

Data Center Fundamentals: The Datacenter as a Computer

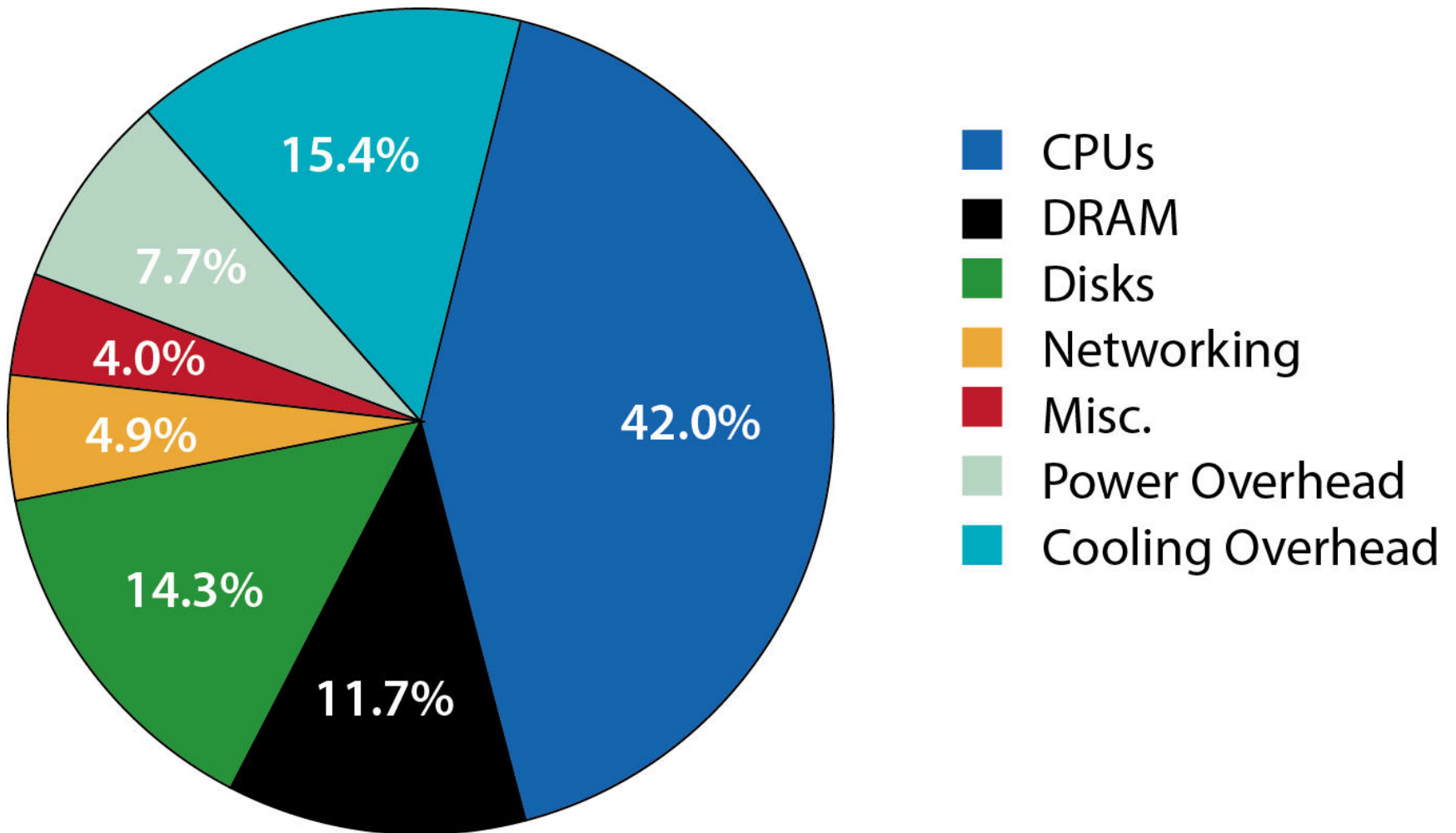
George Porter
CSE 124
February 3, 2015

*Includes material taken from Barroso et al., 2013, and UCSD 222a.

Data Center Costs

- James Hamilton published basic 2008 breakdown
- Servers: 45%
 - CPU, memory, disk
- Infrastructure: 25%
 - UPS, cooling, power distribution
- Power draw: 15%
 - Electrical utility costs
- Network: 15%
 - Switches, links, transit

Data center power usage



Data center power efficiency

- What does power efficiency mean?
- How could you measure power efficiency?
 - For your own home
 - For a single computer
 - For a data center
- Can you directly compare
 - Facebook and Google?
 - Netflix.com and Hulu.com?

Quantifying energy-efficiency: PUE

- PUE = Power Usage Effectiveness
- Simply compares
 - Power used for computing
 - Total power used

$$\text{PUE} = (\text{Facility Power}) / (\text{Computing Equipment power})$$

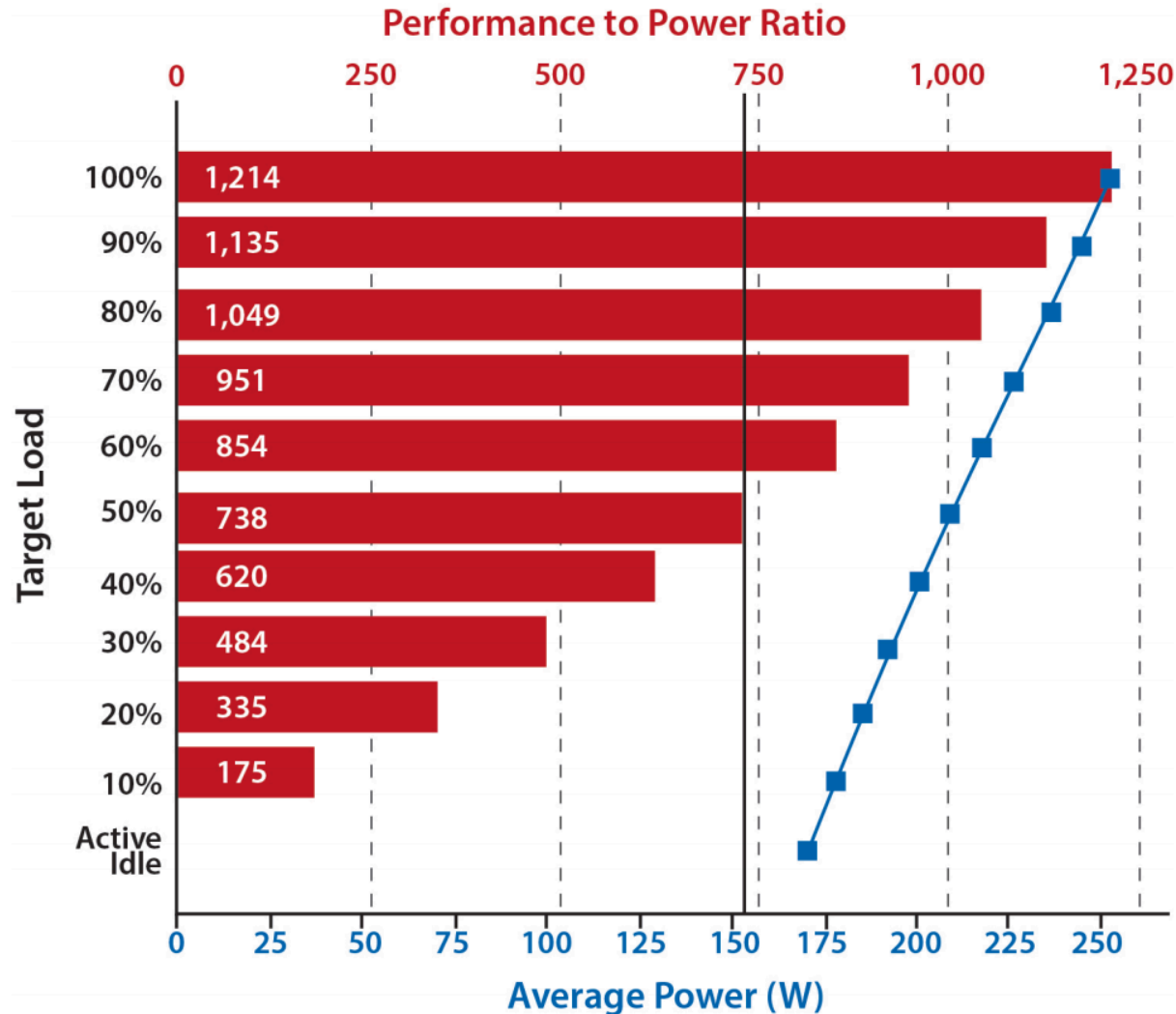
- Historically cooling was a huge source of power
 - E.g., 1 watt of computing meant 1 Watt of cooling!

Google's “Chiller-less” Data Center

- Belgium
- Most of the year it is cool enough to not need cooling
- What about on hot days?
 - Shed load to other data centers!



What about “power saving” features on modern computers?



Cluster Computing and Map/Reduce

George Porter
CSE 124
February 5, 2015

Credits

- Some material taken in part from the Yahoo Hadoop Tutorial
 - More info at: <https://developer.yahoo.com/hadoop/tutorial/index.html>
 - (Used with permission via a Creative Commons Attribution 3.0 Unported License)

Announcements

- Read the Dean and Ghemawat paper linked off the web site to learn about today's topic
 - Midterm 1 is Tuesday
 - In class
 - Closed book; you may bring a study sheet (front and back), and will turn that sheet in with your exam
 - Will cover material up to Jan 27
- No data center material, no Brewer, no MapReduce

Datacenter-scale Internet Applications



An explosion of data

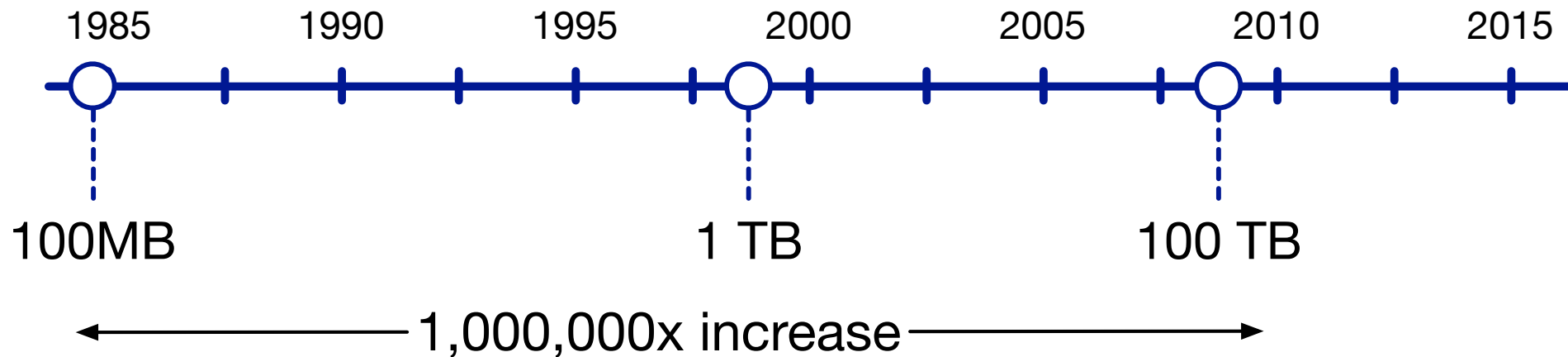
- New types of applications are hosted online
- Key feature is large data sizes
 - Application-specific: Maps, searches, posts, messages, restaurant reviews
 - Networks: friends list, contacts list, p2p apps
 - Web analytics: clickstream (ads), logs, web crawl data, trace data, ...
- Processing needs
 - On-line: searches, IMs, ad-insertion
 - Off-line/batch: compute friends network, identify slow machines, recommend movies/books, understand how users traverse your website



Size of “data-intensive” has grown



Size of “data-intensive” has grown



Map/Reduce motivation

■ Prior experience:

- Lots of “one off” programs to process data
- Each one handles data movement, machine failure, network bandwidth management, ...
- Scheduling, coordinating across the cluster really hard
- 10s of thousands of these programs!

■ Idea:

- Can we identify the common data operations and provide those as a central service?
- Solve data movement, fault tolerance, etc. once, not each time

Functional Programming Example

- LISP, Haskell, ...
- (map f list): apply f to each element of 'list'
 - (map square [1 2 3 4]) $\rightarrow$ (1 4 9 16)
- (reduce g list): apply g pairwise with elements of 'list'
 - (reduce + [1 4 9 16]) $\rightarrow$ 30
- No side effects
- Order of execution doesn't matter
- Question:
 - How would we parallelize map?
 - How would we parallelize reduce?

map() examples

map()

■ “+1”

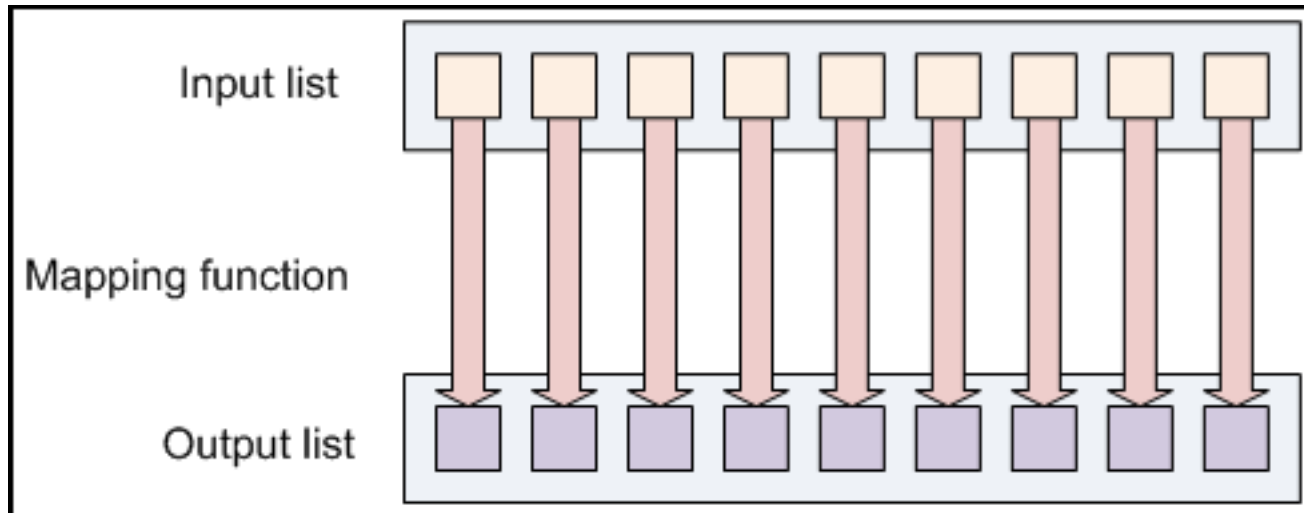
- (2, 4, 5, 8) → (3, 5, 6, 9)

■ iseven()

- (2, 4, 5, 8) → (true, true, false, true)

■ tokenize()

- (“The quick brown fox”) → (“The”, “quick”, “brown”, “fox”)



reduce() Examples

reduce()

■ “+”

- (2, 4, 5, 8) → 19

■ countTrue()

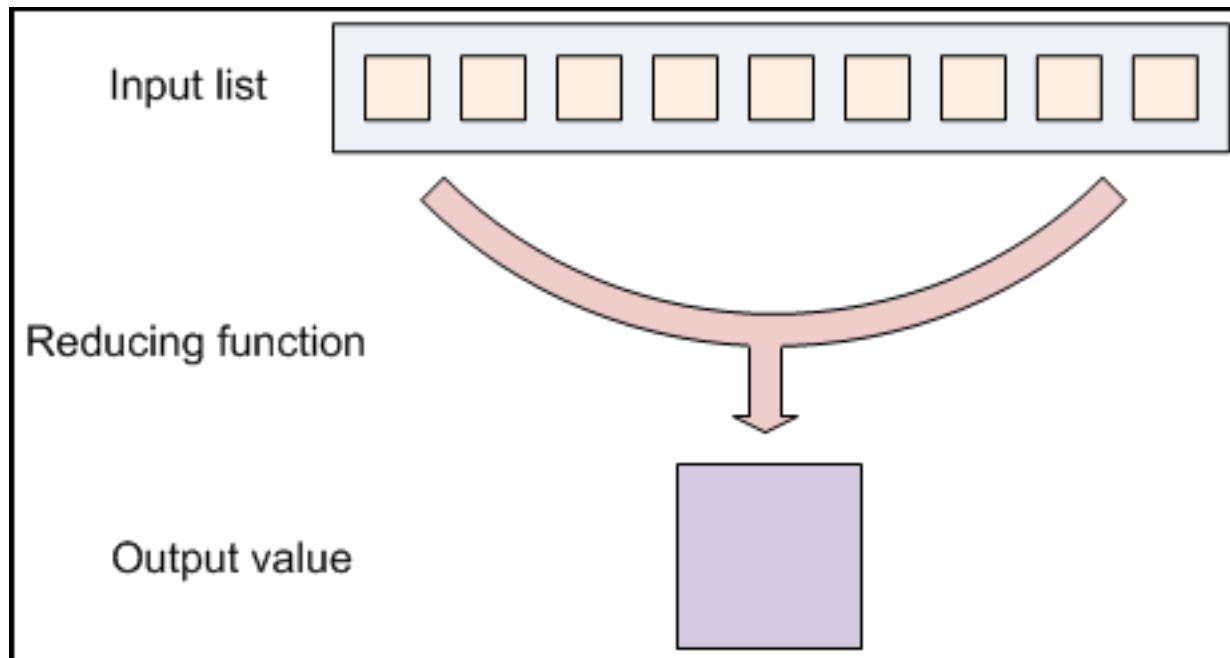
- (true, true, false, true) → 3

■ max()

- (2, 4, 5, 8) → 8

■ min()

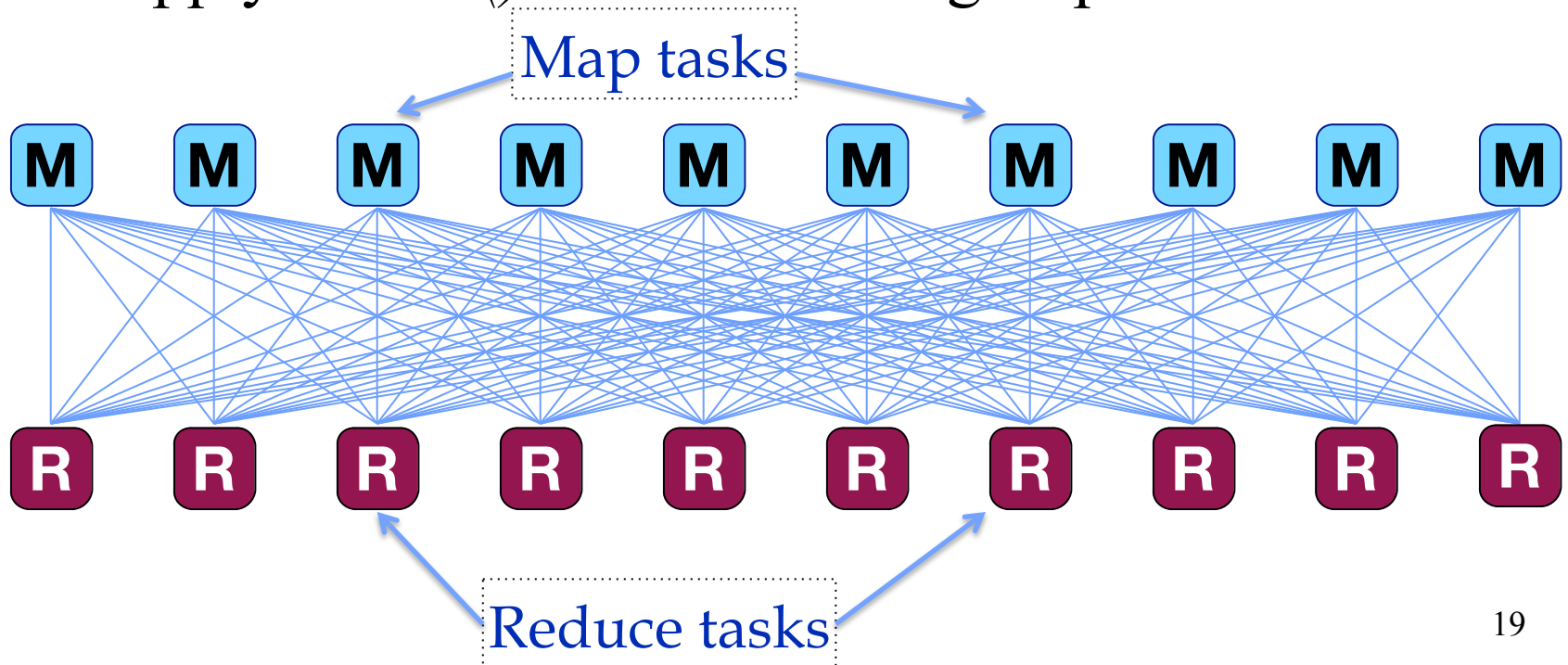
- (2, 4, 5, 8) → 2



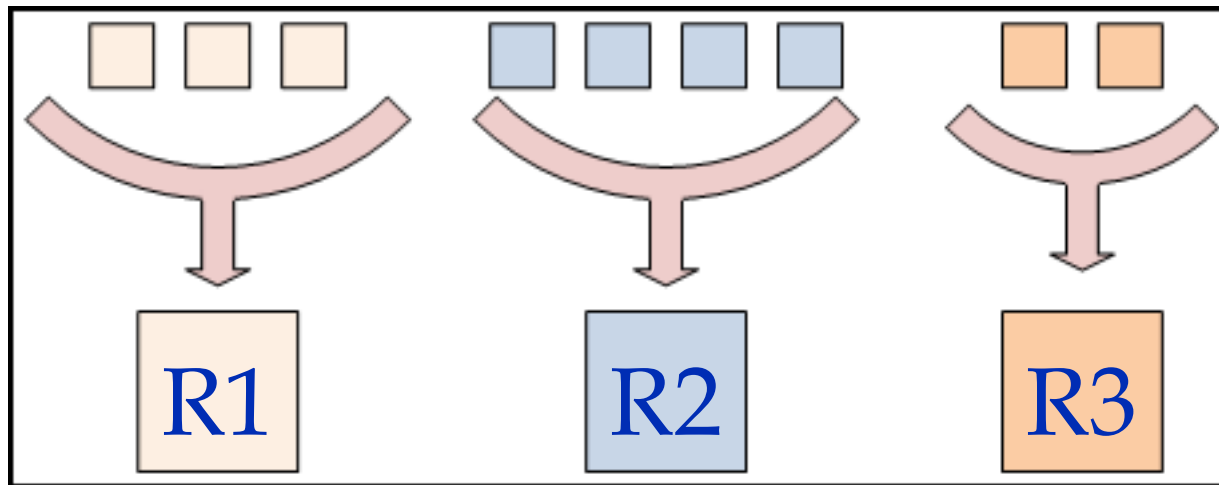
MapReduce Programming Model

Input: Set<key-value pairs>

1. Apply *map()* to each pair
2. Group by key; sort each group
3. Apply *reduce()* to each sorted group



Map/reduce



- Each color represents a separate key.
- All values with same key processed by the same reduce function

Map/Reduce example: wordcount

- $s = \text{“UCSD Computer Science is a top Computer Science program”}$
- wordcount(s) desired result:
 - UCSD: 1
 - Computer: 2
 - Science: 2
 - is: 1
 - a: 1
 - top: 1
 - program: 1

Map/Reduce example: wordcount map

- $s = \text{"UCSD Computer Science is a top Computer Science program"}$
- $\text{map}(s, \text{const}(1))$:
- result: [<"UCSD", 1>, <"Computer", 1>, <"Science", 1>, <"is", 1>, <"a", 1>, <"top", 1>, <"Computer", 1>, <"Science", 1>, <"program", 1>]
- Note repeated values
 - Why?

Map/Reduce example: wordcount shuffle/sort

- Hadoop runtime will gather together key-value pairs with the same key and append the values into a single list:
- input: [<“UCSD”, 1>, <“Computer”, 1>, <“Science”, 1>, <“is”, 1>, <“a”, 1>, <“top”, 1>, <“Computer”, 1>, <“Science”, 1>, <“program”, 1>]
- output: [<“UCSD”, [1]>, <“**Computer**”, [1, 1]>, <“**Science**”, [1,1]>, <“is”, [1]>, <“a”, [1]>, <“top”, [1]>, <“program”, [1]>]

Map/Reduce example: wordcount reduce

- Reduce function (i.e., +) applied to each key (and value list)
- input: [<“UCSD”, [1]>, <“**Computer**”, [1, 1]>, <“**Science**”, [1,1]>, <“is”, [1]>, <“a”, [1]>, <“top”, [1]>, <“program”, [1]>]
- output:
 - UCSD: 1
 - Computer: 2
 - Science: 2
 - is: 1
 - a: 1
 - top: 1
 - program: 1



- Open-source software project
 - Implements Map/Reduce and the Google File System
 - Originally led by Yahoo
 - Where did the name 'Hadoop' come from?
- Based on two Google papers:
 - The Google File System, Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, SOSP, 2003.
 - MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean and Sanjay Ghemawat, OSDI, 2004.

What is Hadoop?

- ...a programming model
 - Way of specifying data processing operations
 - Modeled after functional languages like Haskell or LISP
- ...a distributed system implementing that model
 - Software running in the data center
 - Responsible for executing your program

Despite underlying failures

Handles jobs from many users

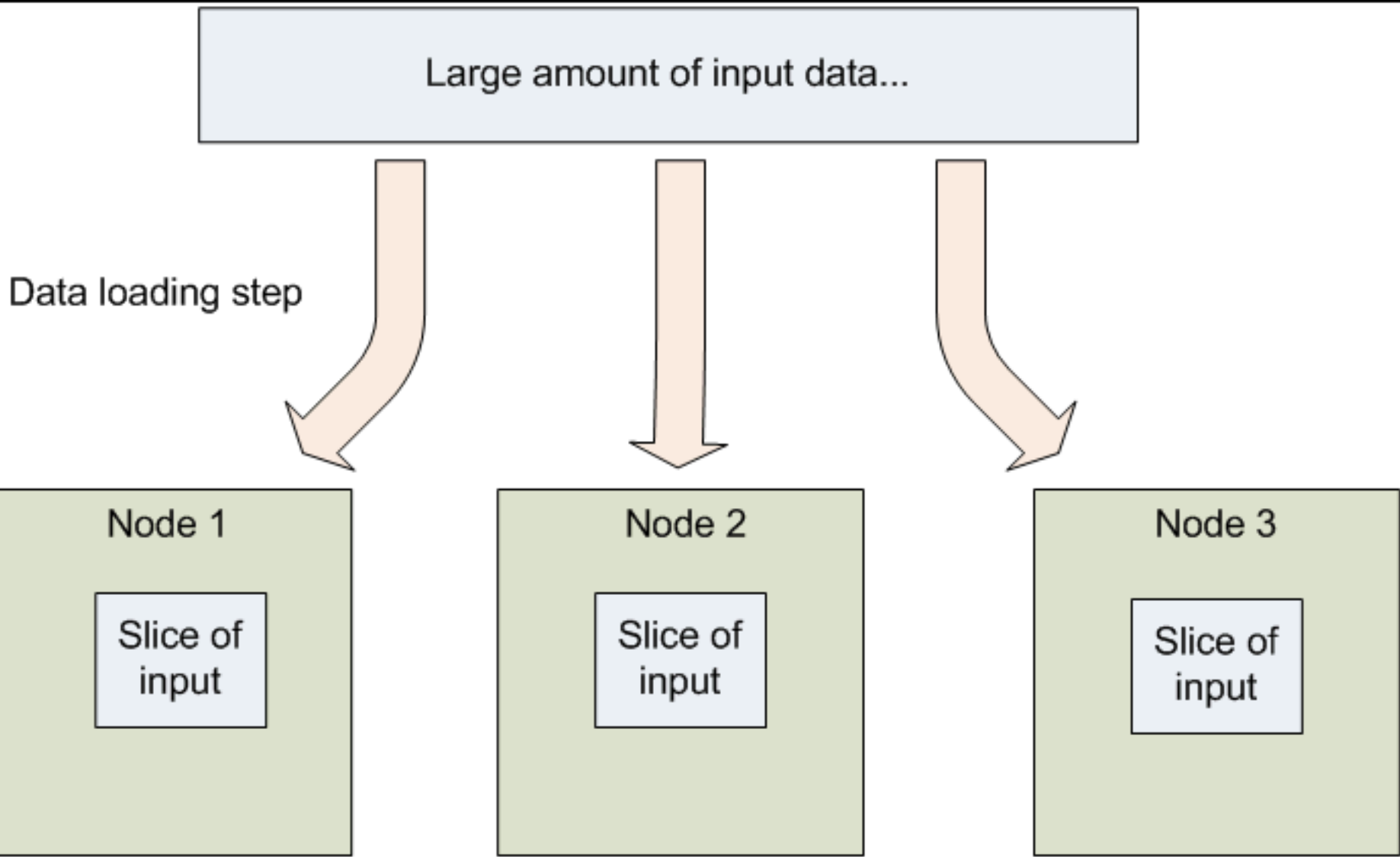
Data center realities

- Computing
 - 2-CPU, 10s of GB of memory, 4 to 8 TB drives
- Networking
 - 2 or 3 level hierarchy
 - 1 or 10 Gb/s from server to top-of-rack switch
- Keep the server cost low
- Reduce the movement of data
 - Move computation to the data
 - (Cross-data center bandwidth can be quite limited)
- Failures are the default case
 - They're really common!

Execution model

- Distribute processing of data across 100-1000s of machines
- Reuse the infrastructure by providing different implementations of `map()` and `reduce()`
- Data oriented—push computation near the data
- Aligned with the realities of data centers
- *Easy to parallelize*

Loading data into HDFS



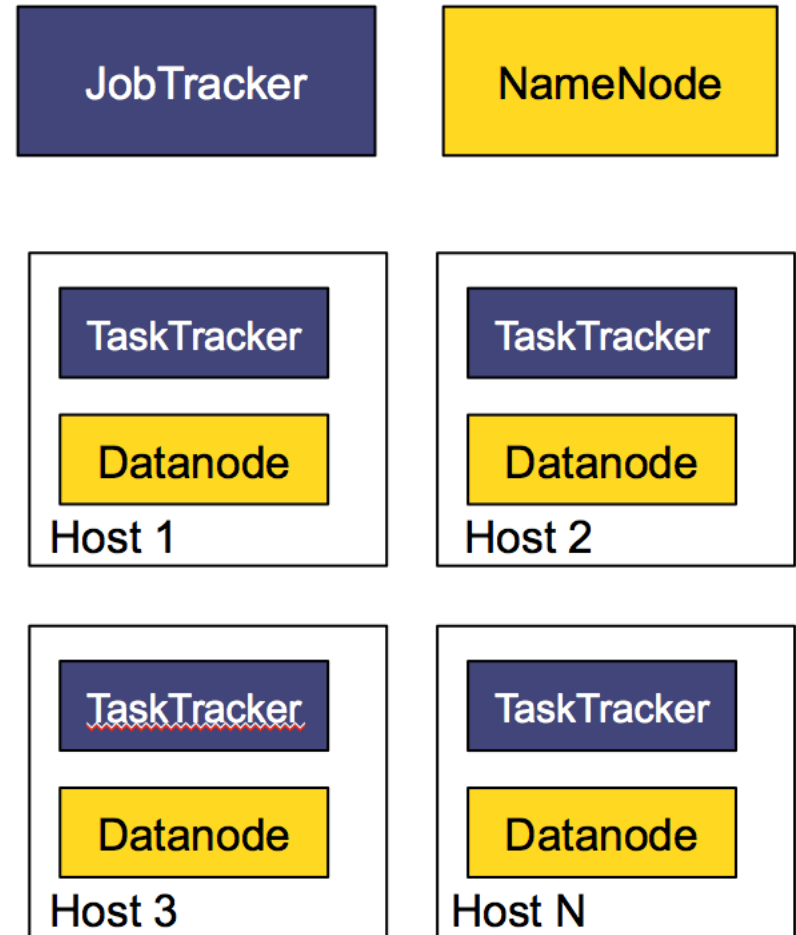
Hadoop Architecture

■ HDFS

- NameNode holds metadata
- Files broken up and spread across DataNodes

■ Map/Reduce

- JobTracker schedules and manages jobs
- TaskTracker executes individual map() and reduce() tasks



Responsibility for metadata and data

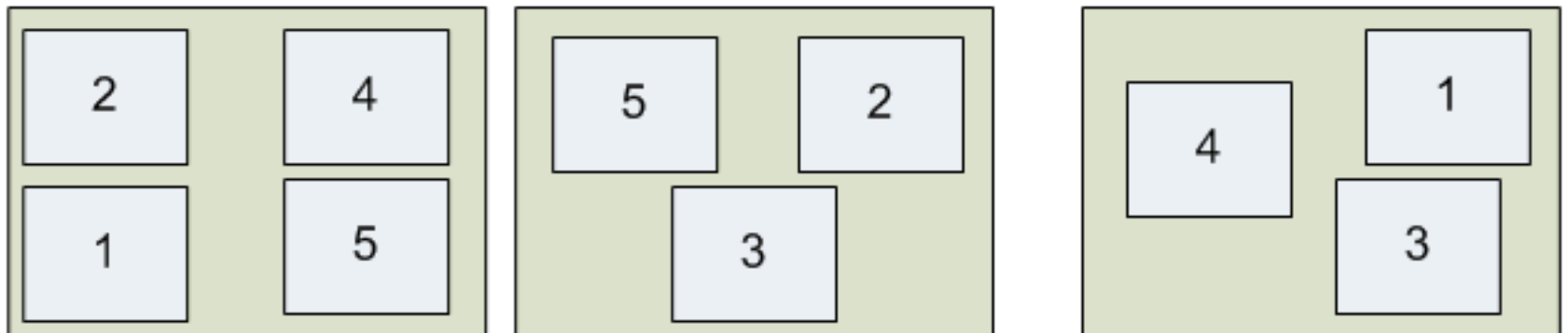
NameNode:
Stores metadata only

METADATA:

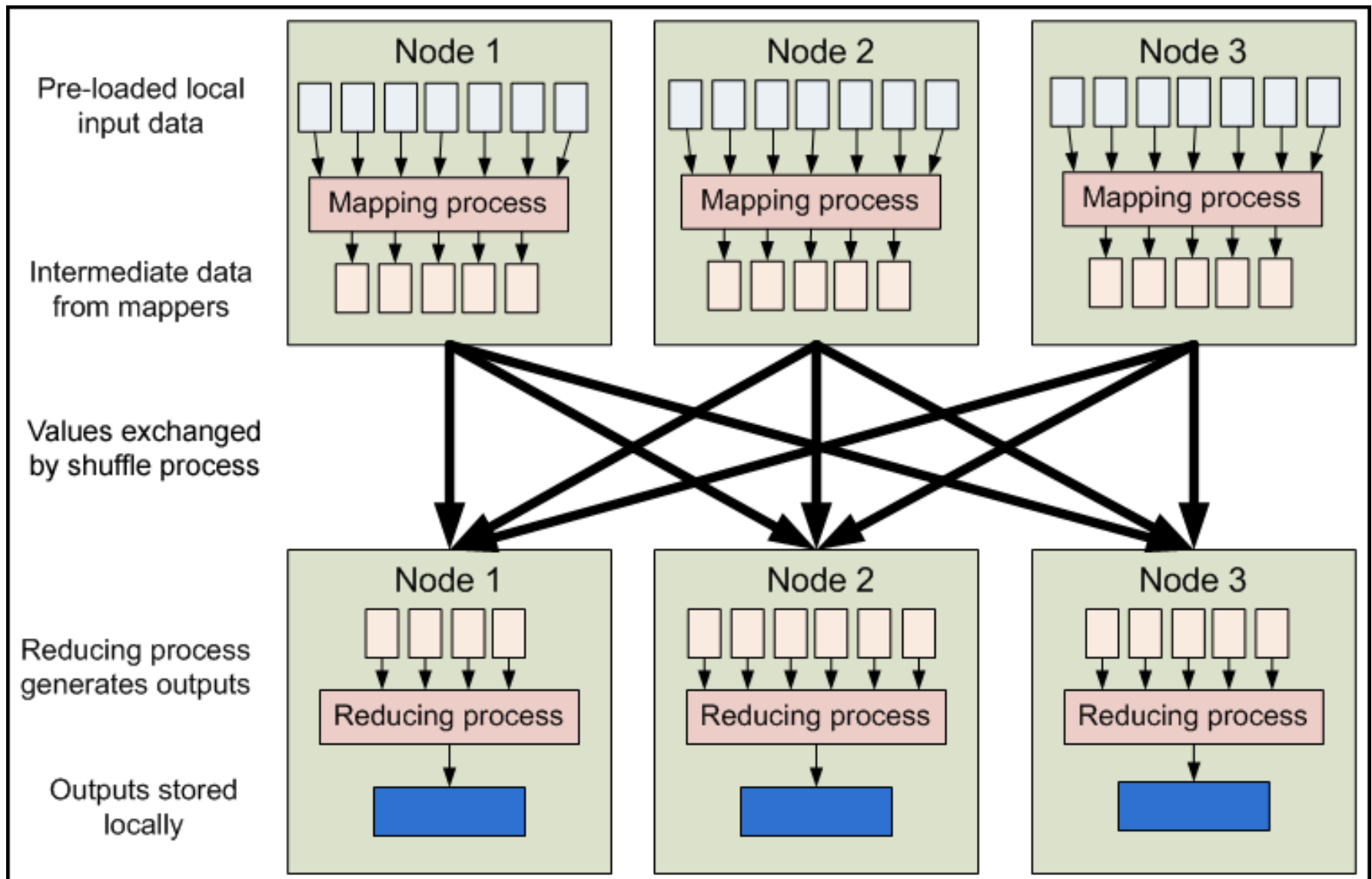
/user/aaron/foo → 1, 2, 4

/user/aaron/bar → 3, 5

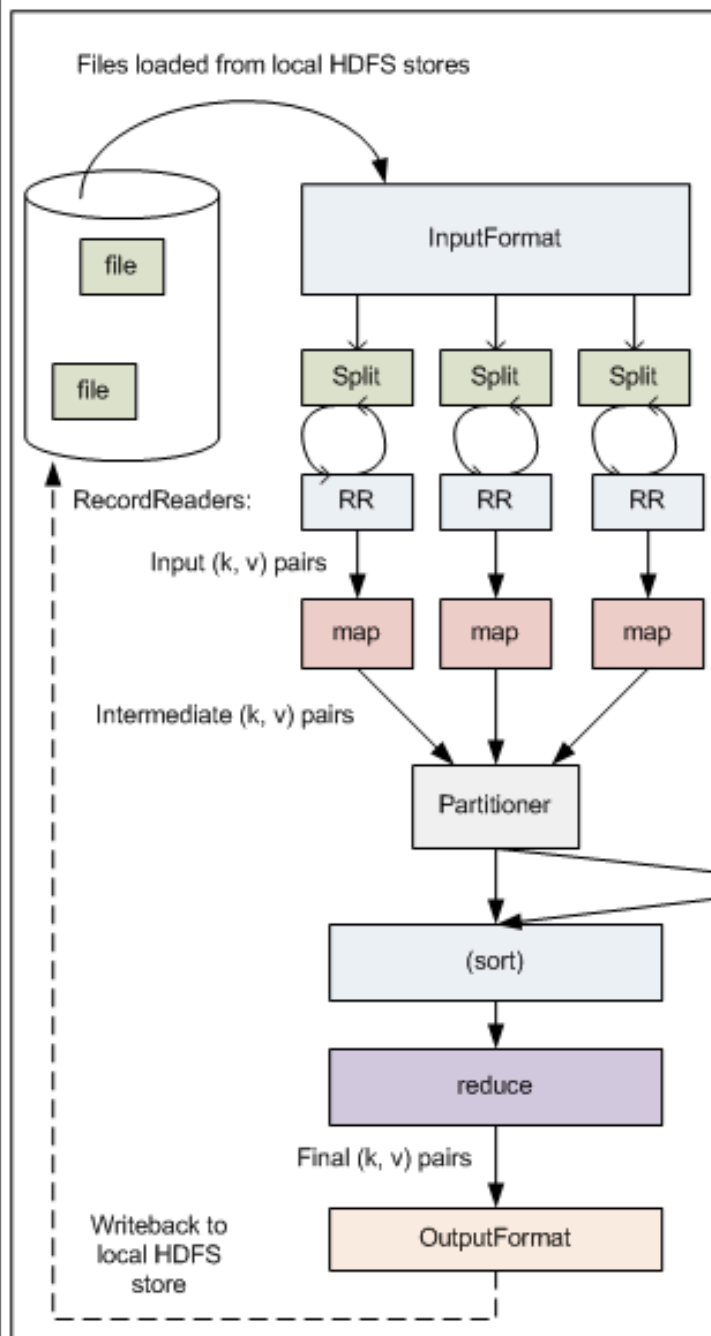
DataNodes: Store blocks from files



Map/Reduce runtime



Node 1



Node 2

