Aridac: Adaptive Resource Isolation of Non-volatile Devices Under Containerized Environment Project Final Report

Xiang Yue
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: xiangyue@andrew.cmu.edu

Xuan Peng
Information Networking Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: xuanpeng@andrew.cmu.edu

Zeyu Wang
Information Networking Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: zeyuwang@cmu.edu

***Abstract*—Container technology makes cloud computing possible by offering resource isolation and scalability, however, resource sharing brings the problem where heavy containers consume most of the resources and break the fairness. To achieve fairness and maintain a good overall system performance, we propose an adaptive resource isolator for block device namely Aridac, which could adjust the disk resource quota of containers in running time.**

1. Introduction

Cloud computing platforms, as a typical instance of infrastructure-as-a-service, nowadays have become the first choice for internet application developers to deploy their services. The main reason behind the popularity is - cloud computing offers reliability, convenience, isolation, and scalability, along with the pay-as-you-go pricing model. To achieve those features, the quality of service must be ensured with the achievement of resources isolation technologies.

Container, as the core part of isolation, creates an illusion for customers as owning the entire system. This lightweight virtualization notion has been widely leveraged by cloud service providers. Based on Linux Cgroup, which offers the ability to isolate resource (e.g. namespace, network I/O, disk I/O), container technologies and applications raised, like Linux Container (LXC) [1] and Docker [2]. Those form the basic units of cloud platforms and are still a great domain for cloud providers to dive deep into.

However, behind the great idea of containerization, there come new problems with resources isolation in practice. One of the glaring issues to resolve is resource allocation fairness, which means avoiding the resources allocated to containers interfering with each other. For instance, when there are several containers within a node and one of them is running heavy workloads with intense I/O operations, the resources of other nodes can be consumed, greatly delaying the delivery of tasks, like the completion of file reading, the transmission of RPC requests, etc. In addition, Xavier et al. [3] has also analyzed and confirmed the interference of heavy disk workload without limits on resource allocation can degrade the overall performance of the system. Therefore, to avoid sudden delay of container processes, a limitation of resource quota is a good way to control.

Although there are already some tools (cgroup, trickle) for adjusting resource quota, the problem is, those tools only offer fixed limitations of resource quota, whereas, in industry, cloud providers can never know how to adjust quota in advance or in real-time - the application in containers and containers themselves are dynamically created, killed and changed frequently. In addition, one of our group members met with the same problem with an intolerant latency of services when resources are consumed mostly by a disk-intensive container. Some works [4] manages to alleviate the issue. Since there are still few efforts working on this topic in both academy and industry, our group planned to look into a dynamic way of isolation adjustment.

We plan to propose the Aridac, an Adaptive Resource Isolation algorithm of Non-volatile Devices under heterogeneous containerized environment. The basic idea is to monitor, collect and analyze the resource usage habit of containers as time goes on, and allocate their resources to ensure fairness under certain rules. The quota of disk I/O can be allocated by, for instance, the priority of the application, or by containers' history maximum usage. And the objective is to achieve fairness. Besides the algorithm, we plan to design the testing workloads and design experiments for comparing fairness between different policies with specific metrics. Also, we will provide the corresponding benchmark data for further research. To narrow down the scope, we focus on disk isolation, which can be extended to the network I/O or other resources allocation scenario. The basic tools of

system design and experiments include cgroup, bash scripts.

The rest of the proposal is organized as follows: Related Work section shows the relevant works of disk I/O isolation; Methodology section shows how we will define the effectiveness of Aridac; Goals includes our promising results of research; Final paper plan illustrates how we will write the final paper.

2. Related Works

Currently, there are not so many efforts on research of container isolation, especially on disk I/O. However, some researchers interested in disk I/O isolation has some works for us to refer. Merchant et al. [5] proposed Maestro, focused on the fairness and performance of disk I/O allocation for different applications. He hypothesized there are different performance targets (throughput or I/O directed) for each application, and developed a policy to allocate the resources to them to reach the best overall performance. Arunagiri et al. [6] developed the FAIRIO algorithm to schedule prorated I/O resources for fair allocation, which is a more general solution. Li et al. [7] provided an automatic framework to monitor storage contention and optimize bandwidth utilization. In addtion, for the different scenario on solid state disks, Jo et al. [8] proposed a I/O load balancer for it.

The above ideas give us a good inspiration for container disk I/O isolation. However, those works are not strongly related to our topic. Take Maestro as an instance, it focuses on application I/O balancing on large disk arrays, so its algorithm takes disk I/O ports and preset application targats in consideration with optimization function, which is not aligned with our topic.

A naive idea is to treat containers as applications and apply the above solutions. However, different from applications, containers' behaviors are more complicated. In this paper, we believe each container has a dynamic usage pattern and is variable from period to period. Therefore, adjusting the policy to allocate resources cannot be avoided.

As for resource isolation tools, Linux control group (cgroup) is the original resource isolation tools, and docker's isolation capacity is built on this. We can easily find the corresonding cgroup of each docker container to limit there resource usage.

3. Methodology

4. Design & Implementation

5. Evaluation

In this section, we aim to prove two major things. First, the scenario of resource preemption exist; second, to test if our policy is able to prevent the resource preemption. and the quota limit can adapt the dynamic desire of application (e.g. increase or decrease).

5.1. Workload Design

- **Workload 1** Two containers A and B running with same rate (200KiB/s) in the beginning. At some point, B's I/O usage goes high without limit (2MiB/s), preempting A's resource.
- **Workload 2** Two containers A and B running with same rate (200KiB/s) in the beginning. At some point, B's I/O usage goes high without limit (2MiB/s), preempting A's resource. After a while, B's usage goes back to normal rate (200KiB/s).

Our policy should prevent the above resource preemption. Also, the quota allocated by our policy should keep align with the containers' desire dynamically. The only difference between two workloads is that, in workload 2, B's usage goes back to its normal rate, where our policy should be able to adjust the quota low.

6. Future Work

7. Conclusion

8. Problems & Lessons Learned

References

[1] "Linux containers." [Online]. Available: http://lxc.sourceforge.net

[2] "Docker." [Online]. Available: https://www.docker.com/

[3] M. G. Xavier, I. C. D. Oliveira, F. D. Rossi, R. D. D. Passos, K. J. Matteussi, and C. A. F. D. Rose, "A performance isolation analysis of disk-intensive workloads on container-based clouds," *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pp. 253–260, 2015.

[4] S. Ahn, K. La, and J. Kim, "Improving I/O resource sharing of linux cgroup for NVMe SSDs on multi-core systems," in *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[5] A. Merchant, M. Uysal, P. Padala, X. Zhu, S. Singhal, and K. G. Shin, "Maestro: quality-of-service in large disk arrays," in *ICAC '11*, 2011.

[6] S. Arunagiri, Y. Kwok, P. J. Teller, R. Portillo, and S. R. Seelam, "Fairio: An algorithm for differentiated i/o performance," *2011 23rd International Symposium on Computer Architecture and High Performance Computing*, pp. 88–95, 2011.

[7] Y. Li, X. Lu, E. L. Miller, and D. D. E. Long, "Ascar: Automating contention management for high-performance storage systems," *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–16, 2015.

[8] M. H. Jo and W. W. Ro, "Dynamic load balancing of dispatch scheduling for solid state disks," *IEEE Transactions on Computers*, vol. 66, pp. 1034–1047, 2017.