Chapter 2:
Map-Reduce and the New Software Stack

崔金华
电子邮箱：jhcui@hust.edu.cn
个人主页: https://csjhcui.github.io/

致谢:课件主要参考《 Mining of Massive Datasets》(Second Edition) J. Leskovec, A. Rajaraman, J. Ullman的相关课件, 特此致谢!!!

Background

- Much of the course will be devoted to large scale computing for data mining
- Challenges:
  - How to distribute computation?
  - Distributed/parallel programming is hard

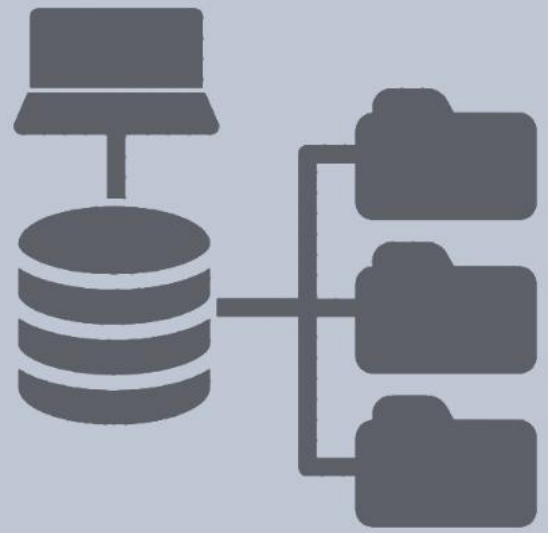
- **MapReduce addresses all of the above**
  - Google's computational/data manipulation model
  - Elegant way to work with big data

Contents

2.1 Distribute File System

2.2 Computational Model: MapReduce

2.3 MapReduce Refinements



Section 2.1: Distribute File System

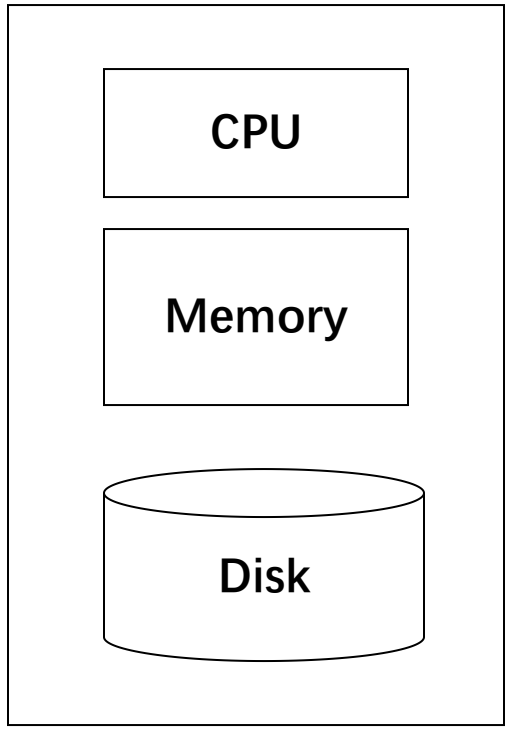
Content

1 Single Node Architecture

2 Cluster Architecture

3 Distributed File System

2.1.1 Single Node Architecture



Machine Learning, Statistics

"Classical" Data Mining

Big→ Not Enough!

2.1.2 Motivation: Google Example

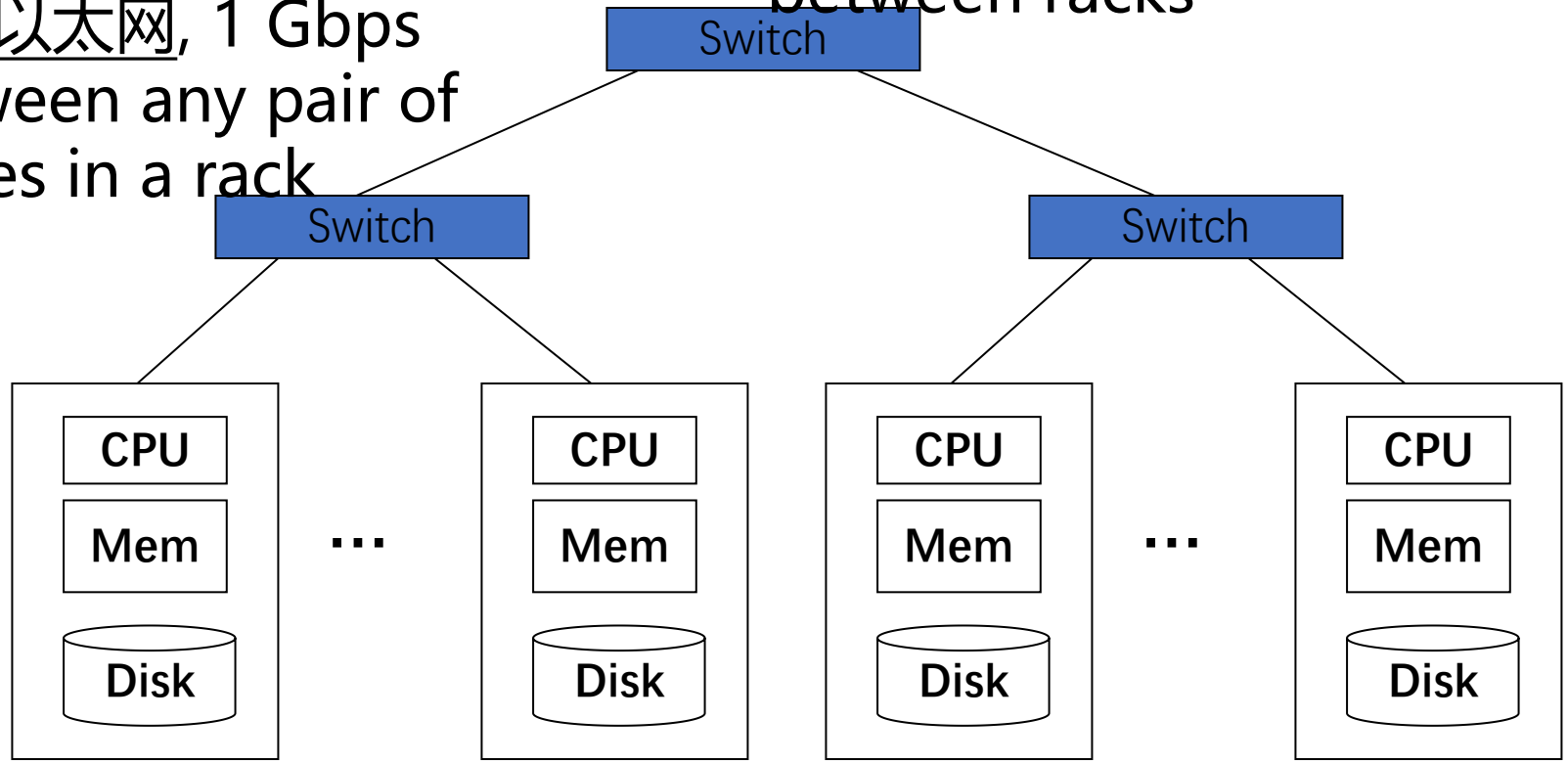
- 200亿网页 x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
  - ~4 months to read the web (**unacceptable!**)
- ~1,000 hard drives to store the web ← then, 2 hours to read the web
- Takes even more to do something useful with the data!

- Today, a standard architecture for such problems is emerging:
  - **Cluster** of commodity Linux **nodes**
  - Commodity network (ethernet) to connect them

2.1.2 Cluster Architecture

万兆以太网, 2-10 Gbps backbone between racks

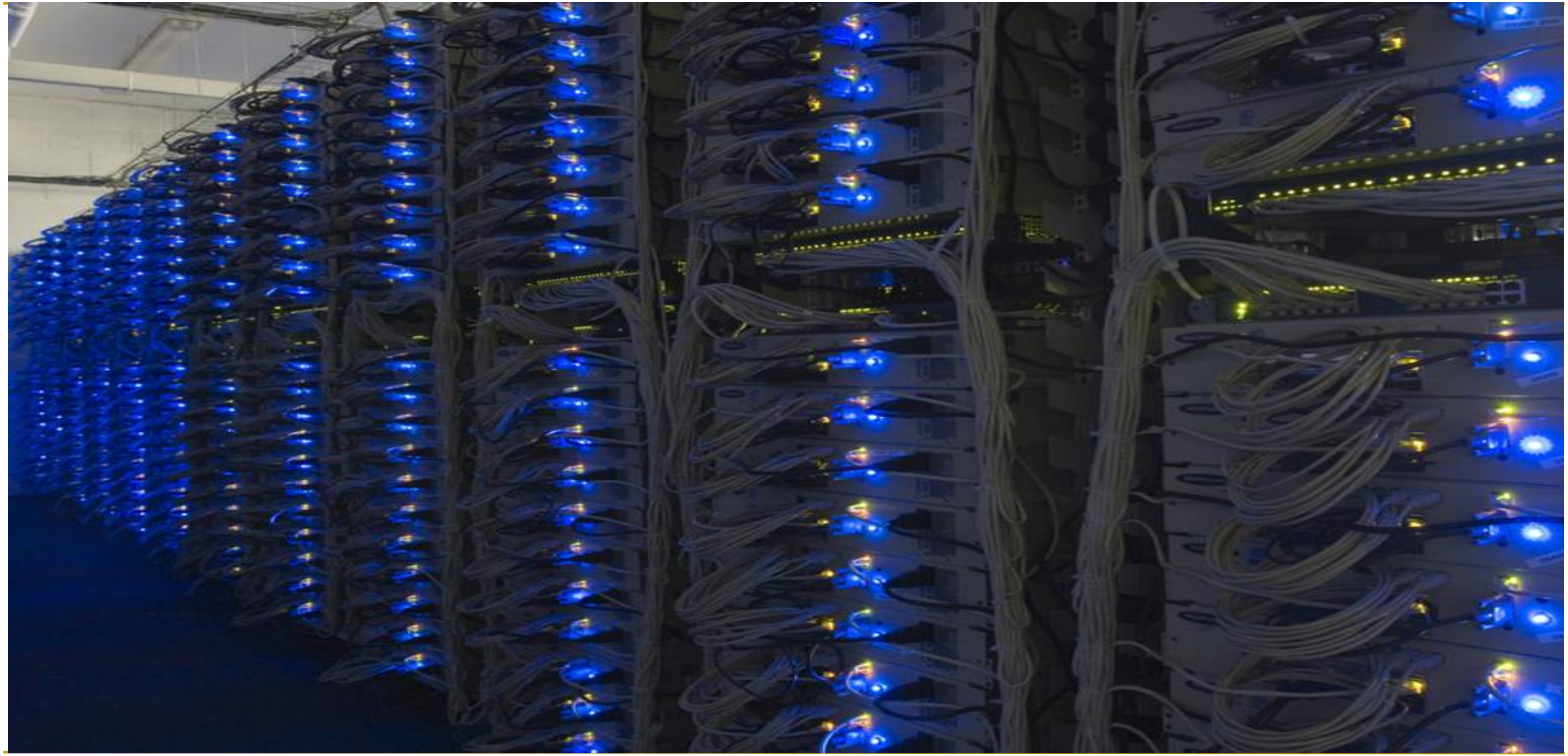
千兆以太网, 1 Gbps between any pair of nodes in a rack



Each rack (机架) contains 16-64 nodes (节点)

In 2011 it was guestimated that Google had 100万台 machines, http://bit.ly/Shh0RO

2.1.2 Cluster Architecture



2.1.2 Large-scale Computing

- **Large-scale computing** for **data mining** problems on **commodity hardware**
- **Still have some challenges:**
  - **1、Network bottleneck. How do you distribute computation?**
  - **2、How can we make it easy to write distributed programs?**
  - **3、Machines fail:**
    - One server may stay up 3 years (1,000 days). If you have 1,000 servers, expect to loose 1/day
    - People estimated Google had 100万 machines in 2011. Then, 1,000 machines fail every day!
    - How to store data under nodes fail? How to compute when some nodes fail?

2.1.2 Idea and Solution

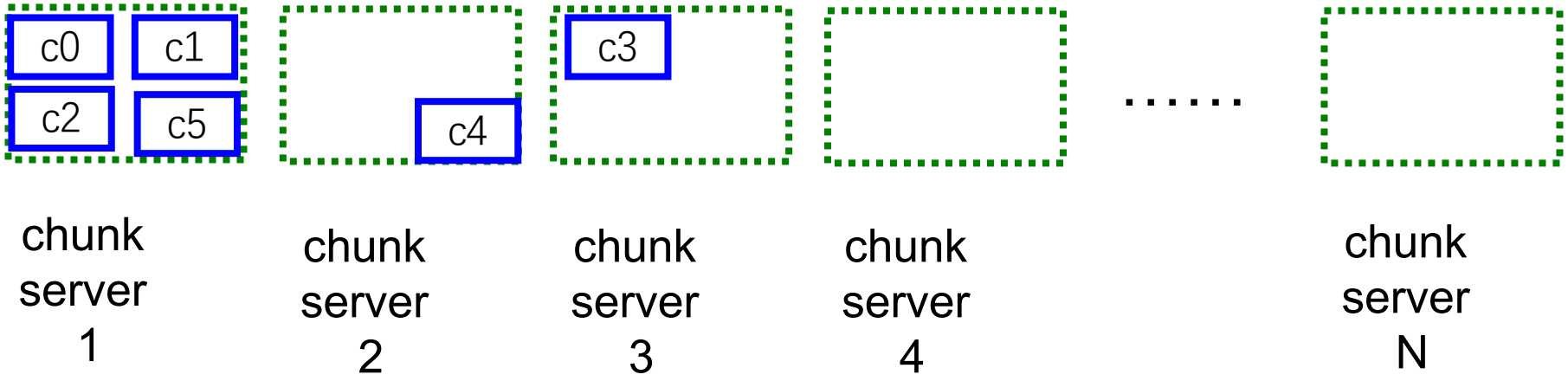
- **Map-Reduce** addresses these problems
  - Store data redundantly: Store files multiple times for reliability
  - Bring computation close to data: minimize data movement
  - Simple programming model: Map-Reduce, hide the complexity of distributed programs

2.1.3 Storage Infrastructure

- Problem:
  - If nodes fail, how to store data persistently?
- Answer:
  - **Distributed File System (DFS, 分布式文件系统)**:
    - Provides global file namespace
    - E.g., Google GFS; Hadoop HDFS;
- Typical usage pattern
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Reads and appends are common

2.1.3 Distributed File System

- **Chunk servers (块服务器)**
  - File is split into contiguous chunks(文件块, 或称块)
  - Typically each chunk is 16-64MB

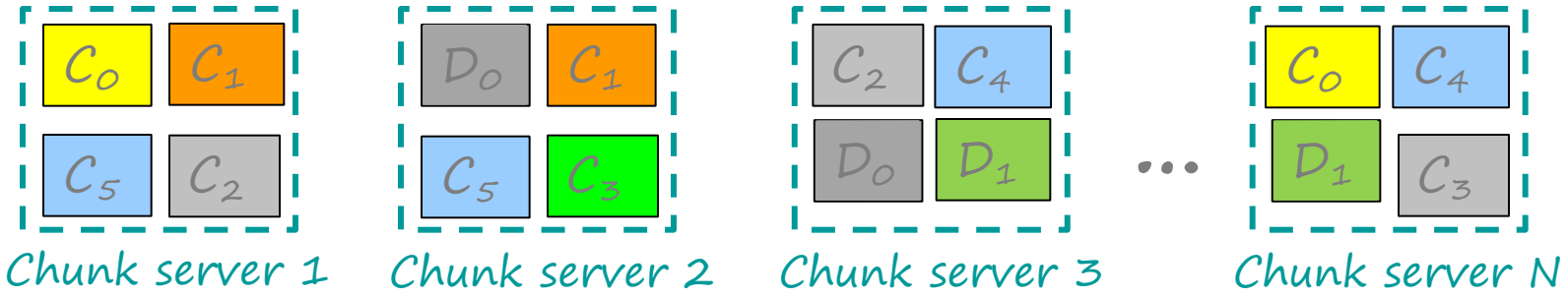


Don't put all your eggs into one basket.

2.1.3 Distributed File System

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST

- **Chunk servers (块服务器)**
  - File is split into contiguous chunks(文件块, 或称块)
  - Typically each chunk is 16-64MB
  - Each chunk replicated (usually 2x or 3x)
  - Try to keep replicas in different racks



Bring computation directly to the data!

Chunk servers also serve as compute servers

2.1.3 Distributed File System

- **Master node (主节点)**
  - a.k.a. Name Node(名字节点) in Hadoop's HDFS
  - Stores metadata about where files are stored
  - Might be replicated
- **Client library for file access**
  - Talks to master to find chunk servers
  - Connects directly to chunk servers to access data

Section 2.2: Programming Model MapReduce

Content

1 What is MapReduce?

2 MapReduce Environment

2.2.1 Programming Model: MapReduce

- **MapReduce** is a programming model for data processing
  - MapReduce的取名来源于该模型中包括map和reduce两个核心操作. 例如通过map操作获取海量网页的内容并建立索引，利用reduce操作根据网页索引处理关键词.

- The power of MapReduce lies in its ability to scale to 100s or 1000s of computers, each with several processor cores

2.2.1 Programming Model: MapReduce

- **Warm-up task:**
  - We have a huge text document
  - Count the number of times each distinct word appears in the file
  - This is called word count task.

- **Sample application:**
  - Analyze web server logs to find popular URLs

2.2.1 Task: Word Count

- **Case 1:**
  - File too large for memory, but all <word, count> pairs fit in memory
  - Method: HashTable
- **Case 2:**
  - Even the <word, count> pairs do not fit in memory
  - Count occurrences of words: words(doc.txt) | sort | uniq -c
    - where words takes a file and outputs the words in it, one per a line
- Case 2 captures the essence of **MapReduce**
  - Great thing is that it is naturally parallelizable

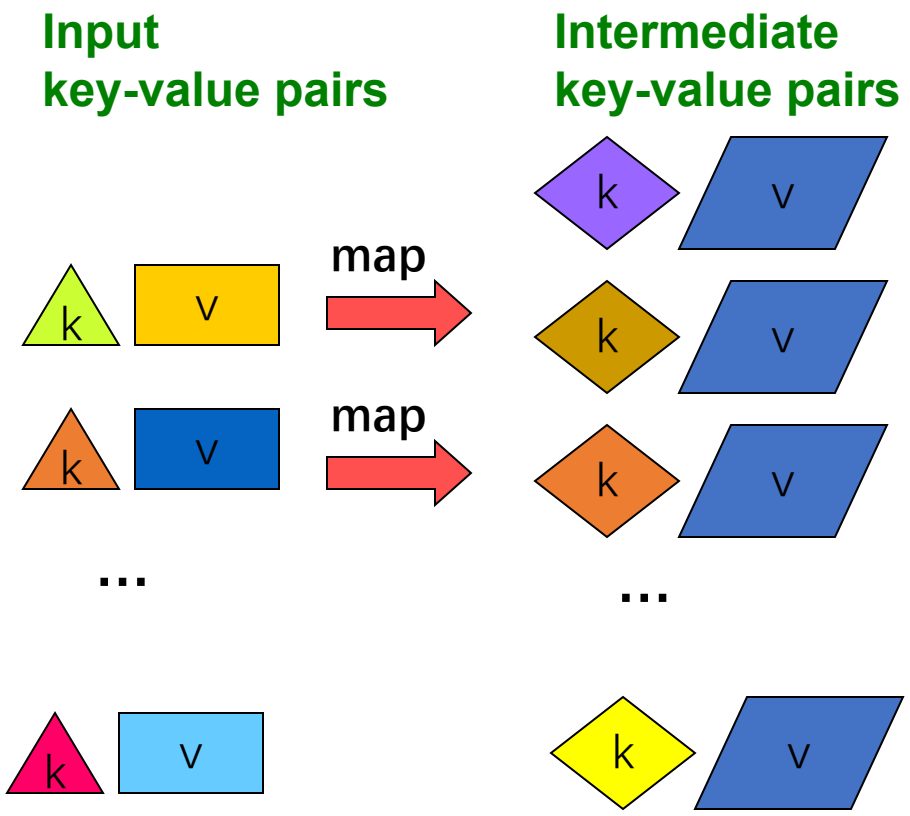
2.2.1 MapReduce: Overview

words(doc.txt) | sort | uniq -c

- Map任务:
  - Scan input file record at a time
  - Extract something you care about
- Group by key(按键分组): Sort and Shuffle
- Reduce任务:
  - Aggregate, summarize, filter or transform
  - Write the result

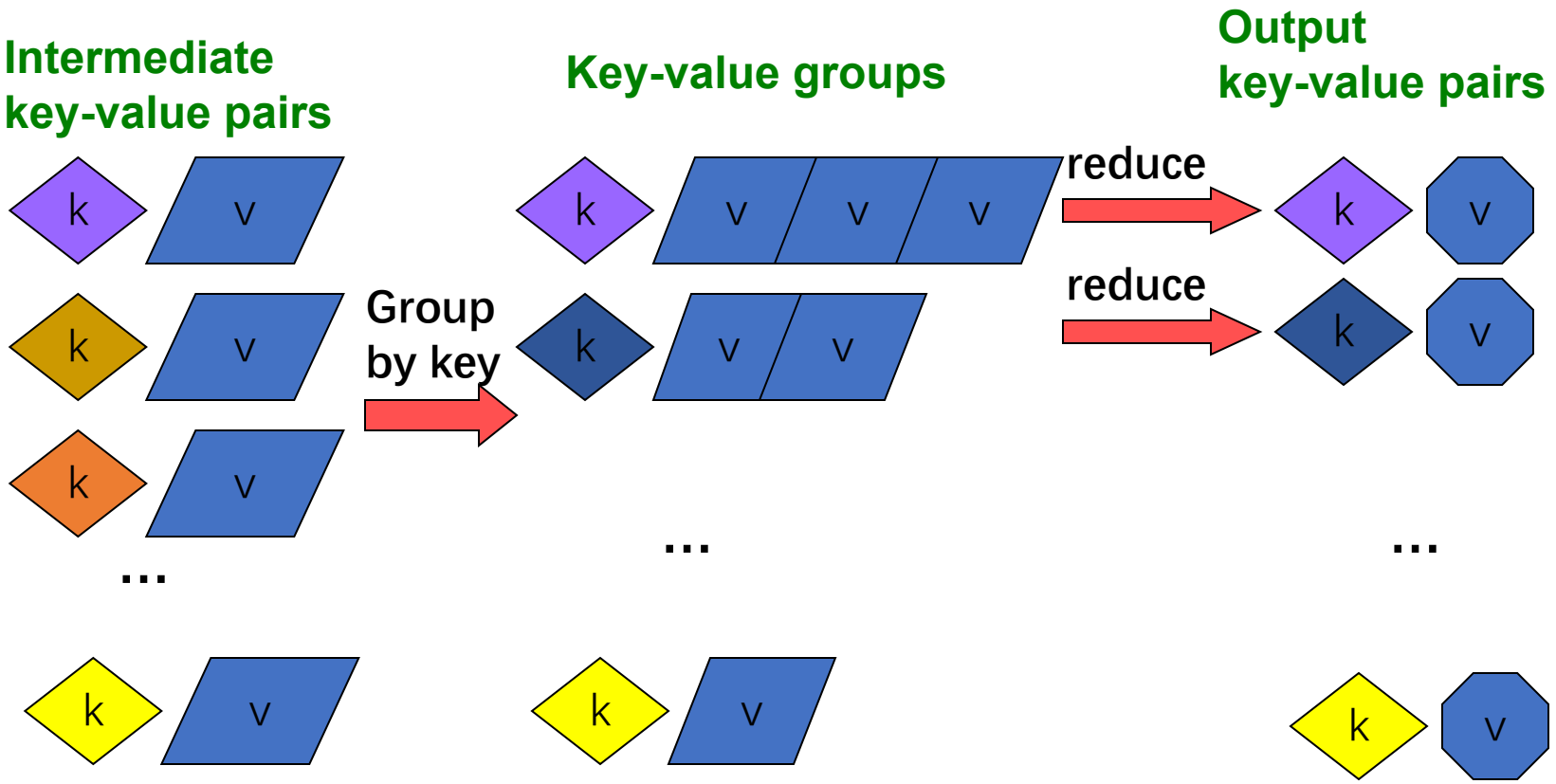
Outline stays the same, **Map** and **Reduce** change to fit the problem

2.2.1 MapReduce: The Map Step



2.2.1 MapReduce: The Reduce Step

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST



2.2.1 More Specifically

- Input: a set of key-value pairs
- Programmer specifies two methods:
  - **Map(k, v)** → <k', v'>*
    - Takes a key-value pair and outputs a set of key-value pairs
      - E.g., key is the filename, value is a single line in the file
    - There is one Map call for every (k,v) pair
  - **Reduce(k', <v'>*)** → <k', v''>*
    - All values v' with same key k' are reduced together and processed in v' order
    - There is one Reduce function call per unique key k'

2.2.1 MapReduce: Word Counting