SQL Scalability

Ali Hassan

January 27, 2026

Abstract

SQL is scalable vertically but not horizontally. As you can see in section 1.

1 The performance of spark SQL library

As we can see from figure 1, spark SQL library has an organic dependency on sql table join. How to evaluate the performance of spark SQL library? This figure answers this question by reduction. Finally, the performance of spark SQL library depends on either the performance of distributed memory parallel merge sort or the performance of distributed memory sample merge sort, etc. The idea is: if we can have a scalable sort algorithm, then we will have a scalable spark SQL library. Do we have a scalable sort algorithm? The answer is NO. We do not have a scalable sort algorithm. To be more precise: we do not have a horizontally-scalable sort algorithm on a distributed memory cluster.

To convince the kind reader, we will prove this argument using two principal sorting algorithms : merge sort and sample sort. Other algorithms are less scalable.

1.1 Merge sort

Merge sort was one of the first sorting algorithms where optimal speedup was achieved, with Richard Cole using a clever subsampling algorithm to ensure constant time merge [par, cre].

But this is a shared memory machine, or what is called Parallel RAM PRAM. This is not a distributed machine algorithm. In other words, the above mentioned algorithms are totally not-applicable over distributed memory cluster, ie, they are not applicable over spark cluster. Figure 1 explains the dependence of spark sql on strictly distributed memory variant of parallel algorithms. This characteristic has played a pivotal impact over the performance of the spark SQL library.

From [JK02], we quote: merge sort is useful in sorting a great number of data progressively, especially when they can be partitioned and easily collected to a few processors. Merge sort can be parallelized, however, conventional algorithms using distributed memory computers have poor performance due to the successive reduction of the number of participating processors by a half, up to one in the last merging stage. Minsoo Jeon et al. present load-balanced parallel merge sort where all processors do the merging throughout the computation. There results show a speedup of 8.2 on 32-processor in sorting of 4M integers.

Speedup of 8 on a 32 nodes is far from expected practical scaling.

1.2 Sample sort

Regarding sample sort [ABSS15], Axtmann et al present a multi-level generalizations of the known algorithms sample sort. In particular their sample sort variant turns out to be very scalable for large p and moderate n, where p is number of nodes and n is the size of list to sort.

While such achievement might have positive applications it is far from the context of spark sql where p is unlimited, small or large, and n is expected to be extra large. Based on what has been shown we can conclude that sample sort is not a candidate for high-performance horizontally scalable algorithm for spark SQL library

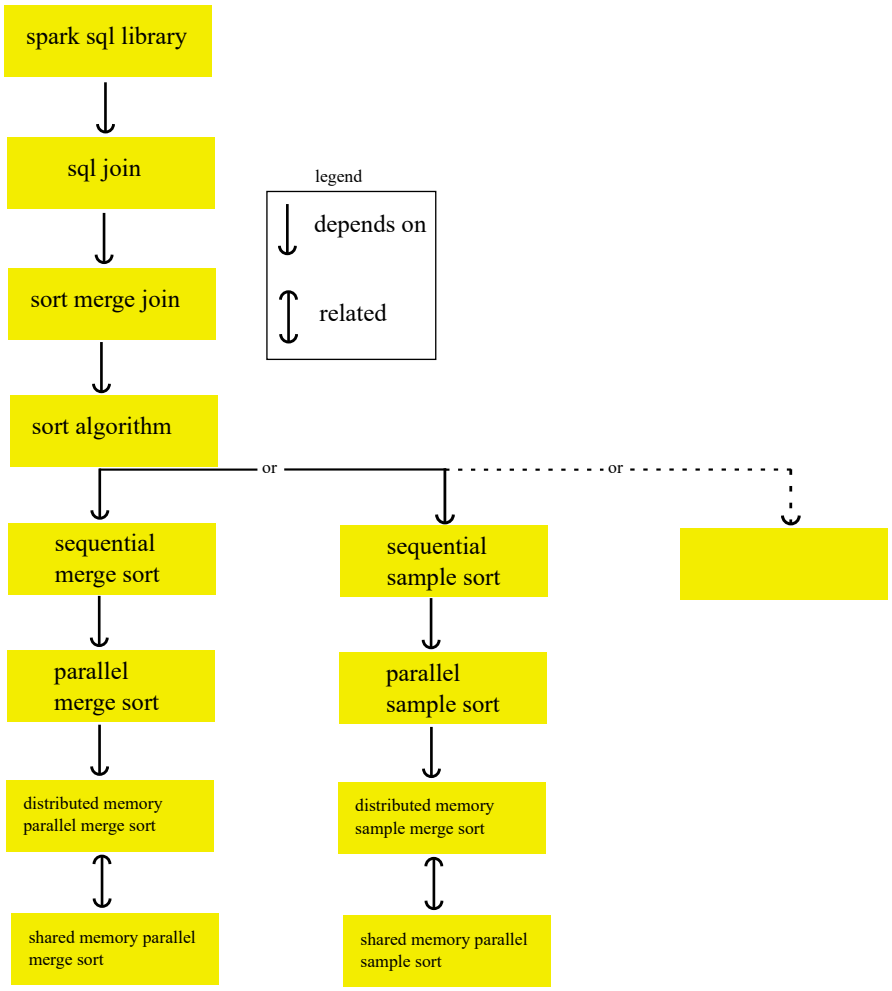

Figure 1: The performance of spark SQL library. As we can see in this figure, spark sql library has an organic dependency on sql table join. How to evaluate the performance of spark sql library? This figure answers this question by reduction. Finally, the performance of spark sql library depends on either the performance of distributed memory parallel merge sort or the performance of distributed memory sample merge sort, etc.

1.3 Conclusion

Back to figure 1, we go buttom up to conclude that spark sql library has limited horizontal scalability based on the scientific absence of a scalable sort algorithm that can be implemented in spark. And as speedup is the best scalability measure, we conclude that we can not expect a good speedup on any scala spark application that uses a spark sql component.

2 Todo

The philosophy of SQL RDBMS is an artificial philosophy based on the concept of key. A relational database that means we have a key that links each table with the other table and the other table. So we have a net of linked tables that SQL Engine can use to have a logical view, which is fundamentally irrelevant to the original data, and this should be investigated because this net of tables is fundamentally counter-performance in big data, so we have to remove it from our design and architecture. And this will be my work in the future. 20 years, please add this as a big chapter with a fundamental use case.
As you can see in section 1

References

[ABSS15] Michael Axtmann, Timo Bingmann, Peter Sanders, and Christian Schulz. Practical massively parallel sorting. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '15, page 13–23, New York, NY, USA, 2015. Association for Computing Machinery.

[cre] Parallel ram. https://en.wikipedia.org/wiki/Parallel_RAM. From Wikipedia, the free encyclopedia.

[JK02] Minsoo Jeon and Dongseung Kim. Parallelizing merge sort onto distributed memory parallel computers. volume 2327, pages 25–34, 05 2002.

[par] Parallel merge sort. https://en.wikipedia.org/wiki/Merge_sort. From Wikipedia, the free encyclopedia.