International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 2, February- 2013

Quality-Aware Real-Time Embedded Database

R. Suresh[1], *Jebin P. L*[2], *G. Kannadhasan*[3]
M.E Embedded System Technologies
S.A Engineering College
Chennai77

Abstract

Recent advances in Embedded system have paved the way for next generation real-time applications that are highly data-driven, where data represent real-world status. An embedded database is an integral part of such applications or application infrastructures. Unlike traditional DBMSs, database functionality is delivered as part of the application or application infrastructure. Embedded databases provide an organized mechanism to access large volumes of data for applications. Instead of providing full features of traditional DBMSs, such as complex query optimization and handling mechanisms, embedded databases provide minimal functionality such as indexing, concurrency control, logging, and transactional guarantees.

Key words: Real-time database, embedded database, transaction tardiness, sensor data freshness, QoS management, Multiple Input/Multiple Output (MIMO) control.

1 INTRODUCTION:

With continuing miniaturization and increasing computation power, computing systems are becoming more and more deeply embedded in everyday life and interact with processes and events of the physical world. The interaction between embedded computers and physical processes usually forms a feedback loop, in which physical processes affect computation in computers embedded in devices like automobiles, and vice versa. Systems featuring such tight combination of, and coordination between, systems' computational and physical elements are broadly called cyber-physical systems (CPS).

1.1 Overview of QRED:

QRED targets real-time embedded devices, which have relatively small main memory compared to their secondary storage. Since the capacity of the secondary storage is usually far greater than the size of main memory, databases bigger than the main memory can be used with support from the secondary storage.

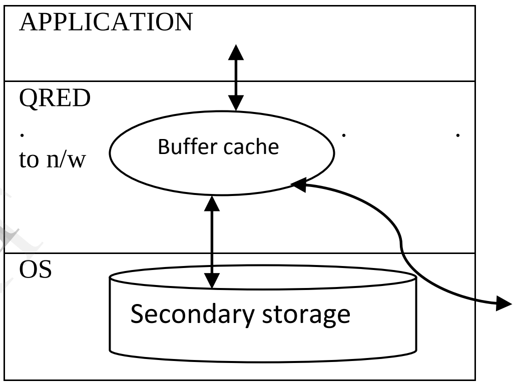

Fig. 1software stack of an embedded system

Fig. 1 shows the software stack of an embedded system, which runs a real-time application with support from a RTEDB. A buffer cache is located in main memory, and it is a cache between the slow secondary storage and the CPU. The buffer cache is global, and shared among transactions to reduce the average response time in accessing data. An I/O request from application(s) for a data object incurs I/O operations to the secondary storage only if the data object is not found in the buffer cache.

2 QOS MANAGEMENT IN QRED

In this we describe our approach to managing the performance of QRED in terms of QoS. We define the QoS Queries in 2.1 and metrics in Section 2.2. An overview of the feedback control architecture of QRED is given in Section 2.3.

2.1 QoS Queries

There are two approaches for processing sensor queries: the warehousing approach and the distributed approach. The warehousing approach represents the current state of- the-art. In the warehousing approach, processing of sensor queries and access to the sensor network are separated. (The sensor network is simply used by a data collection mechanism.) The warehousing approach proceeds in two steps. First, data is extracted from the sensor network in a predefined way and is stored in a database located on a unique front-end server. Subsequently, query processing takes place on the centralized database.

In the distributed approach, the query workload determines the data that should be extracted from sensors. The distributed approach is thus flexible – different queries extract different data from the sensor network – and efficient – only relevant data are extracted from the sensor network. In addition, the distributed approach allows the sensor database system to leverage the computing resources on the sensor nodes: a sensor query can be evaluated at the front-end server, in the sensor network, at the sensors, or at some combination of the three.

2.2 QoS Metrics:

The goal of the system is to maintain QoS at a certain level. The most common QoS metric in real-time systems is deadline miss ratio. The deadlines of transactions are application-specific requirement on the timeliness of the transactions, and the deadline miss ratio indicates the ratio of tardy transactions to the total number of transactions. However, it turns out deadline miss ratio is problematic in RTEDBs because the rate of transaction invocation in embedded databases is very low compared to conventional database systems, which handle thousands of transactions per second. For example, a real-time transaction of a firefighter's PDA, which checks the status of the building, can be invoked on a per-second basis. With such a small number of transactions, the confidence interval of deadline miss ratio can be very wide. This makes the deadline miss ratio not very suitable for a QoS control. Therefore, QRED controls the QoS based on the average tardiness of the transactions. For each transaction, we define the tardiness by the ratio of response time of the transactionto its respective (relative) deadline.

$$tardiness = \frac{response\ time}{deadline}.$$

Another QoS metric, which may pose conflicting requirements, is data freshness. QRED supports the desired data freshness in terms of perceived freshness (PF)

$$PF = \frac{N_{fresh}}{N_{accessed}},$$

Where Nfresh is the fresh data accessed by real-time transactions, and N access is the total data accessed by real-time transactions. When overloaded, the data freshness could be sacrificed in order to improve the tardiness as long as the data freshness is within the user-specified bounds.

2.2.1 I/O Deadline and CPU Deadline:

The definition of tardiness is extended to tardiness in I/O and CPU as follows:

$$tardiness_{i/o} = \frac{response\_time_{i/o}}{deadline_{i/o}},$$

$$tardiness_{cpu} = \frac{response\_time_{cpu}}{deadline_{cpu}}.$$

However, assigning the same static slack factor for both I/O and CPU deadline can be problematic since the ideal slack times for I/O operations and computation change as the system status changes. To this end, QRED dynamically adjusts I/O and CPU deadlines at each sampling period by Algorithm 1.

```
Input: average tardiness i/o and tardiness cpu

Data: State variables Ti/o and Tcpu
If tardiness i/o ≥ tardiness  cpu then
        Ti/o + +; Tcpu = 0;
        δd=α * Ti/o;
        Increase deadline i/o by δd%
Else
        Tcpu + +; Ti/o = 0;
        δd=α * Tcpu;
        Decrease deadline i/o by δd%
End

Deadline cpu=deadline –deadline i/o
```

In Algorithm 1, I/O and CPU deadlines are adjusted by δd% on every sampling period. In a normal state, δd is set to a small number to prevent the high oscillation of deadlines; α is a constant factor and set to 1 in our testbed. However, as a specific resource is being overloaded for consecutive sampling periods, δd increases multiplicatively to speed up the adaptation of pseudodeadlines. Since the sum of the I/O deadline and the CPU deadline is equal to the deadline of the transaction, the multiplicative increase of one pseudodeadline implies the multiplicative decrease of the other pseudodeadline.

2.3 QoS Management Architecture

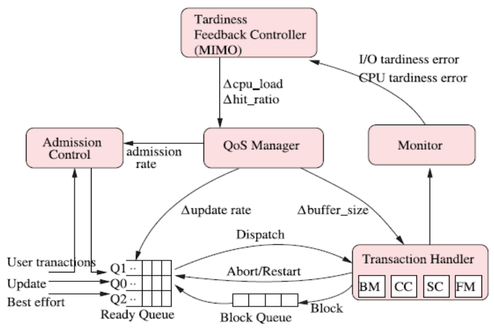

Fig 2 QoS management architecture of QRED

Fig. 2 shows the QoS management architecture of QRED. It consists of the MIMO feedback controller, QoS Manager, performance monitor, transaction handler, admission controller, and ready queue. Admitted transactions are placed in the ready queue. Fig. 2 shows three separate queues in the ready queue. Temporal data updates are scheduled in Q0 and receive the highest priority. Q1 handles real-time user transactions. Non real-time transactions in Q2 have the lowest priority and they are dispatched only if Q0 and Q1 are empty. The transaction handler manages the execution of the transactions. It consists of a freshness manager (FM), a unit managing the concurrency control (CC), a basic scheduler (SC), and a buffer manager (BM). Transactions in each queue are scheduled in FCFS manner. The FM checks the freshness of a data object before accessing it, using the time stamp and the absolute validity interval of the data.

The monitor computes the I/O and CPU tardiness, i.e., the difference between the desired I/O (and CPU) response time and the measured I/O (and CPU) response time at each sampling instant. Based on the errors, the MIMO feedback controller computes the required buffer hit ratio adjustment (Δhit ratio) and CPU load adjustment (Δcpu load). The QoS manager estimates the required buffer size adjustment and update rate adjustment based on Δhit ratio and Δcpu load. Update transactions waiting in the ready queues are discarded according to their update rates. The buffer cache size is adapted by the BM in the transaction handler as requested by the QoS manager.

3 IMPLEMENTATION

Commonly, embedded databases are used as components of open systems consisting of many other interacting software components and applications, instead of as isolated monolithic databases.

3.1 Hardware and Software:

The hardware platform used in the tested is the MSP430 which is ultra low power controller and PIC16F877A. The MSP debug stack is officially supporting the following Operating Systems:

- Windows XP SP3, 32- and 64-bit
- Windows Vista, 32- and 64-bit
- Windows 7, 32- and 64-bit
- Linux Ubuntu 10.04
- Linux Suse 10.3

QRED is an extension of Berkeley DB and QeDB, which is a popular open source embedded database. Berkeley DB provides robust storage features as found in traditional database systems, such as ACID transactions, recovery, locking, multithreading for concurrency, and slave-master replication for high availability. However, QeDB does not provide the QoS support in terms of transaction tardiness and data freshness, which is the main objective for the design of QRED.

3.2 Simulation And Circuit Diagram

Proteus simulation circuit is shown in the following window. PIC 16F877A is taken as a system or controller. The PIC uses its internal clock source operated at 4.0MHz. It has 256 bytes of internal EEPROM which means only the eight most significant bits of DS18B20 output is recorded. This works on the principle of successive approximation method. ADC initialization takes place at the ADC selection port RA0, RA1, RA2. This senses the signal from the sensor node and stored in the register for processing. To boost up the input output operation, to achieve high data freshness, indexing, logging, concurrency control, in terms of reducing the computation time, computation power, MSP 430 is used.

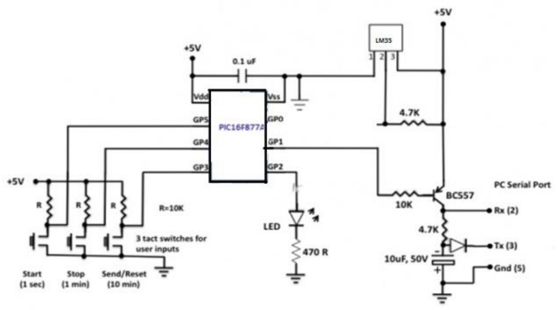

Fig 3 PIC16F877A temperature data logger.

The data logger offers three options for sampling interval i.e. 1sec, which starts data logging, 1min, which stops data logging and 10 min, which is send/reset for transferring data to pc through serial port. The following circuit diagram in fig.4 explains the design simulation of QRED architecture. This simulation also detects the error and based on the error the control feedback assigns the required buffer size for each I/O operation.

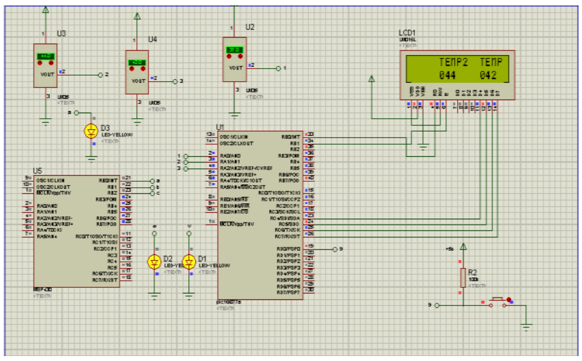
Fig 4 Simulation circuit diagram with MSP430

3.3 Simulation Results

Simulation results are shown on the LCD on user demand and periodical alert about the shooting temperature is made known to the user in case of potential danger both to the user and fire men through PDA. The sensor used in this project is LM35. It is a digital temperature sensor manufactured by Dallas Semiconductor (now MAXIM) that can measure temperature ranging from -55°C to +125°C with an accuracy of ±0.5°C over the range of -10°C to +85°C. This helps in safeguarding the lives of firemen who are carrying out intense operation on a burning building. By using MSP430 the maximum and minimum computation power achieved is between 3.6v and 1.8v. The maximum and minimum threshold voltage is 1.9v and 0.9v.

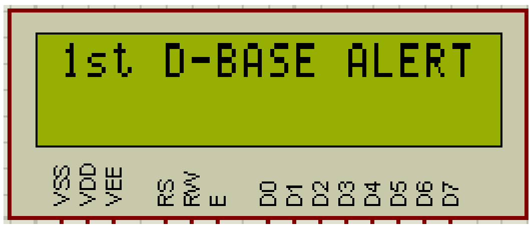

Fig 4 Alert signal output

4 Conclusions And Future Work

The main objective of this project is to establish safety in burning building. The wireless are more flexible and can avoid the trouble of rewiring, because wireless network can meet the moving and changing of topology .It will greatly improve the performance and efficiency of data transmission from the high temperature sensor, and reduce the costs of extending the system. The application of WSN can realize the real-time monitoring of working region.

In the future, we plan to enhance QRED in several different directions through hardware platform. . A real time clock chip can also be added in the project to keep record of the actual time stamp. Experiment results prove that this QRED works well, and can be put forward to practical application. This gives desired timelines of data freshness with improved quality of service in real time applications. This reduces considerable CPU and I/O overload in processor and enhances fast reliable transaction for real time monitoring system.

5 REFERENCES

1)W. Kang, S. H. Son, J. A. Stankovic, and M. Amirijoo, “I/O-aware deadline miss ratio management in real-time embedded databases,” in *The 28th IEEE Real-Time Systems Symposium (RTSS),*

2) J. R. Haritsa, M. Livny, and M. J. Carey, “Earliest deadline scheduling for real-time database systems,” in *In Proceedings of the 12th Real-TimeSystems Symposium,*

3) S. Chen, J. A. Stankovic, J. F. Kurose, and D. Towsley, “Performance evaluation of two new disk scheduling algorithms for real-time systems,” *The Journal of Real-Time Systems,*

4) X. C. Song and J. W. S. Liu, “Maintaining temporal consistency: Pessimistic vs. optimistic concurrency control,” *IEEE Trans. on Knowl.and Data Eng.,*

5) C. Lu, X. Wang, and C. Gill, “Feedback Control Real-Time Scheduling in ORB Middleware,” Proc. Ninth IEEE Real-Time and Embedded Technology and Applications Symp.(RTAS ’03), pp. 37-48, 2003.

6) Y. Lu, T.F. Abdelzaher, and A. Saxena, “Design, Implementation, and Evaluation of Differentiated Caching Services,” IEEE Trans. Parallel and Distributed Systems, vol. 15, no. 5, pp. 440-452, May 2004.

7) S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus, “Using Control Theory to Achieve Service Level Objectives in Performance Management,” Real-Time Systems, vol. 23, nos. 1-2, pp. 127-141, 2002.

8) S. Nath and A. Kansal, “FlashDB: Dynamic Self-Tuning Database for NAND Flash,” Proc. Int’l Conf. Information Processing in Sensor Networks (IPSN), 2007.