



연세대학교
YONSEI UNIVERSITY

Automatic Database Configuration Tuning

Database System

1st semester, 2024

Sein Kwon

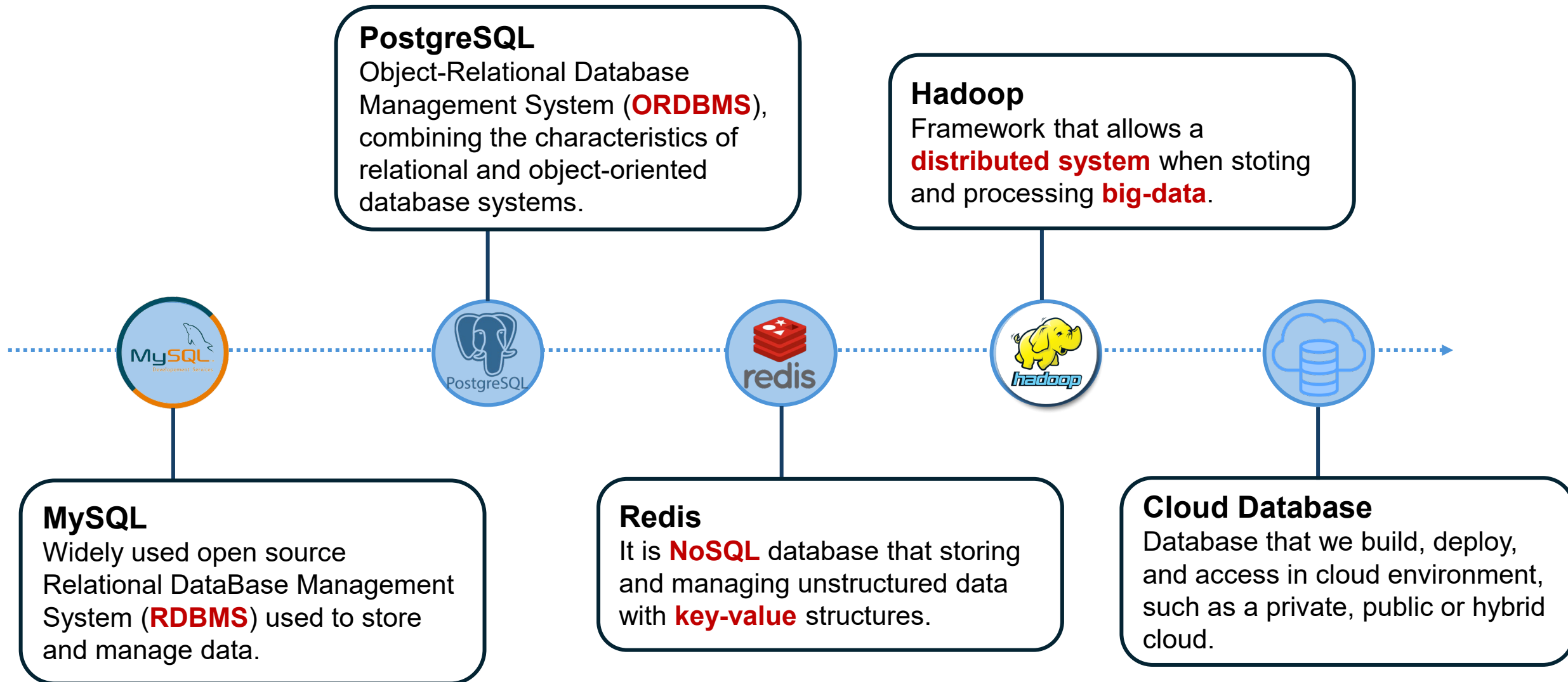
seinkwon97@yonsei.ac.kr

- 1 What is Database Tuning?**
- 2 Automatic Database Configuration Tuning**
- 3 Related Papers**

1 What is Database Tuning?

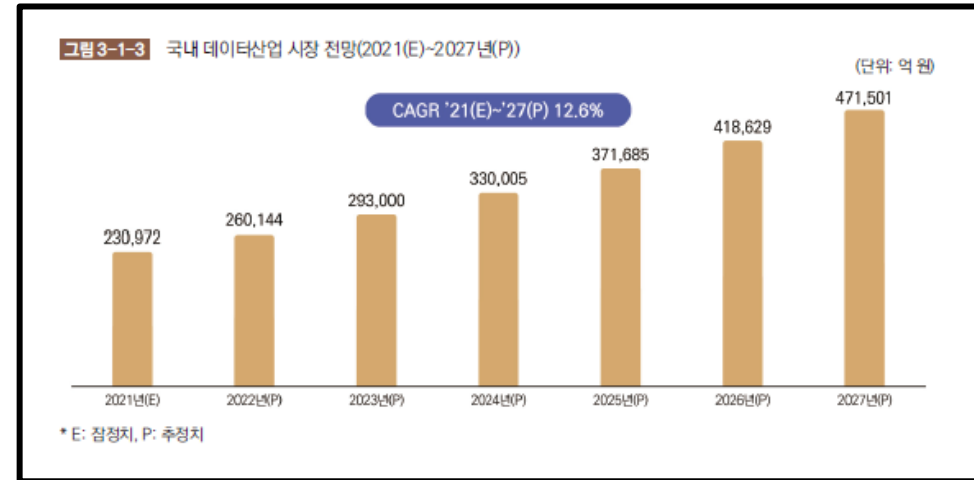
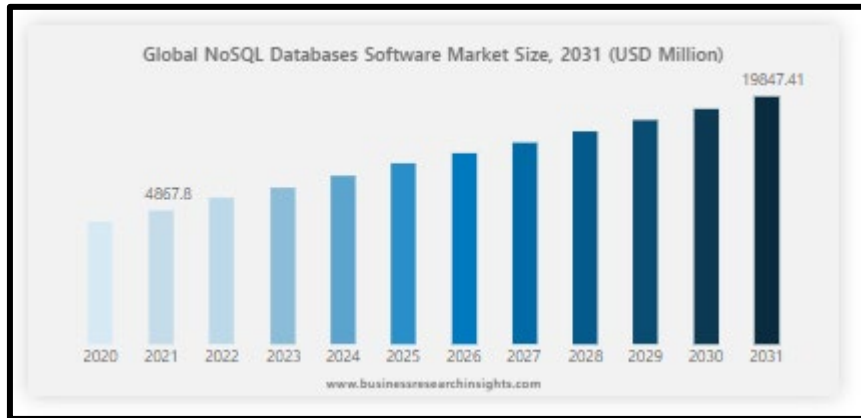
1. What is Database Tuning?

- Type of Database



1. What is Database Tuning?

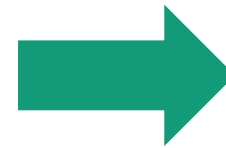
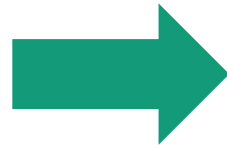
- Database Tuning



- A database not only systematically collects data, but also has the ability to search and delete data.
- So, in most fields, the **use of databases is essential** to systematically collect and manage data.
- Database Industry is continuously expanding, and with the increasing volume of data, the **importance of databases is growing**.

1. What is Database Tuning?

- Database Tuning



- When using various databases, **additional database performance improvement is required** to efficiently process large amounts of data.
- There are several ways to improve database performance, such as SQL tuning, database parameter tuning.

1. What is Database Tuning?

- **Database Tuning**
- To enhance the performance of a Database, there are **various tuning techniques** available.

Techniques	Key Contents
Configuration Tuning	Optimizing database parameter to enhance performance improvements.
Revising Design	Analyzing the impact of information access and deriving performance improvements through.
SQL query optimization	Adjusting SQL queries considering the nature of the optimizer based on analysis of executing query execution plans.
Optimizing distributed architecture	Designing distributed models considering network load.

1. What is Database Tuning?

- **Database Tuning**
- To enhance the performance of a Database, there are **various tuning techniques** available.

Techniques	Key Contents
Configuration Tuning	Optimizing database parameter to enhance performance improvements.
Revising Design	Analyzing the impact of information access and deriving performance improvements through.
SQL query optimization	Adjusting SQL queries considering the nature of the optimizer based on analysis of executing query execution plans.
Optimizing distributed architecture	Designing distributed models considering network load.

1. What is Database Tuning?

- **Database Parameter Tuning**

- Database Parameters are important in controlling and tuning the behavior of a database system.

Techniques	Key Contents
Configuration Tuning	Optimizing database parameter to enhance performance improvements.

① Performance Enhancement

Adjusting parameters related to **memory allocation** or **query execution plans** can **optimize the processing speed** of the database.

② Stability and Reliability

Optimized parameters can increase the **stability and reliability** of the database system. This helps maintain data **consistency** and **integrity** while minimizing system failures.

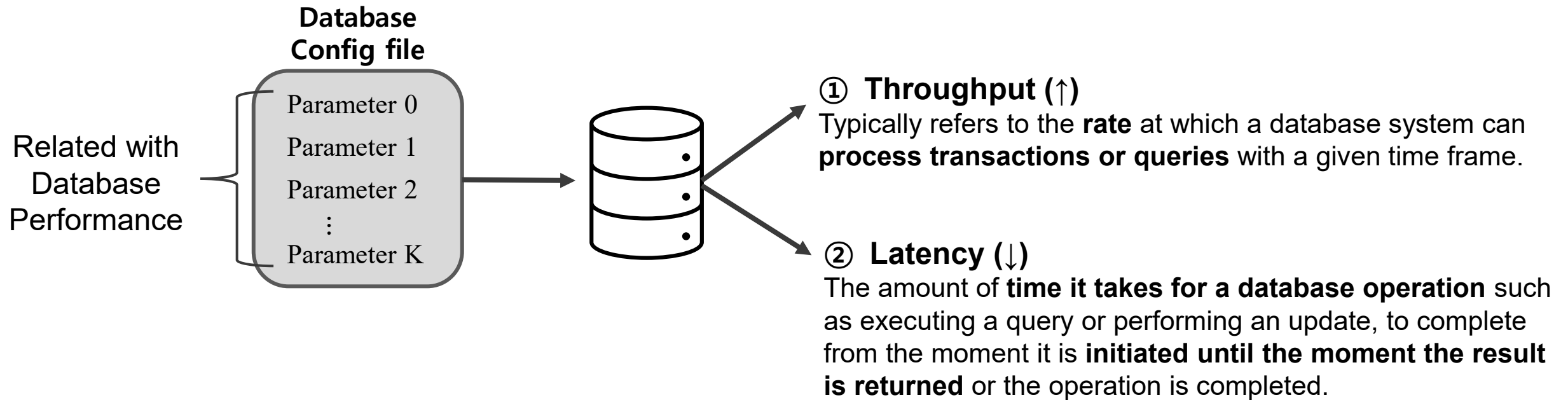
③ Resource Management

Parameter settings are crucial for efficiently managing the **resources** used by the database system (e.g., memory, disk space).

Automatic Database Configuration Tuning

2-1. Database Parameter Tuning

- Database Configuration Tuning



2-1. Database Parameter Tuning

- Database Configuration Tuning

real data

Database Config file

```
[mysqld]
log-error = /var/log/mysqld.log
bind-address = 127.0.0.1
basedir = /usr/local/mysql
datadir = /usr/local/mysql/data

automatic_sp_privileges = 0
back_log = 28160
binlog_cache_size = 1974931
binlog_group_commit_sync_delay = 120594
binlog_group_commit_sync_no_delay_count = 811201
binlog_rows_query_log_events = 0
binlog_stmt_cache_size = 3344116
bulk_insert_buffer_size = 92534360
default_week_format = 5
div_precision_increment = 20
end_markers_in_json = 1
eq_range_index_dive_limit = 81
expire_logs_days = 5
explicit_defaults_for_timestamp = 1
flush_time = 165
ft_min_word_len = 2
ft_query_expansion_limit = 60
general_log = 0
group_concat_max_len = 2077
innodb_adaptive_hash_index_parts = 459
innodb_adaptive_max_sleep_delay = 563658
innodb_autoextend_increment = 865
innodb_buffer_pool_size = 13984341184
innodb_change_buffer_max_size = 5
innodb_cmp_per_index_enabled = 1
innodb_commit_concurrency = 468
```

Database Restart

```
gen@db-server1:~$ service mysql restart
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to restart 'mysql.server.service'.
Multiple identities can be used for authentication:
 1. labadmin,,, (labadmin)
 2. gen
Choose identity to authenticate as (1-2): 2
Password:
==== AUTHENTICATION COMPLETE ====
gen@db-server1:~$
```

Database Metrics

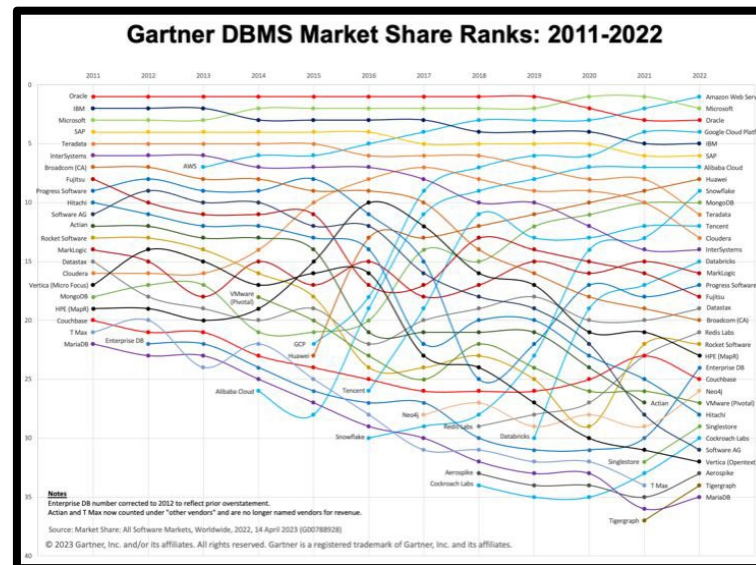
'Throughput (requests/second)': 3782.3240818820987}

'99th Percentile Latency (microseconds)': 1473,

2-1. Database Parameter Tuning

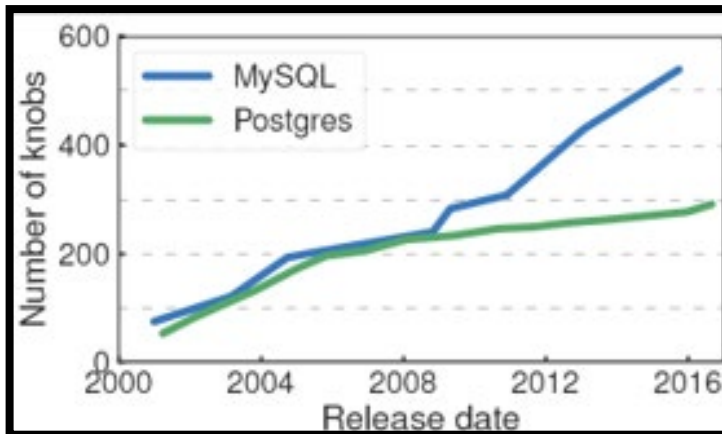
- **Database Configuration Tuning Limitation**

- Traditionally, database parameter tuning involved **Database Administrators (DBA) manually adjusting various database parameter values** to enhance the performance of the database.
- However, with the **increasing number of database parameters** and **increasing database types**, there are limitations to DBA manually tuning parameters for all databases.



2-1. Database Parameter Tuning

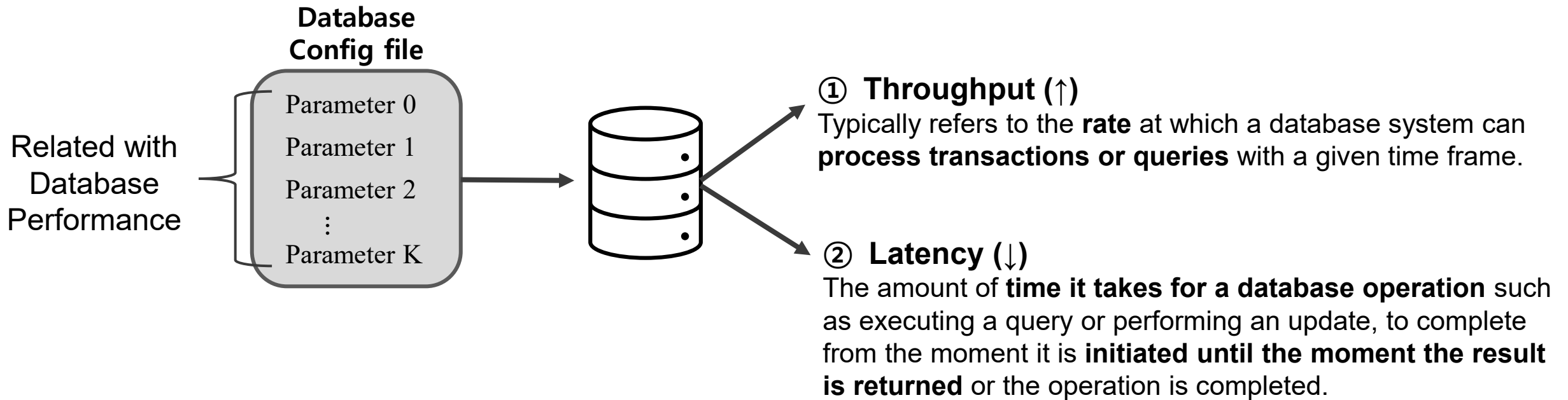
- Database Configuration Tuning Limitation
 - Also, as database versions **are updated with various parameter** configurations, posing challenges for DBA to manually adjust tuning strategies according to version changing.
 - There are so **many different workloads**, it is difficult for DBA to optimize databases for all workloads.



WORKLOAD INDEX	A	AA	B	BB	C	CC	D	DD	E	EE	F	FF
Scale Factor	5000	12000	5000	12000	5000	12000	5000	12000	5000	12000	5000	12000
ReadRecord	50	50	95	95	100	100	95	95	0	0	50	50
InsertRecord	0	0	0	0	0	0	5	5	5	5	0	0
ScanRecord	0	0	0	0	0	0	0	0	95	95	0	0
UpdateRecord	50	50	5	5	0	0	0	0	0	0	0	0
DeleteRecord	0	0	0	0	0	0	0	0	0	0	0	0
ReadModify WriteRecord	0	0	0	0	0	0	0	0	0	0	50	50

2-1. Database Parameter Tuning

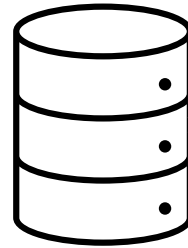
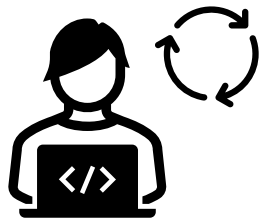
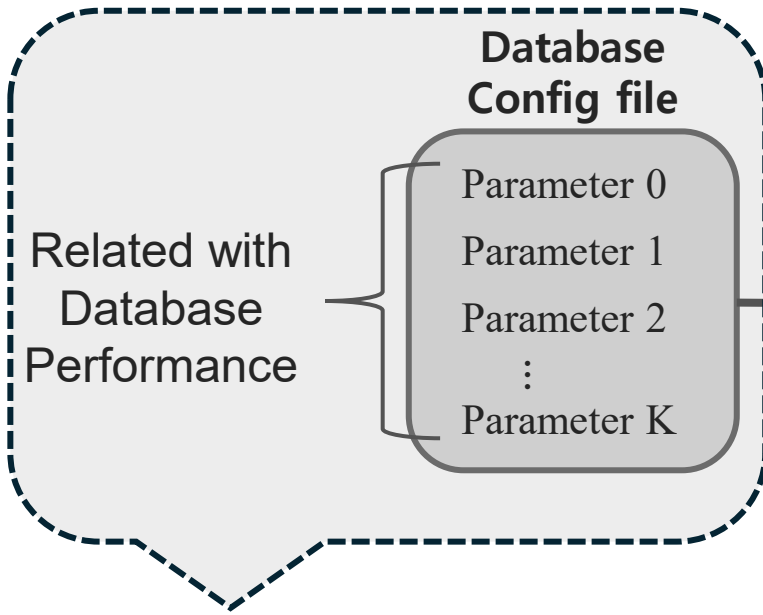
- Database Configuration Tuning Limitation



2-1. Database Parameter Tuning

- Database Configuration Tuning Limitation

①



① **Throughput (↑)**

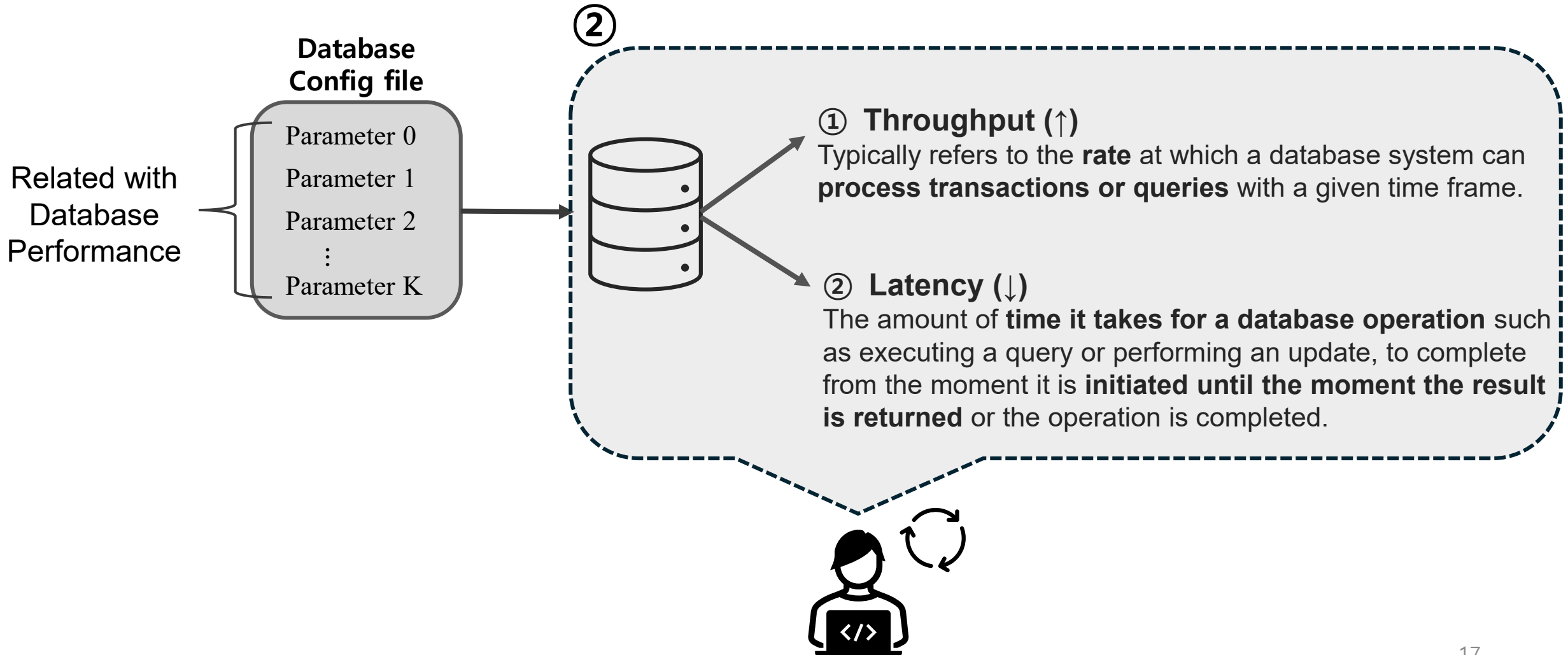
Typically refers to the **rate** at which a database system can **process transactions or queries** with a given time frame.

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.

2-1. Database Parameter Tuning

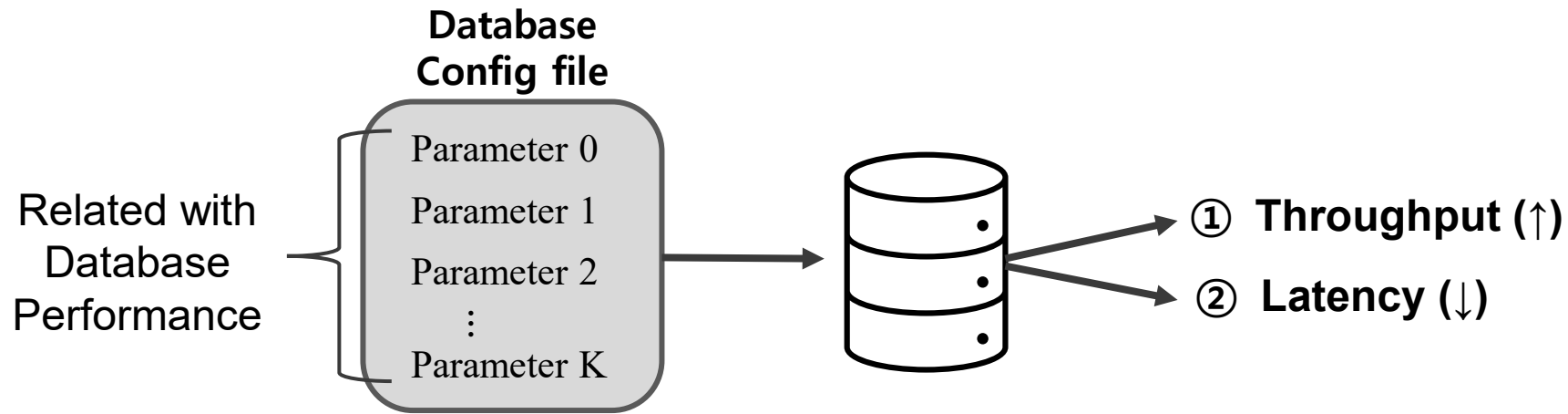
- Database Configuration Tuning Limitation



2-2. Automatic Database Configuration Tuning

Then.. HOW?

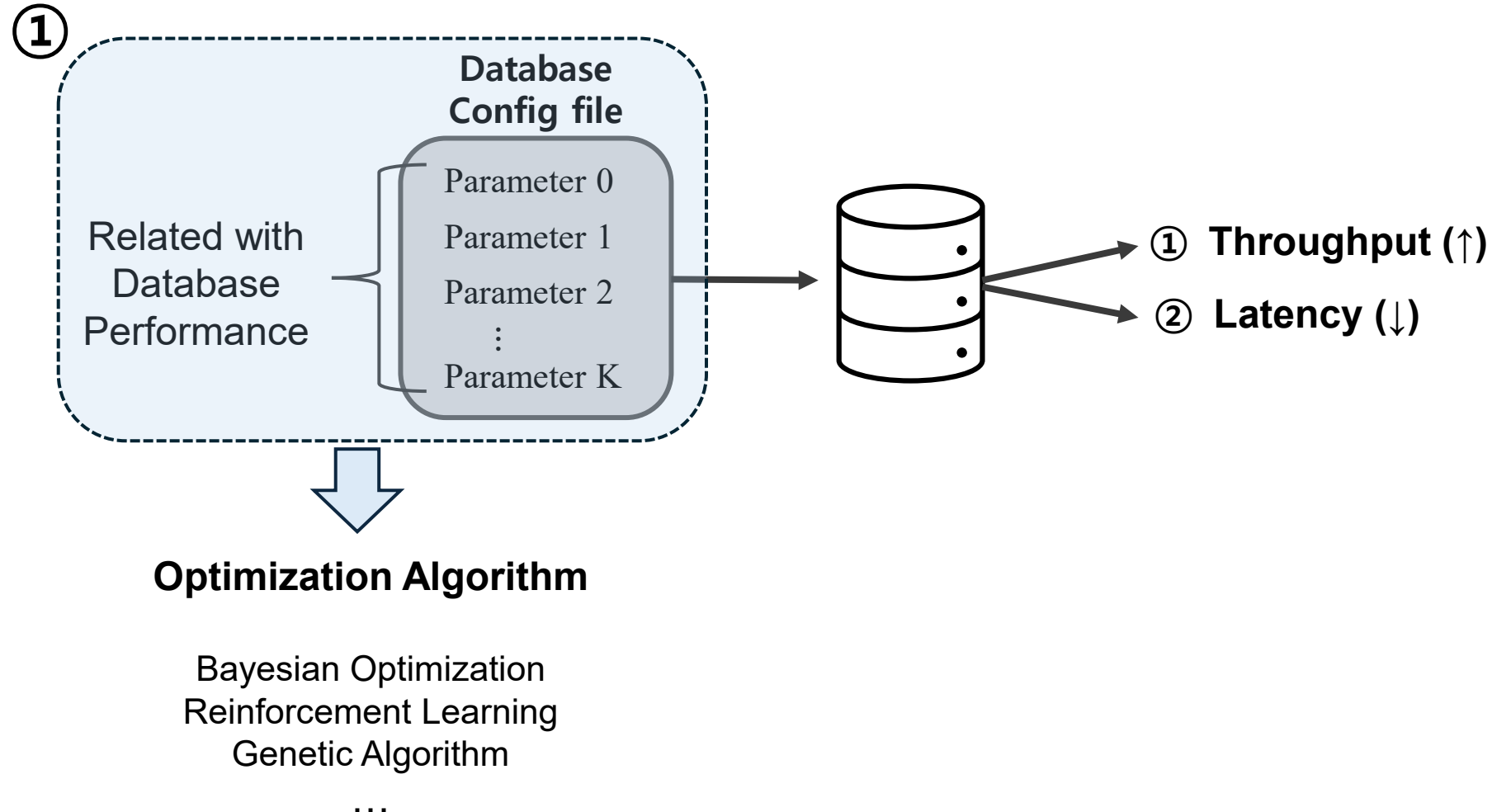
- Automatic Database Configuration Tuning !



2-2. Automatic Database Configuration Tuning

Then.. HOW?

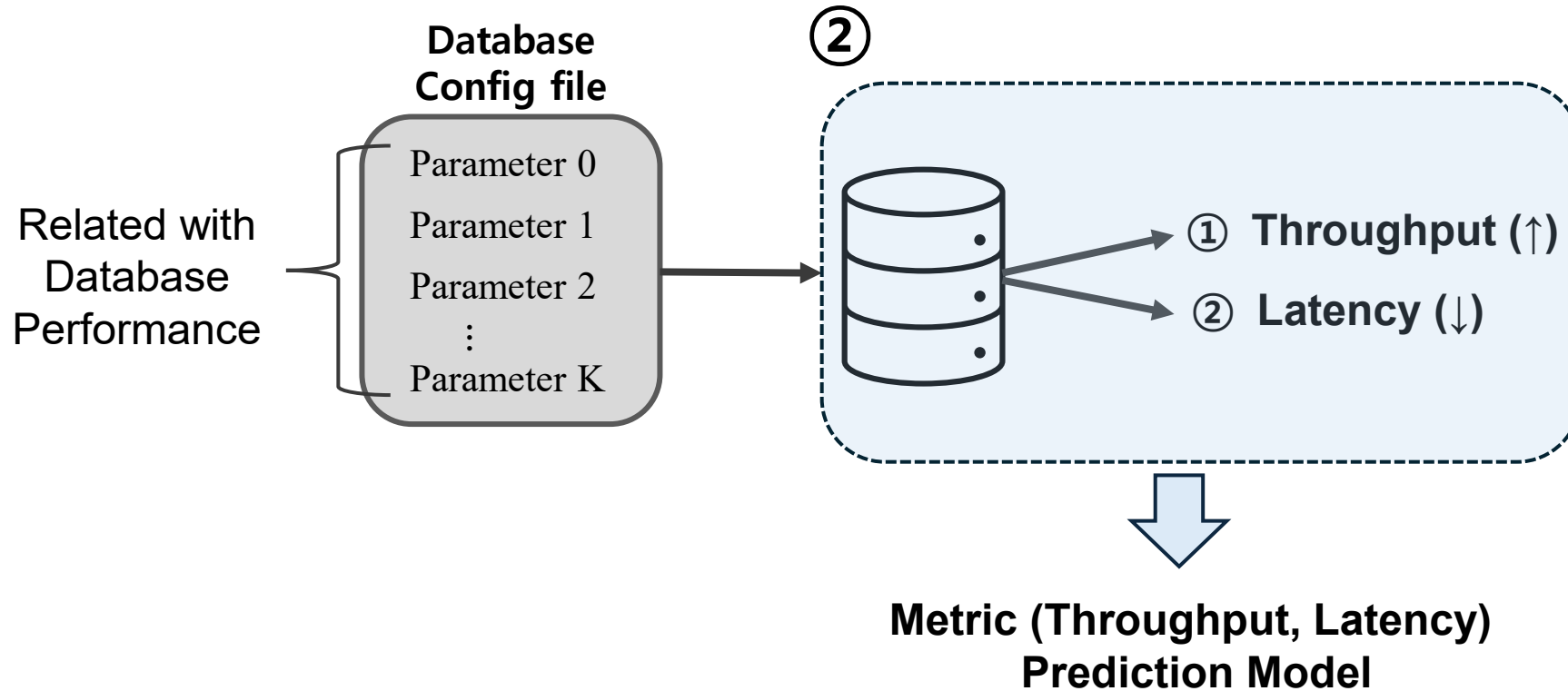
- Automatic Database Configuration Tuning !



2-2. Automatic Database Configuration Tuning

Then.. HOW?

- Automatic Database Configuration Tuning !



2-2. Automatic Database Configuration Tuning

- Automatic Database Configuration Tuning

SIGMOD'17



Automatic Database Management System Tuning Through Large-scale Machine Learning

Dana Van Aken
Carnegie Mellon University
dvanaken@cs.cmu.edu

Andrew Pavlo
Carnegie Mellon University
pavlo@cs.cmu.edu

Geoffrey J. Gordon
Carnegie Mellon University
ggordon@cs.cmu.edu

Bohan Zhang
Peking University
bohan@pku.edu.cn

ABSTRACT

Database management system (DBMS) configuration tuning is an essential aspect of any data-intensive application effort. But this is historically a difficult task because DBMSs have hundreds of configuration “knobs” that control everything in the system, such as the amount of memory to use for caches and how often data is written to storage. The problem with these knobs is that they are not standardized (i.e., two DBMSs use a different name for the same knob), not independent (i.e., changing one knob can impact others), and not universal (i.e., what works for one application may be sub-optimal for another). Worse, information about the effects of the knobs typically comes only from (expensive) experience. To overcome these challenges, we present an automated approach that leverages past experience and collects new information to tune DBMS configurations: we use a combination of supervised and unsupervised machine learning methods to (1) select the most impactful knobs, (2) map unseen database workloads to previous workloads from which we can transfer experience, and (3) recommend knob settings. We implemented our techniques in a new tool called OtterTune and tested it on three DBMSs. Our evaluation shows that OtterTune recommends configurations that are as good as or better than ones generated by existing tools or a human expert.

1. INTRODUCTION


The ability to collect, process, and analyze large amounts of data is paramount for being able to extrapolate new knowledge in business and scientific domains [35, 25]. DBMSs are the critical component of data-intensive (“Big Data”) applications [46]. The performance of these systems is often measured in metrics such as throughput (e.g., how fast it can collect new data) and latency (e.g., how fast it can respond to a request). Achieving good performance in DBMSs is non-trivial as they are complex systems with many tunable options that control nearly all aspects of their runtime operation [24]. Such configuration knobs allow the database administrator (DBA) to control various aspects of the DBMS’s runtime behavior. For example, they can set how much memory the system allocates for data caching versus the transaction log buffer. Modern DBMSs are notorious for having

many configuration knobs [22, 47, 36]. Part of what makes DBMSs so enigmatic is that their performance and scalability are highly dependent on their configurations. Further exacerbating this problem is that the default configurations of these knobs are notoriously bad. As an example, the default MySQL configuration in 2016 assumes that it is deployed on a machine that only has 160 MB of RAM [1]. Given this, many organizations resort to hiring expensive experts to configure the system’s knobs for the expected workload. But as databases and applications grow in both size and complexity, optimizing a DBMS to meet the needs of an application has surpassed the abilities of humans [11]. This is because the correct configuration of a DBMS is highly dependent on a number of factors that are beyond what humans can reason about. Previous attempts at automatic DBMS configuration tools have certain deficiencies that make them inadequate for general purpose database applications. Many of these tuning tools were created by vendors, and thus they only support that particular company’s DBMS [22, 33, 37]. The small number of tuning tools that do support multiple DBMSs still require manual steps, such as having the DBA (1) deploy a second copy of the database [24], (2) map dependencies between knobs [49], or (3) guide the training process [58]. All of these tools also examine each DBMS deployment independently and thus are unable to apply knowledge gained from previous tuning efforts. This is inefficient because each tuning effort can take a long time and use a lot of resources. In this paper, we present a technique to reuse training data gathered from previous sessions to tune new DBMS deployments. The crux of our approach is to train machine learning (ML) models from measurements collected from these previous tunings, and use the models to (1) select the most important knobs, (2) map previously unseen database workloads to known workloads, so that we can transfer previous experience, and (3) recommend knob settings that improve a target objective (e.g., latency, throughput). Reusing past experience reduces the amount of time and resources it takes to tune a DBMS for a new application. To evaluate our work, we implemented our techniques using Google TensorFlow [50] and Python’s scikit-learn [39] in a tuning tool, called OtterTune, and performed experiments for two OLTP DBMSs (MySQL, Postgres) and one OLAP DBMS (Vector). Our results show that OtterTune produces a DBMS configuration for these workloads that achieves 58–94% lower latency compared to their default settings or configurations generated by other tuning advisors. We also show that OtterTune generates configurations in under 60 min that are within 94% of ones created by expert DBAs. The remainder of this paper is organized as follows. Sect. 2 begins with a discussion of the challenges in database tuning. We then provide an overview of our approach in Sect. 3, followed by a description of our techniques for collecting DBMS metrics in Sect. 4.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGMOD’17, May 14 – 19, 2017, Chicago, IL, USA
© 2017 Copyright held by the owner(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4197-4/17/05...\$15.00
DOI: <http://dx.doi.org/10.1145/3035918.3064029>





Product Tour Blog Plans and Pricing Contact Resources ▾ Login Get Started Product Tour

MySQL + PostgreSQL optimization for AWS Aurora + RDS

4x faster. 50% cheaper.

OtterTune uses AI to optimize your Amazon RDS and Aurora database performance and reduces AWS costs.

Get Started Learn More

✓ No credit card required ✓ 30-day free trial

Dashboard: Fleet Level

Overall Health

93

Database: 95%

Resource: 97%

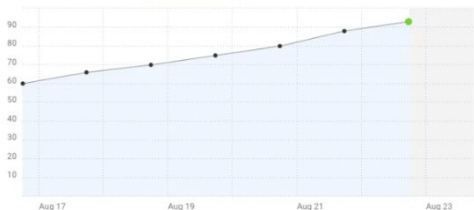
Table: 90%


Index: 93%

Query: 90%

Action Items

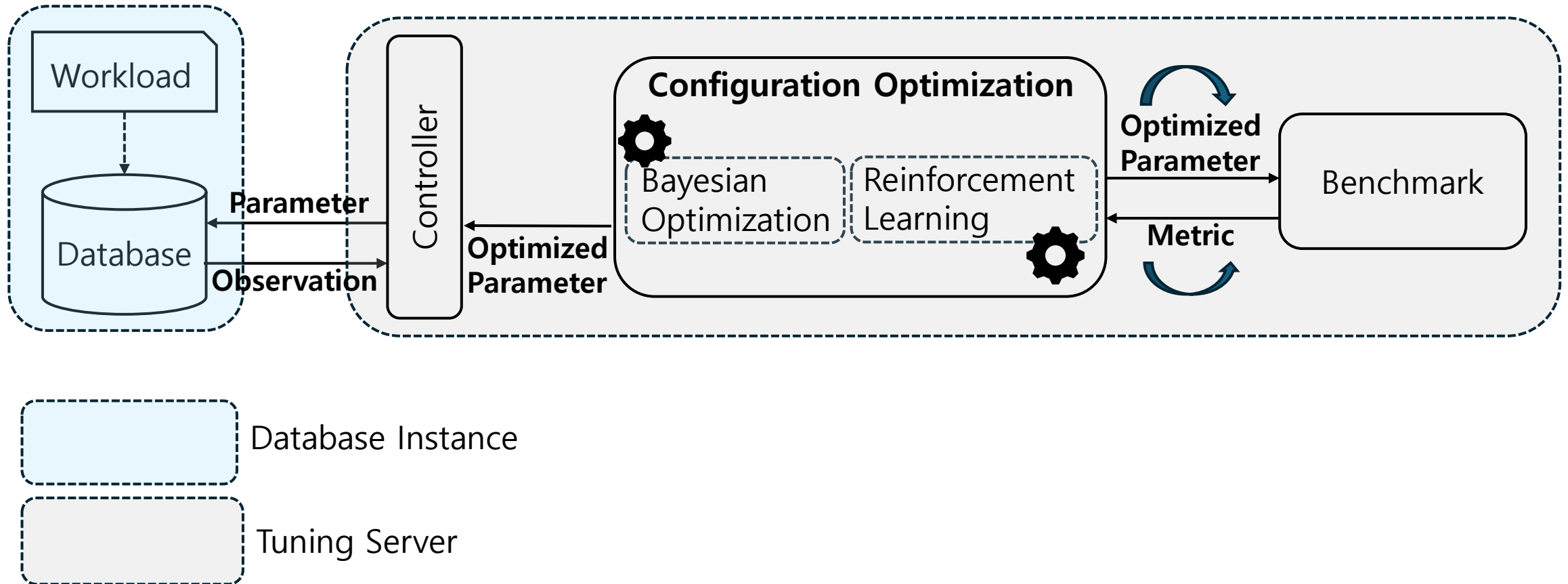
Score History





2-2. Automatic Database Configuration Tuning

- Automatic Database Configuration Tuning



2-2. Automatic Database Configuration Tuning

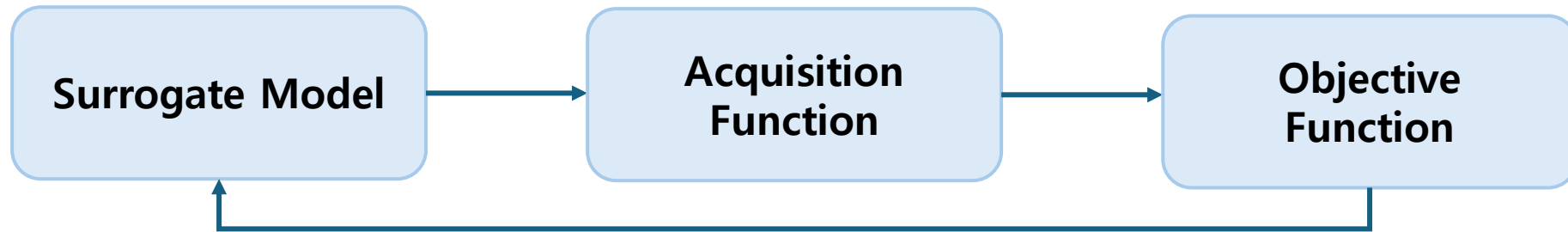
- **Bayesian Optimization (BO)**

- **Bayesian Optimization (BO)** is a popular optimization framework designed to **solve black-box problems** with expensive evaluation costs.
- The main advantage of BO is that it estimates the uncertainty of unseen parameters (=configuration) and **balances exploration and exploitation** when selecting the next parameters to evaluate.
- The main problem for BO to solve is an optimization problem for an unknown object function f (black box algorithm), and we hope to find a global **maximizer of that function**:

$$\max_{x \in A} f(x)$$

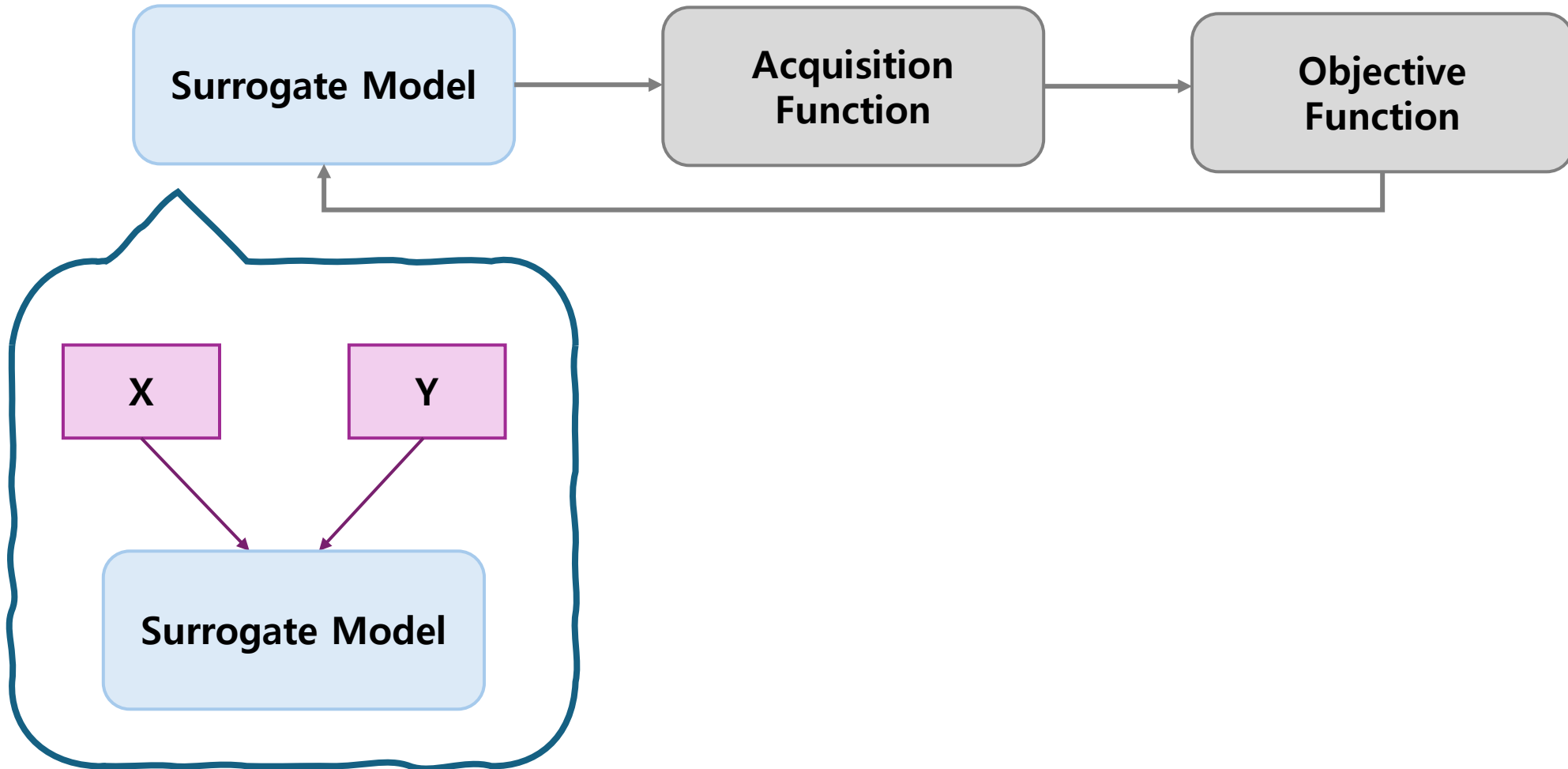
2-2. Automatic Database Configuration Tuning

- Bayesian Optimization (BO)



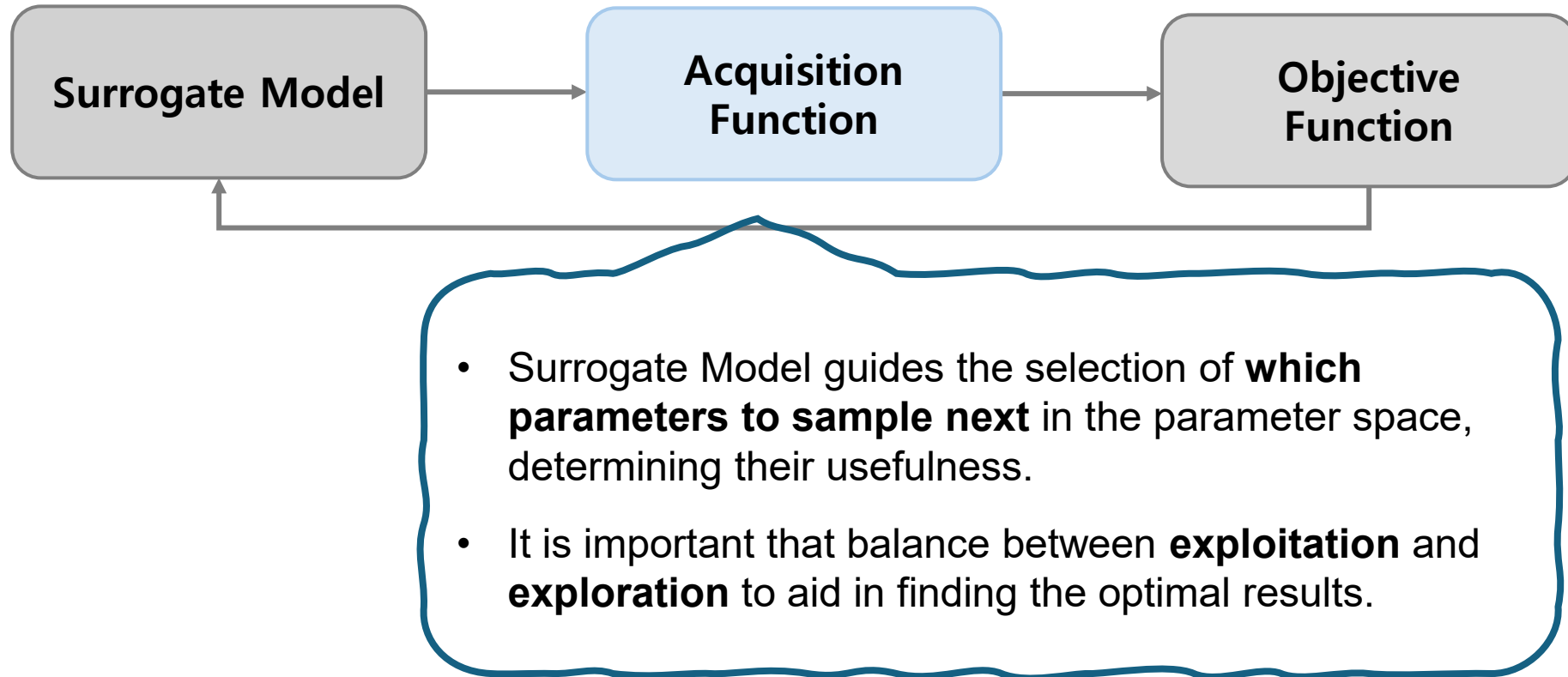
2-2. Automatic Database Configuration Tuning

- Bayesian Optimization (BO)



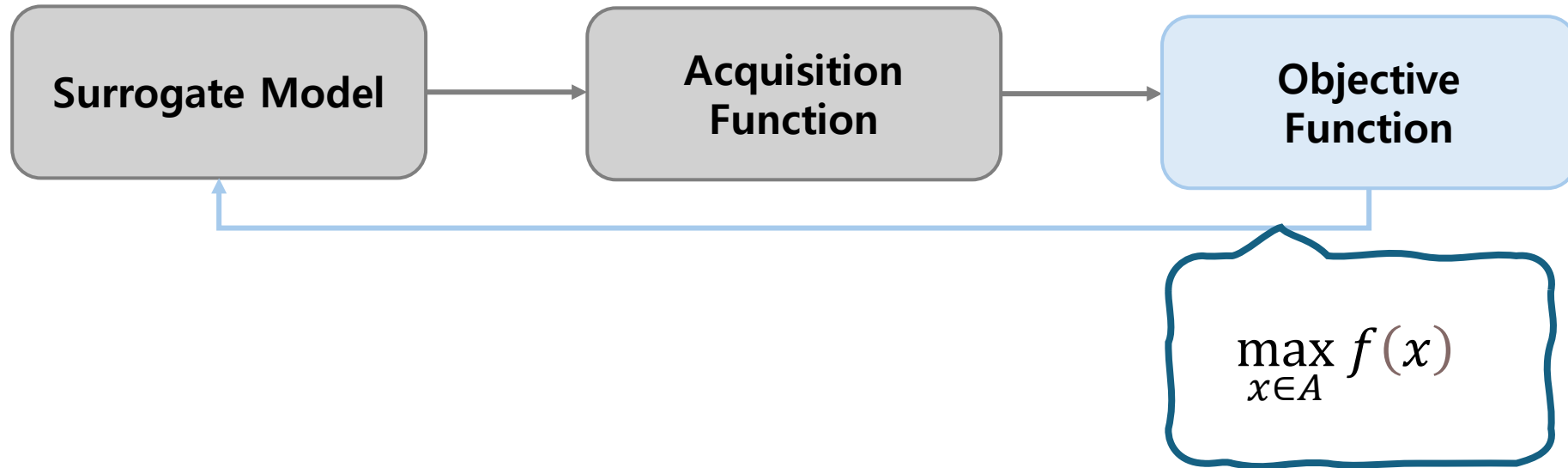
2-2. Automatic Database Configuration Tuning

- Bayesian Optimization (BO)



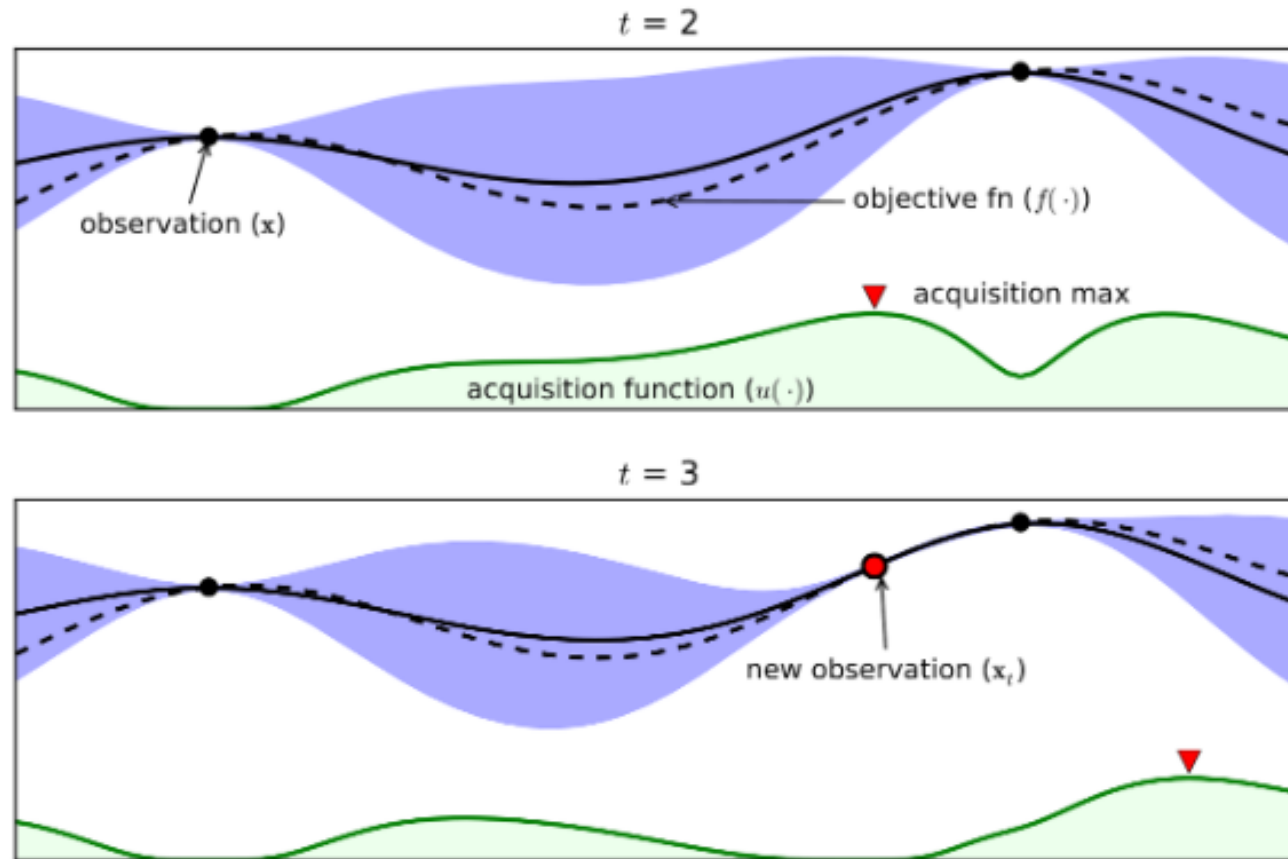
2-2. Automatic Database Configuration Tuning

- Bayesian Optimization (BO)



2-2. Automatic Database Configuration Tuning

- Bayesian Optimization (BO)



3

Related Papers

- Limitation of the existing Database Tuning Method
- How solve the limitations
- Architecture of the papers

3. Related Papers

- **Automatic Database Management System Tuning Through Large-scale Machine Learning**

SIGMOD'17

① Limitation of the existing Database Tuning Method

- **Dependencies**

DBMS tuning guides strongly suggest that a DBA only change one knob at a time.
But, not entirely helpful because changing one knob may affect the benefits of another.

- **Continuous Settings**

There are many possible settings for knobs, and the differences in performance from one setting to the next could be irregular.

- **Non-Reusable Configurations**

The effort that a DBA spends on tuning one DBMS does not make tuning the next one any easier.

- **Tuning Complexity**

The number of DBMS knobs is always increasing as new versions and features are released.

3. Related Papers

- **Automatic Database Management System Tuning Through Large-scale Machine Learning**

SIGMOD'17

② How solve the limitations

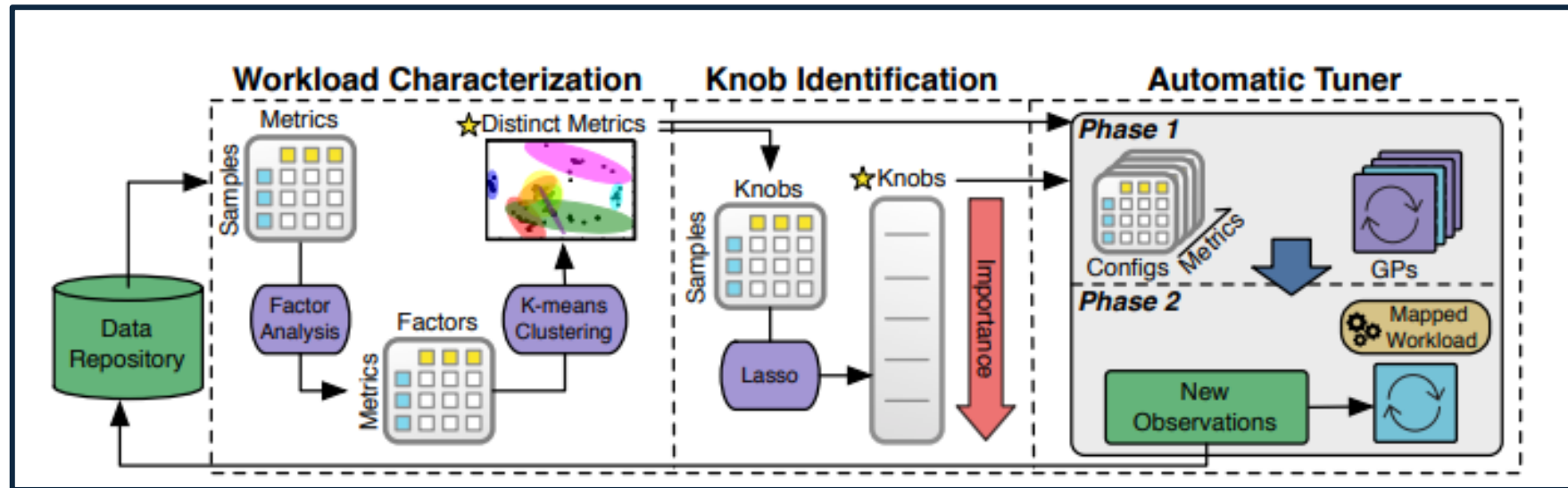
- OtterTune present a technique to **reuse training data** gathered from previous sessions to tune new DBMS deployments.
- Train machine learning (ML) models from measurements **collected from previous tunings**.
 - 1) Select the most important knobs
 - 2) Map previously unseen database workloads to known workloads
 - 3) Recommend knob settings that improve a target objective (throughput, latency)
- OtterTune is a tuning service that works with **any DBMS**.
- It maintains a repository of data collected from previous tuning sessions, and uses this data to build models of how the DBMS responds to different parameter configurations.

3. Related Papers

- Automatic Database Management System Tuning Through Large-scale Machine Learning

SIGMOD'17

③ Architecture

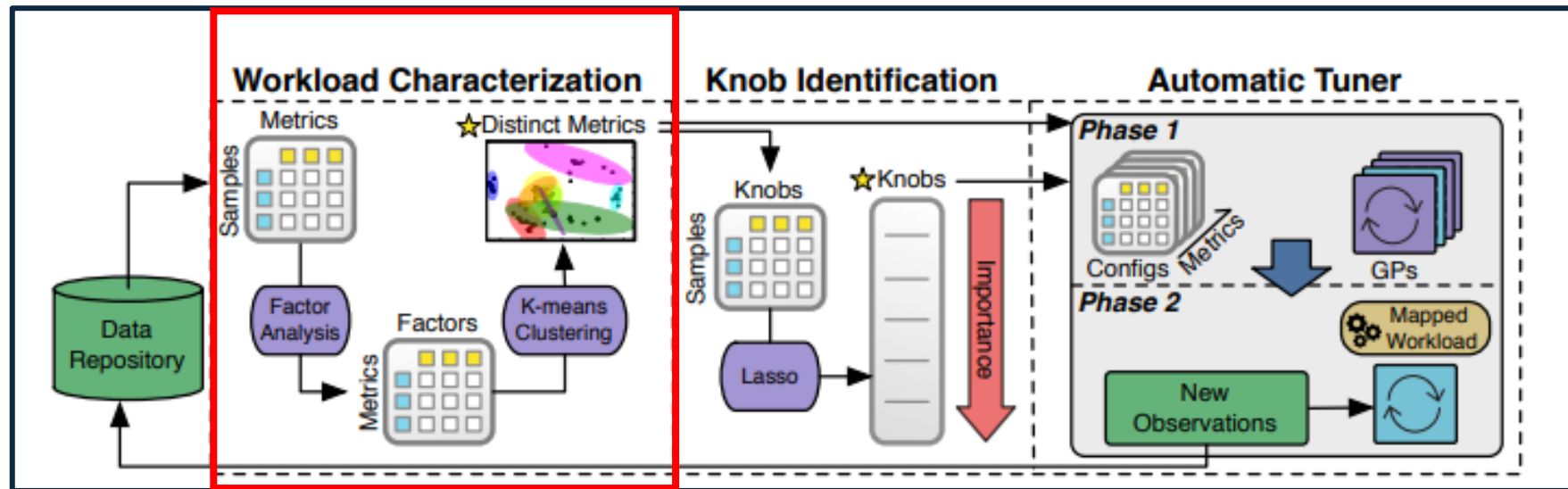


3. Related Papers

- Automatic Database Management System Tuning Through Large-scale Machine Learning

SIGMOD'17

③ Architecture

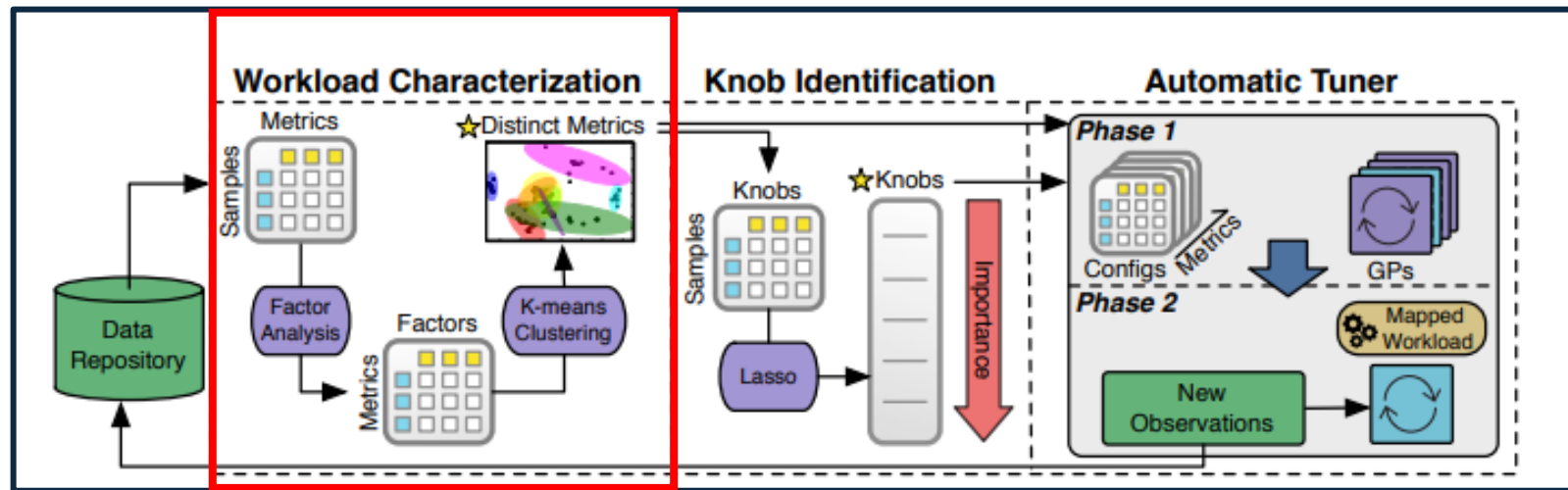


3. Related Papers

- Automatic Database Management System Tuning Through Large-scale Machine Learning

SIGMOD'17

③ Architecture



- **Internal metrics**

indicators used to measure the **internal operation and performance** of the database system.

- **External metrics**

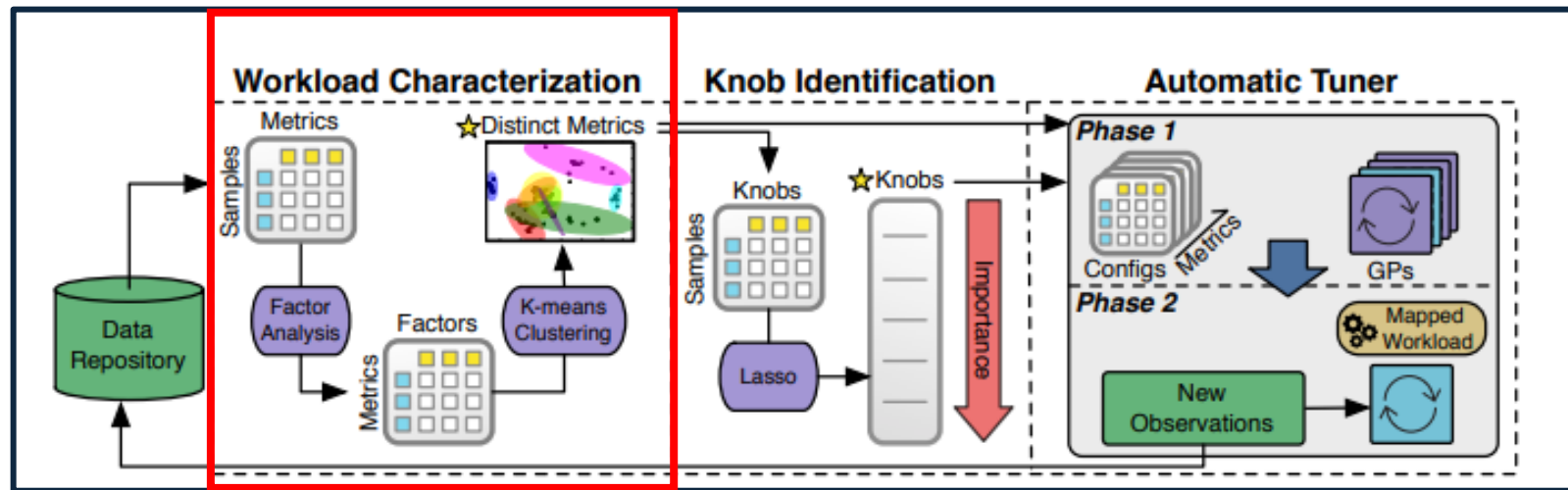
indicators measured **outside the database system**. Primarily used to measure the monitor the behavior and performance of the database from the **perspective of users or applications**.

3. Related Papers

- Automatic Database Management System Tuning Through Large-scale Machine Learning

SIGMOD'17

③ Architecture



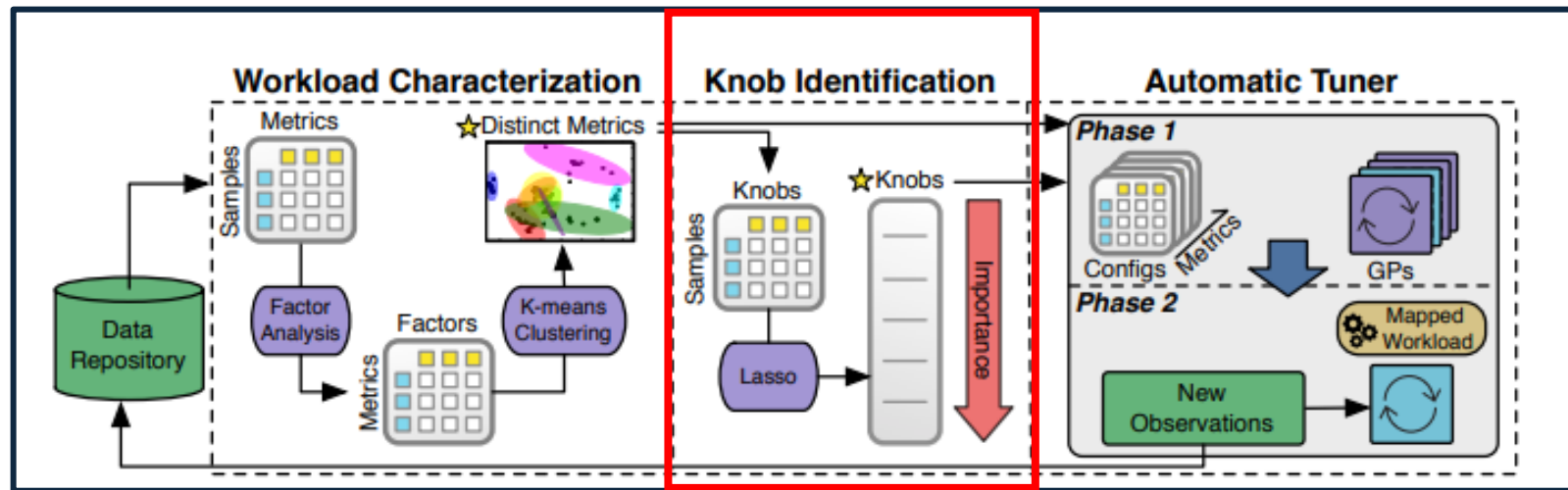
- Internal metrics encompass metrics that are **not directly related to database performance**.
- **Factor Analysis** and **K-means Clustering** are used to **remove unnecessary internal metrics**.
- This method can **reduce the search space** of machine learning algorithms, thereby **accelerating the overall tuning process**.

3. Related Papers

- Automatic Database Management System Tuning Through Large-scale Machine Learning

SIGMOD'17

③ Architecture



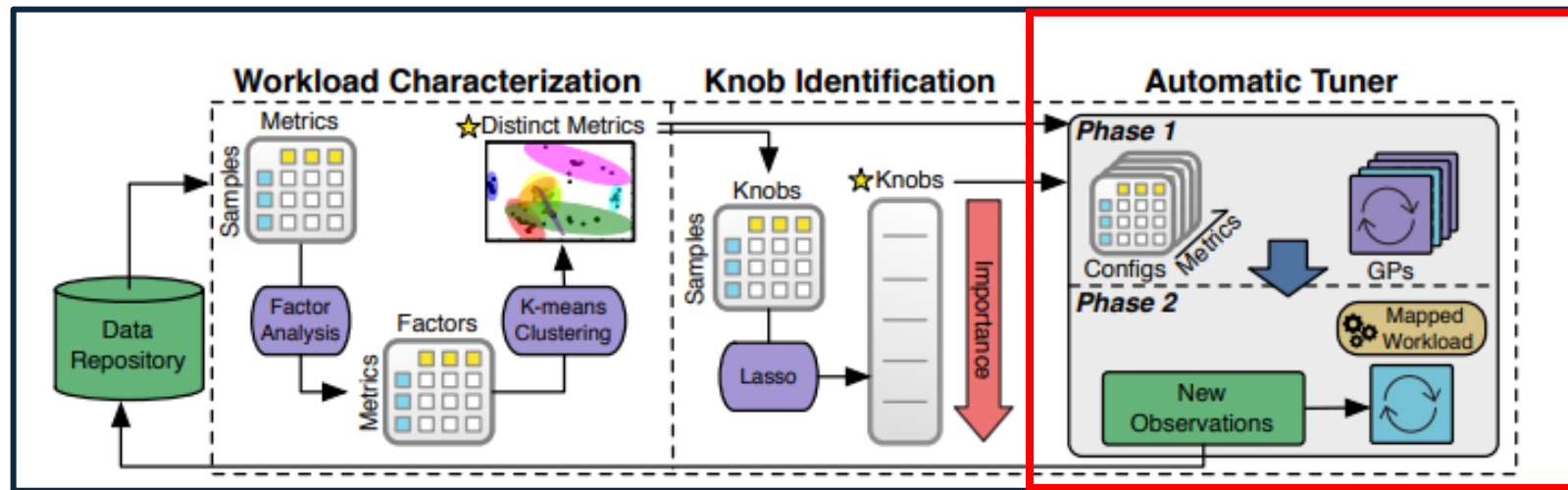
- OtterTune uses a popular feature selection technique for linear regression, called **Lasso**.
- OtterTune must decide how many of these knobs to use in its recommendations.

3. Related Papers

- Automatic Database Management System Tuning Through Large-scale Machine Learning

SIGMOD'17

③ Architecture



- The goal of this step is to **match the target DBMS's workload** with the most similar workload in its repository based on the performance measurements for the selected group of metrics.
- In the next step, OtterTune uses **Gaussian Process regression** to **recommend configurations** that it believes will improve the target metric.

3. Related Papers

- **An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning**

SIGMOD'19

① Limitation of the existing Database Tuning Method

- OtterTune utilize **machine-learning** techniques to collect, process and analyze knobs and recommend possible settings by learning DBA's experiences from historical data.
- However,
 - 1) It cannot optimize the overall performance in an **end-to-end**.
 - 2) OtterTune rely on **large-scale high-quality training samples**.
 - 3) Cannot optimize the knob settings in **high-dimensional continuous space**.

3. Related Papers

- **An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning**

SIGMOD'19

② How solve the limitations

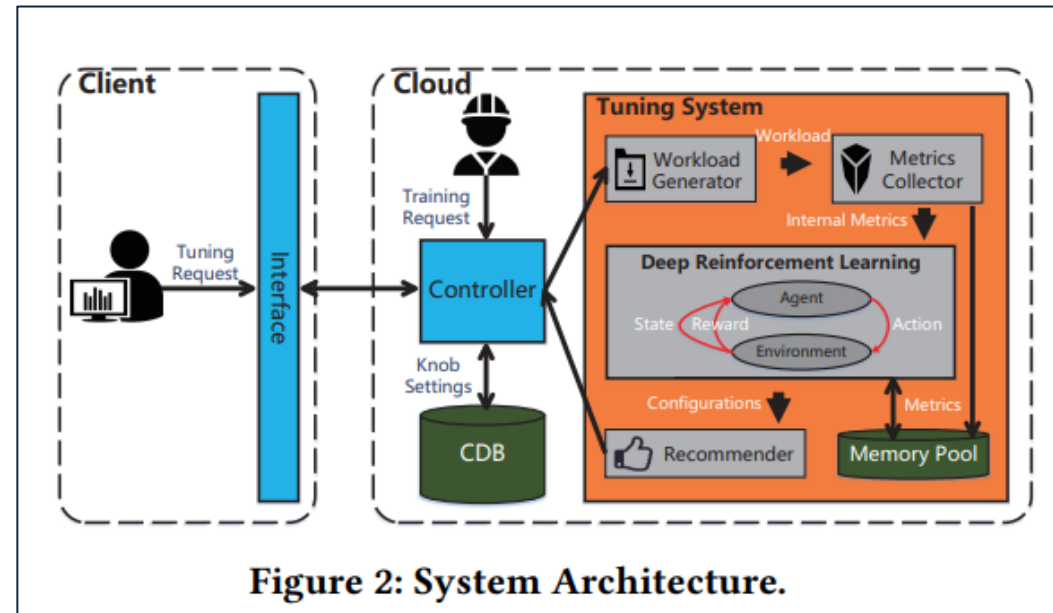
- End-to-end automatic database tuning system that **uses deep RL** to learn and recommend configurations for databases.
- Adopt a **try-and-error manner in RL** to learn the best knob settings with a **limited number of samples**.
- CDBTune utilizes the **deep deterministic policy gradient** method to find the optimal configurations in **high-dimensional continuous space**.

3. Related Papers

- An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning

SIGMOD'19

③ Architecture



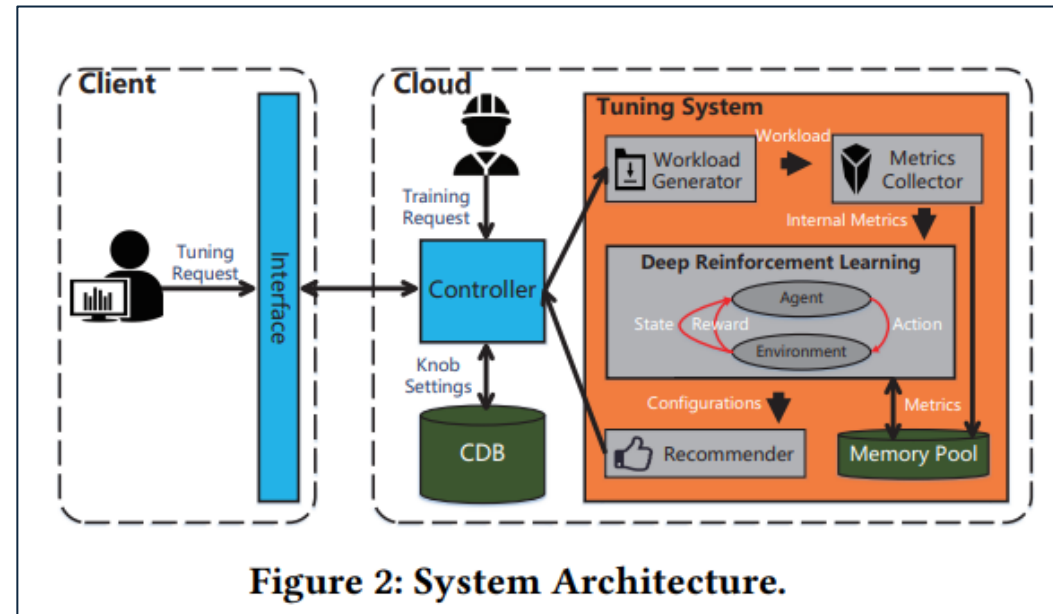
- The workload generator receives tuning requests through the controller, then executes a standard workload.
- The metric collector gathers and preprocesses metric data from the running cloud database, storing the processed data in a memory pool to be used as training samples for reinforcement learning.

3. Related Papers

- An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning

SIGMOD'19

③ Architecture



- The recommender, connecting the client and the cloud, requests parameter modifications from the controller when the reinforcement learning model recommends parameters.

3. Related Papers

- An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning

SIGMOD'19

③ Architecture

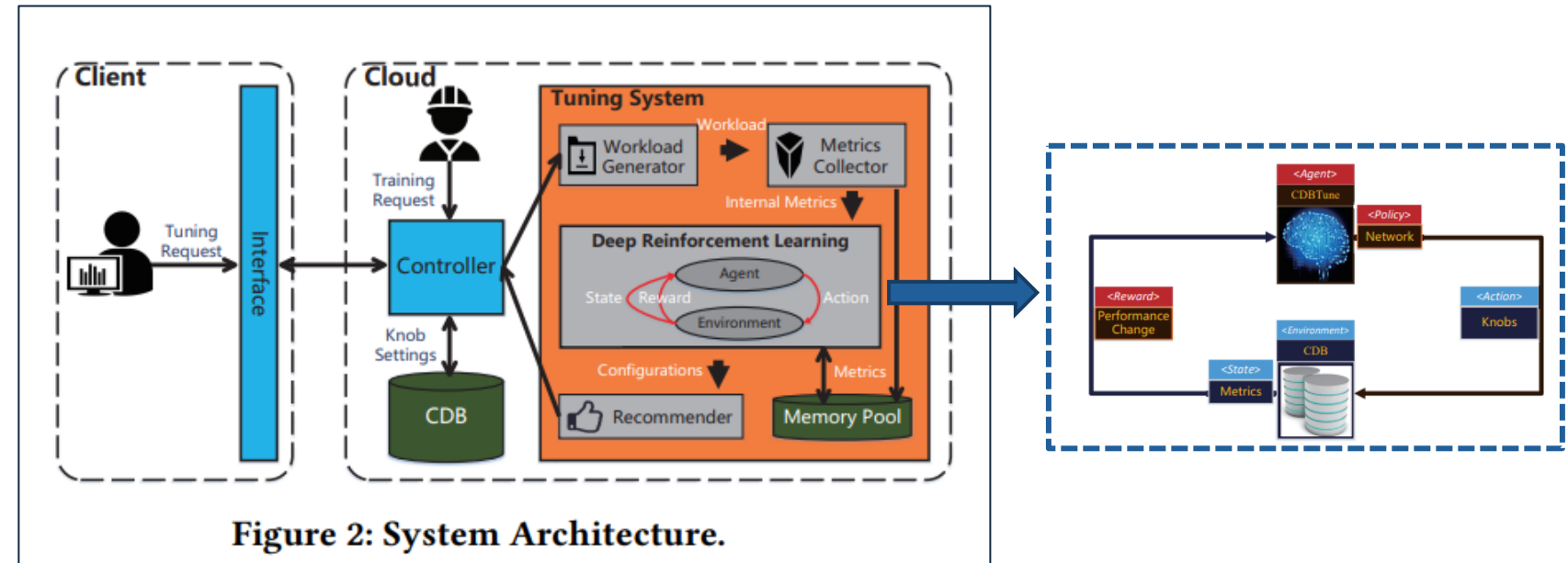


Figure 2: System Architecture.

- The reinforcement learning network employs a combination of DQN (Deep Q Network) and Actor-Critic, specifically the **DDPG** (Deep Deterministic Policy Gradient) algorithm, to conduct parameter tuning.
- The DDPG algorithm has the advantage of obtaining action-value estimates in **continuous spaces**. Furthermore, it can learn policies based on **high-dimensional states and actions**, especially internal performance metrics and parameters.

3. Related Papers

- **Facilitating Database Tuning with Hyper-Parameter Optimization: A Comprehensive Experimental Evaluation**

VLDB'23

① Limitation of the existing Database Tuning Method

- Recently, using automatic configuration tuning to improve the performance of modern database management systems (DBMSs).
 - However, it remains a challenge to select the best solution for database configuration tuning, considering the large body of algorithm choices.
- 1) Missing comparative evaluations of intra-algorithms in different modules.
 - 2) Absence of analysis for high-dimensional and heterogeneous scenarios.
 - 3) Limited solution comparison without a broader view.

3. Related Papers

- **Facilitating Database Tuning with Hyper-Parameter Optimization: A Comprehensive Experimental Evaluation**

VLDB'23

② How solve the limitations

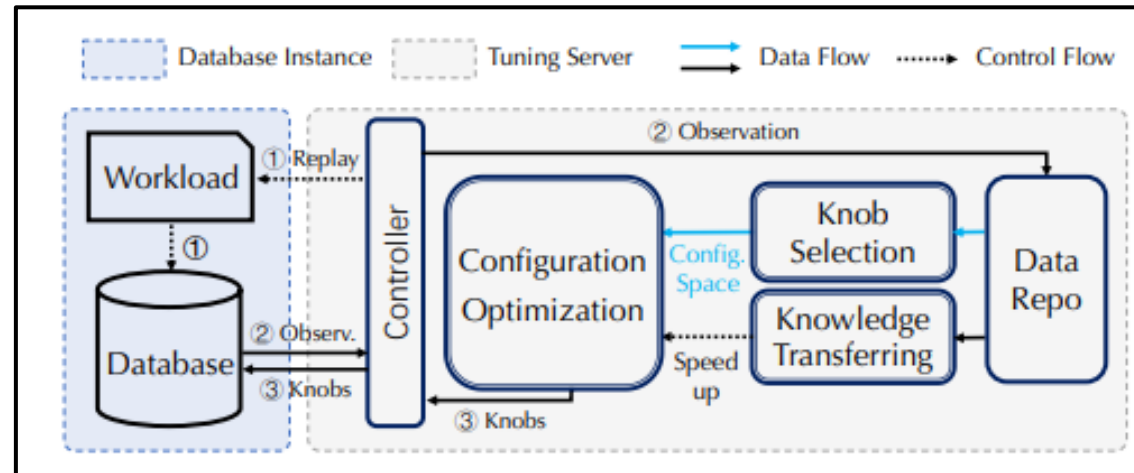
- 1) We present a unified pipeline with three key modules and evaluate the fine-grained intra-algorithms.
 - How to determine tuning parameters?
 - Which optimizer is the winner?
 - Can we transfer knowledge to speed up the target running task?
- 2) We construct extensive scenarios to benchmark multiple optimizers.
- 3) Evaluate other advanced approaches in the HPO field under the same database tuning setting.
 - Many recent database parameter tuning papers approaches in the HPO field could be borrowed to alleviate the challenges in database tuning.
 - Demonstrate that database practitioners can borrow strength from the HPO approached in an out-of-the-box manner.

3. Related Papers

- Facilitating Database Tuning with Hyper-Parameter Optimization: A Comprehensive Experimental Evaluation

VLDB'23

③ Architecture



4. *Future Work*

- New Approach about the Automatic Database Configuration Tuning.
- How to optimize with using all parameters rather than optimizing the Top – K parameters? (subspace)
- How can we apply a method that can produce good results even with a small samples?
- How to tune for various databases that are constantly evolving?
- How to tune in real-world? (continuously changing workloads)



연세대학교
YONSEI UNIVERSITY

Thank You for Listening!