DeeDS Towards a Distributed and Active Real-Time Database System[*]

S.F. Andler, J. Hansson, J. Eriksson, J. Mellin, M. Berndtsson and B. Eftring
Department of Computer Science, University of Skövde, Sweden
email: {sten, jorgen, joakim, jonas, spiff, bengt}@ida.his.se

1 Introduction

There is a great need for building research prototypes of real-time database systems, because existing techniques are often *ad hoc* or concealed in internal solutions, and few, if any, commercial offerings exist. As real-time applications inherently use reactive mechanisms, and a wealth of results now exists on active database management systems [5], we decided to build DeeDS, a [D]istributed Activ[e], Real-Tim[e] [D]atabase [S]ystem prototype. This research effort is challenging due to the unique combination of reactive mechanisms, distributed technology, dynamic scheduling, and integrated monitoring, under a mixture of soft and hard real-time constraints. To make all this feasible with a limited amount of man-power, we have taken a deliberate approach of building with existing components where possible, and including only those features absolutely necessary to obtain the benefit of each of the architectural dimensions. At the same time, features and implementation methods have been chosen to best promote predictability and efficiency in such a complex system, while preserving flexibility for further research in this area [2, 1].

2 DeeDS

2.1 Motivation

When studying complex real-time systems we found that they often require distribution and sophisticated sharing of extensive amounts of data, with full or partial replication of the database. For example, in integrated vehicle systems control we have identified a need for autonomous nodes controlling individual subsystems and handling large amounts of data locally under hard timing constraints (e.g., fuel injection, ignition control), while at the same time requiring parameters from other subsystems on a much less critical time scale (e.g., from the transmission, environment sensors). Similar features have been identified in automated manufacturing and time-constrained distributed naming services.

When building distributed databases, sources of unpredictability are introduced due to network communication and distributed commit processing, as well as disk accesses. Hence, in order to guarantee the timeliness and predictability of the system, the sources of unpredictability must either be eliminated, or the software must become time-cognizant. By placing the database in main-memory, unpredictable disk access delays are avoided [14]. Further, in order to guarantee local data availability and eliminate network communication, the database will be (virtually) fully replicated. Finally, in the studied application scenarios, temporary inconsistencies of the distributed database can be allowed. Hence, updates are made locally and propagated to all nodes in a way that guarantees eventual consistency [6].

Transactions are either periodic or sporadic, where the latter are executed in response to an event or situation. In DeeDS, this reactive behavior is modeled with *event-condition-action* rules (ECA) [11], which allow for efficient situation monitoring, and database constraint enforcing [27].

2.2 Technical Overview

The DeeDS architecture [1] separates application-related functions from critical system services. The former consist of a rule manager module, an object store (OBST [10]), and a storage manager (tdbm) [7]. The latter include scheduling [4], and event monitoring. The underlying distributed real-time operating system kernel is OSE Delta [25].

The rule manager receives event instances from the event monitor and executes rules that are triggered by the event. If the condition is satisfied the action is presented to the scheduler for execution. Rules are extended with the ability to specify timing constraints on triggered actions.

When an event has been detected, efficient selection of potentially triggered rules is important. In DeeDS, rules are associated with specific events in order to reduce unnecessary rule triggering. Thus we only notify those rules which are specifically interested in the event occurrence [3].

[*]This work was supported by NUTEK (The Swedish National Board for Industrial and Technical Development) and the Ministry of Education and Science in Sweden.

To make rule execution predictable, coupling modes are restricted [9], and cascade triggering is limited, in order to attain an upper bound on the maximum execution time of transactions. Moreover, condition evaluation is limited to logical expressions on events and object parameters, and method invocations.

Transactions are scheduled dynamically, and their timing constraints are expressed as parameterized value functions. Parameterization reduces storage requirements and computational cost, compared to arbitrary value functions. The workload consists of transactions with varying deadline criticality, i.e., critical and non-critical. The scheduling algorithm must, during transient overloads, favor critical transactions over non-critical transactions in order to maintain timeliness. Early work on real-time scheduling of multi-level transactions has been carried out in [18, 21]. In [26], priority assignment policies for active real-time databases have been studied and evaluated.

Each critical transaction will have a contingency plan with reduced computational requirements, which is invoked when the original transaction cannot meet its deadline. This could be in response to a transient overload detected by the scheduler. In [17], we defined a scheduling model for a deadline-driven scheduler for OSE Delta.

The critical system services execute on a dedicated processor to simplify overhead cost models, manage event criticality [4], avoid probe-effect [28], and increase concurrency in the system. Moreover, the scheduler continuously monitors the remaining execution time of each transaction using milestones [19].

We thus make use of hybrid monitoring techniques, based on both software and hardware monitoring, for predictable and efficient event monitoring [16]. The interaction between the scheduler and the event monitor is performed during fixed intervals, and can either be *synchronous* (time-triggered), or *asynchronous* (event-triggered). Preliminary results show that the synchronous interaction style results in higher throughput but longer unavoidable minimum delay than the asynchronous interaction style if an incremental scheduler is used [24].

The event monitor detects primitive and composite events (using event graphs [12]) and disseminates the corresponding event instances to subscribing recipients [23], e.g., the rule manager. Predictability is imposed by limiting event constructors [11, 9] and contexts [12]. Incomplete event instances are flushed when they are invalid, e.g., when the transaction, in which their constituents were generated, aborts or commits [9].

3 Related Work

Few research projects on active time-constrained databases have been presented [13]. One of them is HiPAC [11], in which the idea of using production rules as the paradigm for rule specification is proposed, and different coupling modes for condition evaluation and triggered actions are identified. No implementation of the complete system is however made, albeit a strawman architecture has been presented.

In REACH [9], the impact on timeliness of using different coupling modes is investigated, as well as the use of contingency actions. The event monitoring in REACH is distributed in the system, which is possible, but currently not exploited, in DeeDS. REACH uses milestones [8] to detect whether a transaction will meet its deadline. In DeeDS, the use of milestones is extended to detect that a transaction will finish earlier than calculated, and thereby leaves slack time to be used by other transactions. REACH is disk-based, whereas DeeDS is main-memory resident. It should be noted that neither REACH nor HiPAC is distributed.

ARTS-RTDB [22] is a relational, distributed real-time database system. While database nodes in DeeDS are fully replicated and communicate asynchronously, ARTS-RTDB consists of one or more database servers which communicate with clients using synchronous message passing. Moreover, ARTS-RTDB does not include reactive mechanisms.

The idea of using dedicated processors for application processes and scheduling processes, however not necessarily in a database system, was first adopted in the Spring-project [29], in which they focused on giving on-line guarantees for meeting task deadlines. Distributed hybrid monitoring has been investigated in the INCAS project [16], but DeeDS differs by supporting composite events and, currently, only allows local event detection.

4 Conclusions

The DeeDS prototype is currently being implemented for the OSE Delta kernel in a UNIX development environment. We have ported tdbm [20] and extended it for the preemptive environment, and we are in the process of porting OBST and integrating the two in cooperation with OBST developers at FZI. Trial implementations have been made of lazy replication algorithms [15] and simulation results have been obtained for several scheduling techniques.

The DeeDS prototype represents a unique integration of several advanced concepts such as active functionality, distribution, and real-time database systems with hard and soft deadlines. We believe that the key concepts are the use of lazy replication and main memory residency to avoid unpredictable delays, contingency plans to guarantee schedulability, and event criticality and dual processor architecture to make event monitoring and scheduling predictable.

References

[1] S. Andler, M. Berndtsson, B. Eftring J. Eriksson, J. Hansson, and J. Mellin. DeeDS - distributed active real-time database system. Tech. Rep. HS-IDA-TR-95-008, Dept. of Comp. Sci., U. of Skövde, 1995.

[2] S. Andler, J. Hansson, J. Eriksson, and J. Mellin. The distributed reconfigurable real-time database systems project. Tech. Rep. HS-IDA-TR-94-006, Dept. of Comp. Sci., U. of Skövde, Sep 1994.

[3] M. Berndtsson. Reactive Object-Oriented Databases and CIM. In *Proc. 5th Int'l Conf. on Database and Expert System Applications, Athens, Greece*, pp 769–778, Sep 1994.

[4] M. Berndtsson and J. Hansson. Issues in active real-time databases. [5], pp 142–157.

[5] M. Berndtsson and J. Hansson, eds. *Proc. 1st Int'l Workshop on Active and Real-Time Database Systems (ARTDB-95), Skövde, Sweden*, Workshops in Computing. Springer Verlag (London), Jun 1995.

[6] A.D. Birrell et al. Grapevine: An exercise in distributed computing. *Comm. of the ACM*, 25(4):260–274, Apr 1982.

[7] B. Brachman and G. Neufeld. TDBM: A DBM library with atomic transactions. Tech. rep., Dept. of Comp. Sci., U. of British Columbia, 1992.

[8] H. Branding et al. Rules in an open system: The REACH rule system. In N.W. Paton and M.H. Williams, eds, *Rules in Database Systems, Edinburgh*, pp 111–126. Springer-Verlag, 1993.

[9] A. P. Buchmann et al. Building an integrated active OODBMS: Requirements, architecture, and design decisions. Data Engineering, 1995.

[10] E. Casais et al. OBST—an overview. Tech. Rep. FZI.039.1, FZI, Karlsruhe, Germany, 1992.

[11] S. Chakravarthy et al. HiPAC: A research project in active time-constained database management. Final technical report. Tech. Rep. XAIT-89-02, Xerox Adv. Info. Tech., Aug 1989.

[12] S. Chakravarthy and D. Mishra. Snoop: An event specification language for active databases. *Knowledge and Data Engr.*, 13(3), Oct 1994.

[13] J. Eriksson. Real-time and active databases: A survey. Dept. of Comp. Sci. U. of Skövde. In preparation.

[14] H. Garcia-Molina and K. Salem. Main memory database systems: An overview. *IEEE Trans. on Knowledge and Data Engr.*, 4:6:509–516, Feb 1992.

[15] P.M. Gustavsson. How to get predictable updates using lazy replication in a distributed real-time database system. Master's thesis, Dept. of Comp. Sci., U. of Skövde, Sep 1995.

[16] D. Haban and D. Wybranietz. A hybrid monitor for behavior and performance analysis of distributed systems. *IEEE Trans. on Softw. Engr.*, 16(2):197–211, Feb 1990.

[17] J. Hansson. Dynamic real-time scheduling in OSE Delta. Tech. Rep. HS-IDA-TR-94-007, Dept. Comp. Sci., U. of Skövde, Aug 1994.

[18] J. Hansson. Dynamic real-time transaction scheduling with multiple combined performance metrics. Tech. Rep. HS-IDA-TR-94-005, Dept. of Comp. Sci., U. of Skövde, Jun 1994.

[19] F. Jahanian et al. Runtime monitoring of timing constraints in distributed real-time systems. *Real-Time Systems*, 7(3), Nov 1994.

[20] M. Johansson. A storage manager for an experimental distributed real-time database platform. Master's thesis, Dept. of Comp. Sci., U. of Skövde, Sep 1994.

[21] Y. Kim and S.H. Son. *Predictability and Consistency in Real-Time Database Systems*, ch. 21, pp 509–531. Prentice Hall, 1995.

[22] Y-K Kim et al. A database server for distributed real-time systems: Issues and experiences. Tech. rep., Dept. of Comp. Sci., U. of Virginia, 1994.

[23] J. Mellin. Survey of event monitoring in distributed real-time systems. Dept. of Computer Science. University of Skövde. In preparation.

[24] J. Mellin, J. Hansson, and S.F. Andler. Deriving design constraints from a system services model for a real-time DBMS. Tech. Rep. HS-IDA-TR-95-010, Dept. of Comp. Sci., Dec 1995. Subm. for publ.

[25] OSE Delta soft kernel R1.0 getting started & user's guide. ENEA DATA, Sweden, 1995.

[26] B. Purimetla et al. Priority assignment in real-time active databases. Tech. rep., Dept. of Comp. Sci., U. of Massachussets, 1994.

[27] K. Ramamritham. Real-time databases. In *Conf. Proc. Distributed and Parallel Databases*, pp 199–226, Boston, 1993. Kluwer Academic Publishers.

[28] W. Schütz. Fundamental issues in testing distributed real-time systems. *Real-Time Systems*, 7(2):129–157, Sep 1994.

[29] J.A. Stankovic and K. Ramamritham. The Spring kernel: A new paradigm for real-time systems. *IEEE Softw.*, 8(3), May 1991.