

# Evaluating Compressive Sampling Strategies for Performance Monitoring of Data Centers

Tingshan Huang, Nagarajan Kandasamy, Harish Sethu

ECE Department, Drexel University

Philadelphia, PA 19104, USA

Email: th423@drexel.edu, kandasamy@drexel.edu, sethu@drexel.edu

**Abstract**—Performance monitoring of data centers provides vital information for dynamic resource provisioning, fault diagnosis, and capacity planning decisions. However, 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. This paper proposes a low-cost monitoring solution using compressive sampling—a technique that allows certain classes of signals to be recovered from the original measurements using far fewer samples than traditional approaches—and evaluates its ability to measure typical signals generated in a data-center setting using a testbed comprising the Trade6 enterprise application. The results open up the possibility of using low-cost compressive sampling techniques to detect performance bottlenecks and anomalies that manifest themselves as abrupt changes exceeding operator-defined threshold values in the underlying signals.

**Index Terms**—Performance management, online monitoring, compressive sampling;

## I. INTRODUCTION

Online performance monitoring of both the IT infrastructure as well as the physical facility is vital to ensuring the effective and efficient operation of data centers [1]. Examples of monitoring solutions include the Tivoli Monitoring software from IBM for the IT infrastructure [2] and the Data Center Environmental Edge from HP that monitors temperature, humidity, and state of the power network within the data center. The monitored information drives real-time performance management decisions such as dynamic provisioning of IT resources to match the incoming workload, detection and mitigation of performance-related hotspots/bottlenecks, and fault diagnosis. The information also drives decisions of a longer-term nature: intelligent capacity planning that identifies resources that are over-utilized (under-utilized) and aims to improve overall facility utilization by adding (removing) appropriate resources.

We consider a server cluster wherein software-based sensors embedded within the IT infrastructure measure various performance-related parameters associated with the cluster—high-level metrics such as response time and throughput as well as low-level metrics such as processor utilization, I/O activity (disk reads and writes), and network activity (packets sent and received). The information collected by the sensors 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 use (such as analysis aimed at capacity planning) consumes disk space. So, when monitoring a large-scale computing system it is desirable to minimize the above-described costs, the focus of this paper.

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—a recent development in signal processing—states, however, that we can recover a certain class of signals from the original measurements using far fewer samples than techniques that use Shannon's theorem [3]. Compressive sampling contends that many natural signals are sparse—they have concise representations when expressed in the proper basis. This property is used to capture the useful information content embedded in the signal and condense it into a small amount of data. In other words, one can acquire these signals from the underlying system directly in a compressed form. From a viewpoint of reducing the costs associated with online monitoring, compressive sampling allows for a very simple sensing strategy: rather than tailoring the sensing scheme to the specific signal being measured, a signal-independent strategy such as randomized sampling can be used, significantly reducing the intrusion of monitoring on application performance. Also, since signals are acquired directly in compressed form, the network bandwidth required 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.

This paper investigates the feasibility of using compressive sampling to measure signals typically generated in an enterprise-level data center and compares data sampling, encoding, and recovery strategies. We use IBM's Trade6 benchmark, a stock-trading service which allows users to browse, buy, and sell stocks, as our testbed and subject it to a self-similar workload while measuring these signals: response time,

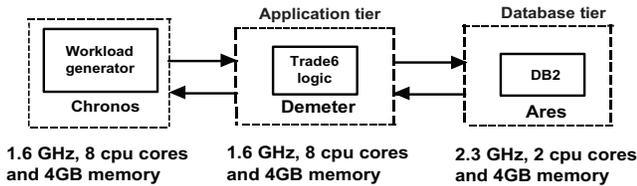


Fig. 1. The overall system architecture hosting the Trade6 service.

CPU utilization, and disk I/O activity in terms of sectors read and written. The measurements are acquired directly in compressed form by encoding the data in an appropriate representation basis. For our experiments, we choose four representation bases—the Fourier basis, and the Haar, Daubechies-2, and Daubechies-4 wavelet bases—to determine the sparsity or conciseness of the data when encoded in each of the basis functions. At the monitoring station, the process of recovering the original signal from these samples is posed as a linear programming problem and solved as such.

When the data-center operator is interested in detecting performance-related bottlenecks or anomalies that manifest themselves as spikes or abrupt changes in the signal exceeding some nominal threshold value, then the signal reconstructed via compressive sampling must preserve the spikes observed in the original trace. This performance is quantified by the receiver operating characteristic (ROC) curve and in this regard compressive sampling performs quite well for all signals considered in this paper. By selecting the threshold value appropriately, a hit rate of 90% can be achieved with a false alarm rate of only 5%. The results reported in the paper open up the possibility of using compressive sampling as a low-cost online monitoring tool in data centers, especially for anomaly detection and long-term capacity planning.

## II. EXPERIMENTAL SETUP

Fig. 1 shows the computing system used in our experiments, comprising three servers networked via a gigabit switch. Virtualization of this system is enabled by VMWare’s ESX Server 3.5 running a Linux RedHat kernel. The system hosts IBM’s Trade6 stock-trading benchmark in which 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 virtual machine on the server Demeter within the application tier. The database component is DB2 which is hosted on the server Ares running SUSE Enterprise Linux. The database maintains 10,000 user accounts and information for 20,000 stocks.

We use Httperf, an open-loop workload generator, to send browse/buy/sell requests to the Trade6 application [4]. The workload follows a self-similar distribution with an arrival rate of 100 requests per second with a 50/50 mix of buy to browse transactions [5]. Every 100 milliseconds, we measure the average end-to-end response time incurred by the requests (*response\_time*). At the database tier, we collect the number of disk sectors read from (*read\_activity*). A sample set of work-

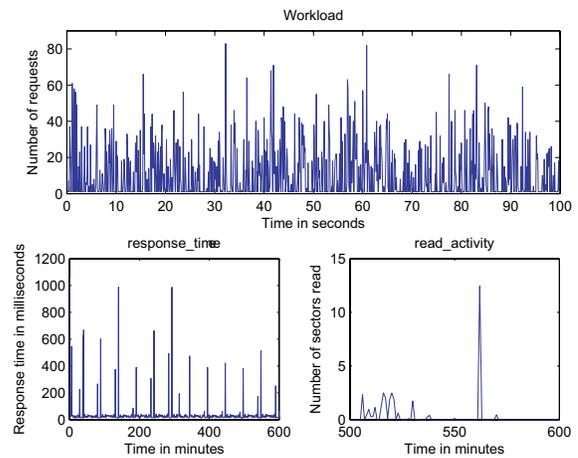


Fig. 2. The self-similar workload presented to the computing system and the various signals collected from the system during an experimental run.

load and parameters/signals collected during an experimental run of the system is shown in Fig. 2.

## III. COMPRESSIVE SAMPLING OF SYSTEM PARAMETERS

This section describes how to acquire the signals of interest from the testbed directly in a compressed form. First, we familiarize the reader with the two key conditions underlying compressive sampling: sparsity and incoherence. The first condition applies to the signals themselves and the second condition affects the way we sample these signals.

Assume that  $\mathbf{d}$ , the data to be sampled, is a vector of length  $N$  and its representation in some basis  $B$  is  $\mathbf{x}$ . In other words,  $d(t) = \sum_{i=1}^N x_i b_i(t) = B\mathbf{x}$ , where  $B = [b_1, b_2, \dots, b_N]$ . For example, if  $B$  is selected to be the Fourier basis, the elements of the vector  $\mathbf{x}$  are Fourier coefficients corresponding to the signal  $\mathbf{d}$ . Also, if at most  $S$  entries in  $\mathbf{x}$  are nonzero, then  $\mathbf{x}$  is called an  $S$ -sparse vector and  $\mathbf{d}$  is said to be sparsely represented in the basis  $B$ .

Using the data collected from our testbed, we now aim to find a basis  $B$  in which this data can be most concisely represented. We consider the following four basis functions: the Fourier basis, the Haar wavelet basis, the Daubechies-2 (or db2) wavelet basis, and the Daubechies-4 (or db4) wavelet basis. These four bases are common for time-frequency analysis on time-series data: Fourier basis is the traditional tool for analyzing the frequency characteristics of a signal while wavelet bases show signal characteristics in both time and frequency domains. Considering the advantages of using wavelets to capture sharp or abrupt changes in the signal, we focus on three commonly used wavelets: the Haar, the simplest wavelet that captures discontinuities in the data; db2, a more complex waveform with more similarity to the data collected from the testbed; and db4, a wavelet that also shows similarity with our data but has a longer waveform, leading to better frequency resolution. A full introduction to the waveforms corresponding to these wavelets may be found in Walker [6].

Given two  $N$ -dimensional bases  $\Psi$  and  $\Phi$ , the coherence between these bases is defined as the largest coherence between

TABLE I

AVERAGE COHERENCE BETWEEN AN  $M \times N$  RANDOM GAUSSIAN SENSING BASIS AND THE DIFFERENT REPRESENTATION BASES WHEN  $N = 1024$ .

Representation basis	Haar	db2	db4	Fourier
$M = 0.1N$	4.57	4.53	4.50	3.39
$M = 0.3N$	4.77	4.75	4.77	3.60
$M = 0.5N$	4.84	4.82	4.82	3.64

any two basis vectors in  $\Psi$  and  $\Phi$ , and is given by

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

where  $\langle \phi_k, \psi_j \rangle$  is the inner product of the basis vectors  $\phi_k \in \Phi$  and  $\psi_j \in \Psi$ . Usually, the coherence between two bases lies in the range of  $[1, \sqrt{N}]$  and when the value of coherence is small we consider the two bases to be uncorrelated or incoherent. When the sensing basis and the representation basis have a small coherence value, and thus uncorrelated, then a signal that is represented as a spike in one basis will be represented as a spread-out waveform in the other. For example, the Dirac delta basis and Fourier basis have a coherence of one; a signal shown as a spike in the Dirac basis is spread out when represented using the Fourier basis. 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. Table I shows the average coherence between an  $M \times N$  Gaussian sensing basis and the various representation bases of interest where  $N$  is the length of the input data and  $M$  is the desired number of samples. As expected, the coherence values are quite low.

To collect  $M$  samples, 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 and the rows are orthonormalized such that the rows have unit norm and are orthogonal to each other. To obtain the samples from the input data, we simply multiply this matrix  $G$  by the vector of data  $\mathbf{d}$ .

Fig. 3 shows the implementation of compressive sampling in our system in which the incoming signal  $\mathbf{d}$  is acquired directly in a compressive form  $\mathbf{y}$ . When a new data item  $d(t)$  arrives at time  $t$ , it is multiplied by the entries in the sampling matrix  $G(t, j)$ ,  $j = 1, \dots, M$ , and the partial products are accumulated into  $y(j)$ . After a period of length  $N \times T$ , where  $T$  is the sampling period, the current values of  $y(j)$  are sent out as the  $M$  samples and then reset back to zero. Effectively,  $y(j) = \sum_{t=1}^M G(j, t)d(t)$ , where  $t = 1, \dots, N$ , and thus  $\mathbf{y} = G \times \mathbf{d}$ .

#### IV. RECOVERING THE ORIGINAL SIGNAL

The process of incoherent sampling gives us a set of values  $\mathbf{y} = G \times \mathbf{d} = G \times B \times \mathbf{x} = A\mathbf{x}$ , where  $A = G \times B$ . To reconstruct the original data  $\mathbf{d}$ , 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

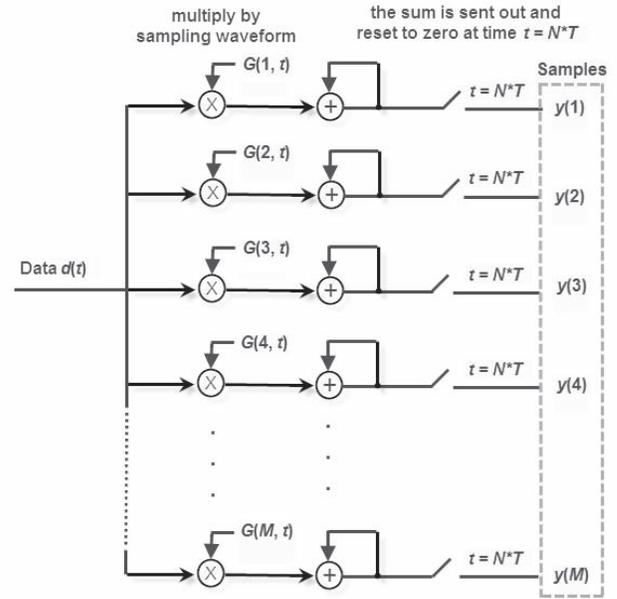


Fig. 3. Implementation of compressive sampling in our system that takes  $N$  data items over a time period as input and returns  $M$  samples.

$$\min_{\tilde{\mathbf{b}} \in \mathcal{R}} \|\tilde{\mathbf{b}}\|_{l_0} \quad \text{subject to: } \mathbf{y} = A\tilde{\mathbf{b}}, \quad (2)$$

where  $\|\tilde{\mathbf{b}}\|_{l_0}$  is the number of nonzero entries in  $\tilde{\mathbf{b}}$ . This problem is under-constrained since matrix  $A$  has more columns than rows; there are infinitely many candidate signals  $\tilde{\mathbf{b}}$  for which  $A\tilde{\mathbf{b}} = \mathbf{y}$ . Furthermore, the problem of minimizing  $l_0$  norm is a non-linear optimization problem which is computationally expensive. However, the problem in (2) can instead be solved by minimizing the  $l_1$  norm, which is the linear programming problem

$$\min_{\tilde{\mathbf{b}} \in \mathcal{R}} \|\tilde{\mathbf{b}}\|_{l_1} \quad \text{subject to: } \mathbf{y} = A\tilde{\mathbf{b}}. \quad (3)$$

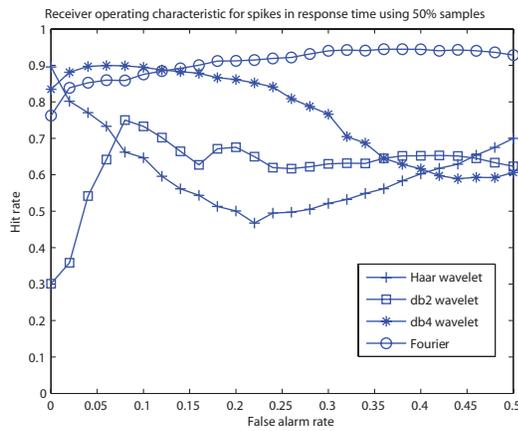
Reconstruction of the original signal using (3) is considered to be exact with probability exceeding  $1 - \delta$ , where  $\delta$  is a very small constant, if the number of samples

$$M \geq C\mu(\Psi, \Phi)^2 S \log \frac{N}{\delta}, \quad (4)$$

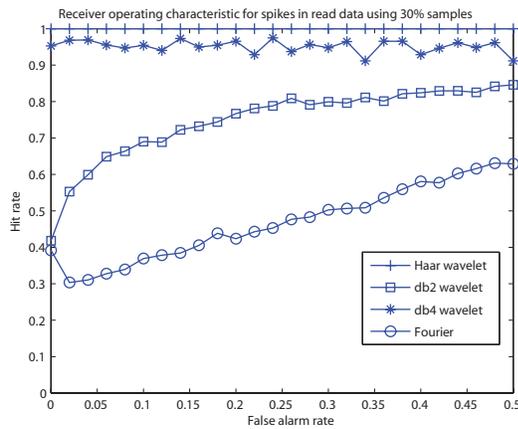
where  $M$  is the number of samples and  $C$  is some positive constant. The direct consequence of (4) is that when the coherence between the representation and sensing bases,  $\mu$ , as well as the sparsity metric,  $S$ , are small, we need only a few samples to recover the original signal exactly with high probability.

#### V. PERFORMANCE EVALUATION

Considering the number of spikes present in the *response\_time* and *read\_activity* data sets, we use the ROC metric to characterize the number of spikes that can be detected using the reconstructed signal under the various basis



(a) ROC curve generated by the reconstructed *response\_time* signal.



(b) ROC curve generated by the reconstructed *read\_activity* signal.

Fig. 4. The ROC curves for the *response\_time* and *read\_activity* signals reconstructed under the four representation bases.

functions. We define a hit as follows: at time instant  $t$  within the original and recovered signals, a spike occurring in the recovered signal matches a similar spike in the original signal. We define a false alarm as follows: at time instant  $t$ , a spike occurring in the recovered signal has no match in the original signal. The ROC plots the hit rate against the false alarm rate. The ROC becomes relevant in situations in which it may not be essential for compressive sampling to recover the original signal exactly, such as when the operator is mainly interested in detecting performance bottlenecks, hot spots, or anomalies affecting the computing system, some of which manifest themselves as spikes or abrupt changes in the signals being monitored.

Figs. 4(a) and 4(b) graph the ROC curves for the *response\_time* and *read\_activity* signals reconstructed under the different representation bases. The plots are generated using 50% and 30% of the samples from the original *response\_time* and *read\_activity* signals, respectively. We change the threshold level that defines a spike, and for each threshold obtain and plot a pair of values: the hit rate and the corresponding false alarm rate. Generally, the false alarm rate

TABLE II  
FALSE ALARM AND HIT RATES ASSOCIATED WITH DIFFERENT THRESHOLD LEVELS FOR THE RECONSTRUCTED *response\_time* SIGNAL.

Threshold level (ms)	146.8	130.0	91.2	68.3
False alarm rate	0.5%	1%	5%	10%
Hit rate	84.2%	86.3%	90.2%	91.4%

TABLE III  
FALSE ALARM AND HIT RATES ASSOCIATED WITH DIFFERENT THRESHOLD LEVELS FOR THE RECONSTRUCTED *read\_activity* SIGNAL.

Threshold level	0.22	0.19	0.14	0.12
False alarm rate	0.5%	1%	5%	10%
Hit rate	85.1%	90.9%	93.2%	95.5%

decreases as the threshold is increased since fewer false spikes occur in the reconstructed signal.

Table II lists the false alarm and hit rates associated with different threshold levels for the reconstructed *response\_time* signal. The original signal was encoded using the db4 wavelet basis. The performance, in terms of ROC, indicates that compressive sampling reconstructs the spikes of interest with high fidelity. A hit rate of 90% can be achieved with a false alarm rate of 5% by setting the threshold to about 100 milliseconds. Decreasing the threshold increases the hit rate at the expense of increasing the false alarm rate as well. Finally, Table III details the effect on the ROC achieved by the reconstructed *read\_activity* signal when varying the threshold level.

## VI. CONCLUSIONS

We have evaluated a low-cost monitoring solution for data centers based on the concept of compressive sampling. Using the Trade6 application, we showed how to acquire measurements corresponding to response time and disk I/O activity directly in a compressed form, and how to reconstruct the full-length signal using a small number of samples. Experiments indicate that the recovered signal adequately preserves the spikes and other abrupt changes present in the original signal. By selecting the threshold value appropriately, a hit rate of 90% can be achieved with a false alarm rate of only 5%. These results open up the possibility of using low-cost compressive sampling techniques to detect performance bottlenecks and anomalies in data centers that manifest themselves as abrupt changes exceeding operator-defined threshold values in the underlying signals.

## REFERENCES

- [1] L. Cherkasova, K. Ozonat, N. Mi, J. Symons, and E. Smiri, "Automated anomaly detection and performance modeling of enterprise applications," *ACM Trans. Comput. Syst.*, vol. 27, pp. 6:1–6:32, Nov. 2009.
- [2] G. Lanfranchi, P. D. Peruta, A. Perrone, and D. Calvanese, "Toward a new landscape of systems management in an autonomic computing environment," *IBM Systems Journal*, vol. 42, no. 1, pp. 119–128, 2003.
- [3] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Proc. Mag.*, vol. 25, no. 2, pp. 21–30, 2008.
- [4] D. Mosberger and T. Jin, "httpperf: A tool for measuring web server performance," *Perf. Eval. Review*, vol. 26, pp. 31–37, 1998.
- [5] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," *IEEE Trans. Networking*, vol. 5, no. 6, pp. 835–846, 1997.
- [6] J. S. Walker, *A Primer on Wavelets and their Scientific Applications*, 2nd ed. Chapman and Hall, 2008.