Scientific Project: Databases for Multi-dimensional Data, Genomics and modern Hardware

David Broneske, Gabriel Campero, Bala Gurumurthy, Andreas Meister, Marcus Pinnecke, Roman Zoun, Gunter Saake

April 1, 2019
Overview

▶ Concepts of this course
▶ Course of action (milestones, presentations)
▶ Overview of project topics & forming project teams
▶ How to perform literature research?
Overview

- Concepts of this course
- Course of action (milestones, presentations)
- Overview of project topics & forming project teams
- How to perform literature research?
- Further lectures:
  - Academic writing (2-3 lectures)
Organization
**Scientific Project: Modules**

**Bachelor**
- **Module:** WPF FIN SMK (Schlüssel- und Methodenkompetenzen)
- 5 CP = 150h \(\Rightarrow\) 42h presence time (3 SWS) + 108h autonomous work

**Master**
- **Module:** Scientific Team Project (Inf, IngInf, WIF, CV)
  - DKE: Methods 2 or Applications
  - DE: Interdisciplinary Team Project
- 6 CP = 180h \(\Rightarrow\) 42h presence time (3 SWS) + 138h autonomous work

*Grade at the end of the course for the whole project team*
Scientific Project: Prerequisite

- Successful programming test in C++/Java/Python
- 1h theoretical test in a seminar room (data and place to be discussed)
- Half of the team members have to pass the test
- Topics:
  - Some language specifics
  - General program understanding
  - Control flow understanding
- You can take all tests and have to pass at least one!
### Scientific Project: Semester Plan

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>01.04.19</td>
<td>02.04.19</td>
<td>03.04.19</td>
<td>04.04.19</td>
</tr>
<tr>
<td></td>
<td>08.04.19</td>
<td>09.04.19</td>
<td>10.04.19</td>
<td>11.04.19</td>
</tr>
<tr>
<td></td>
<td>29.04.19</td>
<td>30.04.19</td>
<td>01.05.19</td>
<td>02.05.19</td>
</tr>
<tr>
<td>Aca-I</td>
<td>06.05.19</td>
<td>07.05.19</td>
<td>08.05.19</td>
<td>09.05.19</td>
</tr>
<tr>
<td>MS II</td>
<td>13.05.19</td>
<td>14.05.19</td>
<td>15.05.19</td>
<td>16.05.19</td>
</tr>
<tr>
<td>Aca-II</td>
<td>20.05.19</td>
<td>21.05.19</td>
<td>22.05.19</td>
<td>23.05.19</td>
</tr>
<tr>
<td></td>
<td>27.05.19</td>
<td>28.05.19</td>
<td>29.05.19</td>
<td>30.05.19</td>
</tr>
<tr>
<td>MS III</td>
<td>10.06.19</td>
<td>11.06.19</td>
<td>12.06.19</td>
<td>13.06.19</td>
</tr>
<tr>
<td></td>
<td>17.06.19</td>
<td>18.06.19</td>
<td>19.06.19</td>
<td>20.06.19</td>
</tr>
<tr>
<td></td>
<td>24.06.19</td>
<td>25.06.19</td>
<td>26.06.19</td>
<td>27.06.19</td>
</tr>
<tr>
<td>MS Final</td>
<td>01.07.19</td>
<td>02.07.19</td>
<td>03.07.19</td>
<td>04.07.19</td>
</tr>
</tbody>
</table>
Scientific Project: Milestones

- Milestone I - Topic, schedule, and team presentation & first results of literature research
- Milestone II - Concept & additional literature research
- Milestone III - Implementation & evaluation setup
- Milestone IV - Final presentation (wrap-up + evaluation results)
Concepts & Content
Lecture, Meetings & Presentation

Lecture & Presentation

- Time/Place: Tuesday, 09:00-11:00, G29 - K058
- Lectures with content of course → all
- Presentation of main milestones (see time table) → each project team

Meetings (Exercise)

- Individual for each project team
- Time and room to be agreed in project teams!
- Presentation of all intermediate results/milestones (informal)
- Discussion, discussion, discussion . . .
Progress of Course

Deliveries

- 4 milestone presentations (*main milestones*)
- Each team member has to present at least once
- Reporting of (sub) milestones in exercises/meetings
- Written paper about literature research (technical report)
- Prototypical implementation
Deliveries and Grading (I)

Technical Report

- Delivery of report at a given time (deadline)
- Quality/Quantity of literature research
- Number of pages
- Quality of paper structure and evaluation
- Own contribution
Deliveries and Grading (II)

Presentation & Discussion

- Quality of scientific presentation (structure, references, time)
- Assessment regarding the content (e.g., results of particular milestones)
- Participation of discussion

Organization

- Strictness
- Communication (just-in-time answers, satisfying time constraints)
- Self-organization (Sharing tasks, internal reporting of current state-of-work, dealing with problems)
- Autonomous working
Deliveries and Grading (III)

Grade consists of:
- Presentations: 30%,
- Implementation: 30%,
- Paper: 30%,
- Soft Skills: 10%

Binding registration: Second Milestone
Objectives & Qualification (I)

**Acquired skills, specific to research**

- Performing literature research
- Understanding and structured reviewing of scientific work
- Autonomous, solution-based reasoning on research task (e.g., finding alternative solutions)
- How to ask? How to adapt a task (extend/reduce)?
- Academic writing
Acquired skills, always needed

- Team management
- Project and time scheduling
- Presentation of results
- Flexibility regarding changing conditions
- Reasoning about solutions ("Why is this the best/not adequate...")
Task & Time Management

Task Management

- Main milestones have to be finished in time
- (Sub) milestones are less strict (but don’t be sloppy)
- Pre-defined work packages ⇒ each project team
  - ...defines sub work packages
  - ...determines responsibilities for these packages
    -(divide&conquer)

Time Management

- Planning of periods
- Regarding capacities and resources
- Considering other tasks and activities
- Reporting of delays immediately to project members!
Role Management

- Possible roles: team leader, developer, researcher, ...
- Work together vs. responsibilities: design, implementation, testing, writing, ...
- Delegate for important roles/work packages
- Assignment of (sub) tasks to role for each milestone
Topic & Project Teams

- Teams with 4 to 6 students (depends on the task)
- Most tasks can be chosen **once**
- **Projects**
  - **Theoretical part**
    - State of the art
    - New ideas
  - **Practical part**
    - Usually in C++, Java, or Python
    - Prototypical implementation
    - Evaluation part
Intro: Learning to Index From Demonstrations

- Deep Reinforcement Learning is poised to revolutionize autonomous data management solutions.
- Learning from demonstrations (partially offline) is highly valuable for adoption into production systems. It might soon become the norm.
- Index selection is one of the most impactful data management applications to study such learning approach.

Your Task

- Literature research: autonomous index selection, basics on deep reinforcement learning, learning from demonstrations.
- Prototypical implementation: Using Facebook’s Horizon.
- Experimental evaluation and analysis.
Topic 2 - Selma: Self-Driving Materialized Views

Intro: DRL for Materialized View Selection

- Identifying and caching important intermediate results can speed-up query processing on read-mostly databases.
- Selection and eviction strategies are cornerstone for leveraging such results, but traditional caching or heuristics have limitations.
- Deep reinforcement learning can help in this opportunistic materialization, overcoming limitations in the SOTA.

Your Task

- Literature research: materialized view management, basics on deep reinforcement learning.
- Prototypical implementation: Novel environment in OpenAIGym.
- Experimental evaluation and analysis, w/state-of-the-art DQN agents in Google Dopamine.
Intro: Deep Bandits for Operator Variant Selection

▶ Multiple devices, multiple optimizations, different data: few operator variants are able to win all the times.
▶ Contextual multi-arm bandits (often with some deep learning) can be used effectively, in online settings, to target the operator selection problem.
▶ How to make this process truly production-ready, considering trade-offs?

Your Task

▶ Literature research: contextual multi-arm bandits, deep bandits.
▶ Prototypical implementation: using existing libraries, provide a framework to support analysis.
▶ Experimental evaluation and analysis.
**Intro**

- Aggregation over sorted data is a trivial task
- Parallel processing uses (expensive!) prefix-sum to get thread result position + total result size
- Fusion of both might reduce effort

**We’ve got**

- Ordered aggregation kernels
- Benchmark suite and parallel processing dispatcher

**Your Task**

- Literature Research: ordered aggregation + operator fusion
- Understanding of OpenCL and its concepts
- Invention of a concept to fuse prefix-sum with aggregation
- Implementation of your concept of the fused operator
- Benchmarking operation with TPCH
Topic 5 - Hash-Based Query Processing Engine

Intro

▶ Input data (always sorted): by index (column store) + sorted by value (b-tree/sorted store)
▶ Operations are built exploiting this property
▶ Similar operation to this is hashing

We’ve got

▶ Hash-based aggregation + benchmark suite

Your Task

▶ Literature Research: hash based processing systems (general operation)
▶ Understanding hashing concepts
▶ Invention of a concept to implement DBMS operation around hashing (selection, aggregation etc)
▶ Implementation of your concept for the operations
▶ Testing with TPC-H benchmark
Intro
- Parallel chunked execution needs data to be synchronized
- Different synchronization mechanisms depending on last operation in pipeline
- Prefix-sum is well-known concept for thread synchronization

We’ve got
- OpenCL based processing framework
- Kernels for basic DBMS operations & prefix-sum kernel
- TPC-H benchmark suite

Your Task
- Literature Research: data scatter/gather + parallel reduce strategies; synchronization mechanisms
- Invention of a concept for merging DBMS operations
- Implementation of primitives for chunking and merging
- Benchmarking operator using TPCH and other competitors
Intro

▶ Multi-core systems got pace demanding for parallel implementations
▶ Efficient string compression (dictionary compression) needed
  ▶ Maps strings to keys in a hash table
  ▶ Efficient parallel insertion and lookup needed
▶ A suitable hash table: Robin Hood Hashing

We’ve got

▶ libcarbon – a framework with a parallelized string dictionary

Your Task

▶ Literature Research: Robin Hood hashing, Parallel/lock-free processing
▶ Implementation of your concept for Robin Hood hashing
▶ Tests against the straight-forward (currently implemented) hash table
Intro

- Elf: multi-dimensional main memory index structure for efficient selections
- Stores data sorting in a multi-dimensional order
- Common data-intensive operator: Sorting

We’ve got

- Elf implementation in C++

Your Task

- Literature Research: Related index structures and sorting algorithms
- Understanding of the Elf and its optimization concepts
- Implementation Sorting Operator for Elf
- Performance evaluation against sequential scans
For TPC-H queries with multi-column selection predicates, we observe a prefix redundancy elimination. In the second predicates on the conceptual level:

- Clause, (b) selectivity, and (c) response time of TPC-H

Elf features cache sensitivity, an optimized structure that is able to exploit the relation between several columns. Using Elf results to a more manageable size. This task has become even easier.

Our results indicate a clear superiority of our approach for multi-column selection predicates (1.72 % for Q6). For TPC-H, queries with multi-column selection predicates, we achieve a speed-up between a factor of five and two orders of magnitude. Thus, most OLAP queries would never use full table scan even smaller than for disk-based database systems. Hence, most OLAP queries would never use full table scan.

Predicate evaluation is an important task in current OLAP systems. One leaves tuning opportunities open that arise if all predicates are evaluated using accelerated scans. However, scanning each column one by one leaves tuning opportunities open that arise if all predicates are evaluated using accelerated scans. Hence, most OLAP queries would never use full table scan.

For efficiency, multi-column selection predicate evaluation, we propose Elf, an index structure that is able to exploit the relation between data of several columns. Using Elf results to a more manageable size. This task has become even easier.

Our results indicate a clear superiority of our approach for multi-column selection predicates.
Topic 9 - Alternative Node Designs for Elf

Intro

- Elf: multi-dimensional main memory index structure for efficient selections
- Stores data sorting in a multi-dimensional order
- Nodes store values and pointers in interleaved fashion
- Alternative layouts seem promising (cf. ART-Tree)

We’ve got

- Elf implementation in C++

Your Task

- Literature Research: Improvements for linearized tree nodes
- Understanding of the Elf and its optimization concepts
- Implementation different node designs for Elf
- Performance evaluation against standard implementation
Topic 10 - Parallel Selections in Elf

Intro

- Elf: multi-dimensional main memory index structure for efficient selections
- Stores data sorting in a multi-dimensional order
- Multi-core architectures demand for a clever parallelization strategy

We’ve got

- Elf implementation in C++

Your Task

- Literature Research: Related parallelization strategies for index structures
- Understanding of the Elf and its optimization concepts
- Implementation multi-threaded traversal variants for Elf
- Performance evaluation against serial implementation
Finding your Team

Topics:

- Topic 1 - Learning to Index from Demos (IndexHorizon)
- Topic 2 - Self-Driving Materialized Views (Selma)
- Topic 3 - Deep Bandits for Variant Selection (DeepOps)
- Topic 4 - Prefix-Sum Less Ordered Aggregation
- Topic 5 - Hash-Based Query Processing Engine
- Topic 6 - Parallel Execution w/ Synchronization Primitives
- Topic 7 - Thread-Safe Task-Parallel Robin Hood Hashing for Main-Memory Databases
- Topic 8 - Order-By Queries on Elf
- Topic 9 - Alternative Node Designs for Elf
- Topic 10 - Parallel Selections in Elf

When do we meet for the programming test?
Literature Research
How to Perform Literature Research

Efficient literature research requires

- Knowledge of *Where* to search
- Knowledge of *How* to search
- Finding adequate search terms
- Structured review of papers
- Knowledge of how to find information in papers
Where to Search (I)

Different websites available that provide large literature databases

1. Google Scholar: http://scholar.google.de/
   - Key word and concrete paper search
   - Often, PDFs are provided

2. DBLP: http://www.informatik.uni-trier.de/~ley/db/
   - Search for keyword, conferences, journals, author(s)
   - BibTex and references to other websites

3. Citeseer: http://citeseerx.ist.psu.edu/about/site
   - keyword, fulltext, author, and title search
   - BibTex and (partially) PDFs are provided
Publisher sites are also a suitable target

- **ACM Digital Library**: [http://portal.acm.org/dl.cfm](http://portal.acm.org/dl.cfm)
  - Keyword, author, conference/literature (proceedings), and title search
  - Bibtex, mostly PDFs and other information are provided

  - Similar to ACM, but only few PDFs
  - Extended access within university network

- **Springer**: [http://www.springerlink.de/](http://www.springerlink.de/)
  - Similar to previous
  - Extended access within university Network

- Further search possibilities: on author, research group or university sites
How to Search

Some hints to not get lost in the jungle

▶ Use distinct keywords (*fingerprint* vs. *fingerprint data*)
▶ Keep keywords simple (at most three words)
▶ Otherwise, search for whole title
▶ Read abstract (and maybe introduction) ⇒ decision for relevance

First insights

▶ Read abstract, introduction and background/related work (coarse-grained) to
  ▶ . . . get a first idea of the approach
  ▶ . . . find other relevant papers
Finding the required information

- Read the paper carefully
- Omit formal parts/sections
- Try to classify (core idea, main characteristics) ⇒ develop classification/evaluation in mind
- Understand the big picture
- Make notes
- Do NOT translate each sentence
Finding your Team

Topics:

► Topic 1 - Learning to Index from Demos (IndexHorizon)
► Topic 2 - Self-Driving Materialized Views (Selma)
► Topic 3 - Deep Bandits for Variant Selection (DeepOps)
► Topic 4 - Prefix-Sum Less Ordered Aggregation
► Topic 5 - Hash-Based Query Processing Engine
► Topic 6 - Parallel Execution w/ Synchronization Primitives
► Topic 7 - Thread-Safe Task-Parallel Robin Hood Hashing for Main-Memory Databases
► Topic 8 - Order-By Queries on Elf
► Topic 9 - Alternative Node Designs for Elf
► Topic 10 - Parallel Selections in Elf