链滴

# Golang 超大文件读取的两个方案

作者：xiaoxiezaijia

原文链接：https://ld246.com/article/1564806399214

来源网站：链滴

1. 流处理方式
2. 分片处理

去年的面试中我被问到超大文件你怎么处理，这个问题确实当时没多想，回来之后仔细研究和讨论了这个问题，对大文件读取做了一个分析

比如我们有一个 log 文件，运行了几年，有 100G 之大。按照我们之前的操作可能代码会这样写：

```
func ReadFile(filePath string) []byte{
    content, err := ioutil.ReadFile(filePath)
    if err != nil {
        log.Println("Read error")
    }
    return content
}
```

上面的代码读取几兆的文件可以，但是如果大于你本身及其内存，那就直接翻车了。因为上面的代码是把文件所有的内容全部都读取到内存之后返回，几兆的文件，你内存够大可以处理，但是一旦上几兆的文件，就没那么好处理了。那么，正确的方法有两种，第一个是使用流处理方式代码如下：

```
func ReadFile(filePath string, handle func(string)) error {
    f, err := os.Open(filePath)
    defer f.Close()
    if err != nil {
        return err
    }
    buf := bufio.NewReader(f)

    for {
        line, err := buf.ReadLine("\n")
        line = strings.TrimSpace(line)
        handle(line)
        if err != nil {
            if err == io.EOF{
                return nil
            }
            return err
        }
        return nil
    }
}
```

第二个方案就是分片处理，当读取的是二进制文件，没有换行符的时候，使用下面的方案一样处理大件

```
func ReadBigFile(fileName string, handle func([]byte)) error {
    f, err := os.Open(fileName)
    if err != nil {
        fmt.Println("can't opened this file")
        return err
    }
    defer f.Close()
```

```
    s := make([]byte, 4096)
    for {
        switch nr, err := f.Read(s[:]); true {
        case nr < 0:
            fmt.Fprintf(os.Stderr, "cat: error reading: %s\n", err.Error())
            os.Exit(1)
        case nr == 0: // EOF
            return nil
        case nr > 0:
            handle(s[0:nr])
        }
    }
    return nil
}
```

希望能对学习 go 语言的同学有所帮助。

作者：bean

链接：https://learnku.com/articles/23559/two-schemes-for-reading-golang-super-large-files

来源： LearnKu 终身编程者的知识社区