Reducer Capacity and Communication Cost in MapReduce Algorithms Design *

Foto Afrati
National Technical University of Athens, Greece

Shlomi Dolev, Ephraim Korach, Shantanu Sharma
Ben-Gurion University, Israel

Jeffrey D. Ullman
Stanford University, USA

Abstract. An important parameter to be considered in MapReduce algorithms is the "reducer capacity," is introduced here for the first time. The reducer capacity is an upper bound on the sum of the sizes of the inputs that are assigned to the reducer. We consider, for the first time, the different sizes of the inputs, which are sent to the reducers. Another significant parameter in a MapReduce job is "communication cost" – the total amount of data transferred – between the map and reduce phases. The communication cost can be minimized by minimizing the number of copies of inputs sent to the reducers.

Problem Statement. We consider two problems, as: (*i*) a problem in which each input should appear with each other input in at least one common reducer, and (*ii*) a problem in which, each input of a set, X, should appear with each input of another set, Y, in at least one common reducer. Our objective is to **find a solution to** both **the problems using a minimum number of reducers so that we have the optimum communication cost** (the total amount of data transferred between the map and reduce phases). However, reducers do not have an *unbounded* memory, and hence, the total size of inputs sent to a reducer is always bounded. We call the *reducer capacity* that restricts a reducer to hold more inputs, and the reducer capacity is an upper bound on the sum of the sizes of the inputs that are assigned to the reducer. *Example*: the main memory of a processor where a reducer executes can work as the reducer capacity. We can achieve the minimum communication cost by using only a single reducer of enough capacity such that all the inputs are assigned to the reducer; however, we do not have parallelism at the reduce phase. Hence, we may need to increase the communication cost by sending inputs to more reducers of a fixed capacity.

Motivation. We demonstrate the new aspect of reducer capacity in the scope a useful special case where an output depends on exactly two inputs. We present two problems where each output depends on exactly two inputs, as follows:

All-to-All problem. In the *all-to-all* (*A2A*) problem, a set of inputs is given, and each pair of inputs corresponds to one output. *Example 1*: Similarity-join. *Example 2*: We are given a set of seven inputs $I = \{i_1, i_2, \ldots, i_7\}$ whose size set is $W = \{0.20q, 0.20q, 0.20q, 0.19q, 0.19q, 0.18q, 0.18q\}$ and reducers of capacity q. In Figure 1, we show two different ways that we can assign the inputs to reducers. The best way to minimize the communication cost is to use three reducers. However, there is less parallelism at the reduce phase as compared to when we use six reducers. Observe that when we use six reducers, all reducers have a lighter load.

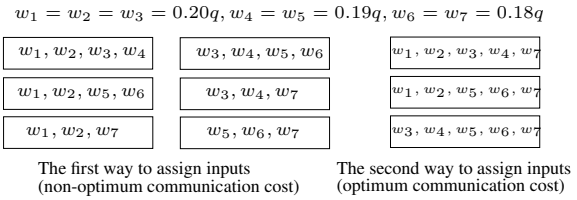

Figure 1. An example to the *A2A problem*.

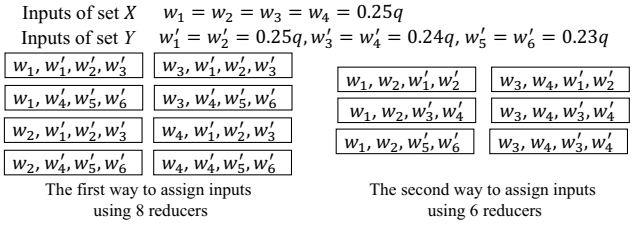

Figure 2. An example to the *X2Y problem*.

X-to-Y problem. In the *X-to-Y* (*X2Y*) problem, two disjoint sets X and Y are given, and each pair of elements $\langle x_i, y_j \rangle$, where $x_i \in X, y_j \in Y, \forall i, j$, of the sets X and Y corresponds to one output. *Example 1*: Skew join. *Example 2*: We are given two sets X of 4 inputs and Y of 6 inputs, see Figure 2, and reducers of capacity q. The best way is to use 6 reducers for minimum communication cost. We cannot obtain a solution for the given inputs using less than 6 reducers. However, the use of 6 reducers results in less parallelism at the reduce phase as compared to when we use 8 reducers.

Solutions. We prove that the optimization problem of using the minimum number of reducers for the *A2A* and the *X2Y* (assignment) problems is NP-hard and present a bin-packing-based approximation algorithm (BPAA), which results in near optimal communication cost, for these problems. We also provide a solution for a large number of inputs of small sizes using an extension of the BPAA.

Tradeoffs. We investigated the following three tradeoffs: (*i*) a tradeoff between the reducer capacity and the total number of reducers, (*ii*) a tradeoff between the reducer capacity and the parallelism, and (*iii*) a tradeoff between the reducer capacity and the communication cost.

* Preliminary versions were presented in DISC 2014, and the 2nd Workshop on Algorithms and Systems for MapReduce and Beyond (BeyondMR) 2015; see also, technical report 14-05 at Department of Computer Science, Ben-Gurion University of the Negev, Israel.