ELEN 4020 Project - Comparison of Parallel Equi-Join using MPI and OpenMP

Uyanda Mphunga - 1168101, Darren Blanckensee - 1147279, Ashraf Omar - 710435
and Amprayil Joel Oommen - 843463

***Abstract*—**

I. Introduction

The join operation concerns the combining of two different tuples on a common join attribute [1]. It is important in information systems, in particular the use of databases since the join operation is the most expensive operation in database query operations in terms of time and data-intensity [2]. This project explores two different approaches to performing a join of two very large tables. The problem is described in Section II and the two different approaches discussed in Sections III and IV. The method of experimentation is outlined in Section V and the results of the experiment are analysed in Section VI. Section VII reviews the project and concludes this document.

II. Problem Description

The objective of this project is to perform an equi-join of two very large tables. An equi-join is a type of join that uses the equality operator as a basis for the join [3] that is if the join attribute in $R_1(A,B)$ is strictly equal to the join attribute in $R_2(A,C)$ the result of the join is inserted into a third table $R_3(A,B,C)$. An illustrated example can be seen in Figure 1. The join must be done using two different algorithms, one that is based in MPI (Message Passing Interface), and one that uses another high-level parallel programming model. The programming model chosen in this project is OpenMP. The two programmes need to be compared with one another in terms of speed-up and scalability when increasing the number of processors and nodes that the program uses. The MPI join algorithm being used is hash-join and the OpenMP join algorithm being used is merge-join. The speed-up comparison is performed by running the two programs with increasing number of processors and the scalability comparison is conducted by increasing the number of nodes the program uses in a cluster.

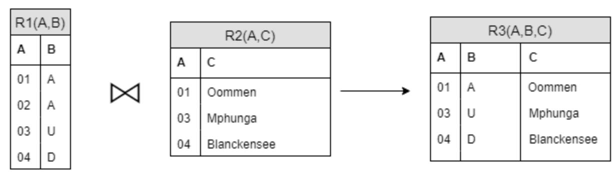

Fig. 1. An example of equi-join between two relational tables

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

III. MPI Join Algorithm – Hash-Join

An overview of the algorithm being used is listed in Algorithm 1

Data: Table $R_1(A, B)$, Table $R_2(A, C)$
Result: $R_3(A, B, C)$
for *all key-value pairs in* R_1 **do**
    read key;
    calculate hash as sum(ASCII) of key;
    calculate (hash%(no. of processes - 1) + 1);
end
send key-value pairs of $R_1(A, B)$ to slave nodes in vector;
for *all key-value pairs in* R_2 **do**
    read key;
    calculate hash as sum(ASCII) of key;
    calculate (hash%(no. of processes - 1) + 1);
    //This is to determine which slave node the key-value pair is sent to.
end
send key-value pairs of $R_2(A, C)$ to slave nodes in vector;
IN EACH SLAVE NODE:
for *all key-value pairs in each vector* **do**
    if *key from* R_1 == *key from* R_2 **then**
        write key-value output to R_3
    end
end

Algorithm 1: MPI Implementation of hash-join

IV. OpenMP Join Algorithm – Merge-Join

The OpenMP join algorithm being implemented is merge-join. The two table are sorted by join attribute. Then the table are scanned and join attributes are compared to one another. For an equi-join, if the join attribute of one entry is strictly equal to the join attribute of the other table's entry, the entries are joined in the output table [4].

V. Experiment Description

VI. Analysis of Results

VII. Conclusion

References

[1] C. Yu and W. Meng, *Principles of Database Query Processing for Advanced Applications*, ser. The Morgan Kaufmann series in data management systems. Morgan Kaufmann, 1998. [Online]. Available: https://books.google.co.za/books?id=aBHRDhrrehYC

[2] P. Mishra and M. H. Eich, "Join processing in relational databases," *ACM Comput. Surv.*, vol. 24, no. 1, pp. 63–113, Mar. 1992. [Online]. Available: http://doi.acm.org/10.1145/128762.128764

[3] w3resource.com. (2018) Sql equi join. Last Accessed: 2018-05-14. [Online]. Available: https://www.w3resource.com/sql/joins/perform-an-equi-join.php

[4] A. Pavlo. (2017) Lecture 19 parallel join algorithms (sorting). Carnegie Mellon University. Online Lecture Slides. [Online]. Available: https://15721.courses.cs.cmu.edu/spring2017/slides/19-sortmergejoins.pdf