Byzantine fault-tolerant storage: Why now, how much does it cost, and what improvements are expected?

James Hendricks
Carnegie Mellon University

Gregory R. Ganger
Carnegie Mellon University

Michael K. Reiter
University of North Carolina at Chapel Hill

January 15, 2008

Modern storage systems must tolerate a variety of faults and errors that result from software bugs, hardware failures, or even malicious users. Many systems take an ad-hoc approach, considering the effects of a particular failure such as a corrupted checksums, a faulty stripe of erasure-coded data, or a misdirected write. Such ad-hoc approaches, however, leave resiliency holes that may result in data loss or corruption. An alternative approach is to tolerate arbitrary, or Byzantine faults. But Byzantine fault-tolerant storage systems often require additional storage nodes, network bandwidth, and computational resources.

This work-in-progress talk will present a Byzantine fault-tolerant storage system that we are developing that overcomes the problems of previous approaches. In particular, a prototype that we have built of our system performs nearly as well as similar cluster-based systems that tolerate only crashes. Our protocol requires no extra storage nodes, and it uses a similar amount of network bandwidth and computational resources. The prototype currently implements an atomic block-store system–that is, clients read and write blocks concurrently (locks are not needed). We hope to expand the prototype to provide a distributed filesystem interface with (near) POSIX semantics.

Byzantine fault tolerance is not yet a common feature in commercial storage systems for three big reasons. First, until recently, crashes were viewed as the primary cause of data loss. Second, upgrading certain legacy monolithic storage architectures to tolerate Byzantine faults would require substantial additional hardware. Third, Byzantine fault tolerance has a (sometimes deserved) reputation for poor performance. These three reasons, however, stand to be overturned over the next few years. As storage systems grow in size and complexity, they are experiencing and must tolerate faults other than crashes, such as software bugs (e.g., corrupted checksums) or hardware faults (e.g., misdirected writes). Our protocol requires no additional hardware for cluster-based storage systems, which are becoming increasingly popular and important. Finally, our protocol is fast–our prototype performs within 10% of systems that tolerate only crashes when reading and writing 64 kB blocks and tolerating up to 6 arbitrarily faulty servers.

Our Byzantine fault-tolerant storage protocol tolerates arbitrary server and client faults, which frees system designers from worrying about particular failure modes (e.g., questions such as, “in which ways can data be silently corrupted by a faulty hard drive”). Technology trends are making protocols such as this one increasingly relevant, and we have shown that such protocols can be fast. If I am allowed to present a work-in-progress talk, I will describe why this problem matters, where the state of the art stands, and what improvements we expect to make over the next year or so.