# Autotuning Systems: Techniques, Challenges, and Opportunities

Brian Kroth
Microsoft
Gray Systems Lab
Madison, WI, USA
bpkroth@microsoft.com

Sergiy Matusevych
Microsoft
Gray Systems Lab
Redmond, WA, USA
sergiym@microsoft.com

Yiwen Zhu
Microsoft
Gray Systems Lab
Mountain View, CA, USA
yiwzh@microsoft.com

## Abstract

The rapid growth of cloud computing and systems has introduced significant complexity in managing and optimizing configurations to meet diverse workload demands across a wide array of hardware. Autotuning systems, leveraging advancements in machine learning and optimization, offer an effective solution to these challenges. By automating configuration tuning, these systems can dynamically adapt to workload changes, optimize performance in real time, and reduce the burden on system administrators. This tutorial provides both theoretical foundations and practical demonstrations of systems autotuning software, with a focus on offline and online optimization methodologies. We present a comprehensive review of state-of-the-art autotuning systems and discuss how they address key challenges, such as handling large configuration spaces, mitigating noise in real-world environments, and ensuring safe and efficient exploration during tuning. To conclude, we offer a hands-on session where participants can experiment with an open-source system and gain experience with real-world tuning scenarios.

## CCS Concepts

• **Computer systems organization → Cloud computing**; • **Computing methodologies → Machine learning**.

## Keywords

Bayesian Optimization, Autotuning, Performance Tuning, Noise Reduction, Performance Variability

**ACM Reference Format:**
Brian Kroth, Sergiy Matusevych, and Yiwen Zhu. 2025. Autotuning Systems: Techniques, Challenges, and Opportunities. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3722212.3725638

## 1 Introduction

The rapid growth of cloud platforms has led to the collection of vast amounts of workload traces and system telemetry from millions of users and applications [20, 42]. This abundance of data, coupled with advances in instrumentation, monitoring, and machine learning (ML) techniques, provides a fertile foundation for the development of autonomous services. Automation in this context facilitates critical tasks such as resource provisioning [66], scheduling [10], data systems query optimization [28, 33], execution, and service tuning [9, 40, 68], achieving significant gains in performance and cost efficiency. Such automation substantially reduces the workload of systems administrators (e.g., DBAs) and has given rise to the concept of autotuning systems—which may also include self-tuning, self-adaptive, or self-managing systems—which have garnered significant attention in modern systems, especially data systems' management. These systems have found widespread applications in real-world database environments [55, 66], driving the evolution of autonomous data services [68]. In this tutorial, we explore recent advancements in configuration tuning, a critical area of research that has garnered significant attention due to its potential to further enhance the performance, cost efficiency, reliability, and adaptability of systems with minimum overhead. We provide an overview of the area, deconstructed under the lens of **deployment strategy** (e.g., offline and online tuning), and further discuss some of the **tuning challenges** of each (e.g., efficiency, robustness, safety, predictability, etc.), as well as some of the **mitigation strategies** for overcoming them (e.g., workload identification, transfer learning, space reduction, range biasing, guardrails, etc.).

Armed with a theoretical background, we then present attendees with a hands-on lab session using MLOS [12, 25], an open-source framework for full-stack autotuning experimentation, to provide a practical demonstration of some concepts discussed in the tutorial.

## 2 Tutorial Information

### 2.1 Target Audience

This tutorial targets systems and database practitioners, researchers, and industry professionals interested in machine learning for systems. It introduces auto-tuning challenges, techniques, and opportunities, requiring no prior knowledge of databases or ML.

### 2.2 Tutorial Breakdown

The tutorial spans 1.5 hours, extendable to 3 hours, and consists of two parts:

**Section 1: Foundations of Autotuning Systems (75 mins).**

(1) **Deployment Scenarios** We begin by categorizing auto-tuning by deployment context to orient attendees.
    **Offline Optimization.** (25 mins)
    - **Bayesian Optimization:** Using trial history to guide tuning via Gaussian Processes and related methods.
    - **Applications and Challenges:** Survey of ML approaches and systems, covering issues like dimensionality reduction, noise, and workload diversity.

    **Online Optimization.** (25 mins)

- **Adapting in Real-Time:** Systems that adjust continuously to evolving workloads.
- **Theory and Methods:** Overview of online learning and reinforcement learning.
- **Applications:** Production challenges and state-of-the-art online tuning systems.

(2) **Common Challenges and Strategies** (25 mins) We highlight recurring issues (e.g., efficiency, safety) and strategies including test acceleration, space pruning, noise handling, and transfer learning.

- **Focus: Workload Identification** A key technique enabling better generalization across systems and workloads.
  - **Motivation:** Transferability and drift adaptation using similarity metrics.
  - **Theory:** ML-based embeddings for compact workload representation.
  - **Applications:** Real-world use and challenges in scaling workload identification to cloud systems.

**Section 2: Hands-On with MLOS (15 mins).**

(1) **Intro to MLOS** Overview of the MLOS framework and its integration of discussed techniques for experimentation and optimization.

(2) **(Optional) Lab Demo** (30 mins)
Hands-on session with:

- Full-stack autotuning using MLOS.
- Experiments across VMs, OS kernels, and DBMS.
- Result visualization and framework extensibility.

## 3 Tutorial Outline

As outlined above, we begin with a high level overview of the end goal, autotuning systems, and describe the two main deployment methods: offline and online tuning.

### 3.1 Introduction

***Motivation: Manual Tuning.*** *"You can't improve what you don't measure."* — This well-known adage, often attributed to Peter Drucker, captures the essence of the optimization process. A system may perform adequately under a given workload and configuration, but how can we ascertain whether it is truly *optimal*? The answer lies in systematic benchmarking and experimentation. To assess optimality, we must evaluate the system under diverse conditions and measure performance metrics that align with our specific goals—such as throughput, tail latency, robustness to workload variation, and operational cost. Optimization becomes even more complex when multiple objectives must be balanced, necessitating techniques such as Pareto frontiers or multi-objective optimization [53].

To illustrate the challenges of manual tuning, we developed an interactive online "game" in which Spark experts adjust commonly tuned configuration parameters, including `maxPartitionBytes`, `autoBroadcastJoinThreshold`, `shuffle.partitions`, and others. Users can observe the hypothetical execution time for each configuration and make informed decisions about subsequent adjustments. Typically, one parameter is modified at a time, enabling a straightforward rollback if performance degradation is observed.

The manual tuning approach underscores the necessity for automation. The exponential growth in the number of possible configurations with each additional parameter makes manual testing infeasible. Moreover, as the number of required experiments increases, so does the time (cost) required to conduct them. To address these challenges, two high-level strategies are often applied to improve a major challenge—*sample efficiency*:

- Accelerating each trial, possibly by adopting surrogate metrics or a series of progressively complex tests (e.g., by adjusting duration or scale factor in a DB benchmark), and
- Reducing the number of trials needed to find a good config.

These strategies are complementary and can be synergistically applied. This tutorial will discuss both strategies, focusing primarily on (1) offline auto-tuning and (2) employing machine learning for sample-efficient optimization.

***Introduction to Autotuning Systems.*** We classify system tuning into two main approaches: (a) *online*, where production system parameters are adjusted at runtime under safety constraints (e.g., [29, 35]); and (b) *offline*, where more flexible tuning is conducted in isolated environments (e.g., [52, 57, 59]). Both approaches face distinct challenges and employ various mitigation strategies.

This tutorial highlights *workload identification*, a technique that enables insight transfer across systems by characterizing workloads [37]. In large-scale cloud deployments, recognizing similar instance behavior supports efficient configuration transfer and cost optimization. Due to workload diversity, accurate classification and similarity assessment are essential.

We explore offline tuning in Section 3.2, online tuning in Section 3.3, shared challenges in Section 3.4, and workload identification in Section 3.4.2. Importantly, these methods can be combined: offline optimizers can be seeded with results from similar systems via workload identification, and online tuning can refine configurations using offline data, creating a feedback loop.

### 3.2 Offline Optimization

*3.2.1* ***Theory: Introduction to Bayesian Optimization.*** From the optimizer perspective, it is convenient to treat the system under optimization as a black box: as long as we can specify how to set the parameters and measure the performance of the system, we can use a variety of strategies to pick the next configuration to test, and, eventually, choose the best-performing configuration.

Grid search and random search are two basic strategies for offline optimization. Both have an advantage of being easy to parallelize: we can run multiple trials in parallel and pick the best configuration at the end. The disadvantage is that they are not *sample-efficient*, and the information gained in one trial or experiment is not used in the subsequent ones. As a result, the system can spend a lot of time testing configurations that are not promising or outright bad.

This is where ML-based optimization comes into play: we can build a model that uses the results of previous trials to suggest the next configuration to test [64]. A more sophisticated model can also use the data from other systems as well as the telemetry data from the system under optimization, and incorporate the user-defined and learned constraints to further narrow down the search space [43, 48, 55].

We will start with the most basic and widely used ML model for optimization: Bayesian Optimization (BO), that has been used by many tuning systems [4, 7, 25, 27, 48, 55, 58]. In this approach, the Gaussian Process (GP) models the objective function (both in mean and variance) for each configuration. An acquisition function, such as Expected Improvement (EI) or Upper Confidence Bound (UCB) can be used to select the optimal candidate for the next round [45].

### 3.2.2 *Applications, Techniques, and Challenges.*

Even though for many practical applications, optimizing a handful of parameters using simple BO on synthetic benchmarks already yields good results [35], every system optimization practitioner very soon faces a number of challenges that make the solution hard to scale.

**Curse of Dimensionality.** Regardless of the optimization method, the number of possible configurations in the real-world scenario grows exponentially with the number of parameters. Hence, it is highly desirable to restrict the search space before and during the optimization process. Below we will survey the main methods for dimensionality reduction in the context of system optimization.

- **Dimensionality Reduction.** Many systems reduce configuration space complexity by encoding configurations into lower-dimensional representations. **Db2une** prioritizes high-impact regions using telemetry [4], while **Proto-X** and **LlamaTune** use latent-space embeddings and projections to enhance sample efficiency [23, 57]. **Rover** leverages SHAP and expert feedback [44], and **OtterTune** applies Bayesian optimization guided by past workloads [48]. **DBTune** selects knobs via Lasso [58], and **OpAdviser** transfers promising regions across tasks. Recent work explores using LLMs to propose knobs and ranges from documentation and source code [14, 27, 47].
- **Inter-Parameter Dependencies.** Encoding known parameter relationships (e.g., constraints or feature toggles) can shrink the search space. Techniques include constrained Bayesian optimization [18], clustering via correlation [1], and structure learning with graphical models [51]. Though data-hungry, such models are promising in transfer learning scenarios. **DBTune** may aid in inferring dependencies [61].
- **Feature Priors.** Another strategy is constraining each parameter's value range using expert rules, LLMs [27], or learned priors. Methods include scaling, value quantization, and prior distributions (e.g., Beta, Normal). **Proto-X** uses histogram-based quantization [57].

**Execution Speed-up.** The systems optimization process is often limited by the execution time of the trials. It makes sense, therefore, to reduce the time it takes to run each trial, run multiple trials in parallel, and use the surrogate models to predict the performance or feasibility of the configurations that have not been tested yet.

- **Parallelization.** It is trivial to speed up the optimization process with grid search or random search by executing multiple trials in parallel, since each can be executed independently. With the ML-based optimization, the situation is more complicated: the model needs to be updated after each trial, and the next configuration to test is selected based on the updated model. The model, therefore, must be able to generate multiple suggestions at once, and make sure the configurations planned for parallel execution are not too close to each other and together make a good coverage of the search space [15].
- **Early Stopping.** For tests optimizing task runtime, if a given trial takes too long to run, it makes sense to stop it early and inform the optimizer that the proposed configuration is clearly suboptimal. Here we can use information from earlier trials to set the cut-off time. One can also use the telemetry collected during the trial to make such a decision.
- **Surrogate Models.** Instead of running an expensive benchmark, one can use a surrogate model to predict the performance of the configuration. This scheme is widely used in the context of Reinforcement Learning, such as **QTune** [29] and **CDBTune** [56]. **OtterTune** [48, 55] uses a machine learning model to predict the feasibility of the configuration suggested by the optimizer and discard the configurations that are likely to perform poorly. **OpAdviser** introduces a meta-learner to predict the most suitable optimizer based on task characteristics, further improving the convergence of the tuning [63]. **GPTuneBand** [65] and **MFIX** [8] incorporates multi-fidelity testing into surrogate modeling to enhance the efficiency and accuracy of performance predictions by leveraging data of varying fidelities (such as from similar tasks).

**Noise Mitigation.** Despite decades of research into isolation by the systems community, real-world systems remain inherently noisy: performance can vary between runs due to factors such as hardware variability, resource contention, and interference in shared cloud environments [30]. Noise like this significantly impacts the optimization process, as optimizers can be misled by unreliable measurements and select suboptimal configurations [16]. The state of the art has typically been statistical measures, e.g., **ResTune** [60] performs each experiment three times and averages the results to reduce observation variances. **Duet Benchmarking** [5] instead proposed side by side measures to evaluate relative performance differences for code changes, which can be akin to config changes. **OtterTune** [48, 55] employs Factor Analysis to reduce dimensionality before clustering workloads for configuration selection, thereby mitigating the impact of noise in the observational data. **[30]** introduces Gaussian white noise into the Gaussian Process (GP) model to enhance prediction accuracy in the presence of noise. In this section, we discuss several other potential strategies to mitigate noise, improve sample efficiency, and enhance the robustness of optimizers, including **TUNA** [17].

- **Micro-Benchmarks.** Instead of evaluating the full workload, micro-benchmarks focus on isolated components of the system to provide a controlled, low-variance measurement of specific performance aspects. These benchmarks are faster to execute and allow optimizers to test configurations in an environment with minimized noise, offering more reliable feedback for fine-tuning individual parameters. However, short runs or microbenchmarks may not be entirely predictive of the full system behavior, leading to the need for *multi-fidelity* optimization.
- **Sideband Signals.** Additional signals or metrics, known as sideband signals, can provide valuable context during the optimization process. For example, delayed signals (e.g., CPU usage, memory pressure, or system logs) can be analyzed to

re-evaluate noisy benchmarks post hoc. Fluctuations in microbenchmarks can be used to infer resource interference [26]. These sideband signals can also be incorporated into surrogate models as auxiliary data to help predict performance more accurately, smoothing out noise and improving the optimizer's decision-making process. For instance, **[1]** proposes to leverage multi-task Bayesian Optimization to optimize multiple performance objectives simultaneously, such as maximizing IOPS (input/output operations per second) while minimizing write amplification and latency.

- **Outlier Detection and Smoothing.** Detecting and filtering outliers from noisy observations helps prevent the optimizer from being misled. Techniques such as moving averages, weighted smoothing, or statistical outlier detection methods (e.g., z-score analysis) can be applied to stabilize performance metrics, ensuring that optimization decisions are based on reliable trends rather than isolated noisy measurements [16].

By combining these strategies, optimization systems can effectively mitigate noise, ensuring sample efficiency and more consistent and reliable performance improvements even in dynamic and noisy real-world environments.

**Non-Representative Benchmarks.** Finally, at the end of this section, we have to point out to the elephant in the room: synthetic benchmarks used for the optimization process are often not representative of the real-world workloads [13, 49]. The reasons for that are numerous: the benchmarks are often designed to stress-test the system, they are not updated to reflect the changes in the system or the workload or the new usage scenarios, and so on. In the sections that follow, we will show how online optimization and workload identification help to mitigate this issue by periodically selecting configs from *similar* workloads tuned offline and then fine tuning online, and further discuss the ways to make the benchmarks more representative of the real-world workloads.

## 3.3 Online Optimization

***Motivation: Adapting in Real-Time.*** While offline tuning addresses the configuration tuning for specific workloads in a more optimal way, in production environments, online tuning also becomes attractive in many scenarios due to: (1) Dynamic workloads: The data size, query patterns, or resource demands may change significantly even for the same requests structure. There may be rarely the need to execute identical queries multiple times, and under varying conditions, as the optimal configuration may differ; and (2) Environmental changes: Factors such as software updates, hardware upgrades, or shifts in network conditions can alter system performance, necessitating real-time adjustments.

Though recommendations obtained from offline tuning can be periodically reapplied, with online tuning, the configuration can be further adaptive to account for changes during the runtime through a tighter loop iterative process. However, there are still several challenges:

- **Workload shifting.** Rapid changes in workload characteristics can lead to suboptimal configuration selections unless the system adapts promptly.
- **Performance regression.** Dynamic adjustments can sometimes degrade performance in unpredictable ways, requiring mechanisms to ensure stability during configuration transitions for Service Level Agreement (SLA) compliance.
- **Explainability.** Studies show that both customers and support engineers often struggle with online approaches due to their potential for unexplainable or hard-to-debug behavior.
- **Noisy data in production environments.** Various factors, such as resource contention, hardware heterogeneity, and unpredictable external influences, can introduce noise into system data, complicating optimization convergence efforts.

In this section, we discuss techniques designed to address these challenges effectively.

***Theory: Online Tuning Algorithms.*** Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment, aiming to maximize cumulative rewards [22]. In the context of online tuning, RL involves three key components: (1) **State:** Represents the environment's status, which may include workload characteristics, resource usage, and current configurations; (2) **Action:** Defines the set of adjustments the agent can make to configuration parameters, such as increasing, decreasing, or setting a specific value; and (3) **Reward:** Provides scalar feedback to evaluate the quality of an action. Rewards are often derived from performance metrics such as throughput, latency, or cost efficiency.

Classic Q-learning algorithms can be used to train the model where a Q-table is learned, representing the value of a state taking a specific action [56]. Actor-Critic methods are commonly used in RL-based tuning systems. These combine policy optimization (actor) and value function estimation (critic) to iteratively improve decision-making [29, 57]. Other methods, such as Monte Carlo Tree Search (MCTS), explore configuration spaces efficiently by balancing exploration and exploitation [52].

Genetic Algorithms (GA) are also widely used in online tuning frameworks due to their ability to handle high-dimensional and complex configuration spaces and easy implementation. GAs operate by: (1) Evaluating configurations using a fitness function that measures their quality relative to an optimization goal; (2) Selecting high-performing configurations as parents; and (3) Generating new configurations (offspring) by combining features of parent configurations (crossover) or adding small random changes (mutation). The iterative process explores the search space efficiently, avoiding local optima and refining solutions until a stopping criterion is met.

Several other approaches extend beyond Reinforcement Learning (RL) and Genetic Algorithms (GA) to address specific challenges in online tuning. Greedy search, for example, simplifies the optimization process by iteratively selecting the best immediate configuration adjustments without exploring a broader set of possibilities [2]. Hybrid bandits combine contextual bandit algorithms with perturbation strategies to dynamically refine configurations, improving sample efficiency and adaptability [46]. As in offline mode, multi-objective optimization is also possible [32]. These approaches complement RL and GA by offering alternative strategies suited to specific use cases and constraints in system tuning. [67] introduced a divide-and-conquer approach to recursively explore the parameter space by discretizing the parameter ranges.

***Applications: Online Tuning.*** Reinforcement Learning (RL) has been successfully applied in various online tuning systems. For instance, **QTune** leverages deep RL to dynamically optimize database query performance by adjusting configurations in real-time [29]. Similarly, **Proto-X** employs RL to coordinate the tuning of multiple components in database systems, using synthesized proto-actions to achieve optimal configurations [57]. **HUNTER** [6] introduces a hybrid architecture by adjusting configurations dynamically based on real-time feedback combining Genetic Algorithms (GA) and Deep Reinforcement Learning (DRL). Genetic Algorithms (GA) are also widely used for their effectiveness in handling complex, high-dimensional configuration spaces. **DAC** utilizes GA to adaptively tune high-dimensional configurations for distributed systems [54], while **RFHOC** combines GA with a random forest-based model to optimize Hadoop configurations [3]. **AutoSteer** uses greedy search to optimize SQL database queries by iteratively improving configurations [2], and **LITE** employs adaptive modeling techniques for tuning Spark configurations based on performance feedback [31].

**Workload Shifting.** While most online tuning algorithms demonstrate good adaptability to workload shifts, several advanced techniques have been developed to further enhance the generalization of trained models in the face of workload changes:

- **OnlineTune** [62] embeds environmental factors as contextual features in its optimization model, enabling dynamic adaptation to changing workloads and environments. By leveraging prior workload context, such as data size and query plan variations, OnlineTune efficiently handles workload shifts using contextual Bayesian Optimization.
- **OPPerTune** [46] introduces the AutoScoper component to inject additional contextual information, such as job type and requests per second (RPS), into its Hybrid Bandit algorithm. AutoScoper uses a decision tree model to select the appropriate tuning instance for a given context, ensuring that the learned tuning strategies align with the operational realities of production for each specific scope.

**Performance Regression.** To ensure performance guarantees during the tuning process, several methods have been proposed:

- **OnlineTune** [62] reduces the optimization process over the entire configuration space into a sequence of subspace optimizations centered around the best configuration estimated so far, gradually converging toward the optimal solution. Additionally, OnlineTune assesses the safety of candidate configurations using the model's lower-bound performance estimates.
- **LOCAT**: Employs Safe Bayesian Optimization to explore Spark SQL configurations while minimizing the risk of performance regressions, ensuring stability during the tuning process.
- **AutoSteer** [2] uses a greedy search strategy to incrementally improve configurations, balancing exploration and exploitation to avoid significant regressions.
- **HUNTER** [6] implements a cloned Cloud Database (CDB) strategy, where cloned database instances are used to verify configuration impacts. This approach isolates tuning experiments, ensuring that suboptimal configurations do not affect the production system. As such, this can be viewed as a hybrid online-offline approach.
- **OPPerTune** [46] combines contextual bandits with a probabilistic approach to perturb configurations around the "current best" candidate, limiting exploration to safe regions of the configuration space and minimizing performance risks at the expensive of potentially achieving a global optimum.
- **[30]** introduces safe exploration by defining a safe region using Gaussian Process predictions, selecting configurations that satisfy constraints, such as runtime thresholds.

**LLM-Related Advancements.** Recent advancements in Large Language Models (LLMs) have led to several novel applications in database tuning, leveraging the power of LLMs to enhance efficiency and performance:

- **DBBert** [47] extracts tuning hints from text documents to identify optimal tuning knobs using large, pre-trained language models, specifically the BERT model.
- $\lambda$**-Tune** [19] explores the feasibility of replacing the entire tuning process with LLM calls. In this approach, all configurations are generated by LLMs as prompts, and an additional configuration selector is developed to identify the optimal candidate.
- **LATuner** [14] utilizes LLMs to identify critical knobs and warm-start the tuning process. Two surrogate models are employed: one based on a Gaussian Process and another guided by LLM-generated prompts to predict performance. Additionally, LATuner uses LLMs to sample effectively in high-value spaces based on sampling prompts.
- **GPTuner** [27] leverages LLMs to aggregate domain knowledge from diverse sources and transforms this knowledge into a structured format, focusing on attributes such as suggested values, minimum and maximum values, and special values for each knob. During the tuning process, GPTuner also employs an LLM-based knob selection mechanism to reduce tuning dimensionality and integrates with Bayesian Optimization to explore the configuration space effectively.

## 3.4 Common Challenges and Strategies

*3.4.1 Challenges.* As outlined above, both offline and online tuning approaches must address a number of challenges which we summarize here, including sample efficiency during exploration and predictability or robustness during exploitation.

*3.4.2 Strategies.* In this section we will summarize some of the commonalities of the strategies used to address those challenges presented in the previous sections such as space reduction, test acceleration, surrogate models, noisy signal handling, etc. (§ 3.2.2).

We will then highlight one a specific technique in detail that can be used to improve sample efficiency through transfer learning of similar workloads and predictability through drift detection.

***Example: Workload Identification***

**Motivation: Comparing Systems.** Assume that by some optimization process we have found the best configuration for a certain system and workload. Now, as it typically happens in the cloud scenario, we want to find other systems and workloads that can benefit from the same setup. Doing so could help reduce the cost of optimization and improve overall service experience. However, we can't use system metrics for this task, as the goal of autotuning is often to reduce system resources, which would lead to an unstable signal. Thus, we need to use application-level workload request info. Yet, even if we have access to the query logs and telemetry data from all systems (which is not always possible for privacy reasons), it is not immediately clear how to measure the similarity between them. This is where Machine Learning can help.

**Theory: Embeddings.** One approach is to represent each workload as a vector, or point in a high-dimensional space [31]. Such vector representation of a complex input is called an *embedding* and is widely used in Machine Learning, e.g., in Natural Language Processing [38], Computer Vision [39], and Recommender Systems [24].

Once we have such vector representations, it is trivial to measure the similarity between the workloads, e.g., using the Euclidean distance or cosine similarity [4, 48, 55]. We can also use the dimensionality reduction techniques to project them into a lower-dimensional space, e.g., using UMAP [34], t-SNE [50], PCA [6], for visualization, or use directly for clustering, classification, and other ML tasks [29].

**Applications and Challenges.** Capturing the embedding of a workload is not trivial for a number of reasons:

- **Query Representation.** Queries can be represented in either raw or structured forms. Raw queries often appear as the text of SQL statements, such as those found in MySQL slow query logs [4, 29]. Structured representations include Abstract Syntax Trees (ASTs), Logical Execution Plans, or Physical Execution Plans [29, 31]. **DB2une** introduces QBERT, a query featurization method that uses a transformer-based model trained on a masked-query plan template prediction task. It leverages query execution plan structures and associated statistics for effective representation. **QTune** develops Query2Vector, which extracts query plans and estimated costs from the database engine and converts this information into a meaningful vector representation. **ResTune** applies TF-IDF (Term Frequency-Inverse Document Frequency) to transform raw SQL query text into vectors, using reserved SQL keywords (e.g., SELECT, UPDATE, JOIN) to identify query patterns [60].
- **Runtime Statistics.** Runtime information, such as execution Directed Acyclic Graphs (DAGs) or query statistics, can provide a detailed characterization of queries. **LITE** [31] combines runtime information from stage-level code and DAGs provided by the scheduler. It creates token-level embeddings for code and node-level embeddings for DAGs. Using a combination of Convolutional Neural Networks (CNNs) and Graph Convolutional Networks (GCNs), LITE generates joint embeddings based on the proposed NECS framework. **OtterTune** [48, 55] featurizes workloads using runtime characteristics such as query execution metrics, resource utilization (e.g., CPU, memory), and lock/contention metrics, providing a holistic view of workload behavior. **[30, 41]** leverage SparkEventLogs to construct features to measure workload similarity, summarizing stage-level (Spark actions/transformations) and task-level (CPU, memory usage) information. In **[30]**, a total of 75 meta-features are used to characterize tasks, which are then leveraged for meta-learning to accelerate optimization across similar tasks. To measure task similarity, instead of capturing the characteristics of the query, **OpAdviser** uses performance model rankings from historical tasks to measure similarity to the target task [63].

Workload identification is also widely used in applications such as query optimization where query behavior modeling is needed. A detailed review of such work can be found at [11].

## 3.5 MLOS

This session includes a hands-on overview and demonstration of MLOS [25], a framework for autotuning and experimentation. We will show how to use MLOS [25] for tuning Linux kernel parameters for Redis, and a DBMS like MySQL or Postgres in order to improve VM and application configurations in the cloud to enhance performance and efficiency, highlighting the advantages of automated benchmarking and optimization and a flexible framework to also allow further research into the space. Azure cloud resources will be provided for attendees to use during the demo. MLOS can be used in a browser via Github CodeSpaces [21] or locally using a `devcontainer` [36] with all the requirements already available, so requirements are minimal.

## 4 Biography

**Brian Kroth** is a Principal Research Software Development Engineer at Microsoft's Gray Systems Lab (GSL), focusing on systems, databases, and machine learning for systems. He earned both his Bachelor's and Master's degrees in Mathematics and Computer Science from the University of Wisconsin-Madison and has been in the IT industry for 25 years. At Microsoft, Brian has worked on resource management and auto-tuning for various parts of the Azure Data systems, optimized software for storage, virtualization, and autoscaling performance, and is one of the primary contributors to the development of MLOS, an infrastructure for automated performance engineering.

**Sergiy Matusevych** is a Principal Data Scientist at Microsoft Grey Systems Lab (GSL), where he applies Machine Learning for Workload Identification and System Optimization. Sergiy has developed the Inference Engine for Teams, worked on Deep Noise Suppression models, ML.NET framework, and served as a PMC Chair of the Apache REEF project. Prior to joining Microsoft, Sergiy was a Research Data Engineer at Yahoo! Research and tried his hand at several ML startups, all of which have failed spectacularly.

**Yiwen Zhu** is a Principal Scientist at Microsoft's Gray Systems Lab (GSL). Her research interests center on the vision of autonomous cloud systems, utilizing machine learning, statistical inference, and operation research techniques. Additionally, Yiwen leads the development of an internal large language model (LLM) application aimed at improving workflows for software engineers.

## Acknowledgments

We would like to acknowledge the many people who have been involved in some aspects of these efforts, including, but not limited to Aditya Lakra, Andreas Mueller, Arshdeep Sekhon, Ashit Gosalia, Carlo Curino, Dario Bernal, Dhruv Relwani, Estera Kot, Eu Jing Chua, Johannes Freischuetz, Karuna Sagar Krishna, Kelly Kostopoulou, Konstantinos Kanellis, Long Tian, Mo Liu, Rahul Challapalli, Rathijit Sen, Roger Barga, Slava Oks, Adam Smiechowski, Greg Lapinski, Rui Fang, Shaily Fozdar, Shivaram Venkataraman, Subru Krishnan, Jordan Henkel, Sule Kahraman, Tengfei Huang, Weihan Tang, Xin He, Yaseen Shady, and many others.

## References

[1] Sami Alabed and Eiko Yoneki. 2021. High-Dimensional Bayesian Optimization with Multi-Task Learning for RocksDB. In *Proceedings of the 1st Workshop on Machine Learning and Systems* (Online, United Kingdom) *(EuroMLSys '21)*. Association for Computing Machinery, New York, NY, USA, 111–119. https://doi.org/10.1145/3437984.3458841

[2] Christoph Anneser, Nesime Tatbul, David Cohen, Zhenggang Xu, Prithviraj Pandian, Nikolay Laptev, and Ryan Marcus. 2023. AutoSteer: Learned Query Optimization for Any SQL Database. *Proc. VLDB Endow.* 16, 12 (Aug. 2023), 3515–3527. https://doi.org/10.14778/3611540.3611544

[3] Zhendong Bei, Zhibin Yu, Huiling Zhang, Wen Xiong, Chengzhong Xu, Lieven Eeckhout, and Shengzhong Feng. 2016. RFHOC: A Random-Forest Approach to Auto-Tuning Hadoop's Configuration. *IEEE Transactions on Parallel and Distributed Systems* 27, 5 (2016), 1470–1483. https://doi.org/10.1109/TPDS.2015.2449299

[4] Alexander Bianchi, Andrew Chai, Vincent Corvinelli, Parke Godfrey, Jarek Szlichta, and Calisto Zuzarte. 2024. Db2une: Tuning Under Pressure via Deep Learning. *Proc. VLDB Endow.* 17, 12 (Nov. 2024), 3855–3868. https://doi.org/10.14778/3685800.3685811

[5] Lubomír Bulej, Vojtěch Horký, Petr Tuma, François Farquet, and Aleksandar Prokopec. 2020. Duet Benchmarking: Improving Measurement Accuracy in the Cloud. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE '20)*. ACM, 100–107. https://doi.org/10.1145/3358960.3379132

[6] Baoqing Cai, Yu Liu, Ce Zhang, Guangyu Zhang, Ke Zhou, Li Liu, Chunhua Li, Bin Cheng, Jie Yang, and Jiashu Xing. 2022. HUNTER: An Online Cloud Database Hybrid Tuning System for Personalized Requirements. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 646–659. https://doi.org/10.1145/3514221.3517882

[7] Stefano Cereda, Stefano Valladares, Paolo Cremonesi, and Stefano Doni. 2021. CGPTuner: a contextual gaussian process bandit approach for the automatic tuning of IT configurations under varying workload conditions. *Proc. VLDB Endow.* 14, 8 (April 2021), 1401–1413. https://doi.org/10.14778/3457390.3457404

[8] Zhuo Chang, Xinyi Zhang, Yang Li, Xupeng Miao, Yanzhao Qin, and Bin Cui. 2024. MFIX: An Efficient and Reliable Index Advisor via Multi-Fidelity Bayesian Optimization. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 4343–4356. https://doi.org/10.1109/ICDE60146.2024.00331

[9] 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.

[10] Andrew Chung, Carlo Curino, Subru Krishnan, Konstantinos Karanasos, Panagiotis Garefalakis, and Gregory R Ganger. 2019. Peering through the dark: An owl's view of inter-job dependencies and jobs' impact in shared clusters. In *Proceedings of the 2019 International Conference on Management of Data*. 1889–1892.

[11] Gao Cong, Jingyi Yang, and Yue Zhao. 2024. Machine Learning for Databases: Foundations, Paradigms, and Open problems. In *Companion of the 2024 International Conference on Management of Data* (Santiago AA, Chile) *(SIGMOD/PODS '24)*. Association for Computing Machinery, New York, NY, USA, 622–629. https://doi.org/10.1145/3626246.3654686

[12] Carlo Curino, Neha Godwal, Brian Kroth, Sergiy Kuryata, Greg Lapinski, Siqi Liu, Slava Oks, Olga Poppe, Adam Smiechowski, Ed Thayer, et al. 2020. MLOS: An infrastructure for automated software performance engineering. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*. 1–5.

[13] Jörg Domaschka, Mark Leznik, Daniel Seybold, Simon Eismann, Johannes Grohmann, and Samuel Kounev. 2021. Buzzy: Towards Realistic DBMS Benchmarking via Tailored, Representative, Synthetic Workloads: Vision Paper. In *Companion of the ACM/SPEC International Conference on Performance Engineering* (Virtual Event, France) *(ICPE '21)*. Association for Computing Machinery, New York, NY, USA, 175–178. https://doi.org/10.1145/3447545.3451175

[14] Chongjiong Fan, Zhicheng Pan, Wenwen Sun, Chengcheng Yang, and Wei-Neng Chen. 2024. LATuner: An LLM-Enhanced Database Tuning System Based on Adaptive Surrogate Model. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, Albert Bifet, Jesse Davis, Tomas Krilavičius, Meelis Kull, Eirini Ntoutsi, and Indrė Žliobaitė (Eds.). Springer Nature Switzerland, Cham, 372–388.

[15] Peter I. Frazier. 2018. A Tutorial on Bayesian Optimization. arXiv:1807.02811 [stat.ML] https://arxiv.org/abs/1807.02811

[16] Johannes Freischuetz, Konstantinos Kanellis, Brian Kroth, and Shivaram Venkataraman. [n.d.]. Performance Roulette: How Cloud Weather Affects ML-Based System Optimization.

[17] Johannes Freischuetz, Konstantinos Kanellis, Brian Kroth, and Shivaram Venkataraman. 2025. Tuna: Tuning unstable and noisy cloud applications. In *Proceedings of the Twentieth European Conference on Computer Systems*. 954–973.

[18] Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. 2014. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14)*. JMLR.org, II–937–II–945.

[19] Victor Giannankouris and Immanuel Trummer. 2024. lambda-Tune: Harnessing Large Language Models for Automated Database System Tuning. arXiv:2411.03500 [cs.DB] https://arxiv.org/abs/2411.03500

[20] Alibaba Inc. 2024. *Alibaba Open Cluster Trace*. Retrieved April 5, 2024 from https://github.com/alibaba/clusterdata/

[21] Github Inc. 2024. Github CodeSpaces. https://github.com/features/codespaces

[22] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[23] Konstantinos Kanellis, Cong Ding, Brian Kroth, Andreas Müller, Carlo Curino, and Shivaram Venkataraman. 2022. LlamaTune: sample-efficient DBMS configuration tuning. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2953–2965.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[25] Brian Kroth, Sergiy Matusevych, Rana Alotaibi, Yiwen Zhu, Anja Gruenheid, and Yuanyuan Tian. 2024. MLOS in Action: Bridging the Gap Between Experimentation and Auto-Tuning in the Cloud. *Proc. VLDB Endow.* 17, 12 (Nov. 2024), 4269–4272. https://doi.org/10.14778/3685800.3685852

[26] Brian Paul Kroth, Carlo Aldo Curino, and Andreas Christian Mueller. 2023. Performance evaluation of an application based on detecting degradation caused by other computing processes. US Patent 11,816,364.

[27] Jiale Lao, Yibo Wang, Yufei Li, Jianping Wang, Yunjia Zhang, Zhiyuan Cheng, Wanghu Chen, Mingjie Tang, and Jianguo Wang. 2024. GPTuner: A Manual-Reading Database Tuning System via GPT-Guided Bayesian Optimization. *Proc. VLDB Endow.* 17, 8 (May 2024), 1939–1952. https://doi.org/10.14778/3659437.3659449

[28] Guoliang Li, Xuanhe Zhou, Shifu Li, and Bo Gao. 2019. Qtune: A query-aware database tuning system with deep reinforcement learning. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2118–2130.

[29] Guoliang Li, Xuanhe Zhou, Shifu Li, and Bo Gao. 2019. QTune: a query-aware database tuning system with deep reinforcement learning. *Proc. VLDB Endow.* 12, 12 (Aug. 2019), 2118–2130. https://doi.org/10.14778/3352063.3352129

[30] Yang Li, Huaijun Jiang, Yu Shen, Yide Fang, Xiaofeng Yang, Danqing Huang, Xinyi Zhang, Wentao Zhang, Ce Zhang, Peng Chen, and Bin Cui. 2023. Towards General and Efficient Online Tuning for Spark. *Proc. VLDB Endow.* 16, 12 (Aug. 2023), 3570–3583. https://doi.org/10.14778/3611540.3611548

[31] Chen Lin, Junqing Zhuang, Jiadong Feng, Hui Li, Xuanhe Zhou, and Guoliang Li. 2022. Adaptive code learning for spark configuration tuning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 1995–2007.

[32] Chenghao Lyu, Qi Fan, Philippe Guyard, and Yanlei Diao. 2024. A Spark Optimizer for Adaptive, Fine-Grained Parameter Tuning. *Proc. VLDB Endow.* 17, 11 (Aug. 2024), 3565–3579. https://doi.org/10.14778/3681954.3682021

[33] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making learned query optimization practical. In *Proceedings of the 2021 International Conference on Management of Data*. 1275–1288.

[34] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software* 3, 29 (2018), 861. https://doi.org/10.21105/joss.00861

[35] Microsoft. 2024. *Configure Autotune for Fabric Spark*. Retrieved June 27, 2024 from https://learn.microsoft.com/en-us/fabric/data-engineering/autotune?tabs=sparksql

[36] Microsoft. 2024. Developing inside a Container. https://code.visualstudio.com/docs/devcontainers/containers

[37] Parimarjan Negi, Matteo Interlandi, Ryan Marcus, Mohammad Alizadeh, Tim Kraska, Marc Friedman, and Alekh Jindal. 2021. Steering query optimizers: A practical take on big data workloads. In *Proceedings of the 2021 International Conference on Management of Data*. 2557–2569.

[38] OpenAI. 2024. *New embedding models and API updates*. Retrieved July 5, 2024 from https://openai.com/index/new-embedding-models-and-api-updates/

[39] Jim R Parker. 2010. *Algorithms for image processing and computer vision*. John Wiley & Sons.

[40] Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C Mowry, Matthew Perron, Ian Quah, et al. 2017. Self-Driving Database Management Systems.. In *CIDR*, Vol. 4. 1.

[41] David Buchaca Prats, Felipe Albuquerque Portella, Carlos H. A. Costa, and Josep Lluis Berral. 2020. You Only Run Once: Spark Auto-Tuning From a Single Run. *IEEE Transactions on Network and Service Management* 17, 4 (2020), 2039–2051. https://doi.org/10.1109/TNSM.2020.3034824

[42] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. 2012. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the third ACM symposium on cloud computing*. 1–13.

[43] Yu Shen, Xinyuyang Ren, Yupeng Lu, Huaijun Jiang, Huanyong Xu, Di Peng, Yang Li, Wentao Zhang, and Bin Cui. 2023. Rover: An Online Spark SQL Tuning Service via Generalized Transfer Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Long Beach, CA, USA) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 4800–4812.

https://doi.org/10.1145/3580305.3599953

[44] Yu Shen, Xinyuyang Ren, Yupeng Lu, Huaijun Jiang, Huanyong Xu, Di Peng, Yang Li, Wentao Zhang, and Bin Cui. 2023. Rover: An online Spark SQL tuning service via generalized transfer learning. *arXiv preprint arXiv:2302.04046* (2023).

[45] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.

[46] Gagan Somashekar, Karan Tandon, Anush Kini, Chieh-Chun Chang, Petr Husak, Ranjita Bhagwan, Mayukh Das, Anshul Gandhi, and Nagarajan Natarajan. 2024. {OPPerTune}:{Post-Deployment} Configuration Tuning of Services Made Easy. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 1101–1120.

[47] Immanuel Trummer. 2022. DB-BERT: A Database Tuning Tool that "Reads the Manual". In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 190–203. https://doi.org/10.1145/3514221.3517843

[48] Dana Van Aken, Andrew Pavlo, Geoffrey J Gordon, and Bohan Zhang. 2017. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM international conference on management of data*. 1009–1024.

[49] Dana Van Aken, Dongsheng Yang, Sebastien Brillard, Ari Fiorino, Bohan Zhang, Christian Bilien, and Andrew Pavlo. 2021. An inquiry into machine learning-based automatic configuration tuning services on real-world database management systems. *Proc. VLDB Endow.* 14, 7 (March 2021), 1241–1253. https://doi.org/10.14778/3450980.3450992

[50] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[51] Minh Vu and My T. Thai. 2020. PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 12225–12235. https://proceedings.neurips.cc/paper_files/paper/2020/file/8fb134f258b1f7865a6ab2d935a897c9-Paper.pdf

[52] Junxiong Wang, Immanuel Trummer, and Debabrota Basu. 2021. UDO: universal database optimization using reinforcement learning. *Proc. VLDB Endow.* 14, 13 (Sept. 2021), 3402–3414. https://doi.org/10.14778/3484224.3484236

[53] Wikipedia. 2023. Pareto front. https://en.wikipedia.org/wiki/Pareto_front.

[54] Zhibin Yu, Zhendong Bei, and Xuehai Qian. 2018. Datasize-Aware High Dimensional Configurations Auto-Tuning of In-Memory Cluster Computing. *SIGPLAN Not.* 53, 2 (March 2018), 564–577. https://doi.org/10.1145/3296957.3173187

[55] Bohan Zhang, Dana Van Aken, Justin Wang, Tao Dai, Shuli Jiang, Jacky Lao, Siyuan Sheng, Andrew Pavlo, and Geoffrey J Gordon. 2018. A demonstration of the ottertune automatic database management system tuning service. *Proceedings of the VLDB Endowment* 11, 12 (2018), 1910–1913.

[56] Ji Zhang, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, Tianheng Cheng, Li Liu, et al. 2019. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In *Proceedings of the 2019 International Conference on Management of Data*. 415–432.

[57] William Zhang, Wan Shen Lim, Matthew Butrovich, and Andrew Pavlo. 2024. The Holon Approach for Simultaneously Tuning Multiple Components in a Self-Driving Database Management System with Machine Learning via Synthesized Proto-Actions. *Proceedings of the VLDB Endowment* 17, 11 (2024), 3373–3387.

[58] Xinyi Zhang, Zhuo Chang, Yang Li, Hong Wu, Jian Tan, Feifei Li, and Bin Cui. 2022. Facilitating database tuning with hyper-parameter optimization: a comprehensive experimental evaluation. *Proceedings of the VLDB Endowment* 15, 9 (2022), 1808–1821.

[59] Xinyi Zhang, Zhuo Chang, Hong Wu, Yang Li, Jia Chen, Jian Tan, Feifei Li, and Bin Cui. 2023. A Unified and Efficient Coordinating Framework for Autonomous DBMS Tuning. *Proc. ACM Manag. Data* 1, 2, Article 186 (June 2023), 26 pages. https://doi.org/10.1145/3589331

[60] Xinyi Zhang, Hong Wu, Zhuo Chang, Shuowei Jin, Jian Tan, Feifei Li, Tieying Zhang, and Bin Cui. 2021. ResTune: Resource Oriented Tuning Boosted by Meta-Learning for Cloud Databases. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 2102–2114. https://doi.org/10.1145/3448016.3457291

[61] Xinyi Zhang, Hong Wu, Yang Li, Jian Tan, Feifei Li, and Bin Cui. 2022. Towards dynamic and safe configuration tuning for cloud databases. In *Proceedings of the 2022 International Conference on Management of Data*. 631–645.

[62] Xinyi Zhang, Hong Wu, Yang Li, Jian Tan, Feifei Li, and Bin Cui. 2022. Towards Dynamic and Safe Configuration Tuning for Cloud Databases. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 631–645. https://doi.org/10.1145/3514221.3526176

[63] Xinyi Zhang, Hong Wu, Yang Li, Zhengju Tang, Jian Tan, Feifei Li, and Bin Cui. 2023. An Efficient Transfer Learning Based Configuration Adviser for Database Tuning. *Proc. VLDB Endow.* 17, 3 (Nov. 2023), 539–552. https://doi.org/10.14778/3632093.3632114

[64] Xinyang Zhao, Xuanhe Zhou, and Guoliang Li. 2023. Automatic Database Knob Tuning: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2023), 12470–12490. https://doi.org/10.1109/TKDE.2023.3266893

[65] Xinran Zhu, Yang Liu, Pieter Ghysels, David Bindel, and Xiaoye S. Li. [n.d.]. *GPTuneBand: Multi-task and Multi-fidelity Autotuning for Large-scale High Performance Computing Applications*. 1–13. https://doi.org/10.1137/1.9781611977141.1 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611977141.1

[66] Yiwen Zhu, Subru Krishnan, Konstantinos Karanasos, Isha Tarte, Conor Power, Abhishek Modi, Manoj Kumar, Deli Zhang, Kartheek Muthyala, Nick Jurgens, et al. 2021. KEA: Tuning an Exabyte-Scale Data Infrastructure. In *Proceedings of the 2021 International Conference on Management of Data*. 2667–2680.

[67] Yuqing Zhu, Jianxun Liu, Mengying Guo, Yungang Bao, Wenlong Ma, Zhuoyue Liu, Kunpeng Song, and Yingchun Yang. 2017. BestConfig: tapping the performance potential of systems via automatic configuration tuning. In *Proceedings of the 2017 Symposium on Cloud Computing* (Santa Clara, California) *(SoCC '17)*. Association for Computing Machinery, New York, NY, USA, 338–350. https://doi.org/10.1145/3127479.3128605

[68] Yiwen Zhu, Yuanyuan Tian, Joyce Cahoon, Subru Krishnan, Ankita Agarwal, Rana Alotaibi, Jesús Camacho-Rodríguez, Bibin Chundatt, Andrew Chung, Niharika Dutta, et al. 2023. Towards Building Autonomous Data Services on Azure. In *Companion of the 2023 International Conference on Management of Data*. 217–224.