Technical Perspective: Data Distribution for Fast Joins

Leonid Libkin
School of Informatics, University of Edinburgh
libkin@inf.ed.ac.uk

A model database theory paper is usually assumed to have several key ingredients:

- it should consider a real data management problem that is of interest in practice;
- it should provide a clean and simple formalism that can be followed by theoreticians and practitioners alike;
- it should provide theoretical results that give us insights about the original practical problem.

In your favorite database theory papers you will surely find all these three ingredients. A recent paper that has them as well – and that serves as the basis for the highlights paper that follows – is the PODS 2015 paper by Ameloot, Geck, Ketsman, Neven, and Schwentick that considers single-round multi-way join algorithms in parallel systems. This brief overview explains why this is so, and hopefully convinces you to read the full highlights paper.

The problem.

Large-scale data analytics and massive parallelism are two concepts that go hand-in-hand; hence the problem of efficient evaluation of join queries is one that is actively studied. The challenges are quite different from the usual join processing, as the dominant factor is no longer the I/O, but rather communication cost. The most drastic way to reduce it is to have just a single round of communication: that is, distribute data to servers, let them do their work, and then collect the results to produce the answer to the join query.

Afrati and Ullman's EDBT 2010 paper initiated the study of such multi-join algorithms. A refinement, *Hypercube*, algorithm was proposed in a PODS 2013 paper by Beame, Koutris, and Suciu and then experimentally evaluated. In those algorithms, the network topology is a hypercube. To evaluate a query, one replicates each tuple in several of its nodes and then lets each node perform its computation.

While the hypercube is a rather natural distribution policy, it is certainly not the only one. But can we reason about single-round join evaluation under *arbitrary* distribution policies?

Also, distribution policies are query-dependent. While finding one policy for all scenarios is of course unrealistic, what about a more down-to-earth requirement: if we already know that a policy works for a query Q, perhaps we can use *the same* policy for another query Q', without redistributing data? These are the questions addressed in the paper.

The formalism.

It is very simple and elegant. A network is a set of node names; a distribution policy simply assigns each fact (a tuple in a relation) to a set of nodes. This is the communication round. The query Q is then executed locally at each node. It is *parallel-correct* if such a distributed evaluation gives the result of Q; that is, tuples in the answer to Q are exactly those that are produced locally at network nodes.

Next, if we have two queries Q and Q', and we know that each distribution policy that makes Q parallel-correct does the same for Q', we say that parallel-correctness transfers from Q to Q'. In this case, the work done for Q in terms of looking for the right distribution policy need not be re-done for Q'.

The results, and what they tell us.

This is a theory paper, and the main results are about the complexity of checking parallel-correctness and parallel-transferability. The paper concentrates on the class of *conjunctive queries*, i.e., slightly more general queries that multi-way joins.

Parallel-correctness, under mild assumptions, is Π_2^p-complete. That is, it is a bit harder than NP or coNP, but still well within polynomial space. In practice, this means that checking whether a distribution policy guarantees correctness for all databases can be done in exponential time. Note that this is a static analysis problem (the database is *not* an input), and exponential time is tolerable and in fact the expected best case for conjunctive queries (as their containment is NP-complete).

The paper then shows that the same problems for conjunctive queries with negations requires (modulo some complexity theory assumptions) *double*-exponential time, i.e., is realistically unsolvable, which means that one needs to restrict attention to simple joins.

Finally, parallel-transferability for conjunctive queries is solvable in exponential time (remember, this is a problem about queries, not about data), and importantly it is in NP for many classes of conjunctive queries, likely multi-joins (which hints at the possibility of using efficient NP solvers to address this problem in practice).

In summary, the paper provides an elegant theoretical investigation of a practically important problem, and gives a good set of results that delineate the feasibility boundary.