Summary:

The paper “From Theory to Practice: Efficient Join Query Evaluation in a Parallel Database System” presents a novel approach to efficiently compute complex join queries on a massively parallel architecture. The authors leverage two independent lines of work: a communication-optimal algorithm for distributed evaluation and a worst-case optimal algorithm for sequential evaluation. They also propose practical optimizations for both algorithms.

Traditional engines may become highly suboptimal if a query is cyclic. Some sequential algorithms like HyperCube are described to solve these problems, which has a worst-case optimal computation result, but they still have some unpractical assumptions such as integral partitions of the available servers. The authors therefore present a practical algorithm to compute optimal configuration for HyperCube and used it in conjunction with a optimization algorithm for choosing variable order of the Tributary join to efficiently decrease the cost in the communication od join queries.

The authors evaluate the performance of different shuffling and joining operations in the aspects of query times, CPU workload and number of shuffled tuples in real datasets to prove the HyperCube combined with Tributary may be the best choice in some circumstances, but it’s still influenced by many factors such as scaling intermediate results.

The authors then gave optimization for HyperCube shuffling and Tributary joining operations. For HyperCube it used breadth-first-search to enumerate integral configurations and maximize the workload, and use the minimize physical servers for partitions. For Tributary it gave the optimal order of variables with the lowest estimated cost in an order of magnitude.

Strong points:

1. Evaluate the performance of different types of shuffling and joining algorithms not only on query times but also on the overall resource utilization and real datasets. This improves the evidence in practical and analyze the performance more precisely.
2. It scaled different sizes of number of joins, input and intermediate results to show that there is no overall best query plan and gave the method to choose different operators concerned of related aspects, which can be really useful to broaden conclusions to similar problems.
3. The authors gave the potential using on sequential multiway join algorithms in the related work part.

Weak points:

1. **Tributary join needs to be coupled with HyperCube shuffle instead of other naïve shuffling functions like broadcast, which may cause large sorting costs.**
2. **In an acyclic query with small intermediate results, regular shuffle outperforms the broadcast and Tributary join for replicating less origin data.**
3. **In the large intermediate results, broadcast may have a better runtime by having less skew in both computation cost and CPU workload.**

Xiaoyan Xue
07.12.2023