멀티미디어 스트림 데이터 지원을 위한 MIDAS-Ⅲ 디스크 스케줄러의 설계

남 중 재[†] · 이 용 규[††] · 김 준[†††]

요 약

본 연구는 한국전자통신연구원에서 개발한 멀티미디어 DBMS의 하부 저장 시스템인 MIDAS-Ⅲ에서 기존에 지원하던 일반화일, BLOB, CLOB 형태의 데이터 이외에 멀티미디어 스트림 데이터를 추가로 지원하기 위한 연구이다. 본 논문에서는 MIDAS-Ⅲ에 새롭게 추가된 데이터 형태인 멀티미디어 스트림에 대한 디스크 입출력 성능을 향상시키기 위해 스트림 데이터의 대용량, 연속재생의 특성을 고려한 데이터 저장 구조를 설계한다. 또한 디스크 스케줄러가 존재하지 않던 기존의 MIDAS-Ⅲ에서 여러 형태의 데이터를 통합 지원하기 위한 2단계 디스크 스케줄러를 설계한다. 멀티미디어 스트림 데이터에 대해서는 데이터의 연속재생 특성을 고려한 선인출 기법을 적용하여 디스크 입출력 접근시간을 감소시켰다. 또한 기존 방법들과는 달리 디스크 스케줄링에서 스트림 데이터에 대하여 우선순위를 부여하지 않고 전체 디스크 접근 요구들을 최적화 하도록 처리한다. 이에 따라 줄어든 시간만큼 BLOB, CLOB 등과 같은 다른 요구들을 처리할 수 있으므로 전체적인 성능을 향상시킬 수 있다.

Design of the MIDAS-Ⅲ Disk Scheduler to Support Multimedia Stream Data

Joong-Jae Nam[†] · Yong-Kyu Lee[††] · June Kim[†††]

ABSTRACT

The purpose of this research is to extend the functionality of MIDAS-Ⅲ, the storage system of the BADA DBMS developed at ETRI, in order to support multimedia stream data. Currently, MIDAS-Ⅲ supports only non-continuous data types such as ordinary data files, BLOB and CLOB. We present a new storage structure designed to improve I/O performance for the new continuous data type. Also we present a new two-level disk scheduler that uses prefetching of multimedia stream data to reduce the number of disk seeks. By using prefetching it does not need to provide high priority to stream data requests, and thus it can process more disk access requests in a given time period.

키워드 : MIDAS, 디스크 스케줄러, 스트림 데이터

1. 서 론

기존 대부분의 저장 시스템들은 하나의 데이터 형태에만 적합하도록 설계되어, 다른 형태를 갖는 데이터의 저장과 검색에 있어서는 성능이 저하되는 단점이 나타나게 되었다.

현재 저장 시스템에 대한 외국의 연구 동향을 살펴보면 저장 시스템에서 단일한 형태를 갖는 데이터만을 지원하는 것이 아니라 텍스트, 오디오, 이미지, 비디오와 같은 다양한 형태를 효율적으로 지원하기 위한 통합 저장 시스템에 대한 연구[4, 5]가 활발히 진행중이다.

다양한 형태의 데이터를 효율적으로 저장, 검색하기 위해서는 각 데이터 형태에 따른 스케줄링 및 버퍼관리 정책이 요구되고 있으며, 특히 인터넷이 보편화되고 인터넷을 통한 실시간 미디어의 전송이 일반화되어감에 따라 끊김 현상이 없이 데이터를 재생할 수 있도록 하기 위한 디스크 스케줄링 및 버퍼링 기법이 중요한 요소로 자리잡고 있다.

특히 연속적 재생이 요구되는 멀티미디어 스트림과 같은 경우 연속 엑세스시의 성능은 곧 서비스의 품질과 결부되기 때문에 엑세스 성능 향상이 중요한 요소로 작용한다. 만일, 이러한 연속 엑세스 성능이 낮다면 데이터 재생이 매끄럽지 못하게 되어 매우 부자연스러운 영상이 될 것이다. 따라서 성능 극대화를 위하여 페이지들을 디스크에서 개별적으로 할당하지 않고, 연속적인 페이지들의 집합 단위로 할당함으로써 연속 엑세스 시 디스크 헤드의 이동시간(seek time)을 최소화 할 수 있는 방안이 고려되어야 한다.

† 준 회 원 : (주)쿼터뷰 연구원
†† 종신회원 : 동국대학교 컴퓨터멀티미디어공학과 교수
††† 정 회 원 : 한국전자통신연구원 데이터베이스엔진 연구팀장
논문접수 : 2000년 2월 8일, 심사완료 : 2000년 12월 13일

따라서 기존 MIDAS-Ⅲ[10]에서 대용량 연속재생 특성을 갖는 멀티미디어 스트림 데이터를 지원하기 위해서는 멀티미디어 스트림 데이터가 갖는 특성에 적합한 새로운 저장구조의 설계가 필요하다. 또한 멀티미디어 스트림 데이터는 일과성 접근 특성을 가지므로, 하나의 데이터 블록을 읽은 후에, 후속 스트림 데이터 블록을 읽어올 확률이 다른 데이터에 비해 배우 높다고 볼 수 있다. 따라서 MIDAS-Ⅲ에서는 여러번에 걸쳐서 주기적인 읽기 연산이 요구되는 스트림 데이터에 대해 선인출(prefetch) 기법을 적용하여 디스크 입출력 횟수를 줄일 수 있다. 이러한 기능과 더불어 각각의 데이터 특성에 적합한 서비스를 제공하기 위한 디스크 스케줄링 기법이 필요하다.

국외에서도 통합 파일 시스템에서 각각의 데이터 특성에 적합한 서비스를 제공하기 위한 스케줄러 설계에 대한 다양한 연구들이 진행되어 왔으며, 이중 2단계 스케줄링 기법을 이용한 연구[3-5]들이 주목을 받고 있다. 텍사스 대학에서 차세대 운영체제의 디스크 스케줄링을 위해 개발된 입출력 스케줄러인 Cello[3]는 각 응용들의 개별적 서비스 요구에 대한 입출력 스케줄러와 이들의 요구를 통합한 통합 스케줄러로 구성되는 2단계 스케줄링 구조를 갖는다. Cello의 특징은 클래스 종속적 스케줄러에서 실시간 작업에 대해서는 SCAN- EDF 알고리즘을 이용하며, 빠른 응답이 요구되는 작업의 경우 통합 큐의 앞에 삽입하여 다른 요구들보다 먼저 처리하도록 하는 것이다. 그러나 이들을 우선 처리하기 위해 디스크 탐구 시간이 늘어나는 단점이 있다.

Texas A&M 대학에서 연구되고 있는 통합 파일 시스템의 Integrated QOS Manager[4]에서도 2단계 스케줄러를 사용하고 있으며, 디스크에 대한 입출력 요구를 주기적 요구(periodic request), 대화형 요구(interactive request), 비주기적 요구(aperiodic request)의 세가지 범주로 구분하여 응용들의 요구 형태에 따라 서로 다른 서비스를 제공한다. 이 시스템에서는 Cello와는 달리 Admission Control 기능을 스케줄러로부터 분리하였고, 대화형 요구에 우선 순위를 부여하는 특징을 가진다.

또한 워싱턴 대학의 MARS[5](Massively-parallel And Real-time Storage) 프로젝트의 일환으로 연구되고 있는 분야로 멀티미디어의 효율적 검색을 지원하기 위한 연구에서도 Cello와 마찬가지로 통합 파일 시스템을 지원하기 위해 2단계 스케줄링 구조를 가지고 있으며 Cello가 운영체제와 독립적인 파일 시스템인 반면 MARS의 스케줄러는 기존 4.4 BSD UNIX의 커널을 확장 설계한 시스템이다.

본 논문에서는 기존 MIDAS-Ⅲ에서 지원하고 있는 데이터 형태에 추가적으로 멀티미디어 스트림 데이터를 지원하기 위해 새로운 저장구조를 설계하고 MIDAS-Ⅲ에서 효율적인 디스크 입출력을 수행하기 위한 디스크 입출력 스케줄링에 대한 연구를 기술한다.

MIDAS-Ⅲ에서 지원하는 기존의 데이터는 일정 크기를 갖는 페이지 단위로 디스크와 메모리사이에 입출력이 이루어진다. 따라서 대용량의 멀티미디어 스트림에 대해서 이러한 페이지 단위의 입출력은 잦은 입출력 수행에 따라 시스템 성능 저하의 요인이 된다. 따라서 이러한 문제를 해결하기 위해 멀티미디어 스트림 데이터를 효율적으로 지원하기 위한 저장구조를 설계한다.

MIDAS-Ⅲ를 위한 디스크 스케줄러 설계에 있어서 멀티미디어 스트림 데이터의 특성을 고려하여 디스크로부터 멀티미디어 스트림을 읽어오는데 발생하는 지연시간을 줄일 수 있도록, 선인출 기법을 적용한 2단계 디스크 스케줄러를 설계하고 이의 성능을 시뮬레이션을 통해 평가한다. 일반 요구와는 달리 스트림 요구는 일정시간 계속하여 연속된 스트림을 접근하므로 한번에 접근하여 읽어 오는 블록 수를 크게 함으로써 디스크 접근 시간을 단축할 수 있다.

본 논문에서 연구한 디스크 스케줄러의 구조는 기존 Cello 등의 2단계 스케줄러와 같은 구조를 지니지만, 멀티미디어 스트림 데이터에 대해서 우선 순위를 부여하는 대신 한번에 여러 블록들을 읽는 선인출 기법을 이용하여 디스크 접근 횟수를 줄이므로 동일한 시간에 처리 가능한 일반요구의 수를 증가시킬 수 있다. 또한 통합 큐의 요구에 대한 디스크 접근 순서를 결정하는데 있어서도 기존의 시스템들과 달리 스트림 데이터 요구에 대해 우선 순위를 부여할 필요가 없으므로, 전체 요구에 대해 LOOK 알고리즘에 의한 디스크 스케줄링 기법을 채택하여 디스크 접근 지연시간을 최소화하도록 한다. 또한 개별 큐의 요구들을 우선 순위에 따라 통합 큐에 배치하는데 따른 알고리즘의 복잡성을 단순화할 수 있으므로 이에 대한 오버헤드를 줄일 수 있다.

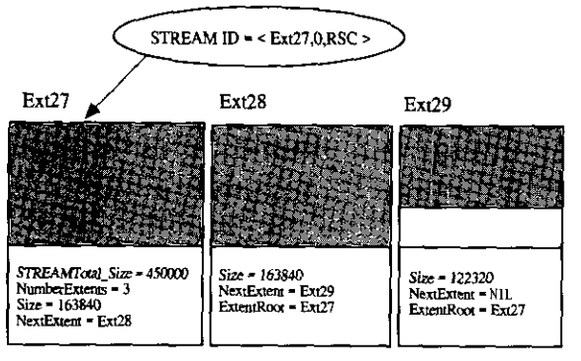
2. 확장된 MIDAS-Ⅲ의 데이터 타입

기존 MIDAS-Ⅲ에서는 일반파일과 음성, 이미지, 동영상을 지원하기 위한 데이터 형태인 BLOB(Binary Large OBject) 그리고 정보검색을 위한 전문(full text)만을 저장하는 데이터 형태인 CLOB(Character Large OBject) 데이터를 처리하기 위한 자료구조를 정의하고 있다. 여기에 연속적 재생이 필요한 멀티미디어 스트림 형태의 데이터를 지원하기 위해 스트림 데이터를 위한 자료구조를 추가해야 하며, 기존의 자료구조에 대해 일부 수정이 요구된다.

MIDAS-Ⅲ에서 기존에 지원하던 BLOB는 부분적인 갱신 또는 삭제를 가능하게 하기 위해 BLOB에 대한 인덱스를 제공하고 있다. 그러나 스트림 데이터는 갱신 연산보다는 읽기 연산이 주로 발생하기 때문에 이러한 색인구조가 필요치 않다. 또한 BLOB는 16KBytes 크기를 갖는 페이지 단위의 연결리스트 형태로 구성되며, 그 최대크기에는 제한이 없다. 따라서 스트림 데이터를 지원함에 있어서도

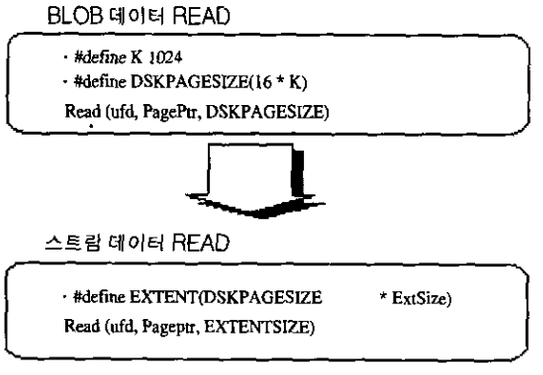
BLOB의 구조를 응용하여 자료구조를 생성할 수 있다. 그러나 스트림 데이터의 내부 구조에 있어서는 기존 BLOB의 내부 구조에서처럼 페이지 단위의 리스트 구조보다는 단위를 확장시켜 연속적 페이지의 집합인 익스텐트 단위의 리스트 구조가 적합하다. 수십에서 수백 MB의 크기를 갖는 하나의 멀티미디어 스트림 데이터를 인접한 익스텐트들에 분할 저장하여 순차적인 읽기 연산을 수행할 수 있는 구조를 갖추어야 한다. 또한 멀티미디어 스트림의 임의 위치에서 미디어 재생이 가능하도록 각 익스텐트들에 대한 정보를 관리할 수 있는 자료구조가 요구된다.

예를 들어 익스텐트를 구성하는 페이지의 개수가 10이고 페이지의 크기가 16KBytes라 가정하였을 때, 450,000Bytes 크기를 갖는 멀티미디어 스트림 데이터의 저장구조를 그림으로 표현하면 다음 (그림 1)과 같은 형태가 된다.



(그림 1) 스트림 데이터의 내부구조

(그림 1)에서 STREAMTotal_Size는 스트림 데이터의 전체 크기(Bytes)를 의미하며, NumberExtents는 스트림 데이터를 저장하는 데 필요한 익스텐트의 개수를 말한다. 그리고 Size는 익스텐트의 크기(Bytes)이고, NextExtent는 동일한 스트림 데이터를 저장하는 후속 익스텐트의 식별자를 표시한다.



(그림 2) BLOB와 스트림 데이터의 읽기 연산

MIDAS-Ⅲ에서 데이터에 대한 접근은 MIDAS-Ⅲ API를 통하여 데이터가 속하는 페이지를 공유메모리 내 버퍼로 가져온 후 이루어진다. MIDAS-Ⅲ에 멀티미디어 스트림 데이터를 지원하기 위해 접근하고자 하는 데이터의 페이지 식별자(PID)가 멀티미디어 스트림 데이터를 가리키게 되면 해당 익스텐트 전체를 읽도록 해야 한다. 따라서 다음 (그림 2)와 같은 형태의 읽기 연산이 수행되어야 한다.

한번의 READ 연산에 의해, 페이지 단위가 아닌 익스텐트 단위의 데이터를 디스크에서 메모리 버퍼로 가져올 수 있으므로, 멀티미디어 스트림 데이터에 대한 읽기 연산 수행시 잦은 디스크 접근 회수에 따른 디스크 입출력 지연시간을 줄일 수 있다.

또한 기존 MIDAS-Ⅲ에서 멀티미디어 스트림 데이터를 지원하기 위해 기존 MIDAS-Ⅲ의 BLOB와 CLOB 데이터의 공간 식별자, 공간 설명자, 데이터 식별자 이외에 스트림 데이터 공간을 구별하기 위해 사용되는 스트림 데이터 공간식별자, 스트림 데이터 공간에 대한 정보 유지를 위한 스트림 데이터 공간 설명자가 필요하다. 그리고 각각의 스트림 데이터를 식별하기 위한 스트림 데이터 식별자가 기존 MIDAS-Ⅲ의 자료구조에 추가되어야 한다.

3. 스트림 데이터 지원을 위한 디스크 스케줄링

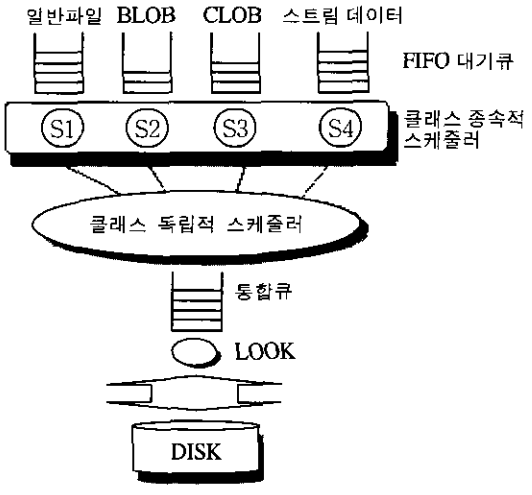
본 논문에서는 MIDAS-Ⅲ에서 일반화일, BLOB, CLOB, 멀티미디어 스트림과 같은 다양한 형태의 데이터를 지원하고, 멀티미디어 스트림 데이터에 대한 디스크의 입출력 성능을 향상시키기 위한 디스크 스케줄링 기법에 대해 연구하였다.

기존의 스케줄링 방식에서는 데이터 형태에 따라 디스크 입출력 요구를 대화형 요구, 실시간 요구, 일괄처리형 요구의 세가지로 구분하였으며, 대화형 요구에 대해서는 통합 큐의 앞쪽에 삽입하여 처리하여 우선 순위를 갖도록 하였으며, 비실시간 요구에 대해서는 통합 큐의 뒤쪽에 위치시켜 처리함으로써 상대적으로 낮은 우선 순위를 부여하였다. 그러나 멀티미디어 스트림과 같은 주기적 실시간 요구에 대해서는 서비스의 품질을 고려하여 통합 큐의 임의의 위치에 요구를 삽입하게 되므로 멀티미디어 스트림 데이터를 엑세스하기 위해 SCAN 순서에 의해 최적화된 디스크 헤드의 이동 순서가 뒤바뀌게 된다. 이에 따라 디스크의 엑세스 성능이 저하되는 단점이 존재한다. 본 논문에서 제시한 스케줄링 방식은 웹 서버 등과 같이 네트워크를 통해 이루어지는 데이터 서비스에 적용하기 위한 방법이다. 이러한 서비스의 특성상 디스크 엑세스 성능보다 네트워크의 성능이 보다 중요한 성능 요소이므로 디스크 엑세스시 대화형 요구와 같은 특정 작업에 우선 순위를 부여할 필요성이 감소한다. 즉, 디스크 접근에서의 약간의 지연은 네트워크의 지연에 비해 상대적으로 작다고 할 수 있다. 따라서 통합 큐의 모든 요구들을 LOOK 순서로 처리하여 디스크의 엑

세스 성능을 향상시킬 수 있다. 또한, 각각의 요구를 처리하는데 있어서 일반 데이터와 멀티미디어 스트림을 동시에 효율적으로 처리하기 위한 방안으로 데이터의 연속된 읽기 연산이 필요한 멀티미디어 스트림을 엑세스 할 때, 스트림 데이터의 특성을 고려하여 다음에 읽을 데이터를 미리 메모리에 적재하는 선인출 기법을 이용해 완전한 실시간 서비스를 지원할 수 있도록 하였다. 결과적으로 스트림 데이터 선인출에 따라 후속 스트림 데이터를 엑세스하기 위해 디스크에 접근하는 횟수를 줄임으로 읽기 성능을 향상시킬 수 있다.

3.1 2단계 디스크 스케줄러

먼저 스케줄링 구조를 살펴보면, 2단계 디스크 스케줄러를 설계하여 각 데이터 형태별로 스케줄링이 이루어진 후 개별 큐에 삽입된 입출력 요구들을 통합하여 처리하도록 구성하였다. 다음 (그림 3)은 2단계 디스크 스케줄러에서 각 스케줄러의 역할과 구성을 보여준다.



(그림 3) 2단계 스케줄러의 구조

3.1.1 클래스 독립적 스케줄러

클래스 독립적 스케줄러에서는 각각의 대기 큐에 존재하는 요구들을 처리하기 위해 클래스 종속적 스캐줄러에 디스크 대역폭(bandwidth)을 할당하고 클래스 종속적 스케줄러에서 선택된 요구들을 통합 큐에 삽입하고 통합 큐에 존재하는 요구들을 처리하기 위해 LOOK 알고리즘을 이용하여 각 요구들에 대한 디스크 접근 순서를 최적화 시킨다. 각 클래스에 대한 대역폭은 파라미터에 의해 부여되나 각 클래스의 대기 큐의 상태에 따라 동적으로 변화한다. 즉, 각 클래스의 큐에 대기하는 요구들의 개수를 파악하여 각 클래스에게 할당하는 대역폭을 조정하면 대기 큐의 상태에 따라 동적인 스케줄링이 가능하다. 그리고, 멀티미디어 스트림에 대한 요구 처리시 정해진 블록 수만큼의 스트림 데이터를 선인출하며, 하나의 요구는 몇 번의 라운드를 주기로 번갈아 가며 처리된다.

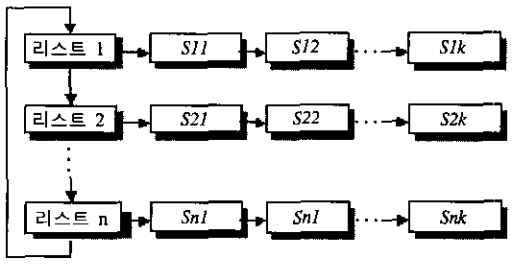
3.1.2 클래스 종속적 스케줄러

클래스 종속적 스케줄러에서는 할당받은 대역폭에 따라 대기 큐에 존재하는 개별 데이터 형태의 요구들을 대상으로 대기 큐에 있는 작업 요구를 선별하여 클래스 독립적 스케줄러에 전달한다. 스트림 데이터 스케줄러에서는 스트림 데이터에 대한 요구 발생시 요구된 스트림 데이터에 대한 선인출을 수행할 수 있도록 별도의 서브 큐를 두어 대역폭을 할당받은 스트림 데이터의 후속 스트림 블록들을 리스트 형태로 유지한다.

3.2 선인출 기법을 이용한 디스크 스케줄링

연속적 재생이 요구되는 멀티미디어 스트림과 같은 경우 끊김 현상 없이 데이터를 재생하기 위해 디스크의 데이터를 읽어와 버퍼에 저장하고 버퍼로부터 데이터를 읽어 사용자에게 전달하여야 한다. 따라서 멀티미디어 스트림에 대한 입출력 요구시 2단계 디스크 스케줄링을 이용한 선인출 기법을 적용하여 다음 라운드에 읽어올 스트림 데이터를 메모리에 사전 적재하여, 디스크 성능에 있어서의 지연 요소인 탐구시간을 줄임으로 디스크 입출력에 따른 지연시간을 감소시킬 수 있다. 일반적인 선인출된 블록들이 사용되지 않을 경우 오버헤드가 발생하게 되지만, 스트림 데이터는 연속해서 읽는 것이 일반적이므로 오버헤드가 작게 된다. 또한 2단계 디스크 스케줄러를 이용한 선인출 기법을 적용하면 동일한 시간에 기존 방법들보다 더 많은 일반 요구들을 처리할 수 있다는 장점을 가지게 된다.

하나의 라운드에서 파라미터로 정의한 선인출 블록의 수만큼을 선인출하고 n 라운드 후에 선인출한 이후의 스트림 데이터를 읽어오게 된다. 이를 위해 다음 (그림 4)와 같이 스트림 데이터 요구들을 n개의 서브 리스트로 분할하고, 이를 환형 리스트로 연결하여, 한 라운드에 하나의 서브 리스트를 번갈아 처리할 수 있도록 하는 자료구조가 필요하다.



(그림 4) 스트림 데이터 처리를 위한 자료구조

위와 같은 구조에서 하나의 라운드에 읽어올 데이터에 대한 요구들을 n개의 리스트로 유지하며, n개의 리스트 헤더는 환형 리스트(circular list)의 형태를 갖게된다. 멀티미

디어 스트림 데이터에 대한 읽기 연산 수행시 이러한 자료 구조를 통해 한 라운드에 하나의 리스트씩 번갈아 가며 처리하여 n 라운드에 모든 스트림 데이터에 대한 요구들을 한 번씩 처리하게 된다. 결과적으로 한 라운드에서 처리해야 할 요구들의 수가 감소하게 되므로 감소하는 요구들의 수만큼 디스크에 대한 접근 횟수가 줄어들게 되어 디스크 엑세스에서의 커다란 성능지연요소인 탐구시간을 줄일 수 있게 된다.

이와 같은 스케줄링 알고리즘 중에서 먼저 일반 응용의 요구에 대한 스케줄링 알고리즘을 기술하면 (그림 5)와 같다. 여기서는 통합 스케줄러로부터 할당받은 대역폭으로 대기 큐의 모든 요구들을 처리할 수 있으면 이들을 모두 처리하지만, 그렇지 않을 경우는 일부의 요구들을 선착순으로 처리하는 것을 설명하고 있다.

```
loop forever
{
  통합 스케줄러로부터 대역폭을 할당받음;

  if 대기 큐의 요구들을 할당된 대역폭으로 처리 가능하다면,
     대기 큐의 모든 요구들을 선택;

  else 대기 큐의 요구들을 할당된 대역폭의 범위 내에서 FCFS
       방식으로 선택;

  통합 스케줄러에 선택된 요구들을 전달;
} end loop
```

(그림 5) 일반 응용의 요구에 대한 스케줄링

(그림 6)은 스트림 요구에 대한 스케줄링 알고리즘이다. 이것도 앞의 설명과 마찬가지로 모든 요구의 처리가 불가능하면 선착순으로 처리한다.

```
loop forever
/*스트림 데이터 큐는 n 개의 서브 큐들로 구성되어 한 라운드에
하나의 서브 큐를 순서대로 처리*/
{
  통합 스케줄러로부터 대역폭을 할당받음;

  if 다음에 처리될 서브 큐의 요구들을 할당된 대역폭으로 처리
    가능하다면, 서브 큐의 모든 요구들을 선택;

  else 서브 큐의 요구들을 할당된 대역폭의 범위내에서 FCFS 방
       식으로 선택;

  통합 스케줄러에 선택된 요구들을 전달;
} end loop
```

(그림 6) 스트림 데이터 요구에 대한 스케줄링

(그림 7)은 통합 스케줄러의 스케줄링 알고리즘으로 클래스 종속적 스케줄러에 대역폭을 할당해 주고 각 스케줄러로부터 선택된 요구들을 LOOK 순서대로 정렬하여 처리한다. 특히 어떤 요구가 스트림 요구일 때는 미리 정의된 선인출 블록 수만큼 선인출한다.

```
loop forever
{
  클래스 종속적 스케줄러에 대역폭 할당;

  각각의 클래스 종속적 스케줄러로부터 선택된 요구들을 전달
  받아 통합 큐에 삽입;

  LOOK 순서에 의해 통합 큐의 요구들을 정렬;

  while (통합 큐 != empty)
  {
    if 디스크 입출력을 수행할 요구가 스트림 데이터에 대한
    요 구일 경우
    {
      디스크로부터 해당 스트림 데이터들을 읽어 처리하고
      각각의 요구에 대해 선인출 블록 수(n) 만큼의 후속 스
      트림을 선인출;
    }

    else 일반 응용에 대한 요구를 처리하기 위해 디스크로부
    터 정해진 크기의 블록을 읽음 ;
  } end while
} end loop
```

(그림 7) 통합 스케줄러에서의 디스크 스케줄링

이러한 선인출 기법의 단점이라면 디스크에서 메모리 버퍼로 한번에 큰 단위의 데이터를 적재하기 때문에 버퍼사용에 있어서의 메모리 이용 효율을 감소시킨다는 단점을 갖는다.

4. 디스크 스케줄러의 성능 평가

본 논문에서는 시뮬레이션을 통하여 멀티미디어 스트림 데이터의 선인출에 따른 디스크 입출력 성능을 비교, 평가하였다. 시뮬레이션에 이용된 디스크는 IBM사의 DORS-32160 모델[11]이며, 성능 매개변수는 <표 1>과 같다. 그리고 한 블록의 크기는 1KBytes로 하였다.

<표 1> 디스크의 성능

사 양	
capability	2.16 GB
sector size	512 Bytes
data head	5
cylinder	6701
sector per track	125
disks	3
성 능	
rotational speed	5400 RPM
media transfer rate	47.4-71.6 Mb/sec
latency average	5.56 ms
탐구 시간	
seek average	8.5 ms
track to track	3.0 ms
full track	15.0 ms

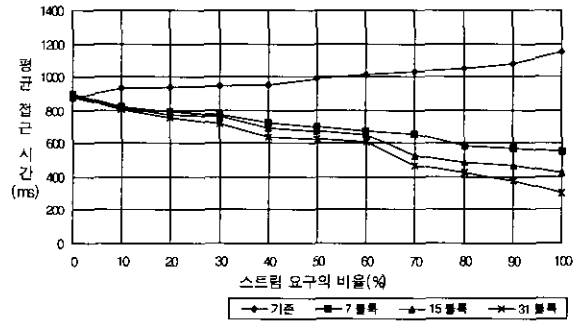
시뮬레이션에 있어서 한 라운드에서 처리하는 요구의 수를 파라미터를 통해 100이라고 정의했을 때 100개의 임의

의 디스크 주소를 생성하여 기존의 방식과 LOOK 순서에 의한 정렬과 선인출 기법을 이용한 기법에서의 디스크의 입출력 시간 및 추가 처리 가능한 요구의 수를 분석하였다.

기존의 2단계 스케줄링 구조에서 사용하는 통합 큐의 스케줄링 방식은 대화형 요구에 대해 통합 큐에서 높은 우선순위를 부여하고, 일괄처리형 요구들은 상대적으로 낮은 우선 순위를 부여하여 SCAN-EDF 방식으로 데이터에 대한 디스크 접근 순서를 결정하였다. 이러한 방식에서 멀티미디어 스트림과 같은 데이터는 주기적인 디스크 접근이 요구되며, 또한 서비스 품질을 유지하기 위해 데이터의 재생 주기에 따라 통합 큐에서의 위치가 결정된다. 따라서 대화형 요구와 일괄처리형 요구들의 SCAN 순서에 의해 최적화된 디스크 접근 순서가 멀티미디어 스트림 데이터를 처리하기 위한 주기적 요구에 의해 뒤바뀌게 되어 디스크의 엑세스 성능을 저하시킨다.

본 연구에서 제시한 2단계 디스크 스케줄러는 기존의 2단계 스케줄러에서 채택하고 있는 통합 큐의 스케줄링 정책 대신에 특정 요구에 대한 우선 순위 부여 없이, 모든 요구들에 대해 LOOK 디스크 스케줄링 방식을 적용하여, 요구들을 처리하기 위한 디스크 엑세스 성능을 향상시켰다. 또한 후속 데이터를 읽을 확률이 높은 스트림 데이터에 대해 선인출 기법을 적용하여 후속 스트림을 미리 메모리로 적재함으로써, 선인출한 데이터 블록의 수만큼 디스크 접근 횟수를 감소시킬 수 있다. 따라서 동일한 크기의 멀티미디어 스트림 데이터를 읽는데 소요되는 엑세스 시간을 줄일 수 있다.

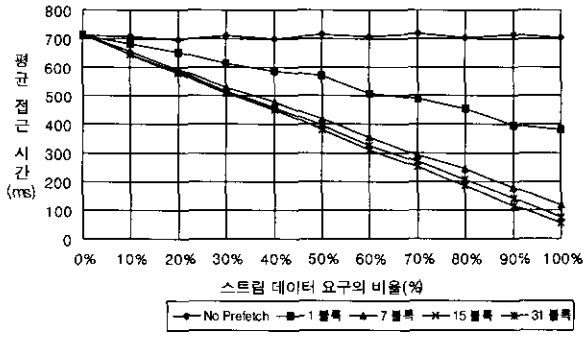
(그림 8)은 기존의 스케줄링 방식에 선인출 기법을 적용했을 때 라운드 (100 요구들) 당 평균 디스크 접근시간을 기존 방식과 비교한 그래프이다. 기존 방식에서는 멀티미디어 스트림 요구의 비율이 증가함에 따라 접근시간이 증가하나, 멀티미디어 스트림에 대해 선인출 기법을 적용했을 때는 멀티미디어 스트림의 비율과 선인출 블록의 크기가 증가함에 따라 접근시간이 감소함을 확인할 수 있다.



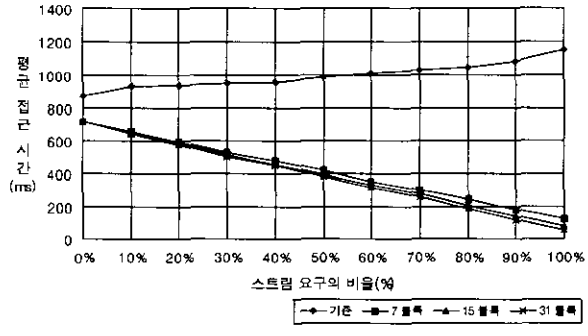
(그림 8) 기존방식에 선인출기법 적용시 라운드당 디스크 접근시간

(그림 9)는 이 논문에서 설계한 LOOK 순서에 의한 정렬과 선인출 기법을 이용한 스케줄러에서 선인출 블록의 개수와 전체 요구에 대한 멀티미디어 스트림의 요구 비율에 따른 라운드 당 디스크 접근시간을 측정한 결과이다. 보는 바와 같이 멀티미디어 스트림의 비율이 높을수록 선인출 블록의 크기가 클수록 디스크의 접근시간이 감소함을 확인할 수 있다.

(그림 10)은 기존의 방식과 통합 큐의 LOOK 순서에 의한 정렬과 선인출 기법을 적용한 스케줄러의 성능을 비교한 결과이다.



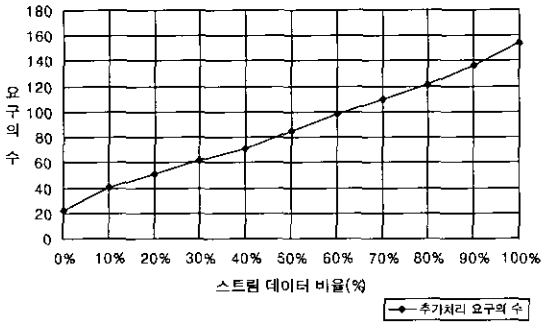
(그림 9) 선인출 블록의 크기와 멀티미디어 스트림 요구의 비율에 따른 라운드당 디스크 접근시간



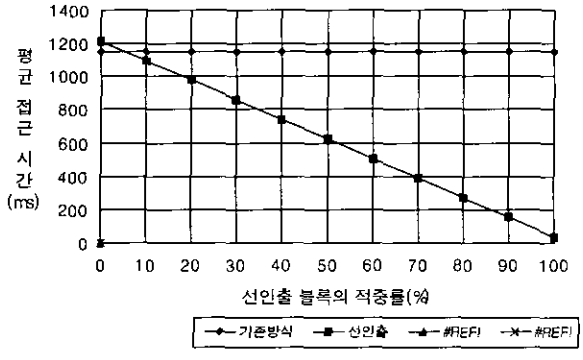
(그림 10) 통합 큐의 LOOK 순서에 의한 정렬과 선인출 기법 적용시 라운드당 디스크 접근시간

(그림 11)은 스트림 데이터의 선인출 블록 크기를 31로 했을 경우 LOOK 순서에 의한 정렬과 선인출 기법을 적용한 스케줄러를 채택했을 때 기존의 방식과 비교해 라운드당 추가적으로 처리할 수 있는 일반요구의 수를 그래프로 나타낸 것이다. 이 결과에서처럼 멀티미디어 스트림의 선인출에 따라 감소된 시간동안 기존 방식보다 많은 요구들을 처리할 수 있다.

(그림 12)는 선인출 기법을 사용할 때 적중률(hit ratio)의 변화에 따른 한 라운드별 디스크 접근시간의 변화를 보여주고 있다. 여기서는 선인출 블록의 수를 31이라 할 때 멀티미디어 스트림 데이터를 대상으로 선인출을 하는 경우와 그렇지 않은 경우를 비교하고 있다. 그림에서 보듯이 적중률이 증가할수록 선인출 기법이 더욱 우수함을 보여주고 있다. 이는 디스크의 탐구 시간과 회전 지연 시간에 비해 데이터의 전송시간이 상대적으로 매우 작기 때문이다.



(그림 11) 통합 큐의 정렬과 선인출 기법의 적용시 라운드당 추가 처리 가능한 요구의 수



(그림 12) 선인출 블록의 적중률에 따른 라운드당 디스크 접근시간

5. 결 론

본 논문에서는 기존 MIDAS-III에서 지원하고 있는 데이터 형태에 추가적으로 멀티미디어 스트림 데이터를 지원하기 위한 MIDAS-III의 저장 구조 확장에 대해 연구하였다. 기존 BLOB의 페이지 단위 저장 구조를 확장하여, 멀티미디어 스트림 데이터를 지원하기 위해 데이터 저장 단위를 연속된 페이지들의 집합인 익스텐트 단위로 확장하여 대용량의 멀티미디어 스트림에 대한 읽기 연산 수행시 디스크로부터 메모리 버퍼로의 데이터 이동시 디스크의 입출력 성능 지연요소인 탐구시간을 줄일 수 있다.

그리고 일반화일, BLOB, CLOB, 멀티미디어 스트림 데이터를 효과적으로 처리하기 위한 디스크 스케줄러 설계에 있어서, 주기적인 읽기 연산이 요구되는 멀티미디어 스트림 데이터에 대해 선인출 기법을 적용하여 주기적으로 발생하는 멀티미디어 스트림에 대한 접근시 발생하는 디스크의 지연시간을 최소화하였다. 따라서 멀티미디어 스트림 데이터에 대한 특별한 스케줄링이 필요치 않으므로 모든 요구들에 대해 LOOK 디스크 스케줄링 방식을 적용하여, 디스크 상에 물리적으로 인접한 데이터를 순대로 접근하므로 디스크 헤드의 이동시간을 최적화 시켜 디스크 엑세스 성능을 향상시켰다.

단점이라면 통합 스케줄러에서 LOOK 순서에 의해 디스크 접근 순서가 결정되므로 더 많은 요구들을 처리할 수 있는 대신에 대화형 작업에 대해 우선 순위를 부여하지 않는다는 점이다. 따라서 이 방식은 웹 서버, proxy 서버, 멀티미디어 서버 등과 같이 대화형 작업의 비중이 낮은 네트워크 응용 환경에 적합하다. 향후에는 본 연구의 결과를 MIDAS-III에 적용하기 위해 버퍼 관리에 관한 연구가 이루어져야 한다.

참 고 문 헌

[1] Jose Renato Santos and Richard Muntz, "Design of the RIO Storage Server," Technical Report 970032, Dept. of Computer Science, Univ. of California at Los Angeles, Aug. 1997.

[2] P. J. Shenoy, P. Goyal, S. Rao, and H. M. Vin, "Design and Implementation of Symphony : An Integrated Multimedia File System," Technical Report TR-97-09, Dept. of Computer Science, Univ. of Texas at Austin, Mar. 1997.

[3] P. Shenoy and H. M. Vin, "Cello : A Disk Scheduling Framework for Next-generation Operating Systems," Technical Report RC20670, T. J. Watson Research Center, 1996.

[4] Ravi Wijayaratne and A. L. Narasimha Reddy, "Integrated QOS management for Disk I/O," Proc. IEEE Conf. on Multimedia Computing and Systems, June 1999.

[5] Milind M. Buddhikot, Xin Jane Chen, Dakang Wu, Guru M. Parulkar, "Enhancements to 4.4 BSD UNIX for Efficient Networked Multimedia in Project MARS," op.cit.

[6] Silberschatz and Galvin, "Operating System Concepts," 4th ed., Addison Wesley, 1994.

[7] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of ACM, pp.46-61, Jan. 1973.

[8] M. J. Carey, R. Jauhari, and M. Linvy, "Priority in DBMS Resource Scheduling," in Proc. of the 15th VLDB Conf., pp.397-410, 1989.

[9] A. Reddy and J. Wyllie, "Disk Scheduling in a Multimedia I/O System," in Proc. ACM Conf. on Multimedia, pp.225-233, Aug. 1993.

[10] 박치항 외 13인, "바다 DBMS를 중심으로 한 데이터베이스 관리 시스템 구조," 한국전자통신연구원, 1997.

[11] IBM, DORS-32160 Hard Disk Spec., 1994.

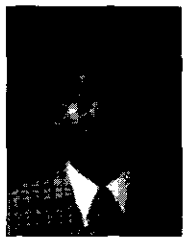
남 중 재

e-mail : bamjjung@hitel.net

1988년 경원대학교 전자계산학과 (학사)

2000년 동국대학교 컴퓨터공학과 (석사)

2000년~현재 (주)쿼터뷰 연구원



이 용 규

e-mail : yklee@dgu.edu

1986년 동국대학교 전자계산학과(학사)

1988년 한국과학기술원 전산학과(석사)

1996년 Syracuse University (전산학박사)

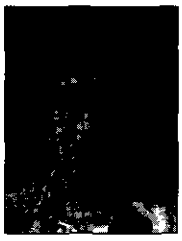
1978년~1983년 정보통신부 국가공무원

1988년~1993년 한국국방연구원 선임연구원

1996년~1997년 한국통신 선임연구원

1997년~현재 동국대학교 컴퓨터멀티미디어공학과 교수

관심분야 : XML 및 웹, 저장시스템, 데이터베이스, 정보검색



김 준

e-mail : jkim@etri.re.kr

1983년 부산대학교 계산통계학과(학사)

1986년 한국과학기술원 전산학과(석사)

1986년~현재 한국전자통신연구원 데이터베이스엔진 연구 팀장

관심분야 : 멀티미디어 DBMS, 자료저장 시스템, 회복 및 동시성 제어