Map-Reduce

Learning Objectives:

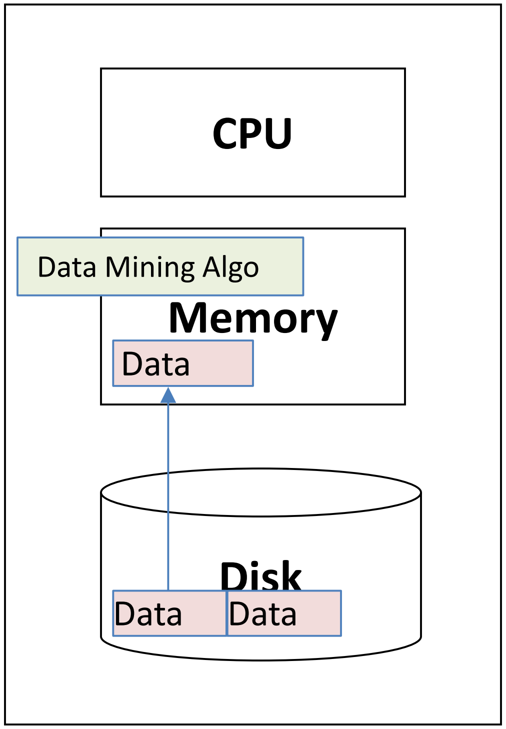
Distributed File System Basics.

Challenges

Why Map-Reduce?

Mining of Massive Datasets

Single Node Architecture

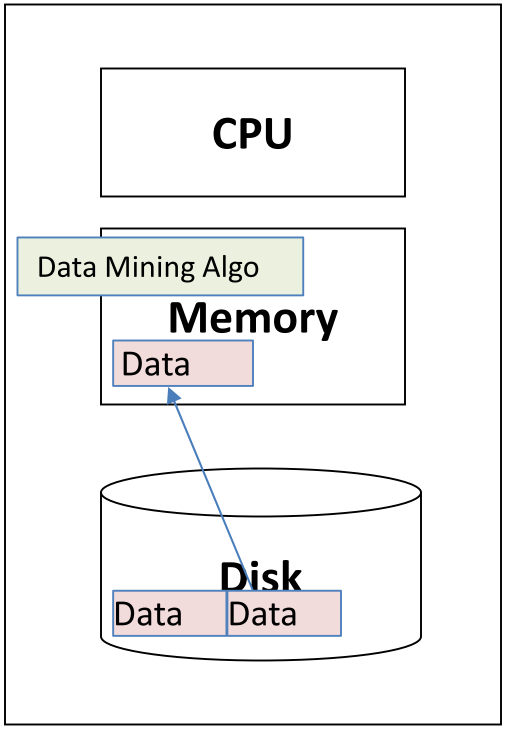


Machine Learning, Statistics

“Classical” Data Mining

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Single Node Architecture



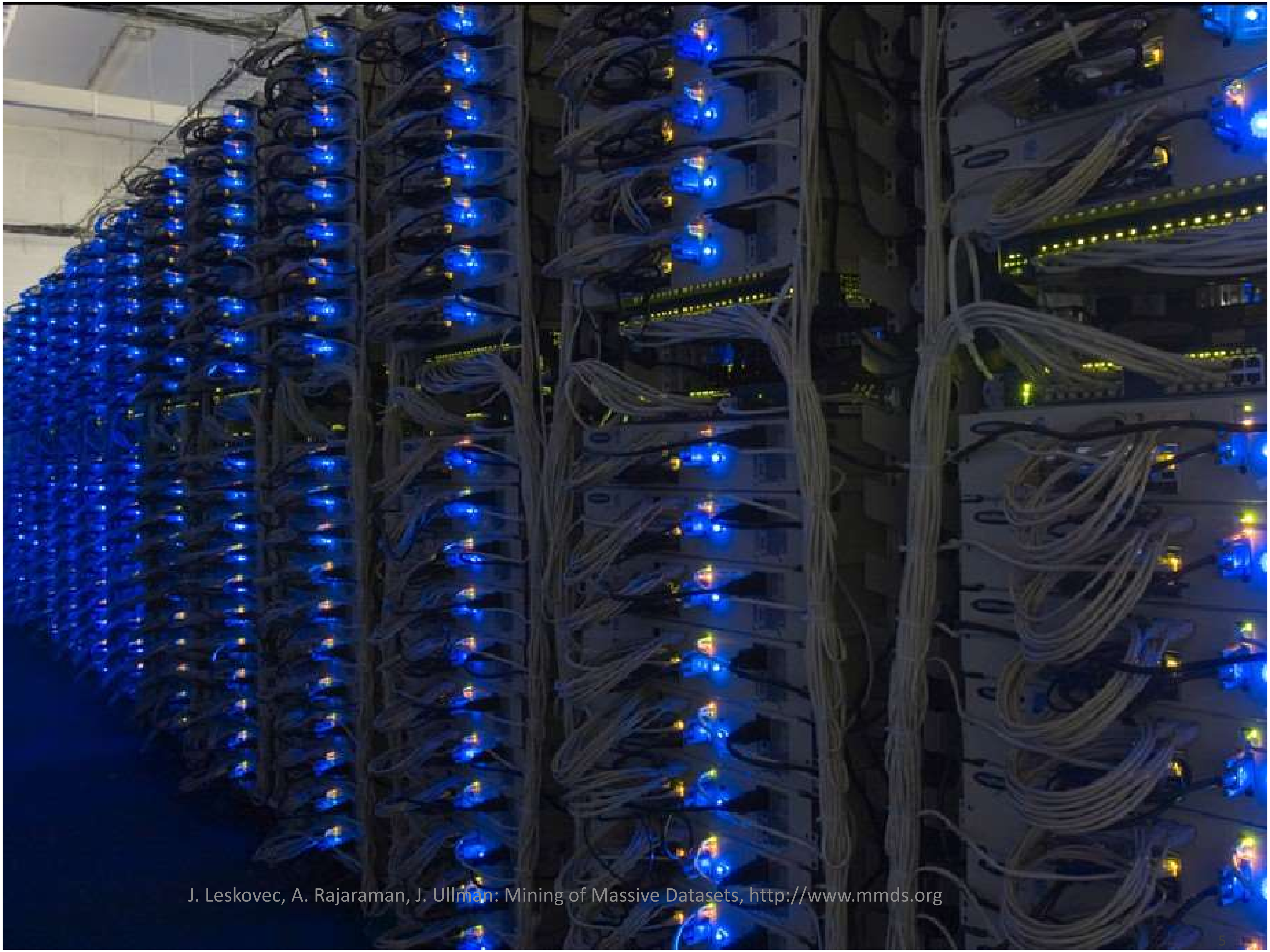
Machine Learning, Statistics

“Classical” Data Mining

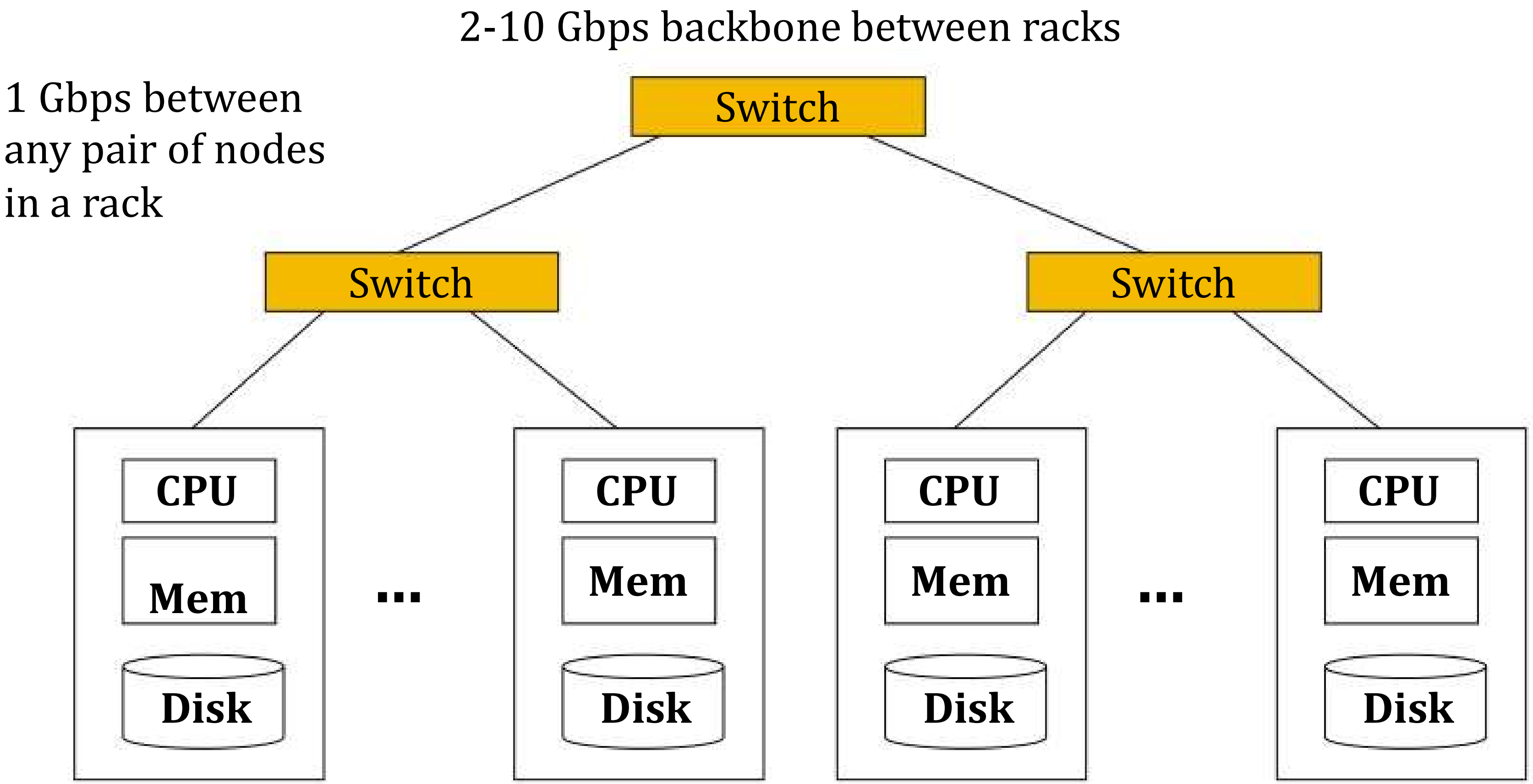
J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Motivation: Google Example

- 10 billion web pages
- Average size of webpage = 20KB
- 10 billion * 20KB = 200 TB
- Disk read bandwidth = 50 MB/sec
- Time to read = 4 million seconds = 46+ days
- Even longer to do something useful with the data



Cluster Architecture


2-10 Gbps backbone between racks
1 Gbps between any pair of nodes in a rack
Switch
Switch
Switch
CPU
Mem
Disk
…
CPU
Mem
Disk
CPU
Mem
Disk
…
CPU
Mem
Disk

Each rack contains 16-64 commodity Linux nodes

In 2011 it was guestimated that Google had 1M machines, http://bit.ly/Shh0RO

Cluster Computing Challenges (1)

- Node failures
  - A single server can stay up for 3 years (1000 days)
  - 1000 servers in cluster => 1 failure/day
  - 1M servers in cluster => 1000 failures/day

- How to store data **persistently** and keep it **available** if nodes can fail?

- How to deal with node failures during a long-running computation?

Cluster Computing Challenges (2)

- Network bottleneck
  - Network bandwidth = 1 Gbps
  - Moving 10TB takes approximately 1 day

- Distributed programming is hard!
  - Need a simple model that hides most of the complexity

Map-Reduce

- Map-Reduce addresses the challenges of cluster computing

  - Store data redundantly on multiple nodes for persistence and availability

  - Move computation close to data to minimize data movement

  - Simple programming model to hide the complexity of all this distributed computing.

Redundant Storage Infrastructure

Distributed File System

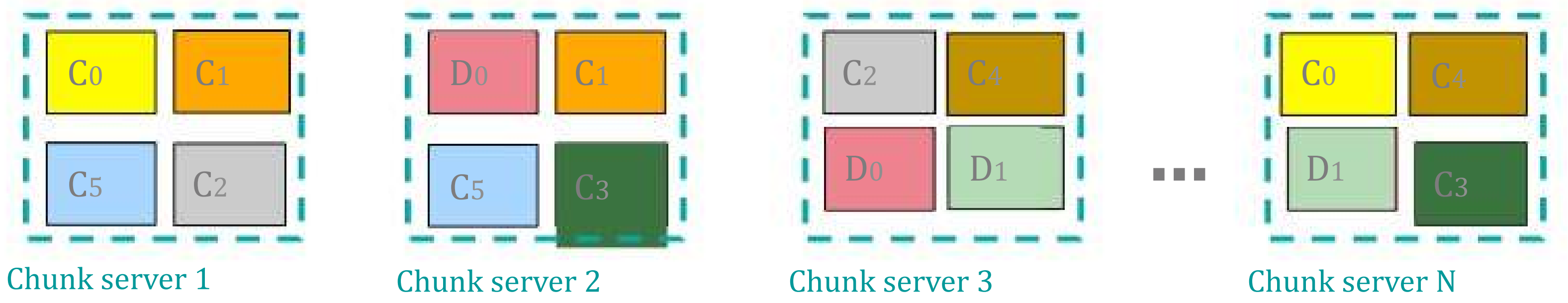
- Provides global file namespace, redundancy, and availability
- E.g., Google GFS; Hadoop HDFS

Typical usage pattern

- Huge files (100s of GB to TB)
- Data is rarely updated in place
- Reads and appends are common

Distributed File System

- Data kept in “chunks” spread across machines
- Each chunk replicated on different machines
  - Ensures persistence and availability



Chunk servers also serve as compute servers

Bring computation to data!

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Distributed File System

Chunk servers

- File is split into contiguous chunks (16-64MB)
- Each chunk replicated (usually 2x or 3x)
- Try to keep replicas in different racks

Master node

- a.k.a. Name Node 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