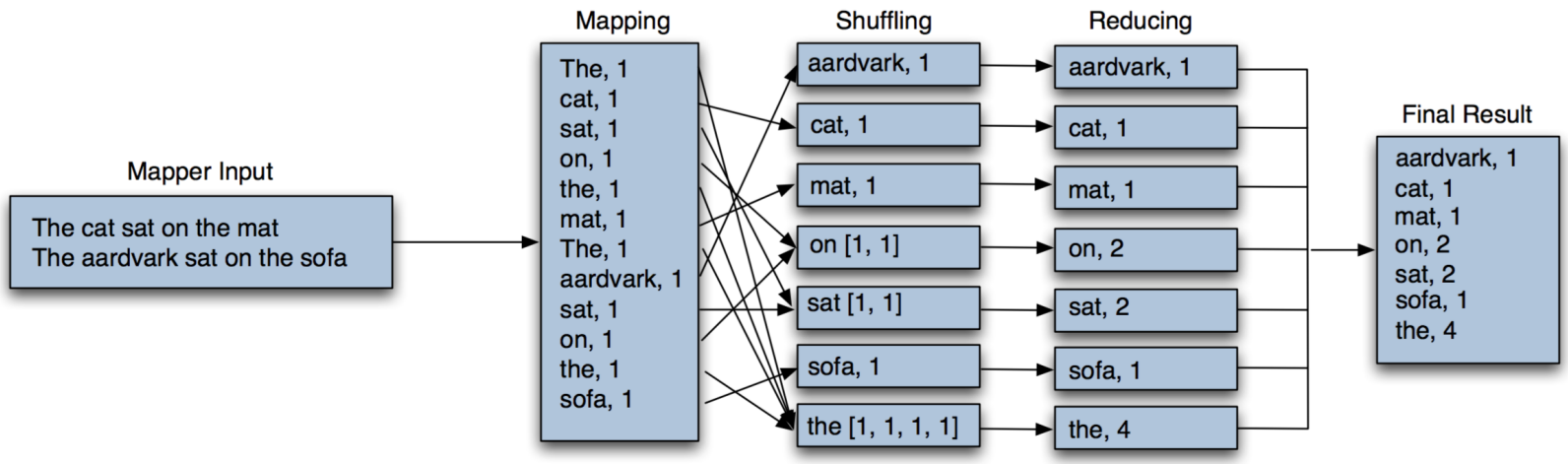
- En aquest exemple es mostra el map d'un node. Al Hadoop cada node aplicarà el map a les seves dades.



Map Reduce

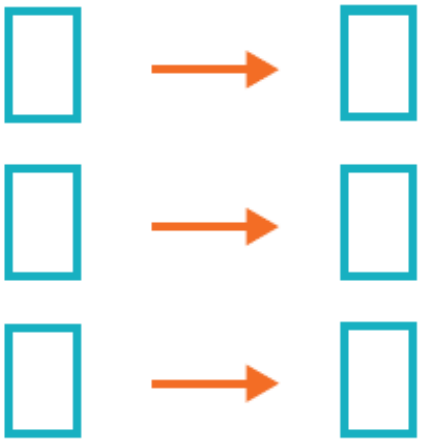
Exemple: comptar paraules. Després d'ordenar es passen al reduce

- En aquest exemple se suposa que només hi ha un fitxer d'entrada. El map-reduce realment combinarà els resultats de tots els maps dels nodes.

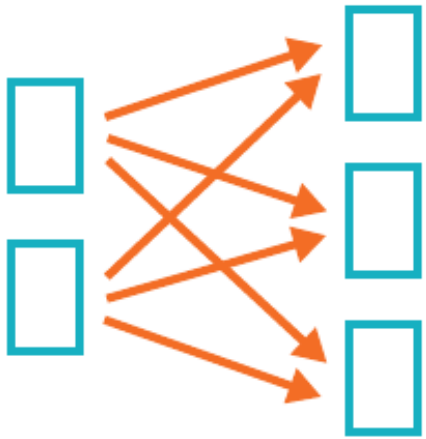


Map Reduce

El map-reduce utilitza dues operacions que Spark també fa servir



Map
Narrow transformation



Shuffle
Wide transformation

Spark

Respecte el Hadoop Map-Reduce, a Spark

- Successives operacions de "narrow transformation" es realitzen, sempre que es pot, amb "canonades", fent totes les operacions que es poden a memòria. Els resultats intermedis no s'emmagatzemen a disc.
- A les operacions que impliquen un "shuffle", Spark mourà dades entre nodes per intentar que un node emmagatzemi les dades associada a una clau. Per això Spark executa els shuffles fent que els nodes "origen" emmagatzemin les dades agrupades a fitxers locals. Aleshores els nodes "destí" agafen els fitxers agrupats de cada node "origen" per agrupar-los en un mateix node i operar després sobre ells (pex agrupar el fitxer de vols d'avió per l'origen i fer càlculs estadístics sobre el retard).

Respecte shuffle a Spark

- Recordar (del tutorial de pràctiques) que a Spark, en fer una operació, Spark la realitza des del principi, des de les dades d'origen. Podem fer servir la instrucció `cache` per indicar quins DataFrames volem fer persistents (a memòria o disc) per optimitzar.
- L'operació resultant de "shuffle" queda emmagatzada a disc automàticament, i Spark farà servir aquestes dades emmgatzemades a disc cada cop que l'usuari necessiti fer operacions sobre elles sense necessitat de començar des del principi. És una optimització que ofereix Spark.