UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

DTIM
www.essi.upc.edu/dtim

MapReduce
Hands-on Exercises

Petar Jovanovic

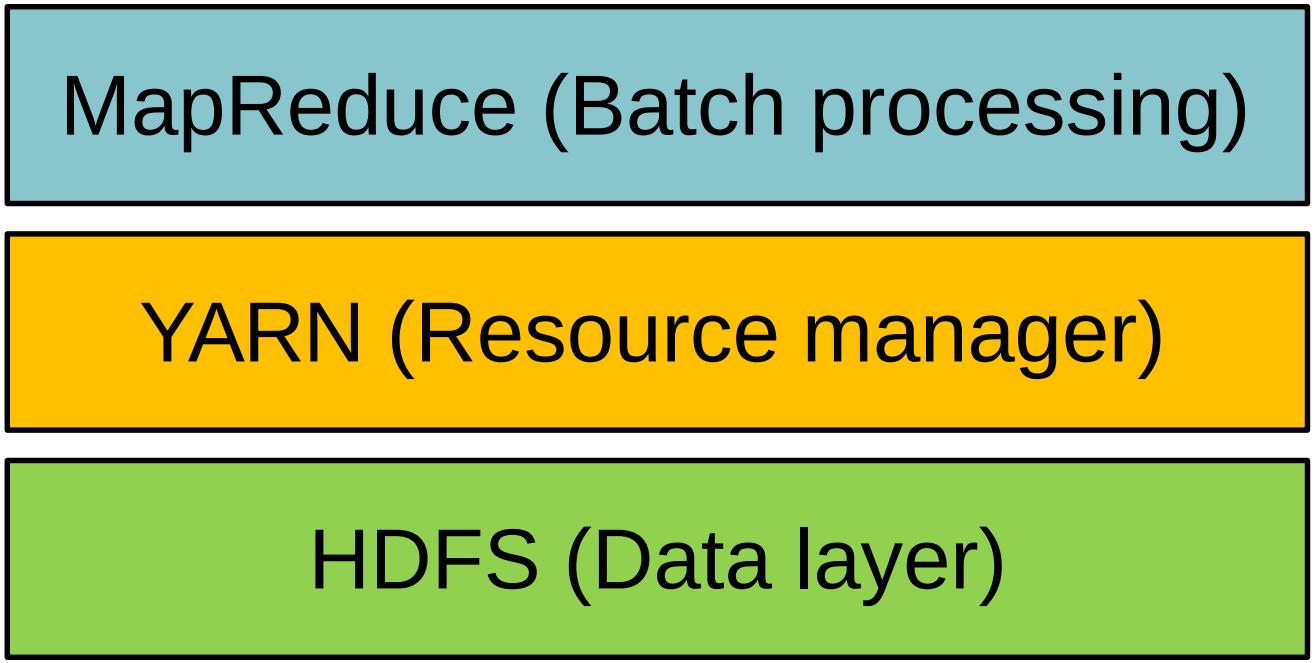
Barcelona; February 6th, 2024

Table of contents

1. Introduction to MapReduce

- The idea of MapReduce is to split the input into chunks that can be processed independently.
- These partial results are then merged into different groups in order to apply group operations (e.g., aggregations).
- Divide-and-conquer for very large data sets
  - Exploits the brute force of the cloud

1. Introduction to MapReduce



2. MapReduce and YARN

- YARN keeps track of the cluster resource usage
- Architecture
  - Master
    - ResourceManager (global)
      - Arbitrates resources among all applications
  - Slaves
    - NodeManagers (node) → Executes jobs

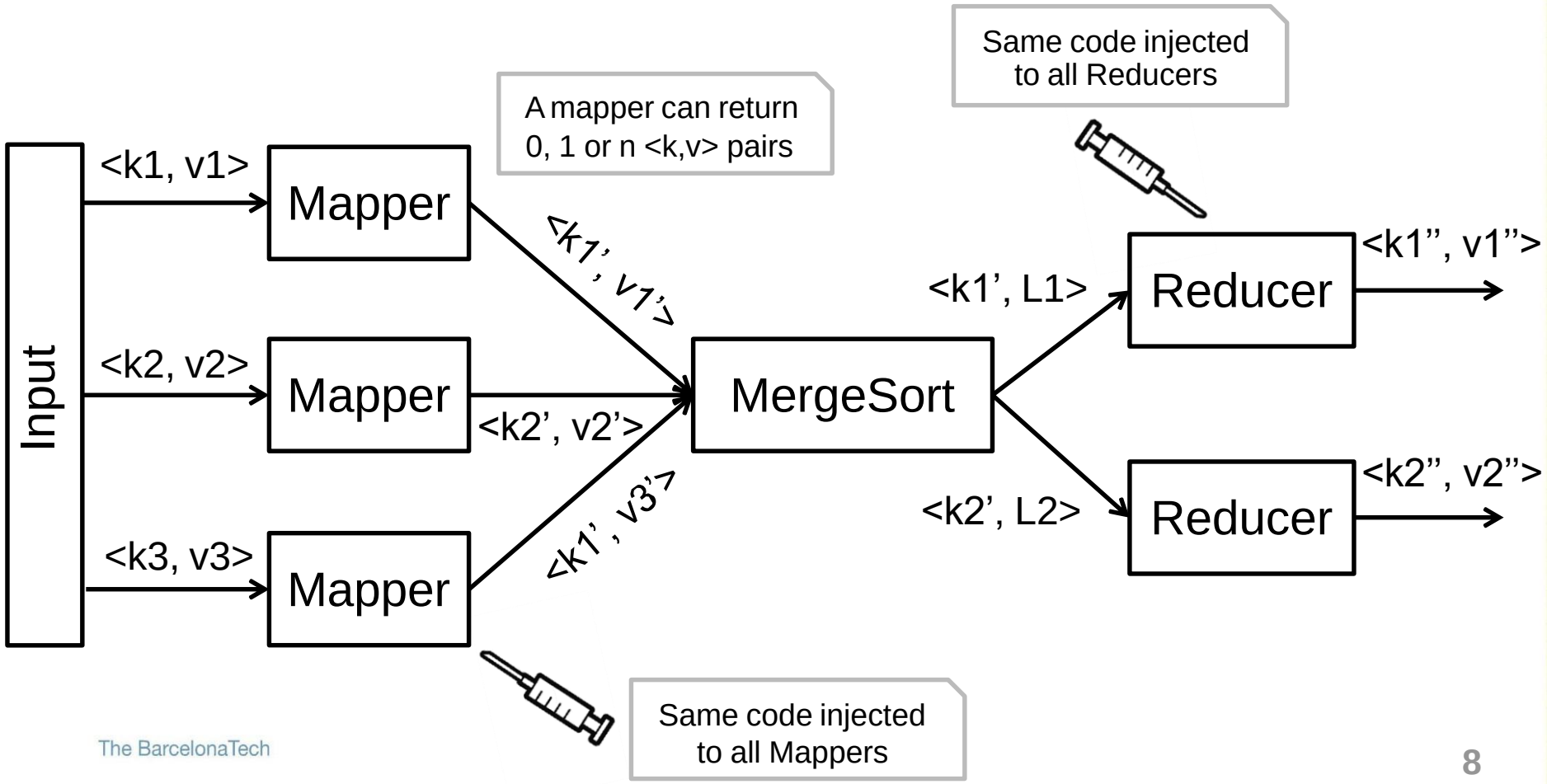
2. MapReduce and YARN

- ApplicationMaster
  - Launches the applications (Negotiates the first container)
  - Monitors and negotiates resources with the ResourceManager
- Scheduler
  - Allocates resources to the running applications
    - Based on application requirements
  - Includes concept *Container*
    - Memory, CPU, disk, etc.

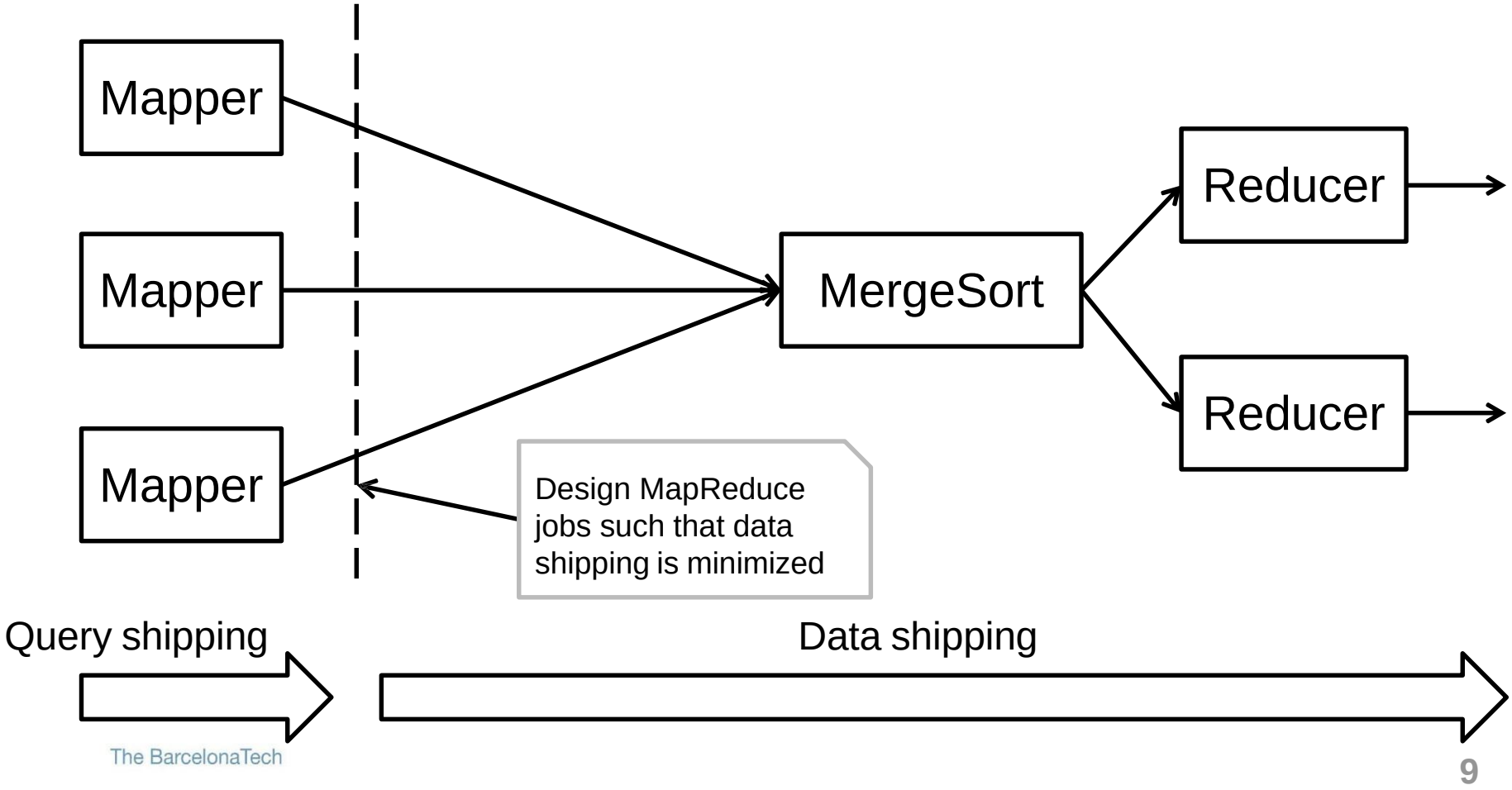
3. The MapReduce framework

- What needs to be implemented?
  - The code that processes input <key,value> pairs
    - The user must inject it
  - The code that merges the partial results
    - Provided by the framework
  - The code that processes grouped <key, list of values>
    - The user must inject it

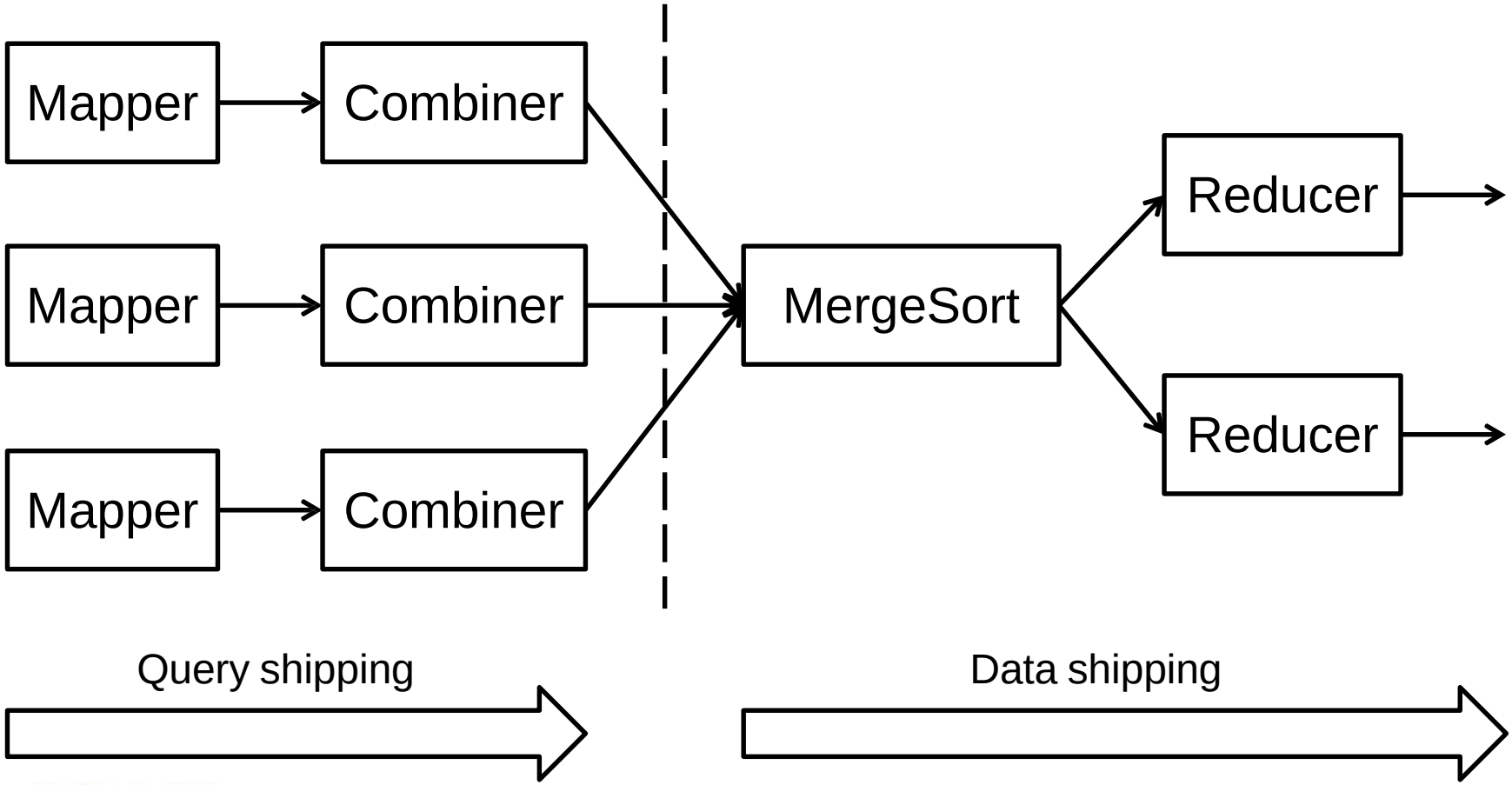
3. The MapReduce framework



3. Anatomy of a MapReduce job



3. Anatomy of a MapReduce job



3. Anatomy of a MapReduce job

- Combiners
  - Are included to exploit data locality at Mapper level
  - Can have their own code but normally they are Mapper-local runs of Reducer code
    - To apply so, Reducer function must be commutative and associative
  - Benefit is:
    - MergeSort cost diminished since Mapper outputs are reduced
      - This includes network and storing intermediate result costs

4. Example: Friends in common

- In a social network (e.g., Facebook) we aim to compute the friends in common
  - This is a value that does not frequently change, so it can be precomputed
- Friends are stored as Person -> [List of friends]
- The input
  - A → B C D
  - B → A C D E
  - C → A B D E
  - D → A B C E
  - E → B C D

4. Friends in common – Map task

- For every friend in the list of friends, the mapper will generate a <k,v> with
  - Key: the input key and one friend in alphabetical order
  - Value: the list of friends
- Keys will be sorted, a pair of friends go to the same reducer

A → B C D
(A B) → B C D
(A C) → B C D
(A D) → B C D

B → A C D E
(A B) → A C D E
(B C) → A C D E
(B D) → A C D E
(B E) → A C D E

C → A B D E
(A C) → A B D E
(B C) → A B D E
(C D) → A B D E
(C E) → A B D E

…

4. Friends in common – Reduce task

- Reducers receive two lists of friends per pair of people

(A B) → (B C D) (A C D E)
(A C) → (B C D) (A B D E)
(A D) → (B C D) (A B C E)
…

- The reduce function intersects the lists of values and generates the same key

(A B) → (C D)
(A C) → (B D)
(A D) → (B C)
…

- Now, when D visits A's profile we can lookup (A D) to see their common friends

5. References

- *Jeffrey Dean, Sanjay Ghemwat.* MapReduce: Simplified Data processing on Large Clusters
- *Tom White*. Hadoop – The Definitive Guide, 3rd edition
- Apache Hadoop Project. http://hadoop.apache.org/
- Finding friends with MapReduce. http://stevekrenzel.com/finding-friends-with-mapreduce