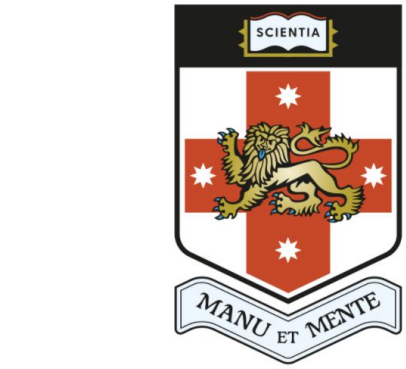


UNSW
SYDNEY

COMP9313 Project 4 : Optimization Report

Li Yu z3492782

1. Method for the project :

- Prefix filtering: minimise the number of prefix items emitted from the mappers

$$p = |record| - ceil(|record| * t) + 1$$

- Positional filtering: Apply this length constrain to minimise the number of (one String of prefix, (rid1+record1,rid2+record2)) candidate pairs

$$|record2| >= |record1| * t$$

2. Code Steps :

2.1 Stage 1: Ordering Input

1. User tokens frequency to make input raw RDD ordered
2. When tokens have same frequency value, token will be ordered by each Int value from small to large.

2.2 Stage 2: Processing prefix

1. Find prefix value for each record following the formula : P = |r| - ceil(|r| * t) + 1
2. Yield (CommonString, rid1+record1) pairs for each prefix string
3. All these pairs are stored in prefixRecordMap RDD.

2.3 Stage 3: Processing prefix

1. Self join with rdds
2. Filter the all candidates to meet the id constrain whose rid1 should less than rid2. For one common prefix string, rid1 only join with rid2 who is larger. This step can also remove half of candidate pairs which are redundant. For example, for id pairs like (1,2) and (2,1). Only id pair (1,2) is keeped.
3. Filter with the length constraints of the record2 to meet the requirement that: |record2| >= |record1| * t
4. After remove impossible candidates pairs, Jaccard similarity computation start.
5. Filter out results which meet the threshold requirement.
6. Reduce the duplicated result to one.
7. Sort the result.

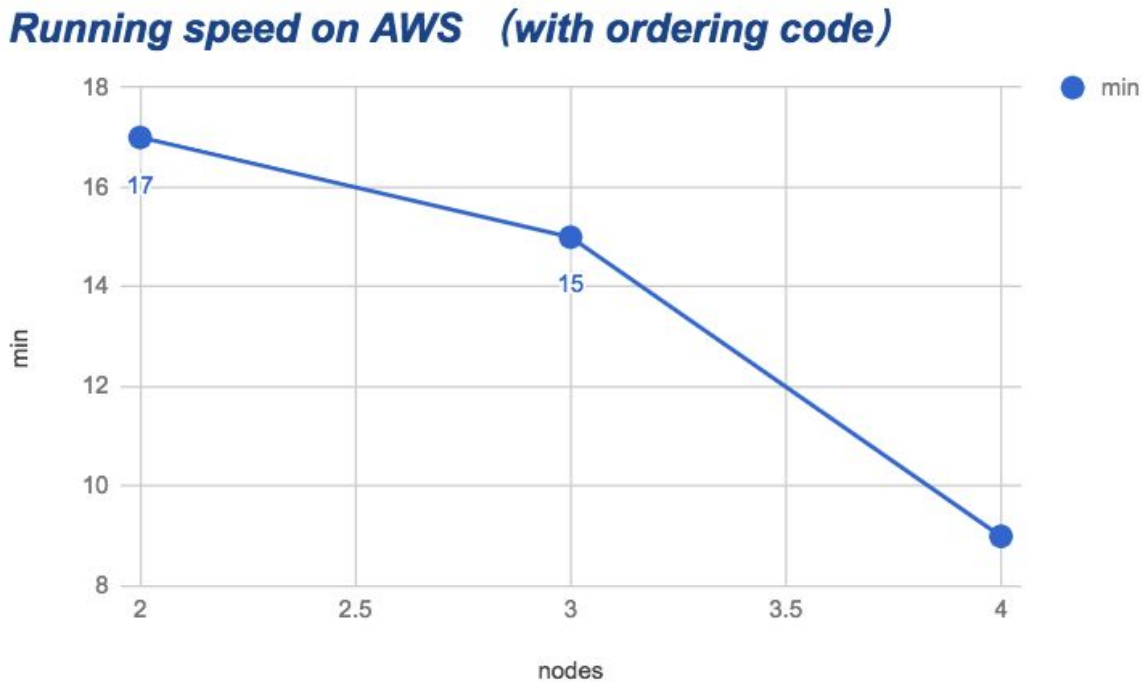
3. RDD operation selection :

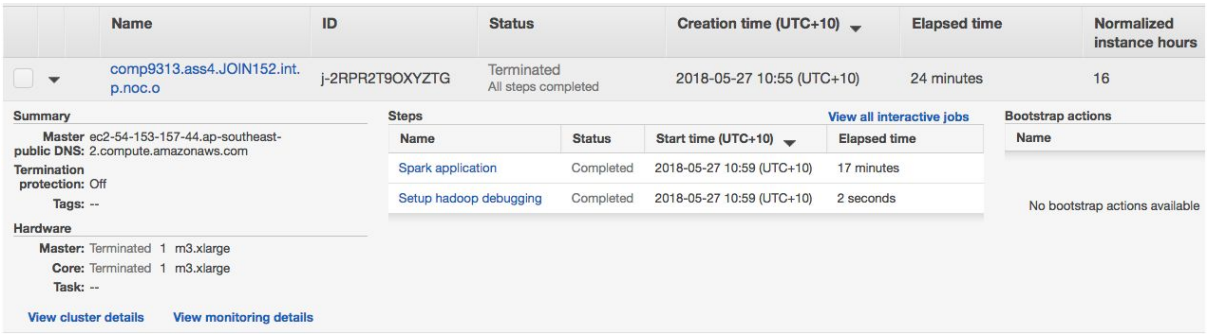
1. Join:Join all possible combinations of one prefix string as the structure below.

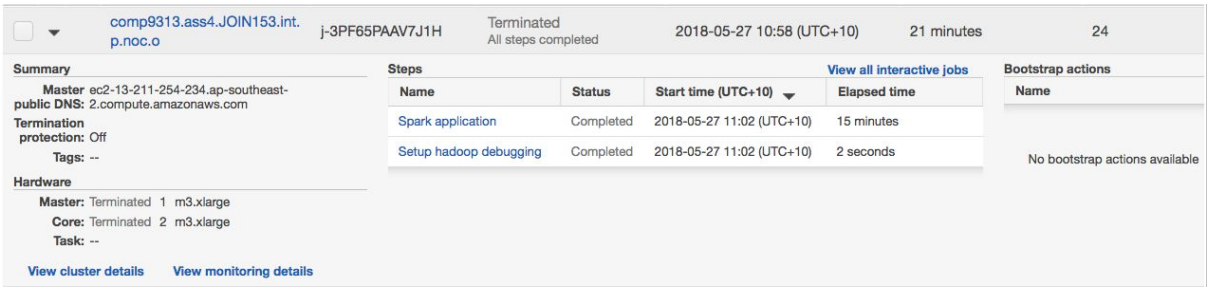
$$(One\ prefix\ string,\ \ (rid1 + record1, rid2 + record2))$$

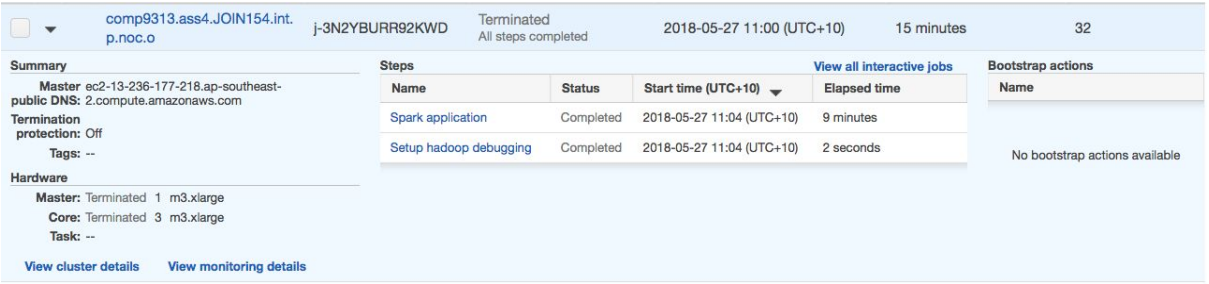
2. Filter : It is a map site operation. Filter is more efficiency than shuffle side operations like reduceByKey, groupByKey.
3. Reduce shuffle times in the middle of the RDD process. Just using reduceByKey once at the end to reduce duplicated results.
4. Only using “collect” for ordering once to keep away from “ out of memory” problem.
5. Use rdd.persist() to keep the original RDD in memory in order to process the data fast.

4. Outcome on AWS:









Cluster	Nodes	Running time on AWS (with ordering process)
Cluster1	2 nodes	17min
Cluster2	3 nodes	15min
Cluster3	4 nodes	9min