CS 5614: (Big) Data Management Systems

B. Aditya Prakash

Lecture #1: Introduction
$600 to buy a disk drive that can store all of the world’s music

5 billion mobile phones in use in 2010

30 billion pieces of content shared on Facebook every month

40% projected growth in global data generated per year vs. 5% growth in global IT spending

$5 million vs. $400 Price of the fastest supercomputer in 1975 and an iPhone 4 with equal performance

235 terabytes data collected by the US Library of Congress by April 2011

15 out of 17 sectors in the United States have more data stored per company than the US Library of Congress

Prakash 2014
Data contains value and knowledge

Prakash 2014

CS 5614: (Big) Data Management Systems
Data and Business

Source: A. Machhanavajjhala
Data and Science

Detecting influenza epidemics using search engine query data

Red: official numbers from Center for Disease Control and Prevention; weekly
Black: based on Google search logs; daily (potentially instantaneously)

http://www.nature.com/nature/journal/v457/n7232/full/nature07634.html
Data and Government

http://www.washingtonpost.com/opinions/obama-the-big-data-president/2013/06/14/1d71fe2e-d391-11e2-b05f-3ea3f0e7bb5a_story.html

http://www.whitehouse.gov/blog/Democratizing-Data


Source: A. Machhanavajjhala
Data and Culture

• Word frequencies in English-language books in Google’s database


Source: A. Machhanavajjhala
Data and ____ your favorite subject

Sports

Journalism

Every year in America, 6,000 killers get away with murder. Why do police fail to solve so many homicides, even with more advanced investigative techniques? Can we stop these murderers from killing again and bring justice to their victims?
Good news: Demand for Data Mining

Demand for deep analytical talent in the United States could be 50 to 60 percent greater than its projected supply by 2018.

Supply and demand of deep analytical talent by 2018:

- Thousand people:
  - 2008 employment: 150
  - Graduates with deep analytical talent: 180
  - Others: 30
  - 2018 supply: 300
  - Talent gap: 140–190
  - 2018 projected demand: 440–490

50–60% gap relative to 2018 supply.

Other supply drivers include attrition (-), immigration (+), and reemploying previously unemployed deep analytical talent (+).

How to extract value from data?

- Manipulate Data
  - CS, Domain expertise

- Analyze Data
  - Math, CS, Stat...

- Communicate your results
  - CS, Domain Expertise
Communication is important!

“The British government spends £13 billion a year on universities.”

– So?
– Try instead
  http://wheredoesmymoneygo.org/bubbletree-map.html#/~/total/education/university

“On average, 1 in every 15 Europeans is totally illiterate.”

– True
– But about 1 in every 14 is under 7 years old!
  http://datajournalismhandbook.org/1.0/en/understanding_data_0.html
What is Data Mining?

- Given lots of data

- Discover patterns and models that are:
  - **Valid**: hold on new data with some certainty
  - **Useful**: should be possible to act on the item
  - **Unexpected**: non-obvious to the system
  - **Understandable**: humans should be able to interpret the pattern
Data Mining Tasks

- **Descriptive methods**
  - Find human-interpretable patterns that describe the data
    - **Example:** Clustering

- **Predictive methods**
  - Use some variables to predict unknown or future values of other variables
    - **Example:** Recommender systems
Course Information

- **Instructor**
  B. Aditya Prakash, Torgersen Hall 3160 F, badityap@cs.vt.edu
  - Office Hours: TBD
  - Include string **CS 5614** in subject

- **Teaching Assistant**
  Vanessa Cadeno, vcedeno@vt.edu
  - Office Hours: TBD

- **Class Meeting Time**
  Mondays, Wednesdays, 2:30-3:45pm, McBryde Hall 134

- **Syllabus:** Relational Database Systems, Big data Technologies (MR and new software stack), Streams, Recommendation Systems, Large Scale Machine Learning, and Graph Mining
Course Information

- Keeping in Touch
  Course web site

  http://www.cs.vt.edu/~badityap/classes/cs5614-Fall14/

  updated regularly through the semester

  – Piazza link on the website
Textbook

- **Required**
  Jure Leskovec, Anand Rajaraman and Jeffery Ullman: Mining Massive Datasets (2\textsuperscript{nd}) Cambridge University Press. 2010

Web page for the book (with FREE PDF!)
www.mmds.org
Textbook

- Recommended (for database internals)

Raghu Ramakrishnan and Johannes Gehrke
Pre-reqs

(A) Should enjoy the course 😊

(B) Background in
   1. Algorithms
   2. Probability and Stats
   3. Undergraduate level Databases
   4. Linear Algebra (helps)
   5. Graph theory (helps)

(C) Graduate-level Programming Skills (i.e. ability to use unfamiliar software, picking up new languages, comfortable with at least one of Python/C/C++/Ruby/Java etc. (Matlab/R a plus))
Course Grading

- Details coming soon (next lecture)

- Broadly
  - Some hws
  - No midterm
  - Take-home Final
  - Project
  - Class participation
Course Project

- 2, or 3 (max) persons per project.
- Major work for this class.
- Pick your own topic
  - You have to justify why the topic is interesting, and relevant to the course, and of suitable difficulty
- Harder way:
  - Joint projects with other courses are also negotiable. In that case, you will need the approval of the instructor, and you also need to clarify exactly what steps will be done for our course, as well as for the other course.
  - Projects related to your dissertation/master-project are also possible, as long as there is no 'double-dipping', i.e., you clearly specify what the project will do, in addition to what you were planning to do for your thesis anyway.
- Ask me if you need help and ideas (I may release a list of suitable topics later)
Course Project

- Proposal
- Milestone
- Final Report
- Poster Presentation (or in-class presentation TBD)
WARM-UP AND BASICS
Relational Databases: What we will cover (next 1 month)

- Implementation
  - What is under-the-hood of a DB like Oracal/MySQL?

- Design
  - How do you model your data and structure your information in a database?

- Programming
  - How do you use the capabilities of a DBMS?

- CS 4604 achieves a balance between
  - a firm theoretical foundation to designing moderate-sized databases
  - creating, querying, and implementing realistic databases and connecting them to applications
CS4604: Course Outline

- **Weeks 1–4: Query/Manipulation Languages and Data Modeling**
  - Relational Algebra
  - Data definition
  - Programming with SQL
  - Entity-Relationship (E/R) approach
  - Specifying Constraints
  - Good E/R design

- **Weeks 5–8: Indexes, Processing and Optimization**
  - Storing
  - Hashing/Sorting
  - Query Optimization
  - NoSQL and Hadoop

- **Week 9-10: Relational Design**
  - Functional Dependencies
  - Normalization to avoid redundancy

- **Week 11-12: Concurrency Control**
  - Transactions
  - Logging and Recovery

- **Week 13–14: Students’ choice**
  - Practice Problems
  - XML
  - Data mining and warehousing

*We will go over all of this quickly!* ☺
What is the goal of a DBMS?

- Electronic record-keeping
  - Fast and convenient access to information

- DBMS == database management system
  - `Relational’ in this class
  - data + set of instructions to access/manipulate data
What is a DBMS?

Features of a DBMS

- Support massive amounts of data
- Persistent storage
- Efficient and convenient access
- Secure, concurrent, and atomic access

Examples?

- Search engines, banking systems, airline reservations, corporate records, payrolls, sales inventories.
- New applications: Wikis, social/biological/multimedia/scientific/geographic data, heterogeneous data.
Features of a DBMS

• Support **massive** amounts of data
  – Giga/tera/petabytes
  – Far too big for main memory

• **Persistent** storage
  – Programs update, query, manipulate data.
  – Data continues to live long after program finishes.

• **Efficient** and **convenient** access
  – Efficient: do not search entire database to answer a query.
  – Convenient: allow users to query the data as easily as possible.

• **Secure, concurrent, and atomic** access
  – Allow multiple users to access database simultaneously.
  – Allow a user access to only to authorized data.
  – Provide some guarantee of reliability against system failures.
Example Scenario

- Students, taking classes, obtaining grades
  - Find my GPA
  - <and other ad-hoc queries>
Obvious solution 1: Folders

- Advantages?
  - Cheap; Easy-to-use

- Disadvantages?
  - No ad-hoc queries
  - No sharing
  - Large Physical foot-print
Obvious Solution++

- Flat files and C (C++, Java...) programs
  - E.g. one (or more) UNIX/DOS files, with student records and their courses
Obvious Solution++

- Layout for student records?
  - CSV (‘comma-separated-values’)
  
  Hermione Grainger,123,Potions,A
  Draco Malfoy,111,Potions,B
  Harry Potter,234,Potions,A
  Ron Weasley,345,Potions,C
Obvious Solution++

- Layout for student records?
  - Other possibilities like
    Hermione Grainger, 123
    Draco Malfoy, 111
    Harry Potter, 234
    Ron Weasley, 345
    123, Potions, A
    111, Potions, B
    234, Potions, A
    345, Potions, C
Problems?

- inconvenient access to data (need ‘C++’ expertise, plus knowledge of file-layout)
  - data isolation
- data redundancy (and inconsistencies)
- integrity problems
- atomicity problems
- concurrent-access problems
- security problems
- ........
Problems-Why?

- Two main reasons:
  - file-layout description is buried within the C programs and
  - there is no support for transactions (concurrency and recovery)

DBMSs handle exactly these two problems
Example Scenario

- RDBMS = “Relational” DBMS
- The relational model uses relations or tables to structure data
- ClassList relation:

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hermione Grainger</td>
<td>Potions</td>
<td>A</td>
</tr>
<tr>
<td>Draco Malfoy</td>
<td>Potions</td>
<td>B</td>
</tr>
<tr>
<td>Harry Potter</td>
<td>Potions</td>
<td>A</td>
</tr>
<tr>
<td>Ron Weasley</td>
<td>Potions</td>
<td>C</td>
</tr>
</tbody>
</table>

- Relation separates the logical view (externals) from the physical view (internals)
- Simple query languages (SQL) for accessing/modifying data
  - Find all students whose grades are better than B.
  - SELECT Student FROM ClassList WHERE Grade > “B”
DBMS Architecture

- Schema Modifications
- Queries
- Modifications

- "Query" Processor
- Storage Manager
- Transaction Manager

Data Metadata

Prakash 2014
CS 5614: (Big) Data Management Systems
Transaction Processing

- One or more database operations are grouped into a “transaction”
- Transactions should meet the “ACID test”
  - **Atomicity**: All-or-nothing execution of transactions.
  - **Consistency**: Databases have consistency rules (e.g. what data is valid). A transaction should NOT violate the database’s consistency. If it does, it needs to be *rolled back*.
  - **Isolation**: Each transaction must appear to be executed as if no other transaction is executing at the same time.
  - **Durability**: Any change a transaction makes to the database should persist and not be lost.
Disadvantages over (flat) files?
Disadvantages over (flat) files

- Price
- additional expertise (SQL/DBA)

(hence: over-kill for small, single-user data sets
But: mobile phones (eg., android) use sqlite)
A Brief History of DBMS

- The earliest databases (1960s) evolved from file systems
  - File systems
    - Allow storage of large amounts of data over a long period of time
    - File systems do not support:
      - Efficient access of data items whose location in a particular file is not known
      - Logical structure of data is limited to creation of directory structures
      - Concurrent access: Multiple users modifying a single file generate non-uniform results
    - Navigational and hierarchical
    - User programmed the queries by walking from node to node in the DBMS.

- Relational DBMS (1970s to now)
  - View database in terms of relations or tables
  - High-level query and definition languages such as SQL
  - Allow user to specify what (s)he wants, not how to get what (s)he wants

- Object-oriented DBMS (1980s)
  - Inspired by object-oriented languages
  - Object-relational DBMS
The DBMS Industry

- A DBMS is a software system.

- Major DBMS vendors: Oracle, Microsoft, IBM, Sybase

- Free/Open-source DBMS: MySQL, PostgreSQL, Firebird.
  - Used by companies such as Google, Yahoo, Lycos, BASF....

- All are “relational” (or “object-relational”) DBMS.

- A multi-billion dollar industry
Fundamental concepts

- 3-level architecture
- logical data independence
- physical data independence
3-level architecture

- view level
- logical level
- physical level
3-level architecture

- view level
- logical level: eg., tables
  - STUDENT(ssn, name)
  - TAKES (ssn, cid, grade)
- physical level:
  - how are these tables stored, how many bytes / attribute etc
3-level architecture

- view level, eg:
  - v1: select ssn from student
  - v2: select ssn, c-id from takes

- logical level

- physical level
3-level architecture

- hence, **physical** and **logical** data independence:
  - logical D.I.:
    - ???
  - physical D.I.:
    - ???
3-level architecture

- > hence, **physical** and **logical** data independence:

- logical D.I.:
  - can add (drop) column; add/drop table

- physical D.I.:
  - can add index; change record order
Database users

- ‘naive’ users
- casual users
- application programmers
- [ DBA (Data base administrator)]
Casual users

**SQL Query:**

```
select *
from student
```

**DBMS Diagram:**

- **Data**
- **Meta-data** (catalog)

- **Query Flow:**
  - `select * from student`

- **DBMS Processes:**
  - Query processing
  - Data retrieval
``Naive’’ users

Pictorially:

app. (eg.,
report generator)

and meta-data =
catalog
App. programmers

- those who write the applications (like the ‘report generator’)
DB Administrator (DBA)

- Duties?
DB Administrator (DBA)

- schema definition (‘logical’ level)
- physical schema (storage structure, access methods)
- schemas modifications
- granting authorizations
- integrity constraint specification
Overall system architecture

- [Users]
- DBMS
  - query processor
  - storage manager
  - transaction manager
- [Files]

Diagram:
- Schema Modifications
- Queries
- Modifications
- “Query” Processor
- Storage Manager
- Transaction Manager
- Data
- Metadata
Overall system architecture

- query processor
  - DML compiler
  - embedded DML pre-compiler
  - DDL interpreter
  - Query evaluation engine
Overall system architecture (cont’d)

- storage manager
  - authorization and integrity manager
  - transaction manager
  - buffer manager
  - file manager
Overall system architecture (cont’d)

- Files
  - data files
  - data dictionary = catalog (= meta-data)
  - indices
  - statistical data
Some examples:

- DBA doing a DDL (data definition language) operation, eg.,
  create table student ...
naive  app. pgmr  casual  DBA  users

emb. DML  DML proc.  DDL int.

app. pgm(o)  query eval.

trans. mgr  buff. mgr  file mgr

query proc.

storage mgr.

data  meta-data
Some examples:

- casual user, asking for an update, eg.:
  
  update student
  
  set name to ‘smith’
  
  where ssn = ‘345’
naive  app. pgmr  casual  DBA  users

emb. DML  DML proc.  DDL int.

app. pgm(o)  query eval.

query proc.

trans. mgr  buff. mgr  file mgr

storage mgr.

data

meta-data
**DDL int.**

**DML proc.**

**query eval.**

**emb. DML**

**app. pgm(o)**

**trans. mgr**

**buff. mgr**

**file mgr**

**query proc.**

**storage mgr.**

**data**

**meta-data**

**naive**

**app. pgmr**

**casual**

**DBA**

**users**
Some examples:

- app. programmer, creating a report, eg
  
  ```c
  main(){
    ....
    exec sql "select * from student"
    ...
  }
  ```
Some examples:

- ‘naive’ user, running the previous app.
Conclusions

- (relational) DBMSs: electronic record keepers
- customize them with `create table` commands
- ask SQL queries to retrieve info
Conclusions contd

main advantages over (flat) files & scripts:

- **logical + physical data independence** (ie., flexibility of adding new attributes, new tables and indices)

- **concurrency control and recovery**