Map Reduce Theory

Aaron Palumbo

Friday, September 11, 2015

Contents

Exercises: 2.5.1, 2.6.1

Exercise 2.5.1:

What is the communication cost of each of the following algorithms, as a function of the size of the relations, matrices, or vectors to which they are applied?

2.5.1 (a)

The matrix-vector multiplication algorithm of Section 2.3.2.

Our matrix-vector multiplication algorithm $(M \cdot v)$ produces a key value pair for each entry in the matirx M. The communication cost is therefore $O(r \times c)$ where r and c are the number of rows and columns of M.

2.5.1 (b)

The union algorithm of Section 2.3.6.

Here, for the union of R and S, the mapper funtion passes key value pairs for each entry in R and S. Therefore the communication cost is the total number of entries in R (r) plus the total number of entries in S (s) or $O(r + s)$.

2.5.1 (c)

The aggregation algorithm of Section 2.3.8.

Here the communication cost of grouping relation $R(A, B, C)$ is just the number of tuples (a, b, c) in the relation R.

Exercise 2.6.1:

Describe the graphs that model the following problems.

2.6.1 (a)

The multiplication of an $n \times n$ matrix by a vector of length n.

Each of the n elements in in each row of the matrix will have an edge to the correpsonding row of the result vector. Each element of the original vector has an edge to each element of the result vector.

2.6.1 (b)

The natural join of $R(A, B)$ and $S(B, C)$, where A, B, and C have domains of sizes a, b, and c, respectively.

As described in the text, the inputs are all possible combinations of A and B (a set of length $a \times b$) and all combinations of B and C (a set of length $b \times c$). The outputs are all combinations of A, B, and C (a set of length $a \times b \times c$). We then have edges from each tuple (a_i, b_j) to each tuple (a_i, b_j, c_*) and likewise (b_m, c_n) to (a_*, b_m, c_n).

2.6.1 (c)

The grouping and aggregation on the relation $R(A, B)$, where A is the grouping attribute and B is aggregated by the MAX operation. Assume A and B have domains of size a and b, respectively.

The inputs are all possible combinations of A and B (a set of length $a \times b$) and the outputs are a set of length a consisting of a_i and each value of B. We then have edges between each input (a_i, b_*) and the corresponding output $(a_i, b_1, \ldots, b_b)$.