Resource Usage Monitoring in Cloud Environment: Approach via Linux Kernel Module

Byeong-Yong Yi
SysCore Lab.
Sejong University
Seoul, South Korea
yibyeongyong@sju.ac.kr

Sang-Hoon Choi
SysCore Lab.
Sejong University
Seoul, South Korea
csh0052@gmail.com

Ki-Woong Park*
Dept. of Computer and Information Security
Sejong University
Seoul, South Korea
woongbak@sejong.ac.kr

***Abstract*—As the demand for cloud services increases recently, the importance of cloud monitoring techniques in cloud environments is being emphasized. The resource usage monitoring is an essential factor in providing metering and autoscaling services on cloud environments. In terms of performance, One of the features of the resource usage monitoring tool is the overhead incurred by the monitoring tool itself. The monitoring tool with negligible overhead offers accurate billing for cloud service and the exact time of auto-scaling service when to scale-in or scale-out. In this paper, we present a low-overhead resource usage monitoring tool that uses a Linux kernel module. The suggested resource usage monitoring tool of this paper shows negligible overhead incurred by the monitoring tool itself compared to the traditional container monitoring approach. The overhead gap was to 20~25 times in the performance indicators such as Task Clock, Page Fault compared to the traditional monitoring approach.**

Keywords—cloud monitoring; container monitoring; resource usage monitoring; low-overhead monitoring

I. Introduction

As virtualization technology advances, the demands for cloud services are also increasing. One of the properties of cloud services is cloud monitoring. The monitoring is a crucial part of the cloud service because it is used to aggregate the resource usage data. This resource usage data can be used in billing and auto-scaling services. Thus, various resource usage measurement tools used in the cloud environment, such as Nagios [1], Zabbix [2], and Ceilometer [3]. The critical performance factor for cloud monitoring is overhead incurred by the monitoring process and resource usage aggregation interval. The cloud monitoring tools typically access and parse resource usage data from the Linux Proc file system [4] or measure resource usage through a group of hypervisor APIs such as libvirt [5]. However, the Linux Proc file system and libvirt API are not built with considering the cloud environment with thousands of containers are in use. This paper is based on the assumption that there is room for additional performance improvement in the current monitoring approach. Therefore, we propose a monitoring approach optimized for resource usage measurement in the cloud environment through the Linux kernel module that minimizes monitoring overhead. Besides, we performed overhead comparison experiments to demonstrate performance improvement.

*: Corresponding author

This paper is organized as follows. Section 2 deals with cloud monitoring related studies. In Section 3, We present the concept of cloud monitoring with the Linux kernel module. In Section 4, Apply the suggested approach directly to the cloud environment and conduct experiments to measure the degree of performance improvement. Section 5 concludes this paper and describes further research.

II. Related Works

A detailed survey on cloud monitoring conducted by Aceto et al [6]. They arranged the related terms of cloud monitoring that were used ambiguously before. They described the reason why monitoring is a crucial part of the cloud environment, the main properties of cloud monitoring, and the open issues on cloud monitoring. Besides, They compared with commercial cloud monitoring tools and open source monitoring tools.

An architectural concept of the cloud monitoring framework called PvD, suggested by Ahmad et al [7]. They tried to collect data from several cloud instances by using the python module like libvirt for minimizing storage, memory, CPU footprint.

Another approach to collect resource usage data is presented by Bhangale et al [8]. A docker container performance monitoring tool called DockSNAP is a push based monitoring tool that can use via web. DockSNAP reduces manual accessing docker-machine processes and can monitor every docker containers using just a single remote system.

The concept of the absolute metric is proposed by Casalicchio et al [9]. The author divided the types of metrics into relative metrics and absolute metrics. The traditional metric called relative metric. The absolute metric accounts for the accurate, actual resource usage in the host. The absolute metric is suitable for the auto-scaling of containers.

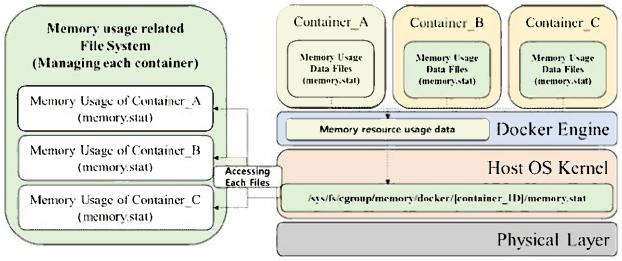

Fig. 1. Traditional Container Resource Usage Monitoring Approach

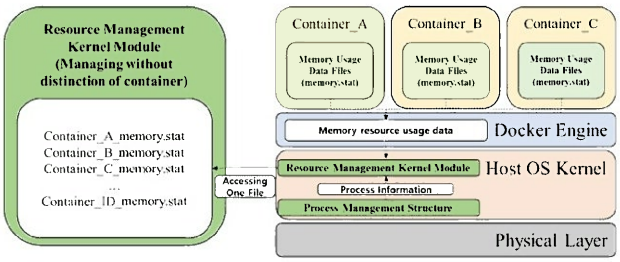

Fig. 2. Suggested Container Resource Usage Monitoring Approach

The relative metric was made for easily controlling, configuring containers, but it is not proper to meet the service level objectives. The author's previous study explained there was a quite value gap between absolute and relative metrics [10].

III. Monitoring via Linux Kernel Module

In this section, we present a Linux kernel module for cloud monitoring. We describe the reason why we use the Linux kernel module and compare with the cloud monitoring approach used in the past. And then, we describe our approach to cloud monitoring.

A. The Linux Kernel Module

The Linux kernel module allows the user-level to use the kernel-level functionality. The Linux kernel module refers to Linux code that can be loaded into the Linux kernel and, when needed, conveniently used kernel-level features [11]. This can prevent memory wastage, enable kernel-level features that are impossible at the user-level, without compiling the entire kernel, and guarantee performance. The Linux kernel module acts as a stepping stone between the user level and the kernel level. The Linux provides resource usage information to users via built-in tools include free, top commands and meminfo, vmstat under the proc file system. Since these are not designed, for the cloud environment from the beginning, container task automation platforms such as Kubernetes [12] measure container resource usage through monitoring tools developed for containers based on cAdvisor [13]. However, cAdvisor often reports performance issues such as increased CPU usage and memory usage of the tool itself, and the problem of cAdvisor can affect many tools built on cAdvisor. Therefore, this paper aimed at developing the monitoring tool optimized for the cloud environment by minimizing the resources wasted in the process of collecting metrics through the Linux kernel module.

B. Comparison between traditional monitoring approach and suggested monitoring approach

The monitoring structure of the traditional and suggested cloud environment, as shown in Fig. 1 and Fig. 2. In the traditional monitoring approach, resource monitoring is conducted by collecting resource usage information from each container using the cgroups [14]. In order to get resource usage information obtained with multiple containers floating, memory usage written file-system of each container had to be accessed. This process causes much more overhead than expected. Given the cloud environment in which thousands of containers are floated, this is hard to ignore in practical situations. The monitoring scheme we proposed in this paper uses the Linux kernel module to guarantee the metric aggregation interval and lower overheads than traditional monitoring approach. The suggested Linux kernel module is based on cgroups to get container memory resource usage. The difference between traditional monitoring methods is the operation to access resource usage data. The suggested approach manages the resource usage data of containers with a single file. Through the list of containers imported from the process list, the resource management kernel module gets the whole resource usage data and managing them with a single file. This module manages resource usage data in the form of resource type with container ID. So, instead of accessing each file system of containers for collecting resource usage data, we can get resource usage data of thousands containers by just one access to single file. This single file accessing process can lower overhead incurred by monitoring than traditional monitoring approach.

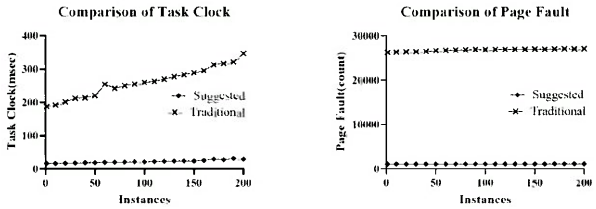

Fig. 3. Overhead Comparison of Traditional Monitoring Approach and Suggested Monitoring Approach

In order to verify the performance of the suggested method, we evaluate our approach to compare with the traditional monitoring approach in the next section.

IV. Implementation and Overhead Evaluation

The experimental environment used Ubuntu 18.04 as the operating system. Memory was allocated 16gb and CPU was assigned 8 processors. The Docker engine version was 19.03.2. Overhead measurements were conducted by sequentially increasing the number of Docker instances from 1 to 200. Perf [15] was used as an overhead measurement tool. The average value of the results of 10 times of overhead measurements was recorded to compare the overhead between the traditional monitoring approach and the suggested monitoring approach. The result of the experiment is shown in Fig. 3. The benchmark indicators used in the experiment are Task Clock and Page Fault. Task Clock means the whole running time of process or task. Page Fault is a metric which occurs when no pages loaded in memory are needed. Experimental results show that there was a difference in task clock, and the difference increases with each instance increase. The difference was up to 20 times. The number of Page Fault occurrences was 25 times different from the beginning.

V. Conclusion

In this paper, we suggested a resource usage monitoring approach in a cloud environment by using the Linux kernel

module. The proposed method showed a performance advantage when measuring metrics about memory. In the future study, we will try to expand available resource usage data such as CPU, disk I/O and network I/O. Also, experiments in this paper were conducted in a limited range of docker environments, but future research will suggest a monitoring approach that can be commonly applied after analyzing various cloud environments.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2017R1C1B2003957) and supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00420, No.2019-0-00273).

References

[1] Nagios: https://www.nagios.org/

[2] Zabbix: https://www.zabbix.com/

[3] Ceilometer: https://docs.openstack.org/ceilometer/latest/

[4] Proc: http://man7.org/linux/man-pages/man5/proc

[5] libvirt: https://libvirt.org/

[6] Aceto, Giuseppe, et al. "Cloud monitoring: A survey." *Computer Networks* 57.9 (2013): 2093-2115.

[7] Ahmad, Bashar, Iris Leitner, and Michael T. Krieger. "PvD: A Lightweight Distributed Monitoring Architecture for Cloud Infrastructures." *2015 IEEE Fourth Symposium on Network Cloud Computing and Applications (NCCA)*. IEEE, 2015.

[8] Bhangale, Pratik, Vishal Rathod, and Mayure Pate. "DockSNAP: Performance Monitoring of Docker Systems."

[9] Casalicchio, Emiliano, and Vanessa Perciballi. "Auto-scaling of containers: The impact of relative and absolute metrics." *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 2017.

[10] Casalicchio, Emiliano, and Vanessa Perciballi. "Measuring docker performance: What a mess!!!." *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*. ACM, 2017.

[11] Salzman, Peter Jay, Michael Burian, and Ori Pomerantz. "The linux kernel module programming guide." (2007).

[12] Kubernetes: https://kubernetes.io

[13] cAdvisor: https://github.com/google/cadvisor

[14] cgroups: http://man7.org/linux/man-pages/man7/cgroups.7.html

[15] perf: http://www.brendangregg.com/perf.html