Distributed Services with Go Notes

Noah Peters

April 10, 2023

Abstract

These are the notes for Travis Jeffery's *Distributed Services with Go: Your Guide to Reliable, Scalable, and Maintainable Systems*. As the book notes, Go has become the most popular language for building distributed services. It details tthe design, developmentt, and deployment of a distributed service. We first are guided through the construction of our storage architecture, then explore the networking of a client and server, and finally we see how to distribute server instances, deploy, and test.
LaTeXtemplate by Pingbang Hu.

Contents

Chapter 1

Let's Go

1.1 Go: An Overview

It's first important to note the advantages of Go, and to see why we are using it to build distributed systems. Go is a particularly good language for distributed systems because it is one that can simultaneously:

(i) compile annd ru programs faster than an interpreted language like Ruby or Python;

(ii) write highly concurrent (parallel) programs;

(iii) run directly on underlying hardware; and

(iv) use modern features like packages

1.2 JSON over HTTP

We begin our construction of distributed systems by building a commit log JSON over HTTP service. This is a widely used API, so it is a practical example at a small scale. For reference, a **commit log** is a data structure for an append-only sequence of records, ordered by time.

File 1 (internal/service/log.go)**.**

```
package server

import (
        "fmt"
        "sync"
)

type Log struct {
        mu sync.Mutex
        records []Record
}

func NewLog() *Log {
        return &Log{}
}

func (c *Log) Append(record Record) (uint64, error) {
        c.mu.Lock()
        defer c.mu.Unlock()
        record.Offset = uint64(len(c.records))
        c.records = append(c.records, record)
        return record.Offset, nil
}
```

```
func (c *Log) Read(offset uint64) (Record, error) {
        c.mu.Lock()
        defer c.mu.Unlock()
        if offset >= uint64(len(c.records)) {
                return Record{}, ErrOffsetNotFound
        }
        return c.records[offset], nil
}

type Record struct {
        Value []byte `json:"value"`
        Offset uint64 `json:"offset"`
}

var ErrOffsetNotFound = fmt.Errorf("offset not found")
```

Chapter 2

Known Bugs

Lecture 2: Second Lecture

9 Sep. 08:00

2.1 Introduction

Nothing is bugs-free. There are some known bugs which I don't have incentive to solve, or it is hard to solve whatsoever. Let me list some of them.

2.1.1 Footnote Environment

It's easy to let you fall into a situation that you want to keep using `footnote` to add a bunch of unrelated stuffs. However, with our environment there is a known strange behavior, which is following.

Example. Footnote![a]

> **Remark.** Oops! footnote somehow shows up earlier than expect![a]
>
> [a] This is a footnote!
> [a] This is another footnote!

Bugs caught![b]

[b] The final footnote which is ok!

As we saw, the footnote in the `Example` environment should show at the bottom of its own box, but it's caught by `Remark` which causes the unwanted behavior. Unfortunately, I haven't found a nice way to solve this. A potential way to solve this is by using `footnotemark` with `footnotetext` placing at the bottom of the environment, but this is tedious and needs lots of manual tweaking.

Furthermore, not sure whether you notice it or not, but the color box of `Remark` is not quite right! It extends to the right, another trick bug...

2.1.2 Mdframe Environment

Though `mdframe` package is nice and is the key theme throughout this template, but it has some kind of weird behavior. Let's see the demo.

Proof of ??. We need to prove the followings.

> **Claim.** $E = mc^2$.

Proof. Nonsense.
Nonsense,
Nonsense,
Nonsense,
Nonsense,
Nonsense. ⊛

■

I expect it should break much earlier, and this seems to be an algorithmic issue of `mdframe`. One potential solution is to use `tcolorbox` instead, but I haven't completely figure it out, hence I can't really say anything right now.

Appendix

Appendix A

Additional Proofs

A.1 Proof of ??

We can now prove **??**.

Proof of ??. See here. ■