作业十一：Hadoop MapReduce

519021910913 黄喆敏

配置过程

根据WordCount 2.0的方法加以变化，MR作业支持两种模式。一种不加-key参数，为统计所有单词的个数；另一种加-key参数，需要加一个关键词文件，统计所有关键词的个数。

- Setup

```
@Override
public void setup(Context context) throws IOException,
    InterruptedException {
  conf = context.getConfiguration();
  caseSensitive = conf.getBoolean("wordcount.case.sensitive", false); //设置对大小写不敏感
  if (conf.getBoolean("wordcount.keyword.patterns", false)) {
    keywordMode = true;
    URI[] keywords = Job.getInstance(conf).getCacheFiles();
    for (URI keyword : keywords) {
      Path keywordPath = new Path(keyword.getPath());
      String keywordFileName = keywordPath.getName().toString();
      parseKeywordFile(keywordFileName);
    }
  }
}

private void parseKeywordFile(String fileName) {
      try {
        fis = new BufferedReader(new FileReader(fileName));
        String pattern = null;
        while ((pattern = fis.readLine()) != null) {
          keywordSet.add(pattern);
        }
      } catch (IOException ioe) {
        System.err.println("Caught exception while parsing the cached file '"
            + StringUtils.stringifyException(ioe));
      }
    }
```

  - 关键词文件被放置在cache file中。开始时，程序调用`wordcount.keyword.patterns`配置。若为真，则代表为关键词模式，读取关键词文件中的关键词列表，加入keywordSet中；反之，则不是关键词模式，不进行上述操作。

- Mapper

```
  @Override
```

```
  public void map(Object key, Text value, Context context
  ) throws IOException, InterruptedException {
    String line = (caseSensitive) ?
        value.toString() : value.toString().toLowerCase();  //转化为小写

    line = line.replaceAll("[0-9]+", "");
    line = line.replaceAll("\"", "");
    line = line.replaceAll("[(),:.?]", "");  //使用正则表达式，去除数字、引号、括号等字符

    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      String nowToken = itr.nextToken();
      if ((keywordMode && keywordSet.contains(nowToken)) || !keywordMode) {
        word.set(nowToken);
        context.write(word, one);
        Counter counter = context.getCounter(CountersEnum.class.getName(),
            CountersEnum.INPUT_WORDS.toString());
        counter.increment(1);
      }
    }
  }
}
```

  - Mapper思路与word count基本保持一致，加了一些预处理操作。首先将string全部转化为小写，并利用正则表达式，去除数字、引号、括号等字符。
  - 若为关键词模式，则判断当前token是否在关键词列表中。若在关键词列表中，则写入context；若不是关键词模式，直接写入context即可。
- Reducer/ Combiner

  Reducer/ Combiner与word count保持一致，不再赘述。

运行结果

- 运行命令时，第一个参数为输入文件夹，第二个参数为输出文件夹，第三个参数为关键词文件。所有图书简介文件均放在输入文件夹中。
- 运行命令`input output -key keyword.txt`，结果如下所示，符合预期。

```
architecture  2
bestselling 5
capitalism  5
chinese 6
computer  8
economic  15
language  10
philosopher 3
political 4
politics  2
principles  8
revolution  4
sociological  4
software  10
students  8
systems 14
techniques  4
university  6
```

参数设置

本次任务的数据规模较小，数据总和没有超过1MB，因此没有进行特殊的参数设置。

Mapper & Reducer 数量

- 在进行map计算前，MapReduce会根据输入文件计算输入分片(Input Split),每个输入分片针对一个map任务，输入分片存储的并非数据本身，而是一个分片长度和一个记录数据位置的数组。在Hadoop2.x中默认的block大小是128M，在1.x中默认的大小是64M。
- 计算map个数时，我们需要考虑以下三个因素：
  - blockSize：HDFS的文件块大小
  - totalSize：输入文件大小
  - inputFileNum：输入文件的数量
- 如果不进行任何设置，默认的map个数是和blockSize相关的，defaultNum=totalSize/blockSize。但是需要注意，**MapReduce的每一个map处理数据是不能跨越文件的**，即minMapNum>=inputFileNum。因此，**mapNum=max(totalSize/blockSize, inputFileNum)**。
- 由于我们每个文件的大小均不超过10k，因此Mapper数量由文件个数决定，即Mapper有5个。
- 我们在Mapper中加入构造函数，验证Mapper的数量，结果如下所示，与前文所述相符。

```
/Users/xtommy/Library/Java/JavaVirtualMachines/azul-13.0.6/Contents/Home/bin/java ...
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Mapper called
Mapper called
Mapper called
Mapper called
Mapper called
```

- 对于Reducer，默认情况下只有一个Reducer，也可以在程序中设定Reducer的个数`job.setNumReduceTasks(number)`。因此，**Reducer的个数为1个**。

Reference

[1] https://www.jianshu.com/p/a9f6d3190606