UE Large Scale Processing @ ESIGELEC
2019/2020

07 – Hadoop MapReduce

Abdel Dadouche
DJZ Consulting

adadouche@hotmail.com
@adadouche

Context

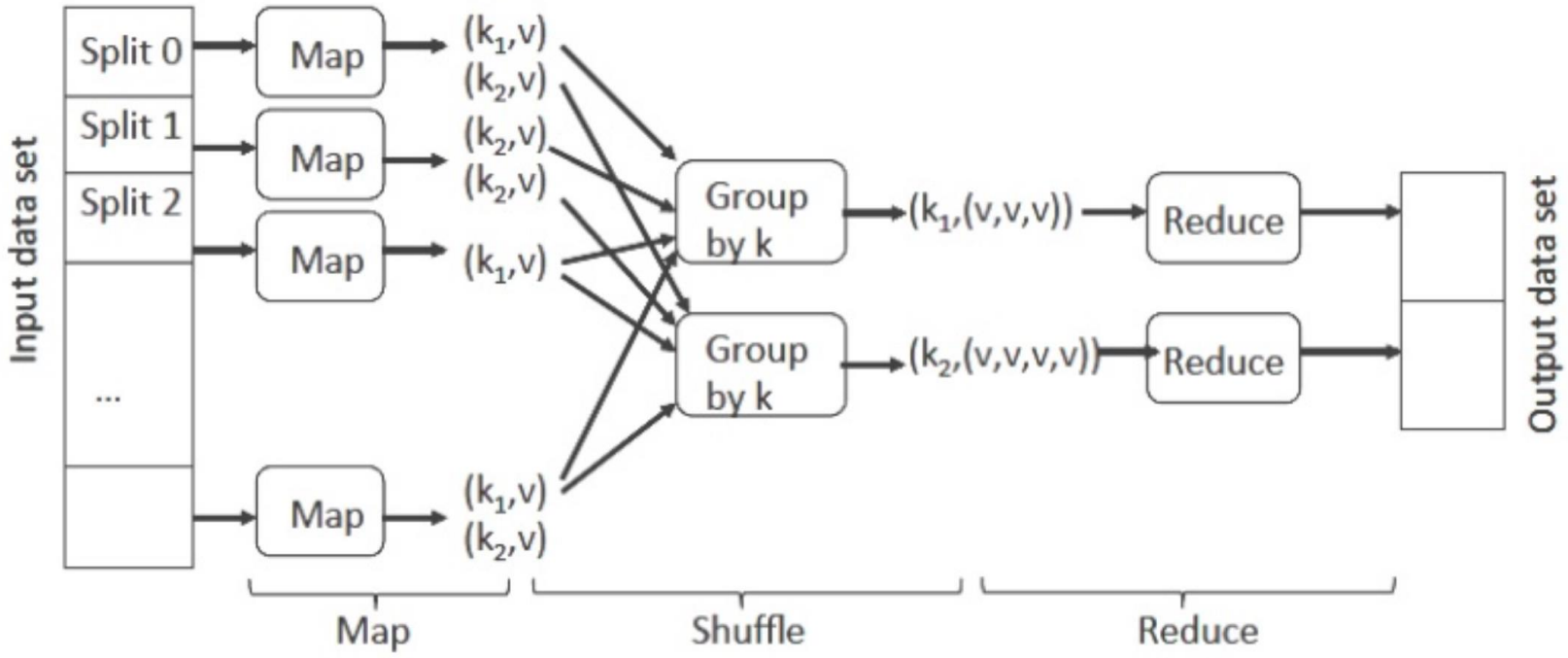
- In order to implement a large scale distributed processing system, you need be able to:
  - Split your process into smaller pieces
  - Apply these smaller processing unit on fragment of your data
  - Reconcile the overall process and result
  - Generalize this approach to all type of problems

MapReduce = Map + Reduce operation

- Map
  - Will transform your data into a key/value pairs where the key will be used to group entries and must be chosen based on the problem to solve
  - The Map operation must be able to run on fragments of the data (data blocks)
  - The execution will take place on the node where the data block is located

- Reduce
  - Process each set of values for each keys produced by the Map
  - Each set of values is processed by a single node
  - There will be one unique result for each set of values (each key) at the end

Here is how to model your problem in 3 ops!



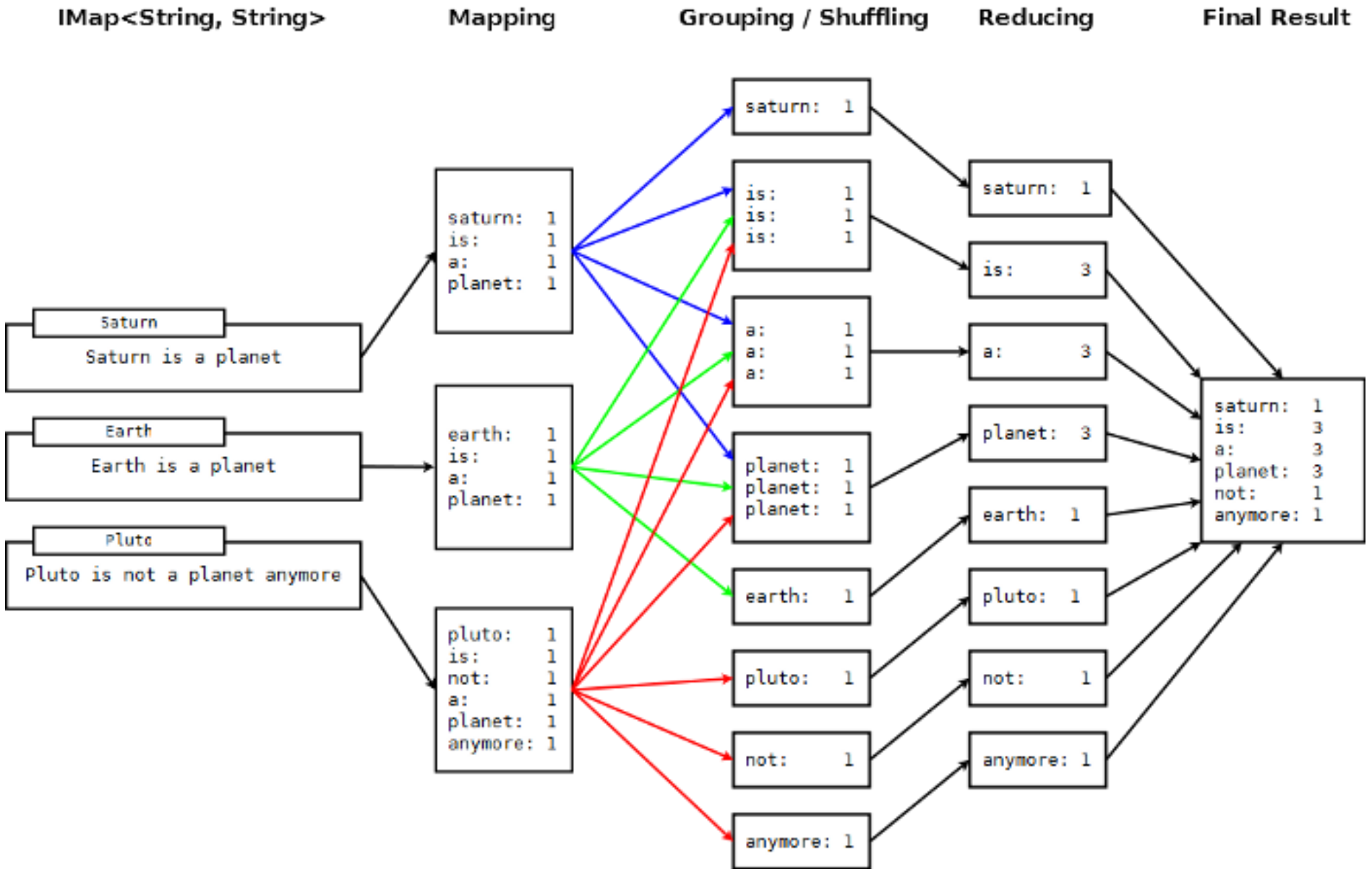
- Split:
  - Represent a data fragment (a data block stored by a data node)
- Map
  - Transform the split into key/value pairs
- Shuffle (optional):
  - Group back the key from each Split
- Reduce
  - Process each set of key/value pairs into a final value

The Word Count example

- A record will correspond to a ling from the text file separated by a “\n”
- Split:
  - You can choose to have a split per row or for multiple rows
  - As long as each record is stored in a unique split
- Map
  - The key will the word itself. Every time the word will found, the value (“count”) will be incremented
- Shuffle
  - Group by “word” the “count” from each Map
- Reduce
  - Sum the “count” for each group of identical “word” from the Shuffle

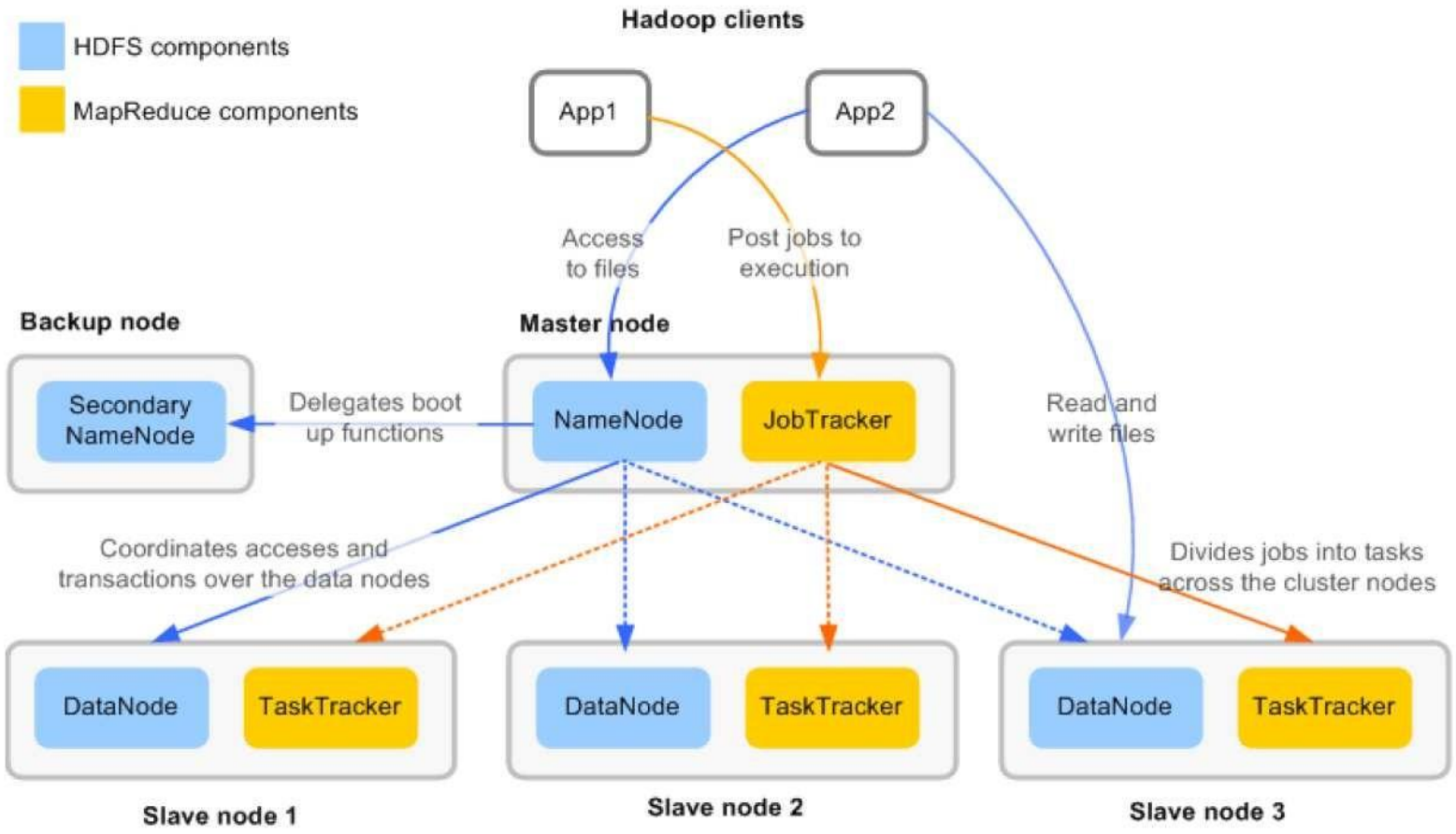
The Word Count example

- On a small file or a single node cluster, the performance gain will be fairly limited
- Now, imagine a cluster which will run 5 Mapper and 4 Reducer with 10 Split
  - The Map operation should take only a fifth of the time
  - The Reduce operation should take only a fourth of the time
  - Hoever, you will need to consider the transfer in between nodes (if any)

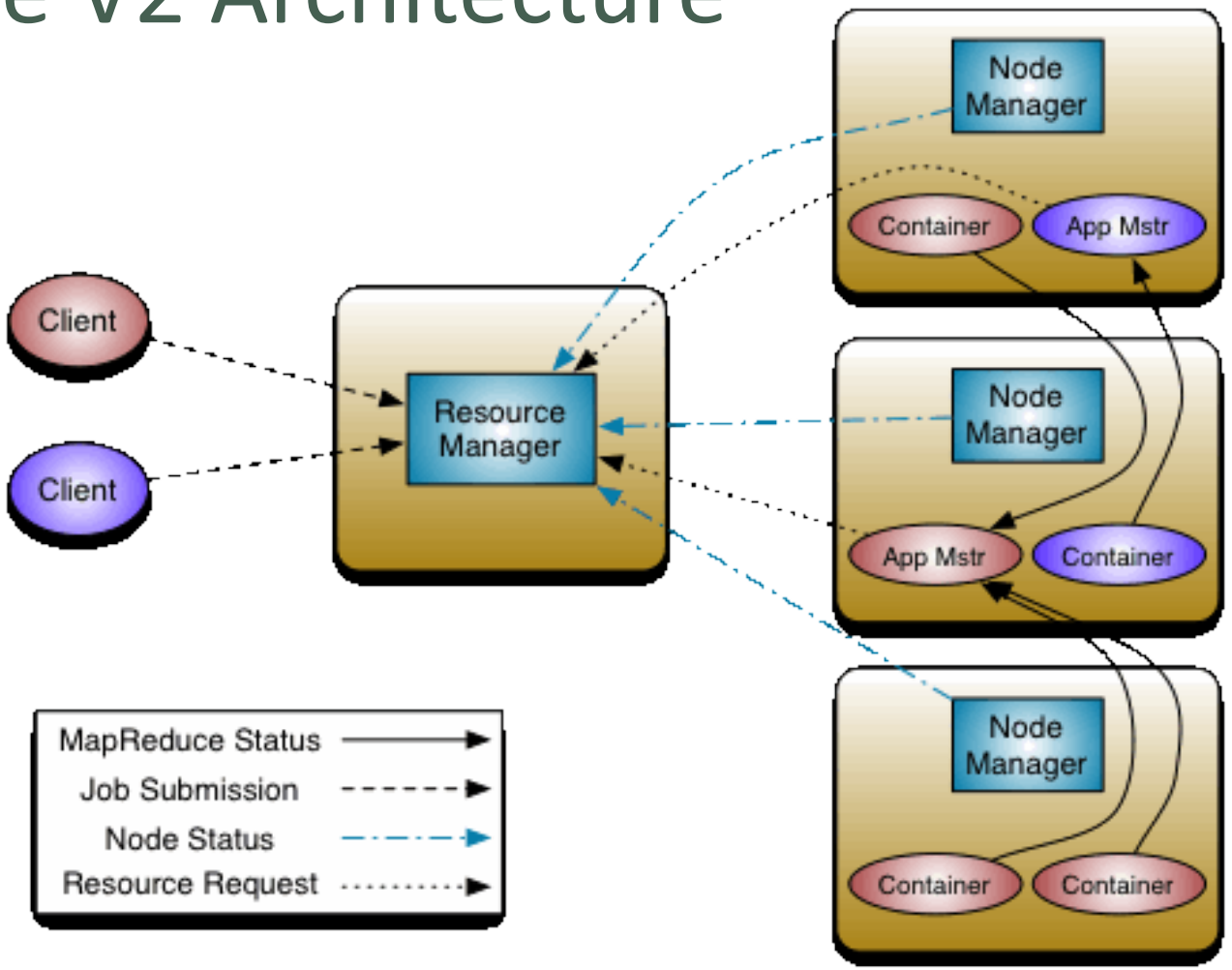


Source: https://docs.hazelcast.org/docs/3.2/manual/html/mapreduce-essentials.html

MapReduce V1 Architecture



MapReduce V2 Architecture



Recap

- Only a few ops: Split, Map, Shuffle, Reduce
- Divide and conquer approach
- Be ready to code in Java!
- The architecture has evolved a lot (with YARN) for more flexibility and better resources allocation & distribution