CS-422 - Project 2

Joachim Desroches (257178) Elias Manuel Poroma Wiri (294650)

2019-05-19

Task 1

We implemented two different algorithms, the first one is MRDataCube proposed by Suan Lee et al. The second one is a naive cube algorithm proposed by Nandi et al. Then we tested them on small, medium and big datasets. Another parameter that we modified is the number of reducers configured when running the spark-submit command, which is represented on each column of the tables below. The last parameter is the number of dimensions of the cube created, it is represented in the rows of the tables below and goes from 3 to 6.

The conclusion we came to is that the naive algorithm performs better when we have a small number of cube dimensions and reducers, whereas the MRDataCube algorithm has better performance almost on every other scenario.

Algorithm: **MRDataCube**
Number of tuples: **6K** (6008)

Time measured in [ms]		Number of Reducers		
		10	**20**	**30**
Number of Cube Attributes (Dimension D of the cube)	**3**	35.3023	65.0589	46.8330
	4	94.6313	65.4914	78.2043
	5	59.2602	126.2222	79.2283
	6	153.0986	35.8788	71.8639

Algorithm: **Naive Algorithm**
Number of tuples: **6K** (6008)

Time measured in [ms]		Number of Reducers		
		10	**20**	**30**
Number of Cube Attributes (Dimension D of the cube)	**3**	25.2163	126.9637	109.1131
	4	122.8320	90.7513	31.2124
	5	84.3463	173.4140	99.6532
	6	170.6035	70.6429	110.9013

Algorithm: **MRDataCube**
Number of tuples: **600K** (600572)

Time measured in [ms]		Number of Reducers		
		10	**20**	**30**
	3	34.8836	25.3094	37.7026
Number of Cube Attributes	**4**	110.9311	67.5825	84.4264
(Dimension D of the cube)	**5**	95.1040	63.6983	35.3521
	6	101.7303	73.5119	104.1054

Algorithm: **Naive Algorithm**
Number of tuples: **600K** (600572)

Time measured in [ms]		Number of Reducers		
		10	**20**	**30**
	3	26.1508	73.3294	89.7775
Number of Cube Attributes	**4**	99.6068	34.2414	126.6087
(Dimension D of the cube)	**5**	138.9725	89.4507	77.1543
	6	118.2258	85.4417	202.6642

Algorithm: **MRDataCube**
Number of tuples: **6M** (6001171)

Time measured in [ms]		Number of Reducers		
		10	**20**	**30**
	3	68.1261	47.2087	112.7991
Number of Cube Attributes	**4**	114.3350	182.5391	146.1181
(Dimension D of the cube)	**5**	70.5354	50.8846	78.0711
	6	121.1696	74.4586	53.2765

Algorithm: **Naive Algorithm**
Number of tuples: **6M** (6001171)

Time measured in [ms]		Number of Reducers		
		10	**20**	**30**
	3	77.6524	95.3614	65.0581
Number of Cube Attributes	**4**	135.8074	120.8034	79.2622
(Dimension D of the cube)	**5**	142.7686	147.9382	115.3302
	6	103.0501	149.1737	324.3084

Task 2

Varying the amount of anchors and the size of the dataset gave the obviously expected results: augmenting the number of anchor points up to the number of cores on the machine/cluster provided significant improve on speeds, whatever the dataset size; however adding more had a slightly negative impact on performance: indeed, it meant that more work had to be performed during assignation and shuffling, but added no additional parallelism to the comparisons done. Hence, unsurprisingly, best performance was achieved with the number of anchors being the number of available cores for the job to run on.

Task 3

The implementation of sampler uses stratified random sampling (Scalable Simple Random Sampling and Stratified Sampling by Xiangrui Meng) in order to get many representative samples, then we choose the ones that fit our queries the best, for this purpose first we choose the set of attributes, then we choose the best size of the sample according to these attributes and our expected accuracy.

This way we make sure that our sampling returns the correct answer to the queries with a very high probability.