CSCI 5673
Distributed Systems

Lecture Set Eleven

MapReduce and Hadoop

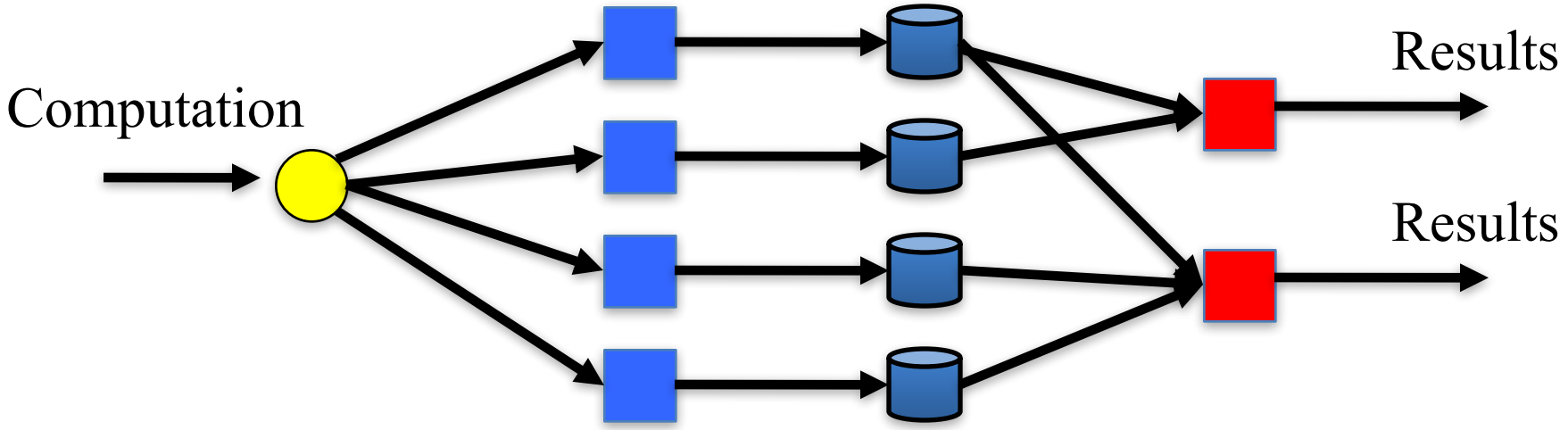
Lecture Notes by
Shivakant Mishra
Computer Science, CU Boulder
Last update: March 07, 2017

Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004.

Hadoop: *http://hadoop.apache.org/*

Google MapReduce

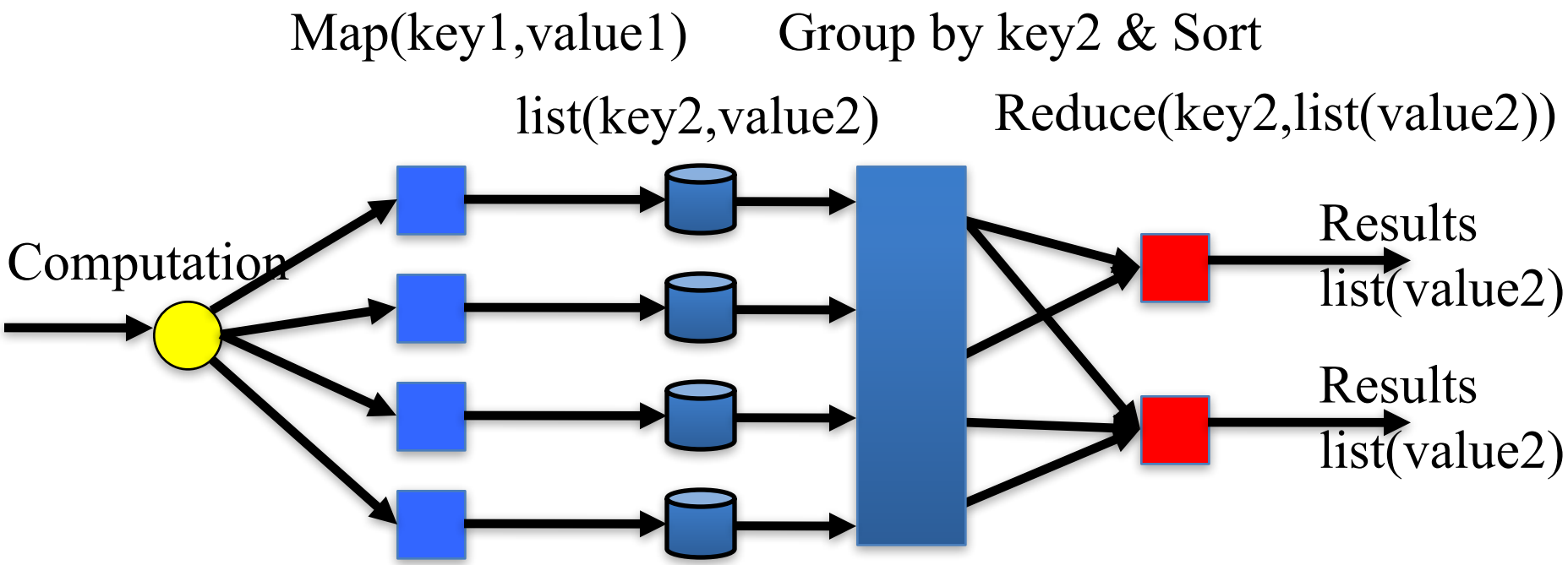
- Want a general way of specifying and automatically parallelizing data computations
  - Distribute this over a large # of commodity PCs
  - General approach: spread then recombine



Example

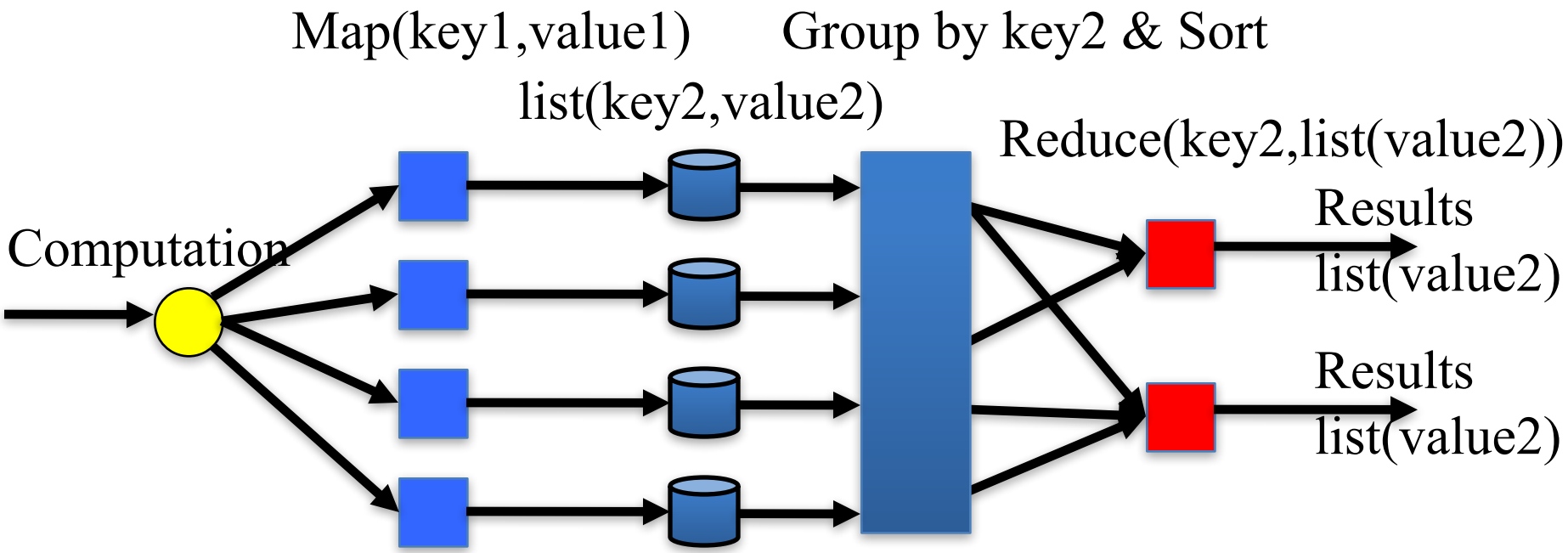
- Count word occurrences in a really big file(s)
  - Could do this on a single PC with a single disk – slow!
  - Single PC with large array of disks – PC is a computation bottleneck
  - Instead, spread the computation over many PCs

MapReduce



- Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs.
- The MapReduce library groups together all intermediate values for each intermediate key I
- The Reduce function, also written by the user, merges values for each key I, streamed by iterator

MapReduce

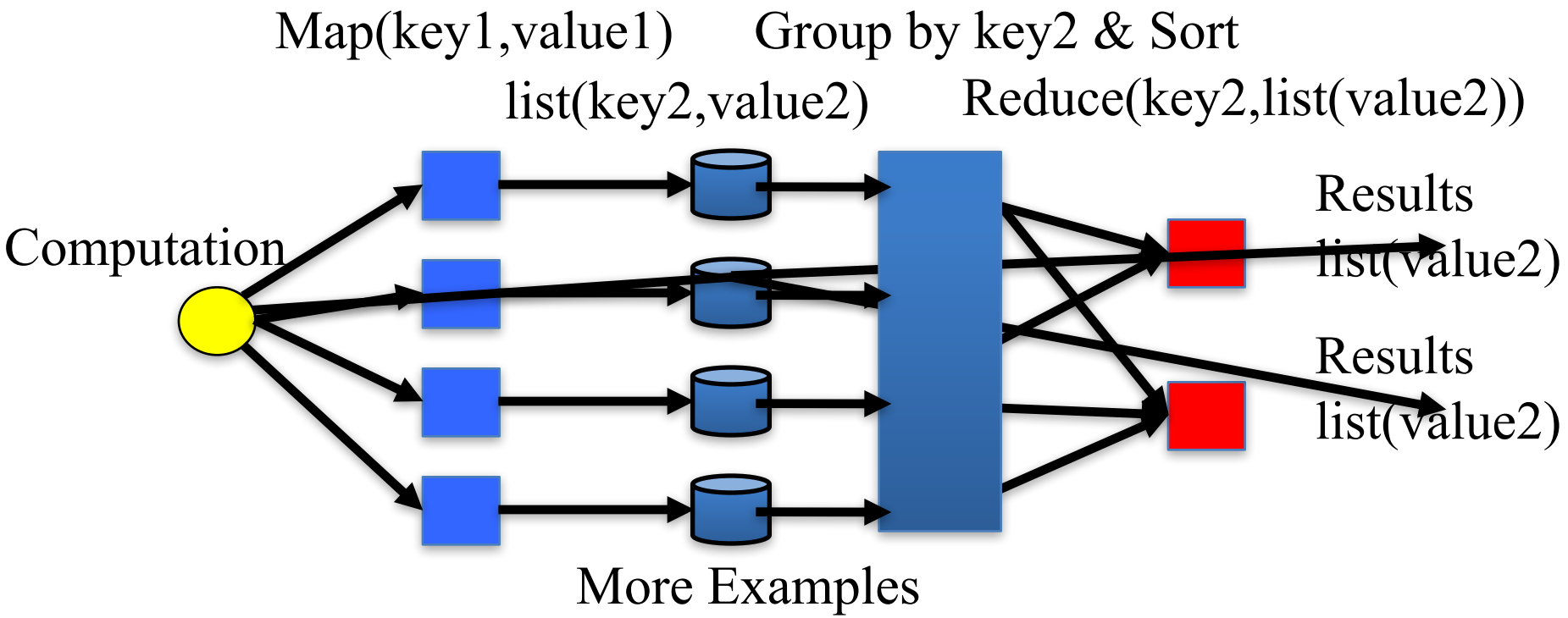


Example: Count Word Occurrences in Really Big File(s)

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

MapReduce



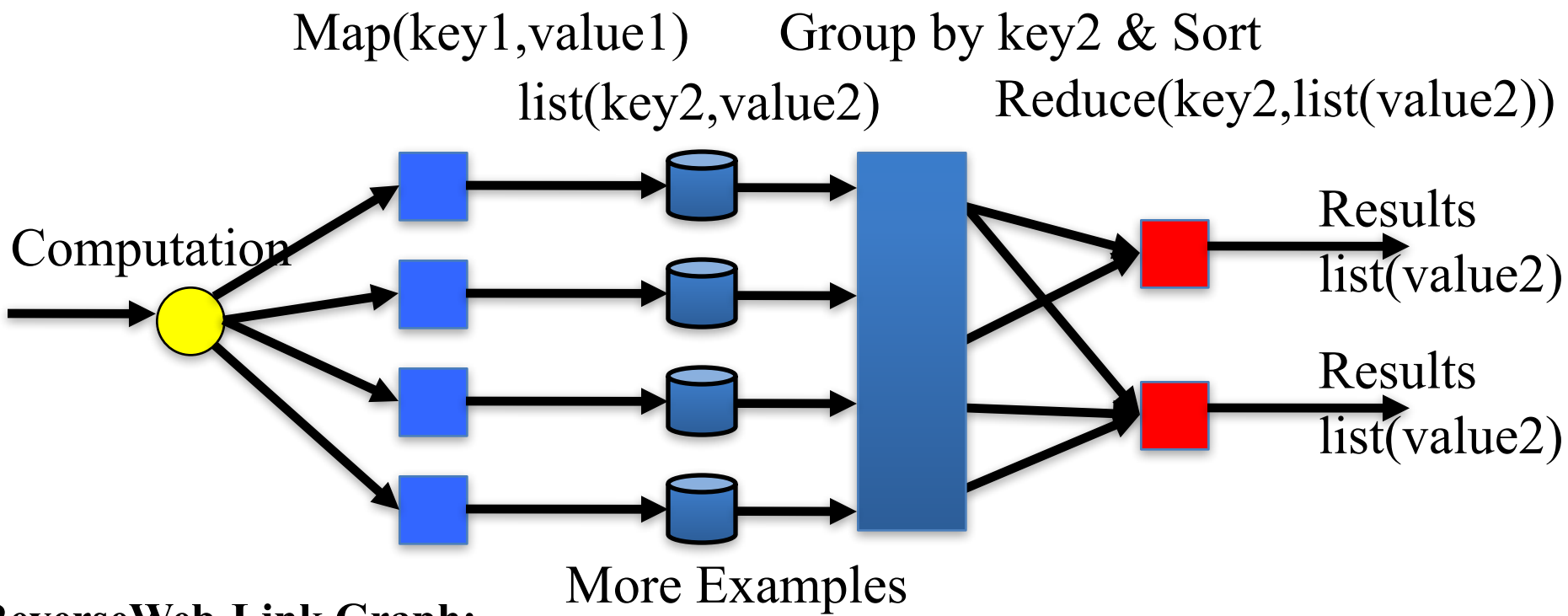
More Examples

- **Distributed Grep:**
  - The map function emits a line if it matches a supplied pattern.
  - The reduce function is an identity function that just copies the supplied intermediate data to the output.

Count of URL Access Frequency:

- The map function processes logs of web page requests and outputs <URL, 1>
- The reduce function adds together all values for the same URL and emits a <URL, total count> pair.

MapReduce



More Examples

ReverseWeb-Link Graph:

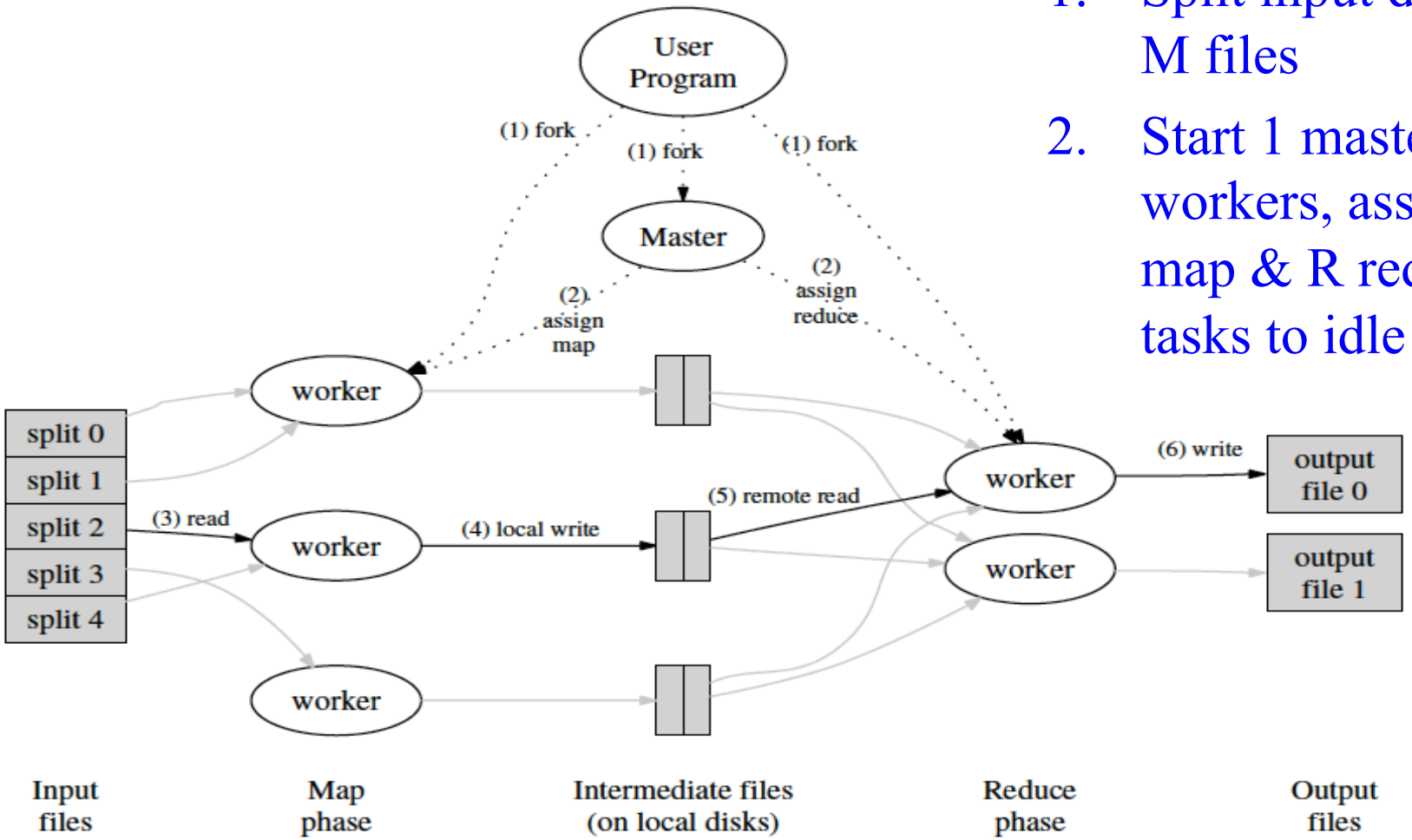
- The map function outputs <target, source> pairs for each link to a target URL found in a page named source.
- The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: <target, list(source)>

Distributed Sort:

- The map function extracts the key from each record, and emits a <key, record> pair.
- The reduce function emits all pairs unchanged.

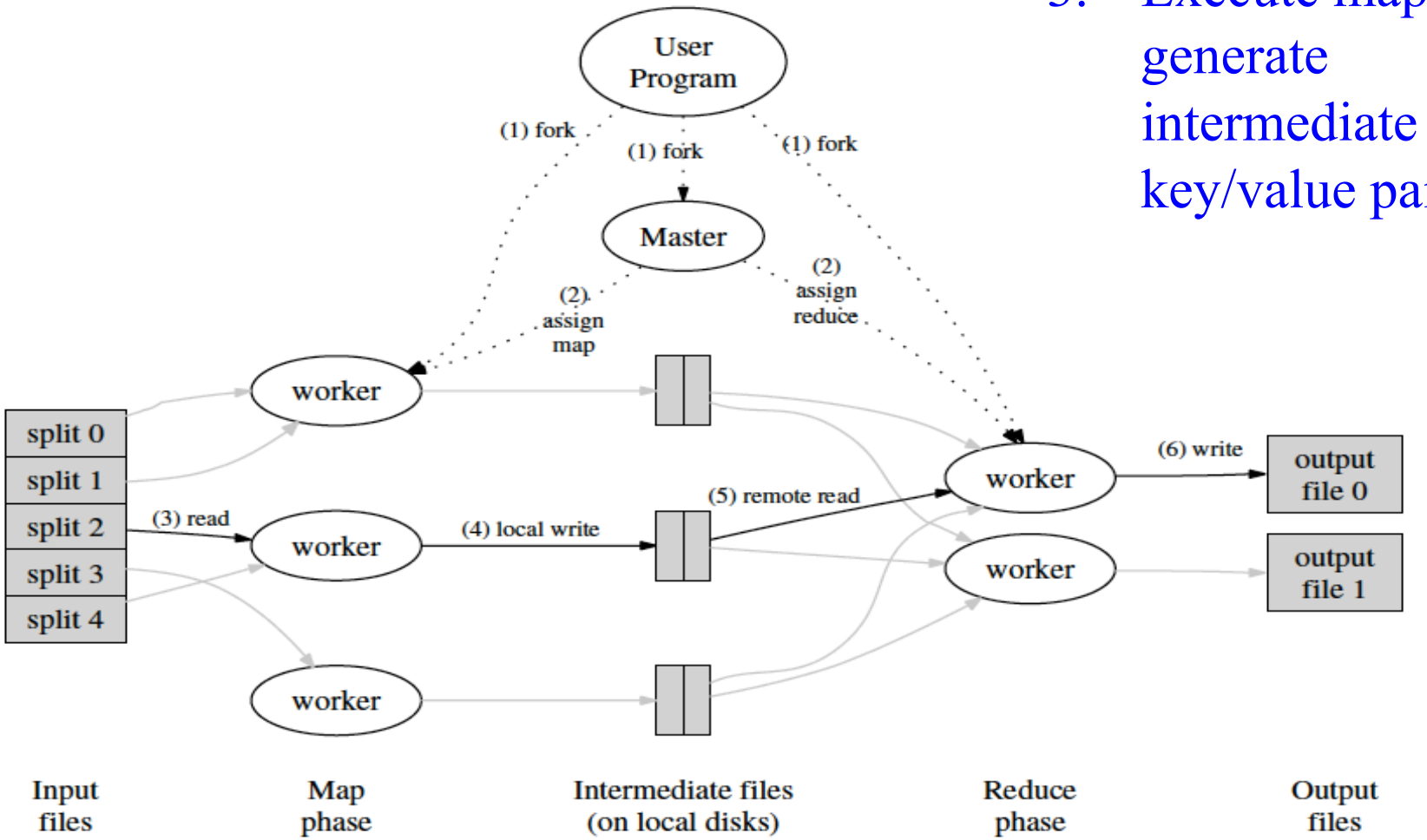
MapReduce Execution Overview

1. Split input data into M files
2. Start 1 master, many workers, assign M map & R reduce tasks to idle workers



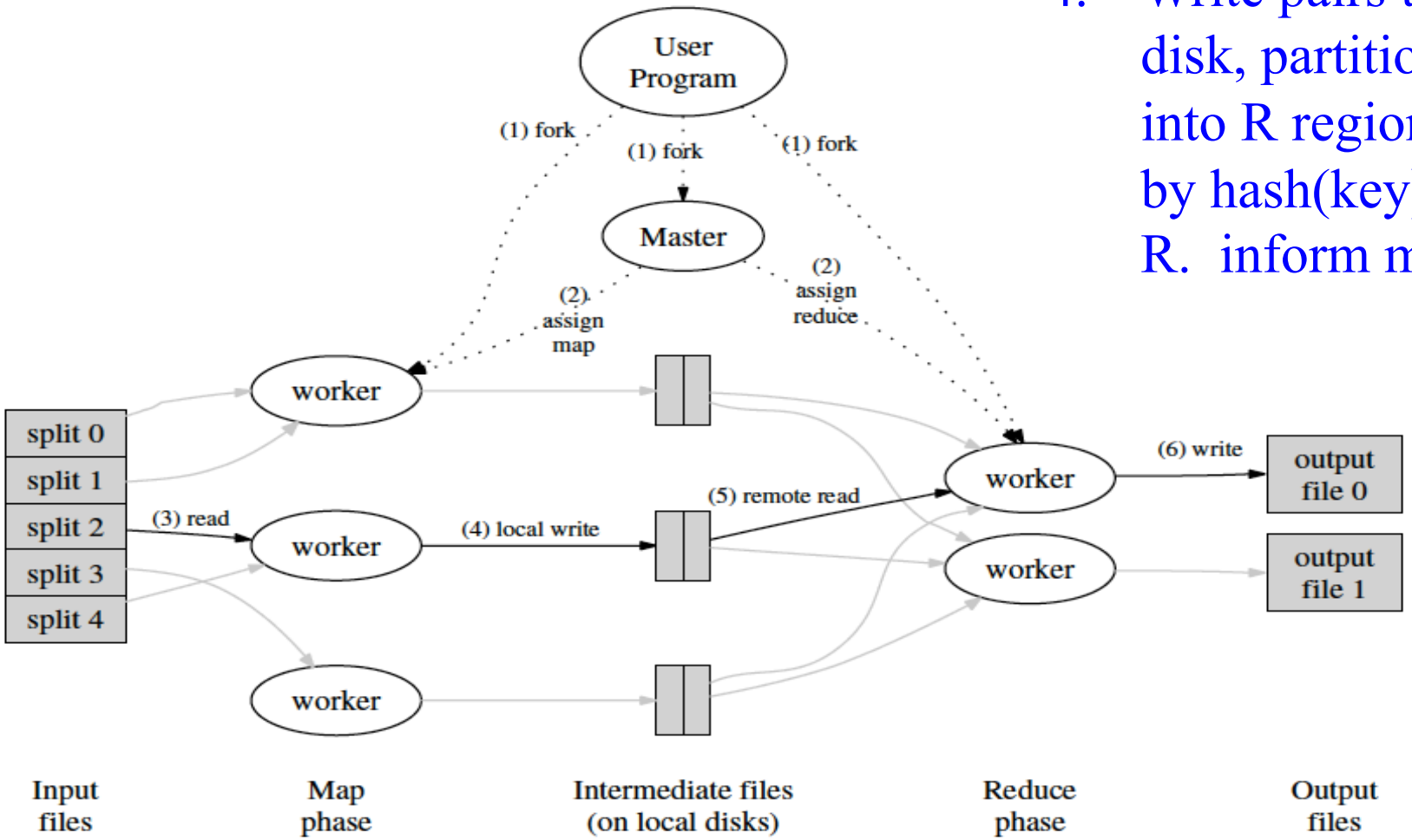
MapReduce Execution Overview

3. Execute map task to generate intermediate key/value pairs



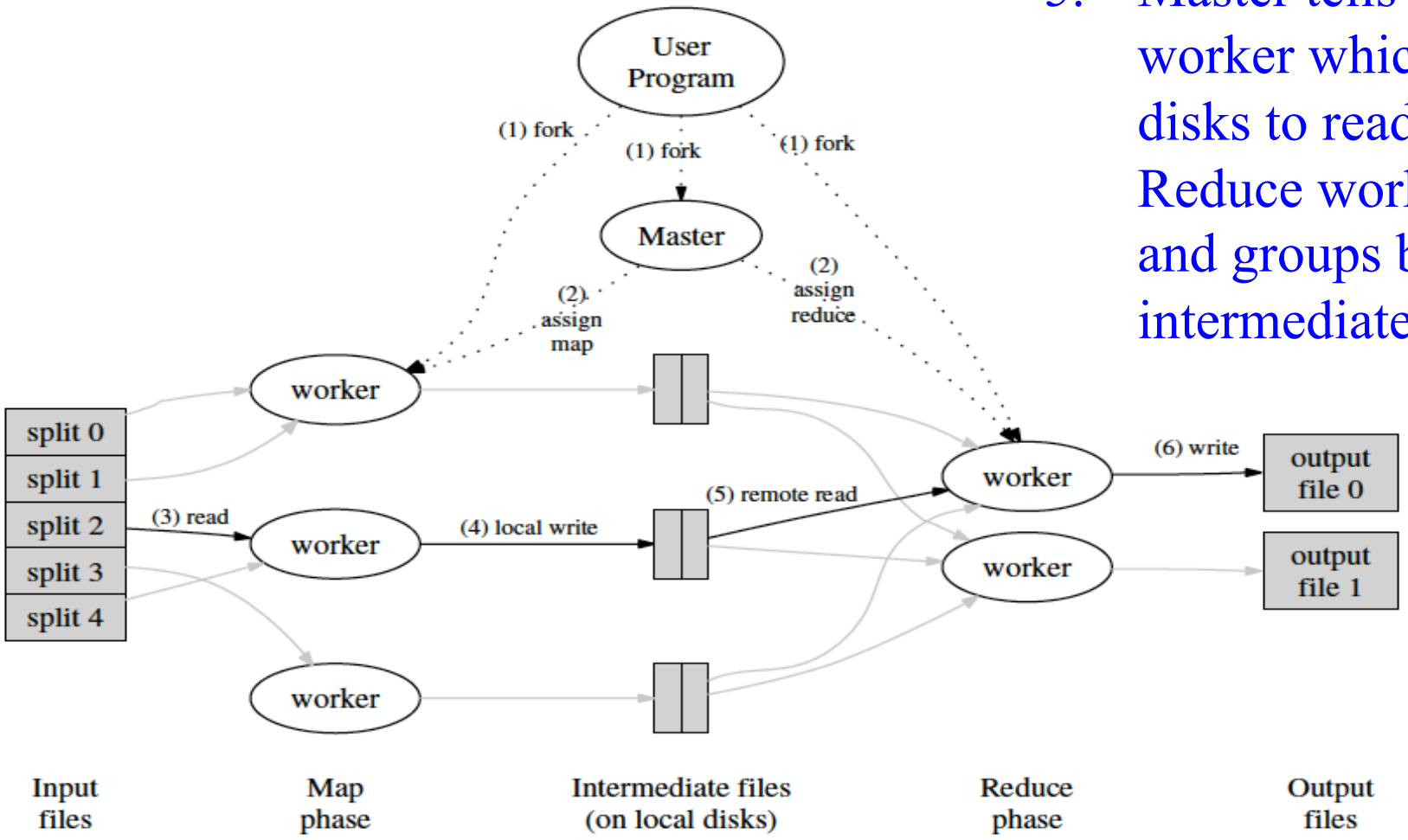
MapReduce Execution Overview

4. Write pairs to local disk, partitioned into R regions, e.g. by hash(key) mod R. inform master



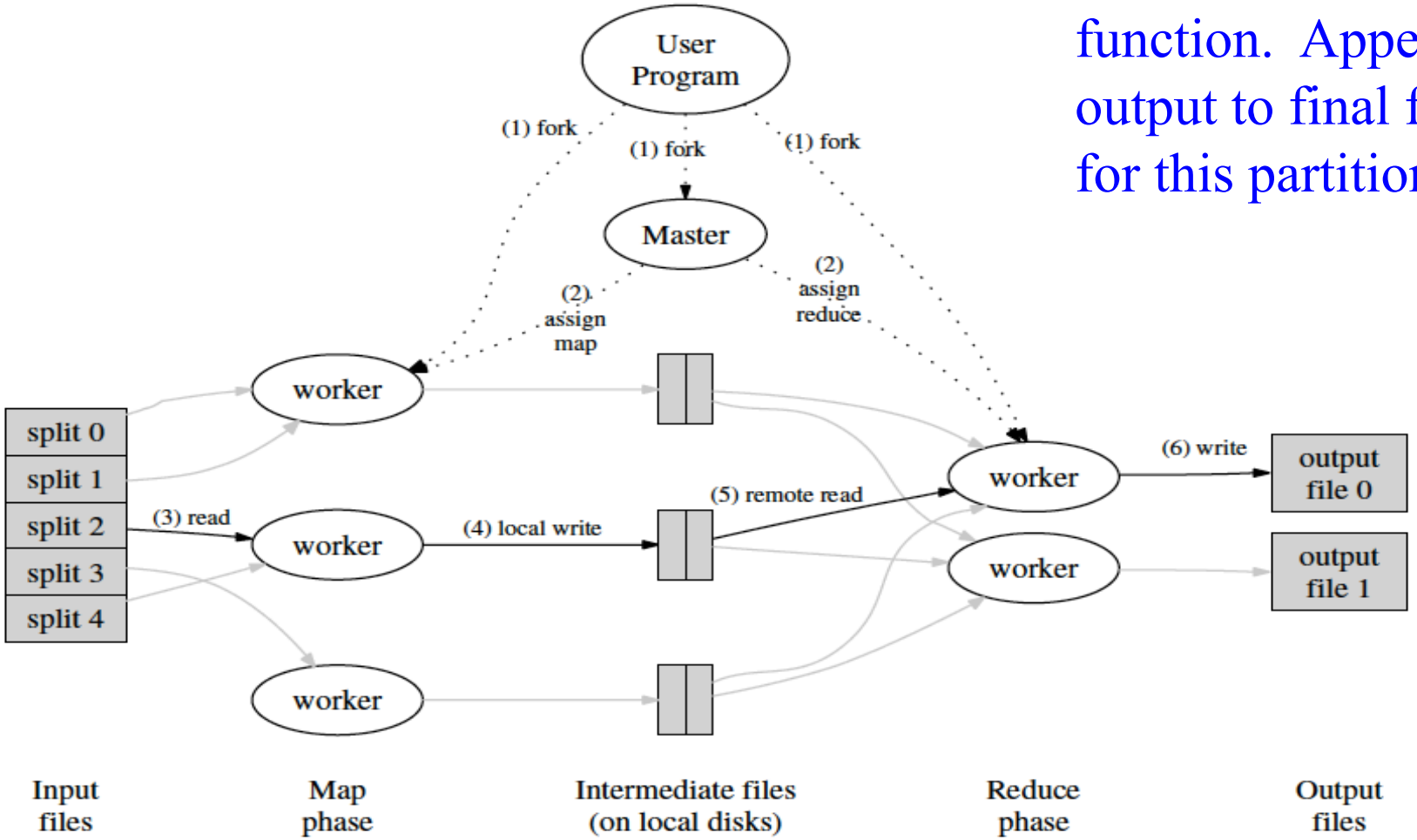
MapReduce Execution Overview

5. Master tells Reduce worker which local disks to read. Reduce worker sorts and groups by intermediate key



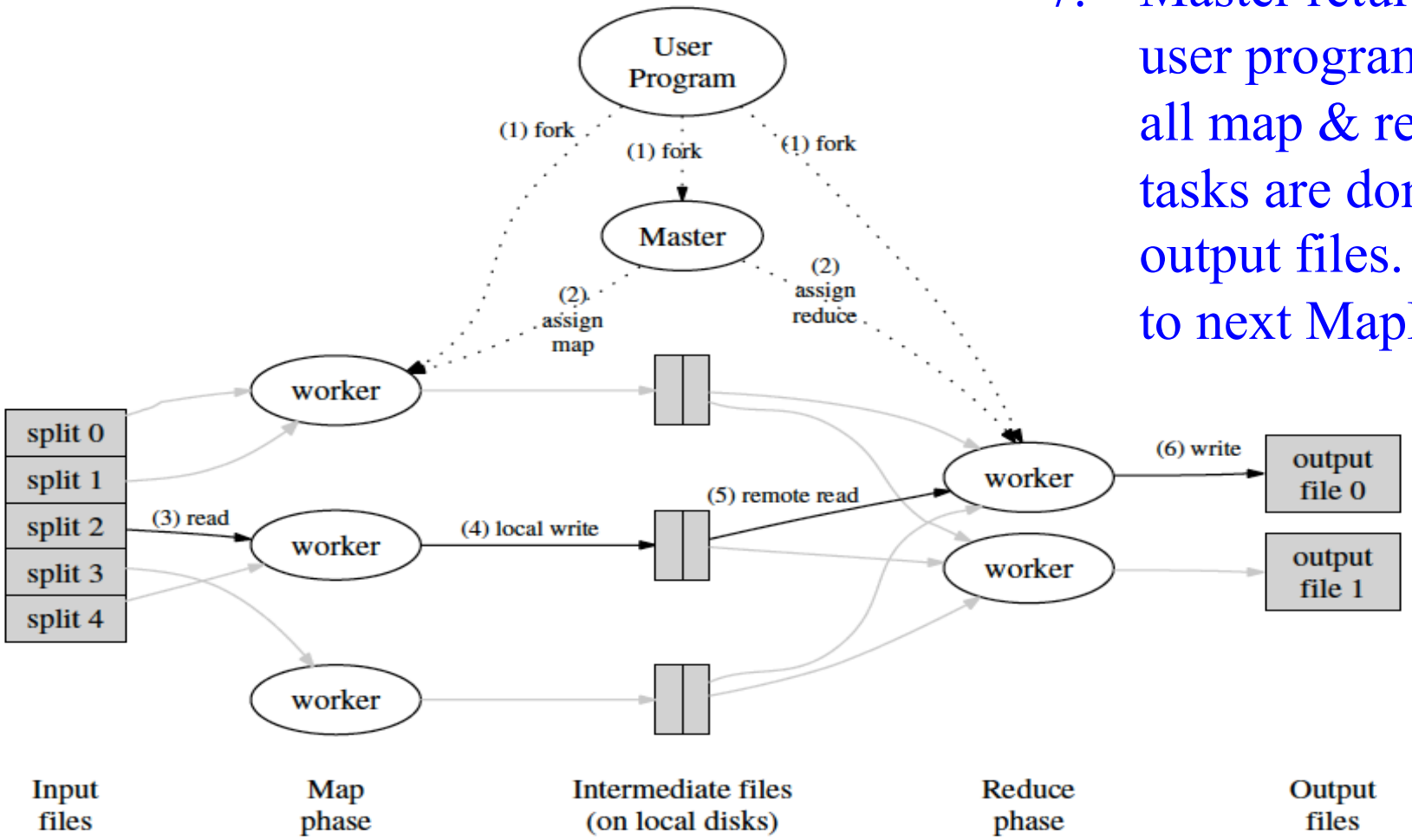
MapReduce Execution Overview

6. Execute Reduce function. Append output to final file for this partition



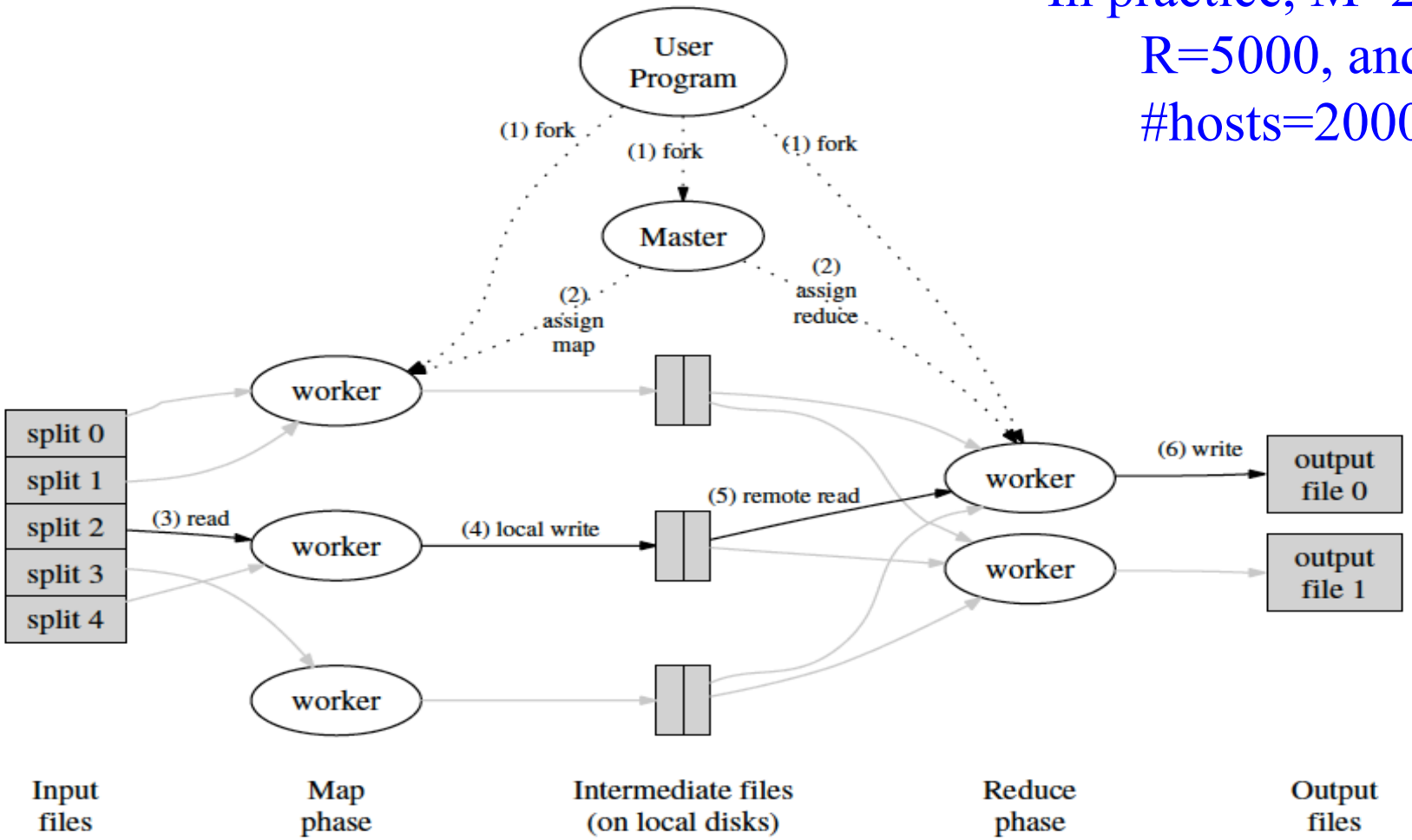
MapReduce Execution Overview

7. Master returns to user program after all map & reduce tasks are done. R output files. (input to next MapReduce)

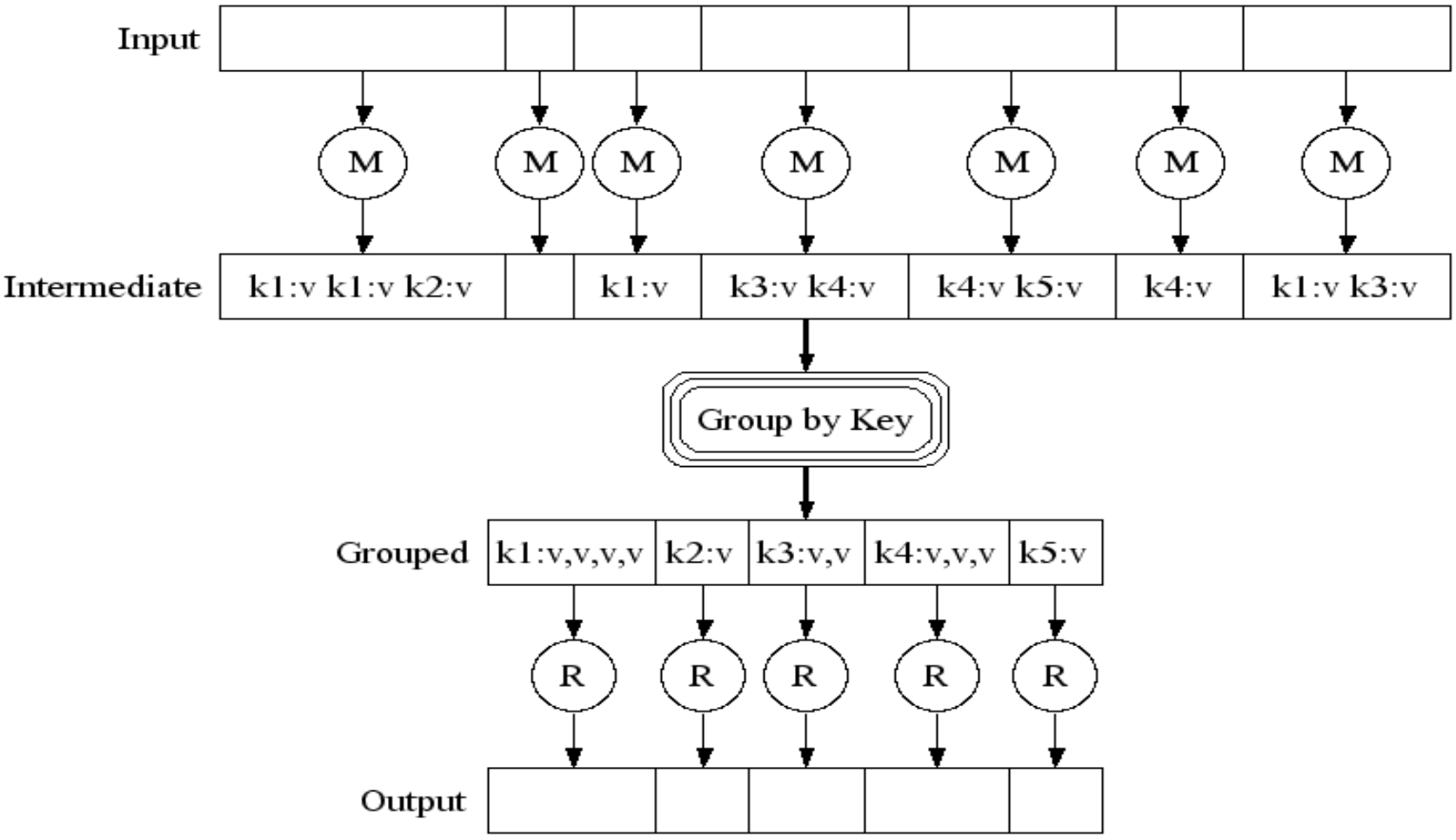


MapReduce Execution Overview

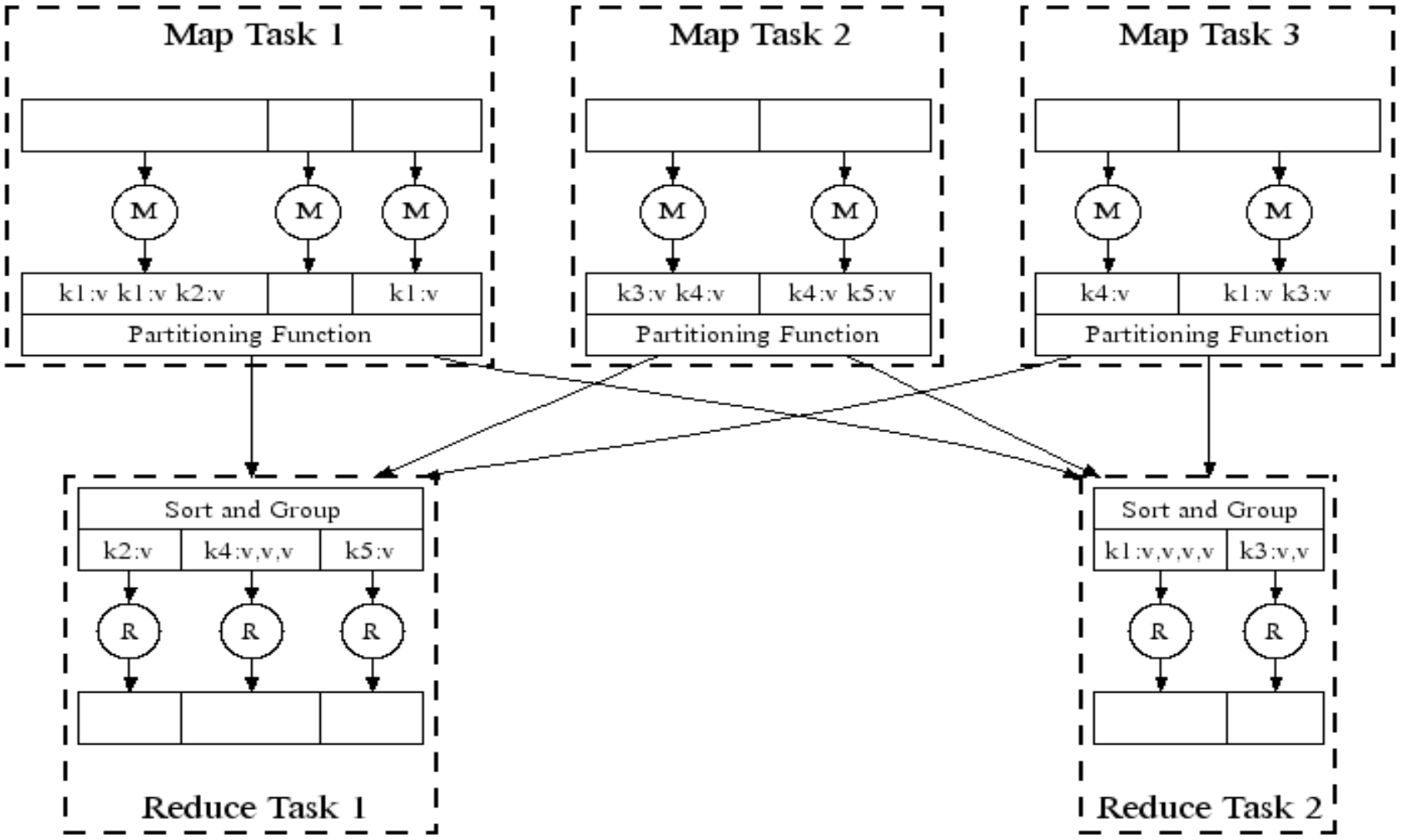
In practice, M=200,000, R=5000, and #hosts=2000



MapReduce: Example



MapReduce in Parallel: Example



MapReduce Fault Tolerance

- Master pings every worker periodically
  - If no response, the worker has failed.
  - Any Map or Reduce tasks assigned to the worker have to be rescheduled
- Highly resilient to failures
  - Entire cluster of 80 machines went down, so Master just reassigned MapReduce tasks and continued executing
- If Master fails, then user program can retry MapReduce
- Also skips bad records

MapReduce Locality

- MapReduce master schedules a map task on a machine that contains a GFS replica of the corresponding input data.
  - If not, schedules a map task near a replica (e.g., on a worker machine that is on the same network switch).
- Results in almost no network bandwidth. Most input data is read locally.

MapReduce Stragglers

- A few Map or Reduce tasks inevitably take a long time to complete = *straggler*
- When a MapReduce operation is close to completion, the master schedules *backup* executions of the remaining in-progress tasks.
  - the task is marked as completed whenever either the primary or the backup execution completes.

MapReduce Stragglers (2)

- Only increases the computational costs by a few percent.
- significantly reduces the time to complete large MapReduce operations
  - Without backup tasks, it takes a MapReduce Sort 44% longer to finish

MapReduce Adoption

- In Google:
  - Used in many projects
  - Used in Google web search. The indexing system takes as input 20 TB of GFS files from Google's web crawling. The indexing process runs as a sequence of five to ten MapReduce operations

- Amazon's EC2 also offers a MapReduce service

Strengths of MapReduce?

- Provides a general way of specifying and automatically parallelizing data computations
- Successfully scales over a large # of distributed commodity PCs
- Highly fault tolerant solution
- Add more from class discussion here…

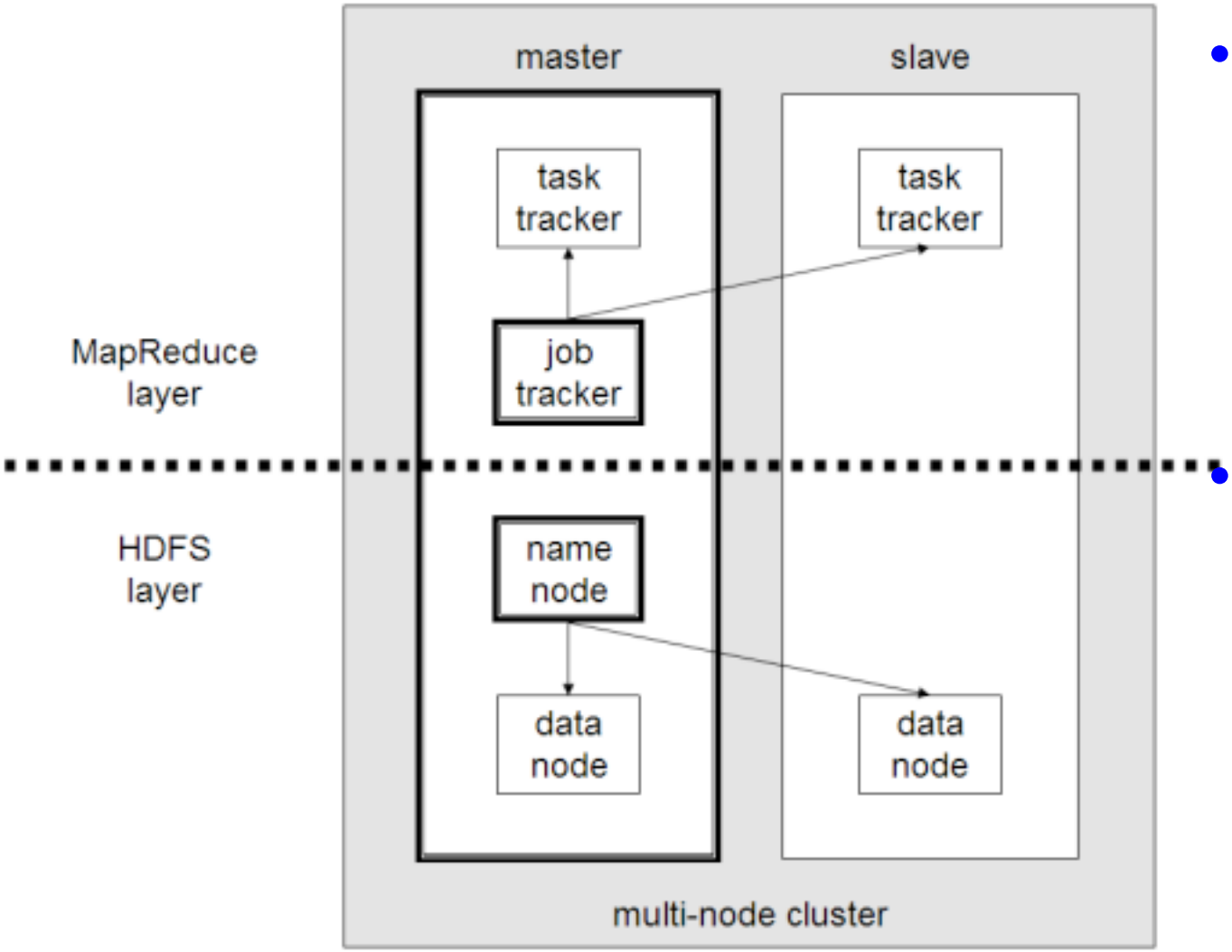
Limitations of MapReduce?

- Are there computational tasks for which MapReduce is not suitable?
- Automated way to convert code to MapReduce-compatible code?
- Add more from class discussion here…

Hadoop

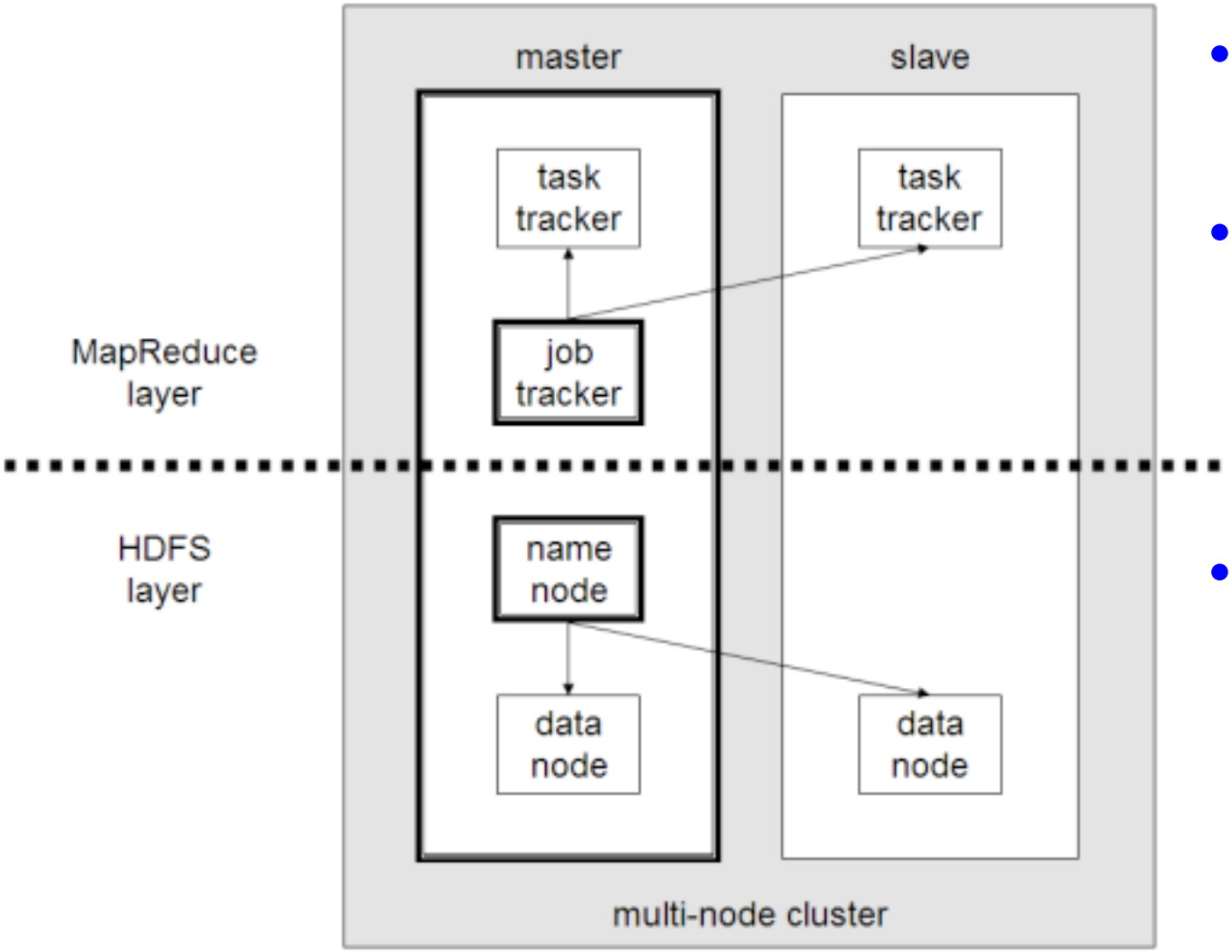
- Open-source software (Apache project) for reliable, scalable, distributed computing
  - Hadoop Common: common utilities
  - Open source implementation of MapReduce
  - Hadoop Distributed File System (HDFS)
  - See *http://hadoop.apache.org/*
  - Written in Java
- Inspired by GFS and MapReduce papers
  - Same assumptions as Google of large clusters of commodity hardware/PCs

Hadoop



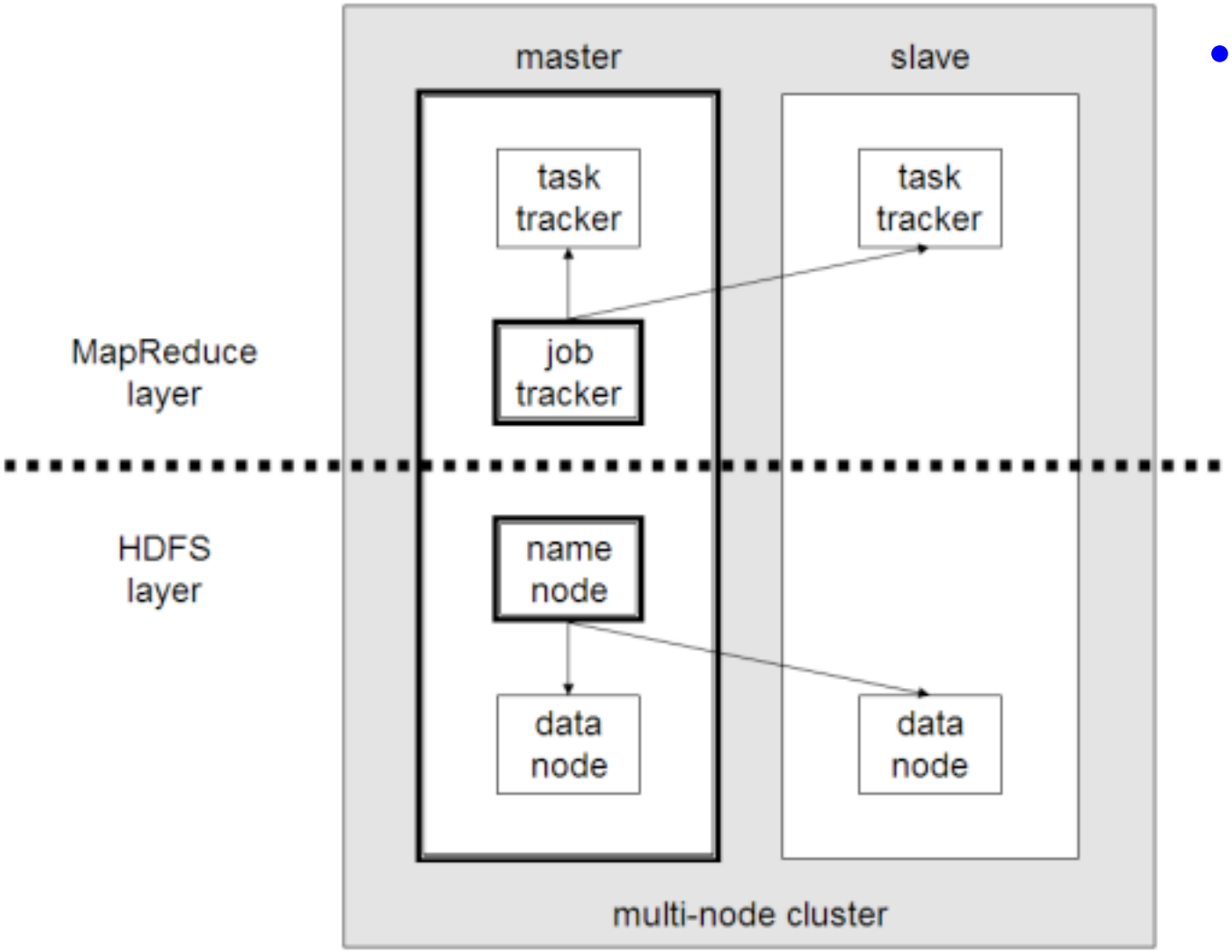
- Client apps submit MapReduce jobs to job tracker
- Job tracker evenly distributes work to task tracker nodes

Hadoop



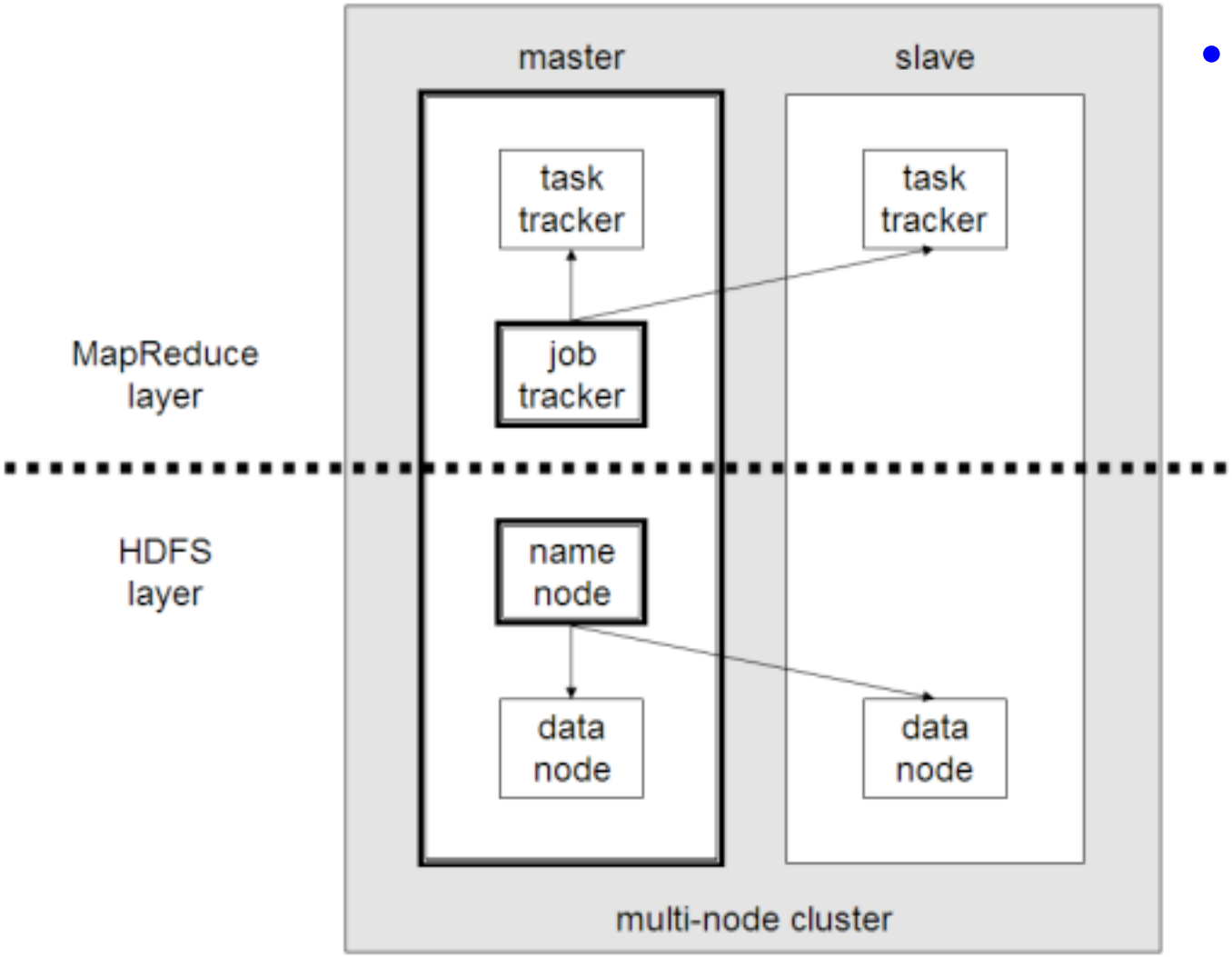
- Similar to MapReduce
- Place tasks near data (only available with HDFS)
- fault tolerance via heartbeat, checkpoint, etc.

Hadoop



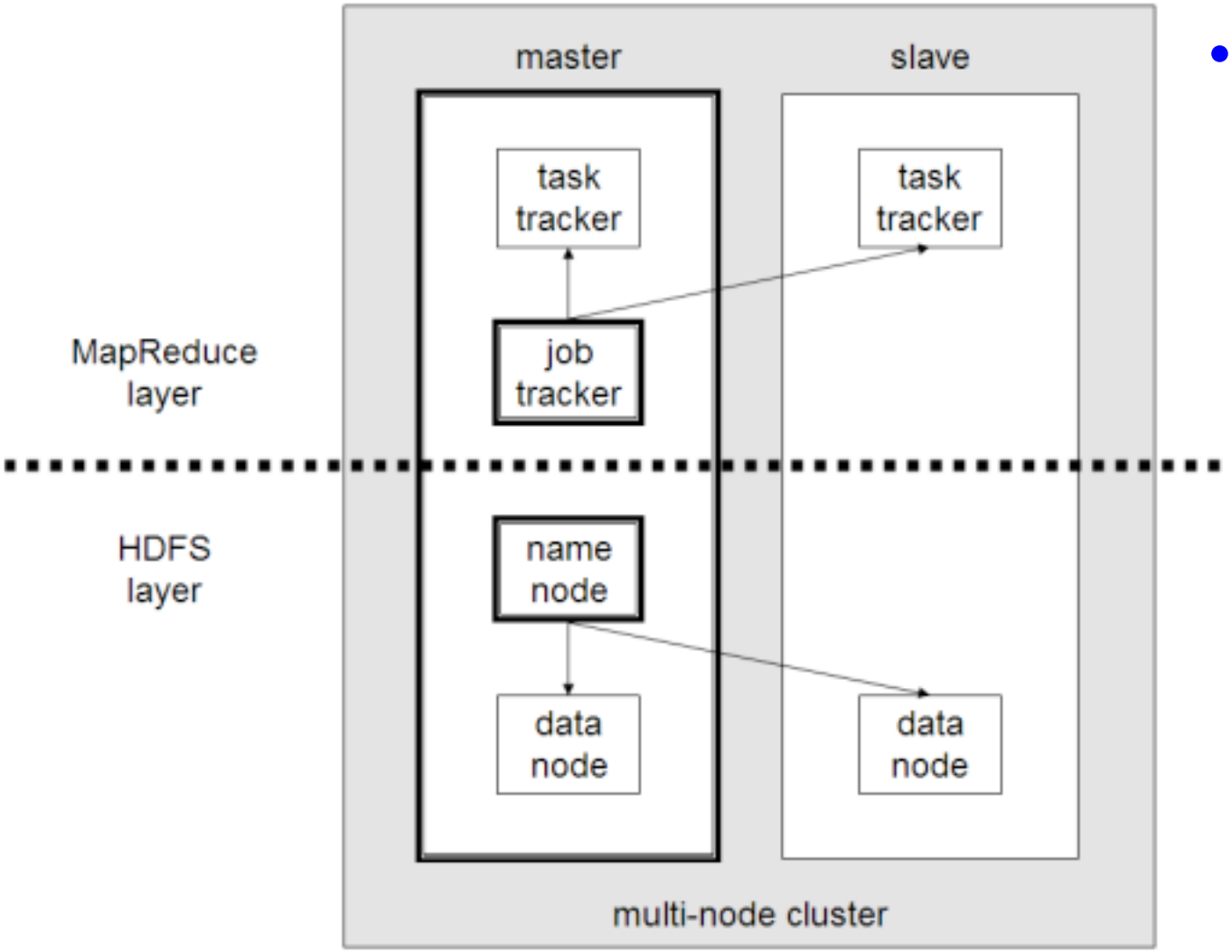
- Can have different job schedulers:
  - FIFO
  - Fair (Facebook)
  - Capacity-based (Yahoo)

Hadoop



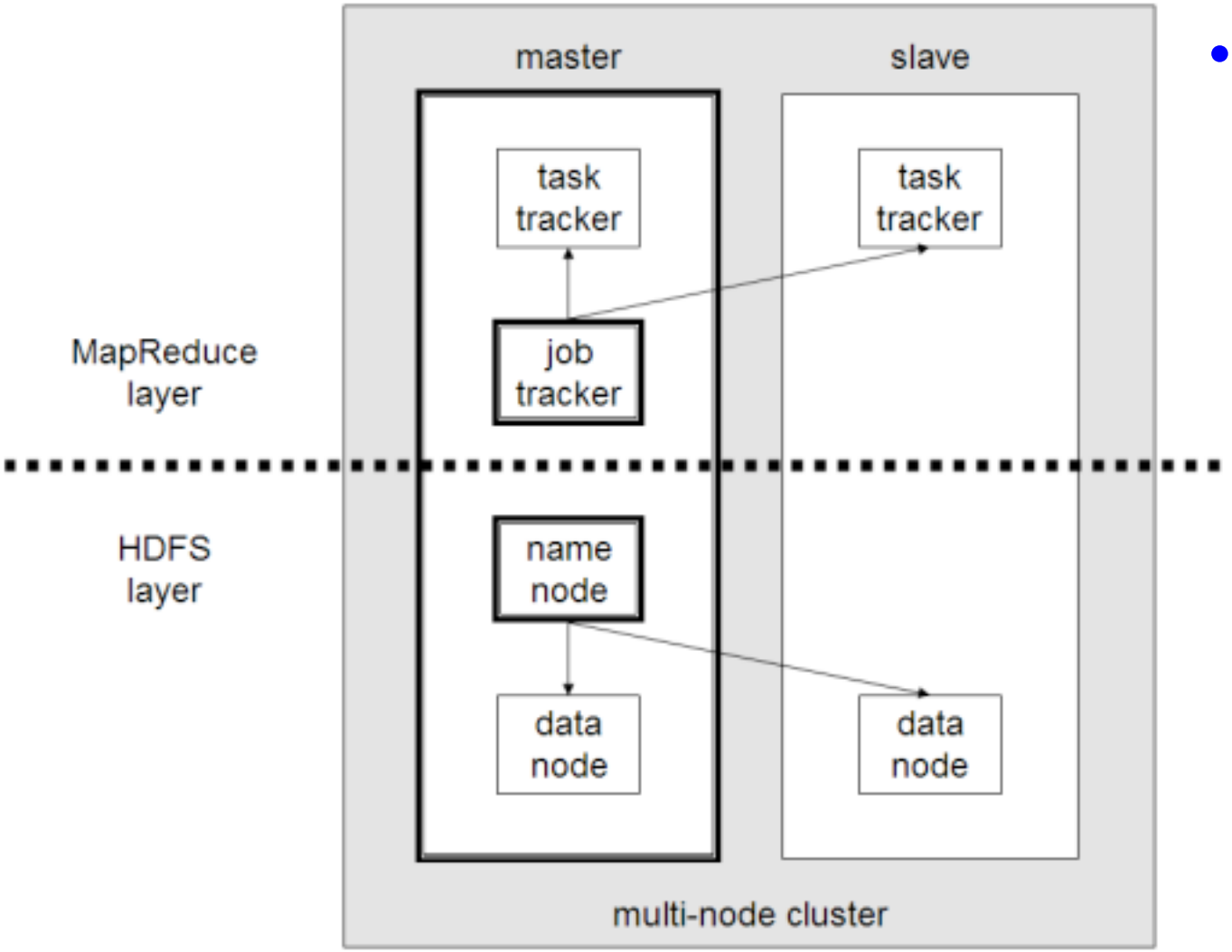
- HDFS like GFS
  - 3 replicas
  - Long files
  - Name node is ~master server
    - Checkpted
    - Replicated
  - Not fully POSIX compliant

Hadoop



- Can use MapReduce without HDFS
  - Amazon S3
  - CloudStore
  - Lose locality advantage

Hadoop



- Can use HDFS w/o MapReduce
  - e.g. Apache Mahout machine learning system

Hadoop Users

- Yahoo is a big contributor of code
  - 10000 node Hadoop cluster for web search in 2009
- Lots of other big users:
  - Facebook, Ebay, IBM, Twitter, rackspace, LinkedIn
  - New York Times used Hadoop over 100 Amazon EC2 instances to convert 4 TB of public domain articles from 1851–1922 into pdf stored in Amazon's S3, took 24 hours
  - In 2010, Facebook claimed that they had the largest Hadoop cluster in the world (21 PB)
    - July 2011 (30 PB), June 2012 (100 PB)
    - November 2012: warehouse grows roughly half a PB per day
  - More than half of the Fortune 50 use Hadoop

Hadoop Users

- Used for massive data operations:
  - Log and/or clickstream analysis,
  - data mining,
  - image processing,
  - marketing analytics, etc.
- Join HUG (Hadoop Users Groups)
  - Boulder area chapter
- In 2007, IBM, Google, & NSF form Academic Cloud Computing Initiative
  - Support Hadoop research