Simple Object Storage - Weekly Report

Nijad Huseynov

June 2023

This week, I implemented basic key-value storage using the hashicorp raft library that I mentioned in report last week in order to gain practical experience with it. My original intent was to utilize the raft on servers hosting meta data. Now that I've given it some thought and done some research, I've come to the conclusion that, in the initial version, one meta data server will be sufficient because it won't be the main bottlenect of the whole design.

Then, I did an analysis of various technical papers, with my main focus on Facebook's Haystack storage system. I have figured out how I am going to store the multiple objects in one file (original paper does not mention how to implement it technically). And I have started the implementing the draft version of the data server based on that.

```
type Needle struct {
    TotalSize uint32
    NameSize  uint32
    Name      []byte
    DataSize  uint32
    Data      []byte
}
```

The above struct is the initial memory representation of the saved object on the storage. **TotalSize** field shows length of the total bytes in object. When reading the object, I will first read first 4 bytes to determine total object size, then based on that I will read whole chunk with the size **TotalSize**. After that, **NameSize** will be fetched similarly and based on that, the name of the object will be determined and so on. Note that, the above struct is the very basic version and I will expand on that.

Additionally, I have collected the data and implemented the following function to generate text data for testing. Currently, collected data consists of 1000 images and 100 pdfs. When the system is ready, I will upload the collected data to it and verify its correct functionality.

```
func getRandomString() string {
    var letters = []byte("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")
    l := rand.Int()%1000 + 1
    b := make([]byte, l)
    for i := 0; i < l; i++ {
        b[i] = letters[rand.Intn(len(letters))]

    }
    return string(b)
}

func generateRandomTextFiles(cnt int) {
    for i := 0; i < cnt; i++ {
        name := fmt.Sprintf("text-%d.txt", i)
        file, err := os.Create("data/" + name)
        if err != nil {
            fmt.Println("Could not create text file")
            return
        }

        _, err = file.Write([]byte(getRandomString()))
        if err != nil {
            fmt.Println("Could not write to the file")
            return
        }

        _ = file.Close()
    }
}
```

Images and pdfs are downloaded from following resoursces respectively: https://github.com/unsplash/datasets and https://github.com/manjunath5496/Open-Access-Books.