Real-time Databases for Embedded Systems

White paper

For real-time systems, emerging commercial main-memory database technology such as eXtremeDB provides a good fit.

Introduction

Real-time systems are now used in many application domains, including network infrastructure, telecommunications and financial markets. As real-time systems continue to evolve, their applications become more complex, and often require timely access to, and processing of, massive amounts of data. At the start of a real-time system development project, it is often assumed that a fast enough data processing engine, created specifically for the system and tightly integrated with its code, will meet real-time requirements. However, often the application's input data must be correlated, merged, or compared across all data objects and across time, for filtering or analysis. In addition, the data must be shared by concurrent tasks that have different functions, time requirements and degrees of importance. These issues are better addressed by a proven database management system (DBMS). As a result, real-time systems increasingly feature a commercial, off-the-shelf DBMS as a part of the architecture, to simplify design, streamline development and improve performance.

Real-time vs. Traditional Databases

Like traditional database systems, real-time database systems (RTDBSs) serve as repositories for data and provide efficient storage, retrieval and manipulation of data. The main differences between conventional and real-time databases lie in temporal requirements of the managed data, timing constraints on transactions, and performance goals.

Temporal consistency of data requires the actual state of the external world and the state represented in the database to be close enough to remain within the limits of the application. The most common performance metric of all databases—whether real-time or not—is transaction response time. For traditional database systems this characteristic boils down to the number of transactions per time unit; this measurement is used heavily in optimizing average response time for "traditional" (not-real-time) applications.

For RTDBS, on the other hand, the typical metric is the number of transactions that violate timing constraints (miss their deadline). But even within real-time systems, tolerance for timing constraint violations varies. The cost of missed transaction deadlines ranges from a diminishing value return, to zero value for missed transactions, to negative value to increasingly negative value. This decreasing value often correlates to data's susceptibility to aging. For example, the value of the last known location of objects (e.g. aircraft on a radar screen) diminishes with time, and after some small time interval becomes irrelevant or useless.

One of the most important differences between the databases used by real-time and non-real-time systems is that while the conventional DBMS aims to achieve good throughput or average response time, the real-time DBMS must provide a predictable response time to guarantee the completion of time-critical transactions. Therefore, the design of a real-time database subsystem should avoid using components that introduce unpredictable latencies, such as disk I/O operations, message passing or garbage collection. Thus real-time databases tend to be designed as in-memory database systems. In-memory databases forego disk I/O entirely, and their simplified design (compared to conventional databases) minimizes message passing. The validity of the real-time data in the database may also be compromised if it cannot be updated fast enough to reflect real-world events. To avoid this, the best solution has been a fully time-cognizant transaction manager. At the very least, the database design should provide some means of transaction prioritization.

Hard Real-time vs. Soft Real-time

Many real-time systems, such as nuclear power plants and fly-by-wire vehicles, cannot tolerate any missed deadlines for critical transactions. These are commonly called hard real-time systems. The

response time requirements of hard real-time transactions are such that a failure to execute by the deadline could result in catastrophic consequences. But the majority of real-time systems and their databases fall into the soft real-time category, where violation of timing constraints results in degraded quality, but is to some degree tolerable.

Many factors make it hard to build RTDBSs—and limit the applicability of commercial, off-the-shelf RTDBS almost exclusively to the soft real-time domain. Due to the critical nature of certain application domains (spacecraft control systems, for instance), whole system designs, including RTDBSs, are custom made with spare capacity. In a hard real-time system, designers often use custom algorithms and schedule excessive amounts of time for a transaction, especially a critical one, just to cover a worst-case scenario. These kinds of systems are often time-driven and time-slice scheduled (in contrast to soft real-time systems, which are often event-driven and priority scheduled). Virtually all commercial database concurrency control approaches have been designed to optimize average-case performance rather than worst-case latency. Adapting these techniques for hard-real time is usually impossible.

However, in the soft real-time database arena, the past several years have seen unprecedented commercial availability, and substantial innovation. Complex soft real-time systems need databases to support concurrent data access and provide well-defined interfaces between software modules, while supporting levels of performance and predictability lacking in traditional databases. In this article we concentrate on soft RTDBS that are based on off-the-shelf hardware and other elements. We will analyze some common real-time systems attributes and usage scenarios, and examine how the modern commercially available RTDBS addresses the requirements.

Process Control

The shape of real-time computing has been formed by the requirements of process control and similar applications. Therefore, the database requirements imposed by such systems may be seen as emblematic of data management demands across a wide variety of soft real-time applications.

Industrial control systems are usually attached to sensors that monitor the state of some real-world process, and to controllers to manipulate valves, magnets, and other things that operate in real-time. Industrial controllers are naturally data-driven devices. Control decisions are made based on the input data and the controller configuration parameters, and are executed through the controller's output channels. Input data comes from the field devices (sensors, transmitters, switches, etc.) via the controller's data acquisition interfaces, from supervisory control systems (PC, DCS, PLC) via external controller links, and from other controllers via inter-controller connections. Output data is directed to field control and indication devices, supervisory systems, and other controllers.

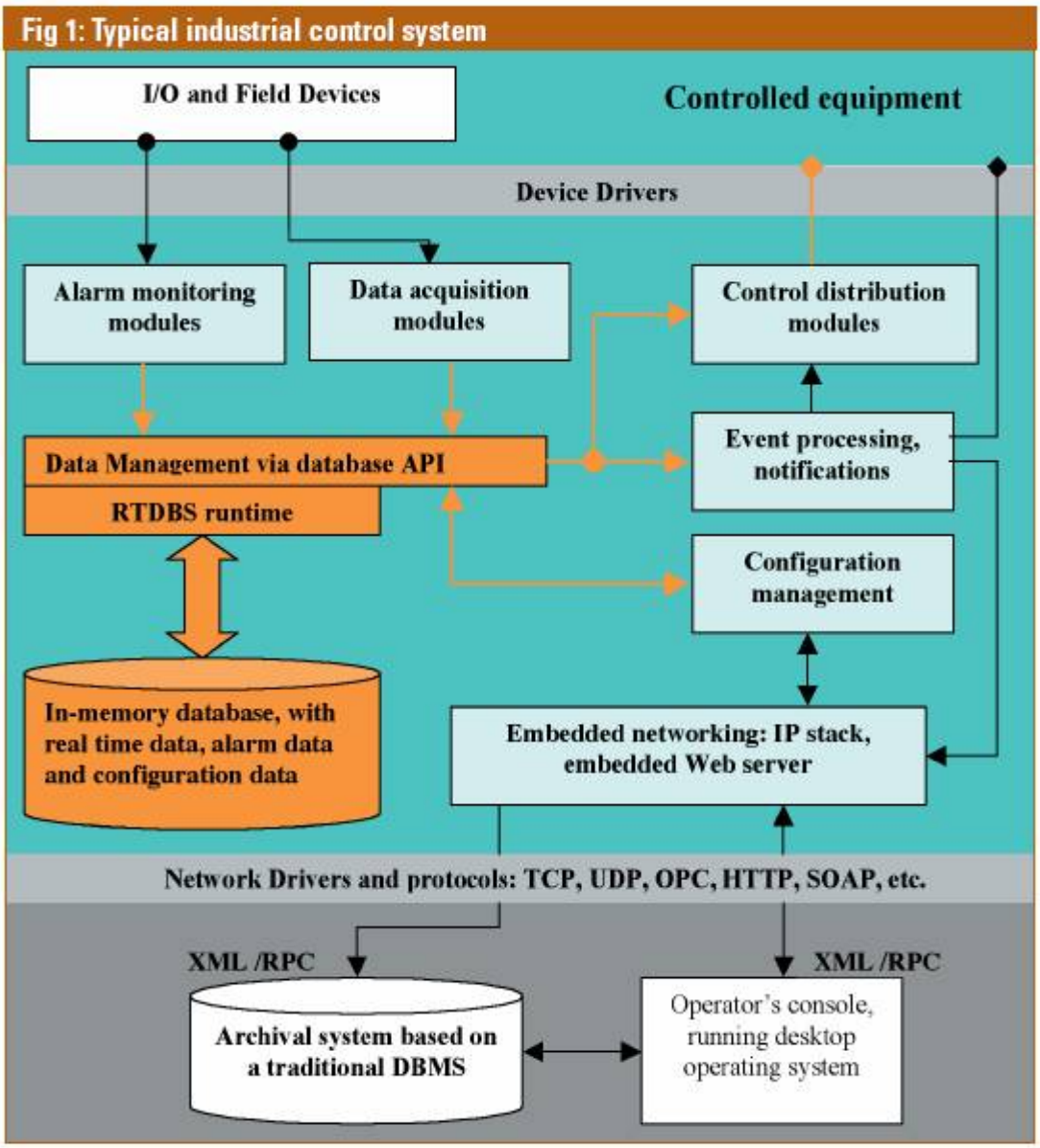
Industrial control is characterized by event-driven operations: system activities are driven largely by interrupts coming from sensors that are bound to specific functions, which transform the sensor signal into a series of signals to the control devices. Often, industrial control systems and their data management subsystems must be highly available because they control real-world processes. The key performance requirement is guaranteed maximum response time under peak load at least for the functions of high priority—otherwise something might crash or explode or expire. Therefore, industrial control systems—and the real-time databases underlying them—must be able to do deadline scheduling

In a typical industrial control system incorporating a real-time database, each device detects the values of some attributes of the real world and makes them available to the database. In turn the database provides the information needed by various system transactions to perform their functions. Figure 1 illustrates this process.

Some other potential real-life application scenarios shed light on the requirements placed on RTDBSs.

Spacecraft Control Systems

The spacecraft control system has multiple antennae, receivers and transmitters. The system must monitor all devices, in addition to monitoring the "health" of the spacecraft itself (power, telemetry, etc.). The volume of data kept by this kind of application is relatively small, but the timing and permanence attributes are quite stringent. In addition to control data, information on each communication channel and device must be maintained. The reliability requirements for such systems dictate redundancy in all system hardware and software components. Almost without exception, spacecraft control systems have not been implemented using commercial databases. The data is kept in memory and directly shared between application modules. However it is conceivable that data management for less critical parts of the spacecraft control systems (such as telemetry archives) could be handled by a commercial RTDBMS. Such data management



This diagram represents the data-centric view of the control process: data collected by sensors in the form of input signals is stored in the real-time database. Output signals are generated by the controller's data management services and are delivered in real-time through the control distribution modules to the controlled equipment (assembly line, motor, generator, combustion engine, etc), to support critical processes.

modules don't fall into the hard-real time category, but rather serve an advisory role to other control subsystems.

Training Simulation Systems

These systems are used to train personnel in such areas as the use of equipment or in strategy, by simulating the real environment and letting trainees interact with it. The timing requirements for such systems are usually derived from the real equipment specifications. The databases for simulation systems usually do not impose any hard real-time requirements, but still need responsiveness.

Telecommunications

Databases play an increasingly important role in telecommunications systems. Developments in

network and switching technologies have made telecom systems and services very data intensive. For example, for network management, the complexity of modern networks creates large amounts of data on network topology, configuration, equipment settings and other parameters. In addition, switches generate large amounts of data on network traffic, faults, etc. The integration of network control, network management, and network administration leads to the use of databases in the core of the network. Architectures such as Intelligent Networks (IN), the Telecommunications Management Network (TMN) and the Telecommunications Information Networking Architecture (TINA) need databases to support their data management services in near real-time. Mobile services entail additional tracking and tracing of mobile equipment. Enhanced provisioning services like the Universal Mobile Telecommunications System (UMTS, or the third generation of mobile network) also require resilient (24x7) near real-time database processing.

The database requirements for the scenarios above vary in their timing requirements, from microseconds to register the electrical current changes or to make routing decisions, to milliseconds for opening and closing valves, to seconds for materials movement.

Commercial real-time database system

Commercial RTDBS technology is still in its infancy, especially when compared to the traditional (non-real-time) database industry, which offers a wide variety of proven commercial products. However, traditional disk-based systems cannot achieve predictable response times in the microseconds or milliseconds range. Main memory is the only technology that fits the profile. Main-memory DBMS are at the heart of real-time databases. Research and development in main-memory database theory and implementation has been active since the mid-eighties. In the past several years it has taken on a new urgency, as inexpensive memory and 64-bit addressing have become a reality.

McObject's eXtremeDB is the latest commercial addition to the family of embedded databases for real-time systems. The main principal behind the design of eXtremeDB is to eliminate performance overhead while providing a predictable and reliable transactional model. McObject's approach works best for pseudo-real-time and real-time systems such as telecommunications equipment, factory floor automation systems, process control, zero-latency consumer electronics devices, and medical equipment.

eXtremeDB is a main-memory storage manager that provides *direct access to data*. eXtremeDB maps its database directly into the application's address space, providing applications with direct pointers to the data elements, eliminating expensive buffer management. This access to data is further optimized by placing the associated access structures on the application's stack. The eXtremeDB runtime code is directly linked with the application so remote procedure calls are eliminated from the execution path. As a consequence, the execution path generally requires just a few CPU instructions.

As a main-memory database, eXtremeDB removes the bottleneck of paging data in and out during I/O operations. To further improve the predictability and performance of database read and write operations, the eXtremeDB runtime uses its own highly optimized memory manager that is responsible for all allocations and de-allocations made by the database runtime. eXtremeDB never relies on the operating system's memory management.

The eXtremeDB transaction manager is implemented via a simple transaction queue, but adds a modern twist: it is possible to assign priorities to transactions at runtime. Five transaction priority levels are supported, from the highest level (ISR) to the lowest (IDLE). McObject promises to extend the priority scheme to support a fully time-cognizant transaction manager in a future version. This approach to the transaction management is well justified when the number of simultaneous transactions are low, and the transactions themselves are short in duration – the time that an application spends in communication with a "lock arbiter" would be comparable with the time to complete the transaction, eliminating any justification for more complex concurrency controls.

eXtremeDB implicitly limits transaction duration (the maximum number of database operations within a transaction is limited), achieving its primary objective of supporting extremely high transaction rates. To provide consistent response times, eXtremeDB implements its own user-space spin-locks to control the mutual exclusion, rather than relying on operating system semaphores. The runtime can be configured to use various techniques such as the "test and set" or the "compare and set" CPU instructions, or direct access to interrupts.

To address the event-driven nature of real-time systems, McObject's eXtremeDB provides

synchronous and asynchronous event processing functionality, which is also designed to be completely self-sufficient with no dependencies on operating system services.

A high availability control interface (HA API) is exposed by the *eXtreme*DB High Availability runtime extensions and provides the means for the application to configure, establish, maintain and terminate *eXtreme*DB HA connections. Unlike other vendors who use *replication algorithms* to provide fault-tolerant system configurations, following the spirit of its design principals, eXtremeDB offers a *time-cognizant two-phase commit protocol* and time-cognizant failure notifications to ensure high availability of the real-time system.

eXtremeDB does not require an operating system to run but if an operating system is available, eXtremeDB can take advantage of it. It is currently available on many RTOS including VxWorks, QNX, Windows CE, real-time Linux platforms, as well as non-real time OS including Linux, Windows NT/2000/XP, Sun Solaris and HP-UX 11.x.

Conclusion

Database systems are designed to manage the persistent data shared among multiple tasks. One of the primary objectives of a database management system is to move the database from one consistent state to the next through the application of transactions supporting the ACID properties (Atomicity, Consistency, Isolation, and Durability). In real-time systems, temporal characteristics of the data are equally important. The majority of real-time systems could be considered as soft-real time systems that do not have critical timing constraints: the expired or missed transactions simply have no value rather than lead to catastrophic consequences. Telecommunications equipment, real-time billing, most process control systems, and consumer electronics devices all fall into this category. For soft real time systems, the modern commercial main-memory database technology such as McObject's eXtremeDB, can provide a good fit. eXtremeDB is designed to operate in the harsh environment of real-time systems, with strict requirements for resource utilization, and is ready to provide the performance and reliability required by real-life applications.