산학기술학회논문지
Vol. 4, No. 2, pp. 108-113, 2003

온라인 모드 클라이언트-클러스터 운영 시스템

박제호* · 박용범*

Management System of On-line Mode Client-cluster

J. Park* and Y. Park*

요 약 고전적인 클라이언트-서버 데이터베이스 시스템은 동시 클라이언트가 많을 경우 범위성에서 한계를 가지는 것은 많은 연구 결과를 통해 알려져 있다. 사용자들의 자료이용의 유사성 기반 다계층 데이터베이스 시스템은 유사한 자료 이용 행태를 나타내는 클라이언트들을 논리적 클러스터들로 분할한다. 그 결과로 클러스터 내부에서의 자료객체 요구 만족도를 최적화하여 서버에 대한 부하는 줄어들 뿐 아니라, 객체 요구에 대한 응답시간은 최소화 된다. 이 시스템의 목적을 위해서 유사한 자료이용 행태에 기반한 클러스터링의 관리가 매우 중요한 구성요소이다. 오프라인 방식은 전체 클러스터링의 질을 최적화하지만, 그 비용과 수행 시기 선택에 따른 안정적인 시스템 성능 관리 측면을 신중하게 고려하여야 한다. 이 논문에서는 자료이용 유형에 생기는 변화를 실시간 인지하여 시스템 구성을 변경하는 방법론을 제안한다. 마지막으로 온라인 변화 인식의 유효성을 예시하고, 온라인 시스템 재구성의 구현 가능성과 기술적 완성도를 검증한다.

Abstract Research results have demonstrated that conventional client-server databases have scalability problem in the presence of many concurrent clients. The multi-tier architecture that exploits similarities in clients' object access behavior partitions clients into logical clusters according to their object request pattern. As a result, object requests that are served inside the clusters, server load and request response time can be optimized. Management of clustering by utilizing clients' access pattern-based is an important component for the system's goal. Off-line methods optimizes the quality of the global clustering, the necessary cost and clustering schedule needs to be considered and planned carefully in respect of stable system's performance. In this paper, we propose methods that detect changes in access behavior and optimize system configuration in real time. Finally this paper demonstrates the effectiveness of on-line change detection and results of experimental investigation concerning reconfiguration.

Key Words : Real-time Recognition of Change in Data Access Behavior, Three-tier Database, On-line Client Clustering

1. 서 론

네트워크로 구성된 환경에서 적정 효율을 제공하는 대용량 자료운영에 대한 중요성이 부각되는 환경에서 클라이언트-서버 데이터베이스는 우수한 성능을 제공한다[5, 7, 10]. 클라이언트-서버 데이터베이스는 비교적 적은 물리적 자원으로 높은 성능개선을 구현하지만, 개선된 시스템 효율에도 불구하고 범위성(scalability) 문제를 아직 내포하고 있다[4]. 디렉터리 계층을 이용한 3계층 클라이언트-서버 구조(3T-CSD)는 이러한 문제를 해결한다[9]. 이 시스템에서는 유사한 자료객체 이용 유형을 보이는 클라이언트들을 하나의 논리적 클러스터로 구성한다. 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. 3T-CSD의 성능은 클라이언트 클러스터링에 의해 많은 영향을 받는다. 이러한 클라이언트 클러스터링의 관리는 주기적으로 축적된 자료이용 정보를 이용하는 오프라인 방식을 채택할 수 있다[1, 2, 3].

본 논문에서는 온라인 클라이언트 클러스터링의 개념을 재검토하고, 온라인 클러스터링을 위한 구조를 제시하고, 온라인 방식을 채택한 자료이용 행태 변화 인식 기능을 위한 방법론과 기술적인 문제를 검증하였다.

2. 다 계층 데이터베이스 구조

3T-CSD은 객체전송 기반이므로 서버는 객체를 공급

*단국대학교 컴퓨터과학과
Tel: 041-550-3483

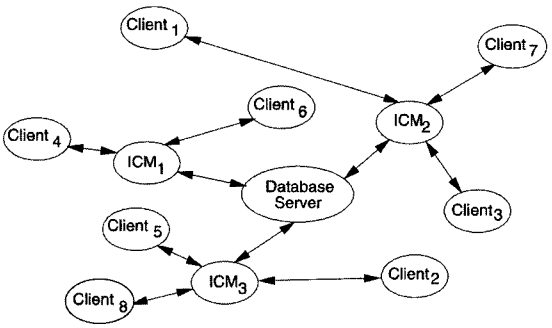

그림 1. 3 계층 데이터베이스 구조

하며 클라이언트는 공급된 객체들을 이용하여 트랜잭션을 수행한다[6]. 클라이언트는 메모리와 하드디스크 캐시 버퍼를 이용하여 공급된 객체들을 임시로 저장하고 관리한다. 따라서 서버는 클라이언트들을 지원하기 위한 기본 기능들만을 수행하며, 로크 테이블과 관련 자료구조를 유지하여 직렬가능 접근를 지원한다. 클라이언트들은 트랜잭션간 캐싱을 지원하므로, 서로 다른 트랜잭션들은 캐시에 저장되어 있는 객체들을 공유할 수 있다.

3T-CSD의 기본 개념은 유사한 자료이용 유형을 갖는 클라이언트들을 논리적 클러스터로 구성한다는 것이다. 여러 가지 정적 클러스터링은 이미 검증되었다[9]. 클러스터링 구성이 확정되면, 중간 캐시 관리자(ICM)가 각 클러스터에 할당된다. 이 ICM들은 구조상 서버와 클라이언트들 중간에 위치하여, 디렉터리 서비스를 제공하고, 클러스터 내부에서 클라이언트 간에 자료객체 공유를 가능하게 한다. 그림 1은 서버, 3개의 ICM들, 그리고 다수의 클라이언트를 포함한 3T-CSD의 전형적인 예제이다.

ICM 계층의 도입은 서버 계층의 데이터를 위한 로크 관리의 변형을 필요로 한다. 서버는 각 클러스터에 허가된 로크들을 관리하며, ICM들은 각기 구성 클라이언트들이 보유하고 있는 자료객체에 대한 정보와 더불어 클라이언트 계층 로크에 대한 정보를 관리하게 된다. 만일 응용프로그램이 자료객체를 요구할 때 해당 사이트에서 요구된 객체를 찾지 못한 경우에는 해당 ICM에 자료객체를 요구한다. 이때에 해당 ICM은 클러스터 내에 있는 다른 클라이언트나 서버에서 요구된 데이터를 공급 할 지를 결정하게 된다. 만일 같은 클러스터에 속하는 클라이언트가 요구된 객체를 보유하고 있을 때는, 서버와의 상호작용 없이 요구된 객체를 원격 클라이언트로부터 전송하게 된다.

3T-CSD를 통해 구현할 수 있는 효율성은 구성된 클러스터링에 크게 의존한다. 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 오브젝트 이용 유형의 변화를 실시간에 인식하고 필요한 클러스터링 변경을 수행 하는 것이다. 이 논문의 핵심은 다양하게 변화하는 클라이언트의 객체이용 행태 표현과 그 행태의 변화를 인지할 수 있는 방법론이다.

3. 필요 구성요소

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 될 수 있다.

(1) 변화 인식 감시자(Change Detection Module): 이 구성요소는 클라이언트들의 자료이용 유형의 변화 유무를 결정하기 위해서, ICM에서 수행되며 클라이언트들의 자료객체 요구에 관련된 정보를 관리할 뿐만 아니라 ICM 클러스터 안에서의 자료객체 요구 빈도도 수집한다.

(2) 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.

(3) 재구성(Reconfiguration): 이 기능은 모든 3계층에 분산되어 있는 기능들의 통합에 의해 수행되며, 순차적으로 클라이언트들을 새로운 클러스터로 이동시키는 것이다. 클라이언트가 이동하는 동안, 이 기능은 관련된 ICM들과 서버에서 필요한 로크 테이블의 변경 사항들을 통합 관리한다.

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 장시간

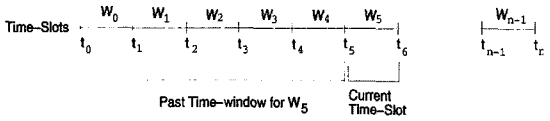

그림 2. 3T-CSD에서의 시간창 구분

이 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 시간창을 사용한다. 첫째는 현재 시간창(Current Time-slot)이고, 둘째는 과거 시간창(Past Time-window)이다. 그림 2는 이 방법을 보여주고 있다. 현재 시간창에서는 데이터 이용과 관련하여 기본적인 통계가 수집되며, 해당 메타데이터들이 생성된다. 과거 시간창은 이미 지나간 시간을 뜻하며, 현재 시간창을 선행하는 시간창들의 연결로 구성된다. 과거 시간창의 주된 목적은 변화의 존재를 결정하기 위한 통계를 한정하기 위한 것이다. 이 방법에 기반 하여, 변화 인식 감시자는 시간창의 길이 동안에 발생된 클라이언트들의 자료객체 이용에 관련된 정보를 수집한다. 하나의 현재 시간창이 종료될 때, 수집된 정보는 메타데이터들로 변환되고, 과거 시간창에 정의된 기간동안 유지된다.

4. 자료객체 이용 유형의 변화 인식

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.

메타데이터를 생성하기 위해서는 수집된 접근 빈도 정보를 사용한다. 구체적으로 한 시간창 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): 한 클라이언트가 ICM에 요구한 자료객체들에 대해서 객체별 빈도를 수집한다.

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

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

위에서 설명한 통계자료들은 4가지의 메타데이터의 생성을 위해 사용되며, 이 작업은 각 현재 시간창 단위로 행해진다. 각 메타데이터의 목적은 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 자료 이용 빈도 리스트에 수집된 자료 이용 유형을 클라이언트 과거 자료 이용 빈도 리스트에 나타난 유형과 비교해서 계산할 수 있다.

(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.

(3) 클러스터링 근접성(Clustering Proximity): 이 메타데이터는 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.

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

감시 관리자는 위에서 기술한 4가지의 메타데이터 스트림들을 이용하여 오브젝트 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 메타데이터 스트림에 적용하여, 4가지 모두에서 변화가 있음 판정을 할 수 있을 때만 최종적으로 클라이언트의 객체 이용 유형에 변화가 있다고 선언한다. 최대 변화율 기반 변화 검출은 다음과 같은 순서로 진행된다.

① 과거 시간창에서 첨두값(peak value)를 결정한다.

② 첨두 값들을 이용하여, 평균 첨두값(ave)과 최대 첨두값(max)을 구한다.

③ 최대 변화율(α)을 구한다.
$\alpha = (\text{max-ave}) / \text{ave}$
④ 현재 시간창의 값 v에 대하여, 변화율을 구한다.
$\beta = (\text{v-ave}) / \text{ave}$
⑤ 만일 ($\alpha/\beta > 1$)이면 변화가 있음으로 판정한다.

만일 변화가 있다고 최종 결정이 내려지면, 서버에 있는 재분할 관리자는 해당 클라이언트를 위해 새로운 클러스터를 결정하는 과정을 시동한다. 온라인 클라이언트 재배정 과정은 서버에서 먼저 최적 클러스터를 결정하고, 다음에 해당 클라이언트를 새로운 클러스터로 옮기기 위한 일련의 작업을 하는 것이다.

5. 저비용 객체이용 행태 변화 감지

최대 변화율을 이용하여 객체사용의 변화 유무를 판단하는 방법은 현재 시간창과 과거 시간창에 대한 메타데이터 스트림을 필요로 한다. 즉 과거 시간창에 대한 메타데이터 스트림의 저장을 의미하며, 경우에 따라서는 많은 기억공간과 계산을 요구한다. 이것은 저비용 행태 변화 감지 방식을 적용함으로써 개선될 수 있다. 먼저, 고려되는 시간 기간 동안 최대 첨두값과 유사 평균값만을 유지하고, 새로운 첨두값이 나타나면 최대 첨두값과 유사 평균값을 계산하여 이를 변화 검출에 이용한다. 새로이 발생한 첨두값을 x_p라 하면 최대 첨두값(x_{max})과 유사 평균값(x_{ave})은 다음과 같이 정의된다.

$$x_{max} = \max(x_{max}, x_p)$$
$$x_{ave} = (n \times x_{ave} + x_p)/(n+1)$$

위의 식에서 n은 가중치로서 1보다 큰 정수이다. 이 정의를 이용하여, 최대 변화율 α와 현재 시간창에 대한 변화율 β는 다음과 같이 계산된다.

$$\alpha = (x_{max} - x_{ave}) / x_{ave}$$
$$\beta = (v - x_{ave}) / x_{ave}$$

6. 구현 사례

온라인 3T-CSD의 성능 검증은 이더넷 LAN으로 연결된 Sun Ultra 워크스테이션들에 시제품을 사용하여 실행되었다. 실험의 목적은 온라인 클러스터링을 3T-CSD에 적용하였을 때, 변화하는 자료객체 행태의 인식 여부를 객체이용 변화율에 따라 점검하는 것이다. 실험 환경은 한 개의 워크스테이션에 서버를 장착하고, ICM들과 클라이언트들은 여러 대의 워크스테이션에 분산시켰다. 표 1은 주요 데이터베이스 환경 설정 값을 보여준다.

각 클라이언트에서는 3초의 간격을 갖는 프아송 트래픽을 이용하여 트랜잭션을 발생시키고, 각 트랜잭션의 CPU 프로세싱 시간은 평균 0.5초를 가지는 지수 분포에 적용시켰다. 데이터베이스 공간에서 핫 영역(Hot area)은 전체의 반으로 설정하였으며, 핫 영역은 50개의 하부 공간(Hot spot)으로 분리하였다. 객체 접근 유형을 조절하기 위하여, 각 클라이언트는 5개까지의 핫스폿을 접근하고, 90%의 오브젝트 요구는 선택된 핫스폿들을 접근하도록 설정하였다. 실험에서는 비용이 많이 드는 메타데이터 스트림을 이용한 변화 감지 방법론을 적용하여 최대 비용을 수집하였다.

첫번째 실험에서 60개의 클라이언트들을 설치하고, 하나의 클라이언트로 하여금 자료이용 유형을 변화하도록 설정하고 그 클라이언트의 자료이용 유형을 중점적으로 관찰했다. 이 클라이언트는 35번째 시간창에서

표 1. 데이터베이스 환경 설정 값

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

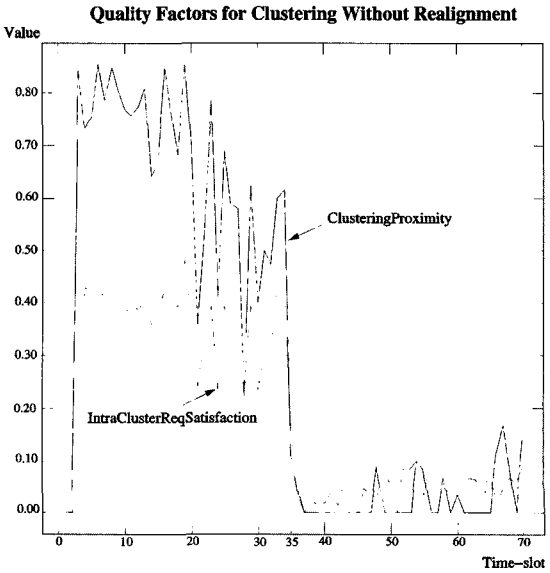

그림 3. 메타데이터 변화

80%의 자료이용 유형을 다른 핫스폿 공간으로 바꾼다. 실험 중에 클러스터링 근접성과 만족도에 대한 메타데이터를 수집하였다. 그림 3에서 볼 수 있듯이 자료이용 유형에 변화가 생겼을 때, 관찰된 클라이언트의 클러스터링 근접성과 만족도가 현격히 감소하는 것을 쉽게 볼 수 있다.

두번째 실험에서는 자료이용 행태 변화율을 변화 시켜, 각 α/β 비율을 수집하였다. 표 2에서 M1은 클러스터링 근접성이고, M2는 클라이언트 자료 이용 유사성이고, M3는 클라이언트 자체 보유도이고, M4는 클러스터링 만족도를 나타낸다. 표에서 관찰할 수 있는 것처럼 자료이용 행태가 100%에서 60%까지 변화를 할 때에는 행태 변화를 인지할 수 있었다. 하지만 40%에서 20%의 변화율에서는 클라이언트 자체 보유도 메타데이터에서의 변화 감지에 실패를 함으로써 전체 변화 감지에 실패를 하였다. 이 결과는 여러 가지 메타데이터를 사용할 경우 과거 시간창 크기를 각 메타데이터에 따라 조절해야 함을 알 수 있다.

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

표 2. 자료이용 행태 변화 감지

행태 변화율	100%	80%	60%	40%	20%
M1	2.88	2.43	2.71	1.69	1.28
M2	6.92	3.22	4.01	1.91	1.48
M3	1.91	1.33	1.02	<1.0	<1.0
M4	6.92	5.94	5.21	5.80	2.51

표 3. CPU 소요량

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

7. 결 론

고전적인 2계층 클라이언트-서버 데이터베이스(CSD들)는 범위성 문제에 있어서 제한된 성능을 보여왔다. 이 문제를 풀기 위하여, 3계층 데이터베이스(3T-CSD)가 제안되었다. 이 논문에서는 온라인 모드에서 클라이언트 클러스터링을 지원하기 위한 자료유형 변화 감지를 검토하였다. 이 논문에서는 시제품을 이용하여 기술적 완성도와 유용성을 검증하였다. 주요 결과로서는 제안된 메타데이터가 충분히 자료이용 행태 변화 감지에 대한 효능성을 가지고 있으며, CPU의 비용이 적절히 수준 이하 였다. 또한 제안된 모든 구성요소의 경비가 시스템을 자료이용 행태 변화에 따라 재구성하는목적을 이루는데 적절하였다.

감사의 글

이 연구는 2003학년도 단국대학교 대학연구비의 지원으로 연구되었음

참고문헌

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

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

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

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

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

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

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

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

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

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