Brief Announcement: Meta-MapReduce A Technique for Reducing Communication in MapReduce Computations*

Foto Afrati[1], Shlomi Dolev[2], Shantanu Sharma[2], and Jeffrey D. Ullman[3]

[1] National Technical University of Athens, Greece.
[2] Ben-Gurion University of the Negev, Beer-Sheva, Israel.
[3] Stanford University, USA.

The federation of cloud and big data activities is the next challenge where MapReduce should be modified to avoid (big) data migration across remote (cloud) sites. This is exactly our scope of research, where only the very essential data for obtaining the result is transmitted, reducing communication, processing and preserving data privacy as much as possible. We propose *an algorithmic technique for MapReduce algorithms*, called *Meta-MapReduce*, that *decreases the communication cost by allowing us to process and move metadata to clouds and from the map to reduce phases*. Details are given below:

Locality of data. Input data to a MapReduce job, on one hand, may exist at the same site where mappers and reducers reside. However, ensuring an identical location of data and mappers-reducers cannot always be guaranteed. On the other hand, it may be possible that a user has a single local machine and wants to enlist a public cloud to help data processing. Consequently, in both the cases, it is required to move data to the location of mappers-reducers. In order to motivate and demonstrate the impact of different locations of data and mappers-reducers, we consider a real example, as: *Amazon Elastic MapReduce*. Amazon Elastic MapReduce (EMR) processes data that is stored in Amazon Simple Storage Service (S3), where the locations of EMR and S3 are not identical. Hence, it is required to move data from S3 to the location of EMR. However, moving all data from S3 to EMR is not efficient if only small specific part of it is needed for the final output.

Communication cost. The *communication cost* dominates the performance of a MapReduce algorithm and is the sum of the total amount of data that is required to move from the location of users or data (*e.g.*, S3) to the location of mappers (*e.g.*, EMR) and from the map phase to the reduce phase in each round of a MapReduce job. In this paper, we are interested in *minimizing the data transferred* in order to avoid communication

* **Accepted in 17th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS, 2015).** More details appear in the technical report 15-04, Department of Computer Science, Ben-Gurion University of the Negev, Israel, 2015. Supported by the project Handling Uncertainty in Data Intensive Applications, co-financed by the European Union (European Social Fund) and Greek national funds, through the Operational Program "Education and Lifelong Learning," under the program THALES, Orange Labs, Rita Altura trust chair in computer science, the Lynne and William Frankel Center for Computer Science, and the Israeli Science Foundation (grant number 428/11).

and memory overhead, as well as to protect data privacy as much as possible. If *few* inputs are required to compute the final output, then it is not communication efficient to move all the inputs to the site of mappers-reducers, and then, the copies of same inputs to the reduce phase.

Meta-MapReduce. We provide a new *algorithmic* approach for MapReduce algorithms, Meta-MapReduce, that decreases the communication cost significantly. Meta-MapReduce (M-MR) regards the locality of data and mappers-reducers and avoids the movement of data that does not participate in the final output. Particularly, *M-MR provides a way to compute the desired output using metadata*[4] (which is much smaller than the original input data) and avoids to upload all data (either because it takes too long or for privacy reasons). It should be noted that we are enhancing MapReduce and not creating entirely a new framework for large-scale data processing; thus, M-MR is *implementable* in state-of-the art MapReduce systems such as Spark or modern Hadoop. In addition, M-MR also allows us to protect data privacy as much as possible in the case of an *honest-but-curious* adversary by not sending all the inputs. Nevertheless, by the auditing process, a *malicious* adversary can be detected. Moreover, in some settings auditing enforces participants to be honest-but-curious rather than malicious, as malicious actions can be discovered and imply punishing actions.

Having the same scenario of locality of input data, in the standard MapReduce, users send their data to the site of mappers before the computation begins. However, in M-MR, users send metadata to the site of mappers, instead of original data.

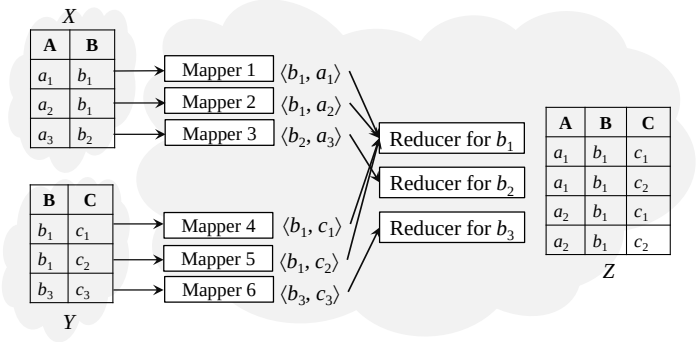

Fig. 1. Equijoin of two relations.

An example of equijoin of two relations $X(A, B)$ and $Y(B, C)$. We present an example to show the impact of different locations of data and mappers-reducers on the communication cost involved in a MapReduce job. *Problem statement*: The join of relations $X(A, B)$ and $Y(B, C)$, where the joining attribute is B, provides output tuples $\langle a, b, c \rangle$, where (a, b) is in A and (b, c) is in C. In the equijoin of $X(A, B)$ and $Y(B, C)$, all tuples of both the relations with an identical value of the attribute B should

[4] The term metadata is used in a different manner, and it represents a small subset, which varies according to tasks, of the dataset.

appear together at the same reducer for providing the final output tuples. In Figure 1, two relations $X(A, B)$ and $Y(B, C)$ are shown, and *we consider that the size of all the B values is very small as compared to the size of values of the attributes A and C.*

Communication cost analysis: In Figure 1, the communication cost for joining of the relations X and Y is the sum of the sizes of all three tuples of each relation that are required to move from the location of the user to the location of mappers, and then, from the map phase to the reduce phase. Consider that each tuple is of unit size, and hence, the total communication cost is 12 for obtaining the final output. Using M-MR, where values of the attribute B work as metadata, there is no need to send tuples having values b_2 and b_3 to the location of computation. Thus, a solution to the problem of equijoin has only 4 units cost plus a constant cost for moving metadata.

Table 1. The communication cost for joining of relations using Meta-MapReduce.

Problems	Communication cost	
	using Meta-MapReduce	using MapReduce
Join of two relations	$2nc + h(c + w)$	$4nw$
Skewed Values of the Joining Attribute	$2nc + rh(c + w)$	$2nw(1 + r)$
Join of two relations by hashing the joining attribute	$6n \cdot log\ m + h(c + w)$	$4nw$
Join of k relations by hashing the joining attributes	$3knp \cdot log\ m + h(c + w)$	$2knw$
n: # tuples in each relations, c: the maximum size of a value of the joining attribute, r: the replication rate, h: # tuples that actually join, w is the maximum required memory for a tuple, p: the maximum number of dominating attributes in a relation, and m: the maximal number of tuples in all given relations.		