CLOUD COMPUTING APPLICATIONS

MapReduce Motivation

Roy Campbell & Reza Farivar

Lesson Outline

- Motivation: Why MapReduce model
- Programming Model
- Examples
  - Word Count
  - Pi Estimation
  - Image Smoothing
  - PageRank
- MapReduce execution

Challenges with Traditional Programming Models (MPI)

- MPI gives you MPI_Send, MPI_Recieve
- Deadlock is possible…
  - Blocking communication can cause deadlock
    - "crossed" calls when trading information
    - example:
      - Proc1: MPI_Receive(Proc2, A); MPI_Send(Proc2, B);
      - Proc2: MPI_Receive(Proc1, B); MPI_Send(Proc1, A);
    - There are some solutions - MPI_SendRecv()
- Large overhead from comm. mismanagement
  - Time spent blocking is wasted cycles
  - Can overlap computation with non-blocking comm.
- Load imbalance is possible! Dead machines?
- Things are starting to look hard to code!

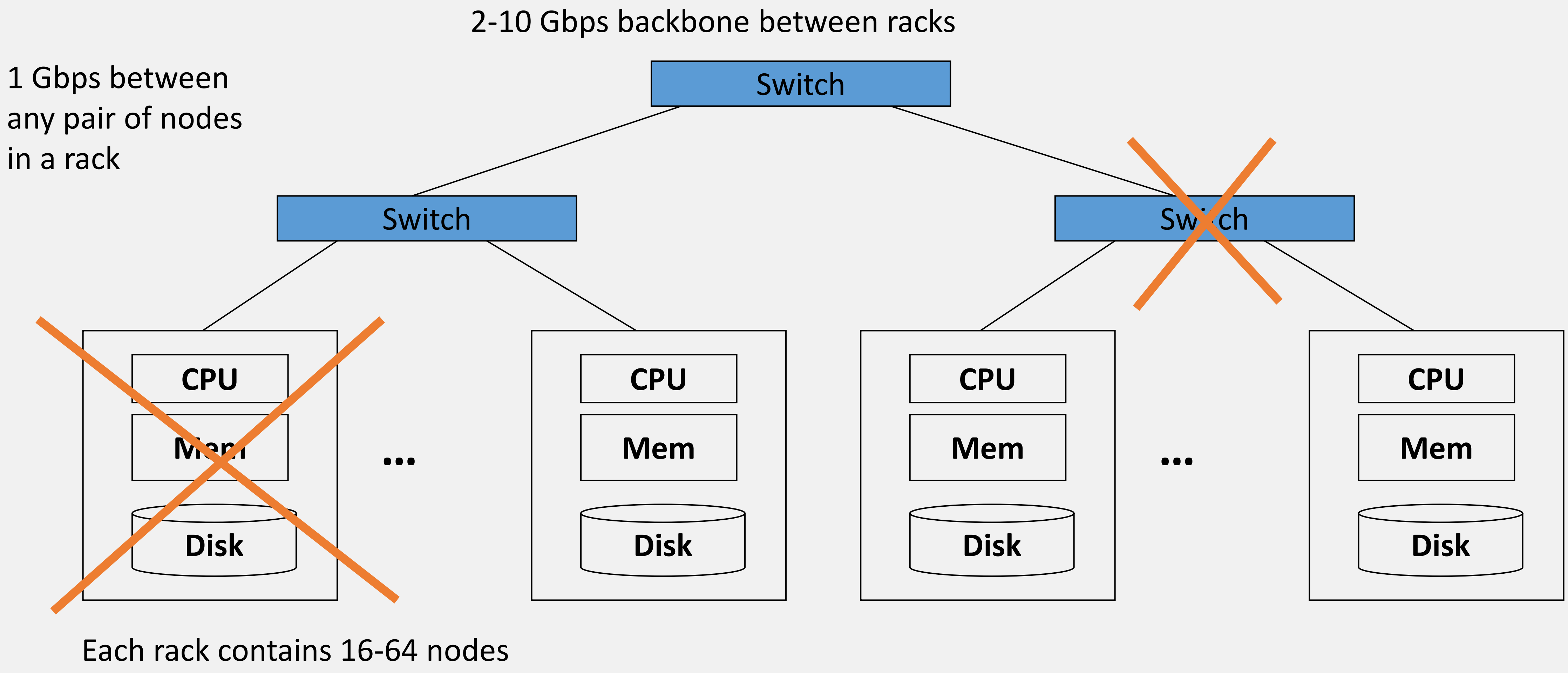
Commodity Clusters

- Web data sets can be very large
  - Tens to hundreds of terabytes
  - Cannot mine on a single server
- Standard architecture emerging:
  - Cluster of commodity Linux nodes
  - Gigabit Ethernet interconnect
- How to organize computations on this architecture?
  - Mask issues such as hardware failure

Solution

- Use distributed storage
  - 6-24 disks attached to a blade
  - 32-64 blades in a rack connected by Ethernet
- Push computations down to storage
  - Computations process contents of disks
  - Data on disks read sequentially from beginning to end
  - Rate limited by speed of disks (speed can get at data)

Cluster Architecture



Stable Storage

- First-order problem: if nodes can fail, how can we store data persistently?
- Answer: Distributed File System
  - Provides global file namespace
  - 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