Compression for Time Series Databases using Independent and Principal Component Analysis

Arnak V. Poghosyan, Ashot N. Harutyunyan, and Naira M. Grigoryan
VMware
{apoghosyan;aharutyunyan;ngrigoryan}@vmware.com

***Abstract*—Reliable management of modern cloud computing infrastructures is unrealizable without monitoring and analysis of a huge number of system indicators (metrics) as time series data stored in big databases. Efficient storage and processing of collected historical data from all "objects" of those infrastructures are technology challenges for this Big Data application. We propose a data compression framework for databases of time series that applies correlation content of the data set. Specifically, the fundamental statistical concepts of independent component analysis (ICA) and principal component analysis (PCA) are employed to demonstrate the viability of the approach. We experimentally show significant compression rates for real data sets from IT systems.**

***Keywords*—** *time series database; lossy and lossless compression; independent and principal component analysis.*

I. INTRODUCTION

Modern cloud computing services are enabled by huge, hierarchically layered, and deeply virtualized infrastructures with complex interrelations between constituent components (Colman-Meixner et al [1]). Real-time management of those giant infrastructures strongly relies on monitoring and measuring every possible performance or capacity indicator of the system and represents a big data science application challenge.

Data science and machine learning approaches are applied to the collected data for behavioral pattern analysis and anomaly detection, problem root causing and other relevant intelligent tasks as predictive analytics in cloud systems. A package of such enterprise solutions and implementations are described in our earlier papers [2]-[5] dealing with both sources of data – structured and log, where, in particular, information-theoretic measures are applied in statistical inferences (see also notes by Vinck [6] on information theory and big data problems).

Similar data-driven solutions enable highly automated and effective control over increasingly complex and dynamic IT ecosystems that guarantee healthy business transactions running on top of those environments, preventing them from outages and losses that can impact millions of users and industry sectors.

Cloud management solutions and products (such as [7]) realize their efficient capabilities via monitoring a massive volume of infrastructure and application-related metrics and by storing them in well-organized specific data structures. This leads to a quite large storage consumption problem. In particular, the time series data storage engines might require terabytes of space for archiving months of data from hundreds of thousands of compute/storage/network resources with hundreds of metrics each. The problem becomes critical and unmanageable for those users who need to store years of data and one of options is quantization of time series subject to a fidelity criterion if the data owner does not need to preserve its original resolution.

We describe a multistage data compression method well-suited for environments self-organized into objects like clusters-/hosts/virtual machines (VM's) where the data within the same object have a higher chance to behave similarly. For that reason, together with standalone metric compression methods, we consider cross-metric approaches which rely on correlations of metrics within the same object.

In spirit of Wyner-Ziv coding [8], we conclude that the correlation helps in compression. Combination of both approaches gives better opportunity to optimally include metric properties into compression process and develop a suitable framework for compression of time series datasets. Experiments performed for real environments show promising results.

Conventional implementations of time series storage engines (see [7]) have serious scalability limitations in terms of storage consumption and disk I/O utilization. For example, it may consume $8KB$ for 1022 data points storing them in the format of time stamp plus float value. These result in quite a large storage space consumption, as in case of a moderate system of $12K$ IT resources/objects with 500 metrics per resource, a high availability (HA) system will consume as much as $4.9TB$ including HA duplication for storing historical data of 6 months. Therefore, it is a practical challenge to find a way to optimize the mentioned storage.

However, a direct implementation of well-known compression methods (see [9]) will not solve the problem. Although the storage space and I/O minimization have the highest priority, in general, a compromise between those optimizations and time of compression-decompression, CPU/memory utilizations is a real challenge. The proposed approach tries to find that optimal trade-off by utilizing natural correlation properties within databases resulting in an efficient compression mechanism. Our current prototype implementation assumes three levels of compression by combining standalone and cross-metric approaches (see Fig. 1).

At the first stage, we analyze time series in terms of their variability. Low-variability data cluster covers two categories of constant and semi-constant metrics. Experiments show that low-variability metrics make a big subset of all metrics and they allow compression by preserving all information content (lossless compression) by a minimum-size representation for rather long history of data.

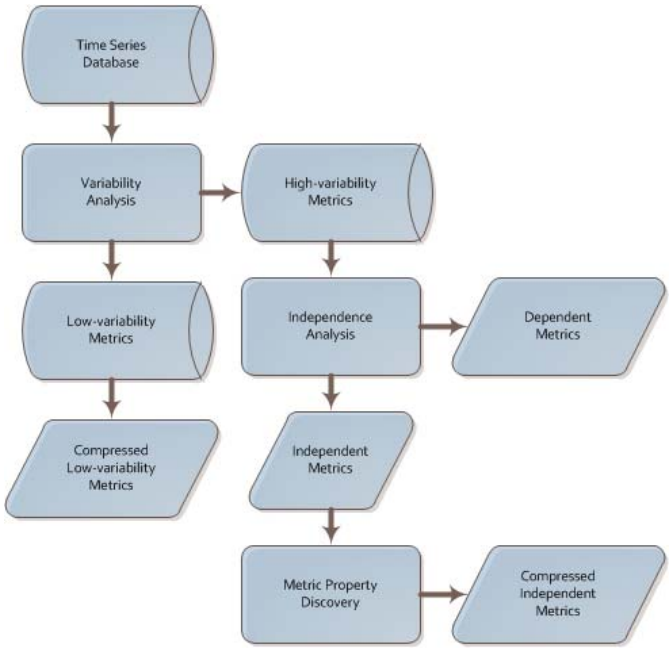

Fig. 1. Recommended data compression framework.

More specifically, let

$$x_k = x(t_k), k = 1, \dots, n$$

be data points of a time series and

$$v_k = x_{k+1} - x_k, k = 1, \dots, n-1$$

be the corresponding variability metric.

The condition $v_k = 0, k = 1, \dots, n-1$, characterizes the constant data. If $x_k = x_{k+1} = c$, then by storing only the value c and time stamps t_1, t_n, we can recover several months of data points with low resource consumption. If the data points for which $v_k \neq 0$ compose small portion of the original set (say less than 20%) then, we deal with semi-constant data. For this category, we can store only the values for which $v_k \neq 0$ with the corresponding time stamps and remove the remaining redundant parts. We assume that for the low-variability data the time stamps can be ignored obtaining better compression rates.

At the second stage, we separate the high-variability data into dependent and independent subgroups, where independent metrics compose a "basis" for an object and dependent metrics are represented as linear combinations of those "basis" elements. This allows effective compression of dependent metrics by storing only the coefficients in the linear combinations.

It is worth noting that the number of independent metrics depends on the accuracy of representation by linear combinations – as higher is the requested accuracy as bigger will be the number of independent elements. However, this allows a smooth transition between lossy and lossless compressions based on a user defined policy. Such a fidelity-based framework can be useful for real-life implementations where the distortion requirement on stored data can be linked to a time importance – as older the data as coarser can be the retained information and vice versa.

The third level deals with the independent metrics by further revealing their properties for optimal standalone metric compression. This level requires less effort as the percentage of independent metrics is rather small (see further examples) and different classical methods can be applied ([9]). Here, we do not discuss in details this stage of compression.

The paper is organized as follows. Section II describes the mathematical background of independence analysis with accuracy control, Section III outlines some implementation problems concerning our compression framework and demonstrates experimental results together with some discussions, and Section IV contains concluding notes.

II. Independence Analysis

Assume a set of s time series (metrics) with the same time stamps

$$X_{(j)}^{\circ} = \{x_{k,(j)}^{\circ}\}_{k=1}^{n}, j = 1, \dots, s, \quad (1)$$

where

$$x_{k,(j)}^{\circ} = x_{(j)}^{\circ}(t_k) \quad (2)$$

is the k-th data point of the j-th time series at the time stamp $t = t_k$.

Let $\sigma_{(j)}$ and $\mu_{(j)}$ be the standard deviation and the mean of $X_{(j)}^{\circ}$, respectively, and assume the following normalization of the preliminary data set

$$x_{k,(j)} = \frac{x_{k,(j)}^{\circ} - \mu_{(j)}}{\sigma_{(j)}}, \quad X_{(j)} = \{x_{k,(j)}\}_{k=1}^{n}, \quad j = 1, \dots, s \quad (3)$$

and let

$$X = (X_{(1)} | \dots | X_{(s)}) \quad (4)$$

be the matrix composed of the normalized time series with the j-th column corresponding to the j-th time series of the normalized set.

In the sequel, we discuss two different approaches for identifying the independent metrics of the column space of X: Independent Component Analysis (ICA) and Principal Component Analysis (PCA).

A. ICA

Within this approach, the basis of the column space of X is composed of the original metrics and the remaining metrics can be represented as linear combinations of the basic elements. Let I be the basis

$$I = \{I_{(1)}, \dots, I_{(m)}\} = \{X_{(i_1)}, \dots, X_{(i_m)}\}, m \leq s, \quad (5)$$

where m be the dimension of the column space of X and $I_{(j)}$ be the j-th basic metric. Then, the remaining metrics $X_{(j)}, j \neq i_k;\ k = 1, \dots, m; j = 1, \dots, s$, have the following representation

$$X_{(j)} = c_1^{(j)} I_{(1)} + \dots + c_m^{(j)} I_{(m)} \quad (6)$$

with some constants $c_k^{(j)}, k = 1, \dots, m; j = 1, \dots s; j \neq i_k$, and consequently,

$$X_{(j)}^{\circ} = \sigma_{(j)}(c_1^{(j)} I_{(1)} + \dots + c_m^{(j)} I_{(m)}) + \mu_{(j)}. \quad (7)$$

Representation (7) leads to lossless compression for dependent metrics (assuming that independents are preserved exactly) as by storing only the basic metrics $I_{(j)}, j = 1, \dots, m$, constants $c_k^{(j)}, \sigma_{(j)}, \mu_{(j)}, k = 1, \dots, m; j = 1, \dots s; j \neq i_k$, the rest of the dependent metrics $X_{(j)}^{\circ}, j = 1, \dots s; j \neq i_k; k = 1, \dots, m$ of the preliminary set can be recovered exactly at the

time stamps $t = t_k$. Moreover, (7) can be used also for lossy compression with a required accuracy. Namely, using only the first m_0 basic metrics, where $m_0 \le m$, we get

$$X^{\circ}_{(j)} \approx \tilde{X}^{\circ}_{(j)}, \quad j \neq i_k;\ k = 1, \dots, m_0; j = 1, \dots, s, \tag{8}$$

where

$$\tilde{X}^{\circ}_{(j)} = \sigma_{(j)}\left(c_1^{(j)} I_{(1)} + \cdots + c_{m_0}^{(j)} I_{(m_0)}\right) + \mu_{(j)}\,. \tag{9}$$

Approximation (8) can be controlled by some distortion measures. Throughout the paper, we discuss the following measures.

Relative Mean Square Error (RMSE):

$$\left\|\tilde{X}^{\circ}_{(j)} - X^{\circ}_{(j)}\right\|_{RMSE} = \frac{std\left(\tilde{X}^{\circ}_{(j)} - X^{\circ}_{(j)}\right)}{\sigma_{(j)}}, \tag{10}$$

that shows accuracy of reconstruction in average (std stands for the standard deviation of the corresponding time series data);

Absolute Error (AE):

$$\left\|\tilde{X}^{\circ}_{(j)} - X^{\circ}_{(j)}\right\|_{AE} = \left|\tilde{X}^{\circ}_{(j)} - X^{\circ}_{(j)}\right|, \tag{11}$$

that shows accuracy of reconstruction of each single data point.

How to find m_0, $I_{(j)}$ and $c_k^{(j)}$?

Step 1: Calculate the correlation matrix $\rho = \left(\rho_{i,j}\right)_{i,i=1}^{s}$ corresponding to X. The elements of ρ are the correlation coefficients (of Pearson product-moment correlation) between each pair of time series, calculated by the following formula

$$\rho_{i,j} = \frac{1}{n-1}\sum_{k=1}^{n} x_{k,(i)} x_{k,(j)}. \tag{12}$$

Step 2: Derive the QR-decomposition ([10]) of ρ

$$\rho = QR, \tag{13}$$

where Q and R are orthogonal and upper triangular matrices, respectively. Sort the diagonal elements of R in descending order of their absolute values and correspondingly reorder the columns of X

$$\left(X_{(i_1)} | \dots | X_{(i_s)}\right) \equiv \left(I_{(1)} | \dots | I_{(s)}\right). \tag{14}$$

Step 3: Take the first $m_0 = 1, \dots, s$ columns of (14) and compose the following matrix

$$I_0 = \left(I_{(1)} | \dots | I_{(m_0)}\right). \tag{15}$$

Coefficients $c_k^{(j)}$ in representation (9) can be found as solutions of the following least square problems:

$$I_0^T I_0 C^{(j)} = I_0^T X_{(j)}, \quad j = 1, \dots, s; j \neq i_k; k = 1, \dots, m_0, \tag{16}$$

where

$$C^{(j)} = \begin{pmatrix} c_1^{(j)} \\ \cdots \\ c_{m_0}^{(j)} \end{pmatrix}. \tag{17}$$

Step 4: Calculate $\tilde{X}^{\circ}_{(j)}$ by (9) and check one of the errors (10) or (11). If

$$\left\|\tilde{X}^{\circ}_{(j)} - X^{\circ}_{(j)}\right\| < \varepsilon \tag{18}$$

for some predefined ε then

$$\left\{I_{(1)}, \dots, I_{(m_0)}\right\} \tag{19}$$

is the required basis. Otherwise, increase the value of m_0 by 1 and return to Step 3.

B. PCA

In this approach the basis of the column space of $\boldsymbol{X}$ is composed of orthogonal time series $\boldsymbol{P_{(1)}}, \dots, \boldsymbol{P_{(m_0)}}$, known as principal components ([11]), which are constructed as linear combinations of the original metrics and, in general, don't coincide with them. All time series of the preliminary set can be approximated as linear combinations of the basic elements

$$X^{\circ}_{(j)} \approx \tilde{X}^{\circ}_{(j)}, j = 1, \dots s, \tag{20}$$

where

$$\tilde{X}^{\circ}_{(j)} = \sigma_{(j)}\left(c_1^{(j)} P_{(1)} + \cdots + c_{m_0}^{(j)} P_{(m_0)}\right) + \mu_{(j)}\,. \tag{21}$$

Distortion in (20) can be controlled by (10) or (11). By increasing the value of m_0 up to m (see (7)), we get an exact formula for lossless compression.

How to find m_0, $P_{(j)}$ and $c_k^{(j)}$?

Step 1: Calculate the correlation matrix $\rho = \left(\rho_{i,j}\right)_{i,j=1}^{s}$ corresponding to X (see (12)).

Step 2: Calculate eigenvalues $\lambda_k, k = 1, \dots s$ (sorted in descending order) and corresponding eigenvectors $v_k, k = 1, \dots s$. Let

$$V = (v_1 | \dots | v_s) \tag{22}$$

be the matrix composed of eigenvectors, where the j-th column corresponds to the j-th eigenvector. Calculate all principal components $P_{(1)}, \dots, P_{(s)}$ as follows

$$P = XV, \tag{23}$$

where

$$P = \left(P_{(1)} | \dots | P_{(s)}\right) \tag{24}$$

is the matrix composed of the principal components.

Step 3: Take the first $m_0 = 1, \dots, s$ principal components and calculate the corresponding constants $c_k^{(j)}$ by solving the following systems of linear equations

$$P_0^T P_0 C^{(j)} = P_0^T X_{(j)}, j = 1, \dots, s,$$

where $P_0 = (P_1 | \dots | P_{m_0})$ and $C^{(j)}$ is defined by (17).

Step 4: Recover the original metrics by (21) and check one of the errors (10), (11). If

$$\left\|\tilde{X}^{\circ}_{(j)} - X^{\circ}_{(j)}\right\| < \varepsilon \tag{25}$$

for some predefined ε then

$$\left\{P_{(1)}, \dots, P_{(m_0)}\right\} \tag{26}$$

is the required basis for the preliminary set of metrics. Otherwise, increase the value of m_0 by 1 and return to Step 3.

III. Experimental Results

We performed experiments on a virtualized private cloud environment and here, we demonstrate only the results concerning the ICA approach. In general, PCA produces similar outcomes, although slightly more accurate in terms of allowing recovery of the dependent metrics with less basis (principal) components with the same precision, but the difference is within 2-3%.

At the same time, the ICA has an advantage allowing to access directly/quickly the independent metrics and compress only the dependents, while in case of PCA, both categories

would be recovered via principal components with additional space requirement for their storage.

We considered a database containing 123,273 metrics from 284 monitored VMs. VMs have different count of metrics and Fig. 2 shows the distribution of their counts across different VMs. Maximum count is 620 and minimum is 294. Average count of metrics across VMs is 434. Metrics have different lengths with almost one-month durations.

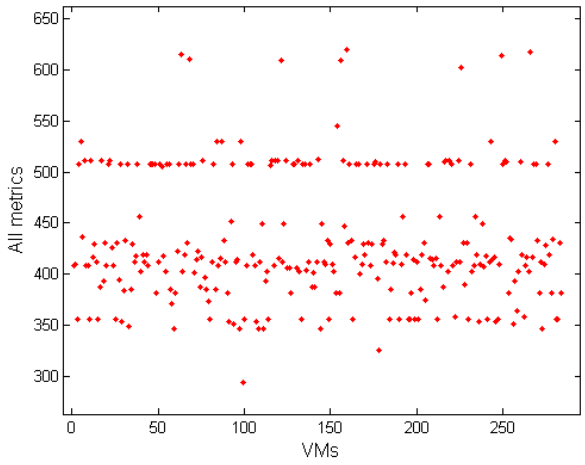

Fig 2. Count of all metrics across VMs.

The mathematical formulations of ICA and PCA share some common implementation problems.

The first problem is the complexity of calculation of the correlation coefficients for big datasets and further calculation of eigenvalues, eigenvectors, and QR-decompositions. In our experiments, we proceed naturally restricting ourselves with the metrics within the same resource. Virtualized big environments are self-organized into objects like IT resources with specific roles and data within the same object have a higher chance to behave similarly and be correlated.

As a rule, cloud management solutions monitor several hundred metrics for each object which is an acceptable size for modern matrix calculations. Moreover, in those solutions the metrics within an object are regrouped into some subgroups with similar types of indicators which maximizes the chance of metrics to be correlated. Hence, the set of metrics of an object or its metric groups are two excellent candidates for correlation analysis.

The second problem is the impact of outliers on the final representation of dependent metrics requiring more independent metrics for better accuracy. Therefore, well-known robust methods [12] for calculating correlation matrices can be useful for our framework. However, here we don't touch this issue and include outliers into our calculations, recovering them with the same accuracy as the other "normal" values.

The third problem is synchronization of time stamps of different metrics. To calculate the correlation matrix for a set of time series it is necessary to have the same values of time stamps within the set. Missing values or different monitoring intervals of some metrics make it impossible to proceed with calculations. One approach to overcome this difficulty can be interpolation for recovering the missing values. The second approach can be smoothing of initial time series by moving average or median methods with synchronization of the corresponding time stamps of smoothed metrics. This will help also to remove some extreme values which usually negatively impact on the final result. However, the latest approach will not allow controlling the accuracy of compression.

In our experiments, we encountered all those mentioned problems. Unfortunately, within VMs the metrics had different monitoring intervals and lengths. Therefore, we regrouped the metrics by their lengths and monitoring intervals and calculated the corresponding correlation matrices only within those groups. Overall, we formed 4917 groups, where the biggest group included 465 metrics and the smallest - 1 metric.

Then, we perform a variability analysis for selecting the constant and semi-constant metrics that compose the low-variability data. The average percentage of constant metrics is 45%. The maximal percentage is 69% and minimal is 30%. The average percentage of semi-constant metrics is 16%. The maximum is 35% and minimum is 1% (see Fig. 3).

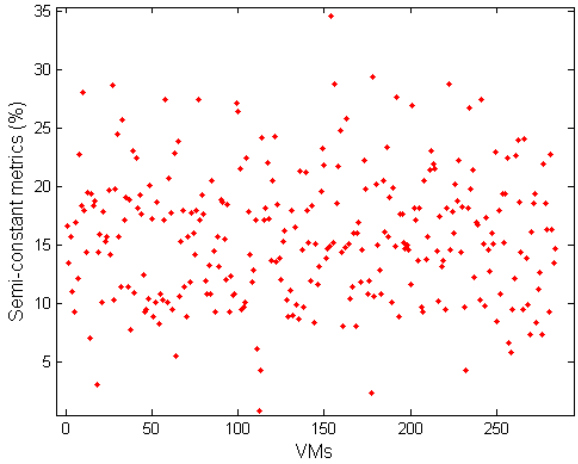

Fig 3. Percentage of semi-constant metrics across VMs.

Fig. 4 shows the distribution of low-variability metrics across different VMs. The maximal percentage of low-variability metrics is 85%, the minimal is 45% and the average across VMs is 60%.

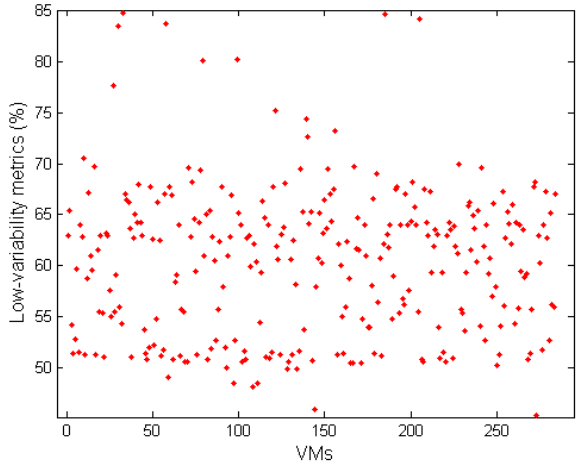

Fig 4. Percentage of low-variability metrics across VMs.

We apply the independence analysis only for those groups where after selection of low-variability metrics the number of high-variability metrics is greater than 2. As an accuracy measure we take RMSE norm (see (10)) with different ε thresholds (see (18)). Fig. 5 presents the count of independent metrics for different VMs when $RMSE \leq 0.01$. Average percentage of independents is 22%.

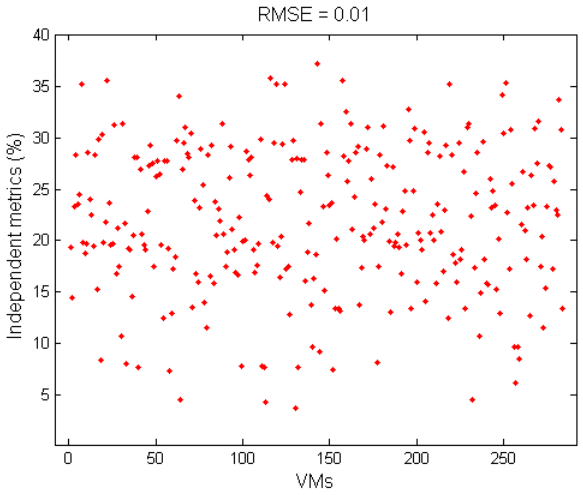

Fig 5. Percentage of independent metrics across VMs for recovering dependent metrics with $RMSE \leq 0.01$.

It is worth recalling that the independence analysis with the required RMSE means that all metrics within a group will be recovered at least per that accuracy. However, as showed our experiments, the dependent metrics were recovered with better accuracies.

Efficiency of our approach can be measured by reduction rate (RR). Under the reduction rate, we mean the percentage of low-variability and dependent metrics against the all metrics

$$RR = \left(1 - \frac{independent\ metrics}{all\ metrics}\right) * 100. \qquad (27)$$

Finally, when $RMSE \leq 0.01$, we observe that from 123,273 metrics 44.3% or 54555 metrics are constant, 15.6% or 19202 are semi-constant, 22.47% or 27701 are independent with the required accuracy and the remaining 21815 metrics or 17.7% can be recovered as linear combinations of independents (see Fig. 6). As a result, we have $RR = 78\%$.

Let us consider the "worst" scenario of a database including only high-variability metrics. By excluding from our experiments the low-variability cluster, we get overall 49520 high-variability metrics from which 27701 metrics are independent (56%) when $RMSE \leq 0.01$ which means that $RR = 44\%$. This shows the percentage of metrics that can be effectively recovered by storing only few coefficients. Ignoring the negligible overhead to store linear combination coefficients, this percentage is the rate of compression for the "worst" class of monitoring data. As mentioned above, we performed experiments for different RMSE thresholds and Fig. 7 shows the corresponding reduction rates subject to chosen RMSE degrees.

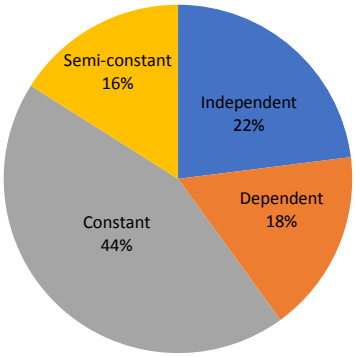

Fig 6. Percentage of different metric categories for $RMSE \leq 0.01$.

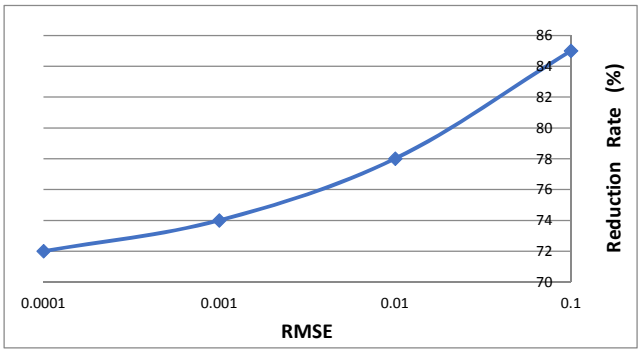

Fig. 7. Reduction rate versus RMSE constraint.

Consider a specific group with 8 CPU metrics (Table 1). One of the metrics is constant and one is semi-constant (see Fig. 8). From the remaining 6 metrics we found 3 independents and 3 dependents (with $RMSE \leq 0.01$). We observed that the actual errors were different and better than the required accuracy. Dependent metrics were recovered from independent components in the following way:

$$\begin{aligned} X_1 &= 9 \cdot 10^{-8} I_1 + I_2 + 1 \cdot 10^{-7} I_3, \\ X_2 &= 0.99997 I_1 + 6 \cdot 10^{-5} I_2 - 4 \cdot 10^{-4} I_3, \\ X_3 &= -7 \cdot 10^{-2} I_1 + 2 \cdot 10^{-3} I_2 + 0.993 I_3, \end{aligned} \qquad (28)$$

and Fig. 9 shows X_1 with its reconstruction error (11) while recovering it by (28).

Table 1. A CPU metric group with RMSEs of dependent components.

Metric name	Category	RMSE
cpu-0 swapwait summation	constant	-
cpu-0 costop summation	semi-constant	-
I_1 = cpu-0 used summation	**independent**	-
I_2 = cpu-0 readyPct	**independent**	-
I_3 = cpu-0 idle summation	**independent**	-
X_1 = cpu-0 ready summation	dependent	$\mathbf{2.3 \times 10^{-7}}$
X_2 = cpu-0 usagemhz average	dependent	$\mathbf{6.5 \times 10^{-4}}$
X_3 = cpu-0 wait summation	dependent	$\mathbf{4.8 \times 10^{-3}}$

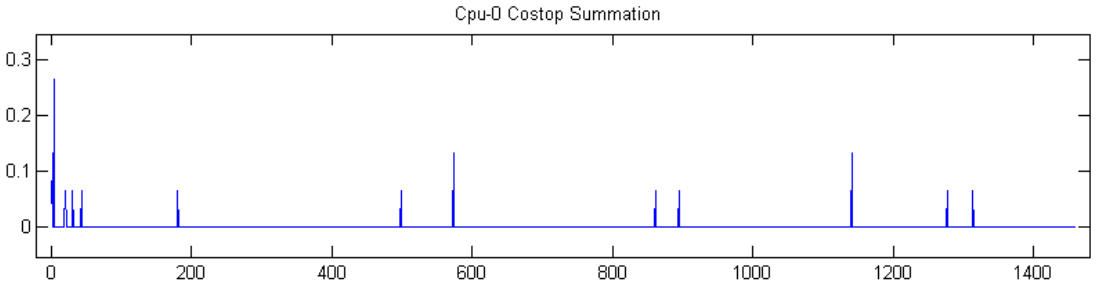

Fig. 8. Example of a semi-constant time series data from Table 1.

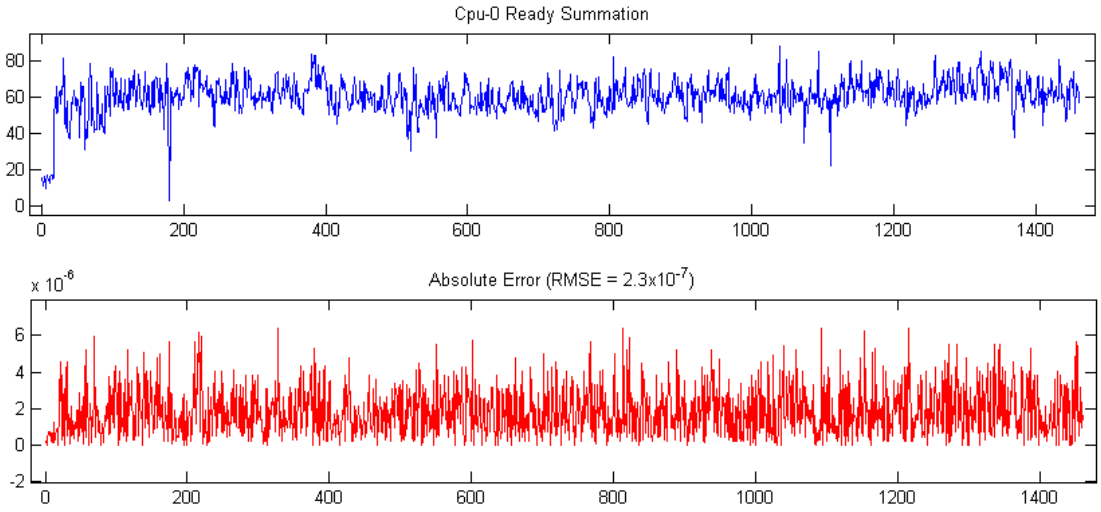

Fig. 9. Graph of X_1 (top) and the corresponding absolute error of reconstruction (bottom).

IV. Conclusions

We proposed a data compression framework for time series databases using variability and independent component analysis (via ICA or PCA) applied to a private cloud environment. Experimentally (see Fig. 7), $15\% - 28\%$ of independent metrics are enough to recover the dependent ones subject to the required accuracy $0.1 \leq RMSE \leq 0.0001$. It means that overall reduction rate is $72\% - 85\%$ for a real data set from a private cloud environment.

In case of the "worst" scenario, when we exclude the low-variability cluster from our time series database, the reduction rate is only $RR = 44\%$ instead of $RR = 78\%$ when the required accuracy is $RMSE \leq 0.01$. When we exclude only the constant cluster, but consider the semi-constant metrics, the reduction rate is $RR = 60\%$ with $RMSE \leq 0.01$.

An intuition behind those observations is that monitoring more object indicators does not necessarily increase the relative dimensionality of the data.

References

[1] C. Colman-Meixner, Ch. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," *IEEE Communications Surveys and Tutorials*, vol. 18, issue 3, pp. 2244-2281, 2016.

[2] A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, "Managing cloud infrastructures by a multi-layer data analytics," Proc. *IEEE International Conference on Autonomic Computing (ICAC)*, July 18-22, Wuerzburg, Germany, pp. 351-356, 2016.

[3] A.N. Harutyunyan, A.V. Poghosyan, N.M. Grigoryan, and M.A. Marvasti, "Abnormality analysis of streamed log data," Proc. *IFIP/IEEE Network Operations and Management Symposium (NOMS)*, May 5-9, Krakow, Poland, pp. 1-7, 2014.

[4] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, "An enterprise dynamic thresholding system", Proc. *USENIX 11th International Conference on Autonomic Computing (ICAC)*, June 18-20, Philadelphia, PA, pp. 129-135, 2014.

[5] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, "Pattern detection in unstructured data: An experience for a virtualized IT environment," Proc. *IFIP/IEEE International Symposium on Integrated Network Management*, Ghent, Belgium, May 27-31, pp. 1048-1053, 2013.

[6] A.J. Han Vinck, "Information theory and big data: Typical or not typical, that is the question," Proc. Workshop on Information Theory and Data Science: *From Information Age to Big Data Era*, Yerevan, Armenia, October 3-5, 2016.

[7] VMware vRealize Operations Manager, http://www.vmware.com/products/vrealize-operations.html.

[8] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, issue 22, pp. 1-10,1976.

[9] D. Salomon and G. Motta, *Handbook of Data Compression*, Springer, 2010.

[10] G.H. Golub, C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 1996.

[11] I.T. Jolliffe, *Principal Component Analysis*, Springer, 2002.

[12] J. Chilson, R. Ng, A. Wagner, and R. Zamar, "Parallel computation of high dimensional robust correlation and covariance matrices, *Algorithmica*, vol. 45, issue 3, pp. 403-431, Springer, July 2006.