공학석사학위논문

클러스터 시스템에서 효과적인 미디어 트랜스코딩 부하 분산 정책

최 면 욱

강원대학교대학원

컴퓨터정보통신공학과

2004년 5월

정 인 범 교 수 지 도

공학석사학위논문

클러스터 시스템에서 효과적인 미디어 트랜스코딩 부하 분산 정책

Effective Media Transcoding Load Distribution Policy in the Cluster System.

강원대학교대학원

컴퓨터정보통신공학과

최 면 욱

최면욱의 석사 학위논문을
합격으로 판정함

2004년 5월

심사위원장 정 인 범 (인)

위 원 이 헌 길 (인)
위 원 김 윤 (인)

클러스터 시스템에서 효과적인 미디어 트랜스코딩 부하 분산 정책

최 면 욱

강원대학교 대학원 컴퓨터정보통신공학과

최근 무선통신 기술의 발전으로 PC뿐만 아니라 PDA, 휴대폰 등 다양한 장치를 통하여 멀티미디어 서비스를 제공 받을 수 있게 되었다. 무선망은 유선망에 비해 대역폭이 상대적으로 매우 낮다. 따라서 높은 비트율을 갖는 스트림을 낮은 비트율로 변환하기 위해 트랜스코딩 기술을 이용한다. 이러한 트랜스코딩시스템에서 효율적으로 스트리밍을 전송하기 위한 효과적인 부하 분산에 관한 연구가 필요하다.

시스템 구현 환경에서는 트랜스코딩시스템 부하분산 연구를 위하여 공개 소스기반인 리눅스를 이용하여 클러스터 트랜스코딩서버와 분배서버로 구성된 시스템을 구현하였다. 트랜스코딩서버에서는 MPEG 프로화일 등급에 따른 트랜스코딩과 시스템자원정보를 분배서버에 보내주는 기능을 구현하였고 분배서버에서는 본 논문에서 제안한 부하분배 알고리즘을 이

용하여 부하를 분산하도록 구현하였다.

본 논문에서 실험은 사용자들의 다양한 영화선호도를 기반으로 한 부하분산 알고리즘을 평가하였다. 부하분산 알고리즘은 기존의 클러스터 웹 서버의 부하분산 알고리즘인 RR(Round Robin) 방식과 WRR(Weight Round Robin)을 응용하여 동적으로 시스템 가중치가 바뀌는 DWRR(Dynamic Weight Round Robin)을 구현하였다. 또한 스트리밍 서비스의 모바일 호스트 등급에 따른 CPU 사용량, 네트워크 대역폭, 메모리의 가중치를 계산하여 부하분산에 사용할 수 있는 RWRR(Resource Weight Round Robin) 알고리즘을 구현하고 평가하였다. 제안된 방법은 기존의 부하분산 알고리즘과 비교하여 효과적인 부하분산이 된다는 것을 실험을 통하여 확인하였다.

감사의 글

석사 논문을 정리하면서 지난 2년간의 대학원 생활이 참으로 빨리 지나간 것 같다는 생각이 들었었습니다. 너무나 빨리 지나간 2년이라는 시간동안 저는 대학원 수업과 연구실 프로젝트를 통해 많은 지식과 경험을 쌓을 수 있었습니다. 하지만, 한편으로는 대학원 시절의 시간을 좀 더 충실하게 사용했으면 하는 아쉬움도 많이 남았습니다. 지금까지의 연구실 생활과 석사 논문을 무사히 마칠 수 있게 도와주셨던 고마운 분들께 이글을 빌어 짧게나마 감사의 마음을 전합니다. 먼저 대학원 생활동안 부족한 저를 이끌어 주시고 논문을 완성하기까지 하나하나 살펴보아 주신 정인범 교수님께 마음 깊이 감사드립니다. 또한 논문에 대해 많은 조언을 주시고 심사를 맡아주셨던 이헌길 교수님과 김윤 교수님께도 진심으로 감사드립니다. 그리고 같은 연구실에 있으면서 많은 도움을 받은 대학원의 이좌형, 방철석, 김병길 님에게 감사하며, 1년 동안 연구실에서 같이 생활하였던 학부 4학년 박총명, 김래영, 문동규 후배님들에게도 모두 감사드립니다.

그리고 저를 낳아주시고 물신 양면으로 뒷바라지 해 주신 어머님께 감사드립니다. 지금까지 절 믿고 지켜봐주신 어머님이 없었다면 저는 지금 이 자리에 있지 못했을 것입니다.

마지막으로 저의 모든 고마운 분들께 보답한다는 생각으로 앞으로 더욱 노력을 하여 사회에 조금이나마 보탬이 되는 사람이 되도록 하겠습니다. 감사합니다.

목 차

표 목 차

그림목차

Ⅰ. 서 론

최근 멀티미디어와 정보통신망의 발전에 따라 영상 정보 서비스에 대한 요구가 날로 다양해지고 있다. 멀티미디어 서비스의 급속한 발전으로 사용자는 유선뿐만 아니라 무선망을 통하여 무선단말기로 멀티미디어 스트림을 전송하고 재생하는 스트리밍 서비스를 받을 수 있게 되었다. 멀티미디어 스트림의 디지털 정보의 영상 정보의 양이 다른 일반 데이터의 정보량에 비하여 매우 크기 때문에 멀티미디어 서비스들은 높은 대역폭을 요구하는 것과 동시에 고성능의 컴퓨터를 필요로 하고 있다. 따라서 이러한 멀티미디어 데이터를 재생 매체의 수용 능력을 고려하여 높은 비트율을 갖는 스트림을 낮은 비트율로 변환하기 위한 기술인 트랜스코딩(Transcoding) 기법이 반드시 필요하다.

무선망에서는 네트워크 대역폭이 유선망보다 상대적으로 열악한 환경을 가지고 있다. 또한 모바일 호스트의 낮은 컴퓨팅 파워와 시스템 자원은 서버로부터 전송되는 높은 품질과 고용량의 멀티미디어 스트림을 적절하게 처리할 수가 없기 때문에 호스트에 따라 스트리밍을 적절한 품질과 크기를 바꾸어 트랜스코딩하는 연구들이 진행 중에 있다[1].

트랜스코딩이란 멀티미디어 컨텐츠(contents)를 어떤 포맷에서 다른 포맷으로 변환하는 과정이다. 트랜스코딩은 호스트 사양에 맞게 미디어 스트리밍의 프레임 전송속도(frame rate), 해상도 및 디스플레이 크기를 조절한다. 뿐만 아니라 MPEG1 포맷을 MPEG4포맷으로 변형한다던지

JPEG 포맷을 GIF 포맷으로 변형하는 작업을 수행하기도 한다. 그리고 유무선망간의 대역폭의 차이에 따른 비트율(bit rate)를 조절하는 기능도 가지고 있다[2].

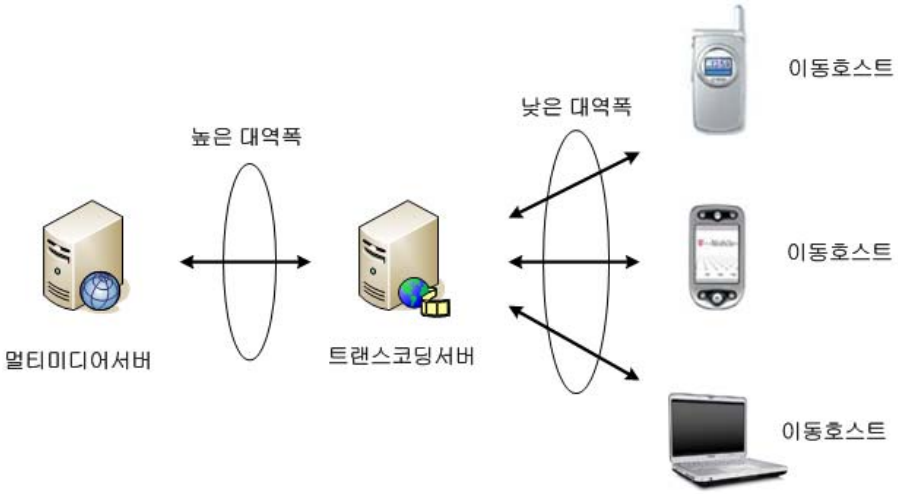

그림 1 트랜스코딩 시스템 기본구조

그림 1은 트랜스 코딩 시스템의 기본 구조이다. 그림1 에서 보는 것처럼 트랜스코딩 시스템은 고화질의 영상데이터를 가지고 있는 멀티미디어 서버와 호스트환경에 맞게 데이터를 변환하는 트랜스코딩 서버로 구성된다. 이동호스트에서 멀티미디어 요청을 하게 되면 인터넷상의 높은 대역폭을 요구하는 미디어 스트리밍을 낮은 비트율의 미디어 스트리밍으로 바꾸어 서비스를 하게 된다.

MPEG 비디오 스트리밍은 무선 환경에 따라 여러 등급으로 나눌 수가 있으며 각 등급에 따라서 CPU, 네트워크 대역폭, 메모리 등의 요구되는 컴퓨팅 자원이 각각 틀리다. 트랜스코딩은 미디어데이터의 형식과 크기 등을 변환하기 때문에 트랜스 코딩을 수행할 시스템의 자원 소모가 많

다. 자원소모가 많은 트랜스코딩시 서버의 자원소모가 스트리밍 등급에 따라 각각 틀리기 때문에 특정 서버에 높은 등급의 트랜스코딩 부하가 집중되지 않도록 효율적으로 처리할 수 있는 자원분배 알고리즘이 중요하게 된다.

본 논문에서는 트랜스코딩시스템 부하분산 연구를 위하여 공개 소스 기반인 리눅스를 이용하여 클러스터 트랜스코딩 시스템을 구현하였다. 트랜스코딩 시스템은 트랜스코딩 서버, 부하분배서버, 클라이언트로 구현하였다. 트랜스코딩서버에서는 MPEG 프로화일 등급에 따른 트랜스코딩과 시스템자원 상태정보를 분배서버에 보내주는 기능을 구현하였고 분배서버에서는 본 논문에서 제안한 부하분배 알고리즘을 이용하여 부하를 분배하도록 구현하였다.

본 논문에서 실험은 사용자들의 다양한 영화선호도 환경에서 트랜스코딩 시스템의 부하분산을 측정하였다. 트랜스코딩 부하분산 알고리즘은 기존의 클러스터 웹서버의 부하분산 알고리즘인 WRR을 보완하여 동적으로 시스템 가중치가 바뀌는 DWRR 알고리즘과 스트리밍 서비스의 모바일 호스트 등급에 따른 부하 가중치를 계산하여 부하배분에 사용할 수 있는 RWRR 알고리즘을 구현하고 측정하였다. 제안된 알고리즘 방법은 기존의 부하분산 알고리즘인 RR, WRR과 비교하여 효과적인 부하분산이 된다는 것을 실험을 통하여 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 관련된 연구들과 트랜스코딩 시스템에 대해 살펴본다. 3장에서는 부하분배 알고리즘인 RR, WRR, LC, DWRR, RWRR 방식에 대하여 알아보고 4장에서는 트랜

스코딩시스템의 구조에 대해서 살펴보겠으며 5장에서는 구현된 시스템의 성능을 측정하고 결과를 분석한다. 6장에서는 본 논문의 결론을 맺고 향후 연구 방향에 대해 기술한다.

Ⅱ. 관련 연구

이 장에서는 무선망의 대역폭에 대한 연구와 멀티미디어 스트리밍의 트랜스코딩을 위한 MPEG 프로화일 등급에 따른 표준에 대해서 살펴본다. 또한 클러스터 트랜스코딩 시스템의 구조와 부하분산 알고리즘에 대해서도 살펴본다.

1. 무선망 대역폭

무선망에 있는 호스트들은 유선망에서의 멀티미디어 스트림을 무선망에서 서비스 받기에는 네트워크 대역폭의 차이가 크기 때문에 적절하지 못하다. 따라서 멀티미디어 스트리밍의 트랜스코딩시 어느 정도의 대역폭으로 변환할 것인가에 대한 무선망의 대역폭에 대한 고려가 필요하다.

무선 통신의 무선랜 표준규약인 802.11은 IEEE 작업 그룹이 개발한 무선랜을 위한 규격 모음으로 802.11, 802.11a 802.11b, 802.11g등 4가지 규격이 이에 속한다[3]. 802.11은 기초적인 무선랜 표준안으로 CSMA/CA 를 지원하고 최대 속도는 2Mbps 이다. 802.11b는 11Mbps의 최대 속도를 제공하는 무선랜 표준이다. 많은 모바일 호스트 제품이 802.11b의 표준을 지원하고 있다. 802.11g 와 802.11a 는 최대 54Mbps 까지 데이터 전송을 지원하는 표준으로 차세대 무선랜을 지원하기 위한 표준이다. 본 논문에서는 802.11과 802.11b의 규격 속도를따른다.

2. MPEG profile

모바일 호스트에 따른 컴퓨팅 파워, 메모리, 네트워크 대역폭이 각각 틀리기 때문에 멀티미디어 스트리밍 서비스 역시 호스트 환경에 따라 달라져야 한다. 이러한 환경에 따른 표준 규격을 설정한 MPEG 프로파일을 살펴보면 다음과 같이 멀티미디어 스트리밍 서비스에 따른 Video size, Frame rate, Bit rate 등을 설정하고 있다[4].

등급	Video size	Frame rate	Bit rate(kbps)
SQCIF	128×96	15	50
QCIF	176×44	15	70
CIF	352×288	26	100
4CIF	704×576	30	200

표 1 MPEG 등급에 따른 프로파일 예

표1을 보면 MPEG 등급에 따른 규격이 모바일 호스트 환경에 따라 화면크기나 비트율이 다르게 설정됨을 알 수가 있다. MPEG 프로파일을 살펴보면 크게 SQCIF(Sub-Quater Common Intermediate Format), QCIF(Quater Common Intermediate Format), CIF, 4CIF급으로 나눌 수가 있다. SQCIF급은 모바일 핸드폰, QCIF급은 PDA, CIF급은 무선 노트북, 4CIF급은 일반 PC급으로 볼 수 있다. 따라서 트랜스코딩 시스템에서의 트랜스코딩 역시 표1에서 보인 MPEG 프로파일에서 설정한 등급으로 나눌 수가 있다.

3. 트랜스코딩 시스템

트랜스코딩 시스템의 구조는 그림 2와 같다. 클라이언트에서 트랜스코딩에 필요한 정보를 트랜스코딩 서버에 전송을 한다. 트랜스코딩 서버에서는 영상 데이터를 클라이언트에서 요청한 파라메터를 기반으로 영상 크기, 비트율, 영상포맷 등을 결정하고 미디어 스트림을 트랜스코딩한 후 사용자에게 전송한다. 예를 들면 트랜스코딩 서버로부터 CIF(352X288)등급의 25프레임/초, 비트율 100kbps의 비디오 스트리밍을 QCIF(176X44)등급의 15프레임/초, 비트율 50kbps의 스트리밍으로 클라이언트에 전송할 수가 있다. 즉, 트랜스코딩 서버에서는 실시간 무선망의 특성에 적합하도록 인코딩 비트율, 해상도, 프레임을 가변적으로 줄일 수가 있다.

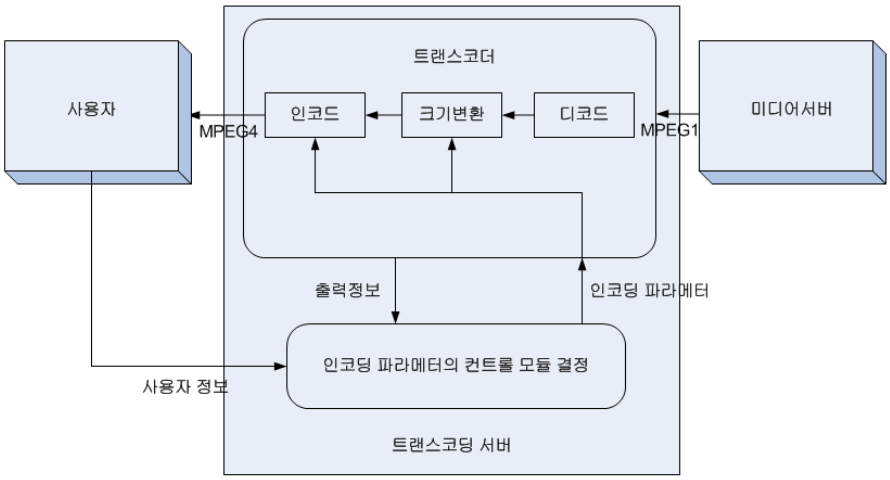

그림 2 트랜스코딩 시스템 구조

기존의 트랜스 코딩 시스템을 구축하는 방법을 살펴보면 미디어 서버에서 클라이언트 등급에 따라 트랜스코딩된 미디어 파일을 보내주는 방식이 있었다[5]. 이러한 방식은 클라이언트의 요구에 대하여 미디어 파일들

을 미리 클라이언트 등급별로 트랜스코딩 하여 저장하는 방식이다. 그림 3의 경우 소스기반의 정적 인코딩 시스템구조를 보여준다. 이러한 시스템은 미디어스트리밍을 트랜스코딩을 하여 전송하는 방식보다 서버의 부하가 심하지 않다는 장점이 있으나 무선랜 환경에서 클라이언트가 이동시 네트워크 변화량에 적응할 수 없다는 점과 모든 미디어 파일들을 각 등급별로 트랜스코딩하여 미디어영상 파일을 저장해야 한다는 단점이 있다. 또한 영상크기, 프레임율, 비트율, 네트워크 대역폭이 다를 경우에도 동일한 등급의 스트리밍을 받기 때문에 무선망 네트워크에서는 적합하지가 않다.

그림 4와 같이 미디어 스트림을 클라이언트 요청에 따라 트랜스코딩하여 미디어 데이터를 전송하는 정적 트랜스코딩 시스템 방식이 있다[5]. 클라이언트에서는 연결된 무선기지국에서 가장 가까운 트랜스코딩 서버를 선택을 하고 스트리밍 서비스를 받게 된다. 하지만 트랜스코딩 서버를 선택함에 있어서 그 이외에 서버를 선택하는 기준이 없기 때문에 특정 트랜스코딩 서버에 집중될 수 있는 단점이 있다.

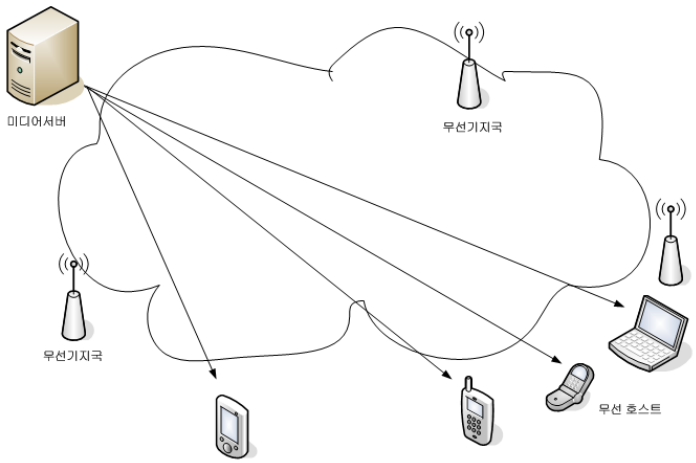

그림 3 소스기반의 정적 인코딩 시스템

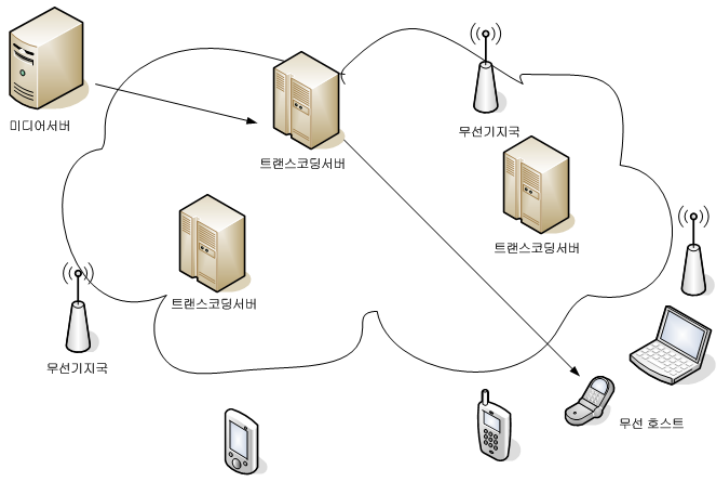

그림 4 정적 트랜스코딩서버 할당

그림 4와 같은 정적 트랜스 코딩 서버는 특정 서버에 집중 요청으로 인한 부하 불균형이 일어날 수가 있기 때문에 이러한 트랜스코딩 서버의 부하 상태를 파악하는 그림 5와 같은 분배서버를 두게 되는 시스템 구조를 사용한다[5].

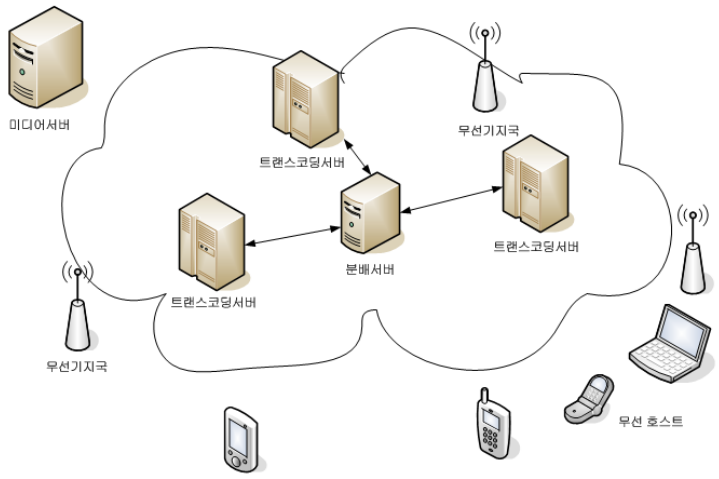

그림 5. 부하분산 트랜스코딩 시스템

분배서버에서는 트랜스코딩 서버와 연동하여 부하분산 정책에 따라 트랜스코딩서버를 선택하게 된다. 클러스터링 서버 부하 분산 정책으로는 RR, WRR, DWRR, LC 알고리즘 등이 있다. 이들 중 DWRR 알고리즘은 구현이 어려운 단점이 있지만 다른 정책보다 전반적 성능이 우수함을 보이고 있다[6]. 하지만 DWRR 방식은 클라이언트 접속 요청이 많아짐에 따라 클러스터 서버들의 작업부하를 계속 측정하여 새로운 가중치를 만들게 되므로 이에 따른 부하가 발생할 수 있다[7].

Ⅲ. 부하분산 정책

이 장에서는 기존의 클러스터웹서버 시스템의 상태 및 분배서버의 알고리즘에 따른 부하분산 정책에 대하여 설명하고 이를 응용하여 본 논문에서 제안하는 DWRR 알고리즘과 RWRR 알고리즘에 대해서도 설명한다.

1. Round Robin

라운드로빈 방식은 클라이언트 요청이 들어오면 차례대로 각 서버에게 할당하는 방식을 말한다. 라운드로빈 방식은 우선순위의 개념이 없고 실제서버의 연결 개수나 반응시간 등은 고려를 하지 않는다. 스케줄링의 기초는 클라이언트 기반이며 실제 서버의 시스템 상태를 고려하지 않기 때문에 효과적인 부하 분산을 기대하기 힘들다. 따라서 서버사이에 부하 불균형이 심각해지는 현상이 발생할 수 있다.

2. Least-Connection Scheduling

최소 접속 스케줄링은 가장 접속이 적은 서버로 요청을 직접 연결하는 방식을 말한다. 각 서버에서 동적으로 실제 접속한 숫자를 세어야하므로 동적인 스케줄링 알고리즘중의 하나이다. 클러스터형 웹서버에서 비슷한 성능의 서버로 구성된 경우에는 아주 큰 요구가 한 서버로만 집중되지 않기 때문에 데이터가 작고 사용자 접속부하가 매우 큰 경우에 효과적으로 분산을 한다.

이종 서버 환경에서 LC알고리즘 방식은 가장 처리가 빠른 서버에 더 많은 네트워크 접속을 처리할 수 있다. 그렇지만 실제로는 TCP의 시간 대기상태 때문에 아주 좋은 성능을 낼 수는 없다. TCP의 시간 대기상태는 보통 2분이다. 그런데 접속자가 아주 많은 웹 사이트는 2분 동안에 몇 천개의 접속을 처리해야 할 경우가 있다. 서버 A는 서버 B보다 처리용량이 두 배일 경우 서버 A는 수천 개의 요청을 빨리 처리하고 시간대기 상황에 직면하게 된다. 그렇지만 서버 B는 몇 천개의 요청이 처리되기만을 기다리게 된다. 따라서 다양한 처리용량을 지닌 서버들로 구성되었을 경우 LC 부하분산 방식이 효율적으로 되지 못할 수 있다.

3. Weighted Round-Robin Scheduling

가중치 기반 라운드 로빈 스케줄링은 실제 서버에 서로 다른 가중치를 설정할 수 있다. 각 서버에 가중치를 임의로 부여할 수 있으며, 여기서 지정한 가중치 값을 통해 처리 순서를 정한다. 기본 가중치는 1로 정한다면 예를 들어 실제 서버가 A, B, C 이고 각각의 가중치가 4, 3, 2 일 경우 스케줄링 순서는 ABCABCABA 가 된다.

라운드 로빈 스케줄링은 가중치 기반 라운드 로빈 스케줄링의 한 종류이며 모든 가중치가 동일한 경우이다. 서버의 가중치를 변경하고 나서 스케줄링 순서를 생성하는 데에는 거의 과부하가 걸리지 않으며 실제 스케줄링에서도 어떠한 과부하도 추가하지 않는다. 그러므로 서버의 가중치가 같은 라운드 로빈 스케줄링만 실행하는 것은 비효율적이다.

4. Dynamic Weighted Round-Robin Scheduling

DWRR 알고리즘은 WRR 알고리즘처럼 시스템 사양에 따라 고정된 가중치를 가지는 것이 아닌 주기적으로 서버의 상태정보를 파악하여 동적 가중치를 계산하여 분배하는 알고리즘이다. 본 논문에서는 클러스터 웹서버의 WRR방식을 응용하여 서버의 상태정보를 이용하여 동적 가중치를 계산하여 부하를 분배하는 시스템을 구현하였다. 서버의 정보를 측정하는 방식에는 주기적으로 서버의 정보를 보내는 방식과 클라이언트에서 요청이 들어왔을 경우에 서버의 정보를 분배서버에 보내는 두 가지 방식이 있다.

그림 6은 서버의 정보를 분배서버에 보내 부하를 측정하는 시스템 구조이다. 주기적으로 서버의 정보를 보내는 방식은 서버의 상태파악을 쉽게 할 수가 있다는 장점이 있으나 서버가 늘어날 경우 분배서버의 네트

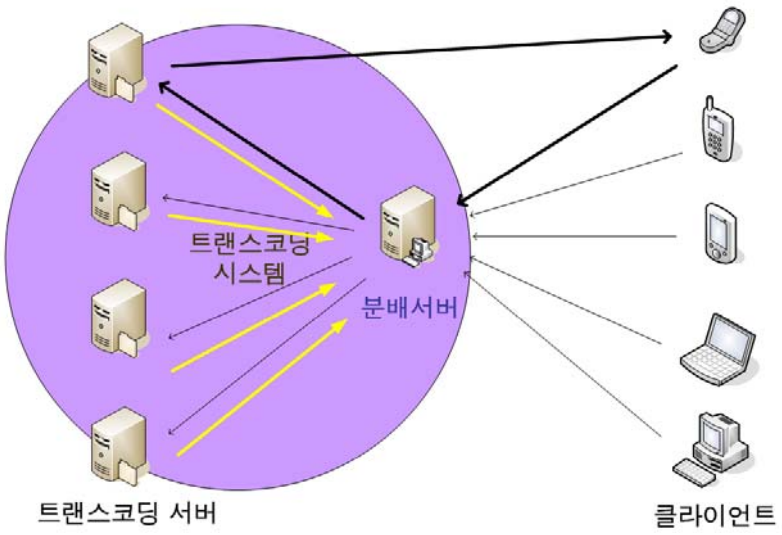

그림 6 Dynamic Weight Round Robin

워크 대역폭 소모가 발생하기 때문에 서버의 확장성에 문제가 발생할 수 있다. 클라이언트에서 요청이 들어올 경우에만 서버의 상태정보를 보내는 방식은 주기적으로 서버의 상태정보를 보내는 방식보다 네트워크 대역폭 소모를 덜 한다는 장점이 있으나 서버가 다운되었을 경우 시스템 다운을 인지하기 까지 지연이 발생할 수 있다. 그리고 동시 접속 클라이언트가 늘어나게 되면 각 분배서버에 부하정보를 요청하고 계산하는데 지연으로 인해 효과적인 자원분배를 할 수 없다는 단점이 있다.

그림 7은 DWRR 알고리즘의 시퀀스 다이어그램을 나타낸 것이다. 트랜스코딩 서버와 분배서버는 TCP 소켓을 생성하여 연결하고 트랜스코딩 서버는 1초에 1번씩 주기적으로 서버의 CPU사용량, 메모리, 네트워크 정보를 분배서버에 보내주게 된다. 분배서버에서는 각 서버의 상태정보를 이용하여 가장 최적의 서버를 선택하게 된다. 클라이언트에서는 두 개의

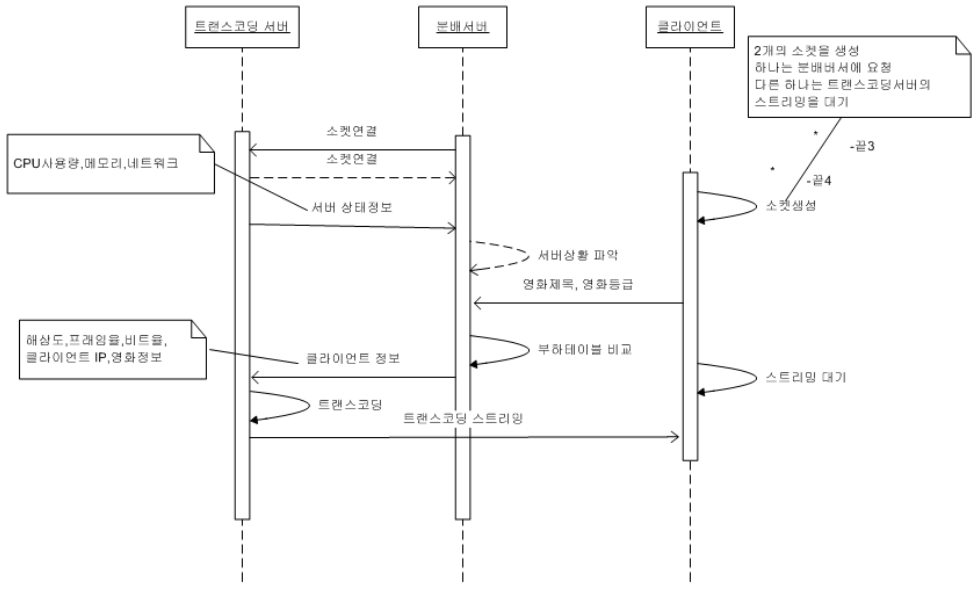

그림 7 DWRR 시퀀스 다이어그램

TCP 소켓을 연결한 후에 하나는 분배서버와 연결하고 다른 소켓 하나는 트랜스코딩서버의 스트리밍을 대기하고 있게 된다.

클라이언트에서 분배서버에 요청을 할 때 영화제목, 영화등급, 해상도, 프레임율, 비트율, 클라이언트 IP정보를 보내게 되고 분배서버에서는 서버의 상태정보 테이블을 이용하여 최적의 트랜스코딩 서버를 선택하게 된다. 선택된 트랜스코딩 서버에 클라이언트 요청정보를 전송하고 전송된 정보를 이용하여 MPEG 프로파일 등급에 따른 트랜스코딩을 한 후 다시 클라이언트에게 스트리밍을 전송하게 된다.

5. Resource Weighted Round-Robin Scheduling

그림 8은 자원 가중치 기반 알고리즘 시스템 구성도이다. RWRR방식은 미리 계산된 가중치를 이용하여 부하를 분산하기 때문에 서버가 크게 늘어나더라도 DWRR 방식처럼 분배서버에 부하가 발생하지 않는다.

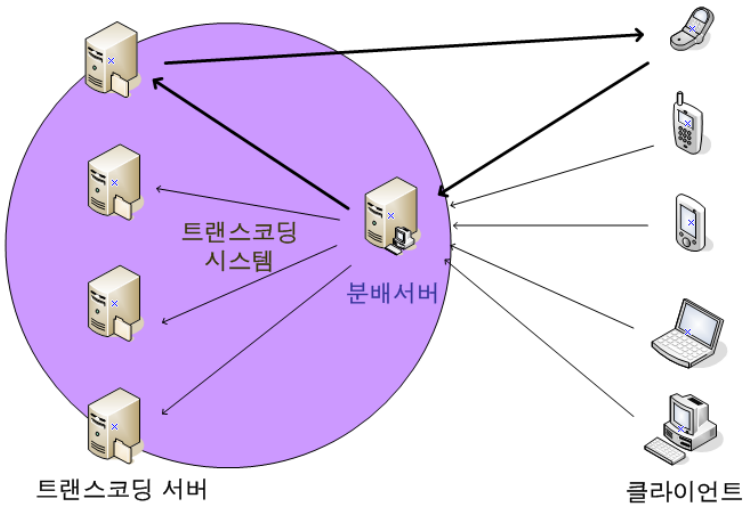

그림 8 Resource Weight Round Robin

그림 9는 RWRR 알고리즘의 시퀀스 다이어그램이다. 트랜스코딩 서버와 분배서버에서 소켓연결설정을 한 후 서버에서는 클라이언트의 요청을 받아 들인다. 클라이언트에서는 두 개의 소켓을 생성하고 하나는 분배서버에 클라이언트 정보를 보내고 트랜스코딩 스트리밍을 대기한다. 분배서버에서는 클라이언트의 정보를 분석하여 미리 계산된 가중치 테이블을 이용하여 트랜스코딩 서버를 선택하게 된다.

가중치 테이블로만 부하분산을 하는 경우에는 트랜스코딩시 서버에서 부하 측정 방식이 정확하지 못하여 부하 불균형이 일어날 수 있다. 이를 보완하기 위해서 트랜스코딩이 완료된 경우나 클라이언트 접속이 끊어졌을 경우에 서버의 정보를 분배 서버에 보낸다. 분배서버에서는 가중치 테이블과 서버의 상태정보를 이용하여 분배서버의 부하는 줄이면서 좀 더 정확한 부하 측정을 하게 된다.

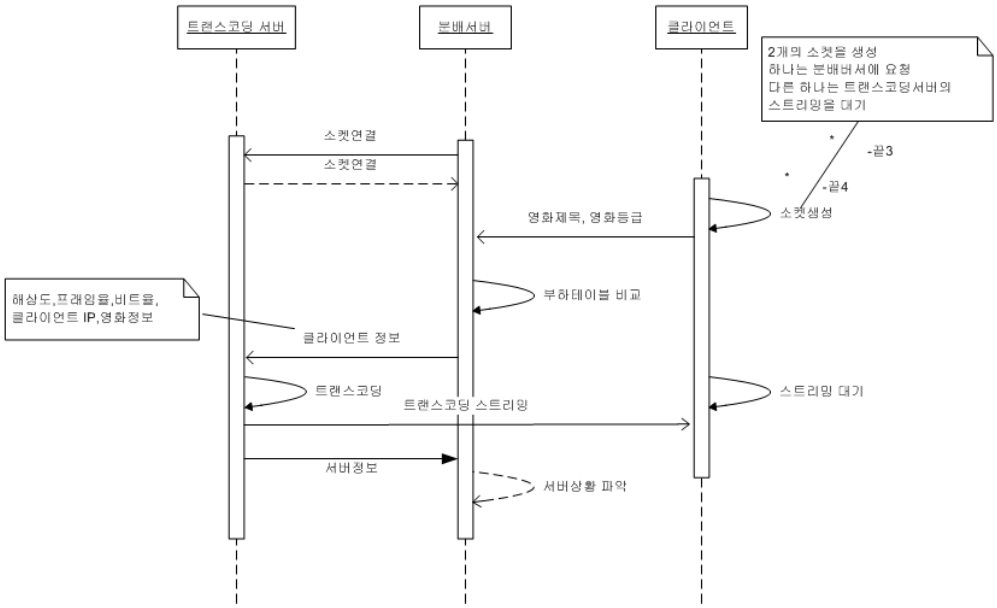

그림 9 RWRR 시퀀스 다이어그램

Ⅳ. 클러스터 트랜스코딩 시스템 구조

이 장에서는 부하분산 알고리즘을 구현한 클러스터 트랜스코딩 시스템에 대하여 설명한다. 트랜스코딩 서버, 분배서버, 클라이언트의 구조 및 연구 환경에 대하여 기술하고 구현된 클러스터 트랜스코딩 시스템의 동작에 대해 살펴본다.

1. 클러스터 트랜스코딩 시스템

시스템의 구조는 그림 10에서 보는 바와 같이 시스템을 관리하고 부하를 분산하는 분배서버와 미디어 데이터를 관리하고 트랜스코딩 데이터를 전송하는 트랜스코딩 서버로 구성된다.

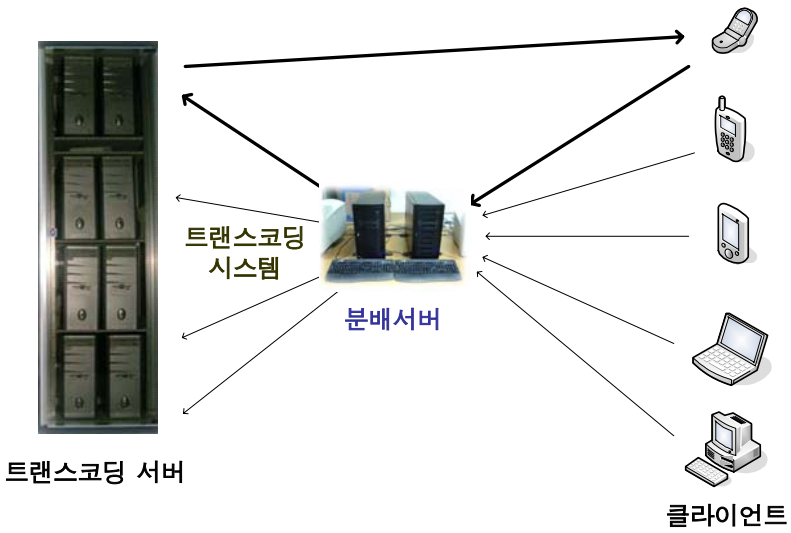

그림 10 클러스터 트랜스코딩 시스템

2. 트랜스코딩 서버

트랜스코딩 서버는 미디어 데이터를 저장하고 있고 분배서버의 요청에 따라 사용자에게 트랜스코딩 스트림을 전송하는 서버이다. DWRR알고리즘 방식의 경우 1초 간격으로 자신의 시스템 자원 상태 정보를 분배서버에게 보고하고 그 외 알고리즘은 서버상태를 파악하지 않고 분배 알고리즘에 따라 부하 분산를 한다.

트랜스코딩 프로그램은 오픈소스 프로젝트의 ffmpeg를 수정하여 사용하였다[8]. ffmpeg는 트랜스코딩시 CPU를 최대로 사용하여 최대한 빨리 트랜스코딩 하기 때문에 실시간으로 스트리밍으로 보내기에는 부적합하다. 따라서 그림 11과 같이 재생시간에 맞추어 1초마다 필요한 프레임을 트랜스코딩 하도록 소스를 수정하였고 트랜스코딩된 데이터를 디스크에 저장하는 것이 아니라 스트리밍 서비스를 하도록 소스를 수정하였다.

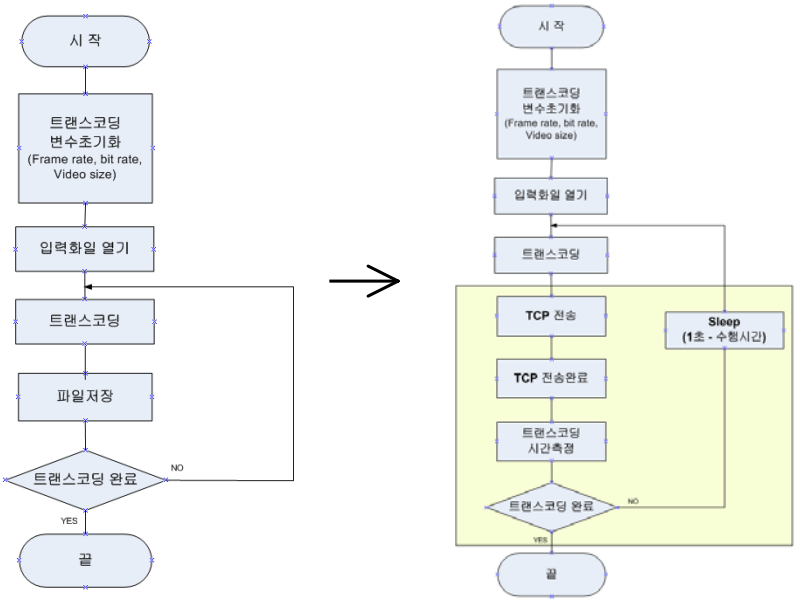

그림 11 수정된 ffmpeg 구조

트랜스코딩된 미디어 스트림은 미리 정의 되어 있는 TCP연결 소켓으로 스트림을 전송하게 되고 DWRR 알고리즘 방식 같은 경우에는 영화의 트랜스코딩 과정이 끝나거나 중간에 클라이언트 접속이 끊길 경우 트랜스코딩 서버의 상태 정보를 분배서버에 전송하게 된다.

그림 12는 트랜스코딩 시스템에 구현된 트랜스코딩 서버의 구성을 보여준다. 클라이언트에서 보내진 상태정보를 분배서버에서 트랜스코딩서버로 전송하게 되면 ffmpeg에서는 디스크에서 해당 영화를 읽어 들인 후 클라이언트에서 요청한 영화제목, 프레임율, 영상크기, 비트율의 정보대로 트랜스 코딩을 하게 된다. 트랜스코딩된 데이터는 클라이언트 IP에 스트리밍 전송을 하게 된다.

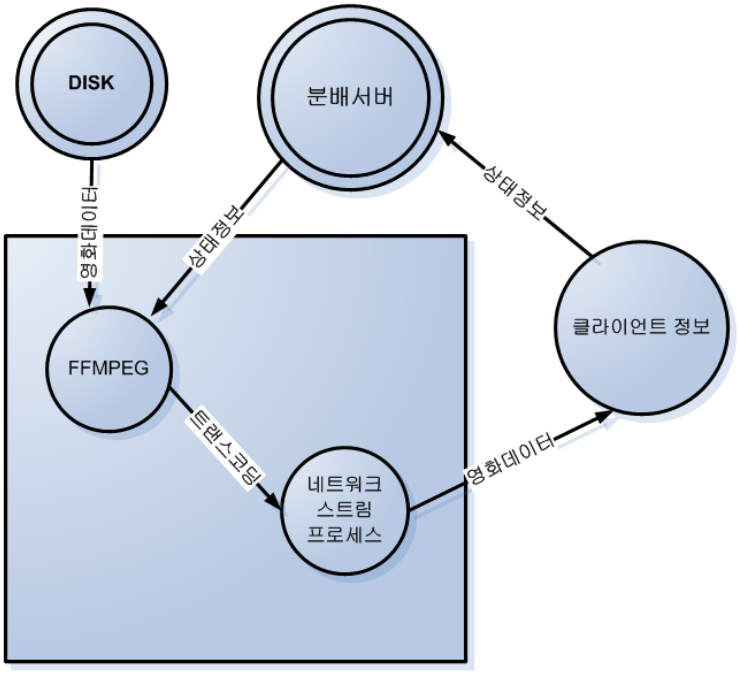

그림 12 트랜스코딩 서버 구성

그림 13은 트랜스코딩 시스템에서 스트리밍 서비스를 하는 실험화면이다. 클라이언트에서 요구하는 영화 데이터를 불러온 후 클라이언트 요구에 따라 영상 크기, 프레임율, 비트율을 트랜스코딩하여 스트리밍 서비스를 하는 모습을 보여준다. 3.avi의 CIF급 미디어파일이 클라이언트로 프레임은 432frame, 화면크기는 1318Kbyte, 비트율은 374.9kbps/s로 트랜스코딩 되어 전송되고 있다.

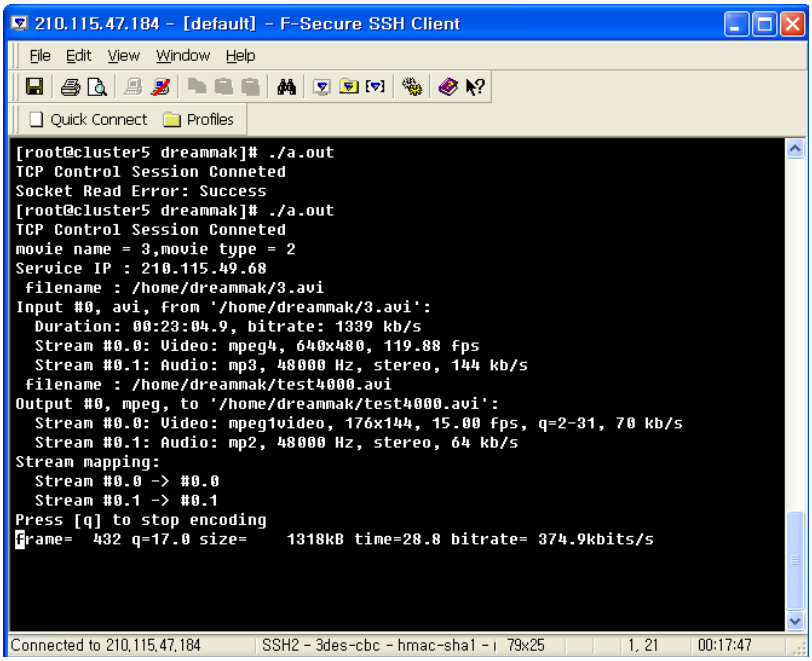

그림 13 트랜스코딩서버 실행화면

3. 분배 서버

분배서버에서는 3가지 알고리즘별로 구현이 되었다. RR, DWRR, RWRR 으로 구현하였으며 알고리즘에 따라 트랜스코딩 서버를 선택하게

된다. 동작과정을 살펴보면 트랜스코딩과 소켓연결 후에 클라이언트에 요청을 받아들이게 된다.

DWRR알고리즘 방식의 경우 각 트랜스코딩 서버에서 1초마다 서버의 CPU, 메모리, 네트워크 상태정보를 수집한다. 각 수집한 데이터는 분배서버에 파일로 저장되게 되고 파일에 저장된 정보를 가지고 CPU가 가장 적은 트랜스코딩 시스템에 다음 클라이언트 요청을 분배하게 된다.

RR, RWRR 알고리즘의 경우에는 순차적으로 트랜스코딩 서버에 부하를 분산하거나 가중치 테이블을 이용하여 부하를 분산한다. 1대의 분배서버로 구성되어 있으며 부하 분산 프로그램은 C로 구현하였다.

그림 15는 분배서버의 RWRR 알고리즘의 부하 분산 화면이다. 클라이언트의 영화요청을 받아들인 후 등급별로 가중치를 계산한 부하테이블을 이용하여 부하를 분산한다. 클라이언트에서 영화 정보, 영화 등급, IP 주소를 읽어 들인 후 영화 등급의 가중치를 가중치 테이블에 더한 후 트랜스코딩서버에 클라이언트 정보를 전송하게 된다. 그림 15에서는 트랜스코딩서버 1, 2번에 qcif, sqcif 급의 클라이언트 요청이 전송되고 가중치가 할당되는 화면이다.

4. 클라이언트

클라이언트는 부하를 발생하는 시스템, 트랜스코딩된 스트리밍 데이터를 읽어 들여 미디어 영상을 상영하는 시스템 2개로 구성되어 있다. 그림 14, 16은 부하발생 화면과 스트리밍 데이터를 읽어 들이는 화면이다.

부하 발생은 총 클라이언트 요청수를 입력하면 부하분포 형태로 각 등급별 요청개수가 할당된다. 등급별 요청개수가 할당되면 서버에 Zipf분포를 따라 랜덤하게 요청개수만큼 부하를 발생시키데 된다. 영상 상영은 mplayer를 이용하였으며 스트리밍 데이터를 읽기위한 소스 수정을 하였다. 그림 14는 총 30개의 부하를 발생했을 경우 Zipf분포를 따라 sqcif, qcif, cif 급으로 나누어 부하를 발생하는 화면이다. 그림 16은 트랜스코딩서버에서 전송된 미디어 데이터를 sqcif급인 128×96 크기로 재생하는 화면이다.

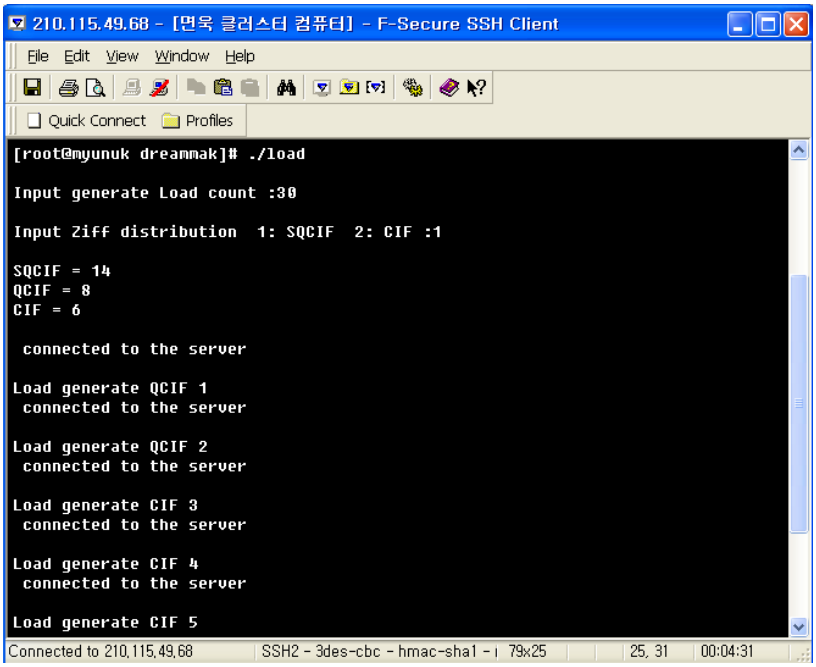

그림 14. 부하발생 프로그램 실행화면

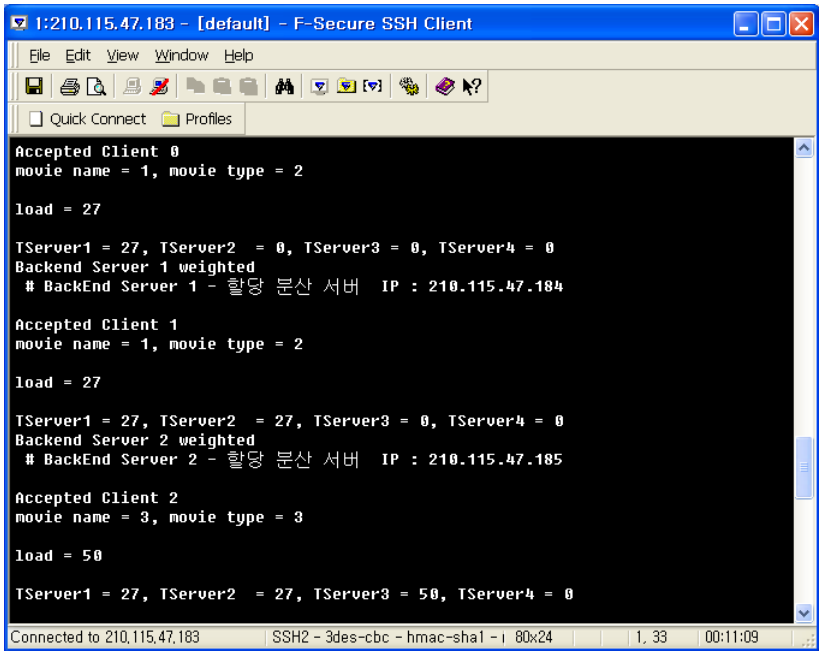

그림 15 분배서버 관리 화면

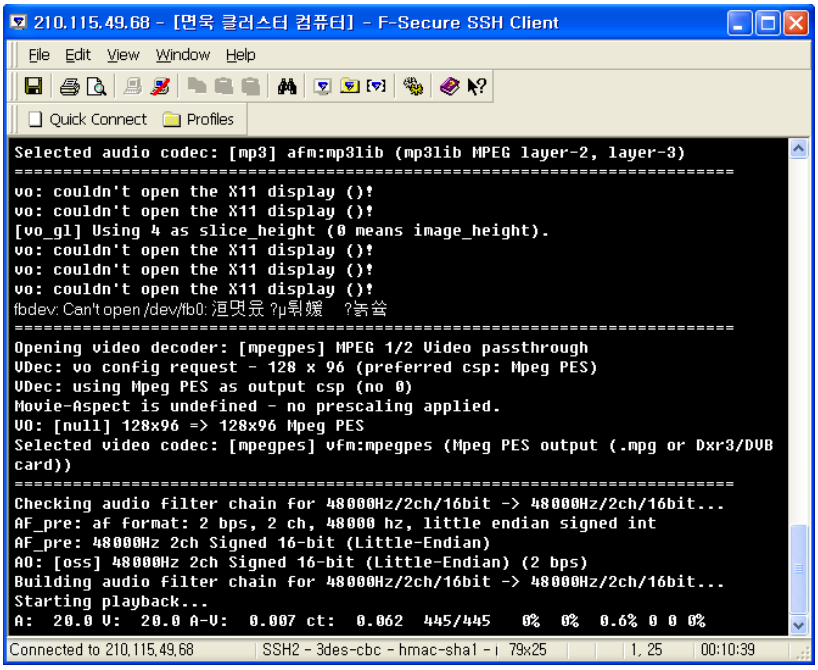

그림 16 클라이언트 재생화면

Ⅴ. 트랜스코딩에서 소모되는 자원량 측정

1. 자원 소모량

자원 가중치는 트랜스코딩시 영화 등급별에 따라 CPU, 네트워크, 메모리 자원소모를 측정하였다. 4CIF급의 영상을 SQCIF, QCIF, CIF 급으로 트랜스 코딩하여 측정을 해보았다.

시스템 사양은 CPU 1.4GHz, 메모리 256M, 네트워크 100Mbps, 운영체제는 리눅스(커널 2.4.22) 을 이용해 측정하였다. 실험 방법은 같은 등급만 트랜스 코딩하는 방법과 랜덤하게 여러 등급을 동시에 트랜스 코딩하는 것으로 측정을 해보았다. 측정 결과 SQCIF, QCIF, CIF 등급으로 트랜스코딩을 할 때 소모되는 자원소모가 각각 일정하다는 것을 알 수 있었다. 표 2는 등급별 트랜스코딩시 소모되는 CPU, Memory, Network 자원 소모 상황을 보여준다.

자원 소모량은 트랜스코딩 할 때마다 약간의 변화는 있었지만 크게 벗어나는 경우는 없었고 거의 고정적인 비율의 자원 소모를 보여주고 있었다. 따라서 트랜스코딩시 자원 소모율이 일정한 것을 기준으로 해서 각 등급별 자원 소비 가중치를 구할 수가 있다. 4CIF급의 요청이 들어왔을 경우 트랜스코딩을 거치지 않고 바로 서비스가 되기 때문에 4CIF급은 트랜스코딩 자원 소모에서 제외 하였다.

등급	CPU	Memory	Network
SQCIF	8.3%	5.7Mb	50kbps
QCIF	8.5%	5.8Mb	70kbps
CIF	16.3%	6.4Mb	100kbps

표 2 등급별 자원 소모량

2. 자원가중치 계산 알고리즘

표 3는 자원가중치 계산 알고리즘을 나타낸다. 클라이언트에서 요청하는 요청등급별로 자원 소모량을 측정하여 각 등급별 가중치를 계산하게 된다. M은 메모리를 나타내며 B는 네트워크 대역폭 C는 CPU를 나타낸다. 미디어스트림 등급을 i개로 나누었을 경우 Qci, Qri, Qmi 는 트랜스코딩시 등급별 소모되는 CPU, 네트워크, 메모리 사용량이다. 즉 트랜스코딩을 SQCIF, QCIF, CIF, 4CIF 이렇게 4등급으로 나누면 Qc_1, Qc_2 Qc_3 Qc_4는 각 등급에서의 트랜스코딩시 요구되는 CPU 사용량을 나타낸다.

서버의 가중치자원 W를 정할 때 트랜스코딩서버에서 클라이언트를 허용할 수 있는 최대 접속수를 계산한다. 각 자원별로 등급별 소모되는 최대 접속수를 계산하여 그 중 클라이언트를 허용할 수 있는 접속수가 가장 작은 자원이 가중치 자원으로 결정된다. 예를 들어 동일한 등급의 미디어스트림을 트랜스코딩시 소모되는 메모리 사용량으로만 분배를 했을 경우에는 11개, 네트워크는 9개 CPU는 4개의 최대 처리량을 가진다고 가정한다면 메모리자원 사용량으로는 최대 11개까지 부하분산을 할 수 있으나 CPU 자원이 5개 이상의 처리를 할 수 없으므로 CPU의 자원으로 부하를 분산해야한다.

서버의 가용 메모리에서 등급별로 소모되는 메모리양은 M / Q_{mi} 로 구해진다. CPU 사용가능한 할당 범위에 등급별 소모되는 CPU의 비율 C / Q_{ci} 그리고 네트워크 대역폭에서 등급별로 소모되는 대역폭은 B / Qri 로 구할 수가 있다. 이중 가장 클라이언트 허용 가용치가 낮은 수치가 서버의 자원 가중치로 설정하고 부하 테이블을 계산한다. 예를들어 CPU가 가장 낮은 허용가중치인 경우 다음의 식1에서 등급별 CPU 사용량 가중치를 계산할 수 있다.

$$W_n = \frac{Qc_n \times 100}{\sum_{k=1}^{i} Qc_k} \quad (n = 1,2,...,i) \quad \text{..... 식 1}$$

현재 i는 4로 4개의 등급으로 미디어 스트림을 구분하였다. 식 1을 살펴보면 CPU가 가장 자원소모가 큰 경우 CPU의 등급별 소모량의 가중치가 계산된다. 해당 등급의 CPU 점유율 Qc_n에 전체 등급별 CPU의 점유율 $\sum_{k=1}^{i} Qc_k$를 나누게 되면 등급별 CPU 가중치가 계산된다. 예를 들어 SQCIF 급의 CPU 점유율이 12%, QCIF 급의 CPU 점유율이 20%, CIF 급의 CPU 점유율이 32%라고 하면 SQCIF 급의 가중치는 식 1에 의하여 18.75, QCIF 급의 가중치는 31.25 CIF급의 가중치는 50이 된다.

```
/* 서버 가중치 측정 */
if ( M / Qmi > B / Qri > C / Qci ) {
        Wn = Qcn × 100 / Σ(k=1..i) Qck    ( n= 1,2,...,i )
}
else if ( B / Qri > C / Qci > M / Qmi ) {
        Wn = Qmn × 100 / Σ(k=1..i) Qmk    ( n= 1,2,...,i )
}
else if ( C / Qci > M / Qmi > B / Qri ) {
        Wn = Qrn × 100 / Σ(k=1..i) Qrk    ( n = 1,2,...,i )
}
```

부하측정 통계
B : Network Bandwith
M : Memory
C : CPU
I : Number of level
Qc_i : CPU rate level
Qr_i : bit rate level
Qm_i : require memory level
W_n : Weight

표 3. 자원가중치 계산 알고리즘

3. 자원가중치 기반 분배 알고리즘

표 4는 자원가중치 기반 분배 알고리즘을 나타낸다. 분배서버에서는 트랜스코딩 서버에서 보내준 상태정보를 파일로 기록하고 그 파일 정보를 읽어 들여 가장 CPU사용량이 적은 트랜스코딩 서버를 선택하게 된다.

표4는 CPU, 네트워크, 메모리 중에서 CPU 가중치를 기반으로 트랜스코딩 서버를 선택하는 방식이다. 클라이언트에서 요청이 들어오게 되면 분배서버에서 클라이언트 정보를 얻게 되고 요청에 대한 처리를 하기 위해 thread_function() 함수를 생성한다. thread_function() 함수는 CPU사용량이 가장 적은 트랜스코딩 서버를 선택하기 위해 chkResource() 함수를 호출한다. chkResource() 함수는 분배서버에 저장된 트랜스코딩서버의 자원상태정보 파일을 읽어 들여 CPU사용량을 비교하고 선택한다. 표 4에서 트랜스코딩 서버의 수는 BACK_END에 저장이 되고 cpu_buff[BACK_END] 배열에 각 트랜스코딩 서버의 수만큼 CPU 사용량이 저장되게 된다. 배열에 각 트랜스코딩서버의 CPU 사용량을 저장하기 위하여 분배서버에 기록된 자원상태정보 파일을 lseek(), read() 함수를 이용하여 상태정보를 읽어 들인 후 cpu_buff 배열에 cpu사용률을 기록한다.

CPU 사용량이 가장 적은 트랜스코딩 서버를 선택하기 위하여 클라이언트 요청이 들어오면 서버의 수만큼 최소의 CPU 사용량 min값과 비교를 하여 최소의 CPU 사용량을 min에 저장을 한다. 비교를 통하여 min값에는 최소 CPU 사용량이 저장이 되고, cpu_buff[i]는 최소 CPU 사용률을 가진 i번째 트랜스코딩 값이 들어가게 된다. i번째 서버의 정보를 최종 서버선택 변수인 idx에 저장을 하게 되고 idx 값을 반환하여 해당 트랜스코딩 서버를 선택하여 부하를 분산하게 된다. 부하를 분산하고 난 후 트

랜스코딩서버 가중치인 weight[idx]에 미디어등급에 따른 가중치를 더하게 된다.

```
void *thread_function(void *arg){
                ...
  target = chkResource();
                ...
  write(back_fd[target], &tmp_info, sizeof(struct client_info ) );
  write(back_fd[target], &client_addr[client_no], sizeof( struct sockaddr_in ) );
                ...
  weight[idx]=weight[idx]+load;        // 트랜스코딩 서버 가중치 증가
                ...
}
int chkResource()
{
        int i, id;                          // BACKEND_NUM : 트랜스코딩 서버 수
        int cpu_buff[BACKEND_NUM];      // 트랜스코딩 서버 CPU사용량
        for (i=0; i<BACKEND_NUM; i++)
        {
         // 자원상태정보 파일을 열어 CPU 데이터를 읽는다.
                lseek(file_fd[i],0L,0);
                read(file_fd[i], &cpu_buff[i], BACKEND_NUM);
                printf("read data : %d₩n",cpu_buff[i]);
                if (cpu_buff[i] < min)
                {
                        min =cpu_buff[i];  // 최소 CPU 사용량 서버 구함
                        idx = i;
                }
        }
        return idx;                         // 트랜스코딩 서버 선택
}
```

표 4. 자원가중치 기반 분배 알고리즘

Ⅴ. 성능 측정 및 결과 분석

이 장에서는 트랜스코딩 시스템의 부하발생기와 부하발생 알고리즘에 대해 기술한다. 또한 트랜스코딩 서버의 부하분산 상태를 측정하고, 측정된 결과를 분석한다.

1. 부하 발생기

성능 측정을 위하여 부하 분산 프로그램을 만들었다. 부하 분산 프로그램은 영화정보, 영화선호도, 등급별 클라이언트 분포도를 기준으로 하여 부하를 발생하게 된다.

표 4 는 부하 발생 알고리즘이다. 클라이언트 요청 총 개수를 정한다음 부하발생 분포형태로 등급별로 개수를 정하게 된다. 예를 들어 총 클라이언트 요청수가 100이고 부하발생이 Zipf분포 형태로 4:3:2:1의 비율을 따른다면 sqcif = 40, qcif = 30, cif = 20, 4cif = 10으로 정해지게 된다.

sqcif, qcif, cif, 4cif 의 개수가 정해지게 되면 랜덤하게 부하를 4가지 중에 하나를 발생시킨다. 부하분포 형태대로 부하를 발생시키기 위하여 할당된 부하 개수를 초과하지 않도록 하였다. 즉 총 클라이언트 요청 개수에서 등급별로 발생빈도를 계산하여 각각의 부하들이 발생되게 하였다.

```
/* Work Load */
   while(sqcif>0 || qcif>0 || cif>0 || 4cif>0)
   {
          val=rand()%4+ 1;

          if(val==1 && sqcif>0){
                    sqcif=sqcif-1;
                    Workload(val);
          }
          else if(val==2 && qcif > 0){
                    qcif=qcif-1;
                    Workload(val);
          }
          else if(val==3 && cif > 0){
                    cif=cif-1;
                    Workload(val);
          }
          else if(val==4 && cif4 > 0){
                    cif4=cif4-1;
                    Workload(val);

    } // while end
}
```

표 5 부하 발생 알고리즘

부하발생 시스템은 부하분배서버에 초당 1개의 부하를 발생시키도록 하였고 부하분배서버에 영화를 요청시 영화제목, 해상도, 비트율, 영화등급을 전송한다. 요청한 영화의 선호도 분포는 Zipf분포를 따른다고 가정하였고 그래프의 기울기의 영향을 미치는 skew factor 는 일반 영화 선호도의 분포를 따르는 값인 0.271을 사용하였다[9].

실험에 사용된 영화의 정보는 표 5와 같다. 영화는 총 10개의 영화를 테스트 했으며 영화 선호도에 따른 분포형태는 그림 17과 같다. 그림 17은 사용자 요청을 총 10,000명으로 했을 때의 영화 선호도 분포를 나타낸것이다. 사용자의 영화 선호도에 따라 부하 발생은 10개의 영화에 대한 요청을 Zipf 분포 형태로 부하를 발생시켰다. Zipf 분포형태로 부하를 발생시키기 위해 클라이언트 총 요청 수에 따른 등급별 부하 분포를 먼저 계산한 후 계산된 분포도에 따라 부하를 램덤으로 발생시키게 된다.

클라이언트는 크게 모바일 휴대폰 기종의 SQCIF급, PDA기종의 QCIF급, 무선노트북 기종의 CIF급, 일반 PC기종의 4CIF급으로 나눌 수 있다. 클라이언트 단말기는 SQCIF급의 사용자가 가장 많고 단말기의 분포형태는 Zipf 분포를 따른다고 가정하였다. 단말기의 skew factor는 0.271을 사용하였다.

구분	영화제목	크기	초당프레임	상영시간
영화1	나루토	640×480	26	35분
영화2	냉정과열정사이	704×384	26	95분
영화3	토토로	640×480	26	30분
영화4	몽키턴	640×480	26	32분
영화5	미도리의 나날	640×480	26	30분
영화6	오늘부터 마왕	640×480	26	30분
영화7	천상천하	640×480	26	35분
영화8	연풍	640×480	26	35분
영화9	유고	640×480	26	30분
영화10	미루모데	640×480	26	30분

표 6 실험에 사용한 영화 정보

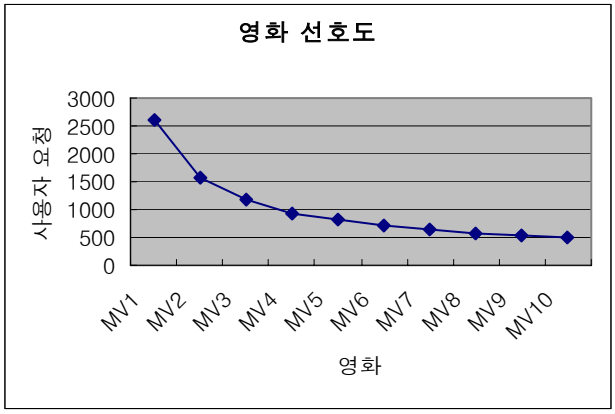

그림 17 영화 선호도 (Zipf 분포 : skew factor = 0.271)

2. 성능 측정 및 결과

2.1 성능 측정 환경

성능 측정은 24대의 트랜스코딩 서버의 자원 소모 상태를 측정하였다. 3개의 이종환경의 클러스터 시스템으로 구축되어 있고 영화는 각각 클러스터 시스템에 저장되어 있다. 각 트랜스코딩 시스템은 100Mbps 스위칭 허브에 연결되고 각 클러스터 시스템의 하드웨어 사양은 표 6, 7, 8과 같다.

CPU	Intel(R) Celeron(R) CPU 1.60GHz
메모리	256M SDRAM
운영체제	RedHat 7.3 (Kernel 2.4.18)
네트워크	100Mbps fast-ehternet

표 7 클러스터 1호 트랜스코딩 서버 × 8

CPU	Intel(R) Celeron(R) CPU 2.00GHz
메모리	1GB SDRAM
운영체제	RedHat 7.3 (Kernel 2.4.18)
네트워크	100Mbps fast-ehternet

표 8 클러스터 2호 트랜스코딩 서버 × 8

CPU	CPU : AMD 애슬론 MP 써러브레드 2200+ Dual
메모리	1GB DDR
운영체제	RedHat 7.3 (Kernel 2.4.18)
네트워크	100Mbps fast-ehternet

표 9 클러스터 3호 트랜스코딩 서버 × 7

2.2 성능 측정 방법

성능 측정은 자체 제작한 모니터링 프로그램을 이용하여 측정하였다. 하나의 클라이언트는 실제 트랜스코딩 서비스를 받아 영상을 상영하고 또 다른 클라이언트는 가상의 부하를 발생시켜 분배서버에 요청을 하게 된다. 트랜스코딩 서버는 가상요청인 경우와 실제 요청인 경우를 구조체 정보를 이용하여 구분한다. 가상 요청일 경우에는 트랜스코딩된 미디어 스트리밍을 부하 발생 클라이언트에서 빈 버퍼로 읽어 들여 미디어스트림을 소모시키고 실제 요청인 경우는 클라이언트에서 영화를 재생시킨다.

2.3 성능 측정 결과

2.3.1 동종 서버 시스템

트랜스코딩 서버의 구성 형태가 같은 기종의 형태로 구성되었을 경우를 측정해 보았다. 시스템 환경은 표7의 클러스터 3호기를 사용하였으며 트랜스코딩 서버 5대와 분배서버 1대 클라이언트 2대로 8대의 시스템을 구성하였다.

등급	CPU (AMD MP 2200)	네트워크 대역폭 (100Mbps)	메모리(1GB)
SQCIF	24.92	11.9	21.35
QCIF	25.52	16.6	21.81
CIF	48.94	23.8	24.09
4CIF	0.6	47.6	32.72

표 10 클러스터 3호기 트랜스코딩 서버 가중치

표 9는 클러스터 3호기의 서버 가중치를 표3 자원가중치 계산 알고리즘을 이용하여 계산한 값이다. 등급이 올라갈수록 CPU, 네트워크 대역폭, 메모리가 증가한다는 것을 알 수가 있는데 4CIF 급에서 CPU자원이 0.6 인 이유는 트랜스코딩 서버에 저장되는 원본 미디어 데이터는 4CIF 급이다. 따라서 클라이언트에서 4CIF급의 영화 요청이 들어왔을 경우에는 4CIF급으로 트랜스코딩하는 것이 아니라 바로 직접 스트리밍 서비스를 하기 때문에 CPU의 부하 발생량이 적다.

RR, DWRR, RWRR 방식의 3가지의 알고리즘을 구현하여 측정하였고 표 5에 있는 10편의 영화를 사용하였다. 부하 발생 횟수는 총 50개의 클라이언트 부하를 발생 시켰고 5대의 서버에 각각 10개의 트랜스코딩 부하를 발생하도록 하였다. 부하 발생은 표4에 있는 부하 발생 알고리즘 형태로 발생 시켰으며 실험 결과는 그림 18, 19, 20과 같다.

그림 18은 Round Robin 방식의 알고리즘 형태로 부하를 발생시킨 결과이다. RR 방식은 클라이언트에서 요청하는 순서대로 분배를 하는 방식이기 때문에 그림 18에서 트랜스코딩 요청 수 9번째의 CPU 사용량을 보면 트랜스 코딩 서버 E는 83% 트랜스코딩서버 D는 42%로 CPU 자원 불균형 형태를 알 수가 있다.

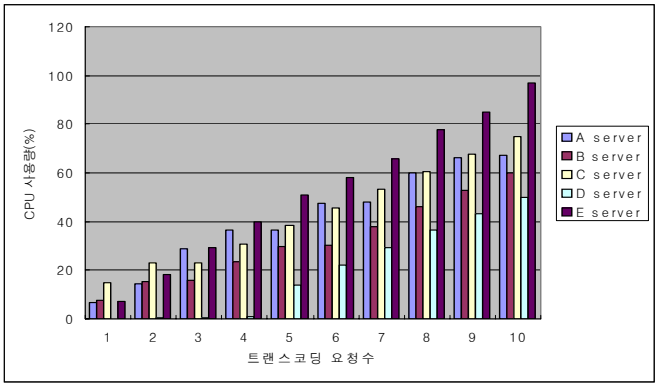

그림 18 Round Robin 알고리즘

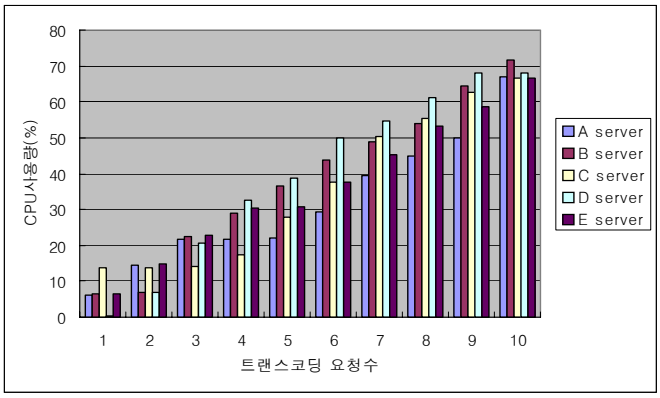

그림 19 Dynamic Weight Round Robin 알고리즘

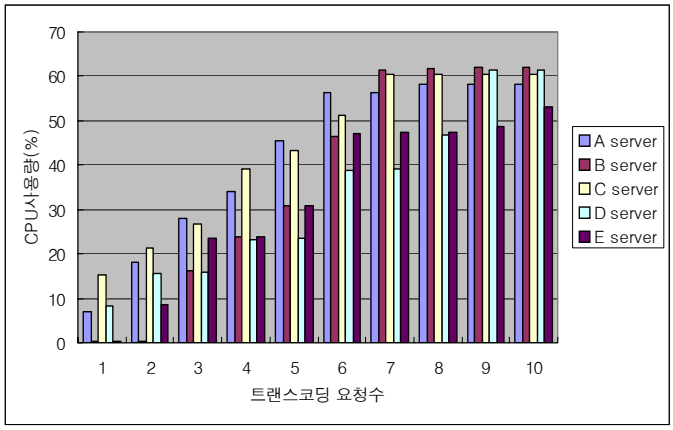

그림 20 Resource Weight Round Robin 알고리즘

RR방식은 서버의 상태를 파악하지 않고 바로 순서대로 분배를 하기 때문에 한쪽 서버에는 SQCIF급의 요청만 다른 서버에서는 4CIF의 트랜스코딩 요청이 몰릴 수 있다. 이를 해결하기 위해 WRR방식의 알고리즘이 있는데 기존의 WRR방식의 알고리즘의 경우는 시스템 성능에 따라 가중치를 주는 방법으로 동종 시스템의 서버의 경우 가중치가 같아 RR과 차이점이 없게 된다.

DWRR 방식에는 2가지가 있는데 하나는 주기적으로 보내는 방식과 다른 하나는 서버에 요청이 들어올 경우에만 상태정보를 분배서버에 보내는 방식이 있다. 서버에 요청이 들어올 경우에 보내는 방식은 트랜스코딩 서버가 다운되었을 경우 그것을 인지하는 지연시간이 발생할 수 있고 클라이언트 연결 요청 횟수가 늘어나면 서버와의 연결에 따른 부하가 발생할 수 있다[10]. 따라서 본 논문에서는 1초마다 서버 상태를 보내는 방식을 채택하였다.

그림 19은 DWRR알고리즘 방식이다. 표 10은 분배서버에서의 부하분산 테이블 관리 방식이다. 1초마다 트랜스코딩 서버에서의 CPU 사용량을 측정하여 분배서버에서 부하분산테이블을 관리를 한다. 분배서버에서는 5개 트랜스코딩 서버의 CPU 사용량을 1초마다 측정하여 클라이언트 요청이 들어왔을 경우 가장 적은 CPU사용량을 가진 서버에 부하를 보내게 된다. 표 10에서 요청3이 들어왔을 경우를 살펴보면 Server E의 CPU 사용량이 3.5%로 가장 낮다. 따라서 분배서버에서는 CPU 사용량이 가장 낮은 Server E를 선택하게 된다.

구분	분배서버	Server A	Server B	Server C	Server D	Server E
요청1	A	3.2%	3.4%	3.6%	4.2%	3.5%
요청2	B	23.6%	3.4%	3.6%	4.2%	3.5%
요청3	C	23.6%	25.4%	3.6%	4.2%	3.5%
요청4	E	23.6%	25.4%	3.6%	4.2%	44.7%
요청5	C	23.6%	25.4%	23.6%	4.2%	44.7%

표 11 DWRR 알고리즘 부하분산 (CPU 사용량)

DWRR 알고리즘은 트랜스 코딩 서버에서 정확한 정보를 측정하기 때문에 3가지 알고리즘 중에 가장 좋은 성능을 보여주고 있다. 그림 19의 1번째 클라이언트 요청이 들어왔을 경우 서버마다 CPU 소모량이 크게 차이가 나는 것은 부하 발생이 랜덤하게 발생하며 SQCIF, QCIF, CIF, 4CIF 급의 요청이 서버에 각각 처음 발생하였을 경우 CPU의 불균형이 일어나게 된다. 하지만 요청횟수가 늘어날수록 서버의 부하 분산이 효율적으로 일어난다는 것을 알 수가 있다.

그림 20은 RWRR 알고리즘 방식이다. RWRR 알고리즘은 DWRR 알고리즘의 성능 확장성을 개선하기 위해 고안된 알고리즘이다. 서버의 가중치를 이용하여 부하 분산을 하기 때문에 RR방식보다 부하분산이 효율적으로 이루어지고 있다는 것을 볼 수가 있다. 그림 20의 5번째 클라이언트 요청 수에서 Server A의 CPU 사용량은 46%이고 Server E는 23%로 CPU의 자원 불균형 현상이 있는데 이는 서버의 가중치를 고정된 수치로 계산하여 부하를 분배했을 경우 트랜스코딩 특성에 따른 CPU변화율 상황을 정확히 반영하지 못하고 있음을 나타낸다. 이러한 문제점을 해결하기 위하여 클라이언트 접속이 끊기거나 트랜스코딩 작업이 완료되었을 때 서버의 상태정보를 보냄으로서 부하분산의 불균형 문제를 보완할 수 있다.

RR, DWRR, RWRR에서 실험에서 관측된 각각의 서버노드들 사이의 부하 불균형 정도를 살펴보면 RR의 경우 요청수 10에서 CPU사용량이 최대와 최소 사이가 55%, DWRR 경우 6%, RWRR 경우 9%로 DWRR, RWRR방식이 효율적 부하분산을 한다는 것을 알 수 있다.

2.3.2 이종 서버 시스템

서버의 구성 형태가 다를 경우 RR, DWRR, RWRR 알고리즘 방식을 측정하였다. 클러스터 1,2,3호기를 이용하여 측정하였고 24대의 트랜스코딩 서버를 사용하였다. 서버들의 사양은 표 6, 7, 8과 같고 부하 분배는 동종 서버일 경우와 동일하다.

그림 21, 22, 23의 실험 결과는 클러스터 1, 2, 3호기 중에서 각각 2대의 서버 자원현황을 그래프로 나타낸 것이다. 표 11은 각 클러스터 시스템에서의 최대 클라이언트 수를 측정한 결과이다. 표 11의 한 칸의 항목은 1개의 서버에서 각 등급별로 트랜스코딩한 개수이다.

등급	클러스터1 (노드×8)	클러스터2 (노드×8)	클러스터3 (노드×7)
SQCIF	8	9	14
QCIF	7	8	12
CIF	5	6	9

표 12 등급별 가능한 최대 클라이언트 수

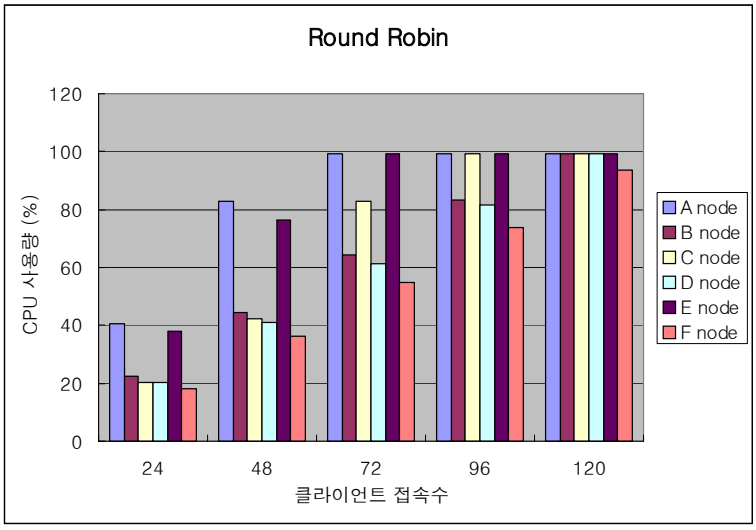

그림 21 이종 서버간의 Round Robin 방식

그림 21는 이종 서버환경간의 Round Robin 방식의 그래프를 나타낸 것이다. A, B 는 클러스터1, C,D는 클러스터2, E, F는 클러스터3의 서버들이다. 이종 환경간의 CPU사용량 분포를 살펴보면 A서버와 E서버에 부하분산이 크게 불균형하다는 것을 알 수가 있다. 이는 이종서버 환경일 경우 서버의 가중치 계산이 틀려지기 때문에 서버가중치가 동등한 동종서버 환경일 때보다 부하분산의 불균형을 가져올 수 있다는 것을 보여준다.

그림 22의 경우는 이종 서버간의 DWRR 가중치 알고리즘 측정 결과이다. 이종 서버환경에서는 DWRR 알고리즘이 최고의 부하 분산 성능을 나타내 주고 있다. 하지만 주기적으로 분배서버에 서버의 상태 정보를 보내주어야 하기 때문에 서버가 크게 늘어나게 되면 분배서버의 네트워크 대역폭이 줄어든다는 단점이 있어 확장성에 문제가 생기게 된다.

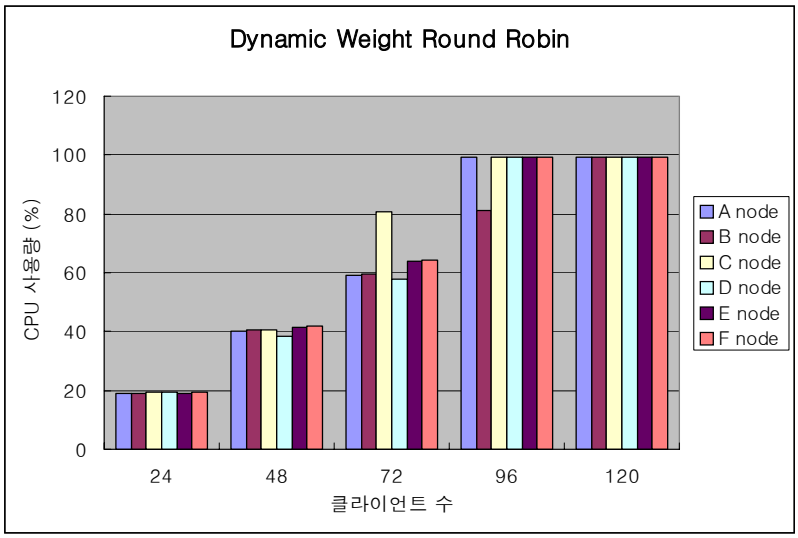

그림 22 이종 서버간의 DWRR 방식

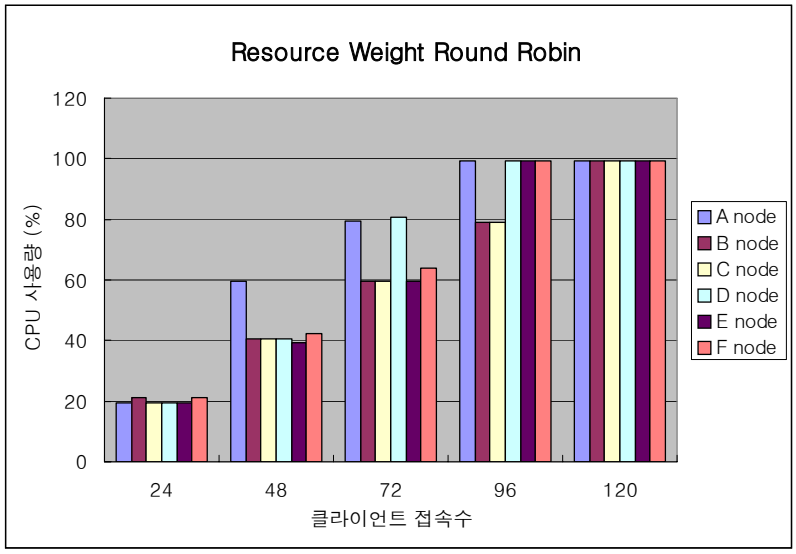

그림 23 이종 서버간의 RWRR 방식

그림 23은 이종 서버간의 RWRR 알고리즘 방식의 그래프를 나타낸 것이다. 클라이언트 48번째에서 A서버의 부하 불균형이 일어나는 것을 볼 수가 있는데 이는 이종 서버 환경에서 자원가중치 값이 서버의 성능에 따라 각각 틀리게 때문에 CPU 변화량을 정확히 반영하지 못한다는 것을 보여주고 있다. CPU 변화량을 보다 정확히 반영하기 위해서 본 논문에서는 클라이언트의 접속이 끊기거나 트랜스코딩이 완료된 경우 서버의 상태 정보를 분배서버에 보내주어서 이러한 부하 불균형 문제를 보완하였다.

2.3.3 성능 확장성

그림 24는 클러스터2호기 노드1대의 트랜스코딩시 CPU 사용량을 보여주고 있다. 10초 간격으로 cif급에서 sqcif급으로 트랜스코딩하는 총 9개의 부하를 발생시켰다. 그림 24의 그래프를 보면 처음 부하 발생시 트랜

스코딩에 따른 CPU 사용량은 18%를 사용을 하고 있으나 부하가 연속적으로 발생함에 따라 점차적으로 CPU 점유율이 11.6%에 수렴한다는 것을 알 수가 있다.

그림 25는 네트워크 사용량을 보여주고 있다. 부하 발생을 8개까지 발생시켰을 겨우 트랜스코딩 서버에서 정상적인 트랜스코딩 처리를 하여 네트워크 점유율이 90kbps로 일정하다가 9번째 부하 발생시 네트워크의 사용량이 줄어들고 있음을 볼 수 있다.

그림24, 25를 비교해 보면 8번째 부하 발생까지 네트워크 사용량이 일정하다가 시간 350초 부근에서 9번째 부하 발생시 네트워크 사용량이 90kbps에서 80kbps로 줄어들고 있다는 것을 볼 수 있다. 이는 8번째 부하 발생까지 CPU에서 트랜스코딩을 정상적으로 처리를 하여 출력 비트율이 정상적으로 나오지만 9번째 부하가 발생하였을 경우 CPU가 정상적으로 트랜스코딩을 하지 못하여 출력 비트율이 낮아짐을 알 수가 있다. 따라서 트랜스코딩 서버의 cif급에서 sqcif으로 트랜스코딩은 최대 처리량이 8개라는 것을 알 수가 있다. 이는 CPU사용량이 99%가 넘어가는 부하발생에서도 정상적인 트랜스코딩이 가능하다는 것을 보여주고 있다.

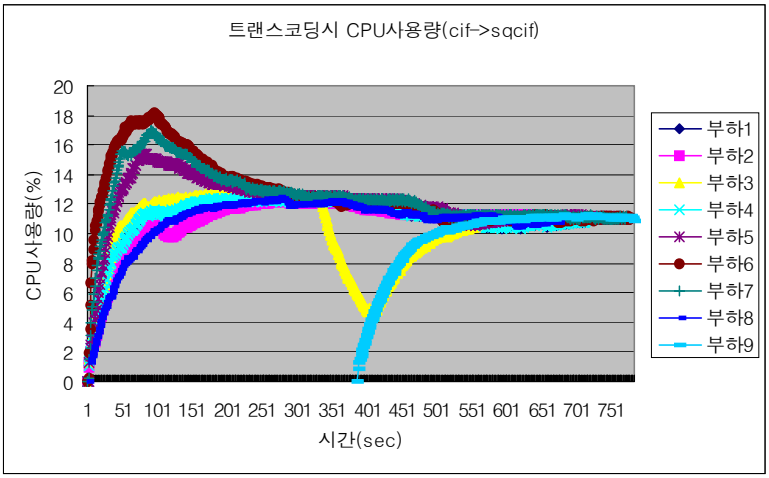

그림 24 트랜스코딩 CPU사용량

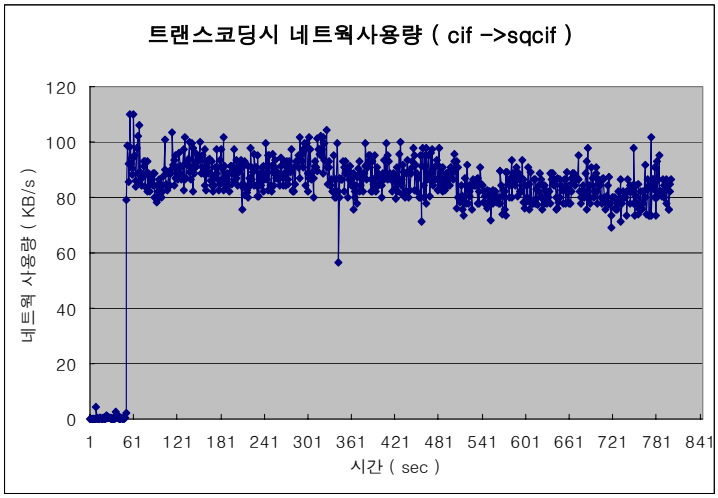

그림 25 트랜스코딩 Network 사용량

그림 26는 이종서버로 구성된 트랜스코딩 서버의 클라이언트 총 접속수 이다. 총 21대의 서버로 구성되어 있으며 트랜스코딩서버 1~7은 클러스터1호기 8~16은 클러스터2호기 17~23은 클러스터3호기로 구성되었다. 부하는 총 294개의 부하를 발생시켰고 Zipf 분포를 따라 등급 분포는 SQCIF 143개, QCIF 86개, CIF 64로 발생되었다.

그림 26를 보면 트랜스코딩 서버가 증가할수록 클라이언트를 받아들일 수 있는 총 사용자 수는 늘어나고 있다는 것을 알 수 있다. 17~23번의 서버에서 그래프의 기울기가 크게 증가하는 것을 볼 수 있는데 클러스터 3호기의 사양이 높아 클라이언트 처리량이 높아졌기 때문이다. RWRR이 가장좋은 성능을 보여주고 있으며 DRWW이 가장 낮은 성능을 보여주고 있다. DWRR이 성능이 가장 떨어지는 이유는 트랜스코딩에 사용되는 CPU사용량과 사용자의 미디어 트랜스코딩 요구를 만족시킬 수 있는 최소한의 CPU사용량과는 차이가 있기 때문이다. RWRR은 트랜스코딩에 따른 필요한 CPU사용량을 측정하여 가중치 테이블을 계산하기 때문에 이

를 이용해 효율적인 부하를 분산한다는 것을 그래프를 통해 알 수 있다.

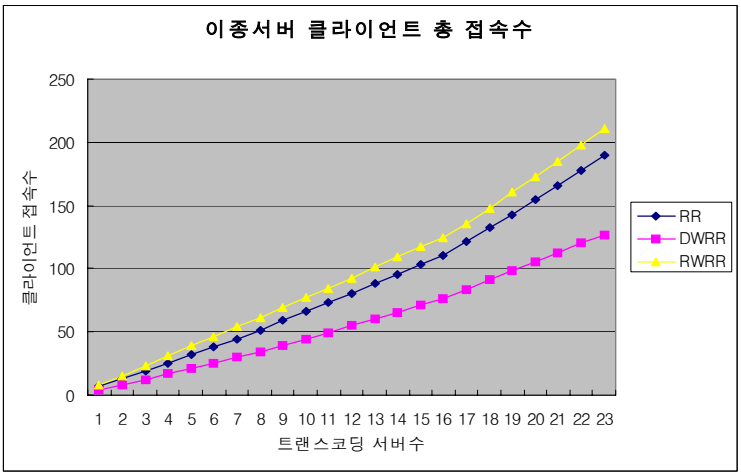

그림 26 이종서버 구조에서 최대 클라이언트 수

Ⅵ. 결론 및 향후 연구

다양한 형태의 클라이언트 무선단말기들이 스트리밍 서비스와 같은 고품질의 멀티미디어 데이터를 안정적으로 전송하고 처리하게 하기 위한 트랜스코딩에 대한 연구가 활발하다. 최근 트랜스코딩에 대한 연구는 사용자가 유선망에 존재하는 서비스들을 무선망에서 원하는 수준으로 제공받을 수 있도록 하고 있다.

본 논문에서는 리눅스 기반의 클러스터 시스템에서 미디어 트랜스 코딩에 대한 연구를 진행하였다. 클러스터 시스템의 효율적인 부하 분산을 위해 자신의 정보를 주기적으로 분배서버에 보내도록 하는 모니터링 프로그램을 구현하였고 분배서버에서 모니터링 프로그램을 이용하여 부하 분산을 하는 DWRR 알고리즘을 구현하였다. 또한 정적 알고리즘인 RR방식과 자원의 가중치 테이블을 이용하여 부하를 분산하는 RWRR알고리즘 방식을 구현하였다.

부하분산 실험 결과 본 논문에서 구현한 RWRR방식과 RR방식의 클러스터 트랜스코딩 시스템이 부하분산에 효율적이라는 것을 알 수 있었으며 DWRR 방식은 CPU 부하 분산에 있어서 실제 서버정보를 이용하여 분배를 하기 때문에 가장 좋은 부하분산을 나타내었다. 그러나 실험을 통한 결과 CPU의 사용율이 99%가 넘어서도 정상적인 트랜스코딩이 가능하기 때문에 CPU 사용율을 이용한 DWRR 방식에서는 성능의 확장성에 있어서 커다란 제약이 따른다. 따라서 이를 보완한 자원가중치 기반의 RWRR 알고리즘 방식은 트랜스코딩시 필요한 최소 CPU 사용율을 이용

한 자원 가중치 테이블로 부하를 분산하여 트랜스코딩시스템 성능의 확장성을 크게 높였다. RWRR 방식에서 자원 가중치 테이블만으로 부하를 분산하면 스트리밍 특성에 따라 트랜스코딩시 자원소모율이 조금씩 달라지기 때문에 최적의 부하 분산을 하는데 어려움이 있었다. 이를 보안하여 트랜스코딩 작업이 끝나거나 클라이언트 접속이 끊겼을 경우 서버의 상태정보를 분배 서버에 보냄으로서 단점을 보안하였고 부하테이블과 실제 서버의 상태정보를 이용하여 서버 부하측정의 정확성을 높임으로서 효율적으로 부하가 분산되는 것을 실험을 통하여 확인했다.

참 고 문 헌

[1] H.Bhradvaj, A. Joshi and S. Auephanwiriyakul. "An active transcoding proxy to support mobile web access." In Proceedings of International Conference on Reliable Distrubuted System, pp 118-123, 1998.

[2] Vetro. A.; Sun, H., "Media Conversions to Support Mobile Users", IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), May 2001

[3] http://www.ieee802.org

[4] http://www.mpeg.org

[5] Sumit Roy, Michele Covell, John Ankcorn, and Susie Wee, "A System Architecture for Managing Mobile Streaming Media Services", Takeshi Yoshimura Streaming Media Systems Group, Hewlett-Packard Laboratories, Palo Alto, CA 94304

[6] J. Song, E.Levy, A.Iyengar, and D. Dias, Design alternatives for scalable web server accelerators, Proceedings of the 2000 IEEE International Symposium on Performance Analysis of Systems and Software(ISPASS),2000.

[7] 최면욱, 정인범, 현종웅(KAIST): 클러스터 웹 서버에서 컨텐츠인식 부하 분산 모델, 한국정보과학회 2003년10월 추계학술발표논문집, 제30권 2호, pp.421 ~ 423

[8] http://ffmpeg.sourceforge.net

[9] C.C.Aggarwal, J.L.Wolf, and P.S.Yu, "On optimal batching policies for viedo-on-demand storage servers," Proc. of IEEE ICMCS'96, pp.253-258, Hiroshima, Japan, June 1996

[10] Surendar Chandra, Carla Schlatter Ellis and Amin Vahdat, "Differentiated Multimedia Web Services Using Quality Aware Transcoding", INFOCOMM 2000

Effective Media Transcoding Load Distribution Policy in the Cluster System

Choi-Myun Uk

Department of Computer, Information and Telecommunications Graduate Engineering School of Kangwon National University

Abstract

In this paper, We proposed a effective media transcoding load distribution strategy. By means of load estimate using load metrics measured in the load balance server, the load balance server with our strategy more timely and accurately estimates server's load without the help of back-end servers. As a result, it can properly react to load imbalance among the servers under transcoding workloads. Our experimental results demonstrate substantial performance improvements over other load balancing strategies used in the transcoding system.