Binary Theta-Joins using MapReduce: Efficiency Analysis and Improvements

Ioannis K. Koumarelas
Dept. of Informatics
Aristotle University
Thessaloniki, Greece
koumarel@csd.auth.gr

Athanasios Naskos
Dept. of Informatics
Aristotle University
Thessaloniki, Greece
anaskos@csd.auth.gr

Anastasios Gounaris
Dept. of Informatics
Aristotle University
Thessaloniki, Greece
gounaria@csd.auth.gr

ABSTRACT

We deal with binary theta-joins in a MapReduce environment, and we make two contributions. First, we show that the best known algorithm to date for this problem can reach the optimal trade-off between the size of the input a reducer can receive and the incurred communication cost when the join selectivity is high. Second, when the join selectivity is low, we present improvements upon the state-of-the-art with a view to decreasing the communication cost and the maximum load a reducer can receive, taking also into account the load imbalance across the reducers.

1. INTRODUCTION

Data analysis on voluminous data, such as clickstream data or data derived from scientific experiments and simulations, has given rise to the establishment of MapReduce as the most popular framework for large-scale processing. Analytical database queries remain a useful tool for big data analyses; however, such queries are being investigated in the MapReduce context rather than within a traditional DBMS environment. Analytical query processing in MapReduce has attracted a lot of interest, and the relevant work has investigated several issues, including indexing, data placement and layouts, optimizations, iterative processing, fair load allocation and interactive processing to name some of them [5]. In this work, we focus on improving the efficiency of join queries executed in MapReduce, for which several proposals already exist [7, 2, 9]. More specifically, we target binary theta-joins, where the join condition between two datasets is arbitrarily complex rather than a simple equation.

Nevertheless, most of the proposals to date tend to be developed on a best-effort basis, without systematically analyzing the inherent trade-offs. Two recent remedies to that have been proposed in [1, 8]. [8] introduces the notion of *minimal* MapReduce algorithms, which are algorithms accompanied by guarantees (up to a small constant) regarding several aspects, such as memory consumption and communication cost. The MapReduce rounds may be bounded but they can be more than one. The work in [1] is complementary and presents a way to compute the lower bounds on communication cost as a function of the maximum input a reducer is allowed to receive for specific problems. This allows to define the trade-off between the load on the reducer side and the *replication rate*. The replication rate is defined as the average ratio of output to input key-value pairs on the map side, and is used as a metric of the communication cost. Further, the work in [1] examines whether known algorithms for those problems can match the lower bounds, provided that they consist of a single MapReduce round.

The algorithms 1-Bucket-Theta and M-Bucket in [7] form the basis of our work. Our first contribution is that we analyze the lower bounds for the binary theta-join problem and we show that the worst-case behaviour of 1-Bucket-Theta matches those bounds. However, such behaviour is expected only when the join selectivity is high. For low selectivities, and with the help of histograms, the more efficient M-Bucket-I and M-Bucket-O algorithms are presented in [7], which aim at minimizing the maximum reducer input and output, respectively. Our second contribution is that we enhance those algorithms through the clustering of histogram buckets. In that way, we can achieve more efficient partitioning of histogram buckets to reducers. The efficiency is measured in terms of the replication rate, the maximum reducer input, and the imbalance across reducers. We show that we can improve the replication rate (i.e., reduce the communication cost) and the maximum reducer input (i.e., reduce the longest running time and the space requirements of reducers) with insignificant impact on load imbalance.

The remainder of this extended abstract is structured as follows. In Sec. 2 we briefly present the 1-Bucket-Theta and M-Bucket algorithms, which we analyze in Sec. 3 and enhance in Sec. 4, respectively. In Sec. 5, we conclude and describe next steps.

2. BACKGROUND

In [7] the problem of performing binary theta joins $S \bowtie_\theta T$ on MapReduce is studied. The core of the approach lies in how the workload is partitioned across reducers. To represent the workload, a *join matrix (JM)* is used. In JMs, each cell corresponds to a pair of tuples, one from each input dataset, to be processed. The JM is split into several regions, where each region is mapped to a reducer. For each region, we can compute the amount of tuples that belong to it, which is the *input cost* of that region and is directly related to the computation and memory load of the associated reducer. For perfect load balancing, we want these regions to

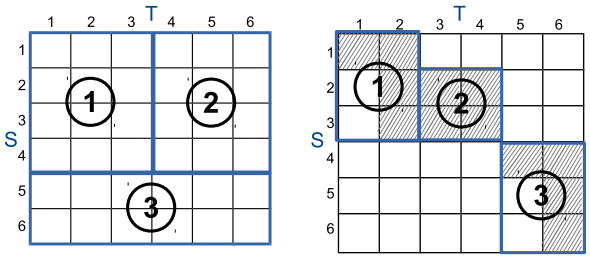

Figure 1: Partitioning the JM in 1-Bucket-Theta (left) and M-Bucket (right).

have equal input cost. In order to accomplish the latter objective, two main algorithms are presented: 1-Bucket-Theta and M-Bucket-I (and its variation M-Bucket-O).

2.1 1-Bucket-Theta

1-Bucket-Theta is the most generic algorithm, since it examines all tuple pairs (as in the Cartesian product), and requires minimal statistical information, namely just the cardinalities of the input. The strong point of the algorithm is the principled way that it partitions the JM, in a way that all JM cells are covered and, at the same time, the maximum reducer input is minimized. The algorithm is shown to be more suitable for high join selectivities (e.g., above 50%). Fig. 1(left) shows an example partitioning across 3 reducers, where there are 6 tuples from S and T, and the input cost of each reducer is 7 (4 tuples from S and 3 from T), 7 (4 from S and 3 from T) and 8 (2 from S and 6 from T), respectively.

2.2 M-Bucket-I

In cases where there are histograms, so that we can safely reason as to whether a specific combination of tuples can satisfy the join condition, and the join selectivity is small, M-Bucket-I outperforms 1-Bucket-Theta. The histograms are equi-depth ones and are produced in a separate MapReduce phase, as explained in [7]. Then, the JM is constructed, where each cell corresponds to a pair of histogram buckets rather than a pair of tuples. As such, the size of a JM need not grow as the size of the input data increases at the expense of histogram buckets of higher depth. From the JM and the join condition, it is straightforward to identify pairs that do not contribute to the result (depicted as white cells in Fig. 1(right). During the partitioning step, a heuristic method is followed, which is not accompanied by guarantees as in 1-Bucket-Theta but yields better results, since it benefits from the fact that most of the JM cells are not valid candidate pairs.

The difference between M-Bucket-I and M-Bucket-O is that the former targets the minimization of the maximum reducer input, whereas the latter targets the minimization of the maximum reducer output. Note that estimating the reducer output based on histograms is prone to significant errors, even when the histograms are accurate.

3. ON THE OPTIMALITY OF 1-BUCKET-THETA

First we define the lower bound on the communication cost of any 1-round MapReduce algorithm for binary theta-joins. As already mentioned, the communication cost is measured using the replication rate metric. Let us examine the steps of the short version of the generic recipe for deriving such bounds from [1]. Given two relations S and T, with sizes $|S|$ and $|T|$, respectively, we have:

- **Size (Number)** of Inputs and Outputs:
  - Inputs: |S|+|T|
  - Outputs: |S||T| (accounting for the worst case, which is the cartesian product)
- **Deriving** $g(q)$: The upper bound of outputs a reducer can produce given q inputs, denoted as $g(q)$, occurs when q is equally divided into input from S and T, i.e., $\frac{q}{2}$ tuples from S and $\frac{q}{2}$ tuples from T. The maximum result of applying the theta join on these two quantities is when we have a cartesian product, thus $g(q) = \frac{q^2}{4}$.
- **Replication Rate** $r(q)$: The quantity $\frac{g(q)}{q}$ equals $\frac{\frac{q^2}{4}}{q} = \frac{q}{4}$, which is monotically increasing in q. Therefore, the replication rate can be computed using the formula: $r(q) \geq \frac{q|O|}{g(q)I} = \frac{4(|S||T|)}{q(|S|+|T|)}$. So, the lower bound on r, r_{lb}, is $\frac{4(|S||T|)}{q(|S|+|T|)}$.

The above formula illustrates the exact trade-off between parallelism and communication cost in binary theta-joins. By increasing the degree of parallelism in order to decrease the input q each reducer receives, the communication cost increases, since, for the lower bound, q and r are inversely proportional to each other.

The next step is to find the upper bound on replication rate of 1-Bucket-Theta. In [7], three partitioning cases are presented, based on the sizes $|S|$ and $|T|$ and the number of available reducer processors p. Due to the limited space, we will examine only the first case in detail.

The first case corresponds to the scenario, where the JM can be exactly covered by $c_S \times c_T$ squares of side-length $\sqrt{|S||T|/p}$. This means that the following conditions hold: $|S| = c_S\sqrt{|S||T|/p}$ and $|T| = c_T\sqrt{|S||T|/p}$, where c_S, c_T are positive integers. For example, if $p = 4$, then the JM in Fig. 1(left) can be exactly covered by 4 squares of side-length 3. Then we have:

- Replication rate of 1-Bucket-Theta (r_{1BT}):
  $r_{1BT} \leq \frac{|S|c_T+|T|c_S}{|S|+|T|} = \frac{\frac{|S||T|}{\sqrt{\frac{|S||T|}{p}}}+\frac{|T||S|}{\sqrt{\frac{|S||T|}{p}}}}{|S|+|T|}$
  $= \frac{2|S||T|}{(\sqrt{\frac{|S||T|}{p}})(|S|+|T|)}$
- Reducer input: $q_{1BT} = 2\sqrt{\frac{|S||T|}{p}}$
- Combining r_{1BT} and q_{1BT}:
  $r_{1BT}q_{1BT} \leq (\frac{2|S||T|}{(\sqrt{\frac{|S||T|}{p}})(|S|+|T|)})(2\sqrt{\frac{|S||T|}{p}}) = 4\frac{|S||T|}{|S|+|T|}$
  which implies that $r_{1BT}(q_{1BT}) \leq r_{lb}$

So, the upper bound of the first case of 1-Bucket-Theta is at most as high as the lower bound of the problem, which means that, for that case, the algorithm is optimal.

Following the same reasoning, the other two cases (Theorems 2 and 3 in [7], respectively), which correspond to different formulas for c_S and c_T, can be examined, for which we have:

- Case 2: $r_{1BT} \leq \frac{4}{q}\frac{|T||S|}{|S|+|T|} = r_{lb}$
- Case 3: $r_{1BT} \leq \frac{8}{q}\frac{|S||T|}{|S|+|T|} = 2r_{lb}$

Overall, the upper bound of the replication rate is at most two times the lower bound, and as such is optimal up to a constant factor. In [3], it is shown that the lower bound can be met for self-joins, which is special case of binary joins.

4. REDUCING THE REPLICATION RATE IN M-BUCKET

The partitioner of M-Bucket-I algorithm operates on a join matrix (JM), where each cell corresponds to a pair of histogram buckets. It tries to fit the cells in rectangular regions; each region is associated with a single reducer. The rationale of our approach is to permute JM's rows and columns, in order to improve the quality of the partitioning phase.

The problem of cell rearrangement can be addressed with several algorithm families, such as *clustering* (e.g., hierarchical, array-based, and so on), *combinatorial optimization* (e.g., bin packing, knapsack) and *bandwidth reduction*. Here, we examine the impact of array-based clustering algorithms and more specifically, we employ the Bond Energy clustering algorithm (BEA) [6], due to its efficiency [4]. The purpose of BEA is to identify natural clusters that occur in complex data arrays, such as JMs. This task is accomplished by permuting the rows and columns of the JM in a way that the numerically larger array elements are clustered together. As the JM of our interest comprises a two-dimensional bitmap array, i.e. the cell values are either 0 or 1 to indicate whether the processing of the corresponding pairs is meaningful or not, we expect all the non-zero values to be grouped as close as possible. The intuition is that, if the JM contains more empty sub-matrices, the mapping of the remainder sub-matrices to reducers will improve.

Our work adds a step of beforehand analysis to the M-Bucket-I/O algorithm, just after the histograms are built and the initial JM is produced. It thus takes place before the actual execution on a MapReduce platform. The quality of a JM is assessed with the help of the following three metrics:

1. *replication rate (rep)*, defined as in the Introduction and [1];
2. *maximum reducer input (mri)*; and
3. *input imbalance (imb)*, defined as the ratio of *mri* to the average reducer input, considering only the non-idle reducers.

Note that the metrics above can be accurately computed from the JM, without requiring the real execution to be completed. Thus, if the JM rearrangement is considered as not beneficial, the execution can switch back to the original JM. That is, it is straightforward to add a post-processing phase, in order to guarantee that we choose the best partitioning between the one based on the original and the one based on the re-arranged JM. Consequently, our proposal does never lead to performance degradation; actually it can lead to significant improvements according to our experiments.

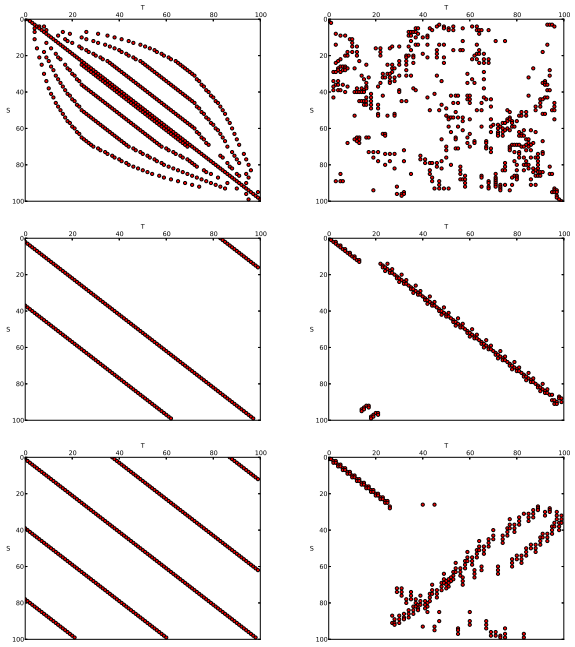

Figure 2: Example JMs before (left) and after (right) applying BEA.

As an example, we extracted a sample of 64M tuples from the Cloud dataset in `http://cdiac.ornl.gov/ftp/ndp026c/ndp026c.pdf`. Fig. 2(top) shows the initial and rearranged JM for a self-join query that retrieves record pairs, for which the absolute difference of the sea level is between 0 and 2, or between 22 and 24, or between 50 and 52, or between 80 and 82 to give an example of a complex range query. The rearranged JM yields 21% lower *rep* and 19% lower *mbi* at the expense of 4% higher *imb*. Next, we proceed to more systematic experiments on synthetic data.

4.1 Experimental Evaluation

We focus on band joins, which is a type of theta-joins that can significantly benefit from M-Bucket. In band joins, the condition is in the form of $R.A - \varepsilon \leq S.A \leq R.A + \varepsilon$. The experimental setup is as follows. We randomly generate synthetic JMs so that the produced JMs vary in the following aspects: join selectivity, number of band conditions, and size of JMs. Then, we compute the statistics of the resulting partitioning to reducers both when we cluster the JM and when we do not. In the first experiment, we assume that the dimensions of the JM are 100×100. We vary the number of available reducers from 10 to 40. Also, the numbers of band conditions examined are 1, 3 and 5. For each band condition, we examined selectivity values of 1%, 5% and 10%.

Fig. 2 shows two more examples of JM rearrangement. From the left column of the middle and bottom row, we can see the typical form of the original synthetic JMs. For each band condition, there is a diagonal stripe of cells, for which the join condition holds. The gaps between such stripes are randomly shifted, so that the JMs are not symmetric; for each condition the selectivity is set to 1%. As we can observe, the effect of the BEA algorithm is optically widely different, but in both cases, there were significant improvements, which we discuss below.

The average impact of BEA on the metrics examined are

	rep	*mri*	*imb*	coverage
Overall	0.846	0.880	1.029	59.26%
Band Selectivity				
1%	0.717	0.735	1.028	66.67%
5%	0.920	0.949	1.014	66.67%
10%	0.928	0.996	1.056	44.45%
Number of Band Conditions				
1	0.987	0.967	0.964	33.34%
3	0.821	0.835	1.010	44.45%
5	0.810	0.873	1.058	100%

Table 1: Average ratio of the BEA-produced JM metrics to the original JM metrics.

	rep	*mri*	*imb*
Overall	0.634	0.649	1.023
Band Selectivity			
1%	0.634	0.649	1.023
5%	0.833	0.875	1.050
10%	0.848	0.900	1.050
Number of Band Conditions			
1	0.979	1	0.988
3	0.737	0.733	0.995
5	0.634	0.649	1.023

Table 2: Ratio of the BEA-produced JM metrics to the original JM metrics for the maximum *rep* drop observed.

summarized in Table 1. The rightmost column of the table shows the percentage of the times that the rearranged JM has led to improvements in the replication rate. Table 2 refers to the maximum improvements regarding replication observed. From these two tables, we can draw the following conclusions. On average, our proposal improves the partitioning in approximately 59% of the times. In those cases, the average decrease in the replication rate is 15%, but it can reach 37%. The improvements become more significant as the number of the band conditions increase and the selectivity becomes lower. On average, when the band selectivity is 1%, the replication rate drops by 28%, while the maximum reducer input decreases by 26%. There is a slight increase in the relative imbalance though. Similarly, we can observe, that, when the number of band conditions is 5, there are improvements in all the cases examined.

We also investigated the impact of the number of reducers, but this was not found to be significant. Finally, note that we considered only the cases where the replication rate is strictly less than that with the original JMs in order to compute *mri* and *imb*. The average values of these two metrics are slightly different if all the measurements are considered.

We conducted an additional experiment, where we increased the dimensions of the JM to 1000×1000 and we further decreased the minimum selectivity of each band condition to 0.1%. The main purpose was to verify our hypothesis that our proposal is more suitable for band joins with multiple conditions, each having a low selectivity. Indeed, in 100% of the cases examined when the selectivity was 0.1% and the number of band conditions was 3 and 5, there was a significant decrease in the replication rate (28.1% on average). The maximum reducer input was also decreased by the same amount, whereas the imbalance remained similar. Overall, when the selectivity is low, there is more space for BEA to yield empty sub-matrices; whereas, when there are fewer band conditions, the differences from the original JMs are less significant.

5. CONCLUSIONS AND FURTHER WORK

We investigate the execution of binary theta-joins using MapReduce. First we analyze the efficiency of the state-of-the-art and second, we propose the usage of a pre-processing clustering algorithm in order to help the partitioning of the map output to reducers. Our proposal was shown to incur significant reductions in the communication cost and the maximum input received by each reducer when the theta clause comprises several conditions, each of low selectivity. A strong point of our approach is that it is not intrusive, in the sense that it can be easily incorporated into the current state-of-the-art proposal in [7], as a pre-processing phase before the actual execution on a MapReduce platform begins. In addition, it is straightforward to assess whether our approach is beneficial for a specific setting, and thus our proposal does not lead to overall performance degradation.

In the future, we plan to focus on more elaborate types of array rearrangement algorithms. Scalability is also an issue, since algorithms such as BEA do not scale to matrices with very large dimensions. Another avenue for further work is to investigate more sophisticated partitioning algorithms to be coupled with JM rearrangement. Harder problems include the investigation of provably optimal techniques for multi-way theta-joins and efficient histogram construction when there are multiple attributes participating in the theta-join condition.

Acknowledgments This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

6. REFERENCES

[1] F. N. Afrati, A. D. Sarma, S. Salihoglu, and J. D. Ullman. Upper and lower bounds on the cost of a map-reduce computation. *PVLDB*, 6(4):277–288, 2013.

[2] F. N. Afrati and J. D. Ullman. Optimizing multiway joins in a map-reduce environment. *IEEE Trans. Knowl. Data Eng.*, 23(9):1282–1298, 2011.

[3] F. N. Afrati and J. D. Ullman. Matching bounds for the all-pairs mapreduce problem. In *IDEAS*, pages 3–4, 2013.

[4] S. Climer and W. Zhang. Rearrangement clustering: Pitfalls, remedies, and applications. *Journal of Machine Learning Research*, 7:919–943, 2006.

[5] C. Doulkeridis and K. Nørvåg. A survey of large-scale analytical query processing in mapreduce. *The VLDB Journal*, pages 1–26, 2013.

[6] W. T. McCormick, P. J. Schweitzer, and T. W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972.

[7] A. Okcan and M. Riedewald. Processing theta-joins using mapreduce. In *SIGMOD*, pages 949–960, 2011.

[8] Y. Tao, W. Lin, and X. Xiao. Minimal mapreduce algorithms. In *SIGMOD*, pages 529–540, 2013.

[9] X. Zhang, L. Chen, and M. Wang. Efficient multi-way theta-join processing using mapreduce. *PVLDB*, 5(11):1184–1195, 2012.