Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

ASCHEDULER: ADOPTION OF QUERY PRIORITY IN QUERY SCHEDULING FOR AVOIDING BAD QUERY MIX

M. ABDUL QADOOS BILAL
Taiyuan University of Technology, Taiyuan, 030000, China.

BAONING NIU
Taiyuan University of Technology, Taiyuan, 030000, China.

NAZIR AHMAD *
The Islamia University of Bahawalpur, Bahawalpur, 63100, Pakistan.
*Corresponding Author Email: nazeerrana@iub.edu.pk

Abstract

Performance of databse management systems relates to low response time with maximum throughput. Therefore, it is necessary to adopt such scheme which can execute the workload with fewer time. Such scheme wich can also helpful in parallel query processing or can process batch queries simultenously, which can be used in developing the query mixes. Therefore, this research proposes a novel scheduler named as 'Ascheduler'. This proposed scheduler assigns the priority using modified MQI (Multiple Query Interaction) to each query on the basis of its resource usage for the completion of its execution which is coming for the developing the query mixes. The developed the query mixes by the proposed scheduler is categorized into best, good, moderate and bad quer mixes. This categorization of the query mixes is on the basis of resource utilization and their response time. It also keeps avoidance of developing bad query mixes by using previously developed query mixes response time and resource utilization by keeping record of priority of each query used in query mix as well as an individual query. The novel proposed scheduler named 'Ascheduler' also reduces the overall execution time of workload and perforem better in the other performace metrices like response time, categorization accuracy and interaction improvement score from the existing schedulers like Forst Come First Serve (FCFS) and Shortest Job First (SJF).

Keywords: Multi-Tasking, Query Mix, Query Set, MPL3, Concurrent Queries, Database, Batch Queries, Parallel Processing, Response Time, Query Scheduler.

1. INTRODUCTION

The basic feature of the computer science field is parallel processing. It allows a computer machine to do several jobs simultaneously, and this feature is called multi-tasking. This opens a new area of research in database management systems. Usually, a single query executes its time slot and fetches data. It is quite demanding to develop such solutions that can enhance the performance of a system with huge database size and meet the high demand for data. It's good to run more than one query simultaneously to avail multi-tasking feature of the computer system.

The sequence of arriving of any query for both concurrent executing along with other batch queries and its execution affects its response time [1] [2]. Response time of a query is the usage of resources for that particular query at a specific time. The same scenario applies to the query mix but at a high level because more than two queries run concurrently at the same time in the query mix. Therefore, those concurrent queries need

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

more resources than a single query [3] [4]. We operate within the realm of a database system handling a diverse workload of queries.

These queries are categorized as $Q1, Q2, \ldots, QT,$ each uniquely characterized by its attributes. These query types possess the flexibility to be instantiated with a range of parameter values, generating numerous distinct query instances. Importantly, each of these query templates is treated as an individual query type.

Parameters: Our optimization problem involves the following parameters:

- *T*: The set of query types, denoted as *Q*1, *Q*2 ,…, *QT*
- *N*: The total number of unique query instances resulting from instantiating query templates with different parameter values.
- *P*: is indeed used in the optimization problem to model the flexibility and customization of query templates, and its plays a vital role in determining the comp ability of query instance with specific time slots.

Variables: We introduce binary decision variables $Xijt$. Representing whether query instance *i* of query type *j* is scheduled for execution at time slot t. These binary variables are integral to modeling the scheduling process.

Objectives: The primary objective is to optimize query scheduling in order to minimize the overall execution time. This can be mathematically expressed as:

Minimize: = $E = \sum i, j, tC(i) \cdot Xijt$

Here, *E* represents the overall execution time, and $C(i)$ is the execution time of query instance *i*.

Constraints: To ensure a feasible solution, we have several constraints:

- *Unique Assignment Constraint*: Each query instance is assigned to exactly one time slot:

$$\sum 1 \sum tXijt = 1 \; for \; all \; i \; and \; j.$$

- *Compatibility Constraint: Each* query instance should be compatible with the parameter values configured for the specific time slot:

$Xijt = 0$ If the parameters of query instance *i* are not compatible with the configuration at time slot t.

Our optimization problem seeks to devise an optimal query execution schedule within a database system, catering to report generation needs. By minimizing the overall execution time while adhering to a set of constraints, we aim to enhance the operational efficiency of the system. The binary decision variables Xij play a key role in determining the scheduling of query instances, thus achieving the desired optimization objectives.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

For measuring the performance of any database system, it is needed to deal with workload management because there are several requests for execution of the database or several queries, from which different queries are executed at different times. Many other factors like resource allocation, resource availability, and data stored on different geographical locations on a disk.

As with the passing of each day, the data volume of databases increases at a very high rate. It becomes a primary reason for decreasing the performance of the database. For any human, it is impossible to manage such large and diverse data. As mentioned earlier, it created the need to build such databases that can handle the issue, as mentioned earlier [1] [2].

Prediction of database queries response time plays a crucial role while managing massive database systems, especially in the workload, which is the result of running various queries at the same time on the request of users because it may enable Database Administer (DBA) to know system behaviour, which helps him in coordination with system and provides some assessment about its performance [3].

Waiting of resources for execution of concurrent queries is based on resource contention, which depends on queuing theory. Any computational graph containing computing units on its edges and numerical information transmitted from its directed edges after calculation in a sequence node to node [4].

In the workload of any database, it is reported that a phenomenon of interaction exists among the queries. It means that a query can be executed in isolation and the combination of more than one. As far as query interaction permits for query execution in conjunction, it also showed that their implementation might positively or negatively affect the execution of the individual query. If execution effect positively, one query utilizes the data in buffer pool directly without waiting, which is called by another query for execution and time for computation on that data is saved. But if the execution effect is negative, then one query interferes with another query execution, and both require different resources, which may cause locking.

Literature reports an important phenomenon named interaction among the database queries while running on the system. Therefore, it may enable the query to execute in isolation or in combination with other queries. The query interaction becomes the cause of positive or negative effects on those queries that are running simultaneously. The interaction of query may direct towards the interaction of query mixes. It may be possible that the interaction of query mixes provides interesting results.

2. RELATED WORK

The purpose of each scheduler is to automatically choose the preferences of queries in any database system [5]. If it is needed to measure the performance of the database system, then managing the workload of that specific database becomes essential. The database's workload develops with the combination of several queries randomly running

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

and several times with each other. Other parameters like availability of resources, resource allocation, and geographical storage location of data on disk.

The volume of data in databases has increased rapidly with each passing day. Therefore, databases perform poorly. The fact that human beings cannot handle such vast and diverse data creates the need to build database-based applications that address these issues [6] [7] [8] [9] [10]. If resources are already in use, the system must wait for those resources to avoid locking conditions. The system also must wait for the required resources to execute the query. In order to execute the desired operation, the CPU, RAM, cache, and I/O devices may become the subject of competition [11] [12] [13].

Query response time may also affect due to these reasons, which are query progress imaging, query arranging, and capacity management [14].

Database query response time plays a crucial role while managing massive database systems, especially in the workload, which may show the result of running the bulk of queries at the same time on the request of users because it may help the database administer (DBA) to the whole work efficiently because of coordination within the system and increase overall performance [15] [16]. A queuing theory determines how long it takes to execute queries based on the necessary resources [17].

A scale information system, just like a search engine, is mainly concerned with the amount of data which they can show the less amount of time (effectiveness). Still, they don't focus on the relevant results, which users want to get (efficiency). Users mostly analyze the speed of data (time for showing results of any search which any user uses) received, not the repetition (number of repetitions for acquiring desired results) in which data is received [18].

Data is retrieved from the database system by executing query execution plans which specify how data is accessed from the database system's source tables. The selection of any query execution plan among several other possible plans is query optimization [19] [20] [21].

Prediction of response time is crucial issue in several database management jobs like scheduling of query [22] [23] [24] visualization of query progress, balancing of load [25] [26] [27]. Formerly cardinality of query execution plans are not accurate [28] [29].

3. METHODOLOGY

This section explains the methodology of the Ascheduler. First of all chooses the queries from which the developing of the work loadis needed, select the a database for conducting experiment. After that different number of repetition is assigned to each query randomly for the development of the workload, then apply the algorithm of Ascheduler with modified MQI (Mutiple Query Interaction) for developing query mixes, the process of query mixes will remains going on and avoidence from the bad query mixes will be managed and improved gradually. The process of the methodology pictorially represented below.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

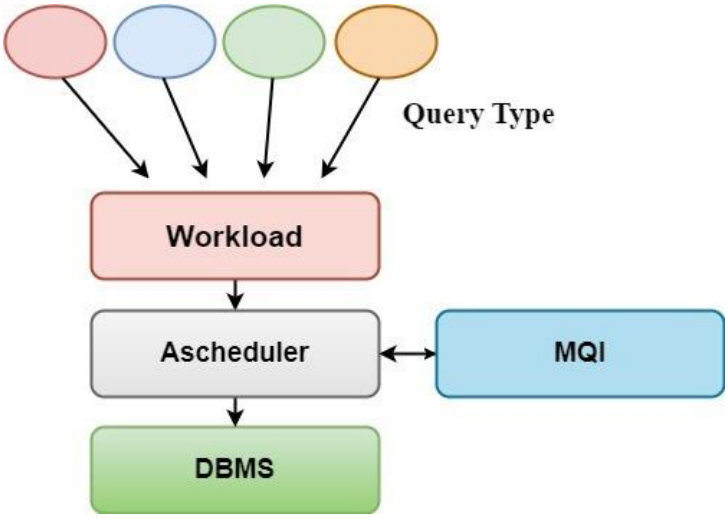

Figure 1: Working Process

3.1 Experimental Evaluation

The details of the hardware and software used in this experiment are given below. Processor: A Dual Intel Xeon Gold 6240 CPUs, each with 20 cores and 40 threads and clocked at 2.60 GHz is used. The primary memory is 128 GB of DDR4 RAM for operating at 2933 MHz. A 1TB NVMe SSD is used for data storage and retrieval. A 10 Gbps Ethernet connections is used for seamless data transfer. Modelling performance has a variety of scenarios and features used to predict database performance. The objectives of performance optimization are parameters tuning query scheduling, and configuration of the system. Seeing response time as a crucial point for focusing provides efficiency in database queries operations.

3.2 Database Workload

This research took scale factor 10 for execution of workload. It used 10 queries of TPC-H benchmark for developing workload, for developing query mixes and for determination of Ascheduler's efficiency. There 125 queries are used as workload by randomly assigning different repetition to each selected query. The DBMS is POSTGRESQL. The POSTGRESQL configuration advisor is manipulated; the parameters of configuration are well tuned.

3.3 Ascheduler

In the dynamic landscape of database management and query optimization, the concept of an "Ascheduler" emerges as a potential game-changer. This scheduler holds the promise of harnessing the power of MQI (Multiple Queue interface) to revolutionize the way queries are sequenced and executed within a complex database infrastructure. In the world of database management and query optimization, scheduler is introduced as a potential game-changing concept. It aims to use MQI to improve how queries are ordered and executed in a complex database system.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

3.3.1. Role of a Scheduler

A scheduler's primary role is to decide the order and priority of tasks (in this case of queries) for processing. "Ascheduler" aims to be fair and optimize resource allocation while ensuring that query execution remains balanced. This is different from traditional scheduling methods that might prioritize tasks based on factors like size or urgency.

3.3.2. Role of MQI

Multiple Queue Interface (MQI), is modified as a dynamic element in this scheduling approach. We modify it by allowing for the management of multiple queues, each with its own attributes and priorities. Queries can be categorized based on factors like their complexity, importance, or resource requirements. The scheduler can then adaptively adjust the sequence in which queries are selected based on this information. This adaptability enables the system to respond in real-time to changing workloads and priorities.

3.3.3. Ascheduler's Workflow Process

The queries can be selected and load to MPL once it would complete the 3 process the again start reverse way.

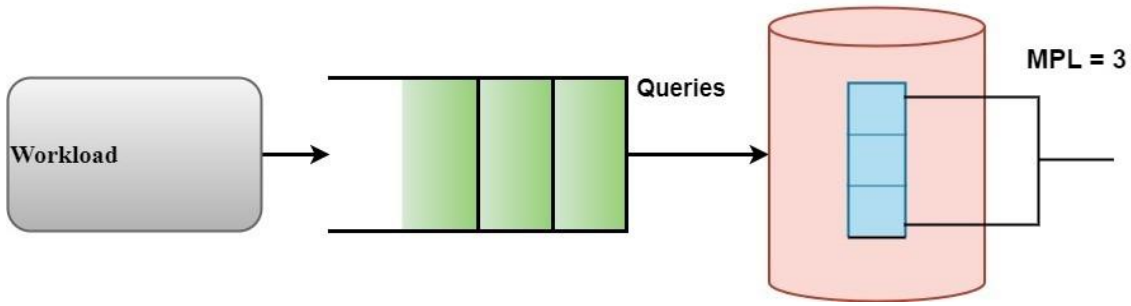

Figure 2: Simplified view of Ascheduler's Mechanism

3.4. Algorithm Process

The process implements a scheduling algorithm, represented by the 'Ascheduler' function. This function takes three input parameters: 'W', a list of query-weight pairs; 'M', which signifies the multiprogramming level; and 'MQI', the threshold for Multiple Query Interaction. The algorithm aims to intelligently scheduler queries to maintain a desired multiprogramming level while adhering to the MQI constraint:

The algorithm begins by initializing an empty list called 'RunningMix', which is used to keep the track of the queries currently running in the query mix. It then evaluates wheatear the number of queries in 'W' is less than multiprogramming level 'M'. If this condition holds, it concludes that no scheduling is necessary, and the function returns 'None'.

In case where the number of queries exceeds or matches the multiprogramming level, the algorithm proceeds by sorting the list 'W' based on the weight (wi) of each query in descending order. This sorting prioritizes heavies' queries.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

The algorithm iterates through the sorted queries in 'W'. For each query, it checks weather the query can be schedules without violating the MQl constrains using the 'can_scheduler_query' helper function. If the query can be scheduled, it returns the name of the query, signifying that this query should be executed next.

If no queries can be scheduled based on the current query mix and the MQI constraint, the algorithm calls the 'identifyGoodMixes' function to in deify potential good query mixes. These query mixes are collection of queries that can be executed while satisfying the MQI constraint. Subsequently, it iterates through the sorted queries again, attempting to find a query that can be schedules within the indentified good mixes.

The 'can_scheduler_query' function checks whether a query can be inserted into a query mix without violating the MQI constraint. It verifies that the query is not already present in the query mix and that adding the query would maintain an acceptable MQI level.

Algorithm 1

```
def Ascheduler(W, M, MQI):
    RunningMix = []  # Initialize an empty list to represent the running mix

    if len(W) < M:
        return None  # No scheduling needed if less than M queries in W
    else:
        # Sort the queries in descending order based on their weights (wi)
        W.sort(key=lambda q: q[1], reverse=True)

        # Iterate through the sorted queries
        for q in W:
            # Check if the query q can be scheduled without violating MQI constraint
            if can_schedule_query(q, RunningMix, MQI):
                return q[0]  # Return the required query

        # If no query can be scheduled in RunningMix with MQI >= 7, find good mixes
        good_mixes = identifyGoodMixes(M, W)
        for q in W:
```


Figure 3: Algorithm of Ascheduler

The 'calculate_MQI' function, which calculates the Multiple Query Interaction, is currently a placeholder. It needs to be tailored to our specific context and should compute the MQI based on a list of queries.

Similarly, the 'identifyGoodMixes' function is a placeholder indented to identify potential query mixes of queries that can be executed tighter while adhering to the MQI constraint.

Finally, the code includes sample data for queries, the multiprogramming level ('M'), and the MQI threshold ('MQI'). The 'Ascheduler' algorithm is invoked using this queries workload to determine the query that should be scheduled next. If a query needs to be

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

schedules, its name is printed. If no query requires scheduling, a message indicating so printed.

```
# Helper function to calculate Multiple Query Interaction (MQI)
def calculate_MQI(queries):
    # Implement MQI calculation logic here
    total_weight = sum(q[1] for q in queries)  # Example: Sum of query weights
    return total_weight

# Placeholder for the identifyGoodMixes function
def identifyGoodMixes(M, W):
    # Implement identifyGoodMixes logic here
    good_mixes = []  # Example: List of good mixes
    return good_mixes


W = [("<q1>", 5), ("<q2>", 8), ("<q3>", 4), ("<q4>", 6)]
M = 3
MQI = 7

# Call the Ascheduler algorithm
required_query = Ascheduler(W, M, MQI)
if required_query:
    print("Required query:", required_query)
else:
    print("No query needs to be scheduled.")
```

Figure 4: MQI for Good query Identification

4. RESULTS

This section describes the effectiveness of the Ascheduler. We present the comparison of other scheduler like First Come First Serve (FCFS) and Shortest Job First (SJF) with the Ascheduler. The Ascheduler is implemented on the POSTGRESQL and TPC-H benchmark standard queries are used for analyzing. The following section is divided into two parts one is experimental setup and other is Effectiveness of Ascheduler.

The evolution of the “Ascheduler” scheduling algorithms involves a comprehensive assess meant of its performance within a computer system’s task management context. The primary focus is on gauging its efficiency and effectiveness. This evaluation entails establishing specific criteria for assessment, which often include response time, turnaround time, waiting time, fairness, and resource utilization. By employing key performance metrics such as average turnaround time, average waiting time, CPU utilization, and throughput, the algorithms’ operational prowess can be accurately measured. To provide a robust evaluation, comparison with established algorithms like First-Come First-Served (FCFS), Shortest Job First (SJF) and Round Robin are essential across different workloads. Simulated scenarios or real-world implementation are utilized to observes the algorithm’s behavior under diverse condition, aiding in the collection of relevant performance data.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

Additionally, factors like sensitivity to parameters changes, resource allocation efficiency, adaptability to dynamic workloads, and scalability are scrutinized to ensure a comprehensive evaluation of “Ascheduler’s” capabilities. The overarching goal is to ascertain how effectively “Ascheduler” manages processes, allocates resource, and adapts to varying scenarios ultimately defining its values within practical computing environments. Changes, resource allocation efficiency, adaptability to dynamic workloads, and scalability are scrutinized.

4.1. Categorization of Query Mixes

In our system, query mixes are meticulously categories primarily based on their instances, a pivotal challenge dealt with by way of the scheduler to streamline efficient execution. This categorization system is designer evaluate query mix effectiveness in phrases of execution efficiency, main to the advent of 4 well-described classification of query mixes.

4.1.1. Best Query Mixes (30.0%)

Occupying the priority, these query mix 30.0% of our allocated resource. They have earned this coveted repute by way of continually demonstrating super performance characterized by using fast response time. Best Query Mixes represent the gold popular, putting the bar for excellence in execution performance notably:

4.1.2. Good Query Mixes (forty 40%)

Directly underneath the “Best Query Mixes” in terms of response time, forty 40% of our assists are channeled into good query mixes. While now not achieving the zenith of the “Best” class, they boast commendable response times that make them necessary in our good query mixes. A couple of example from this category includes:

4.1.3. Moderate Query Mixes (20.0%)

Comprising 20.0% of our allotted assets, the “Moderate Query Mixes” exhibits moderate response time of execution. They play a pivotal function in handling ordinary duties at a widespread priority degree. An instance from this category:

4.1.4. Bad Query Mixes (10.0%)

Lastly, we carefully allocate 10.05 of our resources to the “Bad Query Mixes” These query mixes contains such query mixes that showed poor performance by representing high response time. To protect against performance degradation, they are recognized and punctiliously avoided inside the execution queue:

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

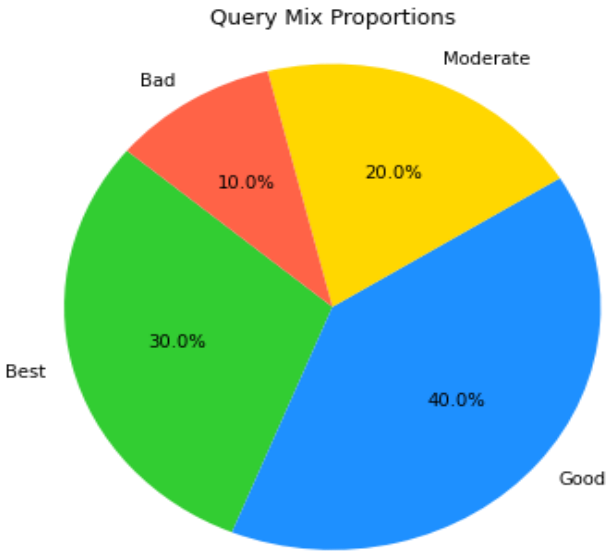

Figure 5: Query Mix with Proportions

4.2. Avoidance of Bad Query Mixes

To mitigate database systems inefficiencies, the scheduler is made to actively recognize and steer clear of “bad” query mixes. Poor query mixes frequently include queries that impede the workload’s overall execution. The scheduler’s avoidance strategy could consist of:

Dynamic Search prioritization: Giving queries in a poor query mix a lower priority or fewer resources.

Isolation: Running problematic queries separately from other queries to reduce undesirable consequence.

Resource Reallocation: Giving a terrible query mix of queries more resources to use in order to speed up their response time. Limiting the execution of such queries from a problematic query mix to avoid system’s overload.

Table 1: Query Mix with Time

Query Mix Category	Proportion (%)	Average Response Time
Best	30	2.5
Good	40	4,0
Moderate	20	6.8
Bad	10	10.2

The scheduler support a timely and effective workload management system by actively avoiding undesirable query mixes and ensuring the availability of best, decent, and moderate query mixes. By reducing delays and bottlenecks brought on by ineffective query mixes, this method optimizes resource allocation and improves user’s experience.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

4.2.1. Query Interaction:

Query interaction exists while queries are executing in parallel. This phenomenon occurs due to resource sharing among the queries executing simultaneously. The interaction of queries represents influences due to existence of bonding in the parallelism. Query interaction primarily affects on the response time of queries. Query interaction is dedicated to elucidating intricacies of the methodology used for calculating query interaction scores, offering a profound understanding enriched with meticulous description. The importance of quantifying interaction between queries within query mixes is underscored, as these interactions play a pivotal role in appropriately categorizing into bad, good and moderate query mixes. This section outlines the quantification process, which meticulously evaluates how individual queries interact with one another this analysis encompasses aspects such as concurrency assessment, resource dependency analysis and data depends evaluation. The outcome of this quantification process is the assignment of query interaction scores, which quantatively represent the degrees of interaction of queries within a query mix. For instance, query mix1 is assigned an interaction score of 0.75, indicating a robust level of interaction among its queries. Query mix2 receives a score of 0.5, denoting a moderate degree of interaction, while query mix3's score of 0.3 suggests a relatively lower level of interaction among its queries. These interaction scores are invaluable in guiding the scheduler's decisions regarding resource allocation, execution order, and overall workload management strategies. By providing a thorough understanding of the methodology behind calculating query interaction scores and offering specific score examples equips readers with the knowledge needed to navigate the complexities of query mix’s categorization and optimize database system’s performance.

4.2.2. Query Interaction Quantification

The scheduler has tools for methodically evaluating how each query mix’s interaction with other inquires. This entails assessing how queries interact in terms of execution duration, resource usage, and performance. During query execution, the scheduler gathers pertinent data and examines patterns of cooperation or conflict.

4.2.3. Impact on Categorization

A crucial input for the classification of query mixes is the quantification of query interaction. The scheduler bases its evaluation of query mix’s overall effectiveness and prospective placement within the established categories on the results from interaction assessments.

4.2.4. Synergy and Contention Identification

The scheduler determines instances of synergy, where particular queries improve one another's performance, and contention, where queries obstruct one another's execution, by interaction quantification. Due to their synergistic effects, synergistic query mixes may contribute to the "best" or "good" query mix categories. On the other hand, a query mix's placement in the "moderate" or "bad" categories may be influenced by query pairs that

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

are producing de-acceleration in the execution and become cause of increase in response time.

4.2.5. Enhanced Categorization Accuracy

The scheduler classifies query mixes with a higher degree of accuracy by taking query interactions into account. With this method, judgments are made more thoroughly and intelligently since it considers not just the performance of each individual query but also the performance of that query in the query mix in conjunction of other queries.

4.2.6. Dynamic Adjustment of Categories

The scheduler can dynamically change query mix categorizations thanks to the quantified interactions. The scheduler may hone its classification over time to more closely represent the observed performance patterns if a query mix's interactions consistently display synergy or contention. In conclusion, a key step in the categorization of query mixes is the measurement of query interactions. The scheduler can effectively classify query mixes into the "best," "good," "moderate “or” bad" categories by having a thorough grasp of how queries interact with one another in any query mix. This improves the workload management system's effectiveness and efficiency.

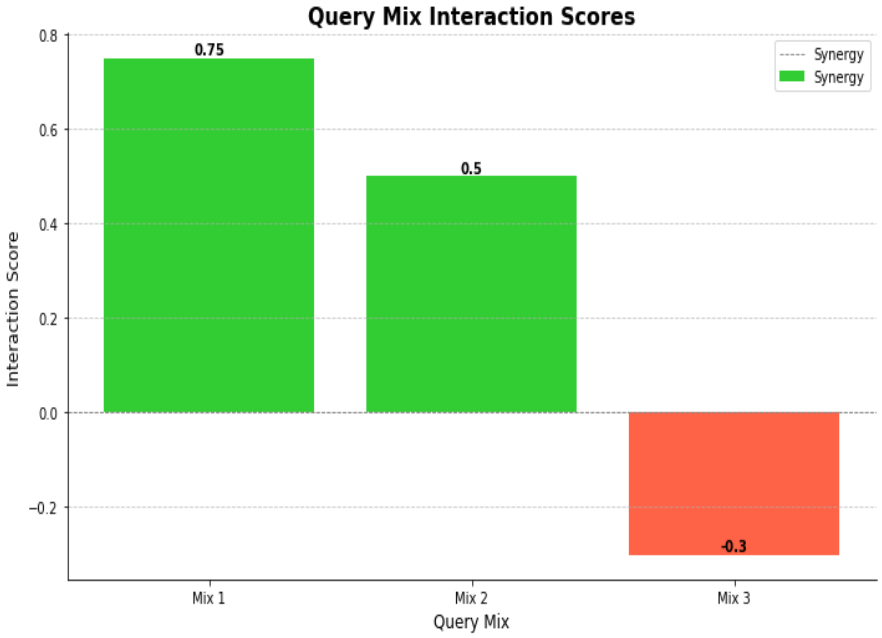

Figure 6: Query Interaction Scores

The process of quantifying query interaction and categorizing query mixes mathematically is systematic approach that enables the scheduler to informed decision about workload management this process involves server interconnected stage that contributes the optimization of resource allocation and performance categorization. At the core of this process is data collection. During the execution of query mixes, the scheduler gathers crucial information such as the execution times and resource usage of individual queries.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

This raw data forms the basis for understanding how queries behave in isolation. Next, the scheduler engages in correlation analysis. By evaluating the correlation between the execution time and resource usage of different queries within a query mix, the scheduler gains insights into the interdependent of these variables This correction serves as a foundational elements for assessing the digress of interaction between queries within a query mix. Building upon correlations, the scheduler proceeds to calculate interaction scores. These scores encapsulate the essence of query interactions by considering the correlation coefficient, the individual execution times of queries, and their resource usage. The interaction score quantifies the extent to which queries collaborate or compete within a query mix with other concurrently running queries. Leveraging the calculated interaction scores, the scheduler then categorizes query mixes into different performance tires.

Positive and high interaction scores indicate synergistic behavior among queries, suggesting that these query mixes could be categorized as "best" or "good." Conversely, negative interaction scores point towards contention, possibly resulting in categorization as "bad" or "moderate" query mixes. The process is iterative and dynamic. As the scheduler encounters and processes more query mixes, it continuously refines its calculations and formulas based on real-world data. This dynamic learning and adaptation ensure that the scheduler's decision-making evolves to accurately capture the nuances of query interactions. Importantly, interaction scores influence resource allocation strategies. Positive interaction scores guide the scheduler to allocate more resources to mixes with collaborative queries, harnessing their combined efficiency. Meanwhile, negative interaction scores trigger resource adjustments to mitigate performance issues stemming from contentions within query mixes.

The scheduler's performance evaluation is an integral part of this process. By comparing the predicted categorizations with the actual performance outcomes, the scheduler identifies any discrepancies and inconsistencies. This feedback loop drives further enhancement of the mathematical models and methodologies used for quantifying query interactions and categorizing mixes. In conclusion, the mathematical approach to quantifying query interactions and categorizing query mixes is a data-driven and iterative process that underpins the scheduler's ability to optimize workload management. By considering correlations, calculating interaction scores, and adapting dynamically, the scheduler ensures optimal resource allocation and performance categorization based on the collaborative or competitive nature of queries within query mixes.

4.3. Effectiveness of Ascheduler

The Ascheduler is more efficient than FCFS and SJF. With the passage of time the FCFS become poor because it can’t avoide from the bad query mixes and SJF depicted worst scanrio overall. The figures represents the overall performance of Ascheduler, FCFS and SJF.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

4.3.1. Ascheduler Vs Other Scheduling Algorithms

We used several scheduling algorithms in our experiment like Ascheduler, First Come First Serve (FCFS) and Shortest Job First (SJF). The primary objective of this research is to test the AScheduler. Quantification of query interaction in query mixes by using labels provides very good results. 125 queries are selected for the workload.

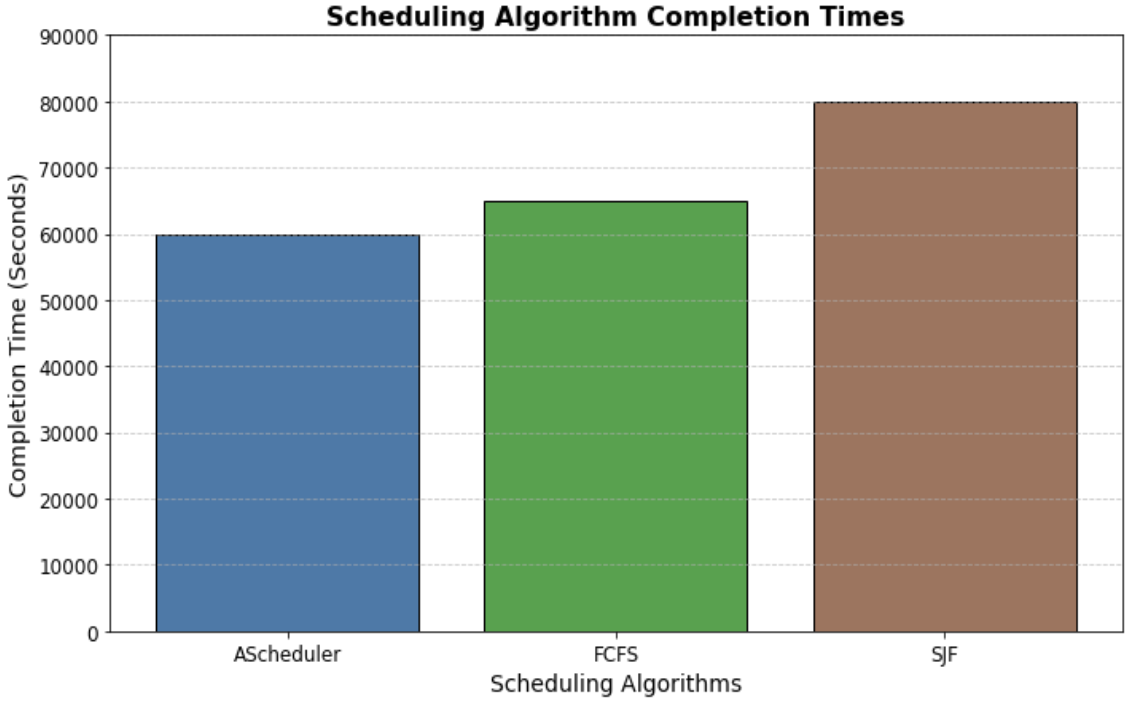

Figure 7: Workload Completion Time of Three Scheduler

Table 2: Efficiency of Ascheduler algorithm with FCFS and SJF

Completion Time	Ascheduler	FCFS	SJF	Ascheduler Saved Time (vs.FCFS)	Ascheduler Saved Time (vs. SJF)
Seconds	50580	54780	56520	4200	5940
Minutes	843	913	942	70	99
Hours	14.05	15.22	15.7	1.17	1.65

4.3.2. Detailed Results of the Experiments

Efficient of A scheduler presents a comparison of completion times for different scheduling algorithms used in computer systems, alongside a mysterious "Ascheduler" algorithm. The completion time represents the total time taken to execute a set of processes or jobs. In this context, "FCFS" (First-Come, First-Served) and "SJF" (Shortest Job First) are two well-known scheduling algorithms in operating systems. The table showcases how the "Ascheduler" algorithm performed in comparison to these established methods. The "Saved Time" columns indicate how much time "Ascheduler" saved compared to both FCFS and SJF.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

Positive values in the "Saved Time" columns suggest that "Ascheduler" performed better in terms of reducing execution time.

- "Ascheduler" completed its tasks in 50,580 seconds, while FCFS required 54,780 seconds, and SJF took a longer 56,520 seconds.
- When evaluating "Ascheduler" against FCFS, it emerged as the clear winner, saving an impressive 4200 seconds (or approximately 70 minutes) in execution time.
- Compared to SJF, "Ascheduler" exhibited even more substantial efficiency, reducing the execution time by a remarkable 5940 seconds (or approximately 99 minutes).

Now, looking at these time measurements from a broader perspective:

- "Ascheduler" took 843 minutes, equivalent to approximately 14.05 hours, to complete its tasks.
- In contrast, FCFS required 913 minutes (around 115.22 hours), and SJF demanded 942 minutes (approximately 15.7 hours).
- The efficiency of "Ascheduler" becomes evident when we note that it saved 70 minutes compared to FCFS and an impressive 99 minutes in comparison to SJF.
- In terms of hours, "Ascheduler" outperformed FCFS by 1.17 hours and surpassed SJF by a substantial 1.65 hours.

Scheduler: This column lists the names of the scheduling algorithms that were evaluated in that experiment. The scheduler include “Ascheduler,” “FCFS” (First Come First Serve), and “SJF” (Shortest Job First).

Response Time (ms): This is column displays the average response time in milliseconds (ms) for each scheduler. The response time is the time taken for query mix to complete execution. For instance, the “Ascheduler” had an average response time of 25 ms, “FCFS” had 40 ms, and “SJF” had 38 ms,

Categorization Accuracy (%): This column presents the percentage of accuracy achieved by each scheduler in categorizing query mixes. It indicates how accurately the scheduler categorized the query mixes into the predefined categories(“best” “good” “moderate, “ or “bad”) The “Ascheduler ” achieved an accuracy of 90 %, “FCFS” had 80% and “SJF” had 70%.

Interaction Score Improvements (%): This column represents the improvement in interaction score accuracy achieved by integrating interaction scores into the categorization process. Interaction scores quantify the level of interaction between queries within a mix The “Proposed Scheduler” showed an improvement of 15%, “FCFS” had no improvement (0%), and “SJF” had an improvement of 10%

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

These results provide insights into performance and accuracy of different scheduling algorithms based on response time, categoirzation accuracy, and the impact of considering interaction scores. The values in the table allow you to compare the performance of the scheduler and draw conclusion about here effectiveness in improving response time and accuracy in categorizing query mixes.

Table 3: Effectiveness of Ascheduler algorithm with FCFS and SJF

Scheduler	Response Time (ms)	Categorization Accuracy (%)	Interaction Improvement Sore (%)
SJF	35	70	10
FCFS	30	80	0
Ascheduler	25	90	15

A comparative assessment of three scheduling algorithms: SJF (Shortest Job First), FCFS (First-Come, First-Served), and "Ascheduler." It evaluates these algorithms across three key performance metrics: Response Time (measured in milliseconds), Categorization Accuracy (expressed in percentages), and Interaction Score Improvement (also in percentage terms). Notably, "Ascheduler" exhibits the shortest response time at 25 ms, indicating its efficiency in promptly addressing tasks.

Ascheduler also leads in categorization accuracy with 90%, showcasing its precision in prioritizing tasks. In terms of enhancing the overall user interaction experience, Ascheduler also excels with a 15% improvement.

These metrics provide insights for selecting the most suitable scheduling algorithm, with "Ascheduler" excelling in response time, categorization accuracy, and leading in interaction score improvement. This shows that Ascheduler perform well in improvement of response time of query mixes by using priority ranking on the upcoming queries for the development of query mixes. It also gives the strategy for avoiding the bad query mixes and attain the good, best and moderate query mixes.

5. CONCLUSION

This paper shows the overall efficiency of the newly proposed query scheduler named ‘Ascheduler’. This scheduler also categorized the query mixes into best, good, moderate and bad query mixes. It kept giving the avoidance from the developing the bad query mixes with the pessage of time by using previously assigning the priority to the queries. It also executed the workload in fewer time from the already exist scheduler like FCFS and SJF. It showed good results from other schedulers in other performance metices like response time, categorization accuracy and interaction improvement score.

Conflict of Interest

Authors declare no confflict of interest.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

References

1) Mateen, B. Raza, M. Sher, M. M. Awais and N. Mustapha, "Workload management: a technology perspective with respect to self-* characteristics," *Artificial Intelligence Review,* vol. 41, pp. 463-489, 2014.

2) Raza, A. Mateen, A. M. M. and M. Sher, "Survey on autonomic workload management: algorithms, techniques and models," *J. Compute,* vol. 3, no. 7, pp. 29-38, 2011.

3) F. Lan, J. Zhang and B. Niu, "Predicting Response Time of Concurrent Queries with Similarity Models," *Big Data Research,* vol. 25, pp. 1-13, 2021.

4) J. Zhang and B. Niu, "A clustering-based sampling method for building query response time models," *Computer Systems Science & Engineering,* vol. 32, no. 4, pp. 319-331, 2017.

5) M. Amjad and J. Zhang, "Gscheduler: A Query Scheduler Based on Query Interactions," in *In Web Information Systems and Applications (WISA): 15th International Conference, WISA 2018, Taiyuan, China, September 14–15, 2018*, 2018.

6) S. Guirguis, M. A. Sharaf, P. K. Chrysanthis, A. Labrindis and K. Pruhs, "Adaptive Scheduling of Web Transactions," in *In 2009 IEEE 25th International Conference on Data Engineering*, 2009.

A. C. Konig, B. Ding, S. Chaudhuri and V. Narasayya, "A Statistical Approach towards Robust Progress Estimation," in *arXiv preprint arXiv: 1201.0234*, 2011.

7) J. Duggan, U. Cetintemel, O. Papaemmanouil and E. Upfal, "Performance prediction for concurrent database workloads," in *In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011.

8) M. B. Sheikh, U. F. Minhas, O. Z. Khan, A. P. P. Aboulnaga and D. J. Taylor, "A bayesian approach to online performance modeling for database appliances using gaussian models," in *In Proceedings of the 8th ACM international conference on Autonomic computing*, 2011.

9) M. Ahmad, D. Songyun, A. Aboulnaga and S. Babu, "Predictiong completion times of batch query workloads using interaction aware models and simulation," in *In Proceedings of the 14th International conference on Extending Database Tchnology*, 2011.

10) M. Ahmad, A. Aboulnaga, S. Babu and K. Munagala, "Modeling and exploiting query interactions in database systems," in *In Proceedings of the 17th ACM conference on Information and knowledge management*, 2008.

11) M. A. Q. Bilal, B. Niu, M.-u.-. Rehman, N. Ahmed, A. Hussain, B. Ahmed, M. Amjad and S. Kanwal, "Creation and Comparison of Query Mix," *International Journal of Computer Applications,* vol. 177, pp. 33-36, 2019.

12) A. Ganapathi, H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan and D. Patterson, "Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning," in *In 2009 IEEE 25th International Conference on Data Engineering*, 2009.

13) M. Akdere, U. Cetintemel, M. Riondato, E. Upfal and S. B. Zdonik, "Learning-based Query Performance Modeling and Prediction," in *In 2012 IEEE 28th International Conference on Data Engineering*, 2012.

14) S. Tozer, A. Aboulnaga and T. Brecht, "Q-Cop: Avoiding bad query mixes to minimize client timeouts under heavy loads," in *In 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, 2010.

15) J. Duggan, O. Papaemmanouil, U. Cetintemel and E. U. Upfal, "Contender: A Resource Modeling Approach for," *EDBT,* pp. 109-120, 2014.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 11:2023
DOI: 10.5281/zenodo.10223943

16) Macdonald, N. Tonellotto and I. Ounis, "Learning to predict response times for online query scheduling," in *In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieva*, 2012.

17) S. M. Mahajan and V. P. Jadhav, "An analysis of execution plans in query optimization," in *In 2012 International Conference on Communication, Information & Computing Technology (ICCICT)*, 2012.

18) M. Joshi and P. R. Srivastava, "Query Optimization: An Intelligent Hybrid Approach using Cuckoo and Tabu Search," *International Journal of Intelligent Information Technologies (IJIIT),* vol. 9, no. 1, pp. 40-55, 2013.

19) S. Chaudhuri, V. Narasayya and R. Ramamurthy, "Estimating progress of execution for SQL queries," in *In Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004.

20) G. Luo, J. F. Naughton, C. J. Ellmann and M. Watz, "Toward a progress indicator for database queries," in *In Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004.

21) J. Li, R. V. Nehme and J. Naughton, "GSLPI: A Cost-Based Query Progress Indicator," in *In 2012 IEEE 28th International Conference on Data Engineering*, 2012.

22) T. J. Wasserman, P. S. D. B. Martin and H. Rizvi, "Developing a characterization of business intelligence workloads for sizing new database systems," in *In Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP*, 2004.

23) J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner and A. Zeir, "Predicting in-memory database performance for automating cluster management tasks," in *In 2011 IEEE 27th International Conference on Data Engineering*, 2011.

24) G. Luo, J. F. Naughton and P. S. Yu, "Multi-query SQL Progress Indicators," in *In Advances in Database Technology-EDBT 2006: 10 International Conference on Extending Database Technologyy*, 2006.

25) W. Wu, X. Wu, H. Hacigumus and J. F. Naughton, "Uncertainty Aware Query Execution Time Prediction," *arXiv preprint arXiv:1408.6589,* pp. 1857-1868, 2014.

26) W. Wu, Y. Chi, H. Hacigumus and J. F. Naughton, "Towards predicting query execution time for concurrent and dynamic database workloads," in *Proceedings of the VLDB Endowment*, 2013.

27) R. Marcus and O. Papaemmanouil, "WiSeDB: A Learning-based Workload Management Advisor for Cloud Databases," *arXiv preprint arXiv: 1601.08221,* vol. 9, pp. 780-791, 2016.

28) B. P. Kang and M. Zaharia, "BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics," in *arXiv preprint arXiv: 1805.01046*, 2018.

29) Al-Masri and Q. H. Mahmoud, "Discovering the best web service: A neural network-based solution," in *In 2009 IEEE International Conference on Systems, Man and Cybernetics*, 2009.

30) D. Ram, L. Miculicich and B. Merve, "Neural Network Based End-to-End Query by Example Spoken Term Detection," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

31) A. Aslam and E. Curry, "Towards a Generalized Approach for Deep Neural Network Based Event Processing for the Internet of Multimedia Things," *IEEE Access,* vol. 6, pp. 25573-25587, 2018.