연속질의의 처리를 위한 이용률 기반의 적응적 메모리 관리 기법

(Adaptive Memory Management Method based on Utilization Ratio to Process Continuous Query)

백 성 하[*] 이 동 욱[*] 어 상 훈[**] 정 원 일[***] 배 해 영[****]
(Sung Ha Baek) (Dong Wook Lee) (Sang Hun Eo) (Weon il Chung) (Hae Young Bae)

요 약 실시간으로 입력되는 스트림을 저장하기 위한 메모리의 크기는 동적으로 변한다. 이 데이터 스트림을 처리하는 연속질의는 저장공간의 크기를 동적으로 관리해야 한다. 이를 위해, 저장되는 현재 데이터 양에 따라 즉시 페이지 단위로 메모리를 할당 및 해제하는 기본적인 메모리 관리자가 연구되었다. 그러나 이 방법은 데이터 스트림을 저장하기 위해 메모리의 할당 및 해제를 매우 빈번하게 수행하게 된다. 또한 질의가 메모리가 부족할 때 즉시 페이지를 할당하기 때문에, 특정 지연되는 질의가 대량의 페이지를 점유하는 문제를 발생시킬 수 있다. 메모리관리자에서 발생하는 이와 같은 문제에 초점을 맞추어, 본 연구는 할당 및 해제 빈도수를 감소시키고, 질의 별로 최대한 균등하게 페이지를 분배하는 메모리 관리 기법을 제안한다. 본 기법은 질의의 페이지 이용률을 이용하여 할당 및 해제 빈도수를 크게 감소시키고, 질의의 지연 상태에 따른 메모리의 할당을 통하여 특정 질의의 메모리 독점을 방지할 수 있다.
키워드 : 데이터 스트림, 메모리 관리, 메모리 풀, 연속 질의

Abstract The volume of memory to store real-time data stream is varied dynamically. Continuous queries processing the data stream must manage the storage volume dynamically. In previous research, according to current volume of data a general memory manager which allocates and releases memory by a page unit is researched.However, the method frequently executes page allocation and release to store data stream. Moreover, particularly delayed queries can monopolize many of pages because the method directly allocates pages when a query has not enough memory. Focusing on the problems in memory management systems, this research proposes a memory management method which reduces the frequency of allocation and release and uniformly distributes pages for queries. The method can reduce the frequency of allocation and release through allocation based on utilization ratio of pages in each query and prevent memory monopoly through memory allocation which considers query delay.
Keywords : Data Stream, Memory Management, Memory Pool, Continuous Query

1. 서 론

최근 센서 장비 및 통신 기술의 발달에 따른 네트워크 트래픽의 처리 및 전자 상거래 거래 데이터, 온라인 주식 등 정보가 빠르게 유입되는 데이터 스트림(Data Stream)에 대한 연구가 활발히 진행 되고 있다. 데이터 스트림 관리 시스템(Data Stream Management System)은 질의를 시스템에 등록하고 데이터 스트림이 입력될 때 조건에 맞는 데이터만 처리하는 연속질의(Continuous Query Language)를 사용한다[1,2,3,4]. 데이터 스트림 관

† 본 연구는 건설교통부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었습니다.
* 인하대학교 컴퓨터정보공학과 박사과정, {shbaek, dwlee}@dblab.inha..ac.kr
** LG전자 MC 선임연구원, eosanghun@lge.com
*** 호서대학교 정보보호학과 전임강사, wnchung@hoseo.edu(교신저자)
**** 인하대학교 컴퓨터정보공학과 조교수, hybae@inha.ac.kr
논문접수 : 2008.10.13 수정일 : 1차 2009.03.12 / 2차 2009.05.25 심사완료 : 2009.05.27

리 시스템은 이와 같이 연속질의를 처리하기 위해 입력되는 데이터 스트림을 질의마다 큐 구조의 메모리 공간인 스트림 큐에 저장한다[5,6].

스트림 큐는 미리 대량의 메모리를 할당 받고 페이지 단위로 이를 관리하는 메모리 풀로부터 질의가 요청한 가용 페이지를 획득하여 생성된다. 그런데 스트림 큐에 저장되는 데이터 스트림은 데이터의 양이 매우 많고, 끊임 없이 입력되며, 입력되는 양이 실시간으로 변한다. 만약 데이터 입력속도가 증가하면 스트림 큐의 저장공간을 초과하는 데이터 스트림이 입력될 수 있다.

저장공간 초과 문제를 해결하기 위해 스트림 큐는 저장 공간이 초과할 때 메모리 풀로부터 여분의 페이지를 즉시 할당 받는 방법을 사용하거나 초기에 스트림 큐가 생성될 때 대량의 페이지를 할당하는 방법을 사용한다[5,6]. 그러나 이 방법들은 몇 가지 문제점을 가지고 있다. 우선 즉시 할당하는 방식은 스트림 큐에 저장되는 데이터의 양이 변할 때 마다 페이지를 할당하거나 해제해야 하는 문제가 있다. 특히 데이터의 입력속도가 급격하게 변하는 경우, 스트림 큐는 연속적으로 페이지의 할당 및 해제 요청을 하게 된다. 이 방식은 각 연속 질의를 위한 여러 개의 프로세스가 메모리 풀에 동시에 할당 및 해제 요청을 하게 되어 메모리 풀에 병목현상을 초래 할 수 있다. 이와 다른 방식인 초기에 대량의 페이지를 할당하는 방법은 할당 반복 문제 및 병목 현상 문제를 어느 정도 해결할 수 있다.

그러나 이 방법은 입력 스트림의 양이 적은 스트림 큐에도 불필요하게 많은 페이지를 할당할 수 있어, 스트림 큐의 페이지 이용률을 크게 감소 시킬 수 있다. 그리고 이 두 기법은 데이터 스트림의 입력속도 및 질의 지연에 관계없이 페이지를 할당하기 때문에 데이터 스트림의 입력 속도가 빠른 특정 지연된 질의의 스트림 큐가 페이지를 대량으로 점유하는 문제가 발생할 수 있다.

그 밖에 저장공간 초과 문제를 해결하기 위한 방법으로 부하제한기법(Load Shedding)이 연구 되었다[7,8,9]. 부하제한은 스트림 큐에 여분의 공간이 없는 경우 스트림 큐에 저장된 데이터의 일부를 제거하는 방법이다. 이 방법은 아직 질의 처리에 이용되지 않은 데이터를 제거하기 때문에 질의의 정확도를 감소시킨다. 이 부하제한은 메모리 풀의 가용 페이지가 존재하지 않아 더 이상 스트림 큐에 새로운 페이지를 할당할 수 없는 경우 발생한다. 그러므로 스트림 큐가 사용되지 않는 페이지를 많이 점유하면, 메모리 풀의 가용 페이지가 빠르게 감소할 수 있고, 결과적으로 이는 불필요한 부하제한을 발생시켜 질의 정확도를 감소시킬 수 있다.

본 논문에서는 메모리 풀의 페이지를 할당 및 해제하는 빈도수를 감소시켜 재할당 시간과 메모리 풀의 병목현상을 감소시키고, 스트림 큐의 이용률을 최대한 높여 불필요한 부하제한을 감소시키는 메모리 관리 기법을 제안한다. 이 기법은 스트림 큐의 저장 공간을 초과하는 경우 즉시 새로운 페이지를 할당하고, 사용되지 않는 페이지가 발생해도 즉시 해제하지 않는 게으른 관리 방식을 사용한다. 페이지 해제는 일정 주기 마다 각 스트림 큐에 할당되는 페이지들의 이용률과 데이터 스트림의 입력 속도를 반영하여 이용률이 감소하는 페이지를 선별적으로 해제한다.

이 기법은 페이지 해제를 주기적으로 수행하기 때문에 기존 기법보다 페이지의 해제 빈도수를 크게 감소 시킬 수 있고, 스트림의 입력속도가 감소 했다 증가하는 경우 아직 할당된 페이지를 해제하지 않았으므로 페이지를 재할당하는 빈도수 역시 크게 감소 시킬 수 있다. 이와 같이 이 기법은 페이지 할당 해제 빈도수가 데이터 스트림 입력의 속도 변화에 크게 영향을 받지 않는다.

그러나 제안 기법은 페이지 해제를 지연시키기 때문에 스트림 큐가 불필요한 페이지를 점유하여 메모리 풀의 가용 페이지 부족 현상이 발생할 수 있다. 이 문제는 가용 페이지 부족 현상 발생 시 데이터 스트림의 입력속도가 감소하는 스트림 큐의 사용되지 않는 페이지를 우선적으로 해제하여 해결 할 수 있다. 데이터 스트림의 입력속도는 각 스트림 큐의 페이지의 할당 해제 발생을 가지고 예측 할 수 있다. 또한 이 기법은 질의의 지연여부를 파악하여 지연된 질의에 부하제한을 적용시켜 특정 스트림 큐가 계속적으로 페이지를 할당하여 대량의 페이지를 독점적으로 점유하는 문제를 해결할 수 있다. 이는 메모리 풀의 페이지를 모든 등록된 질의에 적절하게 분배할 수 있고, 결과적으로 특정 질의의 페이지 점유로 인해 다른 질의가 사용할 페이지 부족 문제를 해결하여 부하제한 발생을 최소화 하여 전체적인 질의의 정확도를 크게 향상시킬 수 있다.

2. 관련 연구

2.1 연속질의와 스트림 큐

데이터 스트림을 처리하기 위해 브랜다이스와 브라운 대학이 연구한 Aurora, 스탠포드 대학의 STREAM, 시공간 데이터 스트림을 위한 SOLE등 다양한 데이터 스트림 관리 시스템이 연구되고 있다[6,13,14,15]. 이 데이터 스트림 관리 시스템은 시스템에 입력되는 데이터를 메모리 상에서 실시간으로 처리하는 연속질의를 사용한다. 기존의 질의는 디스크에 저장된 정적인 데이터를 처리하는 반면, 연속질의는 데이터 스트림과 같이 실시간으로 변하는 데이터를 처리한다.

[그림 1]과 같이 기존의 데이터 베이스 관리 시스템(DBMS)은 디스크에 저장된 대용량 데이터로부터 질의를 처리하기 때문에, 디스크에 저장된 데이터 페이지의 일부를 버퍼 풀(Buffer Pool)로 옮기고 다수의 질의가 이 버퍼 풀을 공유하여 처리 된다. 기존 DBMS는 데이터의 양이 많고 정적이기 때문에 이 방법을 주로 이용하였다. 반면, 데이터 스트림은 기존 DBMS의 데이터와 다르게 동적인 특성을 갖기 때문에 질의 별로 고유한 데이터 저장소인 스트림 큐를 가지고 이곳에 실시간으로 입력되는 데이터를 저장하고 즉시 처리하는 질의 처리 방식을 사용한다[6]. 그러므로 각 질의를 위한 저장공간이 필요하

고 이 저장공간의 크기를 최적으로 관리하는 기법이 필요하다.

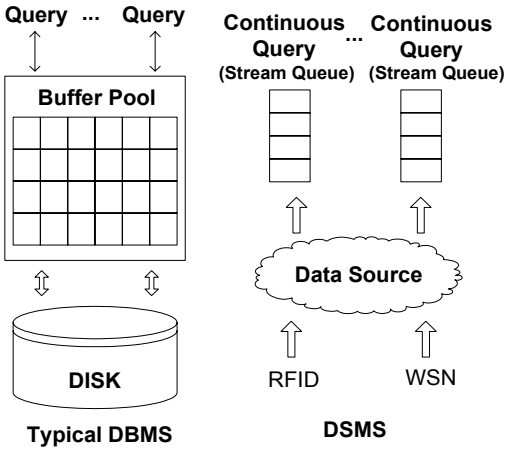

그림 1. DBMS와 DSMS에서 질의처리를 위한 메모리 공간의 사용 방법

2.2 동적 메모리 할당 기법

데이터 스트림은 입력되는 데이터 양이 실시간으로 변하므로, 이를 저장하기 위한 스트림 큐는 그 저장 공간이 동적으로 변해야 한다. 일반적으로 동적인 메모리 관리를 위해 메모리 풀을 사용한다. 메모리 풀은 생성될 때 대용량의 메모리를 할당 받고, 이 메모리를 고정된 크기의 페이지로 분리한다. 페이지의 할당 요청 시, 메모리 풀은 여분의 페이지들을 요청한 만큼 제공하고, 사용되지 않는 페이지의 해제 요청을 하면, 이 페이지를 여분 페이지로 관리한다. 대부분의 데이터 스트림 관리 시스템은 메모리 풀을 사용하여 스트림 큐를 동적으로 관리한다. STREAM과 SOLE은 페이지화된 버퍼 풀을 공유하여 시놉시스와 큐에 페이지를 제공하고[6,13], 상용시스템인 Coral8 역시 메모리 시스템을 지원한다[15].

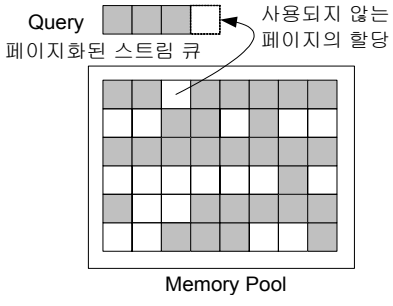

그림 2. 메모리 풀과 할당 방법

메모리 풀을 이용한 데이터 스트림 관리 시스템의 메모리 관리기는 동적 메모리 관리를 위해서 두 가지 정책을 사용할 수 있다. 첫 번째 정책은 스트림 큐의 공간을 초과할 때 마다 메모리 풀로부터 새로운 데이터 페이지를 계속적으로 할당 받고 페이지를 사용하지 않으면 즉시 해제하는 방법이다(DA: Dynamic Allocate). 특히 Aurora의 메모리 관리기인 ASM(Aurora Storage Manager)는 초기에 하나의 페이지를 할당 받고 메모리를 초과하는 데이터가 입력되면 즉시 새로운 페이지를 할당한다[14].

그러나 이와 같은 방법은 몇 가지 문제가 있다. 데이터의 입력의 변화에 따라 페이지를 계속적으로 할당해야 하여 할당 비용을 발생시킬 수 있다. 특히 질의 스케줄을 사용하는 경우, 질의가 대기상태이면 데이터가 순간적으로 누적되어 다수의 페이지를 요구하게 되고, 질의가 처리되어 데이터를 모두 처리하면 모든 페이지를 다시 해제하기 때문에, 반복적인 할당 해제 비용이 크게 요구된다[10,11].

두 번째 방법은 메모리를 초기에 스트림 큐가 생성될 때 페이지를 대량으로 할당하는 방법이다(MA : Maximum Allocate). 대부분의 데이터 관리 시스템이 초기 할당되는 페이지의 양을 설정 할 수 있다. Coral8과 같은 경우도 초기에 할당되는 메모리 크기를 크게 할당하여 처리 속도를 향상하는 옵션이 있다[15]. 이때도 스트림 큐를 초과하는 데이터가 입력되면 페이지를 할당하게 된다. 그러나 이 방법은 스트림 큐가 초과되는 상황이 거의 발생하지 않기 때문에, 할당해제의 수행 횟수를 크게 감소 시킬 수 있다. 하지만 스트림의 입력의 특징 때문에, 적절한 스트림 큐의 최대치를 예측하기 어렵고, 스트림의 입력량이 감소할 때 사용하지 않는 불필요한 페이지를 스트림 큐가 보유하기 때문에 메모리 이용률이 크게 감소한다.

2.3 부하제한기법

때때로 데이터 스트림은 매우 많은 데이터가 입력되어 메모리 풀의 가용 페이지 모두 사용할 수 있다. 이 경우는 대량의 데이터 스트림이 입력되어 질의가 지연되고, 더 이상 데이터를 스트림 큐에 저장할 수 없는 경우이다. 이 경우는 스트림 큐에 저장된 데이터의 일부를 버리고, 그곳에 새로운 데이터 스트림을 저장할 수 있도록 하는 부하제한기법을 이용한다[5,7,8,9].

부하제한기법은 데이터의 일부를 버리기 때문에 메모리 풀의 가용 페이지가 없는 경우에 효과적으로 대응 할 수 있다. 그러나 부하제한기법은 유실된 데이터로 인해 질의의 정확도를 감소시킬 수 있다. 특히 질의 결과에 영향을 미칠 수 있는 중요한 데이터를 버린 경우 질의 정확도를 크게 감소시킬 수 있다. 그러므로 부하제한의 발생 빈도를 줄이기 위해 스트림 큐에 할당된 메모리의 이용률을 최대한 높이는 방법이 필요하다.

3. 적응적 메모리 관리 기법

본 장에서는 연속질의에서 스트림 큐에 할당된 페이지

화된 메모리의 이용률과 데이터 스트림의 유입속도의 변화를 고려하여 동적으로 페이지의 할당 및 해제를 지원하는 AV 메모리 관리 기법을 설명한다.

본 AV 기법은 페이지의 할당 해제 빈도수를 감소시키기 위해 “즉시 할당-지연된 해제(Direct Allocation Delayed Release)”를 이용한다. 스트림 큐의 저장공간을 초과하는 데이터가 입력되면 메모리 풀로부터 즉시 새로운 페이지를 할당 받고, 이 데이터가 처리되어 사용하지 않는 페이지가 발생해도 이를 즉시 해제하지 않고 특정 주기 마다(여기서는 1분으로 가정) 스트림의 유입속도가 감소하는 질의의 이용되지 않는 페이지를 선별적으로 해제한다. AV 기법은 매 주기마다 스트림 큐에 할당된 페이지의 이용상태를 이용하여 입력 스트림의 증감속도를 파악할 수 있다.

페이지 해제의 지연은 질의 마다 사용하지 않는 페이지를 보유할 수 있기 때문에, 메모리 풀의 가용 페이지를 빠르게 감소 시킬 수 있다. 그래서 메모리 풀의 가용 페이지가 존재하지 않을 때, 이미 할당된 페이지 중 사용되지 않는 페이지를 해제하여 할당을 필요로 하는 질의가 이용하도록 할 수 있다. 해제 할 페이지는 가장 이용확률이 낮은, 즉 다음 주기 때 해제될 가능성이 높은 것이 적합하다. 그래서 AV기법은 입력속도가 감소(해제단계의 스트림 큐) 중이고 가장 많은 페이지가 할당된 스트림 큐에서 사용되지 않는 페이지를 선택하여 해제한다.

그런데 위와 같은 방법은 한가지 문제점을 가지고 있다. 일반적으로 가용페이지가 부족한 경우는 대부분 데이터 스트림이 대량으로 입력되어 질의가 지연될 때 발생한다. 이 경우 모든 질의의 데이터 스트림이 대량으로 입력되는 것이 아니고 특정 질의의 데이터 스트림이 대량으로 입력되는 경우가 많다. 그리고 이와 같이 대량으로 데이터 스트림이 입력되는 질의가 대부분 질의 지연 현상을 일으키게 된다. 이와 같이 질의가 지연되고 데이터의 순간 누적률이 높은 질의에 계속적으로 페이지를 할당하게 되면 페이지가 특정 질의에 집중적으로 할당될 수 있다. 특정 질의에 페이지가 편중되는 이 현상은 지연되지 않거나 적은 페이지로 처리 가능한 다른 질의들이 페이지를 할당 받지 못하여 부하제한을 야기하고 전체적인 질의 정확도를 크게 감소 시킬 수 있다.

그러므로 가용페이지가 부족한 상황에 질의가 지연되는 경우 이 질의에는 페이지를 할당하지 않고 부하제한을 적용한다. 질의 지연 여부의 결정은 질의에 포함된 슬라이딩 윈도우를 통해 알 수 있다. 슬라이딩 윈도우의 실행 영역은 특정 유한한 범위로 고정되어 있는데, 이 슬라이딩 윈도우가 아직 과거 시간에 있거나 슬라이딩 윈도우의 범위를 벗어나는 데이터가 대량으로 입력되는 것을 통해 질의 지연 여부를 판단할 수 있다.

지금까지 본 논문에서 제안하는 이용률을 기반으로 한 AV기법에 대해 전반적으로 설명하였다. 그리고 3.1절은 AV기법에 관련된 용어 및 메모리 관리구조에 대해서 설명하고 3.2절은 AV기법에서 메모리 할당 및 해제 방법에 대해서 설명한다. 마지막으로 3.3절에서는 가용메모리 부족상황에서의 메모리 할당 및 해제 기법에 대해서 설명한다.

3.1 GeoSensor 네트워크에서 공간 조인

본 절에서는 AV기법에서 사용되는 각종 구조들과 이용률 및 기타 관련 정보를 나타내기 위한 용어들에 대해서 설명한다.

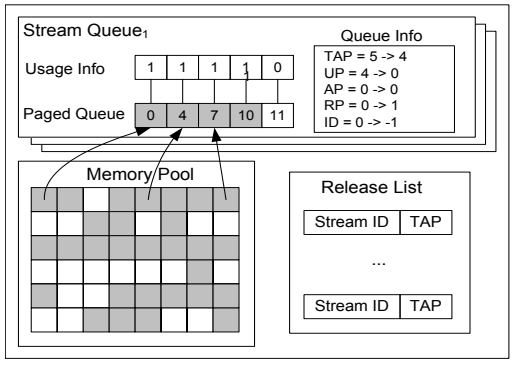

그림 3. AV기법을 이용한 메모리 관리자의 구성요소

[그림 3]은 AV관리기법을 사용하는 메모리 관리자의 구성요소를 묘사하고 있다. 이 메모리 관리자는 질의마다 할당되는 스트림 큐(Stream Queue)와 페이지를 미리 할당하고 관리하는 메모리 풀(Memory Pool), 페이지 해제를 위한 해제 리스트(Release List)의 3가지로 구성된다.

메모리 풀은 일반적인 메모리 풀과 동일하게 대량의 페이지를 미리 할당하고 할당 및 해제 요청에 따라 페이지를 관리한다. 다음으로 스트림 큐는 페이지로 구성된 큐(Paged Queue)와 각 페이지의 사용상태를 저장하는 사용정보(Usage Info), 주기적인 페이지 해제를 위해 큐의 상태 변화를 관리하는 큐 정보(Queue Info)로 구성된다. 그림에서 스트림 큐의 각 페이지는 번호를 가지고 있는데 이것은 메모리 풀 내의 각 페이지의 번호를 구분하는 페이지 ID이다. 이 그림에서 큐는 5개의 페이지로 구성되어 있고 4개의 페이지(0,4,7,10)가 사용 중이고 1개의 페이지(11)이 사용되지 않았다. 사용정보는 1인 경우 사용 중인 페이지를, 0인 경우 사용되지 않은 페이지를 나타낸다. 사용정보는 주기적으로(1분) 갱신 된다. 마지막으로 큐 정보는 5개의 변수로 구성되는데 각 변수의 설명은 다음과 같다.

할당 된 총 페이지의 수 : *TAP*(Totally Allocated Page)
현재 사용된 페이지의 수 : *UP*(Used Page)
해당 주기에 할당된 페이지 수 : *AP*(Allocated Page)
해당 주기에 해제된 페이지 수 : *RP*(Released Page)
증가-감소 상태 : *ID*(Increase Decrease)

TAP는 현재 큐에 할당된 총 페이지 수를 나타낸다. 그리고 UP는 현재 주기 동안 사용된 페이지의 수이다. 현재 사용 중이지 않아도 해당 주기 동안 한번이라도 페

이지가 사용되면 사용된 페이지라고 간주한다. AP는 해당 주기 동안 새롭게 할당된 페이지가 있으면 그 수를 기록한다. RP는 주기 별로 큐를 해제할 때 해제되는 페이지가 있으면 그 수를 기록한다. 마지막으로 ID는 증가 감소 상태를 나타내는데, 1이면 현재 주기 동안 할당이 된 경우, 0이면 할당 및 해제가 일어나지 않은 경우, -1이면 해제가 된 경우를 나타낸다.

[그림 2]의 큐 정보에서 TAP는 5인데, 현재 5번째 페이지가 사용된 적이 없으므로, 이 상태에서 주기적인 메모리 해제를 위해 큐 정보를 검사하면 11번 페이지를 해제해야 한다. 이때 TAP는 4로 변화된다. 그리고 현재 UP는 4개의 페이지만 사용 중이므로 4이고 큐 정보 검사 후 0으로 초기화 된다. AP는 현 주기 때 할당된 페이지가 없으므로 0이다. RP는 0인데, 큐 정보 검사 후 한 개의 페이지를 해제하므로 1로 변한다. 현재 ID는 0(증가 감소하지 않음)인데 한 개의 페이지를 해제하는 감소상태이므로 -1로 변한다.

3.2 메모리 할당 및 해제

본 절에서는 메모리 풀의 가용페이지가 충분한 상황(가용 페이지가 임계 치 이하)에서 할당 및 해제하는 방법을 설명한다. 이 상황에서 스트림 큐를 초과하는 데이터가 입력되면 이 기법은 즉시 새로운 페이지를 할당한다.

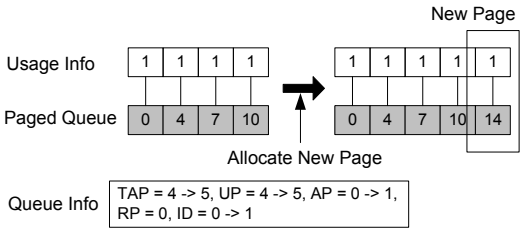

그림 4. 스트림 큐에 새로운 페이지의 할당

[그림 4]는 스트림 큐에 새로운 페이지를 할당하는 과정을 묘사한다. 현재 스트림 큐는 4개의 페이지를 가지고 있다. 이때 스트림 큐를 초과하는 데이터가 입력되면, 스트림 큐는 메모리 풀로부터 새로운 페이지를 하나 할당받고(14번 페이지) 이를 스트림 큐의 뒤에 연결한다. 그리고 새롭게 할당된 다섯 번째 페이지의 사용정보(Usage Info)는 1로 설정된다. 그리고 큐 정보는 위와 같이 페이지가 하나 증가했으므로 TAP는 5로, UP는 5로, AP는 1로 변경되고, 페이지의 수가 증가하고 있는 증가 상태이므로 ID는 1로 변경된다.

사용되지 않는 페이지를 해제하기 위해, 스트림 해제 프로세스는 매 주기 마다 큐 검사를 하여 큐의 사용되지 않는 페이지가 임계치(전체의 10%)를 초과하는 스트림 큐의 페이지들을 해제한다.

[그림 5]은 페이지가 지속적으로 할당되다가 사용량이 감소하여 페이지의 일부를 해제해야 하는 스트림 큐의 상태를 묘사하고 있다. 이 스트림 큐는 현재까지 20개의 페이지가 할당되어 있다. 그리고 현 주기에서 페이지가 12개만이 사용되었다. 그러므로 이용되지 않는 페이지의 비율은 40%(8/20 * 100)이므로 페이지가 해제 되어야 한다. 스트림 큐는 페이지 해제 알고리즘 수행 후 TAP는 12, RP는 8, ID는 -1로 변경된다.

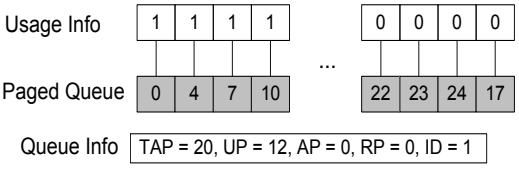

그림 5. 페이지가 해제되어야 하는 스트림 큐

3.3 가용페이지 부족 시 메모리 할당 및 해제

3.2절에서 설명한 메모리 할당 방법은 이전에 스트림 큐에 할당되었지만 현재 사용하지 않는 페이지가 존재해도 이를 즉시 삭제하지 않기 때문에 메모리 풀의 가용페이지가 부족한 경우 이 사용하지 않는 페이지를 이용하여 페이지 할당에 이용할 수 있다. 그러나 가용페이지가 부족할 때 무차별적으로 페이지를 할당하게 되면, 지연되는 특정 질의가 페이지를 독점적으로 점유할 수 있기 때문에 질의의 지연 상태에 따라 페이지를 할당해야 한다. 질의 지연은 질의에 포함된 슬라이딩 윈도우의 실행 영역을 가지고 판단할 수 있다.

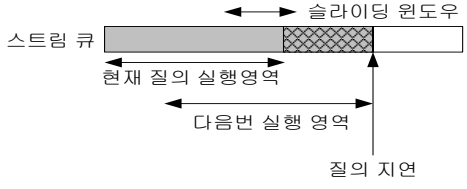

그림 6. 연속질의의 슬라이딩 윈도우와 질의지연

연속질의는 슬라이딩 윈도우가 이동하면서 슬라이딩 윈도우에 포함되는 스트림을 질의처리에 이용한다. [그림 6]는 슬라이딩 윈도우의 현재 실행 영역과 다음 실행 될 영역을 보여 주고 있다. 이때 다음 번 실행 영역을 초과하는 데이터가 스트림 큐에 누적되면, 현재 연속질의가 최근 데이터를 처리하지 못하는 지연상태이다. 질의 지연은 질의처리 시간이 길거나 데이터 스트림이 대량으로 입력되는 경우 주로 발생한다. 특히 질의 지연이 지속되는 경우 처리되지 못한 스트림이 고속으로 누적되기 때문에, 스트림 큐는 대량으로 페이지의 할당 요청을 할 수도 있다.

이와 같이 특정 지연되는 질의를 위해서 페이지를 집중적으로 할당하면, 지연되지 않는 질의가 페이지를 할당받지 못하는 문제가 발생 할 수 있다. 그래서 질의가 지연되고 전체 메모리 풀의 가용 페이지가 부족하게 된 경우(임계치를 초과)에 메모리 풀은 가용 페이지를 할당하

지 않고 부하를 분산 시키는 방법을 사용할 수 있다(부하 제한 기법은 본 논문에서 다루지 않는다). 만약 지연되지 않는 질의이면, 입력속도가 감소 중이고 많은 페이지가 할당된 스트림 큐의 페이지를 해제하고 재할당 해주고, 입력속도가 감소하지 않는 스트림 큐가 존재하지 않으면 메모리 풀의 남은 가용 페이지를 할당해 준다.

입력 속도가 감소하는 스트림 큐의 페이지를 고속으로 검색하고 해제하기 위해 메모리 관리자는 해제 리스트를 구축한다. 해제되는 페이지는 해제 리스트의 첫 번째 노드로 선택된다. 해제 리스트는 메모리 풀의 가용페이지가 부족할 때 구축되고 매 주기마다 업데이트 되는데, 입력속도가 감소하는 상태의 스트림 큐를 가지고 할당된 페이지 수를 기준으로 정렬되어 구축된다. 해제 리스트를 구축하는 것은 페이지 수를 기준으로 정렬하는 것과 동일하다. 그러므로 여기서는 고속으로 정렬이 가능한 힙정렬을 이용하여 입력 속도가 감소 중인 스트림 큐를 할당된 페이지 수를 기준으로 정렬한다. 힙정렬을 이용하면 nLog(n) 시간에 정렬이 해제리스트가 구축될 수 있다.
그럼 이제부터 구체적으로 페이지 할당 알고리즘 및 주기적인 페이지 해제 및 해제 리스트 구축 알고리즘에 대해서 설명한다.

알고리즘 1. 페이지 할당 알고리즘

```
Input
 SQ : StreamQueue
Output
 True or False

Procedure PageAllocate()
Initialize
01 usage = MemoryPool.GetUsageRate()
02 threshold = 0.9 //90%로 임계치 설정
Begin
01 PageAllocate()
02 {
03  if( usage > threshold )
04  {
05    if( SQ.GetLastTime() > SQ.NextWinTime() )
06      return false;
07    if( ReleaseList.IsEmpty() == false )
08      Page = ReleaseList.GetMaxHeap()
09    else
10    {
11      if( MemoryPool.IsEmpty() == true )
12        return false
13      else
14        Page = MemoryPool.AllocatePage()
15    }
16  else
17  {
18    if( MemoryPool.IsEmpty() == true )
19      return false
20    else
21      Page = MemoryPool.AllocatePage()
22  }
23  SQ.AddPage(Page)
24  SQ.TAP = SQ.TAP + 1
25  SQ.UP = SQ.UP + 1
26  SQ.AP = SQ.AP + 1
27  SQ.ID = 1
28  SQ.bUsed[TAP-1] = 1
29  return true;
30 }
End
```

<알고리즘 1>은 페이지 할당 알고리즘으로 메모리 풀의 메모리가 충분할 때와 충분하지 않은 상황에 따라 스트림 큐에 페이지를 할당하는 방법을 보여준다. 알고리즘은 먼저 메모리 풀의 사용량인 usage를 초기화 하고, threshold에 메모리 풀의 가용임계치를 설정한다. 3줄은 현재 메모리 풀의 사용량이 임계치(90%)를 초과했는지 확인한다. 90% 초과시 5줄은 질의 지연을 확인한다. 현재 스트림 큐에 저장된 마지막 튜플의 시간(GetLastTime)이 현재 실행되는 윈도우의 다음 실행시간(NetWinTime)을 초과하는지 확인하여 초과시 페이지를 할당하지 않고 false을 반환한다. 만약 질의가 지연되지 않는 경우는 해제리스트에서 가장 할당된 페이지가 많은 첫 번째 스트림 큐의 페이지를 선택한다(GetMax- Heap : 해제리스트는 힙정렬 되어있다). 만약 해제리스트에 페이지가 존재하지 않으면 14줄과 같이 메모리 풀에서 가용 페이지가 선택 된다(AllocatePage). 16줄은 메모리 풀의 가용 페이지량이 임계치를 초과하지 않았을 때 처리하는 것으로, 21줄과 같이 메모리 풀에서 가용페이지 하나가 선택된다.(AllocatePage) 그리고 23줄은 선택된 페이지를 가지고 스트림 큐에 추가하고, 스트림큐의 전체 할당량(TAP), 사용량(UP), 할당갯수(AP)를 증가시키고, 증감상태(ID)를 증가상태(1)로 설정한다.

알고리즘 2. 페이지 해제 알고리즘

```
Input
 SL : StreamQueue Array
Output
 Nothing

Procedure PagesRelease()
Initialize
01 Threshold = 0.9
```

```
Begin
01 ReleasePages()
02 {
03  for( i=0; i<StreamQueueCnt; i++ )
04  {
05   SL[i].RP = 0
06   Usage = SL[i].UP / SL[i].TAP * 100
07   if( Usage < Threshold )
08   {
09    SL[i].ID = -1
10    for( j=SL[i].TAP-1; j>= SL[i].UP; j-- )
11    {
12     Page = SL[i].ReleaseLastPage()
13     SL[i].TAP = SL[i].TAP - 1
14     SL[i].RP = SL[i].RP + 1
15     MemoryPool.ReleasePage(Page)
16    }
17    for( j=0; j<SL[i].TAP; j++ )
18        SL[i].bUsed[j] = 0
19    ReleaseList.InsMax(SL[i].QueueID,SL[i].TAP)
20    }
21    else
22     SL[i].ID = 0
23    SL[i].AP = 0
24    SL[i].UP = 0
25   }
26 }
End
```

<알고리즘 2>는 페이지 해제 알고리즘으로 주기적으로 사용되지 않는 페이지를 해제하기 위해 사용된다. 먼저 6~7줄에서 시스템에 등록된 모든 스트림 큐의 사용량이threshold(90%)보다 낮은 것을 찾는다. 낮은 스트림 큐가 존재하면 10~16줄에서 증감상태(ID)가 감소(-1)로 설정되고, 사용되지 않은 모든 페이지가 해제된다. 그리고 15줄에서 해제된 각 페이지는 메모리 풀로 반환된다. 해제가 진행된 스트림 큐는 페이지가 감소상태이므로 19줄에서 해제리스트(ReleaseList)에 추가된다. 이때 해제리스트는 스트림 큐의 할당량을 기준으로 힙 정렬된다. 마지막으로 스트림 큐의 할당량(AP) 및 사용량(UP)은 0으로 초기화 된다.

4. 성능분석

4.1 평가환경

실험 평가에 사용된 시스템 환경은 CPU가 팬티엄 4 3.0 GHz이고 메모리는 4GB이다. 실험에 사용된 데이터는 상용 이벤트 처리 시스템인 Coral 8에서 제공하는 데이터 셋을 이용한다.[10] Coral8은 금융(Finace)데이터 및 RFID데이터, 센서데이터, 리눅스 로그데이터(Syslog) 등 다양한 데이터 셋을 제공한다. 실험 데이터의 크기는 4~100바이트 정도로 다양하다. 실험에 사용되는 데이터는 이 데이터 셋을 초당 1,000건에서 100,000건의 다양한 속도로 입력하는 데이터 전송프로세스에 의해 생성된다. 메모리 풀의 초과를 빠르게 측정하기 위해 메모리 풀은 64MB로 할당하고 페이지의 크기는 512Byte로 설정하였다. 스트림 큐마다 설정되는 기본 페이지는 2개가 할당되어 총 크기는 1024Byte이다.

4.2 성능평가

본 논문에서 제안하는 기법의 우수성을 증명하기 위해, 기존의 일반적인 페이지 할당 기법인 DA와 MA기법과 제안기법(AV)을 가지고 페이지 할당 및 해제 빈도와 부하제한 발생 빈도와 각 질의 별 할당된 페이지의 이용률을 비교한다. 데이터 입력속도가 실시간으로 변하는 다양한 스트림 입력으로 인해 스트림 큐에 데이터가 누적되어 추가적인 페이지의 할당을 요구하게 될 때, 메모리 할당과 해제의 빈도수를 측정한다. 이 할당 해제 빈도수를 통하여 메모리 풀에서의 병목현상 및 질의지연이 감소하는 것을 보일 수 있다. 또한 데이터 입력속도가 증가하여 메모리 풀의 가용페이지가 부족하게 되는 경우, 부하제한을 수행하는 질의의 수를 측정한다. 부하제한의 수를 측정하여 전체적인 질의 정확도가 증가하는 것을 보일 수 있다.

4.3 할당해제 빈도 수

AV기법은 페이지의 할당 및 해제의 빈도 수를 크게 감소 시킬 수 있다. 이를 측정하기 위해, 본 실험은 다양한 실험 데이터를 사용하는 질의의 수를 증가시키면서 할당 및 해제의 빈도수를 측정하였다. 동시에 실행되는 질의 수는 1개에서 100개까지 다양하게 측정하였다.

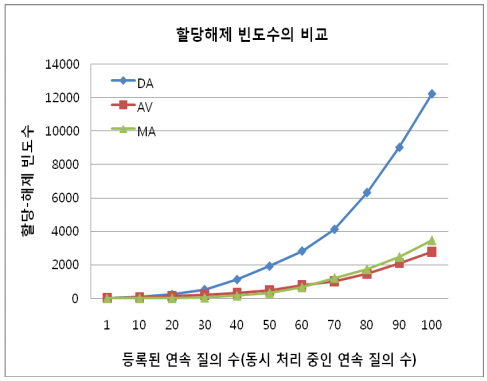

그림 7. 할당해제 빈도수의 비교 분석

[그림 7]의 실험 결과를 통해, 등록된 질의수가 증가하

면 AV기법이 할당 및 해제 빈도수가 가장 적게 발생함을 알 수 있다. AV기법은 해제를 지연시키고 1분간 평균적인 이용률을 기반으로 해제하기 때문에, 즉시 해제하는 DA기법보다 할당 및 해제 빈도수가 크게 감소한다. 특히 질의 수가 증가하여 질의 스케줄링을 하게 되면 질의가 순간적으로 실행이 중지되는 경우가 많이 발생하기 때문에 각 질의 별 스트림 큐의 데이터 변화량이 급증하게 된다. 그래서 즉시 할당 해제 하는 DA기법은 질의 수가 증가함에 따라 할당 및 해제 빈도수가 크게 증가하게 된다. MA기법은 질의 수가 적은 경우는 데이터가 거의 누적되지 않기 때문에 할당 및 해제가 거의 발생하지 않는다. 그러나 질의수가 증가하여 데이터가 누적되기 시작하면 DA와 동일하게 할당 및 해제를 하기 때문에 빈도수가 크게 증가하기 시작한다.

4.4 부하제한 발생 빈도 수

AV기법은 질의 지연을 고려하여 페이지를 질의 별로 고루 분배하기 때문에 전체적인 메모리 이용률이 크게 향상되어 부하제한 발생률을 최소화 할 수 있다. 이와 같이 부하제한 발생률의 감소를 보이기 위해 본 실험에서는 등록된 질의의 수를 증가시키면서 부하제한의 발생 빈도수를 측정하였다. 동시에 실행되는 질의는 1에서 100개까지 다양하게 측정하였다.

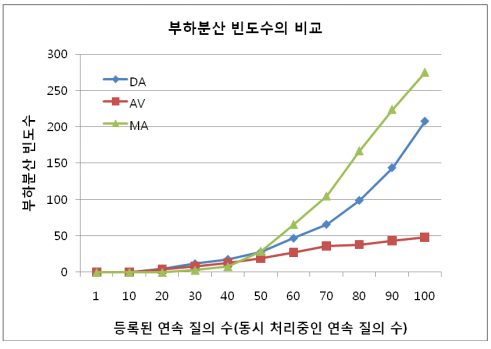

그림 8. 부하제한 빈도수의 비교 분석

[그림 8]을 통해 등록된 질의의 수가 증가하여 40개 이상이 처리 될 때, AV기법이 DA와 MA 기법보다 부하제한 빈도수가 크게 감소함을 알 수 있다. AV기법은 전체적인 가용페이지 부족을 야기하는 특정 스트림 큐가 지연되면 이 스트림이 독점적으로 페이지를 점유할 수 없도록 하기 때문에 전체적으로 페이지가 고르게 분배된다. 결과적으로 가용페이지가 부족하면 그 원인에 해당하는 질의를 부하제한 시켜 빠르게 가용페이지를 확보하고, 다른 질의가 페이지 부족으로 인해 불필요한 부하제한 발생빈도를 크게 감소시켜 DA와 MA기법보다 전체적인 부하제한 빈도수를 크게 감소시킬 수 있다. MA기법은 질의가 50개 이상 동시 처리 될 때, 부하제한 빈도수가 급격하게 증가하는데, 이것은 MA기법은 사용하지 않는 페이지가 많이 할당되어 메모리 부족현상이 발생하기 때문이다. AV기법은 이 부하제한의 감소를 통해 전체적인 질의 정확도가 크게 향상 될 수 있다.

4.5 스트림 큐에 할당된 페이지의 이용률

AV기법은 페이지 해제를 지연 시키지만, 주기적으로 해제를 하기 때문에 전반적으로 메모리 이용률이 높다. 본 실험에서는 평균 메모리 이용률을 측정하기 위해 등록된 질의의 수를 증가시키면서 각 질의에 할당된 페이지들의 사용량을 측정하였다. 동시에 실행되는 질의는 1에서 100개까지 다양하게 측정하였다.

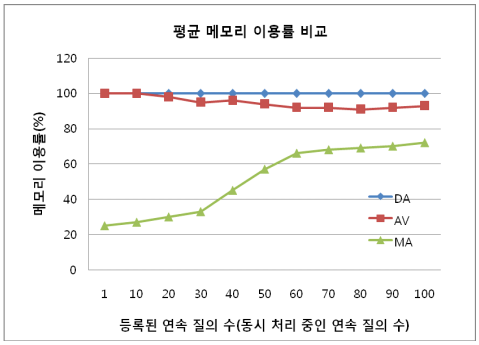

그림 9. 평균 메모리 이용률의 비교 분석

[그림 9]를 통해 등록된 질의의 수가 증가해도 AV기법의 메모리 이용률은 거의 90%이상을 유지함을 알 수 있다. DA기법은 동적으로 즉시 할당 및 해제를 하기 때문에 100%의 정확도를 유지한다. MA기법은 초기에 많은 페이지를 스트림 큐에 할당하므로 전반적으로 메모리 이용률이 낮은 것을 알 수 있다. 데이터가 많이 누적되는 질의수가 40~50인 구간에서 큰 폭으로 이용률이 증가하지만 80% 이상으로 증가하지는 않고 있다. AV기법은 DA기법에 비해 이용률이 다소 낮지만 90%정도의 우수한 이용률을 보이고, 만약 메모리 풀의 가용페이지가 부족한 경우에도 스트림 큐에 할당된 사용되지 않는 페이지를 제공할 수 있기 때문에 DA기법 보다 다소 낮은 이용률이 부하제한 발생에 거의 영향을 미치지 않는다.

5. 결론

본 논문에서는 데이터 스트림 환경에서 기존에 주목 받지 못한 메모리 관리자에 초점을 맞추어 빈번한 페이지 할당 및 해제 빈도수를 크게 감소시키고, 질의상태에 따라 차등적인 페이지 분배를 통하여 전체적인 질의 정확도를 향상 시킬 수 있는 이용률 기반의 메모리 관리 기법을 설명하였다. 본 기법에서 제안한 이용률을 기반으로 한 할당 및 해제를 통하여 메모리 관리자에서 발생한 불

필요한 병목현상을 크게 감소시킬 수 있음을 보였고, 차등적인 페이지 할당으로 인해 부하제한을 크게 감소시킬 수 있음을 보였다.

향후 연구로는 메모리 관리자에서 할당하는 페이지의 물리적인 위치를 근접시키는 것이다. 빈번한 할당해제로 인해, 스트림 큐 상에 페이지들의 물리적인 간격이 멀어져 발생하는 불필요한 메모리 접근 시간을 감소 시킬 수 있다. 특히 수많은 질의처리가 발생하는 스트림 큐에서는 큰 성능 향상을 얻을 것으로 예상된다.

참 고 문 헌

[1] R. Motwani, J. Widom, A. Arasu, "Query Processing, Resource Management, and Approximation in a Data Stream Management System," CIDR, 2003.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, "Models and Issues in Data Stream Systems," PODS, 2002, pp. 1-16.

[3] D. J. Abadi, D. Carney, U. Centintemel, M. Cherniack, C. Convey, "Aurora: A New Model and Architecture for Data Stream Management," VLDB Journal, 2003, pp. 120-139.

[4] S. Chandrasekharan, FranklinJ. "Streaming queries over streaming data," VLDB Conf, 2002, pp. 203-214.

[5] L. Golab, M. Tamer Ozsu, "Issues in Data Stream Management," SIGMOD Record. ACM. Vol.32 No. 2, 2003, pp. 6-14.

[6] A. Arasu, S. Babu, J. Widom, "The CQL continuous query language: semantic foundations and query execution," VLDB Journal, Vol.15 No. 2, 2006, pp. 121-142.

[7] N. Tatbul, U. Cetintemel, S. Zdonik, M. Chemiack, M. Stonebraker, "Load Shedding in a Data Stream Manager," VLDB Conf, 2003, pp.309-320.

[8] N. Tatbul, S. Zdonik, "Window-Aware Load Shedding for Aggregation Queries over Data Streams," VLDB, 2006, pp. 799-810.

[9] B. Babcock, M. Datar, R. Motwani, "Load shedding for Aggregation Queries over Data Streams," ICDE, 2004, pp.1-12.

[10] B. Babcock, S. Babu, M. Datar, "Chain: Operator Scheduling for Memory Minimization in Data Stream Systems," ACM SIGMOD, 2003, pp. 253-263.

[11] M. Hammad, M. Franklin, "Scheduling for shared window joins over data streams," Submitted for publication, 2003, pp. 297-308.

[12] www.coral.com

[13] M.F. Mokbel,, W.G. Aref, "SOLE:scalable on-line execution of continuous queries on spatio-temporal data streams," VLDB Journal, 2008, pp. 971-995.

[14] J.A. Daniel, D. Carney, "Aurora:a new model and architecture for data stream management." VLDB Journal, 2003, pp. 120-139.

[15] Coral8 Administrator's Guide.

백 성 하

2005년 인하대학교 컴퓨터공학부 졸업 (이학사)
2007년 인하대학교 컴퓨터 정보공학과 (공학석사)
2007년~현재 인하대학교 컴퓨터 정보공학과(박사과정)
관심분야는 데이터 스트림, 클러스터, 위치기반 서비스

이 동 욱

2003년 상지대학교 전자계산공학과 (이학사)
2005년 인하대학교 컴퓨터 정보공학과 (공학석사)
2005년~현재 인하대학교 컴퓨터 정보공학과(박사과정)
관심분야는 공간데이터웨어하우스, 공간정보관리, 유비쿼터스 환경을 위한 SDBMS

어 상 훈

2003년 인하대학교 컴퓨터 정보공학과 (공학사)
2008년 인하대학교 컴퓨터 정보공학과 (공학박사)
2008년~현재 LG전자
관심분야는 공간SDBMS, 센서 네트워크

정 원 일

1998년 인하대학교 전자계산공학과 (공학사)
2004년 인하대학교 컴퓨터 정보공학과 (공학박사)
2004년~2006년 한국전자통신연구원 (선임연구원)
2007년~현재 호서대학교 정보보호학과 (전임강사)
관심분야는 데이터베이스, 데이터스트림, 이동객체

배 해 영
1974년 인하대학교 응용물리학과 (공학사)
1978년 연세대학교 대학원 전자계산학과 (공학석사)
1989년 숭실대학교 대학원 전자계산학과 (공학박사)
1985년 Univ. of Houston 객원교수
1992년~1994년 인하대학교 전자계산소 소장
1982년~현재 인하대학교 컴퓨터공학부 교수
1999년~현재 지능형 GIS연구센터 센터장
2000년~현재 중국 중경우전대학교 대학원 명예교수
2004년~2006년 인하대학교 정보통신대학원 원장
2006년~2009년 인하대학교 대학원장
관심분야는 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스