GodBee

QuyenPT3, TranNDC

01 INTRODUCTION

Project Overview
Milestones

02 ZPKV service

How to implement
Demo

03 GOALS & EXPERIENCE

Deliverables

01 Introduction

Build a key-value store in C/C++
using B-Tree & B+Tree data structure.
Use GoLang and gRPC to build a service that calls to the key-value stores

Overview

MILESTONES

1ST MILESTONE

Theory

2ND MILESTONE

Implement B-Tree/B+Tree
Key-Value Storage
on memory

3RD MILESTONE

Implement B-Tree/B+Tree
Key-Value Storage
on disk

4TH MILESTONE

Implement
Key-Value Storage Service

Theory

1. **Overview**
   - Data structure: B-Tree B+Tree
   - Key-value storage
2. **Operating System**
   - Reader Writer Problem
   - Write Amplification
   - Memory Allocation

1st Milestone

Implement KV Storage on memory

- Implement KV Storage on memory with B-Tree/B+Tree
- Unit test
- Test memory leak

2nd Milestone

Implement KV Storage on disk

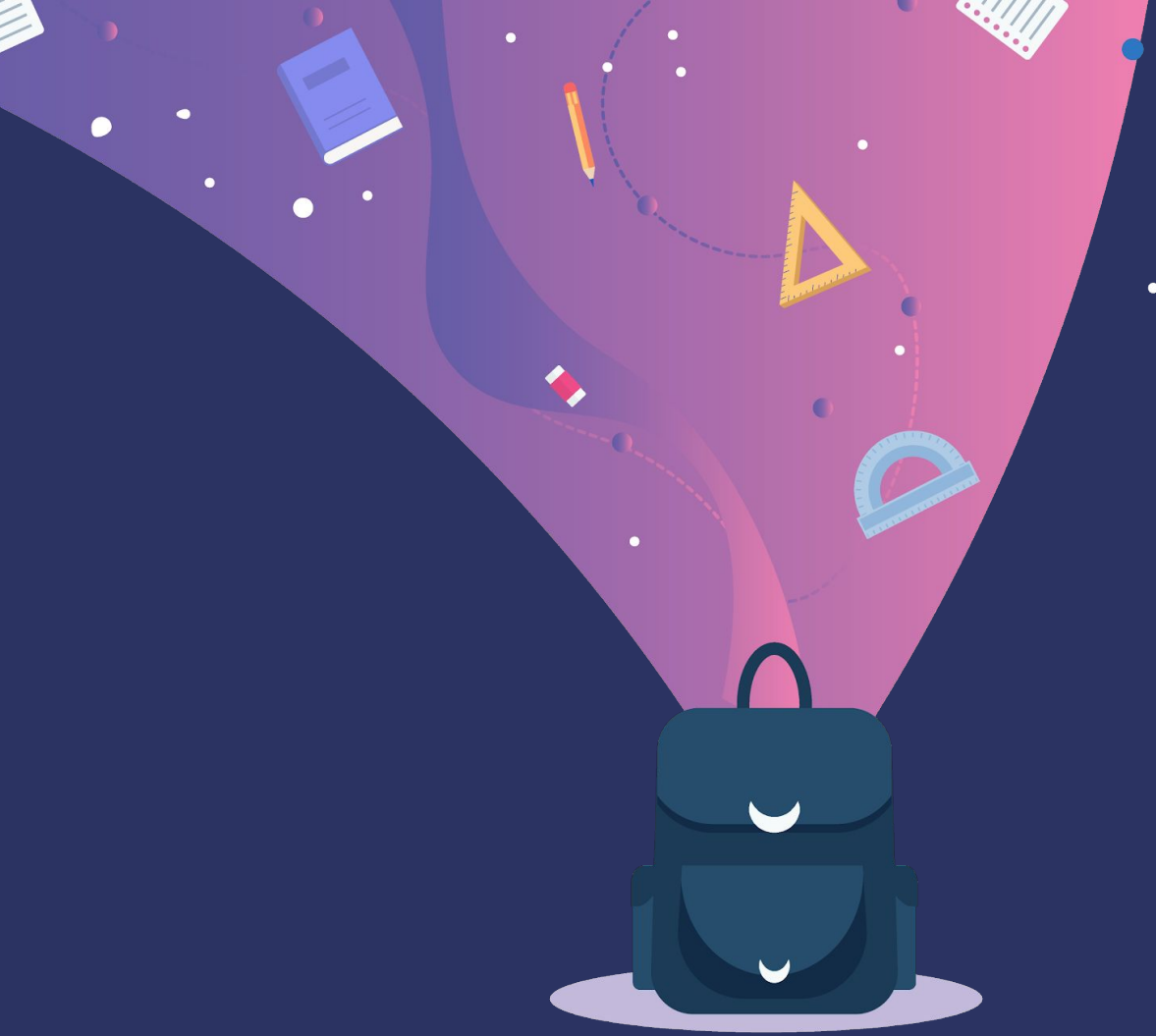
- Implement KV Storage on disk with B-Tree/B+Tree
- Unit test
- Test memory leak
- Handle concurrency

3rd Milestone

Implement
KV Storage Service

1. **GoLang basic**
   - CGO, protobuf, gRPC
2. **Implement service**
   - Use gRPC to implement service in GoLang
   - Combine between B-KV Storage & B+KV Storage
   - Unit test
   - Dockerfile
   - Benchmark with Locust and Boomer

4th Milestone

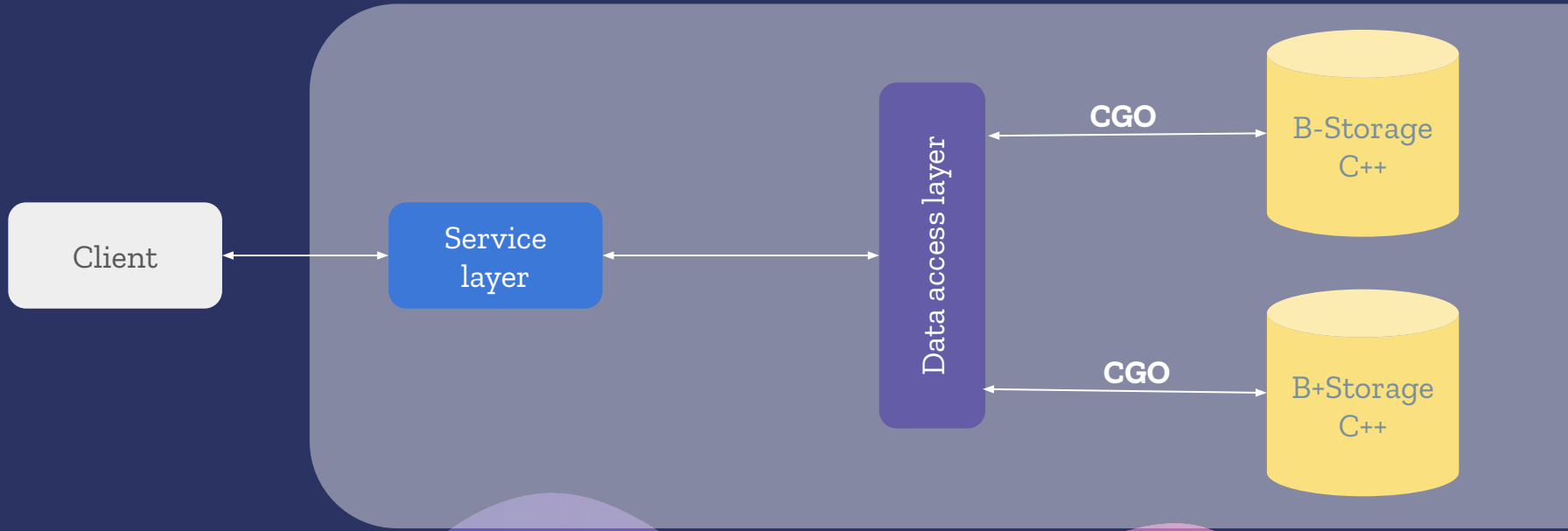


02

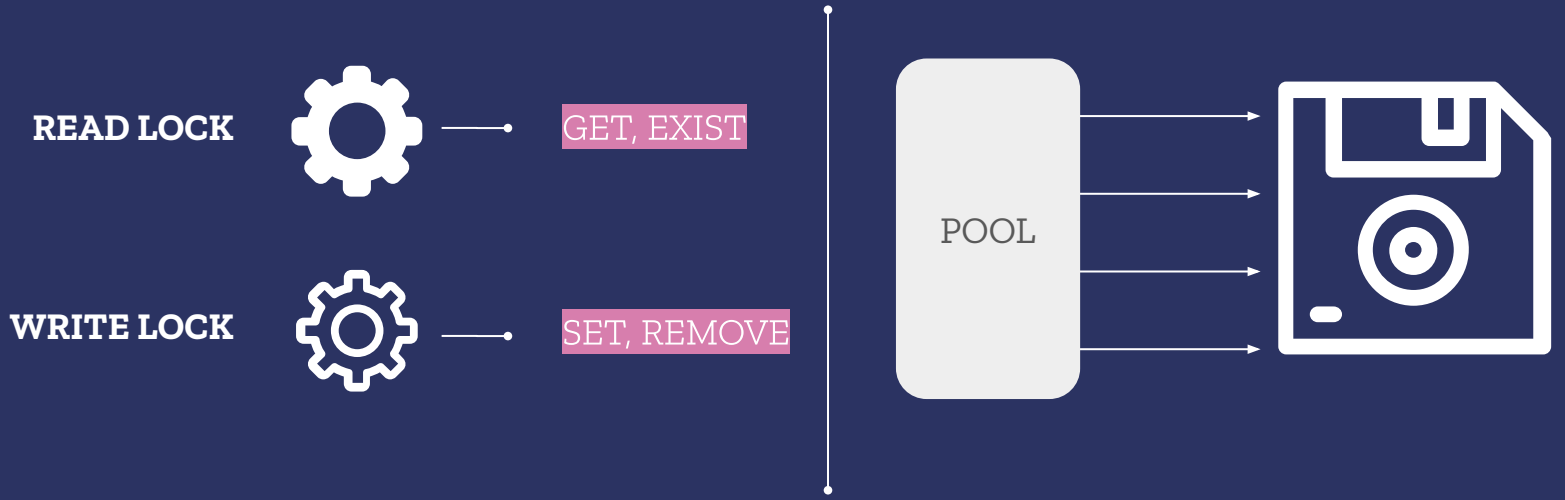
GodBee

Implementation
Demo

GodBee Architecture



Handling Concurrency



Call C++ methods from Golang

```
class BPlusTreeStore{
public:
    BPlusTreeStore(int maxDegree);
    char* get(char* key);
    void set(char* key,
         char *value);
    bool exist(char* key);
    bool remove(char* key);
    ~BPlusTreeStore();
};
```

```
typedef void* GBPlusTreeStore;
GBPlusTreeStore GBPlusInit(void);
void GBPlusFree(GBPlusTreeStore);
void GBPlusSet(GBPlusTreeStore,
    char * key, char * value);
char* GBPlusGet(GBPlusTreeStore,
    char * key);
int GBPlusRemove(GBPlusTreeStore,
    char * key);
int GBPlusExist(GBPlusTreeStore,
    char * key);
```

Call C++ methods from Golang

```
GBPlusTreeStore GBPlusInit()
{
    BPlusTreeStore * btree
     = new BPlusTreeStore();
    return (void*)btree;
}
void GBPlusFree(GBPlusTreeStore btree)
{
    BPlusTreeStore *pTree
     = (BPlusTreeStore *)btree;
    delete pTree;
}
```

```
void GBPlusSet(GBPlusTreeStore btree,
     char *key, char *value)
{
    BPlusTreeStore *pTree
          = (BPlusTreeStore *)btree;
    pTree->set(key, value);
}
char *GBPlusGet(GBPlusTreeStore btree,
     char *key)
{
    BPlusTreeStore *pTree
          = (BPlusTreeStore *)btree;
    return pTree->get(key);
}
```

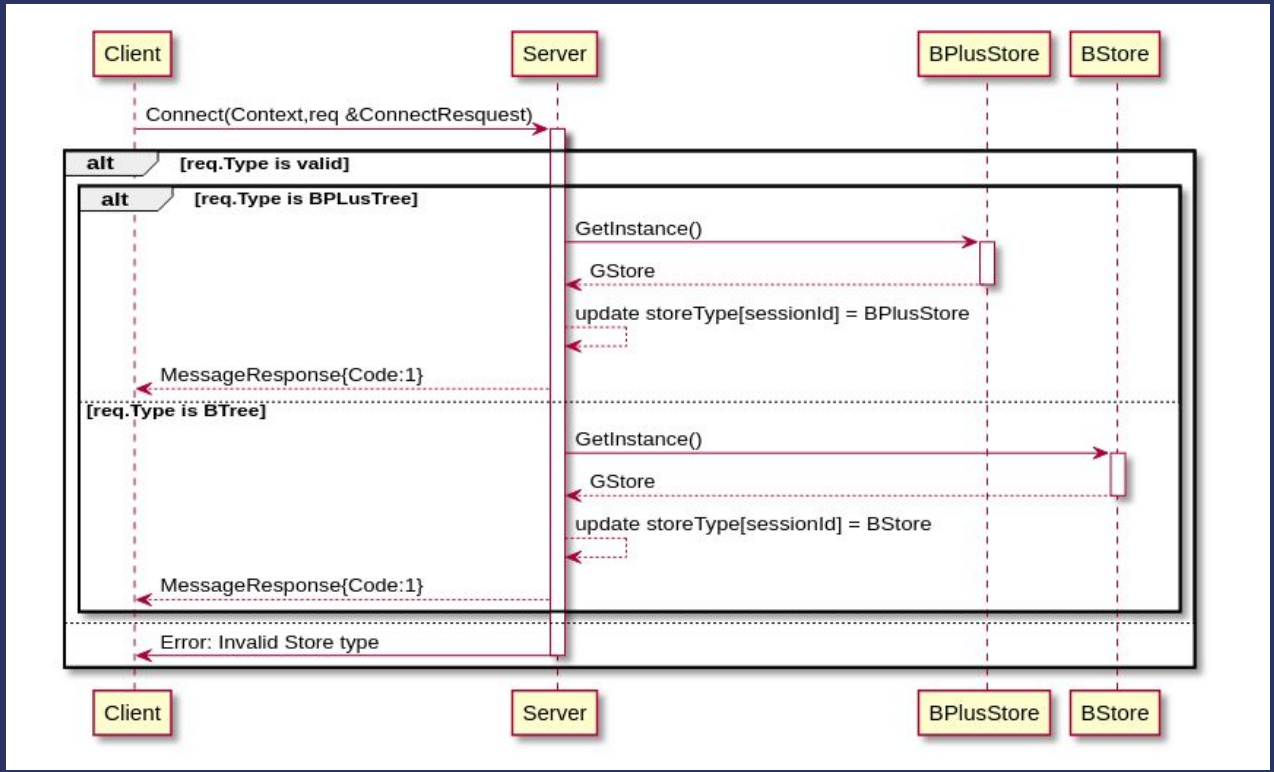
Call C++ methods from Golang

```
/*
#include "GBPlusTreeStore.h"
#include <stdlib.h>
*/
import (
    "C"
)
type GBPlusStore struct {
    tree C.GBPlusTreeStore
}
var gS GBPlusStore
var once sync.Once
type GBPlusStore struct {
    tree C.GBPlusTreeStore
}
```

```
func GetInstance() GBPlusStore {
    once.Do(func() {
        gS.tree = C.GBPlusInit()
    })
    return gS
}
func (gS GBPlusStore) Set(k string,
v string) {
    key := C.CString(k)
    value := C.CString(v)
    C.GBPlusSet(gS.tree, key, value)
    C.free(unsafe.Pointer(key))
    C.free(unsafe.Pointer(value))
}
```

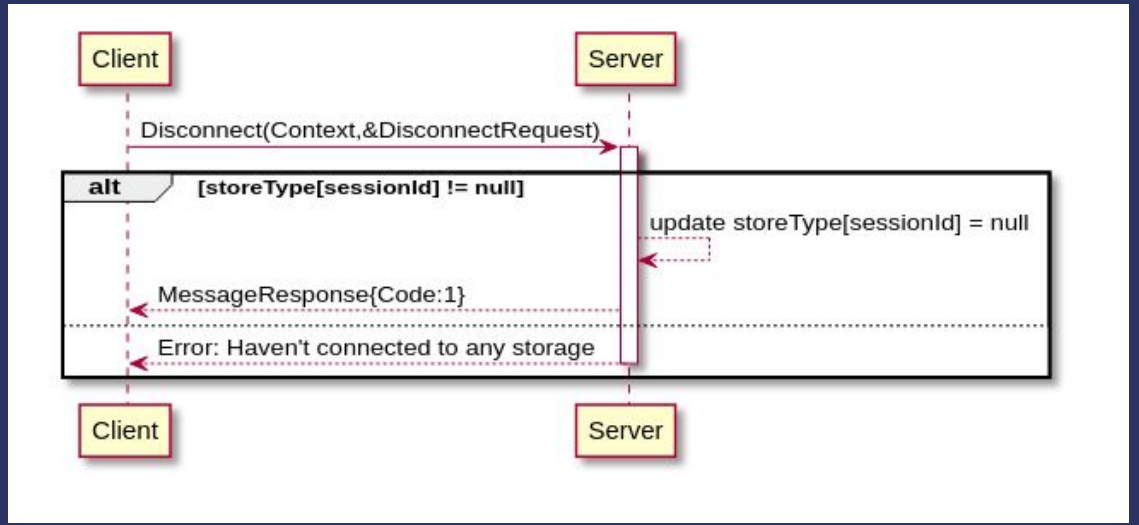
API Sequence Diagram

Connect

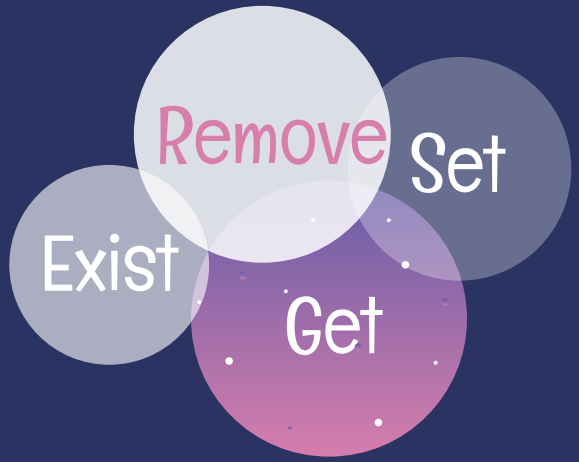


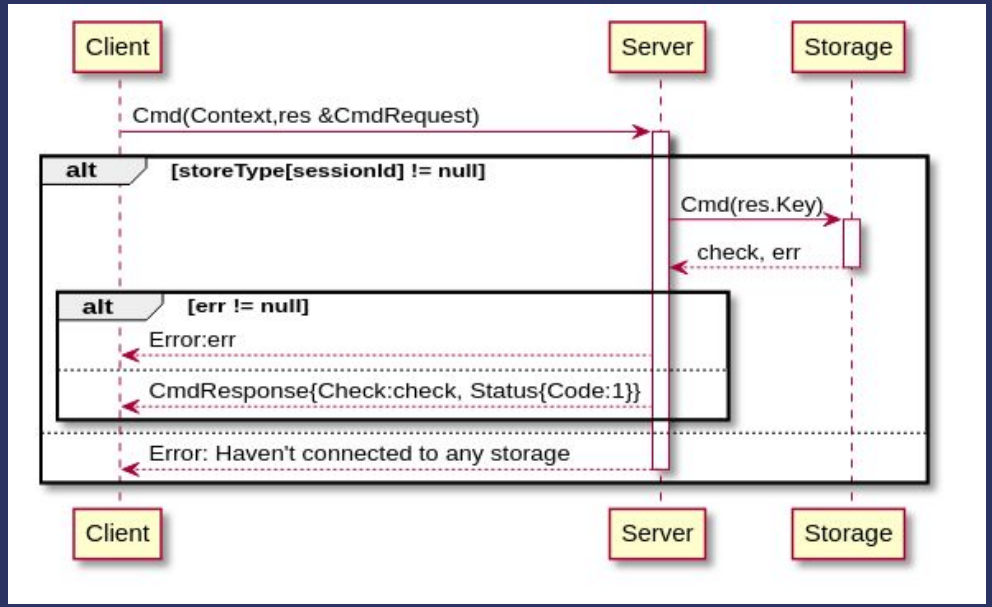
API Sequence Diagram

Disconnect



API Sequence Diagram





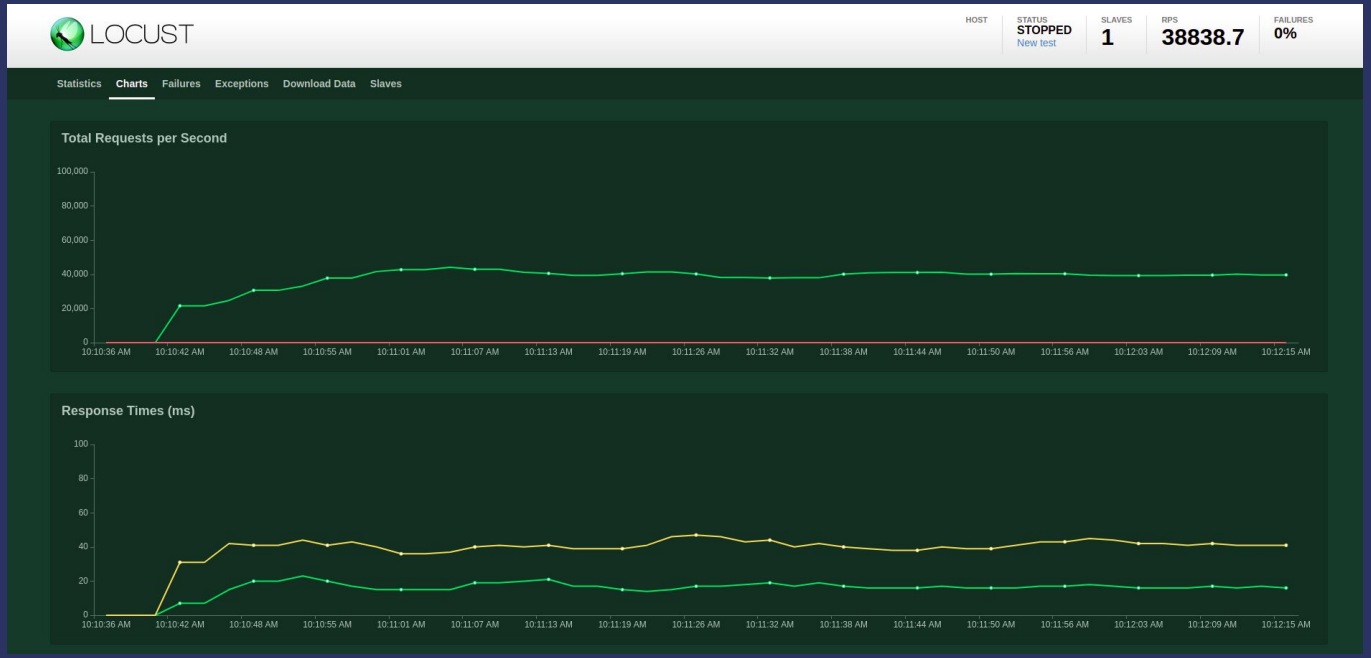
Benchmark Result

B-Storage

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
tcp	Connect	872166	0	15	30	17	1	209	4	8473.3	0
tcp	Disconnect	807927	0	16	30	18	1	222	38	8488	0
tcp	Exist	801710	0	16	31	18	1	222	36	8336.8	0
tcp	Get	811000	0	16	31	18	1	225	51	8314.4	0
tcp	Remove	253154	0	27	200	61	1	1294	33	2604	0
tcp	Set	268809	0	27	180	58	1	1332	32	2622.2	0
	Aggregated	**3814766**	**0**	**17**	**33**	**24**	**0**	**1332**	**32**	**38838.7**	**0**

Benchmark Result

B-Storage



Benchmark Result

B+Storage

LOCUST

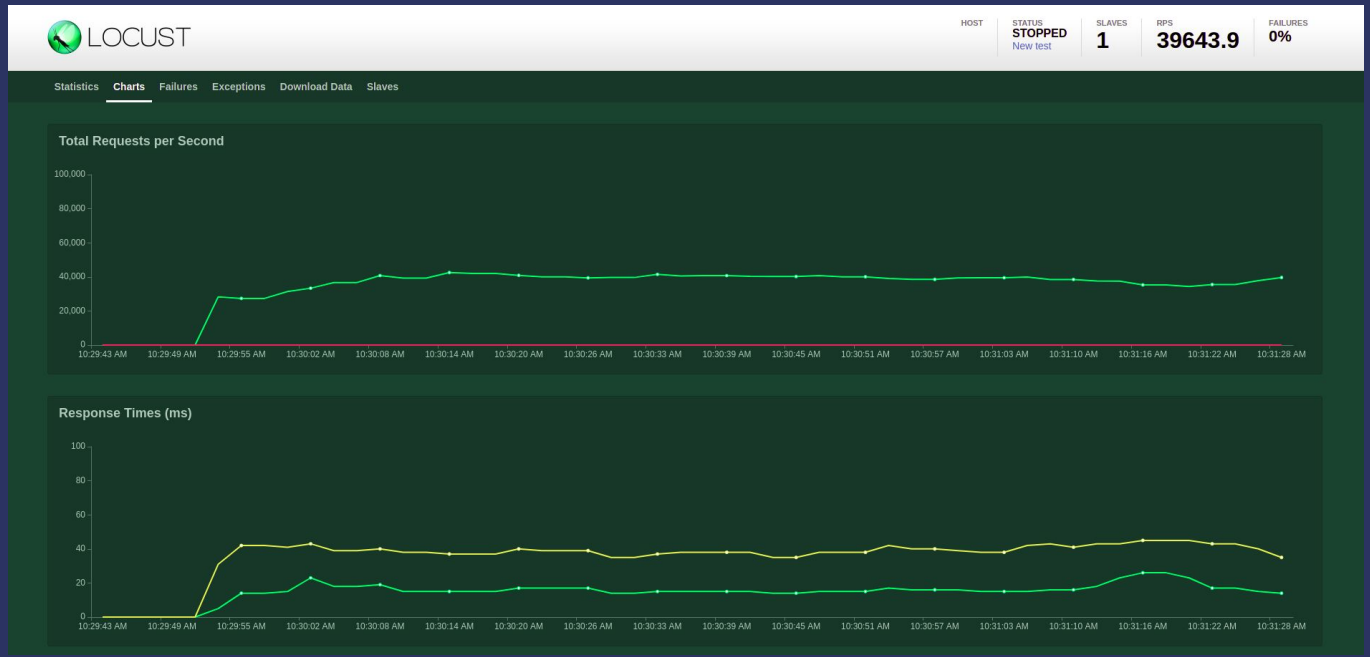
HOST	STATUS	SLAVES	RPS	FAILURES
	STOPPED New test	1	39643.9	0%

Statistics | Charts | Failures | Exceptions | Download Data | Slaves

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
tcp	Connect	915518	0	15	31	17	1	183	4	9264.1	0
tcp	Disconnect	840581	0	15	31	18	1	188	38	9238.9	0
tcp	Exist	839368	0	15	31	18	1	182	36	9167.6	0
tcp	Get	852417	0	15	31	18	1	188	42	9146	0
tcp	Remove	191269	0	31	310	83	1	2510	34	1412.7	0
tcp	Set	205059	0	30	300	79	1	1726	32	1414.6	0
	Aggregated	**3844212**	**0**	**16**	**34**	**24**	**1**	**2510**	**30**	**39643.9**	**0**

Benchmark Result

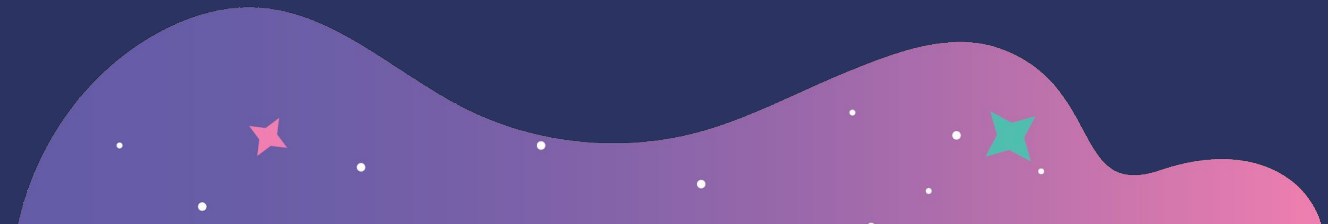
B+Storage



Benchmark Result

	Storage	Connect	Disconnect	Exist	Get	Remove	Set	Aggregated
95%	B-Store	35	35	36	36	300	290	41
	B+Store	36	36	36	36	410	410	40
99%	B-Store	49	49	50	51	450	440	250
	B+Store	46	47	48	48	610	610	310
100%	B-Store	210	220	220	230	1300	1300	1300
	B+Store	180	190	180	190	2500	1700	2500

DEMO



03
Experience

Handling concurrency when working with files

Learn Golang programming language

Improve reading and researching skills

Reinforce knowledge about Operating System

Improve debugging C++ skills

Teamwork, time management

Q&A

THANKS FOR LISTENING