Dual Core 시스템에서 효율적인 공유 메모리 사용 기능 구현

장승주*
*동의대학교 컴퓨터공학과
e-mail:sjjang@deu.ac.kr

Implementation of the Efficient Shared Memory in the Dual Core System

Jang, Seung Ju*
*Dept of Computer Engineering, Dong-eui University

요 약

본 논문은 Linux에서 사용되는 Shared Memory는 동일한 메모리 영역에 여러 개의 프로세스가 접근할 수 있도록 해 주는 기술이다. 본 논문에서는 Shared Memory의 큰 두 갈래 중 커널 단계에서 처리 되는 SVR(System V Release) 형식의 Shared Memory를 다룬다. 본 논문에서는 리눅스 운영체제의 공유 메모리 기능을 Dual Core 시스템에서 동작하도록 구현한다.

1. 서론

본 논문에서는 SVR IPC에서 사용되는 기술 중 공유 메모리(Shared Memory)를 Dual Core 시스템 상에서 동작하도록 설계한다.

복잡한 프로그래밍 환경에서 다수의 프로세스들은 서로 협력하기 위하여 서로 통신하고 자원과 정보를 공유한다. 커널에서는 이것이 가능한 방법을 제공하는 데, 이를 프로세스간 통신(Inter-Process Communication) 또는 IPC라 부른다 [1-3]. 프로세스간 통신을 하는 목적은 데이터 전송, 데이터 공유, 사건 전송, 자원 공유, 프로세스 제어를 하기 위함이다. 이를 위하여 전통적인 Unix에서는 시그널(Signal), 파이프(Pipe), 프로세스 추적(Process Tracing)의 기능을 사용하였다. 그러나 많은 사용자 프로그램들에서는 이것만으로 IPC를 충족할 수가 없었으며, 이것은 System V 릴리즈 버전에서 메시지 큐, 세마포어, 공유 메모리 기법을 사용하게 된다 [4].

Shared Memory 기법은 시스템의 가상 메모리(Virtual Memory) 구조에 크게 의존하며, 데이터의 복사나 시스템 요청을 사용하지 않고 많은 양의 데이터를 공유하는 매우 빠르고 융통성 있는 기법을 제공한다. 그러나 이 기법은 동기화 기법을 제공하지 않는다는 단점이 있다. 두 개의 프로세가 있다고 가정하면, 공유 메모리 영역의 내용을 동시에 변경하려고 할 경우, 커널은 이를 순차적으로 처리를 해야 하나, 그렇게 하지 않으면, 쓰인 데이터는 엉키게 된다. 그래서 공유 메모리를 사용하는 프로세스들은 프로세스들 간에 동기화를 하는 프로토콜을 고안해야 하며, 이것은 공유 메모리 성능에 영향을 주는 요소로 적용할 수 있다.

2. Dual Core 시스템에서 동작하는 Shared Memory 기능 구현

Shared Memory는 물리적인 메모리의 특정 영역에 대하여 프로세스들에 의해 공유되어지는 공통의 영역을 구성한다. Shared Memory는 공통의 영역을 프로세스들이 사용할 수 있게 해준다.

그림 1. Shared Memory 구조

그림 1.은 Shared Memory의 구조를 나타낸다. Shared Memory는 물리적인 메모리의 특정 영역에 대하여 프로세스들에 의해 공유되어지는 그림 1.의 Critical Section(임계 영역)이라 할 수 있다. 프로세스들은 이 Shared Memory 영역을 자신의 가상 메모리 영역에 부착(Attach)하여 사용한다. Critical Section은 데이터 읽기/쓰기의 시스템 호출을 사용하지 않고 다른 방법으로 접근하기 때문에 프로세스 간 데이터를 공유하는 기법 중 가장 빠르다 할 수 있다. 예를 들어, 하나의 프로세스가 공유 메모리의 영역에 기록(Write)을 하게 된다면, 이 내용은 이 영역을 공유하는 모든 프로세스들에 의해 바로 보여 지게 되어 아주 빠르게 데이터를 공유하게 된다.

그림 2. 2 CPU에서의 IPC 구조

그림 2.는 2 CPU에서의 IPC(본 논문에서는 Shared Memory)의 구조이다. 서로 다른 CPU에 서로 다른 운영체제가 IPC 메카니즘 중 Shared Memory 기법으로 서로의 데이터를 공유하는 모습이다.

그림 2.와 같이 서로 다른 CPU를 사용할 경우에 각각의 CPU에서 Shared Memory를 통하여 데이터를 공유하거나 정보를 주고받을 수 있다. 또한 서로 다른 CPU에서 동시에 다른 프로세스가 수행될 수 있다. 프로세스들의 수행양이 많을 경우 단일 프로세서에서는 프로세스 수행에 시간이 많이 소용된다. 이는 단일 프로세서는 프로세스들을 순차적으로 처리하기 때문이다. 한 프로세스가 CPU 사용의 최대 시간(time slice)을 넘을 정도로 긴 수행을 해야 하는 경우라 가정한다. 단일 프로세서는 CPU 사용의 최대 시간을 사용한 프로세스1에 대해서 CPU 사용을 중지 시키고 프로세스1은 Sleep 상태로 들어서고, 다음 사용자인 프로세스2가 CPU를 할당 받는다. 이럴 경우 프로세스 수행 시간의 거의 2배 가까운 시간이 소요되어서야 프로세스의 수행이 종료 될 수 있다. 하지만 그림 2.와 같이 두 개의 CPU를 사용하게 된다면, 혹은 두 개의 CPU를 사용하는 것과 같은 듀얼 코어 프로세서를 사용하게 된다면 두 개의 프로세스를 동시에 수행할 수 있어 성능 향상에 도움이 된다.

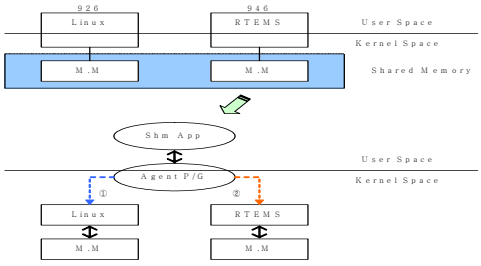

그림 3. Dual Core에서 Shared Memory 설계

그림 3.은 Dual CPU에서 서로 다른 운영체제, ARM 926은 Linux 운영체제를 ARM 946은 RTEMS 운영체제를 사용할 경우 Shared Memory를 구현하기 위한 구상도 이다.

운영체제가 서로 다른 경우 사용하는 Shared Memory의 구조가 다를 수 밖에 없다. 이러한 문제점을 보완하고자 그림 3.의 "Agent Program"을 도입하였다. "Agent Program"은 그림 3.의 ①, ②와 같이 서로 다른 운영체제에서 생성한 Shared Memory로의 링크를 행해준다.

그림 3.의 ①은 Linux 운영체제에서Shared Memory를 생성하였다. Linux 운영체제의 관리 아래에서 동작하고 있는 Shared Memory를 RTEMS 운영체제에서 필요 하게 될 경우 Agent Program에서 그림 3.의 ①과 같은 링크를 생성하여 Linux 상의 Shared Memory를 사용할 수 있게 해준다.

그림 3.의 ②는 그림 3.의 ①과 반대로 RTEMS 운영체제에서 생성한 Shared Memory에 Linux 시스템에서 동작하는 프로그램이 접근하고자 할 경우 그림 3.의 ②와 같은 링크가 생성되어 RTEMS 운영체제 상의 Shared Memory를 사용할 수 있게 해준다.

설계 방법은 두 개의 코어 중에서 master와 slave를 지정한다. 공유 메모리 기능의 구현은 agent를 통해서 이루어진다. Agent내 master node의 기능을 이용해서 slave의 기능을 이용한다. Master와 slave간에 데이터 전송 방식은 RPC 메커니즘을 이용한다.

Dual Core의 서로 다른 CPU에서 독립적으로 갖고 있는 Main Memory에서 할당되는 Shared Memory 정보를 저장하기 위해 새로운 자료구조를 만들었다.

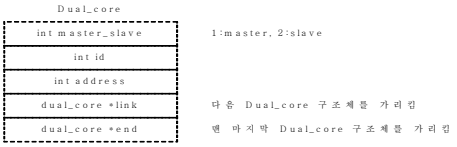

그림 4. 새로운 자료구조

그림 4.는 Shared Memory 정보를 저장하기 위한 새로운 자료구조이다. 자료구조의 중요부분은 master와 slave를 구분할 수 있는 변수인 master_slave 변수와 link, end 변수이다.

master_slave 변수의 경우 그 값이 '1'이면 master, 즉 원래의 운영체제인 Linux 운영체제에서 생성되고 관리되어 진다는 의미이다. master_slave 변수의 값이 '2'이면 slave, 즉 원래의 운영체제가 아닌 다른 운영체제인 RTEMS 운영체제에서 생성되고 관리되어 진다는 의미이다.

3. 결론

본 논문에서 제안하는 Dual Core 시스템에서 공유 메모리 기능 설계는 듀얼 코어를 활용하여 기존의 단일 처리기 시스템에서보다 성능을 향상시킬 수 있다. 공유 메모리를 이용한 프로세스의 동작이 별개의 CPU에서 동작되도록 함으로써 성능 향상을 꾀한다. 본 논문은 대부분의 유닉스 시스템에서 사용되는 IPC 메카니즘 중 Shared memory의 기능을 Dual Core 시스템에서 동작할 수 있도록 구현했다..

참고문헌

[1] 김선욱, 오재근, 한영선, 최홍욱, 김철우, "임베디드 응용 프로그램 성능 분석", 대한전자공학회:학술대회지, 대한전자공학회 03년도 하계종합학술대회 논문집III, pp. 1355-1358, 2003.
[2] 박재호, "임베디드 리눅스", 한빛미디어, 2002.11.[3] Uresh Vahalia, "UNIX Internals", 홍릉과학출판사, 2001
[3] 박윤미, 성영락, "임베디드 리눅스시스템 설계 및 구현", 한국정보과학회2003 봄 학술발표논문집, pp. 214-216, 2003[4] 한동훈, "시스템V IPC - 공유메모리", 1997
[4] 김용준, "공유메모리를 이용한 채팅방의 원리"
[5] 한동훈, "공유메모리 vs 세마포어를 이용한 chat program", 1997