클라이언트-서버 데이터베이스에서의 온라인 클라이언트 재배치

박 용 범† · 박 제 호†

요 약

일반적인 2 계층을 기본으로 하는 데이터베이스 시스템은 병행 클라이언트가 많을 경우 성능면에서 그 한계를 가진다. 이 문제를 해결하기 위하여, 사용자들의 자료 이용의 유사성을 이용한 3 계층 데이터베이스 시스템이 제안되었다. 이 시스템에서 클라이언트들은 오프라인 형식의 클러스터들로 나뉘어지며, 가능한 경우 자료객체 요구는 서버와의 상호작용 없이 클러스터 내부에서 처리되게 된다. 이러한 구조는 서버와 클라이언트들 사이에 새로운 계층을 도입함으로써 가능해진다. 이 논문에서는 자료이용 유형이 변화하는 환경에서 클라이언트의 배치문제를 제시하고, 그 해결책으로 온라인 클라이언트 클러스터링을 제안한다. 이 방법은 환경 변화에 적응할 수 있는 시스템 재구성과 클라이언트의 재배치에 대한 필요성을 부각시킨다. 마지막으로 온라인 클라이언트 클러스터링의 유효성을 예시하고, 온라인 시스템의 재구성의 구현 가능성과 기술적 완성도를 검증한다.

Realignment of Clients in Client-server Database System

Young B. Park† · J. Park†

ABSTRACT

Conventional two-tier databases have shown performance limitation in the presence of many concurrent clients. To this end, the three-tier architecture that exploits similarities in client's object access behavior has been proposed. In this system, clients are partitioned into clusters, and object requests can be then served in inter-cluster manner. Introducing an intermediate layer between server(s) and clients enables this. In this paper, we introduce the problem of client realignment in which access behavior changes, and propose on-line client clustering. This system facilitates adaptive reconfiguration and redistribution of sites. The core issue in this paper is to demonstrate the effectiveness of on-line client clustering. We experimentally investigate the performance of the scheme and necessary costs.

키워드 : 3 계층 데이터베이스(Three-tier Database), 온라인 클라이언트 클러스터링(On-line Client Clustering)

1. 서 론

클라이언트-서버 데이터베이스는 기업정보 서버들이나, 웹을 이용한 관리/회계 시스템들, 전자상거래 엔진등에 꾸준한 추세로 그 이용이 증가하고 있다. 이러한 네트워크로 구성된 환경에서는 적정 효율을 제공하는 대용량 자료운영에 대한 중요성이 부각된다[5, 7, 10]. 클라이언트-서버 데이터베이스는 비교적 적은 물리적 자원으로 높은 성능개선을 구현하였다[6]. 하지만, 클라이언트-서버 데이터베이스는 개선된 시스템 효율에도 불구하고 범위성(scalability) 문제를 아직 내포하고 있다[4]. 이 문제에 대한 해결방법으로 클라이언트 클러스터링을 이용한 3 계층 클라이언트-서버 구조(3T-CSD)가 소개되었다[9]. 클라이언트 클러스터링은 유사한 자료객체 이용 유형을 보이는 클라이언트들을 하나의 논리적 클러스터로 구성한다는 기본 개념에서 출발한다. 우수한 클라이언트 클러스터링은 서로 상이한 클라이언트 그룹간의 상호작용을 줄임으로서 클라이언트-서버 데이터베이스의 범위성 문제를 해소하고, 동시에 클라이언트의 자료객체 요구를 클러스터 안에서 만족시킴으로써 향상된 트랜잭션 처리를 가능하게 한다. 이를 위해서 3T-CSD는 클라이언트 그룹들의 관리를 위해서 서버와 클라이언트들 사이에 디렉터리 계층을 사용한다. 따라서 3T-CSD의 성능은 클라이언트 클러스터링에 의해 많은 영향을 받는다. 변화하는 클라이언트들의 자료이용 유형을 분석하고 클러스터링

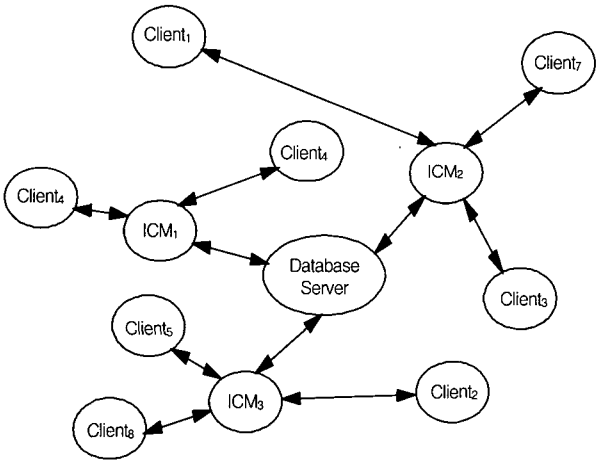
※ 본 연구는 2003학년도 단국대학교 교내 연구비의 지원으로 수행되었음.
† 정 회 원 : 단국대학교 컴퓨터과학 교수
논문접수 : 2002년 7월 30일, 심사완료 : 2003년 5월 12일

에 적용시키기 위해서는 주기적으로 축적된 자료이용 정보를 이용하는 오프라인 클러스터링을 수행할 수 있다[1-3].

본 논문에서는 온라인 클라이언트 클러스터링의 개념을 검토하고, 온라인 클러스터링을 위한 구조를 제시하고, 필요한 기능들을 지원하는 구성요소 구현 가능성과 기술적인 문제를 검증하였다. 구성요소로서 클라이언트 자료이용 유형 변화 인식, 클라이언트 재할당, 그리고 클러스터링 재구성을 중점적으로 논의하였다.

2. 다 계층 데이터베이스 구조

3T-CSD 환경에서 요구된 객체들은 객체전송 기반 방법론에 따라 서버로부터 공급되며 클라이언트는 공급된 객체들을 이용하여 트랜잭션을 수행한다[6]. 클라이언트는 공급된 객체들을 임시로 저장하고 관리하기 위해 메인 메모리와 하드디스크를 캐시 버퍼로 사용한다. 따라서 서버는 클라이언트들을 지원하기 위한 기본 기능들만을 수행한다. 아울러 서버는 로크 테이블과 관련 자료구조를 유지함으로써 자료에 대한 직렬가능 접근을 지원한다[8]. 또한 트랜잭션간 캐싱을 지원하기 때문에 서로 다른 트랜잭션들도 캐시에 저장되어 있는 객체들을 공유할 수 있다.



(그림 1) 3 계층 클라이언트-서버 데이터베이스 구조

3T-CSD의 기본 개념은 유사한 자료이용 유형을 갖는 클라이언트들을 논리적 클러스터로 구성한다는 것이다. 여러 가지 정적 클러스터링은 이미 검증되었다[9]. 유전자 알고리즘은 우수한 클러스터링을 생성하지만. 탐욕적 알고리즘과 비교하였을 때 유전자 알고리즘을 이용한 방법은 수행시간이 너무 길다는 단점이 있다[9]. 클러스터들의 개수와 클러스터링 구성이 확정되면, 중간 캐시 관리자(ICM)가 각 클러스터에 할당된다. 이 ICM들은 구조상 서버와 클라이언트들 중간에 위치하여, 디렉터리 서비스를 제공하고, 클러스터 내부에서 클라이언트간에 자료객체 공유를 가능하게 한다. (그림 1)은 서버, 3개의 ICM들, 그리고 다수의 클라이언트를 포함한 3T-CSD의 전형적인 예제이다.

ICM 계층의 도입은 서버 계층의 데이터를 위한 로크 관리에 변형이 필요하게 만든다. 고전적인 클라이언트-서버 모델에서는 모든 클라이언트들과 관련해 로크 테이블이 관리되는 반면 3T-CSD에서의 서버는 각 클러스터에 허가된 로크들을 관리하게 된다. 한편 ICM들은 각기 구성 클라이언트들이 보유하고 있는 자료객체에 대한 정보와 더불어 클라이언트 계층 로크에 대한 정보를 관리하게 된다. 만일, 하나의 트랜잭션 수행을 위해서 자료가 필요하게 되면, 해당 클라이언트는 먼저 국부적 사이트에 장착된 메모리/하드디스크 캐시에 요구된 자료객체 및 로크가 있는 지를 검색한다. 만일 해당 사이트에서 요구된 자료객체를 찾지 못한 경우에는 해당 ICM에 자료객체를 요구한다. 이때에 해당 ICM은 클러스터 내에 있는 다른 클라이언트나 서버에서 요구된 데이터를 공급 할 지를 결정하게 된다. 클라이언트는 콜백 요구를 받거나, 캐시에 빈 공간을 마련할 필요가 있을 때만 객체와 관련 로크를 반환한다. 이런 경우에 클라이언트와 ICM은 서버와 ICM에 있는 로크 테이블이 적절하게 상황을 반영하여 변경되었는지를 확인할 필요가 있다.

3T-CSD를 통해 구현할 수 있는 효율적 시스템의 운영은 구성된 클러스터링에 크게 의존한다. 클러스터링의 질을 측정할 수 있는 중요 지표는 클러스터의 결합력으로, 이는 대체로 클러스터 내부 자료공간 중에서 몇몇 작은 부분들에 대해 대부분의 자료요청이 이루어지거나 클러스터 간에 자료객체나 시스템 운영에 필요한 메시지의 교환이 적은 것으로 증명되어 질 수 있다. 과도하게 변화하는 자료객체 이용 유형은 다른 클러스터에 속해 있는 클라이언트들이 사용하는 데이터들을 접근하게 되는 현상을 만들어 낸다. 만일 이러한 현상이 지속되거나 확대된다면, 3T-CSD로부터 유도해 낼 수 있는 성능적 장점은 아주 미미하게 되거나 그 의미를 잃어버린다. 이러한 상황을 방지하기 위해 오프라인 클러스터링을 주기적으로 또는 시스템 사용량이 적은 시간대에 실행시킬 수 있다. 하지만, 더욱 바람직한 방법은 객체 이용 유형의 변경을 자동적으로 인식하고 필요한 클러스터링 변경을 수행 하는 것이다. 이것이 이 논문에서 핵심으로 삼는 문제이며, 그 목적을 위해서 기본적인 3T-CSD 모델의 여러 구성요소의 기능성을 개선하고자 한다. 서비스의 지속성과 전체 시스템의 성능악화를 방지하기 위해 변화하는 자료이용 유형을 감시 할 뿐 아니라 변화에 적응하여 클라이언트들을 최적의 클러스터에 재배치하는 방법론을 제시하고자 한다.

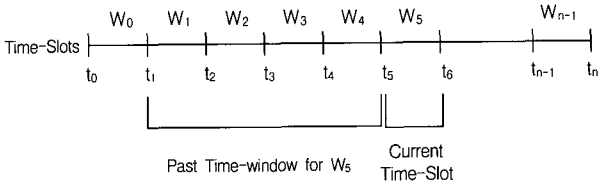
3. 온라인 클라이언트 클러스터링

3T-CSD 시스템 구성요소는 클라이언트들의 자료이용 유형의 변화를 감지할 수 있어야 하며, 시스템 운영에 문제를 야기시키지 않으면서 클러스터들의 구성 클라이언트를 변경할 수 있어야 한다. 이러한 기능은 다음의 3가지 기능적 구성요소로 구체화되고, 시스템 컴포넌트로 구현될 수 있다.

- 변화 인식 감시자(Change Detection Module) : 이 구성요소는 클라이언트들의 자료이용 유형의 변화 유무를 결정한다. 클라이언트의 자료객체 요구는 ICM에 의해 관찰되어지며, 수집된 정보는 빈도 리스트 형태로 관리된다. 변화 인식 감시자는 ICM에서 수행되며 클라이언트들의 자료객체 요구에 관련된 정보를 관리할 뿐만 아니라 ICM 클러스터 안에서의 자료객체 요구 빈도도 수집한다.
- 재분할 관리자(Reclustering Manager) : 이 구성요소는 변경된 자료이용 유형을 나타내는 클라이언트들을 위해 최적의 클러스터링을 결정한다. 만일 적절한 클러스터가 존재하고 있지 않을 때에는 새로운 클러스터를 필요한 ICM과 더불어 생성시킨다. 재분할 관리자는 서버에서 수행되며, 클러스터 단위의 자료객체 이용과 클러스터 안에서 만족된 요구 유형 정보를 관리한다.
- 재구성(Reconfiguration) : 이 기능은 순차적으로 클라이언트들을 새로운 클러스터로 이동시키는 것이다. 클라이언트가 이동하는 동안, 이 기능은 관련된 ICM들과 서버에서 필요한 로크 테이블의 변경 사항들을 통합 관리한다. 이 기능은 모든 3 계층에 분산되어 있는 기능들의 통합에 의해 완성된다.

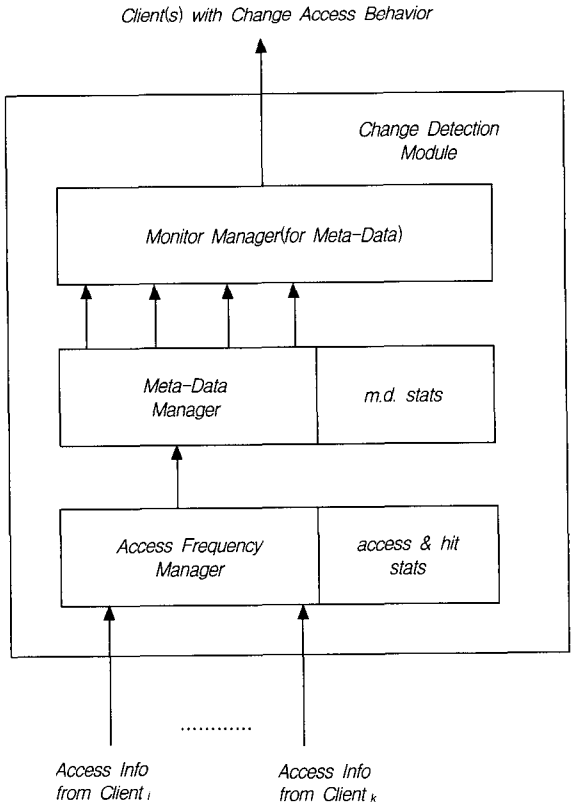
클라이언트들의 자료이용 유형을 분석하기 위해서는, ICM의 기능에 부가되는 변화 인식 감시자들은 클러스터 내부에서의 자료객체 이용 유형을 계수 및 빈도 측정으로 표현해야 한다. 아울러 자료 이용 유형의 변화을 감지하기 위해, 해당 클라이언트들에 의해 형성되는 자료이용 통계를 메타데이터로 변환시킨다. 이러한 정보를 수집하고 관리하는 빈도와 수집된 통계의 양은 매우 중요한 의미를 갖는데, 그 이유는 경우에 따라서는 자료이용 유형 변화에 대한 결정을 내리기까지 장시간이 소요될 수도 있기 때문이다. 이 문제를 방지하기 위해, 변화하는 자료이용 유형의 정확한 관리와 3T-CSD의 작업들을 동기화 할 수 있는 효능적인 시간의 표현이 필요하다. 따라서, 3T-CSD의 수행되는 시간 축은 같은 크기를 가지는 시간간격으로 나누어진다.

온라인 클러스터링에서는 2가지의 시간 창을 사용한다. 첫째는 *현재 시간* 창(Current Time-slot)이고, 둘째는 *과거 시간* 창(Past Time-window)이다. (그림 2)는 이 방법을 보여주고 있다. 현재 시간 창에서는 데이터 이용과 관련하여 기본적인 통계가 수집되며, 해당 메타데이터들이 생성된다. 과거 시간 창은 이미 지나간 시간을 뜻하며, 현재 시간 창을 선행하는 시간 창들의 연결로 구성된다. 과거 시간창의 길이는 S×M으로 표현되는데, S는 사용되는 시간차의 크기이고 M은 시스템 환경에 설정되는 값이다. 과거 시간창의 주된 목적은 변화의 존재를 결정하기 위한 통계를 한정하기 위한 것이다. 이 방법에 기반 하여, 변화 인식 감시자는 시간 창의 길이 동안에 발생된 클라이언트들의 자료객체 이용에 관련된 정보를 수집한다. 하나의 현재 시간 창이 종료될 때, 수집된 정보는 메타데이터들로 변환되고, 과거 시간 창에 정의된 기간동안 유지된다.



(그림 2) 3T-CSD에서의 시간창 구분

4. 자료객체 이용 유형의 변화 인식



(그림 3) 변화 인식 감시자

(그림 3) 변화 인식 감시자는 자료객체 이용에 있어서 변화를 보이는 클라이언트들의 재배치를 활성화 시킨다. 이를 위해서는 자료수집, 메타데이터 관리, 그리고 메타데이터를 이용한 데이터 이용 유형의 변화 인식 등이 지원되어야 한다.

접근 빈도 관리자(Access frequency manager)는 클러스터를 구성하는 클라이언트들의 자료이용과 클러스터내에서의 만족 유형에 관련된 정보들을 수집한다. 수집된 데이터를 이용하여, 메터데이터 관리자(meta-data manager)는 각 클라이언트와 각 현재 시간 창에 대해 4 가지의 메타데이터를 생성한다. 중요성이 없는 메타데이터들이 제거된 후, 감시 관리자(monitor manager)는 과거 시간 창과 현재 시간 창의 메타데이터에 의거하여, 각 클라이언트의 자료 이용 변화 유무를 결정한다.

메타데이터를 생성하기 위해서는 접근 빈도 관리자가 수집하는 정보를 사용한다. 구체적으로 한 시간 창 안에서의 클라이언트의 행동은 자료이용과 클러스터 내부에서 만족된 자료요구로 정의된다. 접근 및 만족도 통계(Access and hit stats) 자료 구조들은 다음의 빈도 정보를 유지한다.

① 클라이언트 자료 이용 빈도 리스트(Client Site Access : CSA) : 이 리스트는 한 클라이언트를 위해 한 개의 시간 창 동안의 서버로부터 또는 클러스터 내부에서 서비스된 요구에 대해서 객체별 빈도를 관리한다.

② 클라이언트 자료 만족 빈도 리스트(Client Site Hit : CSH) : 이 리스트는 한 클라이언트의 자료객체 요구들 중에서 같은 클러스터에 속하는 다른 클라이언트로부터 서비스된 요구에 대해 객체 별 빈도를 관리한다.

③ 클라이언트 과거 자료 이용 빈도 리스트(Last CSAs : LCSA) : 이 리스트는 한 클라이언트가 과거 시간 창 범위 내에서 요구한 자료이용 빈도 리스트들을 리스트화하여 관리한다.

위에서 설명한 클라이언트의 행동을 표현하는 통계자료들은 4 가지의 메타데이타의 생성을 위해 사용되며, 이 작업은 각 현재 시간창 단위로 행해진다. 각 메타데이타의 목적과 산출방법을 다음과 같다.

① 클라이언트 자료 이용 유사성(Client Access Pattern Consistency) : 만일 클라이언트가 최근에 자료이용 행동에서 급격한 변화를 보였다면, 과거와 최근 자료이용 빈도 리스트들 사이의 중복도가 낮을 것으로 기대된다. 이러한 변화를 수치화하기 위해서, 클라이언트 과거 자료 이용 빈도 리스트에서 존재하는 빈도들을 축적하여 하나의 새로운 빈도 리스트를 만든다. 이를 A(i, j)라고 하자. 또한 CSA(i, j)를 클라이언트 j가 i번째 시간 창에서 이용한 자료객체들에 대한 빈도 리스트라고 하자. 이 두개의 빈도 리스트는 비트맵으로 전화된다. 한 객체를 위한 비트는 만일 리스트 안에 있는 빈도의 평균값보다 크면 1로 설정한다. 이제 c(i, j)와 a(i, j)를 각기 CSA(i, j)와 A(i, j)에 상응하는 비트맵이라고 하면, 클라이언트 자료 이용 유사성은 다음과 같이 표현될 수 있다.

$$S^{client}(i, j) = |c(i,j) \text{ AND } a(i,j)| \; / \; |c(i,j)| \qquad (1)$$

여기서 || 표시는 비트맵 안에서 1의 갯수이며 AND는 비트별 AND이다.

② 클라이언트 자체 보유도(Count of Object Not Present at a Client's Site) : 만일 클라이언트가 참조 집약성(reference locality)을 가지고 있다면, 재사용되는 객체는 그 클라이언트의 단기/장기 메모리 공간에 저장되어 있을 가능성이 높다. 그러므로, 해당 ICM에 요구되어지는 객체의 빈도는 낮을 것이다. 하지만, 그 클라이언트의 자료 이용 유형에 변화가 생기게 되면, 새로운 유형에 의거한 자료 요구는 자체 메모리 공간에서 만족될 수 없기 때문에, 자료객체 요구는 급격히 늘어날 것이다. 이것은 CSA(i, j)에 기록된 빈도들의 합계로 측정될 수 있다.

③ 클러스터링 근접성(Clustering Proximity) : 이 메타데이타는 현 시점에서 클라이언트가 속해 있는 클러스터가 얼마나 적절한 지를 수치화한다. 만일 현재의 클러스터가 최적이라면, 클러스터 구성멤버를 전부 고려할 때 매우 높은 유사성을 나타낼 것으로 기대된다. 시간창 j에서 클라이언트 i의 클러스터링 근접성은 다음과 같이 계산되어 질 수 있다.

$$S^{cluster}(i, j) = |c(i, j) \text{ AND } (b(i, j) - a(i, j)| \; / \; |c(i, j)| \qquad (2)$$

여기서 b(i, j)는 해당 클러스터에서 이용된 모든 객체 접근 유형을 나타낸다.

④ 클러스터링 만족도(Intra-cluster Satisfaction Ratio) : 이 메타데이타는 요구된 자료객체 중에서 클러스터 범위 안에서 만족된 백분율을 나타낸다.

각 시간 창에 대하여, 메타데이터 관리자는 접근 빈도 관리자가 관리하는 자료들을 이용해 위에서 설명한 메타데이타들을 생성한다. 또한 감시 관리자는 위에서 기술한 4 가지의 메타데이타 스트림들을 이용하여 객체 이용 유형 변화를 판단한다. 여기서 주요 문제는 여분의 중요하지 않은 데이터를 평균값에 기반 한 방법으로 제거하고 중요 변화만을 고려하여 최종 결정을 내린다는 점이다. 3T-CSD에서는 최대 변화율(maximum divergency ratio)에 근거한

방법을 각 4가지의 메타데이타 스트림에 적용하여, 4가지 모두에서 변화가 있음 판정을 할 수 있을 때만 최종적으로 클라이언트의 객체 이용 유형에 변화가 있다고 선언한다. 만일 변화가 있다고 최종 결정이 내려지면, 감시관리자는 서버에 있는 재분할 관리자에게 이 사실을 통보함으로써 해당 클라이언트를 위해 새로운 클러스터를 결정하는 과정을 시동한다.

언라인 클라이언트 재배정 과정은 서버에서 먼저 최적 클러스터를 결정하고, 다음에 해당 클라이언트를 새로운 클러스터로 옮기기 위한 일련의 작업을 하는 것이다. 첫째 과정은 재분할 관리자에 의해 수행되며, 둘째 과정은 재구성 관리자에 의해 수행된다. 재분할 관리자는 현재 시간 창에서 수집된 해당 클라이언트의 자료 이용 유형을 클러스터들의 최근/과거 자료 이용 유형과 비교하여 메타데이타를 계산한다. 유사한 방법으로 클라이언트의 자료 이용 유형은 현재/과거 시간 창에서 수집된 클러스터 내부 만족 유형과 비교된다. 따라서 이 과정을 통하여 각 클러스터당 4가지 메타데이타가 생성된다. 마지막 결정은 각기 메타데이타에 특정 비중치를 주어 합계를 내어 하나의 값으로 치환을 한다. 여기서 최고치를 갖는 클러스터가 최적 클러스터로 결정된다. 이 방법은 단시간 내에 새로운 최적 클러스터를 결정해야 하는 필요성에 의한 것이다. 만일 여러 클라이언트를 전부 고려하는 최적화 알고리즘을 적용시킨다면, 시스템의 작업에 막대한 영향을 줄 수 있다. 이는 클라이언트를 옮기는 과정에서 로크 테이블등의 중요 자료구조가 시스템 재구성이 완료될 때까지 접근할 수 없기 때문이다.

5. 시스템 재구성

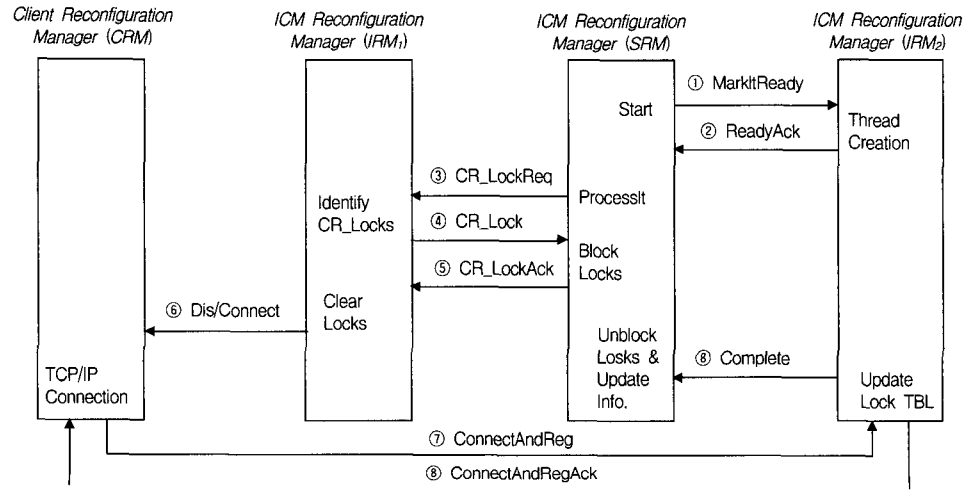
재구성 관리자는 클라이언트를 하나의 클러스터에서 다른 클러스터로 옮길 때 발생하는 문제들을 해결하기 위해 서버, ICM, 그리고 클라이언트들에 분산되어 있는 구성요소들을 이용한다. 클라이언트 이주시, 가장 중요한 문제는 ICM들과 서버에 있는 로크 테이블에서 생기는 변경을 순차적으로 처리하는 것이며, 이 때문에 생기는 트랜잭션 수행의 지연을 최소화 시키는 일이다. (그림 4)는 이를 위해 제안된 프로토콜을 보여준다. 서버 재구성 관리자(SRM)는 프로토콜의 전체 수행과정을 관장한다. 또한 SRM은 클라이언트에 있는 국부 재구성 관리자(LRM)들과 그리고 클러스터 재구성 관리자(CRM)들과 상호협조를 한다.

프로토콜을 설명하기 위해서, 여기서는 ICM_1에 의해 지원되는 한 클라이언트가 ICM_2로 옮겨지는 상황을 가정한다. (그림 4)에서 LRM은 클라이언트에서 수행되며, CRM_1과 CRM_2는 ICM_1과 ICM_2에서 각기 수행된다.

클라이언트의 이동이 결정되면, 온라인 재분할 매니저는 SRM에 메시지를 보낸다. 이 때, SRM은 "MakeItReady" 메시지에 클라이언트의 식별자와 함께 CRM_2로 보내게 된다. 이 메시지를 받으면 CRM_2는 새로운 구성 클라이언트를 받아 들일 준비를 한고, SRM에 "ReadyAck" 메시지를 보냄으로써 준비완료를 알린다.

SRM은 "ReadyAck" 메시지를 받고, "CR_LockReq" 메시지를 CRM_1에 보내 클라이언트 이동에 따른 서버의 로크 테이블에 변경 요소가 있는 로크 정보를 요구한다. CRM_1은 "CR_LockReq" 메시지를 받고, 먼저 클러스터 로크 테이블에 생기는 변화때문에 영향을 받는 객체 요구의 수행을 보류시키고, 로크 테이블을 변경한다. 이 과정이 끝나면 CRM_1은 서버 로크 테이블에 영향을 미치는 로크 정보를 서버에게 보낸다.

이 때 서버는 바로 전에 받은 로크들에 의해 영향을 받는 객체 요구들을 클라이언트 이동이 끝날 때까지 보류시



(그림 4) 클러스터링 재구성 프로토콜

킨 다음 CRM_1에게 "CR_LockAck" 메시지를 보낸다. 이 메시지를 받은 CRM_1은 "Dis/Connect" 메시지를 클라이언트에게 보냄으로써 클라이언트 이동이 결정되었음을 알린다. 또 한편으로는 CRM_1은 보류되었던 객체 요구들의 수행을 다시 시작한다.

클라이언트가 "Dis/Connect" 메시지를 받으면, 먼저 응용 프로그램들의 객체 요구를 보류하고, ICM_2와 커뮤니케이션을 시작한다. 이 때 클라이언트는 CRM_2에게 현재 보유하고 있는 로크들에 관한 정보를 "ConnectAndReg" 메시지에 포함시켜 보낸다. 이 정보를 받은 CRM_2는 로크 테이블에 새로운 로크들을 등록하는 한편 "ConnectAndRegAck" 메시지를 클라이언트에게 보낸다. 동시에 CRM_2는 "Complete" 메시지를 SRM에게 보냄으로써 서버에서 보류된 객체 요구들의 수행을 재개 시킨다. 클라이언트는 "ConnectAndRegAck" 메시지를 받으면 응용 프로그램으로부터의 요구 수행을 재개한다.

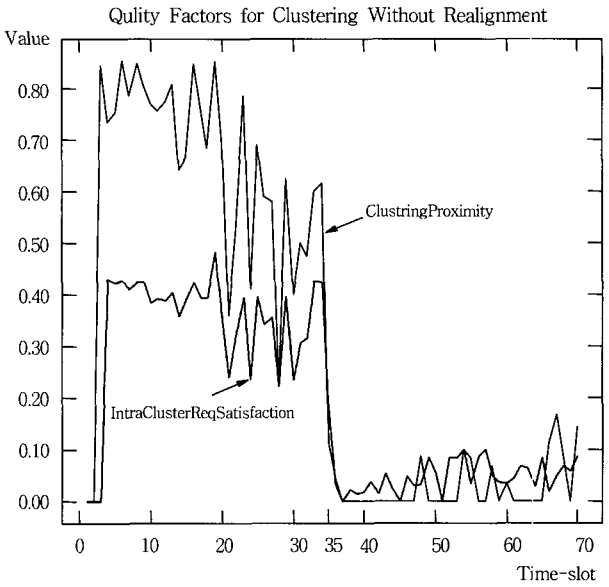
6. 구현 사례

온라인 3T-CSD의 성능 검증은 네트워크로 연결된 Sun Ultra 워크스테이션들에서 시스템을 제작하여 실행되었다. 실험의 목적은 온라인 클러스터링을 3T-CSD에 적용하였을 때의 시스템의 성능 개선과 클라이언트 이동시 생기는 시스템 자원의 소비를 알아보고자 함이다. 실험을 위하여, 서버는 한 개의 워크스테이션에 장착하였으며, ICM들과 클라이언트들은 여러 대의 워크스테이션에 분산시켰다. <표 1>은 주요 데이터베이스 환경 설정 값을 보여준다.

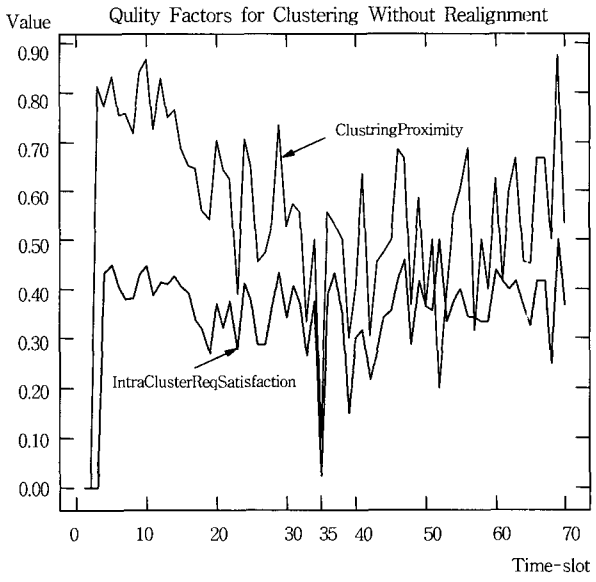
〈표 1〉 데이터베이스 환경 설정 값

환경 변수	설정 값
데이터베이스 크기	10,000객체
서버 메인 메모리 크기	2,500객체
클라이언트 디스크 캐시 크기	200객체
클라이언트 메모리 캐시 크기	100객체
트랜잭션에 필요로 하는 객체 수 (최소, 평균, 최대)	(1, 5, 10)

각 시스템 구성간의 커뮤니케이션은 TCP/IP 소켓을 사용하였으며, ICM은 다중스레드 기술을 이용하여 각 클라이언트를 하나의 스레드가 담당하게 하였다. 각 클라이언트에서는 3초의 간격을 갖는 프아송 트래픽을 이용하여 트랜잭션을 발생시켰다. 특히, 각 트랜잭션의 CPU 프로세싱 시간은 평균 0.5초를 가지는 지수 분포에 적용시켜 실험하였다. 데이터베이스 공간에서 핫 영역(Hot area)은 전체의 반으로 설정하였으며, 50개의 작은 공간(Hot spot)으로 분리하였다. 각 클라이언트는 5개까지의 핫스폿을 접근할 수 있으며, 90%의 자료객체 요구는 선택된 핫스폿들을 접근하도록 설정하였다. 이러한 설정은 클라이언트들의 객체 접근 유형을 구현하기 위한 것이다.



(그림 5) 온라인 클러스터링 재구성 기능이 없을 경우



(그림 6) 온라인 클러스터링 재구성 기능이 존재할 경우

첫 번째 실험에서 60개의 클라이언트들을 시스템에 설치하고, 하나의 클라이언트로 하여금 자료이용 유형을 변화하도록 설정하고 중점적으로 관찰했다. 이 클라이언트는 35번째 시간 창에서 80%의 자료이용 유형을 다른 핫스폿 공간으로 바꾼다. 실험 중에 클러스터링 근접성과 만족도에 대한 메타데이터를 수집하였다. (그림 5)에서 볼 수 있듯이 재구성 기능이 없을 경우, 관찰된 클라이언트의 클러스터링

근접성과 만족도가 현격히 감소하는 것을 쉽게 볼 수 있다.

두 번째 실험으로 그려진 (그림 6)은 자료이용 유형이 변함에도 불구하고 온라인 클라이언트 재배정 기능이 클러스터링 만족도를 계속 유지 시키는 것을 확인할 수 있다. 35번째 시간 창에서 감소된 메타데이터 수치들이 원래 평균치로 빠르게 회복되는 것을 볼 수 있다. 이는 자료이용 유형의 변화를 시스템이 감지, 새로운 적정 클러스터에 클라이언트를 배정했기 때문이다. (그림 5)와 (그림 6)은 제안된 변화 인지 방법의 효율성을 입증하는 것이다. 종합적으로 고려를 할 때, 온라인 클라이언트 재배정을 3T-CSD에 도입함으로써 자료이용 유형이 변할 때에도, 적절하게 구성된 3T-CSD의 성능 강점을 유지하면서 원활하게 시스템이 운영되는 것을 알 수 있다.

〈표 2〉 CPU 소요량

기능적 컴포넌트	소요 시간
클라이언트 당 자료이용 변화 인식	0.0009초
클러스터 당 리클러스터링	0.003초
한 클라이언트의 이동	0.020초

<표 2>는 온라인 3T-CSD를 수행하면서 온라인 클라이언트 재배정에 소요되는 여러 가지 다른 모듈에서의 CPU 자원의 소비를 보여준다. 첫 번째 값은 ICM에서 수집한 것이고 다른 것들은 서버 모듈들에서 기록한 것이다. 재분할에 필요한 자원 소비는 설치된 클러스터 수에 비례한다. 60개의 클라이언트가 설치되었을 때의 평균 온라인 자료 이용 유형 분석은 각 시간창당 평균 0.054초가 소요된다. 6개의 클러스터가 형성되었을 때, 하나의 클라이언트를 재 배정하기 위해서는 0.038초가 소요된다. 위의 값들과 <표 2>에 나타난 수치들을 고려해 볼 때, 온라인 클라이언트 재배정 기능이 필요로 하는 비용은 최소화 되었다고 본다.

7. 결 론

고전적인 2 계층 클라이언트-서버 데이터베이스(CSD들)는 범위성 문제에 있어서 제한된 성능을 보여왔다. 이 문제를 풀기 위하여, 3 계층 데이터베이스(3T-CSD)가 제안되었다. 이 논문에서는 온라인 모드에서 클라이언트 클러스터링을 지원하는 기능들을 검토하였다. 이 기능들의 목적은 급격하게 변하는 자료이용 유형 존재시에도 3T-CSD의 성능을 유지하는 것이다. 이를 위해 변화 검증, 재분할, 그리고 시스템 구성 변화등에 필요한 모듈들이 제안 되었다. 이 논문에서는 시제품(prototype) 시스템을 이용하여 그 기술적 완성도와 유용성을 검증하였다. 주요 결과로서는 온라인 클라이언트 클러스터링은 클라이언트들의 급격한 자료이용 유형에 유연하게 시스템을 재구성하여 3T-CSD의 성능적 장점을 유지하였고, 그에 필요한 비용이 적절한 수준 이하 였다.

참 고 문 헌

[1] J. Andrade, M. Carges and M. MacBlane, "The Tuxedo System : AN Open On-line Transaction Processing Environment," Data Engineering Bulletin, 17(1), 1994.

[2] M. Blaze and R. Alonso, "Dynamic Hierarchical Caching in Large-scale Distributed File Systems," In Proceedings of the 12th International Conference On Distritubed Computing Systems, Yokohama, Japan, June, 1992.

[3] M. Dahlin, C. Mather, R. Wang, T. Anderson and D. Patterson, "A Quantitative Analysis of Cache Policies for Scalable Network File Systems," In Proceedings of the Sigmetrics Conference on Measurement and Modeling of Computer Systems, Nashiville, Tennessee, pp.150-160, May, 1994.

[4] A. Delis and N. Roussopoulos, "Performance Comparison of Three Modern DBMS Architectures," IEEE Transactions on Software Engineering, 19(2), pp.120-138, February, 1993.

[5] D. Fang and S. Ghandeharizadeh, "An Experimental System for Object-Based Sharing in Federated Databases," VLDB Journlal, 5(2), pp.151-165, 1996.

[6] M. Franklin, M. Carey and M. Livny, "Transanctional Client Server Cache Consistency : Alternatives and Performance," ACM Transactions on Database Systems, 22(3), pp.315-363, 1997.

[7] L. Liu, C. Pu and W. Tang, "WebCQ : Detecting and Delivering Information Changes on the Web," In Proceedings of International Conference on Information and Knowledge Management, Washington, DC, 2000.

[8] E. Panagos, A. Biliris, H. Jagadish and R. Rastogi, "Client Based Logging for High Performance Distributed Architectures," In Proceedings of the 12th International Conference on Data Engineering, New Orleans, LA, USA, pp.344-351, Feb-March, 1996.

[9] J. Park, V. Kanitkar and A. Delis. Distributed and Parallel Databases, An International Journal (DAPD), 10(2), pp.161-198, September, 2001.

[10] J. Tohompson, "Web-Based Enterprise Management Architecture," IEEE Communications Magazine, pp.80-86, March, 1998.

박 용 범

e-mail : ybpark@dankook.ac.kr

1985년 서강대학교 전자계산학 학사

1987년 N.Y. Polytechnic Univ. Computer Sci. M.S.

1991년 N.Y. Polytechnic Univ. Computer Sci. Ph.D.

1992년 현대전자 산업개발 산전연구소 선임연구원

1993년~현재 단국대학교 컴퓨터과학 부교수

관심분야 : 시스템 구성 방법론, 소프트웨어 개발 방법론, 컴퓨터 게임 개발

박 제 호

e-mail : dk_jhpark@dankook.ac.kr

1985년 서강대학교 전자계산학 학사

1993년 N.Y. Polytechnic Univ. Computer Sci. M.S.

2001년 N.Y. Polytechnic Univ. Computer Sci. Ph.D.

2001년 System Architect at Voicemate (NYC, USA)

2003년~현재 단국대학교 컴퓨터과학 전임강사

관심분야 : DB구조, 분산 DB, Information Retrieval