15th International Workshop DAMSS 2024
November 28 – November 30, 2024, Druskininkai

VILNIUS TECH
Vilnius Gediminas Technical University

Increasing Availability of Low Latency Stateful Microservices

Kęstutis Pakrijauskas, Dalius Mažeika
Vilnius Gediminas Technical University, Saulėtekio al. 11 Vilnius, Lithuania

Summary. Releasing new software versions, performing maintenance and activities against a highly available stateful microservice requires a failover between its nodes. Connection pooling allows to reuse established connections towards a service reducing the latency. On the other hands there is an even greater impact on availability as client connections are kept alive. The impact on availability of the stateful microservice can be reduced by observing the state of client connections and terminating them based on their status. The proposed method of graceful failover to redirect database requests from one node to another with near-zero impact on its client by forcefully terminating idle client connections. A set of experiments was performed to evaluate the proposed method. Results have shown that near-zero downtime can be achieved during a graceful failover minimizing the impact on availability budget of a stateful microservice.

INTRODUCTION

Reused connections allows to reduce the latency towards a database. An established connection remains open and waiting for incoming queries. Thus, without explicitly checking its status on either the database or the client side, it is unclear if a query is being executed or not. A database proxy, sitting between a client and a database cluster, can re-direct new connections to other nodes of the cluster. In an event of a node shutdown, a terminated idle connection, given a proper retry mechanism is configured on its client, is re-established towards another database node. Evey client connection on the database node slated for shutdown is examined and, once idle, terminated with negligible impact on the client.

DESCRIPTION OF GRACEFUL FAILOVER METHOD

Transition of Client Connections

- At event *e1*, *db-node-1* is marked as being shut down, but other action is taken against it.
- At *e2*, database proxy cordons the *db-node-1* disallowing new connections towards it.
- At *e3* the *db-node-1* begins terminating incoming client connections.
- The *pooled-connection-n* connection to enters ‘sleeping’ state. This state of the connection is then detected by the *db-node-1*, and at *e4* it is terminated.
- At *e5, pooled-connection-1* is reestablished, the database proxy directs it to *db-node-2*.
- At *e6,* once all connections have been transitioned to *db-node-2*, *db-node-1* is marked as ready to shutdown.

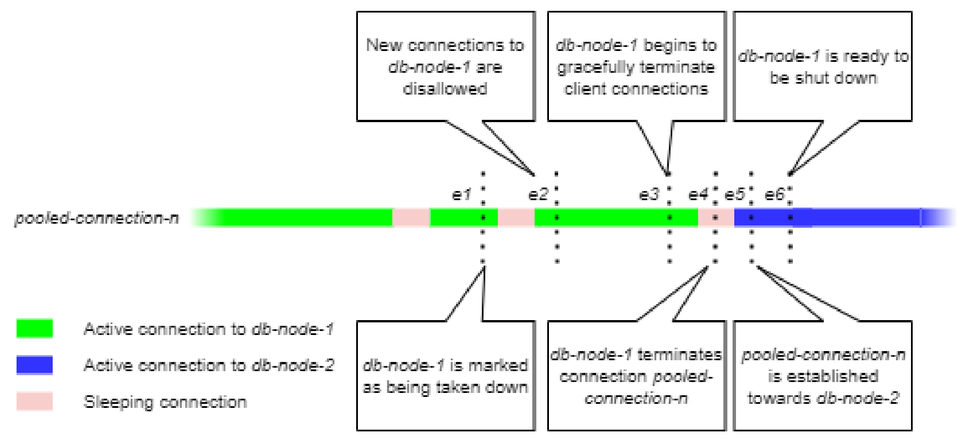

Fig. 1 Client connection transition

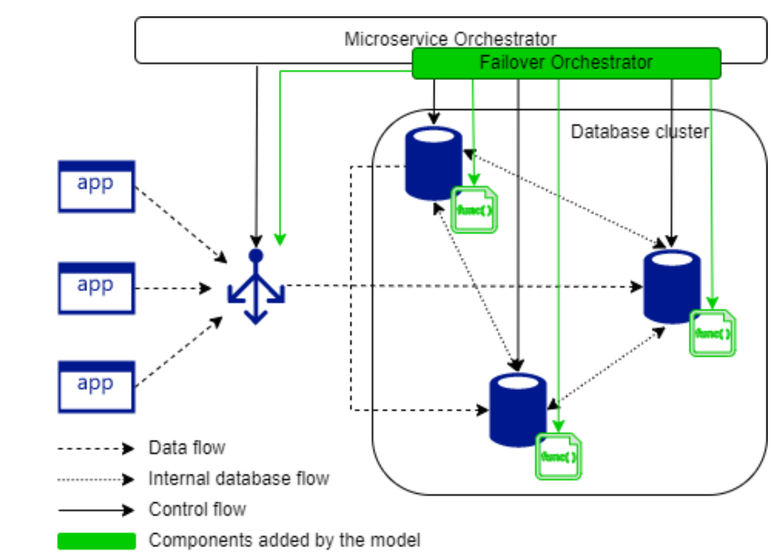

Fig. 2 Components of the proposed method

Components of the Proposed Method

- Database cluster setup for multi-Primary replication
- Proxy layer that can direct client requests to a specific node
- A client connection termination mechanism, say a stored procedure of a function
- A connection pool to enable low latency database connections
- A retry mechanism on the client side to handle terminated connections
- An orchestrator to oversee the failover.

Multi-Primary replication would allow distributed database cluster to continue serv-ing client requests while a graceful failover is happening.

A proxy layer, whether an appliance or a service, would allow to balance requests between nodes while switchover is performed.

The orchestrator issues commands to database nodes and database proxy during the entire managed failover process

RESULTS OF THE EXPERIMENT

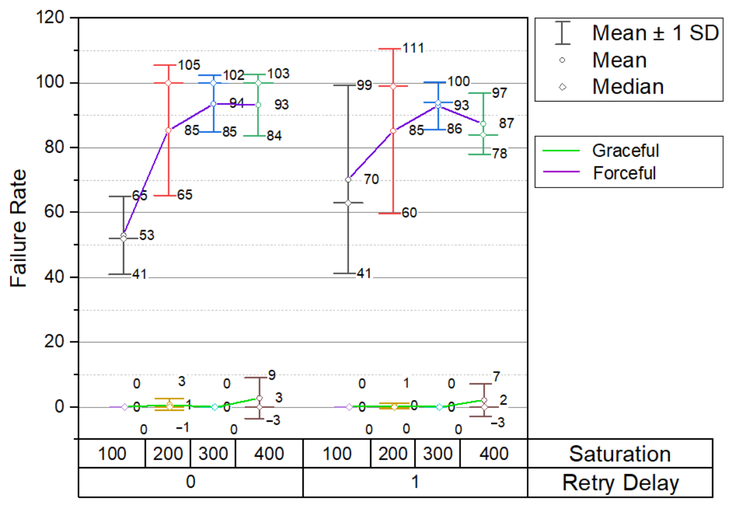

Fig. 3 Dependency of failure rate on failover method, saturation and retry delay

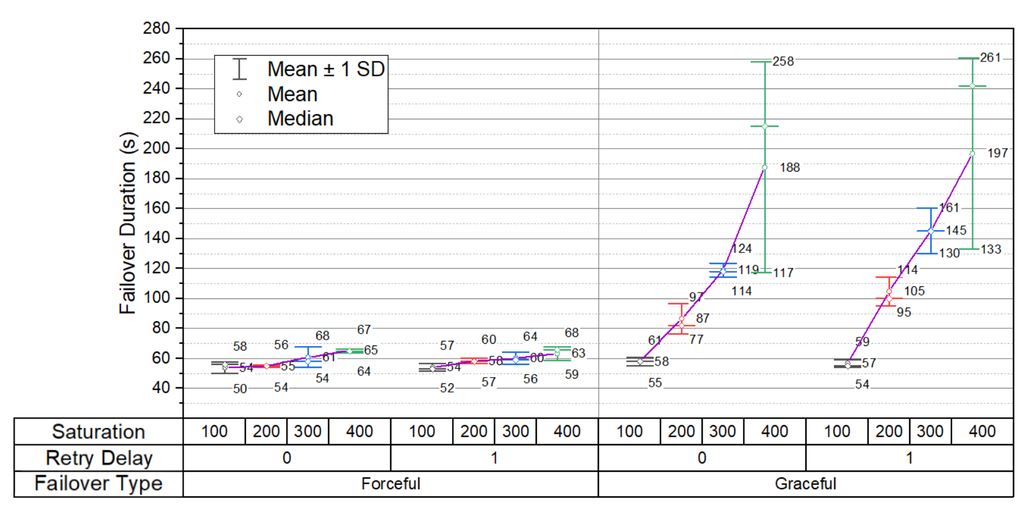

Fig. 4 Dependency of failover duration on failover method, saturation and retry delay

The proposed method of graceful failover allows to reduce the mean failure rate to near zero. Although failure rate is still affected by saturation, the impact is insignificant compared to forceful failover.
Failover duration increases along with saturation, and the process is slowed down by the necessity to inspect the high number of connections

REFERENCES

1. Pakrijauskas K., Mažeika D. A Method of Transparent Graceful Failover in Low Latency Stateful Microservices. *Electronics* **2022**, *11(23)*