

1 研究目的

探究MapReduce中Combiner对单词计数作业的影响。

2 研究内容

1. Combiner是否能够有效减少Shuffle阶段的数据量？
2. 在不同的key分布（均匀分布与数据倾斜）下，其性能提升效果有何差异？
3. 是否所有场景都适合使用Combiner？

3 实验

3.1 实验环境

硬件

集群一共含3个节点，分别是master节点foriel，slave节点u20、u22。

foriel节点：CPU核数16，内存大小15Gi，网络带宽1000Mb/s，存储类型SSD。

u20节点：CPU核数4，内存大小8.7Gi，网络带宽1000Mb/s，存储类型SSD。

u22节点：CPU核数16，内存大小7.5Gi，网络带宽1000Mb/s，存储类型SSD。

软件

foriel节点：Ubuntu24.04.2，JDK版本11.0.28，hadoop-3.4.2。

u20节点：Ubuntu24.04.2，JDK版本11.0.27，hadoop-3.4.2。

u22节点：Ubuntu22.04.5，JDK版本11.0.29，hadoop-3.4.2。

3.2 实验负载

准备两个数据集 `data_uniform.txt` 和 `data_skewed.txt`，前者为单词均匀分布的数据集，后者为单词严重倾斜分布的数据集，其中 `data_uniform.txt` 约627MB，`data_skewed.txt` 约595MB。

3.3 实验步骤

组长刘烨铭负责环境搭建部分，组员张笑铖负责实验部分：

1. 首先，刘烨铭搭建好一个Hadoop集群，选择刘烨铭的主机 foriel 作为master节点，张笑铖主机 u22 和陈岩松的虚拟机 u20 作为slave 节点。

```
leafriel@foriel:~$ jps
90753 JobHistoryServer
124982 ResourceManager
99819 NameNode
149289 Jps
leafriel@u20:~$ jps
24192 NodeManager
28076 Jps
20236 DataNode
leafriel@u22:~$ jps
23203 Jps
15368 DataNode
19468 NodeManager
```

其中，组长将三个节点的用户名都设为了leafriel，没有该账户的节点就新建账户leafriel，目的是隔离环境方便管理。

2. 张笑铖准备实验代码：登录foriel节点，创建文件WordCountExperiment.java（见code文件夹）并编译。

```
mkdir classes
javac -classpath $(hadoop classpath) -d classes WordCountExperiment.java
jar -cvf wc_experiment.jar -C classes/ .
```

3. 郑一钒把数据集和需要执行的指令发给张笑铖，张笑铖分别执行四个任务：均匀分布&无Combiner，均匀分布&有Combiner，数据倾斜&无Combiner，数据倾斜&有Combiner。等四个任务执行完毕后，连接Hadoop的Web UI (<http://foriel:8088>)，点击每个任务

History, 查看各自的Counters, 最后由 张笑铖 进行数据记录与分析。需要执行的指令如下:

```
cd ~  
hadoop jar wc_experiment.jar WordCountExperiment /input/uniform /output/uniform_no_comb false  
hadoop jar wc_experiment.jar WordCountExperiment /input/uniform /output/uniform_with_comb true  
hadoop jar wc_experiment.jar WordCountExperiment /input/skewed /output/skewed_no_comb false  
hadoop jar wc_experiment.jar WordCountExperiment /input/skewed /output/skewed_with_comb true
```

组员 郑一钒 负责准备数据部分:

生成均匀数据 (data_uniform.txt):

```
import random  
  
words = ["apple", "banana", "orange", "grape", "melon", "peach", "lemon", "cherry"]  
# 生成约 200MB - 500MB 的数据  
# 关键点: 每 10 个单词换一行  
with open("data_uniform.txt", "w") as f:  
    for i in range(1000000000):  
        f.write(f"{random.choice(words)} ")  
        if i % 10 == 0: # 每10个单词插入一个换行符  
            f.write("\n")  
print("Data generation complete.")
```

生成倾斜数据 (data_skewed.txt) :

```
import random  
  
words = ["apple", "banana", "orange", "grape", "melon", "peach", "lemon", "cherry"]  
# 目标: 生成数据倾斜文件  
# "apple" 出现的概率设为 90%, 模拟热点 Key  
# 必须加入换行符防止 OOM  
print("Generating skewed data ...")  
with open("data_skewed.txt", "w") as f:  
    for i in range(100000000): # 循环次数决定文件大小  
        if random.random() < 0.9:  
            f.write("apple ") # 制造倾斜  
        else:  
            f.write(f"{random.choice(words[1:])} ") # 其他单词  
        # 关键修正: 每 10 个单词换一行  
        if i % 10 == 0:  
            f.write("\n")  
print("Skewed data generation complete.")
```

最终MapReduce部署成功:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed	Elapsed Time
2025.11.25 14:05:57 CST	2025.11.25 14:07:23 CST	2025.11.25 14:07:48 CST	job_1764049838792_0006	word count experiment	leafriel	root.default	SUCCEEDED	5	5	1	1	00hrs, 00mins, 24sec
2025.11.25 14:05:50 CST	2025.11.25 14:05:52 CST	2025.11.25 14:07:12 CST	job_1764049838792_0005	word count experiment	leafriel	root.default	SUCCEEDED	5	5	1	1	00hrs, 01mins, 20sec
2025.11.25 14:04:12 CST	2025.11.25 14:04:15 CST	2025.11.25 14:04:38 CST	job_1764049838792_0004	word count experiment	leafriel	root.default	SUCCEEDED	5	5	1	1	00hrs, 00mins, 22sec
2025.11.25 14:01:26 CST	2025.11.25 14:01:30 CST	2025.11.25 14:02:53 CST	job_1764049838792_0003	word count experiment	leafriel	root.default	SUCCEEDED	5	5	1	1	00hrs, 01mins, 22sec

3.4 实验结果和分析

实验场景	启用 Combiner	执行时间 (ms)	Map Output Records	Combine Input Records	Combine Output Records	Reduce Shuffle Bytes	Reduce Input Records
均匀分布	否	135,010	100,000,000	0	0	1,237,504,455	100,000,000
均匀分布	是	49,730	100,000,000	100,000,272	312	525	40

实验场景	启用 Combiner	执行时间 (ms)	Map Output Records	Combine Input Records	Combine Output Records	Reduce Shuffle Bytes	Reduce Input Records
数据倾斜	否	127,560	100,000,000	0	0	1,204,287,500	100,000,000
数据倾斜	是	57,630	100,000,000	100,000,264	304	525	40

结果分析（上面四组实验记为组1-4，分析对应研究内容）：

1. 通过对比组1与组2（或组3与组4）可以发现，开启Combiner后，**Shuffle Bytes**（网络传输数据量）呈现数量级下降。原因：Combiner对输出的中间结果进行局部的聚合操作，因此能极大减少Map端写入磁盘的数据量以及Shuffle阶段占用的网络带宽，从而显著减轻Reducer的拷贝和归并压力。
2. 数据倾斜场景下（组4）的Reduce Shuffle Bytes达到了和均匀分布一样的最低值，均匀分布场景下的执行时间提升率（组2）略高于倾斜场景。原因：在均匀分布场景中，Combiner的聚合任务是分散且均衡的，避免了CPU资源的过度集中消耗。因此，Combiner带来的I/O节省效果得以更充分地转化为总执行时间的缩短。
3. 虽然实验显示Combiner效果显著，但并非所有场景都适用。

适用场景： Combiner的操作必须满足结合律和交换律。例如求和、求最大值、求最小值均适用。

不适用场景： 对于求平均值等操作，不能直接使用Combiner。例如Mapper1输出 $<10, 20>$ ，Mapper2输出 $<30, 40, 50>$ ，用Combiner直接求平均，Mapper1发送15，Mapper2发送40，Reducer最终得到 $(15+40)/2=27.5$ ，而真实平均值为 $(10+20+30+40+50)/5=30$ 。所以在使用Combiner前，必须严格验证算法逻辑是否允许在Map端进行局部预聚合。

3.5 结论

本次实验通过两个约600MB数据集在5个Mapper和1个Reducer的环境下验证了Combiner的机制。实验数据表明，Combiner能够有效减少Shuffle阶段的网络I/O和磁盘I/O，大幅提升作业执行效率。

3.6 分工

贡献度大小降序排列：

刘烨铭：搭建环境耗时最长。

张笑铖：记录实验结果和分析。

郑一钒：生成合适格式的数据集进行实验。

陈岩松：提供一台虚拟机。