Journal of Korea Multimedia Society Vol. 9, No. 7, July 2006(pp. 904-913)

# VoD 시스템을 위한 동적 작업부하조정기 모델

박정윤[†], 김중환[††], 김상철[†††]


## 요 약

VoD(Video on Demand) 시스템의 QoS(Quality of Service)는 클라이언트들의 스트림 서비스 요청이 시스템으로부터 거부되거나 서비스 도중에 손실되는 장애현상의 발생을 줄임으로써 향상시킬 수 있다. 본 연구에서는 VoD 시스템의 작업과정을 주기적으로 모니터링하며, 동적으로 작업부하(workload)를 조정하여 장애현상의 발생을 줄임으로써 성공적인 스트림 서비스 처리량(throughput)을 증가시킬 수 있는 동적 작업부하조정기(Dynamic Workload Balancer) 모델을 제안하였다. 제안된 모델은 에이전트개념을 사용하여 VoD 시스템과는 서로 독립적으로 작동할 수 있게 하였으며, VoD 시스템과 인터페이스를 하는 에이전시(Agency) 부분과 작업부하 조정에 필요한 조치를 추론하거나 학습하는 인텔리전스(Intelligence) 부분으로 구성된다. 그리고 제안된 모델을 VoD 시스템에 적용하는 경우에 실효성이 있는지를 시뮬레이터를 구현하여 실험하였다. 실험의 결과, 제안된 모델을 적용하면 기존의 방법을 적용한 경우보다 상대적으로 QoS가 향상된다는 것을 알 수 있었다.


# A Model of Dynamic Workload Balancer for VoD Systems

Jeong Yun Park[†], Joong Hwan Kim[††], Sangchul Kim[†††]


## ABSTRACT

The QoS(Quality of Service) of VoD(Video on Demand) systems can be improved by reducing the faults that the stream service request from clients is rejected by the VoD server or lost during the service. This paper proposes a model of Dynamic Workload Balancer that can adjust the workload dynamically through the periodical monitoring of the VoD system, and so increase the throughput. The proposed model runs independently from the VoD system by adopting the agent concept. The model comprises the agency part interfacing the VoD system and the intelligence part reasoning or learning the facts required for the adjustment of workload. An experiment of implementing a simulator was conducted to see whether or not application of the proposed model to VoD system is effective. As a result of the experiment, it was found that the throughput was relatively increased when the proposed model was applied compared to existing means.


**Key words:** VoD, QoS, Dynamic Workload Balancer(동적 작업부하조정기)

## 1. 서 론

VoD(Video on Demand) 서비스는 클라이언트들이 VoD 시스템에 있는 제한된 자원을 공유하면서 이루어 진다. 일반적으로 가용 자원은 항상 제한되어 있기 때문에 VoD 시스템에 부여되는 작업부하가 증가하면, 클라이언트가 요구한 스트림 서비스 요청이 거부되거나 서비스 도중에 손실되는 장애현상이 발

※ 교신저자(Corresponding Author) : 박정윤, 주소 : 경기도 용인시 처인구 모현면 왕신리 산 89(449-791), 전화 : 02)953-4021, FAX : 031)333-5472, E-mail : jypark@hufs.ac.kr
접수일 : 2005년 6월 9일, 완료일 : 2006년 3월 6일
[†] 한국외국어대학교 컴퓨터공학과
[††] 한국외국어대학교 컴퓨터공학과 교수 (E-mail : jhkim@hufs.ac.kr)
[†††] 한국외국어대학교 컴퓨터공학과 교수 (E-mail : kimsa@hufs.ac.kr)
※ 본 논문은 2006년도 한국외국어대학교 교내연구비 지원에 의하여 연구되었음.

생하게 된다. 이는 VoD 시스템의 QoS(Quality of Service)를 저하시키는 가장 큰 요인이 된다[1,2]. 따라서 이와 같은 운용환경에서 가능하면 QoS의 저하를 적게 하기 위해서는 주어진 자원을 효율적으로 공유할 수 있게 하여야 한다[3-6]. 이러한 문제를 해결하기 위한 연구는 VoD 서버에 요구되는 요청들을 스케줄링하는 방법과 파일 저장장소로부터 스트림 데이터를 제공할 때 버퍼링을 하는 방법으로 구분할 수 있다. 요구되는 요청을 스케줄링하는 방법은 VoD 시스템의 자원을 요청들이 가장 효율적으로 공용할 수 있게 처리 순서를 조정하는 방법이며, 대표적인 스케줄링 정책으로는 브릿징(bridging), 피기백킹(piggybacking), 배칭(batching) 등이 있다[7]. 파일 저장장소로부터 스트림 데이터를 제공할 때 버퍼링을 하는 방법은 주로 효율적으로 버퍼를 관리하는 방법에 대한 연구들이다. 일반적으로 버퍼를 관리하는 방법으로는 버퍼가 필요할 때 마다 교체하거나 선 인출하는 방법을 사용한다. VoD 서버에 적용하는 대표적인 방법으로는 BASIC/DISTANCE, Interval Caching, 정적 버퍼 분할 방법, 적응 버퍼 분할 방법 등이 있다[8]. 그리고 VoD 서버의 버퍼가 아니라 각 노드의 버퍼를 이용하여 버퍼의 활용률을 높일 수 있는 동적 버퍼 분할 방법이 있다[9].

본 연구에서도 기본적으로 작업부하를 조정하기 위하여 VoD 서버에 요구되는 요청을 스케줄링하는 방법을 사용한다. 그러나, 기존의 연구들과는 달리 매 주기시간 마다 VoD 서버 및 파일 저장장소의 요청을 동시에 동적으로 스케줄링하며, VoD 서버가 파일 저장장소에 요청을 연결할 때 작업부하가 효율적으로 조정되도록 최적화 과정을 실행한다. 즉, VoD 시스템의 VoD 서버와 파일 저장장소를 클라이언트들이 효율적으로 공유하도록 작업부하를 적절히 조정한다. 이와 같은 처리는 주기적으로 VoD 시스템의 작업과정을 모니터링하여 필요한 데이터를 수집하고 분석하여 작업부하 조정에 필요한 조치를 추론하며, 이를 동적으로 실행해야 한다. 이와 같은 개념을 바탕으로 제안된 동적 작업부하조정기(Dynamic Workload Balancer) 모델은 에이전트개념[10]을 사용하여 VoD 시스템과는 서로 독립적으로 작동할 수 있게 하였으며, VoD 시스템과 인터페이스를 하는 에이전시(Agency) 부분과 작업부하 조정에 필요한 조치를 추론하거나 학습하는 인텔리전스(Intelligence) 부분으로 구성된다[11-14].

VoD 시스템의 VoD 서버는 클라이언트들로부터 스트림 서비스 요청을 받으면 필요한 데이터가 있는 파일 저장장소로부터 데이터를 제공 받아 처리하게 된다. 이때 VoD 서버가 동시에 처리 할 수 있는 최대 허용 스트림의 수는 제한되어 있기 때문에 바로 서비스를 받지 못하면, 클라이언트들의 요청은 VoD 서버의 서비스를 받기 위해 대기상태에 있다가 처리된다. 또한 파일 저장장소로부터 데이터를 제공 받을 때도 파일 저장장소의 제한된 처리 능력과 요청들 사이의 저장장소 선택 경쟁에 의한 간섭 효과 때문에 바로 서비스를 받지 못하면 각 파일 저장장소의 연결 대기상태에 있다가 처리된다. 이와 같은 처리과정은 논리적으로 대기행렬 시스템으로 모델링 할 수 있다[15]. 본 연구에서 제안된 동적 작업부하조정기 모델은 대기행렬 시스템 모델을 기반으로 한다. 즉, VoD 서버와 각 파일 저장장소에는 대기행렬이 있으며, 대기행렬에서 대기하고 있는 요청들의 대기 순서를 일정한 주기시간 마다 조정하여 평균 대기시간을 최적화 한다. 그리고 VoD 서버가 요청을 파일 저장장소에 연결할 때 동일한 데이터를 가지고 있는 파일 저장장소 중에서 바로 전 주기시간 동안 대기행렬에서의 평균 대기시간이 최소인 파일 저장장소에 연결한다. 이와 같은 방법은 결과적으로 VoD 서버와 파일 저장장소에서의 평균 대기시간을 감소시켜서 성공적인 스트림 서비스 처리량을 증가시키게 된다. 제안된 모델에서는 이와 같은 처리를 추론 모듈과 지식 베이스로 구성된 인텔리전스 부분이 최적화 과정을 수행할 조치를 생성하면, 데이터 수집 모듈과 실행 모듈로 구성된 에이전시 부분이 VoD 시스템과 인터페이스를 하며 실행을 한다.

동적 작업부하조정기 모델은 워크스테이션급 서버 시스템에서 운용 가능하게 제안되었다. 그리고 VoD 시스템에 제안된 모델을 적용하는 경우에 실효성이 있는지를 실험하였다. 실험은 제한된 자원을 가진 VoD 시스템을 설정하여 VoD 시스템 및 동적 작업부하조정기의 기능 처리 과정을 시뮬레이션 하는 방법을 사용하였다. 실험을 위해 구현한 시뮬레이터는 요청계수기, VoD 서버 기능처리기, 파일 저장장소 기능처리기로 구성된다. 구현한 시뮬레이터로 실험을 한 결과, 동적 작업부하조정기 모델을 적용하였을 경우에 VoD 서버에 요구되는 요청을 스케줄링하는 기존의 방법을 적용한 경우보다 상대적으로 QoS가 향상된다는 것을 알 수 있었다.

## 2. 동적 작업부하조정기 모델

### 2.1 모델의 구성

동적 작업부하조정기 모델은 일반적인 지능형 에이전트 시스템(intelligent agent system)과 같이 정보나 지식을 해석하여 추론하거나 학습할 수 있는 인텔리전스 부분과 스스로 환경의 변화에 따라 적절한 조치를 취할 수 있는 에이전시 부분으로 구성되어 있다. 그림 1은 동적 작업부하조정기 모델의 구성도이다. 데이터 수집 모듈의 데이터 수집 엔진은 VoD 서버와 파일 저장장소로부터 최적화 과정에 필요한 데이터를 수집하고 분석하여 성능 분석 데이터베이스에 저장한다. 추론 엔진은 성능 분석 데이터베이스의 데이터를 사용하여 최적화 과정에 필요한 조치를 지식 베이스에 저장된 규칙에 따라 생성하여 실행모듈로 전달한다. 그리고 생성된 새로운 조치는 학습 엔진에 의해 학습되어 지식 베이스에 다시 저장한다.

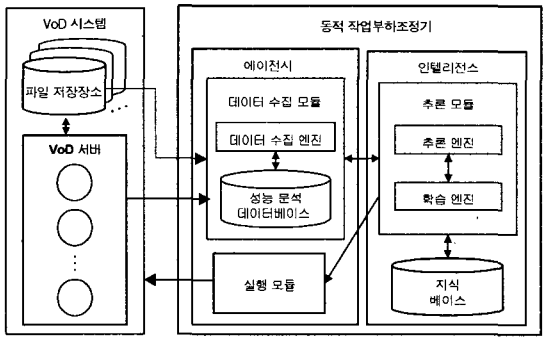


그림 1. 동적 작업부하조정기 모델의 구성도

### 2.2 에이전시

(1) 데이터 수집 모듈

동적 작업부하조정기 모델의 데이터 수집 엔진은 클라이언트의 스트림 서비스 요청이 VoD 서버 및 파일 저장장소에서 처리되는 과정에서 발생하는 아래와 같은 이벤트로 VoD 시스템의 상태변화를 파악하여 필요한 데이터를 수집한다.

- OnVServerConnect : 요청이 VoD 서버에 연결.
- OnVServerRejecct : 요청이 VoD 서버로부터 연결 거부.
- OnVServerSeize : 요청이 VoD 서버의 프로세서를 점유.
- OnFStorageConnect : 요청이 파일 저장장소에 연결.
- OnFStorageSeize : 파일 저장장소 점유.
- OnRequestLoss : 요청의 손실.
- OnResourcesRelease : 파일 저장장소 및 VoD 서버의 프로세서의 점유를 해제.

이와 같은 이벤트들을 이용하여 표 1과 같은 분석

표 1. 분석에 필요한 데이터의 수집

```
' Called when a client connects to the VoD server.
Sub NSUnicastMgr1_OnVserverConnect(datetime,
status, clientid, ipaddress, port)
    :
end sub  //  TA

' Called when a client rejected from a VoD server.
Sub    NSUnicastMgr1_OnVserverReject(datetime,
status, clientid, ipaddress, port)
    :
end sub  // 클라이언트의 요청이 받아들여지지 않은
경우

' Called when a VoD server seized and starts to
playing a file
Sub    NSUnicastMgr1_OnVserverSeize(datetime,
status, clientid, filename)
    :
end sub  // TS

' Called when a request connects to the file storage.
Sub OnFStorageConnect(datetime, status, clientid,
ipaddress)
    :
end sub  //  FTA

' Called when a file storage seized and starts to
playing a file
Sub OnFStorageSeize(datetime, status, clientid, fil-
ename)
:
end sub  // FTS

' Called when a request losses from a file storage.
Sub OnRequestLoss(datetime, status, clientid, ipad-
dress)
    :
end sub  // 요청이 손실된 경우

' Called when a service done
Sub
NSUnicastMgr1_OnResourcesRelease(datetime,
status, clientid, filename )
    :
end sub  //  TE, FTE
```

에 필요한 데이터를 수집하고, 수집된 데이터들은 그림 2와 같은 테이블 형태로 성능 분석 데이터베이스에 저장된다. 그림 2에서 stream_data 테이블은 스트림을 구분하는 s_id, 스트림의 제목인 title과 스트림이 어느 파일 저장장소에 위치해 있는지를 나타내는 location 필드로 구성된다. vsrv_data테이블은 요청을 한 클라이언트를 구분하는 c_id, 스트림을 구분하는 s_id, 클라이언트의 요청이 VoD 서버의 대기행렬에 도착한 시간 TA, 요청이 VoD 서버로부터 서비스를 받기 시작하는 시간 TS, 요청이 VoD 서버로부터 서비스 종료되는 시간 TE, VoD 서버에서 서비스 중에 있는 스트림의 개수 NS, VoD 서버가 동시에 서비스할 수 있는 최대 스트림 개수 N, VoD 서버의 대기행렬에서의 평균 대기시간 w_time, 요청이 거부될 확률 rjt_p 필드로 구성된다. 그리고 fs_data 테이블은 파일 저장장소를 구분하는 fs_id, 스트림을 구분하는 s_id, 요청이 파일 저장장소의 대기행렬에 도착한 시간 FTA, 요청이 파일 저장장소로부터 데이터를 제공받기 시작하는 시간 FTS, 요청한 데이터의 제공이 파일 저장장소로부터 종료되는 시간 FTE, 가동중인 파일 저장장소의 개수 NA, 파일 저장장소 대기행렬에서의 대기시간 fw_time, 파일 저장장소의 요청이 손실될 확률 lss_p 필드로 구성된다. stream_data 테이블의 기본 키는 s_id이며 vsrv_data 테이블의 기본 키는 c_id이고 fs_data테이블의 기본 키는 fs_id이다. 각 테이블들은 조인(join)되어 필요한 데이터만 추출할 수 있다.

(2) 실행 모듈

실행 모듈은 인텔리전스의 추론 모듈로부터 생성된 조치를 전달 받아서 VoD 시스템이 이를 실행할 수 있도록 인터페이스를 한다. 즉, 실행 모듈은 VoD 서버가 포화상태가 되어 요청이 거부될 확률이 일정

| s_id | title | location |
|---|---|---|

<stream_data 테이블>

| c_id | s_id | TA | TS | TE | NS | N | w_time | rjt_p |
|---|---|---|---|---|---|---|---|---|

<vsrv_data 테이블>

| fs_id | s_id | FTA | FTS | FTE | NA | fw_time | lss_p |
|---|---|---|---|---|---|---|---|

<fs_data 테이블>

그림 2. 성능 분석 데이터베이스의 설계

수준 이상이 되면 서비스 요청이 거부될 확률을 공지하며, VoD 서버와 각 파일 저장장소의 대기행렬에서 대기하고 있는 요청들의 대기시간이 일정한 허용한계시간(추론 엔진에서 추정한 평균 대기시간)을 초과할 가능성이 클 경우에는 추정되는 평균 대기시간을 공지한다. 그리고 VoD 서버가 요청을 파일 저장장소에 연결할 때 동일한 데이터를 가지고 있는 파일 저장장소 중에서 대기행렬의 평균 대기시간이 최소로 추정되는 파일 저장장소에 연결하는 조치를 실행하고, VoD 서버와 각 파일 저장장소의 대기행렬에 있는 요청들의 대기순서를 조정하여 평균 대기시간을 최적화하는 조치를 실행한다.

## 2.3 인텔리전스

(1) 추론 모듈

추론 모듈은 추론 엔진과 학습 엔진으로 구성된다. 추론 엔진은 데이터 수집 모듈의 성능 분석 데이터베이스의 데이터를 사용하여 지식 베이스에 정의된 규칙에 따라 VoD 서버와 각 파일 저장장소의 성능을 최적화할 수 있는 조치들을 생성한다. 이와 같은 조치들은 이벤트 공지, 요청 연결 및 요청 대기 상태의 최적화 과정으로 구분된다. 본 연구에서는 VoD 시스템의 상태를 VoD 서버의 상태와 각 파일 저장장소의 상태로 구분하여 정의하였다. 즉, VoD 서버의 상태는 동시에 서비스하고 있는 스트림의 수 $NS(0 \le NS \le N)$와 대기행렬에 있는 요청의 수 $RQ(0 \le RQ \le k)$에 의하여 구분되며, 파일 저장장소의 상태는 가동중인 파일 저장장소의 수 $NA(0 \le NA \le M)$와 i 파일 저장장소의 대기행렬에서 대기 중인 스트림의 수 $AQ_i(1 \le i \le M,\ 0 \le AQ_i < N)$에 의하여 구분된다. 여기서 VoD 서버가 임의 시점에 동시에 서비스를 하고 있는 스트림의 수 $NS = NA + \sum AQ_i$가 된다. 이와 같이 구분되는 각각의 상태와 임의 시점에서 VoD 서버와 각 파일 저장장소의 대기행렬에서의 대기시간이 일정한 허용한계시간을 초과하느냐 하지 않느냐에 따라 구분되는 상태들의 조합에 의해 규칙을 표 2와 같이 정의한다.

추론 엔진은 이러한 규칙에 따라 다음과 같은 과정을 생성하여 실행 모듈에 전달한다.

- 이벤트 공지 과정: 규칙 1과 규칙 2를 적용하는 과정이다.
- 요청 연결의 최적화 과정: VoD 서버가 요청을

표 2. 추론모듈에서 적용되는 규칙

| | |
|---|---|
| 규칙 1 | if (RQ == k)<br>서비스 요청이 거부될 확률을 공지하는 조치를 생성; |
| 규칙 2 | if ((VoD 서버의 에서의 대기시간 > 허용한계시간) OR (i파일 저장장소의 에서의 대기시간 > 허용한계시간))<br>현재의 평균 대기시간을 공지하는 조치를 생성; |
| 규칙 3 | if ($AQ_i$ > 0 AND 1 ≤ i ≤ NA) {<br>1. 일정한 주기시간마다 가동 중인 모든 파일 저장장소에 대하여 대기행렬의 평균 대기시간을 계산하여 결과를 성능 분석 데이터베이스에 저장;<br>2. VoD 서버가 요청을 파일 저장장소에 연결할 때 저장된 데이터를 사용하여 동일한 데이터를 제공할 수 있는 파일 저장장소 중에서 대기행렬의 평균 최소인 파일 저장장소로 연결시키는 조치를 생성;<br>} |
| 규칙 4 | if ((RQ > 0) OR ($AQ_i$ > 0 AND 1 ≤ i ≤ NA))<br>일정한 주기시간마다 VoD 서버와 각 파일 저장장소의 대기행렬에 있는 요청들의 대기순서를 조정하여 평균 대기시간을 최적화하는 조치를 생성; |

파일 저장장소에 연결할 때 규칙 3을 적용하는 과정이다.

- 요청 대기 상태의 최적화 과정: 일정한 주기시간마다 규칙 3과 규칙 4를 적용하는 과정이다. 대기순서를 조정하는 방법은 기본적으로 VoD 서버와 각 파일 저장장소의 대기행렬에 대하여 스트림 서비스 시간이 짧게 추정된 요청이 먼저 서비스 받을 수 있게 하고, 이에 추가하여 일정한 주기기간마다 규칙 3을 적용하여 VoD 서버의 대기행렬에 대하여는 상대적으로 평균 대기시간이 짧게 추정된 파일 저장장소의 서비스를 필요로 하는 요청이 먼저 서비스 받을 수 있도록 대기순서를 조정한다.

추론 엔진에 의해 생성된 조치들은 학습 엔진에도 전달된다. 본 연구에서는 학습 엔진을 그림 3과 같은 확률 그래프 모델로 설계 하였다. 확률 그래프 모델에서 노드는 특정한 상태나 규칙으로 구성되고, 노드와 노드사이의 연결은 이들 사이의 종속관계를 나타낸다. 이러한 확률 그래프 모델을 이용하면 부모 노드가 발생할 확률 분포를 알 때 이들의 결합 확률 분포로서 확률적으로 종속관계에 있는 자식 노드가

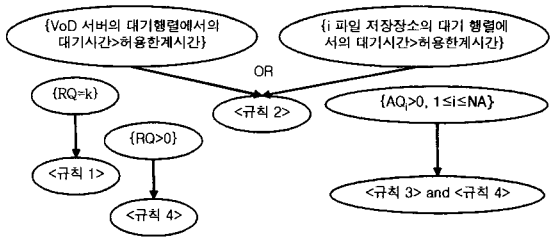


그림 3. 확률 그래프 모델

발생할 확률 분포를 구할 수 있다. 따라서 학습 엔진은 추론 엔진으로부터 생성된 조치가 전달되면, 그 조치의 생성과 연관된 상태들이 발생할 확률 분포를 구하고, 이를 이용하여 확률 그래프 모델에 따라 각 규칙을 적용할 확률 분포를 구하며, 이를 바탕으로 학습된 결과를 지식 베이스에 저장한다. 그리고 새로운 상태나 규칙이 정의되는 경우에는 확률 그래프 모델의 구성도 변화하게 된다.

(2) 지식 베이스

추론 모듈에 의해 습득된 지식은 규칙 방법을 이용하여 지식 베이스에 표현할 수 있다. 지식 베이스의 구축은 문제 영역을 VoD 서버와 파일 저장장소 두 개의 영역으로 분할하여 주어진 추론 기법[16,17]에 따라 표현하고 최종적으로 이들을 통합하는 방식을 사용한다. 추론방향은 주로 전향(forward) 추론을 사용하며, 필요에 따라 후향(backward) 또는 혼합(mixed) 추론을 사용하기도 한다. 규칙 클래스 선언을 적용한 예는 표 3과 같으며 규칙 클래스에 속하는 규칙은 표 4와 같이 정의된다.

표 3. 규칙 클래스 선언을 적용한 예

```
Rule class <waiting time> (<VoD server>)
{
comment : "VoD 서버 대기행렬에서의 대기시간
          계산"
direction : forward
}

Rule class <waiting time> (<file storage>)
{
comment : "파일 저장장소 대기행렬에서의 대기시간
          계산"
direction : forward
}
```

표 4. 규칙 클래스에 속한 규칙 정의

```
Rule Rule1
{
if : (<{RQ=k}>)
        then : (<규칙 1>)
}
Rule Rule2
{
if : (<{VoD 서버의 대기행렬에서의 대기시간>허용
      한계시간}>)
or : (<{i 파일 저장장소의 대기행렬에서의 대기시간
     >허용한계시간}>)
then : (<규칙 2>)
}
Rule Rule3
{
if : (<{AQi>0, 1≤i≤NA}>)
then : (<규칙 3>)
 }
Rule Rule4
{
if : (<{RQ>0}>)
or : (<{AQi>0, 1≤i≤NA}>)
then : (<규칙 4>)
 }
```

## 3. 실 험

### 3.1 시뮬레이션

(1) 시뮬레이션 모델

시뮬레이션 모델은 VoD 시스템이 스트림 서비스 요청을 처리하는 과정을 그림 4와 같은 시스템 모델에서 작업부하, 요청 연결의 최적화, 요청 대기상태의 최적화를 처리하는 과정으로 설정되었다.

- 작업부하 과정: 클라이언트들이 스트림 서비스를 VoD 서버에 요청하여 작업부하의 기본 단위인 트랜젝션을 생성하는 과정이다. 시뮬레이션 모델에서는 이와 같은 작업부하 과정을 트랜젝션들이 서로 독립적으로 생성된다고 가정하여 포와송 과정(Poisson process)으로 보았다. 즉, 작업부하의 기본 단위인 트랜젝션들은 서로 독립적으로 생성되고, 서비스 소요시간은 균등분포(uniform distribution)를 따르며, 생성과 생성 사이의 시간은 지수분포(exponential distribution)을 따른다고 본다. 그리고 동적 작업부하조정기에서는 VoD 서버와 파일 저장장소를 일정한 주기시간 동안 모니터링하여 데이터를 수집하고, 수집된 데이터는 다음 주기시간 동안에 발생하는 이벤트

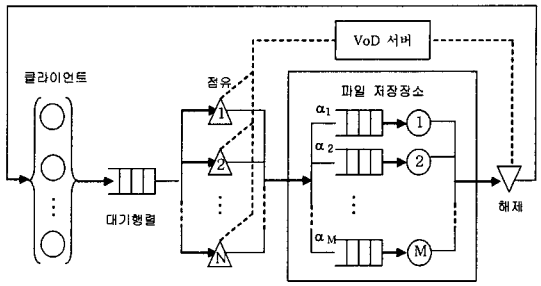


그림 4. VoD 시스템의 시스템 모델

처리에 이용된다. 이때 임의의 한 주기시간 동안의 평균 요청발생율은 주기시간 동안 발생한 총 요청의 수를 주기시간으로 나눈 값이다. 따라서 시뮬레이션 모델에서도 시스템 매개 변수로 조정할 수 있는 임의의 한 주기시간 동안의 평균 요청발생율은 일정하다고 본다. 즉, 임의의 한 주기시간 동안에 생성되는 트랜젝션의 생성과 생성 사이 시간의 분포는 평균이 일정한 지수분포를 따른다. 그리고 시뮬레이션 모델에서 데이터 수집 모듈의 기능은 그림 2에서 설계한 테이블들의 필드 값을 정의하면 아레나[18]에서 제공하는 도구에 의하여 자동으로 필요한 데이터를 수집하여 성능 분석 데이터베이스에 저장하게 된다.

- 요청 연결의 최적화 과정: 요청 연결의 최적화 과정은 다음과 같다.

단계1: 주기시간 설정;

단계2: 주어진 주기시간에 대해 파일 저장장소의 대기행렬에 있는 요청들의 평균 대기시간 계산;

단계3: 평균 대기시간 테이블에 각 파일 저장장소별로 구분하여 평균 대기시간을 저장;

단계4: while (TRUE) {
if (VoD 서버가 요청을 파일 저장장소에 연결처리가 가능) break;
}

단계5: VoD 서버 대기행렬의 맨 앞에 있는 요청이 요구하는 스트림 데이터 종류 확인;

단계6: 평균 대기시간 테이블을 참조하여 요구된 스트림 데이터 종류를 제공할 수 있는 파일 저장장소 중에서 평균 대기시간이 최소인 파일 저장장소를 찾음;

단계7: 선택된 파일 저장장소에 요청을 연결;

- 요청 대기 상태의 최적화 과정: 요청 대기상태의 최적화 과정은 다음과 같다.

단계1: 주기시간 설정;

단계2: VoD 서버와 파일 저장장소의 대기행렬에 있는 요청들을 스트림 서비스 시간이 짧게 추정된 요청이 먼저 서비스 받을 수 있게 대기순서를 조정;

단계3: 주어진 주기시간에 대하여 파일 저장장소의 대기행렬에 있는 요청들의 평균 대기시간을 계산;

단계4: 평균 대기시간 테이블에 각 파일 저장장소별로 구분하여 평균 대기시간을 저장;

단계5: 상대적으로 평균 대기시간이 짧은 파일 저장장소가 앞에 오게 평균 대기시간 테이블을 정렬;

단계6: 평균 대기시간 테이블을 참조하여 VoD 서버의 대기행렬에 있는 요청들을 평균 대기시간이 짧게 추정된 파일 저장장소의 서비스를 요구하는 요청이 먼저 서비스 받을 수 있도록 대기순서를 조정;

(2) 시뮬레이터의 구현

시뮬레이션 모델에서 설정된 작업부하, 요청대기 상태의 최적화, 요청 연결의 최적화 과정은 그림 5와 같이 요청계수기, VoD 서버 기능처리기, 파일 저장장소 기능처리기로 구성된 시뮬레이터로 구현하였다. 구현된 시뮬레이터는 추후에 확장이 용이하게 각 부 모듈들이 독립적으로 작동하며, 필요에 따라 서로 상호작용을 할 수 있게 구성하였다. 그리고 각 부 모듈들은 아레나(Arena)[18]가 제공하는 Basic Process 패널(Create 모듈, Assign 모듈, Decide 모듈, Dispose 모듈, Record 모듈, Process 모듈), Advanced Process 패널(Delay 모듈), Blocks 패널(Queue 모듈, Seize 모듈)을 사용하여 구현하였다.

- 요청계수기: 시뮬레이션이 진행되는 동안 매 주기시간마다 VoD 서버로부터 서비스 거부된 트랜잭션의 수를 파악하고 시뮬레이션이 종료되면 시뮬레이션 기간 동안 모두 몇 개의 요청이 거부되었는지를 누적한다. 요청 계수기의 처리 흐름도는 그림 6과 같다. Create Request는 시뮬레이션이 시작될 때 논리를 구현하고 시스템을 바꾸는 목적으로 참조하는 하나의 트랜잭션을 생성시킨다. 이 트랜잭션은 정해진 주기시간 동안 지연된 후 다음 주기시간을 정하고, 정해진 주기시간이 모두 지나면 소멸된다. Assign Cycle Time은 주기시간을 나타내는 변수 Cycle Time을 0으로 초기화한다. Assign Variables에서는 현재 진행되고 있는 주기시간을 나타내는 Cycle Time 변수를 Cycle Time = Cycle Time + 1이 되도록 한다. 그리고 한 주기시간 및 전체 시뮬레이션 시간 동안 거부된 요청의 수를 파악한다. Check Cycle Time은 시뮬레이션이 진행되어야 할 주기시간이 더 남아있는지를 조사하여 남아있으면 트랜잭션을 Delay for Cycle Time으로 보내 정해진 주기시간 동안 지연시키고, 그렇지 않으면 Dispose of Request로 이동하여 트랜잭션을 소멸시킨다.

- VoD 서버 기능처리기: 클라이언트의 요청을 VoD 서버가 처리하는 부분이며, 처리 흐름도는 그림 7과 같다. Create Arrivals에서는 VoD 서버에 트랜잭션의 도착과 도착 사이의 시간의 분포가 특정한 평균 요청발생율을 갖는 지수 분포를 따르도록 한다. Queue와 Seize에서는 도착한 트랜잭션을 처리 가능

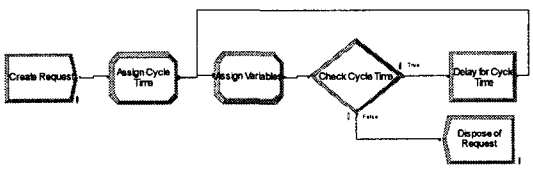


그림 6. 요청계수기의 처리 흐름도

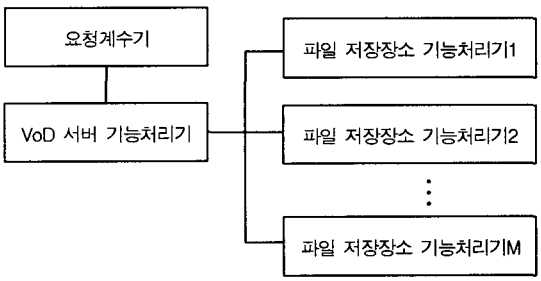


그림 5. 시뮬레이터의 구성

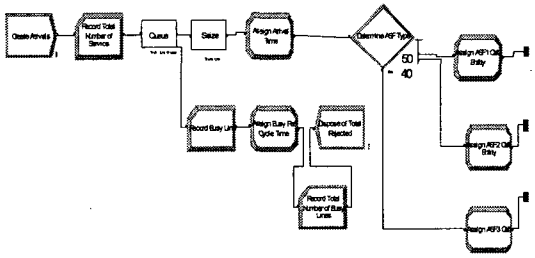


그림 7. VoD 서버 기능처리기의 처리 흐름도

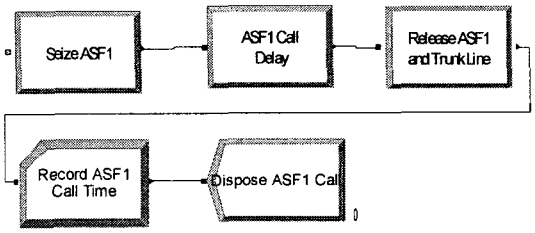


그림 8. 파일 저장장소 기능처리기의 처리 흐름도

한 프로세서가 있는지 조사하여 처리 가능하면 도착한 트랜젝션을 서비스할 프로세서에 할당한다. 그렇지 않을 경우 요청 대기 상태의 최적화 과정에서 설정된 규칙에 따라 대기행렬에서 대기하게 한다. 이때 대기행렬이 포화상태이면 트랜젝션은 소멸되며, 요청은 거부된다. Assign Arrival Time에서는 프로세서가 할당된 트랜젝션이 서비스 받기 시작하는 현재 시간을 트랜젝션에 저장한다. Determine ASF Type에서는 서비스 받을 파일 저장장소를 요청 연결의 최적화 과정에 따라 결정한다. 그리고 요청 대기 상태의 최적화 과정 중 마지막으로 Record Busy Line과 Assign Busy Per Cycle Time에서는 VoD 서버에서 거부되어 소멸된 트랜젝션의 수를 누적한다.

- 파일 저장장소 기능 처리기: 요청한 스트림 데이터를 제공하며, 처리 흐름도는 그림 8과 같다. Seize ASF은 트랜젝션이 서비스를 받기 위하여 특정한 파일 저장장소를 할당한다. 이때 할당된 파일 저장장소가 사용 중이면 할당된 파일 저장장소의 대기행렬에서 대기하며, 요청 대기 상태의 최적화 과정에서 설정된 규칙에 따라 대기순서가 조정된다. ASF Call Delay는 트랜젝션이 특정한 파일 저장장소에서 서비스 받는데 소요되는 시간을 생성한다. Release ASF and Trunk Line은 서비스가 종료되면 트랜젝션이 서비스를 받기 위해 점유했던 특정한 파일 저장장소와 VoD 서버의 프로세서를 해제한다. Record ASF Call Time은 파일 저장장소에서 필요한 데이터를 제공받는데 소요된 시간을 기록한다. Dispose ASF Call은 서비스가 끝난 트랜젝션을 소멸시킨다. 이와 같은 처리 흐름은 다수의 파일 저장장소 기능 처리기가 있는 경우에도 모두 동일하다.

## 3.2 결과 분석

동적 작업부하조정기 모델의 실효성을 검정하기 위하여 그림 4에서 $\alpha_1$=0.5, $\alpha_2$=0.4, $\alpha_3$=0.1로 추정되는 3대의 파일 저장장소가 있는 VoD 시스템에서 표 5에 주어진 운용환경으로 본 연구에서 구현한 시뮬레이터로 실험하였다.

실험은 동일한 운용환경 하에서, 본 연구에서 제안한 동적 작업부하조정기를 적용하는 경우와 VoD 서버에 요구되는 요청을 스케줄링하지 않거나 배칭(batching) 방법[19]으로 스케줄링하는 경우를 비교하였다. 배칭 방법에서는 먼저 요구된 요청을 먼저 서비스하는 FCFS 배칭 정책을 적용하였다. 그리고 배칭의 크기를 다양하게 하여 얻어진 실험 결과 중에서 가장 좋은 성능을 나타낸 것을 선택하였다. 실험의 결과 그림 9에서 보면 동적 작업부하조정기 모델을 적용하는 경우에는 평균 요청발생율이 0.06이상인 경우에 성공적인 스트림 서비스 처리량이 상대적으로 증가한다는 것을 알 수 있다. 즉 작업부하가 일정 수준 이상이 되는 VoD 시스템에 동적 작업부하조정기 모델을 적용하면 QoS 향상의 실효성이 있음을 나타낸다.

표 5. 시뮬레이션 모델의 운용환경

| | |
|---|---|
| 포와송 과정을 따르는 평균 요청발생율(분) | 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20, 0.22, 0.24, 0.26, 0.28, 0.30 |
| VoD 서버가 동시에 서비스 할 수 있는 최대 허용 스트림의 수, N | 5 |
| 대기행렬의 용량 | 10 |
| 파일 저장장소의 수, M | 3 |
| 파일 저장장소의 대기행렬 용량 | N-1 |
| 주기시간 | 10분 |
| 트랜젝션의 서비스 소요시간 분포 | 30±10인 균등분포 (Uniform Distribution) |
| 총 시뮬레이션 시간 | 1080분 |

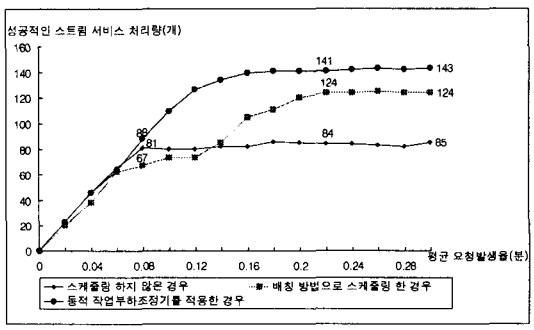


그림 9. 성공적인 스트림 서비스 처리량

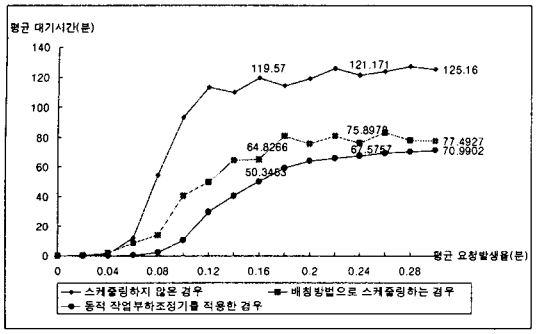


그림 10. VoD 서버에서의 평균 대기시간

성공적인 스트림 서비스 처리량이 증가한다는 것은 VoD 서버로부터 서비스 거부된 요청의 수가 감소되거나 요청이 VoD 서버 및 파일 저장장소에서 서비스를 받기 위해 대기하는 평균 대기시간이 감소된 결과이다. VoD 서버로부터 거부된 요청의 수와 VoD 서버 및 각 파일 저장장소에서의 평균 대기시간도 실험 과정에서 측정할 수 있다. 참고로 파일 저장 장소의 대기시간을 제외한 VoD 서버에서의 평균 대기시간만을 측정한 결과는 그림 10과 같다. 그림 10에서 보면, 동적 작업부하조정기를 적용하는 경우에 상대적으로 평균 대기시간이 감소함을 알 수 있다.

## 4. 결 론

VoD 시스템의 QoS를 향상시키기 위하여 본 연구에서는 VoD 시스템의 작업과정을 모니터링하여 동적으로 작업부하를 조정하는 동적 작업부하조정기 모델을 제안하였다. 제안된 동적 작업부하조정기 모델의 실효성을 검정하기 위하여 모델의 일부 기능을 실행할 수 있는 시뮬레이터를 구현하여 실험하였다. 그 결과, 동적 작업부하조정기 모델을 적용한 경우에 VoD 서버에 요구되는 요청을 스케줄링하지 않거나 배칭 방법으로 스케줄링하는 경우보다 성공적인 스트림 서비스 처리량이 상대적으로 증가한다는 것을 알 수 있었다. 따라서 동적 작업부하조정기를 VoD 시스템에 적용하면 VoD 시스템의 QoS를 향상시킬 수 있다고 본다. 앞으로 본 연구에서 제안한 모델의 설계 개념에 따라 동적 작업부하조정기를 구현하여 실제로 VoD 시스템에 적용해 보는 연구와 모델의 설계 개념을 단일 VoD 시스템에 국한하지 않고 다중 VoD 시스템의 경우로 확장하는 연구가 필요하다고 본다. 그리고 VoD 시스템에 부여되는 작업부하를 조정하기 위하여 파일 저장장소로부터 스트림 데이터를 제공할 때 다양한 미디어 매체들의 특성을 고려하여 버퍼링을 하는 방법에 관한 연구와 최근 VoD 시스템에서 저장장소의 효율성을 높일 수 있는 스트라이핑 기법을 사용하는 경우를 고려한 설계 모델의 개발도 필요하다고 본다.

## 참 고 문 헌

[1] D. James Gemmell, Harrick M. Vin, and Dilip D. Kandlur, P. Venkat Rangan and Lawrence A. Rowe, "Multimedia Storage Servers : A Tutorial," *IEEE Computer*, Vol. 28, No. 5, pp. 40-49, 1995.

[2] Wolf J. L., Yu P. S., and Shachnai H., "Disk load balancing for video-on-demand systems," *Multimedia Syst.* Vol. 5, No. 6, pp. 358-370, 1997.

[3] Z. D. Wu, "Design and Analysis of Video-on-Demand Servers," *IEEE Proceedings*, Vol. 2, pp. 773-778, Nov. 1998.

[4] Tantaoui M. A., Hua K. A., and Do T. T., "BroadCatch: a periodic broadcast technique for heterogeneous video-on-demand," *IEEE Transactions on Broadcasting*, Vol. 50, Issue 3, pp. 289-301, Sept. 2004.

[5] Wallace K. S. Tang, Eric W. M. Wong, Sammy Chan, and K. T. Ko, "Optimal video placement scheme for batching VOD services," *IEEE Transactions on Broadcasting*, Vol. 50, No. 1, pp. 16-25, Mar. 2004.

[6] D. Tran, K. Hua, and M. Tantaoui, "A Multi-Multicast Sharing Technique for large-scale Video Information Systems," *IEEE Internationl Conference on Communications*, Vol. 4, pp. 2496-2502, 2002.

[7] L. Golubchik, J. Lui, and R. Muntz, "Reducing I/O Demand in Video-On-Demand Storage Servers," *ACM Sigmetrics Conference*, Ottawa, Canada, pp. 25-36, May 1995.

[8] D. Rotem and J. L. Zhao, "Buffer Management for Video Database Systems," *Proceeding of International Conference on Data Engineer-*

*ing*, Taipei, Taiwan, Mar. 1995.

[9] A. Dan and D. Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads," *Proceeding of Multimedia Computing and Networking*, SPIE, San Jose, CA, Jan. 1996.

[10] Alper Caglayan and Colin Harrison, *Agent Sourcebook*, Wiley computer publishing, New York, 1997.

[11] T. Mitchell., *Machine Learning*, McGraw-Hill, New York, 1997.

[12] Lerman K., Minton S. N., and Knoblock C. A., "Wrapper Maintenance: A Machine Learning Approach," *Journal of Artificial Intelligence Research*, Vol. 18, pp. 149-181, 2003

[13] Gerhard Weiss, *Multiagent Systems: a modern approach to distributed artificial intelligence*, The MIT Press, Cambridge, Massachusetts, London, England, 2000.

[14] 김중환, 박정윤, "WMS의 성능 분석을 위한 에이전트 모델," 정보산업공학논문집, 제5집, pp. 36-45, 한국외국어대학교정보산업공학연구소, 2001.

[15] Arnold O. Allen, *Probability, Statistics, and Queuing Theory With Computer Science Applications*, Academic Press, San Diego, 1978.

[16] Cuena J., Josefa Z. Hernandez, and Molina M., "Knowledge-based Models for adaptive Traffic Management," *Transportation Research*, Vol.3, No.5, pp. 311-337, 1995.

[17] 이재규, 최형림, 김현수, 서민수, 주석진, 지원철, 전문가시스템 원리와 개발, 법영사, 1996.

[18] W. David Kelton, Randall P. Sadowski, and Deborah A. Sadowski, *Simulation with Arena, 2/e*, McGraw-Hill. 2002.

[19] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On Optimal Batching Policies for Video-on-Demand Storage Servers," *Proc. of the IEEE Int'l conf. on Multimedia Systems*, pp. 253-258, June 1996.

**박 정 윤**

1991년 2월 가톨릭대학교 수학과 학사
1995년 2월 한국외국어대학교 교육대학원 전자계산교육과 석사
2006년 8월 한국외국어대학교 컴퓨터공학과 박사
관심분야 : 멀티미디어

**김 중 환**

1978년 서울대학교 응용수학과 학사
1980년 고려대학교 산업공학과 석사
1984년 고려대학교 산업공학과 박사
1974년 육군사관학교 수학과 부교수
1985년~현재 한국외국어대학교 교수
관심분야 : 소프트웨어공학

**김 상 철**

1983년 서울대학교 컴퓨터공학 학사
1994년 미시간주립대학교 컴퓨터공학 박사
1983년 3월~1994년 8월 ETRI, 선임연구원
현재 한국외국어대학교 컴퓨터공학과 교수
관심분야 : 게임, 멀티미디어