□ 기술해설 □

마이크로커널 기술에 대한 고찰

한국전자통신연구소 최효진* · 임기욱**

목 차

1. 서 론

일반적으로 컴퓨터 시스템은 응용 프로그램, 운영체제, 하드웨어의 세 가지 기능적 계층으로 구성된다. 응용 프로그램은 알고리즘에 따라 사용자가 필요로 하는 기능을 수행하며, 운영체제에게 시스템 자원들의 액세스를 빈번히 요청한다. 또한, 운영체제는 하드웨어 기능을 기반으로 프로그램들의 동시 작업과 협동 처리를 지원하고 프로그램 사이 변환 등을 수행한다. 이러한 컴퓨터 운영체제는 1950년대 중반에 IBM 701 시스템용으로 처음 만들어진 이후로 응용 측의 요구 및 하드웨어 기술의 발달에 따라 다양한 수준의 기술로서 꾸준히 변천되고 있다.

운영체제의 현재 가장 일반적인 구조는 통합(integrated) 커널 또는 단일(monolithic) 커널 형태이다. 통합 커널은 응용 프로그램에 대한 시스템 서비스 및 제어와 하드웨어에 대한 제어를 수행하는 기능들이 하나의 프로그램 형태로 구현되어 동일 주소공간에서 수행된다. UNIX, Windows, OS/2, DOS 등 현재 사용되고 있는 대부분의 운영체제들이 통합 커널 구조에 해당된다. 이러한 구조에 대응되는 형태로서 마이크로커널 구조가 있다. 기존 운영체제 커널의 기능중 가장 필수적인 것들만 분리하여 작은 핵부분을 만들고 컴퓨터의 최고 권한 모드에서 수행한다. 나머지 부분은 일반적으로 사용자 모드에서 수행되는 응용의 집합 형태로 제공된다. 현재는 마이크로커널 구조가 널리 확산되고 있는데, 이러한 배경에는 운영체제에 대한 다음과 같은 다양한 요구사항들에 부응하기 위함이다.

오늘날 시스템 제작자 및 통합자들이 해결해야 할 주요 기술적 과제는 현재의 개방 시스템 환경을 혼란시키지 않는 형태로 발전시켜서 보다 새롭고 복잡한 기능적 요구사항과 급속히 발전되고 있는 하드웨어 기술을 수용하는 것이다[1]. 기능적 요구사항들은 기존의 데스크 톱 및 클라이언트-서버형의 지원 외에 고도의 병렬성, 클러스터 시스템, 고가용성 및 고장 감내성, 실시간 처리, 객체지향 환경, 내장형 제어부터 대규모 병렬 분산 응용까지 일관성 있는 응용 프로그래밍 인터페이스로 확장성 등의 지원을 포함한다. 이와 같은 기능들을 다양한 하드웨어 플랫폼에 신속히 수용하려면 소프트웨어 공학적 접근이

*정회원
**종신회원

필요하다[2]. 즉, 증가되는 운영체제의 복잡도 문제를 해결하기 위하여 잘 정의된 모듈 구조에 바탕을 둔 단순성, 이식성, 확장성, 유지 보수성의 달성을 필요로 한다.

또한, 이와 같은 기술적인 사항들을 적은 비용으로 신속히 채용하여 시장에 제품을 내어놓을 수 있게 하는 운영체제의 발전 방향도 중요하다.

세계 주요 업체들의 운영체제 기술의 개발 동향을 보면 여태까지 계속되어온 운영체제 구조에 대한 논쟁의 기반들은 미묘하게 바뀌고 있다[3]. 마이크로소프트, IBM, 노벨사의 USG (Unix Systems Group), OSF(Open Software Foundation), 썬 마이크로시스템즈 등등의 업체들이 마이크로커널, 객체, 다중 개성(personalities)이라는 공통의 주제들에 급속히 집중되고 있다. 다시 말해서, 마이크로커널이라는 작은 커널 위에 객체 지향 서비스와 에뮬레이션 서브시스템 즉, 여러 개성들의 동시 지원 여부에 대한 논쟁은 더 이상 없어지고 있다. 모든 업체들이 그렇게 하기 때문이며, 문제는 이러한 유형으로 운영체제를 구축할 것인가에 대한 여부가 아니라 어떻게 올바르게 할 수 있는 가이다.

본 논문은 다음과 같이 구성된다. 2장에서는 마이크로커널 구조의 특징, 기존 통합 커널에 비교한 장단점, 마이크로커널을 기반으로 하는 운영체제의 사례와 설계시 제기되는 문제들을 살펴본다. 운영체제를 개발하는 여러 업체에서 채택하고 있는 마이크로커널의 기본 구조는 CMU(Carnegie Mellon University)의 Mach와 Chorus Systems사의 CHORUS Nucleus에 기반을 두고 있는데, 3장과 4장에서 각 마이크로커널의 특징과 함께 각 커널을 채택한 상용 운영체제에 대하여 기술한다. 마지막으로, 5장의 결론에서는 마이크로커널의 바람직한 연구 방향을 기술한다.

2. 마이크로커널 기반 운영체제 개념

대부분의 운영체제 개발 업체들은 통합 커널로부터 마이크로커널 구조로 옮겨가고 있다[1, 3]. 마이크로커널은 운영체제의 핵이 되는 최소 커널이며 모듈화된 운영체제의 기초를 제공함으로서 앞서 언급한 요구사항들을 효과적으로 해결할 수 있게 한다.

2.1 마이크로커널 구조

마이크로커널 기반의 운영체제는 그림 1에서 보여주는 바와 같이 최소 커널 위에 수행되는 시스템 서버들의 모임으로 구조화된다. 마이크로커널 구조는 두 가지의 주요 아이디어에 기초하고 있다[4]. 첫째, 아주 최소 부분을 제외한 모든 코드가 하드웨어에 무관하게 작성되도록 한다. 둘째, 모든 운영체제에 공통적인 하위 기능들은 운영체제의 특징을 보여주는 상위 요소들로부터 완전히 분리되도록 한다. 이러한 두 가지 목표는 운영체제가 고도로 모듈화될 것을 요구한다.

앞서 마이크로커널을 기존 운영체제 커널의 기능중 가장 필수적인 것들만 분리하여 만든 작은 핵 부분이라고 정의했는데, 이러한 개념에 기초하여 커널이라는 단어에 마이크로 외의 다양한 수식어(minimal, tiny, nano, lightweight)가 사용되고 있다[5,6]. 또한, 마이크로커널에 포함되는 기능들의 범위도 업체에 따라 약간씩 차이가 있다. 그러나 공통적인 기능들은 태스크 및 쓰레드 관리의 하위 부분, 프로세스간 통신(IPC) 및 동기화, 메모리 관리의 하위 부분, 최소의 디바이스 관리, 시스템 운영중 발생하는 각종 인터럽트 처리를 포함한다.

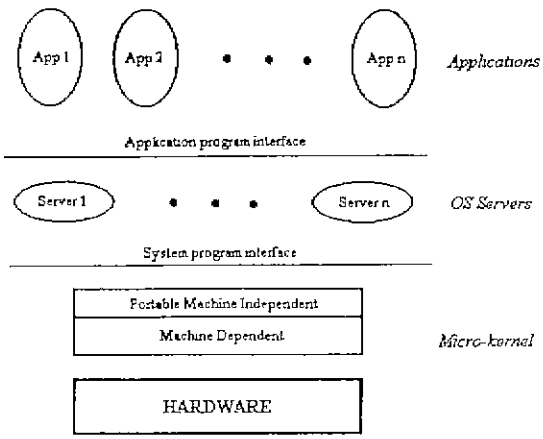

그림 1 마이크로커널 기반 운영체제의 구조

화일 서비스, 표준 UNIX 인터페이스 등과 같은 운영체제 서비스의 상위 부분들은 사용자 수준의 서버 프로그램으로 구현된다. 이러한 서버들도 업체에 따라 단일 서버 또는 다중 서버 형태로 구현되고 있다. 서버들은 주소공간 사이에 걸치는 IPC를 사용하여 상호동작함으로써 기존의 운영체제 서비스를 실현한다. 따라서, IPC 구조는 운영체제의 전체 성능에 가장 심각한 영향을 미친다[7,8].

2.2 통합 커널 구조에 비교한 장단점

마이크로커널 기반 운영체제의 가장 중요한 특징은 모듈화로서 시스템 구성요소를 서로 분리하고, 요소들간 통신 구조로서 메시지 전달 방식을 사용하는 점이다. 여기서 마이크로커널은 응용 또는 시스템 서버들과 하드웨어 사이에 교통 경찰관의 역할을 수행한다. 즉, 메시지들을 검증하여 요소들 간에 전달하며 하드웨어에 대한 액세스를 수락한다. 메시지 전달 방식의 통신 기법은 본질적으로 분산처리, 다중처리, 클러스터링, 대규모 병렬 처리 구조에 아주 적합하다. 이러한 특징들로 인하여 마이크로커널 구조는 기존의 통합 커널에 비하여 다음과 같은 잇점들을 가진다[1,3,4,9].

◦ 효율적인 분산 컴퓨팅 환경 지원

메시지 전달에 기반을 두고 운영체제 기능이 모듈화되어 있음으로 인하여 운영체제 전체가 하나의 지역 컴퓨터에 존재할 필요성은 더이상 없어진다. 예로서, 화일 시스템 서버는 다른 컴퓨터에 두고서 네트워크를 통한 화일 서비스가 가능하다. 이러한 방식은 이동 컴퓨팅 환경을 상당히 개선할 수 있게 한다. 따라서, 마이크로커널 기반의 운영체제를 갖는 PDA(Personal Digital Assistant)가 UNIX 또는 Windows 등을 탑재한 시스템과 같은 강력한 기능을 발휘할 수 있게 할 수 있다.

◦ 시스템 기능의 우수한 확장성 제공

마이크로커널 구조의 사용은 운영체제를 경제적인 방식으로 개발할 수 있게 한다. 기존의 통합 운영체제 환경에서는 제품의 정기적인 릴리즈에서 새로운 기능을 발표하고 시스템 제작자들은 전체 시스템의 재구축으로 그 기능을 채택할 수 있었다. 그러나, 마이크로커널 구조에서는 새로운 서비스의 추가와 기존 서버의 교체가 기존 구성요소나 마이크로커널 자체의 변경없이 가능하므로, 기능의 추가 또는 변경은 운영체제 전체를 다시 구축하여 시험하지 않고 모듈별로 가능하다. 이러한 방식은 객체 지향형 소프트웨어 개발과 개념적으로 유사하다.

◦ 향상된 재배치성(reconfigurability) 제공

시스템 제작자의 관점에서 보면, 컴퓨터 시스템마다 운영체제의 모든 기능을 제공하는 전략 대신에 필요한 기능만을 선택할 수 있게 함으로서 하나의 기본 시스템을 다양한 사용자들에게 제공할 수 있다. 또한, 하나의 운영체제가 팜톱 시스템부터 테라플롭스 병렬 시스템까지 지원할 수 있다.

◦ 우수한 성능 확장성 제공

운영체제 서비스의 분산으로 시스템 자원에 대한 경쟁을 현격히 줄일 수 있으므로 약결합 형태의 대규모 병렬 처리(MPP) 시스템에 적합하다. Intel, Cray, IBM 등 MPP 시스템들의 거의 대부분이 마이크로커널 기반의 운영체제를 사용하고 있다.

◦ 높은 이식성 제공

운영체제를 다른 하드웨어 플랫폼에 이식시 마이크로커널의 기계 의존적인 부분만 영향을 받는데, 커널내 기계 의존적인 부분을 국부화함으로서 시스템 이식성을 증대시킨다. 그리고, 디바이스 드라이버 코드는 프로세서 종류 또는 디바이스가 기계에 부착되는 방식보다 지원할 디바이스의 종류에 따라 구조화될 수 있다.

◦ 향상된 신뢰성 제공

신뢰성은 운영체제 코드 자체의 신뢰성과 가용성 및 고장감내의 측면에서 살펴볼 수 있다. 복잡도가 엄청나게 높은 기존의 통합 커널에서는 코드 자체의 신뢰도를 보장하기가 거의 불가능하지만, 마이크로커널 운영체제는 잘 정의된 인터페이스를 갖춘 모듈들로 구성되므로 신뢰성을 해치지 않으면서 기능 확장이 가능하다.

운영체제 서비스의 분산은 시스템의 높은 가용성 또는 고장감내의 유망한 전략으로 등장하고 있다. 독립적인 시스템을 네트워크로 연결한 분

산 시스템 환경뿐만 아니라 Cray, Tandem, Unisys와 같은 대형 시스템 제작 업체들도 소프트웨어 수준의 고장감내를 달성하기 위한 수단으로 마이크로커널 사용을 추진하고 있다. 이러한 이유는 노드들을 더욱 약하게 결합함으로서 결함의 분리가 더욱 용이해지고, 다중프로세서 구조에서는 핵심 시스템 서비스를 여러 노드에서 수행하여 한 노드의 고장을 감내할 수 있기 때문이다. 또한, 대부분의 시스템 서비스가 사용자 모드에서 수행되므로 어떤 서비스가 고장나더라도 시스템 자체는 무사할 수 있으며 복구도 통합 운영체제보다 용이하다.

◦ 효율적인 실시간 처리 기반 지원

실시간 기능의 구현에 필수 요소인 선점(preemption)을 마이크로커널 환경에서는 효율적으로 지원할 수 있으며, 실시간 응용에 맞도록 운영체제 서버의 변경도 용이하다.

◦ 개선된 유지보수성 지원

마이크로커널 구조는 확장이 용이하고 하나의 기본 운영체제 위에서 다양한 기능이 제공되므로 시스템의 유지보수를 쉽게 한다. 또한, 기존 운영체제에 비하여 사용하기가 간단할 뿐만 아니라 이해하기 쉽다.

◦ 내장형(embedded) 응용에 적합성 지원

기본 기능만 갖춘 대부분 마이크로커널의 크기는 수십 키로 바이트이므로 내장형 응용에서 온칩(on-chip)화가 가능하다. 최근에 Motorola사는 Chorus 마이크로커널을 가지는 PowerPC 계열의 프로세서 개발을 발표하였다[3].

◦ 다중 개성 지원의 용이

OSF의 연구 책임자인 Ira Goldsteign은 마이크로커널을 개성이 떨어져 나간 운영체제라고 했다[3]. 여기서, 개성이란 Windows, UNIX, OS/2 등의 시스템 인터페이스, 화일 시스템, 응용 프로그램 인터페이스의 집합을 의미한다. 마이크로커널 구조에서는 공통의 하부 서비스를 사용함으로 단일 호스트에서 여러 운영체제 환경들의 공존을 사용자 수준의 프로그램으로서 지원할 수 있다.

마이크로커널 구조는 기존 운영체제에 비교하여 여러 장점을 가지는 반면에 다음과 같은 단점들을 가진다.

◦ 상용 시스템에서 비성숙성

마이크로커널 기반 운영체제의 상업적 등장은 비교적 최근에 이루어 졌으며, 그 종류도 제한되어 있다. 또한, 지금까지는 MPP 시스템과 같은 대규모 상용 환경에 적합함을 보여주고 있다. 그러나, 세계적 주요 시스템 제작사 및 운영체제 업체들이 그들의 새로운 운영체제를 마이크로커널 구조에 기반 하여 개발하고 있다[1,3].

◦ 운영체제 성능의 감소

운영체제내 서비스 사이에 프로시쥬어 호출로서 통신하는 통합 운영체제에 비하여 메시지 전달 방식을 사용하는 마이크로커널 기반 운영체제가 성능이 뒤떨어짐은 당연하다. 따라서, 마이크로커널 개발에서 성능 개선이 가장 큰 문제로 등장하고 있다. OSF에서 1994년 중반에 발표한 Mach 마이크로커널의 개선판(MK6)에서는 통합 커널에 비하여 최악의 경우 2~5%의 성능 손실만 있고 일부 환경에서는 오히려 더 우수하도록 개선하였다[8,10].

◦ 운영체제 크기의 증가

Quantum Software Systems사의 마이크로커널 QNX를 제외한 대부분의 마이크로커널 기반 운영체제의 실제 크기는 통합 커널의 경우보다 크다. 예로서, 인텔 시스템용 UNIX SVR3.2 통합 커널은 800 키로 바이트이지만 Chorus Systems사의 Chorus/Mix SVR3.2 마이크로커널 운영체제는 841 키로 바이트이다[1]. 그러나, 사용자 모드에서 수행되는 운영체제 서비들은 스왑아우트 가능함으로 운영체제 크기의 증가는 크게 문제되지 않는다.

2.3 마이크로커널 기반 운영체제 개발 동향

현재까지 발표되었거나 개발중인 마이크로커널 기반 운영체제들은 각각의 구현 방향에 따라 나름대로의 특징을 가진다. 마이크로커널의 구조는 분산 시스템용 운영체제인 네덜란드 Vrije 대학의 Amoeba[11], 실시간용 QNX[12], Unisys사의 은행 업무용 CTOS[3], Key Logic사의 객체지향 운영체제인 KeyKOS[13] 등의 여러

사례에서 이미 채택되었다. 그리고, Sun사의 객체지향 운영체제인 SpringOS와 마이크로소프트사의 Cairo도 마이크로커널을 사용할 것이다. 마이크로소프트사의 Windows NT는 순수한 마이크로커널은 아니지만 그 특성은 충분히 갖고 있다[1,3,9].

그러나, 오늘날 상용 마이크로커널 기술의 주요 원천은 CMU의 Mach 3.0과 Chorus Systems사의 CHORUS Nucleus이다[9]. Mach 기반 운영체제는 Next 컴퓨터 사에서 Mach 2.5 기반의 NextStep을 개발한 이후, OSF의 OSF/1 MK와 IBM의 Workplace OS도 Mach 3.0을 기반으로 하고 있다. 또한, Taligent Operating Environment도 IBM의 Workplace OS를 사용할 예정이다. CHORUS Nucleus를 기반으로 하는 운영체제는 CHORUS/MiX가 있으며, Novell USG와 Tandem사 등에서도 개발을 진행중이다.

마이크로커널 구조는 모든 차세대 운영체제의 중심 설계 항목이지만, 어떻게 구현하는 것이 최선의 방향인지에 대해서는 업체마다 의견을 달리하고 있다. 특히, 운영체제의 기능중 가장 필수적인 기능을 분리한 것이 마이크로커널이라 했는데, 어느 정도가 필수적인 것인지에 대해서 논란이 많다. 디바이스 드라이버의 경우를 보면, OSF에서는 모든 드라이버들을 마이크로커널 내부에 둔다. 그러나, IBM과 Chorus에서는 기본적으로 마이크로커널 외부에 두지만 드라이버 코드의 일부는 인터럽트 발생을 제어하기 위하여 커널 영역에 두고 있다. 이와 같은 분리는 동적 재구성을 가능하도록 구현한 것이지만 OSF도 이러한 개념은 지원한다. 디바이스 드라이버를 사용자 모드의 프로세스로 하는 또다른 이유는 데이터베이스 응용 등이 디스크 액세스의 특수 유형에 따라 최적화한 드라이버를 가질 수 있고 이식성을 높일 수 있기 때문이다. 프로세스 스케쥴링의 예를 보면, IBM에서 구현한 Mach에서는 스케쥴링 정책은 커널 외부에 두고 프로세스 지명(dispatch)만을 위하여 커널을 이용한다. 이러한 방식은 외부 스케쥴러와 커널 상주 지명기 사이에 긴밀한 협력을 필요로 한다.

운영체제 기능을 어느 정도의 수준에서 마이크로커널 내외부로 분리하여 두는가하는 문제 외에 다음의 사항도 업체의 구현에 따라 다르다. 첫째, 마이크로커널에 실시간 처리와 같은 특정 응용 분야에 적합한 기능의 추가 또는 분산 공유 메모리용 코드 등의 추가 여부이다. 둘째, UNIX와 같은 전통적인 운영체제의 코드를 얼마나 그대로 사용하는 가이다. 마지막으로, 성능 향상을 위하여 운영체제 서버들을 마이크로커널과 동일한 수행 모드에 둘 수 있는 여부이다.

3. Mach 마이크로커널 계열

CMU의 Mach 운영체제는 처음부터 메시지 전달 방식에 기반하여 개발되었으므로 프로그램의 수행시 필요한 상호작용은 다른 프로그램과 메시지를 주고받음에 의하여 이루어진다. 이러한 방식으로 Mach는 서버-클라이언트 모델에 효과적으로 활용되고 있다[14]. Mach의 초기 버전은 통합 커널의 구조였으나 Mach 3.0에서 마이크로커널의 개념이 제대로 반영되었다. Mach 3.0은 대학에서 개발한 연구용 결과물에 불과하지만 OSF와 IBM에 의한 꾸준한 안정화 및 개선으로 상용 마이크로커널 기반 운영체제의 밑바탕이 되고 있다. 다음에서는 이러한 Mach 3.0의 특징을 살펴보고, Mach 3.0을 기반으로 개발되고 있는 IBM의 Workplace OS와 OSF의 OSF/1에 대하여 기술한다.

3.1 Mach 3.0 마이크로커널 특징

NeXT사의 운영체제 NextStep에 사용되었던 Mach 2.5와 같은 초기 버전은 마이크로커널에 결합된 단일 BSD 4.3 UNIX 서버를 가진다. 그러나 서버와 마이크로커널은 모두 커널 모드에서 수행되고 커널은 서비스에 강결합되어 있는 형태이므로 통합 커널 구조이다. Mach 3.0은 Mach 관련 핵심 기능과 상위 시스템 계층의 지원에 필수적인 특징만 제공하도록 설계되었다. 이것은 BSD 서비스뿐만 아니라 모든 운영체제가 구축될 수 있는 기반의 제공을 의도한 것이다.

Mach 마이크로커널은 커널 모드에서 수행되는 코드의 크기를 최소로 하고 사용자 모드에서 응용 프로그램으로 수행되는 코드의 양을 최대

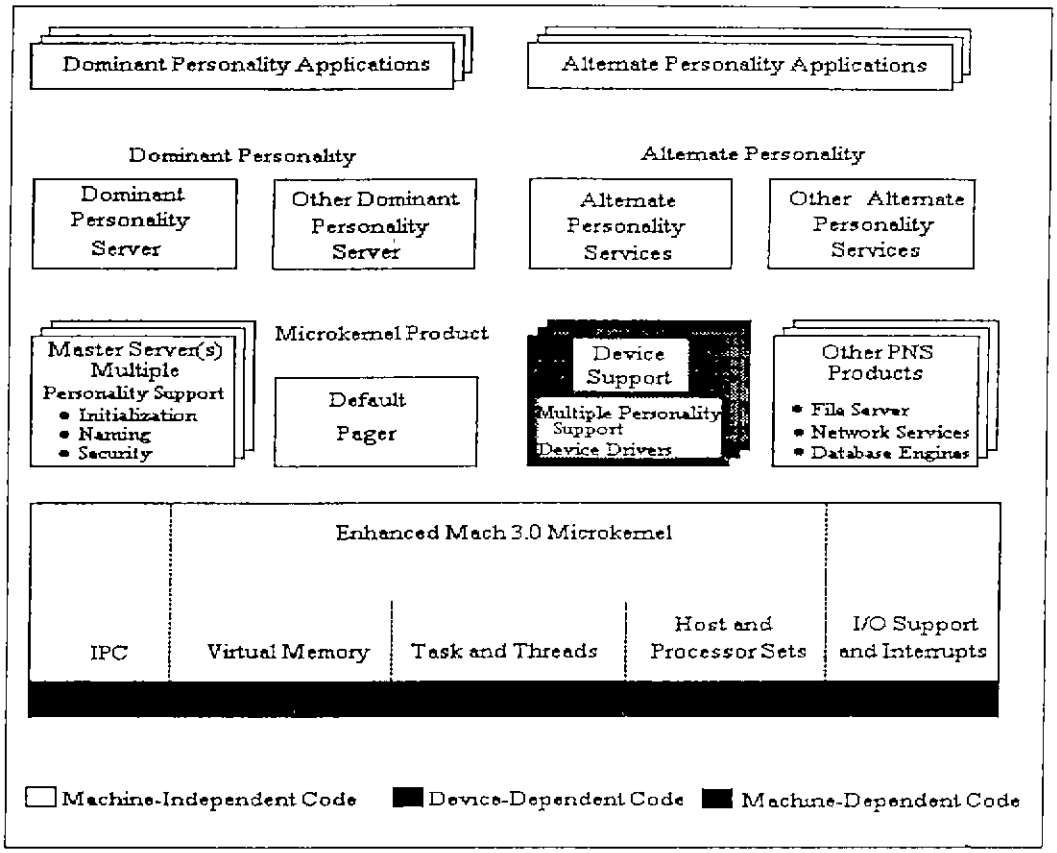

그림 2 IBM Workplace OS의 구조

화할 수 있도록 제한된 기능들만 수행한다[14, 15]. 마이크로커널이 제공하는 서비스는 다음과 같이 5가지로 분류된다(그림 2 참조).

◦ 태스크 및 쓰레드

태스크는 주소공간과 시스템 및 서버 기능에 통신 접근 도구들로서 구성되는 자원의 집합이며, 쓰레드는 Mach 커널에 의하여 스케쥴링되는 수행 실체이다. 쓰레드의 스케쥴링은 응용 프로그램에 의하여 제어될 수 있다.

◦ 프로세스간 통신(IPC)

쓰레드간 통신을 위하여 포트(port)라는 도구를 제공한다. 커널내 모든 서비스, 자원, 기능 등은 포트에 의하여 표현되며 해당 포트에 메시지를 보냄으로서 조작될 수 있다. 다중 컴퓨터 환경하의 IPC는 NORMA(NO Remote Memory Access) 모델을 기본으로 하고 있다.

◦ 가상 메모리 관리

페이징 하드웨어를 제어하고 물리적 메모리를 메모리 오브젝트 내용의 캐쉬로서 관리하는 것이 주임무이다. 메모리 오브젝트는 포트로서 표현되므로 커널과 커널 외부에 존재하는 메모리 관리자 사이의 통신은 해당 포트에 메시지를 보냄으로서 이루어진다. 이러한 커널과 메모리 관리자간 통신을 위한 규격인 EMMI(External Memory Manager Interface)가 지원된다.

◦ 호스트 및 프로세서 집합 서비스

다중처리와 다중컴퓨터 환경을 지원한다. 커널 수준의 쓰레드 제어 인터페이스 및 프로세서 집합에 대한 쓰레드의 맵핑을 제공한다. 또한, 네트워크 환경에서 각 기계는 자신의 기계마다 Mach 커널을 수행할 수 있으며, Mach 개념을 분산된 하드웨어에 투명하게 맵핑할 수 있다.

◦ 입출력 지원 및 인터럽트 서비스

Mach 커널은 낮은 수준의 디바이스 지원 기능을 가진다. 각 디바이스는 포트로 표현되므로 데이터의 전달과 디바이스의 제어는 메시지에 의하여 이루어진다.

Mach 마이크로커널에서 제공하는 서비스들은 응용 프로그램들의 추상적 처리 환경을 정의하는데 필요한 것들이다. 또한, 응용 프로그램들을 클라이언트 또는 서버로 수행되도록 하여 다른 응용과 함께 작업할 수 있게 한다. 화일 시스템, 디바이스 지원, 기존 프로그래밍 인터페이스와 같은 상위 서비스는 마이크로커널 위에서 사용자 모드로 수행되는 서버 응용 프로그램들에 의하여 제공된다. 사용자 프로그램에서 시스템 호출 또는 트랩의 처리는 에뮬레이션 라이브러리를 통하여 이루어진다.

3.2 Workplace OS

IBM은 차기 운영체제인 Workplace OS를 Mach 3.0 마이크로커널에 기반을 두고 개발하고 있다. Workplace OS의 하드웨어 플랫폼은 초기엔 인텔 및 PowerPC 프로세서이지만 여러 플랫폼으로 확장될 것이다. 하나의 마이크로커널 위에 OS/2, DOS/Windows, 여러 UNIX 버전의 개성들이 공존할 수 있는 PNS(Personality Neutral Services)를 제공한다. PNS들만 포함하는 마이크로커널 제품을 OEM으로 제공 가능하다. 두개 이상의 개성이 공존할 때, 그중 하나는 지배(dominant) 개성의 역할을 하고 나머지는 대체(alternative) 개성이다. 지배 개성은 자신이 에뮬레이션 하는 운영체제에 관련되는 것외에 시스템 주변업무(housekeeping)도 처리한다. Workplace OS의 구조는 그림 2와 같다.

IBM은 Mach 3.0 마이크로커널을 병렬처리와 실시간 작업에 적합하도록 OSF와 협력하여 다음의 세 분야에서 개선하였다. 첫째, Mach 시스템의 상용화 작업으로서 오류 수정과 코드 및 유지보수성의 개선을 포함한다. 둘째, 여러 분야에서 성능 및 기능을 개선했는데, IPC 성능 향상, 무형(untyped) IPC 제공, 새로운 동기자와 세마포어와 같은 실시간 지원 기능을 포함한다. 셋째, 실제 마이크로커널의 크기를 더 축소할 수 있도록 새로운 디바이스 드라이브 모델을 개발하는 등의 새로운 구조화를 수행하였다.

그림 2에 나타낸 마이크로커널 위의 서버(master servers, default pager, device support)들은 사용자 모드에서 수행되며 특정 개성에 독립적이다. 디바이스 드라이버와 프로세스 스케쥴링 기능의 대부분이 마이크로커널 외부로 옮겨진 것은 앞서 언급했다. 메모리 관리도 마이크로커널과 PNS 사이에 분할된다. 커널은 페이징 하드웨어를 제어하며, 커널 외부에 존재하는 페이져는 페이지 교체 정책을 결정한다. 페이져와 기타 가상 메모리 기능은 모두 화일 서버에 포함되어 있다. 그리고, 디폴트 페이져는 Workplace OS를 부팅하는데 사용된다.

Workplace OS에서 모든 서비스들은 C 라이브러리 형태의 프로시쥬어 기반으로 제공된다. 차후에 이러한 서비스들은 객체지향형 개발에 사용될 수 있는 시스템 오브젝트의 기반이 될 것이다. 현재 IBM은 객체지향 환경 개발을 Taligent와 공동 개발하고 있으며, SOM(System Object Model)과 DSOM(Distributed System Object Model)도 지원하게 될 것이다. 그리고, Workplace OS가 PNS를 기반으로 하여 더 많은 개성을 제공하고, 부속 시스템부터 대규모 병렬 시스템까지 유연하게 지원할 수 있도록 개발을 진행중이다.

3.3 OSF/1

OSF의 운영체제인 OSF/1은 처음부터 Mach 2.5를 기반으로 개발되었다. OSF/1 버전 1.2까지는 Mach 2.5 기반의 통합 운영체제이지만, 1994년 중반에 발표된 OSF/1 1.3과 모든 OSF/1 AD(Advanced Development) 버전들은 Mach 3.0 마이크로커널에 기반을 두고 있다. OSF는 Mach 3.0 마이크로커널을 IBM과 마찬가지로 성능과 기능 면에서 많은 개선을 하면서 OSF/1 및 OSF/1 AD의 개발에 반영하고 있다.

OSF에서 개발한 마이크로커널 기반의 초기 운영체제인 OSF/1 1.3은 IBM의 Workplace OS와 달리 단일 서버 구조의 OSF/1 인터페이

스를 제공한다. 이렇게 구현한 이유는 운영체제의 안정성을 기존 OSF/1으로부터 보존하기 위함이였다. 재 사용한 코드의 양은 90% 이상이다. OSF/1 1.3에는 OSF/1 서버만 구현되어 있지만, 다른 운영체제의 개성들을 추가하는 것은 쉽다.

OSF의 발표에 의하면 OSF/1 1.0에서 출발한 운영체제 계열은 1.3 버전이 마지막이고, AD 버전을 통하여 개선된 Mach 마이크로커널 기술을 반영할 것이다[10]. OSF/1 AD는 클러스터링 및 대규모 병렬처리 시스템 환경을 지원하는 운영체제로 1993년 하반기에 버전 1.0이 발표되었다. 1994년 중반에 OSF/1 AD 2.0이 발표될 예정인데, 마이크로커널 버전 6(MK6)위에서 기존의 단일 서버가 화일, 네트워크, 네임, 프로세스, 파이프 서버 등으로 분할된 다중 서버 구조를 가질 것이다.

Mach 마이크로커널에 대한 기능 및 성능 개선은 꾸준히 이루어지고 있다. 기능적인 면에서 효율적인 클러스터링 및 대규모 병렬처리를 지원하기 위하여 개선된 NORMA IPC, 분산 공유 메모리 기능인 XMM(eXtended Memory Management) 등을 MK6에서 추가하였다. 또한, 디바이스 드라이버 등의 기능들을 IBM과 유사한 형태로 분리하는 것외에 1994년 말에 발표될 MK 7에서는 실시간 기능이 추가되며 운영체제 신뢰성 및 고장감내 기능들도 계속 추가될 예정이다.

Mach 마이크로커널 구조에서 서버들은 시스템 관리자의 결정에 따라 사용자 모드 또는 시스템 모드에서 수행될 수 있다. 시스템 모드에서 서버의 수행은 메시지 전달 대신에 프로시쥬어 호출을 사용하고 서버의 코드가 메모리에 항상 상주하기 때문에 성능을 향상시킬 수 있다. MK6에서는 시스템 호출용 에뮬레이션 라이브러리를 제거하거나 쓰레드 제어 방법 등의 개선으로 통합 커널에 비한 성능 손실을 5% 이내로 줄였다[8].

OSF는 IBM을 위시한 대규모 업체들의 대부분이 OSF/1 운영체제의 원시코드를 사용하고 있거나 기술 개발에 참여하고 있는 잇점을 가지고 있다. 특히, 최근에 발표되었거나 개발중인 대규모 병렬처리 컴퓨터들의 운영체제는 대부분 마이크로커널을 기반으로 하고 있으며 또한 OSF/1을 사용하고 있다.

4. Chorus 마이크로커널 계열

프랑스의 Chorus Systems사는 상용의 마이크로커널 기반 운영체제를 개발한 첫 번째 회사이다[16]. Chorus 마이크로커널 구조는 메시지 기반의 CHORUS Nucleus를 기반으로 하고 있다. 다음에서는 Chorus 마이크로커널의 특징과 UNIX 인터페이스의 CHORUS/MiX에 대하여 기술한다.

4.1 CHORUS Nucleus 특징

Chorus 운영체제는 앞서 설명한 마이크로커널에 해당하는 Nucleus와 Nucleus 위에 공존하는 서브시스템들로 구성된다[3,17]. 여기서, 서브시스템이란 독립적인 서버들의 집합으로 특정 운영체제의 개성을 제공한다. Nucleus는 CMU의 Mach 마이크로커널에 비교하여 크기가 50내지 60 키로 바이트로 작으며, 실시간 처리 기능이 강화되어 있다. 또한, Mach 마이크로커널에 비하여 안정된 것으로 알려져 있다.

CHORUS Nucleus는 1982년에 발표된 버전 0부터 출발하여 현재는 버전 3에 해당하는 네 번째 판에 이르고 있다. 그 동안 분산 시스템 분야의 주요 학술 연구들로부터 핵심 개념들을 받아들였다. 즉, Stanford 대학의 System V로부터 메시지 전달, CMU의 Mach로부터 쓰레드 및 분산 가상 메모리, Amsterdam Vrije 대학의 Amoeba로부터 네트워크 주소관리 기술을 받아들여 사용하고 있다.

Nucleus의 구조는 그림 3에 나타내었으며, 다음과 같은 4개의 기능으로 구성된다. 그림에서 볼 수 있듯이 슈퍼바이져와 메모리 관리의 일부만 기계 의존적이다.

◦ 멀티태스킹 실시간 처리자

지역 처리기의 할당을 제어하고 우선순위 기반의 선점 허용 방식으로 CHORUS 쓰레드들을 스케쥴링한다. 또한, 쓰레드의 세밀한 동기화와 쓰레드간 통신을 위한 기본 기능을 제공한다.

◦ 가상 메모리 관리자

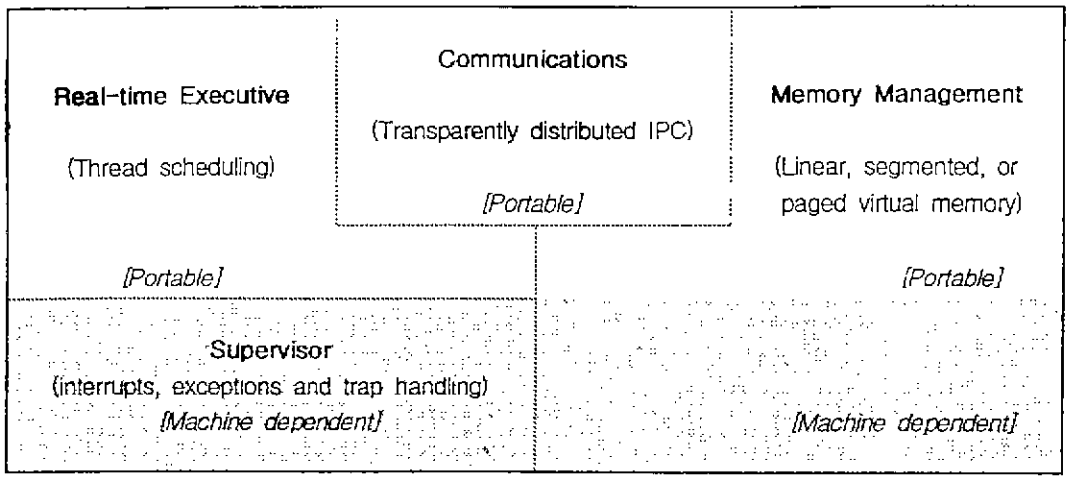

그림 3 CHORUS Nucleus 구조

메모리 구조의 전 범위를 지원할 수 있는 가상 메모리 관리 기능을 제공한다. 맵퍼(mapper)라고 하는 시스템 액터(actor)가 분산된 메모리 사이의 일관성 유지를 수행한다. 여기서, 액터는 UNIX의 프로세스와 유사한 것으로 하나 이상 쓰레드의 수행 컨텍스트를 제공하며, Mach의 태스크에 대응된다.

◦ 하드웨어 슈퍼바이저

인터럽트, 예외(exception), 트랩들을 동적으로 정의되는 디바이스 드라이버와 다른 실시간 사건 처리기에 지명한다. 시스템 호출에 사용하는 트랩은 슈퍼바이저 액터들을 통하여 효과적으로 처리된다. 이러한 액터는 사용자 프로그램에 별개의 모듈로 적재되어 하드웨어 이벤트를 직접 액세스할 수 있고 Nucleus와도 서브루틴 형식으로 직접 호출 및 복귀할 수 있게 한다.

◦ 프로세스간 통신(IPC) 관리자

시스템 전역에 걸친 포트 사이에 메시지를 전달한다. 두 종류의 통신 모드를 지원하는데, 메시지의 비동기 송수신 프로토콜과 완전한 클라이언트-서버 개념의 RPC이다. 단순한 바이트 스트링에 불과한 가장 간단한 형식의 메시지를 사용하며 흐름 제어 또는 보안 검사도 하지 않으므로 필요한 기능의 추가는 서브시스템 수준에서 구현될 수 있다.

4.2 CHORUS/MiX

CHORUS 구조에서 서브시스템들은 Nucleus의 서비스를 사용하여 상위 서비스들을 제공하는 것으로서 간단히 말해서 Nucleus 위에 구축된 운영체제이다. CHORUS/MiX는 서브시스템이 UNIX 인터페이스를 제공하는 운영체제이며, 현재 두 종류가 발표되었다(그림 4). 1988년에 발표된 CHORUS/MiX V3.2는 SVR3.2 버전이며, 1991년에 발표된 CHORUS/MiX V4.0은 SVR4.0 버전이다. 모두 기존 UNIX에 다중 쓰레드, 실시간 처리, 분산 처리 기능들이 보강되었다. CHORUS/MiX외에 CHORUS/Fusion이 있는데, 이는 SCO(Santa Cruz Operations) 개방 시스템 소프트웨어를 위한 버전으로 실시간 처리와 클러스터링 기능이 강화되었다.

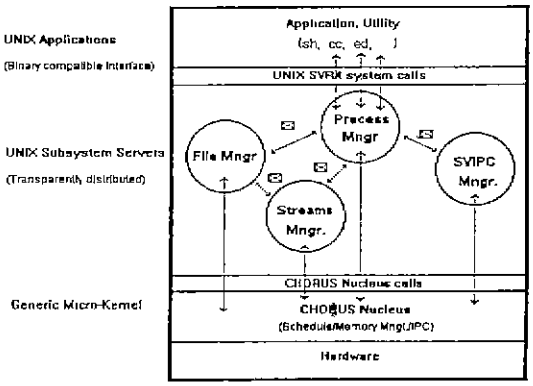

그림 4 CHORUS/MiX 구조

Chorus 운영체제는 프랑스 최대의 통신 업체인 Alcatel사의 모든 통신 장비의 운영체제로 사용되고 있으며, 영국의 ICL사에서 개발한 상용의 대규모 병렬 시스템인 GoldRush에도 사용되었다. Novell USG는 SVR4.2 MP와 NetWare가 결합된 UnixWare 2.0의 인터페이스를 마이크로커널 기반 위에서 제공하도록 개발을 시작한 UnixWare MK에 Chorus 마이크로커널 기술을 사용하고 있다. 또한, Tandem, Unisys사 등에서 개발하고 있는 대규모 병렬 시스템에도 사용될 예정이다. 현재, Chorus사는 객체지향 환경의 Chorus 운영체제 환경을 개발하기 위하여 INRIA와 Esprit 참여 업체들과 함께 COOL(Chorus Object-Oriented Layer) 과제를 수행중이다.

5. 결 론

마이크로커널 구조의 특징과 그 구조에 기반을 둔 운영체제들에 대하여 살펴보았다. 거의 모든 주요 업체들이 마이크로커널 기반의 운영체제를 이미 가지고 있거나 개발중인 것으로 보아 요즘의 운영체제 분야에서 마이크로커널이라는 말은 하나의 유행어가 되었음은 틀림없는 사실이다. 최신의 운영체제는 거의 모두 마이크로커널 구조에 의거하여 구축되고 있기 때문이다.

지금까지 운영체제 시장에서 명확한 승자가 된 업체는 없으며, 각 업체는 다른 업체의 응용들을 지원할 수 있는 방향으로 개발을 추진하고 있다. 공통된 기술 동향은 마이크로커널을 기본으로 하여 모듈형 운영체제 구조를 설계하고 객체 개념과 다중 개성을 지원함으로써 사용자 및 시스템 제작자의 다양한 요구를 신속히 만족시키려 하는 것이다.

운영체제의 분야에 새로이 요구되는 필수적 사항은 개방형 마이크로커널이다[6]. 즉, 개방형 운영체제가 응용 프로그래머에게 제공하는 잇점을 마이크로커널을 기반으로 하여 서버들을 개발하는 시스템 프로그래머에게도 동일하게 제공해야 한다. 이러한 관점에서 볼 때 앞으로 요구되는 마이크로커널의 특징은 다음과 같다.

첫째, 마이크로커널은 개방 시스템 프로그램 인터페이스(SPI)를 정의할 수 있어야 한다. 개방 SPI는 다양한 운영체제 서버를 지원할 수 있고 기존의 모든 시스템 소프트웨어가 새로운 개방 마이크로커널 운영체제의 표준으로 이전될 수 있게 한다. 또한, 제3자의 미들웨어 업체들을 위한 새로운 시장이 열릴 수 있다. 미들웨어 서버는 개방 SPI에서 수행되는 운영체제 부품에 해당하며, 데이터베이스 관리 시스템이 하나의 예이다. 둘째, 마이크로커널들은 보드, 컴퓨터, 복잡한 기계, 네트워크 구조들뿐만 아니라 모든 종류의 프로세서에 완전히 이식 가능하거나 실제 이식되어 있어야 한다. 그에 따라서, 기업 컴퓨팅 환경 구축시 동일한 SPI의 개방형 마이크로커널이 모든 범위의 컴퓨팅 요소에 사용될 수 있다. 셋째, 마이크로커널 수준의 상호 운용성을 가져야 한다. 이것은 미들웨어 운영체제 서버들이 수행될 위치에 대한 완벽한 유연성을 가지는 약결합 시스템 모습의 지원을 가능하게 한다.

이와 같은 특징을 만족하는 마이크로커널 시스템은 과거 UNIX의 경우와 같이 운영체제 시장의 흐름 변화에 선두 역할을 할 것이다. 운영체제 커널 부품 시장의 출현은 수년 전 마이크로프로세서의 경우와 유사하게 이러한 부품들은 다음과 같이 계층화될 수 있을 것이다.

운영체제 부품의 최하층은 마이크로 프로세서 계열에 밀접히 결합된 마이크로커널의 계열로서 단순한 처리기 및 디바이스로부터 고성능의 복잡한 하드웨어까지 확장가능한 것이다. 이보다 상위 층은 메모리 관리, 네트워크 관리 등과 같은 분야에서 마이크로커널의 서비스를 보완하는 기본적인 운영체제 커널 부품 계열이다. 그보다 상위 층은 운영체제 커널 개성들을 지원하는 서버 계열인데, 프로세스 관리자, 화일 및 네트워크 서비스를 예로 들 수 있다. 최상층은 운영체제 개성에 가치를 더한 미들웨어 서버들로서 데이터베이스 관리자, 시스템 및 네트워크 관리자 등이 해당된다.

마이크로커널 구조는 컴퓨터 시스템 업체, 시스템 소프트웨어 제작자, 응용 프로그래머들에게 기존 통합 커널에 비교하여 수많은 잇점을 가져다주는 것은 사실이다. 개방형 마이크로커널 구조로 발전되면 운영체제 기능에 대한 변화가 과거 UNIX의 경우보다 더 혁명적으로 일어날 것

으로 기대된다. 따라서, 향후 마이크로커널 구조에 대한 연구 개발은 더욱 활발하게 이루어질 것으로 예상된다.

참고문헌

[1] R. Farrow, "Microkernels: The Sole of a New OS," UNIXWORLD, pp. 62~64, November 1993.

[2] M. Gien, "Next Generation Operating Systems Architecture," Technical Report No. CS/TR-91-104, Chorus Systems, July 1991.

[3] J. Udell, "The Great OS debate," BYTE, pp. 117~168, January 1994.

[4] A. G. Taylor, Microkernel Technology, UNIXWORLD: DIRECT, 1994.

[5] J. Nacon, CONCURRENT SYSTEMS: An Integrated Approach to Operating Systems, Database, and Distributed Systems, p. 48, Addison-Wesley Publishing Company, 1993.

[6] M. Gien, "Microkernel-Based Operating Systems Take a New Direction," UniForum Monthly, pp. 46~47, September 1994.

[7] B. Bershad, "The Increasing Irrelevance of IPC Performance for Microkernel-based Operating Systems," Proceedings of USENIX Workshop on Micro-Kernels and Other Kernel Architectures, pp. 205~211, April 1992.

[8] M. Condict, et al., "Microkernel Modularity with Integrated Kernel Performance: Improving the Performance of MK OSF/1 Via Collocation, RPC Short Circuiting and Optimization," Collected Papers, The OSF Research Institute, April 1994.

[9] M. A. Goulde, "Tomorrow's Microkernel-Based Unix Operating Systems," Open Information Systems, Vol. 8, No. 8, pp. 1~16, August 1993.

[10] OSF, "Introduction to the OSF Research Institute," Materials from OSF's 6th Anniversity International Members Meeting, Hotel Conrad, Brussels, May 24-26, 1994.

[11] R. V. Renesse and A. S. Tanenbaum, "Short Overview of Amoeba," Proceedings of USENIX Workshop on Micro-Kernels and Other Kernel Architectures, pp. 1~10, April 1992.

[12] D. Hilderbrand, "An Architectural Overview of QNX," Proceedings of USENIX Workshop on Micro-Kernels and Other Kernel Architectures, pp. 113~123, April 1992.

[13] A. C. Bomberger and N. Hardy, "The KeyKOS Nanokernel Architecture," Proceedings of USENIX Workshop on Micro-Kernels and Other Kernel Architectures, pp. 95~112, April 1992.

[14] M. Accetta, et al., "Mach: A New Kernel Foundation for UNIX Development," Proceedings of Summer 1986 USENIX Conference, July 1986.

[15] K. Loepere, "OSF Mach Final Draft: Kernel Principles," Books in OSF Mach series (available via anonymous FTP from riftp.osf.org), OSF and CMU, May 3 1993.

[16] Chorus Systems, "Background Information," published in Chorus Systems, June 1994.

[17] M. Rozier, et. al., "Overview of the Chorus Distributed Operating System," Proceedings of USENIX Workshop on Micro Kernels and Other Kernel Architectures, pp. 39~69, April 1992.

최 효 진

1975 1983 경북대학교 전자공학과 학사
1993 정보처리 전자계산조직 응용 기술사
1994 충남대학교 전산학과 석사
1983 ~현재 한국전자통신연구소 선임연구원
관심 분야 : 운영체제, 병렬처리 시스템 구조

임 기 욱

인하대학교 전자공학과 학사
한양대학교 전자계산학과 석사
인하대학교 전자계산학과 박사
1977 ~1988 한국전자통신연구소 시스템소프트웨어 연구실장
1988 ~1989 캘리포니아 주립대(U.C. Irvine) 방문 연구원
1989 ~현재 동연구소 컴퓨터연구단 시스템연구부장
관심분야 : 운영체제, 데이타베이스시스템, 시스템 구조