디스크 배열을 이용한 실시간 멀티미디어 저장 서버에서의 스케쥴링 기법

(A Scheduling Method for Real-Time Multimedia Storage Server using Disk Arrays)

조진성, 신현식

서울대학교 컴퓨터공학과

요 약

본 논문에서는 멀티미디어 재생 응용을 위한 실시간 저장 서버를 설계하고 성능 향상을 위한 효율적인 스케쥴링 기법을 제시한다. 실시간 저장 서버는 고속망에 연결된 사용자에 멀티미디어 화일의 실시간 검색을 위한 서비스를 제공하며 멀티미디어 화일의 삽입, 삭제 및 변경등의 비실시간 작업을 수행한다. 실시간 저장 서버의 설계 목표는 동시에 서비스 가능한 사용자의 수를 최대화 시키는 것이며 사용자 서비스의 실시간성 및 연속성이라는 제약조건을 만족시키야 한다. 위의 설계 목표와 제약조건을 만족시키기 위해 효율적인 멀티미디어 블럭의 검색을 지원하는 라운드 스케쥴링을 제안한다. 이는 디스크 유휴시간을 최소화한다. 또한 탐색시간을 최적화하는 RR-SCAN 디스크 암 스케쥴링을 제안한다. 디스크 구조는 성능 분석을 통해 디스크 배열을 대상으로 하였으며, 데이타를 블럭 단위로 디스크에 스트라이핑시킨다.

Abstract

In this paper we design a real-time storage server(RTSS) for remote multimedia playback applications and present an effective scheduling method for its high performance. RTSS performs the real-time retrieval of multimedia files for the remote clients over high speed network as well as non real-time insert, delete, update operations. The design goal is to maximize the number of clients which can be serviced simultaneously. RTSS must, however, maintain the continuity of each medium. To satisfy both the design goal and the constraint, we propose a new scheduling scheme, called round scheduling, to effectively retrieve multimedia

disk blocks. The round scheduling is shown to minimize disk idle time. We also present SCAN-like disk arm scheduling that optimizes seek time to access multimedia blocks. Disk architecture for our server adopts a disk array, where data blocks are striped across all the disks.

1 서 론

최근 컴퓨터, 통신, 저장 매체 기술의 발달은 고속망에 연결된 사용자에 멀티미디어 정보에 대한 실시간 서비스를 가능하게 하였다[1, 2]. 그러나 이 서비스를 제공하는 정보 시스템은 동시에 많은 사용자를 처리할 수 있는 고성능의 저장 서버를 필요로 한다. 이러한 저장 서버 개발시 문제점은 취급하는 데이타가 기존의 데이타와는 다른 특성을 갖는다는 것이다. 즉, 디지탈화된 멀티미디어 데이타의 검색 또는 저장시 사용자 서비스의 실시간성 및 연속성이 보장되어야 한다. 따라서 이에 대한 연구가 활발히 이루어지고 있다[2, 3, 4, 5, 6, 7, 8].

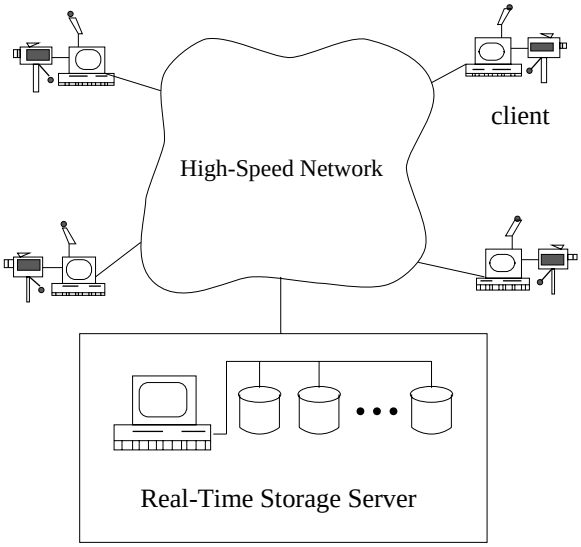

그림 1: 실시간 저장 서버의 구성

본 논문에서는 비디오, 오디오등 멀티미디어 데이타의 효율적인 검색을 위한 스케쥴링 기법을 제시하고 이를 이용한 실시간 저장 서버(Real-Time Storage Server)를 설계한다. 실시간 저장 서버는 영화, 교육용 프로그램, 광고 등의 멀티미디어 정보를 디지탈 형태로 저장하여 그림 1과 같이 고속망에 연결된 사용자에 서비스를 제공한다. 사용자는 영화나 프로그램 제목등으로 등록된 멀티미디어 객체를 선택하여 실시간 재생을 요구하고 실시간 저장 서버는 필요한 자원들을 확보한 경우 연결을 승인하고 멀티미디어 데이타를 실시간으로 검색 및 전송함으로써 서비스가 이루어진다. 미디어의 재생시 각 미디어의 연속성(continuity)과 미디어간의 동기화(synchronization)가 반드시 이루어져야 하고 가능한 많은 사용자를 동시에 서비스해야 한다[2, 4]. 이를 위해 실시간 저장 서버는

다음과 같은 필요 조건을 만족시켜야한다. 첫째, 대용량의 디스크를 가져야하며 디스크의 전송 대역폭(bandwidth)이 커야 한다. 둘째, 서비스 가능한 사용자의 수를 극대화 시키고 각 사용자 서비스의 실시간성 및 연속성을 보장하는 메카니즘을 제공해야 하며, 디스크의 사용율을 최대화하고 미디어의 QOS(Quality Of Service)를 만족시키는 실시간 디스크 스케쥴링을 제공해야 한다.

본 논문에서는 첫번째 조건을 만족시키기 위해 디스크 배열을 이용한다[1]. 또한 데이타를 블럭 단위로 스트라이핑(block striping)하고 패리티 드라이브를 각 디스크에 분산시키는 5단계 RAID 모델을 대상으로 하였다[9]. 5단계 RAID는 읽기 연산시 좋은 성능을 발휘하고[10] 실시간 저장 서버의 역할이 검색 위주라는 측면에서 가장 적합한 모델이라고 할 수 있으며, 4절에서 그 정당성을 증명한다. 또한 최근의 연구에 의해 가장 우수한 성능을 보임이 실험을 통해 제시되었다[18]. 그런데 RAID에서 신뢰도를 위한 여분의 디스크는 데이타 손상시 즉각적인 복구를 위한 것이므로 이러한 기능은 제외시킨다. 즉, 영화, 교육용 프로그램등의 응용을 고려할때 다른 저장 장치에 복사본을 유지하는 경우가 대부분이므로 신뢰도를 향상시키기 위한 기법은 고려하지 않는다.

두번째 조건을 위한 연구가 현재 활발히 진행되고 있다. [5]에서는 하나의 대용량 디스크에서 화일이 연속적으로 배치되어 있다는 가정을 기초로 하여 서비스 가능한 사용자를 최대화시키기 위해 minimal feasible WAS 알고리즘을 제안하였다. 그러나 계산의 복잡도가 매우 크다[2]. 또한 여유시간(slack time)을 토대로 한 디스크 스케쥴링 정책을 제시하였으나 탐색시간(seek time)을 최적화하기 위한 고려는 이루어지지 않았다. [2, 3]에서는 디스크 블럭을 적절히 분산시킴으로써 미디어의 연속성을 얻을 수 있다는 가정에서 출발하여 사용자 수를 최대화 시키는 QPMS(Quality Proportional Multi-subscriber Servicing) 알고리즘을 제시하였으나 디스크 스케쥴링과의 관련 연구는 이루어지지 않았다. 또한 [18]에 의하면 화일이 연속적으로 배치되었을 경우가 가장 성능이 우수함이 제시됨으로써 디스크 블럭의 분산은 불필요한 오버헤드를 초래할 가능성이 존재한다.

디스크 배열의 블럭 스트라이핑 기법을 이용한 연구는 [4]에 나와있다. 사용자 서비스의 실시간성 및 연속성을 보장하기 위한 새로운 검색의 수용 여부 검사 조건은 유도되어 있으나 효율적인 관리 기법은 제시되지 않아 디스크의 사용율 저하(under-utilization) 현상이 발생될 수 있다. 본 논문에서는 디스크의 유휴시간(idle time)을 최소화 시키는 라

[1] 디스크 배열은 현재의 기술로 144MB/s까지의 대역폭을 제공할 수 있다[17].

운드 스케쥴링을 제안하며, 이와 함께 디스크 암(arm) 스케쥴링으로서 탐색시간을 최적화하는 RR-SCAN을 제안한다. 이 스케쥴링 기법을 이용하여 설계된 실시간 저장 서버는 사용자 서비스의 실시간성 및 연속성을 만족함은 물론, 보다 많은 사용자를 동시에 서비스 할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 실시간 저장 서버의 응용 모델을 설정하기 위한 가정을 세운다. 3절에서는 효율적인 스케쥴링 기법을 이용한 실시간 저장 서버를 설계하고 4절에서 디스크 구조에 대해 살펴본 후, 5절에서 결론을 맺는다.

2 응용 모델의 설정

본 절에서는 다음과 같은 가정을 세움으로써 실시간 저장 서버의 응용 모델을 설정한다. 미디어 유닛(media unit)이란 각 미디어의 기본 단위로 비디오의 경우 프레임, 오디오의 경우 샘플, 텍스트의 경우 문자에 해당되고, 멀티미디어 블럭(multimedia block)은 디스크 저장 및 검색의 기본 단위로 미디어 유닛의 집합으로 구성된다. 멀티미디어 화일(multimedia file)은 사용자 요청의 기본 단위로서 다수의 멀티미디어 블럭으로 구성되며, 디렉토리는 멀티미디어 화일의 위치 및 속성을 저장하는 자료구조를 말한다.

가정 1 **사용자 컴퓨터는 실시간 저장 서버에서 필요로 하는 처리능력을 갖추고 있다.**

사용자 컴퓨터는 압축된 멀티미디어 화일의 복원(decompression), 미디어간의 재동기화, 네트워크상의 지터(jitter)를 처리하기 위한 버퍼링등을 담당해야 한다[6, 7]. 또한 서버측에서의 스케쥴링 및 선반입(prefetch) 알고리즘을 수행시키는데 필요한 사용자의 요건도 모두 갖추고 있음을 뜻한다.

가정 2 **실시간 저장 서버에서 전송한 데이타 패킷은 순서대로, 오류없이, 고정된 혹은 편차가 크지 않은 전송 지연 시간 내에 사용자 컴퓨터에 도착된다.**

네트워크에서의 데이타 패킷 전송 지연은 사용자측에서 멀티미디어 화일의 실시간 재생을 불가능하게 만드므로 실시간 저장 서버의 데이타 패킷을 고정된 전송 지연 시간 내에 전송시키는 메카니즘이 제공되어야 한다[11].

본 논문에서는 가정 1과 가정 2에 의해 멀티미디어 데이타의 연속성을 서버측면에서만 고려한다. 따라서 모든 데이타 블럭을 종료시한(deadline)내에 전송시켜야하

는 경성 실시간 제약조건(hard real-time constraints)을 만족시키는 실시간 저장 서버만을 대상으로 한다.

가정 3 실시간 저장 서버에 저장된 데이타의 삽입, 삭제 및 변경등의 연산에 대해 일반 사용자는 수행 불가능하고, 특정 권한을 갖는 시스템 관리자에 의해서만 가능하다.

일반 사용자는 공유 데이타의 검색만이 가능하며 공유 데이타의 관리는 특정 권한을 갖는 시스템 관리자가 담당한다. 이 가정의 중요한 의미는 시스템 관리자의 연산은 비실시간적인 특성을 가지므로 일반적인 사용자의 실시간 연산의 배경 작업(background job)으로 수행시킬 수 있다는 점이다.

가정 4 여러 미디어를 동기화된 형태로 한 블럭에 저장한다.

멀티미디어 블럭은 미디어가 배치되어있는 방식에 따라 동종 블럭(homogeneous block)과 이종 블럭(heterogeneous block)으로 나눌 수 있다[2][3]. 동종 블럭의 검색시에는 각 미디어 블럭을 접근해야 하므로 미디어 갯수만큼의 디스크 탐색시간이 필요하나 이종 블럭의 검색시에는 단 한번의 디스크 탐색이면 충분하다. 탐색시간은 CPU 명령에 비해 상당히 큰 시간이므로 시스템의 성능을 좌우하는 큰 척도가 된다[9]. 따라서 실시간 검색이 주가 되는 저장 서버에 적합한 형태는 이종 블럭 방식이라고 할 수 있으며 미디어간의 동기화를 자동적으로 수행할 수 있는 잇점을 추가로 얻을 수 있다.

가정 5 실시간 저장 서버에 저장된 한 멀티미디어 화일내의 멀티미디어 블럭당 재생 시간은 일정하다.

위의 가정은 경성 실시간 스케쥴링을 위한 분석을 가능하게 만든다[3]. MPEG[13]과 같은 VBR (Variable Bit Rate) 압축기법을 사용할 경우 정적인(deterministic) 서비스가 불가능하므로 지터(jitter)가 발생될 수 있다.

3 실시간 저장 서버의 설계

[2]동종 블럭은 각 멀티미디어 블럭에 한 미디어만이 저장되고 이종 블럭은 여러 미디어가 동기화되어 저장된다.

[3]이 가정은 [4]에서 'normalized stream', [12]에서 'heterogeneous stream'이라고 불린다.

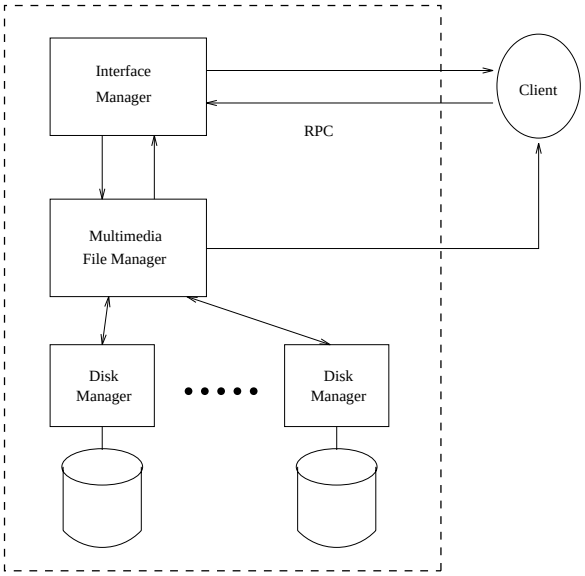

그림 2: 실시간 저장 서버의 논리적 기능 구조

실시간 저장 서버의 논리적 기능 구조 및 상호 관계는 그림 2와 같으며 각각의 기능을 살펴보면 다음과 같다: 인터페이스 관리자(interface manager)는 사용자 요청[4]을 받아 분석한 후 멀티미디어 화일 관리자에 서비스를 요청한다. 멀티미디어 화일 관리자(multimedia file manager)는 사용자 요청에 대한 상위 레벨 서비스(멀티미디어 화일 서비스)를 수행한다. 즉, 사용자의 새로운 검색 요청에 대한 수행 가능 여부 검사를 수행하고 디스크 관리자에 주기적인 선반입을 요구한다. 또한, 디스크 관리자에 의해 선반입된 멀티미디어 블럭을 통신 채널을 통해 사용자에 전송하는 역할도 담당한다. 디스크 관리자(disk manager)는 디스크 유휴시간을 최소화하는 라운드 스케쥴링 및 탐색시간을 최소화하는 디스크 암 스케쥴링을 담당하고 실제 디스크를 접근함으로써 사용자 요청에 대한 하위 레벨 서비스(디스크 서비스)를 수행한다. 디스크 관리자는 디스크 배열하의 각 디스크마다 하나씩 존재한다.

[4]일반적인 사용자 요청으로는 play, stop, pause, resume, rewind, fastforward 등이 있고, 시스템 관리자의 요청은 open, close, delete, record, insertblock, deleteblock 등이 있다.

표 1: 본 논문에서 사용된 기호

기호	설명	단위
T_{seek_max}	디스크의 최대 탐색시간	sec
$T_{seek\_max}^{large}$	대용량 디스크의 최대 탐색시간	sec
$T_{seek\_max}^{small}$	소용량 디스크의 최대 탐색시간	sec
R	디스크의 데이타 전송율	bits/sec
R^{large}	대용량 디스크의 전송율	bits/sec
R^{small}	소용량 디스크의 전송율	bits/sec
T_{play}	멀티미디어 블럭의 재생 시간	sec/block
s	멀티미디어 블럭의 크기	bits/block
m	디스크의 갯수	

3.1 멀티미디어 화일 서비스

실시간 저장 서버는 동시에 여러 사용자의 요청을 서비스해야 한다. 가장 좋은 경우는 모든 사용자가 같은 멀티미디어 화일을 요청한 경우로 단지 하나의 멀티미디어 화일을 디스크로부터 읽어 모든 사용자에 방송하면 된다. 그러나 여러 사용자들이 다른 멀티미디어 화일을 요청한 경우가 일반적이다. 따라서 사용자 요청에 대한 실시간 검색을 보장하여 멀티미디어 데이타의 연속성을 얻기 위한 기법이 필요하다. 본 절에서는 이를 위하여 새로운 검색의 수용 여부 검사를 위한 조건을 유도하고, 실시간 제약조건을 만족시키기 위한 선반입 및 데이타 전송을 설명한다.

3.1.1 새로운 검색의 수용 여부 검사 (Feasibility Test)

먼저 한명의 사용자에 대한 서비스 과정을 살펴보자. 한 멀티미디어 블럭의 재생 시간은 T_{play}이므로 매 T_{play} 시간마다 멀티미디어 블럭의 검색이 요구된다. 하나의 멀티미디어 화일은 디스크 배열에 블럭 스트라이핑되어 저장되어 있으므로 첫번째 블럭은 첫번째 디스크에서, 두번째 블럭은 두번째 디스크에서, j번째 블럭은 $(j \bmod m)$번째 디스크에서 검색된다. 그림 3은 m=4일때의 상황을 나타낸다.

매 T_{play} 시간마다 발생하는 멀티미디어 블럭의 검색 요청은 디스크 배열하의 여러 디스크에 분산되어, 각 디스크에서는 한 블럭의 검색 후 $m \times T_{play}$ 시간 이후에 멀티미디어 블럭의 검색 요구가 도착된다. 따라서 $m \times T_{play}$ 시간 이내에 한 멀티미디어 블럭의 검색

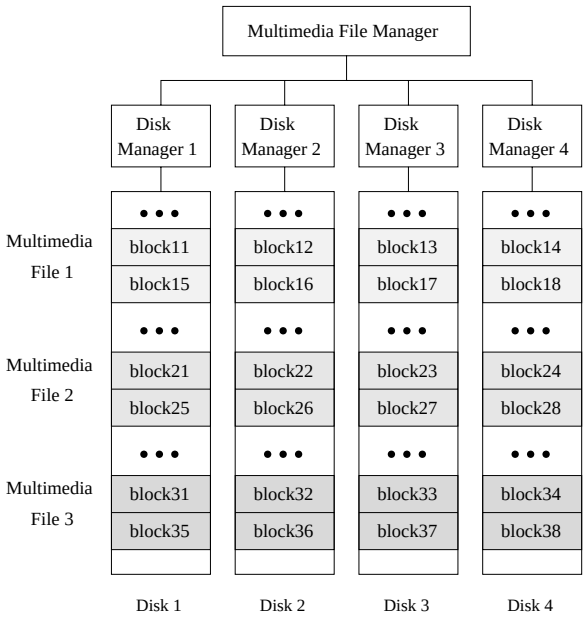

그림 3: m=4일때의 디스크 서비스

을 완료하여야 한다. 디스크의 한 블럭을 읽는 시간은 탐색시간과 데이타 전송시간으로 구성되므로 각각의 디스크는 다음의 조건을 만족시켜야 한다.

$$T_{seek\_max} + \frac{s}{R} \leq m \times T_{play} \tag{1}$$

n개의 검색이 요청된 경우로 확장시켜 보자. 멀티미디어 화일의 검색을 위한 요청 $r_1, r_2, \cdots, r_n$에 대한 서비스는 라운드-로빈 방식[4]에 의해 각 요청마다 한 블럭씩 이루어진다. 예를 들어 그림 3의 경우, block11, block21, block31, block12, block22, block32, $\cdots$의 순으로 서비스된다. 따라서 한 디스크내에서도 n개의 요청을 단위로 라운드가 형성되어 멀티미디어 블럭이 검색된다. 즉, 한 디스크에서 j 번째 라운드에서는 B_j^1, B_j^2, $\cdots$, B_j^n 블럭이 서비스되며 라운드는 주기적으로 반복된다. 여기에서 B_j^i는 r_i의 해당 디스크에서 j번째 멀티미디어 블럭이다[5]. 각 디스크에서 라운드 단위의 검색이 이루어지는 한편, 사용자 서비스의 연속성을 위한 디스크간의 동기화는 멀티미디어 화일 관리자의 선반입 명령을 통해 이루어진다. 각 라운드에서 n개의 멀티미디어 블럭이 검색되어야 하

[5]그림 3의 경우 디스크1에서는, 첫번째 라운드에서 block11, block21, block31, 두번째 라운드에서는 block15, block25, block35가 검색된다.

므로 이에 소비되는 시간은

$$n \times (T_{seek\_max} + \frac{s}{R}) \tag{2}$$

이다. 각 멀티미디어 블럭, B_j^i는 자신의 종료시한, 즉 $m \times T_{play}^i$내에 검색되어야 하므로 한 라운드의 주기는 n개의 요청 중 블럭당 최소 재생시간을 갖는 멀티미디어 화일에 의해 결정되어, $m \times T_{play}^{min}$이 된다($T_{play}^{min} = \min_{1 \le i \le n} T_{play}^i$). 따라서 식(1)은 n개의 요청을 서비스하기 위하여 다음과 같이 확장된다.

$$n \times T_{seek\_max} + n \times \frac{s}{R} \le m \times T_{play}^{min} \tag{3}$$

디렉토리를 접근하는 시간은 고정된 상수값으로 가정할 수 있으므로 분석에서 제외시켰으며, 디스크 블럭의 검색 시간의 한 요소인 회전 지연 시간(rotational latency)은 일반적으로 상수로 처리하기 때문에 최대 탐색시간에 포함시켜 고려될 수 있으므로 위의 분석에서는 생략하였다.

3.1.2 선반입 (Prefetch) 및 데이타 전송

새로운 검색의 수용 여부 검사를 마친 사용자의 검색 요청에 대한 서비스는 초기의 시작 준비 시간(start-up time) 이후 미디어의 연속성을 얻을 수 있도록 일정한 속도로 멀티미디어 블럭을 전송함으로써 이루어진다. 실시간성을 보장하여 멀티미디어 블럭을 전송시키기 위해서는 멀티미디어 블럭의 선반입을 수행해야 하므로 멀티미디어 화일 관리자는 주기적으로 디스크 배열하의 각 디스크 관리자에 동기화를 고려하여 선반입 명령을 내리고 디스크 관리자는 종료시한, 즉 주기내에 선반입을 완료해야 한다. 이때 3.2.2절에서 제안하는 디스크 암 스케쥴링 알고리즘의 특성상 한 사용자 요청에 대해 두개의 버퍼가 필요하므로 n명의 사용자를 처리하기 위해서는 $2n$개의 버퍼가 필요하다.

3.2 디스크 서비스

기존의 실시간 데이타베이스 시스템에서의 디스크 스케쥴링은 데이타들의 일관성을 유지함은 물론 모든 트랜잭션들을 종료시한 내에 서비스함을 목적으로 한다[14, 15, 16]. 그러나 멀티미디어 재생 응용을 위한 실시간 저장 서버에서는 디스크 요청의 특성이 기존의 실시간 데이타베이스 시스템과는 상당한 차이를 나타낸다. 첫째, 발생하는 요청이

읽기 위주이다. 더구나 쓰기 연산은 비실시간적인 특성을 가지므로 실시간 디스크 스케쥴링은 읽기 연산만을 대상으로한다. 둘째, 기존의 실시간 데이타 베이스 시스템에서 발생하는 트랜잭션들은 예측이 불가능하나, 실시간 저장 서버에서는 한 멀티미디어 화일내에서 주기적인 디스크 요청이 발생되므로 그 동작 형태가 예측 가능하다. 따라서 종료시한을 만족시키는 실시간 디스크 스케쥴링이 쉬워지고, 전체 탐색시간의 감소등으로 인해 기존의 실시간 디스크 스케쥴링보다 효율적으로 동작할 수 있다. 세째, 경성 실시간 제약 조건을 만족시키는 스케쥴링이어야 한다. 본 논문에서는 위의 특성들을 감안하여 디스크 스케쥴링을 라운드 스케쥴링과 디스크 암 스케쥴링으로 나누어 고려하였으며 이러한 디스크 서비스는 디스크 배열하의 각 디스크 관리자에 의해 수행된다.

3.2.1 라운드 스케쥴링 (Round Scheduling)

사용자 요청 r_i에 대한 각 멀티미디어 화일의 블럭당 재생 시간 T^i_{play}는 서로 다르다. 만일 $T^1_{play} = T^2_{play} = \cdots = T^n_{play} = T_{play}$ 이라면 매 $m \times T_{play}$ 시간마다 한 블럭씩 소비하므로 데이타가 누적되지 않는다. T^i_{play}가 다를 경우에는 블럭당 최소 재생 시간을 갖는 멀티미디어 화일을 기준으로 라운드가 형성되므로 블럭당 재생시간이 큰 멀티미디어 화일의 경우에는 데이타의 누적 현상이 발생된다[6]. 따라서 데이타의 누적을 방지하기 위하여 매 라운드마다 검색해야 할 블럭들을 스케쥴링해야한다. 이를 라운드 스케쥴링이라 부르기로 하며 그림 4에 간단한 예를 보인다. 이때 $m \times T^1_{play} = 1$, $m \times T^2_{play} = 2$, $m \times T^3_{play} = 3$, $m \times T^4_{play} = 1.5$이다.

그림 4에서 보듯이 첫번째 라운드에서는 B^1_1, B^2_1, B^3_1, B^4_1 블럭이 서비스되고 두번째 라운드에서는 B^1_2, B^4_2 블럭이 서비스된다. 그런데 두번째 라운드에서는 두 블럭만을 검색하므로 디스크의 유휴시간이 존재하게된다. 따라서 새로운 검색의 수용 여부 검사에서 실패한 사용자 요청도 이러한 유휴시간의 활용으로 서비스될 수 있다. 예를 들면 그림 4의 경우에서 동시에 4명의 사용자를 서비스 할 수 있다고 가정할때, 즉 한 라운드에서 4블럭만을 검색할 수 있을때 $m \times T^5_{play} = 1.5$인 r_5와 $m \times T^6_{play} = 2$인 r_6의 요청이 추가로 발생했을 경우 새로운 검색의 수용 여부 검사는 실패한다. 그러나 디스크 유휴시간의 활용으로 r_5와 r_6이 서비스 가능해진다. 그림 5는 이 예에 대한 스케쥴링 예를 보인다.

[6][4]에서는 이 문제에 대해 단지 멀티미디어 블럭의 검색을 중단한다. 따라서 디스크의 사용율 저하 현상이 발생된다.

라운드:	(1)	(2)	(3)	(4)	(5)	(6)	(7)	$\cdots$
r_1 :	B_1^1	B_2^1	B_3^1	B_4^1	B_5^1	B_6^1	B_7^1	$\cdots$
r_2 :	B_1^2		B_2^2		B_3^2		B_4^2	$\cdots$
r_3 :	B_1^3			B_2^3			B_3^3	$\cdots$
r_4 :	B_1^4	B_2^4		B_3^4	B_4^4		B_5^4	$\cdots$
재생시간:	0	1	2	3	4	5	6	$\cdots$

r_1 , r_2 , r_3 , r_4 : 사용자 요청
B_j^i : 해당 디스크에서 r_i의 j번째 멀티미디어 블럭

그림 4: 한 디스크에서의 라운드 스케쥴링 예

라운드:	(1)	(2)	(3)	(4)	(5)	(6)	(7)	$\cdots$
r_1 :	B_1^1	B_2^1	B_3^1	B_4^1	B_5^1	B_6^1	B_7^1	$\cdots$
r_2 :	B_1^2		B_2^2		B_3^2		B_4^2	$\cdots$
r_3 :	B_1^3			B_2^3			B_3^3	$\cdots$
r_4 :	B_1^4	B_2^4		B_3^4	B_4^4		B_5^4	$\cdots$
r_5 :		B_1^5	B_2^5		B_3^5	B_4^5		$\cdots$
r_6 :		B_1^6		B_2^6		B_3^6		$\cdots$
재생시간:	0	1	2	3	4	5	6	$\cdots$

r_1 , r_2 , r_3 , r_4 , r_5 , r_6 : 사용자 요청
B_j^i : 해당 디스크에서 r_i의 j번째 멀티미디어 블럭

그림 5: 한 디스크에서의 효율적인 라운드 스케쥴링 예

효율적인 라운드 스케쥴링으로 말미암아 서비스 불가능했던 r_5, r_6의 요청은 비록 초기의 지연시간은 존재하나(그림 5에서는 1초) 각각의 주기내에 검색되어 멀티미디어 데이타의 연속성을 보장할 수 있다.

정리 1 새로운 사용자 요청 r_a는 다음을 만족시키는 임의의 집합 Q가 존재하면 서비스 가능하다.

$$\sum_{i \in Q} k_i \quad \leq \quad 1 \quad \text{and} \quad a \in Q \tag{4}$$

$$\text{단,} \quad k_i \; = \; \frac{T_{play}^{min}}{T_{play}^{i}} \;\; , \;\; Q \; = \; \{i \mid r_i\text{는 사용자 요청}\}$$

증명 k_i는 사용자 요청 r_i에 대하여 라운드당 검색하는 평균 블럭수로 1이하의 값을 갖는다[7]. 즉, r_i는 한 라운드에 k_i 블럭을 읽는다. 어떤 집합 Q에 대해 $\sum_{i \in Q} k_i \leq 1$ 인 경우 $\{r_i \mid i \in Q\}$ 는 평균적으로 한 라운드당 한 블럭을 읽는 시간에 서비스될 수 있다. 즉, $r_i,\ i \in Q$ 는 $\lfloor 1/k_{min} \rfloor$ 라운드마다 $\lfloor k_i/k_{min} \rfloor$ 블럭씩만을 읽으면 된다($k_{min} = \min_{i \in Q} k_i$). 따라서 디스크의 유휴시간에 r_a는 서비스 가능하다. □

그림 5에서 r_5와 r_6에 대해 정리 1을 적용해 보자. $k_5 = 2/3, k_6 = 1/2$로 각각 $k_3 = 1/3$, $k_2 = 1/2$인 r_3, r_2와 함께 집합 Q를 이룬다. 따라서 r_2와 r_6는 매 라운드마다 번갈아 검색되고 r_3과 r_5는 세 라운드마다 각각 한 블럭, 두 블럭씩 검색되므로 사용자에 멀티미디어 데이타의 연속성을 보장하는 서비스를 제공할 수 있다. 그림 6은 정리 1을 이용한 효율적인 라운드 스케쥴링 알고리즘을 나타낸다.

3.2.2 디스크 암 스케쥴링 (Disk Arm Scheduling)

실시간 저장 서버의 설계 목표는 서비스 가능한 사용자 수를 최대화 시키는 것이다. 이를 위해 본 논문에서는 디스크 탐색시간을 최적화(optimize)하는 디스크 암 스케쥴링인 RR-SCAN을 제안한다. RR-SCAN은 변형된 라운드-로빈 방식과 SCAN 스케쥴링의 혼합으로, 주기적인 디스크 요청이라는 특성에 기인하며, 이러한 디스크 요청은 라운드 스케쥴링에 의해 발생된다. RR-SCAN은 디스크 블럭들을 디스크 헤드의 탐색 방향으로 스케쥴링하고 다음 라운드를 위하여 디스크 헤드를 계속 진행시켜 가장 바깥쪽 또는 가

[7] r_i에 대해 $T_{play}^{i} = T_{play}^{min}$ 일때 $k_i = 1$이다.

```
procedure Round_Scheduling( n, {T^1_play, T^2_play, ···, T^n_play}, l, {Q_1, Q_2, ···, Q_l})
begin
        T^min_play = min_{1≤i≤n}(T^i_play);
        for i := 1 to n do          /* k_i , S_i 초기화 */
                k_i := T^min_play / T^i_play;
                S_i := 0;
        endfor

        for j := 1 to l do          /* start_round_i 초기화 */
                tmp := 1;
                for each i in Q_j do
                        start_round_i := tmp;
                        tmp := tmp + 1;
                endfor
        endfor
        for i := 1 to n do
                if ( start_round_i = 0 ) then
                        start_round_i := 1;
        endfor

        round := 1;
        while ( all requests have been serviced ) do          /* main loop */
                for i := 1 to n do
                        if ( round = start_round_i ) then /* 서비스 시작 */
                                S_i := S_i + 1;
                                schedule i;
                        endif
                        if ( round > start_round_i ) then
                                if ( S_i < k_i ) then          /* 데이타 부족 */
                                        S_i := S_i + 1;
                                        schedule i;
                        /* round^th 라운드에서 r_i에 대한 멀티미디어 블럭을 검색함 */
                                endif
                                S_i := S_i - k_i;          /* 데이타 소비 */
                        endif
                endfor
                Send scheduled list to Disk Arm Scheduler;
                round := round + 1;
        endwhile
end
```

그림 6: 라운드 스케쥴링 알고리즘

장 안쪽 실린더에 위치시킴으로써 라운드가 종료된다. 따라서 한 라운드내에서의 디스크 탐색시간은 T_{seek_max}로 고정된다[8]. 또한 종료시한, 즉 라운드의 주기내에 블럭의 검색 가능성 여부는 멀티미디어 화일 관리자의 새로운 검색의 수용 여부 검사에서 이루어지므로, RR-SCAN은 미디어의 연속성이라는 경성 실시간 제약 조건을 만족시킨다. 또 다른 장점은 RR-SCAN의 성능이 디스크의 블럭 배치(disk layout)에 무관하다는 것이다. 한편, RR-SCAN의 단점은 한 사용자 당 두개의 버퍼가 필요하다는 것이다[9]. [12]에서 버퍼의 크기를 최적화하는 디스크 암 스케쥴링이 제안되었으나 실시간 저장 서버의 궁극적인 설계 목표는 사용자 수의 극대화이다. 다음의 정리에 의해 RR-SCAN은 최적임을 보이며 그림 7은 디스크 암 스케쥴러의 동작 개요를 나타낸다.

정리 2 RR-SCAN은 최적의 디스크 암 스케쥴링 방법이다. 즉, 한 라운드에서 n개의 요청에 대한 처리 시간의 최대 하한(greatest lower bound)은 $T_{seek\_max} + n \times s/R$ 이다.

증명 한 라운드에서 n개의 요청을 처리하는 시간은 디스크 탐색시간인

$$\sum_{i=1}^{n} T_{seek}^{i} \tag{5}$$

과 데이타 전송 시간인

$$n \times \frac{s}{R} \tag{6}$$

으로 나누어 생각할 수 있다. 여기에서 식(6)은 고정된 상수값으로 가변적인 부분은 식(5)이다. 식(5)를 최소화 시키기 위하여 RR-SCAN에서는 디스크 헤드의 탐색 방향이 바뀌지 않도록 요청을 배열하여 처리하고, 다음 라운드를 위하여 디스크 헤드는 계속 진행하여 가장 바깥 또는 가장 안쪽의 실린더에 위치함으로써 라운드가 종료된다. 따라서 RR-SCAN에서 한 라운드에 n개의 요청을 처리하는 시간은 $T_{seek\_max} + n \times s/R$ 이다. 한편, 식(5)의 최대 하한은 가장 안쪽 실린더의 블럭과 가장 바깥쪽 실린더의 블럭이 요

[8]디스크의 탐색시간이 탐색거리와 선형 비례하지 않은 voice coil 모델[15]에서는 T_{seek_max} = (트랙 수 × 트랙간 탐색시간)으로 정의된다.

[9]매 라운드마다 검색되는 순서가 바뀌므로 최악의 경우 연속해서 두 블럭을 검색할 수 있다.

```
procedure Disk_Arm_Scheduler()
begin
        arm_status := OUTMOST;
        Move disk arm outmost;
        while ( True ) do
                Get {B_1, B_2, ..., B_n} from Round Scheduler;
                {B'_1, B'_2, ..., B'_n} := RR-SCAN({B_1, B_2, ..., B_n}, arm_status);
                for each B'_i in {B'_1, B'_2, ..., B'_n}
                        Read block B'_i;
                endfor
                /* 다음 라운드를 위해 디스크 암을 끝까지 진행시킴 */
                if ( arm_status = OUTMOST )
                        Move disk arm inmost;
                        arm_status = INMOST;
                else        /* arm_status = INMOST */
                        Move disk arm outmost;
                        arm_status = OUTMOST;
                endif
        endwhile
end
```

그림 7: 디스크 암 스케줄러의 동작

청된 경우로 T_{seek_max}이다. 따라서 어떠한 스케줄링 정책에 의해서도

$$\sum_{i=1}^{n} T_{seek}^{i} + n \times \frac{s}{R} \geq T_{seek\_max} + n \times \frac{s}{R} \tag{7}$$

이므로, RR-SCAN은 최적이다. □

라운드 스케줄링과 RR-SCAN에 의하여 식(3)은 식(8)로 개선되며 식(8)은 최종적인 새로운 검색의 수용 여부 검사 조건을 나타낸다.

$$T_{seek\_max} + n \times \frac{s}{R} \leq m \times T_{play}^{min} \text{ or } \exists Q \text{ such that } \sum_{i \in Q} k_i \leq 1 \tag{8}$$

식(3)과 식(8)로부터 라운드 스케줄링과 RR-SCAN에 의한 성능 향상을 정량적으로 분석할 수 있으며 그림 8에 그 결과를 보인다. 여기에서, T_{seek_max} = 25msec, s = 60KB,

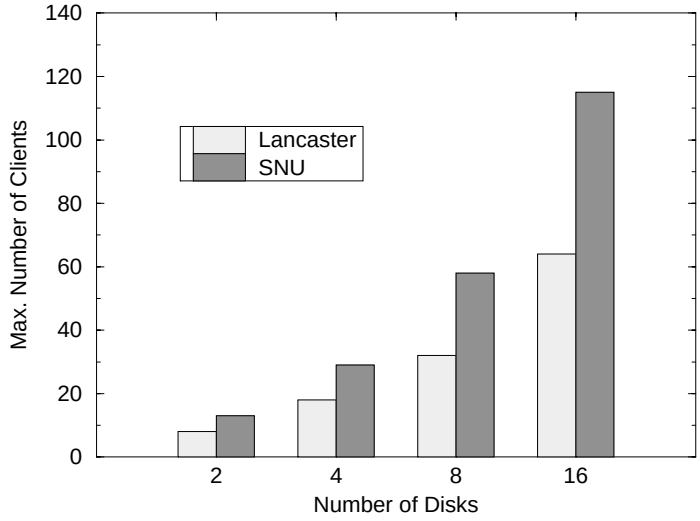

그림 8: 본 논문에서 제시한 스케쥴링 기법에 의한 성능 향상

$R = 2$MB/sec 로, 사용자 요청 r_i에 대한 블럭당 재생시간 T^i_{play}는 $200 \sim 440$ msec[10]의 균등 분포(uniform distribution)로 가정하였다.

4 디스크 구조에 대한 고찰

본 절에서는 SLED(Single Large Expensive Disk)와 디스크 배열의 성능 분석을 통하여 실시간 저장 서버에 가장 적합한 디스크 구조는 디스크 배열임을 보인다. 성능 분석의 척도는 서비스 가능한 사용자의 최대수가 되며 그 값은 새로운 검색의 수용 여부 검사의 조건식으로부터 유도된다. 분석을 간단히 하기 위하여 디스크 스케쥴링은 본 논문에서 제안한 라운드 스케쥴링과 RR-SCAN은 고려하지 않고 단순하면서도 일반적으로 공평하다고 알려진 선입선출(FCFS) 정책을 대상으로 하였고 CPU 오버헤드는 고려하지 않았다.

디스크 배열의 구조에서 블럭 스트라이핑 기법을 사용한 경우를 3.1.1절에서 살펴보았다. n개의 요청을 동시에 서비스하기 위한 조건은

$$n \times T^{small}_{seek\_max} + n \times \frac{s}{R^{small}} \leq m \times T^{min}_{play} \quad (9)$$

[10]멀티미디어 화일의 재생율(playback rate)을 MPEG-I 요구사항인 1.5Mbps로 가정하여 $s/1.5$Mbps$\pm$120으로 산출하였다.

이므로 n의 최대값 n_{max}는

$$n_{max}^{block\_striping} = \left\lfloor \frac{m \times T_{play}^{min}}{T_{seek\_max}^{small} + s/R^{small}} \right\rfloor \quad (10)$$

이다.

SLED의 경우에는 한번의 탐색시 하나의 블럭을 읽는 것은 너무 큰 오버헤드를 초래할 수 있으므로 한번의 탐색시 k개의 블럭을 읽는다고 가정한다. 실시간 저장 서버의 구현시, 다른 응용과는 달리 디스크 블럭의 크기는 대개 60KB 이상이다. 따라서 버퍼의 크기등을 고려할때 k값은 디스크 배열하의 디스크 갯수 m보다 작다고 할 수 있다. SLED에서 n개의 검색을 처리하기 위한 조건식 및 n_{max}는 다음과 같다.

$$n \times T_{seek\_max}^{large} + n \times k \times \frac{s}{R^{large}} \leq k \times T_{play}^{min} \quad (11)$$

$$n_{max}^{SLED} = \left\lfloor \frac{k \times T_{play}^{min}}{T_{seek\_max}^{large} + k \times s/R^{large}} \right\rfloor \quad (12)$$

디스크배열에서 데이타를 바이트 단위로 스트라이핑시키는 바이트 스트라이핑(byte striping)기법에 대해서도 살펴보자. 이는 3단계 RAID에 해당되며 수퍼 컴퓨터 I/O와 같이 디스크 블럭이 대단위로 요구될때 좋은 성능을 보인다[9, 10]. 바이트 스트라이핑은 데이타 전송율이 m배 늘어난 SLED로 모델링될 수 있으므로 n_{max}는 다음과 같다.

$$n_{max}^{byte\_striping} = \left\lfloor \frac{k \times T_{play}^{min}}{T_{seek\_max}^{small} + k \times s/(m \cdot R^{small})} \right\rfloor \quad (13)$$

식(10), (12), (13)을 비교함으로써 디스크 구조에 대한 성능을 비교할 수 있는데, 이때 SLED로는 IBM 3380 모델 AK4 메인 프레임 디스크를, 디스크 배열에는 Conner Peripherals의 CP 3100 소형 컴퓨터 디스크를 대상으로 하였다. 따라서 $R^{large} : R^{small} = 3 : 1$, $T_{seek_max}^{large} : T_{seek_max}^{small} = 2 : 1$이 된다[9].

정리 3 실시간 저장 서버에 가장 적합한 디스크 구조는 세개 이상의 디스크로 구성되는 디스크배열하의 블럭 스트라이핑이다. 즉, $n_{max}^{SLED} \leq n_{max}^{block\_striping}$, $n_{max}^{byte_striping} \leq n_{max}^{block_striping}$이다. (단, $k \leq m$)

증명 식(12)와 식(13)으로부터,

$$\begin{aligned} n_{max}^{SLED} &< n_{max}^{byte\_striping} \quad \text{if } m \geq 3 \\ n_{max}^{SLED} &> n_{max}^{byte\_striping} \quad \text{if } m = 2 \end{aligned} \tag{14}$$

이 성립한다. 또한 식(10)과 식(13)으로부터,

$$\begin{aligned} n_{max}^{byte\_striping} &\leq n_{max}^{block\_striping} \quad \text{if } k \leq m \\ n_{max}^{byte\_striping} &> n_{max}^{block\_striping} \quad \text{if } k > m \end{aligned} \tag{15}$$

이다. 따라서 세개 이상의 디스크로 구성되는 디스크 배열에 대하여 $k \leq m$ 일때, 식(14)와 식(15)로부터 다음의 식이 성립한다.

$$n_{max}^{SLED} < n_{max}^{byte\_striping} \leq n_{max}^{block\_striping} \tag{16}$$

□

5 결론 및 향후 과제

현재 멀티미디어 재생 응용에서 가장 큰 병목 현상은 디스크 시스템에서 발생된다. 그러나 한 사용자의 요청에 대하여 예측 가능하고 주기적인 디스크 요청이 발생되는 특성을 감안하여, 본 논문에서는 디스크의 유휴시간을 최소화하는 라운드 스케쥴링과 탐색시간을 최소화하는 RR-SCAN을 제안하였다. 이러한 효율적인 스케쥴링 기법을 이용하여 실시간 저장 서버를 설계하였으며 가장 적절한 디스크 구조인 디스크 배열의 블럭 스트라이핑을 대상으로 하였다. 본 논문에서 설계한 실시간 저장 서버는 개념적으로 간단하므로 쉽게 구현될 수 있을 것으로 생각된다.

실시간 저장 서버와 관련하여 많은 연구 과제가 남아있다. 첫째, VBR 압축 기법 사용시의 문제점을 해결해야 한다. 둘째, 여러명의 사용자가 서로 다른 시간에 같은 멀티미디어 화일을 요청한 경우 디스크 캐쉬를 통한 효율적인 서비스가 가능하다. 세째, 사용자가 우선 순위를 갖고 검색을 요청한 경우에 이에 대한 적절한 서비스를 수행해야 한다. 네째, 멀티미디어 블럭의 공유에 의해 저장 효율을 높일 수 있다. 다섯째, 멀티미디어 블럭 크기는 실시간 저장 서버의 구현시 버퍼의 크기등 여러 요소간의 장단점을 고려하여 결정되어야 한다.

참고 문헌

[1] W. D. Sincoskie, "Video On Demand: Is it feasible?," *IEEE GLOBECOM'90*, pp. 201–205, 1990.

[2] P. V. Rangan, H. M. Vin, and S. Ramanathan, "Designing an on-demand multimedia service," *IEEE Communications*, Vol.30, No.7, pp. 56–65, 1992.

[3] P. V. Rangan and H. M. Vin, "Desining file systems for digital video and audio," *Proceedings of the 13th ACM Symposium on Operationg Systems Principles (SOSP'91), Operating Systems Review*, Vol.25, No.5, pp. 81–94, 1991.

[4] P. Lougher and D. Shepherd, "The design and implementation of a continuous media storage server," *Proceedings of the 3rd International Workshop on Network and OS Support for Digital Audio and Video*, San Diego, California, pp. 63–74, 1992.

[5] D. P. Anderson, Y. Osawa, and R. Govindan, "Real-time disk storage and retrieval of digital audio/video data," *UC Berkeley Technical Report*, No. UCB/CSD 91/646, 1991.

[6] D. P. Anderson and G. Homsy, "A continuous media I/O server and its synchronization mechanism," *IEEE Comp., Spec. Issue on Multimedia Info. Sys.*, pp. 51–57, 1991.

[7] P. V. Rangan and S. Ramanathan, "Rate-based feedback techniques for continuity and synchronization in multimedia retrieval over high-speed networks," *UC San Diego Technical Report*, No. CS92–230, 1992.

[8] R. Staehli and J. Walpole, "Constrained-latency storage access," *IEEE Computer*, pp. 44–53, 1993.

[9] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," *ACM SIGMOD*, pp. 109–116, 1988.

[10] P. M. Chen, G. A. Gibson, R. H. Katz, and D. A. Patterson, "An evaluation of redundant arrays of disks using an Amdahl 5890," *ACM SIGMETRICS*, pp. 74–85, 1990.

[11] B. Wolfinger and M. Morau, "A continuous media data transport service and protocol for real-time communication in high speed networks," *Proceedings of the 2nd International Workshop on Network and OS Support for Digital Audio and Video*, Heidelberg, Germany, pp. 171–182, 1991.

[12] M.-S. Chen, D. D. Kandlur, and P. S. Yu, "Optimization of the grouped sweeping scheduling (GSS) with heterogeneous multimedia streams," *Proceedings of the First International Conference on Multimedia*, Anaheim, California, pp. 235–242, 1993.

[13] D. L. Gall, "MPEG: A video compression standard for multimedia applications," *Communications of ACM*, Vol. 34, No. 4, pp. 35–45, 1991.

[14] R. K. Abbott and H. Garcia-Molina, "Scheduling I/O requests with deadlines: A performance evaluation," *Proceedings of Real-Time Systems Symposium*, Lake Buena Vista, Florida, pp. 113–124, 1990.

[15] K. Hwang and H. Shin, "Real-time disk scheduling based on urgent group and shortest seek time first," *Proceedings of the 5th Euromicro Workshop on Real-Time Systems*, Oulu, Finland, pp. 124–130, 1993.

[16] J. R. Haritsa, M. Livny, and M. J. Carney, "Earliest deadline scheduling for real-time database systems," *Proceedings of Real-Time Systems Symposium*, San Antonio, Texas, pp. 232–242, 1991.

[17] 심원세, 신상석, "입출력 구조 동향," 주간 기술 동향 93–06, pp. 15–29, 1993.

[18] 서매실, 김치하, "디스크 배열에서의 멀티미디어 데이타 할당 방법," 한국정보과학회 논문지, 제21권, 제5호, pp. 887–899, 1994.