Databases in Control and Simulation

Esa Falkenroth, Peter Loborg, Anders Törne
Department of Computer and Information Science,
Linköping University,
S-582 60 Linköping, Sweden
{esafa, andto, petlo}@ida.liu.se

Abstract

This position paper describes the real-time database research at the Real-Time Systems Laboratory, Linköping University. The current research involves the application of database technology to the control and simulation of physical processes and has two main directions: adequate activity languages to describe high level operations on objects in the environment or in the simulation, and mechanisms to embed databases into control and simulation systems, and adequate data structures.

1 Introduction

Our work focuses on the application of databases in real-time software systems and, more specifically, the application dependent requirements put on the embedded database and the realization of the corresponding database technology to satisfy these requirements. In this position paper, we describe our approach to embedding real-time databases in control systems and simulators. This paper will describe our approach in both aspects. We will also discuss the similarities between control and simulation applications, with respect to the use of databases.

Currently a large research area within real-time databases and real-time systems (see for example [1]) deals with the issue of predictable timely behaviour of the units of operation - transactions and tasks respectively [2][3][4][5][6][7][8]. Most real time system approaches are related to or based on the rate monotonic priority scheduling method [9]. As previously state in [3] - "Scheduling theories for fixed priority scheduling are now sufficiently mature that a genuine engineering approach to the construction of hard real-time systems is possible" - there is sufficient understanding of these issues in the real-time systems community to try them out in realistic industrial settings. Correspondingly, this view of real-time system design has been transformed in the database community to issues of serializability and concurrency control.

The presented work do not focus on the predictable timely behaviour issues of neither real-time systems nor real-time databases. Our approach is instead the combination of the two required technologies. The database technology is used to handle the data management problem and the real-time system methods are used to handle the timing issues.

2 Databases in control applications

We aim at control applications ranging from large process control systems for, e.g. atomic power plants, to highly autonomous missiles with embedded real time software. The applications can operate in highly complex environments that may involve thousands of sensors and/or a highly dynamic environment.

A common software design problem in this type of applications is the handling of the timing requirements derived from the controlled processes in the external environment and control considerations [4][5]. In traditional control theory, computer-based control systems operate at an algorithmic level or at a level close to the external process [10]. At this level the data management problems are usually small. Sharing data between different applications or access to data is simple since the data is local and is normally not used outside the local control algorithm.

Control engineering usually refer to what is called a model of the controlled and the controlling system. When the controlled system gets complex, it becomes difficult to use the usual methods, like linear system identification. In these complex systems we use adaptive methods instead. An adaptive model means that the controlling system is parameterized and the parameters are set by an external feedback loop. The theory for how to generate the parameters is called *adaptive control* [11]. Our approach takes the same view but at the level of control software instead of at the level of model of the controlling system. The database is used as middleware between the program determining the parameters and the control algorithms.

Furthermore, in larger and more complex control systems, the data management and information processing becomes difficult. These difficulties motivate the integration of real-time software and database functionality. Two different research approaches can be discerned:

- Adding database functionality to the existing control systems.
- Developing databases with real-time functionality.

The first approach is used in several commercial manufacturing and process control systems. The practice in industry is to label the data management facilities in the control systems as a 'distributed database' [12]. However, the functionality is more similar to a distributed data object management system without query language, optimizers, and data modelling languages normally found in databases [13]. The data integrity issue in these systems is a well-known problem. Today, the problem is solved by static scheduling, which take care of data dependencies and mutual exclusion. This approach becomes cumbersome for large systems [14]. Clearly, systems like the next generation aircrafts and missiles must build on a different data management strategy [15]. The awareness of the database technology is present - but is met by the performance counter argument.

The second approach is popular in the current real-time database research. The real-time functionality is integrated into the database. Work has been done in the real-time systems community on methods and formalisms for timing analysis to guarantee the timing requirements of hard real time software systems [6][7]. The same principles can also be applied to database transaction management with hard real-time constraints [8]. Independent of the approach taken in the integration, the guarantee of timely behaviour and performance is essential.

2.1 Our Approach to Databases in Control

It is likely that real complex control systems will be structured as a layered system. All software in such a system that requires high performance is constructed using traditional techniques with closed feedback algorithms. The need for a database functionality in this type of software is generally not large. The information needed for cooperative functions, global analysis, high level reactions, and operator presentation, can be filtered out at a level necessary for the functionality, but without disturbing the high performance software. This information can then be processed by a database. The hard timing requirements on the database transactions remain, but the level of timeliness has been shifted to more attractive regions from a database perspective.

Our approach is neither to incorporate database functionality in the real time software system, nor to add real time functionality to a database management system, but to apply database technology selectively within the real-time system and to add on some real time functionality in the database. This provides a flexibility to choose what parts should be implemented in the database and what parts should be implemented as high performance real-time software. It also shifts some of the focus to issues related to the interface between the real-time system and the real-time database.

A main memory database approach is feasible in such a layered system. The database can be used for situation analysis and activity control for the high level functionality, like high level tracking, mission control, and control mode switching. We have presented a unified architecture for this [16] based on earlier work on factory automation control [17][18] and main memory databases [19]. The implementation is described in [20].

The basic elements of the architecture are a control application language for activity modelling, an active object-relational main-memory database for data modelling and situation analysis, a run time scheduler for the activity operations, a communication link to a high performance real time OS, and a database model of the controlled environment which provides the conceptual interface between the real time system and the real time database. The architecture supports distribution to different physical machines.

The present work includes the development of activity languages (process control languages). We are investigating the problem of error recovery, specification of exception handling, and support of fast resynchronization and restart of controllers and controlled system after an error. In manufacturing control, the errors are normally caused by the breakdown of a physical subsystem such as a sensor or actuator. During the repair of the faulty component, the operator must usually change the state of the machine. If the state is changed, then the controller will have an inconsistent view of the controlled system.

The support for fast resynchronization and restart is based on the existence of a model of each device used in the controlled system. At compile time, the program will be analyzed with respect to this model, resulting in a multitude of possible restart points in the program as well as detection of errors of a class normally not found until runtime. If the device model also contains timing information for the services offered, then the timing properties can be analyzed. We are aiming at tools for verifying timing properties at the activity language level.

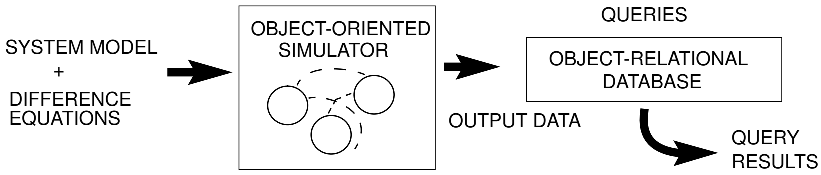

Figure 1: Object-oriented simulator

The long term goal is a design environment for modeling of large real-time systems that allow the designer to explore different alternatives and to verify the timeliness of the design.

In cooperation with ABB Industrial Systems, we are investigating if a layered approach can be used in systems based on emergent standards like IEC-1131. The languages in IEC-1131-3 use a well-defined type system and type checking policy which is a major improvement upon most vendor supplied PLC languages until now. The languages in the standard contain mechanisms for procedural abstraction. Therefore, it is easy to write programs with a good control structure. But the languages lack the necessary mechanisms for encapsulation of data. If data abstraction is used in the applications, it is the responsibility of each programmer to adhere to the abstraction layers.

3 Databases in simulations

One of the main problems with simulations is the laborious pre- and post-processing stages. The preprocessing stage involves translation of a mathematical model to a computational model and a data model. The postprocessing stage involves searching for abnormal or erroneous behaviour in large amounts of output data. The analysis also involves ad hoc queries from the engineer which makes a database approach reasonable.

A simulation model of the control system and the controlled environment will be developed and stored in the object-relational database AMOS[1]. The AMOS database is a research platform for experimenting with specialized database systems. It contains all traditional database facilities such as a data dictionary, a query language, transactions, and database procedures. A foreign function interface allows external simulation routines written in C to be linked to the database. The database also has active rule facilities that can be used to monitor the results of the simulation dynamically.

Mode changes in the simulation will be controlled by events occurring in the discrete model of the simulated system in the database. For example, active rules defined over the discrete model may trigger the mode changes. Simulation and discrete model development tools will be interfaced with the database. The basic idea has been developed in the ARAMIS-project at RTSLAB.

The hypothesis is that the use of databases for modelling, model storage and run-time support has the following benefits:

- It will be possible to query ad hoc about model state during the simulation, for example if the simulation is time consuming or is running in real-time for extended time periods.
- It provides a common data model, i.e. analysis tools and model generators used by the system can easily be exchanged
- The database can be integrated with distributed simulation of different subsystems and provide the interface between these subsystems

To investigate the scope of these benefits, a study has been done involving a simulator for hydraulic circuits developed at the Division of Fluid Power Technology, Linköping University. The simulator is based on transmission line delays. In conventional simulators, all differential equations are solved simultaneously by a centralized solver. To achieve higher performance, the current simulator utilizes transmission line modeling. The limited signal propagation speed in physical systems is used to partition the simulation into weakly coupled subsystems where the differential equations can be solved locally [22]. This makes it possible to simulate complex systems in reasonable time since the computational effort becomes linear with the size of the simulated system. A prototype of a simulator with an embedded database was implemented.

1. Developed at EDSLAB (Prof. T. Risch), Linköping University

4 Combined control and simulation

Simulation of control systems is related to real-time control. In the simulation, both the control system and the controlled system can be simulated to verify the design. The simulation can also interact with real machinery and hardware which requires a database with real-time functionality.

4.1 Interfacing

Our approach to databases in control — i.e., using the database as the abstraction barrier between the high level operations described in an activity language and the low level real-time control - makes it possible to perform so-called hardware in the loop simulations in a flexible way. Parts of the controlled environment can be simulated and parts can be real hardware. The abstraction barrier can either provide the communication/interface to the physical objects or to simulations of the physical objects.

The abstraction barrier is realized by an object oriented data model of the state of the controlled objects. The activity language describes the discrete changes to occur in these objects and the data objects interact with the environment via a servomechanism. From this perspective it is irrelevant if the environment is simulated or consist of actual hardware.

4.2 Indexing techniques

So far the database has been viewed as a model and process data repository. Both for control and simulation, the data is normally in the form of a time sequence — either used as a log or for the possibility to refer to older data in the control functions. A particular investigation has therefore been made of how to access such data.

Our work on indexing techniques considers the storage of time stamped data and the subsequent retrieval of the information, where the speed of insertion of new data is of major concern. Typical applications are data logging and subsequent postmortem analysis in process control systems, alarm and other monitoring functions in run time in process control systems, and the storage of and subsequent access to simulation data. The requirement for fast and timely insert is unavoidable in the generation of simulation data and sensor data in these applications. For this purpose a new high performance indexing techniques is investigated [21].

We have initially identified some fundamental characteristics of simulation and sensing applications:

- Because of the nature of simulation and sensing applications, the insertions of new data typically occur with increasing time stamps.
- The time stamped information comes from several sources and the number of time stamps is several orders of magnitude larger than the number of data sources.
- The time stamps are generally not equidistant — however, locally equidistant sections can exist.
- The volume of stored time stamped data will be large, easily several gigabyte.
- The applications require high performance: 10^4-10^5 updates per second.
- The application will typically operate in two phases: producing the data and analyzing the data. However, these tasks can be interleaved or even concurrent as in interactive simulation or data logging of sensors.
- Deletion of time stamped data will generally not occur during the analysis phase.
- After the data production phase, the stored information very seldom change. The accesses are predominantly read-only.
- The application may want to undo (roll-back) all updates done after a certain point in time.
- Furthermore, in many applications the volume of data will continuously expand. This requires a mechanism to expand the storage structure and a mechanism to dispose, archive, and/or compress the oldest items.
- Database recovery in simulation applications can be achieved by restarting or rerunning the simulation. In control applications, the recovery is more complicated. *If* the control system can restart from a blank state or a state with corrupt data the recovery can be handled. Normally this should be true to satisfy the robustness requirement.

These properties and restrictions are exploited in the proposed storage structure and indexing technique [21] to achieve better performance than existing temporal indexing techniques which do not make these assumptions. The requirement for fast insertion leads to the idea of a simple array-based storage structure. A minimum of housekeeping is done at insert time to speed up the insertion. To solve the problem of fast lookup we propose the use of computational indexes based on super-linear equation solving techniques from the area of numerical analysis.

5 Summary

Current database research at the laboratory has two main directions:

- The development and investigation of activity modelling languages for process control. Currently this work involves error recovery support and the use of the database as a data abstraction interface between high level control and the actual control algorithms.
- The application of database technology to the field of simulation. The issues involved are: storage structures for fast insert, integration with current simulation technology and the use of activity modelling languages.

This means that concurrency control and predictability issues are considered, but not focused. Most of the suggested methods to handle these issues in real-time databases can be integrated in the general architecture of our approach.

6 Acknowledgement

This work is supported by the Center for Industrial Information Technology, Linköping University (CENIIT), and The Swedish National Board for Industrial and Technical Development (NUTEK). We also acknowledge the efforts of Saab Military Aircraft, Saab Dynamics and ABB Industrial Systems in the cooperation projects. Similarly we acknowledge the efforts of EDSLAB, IDA and the division of Hydraulics, IKP at Linköping University.

References

[1] Sang.H Son (ed): *Advances in Real-Time Systems*, Prentice Hall, NJ, 1995.

[2] B. Kao, H. Garcia-Molina: An Overview of Real-Time Database Systems, in Sang.H Son (ed): *Advances in Real-Time Systems*, Prentice Hall, NJ, 1995.

[3] Alan Burns: Preemptive Priority Based Scheduling: An Appropriate Engineering Approach, in Sang.H Son (ed): *Advances in Real-Time Systems*, Prentice Hall, NJ, 1995.

[4] M.H. Graham: Issues in Real-Time Data Management, *J. Real-Time Systems*, 4, 185-202 (1992)

[5] J.A. Stankovic: Misconceptions about Real-time Computing, *IEEE Computer*, **21**, 10, 10-19 (1988).

[6] A. Burns, A.J. Wellings: Engineering a Hard Real-time System: From Theory to Practice, *Software - Practice and Experience*, **25**, 7, 705-726 (July 1995), John Wiley.

[7] D.I. Katcher, H. Arakawa, J.K. Strosnider: Engineering and Analysis of Fixed Priority Schedulers, *IEEE Trans Software Engineering*, 19, 9, 920-934, (1993).

[8] Young-Kuk Kim, Sang H. Son: Predictability and Consistency in Real-Time Database Systems, in Sang.H Son (ed): *Advances in Real-Time Systems*, Prentice Hall, NJ, 1995.

[9] C.L.Liu, J.W. Layland: Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment, *Journal of the ACM*, 20(1), 46-61 (1973).

[10] K.J.Åström, B.Wittenmark: *Computer Controlled Systems*, Prentice Hall, N.J., 1984.

[11] K.J.Åström, B.Wittenmark: *Adaptive Control*, Addison Wesley 1989.

[12] R. K. Kramer, M. K. Senehi: *Feasibility Study - Reference Architecture for Machine Control System Integration.,* Nat. Inst. of Standards and Technology.

[13] Erik Gyllensvärd, Mladen Kap, Göran Kristensen, Daniel Larsson, Dave Peterson: *The Object Request Broker for Advant® Technology*, unpublished, ABB Industrial Systems, Vasteraas, Sweden 1994.

[14] Dag Folkesson: *Principles for realtime execution in some swedish avionics systems,* unpublished document/seminar at IDA, Linköping University, april 1994. (in swedish).

[15] Current projects at RTSLAB, ID; Linköping University in collaboration with Saab Military Aircraft and Saab Dynamics AB.

[16] P. Loborg, T. Risch, M. Sköld, A. Törne: Active Object Oriented Databases in Control Applications, in proc of 19th Euromicro Conference, *Microprocessing and Microprogramming*, **38**, 1-5 , pp. 255--264, Sept. 1993.

[17] P. Loborg, P. Holmbom, M. Sköld, A. Törne: A Model for the Execution of Task Level Specifications for Intelligent and Flexible Manufacturing Systems, *Integrated Computer-Aided Engineering*, **1**, 3, pp. 185--194, 1994.

[18] P. Loborg, A. Törne: A Layered Architecture for Real Time Applications, Proc. of the *7th Euromicro Workshop on Real-Time System*s, Odense, Denmark, June 1995

[19] G. Fahl, T. Risch, M. Sköld: AMOS - An Architecture for Active Mediators, In *Proc. Workshop on Next Generation Information Technologies a d Systems* (NGITS'92), Haifa, Israel, June 1993

[20] E. Falkenroth, T. Risch, A. Törne: Using an Embedded Active Database in a Control Systems Architecture, Proc. *2nd Intl. Conf. on Appl. of Databases*, San Jose, CA, dec 1995.

[21] E. Falkenroth: Computational Indexes for Temporal Simulation Data, accepted to *8th Intl. Conf. on Scientific and Statistical Database Management*, Stockholm, Sweden, July 1996

[22] Jansson, A.: *Fluid Power System Design: A Simulation Approach*, PhD thesis, Dissertation No. 356, Division of Fluid Power Technology, Dept of Mech. Eng., Linkoping University, Sweden