■ Information about PIs

Names: Lorenzo Alvisi, Mike Dahlin, and Michael Walfish

Affiliation: The University of Texas at Austin

Email addresses: {lorenzo,dahlin,mwalfish}@cs.utexas.edu

Topics:

2. Data portability, consistency, and management
6. cloud security, privacy, and auditing

Abstract of prior cloud work: The PIs have built *Depot*, a cloud storage system that minimizes trust assumptions. Depot tolerates buggy or malicious behavior by *any number* of clients or servers, yet it provides safety and liveness guarantees to correct clients. Depot provides these guarantees using a two-layer architecture. First, Depot ensures that the updates observed by correct nodes are consistently ordered under Fork-Join-Causal consistency (FJC). FJC is a slight weakening of causal consistency that can be both safe and live despite faulty nodes. Second, Depot implements protocols that use this consistent ordering of updates to provide other desirable consistency, staleness, durability, and recovery properties. Our experimental evaluation suggests that the costs of these guarantees are modest and that Depot can tolerate faults and maintain good availability, latency, overhead, and staleness even when significant faults occur.

Abstract of future research problems

We want to remove trust *from* the cloud. This is not the same thing as making the cloud more trust*worthy*, which is about giving people reasons to assume the cloud's correct operation. In contrast, we would like for people not to *have* to make such assumptions. Ideally, they could treat providers as black boxes and get guarantees via local verification.

Context: the outsourcing trend and what it means for trust. Service providers (SPs) now offer storage, computation, and managed desktop services at costs far lower than in the traditional scenario, where autonomous entities host these functions themselves. At the same time, there are major impediments to outsourcing: customers worry about (1) the *integrity of data and computations*, (2) the *availability* of their data in the case of an SP going out of business, (3) the *privacy* of their data, and (4) *lock-in*. More generally, an SP is, to a client, *a black box controlled by another party*. Indeed, SPs can experience software bugs, correlated manufacturing defects, misconfigured servers and operator error, malicious insiders, bankruptcy, fires, and more. Thus, it seems prudent to design systems so that clients do not have to make strong assumptions about the SP's design, operation, and status. In fact, *minimizing trust* promises to help SPs too: today, a significant barrier to adopting so-called *cloud services* is that many organizations hesitate to place trust in the cloud.

Scope and approach. For each of three critical tasks—storage, function computation, and repair—we will demonstrate systems that give a client guarantees from a service provider without assuming the provider's correctness. Our immediate focus *integrity* guarantees, though we consider *availability* and *privacy*. (A comprehensive solution to privacy is critical and is part of our broader research agenda, along with the other issues above.) To tackle the specific thrusts below, we will compose methods from multiple research domains. Each thrust will be approached by careful design, rooted in models or theory, followed by implementing and experimentally evaluating practical systems and demonstration applications.

(1) Storage. We will build a storage system under minimized trust assumptions at acceptable cost. The expected outcome is a radical design point: even if all other clients or servers misbehave, a correct client gets a coherent view of shared data that incorporates both its own writes to the data, and those of the other reachable clients. Key innovations include (a) a protocol that forces every change to the data to name its antecedents, which (i) creates a well-defined ordering of changes to the data and (ii) limits misbehavior to forcing a partition in the system; (b) a method for correct clients to recover from such partitions, allowing them to keep operating in the face of attacks; and (c) leveraging client-side storage.

(2) Function computation. We aim to build a system in which a client specifies a computation to a server; the server executes it, and returns the purported output and some auxiliary information; the client uses the auxiliary information to verify that the output is correct; and the verification is far more efficient than simply carrying out the computation. We propose to adapt the theory of probabilistically checkable proofs (PCPs), in which one party constructs a proof, and the verifying party checks the proof by examining only tiny pieces. Our early estimates suggest that PCPs can be incorporated into a practical system. A key contribution, besides the built system, will be to challenge the folklore that PCPs are impractical.

(3) Repair. Today, computer repair resembles television repair: the customer brings the computer into the shop, or calls a technician (or family member) to request a visit. Yet, the overwhelming majority of computer repair issues involve only software. Moreover, computers are increasingly virtual machines—*which could be repaired anywhere*. Thus, our aim is to realize a novel vision: let a customer ship a computer into the cloud and get software-based problems fixed asynchronously. Two key problems are protecting the *integrity* and *privacy* of the customer's data from an untrusted repairer. Our approach will assume that the repairer determines the desktop image but that the customer runs this image on a virtual machine monitor (VMM) that the repairer does not control. We propose to address both problems by enforcement in the VMM, which will validate repairer-supplied logs, keep private data encrypted, and restrict information leakage.

Broader impacts. By making outsourcing safer, the proposed research will not only make customers better off but also spur the adoption of cloud services. This will mean more people paying less for computing, which will produce beneficial effects throughout the computational ecosystem.