Research Statement

Tianyin Xu
Department of Computer Science
University of Illinois Urbana-Champaign

My research focuses on building reliable computer systems that empower next-generation cloud and datacenter computing. Today, nearly all large-scale computing, no matter whether for supporting people's daily life or for enabling science and discovery, relies on massive, highly distributed, and rapidly evolving computer systems. Given prevalent hardware faults, software bugs, and human mistakes, together with inherent system complexity at scale (e.g., concurrency and asynchrony), the reliability of cloud-scale systems has been a grand challenge of the past decade. Emerging paradigms such as microservices, serverless computing, and hybrid cloud models further expand the reliability challenge by increasing the complexity, dynamics, and heterogeneity of these critical systems. I have been addressing open, fundamental problems to enable the design and implementation of reliable cloud-scale systems for emerging computing paradigms.

To ensure that my research is relevant and practical, I study why state-of-the-art computer systems fall short in delivering expected reliability, despite decades of research and development efforts, and think through the opportunities for innovation to make technical contributions. Together with my students and collaborators, I have made progress on the problems towards the overarching research goal. The results were published at computer systems research conferences and workshops (OSDI, SOSP, NSDI, EuroSys, HotOS); the interdisciplinary nature of the research also leads to publications at venues of other related research areas, including software engineering (ICSE, FSE, ISSTA, OOPSLA), computer architecture (ISCA, ASPLOS, MICRO, HPCA), computer networks (SIGCOMM, IMC, MobiSys, MobiCom), and computer security (CCS, NDSS). The research receives recognitions, including an NSF CAREER Award, an Intel Rising Star Faculty Award, a Facebook Distributed Systems Research Award, a Best Paper Award at ASPLOS'20, two SIGSOFT Distinguished Paper Awards, a CACM Research Highlight, a Gilles Muller Best Artifact Award at EuroSys'23, etc. Some of the research has led to practical impacts, including adoption by the Linux kernel and detection of severe software defects that affect critical systems such as Kubernetes, Hadoop, and Linux. The research projects are well funded by NSF and industry grants from VMware, IBM, Intel, Facebook, Azure, etc.

1 Cloud System Configuration and Operation

Correct operations are critical to cloud system reliability in production. Today, cloud services undertake frequent, continuous operations, e.g., hundreds to thousands of configuration changes deployed daily. With the high velocity of changes, faulty operations and misconfigurations inevitably have become major causes of catastrophic system failures and service outages. In fact, operations are increasingly being carried out by operation *programs* that automate tedious, human-based operations. My research thrives to effectively validate the safety of configuration changes and the correctness of operation programs.

Configuration Testing. I have been working on *configuration testing* with an exciting vision of continuously testing configuration changes in a rigorous way, like how software code is tested today. The key idea is to connect system configurations to software tests so that configuration changes can be tested in the context of code affected by the changes. We introduce a new type of tests, termed *Ctests*, and show that Ctests fundamentally outperform existing approaches by detecting configuration changes that trigger software bugs and subtle, sophisticated misconfigurations. We develop automated techniques to generate Ctests by transforming existing and abundant software tests in mature projects. Since we presented the first paper of Ctest at OSDI'20, we received generous support, including an NSF CAREER Award and a Facebook Distributed Systems Research Award to develop practical configuration testing techniques.

Since then, I have been worked on addressing practical issues of configuration testing, with the goal of making it fast, low-cost, and easy to adopt. We developed unified regression testing techniques for both configuration and source-code changes—unifying Ctests and existing software tests. The unified tests can significantly reduce maintenance and infrastructure costs. We also developed test-case prioritization for Ctests to speed up the detection of misconfigurations by reordering Ctest executions. Recently, I am working on leveraging Ctests to fuzz the configuration interface to detect configuration-related bugs. The research was published at ISSTA'21 and ICSE'23, and received a SIGSOFT Distinguished Paper Award.

End-to-End Testing for Operation Correctness. Going beyond configuration management, I have been working on ensuring operation correctness of cloud systems with end-to-end testing of operation programs (termed *operators*). In the modern operation pattern, an operation declares a desired system state and the operator automatically reconciles the system to that declared state. Operation correctness is challenging due to enormous system-state space and complex operation interface. A correct operator must satisfy correctness properties of its own code; it must also maintain managed systems in desired states. We developed the first automatic end-to-end testing technique, named Acto, for cloud system operators. Acto uses a state-centric approach to test an operator together with a managed system. It models operations as state transitions and systematically realizes transition sequences to exercise supported operations in various scenarios. Its oracles automatically check if a system's state is as desired. Acto has detected 56 serious bugs (42 confirmed and 30 fixed) in eleven Kubernetes operators. Acto was published at SOSP'23 and selected as a solution showcase at KubeCon EU. The team received a grant from IBM for open-sourcing Acto to the broader community.

I continue to work on testing and verifying operation correctness, focusing on addressing semantic gaps between operation logic and managed systems, and on runtime monitoring and recovery of misoperations.

2 Cloud Infrastructure Reliability and Fault Tolerance

Modern cloud infrastructures like Kubernetes, Borg, and Twine are architected as a fleet of loosely coupled microservices known as *controllers*. Each controller independently observes the current system state and continuously reconciles the system to the desired state. The system states are maintained in fault-tolerant data stores like ZooKeeper and etcd. The reliability of cloud infrastructures is highly challenging—controllers run within complex, dynamic, and distributed environments; they must safely drive the system to desired states, both individually and collectively, while tolerating unexpected failures, network interruptions, and asynchrony. If controllers are not robust, they lead to severe consequences such as service outages, data loss, and security issues. I have been working on building highly reliable cloud infrastructures by hardening existing infrastructure systems like Kubernetes and developing new ones with verified reliability.

Reliability Testing of Cloud Infrastructures. We developed the first automatic reliability testing technique for cluster-management controllers named Sieve. Sieve drives controllers to their potentially buggy corners by systematically and extensively perturbing the controller's view of the current cluster state in ways it is expected to tolerate. It then compares the cluster state's evolution with and without perturbations to detect safety and liveness issues. Sieve is highly usable. It does not require formal specifications of the controller or the infrastructures, hypotheses about vulnerable regions in the code, or specialized test inputs. It does not rely on expert-written assertions either. This degree of usability is key to making reliability testing broadly accessible to the rapidly increasing number of custom controllers. Sieve has detected 41 serious safety and liveness bugs (31 confirmed and 15 fixed) in ten popular controllers with a low false-positive rate of 2.3%. The work was published at OSDI'22 and was presented at KubeCon and CloudNativeCon.

Sieve is grounded on our work that reasons about the reliability of cloud infrastructures using a model called partial history. Partial histories effectively describe how each controller makes decisions based on observing a subset of changes to the world around them. We show that partial histories have immense explanatory power and utility over the state of the art. The work was published at HotOS'21.

I am currently working on building formally verified cluster-management controllers with guaranteed safety and liveness and thus free of the types of bugs found by Sieve. I am also exploring runtime verification techniques to monitor controller runtime behavior and recover from unexpected failures.

Cross-System Interaction Reliability. Today's cloud infrastructures are orchestrated by multiple interacting, interdependent systems, each of which provides important services (the controller-based architecture of Kubernetes is such an example). Therefore, cloud infrastructure reliability is constructed by not only the reliability of each individual system, but also the reliability of their interactions. In practice, we observe that failures of cloud infrastructures are often manifested through the interactions across systems. I am working on understanding and addressing cross-system interaction reliability. As a first step, we conducted the first analysis of cross-system interaction failures in modern cloud systems, leading to many insights on assuring and guaranteeing reliable interactions. The work was published at EuroSys'23 and was invited for presentation at SREcon24 Americas. I am continuing to work on testing and formal verification of cross-system interactions, supported by a Facebook Core Systems Faculty Gift grant.

Fault-Tolerant Infrastructures for Emerging Faults. I have been developing novel techniques to enable cloud systems to tolerate emerging non-crashing fault models including *fail-slow faults* and *silent data corruption*. We develop DepFast, an expressive programming framework for writing fault-tolerant distributed systems. DepFast offers a synchronous programming style, making the code easy to write and understand; meanwhile, it provides an event interface with a clean abstraction between the protocols and low-level utilities (e.g., RPC and disk I/O). DepFast was published at HotOS'21 and USENIX ATC'22. Recently, I started to investigate fault-tolerant designs for silent data corruption due to CPU defects.

3 Cloud Application Reliability

Even with reliable cloud infrastructure and cloud systems, application reliability is still a challenging problem. Today, more and more applications are emerging towards a cloud-based programming model where applications depend on cloud services for core functionalities. Such cloud-based practice greatly simplifies application deployment and realizes cloud benefits. However, it imposes emerging challenges for addressing fault models of opaque cloud backends and less predictable Internet connections. I am working on understanding and addressing cloud application reliability. We developed a taxonomy of new bugs that render cloud-based applications vulnerable to common transient faults. We show that (mis)handling transient error(s) of even one interaction of cloud API call can adversely affect application correctness.

We took a first step to address the reliability challenges by building a push-button reliability testing tool named Rainmaker, as a SDK utility for cloud-based applications. Rainmaker helps applications anticipate the myriad of errors under the cloud-based fault model—it encodes automatic fault injection policies to cover taxonomized bug patterns and automatic oracles that embrace existing in-house software tests. Rainmaker has detected 73 bugs (55 confirmed and 51 fixed) in eleven popular Azure and AWS based applications. The work was published at NSDI'23. I continue to work on cloud-based application testing with a focus on developing high-fidelity local testing environment to support efficient, continuous application testing.

4 Operating Systems and Virtualization

In addition to cloud reliability research, I have been working on the foundations of the cloud and datacenter computing stack, including the operating system (OS) and virtualization technologies.

OS Kernel Extensibility. Kernel extensibility is critical to cloud system operation. Kernel extensions such as BPF extensions are increasingly used for security, tracing, as well as network and storage optimizations. I have been working on improving the performance and expressiveness of kernel extensions. We developed a novel cache system named Draco to accelerate security checks for system calls (implemented as BPF extensions). Draco was published at MICRO'20 and part of its design is adopted by Linux. We also developed cross-optimizations of OS kernel extensions (KFuse, EuroSys'22) and Seccomp-eBPF support (LPC'23). Recently, I have been working on rethinking the expressiveness of kernel extensions and on developing a Rust-based kernel extension framework, with the preliminary results published at HotOS'23.

I also worked on minimizing the OS kernel code to reduce the attack surface and memory footprint. We developed a tool called Cozart to automatically reconfigure the OS kernel based on the application demands. The work was published at SIGMETRICS 2020 and selected as a CACM Research Highlight.

Memory Management. I have been working on addressing a fundamental bottleneck of memory-centric computing in virtualized clouds—virtual memory translation. We developed Elastic Cuckoo Page Table (ECPT), a novel page table design that transforms the sequential pointer-chasing operation in traditional x86 radix page tables into parallel lookups. We designed both the native and the nested ECPT for container and virtual machine based cloud infrastructures. The results were published at ASPLOS'20 and ASPLOS'22, respectively. The research received a Best Paper Award at ASPLOS'20, and an honorable mention of IEEE Micro Top Picks 2021. With the architectural design in place, I have been working on the OS support. Our insight is that hardware-based memory translation performance can be significantly improved by designing novel page-table allocation in the OS. The insights led to a new design, termed Direct Memory Translation (DMT), which can achieve ECPT performance (and outperform it in virtualized environments) while being backward compatible with x86. The work will appear at ASPLOS'24. Recently, I am working on designing an extensible OS interface to accommodate heterogeneous memory translation hardware architectures (e.g., x86-based radix tree, ECPT, and DMT) with the goal of effecitvely supporting them in modern OSes.