M3K 에서 IPC 컴포넌트 설계 및 구현

김 영호° 고 영웅 유 혁
고려대학교 컴퓨터학과

{yhkim, yuko}@os.korea.ac.kr, hxy@joy.korea.ac.kr

Design and Implementation of IPC Component in M3K

Young-ho Kim°, Young-woong Ko, Chuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

M3K(MultiMedia MicroKernel)는 멀티미디어에서 요구하는 실시간 특성을 지원할 수 있는 것을 목표로 하고 있으며, 이를 위해서 마이크로 커널 구조로서 설계되었다. 마이크로 커널은 커널 내부에서 발생하는 지연 시간이 작고 예측 가능하므로 실시간 시스템을 지원하기에 적합하다. 그러나 서버간의 빈번한 메시지 교환에 따른 IPC 병목현상은 전체 시스템의 성능을 저하시키고, 외부 이벤트에 대한 실시간 처리를 어렵게 한다. 본 연구에서는 M3K 에서 실시간 특성을 지원할 수 있도록 IPC 를 설계 및 구현하는 것을 목표로 하고 있다. 이에 대한 접근방법으로서 IPC 중에 발생되는 쓰레드 간의 문맥 전환을 소프트웨어적으로 구현하고, IPC 를 우선 순위가 부여된 시그널 객체를 이용하여 처리하고 있다. 따라서 빈번하게 발생되는 문맥 전환의 비용을 최소화함으로써 캐쉬 미스 및 TLB 미스를 줄이고, 우선 순위가 높은 이벤트나 IPC 부터 처리될 수 있게 한다.

1. 서론

마이크로 커널 구조를 갖는 운영체제에서는 대부분의 운영체제 서비스가 서버를 통해서 제공된다. 예를 들어 네트워크 인터페이스를 통해 데이터가 들어오면 먼저 디바이스 드라이버를 제공하는 서버에서 커널 영역으로 데이터를 복사하고, 이후에 이 데이터를 필요로 하는 다른 서버 영역으로 복사하게 된다. 따라서 커널이 수행되는 동안 커널과 서버 그리고 서버와 서버 사이에는 빈번한 메시지 교환이 발생하게 된다. 그러므로 마이크로 커널 구조에 있어서 IPC(Inter Process Communication)의 성능은 전체 시스템 성능을 결정짓는 중요한 요소가 된다.

IPC 성능을 최적화하기 위해서 본 논문에서는 다음과 같은 두 가지 방법을 도입하고 있다. 첫째, IPC 중에 발생되는 쓰레드간의 문맥 전환을 소프트웨어적으로 구현함으로써 빈번하게 발생되는 문맥 전환의 비용을 최소화한다. 둘째, IPC 를 우선 순위가 부여된 시그널 객체를 이용하여 처리함으로써 보다 우선 순위가 높은 이벤트나 IPC 부터 처리하게 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존의 다른 운영체제에서 IPC 성능 향상을 위해 연구된 결과를 설명한다. 3 장에서는 본 연구에서 도입한 IPC 개념을 M3K 커널에 구현할 때의 내부구조와 방법에 대해서 설명하고, 마지막으로 4 장에서는 결론과 향후 연구과제에 대해서 기술한다.

2. 관련 연구

마이크로 커널 구조에서 발생하는 성능상의 문제점을 해결하기 위해서 기존에 접근되었던 방법을 살펴보면 크게 두 가지 방법으로 요약할 수 있다. 첫번째 방법은 자주 사용되는 서버의 기능을 마이크로 커널 내부에 두는 것으로서 Mach 3.0[5]에서는 AFS 캐쉬 서버를, Amoeba[4]에서는 파일 서버를 커널 내부에 두어서 성능을 향상시켰다. 이밖에 Exokernel[1]과 SPIN[2]은 서버의 기능을 모듈로 작성한 후, 필요에 따라 사용자 영역에서

본 연구는 한국과학재단의 특정기초 연구과제 연구비 지원에 의한 결과임 (과제번호 97-01-00-09-01-3)

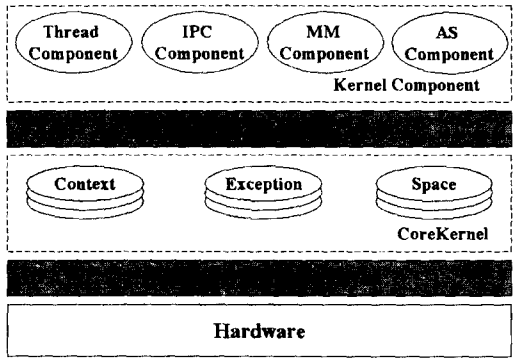
커널 내부로 이동시켜서 동적으로 수행시키는 방법을 사용한다. 따라서 이 방법은 마이크로 커널이 모노리틱 커널처럼 크기가 커지는 것을 막는 장점을 가지게 된다.

두 번째는 소프트웨어 및 하드웨어 아키텍처의 특성을 이용한 기법으로써 인터페이스, 알고리즘, 코딩 단계에 있어서 가능한 방법을 모두 사용해서 최적화 시키는 방법이다. L4/L3[3]에서는 캐쉬 미스와 TLB 미스를 줄이고, 레지스터를 최대한 활용하는 방안, 메시지 복사 회수를 줄이는 방법 그리고 lazy scheduling 과 같은 방법을 사용함으로써 IPC 의 성능을 향상시키고 있다.

3. M3K 에서의 IPC 설계 및 구현

3.1 M3K 아키텍처와 IPC 컴포넌트의 특성

M3K[6][7]는 멀티미디어를 지원하기 위해 제작된 마이크로 커널로, 하드웨어를 추상화한 코어커널과 이들이 제공하는 인터페이스를 이용하여 실제 커널 서비스를 제공하는 커널 컴포넌트로 이루어 진다. 그림 1 에서 보는 바와 같이 코어 커널은 실행 문맥, 주소 공간 그리고 예외 상황을 처리하기 위해 각각 문맥(context), 주소(space), 예외(exception) 객체로 이루어져 있다. 또한 커널 컴포넌트에는 커널 내에서 필요한 메모리를 동적으로 관리하는 메모리 관리 컴포넌트(MM component), 쓰레드가 실행될 가상 주소공간을 관리하는 주소공간 컴포넌트(AS component) 그리고 IPC 를 관리하는 IPC 컴포넌트(IPC component) 등이 있다.



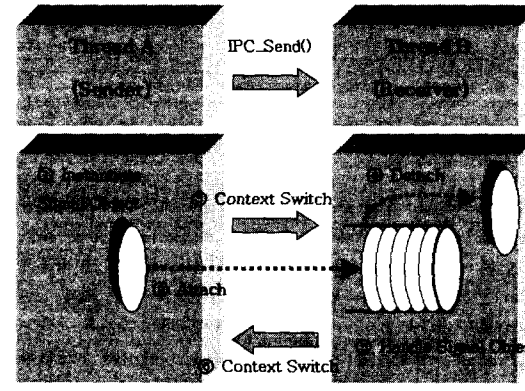
[그림 1] M3K 커널 모델

IPC 컴포넌트는 메시지 패싱(message passing) 방식에 의해서 수행되며, IPC 에 참여하는 쓰레드간에 버퍼를 사용하지 않는 동기식(synchronous) IPC 이다.

3.2 IPC 내부 구조 및 동작 방식

M3K 에서 IPC 처리 과정은 크게 IPC 계층과 시그널 계층으로 나누어진다. 첫번째 IPC 계층은 IPC 요청을 처리하기 위해서 필요한 시그널 객체를 생성한 후 시그널 계층으로 전달하는 작업을 한다. 이때 한번의 IPC 호출은 여러 번의 시그널 객체를 생성시킬 수 있다. 두 번째 시그널 계층은 우선순위를 갖는 시그널 객체를 대상(target) 쓰레드에 전달하거나, 전달된 시그널 객체를 처리하는 일련의 작업을 수행한다.

이때 시그널 객체는 시그널, 시그널을 처리하는 핸들러 그리고 시그널 송신 쓰레드에 대한 정보를 포함하고 있다. 이러한 시그널 객체의 전달은 해당 시그널 객체를 송신 쓰레드가 생성해서 수신 쓰레드의 스택에 저장함으로써 이루어진다. 그리고 이렇게 전달된 시그널 객체는 수신 쓰레드가 시그널 객체에 포함된 핸들러를 호출함으로써 시그널에 대한 처리를 마치게 된다. 모든 쓰레드에는 이러한 시그널 객체들을 저장하기 위한 스택을 가지게 된다. 이 때 스택 구조를 사용하는 이유는 외부로부터 전달된 시그널 객체의 우선순위를 유지해서 보다 우선순위가 높은 쓰레드로부터 온 시그널을 처리하기 위해서다. 그림 2 는 IPC 가 처리되는 일련의 과정을 나타내고 있다.



[그림 2] IPC 처리 과정

그림 2 에서 설명하고 있는 IPC 처리과정은 다음과 같은 단계로 수행된다.

1. IPC 인터페이스를 호출한다.
2. 호출된 인터페이스에서는 시그널 객체를 생성(①)하고 이를 대상 쓰레드의 시그널 스택에 저장(②)한다.
3. 대상 쓰레드로 문맥을 전환(③)한다.
4. 문맥이 전환된 대상 쓰레드는 자신의 시그널 스택에서 가장

위에 있는 시그널을 가져온다.(④)

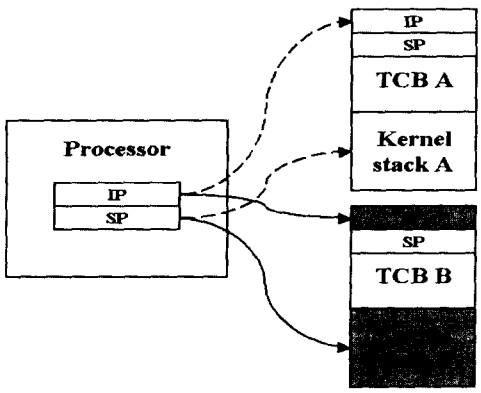
5. 가져온 시그널에 해당하는 핸들러를 호출(⑤)한다. 이 때 시그널은 임계영역에서 처리되므로 한 쓰레드가 동시에 서로 다른 시그널을 처리할 수는 없게 된다.

6. 방금 처리된 쓰레드 보다 우선 순위가 높은 시그널이 스택에 존재하는지 조사한다. 만약 우선 순위가 높은 시그널이 스택에 존재하면 4 번부터 위 과정을 반복한다.

7. IPC 를 호출한 쓰레드로 문맥 전환(⑥)을 한다

위의 과정에서 알 수 있듯이, 외부 이벤트로 발생된 IPC 의 응답 시간을 최소화하기 위해서는 이벤트 소스의 우선 순위를 변경하면 된다. 또한 위의 과정에서 하나의 IPC 를 완전히 수행하기 위해서는 최소한 두 번의 문맥전환이 이루어지는 것을 알 수 있다. 따라서 쓰레드간의 IPC 는 발생된 IPC 의 수보다 많은 문맥 전환을 초래하게 된다.

이러한 문맥 전환 비용을 줄이기 위해서 M3K 에서는 커널 레벨 쓰레드와 소프트웨어적인 문맥 전환을 구현하고 있다. 커널 레벨 쓰레드는 커널의 주소 공간을 공유하는 독립적인 실행 개체이며 자신만의 커널 스택과 IP(Instruction Pointer)를 갖는다. M3K 에서 IPC 의 처리는 이러한 커널 쓰레드간의 시그널 객체 전달과 문맥 전환을 통해 이루어 진다. 이 때 문맥 전환이 하드웨어적으로 일어난다면 오히려 TLB 미스와 같은 불이익을 초래할 수 있다. 즉 커널 쓰레드간의 문맥 전환은 같은 주소 공간에서 처리될 수 있기 때문에 다른 사용자 주소 공간으로의 전환은 불필요하다. 따라서 커널 쓰레드들의 문맥 전환을 소프트웨어적으로 구현함으로써 IPC 성능을 향상시킬 수 있다.



[그림 3] 쓰레드 문맥 전환 (A→ B)

그림 3 을 보면, 커널 쓰레드 A 가 쓰레드 B 로 문맥 전환을 하기 위해서 먼저 쓰레드 A 의 IP 와 커널 스택 포인터를 TCB 에 저장하고 전환될 쓰레드 B 의 TCB 로부터 새로운 IP 와 커널 스택 포인터 값들을 프로세서 레지스터로 적재하고 있다. 따라서 하드웨어적인 문맥 전환을 하지 않고도 쓰레드 A 의 문맥에서 쓰레드 B 의 문맥으로 전환해서 작업 할 수 있다.

4. 결론 및 향후 연구과제

IPC 는 마이크로 커널의 성능을 결정하는 중요한 요소가 된다. 이에 본 논문에서는 IPC 컴포넌트를 설계함에 있어서 성능을 최적화하기 위하여 소프트웨어적인 문맥 전환과 우선 순위가 부여된 시그널을 도입하였다. 따라서 IPC 컴포넌트는 문맥전환에 따른 TLB 미스와 같은 오버헤드를 줄일 수 있으며, 긴급한 외부 이벤트 및 IPC 요청에 대해서 응답 시간을 최소화 할 수 있다.
앞으로 기존에 알려진 IPC 최적화 방안을 M3K 아키텍쳐에 적용시키고, 기존에 마이크로 커널 IPC 성능 향상을 위해 연구되었던 결과(L4/L3, Exokernel, SPIN)와 성능을 비교 평가하는 연구를 진행할 예정이다.

참 고 문 헌

[1] D.R. Engler, M.F. Kaashoek, and J. O'Toole Jr, Exokernel: and operating system architecture for application-specific resource management, In Proceedings of the 15th ACM Symposium on Operating Systems Principles, pp. 251-266, December, 1995.

[2] Brian Bershad, Craig Chambers, Susan Eggers, and et al. SPIN – an extensible microkernel for application-specific operating system services, Technical Report 94-03-03, Dept. of Comp. Sci. and Eng., University of Washington, Seattle, February 1994.

[3] Jochen Liedtke, On microkernel construction, In Proceedings of the 15th ACM Symposium Operating System Principles (SOSP) (Copper Moutain Report, Colo., Dec. 1995). ACM Press, New York, 1995. pp 237-250.

[4] A.S. Tanenbaum, M.F. Kaashoek, R.van Renesse, and H. Bal, The Amoeba Distributed Operating System – A Status Report, Computer Communications, vol. 14, pp. 324-335, July/Aug. 1991.

[5] M. Acceta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, M. Young, MACH: A New Kernel Foundation for UNIX Development, In Proceedings of USENIX, summer 1986.

[6] 양순섭, 고영웅, 조유근, 신현식, 최진영, 유혁, "컴포넌트 기반 커널을 위한 프레임워크" 춘계 정보과학회 학술대회 논문집, 1999.

[7] 김영호, 고영웅, 유혁, "M3K 에서 쓰레드 컴포넌트 구현" 추계 정보과학회 학술대회 논문집, 1999.