Workload-Aware Performance Tuning for Autonomous DBMSs

Zhengtong Yan
University of Helsinki, Finland
zhengtong.yan@helsinki.fi

Jiaheng Lu
University of Helsinki, Finland
jiaheng.lu@helsinki.fi

Naresh Chainani
Amazon AWS, USA
nareshkc@amazon.com

Chunbin Lin
Amazon AWS, USA
lichunbi@amazon.com

***Abstract*—Optimal configuration is vital for a DataBase Management System (DBMS) to achieve high performance. There is no one-size-fits-all configuration that works for different workloads since each workload has varying patterns with different resource requirements. There is a relationship between configuration, workload, and system performance. If a configuration cannot adapt to the dynamic changes of a workload, there could be a significant degradation in the overall performance of DBMS unless a sophisticated administrator is continuously re-configuring the DBMS. In this tutorial, we focus on autonomous workload-aware performance tuning, which is expected to automatically and continuously tune the configuration as the workload changes. We survey three research directions, including 1) workload classification, 2) workload forecasting, and 3) workload-based tuning. While the first two topics address the issue of obtaining accurate workload information, the third one tackles the problem of how to properly use the workload information to optimize performance. We also identify research challenges and open problems, and give real-world examples about leveraging workload information for database tuning in commercial products (e.g., Amazon Redshift). We will demonstrate workload-aware performance tuning in Amazon Redshift in the presentation.**

***Index Terms*—Workload, Classification, Forecasting, Autonomous, DBMS, Tuning**

I. Motivation

Database configurations, such as physical and logical design as well as resource allocation (e.g., CPU, memory and IO resources), are vital for DBMSs to achieve high performance. However, the rapid growth in data and the high complexity of time-varying workloads make the configuration tasks extremely challenging, especially when organizations are moving their DBMSs to managed services in cloud environments. An ideal solution is the autonomous DBMS which is expected to automatically and constantly tune itself by adapting to workload changes [1].

The behavior of an autonomous DBMS can be described using the following formula which demonstrates a mechanism for workload-aware performance tuning [2]:

$$f : \text{configuration} \times \text{workload} \rightarrow \text{performance} \qquad (1)$$

where the configuration consists of the hardware setup, software setup, database physical and logical design, etc, the workload features include workload types, workload shifts, workload patterns (e.g., periodic pattern and arrival patterns), and the performance is measured by one of the metrics (response time, throughput, reliability, etc.) or their combination.

However, there are (at least) three significant challenges on the way to workload-aware performance tuning for an autonomous DBMS: **(i) Workload's diversity and heterogeneity:** A DBMS may have a large number of instances and each instance might encompass various types of workload. Different types have different features, for instance, transactional workloads are comprised of short-running transactions which modify few records while analytical workloads are usually long-running read-only queries which process considerable amount of data [3]. **(ii) Workload's dynamic characteristics:** As new users and applications appear ever more frequently, workload size and patterns may constantly evolve over time. For example, the database might be extended with more users, thereby adding a large number of queries to the workload in an unpredictable way. The performance of a DBMS could rapidly degrade when previously tuned configurations cannot match the requirements of incoming queries. **(iii) Workload's complex influence on performance:** It is difficult to quantify the amount of hardware resources for changing workloads and decide how much performance is compromised for a given workload [4]. Moreover, searching the optimal configuration for a given workload is usually an NP-Hard problem as the solution lies in high-dimensional and continuous space.

Table I summarizes the categories and literature on workload-aware performance tuning for an autonomous DBMS. There are three major topics: (i) Workload Classification, (ii) Workload Forecasting, and (iii) Workload Tuning. The goal is to enable the DBMSs to continuously and automatically adjust databases' configurations by analyzing the evolving workload and making optimal decisions for identified workload types. In this tutorial, we will first provide an overview and motivating examples of workload-aware performance tuning for autonomous DBMSs. Next, we will introduce the different categories to classify workloads, workload forecasting, and workload-aware tuning methods by presenting the essential characteristics of each category. Finally, we will highlight real-world applications and systems for workload-aware tuning and identify research challenges.

To the best of our knowledge, this is the first tutorial to discuss the state-of-the-art research and industrial trends in the context of workload-aware performance tuning for autonomous DBMSs. The previous tutorials only focus on one aspect of database tuning (e.g., adaptive replication and partitioning [33], parameter tuning [34]) regardless of the

2375-026X/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00267

TABLE I
AN OVERVIEW OF MAIN RESEARCH CATEGORIES AND LITERATURE ON WORKLOAD-AWARE PERFORMANCE TUNING

	Category	Literature
Workload Classification	Unsupervised learning	[5], [6]
	Supervised learning	[7], [8]
	Experiment-driven	[9], [10]
	Reasoning	[11], [12]
Workload Forecasting	Machine learning	[13]–[15]
	Time-series analysis	[8], [16]–[18]
	Stochastic process modeling	[19]–[22]
Workload-based Tuning	Machine learning	[23]–[25]
	Reinforcement learning	[26]–[28]
	Rule-based	[29], [30]
	Cost modeling	[31], [32]

workload information. This tutorial, on the other hand, focuses on how to obtain the workload information and tune a DBMS based on the estimated workloads.

II. TUTORIAL CONTENTS

The tutorial is organized in a sequence of six sections, whose contents are summarized as follows.

A. Section I: Motivation and background

At the beginning of this tutorial, we will review the challenges of autonomous database tuning, especially from the high complexity of time-varying workloads in cloud environments. An autonomous DBMS needs to tune itself based on the dynamic workload information to obtain an optimal performance.

B. Section II: Workload Classification

The first and key step towards an autonomous DBMS is the ability to accurately classify the workloads because the workload types are an important criterion for database performance tuning. The types of workloads may include Online Analytical Processing (OLAP), Online Transactional Processing (OLTP), Decision Support System (DSS), Business Intelligence (BI), and even hybrid and interactive workloads (e.g., mixed fast transactions and complex analytical queries that run concurrently and interact with each other [35]).

The existing classification methods can be divided into the following four groups: **(i) Unsupervised learning**: The aim of this method is grouping the workload into general classes by exploiting some metrics (e.g., distance function, resource usage) to measure the similarity between the workloads. Examples are clustering [5] [24] and SemiDiscrete Decomposition [6]. **(ii) Supervised learning**: Workload classification can be solved by machine learning models, such as Classification & Regression Tree [7], and Decision Tree Induction [8]. **(iii) Experiment-driven**: This approach utilizes experimental techniques, such as planning experiments [9] and experimental sampling [10], to recognize workload type and reconfigure resources accordingly. **(iv) Reasoning**: The workload classification can also be solved through reasoning with existing solutions that are stored in the case-base repository. This approach is called Case-Based Reasoning (CBR) [11] [12] which can adapt new cases and does not require retraining of data.

C. Section III: Workload Forecasting

Database workloads in the real-world usually change dynamically. To be totally autonomous, the DBMSs must be capable of handling the dynamic workloads by predicting the workload changes based on the historical data [13]. The estimated workload characteristics and changes may consist of: (i) **expected arrival rate of queries**, (ii) **running time of each query**, (iii) **structural and periodic patterns**, (iv) **workload shifts**, (v) **the next transaction or query**, (vi) **memory usage of each query**, etc.

The existing approaches can be divided into the following three categories of approaches can be applied to predict the above characteristics:

- **Machine learning:** The workload prediction could be posed as a classification or regression problem in which data-driven methods predict the workload changes directly from observed data. Representative models are Ensemble Learning [13], Deep Q-Network [14], and LSTM-based auto-encoder [15].
- **Time-series analysis:** Both the short- and long-term workloads can be seen as a time sequence, whose future values and periods can be estimated by time-series analysis techniques which may be broken down into two main areas:
  - *Time domain analysis* uses techniques such as Sparse Periodic Auto-Regression (SPAR) [16], Moving Averages (MA), Polynomial Regression (PR) [8], and Auto Regressive Integrated Moving Average(ARIMA) [17].
  - *Frequency domain analysis* utilizes methods such as Discrete Fourier Transform (DFT) and Interval Analysis [18] to transfer the workload series into a frequency domain to find some complex periodic patterns.
- **Stochastic process modeling:** Different from the time-series analysis, random process models focus on the probability properties of workload, such as Markov properties [19]–[22].

D. Section IV: Workload-based Tuning and Scheduling

1) Workload-based Tuning: Optimal database physical and logical design, as well as resource allocation, are affected heavily by the dynamic changes of workloads [20] [36]. This is because there is a certain relationship between database (physical/logical) states, workloads, and database resources. Given workload types and changes, a model is then needed to decide the tuning actions. Two critical problems must be solved: 1) how to extract and utilize the workload information, and 2) how to find the optimal tuning actions (e.g., what sort of indexes are the most effective and how much amount of hardware resource is needed for future workloads).

The existing methods can be divided into the following four categories: **(i) Machine learning**: These approaches utilize

machine learning models to encode the query information and make tuning decisions. Examples are Deep Neural Network [23], Feedforward Neural Network [24], and Pairwise Deep Neural Network [25]. **(ii) Reinforcement learning**: Tuning the database configurations is natural to address by reinforcement learning methods, such as Deep Q-learning [26], Deep Deterministic Policy Gradient [27], and Double-State Deep Deterministic Policy Gradient [28]. Those methods are adaptive to dynamic workload changes. **(iii) Rule-based**: Based on the domain knowledge of database engines or experiences of human experts, the conversion rules modelling [29] and hierarchy of rules [30] are introduced to determine the resource demands or optimal physical design of the workloads. **(iv) Cost modeling**: These approaches build accurate prediction models to evaluates the expected execution cost of different configurations. Representative works include [31] and [32].

The corresponding tuning tasks consist of two main aspects: **(i) Database design**. Based on the workload changes, the databases need to evolve the physical design, such as indexes, materialized views, partitioning, and storage to achieve the optimal performance. Sometimes the databases also have to redesign the schema according to the workload information. **(ii) Resource provisioning**. In order to support a new workload not yet deployed in a production environment, the database must estimate the needed hardware resources, including CPU, RAM, Disk I/O, buffer pool size and page size, etc.

2) Workload Scheduling: Workload scheduling which aims at determining the execution order of queries, is also a critical and challenge task. It can have a significant impact on query performance and resource utilization. The current techniques about workload scheduling include machine learning-based, such as SmartQueue [37], and reasoning-based, like Qshuffler [11].

E. Section V: Real-World Applications

Workload-aware performance tuning is widely used in cloud databases like Amazon Redshift [38]. Redshift trains prediction models on each cluster so that the decisions are optimal for the workload served by that cluster. To have a better prediction quality and account for drift, the underlying models are refreshed periodically. The models predict the query execution time and memory usage for each query in the workload. With this knowledge, the workload manager (WLM) [39] makes scheduling and resource allocation decisions wisely. For example, WLM assigns the right amount of memory to queries to fully utilize resources and schedules short-running queries before longer queries in terms of the execution time. In addition, when long running queries with large amount of resource usage block incoming short queries, WLM will preempt the long running queries to make room for short queries. Further more, when the workload is heavy and the current cluster resources are fully utilized, WLM will automatically scale queries out to new clusters.

Amazon Redshift also makes decisions for optimal physical data organization by monitoring query patterns. The distribution key advisor uses combinatorial optimization techniques to recommend optimal data distribution to minimize network communication among nodes of a cluster.

F. Section VI: Open Problems

In this section, we will summarize some open problems that must be addressed to ensure the effectiveness of workload-aware tuning:

1) **Robust workload classification and forecasting**: How to realize robust classifiers and predictors for problematic workloads (e.g., the noisy workload patterns).
2) **Tuning with inaccurate workload information**: How to enhance the performance even in spite of the fact that the workload prediction may be inaccurate.
3) **Online update of tuning models**: How to effectively retrain and update the models for new data after deploying them in the real industrial environment (e.g., in the cloud).

III. Tutorial Organization

The tutorial is planned for 3 hours and is divided into the following parts:

Motivation (5'). We motivate the need for workload-aware performance tuning with several applications/scenarios.

Workload classification (30'). We introduce key approaches of workload classification and compare the differences between the various categories.

Workload forecasting (30'). We show how the workload forecasting methods can be used to predict various parameters.

Workload-aware tuning and scheduling (60'). We summarize the key approaches to workload-based tuning and workload scheduling.

Real application and demonstration (30'). We discuss some real applications of workload-aware performance tuning and give a demo to show the pipeline of workload-aware tuning.

Open problems (15'). We conclude with a discussion of open problems and challenges for workload-aware tuning.

Summary (10'). We summarize this tutorial and give our critical thoughts to workload-aware tuning.

IV. Goals of the Tutorial

A. Learning Outcomes

The main learning outcomes of this tutorial includes: (1) Motivation and background of workload-aware tuning. (2) An overview of workload-aware tuning approaches with respect to workload classification, workload forecasting, and workload-based tuning. (3) Comparison of features, strengths, and applications of these three topics of workload-aware tuning. (4) A take-away message for achieving workload-aware autonomous DBMS and a discussion of research challenges and open problems. (5) A real-world hands-on demonstration of workload-aware tuning.

B. Intended Audience

This tutorial is intended for a broad scope of audience ranging from database systems researchers to industry practitioners, with a focus on automatic database tuning. Basic knowledge in database workload and configuration is sufficient

to follow this tutorial. Some background in machine learning and reinforcement learning techniques would be useful.

V. Tutorial Presenters

Zhengtong Yan is a doctoral student at the University of Helsinki. His research topics include multi-model autonomous databases and cross-model query optimization.

Jiaheng Lu is a professor at the University of Helsinki. His main research interests lie in the database systems specifically in the challenge of efficient data processing from real-life, massive data repositories and the Web. He has written four books on Hadoop and NoSQL databases, and more than 100 papers published in SIGMOD, VLDB, TODS, and TKDE, etc.

Naresh Chainani is a senior software development manager at Amazon Web Services (AWS) working on query processing, query performance, distributed systems, and workload management. He is passionate about building high-performance databases that are easy to use and has multiple papers and patents.

Chunbin Lin is a software engineer at Amazon Web Services (AWS) and he is working on AWS Redshift. He completed his Ph.D. in computer science at the University of California, San Diego (UCSD). His research interests are distributed database management and big query analytics. He has more than 30 papers published in SIGMOD, VLDB, VLDB J, and TODS, etc.

References

[1] A. Pavlo, G. Angulo, J. Arulraj, H. Lin, J. Lin, L. Ma, P. Menon, T. Mowry, M. Perron, I. Quah, S. Santurkar, A. Tomasic, S. Toor, D. V. Aken, Z. Wang, Y. Wu, R. Xian, and T. Zhang, "Self-Driving Database Management Systems," in *CIDR*, 2017.

[2] L. Liu and M. T. Özsu, *Encyclopedia of Database Systems*. New York: Springer, 2018.

[3] M. Vogt, A. Stiemer, and H. Schuldt, "Icarus: Towards a Multistore Database System," in *Big Data*. IEEE, 2017, pp. 2490–2499.

[4] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala, "Database Tuning Advisor for Microsoft SQL Server 2005," in *SIGMOD*. ACM, 2005, pp. 930–932.

[5] M. Holze, C. Gaidies, and N. Ritter, "Consistent On-line Classification of DBS Workload Events," in *CIKM*, 2009, pp. 1641–1644.

[6] T. J. Wasserman, P. Martin, D. B. Skillicorn, and H. Rizvi, "Developing A Characterization of Business Intelligence Workloads for Sizing New Database Systems," in *DOLAP*. ACM, 2004, pp. 7–13.

[7] Z. Zewdu, M. K. Denko, and M. Libsie, "Workload Characterization of Autonomic DBMSs Using Statistical and Data Mining Techniques," in *WAINA*. IEEE, 2009, pp. 244–249.

[8] S. Elnaffar and P. Martin, "The Psychic–Skeptic Prediction Framework for Effective Monitoring of DBMS Workloads," *Data & Knowledge Engineering*, vol. 68, no. 4, pp. 393–414, 2009.

[9] M. Ahmad, A. Aboulnaga, S. Babu, and K. Munagala, "Interaction-aware scheduling of report-generation workloads," *The VLDB Journal*, vol. 20, no. 4, pp. 589–615, 2011.

[10] M. Ahmad, A. Aboulnaga, and S. Babu, "Query Interactions in Database Workloads," in *DBTest*, 2009, pp. 1–6.

[11] M. Abdul, A. M. Muhammad, N. Mustapha, S. Muhammad, and N. Ahmad, "Database Workload Management Through CBR and Fuzzy Based Characterization," *Applied Soft Computing*, vol. 22, pp. 605–621, 2014.

[12] B. Raza, Y. J. Kumar, A. K. Malik, A. Anjum, and M. Faheem, "Performance Prediction and Adaptation for Database Management System Workload Using Case-Based Reasoning Approach," *Information Systems*, vol. 76, pp. 46–58, 2018.

[13] L. Ma, D. V. Aken, A. Hefny, G. Mezerhane, A. Pavlo, and G. J. Gordon, "Query-based Workload Forecasting for Self-driving Database Management Systems," in *SIGMOD*. ACM, 2018, pp. 631–645.

[14] G. C. Durand, M. Pinnecke, R. Piriyev, M. Mohsen, D. Broneske, G. Saake, M. S. Sekeran, F. Rodriguez, and L. Balami, "GridFormation: Towards Self-driven Online Data Partitioning Using Reinforcement Learning," in *aiDM*, 2018, pp. 1–7.

[15] S. Jain, B. Howe, J. Yan, and T. Cruanes, "Query2Vec: An Evaluation of NLP Techniques for Generalized Workload Analytics," *arXiv:1801.05613*, 2018.

[16] R. Taft, N. El-Sayed, M. Serafini, Y. Lu, A. Aboulnaga, M. Stonebraker, R. Mayerhofer, and F. Andrade, "P-store: An Elastic Database System With Predictive Provisioning," in *SIGMOD*. ACM, 2018, pp. 205–219.

[17] A. S. Higginson, M. Dediu, O. Arsene, N. W. Paton, and S. M. Embury, "Database Workload Capacity Planning using Time Series Analysis and Machine Learning," in *SIGMOD*, 2020, pp. 769–783.

[18] M. Holze, A. Haschimi, and N. Ritter, "Towards Workload-aware Self-management: Predicting Significant Workload Shifts," in *ICDE Workshops*. IEEE, 2010, pp. 111–116.

[19] A. Pavlo, E. P. Jones, and S. Zdonik, "On Predictive Modeling for Optimizing Transaction Execution in Parallel OLTP Systems," *arXiv:1110.6647*, 2011.

[20] M. Holze and N. Ritter, "Autonomic Databases: Detection of Workload Shifts with n-Gram-Models," in *ADBIS*. Springer, 2008, pp. 127–142.

[21] C. Sapia, "PROMISE: Predicting Query Behavior to Enable Predictive Caching Strategies for OLAP Systems," in *DaWaK*. Springer, 2000, pp. 224–233.

[22] N. Du, X. Ye, and J. Wang, "Towards Workflow-driven Database System Workload Modeling," in *DBTest*, 2009, pp. 1–6.

[23] C. Tang, Z. Dong, M. Wang, Z. Wang, and H. Chen, "Learned Indexes for Dynamic Workloads," *arXiv:1902.00655*, 2019.

[24] B. Mozafari, C. Curino, A. Jindal, and S. Madden, "Performance and Resource Modeling in Highly-concurrent OLTP Workloads," in *SIGMOD*. ACM, 2013, pp. 301–312.

[25] J. Tan, T. Zhang, F. Li, J. Chen, Q. Zheng, P. Zhang, H. Qiao, Y. Shi, W. Cao, and R. Zhang, "iBTune: Individualized Buffer Tuning for Large-scale Cloud Databases," in *VLDB*, 2019, pp. 1221–1234.

[26] B. Hilprecht, C. Binnig, and U. Röhm, "Learning A Partitioning Advisor for Cloud Databases," in *SIGMOD*. ACM, 2020, pp. 143–157.

[27] J. Zhang, Y. Liu, K. Zhou, G. Li, Z. Xiao, B. Cheng, J. Xing, Y. Wang, T. Cheng, L. Liu *et al.*, "An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning," in *SIGMOD*. ACM, 2019, pp. 415–432.

[28] G. Li, X. Zhou, S. Li, and B. Gao, "QTune: A Query-aware Database Tuning System with Deep Reinforcement Learning," in *VLDB*, 2019, pp. 2118–2130.

[29] C. de Lima and R. dos Santos Mello, "A Workload-Driven Logical Design Approach for NoSQL Document Databases," in *iiWAS*, 2015, pp. 1–10.

[30] S. Das, F. Li, V. R. Narasayya, and A. C. König, "Automated Demand-driven Resource Scaling in Relational Database-as-a-Service," in *SIGMOD*. ACM, 2016, pp. 1923–1934.

[31] S. Idreos, N. Dayan, W. Qin, M. Akmanalp, S. Hilgard, A. Ross, J. Lennon, V. Jain, H. Gupta, D. Li *et al.*, "Design Continuums and the Path Toward Self-Designing Key-Value Stores That Know and Learn," in *CIDR*, 2019.

[32] I. Alagiannis, S. Idreos, and A. Ailamaki, "H2O: A Hands-free Adaptive Store," in *SIGMOD*. ACM, 2014, pp. 1103–1114.

[33] B. Glasbergen, M. Abebe, and K. Daudjee, "Tutorial: Adaptive Replication and Partitioning in Data Systems," in *Middleware '18 Tutorials*. ACM, 2018, pp. 1–5.

[34] J. Lu, Y. Chen, H. Herodotou, and S. Babu, "Speedup Your Analytics: Automatic Parameter Tuning for Databases and Big Data Systems," in *VLDB*, 2019, pp. 1970–1973.

[35] J. Arulraj, A. Pavlo, and P. Menon, "Bridging the Archipelago Between Row-stores and Column-stores for Hybrid Workloads," in *SIGMOD*. ACM, 2016, pp. 583–598.

[36] M. Holze and N. Ritter, "Towards Workload Shift Detection and Prediction for Autonomic Databases," in *PIKM*, 2007, pp. 109–116.

[37] C. Zhang, R. Marcus, A. Kleiman, and O. Papaemmanouil, "Buffer Pool Aware Query Scheduling via Deep Reinforcement Learning," *arXiv:2007.10568*, 2020.

[38] Amazon Redshift, https://docs.aws.amazon.com/redshift.

[39] Amazon Redshift WLM, https://docs.aws.amazon.com/redshift/latest/dg/automatic-wlm.html.