Project II - Database System

Sutandi, I Made Sanadhi

May 19, 2018

Notes:

- The measurements were conducted on Mac OSX (10.12.6), with 1.6 GHz Intel Core i5 CPU and 8GB memory.
- For task 1.1 & 1.2, the number of reducers is set to 4.
- For task 2.1, the number of reducers is set to 10.
- For task 2.2, the bucket size is set to 1000.

1 Task I - Implementation of Cube Operator

1.1 Variation of Input Size

Query:

```
SELECT lo_suppkey, lo_shipmode, lo_orderdate, SUM (lo_supplycost)
FROM LINEORDER_{SIZE}
CUBE BY lo_suppkey, lo_shipmode, lo_orderdate
```

Result

Execution time over different input size			
Input Size	Small	Medium	Big
Two-phase	2.06 s	17.75 s	202.18 s
Naive	1.99 s	22.09 s	150.20 s

Observation

The results indicate that the naive algorithm is in general better, especially for small and big datasets. In single-machine setup, the two-phase algorithm brings more load and tasks, thus outweighs the benefit of smaller data size in reducers. Since each dataset fits in memory and has less than 10M tuples, it is hard to observe the advantage of two-phase algorithm. In the medium dataset, it may have more data with the same grouping of cube attributes so that the first reduce phase can decrease the data size significantly and can increase the overall performance.

1.2 Variation of Number of Cube Attributes

Query:

```
SELECT {CUBE_ATTRIBUTES}, SUM (lo_supplycost)
FROM LINEORDER_Medium
CUBE BY {CUBE_ATTRIBUTES}
```

Result

Execution time over different number of cube attributes				
Number of Cube Attributes	3	4	5	6
Two-phase	19.14 s	31.62 s	100.44 s	262.42 s
Naive	21.25 s	26.40 s	83.53 s	248.82 s

Observation

The results, again, shows that naive approach is better as the number of cube attribute $|D|$ increase. As we add attributes into cube, we can observe that the data is dominated by distinct combinations of cube attributes (e.g. lo_suppkey + lo_shipmode + lo_orderdate + etc). Therefore, the first reduce will not decrease the data size significantly and it will just add another load to the algorithm. The naive approach has more benefit in having less work for string operation, while two-phase needs twice string operation in first phase and second phase respectively.

1.3 Variation of Number of Reducers

Query:

```
SELECT lo_suppkey, lo_shipmode, lo_orderdate, SUM (lo_supplycost)
FROM LINEORDER_Medium
CUBE BY lo_suppkey, lo_shipmode, lo_orderdate
```

Result

Execution time over different number of reducers					
Number of Reducers	4	8	12	16	20
Two-phase	17.74 s	17.91 s	21.08 s	23.74 s	25.56 s
Naive	21.57 s	20.80 s	22.47 s	24.24 s	26.48 s

Observation

In single-machine setup, increasing the number of reducers only affects the number of pipelined operations (instead of parallel operations) by splitting the data into smaller pieces as the number of reducers increase. Thus data splitting and deploying more reducers increase the overhead of operation and context switching. Consequently, both performance of two-phase and naive algorithm will experience degradation if the reducers keep increasing. We can see that the ideal number of reducers in this setup is 8.

2 Task II - Implementation of Theta Join Operator

Query:

```
SELECT COUNT(*)
FROM INPUT1_2K left, INPUT2_2K right
WHERE left.num >= right.num
```

2.1 Variation of Bucket Size

Result

Execution time over different size of bucket							
Bucket Size	100	300	500	1000	2000	5000	10000
Execution time	69.49 s	67.15 s	55.32 s	53.37 s	62.58 s	79.64 s	118.99 s

Observation

We can see that it this scenario, the optimal size of each bucket is 1000. The smaller the bucket size, the more we need to add reducers. In this single-machine setup, this leads to performance degradation caused by wait-operation of reducers and context switching (pipelined execution). On the other hand, if the bucket size is too big, then the overall performance suffers to reducers that process too many data (bottleneck) even though we have only a few reducers.

2.2 Variation of Number of Reducers

Result

Execution time over different number of reducers							
Number of Reducers	10	25	50	100	250	500	1000
Execution time	54.11 s	52.52 s	53.91 s	53.77 s	53.61 s	53.28 s	53.99

Observation

Following the given details on the project description and TAs explanation, the final number of reducers is independent of the number of reducers that initially given to the program. The final number of reducers is significantly influenced by the bucket size. The initial reducers number implies only the number of horizontal and vertical boundaries to consult the histogram, which follows the given formula. As the number rise, it will enhance the histogram dimension. We can observe that there is no obvious difference of the execution time as the initial number of reducers increases.

3 Conclusions

3.1 Two-Phase MapReduce vs Naive Algorithm for CUBE

It is not very deterministic to compare performance of Two-phase MapReduce and Naive algorithm in the single-machine setup because the level of parallelism is limited by single machine's processing power. Thus the difference in term of performance is not very clear. To gain more overall performance over the overhead of additional operations, two-phase MapReduce should be able to reduce data during the first phase significantly. Since the datasets, including the big table, all fit into main memory, naive algorithm outperforms two-phase in experiment 1.1 and 1.2. There is a high chance that naive algorithm will fail in the first phase (map) of the algorithm, if it is being run on real clusters environment in which every node has only small amount of memory. In this situation, two-phase mapreduce will benefit more with controlled or reduced intermediate data size.

3.2 Theta Join over Spark

The implementation follows not only M-Bucket-I but also M-Bucket-O algorithm. This approach leads to the huge amount of final reducers since each bucket must satisfy not only $|R|$ + $|S|$ condition but also $|R||S|$ condition. However, this method ensures the uniform distribution of input and output across all reducers and makes the overall join operation to be significantly faster since the workload is balanced for all reducers.