실시간 커널 : RTLINDA

박영환
한성대학교, 정보전산학부

Real-Time Kernel : RTLINDA

PARK, Young-Hwan
Information & Computer Science Division

요 약

광범위한 범위에서 사용되고 있는 실시간 시스템을 위하여 실시간 커널의 사용은 필수적이다. 본 논문에서는 다양한 분야에서 사용될 수 있는 이식성이 높고, 목적 시스템에 필요한 커널 서비스만을 선택하여 커널의 크기를 조절할 수 있는 실시간 커널의 개발하였다. 이 커널은 병렬 커널로의 확장성을 고려하여 LINDA 개념을 기초로 하여 개발되었는데, 기존에 개발된 실시간 커널 HLINDA와 비교하였을 때 보다 좋은 성능을 보여주었다. 또한 시스템 초기화 부분과 콘텍스트 스위치 등, 몇 부분을 제외한 전 커널의 95%이상을 C-언어로 작성하여 높은 이식성을 갖고 있다

1. 서 론

실시간 시스템은 공장의 조그마한 기계의 제어에서부터 지능형 자동차, 모빌 로보트, 대형 여객기의 제어나 비행 관제 시스템에 이르기까지 광범위한 분야에서 사용되고 있다.

이러한 실시간 시스템을 구성하는 소프트웨어로는 실시간 응용 프로그램과 그것을 쉽게 개발하고 실행할 수 있게 하기 위한 실시간 커널이 있다. 이러한 커널들은 그 크기나 제공하는 서비스에 따라서 아주 다양하게 개발되어 사용되고 있다. 따라서 목적하는 실시간 시스템에 알 맞는 크기와 필요로 하는 서비스 그리고 컴퓨터 하드웨어를 고려하여 실시간 커널을 선택하여야 한다. 이러한 실시간 커널 선택의 문제점과 그 커널의 가격 때문에 목표로 하는 실시간 시스템에 적당한 커널을 스스로 개발하여 사용하기도 한다.

우리 나라의 경우도 광범위한 분야에서 실시간 시스템을 사용하고 있지만 일반적으로 외국에서 개발된 실시간 시스템을 구입하고 단지 필요로 하는 실시간 응용 프로그램을 개발하여 실시간 시스템을 구성하고 있다. 결국 실시간 시스템의 크기나 그 사용목적에 따라서 여러 가지 실시간 커널을 도입하여 사용하고 있는 실정이다.

이러한 문제점을 극복하기 위해서는 우리도 안정적이며, 이식성이 높고, 그 크기를 조절할 수 있으며, 제공되는 서비스를 목적으로 하는 실시간 시스템에 맞게 제 구성할 수 있는 실시간 커널을 개발할 필요가 있다. 본 연구에서는 이와 같은 목적에 부합하는 실시간 커널 RTLINDA를 개발하였다.

이 논문은 1998년도 한성대학교 교내연구비에 의하여 일부 지원되어 연구되었음

2. LINDA

LIND는 병렬 프로그램 개발상의 어려움을 극복하고자 예일대(Yale University)의 David Delernter과 그 팀에 의하여 제안된 병렬 프로그램의 한 모델이다[1]. LINDA 개념을 통하여 프로세스(process)에서 시간적인 그리고 공간적인 문제를 분리해 냄으로써 순차적 프로그램의 개발에 들어가는 노력과 같은 정도의 노력으로 병렬 프로그램을 개발할 수 있게 되었다[1]

이 모델은 튜플(tuple)과 튜플-스페이스(tuple-space) 그리고 4개의 기본적인 프리미티브(primitive) 들로 이루어져 있다. 튜플은 각 튜플 고유의 키(key) 필드(field)와 데이터의 집합 필드로 구성되어 있다. 그리고 튜플-스페이스는 튜플 들로 구성되어 있는 일종의 공용 연상 기억장치(shared associative memory)로, 이 튜플-스페이스에서는 주소가 아니라 바로 각 튜플의 고유 키를 통하여 원하는 튜플을 찾을 수 있다. 원하는 튜플을 쓰거나 읽기 위하여 out(), eval() 과 in(), read()라는 4개의 프리미티브가 사용된다.

2.1 out()과 eval()

out(key, datas)을 통하여 하나의 튜플을 튜플-스페이스에 삽입할 수 있다. 그리고 이렇게 삽입하는 과정에서 이 튜플을

읽기 위하여 in()이나 read()를 실행시킨 프로세스가 있나 검사하여 이 들을 깨운(weak-up) 후 프로세스 레디 큐(ready queue)에 추가 시킨다 eval(key, func(e), true)은 기본적으로 out()과 같은 일을 하나, 우선 func(e)를 실행하여 그 결과가 사실이면 그 결과를 데이터로 하여 튜플을 생성하여 튜플-스페이스에 삽입 시킨다.

2.2 in()과 read()

in() 과 read()는 튜플-스페이스에서 원하는 튜플을 읽는 프리미티브들 이다. 두 프리미티브의 차이점은 read()는 튜플을 읽은 후 그 튜플을 지우지 않고 남겨놓는 반면 in()은 원하는 튜플을 읽은 후 그 튜플을 튜플-스페이스에서 지워버린다. 그리고 또 다른 두 프리미티브의 공통점은 원하는 튜플을 찾을 수 없는 경우, 그 프리미티브를 실행시킨 프로세스가 수면상태(sleep state)에 들어가고, 그 후 원하는 튜플을 삽입하는 out() 이나 eval()에 의해서 깨어나(wake-up) 원하는 튜플을 읽게된다.

2.3 장점

LINDA 개념에서 프로세스(process)로부터 시간적인 그리고 공간적인 문제를 분리해 냄으로써 순차적 프로그램의 개발에 들어가는 노력과 같은 정도의 노력으로 병렬 프로그램을 개발할 수 있다고 하였는데, 이러한 특성은 바로 튜플-스페이스와 위의 4가지 프리미티브 덕분이다. 즉 병렬처리에서 꼭 필요한 프로세스간 통신(IPC : inter process communication)을 수행하는데 있어 필요한 정보의 입,출력이 주소가 아닌 키를 매개로 이루어지므로, 여러 개의 프로세스 개발 시 데이터의 공간적인 위치 문제를 고려함이 없이 프로그램을 할 수 있다. 또한 원하는 튜플이 없으면 프로세스가 자동적으로 수면상태에 들어가고, 그 원하던 튜플이 삽입되면 또한 자동적으로 깨어나게 됨으로써 프로세스간의 동기화(inter process synchronization)를 고려하지 않고 병렬 프로그램을 개발할 수 있게 된다. 결과적으로 LINDA 개념을 통하여 병렬 응용프로그램의 개발 시 공간적인 그리고 시간적인 제약을 뛰어넘을 수 있게 된다.

2.4 실시간 병렬 커널 HLINDA

이러한 LINDA 개념에 기초한 여러 가지 시스템의 구현에 관한 연구가 진행되었으며[2][3], 이 개념을 실시간 시스템으로 확장하려는 연구도 진행되었다[4]. 이러한 연구의 결과, 분산 메모리 병렬 시스템(Distributed Memory Parallel System)을 위한 실시간 병렬 커널 HLINDA[4]가 개발되었다.

이 커널은 계층적 튜플-스페이스(Hierachical Tuple-Space)를 정의함으로써 분산 메모리 병렬 시스템에서 하나의 가상적인 튜플-스페이스를 유지하기 위한 통신 비용을 현저하게 줄일 수 있었고, 또한 프로세서간에 예측 가능한 통신을 구현할 수 있었다.

하지만 HLINDA는 튜플-스페이스를 구현함에 있어서 연결 리스트(linked list)를 사용하여 튜플의 탐색, 삽입, 삭제를 실행하게 하였는데, 사용자의 입장에서는 이 리스트의 길이가 가변적이므로, 그 처리 시간이 결정적(deterministic)이지 못하다는 문제점을 갖고 있다 또한 LINDA 개념서부터 존재하는 read()와 in() 의 처리 역전에 따른 데드락(dead-lock) 문제를 그대로 갖고 있다.

3. RTLINDA

본 논문에서는 HLINDA의 장점을 살리고 HLINDA의 문제점을 해결할 수 있는 RTLINDA를 제안한다.

우선 RTLINDA에도 계층적 튜플-스페이스를 도입하여 통신의 비용을 절감하였다. 또한 read() 와 in()을 통합하여 하나의 프리미티브 in()으로 대체함으로써 read() 와 in() 의 처리 역전에 따른 데드락(dead-lock) 문제를 해결하였다.

이러한 RTLINDA는 프로세스 매니저, 스케줄러, 메모리 매니저, 그리고 IPC(Inter Process Communication : 프로세스간 통신부) 등으로 이루어졌다.

3.1 프로세스 매니저

RTLINDA는 주기적 프로세스(periodic process)와 비 주기적 프로세스(non-periodic process), 그리고 스포라딕 프로세스(sporadic process)가 존재한다 이러한 프로세스들을 제어하기 위하여 다음과 같은 프리미티브들이 제공된다.

```
create_prs(int prs_adrs, int priority, int period, int deadline(offset));
delete_prs(int pid);
suspende_prs(int pid, int key, void **data_in); (by in() operation)
sleep(int key, int period),
resume_prs(int key, void **data_in); (by out() operation)
msignal(int tuple_space_hierarchy, int key),
```

3.2 IPC

RTLINDA에서 IPC는 정보의 교환과 프로세스간 동기화를 위하여 사용된다. 이러한 IPC를 위하여 두 가지의 커널 프리미티브 out() 과 in()이 사용자에게 제공되는데 그 형식은 다음과 같다.

```
out(int ts_hierarchy, int key, int in_counter, void **data, int data_size);
in(mt key, void **data),
```

3.3 메모리 매니저

메모리를 운용하는 여러 가지 알고리즘이 개발되어 사용되고 있는데[5] 본 커널에서는 처리 속도에 있어서 가장 효율적인 버디(Buddy) 알고리즘을 사용하였다. 물론 이 알고리즘은 제한된 메모리를 갖게되는 삽입 실시간 시스템(Embedded Real-Time System)에서는 공간 사용에 있어서 메모리의 적지 않은 부분이 낭비되는 문제점을 안고 있지만, 빠른 메모리 운용 요구를 충족시키기 위하여 위의 방법을 사용하였다.

3.4 스케줄러

실시간 시스템을 위하여 여러 가지 스케줄링 알고리즘이 개발되어 이용되고 있는데, RTLINDA에서는 EDF(Earlist Deadline First)알고리즘을 기본으로 프로세스의 종류에 따라 스케줄링 하는 방법을 달리 함으로써 아주 효율적인 스케줄링 알고리즘을 정의하였다.

우선 주기적 프로세스(periodic process)는 가장 높은 우선순위 0을 갖으며, 프로세스 종료 시한인 데드라인(deadline)이 없다. 그리고 프로세스 생성 시 주어진 offset 값과 주기에 따라 프로세스의 시작 순간이 결정된다. 한 순간에 활성화 되어야할 주기적 프로세스가 2개 이상일 경우, 먼저 생성된 프로세스가 우선 순위가 높다.

또한 RTLINDA에서는 주기적 프로세스의 변형으로서 스포라딕 프로세스(sporadic process)를 정의하였다. 즉, 주기적 프로세스 중 그 주기가 무한대인 것은 한 번 실행된 후에 스스로는 다시 실행될 수가 없다. 이러한 프로세스는 어떤 다른 프로세스나 신호에 의해서 그 주기 카운터의 값이 변경됨으로써 실행될 수 있다. 따라서 이러한 프로세스는 스포라딕 프로세스로서의 역할을 할 수 있다.

비 주기적 프로세스에는 데드라인 이 있는 실시간 프로세스와 데드라인이 존재하지 않는 비 실시간 프로세스가 존재한다. 이 중 실시간 프로세스는 우선 순위 1 인 프로세스이며 이들은 EDF 알고리즘에 의하여 스케줄 되고, 비 실시간 프로세스는 우선 순위가 2 에서 9 까지 이며, 이들은 라운드 로빈(round robin) 방식에 의해서 스케줄 된다.

3.5 RTLINDA의 특징과 성능

RTLINDA의 기본 설계 이념은 빠르고 작은 커널의 구현이었다. 이를 위하여 우선 프로세스 테이블(process table)의 IPC에 사용되는 튜플-스페이스의 크기를 사용자가 커널이 구동되기 전에 결정하며 이 크기는 커널이 정지할 때 까지 유지된다 이와 같은 설계 방식을 통하여 프로세스의 생성과 소멸, 그리고 튜플의 삽입과 삭제에 따른 메모리 관리 시간을 줄일 수 있다.

또한 튜플-스페이스이 구조를 배열을 사용하여 구성하였기 때문에 배열의 인덱스가 튜플의 키 역할을 하게 함으로써 빠른 튜플-스페이스의 조작이 가능하다

그리고 모든 커널 서비스의 구동 시간이 예측 가능하거나 어느 한계 이하에 존재함으로써(표 1) 실시간 시스템 구성에 꼭 필요한 예측 가능성을 충족시키고 있다(ADSP2106x, 33MHz에서 10 개의 주기적 프로세스와 3개의 비 주기적 프로세스 그리고 3개의 튜플이 있는 시스템에서의 평균 클럭 사이클 수와 평균 수행 시간). 이와 같은 특성을 갖고있는 RTLINDA를 HLINDA와 비교하여 그 성능의 우수성을 알아보았다(RTLINDA는 TMS320C40에서의 클럭 사이클 수)(표 2)[6].

커널 서비스	명령어 사이클 수	시간(μsec)
create_prs	215.8	6.54
delete_prs	205.8	6.24
suspend_prs	157.5	4.77
resume_prs	227.5	6.89
out	245.7	7.45
in	78	2.36

표 1. RTLINDA 커널의 성능

커널 서비스	RTLINDA	HLINDA
create_prs	215.8	1177
suspend_prs	157.5	695
resume_prs	227.5	187
out	245.7	5029
in	78	6418

표 2. HLINDA와의 성능 비교

4. 결론

LINDA와 HLINDA 장점을 이용한 실시간 커널 RTLINDA를 구현하였다. RTLINDA를 통하여 LINDA의 기본 개념을 확장, 실시간 커널에 원용하였고, HLINDA의 문제점 들을 해결할 수 있었다. 그리고 커널을 구현함에 있어 사용상의 몇 가지 유연성들을 제한함으로써 기존 HLINDA 보다 우수한 성능을 발휘할 수 있게 되었다. 앞으로의 연구에서는 본 커널에 내재되어 있는 병렬성을 발휘할 수 있도록, 실시간 병렬 커널에로의 확장을 위하여 IPC 부분과 프로세스 동기화 부분을 확장하고 2개 이상의 프로세서로 구성된 병렬 컴퓨터에서 안정성과 효율성을 검증하고자 한다.

참고문헌

[1] S.Ahuja, N.Carriero, and D.Gelernter, "Linda and Friends", Computer, Vol. 19, No. 8, Aug. 1986

[2] S Ahuja, N.Carriero, D.Gelernter, and V.Krishnaswamy "Matching Language and hardwarefor Parallel Computation in the Linda Machine", IEEE Trans. on Comp, Vol 37, No.8, Aug. 1988

[3] D.Bakken, and R Schlichting "Supporting Fault-Tolerant Parallel Programming in Linda", IEEE Trans. on PDS, Vol 6, No 3, Mar. 1995.

[4] E Yao, B Jardin, Y H Park, and K.M Hou, "Real-Time Multiprocessor Kernel Hierarchical LINDA", 4th Int Conf on Signal Processing Application and Technology, Santa-Clara, California, USA, Oct, 1993

[5] A.Tanenbaum, "Operating System : Design and Implementation", Prentice Hall International, 1987.

[6] E.Yao, "HLINDA Un noyau temps reel multiprocesseur", 박사학위논문, Universite de Technologie de Compiegne, France, 1996.