2

REQUIREMENTS AND ISSUES ON REAL-TIME DATABASE SYSTEMS

Kwei-Jay Lin* and Farnam Jahanian**

*Dept. of Electrical and Computer Engineering
University of California, Irvine
Irvine, CA 92697-2625, USA

** University of Michigan
Dept. of EECS
Ann Arbor, MI 48109-2122, USA

1 INTRODUCTION

Real-time systems are now being used for such applications as avionics, space projects, process control, financial market, telecommunication, and air traffic control systems. For these applications, data about the target environment must be continuously collected from the real world and processed in a timely manner to generate real-time responses. In most cases, the amount of information that can be made available to a real-time system is almost unlimited. The questions are how much is enough for the application and how fast the system can handle. These two fundamental questions must be carefully evaluated by real-time system engineers to reach a good balance between the system cost and the computing capability.

Answers to the first question are often specified by the applications. For example, avionics systems would like to obtain the readings on every instrument at least once in every 50 ms while program trading systems can work with a price quote for every stock in every few seconds. From the desired data rate, one can figure out the data volume in the data flow. Given that, real-time system engineers must answer the latter question to see if a real-time system is capable of meeting the requirement. They must implement a system that can process the information sufficiently fast so that incoming data will not be back-logged. Moreover, the system must handle sporadic requests from users and unexpected events from the environment that have strict deadlines. Real-time systems must utilize the resources available and employ intelligent resource management and scheduling policies to ensure that all critical timing constraints are met. In this
way, a real-time system can keep up with the real world and produce useful responses required by the applications.

Using this dataflow-oriented model, which is commonly adopted in many automatic control systems, it appears that all we need is a fast enough data processing engine to meet the real time requirements. However, this may not be the case since the input data in some applications must be correlated, fused and compared both across objects and time so that more intelligent analyses can be conducted. In addition, data may be shared by many concurrent tasks that have different functionalities, criticalness and timing constraints. All these capabilities can be better provided by a powerful database management (DBMS) sub-system. Therefore, some real-time systems include DBMS as part of the systems to simplify the system architecture and to improve the performance.

One of the most important differences between the databases used by a non-real-time system and those used by a real-time system is that the former are usually designed to achieve a good throughput or average response time, whereas the latter must be designed to provide a predictable response time in order to guarantee the completion of any time-critical transaction. Failure to complete a real-time transaction in time may cause severe damage to both the host environment and the database itself. The host environment may be jeopardized due to the lack of response from the system during critical events. The validity of data in the database may also be compromised if some data cannot be updated fast enough to provide a true reflection of the real world. Therefore, it is important for a real-time database (RTDB) system to meet the timing constraints of real-time transactions.

Another important difference is that conventional database systems are designed to maintain database integrity. Data consistency, which means a database state without contradictory information, is used to judge the correctness of a database instance. The response time aspect is often considered as a quality issue in the sense that a short response time is always desirable. This is not the case for real-time applications for which data consistency is a quality issue but the deadline is a correctness issue.

The design of real-time databases is thus performance- and semantic-dependent. It must consider factors such as hardware configuration, system workload and user pattern to improve the performance. Real-time database designers may also utilize different temporal semantics in transaction scheduling algorithms, concurrency control protocols, disk caching, and buffer management protocols to meet the timing constraints defined by real-time applications.