Map-Reduce

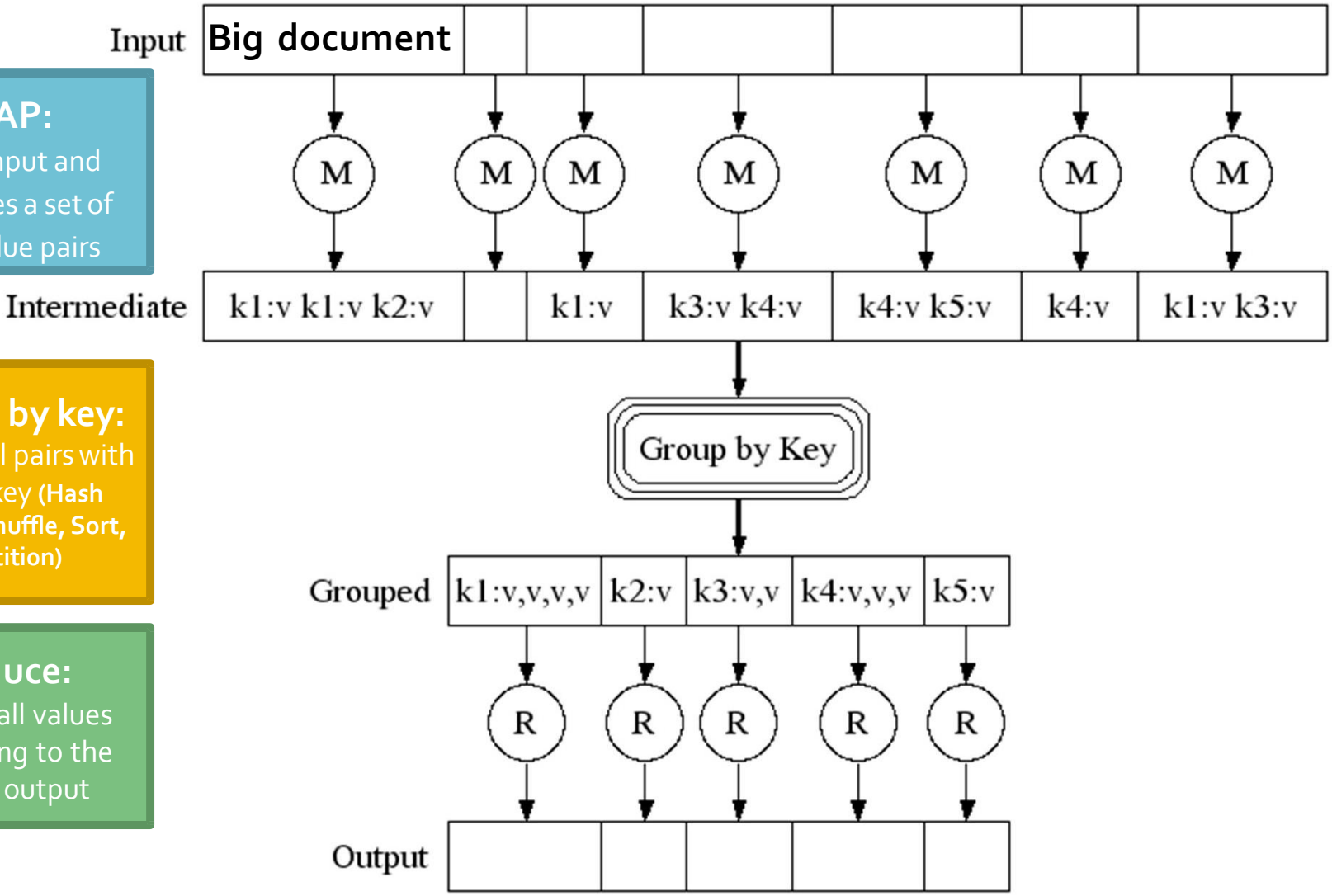
Scheduling and Data Flow

Map-Reduce: A diagram

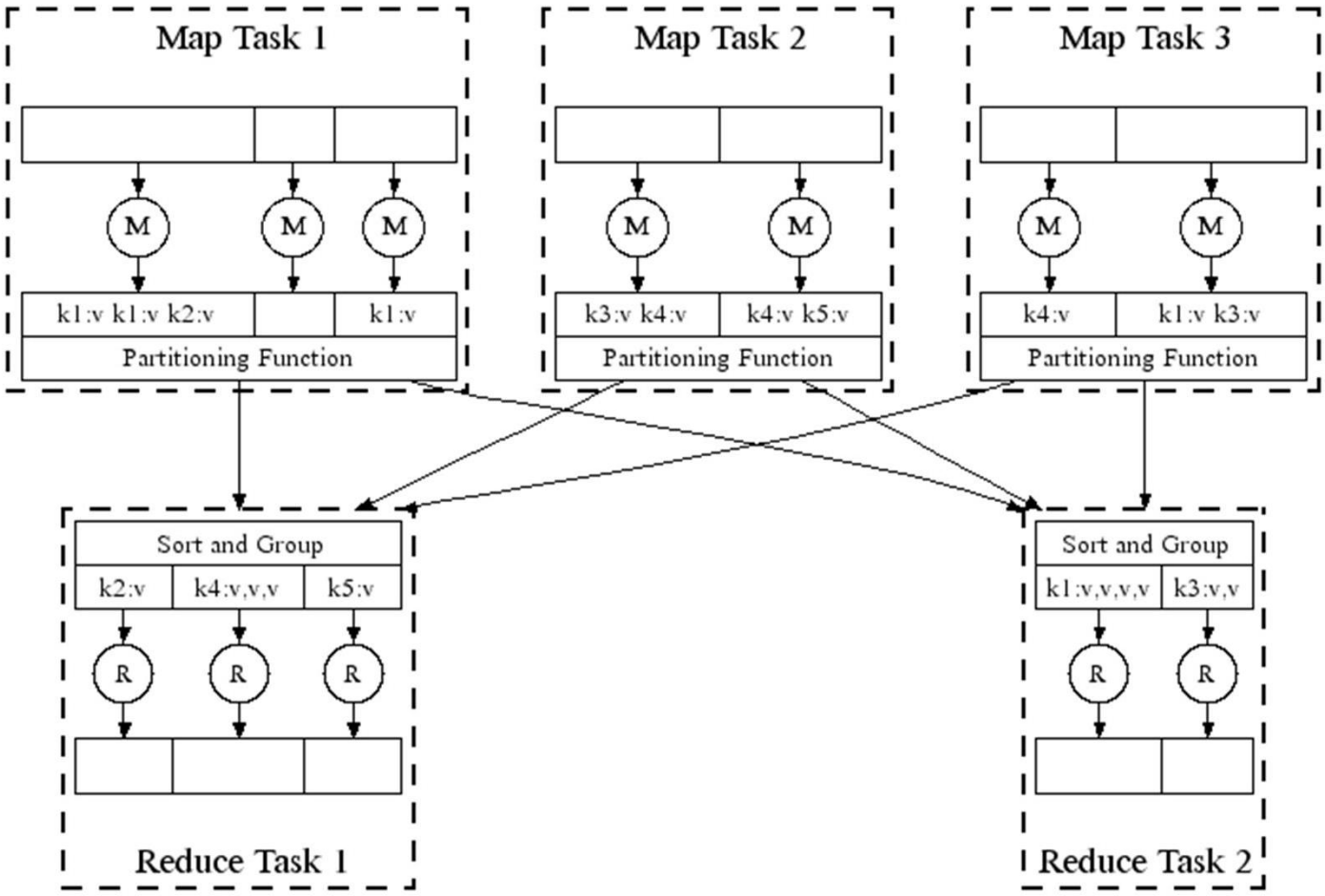
MAP:
Read input and produces a set of key-value pairs

Group by key:
Collect all pairs with same key **(Hash merge, Shuffle, Sort, Partition)**

Reduce:
Collect all values belonging to the key and output



Map-Reduce: In Parallel



All phases are distributed with many tasks doing the work

Map-Reduce: Environment

Map-Reduce environment takes care of:

- Partitioning the input data
- Scheduling the program's execution across a set of machines
- Performing the **group by key** step
- Handling node failures
- Managing required inter-machine communication

Data Flow

- Input and final output are stored on the distributed file system (DFS):
  - Scheduler tries to schedule map tasks "close" to physical storage location of input data
- Intermediate results are stored on local FS of Map and Reduce workers
- Output is often input to another MapReduce task

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

Dealing with Failures

- **Map worker failure**
  - Map tasks completed or in-progress at worker are reset to idle.
  - Idle tasks eventually rescheduled on other worker(s)
- **Reduce worker failure**
  - Only in-progress tasks are reset to idle
  - Idle Reduce tasks restarted on other worker(s)
- **Master failure**
  - MapReduce task is aborted and client is notified

How many Map and Reduce jobs?

- *M* map tasks, *R* reduce tasks
- **Rule of 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

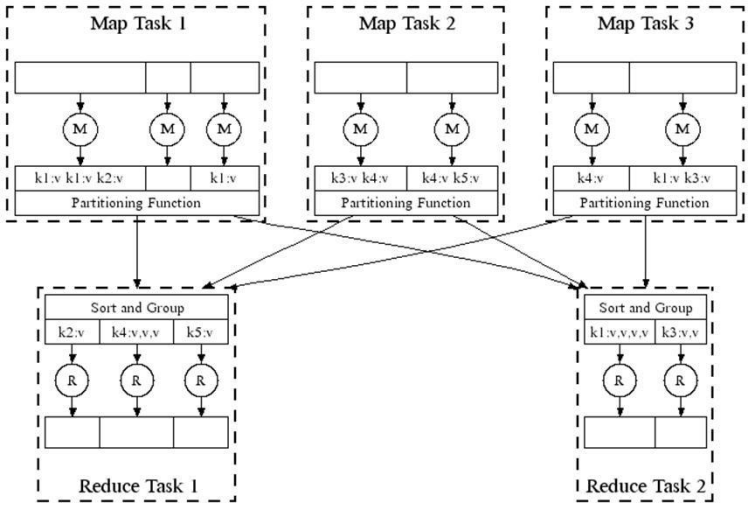
Map-Reduce

Refinements
Implementations

Refinement: Combiners (1)

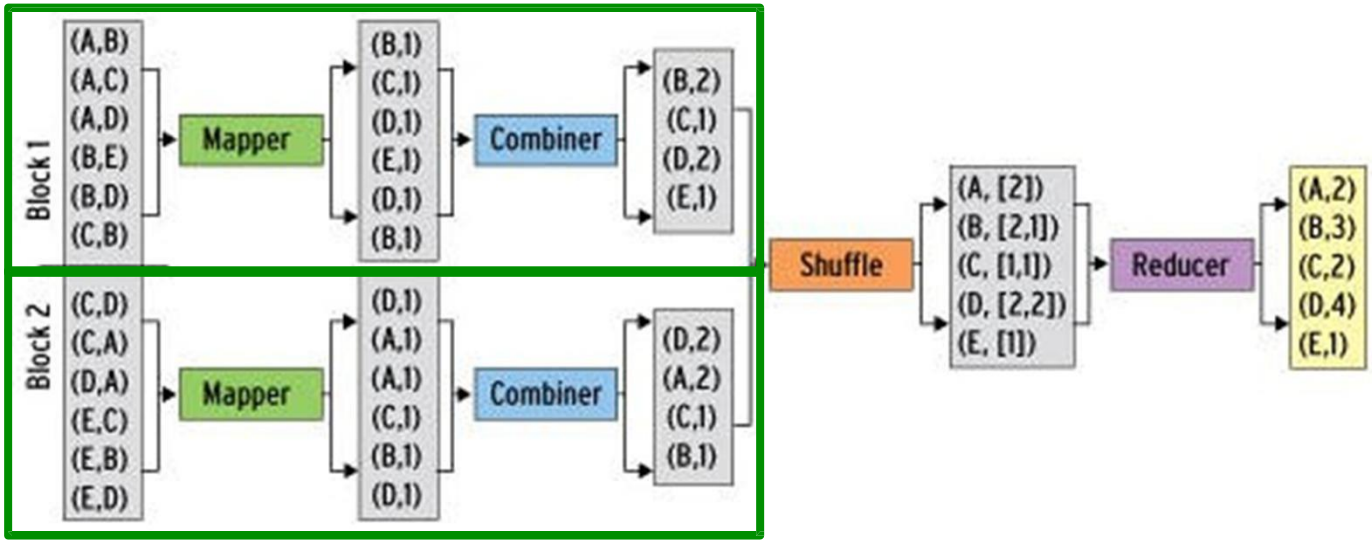
- Often a Map task will produce many pairs of the form $(k,v_1), (k,v_2), \ldots$ for the same key k
  - E.g., popular words in the word count example

- **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



Refinement: Combiners (2)

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



  - Much less data needs to be copied and shuffled!

Refinement: Combiners (3)

- Combiner trick works only if reduce function is commutative and associative.
- Sum

$$2 + (5 + 7) = (2 + 5) + 7$$

- Average

- Median

Refinement: Partition Function

- **Want to control how keys get partitioned**
  - The set of keys that go to a single reduce 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

Pointers and Further Reading

Reading

- Jeffrey Dean and Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters
  - http://labs.google.com/papers/mapreduce.html
- Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung: The Google File System
  - http://labs.google.com/papers/gfs.html

Resources

- Hadoop Wiki
  - Introduction
    - http://wiki.apache.org/lucene-hadoop/
  - Getting Started
    - http://wiki.apache.org/lucene-hadoop/GettingStartedWithHadoop
  - Map/Reduce Overview
    - http://wiki.apache.org/lucene-hadoop/HadoopMapReduce
    - http://wiki.apache.org/lucene-hadoop/HadoopMapRedClasses
  - Eclipse Environment
    - http://wiki.apache.org/lucene-hadoop/EclipseEnvironment
- Javadoc
  - http://lucene.apache.org/hadoop/docs/api/

Resources

- Releases from Apache download mirrors
  - http://www.apache.org/dyn/closer.cgi/lucene/hadoop/
- Nightly builds of source
  - http://people.apache.org/dist/lucene/hadoop/nightly/
- Source code from subversion
  - http://lucene.apache.org/hadoop/version_control.html