A Reliablity Analysis Framework for Cloud Storage Systems

Mai Zheng[†] Joseph Tucek[‡] Feng Qin [†] Mark Lillibridge[‡]
[†] *The Ohio State University* [‡] *HP Labs*

1 Motivation

People have high expectations of data storage. It is frustrating when your browser crashes, but it is unacceptable when the payroll database loses your paycheck, and heartbreaking when a cloud storage service loses your irreplaceable family photos. Thus, we need highly reliable storage systems that can protect data no matter what happens.

However, such high standard of reliability is difficult to achieve. Besides normal cases, storage systems must be able to handle many adverse events, such as system crashes, disk failures, power outages, and so on. So the reliability guarantees usually comes at a cost in high complexity, especially when high performance must be achieved as well. This is particularly true for cloud storage systems (e.g, distributed file systems, distributed databases, cloud storage services, etc.), which must be built on top of the already-complicated storage stack on each individual machine (e.g., block devices, logical volume manager, local file systems, etc.) and must consider various failures as the norm rather than the exception [2]. As a result, today's storage systems become more and more difficult to test and to reason about.

2 Our Preliminary Studies

As a first step towards building highly reliable storage systems, we have built two reliability analysis frameworks for storage systems on a single machine [3, 4]. The first framework [4] focuses on flash-based solid-state disks (SSDs). We evaluate fifteen commodity SSDs from five different vendors using more than three thousand fault injection cycles in total. Our experimental results reveal that thirteen out of the fifteen tested SSD devices exhibit surprising failure behaviors under power faults, including bit corruption, shorn writes, unserializable writes, metadata corruption, and total device failure. The second framework [3] focuses on databases running on a single machine. We design special workloads to check the atomicity, consistency, isolation, and durability (ACID) properties under an idealized fault model. The framework includes a record/replay subsystem to allow the controlled injection of simulated faults, a ranking algorithm to prioritize where to fault based on patterns collected from traces, and a multi-layer tracer to diagnose root causes. Using our framework, we study 8 widely-used databases, ranging from open-source key-value stores to high-end commercial OLTP servers. Surprisingly, all 8 databases exhibit erroneous behavior.

The issues exposed by our frameworks is a wake-up call. Even for storage systems on a single machine, the traditional testing methodology may not be enough. Instead, we need carefully-designed workloads as well as intelligent fault injection techniques. Moreover, we find that there are gaps of understanding or assumptions among different communities (e.g., file systems developers and database developers) in terms of the behavior of the interface between two different layers in the storage stack. The situation will likely be worse if more storage layers or more machines are involved.

3 Our Proposed Research on CloudLab

Analyzing the storage systems on a single machine, although important, is far from enough. In this big data era, many data are now stored and managed in the cloud. If even the relatively matured single-machine storage systems can exhibit unexpected or erroneous behaviors, it becomes important as well as emergent that we perform similar in-depth analysis on cloud storage systems, which add more layers on top of the local storage stack and are responsible for protecting much more data.

The basic idea of the distributed reliability analysis framework is simple: workloads and checking logic to stress the storage systems and check consistency and integrity; fault-injection subsystem to simulate certain failure behavior; record-and-replay subsystem to allow effi-

cient testing and reproduction; tracing subsystem to help diagnosis. However, there are multiple challenges.

The first challenge of building such an analysis framework is to setup a cloud storage system that we have full control of. To this end, we need an infrastructure like CloudLab [1], which can provide sliced hardware resources and allows us to build a distributed systems with storage software that we are interested in.

One feature of CloudLab, which is "control and visibility all the way to the bare metal", is extremely important to our analysis framework. To achieve high-fidelity and make the analysis framework transparent to the cloud storage systems under testing, we need to carefully design where to inject faults. We will likely consider software layers below the systems under testing. Thus, full access to the low-layer software is necessary.

Moreover, to fully understand the potential failures exposed to the user by the cloud storage systems, we will study the failure propagations among different layers. This requires monitoring the interactions among different layers and gathering corresponding traces, from the block devices in the local storage stack to the distributed storage software. Also, to make the analysis more comprehensive, we will tune the parameters of each individual layer in the storage stack. For example, we will consider the impact of changing the journaling mode of the local file system on the behavior of a higher-level distributed databases. All of these require the capability of accessing and modifying each and every software layer in the cloud storage systems.

The cloud environment introduces many other challenges for the analysis. For example, synchronization of events from different machines, which is necessary for combining different traces, is a new issue to the analysis framework. To solve these challenges, our framework will likely incorporate the idea of some classic solutions to these classic problems in distributed systems. It would be helpful if infrastructures like CloudLab could also provide some functionalities commonly used in the distributed environment (e.g., synchronization primitives) for building high-level software framework.

References

[1] CloudLab. http://www.cloudlab.us.

[2] GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2003), SOSP '03, ACM, pp. 29–43.

[3] ZHENG, M., TUCEK, J., HUANG, D., QIN, F., LILLIBRIDGE, M., YANG, E. S., ZHAO, B. W., AND SINGH, S. Torturing databases for fun and profit. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)* (Broomfield, CO, Oct. 2014), USENIX Association, pp. 449–464.

[4] ZHENG, M., TUCEK, J., QIN, F., AND LILLIBRIDGE, M. Understanding the robustness of SSDs under power fault. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST'13)* (2013).