ODBD 2023

Ontario DataBase Day – Program

December 13, 2023

Program Overview

Start Time	End Time	Talk Information	*page*
09:15	09:30	Welcome/Opening	
09:30	09:50	Update-Aware Information Extraction *Besat Kassaie*, University of Waterloo	2
09:50	10:10	Towards Efficient and Reliable Data Curation for Machine Learning *Naiqing Guan*, University of Toronto	3
10:10	10:30	Towards Next-Generation Question Answering Over Knowledge Graphs Systems via Accurate Benchmarking and Large-Scale Training *Abdelghny Orogat*, Carleton University	4
10:30	10:50	Quantum-safe Blockchain–Data security perspective *Ajmery Sultana*, Algoma University	5
		Coffee break	
11:20	11:40	Optimizing Recursive Joins in Graph Database Management Systems *Anurag Chakraborty, University of Waterloo*	6
11:40	12:00	Do Programming Languages need Query Languages? *Jelle Hellings*, McMaster University	7
12:00	12:20	Eventually Durable Replicated State Machines *Kriti Kathuria and Kenneth Salem, University of Waterloo*	8
12:20	12:40	Trajectory Data Mining in the Age of Big Data and AI *Manos Papagelis*, York University	9
		Lunch	
13:40	14:00	sGradd: Towards RELIABLE S.t..r...e....ami..ng Graph Analytics *Aida Sheshbolouki*, University of Waterloo	10
14:00	14:20	GaussDB Evolution and Research Directions *Ronen Grosman and Yao Wu*, Huawei Canada	11
14:20	14:40	Incremental Computations of Connectivity Queries in Sliding Windows over Streaming Graphs *Chao Zhang*, University of Waterloo	12
14:40	15:00	Explanation Scores in Data Management *Leopoldo Bertossi*, SKEMA Business School, Montreal, Canada	13
		Coffee break	
15:30	15:50	Math Information Retrieval using a Conventional Search Engine *Frank Wm. Tompa*, University of Waterloo	14
15:50	16:10	Scaling Storage Engines for 100x Big Data *Niv Dayan*, University of Toronto	15
16:10	16:30	Bringing the Power of Object Stores to SAP IQ *Güneş Aluç*, SAP Labs	16
16:30	16:50	Pythia: A Neural Model for Data Prefetching *Akshay Arun Bapat*, University of Toronto	17
		Reception	

Update-Aware Information Extraction

Besat Kassaie*

David R. Cheriton School of Computer Science
University of Waterloo
bkassaie@uwaterloo.ca

Abstract

Information extraction programs (extractors) can be applied to documents to isolate structured versions of some content by creating tabular records corresponding to facts found in the documents. Most optimization techniques deployed in information extraction systems assume that source documents are static. Instead, extracted relations can be considered to be materialized views defined by a language built on regular expressions. Using this perspective, we provide an efficient verifier that can be used to avoid the high cost of re-extracting information after a batch update. In particular, we propose an efficient mechanism to identify updates for which we can autonomously compute an extracted relation. We present experimental results that support the practicality of this mechanism in real world extraction systems.

1 Introduction

When extracted relations or source documents are updated, we wish to ensure that those changes are propagated correctly. That is, we recommend that extracted relations be treated as materialized views over the document database. Within this context, we tackle two research challenges; I) Because extraction is prohibitively expensive, efficiently maintaining extracted relations up-to-date is crucial [5, 4]. II) To maintain system consistency, it is essential to translate updates on extracted views into corresponding document updates [3].

In this talk, I start by exploring update-aware information extraction, shedding light on the aforementioned critical issues that arise when dealing with updates. Next, I delve into our research on autonomously computable information extraction. Additionally, I highlight key findings from our experimental results for this problem, demonstrating whether our approach can be used effectively in realistic update and extraction scenarios.

*This talk is based on joint work with Frank Wm. Tompa: fwtompa@uwaterloo.ca

2 Autonomously Computable Information Extraction

We apply static analysis to programs that specify extractors and updates in order to determine whether re-extraction can be avoided or reduced. Given a program defined as a document spanner [2] and an update specification, we determine sufficient conditions for autonomously re-computing extracted spans of an updated document. In particular, we propose three sufficient conditions for updates with respect to an extraction program. We prove that we require time and space that are polynomial in the size of the extraction program and the update specification to perform five required tests to determine that the revised extracted relation can be computed autonomously. Finally, we describe experiments with realistic extractors conducted on two real-world datasets to conclude that the runtime overhead imposed by our verification is small in practice when compared to re-evaluating extractors, even if the re-evaluation is performed incrementally [1].

References

[1] Fei Chen, AnHai Doan, Jun Yang, and Raghu Ramakrishnan. Efficient information extraction over evolving text data. In *Proc. 24th ICDE*, pages 943–952. IEEE Computer Society, 2008.

[2] Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Document spanners: A formal approach to information extraction. *J. ACM*, 62(2):12:1–12:51, 2015.

[3] Besat Kassaie and Frank Wm. Tompa. Predictable and consistent information extraction. In *Proc. DocEng '19: ACM*, pages 14:1–14:10. ACM, 2019.

[4] Besat Kassaie and Frank Wm. Tompa. A framework for extracted view maintenance. In *Proc. DocEng '20: ACM*, pages 16:1–16:4. ACM, September 2020.

[5] Besat Kassaie and Frank Wm. Tompa. Autonomously computable information extraction. *Proc. VLDB Endow.*, 16(10):2431–2443, 2023.

Towards Efficient and Reliable Data Curation for Machine Learning

Naiqing Guan

Department of Computer Science
University of Toronto
naiqing.guan@mail.utoronto.ca

1 Introduction

Modern machine learning models require large training datasets to achieve good accuracy, yet manual labelling and curation of large datasets are both expensive and time-consuming. Thus, acquiring labelled datasets has become one of the main bottlenecks in applying machine learning in practical scenarios. This has motivated researchers to investigate approaches to reduce annotation costs and instigated corporate activity on labelling services.

The programmatic weak supervision (PWS) framework [Ratner et al.(2016), Ratner et al.(2017), Zhang et al.(2022)] provides an approach to automatically label large datasets without manually annotating specific instances. In the PWS framework, users represent weak supervision sources in the form of label functions (LFs), which are programs that provide noisy labels to a subset of data. Since the label functions have varying accuracy and may exhibit ad-hoc correlations, a label model is designed to aggregate noisy, weak labels into probabilistic labels. The aggregated labels are then used to train the downstream model.

While the PWS framework reduces annotation costs, it still has some limitations. First, the design of LFs requires substantial endeavours and domain expertise, while automatic LF design is still challenging. Secondly, the labels generated by the PWS framework are usually noisy, which deteriorates the performance of downstream models. How to evaluate, control and improve the quality of LFs and generated labels requires further investigation.

My research aims to improve the efficiency and reliability of data curation for machine learning, with a focus on the PWS framework. In this presentation, I will discuss two of my recent works in this direction, focusing on automatic LF design and improving label quality, respectively.

2 Presentation Outline

In the first part of the presentation, I will describe the background for efficient data curation and introduce the PWS framework.

In the second part, I will describe two of my recent works in enhancing the efficiency and reliability of the PWS framework. I will first introduce DataSculpt [Guan et al.(2023)], which automatically designs LFs by prompting large language models. We explored an expansive design landscape in DataSculpt and identified the strengths and limitations of contemporary LLMs in LF design. Then I will briefly describe ActiveDP, which combines PWS with active learning [Settles(2012)] to combine the strengths of both paradigms and improve the label quality.

In the third part, I will describe the limitations of the current PWS framework and propose some future research directions in this area.

References

[Guan et al.(2023)] Naiqing Guan, Kaiwen Chen, and Nick Koudas. 2023. Can Large Language Models Design Accurate Label Functions? arXiv:2311.00739 [cs.CL]

[Ratner et al.(2017)] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 269.

[Ratner et al.(2016)] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems* 29 (2016), 3567–3575.

[Settles(2012)] Burr Settles. 2012. Active learning. *Synthesis lectures on artificial intelligence and machine learning* 6, 1 (2012), 1–114.

[Zhang et al.(2022)] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2022. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433* (2022).

Towards Next-Generation Question Answering Over Knowledge Graphs Systems via Accurate Benchmarking and Large-Scale Training

Abdelghny Orogat
School of Computer Science
Carleton University

Abstract

Knowledge graphs (KGs) serve as pivotal information reservoirs, but their intricate structures and the need for expertise in structured query languages limit their accessibility mainly to experts. While the vast potential of KGs in Question Answering (QA) systems is acknowledged, their complexity often requires users to possess specific and advanced querying skills.

Since 2010, the advent of QA systems has ushered in a new era where users can employ natural language for queries, simplifying the process. The creation of numerous benchmarking datasets aids in training these modern QA systems, yet choosing the ideal dataset remains challenging. Current datasets, like *QALD*, come with their own set of limitations, emphasizing the urgency to refine them as KGs gain prominence in sectors such as healthcare and finance. Notably, these benchmarks often lack quantitative comparisons and employ diverse creation methods. Additionally, while knowledge graphs continually evolve, benchmarks remain static, failing to capture the latest updates and nuances.

In light of these challenges, our research offers a comprehensive solution for both benchmarking and training QA systems that are specifically optimized for KGs. Central to our approach is the introduction of CBench [4, 1]. CBench is an innovative, extensible benchmarking suite devised to provide in-depth analyses of existing QA benchmarks. It delves into the intricate linguistic, syntactic, and structural attributes of the questions and queries contained within these benchmarks. Our assessments revealed notable variations across benchmarks in relation to these properties, making the selection of a mere subset of them an unreliable practice for QA system evaluation. CBench stands out by providing not just conventional metrics but also a granular analysis of the linguistic and structural attributes of both answered and unanswered questions. This detailed insight empowers QA system developers by highlighting areas where their systems excel and pinpointing where improvements are needed.

Furthermore, our research introduces Maestro [3, 2], an avant-garde benchmark generation system specifically tailored for QA over KGs. Maestro is equipped to produce a benchmark for any KG, provided the KG itself and, if available, a relevant text corpus that encompasses the KG's domain. The benchmarks crafted by Maestro are exhaustive, encapsulating all salient properties of natural language questions and structured queries documented in existing literature, given that the targeted KG embodies these properties. A standout feature of Maestro is its ability to generate superior-quality natural language questions with diverse phrasings, rivaling those generated manually, thereby ensuring a more robust evaluation of QA systems.

Future Work

Building on Maestro's ability to generate annotated questions and queries, there's potential for integration with large language models (LLMs). By utilizing Maestro's output, including questions, annotated counterparts, and structured queries, we aim to train an LLM to create a superior QA system. The combination of Maestro's detailed benchmarks and LLM capabilities promises to set new standards in the QA domain over KGs.

References

[1] A. Orogat and A. El-Roby. CBench: Demonstrating Comprehensive Evaluation of Question Answering Systems over Knowledge Graphs Through Deep Analysis of Benchmarks. *Proceedings of the VLDB Endowment (PVLDB)*, 14(12), 2021.

[2] A. Orogat and A. El-Roby. SmartBench: Demonstrating Automatic Generation of Comprehensive Benchmarks for Question Answering over Knowledge Graphs. *Proceedings of the VLDB Endowment (PVLDB)*, 15(12), 2022.

[3] A. Orogat and A. El-Roby. Maestro: Automatic generation of comprehensive benchmarks for question answering over knowledge graphs. *Proceedings of the ACM on Management of Data*, 1(2):1–24, 2023.

[4] A. Orogat, I. Liu, and A. El-Roby. CBench: Towards Better Evaluation of Question Answering Over Knowledge Graphs. *Proceedings of the VLDB Endowment (PVLDB)*, 14(8), 2021.

Quantum-safe Blockchain- Data security perspective

Ajmery Sultana, Assistant Professor

Department of Computer Science
Algoma University

1 Description of the presentation

Blockchain technology consists of a distributed ledger that operates through a decentralized network of data blocks, sequentially connected and regulated by consensus mechanisms [1]. Initially developed to underpin cryptocurrencies like Bitcoin, broader business and technological sectors now recognize blockchains' potential applicability across various fields [2], including healthcare [3], communication [4], and smart grids [5]. Blockchains currently rely on established cryptographic techniques to maintain security. However, the emergence of quantum computing is shifting the security landscape, as some of the current encryption methods may be compromised by the power of quantum processors [6]. Therefore, the adoption of advanced encryption protocols within the realm of post-quantum cryptography is becoming imperative. This presentation will delve into the latest advancements in blockchain methods that are fortified by post-quantum cryptography. It will highlight quantum-proof blockchain models tailored for diverse platforms and use-cases, addressing security challenges and offering remedies. Additionally, it will chart out avenues for future exploration in this cutting-edge area.

References

[1] W. Wang, Y. Yu, and L. Du, "Quantum blockchain based on asymmetric quantum encryption and a stake vote consensus algorithm," Scientific Reports, 12(1), 8606, 2022.

[2] J. J. Bambara, and P. R. Allen, "Blockchain. A practical guide to developing business, law and technology solutions," New York City: McGraw-Hill Professional, 2018.algorithm," Scientific Reports, 12(1), 8606, 2022.

[3] I. Yaqoob, K. Salah, R. Jayaraman, and Y. Al-Hammadi, "Blockchain for healthcare data management: opportunities, challenges, and future recommendations," Neural Computing and Applications, pp. 1-16, 2021.

[4] L. Zhang, K. Cheng, Y. Xu, and H. Zhu, "A General Access Architecture for Blockchain-Based Semi-Quantum 6G Wireless Communication and its Application," International Journal of Theoretical Physics, 61(4), 109, 2022.

[5] B. Khan, I. Ul Haq, S. Rana and H. Ul Rasheed, "Secure Smart Grids: Based on Post-Quantum Blockchain," 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, pp. 653-658, 2022.

[6] E. Karacan, S. Akleylek, and A. Karakaya, "PQ-FLAT: a new quantum-resistant and lightweight authentication approach for M2M devices", IEEE 9th International Symposium on Digital Forensics and Security (ISDFS), Elazig, Turkey, pp. 1-5, 2021.

Optimizing Recursive Joins in Graph Database Management Systems

Anurag Chakraborty

David R. Cheriton School of Computer Science
University of Waterloo
a8chakra@uwaterloo.ca

1 Abstract

Recursive joins such as shortest, all shortest, and variable length path queries are a core feature provided by graph database management systems. However, these queries are computationally expensive for large datasets and frequently suffer from high execution time due to skew from a few nodes being highly connected.

Existing work on efficiently executing these queries proposes using a morsel-driven parallelism [1] (MDP) approach for bulk path finding. The MDP approach assigns fixed-size "morsels" to threads, which comprise a set of starting nodes from which the path traversal needs to be started. This does not address the skew problem since it does not enable multiple threads to work on the same recursive join computation at the same time.

This presentation will focus on efficiently executing recursive join queries by integrating a hybrid parallelization scheduler into a GDBMS [2] query pipeline. We will examine the scheduler's design within our recursive join operator, morselized workload distribution between worker threads, query execution plans, the tradeoffs between various approaches such as MS-BFS [3] and direction optimizing BFS [4] and when to trigger which approach. We will also explore the use of lock free data structures used internally by the operator to handle query cases such as returning path length or returning graph paths to support the concurrent progress of multiple recursive join computations and experimental evaluation results obtained on large graphs (LDBC, LiveJournal, Graph500).

References

[1] D. ten Wolde, T. Singh, G. Szárnyas, and P. A. Boncz, "DuckPGQ: Efficient Property Graph Queries in an Analytical RDBMS," in Proceedings of the 13th Conference on Innovative Data Systems Research (CIDR 2023)

[2] X. Feng, G. Jin, Z. Chen, C. Liu and S. Salihoğlu, "Kùzu Graph Database Management System," in Proceedings of the 13th Conference on Innovative Data Systems Research (CIDR 2023)

[3] M. Then, M. Kaufmann, F. Chirigati, T. HoangVu, K. Pham, A. Kemper, T. Neumann and Huy T. Vo, "The More the Merrier: Efficient Multi-Source Graph Traversal," VLDB 2014

[4] S. Beamer, K. Asanovic and D. Patterson, "Direction-Optimizing Breadth-First Search," SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis

Do Programming Languages need Query Languages?

Jelle Hellings

Department of Computing and Software
McMaster University
1280 Main Street West, Hamilton, ON, Canada

1 Introduction

Data processing plays a central role in many general-purpose programs. This central role is underlined by the functionality included in standard support libraries provided by modern programming languages. Take, for example, the C++ standard library: this library includes several efficient *data structures* to represent data collections and a plethora of *algorithms* to operate on these data collections. Indeed, the algorithms in the C++ standard library can even be used to perform all elementary relational algebra operations. Furthermore, the recently added `<ranges>` functionality even allows for the high-level expression of data processing operations via *views*.

To illustrate a high-level data processing task using *views*, consider the following program that *queries* for parents of children living in Hamilton:

```
using namespace std::views;

auto where_pred = [](auto l)
    { return l.place == "Hamilton"; };
auto product_pred = [](auto t)
    { auto [po, p] = t;
      return po.child == p.name; };

for (auto [po, p] : cartesian_product(
            parents,
            persons | filter(where_pred)) |
                filter(product_pred)) {
    std::cout << po.parent << std::endl;
}
```

Similar functionality exists in most major programming languages, e.g., LINQ in C# and other .NET languages, Streams in Java, and list comprehensions in Python.

Although these data processing functionalities do provide the ability to express complex queries, they do not guarantee performance: it is up to the programmer to ensure an efficient program. A programmer can do so by selecting the proper ways to structure, store, and maintain the data; the proper operations to perform on the data; and the most efficient order of these operations. In the example provided above, the programmer made several poor decisions, e.g., by excessive copying and by performing a potentially-expensive *Cartesian product*.

The attention to detail required for a performant data processing program in C++ is in sharp contrast to how any database system with a high-level query language operates. For example, in a Datalog-based database system, one could express the above query via the query:

```
Result(parent) :- parents(parent, c),
                  persons(c, "Hamilton").
```

The above Datalog query is significantly simpler than the C++ program provided before. Furthermore, it is highly likely that the database system will produce a query evaluation strategy that is close to optimal (e.g., by using any indices available on parents and persons) and much more efficient than the approach expressed by the C++ program.

2 Problem Statement

Not all data processing tasks happen in an environment in which a database system is available. Hence, shifting data processing tasks toward database systems for efficiency reasons is not always an option.

As an alternative, we propose a support library via which one can *embed* high-level database-like data abstractions and queries within the program (as-if these where any ordinary data structure or algorithm). In specific, our support library embeds support for Datalog queries and for specifying data structures that manage relational data (including primary key and foreign key constraints) into C++.

A crucial part of our approach is the development of a *compile-time query optimizer* than can produce highly-efficient query evaluation algorithms given only the information available when compiling source code: the queries itself and the data structures on which these queries will be evaluated. By performing query optimization *once* at compile-time, we are able to apply the *zero-cost principle* within our proposed library by eliminating any unnecessary overheads and, hence, provide performance close to (or even surpassing) carefully-crafted data processing algorithms.

Eventually Durable Replicated State Machines

Kriti Kathuria, Ken Salem

University of Waterloo
first.last@uwaterloo.ca

Consider a replicated key-value (KV) store as an example of a replicated state machine. It consists of a key-value store at each site and a replicated log. The interface it exposes is *put(key, value)* and *get(key)*. *put* creates an entry in the replicated log, and once the entry has achieved majority replication, creates/updates the key-value pair. *get* returns the latest value of the key. The KV store guarantees that if a *put* succeeds, a subsequent *get* will see its effect regardless of failures. This write durability can be guaranteed because the *put* does not return until it has achieved majority replication. Therefore, durability is an expensive guarantee.

Hence, latency-sensitive applications may choose to forgo durability for better performance in an ad-hoc manner, like acknowledging put operations without waiting for them to fully replicate. We present Eventually Durable (ED) replicated state machines and establish a principled approach for the applications to reason about performance/durability tradeoffs they already make.

Like a regular KV store, an ED KV store also consists of a key-value store at each site and a replicated log. It also exposes a *put* and a *get* interface. *get* works like in the regular KV store. *put*, on the other hand, does not guarantee durability when it returns. A put may become durable eventually, after it returns. In the meantime, the application may proceed under the assumption that the put will eventually become durable. That is, the application can speculate on the eventual durability of the put operation.

Additionally, the ED KV store provides a *sync* operation which returns after all preceding puts, and as a result, values read by preceding gets, have become durable. *sync* is a tool for the application to resolve speculation and thus, manage the risks associated with durability speculation.

Applications accept the risk that the ED KV store will lose some acknowledged puts in the event of a failure. The ED model provides clear failure semantics that allow applications to reason about possible data loss. Specifically, the ED model guarantees that a failure will result in the resolution of all existing speculation — the speculative entries that survive the failure will become durable and will survive forever. The entries that get lost will never reappear. In this way, there will only ever be a single speculative "future" for the ED KV store.

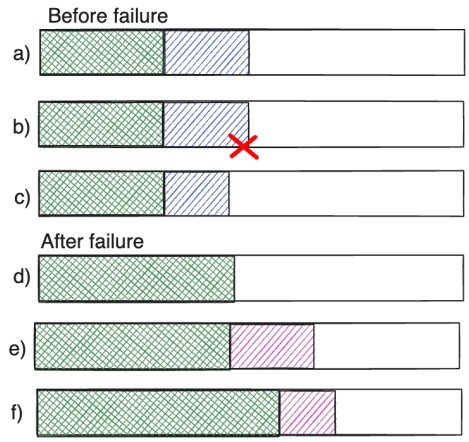

Figure 1: Behaviour of the replicated log

Fig.1 illustrates the behavior ED KV store's underlying replicated log across a failure. Green depicts the durable portion of the log and blue/pink depicts the speculative portion. (a, b, c) show the log before a failure, and (d, e, f) show the log after the failure. At first, the log contains some durable entries and some speculative entries (a). Then there is a failure (b) due to which some speculative entries disappear (c). After the failure, the surviving pre-failure speculation has become durable (d). The application starts speculating again (e) and as time progresses, a prefix of the new speculative entries is made durable, and more speculative entries are added (f).

To support an ED replicated state machine, we have developed an ED variant of the Raft consensus algorithm[1]. ED Raft acknowledges new log proposals without waiting for replication. Failures lead to the loss of speculative log proposals. We show that ED Raft supports the failure semantics described above.

References

[1] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, page 305–320, USA, 2014. USENIX Association.

Trajectory Data Mining in the Age of Big Data and AI

Manos Papagelis

Electrical Engineering and Computer Science
Lassonde School of Engineering
York University
papaggel@eecs.yorku.ca

1 Introduction

Trajectory Data. *Trajectory data* consists of records that capture the movement of objects or entities over time, typically provided in the form of triplets $\langle o, t, (x, y)\rangle$, representing that an object o at timestamp t was at location with coordinates (x, y). The proliferation of location-based technologies, geo-enabled smart devices, and advanced global positioning systems has resulted in the accumulation of vast amounts of trajectory data.

Trajectory Data Mining. *Trajectory data mining* focuses on extracting valuable patterns, information, and insights from trajectory data, and it has been an active research direction for a long time [1, 2, 3]. Mining interesting patterns and extracting useful information from trajectories can find application in diverse domains, including intelligent transportation systems, urban planning and environmental monitoring, and public health. Due to the broader impact, several trajectory-related research problems have been of interest, including trajectory similarity, prediction, clustering, classification, simplification, outlier detection, and imputation.

2 Our Research Contributions

Over the last years, we have been revisiting classic trajectory data mining problems through the lens of modern technologies and methods, including *big data analysis*, *graph mining / network analysis*, and *deep learning methods*. Our journey includes methods for:

- **Trajectory simplification** (ACM SIGSPATIAL '23): We presented PathletRL, a deep reinforcement learning method for identifying a small set of trajectory building blocks, known as *pathlets*, that can compactly represent a vast number of trajectories.
- **Generating higher-order trajectory data** (ACM SIGSPATIAL '23). We presented Point2Hex, a method and tool for generating higher-order mobility flow datasets from raw trajectory data.
- **Trajectory-user linking** (IEEE MDM '23): We presented tulhor, a Transformer-based model that links anonymous trajectories to the respective users.
- **Trajectory network analysis** (IEEE MDM '18; GeoInformatica, 23, '19; IEEE BigData '18; IEEE MDM '20): We presented methods for modeling interactions of moving objects using graphs, including methods for (*i*) trajectory node centrality computation and for (*ii*) mining pedestrian group dynamics.
- **Transportation Optimization** (ACM SIGSPATIAL '22, ACM SIGSPATIAL '22): We presented methods for (*i*) forecasting the performance of road intersections, and for (*ii*) the vehicle navigation problem.
- **Mobility and Epidemics** (ACM SIGSPATIAL/SpatialEpi '23, ACM SIGSPATIAL/SpatialEpi '23, IEEE MDM '22): We presented methods for modeling epidemic spreading in mobility networks.

3 Presentation Structure

In this presentation we will delve into our recent endeavors on trajectory data mining, emphasizing its contemporary and evolving nature that stems from the incorporation of novel deep learning methods for addressing long-established problems. Depending on time, we will present our recent work on trajectory-user linking, a trajectory classification problem aimed at connecting anonymous trajectories to their respective users (IEEE MDM '23). We will also present our recent work on trajectory dictionary construction, which aims at constructing a trajectory pathlet dictionary (ACM SIGSPATIAL '23).

References

[1] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. Spatio-temporal data mining: A survey of problems and methods. *ACM Comp. Surveys*, 51(4), Aug 2018.

[2] Ali Hamdi et al. Spatiotemporal data mining: a survey on challenges and open problems. *Artificial Intelligence Review*, 55(2):1441–1488, Feb 2022.

[3] Yu Zheng. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3), May 2015.

*s*Gradd: Towards RELIABLE S.t..r...e....ami..ng Graph Analytics

Aida Sheshbolouki

David R. Cheriton School of Computer Science
University of Waterloo
aida.sheshbolouki@uwaterloo.ca

1 Transient Concepts in Graphs

The continuous generation of relations among entities and the growing need to run queries over them necessitate Streaming Graph Management Systems (SGMS, such as *s-graffito*[1]). These systems deal with dynamic and high velocity and volume of the data arrivals while generating reliable outputs. SGMS generates unreliable outputs when input data is new, temporal, incomplete, or adversely manipulated and the system does not recognise and manage it properly [2, 3]. A main cause identified for this problem is Concept Drift (CD), which occurs when a change in a hidden context induces changes in a target concept [4]. Understanding, detecting, and adaptation to CD in streaming data is (i) challenging due to stateful and blocking operations, and (ii) impactful in a variety of practical scenarios [5].

The literature is mostly focused on black-box drift detection and adaptation integrated within supervised learning systems, assume independence of data instances, and the target concepts defined as class labels. These assumptions and design choices do not always work. In this talk, I will discuss the challenges of designing a CD detection framework and introduce *sGradd*, a streaming graph framework for drift ddetection.

2 Challenges

CD in data streams is commonly considered as a change in underlying probability distribution of data points, which are generated independently [1]. However, streaming graph records are usually interconnected and dependent. Moreover, current definition implies detection based on comparing data distribution over window samples using fixed size sliding windows. When the streaming rate is highly dynamic with significant rises, adaptive window sizes are more efficient. Thus, we need a CD definition to enable any detection solution in streaming graphs.

The challenges are designing an unsupervised CD detection method and also an effective performance evaluation. Real-world streaming data with drift labels are not easy to acquire. Moreover, the accuracy and latency of the detection are tightly bounded together and different drift patterns require different examinations. I will discuss how we approached these challenges and what tools and techniques we developed to address them.

[1] dsg-uwaterloo.github.io/s-graffito/

3 sGradd

I will introduce *sGradd*, which detects drift in the generative sources. This detection is the first operation when data becomes available in input monitor of SGMS. *sGradd* has two main components: one for data management and the second for drift detection constructed based on multidisciplinary techniques. I will explain how *sGradd* ingests data, updates analytic primitives, performs CD detection, and streams out drift signals with descriptions about their occurrence to inform the next analytics.

References

[1] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proc. 30th Int. Conf. on Very Large Data Bases*, pages 180–191, 2004.

[2] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Trans. Knowl. and Data Eng.*, 31(12):2346–2363, 2019.

[3] Navid Malekghaini, Elham Akbari, Mohammad A Salahuddin, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, and Stephane Tuffin. Data drift in dl: Lessons learned from encrypted traffic classification. In *2022 IFIP Networking Conf. (IFIP Networking)*, pages 1–9, 2022.

[4] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[5] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In *Big Data Analysis: New Algorithms for a New Society*, pages 91–114. 2016.

GaussDB Evolution and Research Directions

Ronen Grosman, Yao Wu

Huawei Canada
ronen.grosman@huawei.com,wuyao18@huawei.com

1 Introduction to GaussDB

GaussDB is a distributed relational database from Huawei. It supports intra-city cross-AZ deployment [1] with zero data loss. With a distributed architecture, GaussDB supports petabytes of storage and contains more than 1,000 nodes per DB instance. It is highly available, secure, and scalable and provides services including quick deployment, backup, restoration, monitoring, and alarm reporting for enterprises. The overall architecture of a distributed instance of GaussDB is as Figure 1, which is a share-nothing distributed cluster with log replication HA. GaussDB is built on top of openGauss [1], which is a multi-core-oriented open-source relational database that provides ultimate performance, full-link service and data security, AI-based tuning [2], and efficient O&M capabilities. This leading database at enterprise level is developed in collaboration with global partners and is released under the Mulan Permissive Software License v2. openGauss deeply integrates Huawei's years of R&D experience in the database field and continuously builds competitive features based on enterprise-level scenario requirements.

2 Challenges and Opportunities of GaussDB

Huawei is actively working with industry and ecosystem partners worldwide, promoting joint innovation among businesses, academia, research institutes, and users to create practical value based on the needs of different industries. We promote cross-domain and cross-technology collaboration in various forms, in order to tackle real-world problems that different industries face. The openGauss kernel was derived from PostgreSQL and focused on building advance features for architecture, transactions and storage engines with performance optimization. It is deeply optimized for ARM architecture and retains compatibility with x86 architecture. In this presentation, we will cover the main challenges and opportunities for openGauss and GaussDB, e.g., disaggregated architecture, hybrid data processing, unified storage engine [3]. Huawei will continue to work with developers and partners in building the openGauss community for a prosperous global ecosystem.

[1] https://docs.opengauss.org/en

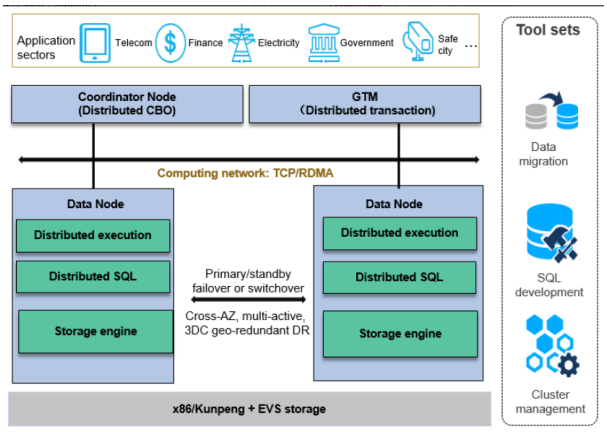

Figure 1: GaussDB Architecture

3 Open Discussions

In this section, we will discuss a few open challenges from an industry perspective towards the database community [4].

References

[1] Weixing Zhou, Qi Peng, Zijie Zhang, et al. Geogauss: Strongly consistent and light-coordinated oltp for geo-replicated sql database. *Proceedings of the ACM on Management of Data*, 1(1):1–27, 2023.

[2] Guoliang Li, Xuanhe Zhou, Ji Sun, et al. opengauss: An autonomous database system. *Proceedings of the VLDB Endowment*, 14(12):3028–3042, 2021.

[3] Hillel Avni, Alisher Aliev, Oren Amor, et al. Industrial-strength oltp using main memory and many cores. *Proceedings of the VLDB Endowment*, 13(12):3099–3111, 2020.

[4] Guoliang Li, Haowen Dong, and Chao Zhang. Cloud databases: New techniques, challenges, and opportunities. *Proceedings of the VLDB Endowment*, 15(12):3758–3761, 2022.

Incremental Computations of Connectivity Queries in Sliding Windows over Streaming Graphs

Chao Zhang

David R. Cheriton School of Computer Science
University of Waterloo
chao.zhang@uwaterloo.ca

1 Introduction

Graphs have been the natural representation of data in many domains. With graph structured data, the most interesting operation is to compute *connected components* (CCs) [4], which are subsets of vertices in a undirected graph such that all vertices in the subset are connected via paths. Analyzing CCs has wide applications in practice, including social networks, transport networks, etc.

In modern data-driven applications, stream processing [5] is of significant importance. In stream processing, computations are typically applied in *sliding windows* [1] that are continuous finite subsets of streams over the infinite input stream. Sliding windows are defined using two parameters *range* and *slide*. For instance, a sliding window with range 3 hours and slide 2 minutes includes all the streaming data of the last 3 hours and the window is updated every 2 minutes, *i.e.*, deleting expired streaming data and inserting new streaming data.

The naive approach to compute sliding window connectivity is to traverse the streaming graph in each window instance of the sliding window, *e.g.*, performing breadth-first-search (BFS) in each window instance. Apparently, the naive approach cannot meet the requirement of real-time processing, which asks for high-through and low-latency computations. A non-trivial method is to use the well-known fully dynamic connectivity (FDC) data structures [2, 3]. Specifically, FDC supports 3 operations: `insert`, `delete`, and `query`. Obviously, the `insert` and `delete` operations supported by FDC can be used to deal with the updates required by sliding windows. The main bottleneck of the FDC approach is that the `delete` operation can have high latency as it requires traversing the entire graph in the worst case.

We design the bidirectional incremental computation (BIC) model to efficiently compute sliding window connectivity, which can reduce the problem of sliding window connectivity into a bidirectional computation. The main idea of BIC is that (i) streaming edges with contiguous timestamps are grouped to form disjoint chunks; (ii) window instances are split according to chunks; (iii) queries are processed by applying partial computations in chunks followed by merging the corresponding partial results. Specifically, we compute two kinds of buffers for each chunk: *forward* and *backward* buffers. Forward buffers are computed incrementally by scanning streaming edges *from the first to the last* in chunks while backward buffers are computed in the same way except that streaming edges are scanned *from the last to the first* in chunks. These two kinds of buffers are stored and merged to compute the query result of each window instance. Consequently, the overhead of performing the costly `delete` operation can be completely avoided.

In this presentation, we will elucidate the intricacies of incremental computations within the *forward* and *backward* buffers, as well as expound upon the merging operation. Our work is ongoing, and in addition to detailing our approach, we will also present preliminary experimental results in comparison to state-of-the-art methods based on FDC data structures.

References

[1] L. Golab and M. T. Özsu. Issues in data stream management. *ACM SIGMOD Rec.*, 32(2):5–14, jun 2003.

[2] M. R. Henzinger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, 1999.

[3] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

[4] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu. The ubiquity of large graphs and surprising challenges of graph processing: extended survey. *VLDB J.*, 29(2):595–618, 2020.

[5] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy. Storm@twitter. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, page 147–156, 2014.

Explanation Scores in Data Management

Leopoldo Bertossi*

SKEMA Business School Canada
Montreal, Canada
leopoldo.bertossi@skema.edu

1 Introduction

In data management, artificial intelligence (AI), and machine learning (ML) in particular, one wants *explanations* for certain results. For example, for query answers in databases (DBs). In ML, one wants explanations for automated classification results. Explanations that are based on *numerical scores* assigned to elements of a model that may contribute to an outcome have become popular. These *attribution scores* attempt to capture the quantitative degree of relevance of a tuple to a query answer; or a of a feature value to the label assigned to an entity.

In this presentation, we will survey some of the recent advances on the definition, use and computation of score-based explanations for query answering in DBs, and some extensions for ML. Special emphasis is placed on the use of counterfactual reasoning for score specification and computation. This presentation is heavily influenced by our recent research.

2 Explanation Scores

Different scores have been proposed in the literature. Among them we find the *responsibility score* as found in *actual causality* [7, 6], where the notion of *counterfactual intervention* is fundamental. In data management, responsibility, in the form of a *Resp*-score has been used to quantify the strength of a tuple as a cause for a query result [9, 3].

Database repairs are common when dealing with inconsistent DBs [2]. Connections between repairs and actual causality in DBs has been useful to obtain complexity and algorithm results for responsibility [3]. On the basis of database repairs, a measure (or global score) to quantify the degree of inconsistency of a DB has also been introduced.

The *Resp* score has to be generalized to deal with non-binary features in ML [4], which could also be used to define a fine-grained responsibility in DBs at the attribute level. The *causal-effect score* has also been defined and applied to explain query answers in DBs [10].

The Shapley value of *coalition game theory* can be used to define attribution scores in DBs [8, 5]. Since *several tuples together*, much like players in a coalition game, are necessary to produce a query result, some may contribute more than others to a *game function* represented by the query result.

The Shapley value has also been used to define explanation scores to feature values in ML-based classification. Since its computation is intractable in general, tractable classes of models have been identified [1].

*Prof. Emeritus, Carleton University, Ottawa, Canada

References

[1] Arenas, M., Barcelo, P., Bertossi, L. and Monet, M. On the Complexity of SHAP-Score-Based Explanations: Tractability via Knowledge Compilation and Non-Approximability Results. *Journal of Machine Learning Research*, 2023, 24(63):1-58.

[2] Bertossi. L. *Database Repairing and Consistent Query Answering*. Synthesis Lectures in Data Management. Morgan & Claypool, 2011.

[3] Bertossi, L. and Salimi, B. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. *Th. Comp. Sys.*, 2017, 61(1):191-232.

[4] Bertossi, L., Li, J., Schleich, M., Suciu, D. and Vagena, Z. Causality-Based Explanation of Classification Outcomes. Proc. 'Data Management for End-To-End Machine Learning', DEEM WS at SIGMOD 2020.

[5] Bertossi, L., Kimelfeld, B., Livshits, E. and Monet, M. The Shapley Value in Database Management. *ACM Sigmod Record*, 2023, 52(2):6-17.

[6] Chockler, H. and Halpern, J. Y. *Resp*onsibility and Blame: A Structural-Model Approach. *Journal of Artificial Intelligence Research*, 2004, 22:93-115.

[7] Halpern, J. and Pearl, J. Causes and Explanations: A Structural-Model Approach. Part I: Causes. *The British J. Phil. of Sci.*, 2005, 56(4):843-887.

[8] Livshits, E., Bertossi, L., Kimelfeld, B. and Sebag, M. The Shapley Value of Tuples in Query Answering. *Log. Methods Comput. Sci.*, 2021, 17(3).

[9] Meliou, A., Gatterbauer, W., Moore, K. F. and Suciu, D. The Complexity of Causality and Responsibility for Query Answers and Non-Answers. Proc. VLDB, 2010, pp. 34-41.

[10] Salimi, B., Bertossi, L., Suciu, D. and Van den Broeck, G. Quantifying Causal Effects on Query Answering in Databases. Proc. TaPP, 2016.

Math Information Retrieval using a Conventional Search Engine

Frank Wm. Tompa

David R. Cheriton School of Computer Science
University of Waterloo
fwtompa@uwaterloo.ca

1 Introduction

Documents in the STEM disciplines rely heavily on the use of math formulas to express knowledge. Searching a corpus of such documents, therefore, requires that a search engine be effective in matching formulas and math terminology, as well as natural language text.

In our presentation, we describe a simple representation for features extracted from math formulas, our implementation of a prototypical search engine, and performance results against benchmarks created to evaluate math-aware search engines for community question-answering.

This research is being conducted in collaboration with Andrew Kane (PhD 2014, Waterloo).

2 Further Details

Effective math information retrieval has been under investigation by several students who have worked under my direction [3, 1, 5] resulting in a best paper award [2] and a best of labs designation [6]. Notably, this last paper describes how natural language mathematical questions can be automatically transformed into formal queries consisting of keywords and formulas and how the resulting formal queries can be effectively executed against a corpus. A key component of our approach has been to represent each formula as a bag of math features and to treat those features as simple search terms. This approach can be adopted by any conventional, text-based search engine, including one developed recently to explore various aspects of search technology.[1]

We describe the three major processing steps used in our system:

1. query construction (how to convert natural language questions into formal queries by selecting and augmenting the text and formulas),
2. mapping formal queries to search terms (especially, how to choose suitable features to represent math formulas), and
3. indexing and querying with the search engine (how to run queries efficiently and rank results effectively).

We also summarize some of our experimental results from the ARQMath Labs [4], a benchmark based on a collection of questions and answers from Math Stack Exchange (MSE) between 2010 and 2018 consisting of approximately 1.1 million question-posts and 1.4 million answer-posts. The main task presents experimenters with 100 mathematical questions (selected by the organizers from MSE question-posts in a subsequent year) and asks for ranked lists of potential answers among existing answer-posts in the collection.

Finally, we outline what research we intend to undertake to improve each of the three processing steps.

[1] https:// github.com/andrewrkane/mtextsearch

References

[1] Dallas J. Fraser. Math information retrieval using a text search engine. Master's thesis, University of Waterloo, Cheriton School of Computer Science, 2018.

[2] Dallas J. Fraser, Andrew Kane, and Frank Wm. Tompa. Choosing math features for BM25 ranking with Tangent-L. In *DocEng 2018*, pages 17:1–17:10, 2018.

[3] Shahab Kamali. *Querying Large Collections of Semistructured Data*. PhD thesis, University of Waterloo, Cheriton School of Computer Science, 2013.

[4] Behrooz Mansouri, Vit Novotný, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. Overview of ARQMath-3 (2022): Third CLEF lab on Answer Retrieval for Questions on Math. In *CLEF 2022*, volume 13390 of *LNCS*. Springer, 2022.

[5] Yin Ki Ng. Dowsing for math answers: Exploring MathCQA with a math-aware search engine. Master's thesis, University of Waterloo, Cheriton School of Computer Science, 2021.

[6] Yin Ki Ng, Dallas J. Fraser, Besat Kassaie, and Frank Wm. Tompa. Dowsing for math answers. In *CLEF 2021*, volume 12880 of *LNCS*, pages 201–212, 2021.

Scaling Storage Engines for 100x Big Data

Niv Dayan

Department of Computer Science
University of Toronto
nivdayan@cs.toronto.edu

1 Introduction

Our society is creating and storing exponentially increasing amounts of data. What is often less thought of are the storage engines that maintain this data and facilitate the extraction of knowledge from it. In the late-2000s, a new class of storage engines emerged that prioritize the efficiency of ingesting new data. They include Google's BigTable, Amazon's DynamoDB, Facebook's RocksDB, as well as Apache Cassandra and HBase. These engines have become indispensable for a wide range of applications, including cloud storage, blockchain, machine learning, etc. Nevertheless, a lingering problem is that their performance deteriorates with respect to the amount of data they store. This, in turn, causes applications running on top to have to spend disproportionately more time, energy, and hardware in order to, say, transact on a blockchain, train a deep learning model, or add a photo to the cloud. This talk will discuss how to allow such storage engines to function more efficiently as the big data that they store continues to grow.

2 Background

LSM-Tree. Modern storage engines streamline new application data into storage (disk or SSD) as small sorted files, which are later merged into larger sorted files. This organization is known as a log-structured merge-tree (LSM-tree). With LSM-tree, merging files more eagerly creates higher overheads for writes but allows for faster queries as there are fewer files to search. This trade-off is controlled by a compaction policy, which dictates which files to merge under which conditions.

Filters. Each file of an LSM-tree is assigned a "filter" in fast memory (DRAM chips). A filter is a compressed approximate representation of a file that takes up little space. Filters can be quickly searched to rule out files that do not contain the target data. Thus, they eliminate unnecessary accesses to slower storage. The more space a filter is assigned, the more accurate it becomes thus allowing queries to rule out the file with a higher probability. An LSM-tree implements a filtering policy to decide how much memory to assign each filter. Together, the filtering and compaction policies govern a three-way trade-off between the overheads of queries, writes and space

3 Research Problem

As with most tree structures, LSM-tree's query and write overheads grow logarithmically with respect to the data size. The intuition is that as the data grows, the number of files that must be queried and merged grows too. In our current era of exponential data growth, logarithmic scalability implies linearly increasing overheads with respect to time. The outcome is rapidly deteriorating performance. While it is possible to offset one of the overheads growing by another (e.g., by merging more eagerly or allocating larger filters to prevent query overheads from increasing), it is impossible with existing designs to keep all three overheads steady at the same time as the data grows. This begs a question: is it possible to achieve sub-logarithmic query and write costs for an LSM-tree, all without hurting space?

4 Talk Content

This talk will discuss a series of papers that tackle the problem of how to better scale performance as the data grows in the context of LSM-trees. The talk can be given at different lengths, from 15 minutes to an hour, depending on the amount of time available. It will cover at least two and at most all of the following papers: Monkey (SIGMOD 2017), Dostoevsky (SIGMOD 2018), LSM-bush (SIGMOD 2019), Rosetta (SIGMOD 2020), Chucky (SIGMOD 2021), and Spooky (VLDB 2022). These papers show how to co-design the LSM tree's compaction policy with its filters in ways that lead to asymptotic improvements in both query and insertion throughput, meaning that performance deteriorates more slowly or not at all as the data grows. This talk will conclude with a vision towards amorphous storage engines and data structures, which self-design to optimize any application workload.

Bringing the Power of Object Stores to SAP IQ

Güneş Aluç

SAP Labs, Canada
gunes.aluc@sap.com

1 Introduction

In this presentation, we describe our journey of transforming SAP IQ into a relational database management system (RDBMS) that utilizes cheap, elastically scalable object stores in the cloud [3, 4]. SAP IQ is a three-decade old, disk-based, columnar RDBMS that is optimized for complex online analytical processing (OLAP) workloads. Traditionally, SAP IQ has been designed to operate on shared storage devices with *strong consistency guarantees* (e.g., high-caliber storage area network devices). Therefore, deploying SAP IQ on the cloud, as is, would have meant utilizing storage solutions that provide a POSIX compliant file interface and strong consistency guarantees, but at a much higher monetary cost. These costs can accumulate easily to diminish the economies of scale that one would expect on the cloud, which can be undesirable. Instead, we have enhanced the design of SAP IQ to operate on cloud object stores such as AWS S3 [1] and Azure Blob Storage [2]. Object stores rely on a weaker consistency model, and potentially have higher latency; however, because of these design trade-offs, they are able to offer (i) better pricing, (ii) enhanced durability, (iii) improved elasticity, and (iv) higher throughput. By enhancing SAP IQ to operate under these design trade-offs, we have unlocked many of the opportunities offered by object stores. Experiments using the TPC-H benchmark demonstrate that we can gain an order of magnitude reduction in data-at-rest storage costs while improving query and load performance.

2 Contributions

When developing the cloud-native version of SAP IQ, we have decided to exploit the strengths of the product as much as possible and avoid reinventing the wheel. In particular, SAP IQ benefits from three decades of research and development when it comes to techniques such as data compression, partitioning, indexing, prefetching and loading that we wanted to exploit. Consequently, in this presentation, we discuss the following aspects of our design:

1. SAP IQ makes a clear distinction between the logical and the physical representation of pages in the system; therefore, we directly map logical pages to objects in object stores;
2. We enforce a “never write an object twice" policy in the transaction and buffer managers to handle the weaker consistency model used in object stores;
3. We implement techniques for efficiently allocating object keys in a multi-node setting and discuss the lessons we have learnt in working with object stores; and lastly
4. We discuss the design considerations of the Extended Cache Manager (ECM), that acts as a read/write cache between the existing buffer manager and the object store [4].

References

[1] Amazon Simple Storage Service (S3). http://aws.amazon.com/s3/.

[2] Azure Blob Storage. https://azure.microsoft.com/en-us/services/storage/blobs/.

[3] M. Abouzour, G. Aluç, I. T. Bowman, X. Deng, N. Marathe, S. Ranadive, M. Sharique, and J. C. Smirnios. Bringing cloud-native storage to SAP IQ. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2410–2422. ACM, 2021.

[4] S. Shedge, N. Sharma, A. Agarwal, M. Abouzour, and G. Aluç. An extended SSD-based cache for efficient object store access in SAP IQ. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, pages 1861–1873. IEEE, 2022.

Pythia: A Neural Model for Data Prefetching

Akshay Arun Bapat

University of Toronto
akshay.bapat@mail.utoronto.ca

1 Instance Optimized Database

Traditional databases are often general-purpose software systems that are typically not built for a specific workload or a data distribution. In general, these systems might provide good performance but probably not the optimal one. Recently, there has been a push towards leveraging machine learning based techniques to build database systems that are self-tuned or instance-optimized [4, 5, 3]. So far, it has achieved significant success in cardinality estimation and query optimization leading to faster query execution. However, there has been limited effort on optimization opportunities that are one level deeper, namely how to tailor the RDBMS buffer management module for correlated query workloads.

Buffer management [2, 1] has played a central role in improving RDBMS performance. Such improvement for a single query is achieved by exploiting locality of reference for page accesses and for multiple queries by enabling possible sharing of frequent page requests across queries. Both of these improvements can be pronounced if one can predict the access patterns of a query fairly accurately. An effective buffer manager ensures that the pages that are likely to be requested in the near future are prefetched in the buffer pool so that they do not result in costly blocked I/O operations later. Since predicting future access patterns is a challenging problem, RDBMSs employ empirical algorithms based on frequency and recency [2, 1]. While these approaches work well for simple access patterns, they often fail for more diverse and complex access patterns inherent in a typical OLAP setting.

2 Predicting query access patterns

Access patterns in databases are complex and challenging to predict. The access patterns of a complex SQL join query involving multiple relations are influenced by numerous factors such as selectivity (index scan or sequential scan), the join algorithm used (nested loop vs hash join), the order in which the relations are joined and so on. The use of indexes also violate sequential access patterns and results in *irregular sequences* due to the interleaving of accesses for index and base table pages.

Our empirical analysis shows that NLP techniques do not work well as the access patterns are too irregular (thereby having limited temporal patterns) but also very long (such as millions of tokens for a large relation). They also have distributional properties (frequency, co-occurrence and length of sequences) that are inimical to NLP based approaches. Additionally, they also require significant time for both training and inference.

`PYTHIA` is a deep ML predictive model tailored to predictions of the access patterns of complex correlated query workloads. It consists of two key components – a predictor and a prefetcher. Given a query, the goal of the predictor is to accurately output the relevant page accesses. The prefetcher then *asynchronously* fetches these pages and places them in the buffer pool.

We conduct extensive experiments with `PYTHIA` integrated into Postgres and demonstrate that it achieves significant accuracy and a speedup of upto 6x for queries in the DSB OLAP benchmark.

References

[1] Hong-Tai Chou and David J DeWitt. An evaluation of buffer management strategies for relational database systems. *Algorithmica*, 1(1-4):311–336, 1986.

[2] Wolfgang Effelsberg and Theo Haerder. Principles of database buffer management. *ACM Transactions on Database Systems (TODS)*, 9(4):560–595, 1984.

[3] Guoliang Li and Xuanhe Zhou. Machine learning for data management: A system view. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 3198–3201. IEEE, 2022.

[4] Guoliang Li, Xuanhe Zhou, and Lei Cao. Ai meets database: Ai4db and db4ai. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2859–2866, 2021.

[5] Dimitris Tsesmelis and Alkis Simitsis. Database optimizers in the era of learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 3213–3216. IEEE, 2022.