RESEARCH STATEMENT

Lin Ma (lin.ma@cs.cmu.edu)

Database management systems (DBMSs) are a critical component in almost every modern data-intensive application. But these systems are hard: they are hard to configure, hard to optimize, and hard to maintain because they have complex functionalities that are needed to support large datasets and complicated queries. A successful DBMS deployment demands careful management of resource allocations, knob configurations, and physical design choices. It is impossible for humans to tailor each system based on the corresponding application. The problem has only became worse because the proliferation of the cloud has made the number of databases in the world increase significantly in the last decade.

A DBMS that autonomously manages itself and removes the need for human administration has been the "holy grail" of database research in the last 50 years. But nobody has achieved to build such a system yet. The work in this area has either been advisory tools that make recommendations to humans or semi-automated cloud services that are reactionary to workload shifts and focus on one system aspect at a time. These methods do not enable full autonomy because they cannot take a comprehensive viewpoint of the administration process like a human. With the recent advancement of machine learning (ML), artificial intelligence (AI), and hardware technologies, we now have the opportunity to build such autonomous DBMSs that approach or even surpass human abilities.

My research focuses on *designing new database system architectures for autonomous operations leveraging ML and AI to remove the human administration impediments*. A real autonomous DBMS needs to control all system aspects while also be reliable and maintainable. My approach combines both (1) principled architecture designs that ensure system efficiency and extensibility and (2) ML algorithms that are best suited for the particular sub-problem. As my work on a new self-driving DBMS shows, a modularized system architecture facilitated by ML enables efficient and holistic control while generating explainable actions. But existing ML algorithms may not always satisfy the practical requirements of autonomous DBMS operations. Thus, it is also necessary to better understand how to apply ML techniques appropriately and innovate new algorithms and models, as my collaboration with Microsoft Research demonstrates.

I now discuss my current research projects that validate my vision to combine ML techniques with innovative system design to support autonomous DBMS operations. I then present the future directions that I want to pursue.

Self-Driving Database Systems

I spent the majority of my PhD time on developing a **self-driving DBMS** [11], called **NoisePage** [2]. My design of NoisePage's architecture takes inspiration from self-driving vehicles. A simplified self-driving car architecture consists of (1) a perception system, (2) mobility models, and (3) a decision-making system. The perception system observes the vehicle's surrounding environment and estimates the potential state, such as other vehicles' movements. The mobility models approximate a vehicle's behavior in response to control actions. Lastly, the decision-making system uses the perception and the models' estimates to select actions to accomplish the driving objectives. NoisePage's self-driving architecture consists of three frameworks with analogs to self-driving cars: (1) workload forecasting, (2) behavior modeling, and (3) action planning. The forecasting system is how the DBMS observes and predicts the application's future workload. The DBMS then uses these forecasts with its behavior models to predict the impact of self-driving actions (e.g., changing knobs, creating indexes) relative to the target objective function (e.g., latency, throughput). The DBMS's planning system selects actions that improve this objective function. I now further discuss each of these frameworks.

Self-driving DBMSs should automatically choose when to apply which actions without any human intervention. Achieving this ability requires predicting the workload in the future so that the system can determine the proper time window to make changes. For example, the system should schedule expensive changes (e.g., creating an index) when the workload volume is low to avoid affecting normal workload operations. However, forecasting database workloads is challenging because modern DBMS applications may execute millions of queries per day. These queries can have varying patterns (e.g., daily transactional-analytical cycles), and building forecasting models for all of them can be expensive. I developed a framework to succinctly forecast the workload for self-driving DBMSs, called **QueryBot 5000** (QB5000) [1]. QB5000 continuously clusters queries based on their arrival rate temporal patterns and seamlessly handles different workload patterns and shifts. It then builds an ensemble of time-series forecasting models to predict query clusters' various arrival patterns. The key advantage of our approach over previous forecasting methods is that the data we use to train our models is independent of the hardware and the database design. Thus, the DBMS does not have to rebuild the models if its hardware or configuration settings change. The results that I published in SIGMOD demonstrate that QB5000 can efficiently forecast the expected future workload with minimal accuracy loss [5].

A self-driving DBMS also needs models to predict and explain the behavior of potential actions (e.g., changing knobs, creating an index) given the forecasted workload for its decision-making. This is similar to how self-driving cars use mobility models to estimate the effect of their actions, such as turning the steering wheel. Existing DBMS modeling techniques are either (1) "clear-box" analytical methods that are difficult to migrate to a new DBMS and require onerous redesign under system updates, or (2) "opaque-box" ML methods that are expensive to train and difficult to generalize across workloads (e.g., when the database size changes). I proposed a behavior modeling framework, called **ModelBot 2** (MB2) [8], that decomposes a DBMS's internal architecture into small, independent operating units (OUs) (e.g., building a hash table, flushing log records). MB2 then automatically selects the best ML method to train an OU-model for each OU that predicts its runtime, resource consumption, and performance impact for the current DBMS state. Compared to common opaque-box methods that use a monolithic model for the entire DBMS, these OU-models have smaller input dimensions, require less training time, and provide performance insight to each DBMS component. MB2 also provides a principled method for data generation and training that allows the system to use the same set of OU-models trained offline for any dataset or workload online. Compared to existing DBMS modeling techniques, our results published in SIGMOD show that MB2 is up to 25× more accurate in predicting the DBMS performance for a given state, especially providing better generalization across varying workloads. These OU-models also provide explanations on the self-driving actions' cost and benefit expectations.

Lastly, a self-driving DBMS needs to determine the sequence of actions to apply given the workload forecasts and behavior predictions to complete the autonomous optimization. The action sequence may span short and long horizons, and finding the best sequence among various potential actions over these horizons is an expensive constrained optimization problem with an exponential search space. The action planning also relies on the behavior models to estimate the actions' impact. And these model inferences may incur high overhead, which poses more challenges to the planning efficacy. I developed an action planning framework for self-driving DBMSs, called **PilotBot 0** (PB0), leveraging control theory and recent advances in AI. PB0 uses the receding horizon control scheme to plan an action sequence optimized for varying horizons. It also adapts the Monte Carlo tree search (MCTS) method to approximate the complex optimization problem. Furthermore, PB0 uses a two-level caching design to accelerate the model inference and reduce the planning cost. Our initial results show that PB0 chooses actions that result in up to 55% better performance than existing planning methods for DBMSs [7].

Developing a self-driving architecture from the ground up is a challenging system engineering process [3, 12]. In addition to leading the team at CMU working on the above projects, I have also been involved in many other facets of NoisePage's development. These projects include a JIT-compiled query engine [9, 10], a unified transaction management framework [14], and an indexing technique with a low memory footprint [13]. I believe these experiences have prepared me with valuable system development and engineering skills to pursue my future research goals.

Other Projects

ML-enhanced Cloud Database Systems

Despite the promising results of using ML to enhance DBMS performances, one common challenge that limits the practicality of these ML methods is that they may not generalize well when the DBMS deploys them on a new workload. In many database applications, the data distribution seen after deployment may differ from that of the training data due to the system complexity and workload diversity, which can result in huge prediction errors. In collaboration with Microsoft Research, I developed a method to address this challenge for cloud DBMSs by collecting additional training data for the target workload encountered in deployment. My approach leverages the cloud's ability to create a database copy and collect new labels on the copy to improve the model generalization. The key insight is to narrow down the data collection space to the prediction tasks requested during the deployment and use active learning techniques to choose the best labels to collect with minimal overhead. I developed a novel active learning algorithm, called **Holistic Active Learner** (HAL) [6], tailored to the unique requirements of learning tasks in ML-enhanced databases: robust, cost-sensitive, and batch-friendly. I evaluated HAL using workload data from the Microsoft Azure SQL database. The results that I published in SIGMOD show that HAL reduces the prediction error of ML models on the target workload by 75% only using about 100 additionally collected labels.

Modern Storage Hardware for In-Memory Database Systems

I have also explored methods to utilize modern storage hardware for in-memory DBMSs. In-memory DBMSs outperform disk-oriented systems on many workloads because they eliminate legacy components that inhibit performance, such as buffer pool management. But the speedup comes with the restriction that the database needs to fit in memory. To overcome this limitation, some in-memory DBMSs can move cold data out of volatile DRAM to secondary storage. Although there have been several implementations proposed for this type of cold data storage, there has not been a thorough evaluation of the design decisions in implementing this technique. Such decisions include policies for when to evict tuples and how to bring them back when they are needed. Furthermore, the performance characteristics of the storage device can impact the effectiveness of these policies. In collaboration with Intel Labs, I explored cold-data-storage policies for an in-memory DBMS with five different storage technologies, including the new Optane persistent memory storage device provided by Intel. Our results published in DAMON shows that tailoring the policy for the storage technology's characteristics improves system throughput by up to 3× over a generic configuration [4].

Future Research

I want to continue to investigate how to make autonomous DBMSs reach the ability of humans and satisfy the scalability needs of modern DBMS deployments. I am also interested in developing automated methods for discovering new system optimization techniques that are beyond what humans can devise. For example, recent AI systems have made winning moves that "no human ever would" in Go games and generated hypotheses that human experts have not thought about in cancer reoccurrence. Achieving such goals will likely involve collaborative innovations from many areas of computer science, such as databases, machine learning, software engineering, and human-computer interaction (HCI).

Robustness and Scalability of Autonomous Database Systems

My previous work has shown the feasibility of building a single-node self-driving DBMS. There are still open research problems on how to make these autonomous systems more robust for real-world deployments. For example, it would be desirable to monitor errors (e.g., inaccurate model predictions) in the deployment and incorporate the feedback into the system's control. However, there are many design decisions that require further investigation, such as how to determine the feedback granularity and how to properly combine the knowledge in the previous models with this new information.

Cloud platforms provide DBMSs the opportunity to scale resources on demand. Thus, it is beneficial if a self-driving DBMS can automatically perform such scaling and control multiple nodes together. The workload forecasting and behavior modeling frameworks that I developed provide the necessary information for the system to make such decisions considering all system aspects holistically. But coordinating the decisions from all nodes in a distributed DBMS can result in an action search space orders of magnitude larger than the individual ones. It remains an unsolved challenge how the nodes in a distributed DBMS should reach a collective decision and what the proper system architecture should be.

Lastly, I want to explore the role of humans in future intelligent software systems. Traditional DBMS metrics, such as page access statistics and transaction status summaries, may not be sufficient to communicate to a human about the optimization decision made by the system. Likewise, it is unclear how a self-driving DBMS should incorporate directions from humans in its decision-making logic. Properly addressing such challenges requires collaboration between the database and HCI communities.

System Architectures for New Pricing Models

I am also interested in designing DBMS architectures that account for the new pricing models of the database services in the cloud. Existing DBMSs try to optimize performance metrics (e.g., throughput or latency) by using all the hardware resources available. But there are new pricing models for database services, especially for the software-as-a-service and serverless applications in the cloud: the cloud provides virtually unlimited resources, and vendors charge users by the system's resource consumption, such as IOPS, CPU time, and memory size.

These new models bring interesting challenges for DBMSs: the system will need to support various optimization goals, such as reducing the query latency while minimizing the monetary cost based on the pricing model. Furthermore, there are exciting opportunities to revisit the canonical database techniques and develop new DBMS architectures tailored for these resource-centered pricing models. I believe we can improve the system efficiency with less cost by jointly considering the system architecture design and autonomous optimizations facilitated by ML.

Automated Optimizations for Data Pipelines

DBMSs are not the center of the universe anymore. Modern data-intensive applications are built as pipelines comprised of many data services. For example, an application may need multiple ingestion services to acquire data from different sources. It might also need storage and processing services to combine, clean, normalize, and transform this data. Lastly, there can be different consumer services of the data, such as business intelligence and ML applications. Collectively optimizing the systems of all the services in a data pipeline may have significant performance benefits, but it is extremely difficult for humans to reason about all these systems' interactions with various configurations.

Given this, I am interested in exploring automated techniques that co-optimize the systems across a data pipeline. This is a challenging task since these systems have independent behavior without a unified architecture. I want to investigate the interaction pattern among these systems and create a global self-driving platform for data pipelines that leverages succinct modeling abstractions, powerful ML algorithms, and efficient architecture designs.

References

[1] QueryBot 5000. https://github.com/malin1993ml/QueryBot5000, 2019.

[2] NoisePage. https://noise.page/, 2021.

[3] Matthew Butrovich, Wan Shen Lim, Lin Ma, John Rollinson, William Zhang, Yu Xia, and Andrew Pavlo. Tastes great! less filling! high performance and accurate training data collection for self-driving database management systems. *Under submission to SIGMOD*, 2021.

[4] Lin Ma, Joy Arulraj, Sam Zhao, Andrew Pavlo, Subramanya R Dulloor, Michael J Giardino, Jeff Parkhurst, Jason L Gardner, Kshitij Doshi, and Stanley Zdonik. Larger-than-memory data management on modern storage hardware for in-memory oltp database systems. In *Proceedings of the 12th International Workshop on Data Management on New Hardware*, page 9, 2016.

[5] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J Gordon. Query-based workload forecasting for self-driving database management systems. In *Proceedings of the 2018 International Conference on Management of Data*, pages 631–645, 2018.

[6] Lin Ma, Bailu Ding, Sudipto Das, and Adith Swaminathan. Active learning for ml enhanced database systems. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 175–191, 2020.

[7] Lin Ma, William Zhang, Jie Jiao, Matthew Butrovich, Wan Shen Lim, and Andrew Pavlo. Pb0: Randomized action planning for self-driving database management systems. *In preparation for submission*, 2021.

[8] Lin Ma, William Zhang, Jie Jiao, Wuwen Wang, Matthew Butrovich, Wan Shen Lim, Prashanth Menon, and Andrew Pavlo. Mb2: Decomposed behavior modeling for self-driving database management systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1248–1261, 2021.

[9] Prashanth Menon, Amadou Ngom, Lin Ma, Todd C Mowry, and Andrew Pavlo. Permutable compiled queries: dynamically adapting compiled queries without recompiling. *Proceedings of the VLDB Endowment*, 14(2):101–113, 2020.

[10] Amadou Ngom, Prashanth Menon, Matthew Butrovich, Lin Ma, Wan Shen Lim, Todd C Mowry, and Andrew Pavlo. Filter representation in vectorized query execution. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)*, pages 1–7, 2021.

[11] Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C Mowry, Matthew Perron, Ian Quah, et al. Self-driving database management systems. In *Conference on Innovative Data Systems Research*, 2017.

[12] Andrew Pavlo, Matthew Butrovich, Lin Ma, Prashanth Menon, Wan Shen Lim, Dana Van Aken, and William Zhang. Make your database system dream of electric sheep: towards self-driving operation. *Proceedings of the VLDB Endowment*, 14(12):3211–3221, 2021.

[13] Huanchen Zhang, David G Andersen, Andrew Pavlo, Michael Kaminsky, Lin Ma, and Rui Shen. Reducing the storage overhead of main-memory oltp databases with hybrid indexes. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1567–1581, 2016.

[14] Ling Zhang et al. Everything is a transaction: Unifying logical concurrency control and physical data structure maintenance in database management systems. In *Conference on Innovative Data Systems Research*, 2021.