

DB Workload Management through Characterization and Idleness Detection

Abdul Mateen

Department of Computer Science,
Federal Urdu University of Arts,
Science & Technology, Islamabad
45570, Pakistan;
abdulmateen@fuuastisb.edu.pk

Khawaja Tahir Mahmood

Department of Computer Science,
Federal Urdu University of Arts,
Science & Technology, Islamabad
45570, Pakistan;
khawajatahirmahmood@gmail.com

Seung Yeob Nam*

Department of Information and
Communication Engineering
Yeungnam University
Gyeongsan, South Korea;
synam@ynu.ac.kr

Abstract—It is difficult to handle the database (DB) workload due to the huge increase in data, the functionality demand from the user, and the rapid changes in data. It is not easy to manage the DB workload, which therefore leads to malnourishment. To get efficient results, there must be complete knowledge about the type and changes in workload. The versatility and complexity of DBMSs led the DB researchers towards new philosophy and thoughts. A novel approach is introduced for DB workload management through characterization, scheduling, and database idleness detection. In workload characterization, workload is observed, and effective workload characterization parameters are selected. After that, scheduling is performed in order to arrange the DB workload to reduce the waiting time for each workload. Lastly, database idleness is identified at run-time and exploited for system as well as user-initiated workloads to improve efficiency. The proposed approach for workload management is validated through experiments using benchmark workloads.

Keywords—Workload, Autonomic, Characterization, Idleness Detection, Scheduling

I. INTRODUCTION AND BACKGROUND

The traditional DBMS are managing the data in from decays, and due to this reason, it is one of the core components of any company or business. The workload in DBMS has been grown due to large amount of data that cannot be handled by the human. The worth of DBMS was more than 15 billion dollars in 2005, with 10% annual growth that is also one of the major causes to increase of ownership [1]. Many techniques and models are presented in order to manage the workload efficiently. The main challenge in DBMS is to manage the workload efficiently which is complex, tricky, and comprised of complex queries. These workloads use various resources and process huge data. Usually, important workloads require an immediate response as compared to routine workloads. On the other hand, DBMS also have to manage the system-oriented tasks that run regularly (e.g., index, statistics management etc.).

The proposed research is being conducted to manage the complex and tricky workloads efficiently. The number of tasks assigned to a worker in a given amount of time is referred to as their workload. In computer science, it is defined as the processing time required to complete a task, which may include some programming [2]. The amount of work that a database management system (DBMS) produces or can produce in a given time interval is referred to as the database workload. A more concrete definition of the DB workload could be the set of structured query language statements, which could be OLTP or DSS, sessions, commands, and other regular operations. The OLTP workload is composed of tinny transactions that may be insert, delete, and so on, on the other hand the DSS workload requires long time for completion, many resources and involves much computations. Prior to the 1980s, capacity

planning was introduced with the goal of cost sharing and as a method of forecasting future organizational demand. It is used to forecast the organization's future needs. Capacity refers to the maximum amount of work that an organization can complete in a given period of time. Workload management capacity planning always ensures that functions meet their production targets while staying within budget. In terms of workload change, capacity planning's primary concern is with resources, their management, configuration, and impact. Characterization and performance are used to plan capacity [2-3]. Later, the capacity planning research was converted to Resource-Oriented Workload, which maintained the service level concept by allocating resources to applications while maximizing resource utilization.

The goal of a performance-oriented task is to maximize resource efficiency while minimizing laborious effort. It executes the adaptation by configuring and optimizing without any human input [4], and is sometimes referred to as autonomic workload management. Finally, the evolution of the workload has moved from capacity planning to resource orientation to performance orientation. Technology-enabled cost sharing and Service Level Objectives (SLOs) are the cornerstones of the evolution of workload. The SLO, which details the agreement's objective, is the most important part of a service level agreement (SLA) between a service provider and a client. By comparing the proposed solution to these well-known methods, the effectiveness of the solution is evaluated. This approach is used for two reasons: first, none of the DBMS providers (Oracle, SQL Server, DB2) disclose information about internal workings of their products, and second, it is challenging to change the internals of the DBMS because the source code is not readily available. This approach was also used by earlier researchers, such as Mumtaz et al [17], to demonstrate the efficacy of their suggested strategy. Another well-known and effective method for comparison is time sharing, in which DB workloads are run for specified intervals of time and may be completed, or they may be briefly halted and resumed when it is their turn. This article is organized as following: Section II discusses the related work. Section III describes the steps to characterize, schedule and identification of idleness detection in the DB workload. In section IV, experimental results are illustrated, and finally section V concludes the article with future directions.

II. STATE OF THE ART

Many researchers are working to increase the self-management, dependability, and efficiency of DBMSs. Here, the literature review is divided into two areas, namely, the sections on characterization and scheduling strategies. A number of these are covered in the section that follows and are pertinent to the solution for workload characterization and scheduling in DBMSs and DWs.

Two utility functions (dynamic resource allocation and query scheduling) are introduced [13] to optimize the autonomic systems according to the specific goals. It is discussed how these two utility functions can be incorporated in DBMSs. These functions convert the BI importance policies into low-level policies through which performance objectives of the organization are achieved. It also advises better techniques for some workload executing on the DBMS under particular circumstances. DBMS congestion indicators were identified and discussed by Zhang et. al. that consists of DB monitor metrics. The system should have the ability to detect the congestion and then work according to the situation. The approach is then validated by performing various experiments and these indicators can vary according to the congestion. However, the research did not identify or predict system performance problems in detail. Moreover, the experiment is performed only taking the specific scenario. A model is proposed [5] to identify and characterize the workload; and effective workload parameters are identified. [6, 7] proposed the classifier that works on the basis of workload characteristics and manages the workload characteristics through decision tree induction. However, there is no discussion about the distinct workload parameters which are most effective to distinguish the workload in the proposed classification. Research in [8] examines the SQL statements and views with respect to their composition; and classifies the data. The efficiency is increased by finding out the criteria for DB workload. However, there is no discussion about the generation of instructions for the physical design automatically. An analysis of characterization technique for Business Intelligence (BI) workload is discussed through clustering technique [9] where the Singular Value Decomposition (SVD) and SemiDiscrete Decomposition (SDD) clustering algorithms are applied. The proposed approach uses resource related parameters that include number of joins, CPU usage and Input/ Output rate. For each workload class data is validated and resource demand is identified and finally a report is generated that reveals a priority wise list of hardware configuration. The characterization of workload is done by performing five steps which are components identification, selection of characterizing parameters, input data normalization, workload grouping into classes and finally identification of each class. The proposed characterization technique is performed on TPC-H benchmark like data but with some assumption based parameter. The technique is limited as for characterization it only considers one parameter i.e. user resource demand. A technique is proposed [10] where the State Transition graph was introduced for characterizing the workload. It shows the same pattern for those customers who are doing similar type of activities. After finding this information, clustering is used to characterize the workload. However, the proposed technique had some problems such as maximum session drops when there is a large number of customers and no mechanism is described to extract these session drops. An approach [11] is introduced that is used to observe the DB workload by using the n-Gram model. It consists of an API that determines workload shifts with two major steps; first observing the workload and comparing with the workload model and configuration analysis is done with the shift. Experiments are performed on the DVD shop dataset where different workload shift scenarios are taken. The DB workload in E-commerce workload is characterized by using the black box attributes [12] where parameters such as disk

usage, response time and CPU time are used. It is observed through experiments that dynamic cache has the ability to reduce response time. It was also highlighted in the research that efficiency of the Hybrid cache is better than Table and Query cache. The experiments are performed on the TPC-W benchmark to validate the black box approach. A framework [13] for autonomic workload management is proposed which monitors and controls the flow of the DB workloads dynamically without any human involvement. The framework mainly comprises of multiple workload management techniques and performance monitor functions. Whenever the performance degrades, the proposed approach finds out that issue and handles it intelligently by the help of indicators. The experiments and results are verified by using a prototype implemented in DB2. One issue in this research is that the workload is assumed to be constant which is heterogeneous in real cases. The other problem with the research is that prototype is built instead of complete system. A framework [14] is proposed to handle workload by stopping and restarting queries. When a query is resumed it always restarts from the stopped position. The proposed framework has the ability to manage single Query Execution Plan (QEP) and fails to restart query when records are updated. A query suspension and resumption [15] technique is introduced where asynchronous checkpoints are used. In case of query suspension all the resources are released while in case of resumption the required resources are resumed. The technique is also implemented in a tool named as PREDATOR and proved that this tool has better results than others. It allows to suspend the query as compared to previous techniques where switching is performed between individual operators. However the proposed technique does not re-optimize the given query and memory wastage is also less. A technique to manage the OLTP workload is discussed [16] where MPL is set by using the parameters (CPU and disk) and used feedback controller. [17] discussed a framework Query Shuffler (Qshuffler) where DB workload is managed by finding out dependency of query. This proposed research works without knowing database and causes of query dependency. It is implemented in DB2 and experiments are performed over the TPC-H benchmark workload. As per author's claim, it is four times better than the FIFO scheduling. As larger jobs have to wait for long time therefore average execution time is larger. The Mixed Workload Scheduler (MWS) is introduced [18] where a non-preemptive scheduling was used in order to manage the BI workload. It takes assumption-based parameters and considers queries at the same level. Moreover, the work does not consider the interaction between various queries and have no ability to set the MPL dynamically. There are three parts of the manager, first is Admission control that makes batches of queries according to the available memory. In this research, priority is given to that query which requires maximum memory. The underload and overload problem is managed by using the Priority Gradient Multiprogramming (PGM). The manager cannot manage the interactive and ad-hoc queries. Moreover, smaller queries remain suspended until the completion of long queries. The Query Progress Indicator (PI) for single query with a graphical user interface [19] is introduced to represent the status of query and provides information about the remaining execution time. It is limited to single query and does not consider the query dependency. A Multi-query PI [20] displays the time of running queries with the consideration of query dependency over others. The technique is the first attempt for a multi-

query PI. It has also the ability to predict future queries with an efficient workload management. The proposed PI is implemented in POSTGRE SQL and various experiments are performed to show its effectiveness. A Query Patroller (QP) for IBM DB2 is discussed in [4] which manage the database requests from the users as per availability of resources. It divides the workload into classes on the basis of their cost and Multi-Programming Level (MPL) threshold. Usually, DBA assign privileges to different resources in QP that analyze and schedule the workload after observation. Saturation is avoided by limiting the workload with maximum resource usage. The proposed advisor for tuning and optimization of query [21] is used to manage the workload in Oracle. The SQL Tuning Advisor is responsible to generate the recommendations for the given workload that is further used to generate best Query Execution Plan (QEP). It assists the query optimizer when fails due to heavy workload. The proposed manager for handling resources in Oracle [22] has the ability to allocate resources to given workload. OLTP workload execute before the DSS workload when there is an enormous number of requests from the users. ODRM also facilitate administrator by defining and scheduling policies for different requests. There are three main components Resource of the ORDM (Consumer Group, Resource Plan and Resource Plan Directive). Economic model is introduced by [23] that is used to manage the workload proactively. The workload is divided into various classes on the basis of cost and time limit. Experiments are performed with a claim that the proposed model is scalable with the workload size. A Priority Adaptation Query Resource Scheduler [24] manages the multiclass queries in an efficient way. The MPL is set according to the given workload by using the miss ratio projection and resource utilization heuristics. The proposed research has the ability to handle static workload; however, fails when there is rapid change in queries. A framework for the adaptation of workload [2] is introduced to manage the workload by using four modules and the workload is predicted by the Kalman Filter. The experiments of the proposed research are performed on the stable workload and can handle only the linear workload.

There is no identification of effective workload parameters in previous research. Some of these used the values in various calculations which are either estimated or static. Due to this reason, there results are inaccurate. Moreover, none of the above strategy or tool finds out the proportion of the OLTP and DSS by using the Fuzzy Logic. In this situation, a novel workload management approach is required where the database requests must be handled in an efficient way.

III. DB WORKLOAD MANAGEMENT

The proposed research work consists of three major modules that include workload characterization, scheduling and idleness detection. These modules are discussed as:

A. Characterization of DB Workload

Main components of the DB workload management are workload, resources and objectives which work in a loop to facilitate users. Workload has evolved from capacity planning to resource sharing and finally performance oriented [2]. Steps of the proposed workload management approach are shown in Fig 1. A Case Base Reasoning (CBR)

algorithm is adopted that uses the parameters to find out the workload type. It is explained in our previous research [24].

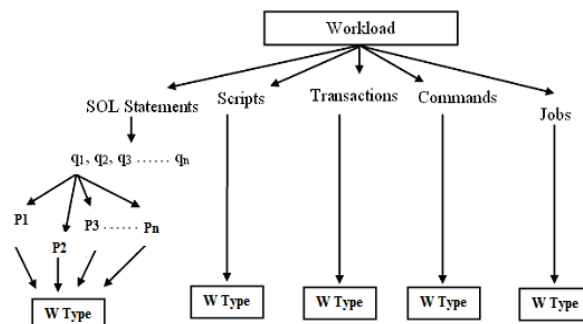


Fig. 1. Identification of DB Workload Type [24]

The suggested database workload characterization technique is verified and shown to be effective in classifying the workload type. For our experiments, we have employed the TPC-W benchmark workloads (ordering profile as OLTP and browsing profiles as DSS workloads). The values of the selected MySQL status variables are retrieved using the display status statement of MySQL and recorded in the database to help identify the type of workload. The workload is built using threads created by the client computers, and while it is running, values for the status variable are logged every 5 seconds. According to the workload characterization for the browsing workload, DSS accounts for 89.3 percent of the burden and OLTP for 11.7 percent. When it comes to ordering workload, it indicates that 8.5% of the workload is DSS and the rest is OLTP. The labor associated with the shopping-related activities is then carried out. Following its execution and analysis, it was discovered that the shopping profile has a workload split of 77% DSS and 23% OLTP. Due to the definition of the benchmark and our awareness of particular workload attributes, the outcomes of these trials are in line with what we had anticipated. To obtain accurate findings, the same tests are also run repeatedly.

B. Fuzzy Based (FB) Scheduler

The process of scheduling involves arranging the workloads produced by users and the system in a particular way for quicker execution. It is crucial to address all underlying problems and to set up the workload in the database so that each request is handled with the greatest possible CPU efficiency. Both internal and external scheduling can be done in DBMSs [16]. It is not required for DBMSs to have solely OLTP or DSS workload, i.e. workload that is 100% OLTP and 0% DSS or 100% DSS and 100% OLTP. If the workload is thought of as crisp, higher performance from the DBMS cannot be attained due to the mix of workload. The workload behavior in DBMS is typically non-deterministic; for instance, w1 and w2 are composed of 60% OLTP and 40% DSS transactions, respectively, and 20% OLTP and 80% DSS transactions. There is a need of a scheduler that can differentiate the OLTPness and DSSness of the incoming workload. The Fuzzy logic is adopted here to find out the workload type proportion from the database requests. Fuzzy logic is based on the Fuzzy Set Theory [25] where the value is always from 0 to 1. The crisp logic has only the Boolean value however in case of Fuzzy logic degree of the truthiness or falseness is identified. Following section discusses the architecture of the proposed scheduler for DBMS that uses the Fuzzy logic to

calculate the proportion of OLTP and DSS. Finally, Fuzzy rules and membership functions are applied and database requests are arranged accordingly.

Fuzzification, Fuzzification Rules and Defuzzification are the main steps of the FB scheduler. Membership function is used to calculate value in the range of 0 and 1. The Gaussian method is adopted in this research due to its simplicity and easy conversion to the Fuzzy set. User Priority (UP), DSS and OLTP are found by the membership function.

Consider a DB workload 'w' with various database queries is executed over MySQL 5.1. Proportion of both workload types is identified for all database requests. The OLTPness and DSSness of each workload is calculated. The proportion of OLTP workload is 0.473% and DSS workload is 0.527. Subsequently, these proportions are calculated for some specific time. A particular workload that is generated by the users during some specific time t . There were six sub-workloads from different users that are given to DBMS with different priorities. OLTP and DSS proportion is identified by adopting the same procedure. Finally, these sub-workloads are executed on the basis of these calculations to minimize the waiting time. After the fuzzification, the Fuzzy Rules in the form of "If Condition then Action" are designed and expressed in terms of Fuzzy words. In this scenario, eight (8) priorities (0 to 7) may be assigned to each user. The user with 0 priority has the highest priority and 7 for the lowest priority. There are 8 classes (Class 0 to 7) of the workload for execution on the basis of their size and importance. The execution of workloads starts from class 0 to class 7. It has been observed after experiments that much time is required for the calculation. So, number of classes minimized. The High class consists of the workloads with smallest time and high priority. In defuzzification step, final output is produced by conversion of Fuzzy Set to their corresponding output values. Workload type is found out by considering the above OLTP and DSS proportion.

C. Idleness Detection

The idleness detection approach constructs and stores sample data in the CBR as depicted in Table 1. The threads create a variety of workloads, and each thread is in charge of executing workloads that include OLTP, DSS, or a mix of query types. The Workload ID, OLTP Transactions, DSS Transactions, and DBMS Idleness are among the parameters saved in the CBR. The column Workload ID represents the automatically generated unique number, OLTP Transaction column shows the number of OLTP queries in the workload, DSS Transaction column shows the number of DSS queries in the workload, and the DBMS Idleness column represents the amount of time the given database workload was idle.

TABLE 1. CBR DATA FOR DBMS IDLENESS DETECTION

Workload ID	OLTP Transaction	DSS Transaction	DBMS Idleness (%)
1	12	5	28
2	5	3	57
3	7	11	19
4	15	5	33
5	31	3	26
6	2	5	19
.	.	.	.
.	.	.	.

In order to find the DBMS idleness, four steps are adopted that is Retrieve, Reuse, Revise and Retain. The

initial step to identifying the DBMS inactivity for the incoming workload is matching the OLTP and DSS Transactions of the currently running workload with cases that have previously been saved in the CBR.

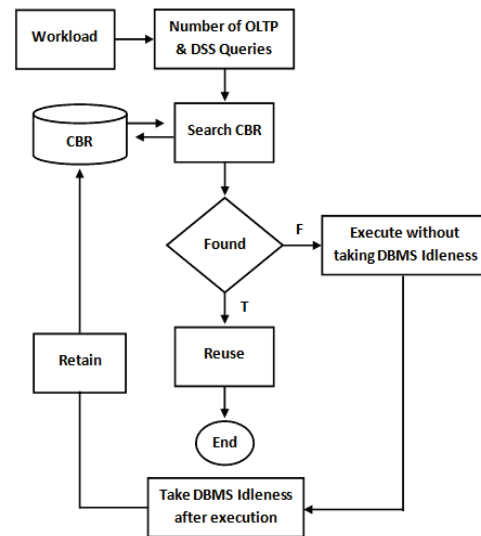


Fig. 2. Steps to Identify the DBMS Idleness

The retrieval is carried out using the problem description. It is vital to keep in mind that if equivalent instances are not found, cases may be incorrectly matched, leading to wrong results and judgments. This is essential since accuracy depends on how many cases are saved and how many matched cases can be retrieved. It is possible to gauge how similar two cases are by calculating the distance between them using similarity matrices that can be offered by experience or experts, for instance. Four further sub-steps are included in the retrieval stage: finding related features, making an initial match, searching, and selecting the matched instances from the CBR. The values of the selected features are then extracted from the incoming database workload. The workload in this scenario is used to determine the number of OLTP and DSS transactions or requests. The gap between the CBR recordings and the incoming workload's characteristics is calculated. Here, the Euclidean Distance technique is used to redetermine the matched or nearest cases [26, 27]. It is one of the easiest and quickest methods to locate the matched scenarios. The distance between the characteristics of the incoming workload and the stored examples is determined using the Euclidean Distance equation. After determining the distance between each CBR record and the matching record, the related DBMS idleness is used to determine the number of free cycles. However, if there isn't a match, the workload runs regardless of the DBMS's idleness, and thereafter, the DBMS's idleness throughout execution is recorded in the CBR along with the quantity of OLTP and DSS requests as a new case for later usage as shown in Fig 2.

Let's assume the DB workload consists of nine OLTP and thirteen DSS transactions. To find the DBMS idleness against this workload, we must now employ the match cases from the CBR. As indicated in Table 2, the values of the CBR and new workload for the workload parameter in this instance are not translated into the range [0, 1]. The gap between the new workload and every record stored in the CBR is calculated using the aforementioned Euclidean Distance calculation. The DBMS inactivity for the new job

originates from the row/record with the lowest distance value. In this example, the minimum Euclidean distance is 2.82842712, and the corresponding DBMS idleness is 19 (highlighted row 3). Consequently, the DBMS Idleness of the incoming workload will also be 19. The stated DBMS idleness is used by system processes. However, the workload continues to run even while the DBMS is not idle if neither an exact nor a close match can be obtained. After the task is over, it is maintained with the actual DBMS inactivity.

TABLE 2. EUCLIDEAN DIFFERENCE CALCULATION

CBR Data (in the range [0, 1])			New Workload (in the range [0, 1])		Euclidean Difference
OLTP	DSS	CPU Idleness (%)	OLTP	DSS	
12	5	28	9	13.0000	8.54400375
5	3	57	9	13.0000	10.77032961
7	11	19	9	13.0000	2.82842712
15	5	33	9	13.0000	10.00000000
31	3	26	9	13.0000	24.16609195
2	5	19	9	13.0000	10.63014581
.	.	.	.	.	.
.	.	.	.	.	.

After the Retrieve step is finished, the Reuse step begins, and it consists of two smaller steps. the distinctions between the calculations for the present and retrieved instances, followed by the identification of the cases that need to be added as new cases to the CBR. The revise step is required when some inaccuracy is found where domain knowledge is used to correct the results. Here, the Substitution adaptation method is adopted, where the transformation adaptation alters the structure of the solution. Finally in retain phase, only valuable information of the given case is inserted into the CBR for future use.

IV. RESULTS AND DISCUSSION

The results of this work are compared with FIFO, and PB techniques where average waiting time (AWT) is calculated for each by executing the DB workloads (w1, w2, w3, w4, w5, and w6). FIFO technique works on first in first out basis, SJF scheduling technique executes from smaller to larger request while PB executes on the basis of user priority. The results achieved from the proposed workload management are analyzed with other techniques. The wait time of proposed and others approaches is calculated as shown in following Table 3 and 4 (execution time of the selected workloads is 0 to 100 minutes).

TABLE 3. RESULT OF THE FIFO AND PROPOSED APPROACH (PA)

AO	Priority	OLT P %	DSS %	OLT P ET	DSS ET	Total Time	FIFO WT	PA WT
2	2	0.27	0.73	0.9	11.3	12.2	7.2	45.4
4	6	0.56	0.44	3.2	5.4	8.6	25	36.8
1	2	0.34	0.66	1.9	5.3	7.2	0	5.6
3	0	1	0	5.6	0	5.6	19.4	0
6	1	0.23	0.77	0.2	6.9	7.1	40.4	12.8
5	3	0.3	0.7	0.6	6.2	6.8	33.6	18.4
Average Wait Time							22.1	19.83

where AO are representing arrival order of the workloads.

TABLE 4. RESULTS OF THE PS AND PA

AO	Priority	OLT P %	DSS %	OLT P ET	DSS ET	Total Time	PS WT	PA WT
2	2	0.27	0.73	0.9	11.3	12.2	14.3	45.4
4	6	0.56	0.44	3.2	5.4	8.6	40.8	36.8
1	2	0.34	0.66	1.9	5.3	7.2	23.7	5.6
3	0	1	0	5.6	0	5.6	0	0
6	1	0.23	0.77	0.2	6.9	7.1	7.2	12.8
5	3	0.3	0.7	0.6	6.2	6.8	34.6	18.4
Average Wait Time							20.11	19.83

TABLE 5. AWT OF FIFO, PB AND PROPOSED APPROACH (PA)

Scenario	FIFO AWT	PB AWT	PA AWT
1	20.93	20.11	19.83
2	16.5	19.8	13.8
3	21.8	17.6	14.9

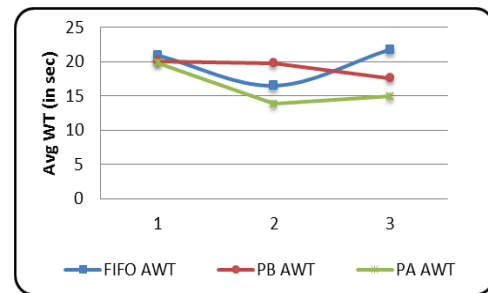


Fig. 3. AWT of FIFO, PB and Proposed Approach

After these experiments, two main problems are identified as shown in Table 5 and Fig. 3; first, smallest workloads have to wait for a long time in FIFO as well as PB; and second, both does not consider the importance i.e., whether the workload is generated from the higher management or not. In FIFO and SJF all the requests are considered at same level and executes from smallest to longest request without considering its importance for higher management. On the other hand, PB executes the workloads by only considering its priority. The proposed research work in all scenarios is better than other existing and well-known workload management techniques.

V. CONCLUSION AND FUTURE WORK

The study recommended a strategy for managing DB workloads that estimates each job's percentage of OLTP and DSS to determine how it is carried out. The workload parameters that are more helpful for classifying and characterizing the task are identified. The workload type is determined using the CBR approach, and the percentage of OLTP and DSS is determined using fuzzy logic. Once the OLTP and DSS percentages are established, the complete workload is arranged for better execution. Results from experiments with workloads that are similar to those for OLTP and DSS show that the suggested characterization and scheduling procedures are effective and take less time, taking workload size and importance into account. The assessments also provide a comparison with other well-known workload management programs. The suggested strategy's results are compared to First in First Out, Priority

Based, and Smallest Job First strategies. The main objective of the analysis and comparison with other commonly used techniques is to show the merits of the proposed study. The logical makeup of the workload, a clustering strategy to combine query templates, and any learning method that may forecast the rate at which the DB workload will arrive will all be considered in future workload.

ACKNOWLEDGMENT

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant 2020R1A2C1010366, and in part by the Basic Science Research Program through the NRF funded by the Ministry of Education under Grant 2021R1A6A1A03039493.

REFERENCES

1. M. C. Huebscher, and J. A. McCann (2008). A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys (CSUR)*, vol. 40, no. 3, 1-28.
2. B. Niu, P. Martin, W. Powley, R. Horman, and P. Bird. (2006, October). Workload adaptation in autonomic DBMSs. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research* (pp. 13-es).
3. S. S. Lightstone, G. Lohman, and D. Zilio. (2002). Toward autonomic computing with DB2 universal database. *ACM Sigmod Record*, vol. 31, no. 3, 55-61.
4. Z. Zewdu, M. K. Denko, and M. Libsie. (2009, May). Workload characterization of autonomic DBMSs using statistical and data mining techniques. In *2009 International Conference on Advanced Information Networking and Applications Workshops* (pp. 244-249). IEEE.
5. S. Elnaffar, P. Martin, and R. Horman. (2002, November). Automatically classifying database workloads. In *Proceedings of the eleventh international conference on Information and knowledge management* (pp. 622-624).
6. S. S. Elnaffar, and P. Martin. (2004, November). An intelligent framework for predicting shifts in the workloads of autonomic database management systems. In *Proc of 2004 IEEE International Conference on Advances in Intelligent Systems—Theory and Applications* (No. 15-18, pp. 1-8).
7. S. Y. Philip, H. U. Heiss, S. Lee, and M. S. Chen. (1992). On Workload Characterization of Relational Database Environments. *IEEE Trans. Software Eng.*, vol. 18, no. 4, 347-355.
8. T. J. Wasserman, P. Martin, D. B. Skillicorn, and H. Rizvi. (2004, November). Developing a characterization of business intelligence workloads for sizing new database systems. In *Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP* (pp. 7-13).
9. D. A. Menascé, V. A. Almeida, R. Fonseca, and M. A. Mendes. (1999, November). A methodology for workload characterization of e-commerce sites. In *Proceedings of the 1st ACM conference on Electronic commerce* (pp. 119-128).
10. M. Holze, and N. Ritter. (2008). Autonomic databases: Detection of workload shifts with n-gram-models. In *Advances in Databases and Information Systems: 12th East European Conference, ADBIS 2008, Pori, Finland, September 5-9, 2008. Proceedings 12* (pp. 127-142). Springer Berlin Heidelberg.
11. F. Liu, Y. Zhao, W. Wang, and D. Makaroff. (2004, October). Database server workload characterization in an e-commerce environment. In *The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings.* (pp. 475-483). IEEE.
12. M. Zhang. (2014). Autonomic workload management for database management systems (Doctoral dissertation, Queen's University), Kingston, Ontario, Canada.
13. S. Chaudhuri, R. Kaushik, and R. Ramamurthy. (2005, June). When can we trust progress estimators for SQL queries?. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 575-586).
14. B. Chandramouli, C. N. Bond, S. Babu, and J. Yang. (2006, April). On suspending and resuming dataflows. In *2007 IEEE 23rd International Conference on Data Engineering* (pp. 1289-1291). IEEE.
15. B. Schroeder, M. Harchol-Balter, A. Iyengar, E. Nahum, and A. Wierman. (2006, April). How to determine a good multi-programming level for external scheduling. In *22nd International Conference on Data Engineering (ICDE'06)* (pp. 60-60). IEEE.
16. M. Ahmad, A. Abounaga, S. Babu, and K. Munagala. (2008, April). Qshuffler: Getting the query mix right. In *2008 IEEE 24th International Conference on Data Engineering* (pp. 1415-1417). IEEE.
17. A. Mehta, C. Gupta, S. Wang, and U. Dayal. (2009, March). rFEED: a mixed workload scheduler for enterprise data warehouses. In *2009 IEEE 25th International Conference on Data Engineering* (pp. 1455-1458). IEEE.
18. A. Mehta, C. Gupta, and U. Dayal. (2008, March). BI batch manager: a system for managing batch workloads on enterprise data-warehouses. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology* (pp. 640-651).
19. G. Luo, J. F. Naughton, and P. S. Yu. (2006). Multi-query SQL progress indicators. In *Advances in Database Technology-EDBT 2006: 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006 10* (pp. 921-941). Springer Berlin Heidelberg.
20. B. Dageville, D. Das, K. Dias, K. Yagoub, M. Zait, and M. Ziauddin. (2004, August). Automatic sql tuning in oracle 10g. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (pp. 1098-1109).
21. A. Rhee, S. Chatterjee, and T. Lahiri. (2001). The Oracle database resource manager: Scheduling CPU resources at the application level. *HPTS*, vol. 1, no. 2, 2-4.
22. S. Krompass, H. Kuno, J. L. Wiener, K. Wilkinson, U. Dayal, and A. Kemper. (2009, March). Managing long-running queries. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* (pp. 132-143).
23. H. Pang, M. J. Carey, and M. Livny. (1995). Multiclass query scheduling in real-time database systems. *IEEE Transactions on knowledge and data engineering*, 7(4), 533-551.
24. M. Abdul, A. M. Muhammad, N. Mustapha, S. Muhammad, and N. Ahmad. (2014). Database workload management through CBR and fuzzy based characterization. *Applied Soft Computing*, vol. 22, 605-621.
25. L. Zadeh, (1965). *Fuzzy Sets and Systems*, Fox Journal, editor. System Theory, Brooklyn, NY: Polytechnic Press, pp. 29-39.
26. E. Deza, M. M. Deza, M. M. Deza, and E. Deza. (2009). *Encyclopedia of distances* (pp. 1-583). Springer Berlin Heidelberg, ISBN 978-3-642-00234-2/ebook, Springer.
27. P. E. Danielsson. (1980). Euclidean distance mapping. *Computer Graphics and image processing*, vol. 14, no. 3, 227-248.