Efficiently Massively Parallel Join Optimization for larger queries

Data analytics is very important for understanding business and making important decision to maximize profits. Usually the analytical queries are generated automatically but it can be big in size and can contain hundreds of tables. Prior works have focused on sophisticated data layouts, scale out systems or JIT code generation. However, query optimization is often neglected. It is an NP hard problem and therefore difficult to solve. For eg. Postgres takes 160 secs for 21 relation join query and sparksql takes 1000 sec for 18 relation for generating an optimal plan. To handle this, different systems implement a heuristic threshold. If the optimal plan is missed, it takes a lot of time since finding the right heuristic is difficult. In this paper, they use Dynamic Programming(DP) for join order optimization. They avoid cross product since they do not form an optimal join order. The efficiency is determined based on two parameters: number of join pairs evaluated and parallelizability. Here, a novel parallel join order optimization MPDP that can be executed over GPUs or CPUs. They augment MPDP with IDP to explore larger state space. With the help of this algorithm, the postgres time limit is pushed from 12 to 25 relation. MPDP can be incorporated into IDP2. IDP2 might get stuck because of poor local optimum hence they have tried UnionDP which leverages the graph topology for such queries. The results shows that the MPDP(GPU) optimization time is much less than Postgres. MPDP scales sub-linearly beyond 6 threads. Hence they have chosen different AWS size instance. For smaller queries, postgres and DPCCP are cheaper. MPDP is cheaper for larger queries. Hence, this heuristic solution allows to explore larger search space for very large join queries.

Comments:
I really appreciate them mentioning lower is better in the figure 2.
Having the option of cpu and gpu modes makes it more independent of the architecture which is good if some startup doesn't have enough funding to get a gpu.
Is this approach better or the deep learning approach of Balsa is better? Which model takes less latency?
I don't think I understand what does collaborative context collection actually means and how it avoids the if condition divergences. Also would really appreciate a refresher of graphs terminology during the paper discussion.
I think they did a very good job with the benchmark evaluation unlike the steered query paper.