CS10: The Beauty and Joy of Computing

Lecture #20
Distributed Computing

UC Berkeley EECS
Summer Instructor
Ben Chun

2012-07-25

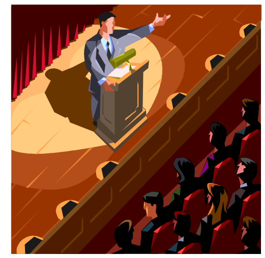
GOOGLE SCIENCE FAIR WINNER

Brittany Wenger wrote a neural net that analyzes diagnostic test data to detect breast cancer – and it performs better than commercial software.

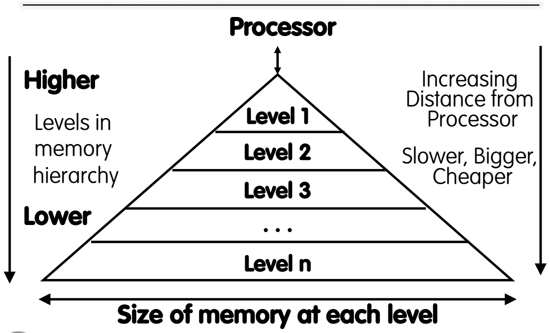
`http://bit.ly/NVusUb`

Lecture Overview

- **Basics**
  - Memory
  - Network
- **Distributed Computing**
  - Themes
  - Challenges
- **Solution! MapReduce**
  - How it works
  - Our implementation



Memory Hierarchy

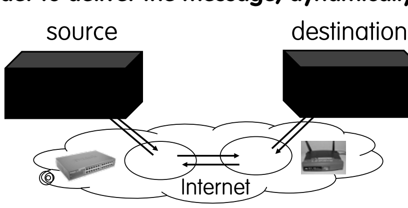


Memory Hierarchy Details

- **If level closer to Processor, it is:**
  - Smaller
  - Faster
  - More expensive
  - Subset of lower levels
    - …contains most recently used data
- **Lowest Level (usually disk) contains all available data (does it go beyond the disk?)**
- **Memory Hierarchy Abstraction presents the processor with the illusion of a very large & fast memory**

Networking Basics

- **source encodes and destination decodes content of the message**
- **switches and routers use the destination in order to deliver the message, dynamically**



Networking Facts and Benefits

- **Networks connect computers, sub-networks, and other networks.**
  - Networks connect computers all over the world (and in space!)
  - Computer networks...
    - support asynchronous and distributed communication
    - enable new forms of collaboration

en.wikipedia.org/wiki/FLOPS

Performance Needed for Big Problems

- **Performance terminology**
  - the FLOP: FLoating point OPeration
  - "flops" = # FLOP/second is the standard metric for computing power
- **Example: Global Climate Modeling**
  - Divide the world into a grid (e.g. 10 km spacing)
  - Solve fluid dynamics equations for each point & minute
    - Requires about 100 Flops per grid point per minute
  - Weather Prediction (7 days in 24 hours):
    - 56 Gflops
  - Climate Prediction (50 years in 30 days):
    - 4.8 Tflops
- **Perspective**
  - Intel Core i7 3970X Desktop Processor
    - ~120 Gflops
    - Climate Prediction would take ~3 years

www.epm.ornl.gov/chammp/chammp.html

What Can We Do? Use Many CPUs!

- **Supercomputing – like those listed in `top500.org`**
  - Multiple processors "all in one box / room" from one vendor that often communicate through shared memory
  - This is often where you find exotic architectures
- **Distributed computing**
  - Many separate computers (each with independent CPU, RAM, HD, NIC) that communicate through a network
    - Grids (heterogenous computers across Internet)
    - Clusters (mostly homogeneous computers all in one room)
      - Google uses commodity computers to exploit "knee in curve" price/performance sweet spot
  - It's about being able to solve "big" problems, not "small" problems faster
    - These problems can be data (mostly) or CPU intensive

en.wikipedia.org/wiki/Distributed_computing

Distributed Computing Themes

- **Let's network many disparate machines into one compute cluster**
- **These could all be the same (easier) or very different machines (harder)**
- **Common themes**
  - "Dispatcher" gives jobs & collects results
  - "Workers" (get, process, return) until done
- **Examples**
  - SETI@Home, BOINC, Render farms
  - Google clusters running MapReduce

Peer Instruction

1. **Writing & managing SETI@Home is relatively straightforward; just hand out & gather data**
2. **The majority of the world's computing power lives in supercomputer centers**

```
   12
a) FF
b) FT
c) TF
d) TT
```


Peer Instruction Answer

1. **The heterogeneity of the machines, handling machines that fail, falsify data. FALSE**
2. **Have you considered how many PCs + game devices exist? Not even close. FALSE**

1. **Writing & managing SETI@Home is relatively straightforward; just hand out & gather data**
2. **The majority of the world's computing power lives in supercomputer centers**

```
   12
a) FF
b) FT
c) TF
d) TT
```

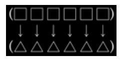

en.wikipedia.org/wiki/Embarrassingly_parallel

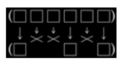
Distributed Computing Challenges

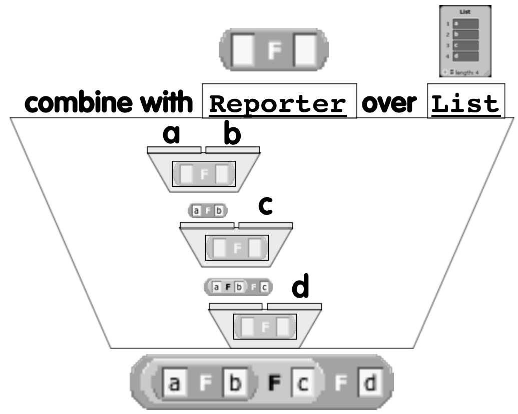
- **Communication is fundamental difficulty**
  - Distributing data, updating shared resource, communicating results, handling failures
  - Machines have separate memories, so need network
  - Introduces inefficiencies: overhead, waiting, etc.
- **Need to parallelize algorithms, data structures**
  - Must look at problems from parallel standpoint
  - Best for problems whose compute times >> overhead

Review

- Functions as Data
- Higher-Order Functions
- Useful HOFs (you can build your own!)
  - map `Reporter` over `List`
    - Report a new list, every element `E` of `List` becoming `Reporter(E)`
  - keep items such that `Predicate` from `List`
    - Report a new list, keeping only elements `E` of `List` if `Predicate(E)`
  - combine with `Reporter` over `List`
    - Combine all the elements of `List` with `Reporter(E)`
    - This is also known as "reduce"
- Acronym example
  - keep → map → combine



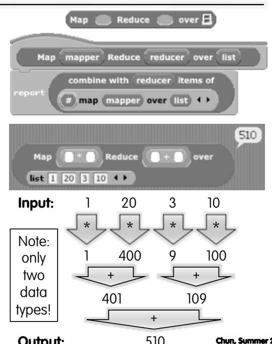




en.wikipedia.org/wiki/MapReduce

Google's MapReduce Simplified

- We told you "the beauty of pure functional programming is that it's easily parallelizable"
  - Do you see how you could parallelize this?
  - Reducer should be associative and commutative
- Imagine 10,000 machines ready to help you compute anything you could cast as a MapReduce problem!
  - This is the abstraction Google is famous for authoring
  - It hides LOTS of difficulty of writing parallel code!
  - The system takes care of load balancing, dead machines, etc.



MapReduce Advantages/Disadvantages

- Now it's easy to program for many CPUs
  - Communication management effectively gone
  - Fault tolerance, monitoring
    - machine failures, suddenly-slow machines, etc are handled
  - Can be much easier to design and program!
  - Can cascade several (many?) MapReduce tasks
- But … it might restrict solvable problems
  - Might be hard to express problem in MapReduce
  - Data parallelism is key
    - Need to be able to break up a problem by data chunks
  - Full MapReduce is closed-source (to Google) C++
    - Hadoop is open-source Java-based rewrite

Summary

- Systems and networks enable and foster computational problem solving
- MapReduce is a great distributed computing abstraction
  - It removes the onus of worrying about load balancing, failed machines, data distribution from the programmer of the problem
  - (and puts it on the authors of the MapReduce framework)