Anomaly Detection in Computer Systems using Compressed Measurements

Tingshan Huang, Nagarajan Kandasamy, Harish Sethu
ECE Department, Drexel University
Philadelphia, PA 19104, USA
Email: {th423, kandasamy, sethu}@drexel.edu

***Abstract*—Online performance monitoring of computer systems incurs a variety of costs: the very act of monitoring a system interferes with its performance and if the information is transmitted to a monitoring station for analysis and logging, this consumes network bandwidth and disk space. Compressive sampling-based schemes can help reduce these costs on the local machine by acquiring data directly from the system in a compressed form, and in a computationally efficient way. This paper focuses on reducing the computational cost associated with recovering the original signal from the transmitted sample set at the monitoring station for anomaly detection. Towards this end, we show that the compressed samples preserve, in an approximate form, properties such as mean, variance, as well as correlation between data points in the original full-length signal. We then use this result to detect changes in the original signal that could be indicative of an underlying anomaly such as abrupt changes in magnitude and gradual trends without the need to recover the full-length data. We illustrate the usefulness of our approach via case studies involving IBM's Trade Performance Benchmark using signals from the disk and memory subsystems. Experiments indicate that abrupt changes can be detected using a compressed sample size of 25% with a hit rate of 95% for a fixed false alarm rate of 5%; trends can be detected within a confidence interval of 95% using a sample size of only 6%.**

***Index Terms*—Online monitoring, anomaly detection, compressive sampling, principal component analysis.**

I. Introduction

Online monitoring of performance-related metrics is a necessary first step towards detecting anomalies in computer systems. Measurements may include high-level metrics such as response time and throughput as well as low-level ones such as processor utilization, disk I/O, memory, and network activity. The monitored information helps detect performance-related hotspots and bottlenecks as well as incipient faults associated with gradual resource exhaustion—the so-called software aging problem [1]–[3]. In the case of intermittent problems that are hard to isolate, browsing back through historical data can help identify and localize recurring problems affecting the same portion of the computing infrastructure at different times. The data can also help detect security breaches resulting in the computers being infected by malicious software [4].

We consider a server cluster wherein software-based sensors embedded within the infrastructure measure various performance-related parameters associated with the cluster. The measured information is transmitted over a network to a monitoring station for data analysis and visualization. Online monitoring, however, incurs a variety of costs. First, the very act of monitoring an application interferes with its performance. If sensing-related code is merged with the application code, this change may interfere with the timing characteristics of the application or if sensors execute as separate processes, they contend for CPU resources along with the original application. Transmitting the monitored data over a network consumes bandwidth. Finally, logging the data for future analysis consumes disk space. So, when monitoring a large-scale computing system it is desirable to minimize the above-described costs.

Traditional methods of sampling signals use Shannon's theorem: the sampling rate must be at least twice the signal bandwidth to capture all the information content present in the signal. The theory of *compressive sampling* (*CS*) states, however, that we can recover a certain class of signals from the original measurements using far fewer samples than that used by techniques that rely on Shannon's theorem [5]–[8]. In previous work reported in [9]–[11], we have developed CS-based sampling strategies for the performance monitoring of computing systems that can acquire signals of interest from the underlying system *directly* in a compressed form. The methods exploit the fact that the signals often can be sparsified—that is, encoded concisely—under an appropriate representation basis and that the sampling rate itself can be tuned as a function of sparsity. We show experimentally that the recovered signals can be used to detect, with high confidence, the existence of trends within the original signal as well as abrupt changes where the signal's magnitude exceeds some threshold value. Detection of these anomalies is achieved using a substantially reduced sample size—a reduction of more than 70% when compared to the standard fixed-rate sampling method.

Compressive sampling allows for a very simple sampling strategy on the local machine. Rather than tailoring the sensing scheme to capture specific properties in the underlying signal, a *signal-independent strategy* such as randomized sampling can be used, significantly reducing the intrusion of monitoring on application performance [11]. Also, since signals are acquired directly in compressed form, the network bandwidth needed to transmit these few samples to the monitoring station is reduced and so is the hard-disk space required to store them.

When operators wish to analyze the original signal, there is a way to use numerical optimization to reconstruct the full-length signal from the sample set. The reconstructed signal allows for real-time anomaly detection and diagnosis, and also helps drive decisions of a longer-term nature such as intelligent

978-1-5090-0406-5/15/$31.00 ©2015 IEEE

capacity planning. This paper focuses on reducing the computational cost incurred by the monitoring station, associated with recovering the original signal from the sample set and analyzing it for anomalies. The recovery process is typically posed as a linear programming (LP) problem and solved under some sparsity assumptions using a class of reconstruction algorithms called basis pursuit or iterative hard thresholding pursuit [12]. Though modern LP solvers are quite efficient, each monitoring station may be responsible for recovering and analyzing hundreds of signals belonging to many servers, making it important to reduce the corresponding overhead. The paper makes the following contributions towards this goal:

- We prove from a theoretical viewpoint that the compressed samples preserve statistical properties of the original data such as variance and mean. This result allows for the detection of abrupt changes and trends by *directly analyzing* just the compressed samples without having to reconstruct the full-length signal.
- Since the sampling process is just a linear projection of the original data, we also prove that the compressed samples approximately *preserve spectral properties* such as correlation between data points, the length of the data vectors, as well as the distance between two vectors, under such a projection. This result allows for well-known anomaly detection methods such as principal component analysis (PCA) to be used directly on the compressed samples; performance is almost equivalent to the case in which the raw data is completely available.

We illustrate the usefulness of the approach via case studies using IBM's Trade Performance Benchmark (also known as Trade6). We measure signals from the disk and memory subsystems using a CS-based sampling strategy, and analyze the compressed samples for possible anomalies. The first scenario involves detecting abrupt changes in the signal during which the magnitude exceeds some nominal threshold value. In the second scenario, we wish to detect the gradual deterioration of system performance, say over hours or days, associated with software aging by statistically analyzing the appropriate signals for the existence of trends [2], [13]. We use a long-running Trade6 application having a small memory leak and evaluate the ability of the approach to estimate a positive slope in the compressed data even in the presence of seasonal variations and periodicity in the signal. Finally, we evaluate the efficacy of applying PCA to the compressed samples to detect abrupt changes in the signal.

Abrupt changes can be detected using a sample size of 25% with a hit rate of 95% for a fixed false alarm rate of 5%; trends can be detected with a confidence interval of 95% using a sample size of 6%. Finally, the hit rate achieved by the PCA-based analysis when using the compressed samples to detect abrupt changes is higher than 95% when the false alarm is fixed at 0.5%. The corresponding sample size is about 18%. These results point to the feasibility of adopting a two-step anomaly detection process at the monitoring station: the received compressed data is examined for possible anomalies; if one is suspected, the relevant portion of the signal is fully reconstructed to localize and further analyze the anomaly.

The paper is organized as follows. Section II describes our experimental setup and familiarizes the reader with compressive sampling. Section III describes the theoretical basis behind being able to detect anomalies in compressed data and Section IV presents the case studies that evaluate the performance of the approach. Section V discusses related work and Section VI provides some concluding remarks.

II. Preliminaries

We describe our experimental setup and familiarize the reader with the basic concepts underpinning compressive sampling and the subsequent signal reconstruction.

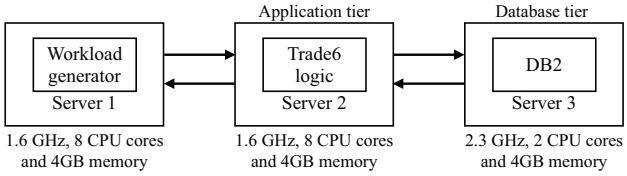
A. Experimental Settings

Figure 1(a) shows the system setup used in our experiments, comprising three servers networked via a gigabit switch. Virtualization of this system is enabled by VMWare's ESX Server running a Linux RedHat kernel. The operating system on the virtual machine (VM) is the SUSE Enterprise Linux Server Edition. The system hosts IBM's Trade6 benchmark, a stock-trading application which allows users to browse, buy, and sell stocks. Users can perform dynamic content retrieval as well as transaction commitments requiring database reads and writes, respectively. The application logic for Trade6 resides within the IBM WebSphere Application Server, which in turn is hosted by the VM on the server within the application tier. The database component is DB2, hosted on the server running SUSE Enterprise Linux. The database maintains 500 user accounts and information for 3500 stocks.

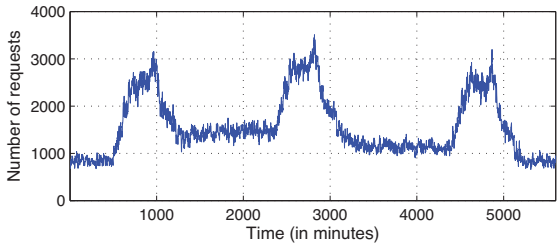
We use httperf [14], an open-loop workload generator, to send a mix of buy/browse transactions to the Trade6 application over a period of 48 hours. The workload traces are synthesized to reflect realistic operating scenarios such as time-of-day variations as well as bursty traffic where request rates vary significantly within short time periods. A sample workload is shown in Fig. 1(b), having an average arrival rate of 50 requests per second with a 50/50 mix of buy/browse transactions. Each data point in the figure represents the aggregated workload within a 30-second interval.

Our experiments use the following metrics contained within the `/proc` pseudo file system at the application tier, specifically the contents of `/proc/meminfo` that report real-time information about memory usage in Linux systems:

- *MemFree*. This quantity reflects the amount of physical memory left unused in the system.
- *CommittedAS*. This quantity reflects the total amount of memory allocated by processes in the system using `malloc()` calls, even if the memory has not been used by them as of yet. For example, a process may allocate 1 GB of memory but only touch 100 MB of it. Although the current memory usage is only 100 MB, the 1 GB allocation is memory that has been committed by the memory subsystem to the process and can be used at any time by the process.
- *PageTables*. This quantity reflects the amount of memory dedicated to the lowest level of page tables.
- *AnonPages*. This quantity tracks the amount of anonymous or non-file backed pages mapped to page tables responsible for the user space.

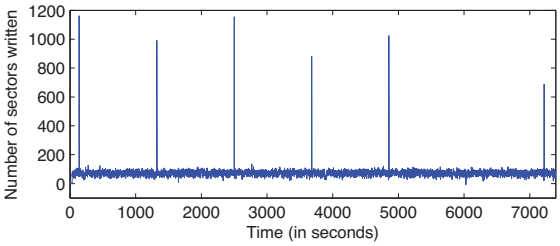


(a) The Trade6 application.

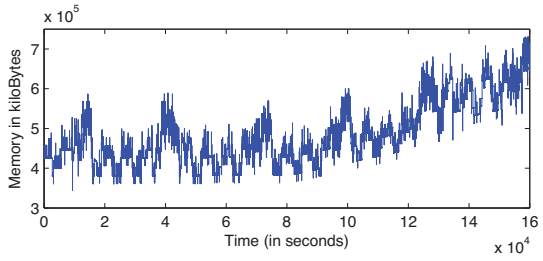


(b) Dynamic workload trace provided to the Trade6 application.

Fig. 1. The system architecture hosting the Trade6 application and an example of the workload trace provided to the testbed in our experiments. Incoming requests are plotted in granularity of 30 seconds.



(a) Disk sectors written, the *write_activity* signal.



(b) Memory allocated to proesses, the *CommittedAS* signal.

Fig. 2. Signals corresponding to: (a) I/O activity collected at the database tier showing the number of sectors written to disk and (b) total amount of memory allocated by processes in the system. Note that a memory leak has been injected around the 24 hour mark.

In addition to these features, we also use disk I/O activity measurements (sectors read/written) collected at the database tier as part of our evaluation. Our goal is to show that the compressed samples can be used to detect various system anomalies. The above-listed features were chosen to support one of the case studies: detection of memory leaks. Here we have chosen low-level metrics that are most likely impacted by this fault. Figure 2 plots two of the features collected during an experimental run of the system lasting 48 hours. The data points are sampled once every two seconds.

B. Compressed Sampling of Signals

The fundamental premise behind signal compression is that many natural signals are sparse in that they have concise representations when expressed in the proper basis; this sparsity determines the quality of the subsequent reconstruction. Using the data collected from our testbed, we show how to find basis functions in which this data can be most concisely represented.

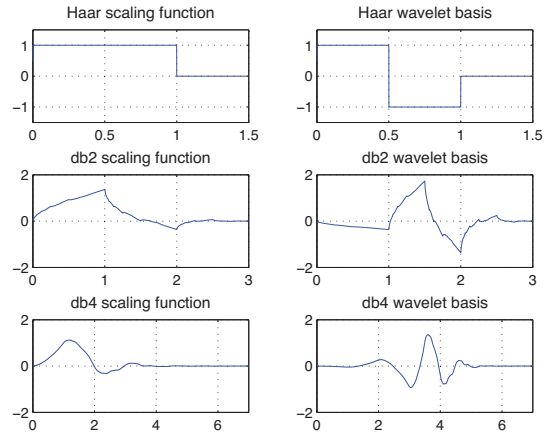


Fig. 3. Waveforms corresponding to three members of the Daubechies wavelet family. The Haar is the simplest wavelet that captures discontinuities in the data; db2 and db4 also show similarity with our data but have longer waveforms, leading to better frequency resolution.

TABLE I
PERCENTAGE OF COEFFICIENTS NEEDED TO KEEP THE RELATIVE ERROR WITHIN 1% UNDER EACH REPRESENTATION BASIS.

Signal	Haar	db2	db4
AnonPages	0.20%	0.51%	0.66%
Mapped	0.16%	0.37%	0.46%
CommittedAS	0.85%	1.27%	1.49%
PageTables	0.57%	0.64%	0.83%

Let us denote the data to be sampled as $\mathbf{d}$, a vector of length N, and its representation in basis $\mathbf{B}$ as $\mathbf{x}$. In other words, $\mathbf{d} = \sum_{i=1}^{N} x_i \mathbf{b}_i = \mathbf{Bx}$, where $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N]$. For example, if $\mathbf{B}$ is selected to be the Haar wavelet basis, the elements of the vector $\mathbf{x}$ are coefficients of the wavelet decomposition for signal $\mathbf{d}$. Also, if at most S entries in $\mathbf{x}$ are nonzero, then $\mathbf{x}$ is called an S-sparse vector; if S is small, $\mathbf{d}$ is said to be sparsely represented in the basis $\mathbf{B}$.

As possible basis functions, we consider the following members of the Daubechies wavelet family that can capture signal characteristics in both time and frequency domains: db1 (also known as Haar), db2, and db4 wavelet basis.[1] These wavelets are able to capture sharp or abrupt changes in the signal. Figure 3 shows the waveforms for these wavelets. We refer the reader to Walker for a primer on wavelets and their scientific applications [15]. We analyze the basis functions in terms of how concisely they encode the data collected from our system. Table I summarizes the percentage of coefficients needed to maintain the difference between the original and reconstructed signals within 1% for each of the bases. In this respect, the Haar wavelet represents our data most concisely.[2]

[1]Here 'db' is short for Daubechies and the number after it represents the number of vanishing moments for the corresponding wavelet basis.

[2]Though beyond the scope of the reported work, rather than manually having to select a basis, a best-basis selection algorithm that automatically adapts the representation basis to the structure of the underlying signal being sampled can be developed, as in Huang *et al.* [11].

When the signal can be represented sparsely in an appropriate basis, it can be acquired from the system directly in a compressed form using a signal-independent strategy such as randomized sampling. This process relies on a key concept called *coherence*: given two N-dimensional bases Ψ and Φ, the coherence between these bases is defined as the largest coherence between any two basis vectors in Ψ and Φ as

$$\mu(\Psi, \Phi) = \sqrt{N} \max_{1 \le k,j \le N} \left| \langle \phi_k, \psi_j \rangle \right|,$$

where $\langle \phi_k, \psi_j \rangle$ is the dot product of the vectors ϕ_k and ψ_j. Typically the coherence between the two bases lies between 1 and $\sqrt{N}$, and when the value of coherence is small we consider the two bases to be uncorrelated or incoherent. When the sensing and representation bases are uncorrelated, a spike in one basis will be represented as a spread-out waveform in the other. This property allows us to capture the complete information present in the original data using a small number of samples obtained by incoherent sampling.

As a sampling strategy to collect measurements from our testbed, we choose Gaussian random matrices that have a low coherence of $\sqrt{2 \log N}$ relative to any representation matrix with high probability. Prior to sample collection, we generate an $M \times N$ Gaussian random matrix G as the underlying sampling matrix. Elements in the matrix are independently chosen from a standard Gaussian distribution of zero mean and variance $1/M$. To obtain the samples from the input data, we simply multiply this matrix G by the vector of data $\mathbf{d}$. For example, assume the data to be sampled is an $N \times 1$ vector

$$\mathbf{d} = \begin{pmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,N} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N,1} & B_{N,2} & \cdots & B_{N,N} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = B\mathbf{x},$$

where B is an $N \times N$ matrix corresponding to the Haar wavelet basis and $\mathbf{x}$ is the representation of $\mathbf{d}$ in that basis. Suppose we wish to obtain a $M \times 1$ vector of samples $\mathbf{y}$. The data is multiplied with a $M \times N$ Gaussian matrix G such that $\mathbf{y} = G\mathbf{d} = GB\mathbf{x} = A\mathbf{x}$, where $A = GB$ is a $M \times N$ matrix.

C. Recovering the Original Signal

The process of incoherent sampling gives us a set of measurements $\mathbf{y} = G\mathbf{d} = GB\mathbf{x} = A\mathbf{x}$, where $A = GB$. To reconstruct the original full-length data $\mathbf{d}$ at the monitoring station, we must solve this inverse problem: given a vector $\mathbf{y}$ of length M and matrix A of size $M \times N$ where $M \ll N$, find a sparse vector $\tilde{\mathbf{x}}$ of length N such that $\mathbf{y} = A\tilde{\mathbf{x}}$. In other words, we are looking for $\tilde{\mathbf{x}}$ as a solution to

$$\min_{\mathbf{b} \in \mathcal{R}^N} \| \mathbf{b} \|_0 \quad \text{subject to: } \mathbf{y} = A\mathbf{b}, \tag{1}$$

where $\|\mathbf{b}\|_0$ is the l_0 norm of $\mathbf{b}$, i.e. the number of nonzero entries in $\mathbf{b}$. This problem is under-constrained since the matrix A has more columns than rows; there are infinitely many candidate signals $\mathbf{b}$ for which $A\mathbf{b} = \mathbf{y}$. Minimizing the l_0 norm is a computationally expensive nonlinear optimization problem. However, the problem can be recast as one of minimizing the l_1 norm which is a linear programming problem as

$$\min_{\mathbf{b} \in \mathcal{R}^N} \|\mathbf{b}\|_1 \quad \text{subject to: } \mathbf{y} = A\mathbf{b}, \tag{2}$$

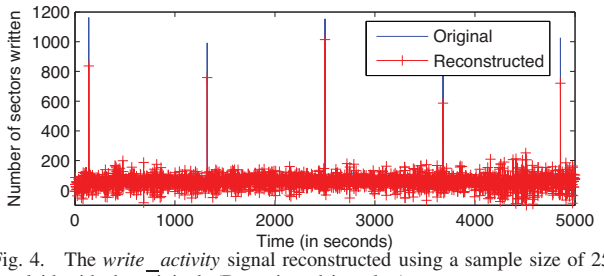
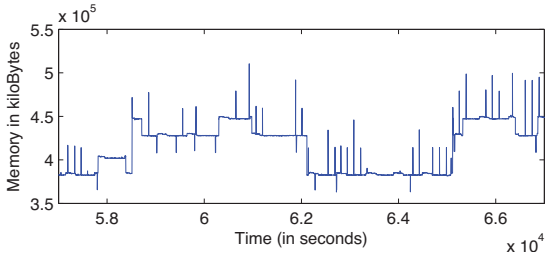
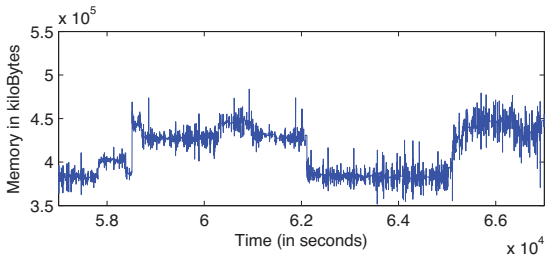


Fig. 4. The *write_activity* signal reconstructed using a sample size of 25%, overlaid with the original. (Best viewed in color.)



(a) The original *CommittedAS* signal.



(b) The reconstructed *CommittedAS* signal.

Fig. 5. The original and reconstructed *CommittedAS* signals. The reconstruction is achieved by compressive sampling using a sample size of 30%.

and solved using a class of reconstruction algorithms called basis pursuit or iterative hard thresholding pursuit (HTP) previously proposed by Foucart [12].

Figures 4 and 5 show the original *write_activity* and *CommittedAS* signals along with their reconstructed versions obtained via compressive sampling. For a given sample size, compressive sampling is superior to a random sampling strategy in terms of capturing spikes and abrupt changes in the full-length signal.

III. Anomaly Detection using Compressed Samples

The optimization problem posed in (2) can be solved efficiently by modern LP solvers. However, a monitoring station may be responsible for recovering and analyzing hundreds of signals belonging to many servers. In the following, we prove from a theoretical viewpoint that key statistical properties of the original data such as the variance, mean, and correlation between data points are preserved in approximate form within the compressed samples. We also show how these properties can be exploited to detect, with high confidence, data spikes and trends by analyzing just the compressed samples without having to reconstruct the full-length signal.

We first describe the compressive sampling process on each local server in greater detail since this knowledge is crucial to understanding the claims and the proofs presented later

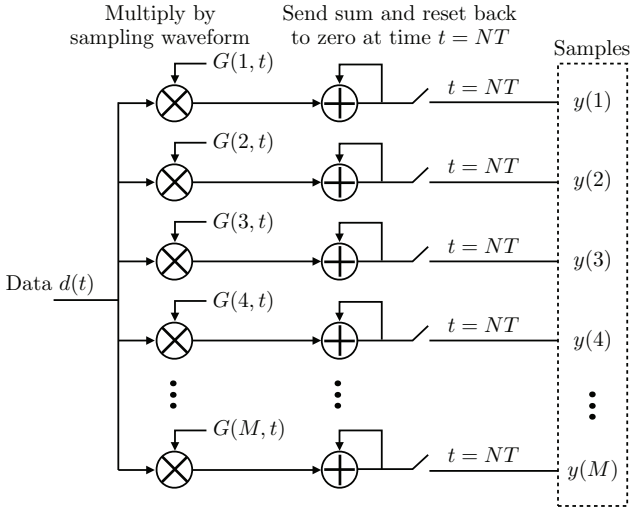


Fig. 6. An implementation of compressive sampling that takes N data items over a time window as input and returns M samples, where $M \ll N$.

in this paper. As shown in the schematic in Fig. 6, this process involves multiplying the incoming signal $\mathbf{d}$ (treated as a vector) with a sampling matrix to form a set of compressed measurements $\mathbf{y}$. When a new data item $d(t)$ arrives at time t, it is multiplied by the entries in the sampling matrix $G(i,t), i = 1,\ldots,M$ and the partial products are accumulated into $y(i)$. After a period of length $N \times T$, where T is the sampling period, the current values of $y(i)$ are sent out as the M samples to the monitoring station, and then reset back to zero. So, effectively, $y(i) = \sum_{t=1}^{N} G(i,t)d(t)$, where $i = 1,\ldots,M$, giving us $\mathbf{y} = G\mathbf{d}$.

The sampling matrix is often designed to have random Gaussian entries [5]. Thus, the generated compressed samples are linear combinations of the original data, and these samples are able to preserve certain properties of the original data.

A. Detection of spikes from compressed samples

We show in the following that a data spike, i.e., an abrupt rise and fall in the value of a data item, within a time window leads to a significant change in the corresponding compressed-sample variance. Let $\mathbf{d}$ denote the vector of data within a window of length N exhibiting no data spikes. Let $\mathbf{d}'$ denote another length-N data series whose entries have the same magnitude as those in $\mathbf{d}$ with the exception of the n-th entry, $d(n)$, i.e., $d'(t) = d(t)$, $t = 1,\cdots,n-1,n+1,\cdots,N$, and $d'(n) = d(n) + \Delta$. If G is the $M \times N$ sampling matrix, then the compressed samples of length M corresponding to $\mathbf{d}$ and $\mathbf{d}'$ are $\mathbf{y} = G\mathbf{d}$ and $\mathbf{y}' = G\mathbf{d}'$, respectively.

Let us denote the sample mean of $\mathbf{y}$ and $\mathbf{y}'$ as $\mu(\mathbf{y})$ and $\mu(\mathbf{y}')$, respectively; denote the (unbiased) sample variance of $\mathbf{y}$ and $\mathbf{y}'$ as $\sigma^2(\mathbf{y})$ and $\sigma^2(\mathbf{y}')$, respectively. Let G_j denote the j^{th} column of G and let $\mu(G_j)$ denote the mean of the j^{th} column of G.

Let $\sigma^2(G_j)$ denote the (unbiased) sample variance of G's j-th column, that is $\sigma^2(G_i) = \frac{1}{M-1}\sum_{i=1}^{M}[G(i,j) - \mu(G_j)]^2$. Let $\sigma_G(j,t)$ denote the sample covariance between the j-th column and the t-th column of G, $\sigma_G(j,t) = \frac{1}{M-1}\sum_{i=1}^{M}[G(i,j) - \mu(G_j)][G(i,t) - \mu(G_t)]$.

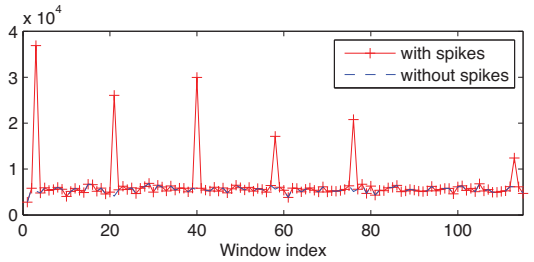


Fig. 7. The sample variance of the compressed samples for *write_activity* data and the spike-free data in each time window.

In the Appendix, we prove that the change in sample variance as a result of the data spike of size Δ in the n-th element of $\mathbf{d}$ leads to a change in the variance given by

$$\sigma^2(\mathbf{y}') - \sigma^2(\mathbf{y}) = \sigma^2(G_n)\Delta^2 + 2\Delta\sum_{t=1}^{N} d(t)\sigma_G(n,t). \qquad (3)$$

In the proof of (3), we also prove that $\mu(\mathbf{y}') = \mu(\mathbf{y}) + \mu(G_n)\Delta$. Note that since entries of G are independently generated random Gaussian variables, the expected column mean of G is 0. As a result, the expected difference between $\mu(\mathbf{y})$ and $\mu(\mathbf{y}')$, i.e., $\mu(G_n)\Delta$, is 0. A large Δ, therefore, cannot be expected to cause a significant change in the sample mean.

The expected column variance of G is $1/M$. The fact that two columns of G are independent of each other leads to zero covariance between any two columns of G. As a result, the expected value of $\sigma^2(G_n)$ is $1/M$ and the expected value of $\sigma_G(n,t)$ is 0. The expected value of the difference between $\sigma^2(\mathbf{y})$ and $\sigma^2(\mathbf{y}')$, i.e., $\sigma^2(G_n)\Delta^2 + 2\Delta\sum_{t=1}^{N} d(t)\sigma_G(n,t)$, is therefore equal to Δ^2/M. Therefore, a large Δ may not lead to a significant change in the sample mean, but it will lead to a significant change in the sample variance.

This inference from (3) is consistent with our observations. For the *write_activity* data, we set the window length to $N = 64$ and the sample size to $M = 16$; so for every 64 data points, 16 compressed samples are generated. We then calculate the sample variance for each window. We also repeat the sampling and variance calculation on the data without any spikes. An overlay of the sample variance for the data with and without spikes is shown in Fig. 7. Windows containing spikes have extremely large variance compared with other time windows.

Our method to detect data spikes or abrupt changes is to collect data in each time window using compressive sampling, and calculate the sample variance of each time window. We then detect a window that has a variance exceeding a threshold as the window that contains spikes or abrupt changes. To select the proper threshold for the sample variance, we learn its distribution by training a dataset that contains no spikes. After obtaining the set of variance for each time window, we obtain the sample mean μ and sample variance σ, and set the threshold to $\mu + \sigma z_{1-\alpha}$, where $z_{1-\alpha}$ is the upper $1-\alpha$ critical value of a standard normal distribution.

During the detection phase, a time window that has a sample variance above this threshold is said to contain spikes or abrupt changes. To find out the exact time stamp within the particular window for the spikes, one can use the reconstruction algorithm, such as HTP, on samples of the detected time window

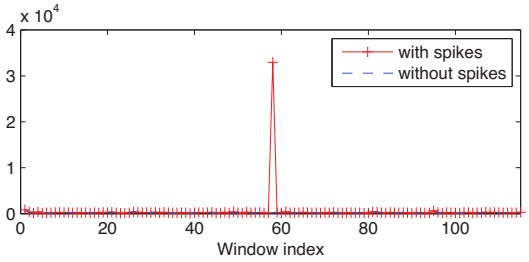


Fig. 8. The variance of the random samples collected for *write_activity* in each time window.

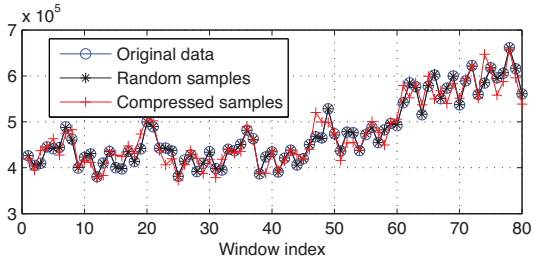


Fig. 9. The average value of the original *CommittedAS* data in each time window overlaid with the scaled average of the compressed samples and random samples.

to recover the original data.

Note that compressive sampling preserves the data spikes better than random sampling. To compare these two sampling techniques, we use random sampling to collect the same amount of samples from the *write_activity* data, and show the sample variance in Fig. 8. The sample variance is extremely large only for the window that contains the fourth spike. The other spikes are not selected as the samples, which is expected since a spike in the original data is selected as a sample with probability M/N.

B. Detection of trends from compressed samples

In the following, we show that the compressed samples preserve the mean of the original data. Let $\mu(G)$ be the average of all entries of G, $\mu(G) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{t=1}^{N} G(i,t)$. As before, denote the mean of $\mathbf{d}$ and $\mathbf{y}$ as $\mu(\mathbf{d})$ and $\mu(\mathbf{y})$, respectively. We prove in the Appendix that

$$\mu(\mathbf{y}) \overset{P}{\approx} N\mu(G)\mu(\mathbf{d}), \tag{4}$$

where $\overset{\mathrm{P}}{\approx}$ stands for being approximately equal with high probability. In other words, the sample mean is approximately the mean of the original data scaled by $N\mu(G)$. Given this property, when the original data increases or decreases in magnitude, the mean of the compressed samples increases or decreases proportionally.

The implications of (4) are consistent with our simulation results. For the *CommittedAS* dataset, we use time bins that each includes 1024 data points in 34 minutes. Sampling is applied on each time bin separately. The sample size for compressive sampling is 3.12%. Figure 9 shows the average of the original data in each time window compared with the scaled mean of the compressed samples (scaled by $1/N\mu(G)$). The two sets of average values are similar and both of them can capture the increasing trend in the *CommittedAS*. We also show the average of random samples in Fig. 9 and find that the values are closer to the average of the original data. However, the process of random sampling loses information of the original data. The original data cannot be fully recovered using the random samples. On the other hand, the samples collected via compressive sampling can be used to reconstruct the original data with much greater fidelity.

To detect the increasing trend in *CommittedAS*, we exploit the fact that the average of the original data within each time window is preserved in scaled form by the compressed samples. To estimate the global trend, we use a 24-hour sliding window, which includes 40 time bins, and move the sliding window by one time bin at each step. We use the linear model $d = a \times t + b$, where d is the average of compressed samples within each time bin, t is the beginning time of the sliding window, a is the slope, and b is the intersection. For each sliding window, 40 (d,t) pairs are applied to this model and the slope within each sliding window is estimated along with the 95% confidence interval. We use the slope estimates to check whether an increasing trend or a decreasing trend exists in the original data.

C. PCA-based detection from compressed samples

Principal component analysis (PCA) is a dimension reduction technique that is frequently used for anomaly detection [16], [17]. It transforms a high-dimensional dataset into new bases called principal components ordered by the strength of the correlations exhibited by the data along their respective directions. As a result, the first principal component captures the strongest correlation pattern of the original data, the second principal component captures the second strongest correlation pattern, and so on [18]. The first few principal components are often chosen as the signature pattern of the data.

For anomaly detection purposes, PCA is typically applied on the original raw data. In this work, on the other hand, we apply PCA directly on the compressed samples to extract normal patterns of behavior present in the original data. These extracted features are then exploited to detect occurrences such as spikes in this data. Our method comprises two phases: training and detection. During training, we use a large dataset to obtain its signature pattern. To remove extreme values from the dataset, we replace the top 0.1% of the data with its median value. For every N data points, M samples are generated via compressive sampling. Assume the sample from the t-th time window is the length-M vector $\mathbf{s}_t$ and that the obtained samples from T windows are $[\mathbf{s}_1, \cdots, \mathbf{s}_T]$. Applying PCA on the sample sets gives us M principal components $\mathbf{p}_1, \cdots, \mathbf{p}_M$, each a length-$M$ vector.

We then study the strength of the correlation captured by each of the principal components and choose the top k among them which capture more than 95% of the correlations. We then define the normal pattern of the data using the first k principal components as $P_k = [\mathbf{p}_1, \cdots, \mathbf{p}_k]$, also referred to as the *normal subspace*. The *anomalous subspace* is the subspace orthogonal to the normal subspace.

In the detection phase, we subtract the statistical mean $\mu(\mathbf{y})$ from the compressed data $\mathbf{y}$, project it onto the normal subspace P_k, and then examine the difference between the

projection and the original test data $\mathbf{y} - \mu(\mathbf{y}) - P_k P_k^T(\mathbf{y} - \mu(\mathbf{y}))$. We then compare the norm of the difference $r_y = \|\mathbf{y} - \mu(\mathbf{y}) - P_k P_k^T(\mathbf{y} - \mu(\mathbf{y}))\|^2$, which we call the *projection residual*, to a certain threshold, and issue an alert indicating an anomaly when it exceeds the threshold. We set the threshold to $\sqrt{2(N-k)(N/M+1)}z_{1-\alpha}+N-k$; the choice of this threshold is explained later in this section.

Rationale behind the method. The PCA-based detection described above is based on the fact that the projection residual r_y as a result of our detection process is sufficiently similar to r_d to allow the use of r_y (in place of r_d) for anomaly detection. Here, r_d is the residual obtained using the full-length data.

Let Σ_d be the covariance matrix of the original data $\mathbf{d}$ with $\Sigma_d = U\Lambda U^T$ being the singular value decomposition of Σ_d where U is a unitary matrix and Λ is a diagonal matrix. The columns of U, $\mathbf{u}_i$, are the eigenvectors/principal components of the original data and the diagonal entries of Λ, λ_i, are the corresponding eigenvalues. Let Σ_y be the covariance matrix of the compressed samples with $\Sigma_y = V\Lambda^* V^T$, where $V = [\mathbf{v}_1, \cdots, \mathbf{v}_N]$, being the principal components of Σ_y. The diagonal entries of Λ^*, λ_i^*, are the eigenvalues.

The sampling matrix for compressive sampling is a random Gaussian matrix, the entries of which follow the Gaussian distribution of zero mean and variance $1/M$. As a result, the obtained compressed samples are linear combinations of the original data. Such linear projections should preserve some spectral properties of the original data.

Our detection method relies on one major finding. We find that when the test data is normal, key statistical properties of the distribution of r_y, the projection residual as a result of applying the subspace method directly on the compressed samples, is related to those of the distribution of r_d, the projection residual of the original data. More specifically, we prove in Section VIII-C in the Appendix that

$$E(r_y) - E(r_d) = O(1). \tag{5}$$

That is, the difference between $E(r_y)$ and $E(r_d)$ is a constant independent of M or N. Further, it is known from [19] that

$$\frac{\text{var}(r_y)}{\text{var}(r_d)} = 1 + \gamma + z(M), \tag{6}$$

where $z(M) = O(1/\sqrt{M})$, $N/M = \gamma$ for large N, $\lambda_1 > 1 + \sqrt{\gamma}$, and Λ_d follows the spiked covariance model, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > \lambda_{k+1} = \lambda_{k+2} = \cdots = \lambda_N = 1$.

The above results show that the mean of the two projection residuals differ by a constant and the variance of one is a scaled version of the other. As a result, it can be stated that the principal components of the compressed samples preserve some of the correlation relationship within the original data.

Under the spiked covariance model, the mean of r_d and r_y are both $\sum_{i=k+1}^{N} \lambda_i = N - k$; the variance of r_d is $\sum_{i=k+1}^{N} 2\lambda_i^2 = 2(N-k)$, and the variance of r_y is approximately $2(1 + N/M)(N - k)$. So, during the detection process, the threshold for the projection residual r_y is set to

$$\sqrt{2(N-k)(N/M+1)}z_{1-\alpha} + N - k.$$

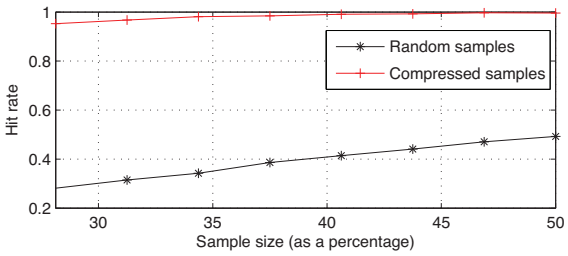


Fig. 10. An overlay of the hit rate achieved as a result of using the compressed samples versus that of random sampling.

IV. Performance evaluation

We use the compressed samples for anomaly detection under the following two scenarios:

- The system operator wishes to detect performance-related bottlenecks or anomalies that manifest themselves as the magnitude of the signal exceeding some nominal threshold value—as spikes or abrupt changes. The performance in this regard is quantified by a hit-rate metric.
- The operator wishes to detect gradual performance deterioration, say over hours or days, associated with software aging or resource exhaustion by analyzing the signals for the existence of trends. Common causes involve resource exhaustion due to memory leaks and bloat, unreleased network sockets and file locks, and unterminated threads. Here the performance is quantified by the confidence with which the samples can be used to estimate a positive slope, if one exists.

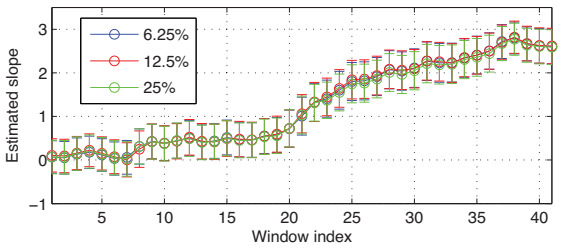
A. Detection of Spikes and Abrupt Changes

This case study uses the *write_activity* signal shown previously in Fig. 2(a) to evaluate how well threshold violations are detected from the compressed samples, specifically abrupt changes and spikes present in the original full-length signal. We use a hit-rate metric to characterize the number of abrupt changes that can be detected by defining a hit as follows: given a time window t within the original data and the compressed samples, a spike occurring in the compressed samples matches a similar spike seen in the original signal.

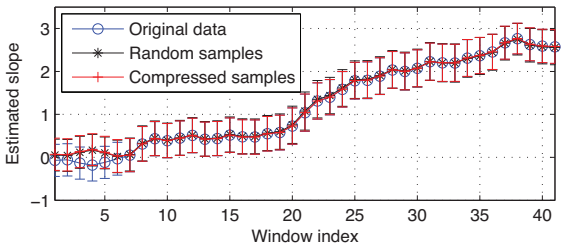
Figure 10 shows the result of applying our method on the *write_activity* data as a function of the sample size. The length of each observation window is set to $N = 64$ data points. Once the sample size exceeds 28%, the hit rate is higher than 95% when the false alarm is fixed at 0.5%. (A sample size smaller than 28% would lead to a false alarm higher than 0.5%.) We find this result is significantly better than that obtained using randomly selected samples from the original signal. The hit rate achieved by random sampling is linear with the sample size, since a higher sampling rate proportionally increases the probability of sampling the spikes present in the original data.

B. Detection of Trends

The second case study shows that analysis of the compressed samples can detect trends within the full-length signal that may indicate a system resource being slowly exhausted—such as memory. We use a long-running Trade6 application

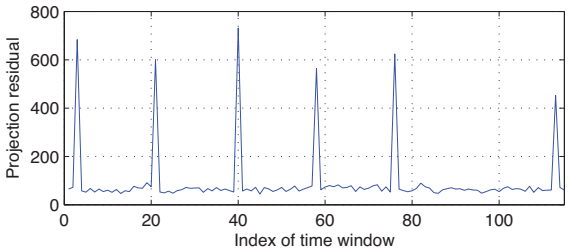


(a) Estimation of slope using compressed samples.

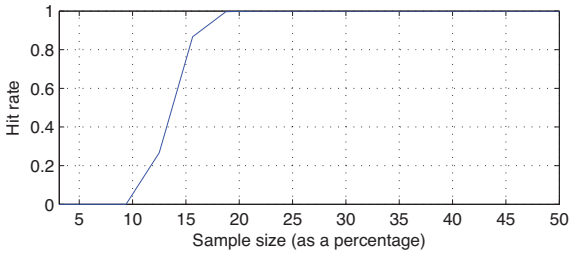


(b) Comparision of the slope estimates.

Fig. 11. Slope estimates obtained for *CommittedAS* using the compressed data for different sample sizes; overlay of the estimated slope values obtained using the original, compressed, and randomly sampled data.



(a) The projection residuals.



(b) Hit rate versus sample size.

Fig. 12. The projection residual of the *write_activity* signal is shown in (a) wherein windows containing a spike have extremely large projections on the anomalous subspace. The achieved hit rate is shown in (b) when the false alarm rate is fixed as 0.5%. The length of the measurement length is set to $N = 64$ and the sample size varies from 3% to 50%.

executing over a period of 48 hours and inject a small memory leak of about 100 KB/minute at around the 24-hour mark; referring back to Fig. 2(b), note the increasing trend in the *CommittedAS* dataset.

Figure 11 summarizes the results. The quality of the slope estimate, in terms of the confidence interval, obtained using the compressed data is relatively insensitive to the sample size as it varies from 6.25% to 25%. Also, the overlay of the estimated slope values obtained using the original, compressed, and randomly sampled data, shown in Fig. 11(b), shows only minor differences in the achieved quality. The sample size for both the compressive and random sampling methods was set to 6.25%. The use of compressive sampling over random sampling is however still advantageous since it allows us to recover the full-length signal with much higher fidelity.

C. PCA-based Detection of Spikes and Abrupt Changes

The final case study deals with using PCA to analyze the *write_activity* dataset for spikes and abrupt changes.[3] Figure 12(a) shows the projection of *write_activity* on to the anomalous subspace—obtained via the previously described training process—for a sample size of 20%. Observation windows containing spikes have significantly higher projection residuals than those without spikes. Figure 12(b) shows the achieved hit rate as a function of sample size when the false alarm rate is fixed. With the false alarm rate fixed at 0.5%, the PCA-based method achieves a hit rate greater than 95% using a sample size of about 18%.

[3]A MATLAB implementation of our PCA-based solution for detecting spikes is available via a GitHub repository located at https://github.com/TingshanHuang/AnomalyDetectionWithCompressedMeasurements. The result in Figure 12(b) is reproducible using the provided dataset and demo script.

V. Related work

Compressive sampling has recently gained traction as a low-cost monitoring solution in varied fields such as wireless sensor networks [20], power grids [21], and microprocessor design [22]. Our previous research applied the concept to the performance monitoring of computing systems [9]–[11]. The emphasis in [9], [10] was on studying the feasibility of using fixed-rate compressive sampling for monitoring server systems. More recently. we developed an adaptive-rate model that exploits any time-varying sparsity in the signal to further reduce the number of collected samples [11]. Tuma *et al.* study the applicability of compressive sampling for fine-grained monitoring of processor performance and evaluate its performance on signals representing micro-architecture counters within a core [22]. They show that compressive sampling can recover these signals if one can identify the bases in which the signals can be sparsely represented.

Lakhina *et al.* develop a PCA-based method for real-time detection of anomalies in computer networks [23]. Here, PCA is applied to high-dimensional network-wide traffic data to extract normal traffic patterns which have a highly reduced dimension. During the detection phase, traffic that does not correspond to the normal pattern is identified as an anomaly. The patterns extracted by this approach is also called a normal subspace; so it is also referred to as the PCA-based subspace method. Recent extensions of this method for network anomaly detection include [24]–[26]. We have also seen PCA-based methods developed for anomaly detection in cloud computing systems [16]. Here, PCA is applied to the run-time performance data collected from each server to extract relevant features, which are then used to train decision tree classifiers for anomaly detection. The aforementioned techniques for

anomaly detection have been shown to be quite effective in detecting anomalies affecting network and computing systems. However, these techniques require the full-length data stream to perform the necessary analysis.

There have been a few recent attempts focused on anomaly detection using compressed data obtained from the underlying system [19], [27]. These papers show that the performance of spectral-based methods (e.g., the PCA-based subspace method) using only knowledge of the compressed samples is similar to that of knowing the original data. Motivated by the fact that the samples should preserve more information than just the spectral properties of the original data, such as the sample mean and variance, our work extends their use to detect abrupt changes and trends in the raw data.

VI. Conclusions

From a viewpoint of reducing monitoring costs, compressive sampling allows for a simple, randomized sensing strategy to acquire the signals of interest on the local server; the network bandwidth needed to transmit these samples to the monitoring station is also significantly reduced. When operators wish to analyze the original signal, there is a way to use numerical optimization to reconstruct the signal from the sample set. We have shown that the compressed samples preserve important statistical properties of the original signal such as the mean, variance, and correlation. This result can be used to reduce the processing cost associated with data analysis on the monitoring station via a two-step process: the compressed samples are first examined for characteristics of interest such as spikes, abrupt changes, and trends; if necessary, the full-length signal is then reconstructed to find out a more accurate time stamp of occurrence within a particular window.

VII. Acknowledgments

This work was partially funded by NSF Award 1228847.

VIII. Appendix

A. Proof of Equation (3)

In Section III, we stated that a data spike of size Δ in a single entry of $\mathbf{d}$ leads to an expected change of Δ^2 in the sample variance of the compressed samples. It follows from the following claim, also stated in (3). Using notation developed in Section III, we prove the claim in this section.

Claim 1: If $\mathbf{y}'$ and $\mathbf{y}$ are the compressed sample vectors with and without a spike of size Δ in one element of the original data vector $\mathbf{d}$, then:

$$\sigma^2(\mathbf{y}') - \sigma^2(\mathbf{y}) = \sigma^2(G_n)\Delta^2 + 2\Delta \sum_{t=1}^{N} d(t)\sigma_G(n,t)$$

Proof: Recall that $\mathbf{d}'$, the data vector with the spike of size Δ, is the same as $\mathbf{d}$ except for its n-th element for which $d'(n) = d(n) + \Delta$. For $i \neq n$, $d'(t) = d(t)$.

First, we show that the sample mean of $\mathbf{d}'$ differs from that of $\mathbf{d}$ by $\mu(G_n)\Delta$.

$$\begin{aligned}
\mu(\mathbf{y}) &= \frac{1}{M}\sum_{i=1}^{M} y(i) = \frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{N} G(i,t)d(t) \\
&= \sum_{t=1}^{N}[\frac{1}{M}\sum_{i=1}^{M} G(i,t)]d(t) = \sum_{t=1}^{N}\mu(G_t)d(t) \\
\mu(\mathbf{y}') &= \sum_{t=1}^{N}\mu(G_t)d'(t) = \sum_{t=1}^{N}\mu(G_t)d(t) - \mu(G_n)d(n) + \mu(G_n)d'(n) \\
&= \sum_{t=1}^{N}\mu(G_t)d(t) + \mu(G_n)\Delta = \mu(\mathbf{y}) + \mu(G_n)\Delta
\end{aligned}$$

In the following we show that due to the change by Δ the sample mean of $\mathbf{d}'$ differs by $\sigma^2(G_n)\Delta^2 + 2\Delta\sum_{t=1}^{N} d(t)\sigma_G(n,t)$, where $\sigma^2(G_n)$ is the sample variance of G's n-th column and $\sigma_G(n,t)$ is the sample covariance between the n-th column and t-th column of G.

Note that, since $y'(i) = y(i) + G(i,n)\Delta$, $y'(i) - \mu(\mathbf{y}')$ is equivalent to $y(i) - \mu(\mathbf{y}) + [G(i,n) - \mu(G_n)]\Delta$. This enables us to break $\sigma^2(\mathbf{y}') = \frac{1}{M-1}\sum_{i=1}^{M}[y'(i) - \mu(\mathbf{y}')]^2$ into three parts:

$$\begin{aligned}
&\frac{1}{M-1}\sum_{i=1}^{M}\{y(i) - \mu(\mathbf{y})\}^2 \\
&+\frac{1}{M-1}\sum_{i=1}^{M}[G(i,n) - \mu(G_n)]^2\Delta^2 \\
&+\frac{1}{M-1}\sum_{i=1}^{M}\left\{2\sum_{t=1}^{N}[G(i,t) - \mu(G_t)]d(t)[G(i,n) - \mu(G_n)]\Delta\right\}
\end{aligned}$$

The first part is the (unbiased) sample variance of $\mathbf{y}$, $\sigma^2(\mathbf{y})$. By the definition of $\sigma^2(G_n)$, $\sigma^2(G_n) = \frac{1}{M-1}\sum_{i=1}^{M}[G(i,n) - \mu(G_n)]^2$. Therefore, the second part is equivalent to $\sigma^2(G_n)\Delta^2$. By the definition of $\sigma_G(n,t)$, $\sigma_G(n,t) = \frac{1}{M-1}\sum_{i=1}^{M}[G(i,n) - \mu(G_n)][G(i,t) - \mu(G_t)]$; therefore, the third part equals $2\Delta\sum_{t=1}^{N} d(t)\sigma_G(n,t)$.

As a result, the following holds true:

$$\sigma^2(\mathbf{y}') = \sigma^2(\mathbf{y}) + \sigma^2(G_n)\Delta^2 + 2\Delta\sum_{t=1}^{N} d(t)\sigma_G(n,t)$$

■

B. Proof of Equation (4)

In Section III, we stated that it is possible to detect a trend by examining only the changes in the mean of the compressed samples instead of the original data. This follows from the following claim:

Claim 2: The expected value of the sample mean, $E[\mu(\mathbf{y})]$, and the mean of the original data, $\mu(\mathbf{d})$, are related by:

$$\mu(\mathbf{y}) \overset{\mathrm{P}}{\approx} N\mu(G)\mu(\mathbf{d})$$

where $\overset{\mathrm{P}}{\approx}$ stands for approximately equal with high probability, N is the length of $\mathbf{d}$, G is the sampling matrix for compressive sampling, and $\mu(G)$ is the average of entries in G.

Proof: From the earlier proof in Section VIII-A, we know that the compressed sample mean $\mu(\mathbf{y})$ is equal to

$\sum_{t=1}^{N} \mu(G_t)d(t)$. By the definition of $\mu(G)$, we have $\mu(G) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{t=1}^{N} G(i,t) = \frac{1}{N}\sum_{t=1}^{N}\mu(G_t)$, which leads to the following:

$$\frac{\mu(G_t)}{N\mu(G)} = \frac{\mu(G_t)}{\mu(G_t) + \sum_{j=1,j\neq t}^{N}\mu(G_j)} = \frac{1}{1 + \frac{\sum_{j=1,j\neq t}^{N}\mu(G_j)}{\mu(G_t)}} \quad (7)$$

Note that entries of G, $G(i,t)$, are independent and follow Gaussian distribution with zero mean and variance $1/M$. As a result, the t-th column mean of G, $\mu(G_t) = \frac{1}{M}\sum_{i=1}^{M} G(i,t)$, also follows a Gaussian distribution with zero mean, but with a variance of $1/M^3$ (denoted by $\mathcal{N}(0, 1/M^3)$). Therefore, $\sqrt{M^3}\mu(G_t)$ can be said to follow $\mathcal{N}(0,1)$.

Similarly, $\sum_{j=1,j\neq t}^{N}\mu(G_j)$ follows $\mathcal{N}(0,(N-1)/M^3)$. As a result, $\sqrt{M^3/(N-1)}\sum_{j=1,j\neq t}^{N}\mu(G_j)$ can be said to follow $\mathcal{N}(0,1)$.

The ratio of the above two independent variables which both follow the standard normal distribution $\mathcal{N}(0,1)$ is given by:

$$\frac{\sqrt{\frac{M^3}{N-1}}\sum_{j=1,j\neq t}^{N}\mu(G_j)}{\sqrt{M^3}\mu(G_t)} = \frac{1}{\sqrt{N-1}}\frac{\sum_{j=1,j\neq t}^{N}\mu(G_j)}{\mu(G_t)}$$

Thus, $\frac{1}{\sqrt{N-1}}\frac{\sum_{j=1,j\neq t}^{N}\mu(G_j)}{\mu(G_t)}$ follows the standard Cauchy distribution, denoted by Cauchy$(0,1)$.

We now apply the fact that if a random variable X follows Cauchy(μ,σ), then the linear transformation $\alpha X + \beta$ follows Cauchy$(\alpha\mu+\beta, |\alpha|\sigma)$ [28]. This implies that $1 + \frac{\sum_{j=1,j\neq t}^{N}\mu(G_j)}{\mu(G_t)}$ follows Cauchy$(1, \sqrt{N-1})$.

Similarly, we again apply another result on transformation of Cauchy distributions: if X follows Cauchy(μ,σ), then $1/X$ follows Cauchy$(\mu/c, \sigma/c)$ where $c = \mu^2 + \sigma^2$ [28]. This implies that the right hand side of (7) follows Cauchy$(1/N, \sqrt{N-1}/N)$.

Given (7) and given the probability density function of the above Cauchy distribution with mean $1/N$ and small variance, it follows that $\mu(G_t)/(N\mu(G))$ is $1/N$ or close to it with high probability. Since $\mu(\mathbf{y})$ is equal to $\sum_{t=1}^{N}\mu(G_t)d(t)$, $\frac{\mu(\mathbf{y})}{N\mu(G)}$ is close to $\sum_{t=1}^{N}\frac{1}{N}d(t) = \mu(\mathbf{d})$ with high probability. That is, $\mu(\mathbf{y})$ is close to $N\mu(G)\mu(\mathbf{d})$ with high probability. ■

C. Proof of Equation (5)

Under normal conditions, r_y, the projection residual of the compressed samples, has an expected mean that differs by a constant from that of r_d, the projection residual of the original data. This, along with the result on the variance of these two distributions ((6)), allows the use of r_y for anomaly detection instead of r_d.

Claim 3: When the original data $\mathbf{d}$ contains no spikes, the expected mean of $r_y = \|(\mathbf{y}-\mu(\mathbf{y})) - V_k V_k^T(\mathbf{y}-\mu(\mathbf{y}))\|^2$ is related to expected mean of $r_d = \|(\mathbf{d}-\mu(\mathbf{d})) - U_k U_k^T(\mathbf{d}-\mu(\mathbf{d}))\|^2$ by

$$E(r_y) - E(r_d) = \bar{O}(1)$$

Proof: First, we show that the mean of r_d is $E(r_d) = \sum_{i=k+1}^{N}\lambda_i$. Let I_N denote an $N \times N$ identity matrix. By the definition of r_d, we have:

$$\begin{aligned} r_d &= \|(\mathbf{d}-\mu(\mathbf{d})) - U_k U_k^T(\mathbf{d}-\mu(\mathbf{d}))\|^2 \\ &= (\mathbf{d}-\mu(\mathbf{d}))^T(I_N - U_k U_k^T)(\mathbf{d}-\mu(\mathbf{d})) \\ &= (\mathbf{d}-\mu(\mathbf{d}))^T(UU^T - U_k U_k^T)(\mathbf{d}-\mu(\mathbf{d})) \\ &= (\mathbf{d}-\mu(\mathbf{d}))^T\sum_{i=k+1}^{N} u_i u_i^T(\mathbf{d}-\mu(\mathbf{d})) \\ &= \sum_{i=k+1}^{N}\|u_i^T(\mathbf{d}-\mu(\mathbf{d}))\|^2 \end{aligned}$$

Since $\|u_i^T(\mathbf{d}-\mu(\mathbf{d}))\|^2$ captures the variance of $\mathbf{d}$ along u_i^T, we have $E(r_d) = \sum_{i=k+1}^{N} E(\|u_i^T(\mathbf{d}-\mu(\mathbf{d}))\|^2) = \sum_{i=k+1}^{N}\lambda_i$. Similarly, the mean of r_y is $E(r_y) = \sum_{i=k+1}^{M}\lambda_i^*$.

Next, we show that r_d is similar to r_y by proving $E(r_y) - E(r_d)$ is a constant independent of M or N, or equivalently, $\sum_{i=k+1}^{M}\lambda_i^* - \sum_{i=k+1}^{N}\lambda_i = \bar{O}(1)$. The proof is completed in two parts: (i) by proving that $E(\sum_{i=1}^{N}\lambda_i) = E(\sum_{i=1}^{M}\lambda_i^*)$, and (ii) by proving that $E(\sum_{i=1}^{k}\lambda_i^*) - E(\sum_{i=1}^{k}\lambda_i) = \bar{O}(1)$.

Part (i): We first prove that $E(\sum_{i=1}^{N}\lambda_i) = E(\sum_{j=1}^{M}\lambda_j^*)$. Note that $\sum_{j=1}^{M}\lambda_j$ is also the trace of the covariance matrix Σ_d, i.e., $\mathrm{tr}(\Sigma_d) = \sum_{i=1}^{N}\Sigma_d(i,i) = \sum_{i=1}^{N}\lambda_i$. Similarly, $\mathrm{tr}(\Sigma_y) = \sum_{i=1}^{M}\Sigma_d^*(i,i) = \sum_{i=1}^{M}\lambda_i^*$. Recall that $\mathbf{y} = G\mathbf{d}$, we have $\Sigma_y = G\Sigma_d G^T = GU\Lambda U^T G^T$. The i-th diagonal entry of Σ_y, therefore, is given by:

$$\sum_{k=1}^{N}\lambda_k\Big(\sum_{j=1}^{N}G(i,j)u_k(j)\Big)\Big(\sum_{j=1}^{N}u_k(j)G(i,j)\Big)$$

As a result, the trace of Σ_y can be rewritten as

$$\begin{aligned} \mathrm{tr}(\Sigma_y) &= \sum_{i=1}^{M}\sum_{k=1}^{N}\lambda_k\left(\sum_{j=1}^{N}G(i,j)u_k(j)\right)\left(\sum_{j=1}^{N}u_k(j)G(i,j)\right) \\ &= \sum_{k=1}^{N}\lambda_k\sum_{i=1}^{M}\left(\sum_{j=1}^{N}u_k(j)G(i,j)\right)\left(\sum_{j=1}^{N}G(i,j)u_k(j)\right) \\ &= \sum_{k=1}^{N}\lambda_k u_k^T G^T G u_k \end{aligned}$$

The difference between the traces of Σ_y and Σ_d is $\sum_{k=1}^{N}\lambda_k u_k^T(G^TG - I_N)u_k$, where I_N is an $N\times N$ identity matrix. Note that entries of matrix G follow $\mathcal{N}(0,\frac{1}{M})$. Therefore, $E(G^TG) = I_N$, and $E(\mathrm{tr}(\Sigma_y)) = E(\mathrm{tr}(\Sigma_d))$.

Part (ii): We next prove that $\sum_{i=1}^{k}\lambda_i \overset{P}{\approx} \sum_{j=1}^{k}\lambda_j^*$.

Consider a length-N vector $\mathbf{x}$ that follows a Gaussian distribution $\mathcal{N}(0,\Sigma_x)$ where $\Sigma_x = \mathrm{diag}\{\lambda_1^{(x)},\cdots,\lambda_N^{(x)}\}$ is a diagonal matrix that follows the spiked covariance model. It is shown in [29] that if we build $X = [\mathbf{x}_1,\cdots,\mathbf{x}_M]$ with M realizations of $\mathbf{x}$, then the i-th largest eigenvalue of matrix $S = \frac{1}{M}XX^T$, $\lambda_i^{(s)}$, satisfies:

$$\sqrt{M}(\lambda_i^{(s)} - \lambda_i^{(x)}) - \sqrt{M}\frac{\gamma\lambda_i^{(x)}}{\lambda_i^{(x)}-1} \sim \mathcal{N}(0, 2\lambda_i^{(x)2} - \frac{\gamma 2\lambda_i^{(x)2}}{(\lambda_i^{(x)}-1)^2}) \quad (8)$$

as $M,N\to\infty$ and $\frac{N}{M}\to\gamma$. Applying the result of [29] to our case, we build X as $X = \Lambda^{\frac{1}{2}}U^TG^T$. The i-th column of X is $\mathbf{x}_i = \Lambda^{\frac{1}{2}}U^T g_i^T$. Due to randomness of $G(i,j)$, we have the mean of $\mathbf{x}_i$

$E(\mathbf{x}_i) = \Lambda^{\frac{1}{2}} U^T E(g_i^T) = 0$, and $E(\mathbf{x}_i \mathbf{x}_i^T) = \Lambda^{\frac{1}{2}} U^T E(g_i^T g_i) U \Lambda^{\frac{1}{2}} = \Lambda^{\frac{1}{2}} U^T \frac{1}{M} I_N U \Lambda^{\frac{1}{2}} = \frac{1}{M}\Lambda$. Therefore, $\lambda_i^{(x)} = \lambda_i / \sqrt{M}$. On the other hand, $S = \frac{1}{M} X X^T$ has the same eigenvalues with those of $\frac{1}{M} X^T X = \frac{1}{M} G \Sigma_d G^T = \frac{1}{M} \Sigma_y$ which implies that $\lambda_i^{(s)} = \lambda_i^* / \sqrt{M}$, $i = 1, \cdots, M$. Applying the result of [29] in our case allows us to relate λ_i^* to λ_i as:

$$(\lambda_i^* - \lambda_i) - \frac{\gamma \lambda_i}{\lambda_i - 1} \sim \mathcal{N}(0, 2{\lambda_i}^2 - \frac{\gamma 2 {\lambda_i}^2}{(\lambda_i - 1)^2}) \qquad (9)$$

Using this relationship between λ_i^* and λ_i, we get that the difference between $E(\sum_{i=1}^k \lambda_i^*)$ and $E(\sum_{i=1}^k \lambda_i)$ is only a constant since $E(\sum_{i=1}^k \lambda_i^* - \sum_{i=1}^k \lambda_i) = E(\sum_{i=1}^k \gamma \lambda_i / (\lambda_i - 1)) = O(1)$.

Combining the results in Parts (i) and (ii) above, we have:

$$E(r_y) - E(r_d) = (\sum_{i=1}^{M} \lambda_i^* - \sum_{i=1}^{M} \lambda_i) - (\sum_{i=1}^{k} \lambda_i^* - \sum_{i=1}^{k} \lambda_i) = O(1)$$

■

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[2] A. Avritzer *et al.*, "Performance assurance via software rejuvenation: Monitoring, statistics and algorithms," in *Proc. IEEE Conf. Dependable Syst. Netw. (DSN)*, 2006, pp. 435–444.

[3] K. Vaidyanathan and K. S. Trivedi, "A comprehensive model for software rejuvenation," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 2, pp. 124–137, April 2005.

[4] R. Canzanese, M. Kam, and S. Mancoridis, "Toward an automatic, online behavioral malware classification system," in *Proc. IEEE 7th Int'l Conf. Self-Adaptive and Self-Organizing Systems (SASO)*, 2013, pp. 111–120.

[5] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Proc. Mag.*, vol. 25, no. 2, pp. 21–30, 2008.

[6] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[7] E. J. Candès and T. Tao, "Near optimal signal recovery from random projections: Universal coding strategies?" *IEEE Trans. Inform. Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[8] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[9] T. Huang, N. Kandasamy, and H. Sethu, "Evaluating compressive sampling strategies for performance monitoring of data centers," in *Proc. ACM 9th Int'l conf. Autonomic computing*, 2012, pp. 201–210.

[10] ——, "Evaluating compressive sampling strategies for performance monitoring of data centers," in *Proc. IEEE Network Operations & Management Symp. (NOMS)*, 2012, pp. 655–658.

[11] T. Huang, N. Kandasamy, H. Sethu, and M. C. Stamm, "An efficient strategy for online performance monitoring of datacenters via adaptive sampling," ECE Department, Drexel University, Tech. Rep., May 2015, Can be accessed at www.ece.drexel.edu/kandasamy/adaptive_cs.pdf.

[12] S. Foucart, "Hard thresholding pursuit: An algorithm for compressive sensing," *SIAM J. Numer. Anal.*, vol. 49, no. 6, pp. 2543–2563, 2011.

[13] L. Li, K. Vaidyanathan, and K. Trivedi, "An approach for estimation of software aging in a web server," in *Proc. Symp. Empirical Softw. Eng.*, 2002, pp. 91–100.

[14] D. Mosberger and T. Jin, "httperf: A tool for measuring web server performance," *Perf. Eval. Review*, vol. 26, no. 3, pp. 31–37, 1998.

[15] J. S. Walker, *A Primer on Wavelets and their Scientific Applications*, 2nd ed. Chapman and Hall, 2008.

[16] S. Fu, "Performance metric selection for autonomic anomaly detection on cloud computing systems," in *IEEE Global Communications Conference, Exhibition & Indudstry Forum (GLOBECOM)*, 2011, pp. 1–5.

[17] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in *IEEE Int'l Symp. Reliable Distributed Systems*, Sept 2013, pp. 205–214.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[19] Q. Ding and E. D. Kolaczyk, "A compressed pca subspace method for anomaly detection in high-dimensional data," *IEEE Trans. Information Theory*, vol. 59, no. 11, pp. 7419–7433, 2013.

[20] C. Caione, D. Brunelli, and L. Benini, "Distributed compressive sampling for lifetime optimization in dense wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 30–40, 2012.

[21] R. Ward, "Compressed sensing with cross validation," *IEEE Transactions on Information Theory*, vol. 55, no. 12, pp. 5773–5782, 2009.

[22] T. Tuma, S. Rooney, and P. Hurley, "On the applicability of compressive sampling in fine grained processor performance monitoring," in *IEEE International Conference on Engineering of Complex Computer Systems*, 2009, pp. 210–219.

[23] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 219–230, Aug. 2004.

[24] C. Pascoal, M. Rosario de Oliveira, R. Valadas, P. Filzmoser, P. Salvador, and A. Pacheco, "Robust feature selection and robust PCA for Internet traffic anomaly detection," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1755–1763.

[25] T. Kudo, T. Morita, T. Matsuda, and T. Takine, "PCA-based robust anomaly detection using periodic traffic behavior," *Proc. IEEE Int'l Conf. on Communications*, pp. 1330–1334, June 2013.

[26] C. Callegari, L. Gazzarrini, S. Giordano, M. Pagano, and T. Pepe, "Improving PCA-based anomaly detection by using multiple time scale analysis and kullback–leibler divergence," *International Journal of Communication Systems*, vol. 27, no. 10, pp. 1731–1751, 2014.

[27] D.-S. Pham, S. Venkatesh, M. Lazarescu, and S. Budhaditya, "Anomaly detection in large-scale data stream networks," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 145–189, 2014.

[28] A. Papoulis and S. U. Pillai, "Probability, random variables, and stochastic processes," *Tata McGraw-Hill Education*, 2002.

[29] D. Paul, "Asymptotics of sample eigenstructure for a large dimensional spiked covariance model," *Statistica Sinica*, vol. 17, no. 4, p. 1617, 2007.