A PCA Analysis for Traffic Anomaly Estimation

Chunghan Lee
Toyohashi University of Tech.
lch@ss.cs.tut.ac.jp

Hirotake Abe
Osaka University
habe@cmc.osaka-u.ac.jp

Toshio Hirotsu
Hosei University
hirotsu@hosei.ac.jp

Kyoji Umemura
Toyohashi University of Tech.
umemura@tut.jp

ABSTRACT
Network testbeds based on virtualization technology have been used in network research. Network measurement is affected by shared and virtualized resources on a node. Oversize packet spacings, which are caused by CPU scheduling latency, can be a major cause of throughput instability and imprecise network measurement on a virtualized network testbed. The packet spacing which is larger than the TCP transmission period involves packet transmissions, which results in severe throughput. We call this situation an 'anomaly'. Although CPU availability is an important criterion to estimate the anomalies, idle CPUs are consumed by measuring the CPU availability. We observe resource state during throughput measurement, and apply principal component analysis (PCA) to a seven dimensional matrix of the resource state to obtain criteria, instead of the CPU availability. We show that the top two principal components account for 95% of the original data set, and are sufficiently enough to describe the original resource state. Component loadings and a scatter plot of the first and second component scores provide us with a simple view of the resource state for the anomaly estimation. The first component can be presented as workloads, and the second one as the difference in the resource state. The first and second principal component enable the anomalies to be estimated.

KEY WORDS
Network measurement, Virtualization, PlanetLab, Principal component analysis

1 Introduction

Since virtualization technologies have progressed, network testbeds, such as Emulab [1], StarBED [2], and PlanetLab [3][4], using these technologies are used in many fields of network study. PlanetLab, which is a virtualized network testbed over the Internet, has been used to investigate the validity of measurement tools [5] and prediction methods [6]. As of August 2010, it has grown to 1,000 machines spanning more than 500 sites and 40 countries. In it, a node based on the Linux-VServer shares resources, such as central processing unit (CPU), memory, and I/O interfaces. A platform called a 'sliver' is provided as a virtualized environment to users and multiple slivers can be run simultaneously at each node. A set of these slivers participating in the same activity at different nodes is called a 'slice'. We call a slice that used at least 0.1% of the CPU in the last five minutes as a 'live slice', and contains a process as a 'active slice'.

Because the resources are shared by many users and managed by resource brokers, this impact can be affected to the other users on the node. Moreover, it has affected network measurement [7][8][9] and throughput instability [10]. In particular, oversize packet spacings, which are caused by CPU scheduling latency, can be a major cause of throughput instability and imprecise network measurement on the virtualized network testbeds. The packet spacing which is larger than the TCP transmission period involves packet transmissions, which results in severe throughput. We call this situation an 'anomaly'. Although CPU availability is an important criterion to estimate the anomalies, idle CPUs are consumed by measuring the CPU availability.

The aim of our study is to obtain criteria, instead of the CPU availability, for the anomaly estimation. The goal is required to establish a throughput prediction method for network environment based on virtualization technology. To achieve the goal, we are using PlanetLab as a virtualized network testbed. We measure network throughput with resource state to observe the anomalies on the virtualized testbed. Next, we investigate the anomalies using packet-level analysis, and aggregate the resource state at anomalies and no anomalies into a dataset. Finally, we apply principal component analysis (PCA) to a matrix of the resource state to obtain criteria.

In this paper, we present the analysis of resource state using PCA for the anomaly estimation. The main contributions of this work are:

The top two principal components account for 95% of the original data set, and are sufficiently enough to describe the original resource state.

Component loadings and a scatter plot of the first and second component scores provide us with a simple view of the resource state for the anomaly estimation. The first component can be presented as workloads, and the second one as the difference in the resource state at anomalies and no anomalies.

Instead of the CPU availability, the first and second component enable the anomalies to be estimated. In the second component, live and active slice count are

larger than total CPU usage, and an anomaly estimator can be designed without total CPU usage.

The rest of this paper is organized as follows. Firstly, we describe related work in Section 2. Secondly, analysis methodology is explained in Section 3. Next, we discuss analysis results using PCA in Section 4. Finally, we conclude the paper with a summary of the main points in Section 5.

2 Related work

2.1 Fluctuation in packet spacing

Peterson et al. [7] deployed a packet forwarding overlay between Seattle and Washington, D.C on the virtualized testbed and used ping packets to compare round trip time (RTT) between the Seattle and D.C. nodes for the network and overlay. The RTT of network was constant while that of the overlay varied widely. The cause of the fluctuation in RTT was CPU scheduling latency. Although the scheduling latency at a node will be a serious problem for network applications, no consideration has been given to the relationship between packet spacing fluctuation and scheduling latency.

Spring et al. [8] showed that the load prevents accurate latency measurement and precise spacing for packet trains. They ran traceroute and tcpdump in parallel to acquire timestamps between the application and kernel levels and showed the differences between application and kernel-captured timestamps when sending probes and receiving responses. Moreover, they transmitted packet trains to determine how the CPU load impaired precisely-spaced packets. However, they showed no clear conditions for the type of load.

Lee et al. [10] investigated that the anomalies on the virtualized network testbed, and found CPU consumption on per-slice basis is an important parameter for the anomaly estimation. However, this approach consumed idle CPUs to measure the CPU availability. If there are users to use CPUs on the node, the monitoring program will be an obstacle to allocate CPUs to the users.

Wang et al. [9] found that the networking performance between Amazon EC2 instances demonstrated very different characteristics, such as abnormal large delay variations and unstable TCP/UDP throughput, caused by end host virtualization. Thus, the anomalies occurred on the network environment based on virtualization technology.

2.2 Monitoring systems for the virtualized testbed

CoMon [11][12] is a centralized resource monitoring system for the virtualized testbed. It provides views of the virtualized testbed, such as node-centric and slice-centric information. Moreover, it has been used for selecting nodes and identifying problems on the virtualized testbed. Because it gathers data every five minutes, the data granularity is limited and the data type makes it hard to estimate fluctuation in packet spacing.

Clue [13] is an anomaly detection system for the virtualized testbed. However, this system focused on detecting anomalous behavior for the virtualized testbed and it used data on CoMon only.

Slicestat [14] provides slice-level resource consumption information at each node on the virtualized testbed. It does not provide node-level information, such as SSH failing and shutdown. In these monitoring systems, however, the authors did not discuss any unstable conditions occurring in their network experiments.

3 Analysis methodology

3.1 Throughput measurement with resource state

We empirically select four node pairs (eight nodes), which we refer to as nodes α and β, γ and δ, κ and λ, and μ and ν. These nodes are located at different sites across North America and Europe, and composed of two or four independent CPU cores and physical memory of approximately 3 GB (gigabytes). We measure RTT using Internet Control Message Protocol (ICMP); their basic characteristics are given in TABLE 1.

A prediction method has been previously proposed using a pair of different-sized connections. This method, which we call 'connection pair', uses a small sized probe transfer to predict the throughput of a large sized data transfer. Wolski et al. [15][16] used different-sized pairs of connections and empirically established the basic probe size as 64 KB (kilobytes) for Network Weather Service (NWS). However, no consideration was given to the network environment based on virtualization technology. We generate the various sizes of the connection pair to observe the anomalies and to establish a throughput prediction method in the virtualized network. Moreover, we observe resource state per slice on the node using slicestat during throughput measurement. Measurement methodology is described in Figure 1. The size of the connection pair is shown in Table 2.

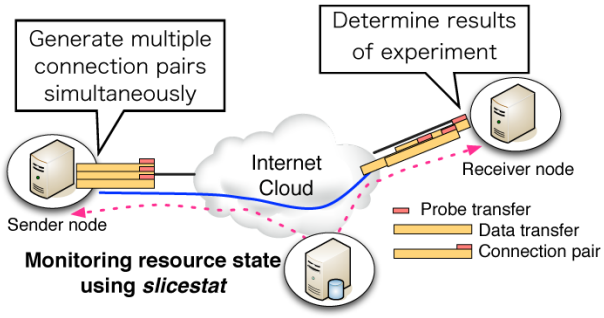


Figure 1. Connection pair measurement method with resource monitoring.

Table 1. Node characteristics on virtualized testbed.

Node name	CPU speed [Ghz]	CPU cores	Memory [GB]	Node pair (receiver, sender)	Mean RTT [ms]
α	Xeon 2.66	2	2.96	node α, node β	42.8
β	Xeon 2.4	4	3.46		
γ	Core2Quad 2.66	4	3.42	node γ, node δ	83
δ	Core2Quad 2.66	4	3.21		
κ	Core2Quad 2.66	4	3.42	node κ, node λ	20
λ	PentiumD 3.2	2	3.47		
μ	CoreDuo 2.33	2	3.45	node μ, node ν	90
ν	Xeon 3.4	2	2.97		

Table 2. Probe and data size combinations for connection pairs.

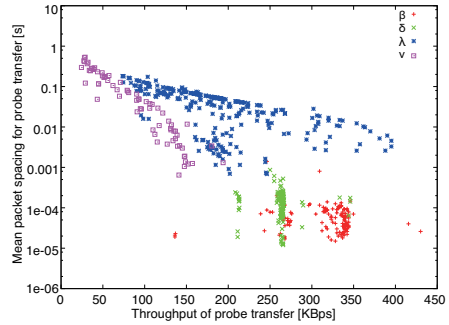
Index	Probe size [KB]	Data size [MB]	Number of connection pairs
1	16	8	3
2	16	16	3
3	32	16	3
4	64	16	3
5	64	32	3

3.2 Data sources

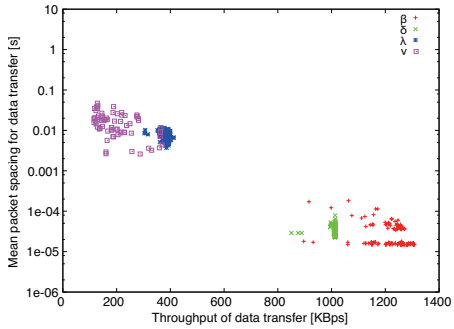
Although the measurement granularity of resource state was 1 minute, the slices on the node, where the resource state was busy, were frequently scheduled off the CPUs, and the granularity was unstable. We gathered throughput measurement results through the connection pairs and approximately 8,000 resource state results at all the pairs for 48 hours. In packet-level analysis, there were no significant changes in the RTT and packet loss rate of connection pair. We depict the mean values of these network metrics in Table 3. We found the anomalies at the sender nodes λ and ν. These were a major cause of throughput instability. Mean packet spacing and throughput of connection pair at all the sender nodes are shown in Figure 2. Moreover, the anomalies occurred both probe and data transfer. Mean packet spacing for connection pair at all the sender nodes is shown in Figure 3. A dataset consists of the resource state at all the sender nodes, thus we aggregate the resource state at anomalies and no anomalies into the dataset.

Table 3. Mean rtt and packet loss rate of connection pair.

Node pair	RTT [ms]	Loss rate [%]
α - β	48.5	0.069
γ - δ	60.2	0.478
κ - λ	22.1	0.002
μ - ν	97.5	0.007



(a) Probe transfer



(b) Data transfer

Figure 2. Mean packet spacing and throughput of connection pair at all sender nodes.

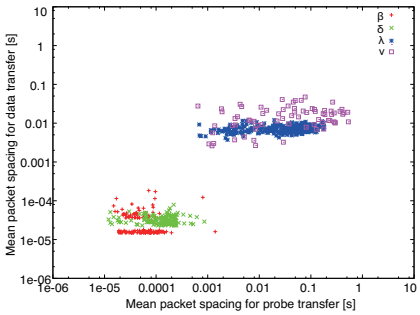


Figure 3. Mean packet spacing for connection pair at all sender nodes.

Table 4. Mean usage of the resource state with anomalies at all sender nodes

Sender node	Anomalies	CpuR [%]	MemR [%]	Proc	CurProc	LiveC	ActiveC	TotalC
β	x	62.61	55.39	687	8.833	6.408	149.8	156.2
δ	x	21.65	29.52	335.4	1.617	1.606	37.5	39.11
λ	o	77.77	63.54	638.1	11.48	10.1	107.6	117.7
ν	o	73.58	80.1	662	15.8	14.54	102.3	116.9

3.3 Features of resource state

We select seven resource state features (total CPU usage, total Memory usage, the total number of processes, the number of current processes, the number of live slices, the number of active slices, the total number of slices). The total CPU usage and the memory usage are consumption rates (%) used at the all slices on the node. A process using the CPU cycle at the moment is called a 'current process'. We call a slice that used at least 0.1% of the CPU in the last five minutes as a 'live slice', and contains a process as a 'active slice'. Because multiple slivers are running simultaneously, and share the resource on the node, the number of the slices is one of important features. The features of the resource state are :

CpuR : Total CPU usage

MemR : Total Memory usage

Proc : The total number of processes

CurProc : The number of current processes

LiveC : The number of live slices

ActiveC : The number of active slices

TotalC : The total number of slices

Mean usage of the resource state with the anomalies at all the sender nodes is shown in Table 4. While there were significant differences in the usage at node δ, there were no significant differences in the usage at node β for the anomaly estimation. We describe minimum, mean, and maximum CpuR at all sender nodes in Figure 4. Although the maximum CpuR at node β was increased to approximately 100 %, the anomalies did not occur. Thus, CpuR would be inappropriate for the anomaly estimation. Moreover, it would be hard to design an anomaly estimator with these features directly. The resource state is quantified by the features for a PCA analysis.

3.4 Principal component analysis

PCA is a dimensionality-reduction technique that is widely used for applications such as dimensionality reduction, lossy data compression, feature extraction, and data visualization. Moreover, PCA has been used in internet traffic

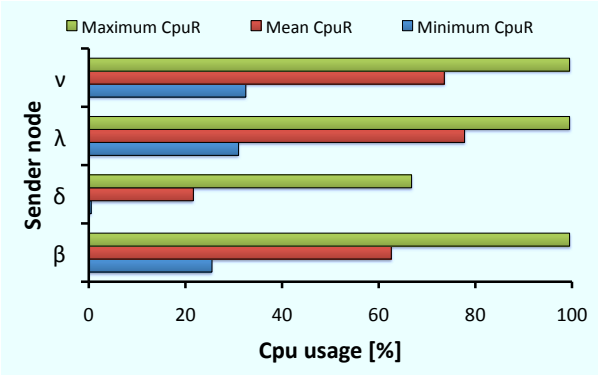


Figure 4. Minimum, mean, and maximum CpuR at all sender nodes.

analysis [17][18]. It seeks a space of lower dimensionality, known as the principal subspace, such that the orthogonal projection of the data points onto this subspace maximizes the variance of the projected points. An alternative definition of PCA is based on minimizing the sum-of-squares of the projection errors. If we consider the general case of an M-dimensional projection space, the optimal linear projection for which the variance of the projected data is maximized is now defined by the M eigenvectors u_1, u_2,...,u_i,...,u_M of the data covariance matrix S corresponding to the largest M eigenvalues λ_1, λ_2,...,λ_i,...,λ_M. The eigenvector u_1 is known as the first principal component, and it has the maximum entropy of data. The eigenvalues are arranged from large to small, so that $\lambda_1 \geq \lambda_2 \geq$, ..., $\lambda_i \geq$, ..., $\geq \lambda_M$. Moreover, they are expressed as a percent of the total variance. The cumulative percentage of total variance s_i accounts for by the current and all preceding proportions. The sum of all the eigenvalues is equal to the number of variables. There is no obvious meaningful components from the trivial components. Most researchers would agree that the first and second components are probably meaningful, but it is difficult to decide exactly. Previous works [19][20] recommended $\lambda_i > 0.7$ or $s_i > 0.9$ as the meaningful components.

4 Analysis results

We applied PCA to a seven dimensional matrix of the resource state, and the eigenvalues and the cumulative percentage of total variance are described in Table 5. The top

Table 5. Eigenvalues and cumulative percentage of total variance.

PC	1	2	3	4	5	6	7
λ_i	5.983	0.660	0.240	0.092	0.016	0.009	0.000
s_i	0.855	0.949	0.983	0.996	0.999	1.000	1.000

Table 6. Component loadings.

PC	1	2	3	4	5	6	7
CpuR	-0.355	-0.294	-0.887	-0.025	0.006	0.015	0.000
MemR	-0.387	0.227	0.109	-0.837	0.093	0.279	0.000
Proc	-0.402	-0.148	0.200	-0.102	0.133	-0.865	0.000
CurProc	-0.371	0.483	-0.017	0.420	0.658	0.138	0.000
LiveC	-0.365	0.536	-0.044	0.207	-0.718	-0.067	0.122
ActiveC	-0.373	-0.464	0.303	0.180	-0.039	0.296	0.658
TotalC	-0.390	-0.322	0.261	0.193	-0.153	0.251	-0.743

two principal components account for 95% of the cumulative percentage, and are sufficiently enough to describe the original resource state. Component loadings are the correlation coefficients between the variables and principal components. It is defined as $Component\ loading = \sqrt{\lambda_i}u_i$. These values are shown in Table 6.

As mentioned earlier, the first principal component captures the maximum entropy of the data. In the analysis results, the first component has a negative relationship of the all resource features. It depicts the workloads on the node. If a principal component score at the first component is close to the positive, the resources on the node are idle. Conversely, if the score at the first component is close to the negative, they have busy states. For example, when CPU and memory are used by the user, the other resource features are increased. The second component has a positive relationship of MemR, CurProc, and LiveC and a negative relationship of CpuR, Proc, ActiveC, and TotalC. The second component relates to the difference in the resource state. Moreover, LiveC and ActiveC were larger than CpuR. Thus, this result implies that these features are more important than CpuR, and an anomaly estimator can be designed without CpuR. The other components have the very small eigenvalues, and it would be hard to describe the resource state.

We show the scatter plot of the first and second component scores in Figure 5. It provides us with a simple view of the resource state for the anomaly estimation. In the scatter plot, there were three cases (A, B, and C). The scores of the first component at case A were close to the negative, and the scores at the second component were higher than case B. The resource state at case A was the state at the nodes λ and ν. Thus, case A is the resource state of anomalies. Although the scores of the first component at case B were similar to case A, the scores at the second component were lower than case A. These scores indicate the resource state at the node β, and there are no anomalies. The scores of the first component at case C are greater than zero. The resource state was idle, and this case corresponds to the state at the node δ. Instead of the CPU availability, the first and second component enable the anomalies to be estimated.

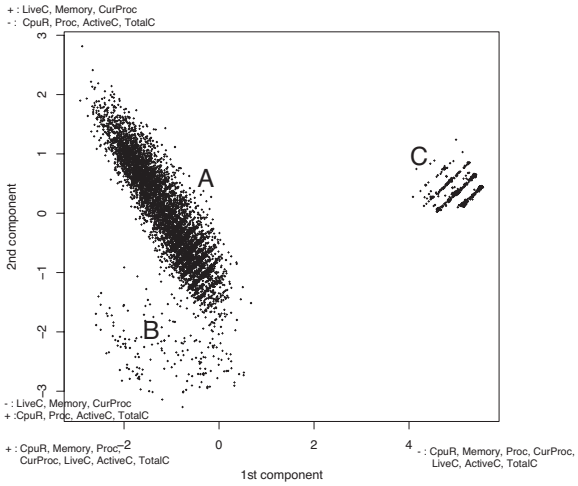


Figure 5. The first and second component scores.

5 Conclusion

We applied PCA to the seven dimensional matrix gathered from the resource state at anomalies and on anomalies to establish the criteria of the anomaly estimation. In the analysis results, we showed that the top two principal components account for 95% of the original data set, and are sufficiently enough to describe the original resource state. Component loadings and a scatter plot of the first and second component scores provide us with a simple view of the resource state for the anomaly estimation. The first and second component, instead of the CPU availability, enable

anomalies to be estimated. In the second component, live and active slice count are larger than total CPU usage, and an anomaly estimator can be designed without total CPU usage.

In future work, we will gather the dataset from multiple sites on the virtualized network testbed, design the anomaly estimator with the criteria. If we observe decreases in throughput on the virtualized testbed, the estimator will calculate the first and second component scores from the resource state, compare them with the scores of the anomalies, and estimate whether the anomaly or not. Finally, we will evaluate the accuracy rate of anomaly estimation using the estimator.

Acknowledgment

This work was supported in part by JSPS KAKENHI (22500072 and 22700029) and Global COE Program "Frontiers of Intelligent Sensing" from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] Emulab, http://www.netbed.org/.

[2] StarBED, http://www.starbed.org/.

[3] PlanetLab https://www.planet-lab.org/.

[4] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating system support for planetary-scale network services," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 19–19.

[5] J. Sommers and P. Barford, "An active measurement system for shared environments," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2007, pp. 303–314.

[6] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante, "Characterizing and predicting tcp throughput on the wide area network," in *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 414–424.

[7] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir, "Experiences building planetlab," in *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 351–366.

[8] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using planetlab for network research: myths, realities, and best practices," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 17–24, 2006.

[9] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *INFOCOM'10: Proceedings of the 29th conference on Information communications*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 1163–1171.

[10] C. Lee, H. Abe, T. Hirotsu, and kyoji Umemura, "Analysis of anomalies on a virtualized network testbed," in *CIT '10: Proceedings of the 10th IEEE International Conference on Computer and Information Technology*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 297–304.

[11] CoMon http://comon.cs.princeton.edu/.

[12] K. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planetlab," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, 2006.

[13] A. Chandra, R. Prinja, S. Jain, and Z. Zhang, "Co-designing the failure analysis and monitoring of large-scale systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, pp. 10–15, 2008.

[14] Slicestat http://codeen.cs.princeton.edu/slicestat/.

[15] M. Swany and R. Wolski, "Multivariate resource performance forecasting in the network weather service," in *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002, pp. 1–10.

[16] R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *Journal of Future Generation Computing Systems*, vol. 15, pp. 757–768, 1999.

[17] K. Fukuda, T. Hirotsu, O. Akashi, and T. Sugawara, "A pca analysis of daily unwanted traffic," *Advanced Information Networking and Applications, International Conference on*, vol. 0, pp. 377–384, 2010.

[18] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," in *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2007, pp. 109–120.

[19] I. T. Jolliffe, "Discarding variables in a principal component analysis. i: Artificial data," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 21, no. 2, pp. 160–173, 1972.

[20] R. B. Cattell, "Factor analysis: An introduction to essentials ii. the role of factor analysis in research," *Biometrics*, vol. 21, pp. 405–435, 1965.