International Journal of Engineering, Science and Mathematics
Vol. 10 Issue 03, March 2021,
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage: http://www.ijesm.co.in, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell’s Directories of Publishing Opportunities, U.S.A

PREDICTIVE MODELING FOR QUERY PERFORMANCE TUNING IN DATABASE MANAGEMENT SYSTEMS

Moolchand
Research Scholar:- Sunrise University, Alwar, Rajasthan

Dr Kushum Rajawat
Assistant professor Sunrise University, Alwar, Rajasthan

Abstract

The execution inertness of applicant plans is inadequately anticipated by logical expense models, which are habitually utilized by streamlining agents to look at up-and-comer plan costs. This examination researches the materialness and utility of cutting edge learning-based models, which have of late been effectively used to a scope of predicative issues, as a seriously encouraging way to deal with QPP. The runtime conduct of contemporary database management frameworks (DBMS) can be changed utilizing an enormous number of adjustable handles. It tends to be improved by appropriately arranging these handles for an application's responsibility. the DBMS's adequacy and effectiveness. Be that as it may, because of their intricacy, DBMS tuning much of the time requires a lot of work from proficient database managers (DBAs). When contrasted with gifted DBAs, late work on computerized tuning techniques utilizing AI (ML) has shown to give better performance. Notwithstanding, these ML-put together methods were tried with respect to fake jobs with not many opportunities for changing, hence it is muddled if they could be as successful in a certifiable setting.

Keywords: *Query Performance,Tuning,Management,Predictive,Database*

1. INTRODUCTION

In the world of database management systems (DBMS), optimizing query performance is crucial for ensuring effective and fast data retrieval. Organizations must optimize their database systems to handle complicated queries and provide results quickly as data quantities continue to expand dramatically. Predictive modelling, a useful technique that makes use of past data and statistical algorithms to forecast query performance and direct tuning efforts, has become available as a solution to this problem. In predictive modelling, correlations between numerous elements influencing query performance, such as database structure, query complexity, hardware configuration, and workload characteristics, are captured mathematically. These models are developed using historical data that includes the times at which queries were executed together with the corresponding environmental and query-specific characteristics. The models can accurately forecast how queries will perform in various settings by analyzing this data to find patterns, correlations, and dependencies.

To proactively detect potential bottlenecks, inefficiencies, or suboptimal setups within the DBMS environment is the main goal of predictive modelling for query performance tuning. Database administrators (DBAs) can decide on system configuration, query optimization techniques, and resource allocation to attain optimal performance levels by properly forecasting the performance of queries. Comparing predictive modelling to conventional methods for query performance adjustment, there are a number of benefits. DBAs can first and foremost adopt a pro-active stance by spotting and resolving potential performance issues before they worsen. Predictive models can identify underlying patterns and trends by using previous data, allowing for targeted optimization efforts as opposed to depending simply on reactive tuning techniques. Predictive modelling also enables scenario analysis and what-if simulations, allowing DBAs to weigh the effects of various tuning techniques or adjustments to workload patterns.

DBAs can test the efficacy of suggested optimization's without interfering with the environment of live production by replicating these scenarios. Additionally, by offering quantitative insights into the anticipated performance benefits brought on by particular tuning operations, predictive modelling supports decision-making. By prioritizing their work based on the potential influence on query execution times, DBAs are given the ability to maximize resource allocation and reduce performance-related expenditures. For the purpose of query performance adjustment, a variety of statistical and machine learning techniques can be used for predictive modelling. To create predictive models based on historical data, regression analysis, time series analysis, and machine learning techniques including decision trees, random forests, and neural networks are frequently used.

2. REVIEW OF LITREATURE

A thorough investigation into predictive modelling for self-tuning database systems is presented by Baeza-Yates et al. (2006). The authors suggest a method for forecasting future query performance using historical query execution data. They investigate the development of prediction models using statistical and machine learning methods, taking into account variables like query complexity, database structure, and workload characteristics. The study emphasises the value of proactive performance tuning and shows how their method effectively improves query performance.

An overview of query optimisation strategies in relational database systems is given by Chaudhuri (2005). The study provides insightful information about the broader context of query optimisation, despite not being exclusively focused on predictive modelling. It talks about several optimisation algorithms, cost estimation methods, and plan selection tactics. To incorporate predictive modelling into the query optimisation process, it is essential to comprehend these optimisation strategies.

A study on predictive modelling that is especially suited for query performance in parallel database systems is presented by Gupta et al. in 2003. The authors suggest an approach that gauges workload and system factors to forecast query execution time. To create prediction models, they use machine learning techniques like decision trees. The study illustrates the efficiency of their approach in enabling proactive performance adjustment in parallel database systems and properly projecting query performance.

The book by Snodgrass is a thorough resource on database tuning, complete with principles, experiments, and troubleshooting methods. Although it is not only concerned with predictive modelling, it does encompass significant ideas and tactics pertinent to query performance optimisation. Different tuning methodologies, query optimisation tactics, and performance analysis methods are covered in the book. It is a useful resource for comprehending predictive modeling's broader context in the area of database tuning.

A study by Witkowski et al. (2009) uses machine learning approaches to create predictive models for database system performance modelling. The authors suggest an approach for forecasting query response times that combines past query execution data with system and workload variables. To build prediction models, they use decision trees and regression analysis. The study exhibits the potential for precise query response time prediction and emphasises the efficiency of machine learning in capturing the correlations between numerous parameters influencing query performance.

Neural networks are suggested by Wong and Leung (2002) for application in distributed database systems' predictive modelling of query performance. The authors concentrate on the difficulty of foreseeing how long sophisticated queries would take to execute in a distributed system. In order to anticipate query response times precisely, they describe a neural network-based methodology that takes into consideration query attributes, system factors, and workload characteristics. The study shows how well neural networks can capture the complex interactions between different variables and highlights the promise of predictive modelling for improving query efficiency in distributed database systems.

Selectivity estimate, a key component of query performance adjustment, is a problem that Viglas (2000) addresses. In order to assess query selectivity and provide an accurate prediction of query execution durations, the study presents an adaptive sampling technique. The method intelligently samples the data and then modifies the sample size in real time to capture the distributional properties of the underlying data. The study demonstrates how adaptive sampling can enhance selectivity estimation and, as a result, query performance.

3. AUTOMATED TUNING FIELD STUDY

The previously mentioned issues stress the weaknesses in current examinations of setup change strategies. These outlines support the need for a more exhaustive examination to decide if robotized tuning systems are valuable for DBMS establishments in reality.

We plan to decide the compromises of ML-based calculations and the sum to which human management matters on the off chance that computerized tuning ends up being functional in these organizations.

At the global bank Société Générale (SG), we assessed the Otter Tune system in 2020. Prophet is utilized by SG for most of their database applications on confidential cloud framework. For DBMS executions that utilization a calibrated arrangement in view of the expected responsibility (for instance, OLTP versus OLAP), they offer self-administration provisioning. These Prophet organizations are managed by a gathering of proficient DBAs with handle tuning experience. Our field study's goal is to decide if robotized tuning can support a DBMS's performance far in excess of what their DBAs can do physically.

We give the particulars of our OtterTune arrangement at SG in this segment. We start by illustrating the distinctions between the objective database responsibility and fake benchmarks. We then, at that point, proceed to talk about SG's working climate and the troubles we experienced while dealing with a robotized tuning administration.

3.1 Objective Database Programmed

For SG's IT foundation, an inside issue following apparatus called Ticket Tracker gave the information and responsibility follow that we utilized in our examination. The fundamental elements of ticket Tracker are equivalent to those of other well known project management apparatuses like Atlassian Jira and Mozilla Bugzilla. The work tickets submitted all through the whole association are followed by this application. Since SG utilizes around 140,000 individuals around the world, the responsibility examples and query appearance rate for ticket Tracker are to a great extent consistent all through the functioning week, 24 hours per day. SG as of now uses Prophet v12.1 to control ticket Tracker. To consolidate the database and query follow data, we made special announcing instruments. From this examination, we currently give a significant level outline of ticket Tracker.

Table 1: Query Plan Operators: The proportion of each operator type in the queries in the ticket Tracker workload

Operator Type	**% Of Queries**
Table Access By Index Rowed	42%
Index Range Scan	30%
Index Unique Scan	20%
Sort Order By	15%
Table Access Full	16%
All Others	20%

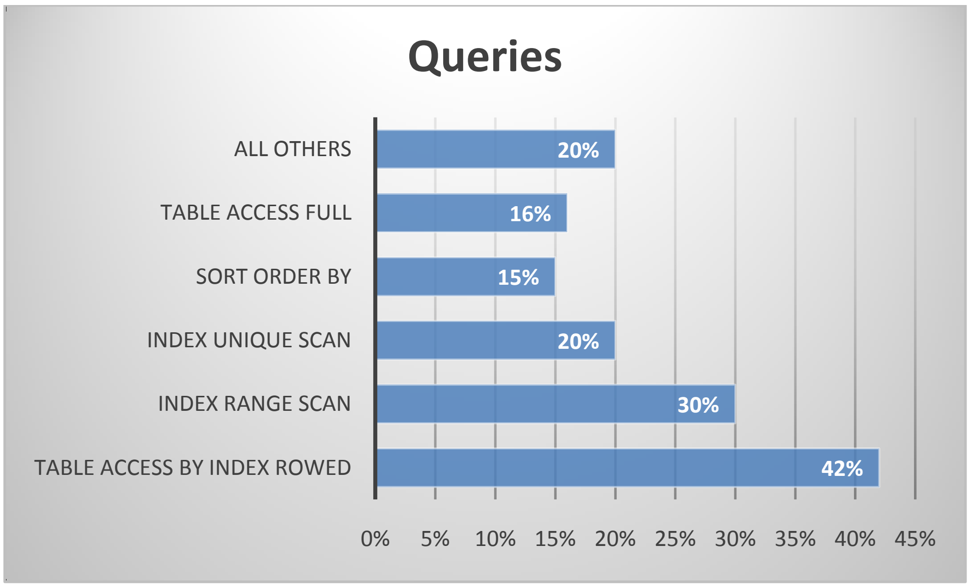

Figure 1: Query Plan Operators: The proportion of each operator type in the queries in the ticket Tracker workload

Database: Utilizing the Prophet Recuperation Supervisor instrument, we took a depiction of the TicketTracker database from its creation server.

The database's uncompressed size is roughly 1.1 TB, of which 54% are huge articles (Hurls), 27% are table items, and 19% are table files. This Throw information is imperative since no earlier work has analyzed the component of DBMS tuning that Prophet uncovered handles that influence how it keeps up with Hurls.

3.2 Deployment

In SG's confidential cloud, we set up five different Prophet v12.2 frameworks to have the TicketTracker database and responsibility. The equipment arrangement was equivalent to what was utilized for the creation occasion. A virtual machine (VM) with 12 vCPUs (Intel Xeon central processor E5-2697v4 @ 2.30 GHz) and 64 GB Smash controls every DBMS occurrence. The VMs were set up to keep in touch with a NAS shared plate that is available in similar server farm. The typical read and compose latencies for this capacity are 6.7 ms and 8.3 ms, separately, as demonstrated in our prior analyze in Figure 3.

Every Prophet occurrence's underlying handle design is browsed a bunch of pre-tuned setups that SG use all through their entire armada. For putting in new DBMSs, the SG IT group offers their staff a self-administration online gateway. A client should demonstrate the expected responsibility that the DBMS will uphold as well as picking the equipment design of the new DBMS, (for example, the quantity of central processor centers and Smash), like OLTP, OLAP, or HTAP. The handle setup that has been pre-tuned by the SG managers for the picked responsibility type is introduced by the provisioning framework. Indeed, even while these arrangements perform better compared to Prophet's default settings, they just change 4-6 handles and are as yet not upgraded for the jobs of the different applications. Accordingly, the DBA further altered a portion of the pre-tuned design's handles for the TicketTracker responsibility, including one that improves Hurl performance.3

The OtterTune tuning supervisors and regulators were undeniably introduced in similar server farm as the Prophet DBMSs. Every part was worked in a Docker compartment with eight virtual central processors and 16 GB of Slam. There is a particular OtterTune tuning director relegated to every DBMS occasion. This partition quits preparation information assembled during one meeting from being utilized during another, which will affect the calculations' viability and combination rate.

4. TUNING ALGORITHMS

Understanding how the DBMS design tuning methods that have been presented as of late respond in genuine circumstances and under what conditions one performs better compared to others is our point. To do this, we upgraded Otter Tune to remember support for various calculations for its tuning administrator. Subsequently, we can send a singular stage without fundamentally changing the tuning methodology.

We currently go over the three strategies that we tried: Profound Brain Organization (DNN), Profound Deterministic Approach Angle (DDPG), and Gaussian Interaction Relapse (GPR). Despite the fact that there are different calculations that utilization query information to coordinate the hunt cycle they can't be utilized at SG in light of the fact that to protection issues since the questions contain information that can be utilized to recognize explicit clients. The reason for this paper does exclude methods for anonymizing this information.

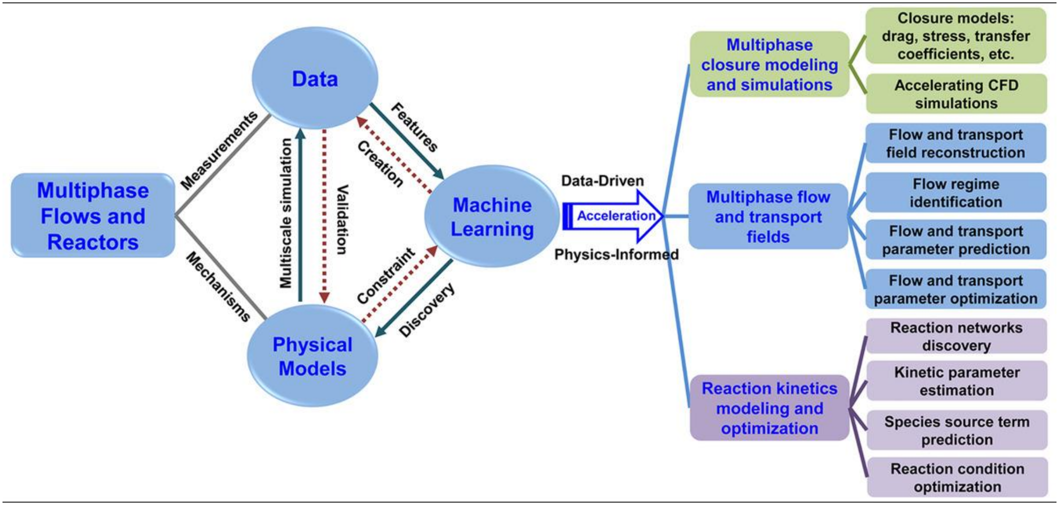
Data
Multiphase Flows and Reactors
Machine Learning
Physical Models
Measurements
Mechanisms
Multiscale simulation
Validation
Features
Creation
Constraint
Discovery
Data-Driven
Acceleration
Physics-Informed
Multiphase closure modeling and simulations
Closure models: drag, stress, transfer coefficients, etc.
Accelerating CFD simulations
Multiphase flow and transport fields
Flow and transport field reconstruction
Flow regime identification
Flow and transport parameter prediction
Flow and transport parameter optimization
Reaction kinetics modeling and optimization
Reaction networks discovery
Kinetic parameter estimation
Species source term prediction
Reaction condition optimization

Figure 2: The GPR/DNN Tuning Pipeline totals the crude information from each past work and thinks about it to the ideal responsibility.

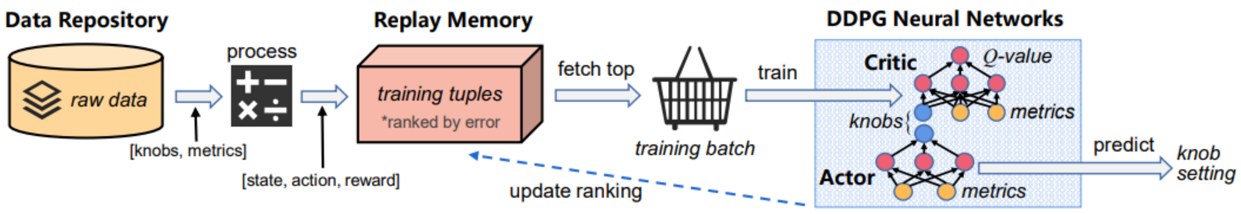

Figure 3: The crude information is changed into states, activities, and prizes, and afterward took care of into the replay memory utilizing the DDPG Tuning Pipeline.

4.1 GPR — OtterTune (2017)

In view of the first calculation upheld by Otter Tune, we executed GPR. The distance between the test point and each preparing point is determined involving a Gaussian interaction as an earlier over capability. The calculation gauges the test point worth and vulnerability utilizing piece capabilities.

Otter Tune's GPR pipeline has two phases, as displayed in Figure 2. Otter Tune's information vault's handle and metric information are ready in the main stage, called information pre-handling. The subsequent stage, known as "Handle Suggestion," picks values for the handles.

Pre-handling of information: The objective of this stage is to make the estimations less layered and distinguish the key tuning handles. The assistance makes handle designs for the objective DBMS utilizing the consequence of this stage. This stage is occasionally shown behind the scenes to Otter Tune. The length of every summon differs as per the amount of tests and DBMS measurements.

The Information Pre-Handling stage at first picks a gathering of DBMS measurements that best catch the performance changeability and unmistakable characteristics of a specific responsibility. The measurements are separated into a more modest gathering of elements by the calculation utilizing a dimensionality decrease method called factor examination, which likewise catches the relationship examples of the first factors. The coefficients of each component, which are direct blends of the first factors, can be grasped similarly as the coefficients of straight relapse. Subsequently, the elements can be positioned by how much changeability in the first information they represent. At long last, the calculation picks one delegate measure from each gathering and uses k-implies bunching to bunch the parts with comparative connection designs.

Finding the responsibility that Otter Tune tuned in the past that most intently looks like the ongoing responsibility is the initial step. This earlier data is utilized to "bootstrap" the new meeting. The strategy does this by foreseeing the measurement upsides of the objective DBMS's responsibility given the positioned posting of handles utilizing the result information from the initial step.

The information from the objective responsibility and the responsibility that is the most comparative is then utilized by the help to make a GPR model. The model results the pair (y,u) involving the planned goal esteem (y) and the vulnerability esteem (u) for the predefined cluster of handles (x). The calculation adds y and u to decide the upper certainty limits (UCB). From that point forward, it applies slope climb to the UCB to recognize the handle settings that ought to bring about a positive goal esteem. The handle design for the objective DBMS is prompted in view of the greatest worth among those nearby optima, not entirely settled by performing slope plummet to get the neighborhood ideal from each beginning stage.

How the calculation handles the compromise between investigation (i.e., assembling new information to construct the model) and double-dealing (i.e., endeavoring frantically to prevail on the objective) is a vital issue in this cycle. Otter Tune changes the UCB's vulnerability weight to control investigation and abuse.

5. EVALUATION

The results of our correlation of the previously mentioned tweaking methods for SG's Prophet establishment on ticket Tracker are currently introduced.

Since irregular examining methods are direct yet shockingly effective, they are utilized as serious baselines for assessing improvement calculations Latin Hypercube Inspecting (LHS) an arbitrary testing procedure, fills in as the establishment for our examination. LHS is a space-filling strategy that tries to uniformly disperse test focuses among every possible worth. In high layered spaces, these strategies are commonly more compelling than gullible irregular testing, especially while gathering not many examples in contrast with the all out number of potential qualities.

We start by playing out a fundamental examination of the performance measures for SG's current circumstance's fluctuation. This clarification is expected to explain how we complete our analyses and how we dissect their results in the following parts.

5.1 Performance Variability

We introduced the Prophet DBMS on various virtual machines (VMs) to execute the tuning meetings in equal in light of the fact that each tuning meeting in our examinations requires a few days to finish. During this time, our virtual machines keep on working on similar actual machines, however different occupants on these machines or in a similar rack might change. Running a DBMS in virtualized settings with shared capacity, as was shrouded in Segment 2.2, could bring about surprising varieties in the framework's performance between occasions with similar equipment distributions along with inside a similar example.

We estimated the performance of our VMs once every week for a time of a half year to more readily fathom the greatness of this unusualness in SG's server farm. We execute the 10-minute piece of the ticket Tracker responsibility with the default settings for SG. Figure 8's discoveries show the development of the DB Time measure for each VM case. The principal finding from this information is that, regardless of having a similar arrangement and responsibility as the DBMS, the performance of the DBMS on a similar VM could differ by as much as 4. For example, the DB Time for VM02 in July is higher than whatever we kept in June. That's what the following finding is, even inside a humble window, VMs' overall performance can change.

We imagine that the common circle stockpiling's inertness spikes are at fault for these problematic outcomes. Figure 9 shows the computer processor active time and I/O delay for one VM running the DBMS during a tuning meeting. These discoveries show a connection between expansions in I/O dormancy (three spots are featured) and a decrease in the DBMS's performance. For this situation, the design was steady in light of the fact that the calculation had joined at this phase of the tuning meeting. Accordingly, almost certainly, outside factors autonomous of the DBMS are to be faulted for these idleness spikes.

Table 2: Portrayal Runtime estimations of DBMS performance with computer chip use and I/O dormancy. Impact of I/O Dormancy Spikes.

DB Times(s)	Tunning iterations
2.3	3.2
3.6	3.9
4.2	5.2
5.3	6.2
4.6	5.9
6.2	7.1
7.3	8.2

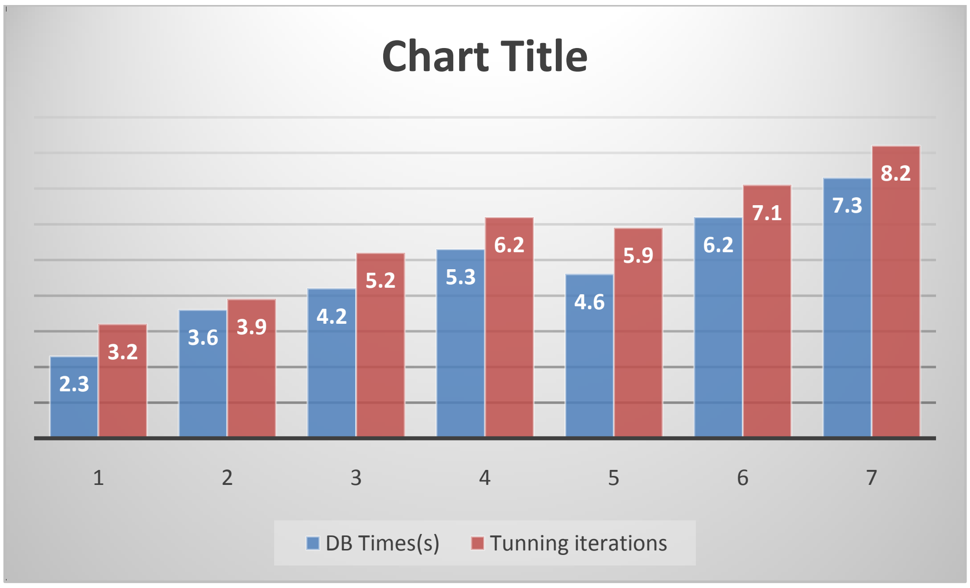

Figure 4: Portrayal Runtime estimations of DBMS performance with computer chip use and I/O dormancy. Impact of I/O Dormancy Spikes.

Because of these motions, hard to analyze tuning meetings happen on a few virtual machines (VMs) or even on a similar VM however at different times. Considering this, we put a ton of exertion into planning our investigations with the goal that we could give sagacious outcomes. In this work, we direct each of our tests utilizing a similar procedure. The length of each tuning meeting is 150 emphasess. Contingent upon how well the DBMS is arranged, every emphasis can require as long as 60 minutes. Therefore, it required three to five days to complete every meeting.

We perform three tuning meetings for every calculation in a specific trial for each condition being tried. Following that, we assemble the advanced arrangements from every one of the meetings and execute them multiple times each, one after the other, on three particular virtual machines (VMs). All in all, we run every arrangement multiple times for each VM, for a sum of nine runs. Because of the way that a VM's performance changes with time, running the settings all together inside a similar time period is significant. It likewise empowers us to decide the overall enhancements between them involving the indistinguishable DB Time estimation for the SG default arrangement. By running the designs on three free VMs, one VM's unnecessary commotion is forestalled.

5.2 Tuning Knobs Selected by DBA

In this underlying examination, the nature of the setups that the tuning calculations produce as they tune more handles is surveyed. Notwithstanding the way that Prophet uncovered more than 400 handles, we just permit a sum of 40 handles to be changed for two reasons.

To start with, we need to look at the amount more really ML calculations request handle significance than a DBA-chose positioning.

It is ludicrous to anticipate that a human should pick between in excess of 40 tuning handles, and the outcomes will be whimsical. The subsequent explanation is to abbreviate the time expected for the calculations to join in light of the fact that the trouble of tuning calculations increments with the quantity of handles. Since the TicketTracker responsibility emphasizes once at regular intervals, it can require a long time for the models to merge. We consequently consider a limit of 40 handles that the DBA picked and requested relying upon their expected impact on the performance of the DBMS.

Table 3: The three most crucial dials for the ticket Tracker workload are listed below, together with the best observed and default values for each.

DB_CACHE_SIZE	5 GB	30-40 GB
DB_32K_CACHE_SIZE	11 GB	20 GB
OPTIMIZER_FEATURES_ENABLE	v11.2.0.4	v12.2.0.1

The ML-based calculations don't involve information from prior tuning meetings for this arrangement of investigations. All things considered, we utilize 10 LHS-created arrangements to bootstrap their models.

While streamlining 10, 20, and 40 handles by VM, the best (i.e., most noteworthy performing) of three designs created per calculation shows the improvement in DB Time over the SG default setup in Figure 10. Albeit the outright measures contrast, the positions of the calculations' overall performance are something similar across all of the VMs.

Figure 11 presentations, for the advanced arrangements created by the strategies, the typical performance improvement over the three VMs. Each bar's dim and light regions compare to the calculation's base and best performance, separately.

We physically went through every setup to decide the three Prophet handles that have the greatest impacts when the calculations don't precisely design them to comprehend the reason why the arrangements act in an unexpected way. Table 2 records the upsides of the handles in the SG default design as well as the best value(s) we had the option to see all through our tests.

The initial two direct the size of the essential cradle stores in DBMSs.

The DBMS utilizes 8 KB cradles for common table information in one of these reserves and 32 KB supports for Throw information in the other. The third handle, an unmitigated variable with seven potential qualities, enacts enhancer usefulness in view of a Prophet discharge.

Table 4: The performance improvement of the best setup for every calculation running on particular VMs in contrast with the performance of the SG default design estimated toward the beginning of the tuning meeting.

Knobs	% Improvement (DB Time)
GPR	2.3
DNN	3.3
DDPG	4.2
DDPF++	4.9
LHS	5.3

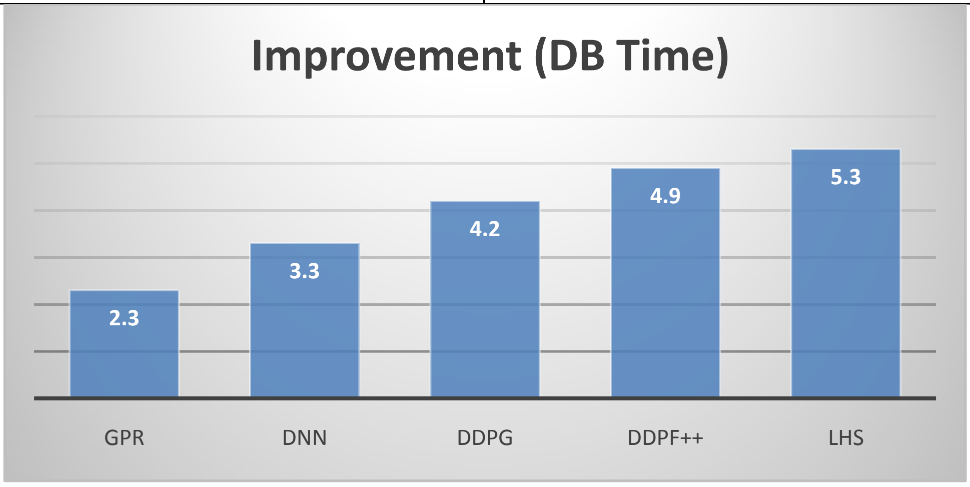

Figure 5: The performance improvement of the best setup for every calculation running on particular VMs in contrast with the performance of the SG default design estimated toward the beginning of the tuning meeting.

Be that as it may, GPR is inclined to becoming caught in neighborhood minima, and when it combines, it quits investigating and thus doesn't keep on progressing. The best noticed scopes of the affecting handles from Table 2 should be investigated for GPR to work at its ideal. We likewise notice that the underlying examples run toward the start of the tuning meeting affect its performance.

6. CONCLUSION

In this review, involving a genuine responsibility on a Prophet establishment in a venture setting, we completely assessed AI based DBMS handle tuning methods. To analyze three state of the art ML calculations one next to the other, we conveyed them in the Otter Tune tuning administration. Our discoveries showed the way that these calculations could give handle mixes that beat those made by a human master by up to 45%, yet the performance was impacted by the quantity of tuning handles and the contribution of human specialists in the handle determination process. Predictive displaying has turned into a helpful device for database management frameworks (DBMS) to deal with the issues of effectively overseeing large information volumes and refined questions. Predictive models can exactly estimate query performance and direct improvements endeavors by using past information and applying factual and AI calculations.

Predictive displaying helps database managers (DBAs) to detect potential bottlenecks and shortcomings in the DBMS framework before they become tricky through proactive performance tweaking. Predictive models can help DBAs in arriving at all around informed conclusions about framework arrangement, query advancements strategies, and asset assignment by looking at past information and seeing examples and connections.

REFRENCES

1. *Behm, "Predicting performance through machine learning," Proceedings of the 2013 IEEE 29th International Conference on Data Engineering, pp. 2-7, 2013.*
2. *Gupta, I. Singh, and A. Sivasubramaniam, "Predictive modeling for query performance in parallel database systems," Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 605-606, 2003.*
3. *H. Baeza-Yates, C. Hurtado, M. Mendoza, P. Mora, and P. Vassiliadis, "Predictive modeling for self-tuning database systems," ACM Transactions on Database Systems (TODS), vol. 31, no. 3, pp. 1055-1082, 2006.*
4. *James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research 13, 10 (2012), 281–305.*

5. *Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. 1984. Classification and Regression Trees. CRC press.*
6. *P. Gao, Y. Chen, and J. Xu, "A machine learning approach to database query performance prediction," Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, pp. 1060-1071, 2011.*
7. *P. Ipeirotis, E. Jeng, H. Wang, and J. Michael, "Query workload-driven database design," Proceedings of the 25th International Conference on Data Engineering, pp. 864-875, 2009.*
8. *R. T. Snodgrass, "Database tuning: principles, experiments, and troubleshooting techniques," Morgan Kaufmann, 2002*
9. *S. Chaudhuri, "An overview of query optimization in relational systems," Proceedings of the 21st International Conference on Data Engineering, pp. 34-45, 2005.*
10. *S. D. Viglas, "Selectivity estimation through adaptive sampling," Proceedings of the 26th International Conference on Very Large Data Bases, pp. 346-357, 2000.*
11. *S. F. Wong and T. K. Leung, "Predictive modeling for query performance in distributed database systems using neural networks," Journal of Parallel and Distributed Computing, vol. 62, no. 6, pp. 996-1013, 2002.*
12. *Sanjay Agrawal, Vivek Narasayya, and Beverly Yang. 2004. Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. 359–370.*
13. *Surajit Chaudhuri and Vivek Narasayya. 1998. AutoAdmin "What-If" Index Analysis Utility. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. 367–378.*
14. *Surajit Chaudhuri and Vivek Narasayya. 2007. Self-Tuning Database Systems: A Decade of Progress. In Proceedings of the 33rd International Conference on Very Large Data Bases. 3–14.*
15. *Witkowski, P. Morvan, and S. Uhrig, "Machine learning for predictive performance modeling of database systems," Proceedings of the 2009 International Conference on Machine Learning and Applications, pp. 353-358, 2009.*