**PAIR AND STRIPES**

1. When executing any variant of your WordCount job using the input file laboratory/gutenberg\_big.txt, how many **map tasks** are launched?

There are 79 map tasks are launched in this pjob

1. How does the number of reducers affect performance?

Increase network overhead. It waste the time to shuffle and arrange to reducers. **<?>**

|  |  |
| --- | --- |
| Number of reducers | Run time Reducer |
| 3 | 1365282 ms |
| 10 | 1945380 ms |
| 100 | 3040676 ms |

1. How many reducers can be executed in parallel?

Lower the cost of failure and number of reducers can run parallel depends on #CPU and #Cores in cluster. We don’t know how many reducer exactly run in parallel. **<?>**

1. Use the JobHistory web interface to examine job counters for all three variants of your WordCount job: can you explain the differences among them?
   * **[Hint]** For example, look at the amount of bytes shuffled, but also try to spot other differences

|  |  |  |  |
| --- | --- | --- | --- |
|  | Basic | IMC | Combiner |
| Map output record | 1779854067 | 42257238 | 42257238 |
| Reduce shuffle bytes | 1385612100 | 377101925 | 377101925 |
| CPU spent (ms) | 6221130 | 826430 | 826430 |
| Total time spent by all maps in occupied slots (ms) | 4915987 | 940474 | 940474 |
| Total time spent by all reducers in occupied slots (ms) | 1365282 | 85831 | 85831 |

1. Can you explain how does the distribution of words affect your Job?
   * **[Hint]** You should look at any skew in the distribution of execution times of your tasks.

* If the word appear frequently in corpus, it makes imbalance between reducers (load is received by reducer).
* Reducer receive high-load finish later reducer receive low-load => that affects the whole system, low-load reducer wait high-load reducer to finish the job

Straggler

#### **Questions (Co-occurrence)**

Answer the following questions (in a simple text file):

1. How does the number of reducer influence the behavior of the Pairs approach?

* With 1 reducer, the job took 4hrs, 30mins, 58sec. Whereas with 2 reducers, it took 3hrs, 15mins, 59sec. The number of reducers should be more than one to accomplish the job faster but there surely should be a maximum number of reducers so that time isn’t wasted through transmitting data in the network. You can see from result, using double reducers doesn’t mean reducing running time by two

1. Why does TextPair need to be Comparable?

* In stage Shuffle and Partition, key from mapper need to be arranged to reducers. The policy of this process is all records have same key have to be moved to a same reducer.

1. Can you use the implemented reducers as *Combiners*?

* Yes, the operation sum is distributive that not affects to logic of the algorithm

1. How many output bytes are spilled by the mappers to produce intermediate files? Keep this value in mind and compare to the "stripes" approach, next.

* Map output bytes: 375966464374

### **Questions (Stripes)**

1. Can you use the implemented reducers as *Combiner*?

* In my version, it can’t be because output of those process is different

1. Do you think Stripes could be used with the in-memory combiner pattern?

* You can.

1. How does the number of reducer influence the behavior of the Stripes approach?

|  |  |  |
| --- | --- | --- |
|  | 1 Reducer | 2 Reducers |
| Average Shuffle Time | 16mins, 40sec | 17mins, 34sec |
| Average Merge Time | 8mins, 10sec | 4mins, 45sec |
| Average Reduce Time | 3hrs, 36mins, 2sec | 2hrs, 1mins, 18sec |

1. Why StringToIntMapWritable is not Comparable (differently from TextPair)?

* It’s the value part, it means we don’t need compare or sorted those value so that does not affect to shuffle and partition phrase.

1. Using the JobHistory Web Interface, compare the shuffle phase of *Pair* and *Stripes* design patterns. How many output bytes are spilled by the mappers to produce intermediate files?

* Due to the number of map output bytes difference, Pairs take more time (Average Shuffle Time 27mins, 50sec ) for shuffle than Stripes (Average Shuffle Time 17mins, 34sec)
  + Map output bytes: 196437791855 for Stripes
  + Map output bytes: 375966464374 for Pairs

**ORDER INVERSION**

1. Do you think the Order Inversion approach is 'faster' than a naive approach with multiple jobs? Think about a compound job in which you compute the numerator and the denominator separately, and then perform the computation of the relative frequency

* Yes. It is faster. Perform multiple job will costs I/O disk to write intermediate output for another job

1. What is the impact of the use of a 'special' compound key on the amounts of shuffled bytes?

* Without combiner, amounts of shuffled increase 2 times. But with combiner but it is not noticeable.

1. How does the default partitioner works with TextPair? Can you imagine a different implementation that does not change the Partitioner?

* As default, it will use both 2 Textpair to hash, but we want all records with same first key in reducer so we only use first key to hash. We can use STRIPES approach to not change Partitioner

1. For each key, the reducer receives its marginal before the co-occurrence with the other words. Why?

* Because we need to count number of occurrence of that word first. In sorting, we set priority of ASTERISK higher than other word.