A Generic High-Availability Solution to Next-Generation Mobile Core Networks

Yi Chen, Chien Chen, Je-Wei Chang, Jyh-Cheng Chen

National Yang Ming Chiao Tung University (Taiwan)

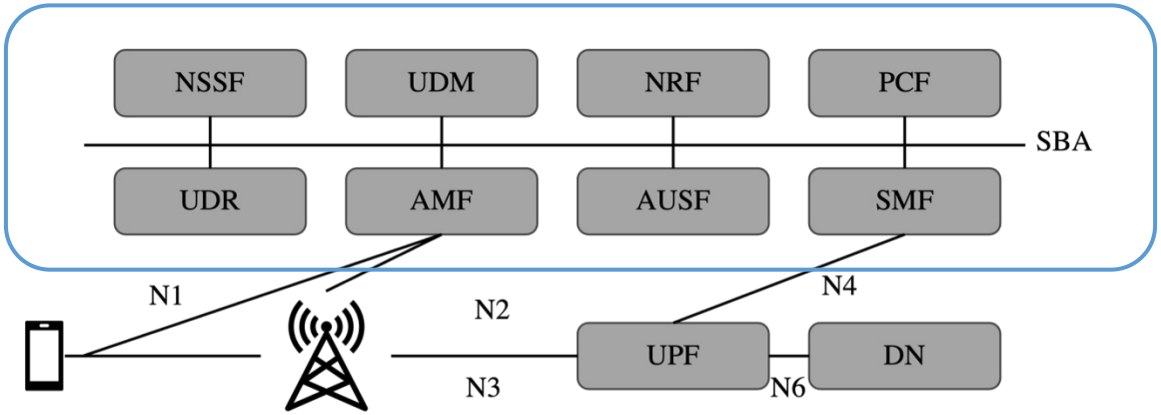
Outline

- Introduction
- Background
- Related Works
- Challenges
- Proposed Solution
- Implementation
- Evaluation
- Conclusion
- Future Works

Introduction

- We not only care about functionality for commercial 5G Core networks but availability is also important.
- The stateful service needs to store the user-related contexts in a permanent storage solution to achieve high availability.
  - ➢ It's a burden to Small and Medium Enterprise.
- Thus, we need a solution to enhance the network service's availability with the smallest cost.

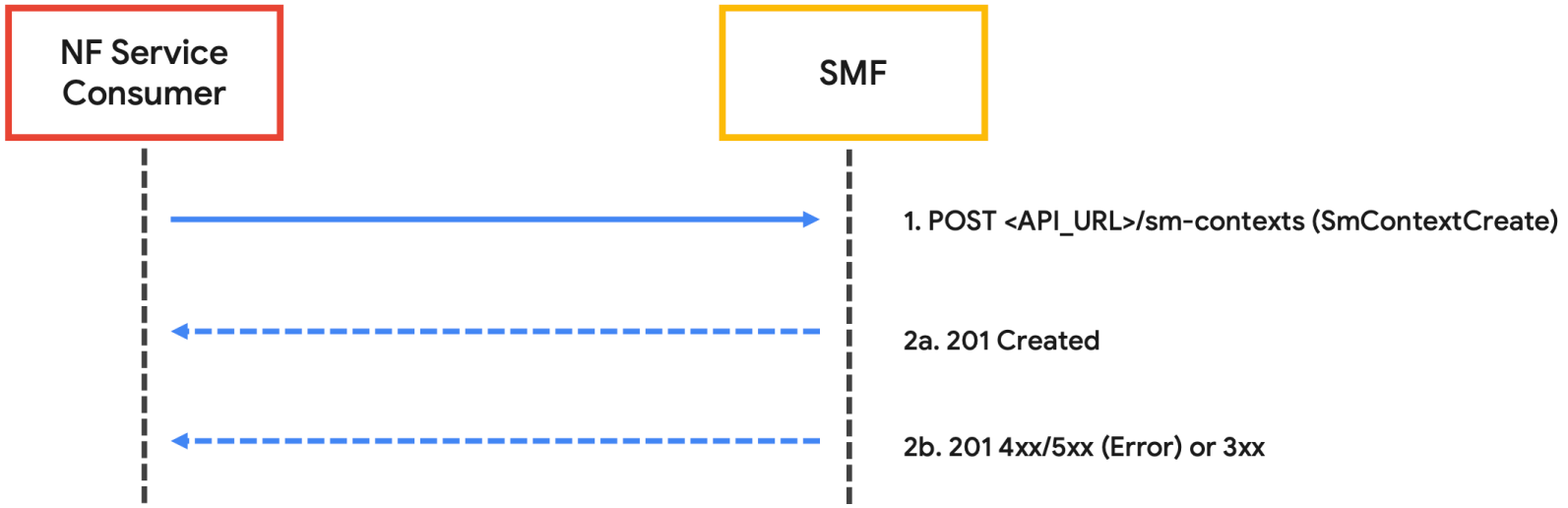
Background – 5GC overview



- All of the network functions (except for UPF) use a Service-Based Interface to communicate with each other.
- 5G Core leverages the open interface and the service discovery mechanism, it brings the possibility for cloud deployment.

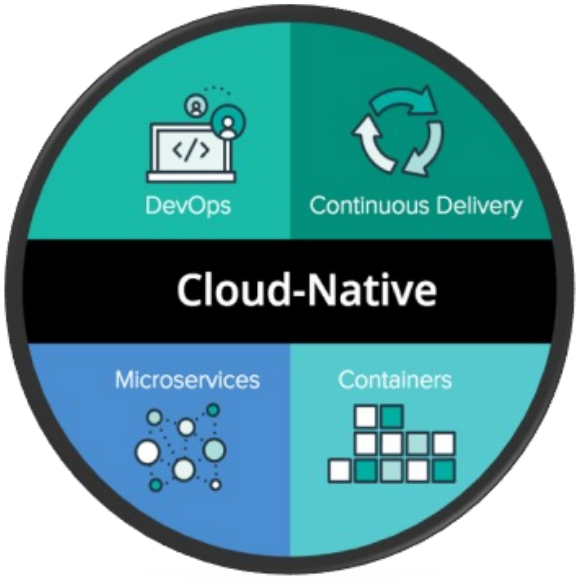
Background – Service Based Interface

SBI processing flow:



Background – Cloud-Native

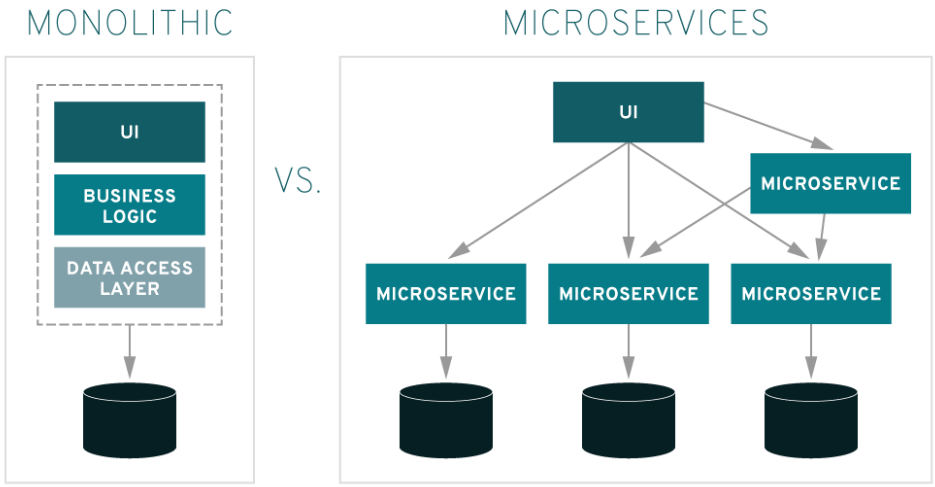
Cloud-native enables scalable app development and deployment in modern environments like public, private, and hybrid clouds.



Ref: https://medium.com/velotio-perspectives/cloud-native-applications-the-why-the-what-the-how-9b2d31897496

Background - Microservice

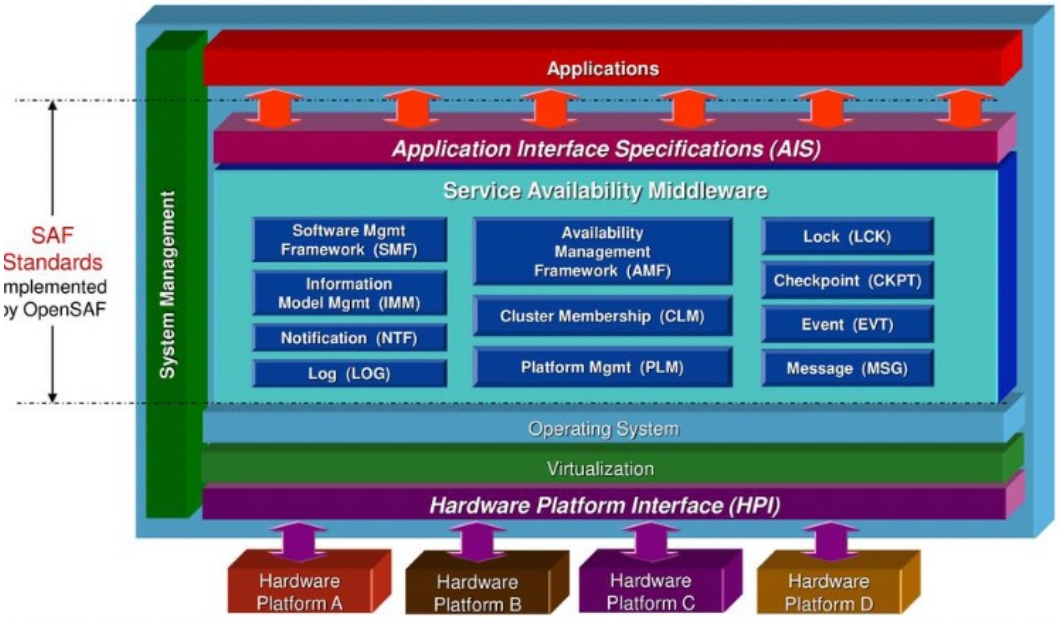
- An application is built as independent components that run each application process as a service.
- These services communicate via a pre-defined interface using lightweight APIs.
- Stateless service is better than stateful service.



Ref: https://medium.com/velotio-perspectives/cloud-native-applications-the-why-the-what-the-how-9b2d31897496

Background - OpenSAF

- OpenSAF is an open-source project, which provides multiple service availability middleware across the full range of availability and manageability requirements.

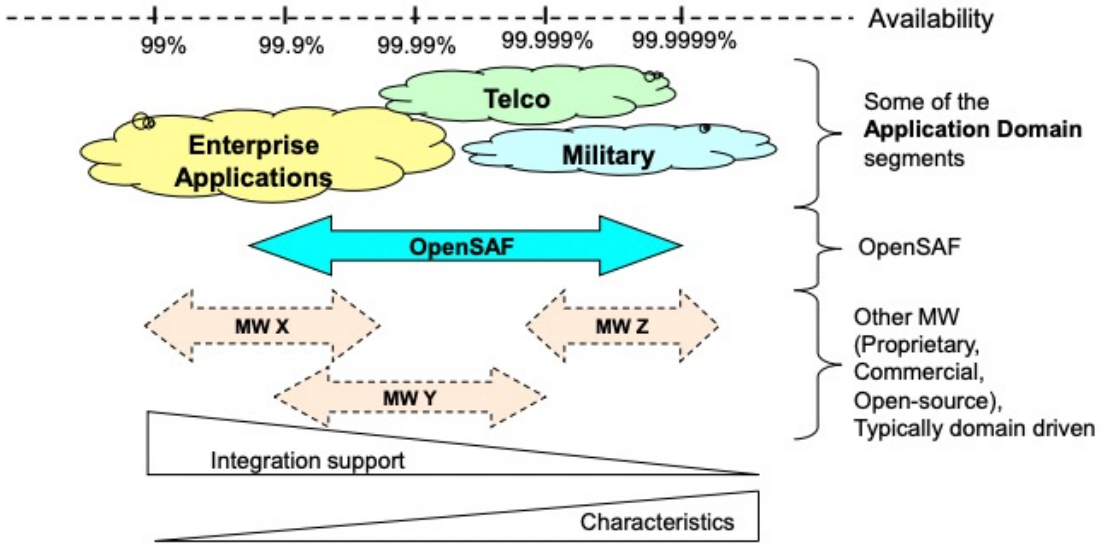


Ref: https://slideplayer.com/slide/12910555/

Background - HA

- HA (High-Availability) can be evaluated by the formula below:

$$A = \frac{\mathrm{MTBF}}{\mathrm{MTBF} + \mathrm{MTTR}}$$



Ref: http://events17.linuxfoundation.org/sites/events/files/slides/OpenSAF%20for%20Carrier%20Grade%20HA_0.pdf

Related Work

Compare with existing implementations

Project	HA metrics	Open source	Note
OpenSAF	99.9999%	v	1. Only provides C/C++ libraries. 2. Only can store the char array in the checkpoint
GO-CPSV	99.9999%	v	1. Support Golang 2. Support multiple storage solution 3. Enhance the concurrency
MongoDB	99.99% up with Replica Set	x	1. Need another workforce for ensuring the reliability

Related Work

Compare with existing implementations

Project	HA supported?	Supported NFs	Open source	Note
free5GC	x	x	v	
SD Core	v	AMF & SMF	v	Based on free5GC v3.0.5
Open5GS	x	x	v	
Stateless5g	v	AMF, SMF, and UPF (SM Context only)	v	Based on Open5GS
EN5GC	v	UDR	v	Based on free5GC (compose) and GO-CPSV

Related Work

For existing software projects, development team needs to face:

- Diverse deployments
- Support HA for legacy code
- Service observability

Challenges

For existing software projects, development team needs to face:

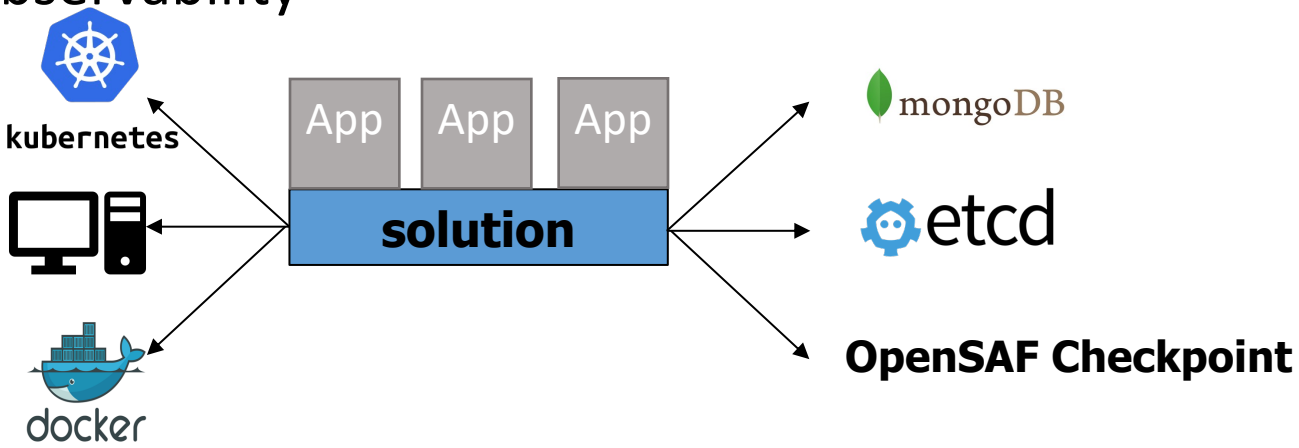
- Diverse deployments
- Support HA for legacy code
- Service observability

Challenges - 1

Diverse deployments

For existing software projects, development team needs to face:

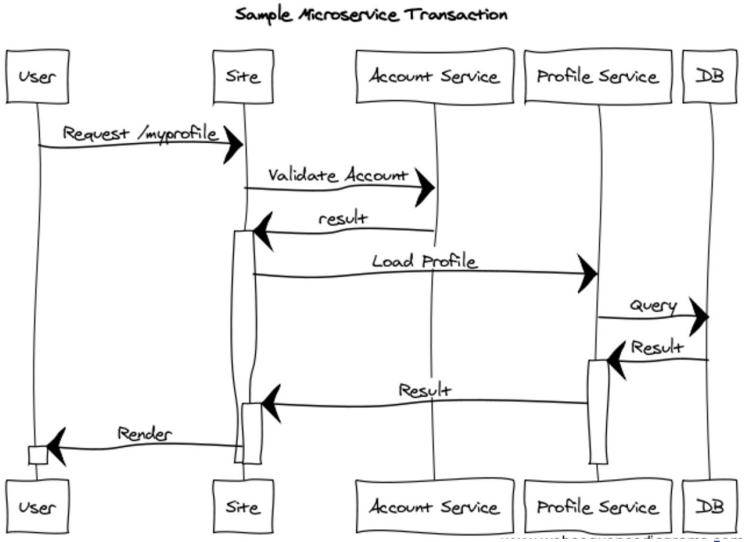
- Diverse deployments
- Support HA for legacy code
- Service observability



Challenges - 2

Support HA for legacy code

- Some of network functions are stateful network service.
- For different use cases, service has its special backup mechanism (usually procedural or transactional update).

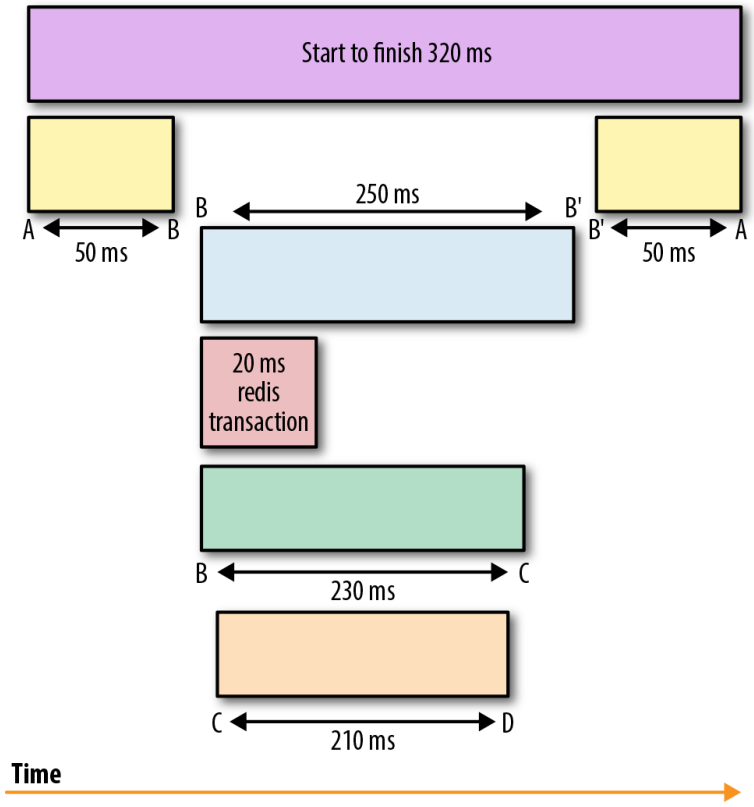


Ref: https://tanzu.vmware.com/developer/guides/distributed-tracing/

Challenges - 3

Service observability

- Tracing
- Logging
- Metrics



Ref: https://www.oreilly.com/library/view/distributed-systems-observability/

PROPOSED SOLUTIONS

GO-CPSV: Context backup solution for Golang

HA-UDR: Enhanced UDR based on GO-CPSV

GO-CPSV

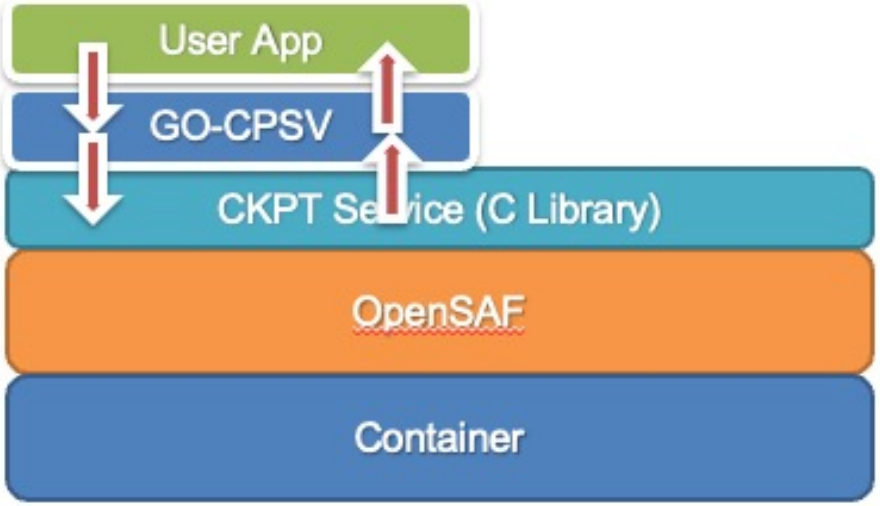
GOlang CheckPointSerVice

- Context backup solution
- Reach 99.9999% HA metrics (leverage from the OpenSAF checkpoint)
- Lightweight package (Golang)
- Cloud Native based service
- Support to expand storage solution
- less migration efforts
- High affinity to service observability

Cloud-Native based service

Solved: Diverse deployments

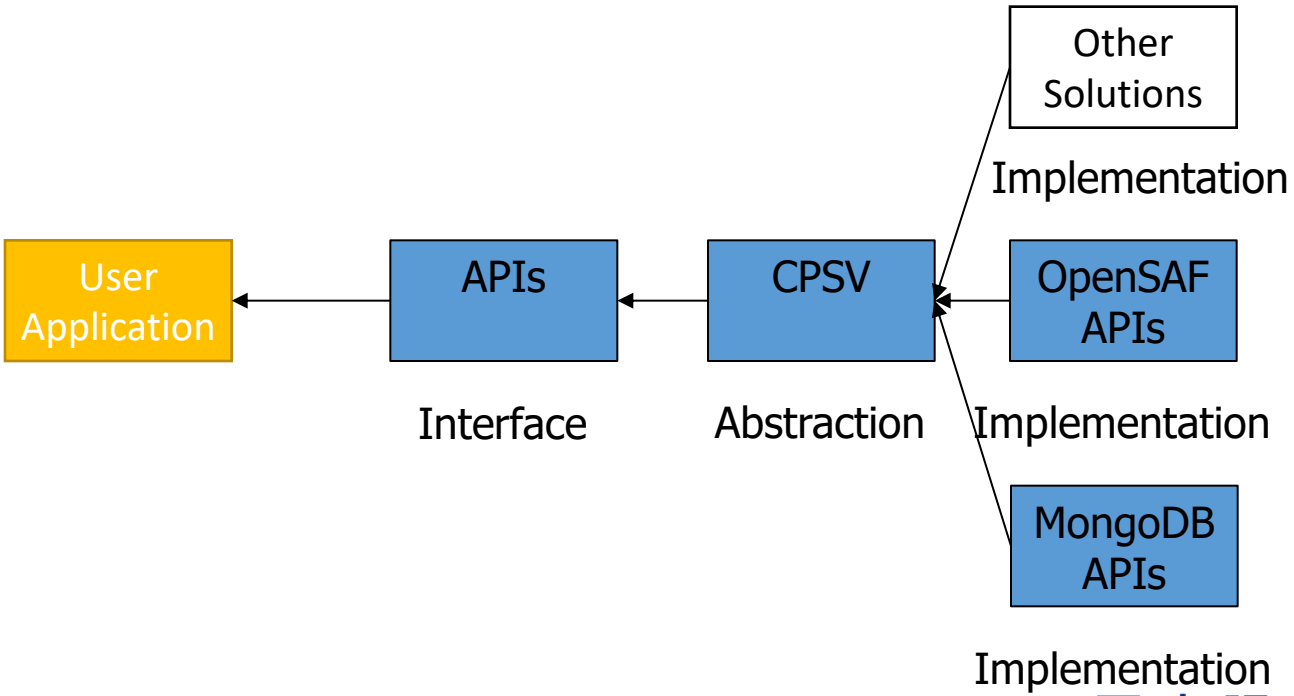
- Familiar with Cloud-Native base application:
  - GO-CPSV provides official container image
  - GO-CPSV has been proven being able to execute on docker compose and kubernetes



Expand storage solutions

Solved: Diverse deployments

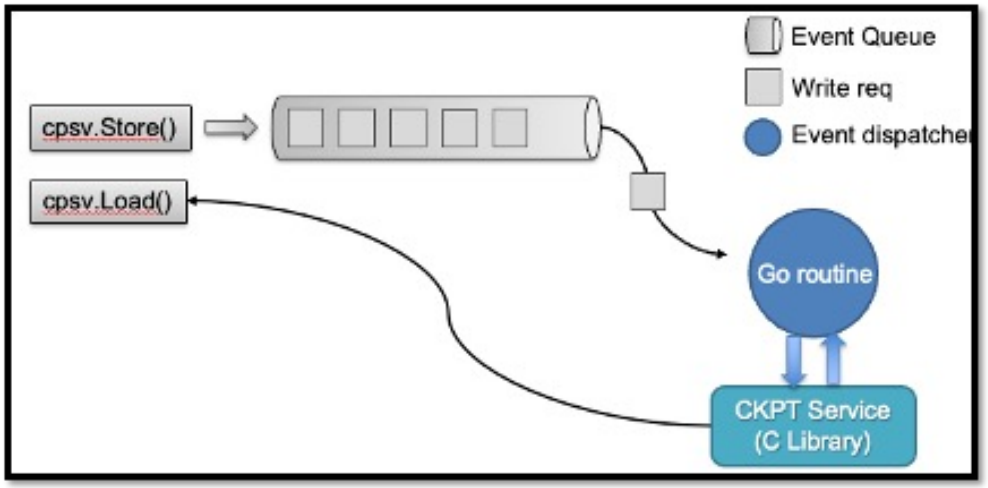
- The GO-CPSV follows Dependency Inversion Principle:



Less migration efforts

Solved: Support HA for legacy code

- Developers doesn't need to understand how the OpenSAF works.
- They only need to care about when the application need to write/read the state to/from checkpoint.



High affinity to service observability

Solved: Service Observability

GO-CPSV support lifecycle hooks feature

- Developer can inject pre-defined hook functions into GO-CPSV.
- These functions will be invoked when:
  - BeforeOperation
  - OperationSuccess
  - OperationFailed
- It enabling high affinity for supporting logging/tracing/metrics collection.

HA-UDR

Enable UDR to support high availability

HA-UDR was modified by the free5GC project with GO-CPSV:

- PoC for GO-CPSV
- Must running on OpenSAF cluster environment
- Backup the Context when request received from SBI

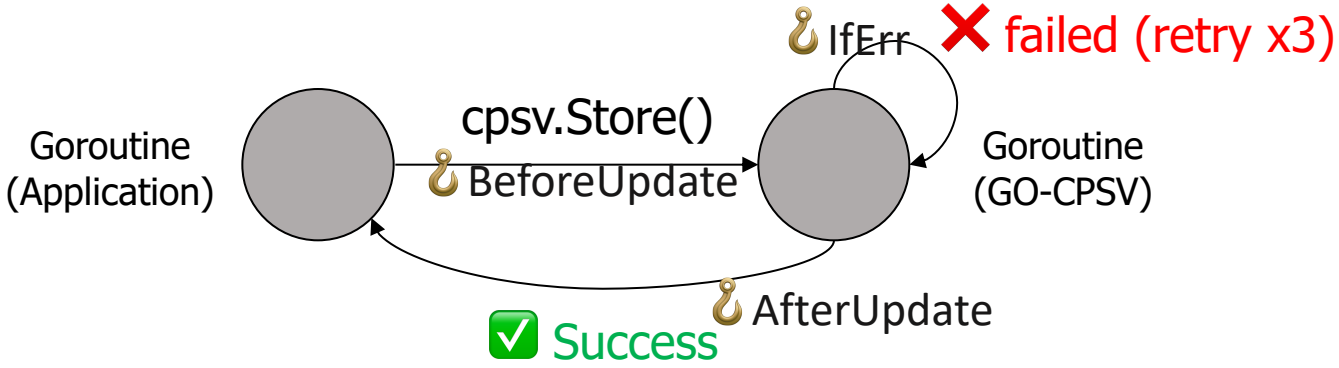
Implementation

Overview

- GO-CPSV
  - Transactional storing & LifeCycle hooks
  - Concurrency Optimization
  - Graceful shutdown
  - Continuous integration
  - Example
- HA-UDR
  - Processing flow
  - Deployment

Implementation

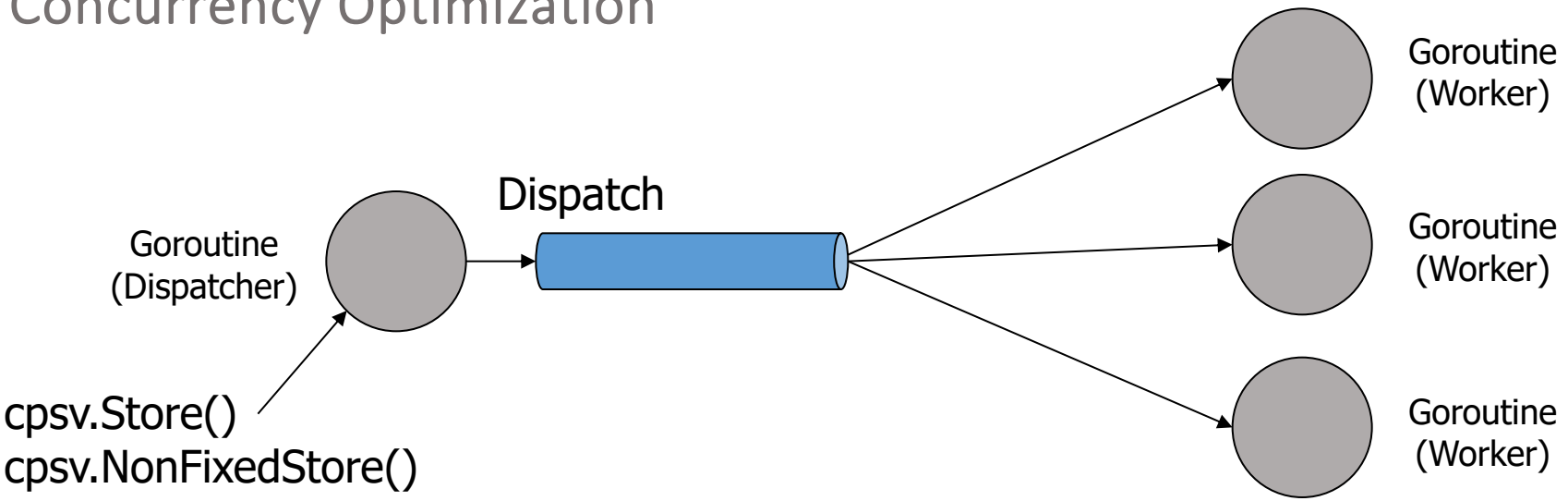
GO-CPSV::Transactional storing



- Re-transmission time is three by default, but the developer can configure this setting by invoking SetResentMax()
- LifeCycle hooks bring flexibility to the developer for injecting the specific handler (for tracing, logging, metrics collection, cache, change stream…)

Implementation

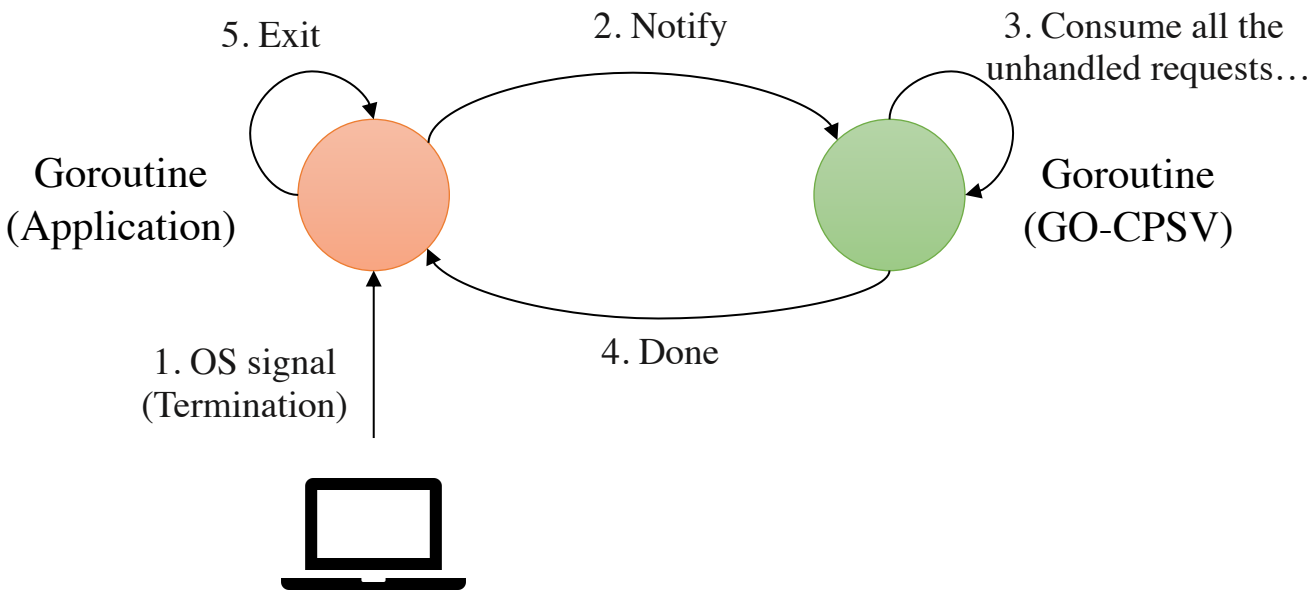
Concurrency Optimization



- GO-CPSV adopts a worker pool pattern to implement the storing handler. It reduces the response time significantly (⬇ 53% when 10 workers running)
- The amounts of workers is three by default, but the developer can configure this setting by invoking SetWokerNum()

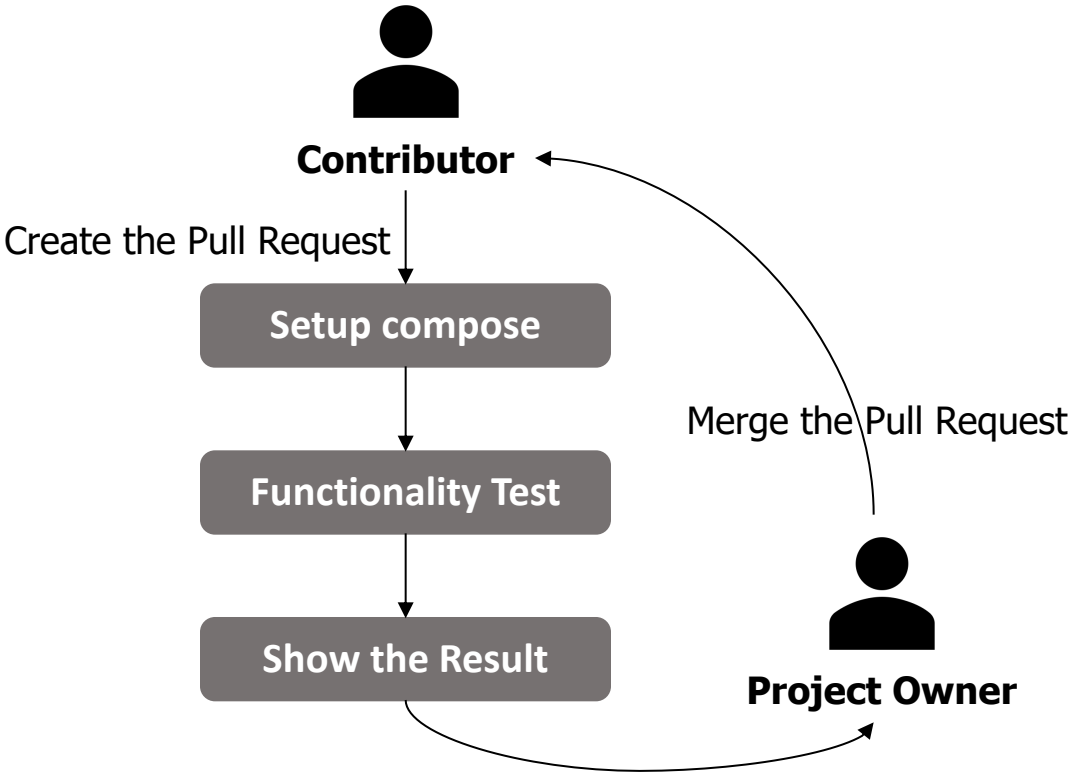
Implementation

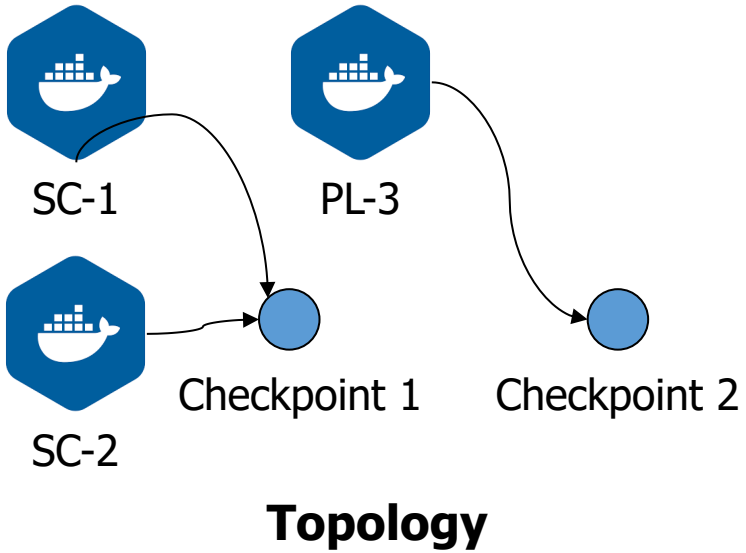
GO-CPSV::Graceful shutdown



Implementation

GO-CPSV::Continuous integration





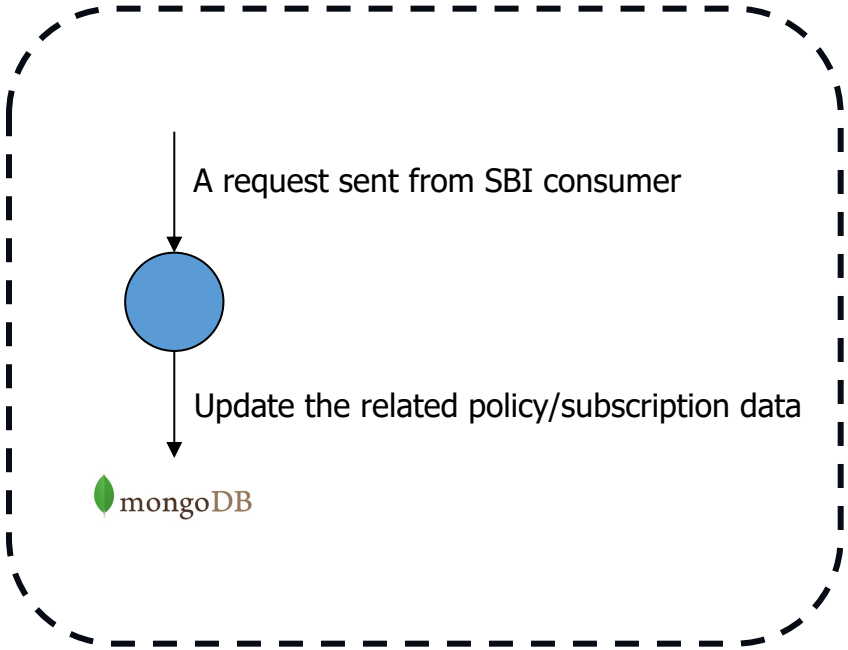
Implementation

GO-CPSV::Example

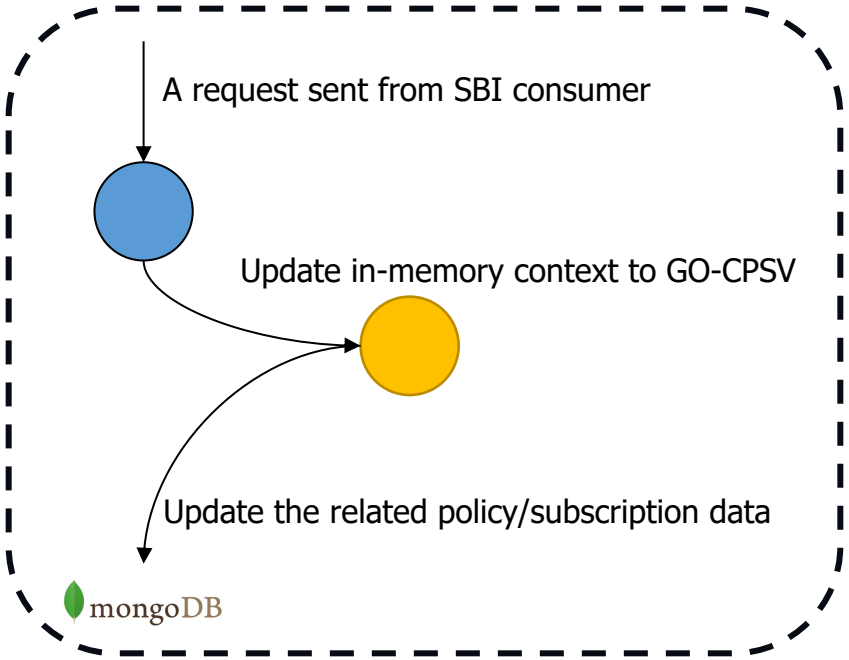
```go
func success(ctx context.Context) {
    res, err := cpsv.GetResult(ctx)
    if err != nil {
        log.Fatalln("failed to get result", err)
    } else {
        log.Println("success", res.SecId, res.Data)
    }
}

func main() {
    cpsv.Start("safCkpt=TEST2,safApp=safCkptService",
        cpsv.SetSectionNum(100000), cpsv.SetSectionSize(2000),
        cpsv.SetWorkerNum(10), cpsv.SetLifeCycleHooks(beforeUpdate, success,
    fail))

    for i := 0; i < 10000; i++ {
        data, _ := json.Marshal(i)
        cpsv.NonFixedStore(fmt.Sprintf("%d", i), data, len(data))
    }

    for i := 0; i < times; i++ {
        if _, err := cpsv.NonFixedLoad(fmt.Sprintf("%d", i)); err == nil {
            continue
        } else {
            fmt.Println(err)
        }
    }
    cpsv.Destroy()
}
```

Implementation

HA-UDR: processing flow



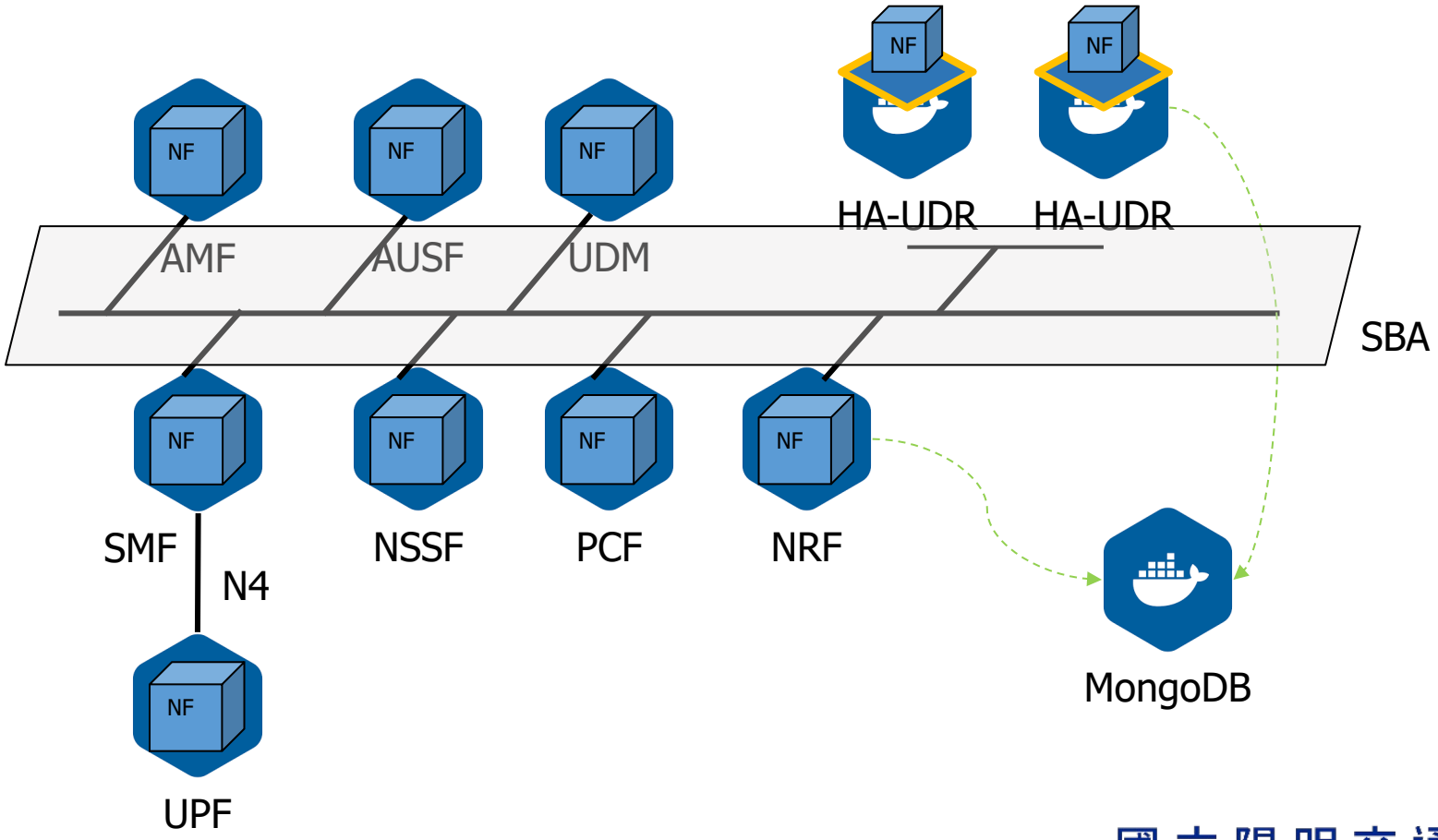
UDR



HA-UDR

Implementation

HA-UDR: deployment



Evaluation

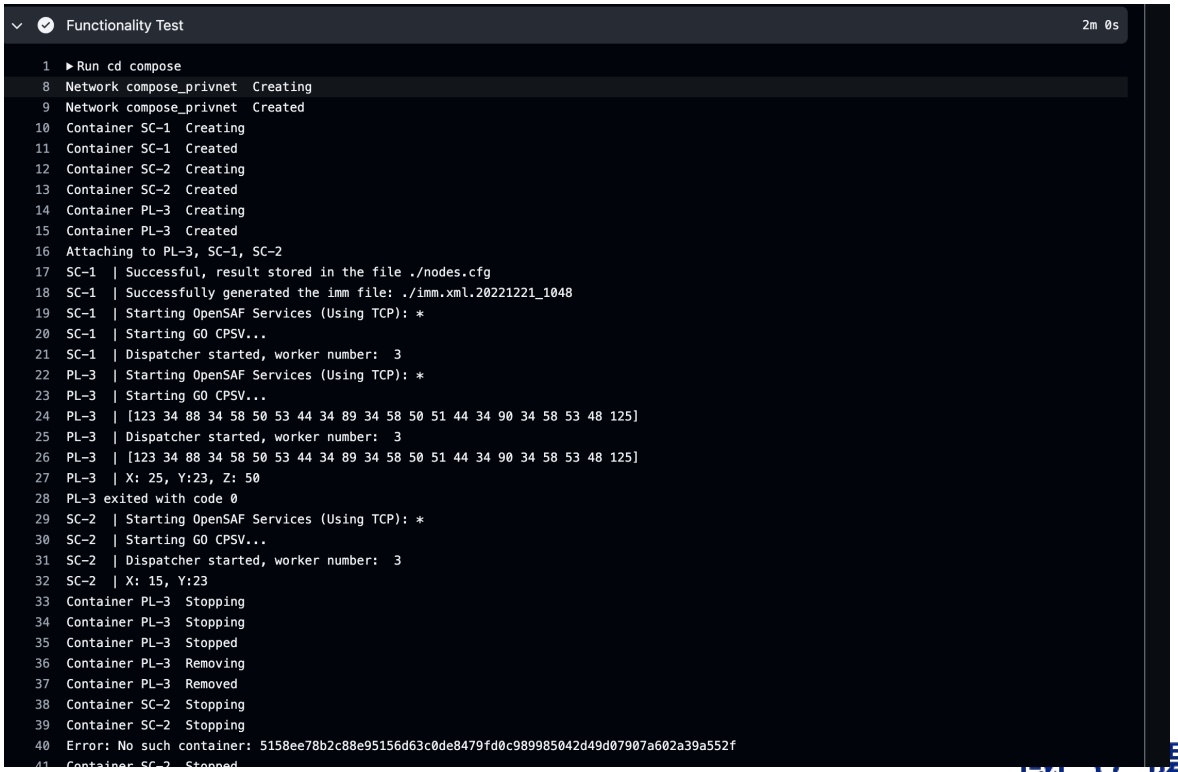
Testing Environment

- OS: Ubuntu 18.04 LTS
- CPU: QEMU (4C4T)
- RAM: 8GB
- Use docker-compose for simulating the OpenSAF cluster

Evaluation

Functionality: Context Recovery

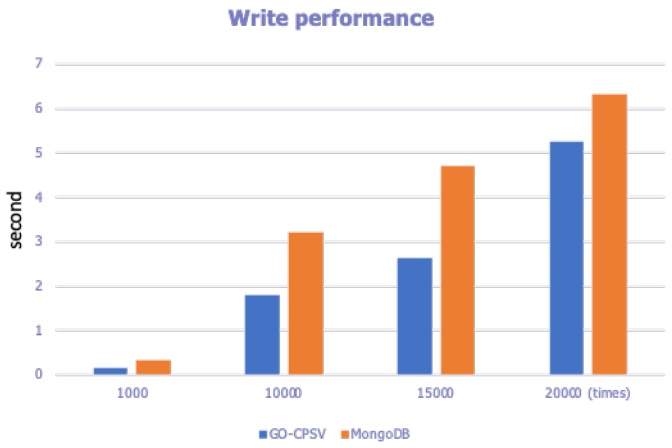
Validate the correctness on the writing and reading the user-defined data to/from GO-CPSV



Evaluation

Performance: W/R Test

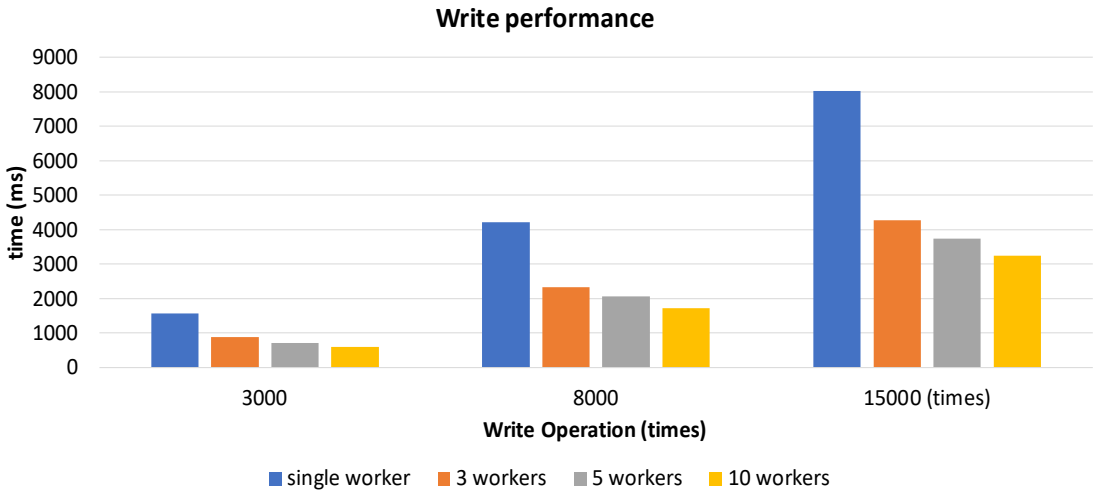
The testing will write and read the user-defined structure data to the GO-CPSV (3 workers), and compare the response time with MongoDB.



Evaluation

Performance: W/R Test for different worker sets

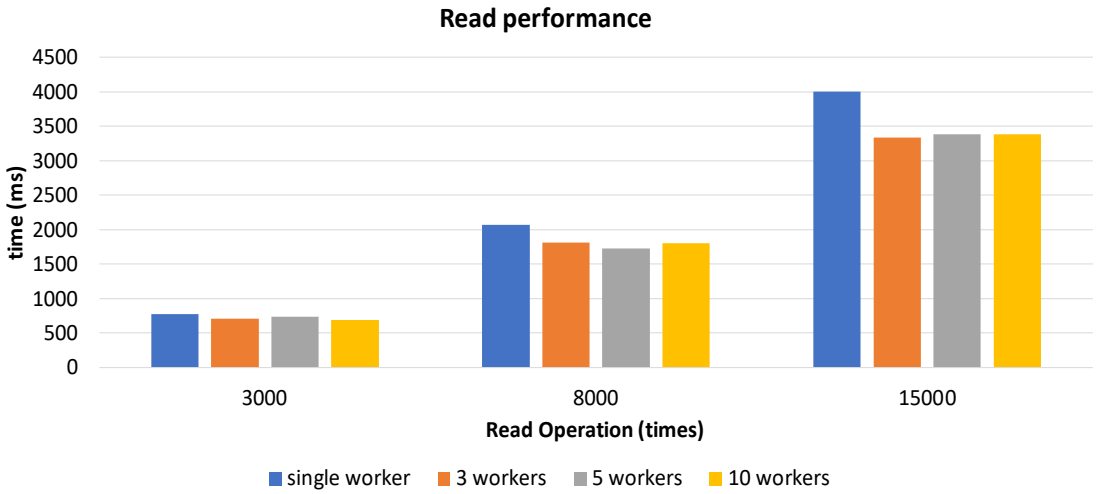
The testing will write and read the user-defined structure data to the GO-CPSV, with different worker sets



Evaluation

Performance: W/R Test for different worker sets

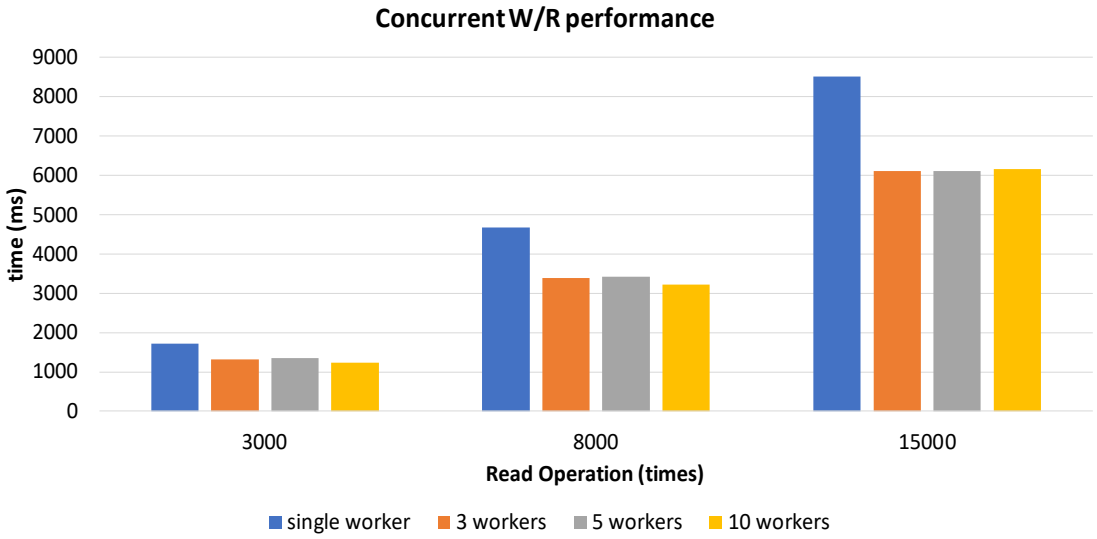
The testing will write and read the user-defined structure data to the GO-CPSV, with different worker sets



Evaluation

Performance: W/R Test for different worker sets

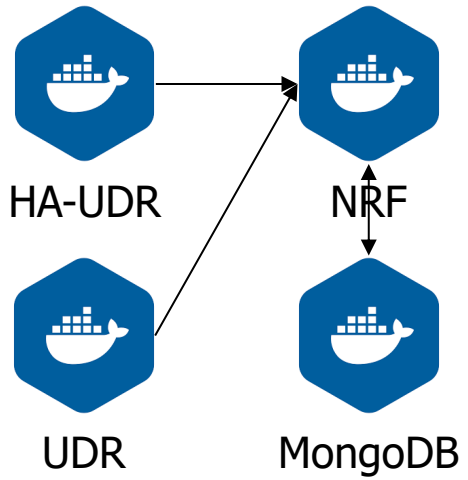
The testing will write and read the user-defined structure data to the GO-CPSV, with different worker sets



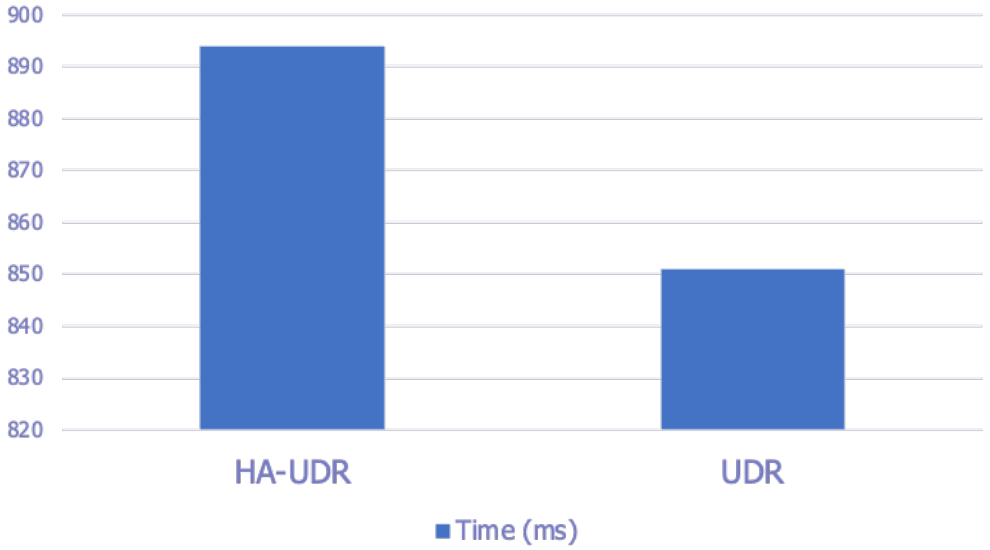
Evaluation

Throughput of modified Network Function

- Compared the http stress test performance of the original UDR and HA-UDR (1000 req)
- Target API: POST /nudr-dr/v1/policy-data/subs-to-notify



Topology



Conclusion

- GO-CPSV helps developer to archive HA to their legacy projects with less migration efforts
- Suitable for cloud-native based application
- High affinity for service observability
- Low latency (compared with existing solution)

Future Works

- Support to manage multiple checkpoints
- More storage solutions supported by default
- Change Stream (subs & notify)