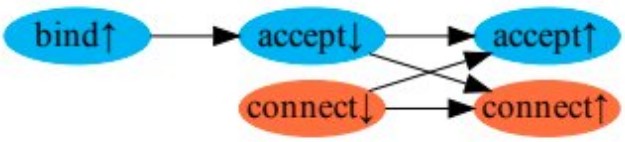
Box of Pain: A new tracer and fault injector for distributed systems

3 May 2019, by Ingrid Fadelli



The happens-before relationship of accept and connect system calls that Box of Pain derives. The colors indicate different threads. Box of Pain is able to derive that connect? occurs after accept?, because the latter causes the former. Credit: Bittman, Miller & Alvaro.

In computer science, distributed systems are systems with components located on different devices, which communicate with one another. While these systems have become increasingly common, they are typically filled with bugs.

A few researchers have tried to develop tools to find bugs in distributed systems and remove them, yet a tangible solution has not yet emerged. Overall, tools that 'perturb' executions can test how robust a system is to faults, while tools that 'observe' executions allow researchers to better understand the system-wide effects of such faults.

Most existing approaches and techniques for fault detection and debugging are incomplete or based on testing, which means that they might be helpful to find bugs but not to eliminate them. Aware of this gap in the literature, a team of researchers at UC Santa Cruz has recently developed a new technique, called Box of Pain, for tracing and fault injection in unmodified distributed systems.

"Our lab is obsessed with fault tolerance," Peter Alvaro, one of the researchers who carried out the study, told TechXplore. "Distributed systems, that is, systems that require the cooperation of a collection of independent computers to fulfil their purpose, are ubiquitous, but they are extremely hard to reason about, program and debug. Fault injection techniques can increase confidence that distributed systems can actually tolerate the faults (e.g. machine crashes, network partitions, etc.) that they were designed to tolerate, while observability infrastructure (e.g. tracing) can help us better understand how these systems function during faults."

Alvaro's lab primarily focuses on an area of research called experiment selection, which entails automatically choosing faults that are most likely to drive a system into a bad state. He and his colleagues use tracing infrastructure to observe executions, build models of systems that produce traces and then use these models to identify 'interesting' faults to inject into a system.

"Unfortunately, our approach assumes that systems are already equipped with both tracing and fault injection infrastructure; to first observe, and ultimately perturb the system execution," Alvaro said. "In practice, many systems are not, and adding these capabilities can be expensive and time consuming. We found ourselves asking what it would take to be able to perform tracing and fault injection transparently, on unmodified systems, so that we could apply our bug-finding approaches to 'any' distributed software. Box of Pain is where we eventually landed."

Box of Pain, the approach devised by the researchers at UC Santa Cruz, is essentially a framework for tracing a complex computer system to better understand its behavior, simulate faults within it and observe what happens when something goes wrong. For instance, Box of Pain can simulate a broken network and compare the behavior of the programs with their behavior under normal conditions.

"Our technique does this by observing key events

in the programs' behaviors, such as communication events, crashes, and exit conditions," Daniel Bittman, another researcher involved in the study, explained. "Using this information, it live-builds an understanding of how computers interact, allowing automated bug-finding software to experiment with disrupting systems automatically."

In contrast with other fault injection systems, Box of Pain employs a lightweight approach to tracing, focusing on simulating the effects of partial failures on communication rather than exploring the failures themselves. In their study, the researchers evaluated their technique and found that it attained highly promising results, both in observing faults and perturbing distributed systems.

"One important finding was what we were able to do with our somewhat limited view of a complex computer system," Bittman said. "Since our goal was to understand the behavior of a system transparently (that is, without needing to make changes to the system under study), the information we collect about it is pretty generic."

According to Bittman, a key first step in their research was to show that they could successfully reconstruct the communication pattern of a complex system just by looking at the individual events of each process and that this could be done in real-time. This is crucial because the researchers wanted their model of fault injection to allow them to tell a system: 'drop all communication between program A and B after B sends a message to A'. If they were unable to reconstruct a system's communication pattern until after it finished running, however, this phrase would be impossible to convey.

"A second important finding was the number of ways in which a particular system execution could differ while achieving the same result," Bittman added. "Several interacting computers might run at different speeds, and therefore the way they interact might be different between runs of the same system with the same inputs, even if the result of the execution is the same. This has an unfortunate consequence: deciding when to inject a fault in a system becomes much harder. However, we were able to provide some initial evidence that the problem, in practice, is not as bad as it might seem."

The results gathered by Alvaro, Bittman and their colleague Ethan Miller have substantial implications for fault injection, as their approach could make deciding on and performing fault injection experiments far easier. In addition, their study could inform the development of debugging frameworks, which would report to developers with what confidence level their system is bug-free under particular circumstances.

"This research has only just begun," Alvaro said. "in fact, as we readily admit in the paper, we have barely even begun using Box of Pain for its stated purpose of finding and isolating bugs in distributed systems. We published this early report because we were excited to tell the community about the development."

According to Alvaro, there are two key directions in which their research could be developed further in the near future. Firstly, although their study provides tantalizing initial evidence supporting their hypotheses, future studies might need to run more experimental tests to further evaluate their assumptions.

"We argue that a distributed fault injector need only focus on perturbing edges in a system's communication graph to find the most interesting bugs, massively shrinking the 'surface area' on which we need to focus," Alvaro explained. "We now need to show that this is true by finding some new bugs! What is more, we argue that although the space of 'possible' executions is exponentially large and intractable to cover, the likelihood of different executions (at the level of abstraction we capture in the communication graph) drops off very steeply, making it it possible to mostly cover this space efficiently."

To show that the effect they observed is true and can be generalized across different scenarios, the researchers will need to scale up their experiments to larger and richer systems. In the long-term, they also envision a tight integration of Box of Pain with a targeted experiment selector, such as lineage-driven fault injection, as this might help to

generalize this selector to arbitrary distributed infrastructures.

"Over the next six months our lab plans to experiment on data stores such as Cassandra, Redis and MongoDB, on message queues such as Kafka and RabbitMQ, and on coordination services such as EtcD and Zookeeper," Alvaro added. "We also plan to explore pedagogical applications of Box of Pain, choosing custom fault injection schedules for project submitted by students in UC Santa Cruz's distributed systems course. This way, it can assist instructors in grading student projects as well as assist students by providing rich explanations of any bugs that it identifies in their programs."

The study carried out by Alvaro, Bittman and Miller was pre-published on *arXiv* and has been accepted for publication by HotCloud 2019, a workshop on cloud computing that will take place in July in Renton, Washington. This workshop will be a great chance to solicit feedback about Box of Pain from the distributed systems community, which could help the researchers to determine which avenues for future work they should pursue first.

More information: Co-evolving tracing and fault injection with Box of Pain. arXiv:1903.12226 [cs.DC]. arxiv.org/abs/1903.12226

APA citation: Box of Pain: A new tracer and fault injector for distributed systems (2019, May 3) retrieved 16 October 2021 from https://techxplore.com/news/2019-05-pain-tracer-fault-injector.html