CIS 560 – Database System Concepts

Lecture 30

NoSQL

November 15, 2013

Credits for slides: Wisdom, Alberton, Pokorný, Hoekstra.

Reminders

- Assignment 9 (query optimization) due 11/15
- Exam 2 (assignments 6-9) – 11/20
  - Assignments 6-9
  - Lecture notes 18-28
  - Textbook 17.1-17.4, 18.1-18.3, 18.8, 14.1-14.2, 15.1-15.6, 16
- Project - DB implementation and queries due 11/22
- Quiz from NoSQL lectures – 12/06

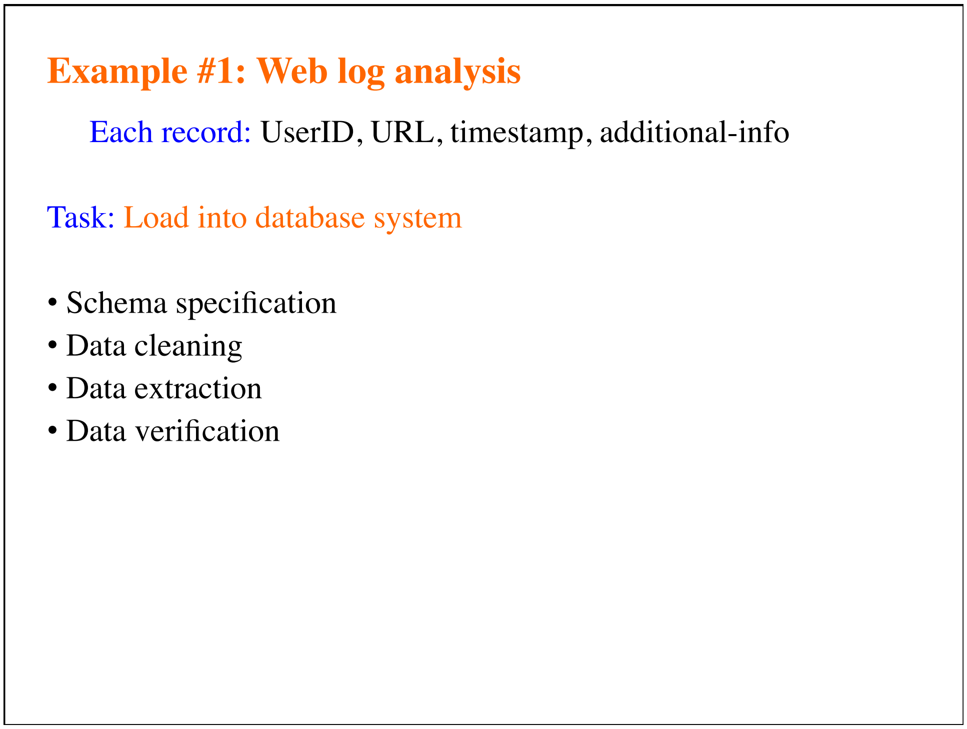
Where we are

- Last: NoSQL introduction/motivation, CAP theorem, discussed why/when to use NoSQL using sample tasks.
- Today: MapReduce framework
  - MapReduce: Simplified Data Processing on Large Clusters. Jeffrey Dean and Sanjay Ghemawat. OSDI'04.
  - Data-Intensive Text Processing with MapReduce, Jimmy Lin and Chris Dyer, 2010.

    http://lintool.github.io/MapReduceAlgorithms/
- Next: Hive, Pig Latin
  - Hive – A Petabyte Scale Data Warehouse Using Hadoop
  - Pig Latin: A Not-So-Foreign Language for Data Processing

Recognition over past decade or so:

> Not every data management/analysis problem is best solved using a traditional relational DBMS

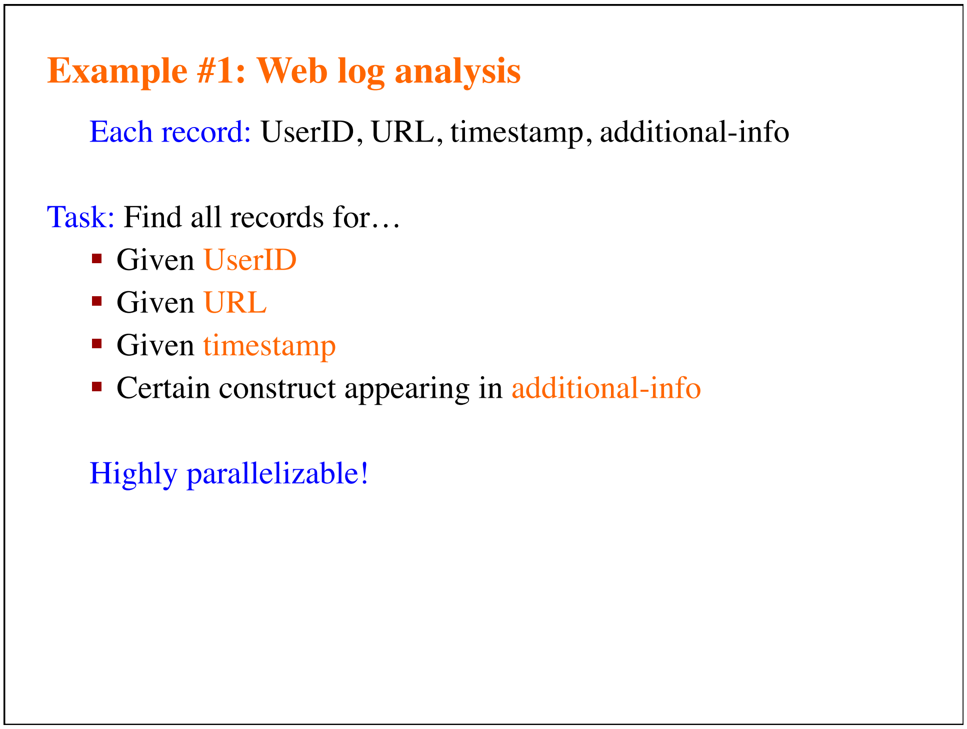


Example #1: Web log analysis

Each record: UserID, URL, timestamp, additional-info

Task: Load into database system

- Schema specification
- Data cleaning
- Data extraction
- Data verification



Example #1: Web log analysis

Each record: UserID, URL, timestamp, additional-info

Task: Find all records for…

- Given UserID
- Given URL
- Given timestamp
- Certain construct appearing in additional-info

Highly parallelizable!

Example #1: Web log analysis

Each record: UserID, URL, timestamp, additional-info

Task: Find all pairs of UserIDs accessing same URL

Example #1: Web log analysis

Each record: UserID, URL, timestamp, additional-info
Separate records: UserID, name, age, gender, …

Task: Find average age of user accessing given URL

Example #2: Social-network graph

Each record: UserID_1, UserID_2
Separate records: UserID, name, age, gender, …

Task: Find all friends of a given user

Example #2: Social-network graph

Each record: UserID_1, UserID_2
Separate records: UserID, name, age, gender, …

Task: Find all friends of friends of a given user

Example #2: Social-network graph

Each record: UserID_1, UserID_2
Separate records: UserID, name, age, gender, …

Task: Find all women friends of men friends of a given user

Example #2: Social-network graph

Each record: UserID_1, UserID_2
Separate records: UserID, name, age, gender, …

Task: Find all friends of friends of friends of … friends of given user

Example #3: Wikipedia pages

Large collection of documents

Combination of structured and unstructured data

Task: Retrieve introductory paragraph of all pages about U.S. presidents before 1900

NoSQL Systems

Alternative to traditional relational DBMS

+ Flexible schema
+ Quicker/cheaper to set up
+ Massive scalability
+ Relaxed consistency → higher performance & availability

– No declarative query language → more programming

– Relaxed consistency → fewer guarantees

NoSQL – Two Main Incarnations

- NoSQL framework - MapReduce
  - Originally from Google, open source Hadoop
    - No data model, data stored in files
    - User provides specific functions
    - System provides data processing "glue", fault-tolerance, scalability
- NoSQL ``databases"

Four (emerging) NoSQL Categories

- Key-value stores
  - Based on Distributed Hashtables (DHTs)/ Amazon's Dynamo paper *
  - Data model: (global) collection of K-V pairs
  - Example: Voldemort
- Column Families
  - BigTable clones **
  - Data model: big table, column families
  - Example: HBase, Cassandra, Hypertable

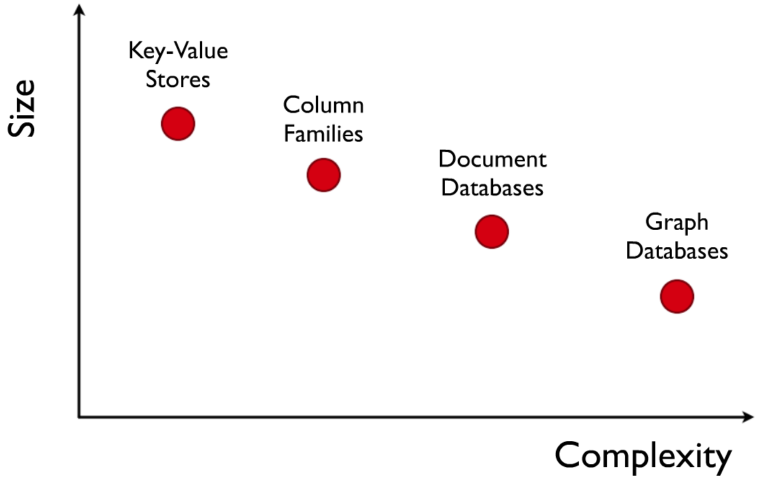
*G DeCandia et al, Dynamo: Amazon's Highly Available Key-value Store, SOSP 07

** F Chang et al, Bigtable: A Distributed Storage System for Structured Data, OSDI 06

Four (emerging) NoSQL Categories

- Document databases
  - Inspired by Lotus Notes
  - Data model: collections of K-V Collections
  - Example: CouchDB, MongoDB
- Graph databases
  - Inspired by Euler & graph theory
  - Data model: nodes, relations, K-V on both
  - Example: AllegroGraph, VertexDB, Neo4j

Focus of Different Data Models



Slide from neo technology, “A NoSQL Overview and the Benefits of Graph Databases"

Advantages of NoSQL (Why NoSQL)

- Support for a specific problem / situation
- No need to think in terms of relations but in terms given in a situation (e.g. documents, nodes, ...)
- In most cases freely available
- In most cases open source
- Fast averages

w/ 50GB	Writes	Reads
MySQL	~300 ms	~350 ms
Cassandra	0.12 ms	15 ms

- However
  - Data quality could get lost if the possibilities of NoSQL databases are used too extensively or not in the right domain.

NoSQL – Two Main Incarnations

- NoSQL framework - MapReduce
  - Originally from Google, open source Hadoop
    - No data model, data stored in files
    - User provides specific functions
    - System provides data processing “glue”, fault-tolerance, scalability
- NoSQL ``databases”

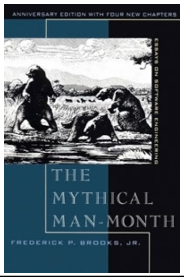
What's the point of MapReduce?

- It's all about the right level of abstraction
  - The von Neumann architecture has served us well, but is no longer appropriate for the multi-core/cluster environment
- Hide system-level details from the developers
  - No more race conditions, lock contention, etc.
- Separating the *what* from *how*
  - Developer specifies the computation that needs to be performed
  - Execution framework ("runtime") handles actual execution

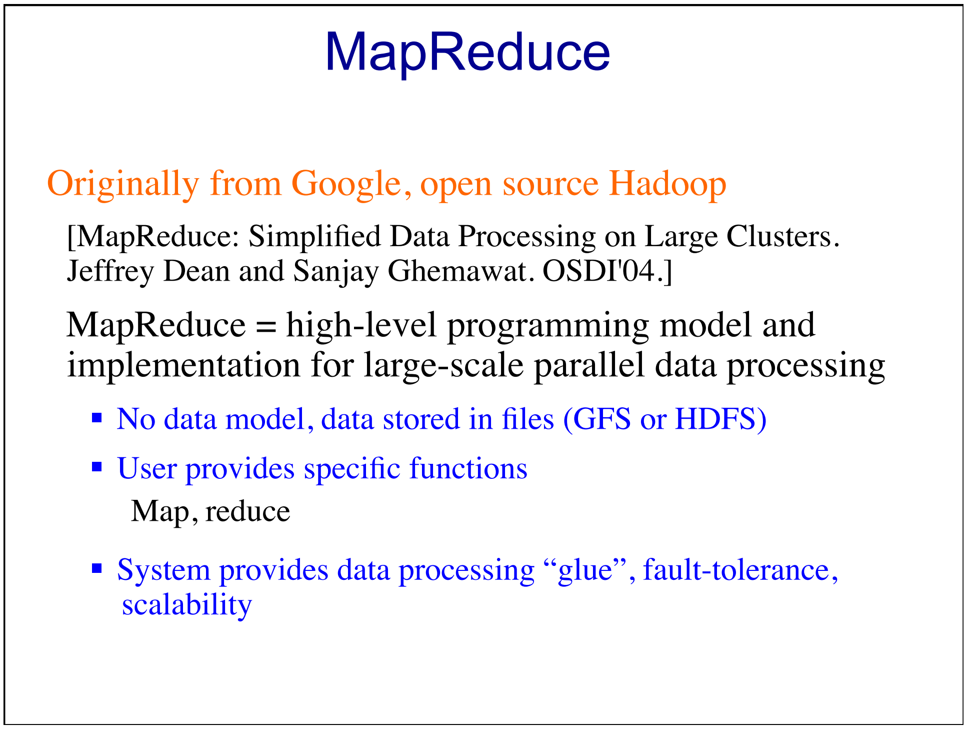
The datacenter *is* the computer!

"Big Ideas"

- Scale "out", not "up"
  - Limits of symmetric multi-processing (SMP) machines and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE
MYTHICAL
MAN-MONTH
FREDERICK P. BROOKS, JR.

"adding manpower to a late software project makes it later"



MapReduce

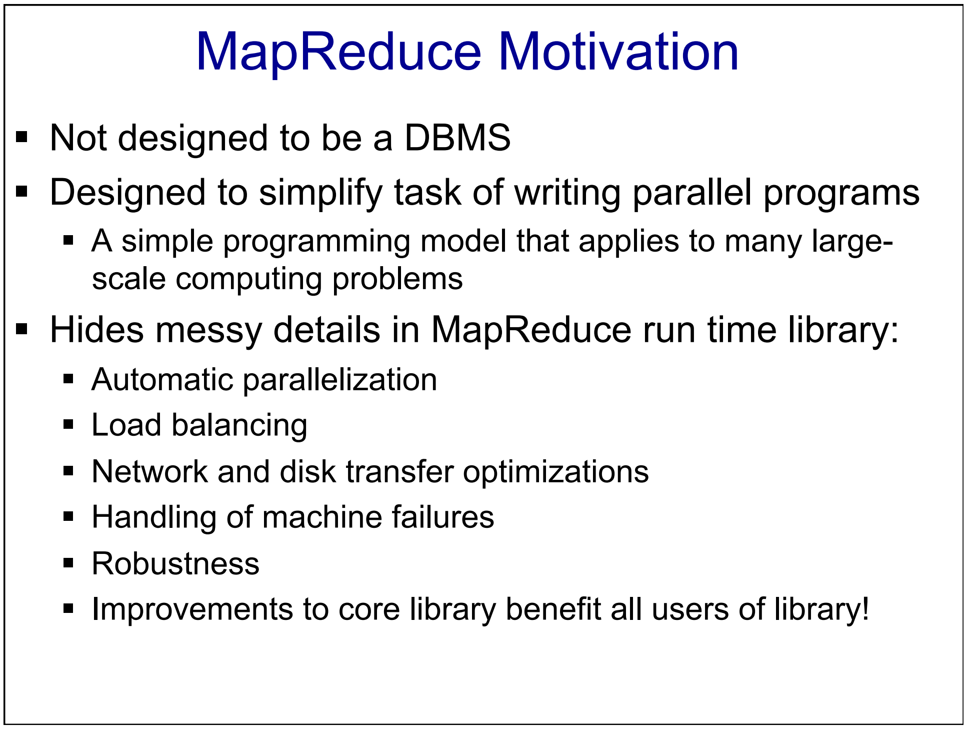
Originally from Google, open source Hadoop

[MapReduce: Simplified Data Processing on Large Clusters. Jeffrey Dean and Sanjay Ghemawat. OSDI'04.]

MapReduce = high-level programming model and implementation for large-scale parallel data processing

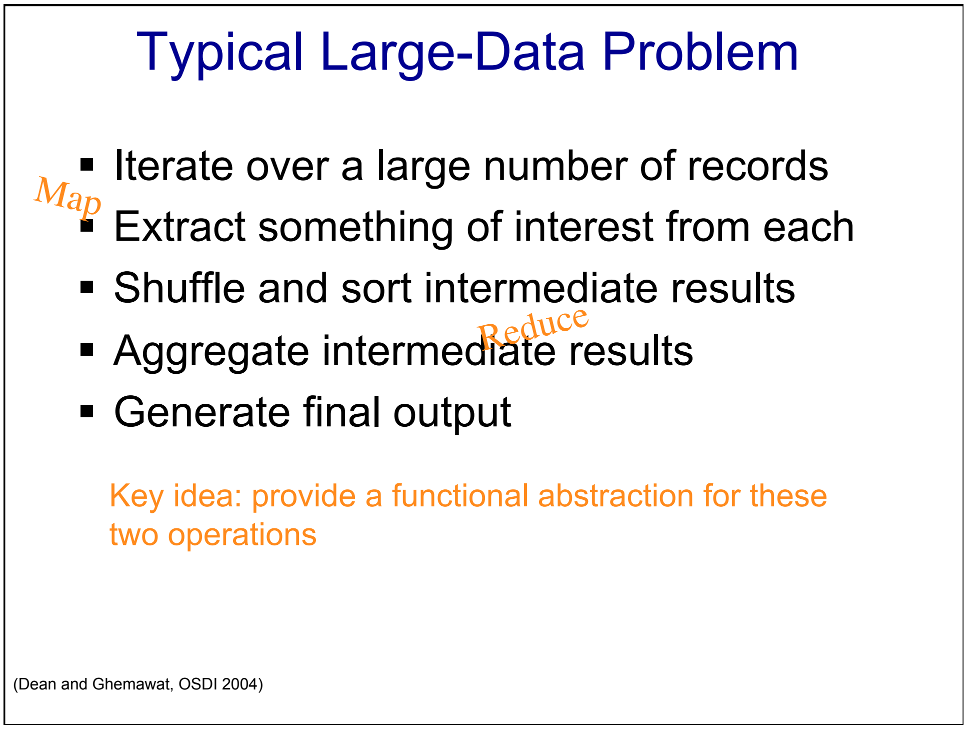
- No data model, data stored in files (GFS or HDFS)
- User provides specific functions

  Map, reduce
- System provides data processing "glue", fault-tolerance, scalability



MapReduce Motivation

- Not designed to be a DBMS
- Designed to simplify task of writing parallel programs
  - A simple programming model that applies to many large-scale computing problems
- Hides messy details in MapReduce run time library:
  - Automatic parallelization
  - Load balancing
  - Network and disk transfer optimizations
  - Handling of machine failures
  - Robustness
  - Improvements to core library benefit all users of library!



Typical Large-Data Problem

- Iterate over a large number of records

Map

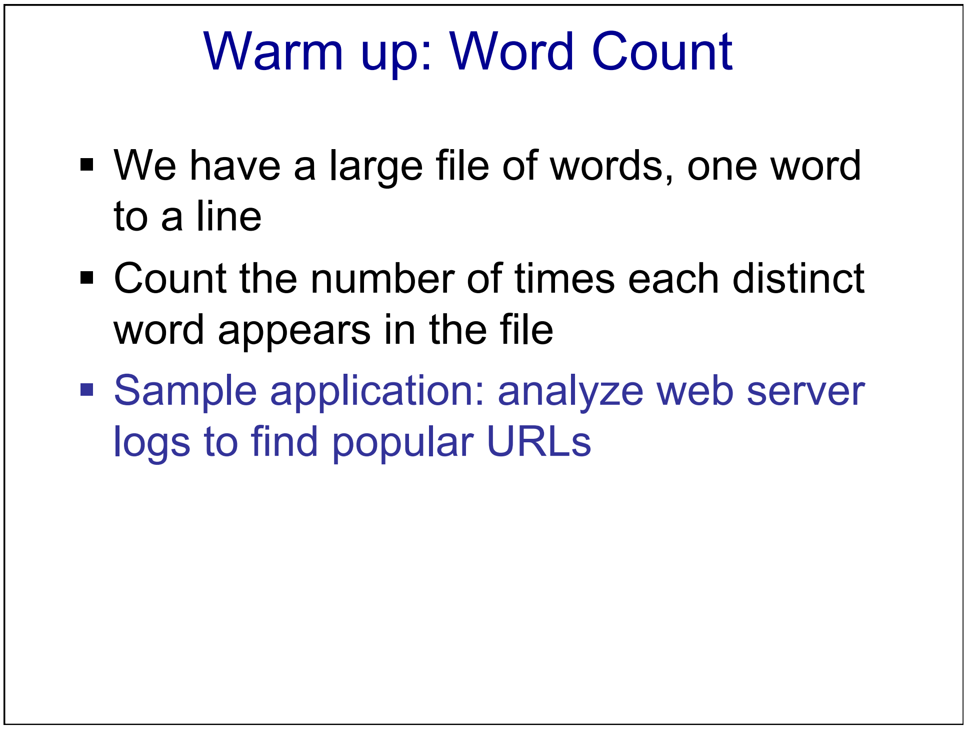
- Extract something of interest from each
- Shuffle and sort intermediate results

Reduce

- Aggregate intermediate results
- Generate final output

Key idea: provide a functional abstraction for these two operations

(Dean and Ghemawat, OSDI 2004)



Warm up: Word Count

- We have a large file of words, one word to a line
- Count the number of times each distinct word appears in the file
- Sample application: analyze web server logs to find popular URLs