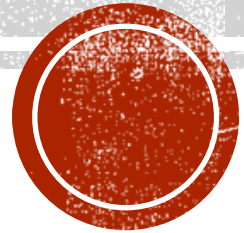
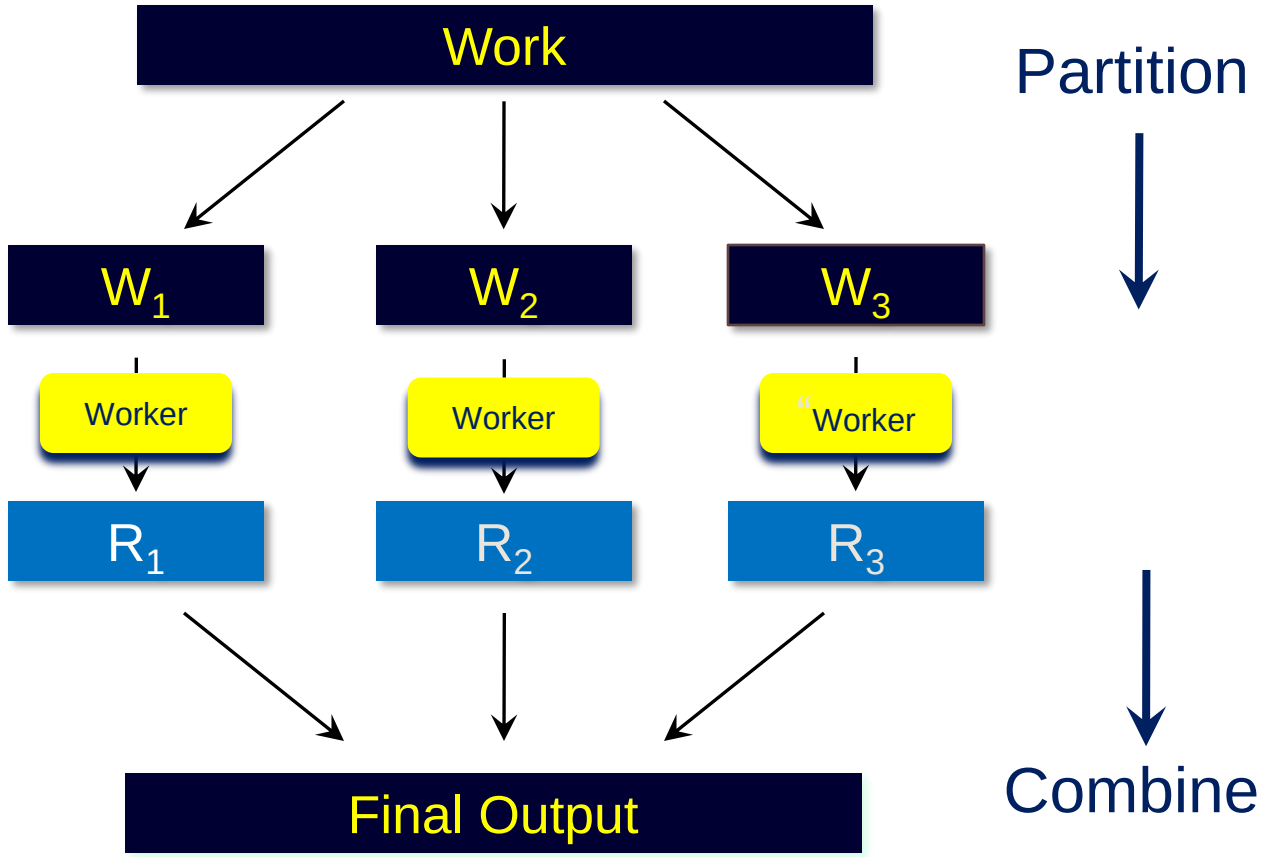
MAPREDUCE – CONT'D



DIVIDE AND CONQUER



PARALLELIZATION CHALLENGES

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers fail?

DISTRIBUTED WORKERS COORDINATION

- Coordinating a large number of workers in a distributed environment is challenging
  - The order in which workers run may be unknown
  - The order in which workers interrupt each other may be unknown
  - The order in which workers access shared data may be unknown

MAPREDUCE

DATA-INTENSIVE PROGRAMMING MODEL

- Users specify the computation in terms of a **map()** and a **reduce()** function
  - **map** (k, v) → <k', v'>*
  - **reduce** (k', v') → <k', v'>*

- Underlying runtime system (YARN + HDFS)
  - Automatically parallelizes the computation across large-scale clusters of machines
  - Handles machine failures, communications and performance issues.

WHAT ARE MAP AND REDUCE FUNCTIONS?

- At a high-level of abstraction, MapReduce codifies a generic "recipe" for processing large data set
  - Iterate over a large number of records
  - Extract something of interest from each
  - Shuffle and sort intermediate results
  - Aggregate intermediate results
  - Generate final output

Map

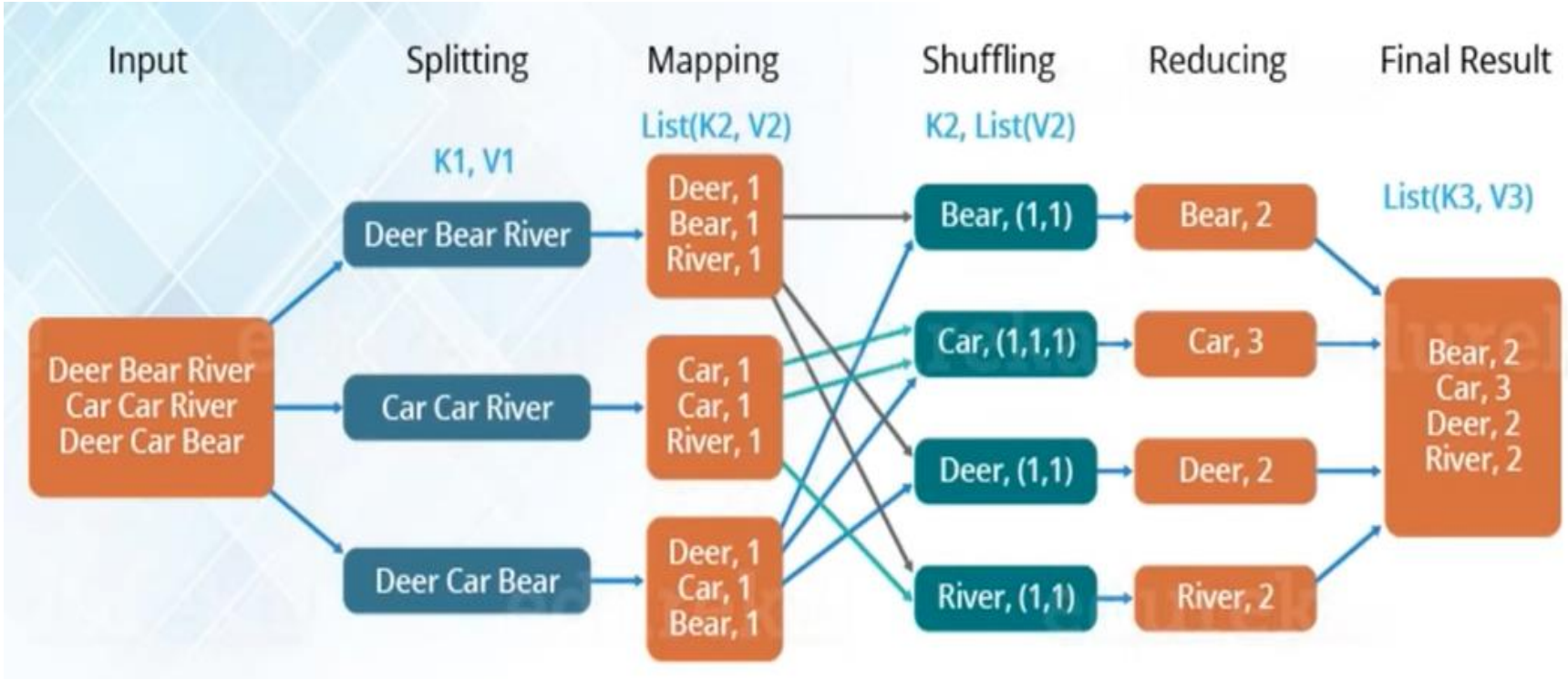
Shuffle

Reduce

Basic Tenet of MapReduce is Enabling a Functional Abstraction for the Map() and Reduce() operations

By default, Hadoop will take care of the Shuffle process for you!

WORDCOUNT EXAMPLE IN HADOOP MAPREDUCE



MAPREDUCE "RUNTIME" BASIC FUNCTIONS (PROVIDED BY HADOOP)

- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles "process distribution"
  - Moves **processes to data**, not data to processes
- Handles synchronization among workers
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults, dynamically
  - Detects worker failures and restarts

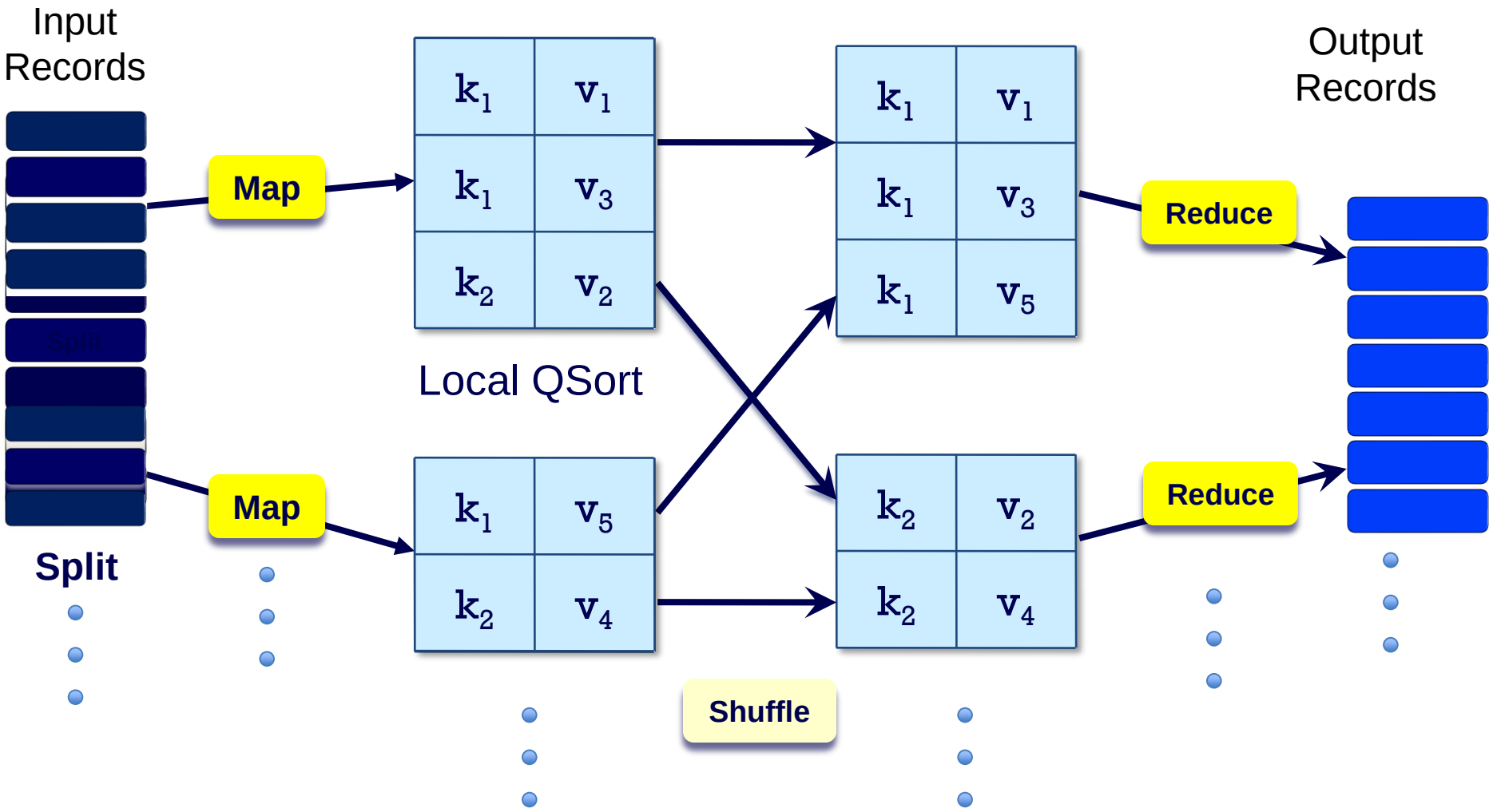
MAPREDUCE DATA FLOW

- A MapReduce job is a unit of work to be performed
  - Job consists of the **MapReduce Program**, the **Input data** and the **Configuration Information**
- The MapReduce job is divided it into two types of tasks – ***map*** tasks and ***reduce*** tasks
- The Input data is divided into fixed-size pieces called ***splits***
  - One map task is created for each split
  - The map function is run on each split
- Configuration information indicates where the input lies and the output is stored

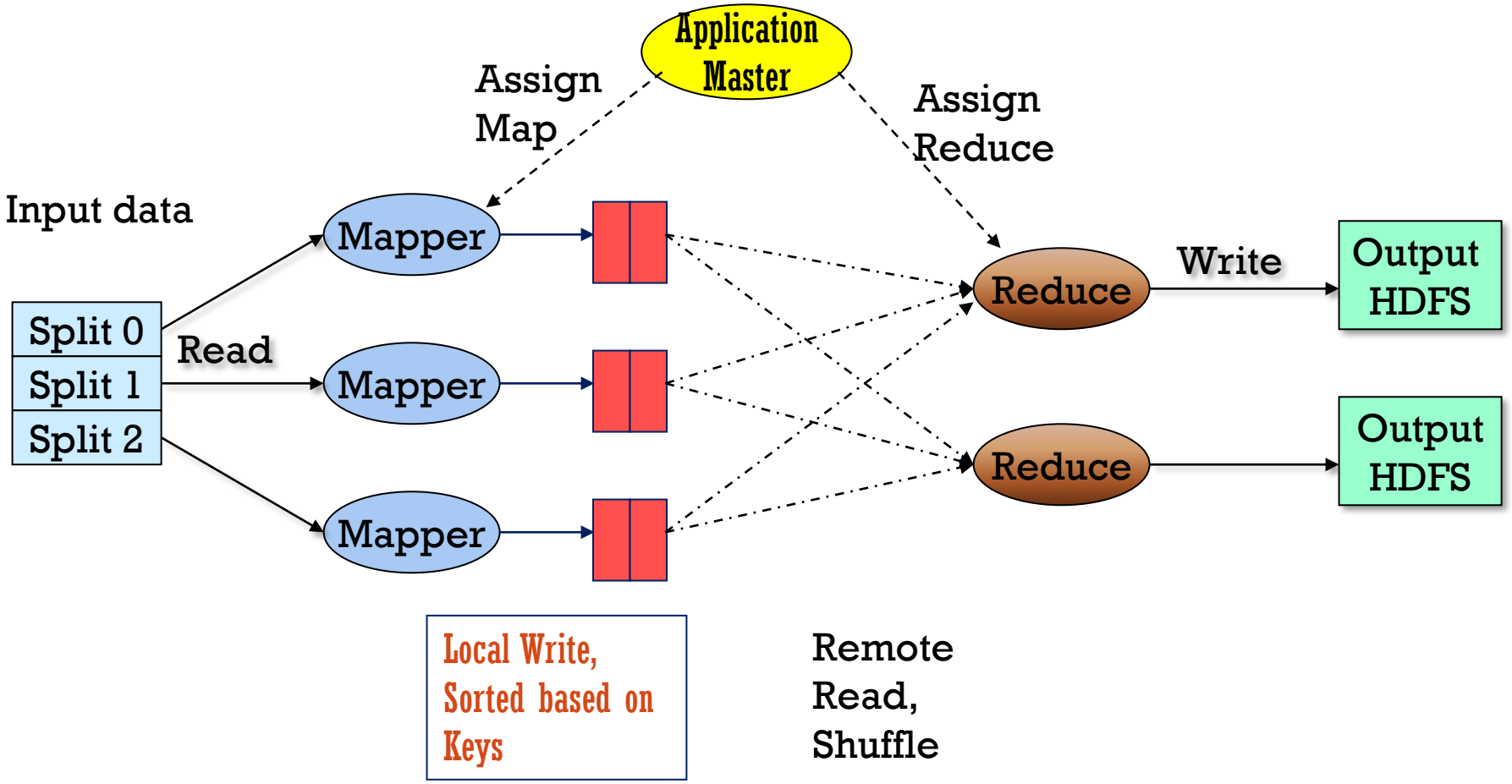
FROM MAP TO REDUCE

- Select a number R of reduce tasks.
- Divide the intermediate keys into R groups,
  - Use an efficient hashing function
- Each Map task creates, at its own processor, R files of intermediate key-value pairs, sorted by key, and one for each Reduce task
- Its output file is transfer to the Reducer's container
- Once all Mapper tasks results are transferred to the Reducer, it merges the results and compute the final output
- The Reducer then writes the output to HDFS

MAPREDUCE DATA FLOW



DISTRIBUTED EXECUTION OVERVIEW



READING

- https://www.guru99.com/introduction-to-mapreduce.html