Unit and Fuzz Testing in Evaluating MongoDB Reliability

Youngjae Moon, Lisa Liu, Robert Sheng

VANDERBILT UNIVERSITY

Contents

Project Overview

- This project aims to simplify the deployment and evaluation of MongoDB by:
  - Automating the setup of MongoDB clusters using Docker and Kubernetes.
  - Providing a suite of unit tests to validate basic functionality of MongoDB interactions.
  - Integrating fuzz testing, a type of automated software testing, through Google Atheris to identify potential vulnerabilities.

Features

- Automated MongoDB Setup
  - Easily deploy a MongoDB cluster on Chameleon Cloud using Docker and Kubernetes.
- Unit Testing
  - Validate core MongoDB operations such as CRUD operations and replica set configurations.
- Fuzz Testing
  - Stress-test the MongoDB client or server with random or invalid inputs to identify crashes and vulnerabilities.
- Kubernetes StatefulSet
  - Deploy and manage MongoDB clusters on Kubernetes with persistent storage and replica sets.

VANDERBILT UNIVERSITY

Tools and Technologies

- Docker
  - For containerization of MongoDB.
- Kubernetes
  - For orchestrating MongoDB clusters.
- Python 3.12.6
  - For running benchmark and test scripts.
- Google Atheris
  - For fuzz testing.

VANDERBILT UNIVERSITY

VM Distribution

- This project employes Docker and Kubernetes across a four-VM setup to automate the deployment, configuration, & testing of MongoDB clusters.
- Each VM is designated for specific roles to maximize efficiency, scalability, and reliability.

VM Roles

- VM1: Control and Orchestration Node
  - Master node in Kubernetes.
  - Deploys all pods.
- VM3: Primary MongoDB Replica Set
  - Runs Kubernetes pod that hosts the primary MongoDB replica set.
- VM4: Secondary MongoDB Replica Set + MongoDB Client
  - Runs Kubernetes pod that hosts the secondary MongoDB replica set.
  - Runs Kubernetes pod that runs a Python MongoDB client to interact with the MongoDB pods.

VM Roles

- This VM distribution maximizes the project’s flexibility by leveraging Docker for containerized MongoDB deployments, and Kubernetes for high-availability and clustered management.
- This setup facilitates comprehensive testing of MongoDB across different environments, ensuring reliability, scalability, and performance.

Test Metrics – Functional

- CRUD Operation Metrics
  - Insert Latency (ms): Measure the time taken to insert a single document or batch of documents.
  - Query Latency (ms): Evaluate the time taken to retrieve documents based on primary key lookup, range queries, compound key lookups, or text search queries (if indexes are enabled).
  - Update Latency (ms): Measure the time to update one or more fields in existing documents.
  - Delete Latency (ms): Measure the time to delete documents based on specific criteria.
- Replica Set Metrics
  - Replication Lag (ms): Time difference between when a write is applied to the primary node and when it is replicated to secondaries.

VANDERBILT UNIVERSITY

Test Metrics – Performance

- Throughput
  - Operations per Second (ops/s): Measure the number of CRUD operations MongoDB can process per second under a specific workload.
- Resource Utilisation
  - CPU Utilisation (%): Measure CPU usage under various workloads.
  - Memory Utilisation (%): Track memory usage during read and write-intensive operations.
  - Disk I/O (MB/s): Monitor disk reads/writes during heavy data insertions or queries.

Test Metrics – Reliability

- Durability
  - Data Loss on Failure: Measure the extent of data loss (if any) under sudden node or cluster failures.

Test Metrics – For Fuzz Testing

- Resilience
  - Crash Rate (% of operations): Percentage of operations causing crashes during fuzz testing.
- Vulnerability Metrics
  - Edge Case Coverage: Number and variety of edge cases detected during fuzz testing.
  - Execution Paths Tested: Percentage of code paths executed during fuzz testing.

Test Metrics – Benchmark

- Load Testing
  - Sustained Performance: Measure system performance over an extended period (e.g., 1-hour test).

Team Member Roles

- Robert Sheng
  - Writing codes in VM1, VM3 and VM4 for MongoDB setup through Docker and Kubernetes.
  - Generating various test metrics for evaluation.
- Lisa Liu
  - Writing codes in VM1, VM3 and VM4 for MongoDB setup through Docker and Kubernetes.
  - Generating various test metrics for evaluation.
- Youngjae Moon
  - Writing test codes in VM1
  - Making up slides and final report

Future Work

- CI/CD Integration
  - Automate deployment and testing pipeline using GitLab or GitHub Actions.
- Develop more sophisticated test metrics.
  - And achieve high coverage.

- Note: Demo will be done through recording a video.

Conclusion

- This project automates MongoDB deployment and testing, enhancing reliability and reducing setup time.
- By integrating unit and fuzz testing, it ensures MongoDB is robust under various conditions.

For more information…

https://github.com/Pingumaniac/Unit_and_Fuzz_Testing_in_Evaluating_MongoDB_Reliability

Thank you!

Any questions?