Databases and Data-Intensive Systems

Computer Science Day
May 23rd 2014
Aarhus University
Staff

- Ira Assent, associate professor
- Christian S. Jensen, professor (part-time)
- Bin Yang, postdoc
- Chenjuan Guo, postdoc
- Sean Chester, postdoc
- Manohar Kaul, Ph.D. student
- Yu Ma, Ph.D. student
- Barbora Micenková, Ph.D. student
- Michael Lind Mortensen, Ph.D. student
- Laura Radaelli, Ph.D. student
- Anders Skovsgaard, Ph.D. student
- Qiang Qu, Ph.D. student
- Kenneth Sejdenfaden Bøgh, Ph.D. student

Contact info at http://cs.au.dk/research/areas/data-intensive-systems/
Data-Intensive Systems

- Focus on solving “data intensive” tasks
- Query processing for databases
- Mining “big data” for pattern discovery
- Formalize, implement and test new algorithms
- Inventing new and improving existing types of queries
Example: Decision making

WallViz:

Improving decision making from massive data collections using wall-sized, highly interactive visualizations

- Decisions are increasingly informed by analysis of massive data collections
- Manual exploration hard; automatic analysis infeasible
- Results in information overload, poor decisions

- Visualization helps deal with massive data collections

Human-Centered Computing (DIKU, Copenhagen)

Case partners in health care, finance, sustainability

Evaluation, field studies

Data management (AU)
Entry point generation with the Skyline

• Given a set of records on n attributes.
  - e.g., hotels, with distance to the beach and price.

• The skyline contains those records for which:
  - The top choice of every user with monotonic preference functions* is in the skyline.
  - Every record in the skyline is the top choice for some user with monotonic preference functions.

* e.g., a hotel 100m from the beach is preferred to a hotel with the same characteristics but 1km from the beach.
Challenges of skylines

- Expensive to compute → Need solutions for scaling
- We are researching approaches to parallelization
  - First to do this on GPUs
  - Currently working on parallelization on CPUs
  - How to approximate

- Large input → large results
- Selecting a subset of the result
  - Representative skyline
  - Introducing ranking
  - Adding constraints
Data mining

- Automatically identify interesting patterns in data (as opposed to queries that specify interesting results)

- Outlier detection:
  - Find “anomalies” that differ from rest of data
    - E.g. credit card fraud detection
  - eData project: outlier detection for eScience

- Subspace clustering:
  - detecting groups in high dimensional data

- Predicting customer shopping
  - E.g. Target predicted pregnancy before family knew
Courses

• Specialization Column (with progression)
  - Q1+Q2: Machine Learning - joint course with other research groups
  - Q3: Selfstudy on either data mining or query processing techniques
  - Q4: Multidimensional databases (2014, 2016, etc)
  - Q4: Indexing of disk based data (2015, 2017, etc)

• Related optional courses
  - Q1: Algorithms in bioinformatics – sequences
  - Q3: Algorithm engineering

• PREP: Practical REsearch Project
  - Get a feel for research!
How We Typically Work

- We target some real problem that we find interesting.
- We define the problem precisely.
- We develop a solution that is typically a data structure or an algorithm, i.e., a concrete technique.
- To evaluate, we build prototypes.
  - These are built for the purpose of studying the properties of our solutions.
  - We are often interested in performance, e.g., runtime, space usage, communication cost.
- For some solutions we state formal properties that we then prove, e.g., the correctness of a particular technique.
- Brief: isolate and define problem, construct, then evaluate
How We Typically Work

• We target some real problem that we find interesting.
• We define the problem precisely.
• We develop a solution that is typically a data structure or an algorithm, i.e., a concrete technique.
• To evaluate, we build prototypes:
  • These are built for the purpose of studying the properties of our solutions.
  • We are often interested in performance, e.g., runtime, space usage, communication cost.
• For some solutions, we state formal properties that we then prove, e.g., the correctness of a particular technique.
• Brief: isolate and define problem, construct, then evaluate.

Curious? Come talk to us!

We are looking for:

- student programmer
- Master students
- Phds