COMP 543: Tools & Models for Data Science

Big Data: An Intro to MapReduce

Chris Jermaine & Risa Myers

Rice University

15 Years Ago...

- Say you had a big data set, wanted platform to analyze it
- What is “big”?
    - Too large to fit in RAM of an expensive server machine
- Is 5GB in 2002, a couple of terabytes in 2018

You Would Buy a DB System

- Cost A LOT of money
- Performance often unpredictable, or just flat out poor
- Software insanely complicated to use correctly
- Software stack too big/deep, not possible to unbundle
  - If you are doing analysis, ACID not important
  - And yet, you pay for it (money, complexity, performance)
- Difficult to put un- or semi-structured data into an SQL DB

Or You Roll Your Own Software

- Costly, time consuming
  - A $10M software feature might eat up most of the IT budget for a single firm
- Requires expertise not always found in house
- Risky: high potential for failure

Many People Just Don’t Like SQL

- People uncomfortable with declarative programming
  - We love it!
  - But user doesn’t really know what’s happening under the hood
  - Makes many programmers uncomfortable
- Also, not easy/natural to specify important computations
  - Especially data mining and machine learning

By Early-Mid 2000's...

- The Internet companies (Google, Yahoo, etc.)...
  - ...had some of the largest databases in the world
  - But they never used classical SQL databases for webscale
- How'd they do it?
  - Many ways...
  - But paradigm with most widespread impact was MapReduce
  - First described in a 2004 academic paper, appeared in OSDI
  - Easy read! Do a search on "Google MapReduce paper"

What Is MapReduce?

- It is a simple data processing paradigm
- To process a data set:
  - You have two pieces of user-supplied code
  - A Map code
  - And a Reduce code
- These are run in a huge shared-nothing compute cluster
  - Using three data processing phases
  - A Map phase
  - A Shuffle phase
  - And a Reduce phase

First: What Is Shared-Nothing?

- Store/analyze data on a large number of commodity machines
  - Local, non-shared storage attached to each of them
  - Only link is via a LAN
  - Shared nothing refers to no sharing of RAM, storage
- Why good?
  - Inexpensive, built out of commodity components
  - Compute resources scales nearly linearly with money
  - Contrast to shared RAM machine with uniform memory access

MapReduce: The Map Phase

- Input data are stored in a huge file
  - Contains a simple list of pairs of type (key1,value1)
- Have a UDF of the form Map(key1,value1)
  - outputs a list of pairs of the form (key2, value2)
- In the Map phase of the MapReduce computation
  - The Map function is called for every record in the input
  - Instances of Map run in parallel all over the cluster

MapReduce: The Shuffle Phase

- Accepts all of the (key2, value2) pairs from the Map phase
  - And it groups them together
- After grouping, all of the pairs
  - From all over the cluster having the same key2 value
  - Are merged into a single (key2, list <value2>) pair
- Called a Shuffle...
  - Because this is where a potential all-to-all data transfer happens

MapReduce: The Reduce Phase

- Have a user-supplied function of the form
  - Reduce (key2,list <value2>)
  - Outputs a list of value2 objects
- In the Reduce phase of the MapReduce computation
  - Reduce function is called for every key2 value output by the Shuffle
  - Instances of Reduce run in parallel all over the compute cluster
  - The output of all of those instances is collected
  - Put in a (potentially) huge output file

MapReduce Is a Compute Paradigm

- It is not a data storage paradigm
    - But any MapReduce system
    - Must read/write data from some storage system
- So MapReduce strongly linked with the idea of a distributed file system (DFS)
    - Allows data to be stored/accessed across machines in a network
    - Abstracts away differences between local and remote data
    - Same API to read/write data
    - No matter where data is located in the network

Distributed File Systems for MR

- DFSs have been around for a long time
  - First widely used DFS was Sun's NFS, first introduced in 1985
- Unlike classical DFS, MapReduce DFS sits on top of each machine's OS
  - Lives in "user space"
  - The OS is not aware of the DFS
  - You can't mount it anywhere
- Why on top of, not in the OS?
  - Hetrogeneity no problem
  - Easily portable (JVM)
- Even as MapReduce becomes less popular, MR DFS lives on!

Questions?