《分布式编程模型与系统》期末考查作业

学号:10195501409　　　　姓名：卞思頔　　　　成绩:

1. 实验目的

验证 Spark 和 MapReduce 执行 PagerRank 迭代应用的性能差异

2. 设计的思路

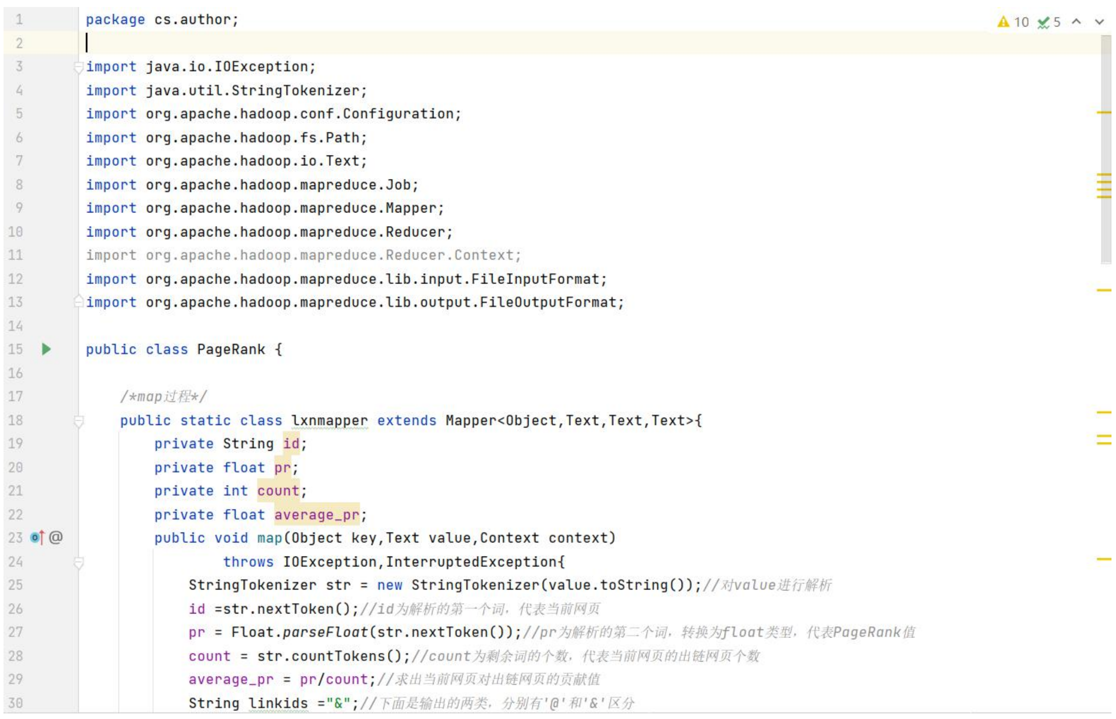
a) MapReduce 执行迭代计算过程中会反复读写 HDFS，因此可以在 HDFS 中观察到每一轮迭代的输出结果
b) MapReduce 会提交一系列的作业，而 spark 仅有一个应用，在 Yarn 的 UI 显示会不一样
c) 对于同样规模的数据集，spark 执行时间应当更短

3. 实验设置

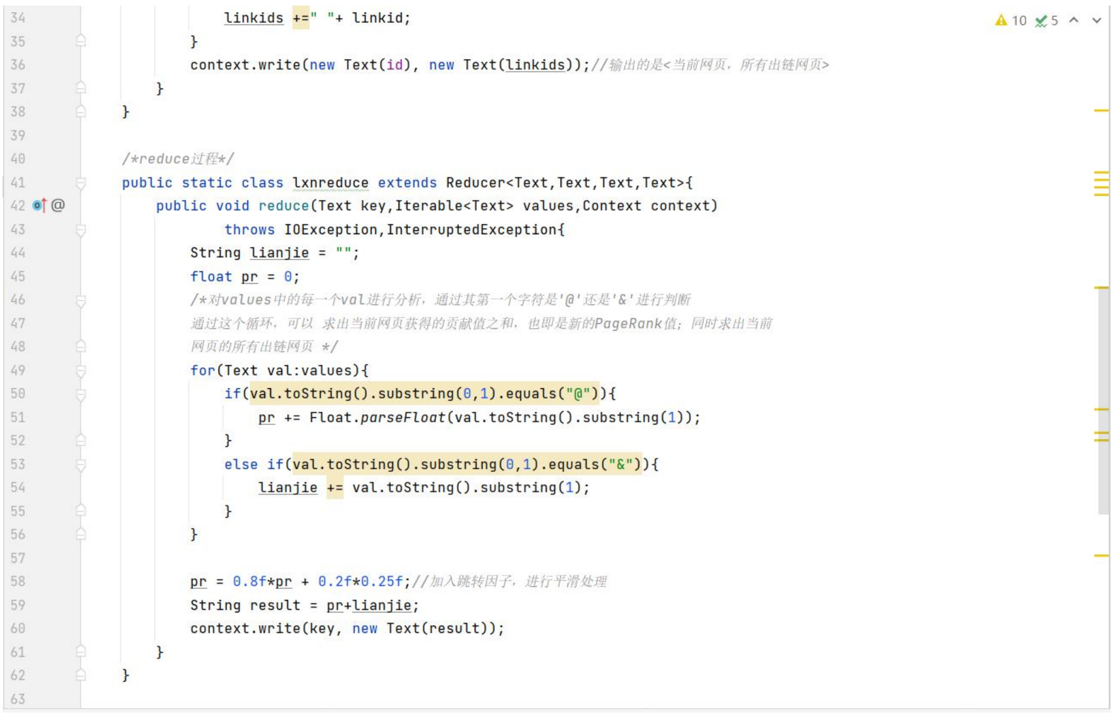
操作系统版本：Centos7，Hadoop3.0.0、Spark3.0.0
数据集名称：input.txt

4. 实验过程

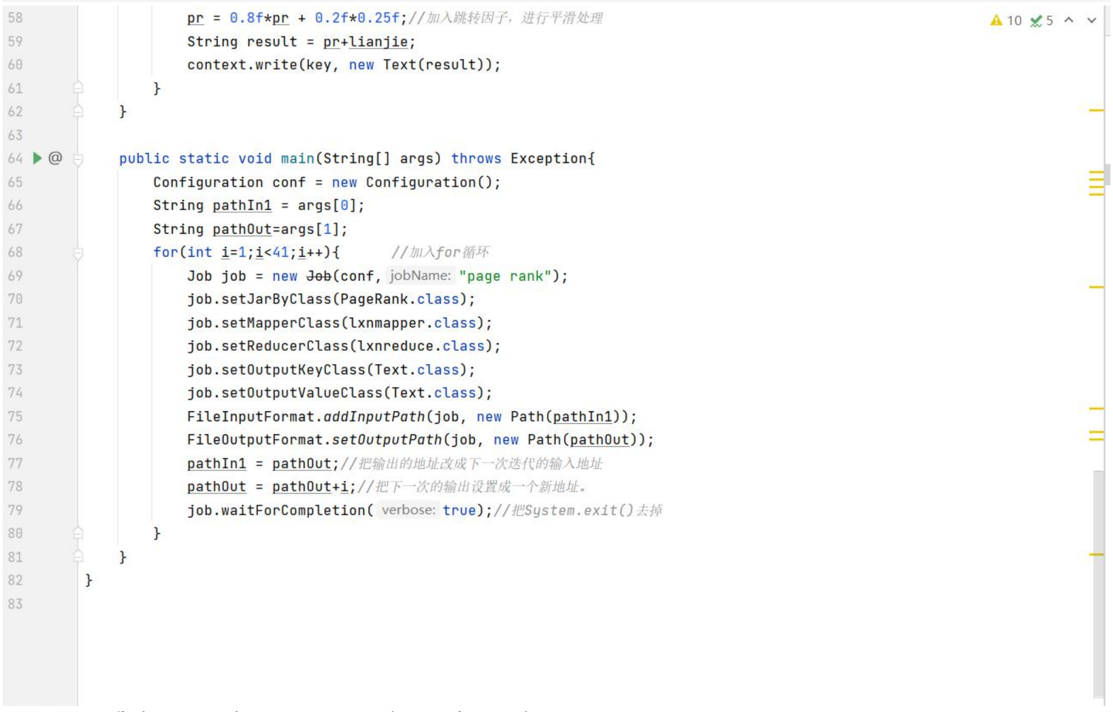
(1) 针对 MapReducer 我们实现一个 PageRank，并在 hadoop 集群上进行运行，进行 10 轮的迭代，查看运行的结果



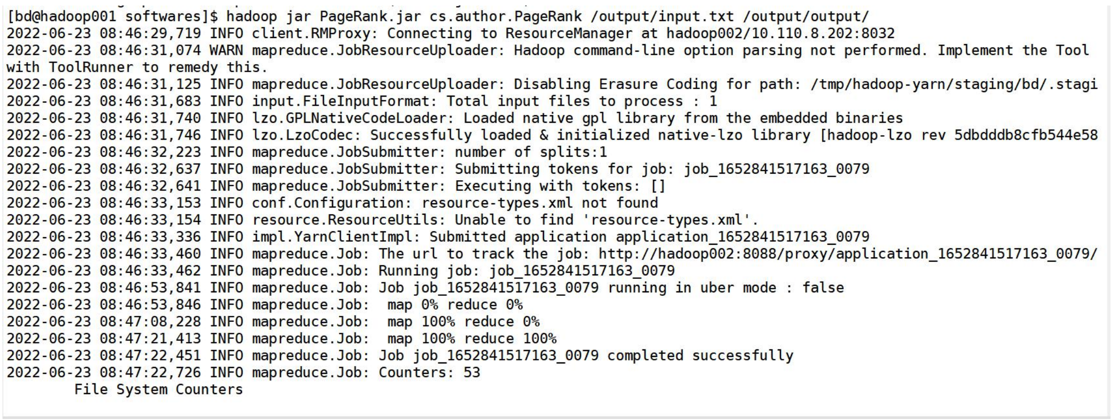
```java
package cs.author;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class PageRank {

    /*map过程*/
    public static class lxnmapper extends Mapper<Object,Text,Text,Text>{
        private String id;
        private float pr;
        private int count;
        private float average_pr;
        public void map(Object key,Text value,Context context)
                throws IOException,InterruptedException{
            StringTokenizer str = new StringTokenizer(value.toString());//对value进行解析
            id =str.nextToken();//id为解析的第一个词，代表当前网页
            pr = Float.parseFloat(str.nextToken());//pr为解析的第二个词，转换为float类型，代表PageRank值
            count = str.countTokens();//count为剩余词的个数，代表当前网页的出链网页个数
            average_pr = pr/count;//求出当前网页对出链网页的贡献值
            String linkids ="&";//下面是输出的两类，分别有'@'和'&'区分
```



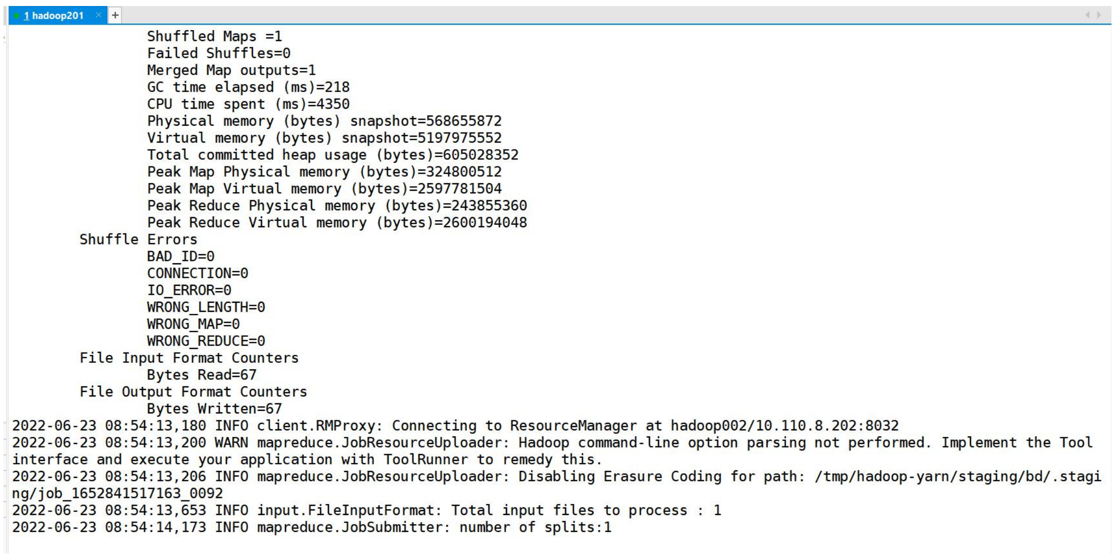
```java
                linkids +=" "+ linkid;
            }
            context.write(new Text(id), new Text(linkids));//输出的是<当前网页, 所有出链网页>
        }
    }

    /*reduce过程*/
    public static class lxnreduce extends Reducer<Text,Text,Text,Text>{
        public void reduce(Text key,Iterable<Text> values,Context context)
                throws IOException,InterruptedException{
            String lianjie = "";
            float pr = 0;
            /*对values中的每一个val进行分析，通过其第一个字符是'@'还是'&'进行判断
            通过这个循环，可以 求出当前网页获得的贡献值之和，也即是新的PageRank值；同时求出当前
            网页的所有出链网页 */
            for(Text val:values){
                if(val.toString().substring(0,1).equals("@")){
                    pr += Float.parseFloat(val.toString().substring(1));
                }
                else if(val.toString().substring(0,1).equals("&")){
                    lianjie += val.toString().substring(1);
                }
            }

            pr = 0.8f*pr + 0.2f*0.25f;//加入跳转因子，进行平滑处理
            String result = pr+lianjie;
            context.write(key, new Text(result));
        }
    }

```



```java
            pr = 0.8f*pr + 0.2f*0.25f;//加入跳转因子，进行平滑处理
            String result = pr+lianjie;
            context.write(key, new Text(result));
        }
    }

    public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();
        String pathIn1 = args[0];
        String pathOut=args[1];
        for(int i=1;i<41;i++){      //加入for循环
            Job job = new Job(conf, jobName: "page rank");
            job.setJarByClass(PageRank.class);
            job.setMapperClass(lxnmapper.class);
            job.setReducerClass(lxnreduce.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(Text.class);
            FileInputFormat.addInputPath(job, new Path(pathIn1));
            FileOutputFormat.setOutputPath(job, new Path(pathOut));
            pathIn1 = pathOut;//把输出的地址改成下一次迭代的输入地址
            pathOut = pathOut+i;//把下一次的输出设置成一个新地址。
            job.waitForCompletion( verbose: true);//把System.exit()去掉
        }
    }
}
```



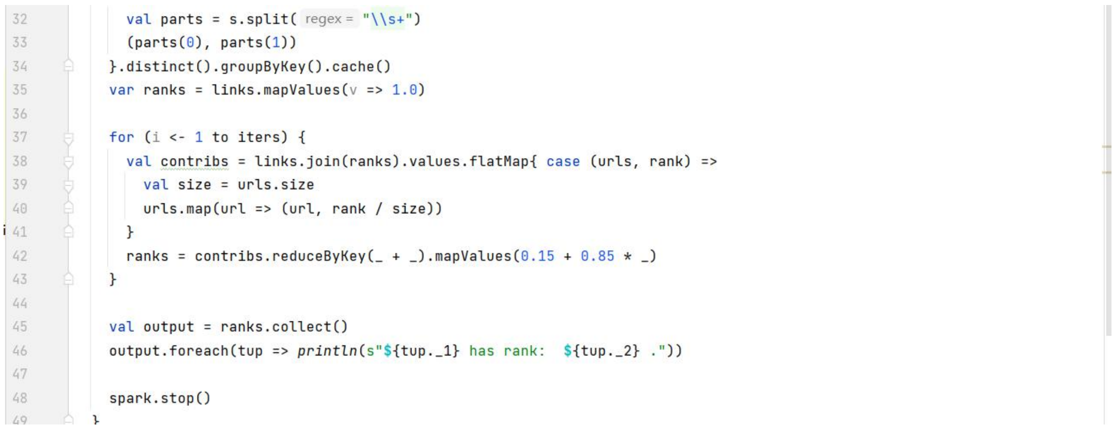
```
[bd@hadoop001 softwares]$ hadoop jar PageRank.jar cs.author.PageRank /output/input.txt /output/output/
2022-06-23 08:46:29,719 INFO client.RMProxy: Connecting to ResourceManager at hadoop002/10.110.8.202:8032
2022-06-23 08:46:31,074 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
with ToolRunner to remedy this.
2022-06-23 08:46:31,125 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/bd/.stagi
2022-06-23 08:46:31,683 INFO input.FileInputFormat: Total input files to process : 1
2022-06-23 08:46:31,740 INFO lzo.GPLNativeCodeLoader: Loaded native gpl library from the embedded binaries
2022-06-23 08:46:31,746 INFO lzo.LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev 5dbdddb8cfb544e58
2022-06-23 08:46:32,223 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-23 08:46:32,637 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1652841517163_0079
2022-06-23 08:46:32,641 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-23 08:46:33,153 INFO conf.Configuration: resource-types.xml not found
2022-06-23 08:46:33,154 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-23 08:46:33,336 INFO impl.YarnClientImpl: Submitted application application_1652841517163_0079
2022-06-23 08:46:33,460 INFO mapreduce.Job: The url to track the job: http://hadoop002:8088/proxy/application_1652841517163_0079/
2022-06-23 08:46:33,462 INFO mapreduce.Job: Running job: job_1652841517163_0079
2022-06-23 08:46:53,841 INFO mapreduce.Job: Job job_1652841517163_0079 running in uber mode : false
2022-06-23 08:46:53,846 INFO mapreduce.Job:  map 0% reduce 0%
2022-06-23 08:47:08,228 INFO mapreduce.Job:  map 100% reduce 0%
2022-06-23 08:47:21,413 INFO mapreduce.Job:  map 100% reduce 100%
2022-06-23 08:47:22,451 INFO mapreduce.Job: Job job_1652841517163_0079 completed successfully
2022-06-23 08:47:22,726 INFO mapreduce.Job: Counters: 53
        File System Counters
```



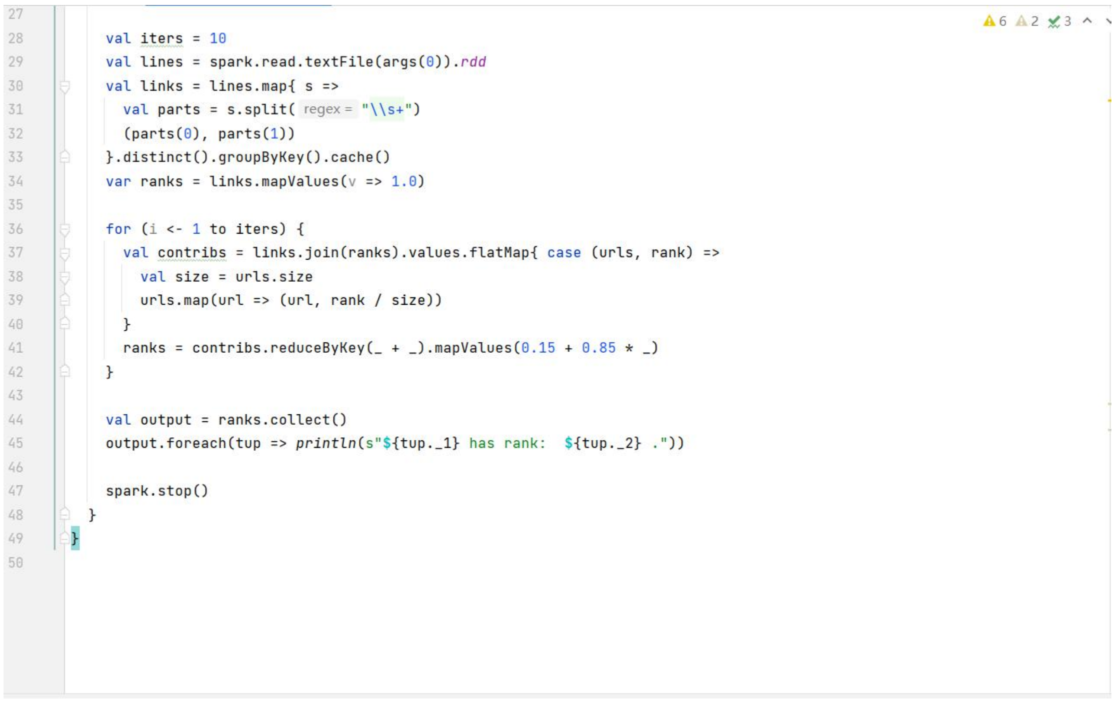
```
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=218
                CPU time spent (ms)=4350
                Physical memory (bytes) snapshot=568655872
                Virtual memory (bytes) snapshot=5197975552
                Total committed heap usage (bytes)=605028352
                Peak Map Physical memory (bytes)=324800512
                Peak Map Virtual memory (bytes)=2597781504
                Peak Reduce Physical memory (bytes)=243855360
                Peak Reduce Virtual memory (bytes)=2600194048
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=67
        File Output Format Counters
                Bytes Written=67
2022-06-23 08:54:13,180 INFO client.RMProxy: Connecting to ResourceManager at hadoop002/10.110.8.202:8032
2022-06-23 08:54:13,200 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
interface and execute your application with ToolRunner to remedy this.
2022-06-23 08:54:13,206 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/bd/.stagi
ng/job_1652841517163_0092
2022-06-23 08:54:13,653 INFO input.FileInputFormat: Total input files to process : 1
2022-06-23 08:54:14,173 INFO mapreduce.JobSubmitter: number of splits:1
```

可以看到 10 次迭代 Mapreduce 花费了 480s 时间

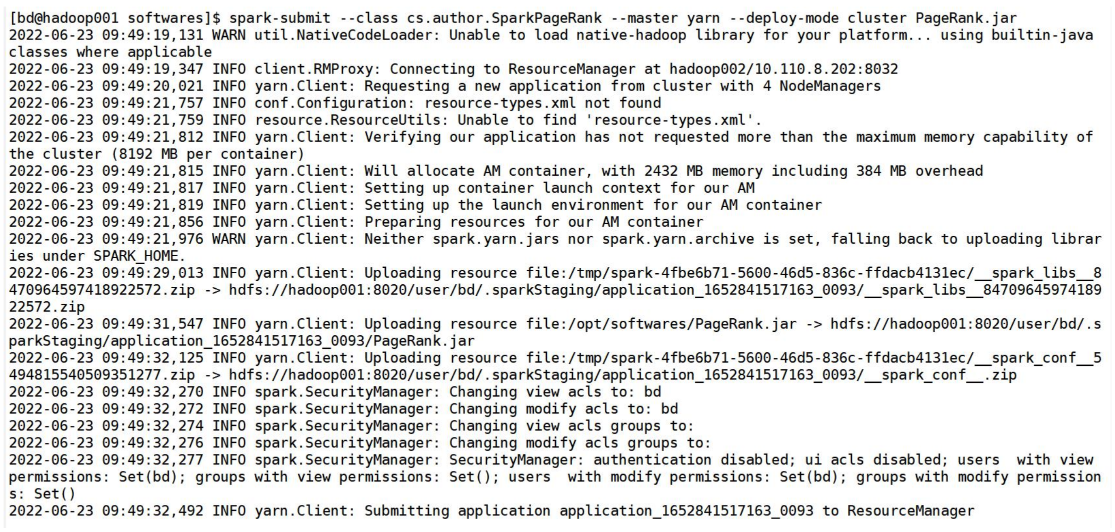
(2) 针对 Spark 我们实现一个 PageRank，并在 HDFS 上运行，在 Yarn 的 UI 上查看运行的时间和结果



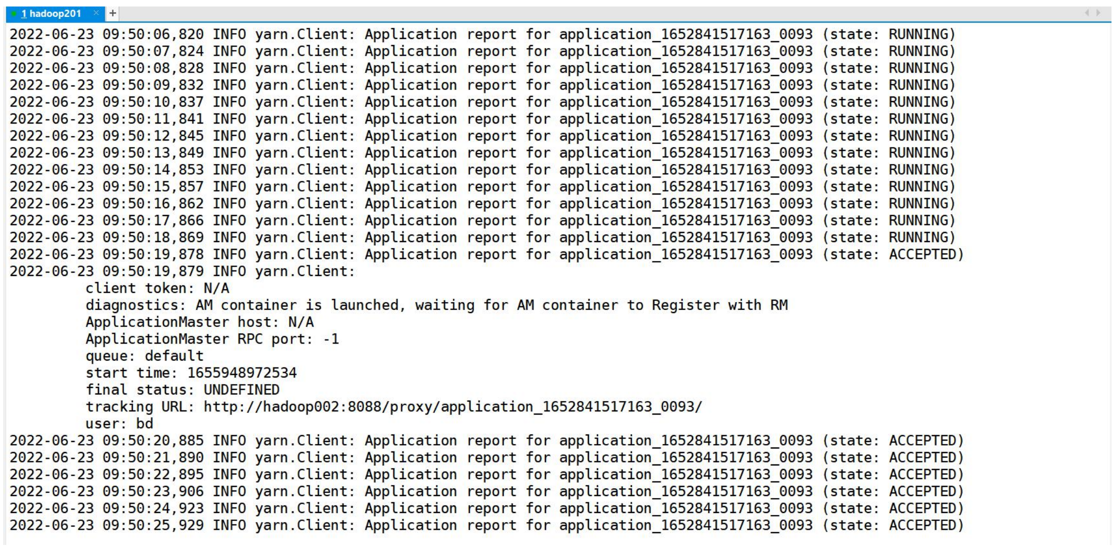
```
      val parts = s.split( regex = "\\s+")
      (parts(0), parts(1))
    }.distinct().groupByKey().cache()
    var ranks = links.mapValues(v => 1.0)

    for (i <- 1 to iters) {
      val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =>
        val size = urls.size
        urls.map(url => (url, rank / size))
      }
      ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
    }

    val output = ranks.collect()
    output.foreach(tup => println(s"${tup._1} has rank:  ${tup._2} ."))

    spark.stop()
  }
```



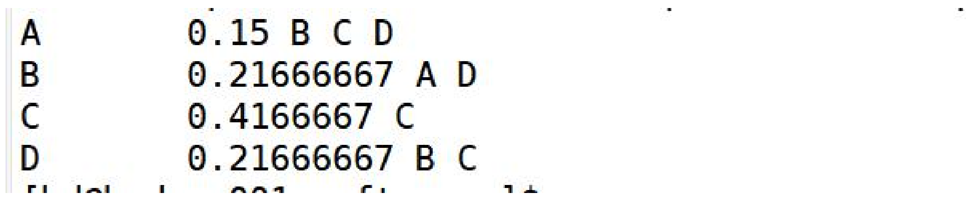
```scala
    val iters = 10
    val lines = spark.read.textFile(args(0)).rdd
    val links = lines.map{ s =>
      val parts = s.split( regex = "\\s+")
      (parts(0), parts(1))
    }.distinct().groupByKey().cache()
    var ranks = links.mapValues(v => 1.0)

    for (i <- 1 to iters) {
      val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =>
        val size = urls.size
        urls.map(url => (url, rank / size))
      }
      ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
    }

    val output = ranks.collect()
    output.foreach(tup => println(s"${tup._1} has rank:  ${tup._2} ."))

    spark.stop()
  }
}
```



```
[bd@hadoop001 softwares]$ spark-submit --class cs.author.SparkPageRank --master yarn --deploy-mode cluster PageRank.jar
2022-06-23 09:49:19,131 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
2022-06-23 09:49:19,347 INFO client.RMProxy: Connecting to ResourceManager at hadoop002/10.110.8.202:8032
2022-06-23 09:49:20,021 INFO yarn.Client: Requesting a new application from cluster with 4 NodeManagers
2022-06-23 09:49:21,757 INFO conf.Configuration: resource-types.xml not found
2022-06-23 09:49:21,759 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-23 09:49:21,812 INFO yarn.Client: Verifying our application has not requested more than the maximum memory capability of
the cluster (8192 MB per container)
2022-06-23 09:49:21,815 INFO yarn.Client: Will allocate AM container, with 2432 MB memory including 384 MB overhead
2022-06-23 09:49:21,817 INFO yarn.Client: Setting up container launch context for our AM
2022-06-23 09:49:21,819 INFO yarn.Client: Setting up the launch environment for our AM container
2022-06-23 09:49:21,856 INFO yarn.Client: Preparing resources for our AM container
2022-06-23 09:49:21,976 WARN yarn.Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading librar
ies under SPARK_HOME.
2022-06-23 09:49:29,013 INFO yarn.Client: Uploading resource file:/tmp/spark-4fbe6b71-5600-46d5-836c-ffdacb4131ec/__spark_libs__8
470964597418922572.zip -> hdfs://hadoop001:8020/user/bd/.sparkStaging/application_1652841517163_0093/__spark_libs__84709645974189
22572.zip
2022-06-23 09:49:31,547 INFO yarn.Client: Uploading resource file:/opt/softwares/PageRank.jar -> hdfs://hadoop001:8020/user/bd/.s
parkStaging/application_1652841517163_0093/PageRank.jar
2022-06-23 09:49:32,125 INFO yarn.Client: Uploading resource file:/tmp/spark-4fbe6b71-5600-46d5-836c-ffdacb4131ec/__spark_conf__5
494815540509351277.zip -> hdfs://hadoop001:8020/user/bd/.sparkStaging/application_1652841517163_0093/__spark_conf__.zip
2022-06-23 09:49:32,270 INFO spark.SecurityManager: Changing view acls to: bd
2022-06-23 09:49:32,272 INFO spark.SecurityManager: Changing modify acls to: bd
2022-06-23 09:49:32,274 INFO spark.SecurityManager: Changing view acls groups to:
2022-06-23 09:49:32,276 INFO spark.SecurityManager: Changing modify acls groups to:
2022-06-23 09:49:32,277 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view
permissions: Set(bd); groups with view permissions: Set(); users  with modify permissions: Set(bd); groups with modify permission
s: Set()
2022-06-23 09:49:32,492 INFO yarn.Client: Submitting application application_1652841517163_0093 to ResourceManager
```



```
2022-06-23 09:50:06,820 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:07,824 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:08,828 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:09,832 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:10,837 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:11,841 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:12,845 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:13,849 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:14,853 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:15,857 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:16,862 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:17,866 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:18,869 INFO yarn.Client: Application report for application_1652841517163_0093 (state: RUNNING)
2022-06-23 09:50:19,878 INFO yarn.Client: Application report for application_1652841517163_0093 (state: ACCEPTED)
2022-06-23 09:50:19,879 INFO yarn.Client:
         client token: N/A
         diagnostics: AM container is launched, waiting for AM container to Register with RM
         ApplicationMaster host: N/A
         ApplicationMaster RPC port: -1
         queue: default
         start time: 1655948972534
         final status: UNDEFINED
         tracking URL: http://hadoop002:8088/proxy/application_1652841517163_0093/
         user: bd
2022-06-23 09:50:20,885 INFO yarn.Client: Application report for application_1652841517163_0093 (state: ACCEPTED)
2022-06-23 09:50:21,890 INFO yarn.Client: Application report for application_1652841517163_0093 (state: ACCEPTED)
2022-06-23 09:50:22,895 INFO yarn.Client: Application report for application_1652841517163_0093 (state: ACCEPTED)
2022-06-23 09:50:23,906 INFO yarn.Client: Application report for application_1652841517163_0093 (state: ACCEPTED)
2022-06-23 09:50:24,923 INFO yarn.Client: Application report for application_1652841517163_0093 (state: ACCEPTED)
2022-06-23 09:50:25,929 INFO yarn.Client: Application report for application_1652841517163_0093 (state: ACCEPTED)
```



```
A        0.15 B C D
B        0.21666667 A D
C        0.4166667 C
D        0.21666667 B C
```

可以发现 spark 执行了 100s 左右，

(3) 查看 MapReducer 和 Spark 运行的时间的差异，得出结论。

5. 结论

在运行中我们可以发现，在迭代 10 次的情况下，Spark 的运行时间为 Mapreduce 的仅为 1/4 到 1/5，如果增加迭代次数或者增加数据量，则运行时间能更加大大缩减，spark 能对 mapreduce 存在极大的性能优势。

mapreduce 是基于磁盘进行计算，与磁盘存在大量的磁盘 IO 交换，而 Spark 是基于内存计算，基于 DAG 的任务调度机制,则能大大提升计算的速度和效率。

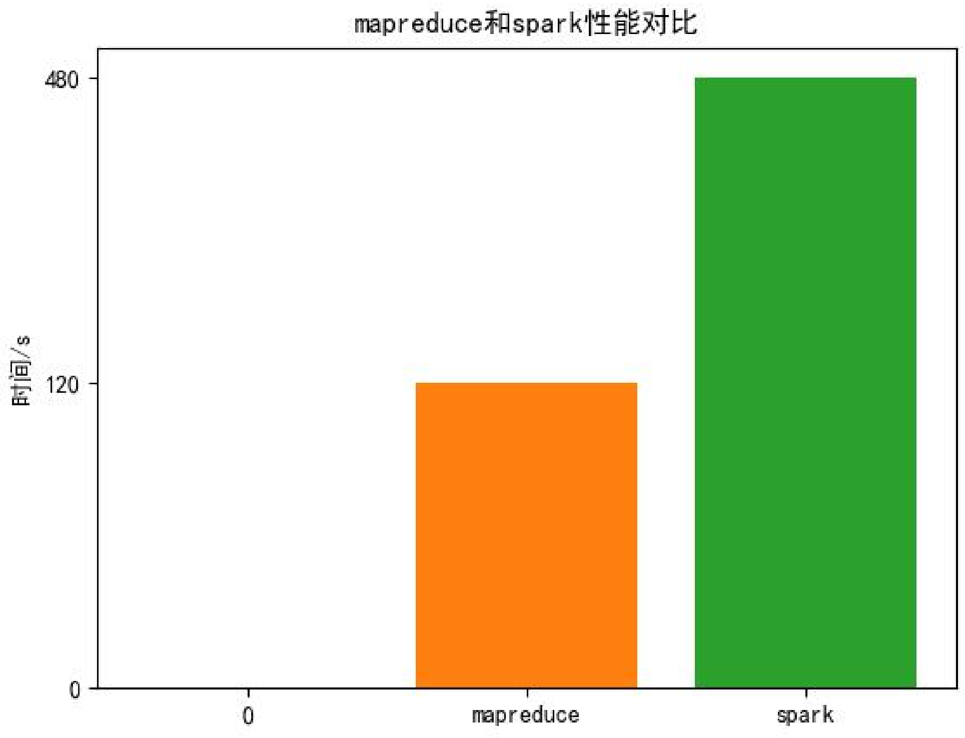
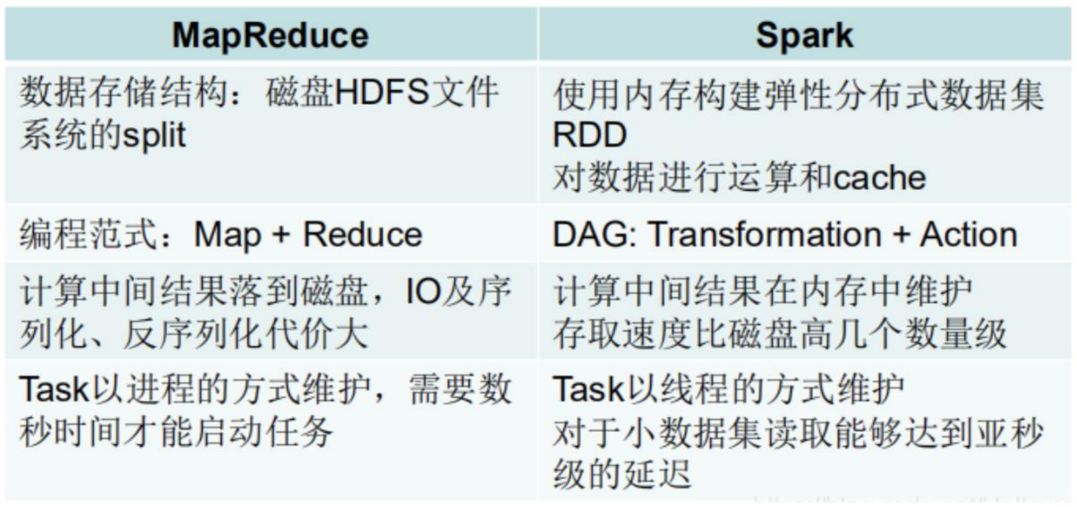


图 1 mapreduce 和 spark 进行 pagerank 运算时间对比



MapReduce	Spark
数据存储结构：磁盘HDFS文件系统的split	使用内存构建弹性分布式数据集RDD 对数据进行运算和cache
编程范式：Map + Reduce	DAG: Transformation + Action
计算中间结果落到磁盘，IO及序列化、反序列化代价大	计算中间结果在内存中维护 存取速度比磁盘高几个数量级
Task以进程的方式维护，需要数秒时间才能启动任务	Task以线程的方式维护 对于小数据集读取能够达到亚秒级的延迟

图 2　mapreduce 和 spark 进行运算时的区别

附录：

实验代码地址：

https://github.com/Emma-0129/Distributed-System-Experiment