반도체및디스플레이장비학회지 제3권 제4호(2004년 12월)
Journal of the Semiconductor & Display Equipment Technology, Vol. 3, No. 4. December 2004.

무선 근거리 통신망 환경을 위한 다단계 데이터베이스 시스템

박 제 호†

†단국대학교 컴퓨터과학과

Multi-tier Database System for Wireless LAN Environment

Je Ho Park†

†Dankook University Computer Science

ABSTRACT

As the usage of wireless LAN becomes common in working environment, the number of database systems that support both wired and wireless users increases rapidly. The characteristics of wireless LAN that its speed is slow relatively comparing to wired network and the users in its environment connects to different communication points as they moves creates another challenge to be resolved in database systems. In the environment of hybrid communication systems, wired and wireless for voluminous data amount and a number of users, the two layer architecture of the conventional client-server database systems has limitation in the system performance. This is due to that server is the only point of data service in client-server database systems. In this paper, we discuss a new extended database system architecture that data services are distributed among servers and clients based on user database access patterns in order to improve system performance. We analyze the expected system performance by using simulation technique and prove the practical utilization of the system by demonstrating experimental results.

Key Words : Database, Client-server, Wireless, Local Area Network, Design, Clustering

1. 서 론

무선 근거리 통신망은 시스템에 구성을 할 때 허브에서 가입자 단자까지 유선 대신 전파와 같은 무선을 이용하는 기술이다. 액세스포인트 장비에서 단말장비까지 전자기파를 이용하기 때문에 유선 네트워크에 비해 상대적으로 신속한 시스템 구축이 가능하며, 유선으로 설치가 어려운 환경까지도 네트워크를 확장시킬 수 있는 이동성, 유연성, 휴대성, 간편성 등의 이점으로 응용분야가 확산되고 있다. 이러한 무선 근거리 통신망의 사용은 기존에 설치되어 있는 유선 기반 시스템과 연동되면서 또 다른 혼합형 시스템 환경을 만들어 내고 있다. 일반적으로 대다수 산업적인 생산 시스템에 사용되는 데이터베이스는 그 양이 방대할 뿐만 아니라 동시에 데이터를 사용하는 사용자가 많다는 특징을 보이고 있다. 따라서 시스템을 구성하는 네트워크가 유무선 시스템으로 혼합적인 형태일 경우 데이터베이스 시스템은 다른 통신 접근 경로를 이용하는 사용자들을 모두 고려하여 효율적인 구조를 갖추어야 한다.

네트워크로 연결된 시스템 환경하에서 대용량의 데이터와 시스템을 사용하는 많은 수의 동시 사용자를 지원하는 문제는 많은 새로운 방법에 대한 연구를 촉진시켰다. 그 결과 데이터를 관리하는 서버와 고성능의 워크스테이션을 이용하는 클라이언트-서버 데이터베이스 시스템의 개발과 그 사용은 현재 대다수의 데이터베이스 응용에 적용되어 원활한 데이터의 운영과 개선된 데이터 서비스라는 우수함을 보여왔었다[5, 7]. 하지만, 시스템에 동시 접근하는 사용자의 수가 일정 한계를 넘을 경우 시스템의 성능이 현격히 저하되는 현상이 발견되어 새로운 문제 즉 범위성(scalability) 문제는 클라이언트-서버 시스템의 내재된 문제로 많은 개선책에 대한 연구가 진행되어 왔다. 해결 방법 중에 한 가지는 클라이언트 사이트에 이미 사용된 데이터를 임

†E-mail : dk_jhpark@dankook.ac.kr

시로 저장하는 캐싱(caching)이 유효한 것으로 알려져 있다[6]. 또한 고속 네트워크의 실용적인 보급이 확대되면서 네트워크를 통해 연결된 사용자가 저장하고 있는 데이터를 이용하는 방법도 제안되었다[3].

적은 양의 단순 데이터를 위한 데이터베이스 환경과 달리 데이터의 양이 많고 다수의 사용자가 데이터를 공유하면서 협업을 기반으로 특정 목적을 달성하는 작업 환경에서의 데이터베이스 시스템은 또 다른 응용환경의 특성을 보여준다. 이러한 환경에서는 각 사용자는 논리적 측면에서 데이터베이스의 특정 부분(working area)을 많이 접근하는 양상을 보인다[9]. 또한 사용자 간에는 이러한 특정 부분을 공유하는 현상도 볼 수 있다. 또한, 각 특정 부분 데이터베이스에 접근하는 사용자들은 다른 특정 부분에 접근하는 빈도수가 적은 것으로 연구되었다[1, 9]. 이러한 양상은 여러 가지 응용 분야에서 그 예를 찾아볼 수 있으며, 반도체 디자인 분야에서도 유사한 경우를 찾아볼 수 있다[10]. 상위 환경과 유사한 데이터 접근 형태를 보이는 환경에서 클라이언트-서버 데이터베이스를 적용하였을 때, 클라이언트 캐싱은 접근된 데이터를 서버가 아닌 자체 캐싱 저장소에서 찾을 수 있는 확률이 높기 때문에 캐싱을 이용하여 성능을 개선할 수 있다[2, 4~6]. 이 논문에서 제안된 구조는 유사한 데이터를 사용하는 클라이언트들을 하나의 논리적 클러스터로 구성하여, 기존의 클라이언트-서버가 가지는 범위성의 한계를 해결하고자 한다.

2. 캐싱 서버를 이용한 구조

클라이언트-서버 데이터베이스에서 모든 클라이언트가 발생하는 데이터에 대한 요구는 서버로 전달된다. 클라이언트로부터 발생되는 데이터 요구는 서버 캐싱이 적용되었을 경우, 먼저 요구된 데이터가 캐시에 저장되었는가 하는 여부를 알기 위해 자체 캐시 내용에 대한 정보를 저장하고 있는 자료구조를 데이터 식별자(object ID)와 비교를 통해 검색한다. 요구된 데이터에 대한 캐시 검색이 실패했을 경우에는 데이터를 저장하고 있는 서버로 데이터 요구 메시지가 전달된다. 만일 캐시에 성공할 경우에는 데이터 객체가 클라이언트로 전송된다. 서버 캐시에 저장되어 있지 않은 데이터를 관리하는 서버는 요구된 데이터를 먼저 캐시관리자에 전송하여 클라이언트에게 전달되기 전에 서버 캐시에 저장을 한다. 다수의 동시 클라이언트가 데이터에 접근하는 환경에서 한 클라이언트가 데이터 갱신을 할 때는 다른 모든 클라이언트로부터 갱신에 대한 동의를 구하기 위해 자료소환(callback) 방법을 사용한다. 갱신 허락은 모든 클라이언트로부터 동의를 구한 다음에야 이루어진다. 이와 같이 클라이언트-서버 데이터베이스 환경에서는 서버가 데이터를 저장하고 관리하는 총괄적 역할을 수행하고, 아울러 데이터의 읽기/쓰기 작업 간의 조절을 담당하고 있다.

서버 작업량을 감소시켜 시스템 성능을 높이기 위해서, 서버와 클라이언트 계층 사이에 새로이 캐싱 서버(ICS) 계층을 사용하는 방법이 있다. 클라이언트 집합은 다수의 고정 클러스터로 분할하여 한 클러스터 당 담당 캐싱 서버를 설정하는 것이 새로운 방법의 기본 구성이다. 따라서 데이터 요구는 먼저 캐싱 서버에서 서비스 만족 여부를 검사하고, 클러스터 내부에서 만족되지 못한 데이터 요구만이 서버 계층에 의해 충족된다. 결과적으로 데이터 서비스의 많은 양을 서버가 아닌 클러스터 범위 안에서 해결할 수 있게 되어, 전체 시스템의 효율을 높일 수가 있다. 이러한 구조는 Fig. 1 에 예시되어 있다.

이 같은 구조에서 자료소환 과정은 두 개의 계층간 자료소환으로 분할 된다. 데이터갱신을 할 때 먼저 갱신허가 신청을 해당 ICS에게 전송한다. 이 때 ICS는 서버에게 요구사항을 전달하고, 서버는 해당 데이터를 소환한다. 자료소환 요구는 중간 계층인 ICS들에게 전달되고, ICS가 담당하고 있는 클러스터에 속하는 클라이언트로부터 해당 데이터를 모두 소환한 다음 서버에 자료소환에 대한 동의를 전송한다. 모든 ICS로부터 자료송환 동의를 모두 받았을 때 서버는 자료갱신 요구를 발송한 ICS에게 허가 발송을 전송한 다음 ICS는 자료갱신을 요청한 클라이언트에게 허가사항을 통보한다.

이러한 삼중 구조는 서버에 집중되는 작업요구를 분산하는 효과를 가져오게 되어 전체 시스템의 효율을 높이는 효과를 가져올 수 있다. 이러한 구조는 전체 클라이언트를 몇 개의 클러스터로 나누어 구성할 수 있

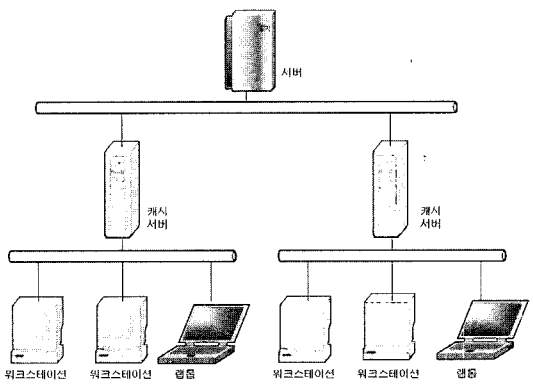

Fig. 1. Architecture using cache server

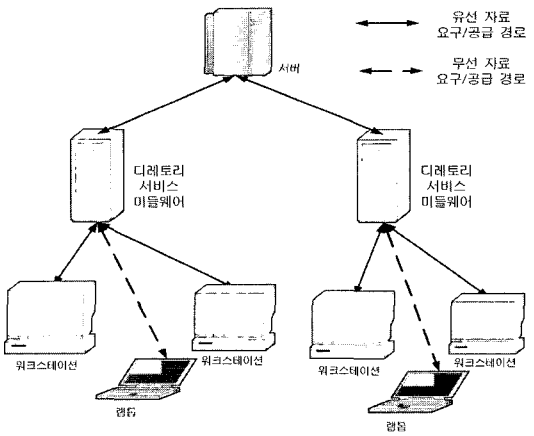

Fig. 2. Architecture using logical client cluster.

으며, 구성 요소들은 네트워크 분할을 이용할 수 있다. 하지만, 네트워크 분할을 이용할 경우, 클라이언트의 소속 클러스터 변경을 해야 할 경우 많은 수작업이 필요함으로 또 다른 문제에 직면한다. 이에 대한 해결책은 클러스터를 논리적인 연결을 이용하여 구성하는 것이다. 따라서, 새로운 논리적 연결 기반 네트워크 구성은 Fig. 2 와 같이 보인다.

논리적 클러스터 구성에 있어서 소속 클라이언트의 선택은 클라이언트의 데이터 이용 패턴의 분석에 기초를 둔다. 클러스터링 알고리즘은 클라이언트의 과거 데이터 접근 정보를 토대로 접근 행태 유사성에 기반하여 클러스터에 대한 소속을 결정할 수 있다. 이 논문에서는 상위 구조의 범위성 문제를 해결할 수 있는 방법임을 검증하기 위해 수행한 실험 결과를 제시한다.

3. 구현 사례

CSIM을 이용한 시뮬레이션은 세 가지 형태로 구성되었다. 각 시뮬레이션은 클러스터의 수를 증가시켜 한 개(SCCA-1), 다섯 개(SCCA-5), 열 개(SCCA-10)의 클러스터로 구성되는 되는 시스템을 두 계층(FCA) 환경과 성능을 비교하였다. 수집한 실험 수치로는 클라이언트 수를 증가시키면서, 서버 수행률을 측정하였다.

Fig. 3은 사용자 증가 시 서버 수행률 을 보여주고 있다. 결과에서 볼 수 있듯이 두 계층 데이터베이스를 적용할 때 클라이언트의 수에 상관없이 서버의 사용률이 제일 많은 것으로 나타난다. 클러스터를 하나로 구성하였을 때, 클러스터의 내부에서 만족 가능한 데이터 요구의 양이 최대가 되어 서버의 수행률은 최저를 나타나고 있다. 클라이언트의 수가 140 일 때, 클러스터

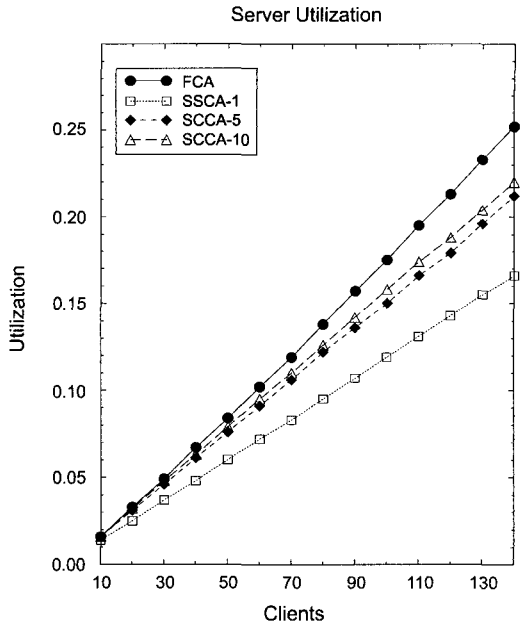

Fig. 3. Server utilization as the number of clients increases.

하나를 사용하는 것이 다섯 개를 사용할 경우보다 약 10%가 넘는 서버의 수행률이 감소되는 것으로 시뮬레이션 결과에서 나타났다. 이는 클러스터 내부 만족도가 여러 개의 클러스터보다는 한 개의 클러스터로 구성될 때 이론적으로 높기 때문이다. 하지만, 실제 상황에서는 서비스 대기 시간이 길어 이 점을 시제품을 통해 확인한다.

실제 시스템 구현은 워크스테이션 여러 대를 이용하여 수행되었다. 여러 구성을 비교하기 위하여 두 계층 데이터베이스 시스템과 제안된 시스템을 구현하였다. 특히 ICS 계층에 캐싱 기능의 유무에 따른 비교를 위하여 제안된 시스템은 2가지로 분리하여 구현하였다. 데이터 요구 발생은 3초 간격을 가지는 프아송 트래픽을 적용하고, 각 데이터에 대한 작업 시간은 0.5 초의 평균을 설정한 지수 분포를 적용시켰다. 데이터베이스 영역은 접근 빈도가 높고 낮음에 따라 두 영역으로 나누고, 각 클라이언트는 접근 빈도가 높은 영역에 약 90 % 정도의 데이터 접근을 하도록 시스템 설정하였다. 클러스터링은 클라이언트들로 하여금 선호하는 데이터 영역을 무작위로 선택하게 한 뒤 수집한 행태 정보를 클러스터링 알고리즘에 적용하여 얻은 결과를 시스템 초기 클러스터링 설정에 사용하였다. 클러스터링 알고리즘은 탐욕적 알고리즘(greedy algorithm)과 유전자 알고리즘(genetic algorithm)을 적용하였다. 정보 수집에 이용된 선호도는 시스템 운영 동안에도 같도록 하

Table 1. System environment values

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

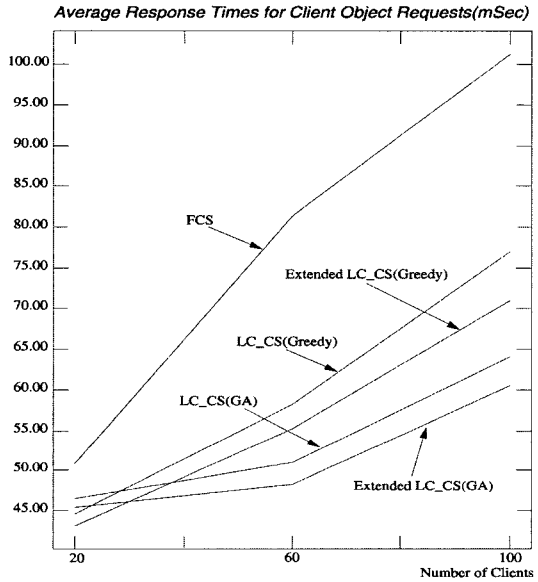

Fig. 4. Average Response Times for Client Object Requests (mSec).

였다. Table 1은 실험에 사용된 데이터베이스 환경에 사용된 수치들을 정리하였다.

Fig. 4는 두 계층 데이터베이스(FCS)와 제안된 시스템에서 미들웨어에 캐싱 기능이 있는 경우(Extended LC_CS)와 없는 경우(LC_CS)에 클러스터 구성을 위해 유전자 알고리즘(GA)과 탐욕적 알고리즘(Greedy)을 적용한 결과를 보여 주고 있다. 그림에서 알 수 있는 것처럼 제안된 시스템은 중간 계층의 캐싱 유무와 클러스터링의 질에 상관없이 기존의 데이터베이스 시스템보다 개선된 서비스를 제공하고 있다.

4. 결 론

클라이언트-서버 데이터베이스는 범위성 문제에 있어서 제한된 효율성이라는 문제를 내포하고 있다. 사용자의 통신 방법이 유무선으로 혼용되는 환경에서 이 문제는 더욱 심각해진다. 이에 대한 대안으로 사용자의 데이터 사용 형태에 기반하여 계층을 확장하고 중간 계층에 서버로의 데이터 요구의 일부를 처리하는 계층을 가진 새로운 데이터베이스의 구조를 제안하였다. 시제품을 이용한 실험에서 제안된 시스템은 클라이언트-서버 시스템보다 데이터 서비스 면에서 월등히 개선되는 것을 알 수 있다. 같은 경향을 시뮬레이션을 통해서도 알 수 있다.

참고문헌

1. Helal, S., Hammer, J., Zhang, J., and Khushraj, A., "A Three-tier Architecture for Ubiquitous Data Access," In Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Beirut, Lebanon, June 25-29, 2001.
2. Blaze, M. and Alonso, R., "Dynamic Hierarchical Caching in Large-scale Distributed File Systems," In Proceedings of the 12th International Conference On Distritubed Computing Systems, Yokohama, Japan, 1992.
3. Dahlin, M. Mather, C. Wang, R. Anderson, T. and Patterson, D. "A Quantitative Analysis of Cache Policies for Scalable Network File Systems," In Proceedings of the Sigmetrics Conference on Measurement and Modeling of Computer Systems, pp. 150-160, Nashiville, Tennessee, 1994.
4. Delis, A. and Roussopoulos, N., "Performance Comparison of Three Modern DBMS Architectures," IEEE Transactions on Software Engineering, Vol. 19, pp. 120-138, 1993.
5. Fang, D. and Ghandeharizadeh, S., "An Experimental System for Object-Based Sharing in Federated Databases," VLDB Journlal, Vol. 5, pp. 151-165, 1996.
6. Franklin, M. Carey, M. and Livny, M., "Transanctional Client Server Cache Consistency: Alternatives and Performance," ACM Transactions on Database Systems, Vol. 22, pp. 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 Rastogim, R., "Client Based Logging for High Performance Distributed Architectures," In Proceedings of the 12th International Conference on Data Engineering, pp. 344-351, New Orleans, LA, USA, 1996.

9. Ranft, M. A., Rehm, S. and Dittrich, K. R., "How to Share Work on Shared Objects in Design Databases," In Proceedings of Sixth International Conference on Data Engineering, Los Angeles, CA, 1989.

10. Wolf, W., "An object-oriented, procedural database for VLSI chip planning," In Proceedings of the 23rd ACM/IEEE conference on Design automation, Las Vegas, Nevada, United States, pp. 744-751, 1986.