Fast Distributed Complex Join Processing

This abstract presents a new approach for optimizing big data analytics in parallel distributed systems. The authors note that previous works have primarily focused on reducing the communication cost of complex join queries, but have neglected the potential computational intensity of the query. The authors propose the Adaptive Distributed Join (ADJ) approach to address this issue by co-optimizing communication, pre-computing, and computation costs. The authors claim that their approach finds an optimal query plan by exploring cost-effective partial results through cost-estimation by sampling. The results of their experiments indicate that ADJ outperforms existing multi-way join methods by a substantial margin.

Overall, the abstract provides a clear and concise overview of the problem being addressed and the solution proposed. The results of the experiments provide strong evidence for the effectiveness of the ADJ approach. However, the abstract could benefit from a clearer explanation of the specific advantages of the ADJ approach over existing methods, as well as a more detailed description of the cost-estimation process. Additionally, the abstract could be more impactful with a discussion of the potential applications and real-world implications of the ADJ approach.

This is a well-written introduction to a research paper that aims to reduce the total cost of processing join queries in data analytics engines. The authors provide a clear overview of the current state-of-the-art (the HCubeJ algorithm) and its limitations, which is the unbalance between computation and communication cost. The authors then state their main contributions, which are: (1) introducing a mechanism to trade computation cost with communication and precomputing cost, (2) finding cost-effective precomputed partial results from the huge search space, and (3) conducting extensive performance studies to confirm that their approach is faster than previous approaches in terms of the total cost. The introduction provides a clear overview of the problem and the authors' solution, and sets the stage for the rest of the paper.

This passage provides an overview of the Leapfrog and HCube Join algorithms for evaluating join queries in a distributed database system. The Leapfrog algorithm is described as a state-of-the-art sequential join algorithm that processes the join query by finding the tuples iteratively based on the attribute order. The HCubeJ algorithm is built on two algorithms, HCube and Leapfrog, where HCube is a communication optimal shuffling method and Leapfrog is the in-memory sequential multi-way join algorithm that processes the join query at each server. The algorithm is described in a technical manner, and it would be beneficial for the reader to have a background in database systems and algorithms.

The paper presents a method to minimize the total cost of communication cost, computation cost, and pre-computing cost for a join query in a database system. The authors first present an example to illustrate their main idea and then formalize their problem statement. The problem is to find the optimal query plan, which is the combination of a query candidate and an attribute order, that minimizes the total cost.

The authors propose a prototype system called ADJ, which reduces the search space by considering only a limited number of joins that lower the join cost and explores cost-effective query plans based on an optimal hypertree constructed for the query. The cost model and the cardinality estimation are also discussed in the paper.

The paper is well-organized and presents a clear problem statement. The authors' approach to reducing the search space and exploring cost-effective query plans is intuitive and appears to be well thought-out. However, more details on the implementation and evaluation of the proposed system would be helpful in understanding the effectiveness of the approach.

This related work section provides a good overview of previous works on distributed multiway join. The authors highlight the limitations of traditional methods such as Spark and introduce the recent approaches such as HCube and HCubeJ. The comparison between previous works and their own work is well established. However, the section could benefit from more specific details and analysis on how these previous works influenced the design of their own work, and how their work differs from and advances these previous works. Additionally, the section could be improved by including more recent related works that have been published since 2021.

The paper presents an experimental study on the performance of a proposed method called ADJ for complex join queries in a distributed environment. The authors compare ADJ with four state-of-the-art methods, SparkSQL, HCubeJ, HCubeJ + Cache, and BigJoin, on a cluster of a coordinator server and 7 follower servers using wall clock time as the measurement of performance.

The study investigates the effectiveness of attribute order pruning and the cost and effectiveness of co-optimization in the performance of ADJ. Results showed that valid attribute orders produced fewer intermediate tuples compared to invalid attribute orders and co-optimization effectively trades computation with communication with a low query optimization cost.

The scalability of the system was also tested by varying the number of workers of Spark from 1 to 28. Results showed near-linear speed up on query Q2 and Q3 and limited scalability on query Q1.

Overall, the paper presents a thorough experimental evaluation of the performance of ADJ in a distributed environment and provides insights on the benefits and limitations of the method.

The conclusion of this paper is well-written and provides a clear overview of the research performed. However, there are a few suggestions for improvement:

Add a sentence to summarize the results of the extensive experiments. It would be helpful to have a quantitative or qualitative statement on how the proposed method has been shown to be effective.

Add a sentence to put the results in context and explain their significance. For example, how does the proposed method compare to existing methods in terms of performance?

Consider adding a sentence or two to discuss any limitations of the proposed method and potential areas for future work.

Consider revising the language to make it clearer and more concise. For example, instead of saying "This paper studies the problem of co-optimize communication and computation cost in a one-round multi-way join evaluation", consider saying "This paper investigates co-optimizing communication and computation cost in a one-round multi-way join evaluation."