CSE 486/586 Distributed Systems
Data Analytics

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

- RPC enables programmers to call functions in remote processes.
- IDL (Interface Definition Language) allows programmers to define remote procedure calls.
- Stubs are used to make it appear that the call is local.
- Semantics
  - Cannot provide exactly once
  - At least once
  - At most once
  - Depends on the application requirements

Two Questions We'll Answer

- What is data analytics?
- What are the programming paradigms for it?

Example 1: Scientific Data

- CERN (European Organization for Nuclear Research) @ Geneva: Large Hadron Collider (LHC) Experiment
  - 300 GB of data per second
  - "15 petabytes (15 million gigabytes) of data annually – enough to fill more than 1.7 million dual-layer DVDs a year"

Example 2: Web Data

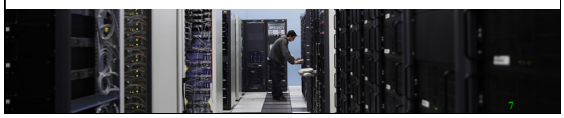
- Google
  - 20+ billion web pages
    - » ~20KB each = 400 TB
  - ~ 4 months to read the web
  - And growing…
    - » 1999 vs. 2009: ~ 100X
- Yahoo!
  - US Library of Congress every day (20TB/day)
  - 2 billion photos
  - 2 billion mail + messenger sent per day
  - And growing…

Data Analytics

- Computations on very large data sets
  - How large? TBs to PBs
  - Much time is spent on data moving/reading/writing
- Shift of focus
  - Used to be: computation (think supercomputers)
  - Now: data

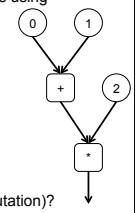
Popular Environment

- Environment for storing TBs ~ PBs of data
- Cluster of cheap commodity PCs
  - As we have been discussing in class…
  - 1000s of servers
  - Data stored as plain files on file systems
  - Data scattered over the servers
  - Failure is the norm
- How do you process all this data?



Turn to History

- Dataflow programming
  - Data sources and operations
  - Data items go through a series of transformations using operations.
  - Very popular concept
- Many examples
  - Even CPU designs back in 80's and 90's
  - SQL, data streaming, etc.
- Challenges
  - How to efficiently fetch data?
  - When and how to schedule different operations?
  - What if there's a failure (both for data and computation)?



Dataflow Programming

- This style of programming is now very popular with large clusters.
- Many examples
  - MapReduce, Pig, Hive, Dryad, Spark, etc.
- Two examples we'll look at
  - MapReduce and Pig

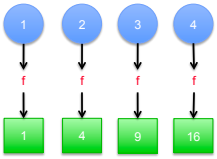
What is MapReduce?

- A system for processing large amounts of data
- Introduced by Google in 2004
- Inspired by map & reduce in Lisp
- OpenSource implementation: Hadoop by Yahoo!
- Used by many, many companies
  - A9.com, AOL, Facebook, The New York Times, Last.fm, Baidu.com, Joost, Veoh, etc.

Background: Map & Reduce in Lisp

- Sum of squares of a list (in Lisp)
- (**map** square '(1 2 3 4))
  - Output: (1 4 9 16)

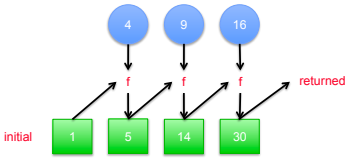
  [processes each record individually]



Background: Map & Reduce in Lisp

- Sum of squares of a list (in Lisp)
- (**reduce** + '(1 4 9 16))
  - (+ 16 (+ 9 (+ 4 1)))
  - Output: 30

  [processes set of all records in a batch]



Background: Map & Reduce in Lisp

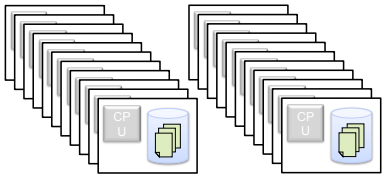
- Map
  - processes each record individually
- Reduce
  - processes (combines) set of all records in a batch

What Google People Have Noticed

- Keyword search
  - **Map** – Find a keyword in each web page individually, and if it is found, return the URL of the web page
  - **Reduce** – Combine all results (URLs) and return it
- Count of the # of occurrences of each word
  - **Map** – Count the # of occurrences in each web page individually, and return the list of <*word*, #>
  - **Reduce** – For each word, sum up (combine) the count
- Notice the similarities?

What Google People Have Noticed

- Lots of storage + compute cycles nearby
- Opportunity
  - Files are distributed already! (GFS)
  - A machine can processes its own web pages (map)



Google MapReduce

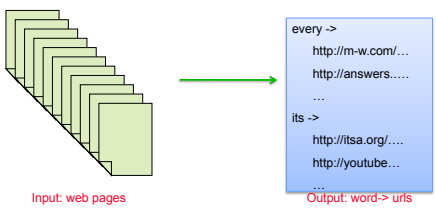
- Took the concept from Lisp, and applied to large-scale data-processing
- Takes two functions from a programmer (*map* and *reduce*), and performs three steps
- Map
  - Runs *map* for each file individually in parallel
- Shuffle
  - Collects the output from all *map* executions
  - Transforms the *map* output into the *reduce* input
  - Divides the *map* output into chunks
- Reduce
  - Runs *reduce* (using a *map* output chunk as the input) in parallel

Programmer's Point of View

- Programmer writes two functions – *map()* and *reduce()*
- The programming interface is fixed
  - map (in_key, in_value) ->
    list of (out_key, intermediate_value)
  - reduce (out_key, list of intermediate_value) ->
    (out_key, out_value)
- Caution: not exactly the same as Lisp

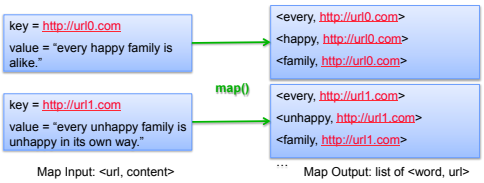
Inverted Indexing Example

- Word -> list of web pages containing the word



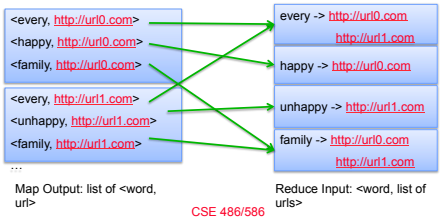
Map

- Interface
  - Input: <in_key, in_value> pair => <url, content>
  - Output: list of intermediate <key, value> pairs => list of <word, url>



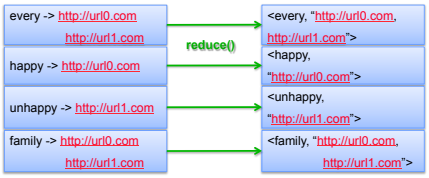
Shuffle

- MapReduce system
  - Collects outputs from all *map* executions
  - Groups all intermediate values by the same key

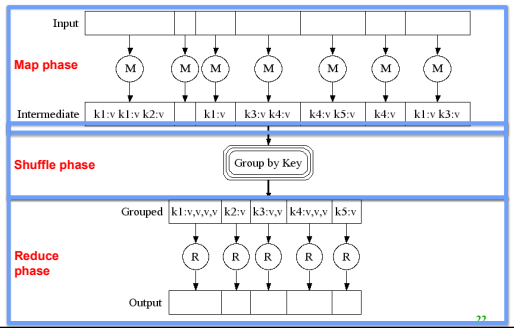


Reduce

- Interface
  - Input: <out_key, list of intermediate_value>
  - Output: <out_key, out_value>



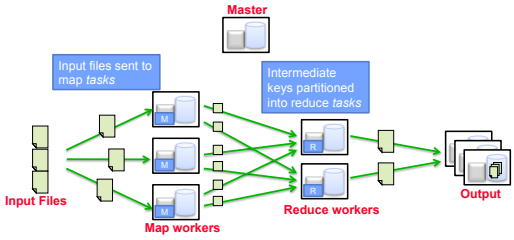
Execution Overview



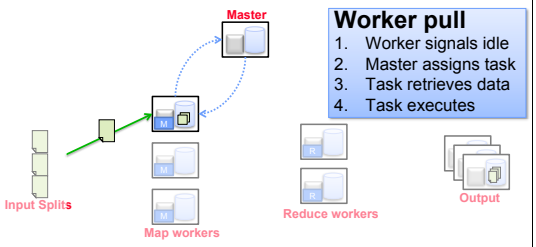
Implementing MapReduce

- Externally for user
  - Write a map function, and a reduce function
  - Submit a job; wait for result
  - No need to know anything about the environment (Google: 4000 servers + 48000 disks, many failures)
- Internally for MapReduce system designer
  - Run map in parallel
  - Shuffle: combine map results to produce reduce input
  - Run reduce in parallel
  - Deal with failures

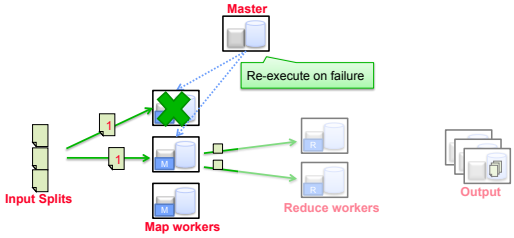
Execution Overview



Task Assignment



Fault-tolerance: Re-execution



Machines Share Roles

- So far, logical view of cluster
- In reality
  - Each cluster machine stores data
  - And runs MapReduce workers
- Lots of storage + compute cycles nearby

Problems of MapReduce

- Any you can think of?
  - There's only two functions you can work with (not expressive enough sometimes.)
  - Functional-style (a barrier for some people)
- Turing completeness (or computationally universal)
  - If it can simulate a single-taped Turing machine.
  - Most general languages (C/C++, Java, Lisp, etc.) are.
  - SQL is.
  - MapReduce is not.

Pig

- Why Pig?
  - MapReduce has limitations: only two functions
  - Many tasks require more than one MapReduce
  - Functional thinking: barrier for some
- Pig
  - Defines a set of high-level simple "commands"
  - Compiles the commands and generates multiple MapReduce jobs
  - Runs them in parallel

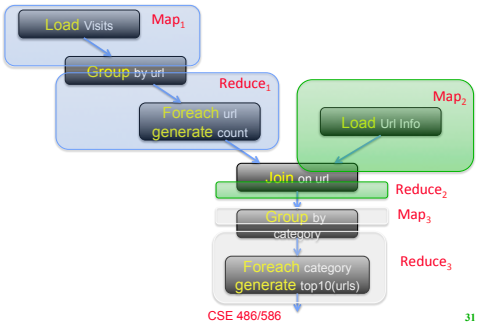
Pig Example

```
load '/data/visits';
group visits by url;
foreach gVisits generate url, count(visits);

load '/data/urlInfo';
join visitCounts by url, urlInfo by url;

group visitCounts by category;
foreach gCategories generate top(visitCounts,10);
```

Pig Example



Summary

- Data analytics shifts the focus from computation to data.
- Many programming paradigms are emerging.
  - MapReduce
  - Pig
  - Many others

More Details

- Papers
  - J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI 2004
  - C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: A Not-So-Foreign Language For Data Processing," SIGMOD 2008
- URLs
  - http://hadoop.apache.org/core/
  - http://wiki.apache.org/hadoop/
  - http://hadoop.apache.org/pig/
  - http://wiki.apache.org/pig/
- Slides
  - http://labs.google.com/papers/mapreduce-osdi04-slides/index.html
  - http://www.systems.ethz.ch/education/past-courses/hs08/map-reduce/slides/intro.pdf
  - http://www.cs.uiuc.edu/class/sp09/cs525/L4tmp.B.ppt
  - http://infolab.stanford.edu/~usriv/talks/sigmod08-pig-latin.ppt

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).