Received 24 November 2023, accepted 10 December 2023, date of publication 15 December 2023, date of current version 22 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343455

APPLIED RESEARCH

TATune: A RocksDB Knob Tuning System Based on Transformer

YUN-ZHANG HU AND HUI WANG
School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 201418, China
Corresponding author: Hui Wang (zgwangh@163.com)

ABSTRACT RocksDB is a powerful database engine that offers a wide range of adjustable knobs, which greatly influence its performance. However, configuring RocksDB manually for optimal performance is challenging due to the large number of available knobs and their complex settings. To address this issue, we propose Transformer Adaptive Gentic Algorithm Tune(TATune), an auto-tuning system for RocksDB knobs. In TATune, knob configuration files for RocksDB are randomly generated and executed at different preset workloads first. Subsequently, the correlation between the knob and RocksDB performance is learned by the prediction model based on Transformer. Finally, an adaptive genetic algorithm that utilizes the prediction model as a fitness function to recommend the RocksDB knob setting. Additionally, a novel optimization metric is also proposed to evaluate the performance of the auto-tuning RocksDB knob system. TATune is compared with other approaches to configure RockDB knobs on six distinct workloads. The results indicate that TATune is effective and achieves significant performance improvement across various target workloads. The final average optimization performance is 26% better than K2vTune and 72% better than RTune.

INDEX TERMS RocksDB, auto-tuning, knob configuration, transformer, adaptive genetic algorithm.

I. INTRODUCTION

The continuous growth of information technology has resulted in an ever-increasing amount and diversity of data being generated daily, particularly with the rapid proliferation of unstructured data. Relational databases are ineffective in handling unstructured data, and their distributed implementation is challenging, rendering them incapable of responding promptly to large amounts of data. In response, key-value databases including RocksDB have been proposed as solutions.

Developed by Facebook, based on Google's open-source LevelDB, RocksDB is a key-value database structured on the Log-Structured Merge-tree(LSM-tree) [1]. The in-place update method of LSM-tree leads to excellent write performance [2]. However, an in-place update incurs a loss in read performance, which is especially detrimental to read-intensive applications such as order systems and search engines. Improving read performance requires writing additional information, such as indexes, which can result in write amplification [3], [4], [5] and space amplification [6]. LSM-tree performance is primarily enhanced by optimizing the timing of compression strategies [7], [8], [9], employing key-value separation [10], [11], and utilizing automatic tuning [12], [13], [14]. To mitigate database performance fluctuations caused by compression and merging operations, the timing of compression strategies can be optimized by reducing these operations under high-load conditions, given their significant disk input and output requirements. Key-value separation stores the values separately to avoid unnecessary movement of the data values during compression and merging. This is because the compression and merging operations only require comparing the keys of the merged data. Given that tuning LSM-tree can be challenging due to the many tuning spaces and potential interdependencies between them, automatic tuning primarily involves parameter tuning, merging strategy adjustment, and dynamic allocation of memory for Bloom filters to facilitate the tuning process.

According to previous research, the configuration of knobs has a considerable effect on database performance [15].

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Huang.

However, optimizing the RocksDB knob is difficult as there are many knobs available for debugging that may affect each other. To solve this problem, this study introduces an auto-tuning system, Transformer Adaptive Gentic Algorithm Tune(TATune), for RocksDB knobs. Firstly, it begins by creating a dataset comprising RocksDB knob configurations and database performance indicators. Subsequently, the neural network is trained on this dataset to obtain the feature vectors of the knobs under specific workloads. Finally, Knobformer is utilized to construct a prediction model that forecasts the connection between the feature vectors of knobs and their external indicators. The fitness function of an adaptive genetic algorithm is the prediction model, which is used to optimize the configuration of the RocksDB knobs.

The main contributions of this study are as follows:(1) We produce 12 distinct datasets of read-intensive workloads, each comprising 20,000 configurations of RocksDB knobs. (2) We suggest Knobformer, an enhanced transformer-based model, to forecast the connection between RocksDB knob configurations and RocksDB performance and verified that Knobformer has an advantage over the classical Transformer in predicting knob external metrics. (3) We introduce an adaptive genetic algorithm to improve the global optimization performance by allowing the model to avoid local optima more effectively. (4) We present a new formula for the RocksDB performance score to more accurately assess the optimization performance of genetic algorithm.

II. RELATED WORK

A. ROCKSDB

RocksDB is a key-value database that utilizes the LSM-tree structure [16]. It comprises a Memtable, Sorted String Table (SST, for short), and Compaction. Memtable is an in-memory data structure that employs a skip list for implementation. When the storage of Memtable reaches its capacity, RocksDB automatically transforms Memtable into an Immutable structure and creates a new Memtable to receive further data. In addition, the immutable structure periodically transforms into SST files, which are then stored on the disk. SST is a disk-based data structure that stores data in a format that is suitable for persistent storage. Specifically, SST is composed of several components, including the data block, index block, meta block, meta index block, and footer [17]. The data block stores compressed key-value pairs to achieve high storage efficiency while minimizing the impact on read and write performance. The index block stores metadata associated with the data block. Meta block stores additional information, such as Bloom filters [18], [19], to enhance query performance. The meta-index block stores index pointers in the meta-block. The footer stores index pointers for both the meta index and the index block. The LSM-tree manages SST files in layers for fast retrieval, and the number of layers has a varying degree of impact on RocksDB's performance. Compaction scans the SST files in level$_i$ and level$_{i+1}$, identifies the intersecting SST files, merges them, and places them in leveli+1 [20]. Level0 stores Immutable files directly in memory, and the SST files within the level do not guarantee group order to avoid possible write bottlenecks. Conversely, the SST files in other levels ensure group order. Compaction prevents excessive invalid data from affecting read performance, and the periodic use of compaction can effectively reduce the read and space amplification of LSM-tree to ensure data consistency. However, the compaction strategy requires moving and merging data which causes write amplification [21], [22]. Different compaction strategies and trigger conditions affect the final performance of RocksDB.

B. ROCKSDB KNOB TUNING

In recent years, scholars have conducted research on tuning RocksDB knobs. RTune [23] selects a workload that is most similar to the target workload and then creates a combined workload using the Mahalanobis distance that is as close to the actual target workload as possible. Subsequently, a deep neural network model with a combined workload is trained and used as the fitness function for the genetic algorithm. Finally, RTune applies a genetic algorithm to find the best solution for the original target workload using the fitness function provided by the trained neural network model. However combine workload loses the specificity of each workload in order to maintain universality. This in turn leads to poor performance in predicting new types of workloads. K2vTune [24] uses Euclidean distance to find a workload similar to the target workload and trained a prediction model as the fitness function in the genetic algorithm. K2vTune proposes a new method, Knob2vec, to generate knob feature vectors by learning the workload characteristics of the prediction model. It constructs a prediction model using a gated recurrent unit (GRU) model [25] and an attention mechanism [26]. This model structure learns the correlation between knobs and database performance and provides a basis for the genetic algorithm to recommend knob configurations. However, the K2vTune method has several limitations. Firstly, Euclidean distance does not consider the relationships between various characteristics. Secondly, calculating the current GRU value relies on the previous value, resulting in a slow computation and biased prediction results. Thirdly, the mutation and crossover rates used in the genetic algorithm for recommending knob configurations are fixed values that may easily drop into local optima. Finally, the score evaluation formula in this method can only represent the difference before and after optimization. It cannot indicate whether there is a difference between positive and negative optimizations.

III. METHODOLOGY

A. TATUNE OVERVIEW

Figure 1 presents an overview of the TATune structure. Initially, 20,000 knob configuration files for RocksDB are randomly generated and executed at different preset workloads. The configuration parameters of the knobs are recommended

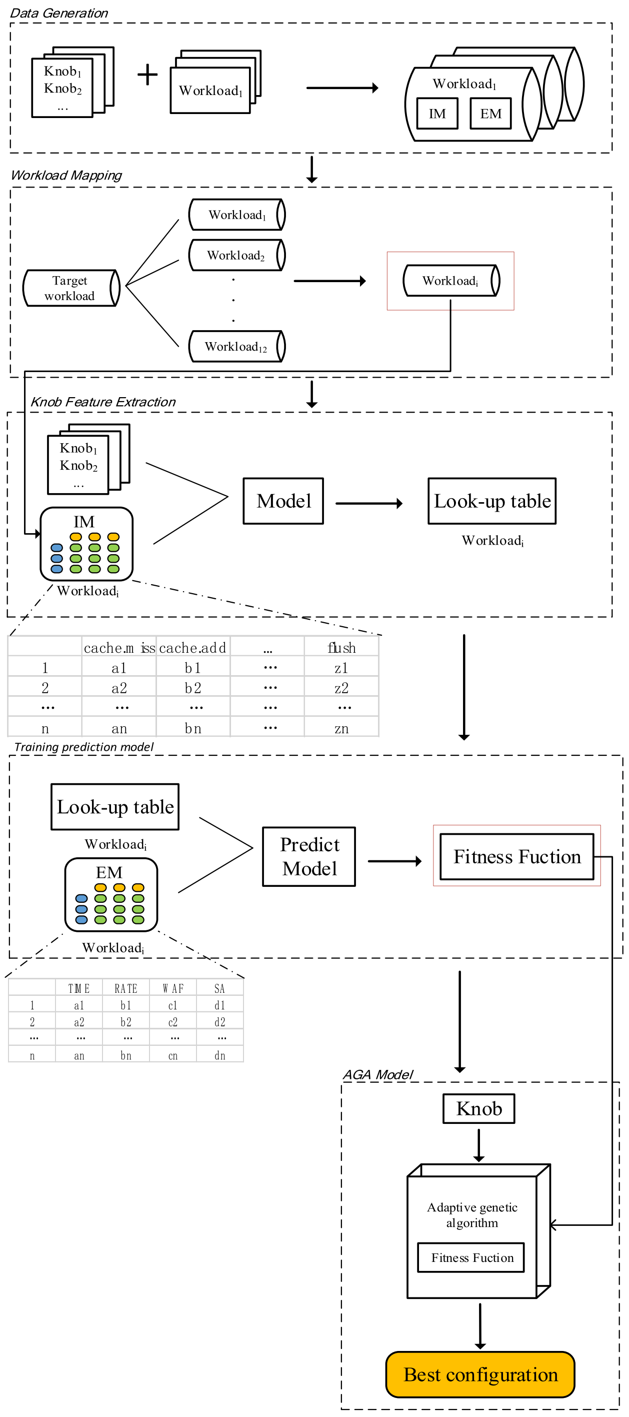


FIGURE 1. Overview of TATune.

by experts in the field of databases [23]. Secondly, the Mahalanobis distance is computed between the external metrics (EM) of the preset workloads and the internal metrics (IM) of the target workload to identify the preset workload with the minimum distance to the target workload. Subsequently, the knob configurations are one-hot encoded and trained by a neural network to extract their feature vectors. Finally, the prediction model is obtained by training the transformer model using the extracted feature vectors of the knobs and the external metrics. An adaptive genetic algorithm, which uses the accepted prediction model as a fitness function, recommends the best configuration of the knob.

B. WORKLOAD MAPPING

Training specific workload indicators in a read-intensive environment can be a tedious and time-consuming task due to the complexity of the workload. Hence, workload mapping is crucial. We find the preset workload with the smallest distance to the target workload to be the most similar workload to the target workload by using Mahalanobis distance to calculate the distance between the internal metrics of the target workload and the internal metrics of the 12 preset workloads. Although Euclidean distance is simple and effective, it cannot consider the relationship between various characteristics as there are over 300 dimensions in the internal metrics and over 140 dimensions after data preprocessing. Mahalanobis distance [27] can consider the relationship between various characteristics by calculating the covariance matrix and can better adapt to the features and changes of the data. The specific formula of the Mahalanobis distance is shown as equation (1):

$$D_M(x, y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)} \tag{1}$$

where x and y are the data of the same dimension, T denotes matrix transpose, Σ is the covariance matrix between the two workloads.

C. EXTRACTION OF KNOB FEATURE VECTORS

The process of learning the feature vectors of knob configurations is illustrated in Figure 2-a. The count of all values for each knob is used to determine the length of the one-hot encoding, which is based on the number of elements in the set. For instance, if $\mathbf{knob_i}$ has a value of 3, and the range of values is [1], [4] (constructed by the set of all configuration values), then the one-hot encoding of $\mathbf{knob_i}$ is [0,0,1,0]. These one-hot encodings are then concatenated to form a sparse vector $\boldsymbol{S} \in \boldsymbol{R}^{1\times d}$. In the second step, the obtained sparse vector V and internal metrics are trained using a neural network. The weight of the linear layer of the model is regarded as the feature vector of the knob, which is then used as the look-up table, as illustrated in Figure 2-b.

Figure 3 illustrates the process of querying the feature vector of a given knob in the query table, which involves two steps. In the first step, the query table is searched for the one-hot encoding of the given knob, and the position of the corresponding feature vector in the query table is determined based on the position of the one in the one-hot encoding. The feature vectors of all knobs are then combined to form a feature matrix, which serves as the training data for the prediction model.

D. IMPROVED ADAPTIVE GENETIC ALGORITHM

To efficiently generate and evaluate knob configurations in order to optimize knob configurations for the purpose of

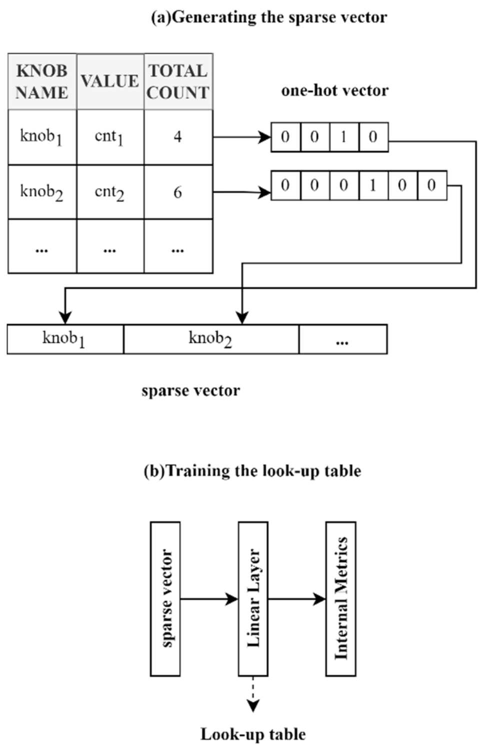


FIGURE 2. Knob feature extraction.

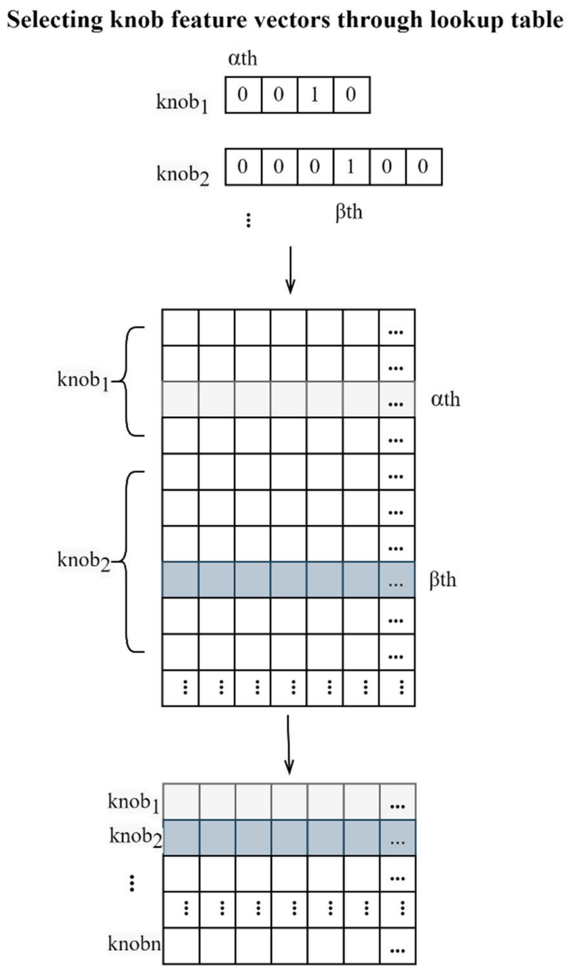


FIGURE 3. Query and splice knobs.

improving the performance of rocksdb, and the properties of genetic algorithms are certainly very suitable for our needs, as each knob can be treated as a gene. That is, the optimization process is to generate knob configurations randomly, then evaluate the resulting knob configurations by an adaptive function, and optimize the generated knob configurations according to the evaluation results. The genetic algorithm(GA) is a stochastic global search optimization method that imitates the biological evolution process and achieves optimization by employing operations such as selection, crossover, and mutation of chromosomes. The adaptive genetic algorithm(AGA) uses crossover and mutation operations with adaptive probabilities to maintain population diversity and preserve the convergence ability of the GA.However, traditional AGA [28] do not take into account the fact that individual fitness values in the population may pull up the overall average fitness value, resulting in a situation of falling into a local optimum.Our proposed improved adaptive genetic algorithm(IAGA) is used to reduce the cases where the AGA algorithm falls into a local optimal solution by considering the sample distribution and number of iterations.The flowchart of the IAGA is presented in Figure 4. The pseudocode of the IAGA employed in this study is presented in Table 1.

1) POPULATION INITIALIZATION

In our research, the population initialization process consists of three distinct stages: firstly, generate the knob configuration randomly. Secondly, the generated knob configuration is coded one-hot. Finally, the one-hot codes are transformed into feature vectors by the lookup table mentioned above.

2) SELECTION

To preserve the most optimal individuals within the population, we undertake a process that involves eliminating the lowest 50% of individuals based on their fitness values. Subsequently, from the retained 50% of individuals, we employ crossover and mutation operations to generate new individuals in a quantity equal to half of the original population size. This approach ensures the creation of a new population that maintains the same size as the previous one. The fitness function will be extensively described in the following predictive model.

3) CROSSOVER AND MUTATION

In our adaptive genetic algorithm, f_{max} represents the maximum fitness value of individuals within the population, while f_{min} represents the minimum fitness. A higher f_{max} indicates better overall fitness and suggests the presence of superior samples within the population. The purpose of evaluating both f_{max} and f_{min} is to assess the distribution of fitness levels among individuals in the population. The f_{max} and f_{min} provide upper and lower bounds for individual fitness within the population. They serve as valuable metrics for evaluating the convergence of the algorithm and the extent of coverage in the search space.Crossover and mutation operated to eliminate 50% of the population with low fitness values and determine the maximum fitness value f_{max}, the minimum fitness value f_{min},and the average fitness value f_{avg} of the remaining population. Subsequently, compute the ratio d, which is the number of samples with higher fitness values divided by the total number of samples, where the samples with higher fitness values are within the range interval of $\left(f_{avg}, f_{max}\right)$. If d is small, this indicates that there are many individuals with lower fitness values in the population and only a few individuals have higher fitness values, requiring the mutation and crossover rates to be increased. Then, the samples are crossed and mutated based on the crossover and mutation rates, and the outcomes are integrated with the initially retained samples to create a new population. The formulas for the crossover rate and mutation rate of the improved genetic algorithm are presented in equation (2) and equation (3), respectively.

$$P_c = \frac{f_{max} - f'}{t \cdot \left(f_{max} - f_{avg}\right)} (1-d) \tag{2}$$

$$P_m = \frac{f_{max} - f}{t \cdot \left(f_{max} - f_{avg}\right)} (1-d) \tag{3}$$

In the IAGA, P_c and P_m denote crossover and mutation rates, respectively. The number of iterations is denoted by t, whereas f' denotes the maximum fitness value of the two individuals about to cross and f denotes the fitness value of the current individual. Ideally, as the number of iterations increases, the fitness values of the samples will approach the optimal fitness value. However, high crossover and mutation rates in the later stages may cause unnecessary fluctuations. An increase in d indicates that a large proportion of individuals in the population possess fitness values higher than the average fitness value. In this case, the mutation and crossover rates should be reduced to prevent population degeneration.The pseudo-code of IAGA can be seen in Table 1.

E. PREDICTIVE MODEL

The fitness function of the improved adaptive genetic algorithm has a significant impact on the accuracy of recommendations for knob configurations. To obtain more precise predictions, we utilized Knobformer, a model enhanced by Transformer, to train the relationship between knob configurations and external indicators.

The conventional transformer replaces the conventional RNN module with an attention mechanism. The model consists of an encoder and a decoder, which map the query and a set of "key-value pairs" to the output. In this study, the encoder maps the input to the hidden layer, and the decoder

TABLE 1. Improved Adaptive Genetic Algorithm Pseudo-code.

```
IAGA
Input:    Population size $n$, Number of Generation $g$ , Predictive
          model $M$
Output:   Best solution $Y_b$
1:        begin
2:          Generation initial population of chromosomes $Y_i$
3:          Set counter $cnt = 0$
4:          for $cnt < g$do
5:            Compute the fitness value of each chromosome:
          $fit = M(Y_i)$
6:            Select half of the top fitness value $Y_{best}$
7:            Calculate mutation rate and crossover rate and $d$
8:            Apply crossover operation on $Y_{best}$
9:            Generate half of $n$ chromosomes $Y_{new}$
11:           Apply mutation on $Y_{new}$
12:           Combine $Y_{best}$ and $Y_{new}$:$Y_b = [Y_{best}: Y_{new}]$
13:         end for
14:         Return the best solution,$Y_b$
15:       end
```

maps the hidden layer to the external metrics. The role of the feed-forward neural network layer is to transform the output of the self-attention layer nonlinearly to acquire a new sequence. Residual connections and normalization are included after each sub-layer to prevent gradient vanishing and overfitting. In addition, the encoder uses a multi-head attention mechanism and position encoding to boost the expression ability of the model. Position encoding adds a vector that represents the position information of each element in the input sequence and combines it with the word embedding vector to enable the model to capture the order relationship of the elements in the sequence. The decoding layer also comprises multiple identical layers, each with two sub-layers: a multi-head attention mechanism and a feed-forward neural network layer.

1) KNOB RELATION CONVOLUTION MODULE

The self-attention mechanism in the Transformer model [29] is limited to learning only the features of individual knobs and cannot account for the influence of inter-knob relationships on the final prediction results. The knob relation convolution module, which is used to extract the relationships between knobs, is proposed in this study to overcome the drawback of the self-attention mechanism in the Transformer model. This module can perceive local correlations between the knob features. Integrating this module into the Transformer encoder can transfer and interact knob features between different layers, further improving the model's expressive power and performance. The knob relation convolution module convolves the rows of the input matrix to generate a feature matrix that encapsulates the interdependence between the knobs and can be expressed as equation (4):

$$Y = \mathrm{ReLU}\left(\mathrm{Conv1d}\left(X^T\right)\right)^T \tag{4}$$

Here, X represents the input tensor, X^T represents its transpose along rows and columns, Conv1d represents the 1D convolution operation with the specified parameters, *ReLU*

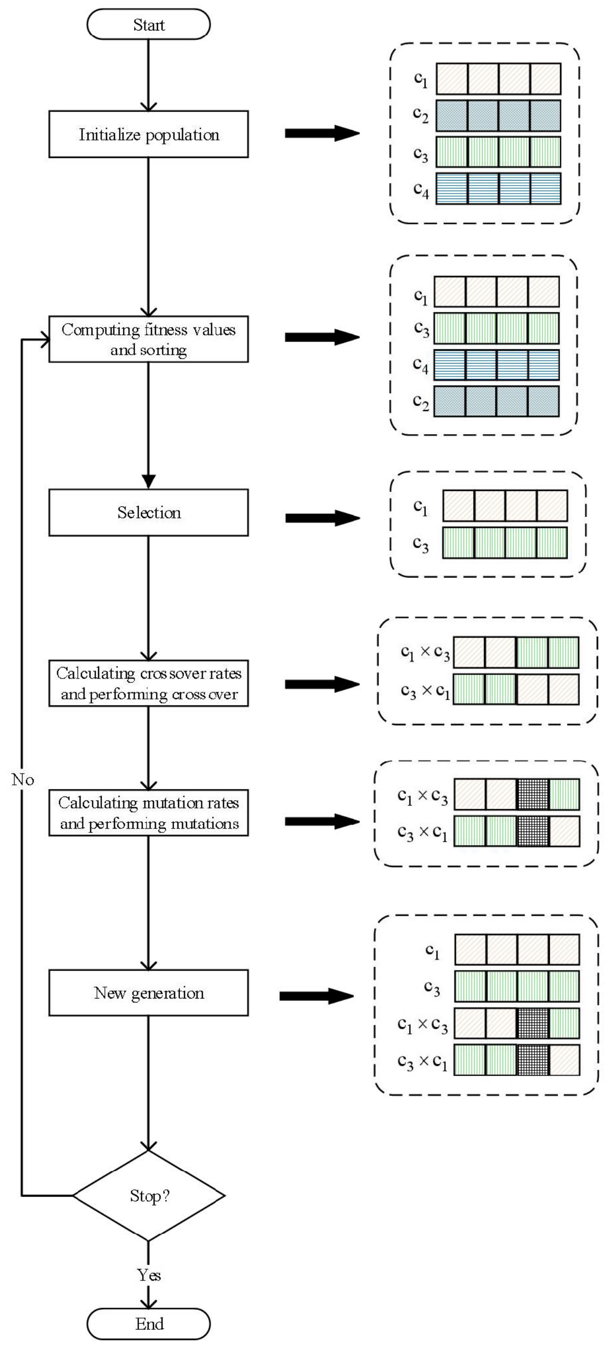

Start
Initialize population
Computing fitness values and sorting
Selection
Calculating crossover rates and performing crossover
Calculating mutation rates and performing mutations
New generation
Stop?
No
Yes
End
c_1
c_2
c_3
c_4
$c_1 \times c_3$
$c_3 \times c_1$

FIGURE 4. Improved adaptive genetic algorithm for TATune.

represents the rectified linear unit activation function. The reason for transposing the input tensor before and after convolution is to ensure that the convolutional kernel operates on each row of the original input tensor separately. By transposing the tensor before convolution, we effectively swap its row and column dimensions, so that convolution is performed along rows instead of columns. The transpose operation after convolution simply restores the original shape of the tensor.

The enhanced encoder architecture is illustrated in Figure 5. The model takes the knob feature vectors as inputs and feeds them into the decoder after passing through the multi-head attention and feedforward neural network. The encoder architecture is composed of fully connected layers, a GELU activation function, and fully connected layers as part of a feedforward neural network. Residual connections are utilized around the multi-head attention and feedforward neural network, and layer normalization is applied to normalize the output of both networks. The encoder transforms knob feature vectors into coding features.

2) MULTI-HEAD ATTENTION

The multi-head attention mechanism is used to segment the input sequence into multiple subspaces and to perform self-attention calculation on each subspace individually. Figure 6 illustrates the structure of the multi-head attention mechanism. After entering the attention module, the feature vector X is divided into the query matrix $Q \in \mathbb{R}^{n \times d_k}$, key matrix $K \in \mathbb{R}^{n \times d_k}$, and value matrix $V \in \mathbb{R}^{n \times d_v}$. The similarity matrix S, with a size of $(n \times n)$, can be obtained by performing dot product multiplication on Q and K, representing the similarity between all points. The global attention feature of size $(n \times d_v)$ can be obtained by multiplying the similarity matrix S by V, and direct residual calculation can be performed because the dimension d of the feature vector is equal to the dimension d_v of the global attention feature. The formula for the conventional attention mechanism are equation (5) and (6):

$$\mathrm{Atten}\left(Q, K, V\right) = \rho\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

$$\rho\left(M\right) = \sigma_r\left(M\right) \tag{6}$$

Scaling factor $\sqrt{d_k}$ is used to prevent the result from being too large or too small. The notation ρ denotes the softmax normalization operation applied to the matrix, while ρ represents the softmax normalization operation on a matrix, σ_r denotes the softmax operation applied to the rows of the matrix. The time complexity of the attention mechanism is $\mathrm{O}\left(dn^2\right)$. Specifically, when given input matrices $Q \in n \times d_k$, $K \in n \times d_k$, and $V \in n \times d_v$, the traditional attention module computes $S = QK^T \in n \times n$ by performing a matrix multiplication between Q and K^T. The final output is obtained by multiplying S and V, leading to a time complexity of $O\left(dn^2\right)$.

IV. EXPERIMENTS AND RESULTS

A. DATESETS AND EXPERIMENTAL ENVIRONMENT

We generate 12 different workloads using the built-in benchmark tool db_bench in RocksDB. Each workload generate 20,000 data points to produce adequate training samples that cater to varying degrees of read-intensive workload environments. To reduce computation, this study selects 22 representative knobs for tuning optima of the numerous knobs present in RocksDB. These knobs include open_files, num_levels, etc., as listed in Table 2. Table 3 lists the precise details of the preset and target workloads for this experiment. The key size of each data is fixed at 16B, and the total size

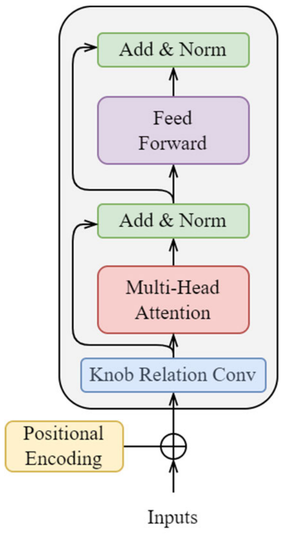


FIGURE 5. Encoder structure diagram.

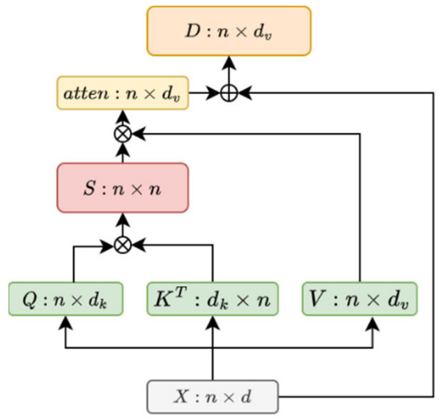


FIGURE 6. Multi-head attention.

TABLE 2. Knob configuration table.

Knob configuration
max_background_compactions
max_background_flushes
write_buffer_size
max_write_buffer_number
min_write_buffer_number_to_merge
compaction_pri
compaction_style
level0_file_num_compaction_trigger
level0_slowdown_writes_trigger
level0_stop_writes_trigger
compression_type
bloom_locality
open_files
block_size
cache_index_and_filter_blocks
max_bytes_for_level_base
max_bytes_for_level_multiplier
target_file_size_base
target_file_size_multiplier
num_levels
memtable_bloom_size_ratio
compression_ratio

of all keys and values is set at 1GB. As a result, the size of a single value is computed as $\frac{\text{data}_{\text{size}}}{\text{data}_{\text{nums}}} - \text{key}_{\text{size}}$, where $\text{data}_{\text{size}}$ is the overall size of the data, $\text{data}_{\text{nums}}$ is the total number of data, and key_{size} represents the size of a single key. The specific data format used in this experiment is shown in Table 3.

The experiments are conducted on a machine equipped with Intel(R) Core(TM) i7-11800H processors and 16GB RAM, NVIDIA GeForce RTX 3060. The operating system used to run RocksDB is 64-bit Ubuntu 18.04 and the operating system used to train the model is 64-bit windows 11. The TATune system is written using the PYTHON programming language in the PyCharm integrated development environment.

B. EVALUATION

1) GENETIC ALGORITHM EVALUATION METRICS

The study selects external metrics including time(TIME), rate(RATE), write amplification factor (WAF), and space amplification factor (SA). The TIME represents the total amount of time required by RocksDB to process data, where a lower value indicates better performance. The RATE represents the amount of data processed per second, where a higher value indicates better performance. The WAF represents the ratio of the actual amount of data written to the expected amount of data written by the user, where a lower value indicates better performance. The SA represents the actual size of data stored in the LSM-tree. Because LSM-tree adopts an in-place update strategy, when data are updated, LSM-tree does not delete the original data but reinserts new data. Therefore, there may be some invalid SST files in LSM-tree;hence, the lower the spaceamplification value, the better the performance. The score is calculated as the sum of the proportions of each metric. The formula is as follows in equations (7),(8), (9), (10), and (11):

$$score = \omega_1 log\,(T) + \omega_2 log\,(R) + \omega_3 log\,(W) + \omega_4 log\,(S) \tag{7}$$

$$T = \begin{cases} \frac{\text{TIME}_{\text{def}}}{\text{TIME}}, & \frac{\text{TIME}_{\text{def}}}{\text{TIME}} > \frac{3}{4} \\ \frac{3}{4}, & \frac{\text{TIME}_{\text{def}}}{\text{TIME}} \le \frac{3}{4} \end{cases} \tag{8}$$

$$R = \begin{cases} \frac{\text{RATE}}{\text{RATE}_{\text{def}}}, & \frac{\text{RATE}}{\text{RATE}_{\text{def}}} > \frac{3}{4} \\ \frac{3}{4}, & \frac{\text{RATE}}{\text{RATE}_{\text{def}}} \le \frac{3}{4} \end{cases} \tag{9}$$

TABLE 3. RocksDB knob configuration optimized for preset workloads and target workloads.

	Pre-set workloads		
	Value size (B)	Total data	Read/write ratio
0			50%
1	4096	261124	90%
2			70%
3			50%
4	1024	1032444	90%
5			70%
6			50%
7	65536	16380	90%
8			70%
9			50%
10	16384	65472	90%
11			70%
	Target workloads		
	Value size (B)	Total data	Read/write ratio
12	8192	130816	80%
13			60%
14	32768	32752	80%
15			60%
16	2048	520223	80%
17			60%

$$W=\begin{cases}\dfrac{\mathrm{WAF_{def}}}{\mathrm{WAF}}, & \dfrac{\mathrm{WAF_{def}}}{\mathrm{WAF}}>\dfrac{3}{4}\\[2ex] \dfrac{3}{4}, & \dfrac{\mathrm{WAF_{def}}}{\mathrm{WAF}}\leq\dfrac{3}{4}\end{cases}\tag{10}$$

$$S=\begin{cases}\dfrac{SA_{def}}{SA}, & \dfrac{\mathrm{SA_{def}}}{\mathrm{SA}}>\dfrac{3}{4}\\[2ex] \dfrac{3}{4}, & \dfrac{\mathrm{SA_{def}}}{\mathrm{SA}}\leq\dfrac{3}{4}\end{cases}\tag{11}$$

where $\mathrm{TIME_{def}}$, $\mathrm{RATE_{def}}$, $\mathrm{WAF_{def}}$, and $\mathrm{SA_{def}}$ are the default configuration values for each metric, which are set as the default settings for RocksDB knob configuration in this study. T, R, W, and S are determined by the ratio of the default configuration value of each metric to its actual value. A logarithmic function is used to prevent a single metric from having a disproportionately large impact on the overall score. The weights ω_1, ω_2, ω_3 and ω_4 represent the impact of each item on the score, and in our experiment, they are all equally set to 0.25. Since represents time, the less time the better, so when TIME is larger than $\mathrm{TIME_{def}}$, the $\frac{\mathrm{TIME_{def}}}{\mathrm{TIME}}$ ratio will be less than 1, which is a negative optimization; similarly, when WAF and SA are larger than $\mathrm{WAF_{def}}$ and $\mathrm{SA_{def}}$, respectively, $\frac{\mathrm{WAF_{def}}}{\mathrm{WAF}}$ and $\frac{\mathrm{SA_{def}}}{\mathrm{SA}}$ are negative optimization. And RATE indicates the rate, the faster the rate the better the effect, RATE is greater than $\mathrm{RATE_{def}}$, $\frac{\mathrm{RATE}}{\mathrm{RATE_{def}}}$ is greater than 1, indicating positive optimization. We use a piecewise function to prevent the logarithmic function from producing unfair results when a metric is negatively optimized and the score decreases too quickly. The value, $\frac{3}{4}$, is determined to be the best value after multiple experiments.

2) PREDICITIVE MODEL EVALUATION METRICS

The mean square error (MSE) and the Pearson correlation coefficient (PCC) are chosen as evaluation metrics to assess the predictive performance of the model in this study. MSE and PCC can be represented by equation (12) and equation (13), respectively.

$$\mathrm{MSE}=\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2\tag{12}$$

$$\mathrm{PCC}=\frac{1}{n}\frac{\sum_{i=1}^{n}\left(y_i-\bar{\mu}\right)\left(\hat{y}_i-\bar{\hat{\mu}}\right)}{\sqrt{\sum_{i=1}^{n}\left(y_i-\bar{\mu}\right)}\sqrt{\sum_{i=1}^{n}\left(\hat{y}_i-\bar{\hat{\mu}}\right)}}\tag{13}$$

where y represents the true value, $\hat{y}$ represents the model's predicted value, n is the total number of evaluation samples, $\bar{\mu}$ is the mean value of the true values of the samples, and $\bar{\hat{\mu}}$ is the mean value of the model's predicted values of the samples. The range of values for the mean square error is $[0, +\infty)$. The closer the mean square error value is to zero, the more accurately the predicted value approximates the true value. The range of values for the Pearson correlation coefficient is [1, -1], with negative values indicating a negative correlation, a value of zero indicating no linear relationship, and positive values indicating a positive correlation. Pearson correlation coefficient values between [0.8,1.0] indicate an extremely strong correlation between predicted and true values, values between [0.6,0.8) indicate a strong correlation, and values between [0.4,0.6) indicate a moderate correlation.

C. ABLATION EXPERIMENTS

Based on K2vTune, our research introduces three main improvements, (1) replacing the original Euclidean distance with the Mahalanobis distance during the workload mapping phase, (2) utilizing the Knobformer model to predict the model during the model prediction phase and (3) adopting IAGA in the final prediction score phase instead of GA. To ensure fairness, the final scores are calculated based on the metrics proposed in this study, and R8W2_32 is selected from the final results, as shown in Table 4. From the results, it can be seen that RocksDB scored 5.8 points using the Mahalanobis distance, which is 3% higher than the 5.6 points using Euclidean distance. This is because that even after data preprocessing, there are still more than 140 internal indicators, and the scale of each indicator is different, and the impact on external indicators is also different. However, the Euclidean distance is not capable of dealing with high dimensions, while the Mahalanobis distance can deal with it well. When we added the Knobformer model to K2vTune, the RockDB score increased by 14% compared with the original K2vTune. This is because that the Knobformer is more accurate in predicting external indicators of the model. When the IAGA is added to the final prediction score phase, the RocksDB score increases

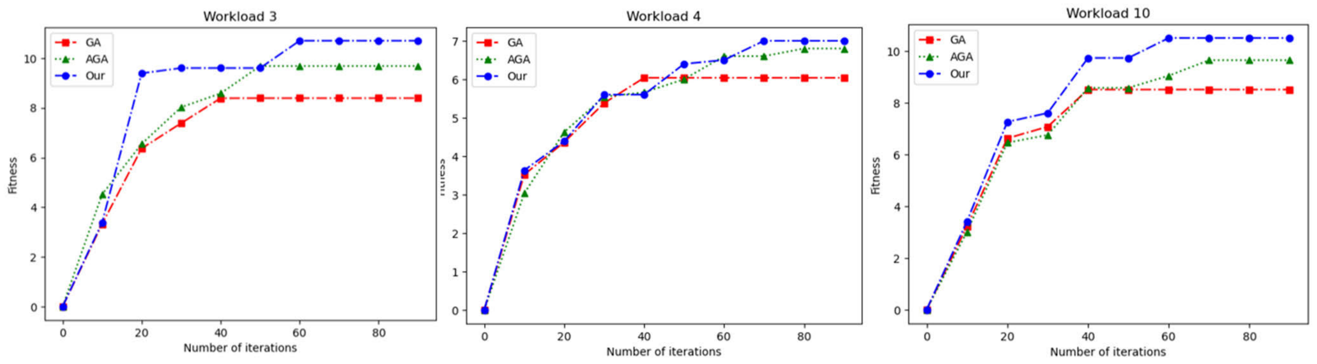


FIGURE 7. Comparison of different genetic algorithms with different workloads.

TABLE 4. Results of ablation experiments at R8W2_32.

Marginal distance	Knobformer	IAGA	Score
			5.6
√			5.8
	√		6.4
		√	6.4
√	√	√	7

by 14% compared with the original K2vTune.This is because the improved adaptive genetic algorithm dynamically adjusts the mutation and crossover rates to avoid premature convergence, which may occur with the traditional genetic algorithm when the maximum value cannot be found.

1) ADAPTIVE GENETIC ALGORITHM

To evaluate the advantages of our proposed improved adaptive genetic algorithm over GA and AGA in the optimization of the RocksDB knob configuration, we use the prediction model as the fitness function to compare the respective fitness values of these three algorithms under workloads 3, 4, and 10, the results can be seen in figure7. The fitness of the vertical axis in Figure 7 represents the RocksDB score, and the larger the fitness value, the better. From Figure 7, it is evident that after the 20th iteration of the IAGA algorithm in workload 3, it outperforms both AGA and GA. The AGA algorithm exhibits superior performance over GA starting from the 10th iteration. In workload 4, IAGA surpasses GA after the 50th iteration and surpasses AGA after the 70th iteration, while AGA outperforms GA after the 60th iteration. Moving to workload 10, IAGA starts to outperform AGA and GA after the 20th iteration, and AGA outperforms GA after the 40th iteration. In summary, based on the above observations, it can be concluded that IAGA exhibits superior performance compared to AGA and GA. This is due to the dynamic adjustment of the crossover rate and mutation rate of the improved adaptive genetic algorithm based on the characteristics of the population and the current optimization state.

2) PREDICITIVE MODEL

The accuracy of the fitness function model significantly affects the optimization results of the final knob configuration when recommending RocksDB knob configurations. To address this, we design the Knobformer prediction model for the fitness function and compare it with several other neural network models, including GRU, BiGRU, GRU+Atten, BiGRU+Atten, and Transformer. To evaluate the performance of these models, we use the PCC and MSE to estimate the linear correlation and mean squared error between the predicted and actual indicators. Table 5 presents the evaluation results of the various prediction models for workloads 3, 4, and 10. Our results indicate that the Knobformer model outperforms the Single, GRU, BiGRU, and Transformer models in most external indicators (marked in bold in Table 5). For workload 3, Knobformer achieves the highest scores in TIME, RATE and SA, whereas GRU+Atten performed best in WAF. However, the prediction accuracies for TIME and RATE are considerably low. As the model predicts multiple indicators and there is a trade-off between the accuracy of each indicator, Knobformer and Transformer have the most accurate predictions overall, with Knobformer having the best predictions in TIME, RATE, and SA. Therefore, the Knobformer model produced the best prediction results for workload 3. The results on workload 10 show a similar trend in terms of accuracy, with Knobformer achieving the best results in predicting TIME, WAF, and SA. BiGRU+Atten performed best in RATE prediction, while the difference between its prediction accuracy in TIME and that of Knobformer and Transformer is slight. There is a significant difference in the MSE indicator in SA, indicating that the Knobformer model performed best for workload 10. For workload 4, Knobformer's performance remained stable, achieving the best results in predicting RATE, WAF, and SA. The MSE indicator for predicting TIME is only approximately 0.01 different from that of the top-performing GRU model. The GRU+Atten and BiGRU+Atten models with attention mechanisms perform so poorly that the predicted results are unreliable in workload 4 while performed well in workloads 3 and 10.

TABLE 5. Comparison of the accuracy of various forecasting models.

Workload	Method	TIME		RATE		WAF		SA	
	Model	MSE	PCC	MSE	PCC	MSE	PCC	MSE	PCC
3	Single	0.99	0.3914	36.07	0.7579	0.27	0.9510	0.41	0.9999
	GRU	2.86	0.2368	29.08	0.7958	0.24	0.9564	0.25	0.9999
	GRU+Atten	4.69	0.1987	34.86	0.7583	**0.20**	**0.9668**	2.26	0.9995
	BiGRU+Atten	0.06	0.8290	1.44	0.8847	0.20	0.9665	0.33	0.9999
	Transformer	0.05	0.9129	1.37	0.9492	0.22	0.9613	0.30	0.9999
	Our	**0.04**	**0.9336**	**1.37**	**0.9494**	0.21	0.9643	**0.24**	**0.9999**
Workload	**Method**	**TIME**		**RATE**		**WAF**		**SA**	
	Model	MSE	PCC	MSE	PCC	MSE	PCC	MSE	PCC
4	Single	1.31	0.8927	0.95	0.9027	0.87	0.8445	6.58	0.9995
	GRU	**0.86**	**0.9305**	0.73	0.9254	0.29	0.9512	3.21	0.9997
	GRU+Atten	15.45	0.2324	13.48	-0.256	23.73	-0.031	6672	0.5942
	BiGRU+Atten	13.98	0.3911	15.71	-0.402	24.79	-0.169	5428	0.9439
	Transformer	0.89	0.9286	0.71	0.9275	0.27	0.9544	2.74	0.9998
	Our	0.87	0.9296	**0.69**	**0.9331**	**0.25**	**0.9614**	**1.91**	**0.9999**
Workload	**Method**	**TIME**		**RATE**		**WAF**		**SA**	
	Model	MSE	PCC	MSE	PCC	MSE	PCC	MSE	PCC
10	Single	0.32	0.9584	2.67	0.9712	0.23	0.9164	3.92	0.9998
	GRU	0.26	0.9645	2.22	0.9764	0.16	0.9374	1.18	0.9999
	GRU+Atten	0.26	0.9644	2.32	0.9749	0.18	0.9319	2.30	0.9998
	BiGRU+Atten	0.26	0.9645	**2.15**	**0.9768**	0.16	0.9390	2.60	0.9998
	Transformer	0.26	0.9653	2.42	0.9739	0.14	0.9490	0.44	0.9999
	Our	**0.26**	**0.9655**	2.31	0.9749	**0.13**	**0.9597**	**0.42**	**0.9999**

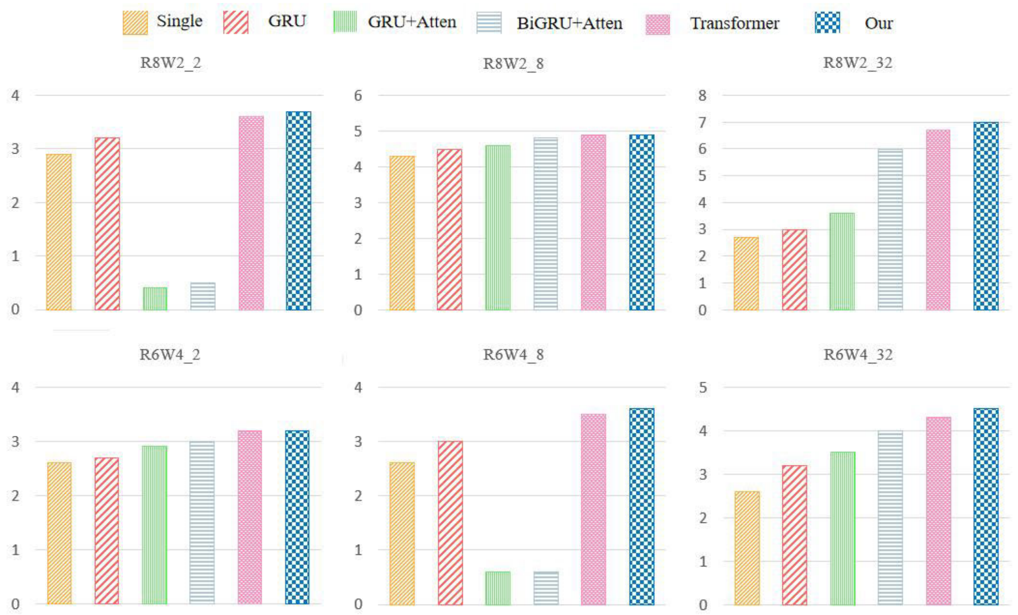


FIGURE 8. Comparing the predictive performance of different model rows on the final tuning performance.

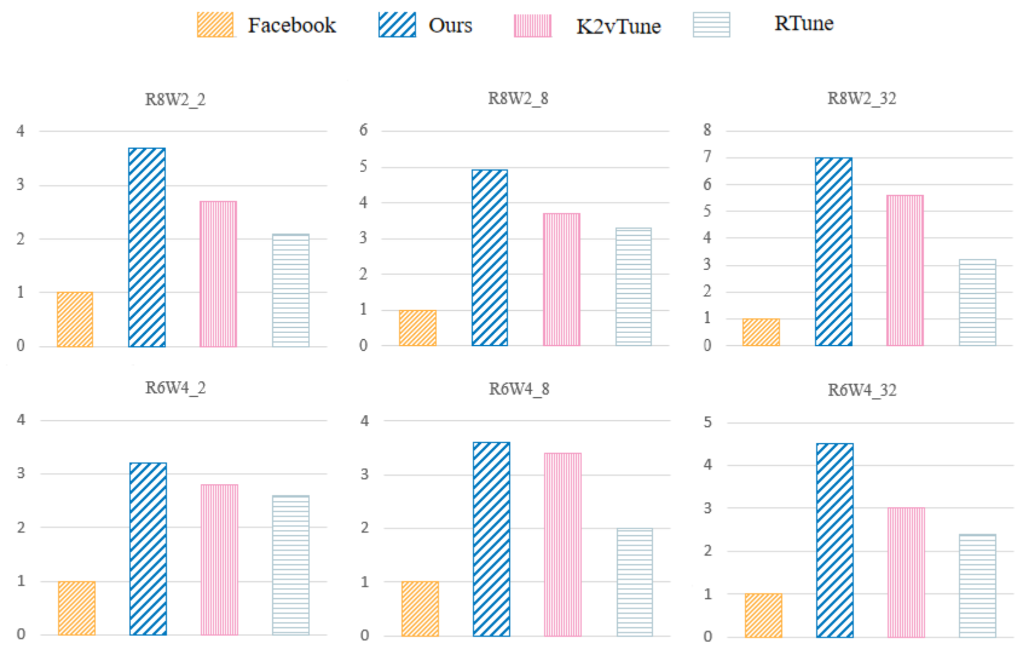


FIGURE 9. Comparing the results of different methods for optimizing RocksDB knob configurations.

To verify that the accuracy of the prediction model affects the final tuning score, the scores of the prediction models are compared by benchmark testing on the recommended configurations, as shown in Figure 8.The results show that Knobformer performs best for all workloads. In addition, models that achieve higher accuracy typically receive higher scores when tested against actual benchmarks.

D. TUNING PERFORMANCE EVALUATION

Our proposed RocksDB optimization configuration performance and improvement method are compared with the optimization configurations of K2vTune, RTune, and Facebook. Among them, Facebook's optimization configuration is the default setting for RocksDB. The score results of these optimization methods under different target workloads are displayed in Figure 9. The results indicate that our proposed method outperforms all other method. K2vTune's score ranks second among the four workloads, and its score in the R6W4_8 workload is almost comparable to that of our proposed method, as depicted in Figure 9 for R6W4_8. The final average optimization performance is 26% better than K2vTune and 72% better than RTune.

V. CONCLUSION

Since RocksDB contains many configuration parameters, users and administrators find it challenging to optimize RocksDB storage performance by adjusting the appropriate parameter values. This study proposes a three-step RocksDB configuration optimization system, TATune, to automate this process and identify more optimal parameter values. Firstly, the Mahalanobis distance is used to map the target workload samples to a preset workload. Subsequently, a neural network model is employed to train the preset workload samples to obtain their feature vectors using the knob configuration and internal indicators of the preset workload. Finally, a transformer model is used to train a predictor that estimates external indicators. This predictor is used as the fitness function of the improved adaptive genetic algorithm to recommend the best configuration. The experimental results demonstrate that our proposed TATune system exhibits significant improvements than other compared methods in recommending knob configurations for RocksDB. Our proposed knobformer improves the performance of RocksDB knobs by 16% over k2vTune. The prediction model and genetic algorithm proposed in this study improves the performance of RocksDB better. However, the accuracy of the predictive model and knob performance optimization still need to be improved.

For future works, we shall embark on an exploration of intelligent optimization algorithms, to discern their impact upon the performance of recommended RocksDB knob configurations, thereby elevating its overall efficiency. The realm of deep learning has witnessed a remarkable surge in large-scale language models over the past couple of years. In light of this, we shall employ the open-source,large language model to delve into how this model can optimize the recommended knob configurations for RocksDB.

REFERENCES

[1] (2023). *RocksDB: A Persistent Key-Value Store for Fast Storage Environments*. [Online]. Available: https://rocksdb.org

[2] C. Luo and M. J. Carey, "LSM-based storage techniques: A survey," *VLDB J.*, vol. 29, no. 1, pp. 393–418, Jan. 2020.

[3] K. Ouaknine, O. Agra, and Z. Guz, "Optimization of RocksDB for redis on flash," in *Proc. Int. Conf. Compute Data Anal.*, Lakeland, FL, USA, May 2017, pp. 155–161.

[4] Z. Cao, S. Dong, S. Vemuri, and D. H. C. Du, "Characterizing, modeling, and benchmarking RocksDB key-value workloads at Facebook," in *Proc. File Storage Technol.*, Jan. 2020, pp. 209–223.

[5] H. Kim, J.-H. Park, S. H. Jung, and S.-W. Lee, "Optimizing RocksDB for better read throughput in blockchain systems," in *Proc. 23rd Int. Comput. Sci. Eng. Conf. (ICSEC)*, Phuket, Thailand, Oct. 2019, pp. 305–309.

[6] S. Dong, M. Callaghan, L. Galanis, D. Borthakur, T. Savor, and M. Strum, "Optimizing space amplification in RocksDB," in *Proc. Conf. Innov. Data Syst. Res.*, vol. 3, Jan. 2017, p. 3.

[7] O. Balmau, F. Dinu, W. Zwaenepoel, K. Gupta, R. Chandhiramoorthi, and D. Didona, "SILK: Preventing latency spikes in log-structured merge key-value stores," in *Proc. USENIX Annu. Tech. Conf.* Jan. 2019, pp. 753–766.

[8] P. Jin, J. Li, and H. Long, "DLC: A new compaction scheme for LSM-tree with high stability and low latency," in *Proc. Int. Conf. Extending Database Technol.*, Jan. 2021, pp. 547–557.

[9] S. Sarkar, T. Papon, D. Staratzis, and M. Athanassoulis, "Lethe: A tunable delete-aware LSM engine," 2020, *arXiv:2006.04777*.

[10] L. Lu, T. S. Pillai, H. Gopalakrishnan, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "WiscKey: Separating keys from values in SSD-conscious storage," *ACM Trans. Storage*, vol. 13, no. 1, pp. 1–28, Feb. 2017.

[11] Y. Li, H. H. W. Chan, P. P. C. Lee, and Y. Xu, "Enabling efficient updates in KV storage via hashing," *ACM Trans. Storage*, vol. 15, no. 3, pp. 1–29, Aug. 2019.

[12] N. Dayan, M. Athanassoulis, and S. Idreos, "Monkey: Optimal navigable key-value store," in *Proc. ACM Int. Conf. Manage. Data*, Chicago, IL, USA, May 2017, pp. 79–94.

[13] H. Lim, D. G. Andersen, and M. Kaminsky, "Towards accurate and fast evaluation of multi-stage log-structured designs," in *Proc. File Storage Technol.* Feb. 2016, pp. 149–166.

[14] N. Dayan and S. Idreos, "Dostoevsky: Better space-time trade-offs for LSM-tree based key-value stores via adaptive removal of superfluous merging," in *Proc. Int. Conf. Manage. Data*, Houston, TX, USA, May 2018, pp. 505–520.

[15] Y. Ishihara and M. Shiba, "Dynamic configuration tuning of working database management systems," in *Proc. IEEE 2nd Global Conf. Life Sci. Technol. (LifeTech)*, Kyoto, Japan, Mar. 2020, pp. 393–397.

[16] F. Mei, Q. Cao, H. Jiang, and L. T. Tintri, "LSM-tree managed storage for large-scale key-value store," in *Proc. Symp. Cloud Comput.*, Santa Clara, CA, USA, Sep. 2017.

[17] (2023). *RocksDB BlockBasedTable Format*. [Online]. Available: https://github.com/facebook/rocksdb/wiki/Rocksdb-BlockBasedTable-Format

[18] M. Naor and Y. Eylon, "Bloom filters in adversarial environments," *ACM Trans. Algorithms (TALG)*, vol. 15, no. 3, pp. 1–30, Jun. 2019.

[19] N. Dayan and M. Twitto, "Chucky: A succinct cuckoo filter for LSM-tree," in *Proc. Int. Conf. Manage. Data*, Jun. 2021, pp. 365–378.

[20] S. Sarkar, D. Staratzis, Z. Zhu, and M. Athanassoulis, "Constructing and analyzing the LSM compaction design space," in *Proc. VLDB Endowment*, vol. 14, no. 11, pp. 2216–2229, Oct. 2021.

[21] F. Pan, Y. Yue, and J. Xiong, "DCompaction: Delayed compaction for the LSM-tree," *Int. J. Parallel Program.*, vol. 45, no. 6, pp. 1310–1325, Nov. 2016.

[22] Y. Yue, B. He, Y. Li, and W. Wang, "Building an efficient put-intensive key-value store with skip-tree," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 961–973, Apr. 2017.

[23] H. Jin, J. Lee, and S. Park, "RTune: A RocksDB tuning system with deep genetic algorithm," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2022, pp. 1209–1217.

[24] J. Lee, J. Choi, S. Seo, and S. Park, "K2vtune: Automatic database tuning with knob vector representation," 2022, doi: 10.2139/ssrn.4225456.

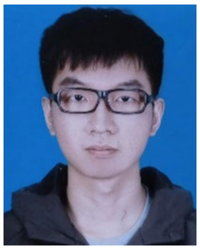
[25] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1724–1734.

[26] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*.

[27] J. Yang and C. Delpha, "An incipient fault diagnosis methodology using local Mahalanobis distance: Detection process based on empirical probability density estimation," *Signal Process.*, vol. 190, Jan. 2022, Art. no. 108308.

[28] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 4, pp. 656–667, Apr. 1994.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and Ł. Kaiser, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.



YUN-ZHANG HU received the B.Eng. degree from North Minzu University, in 2020. He is currently pursuing the master's degree with the School of Computer and Information Engineering, Shanghai Institute of Technology. His research interests include distributed storage and intelligent computing.

HUI WANG received the Ph.D. degree in theory and control engineering from the East China University of Science and Technology, in 2009. He was an Academic Visitor with the School of Engineering and Built Environment, Glasgow Caledonian University, from 2013 to 2014. He is currently an Associate Professor with the School of Computer Science and Information Engineering, Shanghai Institute of Technology. He has published over 30 research articles. His current research interests include intelligent computing and cloud computing. He is a member of ACM and the IEEE Computer Society.