

# MPEG-1 스트림의 재구성을 통한 시간적 다중해상도 비디오 재생 기법

## (Reconstructing MPEG-1 Streams for Temporal Multi-resolution Video Playback)

### 요약

비디오 스트림은 매체의 특성상 여러가지 해상도로 재생될 수 있다. 그러나 현재의 표준 압축 기법들은 이를 지원하지 못하거나 아직 구현되고 있지 않다. 본 논문에서는 MPEG-1 비디오 스트림에 시간적 다중해상도 재생이 가능토록하기 위한 기법을 제시한다. 우선 일반적인 다중해상도 비디오 모델을 제안하고 이에 따라 MPEG-1 비디오 스트림을 재구성한다. 다중해상도 비디오 모델에 의해 재구성된 MPEG-1 비디오 스트림은 제안된 디스크 배치 기법에 의해 디스크 배열 모델에 기반한 서버에 저장되고 사용자가 요구한 해상도로 검색되어 디스크 및 네트워크 대역폭등의 자원을 효율적으로 사용하게 된다. 본 논문에서 제안한 기법에 의하면 빨리감기 등의 대화형 연산 또한 추가의 오버헤드 없이 가능하다. 그리고 제안한 기법의 실용성 및 효율성을 증명하기 위해 다중해상도 주문형 비디오 서비스의 프로토타입을 구현하며 다양한 실험을 통하여 성능을 측정하고 분석한다.

### Abstract

A video stream can be played back with various resolution levels due to its characteristics. Current compression techniques, however, do not support the multi-resolution playback. In this paper, we present a technique that MPEG-1 video streams can be played back with multiple resolution levels in temporal dimension. First, we propose the multi-resolution video model, and reconstruct MPEG-1 video streams based on the model. The reconstructed MPEG-1 video streams are stored on the disk-array-based server according to the proposed disk layout scheme. The streams are retrieved with respect to the requested resolution, so that the server resources, such as disk and network bandwidth, are effectively utilized. The technique proposed in this paper supports interactive operations like fastforward without additional overhead. In order to show the efficiency of the proposed technique, we implement the prototype of multi-resolution VOD services and perform various experiments on it for the performance analysis.

# 1 서론

정보화 사회로의 급속한 발전으로 나타난 멀티미디어 응용 분야에서는 방대한 양의 정보를 효율적으로 다루기 위하여 압축기술을 사용한다. 즉, 멀티미디어 부가 정보 서비스를 수행하는 서버는 많은 양의 자료를 저장하기 위해 압축기술이 필요하며, 실시간으로 동영상을 전송하는 화상회의 등의 응용은 압축된 동영상 데이터를 전송함으로써 필요한 네트워크 대역폭을 줄인다. 이러한 압축기술로는 실시간 동영상 서비스를 위한 H.261(p×64)[1], 저장된 동영상 서비스를 위한 MPEG[2] 등이 제안되어 널리 사용되고 있다.

한편, 동영상 서비스의 품질은 제공되는 비디오의 해상도(resolution)에 의해 결정된다. 그런데, 사용자는 여러가지 품질의 서비스를 원하는 경우가 일반적이다. 즉, 사용자마다 원하는 화면의 크기가 다르며, 접속된 네트워크의 종류도 다르다. 또한, 한 사용자의 경우에도 때에 따라 최상의 해상도로 검색하고 싶을 때가 있는 반면 낮은 해상도로 빠른 서비스를 원할 때가 있다. 그러나, 현재 널리 사용되는 압축기술은 이와 같은 다중해상도(multi-resolution)를 지원하지 못한다.

이에 따라 다중해상도 비디오 코딩의 개념이 제안되었다. 이때 사용자에게 제공되는 비디오의 데이터율(data rate)은 해상도에 비례해야 효율적인 서비스가 이루어짐을 고려해야 한다. Chiueh와 Katz[3]는 가우시안 피라미드(Gaussian Pyramid)와 라플라시안 피라미드(Laplacian Pyramid)에 기반한 다중해상도 비디오 코딩을 제안하고 이를 캘리포니아 대학의 이차원 디스크 배열(disk array)에 배치하였다. 이에 따라 추가의 오버헤드 없이 다중해상도 비디오 서비스를 제공함은 물론 서버의 효율이 높아짐을 밝히고 있다. 또한, Keeton과 Katz[4]도 확장가능한 압축 알고리즘에 의한 다중해상도 비디오를 이차원 디스크 배열에 배치하는 기법을 제안하였다. 그러나 위의 연구들은 특정 알고리즘에 기반한 압축기법으로 그 실용성에 문제점을 갖고 있을 뿐더러, 다중해상도의 서비스 품질 관리 측면 보다는 서버의 입출력율, 사용자의 응답시간 등에 초점이 맞추어져 있다. 최근 MPEG-1을 확장하여 4~60 Mbps의 스트림을 대상으로 하는 MPEG-2[5]에서는 가변분해능력(scalability)을 표준에 포함시켜 다중해상도를 지원토록하고 있으나 아직은 구현되고 있지 않다.

본 논문에서는 멀티미디어 정보 서비스를 수행하는 서버를 대상으로 하여 사용자에게 다중해상도의 서비스 품질(Quality of Service, 이하 QoS)을 제공하는 기법을 제안하고 그 프로토타입을 구현하고자 한다. 우선, 다중해상도 비디오 모델을 제안하고 이에 기반하여 기존의 MPEG-1 비디오 스트림을 재구성한다. 다중해상도 비디오 모델에 의해 재구성된 MPEG-1 비디오 스트림은 제안된 디스크 배치 기법에 의해 디스크 배열 모델에 기반한 서버에 저장되고 사용자가 요구한 해상도로 검색되어 디스크

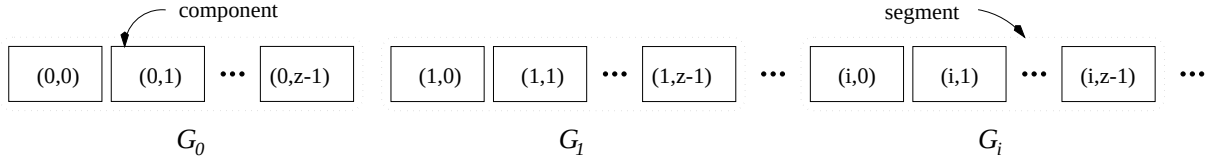


그림 1:  $z$  레벨 다중해상도 비디오 모델

및 네트워크 대역폭등의 자원을 효율적으로 사용하게 된다. 이때 사용자의 빨리감기(fastforward)와 같은 대화형 연산(interactive operation) 또한 추가의 오버헤드 없이 서비스된다. 그리고 본 논문에서 제안한 기법의 실용성 및 효율성을 증명하기 위해 다중해상도 주문형 비디오 서비스의 프로토타입을 구현한다. 또한, 구현된 프로토타입으로부터 성능에 관한 여러가지 실험이 수행되어 최적의 성능을 발휘하는 파라미터를 추출한다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에서 제안하는 기법이 소개된다. 즉, 다중해상도 비디오 모델, MPEG-1 스트림의 재구성, 디스크 배치 기법이 제안되며 제안된 기법에서의 대화형 연산의 지원 방법에 대해 논한다. 3절에서 제안된 기법의 구현 내용을 언급하며 이를 바탕으로 다중해상도 주문형 비디오 서비스를 구축한다. 4절에서 구현된 프로토타입의 성능을 분석하고 5절에서 결론을 맺는다.

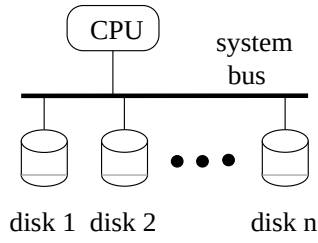
## 2 다중해상도 비디오

### 2.1 다중해상도 비디오 모델

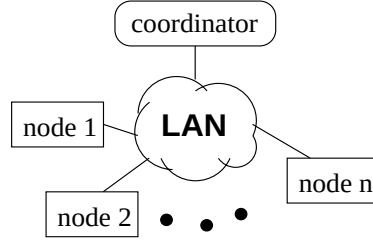
일반적으로 비디오의 해상도는 색도(chroma), 공간(spatial), 시간(temporal)의 세가지 차원에서 규정된다. 색도 해상도는 화소당 비트 수로 표현되고, 공간 해상도는 화면의 크기와 관련되며, 시간 해상도는 초당 프레임 수로 정의될 수 있다. 이 세가지 차원의 해상도에 의해 비디오 스트림은 다중해상도의 형태로 압축될 수 있다. 다음에서 제안하는 다중해상도 비디오 모델은 특정 알고리즘을 가정하지 않는다.

그림 1은  $z$  레벨 다중해상도 비디오 스트림을 나타낸다. 여기에서 하나의 스트림은 세그먼트(segment)의 집합으로 이루어지고 한 세그먼트는  $z$ 개의 요소(component)로 구성된다. 즉, 비디오 스트림  $S_i$ 에 대하여,

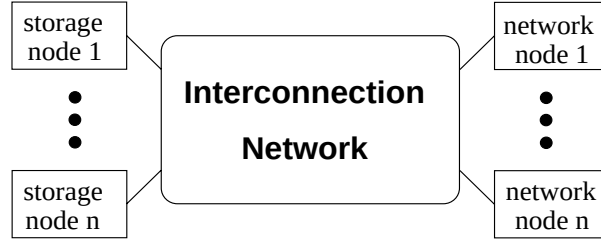
$$S_i = \{G_s^i \mid G_s^i \text{ 는 세그먼트, } 0 \leq s < m\}$$



(a) 단일 서버



(b) 분산 서버



(c) 병렬 서버

그림 2: 서버 모델

$$G_s^i = \{(s, c) \mid (s, c) \text{ 는 요소, } 0 \leq c < z\}$$

이고, 여기에서  $s$ 와  $c$ 는 각각 세그먼트와 요소를 나타낸다. 그리고  $m$ 은 세그먼트의 수, 즉 스트림의 길이를 나타낸다. 디스크에는 요소별로 나뉘어 저장되며 하나의 요소는 여러개의 고정 크기 디스크 블록에 저장된다. 이때 비디오 스트림의 해상도는 세그먼트내에서의 요소의 갯수로 정의된다. 따라서  $k$  레벨 해상도는 각 세그먼트마다  $k$ 개의 요소를 함침으로써 얻어진다. 즉,  $S = \{(s, c) \mid 0 \leq s < m, 0 \leq c < k\}$ 의 요소가 서비스된다. 제안된 다중해상도 비디오 모델에서 각 비디오 스트림은  $z$ 개의 해상도, 즉 QoS를 가지며 QoS 파라미터는 세그먼트내에서의 요소의 갯수  $k$ 로 표현된다.

## 2.2 디스크 배치

일반적으로 멀티미디어 정보 서비스를 제공하는 서버는 대용량의 데이터를 저장하고 높은 대역폭을 얻기 위해 디스크 배열에 기초한 구조를 채택한다[6]. 본 논문에서도 서버 구조를 위해 디스크 배열 모델을 대상으로한다. 그러나 시스템의 규모를 고려하여 디스크 배열 모델은 그림 2와 같이 다수의 디스크를 가진 단일 시스템으로부터 다수의 노드로 구성된 분산/병렬 시스템까지 확장될 수 있다. 대부분의 멀티미디어 서버는 이러한 구조하에서 설계 및 구현되었다(그림 2(a)[7], (b)[8, 9], (c)[10, 11]).

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|--------|--------|--------|--------|
| (0,0)  | (0,1)  | (0,2)  | (0,3)  |
| (1,3)  | (1,0)  | (1,1)  | (1,2)  |
| ...    | ...    | ...    | ...    |
| (0,3)  | (0,0)  | (0,1)  | (0,2)  |
| (1,2)  | (1,3)  | (1,0)  | (1,1)  |
| ...    | ...    | ...    | ...    |
| (0,2)  | (0,3)  | (0,0)  | (0,1)  |
| (1,1)  | (1,2)  | (1,3)  | (1,0)  |
| ...    | ...    | ...    | ...    |
| (0,1)  | (0,2)  | (0,3)  | (0,0)  |
| (1,0)  | (1,1)  | (1,2)  | (1,3)  |
| ...    | ...    | ...    | ...    |

(a)  $z = d$

| Disk 0 | Disk 1 | Disk 2 |
|--------|--------|--------|
| (0,0)  | (0,1)  | (0,2)  |
| (0,3)  | (1,0)  | (1,1)  |
| (1,2)  | (1,3)  | (2,0)  |
| (2,1)  | (2,2)  | (2,3)  |
| ...    | ...    | ...    |
| (0,2)  | (0,0)  | (0,1)  |
| (1,1)  | (0,3)  | (1,0)  |
| (2,0)  | (1,2)  | (1,3)  |
| (2,3)  | (2,1)  | (2,2)  |
| ...    | ...    | ...    |

(b)  $z > d$

|       | Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |         |
|-------|--------|--------|--------|--------|--------|---------|
| $S_1$ | (0,0)  | (0,1)  | (0,2)  | (0,3)  |        | $G_0^1$ |
|       | (1,3)  | (1,0)  | (1,1)  | (1,2)  |        | $G_1^1$ |
|       | ...    | ...    | ...    | ...    | ...    |         |
| $S_2$ |        | (0,0)  | (0,1)  | (0,2)  | (0,3)  | $G_0^2$ |
|       |        | (1,3)  | (1,0)  | (1,1)  | (1,2)  | $G_1^2$ |
|       | ...    | ...    | ...    | ...    | ...    |         |
| $S_3$ | (0,3)  |        | (0,0)  | (0,1)  | (0,2)  | $G_0^3$ |
|       | (1,2)  |        | (1,3)  | (1,0)  | (1,1)  | $G_1^3$ |
|       | ...    | ...    | ...    | ...    | ...    |         |
| $S_4$ | (0,2)  | (0,3)  |        | (0,0)  | (0,1)  | $G_0^4$ |
|       | (1,1)  | (1,2)  |        | (1,3)  | (1,0)  | $G_1^4$ |
|       | ...    | ...    | ...    | ...    | ...    |         |

(c)  $z < d$

그림 3: 다중해상도 비디오의 디스크 배치 예 ( $z$ : QoS 레벨,  $d$ : 디스크 수)

디스크 배열에 비디오 스트림을 배치할 때에는 디스크의 높은 대역폭을 보장함은 물론 스트림의 요청된 해상도에 상관없이 디스크의 부하 균배를 얻을 수 있어야 한다. 본 논문에서는 위의 요구조건을 만족하는 디스크 배치 기법을 제안하고자 한다. 우선, 그림 3을 통해 디스크 배치의 예를 살펴봄으로써 제안하는 배치 기법을 설명한다.

먼저, 디스크의 갯수  $d$ 와 스트림의 QoS 레벨  $z$ 가 같은 경우( $z = d$ )가 그림 3(a)에 나와있다. 이 때에는 한 세그먼트내의 요소들이 모든 디스크에 나뉘어 연속적으로 저장된다. 또한 디스크의 부하 균배를 위하여, 연속적인 세그먼트  $G_i$ 와  $G_{i+1}$ 의 첫번째 요소( $(i, 0)$ 과  $(i + 1, 0)$ )는 이웃한 디스크(디스크  $j$ 와 디스크  $j + 1$ )에 저장된다. 그리고, 각 스트림의 시작위치(첫번째 세그먼트의 첫번째 요소,  $(0, 0)$ )는 다수의 스트림이 동시에 요청된 상황의 디스크 부하균배를 고려하여 각 디스크에 분산시킨다. 예를 들어, 그림 3(a)의  $S_1$ 이 2 레벨의 QoS로 서비스 될때 첫번째 세그먼트는 디스크 0과 디스크 1에서 요소  $(0, 0)$ ,  $(0, 1)$ 이 검색되고 두번째 세그먼트는 디스크 1과 디스크 2에서 요소  $(1, 0)$ ,  $(1, 1)$ 이 검색된다.

$z > d$ 인 경우에는 한 세그먼트내의 요소가 한 디스크에 중복되어 저장될 수 밖에 없다. 그러나 그림 3(b)에서 보듯이 위에서 언급한  $z = d$ 의 경우와 유사한 방법으로 저장할 수 있다. 그리고  $z < d$ 인 경우에 한 세그먼트내의 요소들을 단지  $z$  개의 디스크에만 할당하고, 할당되지 않은 디스크들을 균일하게 분포시킨다면  $z = d$ 와 논리적으로 동일한 배치를 얻을 수 있다(그림 3(c) 참조).

위에서 예를 들어 언급한 디스크 배치 기법을 정리하면 다음과 같다. 비디오 스트림  $S = \{(s, c) \mid 0 \leq s < m, 0 \leq c < z\}$ 에 대하여 디스크  $i$ 에 저장되는 디스크 블록의 집합  $D_i$ 는

$$D_i = \{(s, c) \mid 0 \leq s < m, c = (i + j - s) \bmod d + l \cdot d, 0 \leq l < \left\lceil \frac{z}{d} \right\rceil\} \quad \text{if } z \geq d \quad (1)$$

이다<sup>1</sup>. 여기에서 스트림  $S$ 의 시작위치는 디스크  $j$ 이다. 그리고  $z < d$ 인 경우는  $z = d$ 인 경우와 논리적으로 동일하므로 같은 특성을 갖는다. 본 논문에서 제안된 다중해상도 비디오 배치 기법은 요청된 QoS 레벨에 무관하게 디스크의 부하균배를 얻을 수 있음이 [12]에 증명되어 있다<sup>2</sup>.

<sup>1</sup> 식 (1)에서  $\lceil \cdot \rceil$ 은 천정(ceiling) 함수, 즉 올림을 뜻한다.

<sup>2</sup> [12]에서는 이동 계산 환경하에서 다중해상도 비디오 배치 기법을 제안한다. 본 논문에서는 이를 일반화하여 실제 시스템에서 구현하였다.

|          |                          |
|----------|--------------------------|
| 시퀀스 층    | (sequence layer)         |
| GOP층     | (group of picture layer) |
| 픽처 층     | (picture layer)          |
| 슬라이스 층   | (slice layer)            |
| 매크로 블록 층 | (macro block layer)      |
| 블록 층     | (block layer)            |

그림 4: MPEG-1 비디오 스트림의 계층적 구조

## 2.3 대화형 연산의 지원

디스크 또는 네트워크 등의 추가의 자원을 필요로하지 않는 대화형 연산의 지원 또한 멀티미디어 응용 분야에서 큰 이슈로 대두되었다. 이를 위해 별도의 스트림을 저장하는 방법과 디스크 배열에서 필요한 블록만을 취하는 방법[13]등이 제안되었다.

본 논문에서 제안한 다중해상도 비디오 모델 및 디스크 배치 기법에 의하면 추가의 오버헤드없이 빨리감기등의 대화형 연산을 지원할 수 있다. 예를 들어, 4 레벨의 해상도로 정상 재생중인 스트림에 대해 빨리감기 요청이 도착한경우, 해상도를 2 레벨로 낮추면 2배의 빨리감기, 1 레벨로 낮추면 4배의 빨리감기가 같은 디스크 및 네트워크 대역폭내에서 수행될 수 있다. 빨리감기중에 낮추어진 해상도는 큰 문제가 되지 않는다.

## 2.4 MPEG-1 비디오 스트림의 재구성

2.1절에서 제안된 다중해상도 비디오 모델은 [3]의 피라미드 코딩 알고리즘, [4]의 서브밴드(subband) 코딩 알고리즘 및 기타 확장가능한 코딩 알고리즘에 의해 구현될 수 있다. 이때 세그먼트의 각 요소는 색도, 공간, 시간적 차원에서 각 알고리즘의 특성에 따라 구성된다. 그러나 이러한 알고리즘은 표준화가 이루어지지 않아 널리 사용될 수 없다. 본 논문에서는 저장된 비디오(stored video)의 표준인 MPEG 스트림을 재구성함으로써 제안된 다중해상도 비디오 모델을 구현하고자 한다.

우선 MPEG-1 비디오 스트림의 비트열은 그림 4와 같은 계층적 구조를 가지며 각 층의 시작코드에 의해 구별된다[14]. 최상위 층인 시퀀스 층은 화면 형태등을 지정하는 시퀀스 헤더로 구성되며 비트열 도중에 다시 존재할 수 있다. 그다음 GOP 층은 하나 이상의 픽처를 포함하여 임의 접근(random access)의 단위가 된다. 픽처층에는 I, P, B, D 종류의 픽처가 저장되며 이는 여러개의 슬라이스로 구

성된다. 각 슬라이스는 연속된 매크로 블록으로 구성되며 블록층은 필요한 DCT 계수를 포함한다.

MPEG-1 비디오 스트림의 비트열을 살펴보면 색도와 공간의 차원에서는 다중해상도를 가질수 없음을 쉽게 알 수 있다. 따라서 본 논문에서는 MPEG-1 비디오 스트림에 다중해상도를 부여하기 위해 시간 차원을 이용한다. MPEG-1 비디오를 본 논문에서 제안한 다중해상도 비디오 모델로 변환하면 다음과 같다.

- 하나의 GOP가 다중해상도 비디오 모델의 세그먼트에 해당된다.
- $I$  픽처가 세그먼트의 첫번째 요소에 해당된다.
- $P$  픽처들이 그다음 요소들을 구성한다.
- $B$  픽처들이 세그먼트의 나머지 요소들을 구성한다.

본 논문에서 제안한 비디오 모델의 해상도, 즉 QoS는 세그먼트 내에서의 요소의 갯수로 정의된다. 따라서, MPEG-1 비디오의 경우에는 한 GOP내에서의 픽처의 갯수에 의해 QoS의 레벨이 결정된다. 이것은 MPEG-1 표준의 압축 파라미터인  $N$ 과  $M$ 에 의해 달라지게 되는데,  $N$ 은 한 GOP 내의 픽처 수를  $M$ 은  $I$  또는  $P$  픽처가 나타나는 주기를 나타낸다. 예를들어,  $N = 15$ ,  $M = 3$ 일 경우 GOP 내에서의 픽처들의 배열은  $I_1 B_1 B_2 P_1 B_3 B_4 P_2 B_5 B_6 P_3 B_7 B_8 P_4 B_9 B_{10}$ 이 된다. 이경우 최대 15가지 레벨의 해상도를 부여할 수 있는데, 7가지 레벨의 해상도는  $(I_1)$ ,  $(P_2)$ ,  $(P_4)$ ,  $(P_1)$ ,  $(P_3)$ ,  $(B_1, B_3, B_5, B_7, B_9)$ ,  $(B_2, B_4, B_6, B_8, B_{10})$ 의 7개 요소가 한 세그먼트를 구성함으로써, 3가지 레벨의 해상도는  $(I_1)$ ,  $(P_1, P_2, P_3, P_4)$ ,  $(B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10})$ 의 3개 요소가 한 세그먼트를 구성함으로써 얻어질 수 있다.

### 3 다중해상도 비디오 관리자(MRVman)의 구현

본 절에서는 2절에서 제안된 기법의 실용성 및 효율성을 제시하기 위해 MPEG-1 비디오 스트림을 대상으로 다중해상도 서비스를 수행하는 다중해상도 비디오 관리자(Multi-resolution Video Manager, 이하 MRVman)<sup>3</sup>를 구현한다. 구현의 용이성을 위해 그림 2(a)의 모델을 대상으로 하였으나 시스템의 규모에 따라 그림 2(b), (c)의 모델로 확장될 수 있음은 쉽게 알 수 있다. 한편, 이와 유사한 환경에

---

<sup>3</sup>MRVman의 소스코드는 <http://cselab.snu.ac.kr/~cjs/research/MRVman.html>에서 구할 수 있다.



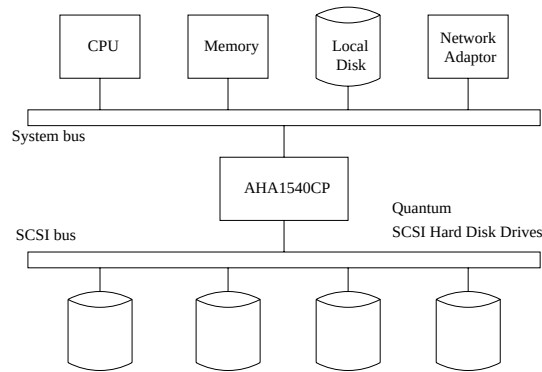


그림 5: MRVman의 하드웨어 환경

서 디스크 배열 관리자(DAman)<sup>4</sup>가 구현된 바 있다[15]. DAman은 새로운 기능을 쉽게 추가시킬 수 있도록 모듈화된 구조로 구현되어 있다. MRVman은 DAman의 화일 시스템과 스트라이핑 정책을 수정하여 구현되었다.

### 3.1 구현 환경 및 전체 구조

우선 MRVman의 하드웨어 환경은 그림 5와 같이 AHA-1540CP SCSI 어댑터와 Quantum 850MB SCSI 디스크 4개를 장착한 Pentium PC이다. 그 위에 QNX 실시간 마이크로커널 운영체제[16]가 탑재된다. QNX는 마이크로커널 운영체제이므로 디바이스 드라이버들은 응용 프로세스 수준의 서버로 동작한다<sup>5</sup>. 따라서 시스템 확장시에 융통성을 가지며 디버깅이 유리하다. 또한 메시지 및 프로세스의 우선순위 할당, 스케줄링 정책의 설정 등 실시간 관련 기능을 제공하므로 MRVman을 구현하는데 적합한 운영체제라 할 수 있다.

MRVman은 각 기능별 4개의 관리자로 구성되어 있으며 그 전체 구조는 그림 6과 같다. 각 관리자간의 제어흐름은 그림 6과 같이 관리자간의 서비스 요청으로 이루어지고 인터럽트 관리자는 인터럽트 핸들러에 의해 요청이 전달된다. SCSI 관리자는 SCSI 어댑터와 SCSI 디스크를 제어하여 실제 디스크로부터 데이터를 접근한다. 스트라이핑 관리자에 의해 사상된 SCSI 디스크 블록 번호에 의해 SCSI 명령을 내리며, SCSI 명령의 완료시 인터럽트 핸들러에 의해 보내지는 프락시(proxy) 메시지에 의해 인터럽트 관리자가 구동되어 발생된 인터럽트를 처리한다. 다중해상도 비디오를 다루기 위한 화일 시스템 관리자와 스트라이핑 관리자는 다음절에서 보다 자세히 언급한다. 한편, 응용 프로그램

<sup>4</sup>DAman의 소스코드는 <http://cse.snu.ac.kr/~cjs/proj/DAman/DAman.html>에서 구할 수 있다. DAman은 소프트웨어 디스크 배열 구현으로 No striping, AID3, AID5의 세가지 스트라이핑 정책을 지원한다.([15] 참조)

<sup>5</sup>MRVman의 프로세스 우선순위는 다른 디바이스 드라이버와 같은 수준을 유지한다.

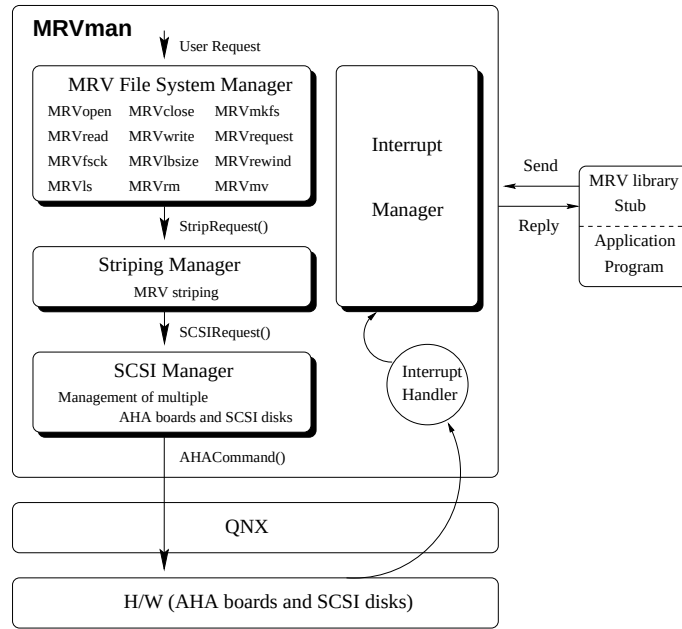


그림 6: MRVman의 전체구조

램은 MRVman의 실행 라이브러리와 함께 링크되며 이를 통해 MRVman에 서비스 요청 메시지가 전달되고 결과를 얻는다.

### 3.2 다중해상도 비디오 화일 시스템

먼저 MPEG-1 비디오 화일의 디스크 배치에 대해 설명한다. 프로토타입에서는 해상도의 차이를 확연히 느낄 수 있도록 고/중/저의 세가지 해상도를 구현하였다. 따라서, 2.4절에서 언급한대로 한 세그먼트는 하나의 GOP에 해당하고 세개의 요소로 구성되며, 첫번째 요소는 I 픽처, 두번째 요소는 P

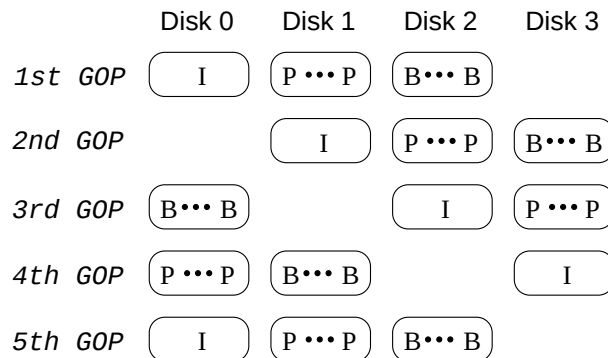


그림 7: 프로토타입에서 MPEG-1 비디오 화일의 디스크 배치

|         |                   |
|---------|-------------------|
| 수퍼 블록   | (super block)     |
| 디렉토리 블록 | (directory block) |
| 메타 블록   | (meta block)      |
| 비트맵 블록  | (bitmap block)    |
| 데이터 블록  | (data block)      |

그림 8: 다중해상도 비디오 파일 시스템의 구조

픽처의 집합, 세번째 요소는 B 픽처의 집합으로 이루어진다. 프로토타입은 4개의 디스크를 가지므로 그림 3(c)의 디스크 배치 기법에 의해 비디오 파일을 배치하면 그림 7과 같다. 이때, MPEG-1 비디오 파일의 픽처 구성과 디스크 배치에 관련한 다음과 같은 메타 정보가 유지되어야 한다.

```
struct meta_info {
    int picture_type;
    int size;
    int disk;
    int sector_number;
    int sector_count; };
```

이러한 메타 정보는 MRVman의 스트라이핑 관리자에 의해 관리된다.

한편, MRVman은 QNX 파일 시스템과는 별도의 다중해상도 비디오 파일 시스템을 갖는다. 이 파일 시스템은 그림 8과 같은 구조를 가지며 수퍼 블록, 디렉토리 블록, 메타 블록, 비트맵 블록은 디스크 배열의 대역폭을 보장하기 위하여 QNX 파일 시스템에 저장한다. 따라서 디스크 배열에는 오직 데이터 블록들만이 저장된다. 수퍼 블록은 각 관리자를 위한 정보들을 저장하고 디렉토리 블록은 저장된 비디오 파일의 이름, 메타 파일의 이름, 파일 크기 등을 저장한다. 메타 파일은 메타 블록에 저장된다. 그리고 비디오 파일은 크기가 매우 크므로 UNIX 파일 시스템과 같은 계층적 구조가 필요하지 않다. 따라서 다중해상도 비디오 파일 시스템은 평면적인 구조를 가지며 한 파일은 연속적인 데이터 블록에 할당된다. 이때 데이터 블록의 할당 상황을 비트맵 블록이 관리한다.

그리고 MRVman에서 지원하는 12개의 시스템 호출을 기반으로(그림 6참조) 그림 9와 같은 16개의 실행 라이브러리(run-time library)를 제공하고, 또한 이를 이용하여 표 1의 파일 시스템 관리 유틸리티를 구현하였다<sup>6</sup>. 다음에 MRVman의 기본 역할인 비디오 파일의 저장 및 검색에 대해 간략히 설

<sup>6</sup>MRVman, 실행 라이브러리, 및 파일 시스템 관리 유틸리티에 대한 사용법은 <http://cselab.snu.ac.kr/~cjs/>

```

int      mrv_open(char *file_name, int flag);
int      mrv_close(int handle);
int      mrv_read(int handle, char __far *buf, int max_size, int QoS_level);
int      mrv_write(int handle, char __far *buf, int size, int picture_type);
int      mrv_request(int n, Req_Blk_t *req_blk);
int      mrv_rewind(int handle);
int      mrv_mkfs(int logical_block_size);
int      mrv_fsck(void);
int      mrv_lbsize(void);
int      mrv_ls(DirEnt_t *dir, char *name);
int      mrv_rm(char *file_name);
int      mrv_mv(char *source, char *dest);
char __far * mrv_malloc(int size);
int      mrv_free(char __far *pointer);
char *   far2near(char __far *pointer, int size);
int      mrv_errmsg(void);

```

그림 9: 실행 라이브러리 함수

명한다.

- 저장 다중해상도 화일 시스템은 다른 화일 시스템과는 달리 MPEG-1 비디오 화일을 파싱(parsing)하여 픽처 종류별로 스트라이핑하여 저장하여야 한다. 따라서 저장은 표 1의 cp2mrv 명령을 통해서만 가능하다. cp2mrv 명령은 MPEG-1 비디오 화일을 파싱하여[14] 그림 9의 mrv\_write 함수를 통해 픽처 단위로 다중해상도 화일 시스템에 저장한다. 이때 MRVman은 픽처 종류에 의해 그림 7과 같이 디스크에 배치하며 메타 정보를 기록한다.
- 검색 검색은 그림 9의 mrv\_read 함수에 의해 이루어진다. cpmrv2 및 비디오 서버등의 응용 프로그램은 mrv\_read 함수를 사용함으로써 다중해상도의 비디오를 검색할 수 있다. MRVman은 mrv\_read로부터 주어진 해상도의 비디오 데이터를 버퍼 크기만큼 검색한다. 이때 검색 단위는 픽처로 한다. 예를 들어 주어진 버퍼 크기가 10개의 픽처를 수용할 수 있을 경우 10개의 픽처 데이터가 디스크 배열로부터 동시에 읽혀진다. 그런데 사용자 영역의 버퍼에는 원본 MPEG-1 비디오의 픽처 순서를 유지시킴을 목표로 한다. 현재 디스크에는 픽처 종류별로 스트라이핑되어 저장되어 있으므로 이를 원래 순서로 재구성하여 보내야 한다. 이는 기존의 MPEG-1 재생기(player)를 사용할 수 있게 하기 위함이다.

---

research/MRVman.html에 나와있다.

표 1: 다중해상도 비디오 화일 시스템 관리 유틸리티

| 명령      | 설명                  |
|---------|---------------------|
| cp2mrv  | 화일 시스템으로 화일을 복사     |
| cpmrv2  | 화일 시스템으로부터 화일을 복사   |
| mkmrvfs | 화일 시스템을 초기화         |
| mrvls   | 디렉토리 항목을 리스트        |
| mrvrm   | 화일을 지움              |
| mrvmv   | 화일의 이름을 변경          |
| mrvfsck | 화일 시스템의 일관성 검사 및 축약 |

본 논문에서 구현한 MRVman은 다음과 같은 특징을 갖는다. 첫째, 기존의 MPEG-1 비디오 화일을 재구성함으로써 다중해상도의 서비스를 지원한다. 이는 사용자의 요구, 사용자 컴퓨터의 처리 능력, 접속된 네트워크의 대역폭등에 적합한 최적의 서비스가 가능함을 뜻한다. 또한, 초과된 사용자로 인해 대역폭등의 자원이 부족할 경우 사용자에게 의해 허락된 수준으로 해상도를 낮춤으로써 추가의 사용자를 서비스할 수 있다. 둘째, 빨리감기 등의 대화형 연산을 지원한다. 추가의 자원을 소비하지 않고 대화형 연산을 지원하는 것은 아직 해결되지 못한 문제이다. 셋째, 다중해상도 및 대화형 연산의 서비스시 단지 필요한 데이터만을 사용한다. 즉, 디스크 또는 네트워크 대역폭 등 서버의 자원을 효율적으로 사용함으로써 보다 더 많은 사용자를 서비스할 수 있다. 넷째, 본 논문에서 제안한 디스크 배치 기법을 이용함으로써 디스크 배열의 디스크간에 완벽한 부하균배를 얻는다. 이 또한 디스크 대역폭을 최대한 이용할 수 있게 한다. 다섯째, 한 화일에 대한 디스크 블록을 물리적으로 연속하게 배치함으로써 탐색 시간의 오버헤드를 줄인다. 그러나 이러한 블록의 배치는 화일의 빈번한 삽입, 삭제시 단편화(fragmentation) 현상을 초래할 수 있으나 대상 응용이 WORM(Write Once Read Many)이라는 점을 고려하면 큰 문제가 되지 않는다. 또한, 심한 단편화 현상시 표 1의 mrvfsck 명령으로 축약(compaction)할 수 있다. 여섯째, 버퍼 캐쉬를 사용하지 않는다. 비디오 서버와 같은 대상 응용을 고려할때 버퍼 캐쉬의 크기로 수십 MB를 사용하지 않는 한, 캐쉬 적중(hit)은 발생하지 않을 확률이 매우 높다. 따라서 저장시에는 DMA를 통하여 사용자 영역에서 디스크로 직접 쓰여진다. 그러나 검색시에는 픽처들의 순서를 재구성해야 하므로 버퍼를 사용한다. 재구성은 사용자 영역으로 복사시 일어나므로 추가의 메모리 복사는 발생하지 않는다.

### 3.3 다중해상도 주문형 비디오 시스템

본 연구진은 다중해상도 비디오 재생을 확인하기 위하여 MRVman을 기반으로 다중해상도 주문형

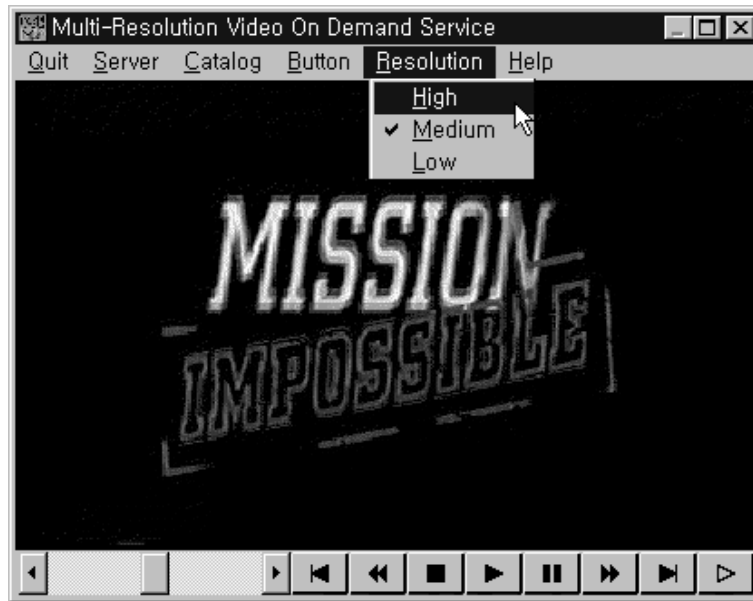


그림 10: 클라이언트 프로그램의 화면

비디오 시스템을 구축하였다. 서버와 클라이언트는 이더넷(Ethernet)으로 연결되고 사용자 요청등의 패킷은 TCP 프로토콜을, 데이터 패킷은 UDP 프로토콜을 사용한다. 클라이언트는 Windows '95 상에서 Real-Magic MPEG 디코더 보드를 사용하여 구현되며 그림 10과 같은 화면 구성을 갖는다. Real-Magic MPEG 디코더 보드에서 제공하는 기존의 라이브러리<sup>7</sup>만 가지고도 고/중/저의 세가지 해상도를 지원할 수 있었다<sup>8</sup>. 또한 빨리 감기를 구현할 수 있었는데, MPEG-1 스트림의 특성상 특정 배수의 빨리 감기는 지원되지 않는다.

## 4 성능 분석

본 절에서는 MRVman으로부터 그 성능을 측정하여 분석한 결과를 제시한다. 이때 MPEG-1 비디오는 영화 “Mission Impossible”의 약 20분간의 분량을 MPEG-1 인코더[17]로 압축한 화일(이하 mission.mpg)을 이용하였다. 그리고 MRVman의 수행시간은 본 연구진이 제작한 0.1 마이크로초( $\mu s$ )의 해상도를 갖는 타이머 보드를 통하여 측정하였다.

<sup>7</sup><http://www.sigmadesigns.com>에서 구할 수 있다.

<sup>8</sup>중 또는 저 해상도의 서비스 스트림의 데이터율에 기반하여 시간 지연을 준다.

표 2: MRVman 각 관리자의 평균 수행시간

| 관리자        | 수행시간 (읽기)       |
|------------|-----------------|
| 요청 전달      | 94.0 $\mu s$    |
| 화일 시스템 관리자 | 389.1 $\mu s$   |
| 스트라이핑 관리자  | 4839.2 $\mu s$  |
| SCSI 관리자   | 320.7 $\mu s$   |
| 인터럽트 관리자   | 16918.4 $\mu s$ |

#### 4.1 MPEG-1 비디오 데이터

mission.mpg 화일을 분석한 결과<sup>9</sup>,  $N = 15$ ,  $M = 3$ 의 파라미터로 코딩되어 한 GOP는 1개의  $I$  픽처, 4개의  $P$  픽처, 10개의  $B$  픽처로 구성되어 있음을 알 수 있었다. 그리고  $I$ ,  $P$ ,  $B$  픽처의 평균 크기는 각각 16657, 8183, 4399 바이트로 그 비는 3.79:1.86:1이 된다. 따라서 고해상도는 1.5Mbps, 중해상도는 0.8Mbps, 저해상도는 0.18Mbps의 데이터율로 서비스 된다. 그런데 이러한 픽처간의 크기 비율은 MPEG 스트림의 종류에 따라 매우 다르다. [18]의 연구 결과를 살펴보면, 영화의  $I$ ,  $P$ ,  $B$  픽처의 평균 크기 비율은 8.2:2.1:1, 움직임이 많은 스포츠 경기의 중계는 5.1:2.4:1, 뉴스등 움직임이 적은 스트림은 9.5:3.0:1로 나타나있다. 따라서 스트림의 종류와 압축 파라미터( $N$  및  $M$ )에 따라 2.4절에서 언급한 방법에 의해 여러가지 해상도를 부여할 수 있음을 알 수 있다.

#### 4.2 MRVman 각 관리자의 수행시간

먼저 MRVman의 각 기능별 관리자의 평균 수행시간을 살펴보면 표 2와 같다. 이 값은 하나의 `mr_read` 함수를 통하여 1개의 GOP 데이터 전체(15개의 픽처)를 검색하는 데 소요된 시간이다. 이 값은 수십 ms의 디스크 접근 시간에 비해 상대적으로 작은 값을 나타낸다. 표 2에서 보듯이 스트라이핑 관리자와 인터럽트 관리자가 상대적으로 큰 부분을 차지하는데, 스트라이핑 관리자는 메타정보를 읽어오기 때문이며 인터럽트 관리자는 사용자 영역으로 데이터를 복사하는 데 드는 오버헤드 때문인 것으로 사료된다. 데이터 복사 오버헤드를 살펴보기 위해 DMA를 사용하는 쓰기시 인터럽트 관리자의 수행시간을 측정한 결과, 1개의 GOP 데이터를 저장하는데 평균 1672  $\mu$ 초가 소요되었다. 이는 읽기시의 10%인 것으로 보아 데이터 복사 오버헤드는 인터럽트 관리자 수행시간의 90%를 차지하는 것을 알 수 있다.

<sup>9</sup>MPEG-1 화일에 대한 분석기는 <http://cselab.snu.ac.kr/~cjs/research/MPEGAnal.html>에서 구할 수 있다.

표 3: 논리 블록 크기에 대한 저장 오버헤드 및 검색 시간

| 논리 블록의 크기 | 저장 오버헤드 | 검색 시간            |
|-----------|---------|------------------|
| 0.5 KB    | 3.8 %   | 116504.2 $\mu$ s |
| 1 KB      | 9.3 %   | 94869.7 $\mu$ s  |
| 2 KB      | 19.2 %  | 104482.4 $\mu$ s |
| 4 KB      | 33.9 %  | 120853.0 $\mu$ s |

표 4: 디스크 수에 대한 검색 시간

| 디스크 수 | 검색 시간            |
|-------|------------------|
| 1     | 118159.0 $\mu$ s |
| 2     | 109420.7 $\mu$ s |
| 3     | 94574.9 $\mu$ s  |
| 4     | 94869.7 $\mu$ s  |

### 4.3 논리 블록 크기의 영향

MRVman은 3.2절에서 언급한 바와 같이 MPEG-1 비디오 화일을 파싱하여 픽처단위로 디스크에 저장한다. 이때 디스크에 저장하는 단위는 논리 블록(logical block)이 되며 일반적인 운영체제에서는 1 ~ 4 KB를 그 크기로 한다. 일반적으로 논리 블록의 크기는 디스크의 저장 효율 및 검색 성능에 있어 큰 영향을 미친다. 논리 블록의 크기가 클 경우에는 저장 효율이 나빠지나 검색 성능은 우수할 것으로 추정되고, 논리 블록의 크기가 작을 경우에는 저장 효율은 좋아지나 검색 성능이 저하될 가능성이 존재한다. 따라서 논리 블록의 크기에 따른 저장 효율 및 검색 시간을 MRVman에서 측정한 결과 표 3과 같았다. 표 3에서 저장 오버헤드는 논리 블록 단위로 할당된 전체 크기를 기준으로 실제 데이터가 저장되지 않은 부분을 계산하여 그 오버헤드를 나타낸 것이고, 검색 시간은 앞의 실험과 마찬가지로 1개의 GOP 데이터를 검색하는데 소요된 평균 시간을 나타낸다.

표 3의 결과를 살펴보면 1KB의 논리 블록이 검색 시간 측면에서 약 10 ~ 20 ms 정도 우수한 성능을 발휘하는 것을 알 수 있다. 그리고 1KB 이상 크기의 논리 블록에서는 오버헤드 부분이 너무 커서 검색 시간에서도 장점을 취하지 못하는 것으로 나타났다. 따라서 MRVman에서 최적의 논리 블록 크기는 1KB라는 것을 알 수 있다.



표 5: 해상도에 대한 검색 시간

| 해상도 | 검색 시간           |
|-----|-----------------|
| 고   | 94869.7 $\mu s$ |
| 중   | 50308.7 $\mu s$ |
| 저   | 25307.7 $\mu s$ |

표 6: 복수 스트림에 대한 검색 시간

| 스트림의 수 | 검색 시간            |
|--------|------------------|
| 1      | 94869.7 $\mu s$  |
| 2      | 163205.3 $\mu s$ |
| 3      | 244365.6 $\mu s$ |
| 4      | 326490.1 $\mu s$ |
| 5      | 396571.3 $\mu s$ |

#### 4.4 디스크 수 및 해상도의 영향

표 4는 디스크 수를 변화시키면서 그에 따른 검색 시간을 측정한 결과를 나타낸다. 현재 비디오의 해상도가 고/중/저 세가지로 주어지므로 3개와 4개의 디스크에 스트라이핑한 경우가 같은 응답시간을 보였다. 그러나 4개의 디스크의 경우 유틸 디스크에 의한 대역폭으로 보다 많은 사용자를 서비스 할 수 있음은 자명하다. 디스크의 갯수가 1 또는 2 일 경우에는 분산되어 있던 디스크 검색 요청이 한 디스크로 집중됨으로써 평균 검색 시간이 증가하였음을 알 수 있다. 그런데 그 차이가 10 ~ 20 ms로 나타난 것은 필요한 디스크 블록이 연속적으로 저장되어 탐색 시간에 대한 오버헤드가 거의 없기 때문인 것으로 사료된다. 그러나 복수 스트림의 서비스 요청시 그 차이가 클 것으로 예상된다.

지금까지의 실험은 모두 고 해상도로 검색할 경우의 성능을 제시하였다. 해상도를 낮추었을 경우 검색되는 픽처의 수가 작아지므로 검색 시간이 줄어들게 된다. 표 5가 그 결과를 나타낸다. 따라서 MRVman의 성능이 포화된 상황에서는 기존 비디오 스트림의 해상도를 낮춤으로써 추가의 사용자를 서비스 할 수 있다.

표 7: 각 디스크에서 검색되는 디스크 블록의 수 ( $m = 1000$ )

| 해상도 | 디스크 0 | 디스크 1 | 디스크 2 | 디스크 3 |
|-----|-------|-------|-------|-------|
| 고   | 25247 | 25280 | 25248 | 25254 |
| 중   | 12954 | 12955 | 12938 | 12931 |
| 저   | 2620  | 2563  | 2545  | 2598  |

#### 4.5 복수 스트림의 영향

위의 실험들은 모두 단일 비디오 스트림에 대한 성능을 언급하였다. 그러나 비디오 서버 등의 대상 응용을 고려하면 복수 스트림의 서비스가 필수적이다. 비디오 서버에서는 요청된 사용자에게 대한 디스크 블록을 주기적으로 검색하는 것이 일반적이다[6]. 이를 위해 MRVman에서는 복수개의 읽기/쓰기 요청을 처리하기 위해 `mrv_request` 함수를 라이브러리로 제공한다. 표 6에서 `mrv_request` 함수를 통한 복수 스트림에 대한 검색 시간을 측정한 결과를 보인다. 스트림의 수에 따라 검색 시간이 비례하여 증가하지 않는 것은 디스크 부하 균배에 기인한다. 즉, 디스크 배치 기법을 살펴볼 때, 1개의 GOP 검색시 유틸 디스크들이 존재하는데 복수 스트림시에는 부하 균배가 이루어진 상태에서 모든 디스크들이 동작하기 때문이다.

#### 4.6 디스크 부하균배

본 논문에서 제안한 다중해상도 배치 기법에서 비디오 스트림은 요소별로 디스크에 저장된다.  $(s, c)$  요소( $s$ 번째 세그먼트의  $c$ 번째 요소)가  $x_{s,c}$  개의 디스크 블록에 저장된다고 할때, 비디오 스트림  $S = \{(s, c) | 0 \leq s < m, 0 \leq c < z\}$ 를  $k$  레벨로 서비스하기 위해서는  $\{(s, c) | 0 \leq s < m, 0 \leq c < k\}$ 의 요소를 검색해야 한다. 따라서 총  $\sum_{s=0}^m \sum_{c=0}^{k-1} x_{s,c}$  개의 디스크 블록을 검색해야 한다. 이때 디스크에서 검색되는 디스크 블록의 갯수가 각 디스크마다 균일해야 자원의 효율적인 이용이 가능해진다.

2.2절에서 언급했듯이 다중해상도 배치 기법은 요청된 QoS 레벨에 무관하게 디스크의 부하균배를 얻을 수 있음을 표 7에서 살펴볼 수 있다. 비록 가변 비트율 스트림의 특성상 각 요소의 크기가 다르지만 효율적인 배치 기법으로 인해 디스크의 부하균배를 얻을 수 있다.

## 5 결 론

본 논문에서는 MPEG-1 비디오 스트림에 다중해상도 재생이 가능하도록 하기 위한 기법을 제시하고 프로토타입을 구현함으로써 그 실용성을 증명하였다. 이를 위해 우선 일반적인 다중해상도 비디오 모델을 제안하였고 MPEG-1 비디오 스트림을 재구성하여 다중해상도 비디오 모델에 사상함으로써 새로운 압축 기법에 의하지 않고도 기존의 MPEG-1 비디오 스트림에 다중해상도를 부여하였다. 다중해상도 비디오 모델에 의해 재구성된 MPEG-1 비디오 스트림은 제안된 디스크 배치 기법에 의해 디스크 배열 모델에 기반한 서버에 저장되고 사용자가 요구한 QoS로 검색되어 디스크 및 네트워크 대역폭을 효율적으로 사용하게 된다. 이때 빨리감기와 같은 대화형 연산 또한 추가의 오버헤드 없이 서비스 된다.

제안된 기법을 구현한 MRVman은 QNX 실시간 마이크로커널 운영체제하의 서버로 존재하고 복수개의 SCSI 디스크를 제안된 기법에 의해 관리한다. MRVman은 화일 시스템 관리자, 스트라이핑 관리자, SCSI 관리자, 인터럽트 관리자로 구성되어 기능의 추가/변경/삭제가 용이한 구조로 설계되어 있다. 그리고 비디오 서버 등의 응용 프로그램을 위하여 16개의 실행 라이브러리를 제공하고 7개의 화일 시스템 관리 유틸리티를 구현하였다. 이를 바탕으로 다중해상도 주문형 비디오 서비스를 구축하였으며, 비디오 서버는 MRVman과 같은 환경에서, 그리고 클라이언트는 MPEG 디코더 보드를 장착한 윈도우즈 환경에서 동작한다. 구현된 프로토타입에서 고/중/저 세가지 해상도의 서비스와 빨리감기 연산을 확인할 수 있었다. 마지막으로 여러가지 성능 측정을 위한 실험을 수행하여 MRVman의 성능을 분석함은 물론, 최적의 성능을 위한 파라미터를 추출하였다. 본 논문에서 구현한 프로토타입은 대규모 병렬/분산 시스템으로 확장되어 주문형 비디오 서비스에 다중해상도를 부여할 수 있을 것으로 사료된다.

## 참고 서적

- [1] M. L. Liou, "Overview of the p×64 kbps vico coding standard," *Communications of ACM*, Vol. 34, No. 4, pp. 59–63, 1991.
- [2] D. L. Gall, "MPEG: A video compresstion standard for multimedia applications," *Communications of ACM*, Vol. 34, No. 4, pp. 46–58, 1991.
- [3] T. Chiueh and R. H. Kats, "Video representation for parallel disk arrays," *Proc. of ACM Multimedia '93*, pp. 401–409, 1993.
- [4] K. Keeton and R. H. Kats, "The evaluation of video layout strategies on a high-bandwidth file server," *Proc. of Network and Operating System Support for Digital Audio and Video*, pp. 237–248, 1993.

- [5] ISO/IEC 13818, "Information technology - generic coding of moving pictures and associated audio"
- [6] D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia storage servers: A tutorial," *IEEE Computer*, Vol. 28, No. 5, pp. 40–49, 1995.
- [7] F. A. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID<sup>TM</sup> – A disk array management system for video files," *Proc. of ACM Multimedia '93*, pp. 393–400, 1993.
- [8] C. S. Freedman and D. J. DeWitt, "The SPIFFI scalable video-on-demand system," *Proc. of 1995 ACM SIGMOD*, pp. 352–363, 1995.
- [9] W. J. Bolosky, et. al., "The tiger video fileserver," *Proc. of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 97–104, 1996.
- [10] J. Cho and H. Shin, "Scheduling algorithms in a large-scale VOD server," *Proc. of IPPS'97 Workshop on Parallel Processing and Multimedia*, pp. 17–25, 1997.
- [11] A. L. N. Reddy, "Scheduling and data distribution in a multiprocessor video server," *Proc. of International Conference on Multimedia Computing and Systems '95*, pp. 256–263, 1995.
- [12] J. Cho and H. Shin, "Multi-resolution video layout for multimedia servers in mobile computing environment," *submitted to International Conference on Mobile Computing and Networking '98*, 1998.
- [13] M.-S. Chen, D. D. Kandlur, and P. S. Yu, "Support for fully interactive playout in a disk-array-based video server," *Proc. of ACM Multimedia '94*, pp. 391–398, 1994.
- [14] ISO/IEC 11172, "Information technology - coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s -"
- [15] 조진성, 김태현, 김영구, 성민영, 신현식, "마이크로커널 환경에서의 비디오 서버를 위한 디스크 배열 관리자," *한국정보과학회 봄 학술발표논문집*, pp. 335–338, 1996.
- [16] "QNX - System architecture," *QNX Operating Systems Manual*, QNX Software Systems Ltd., 1993.
- [17] "OptiVideo MPEG Compression," *MPEG-1 Encoder User's Guide*, OptiVision, Inc., 1995.
- [18] O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," *University of Wuerzburg, Institute of Computer Science Research Report Series*, Report No. 101, Feb., 1995. (<ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/>)