# Segmentation Based Online Performance Problem Diagnosis

Jingwen Zhou*†, Zhenbang Chen†, and Ji Wang†
*Science and Technology on Parallel and Distributed Processing Laboratory,
National University of Defense Technology, Changsha, China
†College of Computer, National University of Defense Technology, Changsha, China
Email: {jwzhou, zbchen, wj}@nudt.edu.cn

***Abstract*—Currently, the performance problems of software systems gets more and more attentions. Among various diagnosis methods based on system traces, principal component analysis (*PCA*) based methods are widely used due to the high accuracy of the diagnosis results and requiring no specific domain knowledge. However, according to our experiments, we have validated several shortcomings existed in PCA-based methods, including requiring traces with a same call sequence, inefficiency when the traces are long, and missing performance problems. To cope with these issues, we introduce a segmentation based online diagnosis method in this poster.**

***Index Terms*—performance problem diagnosis; principal component analysis; system trace; software reliability**

## I. Introduction

Nowadays, the performance of software systems gets more and more attentions from academia and industry, since performance problems may bring enormous loss. For example, on June 29th 2010, a system of Amazon experienced intermittent performance problems for three hours, which made Amazon lose about 1.75 million dollars per hour [1]. Due to the complexity of modern software systems, performance problems can hardly be solved in design stage with traditional techniques, such as testing, validation and verification, especially for large-scale distributed software systems. As a complement, online methods are employed for further improving the reliability of software systems, like runtime monitoring.

Many existing online methods diagnose performance problems based on system call sequences, or called *traces*, which are obtained by various tracing systems or inferred from system logs. The trace records the execution processes of system operations, *e.g.*, the trace "*socket*, *bind*, *listen*, *connect*, *accept*" records the process of a TCP connection operation. Concurrently, the trace also records the related performance metric values of each call, such as execution time and resource usage. Among various technologies, principal component analysis (*PCA*) [2] is widely used in analyzing traces, due to the high accuracy of the diagnosis results and requiring no specific domain knowledge [3]. By separating the space of metric values into normal and abnormal subspaces, the PCA-based methods (*e.g.*, [3], [4]) effectively detect the abnormal traces from a trace set and locate the corresponding root causes in each abnormal trace.

However, according to our experiments, we also validated three shortcomings existed in these PCA-based methods. 1) *Requiring traces with a same call sequence*. PCA is meaningful only for a cluster of traces with a same call sequence. However, the traces of real-world software systems are often different from each other, even for a same operation. Hence, when diagnosing, some trace clusters would be small, which influences the accuracy of the diagnosis results. 2) *Inefficient with long traces*. PCA is a computation-intensive process, especially when calculating the principal components. These methods become inefficient when dealing with long traces, which widely exist in real-world software systems. 3) *Missing performance problems*. The threshold in PCA used for judging abnormal traces relates to each trace instance in the trace cluster. When some very abnormal traces exist, the less abnormal ones would be neglected.

In this poster, we give a segmentation based diagnosis method to improve the traditional PCA-based methods. Two key ideas lead us to this approach. First, *a complex system operation can be divided into many simple actions*. For example, the operation of read a file from Hadoop Distributed File System (HDFS) [6] composes of many actions of reading data blocks. Therefore, we segment long traces into shorter segments before starting PCA and integrate the results after finishing the PCA. The segmentation process brings following benefits. 1) Segments can be treated as short traces, on which PCA is pretty efficient. 2) Simple actions have a higher probability of containing a same call sequence, which potentially leads to a larger capacity of some trace clusters and hence increases the accuracy of the diagnosis results. 3) The whole diagnosis process can be well supported by various parallel technologies. Second, *removing the most abnormal traces is helpful for detecting the less ones*. Since the very abnormal traces conceal the less ones, we repeat the diagnosis process and remove the detected abnormal traces from the trace clusters after each diagnosis. The repeated process leads to finding more problems.

## II. Experiments to Validate the Shortcomings

To validate the aforementioned shortcomings, we implemented a PCA-based method (Ref. [3]) and evaluated it on an open trace data set, called TraceBench [5], which is collected on a real HDFS system. The experiment composes of two parts: fixing the trace length (7 in our experiment) and changing the trace number, and fixing the number (set

TABLE I
EXPERIMENTAL RESULTS

| ***Number*** | 100 | 200 | 300 | 400 | 500 |
|---|---|---|---|---|---|
| ***Time (ms)*** | 31 | 65 | 95 | 130 | 154 |
| ***#Anomaly*** | 12,12,0 | 17,17,0 | 19,2,0 | 30,1,0 | 30,1,0 |
| ***Length*** | 33 | 66 | 99 | 134 | 167 |
| ***Time (ms)*** | 84 | 1,413 | 8,359 | 35,269 | 100,023 |
| ***#Anomaly*** | 0,0,0 | 1,1,0 | 1,1,0 | 1,1,0 | 0,0,0 |

to 7) and changing the length. Table I shows the results, where the items in the *Time* row indicate the diagnosis time of each trace cluster and the items in the *#Anomaly* row are formatted as (total abnormal traces),(correctly detected abnormal traces),(incorrectly detected abnormal traces).

The diagnosis time increases slowly with the trace number and fast with the trace length. Actually, most diagnosis time is spent on calculating the principal components of a performance matrix, whose computation complexity is $O(nl^2)$ [4], where $n$ represents the trace number and $l$ represents the trace length. Hence, the idea of dividing long traces into short segments is helpful for improving the efficiency of diagnosis.

Considering the diagnosis results, the number of incorrectly detected abnormal traces (or called false alarms) in each diagnosis is 0, which indicates the high accuracy of PCA-based methods. On the other hand, the method may miss performance problems. As an example, only 1 of 30 abnormal traces are detected when the trace number is 400 in the upper part of Table I, but another 27 abnormal traces are correctly found in the next diagnosis after removing the detected one. This example motivates our idea of repeating the diagnosis process.

## III. SEGMENTATION BASED DIAGNOSIS

Fig. 1 shows the architecture of our method, in which the content surrounded by the dashed boxes illustrates the process of a traditional PCA-based method, *i.e.*, Ref. [3]. This method mainly consists of three steps: 1) clustering traces into different clusters according to the call sequences and constructing the performance matrices from the performance metric values for each cluster; 2) detecting the abnormal traces in each cluster by applying PCA on each corresponding performance matrix; 3) locating the problematic calls in each abnormal trace and giving corresponding information of these calls. More details can be referred to Ref. [3] and Ref. [4]. Comparing with this traditional method, we add some steps, including segmenting, re-clustering, integrating, removing and repeating.

Before diagnosing, we divide the trace clusters into segment clusters according to the segmentation of corresponding call sequences, and then re-cluster these segment clusters, to reduce the computation during PCA and to potentially enlarge the capacities of certain clusters. Several strategies can be adopted during segmenting, including semantic segmentation, sub-tree based segmentation, manual segmentation, and so on, which we will deeply investigate in future. After diagnosing on each segment cluster, we integrate the results. We consider that the problems detected in a segment are also the problems

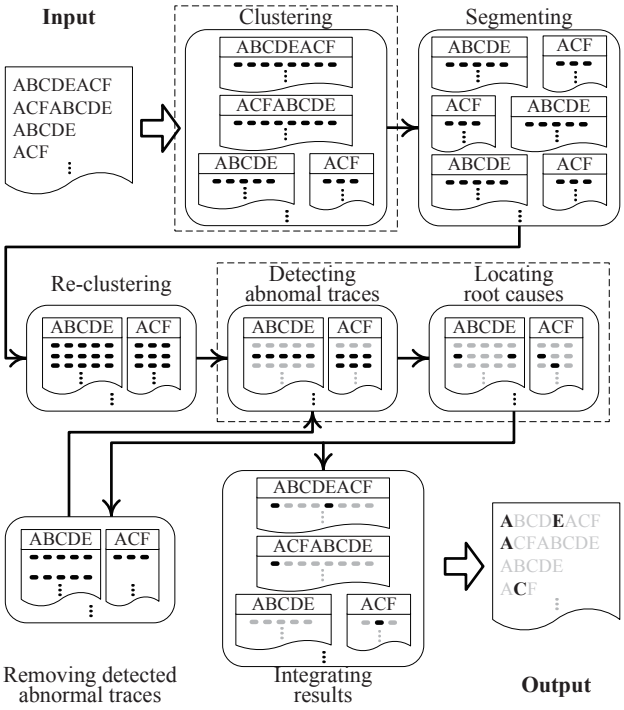


Fig. 1. The architecture of our method.

existed in the corresponding trace. Therefore, we define the problems of a trace as all the problems found in the contained segments. The whole process can be implemented as a MapReduce [6] job, which can greatly expedite the diagnosis.

For each cluster, we repeat the diagnosis process and remove the detected traces after each diagnosis, to find more problems. Once one or more of following conditions are satisfied, the repeated process stops: 1) no more abnormal traces are detected; 2) the proportion between the number of detected traces and the number of all traces exceeds a threshold; 3) the detected traces greatly decrease than the previous analysis. The repeated process is deemed to find more problems and also may bring more false alarms, where the mechanisms like voting and ranking are good choices for dealing with this side effect.

## ACKNOWLEDGMENT

This work is supported by the National 973 Program of China (No. 2014CB340703) and the National Natural Science Foundation of China (No. 61472440 and No. 61272140).

## REFERENCES

[1] C. Wang, S.P. Kavulya, J. Tan, L. Hu, M. Kutare, *et al*, "Performance troubleshooting in data centers: An annotated bibliography," ACM SIGOPS Operating Systems Review, vol. 47, no. 3, pp. 50–62, 2013.
[2] I. Jolliffe, "Principal component analysis," Springer, 2002.
[3] H. Mi, H. Wang, G. Yin, H. Cai, Q. Zhou, *et al*, "Performance problems diagnosis in cloud computing systems by mining request trace logs," in *Proc. of NOMS 2012*, 2014, pp. 893–899.
[4] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proc. of SIGCOMM 2004*, 2014, pp. 219–230.
[5] J. Zhou, Z. Chen, J. Wang, Z. Zheng, and M.R. Lyu, "TraceBench: An open data set for trace-oriented monitoring," in *Proc. of CloudCom 2014*, 2014, pp. 519–526.
[6] Apache. Hadoop. [Online]. Available: http://hadoop.apache.org/