Research Statement

Zu-Ming Jiang
Department of Computer Science, ETH Zurich

Overview

My **research mission** is to *realize reliable and secure systems* for modern computer science. Toward this mission, I focus my research on **systems** and **security**. System software, including databases and operating systems, have been universally deployed and become the de facto basis of our lives. The reliability and security of system software are critical to ensure the robustness of modern computing environments. However, ensuring their reliability and security is challenging because of the inherent characteristics of system software: large code scale, complex implementations, and diverse functionalities. To tackle this challenge, I have developed novel techniques and provided theoretical guarantees for automatically testing and validating system software. My research appeared in the top-tier venues from the systems (OSDI) and security (USENIX Security and NDSS) communities. My work also significantly improved the reliability and security of widely-used systems, especially database management systems (*e.g.*, MySQL, Postgres, and SQLite), and was recognized and adopted by the industry (*e.g.*, ClickHouse Inc., PingCap, OceanBase).

The **research philosophy** behind my work centers around *granularity shifting*. Abstraction is at the center of computer science,[1] while the granularity we work at determines the abstraction we can perceive. Specified to system reliability, an apposite granularity to view the systems enables us to discover and focus on the essence of their reliability and security problems. By shifting granularity (*e.g.*, finer-grained construction for system inputs), we can explore the deeper-level principles behind the complex system mechanisms (*e.g.*, complex execution scheduling) and design elegant and practical solutions. During my Ph.D. research, I applied my philosophy on database management systems (DBMSs), a critical and ubiquitous kind of foundational system software, whose reliability and security govern the usability and correctness of the diverse data-driven applications (*e.g.*, bank systems). A reliable and secure DBMS should consistently and correctly process both expected and unexpected user inputs. In other words, a central mission of DBMS reliability is to ensure the consistency and correctness of system execution under arbitrary inputs. Motivated by this mission, I tackled three fundamental problems of database systems: SQL generation, transaction validation, and query checking.

Ongoing and Past Research

Empowering Input Generation

Vulnerabilities in DBMSs can cause critical security impacts like data leakage and system paralysis. Detecting DBMS vulnerabilities is challenging, especially the vulnerabilities in the deep DBMS code related sophisticated query optimization and execution. Covering the deep DBMS code requires the query inputs to be (1) valid to be able to pass the syntactic and semantic checks and (2) complex to have the potential to be optimized or accelerated. The problem remained unsolved due to the conflicts between query complexity and validity (*e.g.*, complex generated queries are more difficult to be guaranteed valid).

My research tackles this problem by *shifting the granularity of generation*, *i.e.,* from query generation to statement generation. Queries consist of sequences of statements, and each statement can reference attributes defined in the former statements. The essence of generating complex and valid queries is to build complicated and correct references among statements. Based on this perspective, I developed *Dynamic Query Interaction* (DQI) [1], which decouples query generation into multiple separated statement generation. DQI leverages database schema, a general data structure maintained by DBMSs, to perceive the state changes caused by each statement. In this way, each statement is guaranteed to reference the latest and accurate database attributes, and thus DQI can always construct correct references in complex statements, which in the end compose complex and valid queries. By deploying DQI on popular DBMSs like MySQL and SQLite, I have found **40 bugs** in these systems, **19** of which have been recognized as impactful vulnerabilities and assigned **CVE IDs**. This work was well appreciated and published in a top-tier security venue, **USENIX Security 2023**.

Validating Transaction Consistency

Database transactions are widely adopted in critical domains like banking and transportation to ensure the robustness of

[1] Barbara Liskov. "The Power of Abstraction." Turing Award Lecture, 2008.

applications. While transactions are critical, no prior approach was capable of validating the consistency and correctness of interleaved transactions that contain predicates or complex operations; two key challenges exist in the transaction support of DBMSs: (1) the **diversity of isolation levels**, and (2) the **complexity of transaction semantics**. These two challenges make it extremely difficult to infer the expected results of transactions.

My research tackles this problem by *shifting the granularity of dependencies*, *i.e.,* from transaction dependencies to statement dependencies. Isolation levels are specified to regulate the allowed combination of transaction dependencies, but transaction dependencies are too coarse-grained and cannot precisely describe transaction semantics, which is determined by how statements inside transactions interleave and how these statements operate data. To model the underlying principles of isolation levels and extract the essence of transaction semantics, I introduce a new concept, *statement dependency*, with seven formally defined subclasses. I also prove a dependency theorem [2], which illuminates a new direction for transaction validation: reorder the statements inside the transactions in a dependency-preserving manner, and check whether the original transactions produce the same results as the reordered test cases. The theorem is independent of the isolation levels DBMSs use and properly corresponds to transaction semantics. To identify statement dependencies in the transaction executions, I propose SQL-level instrumentation, which is proven to be complete for identifying all potential statement dependencies. Based on the theoretical guidance and guarantees, I implemented TxCheck [2], which has found 56 bugs in three popular DBMSs, and the paper based on this work was published and presented at **OSDI 2023**.

Checking Query Correctness

DBMSs can suffer from logic bugs, breaking their soundness and causing the systems to produce incorrect query results. Detecting logic bugs remains challenging due to the complex semantics a query statement may have (*e.g.*, involving correlated subqueries, and multiple joins), which impedes existing approaches from inferring the expected query results. Before my work, no prior approach could generally check the result correctness of an **arbitrary** query statement.

My research tackles this problem by *shifting the granularity of transformation*, *i.e.,* from statement transformation to expression transformation. A typical methodology to validate the correctness of a query statement is to transform it into another statement and check the result consistency (*e.g.*, equivalence) between these two. However, such a methodology cannot be generally adopted because it needs to analyze the query semantics, which can be intractable when query statements become complex. The key contribution of my research is to eliminate such intractability by focusing on the finer-grained units of a query statement, namely expressions. An expression can refer to a function, operation, column variable, constant, *etc.* Versus a statement, whose semantics can be highly complex, expressions are more succinct, universal, and localized. Focusing on these finer-grained units enables us to transform a query statement by transforming its expressions, thus preserving the overall semantics of the query without needing to understand them. In the end, the approach validates query result correctness by checking whether the query statement still produces the same results after its expressions are transformed. I realized this idea as the EET tool [3]. EET has found 66 bugs in five popular DBMSs, some of which were long latency (*e.g.*, one bug in PostgreSQL had existed for over 20 years). EET was also well appreciated by the industry receiving highly positive testimonials from DBMS developers and will be integrated into the CI of well-known DBMSs (*e.g.*, ClickHouse). The research was published and presented at **OSDI 2024**.

Future Research

By shifting the granularity of my focus, I envision future research where I can make significant contributions. Specifically, shifting the granularity regarding system scales, I am especially motivated to research the reliability problems of distributed systems, which are universally deployed in large production environments. Shifting the granularity regarding system designs, I am driven to understand and tackle the difficult problems of emerging systems in data science, which share similar characteristics with traditional data-intensive systems (*e.g.*, DBMSs) but have unique implementations and designs (*e.g.*, new architectures). Shifting the granularity regarding system research, I started by validating parts of system code (*e.g.*, via testing) and plan to move toward fully verified systems whose correctness is guaranteed by advanced system foundations.

Reliability of Distributed Systems

Modern computing services, such as cloud platforms, demand highly reliable distributed systems. Compared to traditional system software, distributed systems are larger and more complex due to the necessity of distributed event handling and data

synchronization. Distributed systems have various designs and implementations, including distributed databases (RethinkDB, AerospikeDB), distributed filesystems (HDFS, GlusterFS), and distributed coordination systems (ZooKeeper, etc.). Because of these inherent characteristics, realizing reliable and secure distributed systems that can correctly process arbitrary system inputs is extremely difficult. Specifically, two key challenges stand out: (1) various types of system inputs that work for different distributed systems, and (2) no standard test oracles for validating the correctness of different system outputs.

Granularity shifting can shed light on this research direction. I plan to research the inputs of distributed systems from three finer-grained aspects to extract their principles: (1) external events like client requests and control commands, (2) internal events like network congestion, and node crashes [4, 5], and (3) event concurrency [6] (*e.g.*, time intervals between events). To validate the correctness of system outputs, I plan to explore two potential and promising directions: (1) theoretically proving the valid scope of system outputs by exhaustively enumerating the finer-grained effects of each system event, and (2) inferring expected outputs by constructing semantically equivalent execution scenarios where the finer-grained units (*e.g.*, events or event intervals) are adjusted.

Trustworthy Data Science

Due to the importance of data in our modern society, more and more innovative and revolutionary techniques are proposed to help significantly evolve data science. For example, new kinds of DBMSs (*e.g.*, graph DBMSs [7], spatial DBMSs) have been realized to efficiently manage data whose structures follow specific designs. Quantum databases are emerging and may revolutionize data storage. These databases leverage quantum mechanics, such as superposition and entanglement, to process information in fundamentally different but potentially much more efficient ways. In addition, the rapid development of AI, *e.g.*, large language models (LLMs), intensifies the demand for accurate and high-performance data retrieval. In this setting, how to define and ensure the reliability and security of these new systems remains an outstanding scientific and technical challenge.

Granularity shifting can guide the exploration of these emerging domains. To identify the reliability problems of these new systems, we can look at the finer-grained and critical features where new systems have unique designs or implementations. These features can inform which parts of the systems are critical and what types of reliability problems to focus on. For example, one key feature of quantum databases is how they use quantum mechanics to store data. Therefore, to improve the reliability of such systems, we should research how to effectively cover the system logic of their storage engine and validate the correctness of corresponding system behaviors.

Foundation of Verified Systems

Testing can validate only parts of system logic covered by their generated test cases, while verification at the system level can provide comprehensive guarantees for the entire system. However, establishing fully verified systems is extremely challenging because (1) the scales of systems are large (*e.g.*, millions of lines of code) and verification techniques are known to be difficult to scale to large, real-world systems; and (2) the semantics and properties of system behaviors are complex to model.

The philosophy of granularity shifting can inspire and guide how we explore and tackle the above challenges. Specifically, to realize a fully verified system, we can separately verify the finer-grained parts of the system, *i.e.*, each system component. In this way, we can dramatically reduce the scale and complexity of the systems to be verified. Formally, given a system S with components C_0, C_1, …, C_n, verifying a system can be decomposed to verifying each system component:

$$V(S) = V(C_0) \wedge V(C_1) \wedge \ldots V(C_i) \wedge \ldots V(C_n)$$

By focusing on a single system component, we can extract more concise and accurate semantics that concern only the behaviors of the target component. Furthermore, based on the extracted semantics, we can identify the properties that collaboratively guarantee the correctness of component implementations. Formally, given a system component C and a set of properties P_0, P_1, …, P_m, we can further apply granularity shifting on component verification by separately verifying each property of the target component, as described in the following:

$$V(C) = V(C, P_0) \wedge V(C, P_1) \wedge \ldots V(C, P_i) \wedge \ldots V(C, P_m)$$

Such a methodology, which we refer to as gradual verification, significantly facilitates system verification by reducing the complexity of both the target systems and the required properties. By individually verifying each property of each system component and gradually combining the verified proofs, ultimately, we can realize a fully verified system whose correctness is guaranteed by a series of finer-grained verification steps.

Reference

[1] **Zu-Ming Jiang**, Jia-Ju Bai and Zhendong Su, "DynSQL: Stateful Fuzzing for Database Management Systems with Complex and Valid SQL Query Generation," in Proceedings of the 32nd USENIX Security Symposium (Security'23), Anaheim, 2023.

[2] **Zu-Ming Jiang**, Si Liu, Manuel Rigger and Zhendong Su, "Detecting Transactional Bugs in Database Engines via Graph-Based Oracle Construction," in Proceeding of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI' 23), Boston, 2023.

[3] **Zu-Ming Jiang** and Zhendong Su, "Detecting Logic Bugs in Database Engines via Equivalent Expression Transformation," in Proceeding of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI' 24), Santa Clara, 2024.

[4] **Zu-Ming Jiang**, Jia-Ju Bai, Kangjie Lu and Shi-Min Hu, "Fuzzing Error Handling Code using Context-Sensitive Software Fault Injection," in Proceedings of the 29th USENIX Security Symposium (Security'20), Boston, 2020.

[5] **Zu-Ming Jiang**, Jia-Ju Bai, Julia Lawalla and Shi-Min Hu, "Fuzzing Error Handling Code in Device Drivers Based on Software Fault Injection," in Proceedings of the 30th International Symposium on Software Reliability Engineering (ISSRE'19), Berlin, 2019.

[6] **Zu-Ming Jiang**, Jia-Ju Bai, Kangjie Lu, Shi-Min Hu, "Context-Sensitive and Directional Concurrency Fuzzing for Data-Race Detection," in Proceedings of the 29th Network and Distributed System Security Symposium (NDSS'22), San Diego, 2022.

[7] Dominic Wüst*, **Zu-Ming Jiang***, Zhendong Su (*: Equal Contributions), "Dinkel: Testing Graph Database Engines via State-Aware Query Generation", in arXiv, 2024.