International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

A Novel Gmtds Algorithm for Dynamic Management of Transaction under Different Workload Condition

Mohammad Sharfoddin Khatib, Mohammad Atique

Abstract: ***In today's scenario large enterprise world spread across different locations, continents or having a diverse presence over the globe, where data is enormous and handling such large data over distributed computing becomes critical in real-time database system. The transaction management system for the distributed environment must ensure that the sequence of updates in the stable warehouses in different locations is confirmed or cancelled safely as a single complete unit of work. Working with Real-Time Database System (RTDBS) and that to on a distributed computing system is a tough task. When we work with distributed environment over larger database, we need to take care of the transaction time period as well as number of transactions that are actually executed (committed) and number of transactions fail. The application on dynamic RTDBS becomes more complex when certain deadlines need to be completed. In this paper, we had carried out the test of CRUD (Create, Read, Update, and Delete) operation on transactional Real-time databases in real time dynamic distributed environment by using the existing EDF, GEDF algorithm and we had compared these algorithms with our proposed GMTDS algorithm in standalone and distributed environment with a dynamic self adaptive approach for management of transactions.***

Keywords: ***RTDBS, NoSQL Databases, Transaction Management, GMTDS, GEDF, RDBMS, FCFS, EDF, MDTS.***

I. INTRODUCTION

As the world becomes smarter and more computing demands are increasing. Many converging technologies are emerging, such emerging IT delivery model is cloud computing. The system's objective is to meet deadlines. The two-phase confirmation protocol is used to resolve these problems. The proper scheduling of transactions and the management of their execution time are important factors in the design of such systems.

Manuscript received on February 10, 2020.
Revised Manuscript received on February 20, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Mohammad Sharfoddin Khatib*, Faculty of Computer Science and Engineering Department, Anjuman, College of Engineering and Technology, Nagpur, India. Email: sharfoddin.khatib@gmail.com
Prof. Dr. Mohammad Atique, P.G. Department of Computer Science and Engineering, Sant Gadge baba Amravati University, Amravati, India. Email: mohammadatique@sgbau.ac.in

Transaction processing on any database system can be restricted in real time. In the event of failure, it is not possible to provide this guarantee. The proper scheduling of transactions and the Management of their execution are the important factors in the design of such systems.

The performance of the system is usually measured in terms of the number of transactions completed before the deadline. Transactions that miss processing are simply deleted or cancelled and executed in the system without being discarded until completion.

To address the problems of Dynamic management of real time databases, various approaches had been proposed. Our main objective in this paper is to minimize the transactions execution time by applying our proposed GMTDS algorithm.

The paper is organized as per the following sections. A literature review of the existing system is mentioned in section 2. In section 3, 4, 5, 6, 7 proposed work is focussed and the detail comparison experiments of EDF, GEDF and proposed GMTDS algorithm in standalone and distributed environment with a dynamic self adaptive approach for management of transactions. In section 8 the results is discussed, in section 9 the general conclusion is mentioned.

II. RELATED WORK

Concurrent transactions must observe protection mechanisms or timestamp protocol or optimistic concurrency management protocol. Global transactions can encounter various resistances, such as host failure, network association failure or dead end and finally global transactions are responsible for recovering aborted transactions. Web-based and web-based distributed businesses are practical with normal distributed transaction processing systems [1]. Transactions initiate all discrete tasks that must be performed as a unit to achieve a goal.

Lingli LI, Hongzhi WANG and others study the problems associated with managing uncertain data, present cutting-edge solutions and provide instructions for future research in this field. A distributed storage model with multidimensional parity control has been described, which has less redundancy than replication and reliability at the level of complex systems that are used in the Reed-Solomon algorithm [2].

The theoretical justification for the multidimensional parity verification method and algorithm is performed, as well as the estimation of reliability parameters and their advantages over existing distributed memory methods. The technology is implemented in distributed databases [3].

To improve the security of economic finance management data storage a multi-level encryption algorithm has been proposed for the storage of distributed financial management database based on partial merger and information clustering encryption. As part of the arithmetic key cryptography system a structure model for the structure of distributed data storage for economic financial management is constructed and cryptographic statistical resources for economic financial management data are extracted [4]. In a conventional designated environment transactions are short-lived and sources are blocked during the transaction and participants act below the transaction manager. Transactions distributed over Internet and Internet services are long. Participants may now not be able to reserve their assets for an extended period.A network based primarily on the conversation infrastructure allocated between members is not always reliable at the moment. It depends on the pattern of the verbal exchange network. An online transaction may require triumph even if only some people choose to verify the transaction and others cancel it. All activities in the distributed environment are recorded. A transaction that must be undone means compensation that also performs transactions to restore the system to its previous state [5].

As the world becomes smarter and additional computing, IT demands increase. Many converging applied sciences are emerging as the emerging model of IT delivery: cloud computing. There are many transaction complexities to deal with concurrency control and recovery of the distributed database systems. The two-phase commit protocol is used to solve these problems. To guarantee a certain atomicity of the transaction, confirmation protocols are implemented in the distributed database system.

A uniform commitment is guaranteed using a commitment protocol in the execution of an assigned transaction to ensure that all participating sites agree with the latest result. The result can be a confirmation and cancellation condition. In a real-time database device, the transaction processing device designed to handle workloads, the place where transactions have full deadlines.

To perform certain atomicity transactions, the confirmation protocol is applied to a distributed database system. Experimental results of scheduling transactions below the range of workloads and the unique configuration of the system are evaluated using this simulation [6]. Many database researchers have proposed types of confirmation protocols, such as two-phase confirmation and nested two-phase confirmation, presumed confirmation and presumption cancellation, transmission of two-phase confirmation, three-phase confirmation etc. This requires the exchange of some messages, in more than one phase, between the participating sites where the waived transaction was executed. Several log records are generated to make eternal changes to the statistics disk, which bothers an additional time of execution of the transaction. Proper scheduling of transactions and management of their execution time are necessary factors to design these systems [7] [8] [9].

Transaction processing on any database system can be restricted in real time. Programming transactions with closing dates in a database machine residing in the memory of a single processor were developed and evaluated by simulation. Real action such as firing a gun or allocating money may now also not be worthwhile. Properly scheduling transactions and managing their execution time are important elements in the design of such systems. In this database, the overall performance of the confirmation protocol is generally measured in terms of a variety of transactions that are executed before their deadlines. The transaction that skips its deadlines before the end of processing is simply deleted or aborted and discarded from the device without being completed until its completion. Real-time disk programming plays an important role in time-constrained applications. The real-time database system depends not only on strict data consistency requirements, but also on the timing of the results [10]. Several previous RTDBS studies have been conducted to address the problem of scheduling transactions in order to minimize the number of failed transactions. The main objective of RTDBS is to maximize the value obtained by transactions on time and to minimize the number of failed transactions [11] [12]. Priority assignment is a key issue in the transaction scheduling algorithm. It is affected by many factors, such as resource requirements, degree of urgency, time constraints, etc. Depending on the type of transaction, design a multi-dynamic priority assignment policy using deadline and downtime [13].

III. PROPOSED WORK

The proposed model is executed in a standalone system and a client server environment. In distributed transactions we had considered one process naming Master and another set of processes called child or cohorts. The child executes the process transactions at various distributed network domains and that we had accessed over the distributed client-server architecture. The main server process the setup of the collection of all client network datasets which interns execute the individual process on their own. When the client completes the execution process, it sends signals of acknowledgments to the Server.

The server consists of various sizes of the dataset on its sites and then the client triggers the CRUD set of queries paralleled. The client executes the steps and transaction is committed, maintains its log on client-side and sends positive data to sever that transaction is either successfully committed or not.

The client had executed the prepared statement or SQL queries and in-between the client can't commit or abort the set of queries getting already executed until the server sends some kind of deciding whether to abort the transactions or commit the transactions. Whereas on the other side every client who decides to abort the transaction will maintain its log file and sends a negative signal to the server. Once the entire client sends the transaction messages to the server than from server the second phase is executed. If all clients send a positive acknowledgment to sever, the server maintains its log and sends a commit message to all clients. If the server receives any negative message from any client about the transactions, then the server sends the aborting transaction message to the respective client and also maintains its log. The clients after receiving the abort message, execute the respective abort command, write and abort log and sends the message to the server. At the end of the Server end, after receiving all the messages from clients, the server log file is again updated and keeps ready the state for the next transactions.

IV. TRANSCATIONAL SCHEDULING USING EDF ALGORITHM

The First Earliest Deadline First algorithm is an analogy of FCFS. Requests are classified according to the deadline and the request with the oldest deadline is answered first. By assigning priorities to transactions, an earlier term policy minimizes the number of transactions that are delayed on systems that operate under low or moderate levels of resource and data contention. Hence the first deadline gives top priority to transactions that have the shortest time left to complete. However, the performance of the transaction deadline decreases in an overloaded system. This is because, under a heavy load, transactions get top priority when they are close to deadlines. Achieving high priority at this advanced stage may not allow enough time for transactions to be completed ahead of schedule. Under heavy load, a weakness of the Earliest Deadline priority schedule is that it assigns the highest priority to transactions that are close to missing their deadlines, thus delaying other transactions that might still be able to meet their deadlines. The EDF only consider the order of deadlines and introduces huge amount of seek-time costs with poor disk throughput.

A. Illustration of EDF

EDF arranges the transactions in the increasing order of their deadlines.
Minimum Deadline ----- > Highest Priority
Maximum Deadline ---- > Lowest Priority
Following table and graph shows the average response time in milliseconds when we had executed the transaction by increasing the size of database and given certain deadline at runtime.

Table I: Average Response Time

Transaction Record Size	Average Response Time(in MS)
10000	3012
20000	6063
30000	9601
40000	11734
50000	14940
60000	16800
70000	19234
80000	21568
90000	24562
100000	29192

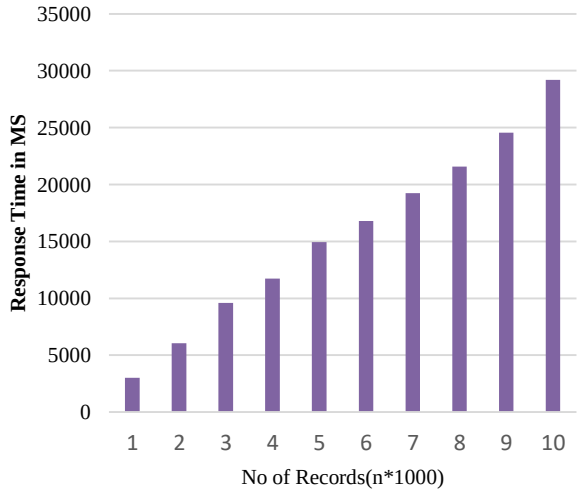
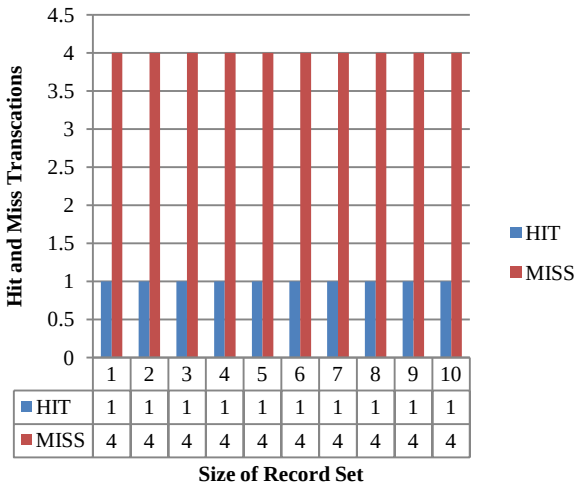


Fig. 1 Response Time Taken for Various Size of dataset

When we analysed by applying EDF algorithm dynamically on various size database ranging from 10000 to 100000, within a deadline mentioned at run time , we observed that the transaction with get hit is only 1 on or average and 4 are getting missed. Hence ration if we calculate is only 20%. Following graph show the hit and miss transaction before deadlines in the transactions on database size 10000 to 100000.

Table II: No of Hit and Miss for EDF Algorithm

Size of Record Set	Total Hit	Total Miss
10000	1	4
20000	1	4
30000	1	4
40000	1	4
50000	1	4
60000	1	4
70000	1	4
80000	1	4
90000	1	4
100000	1	4



	1	2	3	4	5	6	7	8	9	10
HIT	1	1	1	1	1	1	1	1	1	1
MISS	4	4	4	4	4	4	4	4	4	4

Fig. 2 No of Hit and Miss for EDF Algorithm

V. TRANSCATIONAL SCHEDULING USING GEDF ALGORITHM

In the GEDF algorithm based on the EDF algorithm, where transactions with "similar" terms are grouped (i.e., terms very close to each other) and the Shortest Job First (SJF) algorithm is used to schedule transactions within a group. It is used both in overload and under load conditions. It is useful for real-time systems, as well as for applications known as "approximate algorithms" and "algorithms at any time", where applications generate more results or account rewards with longer run times. A ti transaction in a real-time system is defined as ti = (ri, ei, Di, Pi); where you laugh is your launch time (or your arrival time); ei is the worst case or the average execution time; Di is your term and Pi is the periodicity of the transaction. This algorithm generated a fixed number (N) of request with different arrivals, execution times and deadlines. Orders must be mutually independent.

A group in the GEDF algorithm depends on a group range parameter Gr. Tj belongs to the same group as you if di <dj <(di + Gr * (di - t)), where t is the current time, 1 <i, j <N. In other words, orders are very close deadlines. Groups are scheduled based on EDF. All requests in a group with an earlier deadline are considered for scheduling. Requests in a group with later deadlines are schedule requests within a group using the shortest Job First. As shortest Job First results in more completions, intuitively, GEDF should lead to a higher success rate than pure EDF.

A. GEDF Algorithms

QGEDF is a queue for GEDF programming. The current time is represented by t. | QGEDF Represents the length of the tail. A group in the GEDF algorithm is defined as

GEDF Group = {Tk | Tk € QGEDF, dk-d1 ≤ D1 * Gr, 1 ≤ k ≤m Where m ≤ | QGEDF}

D1 = deadline of the first transaction, that is, the lowest

Gr = group rating factor = 0.4 (that is, all transactions whose terms are within 40% of the current transaction term are in the same group).

a. Enqueue (QGEDF, T)

If (Ti's deadline d >t) then

Insert transaction T into QGEDF by EDF i.e. di ≤ di+1 ≤di+2,

Where Ti, Ti+1, Ti+2 € QGEDF, 1 ≤ i ≤|QGEDF| - 2;

End

b. Tmin = Dequeue (QGEDF)

If QGEDF ≠ ø then

Find a transaction Tmin with emin=min

{ ek|Tk € QGEDF, dk – d1 ≤ Gr * D1, 1 ≤ k ≤ m, where m ≤ |QGEDF| };

Run it and delete Tmin from QGEDF;

End

Queuing is called when transactions arrive and queuing is called when the disk is inactive. The algorithms classify transactions in each group, which can generate more overhead during execution than EDF.

B. Illustration of GEDF

Step I: Insert the transactions in EDF order.

T1 T0 T2 T4 T5 T7 T6 T3

6 10 15 17 19 21 24 26 → Deadlines

d0 – d1 = 10 – 6 ≤ 2.4 is False

G1 = {T1}

Group2 G2:

d0 * 0.4 = 10 * 0.4 = 4

d2 – d0 = 15 – 10 ≤ 4 is False

G2 = {T0}

Step II: Repeating the same procedure the final groups are formed:

G1 = {T1}

G2 = {T0}

G3 = {T2, T4, T5, T7}

G4 = {T6, T3}

Step III: Apply SJF (Shortest Job First) in each group.

The transaction having less Average Execution Time (AET) will be served first. If the AET is same then check Arrival Time. The transaction having minimum Arrival Time will be served first.

For group G3

G3 = T2 T4 T5 T7

7.5 6 7.5 7.5 → AET

0 5 4 6 → Release Time

After arranging the transactions of G3 in increasing order of arrival time for the same AET

G3 = T4 T2 T5 T7

In the same way,

G4 = T6 T3

Final schedule for GEDF will be

T1 T0 T4 T2 T5 T7 T6 T3

C. Performance Analysis of GEDF Algorithm

Following table and graph shows the average response time in milliseconds when we had executed the transaction by increasing the size of database and given certain deadline at runtime by using GEDF Algorithm.

Table III: Average Response time for GEDF

Transaction Record Size	Average Response Time(in MS)
10000	2699
20000	4767
30000	6916
40000	9207
50000	11469
60000	13042
70000	16278
80000	18196
90000	20513
100000	25174

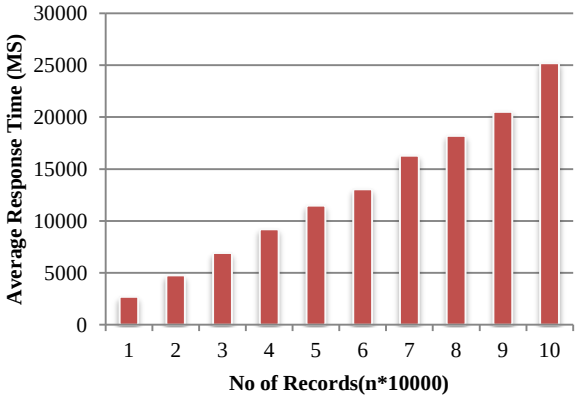


Fig. 3 Average Response Time

When we analysed by applying gEDF algorithm dynamically on various size database ranging from 10000 to 100000, within a deadline mentioned at run time , we observed that the transaction get hit is only 2 on an average and 3 are getting missed. Hence ratio, if we calculate is only 40%. Following graph show the hit and miss transaction before deadlines in the transactions on database size 10000 to 100000.

Table IV: No of Hit and Miss for GEDF Algorithm

Size of Record Set	Total Hit	Total Miss
10000	2	3
20000	2	3
30000	2	3
40000	2	3
50000	2	3
60000	2	3
70000	2	3
80000	2	3
90000	2	3
100000	2	3

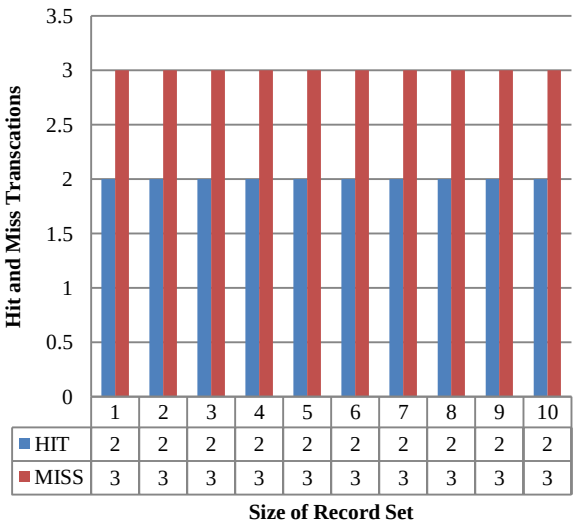


Fig. 4 No of Hit and Miss for GEDF Algorithm

VI. TRANSCATIONAL SCHEDULING USING GMTDS ALGORITHM

In our proposed algorithm we have divided transactions into three groups HT, ST and NT same as in MDTS except that before giving service to the grouped transactions, GEDF algorithm is applied to further divide them into groups. And then transactions are served considering highest priority group first. Proposed algorithm based on MDTS and GEDF algorithms is given below:

- As transactions are separated in three levels by type, ou seja, real time transaction (HT), real time transaction (ST), real time transaction (NT). Its priorities are defined as: Prio (HT)> Prio (ST)> Prio (NT). Next, different types of transactions use different programming policies to assign priorities.
- Lowest priority to the non-real time transaction. Here one transaction is consider for NT
- Initially we assign the level no. to each transaction randomly
- According to level no. we decide which is HRT, SRT and Non-Real Time and then assign priority to each transaction.
- Apply G-EDF on HT and ST groups and schedule the transactions considering highest priority group first.
- For hard transaction

$Pi = (\alpha * D + (1-\alpha) * S)$

Where, $S = de-(t0 + AET)$

$D = de-t0$

- 7. For soft transaction

$Pi = S$

Where, $S = de-(t0 + AET)$

A. Performance Graph for GMTDS

Following table and graph shows the average response time in milliseconds when we had executed the transaction by increasing the size of database and given certain deadline at runtime by using GMTDS Algorithm.

Table V: Average Response Time for GMTDS Algorithm

Transaction Record Size	Average Response Time(in MS)
10000	2671
20000	4734
30000	6705
40000	8850
50000	11326
60000	12718
70000	14310
80000	17807
90000	18253
100000	23192

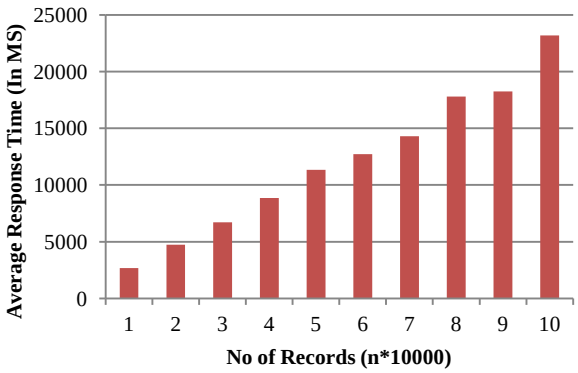


Fig.5 Average Response Time for GMTDS Algorithm

When we analyzed by applying GMTDS algorithm dynamically on various size database ranging from 10000 to 100000, within a deadline mentioned at run time , we observed that the transaction that get hit is only 4 on an average and 1 are getting missed. Hence ratio, if we calculate is only 80%. Following graph show the hit and miss transaction before deadlines in the transactions on database size 10000 to 100000.

Table VI: No of Hit/Miss for GMTDS Algorithm

Size of Record Set	Total Hit	Total Miss
10000	4	1
20000	4	1
30000	4	1
40000	4	1
50000	4	1
60000	4	1
70000	4	1
80000	4	1
90000	4	1
100000	4	1

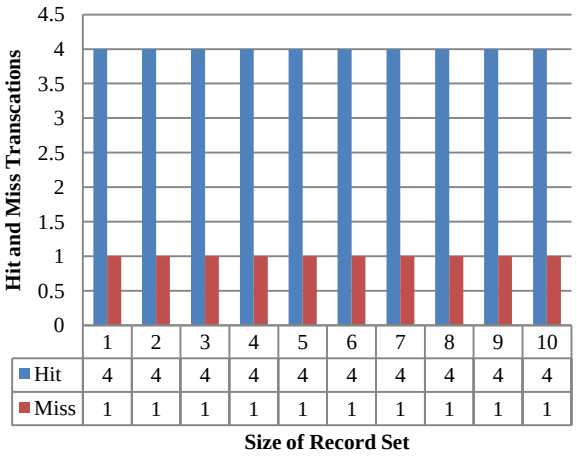


Fig. 6 No of Hit/Miss for GMTDS Algorithm

VII. PROPOSED DYNAMIC SCHEDULING GMTDS ALGORITM

In all above methodologies which is implemented with the various deadlines and when we observe various seek time or actual execution time, we found that for small amount of data size (ranging from 1000,5000,10000 data size), the execution time or response time is low in terms of EDF algorithm and when data size is in range of 10000 to 30000 size then GEDF is better. Whereas although GMTDS performs better for all datasets size but still we can consider it better response time for data size greater than 40000. Based on this observation we had developed our proposed methodologies with time constraint as parameters and dynamic self-adaptive system for real time database management is proposed. Following algorithm explains the same.

A. Algorithm: Dynamic Management

START

- Get the data size of the database
- If the ds>1000 and ds<20000
  select procedure for edf Algorithm
  else if ds>20000 and ds<40000
  select procedure for GEDF algorithm
  else
  select procedure for GMTDS algorithm
- repeat steps 1 and 2 till end of execution

END

VIII. RESULTS AND DISCUSSIONS

To perform the results on the dataset which we had kept for consideration, the following parameter setting is done.

Parameters	Description
No of client Sites	3
Database Size	Ranging from 1000 records to 100000 records
CPU Process	Core I3 2.2 GHz processor
RAM Size	8 GB
Arrival Rate	4/5 jobs per sec

A. Results and Discussion of Proposed Dynamic Adaptive Approach using GMTDS

In this section, the results of the proposed new algorithm are compared with the previous ones. First deadline (EDF), the group first deadline (GEDF) and the new GMTDS approach do able in a simulator. The simulation is disabled to model as requests from the database and the number of records (database number). The return of the algorithm is related to the number of successful transfers and the response time.

Based on experimental results, our GMTDS approach can support more transactions than other approaches. In particular, according to previous experiments, our approach can successfully complete missed transactions for other approaches. This is because our approach applies the first approach with the shortest research time and also verifies viability. As a result, compared to previous approaches, more transactions can be made ahead of time.

Table VII: Comparison of algorithm for No of Hit/Miss

No of Records	EDF	GEDF	GMTDS
10000	1	2	4
20000	1	2	4
30000	1	2	4
40000	1	2	4
50000	1	2	4
60000	1	2	4
70000	1	2	4
80000	1	2	4
90000	1	2	4
100000	1	2	4

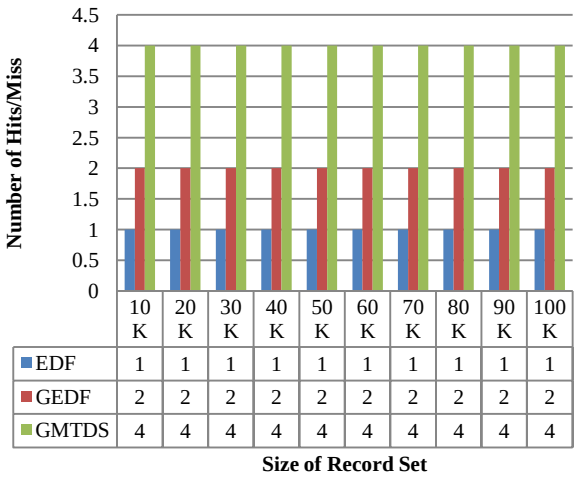


Fig.7 Comparison of algorithm for No of Hit/Miss

The response time or fulfil time, that is, start time of the transaction till its completion time is another important factor in measuring performance of a real-time database management system algorithm. If the same input requests are given, a well-behaved real-time database management system algorithm should finish the schedule as quickly as possible to maximize data throughput. Table 8 summarizes the average response time (ms) by taking 10000 to 100000 real-time transactions for comparison.

Notably, based on the above experiments, in our approach response time is less as compared to other approaches. This is because our approach applies feasibility technique. The transactions are only served if they are feasible. As a result, compared to previous approaches throughput of the system is improved.

Table VIII: Average Response Time Comparison

	EDF	GEDF	GMTDS
10000	3012	2699	2671
20000	6063	4767	4734
30000	9601	6916	6705
40000	11734	9207	8850
50000	14940	11469	11326
60000	16800	13042	12718
70000	19234	16278	14310
80000	21568	18196	17807
90000	24562	20513	18253
100000	29192	25174	23192

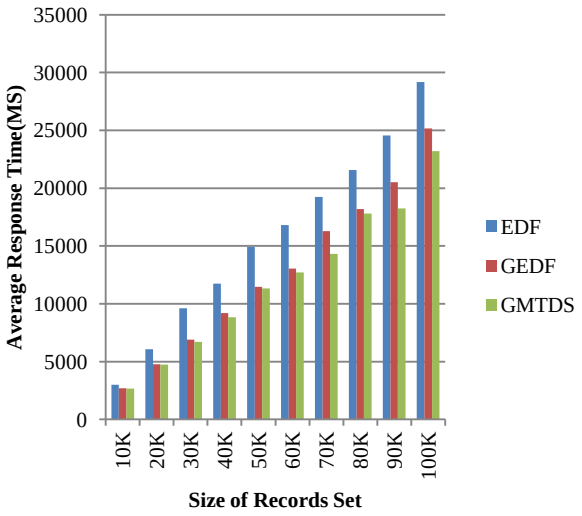


Fig.8 Comparison of Average Response Time

B. Results and Discussion in Dynamic Self Adaptive approach using GEDF and GMTDS algorithm.

In this approach, we are comparing the results between two algorithms i.e. GEDF and GMTDS. The dynamic management of real time database system algorithms group Earliest Deadline First (GEDF) and our approach Feasible GMTDS are simulated. The simulation is built to model the database requests and number of records (database size). The performance of algorithm is tested with number of successful transactions and response time.

The number of transactions supported, that is, number of transactions that are hit is one of the most important factors in measuring performance of a real-time dynamic management system scheduling algorithm. Being already we had discussed in section VII that we had designed the dynamic self-adaptive approach based on average response time and number of hit and miss transaction of various algorithms and we had selected that our proposed GMTDS approach performs better when the record size goes beyond 40000 records in a table. Hence we had implemented our system of dynamic self-adaptive in such a way that till record size of 10000 to 40000, the GEDF algorithm is selected and executed and beyond 40000 record set till 100000 record set, GMTDS algorithm is selected dynamically. We then calculate number of hit and miss transactions of these two algorithms. We had also compared the average response time when these two algorithms had been selected one by one from 10000 to 100000 record set. It is still observed that our proposed GMTDS algorithm is outperforming GEDF algorithm even in self adaptive dynamic management. In Table IX, we summarize number of database requests that can hit after selecting the algorithm. Table X gives us average response time in self adaptive approach after selecting the GEDF and GMTDS algorithm dynamically.

Table IX: No of Hit Transaction in Dynamic self-Adaptive Mechanism

No. of Record sets	GEDF (No of Hits)	GMTDS (No. of Hits)
10000	2	
20000	2	
30000	2	
40000	2	
50000		4
60000		4
70000		4
80000		4
90000		4
100000		4

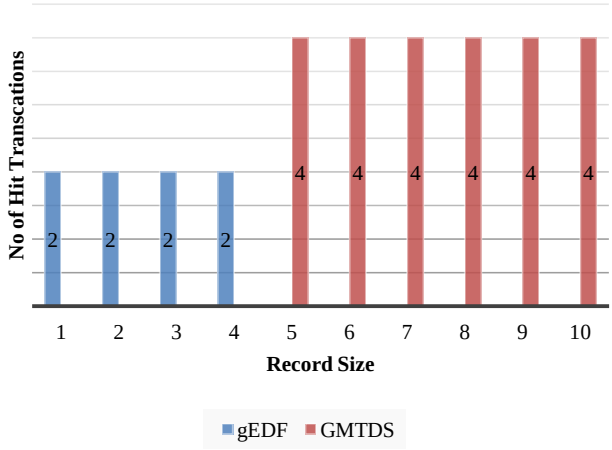


Fig.9 No of Hit Transaction in self-adaptive Dynamic

In table X below, the average response time after selecting the GEDF or GMTDS algorithm dynamically is recorded. Here again it proves that response time of GMTDS is less compared to GEDF algorithm.

Table X: Average Response time in Dynamic Self Adaptive

No. of Record sets	gEDF	GMTDS
10000	2880	2730
20000	5201	4892
30000	7018	6810
40000	9218	8927
50000	11510	11419
60000	14412	12760
70000	17910	14591
80000	18756	17992
90000	20702	18342
100000	25754	23929

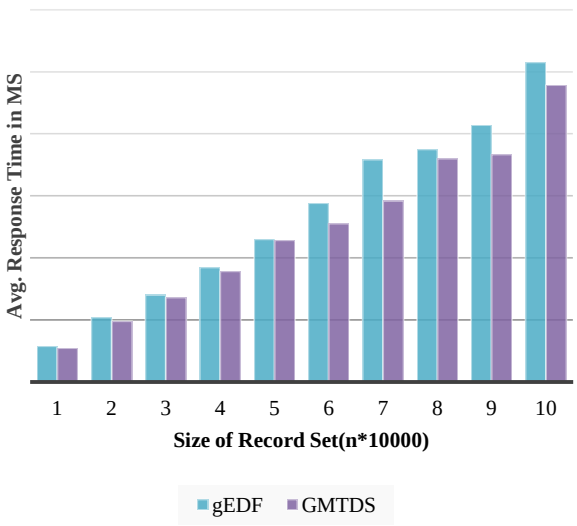


Fig.10 Average Response Time (MS) in self-adaptive dynamic approach

C. Results and Discussion in Dynamic Self Adaptive approach on client server architecture.

In this section, we had made two machines as client server architecture where on one machine we had kept the dataset ranging from 10000 to 100000 in number as we had done in earlier sections. On Second machine which we had treating as client machine we are running the self-adaptive dynamic selection of algorithm where we had taken the algorithm again as EDF, GEDF and GMTDS. We are calculating average response time to execute the task of all datasets and here also we had observed that our proposed GMTDS algorithm performs better even on dynamic selection in client server architecture. Table XI show the average response time in Millisecond's and Figure 11 show the analysis graphs of all three algorithms.

Table XI: Average Response time in Client Server Architecture

No of Record Sets	EDF	gEDF	GMTDS
10000	9234	8451	6230
20000	11124	9340	7823
30000	13783	10293	8945
40000	14891	12902	10342
50000	16903	13982	12091
60000	18567	14996	13128
70000	21091	16785	14893
80000	23981	18012	16930
90000	26789	20992	18945
100000	28970	23967	20893

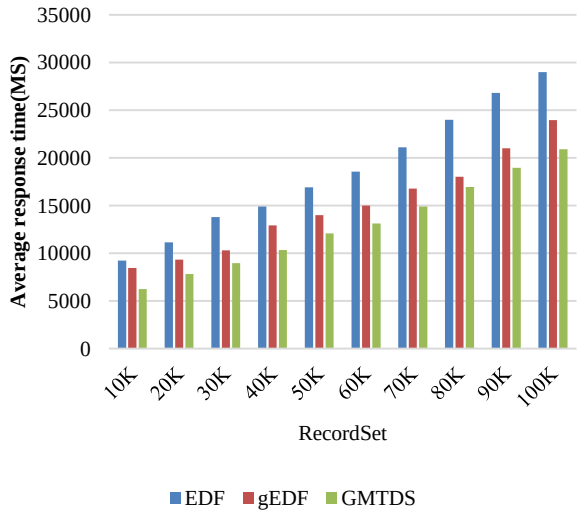


Fig.11 Average Response Time in Client Server Architecture

IX. CONCLUSION AND SUMMARY

In this work of self adaptive dynamic transaction management system we had performed a series of experimentation on different workload condition of transactions and analyze the performance of transaction management on a standalone system and in client-server architecture by using various transactional algorithms like EDF, GEDF and GMTDS. We had also done the experimental evaluation with our dynamic approach of GMTDS on various runtime operations by implementing CRUD approach. The scenario was tested under different workloads and execution methodologies. The scheduling of accessing the datasets through various commands is analysed so that it can meet the deadlines and minimizes the number of transaction which messes the deadlines.

We had observed that the performance of our dynamic approach of proposed GMTDS algorithm is much better in comparison to other algorithms like EDF, GEDF. The number of hit and miss ratio is almost 80% in case of our proposed algorithm, where as it is 60% in case of GEDF and 20% in case of EDF algorithm which proves that by performance perspective our proposed algorithm is better. Not only this but even the average response time is also far better than in our proposed GMTDS algorithm compared to other two algorithms.

Our future scope involves the implementation of a proposed GMTDS algorithm on multiple distributed systems and with NOSQL databases where we can observe quite reduction in response time as the databases will be in non-relational type.

REFERENCES

1. Jayanta Singh, S.C Mehrotra, 2006, "Performance analysis of a Real Time Distributed Database System through simulation" *15th Iasted International Conf. On Applied Simulation & Modelling, Greece*
2. Lingli LI, Hongzhi WANG , Jianzhong LI, Hong "A survey of uncertain data management"*Higher Education Press and Springer-Verlag GmbH Germany, part of Springer Nature 2018*
3. Kakim Sagindykov, Kalin Dimitrov "Justification of the Method and Algorithm of Multidimensional Parity Control in Distributed Databases of Information Systems*" IEEE National Conference with International Participation Conference "Electronica 2019", May 16 - 17, 2019, Sofia, Bulgaria*
4. Ramamritham,Son S. H,, DiPippo L,2004, Real-Time Databases and Data Services, *Real-Time Systems J., vol. 28, 179-216.*
5. Nystrom D, Nolin M, 2006, Pessimistic Concurrency Control and Versioning to Support Database Pointers in Real-Time Databases, *Proc. 16th Euromicro Conf. on Real-Time Systems*
6. S.Y.Amdani, M.S. Ali-2013, "An Improved Group-EDF: A Real Time Disk Scheduling Algorithm" , *International Journal of Computer Theory and Engineering, Vol 5,No.6, December 2013*
7. M S Khatib, Prof. Dr. Mohammad Atique 2017, Adaptive EndState Concurrency Control Real-Time Distributed Database, *International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2017)*
8. Dawei Sun, Guangyan Zhang, Shang Gao "Data Management across Geographically-Distributed Autonomous Systems: Architecture, Implementation, and Performance Evaluation" *2019 IEEE 21st International Conference on High Performance Computing and Communications*
9. Chengya Shang, Xianqiang Bao*, Lijun Fu, Li Xia, Xinghua Xu, Chengcheng Xu "A Novel Key-value based Real-time Data Management Framework for Ship Integrated Power Cyber-Physical System"*2019 IEEE PES Innovative Smart Grid Technologies Asia*
10. S.Y. Amdani, M.S. Ali-2011," An Overview of Real Time Scheduling Algorithms." *International Journal on Emerging Technologies 2(1): 126-130(2011).*
11. Mayuresh Kunjir "Speeding up AutoTuning of the Memory Management Options in Data Analytics*" Springer Science+Business Media, LLC, part of Springer Nature 2020*
12. Mohammad Sharfoddin Khatib, Mohammad Atique "FGSA for optimal quality of service based transaction in real-time database systems under different workload condition" *Springer Science Business Media, LLC, part of Springer Nature 2019*
13. XIONG Xin,SHEN Ya-hui1,FENG Guo-li,GUO Ming-jing ,WANG Bin "Research on Multi-level distributed Storage method of Economic and Financial Management Database" *IEEE 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*

AUTHORS PROFILE

Mohammad Sharfoddin Khatib, received a BE from Government College of Engineering, Amravati, in 1993, and ME from Prof. Ram Meghe Institute of Technology and Research Engineering, Badnera, Amravati in 2001. He is a life member of ISTE, a life member of CSI. He is currently pursuing for a doctorate degree under the supervision of Prof. Dr. Mohammad Atique in the faculty of Computer Science and Engineering at PG Department of Computer Science, Sant Gadge Baba Amravati University, Amravati. He is presently working as an Associate Professor in the Department of Computer Science and Engineering, Anjuman College Of Engineering & Technology, Nagpur.

Mohammad Atique, received BE in 1990, ME in 1997 and Ph.D. in 2009, respectively, in Computer Science and Engineering from the Sant Gadge Baba Amravati University, Amravati. He is a member of IETE, New Delhi, IE, Kolkata and principal member of CSI, Mumbai. He worked as an associate professor in computer science and engineering at the Government Engineering Faculty, Amravati, from September 1990 to November 2007. He currently works as a professor in P.G. Department of Computer Science and Engineering at Sant Gadge Baba Amravati University, Amravati, since October 2011. His area of interest includes image recovery, image processing, neural networks, parallel processing, real-time distributed systems, bases of distributed data and soft computing. He is the author of more than 70 research articles and has published two patents.