Introduction I

Dmitri Loguinov
Texas A&M University

September 1, 2015
Agenda

- Syllabus
- Homework requirements
- Motivation
- What is data computing?
- What is scalability?
- Wrap-up
Syllabus

• Office hours
  – TR 5:10-6:10 pm in 515C HRBB
  – Website: http://irl.cse.tamu.edu/courses/689
  – Forum: https://piazza.com/tamu/fall2015/csce689

• All lectures and homework will be on the website
  – Including hints on using Latex and various support files

• Final grades
  – A: 90-100%
  – B: 80-89%
Syllabus 2

• Assignments/exams
  – Midterm: 20%
  – Final: 20%
  – Homework (4): 60%

• Exams cover half a semester each

• Team work is not allowed

• Reminder: you may not pass any material from the web, other students, or publications as your own
Homework

• Homework involves analysis and implementation

• Coding must be done in Windows C/C++
  – Use MS Visual Studio 2015
  – Zip entire solutions (suitable for compilation) and submit through csnet.cs.tamu.edu
  – Include .sln, .cpp, .h, and .vc*proj* files
  – Delete all others, but preserve the directory structure
  – The deadline is noon on the day it is due

• Bring to class
  – A hardcopy of your report and program
Homework 2

• Report contents
  – Explain your implementation
  – Outline ideas to further improve efficiency
  – Specify the shortcomings of the studied algorithms (if any)
  – Explain non-trivial phenomena and interesting points
  – Discuss your conclusions and material learned

• Each homework wins a reward if your code is the fastest in class
  – Evaluated in two scenarios (slow/fast disk) on IRL server
  – Two rewards are needed to skip one test

• Since you don’t have access to benchmark server
  – Make sure your code produces the least amount of I/O
  – Has the lowest CPU complexity, maximum parallelization
Motivation

• This course focuses on efficient manipulation of large datasets (commonly known as Big Data)
  – Input files measure tera-, peta-, or soon exabytes
  – Do not fit in RAM, or even a single hard drive

• Examples
  – Twitter: generates 8 TB of content per day
  – Facebook: 40 PB accumulated, 100 TB/day created by users
  – Google page ranking: graphs with 30T nodes (1.4 PB)
  – Google HTML: 1T pages * 30 KB = 30 PB
  – Inverted index for search queries: 1 PB
  – Youtube uploads: 200 TB/day
  – Scientists: space telescopes produce 70 PB/year worldwide
- Ebay: 50 PB warehouse, 1T rows in database
- Cars: Ford Fusion estimated @ 25 GB/hour
- Financial markets: NYSE with 1 TB/day
- Retail: Walmart operated the first commercial database to reach 1 TB (in 1992); it was 2.5 PB in 2008

• **Big Data analytics** is a separate field now
  - Projected as a $125B market in 2015
  - Value in hardware and software for data centers, programmers, algorithms, patents, business advantage from data-mining
Motivation 3

- Exponential growth in data generation
  - Annual world-wide production doubles very two years
  - Around 8 ZB in 2015, perhaps 32 ZB by 2019
  - Internet traffic 900 EB/year according to Cisco

- Traditional databases
  - Use a structured data model (e.g., items stored in table format, rows sorted by key), rely on disk seeking (e.g., B+ trees, binary search), enforce transaction consistency

- Data computing often deals with unstructured data
  - The needed information is scattered all over input
  - The objective is not to insert/delete/retrieve items, but rather process huge input to answer certain questions
Motivation 4

- **Example**: find the most commonly href’ed URL in 100TB of HTML

- Operations on unstructured datasets
  - Linear complexity: scanning (e.g., capture/storage, playback)
  - Super-linear complexity: sort (e.g., aggregation, statistics, recommendations, ranking, usage analysis)

- Big Data computing uses the *streaming* model
  - Sequential scanning of input files, no disk seeking
  - Much faster than transactional databases

- Algorithms for Big Data processing
  - Must be efficient, scalable, easy to extend

- This is our topic in the rest of the semester
What is Data Computing

• Assume an input file that requires extraction of certain information scattered all over it
  - Example: 3TB out-graph from which we need 100 nodes with the largest in-degree
  - Graphs are stored as adjacency lists \((y, d_{\text{out}}(y), x_1, x_2, \ldots)\)

• This is usually solved by a MapReduce computation
  - First published by Google in 2004, now de-facto standard
  - Many proprietary versions exist, but the open-source market is dominated by Hadoop (created by Yahoo in 2005, later moved to Apache)

• MapReduce is a functional language
  - The system calls user-provided functions Map(), Reduce()
What is Data Computing 2

• The mapper accepts an iterator over the file and outputs (key, value) pairs
  – Allows users flexibility to process input any way they want

• Depending on the type of data, the iterator could produce one line of text, one record, or one neighbor list
  – Custom iterators can be coded as well

• The key type must support comparison, whereas the value field can be arbitrary
  – The two functions (f, g) are determined by the user

```c
Map (Iterator *input)
{
    while (!input->empty()) {
        x = input->next();
        output (f(x), g(x))
    }
}
```
What is Data Computing 3

- The reducer accepts a key and a list of all values that were earlier mapped to it.
  - We can view the reducer as applying a transformation \((k, v_1, v_2, \ldots) \rightarrow (k, h(v_1, v_2, \ldots))\) using some combiner function \(h\).

- Note that \(f\) must be scalar, while both \(g\) and \(h\) could produce vectors of elements.

- These two abstractions allow MapReduce to:
  - Transparently distribute data over multiple hosts.
  - Free the user of the burden to write the underlying code.
  - Handle input when the value list is longer than RAM.

```c
Reduce (KeyType key, Iterator *values)
{
    while (!values->empty()) {
        val = values->next();
        // compute/augment result
    }
    output (key, result)
}
```
What is Data Computing 4

- To handle reducer output larger than RAM, the model can be slightly changed →

- **Example**
  - Program accepts the out-graph and produces for each node x its in-degree $d_{in}(x)$

- **How does MapReduce work?**
  - It must detect all duplicate keys, bring the values together
  - Commonly done with distributed, external-memory sorting

- **Example**
  - List the top-10 highest in-degree nodes
What is Data Computing 5

- Assume persistent variables are available
- After all input is finished, your function cleanup() is called

• MapReduce splits input file into many chunks
  - These are distributed to various computers, each running multiple instances of Map()
  - Map results are similarly distributed across servers, which run multiple Reduce() jobs

• Constraint: all values for a given key must be brought into the same host, then reduced locally

• In that case, the basic top-10 program needs change
  - Each server will produce its local top-10 list
  - These results need to be reduced into one global list
What is Data Computing 5

• Another problem is MapReduce doesn’t know how to split the input to allow the mapper to work correctly

• Example
  – List the most-frequent href’ed URL in a large HTML file
  – Can we use the default line-by-line iterator?

• Multiple input files require several types of mappers
  – All of them use a common reducer

• Example
  – Trace analysis: given (crawled page ID, timestamp, out_link1, out_link2, ...), determine the first time t_i each URL i was encountered during the crawl
  – Do the same for the top-10 nodes with highest in-degree
What is Scalability

• Certain algorithms are deemed “unscalable”
  – But what does this mean?

• Notation
  – Assume \( n \) is the size of input (in records, bytes, nodes), \( S \) is the number of servers, \( c \) is the number of cores per host, \( R \) is their RAM size, and \( D \) is the corresponding disk space

• Scalability assesses how well the method performs (in terms of runtime) as \( n \to \infty \)
  – Fixed resources: \((S, c, D, R)\) are all constant
  – Vertically scaled: \( S \) constant, but \((c, D, R)\) increase with \( n \)
  – Horizontally scaled: \( S \) increases with \( n \), the other parameters constant
What is Scalability 2

- Vertical scaling has obvious limitations
  - Modern data centers and cloud networking utilize horizontally scaled infrastructure (many cheap servers)
  - Google, Facebook, Microsoft, Amazon lead in cluster size
  - Well over 1M servers per company

- MapReduce is a framework that scales well
  - Data partitioning allows each host to be loaded with small chunks of computation, easily parallelized to all cores
  - Chunks are shuffled during the reduce phase to bring all keys in a given range to a single host
  - As $n \to \infty$, runtime remains constant if $n/S = \Theta(1)$