ChimeraTL: Transfer Learning in DBMS with Fewer Samples

Tatsuhiro Nakamori[3], Shohei Matsuura[1], Takashi Miyazaki[1],
Sho Nakazono[1], Taiki Sato[1], Takashi Hoshino[2], Hideyuki Kawashima[3]
LY Corporation[1], *Cybozu Labs*[2], *Keio University*[3]

***Abstract*—In the field of database management systems (DBMS), it is essential to build a performance prediction model with less data from the target environment, motivating the application of transfer learning. While some parameters in DBMS have similar effects on performance across different hardware environments, others can have varying effects that depend on underlying hardware limitations. Previous studies do not leverage this information to improve the transfer learning. We propose ChimeraTL, a novel method that accounts for different parameter types to enhance transfer learning. Our experiments demonstrate that ChimeraTL needs only 50% of samples that state-of-the-art methods require to minimize the prediction error to under 10%.**

***Index Terms*—Transfer Learning, DBMS**

I. Introduction

A. Motivation

Database management systems (DBMS) are an integral component of applications. One aspect of Database management systems (DBMS) that makes it essential in many applications is its adaptivity. DBMS has many configuration parameters (e.g. 190 in MySQL [1] and 170 in PostgreSQL [2], [3]) that users can adjust to optimize to specific settings and meet the user requirements. Recent studies have emphasized building performance prediction models instead of relying on heuristics to find optimal configurations [4].

While existing studies target learning performance prediction models for unseen workloads [3]–[7], they assume that the model is learned from the same hardware environment where it is used. In reality, DBMS users can test the performance of the DBMS with various configurations in a testing environment and build a performance prediction model to estimate the performance of the DBMS in the production environment [8]. However, since a testing environment is often a scaled-down version of a production environment, the model learned from the testing environment may not be accurate in predicting the performance of the production environment. What developers can do instead is to extract the knowledge of the testing environment that can be reused to build an accurate model for the production environment.

Such an approach to reduce the learning cost by utilizing the data from a different but related environment is an example of *transfer learning* [9], [10]. Our motivation is to apply transfer learning techniques to learn a DBMS performance prediction model using fewer samples from the target environment.

This paper is based on results obtained from "Research and Development Project of the Enhanced Infrastructures for Post-5G Information and Communication Systems" (JPNP20017) and (JPNP16007) commissioned by the New Energy and Industrial Technology Development Organization (NEDO) and JSPS KAKENHI Grant Number 22H03596.

B. Problem

Transfer learning for configurable software systems has been discussed in previous literature [9]–[12]. While previous transfer learning approaches [9]–[11] are effective in specific settings, they must consider the following properties of DBMS parameters to make the transfer of knowledge more effective.

1. Parameters are not binary: Previous studies focused their evaluation on cases in which the configuration parameters were *binary*, i.e., the parameter is either on or off [9], [11], [12], but the parameters in DBMS often have a wide range of values (e.g. buffer pool size in MySQL [13]). Since there are numerous parameters with wide value ranges, existing transfer learning approaches would require more samples than desired to learn an accurate model for the vast configuration space.

2. Parameter effects can be machine specific: Although studies have shown that the performance functions of many parameters between two environments have a linear relationship [11], [12], employing a strategy based on the assumption that all parameters follow this pattern can cause a significant bias in the model. Fig. 1 shows the relationship between the throughput of two environments for varying configurations in LineairDB [14]. It is evident that the relationship is not linear because two machines have different saturation points when the number of threads is varied. This example shows that some parameters are machine specific, having different effects on performance depending on the hardware limitations.

To build an accurate performance model of DBMS with fewer samples from the target environment, we need to consider the above properties by exploiting the knowledge of parameters that are consistent in their performance impact across diverse hardware environments while avoiding the inaccuracies caused by transferring the knowledge of parameters that have machine-specific effects. Nonetheless, to our knowledge, no existing transfer learning technique considers such a design.

C. Contribution

We present ChimeraTL, a transfer learning pipeline to build an accurate performance model of DBMS with fewer samples from the target environment. It combines three existing methods: ModelShift [11], DataReuseTL [10], and L2S [9], in a way that maximizes their advantages and minimizes their disadvantages. It consists of three procedures: (1) parameter selection [9] and separation, (2) linear transformation learning [10], [11], and (3) machine-specific parameter learning.

First, ChimeraTL starts by selecting parameters that have significant impact on performance and separating them into *universal parameters* (parameters that have similar performance effects across different environments) and *machine-specific parameters* (parameters that have different performance effects depending on the hardware limitations). While parameter selection has been a common preprocessing technique in DBMS performance prediction models [1], [4], to the best of our knowledge, none of the previous studies have considered classifying different types of parameters. Previous methods have assumed that all parameters have same behavior in different environments. This assumption causes a negative transfer to the model learning in the target environment. Parameter separation in ChimeraTL is a novel technique that allows the integration of ModelShift, DataReuseTL, and L2S while minimizing the negative transfer that may occur in each method.

After identifying the universal parameters, ChimeraTL samples data of the universal parameters from the target environment. Using the sampled target data and the source data of the same parameters, ChimeraTL learns a linear transformation between the two environments. ChimeraTL then linearly transforms the source data of universal parameters and uses them as additional training data to build a performance prediction model for the target environment. By linearly transforming the source data, ChimeraTL can reuse the knowledge of the source environment while avoiding the bias caused by the difference between the source and target environments. Once there is enough data on the universal parameters, ChimeraTL prioritizes sampling the data of machine-specific parameters from the target environment.

Compared with learning the model from scratch in the target environment, ChimeraTL minimizes the prediction error to less than 10% using 70% fewer samples. None of the previously noted state-of-the-art transfer learning techniques [9]–[11] can achieve this level of accuracy with the same number of samples.

II. Preliminaries

A. Objective

The goal of transfer learning is to reduce the learning cost in a target environment by utilizing the data from a different but related environment [9], [10]. In this paper, we regard the number of samples from the target environment to be the indicator of the cost.

We define a sample as a pair of DBMS parameter settings and the corresponding measure of DBMS performance. In order to collect one sample, we set the parameters to desired values, run a workload for the DBMS to process, and measure the performance during the workload. The duration of one sampling iteration depends on the workload. Previous study has taken the average performance over 5 minutes as one sample [4]. By using this sample as a training data, performance prediction models can learn the relationship between the parameters and the performance.

B. Transfer Learning Methods

In this paper, we compare the transfer learning methods for the performance prediction of configurable software systems.

ModelShift [11] is a type of transfer learning where the performance model learned from the source environment is linearly transformed to predict the performance of the target environment as in Fig. 1. The previous study has shown that ModelShift requires less than 10 samples from the target environment to learn appropriate linear transformation coefficients [11].

DataReuseTL [10] uses the source environment data and data samples from the target environment to build a prediction model. DataReuseTL works well even with a few samples from the target environment when the source and target environments have minor differences.

L2S [9], [12] identifies important parameters likely to be shared between the source and target environments and samples the data of those parameters to build a model for the target environment. Because this method does not use the source environment data for model construction, there are no concerns about the bias caused by the difference between the source and target environments.

C. Universal and Machine-specific Parameters

Parameters in DBMS (and other configurable software systems) can be categorized into two types: *universal* and *machine-specific*. Universal parameters affect the performance of a DBMS similarly in different environments. Formally, a parameter $\boldsymbol{p_i}$ is universal if it satisfies the following equation:

$$f_{tgt}(\boldsymbol{p_i}) = \beta \times f_{src}(\boldsymbol{p_i}) + \beta_0 \quad (1)$$

where $f_{tgt}(\boldsymbol{p_i})$ represents the performance function in a target environment, and $\beta \times f_{src}(\boldsymbol{p_i}) + \beta_0$ denotes a linear transformation of the performance function in a source environment. Fig. 2a shows an example of a universal parameter in LineairDB. The figure shows that the performance function of the parameter has similar shapes in different environments and that the performance in one environment can be approximated by linearly transforming the performance in another. Considering that it takes less than 10 samples from the target environment to learn the linear transformation [11], the data of universal parameters from the source environment should be exploited to build a model using fewer samples.

The remaining parameters are machine-specific parameters that behave differently in different environments. Fig. 2b shows an example of such a parameter. The three environments in the figure have varying trends for the same parameter value. Using the data of machine-specific parameters from the source environment to build a model for the target environment is problematic because the data from the source may not reflect the actual relationship between the parameter values and performance in the target.

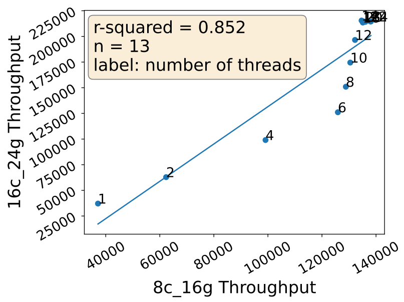
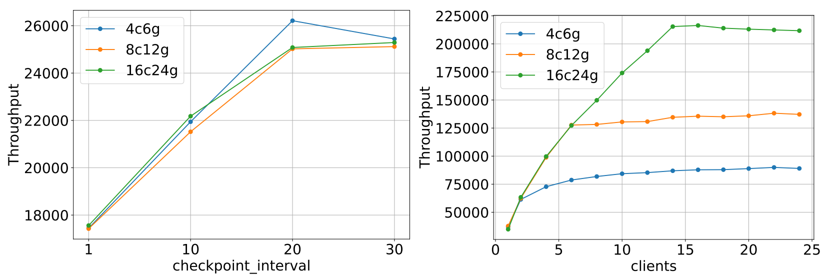


Fig. 1: **Association of LineairDB [14] throughput between two environments and a linear regression model** – Each point shows a parameter value. `8c_16g` represents `8 core, 16 GB` RAM machine.



(a) Universal parameter (b) Machine-specific parameter

Fig. 2: **Performance functions of universal and machine-specific parameters** – Each line represents the performance function of a parameter in a different environment. For example, 4c6g represents the performance function of a parameter in an environment with 4 CPU cores and 6 GB of memory.

D. Limitations of Existing Methods

Each method described in Section II-B does not consider the difference between universal and machine-specific parameters, which leads to disadvantages in different ways. ModelShift and DataReuseTL work under the assumption that all parameters are universal. Since machine-specific parameters in DBMS may affect the performance differently depending on the underlying hardware environment, the resulting model may suffer from a bias caused by the difference between the source and target environments. Similarly, L2S does not distinguish between machine-specific and universal parameters. Because L2S does not use the source environment data for model construction, there are no concerns about the bias. However, it does not exploit the source environment data of universal parameters, losing on the opportunity to reduce the amount of samples necessary from the target environment to build a practical model. This is especially problematic in the case of DBMS. Since the parameter space of DBMS is vast, L2S has to compensate for the lack of source environment data by collecting more samples from the target environment.

We need to differentiate between universal and machine-specific parameters to build a high-accuracy DBMS performance prediction model using fewer samples from the target environment.

III. ChimeraTL

ChimeraTL is a transfer learning method that combines the elements of ModelShift, DataReuseTL, and L2S in a way that maximizes their advantages and minimizes their disadvantages. It consists of three procedures: (1) parameter selection and separation, (2) linear transformation learning, and (3) machine-specific parameter learning.

A. Parameter Selection and Separation

The first step of ChimeraTL is to select parameters that significantly impact performance and separate them into universal and machine-specific parameters. In the studies of DBMS performance models, parameter selection is a common practice

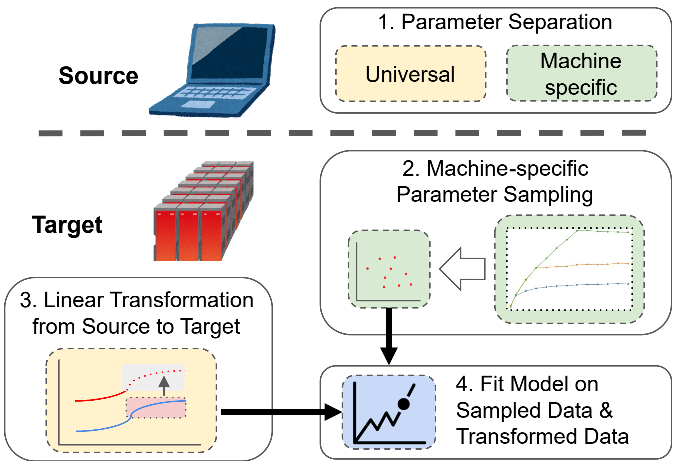

Source
1. Parameter Separation
Universal
Machine specific
Target
2. Machine-specific Parameter Sampling
3. Linear Transformation from Source to Target
4. Fit Model on Sampled Data & Transformed Data

Fig. 3: **ChimeraTL Pipeline**

because the number of parameters in DBMS is usually large, and not all of them significantly impact performance [1], [4]. As in L2S [9], we use stepwise regression [15] to automatically select impactful parameters.

While many studies have incorporated parameter selection into their approach, to the best of our knowledge, no existing method separates parameters into universal and machine-specific ones. Therefore, we propose an algorithm that uses Bhattacharyya distance [16] to separate parameters based on the difference in their performance functions between two environments.

Bhattacharyya distance is a measure of the similarity between two probability distributions. We use Bhattacharyya distance for parameter separation because it considers the entire distribution, accounting for variations beyond mean, variance, or absolute differences [17]. Additionally, Bhattacharyya distance's symmetric nature allows straightforward threshold setting, providing consistent interpretation regardless of the chosen reference distribution.

The parameter separation algorithm is shown in Algorithm 1. The process requires data of performance functions

Algorithm 1 Parameter separation

```
Require:
    $\boldsymbol{P}$: set of parameters
    $f_1$: performance function in primary source environment
    $f_2$: performance function in secondary source environment
    $T$: threshold
Ensure:
    $\boldsymbol{P_{uv}}$: set of universal parameters
    $\boldsymbol{P_{ms}}$: set of machine-specific parameters

 1: $\boldsymbol{P_{uv}} \leftarrow \emptyset$
 2: $\boldsymbol{P_{ms}} \leftarrow \emptyset$
 3: for $\boldsymbol{p_i} \in \boldsymbol{P}$ do
 4:     $p_1(\boldsymbol{p_i}) \leftarrow$ normalize$(f_1(\boldsymbol{p_i}))$
 5:     $p_2(\boldsymbol{p_i}) \leftarrow$ normalize$(f_2(\boldsymbol{p_i}))$
 6:     $d \leftarrow$ BhattacharyyaDistance$(p_1(\boldsymbol{p_i}), p_2(\boldsymbol{p_i}))$
 7:     if $d < T$ then
 8:         $\boldsymbol{P_{uv}} \leftarrow \boldsymbol{P_{uv}} \cup \boldsymbol{p_i}$
 9:     else
10:         $\boldsymbol{P_{ms}} \leftarrow \boldsymbol{P_{ms}} \cup \boldsymbol{p_i}$
11: return $\boldsymbol{P_{uv}}$, $\boldsymbol{P_{ms}}$
```

in primary and secondary source environments. Under the assumption that it is cheap to collect data outside the target environment, we set up two docker containers with different resource restrictions in a source environment to obtain distinct data. The initial step of the algorithm is to convert each performance function of a parameter into a probability distribution (Line 4, 5). Then, the Bhattacharyya distance between the two probability distributions is calculated (Line 6). If the distance is smaller than a threshold T, the parameter is considered universal, and otherwise, it is considered machine-specific (Line 7).

B. Learning in Target Environment

Once the parameters are separated into universal and machine-specific parameters, ChimeraTL is ready to collect data from the target environment. For the first few samples, ChimeraTL focuses on collecting data of universal parameters to learn the linear transformation between the source and target environments. With just a few samples, ChimeraTL can learn the linear transformation and reuse the source environment data of universal parameters for target model construction. After collecting a set amount of samples for linear transformation learning, ChimeraTL prioritizes collecting data of machine-specific parameters unavailable in the source environment.

One iteration of the sampling process is shown in Algorithm 2. The sampling rate of universal parameters (ur) and the number of samples for priority switching (N) are the two parameters of ChimeraTL that control the sampling process.

At the end of each iteration, ChimeraTL fits a performance prediction model using the collected data. We use Gaussian Process (GP) regression [18] as the model because it is known to be effective in modeling configurable systems [4], [6], [7], [9], [10].

1) Linear Transformation Learning: Learning the linear transformation between the source and target environments is an essential step that enables ChimeraTL to reuse the data

Algorithm 2 Sampling in the target environment

```
Require:
    $\boldsymbol{P_{uv}}$: set of universal parameters
    $\boldsymbol{P_{ms}}$: set of machine-specific parameters
    $N$: number of samples for priority switching
    $ur$: sampling rate of universal parameters
    $f_{src}$: performance function in the source environment
    model: performance prediction model
Ensure:
    $D_{tgt}$: target data
    $D_{uv}^{tgt}$: target data of universal parameters
    $D_{uv}^{src}$: source data of universal parameters

 1: function RunNextIteration
 2:     sample_universal $\leftarrow$ Random() $< ur$
 3:     if sample_universal then
 4:         SampleUniversal()
 5:         if size$(D_{uv}^{tgt}) = N$ then
 6:             $ur \leftarrow 1 - ur$
 7:     else
 8:         SampleMachineSpecific()
 9:     FitModel$(D_{train})$
10:
11: function SampleUniversal
12:     $\boldsymbol{p_i} \leftarrow$ RandomChoice$(\boldsymbol{P_{uv}})$
13:     $D_{uv}^{src} \leftarrow D_{uv}^{src} \cup \{f_{src}(\boldsymbol{p_i})\}$
14:     $D_{uv}^{tgt} \leftarrow D_{uv}^{tgt} \cup \{$sample$(f_{tgt}(\boldsymbol{p_i}))\}$
15:     regression_data $\leftarrow$ MergeOnConfig$(D_{uv}^{src}, D_{uv}^{tgt})$
16:     $\boldsymbol{\beta} \leftarrow$ LinearRegression(regression_data)
17:     $D_{tgt} \leftarrow D_{tgt} \cup \{f_{tgt}(\boldsymbol{p_i})\}$
18:
19: function SampleMachineSpecific
20:     $\boldsymbol{p_i} \leftarrow$ RandomChoice$(\boldsymbol{P_{ms}})$
21:     $D_{tgt} \leftarrow D_{tgt} \cup \{f_{tgt}(\boldsymbol{p_i})\}$
22:
23: function FitModel
24:     universal_data $\leftarrow \{f_{src}(\boldsymbol{p_i}) \mid \boldsymbol{p_i} \in \boldsymbol{P_{uv}}\}$
25:     data $\leftarrow$ Transform(universal_data, $\boldsymbol{\beta}$)
26:     data $\leftarrow \{d \in$ data $\mid$ Config$(d) \notin$ Config$(D_{tgt})\}$
27:     data $\leftarrow$ data $\cup D_{tgt}$
28:     model.fit(data)
```

of universal parameters from the source environment. Reusing the linearly transformed source data dramatically reduces the number of samples from the target environment required to build a model.

Each time a new sample of universal parameters is collected from the target environment, ChimeraTL merges the sample with the source environment data of the same configuration and fits a linear regression model to the merged data (Line 11). The linear regression model is used to transform the universal parameter data in the source environment into an estimation of the performance in the target environment (Line 25). The transformed data is then combined with the samples from the target environment to build a performance prediction model (Line 27).

Unlike ModelShift [11], ChimeraTL does not assume that all the parameters are universal. Constructing a linear regression model using all the parameters would cause negative transfer because the machine-specific parameters may

TABLE I: Parameters of LineairDB

Parameter	Range	Default value
`clients`	1-24	1
`checkpoint_interval`	1-30	30
`epoch_duration`	1-40	40
`prefetch_locality`	0-3	3
`rehash_threshold`	0.1-0.99	0.8

not follow the pattern of the universal parameters. Instead, ChimeraTL only samples the data of universal parameters selected in Section III-A to learn the linear regression. Similarly, the linear regression model is only used to transform the source data of universal parameters, and the source data of machine-specific parameters are not used in the target model construction.

2) Machine-specific Parameter Learning: Initially, ChimeraTL samples the data of universal parameters more frequently than the other parameters. However, the benefit of sampling those data diminishes after learning the linear transformation because the linearly transformed source data can substitute the target environment data of universal parameters for model construction. Once there are enough data of the universal parameters, ChimeraTL prioritizes sampling the data of machine-specific parameters from the target environment to capture the trends of these parameters that are not observable in the source environment.

IV. Evaluation

We compare ChimeraTL with the following four methods: **ModelShift**, **DataReuseTL**, **L2S**, and **L2S+DataReuseTL** (a combined approach of DataReuseTL and L2S [9]).

Although the original papers of ModelShift and DataReuseTL do not conduct parameter selection, we perform parameter selection using stepwise regression [15] for these methods to achieve the best performance. In addition, we use the same GP regression model as the performance prediction model for all the methods to ensure a fair comparison of transfer learning approaches.

A. Experimental Setup

For all the experiments, we use LineairDB [14], an open-source transactional storage engine based on Silo [19], to generate performance data. The parameter space of LineairDB is shown in Table I.

Table II shows the source and target environments used in the experiments. We set up a docker container on each environment to control the amount of resources available. We use two completely different host machines to simulate the case where developers test their software on a local machine and deploy it on a server.

We measure the performance of LineairDB by running YCSB-A workload [20]. For each sampling process, the benchmark is executed for 15 seconds, and the performance is assessed by calculating the average over this duration. In all the experiments, we assume the workload is the same between the source and target environments. This assumption is realistic because application developers test their software for expected workloads before deploying it to the target environment. For the performance metric, we use the throughput of the database, which is the number of transactions processed per second.

TABLE II: **Source and target environments** – The secondary source environment is used for the parameter separation. The host machines for the source and target are Apple M3 MAX *laptop* and PRIMERGY RX2540 M4 *server* with Intel(R) Xeon(R) Gold 6130, respectively.

Docker Environment	#Cores	RAM Size
Primary Source (Laptop)	8	12 GB
Secondary Source (Laptop)	4	6 GB
Target (Server)	24	32 GB

B. Parameter Separation Result

ChimeraTL's parameter separation algorithm categorizes `clients` parameter as machine-specific and the rest as universal. As Fig.2b shows, it is visibly clear that the performance functions of the parameter do not have a linear relationship with each other. The classification results from ChimeraTL align with our intuitive expectations.

The parameter separation completes in about 14 milliseconds, incurring a negligible cost relative to the overall model training procedure. In contrast, it substantially improves the effectiveness of ChimeraTL's transfer learning as discussed in Section IV-D.

C. Comparison with Existing Methods

Fig. 4 shows the number of samples required to achieve certain level of prediction accuracy for each transfer learning method. The x-axis represents the mean absolute percentage error (MAPE) [9], [11], [21] of the performance prediction model, and the y-axis represents the number of samples from the target environment.

While the existing methods require many samples to minimize the prediction error, ChimeraTL can reduce the error to below 40% with just 10 samples. This result demonstrates the effectiveness of ChimeraTL's aim to linearly transform and reuse the source data for the target model construction. ChimeraTL further reduces the prediction error to under 10% with 80 samples, while other methods require at least 150 samples to achieve the same accuracy. ChimeraTL's steady improvement in prediction accuracy with an increased number of samples justifies its strategy to prioritize sampling for machine-specific parameters once enough samples are collected for linear transformation learning. With ChimeraTL achieving the best prediction accuracy for all the number of samples, it is clear that ChimeraTL can build an accurate performance prediction model with fewer samples from the target environment.

Contrary to the findings of the previous work that ModelShift needs less than 10 samples to learn the linear transformation [11], ModelShift requires 80 samples to achieve its best prediction accuracy of just around 40% in our experiments.

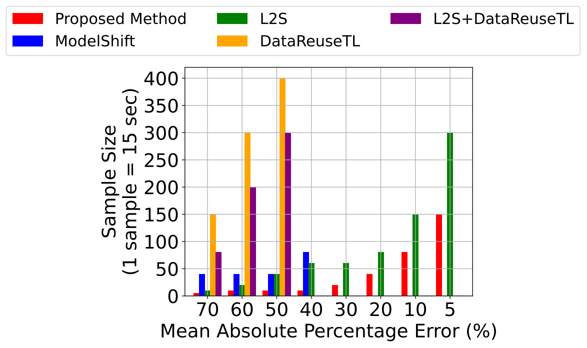


Fig. 4: **Performance prediction accuracy of transfer learning methods** – ModelShift, DataReuseTL, and L2S+DataReuseTL cannot reduce the error below 40% even with all the samples from the target environment.

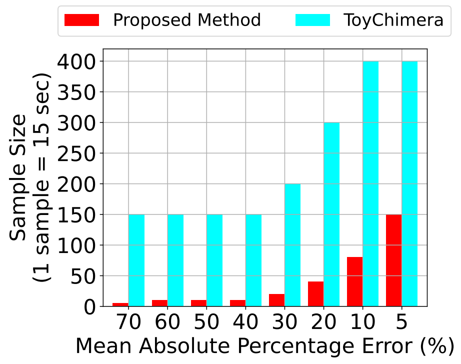


Fig. 5: **Performance prediction accuracy of ChimeraTL and ToyChimera**

This is because there are machine-specific parameters in LineairDB that interfere with learning the linear transformation. DataReuseTL and L2S+DataReuseTL display improvement in prediction accuracy, but their prediction accuracy never reaches below 40% even with all available samples from the target environment because there is a huge difference between the DBMS performance in the source and target environments. Even though ChimeraTL linearly transforms the source data and reuses it for the target model construction, the effect of negative transfer is minimized because ChimeraTL excludes the source data of machine-specific parameters in the process.

In contrast to the methods that suffer from the negative transfer, L2S shows a steady improvement in prediction accuracy as the number of samples increases. Still, L2S generally requires twice as many samples as ChimeraTL to achieve the same prediction accuracy. Since L2S does not use the source environment data at all to train the model, it requires more samples from the target environment to build a model with the same accuracy as ChimeraTL.

D. Importance of Parameter Separation

To understand the effect of parameter separation, we compare ChimeraTL with ToyChimera, a variant of ChimeraTL that does not exclude the data of machine-specific parameters from the linear transformation learning and the source data reuse, and samples data randomly instead of giving priority to specific parameters at different stages of the sampling process.

As shown in Fig. 5, ToyChimera needs much more samples than ChimeraTL does to learn an accurate model. ToyChimera requires 400 samples to reduce the error to below 5% when ChimeraTL can achieve the same accuracy in just 150 samples.

As explained in Section IV-C and shown in Fig. 4, involving machine-specific parameters in the linear transformation learning and reusing all the source data cause negative transfer that significantly degrades the prediction accuracy. ChimeraTL filters out the data of machine-specific parameters for these processes to reuse the source data without suffering from the negative transfer. Consequently, ChimeraTL can build a more accurate performance prediction model with fewer samples from the target environment.

V. Related Work

Although performance prediction models are commonly used to recommend optimal configurations for dynamic workloads [1], [3], [4], [6], few studies, like ResTune [7], leverage knowledge from different hardware environments to build a model. ResTune employs meta-learning to combine models from multiple environments. Its main concern is not on the accuracy itself but on finding optimal configurations for a new workload, which differs from ChimeraTL's focus on accuracy.

Several research efforts in transfer learning [11], [12] have aimed to learn models on new hardware environments efficiently. They showed that software performance in one environment strongly correlates with that in another. Some work [9], [12] found that impactful parameters are likely to be shared between the source and target environments. These results were obtained on configurable systems with binary parameters, and it was unclear if their approaches were applicable to DBMS with non-binary parameters.

In this paper, we found that some parameters in DBMS are universal and can be transformed linearly, but others are not. The difference limited the performance of existing transfer learning techniques that assumed a strong linear relationship between the source and target environments [10], [11].

VI. Conclusion

While some parameters in DBMS have similar effects on performance across different hardware environments, others can have varying effects that depend on underlying hardware limitations. Previous studies do not leverage this information to improve the effectiveness of transfer learning. We proposed ChimeraTL, a novel method that accounts for different parameter types to enhance transfer learning in DBMS. We adopted a novel parameter separation technique which allowed us to employ different transfer learning methods for each group of parameters. Our experiments demonstrated that ChimeraTL needs only 50% of samples that state-of-the-art methods require to minimize the prediction error to under 10%.

References

[1] X. Zhang, Z. Chang, Y. Li, H. Wu, J. Tan, F. Li, and B. Cui, “Facilitating database tuning with hyper-parameter optimization: A comprehensive experimental evaluation,” *Proc. VLDB Endow.*, 2022.

[2] K. Kanellis, R. Alagappan, and S. Venkataraman, “Too many knobs to tune? towards faster database tuning by pre-selecting important knobs,” in *Proc. HotStorage’20*, 2020.

[3] J. Zhang, Y. Liu, K. Zhou, G. Li, Z. Xiao, B. Cheng, J. Xing, Y. Wang, T. Cheng, L. Liu, M. Ran, and Z. Li, “An end-to-end automatic cloud database tuning system using deep reinforcement learning,” in *Proc. SIGMOD*, 2019.

[4] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, “Automatic database management system tuning through large-scale machine learning,” in *Proc. SIGMOD*, 2017.

[5] S. Duan, V. Thummala, and S. Babu, “Tuning database configuration parameters with ituned,” *Proc. VLDB Endow.*, 2009.

[6] X. Zhang, H. Wu, Y. Li, J. Tan, F. Li, and B. Cui, “Towards dynamic and safe configuration tuning for cloud databases,” in *Proc. SIGMOD*, 2022.

[7] X. Zhang, H. Wu, Z. Chang, S. Jin, J. Tan, F. Li, T. Zhang, and B. Cui, “Restune: Resource oriented tuning boosted by meta-learning for cloud databases,” in *Proc. SIGMOD*, 2021.

[8] S. Matsuura and T. Miyazaki, “Real-world challenges of ml-based database auto-tuning,” in *Proc. PACMI*, 2022.

[9] P. Jamshidi, M. Velez, C. Kästner, and N. Siegmund, “Learning to sample: Exploiting similarities across environments to learn performance models for configurable systems,” in *Proc. ESEC/FSE*, 2018.

[10] P. Jamshidi, M. Velez, C. Kästner, N. Siegmund, and P. Kawthekar, “Transfer learning for improving model predictions in highly configurable software,” in *Proc. SEAMS*, 2017.

[11] P. Valov, J.-C. Petkovich, J. Guo, S. Fischmeister, and K. Czarnecki, “Transferring performance prediction models across different hardware platforms,” in *Proc. ICPE*, 2017.

[12] P. Jamshidi, N. Siegmund, M. Velez, C. Kästner, A. Patel, and Y. Agarwal, “Transfer learning for performance modeling of configurable systems: An exploratory analysis,” in *Proc. ASE*, 2017.

[13] Oracle, “Reference manual, ver. 8.0, sec. 15.14, innodb startup options and system variables,” 2023. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html

[14] GitHub, “Lineairdb/lineairdb,” 2023. [Online]. Available: https://github.com/LineairDB/LineairDB

[15] R. R. Hocking, “A biometrics invited paper. the analysis and selection of variables in linear regression,” *Biometrics*, 1976.

[16] A. Bhattacharyya, “On a measure of divergence between two statistical populations defined by their probability distributions,” *Bulletin of the Calcutta Mathematical Society*, 1943.

[17] G. Marsaglia, W. W. Tsang, and J. Wang, “Evaluating kolmogorov’s distribution,” *Journal of Statistical Software*, 2003.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[19] S. Tu, W. Zheng, E. Kohler, B. Liskov, and S. Madden, “Speedy Transactions in Multicore In-Memory Databases,” in *Proc. SOSP*, 2013.

[20] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with ycsb,” in *Proc. SoCC*, 2010.

[21] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, 2006.