내장형 시스템을 위한 실시간 데이터베이스 엔진 설계 및 구현

The Design and Implementation of a Real-Time Database Engine For Embedded Systems

김 흥 섭* 문 승 진**
Heung-Seob Kim Seung-Jin Moon

요 약

정보화 사회가 가속됨에 따라 첨단 기능을 탑재한 많은 기기들이 만들어지고 있다. 기존의 펌웨어로 기능을 구현하는 데는 많은 제약 사항들이 생기게 되었다. 펌웨어의 대안으로 등장하기 시작한 것이 임베디드 운영체제이다. 임베디드 리눅스는 기존의 임베디드 운영체제의 고비용의 문제를 해결할 수 있는 방안으로 주목을 받기 시작을 했으며, 많은 연구가 진행되고 있다. 리눅스는 많은 프로그램을 가지고 있다. 그러나, 임베디드 데이터베이스 프로그램들은 고가의 비용을 요구하고 있다. 본 논문에서 제시하는 ERT DE는 기존의 오픈되어진 소스를 리눅스에서 가능하게 하였으며, 쿼리 레벨에서의 리얼타임 기능을 구현하고자한다.

Abstract

With the progress of information-oriented society, many device with advanced technologies invented by many companies. However, the current firmware technologies have many problems to meet such high level of new technologies. Recently it become necessary new manufactures of information-oriented society to require such embedded operating system as their system-level platform. Embedded linux, which could be an alternative proposal of existing high-cost embedded operating system, become available commercially by many companies. Linux has many programs. But, embedded databases require very high cost. In this paper, we suggest ERT DE which has a small size and is suitable for embedded real-time technologies.

1. 서 론

1999년 후반부터 대중에게 알려지기 시작한 임베디드 시스템은 실제로는 꽤 오래 전부터 우리의 생활 속에서 자리 잡고 있었다. 최근 들어 많은 수의 정보 가전기기들과 개인용 휴대기기들이 나왔으며, 제품들마다 다른 하드웨어의 사양을 지니고 있으며, 각 제품마다 필요로 하는 기능도 상이한 경우가 많다. 이러한 제품들을 기존 Windows등의 운영체제로는 최적화된 기능을 제공하기 힘들게 되었고, 이에 따라 최적화된 성능, 개발 기간 단축, Time-to-Market 등을 제공하기 위해서는 임베디드 운영체제가 대안으로 제시되었다. 이에 따라, 임베디드 리눅스와 리얼타임 리눅스가 많은 부분에서 연구되고 쓰여지고 있다.

기존 임베디드 시스템 장비들은 산업 기기 제어 등을 기반으로 하여, 사용자의 편리성보다는 임베디드 장비의 운용성에 중심을 두었다. 그러나 정보 가전기기와 개인용 휴대기기들은 장비의 운용성 뿐만 아니라 사용자의 편의성도 요구하게 되었고, 갈수록 더 많은 데이터의 수용과 관리를 필요로 하는 이러한 기기들에 대해서 데이터베이스의 필요성이 증가되고 있다. 하지만, 임베디드 리눅스용 데이터베이스로 공개된 오픈 소스는 없고, 리눅스용 데이터베이스 프로그램들이 크기가 크기 때문에 임베디드용으로는 적합하지 않다. 최근

* 정 회 원 : Astonlinux 주임연구원
mac21c@astonlinux.com
** 종신회원 : 수원대학교 컴퓨터과학과 조교수
sjmoon@mail.suwon.ac.kr

상용화된 임베디드 데이터베이스 프로그램들이 등장하고 있으나, 고가의 비용을 요구하고 있다.

본 논문에서는 임베디드 시스템, 리얼타임 시스템, 임베디드 리눅스, 리얼타임 리눅스에 대하여 조사하였으며, 갈수록 필요성이 증대되고 있는 부분인 임베디드 데이터베이스에 대해서도 연구하였다. 본 연구는 Embedded Real-Time Database Engine (ERT DE)에 관한 설계 및 구현을 실시하였으며, 리얼타임 이론이 적용되기 전의 데이터베이스와 리얼타임 이론이 적용된 후의 데이터베이스간의 성능을 비교하여, 리얼타임 이론이 적용되지 않은 데이터베이스와 적용된 데이터베이스 간의 데드라인을 넘어선 트랜잭션의 수나 평균적인 트랜잭션 응답 시간 등을 통해 두 데이터베이스간의 비교를 실시한다. 본 연구의 구현 환경으로는 임베디드 시스템을 구현하기 위해서 MPC860T 칩을 내장하고 있는 테스트용 보드를 사용하였다. MPC860T 테스트용 보드에 운영체제로써 Real-Time Linux (RTLinux)를 올렸다. 이러한 기반 위에 본 논문에서 구현하고자 하는 Embedded Real-Time Database (ERT DE)를 이식하였다. 본 논문의 구성은 2장에서는 관련 연구로 임베디드 시스템과 리얼타임 시스템에 대해서 설명하였고, 3장에서는 본 연구에서 제시하는 ERT DE의 설계 및 구현에 대해서 설명하였고, 4장에서는 리얼타임 이론이 적용되기 전의 데이터베이스 엔진과 리얼타임 이론이 적용된 후의 데이터베이스 엔진에 대한 결과 분석 및 평가를 하였고, 5장에서는 결론 및 향후 과제에 대해서 기술하였다.

2. 관련 연구

2.1 Embedded System

임베디드 시스템이라 함은 특정한 기기에 주어진 작업을 수행하도록 구동시키는 시스템이라 할 수 있다. 첨단 기능이 들어있는 가전제품이나 컴퓨터, 엘리베이터, 공장 자동화 시스템 등등 특정한 기기를 운용할 수 있는 운용체제라면 임베디드 시스템이라 할 수 있다.

1950년대 통신 장비 제어를 위해 등장한 임베디드 시스템은 1970년대 후반부터 표준화된 대량생산이 가능하였으며, 1990년대 초반까지 군사용 제어, 산업 기기 제어 등의 목적으로 많이 사용되어 왔다. 90년대 후반부터 컴퓨터 산업과 정보 가전 기기의 발전으로 임베디드 시스템은 첨단 산업으로 각광 받기 시작하였다[1].

임베디드 분야는 Windows, Linux등의 범용 운영체제를 소형화한 것과 VxWorks, PSOS, Lynx등의 Real-Time Operating System(RTOS)들이 경쟁을 하고 있다. 70년대부터 산업용 임베디드 시장에서 출발한 VxWorks, PSOS, Lynx등의 RTOS들은 현재의 임베디드 시장을 장악하고 있기에 RTOS를 임베디드 운영체제와 동일 시 하는 경향이 있다. 하지만 정확히 말하자면 임베디드 운영체제는 RTOS를 포괄하고 있는 폭넓은 분야이다[2,17].

임베디드 시스템을 설계하거나 개발할 때 다음과 같은 사항을 고려해야한다[1].

- Real-Time, Reactive
- Small Size, Low Weight
- Safe, Reliable
- Harsh Environment
- Cost Sensitivity

2.2 Embedded Database

역사적으로 많은 상용 데이터베이스 산업은 높은 성능의 Online Transaction Processing(OLTP), 복잡한 Query Processing, System 비교를 위한 산업적 표준 지표 등의 요구에 의해서 발전해 왔다. 앞에서 제시한 상용 데이터베이스의 기준들은 임베디드 데이터베이스에는 적당하지 않다. 임베디드 시스템은 전형적으로 상당히 간단한 쿼리들을 수행한다. 임베디드 데이터베이스는 임베디드 시스템에 이식이 되어야 하므로 작은 사이즈로 설.

계되어야 한다.

임베디드 데이터베이스를 설계할 때 다음과 같은 사항들을 고려해야 한다.

- Small footprint
- Robustness
- Easy maintenance

임베디드 데이터베이스가 가져야하는 세 가지 요구사항 중에서 관리(maintenance)와 견고성(robustness)은 가장 큰 이슈이다. 임베디드 데이터베이스는 임베디드 시스템에 이식되어야 하기 때문에 작은 사이즈를 가져야 하고, 따라서 많은 기능들을 제공하지 못하고 있다[3].

사용자는 임베디드 디바이스에 저장되어 있는 데이터를 신뢰할 수 있어야 하고, 사용자가 원활하게 수행되어지는 유닛을 가지기 위해 어떤 유사한 시스템 관리를 일일이 수행하게 해서는 안된다. 다행스럽게, 사용하기 쉽고, 견고성(robustness)있게 만드는 것은 작은 사이즈를 갖게 하는 것과는 상반되는 경우가 많다. 우선 작은 사이즈로 데이터베이스를 만들고, 임베디드 데이터베이스가 가져야할 세 가지 요구사항을 만족시켜야한다.

2.3 Real-Time System

실시간 시스템은 기존의 컴퓨터 시스템과 달리 시스템 동작의 정확성이 논리적 정확성뿐만 아니라 시간적 정확성에서도 좌우되는 시스템을 말한다. 이러한 실시간 시스템의 전형적인 예로서 제어시스템을 들 수 있다. 제어시스템은 감지장치(sensor)로부터 입력을 받아들여 이를 정해진 시간 내에 처리하여 작동장치(actuator)로 출력하며 극히 작은 시간적 오차를 허용한다. 실시간 시스템의 응용분야로는 핵발전소의 제어, 공정제어, 병원의 감시 장치, 항공기 제어, 무기 체계, 우주선의 운항 및 유도 등의 분야를 들 수 있다.

실시간 운영체제는 그 특성상 MS-DOS나 Windows 등과 같은 범용 운영체제의 형태로 쓰이지 않고, 내장 제어 시스템과 같은 특수 목적으로 사용되는 경우가 대부분이다. 실시간 운영체제가 가져야 할 몇 가지 특징을 생각하면 다음과 같다[2,18,19].

- 다중 쓰레드를 지원하고, 선점 가능해야 한다.
- 쓰레드 간의 우선순위를 보장하여야 한다.
- 쓰레드 간의 동기화를 지원해야 한다.
- 운영체제의 행동이 명확해야 한다.

2.4 Real-Time Database

Real-Time Database System(RTDBS)은 어떤 서비스 데드라인까지 작업을 처리할 수 있도록 디자인되어진 트랜잭션 처리 시스템(Transaction Processing System)이다. RTDBS의 목적은 데드라인을 만나기 전에 트랜잭션을 처리하는 것이다[20]. 그러므로, 최소한의 트랜잭션 응답시간(Response time)을 요구하는 상업적인 Database Management System(DBMS)과는 대조가 된다. 여기서 강조하는 것은 트랜잭션의 시간 구속력(Time Constraint)을 만족시키는 것이다. 트랜잭션이 데드라인을 넘기게 되는 이유는 물리적 자원들(CPU, Disk, Memory)과 충돌이나 논리적 자원(data)의 충돌 때문이다[4].

RTDBS는 통신 산업, 무선 통신 시스템, 핵반응 제어, 교통 제어 시스템, embedded 시스템, 로봇 공학, 군사 시스템 등에서 사용되어지고 있다[5].

2.5 Real-Time Linux (RTLinux)

RTLinux[9]는 리눅스에 Hard Real-Time 기능을 부여하기 위하여 시작된 프로젝트이다. RTLinux는 리눅스의 커널을 다시 설계하지 않고, 리눅스의 커널에 리얼타임 기능을 변경하여 사용하고 있다. 리눅스 아래에 RTLinux 커널을 올리고 기존 리눅스 커널을 RTLinux 커널에서 돌아가는 하나의 프로세스로 만들어서 리얼타임 태스크와 기존의 리눅스 커널이 공존하는 형태를 만들었다.

RTLinux 커널은 리얼타임 프로세스를 생성하고 이들을 스케줄링(scheduling)하며, 인터럽트가 발생하였을 때 이를 처리하며 리눅스와의 통신을 담당하는 역할을 한다. 기존 리눅스 커널은 그대로 일반 리눅스 프로세스를 관리하고, 자신의 인터럽트를 처리한다[6].

▪ RTLinux 장점

RTLinux의 구조는 무척 단순하기 때문에 기존 리눅스 프로그래머들이 쉽게 접근할 수 있다는 장점이 있다. 리얼타임 기능이 필요한 부분만 리얼타임 태스크로 구현하고 나머지는 이미 어느 정도 검증되어 있는 안정된 리눅스 상에서 동작하므로 안정적이고, 디버깅이 용이하다는 장점이 있다.

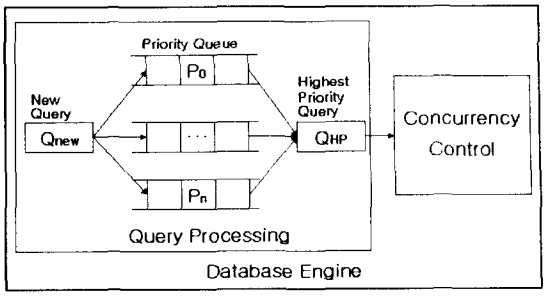
▪ RTLinux 단점

RTLinux는 기존의 리눅스가 가진 기능들을 리얼타임으로 바꾼 것은 아니다. 단지 리얼타임 기능을 추가하였을 뿐, 대부분의 서비스는 기존 리눅스에서 그대로 담당하게 된다. 리눅스 코드의 대부분을 차지하는 디바이스 드라이버도 그대로 사용되고 있으며, 이들에게 리얼타임 기능이 필요한 경우 이들 디바이스 드라이버를 다시 설계해서 만들어야 하는 문제가 있다. 또한 RTLinux는 기존의 리눅스의 기능에 추가를 한 것이기 때문에 여전히 덩치가 크다는 부담이 있다.

3. ERT DE(Embedded Real-Time Database Engine) 설계 및 구현

3.1 ERT DE 설계

ERT DE (Embedded Real-Time Database Engine)은 리얼타임 이론을 적용한 입력과 검색이 빠른 데이터베이스 엔진이다. ERT DE는 ISAM(Indexed Sequential Access Method)[7] 방법을 채택하고 있다. ERT DE는 [8]에서 소개한 데이터베이스 라이브러리 소스를 기반으로 하고 있다. 임베디드 시스템에서 제공하는 용량만 충분하다면 무한대의 데이터베이스를 검색할 수 있다.



(그림 1) Embedded Real-Time Database Engine (ERT DE)

본 논문에서는 [8]에서 소개한 데이터베이스 라이브러리 소스를 Linux에서 돌아갈 수 있도록 변경하였으며, 리얼타임 이론을 접목 시켜서 ERT DE를 설계하였다. ERT DE는 데이터베이스 쿼리 단계에서 다중 큐를 만들어서 리얼타임 이론을 적용하고 있다.

그림 1에서처럼 ERT DE는 n개의 Priority Queue를 포함하고 있고, 이 Priority Queue의 순서에 의해서 상위 쿼리부터 처리하게 된다. ERT DE는 다음과 같은 다섯 가지 쿼리 처리 과정을 준수하여 운용하게 된다.

- 새로운 쿼리(Qnew)를 받아들인다.
- ERT DE는 Qnew의 우선순위(Priority)를 분석하여 해당하는 Priority Queue에 넣는다.
- ERT DE는 Priority Queue (P_0 - P_n)에 쿼리가 있는지 매 주기마다 확인한다(본 논문에서는 1ms 간격).
- Priority Queue에 쿼리가 있을 경우, ERT DE는 최상위 우선순위를 가지고 있는 P_0 Queue에 있는 쿼리들을 처리하기 시작하여 P_n Queue에 있는 쿼리까지 처리한다.
- ERT DE가 현재의 처리해야 할 쿼리보다 상위 우선순위를 가지는 쿼리가 상위 Priority Queue에서 확인이 되면, 상위 Priority Queue의 쿼리를 먼저 처리한다.

또한, ERT DE는 Query 분석 컴포넌트와 Query 처리 컴포넌트로 나눌 수 있다.

3.1.1 Query 분석 컴포넌트

Query 분석 컴포넌트는 받아들인 쿼리를 분석하여 쿼리의 우선순위에 적합한 Priority Queue에 쿼리를 저장하고, Priority Queue에 있는 쿼리 중에서 최상위 우선순위를 가지는 쿼리를 Query 처리 컴포넌트에 넘겨준다.

Query 분석 컴포넌트로 입력되어지는 쿼리는 다음과 같은 양식을 가진다.

```
<priority>, <command>, {<value>, <value>, ...}
```

쿼리의 각 token들은 콤마(',')로 구분한다.

Query 분석 컴포넌트에서는 쿼리를 표 2와 같은 Query Structure로 변환되어진다.

(표 1) ERT DE의 Query Token에 대한 설명

token	Description
priority	쿼리에 대한 우선순위. priority에 의해서 queue와 테이블이 결정된다.
command	수행하고자 하는 명령. INSERT, DELETE, FIND, UPDATE 등
value	처리하고자 하는 값들. table의 attribute의 개수와 일치해야하고, 콤마(',')로 구분한다.

(표 2) Query Structure

```
struct _Query {
    int priority;
    char *command;
    char *tablename;
    char *value;
}
```

(표 3) Priority Queue Structure

```
struct _Priority_Queue {
    struct *_Query;
    struct *next;
    struct *prev;
}
```

Priority Queue는 표 3과 같은 구조를 가지고, 각 Priority Queue는 Double linked list의 구조로 되어 있다.

3.1.2 Query 처리 컴포넌트

Query 처리 컴포넌트는 Query 분석 컴포넌트에서 넘어온 쿼리를 처리하는 데이터베이스 엔진부분이다. Query 분석 컴포넌트에서 넘어온 쿼리를 파싱(parsing)하고, 쿼리의 처리결과를 화면에 출력하여준다. ERT DE는 ISAM을 기반을 하고 있고, 인덱스를 관리하는 CIdx class와 데이터를 관리하는 CHdb class로 구성되어있다.

▪ CIdx Class

인덱스 파일(.idx)을 컨트롤하는 클래스로써 클래스가 실행됨과 동시에 기본적으로 인덱스명과 인덱스 리스트를 메모리에 설정해 줘야한다. 기본적인 인덱스의 크기는 25Byte이다.

▪ CHdb Class

데이터베이스 파일(.db)을 컨트롤하는 클래스 CHdb는 인덱스 파일을 위한 클래스이다. CHdb 클래스로 데이터베이스를 구동시키면 *.idx와 *.db 두 개의 파일이 생성된다. idx 파일은 색인정보가 들어 있는 파일이고, db 파일은 데이터가 저장되어 있는 파일이다. CHdb안에는 CIdx형의 m_Idx가 설정되어 있으며 이 m_Idx가 idx 파일을 컨트롤 한다. CHdb는 데이터를 인덱싱하고 인덱싱한 데이터는 모두 m_Idx가 컨트롤하며, m_Idx에서 얻는 데이터 위치포인터와 데이터 길이 인자를 가지고 데이터를 로드하는 일을 담당한다. 기본적인 데이터 블록의 크기는 1024Byte이다.

3.2 ERT DE 구현 환경

3.2.1 오픈소스 데이터베이스 엔진 포팅

임베디드 시스템으로서 RTLinux를 선택하였고,

MPC860T[13] 칩을 내장하고 있는 테스트 보드를 선정하였다. 보드의 기본적인 구성을 보면 다음과 같다.

(표 4) MPC860T 테스트용 보드 구성

Main Processor	XPC860TZP50D3
EPROM	1 MBytes
FLASH	4/8/16 MBytes
SDRAM	16 MBytes
Ethernet Port	1 (10/100Mbps)
Serial Port	2 Port (WAN0/WAN1)
Console Port	1 Port (RS232C)

환경구축을 위한 작업은 다음과 같은 순서로 진행되었다[14,15,16].

▪ 크로스 개발 환경

임베디드 시스템의 개발 환경은 전형적으로 개발 환경 호스트와 타겟 시스템으로 구성된다. 데스크탑 PC나 UNIX 시스템과 같은 호스트는 메모리나 디스크의 공간, 저장매체, 프린터 그리고 다른 주변기기들이 충분하게 갖추어져 있다. 타겟 시스템은 응용 프로그램을 디버그 하거나 테스트를 하는데 있어서 많은 제약이 따른다. 따라서 임베디드 시스템의 개발 환경은 프로그램을 작성하고 컴파일, 디버깅, 테스트를 할 수 있는 호스트와 호스트에서 생성된 프로그램을 적용시킬 타겟 시스템으로 구성된다. 호스트 컴퓨터와 타겟 보드는 RS232C 시리얼 케이블과 10/100M Ethernet 케이블로 연결되어 있다. RS232C 시리얼 케이블은 타겟보드의 콘솔화면을 호스트 PC에서 볼 수 있도록 연결 시켜준다. 10/100M Ethernet은 RTLinux 커널과 램디스크 이미지를 호스트 컴퓨터에서 타겟 보드로 전송할 때 사용하고 파일 시스템을 NFS로 마운트하거나 응용프로그램을 디버깅 할 때 사용한다.

타겟 시스템에 적재될 리눅스 커널과 부트로더를 개발하기 위해서 PowerPC용 크로스 컴파일 환경을 구성하였다[10,12]. 크로스 컴파일러 구성을 위해서 사용된 프로그램은 표 5의 "Tools" 참고하면 된다.

(표 5) 호스트 개발 환경

▪ OS	Hancom Linux 2.0
▪ Platform	X86 Processor (Intel 350MHz)
▪ Tools	binutils-2.10.91.2
	gcc-2.95.3
	linux-2.4.4
	glibc-2.2.3
▪ BootLoader	PPCBoot-0.9.3
▪ Utility	bash, vi 등

▪ 부트로더 구축

부트로더는 PowerPC 계열에서 가장 많이 사용하고 있는 Embedded PowerPC Linux Boot Project (PPCBoot)[11]를 사용하였다. PPCBoot는 GPL에 따라 개발되고 있는 PowerPC 프로세서를 지원하는 부트로더이고 bootloader가 이미지와 커널의 이미지를 결합한 형태의 이미지를 생성한다.

▪ 램디스크 파일 시스템 구축

본 논문에서 테스트용으로 사용하고 있는 보드는 일반적인 데스크 탑 PC와 달리 하드디스크가 없는 보드이다. 하드디스크가 없는 경우 보통 파일 시스템을 RAM(Read-write Access Memory)에 적재를 한다. 램디스크는 커널에서 지원해주는 것으로 리눅스 커널은 보드에서 사용 가능한 RAM 공간에 파일 시스템을 적재하게 된다. 본 논문에서는 데이터베이스의 원활한 작동을 위해서 보드의 운용에 필요한 가장 최소한의 유틸리티와 라이브러리만을 램디스크 파일 시스템에 포함하였다.

▪ Linux 적재

크로스 컴파일러로 컴파일 되어진 Linux 이미지 (vmlinuz)가 PPCBoot와 결합하여 vmlinux.boot 파일이 생성되고, 이 파일을 ROM(Read-Only Memory)에 적재하여 부팅시킨다.

▪ RTLinux 적재

타겟 시스템에 적재될 커널은 linux-2.4.4에 RTL

inux 커널을 module로써 올리는 것이다. 앞서 말했듯이 RTLinux를 리얼타임 기능을 가진 새로운 커널을 생성한 것이 아니라 Linux 커널에 리얼타임 기능을 추가한 것이다[21].

RTLinux를 컴파일하여 RTLinux용 Module들과 device 화일들을 생성하고, 생성되어진 RTLinux의 Module(rtl.o, rtl_time.o, rtl_sched.o, rtl_posixio.o, rtl_fifo.o)은 insmod 명령에 의해 Linux 커널로 올려진다.

최초에 올려지는 rtl.o화일은 Linux 커널에게 최하위 우선순위를 부여하여 RTLinux 커널이 Linux 커널보다 상위의 우선순위를 가지고 실행하게 된다. 또한 RTLinux에서의 응용프로그램들은 Module Program으로 만들어서 실행하게 된다.

▪ ERT DE 적재

ERT DE의 소스를 이미 구축되어진 PowerPC용 크로스컴파일러를 이용하여 컴파일 한 후, 생성된 실행 파일과 관련 파일들을 Ramdisk Image의 /bin 디렉토리에 넣었다. *.idx화일과 *.db화일들은 ERT DE의 실행화일과 같은 디렉토리에 있어야 한다.

4. 성능 및 결과 분석

4.1 Test 환경 설정

Real-Time 이론을 접목시키기 전의 Database Engine (Normal DE)과 리얼타임 이론을 적용시킨 후의 Database(ERT DE)를 비교하기 위해서 로봇제어 시스템의 데이터를 예로 이용하였다. 본 논문에서 ERT DE를 이식한 것은 개발용 보드이기 때문에 실제적인 데이터를 얻을 수 없기 때문에 로봇제어 시스템에서 얻을 수 있는 데이터에 근거하여 Normal DE와 ERT DE간의 성능 비교를 하였다.

물체를 인식하고, 인식된 물체와의 충돌을 방지하면서 0.1m/sec로 이동하는 로봇의 데이터를 이용하여 테스트를 하였다. 로봇은 3종류의 센서를 가지고 있다.

▪ 초음파 물체 감지 센서

로봇은 초음파를 이용하여 전방의 물체를 감지할 수 있다. 물체감지 센서는 30도의 전향각을 가지고 있으며, 120도 방향의 물체를 감지하기 위해서 4개의 물체 감지 초음파 센서를 가지고 있다. 초음파는 340m/sec의 속도를 가지고 있고, 6.8m를 20ms에 도달 할 수 있다. 따라서, 초음파 물체 감지 센서는 되돌아오는 시간을 계산하여 3.4m 전방에 있는 물체를 감지 할 수 있다. 20ms이하일 경우에 데이터베이스에 센서번호와 도달시간을 저장한다. 거리는 도달시간(ms)/2 * 0.34m 로 계산 할 수 있다. 4개의 초음파 센서는 20ms주기로 동시에 발사한다.

▪ 위치 감지 센서

로봇은 이동을 위하여 4개의 바퀴를 가지고 있다. 바퀴의 속도를 5ms마다 데이터베이스에 저장을 하고, 10ms마다 4개의 바퀴 속도를 동기화 시키고, 바퀴번호, 속도, 이동거리를 위치 감지 센서 테이블에 저장한다. 위치 감지 센서 매 10ms마다 첫 번째 바퀴에는 나오는 속도를 계산하여 로봇의 위치를 계산한다.

▪ 충돌 감지 센서

로봇의 표면에는 충돌감지센서가 장착되어 있어서 만약 로봇이 어떠한 물체에 충돌하게 되면 로봇은 즉각적으로 정지를 하게 되고 현재의 상태를 데이터베이스에 저장하게 된다. 충돌 감지센서에 대한 특정한 주기는 없고, 충돌센서가 충돌을 감지하였을 때 인터럽트를 발생시켜 로봇의 움직임을 중지시키고, 데이터를 저장한다. 충돌 감지 센서는 비주기 데이터이지만 테스트에서는 30ms의 주기와 최상위 우선순위를 가진다.

위에서 설명한 로봇의 데이터를 주기적으로 받아들이기 위해서 ERT DE 내에 타이머를 만들었다. 타이머는 1ms의 주기로 돌며, 초음파 물체 감지 센서나 위치 감지 센서 등의 주기가 오면 큐

리를 만들어서 ERT DE에 쿼리를 전달하게 된다.

Normal DE와 ERT DE의 비교를 위한 테스트의 환경은 다음과 같다.

- CPU : x86 Processor (Intel 350MHZ)
- RAM : 128M
- OS : Linux (Hancom Linux 2.0)

테스트를 위한 트랜잭션은 표 6과 같이 5개로 구성이 되어있으며, 신뢰성을 높이기 위해서 8개의 Test Set을 만들었다. 주기의 기준이 되는 Test set에서의 주기와 데드라인, 실행시간은 표 7과 같다. 8개의 각 Test set은 표 7에서의 주기만을 변경한 것으로 표 7에서의 주기를 기준(1.0)으로 0.6, 0.8, 1.0, 1.2, 1.4, 1.5, 1.6, 1.8, 2.0의 비율을 가지게 된다.

(표 6) 트랜잭션 설명

트랜잭션	설명
T1	물체와 부딪쳤을 경우 최우선 순위로 정보 저장(비주기) 동기화 테이블 테이블에서 정보를 추출하여 정보를 저장하고 90° 방향을 전환하고, 계속적으로 진행한다.(INSERT)
T2	T1에서 초음파 복귀시간이 7ms일 경우(약 1m 이내 접근), 위치 감지 센서 테이블에서 바퀴의 정보를 읽고, 90° 방향 전환 후 바퀴를 동기화 시키고 동기화 테이블 갱신 (FIND, INSERT)
T3	위치 감지 센서 테이블에 바퀴의 정보 입력 (INSERT)
T4	위치 감지 센서 테이블에서 1번 바퀴의 정보를 읽고, 거리를 계산한 후 동기화 테이블에 저장 (FIND, INSERT)
T5	초음파 물체 감지 센서 테이블에 초음파의 정보 입력 (INSERT)

(표 7) 트랜잭션의 주기, 데드라인 작업시간 설명

트랜잭션	주 기	데드라인	실행시간	Priority
T1	30ms	10ms	3ms	1
T2	7ms	7ms	4ms	2
T3	5ms	5ms	3ms	3
T4	10ms	10ms	3ms	4
T5	20ms	10ms	3ms	5

※ Priority는 낮을 수록 우선순위가 높다

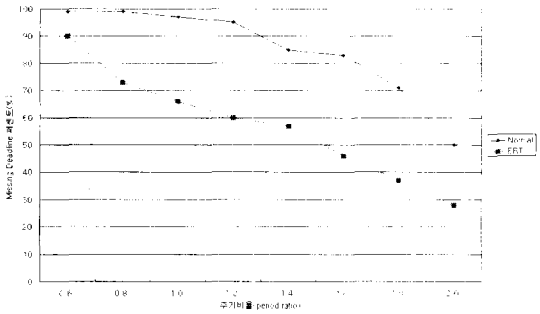
본 논문에서는 문맥교환(Context Switching), 테스트 프로그램 내의 쿼리 생성 시간 등의 오버헤드는 없다고 가정을 하였다.

프로그램의 주기를 원래 동작 주기의 1000배인 1초로 설정했으며, 따라서 5ms의 주기를 가지는 트랜잭션은 5초의 주기로 계산하였다. 트랜잭션 수행을 위한 쿼리는 130번째 주기까지 생성을 하였다.

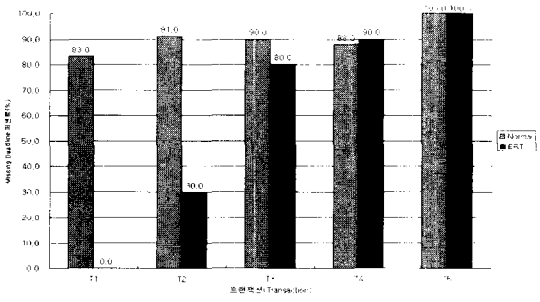
4.2 결과 및 분석

그림 2는 주기 비율별 Missing Deadline의 퍼센트를 비교해 놓은 것이다. ERT DE는 Normal DE에 비해서 데드라인을 넘길 확률이 적으며, 주기가 길어질수록 데드라인을 넘길 확률이 적어짐을 볼 수 있다.

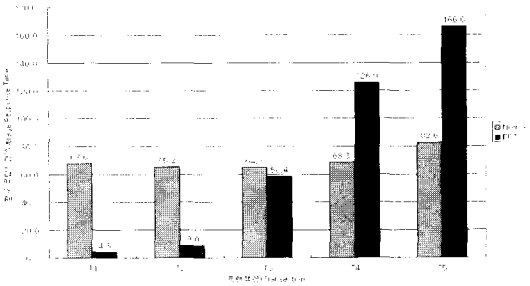
그림 3은 트랜잭션별 Missing Deadline 퍼센트를 막대그래프로 표현해 놓은 것으로 Normal DE는 평균 80%이상의 높은 Missing Deadline율을 보이고



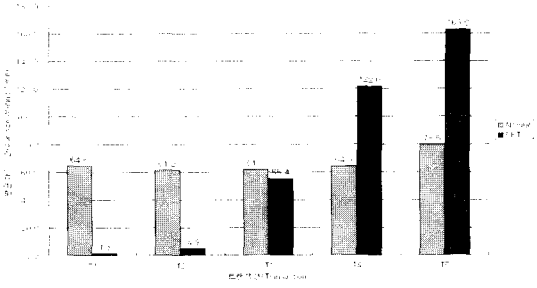
(그림 2) 주기 비율별 Missing Deadline 퍼센트



(그림 3) Missing Deadline percent



(그림 4) 트랜잭션별 평균 응답시간



(그림 5) 트랜잭션별 평균 대기시간

있다. 데드라인을 넘긴 트랜잭션에 의해 다음 트랜잭션이 영향을 받아 높은 비율을 나타내고 있다. ERT DE는 최상위 우선순위를 가진 트랜잭션은 0%, 최하위 우선순위를 가진 트랜잭션은 100%의 Missing Deadline을 가지고 있다. 최하위 우선순위를 가진 트랜잭션은 기아상태(starvation)에 빠져있다.

상위의 우선순위를 가지는 T1와 T2는 Normal DE보다 월등한 성능을 보이고 있으나 반대로 하위의 우선순위를 가지는 T4과 T5는 Normal DE보다 성능이 좋지 않은 것을 알 수 있다.

그림 4와 그림 5는 트랜잭션별 평균 응답시간과 평균 대기시간을 나타내는 그래프이다. 두 개의 그림을 비교해 보면 앞서 말한 바와 같이 Normal DE와 ERT DE 모두 대기시간에 많은 영향을 받고 있음을 알 수 있다. ERT DE에서 우선순위가 낮은 T4와 T5는 데드라인을 넘기는 작업들에 의해서 계속적으로 지연이 되어지고 T5는 기아상태로 빠져든다.

Normal DE와 ERT DE를 비교하였을 때, Normal DE는 모든 면에서 균등한 모습을 보여주고 있고, ERT DE는 데드라인이 짧은(우선순위가 높은) 트랜잭션에 대해서는 응답시간이 빠르고, 데드라인이 긴(우선순위가 낮은) 트랜잭션에 대해서는 응답시간이 느리다.

본 논문에서 예로써 사용한 로봇의 데이터는 최상위 우선순위를 갖는 트랜잭션 T1에 대해서는 데드라인 이내에 트랜잭션을 처리해 내지만, 최하위 우선순위를 가지는 트랜잭션 T5에 대해서는 기아상태(starvation)에 빠지고 있다.

그림 2 "주기 비율별 Missing Deadline 퍼센트"를 참고하였을 때 ERT DE는 본 논문에서 사용한 로봇 데이터보다 약간은 긴 주기와 데드라인을 가진 리얼타임 시스템에서는 보다 높은 신뢰성을 줄 수 있을 것이다.

5. 결론 및 향후 연구 과제

리눅스는 많은 부분에서 사용이 되어지고 있고 여러 종류의 데이터베이스 프로그램들을 가지고 있다. 하지만, 이들 프로그램들은 임베디드 리눅스에서 운용하기에는 프로그램의 사이즈가 너무 크다. 최근 여러 종류의 임베디드용 데이터베이스 프로그램들이 출시되고 있지만, 고비용의 대가를 지불해야 한다. 본 논문에서 제시한 ERT DE는 작은 사이즈의 오픈 소스를 가지고 리눅스용으로 변환을 하였고, 리얼타임 기법을 부가하여 ERT DE를 구현하였다.

본 논문에서 제시한 ERT DE는 실시간적인 응답시간을 요구하는 트랜잭션에 대해서는 좋은 성능을 보이고 있다. 하지만, ERT DE는 멀티 큐의 형식을 취하고 있고, 이러한 방식은 우선순위가 낮은 트랜잭션에 대해서는 기아상태(starvation)의 위험이 많이 노출되어있다.

향후 ERT DE가 보여준 하위 우선순위를 가지는 트랜잭션이 기아상태로 빠지는 것을 막기 위해 시분할(Time-Sharing)을 도입하여 보완할 수 있고, 그 외 태스크 레벨에서의 리얼타임 기법의 도입과 문맥교환(Context Switching), 데이터베이스의 복구

(Recovery), 동시성제어(Concurrent Control) 등은 앞으로 ERT DE가 보완해야 할 것이다.

참 고 문 헌

[1] 배방희, "인퍼노 RTOS 기반의 embedded 웹 단말기 구현 방안에 관한 연구", 한양대학교 산업대학원 공학석사논문, 1999.

[2] 이두원, "실시간 운영체제", http://www.doall.co.kr/, 1999.

[3] Margo Seltzer, Michael Olson, "Challenges in Embedded Database System Adminstration", http://www.sleepycat.com/, 2000.

[4] Jayant R. Haritsa, Miron Livny, Michael J. Carey, "Earliest Deadline Scheduling for Real-Time Database Systems", IEEE Real-Time Systems Symposium, 1991.

[5] Jayant. R. Haritsa, 'Approximate Analysis of Real-Time Database Systems', 10th International Conference on Data Engineering, pp.10~19, 1994.

[6] flyduck, "Real Time과 리눅스, 그리고 RT-Linux", http://www.flyduck.com, 1999.

[7] 이양호저, "파일처리론", 정익사, 1997.

[8] 이상엽저, "Visual C++ 6.0 Bible", 영진 출판사, 1998.

[9] RT Linux Homepage : http://www.rtlinux.org/~rtlinux.

[10] Cross Development for Linux-PPC Homepage http://members.home.net/mmporter/linux/cross.

[11] Embedded PowerPC Linux Boot Project Homepage, http://ppcboot.sourceforge.net.

[12] Montavista Homepage, http://www.mvista.com/.

[13] Motolora PowerPC Library http://e-www.motorola.com/collateral/PPCINDEX.html.

[14] 류 석, "Embedded System용 실시간 운영체제 설계 및 구현", 충남대학교 대학원 컴퓨터공학과 석사논문, 2000.

[15] 한성호, "임베디드 시스템에서 리눅스 커널을 위한 부트로더에 관한 연구" 성균관대학교 정보통신대학원 석사논문, 2001.

[16] 강경태, "내장형 시스템을 위한 리눅스 개발 및 사례분석", 서울대학교 대학원 컴퓨터공학부 석사논문, 2001.

[17] Jerry Epplin, "Embedded Systems Programming", 1997.

[18] C.M.Krishna, Kang G, Shin, "Real-Time Systems", MaGraw-Hill, 1997.

[19] Sang H. Son, "Advances In Real-Time Systems", Prentice Hall, 1995.

[20] Azer Bestavros, "Advances in RTDB Systems Research", ACM SIGMOD Record, vol.25 no 1, 1996.

[21] Michael Barabanov, "A Linux-based Real Time Operating System", 1997/06.

◐ 저 자 소 개 ◑

김 홍 섭

1998년 수원대학교 전자계산학과 졸업(학사)
2000년 수원대학교 전자계산학과 졸업(석사)
2000년~현재 : KESL(Korea Embedded System of Linux) 회장 Astonlinux 주임연구원
관심분야 : 리얼타임 시스템, 임베디드 시스템, 리눅스
E-mail : mac21c@astonlinux.com

문 승 진

1989년 1월~1990년 7월 Canopy Road Software System Analyst
1997년 5월~8월 (주) 맥시스템 부설연구소 소프트웨어 연구실장
1997년 9월~현재 : (주) 맥시스템 부설연구소 자문위원
1997년 9월~한국정보과학회 종신회원,
한국정보처리학회 종신회원,
한국인터넷정보학회 종신회원
2001년 7월~현재 : (주) 이지메딕스 기술 자문위원
2001년 7월~현재 : 수원대학교 자연과학대학 컴퓨터과학과 조교수
관심분야 : 실시간 및 멀티미디어 데이터베이스, 실시간 모바일 데이터베이스, 실시간 내장형 시스템, 실시간 리눅스
E-mail : sjmoon@mail.suwon.ac.kr