URL Shortener Project Presentation

Cloud-Based Data Processing (IN2386)

Prof. Dr. Jana Giceva

Michalis Georgoulakis

Philipp Fent

Group 86

Turker Koc

Ege Kocabas

Technologies

The following technologies are used in the URL-Shortener implementation.

- C++ → Coding language
- gRPC → Node communication
- SQLite → Persistent storage

Services & Remote Procedure Calls

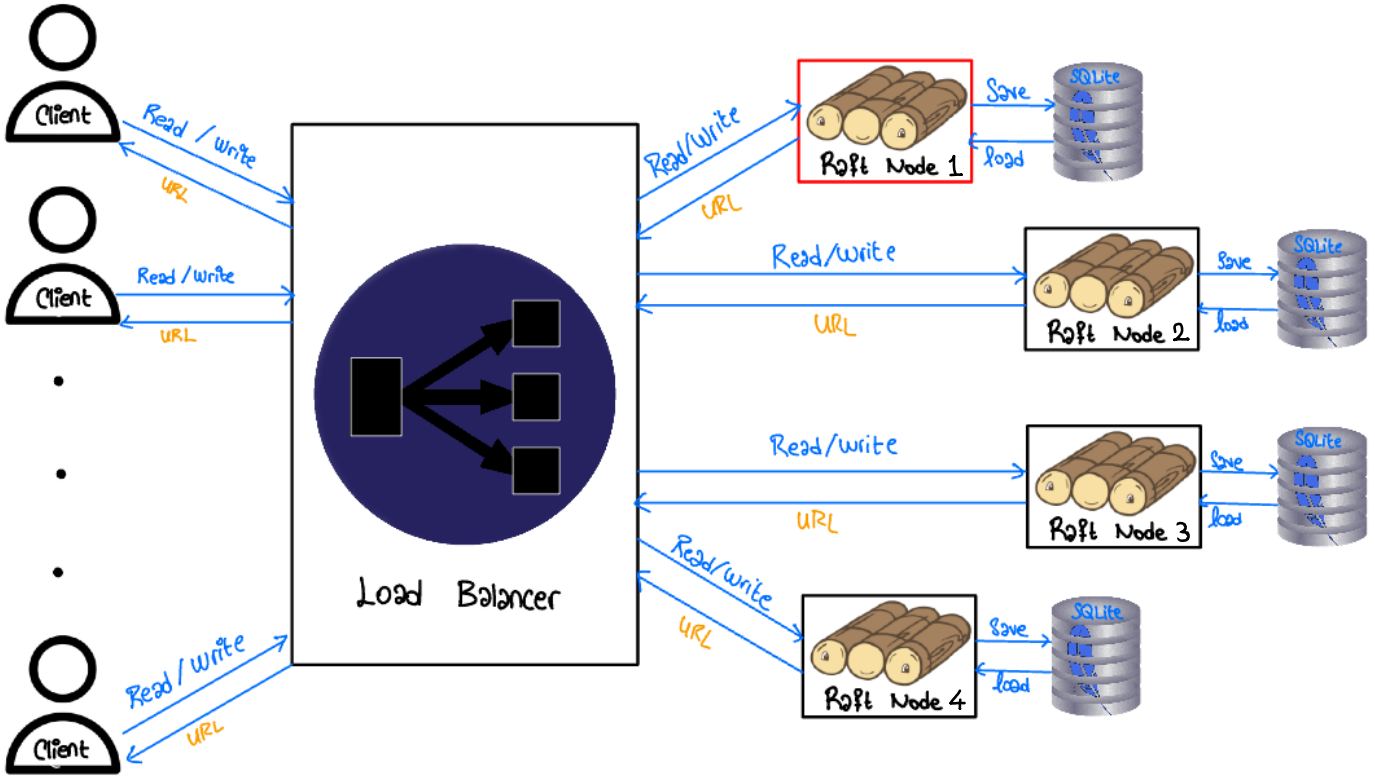
- Inter-node communication

```
service RaftService {
  rpc AddLog (AddLogRequest) returns (AddLogResponse);
  rpc RequestVote (RequestVoteRequest) returns (RequestVoteResponse);
}
```

- Inter-node & Client-node communication

```
service ClientService {
  rpc Write (WriteRequest) returns (WriteResponse);
  rpc Read (ReadRequest) returns (ReadResponse);
}
```

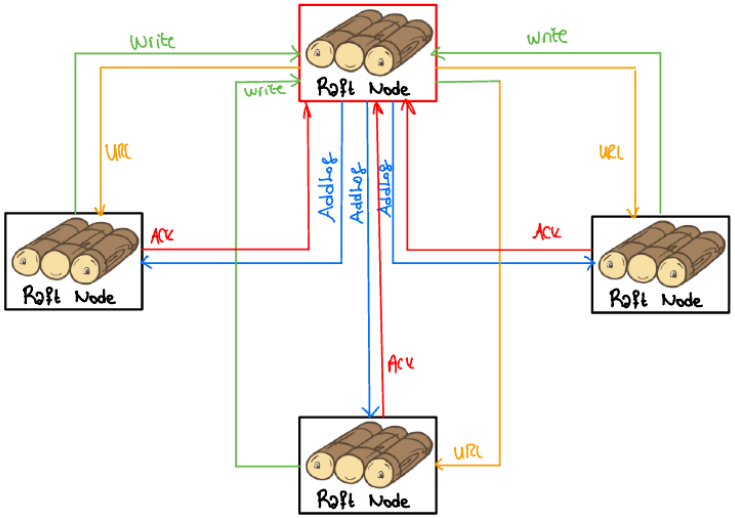
High-level design

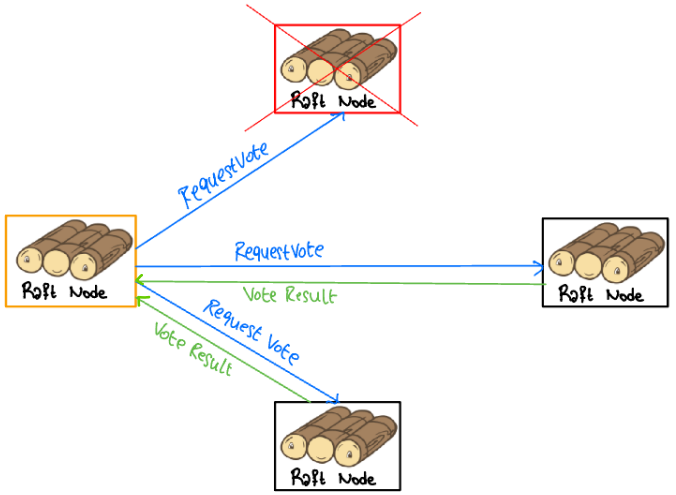


Components

- Client
- Load Balancer
- Raft Nodes
- SQLite

High-level design (cont'd)





Test

- testCorrectness

spawnCluster()

sendExampleWriteRequests()
sendReadRequests()

cleanup()

Test (cont'd)

- testResilience

spawnCluster()

sendExampleWriteRequests()
sendReadRequests()

rebootCluster()

sendReadRequests()
sendExampleWriteRequests2()
sendReadRequests2()

cleanup()

Test (cont'd)

- testWorkload

```
# https://db.in.tum.de/teaching/ws2223/clouddataprocessing/data/clickbench.00.csv
downloadUrlDataset()
spawnCluster()

sendWriteRequests() -> 35-45 ms
sendReadRequests()  -> 30-35 ms

cleanup()
```

Conclusion

- Implemented Raft using C++, gRPC and SQLite.
- Created a load balancer.
- Tested our system.

Future Work:

- Can implement improved Raft Algorithm.
- Config file for IP, Ports and timeouts.