Lab0 实验报告

实验结果

1. 完成 Map-Reduce 框架

`make test_example` 的实验截图

```
paradox@wangzikais-MacBook-Pro lab0 % make test_example
go test -v -run=TestExampleURLTop
=== RUN   TestExampleURLTop
Case0 PASS, dataSize=1MB, nMapFiles=5, cost=52.687ms
Case1 PASS, dataSize=1MB, nMapFiles=5, cost=24.5995ms
Case2 PASS, dataSize=1MB, nMapFiles=5, cost=20.200541ms
Case3 PASS, dataSize=1MB, nMapFiles=5, cost=42.005708ms
Case4 PASS, dataSize=1MB, nMapFiles=5, cost=115.735292ms
Case5 PASS, dataSize=1MB, nMapFiles=5, cost=34.550209ms
Case6 PASS, dataSize=1MB, nMapFiles=5, cost=33.834209ms
Case7 PASS, dataSize=1MB, nMapFiles=5, cost=30.204208ms
Case8 PASS, dataSize=1MB, nMapFiles=5, cost=27.598209ms
Case9 PASS, dataSize=1MB, nMapFiles=5, cost=24.843542ms
Case10 PASS, dataSize=1MB, nMapFiles=5, cost=19.040167ms
Case0 PASS, dataSize=10MB, nMapFiles=10, cost=398.841834ms
Case1 PASS, dataSize=10MB, nMapFiles=10, cost=169.255667ms
Case2 PASS, dataSize=10MB, nMapFiles=10, cost=115.852833ms
Case3 PASS, dataSize=10MB, nMapFiles=10, cost=146.690542ms
Case4 PASS, dataSize=10MB, nMapFiles=10, cost=1.26527375s
Case5 PASS, dataSize=10MB, nMapFiles=10, cost=374.400583ms
Case6 PASS, dataSize=10MB, nMapFiles=10, cost=274.456167ms
Case7 PASS, dataSize=10MB, nMapFiles=10, cost=299.523125ms
Case8 PASS, dataSize=10MB, nMapFiles=10, cost=226.397209ms
Case9 PASS, dataSize=10MB, nMapFiles=10, cost=247.552792ms
Case10 PASS, dataSize=10MB, nMapFiles=10, cost=147.55875ms
Case0 PASS, dataSize=100MB, nMapFiles=20, cost=4.005174375s
Case1 PASS, dataSize=100MB, nMapFiles=20, cost=1.432735125s
Case2 PASS, dataSize=100MB, nMapFiles=20, cost=1.082351625s
Case3 PASS, dataSize=100MB, nMapFiles=20, cost=1.100222083s
Case4 PASS, dataSize=100MB, nMapFiles=20, cost=5.650520166s
Case5 PASS, dataSize=100MB, nMapFiles=20, cost=3.398213s
Case6 PASS, dataSize=100MB, nMapFiles=20, cost=3.392173s
Case7 PASS, dataSize=100MB, nMapFiles=20, cost=3.462572583s
Case8 PASS, dataSize=100MB, nMapFiles=20, cost=2.071815958s
Case9 PASS, dataSize=100MB, nMapFiles=20, cost=1.321319584s
Case10 PASS, dataSize=100MB, nMapFiles=20, cost=1.203850167s
Case0 PASS, dataSize=500MB, nMapFiles=40, cost=24.891221542s
Case1 PASS, dataSize=500MB, nMapFiles=40, cost=6.54918s
Case2 PASS, dataSize=500MB, nMapFiles=40, cost=5.457375417s
Case3 PASS, dataSize=500MB, nMapFiles=40, cost=5.480881083s
Case4 PASS, dataSize=500MB, nMapFiles=40, cost=9.407008334s
Case5 PASS, dataSize=500MB, nMapFiles=40, cost=20.261553625s
Case6 PASS, dataSize=500MB, nMapFiles=40, cost=17.588663375s
Case7 PASS, dataSize=500MB, nMapFiles=40, cost=16.9156345s
Case8 PASS, dataSize=500MB, nMapFiles=40, cost=10.009808417s
Case9 PASS, dataSize=500MB, nMapFiles=40, cost=11.800831083s
Case10 PASS, dataSize=500MB, nMapFiles=40, cost=6.030946125s
Case0 PASS, dataSize=1GB, nMapFiles=60, cost=50.790570916s
Case1 PASS, dataSize=1GB, nMapFiles=60, cost=14.697012333s
Case2 PASS, dataSize=1GB, nMapFiles=60, cost=11.562994083s
Case3 PASS, dataSize=1GB, nMapFiles=60, cost=10.818915292s
Case4 PASS, dataSize=1GB, nMapFiles=60, cost=15.730410416s
Case5 PASS, dataSize=1GB, nMapFiles=60, cost=38.323589625s
Case6 PASS, dataSize=1GB, nMapFiles=60, cost=33.125059834s
Case7 PASS, dataSize=1GB, nMapFiles=60, cost=34.894959541s
Case8 PASS, dataSize=1GB, nMapFiles=60, cost=23.247978416s
Case9 PASS, dataSize=1GB, nMapFiles=60, cost=24.74574375s
Case10 PASS, dataSize=1GB, nMapFiles=60, cost=12.015490542s
--- PASS: TestExampleURLTop (436.75s)
PASS
ok      talent  437.688s
```

2. 基于 Map-Reduce 框架编写 Map-Reduce 函数

`make test_homework` 的实验截图

```
paradox@wangzikais-MacBook-Pro lab0 % make test_homework
go test -v -run=TestURLTop
=== RUN   TestURLTop
[Case0 PASS, dataSize=1MB, nMapFiles=5, cost=10.971583ms
Case1 PASS, dataSize=1MB, nMapFiles=5, cost=14.750917ms
Case2 PASS, dataSize=1MB, nMapFiles=5, cost=9.628667ms
Case3 PASS, dataSize=1MB, nMapFiles=5, cost=32.351083ms
Case4 PASS, dataSize=1MB, nMapFiles=5, cost=87.872625ms
Case5 PASS, dataSize=1MB, nMapFiles=5, cost=6.316417ms
Case6 PASS, dataSize=1MB, nMapFiles=5, cost=5.041125ms
Case7 PASS, dataSize=1MB, nMapFiles=5, cost=5.205958ms
Case8 PASS, dataSize=1MB, nMapFiles=5, cost=13.143ms
Case9 PASS, dataSize=1MB, nMapFiles=5, cost=13.996667ms
Case10 PASS, dataSize=1MB, nMapFiles=5, cost=4.728333ms
Case0 PASS, dataSize=10MB, nMapFiles=10, cost=20.928667ms
Case1 PASS, dataSize=10MB, nMapFiles=10, cost=31.603458ms
Case2 PASS, dataSize=10MB, nMapFiles=10, cost=21.70575ms
Case3 PASS, dataSize=10MB, nMapFiles=10, cost=70.927042ms
Case4 PASS, dataSize=10MB, nMapFiles=10, cost=764.243584ms
Case5 PASS, dataSize=10MB, nMapFiles=10, cost=19.908958ms
Case6 PASS, dataSize=10MB, nMapFiles=10, cost=16.467209ms
Case7 PASS, dataSize=10MB, nMapFiles=10, cost=16.496792ms
Case8 PASS, dataSize=10MB, nMapFiles=10, cost=44.084ms
Case9 PASS, dataSize=10MB, nMapFiles=10, cost=48.660375ms
Case10 PASS, dataSize=10MB, nMapFiles=10, cost=18.712459ms
Case0 PASS, dataSize=100MB, nMapFiles=20, cost=164.047083ms
Case1 PASS, dataSize=100MB, nMapFiles=20, cost=143.6415ms
Case2 PASS, dataSize=100MB, nMapFiles=20, cost=131.904833ms
Case3 PASS, dataSize=100MB, nMapFiles=20, cost=245.820542ms
Case4 PASS, dataSize=100MB, nMapFiles=20, cost=3.809230625s
Case5 PASS, dataSize=100MB, nMapFiles=20, cost=138.614458ms
Case6 PASS, dataSize=100MB, nMapFiles=20, cost=124.436959ms
Case7 PASS, dataSize=100MB, nMapFiles=20, cost=118.554792ms
Case8 PASS, dataSize=100MB, nMapFiles=20, cost=190.904666ms
Case9 PASS, dataSize=100MB, nMapFiles=20, cost=190.94225ms
Case10 PASS, dataSize=100MB, nMapFiles=20, cost=138.849166ms
Case0 PASS, dataSize=500MB, nMapFiles=40, cost=787.889834ms
Case1 PASS, dataSize=500MB, nMapFiles=40, cost=673.339833ms
Case2 PASS, dataSize=500MB, nMapFiles=40, cost=631.4505ms
Case3 PASS, dataSize=500MB, nMapFiles=40, cost=967.835ms
Case4 PASS, dataSize=500MB, nMapFiles=40, cost=9.106934334s
Case5 PASS, dataSize=500MB, nMapFiles=40, cost=695.998625ms
Case6 PASS, dataSize=500MB, nMapFiles=40, cost=613.187ms
Case7 PASS, dataSize=500MB, nMapFiles=40, cost=670.167875ms
Case8 PASS, dataSize=500MB, nMapFiles=40, cost=779.668084ms
Case9 PASS, dataSize=500MB, nMapFiles=40, cost=770.839875ms
Case10 PASS, dataSize=500MB, nMapFiles=40, cost=652.656416ms
Case0 PASS, dataSize=1GB, nMapFiles=60, cost=1.297164333s
Case1 PASS, dataSize=1GB, nMapFiles=60, cost=1.331434084s
Case2 PASS, dataSize=1GB, nMapFiles=60, cost=1.341038542s
Case3 PASS, dataSize=1GB, nMapFiles=60, cost=1.864868875s
Case4 PASS, dataSize=1GB, nMapFiles=60, cost=15.589984541s
Case5 PASS, dataSize=1GB, nMapFiles=60, cost=1.241802792s
Case6 PASS, dataSize=1GB, nMapFiles=60, cost=1.15136475s
Case7 PASS, dataSize=1GB, nMapFiles=60, cost=1.308627083s
Case8 PASS, dataSize=1GB, nMapFiles=60, cost=1.492455041s
Case9 PASS, dataSize=1GB, nMapFiles=60, cost=1.453792666s
Case10 PASS, dataSize=1GB, nMapFiles=60, cost=1.359971625s
--- PASS: TestURLTop (52.59s)
PASS
ok      talent  53.982s
```

实验总结

MapReduce 框架

框架提供了 MapReduce Map 部分的实现，需要补充 Coordinator 和 Worker 里 Reduce 部分的代码。

主要遇到的问题是：

1. 一开始实现的时候没有注意到这个 MapReduce 框架是自动化多轮次的，每轮自动使用上一轮的结果，写好 MapReduce 框架以后才发现要在 round 之间传递中间结果。不过这个问题解决很简单，就是向 notify 传递一个中间结果文件名的列表即可，这个 notify channel 同时起到了向 test 函数通知任务完成的作用。

2. 困扰比较久的是执行时间问题，这个测试样例卡时间比较严，一开始用 8 核的 M1 MacBook Pro 测试会恰好在 1GB 数据量的 case9 超时，后面换了一台 8 核的 Intel 服务器同样超时，确定是需要进行一下优化。分析出主要耗时的部分是文件读写的 I/O 开销和 reduce 阶段的排序，I/O 开销没办法在框架里优化，所以尝试优化排序部分。比较简单的解决思路是不考虑实际应用，用一个大的 map 做合并。

   a. 一开始的方案是把 kv 先读到一个 `intermediate` 键值对数组里，之后调用 sort 根据 key 进行排序：

   ```
   intermediate := []KeyValue{}

   sort.Sort(ByKey(intermediate))
   ```

   b. 后续修改为直接把 kv 读进一个 map 里，再对 map 里的 key 根据字母序进行排序：

   ```
   intermediateMap := make(map[string][]string)
   ```

   这个方案减少了排序的开销，时间从完成 1GB 的 case9 用 600s 优化到完成全部测试耗时 437s。

URLTop10

这部分实验就是基于 example 自己写一个 URLTop10 的 Map 函数和 Reduce 函数。

主要的目标是减少中间的 I/O 开销。实现的思路是分别给两轮 map 加一个 combine 函数，先减少一部分 map 传递给 reduce 的数据量。

在第一轮 Count 阶段，增加了 combine，在每个 map 输出的子集里对 <key, 1> 中 key 相同的求总数：

```
combinedKvs := make([]KeyValue, 0)
for k, v := range clusteredKvs {
  combinedKvs = append(combinedKvs, KeyValue{Key: k, Value: strconv.Itoa(len(v))})
}
```

之后在 reduce 函数里把每个 map 文件里的相同 key 部分的 count 加和。这样减少了大量的中间结果文件读写开销。

做完这部分优化以后，测试时间从之前的 437s 优化到了 58s。

```
ok      talent  4.761s
[paradox@wangzikais-MacBook-Pro lab0 % make test_homework
go test -v -run=TestURLTop
=== RUN   TestURLTop
Case0 PASS, dataSize=1MB, nMapFiles=5, cost=21.499167ms
Case1 PASS, dataSize=1MB, nMapFiles=5, cost=15.942709ms
Case2 PASS, dataSize=1MB, nMapFiles=5, cost=8.865584ms
Case3 PASS, dataSize=1MB, nMapFiles=5, cost=39.180292ms
Case4 PASS, dataSize=1MB, nMapFiles=5, cost=120.441792ms
Case5 PASS, dataSize=1MB, nMapFiles=5, cost=5.626875ms
Case6 PASS, dataSize=1MB, nMapFiles=5, cost=5.526583ms
Case7 PASS, dataSize=1MB, nMapFiles=5, cost=4.968042ms
Case8 PASS, dataSize=1MB, nMapFiles=5, cost=17.318375ms
Case9 PASS, dataSize=1MB, nMapFiles=5, cost=18.058125ms
Case10 PASS, dataSize=1MB, nMapFiles=5, cost=5.273333ms
Case0 PASS, dataSize=10MB, nMapFiles=10, cost=24.063916ms
Case1 PASS, dataSize=10MB, nMapFiles=10, cost=24.54525ms
Case2 PASS, dataSize=10MB, nMapFiles=10, cost=24.311333ms
Case3 PASS, dataSize=10MB, nMapFiles=10, cost=88.016083ms
Case4 PASS, dataSize=10MB, nMapFiles=10, cost=1.242266541s
Case5 PASS, dataSize=10MB, nMapFiles=10, cost=21.412084ms
Case6 PASS, dataSize=10MB, nMapFiles=10, cost=16.867875ms
Case7 PASS, dataSize=10MB, nMapFiles=10, cost=17.395916ms
Case8 PASS, dataSize=10MB, nMapFiles=10, cost=58.408583ms
Case9 PASS, dataSize=10MB, nMapFiles=10, cost=50.883458ms
Case10 PASS, dataSize=10MB, nMapFiles=10, cost=17.066125ms
Case0 PASS, dataSize=100MB, nMapFiles=20, cost=164.448375ms
Case1 PASS, dataSize=100MB, nMapFiles=20, cost=137.327167ms
Case2 PASS, dataSize=100MB, nMapFiles=20, cost=136.49375ms
Case3 PASS, dataSize=100MB, nMapFiles=20, cost=262.379666ms
Case4 PASS, dataSize=100MB, nMapFiles=20, cost=5.586639s
Case5 PASS, dataSize=100MB, nMapFiles=20, cost=143.300042ms
Case6 PASS, dataSize=100MB, nMapFiles=20, cost=135.098416ms
Case7 PASS, dataSize=100MB, nMapFiles=20, cost=143.095334ms
Case8 PASS, dataSize=100MB, nMapFiles=20, cost=210.909833ms
Case9 PASS, dataSize=100MB, nMapFiles=20, cost=216.758917ms
Case10 PASS, dataSize=100MB, nMapFiles=20, cost=157.48475ms
Case0 PASS, dataSize=500MB, nMapFiles=40, cost=887.201666ms
Case1 PASS, dataSize=500MB, nMapFiles=40, cost=699.675625ms
Case2 PASS, dataSize=500MB, nMapFiles=40, cost=615.223667ms
Case3 PASS, dataSize=500MB, nMapFiles=40, cost=962.13825ms
Case4 PASS, dataSize=500MB, nMapFiles=40, cost=10.881082666s
Case5 PASS, dataSize=500MB, nMapFiles=40, cost=669.994083ms
Case6 PASS, dataSize=500MB, nMapFiles=40, cost=609.328416ms
Case7 PASS, dataSize=500MB, nMapFiles=40, cost=620.156666ms
Case8 PASS, dataSize=500MB, nMapFiles=40, cost=766.191417ms
Case9 PASS, dataSize=500MB, nMapFiles=40, cost=785.320208ms
Case10 PASS, dataSize=500MB, nMapFiles=40, cost=606.766875ms
Case0 PASS, dataSize=1GB, nMapFiles=60, cost=1.161555792s
Case1 PASS, dataSize=1GB, nMapFiles=60, cost=1.277308625s
Case2 PASS, dataSize=1GB, nMapFiles=60, cost=1.245427125s
Case3 PASS, dataSize=1GB, nMapFiles=60, cost=1.761732375s
Case4 PASS, dataSize=1GB, nMapFiles=60, cost=16.747195s
Case5 PASS, dataSize=1GB, nMapFiles=60, cost=1.35741125s
Case6 PASS, dataSize=1GB, nMapFiles=60, cost=1.188891458s
Case7 PASS, dataSize=1GB, nMapFiles=60, cost=1.308281208s
Case8 PASS, dataSize=1GB, nMapFiles=60, cost=1.411469958s
Case9 PASS, dataSize=1GB, nMapFiles=60, cost=1.496467291s
Case10 PASS, dataSize=1GB, nMapFiles=60, cost=1.4115425s
--- PASS: TestURLTop (57.69s)
PASS
ok      talent  58.603s
```

观察发现虽然大部分 case 都优化效果显著，但是所有数据量下的 case4 相比修改前会慢很多，查看数据分布发现 case4 主要是重复访问量很小的 url，最多也只有 3 次。所以 count 阶段对 map 数据合并不能减少很多中间结果数据量，反而增加了开销。

考虑在 top10 的 map 阶段再加一个 combine，这个 combine 直接对当前 map 函数的子结果求一次 top10，因为 round2 每个 map 函数里只处理 round1 一个 reduce 的输出，所以 key 和其他 map 是不重复的，reduce 再对每个 map 出的 top10 合并后的结果取 top10 不会影响正确性。得到最终的测试结果，相比第一次在 case4 都有优化，其他 case 因为 url 本来也不多，反而增加了一点开销，但可以接受。