Aridac: Adaptive Resource Isolation of Non-volatile Devices Under Heterogeneous Containerized Environment

Xiang Yue
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: ***@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—The abstract goes here.

1. Introduction

This demo file is intended to serve as a "starter file" for IEEE Computer Society conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds
August 26, 2015

1.1. Subsection Heading Here

Subsection text here.

1.1.1. Subsubsection Heading Here. Subsubsection text here.

2. Methodology

2.1. Evaluation

Supposedly, to evaluate the effectiveness of Aridac, we will carry out benchmark experiments under different kinds of workload. Most experiment setttings will be invariants, including the host machine (physical machine), container software, container version, testing programs, etc. The only variant is the isolation policy. One setting will use Aridac, while the other one simply does not use any.

2.2. System Measurement Metrics

The detailed discussion of the metrics selection is still ongoing and will be posted here soon. This is the very essential component of our project, and it is also a very difficult part, for which we need to be very careful and take many aspects into consideration. So we choose to post a placeholder here, instead of rushing with some metrics that are not good enough.

In general, we want the metrics to be correct, precise, and able to take the semantics of different I/O operations into consideration. For example, between the extremely heavy disk write operations introduced by image pulling during deployment in one container, and moderately heavy disk write operation from a critical web service in the other container, we definitely want Aridac to catch difference among those operations, whether through some OS observability methods or manually injected configurations. It's fair to say that this metrics will determine the effectiveness of Aridac.

3. Goals

The overall goal is to finish implementing Aridac, testing it under various kinds of workload scenarios, measuring how much improvement we achieve, and finally, analyzing the overhead brought by Aridac and discussing the trade-offs.

To evaluate Aridac, we could start from the following perspectives:

- The completeness of implementation. i.e., whether the isolator can be applied to the resource contention scenario that we mentioned before.
- To what extent does Aridac help. i.e., how much peak & average disk throughput improvement can we gain with Aridac.
- What's the price we need to pay? i.e., how many extra machine resource will be consumed by Aridac.

Here, we set 3 concrete goals in terms of the final progress, which represent 75%, 100%, and 125% degree of completion, respectively.

For the 75% completion goal, our aim is to cover the following aspects:

- Finish implementing Aridac, the adaptive disk resource isolator.

- Make sure that Aridac could run successfully without crashing or bugs.
- Aridac achieve a better disk I/O throughput than the non-optimized version.

For the 100% completion goal, we will strive to finely tune the isolation policy in order to achieve an optimal resource sharing among containers, which will be gauged by overall throughput of the system. We will use the following steps to check whether Aridac has achieved this goal:

- A set of tests carrying different workload will be designed to simulate typical resource sharing situations in real-world datacenter containerized environment.
- Aridac must excel in all test cases that we designed.
- Concrete benchmark data will be given, specifying by how much Aridac is leading in terms of system I/O throughput.

Beyond the 100% goal, if everything goes well, we will further develop a fuzzy disk I/O workload generator, so that all possible real-life cases will be covered, incluing assorted corner cases and rare cases. Our anticipation for this 125% goal is that Aridac could survive through this fuzzy test. We will also carry out benchmarking, and do some statistical work to show how well Aridac could be in a totally random environment.

4. Final Paper Plan

In our final paper, we will first introduce the resource isolation problem of the container environment and discuss the background and related work. Next, we will briefly outline the system design and implementation of Aridac. Last, we will describe our experimental setup, including what machines and environments were used, what workload we developed to test our implementation. The most crucial part of the paper will be concerned with describing and evaluating our methods for container resource isolation. We will provide performance analyses, and conclude with a discussion about potential optimization schemes, and if possible, a set of benchmarks for more detailed performance evaluation.

The early stage of our project will be analyzing existing issues caused by weak isolation of the container environment, researching existing relative works on different levels of virtualization technologies and container resource isolation, and finding out a doable logic for performing a dynamic resource isolation scheme in the container environment. Our project will then focus on developing a dynamic resource isolation policy for the container environment, to achieve the goal of balancing the resources among multiple containers running on the same physical machine. We will implement a program working on dynamic resource allocation and isolation for containers sharing the same underly resources, and may try out different policies for resource allocation of network bandwidth and disk I/O of the physical machine to learn their performance characteristics.

We will design and develop a set of workloads to simulate some most common applications running in the industry. Then we will let a few containers run that workload and monitor their resource and performance, to see if any rapid resource obtaining will happen and broke the resource allocation balancing. If we can reproduce the scenario of container resource exhausting caused by weak isolation, then we may follow up by testing whether it strengthens the resource isolation among containers and result in better overall performance.

To measure the success of our project, we will focus on completeness rather than performance. But we will examine if there are any possible optimization for the implementation and leave it as potential future works. Other possible topics include comparisons between static isolation approaches and our dynamic isolation approach, discussing the pros and cons of using different levels of encapsulations and virtualization technologies, and even recursive virtualization for more stable overall performance. We will also try to figure out the weight of different resources in the container environment, discussing trade-offs and complexities. Hopefully, the results and discussions will help us know the effectiveness and generality of the dynamic isolation approach for various working scenarios.

Here are some questions that our experiments will answer:

- What kinds of scenarios will require strong resource isolation for applications running in the container environment?
- Is dynamic resource allocation and isolation beneficial for the overall performance of applications running in the container environment?
- What are the appropriate resource allocation approaches for applications with different characteristics?
- How strong is the isolation provided by the dynamic isolation and allocation approach?
- How stabilized has the performance of the container environment been achieved after adopting stronger isolation?
- What is the overall overhead of our approach?

References

[1] Improving I/O Resource Sharing of Linux Cgroup for NVMe SSDs on Multi-core Systems, Sungyong Ahn, and Kwanghyun La and Jihong Kim, HotStorage, 2016