Science | DOI:10.1145/1538788.1538794

Alex Wright

Contemporary Approaches to Fault Tolerance

Thanks to computer scientists like Barbara Liskov, researchers are making major progress with cost-efficient fault tolerance for Web-based systems.

As more and more data moves into the cloud, many developers find themselves grappling with the prospect of system failure at ever-widening scales.

When distributed systems first started appearing in the late 1970s and early 1980s, they typically involved a small, fixed number of servers running in a carefully managed environment. By contrast, today's Web-based distributed systems often involve thousands or hundreds of thousands of servers coming on and offline at unpredictable intervals, hosting multiple stored objects, services, and applications that often cross organizational boundaries over the Internet.

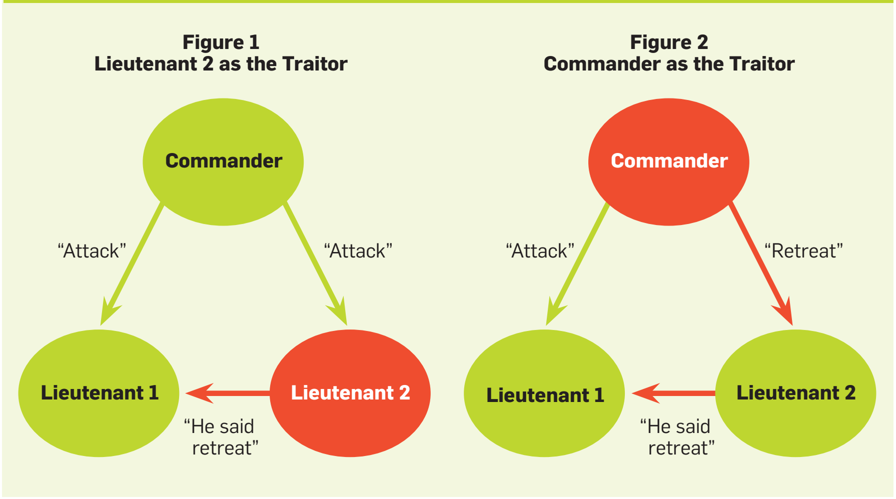
"In a cloud we have relatively few sites that are loaded with a huge number of processors," says Danny Dolev, a computer science professor at The Hebrew University of Jerusalem. "Fault tolerance needs to provide survivability and security within a cloud and across clouds."

In this deeply intertwined environment, software designers have to plan for a bewildering array of potential failure points. Building large-scale fault-tolerant systems inevitably involves trade-offs in terms of cost, performance, and development time.

As Web systems grow, those trade-offs loom larger and larger. "Fault-tolerant systems have always been difficult to build," says University of North Carolina at Chapel Hill computer science professor Mike Reiter. "Getting a fault-tolerant system to perform as well as a non-fault-tolerant one is a challenge."

Fortunately, the research community has been making major strides in this area of late, thanks in part to the contributions of ACM A.M. Turing Award winner Barbara Liskov of Massachusetts Institute of Technology, whose breakthrough work in applying Byzantine fault tolerance (BFT) methods to the Internet has helped point the way to cost-efficient fault tolerance for Web-based systems.

The Byzantine Generals Problem.



In the Byzantine Generals Problem, as defined by Leslie Lamport, Robert Shostak, and Marshall Pease in their 1982 paper, a general must communicate his order to attack or retreat to his lieutenants, but any number of participants, including the general, could be a traitor.

"THE BYZANTINE GENERALS PROBLEM" ACM TOPLAS VOL.4, ISSUE 3 (JULY 1982) DOI: 10.1145/357172.357176

While researchers have developed a number of different approaches to fault tolerance over the years, ultimately they all share a common strategy: redundancy. While hardware systems can employ redundancy at multiple levels, such as the central processing unit, memory, and firmware, fault-tolerant software design largely comes down to creating mechanisms for consistent data replication.

One of the most common approaches to software replication involves a method known as state machine replication. With state machine replication, any service provided by a computer can be described as a state machine, which accepts commands from other client machines that alter the state machine. By deploying a set of replica state machines with identical initial states, subsequent client commands can be processed by the replicas in a pre-determined fashion, so that all state machines eventually reach the same state. Thus, the failure of any one state machine can be masked by the surviving machines.

The origins of this approach to fault tolerance stretch back to the 1970s when researchers at SRI International began exploring the question of how to fly mission-critical aircraft using an assembly of computers. That work laid the foundation for contemporary approaches to fault tolerance by establishing the fundamental difference between timely systems, in which network transmission times are bounded and clocks are synchronized, and asynchronous systems, in which communication latencies have infinite-tail distribution (most messages arrive within a certain time limit but, with decreasingly low probability, messages may be delayed in transit beyond any bound).

The SRI work also helped draw important distinctions between the various types of faults experienced in a system, such as message omissions, machine crashes, or arbitrary faults due to software malfunction or other undetected data alterations. Finally, the SRI work helped to characterize resilience bounds, or how many machines are needed to tolerate certain failures.

The idea of state machine replication was given its first abstract formalization by Leslie Lamport and later surveyed by Fred Schneider. Lamport's work eventually led to the Paxos protocol, a descendant of which is now in use at Google and elsewhere. Lamport used the term "Byzantine" to describe the array of possible faults that could bedevil a system. The term derives from the Byzantine Generals Problem, a logic puzzle in which a group of generals must agree on a battle plan, even though one or more of the generals may be a traitor. The challenge is to develop an effective messaging system that will outsmart the traitors and ensure execution of the battle plan. The solution, in a nutshell, involves redundancy.

While Lamport's work has proved foundational in the subsequent development of Byzantine fault tolerance, the basic ideas behind state machine replication were also implemented in other early systems. In the early 1980s, Ken Birman pursued a related line of work known as Virtual Synchrony with the ISIS system. This approach establishes rules for replication that behave indistinguishably from a nonreplicated system running on a single, nonfaulty node. The ISIS approach eventually found its way into several other systems, including the CORBA fault-tolerance architecture.

At about the same time, Liskov developed viewstamped replication, a protocol designed to address benign failures, such as when a message gets lost but there's no malicious intent.

These pioneering efforts all laid the foundation for an approach to state machine replication that continues to underlie most contemporary work on fault tolerance. However, most of these projects involved relatively small, fixed clusters of machines. "In this environment you only had to worry that the machine you stored your data on might have crashed," Liskov recalls, "but it wasn't going to tell you lies."

Practical Byzantine fault tolerance provides a useful framework for developing fault-tolerant Web systems.

With the rise of the Internet in the mid-1990s, the problem of "lies"—or malicious hacks—rose to the fore. Whereas once state machines could trust each other's messages, they now had to support an additional layer of confirmation to allow for the possibility that one or more of the state machines might have been hacked.

Two groups of developers began exploring ways of applying state-machine replication techniques to cope with a growing range of Byzantine failures. Dahlia Malkhi and Mike Reiter introduced a data-centric approach known as the Byzantine quorum systems principle. In contrast to active-replication approaches like the Paxos protocol, Byzantine quorum systems focus on identifying a set of servers, rather than focusing on the messages, and choosing a set of servers so that they intersect in specific ways to ensure redundancy.

In the mid-1990s, Liskov started her breakthrough work on practical Byzantine fault tolerance (PBFT), an extension of her earlier work on viewstamped replication that adapted the Paxos replication protocol to cope with arbitrary failures. Liskov's approach demonstrated that Byzantine approaches could scale cost-effectively, sparking renewed interest in the systems research community.

While the foundational principles of consistency and replication remain essential, the rapid growth of Web systems is introducing important new challenges. Many researchers are finding that PBFT provides a useful framework for developing fault-tolerant Web systems. "I'm really excited about the recent work Barbara and her colleagues have done on making Byzantine Agreement into a practical tool—one that we can use even in large-scale settings," says Birman, a professor of computer science at Cornell University.

Inspired by Lamport and Liskov's foundational work, Hebrew University's Dolev has been working on an approach involving polynomial solutions to the general Byzantine agreement problem. While his early work in this area 25 years ago seemed largely theoretical, he is now finding practical applications for these approaches on the Web. "My theoretical work was ignited by Leslie [Lamport]," he says. "Barbara's work brought me to look again at the practicality of the solutions."

At Microsoft, researcher Rama Kotla has proposed a new BFT replication protocol known as Zyzzyva, that strives to improve performance by using a technique called speculation to achieve low performance overheads. Kotla is also exploring a complementary technique called high throughput BFT that exploits parallelism to improve the performance of a replicated application.

Also at Microsoft, director Chandu Thekkath has been pioneering an alternative approach to fault tolerance for Microsoft's Live Services, creating a single "configuration master" to coordinate recovery from machine failures across multiple data services. The concept of a configuration master also underlies the design of several other leading services in the live services market, such as Google's Chubby lock server.

Lorezo Alvisi, a professor of computer science at the University of Texas at Austin, and colleagues are probing the possibilities of applying game theory techniques to fault tolerance problems, while Ittai Abraham, a professor of computer science at The Hebrew University of Jerusalem, and colleagues are incorporating security methods into distributed protocols to punish rogue participants and deter against the deviation of any collusion among them.

While these efforts are opening new research frontiers, they remain squarely rooted in the pioneering work on Byzantine fault tolerance that started more than three decades ago. Indeed, many developers are just beginning to encounter this foundational research for the first time. "Engineers are starting to discover and use these algorithms instead of writing code by the seat of their pants," says Lamport.

Many developers still wrestle with the cost and performance trade-offs of fault tolerance, however, and a number of large sites still seem willing to accept a certain degree of system failure as a cost of doing business on the Web.

"The Web is going live," says Ken Birman. "This is going to change the picture for replication, creating a demand from average users."

"The reliability of a system increases with increasing number of tolerated failures," says Kotla, "but it also increases the cost of the system." He suggests that developers look for ways to balance costs against the need to achieve reliability in terms of mean time to failure, mean time to detect failures, and mean time to recover faulty replicas. "We need more research work in understanding and modeling faults in various settings to help system designers choose the right parameters," Kotla says.

Further complicating matters is the rise of mobile devices that are only sporadically connected to the Internet. As people entrust more and more of their personal data to these devices—like financial transactions, messaging, and other sensitive information—the challenge of keeping all that data in sync across multiple platforms will continue to escalate. And the problem of distributed fault tolerance will only grow more, well, Byzantine.

"The Web is going live," says Birman, who believes that the coming convergence of sensors, simulators, and mobile devices will drive the need for increasingly reliable data replication. "This is going to change the picture for replication, creating a demand from average users." When that happens, we may just see fault tolerance coming out of the clouds and back down to earth. C

Alex Wright is a writer and information architect who lives and works in New York City.

Cloud Computing

Cloning Smartphones

A pair of scientists at Intel Research Berkeley have developed CloneCloud, which creates an identical clone of an individual's smartphone that resides in a cloud-computing environment.

Created by Intel researchers Byung-Gon Chun and Petros Maniatis, CloneCloud uses a smartphone's Internet connection to communicate with the phone's online copy, which contains its data and applications, up to several gigabits in size, in the cloud. CloneCloud would make smartphones significantly faster and more powerful, enabling them to perform processor-heavy tasks in the cloud. For example, Chun and Maniatis's CloneCloud prototype, running on Google's Android mobile operating system, conducted a test application involving the facial recognition of photos. Running the application on the Android smartphone took 100 seconds; the phone's clone, operating on a desktop computer in the cloud, completed the task in one second.

According to the researchers, CloneCloud would also provide improved smartphone security, with virus scans of a device's entire file system being conducted in the cloud. Moreover, CloneCloud would improve a smartphone's battery life by having cloud-based computers handle the most processor-intensive tasks.

The CloneCloud research could help with intelligently allocating tasks to the most energy-efficient or fastest processor in a cloud-computing environment. "There will be a family of heterogeneous devices, and you would like to move the computing job to the one that makes most sense; from that standpoint, it is a great idea," said Allan Knies, associate director of Intel Research Berkeley, in an interview with *Technology Review*.

The CloneCloud approach could also help create a computing environment that would make it easier to share data between mobile devices and home-based computers.