Towards Online and Safe Configuration Tuning with Semi-supervised Anomaly Detection

연세대학교 컴퓨터과학과 김정은

2024년 11월

과제명: IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

Background

- Database Parameter Tuning

▪ Database tuning is to **enhance the performance of database**, there are various tuning techniques available.

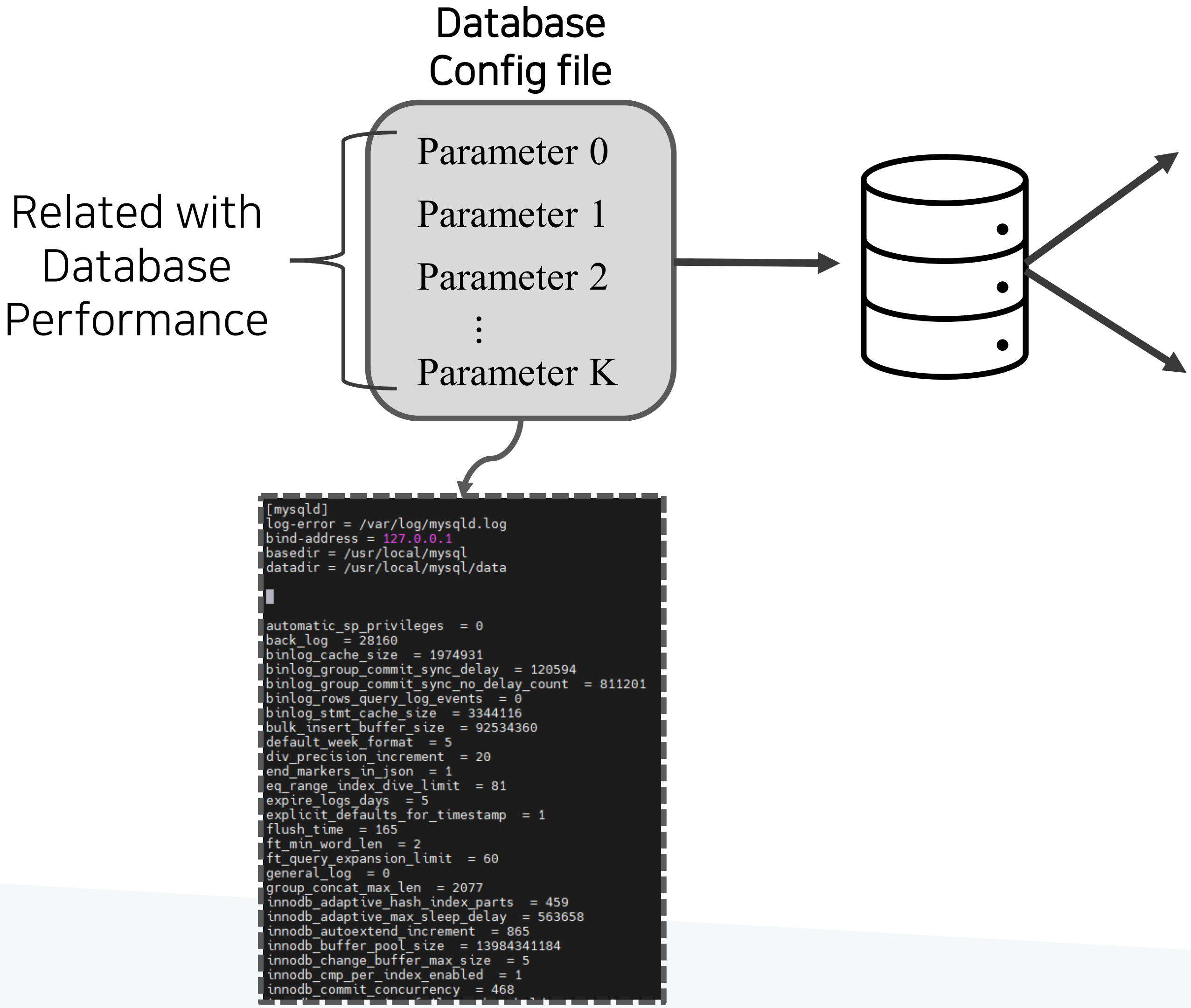
▪ Database Configuration
  - Knob Tuning : Automating parameter optimization.
  - Index Advisor: Recommending indexes for efficient query execution.
  - View Advisor: Suggesting materialized views to improve query performance.
  - SQL Rewriter: Enhancing query structure by rewriting inefficient SQL.

Fig. 1. The overview of DB meets AI.

AI Meets Database: AI4DB and DB4AI (SIGMOD'21)

Background

- Database Parameter Tuning



① Throughput (↑)

Typically refers to the **rate** at which a database system can **process transactions or queries** with a given time frame.

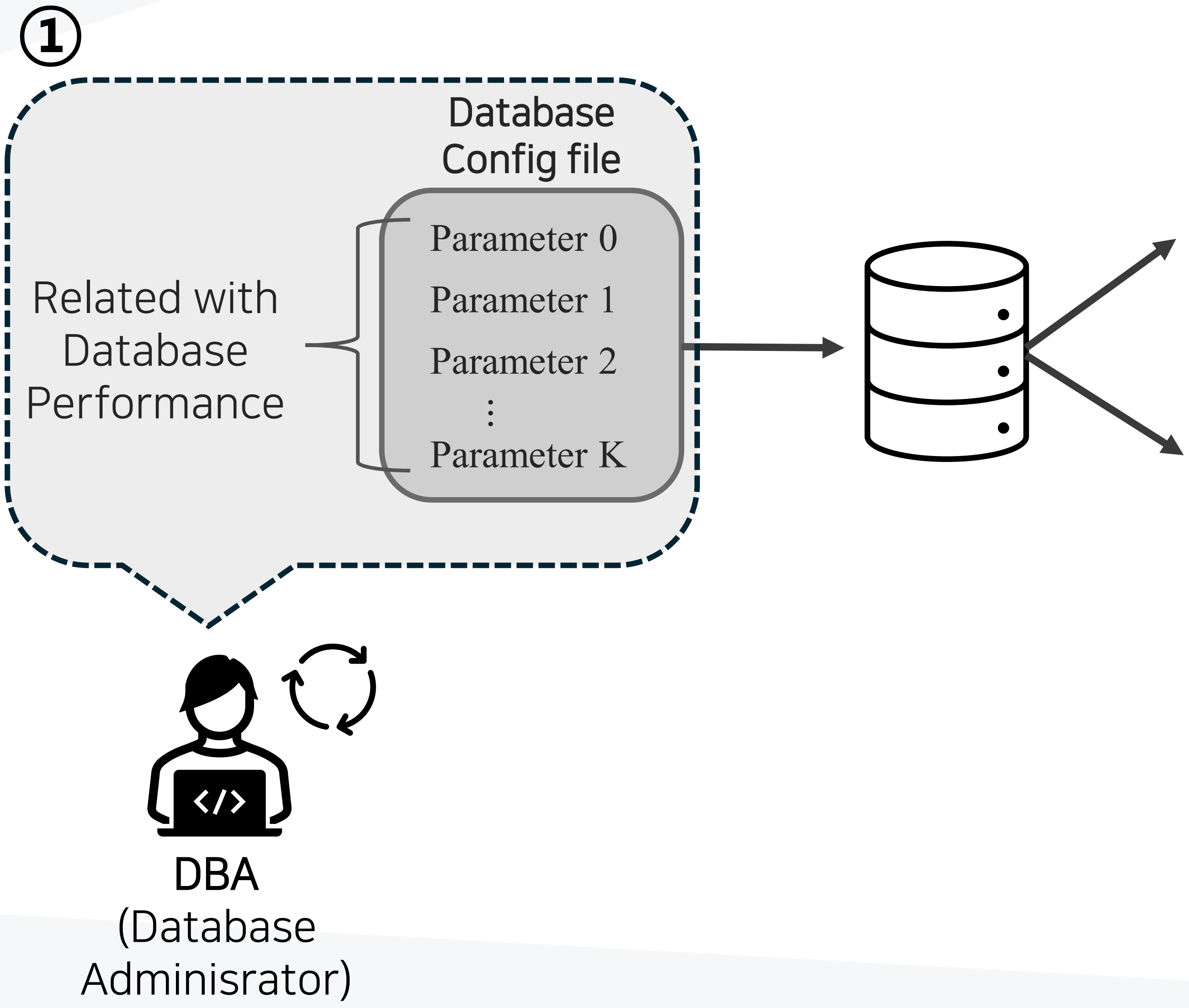
② Latency (↓)

The amount of **time it takes for a database operation** such as executing a query or performing an update, to complete from the moment it is **initiated until the moment the result is returned** or the operation is completed.

https://cloud.google.com/sql/docs/mysql/flags?hl=ko

Background

- Database Parameter Tuning



① **Throughput (↑)**

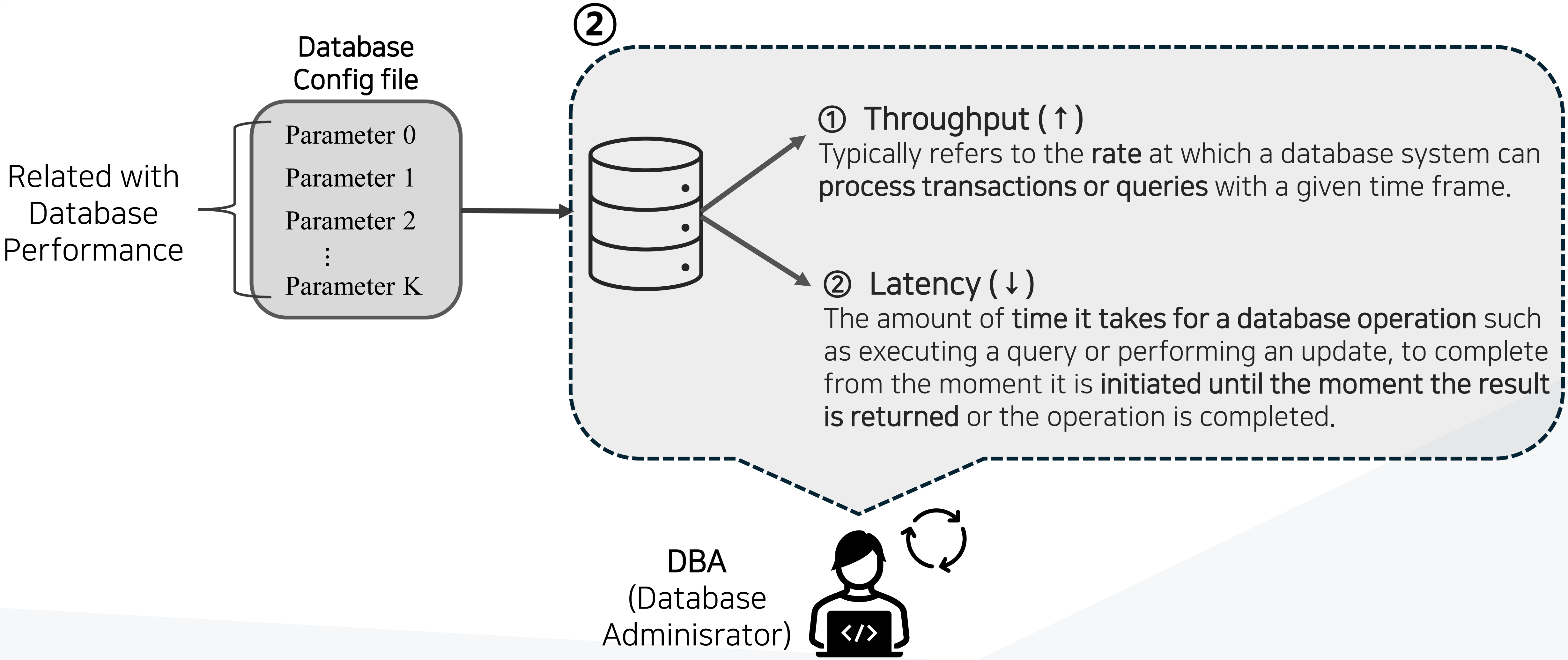
Typically refers to the **rate** at which a database system can **process transactions or queries** with a given time frame.

② **Latency (↓)**

The amount of **time it takes for a database operation** such as executing a query or performing an update, to complete from the moment it is **initiated until the moment the result is returned** or the operation is completed.

Background

- Database Parameter Tuning



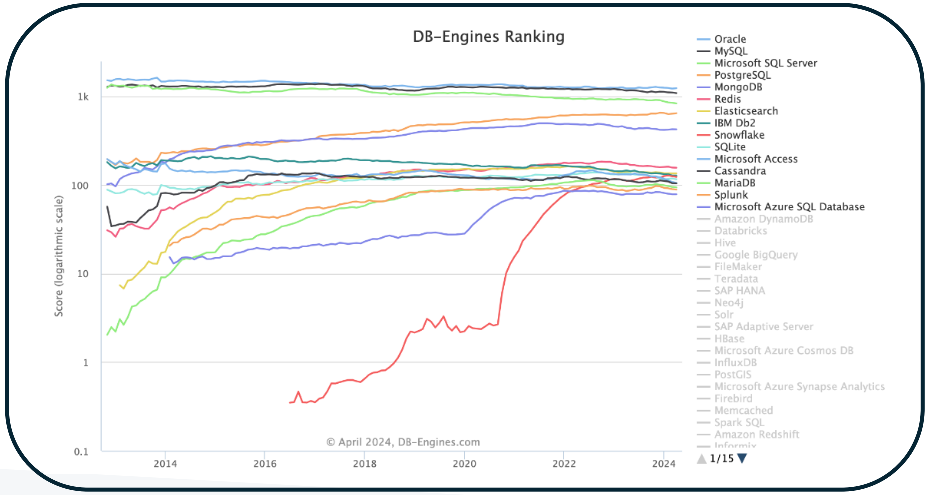
Background

- Database Parameter Tuning Limitation

① **Increasing** number of **database parameters** and increasing **database types.**

② Database database **versions are updated with various parameter configurations**, posing challenges for DBA to manually adjust tuning strategies according to version changing.

③ **Diverse of database workloads**, it is infeasible for DBAs to manually optimize for every possible workloads.



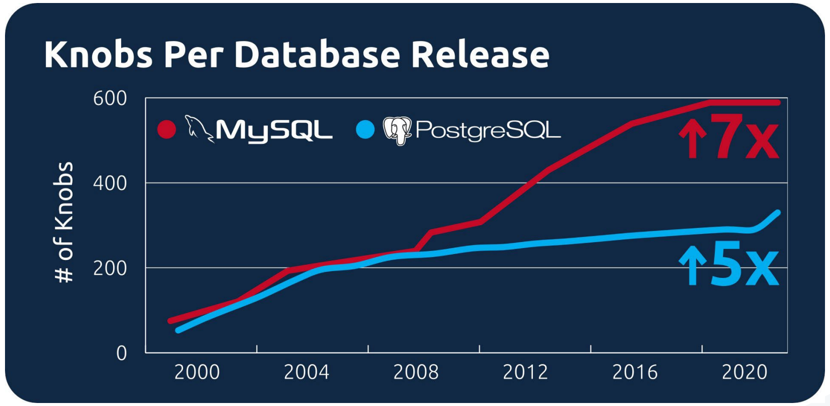


Table 3: MySQL Workload Information

Workload Index	Scale Factor	Data Size	Read	Insert	Scan	Update	Read Modify Write
A	12000	15GB	**50%**	-	-	**50%**	-
B			**95%**	-	-	**5%**	-
E			-	**5%**	**95%**	-	-
F			**50%**	-	-	-	**50%**

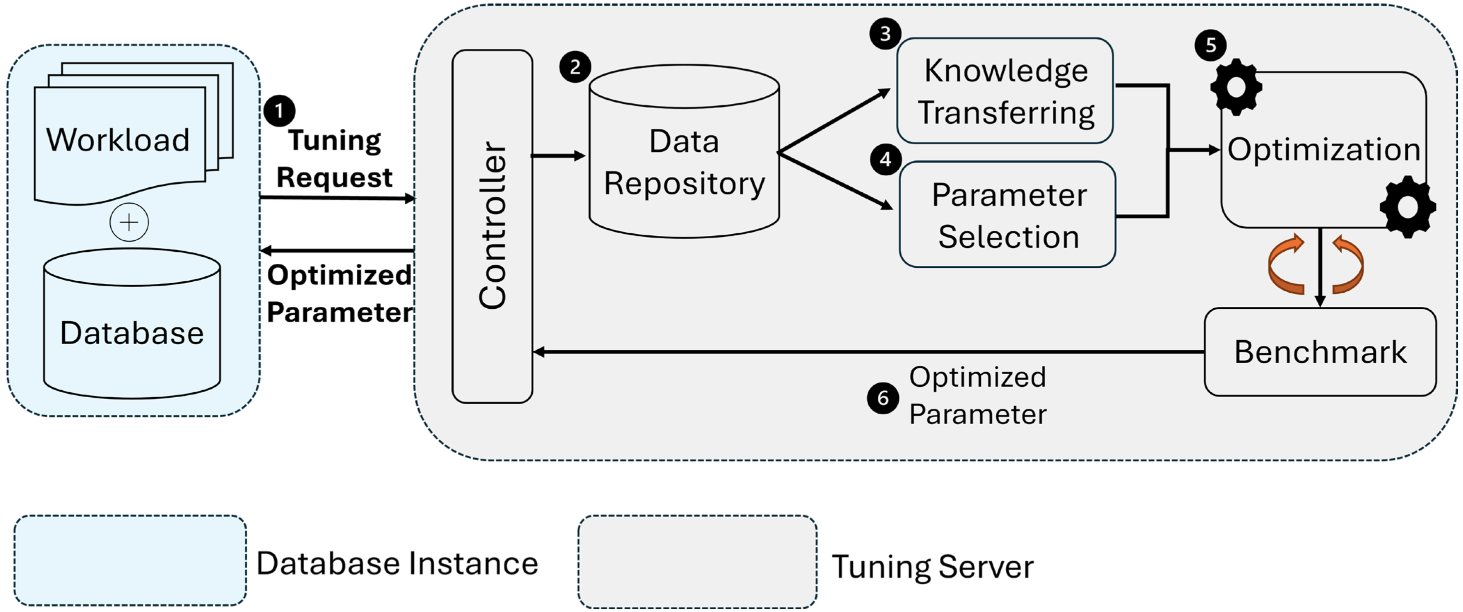
Table 4: RocksDB Workload Information

Workload Index	Value Size	# of Entry	READ	WRITE	UPDATE
R90W10	16384	65472	90%	10%	X
R50W50			50%	50%	
R10W90			10%	90%	
UPDATE			-	-	O

https://db-engines.com/en/ranking
Automatic Database Management System Tuning Through Large-scale Machine Learning SIGMOD'17

Background

- Automatic Database Parameter Tuning



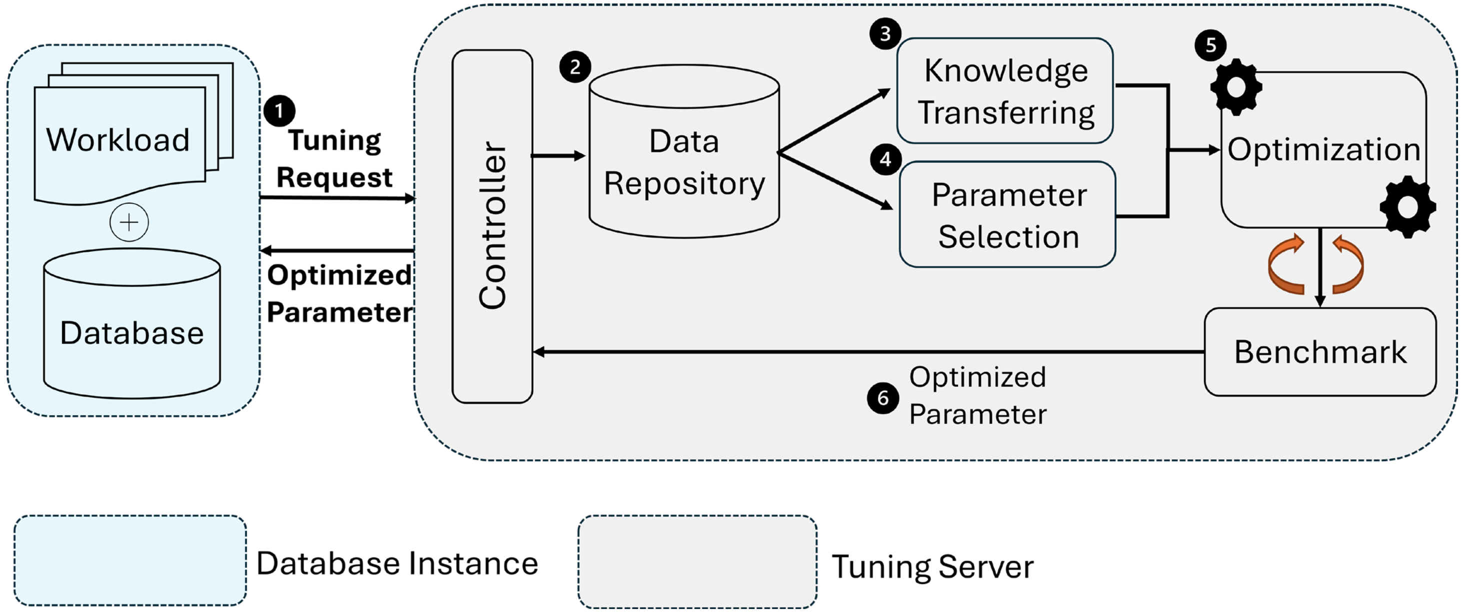
① **Tuning Request.** When the Controller receives information about the DBMS and workload requiring tuning.

② **Data Repository.** During the tuning process, the DBMS and workload information provided in the tuning request are stored in the data repository.

③ **Knowledge Transferring.** To optimize the various workload, this process employs a similarity calculation between the target and the stored workloads in the data repository, utilizing the most similar workload information for the tuning process.

Background

- Automatic Database Parameter Tuning



④ **Parameter Selection.** To address the difficulty of optimization in high dimensional search spaces, the most influential parameters on database performance are selected by a parameter selection algorithm.

⑤ **Optimization.** The optimization algorithm optimizes the top-k parameters that have a significant impact on database performance (④) and information about the target workload (③).

⑥ **Optimized Parameter.** The optimized parameters are passed to the controller, which then applies these parameters in the actual database.

Towards Online and Safe Configuration Tuning with Semi-supervised Anomaly Detection

Haitian Chen
Shenzhen Institute for Advanced Study
University of Electronic Science and Technology of China
Shenzhen, China
haitianchen@std.uestc.edu.cn

Xu Chen
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
xuchen@std.uestc.edu.cn

Zibo Liang
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
zbliang@std.uestc.edu.cn

Xiushi Feng
Shenzhen Institute for Advanced Study
University of Electronic Science and Technology of China
Shenzhen, China
xiushifeng@std.uestc.edu.cn

Jiandong Xie
Huawei Technologies Co., Ltd.
Chengdu, China
xiejiandong@huawei.com

Han Su
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
hansu@uestc.edu.cn

Kai Zheng*
Shenzhen Institute for Advanced Study
University of Electronic Science and Technology of China
Shenzhen, China
zhengkai@uestc.edu.cn

CIKM 2024

Limitation & Contribution

- **L1. Static Workload Dependency**: Existing machine learning-based tuning methods perform well on static workloads but struggle to adapt to dynamic workloads, resulting in performance degradation.
- **L2. Safety Concerns**: Traditional approaches often lack mechanisms to ensure safe configuration sampling, leading to significant performance fluctuations during tuning.
- L3. Inefficiency in Sampling: Existing methods require extensive sampling to achieve optimal configurations, which is inefficient and time-consuming.
- **L4. High Cost of Ownership**: Offline tuning approaches necessitate infrastructure replication, increasing the total cost of ownership (TCO).
- **L5. Business Disruption:** Offline tuning may lead to temporary service halts, making it unsuitable for real-world, continuous-use environments.

Limitation & Contribution

- **C1.** Introduces SafeTune, the first system combining anomaly detection with configuration tuning to enhance safety and performance stability in real-time. Ensures configurations remain above a safety threshold during tuning, reducing risks of performance degradation.
- **C2.** Utilizes semi-supervised anomaly detection for high-quality feature representation. Employs a ranking-based supervised classifier to refine the detection of unsafe configurations.
- **C3**. Demonstrates adaptability to dynamic workloads, ensuring tuning remains relevant as conditions change.
- **C4.** Leverages historical tuning data to provide high-quality initial configurations, significantly accelerating the tuning process.

Method

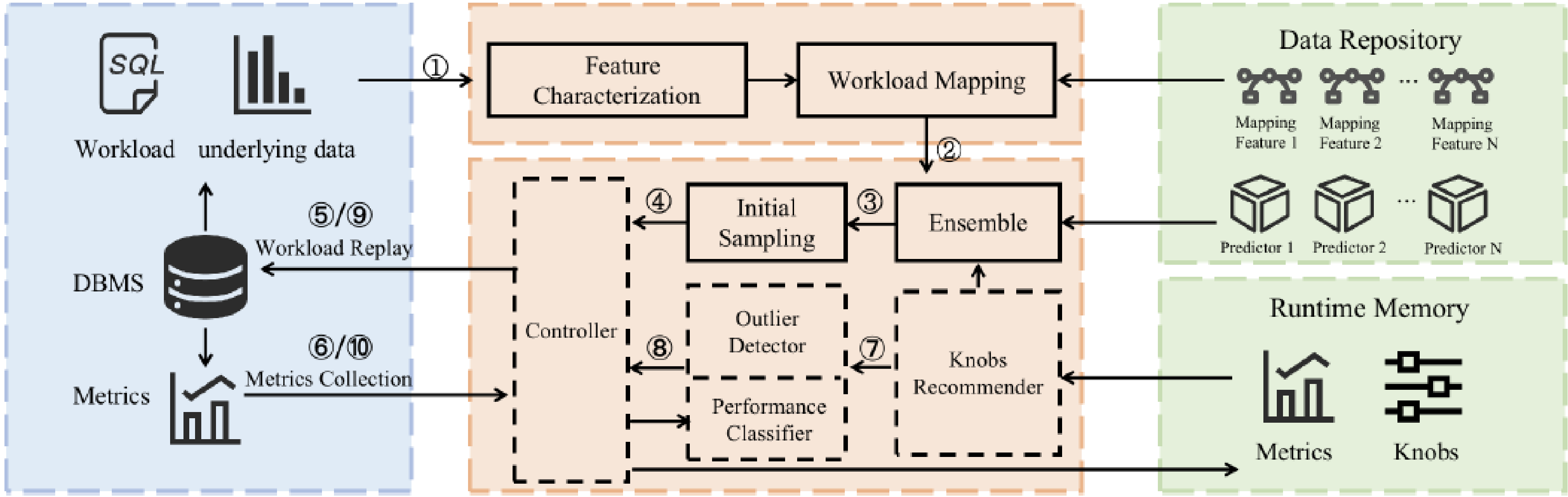


Figure 2: Overview Architecture and Workflow of SafeTune

① Two-Stage Filtering for Safe Configuration

- **Anomaly Detection:** Identifies unsafe configurations by treating them as anomalies using unsupervised methods like KNN and Isolation Forest.
- Transforms configurations into an outlier feature space for robust safety detection.

Method

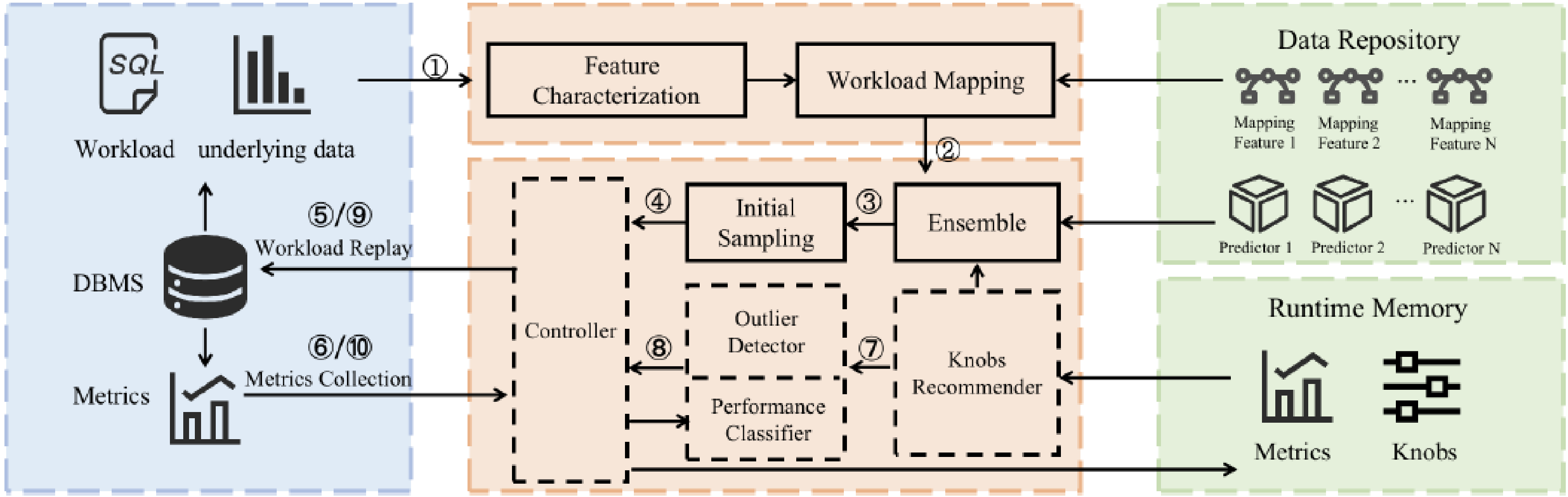


Figure 2: Overview Architecture and Workflow of SafeTune

① Two-Stage Filtering for Safe Configuration

- **Ranking-Based Classification**: Ranks configurations using a supervised classifier (e.g., XGBoost) trained on performance data.
- Refines safety detection by learning from historical tuning observations.

Method

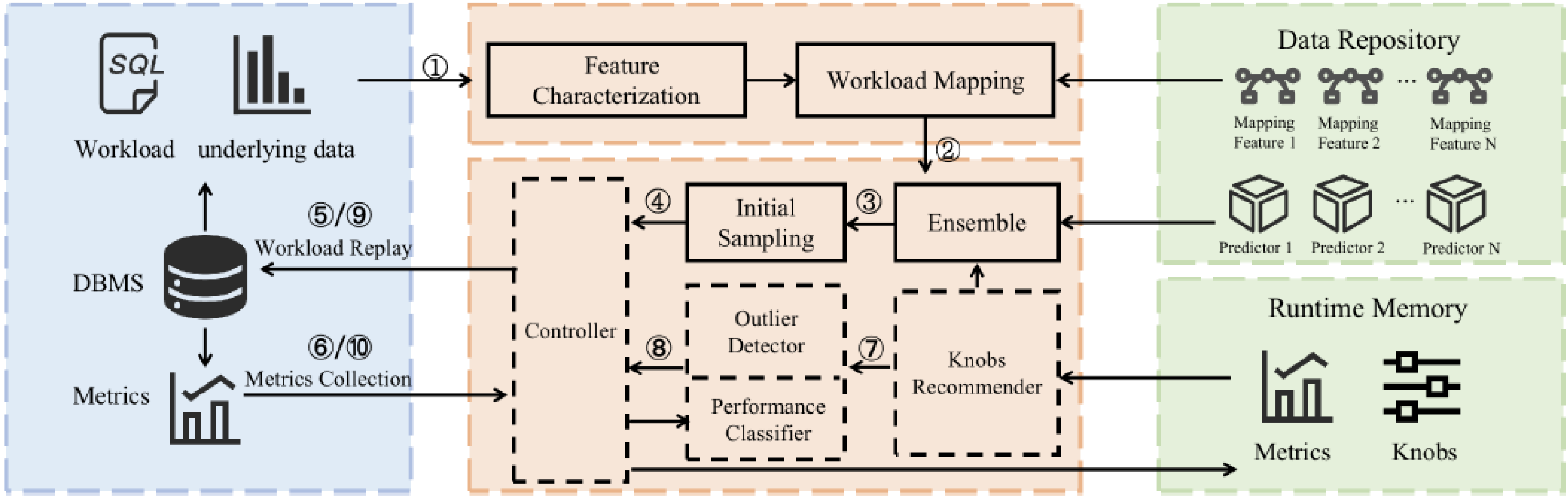


Figure 2: Overview Architecture and Workflow of SafeTune

④ Adapting to Dynamic Workloads

- Divides tuning into sub-tasks and re-initializes each phase with updated knowledge.
- Dynamically updates its anomaly detector and classifier based on the latest observations.

Experiments

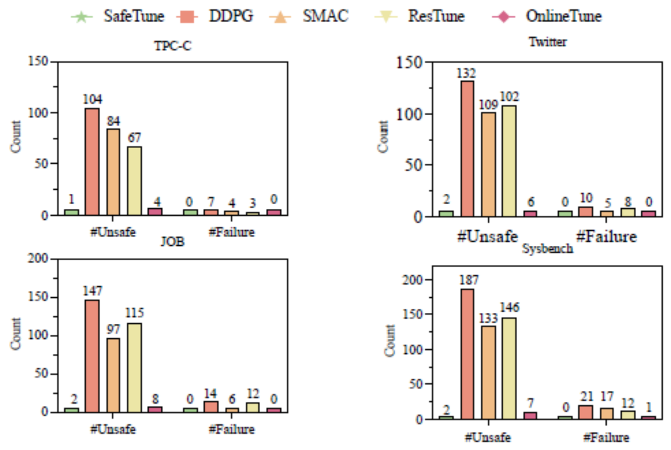


Figure 3: Safety for static workloads: Each workload is evaluated with 300 iterations.

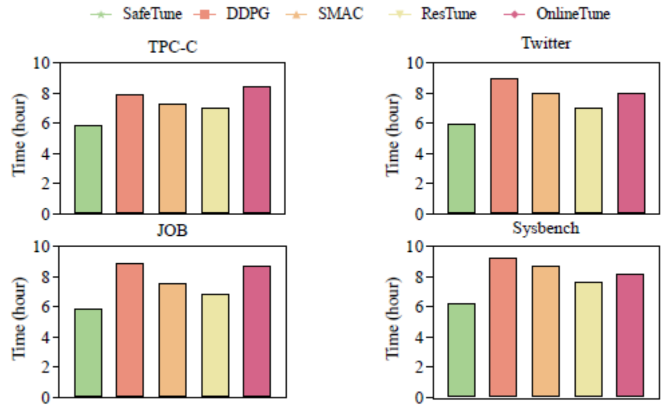


Figure 4: Tuning overhead for static workloads: Time required for each method to converge.

- SafeTune achieves the **highest level of safety**, significantly reducing unsafe configurations and system failures across all workload.
- OnlineTune also maintains safety but shows slightly higher unsafe configurations than SafeTune.
- However, **offline methods** like DDPG, SMAC, and ResTune exhibit **poor safety** performance

Experiments

- Safety Comparison

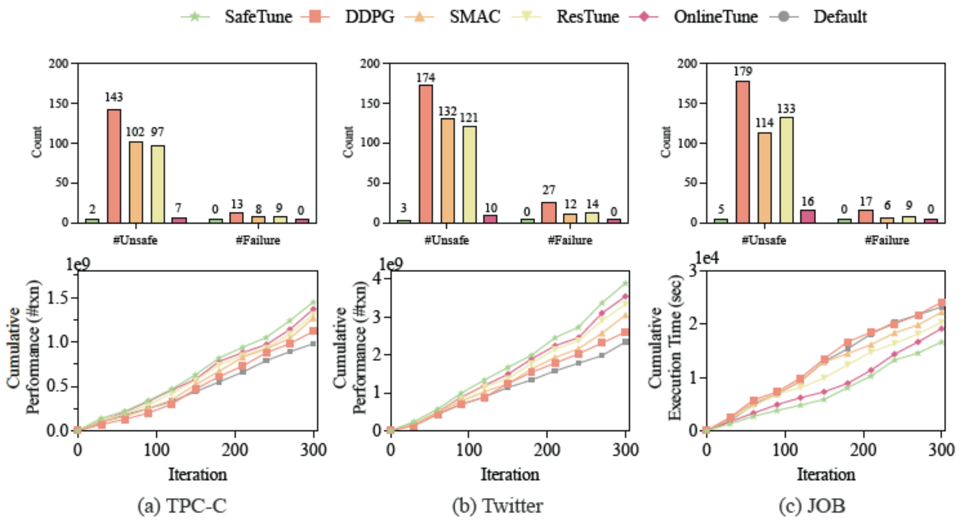


Figure 5: Cumulative performance and safety statistics during the tuning of dynamic workloads.

- SafeTune
  - Significantly **reduces unsafe configurations** and failures compared to all other methods.
  - Consistently achieves **the lowest number of unsafe configurations** (e.g., 2–5 across workloads) and near-zero failures.
- OnlineTune
  - Performs better than offline methods but still has higher unsafe suggestions than SafeTune.

Experiments

- Safety Comparison

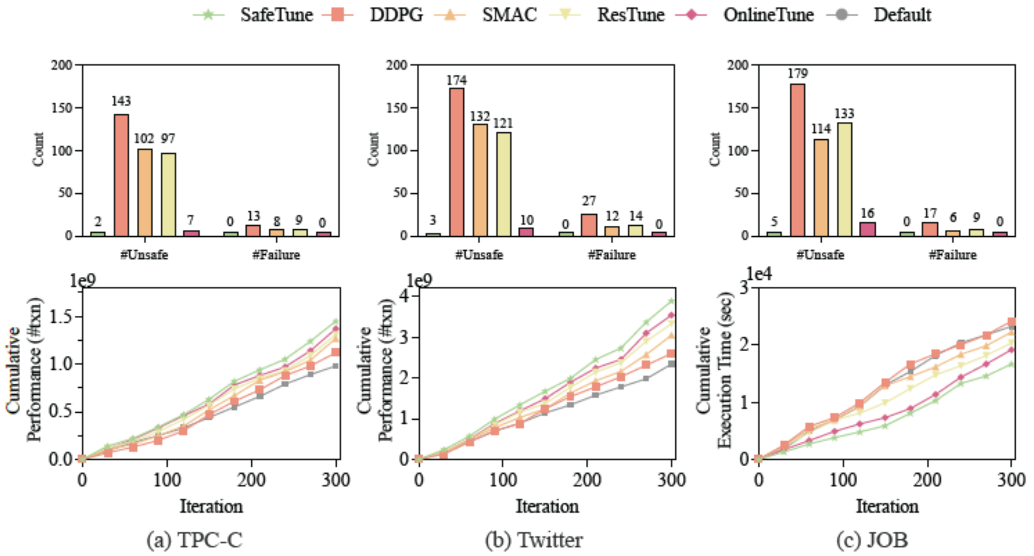


Figure 5: Cumulative performance and safety statistics during the tuning of dynamic workloads.

- Offline Methods (DDPG, SMAC, ResTune)
  - Exhibit a large number of unsafe configurations and failures, highlighting the inability to handle dynamic workloads effectively.

Experiments

- Initialization Sampling

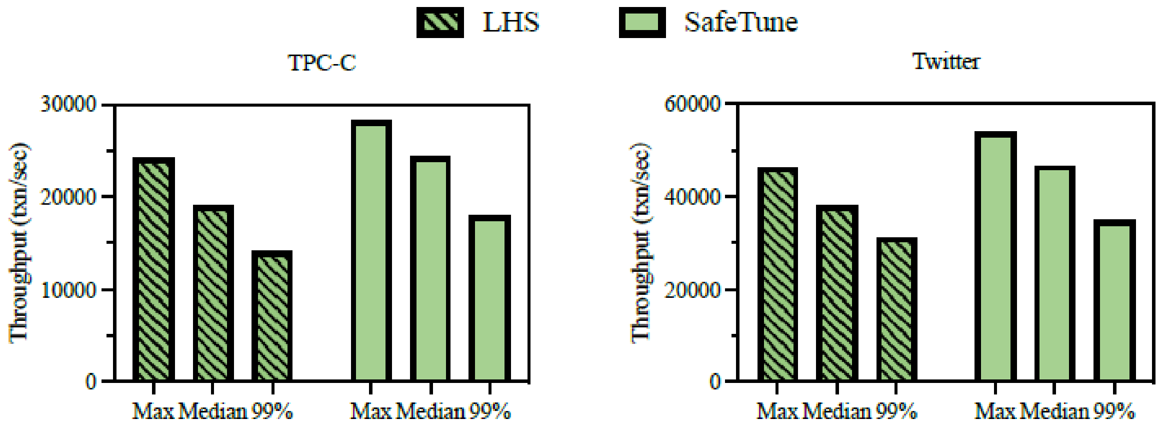


Figure 6: Initialization Sampling: Each method was tested with 15 samples per workload, and the experiment was conducted three times to obtain an average value.

- For both TPC-C and Twitter workloads, SafeTune consistently **achieves higher maximum, median, and 99% throughput** compared to LHS.
- In the TPC-C workload, **SafeTune's maximum throughput** is significantly higher, reflecting its ability to identify more optimal configurations early.
- In the Twitter workload, the gap between SafeTune and LHS is even more pronounced, especially in the maximum throughput metric, showcasing SafeTune's effectiveness in **identifying high-performance configurations.**

Thank You for Listening