마이크로커널 구조가 캐시 메모리의 성능에 미치는 영향

(Effect of Microkernel Structure on Cache Memory Performance)

장 문 석 † 고 건 ††
(Moon-Seok Chang) (Kern Koh)

요 약 모듈화된 구조를 지향하는 현대 소프트웨어 기술의 발전은 캐시 메모리의 성능에 큰 변화를 가져오고 있다. 최근 운영체제 분야에서 새로운 설계 기술로 부각되고 있는 마이크로커널은 모듈화된 구조를 가지고 있어 이식성과 확장성이 우수하지만, 모노리딕 커널에 비하여 성능이 저하되는 현상을 보이기도 한다.

본 논문에서는 마이크로커널 기반 운영체제에서 발생하는 성능 저하의 근본적인 원인을 규명하기 위하여, 커널의 구조적 특성이 캐시 메모리의 성능에 미치는 영향을 정량적으로 분석하였다. Intel Pentium Pro 프로세서 상에서의 실험 결과, 마이크로커널 구조는 모노리딕 커널 구조에 비하여 L1, L2 캐시와 TLB 접근 실패율을 크게 증가시키며, IPC 보다는 캐시 메모리의 효율성이 운영체제 성능에 미치는 영향이 더욱 크다는 사실을 발견하였다. 그리고, 이러한 현상은 마이크로커널의 구조적 특성으로 인하여 빈번히 발생하는 문맥 교환의 영향임을 확인하였다.

Abstract The modern software technology toward modularization has changed the cache accessing behavior dramatically. Many modern operating systems are also departing from the past monolithic structure toward the highly modularized structure referred to as microkernel. Microkernel-based operating systems are more portable and extensible, but are likely to have worse performance.

This paper quantitatively analyzes the effect of microkernel structure on cache memory to identify the primary factor for its performance degradation. Through the experiment performed on a Intel Pentium Pro processor platform, we found that the microkernel structure suffers from remarkably higher misses for L1, L2 cache and TLB than the monolithic one does. We also found that the performance of a microkernel is more dependent on the efficiency of cache memory than IPC. Finally, we found that these results come from the effect of frequent context switches mainly caused by the structural feature of a microkernel.

1. 서 론

컴퓨터 시스템의 성능은 프로세서의 성능과 메모리 시스템의 효율성에 의해 크게 좌우된다. 이 중에서 프로세서 성능은 처리 속도와 함께 지속적으로 향상되고 있음에 반하여, 메모리 접근 속도의 개선은 이에 크게 못 미치고 있다. 메모리 접근 속도가 개선되지 않으면 메모리 접근에 소요되는 프로세서 사이클 시간이 상대적으로 증가하게 되므로, 아무리 빠른 프로세서를 사용한다 하더라도 성능을 제대로 발휘할 수 없다.

프로세서의 처리 속도와 메모리 접근 속도의 차이에 의한 성능 저하를 방지하기 위해 널리 쓰이는 방법은 캐시 메모리(cache memory)를 사용하는 것이다[1]. 캐시 메모리는 보다 빠른 메모리 접근 시간을 필요로 하는 고속 프로세서를 사용하는데 필수적인 요소이다. 특히, 현재 프로세서 개발 분야에서 주류를 이루고 있는

† 비 회 원 : 한국전자통신연구원 운영체제연구팀 연구원
mschang@etri.re.kr
†† 종신회원 : 서울대학교 전산과학과 교수
kernkoh@june.snu.ac.kr
논문접수 : 1998년 7월 20일
심사완료 : 1999년 10월 7일

RISC 프로세서들은 클럭 속도가 수 백 MHz에 이르고 있으며, 보다 많은 메모리 대역폭을 요구하고 있다. 이러한 추세에 비추어 볼 때, 향후 개발될 고성능 컴퓨터 시스템에서 캐시 메모리에 대한 의존도는 더욱 심화될 전망이다.

한편, 현재 진행되고 있는 소프트웨어 기술의 발전은 캐시 메모리의 성능에 많은 변화를 가져오고 있다. 이 중에서 가장 주목할 만한 것은 점차 모듈화(modularization)되어 가고 있는 소프트웨어 구조의 변화이다. 최근 개발되고 있는 소프트웨어들은 이식성과 확장성, 재사용성 등을 제고하기 위하여 세분화된 모듈 구조로 설계되고 있다. 이들은 기존의 소프트웨어에 비하여 프로그램 코드의 크기가 크고, 보다 넓은 영역의 데이타를 참조하므로 워킹셋(working set)의 크기도 크다. 또한 세분화된 함수들을 자주 호출하게 되고 제어 흐름의 변화가 빈번하기 때문에, 기존의 소프트웨어에 비하여 낮은 지역성을 가지고 있다.

모듈화된 구조를 갖는 소프트웨어는 각각의 모듈이 독립된 주소 공간을 갖는 태스크(task)들로 구현되기도 하며, 이 태스크들이 메시지로 통신하면서 작업을 수행하도록 구성되어 있다. 이러한 소프트웨어의 대표적인 예로는 객체 지향 소프트웨어, 메시지 전달 방식의 소프트웨어, 클라이언트/서버 방식의 소프트웨어, 계층화된 구조를 갖는 네트웍 소프트웨어, 대규모 멀티미디어 소프트웨어 등이 있다. 이들은 대부분 기존의 소프트웨어에 비하여 높은 캐시 접근 실패율을 보이고 있으며, 그 결과 전체 시스템의 성능이 크게 저하되는 현상을 보여주고 있다[2,3,4].

이러한 소프트웨어 구조의 변화는 응용 프로그램에만 국한되는 것이 아니다. 대표적인 시스템 소프트웨어인 운영체제와 데이타베이스 분야에서도 소프트웨어 구조의 모듈화가 급속히 진행되고 있다. 새로운 운영체제 설계 기술로 부각되고 있는 마이크로커널(microkernel) 기반 운영체제는 이러한 추세를 따르는 대표적인 시스템 소프트웨어이다[2,5].

운영체제 커널의 기능을 단일 주소 공간 내에 구현하는 모노리딕 커널(monolithic kernel)과는 달리, 마이크로커널 기반 운영체제는 핵심적인 기능을 담당하는 마이크로커널 부분과, 화일 서비스나 네트웍 서비스와 같은 운영체제 상위 수준의 서비스를 제공하는 하나 이상의 서버(server) 태스크들로 구성되어 있다. 이 서버들은 서로 독립적인 주소 공간을 가지고 있으며, 메시지로 통신하면서 기능을 수행한다. 커널과 서버로 모듈화된 운영체제는 새로운 기능을 추가하거나 불필요한 부분을 삭제하기 쉬우므로 확장성이 우수하고 재구성이 용이하며, 소프트웨어 공학적인 측면에서 여러 가지 장점들을 가지고 있다. 이러한 장점들은 마이크로커널 구조를 차세대 운영체제 설계 기술로 부각시키는 원동력이 되었다.

그러나 이러한 장점에도 불구하고, 다른 한편으로는 마이크로커널의 모듈화된 구조가 운영체제의 성능을 저하시키는 원인으로 지적되기도 한다[2,6]. 즉, 일정한 작업 부하를 수행하였을 때, 마이크로커널 기반 운영체제는 동일한 기능을 갖는 모노리딕 커널 기반 운영체제에 비하여 더 오랜 수행 시간을 필요로 한다는 것이다. 이러한 성능 저하 문제는 마이크로커널을 상용화하는데 커다란 걸림돌로 작용해 왔으며, 마이크로커널 구조가 새로운 운영체제 구조로서 올바른 설계 방식인가에 관한 많은 논란을 불러일으키고 있다[2,5,6].

소프트웨어 공학적인 장점과 성능 저하의 단점이 병존하는 마이크로커널 구조가 차세대 운영체제 설계 기술로 정착되기 위해서는, 운영체제의 성능을 정량적으로 평가하고 성능 저하의 근본적인 원인을 규명하는 작업이 반드시 선행되어야 한다. 그리고, 실제로 동작하는 시스템 상에서 소프트웨어 수준 뿐 아니라 하드웨어 수준의 미세한 성능 요소들을 포함하는 정밀한 성능 측정 작업이 수반되어야 한다.

본 논문에서는 마이크로커널의 구조적인 특성이 운영체제의 성능에 미치는 영향을 정량적으로 평가하고, 성능 저하의 근본적인 원인을 규명하는 것을 목적으로 한다. 이를 위하여 마이크로커널 기반 운영체제와 모노리딕 커널 기반 운영체제 상에서 벤치마크 프로그램을 수행하여 성능을 측정하고 결과를 비교 분석하였다. 성능 측정은 운영체제의 구조적 특성에 의해 차이가 발생하는 문맥 교환(context switch), IPC(interprocess communication), 그리고 캐시 메모리와 TLB(Translation Lookaside Buffer)의 접근 실패율에 중점을 두었다. 특히 향후 개발될 고성능 컴퓨터 시스템에서 메모리 시스템의 중요성이 크게 증대되고 있음을 감안하여, 캐시 메모리와 TLB의 영향을 성능 요소에 추가함으로써 성능 평가의 정확성을 제고하였다.

본 논문의 구성은 다음과 같다. 우선 제 2 장에서는 현재의 하드웨어 기술과 운영체제 성능과의 상호 작용에 대한 관련 연구를 기술한다. 제 3 장에서는 마이크로커널의 구조적 특성을 기술하고, 예상되는 주요 성능 저하 요인으로 IPC와 캐시 메모리의 효율성을 제시한다. 제 4 장에서는 마이크로커널과 모노리딕 커널 기반 운영체제 상에서 벤치마크 프로그램을 수행하면서 성능을

측정한 결과를 기술한다. 그리고, 두 운영체제의 구조적 차이에서 발생하는 성능 차이를 분석하여 성능 저하의 주요 요인을 규명하고, 마지막으로 제 5 장에서 결론을 맺는다.

2. 관련 연구

오늘날 운영체제의 성능은 하드웨어 기술 발전과 밀접한 관계를 가지고 있다. 운영체제의 성능은 커널의 하위 기본 서비스들을 얼마나 효율적으로 수행하느냐에 의해 결정되며, 하위 기본 서비스들의 성능은 하드웨어의 지원에 의해 크게 좌우되기 때문이다[7,8].

현대 하드웨어 기술의 대표적인 흐름은 프로세서의 고속화와 RISC 기술의 보편화이다. 이러한 흐름은 보다 빠른 접근 시간과 큰 대역폭을 갖는 메모리 시스템을 요구하고 있으며, 캐시 메모리의 중요성을 더욱 강조하고 있다. 이러한 추세에 비추어 볼 때, 현대 운영체제의 성능을 평가함에 있어 캐시 메모리의 영향은 반드시 포함되어야 할 중요한 성능 요소이다.

운영체제 성능과 캐시 메모리의 연관성에 대한 연구는 Agarwal에 의해 본격적으로 시작되었다[9]. 그 이전까지 운영체제 커널의 영향을 고려한 캐시 메모리 성능 연구는 상당히 미약하였는데, 이는 커널 메모리 참조에 대한 트레이스(trace) 데이타를 얻기가 대단히 어려웠기 때문이다. Agarwal은 커널 메모리 참조를 포함하는 트레이스 기반 시뮬레이션(trace-driven simulation) 실험을 통하여, 운영체제 커널은 일반 응용 프로그램에 비하여 워킹셋의 크기가 크고 반복문의 빈도가 낮으므로 높은 캐시 접근 실패율을 보인다는 사실을 확인하였다. 또한 다중 프로그래밍 환경에서 캐시 접근 실패율은 다중화의 정도(multiprogramming degree)에 따라 크게 변화한다는 사실을 발견하였다.

운영체제 성능과 하드웨어 기술 간의 상호 작용에 관한 연구는 Ousterhout[8]과 Anderson[7]에 의해 본격적으로 시작되었다. 그들은 전체 시스템 성능에서 운영체제의 성능이 차지하는 비중을 강조하고, 하드웨어 기술 발전이 운영체제 성능에 미치는 영향을 심도 있게 분석하였다. 그들은 시스템 호출이나 메모리 복사, 그리고 문맥 교환과 IPC 같은 운영체제의 주요 활동이 시스템의 성능을 크게 좌우한다는 점을 발견하고, 하드웨어 수준의 성능 요소가 운영체제의 성능에 미치는 영향이 크게 증가하고 있다는 점을 지적하였다. 특히 최신 하드웨어일수록 보다 효율적인 메모리 시스템을 요구하므로, 캐시 메모리와 TLB 같은 하드웨어 수준의 성능 요소에 대한 의존도가 더욱 심화되고 있다는 점을 강조하였다. 이러한 주장은 현대 운영체제의 성능을 정확하게 평가하기 위해서는 소프트웨어 수준과 하드웨어 수준의 성능 요소를 반드시 함께 측정해야 한다는 점을 강력히 시사하고 있다.

운영체제 성능과 하드웨어 기술간의 상호 작용에 대한 중요성이 인식되면서, 하드웨어의 핵심 요소인 캐시 메모리와 운영체제 성능간의 관계를 연구하는 움직임은 더욱 활발히 전개되었다. 대표적인 예로, 운영체제 커널에서 발생하는 문맥 교환이 캐시 메모리의 성능에 미치는 영향에 대한 정량적 분석 작업이 Mogul에 의해 시도되었다[10]. 그는 트레이스 기반 시뮬레이션을 통하여 문맥 교환에 의해 발생하는 캐시 접근 실패와 이에 의해 소요되는 프로세서 클럭 시간을 계산함으로써, 문맥 교환의 전체 오버헤드 중에서 캐시 접근 실패에 의한 부담이 큰 비중을 차지한다는 사실을 발견하였다. 즉, 문맥 교환 후 캐시 메모리를 새로운 워킹셋으로 채우기 위해 소모되는 부담이 운영체제가 문맥 교환을 하기 위하여 커널 명령어들을 수행하는 부담보다 훨씬 크다는 것이다.

운영체제의 구조가 캐시 메모리의 성능에 미치는 영향을 분석한 대표적인 연구는 Chen의 연구이다[3]. Chen은 마이크로커널 기반 운영체제인 Mach와 모노리딕 커널 기반 운영체제인 Ultrix 상에서 캐시 메모리와 TLB의 성능을 측정함으로써, 마이크로커널 구조가 메모리 시스템 성능에 상당한 영향을 미친다는 사실을 발견하였다. 즉, 사용자와 커널 주소 공간 사이에서 발생하는 충돌 실패(conflict miss)의 증가로 인하여 Mach가 Ultrix에 비하여 높은 캐시 접근 실패율을 보인다는 사실을 확인하였다. 그러나 Chen은 두 운영체제에서 발생하는 캐시 접근 실패율의 차이가 전체 시스템 성능에 미치는 영향은 크지 않다고 주장함으로써, 마이크로커널의 성능을 옹호하는 입장을 취하였다. 이러한 Chen의 주장은 트레이스 기반 시뮬레이션을 통하여 얻은 결론으로서, 측정 기법의 한계로 인하여 벤치마크 프로그램이 다중 프로그래밍의 영향을 제대로 반영하지 못했고, 시스템 호출의 빈도가 낮아서 마이크로커널에서 발생하는 성능 저하 현상을 제대로 포착하지 못했다는 결정적인 단점을 가지고 있다.

Chen의 연구를 포함하여 기존 연구들이 사용한 트레이스 기반 시뮬레이션은 트레이스 데이타 길이의 한계로 인하여 측정 기간이 상당히 짧고, 문맥 교환의 발생이나 다중 프로그래밍의 효과와 같이 캐시 메모리에 중요한 영향을 미치는 성능 요소를 반영하기가 어렵다. 또한 트레이스 데이타를 추출할 때 발생하는 공간적 시간

적 왜곡에 의하여 발생하는 간섭 현상(interference)을 피하기 어려우므로 성능 측정의 정확도가 크게 저하될 수 있다.

한편, 본 논문에서 캐시 메모리와 함께 주요 성능 요소로 언급되고 있는 TLB(Translation Lookaside Buffer)는 페이징을 기반으로 하는 가상 메모리 시스템에서 가상 주소를 물리적 주소로 번역하는 프로세서 내부의 주소 번역용 캐시이다. TLB 접근이 실패하면 TLB 폴트가 발생하는데, 이 부담은 캐시 접근 실패의 부담에 비하여 훨씬 크다. 최신 프로세서의 경우 TLB 폴트의 부담은 하드웨어적으로 처리하는 경우 수십 사이클, 소프트웨어적으로 처리하는 경우 수백 사이클에 이른다[4]. 특히, 문맥 교환시에 발생하는 가상 주소 공간의 교체는 TLB에 적재해 놓았던 모든 가상 메모리 페이징 정보를 무효화시키는 TLB 플러시(flush)를 유발하므로 메모리 시스템의 성능을 크게 저하시킨다. 더욱이 물리적 캐시를 사용하는 프로세서의 경우, TLB에 의한 주소 번역 과정은 캐시 메모리 접근의 임계 경로(critical path) 상에 존재하므로 TLB 접근 시간은 캐시 적중 시간을 결정하는 중요한 요소로 작용하기도 한다[11].

TLB의 효율성과 운영체제 성능과의 관계에 대한 대표적인 연구는 Uhlig에 의해 발표된 바 있다[4]. Uhlig는 트레이스 기반 시뮬레이션을 통하여 마이크로커널 기반 운영체제에서 모듈화의 정도, 즉 서버의 수를 변화시키면서 TLB의 접근 실패율을 분석하였다. 그 결과 서버의 수가 많을수록 TLB 접근 실패율이 높고 성능이 저하된다는 사실을 발견하였다. 그러나 그는 운영체제의 구조적인 문제를 지적하기보다는, TLB 서비스 루틴을 수정하고 TLB의 크기와 하드웨어적인 구조 개선을 통하여 성능을 개선하는데 주력하였다.

지금까지 언급한 기존 연구의 결과를 정리하면 다음과 같다.

첫째, 컴퓨터 시스템의 성능을 정확히 평가하기 위해서는 운영체제의 성능을 반드시 포함해야 한다. 또한, 실제로 동작 중인 시스템 상에서 다중 프로그래밍의 효과를 반영하여 성능을 측정해야한다.

둘째, 캐시 메모리와 TLB의 성능이 전체 시스템 성능에 미치는 영향은 대단히 크다. 그리고, 이러한 영향은 향후 개발되는 고성능 프로세서에서 더욱 증가할 것으로 예상된다.

셋째, 문맥 교환은 캐시 메모리와 TLB의 성능을 크게 저하시킨다.

이러한 기존 연구 결과와 마이크로커널 구조가 문맥 교환을 빈번히 유발한다는 사실은, 마이크로커널 기반 운영체제의 성능 저하 현상이 캐시 메모리와 TLB의 성능 저하에 기인할 수 있다는 가능성을 강력히 시사하고 있다. 따라서, 마이크로커널의 성능 저하 문제를 규명하기 위해서는 이러한 가능성을 면밀히 검토하는 작업이 절실히 필요하다.

3. 마이크로커널 기반 운영체제

3.1 구조적 특성과 IPC

마이크로커널 기반 운영체제는 일반적으로 마이크로커널과 하나 이상의 서버 태스크들로 구성되어 있다. 마이크로커널은 태스크 관리나 메모리 관리와 같은 운영체제로서 최소한의 핵심 기능만을 수행하고, 그 외의 복합적인 기능은 사용자 수준에서 동작하는 서버 태스크들이 처리하도록 한다. 모듈화된 서버 태스크들은 마이크로커널 또는 다른 서버 태스크들과 IPC를 이용하여 통신하면서 화일 서비스나 네트웍 서비스와 같은 상위 수준의 운영체제 서비스를 수행한다. 이와 같은 마이크로커널의 대표적인 예로는 Carnegie Mellon University에서 개발한 Mach 3.0과 이를 상용화한 OSF/1 MK, 그리고 Chorus Systems사의 Chorus가 있으며, 이밖에도 GMD의 L4를 비롯하여 많은 운영체제들이 마이크로커널을 기반으로 개발되고 있다[2,5].

마이크로커널 기반 운영체제는 소프트웨어 공학적인 측면에서 여러 가지 장점을 가지고 있으나, 다른 한편으로는 모노리딕 커널 기반 운영체제에 비하여 성능이 저하된다는 단점이 지적되고 있기도 하다. 그리고 성능 저하의 가장 큰 요인이 마이크로커널 기반 운영체제에서 빈번히 요청하는 IPC를 수행하는데 소요되는 부담이라는 것이 기존 연구에서 제시하고 있는 공통된 견해이다[2,6,12].

마이크로커널 기반 운영체제가 IPC를 자주 요청하게 되는 원인은 운영체제의 구조적 특성에 있다. 마이크로커널 기반 운영체제는 운영체제의 기능이 마이크로커널과 서버 태스크들에 분산되어 있기 때문에 운영체제 기능을 수행하기 위해서는 이들 사이에 빈번한 통신을 필요로 한다. 특히, 운영체제의 특성상 다른 서버 내에 존재하는 상태 정보들을 읽어 오거나 수정해야하는 경우가 상당히 많으므로, 서버들간에 IPC를 이용한 통신을 빈번히 수행하게 된다.

3.2 시스템 호출과 문맥 교환

마이크로커널 기반 운영체제에서 IPC에 의한 성능 저하 문제가 제기된 이후, IPC의 효율성을 높이기 위하여 많은 연구가 진행되었다. Mach 운영체제의 경우, 송

수신 커널 인터페이스의 통합(*mach_msg()*), continuation 기법 등을 이용하여 IPC의 성능을 개선하였다[12]. 그러나, 이러한 IPC 성능 개선에도 불구하고 모노리딕 커널 기반 운영체제와의 성능 차이는 좀처럼 줄어들지 않고 있다[13]. 이러한 사실은 마이크로커널 기반 운영체제에서 IPC 이외에 성능 저하를 유발하는 보다 근본적인 요인이 있음을 암시하고 있다.

마이크로커널 기반 운영체제의 성능 문제에서 IPC 다음으로 검토해야 할 대상은 바로 문맥 교환이다. 마이크로커널 기반 운영체제는 마이크로커널과 서버 태스크, 사용자 태스크들이 독립된 주소 공간으로 나뉘어져 있는데, 이들이 IPC로 통신하는 과정에서 송신 태스크에서 수신 태스크로의 문맥 교환이 발생하게 된다.

마이크로커널 기반 운영체제에서 문맥 교환이 자주 발생하는 또 하나의 원인은 기존의 운영체제와의 이진 호환성을 제공하기 위하여 시스템 호출을 에뮬레이션으로 처리하는 과정의 비효율성에 있다. 시스템 호출을 트랩(trap) 메커니즘으로 처리하는 모노리딕 커널과는 달리, 마이크로커널 기반 운영체제는 사용자가 요청한 시스템 호출을 서버 태스크에게 IPC를 통하여 전달하는데, 이 과정에서 문맥 교환이 발생하게 된다. 그 대표적인 예로서, 그림 1은 Mach 운영체제가 UNIX 시스템 호출을 처리하는 과정을 보여 주고 있다. 그림 1에서 사용자 태스크에서 요청한 UNIX 시스템 호출을 처리하기 위하여 서버 태스크로(4), 서버 태스크에서 다시 사용자 태스크로(7) 최소한 두 번의 문맥 교환이 발생한다. 이러한 운영체제의 구조적인 특성으로 인한 문맥 교환은 운영체제의 서비스를 여러 개의 서버 태스크에서 분산하여 수행하는 모듈화된 서버 구조에서 더욱 빈번히 발생한다.

문맥 교환이 발생하면 운영체제 커널은 프로세서 레지스터들과 상태 정보를 저장한 다음, 새로운 태스크를 선택하고 이전에 저장한 문맥을 복구하는 일련의 동작을 수행한다. 그러나, 이러한 직접적인 부담보다 더욱 중요한 것은 문맥 교환이 캐시 메모리에 미치는 간접적인 부담이다. 즉, 문맥 교환 후 캐시 메모리를 새로운 태스크의 워킹셋으로 채우기 위해 소모되는 부담이 훨씬 크다는 것이다. 문맥 교환이 캐시 메모리의 성능에 미치는 영향의 중요성은 Mogul에 의해 확인된 바 있다[10].

문맥 교환이 캐시 메모리의 성능을 크게 저하시킨다는 점을 감안할 때, 구조적 특성상 문맥 교환이 빈번히 발생하는 마이크로커널 기반 운영체제에서 캐시 메모리의 성능이 크게 저하됨을 예측할 수 있다. 그리고, 이러한 예측은 마이크로커널 기반 운영체제의 성능 저하에 대한 근본 원인을 캐시 메모리의 성능 저하에서 찾을 수 있다는 새로운 가능성을 시사하고 있다. 다음 장에서는 이러한 가설을 확인하기 위하여 마이크로커널 기반 운영체제와 모노리딕 커널 기반 운영체제에서 캐시 메모리의 성능을 정량적으로 측정하고 그 결과를 분석하기로 한다.

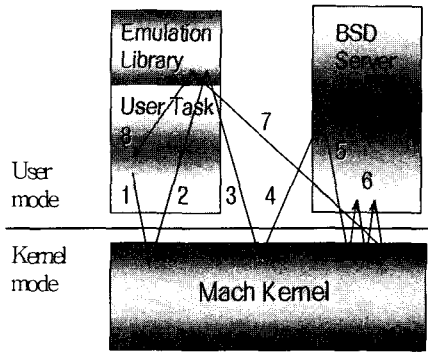


그림 1 Mach 운영체제에서 UNIX 시스템 호출의 처리 과정

4. 성능 측정과 분석

마이크로커널 기반 운영체제와 모노리딕 커널 기반 운영체제의 구조적 차이가 운영체제 성능에 미치는 영향을 분석하는 가장 좋은 방법은, 유사한 커널 코드를 가지면서 상반된 구조를 갖는 두 가지 운영체제의 성능 차이를 비교하는 것이다. 본 논문에서는 마이크로커널 기반 운영체제인 OSF/1 MK 7.2[6]과 모노리딕 커널 기반 운영체제인 OSF/1 IK 1.1[6] 상에서 SPEC SDM 벤치마크 프로그램을 수행하면서 두 운영체제의 성능을 측정하고 결과를 비교 분석하기로 한다. OSF/1 IK 1.1과 OSF/1 MK 7.2는 모두 Mach 마이크로커널을 바탕으로 발전해 온 운영체제로서 원시 코드가 거의 동일하며, 단지 운영체제 구조만이 상이할 뿐이다. 즉, OSF/1 MK 7.2는 Mach 마이크로커널을 기반으로 BSD와 SVR4 운영체제 인터페이스를 지원하는 분리된 서버를 가지고 있으며, OSF/1 IK 1.1은 커널 부분과 서버가 결합된 구조를 가지고 있다. 따라서, 다른 성능 요소를 배제하고 운영체제의 구조적인 차이에 의한 성능 차이를 분석하는데 적합하다.

본 실험에서 두 운영체제는 Intel Pentium Pro 프로세서를 사용하는 하드웨어 환경에서 수행된다. 실험에

앞서 Pentium Pro 프로세서의 특징과 성능 측정 카운터에 관하여 간략히 기술한다.

4.1 Pentium Pro 프로세서의 특징

Intel Pentium Pro 프로세서는 기존의 Intel 80x86 계열의 프로세서와 호환성을 유지하면서, 성능 향상을 위하여 내부적으로 여러 가지 측면에서 발전된 구조를 가지고 있다. Pentium Pro 프로세서는 메모리 서브시스템으로 L1 캐시와 L2 캐시 메모리를 프로세서 칩 내부에 가지고 있다. L1 캐시는 8 KByte 크기의 이중 연관도(2-way associativity)를 갖는 명령어 캐시(ICache)와 8 KByte 크기의 사중 연관도를 갖는 데이타 캐시(DCache)로 나뉘어져 있다. 통합 캐시인 L2 캐시는 사중 연관도를 가진 256KB 또는 512KB 크기를 갖는 SRAM으로서, 프로세서 클럭과 동일한 속도로 동작하는 64비트의 내부 캐시 버스에 연결되어 있다. L1, L2 캐시의 캐시 블록(clock)의 크기는 모두 32 byte이다. 또한 Pentium Pro 프로세서는 독립된 명령어 TLB(ITLB)와 데이타 TLB(DTLB)를 가지고 있다. ITLB는 32개의 엔트리, DTLB는 64개의 엔트리를 가지고 있는데, 모두 사중 연관도를 갖는다[14].

Pentium Pro 프로세서의 또 하나의 특징은 프로세서 자체적으로 하드웨어 수준의 성능 측정 카운터(performance monitoring counter)를 제공하고 있다는 점이다[15]. 이 성능 측정 카운터 이용하면 프로그램의 수행 시간을 프로세서 클럭 단위로 측정할 수 있고, L1, L2 캐시와 TLB 접근 실패율과 같은 프로세서 수준의 성능을 정밀하게 측정할 수 있다. 뿐만 아니라, 운영체제 커널의 활동과 다중 프로그래밍 환경이 캐시 메모리와 TLB 성능에 미치는 영향까지 포착함으로서, 시스템이 실제로 동작하는 환경에서 발휘하는 성능을 정밀하게 측정할 수 있다. 본 논문에서는 이 성능 측정 카운터를 사용하여 두 운영체제의 성능을 측정하였다.

4.2 성능 메트릭

본 실험의 목적은 두 운영체제의 성능을 비교하고, 성능 차이의 원인을 메모리 시스템의 효율성을 중심으로 분석하는 것이다. 이를 위한 첫 단계로 우선 운영체제의 처리율과 문맥 교환, IPC의 횟수를 측정한다. 그리고, 다음 단계에서 메모리 시스템을 구성하는 L1 캐시와 L2 캐시, 그리고 TLB의 접근 실패율을 성능 메트릭(metrics)으로 삼아 보다 세밀한 측정 작업을 수행한다. 끝으로, 종합적인 캐시 메모리 성능을 지수화하기 위하여 명령어당 소요되는 사이클 수 즉, CPI(Clock cycles Per Instruction) 값을 제시한다.

4.3 실험 환경과 벤치마크

본 실험에서 사용한 하드웨어 환경은 64 MByte 주메모리와 2 GByte IDE 하드디스크가 장착된 150 MHz PCI Pentium Pro PC이며, 자세한 하드웨어 사양은 표 1과 같다.

성능 측정에 사용된 작업 부하는 SPEC에서 제공하는 SDET(System Development Environment Throughput) 버전 1.1 벤치마크 프로그램이다[15]. SDET은 UNIX 운영체제가 다중 사용자 환경 하에서 발휘하는 시스템 수준의 처리율을 측정하기 위하여 개발되었다. 본 논문에서는 SPEC SDET Kenbus 0.61 버전을 사용하였다.

표 1 수행 환경의 하드웨어 사양

프로세서	150 MHz Intel Pentium Pro
L1 캐시	8 KB instr. + 8 KB data (write through, 2-way LRU)
TLB	32 for instr. + 64 for data (4-way LRU)
L2 캐시	pipeline-burst 256KB unified write back
버스	PCI with an ISA
주 메모리	64 MByte of 60 ns RAM
하드 디스크	2 GByte IDE

4.4 측정 결과와 분석

본 절에서는 OSF/1 IK 1.1과 OSF/1 MK 7.2 상에서 SDET Kenbus 벤치마크를 실행하면서 성능을 측정한 결과를 분석한다. 실험에서 다중 프로그래밍의 효과를 반영하기 위하여 사용자 수를 1, 4, 8, 12, 16으로 증가시키면서 성능값의 변화를 측정하였다. 또한 측정 오차를 줄이기 위하여 디몬(daemon) 프로세스를 제거하고 매번 시스템을 재부팅하여 10회의 반복 실험을 실시하여 평균값을 구하였다.

앞으로 제시될 표와 그림에서 IK는 통합 커널(Integrated Kernel), 즉 모노리딕 커널을, MK는 마이크로커널을 각각 나타낸다. 또한, 캐시 및 TLB 접근 실패가 발생하는 곳을 구분하기 위하여, IK의 경우 커널과 사용자 태스크로, MK의 경우에는 커널과 사용자, 그리고 UNIX 서버에서의 측정값으로 분리하여 표시하기로 한다.

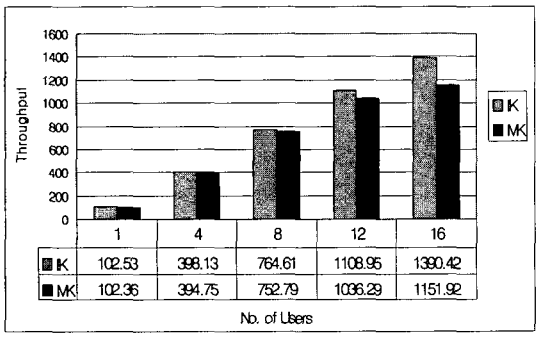
4.4.1 처리율

운영체제의 성능 평가를 위하여 가장 먼저 비교해야 할 측정값은 운영체제의 처리율이다. IK와 MK에서 벤

치마크 프로그램에 대한 처리율을 측정하면 두 운영체제의 성능을 단순 비교할 수 있다.

그림 2는 사용자 수의 변화에 따른 IK와 MK의 처리율의 변화를 보여 주고 있다. 이때, 측정값의 단위는 시간당 처리할 수 있는 벤치마크 스크립트(scripts/hour)의 수이다. 두 운영체제 모두 사용자 수가 증가함에 따라 시스템 유휴 시간(idle time)이 줄어들면서 처리율이 증가하게 되고, 사용자 수가 16에 이르면 포화상태에 접어들게 되어 처리율의 증가가 크게 둔화된다. 그림 2에서 사용자 수가 1일 때 처리율은 거의 차이가 없으나, 사용자 수가 12일 때는 MK의 처리율이 IK의 93%, 16일 때에는 IK의 83%에 불과하게 된다. 즉, 사용자 수가 증가할수록 MK의 성능이 IK에 비하여 크게 저하되는 현상을 보여준다.

제 3 장에서 지적하였듯이, 마이크로커널 기반 운영체제와 모노리딕 커널 기반 운영체제의 성능 차이와 관련된 가장 중요한 두 가지 성능 요소는 문맥 교환과 IPC이다. IK와 MK에서 IPC 발생 횟수의 차이를 측정하기 위해서는 MK의 UNIX 시스템 호출 횟수를 측정해야 한다. IK는 UNIX 시스템 호출을 트랩으로 처리하는 반면, MK는 이를 서버에서 처리하기 위하여 원격 프로시저 호출을 수행할 때 IPC가 발생하기 때문이다. 따라서, UNIX 시스템 호출 횟수가 IK와 MK 간의 IPC 발생 횟수의 차이가 된다.



	1	4	8	12	16
IK	102.53	398.13	764.61	1108.95	1380.42
MK	102.36	394.75	752.79	1036.29	1151.92

그림 2 사용자의 증가에 따른 처리율의 변화(L2 캐시 동작)

표 2는 IK와 MK에서 발생하는 UNIX 시스템 호출과 문맥 교환 횟수를 보여 주고 있다. UNIX 시스템 호출의 횟수는 IK와 MK에서 동일하며, 사용자 수에 거의 비례하여 증가하고 있으나, 문맥 교환은 IK와 MK에서 크게 다른 증가율을 보이고 있다. 즉, IK에서 문맥 교환이 완만한 증가를 보이는 반면, MK에서는 UNIX 시스템 호출과 마찬가지로 사용자 수에 거의 비례하여 급격히 증가하고 있다. 예를 들어, 사용자 수가 16일 때 MK는 IK에 비하여 무려 19배 이상의 문맥 교환을 유발하게 된다. 이것이 바로 MK에서 UNIX 시스템 호출이 문맥 교환을 유발하기 때문에 발생하는 현상이다. 그리고 이 현상은 이미 기술한 바와 같이 커널과 독립된 서버로 나뉘어진 마이크로커널의 구조적 특성에 기인한다.

표 2 시스템 호출과 문맥 교환의 발생 횟수

	사용자수	1	4	8	12	16
UNIX 시스템 호출	MK(=IK)	7,885	30,771	61,356	91,780	122,732
문맥 교환	IK	2,508	3,377	4,493	5,813	7,982
	MK	13,444	38,741	74,762	110,400	152,040

4.4.2 L1 캐시 메모리

Pentium Pro 프로세서의 L1 캐시는 명령어 캐시(ICache)와 데이타 캐시(DCache)로 분리되어 있다. ICache는 IFU(Instruction Fetch Unit)로부터 명령어들을 인출하므로, L1 명령어 캐시 실패율을 구하려면 IFU 명령어 인출 실패율을 측정하면 된다. 그림 3은 명령어당 IFU 명령어 인출 실패율, 즉 IFU 명령어 인출 실패 횟수(IFU_IFETCH_MISS)를 실행된 명령어 수(INST_RETIRED)로 나눈 값이다.

그림 3에서 주목해야 할 것은 MK가 IK에 비하여 훨씬 높은 실패율을 보이고 있다는 점이다. 사용자 수가 1일 때 MK의 실패율은 커널 0.021%, 서버 0.021%로서, 이를 합하면 IK의 커널 0.0067%에 비하여 6.3배의 실패율을 보인다. 이러한 현상은 사용자 수가 증가할수록 심각해지는데, 사용자 수가 16일 때 MK의 실패율은 1.248%(커널 0.578%+서버 0.670%)로서, IK의 0.12%에 비하여 실패율이 10.4배에 이른다.

다음으로, Pentium Pro 프로세서에서 DCU(Data Cache Unit)에 할당된 캐시 블록의 수는 데이타 캐시 접근 실패 횟수를 누적한 값과 같으므로, 이 값을 측정하면 L1 데이타 캐시 접근 실패율을 구할 수 있다. 그림 4는 DCU에 할당된 캐시 블록의 수(DCU_LINES_IN)를 INST_RETIRED로 나눈 값, 즉 명령어 당 DCU 데이타 접근 실패율이다. 그림 4에서도 MK의 실패율은 IK의 실패율에 비하여 훨씬 크다. MK의 경우 사용자 수가 1일 때 커널 0.032%, 서버

0.023%에서 사용자 수가 16일 때 커널 1.00%, 서버 0.74%로 급격히 증가한다. 커널과 서버를 합하여 IK 커널과 비교하면 사용자가 1일 때 5배, 사용자가 16일 때 8.9배가 된다.

두 실험 결과에서 L1 명령어 캐시와 데이타 캐시의 접근 실패율은 MK가 IK에 비하여 훨씬 높고, 사용자의 수가 증가할수록 MK에서 실패율의 증가가 가속화된다는 것을 알 수 있다. 사용자 수의 증가에 따른 실패율의 증가는 태스크 수가 증가할수록 태스크 간의 L1 캐시 메모리에 대한 충돌 실패가 증가하기 때문에 발생하는 현상으로 해석된다.

한편, 실험 결과 IFU_IFETCH_MISS의 최대 실패율이 1.4% 미만의 작은 값으로 나타나고 있는데, 이것은 IFU_IFETCH_MISS가 명령어를 선인출 버퍼(prefetch buffer)에서 인출하는 것을 접근 실패가 아닌 적중된 것

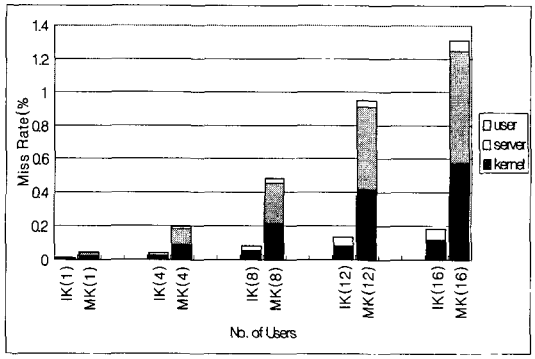


그림 3 IFU 명령어 인출 실패율(IFU_IFETCH_MISS/INST_RETIRED)

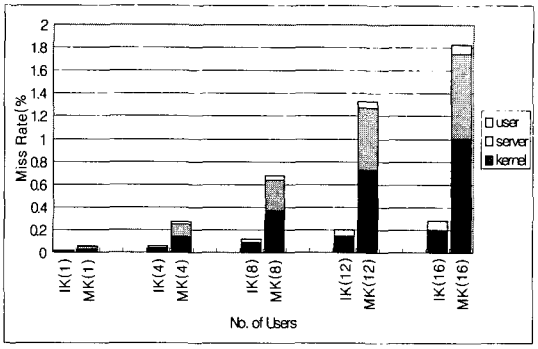


그림 4 DCU 데이타 접근 실패율(DCU_LINES_IN/INST_RETIRED)

으로 간주하기 때문이다. Pentium Pro는 512개 분기 예측 버퍼(branch target buffer)를 이용한 효율적인 분기 예측을 통하여, 명령어 인출에 실패할 확률을 크게 감소시키고 있다.

4.4.3 TLB

그림 5는 IK와 MK에서 사용자 수를 증가시키면서 명령어 TLB(ITLB) 접근 실패 횟수(ITLB_MISS)를 실행된 명령어 수로 나눈 값이다. L1 캐시 실패율의 경우와 마찬가지로, MK는 IK에 비하여 훨씬 높은 ITLB 실패율을 보인다. 즉, ITLB 실패율은 사용자 수가 1일 때 커널 0.0045%, 서버 0.0034%에서 사용자 수가 16일 때 커널 0.1320%, 서버 0.1047%로 급격하게 증가한다. 이를 합하여 IK 커널과 비교하면, 사용자 수가 1일 때 6.1배, 사용자 수가 16일 때 14.6배의 실패율을 보인다.

한편, ITLB 실패율의 증가를 L1 명령어 캐시 실패율 증가와 비교하면, 사용자 수가 1일 때 L1 명령어 캐시 6.1배와 ITLB 6.3배의 실패율을 보이던 것이, 사용자 수가 16일 때 L1 명령어 캐시 10.4배와 ITLB 14.6배로 더욱 격차가 커지는 것을 알 수 있다. 이는 마이크로커널 구조에서 태스크 수의 증가와 빈번한 주소 공간의 교체가 캐시 메모리보다 TLB에 더욱 심각한 영향을 미친다는 사실을 입증하고 있다.

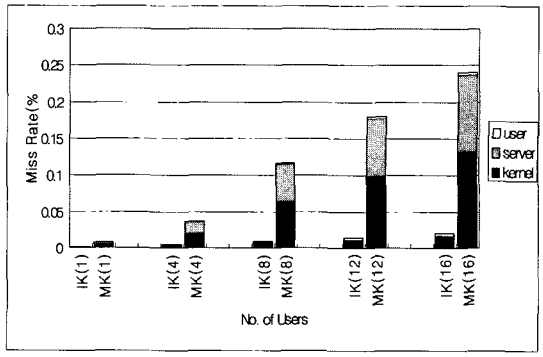


그림 5 ITLB 접근 실패율(ITLB_MISS/INST_RETIRED)

4.4.4 L2 캐시 메모리

그림 6은 L2 캐시 접근 실패율의 변화를 보여주고 있다. 이 값은 L2 캐시에 할당된 블록 수(L2_LINES_IN)를 실행된 명령어 수로 나눈 값이다. L1 캐시에서와 마찬가지로, MK의 L2 캐시 접근 실패율은 IK에 비하여 훨씬 큰 폭으로 증가한다. MK는 사용자 수가 1일 때 커널 0.0089%, 서버 0.0069%에서 사용자가 16일 때 커널 0.284%, 서버 0.261%로 급격히 증가

한다. 커널과 서버를 합하여 IK 커널과 비교하면, 사용자가 1일 때 3.1배에서 사용자가 16일 때 5.2배가 된다. 따라서, MK의 L2 캐시 접근 실패율 역시 IK 커널에 비하여 훨씬 크고, 사용자 수가 증가할수록 L2 캐시 접근 실패율이 지속적으로 증가한다는 사실을 알 수 있다.

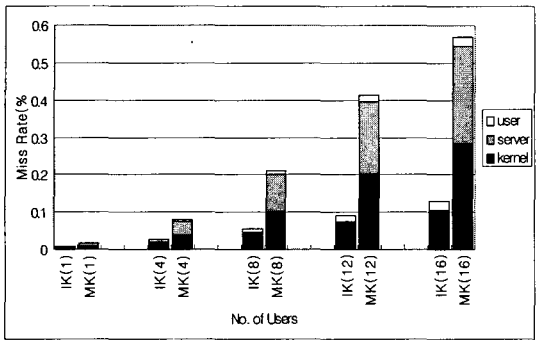


그림 6 L2 캐시 접근 실패율(L2_LINES_IN/INST_RETIRED)

4.4.5 전체적 CPI

지금까지 기술한 성능 요소들의 영향을 종합하여 메모리 시스템의 성능을 평가할 수 있는 척도는 전체적 CPI(Clock cycles Per Instruction)이다. 전체적 CPI는 프로그램을 수행하는 동안 하나의 명령을 수행하는데 소요되는 평균 프로세서 사이클 시간이다. 입출력이 배제된 동일한 하드웨어 환경과 작업 부하에서 CPI 값이 크다는 것은 메모리 시스템의 효율성이 낮다는 것을 의미하며, 시스템의 성능이 저하되는 결과를 가져온다.

그림 7은 벤치마크 프로그램 수행 시 소요된 전체 사이클 시간(CPU_CLK_UNHALTED)을 실행된 명령어의 수로 나눈 값, 즉 전체적 CPI가 된다. 그림 7에서 IK 커널의 경우 전체적 CPI 값은 사용자 수가 1일 때 0.51, 사용자 수가 16일 때 0.62로서, 완만한 증가를 보인다. CPI 값이 1 보다 작은 이유는 Pentium Pro 프로세서가 3개의 실행 유닛(execution unit)을 갖는 수퍼스칼라(superscalar) 구조로 이루어져 있기 때문이다. 그러나, MK에서 CPI 증가율은 IK에서의 증가율보다 훨씬 크다. MK에서 측정된 CPI는 사용자 수가 1일 때 커널과 서버가 0.02와 0.57이고, 사용자가 16일 때 각각 0.62, 0.87로서 하나의 명령을 수행하는데 약 1.6 사이클이 소요된다.

그림 7에서 MK와 IK에서의 CPI 값을 비교하면, 동일한 하드웨어와 작업 부하에서 동작하는 두 운영체제의 CPI 값에 큰 차이가 있으며, 그 차이는 사용자의 수가 증가할수록 더욱 커진다는 것을 발견할 수 있다. 즉, MK에서 측정된 커널과 서버를 합한 CPI 값은 0.59로서, IK의 CPI 값 0.51에 비하여 단지 약 1.2배 큰 값이다. 그러나, 사용자 수가 16일 때에는 MK의 커널과 서버의 CPI 값은 1.48로서, IK의 CPI 값 0.62에 비하여 약 2.4배 큰 값이다. 따라서, MK는 IK에 비하여 전체적 CPI가 크고, 그 격차는 사용자의 수가 증가할수록 더욱 벌어진다는 사실을 알 수 있다.

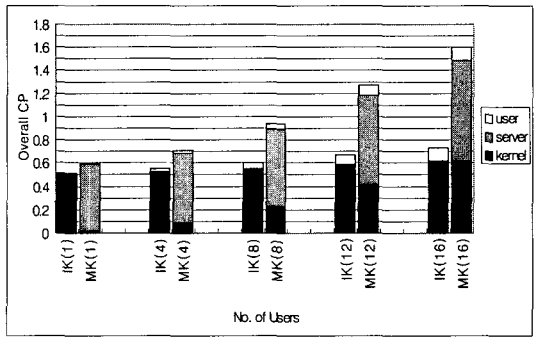


그림 7 전체적 CPI(CPU_CLK_UNHALTED/INST_RETIRED)

4.5 캐시 메모리와 IPC의 비중

지금까지의 실험 결과, MK가 IK에 비하여 성능이 저하된다는 점과, 그 원인이 상당 부분 캐시 메모리 성능 저하에 기인한다는 사실을 알 수 있다. 그러나, 마이크로커널에서 발생하는 성능 저하의 원인 중에서 캐시 메모리가 차지하는 비중을 평가하기 위해서는 다른 성능 요인과 비교 평가하는 실험을 추가로 필요로 한다.

본 절에서는 지금까지 마이크로커널 성능 저하의 주요 요인으로 알려져 왔던 IPC와, 새로운 원인으로 제시된 캐시 메모리에 의한 성능 저하의 비중을 비교 평가한다. 두 가지 요인을 비교 평가하는 방법의 하나는 동일한 환경에서 캐시 메모리를 동작시킨 상태와 정지시킨 상태에서 성능을 측정하여 비교하는 것이다. 캐시 메모리를 정지시키더라도 사용자 태스크에서 호출하는 시스템 호출의 수, 즉 IPC의 수에는 변함이 없다. 따라서, 캐시 메모리를 동작시킨 상태와 정지시킨 상태에서 MK의 처리율과 CPI를 측정하였을 때 두 측정값이 크게 다르다면, 이 차이가 캐시 메모리의 성능이 마이크로커널의 성능 저하에 미치는 영향력을 보여주는 것이라고 할 수 있다. 그리고 만일 캐시 메모리를 정지시킨 상태에서 측정한 IK와 MK의 성능값의 차이가, 캐시 메모리를 동작시킨 상태에서 측정한 IK와 MK의 성능값의 차이보

다 크다면, 캐시 메모리에 의한 성능 저하의 비중이 IPC에 의한 성능 저하보다 크다고 할 수 있을 것이다. 본 절에서는 현대 하드웨어 구조에서 캐시 메모리의 크기가 점차 커지고 있는 점을 감안하여, L2 캐시를 정지시킨 상황에서 성능을 측정하였다.

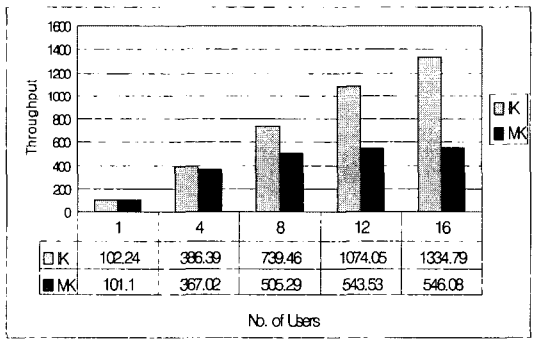
4.5.1 L2 캐시 정지 상태에서의 처리율

그림 8은 L2 캐시를 정지시킨 상태에서 사용자의 수를 증가시키면서 측정한 IK와 MK의 처리율의 변화를 보여주고 있다. 그림 8에서 IK의 경우 사용자 수가 1일 때 102.24, 사용자 수가 16일 때 1334.79로 처리율이 지속적으로 증가하고 있다. 이 값은 앞 절에서 L2 캐시가 동작할 때 측정한 IK의 처리율과 비교하였을 때 96%에 해당하므로, IK의 성능은 L2 캐시 메모리의 역할에 크게 영향받지 않는다는 것을 알 수 있다.

반면, MK의 경우는 IK의 경우와 크게 다르다. 즉, MK는 사용자 수가 1일 때 101.10으로 IK와 비교하여 거의 차이가 없으나, 사용자 수가 8일 때부터 시스템이 포화 상태에 이르게 되어 사용자 수가 16일 때 처리율이 546.08에 불과하다. 이 처리율은 IK에 비하여 절반에도 미치지 못하는 값이다. 따라서, MK의 경우 IK에 비하여 훨씬 먼저 시스템 포화 상태에 이르게 되며, 사용자 수가 증가함에 따라 현격한 처리율의 차이를 보인다는 사실을 알 수 있다.

실험 결과를 통하여 알 수 있는 중요한 사실은, MK에서 IPC에 의한 성능 저하보다는 캐시 메모리에 의한 성능 저하의 비중이 절대적으로 우세하다는 것이다. 이러한 사실은 그림 2와 그림 8을 비교하면 알 수 있다. 그림 2에서 L2 캐시가 동작하고 있을 때 MK의 IK에 대한 처리율의 비율은 사용자 수가 1일 때 99.8 (=102.36/102.53)%, 사용자 수가 16일 때 82.9 (=1151.92/1390.42)%이다. 즉, IK에 대한 MK의 성능 저하가 모두 IPC에 기인한 것이라 가정하면 IPC에 의한 성능 저하의 상한은 17.1 (=100.0-82.9)%가 되는 것이다. 이에 반하여, 그림 8에서 보는 바와 같이 MK에서 L2 캐시를 동작시켰을 때와 정지시켰을 때 처리율의 비율은 사용자 1일 때 98.8 (=101.10/102.36)%, 사용자 수가 16일 때 47.4 (=546.08/1151.92)%로 급격히 하락한다. 즉, MK의 성능 저하가 모두 L2 캐시를 정지시킨 것에 기인한 것이라 가정하면 L2 캐시 메모리에 의한 성능 저하의 상한은 52.6 (=100.0-47.4)%가 되는 것이다. 따라서, 사용자 수가 증가할수록 마이크로커널의 성능 저하에 가장 큰 요인을 미치는 것은 IPC에 의한 영향이 아니라, 캐시 메모리의 영향이라는 사실을 알 수 있다.

실험을 통하여 알 수 있는 또 하나의 사실은 IK의 성능이 L2 캐시의 동작 여부에 거의 영향을 받지 않음에 반하여, MK의 성능은 L2 캐시의 동작 여부에 큰 영향을 받고 있다는 것이다. 그림 2에서 L2 캐시가 동작하고 있을 때 MK의 IK에 대한 처리율의 비율은 사용자 수가 1일 때 99.8%, 사용자 수가 16일 때 82.9%이다. 그러나, 그림 8에서 L2 캐시의 동작을 정지시키면 MK의 IK에 대한 비율이 사용자 1일 때 99.8에서 사용자 수가 16일 때 40.9%로 급격히 하락한다. 따라서, 사용자 수가 8 이상일 때 MK의 성능의 L2 캐시 메모리 의존도는 절대적이라 할 수 있다. 즉, 마이크로커널에서 메모리 참조 요청의 상당 부분이 L1 캐시에서 처리할 수 없으며, 이 요청이 L2 캐시에게 전달됨으로써 L2 캐시의 대역폭을 크게 잠식하고 있다는 사실을 알 수 있다.



	1	4	8	12	16
IK	102.24	386.39	739.46	1074.05	1334.79
MK	101.1	367.02	505.29	543.53	546.08

그림 8 L2 캐시 정지 상태의 처리율

4.5.2 L2 캐시 정지 상태에서의 전체적 CPI

L2 캐시를 정지시킨 상태에서 측정한 처리율의 저하가 캐시 메모리의 효율성의 저하에서 비롯된 것이라는 것을 확인하기 위하여 전체적 CPI 값을 측정하였다. 그림 9는 L2 캐시를 정지시킨 상태에서 측정한 전체적 CPI 값이다. 그림 9에서 IK는 사용자 수가 1일 때 커널 0.517, 사용자 0.001에서 사용자 수가 16일 때 커널 0.937, 사용자 0.302로 완만하게 증가하는 반면, MK의 전체적 CPI는 사용자 수가 증가함에 따라 수 배로 급격히 증가하고 있다. 즉, MK의 CPI는 사용자 수가 1일 때 커널 0.063, 서버 0.606에서 사용자 16일 때 커널 3.030, 서버 3.066으로 증가한다. 이 증가율을 그림 7의 L2 캐시가 동작할 때와 비교하면, 커널과 서버를 합산한 전체적 CPI가 사용자 수가 1일 때 약 13%, 그리고 사용자 수가 16일 때 무려 310%가 증가한 값이다. 즉 MK에서 사용자 수가 16일 때, L2 캐시의 동작을 정지

시켰을 때의 전체적 CPI는 L2 캐시를 동작시켰을 때보다 4배가 넘는 전체적 CPI 값을 갖는다. 이러한 전체적 CPI 값의 증가는 결국 MK에서 성능이 저하되는 결과로 나타난다.

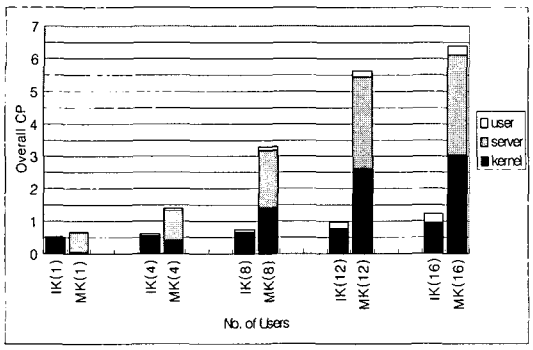


그림 9 전체적 CPI(L2 캐시 정지)

4.6 고찰

지금까지 IK와 MK 운영체제 환경에서 캐시 메모리와 TLB 성능값을 측정하고 결과를 분석하였다. 실험 결과 모든 경우에서 MK가 IK에 비하여 캐시 메모리의 성능이 크게 저하되는 것으로 나타났다. 우선, L1 캐시의 명령어 인출 실패율과 데이타 접근 실패율에서 MK는 IK에 비하여 최고 9~10배의 실패율을 보였다. 그 구체적인 이유로는 워킹셋의 크기와 지역성의 저하를 들 수 있다. 즉, MK에서 마이크로커널과 서버 태스크의 프로그램 크기를 합하면 IK 커널에 비하여 크기가 크고, 운영체제의 기능이 마이크로커널과 서버들에게 분산되어 있으므로 참조하는 워킹셋의 크기도 훨씬 크다. 그리고, MK는 시스템 호출을 에뮬레이션하기 위하여 제어의 흐름이 사용자 태스크, 마이크로커널, 서버 태스크, 에뮬레이션 라이브러리로 빈번히 변화하므로 지역성이 크게 저하된다.

또한, MK와 IK의 L1 캐시 접근 실패율의 차이는 사용자의 수가 증가할수록 더욱 심각하였다. 이것은 사용자 태스크의 수가 증가할수록 워킹셋의 수가 증가하게 되고, 이들 사이에서 충돌 실패가 증가하기 때문이다. 그리고, 수행 중이던 태스크에 의해 캐시 메모리에 적재된 워킹셋이 대체되는 비율은 태스크의 수행이 재개되기 전까지 수행되는 제 3의 태스크의 수에 대체로 비례하므로, 시스템 내에서 활동중인 태스크의 수가 증가할수록 콜드 스타트 미스(cold-start miss)에 의한 캐시 접근 실패율이 증가하기 때문이다. 이러한 현상은 여러 개의 서버 태스크가 동작하는 다중 서버 시스템에서 더욱 심각할 것으로 예측된다.

한편, ITLB 접근 실패율을 측정한 결과에서도 MK가 IK에 비하여 최고 14배 이상 높았으며, 사용자의 수가 증가할수록 더욱 격차가 벌어지는 추세를 보였다. 이는 MK에서 문맥 교환에 의해 주소 공간의 교체가 빈번히 발생하게 되고, 이 때마다 TLB에 적재해 놓았던 모든 가상 메모리 페이징 정보를 무효화시키기 때문이다. 일반적으로 ITLB 접근 실패율은 캐시 메모리의 접근 실패율에 비하여 절대적인 값이 작지만, 접근 실패가 발생할 때마다 주 메모리 내의 페이지 테이블을 한 번 이상 접근해야 하기 때문에 성능 저하에 미치는 영향은 캐시 메모리에 비하여 훨씬 크다.

마찬가지로 L2 캐시 접근 실패율에서도 MK의 커널과 서버는 IK 커널에 비하여 3~5배에 이르는 접근 실패율을 보였다. L2 캐시 접근 실패는 주 메모리로의 접근을 발생시키는데, 일반적으로 Pentium Pro에서 주 메모리에 대한 접근 시간은 최소 40 사이클 이상인 것으로 알려져 있다. 따라서, L2 캐시 메모리에 대하여 3~5배의 L2 캐시 접근 실패율을 보이는 것은 MK와 IK의 메모리 시스템 성능에 큰 차이를 가져오는 중요한 요소가 된다.

한편, IK와 MK 간의 ITLB 실패율의 격차가 증가하는 속도는 L1 캐시 접근 실패율의 격차가 증가하는 속도보다 더욱 빠르다는 사실을 알 수 있다. 이것은 마이크로커널 구조에서 태스크 수의 증가와 빈번한 주소 공간의 교체가 미치는 영향이, 물리적 인덱스(physical index)를 갖는 L1 캐시 메모리보다 가상 인덱스(virtual index)를 갖는 ITLB에 더욱 심각하게 작용한다는 사실을 입증하고 있다. 즉, 가상 주소를 인덱스로 하는 TLB는 태스크 주소 공간의 교체가 발생할 때마다 플러시 연산을 수행하여 TLB 내의 모든 데이타를 무효화시키게 된다. 따라서, 빈번한 문맥 교환으로 주소 공간을 자주 교체하는 마이크로커널 기반 운영체제의 경우 TLB 성능이 크게 저하되고, 이는 결국 운영체제의 성능 저하로 나타난다.

캐시 메모리와 TLB 성능 저하는 결국 전체적 CPI 값의 증가와 시스템 성능의 저하로 귀결된다. 측정 결과 IK는 최고 0.62, MK는 최고 1.48 CPI를 기록하였고, 태스크의 수에 따라 점차 증가하는 추세를 보였다. 동일한 하드웨어 환경과 작업 부하에서 월등히 높은 CPI를 보이는 MK는 IK에 비하여 당연히 낮은 성능을 보이게 된다. 그리고, 그 원인은 MK의 구조적 특성으로 인한 메모리 시스템의 효율성 저하에 기인한다는 것을 확인할 수 있다.

마지막으로, L2 캐시를 정지시킨 상태에서 IK와 MK의 성능 변화를 측정함으로써 마이크로커널 기반 운영체제의 성능 저하에서 IPC의 부담과 캐시 메모리 성능이 차지하는 비중을 비교하여 평가하였다. 측정 결과 시스템 호출의 수, 즉 IPC의 횟수에 차이가 없는 상황에서 MK의 처리율은 IK의 절반에도 미치지 못하였으며, MK의 CPI값은 IK에 비하여 4배 이상으로 증가하는 결과를 보였다. 이 결과는 MK의 성능 저하 원인 중에서 IPC의 비중보다 캐시 메모리 성능에 의한 비중이 훨씬 크다는 사실을 입증하고 있다. 따라서, 마이크로커널 구조에서 발생하는 성능 저하 현상의 근본 원인이 캐시 메모리의 성능 저하에 있음을 확인할 수 있다.

성능 분석을 통하여 성능 저하의 근본적인 원인이 UNIX 시스템 호출을 처리하는 서버로의 문맥 교환과, 이에 수반되는 캐시 메모리 효율성의 저하임을 확인하였다. 따라서, UNIX 서버와 같이 자주 사용되는 서버들을 마이크로커널과 동일한 주소 공간 내에 배치하는 병치 서버(colloacted server) 등의 기법을 사용함으로써, 문맥 교환의 빈도를 줄이고 캐시 메모리의 성능을 개선하는 방법이 고려될 수 있을 것이다.

본 논문은 운영체제와 하드웨어 간의 상호 작용이 전체 시스템 성능에 미치는 영향을 실제로 동작하는 시스템 상에서 정량적으로 측정하고 분석하였다는 점에서 큰 의미가 있다. 이러한 실험 결과는 마이크로커널 기반 운영체제의 차세대 운영체제로서의 적합성을 평가하는데 중요한 자료로 사용될 수 있으며, 마이크로커널 기반 운영체제의 성능을 개선하기 위한 바람직한 방향을 제시하고 있다.

본 논문은 또한, 점차 모듈화 되어 가는 현대 소프트웨어 구조의 변화가 하드웨어 기술 발전 추세와의 상호 작용을 통하여 컴퓨터 시스템의 성능에 어떠한 영향을 미칠 것인가에 대한 개략적인 예측을 가능케 한다. 세분화된 모듈 구조는 캐시 메모리의 성능을 저하시킴으로써, 메모리 시스템의 효율성에 절대적으로 의존하고 있는 고속 프로세서의 성능을 크게 저하시킬 것으로 예상된다. 따라서, 모듈화된 소프트웨어를 개발하는 경우 메모리 효율성을 개선하는데 보다 많은 노력을 기울여야 할 것이다.

5. 결론 및 향후연구과제

마이크로커널 기반 운영체제의 모듈화된 구조는 소프트웨어 공학적인 측면에서 여러 가지 장점들을 가지고 있으나, 운영체제의 성능을 저하시키는 원인으로 작용하기도 한다. 본 논문에서는 마이크로커널 기반 운영체제에서 발생하는 성능 저하의 근본적인 원인을 규명하기 위하여 운영체제의 구조적인 특성이 캐시 메모리 성능에 미치는 영향을 정량적으로 분석하였다. 이를 위하여 Intel Pentium Pro 프로세서 환경에서 개발된 성능 측정 도구 MKperf를 이용하여 모노리딕 커널 기반 운영체제인 OSF/1 IK와 마이크로커널 기반 운영체제인 OSF/1 MK를 대상으로 L1, L2 캐시와 TLB의 성능을 측정하고 결과를 비교 분석하였다.

성능 측정 결과, MK는 IK에 비하여 L1, L2 캐시와 TLB 접근 실패율이 월등히 높으며, 따라서 높은 CPI 값을 갖는다는 사실을 발견하였다. 또한, 독립된 주소 공간을 갖는 서버 구조는 L2 캐시의 대역폭을 크게 소모함으로써, 캐시 메모리의 성능이 기존의 문제점으로 지적되었던 IPC의 부담보다 더욱 큰 비중으로 마이크로커널 기반 운영체제의 성능을 좌우하게 된다는 사실을 발견하였다. 그리고, 이러한 현상들은 모듈화를 지향하는 마이크로커널의 구조적 특성과, 이로 인하여 빈번히 발생하는 문맥 교환의 영향임을 확인하였다.

모듈화된 구조를 지향하는 소프트웨어 개발 분야의 추세와 메모리 효율성에 대한 의존도가 커지는 하드웨어 기술 분야의 추세는 앞으로도 지속될 것으로 보인다. 본 논문은 이러한 소프트웨어 및 하드웨어의 기술 발전의 영향을 강조함으로써 향후 고성능 컴퓨터 시스템을 개발하는데 있어 두 기술의 상호 작용을 고려하여 시스템의 성능을 제고하는데 기여할 것으로 기대된다.

참 고 문 헌

[1] Alan J. Smith, "Cache Memories," *ACM Computing Survey*, Vol. 14, No. 3, Sep. 1982.

[2] Allan Bricker et al., "A New Look at Microkernel-based UNIX Operating Systems: Lessons in Performance and Compatibility," *Proc. of Winter USENIX Conference*, 1991.

[3] J. Bradley Chen and Brian N. Bershad, "The Impact of Operating System Structure on Memory System Performance," *Proc. of 14th ACM SOSP*, 1993.

[4] Richard Uhlig et al., "Design Tradeoffs for Software-Managed TLBs," *ACM Trasactions on Computer Systems*, Vol. 12, No. 3, Aug. 1994.

[5] Jochen Liedtke, "Toward Real Microkernels," *CACM*, Sep. 1996.

[6] Michael Condict et al., "Microkernel Modularity with Integrated Kernel Performance," Operating Systems Collected Papers, Vol. 3, OSF RI, 1994.

[7] Thomas E. Anderson et al., "The Interaction of Architecture and Operating System Design," *Proc. of ASPLOS IV*, 1991.

[8] John K. Ousterhout, "Why Aren't Operating Systems Getting Faster As Fast As Hardware ?," *Proc. of Summer USENIX Conference*, 1990.
[9] Anant Agarwal, John Hennessy, and Mark Horowitz, "Cache Performance of Operating System and Multiprogramming Workloads," *ACM Transactions on Computer Systems*, Vol. 6, No. 4, Nov. 1988.
[10] Jeffrey C. Mogul and Anita Borg, "The Effect of Context Switch on Cache Performance," *Proc. of ASPLOS IV*, 1991.
[11] John Hennessy and David Patterson, *Computer Architecture A Quantitative Approach*, 2nd Ed., Morgan Kaufmann, 1996.
[12] Richard P. Draves et al., "Using Continuations to Implement Thread Management and Communication in Operating Systems," *Proc. of 13th SOSP*, 1991.
[13] Brian N. Bershad, "The Increasing Irrelevance of IPC Performance for Microkernel-Based Operating Systems," *Proc. of 1st USENIX Workshop on Microkernels and Other Kernel Architectures*, 1992.
[14] Intel, *Pentium Pro Family Developer's Manual*, 1996.
[15] Ran Giladia and Niv Ahituv, "SPEC as a Performance Evaluation Measure," *IEEE COMPUTER*, Aug. 1995.

장 문 석

1990년 서울대학교 계산통계학과(학사). 1992년 서울대학교 계산통계학과(석사). 1998년 서울대학교 전산과학과(박사). 1998년 ~ 현재 한국전자통신연구원 운영체제연구팀(선임연구원). 관심분야는 운영체제, 컴퓨터구조, 병렬처리

고 건

서울대학교 공과대학 응용물리학과를 졸업하고 미국 버지니아 대학교에서 1981년 전산학 박사학위를 받음. 1974년 ~ 1976년 KIST 연구원. 1981년 ~ 1983년 Bell 연구소 연구원. 1988년 1월부터 1년간 IBM 왓슨 연구소에서 분산 운영체제에 대해 연구함. 1991년 ~ 1993년 서울대학교 중앙교육연전산원 원장. 1994년 한국정보과학회 학회지편집위원장 역임. 1983년 ~ 현재 서울대학교 전산과학과 교수로 재직중. 관심분야는 운영체제, 컴퓨터 구조, 컴퓨터 시스템 성능 분석, 분석처리 시스템 등임.