Real-Time Data Center's Telemetry Reduction and Reconstruction Using Markov Chain Models

Shuja-ur-Rehman Baig, Waheed Iqbal, Josep Lluis Berral, Abdelkarim Erradi, and David Carrera

***Abstract*—Large-scale data centers are composed of thousands of servers organized in interconnected racks to offer services to users. These data centers continuously generate large amounts of telemetry data streams (e.g., hardware utilization metrics) used for multiple purposes, including resource management, workload characterization, resource utilization prediction, capacity planning, and real-time analytics. These telemetry streams require costly bandwidth utilization and storage space, particularly at medium-long term for large data centers. This paper addresses this problem by proposing and evaluating a system to efficiently reduce bandwidth and storage for telemetry data through real-time modeling using Markov chain based methods. Our proposed solution was evaluated using real telemetry datasets and compared with polynomial regression methods for reducing and reconstructing data. Experimental results show that data can be lossy compressed up to 75% for bandwidth utilization and 95.33% for storage space, with reconstruction accuracy close to 92%.**

***Index Terms*—Data center monitoring, data reconstruction, data reduction, Markov chain models (MMs), polynomial regression (PR), real time, telemetry.**

I. Introduction

NOWADAYS petabytes of digital data are produced mainly by the readily available network-enabled electronic devices, social networks, electronic health-care gadgets, and data centers [1]. This increasing growth of digital data poses special challenges to process, store, and analyze the collected data [2]. Therefore, we required new algorithms, tools, and systems to manage the dramatically growing data.

Cloud computing enables users to host their data and applications in remote data centers mainly due to pay-as-you-go and dynamic-scalability features [3]. These large-scale data centers consist of thousands of servers, organized in racks and interconnected to offer services to a large set of users. Such data centers generate large and continuous streams of telemetry data that are logged and analyzed for multiple purposes, including resource management, workload characterization, resource utilization prediction, capacity planning, and real-time analytics [4]–[7].

Typical telemetry streams contain time-series data about hardware utilization metrics, including central processing unit (CPU), memory, I/O, bandwidth, context switches, interrupts, cache misses, and cycles per instruction (CPI). These telemetry metrics are used in various applications. For example, CPU, memory, network bandwidth, instructions per cycle, cache miss rates, branch predictor statistics, and power consumption utilization data are used to forecast energy consumption and reduction [8], [9]. CPU, memory, disk, and network consumption are used for data center management and resource prediction [10]–[13]. This generation of continuous telemetry streams from all computing and storage nodes poses a significant challenge within the data center in terms of bandwidth consumption and storage requirements. As an example, considering telemetry collection on a data center consisting of 10 000 computing nodes and collecting 12 different measured metrics every second, while dedicating 4 bytes per metric, would require nearly 40 GB storage per day, this is more than 1 TB of storage per month, plus the meta-data overheads for time-series traceability.

Traditional data compression solutions to reduce the data increase the time to collect telemetry data from the computing nodes. Increased interval time does not allow capturing fine-grained resource consumption and may not precisely reflect the resources usage. Also, using compression techniques on floating point values cannot reduce the size significantly and preserve the full precision [14]–[16]. Unfortunately, telemetry streams in many data centers show low or no smoothness and high variation in data. In such cases, even state-of-the-art floating point compression algorithms are still not sufficient to compress the data [17].

On large-scale data centers with millions of hosted applications, the usefulness of telemetry and profiling comes from knowing the behavior patterns more than the exact metrics. The precision is reduced when data are aggregated in bigger time periods. Hence, telemetry time-series data reduction methods need to preserve statistical properties capturing hardware utilization behaviors, specifically, burstiness, rapid growth of utilization, and abnormal hardware utilization patterns.

Manuscript received May 27, 2018; revised November 16, 2018 and February 15, 2019; accepted May 16, 2019. This work was supported in part by the European Research Council (ERC) under the EU Horizon 2020 programme under Grant GA 639595, in part by the Spanish Ministry of Economy, Industry and Competitiveness under Grant TIN2015-65316-P and Grant IJCI2016-27485, in part by the Generalitat de Catalunya under Grant 2014-SGR-1051, in part by the University of the Punjab, Pakistan, and in part by the Qatar National Research Fund (a member of Qatar Foundation) under NPRP Grant # NPRP9-224-1-049. *(Corresponding author: Shuja-ur-Rehman Baig.)*

S.R. Baig, J. L. Berral, and D. Carrera are with Barcelona Supercomputing Center (BSC), 08034 Barcelona, Spain, and also with Universitat Politècnica de Catalunya (UPC), 08034 Barcelona, Spain (e-mail: shujamughal@gmail.com; josep.berral@bsc.es; david.carrera@bsc.es).

W. Iqbal is with the Punjab University College of Information Technology, University of the Punjab, 54590 Lahore, Pakistan (e-mail: waheed751@gmail.com).

A. Erradi is with the Department of Computer Science and Engineering, Qatar University, 2713 Doha, Qatar (e-mail: erradi@qu.edu.qa).

Digital Object Identifier 10.1109/JSYST.2019.2918430

In this paper, we address the collection and compression of large-scale data centers telemetry data. Our proposed solution consists of reducing the telemetry measurements of the data center in real time through online modeling using Markov chains [18]. Then, such models are transmitted to the corresponding logging repositories and stored to enable data reconstruction for posterior use with minimum data loss to preserve the hardware utilization behaviors. The method works on rack level to collect all measurements from the nodes deployed on the rack and then applies Markov chain model (MM) to efficiently reduce the data in real time. The reduced data is logged to allow telemetry data analytics. We also propose an efficient method to combine reduced data with compression to minimize the overall storage requirement for storing telemetry data for a long duration. Thus it minimizes both the storage space and the bandwidth utilization for collecting data center telemetry measurements.

The proposed method has been evaluated using real telemetry datasets and compared with state of the art methods, such as the polynomial regression (PR) method [19] and the dictionary-based compression (ZIP). Several comparison metrics were used, such as computing the amount of data saved in each scenario and the data reconstruction accuracy in case of lossy compression. We evaluate the effectiveness of the methods on telemetry time-series data by calculating and comparing the storage requirements for each method. The reconstruction accuracy is evaluated by comparing data before compression and after reconstruction. First, we compare the statistical similarity between the reconstructed data and the original data using a two-sample Kolmogorov–Smirnov (K–S) hypothesis tests [20]. K–S test is used to identify whether two given one-dimensional sequences belong to the same probability distribution or not. It does not quantify the similarity of the reconstructed data, but indicate whether the reconstruction has statistical resemblance. Then we complement the evaluation by quantifying this similarity using the dynamic time wrapping (DTW) metric [21], [22]. The DTW is a well-known method used to measure the similarity between two given sequences which may vary in speed [22]–[26].

The main contributions of the paper are summarized as follows.

1) Design a system for real-time telemetry data reduction and reconstruction for data centers.
2) Develop and evaluate telemetry data reduction and reconstruction approach using Markov chain models.
3) Compare the proposed data reduction and reconstruction with state-of-the-art polynomial regression (PR) based methods and ZIP compression.
4) Experimental evaluation to study the storage and bandwidth minimization using the proposed solution for telemetry data for different data center sizes.

The rest of the paper is organized as follows. Related work is presented in Section II. Our proposed data reduction and reconstruction system is explained in Section III. Experimental setup details are given in Section IV. The evaluation results are presented in Section V. The benefits in bandwidth and storage reduction for different sizes of data centers using the proposed solution are discussed in Section VI. Finally, conclusion and future work are discussed in Section VII.

II. Related Work

There have been several efforts to reduce exponentially growing digital data for efficient management [27]–[34]. Different methods were proposed, including dimensionality reduction, forecasting models, and compression methods. For example, Bhuiyan *et al.* [35] proposed an IoT framework for event detection and data reduction at data collection time which helps to minimize data transmission across the network and also reduces energy consumption. The proposed framework detects fire events using sensors and rule-based methods. Wu *et al.* [36] developed a dictionary-based compression technique to split the incoming numeric data stream into fixed size blocks and compares them with the already stored blocks using K–S statistical test to measure the similarity. When they identified any existing block similar to the incoming new block, they discarded the incoming block and keep a reference of the old block to be able to regenerate the data, thus helping to significantly reduce the required storage. Another work by Egri *et al.* [37] use dimensionality reduction of multidimensional time-series data. Their approach introduces graph-based clustering using the cross-correlation between the time-series data. The authors focus is on identifying connections among various performance metrics in order to reduce the number of performance metrics to track.

A recent work [38] uses a correlation-based method to reduce data center's monitoring data. The authors identify the correlation between different measurement metrics using Bayesian network models learned from historical data and proposed to use linear regression between correlated metrics. Bayesian network models are directed acyclic graphs showing the relation between metrics in the form of dependant and independent metrics. In this method, the authors reduce the sampling rate of dependent metrics and predict them using linear regression for a given duration, which helps to reduce the data at the data collection stage. Another recent work by Yu *et al.* [39] proposed a method to reduce data sent by edge nodes of IoT devices to the cloud for reducing data transmission time. The method modeled the incoming data as multivariant normal distribution and used Kalman filter to predict mean vector and covariance matrix of the distribution. Both the edge nodes and cloud predict the same values using identical Kalman filters. If the predicted data at both ends do not meet the confidence interval, then data are uploaded from fog to cloud layer, otherwise predicted values are used, which helps to reduce the data movement from fog to cloud layer.

A most recent work [40] in data reduction addresses the problem of transmitting data in smart energy metering infrastructure. The authors proposed a framework to monitor data of energy consumption using smart meters and then aggregate the data for a fixed batch intervals. Then use an already learned forecasting method to compare the new data. If the new data is comparable with the forecasting model then the new data is not sent to the cloud. However, if the forecasting model and the new data are different, then they send the data to the cloud and also update the forecasting method. The proposed system is adaptive according to incoming data as it does not rely on single forecasting method, rather it changes its method to suite the

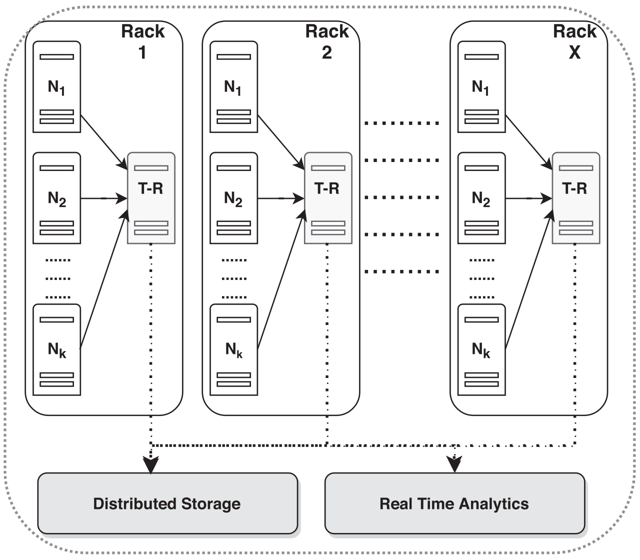

Fig. 1. Schema of the proposed data reduction framework for data centers. Each rack in the data center hosts a T-R component, which continuously collects the telemetry data from each computing node, then performs data reduction before transmitting it to storage and real-time analytic systems.

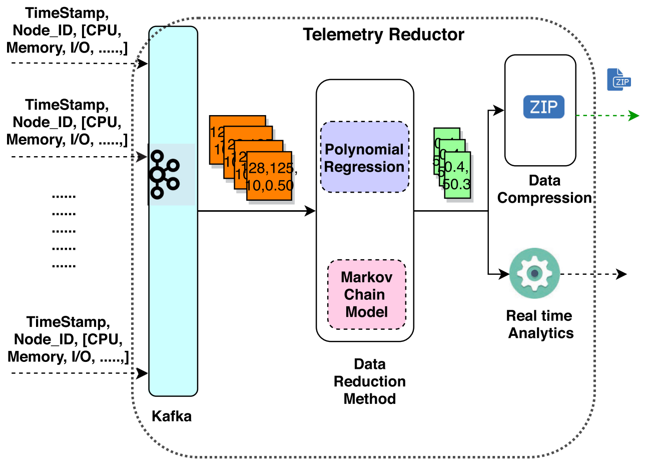

Fig. 2. T-R process the incoming telemetry streams from computing nodes using Kafka and data-reduction method. The output of the T-R can be used to store or provide input to other real-time analytic systems.

current data. However, this work does not address the data storage requirement optimization and only focuses on one-dimensional energy consumption data.

Most of the existing works are based on either dimensionality reduction, forecasting models, or compression methods to reduce data. Our work proposes a novel Markov chain based telemetry data reduction and reconstruction system to efficiently reduce network bandwidth utilization and storage space. The data reduction is performed in real time without reducing the dimensions of input telemetry streams and without using forecasting models that need to update or change whenever input streams are changed. The proposed Markov chain models reduce the input data significantly without updating the model. Moreover, our purposed system perform more reduction compared to dictionary-based ZIP compression methods. To the best of our knowledge, the current state-of-the-art methods do not perform reduction in real time for data centers telemetry streams using Markov chain models. We investigate the use of Markov chain models and compare it with PR and ZIP compression with different settings to reduce the data significantly and reconstruct the data with high accuracy for data center telemetry streams.

III. Data Reduction Framework

Our proposed system to reduce telemetry data provides twofold benefits. First, it reduces the storage space significantly, and second, it minimizes the bandwidth utilization required by telemetry data collection within the data center.

Fig. 1 shows the architecture of the proposed real-time telemetry data reduction system for data centers. Each rack in the data center hosts a telemetry reductor (T-R) component, which continuously collects the telemetry data from each computing node and performs data reduction method before transmitting it to storage and real-time analytic systems. This reduces the data at the rack level and does not transmit the entire data, but only the reduced data within the data center. A telemetry stream data can consist of utilization of CPU, memory, disk, network, memory bandwidth, and other useful metrics.

Fig. 2 shows the T-R process. We used Apache Kafka [41], [42] as a message broker to receive telemetry stream data from the computing nodes. Every computing node publishes telemetry data with the timestamp to Kafka topic, which is consumed by the T-R. Each consumer obtains data from Kafka topics. The consumer reads the incoming streams and splits the data into a predefined batch size (BS) and uses a data reduction method to minimize the data. Then the reduced data is compressed and stored in a data center storage facility. The reduced data is also fed to analytics engines. The telemetry data can be used for real-time monitoring, workload characterization, and anomaly detection purposes. The reduced data sent to other components need to be reconstructed before usage. The above-mentioned architecture is based on out-of-band monitoring where data are coming from devices, such as sensors or logging applications. In case of the regular host, where in-band monitoring is possible, then T-R component could be executed on the host machine without Kafka to reduce the extra overhead of a dedicated machine.

The proposed framework targets two main goals: 1) it minimizes the storage demand for storing telemetry measurement by reducing and compressing the telemetry data; 2) it minimizes the bandwidth utilization by transmitting solely reduced data over the data center network. To achieve such goals, we propose two different methods, namely PR and Markov chain models to be used for telemetry data reduction and reconstruction in real time. We explain both of the methods in the following subsections.

A. *Reduction Through PR Methods*

First, we propose to use PR methods to fit the curve of a given telemetry data stream into a polynomial curve. Then we only need to store the coefficients of the equation fitting the curve.

This method is inspired by similar work [19] that uses linear regression method for data reduction. The PR method is used as a baseline to compare our proposed Markov chain model based approach.

To understand the effect of using PRs for data reduction, if we plan to fit a data-stream function into a 4 polynomial curve then we will only store four coefficients plus the intercept. Assuming that the data stream contains 128 data points, we will be reducing the data from 128 values to only 6 values.

If we train a PR model for each telemetry measurement, given an n-degree PR, and k observations of a specific telemetry dimension a, the coefficients are computed using pseudo-inverse solution to minimize the sum of least squares

$$\begin{pmatrix} \alpha_o \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \left[R^\top R\right]^{-1} R^\top \begin{pmatrix} r_t \\ r_{t-1} \\ \vdots \\ r_{t-k} \end{pmatrix} \tag{1}$$

where the matrix R is defined as

$$R = \begin{pmatrix} 1 & a_t & a_t^2 \dots & a_t^n \\ 1 & a_{t-1} & a_{t-1}^2 \dots & a_{t-1}^n \\ \vdots & \vdots & & \\ 1 & a_{t-k} & a_{t-k}^2 \dots & a_{t-k}^n \end{pmatrix}. \tag{2}$$

Using the values of $\alpha_0, \alpha_1, \dots, \alpha_n$ coefficients and polynomial degree n, we can load the polynomial curve and reconstruct the k data points easily. Therefore, we propose to only store coefficients and polynomial degree information instead of the actual data. Here we monitor and collect the data for a given batch interval and then fit a PR model with a specific degree. After fitting the model, we extract the coefficients from the fitted equation and store these instead of the actual data. Whenever we have to reconstruct a batch of data, we load the coefficient values for that batch and build its corresponding PR model, then we use it to reconstruct the data points for the batch.

B. Reduction Through Markov Chain Models (MM)

Second, we propose to use Markov chain models (MM) [18] for telemetry data reduction and reconstruction. Markov chains are stochastic models describing a sequence of events in which the probability of each event depends only on the previous state of the event. Specifically, we use time-homogeneous discrete-time Markov chains (DTMC) [43] because telemetry data are monitored at discrete time intervals and the state transition probabilities are independent of time. Moreover, the DTMC used with the telemetry data is irreducible ergodic as the proposed system can transit from every state to every other state with positive probability. Fig. 3 shows the irreducible ergodic 2-state Markov chain.

The idea of using MM is to explore that if we can deal with burstiness behaviors among others for data reduction and reconstruction. The burstiness behavior represents sudden spikes and peaks in the telemetry data. In general, it is challenging to reconstruct the burstiness behavior, and we address it by using Markov chain models.

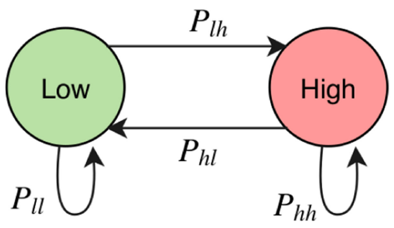

Fig. 3. Two-states Markov chain model showing states and transition probabilities. For transition probabilities, l represents low state and h represents high state.

Let $X_1, X_2, X_3, \dots$ are independent and identically distributed random variables representing telemetry data. We model these random variables as a DTMC model defining the probability of moving from the current state to the next state as

$$\begin{aligned} &Pr(X_{t+1} = s_j | X_t = s_i, \dots, X_2 = s_2, X_1 = s_1) \\ &\quad = Pr(X_{t+1} = s_j | X_t = s_i) \end{aligned} \tag{3}$$

where function $Pr(X_{t+1} = s_j | X_t = s_i)$ is independent of t and denotes the probability of moving from state s_i at time t to state s_j at time $t+1$, symbol " $|$ " represents the conditional probability, and $s \in$ state space (S).

Let $P_{ij}^v, v > 1$ where $P_{ij}^v = P(X_{u+v} = j | X_u = i)$ denotes the probability that after v time units, the chain will transit to state j given that the current state is i at time u. The probability of reaching j from i in n-steps is the sum of all probabilities going from i to j through an intermediate point k. We use Chapman–Kolmogorov equation [44] to compute it as follows:

$$P_{ij}^{u+v} = \sum_{k \in S} P_{ik}^u P_{kj}^v; \quad m \text{ and } v \geq 1, \quad i \text{ and } j \in S. \tag{4}$$

Let $\mathbf{P}^v = (P_{ij}^v)$ be a matrix then Chapman–Kolomogorov equation can be expressed as $\mathbf{P}^{u+v} = \mathbf{P}^u \mathbf{P}^v$. This allows calculating the transition probability matrix P, which reflects the relative frequencies of transitions from one state to another state. The matrix P for n total number of states is represented as follows:

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix}. \tag{5}$$

In our solution, the chain can transit from every state to every other state and is considered as irreducible ergodic Markov chain. For an irreducible ergodic Markov chain, the transition matrix elements must be non-negative, $p_{ij} \geq 0$ and the sum of each row must be equal to 1, therefore, $\sum_{j=1}^n p_{ij} = 1$.

To reduce a given telemetry data, first we convert it into state interval matrix $I = [\dots]_{(n+1)\times 1}$ and then compute state transition frequency matrix $F = [\dots]_{n\times n}$, where n is the total number of states. The state interval matrix contains the threshold values to partition the given data into n states. The state transition frequency matrix contains the transition frequencies, which are

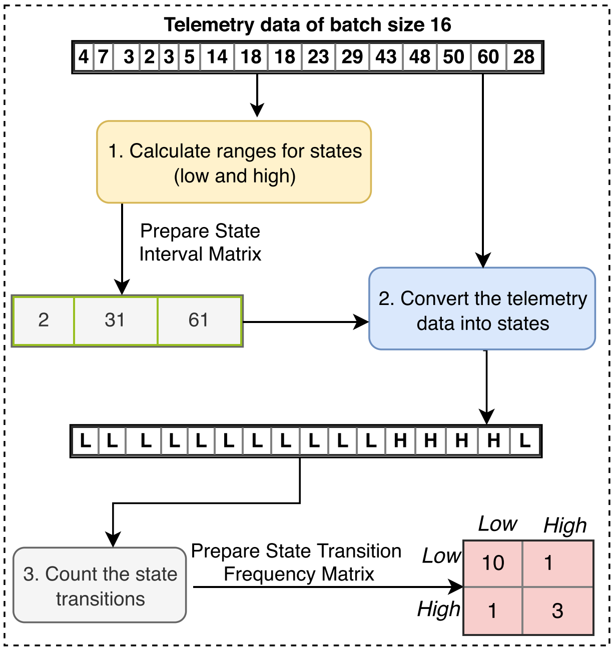
Telemetry data of batch size 16
4 7 3 2 3 5 14 18 18 23 29 43 48 50 60 28
1. Calculate ranges for states (low and high)
Prepare State Interval Matrix
2 31 61
2. Convert the telemetry data into states
L L L L L L L L L L L H H H H L
3. Count the state transitions
Prepare State Transition Frequency Matrix
Low High
Low 10 1
High 1 3

Fig. 4. Telemetry data reduction using 2-state Markov chain model and batch size 16.

computed by simply counting the number of transitions from one state to another.

Fig. 4 shows the process to compute the frequency matrix for reducing telemetry data using a 2-states MM with batch size 16. First, we calculate the state interval matrix I, which stores the partition boundaries. For 2-states MM, the state interval matrix will be of size 3×1 to store the boundaries of the states. Second, we convert each data point into corresponding states using the state interval matrix. Finally, we compute the state transition frequencies matrix F. We only store F and I for each given batch of data to reduce the telemetry data. During data reduction, we do not compute and save transition probability matrix P because it is required during the reconstruction and can be easily calculated using F. Moreover, P consists of floating point data which also introduces storage overhead.

To reconstruct the data for a given batch interval, we first convert the state transition frequency matrix F into state transition probability matrix P using (6)

$$p_{ij} = \begin{cases} \frac{f_{ij}}{\sum_{r=1}^{x} f_{ir}}, & \text{if } \sum_{r=1}^{x} f_{ir} \neq 0 \\ 0, & \text{otherwise.} \end{cases} \qquad (6)$$

During the reconstruction of a batch data, we assume that the current/initial state is known and we need to predict all data points of the given batch size. To explain the reconstruction strategy, let us consider a 2-states MM as shown in Fig. 3. The transition probabilities P can be easily computed from the corresponding transition frequency matrix F, which was stored during the data reduction phase. Let us assume that the current state is *high*, the transition probability from *high* to *low* (p_{hl}) is 0.3, and the transition probability of *high* to *high* (p_{hh}) is 0.7. We generate a random number between 0 and 1. If the generated number is greater than 0.3 then we predict *high* otherwise *low* as the next state. Once we identify the next state, then we look up the lower and boundary values of the predicted state from the state interval matrix I. Finally, we generate another random number within the range of the state boundary and consider it as the reconstructed data point. We repeat this process to identify all data points for the given batch interval.

IV. Experimental Setup

A. Description of the Dataset

We used IBM POWER8 telemetry logs dataset [45] to evaluate our proposed method for telemetry reduction and reconstruction. These logs contain telemetry data generated by executing three representative Spark workloads from the Spark-Bench [46] developed by the IBM and widely tested using POWER8 systems. The dataset logs are collected from executions of the workloads “support vector machines,” “PageRank,” and “Spark SQL.” These workloads are well known in the literature and combine different characteristics to cover a large range of different resource usage behaviors. The dataset contains metrics related to CPU, memory, context switches, memory bandwidth, L2 and L3 cache misses, interrupts, and CPI as a time-series data.

B. Experiment Details

As explained previously, we propose and evaluate two different techniques, namely PR and Markov chain models, to reduce and regenerate telemetry data through modeling. To evaluate the proposed methods, we performed two major experiments, briefly explained in the following subsections.

1) Experiment 1: Data Reduction and Reconstruction Using PR: In this experiment, we study the effect of different polynomial degrees and batch sizes on PR models. The polynomial degree defines the shape of the curve, where a higher degree can be used to fit a complex curve and a lower degree can be used a simple curve. The batch size defines the number of data points used to fit the curve. We consider polynomial degrees 2, 4, 6, 8, and 10 with batch sizes varying from 2, 4, 8, 16, 32, 64, and 128 to fit polynomial curves for data reduction and reconstruction. Many other settings can also be studied, however, the selected settings are sufficient to establish the motivation for using PR for telemetry data reduction and reconstruction because with higher degrees, we get less reduction and with higher batch sizes, we loose the accuracy of the reconstructed data. Once we train a PR model using a specific polynomial degree and a batch size, then we only store the coefficients of polynomial equation learned to fit on the given data points. This helps to reduce the data size significantly as we do not store all data point, but only few coefficient values. Later, these coefficients can be used to regenerate the data points easily. However, an efficient polynomial degree and batch size values should be used to achieve a good data reconstruction accuracy. The higher the degree, the more over fitting will be achieved, but more data (coefficients) will be transmitted.

2) Experiment 2: Data Reduction and Reconstruction Using Markov Models (MM): In this experiment, we study the effect of different MM models with varying the number of states and batch sizes. We consider 2, 3, and 4 state MM on batch sizes varying from 2, 4, 8, 16, 32, 64, and 128 to study the effects on data reduction and reconstruction. Higher batch sizes reduces accuracy as the reconstruction does not retain the information, which is present in the original dataset. On the other hand, higher MM model states yields less reduction. Fig. 3 shows a 2-state MM, in which we divided the input data into two regions, namely, low and high and learn a state transition matrix for telemetry observations using a specific batch size. The state transition matrix contains the probabilities of moving from one state to other using the given input data points of the telemetry metrics. We also create a state interval matrix, which defines the low and high region ranges. These two matrices are learned and stored instead of the complete data, which reduces the data size significantly. Similarly, for 3 and 4 state MM, we increase the number of states to 3 and 4, respectively, and learn state transition metrics and state interval matrix accordingly.

C. Evaluation Criteria

We evaluate both of the proposed methods, in terms of data reduction effectiveness and reconstruction accuracy. For evaluating the effectiveness of data reduction, we calculate the data reduction percentage after applying the proposed methods. For reconstruction accuracy, first, we perform two-sample K–S test to decide whether to accept or reject the produced reconstruction of data after compressing, storing, and reconstructing it. This first evaluation is used to initially discard configurations (PR degrees and MM discrete states) that produce low quality reconstructions. Second, we compute DTW distance between reconstructed data and actual data to quantify the reconstruction error. We explain all these evaluation measures in the following subsections.

1) Data Reduction Percentage: We compute the data reduction percentage by measuring the data stored after applying the reduction method against the original data. A positive value shows a reduction in data size, while a negative value indicates a growth in data size. Therefore, a higher data reduction percentage is better and desirable.

D. Two-Sample K–S Test

We use two-sample K–S test [20] to compare the statistical similarity of the actual data with the reconstructed data. The K–S test is used to determine whether two given one-dimensional sequences belong to the same probability distribution or not. The output of the K–S test is a p-value. A p-value lower than or equal to 0.025 indicates that the given two sequences are not drawn from the same probability distribution. However, a p-value higher than 0.025 indicates that the given two sequences are statistically similar [47], [48]. Therefore, in our evaluation we divided the p-value into two regions, namely *Accepted (A)* when the p-value is greater than 0.025 and *Not Accepted (NA)* when the the p-value is less than or equal to 0.025.

TABLE I
AVERAGE p-VALUES OF TWO-SAMPLE K–S TEST USING PR WITH DIFFERENT DEGREES AND BATCH SIZES FOR USER CPU, MEMORY FREE, CONTEXT SWITCHES, AND MEMORY BANDWIDTH HARDWARE METRICS IN EXPERIMENT 1

user cpu						**context switches**					
BS	**PR2**	**PR4**	**PR6**	**PR8**	**PR10**	**BS**	**PR2**	**PR4**	**PR6**	**PR8**	**PR10**
2	0.945	0.945	0.945	0.945	0.945	**2**	1	1	1	1	1
4	0.917	0.943	0.943	0.943	0.943	**4**	0.934	1	1	1	1
8	0.645	0.769	0.821	0.758	0.758	**8**	0.679	0.864	0.953	0.999	0.999
16	0.397	0.550	0.640	0.686	0.718	**16**	0.309	0.507	0.646	0.750	0.836
32	0.125	0.212	0.268	0.298	0.324	**32**	0.075	0.147	0.204	0.241	0.269
64	0.043	0.080	0.102	0.134	0.168	**64**	~~0.011~~	0.027	0.045	0.067	0.096
128	~~0.005~~	~~0.013~~	~~0.024~~	0.033	0.042	**128**	~~0.004~~	~~0.007~~	~~0.016~~	~~0.010~~	~~0.010~~
memory free						**memory bandwidth**					
BS	**PR2**	**PR4**	**PR6**	**PR8**	**PR10**	**BS**	**PR2**	**PR4**	**PR6**	**PR8**	**PR10**
2	1	1	1	1	1	**2**	1	1	1	1	1
4	1	1	1	1	1	**4**	0.989	1	1	1	1
8	0.991	0.998	0.999	1	1	**8**	0.826	0.914	0.974	0.988	0.988
16	0.924	0.952	0.967	0.974	0.976	**16**	0.500	0.684	0.831	0.873	0.901
32	0.725	0.814	0.849	0.857	0.870	**32**	0.210	0.389	0.473	0.515	0.544
64	0.468	0.555	0.637	0.695	0.725	**64**	0.063	0.132	0.215	0.297	0.371
128	~~0.217~~	0.293	0.363	0.404	0.429	**128**	~~0.012~~	~~0.022~~	0.047	0.075	0.103

The NA values are denoted by red line.

E. Dynamic Time Warping

The K–S test can be used to see whether the generated sequence is statistically comparable to the original one, but it does not quantify the sequential similarity of the reconstructed data. Therefore, to quantify the error against the actual data, we used the DTW distance metric [21], [22]. This is a well-known method used to measure the similarity between two given sequences, which may vary in speed [22]–[26]. For instance, a dramatic increase of CPU usage of two different observations could be identified as similar even both observations speed is different for reaching an abnormal usage. A small value of the DTW test is considered good as it shows that the given two sequences are close to each other. However, a large value of the DTW test is considered bad as it indicates that the two given sequences are not close to each other. Therefore, a lower value of DTW is desirable to consider the reconstructed data similar to the actual data.

V. EXPERIMENTAL RESULTS

In order to study and validate the presented methods, we compare both techniques (PRs and MM) using the described datasets, by evaluating the aforementioned metrics (K–S, DTW, and reduction improvement). Also we compared our approach against directly applying classical data compression mechanisms.

A. Experiment 1: Data Reduction and Reconstruction Using PR

In this first experiment, we calculated the data reduction obtained from using PR models with different data batch sizes and different polynomial degrees. We then evaluated the results using a two-sample K–S test to discard inappropriate solutions. Finally we computed the DTW distance to quantify the quality of the solution. This evaluation is presented in the following subsections.

1) Two-Sample K–S Test: Table I shows the results of a two-sample K–S test. First, we reconstructed the data using the corresponding coefficients, intercept, polynomial degree, and batch sizes for each PR model stored during the data reduction phase.

TABLE II
DATA REDUCTION PERCENTAGE USING PR WITH DIFFERENT POLYNOMIAL DEGREES AND BATCH SIZES FOR EXPERIMENT 1

Batch Size	**PR2**	**PR4**	**PR6**	**PR8**	**PR10**
2	-42.86	-60.36	-77.66	-94.96	-129.43
4	-3.58	-43.35	-51.82	-60.51	-77.89
8	39.21	-3.15	-47.72	-69.19	-78.22
16	68.06	46.78	25.54	4.39	-18.92
32	82.95	75.64	69.69	65.95	60.43
64	~~90.08~~	84.90	78.62	74.28	67.39
128	~~93.08~~	~~91.80~~	~~90.11~~	~~87.18~~	~~86.26~~

The red line values are not accepted and taken from table I. The grey shaded negative values represent data growth.

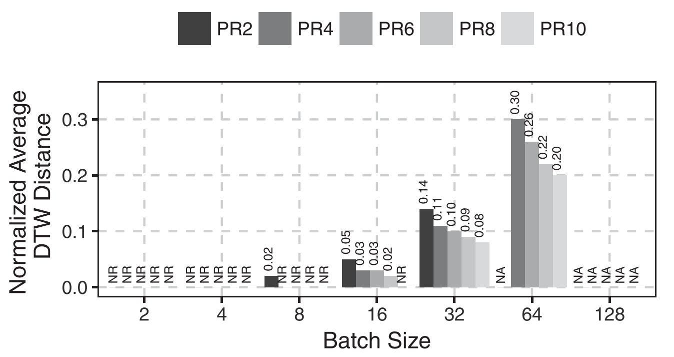

Fig. 5. Normalized average DTW distance for telemetry measures using PR models trained on different batch sizes and polynomial degrees for Experiment 1. (NR = Not Reduced, NA = Not Accepted).

Then, we apply the two-sample K–S test to compute the average p-value for each telemetry measure. A p-value lower than 0.025 is considered bad and does not reflect appropriate similarity with the actual data. However, a p-value higher than 0.025 is considered acceptable and reflecting the statistical feasibility of reconstructed data belonging to the same distribution as the actual data [47], [48]. The *NA* values are denoted by red line in the table.

Most of the PR models reconstructed the data within an acceptable range. For example, for user CPU metric, only polynomial degree 2, 4, and 6 using batch size 128 reconstructed the data with not acceptable range, while all other PR models yielded an acceptable range. In the case of *context switch* metric, all PR models with batch size 128 reconstructed the data within not acceptable range, the same was for PR2 with batch size 64, while all remaining PR models reconstructed data within an acceptable range.

2) Data Reduction Percentage: Table II shows the data reduction percentage using PR models learned for different polynomial degrees and batch sizes for Experiment 1. The negative values in the table show a growth in data size instead of reduction. The negative values are observed due to two reasons. First, whenever the batch size (i.e., the number of data points) is smaller or equal to the polynomial degree used to fit the curve, it results in learning more coefficients than the actual data points. Hence, the data size grows compared to the original data. Second, sometimes the PR models coefficients consist of high precision decimal values, which require more spaces compared to the actual data points. A higher polynomial degree model is very sensitive to the coefficient values and rounding the coefficient values significantly changes the shape of the curve.

To understand the growth of data using PR models, consider PR6 with batch size 2, which increases the data by 77.66%. This is mainly due to the fact that for every batch interval, we need to store six coefficients and one intercept (seven data points in total), while the actual data consists of only two data points. Therefore, we should avoid fitting a curve on a batch size smaller than the polynomial degree used to fit the data. Another case is to consider PR6 with batch size 8 in which data size increase by 47.74%, mainly due to the high precision of the coefficients.

We observed that PR models with a large batch size and small polynomial degrees help in reducing the telemetry data significantly. For example, batch size 64 with PR4 reduces data to 84.90%. However, such PR models may not regenerate the data with good accuracy specifically for bursty and noisy telemetry observations. Therefore, we need to identify an appropriate combination of batch size and polynomial degree to reduce the data size with higher data reconstruction accuracy.

3) DTW Distance: Fig. 5 shows the normalized average DTW distance for PR models learned using different degrees and batch sizes for Experiment 1. The batch sizes without reduction are denoted by NR and batch sizes which are in not acceptable regions are denoted by NA. We observed that on large batch sizes, the DTW distance increases and it decreases by increasing the PR degree. We observed that on batch size 8, only degree 2 (PR2) is reducing the data with 98% accuracy. The accuracy of reconstructed data at batch size 16 varies from 95% to 98% where it reduces to 70% to 80% at batch size 64.

Fig. 6 shows the reconstruction of first 300 data points of PageRank workload of our dataset using different regression models trained on degree 2 with a lower batch size (BS 8) and on degree 10 with a higher batch size (BS 64). We observed that BS 8 with degree 2 yields better reconstruction even for spikes and burstiness comparing to BS 64, but it only achieved 39.21% data reduction. However, BS 64 with degree 10 yields 67.39% data reduction, the K–S test is also acceptable, but the DTW distance for BS 64 model is higher than BS 8 model for all telemetry metrics. This confirms that PR with higher degree and higher batch size yields less accuracy in reconstructed data.

B. Experiment 2: Data Reduction and Reconstruction Using MM

In this second experiment, we calculated the data reduction obtained from using MM with different data batch sizes and different number of Markov states. We then evaluated the results using a two-sample K–S test to discard inappropriate solutions. Finally we computed the DTW distance to quantify the quality of the solution. The following subsections present the evaluation results.

1) Two-Sample K–S Test: Table III shows the results of the two-sample K–S test. Most of the MM reconstructed data are within the acceptable range except few such that the reconstructed user CPU metrics using the 2-state MM with a batch size of 128 are not within the acceptable range. The reconstructed memory free metrics using MM models with batch size 128 are not within the acceptable range. The results conclude the fact that all reconstructed telemetry metrics are

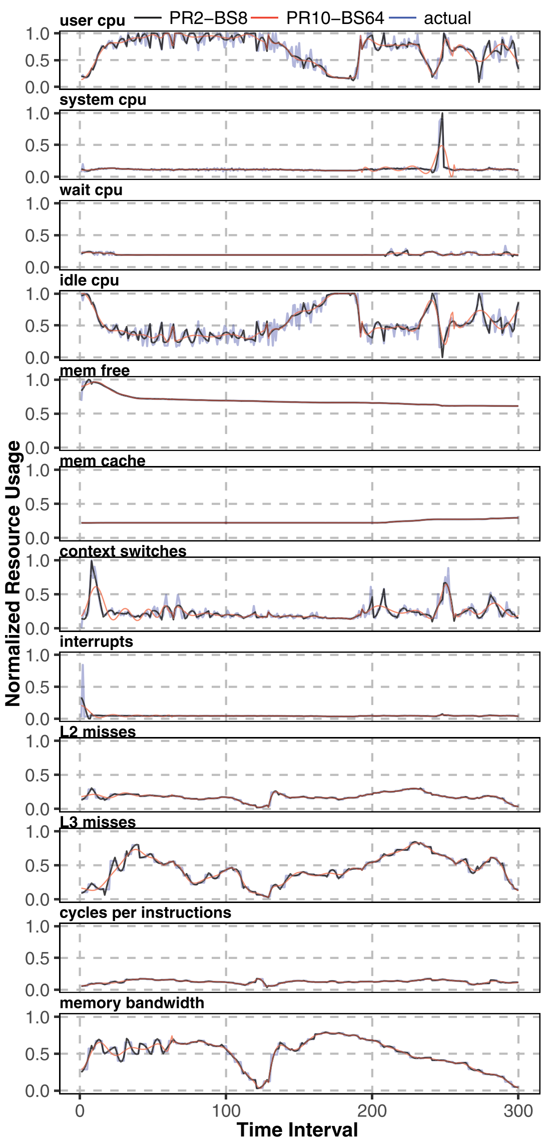

Fig. 6. Reconstruction of PageRank workload telemetry data using polynomial degree 2 with batch size 8 (PR2-BS8) and polynomial degree 10 with batch size 64 (PR10-BS64).

TABLE III
AVERAGE p-VALUES OF TWO-SAMPLE K–S TEST USING MM WITH DIFFERENT STATES AND BATCH SIZES FOR USER CPU, MEMORY FREE, CONTEXT SWITCHES, AND MEMORY BANDWIDTH HARDWARE METRICS IN EXPERIMENT 2

user cpu				**context switches**			
BS	**2MM**	**3MM**	**4MM**	**BS**	**2MM**	**3MM**	**4MM**
2	0.999	0.999	0.999	**2**	0.996	0.996	0.996
4	0.663	0.674	0.663	**4**	0.745	0.766	0.775
8	0.403	0.401	0.408	**8**	0.579	0.622	0.656
16	0.281	0.285	0.294	**16**	0.408	0.485	0.543
32	0.181	0.203	0.215	**32**	0.237	0.331	0.396
64	0.076	0.113	0.126	**64**	0.107	0.189	0.246
128	~~0.023~~	0.037	0.060	**128**	~~0.013~~	0.053	0.092
memory free				**memory bandwidth**			
BS	**2MM**	**3MM**	**4MM**	**BS**	**2MM**	**3MM**	**4MM**
2	1	1	1	**2**	1	1	1
4	1	1	1	**4**	0.800	0.843	0.892
8	0.998	0.998	0.998	**8**	0.271	0.328	0.354
16	0.937	0.927	0.906	**16**	0.115	0.139	0.129
32	0.658	0.580	0.545	**32**	0.082	0.068	0.072
64	0.124	0.088	0.063	**64**	0.057	0.053	0.051
128	~~0.002~~	~~0.003~~	~~0.004~~	**128**	0.026	0.028	0.034

The NA values are denoted by red line.

TABLE IV
DATA REDUCTION PERCENTAGE USING MM WITH DIFFERENT STATES AND BATCH SIZES

Batch Size	**2MM**	**3MM**	**4MM**
2	-30.15	-106.06	-198.15
4	39.00	2.38	-42.06
8	70.05	53.08	31.50
16	83.09	75.13	63.77
32	90.28	84.64	79.38
64	92.81	91.30	87.30
128	~~93.95~~	~~93.16~~	~~92.16~~

The red line values are NA and taken from Table III. The gray shaded negative values represent data growth.

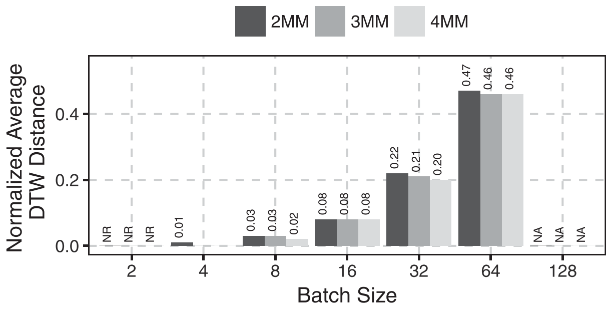

Fig. 7. Normalized average DTW distance for MM trained on different batch sizes and states for telemetry measures. (NR = Not Reduced, NA = Not Accepted).

within the acceptable range when using MM with 2, 3, and 4 states and up to 64 batch size.

2) Data Reduction Percentage: Table IV shows the data reduction percentage for different batch sizes with 2, 3, and 4 state MM. The negative values in the table show a data growth instead of reduction. The negative values are observed whenever the number of data points learned as part of MM model turns higher than actual data points.

We observed that large batch sizes and small state value yield higher data reduction. For example, batch size 64 with 2 states yields 92.81% data reduction. For 2 and 3 states MM, we start observing data reduction after batch size greater than 2. However, for 4-state MM, we observe data reduction for batch sizes greater than 4. Hence, the data reduction depends on the size of the transition probability matrix and the state interval matrix. For example, we need 2 times 2 matrix to store state transition probabilities and 3 data points to represent the state interval matrix. Therefore, for 2-state MM, we need at least seven data points to represent a given batch. We observed 39% data reduction on batch size 4 when the original data points are 4 and MM data points are 7. The reason of this reduction is that some of the telemetry metrics have at maximum nine digits in their actual data, e.g., context switches, interrupts whereas state transition probability matrix contains only counts of moving from one state to another. Thus it requires less storage even if MM data points are higher than the original batch size. This count is later converted into probabilities whenever reconstruction is required.

3) DTW Distance: Fig. 7 shows the normalized average DTW distance for MM learned using different degrees and batch sizes for Experiment 2. The batch sizes without reduction are denoted by NR and batch sizes which are in not acceptable regions

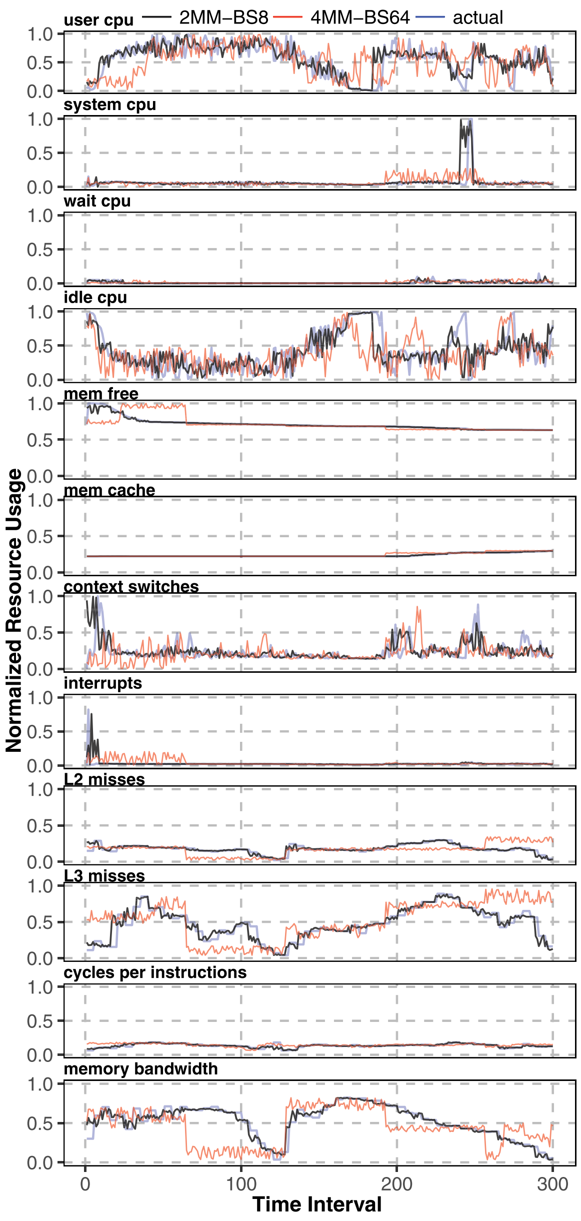

Fig. 8. Reconstruction of PageRank workload telemetry data using 2-state MM with batch size 8 (2MM-BS8) and 4 state MM with batch size 64 (4MM-BS64).

are denoted by NA. We observed that on large batch sizes the DTW distance increases. We also observed that on batch size 4, only 2-state MM model (2MM) is reducing the data with 99% accuracy. The accuracy of reconstructed data at batch size 8 varies from 97% to 98% and at batch size 16, it is 92% where it reduces to 53% to 54% at batch size 64. The MM models do not show any effect on telemetry measurements which do not contain spikes, bursts, or noise. For example, free memory (MEM free) telemetry measurement of our dataset does not have any effect on DTW using different MM models trained on different MM states and batch sizes. Therefore, such type of telemetry measurement can be reconstructed using a small number of MM states.

Fig. 8 shows the reconstruction of first 300 data points of PageRank workload of our dataset using different MM trained using 2 states and a batch size of 8 (2MM-BS8) and 4 states

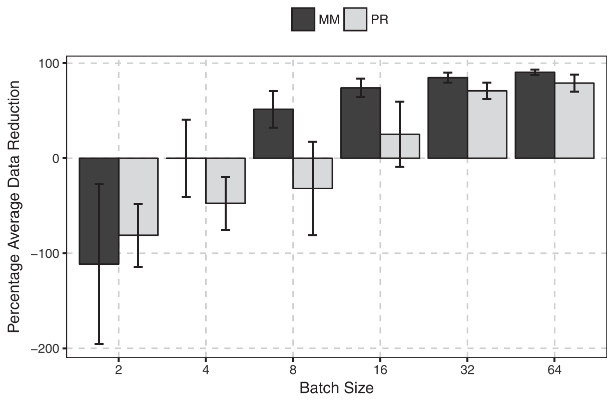

Fig. 9. Comparison of average data reduction percentage using PR and MM methods for telemetry data reduction.

and a batch size of 64 (4MM-BS64). We observed that BS 8 with 2-state MM reconstructed the data appropriately. However, with 4-states model with batch size 64 does not show good reconstruction. The data reduction using 2MM-BS8 model only reduces 70.05% data, while 4MM-BS64 reduces 87.30% of the data. The K–S test is also acceptable for both of these models, while the DTW distance for the BS64 model is higher than the BS8 for all telemetry metrics.

Using a small batch size (BS8) helps to capture the data patterns, well including spikes and burstiness compared to a large batch size (BS64). It shows that spikes and noise in the original data cannot be well captured using larger batch sizes. For example, system CPU had few spikes around 270 s in the actual data. These spikes are well captured using batch size 8, however batch size 64 does not capture these spikes well. The reason for this behavior is mainly due to the fact that MM depends on the state transition matrix, which contains the probabilities of moving from one state to another. Therefore, if there is burstiness in a specific batch, then the probability of having burstiness remains for the whole batch interval. Thus the performance of the MM method in data reconstruction is not always robust using large batch size. We conclude that, if data contains spikes and burstiness then we should use smaller batch sizes with MM method. However, if we can detect that data do not contain spikes and burstiness, then we can use higher batch sizes with MM method. This behavior is well observed in memory free and memory cache telemetry metrics shown in Fig. 8.

C. PR and MM Comparison

To compare the PR and MM methods, we compute the average percentage data reduction and the average DTW distance for all settings (polynomial degrees and states) of these methods on different batch sizes. In this section, we show the comparison of PR and MM methods for data reduction and reconstruction.

Fig. 9 shows the average data reduction on different batch sizes for all settings using PR and MM methods. In average, the batch sizes less than 8 do not yield any data reduction, but instead, they cause data growth. However, a significant gain of 51.54% is observed using MM method on batch size 8 compared to 31.81% for the PR method. For batch sizes higher than 8, MM always outperforms PR in data reduction.

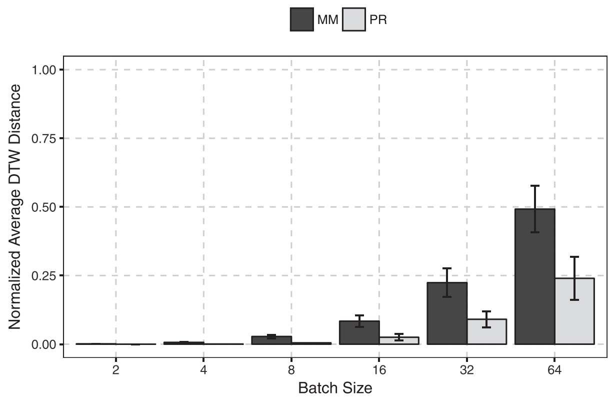

Fig. 10. Comparison of normalized DTW distance for data reconstruction using PR and MM methods.

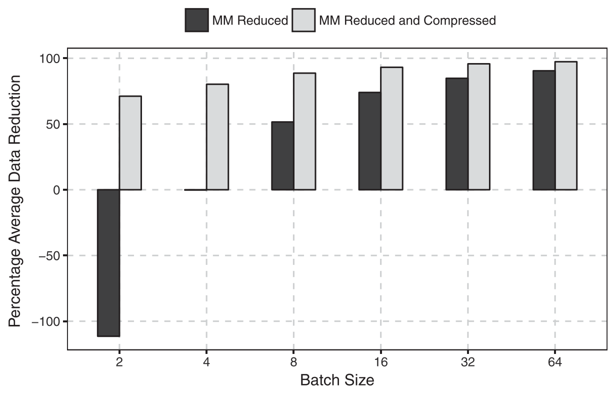

Fig. 11. Comparison of MM Reduced data with MM Reduced and Compressed data. The MM Reduced is the data obtained by applying MM method and MM Reduced and Compressed is the data obtained after applying ZIP compression on the reduced data obtained from MM reduction method.

Fig. 10 shows the average normalized DTW distance for PR and MM method on different batch sizes. In average, the batch sizes less than 16 yields very low DTW distance, less than 0.2, for both PR and MM methods, which reflects a good similarity of the reconstructed data with the actual data. For large batch sizes, e.g., 32 and 64, the PR method outperforms MM in data reconstruction.

Large batch sizes reduce data significantly, however, they perform poorly in data reconstruction similarity. Therefore, from Figs. 10 and 9, we conclude that batch size 16 is appropriate to use with both PR and MM methods because we obtain 25.17% and 74% average data reduction for both PR and MM, respectively, with DTW distance below 0.2 for both methods. However, we prefer to use MM model, mainly due to higher data reduction on batch size 16 although PR method has slightly better DTW.

D. Data Reduction Using ZIP Compression

After comparing the two proposed methods, we evaluate them against the usage of classic lossless compression methods, such as ZIP algorithms. We compared the ZIP algorithm on raw data with applying the ZIP algorithm on the reduced data to see the improvement or overheads.

Fig. 11 shows the comparison of data reduced using MM (MM Reduced) and with the data obtained after applying ZIP compression on the reduced data (MM Reduced and Compressed) for different batch sizes. We observed the most significant data reduction due to ZIP compression in lower batch sizes particularly 2, 4, and 8. This is because, the reduced data contains the count of transition in the transition matrix, which is either 0 or 1 in case of 2 states, 0, 1, or 2 in case of 4 states and 0 to 7 in case of 8 states. However, most of the values in these matrices consist of 0 due to no transition from one state to other, hence the zip compression further reduces the data. After batch size 32, we observed that the difference between MM Reduced and MM Reduced Compressed data is less than 8% on average. The maximum data reduction with zip compression is observed with 2-state MM model, which is 98.24% on batch size 64. The ZIP compression helps to further reduce the data, but for large batch sizes, the effect of ZIP compression is not significant.

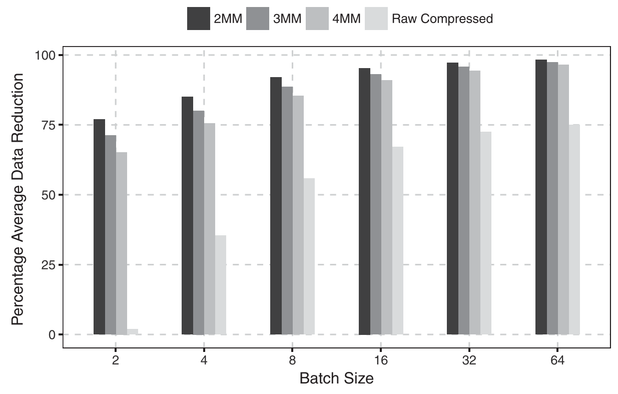

Fig. 12. Comparison of ZIP compression on raw data and ZIP compression on data reduced using MM method.

TABLE V
PERCENTAGE BANDWIDTH REDUCTION WITHIN DATA CENTER USING MM METHOD

Batch Size	2MM	3MM	4MM
2	-100	-212.50	-350
4	0	-56.25	-125
8	50	21.88	-12.50
16	75	60.94	43.75
32	87.50	80.47	71.88
64	93.75	90.23	85.94

Then, using a standard lossless ZIP compression algorithm as a baseline, Fig. 12 shows the comparison of ZIP compression on the raw data with the reduced and compressed data (MM Reduced Compressed) using MM method on varying batch sizes. By just applying ZIP compression on the row data, we achieved 1.88% to 75.02% data reduction for batch size 2 to 64, however our proposed solution yields reduction from 76.95% to 96.48% with 2 and 4 state MM model from batch size 2 to 64, respectively.

E. Bandwidth Reduction Using MM

Table V shows the bandwidth utilization reduction percentage using the proposed solution with 2, 3, and 4 state MM for telemetry data collection within the data center. We consider 4 B float data type to represent the value of actual data and state interval matrix. We observed that for the highest acceptable batch (batch size 64), the state transition matrix can have a maximum

TABLE VI
Comparison of Raw and Reduced Storage (GB) for 30 D and Bandwidth (Kbps) Using 2-State MM With Batch Size 16

Data Center Type	Total Racks	Nodes per Rack	Total Telemetry Sources	30 days Storage (GB)			Bandwidth (Kbps)			
				Raw Uncompressed	Compressed	Reduced Compressed	Raw Uncompressed at Rack	Reduced at Rack	Raw Uncompressed at Datcenter	Reduced at Data Center
small	100	30	300,000	5,793.57	1,158.71	270.56	187.50	46.88	18,750	4,687.50
medium	250	40	1,000,000	19,311.90	3,862.38	901.87	250	62.50	62,500	15,625
large	500	60	3,000,000	57,935.71	11,587.14	2,705.60	375	93.75	187,500	46,875

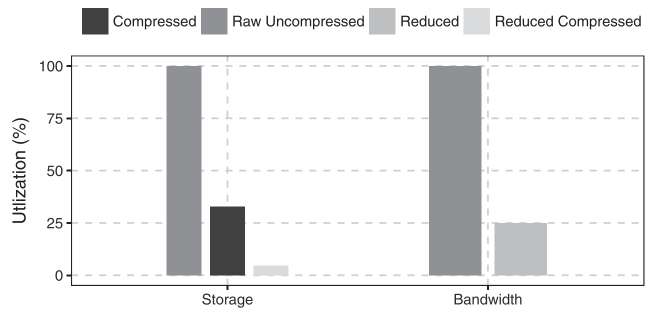

Fig. 13. Comparison of storage usage percentage for raw uncompressed, compressed and reduced compressed (left side). Comparison of bandwidth utilization percentage for raw uncompressed and reduced data (right side).

value of 63 as it contains the count of transition from one state to other. This type of data can be sent using 1 B over the network which reduces the bandwidth utilization significantly. Moreover, higher batch sizes provide a notable reduction in bandwidth utilization within the data center. For example, batch size 32 and 64 yields significant bandwidth reduction varying from 71.88% to 93.75% for different number of MM states.

VI. Benefits of the Proposed Solution in Different Data Centers

Typical telemetry metrics consist of hardware performance counters up to a hundred events [49] related to CPU, memory, network, disk, temperature, etc. In this section, we study the effect on bandwidth utilization and storage space using the proposed solution to monitor and collect the telemetry data in the different sizes of data centers. We considered three different data centers namely small, medium, and large in which we assume that 100 different telemetry metrics are collected from each computing node after a discrete time intervals. A typical telemetry metric takes 8 B to store the information including the timestamp. We selected 2-state MM model with batch size 16 as our purposed solution because we achieve 95.33% of reduction in storage, 75% of reduction in bandwidth with 92% of accuracy, and Fig. 13 shows the comparison of percentage storage space required for raw uncompressed, compressed, and reduced compressed data and it also shows the percentage bandwidth utilization for raw uncompressed and reduced data for the purposed solution.

Table VI shows the storage and bandwidth reduction for all three type of data centers and highlights the merits of the proposed solution. A small data center with 3000 computing nodes, which are deployed on 100 racks. Where each rack hosts 30 computing nodes. To store one day of telemetry data, we require 193.11 GB storage, and in a month that storage requirements increase to 5.65 TB. To store one day of telemetry data on a medium sized data center, e.g., counting with 10 K computing nodes, deployed on 250 racks (considering 40 nodes per rack), we require 643.73 GB storage, and in a month, the storage requirements increase to 18.85 TB. Similarly, for a large data center, e.g., with 30 K computing nodes deployed on 500 racks, we will require 1931.19 GB storage space for one day, and in a month, the storage requirements increase to 56.57 TB.

The proposed method of 2-state MM with a batch size of 16 reduced the storage requirement to 0.26, 0.88, and 2.64 TB for small, medium, and large-scale data centers, respectively, with a 92% reconstruction accuracy. The actual rack level bandwidth utilization for small, medium, and large data centers are 187.50, 250, and 375 Kbps, respectively, which are significantly reduced to 46.88, 62.50, and 93.75 Kbps. Similarly, the actual data center level bandwidth utilization for small, medium, and large sizes are 18 750, 62 500, and 18 7500 Kbps, respectively. Our purposed system reduces this bandwidth utilization to 4687.50, 15 625, and 46 875 Kbps for small, medium, and large scale data centers, respectively.

VII. Conclusion and Future Work

Data centers generate a lot of telemetry data, which is used for many purposes, including resources management, analytics, and optimization. However, the size of telemetry data grows dramatically and considerably increases the storage space and bandwidth utilization within the data center. In this paper, we proposed a Markov chain based method to reduce telemetry data to minimize bandwidth utilization and storage space required to store it within the data center. Our solution outperforms the baseline method based on PR method to reduce and regenerate the telemetry data. We extensively evaluated the effect of batch sizes, number of states in MM, and polynomial degrees in PR. We observed that a larger batch size effectively reduces data, but the reconstruction accuracy is lower. Therefore, we identified that a batch size between 16 and 64 is appropriate to use for data reduction with better reconstruction accuracy. Our experimental evaluation shows that PR-based method required more storage space as compared to Markov chain model based method due to the high precision of coefficients. We also observed that 95.33% storage space and 75% bandwidth utilization can be reduced with 92% accuracy using the proposed solution.

For future work, we are focusing on adaptively identifying the batch size and the number of states in MM to further reduce space and increase the reconstruction accuracy. We also plan to use one MM per metric at the data center level for recurring workloads having similar resource usage requirements. Moreover, we intend to investigate mixture models for telemetry data reduction and reconstruction.

References

[1] D. A. Reed, J. Dongarra, "Exascale computing and big data," *Commun. ACM*, vol. 58, no. 7, pp. 56–68, 2015.

[2] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," in *Proc. VLDB Endow.*, vol. 5, no. 12, 2012, pp. 2032–2033.

[3] W. Iqbal, M. N. Dailey, and D. Carrera, "Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1435–1446, Dec. 2016.

[4] F. Seracini, X. Zhang, T. Rosing, and I. Krüger, "A proactive customer-aware resource allocation approach for data centers," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2014, pp. 26–33.

[5] R. Birke, L. Y. Chen, and E. Smirni, "Usage patterns in multi-tenant data centers: A temporal perspective," in *Proc. 9th Int. Conf. Autonomic Comput.*, 2012, pp. 161–166.

[6] M. Kutare, G. Eisenhauer, C. Wang, K. Schwan, V. Talwar, and M. Wolf, "Monalytics: Online monitoring and analytics for managing large scale data centers," in *Proc. 7th Int. Conf. Autonomic Comput.*, 2010, pp. 141–150.

[7] L. Zhan, T. Z. Fu, D. M. Chiu, and Z. Lei, "A framework for monitoring and measuring a large-scale distributed system in real time," in *Proc. 5th ACM Workshop HotPlanet*, New York, NY, USA, 2013, pp. 21–26.

[8] M. S. Karabinaoğlu and T. Gözel, "Load forecasting modelling of data centers and IT systems by using artificial neural networks," in *Proc. 10th Int. Conf. Elect. Electron. Eng.*, 2017, pp. 62–66.

[9] L. Duan, D. Zhan, and J. Hohnerlein, "Optimizing cloud data center energy efficiency via dynamic prediction of CPU idle intervals," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, 2015, pp. 985–988.

[10] J. Xue, R. Birke, L. Y. Chen, amd E. Smirni, "Spatial-temporal prediction models for active ticket managing in data centers," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 39–52, Mar. 2018.

[11] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, and V. C. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1688–1699, Jun. 2018.

[12] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host cpu utilization in cloud computing using recurrent neural networks," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans.*, 2017, pp. 67–72.

[13] O. Rolik, E. Zharikov, A. Koval, and S. Telenyk, "Dynamie management of data center resources using reinforcement learning," in *Proc. 14th Int. Conf. Adv. Trends Radioelectronics, Telecommun. Comput. Eng.*, 2018, pp. 237–244.

[14] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2016, pp. 730–739.

[15] M. Burtscher and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 18–31, Jan. 2009.

[16] J. Iverson, C. Kamath, and G. Karypis, "Fast and effective lossy compression algorithms for scientific datasets," in *Proc. Eur. Conf. Parallel Process.*, 2012, pp. 843–856.

[17] K. Sayood, *Introduction to Data Compression*. Burlington, Massachusetts, San Francisco, CA: Morgan Kaufmann, 2017.

[18] S. Karlin, *A First Course in Stochastic Processes*. Cambridge, Massachusetts: Academic Press, 2014.

[19] C. Carvalho, L. Leal, M. Lemos, and R. Holanda, "Avoiding data traffic on smart grid communication system," in *Proc. Int. Electron. Conf. Sensors Appl.*, 2014, pp. 1–16.

[20] F. J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *J. Amer. Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[21] P. Senin, "Dynamic time warping algorithm review," Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, USA, Tech. Rep. 855, 2008.

[22] M. Müller, *Information Retrieval for Music and Motion*, vol. 2. Salmon Tower Building, New York: Springer, 2007.

[23] J. Gu and X. Jin, "A simple approximation for dynamic time warping search in large time series database," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.*, 2006, pp. 841–848.

[24] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *Proc. KDD Workshop*, vol. 10, Seattle, WA, USA, 1994, pp. 359–370.

[25] T. Rakthanmanon *et al.*, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 262–270.

[26] T. Oates, L. Firoiu, and P. R. Cohen, "Clustering time series with hidden Markov models and dynamic time warping," in *Proc. IJCAI Workshop on Neural, Symbolic and Reinforcement Learn. Methods for Sequence Learn.*, 1999, pp. 17–21.

[27] M. Kim, "Time-series dimensionality reduction via Granger causality," *IEEE Signal Process. Lett.*, vol. 19, no. 10, pp. 611–614, Oct. 2012.

[28] D. Littau and D. Boley, "Streaming data reduction using low-memory factored representations," *Inf. Sci.*, vol. 176, no. 14, pp. 2016–2041, 2006.

[29] G. Gawde and J. Pawar, "Shape based time series reduction using PCA," in *Proc. Int. Conf. Innov. Inf., Embedded Commun. Syst.*, 2017, pp. 1–4.

[30] T. Sun, H. Sun, and W. Chen, "Dimensionality reduction for interval time series," in *Proc. World Congr. Inf. Commun. Technol.*, 2012, pp. 1115–1120.

[31] J. Wang, S. Yue, X. Yu, and Y. Wang, "An efficient data reduction method and its application to cluster analysis," *Neurocomputing*, vol. 238, pp. 234–244, 2017.

[32] S. Tripathi and S. De, "An efficient data characterization and reduction scheme for smart metering infrastructure," *IEEE Trans. Ind. Inf.*, vol. 14, no. 10, pp. 4300–4308, Oct. 2018.

[33] Z. C. Dagdia, C. Zarges, G. Beck, and M. Lebbah, "A distributed rough set theory based algorithm for an efficient big data pre-processing under the spark framework," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2017, pp. 911–916.

[34] J. Yan *et al.*, "Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 3, pp. 320–333, Mar. 2006.

[35] M. Z. A. Bhuiyan, G. Wang, W. Tian, M. A. Rahman, and J. Wu, "Content-centric event-insensitive big data reduction in Internet of Things," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–6.

[36] K. Wu, D. Lee, A. Sim, and J. Choi, "Statistical data reduction for streaming data," in *Proc. Scientific Data Summit*, New York, NY, USA, 2017, pp. 1–6.

[37] A. Egri, I. Horváth, F. Kovcs, R. Molontay, and K. Varga, "Cross-correlation based clustering and dimension reduction of multivariate time series," in *Proc. IEEE 21st Int. Conf. Intell. Eng. Syst.*, 2017, pp. 000241–000246.

[38] X. Peng and B. Pernici, "Correlation-model-based reduction of monitoring data in data centers," in *Proc. 5th Int. Conf. Smart Cities Green ICT Syst.*, 2016, pp. 1–11.

[39] T. Yu, X. Wang, and A. Shami, "A novel fog computing enabled temporal data reduction scheme in IoT systems," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–5.

[40] M. F. Mohamed, M. El-Gayyar, A. E.-R. Shabayek, and H. Nassar, "Data reduction in a cloud-based AMI framework with service-replication," *Comput. Elect. Eng.*, vol. 69, pp. 212–223, 2018.

[41] K. Goodhope *et al.*, "Building Linkedin's real-time activity data pipeline," *IEEE Data Eng. Bull.*, vol. 35, no. 2, pp. 33–45, 2012.

[42] J. Kreps *et al.*, "Kafka: A distributed messaging system for log processing," in *Proc. NetDB*, 2011, pp. 1–7.

[43] L. Liu, O. Hasan, and S. Tahar, "Formal reasoning about finite-state discrete-time Markov chains in HOL," *J. Comput. Sci. Technol.*, vol. 28, no. 2, pp. 217–231, 2013.

[44] K. S. Trivedi and A. Bobbio, *Reliability and Availability Engineering: Modeling, Analysis, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[45] J. P. D. C. Shuja-ur Rehman Baig and Marcelo Amaral, "Performance characterization of spark workloads on shared NUMA systems," in *Proc. IEEE 4th Int. Conf. Big Data Comput. Service Appl.*, 2018, pp. 41–48.

[46] Spark-bench. [Online]. Available: https://github.com/SparkTC/spark-bench

[47] One and two-tailed tests. [Online]. Available: https://en.wikipedia.org/wiki/One-_and_two-tailed_tests

[48] R. A. Fisher, *Statistical Methods for Research Workers*. Guildford, U.K.: Genesis Publishing Pvt Ltd, 1925.

[49] D. Terpstra, H. Jagode, H. You, and J. Dongarra, "Collecting performance data with PAPI-C," in *Tools for High Performance Computing 2009*. Springer, 2010, pp. 157–173.