Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Multivariate time series collaborative compression for monitoring systems in securing cloud-based digital twin

Zicong Miao[1†], Weize Li[1*†] and Xiaodong Pan[1]

Abstract

With the booming of cloud-based digital twin systems, monitoring key performance indicators has become crucial for ensuring system security and reliability. Due to the massive amount of monitoring data generated, data compression is necessary to save data transmission bandwidth and storage space. Although the existing research has proposed compression methods for multivariate time series (MTS), it is still a challenge to guarantee the correlation between data when compressing the MTS. This paper proposes an MTS Collaborative Compression (MTSCC) method based on the two-step compression scheme. First, shape-based clustering is implemented to group the MTS. Afterward, the compressed sensing is optimized to achieve collaborative compression of grouped data. Based on a real-world MTS dataset, the experimental results show that the proposed MTSCC can effectively preserve the complex temporal correlation between indicators while achieving efficient data compression, and the root mean squared error of correlation between the reconstructed and original data is only 0.0489 in the case of 30% compression ratio. Besides, it is verified that using the reconstructed data in the production environment has almost the same performance as using the original data.

Keywords Cloud monitoring, MTS, Shape-based clustering, Compressed sensing, Collaborative compression

Introduction

With the rapid development of supporting technologies, such as cloud computing, big data, and machine learning, digital twin (DT) has been moving progressively from concept to practice. The data interaction among physical structure, digital model, and human interventions are enhanced by applying cloud computing services [1]. Thus, the cloud-based DT systems operating on big data need to ensure extremely high security and stability in case a cloud service fails, which will not only lead to a decline in user experience but also affect the revenue of the cloud service provider. Amazon's cloud-service network suffered a major outage on December 7, 2021, which leads to disrupting access to many popular sites including many governments, universities, and companies [2]. On December 19, 2022, some services in the Alibaba Cloud Hong Kong region failed, making it impossible to access the websites hosted by key infrastructure operators, including the Macau Monetary Authority, Lotus Guardian, and Macau Daily [3], causing huge economic losses.

Due to the increasing scale and data complexity of cloud services, it is difficult to collect, store, analyze, and visualize data through traditional methods [4]. To effectively manage cloud service systems and large data centers, service providers need to monitor the processes of systems and applications that generate big data, and ensure the reliability of cloud services through real-time monitoring of cloud platforms [5]. Therefore, cloud service monitoring, such as safeguarding performance,

†Zicong Miao and Weize Li contributed equally to this work and co-first authors.

*Correspondence:
Weize Li
liweize@chinatelecom.cn
[1] China Telecom Cloud Computing Corporation, Beijing, China

detecting attacks, data center-wide profiling, state prediction, and anomaly detection [6, 7], is a crucial and essential part of cloud management.

To achieve effective anomaly monitoring, Internet service providers based on cloud computing will deploy monitoring programs at various software and hardware levels of the cloud service system. Thousands to millions of key performance indicators (KPI), such as CPU utilization, queries per second, and service response time are collected [8]. For example, unusual data peaks or abrupt drops in KPI data usually indicate abnormal events of related cloud services [9, 10]. In the actual cloud monitoring production environment, there are a large number of platform components that make up cloud services, and each component collects many indicators. For example, the MySQL database uses hundreds of monitoring indicators [11]. As the complexity of the cloud service system increases, a fault will cause anomalies in multiple indicators, which means there is a correlation between these indicators. The obvious challenge brought by the massive monitoring data is how to counteract the increase in data storage and analysis costs. Practical applications may not need all the original data, but only some effective and aggregated data for analysis. Therefore, a reliable, accurate, and efficient compression method that can preserve the data correlation needs to be applied urgently [12].

General compression methods are divided into lossy compression and lossless compression. Lossy compression methods improve compression ratio by deleting unimportant data, while lossless compression preserves accurate information of the data for reconstruction. Lossy compression is suitable for fault-tolerant scenarios such as image, audio, and video compression, using wavelet transform, compression autoencoder, and neural networks [13]. The lossless compression methods can achieve accurate reconstruction, and they do not incur any information loss. Common lossless compression methods include Lempel–Ziv-Welsh (LZW), Lempel–Ziv-Markov chain algorithm, and adaptive Huffman coding [14]. The Compressed Sensing (CS) algorithm can accurately reconstruct all data with a small amount of sample, it can reconstruct a signal robustly and stably from under sampled noise observations by exploiting the sparse characteristics of the signal [15]. This advantage means CS has the potential to achieve a higher compression ratio. That's why CS has been widely used in the research of time series data compression in recent years [16]. The CS algorithm can be divided into two parts, first, original data are subsampled in advance using a determined sensing matrix such as a Gaussian matrix to obtain the sampled data. Then, the compressed data and sensing matrix are used to recover the original data [17]. Unfortunately, it is observed that CS and other time series compression methods cannot preserve the correlation of the variables, because most existing methods mainly focus on univariate time series (UTS) [18, 19]. Some research has proposed compression methods for MTS but applies data compression to each variable individually while ignoring the correlation between variables [18, 20]. We cannot ignore the data correlation but chase the extreme compression efficiency. Moreover, cloud monitoring datasets do not require the exact waveform data, a reasonably accurate trend of the data is sufficient for anomaly detection. Hence, there exists a tradeoff between the extent of compression achievable and the correlation loss caused by it. To improve monitoring efficiency and utilize the correlation between multivariate data, it is an important and challenging task to design a collaborative compression method for MTS data.

Motivated by the above-mentioned works, this paper presents the MTS collaborative compression (MTSCC) algorithm in combination with the actual scenarios of cloud monitoring applications. According to our investigation, there is currently no research to perform MTS compression in combination with preserving the correlation between cloud monitoring time series data. Making a tradeoff between data correlation and implementing a high compression ratio is a major task for this article. The contributions of this paper are summarized as follows:

- A novel two-step collaborative compression for multivariate cloud monitoring dataset is proposed. The clustering is performed by assessing correlations based on shape differences between each time series in MTS, and then the CS method is applied to compress them temporally.
- We propose performance indices for the MTSCC to evaluate the errors of correlation and reconstruction data, and then carry out the empirical optimization of the operating parameters, including compression ratio concerning the data variability.
- We conduct extensive experimental evaluations using a real-world dataset. It shows that under the premise of 30% compression ratio, the root mean squared error of correlation between the reconstructed and original data is only 0.0489.
- Reconstruction accuracy of the proposed MTSCC for monitoring dataset is validated using an anomaly detection application, it is demonstrated that this approach does not have any adverse effect on the practical application.

The layout of this article is organized as follows. "Background and motivation" section briefly gives an overview of the background and discusses the motivation. "Proposed algorithm" section introduces the

proposed two-step collaborative compression scheme and algorithm. We evaluate our approach in "Experiments and results" section on MTS datasets and give the concludes in "Conclusion and future work" section.

Background and motivation

Cloud monitoring indicators

As a measurement basis for evaluating the performance and stability of cloud computing systems, cloud monitoring indicators are numerical information of monitoring objects collected or aggregated at predefined time intervals (such as 5s or 30s), therefore, they are essentially time series data, with each data point consisting of a timestamp, measured value, and metric name [16, 21]. Monitoring data of cloud services contain thousands or even millions of indicators, each of which consists of tens of thousands of parameters, and to record the complete pattern of monitoring objects, the period of the data varies from a few days to several weeks [8]. The cloud monitoring indicators can be divided into the following three types according to the three levels (IaaS, PaaS, SaaS) of cloud services:

- Infrastructure layer indicators: This layer provides monitoring indicators for cloud infrastructure, typically including operational parameters of servers, network devices, etc., such as CPU utilization, memory utilization, disk I/O, network bandwidth utilization, network latency, and container and virtual machine performance indices. The fluctuation of the server indicators has no periodicity, once the anomaly occurs, the fluctuation will be significant.
- Platform layer indicators: This layer provides monitoring indicators for development platforms on the cloud infrastructure, used to record the running state or performance of components such as big data platforms, middleware, databases, security platforms, etc., such as the number of database requests, response time, application log information, dynamic expansion and load balancing and other key functional indicators. There are numerous types and quantities of monitoring indicators in the PaaS layer.
- Software service layer indicators: This layer provides monitoring indicators of cloud applications and services, including service request frequency and response time, user availability and quantity, application startup and termination time, etc. The monitoring indicators of the software layer generally reflect users' behavior, and the number of requests for services by all users is statistically periodic over time [22].

In summary, monitoring indicators of cloud platforms have the characteristics of numerous types and quantities of indicators, large amounts of data, and high component dependencies between different levels. Anomaly detection of MTS is often required for a single fault.

Multivariate time series monitoring data

The timestamps and values of a single indicator form a UTS, to comprehensively obtain the operational pattern and state of cloud services, the monitoring system continuously collects monitoring data from various levels and indicators. Therefore, the monitoring of a cloud service system often consists of hundreds of UTS, forming MTS. The operation and maintenance personnel usually focus on the operational state of the entire system, rather than the state of a specific indicator, but research shows that there is no single indicator that can capture all system performance issues [23]. Therefore, the MTS data composed of performance monitoring indicators of various layers and components of the cloud platform is gradually getting attention. Figure 1 shows the MTS composed of three indicators, which are used as real-time data input to detect whether there are anomalies in the entire system. The MTS data are highly correlated because of the monitoring variables' spatiotemporal correlation. The red regions in Fig. 1 indicate anomalies, and the three groups of UTS have experienced significant fluctuations.

There are usually two purposes in actual application scenarios for collected monitoring data of cloud service programs and equipment operation. The first one is online anomaly detection, the extremely high real-time and high data sampling frequency is needed for this type, therefore, data compression is not required in such scenarios. In addition to anomaly detection of real-time data, it is also necessary to save the collected data into the database to support the prediction of the state of the cloud platform and provide a basis for decision-making [24]. In this scenario, the real-time performance and integrity requirements of data are not strict, but the collected source data will occupy a large amount of cloud storage space, resulting in increased storage costs and reduced data retrieval efficiency, therefore, data compression will be used to alleviate storage pressure. The operation and maintenance personnel will pay more attention to the recent (within a week or a month) monitoring service program and equipment operating status data. The compressed data needs to retain some sharply changing inflection point data for analysis and judgment, and it should be ensured that the compressed data can be recovered through lazy loading. For relatively long-term stored data (more than six months), a compression method with a higher compression ratio will be selected

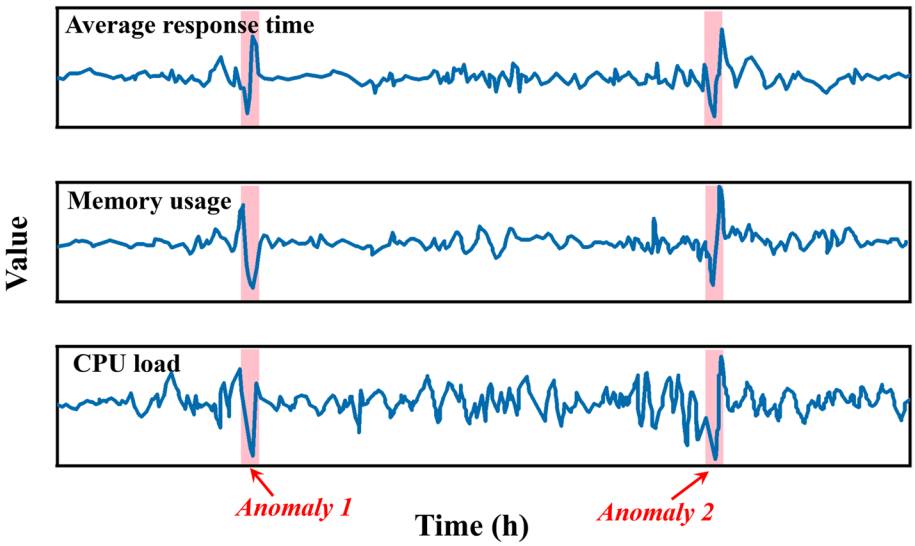

Fig. 1 The MTS consists of three sets of fragments over 24 h from the cloud online service system, two regions with anomalies are marked in red

to balance the actual application effect and storage pressure.

Why time series data compression

Time series data is gaining increasing popularity, rather than processing time series as streams and analyzing only once, the demand for storage of time series data for further analysis is becoming popular [25]. Time series data consists of many data points, and in all fields, timestamps and measurement values dominate storage consumption [21] The generation speed of time series data exceeds the growth of computing and storage capacity [26], and many allocation scenarios cannot afford enough computing resources including storage and network transmission bandwidth to meet the processing needs for time series data.

The sensors of connected vehicles produce about 30 TB of data per day, time series is one of the main components of the generated data, and vehicle manufacturers have to install large capacity disks to preserve the monitoring data. While a 30-TB disk costs nearly $1,200 [25], one month of running data can fill a 960-TB disk, which means the storage cost is more than $35,000. If a 30%-50% compression ratio can be achieved, it can reduce costs by at least $10,000. The same situation also occurs in the field of cloud monitoring, the daily increase in data volume of Kingsoft Cloud Log Service is 200 TB [27], although public clouds can store all data, however, taking Alibaba Public Cloud Hard Disk as an example, the unit price of ESSD PL1 cloud disk is ¥1.00/1 GiB/month [28], based on 6 months of monitoring data retention, the daily data storage cost incurred will exceed $160,000, and the data storage cost incurred for one month of monitoring data will be close to $5 million. Faced with huge storage costs, small operating entities have to limit their storage cost. Besides, most of the data collected in the vehicle network and energy industry is repetitive and low-value density [25], so it is not worth paying such high storage fees for those low-value density data. Therefore, it is necessary to compress time series within a reasonable range of computing resources and time. Although existing data compression methods can be used for UTS compression, when facing MTS compression, the existing methods cannot guarantee the temporal correlation between multivariate indicators, which leads to the compressed temporal data losing or misplacing the timestamp, thereby affecting the effectiveness of reconstructed data in practical applications. We still need to develop more effective compression approaches for MTS when considering the compression ratio and the correlation.

Motivation and challenge

From the current state-of-the-art, most existing time series data compression methods only consider a single variable, and even if some studies propose compression methods for MTS, the time series data is compressed independently during compression, which does not achieve collaborative compression, making it difficult to ensure the correlation of MTS. In the actual monitoring environment of cloud platforms, multiple variables such as received packets, TCP network indicators, CPU metrics, and memory usage are sensed and transmitted over a communication link to the storage serve. When facing dozens or even hundreds of monitoring indicators, the

first step is to design an efficient data grouping method to ensure that monitoring data with strong correlation is compressed as a group as much as possible. At the same time, due to the coherence and density of cloud monitoring data, when a monitoring indicator is sampled at 1-min intervals, it can exceed 10,000 data points in a week. Therefore, how to compress at a lower sampling rate (achieve a high compression ratio) while ensuring data reconstruction ability is another problem that this article aims to solve.

Proposed algorithm

Two-step collaborative compression scheme

This section proposes the two-step mechanism for MTS data compression. The collaborative compression method proposed in this article considers multiple related variables simultaneously, by modeling and utilizing the data correlation of MTS, we can achieve a better compression effect while preserving the data correlation. Figure 2 shows the two-step scheme in the data compression processing unit. In the first step, we adopt a shape-based distance, k-shape clustering method [29] to classify cloud monitoring data, to preserve the correlation between the different variables. In the second step, we compress the clustered data using CS to reduce the storage space of the data. The data reconstruction processing unit is applied in practical scenarios and method validation, the CS reconstruction method can preserve data correlation, and we can group the reconstructed data according to the kept data correlation. These techniques mentioned are briefly discussed below.

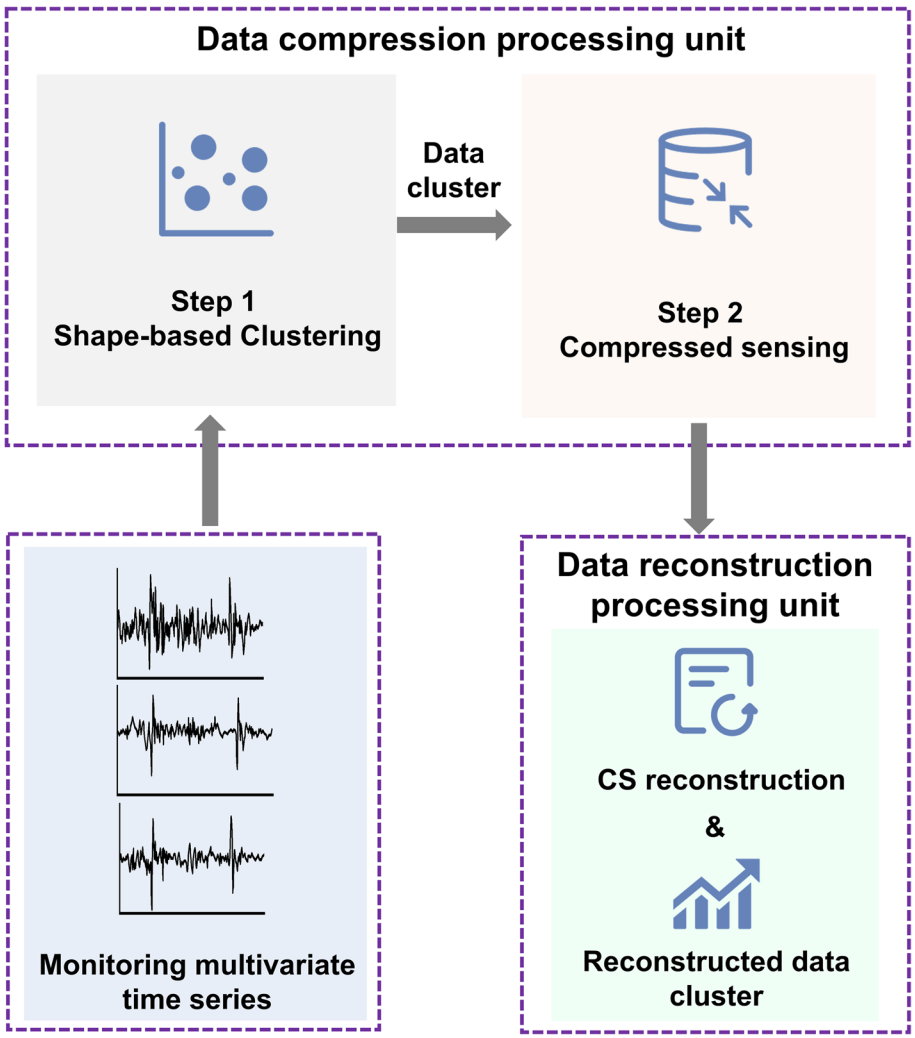

Fig. 2 Two-step scheme consists of shape-based clustering and compressed sensing which is shown in the data compression processing unit, and data reconstruction unit used for validation and practical applications

Shape-based clustering

Clustering is the general problem of partitioning n observations into k clusters, where $k \le n$. The evaluation index selected by k-shape when measuring the proximity of two variables is based on the shape-based distance (SBD), and the cross-correlation is obtained by calculating the SBD between the indexes [29]. Compared with other algorithms such as dynamic time warping (DTW, cDTW), the k-shape can obtain higher computational efficiency when dealing with high-dimensional time series. To calculate the similarity of time series data $\overrightarrow{x} = (x_1, \cdots, x_m)$ and $\overrightarrow{y} = (y_1, \cdots, y_m)$ with the length of m, cross-correlation keeps $\overrightarrow{y}$ static and shifts s of $\overrightarrow{x}$ each time to compute their inner product. We define a cross-correlation sequence $CC_\omega(\overrightarrow{x}, \overrightarrow{y}) = (c_1, \cdots, c_\omega)$, where ω is the possibility of all moves with the length of *2m-1*, defined as follows:

$$CC_\omega(\overrightarrow{x}, \overrightarrow{y}) = R_{\omega-m}(\overrightarrow{x}, \overrightarrow{y}), \omega\epsilon\{1, 2, \cdots, 2m-1\} \tag{1}$$

where $R_{\omega-m}(\overrightarrow{x}, \overrightarrow{y})$ is calculated as follows:

$$R_k(\overrightarrow{x}, \overrightarrow{y}) = \begin{cases} \sum_{l=1}^{m-k} x_{l+k} \cdot y_1, k \ge 0 \\ R_{-k}(\overrightarrow{y}, \overrightarrow{x}), k < 0 \end{cases} \tag{2}$$

Our target is to obtain the ω corresponding to the maximum value of $CC_\omega(\overrightarrow{x}, \overrightarrow{y})$, at this time, ω represents the maximum similarity between $\overrightarrow{x}$ and $\overrightarrow{y}$ after shifting, and the optimal shift is $\overrightarrow{x}_{(s)}$, where $s = \omega - m$. The shape-based distance can be obtained from the cross-correlation metric:

$$SBD(\vec{x}, \vec{y}) = 1 - \max_\omega \left(\frac{CC_\omega(\vec{x}, \vec{y})}{\sqrt{R_0(\vec{x}, \vec{x}) \cdot R_0(\vec{y}, \vec{y})}} \right) \tag{3}$$

The more overlap between the two sequences, the larger the $CC_\omega(\overrightarrow{x}, \overrightarrow{y})$ value, the more similar the shape is, the smaller the distance measure value $SBD(\overrightarrow{x}, \overrightarrow{y})$ is, and then the similarity between the sequences can be obtained. In the process of the k-shape algorithm to obtain the cluster center, the purpose is to find a time series that minimizes the sum of squared distances $\overrightarrow{\mu}_k^*$ between this time series and all other time series in the cluster, and its optimization objective function is:

$$\vec{\mu}_k^* = \arg\max_{\vec{\mu}_k} \sum_{\vec{x}_i \in P_k} \left(\frac{\max_\omega CC_\omega(\vec{x}_i, \vec{\mu}_k)}{\sqrt{R_0(\vec{x}_i, \vec{x}_i) \cdot R_0(\vec{\mu}_k, \vec{\mu}_k)}} \right)^2 \tag{4}$$

where P_k represents the kth cluster, and $\overrightarrow{\mu}_k$ is the cluster center. After cluster analysis, the MTS data is divided into multiple clusters, and the indicators in each cluster are associated with the physical meaning of the corresponding monitoring indicators. Finally, indicators with similar physical meanings can be aggregated together to ensure variable correlation.

Compressed sensing

CS breaks through the limitations of the Nyquist-Shannon sampling theorem, it can achieve perfect signal reconstruction with a smaller sampling, and directly acquire a condensed representation without losing much on the monitoring information [14]. Therefore, it is especially suitable for the perception and collection of large-scale data scenarios such as communication and cloud computing. For the monitoring data of the cloud platform, the process of compressing a set of signals $x \in \mathbb{R}^M$ into an $\tilde{M}$ dimensional measurement value $y \in \mathbb{R}^{\tilde{M}}$ could be expressed as $y = \Phi x$, where $\Phi \in \mathbb{R}^{\tilde{M} \times M}$ is the measurement matrix. Signal sparsity plays a crucial role in compressed sensing, if there is an orthogonal basis Ψ that transforms the signal x into a sparse domain, the monitoring signal can be expressed as $x = \Psi\alpha$. If α has only K non-zero elements, then x is said to be K-sparse in the sparse field $\Psi \in \mathbb{R}^{M \times M}$, which also means that the signal is compressible, thus,

$$y = \Phi x = \Phi\Psi\alpha = \Theta\alpha \tag{5}$$

Where $\Theta = \Phi\Psi$ is called the sensing matrix, only when $\Theta \in \mathbb{R}^{\tilde{M} \times M}$ satisfies the constrained isometric property (RIP) condition can it be ensured that formula (5) has only one K-sparse solution. In order to reconstruct the target sparse signal $\alpha \in \mathbb{R}^M$ from the compressed vector $y \in \mathbb{R}^{\tilde{M}}$, a K-sparse solution of the uncertainty Eq. (5) can be reconstructed through the ℓ_0 minimization, then the process can be expressed as solving the following problem:

$$\hat{\alpha} = \arg\min \|\alpha\|_0 \ \text{ s.t. } y = \Phi\Psi\alpha = \Theta\alpha \tag{6}$$

But sparse matrix reconstruction is NP-complete, and traditional CS reconstruction algorithms are mainly based on methods such as greedy pursuit and convex optimization. Greedy pursuit algorithms like orthogonal matching pursuit (OMP) [30] and basic pursuit (BP) [31] are simple and efficient, but these algorithms do not update the signal support set at each iteration, which may affect the reconstruction probability. The convex optimization relaxes ℓ_0 norm minimization to ℓ_1 norm minimization, and improves prediction accuracy and interpretability through variable selection and regularization. In this paper, the effective convex optimization toolset is selected to calculate the minimum value of the ℓ_1 norm.

Multivariate Time Series Collaborative Compression (MTSCC)

Based on the proposed two-step compression scheme, this section introduces the specific implementation algorithm MTSCC. As shown in Fig. 3, the designed approach consists of the data compression processing unit and data reconstructed processing unit. CS reconstruction of a large number of indicators is inaccurate due to multiple indicators containing single indicators with different shapes, and CS is not suitable for reconstruction in this condition. However, it cannot capture the correlation between univariate indicators by disassembling them into univariate indicators and reconstructing them one by one. Therefore, in the data compression processing unit, the shape-based clustering algorithm is first applied to cluster multiple indicators, and then the sparsity in the same clustered group is used for further compression. This is mainly because indicators in the same group exhibit similar shapes, and the correlation between data is determined by comparing the shape differences between indicators. That's why MTSCC is different from the previous methods. After being processed by the shape-based clustering algorithm, it preserves the correlation between MTS data to a great extent.

The proposed approach is adaptive because the shape-based distance measure in k-shape (comparing the similarity between data) is estimated at runtime. Let monitoring batch data $X_t \in \mathbb{R}^{M \times N}$ be the input MTS matrix, where N is the number of the dimensions, which is the variables measured by the cloud monitoring tools, and M is the number of the samples in each variable. The proposed approach here will further be validated in "Experiments and results" section.

Experiments and results

Experimental design

Datasets used for evaluation

The proposed approach is applied to a Key Indicators Dataset (KID), which is a 4-week-long dataset collected by a monitoring system. This dataset contains 6 indicators, and all of them along with the sampling interval are listed in Table 1.

Where *mem_usage* refers to memory usage; *tcp_insegs* refers to the total number of message segments received; *cpu_sintr* refers to the number of interrupts in the system; *cpu_load* refers to the load of the CPU, which is used to measure how busy the CPU is; *cpu_ctxt* is used to measure the frequency of context switching, the higher the number of switching times, the more frequent task switching; *tcp_outsegs* refers to the total number of message segments sent.

Table 1 Datasets summary

Indicators	Univariate Time Series	Sampling interval and period
Indicator-01	mem_usage	1 min, 4 weeks
Indicator -02	tcp_insegs	
Indicator -03	cpu_sintr	
Indicator -04	cpu_load	
Indicator -05	cpu_ctxt	
Indicator -06	tcp_outsegs	

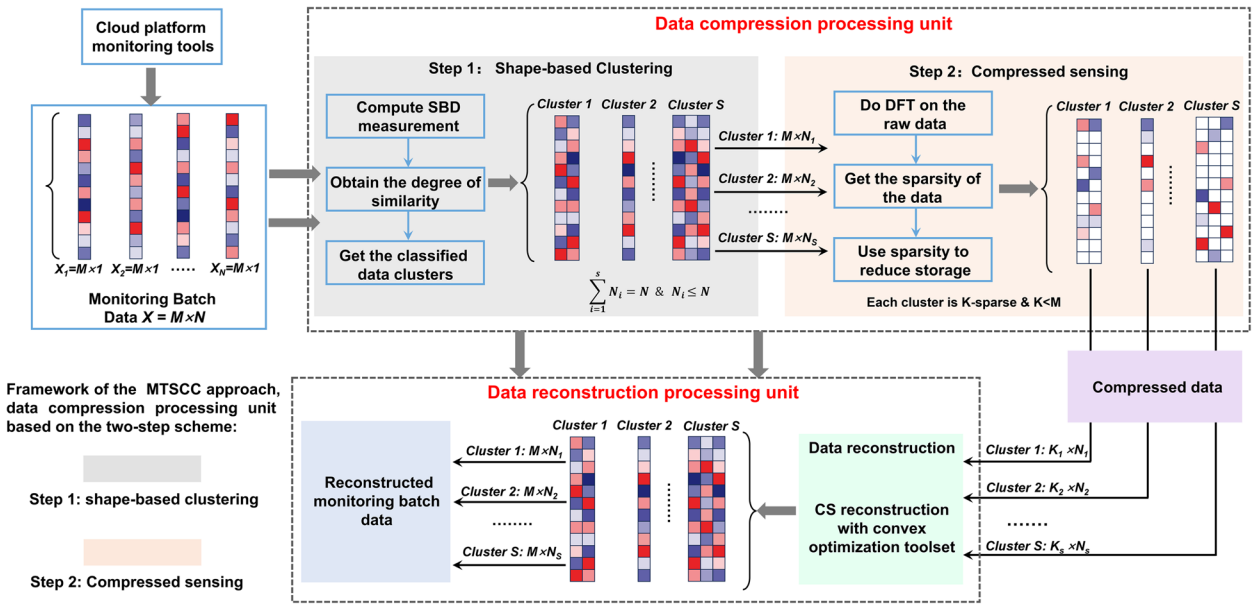

Fig. 3 Framework of the proposed MTSCC algorithm

Performance parameters

The parameters used for the performance evaluation of the MTSCC are described here.

All the variables used here carry the same meaning as mentioned in "Proposed algorithm" section unless explicitly mentioned.

PCC We use the Person correlation coefficient (PCC) to measure the degree of linear correlation between two sets of data. The larger the absolute value of the correlation coefficient, the higher the possibility that the two samples belong to the same class. The PCC r between two samples s_i and p_i is expressed as:

$$r = \frac{\sum_{i=1}^{M}(s_i - \bar{s})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^{M}(s_i - \bar{s})^2}\sqrt{\sum_{i=1}^{M}(p_i - \bar{p})^2}} \tag{7}$$

where $\bar{s} = \frac{1}{M}\sum_{i=1}^{M} s_i, \bar{p} = \frac{1}{M}\sum_{i=1}^{M} p_i$, and M is the number of samples of each sample.

RMSE The errors of correlation and reconstruction induced in the two-step algorithm process are measured in terms of root mean squared error (RMSE), which is calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M}(x_i - \hat{x}_i)^2}{M}} \tag{8}$$

where M is the number of samples of each univariate time series data. We define a cRMSE as the RMSE of the PCC, where x_i and $\hat{x}_i$ are respectively the PCC of actual and the PCC of reconstructed data. Similarly, a dRMSE is defined as the RMSE of the data, where x_i and $\hat{x}_i$ are actual and reconstructed data of the variable under consideration. The larger the absolute value of the RMSE, the greater the error is. The efficiency of the proposed method is tested by calculating the cRMSE and dRMSE.

Implementation method

We compare the MTSCC method with Random Group-based Multivariate Time Series Compression (RG-MTSC), which uses random grouping instead of shape-based clustering in the first step of a two-step scheme. It is necessary to point out that the number of possible random grouping ways of n variables is a Bell number. For example, when the number of variables is 14, there are 190,899,322 ways to group. When n is large, the actual experiment is extremely difficult to traverse all the groups, requiring huge computing resources and calculating time.

For the KID, the number of variables is 6, and there are 203 ways of grouping them, so it is possible to exhaustively traverse and run the experiment in a limited time. Among 203 grouping ways, the group obtained by the shape-based clustering method is called the base group, as shown in Table 2. It can be seen from Fig. 9, which shows the waveforms of indicator-01 to indicator-06, shape-based clustering can intuitively aggregate indicators with similar shapes that match the grouping result in Table 2. Such grouping results also conform to the physical correlation between the variables. In addition to the base group, the other 202 groups are called random groups, which are used in the first step of RG-MTSC.

We perform experiments with the KID to analyze the impact of the compression ratio on the cRMSE and dRMSE of the MTSCC, and then compare and analyze the differences between the MTSCC and RG-MTSC on cRMSE and dRMSE based on a fixed compression ratio, respectively. It should be noted that the original data in all experiments were normalized before use in the two-step scheme.

Impact analysis of compression ratio

We dynamically adjusted the compression ratio in MTSCC and 12 random groups based on RG-MTSC at an interval of 0.02 and analyzed the impact of different compression rates on the cRMSE and dRMSE. Figure 4a shows the variation of cRMSE with the change of compression ratio, from which it can be seen intuitively that when the compression ratio is not higher than 30%, the difference in cRMSE between MTSCC and RG-MTSC was not significant. When the compression ratio exceeds 30%, the cRMSE fluctuates and the difference will gradually become large and indeterminate.

Figure 4b shows the variation of dRMSE with the change of compression ratio. It can be seen that when the compression ratio is lower than 80%, the dRMSE and the compression ratio are linearly positively correlated. When the compression ratio exceeds 80%, the curves of dRMSE fluctuate. To reduce the impact of cRMSE fluctuations and better evaluate the MTSCC and RG-MTSC, we chose a compression rate of 30%. Based on this

Table 2 Group result of shape-based clustering

	Indicators
Base Group	Indicator-01
	Indicator-02, Indicator-06
	Indicator-03, Indicator-04, Indicator-05

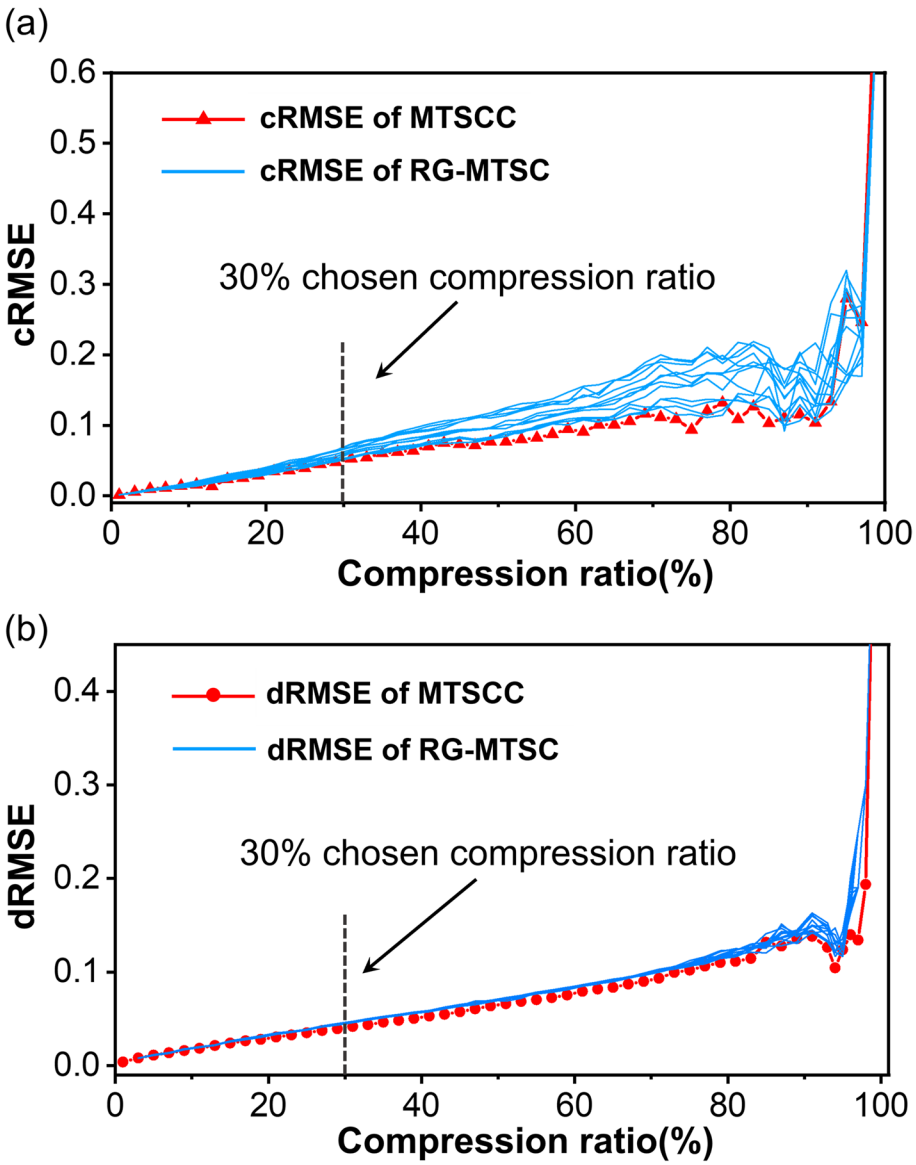

Fig. 4 Effect of compression ratio on (**a**) cRMSE and (**b**) dRMSE

parameter setting, we further evaluate the performance difference between the MTSCC and RG-MTSC.

Comparison of cRMSE

The comparison result of the cRMSE between the MTSCC and RG-MTSC is shown in Fig. 5, the red triangles represent the grouping result based on RG-MTSC, while the pentagram represents the grouping results based on MTSCC. Among all grouping ways, the cRMSE of the base group based on the MTSCC method is better than 96.04% of the random groups based on RG-MTSC, which indicates that the proposed method can better preserve the correlation of data.

The further demonstration in Fig. 6 shows the percentage of the increase of the cRMSE difference between the MTSCC and 12 random groups based on RG-MTSC, we set the cRMSE based on MTSCC grouping to "baseline" which is colored in blue. We can find that the cRMSE based on RG-MTSC increases by at least 9.3% and an average of 28.7%. When the number of variables is large and it is impossible to traverse all the groups to find the optimal group, the MTSCC method can better guarantee the variable correlation after compression than randomly selecting a group.

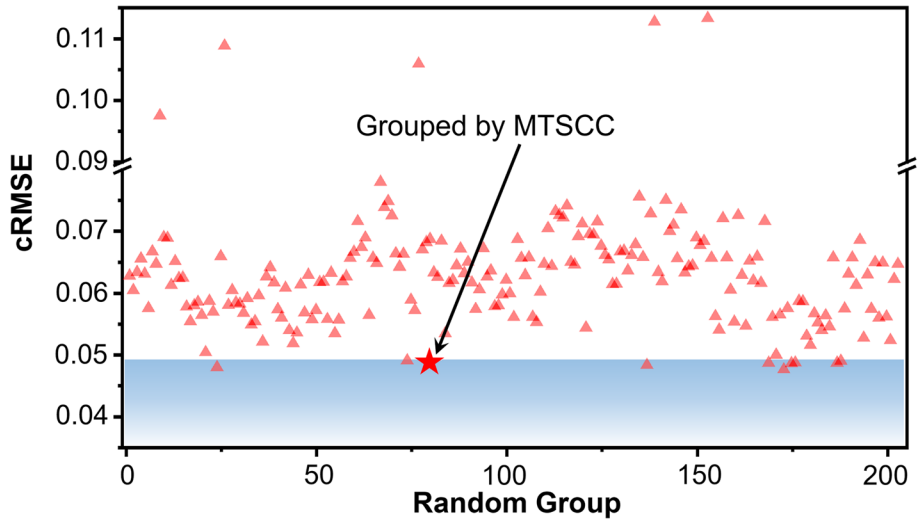

Fig. 5 Comparison of the cRMSE between the MTSCC and RG-MTSC

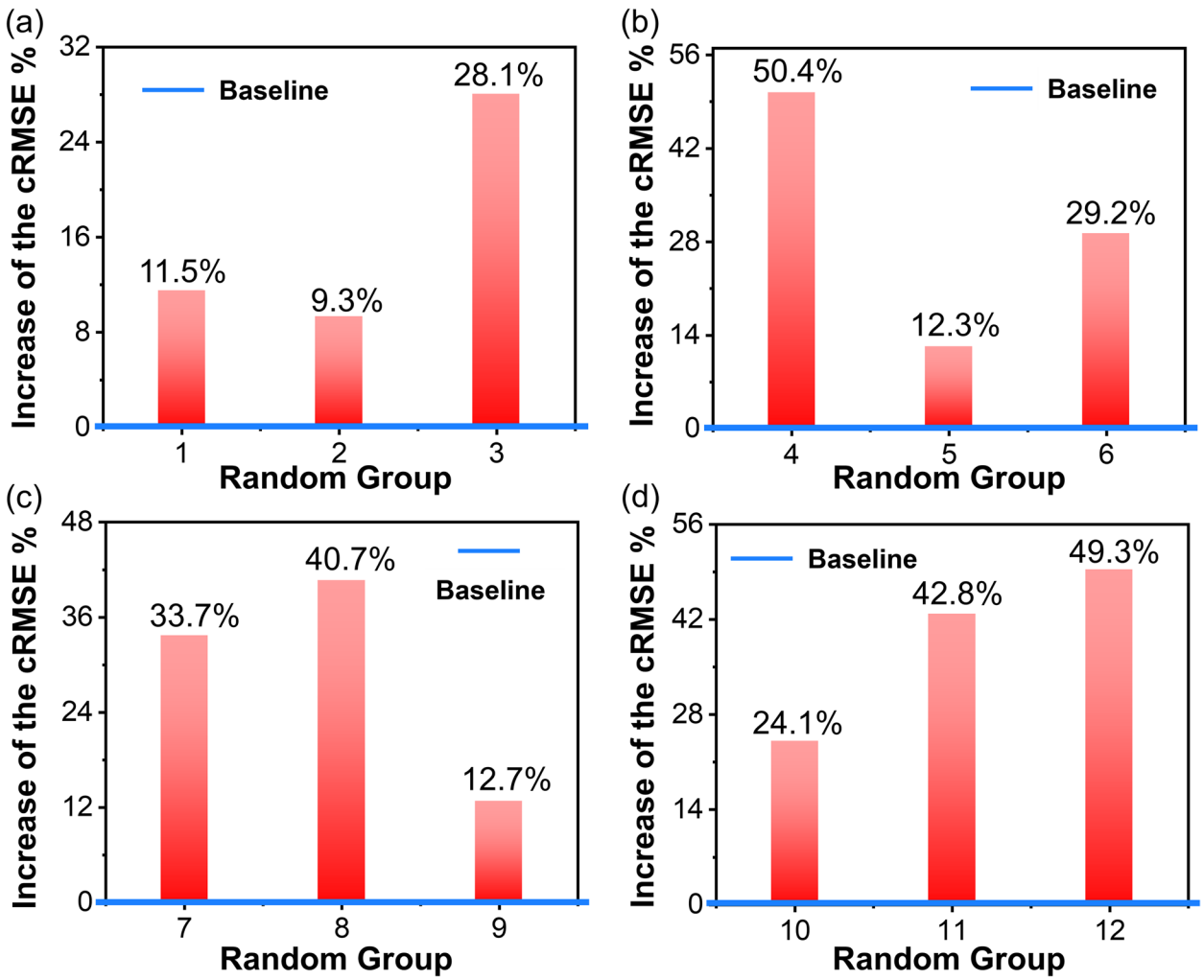

Fig. 6 Percentage increase of the cRMSE between the MTSCC and 12 random groups-based RG-MTSC

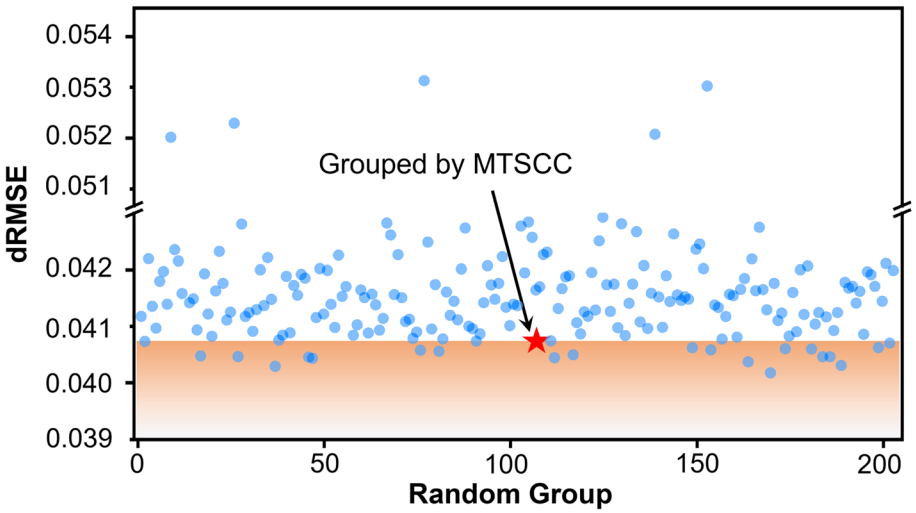

Fig. 7 Comparison of the dRMSE between the MTSCC and RG-MTSC

Comparison of dRMSE

The comparison results of the dRMSE between the MTSCC and RG-MTSC are shown in Fig. 7, the blue circles represent the grouping result based on RG-MTSC. Similar to cRMSE, the dRMSE of the MTSCC method is better than 90.10% of the random groups based on RG-MTSC. Figure 8 further shows the percentage increase of the dRMSE difference between the MTSCC and 12 random groups based on RG-MTSC, we set the dRMSE based on MTSCC grouping to "baseline" which is colored in red. The dRMSE based on RG-MTSC increases by at most 3.86% and an average of 1.9%. Compared with the randomness of the dRMSE of the RG-MTSC, although the MTSCC cannot guarantee the minimum error between the reconstructed data and the original data, it can guarantee a relatively better result.

A result comparison of the original data streams and reconstructed data streams after compression are presented respectively in Fig. 9, the black lines represent the original data and the red lines represent the reconstructed data. For each of the 6 indicators, we intercept a part of the time series data for amplification to demonstrate the reconstruction results. It is observed that the reconstructed data nearly overlap with the original data of the 6 indicators, owing to the high reconstruction accuracy.

Comparison with AMDC

The MTS compression performance of the proposed MTSCC algorithm is compared with that of the recently proposed AMDC algorithm, which we implemented based on the description available at [18]. We mainly compared the reconstruction accuracy and data correlation retention of MTS under the same compression ratio of 30%. Different from the MTSCC proposed in this article, AMDC pays more attention to the dimensionality reduction of MTS data to facilitate subsequent data compression processing, thus ignoring the correlation between MTS data. This is mainly because the Principal Component Analysis (PCA) method is used in the AMDC to reduce the dimensionality of the data. Although during the dimensionality reduction process, PCA is used to retain as much information and correlation of the original data as possible, due to the principal components being linear combinations of the original data, nonlinear relationships and higher-order relationships cannot be preserved. The shape-based clustering algorithm can find the inherent structural differences between data points and try to make the data points in the same cluster related to each other. Therefore, the clustering algorithm can usually better satisfy the requirements in terms of retaining the correlation between MTS, and the subsequent experimental results also prove this.

The dataset is divided into 20 groups and we conduct 20 comparative tests, as shown in Fig. 10, which is a comparison of dRMSE using the MTSCC and AMDC methods. Figure 10a shows 3 sets of data randomly selected from 20 sets of comparative tests, while Fig. 10b shows the statistical comparison of 20 test results. From Fig. 10, it can be found that both algorithms have good accuracy in data reconstruction, and the median and average dRMSE of the two methods are very close. Figure 11 shows the comparison of the cRMSE between the MTSCC and AMDC. We can find from Fig. 11a that the cRMSE based on AMDC increases by at least 22.7% and

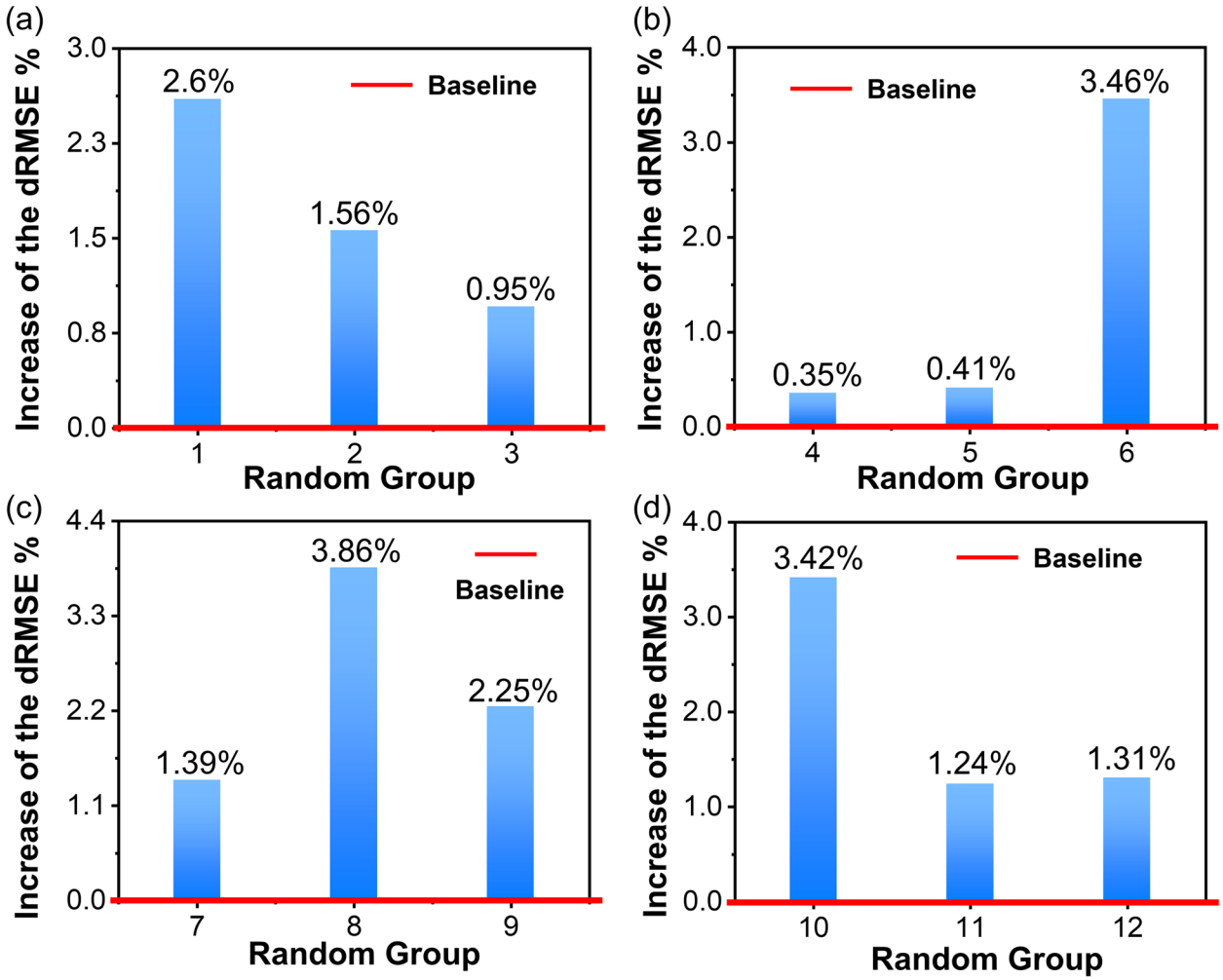

Fig. 8 Percentage increase of the dRMSE between the MTSCC and 12 random groups-based RG-MTSC

an average of 25.5%. From Fig. 11b, the median and average cRMSE based on AMDC are close to or even exceed 0.061, while the median and average cRMSE of the proposed MTSCC are close to 0.045. This illustrates that when the compression ratio is the same, MTSCC and the current state-of-the-art MTS compression method AMDC have comparable performance in data compression-reconstruction accuracy, but MTSCC has better performance in preserving the correlation.

Validation of reconstruction using a real application

In the previous section, the performance of the proposed MTSCC is validated through experiments. In this section, we demonstrate through a practical example to illustrate the proposed approach can be applied in actual production environments without adverse effects on user experience. The monitoring data of the cloud platform is mainly used for anomaly detection, since the MTS contains the complex temporal correlation between single indicators, it can alert regional anomalies. We use the existing MTS anomaly detection algorithm [23] to detect the three sets of MTS in Table 1. The anomaly detection algorithm is an unsupervised deep learning method. In this paper, the original data and reconstructed data are divided into a training set and a test set, respectively. Precision, recall, and F1-score are used to evaluate the effectiveness of the anomaly detection algorithm. The calculation formula for the F1-score is as shown in Eq. 9

$$F1-score=2\times\frac{precision\times recall}{precision+recall} \tag{9}$$

As shown in Table 3, the comprehensive index F1-score of the anomaly detection performance of the original data and the reconstructed data using the MTSCC is listed. The maximum deviation of the F1-score caused by using reconstructed data is 0.05%,

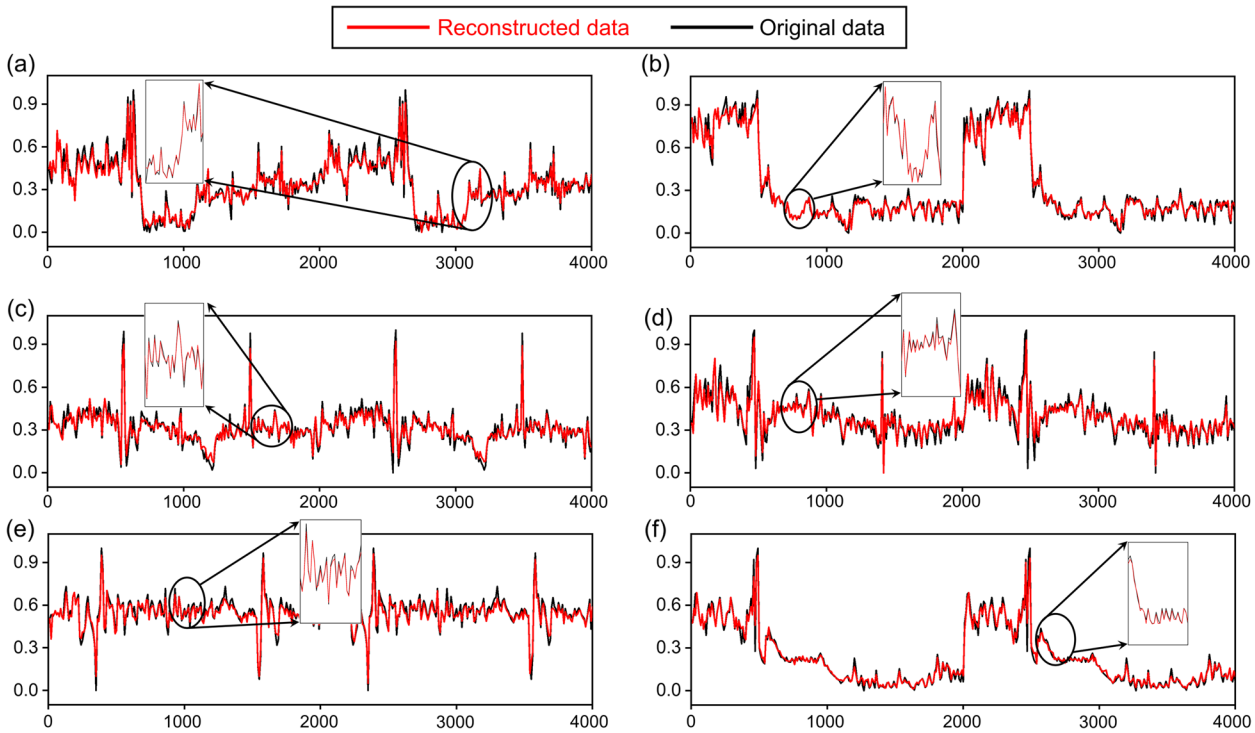

Fig. 9 Comparison of reconstructed data and original data of 6 indicators, all data are normalized, (**a**) to (**f**) corresponding to indicator-01 to indicator-06, indicator-01 in a group; indicator-02 and indicator-06 are in a group; indicator-03 to indicator-05 are in a group

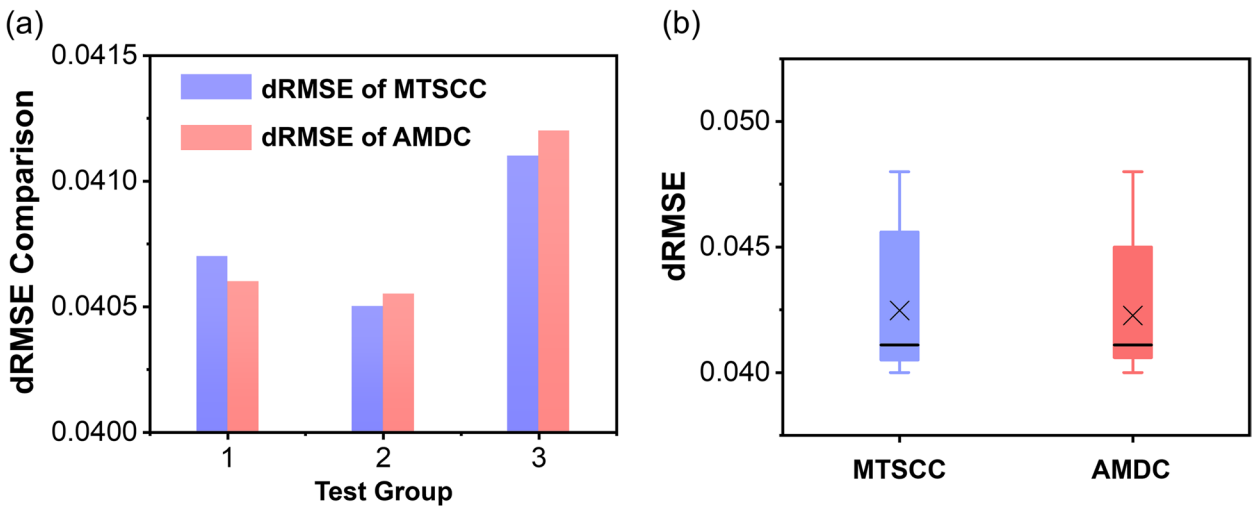

Fig. 10 **a** 3 groups of random comparison results of dRMSE; **b** Statistical results of 20 groups of dRMSE comparison tests

which is quite acceptable. This result is benefited from the adoption of shape-based clustering and CS-based collaborative compression, it makes that data with the same physical meaning has been co-processed during data compression and reconstruction, and can maintain the temporal correlation of the same group of data as much as possible, it provides interpretability for the analysis of anomaly detection results. This result strengthens the claim that MTSCC reduces the space of data storage and can ensure the correlation of data in the process of reconstruction, and the loss of information is practically negligible.

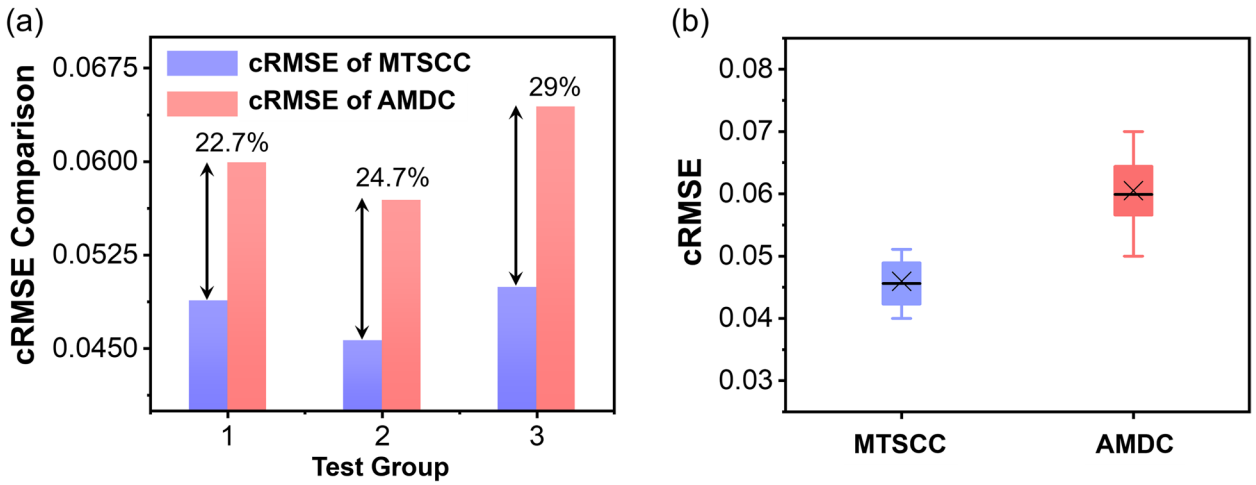

Fig. 11 **a** 3 groups of random comparison results of cRMSE; **b** Statistical results of 20 groups of cRMSE comparison tests

Table 3 F1-score of anomaly detection with original and reconstructed data

Multivariate time series	F1-score (%) with original data	F1-score (%) with reconstructed data
Indicator -01	86.55	86.52
Indicator -02, Indicator -06	89.23	89.26
Indicator -03, Indicator -04, Indicator -05	87.46	87.42

Conclusion and future work

In this paper, a two-step scheme which is called MTSCC is proposed for the collaborative compression of cloud monitoring MTS data. The proposed approach adopts a shape-based clustering algorithm to obtain variable grouping in the first step and uses grouped data as input for CS in the second step. The experimental results show that the MTSCC is superior to the random groups-based RG-MTSC by 96.04% in terms of variable correlation. At the same time, the dRMSE of the MTSCC method was 0.0407 under the premise of a 30% compression ratio, which was 90.10% better than that of the random groups-based RG-MTSC. In addition, we have proved through a practical application that the data reconstructed by the approach can be used for cloud service anomaly detection without affecting user experience. In future work, we will study the operation performance of the MTSCC in more detail, and further reduce the cRMSE and dRMSE while ensuring high operation efficiency. Meanwhile, we will explore the potential applications of MTSCC, such as in cloud storage, the MTSCC method can be used to compress the MTS data before storing it in the cloud, and the MTS data can be compressed while preserving the complex temporal correlation between indicators. Besides, MTS data is often used in machine learning tasks such as predictive modeling or anomaly detection. However, dealing with large-scale MTS data can be computationally expensive. To address this, the MTSCC method can be applied to compress the MTS data before feeding it into machine learning algorithms.

Acknowledgements

This paper is supported by China Telecom Cloud Computing Corporation, the authors would like to appreciate the helpful discussions provided by Dr. Li Deng from Wuhan University of Science and Technology during the writing phase of this work.

Authors' contributions

All authors contributed to the conception, design, and writing of the research project. Zicong Miao led the development of the method and wrote the main manuscript. Weize Li implemented key algorithms conducted experiments and analyzed the results. Xiaodong Pan collected and preprocessed the data, and supervised the experiments. All authors approved the final version of the manuscript.

Funding

The authors received no financial support for the research, authorship or publication of this article.

Declarations

Competing interests

The authors declare no competing interests.

Received: 24 October 2023 Accepted: 21 December 2023
Published online: 10 January 2024

References

1. Lu Q et al (2020) Digital twin-enabled anomaly detection for built asset monitoring in operation and maintenance. Autom Constr 118:103277
2. Amazon. Major outage hits Amazon Web Services.https://www.cbsnews.com/news/amazon-web-services-major-outage-many-sites-affected/?intcid=CNM-00-10abd1h. Accessed 15 July 2023

3. He X. Alibaba cloud breakdown affects Hong Kong and Macau. https://www.guancha.cn/economy/2022_12_19_671980.shtml. Accessed 15 July 2023
4. Rabkin A, Katz R (2010) Chukwa: a system for reliable {Large-Scale} log collection. 24th Large Installation System Administration conference (LISA 10)
5. Zhang X et al (2019) Cross-dataset time series anomaly detection for cloud systems. 2019 USENIX Annual Technical Conference (USENIX ATC 19)
6. Raschid L et al (2003) Monitoring the performance of wide area applications using latency profiles. WWW (Posters)
7. Gu G et al (2008) Botminer: clustering analysis of network traffic for protocol-and structure-independent botnet detection p. 139
8. Li Z et al (2018) Robust and rapid clustering of kpis for large-scale anomaly detection. 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). IEEE, Banff
9. Liu D et al (2015) Opprentice: towards practical and automatic anomaly detection through machine learning. Proceedings of the 2015 internet measurement conference
10. Zhang S et al (2015) Rapid and robust impact assessment of software changes in large internet-based services. Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies
11. Yoon DY, Niu N, Mozafari B (2016) Dbsherlock: a performance diagnostic tool for transactional databases. Proceedings of the 2016 international conference on management of data
12. Shaofei S et al (2021) A reliable data compression scheme in sensor-cloud systems based on edge computing. IEEE Access 9:49007–49015
13. Li C, Zheng R (2019) Load data compression based on integrated neural network model. 2019 Chinese Control And Decision Conference (CCDC). IEEE, Nanchang
14. Ringwelski M et al (2012) The hitchhiker's guide to choosing the compression algorithm for your smart meter data. 2012 IEEE International Energy Conference and Exhibition (ENERGYCON). IEEE, Florence
15. Li S et al (2023) Time series phase unwrapping algorithm using LP-norm optimization compressive sensing. Int J Appl Earth Observ Geoinform 117:103182
16. Ma M et al (2021) {Jump-Starting} multivariate time series anomaly detection for online service systems. 2021 USENIX Annual Technical Conference (USENIX ATC 21)
17. Si J et al (2022) Reconstruction of financial time series data based on compressed sensing. Finance Res Lett 47:102625
18 Chowdhury MR, Tripathi S, De S (2020) Adaptive multivariate data compression in smart metering internet of things. IEEE Trans Industr Inform 17(2):1287–1297
19. Feng H et al (2023) Spatiotemporal prediction based on feature classification for multivariate floating-point time series lossy compression. Big Data Res 32:100377
20. de Souza JC, Assis TM, Pal BC (2015) Data compression in smart distribution systems via singular value decomposition. IEEE Trans Smart Grid 8(1):275–284
21. Yu X et al (2020) Two-level data compression using machine learning in time series database. 2020 IEEE 36th International Conference on Data Engineering (ICDE). IEEE, Dallas
22. Xu H et al (2018) Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. Proceedings of the 2018 world wide web conference
23. Su Y et al (2019) Robust anomaly detection for multivariate time series through stochastic recurrent neural network. Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining
24. Rashid MM et al (2020) A survey on behavioral pattern mining from sensor data in Internet of Things. IEEE Access 8:33318–33341
25. An Y et al (2022) {TVStore}: automatically bounding time series storage via {Time-Varying} compression. 20th USENIX Conference on File and Storage Technologies (FAST 22)
26. Schlossnagle T, Sheehy J, McCubbin C (2021) Always-on time-series database: keeping up where there's no way to catch up. Commun ACM 64(7):50–56
27. Liu B. Kingsoft cloud used Pulsar to handle TB-evel data: https://www.infoq.cn/article/m5nbipdr8bpdcjlu38lv. Accessed 20 July 2023
28. Alibaba Cloud. https://www.aliyun.com/price/product?spm=a2c4g.11186623.0.0.67013021VRN8ZE#/disk/detail/disk
29. Paparrizos J, Gravano L (2015) k-shape: efficient and accurate clustering of time series. Proceedings of the 2015 ACM SIGMOD international conference on management of data
30. Tropp JA, Gilbert AC (2007) Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans Inf Theory 53(12):4655–4666
31 Chen SS, Donoho DL, Saunders MA (2001) Atomic decomposition by basis pursuit. SIAM Rev 43(1):129–159

Publisher's Note