In this section 2.5 , we would introduce a model called Communication-Cost Model. The purpose of the model is to measure the quality of algorithms implemented on a computing cluster. For many application, the bottleneck is the communication between tasks such as moving data, reading input and delivering output. The communication cost is a way to measure the efficiency of an algorithm because of the bottleneck issue. The communication cost of a task is the size of the input to the task and the communication cost of an algorithm is the sum of the communication of all the tasks implementing that algorithm.

Here, we assume that an algorithm is implemented by an acyclic network of tasks and we will use a example to illustrate the computation cost of multiway joins as single MapReduce jobs is more efficient than a 2-way joins. We would use the number of tuples as a measure of size as we are using relational database operations. And we would only consider the communication cost as the dominant cost and count only the input size.. The reasons to exclude output size as cost is because of two reasons, the first one is that if the output of task a is the input of task b, it will double count the cost for both task and the second one is that the output size of the algorithm is usually incomparable to the input size of the algorithm.

In a cluster-computing system, communication costs also influence algorithm selection.

We must also consider wall-clock time, or the time it takes a parallel algorithm to complete.

Below we will use a multiway join as an example to analyze the communication cost of an algorithm. The general theory is outlined as:

(1) Hash function is used to have the relations involved in the natural join of three or more relations to have their values hashed.

(2) The product of the numbers of buckets for each attribute is k, the number of reducers that will be used.

(3) Each of the k reducers has one component for each of the attributes selected at step (1).

(4) The hash function(s) are used to determine certain components of the vector that identify the reducers.

We pick one join as example to illustrate the idea, the join R(A, B)⊳⊲S(B, C)⊳⊲ T(C, D) . Suppose the relations R, S and T have size r, s and t respectively and p is the probability that

1. An R-tuple and a S-tuple agree on B and

2. An S-tuple and a T-tuple agree on C.

Then the communication cost of a MapReduce algorithm joining R and S is O(r+s) and the size of the intermediate join R⊳⊲S is prs. When it joins with T, the communication cost of a second a MapReduce task is O(t+prs). Thus, the total cost of two 2-way joins is O(r+s+t+prs). On the other hand, joining S and T first and then R, the total communication cost is O(r+s+t+pst).

Instead of using two 2 way joins, we can join the three relations at once by a single MapReduce job. We can hash B values into number of b buckets and C values into number of c buckets such that bc = k, which k is the number of reducer and each reducer corresponds to a pair of buckets, B and C respectively. So the reducer is to join the tuples R(u,v), S(v,w) and T(w,x) where h(v) = i and g(w) = j, i and j belong to the b and c respectively. In this way , we will have a total communication cost O(r+2s+t+2√krt) and usually reduce to O(√k).