ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Overview of Overload Management in Real Time Database Systems

Rutuja A. Gulhane[1]
Department of Computer Science and Engineering
Prof. Ram Meghe College of Engineering and Management,
Badnera, Amravati University,
India.

Dr. M. S. Ali[2]
Prof. Ram Meghe College of Engineering and Management,
Badnera, Amravati University,
India.

Abstract - As the complexity of real-time systems and application is going up, the amount of information to be handled by real-time systems increases, motivating the need for database and data service functionality. A conventional DBMS aims to maximize transaction throughput and minimize response time. However, a real-time DBMS aims to provide predictability in transaction processing and offer different quality of services. To address this need, it requires the architecture of a real-time database management system which includes a real-time control layer and provides sophisticated admission control, scheduling and overload management. In this paper, we provide the overview of the existing work done on real-time transaction control layer called MOA (multiclass overload architecture), in the database system architecture. MOA categorized transaction workload into three importance classes as high, medium and low in order to guarantee predictability, overload resolution and service differentiation. Next, we discuss about the various analytical model for real-time transactions which makes the transactions with higher importance have small miss deadline percentage. In addition, the system architecture for real-time database system (RTDBS) has been discussed to achieve a significant performance even in overload situation.

Keywords: Real-Time Database System, Overload Management, Transaction Scheduling.

I. Introduction

As the volume of information being handeled by data-intensive applications faced with timing requirements increasing day-by-day, there is a need to apply database technology to real-time systems. The design objective of conventional databases do not support timing and temporal requirements, and therefore they are not appropriate for real-time applications. A high-performance database which is simply fast and do not have the capability of specifying and enforcing time constraints are also not suitable for real-time applications.

A real-time system must include ability to meet the time constraints which has basic specification and design correctness arguments and that correctness depends not only on the logical result of a computation, but also on the timeliness of its actions. A database system which supports a real-time application can be called 'a real-time database system (RTDBS)'. RTDBS is a database system where at least some transactions have explicit timing constraints (such as deadline). It stores data whose operations execute with real-time response to the transactions of data-intensive applications, such as e-commerce applications, stock market, banking, internet bids, and control systems.

Although real-time transaction processing is complex because in addition to satisfying database consistency requirement, as in traditional database systems, RTDBS timing constraints are an integral part of the correctness criterion. The serializability is a criterion for correctness of concurrent transaction execution. The principles and techniques of transaction management in Database Management Systems need to be applied to real-time applications for efficient storage and manipulation of information. Furthermore, the data in such database has application-acceptable levels of logical and temporal consistency of data, which is able to handle the transactions with the ACID properties: atomicity, consistency, isolation and durability. RTDBS is the more efficient way of handling large amounts of data which gather database from the environment, process it in the context of information acquired in the past to provide timely and temporally correct response.

The scheduling of operations in a RTDBS based on two components: time-critical scheduling and concurrency control. To guarantee the isolation and atomicity properties, it requires the design of real-time concurrency and commit protocols. Traditionally concurrency control real-time protocols can be broadly classified as either pessimistic (or locking) or optimistic (or validation). Pessimistic protocols detect conflicts as soon as they occur and resolve them using blocking that may result in future inconsistencies are detected. Two Phase Locking (2PL) [1] is the most common pessimistic concurrency control protocol. Optimistic protocols such as SCC [2], [3], WAIT-50 [4] detect conflicts at transaction commit time and resolve them using rollbacks (restarts). Examples of real-time commit protocols are OPT [5] and PROMPT [6]. For a system configuration, the primary real-time performance determinants are the transaction scheduling policies for the system resources. Scheduling policies such as EDF [7], EDF-CR [8], [9] are used to assign transaction priority and it helps to schedule transactions within the scheduling queue.

Although all of these feature, the goals of conventional DBMS and real-time DBMS are different. Moreover, real-time DBMS performance objectives differ from those of conventional database system in that maximizing the number of transactions that complete before their deadlines becomes the decisive performance objective, rather than merely maximizing concurrency (or throughput).

II. RELATED WORK

In 1991, J.R. Haritsa et al. [10] have proposed a new priority assignment policy called Adaptive Earliest Deadline (AED). It stabilizes the overload performance of earliest deadline in RTDBS environment. It features a feedback control mechanism that detects overload conditions modifies transaction priority assignments accordingly. They have evaluated the Hierarchical Earliest Deadline (HED) is the extension of AED policy and was designed to handle applications where transactions may be assigned different values. They have shown the results that, both for workloads with limited spread in transaction values and for workloads with pronounced skew in transaction values, the HED policy provided the best overall performance.

In 1992, Sang H. Son et al. [11] have been proposed a new approach to real-time transaction scheduling in which there is two hybrid real-time concurrency control protocols which combine pessimistic and optimistic approaches to concurrency control in order to control blocking and aborting in a more effective manner. The two-phase locking is termed as being pessimistic and an optimistic approach is a natural alternative which schedules all operations hoping that nothing will go wrong, such as non-serializable execution. They have shown the results that over the entire operational range, optimistic schemes outperform the locking-based pessimistic protocol.

In 1993, Ozgur Ulusoy et al. [12] have designed a Real-Time Transaction Scheduling in Database Systems and concentrated on the concurrency control protocol in RTDBS. The authors have evaluated the performance of the protocols through simulations by using a detailed model of a single-site RTDBS. They have proposed a new concurrency control protocols such as Data Priority-based locking protocol (DP) and Optimistic protocol (OP) under conditions of high transaction load and high data contention with improved performance.

In 1994, Y -K. Kim et al. [13] have proposed a database server for distributed real-time systems. They have chosen the ARTS operating system kernel as the basis for the real-time database server. ARTS-RTDB supports both hard and soft real-time transactions. They have provided a flexible programming interface and standard client template to allow quick prototyping. They have incorporated the notion of imprecise computation into RTDB.

In 1994, Brad Adelberg et al. [14] have emulated soft real-time scheduling using traditional operating system schedulers. This paper focuses on soft real-time applications. The authors have addressed methods of emulating real-time scheduling algorithms on top of standard time-share schedulers and developed three strategies for priority assignment to emulate EDF and Least Slack First scheduling within a traditional multi-tasking environment. They have shown the result that the emulation algorithms are comparable in performance to the real-time algorithms and in some instances outperform them. On the other hand, in 1995, Brad Adelberg et al. [15] have applied updated streams in a soft real-time database system. In this paper, four different algorithms are used to schedule both applying updates to imported views and running user transactions perform under different assumptions about data freshness. The authors have discussed about the various properties of updates and views (including staleness) that affect the tradeoff between transaction response time and data freshness. The results have shown that the system performance was affected by the choice of algorithm and and system properties. Whereas in 1996, Brad Alderberg et al. [16] have proposed database support for efficiently maintaining derived data. The authors have propsed the forced delay recomputation algorithm. They have shown that forced delay greatly reduced the recomputation cost while only modestly diminished data timeliness across a wide range of parameter values. It can exploit update locality to improve both data freshness and transaction response time.

In 1996, B. Adelberg et al. [17] have proposed the Stanford Real-time Information Processor (STRIP) which is a soft real-time database system and was built for the UNIX operating system. It is a database designed for heterogeneous environments and provides support for value function scheduling and for temporal constraints on data. Its goals include high performance and ability to share data in open systems. It does not support any notion of performance guarantees or hard real-time constraints and hence cannot be used for the applications we are envisioning in our work. Whereas in 1996, S. F. Andler et al. [18] have proposed Active Real-Time Database System (DeeDS) which is distributed and supports both hard and soft transactions. It is an event-triggered real-time database system using dynamic scheduling of sets of transactions. The reactive behavior is modeled using (event-condition-action) ECA rules. In the current prototype, they do not support temporal constraints of data and multimedia information.

In 1996, J. Taina et al. [19] have a RODAIN, a real-time object-oriented database system for telecommunications. It supports firm real-time database system. In 1996, A. Datta et al. [20] have proposed a multiclass transaction scheduling and overload management in firm real-time database systems. The authors have introduced a dynamic admission control policy and priority based scheduling policy for disk resident RTDBS called Adaptive Access Parameter (AAP) which is a scheduling mechanism for multiclass transactions in RTDBS. The admission control policy of AAP serves dual purposes – overload management as well as bias control towards particular transaction classes. And it leads to dramatic performance improvement both in terms of reducing transaction misses as well as fairness. Whereas in 1996, A. Bestavros et al. [21] have proposed an admission control paradigm for value-cognizant Real-Time Database in which a transaction is submitted to the system as a pair of processes: a primary task and a compensating task. The authors have considered only hard-deadline transactions. The goal of the admission control and scheduling protocol employed in the system is to maximize system profit dynamically.

In 1999, J. A. Stankovic et al. [22] have proposed BeeHive: Global Multimedia Database Support for dependable, real-time applications. It is a virtual database where the data in the database can be located in multiple locations. It includes features along real-time interface, fault tolerance interface, quality of service for audio and video, and security dimensions. The authors have achieved a high degree in the usability of a virtual database system where a user can obtain secure and timely access to time valid data even in the presence of faults.

In 2002, K. -D. Kang et al. [23] have proposed Service Differentiation in Real-Time Main Memory Database to execute transactions in temporally consistent data. Based on the importance of the transactions, they have been classified into several services. Different degree of deadline miss ratio and feedback control has been applied among the service classes to support the miss ratio and freshness guarantees. They have shown the results that the approach can provide the specified QoS when the baseline approaches fail to support the miss ratio and/or freshness guarantees in the presence of unpredictable workloads and access patterns. The target performance is achieved by dynamically adapting the system behaviour based on the current performance error measured by the monitor.

In 2006, M. Amirijoo et al. [24] have proposed a framework for QoS specification and management. It has been developed in real-time databases supporting imprecise computation. The architecture based on feedback control scheduling, and a set of algorithms implementing different policies and behaviors. The approach gives a robust and controlled behavior of real-time databases, even for transient overloads and with inaccurate runtime estimates of the transactions. Further, performance experiments show that the proposed algorithms outperform a set of baseline algorithms, where transactions are scheduled with EDF and feedback control.

In 2006, Leila Baccouche [25] has proposed Multi-Class Overload Architecture (MOA) for Real-time Database Systems: Framework and Algorithms. The architecture includes a real-time control layer which provides sophisticated admission control, scheduling and overload management. They have considered an importance classes that have been categorized as high, medium and low level of transaction workload. They have evaluated the H/M/L dispatching algorithm with the developed simulator and they have shown the results in terms of percentage. The proposed algorithm has made transactions with higher importance have small miss deadline percentage.

In 2012, R. K. Mishra et al. [26] have proposed a novel protocol RCCOS (Replica Concurrency-Control for Overloaded Systems) for overload management and admission control in real-time distributed database systems. It can be easily integrated in current systems to handle overload processors without altering the database consistency which is the main objective of DRTDBSs. Our protocol RCCOS (Replica Concurrency-Control for Overloaded Systems) augments the protocol MIRROR, a concurrency control protocol designed for firm-deadline applications operating on replicated real-time databases in order to manage efficiently transactions when the distributed system is overloaded.

III. ANALYTICAL MODEL FOR REAL-TIME TRANSACTIONS

Real-Time transactions can be distinguished based on the effect of missing their deadlines. They can be grouped into three categories: transactions that have hard deadlines, soft deadlines and firm deadlines.

A. Hard Deadline Transactions

A hard deadline transaction has hard timing constraints that must absolutely be met. The transactions are those which may result in a catastrophe if their deadlines are missed. There is no scope to miss its deadline in hard deadline transactions. Missing a hard-deadline can result in catastrophic consequences. Such systems are known as safety-critical. And we can view such tardy transactions as carrying a large negative value to the system. Typically safety-critical applications (e.g. nuclear power plant control) that respond to life or environment-threatening emergency situations can be classified in this category.

B. Soft Deadline Transactions

A soft real-time application is characterized by a soft deadline whose adherence is desirable, although not critical, for the functioning of the system. Soft deadline transactions have some diminished value to the system even if they complete after their deadlines have expired. Such systems tolerate imprecise results and less quality of service. There is scope to miss its deadline in soft deadline transactions which may lead to performance degradation but do not entail catastrophic results.

C. Firm Deadline Transactions

A firm real-time task, like a soft real-time task, is characterized by a firm deadline whose adherence is desirable, although not critical, for the functioning of the system. Firm deadline transactions, have no value once their deadlines expire, completing them is of no utility and may even be harmful to the system. Thus, tardy transactions are permanently aborted (killed) and discarded as soon as its deadline is missed. Firm deadlines should be met but may be missed occasionally (e.g. during transient overloads).

Real-time database transactions are usually in either firm deadline or soft deadline class. In addition to these timing constraints, as there is a need to maintain consistency between the actual state of the environment and the state as reflected by the contents of the database, it needs to extend the timing correctness requirements in a RTDBS. This leads to the temporal consistency and its two components: absolute consistency between the state of the environment and its reflection in the database, and relative consistency among the reflected data used to derive other data.

The transaction temporal consistency requirements are required for real-time transactions. A transaction τ_i is characterized by the following attributes:

TABLE I

Attributes of Transaction τ_i

Attributes	Meaning	Description
r_i	Ready time	The time in which the transaction arrives to the system.
d_i	Relative deadline	It indicates requirements to complete transaction before the instant *di*.
we_i	Worst case execution time	The execution time of a transaction is data dependant.
re_i	Remaining execution time	It represents the time to complete the transaction.
st_i	Slack time of τ_i	It represents maximum time of transaction can be delayed and still satisfy its deadline. $st_i = d_i - r_i - we_i$.
imp_i	Importance of τ_i	It indicates how much it is critical to the system that the transaction meets its deadline.

Transaction type can have two values local or distributed. A local transaction is created by the local node. A distributed transaction is a sub transaction initiated by a distant node and sent to the current node to be executed.

Transaction can be either periodic or aperiodic. Periodic transactions tend to have *hard* deadlines, characterized by their *period*(*s*) and their required execution time per period. They need to update data frequently. Let *pi* be the invocation period. A periodic transaction is executed each *pi* instants. Usually *pi=di.* Aperiodic transactions tend to have soft deadlines, characterized by unknown arrival instants.

IV. SYSTEM ARCHITECTURE

The system architecture is the Multi-class Overload Architecture (MOA) which allows executing transactions under overload. The transaction deadline is considered as firm in this architecture. Features of the system architecture are:

- It supports concurrent execution of multiclass transactions.
- It reduces miss deadline, function of transaction classes.
- Overload situations detection and resolution.

MOA is a real-time control layer that should be at end integrated to the database system. A multi-class transaction workload consists of high, medium and low importance classes. The system for overload management consists of three components: a transaction controller, a transaction scheduler and a transaction manager. Figure 1 presents the system architecture of MOA.

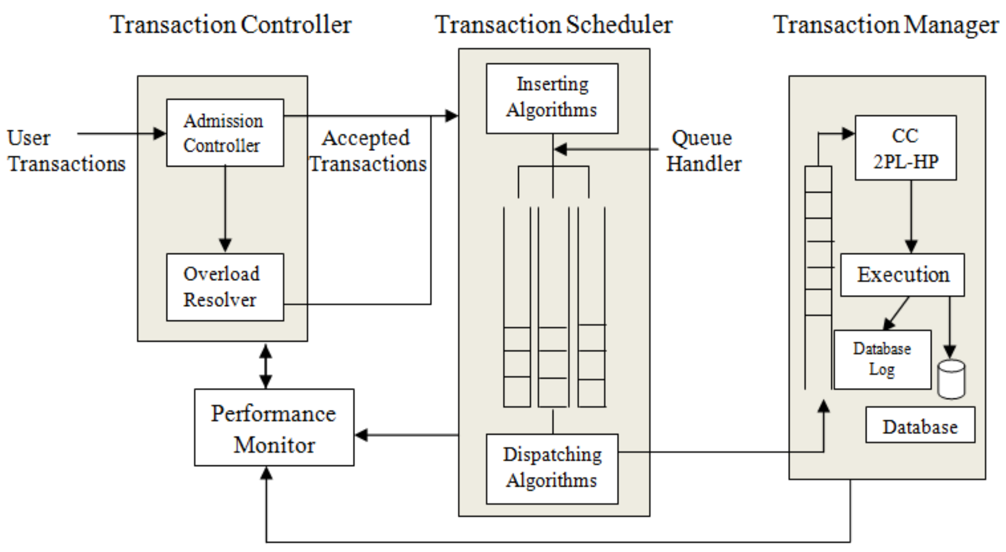


Fig 1. System Architecture

A. The Transaction Controller

The transaction controller requires the admission controller to test the admission of transactions and the overload resolver to solve overload situations.

1) *The Admission Controller:* When a transaction is submitted to the system, an admission controller is employed to decide whether to admit or reject that transaction. The admission controller controls the admission of new transactions and tests the acceptability upon their arrival. The controller makes a decision which depends on the state of the current system and attributes of transaction. The schedule is not computed at that time, the controller simply checks when an acceptance condition is satisfied. If the test succeeds, this means that the transaction scheduler will be able to find a feasible schedule including the new transaction guarantying that each transaction meets its deadline.

2) *The Overload Resolver:* If two transactions conflict then the lower priority transaction has a larger risk of being aborted by the higher priority transactions. Similarly, a transaction with a loose deadline has a larger risk of being aborted by a transaction with a tight deadline. The system does not execute any test for low importance transactions. It prefers to devote time processor to another more significant task. Depending on the characteristics of the transaction, the overload resolver component may be invoked in order to find a solution where the rejected transaction by the admission controller is accepted.

B. The Transaction Scheduler

The transaction scheduler requires inserting and dispatching algorithm. A 3-level priority scheme is used to schedule transactions. To address this need, a priority based scheduling algorithm called H/M/L is applied by the dispatcher which is specific to the transaction multi-class model.

1) *Inserting Algorithms:* Inserting algorithms are the most famous dynamic real-time scheduling algorithms: Earliest Deadline First (EDF) and Least Laxity First (LLF). The transactions accepted by the admission controller are sent to the inserting module. Accepted transactions are inserted in the associated queues according to their priorities. The queues are sorted by an increasing value of the deadlines. This way, the transactions are sorted based on the EDF algorithm which is known to be optimal. The queues are sorted by an increasing value of the transactions slack times in LLF.

2) *Dispatching Algorithms:* Most famous service algorithms are First Come First Serve (FCFS), Weight Fair Queue (WFQ) and Weighted Round Robin (WRR) [27], but they are not appropriate for multiple queues with priority scheduling under real-time constraints. The dispatching algorithms are used to extract transaction from three queues. The transactions scheduler extracts transactions from the HI_queue, MI_queue and LI_queue and inserts them in the TM_queue, a queue used by the transaction manager to execute transactions. Dispatching algorithms adapted to the multi-class transaction model. In relation to the multi-class nature of the transactions (high, medium and low importance) requires more sophisticated admission control and dispatching algorithm.

3) *H/M/L Algorithm:* The H/M/L is a priority based scheduling algorithm which is applicable to the multi-class nature of the transaction model. The H/M/L algorithm extracts H percent of the number of high importance transactions, M percent of the number of medium importance transactions and L percent of the low importance transactions that are ready (i.e. their ready time is reached). No one test is taken by the system for low importance transaction. Therefore, a threshold is defined for the execution of low importance transactions.

Each time the system increments a counter as a new transaction is inserted in the appropriate queue. The threshold is defined on a minimal number of transactions (min_T) in the HI_queue or the MI_queue. The algorithm computes the total number of transactions within each queue and compares it to a threshold min_T. When min_T is reached, the process generates a stop signal to interrupt the execution of low importance transactions. While executing transactions of low importance, if a signal stop is caught, the algorithm interrupts its execution within the LI_queue and returns to the queue responsible of the signal i.e. the queue of which number of transactions reached the threshold.

C. *The Transaction Manager*

The transaction manager models the execution of transactions. It implements the database engine. It is responsible for transaction execution, consistency check, concurrence control and native transaction logging operations. The specification of real-time systems includes the functionalities of real-time constraints that can have a significant impact. Therefore, the consistency check is discarded.

C is considered as the system capacity to execute transactions in parallel. For each transaction extracted from TM_queue, the resource set is checked and once all the required resources allowed, the transaction begins its execution. When a transaction resumes execution, if C is not reached, the transaction manager extracts the front head transaction from the TM_queue and so on. This way, the transaction manager extracts and launches C transactions. The transaction manager applies 2PL-HP (Two phase lock high priority) concurrency control (CC) algorithm which is free from inversion priority.

V. CONCLUSION

In this paper, we have discussed the multi-class overload architecture (MOA) for overload management in real-time database system. Also we have discussed the different transactional model for real-time transactions. MOA mainly consists of set of modules which act together in order to guarantee real-time properties of transactions that is predictability, overload resolution and service differentiation. The transaction scheduler applies dispatching algorithm called H/M/L. As discussed in this paper, H/M/L algorithm is used to schedule transaction in real-time database systems. It is suitable for multi-class scheduling and is starvation free. In the future, algorithm H/M/L will apply percentages of extraction which will be calculated on line and will be dependent on the arrivals of transactions. Future work is to improve the parameters considered.

References

1. J.R Haritsa, M. J. Carey, and M. Livny, 1992. "Data access scheduling in firm real-time database systems", In Journal of RTS, Volume 4, Issue 3, pp. 203-241.

2. A. Bestavros, and S. Braoudakis, 1996. "Timeliness via speculation for real-time databases", In Proceedings 14th IEEE RTS Symposium (RTSS'96), San Juan, Puerto Rico.

3. S. Braoudakis, 1995. "Concurrency control protocols for real-time databases", PhD dissertation, Computer Science Department, Boston University, USA.

4. J.R Haritsa, M. J. Carey, and M. Livny, Dec. 1990. "Dynamic Real-Time Optimistic Concurrency Control", In Proceedings of the Real-Time Systems Symposium, pp. 94-103.

5. R. Gupta, J. Haritsa, K. Ramamritham, and S. Seshadri, Dec. 1996. "Commit processing in distributed real-time database systems", In Proceedings of the 17th IEEE Real-Time Systems Symposium, pp. 220-229, Washington D.C.

6. R. Jayant, J. Haritsa, K. Ramamritham and R. Gupta, Feb. 2000. "The PROMPT Real_Time Commit Protocol", IEEE Transactions on Parallel and Distributed Systems, Volume 11, Issue 2, pp. 160-181.

7. C. L. Liu, and J. Layland, Jan. 1973. "Scheduling algorithms for multiprogramming in hard real-time environments", In Journal of the ACM, 20(1): pp. 46-61.

8. R. Abbot, and H. Garcia-Molina, Aug. 1988. "Scheduling Real-Time Transactions: A Performance Evaluation", In Proceedings of the 14th International Conference on Very Large Database Systems, Los Angeles, California, USA.

9. R. Abbot, and H. Garcia-Molina, Aug. 1989. "Scheduling Real-Time Transactions with Disk Resident Data", In Proceedings of the 15th International Conference on Very Large Database systems, pp. 385–396, Amsterdam, The Netherlands.

10. J.R Haritsa, M. J. Carey, and M. Livny, Dec. 1991. "Earliest-Deadline Scheduling for Real-Time Database Systems", In Proceedings of the 12th IEEE Real-Time Systems Symposium, San Antonio, Texas.

11. Sang H. Son and Juhnyoung Lee, Jun. 1992. “A New Approach to Real-Time Transaction Scheduling”, In Proceedings of the IEEE Conference on Real-Time Systems, pp. 177-182.

12. Ozgur Ulusoy and Geneva G. Belford, Dec 1993. “Real-Time Transaction Scheduling in Database System”, In Proceedings of the ACM Journal on Information Systems, Volume 18, Issue 9, pp. 559-580.

13. Y-K. Kim, M. R. Lehr, D. W. George, and S. H. Song, 1994. "A database server for distributed real-time systems: Issues and experiences", In Proceedings of the Second IEEE Workshop on Parallel and Distributed Real-Time Systems.

14. B. Adelberg, H. Garcia-Molina, and B. Kao, April 1994. “Emulating Soft Real-Time Scheduling Using Traditional Operating System Schedulers”, In IEEE Real-Time Systems Symposium.

15. B. Adelberg, H. Garcia-Molina, and B. Kao, 1995. “Applying Update Streams in a Soft Real-Time Database System”, In Proceedings of the ACM Sigmod Record.

16. B. Adelberg, B. Kao and H. Garcia-Molina, 1996. “Database Support for Efficiently Maintaining Derived Data”, In EDBT Proceedings.

17. B. Adelberg, B. Kao, and H. Garcia-Molina, Mar 1996. "Overview of the STanford Real-Time Information Processor (STRIP)", In SIGMOD Record (ACM Special Interest Group on Management of Data), Volume 25, Issue 1, pp. 34-37.

18. S. F. Andler, J. Hansson, J. Eriksson, J. Mellin, M. Berndtsson, and B. Eftring, Mar 1996. "DeeDS Towards a Distributed and Active Real-Time Database System", In Proceedings of the ACM Sigmod Record, Volume 25, Issue 1, pp. 38-51.

19. J. Taina, and K. Raatikainen, 1996. "RODAIN: A Real-Time Object-Oriented Database System for Telecommunications", In Proceedings of the DART'96 workshop, pp. 12–15.

20. A. Datta, S. Mukherjee, I. Viguier, and A. Bajaj, Mar 1996. “Multiclass Transaction Scheduling and Overload Management in Firm Real-Time Database System”, In Proceedings of the ACM Journal on Information Systems, Volume 21, Issue 1, pp. 29 – 54.

21. A. Bestavros, and S. Nagy, Jan 1996. "An admission control paradigm for value cognizant real-time databases", In the Proceedings of ACM on RTSS, Proceedings of the 17th IEEE Real-Time Systems Symposium, pp. 230-239.

22. J. A. Stankovic, S. H. Son, and J. Liebeherr, 1999. "BeeHive: Global Multimedia Database Support for Dependable, Real-Time Applications", Springer Link on Active, Real-Time and Temporal Database system, Volume 1553, pp. 51-69.

23. K.-D. Kang, S. Son, and J. Stankovic, 2002. "Service Differentiation in Real-Time Main Memory Databases", In the Proceedings of 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 119-128.

24. M. Amirijoo, J. Hansson, and S. Song, Mar 2006. "Specification and Management of QoS in Imprecise Real-Time databases", In Proceedings of the IEEE Transactions on Computers, Volume 55, Issue 3, pp. 304-319.

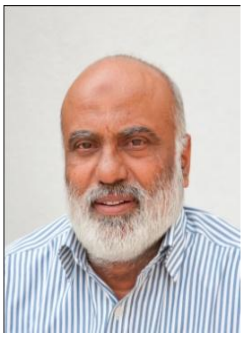
25. Leila Baccouche, 2006. "An Overview of MOA, a Multi-Class Overload Architecture for Real-time Database Systems: Framework and Algorithms", In Proceedings of the IEEE Conference on Computer Systems and Applications, pp. 756-763.

26. R. K. Mishra and Dr. U. Shanker, June 2012. "Overload Management and Admission Control in Real Time Distributed Database Systems", IJCST, Vol. 3, Issue 2, pp. 29-34.

27. C. Semeria, Dec. 2001. "Supporting Differentiated service classes: queue scheduling disciplines", white paper, http://www.juniper.net/solutions/literature/white_papers/200020.pdf.

AUTHOR(S) PROFILE

Miss. Rutuja. A. Gulhane has pursuing of Master of Engineering in Computer Science and Engineering at Sant Gadge Baba Amravati University, Amravati, Maharashtra, India. She had pursued her Bachelor of Engineering degree in Computer Science and Engineering from Sipna College of Engineering, Amravati University in 2012.



Dr. M. S. Ali completed his B.E. (Electronics & Power) from Govt. College of Engineering, Amravati, and Nagpur University in 1981 with First Division. He completed his Masters' degree M.Tech. in Power Electronics from I.I.T. Powai, Mumbai in the year 1984. He obtained his Ph.D. in Electronics Engineering from Sant Gadge Baba Amravati University, Amravati in the year 2006. His main areas of interest are Operating Systems, Intelligent Systems and Java Technology. He is a recognized Ph.D. guide in Computer Science & Engineering in the Faculty of Engineering & Technology at SGB Amravati University. He is the founder Principal at Prof Ram Meghe College of Engineering & Management, Badnera-Amravati since 2009.