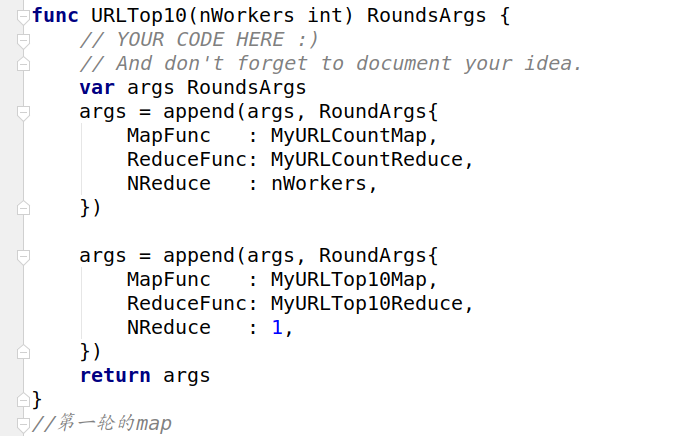


**开始**

根据pprof图我们发现时间大部分还是花在了文件的读写与json序列化上了。对比 mit的那个框架本框架的优点在于



不管map-reduce有多少轮，只要我们在最后一轮reduce将结果文件设置为1。这样就省去在run()函数中的merge阶段。极大的提升了框架的性能。

**分析**

根据上面的pprof我们得出时间大部分花费在文件读写和序列化上。那么我们优化就大概有两种方法。第一种直接使用字符串文件读写抛弃序列化（修改mapreduce.go的worker（）函数即可，可以复用框架提供的top10）。第二种压缩map-reduce的过程，减少无用的数据降低文件读写的压力（即修改urltop10）。

**Example的问题所在**

1. 第一次map重复的url还是存储了
2. 第二次map可以进行裁剪，但是example并没有这么做。
3. Json花费了大量的时间可否替换高效的json库。
4. 为了提高速度，是否可以放弃json直接string文件读写。

我选择了第二种方式。思路就是既然我们只需要获取top10，那么我就可以在第一个阶段统计数量去除重复数据，第二阶段的map阶段将数据裁剪，只留下每个map的top10这样我最后reduce的文件读写压力就会大大减轻。

虽然这里需要排序一次，但是比起大量的文件读写还是好很多的。

那么第一种性能优化我称之为暴力优化，第二种我称之为数据裁剪。