이질적인 분산 환경에서의 MPEG 비디오의 파싱을 위한 스케쥴링 알고리즘

(A Scheduling Algorithm for Parsing of MPEG Video on the Heterogeneous Distributed Environment)

남 윤 영 † 황 인 준 ††
(Yunyoung Nam) (Eenjun Hwang)

요 약 디지털 비디오의 사용이 보편화되면서 비디오에 대한 효율적인 브라우징이나 검색의 요구가 증가하게 되었다. 이러한 연산을 지원하기 위해서는 효과적인 비디오 인덱싱이 결합되어야 한다. 비디오 인덱싱에서 가장 기초적인 단계의 하나는 비디오를 샷과 장면으로 파싱하는 것이다. 일반적으로, 비디오 파싱은 복잡한 연산을 필요로 하기 때문에, 기존의 단일 컴퓨터 환경에서는 많은 시간이 소요된다. 기존의 연구는 일정한 시간 동안에 각 슬레이브들에게 작업을 할당하는 라운드 로빈 방식을 사용하였다. 그러나 이러한 방식은 이질적인 환경에서는 적용하는데 어려움이 있다. 본 논문에서는 이질적인 분산 컴퓨팅 환경에서 사용가능한 병렬 파싱 알고리즘인 사이즈 적응적인 라운드 로빈과 동적으로 사이즈 적응적인 라운드 로빈 방식을 제안하였다. 성능을 비교하기 위해 몇 가지 실험을 하였으며, 그 결과를 분석하였다.

키워드 : MPEG, 분산 컴퓨팅, 병렬 처리, 비디오 파싱

Abstract As the use of digital videos is getting popular, there is an increasing demand for efficient browsing and retrieval of video. To support such operations, effective video indexing should be incorporated. One of the most fundamental steps in video indexing is to parse video stream into shots and scenes. Generally, it takes long time to parse a video due to the huge amount of computation in a traditional single computing environment. Previous studies had widely used Round Robin scheduling which basically allocates tasks to each slave for a time interval of one quantum. This scheduling is difficult to adapt in a heterogeneous environment. In this paper, we propose two different parallel parsing algorithms which are Size-Adaptive Round Robin and Dynamic Size-Adaptive Round Robin for the heterogeneous distributed computing environments. In order to show their performance, we perform several experiments and show some of the results.

Key words : MPEG, distributed computing, parallel processing, video parsing

1. 서 론

지난 몇 년 동안 네트워크의 고속화와 멀티미디어 압축 기술의 발달로 인하여 디지털 미디어의 사용이 급격하게 증가하였고, 사용자는 더욱 다양한 검색을 요구하고 있다. 이러한 변화에 대응하기 위해서는 많은 양의 멀티미디어 데이타를 효율적으로 추출하고 조직화하는 것이 필요하다.

· 본 연구는 과학기술부 국책연구개발 사업인 유전자원지원 활용사업단의 연구비(no. BDM0100211)의 지원에 의해 수행되었습니다.
† 학생회원 : 아주대학교 정보통신전문대학원
youngman@ajou.ac.kr
†† 종신회원 : 고려대학교 전자컴퓨터공학과 교수
ehwang04@korea.ac.kr
논문접수 : 2003년 7월 21일
심사완료 : 2004년 8월 26일

특히, 디지털 비디오는 일반적인 데이타보다 많은 저장 공간과 네트워크의 대역폭을 필요로 한다. 이와 같은 요구에 따라 MPEG(Moving Pictures Expert Group) [1]은 디지털 비디오의 압축 표준(MPEG-1, MPEG-2, MPEG-4, MPEG-7)을 개발하였다. MPEG-1은 Video CD나 MP3에 사용되는 압축 표준이고 MPEG-2는 DVB/ATSC와 같은 디지털 TV에 사용되는 압축 표준이다. MPEG-4는 객체 기반으로 압축을 하며 주로 낮은 대역폭을 가지는 모바일 웹에 사용된다. MPEG-7은 Descriptor(D) 와 Descriptor Schemes(DS)에 의해 동화상을 기술하여 검색에 사용하는 데 사용된다. MPEG은 VOD(video-on-demand), MOD(movie-on-demand), DVD(digital video disc), HDTV(high definition television), 전자 도서관과 같은 멀티미디어 애플리케이션

에 가장 많이 사용되는 압축 표준이다.

비록, 디지털 비디오가 아날로그 비디오보다 압축과 관리면에서 쉽다는 장점이 있지만, 아직까지 대부분의 애플리케이션은 다양한 정보를 담고 있는 비디오를 효율적으로 추출하지 못하고 있다. 이러한 문제를 해결하기 위해, 기본적인 텍스트 검색뿐만 아니라 내용 기반 인덱싱이 필요하게 되었다. 즉, 멀티미디어 시스템에서 멀티미디어 데이타들은 메타데이타와 서로 연관되어 결합되어야 한다[2].

디지털 비디오의 경우, 인덱싱을 위해 샷(shot)과 객체(object)와 같은 요소로 분석하는 연구가 진행되었다. 이러한 비디오 세그멘테이션은 시간적 세그멘테이션과 공간적 세그멘테이션으로 나눌 수 있다. 시간적 세그멘테이션은 샷 경계, 디졸브(dissolve), 페이드(fade), 와이프(wipe)와 같이 일정한 시간내에서 영상의 변화를 검출하는 것을 말한다. 공간적 세그멘테이션은 각 프레임에서 객체의 추출이나 움직임을 파악하는 것을 말한다. 예를 들어, 얼굴 검출, 윤곽선 검출, 텍스트 검출 등이 이에 속한다.

이러한 구조화된 비디오는 효과적인 검색을 가능하게 하므로 비디오 파싱은 많은 디지털 비디오 애플리케이션에서 중요한 역할을 차지하고 있다. 그러나 비디오의 파싱은 복잡한 계산을 필요로 하기 때문에, 일반적인 단일 컴퓨터 환경에서의 많은 시간이 소요된다. 특히, DCT(discrete cosine transform), ME(motion estimation), MC(motion compensation)는 복잡한 계산을 하기 때문에 시간적인 비용이 크다. 예를 들어, 본 연구에 사용된 샷 경계 검출 알고리즘[3,4]을 사용하여 샷 경계를 검출한 결과, 320×240 픽셀인 3477개의 프레임들이 IBPBPBPBPBPP 패턴으로 인코딩된 MPEG-1 비디오를 SUN Sparc Ultra-60에서 샷 경계 검출을 하는데 1,086초가 소요된다. 즉, 초당 3.4 프레임의 속도로 샷을 검출한다.

실시간 인덱싱 처리는 멀티미디어 애플리케이션에서 필요하지만, 어떤 경우는 실시간 처리보다 더 빠른 성능을 요구한다. 예를 들어, 전자 도서관에 100만여개의 비디오 타이틀을 분석하고자 할 때, 한 개의 비디오 타이틀을 비디오 파싱하는 데 한 시간이 소요된다면, 실시간으로 모두 처리할 경우 100년 보다 더 많은 시간이 소요된다. 하지만, 실시간 처리보다 더 빠르게 처리할 수 있다면 단지 몇 달 만에 모두 처리할 수 있다[5].

MPEG의 특성상 디코딩보다는 인코딩에 더 많은 시간이 많이 소요된다. 빠른 MPEG의 인코딩을 위해 하드웨어를 사용할 수도 있지만, 하드웨어의 비용이 큰 문제점이 된다. 이러한 문제를 해결하기 위해 소프트웨어 인코딩에 대한 연구가 진행되어 왔으며, 그 중 하나가 병렬처리를 이용하는 방식이다[6-9]. 또 다른 연구에서는, 고해상도로 압축된 MPEG-2을 실시간으로 디코딩하는 데, 병렬 처리 방식을 사용하였다[10]. 최근에는, 비디오의 인덱싱[3]에 대한 관심이 높아지면서 많은 양의 비디오를 단시간 내에 분석할 수 있는 연구가 진행되었으며, 멀티프로세서 환경에서 병렬 파싱을 하는 연구[11,12]가 있었지만, 멀티프로세서의 비용이 고가라는 문제점을 가진다. 이러한 문제를 해결하기 위해, 본 연구는 근거리로 연결된 여러 유휴 컴퓨터들의 자원을 이용하여 단시간 내에 비디오를 파싱하고자 한다. 특히, 프로세싱 시간을 줄이기 위해, 이질 분산 환경에서 MPEG 비디오의 병렬 파싱을 위한 스케쥴링 알고리즘을 제안한다. 또한, 성능 분석을 위해 기존의 방법인 라운드 로빈 방법과 개선된 두 가지 스케쥴링 알고리즘을 비교 분석하고 그 결과를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 MPEG의 개요와 병렬 비디오 파싱에 대해서 알아보고, 3장에서는 병렬 비디오 파싱 모델에 대해 설명한다. 4장에서는 몇 가지 스케쥴링 방법에 대해서 설명하고, 5장에서는 실험 결과를 보인다. 마지막으로 6장에서는 결론과 향후 계획에 대해서 논의한다.

2. 관련 연구

2.1 MPEG 개요

MPEG 비디오는 일련의 GOP(Group of picture)로 구성되며, GOP는 프레임들로 구성된다. 프레임은 프레임에서의 행으로 구성된 부분인 슬라이스(slice)로 나누어 지며, 슬라이스는 6개의 매크로블록(macroblock)으로 구성된다. MPEG 비디오는 시간적인 블록 기반의 모션 보정과 공간적인 DCT 기반의 압축기법에 의해 압축이 된다. 움직임 정보는 16×16 픽셀 블록을 이용하여 계산되고 공간적 정보와 함께 전달된다. 6개의 블록을 포함하는 매크로 블록은 밝기(Y)를 위한 4개의 8×8 픽셀 블록과 색채(U와 V)를 위한 2개의 8×8 픽셀 블록으로 구성된다.

MPEG은 I-프레임(intra-coded frame), P-프레임(predictive-coded frame), B-프레임(bidirectionally predictive coded frame)으로 나누어 진다. I-프레임은 다른 프레임과 관계없이 인코딩되고 프레임내에서 공간적 상관성을 이용하여 만들어진다. P-프레임은 이전 I-프레임 또는 P-프레임으로부터의 움직임 보상을 이용하여 인코딩되며, B-프레임은 움직임 보상을 위해 이전 또는 이후 프레임 (I-프레임 또는 P-프레임)이 필요하다. 일반적으로 I-프레임은 P-프레임과 B-프레임보다 높은 비트율을 지니고 있는데, I-프레임은 픽셀당 1bit, P-프레임은 픽셀당 0.1bit, B-프레임은 픽셀당 0.015bit

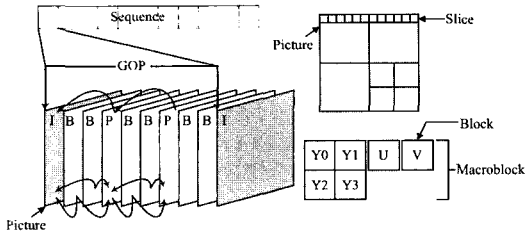


그림 1 I-프레임, P-프레임, and B-프레임로 구성된 MPEG 비디오의 예

의 비트율 가지고 있다.

MPEG 비디오의 프레임은 I, P, B-프레임으로 압축이 된다. I, P, B-프레임의 패턴은 비디오 안에서 반복되는데, 예를 들어 IBBPBBPBBI의 프레임이 하나의 패턴으로 반복된다. 즉, MPEG 비디오는 일련의 GOP로 구성되어지는데, GOP는 적어도 하나의 I-프레임이 포함되어야 하며, 첫 번째 프레임은 I-프레임 또는 B-프레임이고 마지막 프레임은 I-프레임 또는 P-프레임이어야 한다. 따라서 GOP는 독립적으로 디코딩이 가능하다. 본 논문에서는 시작 프레임과 끝 프레임이 모두 I 프레임인 것을 GOP로 정의하여 사용하였다.

2.2 비디오 파싱

비디오 파싱이란 샷/장면 변화, 특수 효과, 카메라 움직임을 검출하여 세그멘테이션을 하는 과정을 말한다. 비디오 세그멘테이션은 시간적인 세그멘테이션과 공간적인 세그멘테이션으로 나눌 수 있다.

시간적인 세그멘테이션은 하나의 샷이 다른 샷으로 변하는 것을 말한다. 이러한 변화는 두 프레임에서 이루어질 경우, 이것을 컷(cut)이라고 한다[13]. 어떤 경우 점진적으로 샷이 변하는 경우가 있는데, 이러한 변화는 페이드 또는 와이프, 그 외의 특수 효과에서 볼 수 있다. 이러한 샷은 팬(pan)과 줌(zoom)과 같은 카메라 움직임에 따라 재분류할 수도 있다.

매크로블록을 이용하여 세크멘테이션을 하는 방법은 I, P, B-프레임으로 구성된 어떠한 MPEG에 대해서도 적용할 수 있으며, 또한 단순히 DCT 정보로만으로도 샷의 변화를 검출할 수도 있다.

2.3 병렬 비디오 파싱

MPEG 비디오의 병렬 파싱은 위에서 설명한 슬라이스, 프레임, GOP 단위로 할 수 있다[5,9,10-12,14,15].

- 슬라이스 단위 : 슬라이스를 작업 큐에 넣은 후 작업 프로세스가 슬라이스를 디코딩하여 DC값의 합을 계산한다. 모든 프로세스는 같은 프레임을 동시에 수행하며 서로 통신을 통해 계산값을 조정해야 한다.
- 프레임 단위 : 작업 프로세스가 인접한 프레임을 디코딩하며 디코딩하는 프레임중에 만약 I-프레임 또는 P-프레임이 없다면 다른 프로세스와 통신을 해야 한다.
- GOP 단위 : GOP는 독립적으로 디코딩을 할 수 있기 때문에 다른 프로세스와 통신할 필요가 없다. 따라서 결과를 더 빠르고 쉽게 얻을 수 있다.

GOP 단위의 병렬 비디오 파싱은 다른 두 가지의 방법보다 통신면에서의 낮은 오버헤드 때문에 속도 향상(speedup)측면에서 좋은 결과를 얻을 수 있다 [9,10-12,14,15]. 또한, MPEG 비디오의 압축에서도 Berkeley의 인코더보다 GOP 단위의 encoder가 성능이 뛰어나다고 연구되었다[9]. 본 논문에서도 속도 향상을 위해 GOP 단위로 파싱을 하며, 단일 프로세서에서 동작하는 샷 검출 알고리즘을 개선하여 멀티 프로세서에서도 가능하도록 수정하였다[3,4].

3. 병렬 비디오 파싱 모델

본 연구에서는 병렬 비디오 파싱을 위해 전형적인 마스터-슬레이브 모델을 이용한다. 그림 2에서 보는 바와 같이 시스템은 네트워크 스토리지, 마스터, 슬레이브로 구성된다. 네트워크 스토리지는 디지털 비디오를 저장할 뿐만 아니라 데이타를 구성하고 관리하는 기능을 한다. 마스터는 작업과 슬레이브를 관리하는 스케쥴러에 의해 동작하며, 스케쥴러는 다음의 세 가지 큐를 가지고 있다. ① 입력 큐 : 슬레이브에게 분배할 작업을 저장한다. ② 출력 큐 : 슬레이브가 파싱한 결과를 저장한다. ③ 슬레이브 큐 : 슬레이브에 대한 정보를 저장한다.

마스터는 파싱할 작업을 준비하고 슬레이브에게 분배하여 파싱한 결과를 재결합하는 역할을 담당하며, 슬레이브는 마스터에게 작업을 요청하여 파싱하고 결과를 마스터에게 보내주는 역할을 한다. 마스터는 스캔 프로세스를 통해 디스크에서 비디오 파일을 읽은 후 프레임 사이즈, GOP수, 평균 비트율 같은 정보를 수집 후 입력 큐에 작업 단위로 쪼개어(split) 입력 큐에 저장하는 역할을 한다. 슬레이브는 입력 큐에 저장된 작업을 파싱 알고리즘을 이용하여 세그멘테이션을 수행하고, 세그멘테이션의 결과는 마스터의 출력 큐에 저장된다. 그러나, 슬레이브의 결과를 반환하는 순서가 다르기 때문에 출력 큐의 저장된 파싱 결과의 순서는 입력 큐의 데이타의 순서와는 다르다. 따라서 순서를 재결합시키는 과정이 필요하며 이러한 과정은 결합(combine) 프로세스를 통해 이루어 진다.

파싱하는 과정에서 여러 가지 요인에 의해 슬레이브가 파싱한 결과를 보내지 않는 상황이 발생하는 경우가 있다. 이러한 경우, 파싱하는 시간이 지연되어 총 소요시간이 증가하거나, 최악의 경우 파싱이 중지(hang-up)될 수 있다. 이러한 상황을 방지하도록 타임-아웃 메카니즘을 사용하여 일정 시간동안 슬레이브가 결과를 보

내지 않으면 다른 슬레이브에게 작업을 할당시켜 전체적으로 파싱하는 작업이 동적으로 이루어 질 수 있도록 하였다.

그림 2는 시스템의 전체적인 구조를 보여주고 있다.

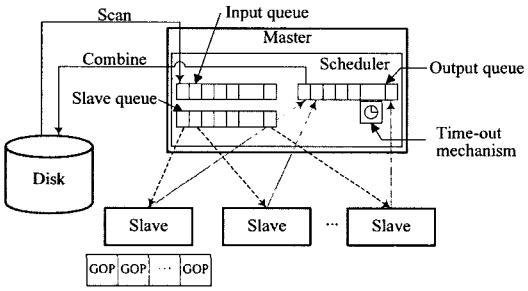


그림 2 시스템의 구조

4. 스케쥴링 알고리즘

본 장에서는 효율적인 병렬 파싱을 위해 가장 간단한 방법인 라운드 로빈(Round Robin)을 소개하고, 본 논문에서는 사이즈 적응적인 라운드 로빈(Size-Adaptive Round Robin)과 동적으로 사이즈 적응적인 라운드 로빈(Dynamic Size-Adaptive Round Robin)을 제안한다. 그리고 제안한 스케쥴링 알고리즘에서 마스터와 슬레이브가 서로 통신하여 작업을 분배하는 방식에 대해서 좀 더 자세히 설명한다.

4.1 라운드 로빈(Round Robin)

라운드 로빈 스케쥴링은 그림 3과 같이 입력 큐에 있는 데이타를 동일한 크기로 나누어 분석하는 방식이다. 3장에서 설명했듯이, 이용 가능한 슬레이브들에 대한 정보를 슬레이브 큐에 저장하고 있으면 스케쥴러가 입력 큐에 있는 작업을 슬레이브 큐에 저장되어 있는 슬레이브들에게 작업을 할당한다. 이 때, 파싱과 인터럽트(interrupt)에 필요한 타이머를 설정한다. 예를 들어, 그림 3처럼 하나의 마스터(M)와 세 개의 슬레이브(S_1, S_2, S_3)가 있다면, 우선 큐에 있는 순서대로 S_1, S_2, S_3에게 작업을 분배한다. 슬레이브들의 성능에 따라서 결과를 반환하는 시간이 달라지는데, 그림에서 보는 바와 같이 S_3이 시간 $T+i(i>0)$에 먼저 끝났기 때문에 M에게 먼저 결과를 반환하고 나서, S_2와 S_1이 순서대로 결과를 M에게 보낸다. 즉, S_3이 M과 통신이 끝나면 슬레이브 큐에 있는 S_2와 S_1이 차례대로 M과 통신을 한다.

이 방식은 하나의 슬레이브가 다른 슬레이브들보다 작업 처리 시간 또는 마스터와 통신하는 시간이 많아지면 마스터 또는 다른 슬레이브들이 휴지(idle)상태로 남아있게 되거나 파싱하는데 소요되는 전체적인 시간이 증가하게 된다. 이러한 문제를 해결하기 위해서 개선된 라운드 로빈 스케쥴링이 필요하다.

그림 4는 라운드 로빈 스케쥴링의 흐름도이다. 우선, Scanning_Input_Stream() 함수를 호출하여 MPEG 스트림의 프로파일(profile)을 생성한 후, 미리 정한 같은 작업의 크기 x에 따라 split() 함수를 통해 MPEG 비디오 스트림을 나누어 입력큐에 저장한다. 이러한 처리과정은 $\lceil \frac{GOP}{x} \rceil$ 만큼 반복하며 split()과 enqueue() 과정이 마치면 작업을 슬레이브들에게 분배한다.

4.2 사이즈 적응적인 라운드 로빈(Size-Adaptive Round Robin)

슬레이브는 시스템의 성능에 따라 계산시간이 서로 다르며, 네트워크 환경의 전송 속도는 항상 변하기 때문에 슬레이브들은 서로 다른 네트워크 대역폭을 가지게 된다. 따라서 각 슬레이브의 효율적인 스케쥴링을 위해서 계산시간과 통신시간이 고려되어야 한다. 만약 마스터가 이러한 환경요인을 고려하지 않고 고정된 크기의 작업을 슬레이브들에게 분배를 하는 라운드 로빈 스케

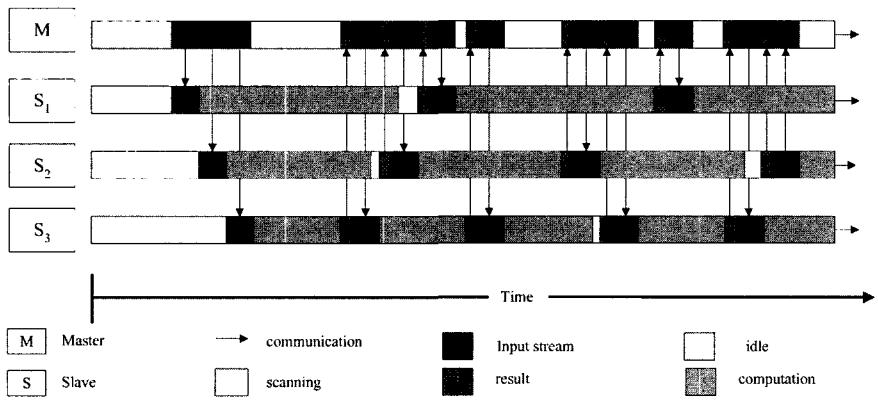


그림 3 라운드 로빈 스케쥴링

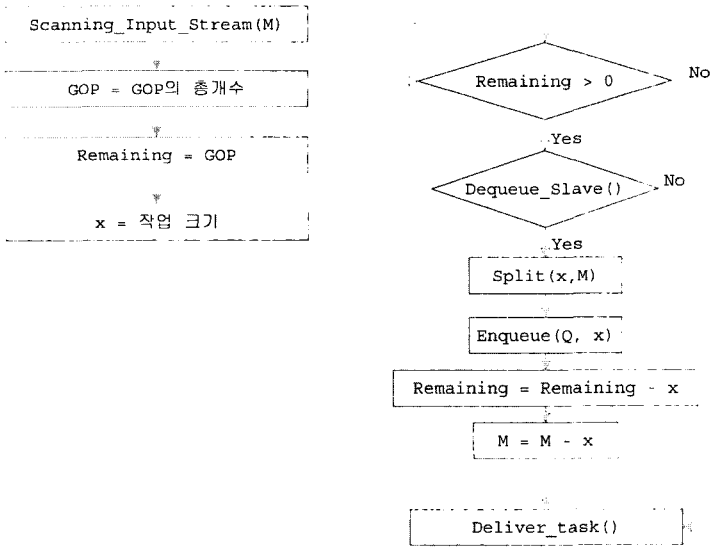


그림 4 라운드 로빈 스케줄링의 흐름도

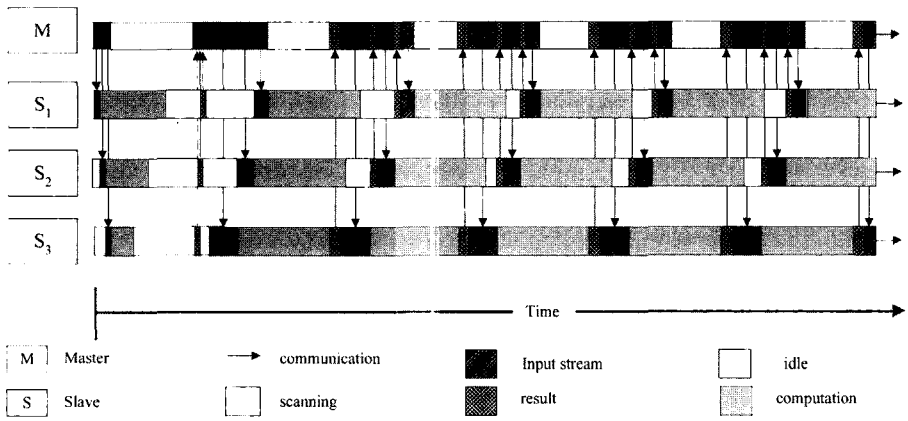


그림 5 사이즈 적응적인 라운드 로빈 스케줄링

줄링을 사용하면, 전체적으로 계산 성능이 낮아지게 된다. 따라서 슬레이브의 계산성능과 네트워크의 대역폭에 따라 작업크기를 조절하여 분배하는 사이즈 적응적인 라운드 로빈 스케줄링이 필요하다.

그림 5와 그림 6은 사이즈 적응적인 라운드 로빈 스케줄링과 흐름도를 보여주고 있다. 마스터는 같은 크기의 샘플파일(SMP)을 각 슬레이브들에게 보내어 Sampling_Slave_Workload() 함수를 호출하여 슬레이브들의 계산 성능과 네트워크 대역폭을 알아낸 후, 마스터는 각 슬레이브들에게 전송할 비디오 스트림의 크기를 조절하여 split()를 실행한다. 나눠진 비디오 스트림은 높은 성능을 지닌 슬레이브 순서로 분배된다. 그림 5에서는 성능이 높은 순서가 $S_3>S_2>S_1$이기 때문에 S_3, S_2, S_1 순서로 분배된다. 이러한 사이즈 적응적인 라운드 로빈 스케줄링은 라운드 로빈 스케줄링보다 효율적이지만, 여전히 마스터와 슬레이브가 데이타 부족으로 인해 휴지상태에 놓일 수 있다. 이런 경우 로드 밸런싱(load balancing)이나 대기하는 동안 다른 작업을 수행하는 방법을 통해 조절할 수 있다. 본 논문에서는 이를 위해 동적으로 사이즈 적응적인 라운드 로빈 스케줄링을 제안한다.

4.3 동적으로 사이즈 적응적인 라운드 로빈(Dynamic Size-Adaptive Round Robin)

위에서 두 가지 스케줄링 방법은 슬레이브의 수가 변하지 않는 환경에서는 유용할 수 있으나 클라이언트의 수가 가변적인 환경에서는 적합하지가 않다. 또한, 위에서 언급했던 마스터와 슬레이브의 휴지상태를 해결하기

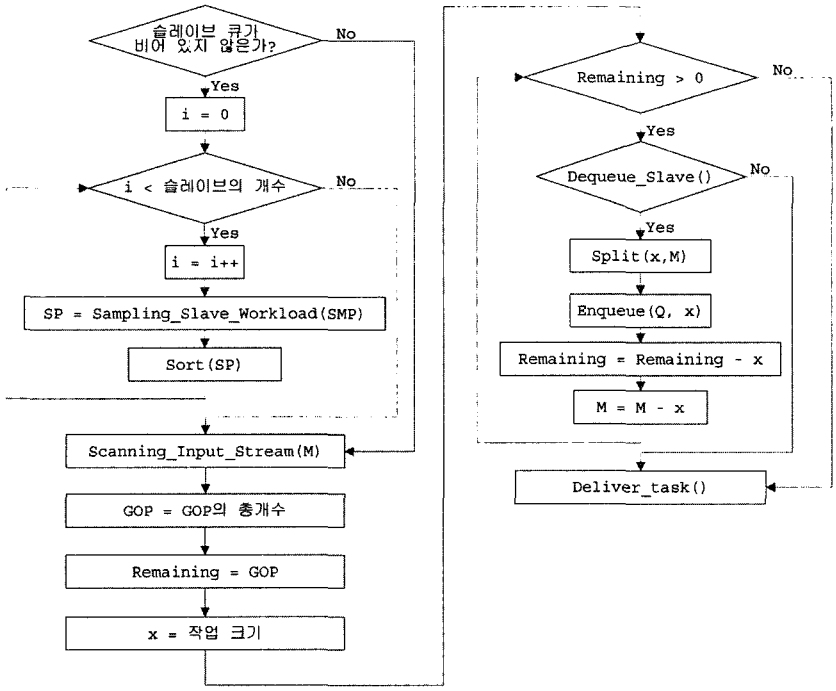


그림 6 사이즈 적응적인 라운드 로빈 스케쥴링의 흐름도

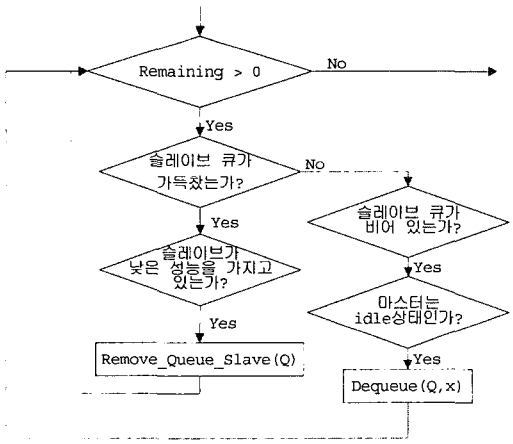


그림 7 동적으로 사이즈 적응적인 라운드 로빈 스케쥴링에서의 로드 밸런싱

위해서 논 블로킹(non-blocking) 전략이 고려되어야만 한다. 마스터의 경우, 슬레이브의 수를 충분히 증가시켜 마스터의 휴지 상태를 해결할 수 있지만, 최적의 개수에서 초과되는 경우 많은 수의 슬레이브가 휴지상태가 되어 자원 낭비를 초래하게 된다. 따라서, 슬레이브의 계산성능과 대역폭을 예상하여 작업의 크기와 슬레이브의 수를 계속 조정함으로써 휴지시간을 줄일 수 있으며 자원을 효율적으로 사용할 수 있다. 슬레이브의 수가 최적의 개수보다 많아지면 마스터의 오버헤드가 발생되고 슬레이브가 휴지상태로 되므로 낮은 성능의 슬레이브들은 작업에서 배제시킨다. 이러한 계속적인 슬레이브의 수의 증가/감소를 통해 최적의 개수로 유지시키는 스케쥴링 알고리즘인 동적으로 라운드 로빈 스케쥴링 방법을 사용하면 휴지상태를 줄일 수 있으며, 슬레이브들의 자원들을 활용하는데 상당한 효과를 얻을 수 있다. 그림 7은 이러한 DSARR 스케쥴링에서 로드 밸런싱의 흐름도를 보여주고 있다.

5. 실험결과

실험을 위하여 SUN Sparc Ultra-60과 이더넷(ethernet)에 연결된 10개의 슬레이브 머신을 사용하여 수행하였으며 사용된 파리미터는 표 1과 같다.

제안한 스케쥴링의 성능을 측정하기 위해서 다섯 가지의 비디오 데이타를 가지고 실험하였으며, 비디오는 MPEG-1, MPEG-4로 압축되어 있다. 빠른 파일 입출

표 1 실험에 사용한 파라미터

파라미터	값
Master CPU	1.8GHz
Slave CPU	0.8GHz ~ 1.8GHz
데이타 전송율	40MB/sec ~ 54MB/sec
네트워크 처리량	10M ~ 100M bps

표 2 Test MPEG video sequences characteristics

Sequence	Format	Frames	Resolution (pixels)	Keys/GOPs	File size (Kbyte)
News1	MPEG 1	3477	320 × 240	298	19,730
News2	MPEG 1	5158	320 × 240	308	29,296
Music Video	MPEG 4	6413	640 × 360	54	68,898
Sports	MPEG 4	11756	640 × 480	131	126,996
Movie	MPEG 4	195934	640 × 480	3568	737,165

력을 위해 RAID(redundant array of inexpensive disks) 스토리지에 저장하였다. 표 2는 실험에 사용한 비디오 스트림의 특성을 보여주고 있다.

표 2에서, News1은 IBPBPBPBPBPP의 프레임 패턴으로 하나의 GOP에는 12개의 프레임으로 이루어져 있다. News2는 IBBPBBPBBPBBPBBPBB의 프레임 패턴으로 하나의 GOP에는 18개의 프레임으로 구성되어 있다. 다른 세 가지 비디오 스트림은 하나의 GOP에 각각 120, 90, 72개의 프레임으로 구성되어 있다.

일반적으로 멀티컴퓨터 환경은 데이타 표현과 메시지 교환 프로토콜을 통해 이루어지는데, 본 시스템에서는 데이타 표현은 XML(Extensible Markup Language)[16]를, 메시지 교환 프로토콜로는 SOAP(Simple Object Access Protocol)[16]을 사용하였다.

실험에서 전체적인 파싱에 소요되는 시간은 첫 번째 슬레이브가 시작해서 마지막 슬레이브가 끝나는 시간으로 정했으며, 성능 분석을 위해서 그림 3과 5에서 보았던 것과 같이 i 번째 슬레이브가 계산하는 데 걸리는 시간은 T^i_{comp}, 통신하는 데 걸리는 시간은 T^i_{comm}, 휴지상태의 시간은 T^i_{idle}으로 정의하며, 전체적인 실행시간 T는 다음과 같이 정의한다.

(i) 슬레이브 j 에서의 계산, 통신, 휴지시간의 합이다.

$$T = T^j_{comp} + T^j_{comm} + T^j_{idle} \tag{1}$$

(ii) 모든 슬레이브들의 전체 시간의 합을 슬레이브 P의 개수의 의해 나눈 값이다.

$$T = \frac{1}{P}(T_{comp} + T_{comm} + T_{idle}) = \frac{1}{P}(\sum_{i=0}^{P-1} T^i_{comp} + \sum_{i=0}^{P-1} T^i_{comm} + \sum_{i=0}^{P-1} T^i_{idle}) \tag{2}$$

그림 8은 News2를 라운드 로빈 스케쥴링을 사용하여 샷을 검출하는데 소요된 계산, 통신, 휴지 시간이다. 단일 머신에서는 파싱하는데 221.2초가 소요되었으나, 8개의 머신에서는 30.5초가 소요되었다. 즉, 슬레이브의 수가 증가함에 따라, 파싱하는 데 소요되는 시간이 감소하였으며 슬레이브의 수가 8개일 때 가장 최소가 되었으며 7.1배의 성능 향상을 얻었다. 6개의 머신에서 5개의 머신보다 더 많은 시간이 소요되었는데, 이것은 특정 슬

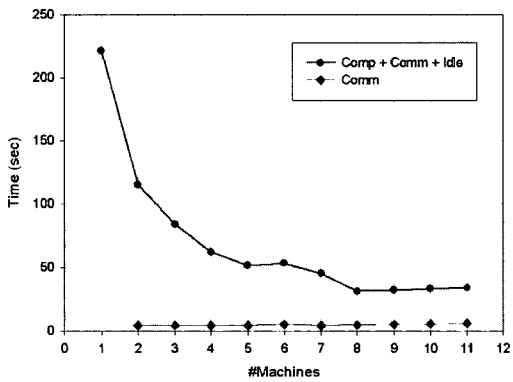


그림 8 Computation, Communication, and Idle time of slaves vs. #machine

레이브가 휴지상태가 되었거나 시간이 초과되어 할당했던 작업을 다른 슬레이브에게 분배했기 때문이다. 머신이 8개 이상일 때 그래프가 증가하지 않은 이유는 초과분의 슬레이브들은 휴지상태가 되었기 때문이다. 따라서 이 경우, 8이 가장 최적합의 슬레이브 개수이다.

스케쥴링 알고리즘을 평가하기 위해서 speedup과 efficiency를 사용한다.

$$speedup = \frac{time\ for\ sequential\ parsing}{time\ for\ parallel\ parsing} \tag{3}$$

$$efficiency = \frac{speedup\ for\ N\ machines}{N} \tag{4}$$

그림 9는 라운드 로빈 스케쥴링을 사용했을 때, 슬레이브 수에 따른 speedup의 변화를 보여주고 있다. 그림에서 보듯이 News2에서는 하나의 머신보다 8개의 머신일 때 7.1배 더 빠른 성능을 보였으며 뮤직 비디오에서는 10개의 머신일 때 7.3배 더 빠른 결과를 보였다. 병렬 비디오 파싱이 모든 경우에서 7배 이상의 빠른 성능을 보였다.

그림 10은 MPEG-4로 압축된 영화에 대해서 라운드 로빈(RR) 스케쥴링과 사이즈 적응적인 라운드 로빈(SARR) 스케쥴링을 사용하였을 때 슬레이브 수에 따른 speedup과 efficiency의 변화를 보여주고 있다. 그림에서 보듯이, SARR은 RR보다 좋은 성능을 나타냈으며, 그림 10(a)에서, speedup은 머신의 수에 따라 선형으로

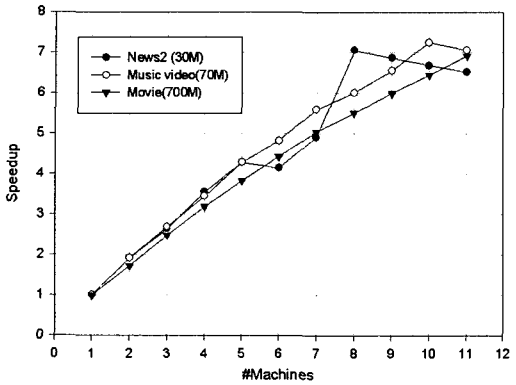


그림 9 Speedup vs. #machine

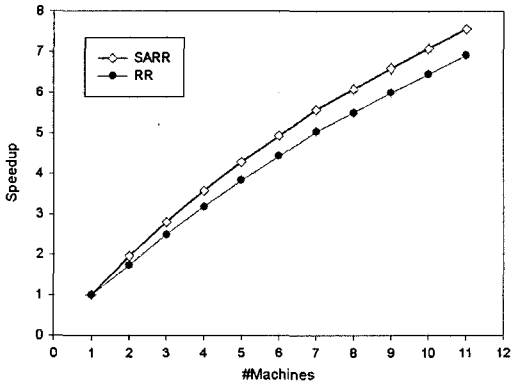
증가하였고, 머신의 수가 11개일 때 RR은 7.0, SARR은 7.6의 speedup이였다. 즉, SARR이 RR보다 8.57% 더 좋은 성능을 얻을 수 있었다. 또한, 그림 10(b)에서 SARR이 RR보다 더 효율적이라는 것을 알 수 있었다. 그러나, 그림 9에서 보았듯이 머신의 개수가 최적일 때보다 많을 경우 더 이상의 개선효과는 얻을 수 없을 것이다. 그 이유는 그림 9에서 보듯이, 머신의 수가 초과되면 통신시간과 휴지시간이 증가를 일으키기 때문이다.

그림 11은 동적으로 사이즈 적응적인 라운드 로빈 스케쥴링을 사용했을 때, 시간에 따른 슬레이브 수의 변화를 보여주고 있다. 만약 머신이 낮은 성능과 네트워크 대역폭을 가지고 있다면 전체적인 작업시간을 증가시키게 되는 원인이 되므로 따라서 마스터는 낮은 성능의 슬레이브는 배제시키고 좋은 성능의 슬레이브를 유지시켜야 한다.

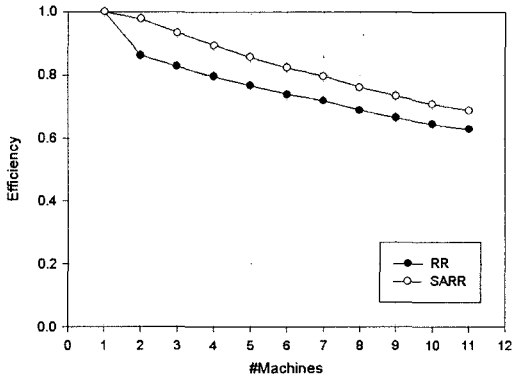
6. 결론 및 향후 계획

본 논문에서는 이질 분산 환경에서 MPEG 비디오의 병렬 파싱을 위한 스케쥴링을 제안하였다. 속도 향상을 위해 GOP 단위로 파싱하였으며, 기존의 라운드 로빈 방식과 비교하여 개선된 두 가지 스케쥴링 알고리즘을 제안하고 성능을 비교 분석하였다.

실험 결과에서 슬레이브의 수가 고정되어 있을 때에는 사이즈 적응적인 라운드 로빈 스케쥴링이 좋은 성능을 보였다. 11개의 슬레이브를 사용하였을 때가 단일 컴퓨터보다 7.6배의 속도 향상을 보였으며, 사이즈 적응적인 라운드 로빈 스케쥴링이 라운드 로빈 스케쥴링보다



(a) Speedup vs. #machine



(b) Efficiency vs. #machine

그림 10 Speedup and Efficiency vs. #machine

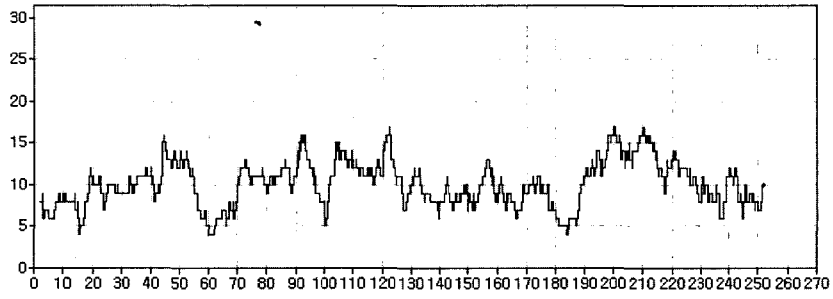


그림 11 #machines vs. Time (available slave machines are 20)

8.57% 더 좋은 성능을 얻을 수 있었다. 동적으로 사이즈 적응적인 라운드 로빈 스케줄링은 네트워크 환경이 신뢰할 수 없고 변화가 심한 분산 네트워크 환경에서 적합한 스케줄링 방법이다. 본 논문에서 제안한 MPEG 비디오의 병렬 파싱은 이질 분산 컴퓨팅 환경에서 이식 가능하고 유연하게 사용될 수 있다. 향후 계획은 좀 더 효율적이고 최적화된 병렬 스케줄링 방법을 개발하는 것이다.

참 고 문 헌

[1] The MPEG home page, http://mpeg.telecomitalialab.com/

[2] Rao, K.R. et al.:Multimedia Communication Systems, Prentice Hall (2002).

[3] Rainer Lienhart. Comparison of Automatic Shot Boundary Detection Algorithms. Storage and Retrieval for Still Image and Video Databases VII 1999, Proc. SPIE 3656-29,

[4] Movie Content Analysis Project (MoCA) http://www.informatik.uni-mannheim.de/informatik/pi4/projects/MoCA

[5] Sun, X.H., Rover, D.T.: Scalability of parallel algorithm-machine combinations, Parallel and Distributed Systems, IEEE Transactions, Vol. 5(6). (1994) 599-613.

[6] Kevin L.G. and Lawrence A.R.:Parallel MPEG-1 Video Encoding. the Picture Coding Symposium (1994).

[7] Moore, J. et al.: Optimal parallel MPEG encoding. Department of Computer Science 4130 Upson Hall, Cornell University.

[8] Shen, K., Rowe, L. A. and Delp, E. J.: A Parallel Implementation of an MPEG1 Encoder: Faster than Real-Time!, the SPIE Conference on Digital Video Compression (1995).

[9] Sandor Bozoki, et al.: Parallel algorithms for MPEG video compression with PVM. EUROSIM (1996) 315-326.

[10] Bilas, A., Fritts, J. and Singh, J. P.: Real time parallel MPEG-2 decoding in software. 11th International Parallel Processing Symposium (1997).

[11] Suchendra M. Bhandarkar, Shankar R. Chandrasekaran: Parallel Parsing of MPEG Video in a Multi-threaded Multiprocessor Environment. IPDPS Workshops (2000) 194-201.

[12] Suchendra M., et al.:Parallel Parsing of MPEG Video. International Conference on Parallel Processing (2001).

[13] Kobla, V. and Doermann, D.:Indexing and Retrieval of MPEG Compressed Video. Journal of Electronic Imaging, Vol. 7(2) (1998) 294-307.

[14] Sun, X.H. and Ni, L.:Scalable Problems and Memory-Bounded Speedup, Journal of Parallel and Distributed Computing, Vol. 19. (1993) 27-37.

[15] Foster, I.: Designing and Building Parallel Programs, Addison-Wesley (1995).

[16] World Wide Web Consortium (W3C) home page, http://www.w3.org/

[17] Grama, A., Gupta, A. and Kumar, V.:Isoefficiency Function: A Scalability Metric for Parallel Algorithms and Architectures. IEEE Parallel & Distributed Technology, Vol.1(9). (1993) 12-21.

[18] Hwang K.: Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw-Hill (1993).

[19] Lin, Y.D., et al.: A hierarchical network storage architecture for video-on-demand services, 21st Conference on Local Computer Networks (1996).

[20] Y. Nam and E. Hwang.:Parallel Parsing of MPEG Video in Heterogeneous Distributed Environment, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2720, (2003) 264-274.

남 윤 영

2001년 아주대학교 정보 및 컴퓨터공학과 학사. 2001년~2003년 아주대학교 정보통신전문대학원 석사. 2003년~현재 아주대학교 정보통신전문대학원 박사과정 관심분야는 데이타베이스, 멀티미디어 시스템, 정보 통합, XML 응용. 병렬처리, 분산컴퓨팅

황 인 준

1988년 서울대학교 컴퓨터공학과(학사). 1990년 서울대학교 컴퓨터공학과(석사). 1998년 Univ. of Maryland at College Park 전산학과(박사). 1998년~1999년 Bowie State Univ., Assistant Professor. 1999년~1999년 Hughes Research Lab. 연구교수. 1999년~2003년 2월 아주대학교 정보통신전문대학원 조교수. 2003년 3월~2004년 8월 아주대학교 정보통신전문대학원 부교수. 2004년 9월~현재 고려대학교 전자컴퓨터공학과 조교수. 관심분야는 데이타베이스, 멀티미디어 시스템, 정보 통합, 전자상거래, XML 응용, 병렬처리, 분산컴퓨팅