The communication-cost model is the model for measuring the quality of algorithms. In some situation this model has more efficient than the straightforward cascade of 2-way joins. In my understanding, usually the communication-cost of a task is the size of the input to the task. And we use bytes to calculate the communication-cost size. However, if we use communication-cost model in a special situation such as use this model in relational database operations, we can use the number of tuples as measure of size to instead of bytes. The communication cost of an algorithm is the total of the communication cost of all the tasks implementing that algorithm. When we utilize the communication cost of an algorithm, we will not consider the process time for all the tasks, while in a special case, the execution time of tasks much longer than other algorithm time. And there are a couple reasons why we will focus on the communication-cost. First, the size of the algorithm execution time always is dependent on the input. Second, processor executes instructions way faster than the typical interconnect speed for a computing cluster even the typical interconnect speed seems very fast. Moreover, when we interconnect and execute multiple compute nodes, the speed of executes time will be slower because the compute nodes will influence each other. The last reason is when the task executes on a compute node and there has been a situation which is the transfer chunk to main memory will spend more time than we execute chunk time. So why we only count the input size, and will not count the output size? Because when a task's output is another task's input, the first task's output already calculated, so we don't need to calculate this task output size again. And there has been another reason, the algorithm output is larger than then algorithm input, the reason is the if we don't summarize or aggregate the outputs, the outputs will not be useful until we handle them.

Wall-clock time is a very important, when we use wall-clock time we need a parallel algorithm to process, after executing the wall-clock time, we can put all the work into one task to reduce the total communication. But this way is very expensive. We usually will separate all the work into different tasks evenly, so when the compute nodes are given, the wall-clock time will be as small as possible.

Multiple joins is a good way to help people understand how the communication cost chooses an algorithm. We assume R(A,B) connect with S(B,C) connect with T(C,D), and the r,s,t is the size of R,S,T. If we join R,S first, using the MapReduce algorithm, the communication-cost is O(r+s), and we connect T, the MapReduce result communication-cost will change to O(t+prs). (prs is the size between the R and S). So, the total communication-cost in this case is O(r+s) + O(t+prs) = O(r+s+t+prs). In above example, there have three relations, so we call case Computation cost of the three-way joins. In three-way join, every reduce must be compute the join part of the three relations, such as in the above example, we compute R with S and output with T to get the commination costs O(r+s) and O(t+prs). And we need to make sure every step of joining time is the smallest, and the execution time is linear with the sum of input size. And there has another way called Star join. Star join is a very common structure in the mining data. The star join usually has multiple tuples such as A(a1,a2,…..) we call A fact table and every ai means the main-key. And for every key there also has a dimension table to give relative information such as D(a1,b11,b12…). For example, if we assume a1 is name, the b11 and b12 could be the address and phone number. And the fact table will be larger than the dimension table. And we can connect the fact table and multiple tuples together and use analytic query to mining the data.