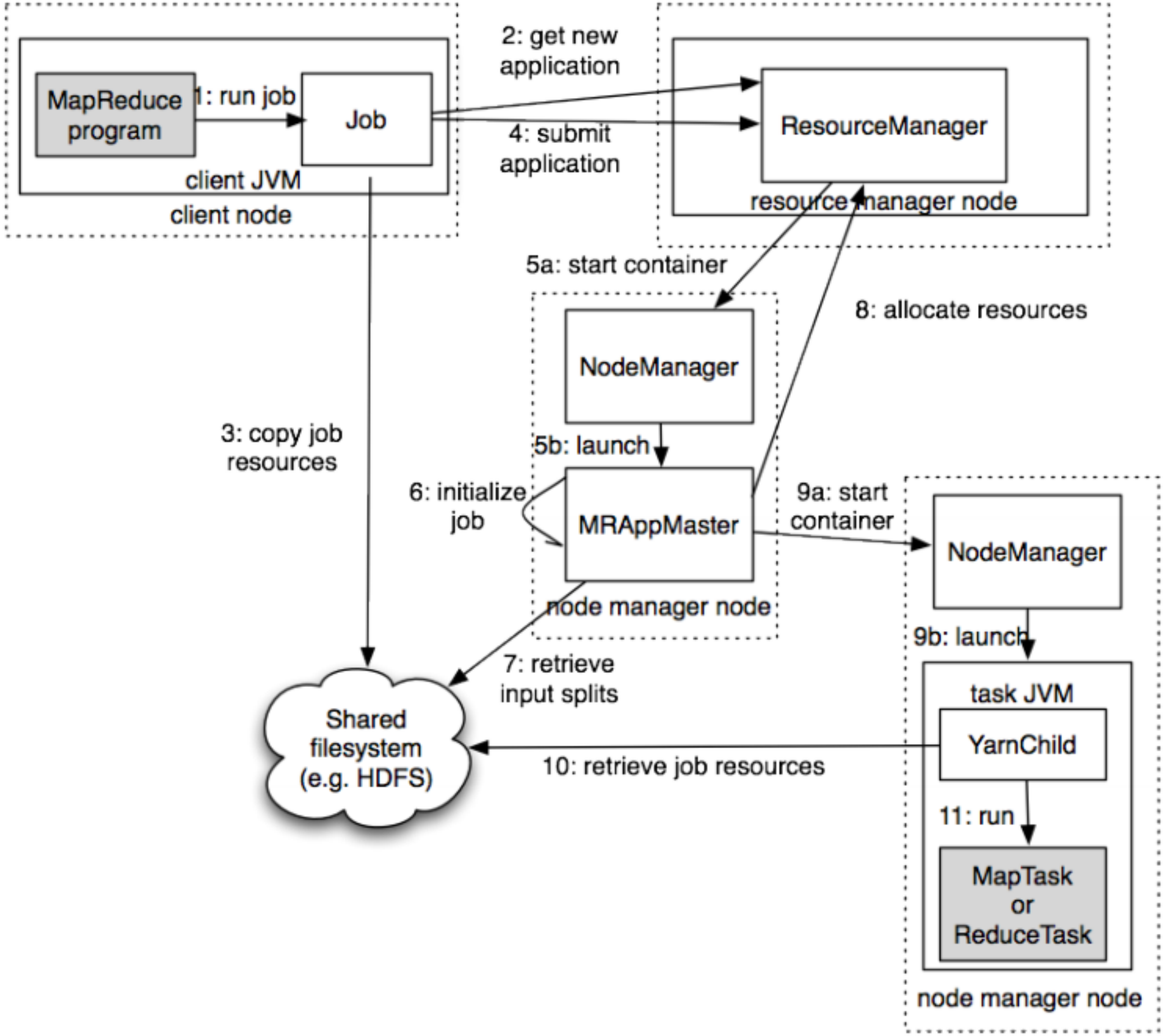
CS60021: Scalable Data Mining

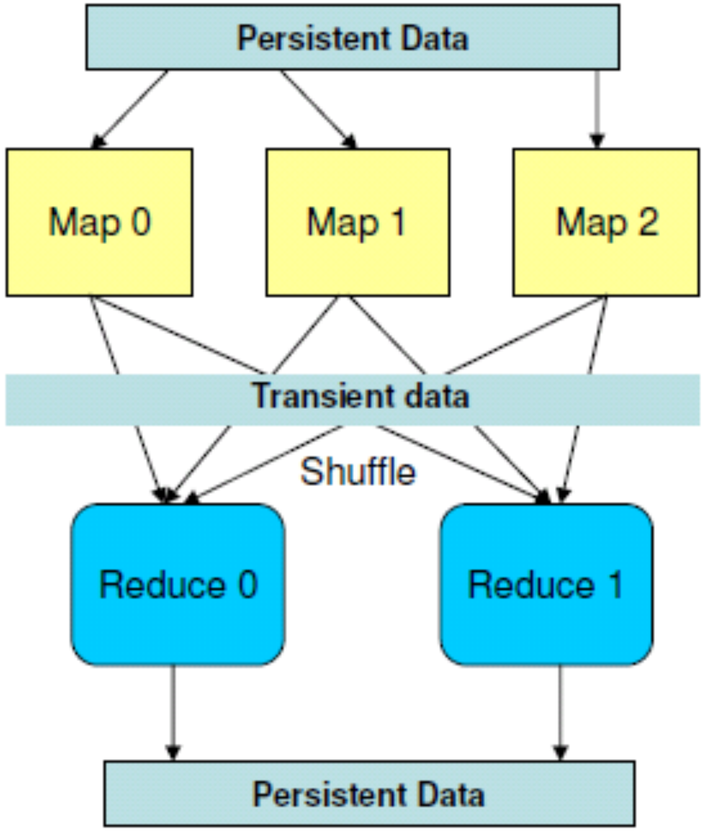
Sourangshu Bhattacharya

Hadoop(v2) MR job

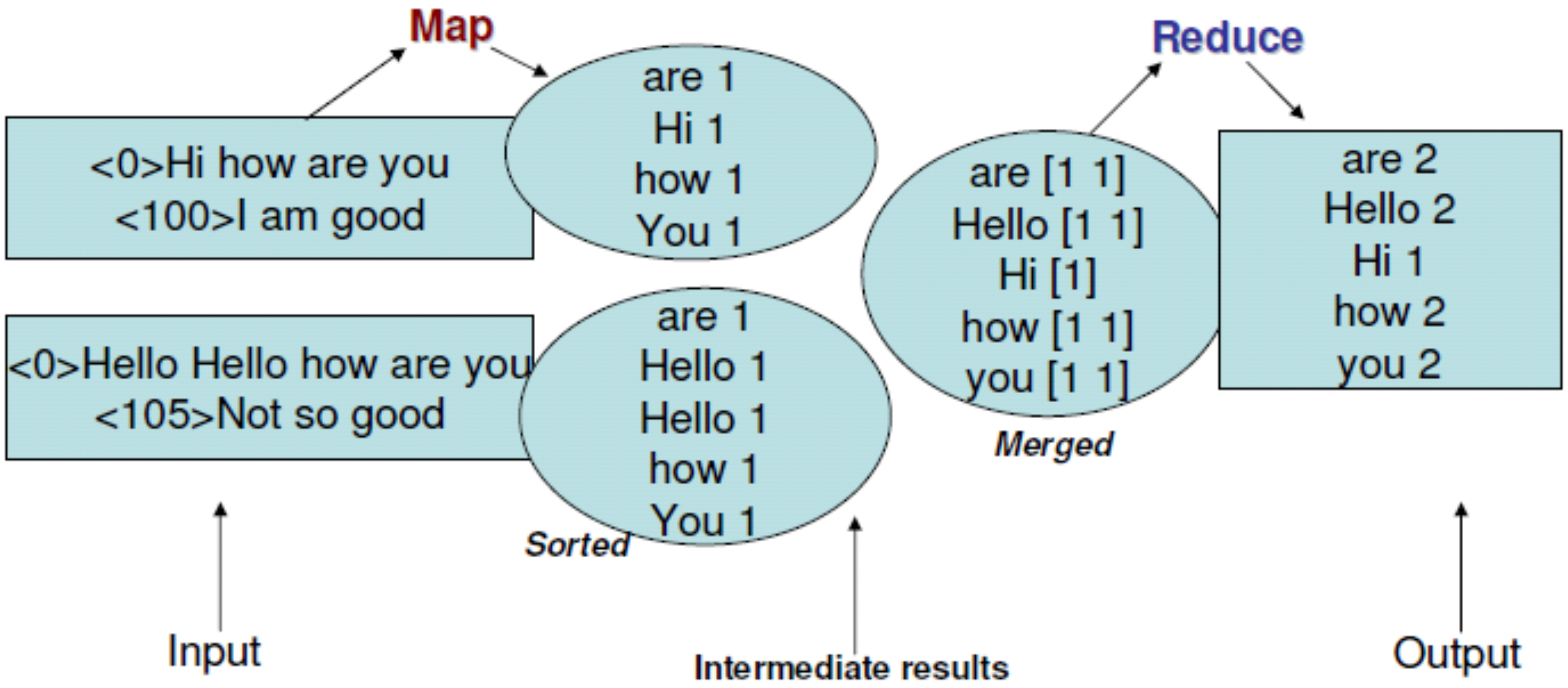


Source: Hadoop: The Definitive Guide

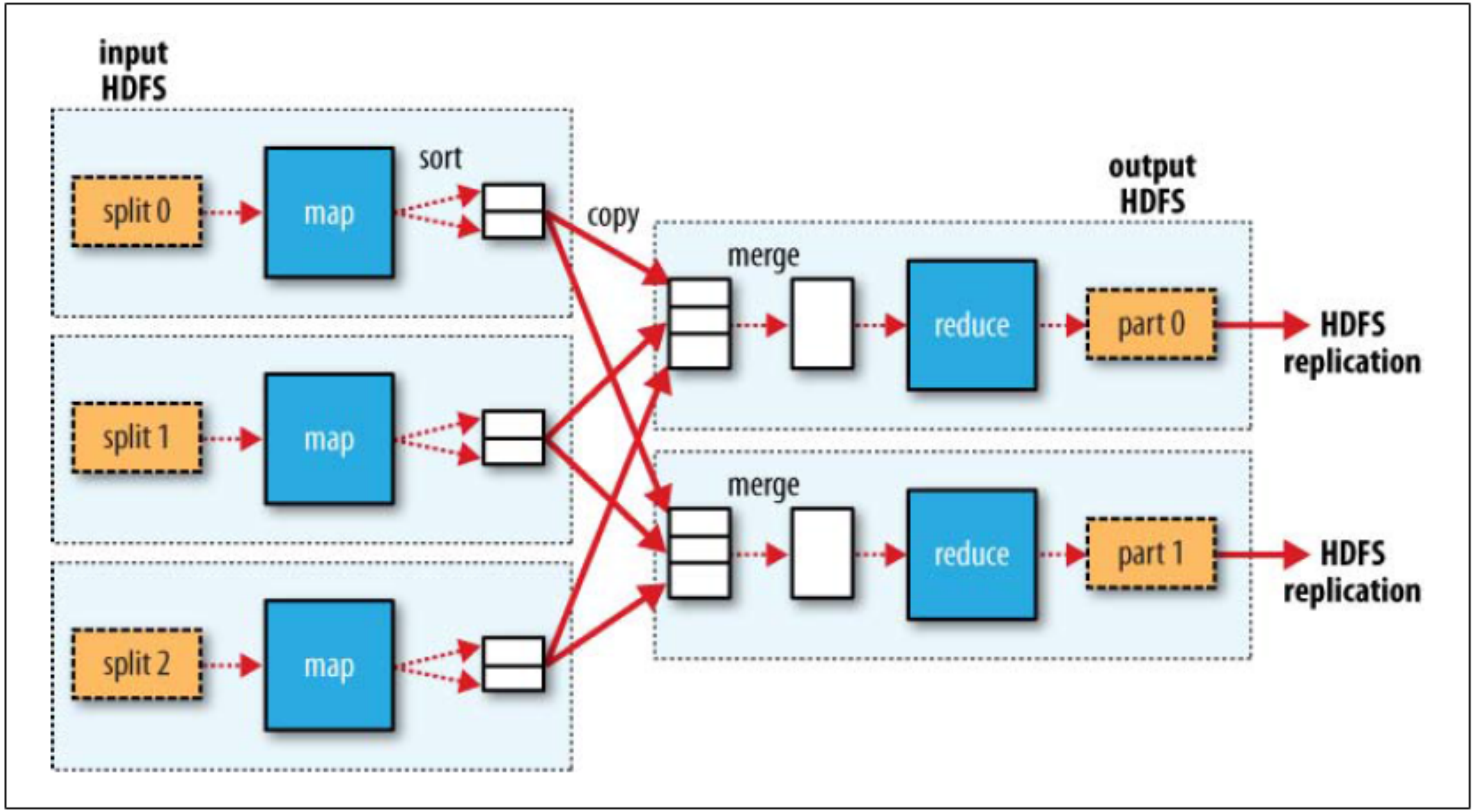
Map Reduce Data Flow



Data: Stream of keys and values

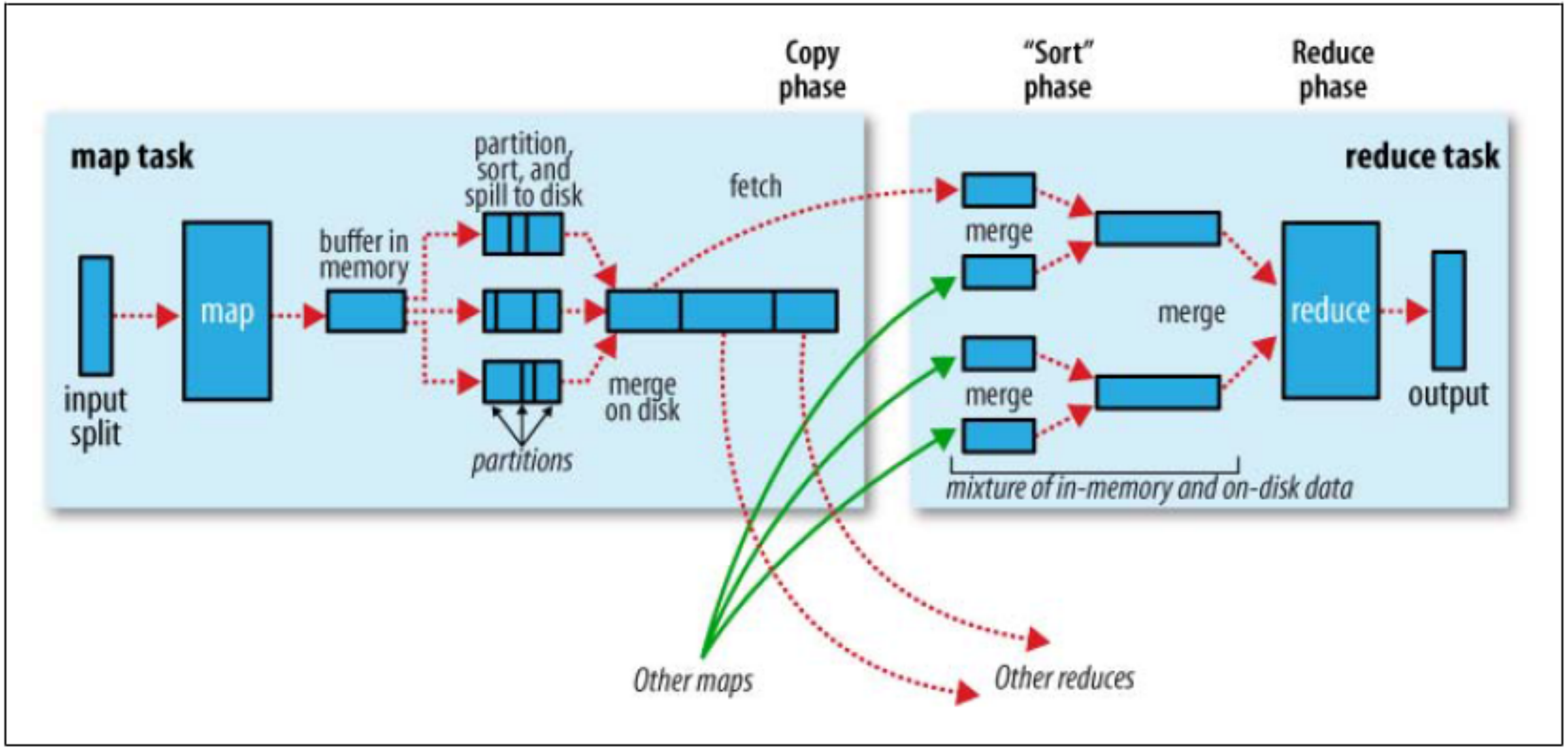


Hadoop MR Data Flow



Source: Hadoop: The Definitive Guide

Shuffle and sort



Source: Hadoop: The Definitive Guide

Data Flow

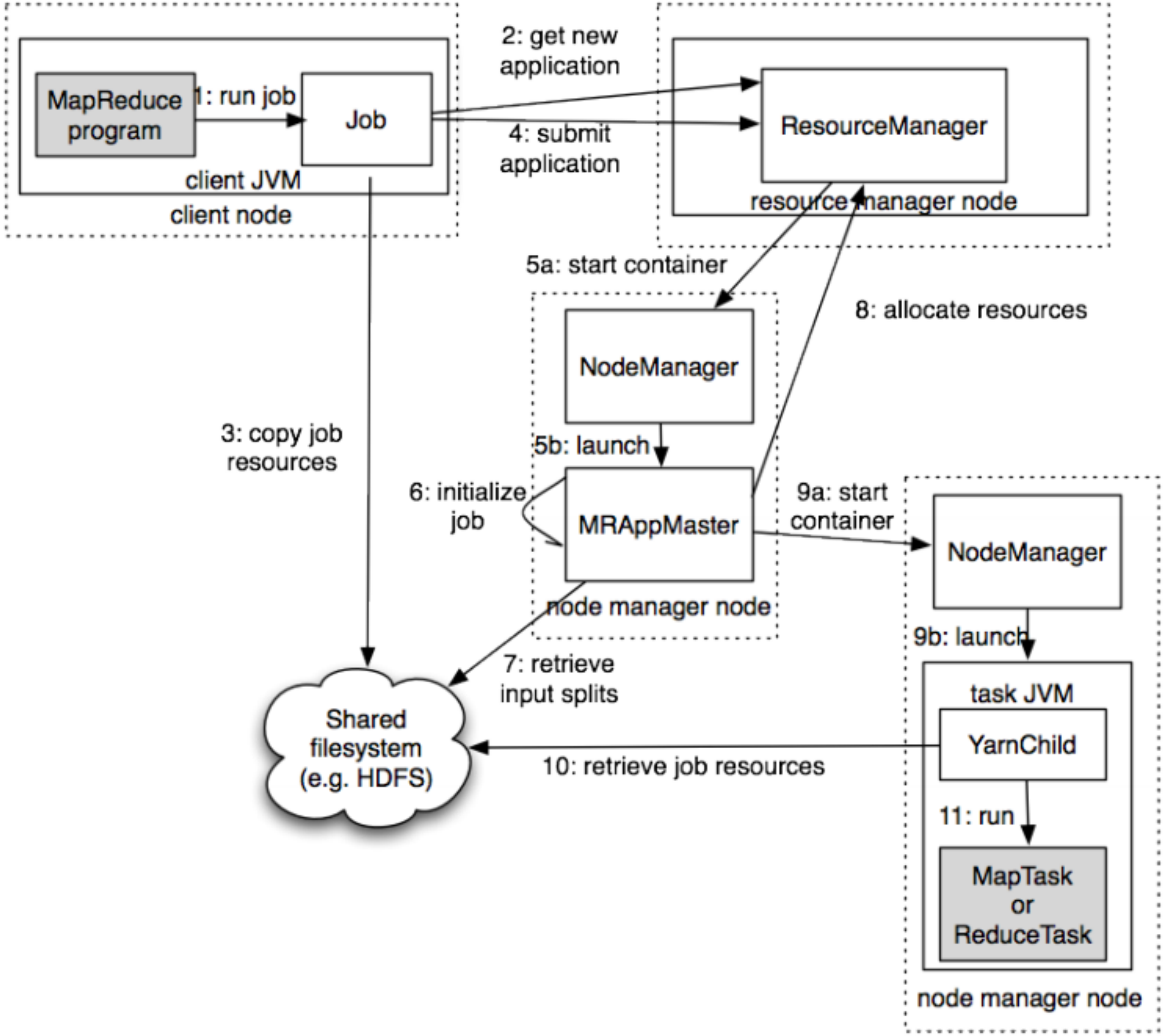
- **Input and final output are stored on a distributed file system (FS):**
  - Scheduler tries to schedule map tasks "close" to physical storage location of input data

- **Intermediate results are stored on local FS of Map workers.**

- **Output** of **Reduce** workers are stored on a **distributed file system**.

- **Output is often input to another MapReduce task**

Hadoop(v2) MR job



Source: Hadoop: The Definitive Guide

Fault tolerance

- Provides scalability and cost effectiveness

- HDFS:
  - Replication

- Map Reduce
  - Restarting failed tasks: map and reduce
  - Writing map output to FS
  - Minimizes re-computation

Coordination: Master

- Master node takes care of coordination:
  - **Task status:** (idle, in-progress, completed)
  - **Idle tasks** get scheduled as workers become available
  - When a map task completes, it sends the master the location and sizes of its *R* intermediate files, one for each reducer
  - Master pushes this info to reducers
- Master pings workers periodically to detect failures

Failures

- ❑ Task failure
  - ❑ Task has failed - report error to node manager, appmaster, client.
  - ❑ Task not responsive, JVM failure - Node manager restarts tasks.

- ❑ Application Master failure
  - ❑ Application master sends heartbeats to resource manager.
  - ❑ If not received, the resource manager retrieves job history of the run tasks.

- ❑ Node manager failure
  - ❑ Restart

Dealing with Failures

- **Map worker failure**
  - Map tasks completed or in-progress at worker are reset to idle
  - Reduce workers are notified when task is rescheduled on another worker

- **Reduce worker failure**
  - Only in-progress tasks are reset to idle
  - Reduce task is restarted

- **Master failure**
  - MapReduce task is aborted and client is notified

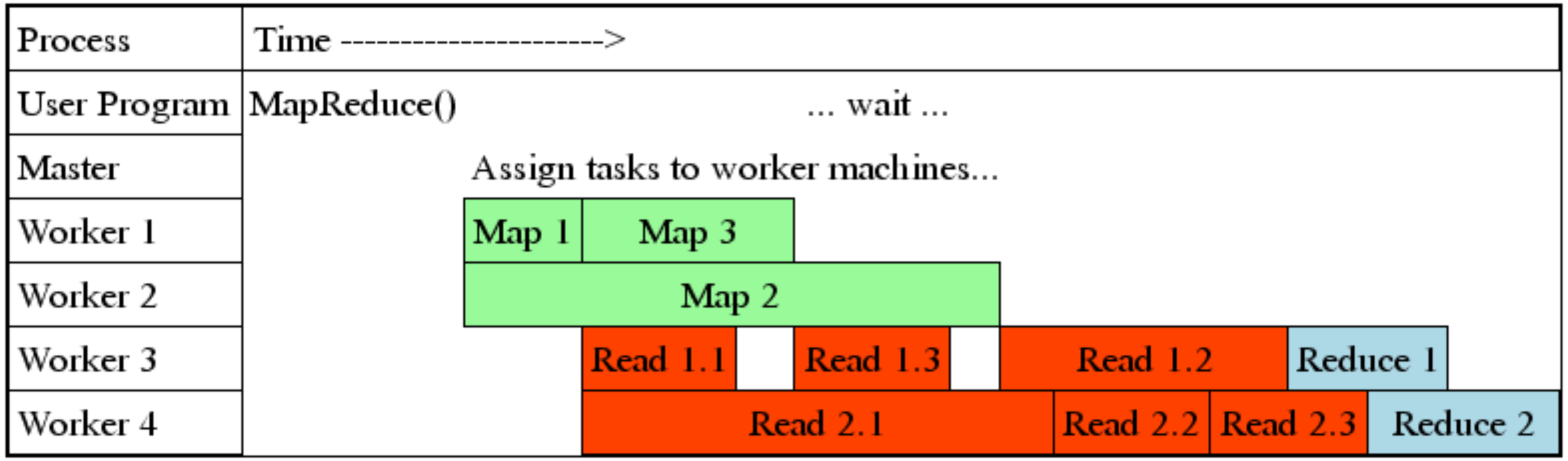
How many Map and Reduce jobs?

- *M* map tasks, *R* reduce tasks
- **Rule of a thumb:**
  - Make *M* much larger than the number of nodes in the cluster
  - One DFS chunk per map is common
  - Improves dynamic load balancing and speeds up recovery from worker failures

- **Usually *R* is smaller than *M***
  - Because output is spread across *R* files

Task Granularity & Pipelining

- **Fine granularity tasks:** map tasks >> machines
  - Minimizes time for fault recovery
  - Can do pipeline shuffling with map execution
  - Better dynamic load balancing



Process	Time ---------------------->
User Program	MapReduce() ... wait ...
Master	Assign tasks to worker machines...
Worker 1	Map 1 Map 3
Worker 2	Map 2
Worker 3	Read 1.1 Read 1.3 Read 1.2 Reduce 1
Worker 4	Read 2.1 Read 2.2 Read 2.3 Reduce 2

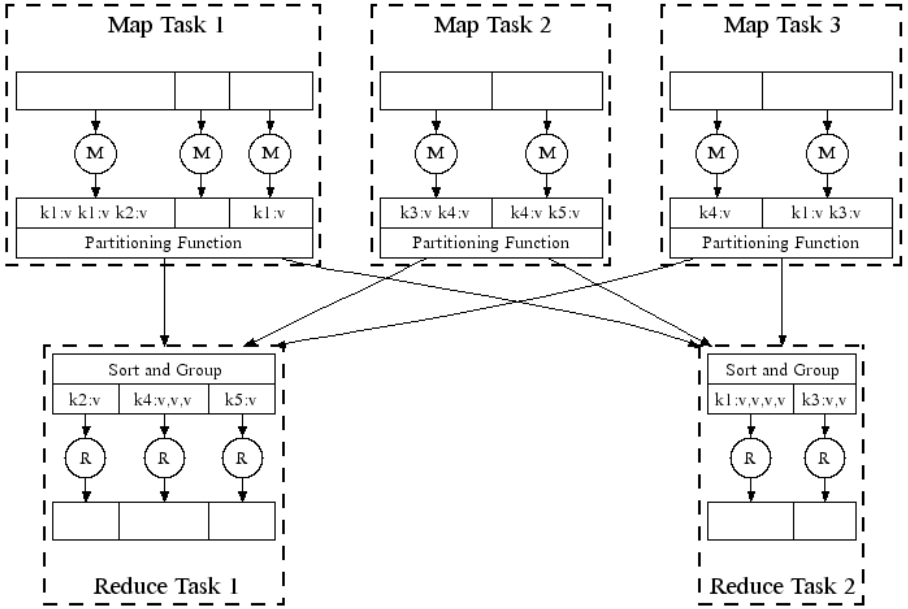
Refinements: Backup Tasks

- Problem
  - Slow workers significantly lengthen the job completion time:
    - Other jobs on the machine
    - Bad disks
    - Weird things
- Solution
  - Near end of phase, spawn backup copies of tasks
    - Whichever one finishes first "wins"
- Effect
  - Dramatically shortens job completion time

Refinement: Combiners

- Often a Map task will produce many pairs of the form *(k,v_1), (k,v_2)*, … for the same key *k*
  - E.g., popular words in the word count example

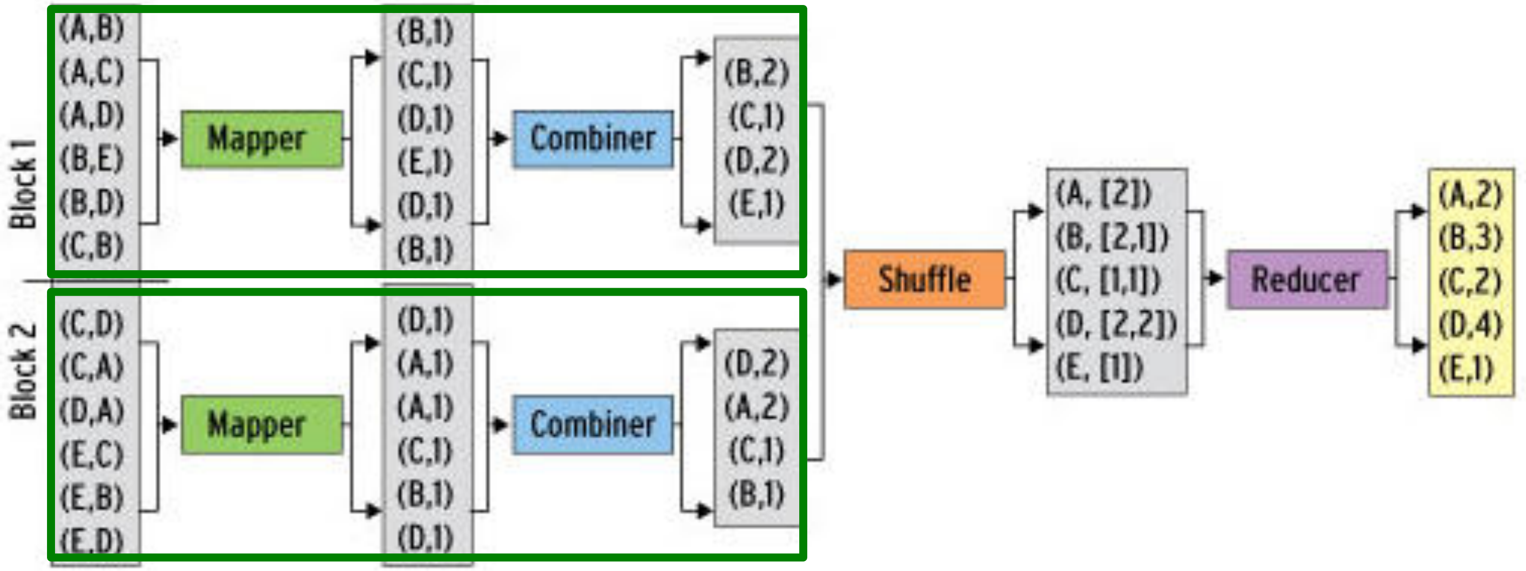
- **Can save network time by pre-aggregating values in the mapper:**
  - combine(k, list(v_1)) → v_2
  - Combiner is usually same as the reduce function



- Works only if reduce function is commutative and associative

Refinement: Combiners

- **Back to our word counting example:**
  - Combiner combines the values of all keys of a single mapper (single machine):



  - Much less data needs to be copied and shuffled!

Refinement: Partition Function

- **Want to control how keys get partitioned**
  - Inputs to map tasks are created by contiguous splits of input file
  - Reduce needs to ensure that records with the same intermediate key end up at the same worker

- **System uses a default partition function:**
  - **hash(key) mod *R***

- **Sometimes useful to override the hash function:**
  - E.g., **hash(hostname(URL)) mod *R*** ensures URLs from a host end up in the same output file

References:

- Jure Leskovec, Anand Rajaraman, Jeff Ullman. **Mining of Massive Datasets.** *2nd edition. - Cambridge University Press.* http://www.mmds.org/

- Tom White. **Hadoop: The definitive Guide.** Oreilly Press.