**性能测试和优化报告**

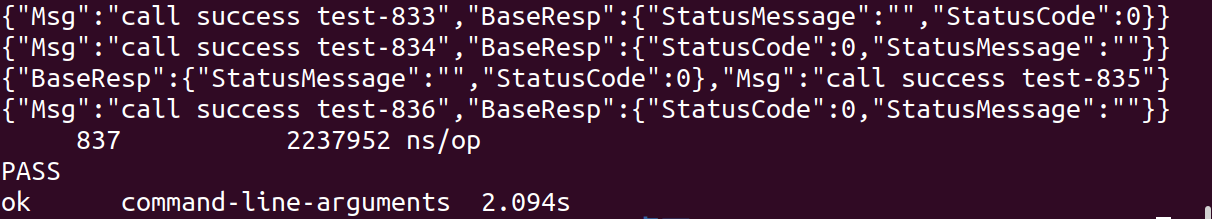
测试人：程泽天 测试平台：Ubuntu20.04 测试时间：2023/7/31

**1. 性能测试**

性能测试基于Golang性能测试框架Benchmark及第三方性能测试工具Apache Benchmark进行

1.1 基准测试

使用Benchmark向AServiceName发送请求，显示数据：

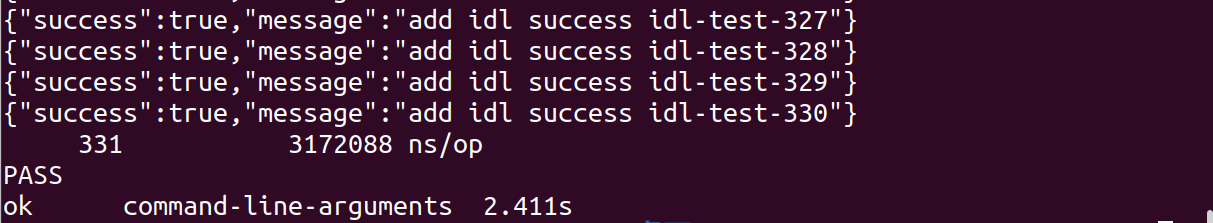
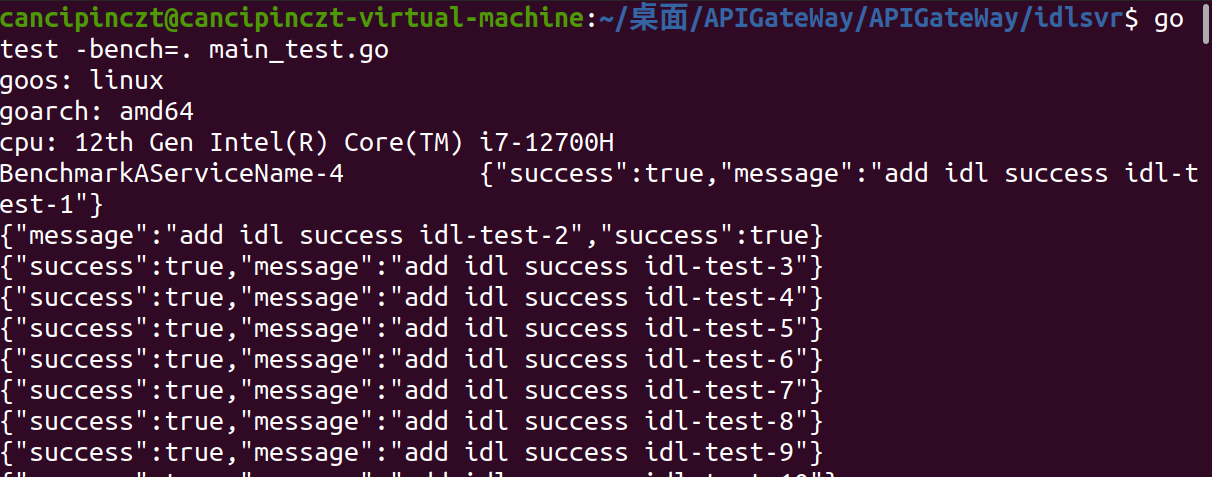


共使用：4个核心

处理次数：837次

每次耗时：2237952ns

使用Benchmark向idlSvr发送请求，显示数据：

共使用：4个核心

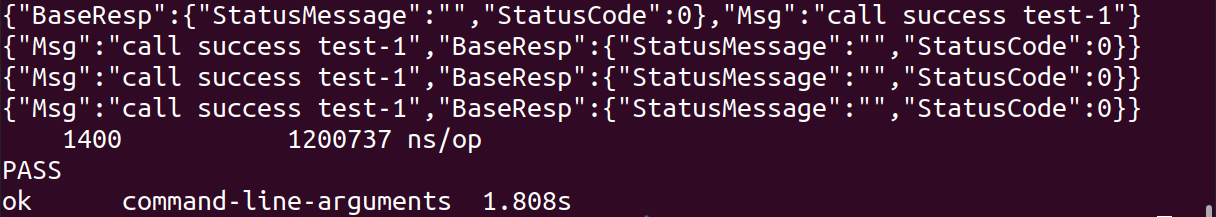
处理次数：331次

每次耗时：3172088ns

1.2 并行测试

用于测量在并发情况下性能指标，可反映被测代码的横向可扩展性

基于Benchmark设置最大并发数为4，显示数据：

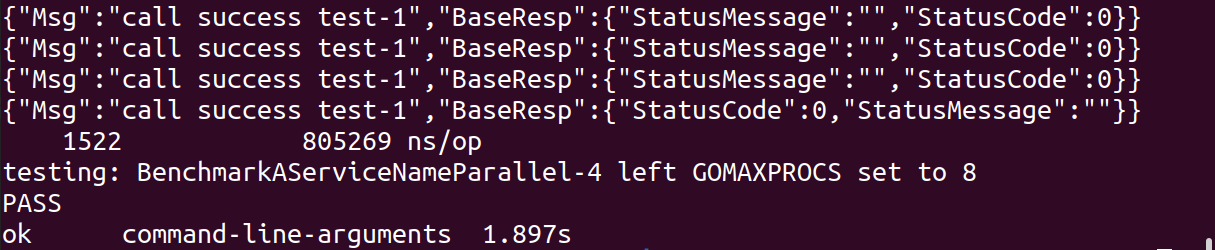
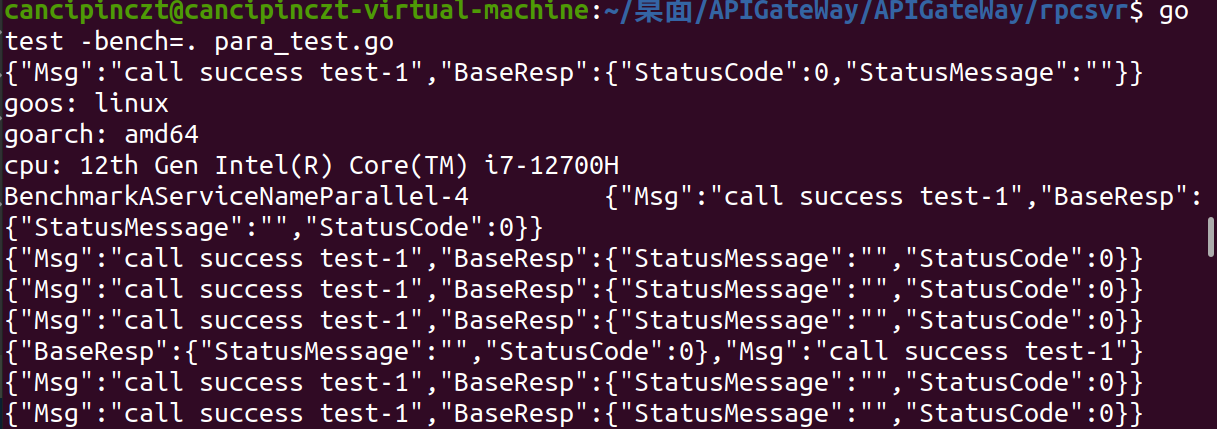


共使用：4个核心

处理次数：1400次

每次耗时：1200737ns

基于Benchmark设置最大并发数为8，显示数据：

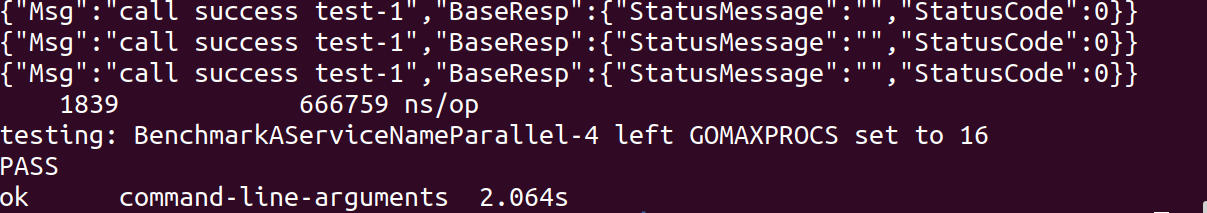
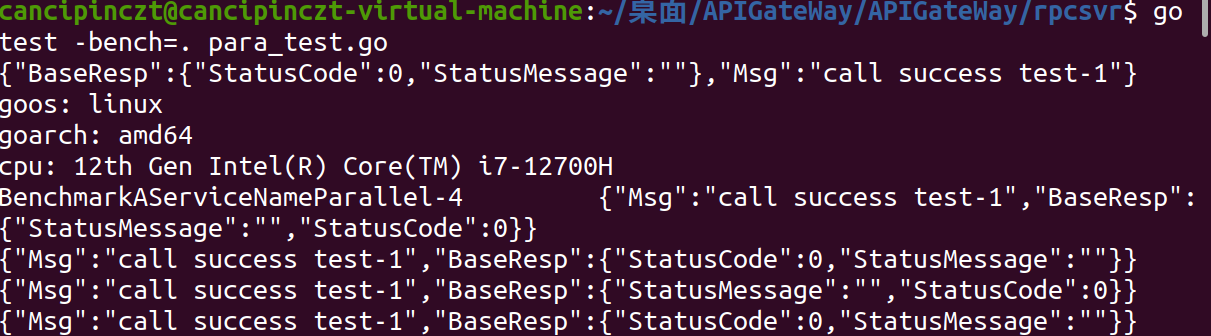


共使用：4个核心

处理次数：1522次

每次耗时：805269ns

基于Benchmark设置最大并发数为16，显示数据：



共使用：4个核心

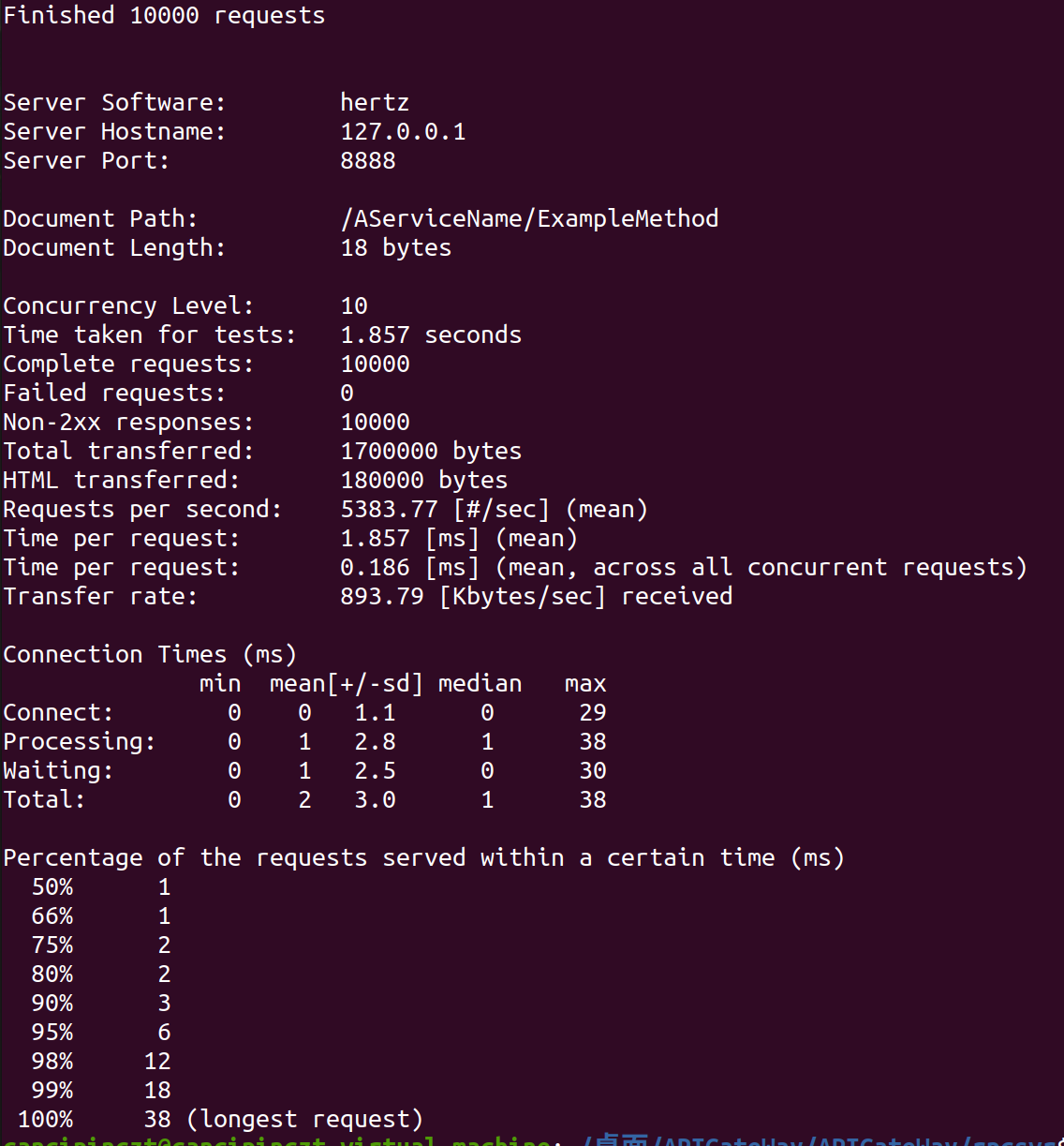
处理次数：1839次

每次耗时：666759ns

测试报告解读：并发数增加，每次耗时会逐渐减少，但被测代码往往有共享资源（例如锁，内存，CPU等），性能指标不一定会随并发数线性增长。

1.3 Apache Benchmark 测试

使用AB向AServiceName发送请求，在BServiceName服务开启之前的数据显示：

Concurrency Level: 10

Complete requests: 10000

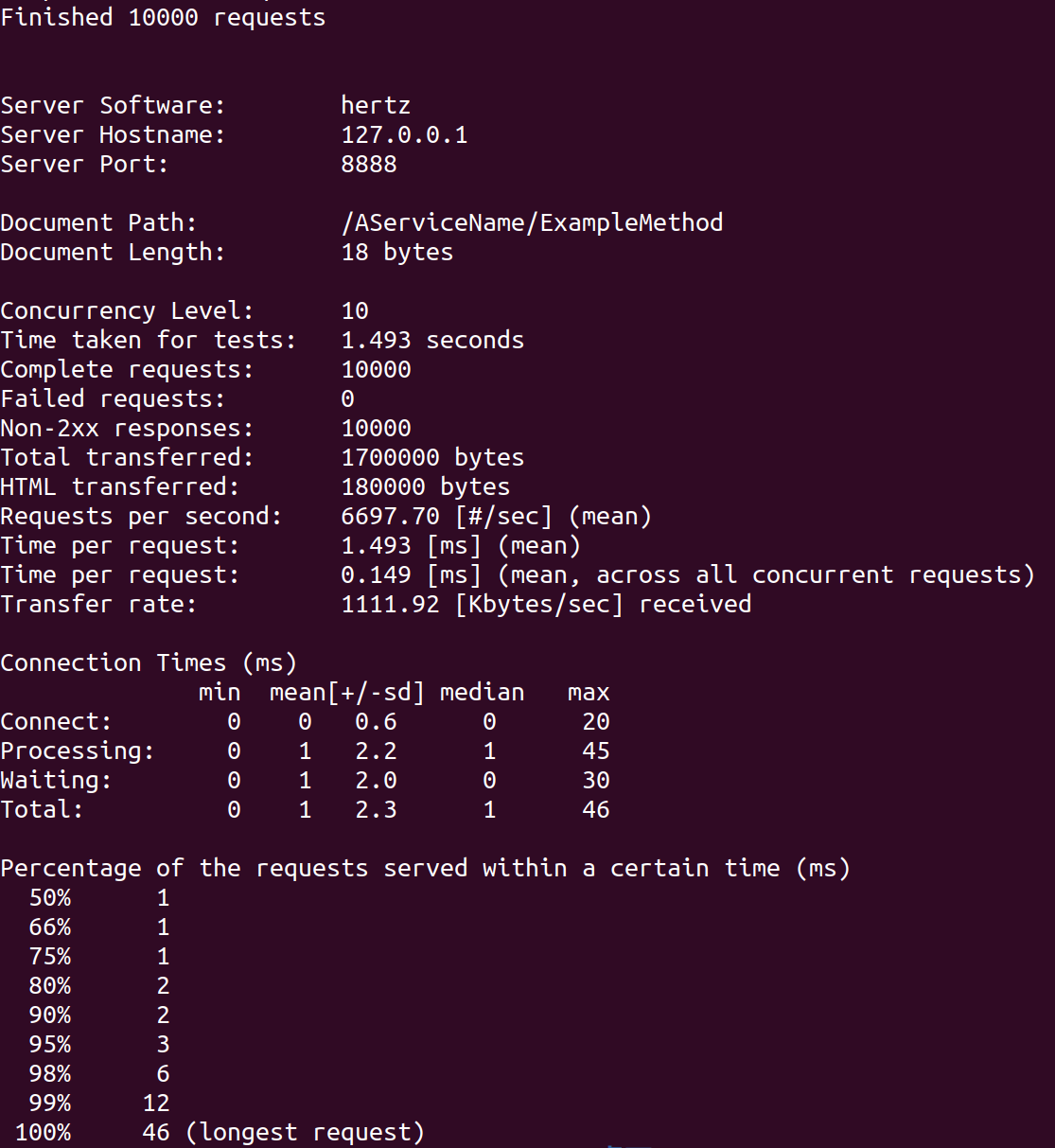
每秒发送5383个请求

95%的请求在6ms内完成

99%的请求在18ms内完成

100%的请求在38ms内完成

使用AB向AServiceName发送请求，在BServiceName服务开启之后的数据显示：

Concurrency Level: 10

Complete requests: 10000

每秒发送6697个请求

95%的请求在3ms内完成

99%的请求在12ms内完成

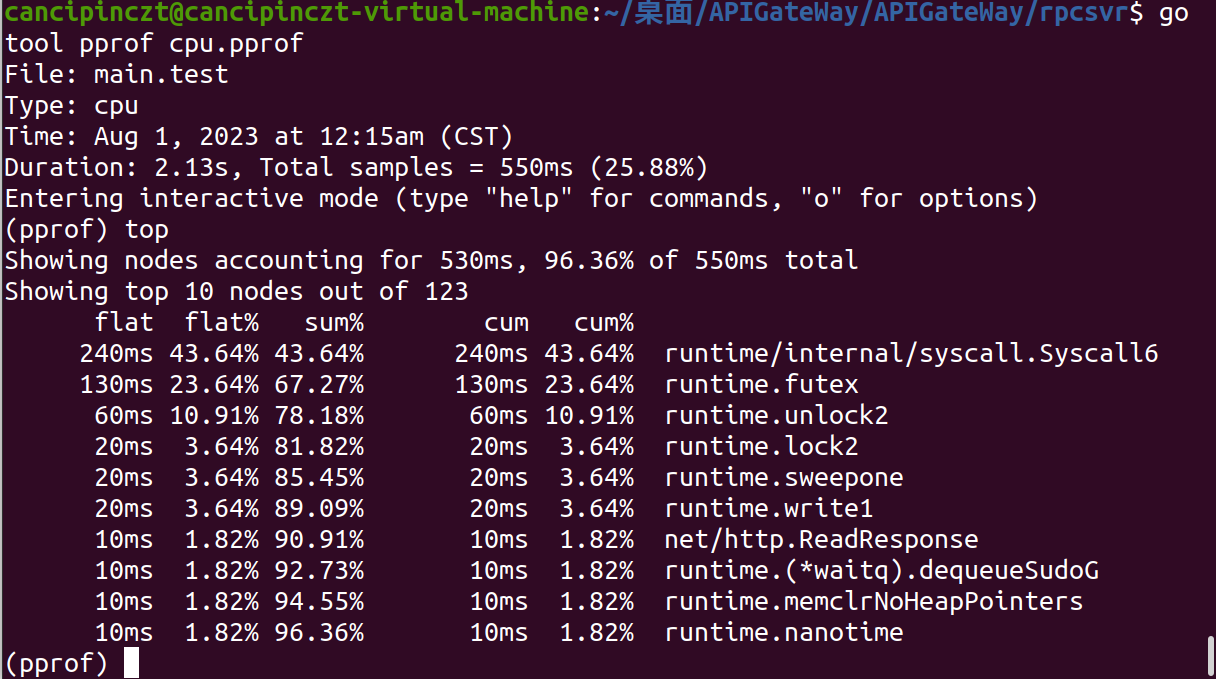
100%的请求在46ms内完成

测试报告解读：开启BServiceName之后，可以实现负载均衡，分担AServiceName的压力，使得每秒发送请求数上升，请求完成时间缩短，请求质量提升。

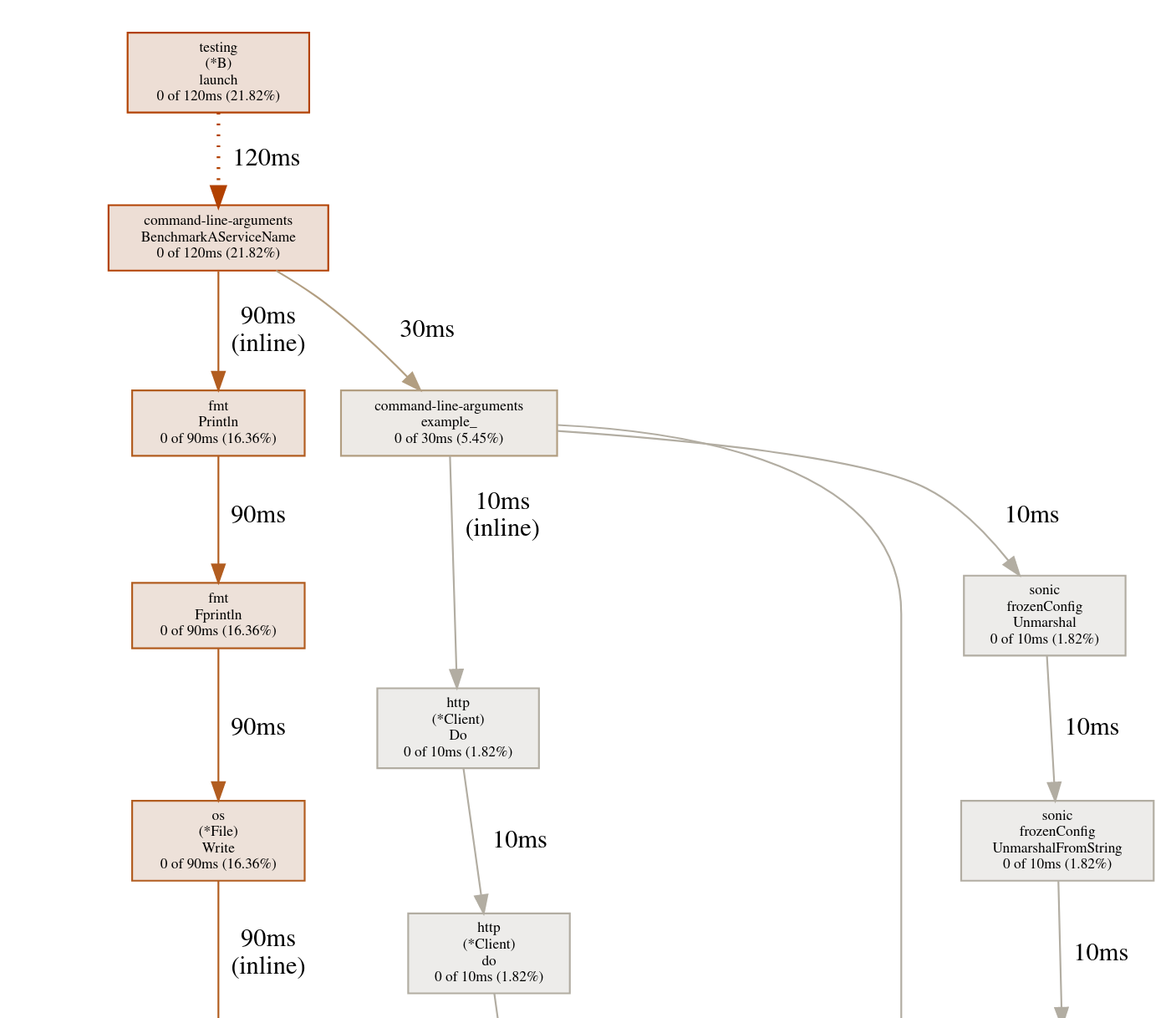
**2. 优化视图**

2.1 pprof 测试 命令行输出结果

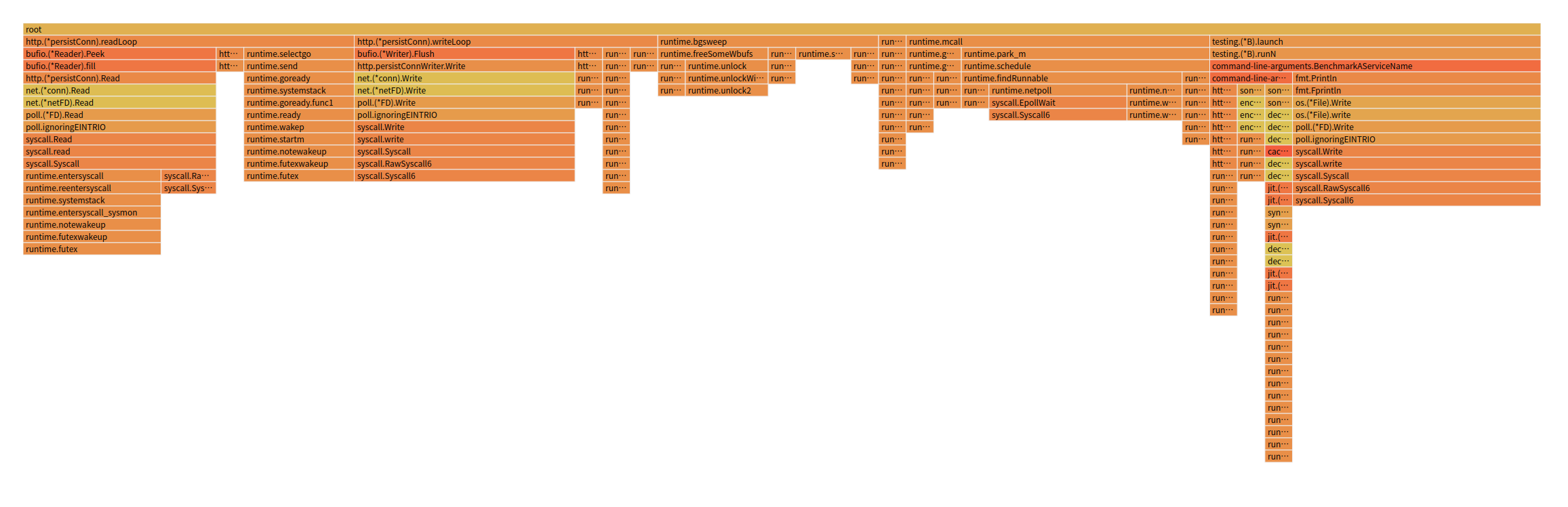
使用pprof对Benchmark测试进行优化观测，数据产出显示：



局部连线图如下：



火焰图如下：



从图和数据可以看出，程序中有一段fmt.Println占用资源比较大

经查，是benchmark测试函数的最后打印了response，我们将它注释掉，再添一行使用resp的语句以通过编译

func BenchmarkAServiceName(b \*testing.B) {

for i := 1; i < b.N; i++ {

newExample := genExample(i)

resp, err := example\_(newExample)

Assert(b, err == nil, err)

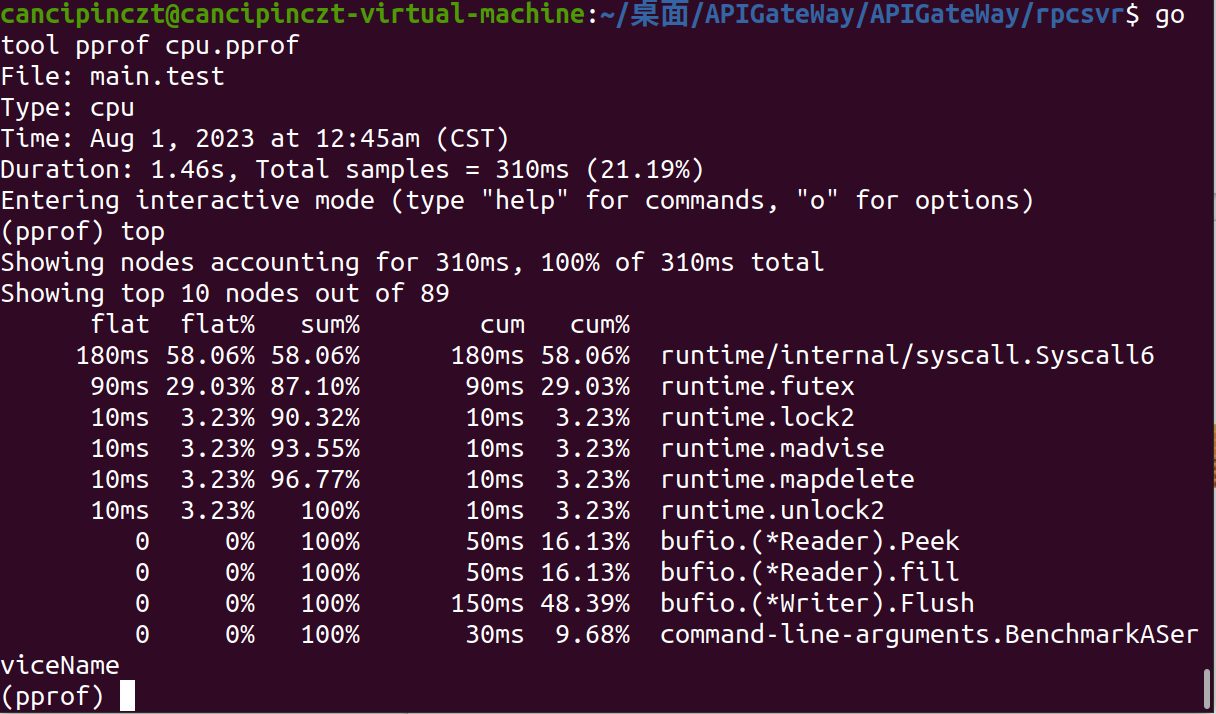
//fmt.Println(\*resp)

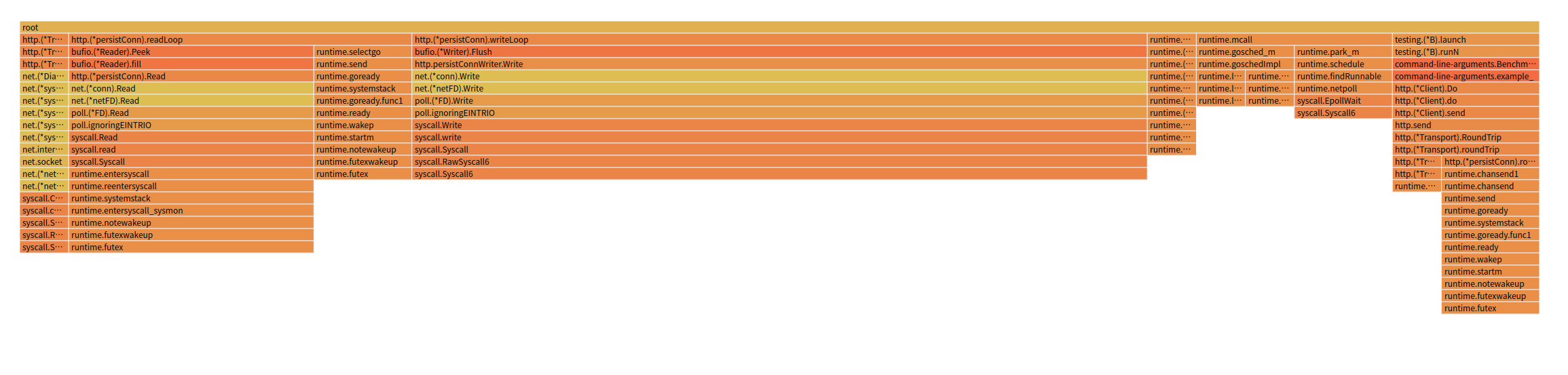
\*resp = \*resp + ""

}

}

重新使用pprof对Benchmark测试进行优化观测，数据产出显示：





可以看到，数据明显好了很多

但性能仍不算高，仍需进一步探索和优化