주문형 비디오 서비스를 위한 실시간 스케쥴링 기능

손 종 문[†] · 김 길 용[††]

요 약

본 논문에서는 주문형 비디오 서버가 필요로 하는 운영체제의 실시간 스케쥴링 기능을 분석 및 구현하였다. 실시간 스케쥴링 요구 조건은 비디오 데이타 전달 경로에 대한 모델 분석을 통하여 수집되었다. 특히, 병목 현상을 일으키는 하부 시스템이 전체 시스템의 실시간 스케쥴링에 미치는 영향을 분석함으로써 비디오 데이타 처리에 적합한 실시간 스케쥴러 및 프리미티브를 구현하였다.

성능 측정에서는 구현된 실시간 스케쥴러의 보장성을 실험하였다. 측정된 데이타는 프로세스가 가진 대부분의 시간 제약 조건이 만족됨을 보였다. 그러나 인터럽터 방식의 네트워크 프로토콜 처리는 실시간 스케쥴링의 가장 큰 장애 요소이다. 또한, 프로세스 수행 시간 간격을 측정함으로써 비실시간 스케쥴러와 실시간 스케쥴러의 차이점을 비교하였다. 측정된 결과에 의하면 비실시간 스케쥴러를 사용하면 프로세스에 할당되는 프로세서 시간을 예측하기 어렵기 때문에 효율적인 비디오 서비스를 위해서는 반드시 실시간 스케쥴러가 사용되어져야 함을 보였다.

Real-Time Scheduling Facility for Video-On-Demand Service

Jong Moon Sohn[†] · Gil Yong Kim[††]

ABSTRACT

In this paper, the real-time facility of the operating system for a VOD(Video On Demand) server have been analyzed and implemented. The requirements of the real-time scheduling have been gathered by analyzing the model of the video-data-transfer-path. Particularly, the influence of the bottleneck subsystem have been analyzed. Thus, we have implemented the real-time scheduler and primitives which is proper for processing the digital video.

In performance measurements, the degree of the guarantee of the real-time scheduler have been experimented. The measured data show that the most time constraints of the process is satisfied. But, the network protocol processing by the interrupt is a major obstacle of the real-time scheduling. We also have compared the difference between the real-time scheduler and the non-real-time scheduler by measuring the inter-execution time. According to the measured results, the real-time scheduler should be used for efficient video service because the processor time allocated to the process can't be estimated when the non-real-time scheduler is used

1. 서 론

주문형 비디오(Video-On-Demand) 시스템은 멀티미디어 데이타베이스, 주문형 비디오 서버, 고속 기간망(high-speed backbone network), 액세스망(access network), STB(Set Top Box) 등으로 구성되어 있다. 이는 온라인 멀티미디어 서비스의 대표적인 형태로서 네트워크를 통한 영화 감상 뿐 아니라 뉴스 및 운동 경기 시청, 온라인 교육 및 홍보, 대화형 비디오 게임 등

† 정 회 원:시스템 공학 연구소 분산 컴퓨팅 연구실
†† 정 회 원:부산대학교 컴퓨터공학과
논문접수:1996년 7월 31일, 심사완료:1997년 8월 22일

그 활용 범위가 매우 넓다. 주문형 비디오 서비스는 사용자의 관점에서 볼 때, 시청 가능한 채널의 수를 대폭적으로 늘려주는 효과를 제공하며, 방영 스케줄에 맞춰 시청할 필요없이 각자의 스케줄에 맞출 수 있고 반복 시청 등이 가능하며, 물리적 이동을 필요로 하지 않는 등 다양한 편의를 제공한다. 이 중 비디오 서버는 그 규모에 따라 N-Cube같은 대형 병렬 처리 컴퓨터, 중소형급 또는 워크 스테이션 등을 사용하여 구현할 수 있다. 다수의 시청자에 대하여 압축된 대용량의 멀티미디어 데이타 스트림을 실시간으로 제공하기 위하여 공통적으로 필요한 기능은 다음과 같이 요약할 수 있다[22].

첫째, 대용량 저장 장치(High Storage Capacity): 상영 시간이 2시간인 영화는 약 1 GBytes의 MPEG-1 화일 또는 약 4.5 GBytes의 MPEG-2 화일로 압축될 수 있다. 비디오 서버는 이러한 대용량 데이타의 저장 및 검색을 효율적으로 수행해야 한다.

둘째, 응답 시간의 최소화(Minimum Response Time): 주문형 비디오 시스템은 사용자의 대화형 조작에 대해서 빠른 응답을 보여야 한다.

셋째, Quality Of Service(QoS)기능의 제공: 사용자의 QoS 요구에 맞는 효율적인 서비스를 제공할 수 있어야 한다.

넷째, 비용 효과적 측면 고려(Cost Effective): 비디오 서버는 실시간 전송 요구를 만족하면서 최대한의 스트림 수를 유지할 수 있어야 한다.

다섯째, 실시간 요구(Real-Time Requirements)의 보장: MPEG-1 스트림은 1.5 Mbps, MPEG-2 스트림은 6 Mbps (HDTV type) 등의 실시간 전송률을 보장해 주어야 한다.

여섯째, 스트림 개념을 지원하는 응용 프로그램 인터페이스(Application Program Interface-API) 제공: 스트림에 대하여 재생(play), 일시 정지(pause), 고속 감기(fast forward), 되감기(rewind)등의 대화형 조작을 지원하여야 한다.

이것을 기술적 측면에서 살펴보면 기존의 시분할 응용 또는 실시간 응용이 요구하는 계산의 정확성, 엄격한 시간 제약 조건을 모두 완화하여 적용할 수 있지만 근본적으로 고성능 하드웨어와 효율적인 운영체제를 필요로 한다. 즉, 디지탈 비디오 데이타를 처리하기 위한 자원이 부족할 경우에는 QoS를 낮추어 제공할 수 있고, 시간 제약 조건을 엄격히 지키지 못하더라도 전체적인 수행에는 큰 영향을 미치지 않는 것이 특징이다. 최근들어 디지탈 비디오 데이타와 같은 연속 매체 응용의 증가에 따라 이러한 특성을 지원하기 위한 컴퓨팅 환경의 범용화가 요구되고 있는데, 기존의 범용 운영체제는 낮은 성능과 보장된 서비스(guaranteed service)의 미비로 인하여 연속 매체 처리의 요구 조건을 만족시키지 못하고 있다. 이에 따라 범용 운영체제 환경에서 연속 매체를 처리하기 위한 자원 관리 기법의 개발이 요구되고 있으며, 하드웨어 환경이 충분한 컴퓨팅 파워를 제공해 주지 못할 경우 자원 관리 기법이 더욱 중요시 되고 있다.

연속 매체 운영체제가 지원해야 하는 기능은 연속 매체 프로세스의 수행을 일반 프로세스의 수행과 분리하고 처리기 용량을 예약하는 실시간 스케줄링 기능, 대용량 데이타를 저장하는 연속 매체 화일 시스템을 비롯한 대용량 데이타 처리 기능, 응용 프로그램에 대한 인터페이스를 제공하는 멀티미디어 API 등 크게 세 가지로 분류할 수 있다. 기존의 실시간 응용의 경우 수행 환경이 비교적 단순하면서 시간 제약을 엄격히 지킬 것을 요구하지만 연속 매체 응용은 수행 환경이 보다 복잡하고 유연성 있는 시간 제약 조건을 가진다. 그래서 연속 매체 응용이 사용하는 시스템 자원의 양과 시간 제약 조건은 동적으로 표현될 수 있고, 이는 운영체제의 자원 관리 기능에 자원 예약과 QoS 개념의 도입을 필요로 한다[7, 8, 10]. 특히 범용 운영체제는 연속 매체뿐만 아니라 문자, 이미지와 같은 이산 매체(discrete media)도 함께 처리하기 때문에 연속 매체의 처리를 보장하기 위해서 예약 기법은 필수적인 것이라 할 수 있다.

연속 매체의 처리가 자원 예약 기법을 기반한 동적 할당이 필요한데도 불구하고 기존의 연구에서 이러한 기법을 적용하지 않은 것은 다음과 같은 이유를 들 수 있다. 첫째, 현 시점에서 자원 이용률은 상업적 측면에서 중요한 데이타가 아니다. 즉 네크워크 대역폭과는 달리 시스템이 필요하는 자원은 사적으로 확장이 가능하다는 점과 프로그램 수행 기록을 유지, 관리하는 것은 스케줄러 구조를 복잡하게 만든다. 둘째, 연속 매체의 처리에 대한 연구는 운영체제 수준

보다 미들웨어 또는 응용 프로그램 수준에서 주로 수행되어지고 있다는 점을 들 수 있다. 셋째, 연속 매체 응용은 그 성질에 따른 기능적 분할로 인하여 실제로 한 시스템에서 다양하게 사용되지 않는다.

본 연구에서는 주문형 비디오 시스템을 구현하기 위하여 필요한 실시간 스케쥴링 요소를 밝히고 자원 예약과 QoS개념을 포함한 실시간 프리미티브를 설계하였다. 주문형 비디오 서버는 펜티엄 처리기, SCSI (Small Computer System Interface), 그리고 패스트이쓰넷(FastEthernet) 카드를 가진 화일 서버급 범용 시스템을 사용하였다. 이러한 범용 시스템하에서 10개 정도의 세션을 서비스하기 위하여 필요한 실시간 스케쥴링 기능을 구현하고, 시스템 자원을 효율적으로 사용하기 위한 시스템 파라메터들을 분석하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 관련된 기존 연구들에 대하여 기술한다. 그리고, 3장에서는 비디오 서버의 모델을 소개하고 비디오 서버가 가지는 스케쥴링 측면에서의 특성에 대하여 분석하였다. 4장에서는 분석된 내용을 기반으로 설계한 실시간 프리미티브, 내부/외부 데드라인에 대하여 기술하였다. 5장에서는 스케쥴러의 구현에 대하여 살펴보고 6장에서는 구현된 서버에 대한 실시간 만족성에 대한 성능 측정치를 논의하였다. 마지막으로 7장에서는 본 논문의 결론을 검토하고 앞으로의 연구 방향에 대하여 기술하였다.

2. 관련 연구

기존 연구 [25]는 독립된 디스크에 영화를 저장하는 비디오 서버 시스템을 제안하였고, [26]은 실리콘 그래픽스 오닉스(Silicon Graphics Onyx) 시스템에 실제 구현된 비디오 서버의 성능에 대하여 연구하였다. [27]은 비디오 서버를 설계와 다중 매체들의 동기화를 위한 메카니즘을 제안하고 있다. 이와 같은 기존의 연구들은 대부분 저장 매체의 구조와 시스템의 최적화를 통하여 수용할 수 있는 가입자를 성능 척도로 사용하고 있다. 그리고 연속 매체 처리를 위하여 운영체제에 의해 지원되어야 할 기능은 [23], [24]에서 잘 정리되어지고 있다. 이와 더불러 실시간 스케쥴링에 대한 대표적인 연구들은 DASH[9], Chorus[6], Real-Time Mach[1, 7], Real-Time Spring[15] 등을 들 수 있다.

DASH에서는 처리기, 디스크, 네트워크 등의 자원을 예약하기 위한 인터페이스나 작업부하를 기술하기 위하여 CM 자원 모델을 사용한다. 작업부하는 메시지로 표현되며 최대 메시지 크기, 최대 메시지 율, 선작업 제한가 파라메터이다. 자원의 할당은 최악의 경우를 기반으로 하여 승인 제어(admission control)이 수행되며 어떠한 상황에서도 예약된 자원의 사용은 엄격히 보장한다.

그리고 예약된 후 사용되지 않는 자원은 돌발적인 비실시간 태스크나 다른 목적으로 사용될 수 있음을 밝히고 있다. 또한 DASH는 고속의 데이타 처리를 위하여 사용자와 커널간의 상호작용을 줄이는데 노력했다. 기존 운영체제에서 스케쥴링, IPC를 수행하기 위한 커널과 사용자 주소 공간들 간의 상호작용이 많은 부담중의 하나이므로, 이를 제거한 "Split-level" 스케쥴러를 제공한다.

CMU[1, 7]에서는 멀티미디어 응용을 지원하기 위하여 예약을 기반으로 QoS 서버를 사용한다. Real-Time Mach는 경성 실시간 처리 기법를 연속 매체 처리에 적용하고 있고 경성 실시간과 연속 매체 처리 기반을 통합하여 제공하려 하고 있다. 이는 처리기 용량 예약을 기반으로 하는데 Real-Time Mach는 이를 "reserve"라 한다. "reserve"는 프로세서 용량(processor capacity)를 할당받기 위한 인터페이스로써 프로그램이 수행되는 주기와 어떤 주기마다 소요되는 처리기 시간을 {최대치(maximum), 희망치(desirable), 최소치(minimum)}으로 기술함으로써 처리기 사용을 예약할 수 있게 한다. 스케쥴러는 이 정보를 이용하여 예약용량을 보호하여 특정 프로그램에 의한 자원 독점을 방지한다. 예약된 처리기 용량은 IPC를 통한 호출시 전달되어 클라이언트 프로그램에 의해 수행되는 서버의 처리기 시간도 포함한다. Real-Time Mach의 연속 매체 지원 기법은 쓰레드가 스스로 자신이 사용할 처리시 시간을 제시한다. 그러나 실제 시스템 운영에 있어서 자신이 수행하고자 하는 쓰레드의 자원 사용량을 측정하는 것은 연속 매체 처리의 특성과는 거리가 멀다고 할 수 있다.

Real-Time Spring[15]은 경성 실시간 응용과 연속 매체 응용을 통합하여 처리함으로써 시스템 구축 비용을 절감할 수 있다는 전제하에 설계되어 분산 다중 처리기 시스템에서 동작한다. 스케쥴러는 모든 태스

크에 대한 최악의 수행 시간에 필요한 자원량을 보장하는 동적인 계획 기반 스케쥴러(planning-based scheduler)이다. 스케쥴링 중에는 휴리스틱(heuristic)에 의해 자원을 할당하고, 태스크가 자원을 사용하는 정책은 "Static", "Flexible", "Proportional", "Individual" 등으로 다양하다.

[21]에서는 연속 매체 화일 서버를 설계할 때 비중단 EDF(Nonpreemptive Earliest Deadline First) 스케쥴링 기법을 적용하여 이 때 필요한 디스크 버퍼 공간, 디스크 상에서의 화일의 배치, 주어진 시스템 환경하에서 서비스 가능한 최대 세션의 수를 분석하였다. 제안된 정리에 의하면 주어진 태스크 들의 주기가 모두 동일하거나, 동일한 수행 시간을 갖고 주기와 시작 시간이 수행 시간의 정수배이면 선점적 EDF 알고리즘과 스케쥴 가능 조건을 갖는다.

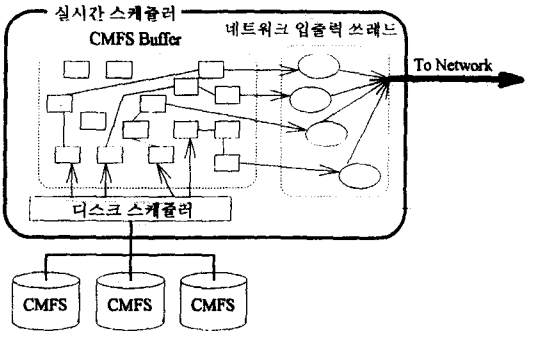
이상에서 살펴본 바에 의하면 실시간 스케쥴링 기능은 자원 예약 기법을 기반으로 하여 특정 실시간 스케쥴링 알고리즘을 적용하고 있을 뿐 아니라 경성 실시간 시스템을 기반으로 만들어진 이유로 인하여 주변 장치들(디스크, 네크워크 등)을 고려하고 있지 않기 때문에 비디오 서버의 태스크 스케쥴링에는 적합하지 않아 이에 대한 개선이 필요하다.

3. 비디오 서버 모델과 요구 조건 분석

3.1 비디오 서버 모델

(그림 1)은 비디오 서버가 내부적으로 수행하는 동작을 표현한 것이며 각 구성 요소는 다음과 같다. CMFS(Continuous Media File System)는 디지탈 비디오 데이타를 저장하기 위하여 개발한 화일 시스템이다. 화일 시스템 구조의 변경, 데이타 전달 기법의 개선, 데이타 블럭의 크기를 확대함으로써 연속 매체 응용에 적합하도록 설계되었다[20]. 디스크 스케쥴러는 사용자로부터 요청된 디지탈 비디오 데이타를 CMFS 버퍼에 연속적으로 읽어주는 프로세스이며, 네트워크 입출력 쓰레드는 버퍼에 존재하는 비디오 데이타를 네트워크로 방출하는 기능을 한다. 이 모든 프로세스들은 운영체제가 제공하는 주기적 프로세스(Periodic Process) 스케쥴러가 제공하는 실시간 스케쥴링 기능에 의해 시간 제약 조건을 만족하게 된다.

실시간 스케쥴링의 측면에서 비디오 서버는 비슷한



(그림 1) 비디오 서버 모델
(Fig. 1) Video server model

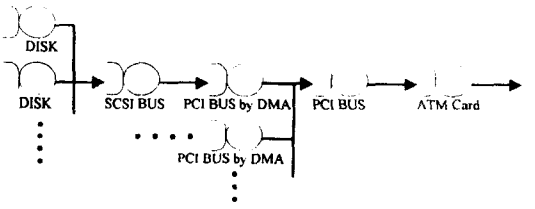
시간 제약을 가지는 쓰레드을 대량으로 가진다. 모든 처리는 하드웨어 수준에서 이루어지기 때문에 대부분의 쓰레드들은 단지 하드웨어 구동 드라이버를 수행한다. 그래서, 쓰레드의 주기는 하드웨어 특성에 의존하고 계산시간은 매우 짧으면서 비선점적이다.

쓰레드가 가진 선점성 여부와 데드라인의 중요도가 다양하기 때문에 기존의 스케쥴링 알고리즘은 비디오 서버의 스케쥴링에 수정없이 적용되지 못한다. 특히, 범용 운영체제에서는 인터럽터 처리가 가장 높은 우선 순위로 스케쥴링되기 때문에 일반적인 쓰레드의 데드라인을 보장하지 못한다. 본 논문에서는 이와 같은 스케쥴링 요구에 대하여 쓰레드의 선점성 여부와 데드라인의 중요도를 고려한 스케쥴링 기능을 비디오 서버에 구현하였다.

3.2 바디오 데이타 전달 경로

(그림 1)과 같은 프로세스 환경에서 사용자에 의한 데이타 검색 요청을 상세히 표현하면 (그림 2)와 같은 성능 모델로 표현할 수 있다. 비디오 서버를 구성하는 프로세스들은 고정된 서비스 요청률, 고정된 서비스 시간, 선입선출 방식의 스케쥴링으로 볼 수 있다. 예를 들어 요청하는 디스크 대역폭이 6 Mbps일 때 디스크 스케쥴러는 초당 N번의 디스크 요구를 발생시킬 수 있고, 만일 10개의 세션이 서비스된다면 초당 (6 Mbps/N)바이트를 읽는 요구가 10N개 발생한다. 디스크에 의해 처리된 요구는 데이타 전달 경로를 따라 네트워크로 보내어 지는데, 이 때 각 요구를 서비스하는 시간은 일정하다고 볼 수 있고 주기적 스케쥴러에 의해 동작한다. 그리고 시스템 내에 SCSI

제어기는 여러개 존재할 수 있고, 각 SCSI 제어기는 여러 개의 디스크를 제어할 수 있으므로 (그림 2)와 같은 큐잉 네트워크 모델로 표현할 수 있다.



(그림 2) 비디오 서버의 데이타 전달 경로에 대한 큐잉 네트워크 모델

(Fig. 2) Queueing network model for the data transfer path of the video server

〈표 1〉 비디오 서버에 대한 큐잉 네트워크 모델의 주요 매개변수

〈Table 1〉 Parameters of queueing network model for the video server

매개 변수	설 명	매개 변수	설 명
R	각 사용자에 의해 요구된 대역폭	Ldma	PCI BUS by DMA의 큐 길이
l	사용자의 디스크 요구율	Ddma	PCI BUS by DMA의 서비스 시간
Ddisk	디스크의 서비스 시간	Dpci	PCI BUS의 서비스 시간
Lscsi	SCSI BUS 큐 길이	Latm	ATM Card의 큐 길이
Dscsi	SCSI BUS 서비스 시간	Datm	ATM Card의 서비스 시간

(그림 2)와 같은 데이타 전달 경로상에서는 사용자의 디스크 요구 간격보다 많은 처리 시간을 요구하는 하부 시스템이 있으면 병목 현상이 발생한다. (그림 2)를 구성하는 각각의 하부 시스템 성능을 살펴보면 디스크는 수 Mbyte/sec, SCSI 버스는 10~40 Mbyte/sec, PCI 버스는 133 Mbyte/sec, ATM 카드는 19 Mbyte/sec(1.544 Mbps)의 성능을 갖는다. 디스크는 다른 하부 시스템과 비교할 때 상대적으로 성능이 낮고 비디오 스트림 서비스의 시작점이기 때문에 (그림 3)과 같은 부등식이 성립한다면 비디오 데이타의 처리는 가능하다고 볼 수 있다.

$$\frac{1}{\lambda} \geq D_{disk} \geq D_{scsi}, D_{dma}, D_{pci}, D_{atm}$$

(그림 3) 디스크 처리시간과 다른 하부 시스템 처리시간과의 관계

(Fig. 3) Relationship between disk processing time and processing time of other subsystem

3.3 디스크가 실시간 스케줄링에 미치는 영향

디스크는 성능이 낮고 디스크 헤드 이동 비용이 비싸며 비선점적인 특징이 있다. 특히 디스크 헤드 이동 비용이 높은 것은 한 번 서비스시에 많은 양의 데이타를 읽도록 요구한다. 초당 6 Mbps를 읽기 위하여 디스크 서비스를 요청할 때 한번 요구하는 비디오 스트림의 양을 증가시켜가며 소요되는 시간을 계산할 경우 아래와 같은 식으로 표현할 수 있고 디스크의 속도가 빠를수록 디스크 헤드 이동 비용에 의하여 소비되는 시간의 비중은 증가한다.

초당 디스크 서비스 시간 =

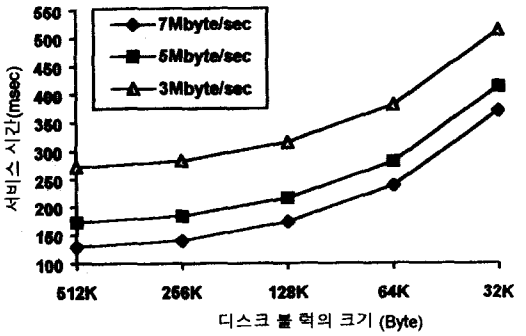
X*(탐색 시간 +지연시간) +6 Mbit/디스크 전송속도

(단, X는 초당 디스크에 접근하는 횟수를 나타낸다.)

(그림 4)는 평균 탐색시간과 지연시간을 11 msec로 고정시키고 디스크 전송속도와 블럭의 크기에 따라 6 Mbit를 서비스하기 위하여 소요되는 시간을 계산한 것이다. 이 그래프에 의하면 디스크의 블럭의 크기가 디스크 서비스 시간에 미치는 영향은 매우 큰 것을 알 수 있는데 특히 블럭의 크기가 작아질수록 디스크 성능의 차이가 작아지는 것을 볼 수 있다. 또한 이 때 소요되는 시간은 디스크가 수용할 수 있는 세션을 직접 제한하고 있다. 그러므로 고성능 디스크를 사용하는 비디오 서버는 효율적인 디스크 서비스를 위하여 매우 큰 블럭을 가져야 한다.

그러나 디스크 블럭의 크기는 디스크의 서비스 시간 뿐 아니라 다음과 같은 시스템 파라메터들을 결정한다.

첫째, 화일 시스템 버퍼 크기를 결정한다. 한 번 요구된 디스크 서비스는 쓰레드의 측면에서 볼 때 비동기적이며, 디스크 측면에서는 비선점적이기 때문에 한 번 요구한 양 만큼의 버퍼를 가지고 있어야 한다. 그러므로 본 연구와 같은 시스템 모델에서 256 Kbyte



(그림 4) 디스크 서비스 시간
(Fig. 4) Disk service time

의 디스크 블럭을 사용할 경우 최소한 읽기 위한 버퍼, I/O 쓰레드가 사용하기 위한 버퍼 등 2개가 필요하고 이들은 세션의 갯수를 곱한 수 만큼 필요하게 된다.

둘째, 디스크 스케쥴러의 수행 주기를 결정한다. 256 Kbyte의 버퍼를 사용할 경우 6 Mbps를 읽기 위해서는 초당 세번의 디스크 요구를 발생시키면 가능하다. 그러므로 디스크 스케쥴러는 사용자가 요구하는 대역폭을 블럭의 크기로 나눈 만큼의 주기를 가진다.

셋째 네트워크 입출력 쓰레드의 주기 하한선을 결정한다. 네트워크 입출력 쓰레드는 화일 시스템 버퍼에 있는 비디오 스트림을 계속적으로 내보낸다. 이 때 사용된 버퍼는 비디오 스트림을 읽기 위하여 재사용되고 새로운 버퍼를 가져온다. 비디오 스트림을 트랜스포트 계층에 내보낼 때는 화일 시스템 버퍼 대 데이타 전송용 버퍼로 데이타에 대한 복사가 발생하고 이것은 ATM 카드에 의해 사용되어진 후 시스템 메모리로 되돌려지게 된다. 그러므로 네트워크 입출력 쓰레드는 빠른 주기로 작은 양의 데이타를 처리하는 것이 필요한데, 이 수행 주기는 적어도 디스크 스케쥴러와 같은 주기이거나 빠른 주기이어야 한다.

이와 같이 살펴볼 때 디스크 블럭은 클수록 전체 시스템의 성능이 향상된다고 볼 수 있다. 그러나 그 크기는 사용자가 요구하는 대역폭, 데이타 전달 경로상에 존재하는 버퍼의 크기에 따라 결정된다. 특히 디스크 캐쉬는 디스크의 논리적 블럭의 크기에 제한하는 파라메터이다. 그러므로 비디오 스트림을 서비스하는 프로세스의 실시간 스케쥴링 파라메터는 다음과 같이 결정하는 것이 바람직하다.

스케쥴링 기준 1: 디스크 관련 쓰레드는 디스크를 가장 효율적으로 이용할 수 있도록 주기가 결정되어야 한다.

스케쥴링 기준 2: 네트워크 입출력 쓰레드는 문맥교환 비용이 성능에 미치는 영향이 적으므로 버퍼의 크기를 최소화하는 방향으로 스케쥴링 파라메터가 결정되어야 한다.

스케쥴링 기준 3: 모든 쓰레드는 비선점적인 하드웨어 구동 드라이버를 수행하므로 비선점적 스케쥴링 이론을 적용하여야 한다.

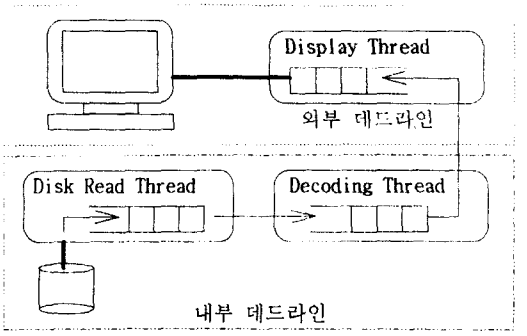
4. 실시간 스케쥴러

본 논문에서는 다양한 환경에서 최적의 성능을 가진 것으로 증명된 EDF 알고리즘을 기반으로 하여 쓰레드의 성질에 따라 데드라인에 대한 우선 순위를 부여하는 기법을 사용하였다. 데드라인에 대한 우선 순위 부여는 EDF 알고리즘의 단점인 비선점적 환경에 대한 적용의 어려움과 과부하시 데드라인 미스의 도미노 현상을 방지하는 역할을 한다.

4.1 쓰레드의 중요도

본 논문에서는 (그림 5)와 같은 프로그램밍 모델을 가정하였다. 연속 매체를 처리하는 하나의 태스크는 여러 개의 쓰레드들로 구성되어 일련의 연속 매체 처리 단계를 구성할 수 있다. 이러한 쓰레드의 수행에서 태스크가 시간 제약 조건을 지키지 못하는 경우는 소리나 영상 처리의 마지막 처리에서의 데드라인 미스나, 단계에서 잡음을 느끼는 것이거나 데이타 전달 경로상에서 디바이스의 특성상 데드라인 미스의 비용이 큰 경우이다. 그러므로 데드라인 미스에는 (그림 5)와 같이 사용자에게 영향을 미칠 수 있는 것과 영향을 미칠수 없는 것으로 구분하여 이를 각각 외부 데드라인(External Deadline), 내부 데드라인(Internal Deadline)이라 하였다.

외부 데드라인을 가진 쓰레드는 내부 데드라인을 가진 쓰레드보다 높은 우선 순위를 가진다. 만일 같은 시각에 수행이 시작되어야 하는 서로 다른 데드라인 종류를 가진 두 개의 쓰레드가 있을 때 외부 데드



(그림 5) 다수의 쓰레드로 구성된 태스크
(Fig. 5) Task that consist of multiple threads

라인을 가진 쓰레드가 내부 데드라인을 가진 쓰레드의 긴급성에 관계없이 우선적으로 수행된다. 그리고 외부 데드라인을 가진 쓰레드가 수행되고 있을 때 데드라인이 보다 긴급한 데드라인을 가진 쓰레드가 발생하거나, 외부 데드라인을 가진 쓰레드가 자신의 퀀텀을 초과하여 수행하고 있을 때에도 이와는 무관하게 외부 데드라인을 가진 쓰레드가 처리기 시간을 점유한다.

데드라인의 종류를 두 가지로 나누는 것은 실시간 쓰레드의 종류를 수행의 중요도에 따라 나누는 것이다. 이러한 기법은 모든 쓰레드의 데드라인을 만족시키지 못하는 환경에서는 그 중요도에 따라 선택적으로 수행함으로써 모든 쓰레드의 데드라인을 만족시키기 위한 스케쥴링보다 스케쥴링 문제를 보다 단순화하고 주변 장치와의 동기화를 고려한 것이다. 본 논문에서의 비디오 서버 모델에서는 네트워크 입출력 쓰레드는 외부 데드라인을 갖고 디스크 스케쥴러는 내부 데드라인을 갖는다.

4.2 실시간 프리미티브

멀티 미디어 응용 프로그램은 사용자가 수행에 필요한 자원의 양을 기술해 줄 것을 요구한다. RT Mach에서는 "runres"와 같은 인터페이스를 통하여 프로그램의 수행주기, 소요 시간 등을 기술한다[12]. 그러나 정적인 처리기 시간 할당(static processor allocation)을 위해서는 소요 시간, 수행 시간을 정확하게 알아야 한다. 범용 시스템을 사용하는 연성 실시간 시스템이 프로세스가 수행이전에 정확한 수행 시간을 찾아내는 것은 불가능하다. 또 수행시간이 모자라는 경우에 주변 장치와 직접 관련된 쓰레드는 계산 중심의 쓰레드와는 달리 비선점(non-preemptible)적이기 때문에 부족한 처리기 시간을 필요한 즉시 제공하여야 한다. 실시간 프리미티브가 가져야 하는 기능을 나열하면 다음과 같다.

첫째, 비실시간 쓰레드를 실시간 쓰레드로 바꾸거나 실시간 쓰레드를 비실시간 쓰레드로 전환할 수 있어야 한다.

둘째, 현재 자원의 용량을 알 수 있어야 한다.

셋째, 자원 사용량을 자원 조절할 수 있어야 한다. 즉, 부족한 자원을 커널에 더 요구하거나 남은 자원을 커널에 반환할 수 있어야 한다.

넷째, 일정량의 자원을 예약할 수 있어야 한다.

다섯째, 비선점적으로 자원을 사용할 수 있어야 한다.

〈표 2〉는 이러한 요구를 만족시키기 위하여 설계한 실시간 스케쥴링 프리미티브들이다.

〈표 2〉 비디오 서버를 위한 실시간 프리미티브
〈Table 2〉 Real-time primitives for the video server

프리미티브	기 능
clock_t ppclock()	실시간 스케쥴러의 클럭을 읽는다.
void qgiveup()	쓰레드에 할당된 처리기 시간을 포기한다.
void preemptive()	쓰레드가 선점적임을 나타낸다.
void nonpreemptive()	쓰레드가 할당받은 퀀텀에 관계없이 비선점적임을 나타낸다.
void qstatus()	현재 쓰레드의 처리기 사용량을 읽어온다.
boolean_t qask()	쓰레드의 처리기 시간이 부족함을 커널에 알린다.
void qrelease()	쓰레드가 잉여 처리기 시간을 가짐을 커널에 알린다.
boolean_t priocntl()	쓰레드의 스케쥴링 파라메터를 바꾸거나 읽는다.

새로이 설계된 실시간 프리미티브는 쓰레드의 수행시에 발생하는 자원 부족/과잉 상태를 시스템에 알리거나 쓰레드의 필요에 따라 스케쥴링 파라메터를 수정할 수 있는 인테페이스를 제공하여 최소한의 정보로 자원을 관리할 수 있게 한다. 또한 처리기 시간 부족으로 발생할 수 있는 선점을 방지하기 위하여 쓰레드가 어떤 시점까지 할당받은 처리기 시간에 관계

```
1.    void stream_scheduler(void *arg)
2.    {
3.
4.        디스크 스케줄러의 주기와 계산 시간을 지정한다.
5.
6.        change2PPclass();
7.        while (1) {
8.                if (디스크에 대한 요구가 없다면) {
9.                        qgiveup();
10.                       continue;
11.               }

12.               nonpreemptive();
13.               for ( 각 세션에 대하여 ) {
14.                       /* 각 세션에 대한 디스크 요구를 발생시킨다. */
15.               }
16.               if (계산 시간이 부족) {
17.                       while(qask(부족한 계산시간의 양) != FAIL)
18.                               부족한 계산 시간의 양 /= 2;
19.           }
20.               else
21.                       qrelease(남은 계산 시간의 양);
22.               preemptive();
23.       } /*  while (1)  */
24.   } /*  void stream_scheduler  */
```

(그림 6) 실시간 프리미티브로 구현된 디스크 스케줄러
(Fig. 6) Disk scheduler that implemented by real-time primitives

없이 사용할 수 있도록 비선점을 선언할 수 있도록 하여 주변 장치와의 연관성도 고려하였다. (그림 6)은 비디오 서버의 디스크 스케줄러를 실시간 프리미티브를 사용한 경우를 나타낸 것이다. [7]의 연구에서도 프로그램 시작시에 예약한 자원을 기반으로 주기적으로 쓰레드 스스로 자원 사용량을 조절할 수 있도록 하였다. 그러나 QoS 개념을 도입하면서도 RT Mach와 본 연구의 아래와 같은 차이가 있다.

첫째, RT Mach는 모든 쓰레드의 데드라인을 만족시키고자 하지만 본 연구에서는 외부 데드라인만을 엄격히 만족시킨다.

둘째, RT Mach는 주변장치와 프로세스의 관계를 고려하지 않지만 본 연구에서는 주변장치와의 관계를 고려하였다.

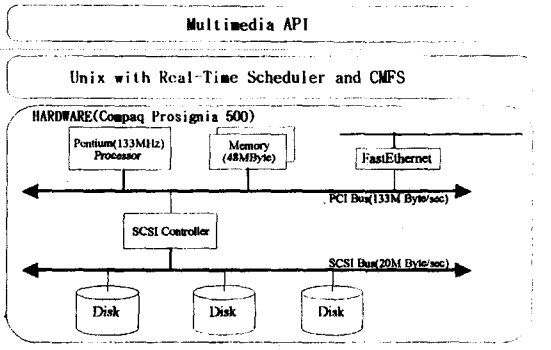
5. 구현 환경

본 연구에서 사용한 기반 하드웨어는 (그림 7)과 같은 범용 하드웨어로써 133MHZ의 펜티엄 처리기, 48Mbyte 메모리, 4Gbyte의 디스크, 그리고 UnixWare 2.0에 실시간 스케줄링 기능과 연속 매체 화일 시스템(CMFS) 기능을 추가한 운영체제를 사용하고 있다. 네트워크 디바이스는 UnixWare용 ATM 드라이브가 없어 패스트이쓰넷(FastEthernet)을 사용하였다.

UnixWare 2.0은 SVR4ES/MP 계열로써 모놀리틱 커널(Monolotic kernel)임에도 불구하고 임계 영역과 다음의 경우를 제외하고는 어디에서나 선점가능하다[16].

1. 처리기가 커널 스핀록을 획득한 경우
2. 인터럽터를 서비스하고 있을 경우
3. 처리기가 한가한(idle) 경우
4. 커널이 명백하게 비선점(non-preemption)을 요구할 때

커널의 선점 가능성(kernel preemptability)은 처리기마다의 상태 변수에 위해 유지되며 커널 선점 요구는 인터럽터로부터 돌아올 때, 스핀 록을 반환할 때, 커널이 명백하게 선점가능한 상태를 만들었을 때(ENABLE_PRMPT() 마크로를 호출)이다.



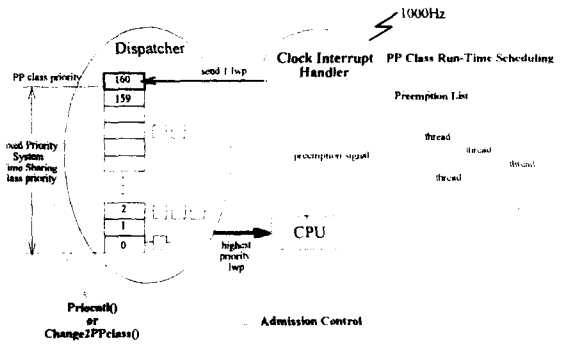
(그림 7) 비디오 서버의 구현 하드웨어
(Fig. 7) Implementation hardware of the video server

본 연구에서는 실시간 스케줄러를 구현하기 위하여 시간 인터럽터 해상도를 100Hz에서 1000Hz로 높였다. 높은 시간 인터럽터 해상도는 처리기 시간의 할당 단위가 작아짐을 의미하는데 이는 매우 짧은 계산 시간을 요구하는 프로세스가 많은 환경에서는 필수적이다.

주기적인 스케줄링을 필요로 하는 쓰레드는 사용자 수준에서는 priocntl 시스템 콜, 커널 수준에서는 Change2PPclass 프리미티브를 사용하여 비실시간 클래스(Time sharing, System, Fixed Priority)로 부터 주기적 스케줄러에 등록된다. priocntl 또는 Change2PPclass 프리미티브에는 쓰레드의 수행 주기와 각 수행 주기

마다 소비하는 처리기 시간을 기술하여 쓰레드가 전체 처리기 시간 중 차지하는 비율을 정하여 처리기 시간을 예약한다. 예를 들어 주기 100 msec, 퀀텀 5 msec인 쓰레드는 매 100 msec마다 5 msec의 처리기 시간을 소비하여 전체 처리기 시간 중 5%를 점유한다.

주기적 스케줄러의 동작 방식과 데이타 구조는 (그림 8)과 같다. UnixWare의 스케줄러는 우선 순위 방식이기 때문에 디스패처로 하여금 수행하고자 하는 쓰레드를 선택하게 하려면 가장 높은 우선 순위 큐에 존재할 때 처리기에게 선점 시그널 보냄으로써 해당 쓰레드는 처리기 시간을 점유하게 된다. 주기적 스케줄러는 클럭 인터럽터를 통하여 이와 같은 기능을 수행하며 매 쓰레드가 한 주기의 수행을 마치게 되면 선점 리스트(Preemption List)에 존재하는 쓰레드들의 다음 주기의 수행 시간을 결정한다.



(그림 8) 실시간 스케줄러
(Fig. 8) Real-time scheduler

6. 성능 분석

6.1 실시간성에 대한 만족도 검사

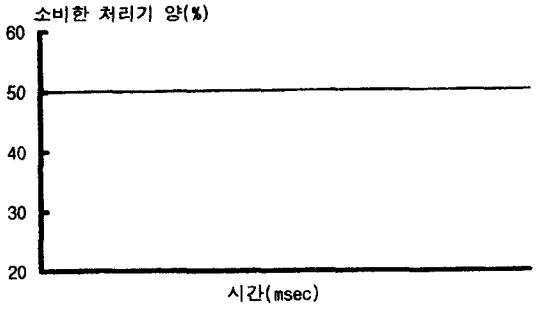
실시간 시스템의 성능을 나타내는 척도는 범용 시스템의 성능을 나타내기 위하여 사용되는 일반적인 척도(단위 시간당 처리 속도)와 함께 쓰레드의 실시간성을 어느 정도 만족시켜 줄 수 있는가 하는 것이다. 우선 본 연구에서는 시간 인터럽터 해상도을 100 Hz에서 1000 Hz로 높였는데 이로 인한 시스템 성능 저하는 3%정도이었다.

쓰레드의 실시간성에 대한 만족도는 문맥 교환이 이미 정해진 선점 시간에 일어나는지, 예약된 처리기 시간이 모두 해당 쓰레드를 위하여 소비되는지에 따라 결정할 수 있다. 즉 본 논문의 연구 대상인 비디오 스트림은 본질적으로 연성 실시간 시스템에 속하기 때문에 msec 단위의 실시간성을 만족한다면 비디오 스트림 서비스의 시간 제약 조건을 만족한다고 할 수 있다. 실시간성의 만족도를 측정하기 위하여 문맥교환 속도, 예약된 처리기 시간의 사용에 대한 데이타를 수집하였다.

문맥교환 속도를 측정하기 위하여 선점 시그널(Preemption Signal)이 발생한 후 실제 문맥 교환이 수행되기 까지의 시간을 측정하였다. 본 논문에서 구현한 시스템의 경우 1 msec의 클럭 주기를 가지고 있는데, 측정치는 모두 0으로 기록되어 선점 시스널의 발생 후 문맥 교환까지는 1 msec이내에 처리되는 것으로 나타났다. 이것은 구현 환경에서 밝힌 바와 같이 UnixWare가 임계 영역과 처리기가 커널 스핀록을 획득한 경우, 인터럽터를 서비스하고 있을 경우, 처리기가 한가한(idle) 경우, 커널이 명백하게 비선점(non-preemption)을 요구할 때를 제외하고는 어디에서나 선점가능하므로 커널 내부 지연이 발생하지 않기 때문이다. 예약된 처리기 시간에 대한 데이타는 계산 중심의 응용 프로그램에 50 msec의 주기마다 25 msec 처리기 시간을 할당한 후(전체 처리기 양의 50%), 수행시 나타나는 커널 모드 시간의 양을 측정하였다. (그림 9)는 시스템이 한가한(idle) 경우 계산 중심의 응용 프로그램이 소비한 처리기 시간의 양을 그래프로 나타낸 것으로 X축은 시간의 경과를 나타내고 Y축은 계산 중심 응용 프로그램이 소비한 처리기의 양을 백분률로 나타내고 있다. 그래프내의 데이타는 8950 msec 동안 측정되었다. 그래프에 의하면 측정 데이타는 전체 처리기 시간의 50%를 정확하게 소비하고 있는데 이것은 문맥 교환 시간이 1 msec이내에 이루어 지고 커널 내 지연이 발생하지 않는 다는 것을 간접적으로 보여주고 있다. 또한 연성 실시간 시스템에서는 주기적으로 발생되는 인터럽터가 실시간 쓰레드의 수행에 영향을 미치지 않는 것으로 볼 수 있다.

본 연구에서는 이와 더불어 타 쓰레드들과의 간섭 현상을 측정하기 위하여 화일 시스템에 대한 읽기와 네트워크에 대한 통신을 수행하는 쓰레드를 백그라운드로 수행시켰다. 다만 본 장에서는 처리기 시간을

측정하는 단위가 1 msec이기 때문에 어느 정도의 오차를 감수하여야만 한다.



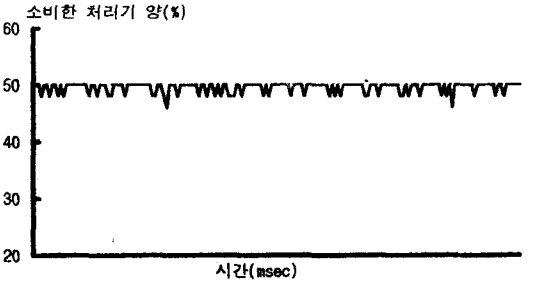
(그림 9) 계산 중심 응용 프로그램을 단독으로 수행하였을 때 소비한 처리기 시간

(Fig. 9) Consumed processor time when compute-bound application is executed alone

(1) 디스크 읽기 쓰레드를 백그라운드로 수행한 경우

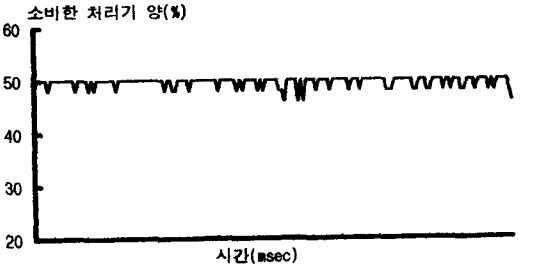
(그림 10)은 백그라운드 프로세스로 vxfs 화일 시스템에 대한 읽기/쓰기를 반복하는 3개의 프로세스를 수행시킨 후 계산 중심 응용 프로그램이 소비한 처리기 양을 측정한 것이다. 그래프에 의하면 간헐적으로 2~4%정도의 처리기 시간을 시스템 내의 다른 동작을 위하여 소비하고 있다. 이것은 디스크에 대한 읽기를 수행하는 쓰레드가 디스크에 대한 요구를 발생시킨 후 읽혀진 데이타는 DMA를 통하여 전송되는데, 이 때 커널 내의 SCSI 드라이브내의 인터럽터 서비스 루틴 수행과 커널 수준의 버퍼에서 사용자 수준 버퍼로 복사될 때 발생하는 것이다. 측정치에 의하면 응용 프로그램이 소비하여야 할 4475msec중 88msec를 인터럽터 처리를 위하여 소비하고 있는데 이는 전체 처리기 시간 중 1.9%정도에 해당되는 적은 양으로써 타 프로세스에 의한 디스크 입출력은 실시간 스케쥴링에 많은 영향을 미치지 않는 것을 알 수 있다.

(그림 11)은 위와 같은 실험 환경에서 5개의 백그라운드 프로세스를 수행시킨 후 측정한 것이다. (그림 14)에 의하면 전체 처리기 시간 중 90 msec정도를 인터럽터와 타 프로세스의 커널 수준 동작에 의해 소비하고 있다. 3개의 백그라운드 프로세스와 비교할 때 많은 차이를 보이지 않고 있다. 그러므로 실시간 스케쥴링 기능은 일반적인 인터럽터 처리 루틴과 타 프로세스에 의해 크게 영향을 받지 않고 수행될 수 있음을 알 수 있다.



(그림 10) 3개 디스크 읽기 프로세스를 수행한 경우 계산 중심 응용 프로그램이 소비한 처리기 시간

(Fig. 10) Processor time that the compute-bounded application consumes when three disk read processes are executed



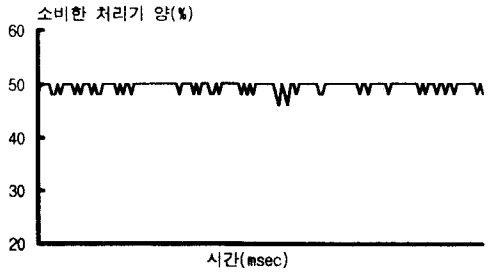
(그림 11) 5개 디스크 읽기 프로세스를 수행한 경우 계산 중심 응용 프로그램이 소비한 처리기 시간

(Fig. 11) Processor time that the compute-bounded application consume when five disk read processes are executed

(2) 연속매체 화일 시스템에 대한 디스크 읽는 쓰레드를 백그라운드로 수행한 경우

(그림 12)는 연속 매체 화일 시스템에 대한 디스크 읽기 쓰레드를 수행하여 디스크 블럭의 크기와 데이타 전달 경로의 개선이 얼마나 영향을 미치는 가를 측정하였다. 측정치에 의하면 전체 수행 시간 중 92msec를 인터럽터 서비스 루틴과 타 프로세스의 수행에 사용하였다. 이러한 처리기 시간은 vxfs 화일 시스템을 사용한 것과 유사한 것으로써 화일 시스템의 구조와

실시간 스케줄링 기능과는 분리되어 동작되고 있음을 보여준다.



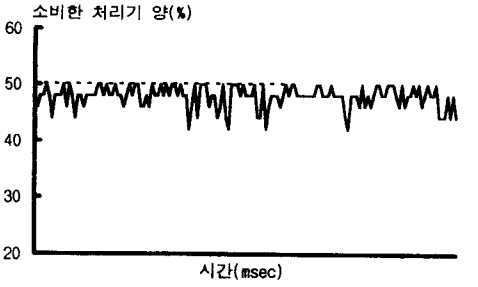
(그림 12) 5개 디스크 읽기 프로세스를 수행한 경우 계산 중심 응용 프로그램이 소비한 처리기 시간

(Fig. 12) Processor time that the compute-bounded application consumes when five disk read processes are executed

(3) 네트워킹 기능을 사용하는 쓰레드를 백그라운드로 수행한 경우

(그림 13)은 네크워크를 통하여 원격 노드와의 통신을 수행하는 프로세스를 백그라운드로 수행하였을 때 계산 중심 응용 프로그램이 소비하는 처리기 시간을 측정하였다. 백그라운드 프로세스는 6 Mbps MPEG-2 비디오 스트림 서비스를 제공하는 1개의 세션을 서비스하였다.

현재 UnixWare는 TCP/IP 프로토콜 처리를 위하여 스트림과 인트럽트를 사용하고 있다. 즉 상위 수준에서 스트림 큐에 데이타를 넣어 두면 이것은 인터럽터를 통하여 프로토콜 처리를 마친 후 네트워크로 내보내어지게 된다. 이러한 구조는 프로토콜 처리가 시스템 내의 어떠한 쓰레드보다 우선 순위가 높아지게 되어 실시간 쓰레드를 가장 높은 우선 순위로 수행할 경우에도 이로 인한 간섭 현상은 피할 수 없게 된다. (그림 13)에서는 응용 프로그램의 소비하여야 할 처리기 시간의 10%(전체 처리기 시간의 5%)를 프로토콜 처리를 위하여 소비하고 있다. 그리고 프로토콜 처리가 차지하는 처리기 시간의 비중이 2~10%로 비교적 높아 퀀텀이 매우 짧은 실시간 쓰레드의 경우 퀀텀을 할당 받는다 하더라도 프로토콜 처리 때문에 처리기 시간을 사용하지 못할 수도 있다. 또한 여러 개의 세션을 서비스할 경우는 이보다 많은 처리기 시간을 프로토콜 처리를 위하여 소비하게 되므로 상위 수준의 프로토콜을 하드웨어로 구현하거나 패스트이쓰넷(FastEthernet) 드라이브에 직접 데이타를 전달하는 기법이 사용되어야 할 것이다.



(그림 13) 6 Mbps의 전송률을 가지는 프로세스를 계산 중심 응용 프로그램이 소비한 처리기 시간

(Fig. 13) Processor time that the compute-bounded application consumes when one process transfers data at 6 Mbps

6.2 실시간 스케줄링의 효용성

본 논문에서는 비디오 서버에 비선점적 쓰레드 스케줄링을 적용하지 않는 경우와 적용한 경우 비디오 서버의 데이타 전송 간격을 측정하여 실시간적 요구조건이 만족되는지 여부를 측정하였다.

〈표 3〉 쓰레드 집합의 시간 제약 조건

〈Table 3〉 Time constraints of the thread set

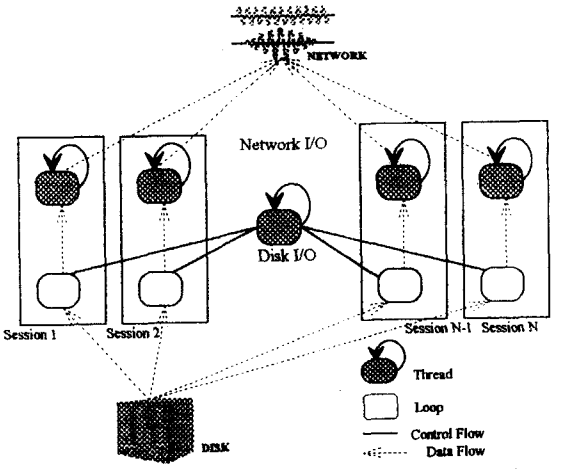
쓰레드	시작 시간	주기	계산 시간
디스크 입출력 쓰레드	스케줄링 알고리즘이 결정	325 msec	수행 중 조절
네트워크 입출력 쓰레드	스케줄링 알고리즘이 결정	25 msec	수행 중 조절

전체 쓰레드 구조는 (그림 14)와 같이 하나의 세션에 대하여 하나의 네트워크 I/O 쓰레드를 할당하고, 디스크 입출력을 위한 쓰레드는 6 Mbps 세션들을 위하여 1개 할당하여 수행하였다. 그리고, 쓰레드의 시간 제약 조건은 〈표 3〉과 같이 지정하였다.

쓰레드의 계산 시간은 쓰레드가 수행 중 필요한 양을 과잉/부족에 따라 스스로 조절할 수 있는 프리미

티브를 제공하였다. 그리고, 디스크 블럭의 크기는 256KByte로 하였기 때문에 디스크 입출력 쓰레드는 325 msec의 주기를 갖고, 네트워크 입출력 쓰레드는 25 msec의 주기를 지정하였다. 그리고, 네트워크 입출력 쓰레드의 데이타는 가상적으로 전송하였다.

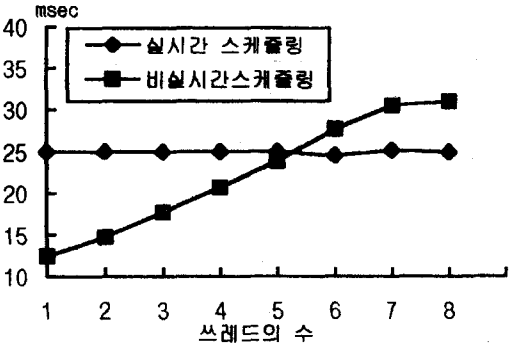
비실시간 스케줄러는 쓰레드가 시간 간격을 두고 수행되기 위하여 한 번 수행 후 12 msec 동안 sleep 상태에 있었다.



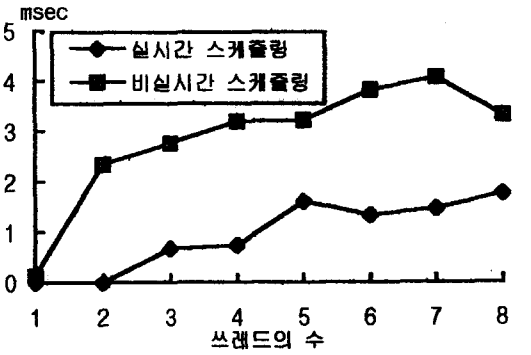
(그림 14) 비디오 서버의 쓰레드 구조
(Fig. 14) Thread structure of the video server

(그림 15)는 실시간 스케줄링을 적용한 경우와 비실시간 스케줄링을 적용한 경우의 평균 데이타 전송 간격을 쓰레드의 수를 증가시켜가며 수집한 데이타이다. 실시간 스케줄링을 수행한 경우는 평균 데이타 전송 간격이 주기와 거의 일치함을 알 수 있다. 그러나 비실시간 스케줄링에서는 쓰레드 수의 증가에 따라 평균 전송 간격이 단조 증가함을 볼 수 있다. 비실시간 스케줄링이 이러한 성능을 보이는 것은 쓰레드는 매 12 msec동안 sleep 상태에 있지만, sleep 상태가 끝난 후 다시 스케줄러에 스케줄링을 받기 위하여 큐에서 대기하기 때문이다.

(그림 16)은 실시간 스케줄링을 적용한 경우와 비실시간 스케줄링을 적용한 경우의 데이타 전송 간격의 표준 편차이다. 이 데이타는 부하가 일정한 경우에도 비실시간 스케줄링에서는 비디오 데이타의 QoS를 만족시키기 어렵기 때문에 실시간 스케줄링이 비실시간 스케줄링에 비하여 안정된 서비스를 제공할 수 있음을 알 수 있다.



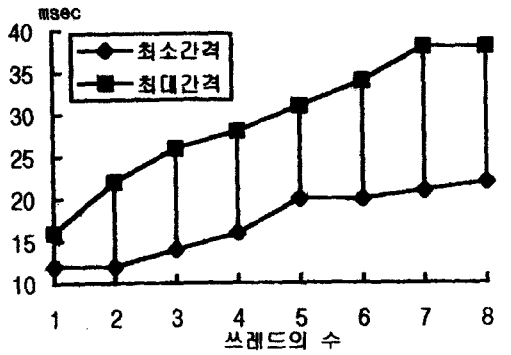
(그림 15) 평균 데이타 전송 시간 간격
(Fig. 15) Average data transfer time interval



(그림 16) 데이타 전송 간격의 표준 편차
(Fig. 16) Standard deviation of data transfer interval

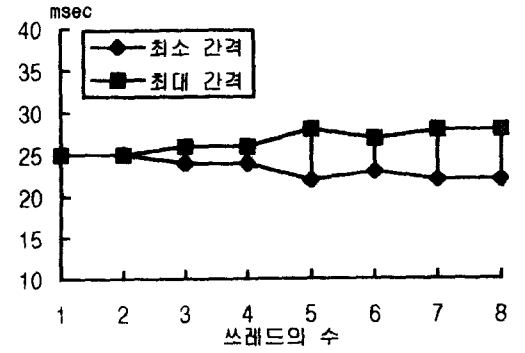
(그림 17)은 비실시간 스케줄링을 적용한 경우 데이타 전송 간격의 최대 최소 값의 차이를 보인다. 데이타에 의하면 부하가 증가할 수록 최대 최소 값의 편차는 증가하고 있기 때문에 비실시간 스케줄링을 사용한 시스템은 시간 제약 조건을 만족시키기 위해서는 매우 비효율적인 시스템이 될 수 있음을 보여준다.

(그림 18)은 실시간 스케줄링을 적용한 경우 데이타 전송 간격의 최대 최소 값의 차이를 보인다. 그래프에 의하면 쓰레드 수의 증가에 따라 편차가 증가하다가 일정 간격을 유지하여 부하가 많은 경우에도 효율적으로 시스템을 운영하면서 안정된 서비스를 제공할 수 있음을 알 수 있다.



(그림 17) 비실시간 스케줄링을 적용한 경우 데이타 전송 최소 및 최대 시간 간격

(Fig. 17) Minimum and maximum time interval of data transfer when non real-time scheduling is adapted



(그림 18) 비선점적 실시간 스케줄링을 적용한 경우 데이타 전송 최소 및 최대 시간 간격

(Fig. 18) Minimum and maximum time interval of data transfer when nonpreemptive real-time scheduling is adapted

측정에 의하면 수행 중 쓰레드들이 소비한 처리기 시간은 2~4 msec정도이었다. 그래서 본 논문에서 제안한 비선점적 EDF 스케줄링의 스케줄 가능성 검사 기준에 모두 부합되었다. 그리고 실험 환경이 완전한 실시간 스케줄링 환경을 제공하지 못하기 때문에 비선점적 EDF 시뮬레이션 프로그램으로 부터 데드라인 미스 발생을 검사하여 모든 데드라인이 만족됨을 확인하였다.

7. 결 론

본 연구에서는 비디오 서버를 구현하기 위한 실시간 스케줄링 요소를 밝히고 그 기능을 구현하였다. 실시간 스케줄러는 새로운 데드라인 개념을 도입한 EDF 알고리즘과 프리미티브를 사용하였다.

시스템 모델 분석에 의하면 디스크는 비디오 서버를 구성하는 다른 하부 시스템과 달리 디스크는 기계적인 메카니즘을 작고 모든 서비스의 시작 지점이었다. 그래서 디스크 블럭의 크기는 관련 쓰레드 뿐만 아니라 다른 쓰레드의 실시간 스케줄링에도 영향을 미치는 것으로 나타났다. 이러한 결과에 의하면 실시간 스케줄링에 있어서 디스크 관련 쓰레드는 디스크의 효율적 이용에, 네트워크 입출력 쓰레드는 시스템 내 버퍼의 최소화에 기준을 두어 비선점적으로 스케줄링 되어야 함을 밝혔다. 또한 분석된 실시간 스케줄링 기준에 따라 내부/외부 데드라인 개념을 도입하고 실시간 프리미티브를 설계하였다.

이러한 기법들은 범용 운영체제를 사용한 연속 매체 응용 프로그램의 사용이 증가하는 시점에서 실시간 스케줄링을 어떻게 수행할 것인지에 대한 하나의 기준을 제시한다. 그리고 외부 디바이스가 실시간 스케줄링에 직접 영향을 미치는 것을 밝힘으로써 기존의 선점적 쓰레드를 사용한 스케줄링보다 실제 환경에서 보다 적용되기 쉽다.

성능 측정에 의하면 연성 실시간 응용인 비디오 서버는 범용 운영체제 하에서도 실시간 스케줄링을 제공한다면 시간 제약 조건을 만족할 수 있음을 알 수 있었다. 그리고 비실시간 스케줄링에 비하여 실시간 스케줄링이 효율적인 시스템을 만들 수 있음을 보였다.

참 고 문 헌

[1] Clifford W. Mercer, Stefan Savage, Hideyuki Tokuda, "Processor Capacity Reserves for Multimedia Operating Systems", CMU-CS-93-157, May 1993.

[2] Carl A. Waldspurger, William E. Weihl, "Lottery Scheduling: Flexible Proportional-Share Resource Management", USENIX OSD I, pp1~11, Nov. 1994.

[3] Andrew Campbell etc., "Integrated Quality of Service for Multimedia Communications" IEEE

NOSSDAV'95, Apr. 1995.

[4] Kang G. Shin, Yi-Chieh Chang, "A Reservation-Based Algorithm for Scheduling Both Periodic and Aperiodic Real-Time Tasks", IEEE Transactions on Computers, pp1405~1419, Dec. 1995.

[5] Klara Nahrstedt, Ralf Steinmetz, "Resource Management in Networked Multimedia Systems", IEEE Computer, pp52~64, May 1995.

[6] G. Coulson, G.S. Blair, P. Robin, D. Shepherd, "Supporting Continuous Media Application In a Micro Kernel Environment", Lancaster University, MPG-94-16, 1994.

[7] Kiyokuni Kawachiya etc., "Evaluation of QOS-Control Servers on Real-Time Mach", IEEE NOSSDAV'95, Apr. 1995.

[8] Andrew Campbell, Geoff Coulson, David Hunchison, "A Quality of Service Architecture", Lancaster University, MPG-94-08, 1994.

[9] Ramesh Govindan, "Operating Systems Mechanisms For Continuous Media", University of California, Berkeley, 1992.

[10] Eoin Andrew Hyden, "Operating System Support for Quality of Service", University of Cambridge, PhD thesis, Feb. 1994.

[11] Tatjana M. Burkow, "Operating System Support for Distributed Multimedia Applications;A Survey of Current Research", University of Cambridge, Pegasus Paper 94-8, June 1994.

[12] _____, "RT-Mach Notes:Version 1.0", CMU, Aug. 1995.

[13] http://www.octacon.co.uk/proj/diamond/index.htm

[14] Craig Federighi, Lawrence A. Rowe, "Distributed Hierarchical Storage Manager for a Video-on-Demand System", UCB//CSD-94-795, February 1994.

[15] _____, "Clock & Callout Handling Synchronization and CPU Scheduling of UNIX SVR4 ES/MP", USL.

[16] Hiroyuki Kaneko, John A. Stankvic, "A Multimedia Server on the Spring Real-Time System", University of Massachusetts, UM-CS-96-011, Jan. 1996.

[17] Andre M van Tilborg, Gary M. Koob, "Foundations of Real-Time Computing-Scheduling and Resource Management", KLUWER ACADEMIC PUBLISHERS.

[18] Clifford W. Mercer, Ragunathan Rajkumar, and Hideyuki Tokuda, "Applying Hard Real-Time Technology to Multimedia", Proceedings of the workshop on the role of real-time in Multimedia/Interactive Computing Systems, Nov. 1993.

[19] 손진곤, 곽덕훈, 이원규, "원격 학습을 위한 방송대 VOD 시스템", 정보과학회지 제 3권 제 12호, 1995년 12월.

[20] 고정국, 남경규, 김길용, "멀티미디어 데이타 스트림을 위한 효율적 화일시스템의 설계 및 구현", 1996년 정보과학회 봄 학술발표 논문집, pp407~410, 1996년 4월.

[21] 백윤철, 고건, "비중단 실시간 스케쥴링을 이용한 연속적인 미디어 화일 서버 설계시의 고려 사항", 정보 과학회 논문지 제 22권 제 2호, pp241~248, 1995년 2월.

[22] Divyesh Jadav, Alok Choudhary, "Designing and Implementing High-Performance Media-on-Demand Servers", Syracuse University, IEEE, 1995.

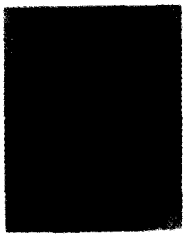
[23] Henning Schulzrinne, "Operating System Issues for Continuous Media", Multimedia System, pp269~280, Apr. 1996.

[24] Ralf Steinmetz, Klara Nahrstedt, "Multimedia: Computing, Communications And Applications", Prentice-Hall, 1995.

[25] B. Ozden, "A Low-Cost Storage Server for Movie on Demand Database", Proceedings of the 20th International Conference on Very Large Data Bases, pp594~605, Sep. 1994.

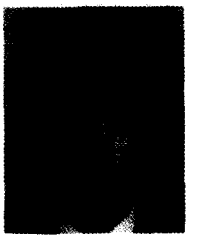
[26] J. Hsie, "Performance of a Mass Storage System for Video-on-Demand", submitted to Journal of Parallel and Distributed Computing.

[27] D. P. Anderson, G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism", IEEE Computer, Vol. 20, No. 10, pp55~57, Oct. 1991.



손 종 문

1991년 부산대학교 컴퓨터공학과 졸업(학사)
1993년 부산대학교 컴퓨터공학과 졸업(공학석사)
1997년 8월 부산대학교 컴퓨터공학과 졸업(공학박사)
1997년 9월~현재 시스템공학연구소 선임연구원
관심분야:운영체제, 실시간 시스템, 분산 멀티미디어 시스템



김 길 용

1981년 서울대학교 수학교육과 졸업(학사)
1983년 서울대학교 대학원 컴퓨터공학과(공학석사)
1988년 서울대학교 대학원 컴퓨터공학과(공학박사)
1983년~1986년 금성반도체(주) 연구원
1994년~1995년 Univ. of Southern California(USC) 객원교수
1988년~현재 부산대학교 컴퓨터공학과 부교수
관심분야:분산 시스템, 멀티미디어 시스템, 실시간 시스템