

# Lab0 实验报告

## 实验结果

### 1. 完成 Map-Reduce 框架

make test\_example 的实验截图：

```
GOROOT=D:\Go #gosetup
GOPATH=C:\Users\21548\go #gosetup
D:\Go\bin\go.exe test -c -o C:\Users\21548\AppData\Local\Temp\GoLand\___1TestExampleURLTop_in_talent.test.exe talent #gosetup
D:\Go\bin\go.exe tool test2json -t C:\Users\21548\AppData\Local\Temp\GoLand\___1TestExampleURLTop_in_talent.test.exe -test.v -te
=== RUN TestExampleURLTop
Case0 PASS, dataSize=1MB, nMapFiles=5, cost=81.1733ms
Case1 PASS, dataSize=1MB, nMapFiles=5, cost=48.2603ms
Case2 PASS, dataSize=1MB, nMapFiles=5, cost=44.3519ms
Case3 PASS, dataSize=1MB, nMapFiles=5, cost=59.5111ms
Case4 PASS, dataSize=1MB, nMapFiles=5, cost=99.0582ms

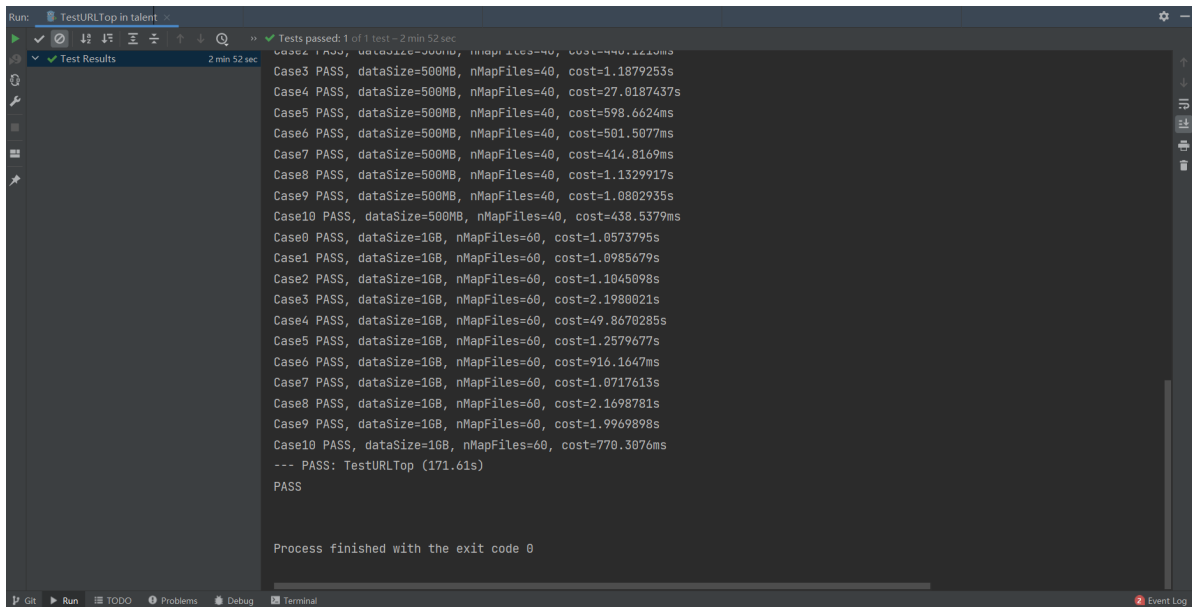
Case7 PASS, dataSize=1GB, nMapFiles=60, cost=1m15.2695289s
Case8 PASS, dataSize=1GB, nMapFiles=60, cost=33.4810493s
Case9 PASS, dataSize=1GB, nMapFiles=60, cost=27.411196s
Case10 PASS, dataSize=1GB, nMapFiles=60, cost=23.590151s
--- PASS: TestExampleURLTop (899.34s)
PASS

Process finished with the exit code 0
```

### 2. 基于 Map-Reduce 框架编写 Map-Reduce 函数

make test\_homework 的实验截图：

```
GOROOT=D:\Go #gosetup
GOPATH=C:\Users\21548\go #gosetup
D:\Go\bin\go.exe test -c -o C:\Users\21548\AppData\Local\Temp\GoLand\___TestURLTop_in_talent.test.exe talent #gosetup
D:\Go\bin\go.exe tool test2json -t C:\Users\21548\AppData\Local\Temp\GoLand\___TestURLTop_in_talent.test.exe -test.v -test.panic
=== RUN TestURLTop
Case0 PASS, dataSize=1MB, nMapFiles=5, cost=3.1813ms
Case1 PASS, dataSize=1MB, nMapFiles=5, cost=4.4577ms
Case2 PASS, dataSize=1MB, nMapFiles=5, cost=4.7383ms
Case3 PASS, dataSize=1MB, nMapFiles=5, cost=50.6802ms
Case4 PASS, dataSize=1MB, nMapFiles=5, cost=76.4167ms
Case5 PASS, dataSize=1MB, nMapFiles=5, cost=3.8702ms
Case6 PASS, dataSize=1MB, nMapFiles=5, cost=2.0817ms
Case7 PASS, dataSize=1MB, nMapFiles=5, cost=3.4117ms
Case8 PASS, dataSize=1MB, nMapFiles=5, cost=9.235ms
Case9 PASS, dataSize=1MB, nMapFiles=5, cost=9.6456ms
Case10 PASS, dataSize=1MB, nMapFiles=5, cost=2.5858ms
Case0 PASS, dataSize=10MB, nMapFiles=10, cost=11.4476ms
Case1 PASS, dataSize=10MB, nMapFiles=10, cost=13.3249ms
Case2 PASS, dataSize=10MB, nMapFiles=10, cost=17.5994ms
Case3 PASS, dataSize=10MB, nMapFiles=10, cost=190.6756ms
Case4 PASS, dataSize=10MB, nMapFiles=10, cost=790.6544ms
Case5 PASS, dataSize=10MB, nMapFiles=10, cost=11.994ms
Case6 PASS, dataSize=10MB, nMapFiles=10, cost=11.5981ms
Case7 PASS, dataSize=10MB, nMapFiles=10, cost=11.7777ms
Case8 PASS, dataSize=10MB, nMapFiles=10, cost=68.1289ms
Case9 PASS, dataSize=10MB, nMapFiles=10, cost=66.2504ms
--- PASS: TestURLTop (66.2504ms)
PASS
```



```
Run: TestURLTop in talent
Tests passed: 1 of 1 test - 2 min 52 sec
Test Results
2 min 52 sec
Case2 PASS, dataSize=500MB, nMapFiles=40, cost=440.1210ms
Case3 PASS, dataSize=500MB, nMapFiles=40, cost=1.1879253s
Case4 PASS, dataSize=500MB, nMapFiles=40, cost=27.8187437s
Case5 PASS, dataSize=500MB, nMapFiles=40, cost=598.6624ms
Case6 PASS, dataSize=500MB, nMapFiles=40, cost=501.5077ms
Case7 PASS, dataSize=500MB, nMapFiles=40, cost=414.8169ms
Case8 PASS, dataSize=500MB, nMapFiles=40, cost=1.1329917s
Case9 PASS, dataSize=500MB, nMapFiles=40, cost=1.0802935s
Case10 PASS, dataSize=500MB, nMapFiles=40, cost=438.5379ms
Case0 PASS, dataSize=1GB, nMapFiles=60, cost=1.0573795s
Case1 PASS, dataSize=1GB, nMapFiles=60, cost=1.0985679s
Case2 PASS, dataSize=1GB, nMapFiles=60, cost=1.1045098s
Case3 PASS, dataSize=1GB, nMapFiles=60, cost=2.1980021s
Case4 PASS, dataSize=1GB, nMapFiles=60, cost=49.8670285s
Case5 PASS, dataSize=1GB, nMapFiles=60, cost=1.2579677s
Case6 PASS, dataSize=1GB, nMapFiles=60, cost=916.1647ms
Case7 PASS, dataSize=1GB, nMapFiles=60, cost=1.0717613s
Case8 PASS, dataSize=1GB, nMapFiles=60, cost=2.1698781s
Case9 PASS, dataSize=1GB, nMapFiles=60, cost=1.9969898s
Case10 PASS, dataSize=1GB, nMapFiles=60, cost=770.3076ms
--- PASS: TestURLTop (171.61s)
PASS
Process finished with the exit code 0
```

## 实验总结

实验过程中遇到的困难：

- 学习go并看懂提供的代码框架，这部分主要难点在于将框架的具体实现和Map-Reduce的理论框架结合起来，理解每段代码的含义和作用。
- 进行Map和Reduce函数优化的部分，首先要对Map-Reduce的原理有一个比较清楚认识，才能去发现原来函数中可以改进的地方，在弄懂原理之后，其实优化也就很直接简单了，原本的两层Map-Reduce其实可以直接删成一层，每一个Map Worker统计自己处理的Map File中url的出现次数，然后汇总给一个Reduce Worker进行统计和求Top 10就完成了整个过程。

对Map-Reduce计算框架的理解：

- Map-Reduce是Google提出的“三驾马车”之一，用于分布式系统的框架的大数据集并行计算。
- Map-Reduce的核心思想其实就是分而治之，将一个大问题拆成很多子问题，每个子问题独立的并行执行，最后再将结果汇总在一起得到最后的答案。
- Map-Reduce的具体实现思想是并行计算（本次lab使用到了go中的协程coroutine--轻量级线程），通过将不同任务动态规划给大规模计算集群中的计算单元，每个计算单元互不干扰的进行计算来实现整体的并行计算。
- Map-Reduce的本质是键值对的转换，Map-Reduce中一个很重要的概念叫做key-value pair，即我们所说的键值对，Map-Reduce通过Map和Reduce操作将键值对域进行了变换，最后依据键值对得到我们想要的结果，理解键值对是理解Map-Reduce的核心。