Index Tuning with Machine Learning on Quantum Computers for Large-Scale Database Applications

Le Gruenwald[1,*], Tobias Winker[2], Umut Çalıkyılmaz[2], Jinghua Groppe[2] and Sven Groppe[2]

[1] *The University of Oklahoma, USA*

[2] *Institute of Information Systems (IFIS), University of Lübeck, Germany*

Abstract

Selecting appropriate index configurations that can minimize query processing costs is essential to database applications. This problem becomes more complicated when the database is replicated on multiple nodes. There have been divergent design index tuning algorithms utilizing either heuristic or optimization methods to solve this NP-Hard problem. However, this index selection problem becomes much more complex for large-scale replicated database applications and efficient algorithms are needed. While quantum computing has been investigated with promising results in several areas of database management, such as query optimization and transaction scheduling, no work exists that studies the divergent design index tuning problem for replicated databases. To fill this gap, in this paper, we provide our vision of a machine learning-based quantum divergent design index tuning algorithm for replicated databases. We first discuss the issues that should be handled when designing such an algorithm. We then describe an algorithm for classical computers that has been shown to perform better than other existing algorithms and present our vision of how to transform the algorithm to its quantum version.

Keywords

Quantum Computing, Index Selection, Replicated Database, Machine Learning

1. Introduction

Given a database consisting of database tables with attributes (columns) and tuples (records) residing on a computer node, a workload of queries, and a space budget, the Index Selection Problem (ISP) [1] is to select an index configuration composed of one or more attributes in the database tables that would minimize the workload processing time. Divergent design index tuning [2, 3, 4] is the ISP problem for a replicated database where the database is replicated on multiple nodes. It utilizes the replication feature to select a set of index configurations, each of which is specialized for a subset of the workload to be run on a node. Due to such specialization, divergent design index tuning has been shown to perform better than the uniform approach where the same index configuration is selected for all nodes.

Divergent design index tuning is an NP-hard problem [2]; therefore, heuristic and optimization approaches [2, 3, 4] have been proposed to solve it on classical computers. However, for large-scale database applications in the era of Big Data where the numbers of computer nodes, database tables, tuples in each database table, attributes in each database table, and queries are very large, which exist in many application domains, such as e-commerce, finance, and healthcare, the optimization problem becomes much harder to solve. In addition, besides retrieval queries, update queries which include deletion, insertion, and modification may also occur frequently, which adds complexity to the problem. Developing a divergent design index tuning algorithm that can work efficiently for such applications is needed.

Recently, quantum computing has attracted a lot of global attention with many companies, research centers, and funding agencies around the world making substantial investments in the field [5, 6, 7, 8, 9]. As reported in [10], in 2019, the IBM Quantum Network had only 40 members, but as of April 2023, there are more than 210 Fortune 500 companies, universities, research labs, and startups participating in quantum computing. However, so far, not much research has been conducted in the area of quantum database management. This can be seen through the scarcity of quantum database research papers published in database journals and conferences. While quantum computing has been shown to offer promising improvements over classical computing with 10^3 times shorter runtime for query optimization [11, 12] and with constant runtime vs. rapidly rising runtime for transaction scheduling [13], which are two NP-Hard problems in database management, it has not been used to solve other database combinatorial optimization problems, such as the ISP and divergent design index tuning problem, which is also NP-Hard [1, 2]. At the same time, a good amount of research done by the

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — International Workshop on Quantum Data Science and Management (QDSM'23), August 28 - September 1, 2023, Vancouver, Canada

*Corresponding author.

ggruenwald@ou.edu (L. Gruenwald); t.winker@uni-luebeck.de (T. Winker); umut.calikyilmaz@uni-luebeck.de (U. Çalıkyılmaz); jinghua.groppe@uni-luebeck.de (J. Groppe); sven.groppe@uni-luebeck.de (S. Groppe)
0000-0001-5196-1117 (S. Groppe)

machine learning community has shown that quantum computing accelerates machine learning algorithms and enables them to learn using fewer data points than classical computing [14, 15, 16, 17, 18, 19]. Therefore, in this paper, we present our vision of leveraging quantum computing to develop a machine learning-based divergent design index tuning algorithm for large-scale replicated database applications.

The rest of the paper is organized as follows. Section 2 discusses the related work on classical divergent design index tuning and the quantum computing background. Section 3 describes the issues that a quantum divergent design index tuning algorithm using machine learning needs to handle. Section 4 presents our vision of such an algorithm. Finally, Section 5 concludes the paper with future research directions.

2. Related Work

In this section, we review the existing work on divergent design index tuning for classical computers and then provide the background on quantum computing.

2.1. Classical Divergent Design Index Tuning

There exist a good number of algorithms that have been proposed to select an index configuration on a single node for the entire query workload [20, 21, 22, 23, 24, 25, 26]. They employ either heuristic, optimization, or machine learning methods to carry out the index selection process for the entire query workload. For databases that are replicated on multiple nodes, instead of selecting the same index configuration for every node, divergent design index tuning algorithms [2, 3] make use of the replicas to select the index configurations, one for a subset of the query workload to be processed on a node. Like the index selection on a single node for the entire query workload, divergent design index tuning also aims to minimize the overall query processing costs. Using a heuristic method, DivDesign [2] first divides the query workload into subsets, distributes each subset to a node, selects an index configuration for each subset, and computes the estimated processing cost of each query on a node using the existing What-if tool [27]. It then redistributes the queries based on the estimated costs and repeats the process until no change is detected for the subsets. The results are the subsets of the query workload and the node and index configuration to run each subset. The algorithm reduces the index maintenance costs as it reduces the number of indexes on each node. However, it does not deal with dynamic query workload and node failure. RITA [3] employs an optimization method to solve the index selection problem. It uses a Binary Integer Program (BIP) to express the problem and uses an existing BIP solver to derive the solutions. While RITA addresses the weaknesses of DivDesign, the What-if tool [28] that it uses to estimate the query processing costs is not available on many database systems.

None of the algorithms, DivDesign and RITA, discussed above is able to learn from its past errors to improve future index selection. Machine learning methods have been used to incorporate learning into the index selection process. Various learned index advisors exist, such as the cost-model reinforcement learning [29], DRL-based index advisor [22], OpenGauss [23], document database index learning [26], SMARTIX [24], and DBA Bandis [25]. However, all these machine learning-based index selection algorithms are designed for a single node. To deal with replicated databases on multiple nodes, DRLindex [30, 31] employs a DRL algorithm to select the index configurations for a cluster replicated database. Its reward function considers the estimated processing cost of the workload and the replica load unbalance factor. While DRLindex [31] shows that DRL is a promising solution for divergent index selection, it solely relies on the query optimizer and only handles single-column indices. DRL Divergent Index Advisor (DINA) [4] is designed to address the limitations of DRLindex. To avoid possible query cost estimation errors due to using a query optimizer, DINA employs two training phases where it learns the efficiency of various possible query workload partitions and their index configurations by creating the indexes and observing the real execution times. It deals with both single-column and multiple-column indices.

All the above algorithms were designed for classical computers. To the best of our knowledge, there is no existing quantum algorithm for divergent design index tuning.

2.2. Quantum Computing Background

In this subsection, we introduce the quantum computing background including the basics of quantum information in Section 2.2.1 and quantum machine learning with a special focus on variational quantum circuits in Section 2.2.2.

2.2.1. Quantum Information and Quantum Computing

Quantum information processing emerged from the need to simulate quantum systems, which cannot be efficiently represented by classical information units [32]. The reason is that quantum systems can assume states that no classical systems can. Unlike a classical bit, a quantum bit (or qubit) can be in a superposition of 0 and 1, which is collapsed on one of them upon measurement [33]. In addition, a system of multiple qubits can be in an entangled state. When in such a state, the outcome of the measurements for the qubits are correlated to each other [33]. These properties make the simulation of a system of qubits (such as many other quantum systems) very hard for a classical computer. They are also the reason for the superiority of a quantum computer, which utilizes qubits, against a classical one [34].

As the superiority of quantum computers results from the special states that a system of qubits can assume, it is important to keep these states undisturbed until the end of the calculation. Unlike classical computers, the states are not discrete in a quantum computer. For n qubits, the quantum state is represented by 2^n complex values with norm 1 [33]. This makes a quantum state more susceptible to noise than a classical one. Environmental noise can cause decoherence in a quantum system, which will result in the loss of information [35].

The first studies on quantum algorithms have only focused on providing a speed-up for problems that are hard to solve on classical computers [36, 37, 38]. These studies ignored the possible limitations of the first generation of quantum computers. In recent years, with the invention of the first quantum computers, the focus of research has shifted to developing quantum algorithms that can work on modest hardware of our age. The first generation of quantum computers which are also called noisy-intermediate scale quantum (NISQ) computers [39], suffers from a small number of qubits and short coherence times. The main solution that has been proposed so far is to create quantum-classical hybrid algorithms that use quantum subroutines to solve smaller problems which are then used to solve the main problem using a classical computer. These algorithms are more suitable to work on NISQ devices.

2.2.2. Quantum Machine Learning and Variational Quantum Circuits

Among the hybrid algorithms, the ones that depend on variational quantum circuits (VQCs) take up a big volume [40]. In these algorithms, quantum circuits that apply parameterized quantum gates are used. These circuits consist of a small number of qubits and quantum gates, and they are run many times during the course of a program. Depending on the outcome of a VQC, a classical optimizer tunes the parameters. The purpose is to find the optimal values of the parameters and to output the result obtained using these values. QAOA [41] and VQE [42] are the most famous examples of such algorithms. An application of these two algorithms to another database problem, namely join order optimization, has already been studied [43]

It has been shown that VQCs can be utilized for machine learning applications, which is not surprising since the best set of parameters can be learned for a given task. Additionally, it has been proven that the representation capacity of a VQC is higher than a classical neural network [44]. This results in VQCs obtaining the same solution quality as neural networks using fewer parameters. Quantum machine learning has already been used to solve join order optimization [45].

3. Issues in Designing a Machine Learning-Based Divergent Design Index Tuning Algorithm

In this section, we first discuss the issues that a machine learning-based divergent design index tuning algorithm needs to address regardless of whether it is designed to run on a classical computer or on a quantum computer. These issues are due to the design requirements of indices, not due to the underlying hardware on which the algorithm runs. We then discuss the issues that exist due to the special characteristics of quantum computers.

3.1. Issues Common to Classical and Quantum Computers

1. Single-Column Indices or Multiple-Column Indices: As a query predicate may contain more than one attribute, an index tuning algorithm that only deals with single attributes may not yield good performance. However, dealing with multiple attributes requires index tuning algorithms to examine many combinations of attributes, which is a complex and expensive process.
2. Query Workload Analysis: The algorithm needs to be able to analyze the query workload to identify the query characteristics, such as the types, frequencies, and predicates of the queries, and take them into consideration in selecting appropriate indices. In addition, as the algorithm is designed for replicated databases on multiple nodes, it also needs to consider the current query workloads on the nodes in order to decide what index configurations should be selected to run which queries on which nodes.
3. Cost Evaluation: The cost of an index configuration consists of many components, such as time to create indices, time to perform I/Os, time to process queries on CPUs, time to update indices due to deletion/insertion/modification queries, and space to create indices. In addition, for replicated databases on multiple nodes, the cost in terms of the degree of load balancing should also be included. The algorithm design also needs to consider whether it should rely on the estimated costs produced by an existing query optimizer or it should rely on the actual query execution time. The former is simpler to use but the resulting

costs may not be accurate due to the errors of the query optimizer. On the contrary, the latter is more complicated as the algorithm will need to be designed in such a way that it can determine when and how the actual query execution information is obtained and how to incorporate such information into its process to improve the next index selection round, even though it can avoid the errors made by the query optimizer.

4. Machine Learning Issues: When using machine learning to predict index configurations as a part of a divergent design index tuning algorithm, the following are some of the additional issues that should also be investigated:
   a) Type of machine learning algorithm: There are different types of machine learning algorithms that can be used [46]: supervised machine learning such as decision tree and neural network; unsupervised machine learning, such as clustering and association rule mining; and reinforcement learning. With supervised learning, a lot of labeled data (training data) must exist prior to being used to build a learning model for index prediction, which is often difficult to obtain in real applications. With unsupervised learning, a lot of data still need to exist for the query patterns discovered to be meaningful, even though no labeled data are required. Reinforcement learning does not require labeled data to be available in advance because it learns as it goes, but a proper reward function must be designed to compute the rewards for the actions taken. A combination of these algorithms can also be used, such as deep reinforcement learning where a neural network trained based on a sample set of labeled data is used inside the reinforcement learning process.
   b) Data Representation, Collection, and Cleaning: What data are needed for the chosen machine learning algorithm, how to represent them, and how to collect them? Data may include attributes in the database, queries, database statistics, and so on. Once they are collected, how to deal with the data quality issues, such as data inconsistency, data missing, and outliers?
   c) Dynamic data: as data in database applications change over time, the machine learning model constructed before the data change may no longer perform well for index selection, then how to detect this phenomenon, and how to rebuild the model incrementally to reflect the new data without having to rerun the machine learning algorithm from the beginning are among the issues that need to be handled.

3.2. Issues Specific to Quantum Computers

1. Data Loading and Encoding: As classical data are stored as bit strings of 0s and 1s, to store and process them in a quantum computer, the first issue is how to encode classical data into qubits representing quantum states in such a way that would minimize the numbers of qubits and the circuit depths. There are various encoding techniques [9, 45]: basic encoding encodes each classical bit into a qubit; angle encoding provides a denser encoding by encoding each real value into a qubit; and amplitude encoding, which provides the densest encoding but also generates the most complex circuit depth, encodes each real value into the amplitudes of the quantum state, allowing 2^n values to be encoded into n qubits. As different encoding algorithms require different numbers of qubits and create different circuit depths, the choice of which algorithm to use should aim at the ability to store the necessary data using the minimum required quantum resources in terms of qubits and gate operations.
2. Data Decoding: This process describes how to retrieve and interpret the results from a quantum computer. There may be different options like choosing the most occurring bit pattern in the measurements, calculating the probabilistic expected value of the quantum circuit, i.e., the expectation value, and the probability distribution of the measured values. There may also be cases where more qubits are measured in the output than needed for the representation of the results, such that these additional values may just be ignored or several measured values are mapped to single ones of the result domain. For example, an index tuning algorithm may retrieve the tuning parameters from a quantum circuit.
3. Quantum Circuit Design: The longer the depths of the quantum circuits implementing the quantum operations designed for the divergent design index tuning algorithm, the more qubits, gates, and control operations are needed, the more errors are encountered due to noise and decoherence of qubits, and the longer it takes to execute the algorithm. It is thus important to design the algorithm in such a way that the circuit depths are minimized. Techniques that optimize quantum circuits, such as gate cancellation, gate merging, and gate synthesis should be investigated [47, 48]. Gate cancellation removes redundant gates such as those that cancel each other out; gate merging merges consecutive gates of the same operation into one single gate; and gate synthesis efficiently decomposes complex gates into sequences of elementary gates available on the hardware being used. While gate synthesis may increase circuit depth at first, its resulting elementary gate sequences may enable fur-

ther circuit optimization via gate cancellation and gate merging.

4. Training Data Handling: Training data is used in the machine learning part of the machine learning-based divergent design index tuning algorithm. On classical computers, the amount of training data and its quality impact the accuracy of the machine learning model. The same is true on quantum computers. However, with errors caused by noise and decoherence of qubits as discussed in the background Section 2.2, an important issue is to determine what would be the sufficient amount and quality of training data to meet the required accuracy of the machine learning model, and subsequently, the accuracy of the divergent design index tuning algorithm, while taking such errors into account.
5. Quantum Machine Learning: As there already exists a good number of quantum machine learning algorithms in the literature [14, 15, 16], an issue to consider is whether one or more such algorithms can be incorporated into the divergent design index tuning algorithm, or whether any of those algorithms needs to be revised before adoption, or a new quantum machine learning algorithm should be developed.
6. Hybrid Algorithms: Due to the limitation in the number of qubits and the decoherence time available on current quantum computers, hybrid approaches that take advantage of both classical and quantum computers have been proposed in areas such as quantum machine learning [14, 15, 16] and quantum query optimization [9, 45], where a part of the algorithms are executed on a classical computer while another part is executed on a quantum computer. To adapt a hybrid approach in quantum machine learning-based divergent design index tuning, some key issues that need to be addressed are to decide which parts of the algorithm should be processed on a classical computer, which parts should be processed on a quantum computer, and how these parts should communicate with each other to produce the intermediate results and the final results, while taking advantage of classical parallelism and quantum parallelism as well as considering load balancing on each computer and the waiting time between different processes.
7. Algorithm Evaluation: To evaluate the quantum speedup in terms of query execution time, theoretical analyses and/or experimental evaluations should identify cases where quantum algorithms can outperform classical counterparts. They should study the impacts of the database size in terms of attributes and tuples, query workload, database update rate, number of nodes, number of qubits, number of gates, circuit depths, decoherence time, and error rate.
8. Properties of Quantum Computers: There is a rapid development and huge improvements in the development of quantum hardware in recent years and it is expected that this progress will continue in the future. However, in order to guide the development of quantum hardware and provide researchers with quantum hardware feedback for their development goals, it is a key point to investigate the following question: Which properties (such as latencies of the quantum gates, supported quantum circuit depths and noise rates) should a future quantum computer have to achieve a certain accuracy and performance improvement over classical hardware for our quantum approach?

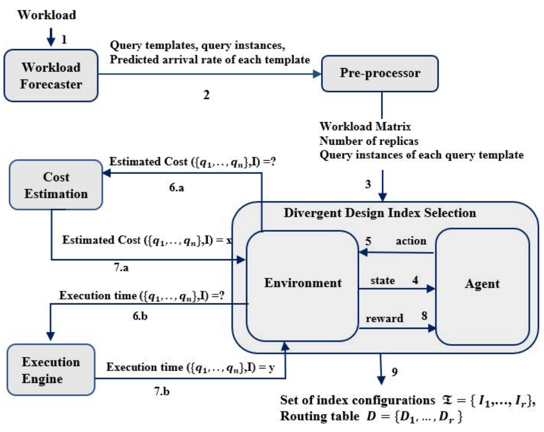
Workload
1
Workload Forecaster
Query templates, query instances, Predicted arrival rate of each template
2
Pre-processor
Workload Matrix
Number of replicas
Query instances of each query template
3
Cost Estimation
Estimated Cost ($\{q_1,\ldots,q_n\}$,I) =?
6.a
Estimated Cost ($\{q_1,\ldots,q_n\}$,I) = x
7.a
Divergent Design Index Selection
Environment
Agent
5 action
state 4
reward 8
Execution time ($\{q_1,\ldots,q_n\}$,I) =?
6.b
Execution Engine
Execution time ($\{q_1,\ldots,q_n\}$,I) = y
7.b
9
Set of index configurations $\mathfrak{I} = \{I_1,\ldots,I_r\}$,
Routing table $D = \{D_1,\ldots,D_r\}$

Figure 1: The architecture of DINA [4]

4. A Proposed Quantum Machine Learning-Based Divergent Design Index Tuning Algorithm

In this section, we first briefly describe DINA [4], a divergent design index tuning algorithm using Deep Reinforcement Learning (DRL) for classical computers that has been shown to perform better than existing algorithms. We then provide our vision of how to extend this algorithm for quantum computers.

4.1. DINA: A DRL Divergent Design Index Tuning Algorithm for Classical Computers

In [4], DINA (DRL Divergent Index Advisor) has been proposed to select indices for a database replicated on multiple nodes. The database copy on each node is called a replica. DINA works for both single-column indices and multiple-column indices. It uses DRL where its DRL agent learns as it goes by exploring different query workload partitioning alternatives among the replicas and the effectiveness of their index configurations. The DRL efficiently explores the large search space. To avoid relying solely on the query optimizer as the estimated costs generated by the query optimizer can be erroneous, it trains the DRL agents in two phases: pre-training using the estimated query cost generated by a query optimizer and the re-training phase using the actual query execution time. As shown in Fig. 1, the algorithm has the three following major steps:

Step 1: Invoke the Workload Forecaster module which runs an algorithm that forecasts the coming query workload [49] to generate a set of query templates, and the query instances and the number of query instances for each query template.

Step 2: Submit the query workload to the Pre-Processing module to derive a set of candidate indices which are the attributes in the WHERE and JOIN clauses in the query templates, and create a workload matrix where rows are the query templates, columns are the candidate indices, and an entry has a value of 1 if the candidate index is an attribute in the WHERE or JOIN clause of the corresponding query template and 0 otherwise.

Step 3: Use DRL to process the query workload to select a proper index configuration for each replica such that the overall query processing cost and the workload skew on the replicas are minimal. The DRL framework is composed of the environment and agent. The environment defines the states of the replicas based on their index configurations and the possible actions for each state that the agent can take and returns a reward value for each action. The current state is represented as a state matrix where rows are the replicas, columns are the candidate indices, and an entry is 1 if the agent has selected the corresponding index for the corresponding replica and 0 otherwise. The possible actions are which query templates to be executed on which replicas and which candidate indices can be used on which replicas. An action at time t denoted as a_t is to select a query template to be executed on a replica and to select an index for the queries of that template.

Given a state at time t denoted as s_t, the agent takes the action predicted by Deep Q-learning [50], the state is then changed from s_t to the new state s_{t+1}, and the agent receives a reward value rw_t. The reward of an action is computed using a reward function based on the query processing cost reduction $reward(I)$ calculated from the reduction in the estimated query processing cost generated by the query optimizer (during the agent pre-training phase) or in the actual query execution cost (during the agent re-training phase), and the inverse of the total workload-skew of the replicas $reward(S)$.

$$reward = \alpha \times reward(I) + \beta \times reward(S)$$

where α and β are the weights obtained by trial and error defining a trade-off between query cost reduction and load-balancing. The details of these functions can be found in [4]. In Deep Q-Learning, the value of each action a in a state s at each time step is computed using a Q-function. The action predicted for the agent to take is the one that has the highest Q value, which is the most rewarding action. In DINA, the Q-function used is the following Bellman equation [51]:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[rw_{t+1} + \gamma max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

where $Q(s_t, a_t)$ is the Q-value for state s_t and action a_t, α is the learning rate, rw is the reward value, γ is the discount rate, and $max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ is the maximum value of the estimated future return for the next state. Due to the large search space, the agent uses a neural network to predict the action to take at each state. The network is trained during the agent pre-training and re-training phases using a random sample of the agent's experience tuples $(s_t, a_t, rw_t, s_{t+1})$ stored in the memory.

Experiments using the TPC-H [52] and TPC-DS [53] database benchmarks have shown that DINA performs better than the existing algorithms, DivDesign [2] and RITA [3]; and the version of DINA that uses the actual query execution time through the re-training phase performs better than the version of DINA that uses the query processing cost estimated by the query optimizer through the pre-training phase. The detailed description and experiment results of DINA can be found in [4].

4.2. Quantum DINA: Our Vision to transform DINA to a quantum approach

Looking at the architecture of DINA in Figure 1, we identify three possible components which can be replaced by a quantum approach. These are the Workload Forecaster, the Divergent Design Index Selection, and the Cost Estimation. We can choose to replace all with quantum approaches or keep some classical ones. We will use variational quantum circuits with their properties to potentially learn with fewer training data [17], such that our new architecture can adapt faster to changes in indices, queries, and data sets.

Workload Forecaster The workload forecaster can take advantage of the probabilistic nature of quantum computing to simulate uncertainty about the expected query workload. We have two possible approaches of mapping quantum states to query templates.

One approach is to assign a query template to each quantum state and the probability of the quantum state is the percentage of queries belonging to this template. Many shots of the quantum circuits are required to get a good approximation of all probabilities.

Another approach is to assign a query template to each qubit and count how often each qubit is measured as 1. This allows the counting of multiple templates with each execution and thus requires fewer shots.

Index Selection We can replace the deep neural network of the classical agent with a VQC. This component has the task of choosing an action from a given state of the environment. It gets the state of the environment and an action as input and returns a predicted reward.

The state of the environment is a $r \times n$ matrix of binary variables. A basis encoding would require $r * n$ qubits and is not feasible. Instead of looking at it as a matrix of binary variables, we can interpret a row or column as an integer in binary code. This allows us to reduce the number of inputs to n or r integers. As we know the number of bits, we know the upper bound of the integer value. This makes it suitable for the scaling required for angle encoding.

The action consists of selecting a replica for a query template and choosing an index configuration. Each query template and each replica can be given an ID, such that we have two additional integers as input. The choice of an index configuration is a set of possible indices. As we know all possible indices, this can be seen as a vector of binary variables, which in turn can again be interpreted as an integer.

With this encoding approach, we get $n + 3$ or $r + 3$ integers as input, and, with angle encoding, we get the same number of qubits.

Cost estimation The cost estimator gets a query template and an index configuration as input and returns the estimated cost for the execution of a query from the given template with the given index configuration. If we use the same encoding method as that used for the index solution, we will have 2 input values, which are the ID of the query template and the set of indices interpreted as an integer. With angle encoding, 2 qubits would be sufficient for the input. As our output should be the estimated cost of the query execution on the replica with the given index, it should be a continuous value. By using the probabilities of a qubit being measured as 1, we can create a continuous output value. If p_1 is the probability of 1 being measured for the first qubit and p_2 the probability of 1 for the second qubit, we can define the cost c as $c = \frac{p_1}{p_2}$. This will result in a value in the interval $[0, \infty]$ with $c = 0$ for $p_1 = 0$ and $c = \infty$ for $p_2 = 0$. The precision of this cost value depends on the number of shots of the quantum circuit.

The suggested encoding and decoding approaches are not the only possible representations of the inputs and outputs. Many other representations exist and an experimental evaluation is needed to find out which representation has the best learning performance while having a reasonable qubit requirement.

5. Conclusions and Future Research

In this paper, we aimed to provide our vision of a quantum divergent design index tuning algorithm that makes use of machine learning to select indices for large-scale replicated database applications. We discussed the issues that the algorithm needs to address. We then presented an algorithm for classical computers and our approaches to how to transform it into a quantum algorithm.

For future research, we plan to formalize and implement our quantum algorithm using one of the widely-used open-source software development kits for quantum computers like the popular qiskit [54]. We will then conduct experiments to evaluate the algorithm's performance and compare the experimental results to those of its classical counterpart.

Acknowledgements

This work is funded by the German Federal Ministry of Education and Research within the funding program quantum technologies - from basic research to market – contract number 13N16090.

References

[1] D. Comer, The difficulty of optimum index selection, ACM Transactions on Database Systems (TODS) 3 (1978) 440–445,.

[2] M. Consens, K. Ioannidou, J. LeFevre, N. Polyzotis, Divergent physical design tuning for replicated databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2012, pp. 49–60,.

[3] Q. Tran, I. Jimenez, R. Wang, N. Polyzotis, A. Ailamaki, Rita: An index-tuning advisor for replicated databases, in: Proceedings of the 27th International Conference on Scientific and Statistical Database Management, 2015, pp. 1–12,.

[4] Z. Sadri, L. Gruenwald, A divergent index advisor using deep reinforcement learning, in: International Conference on Database and Expert Systems Applications (DEXA), 2022, pp. 139–152,.

[5] S. Groppe, J. Groppe, U. Calikyilmaz, T. Winker, L. Gruenwald, Quantum data management and quantum machine learning for data management: State-of-the-art and open challenges, in: The EAI International Conference on Intelligent Systems and Machine Learning (EAI ICISML), 2022.

[6] Ericson, Ericson established quantum research hub in canada, 2023. URL: https://www.ericsson.com/en/press-releases/6/2023/ericsson-establishes-quantum-research-hub-in-canada,.

[7] N. Liu, Cleveland clinic install world's first quantum computer for healthcare research, SDXCentral (2023). URL: https://www.sdxcentral.com/articles/news,.

[8] National Quantum Initiative (2023). URL: https://www.quantum.gov/,, accessed April 2023.

[9] U. Calikyilmaz, S. Groppe, J. Groppe, T. Winker, S. Prestel, F. Shagieva, D. Arya, F. Preis, L. Gruenwald, Opportunities for quantum acceleration of databases, in: Optimization of Queries and Transaction Schedules, Accepted for publication in the Proceedings of The International Conference on Very Large Data Bases (VLDB), 2023.

[10] Forrester, Key factors driving quantum computing into a new chapter, 2023. URL: https://www.forbes.com/sites/forrester/2023/04/07/key-factors-driving-quantum-computing-into-a-new-chapter.

[11] M. Schönberger, S. Scherzinger, W. Mauerer, K. Winter-sperger, H. Safi, M. Franz, L. Wolf, M. Periyasamy, C. Ufrecht, Ready to leap (by co-design)? join order optimisation on quantum hardware, in: ACM SIGMOD/PODS International Conference on Management of Data, 2023.

[12] I. Trummer, C. Koch, Multiple query optimization on the d-wave 2x adiabatic quantum computer, 2015. ArXiv preprint arXiv:1510.06437,.

[13] T. Bittner, S. Groppe, Avoiding blocking by scheduling transactions using quantum annealing, in: The 24th Symposium on International Database Engineering Applications (IDEAS), 2020.

[14] Z. Abohashima, M. Elhosen, E. Houssein, M. W. M, Classification with quantum machine learning: A survey, 2020. ArXiv, arXiv:2006.12270,.

[15] C. Ciliberto, M. Herbster, A. Ialongo, M. Pontil, A. Rocchetto, S. Severini, L. Wossnig, Quantum machine learning: a classical perspective, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474 (2018).

[16] R. Zhao, S. Wang, A review of quantum neural networks: Methods, models, dilemma, 2021. ArXiv, 2109.01840.

[17] M. Caro, H. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, P. Coles, Generalization in quantum machine learning from few training data, Nat. Commun 13 (2022).

[18] H. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, J. McClean, R., Power of data in quantum machine learning, Nat. Commun 12 (2021).

[19] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, Phys. Rev. Lett 113 (2014).

[20] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, M. Syamala, Database tuning advisor for microsoft sql server 2005, in: The ACM SIGMOD International Conference on Management of Data (SIGMOD), 2005, pp. 930–932,.

[21] D. Dash, N. Polyzotis, Ailamaki, A.: Cophy: a scalable, portable, and interactive index advisor for large workloads, PVLDB 4 (2011) 362–372,.

[22] H. Lan, Z. Bao, Y. Peng, An index advisor using deep reinforcement learning, in: Proceedings of the 29th ACM International Conference on Information Knowledge Management, 2020, pp. 2105–2108,.

[23] G. Li, X. Zhou, J. Sun, X. Yu, Y. Han, L. Jin, W. Li, T. Wang, Li, S.: opengauss: An autonomous database system, Proceedings of the VLDB Endowment 14 (2021) 3028–3042,.

[24] G. Paludo Licks, J. Colleoni Couto, P. Fátima Miehe, R. Paris, D. Dubugras Ruiz, F. Meneguzzi, Smartix: A database indexing agent based on reinforcement learning, Applied Intelligence 50 (2020) 2575–2588,.

[25] R. Perera, B. Oetomo, B. Rubinstein, R. Borovica-Gajic, Dba bandits: Self-driving index tuning under ad-hoc, analytical workloads with safety guarantees, in: IEEE 37th International Conference on Data Engineering (ICDE), 2021, pp. 600–611,.

[26] M. Schaarschmidt, A. Kuhnle, B. Ellis, K. Fricke, F. Gessert, E. Yoneki, Lift: Reinforcement learning in computer systems by learning from demonstrations, 2018. ArXiv preprint arXiv:1808.07903,.

[27] S. Chaudhuri, V. Narasayya, Autoadmin what-if index analysis utility, ACM SIGMOD RECORD 27 (1998) 367–368,.

[28] S. Papadomanolakis, D. Dash, A. Ailamaki, Efficient use of the query optimizer for automated physical design, in: The 33rd International Conference on Very Large Data Bases (VLDB), 2007, pp. 1093–1104,.

[29] D. Basu, Q. Lin, W. Chen, H. Vo, Z. Yuan, P. Senellart, S. Bressan, Cost-model oblivious database tuning with reinforcement learning, in: Proceedings of The International Conference on Database and Expert Systems Applications (DEXA), 2015, pp. 253–268,.

[30] Z. Sadri, L. Gruenwald, E. Leal, Online index selec-

tion using deep reinforcement learning for a cluster database., in: Proceedings of The 36th IEEE International Conference on Data Engineering Workshops (ICDEW), 2020, pp. 158–161,.

[31] Z. Sadri, L. Gruenwald, E. Leal, Drlindex: deep reinforcement learning index advisor for a cluster database, in: Proceedings of The 24th Symposium on International Database Engineering and Applications (IDEAS), 2020, p. 1–8.

[32] R. P. Feynman, Simulating physics with computers, in: Feynman and computation, CRC Press, 2018, pp. 133–153.

[33] M. A. Nielsen, I. Chuang, Quantum computation and quantum information, 2002.

[34] D. R. Simon, On the power of quantum computation, SIAM journal on computing 26 (1997) 1474–1483.

[35] M. Schlosshauer, Quantum decoherence, Physics Reports 831 (2019) 1–57.

[36] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th annual symposium on foundations of computer science, Ieee, 1994, pp. 124–134.

[37] L. K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 1996, pp. 212–219.

[38] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Physical review letters 103 (2009) 150502.

[39] J. Preskill, Quantum computing in the nisq era and beyond, Quantum 2 (2018) 79.

[40] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al., Variational quantum algorithms, Nature Reviews Physics 3 (2021) 625–644.

[41] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint arXiv:1411.4028 (2014).

[42] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, J. L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nature communications 5 (2014) 4213.

[43] N. Nayak, J. Rehfeld, T. Winker, B. Warnke, U. Çalıkyılmaz, S. Groppe, Constructing optimal bushy join trees by solving qubo problems on quantum hardware and simulators, in: Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE), Seattle, WA, USA, 2023. URL: https://doi.org/10.1145/3579142.3594298.

[44] Y. Du, M.-H. Hsieh, T. Liu, D. Tao, Expressive power of parametrized quantum circuits, Physical Review Research 2 (2020) 033125.

[45] T. Winker, U. Çalıkyılmaz, L. Gruenwald, S. Groppe, Quantum machine learning for join order optimization using variational quantum circuits, in: Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE), Seattle, WA, USA, 2023. URL: https://doi.org/10.1145/3579142.3594299.

[46] P. N. Tan, M. Steinbach, A. Karpatne, V. Kumar, Introduction to Data Mining, 2nd ed., Pearson, 2018.

[47] Y. Nam, N. Ross, Y. Su, A. Childs, D. Maslov, Automated optimization of large quantum circuits with continuous parameters, npj Quantum Information 4 (2018) 1–12,.

[48] O. Matteo, M. Mosca, Parallelizing quantum circuit synthesis, Quantum Science and Technology 1 (2016) 015003.

[49] L. Ma, D. Van Aken, A. Hefny, G. Mezerhane, A. Pavlo, G. Gordon, Query-based workload forecasting for self-driving database management systems, in: Proceedings of The ACM International Conference on Management of Data (SIGMOD), 2018, p. 631–645.

[50] G. Li, X. Zhou, S. Li, B. Gao, Qtune: A query-aware database tuning system with deep reinforcement learning, in: Proceedings of The International Conference on Very Large Data Bases (VLDB), 2019, pp. 2118–2130,.

[51] M. Wiering, M. Otterlo, Reinforcement Learning: State-of-the-Art, Springer, 2014.

[52] TPC, Tpc-h benchmark, 2023. URL: http://www.tpc.org/tpch/.

[53] TPC, Tpc-ds benchmark, 2023. URL: http://www.tpc.org/tpcds/.

[54] IBM, Qiskit: Open-source quantum development, 2023. URL: https://qiskit.org/.