Big Data and Apache Hadoop's MapReduce

Michael Hahsler

Computer Science and Engineering
Southern Methodist University

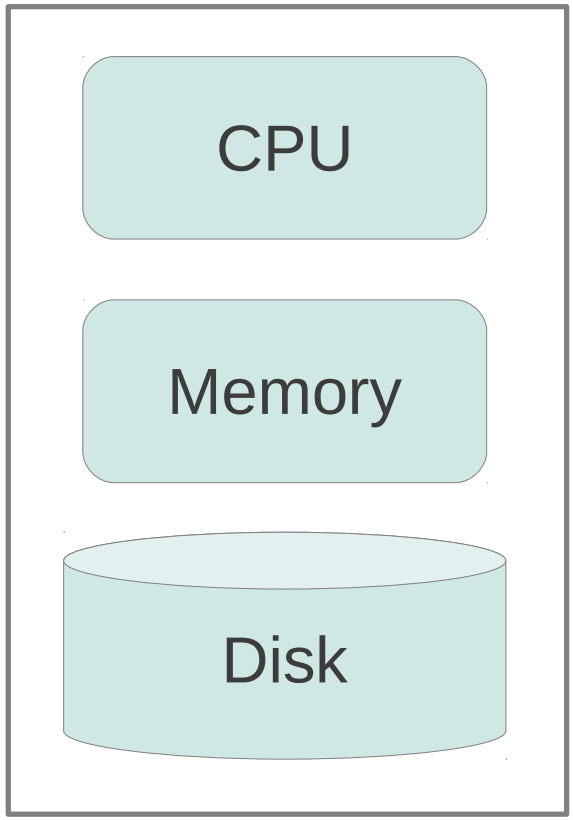
January 23, 2012

SMU | Bobby B. Lyle School of Engineering

Table of Contents

1. Introduction
2. Hadoop Distributed File System
3. MapReduce
4. More Examples

Single Machine



Typical set up for data processing/mining!
What are the problems with big data?

Big Data Challenge

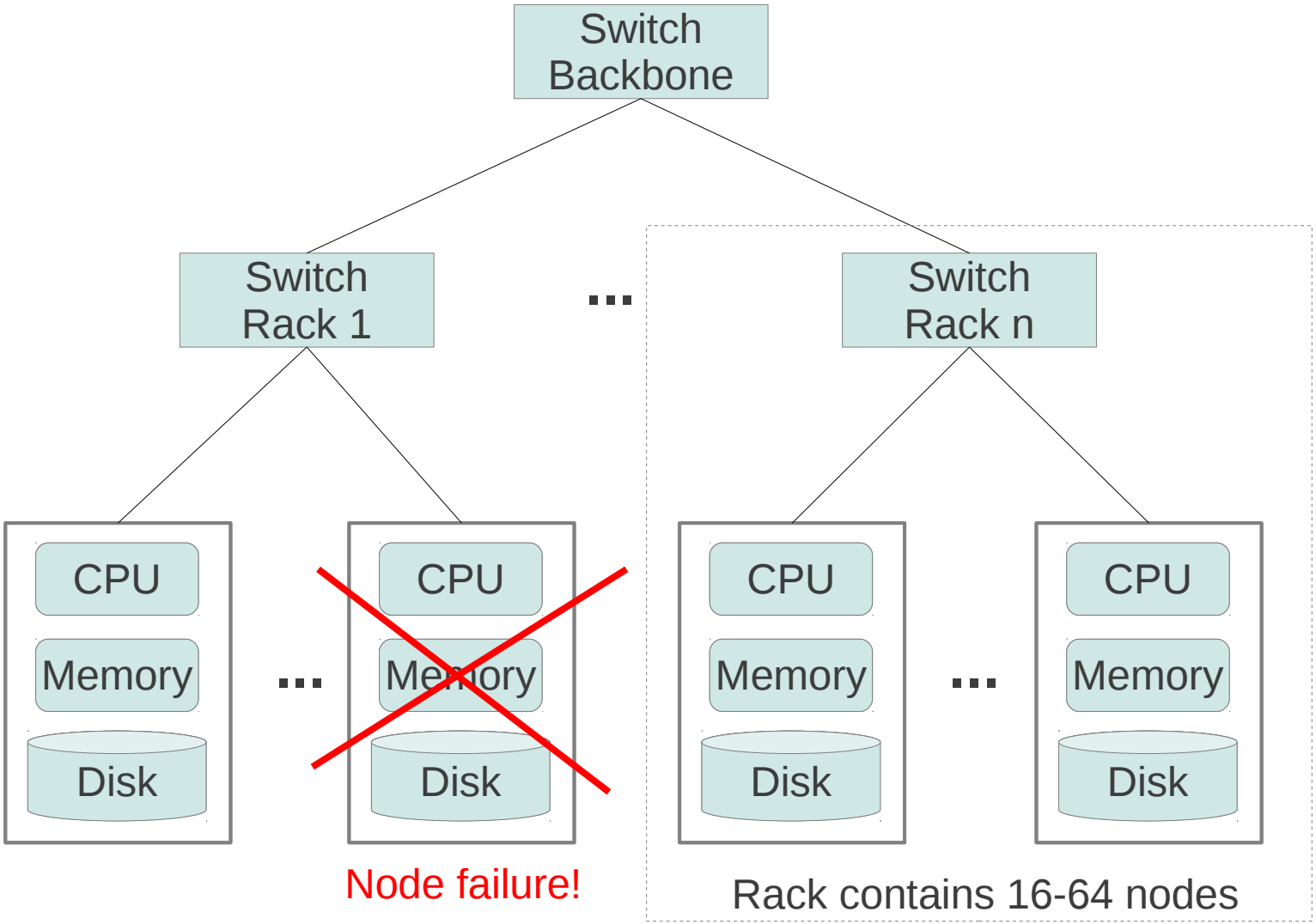
Data Sources

- Internet and social networks
- Sensors and science

Needed Infrastructure

- Scale to thousands of CPUs
- Run on cheap commodity hardware (fault-tolerant hardware is expensive!)
- Automatically handle data replication and node failure
- Data distribution and load balancing
- Easy to implement solutions (thinking in terms of parallel computing is hard!)

Hardware: Cluster Architecture



Software: Apache Hadoop

What is Apache Hadoop?

A software framework that supports data-intensive (petabytes) distributed applications under a free license.

...inspired by Google's MapReduce and Google File System (GFS).

Hadoop provides:

- Distributed file system HDFS
- API to work with MapReduce
- Job configuration and scheduling
- Track progress and utilization

Written in the Java programming language.

What Problem can be solved with Hadoop?

Characteristics

- Processing can easily be made in parallel (simple computations)
- Process large amounts of unstructured data
- Running batch jobs is acceptable

Examples

- Creating statistics (word counting)
- Searching (distributed grep)
- Sorting
- Indexing (postings list)
- Document clustering
- Graph algorithms (E.g. pagerank)

Who uses Hadoop?

- Adobe
- Amazon.com
- AOL
- Facebook
- Google
- IBM
- Microsoft
- NY Times
- Yahoo!
- ...

Source `http://wiki.apache.org/hadoop/PoweredBy`

Table of Contents

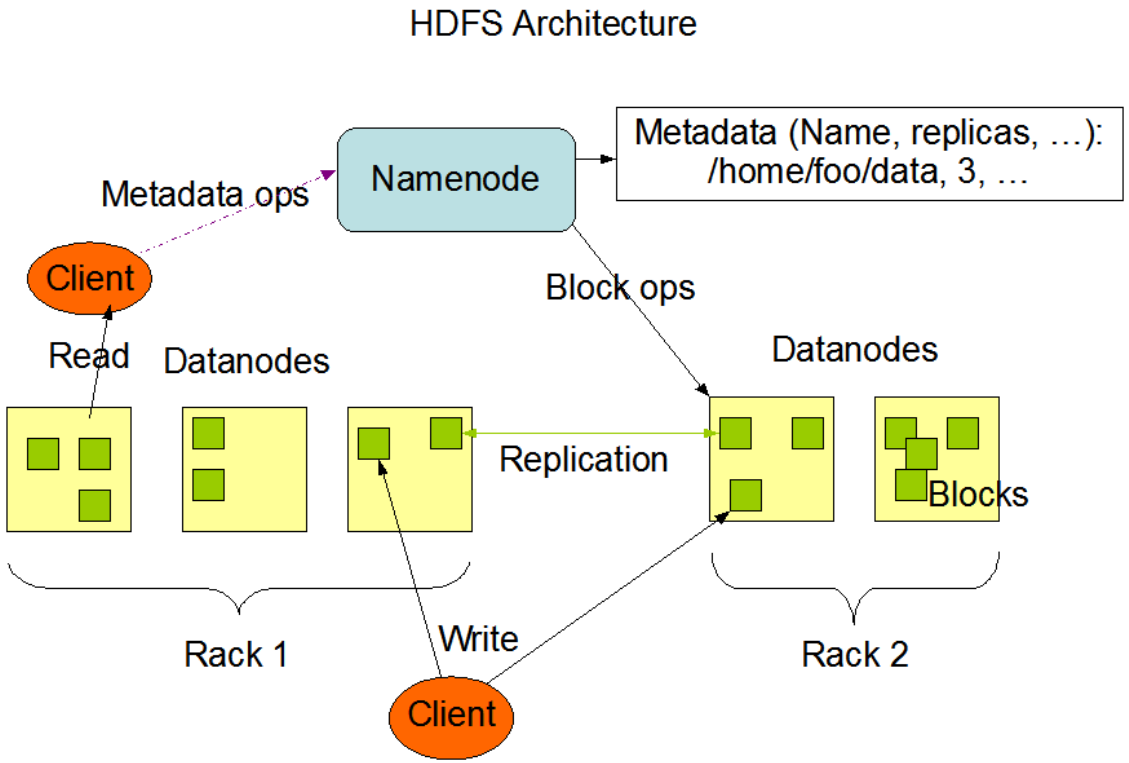
1 Introduction

2 Hadoop Distributed File System

3 MapReduce

4 More Examples

The Hadoop Distributed File System (HDFS)



Source: `http://hadoop.apache.org/common/docs/current/hdfs_design.html`

- Files are split into large block size: 64 MB (typical fs has 4 kB)
- Replication (2-3x on different racks) $\rightarrow$ Fault-tolerance
- Master node stores meta information

Table of Contents

1 Introduction

2 Hadoop Distributed File System

3 MapReduce

4 More Examples

MapReduce

Basic Idea

1. Apply a *map* function to each input element and emit key/value pairs.

$$map(k1, v1) \rightarrow list(k2, v2)$$

2. Summarize the results for each key using a *reduce* function.

$$reduce(k2, list(v2)) \rightarrow (k2, v3)$$

The user only has to specify the map and reduce functions and the framework takes care of the rest!

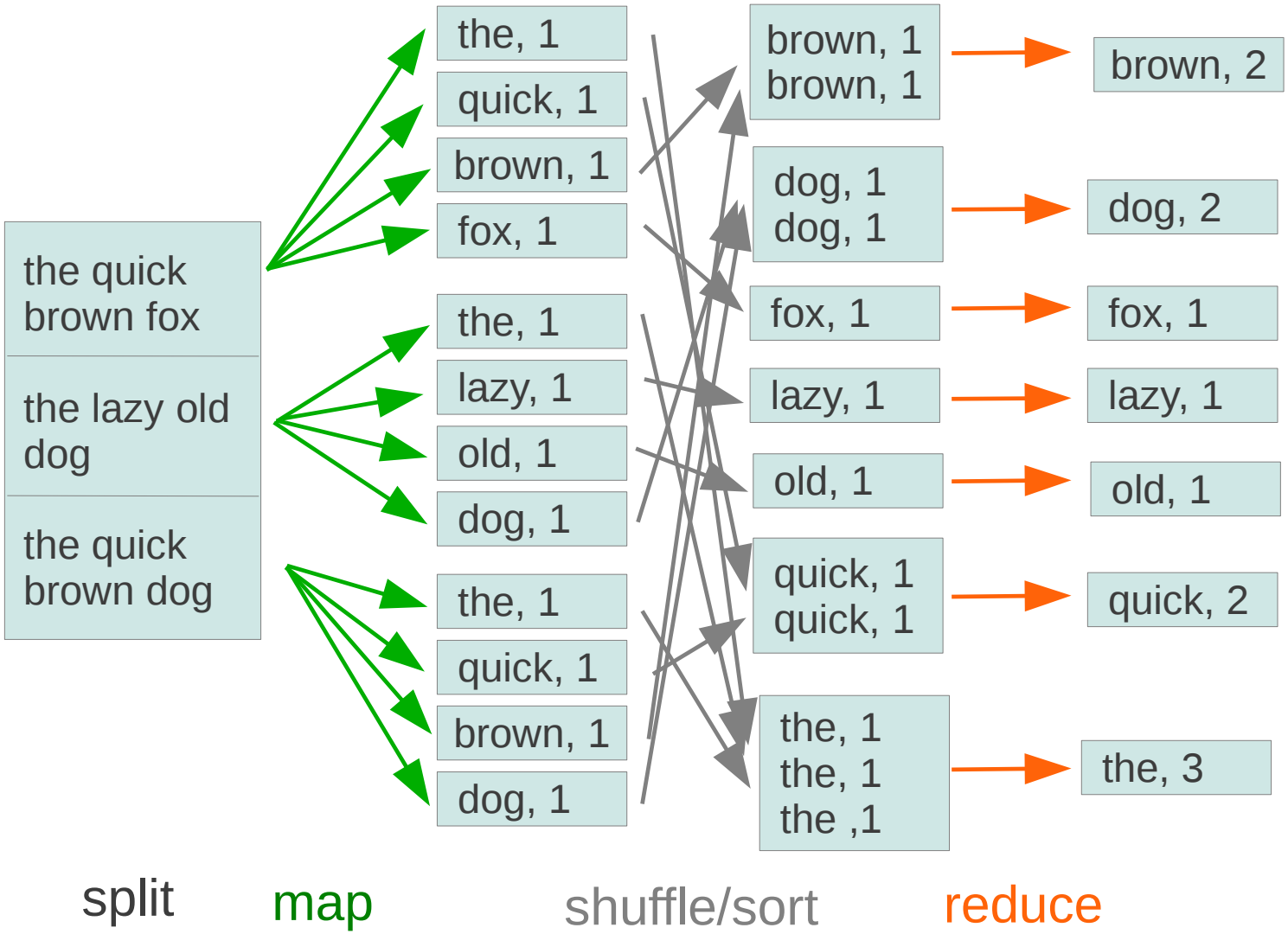
The user does **not** have to think about concurrency, load balancing, data distribution, fault-tolerance!

Example: Word Counting

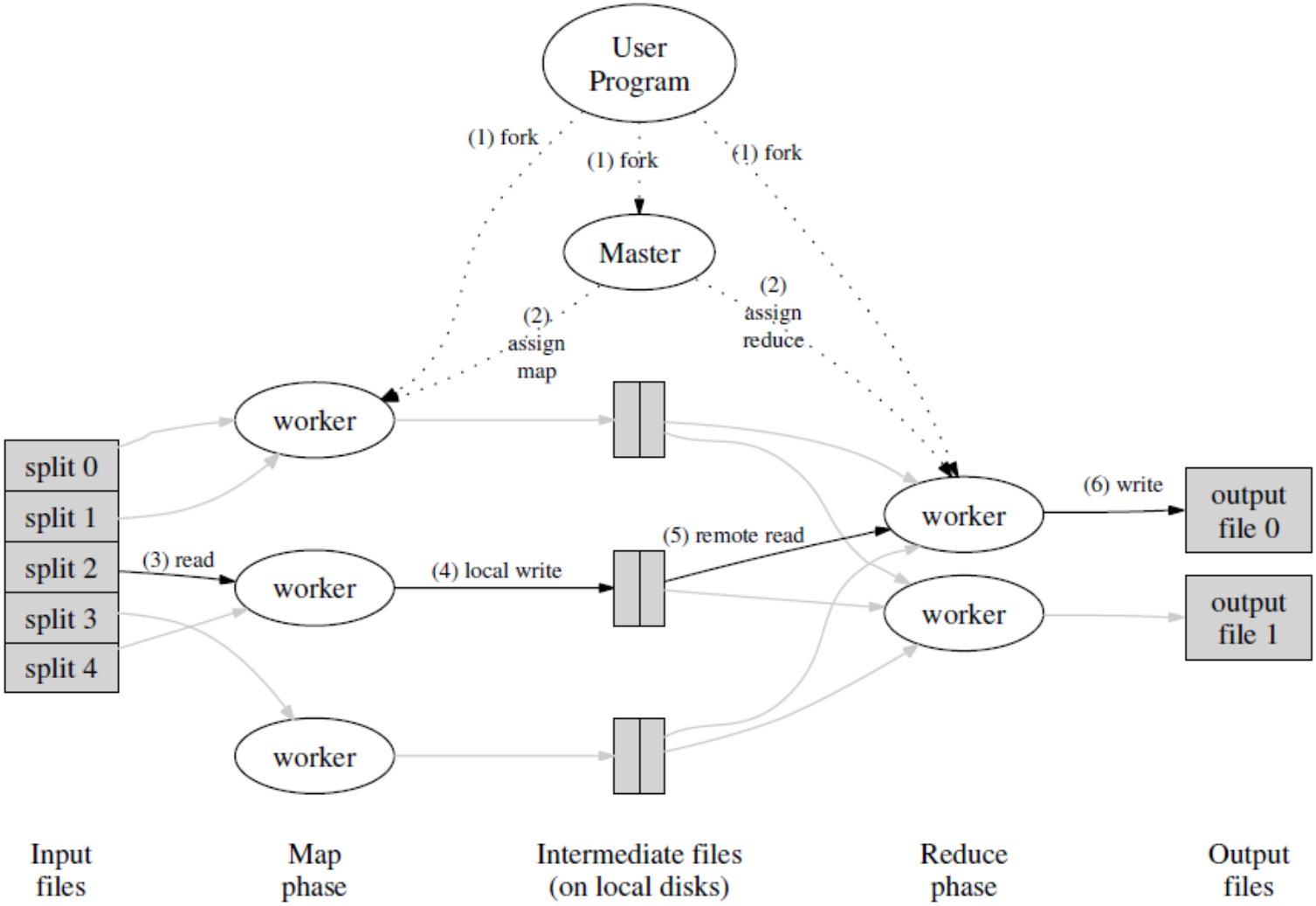
Count how often each word appears in a large number of documents.

```
1  void map(String name, String document){
2      // name: document name
3      // document: document contents
4      for each word w in document:
5      EmitIntermediate(w, "1")
6  }
7
8  void reduce(String word, Iterator partialCounts){
9      // word: a word
10     // partialCounts: a list of aggregated partial counts
11     int sum = 0;
12     for each pc in partialCounts:
13     sum += ParseInt(pc);
14     Emit(word, AsString(sum));
15 }
```

Example: Word Counting

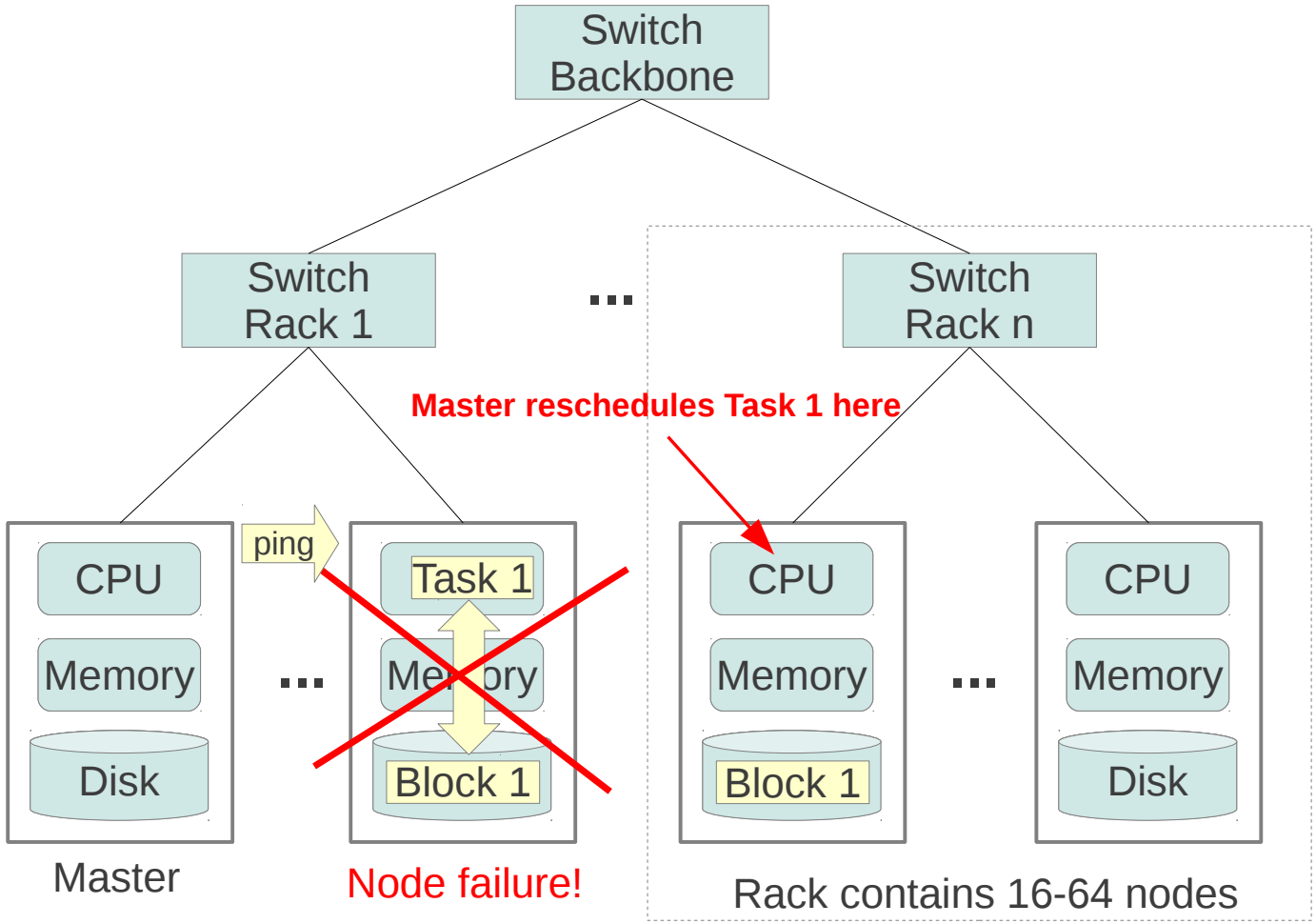


Execution of a MapReduce Job



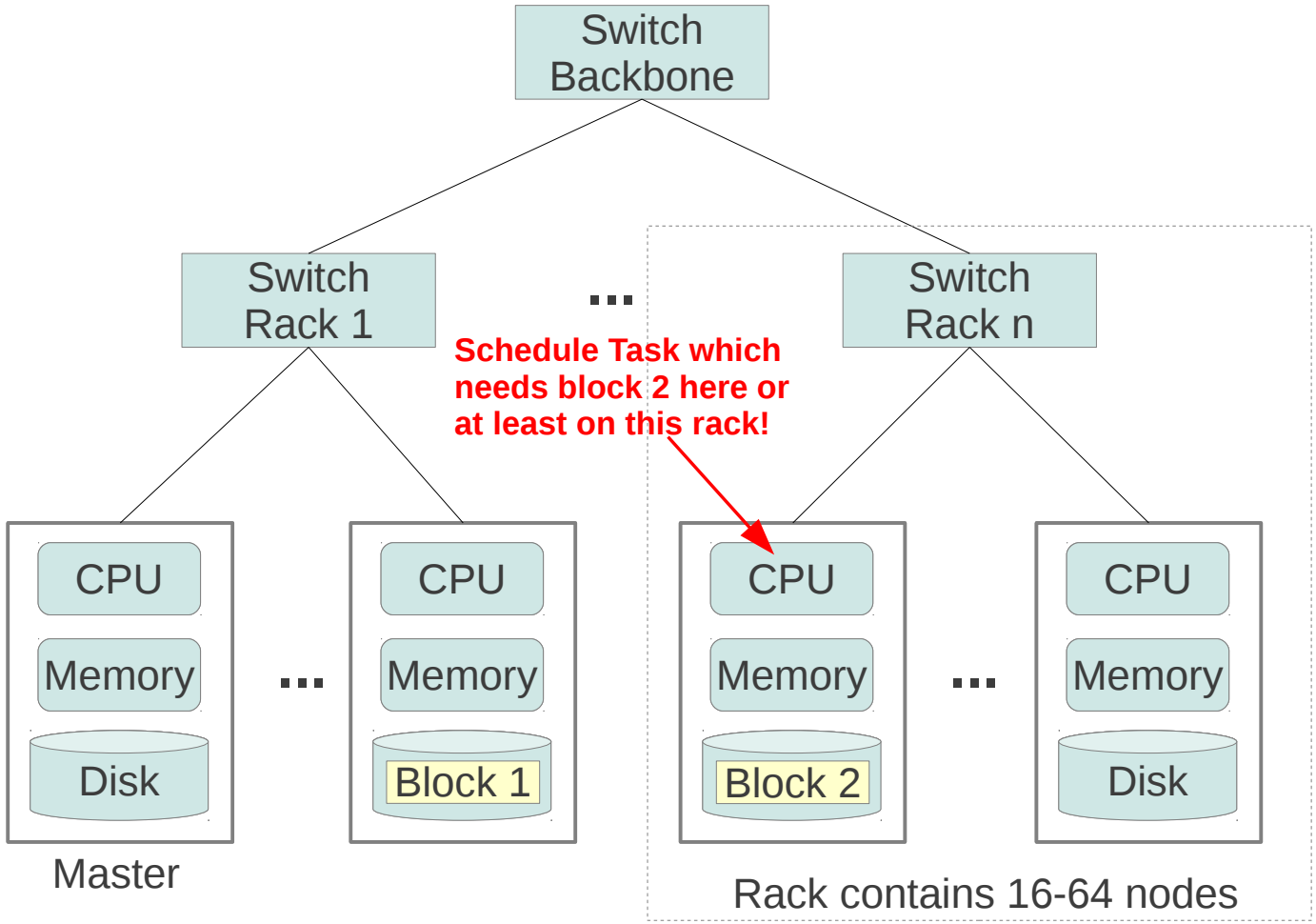
Source: Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, OSDI, 2004.

Fault-tolerance



Master re-executes tasks for failed workers.

Locality



Schedule map tasks near to the data to preserve network bandwidth.

More Properties

- **Load Balancing:** Subdivide work in many small tasks ($\gg$ # of workers). Since the master dynamically assigns tasks to idle nodes, this automatically provides load balancing.
- **Chaining:** MapReduce operations can be chained (output of one operation is input for another operation) to solve more complicated computations.
- **Backup tasks:** Reduce phase can only start after all map tasks are finished (use backup tasks to avoid "stragglers").

Table of Contents

1 Introduction

2 Hadoop Distributed File System

3 MapReduce

4 More Examples

Example: Distributed Grep

Map task?

Reduce task?

Example: Creating an Inverted Index

Map task?

Reduce task?

Related Projects

- **Apache HBase:** open source, non-relational, distributed database modeled after Google's BigTable
- **Apache Hive:** data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Provides a SQL-like query language called HiveQL.
- **Apache Pig:** is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs.
- **Apache Mahout:** free implementations of distributed or otherwise scalable machine learning algorithms on the Hadoop platform.
- **Apache Lucene:** a high-performance, full-featured text search engine library written entirely in Java.

Reading

Jeffrey Dean and Sanjay Ghemawat,
MapReduce: Simplified Data Processing on Large Clusters
`http://labs.google.com/papers/mapreduce.html`

The Apache Hadoop Project
`http://hadoop.apache.org/`