Checking Linearizability / Sequential Consistency of Distributed Database Systems

CS4405 - Analysis of Concurrent and Distributed Programs

February, 2022

Project description

In this project, you will test a distributed database of your choice and check whether its executions satisfy linearizability or sequential consistency properties. [1]

Background Information

Testing distributed systems

The executions of distributed systems involve complex interactions of its concurrent components, asynchrony in the exchanges messages, possible network failures (e.g., delaying or dropping messages, isolating some nodes, partitioning the cluster of nodes) and component failures (e.g., node crashes). Software testing provides a practical method to exercise the system's behavior and check whether it satisfies its guarantees in the existency of asynchrony and failures.

In this project, we will generate test cases that consist of test harnesses (sets of client requests) and their executions under various network and component failures.

Distributed database systems

Distributed database systems distribute a database across multiple nodes in a cluster for increasing scalability and fault tolerance. They replicate the data on several different nodes that improves performance (the nodes/replicas can concurrently serve to client requests) and provides fault-tolerance (if a node/replica is not available, the other nodes/replicas can continue serving the clients). The systems are designed to operate on the replicated data objects concurrently and provide some guarantees on the consistency of the replicated data.

Linearizability and Sequential Consistency

Linearizability [1] and *Sequential Consistency* [2] are standard correctness properties for concurrent data structures and distributed databases. Database systems with linearizability or sequential consistency guarantees provide their clients an illusion of working on a single copy of the data objects.

Linearizability requires that there exists a total order of client operations where (i) the total order respects the real time order of the non-concurrent operations and (ii) the behavior of the operations are consistent with the sequential specification of the data structure.

Sequential Consistency relaxes linearizability in its requirement of real time order of the non-concurrent operations. It requires that operations appear to take place in some total order, and that that order is consistent with the order of operations on each individual process.

We will cover the topic in detail in the lectures of week 6.

System under test for your project:

You can test a distributed database system of your choice, such as Cassandra, CockroachDB, etcd, MongoDB, NebulaGraph, Redis, Riak, YugaByte DB.

[1] The project description is subject to small changes and updates. Please contact the TA's and the teachers if you have any questions.

Roadmap for the project:

The project involves the following steps:

- Set up a cluster of distributed system nodes
- Generate test harnesses, i.e., concurrent client requests to submit to the system
- Run the test harnesses in the existence of asynchrony, network failures, and process failures (e.g., the executions with randomly delayed or dropped network messages, isolated processes, crashing processes)
  - You can generate your own test harnesses and inject faults into executions, or you can use an existing test generation tool, e.g. Jepsen [3].
- Collect the execution histories
- Check linearizability or sequential consistency of the collected histories
  - You can implement a program for checking linearizability/SC of execution histories [4, 5] or use an existing checker, e.g. Knossos [6], Elle [7].

References

[1] Maurice P Herlihy and Jeannette M Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12(3):463–492, 1990.

[2] Leslie Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Transactions on Computers c-28*, 9:690–691, 1979.

[3] Kyle Kingsbury. Jepsen. `http://jepsen.io/`.

[4] Jeannette M. Wing and Chun Gong. Testing and verifying concurrent objects. *J. Parallel Distributed Comput.*, 17(1-2):164–182, 1993.

[5] Gavin Lowe. Testing for linearizability. *Concurrency and Computation: Practice and Experience*, 29(4):e3928, 2017.

[6] Kyle Kingsbury. Knossos. `https://github.com/jepsen-io/knossos/`.

[7] Peter Alvaro and Kyle Kingsbury. Elle: Inferring isolation anomalies from experimental observations. *Proc. VLDB Endow.*, 14(3):268–280, 2020.