An experimental study of online database index selection algorithms with performance guarantees

Tanzid Sultan

August 2024

Project Supervisors:
Dr. William Umboh
Dr. Junhao Gan
Dr. Renata Borovica-Gajic

A Research Proposal Submitted for the COMP90055 subject at the University of Melbourne.

Contents

1 Introduction and Background **1**

1.1 Database index selection problem 1

1.2 The need for online index tuning 1

1.3 Existing methodologies for online index tuning 2

1.4 Lack of efforts in establishing performance guarantees 3

1.5 Limitations of the what-if cost model 4

2 Research Objectives **5**

2.1 Online index selection using MTS: extending WFIT 5

2.2 Dynamic environments: non-stationary and non-stochastic bandits . . . 6

2.3 Comparing WFIT and Bandit Performance 7

3 Project Plan Outline **8**

1 Introduction and Background

1.1 Database index selection problem

In today's era of *Big Data*, organizations across various sectors - such as finance, healthcare, social media and e-commerce - rely on vast amounts of data to carry out their daily operations and for strategic planning and decision-making. A Database Management System (DBMS) is a software system designed to robustly store and manage such collections of data. A DBMS also provides an interface that allows end-users to query and update the data. The speed and efficiency with which a DBMS is able to process user queries/updates depends critically on the *physical design structures* (PDS) available within the system. These are data structures which specify how the data is stored on disk, retrieved and updated. We will restrict our attention to databases which follow the *relational model* [24] in which the data is organized across multiple *tables* related to each other through *keys*.

A PDS on each table primarily consist of *heap files*, in which the data is unsorted, and *clustered indexes* in which the data is sorted according to a *primary key* which uniquely identifies the data records. These PDS do not efficiently support most queries which are based on columns other than the primary key. This is where *secondary/nonclustered indexes*[1] come into play . A database index serves a similar purpose as an index at the end of a book, i.e. it allows us to quickly retrieve the specific rows we are interested in without performing a *full-table scan*. Because the main performance bottleneck in large-scale relational databases is the disk I/O, an intelligently designed set of secondary indexes, also referred to as a *configuration*, can significantly reduce the amount of disk I/O needed to process a query resulting in dramatic speedup.

1.2 The need for online index tuning

Most DBMS do not automatically create secondary indexes. Instead, either an expert database administrator (DBA) needs to manually select and materialize a set of secondary indexes or a non-expert user can invoke *offline* index recommendation tools provided by the DBMS, such as the *Database Tuning Advisor* in Microsoft SQL Server [1]. Such offline selection/tuning methods require a representative workload to be provided based on which index recommendations are generated. This approach poses several challenges, particularly when workload patterns change unpredictably over time due

[1] Secondary indexes come in various types, such as B^+-tree index, hash index, etc. We will restrict our attention to only B^+-tree indexes due to their efficiency, versatility and prevalence in most applications.

to varying user demands or seasonal trends, or when the scale of the database changes due to data growth or increasing number of users. Such dynamic environments would necessitate periodically invoking the offline tuning process from scratch, each time with a new representative workload. These offline methods also don't take into consideration, the overheads associated with generating recommendations and materializing the recommended indexes. This can often lead to a situation where these overheads can cause significant delays, outweighing future query processing costs.

Given these challenges with offline tuning, there is a strong motivation for developing **automated online index selection algorithms** which can continuously monitor evolving database workloads (and perhaps also evolving external environmental factors) and adaptively adjust the index configuration based on observed patterns. More formally, we can frame the algorithm as solving a combinatorial optimization program, where the objective is to make a sequence of decisions, each decision corresponding to selecting and materializing a set of secondary indexes (i.e. configuration), such that the total running time of the entire workload and configuration changes is minimized over an indefinitely long horizon, subject to some memory constraints. The difficulty in designing these online algorithms comes from the fact that each decision has to be made given only knowledge of workloads and environment observed in the past and without knowledge of the future.

A user can only place their trust on an online algorithm if it provides some form of **robust theoretical (worst-case) performance guarantee**. Such a guarantee would ensure that the algorithm performs well under various scenarios and offer protection against arbitrary performance degradation. In this project, we will restrict our attention to online index selection algorithms that satisfy this important criterion.

1.3 Existing methodologies for online index tuning

Substantial efforts have been made to address the need for online index tuning, resulting in a number of notable algorithms over the last few decades. A key component which is common across all of these different approaches is the use of a **cost model** to estimate query execution time in any *hypothetical* index configuration[2]. This cost estimation is a necessary step because, at a high level, each of these approaches perform an *exploration* of some part of the space of all possible configurations to find the best possible configuration according to some specified criterion. Beyond the commonality, these algorithms can be broadly divided into two categories, based on the nature of the cost model.

[2] a hypothetical configuration is one that has not been materialized

In the first category, we have algorithms that rely on the *internal* cost model[3] of the query optimizer, such as [5, 12, 15, 19, 21, 22, 28]. In addition to using this internal cost model, some of these algorithms, such as [5], are also tightly coupled to the DBMS query optimizer and are *invasive* in the sense that they intercept the query plan at various stages of query optimization and make local plan transformations to enumerate and explore a large number of alternative hypothetical configurations and their corresponding estimated costs. Among all approaches in the first category, the **WFIT** algorithm of [21], which uses a variant of the *work function algorithm* from *metrical task systems* (MTS) [3], is most notable due to its theoretical performance guarantee in the form of a bounded *competitive ratio*. To our knowledge, [15] and [21] are the only two works which have applied MTS to the online index selection problem and provided proof of bounded competitive ratio[4], with the latter providing a substantially tighter bound.

In recent years, there has been a shift towards using learning-based approaches for index selection such as [2, 7, 8, 9, 16, 18, 17, 23, 26, 29]. This brings us to the second category of algorithms which use an *external* cost model, that learns directly from observed data, instead of relying on the DBMS what-if interface. [2] and [8] take a reinforcement learning (RL) approach, modeling queries and updates as a Markov decision process (MDP) with states represented by the index configurations, actions represented by configuration changes and rewards represented by query execution and configuration change costs. [2] uses least-squares policy iteration while [8] applies deep Q-learning to solve the MDP. On the other hand, [16, 17, 18] uses a different kind of RL approach in the form of stateless Multi-Arm Bandits (MAB) where bandit actions/arms are represented by configuration changes. Among all the algorithms in the second category, the MAB particularly stands out due to its theoretical performance guarantee in the form of a sub-linear *regret* bound and has been shown in experiments to have superior performance compared to a state-of-the-art offline tuning tool provided by a commercial DBMS.

1.4 Lack of efforts in establishing performance guarantees

Surveying the literature on this topic of online database tuning also reveals that a vast majority of the existing techniques make no direct effort towards providing theoretical performance guarantees. In the study of online optimization, two very different per-

[3] also often referred to as a *what-if* interface [6]

[4] [5] also proves a constant competitive ratio for the special case of a single index, however their analysis does not extend to the general multi-index case

formance metrics are typically used, each of them leading to very different algorithms and methodologies. On the online learning-based algorithms side, the metric of choice is the cumulative **regret** which is defined as the difference between the solution cost of the algorithm and that of the optimal offline *static* solution[5]. On the other side (which includes MTS), the metric used is the **competitive ratio**, which is the worst-case ratio between the solution cost of the algorithm and that of the optimal offline *dynamic* solution. Aside from the obvious difference between these two metrics, one being additive and the other multiplicative in nature, fundamentally the regret measures the algorithms ability to learn a static concept while the competitive ratio measure the ability to learn a dynamic concept [27].

In the realm of database online index selection, the two most notable algorithms which pay attention to the aforementioned performance metrics are:

- MAB algorithm of [16, 17, 18] with its sublinear regret bound
- MTS-based WFIT algorithm of [21] with its bounded competitive ratio

Both MAB and WFIT have individually demonstrated strong performance over a range of benchmarking tests [11, 18], however a direct comparison between the two has not been attempted. Also, despite their good performance, each still possess certain weak-points that we believe can be overcome by implementing appropriate extensions. This will be discussed further in the Section 2 and will form the basis for this research project.

1.5 Limitations of the what-if cost model

In section 1.3, we discussed two categories of online index tuning algorithms, differing by their use of an internal vs. an external cost model. We also noted that over the years, there has been a gradual shift towards the second category, particularly learning-based external cost models. This transition is mainly due to several drawbacks of the query optimizer's internal cost model.

The query optimizer cost estimates can be highly error prone, mainly due to inaccurate table statistics and cardinality estimation [14]. This becomes especially apparent when data distributions are highly skewed, as is often the case in realistic scenarios, where estimations could be off by orders of magnitude. Such errors can lead to large discrepancies between estimated and actual costs and adversely effect index tuning

[5] an optimal offline static solution is also referred to as an optimal *policy* in reinforcement learning literature

tools [4]. The use of arbitrary units for query optimizer cost estimates also makes it difficult, if not impossible, to directly compare these estimates with actual observed costs. There is also no direct way to obtain an index creation cost estimate using the what-if interface in some DBMS[6]. Moreover, what-if calls tend to be slow in practical situations and there is significant variability in the running-time.

2 Research Objectives

2.1 Online index selection using MTS: extending WFIT

The WFIT algorithm [21] based on metrical task systems has the distinct property of having a bounded competitive ratio, which gives us an absolute worst-case theoretical guarantee for how much the solution quality can degrade. This property makes WFIT an attractive choice as an online index selection algorithm. Experimental studies in [11] have compared the performance of several online index selection algorithms, including WFIT and the algorithm from [5], across both static and dynamic workloads. WFIT is able to consistently show superior performance across a wide range of metrics, such as fast convergence under dynamic workloads and low variability in index selection quality.

However, when it comes to the overhead of generating index recommendations, WFIT has been shown to be the slowest, due to the overhead being dominated by the large number of what-if calls made to the query optimizer. Aside from this high overhead of generating recommendations being a weak-point of WFIT, we would also like to argue that the over-reliance on the what-if calls is itself a major weak-point. This can be understood by fact that the query optimizer's internal cost model can be highly inaccurate and inefficient (see Section 1.5). The experiments in [11] fail to expose this weakness as they do not incorporate any datasets which have very high levels of skew in their data distributions.

This motivates the **first goal** for our project, which is to extend WFIT by entirely replacing the use of the internal (what-if) cost model with an *external* cost model based on machine-learning, which incrementally learns from observations and is independent of the query optimizer. This goal directly aims to remedy the aforementioned weak-points of WFIT, which includes increasing the accuracy of cost estimation while significantly reducing the overhead. To achieve this goal, we will execute the following tasks:

[6] e.g. Microsoft SQL Server does not provide the functionality for estimating hypothetical index creation cost

- Identify and implement an algorithm for the external cost model which has fast, efficient and robust performance. From our preliminary survey of possible candidates, we believe an online/incremental ridge regression model [20] with carefully engineered feature inputs might be a good choice.
- Implement the WFIT algorithm and integrate with the external cost model
- Carry out experiments with highly skewed datasets to study and verify the performance benefits of using an external cost model

2.2 Dynamic environments: non-stationary and non-stochastic bandits

The MAB algorithm for online index selection is based on the *stochastic* bandit assumption, according to which the reward for each arm is drawn i.i.d. from a fixed/stationary probability distribution. Under certain real-world settings, this assumption is often violated. For instance, in a multi-tenant cloud environment [10] where compute and storage resources are shared among multiple independent DBMS instances serving large number of users, the load on such shared servers may vary greatly depending on environmental factors beyond the control of the DBMS. As a result, the amount of resources available to a DBMS could change unpredictably which can then affect query processing times. This scenario of a dynamic environment can manifest as non-stationarity of bandit arm reward distributions. The regret bound of the stochastic MAB may not protect us from arbitrary performance degradation in such a scenario.

The experimental investigations in [16, 17, 18] do not consider such dynamic environments and steps for extending the stochastic MAB in this setting have not been explicitly discussed. Therefore our **second goal** for this project is to design controlled experiments for simulating a dynamic environment which could expose the weak point of the stochastic bandit, then consider strategies for extending the MAB index selection algorithm enablilng it to handle dynamic environments. From our preliminary surveys, we have identified two suitable directions for achieving this goal.

One possibility is a straight-forward extension of the contextual stochastic bandit, whereby we directly augment the context vector with additional features which can track changes in the reward distribution [13, 25]. For instance, in a multi-tenant database environment, these features could perhaps include information pertaining to the load that the server is currently experiencing. This contextual information could then be utilized in the decision-making for super-arm selection.

Another direction, which is not so straight-forward, is to utilise a *non-stochastic*[7] contextual bandit algorithm such as [30]. The non-stochastic bandit model removes the assumption of i.i.d. rewards coming from a fixed distributions and allows the rewards to come from any non-stationary distribution, even from an adversarial process. Non-stochastic bandit algorithms vary significantly from deterministic stochastic bandit approaches (e.g. UCB), they require some randomized elements [13] and implementation is known to be highly non-trivial for problems with large action spaces. The non-stochastic bandit algorithm [30] has been shown to have a sub-linear regret bound.

To achieve our second goal, we will be executing the following steps:

- Implement both the simple extension to the stochastic contextual bandit and the non-stochastic bandit algorithm from [30].
- Design controlled experiments simulating non-stationary reward distributions. A very simple strategy that we have identified for simulating a non-stationary reward distribution is to add an extra *shift* value to observed query execution/index creation costs.
- Run the experiments, evaluate and compare the performance of the stochastic bandit with the extended approaches and study the impact of non-stationary rewards.

2.3 Comparing WFIT and Bandit Performance

The WFIT and Bandit algorithms have very different performance guarantees, the former having bounded competitive ratio and the latter having bounded regret. A unified theoretical analysis for comparing these two different metrics directly may not be possible. Hence, in this project, we will not focus on this particular aspect. That being said, our **third and final goal** for this project is to carry out a suite of experiments similar to [16], in order to benchmark the performance of the extended Bandit algorithms, the extended WFIT algorithm, the built-in tuning advisor tool of Microsoft SQL server in addition to a no-index baseline. The experimental aspects of goals 1 and 2 will be covered under goal 3. Detailed logistics regarding the experiments are yet to be fully determined and the depth of experimentation will depend on amount of time remaining after completion of the first two goals.

[7] the non-stochastic bandit is also sometimes referred to as an *adverserial* bandit

3 Project Plan Outline

Given the remaining duration for the project encompasses weeks 4-12 of the current semester, we are providing the following week-by-week plan outline for achieving the tasks in our 3 goals:

- *Weeks 4, 5* - Implement external cost model and extended WFIT for goal 1, run preliminary tests.
- *Weeks 6, 7* - Implement extended Bandit for goal 2, run preliminary tests.
- *Weeks 8, 9* - Implement experiment suites for goal 3, carry out analysis of results
- *Weeks 10, 11, 12* - Prepare thesis report and presentation.

References

[1] Sanjay Agrawal et al. "Database Tuning Advisor for Microsoft SQL Server 2005". In: *VLDB*. Very Large Data Bases Endowment Inc., 2004.

[2] Debabrota Basu et al. "Regularized Cost-Model Oblivious Database Tuning with Reinforcement Learning". In: *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXVIII - Volume 9940*. Berlin, Heidelberg: Springer-Verlag, 2016, 96–132.

[3] Allan Borodin, Nathan Linial, and Michael E. Saks. "An optimal on-line algorithm for metrical task system". In: *J. ACM* 39.4 (1992), 745–763.

[4] Renata Borovica, Ioannis Alagiannis, and Anastasia Ailamaki. "Automated physical designers: what you see is (not) what you get". In: *Proceedings of the Fifth International Workshop on Testing Database Systems*. DBTest '12. Scottsdale, Arizona: Association for Computing Machinery, 2012.

[5] Nicolas Bruno and Surajit Chaudhuri. "An Online Approach to Physical Design Tuning". In: *2007 IEEE 23rd International Conference on Data Engineering*. 2007, pp. 826–835.

[6] Surajit Chaudhuri and Vivek Narasayya. "AutoAdmin "what-if" index analysis utility". In: *SIGMOD Rec.* 27.2 (1998), 367–378.

[7] Bailu Ding et al. "AI Meets AI: Leveraging Query Executions to Improve Index Recommendations". In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD '19. Amsterdam, Netherlands: Association for Computing Machinery, 2019, 1241–1258.

[8] Jianling Gao et al. “Automatic index selection with learned cost estimator”. In: *Information Sciences* 612 (2022), pp. 706–723.

[9] Benjamin Hilprecht and Carsten Binnig. “Zero-shot cost models for out-of-the-box learned cost prediction”. In: *Proc. VLDB Endow.* 15.11 (2022), 2361–2374.

[10] IBM. *What is multi-tenant (or multitenancy)?* [https://www.ibm.com/topics/multi-tenant].

[11] Ivo Jimenez et al. “Benchmarking Online Index-Tuning Algorithms”. In: *IEEE Data Eng. Bull.* 34.4 (2011), pp. 28–35.

[12] Hai Lan, Zhifeng Bao, and Yuwei Peng. “An Index Advisor Using Deep Reinforcement Learning”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020, 2105–2108.

[13] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[14] Guy Lohman. “Is Query Optimization a “Solved” Problem?” In: *ACM SIGMOD Blog* (2014). [https://wp.sigmod.org/?p=1075].

[15] Tanu Malik et al. “Adaptive Physical Design for Curated Archives”. In: *Scientific and Statistical Database Management*. Ed. by Marianne Winslett. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 148–166.

[16] R. Malinga Perera et al. “DBA bandits: Self-driving index tuning under ad-hoc, analytical workloads with safety guarantees”. In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. Los Alamitos, CA, USA: IEEE Computer Society, 2021, pp. 600–611.

[17] R. Malinga Perera et al. “HMAB: self-driving hierarchy of bandits for integrated physical database design tuning”. In: *Proc. VLDB Endow.* 16.2 (2022), 216–229.

[18] R. Malinga Perera et al. “No DBA? No Regret! Multi-Armed Bandits for Index Tuning of Analytical and HTAP Workloads With Provable Guarantees”. In: *IEEE Transactions on Knowledge amp; Data Engineering* 35.12 (2023), pp. 12855–12872.

[19] Kai-Uwe Sattler, Ingolf Geist, and Eike Schallehn. “QUIET: continuous query-driven index tuning”. In: *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*. VLDB ’03. Berlin, Germany: VLDB Endowment, 2003, 1129–1132.

[20] Rob Schapire and Maryam Bahrani. *Online Linear Regression*. https://www.cs.princeton.edu/courses/archive/spring18/cos511/scribe_notes/0411.pdf. [Lecture Notes for COS 511: Theoretical Machine Learning]. 2018.

[21] Karl Schnaitter and Neoklis Polyzotis. "Semi-automatic index tuning: keeping DBAs in the loop". In: *Proc. VLDB Endow.* 5.5 (2012), 478–489.

[22] Karl Schnaitter et al. "On-Line Index Selection for Shifting Workloads". In: *2007 IEEE 23rd International Conference on Data Engineering Workshop*. 2007, pp. 459–468.

[23] Tarique Siddiqui et al. "DISTILL: low-overhead data-driven techniques for filtering and costing indexes for scalable index tuning". In: *Proc. VLDB Endow.* 15.10 (2022), 2019–2031.

[24] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts*. 6th ed. New York: McGraw-Hill, 2010. Chap. 2.

[25] Aleksandrs Slivkins. *Introduction to Multi-Armed Bandits*. 2024. arXiv: 1904.07272 [cs.LG].

[26] Ji Sun and Guoliang Li. "An end-to-end learning-based cost estimator". In: *Proc. VLDB Endow.* 13.3 (2019), 307–319.

[27] Adam Wierman. "A tale of two metrics: competitive ratio and regret". In: (2014). [https://rigorandrelevance.wordpress.com/2014/10/13/a-tale-of-two-metrics-competitive-ratio-and-regret/].

[28] Wentao Wu et al. "Budget-aware Index Tuning with Reinforcement Learning". In: *Proceedings of the 2022 International Conference on Management of Data*. SIGMOD '22. Philadelphia, PA, USA: Association for Computing Machinery, 2022, 1528–1541.

[29] Tao Yu et al. "Refactoring Index Tuning Process with Benefit Estimation". In: *Proc. VLDB Endow.* 17.7 (2024), 1528–1541.

[30] Lukas Zierahn et al. "Nonstochastic Contextual Combinatorial Bandits". In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. Ed. by Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent. Vol. 206. Proceedings of Machine Learning Research. PMLR, 2023, pp. 8771–8813.