Database Systems and Information Management Group
Fak. IV Electrical Engineering and Computer Science
Technische Universität Berlin

Bachelor's and Master's Thesis Topics

Academic Year 2020/21

Last Update: February 15, 2021

Table of Contents

Foreword

This document contains a compilation of thesis topics for both Bachelor's and Master's students pursuing a degree at the Technische Universität Berlin. It reflects the current research activities in the Database Systems and Information Management (DIMA) Group. Students should identify a topic or two and contact the respective advisor via email. Advisor email addresses are listed on this webpage: https://www.dima.tu-berlin.de/menue/team/.

Many DIMA advisors are already working with several students and it may be that they do not currently have the capacity to take on another student. For this reason, you will need to inquire about their availability. In several cases, the topics listed in this document can be customized based on your knowledge and skills. To ease the initial meeting, please complete and forward the *Thesis Request Form*[1] (Table 1) to your prospective advisor. If you have any questions feel free to reach out to Juan Soto (juan.soto@tu-berlin.de) for an appointment.

We wish you great success!

Student Name (Surname, First Name)		Mustermann, Erika
Thesis Level	• B.Sc. or M.Sc.	• M.Sc.
Thesis Topic	• research area(s)	• databases for emerging hardware
Technical Skills	• programming languages	• C++, Java, Scala, Python, R
	• systems	• Flink, Hadoop, Spark
	• software	• DB2, Oracle, Postgres
Foundational Knowledge	• B.Sc. courses completed (marks)	• ISDA (2,0), DBPRA (1,3), DBPRO (1,7), • DBSEM (1,3), DW (1,0)
	• M.Sc. courses completed (marks)	• DBT (1,7), IDBPRA (2,0), AIM-2 (1,3), AIM-3 (1,7), • BDAPRO (1,3), BDASEM (1,3), MHD (1,0), ROC (1,3)
Practical Experience (only complete those that are appropriate)	• research assistant	• Conducted research in query optimization at ...
	• open-source code contributions	• Code contributions to **Apache ProjectX**, https://github.com/Mustermann_Erika/ProjectX/
	• student intern	• Interned at Google in summer 2019, worked on ...
	• full-time employment	• As a BI Analyst (Deutsche Telekom), I performed ...

Table 1. A representative example of a completed Thesis Request Form.

[1] The *Thesis Request Form* as well as the DIMA *Thesis Proposal Template* are both available for download from this webpage: https://www.dima.tu-berlin.de/menue/thesis_opportunities/.

Thesis Opportunities in the DIMA Group

Theses in the DIMA Group are *often tied to ongoing research projects* sponsored by (inter-) national funding agencies and are *commonly written in English (and some in German).* Problems are typically centered on topics in database systems as well as scalable and distributed data management, including:

- benchmarking and performance evaluation,
- data visualization,
- data warehousing, OLAP, SQL Analytics,
- database monitoring and tuning,
- database security, privacy, access control,
- databases for emerging hardware,
- data systems and data management for machine learning,
- distributed and parallel databases,
- graph data management, RDF, social networks,
- knowledge discovery, clustering, data mining,
- machine learning for data management and data systems,
- query processing and optimization,
- spatio-temporal databases,
- storage, indexing, and physical database design,
- streams, sensor networks, complex event processing,
- transaction processing,
- very large data science applications/pipelines.

To pursue a thesis with us, students are generally required to possess:

- *outstanding programming skills* in C++, Java, or Scala,
- *extensive knowledge in database systems* (e.g., IBM DB2, Oracle) or *big data analytics systems* (e.g., Flink, Spark),
- *basic knowledge in the use of an IDE* (e.g., Eclipse, IntelliJ),
- *basic knowledge in the use of a distributed version control system* (e.g., SVN, Git).

Furthermore, to conduct a:

- **Bachelor's thesis**, students *must have successfully completed* **ISDA** and **DBPRA** (at a minimum) *with a grade of good or better* and possibly several other Bachelor's courses offered by DIMA, such as DBPRO, DBSEM, or DW.
- **Master's thesis**, students *must have successfully completed* **DBT** and **IDBPRA** (at a minimum) *with a grade of good or better* and possibly several other Master's courses offered by DIMA, such as AIM-2, AIM-3, BDAPRO, BDASEM, MHD, or ROC.

Moreover, depending on the thesis topic, additional knowledge may be required (e.g., compiler technology, distributed systems, machine learning).

1. Geo-Distributed Data Analysis

Advisor: Dr. Kaustubh Beedkar

Appropriate Target Level: B.Sc., M.Sc.

Keywords: geo-distributed data analysis

Description: Many large organizations today operate data centers that produce large amounts of data at different locations around the globe. Analyzing geographically distributed data is essential to derive valuable insights. Typically, geo-distributed data analysis is carried out by either *communicating all of the data to a central location, where analytics are performed* or *employing a distributed execution strategy to minimize data communication*. However, legal constraints arising from regulatory bodies concerning data sovereignty and data movement (e.g., prohibiting the transfer of data across national borders) as well as technical constraints arising from the use of heterogeneous compute nodes pose serious limitations to existing approaches. Our research explores how to declaratively specify the algorithms as well as the legal and technical constraints to automatically derive distributed execution strategies.

Note: Students are also encouraged to propose their own topic in the ambit of the above research problem.

Prerequisites: *strong programming skills* (preferably in Java); *knowledge in query planning and execution in DBMS*; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

2. Data Stream Modeling and Processing

Advisor: Dr. Alexander Borusan

Appropriate Target Level: B.Sc., M.Sc.

Keywords: data stream management in embedded information systems

Description: Automotive, avionic, and manufacturing applications built atop embedded information systems, typically include two tasks: *monitoring* and *controlling*. Many of these applications continuously generate data streams, which are ordered sequences of data that can only be read once and ideally processed in real-time. From the point of view of data stream management, several tasks need to be solved, in order to: *model the data, process the data* (e.g., incorporating appropriate data structures, employing data reduction techniques), *querying the data*, *scheduling jobs*, as well as *storing the data*. Additionally, over the past decade, *data stream analysis* has become one of the most important tasks.

Examples of topics that could be explored, include

- M.Sc. Thesis: An examination of existing and the development of novel architectures and models for the real-time processing of streaming data originating in embedded information systems. For example, for systems that are employed in the automotive or avionics sector.

- B.Sc. Thesis: An analysis of data reduction techniques suitable for automotive applications involving data streams, including the development of a taxonomy.

Prerequisites: *strong programming skills in Java*; *good writing skills*; *knowledge in Apache Flink*; *knowledge in data streams management systems* (DSMS); (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA); *solid mathematics background*.

3. Scalable Time Series: Modeling and Forecasting

Advisor: Dr. Marcela Charfuelan

Appropriate Target Level: B.Sc., M.Sc.

Keywords: time series processing, sequence analysis, pattern recognition, state space models, complex event processing

Description: In database systems, time series can be viewed as a subset or specific type of data stream. According to the *Encyclopedia of Data Base Systems* [1], data streams consist of sequences of data instances that continuously flow in and out a system with varying update rates. One of the main differences between *(continuous) data streams* and *traditional (i.e., batch, historical) datasets* is that data streams are (primarily) automatically generated. Very often the destination of such streams are other computers or automatic systems [2]. The assumption of time series databases is that consecutive or repeated measurements are taken at spaced and equal time intervals, for example, days, hours, seconds, etc. [3].

Time intervals between successive data points are usually assumed to be uniform (e.g., video, audio, ECG); in some streaming time series the time interval is irrelevant though, and just the temporal order is of significance (e.g., RFIDs, Twitter streams, web click streams). Classical examples of time series include *stock market trading data*, *observation of natural phenomena* (e.g., atmosphere, temperature, wind, humidity), *physiological measurements* in the medical domain, and *telecommunication network data*. More recent examples include *geo-spatial data* (mainly collected using remote sensors), *smart sensor data* (e.g., smart meter measurements for electricity, water, gas and heating), and *smart factory data* (from sensors integrated into manufacturing machinery in the so called Industry 4.0).

The importance and relevance of time series today is reflected in two recent surveys related to *Time Series Management Systems* (TSMS) and *Time Series Data Bases* (TSDB), which describe, analyse and compare systems specialised in storing and querying time series data [4,5]. In general we can say that stream processing is not a requirement for TSMS, some might include it or not; this means that TSMS most of the time will work with batch or historical, already stored, time series. Apart from the TSMS mentioned in the previous surveys, nowadays open-source systems or frameworks for distributed stream data processing have appeared. These distributed systems are capable of processing data on computer clusters, so they can handle big amounts of continuous data. These systems are also optimized to process data in real-time with low latency due to parallel/distributed execution. Examples of these open-source frameworks include Apache Spark streaming or Apache Flink.

Note: Research topics are typically centered on the development of *scalable data analytics* and *distributed stream processing systems* for large-scale time series.

Prerequisites: strong programming skills (e.g., Java, R, Python); good writing skills; (preferably) knowledge in Apache Flink; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM).

References

[1] L. Liu and M. T. Zsu. Encyclopedia of Database Systems. Springer US. 2009. [Online]: http://link.springer.com/101007/978-0-387-39940-9

[2] J. Gama. Knowledge Discovery from Data Streams. CRC Press Taylor & Francis Group. 2010.

[3] J. Han and M. Kamber. Data Mining Concepts and Techniques (2nd Ed.). Morgan Kaufmann. 2006.

[4] S. K. Jensen, T. B. Pedersen, and C. Thomsen. Time Series Management Systems: A Survey. IEEE Transactions on Knowledge and Data Engineering Vol. 29, No. 11, pp. 2581–2600. November 2017. DOI: https://doi.org/10.1109/TKDE.2017.2740932

[5] M. F. Andreas Bader, Oliver Kopp, "Survey and comparison of open source time series databases," in BTW2017 – Workshop band, ser. Lecture Notes in Informatics (LNI), B. Mitschang et al., Eds., vol. P-266. Gesellschaft für Informatik e.V. (GI), 2017, pp. 249–268. [Online]: ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/INPROC-2017-06/INPROC-2017-06.pdf

4. Data Analytics for Massive Time Series

Advisor: Dr. Holmer Hemsen

Appropriate Target Level: B.Sc., M.Sc.

Keywords: scalable signal processing

Description: A time series is a set of observations each recorded at a specific time. Examples of time series are many fold (e.g., electrocardiography curves, stock market, seismic measurements, network load). Time series analysis comprises a wide range of methods, such as *anomaly and outlier detection*, *forecasting*, and *pattern recognition*. The focus of this topic area is on research of methods for the analysis of massive and/or multi-dimensional time series.

Prerequisites: strong programming skills in Java, Scala or Python; good writing skills; (preferably) knowledge in Apache Flink; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM).

5. Intelligent and Scalable Resource Management for Industrie 4.0

Advisor: Dr. Holmer Hemsen

Appropriate Target Level: B.Sc., M.Sc.

Keywords: Industrie 4.0

Description: The goal of Industrie 4.0 is to digitalize, automate and optimize industrial production systems. In many cases, this involves upgrading conventional production systems into cyber-physical systems, often drawing on Internet of Things (IoT) technologies. The focus of this research topic is on methods for the scalable optimization of production lines and the intelligent forecasting of consumable resources to calculate optimal dynamic maintenance strategies.

Prerequisites: strong programming skills in Java, Scala or Python; good writing skills; (preferably) knowledge in Apache Flink; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM).

6. Debugging Machine Learning Systems

Advisor: Dr. Zoi Kaoudi

Appropriate Target Level: B.Sc., M.Sc.

Keywords: data management for machine learning systems

Description: Nowadays, numerous machine learning (ML) libraries and systems have been developed, to build ML models over big data. However, a key stumbling block in truly leveraging big data analytics insights is the need for both *big data* and *ML* debugging. Most of the existing tools and frameworks have been developed for *code*-based debugging and are neither flexible nor sufficient for data and ML debugging. We seek to develop a powerful, flexible and intuitive ML debugging system, that is able to debug an entire ML workflow, from model training and inference to ML diagnostics.

Prerequisites: strong programming skills (e.g., Java); good writing skills; (preferably) knowledge in Apache Flink; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, AIM-3, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA); machine learning knowledge.

7. Scalable Machine Learning Systems for Streaming Graphs

Advisor: Dr. Zoi Kaoudi

Appropriate Target Level: B.Sc., M.Sc.

Keywords: data management for machine learning systems

Description: Many real-world applications require machine learning on graphs that are continuously changing (i.e., *streaming graphs*). Examples include *social networks* (where new friendships are perpetually created), *applications that monitor and predict new connections* (to improve recommendations), and the *Internet of Things* (where millions of participants change their physical location continously). In each of these cases, we need solutions that are able to cope with the highly-dynamic behavior that is prevalent. We seek to develop a system that can run analytics and machine learning tasks over streaming graphs.

Prerequisites: strong programming skills (e.g., Java); good writing skills; (preferably) knowledge in Apache Flink; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA); machine learning knowledge.

8. Secure Federated Schema and Data Matching

Advisor: Dr. Alireza Rezaei Mahdiraji

Appropriate Target Level: M.Sc.

Keywords: federated learning, schema matching, secure computation

Description: *Federated learning* [1] is a distributed machine learning approach that enables training on (geographically) distributed data. In particular, it has been successfully applied in the healthcare domain, where the security and privacy of patient data is paramount. The most common type of federated learning is *horizontal federated learning* (HFL). In HFL, it is assumed that across parties, there is a *common data schema* (feature space) and that their *entity sets* are *largely disjoint* (i.e., horizontally partitioned). In this thesis, the goal is to align schemas across federated parties and securely perform entity matching (without sharing any sensitive data), comparable to what is done in [2, 3]. In the approach discussed in [2], when data schemas are *common* across parties, *entity matching* is performed by embedding the entities of each party in a shared space and computing their similarity. However, when the schemas are *disparate*, a trusted third party is used to devise a common schema. In this thesis, the goal is to develop and implement a novel privacy-preserving schema that is suitable for performing entity matching, akin to [2, 3]. Furthermore, the scalability of the proposed solution should be evaluated, and possibly improved for a federated setting involving multiple parties and across variable dataset sizes.

Prerequisites: strong programming skills; good writing skills; substantial knowledge in databases; familiarity with federated learning; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA);

References

[1] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.
[2] Monica Scannapieco, Ilya Figotin, Elisa Bertino, and Ahmed K. Elmagarmid. Privacy preserving schema and data matching. SIGMOD 2007.
[3] Hardy, Stephen, et al. "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption." arXiv preprint arXiv:1711.10677 (2017).

9. Big Data Processing

Advisor: Dr. Jorge Quiane Ruiz

Appropriate Target Level: B.Sc., M.Sc.

Keywords: scalable data management

Description: Increasingly, applications need to selectively extract relevant data of interest from large volumes of diverse datasets generated at high velocity. In order to cope with today's applications needs and meet the demand, scalable data processing tools and techniques must be employed.

If you have ever wondered how:

(a) *databases cope with complex data*?
(b) *dataflow systems* (e.g., Flink or Spark) *work*?
(c) *to effectively use existing big data systems*?
(d) *big data can help a company or organization*?
(e) *big data helps machine learning*?
(f) *big data will impact the future*?

…. then this area of research will be of interest to you.

Specific Thesis Topics:

1. Benchmarking Intermediate Data Representation for Fast Data Transfers
2. Scalable Trusted Data Processing

Prerequisites: *strong programming skills in Java*; *good writing skills*; *knowledge in Apache Flink*; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

10. Data-Related Ecosystem

Advisor: Dr. Jorge Quiane Ruiz

Appropriate Target Level: B.Sc., M.Sc.

Keywords: scalable data management

Description: Today, data intelligence is monopolized by just a few companies, given that they possess both *sizable amounts of data* and *technological solutions*, to address big data challenges and solve artificial intelligence (AI) problems. Therefore, it is vital to provide novel ways to share data and leverage big data/AI technologies, such as an ecosystem, so that everyone can benefit from the data intelligence era. However, building such an ecosystem is quite challenging, since we do not yet have the right data infrastructure. If this appeals to you, join the Agora Project, which aims to devise a suitable data infrastructure and make a data-related ecosystem possible.

Specific Thesis Topics:

1. Benchmarking Monitoring Systems for Widely-Open Distributed Environments
2. Visual Data Processing

Prerequisites: *strong programming skills in Java*; *good writing skills*; *knowledge in Apache Flink*; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

11. Data Debugging

Advisor: Dr. Jorge Quiane Ruiz

Appropriate Target Level: M.Sc.

Keywords: scalable data management

Description: How many times have you heard that *big data* (bd), *data science* (ds), or *machine learning* (ml) are helping scientists (from varying domains) to make great progress? But, have you ever heard (even once) how hard it is to get those bd, ds, or ml pipelines shiny (i.e., readily accessible) for the applied practitioner? The reality is debugging bd, ds, or ml pipelines is 'taboo': nobody wants to talk about it, but we all suffer from it! Data debugging seeks to break this taboo. We are developing a general-purpose data debugging system that enables the interactive debugging of bd, ds, and ml pipelines.

Specific Thesis Topics:

1. Interactive Data Debugging
2. Visual Data Debugging

Prerequisites: *strong programming skills in Java*; *good writing skills*; *knowledge in Apache Flink*; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

12. An Analysis of Data Analytics Libraries

Advisor: Juan Soto

Appropriate Target Level: B.Sc., M.Sc.

Keywords: data analytics, machine learning, quality assurance, mathematical software

Description: The most recent *Data and AI Landscape* [1] illustrates the growing number of data analytics (e.g., machine learning) libraries [2,3,4,5,6,7,8] that are increasingly available. These libraries are commonly employed in data science pipelines and comprised of numerous underlying mathematical and statistical libraries. However, more research needs to be undertaken to *assess their correctness* (in light of diverse datasets) or *guarantee that they are of high-quality* (e.g., able to cope with machine precision or roundoff errors, employ reliable underlying mathematical software). In this thesis, we want to take a closer look at the reliability of data analytics libraries. The approach to be undertaken includes conducting a theoretical study of the selected algorithms, the generation of synthetic data and selection of appropriate datasets, as well as performing empirical/numerical studies to assess quality.

Prerequisites: data analysis; programming skills; software testing; familiarity with mathematical, statistical, or data analytics libraries; (ideally) completed course work in algorithms, (numerical) linear algebra, machine learning, numerical analysis, probability and statistics as well as DIMA courses (e.g., AIM-3, BDAPRO, BDASEM).

References

[1] Data and AI Landscape 2020. https://shorturl.at/cghvX.
[2] Intel® oneAPI Data Analytics Library. https://shorturl.at/kGMZ9.
[3] Apache MADlib: Big Data Machine Learning in SQL. https://madlib.apache.org/.
[4] Apache Mahout. https://mahout.apache.org/.
[5] H_2O Machine Learning Platform, https://www.h2o.ai/
[6] ScalaNLP Suite of Libraries. http://www.scalanlp.org/.
[7] Apache Spark MLlib. https://spark.apache.org/mllib/.
[8] Apple Turi Create. https://github.com/apple/turicreate.
[9] SE4ML – Software Engineering for AI-ML-based Systems, Dagstuhl Seminar 20091.
[10] Machine Learning Testing: Survey, Landscapes and Horizons. https://shorturl.at/qtyK5a and https://doi.org/10.1109/TSE.2019.2962027.
[11] Shin Nakajima. Quality Assurance of Machine Learning Software. GCCE 2018.
[12] Tutorial on Software Testing & Quality Assurance for Machine Learning Applications from research bench to real world. CoDS COMAD 2020. https://shorturl.at/bfkBP (Short Paper) and https://shorturl.at/dBV25 (Tutorial Materials).
[13] Numerical Issues in Statistical Computing for the Social Scientist. December 2003. https://shorturl.at/irCRW.

13. Resilient Data Management for the Internet of Moving Things

Advisor: Dr. Eleni Tzirita Zacharatou

Appropriate Target Level: M.Sc.

Keywords: adaptive data management, Internet of Things, stream processing, fault tolerance

Description: The Internet of Things (IoT) is a system of interconnected devices that exchange data over networks without human intervention. Some of the most important pieces in the IoT landscape are devices that can move (i.e., continuously change their geo-location over time), such as smartphones and tablets. Today's mobile devices are ubiquitous and their computing capabilities are ever-increasing. They are able to perform data processing tasks, and thereby reduce data communication. However, executing queries over distributed mobile resources is challenging, mainly due to the dynamically changing connectivity among mobile nodes. In my research, I investigate the impact of mobility on an IoT data management system that answers thousands of queries over millions of devices. Furthermore, I explore methods and algorithms to mitigate the effects of mobility and enable robust query execution.

Prerequisites: *good writing skills*; (ideally) completed coursework (or possess experience) in C++ programming and distributed systems (i.e., stream processing engines, databases); (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

14. Tree-based Bitmap Index Compression

Advisor: Dr. Eleni Tzirita Zacharatou

Appropriate Target Level: M.Sc.

Keywords: adaptive data management, indexing, compression

Description: In database systems, *bitmap indexes* are widely used to enable the efficient execution of ad-hoc queries over large amounts of data. A *bitmap index* consists of a set of *bitmaps* (i.e., arrays of bits). By performing bitwise logical operations on (a subset of) bitmaps, bitmap indexes are used to answer queries. Queries over bitmap indexes are fast, because the bitwise logical operations are well-supported by computer hardware. Each bitmap within a bitmap index is typically stored in a *compressed* manner. Several specialized algorithms for *bitmap compression* have been proposed in the literature. Today, the state-of-the-art [1,2] approach is the *Word-Aligned Hybrid* (WAH) compression scheme, which relies on the run-length encoding technique.

Another interesting line of work aims to compress bitmap indexes [3] or plain bitmaps [4] via *tree structures*. Recently authors in [4] have proposed using binary trees to compress bitmaps and using modern hardware primitives (e.g., SIMD) have optimized their initial algorithm. We propose *RUBIK* [3], a novel bitmap index compression scheme, that enables space savings via quad-trees. Its main novelty is the joint compression of all the bitmaps within a bitmap index, rather than compressing each bitmap individually, as performed in [4]. The main goal of this thesis is to devise a *hybrid bitmap index compression scheme*, which ranges between single bitmap compression to a holistic bitmap index compression, in an effort to support diverse query workloads, making the index broadly applicable to various application domains.

Prerequisites: strong C++ programming skills; *good writing skills*; substantial knowledge in databases; (ideally) completed coursework (or possess experience): DBT, DBT-PRA, and other database lab and seminar courses (e.g., BDAPRO, BDASEM, DBPRA, DBPRO, DBSEM, ISDA).

References

[1] K. Wu, E. Otoo, and A. Shoshani. Compressing bitmap indexes for faster search operations. *SSDBM 2002.*
[2] Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. Optimizing Bitmap Indices with Efficient Compression. *ACM Transactions on Database Systems, 31(1), 2006.*
[3] Eleni Tzirita Zacharatou, Farhan Tauheed, Thomas Heinis, and Anastasia Ailamaki. Rubik: Efficient threshold queries on massive time series. *SSDBM 2015.*
[4] Harald Lang, Alexander Beischl, Victor Leis, Peter Boncz, Thomas Neumann, Alfons Kemper. Tree-encoded Bitmaps. *SIGMOD 2020.*

15. Query Optimization, Processing and Execution on Modern CPUs

Advisor: Dr. Steffen Zeuch

Appropriate Target Level: M.Sc.

Keywords: query optimization, query execution on modern CPUs

Description: Over the past decades, database systems have migrated from *disk* to *memory architectures*, such as RAM, Flash, or NVRAM. Research has shown that this migration fundamentally shifts the performance bottleneck upwards in the memory hierarchy. Whereas *disk-based* database systems were largely dominated by disk bandwidth and latency, *in-memory* database systems mainly depend on the efficiency of faster memory components (e.g., RAM, caches, registers). With respect to hardware, the 'clock speed per core' reached a plateau due to physical limitations. To overcome this limit, hardware architects dedicated an increasing number of available on-chip transistors to processors and caches. Unfortunately, the improvements to the *memory bandwidth* far outpaced the improvements to the *memory access latency*. Nowadays, CPUs are able to process data far faster than they are able to transfer data from main memory to caches. Consequently, this trend is referred to as the *memory wall*, which is the main challenge in modern main memory database systems. In this proposed thesis, students will seek to reduce the impact of the memory wall and realize the full potential of the available processing power in modern CPUs for database systems.

Note: Students are encouraged to propose their own topic in the field of query optimization, processing, and execution on modern CPUs.

Prerequisites: strong programming skills in C/C++; *good writing skills*; deep knowledge in database implementation techniques; good understanding of computer architecture; (ideally) knowledge in LLVM, Vtune, MPI, and OpenMP; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, MHD).

16. Adaptive and Decentralized Approaches for Performance Monitoring

Advisor: Xenofon Chatziliadis

Appropriate Target Level: B.Sc., M.Sc.

Keywords: data stream processing and management, performance monitoring

Description: The processing of geo-distributed data streams is a key challenge for many applications, including the Internet of Things (IoT). Cloud-based stream processing engines (SPE) process data centrally and thus require all data to be present in the cloud prior to processing them. However, this centralized approach becomes a bottleneck, when processing data from millions of geo-distributed sensors in a large-scale IoT infrastructure. Currently, a next generation data management system for the IoT is under development called *NebulaStream* (www.nebula.stream), which aims to mitigate the bottleneck using a decentralized approach involving fog devices. A major challenge for a SPE in this unified fog-cloud environment is the monitoring of all devices in a large-scale, heterogeneous topology. Among the performance monitoring issues that we seek to address are:

- efficiently deciding which features/indicators to sample (using state-of-the-art adaptive sampling techniques or machine learning methods) and when to send them to a central monitoring unit (coordinator),
- developing methods to reduce the network traffic in a distributed topology involving millions of IoT devices,
- examining state-of-the-art data storage techniques and determining how to perform complex analytics on monitoring data (arising from millions of nodes) on a single node,
- benchmarking existing monitoring components in modern SPEs, like Flink, Kafka or Heron.

Prerequisites: strong programming skills preferably in C++; *good writing skills*; (ideally) completed coursework in DBT and DBT-PRA (or possess the equivalent knowledge in distributed systems, stream processing engines, and databases).

17. Query Optimization in Distributed Stream Processing Systems

Advisor: Ankit Chaudhary

Appropriate Target Level: B.Sc., M.Sc.

Keywords: query optimization and operator placement in distributed stream processing systems

Description: The processing of geo-distributed data streams is a key challenge for many applications, such as the Internet of Things (IoT). Cloud-based stream processing engines (SPE) process data centrally and thus require all data to be present in the cloud in the cloud prior to processing them. However, this centralized approach becomes a bottleneck, when processing data from millions of geo-distributed sensors in a large-scale IoT infrastructure. Currently, a next generation data management system for the IoT is under development called *NebulaStream* (www.nebula.stream), which aims to mitigate the bottleneck using a decentralized approach involving fog devices. A major challenge for a SPE in this unified fog-cloud environment is the need to execute millions of queries concurrently. *NebulaStream* should be able to optimize and deploy incoming queries at high speed. Currently, we are investigating how to efficiently place operators for a large number of queries in a geo-distributed unified fog-cloud environment. Furthermore, we are conducting research to mitigate the effect of transient failures as well as monitor the effect that running queries have on system performance.

Prerequisites: strong programming skills preferably in C++; *good writing skills*; a good understanding of query optimizers in database management systems; (ideally) completed coursework in DBT and DBT-PRA (or possess the equivalent knowledge in distributed systems, stream processing engines, and databases).

18. Algorithmic Enhancements for the End-to-End Management of Large State in Distributed Stream Processing Engines

Advisor: Bonaventura Del Monte

Appropriate Target Level: M.Sc.

Keywords: state management for distributed data stream processing

Description: We seek to develop novel algorithms to enhance the end-to-end management of large state in distributed stream processing engines. Among the system characteristics (treated as first-class citizens) of interest are *resource elasticity*, *fault tolerance*, *load balancing*, *robust execution of stateful query*, and the *optimization of query plans*.

Concrete Thesis Topics:

1. **Exploring Lightweight Fault Tolerance for IoT Data Stream Processing**
   IoT infrastructures consist of thousands of devices connected over unreliable networks. To support stream processing workloads on an IoT infrastructure, systems require *fault-tolerance* to ensure consistency in the results. In this thesis, students will need to develop novel protocols for lightweight fault-tolerance and thereby enable reliable distributed stream processing. As a starting point, the Chandy-Lamport algorithm [1] for snapshotting a distributed system, and *mergeable replicated data types* [2], in the context of stateful stream processing will be examined.
   Expected Outcome: The design, implementation, and benchmarking of distributed protocols to enable lightweight fault-tolerance and support stateful stream processing

2. **Efficient Storage Backends for Stream Processing**
   Stream processing pipelines usually store state in a key/value data structure (hashmap). In this thesis, students will evaluate current hashmap implementations for purely *in-memory* as well as *disk-based* implementations.
   Expected Outcome: The *design of a benchmark for* as well as *an implementation of* multiple storage backends in *NebulaStream*, and the evaluation of the system across varying workloads. This thesis will be co-advised with Philipp Grulich.

Prerequisites: strong programming skills in C++; good writing skills; a good understanding of distributed database systems as well as stream processing systems; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA).

References

[1] *Distributed Snapshots: Determining Global States of Distributed Systems*, http://lamport.azurewebsites.net/pubs/chandy.pdf.
[2] *Mergeable Replicated Data Types*, http://shorturl.at/cvHJL.

19. Horizontal Federated Learning Model Selection

Advisor: Behrouz Derakhshan

Appropriate Target Level: M.Sc.

Keywords: federated learning, model selection

Description: *Federated learning* [1] is a distributed machine learning approach that enables training on (geographically) distributed data. In particular, it has been successfully applied in the healthcare domain, where the security and privacy of patient data is paramount. The most common type of federated learning is *horizontal federated learning* (HFL). In HFL, it is assumed that across parties, there is a *common data schema* (feature space) and that their *entity sets* are *largely disjoint* (i.e., horizontally partitioned). The problem of model selection, i.e., finding the best performing model among a set of run-configurations [2], arises in federated learning. Although strides have been made [2, 3] to address this problem, there are some open challenges. For example, how to handle heterogeneous resources, how to cope with distributed data, and how to manage the large number of hyperparameters. In this thesis, the goals are to address these challenges, and optimize the model selection process in HFL. Initially, you will investigate existing model selection approaches, then propose a solution that ensures *high resource utilization*, *efficient network-bandwidth usage*, as well as *security and privacy compliance*.

Prerequisites: strong understanding of machine learning basics; good writing skills; strong programming skills (preferably in Python); familiarity with federated learning; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

References

[1] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.
[2] Kumar, Arun, et al. "Model selection management systems: The next frontier of advanced analytics." ACM SIGMOD Record 44.4 (2016): 17-22.
[3] Nakandala, Supun, Yuhao Zhang, and Arun Kumar. "Cerebro: a data system for optimized deep learning model selection." Proceedings of the VLDB Endowment 13.12 (2020): 2159-2173.

20. Decremental Federated Machine Learning

Advisor: Behrouz Derakhshan

Appropriate Target Level: M.Sc.

Keywords: federated learning, incremental/decremental machine learning

Description: *Federated learning* [1] is a distributed machine learning approach that enables training on (geographically) distributed data. In particular, it has been successfully applied in the healthcare domain, where the security and privacy of patient data is paramount. The most common type of federated learning is *horizontal federated learning* (HFL). In HFL, it is assumed that across parties, there is a *common data schema* (feature space) and that their *entity sets* are *largely disjoint* (i.e., horizontally partitioned). Recent laws require organizations to remove user data upon request (i.e., users have the right to be forgotten [2]). This equally applies to machine learning models. This process is referred to as decremental learning (or unlearning). The most common decremental learning approach is to retrain the model on the data after the deletion of the user data. Recent research has sought to optimize the re-training process [3, 4]. However, alternative approaches have also been proposed (e.g., the efficient non-retraining-based “deletion” methods for a subset of the ML models [5, 6, 7]). The goal of this thesis is to devise an efficient decremental learning approach for HFL and evaluate it for select machine learning algorithms.

Prerequisites: strong understanding of machine learning basics; good writing skills; strong programming skills (preferably in Python); familiarity with federated learning; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

References

[1] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.
[2] https://en.wikipedia.org/wiki/Right_to_be_forgotten
[3] Incremental and Decremental Training for Linear Classification, KDD 2014
[4] DeltaGrad: Rapid retraining of machine learning models, PMLR 2020
[5] “Amnesia” - A Selection of Machine Learning Models That Can Forget User Data Very Fast, CIDR 2020
[6] Cao, Yinzhi, and Junfeng Yang. "Towards making systems forget with machine unlearning." 2015 IEEE Symposium on Security and Privacy. IEEE, 2015.
[7] Ginart, Antonio, et al. "Making AI forget you: Data deletion in machine learning." Advances in Neural Information Processing Systems. 2019.

21. Enhancing Interoperability in Polystore Systems

Advisor: Haralampos Gavriilidis

Appropriate Target Level: B.Sc., M.Sc.

Keywords: data federation, interoperability, query languages, query optimization, polystores.

Description: Data stacks today are comprised of many data *storage* and *processing systems* [1]. To gain valuable insights, data scientists need to integrate and analyze all of the available data. To that end, data scientists either *implement custom data pipelines* [2] or *utilize existing hybrid analytic solutions* (i.e., *polystores*) [3-7]. Some of these approaches mirror concepts seen in federated databases [8,9], however, they go beyond the relational model. When composing custom data pipelines, users are forced to implement multiple queries and target them to each data management system (DMS) individually, which they subsequently need to assemble, and manually schedule. Although this approach enables each system to be exploited and optimized individually, to achieve optimal performance, it has some drawbacks. In particular, users need to be experts in all of the query languages and possess knowledge about the internals of each DMS. Additionally, users need to efficiently assemble the intermediate query results using hand-written ETL processes, which decreases the productivity and overall pipeline performance. To overcome these limitations, polystore systems offer middleware that mediates between users and multiple DMSs. Users would then implement declarative polystore queries and these would be transparently optimized and forwarded to each individual DMS. The unified interface frees users from having to query each source individually, and hence improves productivity. However, the performance of polystore queries largely depends on the *polystore optimization strategy* employed. Furthermore, since the expressivity of each polystore query language is limited, the set of supported polystore queries is equally limited.

Prerequisites: good programming skills in Java, Scala, and SQL; good writing skills; some experience in big data frameworks; (ideally) completed several DIMA courses (e.g., DBT, DBT-PRA); interest in compilers.

References

[1] Stonebraker, M., & Çetintemel, U. (2018). *"One size fits all" an idea whose time has come and gone*, Making Databases Work: The Pragmatic Wisdom of Michael Stonebraker (pp. 441-462).
[2] https://airflow.apache.org/
[3] Simitsis, A., Wilkinson, K., Castellanos, M., & Dayal, U. (2009, June). *QoX-driven ETL design: reducing the cost of ETL consulting engagements*, Proceedings of the 2009 ACM SIGMOD International Conference on the Management of Data (pp. 953-960).
[4] Xu, L., Cole, R. L., & Ting, D. (2019, July). *Learning to optimize federated queries*, Proceedings of the 2nd International Workshop on Exploiting Artificial Intelligence Techniques for Data Management @SIGMOD (pp. 1-7).

[5] Agrawal, D., Chawla, S., Contreras-Rojas, B., Elmagarmid, A., Idris, Y., Kaoudi, Z., ... & Papotti, P. (2018). *RHEEM: enabling cross-platform data processing: may the big data be with you!*, Proceedings of the VLDB Endowment, 11(11), 1414-1427.
[6] Duggan, J., Elmore, A. J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., ... & Zdonik, S. (2015). *The bigdawg polystore system*, ACM SIGMOD Record, 44(2), 11-16.
[7] Sethi, R., Traverso, M., Sundstrom, D., Phillips, D., Xie, W., Sun, Y., ... & Berner, C. (2019, April). *Presto: SQL on everything*, 2019 IEEE 35th International Conference on Data Engineering (ICDE) (pp. 1802-1813).
[8] Hsiao, D. K. (1992). Fe*derated databases and systems: part i—a tutorial on their data sharing*, The VLDB Journal, 1(1), 127-179.
[9] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., & Widom, J. (1994). *The TSIMMIS project: Integration of heterogeneous information sources*.

22. Efficiently Embedding High-level User-defined Functions in the NebulaStream IoT Data Management System

Advisor: Philipp Grulich

Appropriate Target Level: M.Sc.

Keywords: stream processing, query compilation, modern hardware, code generation, augmenting data processing, job efficiency using query compilation techniques

Description: Since modern data processing pipelines often employ many programming languages (e.g., Java, Scala, Python), it is paramount that a data processing engine efficiently execute high-level, user-defined functions (UDFs). Unfortunately, it is difficult to optimize UDFs given that they contain arbitrary code. To address this problem, GraalVM, a just-in-time compiler that generates both *machine code* as well as an *LLVM* [2] *intermediate representation* is proposed. In this thesis, you will evaluate *GraalVM* [1] as a runtime for data processing workloads and investigate alternative embedding techniques for the efficient combination of a C++ based query engine (e.g., in NebulaStream) and Graal.

Expected Outcomes:
- investigate novel ways to embed high-level UDFs into NebulaStream,
- prototype a connector to the GraalVM LLVM backend,
- evaluate the implementation and conduct a performance analysis.

Prerequisites: *excellent programming skills in Java and C++; good writing skills*; basic knowledge in compiler theory and LLVM; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA);

References

[1] GraalVM: A High-performance Multilingual Runtime, https://www.graalvm.org/.
[2] LLVM Compiler Infrastructure, http://llvm.org/.

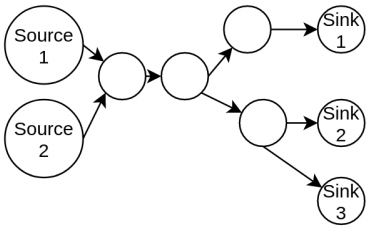
23. Code Generation for Complex Query Plans

Advisor: Philipp Grulich

Appropriate Target Level: M.Sc.

Keywords: stream processing, query compilation, modern hardware, code generation, augmenting data processing, job efficiency using query compilation techniques

Description: Code generation/query compilation is a well known technique that is used to improve the efficiency of modern data processing systems. In this thesis, this technique will be applied to *complex query graphs* (comprised of numerous source and sink operators), which arise in the optimization of merge query plans (e.g., in NebulaStream).



Expected Outcome:

- develop novel approaches to translate merge queries into code,
- devise optimization strategies based on the costs of queries,
- evaluate the implementation and conduct a performance analysis.

Prerequisites: *programming skills in C++*; *good writing skills*; *basic knowledge in data processing on modern hardware;* (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA).

References

[1] Grulich, Philipp M., et al. "Grizzly: Efficient Stream Processing Through Adaptive Query Compilation." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.
[2] Neumann, Thomas. "Efficiently compiling efficient query plans for modern hardware." Proceedings of the VLDB Endowment 4.9 (2011): 539-550.

24. Adaptive Query Execution in NebulaStream

Advisor: Philipp Grulich

Appropriate Target Level: M.Sc.

Keywords: stream processing, query compilation, modern hardware, code generation, augmenting data processing, job efficiency using query compilation techniques

Description: In the Grizzly paper [1], we proposed an adaptive query execution technique that enables a stream processing system to react to changes in data characteristics at runtime. In this thesis, we want to implement this technique in the NebulaStream system, and explore the design space of potential optimizations.

Expected Outcome:

- implement adaptive query compilation for the NebulaStream system,
- devise optimization strategies based on the runtime characteristics,
- evaluate the implementation and conduct a performance analysis.

Prerequisites: *programming skills in C++*; *good writing skills*; *basic knowledge in data processing on modern hardware;* (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA).

References

[1] Grulich, Philipp M., et al. "Grizzly: Efficient Stream Processing Through Adaptive Query Compilation." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

25. Data Stream Summarization Using Custom Hardware (FPGAs)

Advisor: Martin Kiefer

Appropriate Target Level: B.Sc., M.Sc.

Keywords: approximate data analysis using modern hardware, energy efficient computing

Description: My research is centered on reducing the time it takes to conduct data analysis. To achieve this, we seek to leverage modern hardware architectures and employ approximate computing techniques (e.g., data sketching [1]), which is faster and more efficient.

Power-efficient data analysis is an increasingly important problem in the era of big data: The amount of available data continues to increase exponentially and for economic and environmental reasons, we need to ensure that the energy demands required to analyze the data do not grow exponentially as well. In this thesis, we seek to analyze data streams using stream summarization techniques and custom hardware on FPGAs, while taking energy efficiency into consideration.

Prerequisites: *programming skills* in C/C++, OpenCL, VHDL, or Python; *good writing skills*; (preferably) interest in modern hardware; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, MHD).

References

[1] Graham Cormode. 2017. Data Sketching: The approximate approach is often faster and more efficient. In *ACM Queue*, Vol. 15, Issue 2.
https://queue.acm.org/detail.cfm?id=3104030

26. Improving Query Optimization Using Modern Hardware

Advisor: Martin Kiefer

Appropriate Target Level: B.Sc., M.Sc.

Keywords: approximate data analysis using modern hardware

Description: My research is centered on *reducing the time it takes to conduct data analysis*. To achieve this, we seek to leverage modern hardware architectures and employ approximate computing techniques (e.g., data sketching [1]), which is faster and more efficient.

The query optimizer is at the heart of state-of-the-art relational database systems. It derives different execution plans for a given query and selects the cheapest one based on statistics and a cost model. However, this has to be done in a very tight time budget, since query optimization delays query execution. In this thesis, we seek to leverage modern hardware and algorithmics to optimize queries more efficiently. In particular, using bandwidth-optimized kernel density models as learning selectivity estimators on GPUs, to improve the statistics available to the query optimizer.

Prerequisites: *programming skills* in C/C++, OpenCL, VHDL, or Python; *good writing skills*; (preferably) interest in modern hardware; ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

References

[1] Graham Cormode. 2017. Data Sketching: The approximate approach is often faster and more efficient. In *ACM Queue*, Vol. 15, Issue 2.https://queue.acm.org/detail.cfm?id=3104030

27. Scalable GPU Co-processing with Fast, Cache-coherent Interconnects

Advisor: Clemens Lutz

Appropriate Target Level: B.Sc., M.Sc.

Keywords: scalable GPU co-processing, cache-coherent interconnects

Description: GPUs and other modern processors are capable of very fast data processing. For example, a high-end Nvidia GPU is capable of reading from its built-in memory at a rate of 900 GB/s, and compute 14 trillion floating-point operations per second (i.e., 14 TFLOPS). This is more than 20 times faster than regular CPUs. Our aim is to leverage this increased processing power, in order to analyze data more efficiently. In particular, devising novel ways to *access data from a GPU more efficiently* or *execute a SQL JOIN operator faster*.

Prerequisites: *programming skills in* C/C++, CUDA, or OpenCL; *good writing skills*; performance-oriented programming of CPUs and GPUs; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, MHD).

28. Query Optimization in Secure Databases

Advisor: Kajetan Maliszewski

Appropriate Target Level: B.Sc., M.Sc.

Keywords: secure databases, query optimization, join selection, Intel SGX

Description: People are increasingly aware and concerned about the security and privacy of their data. Novel, secure hardware, such as Intel's Software Guard Extensions (SGX) [1] looks very promising. However, there are still some challenges to overcome. We have observed that join algorithms behave very differently in this new environment and as a result the query optimizer needs to be updated accordingly. In this thesis, you will focus on adapting the query optimizer for the Intel SGX [2] solution. In particular, devise new algorithms to choose the right execution plans for join queries. During the course of this thesis, you will gain a good understanding of join algorithms, their execution behavior across varying data input, and secure hardware as well as the low-level mechanisms of both CPUs and Intel SGX.

Prerequisites: *programming skills in C/C++*; *good writing skills*; basic understanding how CPUs and *main memory* work (e.g., cache, transaction lookaside buffer, memory control); knowledge about SQL joins; (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA).

References

[1] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. International Association for Cryptologic Research. https://eprint.iacr.org/2016/086.pdf.
[2] Intel® Software Guard Extensions. https://intel.ly/3bPhxaT.

29. Scaling Streaming Graph Neural Networks

Advisor: Serafeim “Makis” Papadias

Appropriate Target Level: M.Sc.

Keywords: machine learning, representation learning, streaming graphs

Description: Various real-world applications are modeled as *graphs*. For example, social networks are large graphs, where the nodes and edges represent the users and their friendships, respectively. Today, there is great interest in performing machine learning (ML) on graphs. A typical ML application is *link prediction*, which seeks to predict new friendships in social media. Many real-world graphs are inherently dynamic, i.e., new nodes and edges are constantly being added or existing ones deleted. Unfortunately, most *graph neural network* models are unable to harness dynamic data, which can boost the performance of many ML tasks. Recently, a new Dynamic Graph Neural Network (DGNN) was proposed by Ma et al. [1], which is capable of incorporating dynamic data. Although the proposed approach is effective, it is only suitable for small graphs. In this thesis, the goal is to develop a scalable, dynamic graph neural network model for data streams.

Prerequisites: strong *programming skills (e.g., Java)*; *good writing skills*; *knowledge in database systems and machine learning;* (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA).

References

[1] Yao Ma et al. Streaming Graph Neural Networks. In *SIGIR 2020.*

30. High Level Abstraction Layers for Data Processing on Heterogeneous CPU-GPU Architectures

Advisor: Dwi Prasetyo Adi Nugroho

Appropriate Target Level: M.Sc.

Keywords: GPU, query execution, programming interface

Description: To accelerate data processing, it is commonplace today to exploit both CPUs and GPUs. However, there is a drawback: CPUs and GPUs each employ their own programming framework. Unfortunately, this increases the complexity in developing a data processing platform. To mitigate this problem, a natural solution would be to reuse the same code base, and compile the code for both CPUs and GPUs. Recent studies in the HPC domain [1,2] have shown how existing high-level abstraction frameworks (e.g., SYCL[2], OpenCL[3], Kokkos[4]) can help. In this thesis, we will investigate the challenges and opportunities that high-level abstraction layers offer to accelerate data processing.

Prerequisites: *programming skills in C/C++*; *good writing skills*; basic understanding of GPU architectures and parallel execution; (ideally) completed several DIMA courses (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, MHD).

References

[1] Homerding, Brian, and John Tramm. “Evaluating the Performance of the hipSYCL Toolchain for HPC Kernels on NVIDIA V100 GPUs.” Proceedings of the International Workshop on OpenCL. 2020.
[2] Deakin, Tom, and Simon McIntosh-Smith. “Evaluating the performance of HPC-style SYCL applications.” Proceedings of the International Workshop on OpenCL. 2020.

[2] SYCL is a cross-platform abstraction layer that enables code for heterogeneous processors to be written using standard ISO C++ with the host and kernel code for an application contained in the same source file.
[3] OpenCL (Open Computing Language) is a framework for writing programs that execute across heterogeneous platforms consisting of CPUs and GPUs, among other processors or hardware accelerators.
[4] Kokkos is a templated C++ library that provides abstractions to allow a single implementation of an application kernel to run efficiently on different kinds of hardware, such as GPUs, Intel Xeon Phis, or many-core CPUs.

31. The Management of Data Science Processes

Advisor: Sergey Redyuk

Appropriate Target Level: B.Sc., M.Sc.

Keywords: machine learning pipelines, experiment databases, data science processes

Description: The development of modern data science pipelines is an iterative process that captures the specification of the data analytics to be executed. Among the components in these pipelines are the ability to *profile data*, support for *exploratory data analysis*, *data preprocessing* and *feature engineering* capabilities, as well as solutions for *model selection* and *hyperparameter tuning*, *performance evaluation*, *visualization*, and *reporting*. Many of the algorithms employed in these components operate under implicit assumptions that neither novice users nor domain experts are aware of. Moreover, frameworks that support pipeline abstractions (e.g., Scikit-learn, SparkML) typically do not provide optimization strategies for efficient pipeline execution. As a result, creating and executing data science pipelines (represented as complex graphs, comprised of *data manipulation* and *machine learning* operations) efficiently is challenging. Among the challenges to be overcome in the end-to-end management of data science processes, there are several thesis topics: (i) the *declarative specification of data science processes* [1] (e.g., to express complex data science processes in a unified way and enable the systematic comparison of processes and factor analysis), (ii) *experiment databases* [2] (e.g., that store previously executed data science pipelines and enable new ones to be improved), and (iii) the development of *optimization rules* [3], to enable the efficient execution of end-to-end data science pipelines.

Prerequisites: *programming skills in Python*; *good writing skills*; experience with the Python data science toolkit (e.g., pandas, numpy, scikit-learn, keras)*;* (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA); (ideally) *prior experience or completed coursework in machine learning* (e.g., TUB/ML Group: Machine Learning I, TUB/NI Group: Machine Intelligence I).

References

[1] S. Redyuk. Automated Documentation of End-to-End Experiments in Data Science. In Ph.D. Symposium track, IEEE 35th International Conference on Data Engineering (ICDE'19)
[2] Van Rijn et al. OpenML: A collaborative science platform. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases 2013 (pp. 645-649). Springer, Berlin, Heidelberg.
[3] Derakhshan et al. Optimizing Machine Learning Workloads in Collaborative Environments. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.

32. Large-Scale Machine Learning

Advisor: Alexander Renz-Wieland

Appropriate Target Level: B.Sc., M.Sc.

Keywords: large-scale machine learning, parameter servers

Description: It is advantageous to train large-scale machine learning (ML) models on a cluster (instead of a single machine), since it increases the amount of available compute power and memory. However, the drawback is that this approach requires communication among the cluster nodes, in order to synchronize model parameters. For some ML models, this synchronization step is the dominating part of the training process. In this case, it would not be prudent to increase the number of nodes, since this would result in a drop in the performance. To reduce the communication overhead, researchers have developed algorithms to create and exploit parameter locality (e.g., stemming from the training algorithm, ML model, or training data), where each worker updates a subset of the model parameters at a given point in time. These subsets typically change (i.e., workers update different subsets) throughout training process. ML developers typically need to implement locality-exploiting algorithms (LEAs) from scratch, which requires them to possess in-depth knowledge (i.e., low level details) about distributed computing environments. We are developing a system that enables researchers and practitioners to implement LEAs, without the need for detailed distributed computing knowledge. We seek to make parameter servers (the state-of-the-art architecture for distributed ML) usable and more efficient for LEAs. Several thesis topics in this subfield arise, including addressing system-related challenges as well as experimenting with the system along specific ML models.

Prerequisites: *programming skills in C++*; *good writing skills*; (ideally) *completed several DIMA courses* (e.g., AIM-3, BDAPRO, BDASEM, DBT, DBT-PRA, DBPRA, DBPRO, DBSEM, ISDA); (ideally) *prior experience or completed coursework in machine learning* (e.g., TUB/ML Group: Machine Learning I, TUB/NI Group: Machine Intelligence I) and distributed systems (e.g., TUB/DOS Group: Distributed Systems).

33. Data Processing on Heterogeneous CPU/GPU Systems

Advisor: Viktor Rosenfeld

Appropriate Target Level: B.Sc., M.Sc.

Keywords: exploiting GPU hardware for query processing, scheduling query processing tasks on heterogeneous processors

Description: GPUs are powerful co-processors that can be used to speed up query processing. In this context, we offer two thesis topics.

Topic 1: Recent dedicated GPUs contain specialized hardware cores to accelerate matrix computations. This new hardware makes GPUs interesting processors to exploit machine learning for query optimization. In this thesis, you would investigate *which query optimization tasks can exploit this hardware* effectively and *what benefit it provides on query performance*.

Topic 2: On CPUs, the state-of-the-art query processing models are *query compilation* and *vectorization*. They have comparable performance, however, which one is faster depends on the specific query. These processing models have been implemented on GPUs, but it is not yet clear, which one is faster. Due to the cache architecture and high launch overheads, vectorization is more difficult to efficiently implement on GPUs than on CPUs. In this thesis, you would investigate *how to implement vectorization efficiently* and *conduct a detailed comparison with query compilation*.

Prerequisites: *strong programming skills in C/C++, OpenCL, CUDA*; *good writing skills*; *interest in low-level programming*, *processor architectures;* (ideally) *completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA, MHD, DBPRA, DBPRO, DBSEM, ISDA).

34. Complex Event Processing in Distributed Stream Processing Systems

Advisor: Ariane Ziehn

Appropriate Target Level: M.Sc.

Keywords: large-scale complex event processing, stream processing, distributed systems

Description: Complex event processing (CEP) is a commonly employed for the real-time processing of data streams, to detect sequences of events and trigger actions, when a recognized pattern is detected. User-defined rules specify both the events and actions that enable autonomous real-time decision making in a wide range of existing and emerging applications (e.g., live maps, smart street lamps, vehicle pollution control). However, the state-of-the-art CEP engines are still mainly based on central architectures and serial processing models, which prevent low-latency and the real-time processing of big data streams. Given the ever-growing number of IoT devices, their *geographical distribution and environmental heterogeneity*, the limitations of CEP engines become even more critical. In this thesis, we aim to overcome bottlenecks prevalent in CEP engines, to enable large-scale CEP in distributed stream processing systems.

Note: I am on parental leave during the winter term 2020/2021. Nevertheless, we can discuss the topic and start preparing a proposal writing for a thesis start scheduled for summer 2021.

Prerequisites: *programming skills in C++*; *good writing skills*; *basic knowledge of distributed systems* (particularly, stream processing engines); (ideally) *prior experience or completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA).

References

[1] Giatrakos, Nikos, et al. "Complex event recognition in the big data era: a survey." *The VLDB Journal* 29.1 (2020): 313-352.
[2] Ziehn, Ariane. "Complex Event Processing for the Internet of Things." *fog* 1.3: 4. http://ceur-ws.org/Vol-2652/paper01.pdf.
[3] The NebulaStream System for the Management of IoT Data. https://www.nebula.stream/.

35. Standardized Complex Event Processing Benchmark for Big Data Platforms

Advisor: Ariane Ziehn

Appropriate Target Level: M.Sc.

Keywords: large-scale complex event processing, stream processing, distributed systems

Description: Benchmarks for complex event processing (CEP) do not yet exist, due to the absence of a widely accepted *formal framework* and *terminologies*. In order to evaluate the strengths and weaknesses of contemporary CEP solutions (e.g., FlinkCEP, synergies between the CEP engines of Apache Spark and Storm), a benchmarking platform, such as the Yahoo! Streaming Benchmark for analytical stream processing is required. In this thesis, the goal is to create a CEP benchmarking platform prototype and evaluate leading CEP engines. To meet this objective, you will need to *review previous efforts in the benchmarking of CEP* (e.g., Section 3 of the CEP survey [1]), *design a formal framework for a selection of common pattern classes*, and *identify relevant measurements*, such as [2] for stream processing.

Note: I am on parental leave during the winter term 2020/2021. Nevertheless, we can discuss the topic and start preparing a proposal writing for a thesis start scheduled for summer 2021.

Prerequisites: *programming skills in C++*; *good writing skills*; *basic knowledge of distributed systems* (particularly, stream processing engines) and benchmarking; (ideally) *prior experience or completed several DIMA courses* (e.g., BDAPRO, BDASEM, DBT, DBT-PRA).

References

[1] Giatrakos, Nikos, et al. "Complex event recognition in the big data era: a survey." *The VLDB Journal* 29.1 (2020): 313-352.

[2] Karimov, Jeyhun, et al. 2018. Benchmarking Distributed Stream Data Processing Systems. *arXiv preprint arXiv:1802.08496*.

[3] Raasveldt, Mark, et al. 2018. Fair benchmarking considered difficult: Common pitfalls in database performance testing. *Proceedings of the Workshop on Testing Database Systems*.