Data-intensive computing systems

Introduction

Universtity of Verona
Computer Science Department

Damiano Carra

Acknowledgement and contacts

- *Credits*
  - *Part of the course material is based on slides provided by the following authors*
    - *Pietro Michiardi, Jimmy Lin*

- *Contacts*
  - Office hours (➔ Ca' Vignal 2, 1st floor, #82)
    - Thursday, 14.30 – 16.30 (check the website for last-minute changes)
    - Based on agreement (via email)
  - Email:

damiano.carra@univr.it

Information and Background

❑ Main source of information

- course web site
  - Slides
  - Detailed course schedule
    - roughly: 2 hours (theory) + 2 hours (lab) per week
  - Note that the schedule may change, so keep checking it!

❑ Background

- Necessary: Java programming
- Suggested: Basic Database course

Exam

❑ Based on a project

- Design and implementation of solutions to analyze different data sets
- Focus on the efficiency and the performance of the proposed solution

❑ The project output will be

- The implementation (source code)
- A technical report with
  - implementation details of the solution
  - results of the analysis of the data sets
  - performance analysis
    - varying cluster size or system parameters

    ➔ The code will probably be used on a real cluster of machines… still working on that, so stay tuned

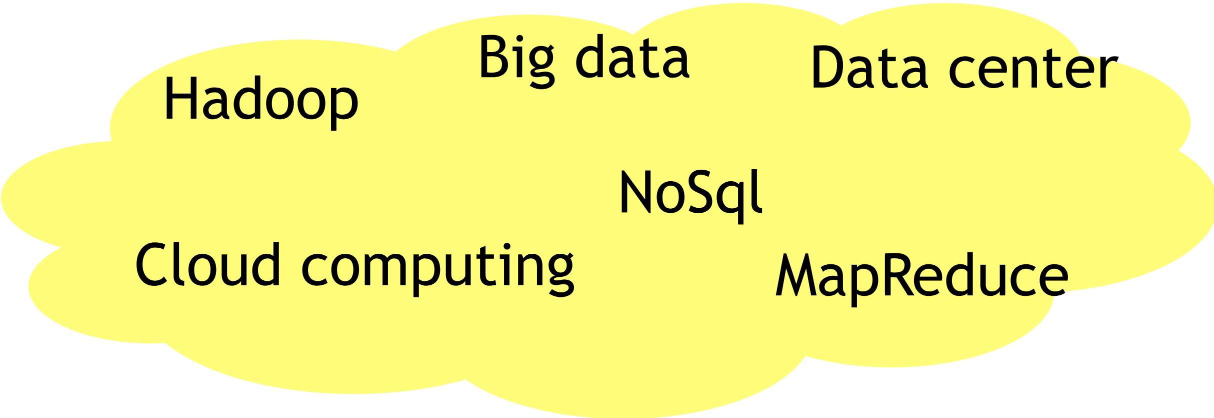
Course material

- The principal textbooks for this course are:
  - Jimmy Lin, Chris Dyer: “Data-Intensive Text Processing with MapReduce”
    - The pdf can be downloaded here: http://lintool.github.io/MapReduceAlgorithms/ed1n.html
  - Tom White: “Hadoop: The Definitive Guide”
    - A copy will be available at the library
  - A. Rajaraman, J. Leskovec, J.D. Ullman: “Mining of Massive Datasets”
    - Not necessary, it covers many other topics, but some chapters are interesting
    - The pdf can be downloaded here: http://infolab.stanford.edu/~ullman/mmds.html
- Readings from other sources will be pointed during the classes.

- IMPORTANT: The slides are a reference to the topics covered during the course
  - Their content has much less information than the textbooks

Introduction and motivations

A lot of keywords…



- ❑ After this course, these keywords (and much more) will have, hopefully, a meaning

- ❑ Let's start with… Big data

How much data?

- ❑ Google → 20 PB/day (2008)
- ❑ Facebook → 90 TB/day (2010)
- ❑ LSST → 3 TB/day of image data
- ❑ LHC → 10/15 PB/year
- ❑ and much more…
  - Amazon, NYT, DNA sequencing

- ❑ Is a lot of data enough for big data?
  - **Volume, Velocity, Variety**

Challenges

- ❑ Traditional parallel supercomputers are not the right fit for many problems (given their cost)
  - Optimized for fine-grained parallelism with a lot of communication
  - Cost does *not* scale linearly with capacity
- ➔ Clusters of commodity computers
  - Even more accessible with pay-as-you-go cloud computing

Parallel computing is hard!

Fundamental issues
scheduling, data distribution, synchronization, inter-process communication, robustness, fault tolerance, …

Different programming models
- Message passing
- Shared memory

Architectural issues
Flynn's taxonomy (SIMD, MIMD, etc.), network typology, bisection bandwidth
UMA vs. NUMA, cache coherence

Common problems
livelock, deadlock, data starvation, priority inversion…
dining philosophers, sleeping barbers, cigarette smokers, …

Different programming constructs
mutexes, conditional variables, barriers, …
masters/slaves, producers/consumers, work queues, …

The reality: programmer shoulders the burden of managing concurrency…

How to process big data?

- ❑ We are looking at newer
  - Programming models
  - Supporting algorithms and data structures
    - More data leads to better accuracy
    - With more data, accuracy of different algorithms converges
- ❑ NSF refers to it as "data-intensive computing" and industry calls it "big-data" and "cloud computing"

How to process Big-data? Main Ideas

- ❑ Scale "out", not "up"
- ❑ Assume failures are common
  - Probability of "no machine down" decreases rapidly with scale…
- ❑ Move processing to the data
  - Bandwidth is scarce
- ❑ Process data sequentially
  - Seeks are *very* expensive
- ❑ Hide system-level details from the application developer

Big-Data: Targeted problems

- Embarrassingly parallel problems
  - Simple definition: independent (shared nothing) computations on fragments of the dataset
  - It's not easy to decide whether a problem is embarrassingly parallel or not
- Batch processing of data-intensive workloads
  - Involving (mostly) full scans of the dataset
  - Generally not processor demanding
    - E.g., read and process the whole Internet dataset from a crawler
  - Relevant datasets are too large to fit in memory

This course

- We will study current BigData solutions
  - Systems challenges
  - Programming models
  - Dealing with failures
- We will look at some applications
  - Information retrieval, data mining, graph mining, traffic processing, …
- Possibly
  - Identify shortcomings, limitations
  - Address these!

Basic example: Word count

- ❑ Assume to have a large collection of texts
  - e.g., Web pages from the whole Internet
- ❑ We would like to count how many times each word is mentioned all over the collection
  - it represents the basis for more complex computations, such as frequencies, pairings, etc
- ❑ Assuming that the collection is distributed among N machines, how would you proceed?

Basic example: Word count

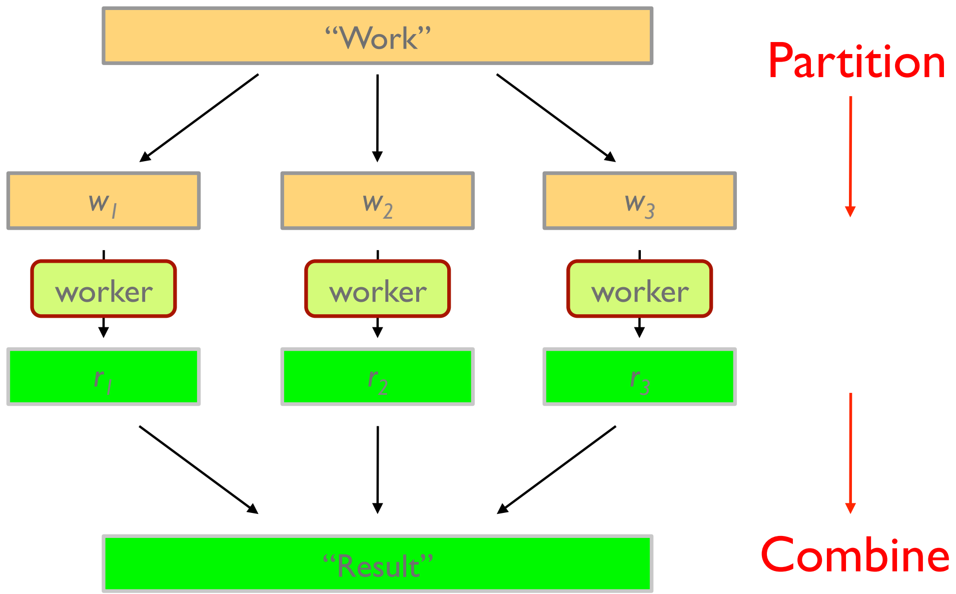
- ❑ In a single machine, the solution is trivial
  - final output: [(fog, 3), (winter, 2), (and, 4), …]

Basic example: Word count

- In a single machine, the solution is trivial
  - final output: [(fog, 3), (winter, 2), (and, 4), …]

- With multiple machines
  1. Use the solution for the single machine in each machine
     - intermediate output: [(fog, 3), (winter, 2), (and, 4), …]
  2. Join the results collected from the different machines and produce the final output
     - final output: [(tree, 8), (fog, 13), (cold, 3), (winter, 6), (and, 22), …]

Divide and Conquer



Parallelization Challenges

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers die?

What's the common theme of all of these problems?

Common Theme?

- Parallelization problems arise from:
  - Communication between workers (e.g., to exchange state)
  - Access to shared resources (e.g., data)
- Thus, we need a synchronization mechanism

Managing Multiple Workers

- Difficult because
  - We don't know the order in which workers run
  - We don't know when workers interrupt each other
  - We don't know when workers need to communicate partial results
  - We don't know the order in which workers access shared data
- Thus, we need:
  - Semaphores (lock, unlock)
  - Conditional variables (wait, notify, broadcast)
  - Barriers
- Still, lots of problems:
  - Deadlock, livelock, race conditions...
  - Dining philosophers, sleeping barbers, cigarette smokers...
- Moral of the story: be careful!

In summary

- Concurrency is difficult to reason about
- Concurrency is even more difficult to reason about
  - At the scale of datacenters and across datacenters
  - In the presence of failures
  - In terms of multiple interacting services
- Not to mention debugging…
- The reality:
  - Lots of one-off solutions, custom code
  - Write you own dedicated library, then program with it
  - Burden on the programmer to explicitly manage everything

Parallel computing: Concerns

- A parallel system needs to provide:
  - Data distribution
  - Computation distribution
  - Fault tolerance
  - Job scheduling

Parallel computing: Concerns

- A parallel system needs to provide:
  - Data distribution
  - Computation distribution
  - Fault tolerance
  - Job scheduling

- **The execution framework should hide these system-level details**
  - **Separate the *what* from the *how***

A final thought

Chris Stucchio

Home Blog Code Work

Don't use Hadoop - your data isn't that big

Posted: Mon, 16 Sep 2013

big data ,buzzwords ,hadoop

Tweet 1,892 Follow @stucchio 681

Like Send 879 people like this.

+1 +397 Recommend this on Google

"So, how much experience do you have with Big Data and Hadoop?" they asked me. I told them that I use Hadoop all the time, but rarely for jobs larger than a few TB. I'm basically a big data neophite - I know the concepts, I've written code, but never at scale.

The next question they asked me. "Could you use Hadoop to do a simple group by and sum?" Of course I could, and I just told them I needed to see an example of the file format.

They handed me a flash drive with all 600MB of their data on it (not a sample, everything). For reasons I can't understand, they were unhappy when my solution involved `pandas.read_csv` rather than Hadoop.

Hadoop is limiting. Hadoop allows you to run one general computation, which I'll illustrate in pseudocode: