



HAL
open science

FSObserver: A Performance Measurement and Monitoring Tool for Distributed Storage Systems

Xiao Zhang, Lanxin Kong, Shunyi Zhu, Zhanhuai Li, Xiaonan Zhao

► **To cite this version:**

Xiao Zhang, Lanxin Kong, Shunyi Zhu, Zhanhuai Li, Xiaonan Zhao. FSObserver: A Performance Measurement and Monitoring Tool for Distributed Storage Systems. 15th IFIP International Conference on Network and Parallel Computing (NPC), Nov 2018, Muroran, Japan. pp.142-147, 10.1007/978-3-030-05677-3_14 . hal-02279557

HAL Id: hal-02279557

<https://inria.hal.science/hal-02279557v1>

Submitted on 5 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

FSObserver: A Performance Measurement and Monitoring Tool for Distributed Storage Systems

Xiao Zhang^{1,2}, Lanxin Kong¹, Shunyi Zhu¹, Zhanhuai Li^{1,2}, Xiaonan Zhao^{1,2}

¹ School of Computer Science Northwestern Polytechnical University Xi'an China

² MIIT Key Laboratory of Big Data Storage and Management, Xi'an China
zhangxiao@nwpu.edu.cn

Abstract. It is a big challenge to measure and monitor the performance of a large-scale distributed storage system accurately. We present a flexible approach based on the message analysis, named FSObserver, which can accurately and fine-grained trace individual request or response by observing network traffic. Experiments results show that our approach can get accurate performance with slight performance degradation.

1 Introduction

Over the past few years, there are tremendous efforts to evaluate and debug the performance problems of large-scale distributed storage systems. Some practitioners concentrate on monitoring individual devices and machines independently. Some researchers focus on detailed analysis of all messages by inserting some unique IDs into messages during instrumenting the system. Some others are immersed in the study of the storage system log[1]. System evaluation based on inner messages analysis has been intensively studied in [2][3]. When designing a monitoring and evaluation system, we should consider the independence, accuracy, high performance, and broad-applicability.

In this paper, we propose FSObserver, an out-of-band approach to capture performance related messages between clients and servers. It extracts performance characteristics from the messages. The core idea of FSObserver is to capture the request and reply messages between clients and servers. It extracts the time, size and operations information from messages. The size of this information is very small compared with the size of the messages. By analyzing each individual message, we can accurately evaluate the performance characteristics such as IOPS, throughput, and latency.

In the following sections, we discuss how to monitor and evaluate the Ceph distributed file system, and show the experiment results.

The remainder of the paper is organized as follows. In Section 2 we present the related work that evaluates Ceph and other distributed storage systems. Section 3 describes the architecture of FSObserver, and explains how it works on the Ceph distributed file system. Section 4 illustrates the performance evaluation results of FSObserver and other widely used tools fio³ respectively. Finally, we conclude this paper and present the future work of this study.

³ <http://freshmeat.net/projects/fio/>

2 Related work

Performance of a distributed storage system is very important in data centers. Past studies proposed various methods to debug and diagnose the systems. They concentrated on the in-band and out-of-band monitoring systems, black box and white box, intrusive, and log analysis and so on. There are several tools developed to monitor the performance of Ceph clusters⁴. Many black-box diagnosis techniques have been devised for performance evaluation in distributed systems. Dianna *et al.* used 5 tools to evaluate the performance and scalability of the Ceph distributed storage system[4]. Wang *et al.* evaluated the file and block I/O performance and scalability of Ceph, using a commercial high-end storage system[5]. Computer-system logs provide a glimpse into the states of a running system, and system diagnostics research around logs never stopped[6].

3 Design and implement

3.1 Architecture of FSObserver

The FSObserver are designed to measure different aspects of performance in a large-scale distributed file system by packets analysis. We designed a flexible packet analyzer, which can capture related packets and save a little information from the payload. The analyzers can be turned on/off by a controller. These designs can get performance data without too much impact on the system. Fig. 1(a) shows the architecture of FSObserver. There are 3 kinds of components in the FSObserver. The recorder processes can be turned on/off dynamically.

- Recorder

It captures related packets using libpcap⁵, which is a portable C++ library for network traffic capture. When a recorder process captures a read request packet, it analyzes the header of payload and outputs key information, such as time, transaction id, data length. We measure the impact of the recorder in several read/write scenarios. The performance impact is less than 5%.

- Controller

It can start/stop some observers according to the administrator's input. The FSObserver can be used flexibly for various purposes.

- Observer

This is a python program for analyzing the output of observers. It gets IOPS by counting how many finished IO requests in a given period in the results. In our prototype, we output the performance information into a text file inside the nodes under test. Meanwhile, we run the observer on the same nodes. The performance characteristic of the nodes including IOPS, throughput, and latency can be calculated through one sequential scan of the text file. In a large-scale system, we can use the mechanism similar to ganglia. As

⁴ <http://www.ceph.com/performance/>

⁵ <http://www.tcpdump.org/>

shown in Fig. 1(b), we divide nodes into different monitor group. The nodes in one group save the raw performance data in a database like MySQL or RRDtool.

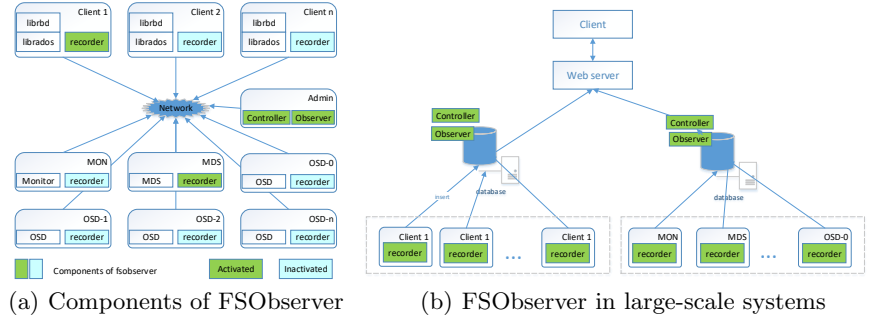


Fig. 1. Architecture of FSObserver

3.2 Implement of FSObserver for Ceph

After capturing a related packet using libpcap, it extracts the necessary information for performance measurement. Our main purpose is to get performance data, so we only need to analyze the messages with tag equals to 0x07. Further, we can only capture and analyze messages from a specific client.

The observer program analyzes the results from recorders. It can get IOPS, throughput and latency data from the results. For example, IOPS is calculated by counting the number of transactions finished in a given period. To analyze the performance of a certain client, we only need to deploy a recorder on the client. We can also get the same metric from records from all related OSD nodes with the client. We put the implementation on the GitHub⁶.

4 Evaluation

We evaluate the accuracy and application of our tools. First, we compare the test results of FSObserver and widely used benchmarks to show the accuracy of our tools. Then, we measure the performance of a real user application to demonstrate how to use FSObserver in a real environment. The test environment comprised 10 commodity servers. 6 nodes work as Ceph servers, 4 nodes act as Ceph clients. The release version of Ceph is 12.2.4 Luminous.

4.1 Block storage interface

We use fio to test the performance of block storage interface. In our environments, we first test the performance use fio, then we lunch FSObserver and test the performance with fio again. We get two performance data from fio, and one performance data from FSObserver.

⁶ <https://github.com/zhangxiao2000/fsobserver>

There are 6 different workloads used in our test, including sequence read, write, and mixed workloads and random read, write, and mixed workloads. For each workload, we test performance with different block sizes from 4k to 128k. Due to the page limitations, we only show the results of read and write. From these figures, the results of fio are almost the same, while one is taken without a recorder, the other was taken with FSObsver is working. The CPU and memory used by FSObsver are also very small. According to our experiments, it only used less than 0.3% CPU during the whole test.

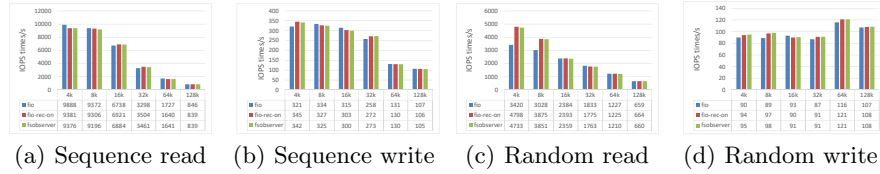


Fig. 2. The IOPS measured by fio and FSObsver

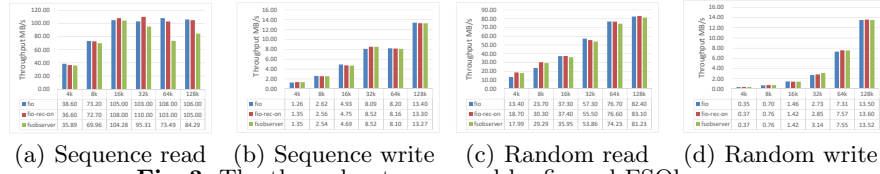


Fig. 3. The throughput measured by fio and FSObsver

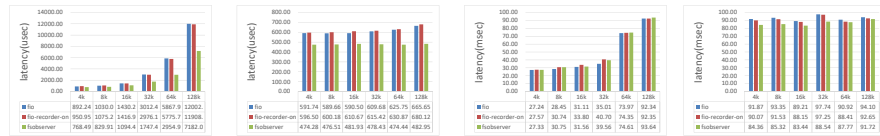


Fig. 4. The latency measured by fio and FSObsver

4.2 Capture real workloads

In this part, we demonstrate how to get the I/O sequence of a real application. The process of compiling a Linux kernel is a complex task. There are 67 thousand files in the Linux kernel 4.16.4. During the compiling process, several compilers read thousands of files and generate about 71 thousand new files. Fig.5 shows the IO throughput per minutes during the compiling process. We can find that during the compiling process, the write throughput is kept at a high level.

5 Conclusion

In this paper, we present a flexible performance monitoring tool for large-scale distributed storage systems. We have implemented it for Ceph. The experiments

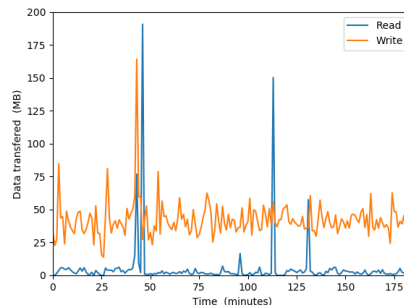


Fig. 5. Real workloads of compiling a Linux kernel

show that it can get coincident performance data with other widely used tools. We compared the accuracy with wide adapted benchmarks and measure a performance for a real application.

Acknowledgment

This work is supported by the NFS of China under Grant No.61472323 and No.61502392 and the Ministry of Science and Technology of China, National Key Research and Development Program (No.2018YFB1004401).

References

1. D. Yuan, J. Zheng, S. Park, Y. Zhou, and S. Savage, “Improving software diagnosability via log enhancement,” *ACM Transactions on Computer Systems (TOCS)*, vol. 30, no. 1, p. 4, 2012.
2. Y. Zhao, Y. Cao, Y. Chen, M. Zhang, and A. Goyal, “Rake: Semantics assisted network-based tracing framework,” *IEEE Transactions on Network and Service Management*, vol. 10, no. 1, pp. 3–14, 2013.
3. M. Määttä and T. Rätty, “Automatic model creation to support network monitoring,” *IEEE Access*, vol. 2, pp. 142–152, 2014.
4. D. Gudu, M. Hardt, and A. Streit, “Evaluating the performance and scalability of the ceph distributed storage system,” in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 177–182.
5. F. Wang, M. Nelson, S. Oral, and Atchley, “Performance and scalability evaluation of the ceph parallel file system,” in *Proceedings of the 8th Parallel Data Storage Workshop*. ACM, 2013, pp. 14–19.
6. A. Oliner, A. Ganapathi, and W. Xu, “Advances and challenges in log analysis,” *Communications of the ACM*, vol. 55, no. 2, pp. 55–61, 2012.