Performance Evaluation and Analysis of GPU Join Processing

Wenbo Sun, Rihan Hai
TU Delft

With the development of parallel hardware and instructions, many studies have proposed efficient parallel algorithms to accelerate relational join processing. Recent emerging high-performance hardware further improves parallelism and memory bandwidth by order of magnitudes, which inspires a number of studies on GPU-compatible join algorithms.

[3] conducted a systematical study on 4 major join algorithms (sort-merge join, hash join, indexed nested loop join, and non-indexed nested loop join), disassembling them into 6 parallel primitives and implementing the algorithms with their combinations. The results show the GPU implementations significantly outperformed CPU-based ones up to seven times. Their follow-up research [2, 5] evaluated the algorithms on up-to-date hardware and gained much higher speedup due to an enlarged performance gap between CPU and GPU.

Alternatively, [1] proposed a method evaluating relational join operation with Matrix Multiplication (MMJoin). The method extracts a common domain from the keys of two tables to be joined, then constructs matrices indicating the positional relations between tuples and the domain. The join result can be computed through the multiplication of the matrices. Yet, the best-known MM algorithm has $O(n^{2.373})$ complexity which is obviously higher than $O(M+N)$ complexity of indexed join, making MMJoin not widely used in traditional database systems.

However, the MMJoin shows potential in the context of massive arithmetic processing units in modern GPUs. In contrast to the memory-bounded GPU join algorithms, the MMJoin performs intensive arithmetic computing. The difference in execution patterns also implies that, in some cases, MMJoin may outperform the classic GPU join algorithms. TCUDB [4] implemented the multi-way MMJoin and aggregations with Sparse Matrix Multiplication (spMM). The performance evaluation on real datasets shows that the spMM join outperforms the hash join [6] by up to 5 times. Nevertheless, the experiments only covered limited data characteristics and didn't compare with the 4 classic GPU join algorithms.

In our ongoing research, we re-evaluate the TCUDB with synthetic datasets covering extensive data characteristics (e.g., shape, selectivity, skewness, etc.). Furthermore, we extend the evaluation with multiple metrics, including execution time, memory bandwidth usage, peak memory usage, and GPU core utilization. Apart from experimental evaluation, we also attempt to explore the turnover point of operator choice between MMJoin and classic join through quantitative performance models. The result of such analysis will contribute to novel cost models for query processing integrated with linear algebra.

References

[1] R. R. Amossen and R. Pagh. Faster join-projects and sparse matrix multiplications. In *Proceedings of the 12th International Conference on Database Theory*, pages 121–126, 2009.

[2] B. He, M. Lu, K. Yang, R. Fang, N. K. Govindaraju, Q. Luo, and P. V. Sander. Relational query coprocessing on graphics processors. *ACM Trans. Database Syst.*, 34(4), dec 2009.

[3] B. He, K. Yang, R. Fang, M. Lu, N. Govindaraju, Q. Luo, and P. Sander. Relational joins on graphics processors. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 511–524, New York, NY, USA, 2008. Association for Computing Machinery.

[4] Y.-C. Hu, Y. L. Li, and H.-W. Tseng. Tcudb: Accelerating database with tensor processors. In *Proceedings of the 2022 International Conference on Management of Data*, 2022.

[5] J. Paul, B. He, S. Lu, and C. T. Lau. Revisiting hash join on graphics processors: A decade later. *Distributed and Parallel Databases*, 38(4):771–793, 2020.

[6] Y. Yuan, R. Lee, and X. Zhang. The yin and yang of processing data warehousing queries on gpu devices. *Proceedings of the VLDB Endowment*, 6(10):817–828, 2013.