IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.11691

Response Time Optimization for Skyline Queries

Harshal Bodhare[1], Abhishek Bhosale[2], Amit Bhadke[3], Shayan Shaikh[4], Dr. Rupali Kulkarni[5]

Student, Computer Engineering, K.K.W.I.E.E.R, Nashik, India[1-4]

Professor, Computer Engineering, K.K.W.I.E.E.R, Nashik, India[5]

Abstract: Skyline queries are one of the most commonly used query operators for locating query results that only return data items whose dimension vector is not dominated by any other data item in the database. Skyline queries have been included into various sorts of databases, including complete, incomplete, and uncertain, due to their utility and ubiquity. In the early phases of a knowledge-discovery process, the skyline query and its variant queries are useful functions. The skyline query and its variants identify a collection of important items that outperform the dataset's other common objects. Such knowledge-discovery queries must be computed in parallel distributed systems in order to manage huge data. We will use typical datasets in this research to parallelize skyline computations utilising various strategies such as parallel data pre-processing, parallel skyline computation using multithreading, and multiprocessing. The ultimate goal is to reduce response time using parallel computation. We shall be able to give the approach for efficient, parallel skyline calculation near the end of the paper.

Keywords: Skyline query, Preference queries, Skylines, SQL, Algorithms, Database, Multithreading, multiprocessing.

I. INTRODUCTION

Skyline queries return tuples that are "promising" in terms of the user's interest dimensions. Users frequently query popular datasets, and the dimensions of the user searches frequently overlap. Repeating computations on big datasets for such frequent, overlapping skyline queries results in unacceptable response time. In cases when the query dimensions differ somewhat from the common dimensions, reusing earlier results can help avoid or reduce additional computing expenses. This project focuses solely on this issue, with the goal of improving the response time of results generated against popular and large databases. We are frequently asked to conduct operations on huge datasets. If there are more than two parameters by which the query should be searched, the time it takes to perform the query increases. So we're using the skyline query to speed up the process.

II. BACKGROUND

Mingjie Tang, Yongyang Yu, Walid G. Aref, Qutaibah M. Malluhi, and Ouz- zani[6] introduce a new efficient three-phase technique for Skyline requests are processed on such a huge scale. During the early phase of pre-processing, the Z-order curve is used to partition data. They make use of an updated information partitioning technique that treats data partitioning as a problem of optimization to keep intermediate data as small as possible each compute in the second phase. The node divides the data data points into separate subsets and then performs the calculations. Each subset's skyline calculation to produce skyline candidates in parallel. They generate an index and use an efficient algorithm in the final phase to merge the skyline candidates that were generated.

S. Borzsony, D. Kossmann, and K. Stocker [2] propose using a Skyline operation to extend database systems. This operation eliminates a group of items. A set of intriguing data items from a possibly vast set of point has been made if it isn't dominated by another topic, it's intriguing. They demonstrate how SQL can be useful. Added the ability to pose Skyline queries, as well as display and compare different algorithms. To put the Skyline operation into action and demonstrate how it can be computed in conjunction with other database processes. They demonstrated how a database system can be used. To compute the Skyline of a set of points, the function can be extended. We proposed the SKYLINE OF clause as a simple addition to SQL's SELECT statement, presented and experimentally evaluated alternative Skyline computation algorithms, discussed how indices can be used to support the Skyline operation, and described how the Skyline intersects with other query operators.

Wei-Mei Chen, Meng-Zong Liou, Yi-Teng Shu [5] The skyline query is a powerful data analysis technique for generating multi-criteria decisions. The database community gave it a lot of attention. As multi-core architectures become more common, they introduce a novel parallel skyline query algorithm that can be used on multi-core and multiprocessor systems to retrieve skyline points in a progressive manner as they are identified. They present their findings in this paper suggested a parallel skyline technique that can reduce unnecessary computations and improve skyline query parallelism.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022
DOI: 10.17148/IJARCCE.2022.11691

Experiment outcomes demonstrate that our technique successfully uses the properties of many cores to increase skyline calculation performance for huge high-dimensional datasets.

Foto N. Afrati, Paraschos , Koutris, Dan Suciu, Jeffrey , D. Ullman [1] They design and assess parallel skyline query techniques in this paper. They employ both the MP model proposed by Koutris and Suciu (2011) and a variant of the model proposed by Afrati and Suciu (2012) as a framework for parallel computation. They present a 2-step technique for any dataset dimension in the KKWIEER, Department of Computer Engineering 2021-2022 5 MP model, as well as a 1-step algorithm for 2 and 3 dimensions. Finally, they provide a 1-step technique for any number of dimensions in the GMP model, as well as a 1-step algorithm for uniform distributions of data points in the MP model.

III. METHODOLOGY

- **Brute Force:** The brute force solution is simply to calculate the total distance for every possible route and then select the shortest one. This is not particularly efficient because it is possible to eliminate many possible routes through clever algorithms. The time complexity of brute force is O(mn), which is sometimes written as O(n*m). So, if we were to search for a string of "n" characters in a string of "m" characters using brute force, it would take us n * m tries.

- **Block Nested Loop (BNL):** The Block Nested Loop approach is easy and fastest amongst the linear searching algorithms. It maintains a list of skyline points which is initialized to the first element of dataset, and then iterated through the dataset to find a dominating tuple, if a dominating tuple is found then it is added to the skyline list and the non-dominating tuple is removed from the list. If there comes a tuple which is neither dominating or non-dominating then that tuple is added to the skyline list. Here, there are one outer loop and one nested loop. the outer loop iterates through skyline list and the nested loop iterates through dataset. Hence this makes the total iteration much smaller.

- **Multithreading:** Multithreading is a model of program execution that allows for multiple threads to be created within a process, executing independently but concurrently sharing process resources. Depending on the hardware, threads can run fully parallel if they are distributed to their own CPU core.

    - Algorithm for Multithreading:
        1. Divide data set into n parts.
        2. Create n threads.
        3. Assign each part to single thread.
        4. Get result of each thread.
        5. Combine results of each thread.
        6. Pass this result to another single thread.
        7. And get finalize result.
        8. Calculate response time taken by above steps.

- **Multiprocessing:** Multiprocessing, in computing, a mode of operation in which two or more processors in a computer simultaneously process two or more different portions of the same program (set of instructions). Multiprocessing is the use of two or more central processing units within a single computer system. The term also refers to the ability of a system to support more than one processor or the ability to allocate tasks between them.
    - **Algorithm for Multiprocessing:**
        1. Divide data set into n parts.
        2. Create n processes.
        3. Assign each part to single process.
        4. Get result of each process.
        5. Combine results of each process.
        6. Pass this result to another single process.
        7. And get finalize result.
        8. Calculate response time taken by above steps.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.11691

IV. RESULTS

- **Response time for All Algorithms:**

Dataset Size	SQL	Brute Fore	BNL	MultiProcessing	Multithreading
100000	1.470	1.778	0.967	0.489	0.606
200000	2.781	3.809	1.952	0.802	1.161
300000	5.538	5.642	2.957	1.165	1.746
400000	6.161	7.502	3.878	1.481	2.261
500000	8.781	9.849	4.885	1.766	2.829
600000	7.895	11.996	5.866	2.132	3.422
700000	11.334	14.815	6.862	2.601	4.010
800000	12.270	16.516	7.946	3.290	4.581
900000	12.594	19.398	8.760	3.144	5.781
1000000	16.686	23.401	11.070	3.512	6.566

BNL = Block nested Loop

Fig 1: Response Times for All Algorithms

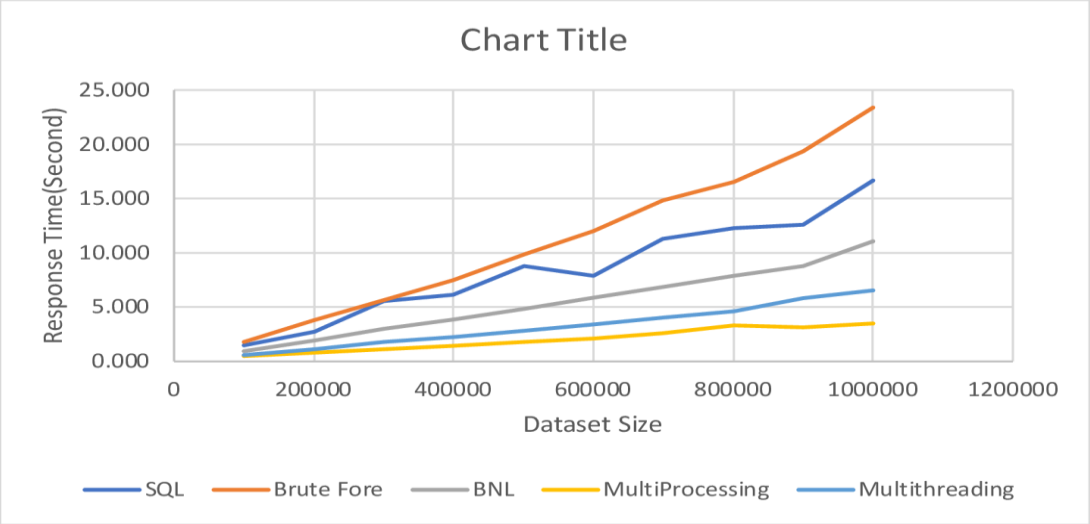

Fig 2: Response Time Graph for All Algorithms

In the above table (Fig. 1), the response time is noted for various methods and algorithms like SQL, Brute Force, Block Nested Loop (BNL), Multithreading, and Multiprocessing. In the graph (Fig. 2), we have plotted the response time vs dataset size graph. From the graph, it's clear that Brute Force and SQL take more response time compared to other algorithms. Furthermore, when multiprocessing and multithreading are applied to BNL, it gives a less responsive time than BNL. Here we have fixed the number of threads and processes to 4.

- **Response Time for Multithreading:**

No of Threads	Time
1	6.053
2	6.200
3	5.990
4	5.966
5	6.000
6	6.110
7	6.019
8	5.995

Data base size : 1000000

Fig 3: Response Time data for Multithreading

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.11691

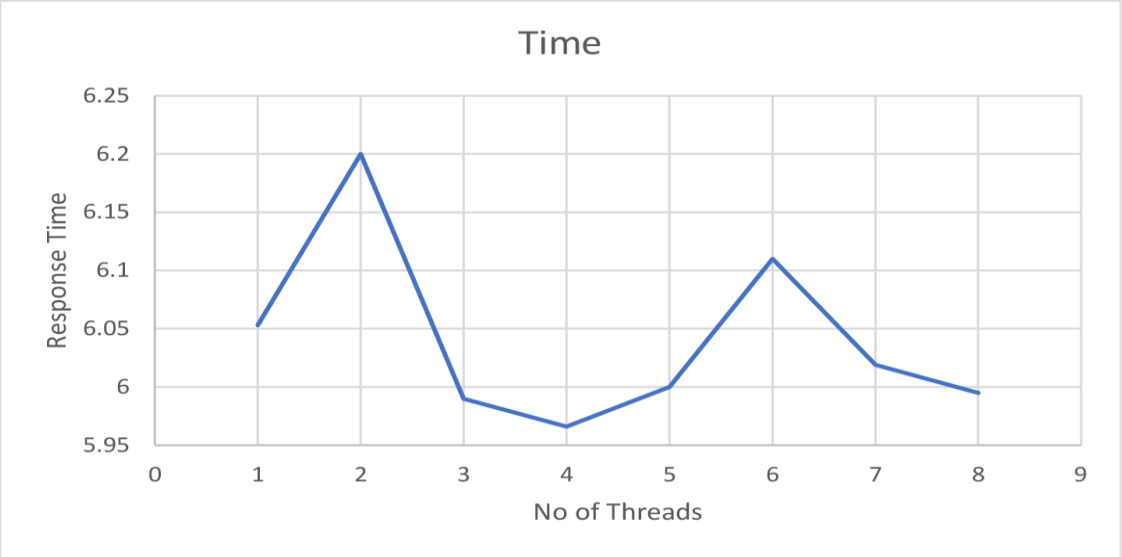

Fig 4: Response Time Graph for Generalized Multithreading.

In the above table (Fig. 3), and graph (Fig. 4), the response time in seconds for multithreading is shown. Here we have fixed the data set size, i.e., 100,000 records and a two-dimensional query. And we have increased the number of threads from 1 to 8. From the above graph, we can say that as the number of threads increases, the algorithm takes less time and after a certain number of threads, it remains constant.

- **Response Time for Multiprocessing:**

No of process	Time
1	0.705
2	0.427
3	0.355
4	0.386
5	0.368
6	0.365
7	0.396
8	0.404

Data base size : 100000

Fig 5: Response Time data for Multiprocessing

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 6, June 2022

DOI: 10.17148/IJARCCE.2022.11691

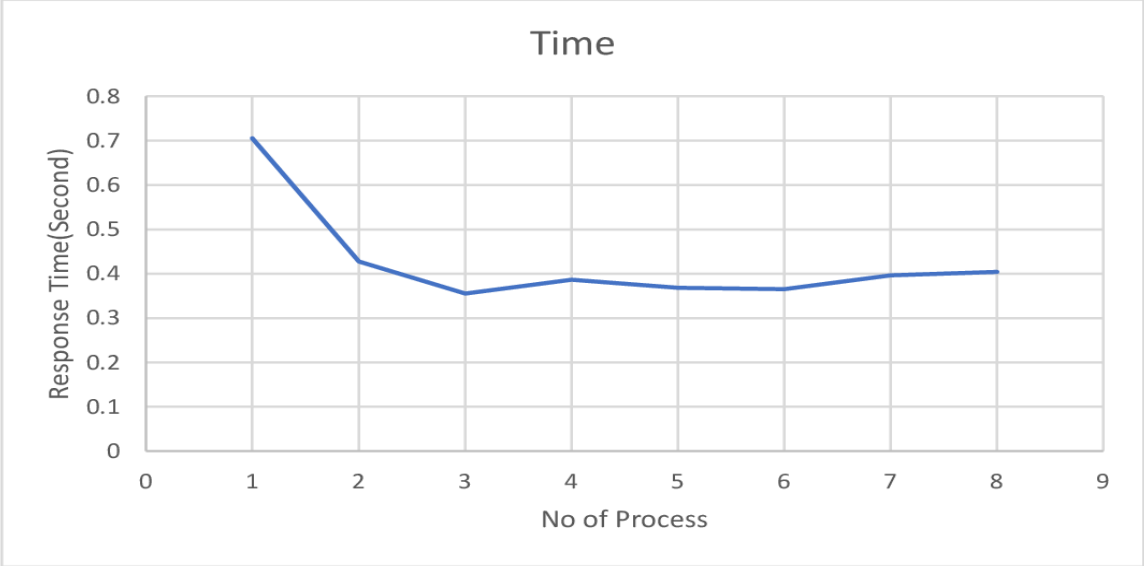

Fig 6: Response Time Graph for Generalized Multiprocessing.

In the above table (Fig. 5), and graph (Fig. 6), the response time in seconds for multiprocessing is shown. For this, we have used two-dimensional queries. As we can see in the table, we have increased the data set size by 100,000 and recorded the response time. Using this recording, we have plotted the graph. From the above graph, we can conclude that as the number of processes increases, the algorithm takes less response time and after a certain number of processes it stabilizes, i.e., it doesn't reduce.

V. CONCLUSION

We compared each tuple with each other after picking the dataset using the brute force method, and created a list in which the dominating tuple was added to the final list. A list is first initialized to the first tuple in a Block Nested Loop (BNL), and subsequently tuples are added or removed according on whether they are dominating or not dominating. As a result of executing the operations of collecting information from datasets utilising various mediums, such as the brute force approach and the BNL algorithm for skyline queries, we discovered that the BNL algorithm for skyline queries yields the best results and delivers the nearest tuple as necessary. Several multi-criteria decision support tools use skyline queries. A skyline query returns the objects in a dataset that cannot be dominated by any other objects given a dominance relationship. We get a faster response time in multiprocessing because the code written is more CPU-based and multiprocessing does it more efficiently compared to multithreading.

REFERENCES

[1]. Foto N. Afrati, Paraschos Koutris, Dan Suciu, and Jeffrey D. Ullman. Parallel skyline queries. Theory of Computing Systems, 57(4):1008–1037, Nov 2015.
[2]. Stephan Borzsony, Donald Kossmann, and Konrad Stocker. The skyline operator. In Proceedings 17th international conference on data engineering, pages 421–430. IEEE, 2001.
[3]. R. Kulkarni and Bashirahamad Momin. Parallel skyline computation for frequent queries in distributed environment. pages 374–380, 03 2016.
[4]. R. Kulkarni and Bashirahamad Momin. Skyline Computation for Big Data. 10 2017.
[5]. Meng-Zong Liou, Yi-Teng Shu, and Wei-Mei Chen. Parallel skyline queries on multi-core systems. In 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies, pages 287–292, 2013.
[6]. Mingjie Tang, Yongyang Yu, Walid G Aref, Qutaibah M Malluhi, and Mourad Ouzzani. Efficient parallel skyline query processing for high-dimensional data. IEEE Transactions on Knowledge and Data Engineering, 30(10):1838–1851, 2018.
[7]. Ji Zhang, Xunfei Jiang, Wei-Shinn Ku, and Xiao Qin. Efficient parallel skyline evaluation using map-reduce. IEEE Transactions on Parallel and Distributed Systems, 27(7):1996–2009, 2015.