캐쉬 Forward/Backward기법을 이용한 데이터 검색에 관한연구

김수장
건국대학교 정보통신원

A Study on the Data Retrieval By Using a Cache Forward/Backward Technique

Soo-Jang Kim
Information & Communications Insitute of Konkuk University

Abstract - 최근, 인터넷 사용자가 급증하면서 빠른 서비스에 대한 문제가 큰 관심이 되고있다. 특히 . 데이터베이스 시스템에서 저장 삭세 수정 등은 사용자에게 긴 대기시간을 요구할 수도 있기 때문에 사용자의 불평을 야기할 수 있다. 이 논문에서는 3-티어 방식에서 요즘 많이 사용되는 application server의 cache에 대해서 말하고자 한다. 기존 application server는 데이터를 application server cache에 저장하여 같은 데이터를 서비스할 경우 server의 cache를 사용하지만 이 논문에서 제안하는 것은 접속된 client를 관리하여 각각의 client에 cache를 만들고 application server나 또는 데이터베이스 server가 서비스를 하지 못할 경우는 가장 최근의 데이터를 가지고 있는 client를 찾아 client cache에 있는 데이터를 서비스 하자는 것이다.

1. 서 론

요즘 많이 사용하는 것이 캐쉬 서버인데 사용자가 기존에 서비스했던 같은 데이타를 요구할 경우 데이타를 서버에서 다시 가져다 서비스하기 보다는 캐쉬 서버에 저장된 데이타를 보내줌으로 서버에 대한 부하를 줄일 수 있고, 또한 데이타를 압축해서 보냄으로 네트워크 부하를 감소시킬 수 있다.

본 논문에서 제시하고자 하는 것은 기존의 캐쉬 forward 아키텍쳐에 캐쉬 backward 아키택쳐를 추가하여 각각의 클라이언트 캐쉬에 있는 데이타를 서버에서 사용할 수 있게 하자는 것이다. 캐쉬 backward를 사용하면 데이타베이스 서버에서 서비스를 할 수 없는 경우 같은 데이타를 가지고 있는 클라이언트들을 찾아 데이타를 서비스함으로 사용자의 기다림 없이 신속한 서비스가 가능하게 된다.

본 논문에서는 캐쉬 forward/backward 아키텍쳐를 설계 및 구현하고, 구현한 프로그램을 실행함으로 캐쉬 forward/backwa-rd 아키텍쳐를 사용한 경우와 그렇지 않는 경우의 서비스 속도의 차이를 비교해 보았다.

2. 본 론

2.1 연결 케쉥

연결 케쉥은 데이타베이스 수행을 최고로 하기 위해서 데이타베이스 서버에 대한 기존 연결을 재사용함으로 각각의 클라이언트에 대한 요구가 있을 경우 연결 인스턴스를 다시 생성하지 않게 되어 부하를 줄일 수 있다 [20].

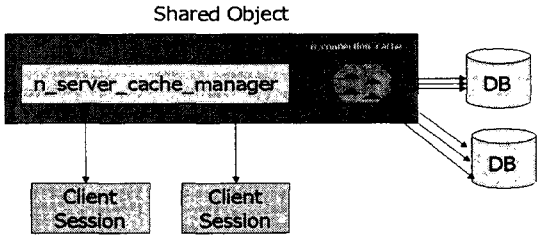

그림 2.1 연결 케쉬 아키텍쳐

그림 2.1은 연결 캐쉬의 아키텍쳐를 보여주고 있다. 일반적으로 서버는 클라이언트가 데이타베이스 서버에 대한 연결을 요구할 경우 처음에는 데이타베이스 서버에 대한 연결 객체를 생성하여 연결을 해준다. 그러나, 요구가 끝나면 데이타베이스 서버에 대한 연결을 물리적으로 끊어버리지 않고 논리적으로만 연결을 끊고 연결 객체을 연결 캐쉬에 넣어서 여전히 데이타베이스 서버에 대한 연결이 열린 상태가 되도록 한다.

2.2 캐쉬 Forward 아키텍처

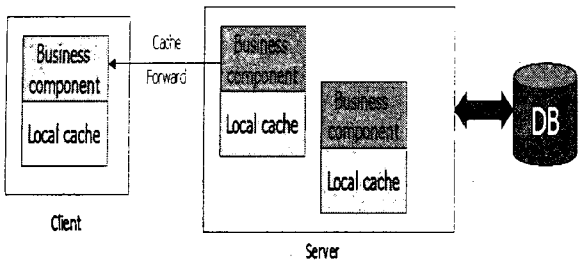

그림 2.2 캐쉬 forward 아키텍쳐

캐쉬를 클라이언트에 두어 서버에서 캐쉬를 제어하는 것이 캐쉬 forward 아키텍쳐이다. 이 방법에서는 서버 측의 데이타가 갱신된 경우 서버에서 클라이언트들을 접근하여 클라이언트의 캐쉬를 갱신함으로 데이타 일관성을 유지한다 [16].

2.3 서버 푸시

푸시 기술 개발 도구들이 채택하고 있는 방법은 풀 방식을 이용한 푸시와 멀티케스팅을 기반으로 한 푸시의 두 종류로 구분될 수 있다. 푸시를 가장한 풀 방식은 대부분의 푸시 기술 개발 도구들이 채택하고 있는 방법으로 사용자가 정의해 둔 시간 간격으로 서버에 접속해 변경된 컨텐트만을 다운로드한다. 앞에서도 언급한 것처럼 다수의 사용자에게 컨텐트를 보내주기 위해서 서버는 많은 양의 사용자 데이타베이스를 관리하여야 한다. 그러므로, 사용자가 증가할수록 서버에 걸리는 부하가 커진다는 단점이 있다 [10].

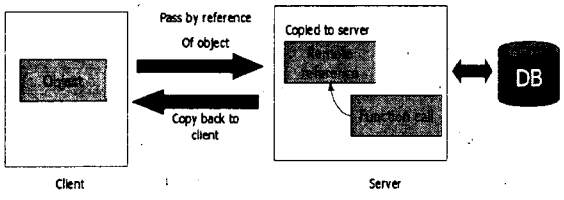

그림 2.3 클라이언트에게 메시지 푸싱하는 과정

그림 2.3은 서버에서 클라이언트에게 메시지를 보낼 경우의 처리과정을 보여준다. 이것은 서버 푸시 (push) 기술을 이용하는 것으로 클라이언트는 서버에 접속 시 서버가 자신을 호출 가능하도록 서버에게 객체 참조를 보낸다. 서버는 리모트 객체라고 부르는 이 객체에 대한 참조를 생성하고 복사하여 서버에 보관한다. 서버에서 다시 클라이언트를 호출할 경우 리모트 객체 참조에 대한 함수 호출를 네트워크를 통해 클라이언트에 다시 보낸다 [5].

2.4 서버 프로그램 설계 및 구현

본 논문에서 설계한 캐쉬 forward / backward 시스템의 서버 프로그램 클래스 다이어그램은 그림 2.4와 같다.

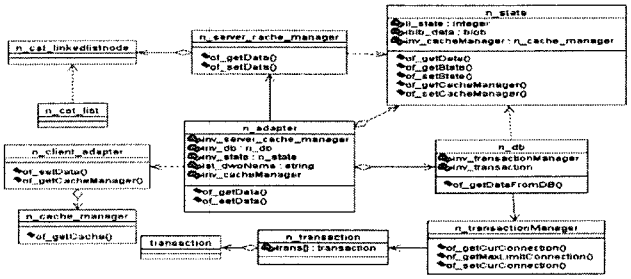

그림 2.4 서버 프로그램 클래스 다이어그램

2.5 클라이언트 프로그램 설계 및 구현

본 논문에서 설계한 캐쉬 forward/backward를 구현하기 위한 클라이언트 프로그램 클래스 다이어그램은 그림 2.5과 같다.

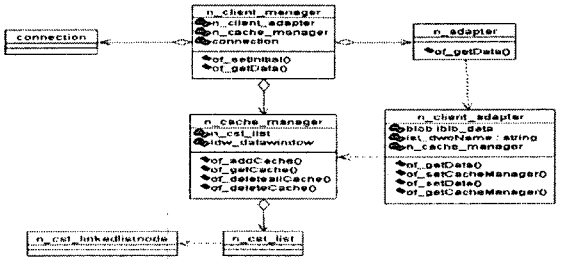

그림 2.5 클라이언트 프로그램 클래스 다이어그램

2.6 데이타 갱신 처리

본 논문에서는 데이타 갱신을 처리하기 위해 클라이언트의 데이타가 변경된 경우 변경된 데이타에 대한 상태 플래그와 데이타 버퍼를 서버에 보내고, 서버에서는 변경된 상태 플래그의 데이타만 데이타베이스 서버에서 수정한다.

클라이언트의 데이타와 서버의 데이타를 동기화 시키기 위해서는 변화가 있을 때 데이타 버퍼와 상태 플래그를 클라이언트와 서버사이에 backward 또는 forward 시킴으로 데이타 일관성을 유지한다.

그림 2.6은 클라이언트의 데이타가 갱신된 경우 서버에 필요한 정보를 보내는 과정을 보여준다.

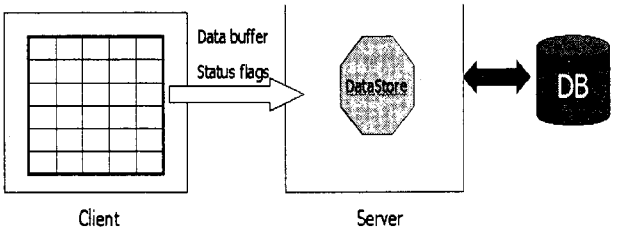

그림 2.6 데이타 갱신 처리

2.7 데이타 갱신 처리 과정

클라이언트에서 데이타를 변경한 경우, 즉 삽입, 갱신, 삭제를 한 경우 변경된 데이타 및 상태 플래그는 서버의 리모트 객체에 전달이 된다. 서버 프로그램은 전달된 상태 플래그를 보고 변경된 데이타에 한해서 데이타베이스 서버에서 해당 데이타를 갱신하도록 한다.

갱신이 성공적으로 수행되면 변경을 요청한 클라이언트에게 성공적으로 수행되었음을 알리기 위해 클라이언트의 상태 플래그를 초기화시킨다. 서버는 데이타가 갱신되었음을 알리기 위해 같은 데이타를 검색하고 있는 클라이언트를 캐쉬에서 찾아 각각의 클라이언트에게 해당 데이타가 갱신되었음을 알리고 변경된 데이타를 보낸다.

변경된 데이타를 받은 각각의 클라이언트는 서버가 보낸 데이타로 갱신함으로써 데이타 일관성을 유지한다. 그림 2.7은 데이타 갱신 처리 과정을 보여준다.

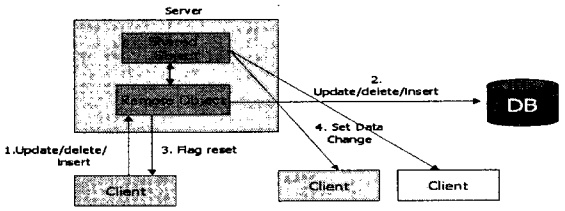

그림 2.7 데이타 갱신 수행 과정

2.8 공유 객체를 이용한 클라이언트 정보 관리

본 논문에서는 서버에 접속한 클라이언트의 정보를 관리하기 위해서 공유 객체를 사용하는데, 공유 객체는 다수의 클라이언트들이 공유할 수 있는 객체이며 분산환경에서 상태정보를 유지관리 하는데 용이하게 한다. 특히, 공유 개체를 사용함으로써 데이타베이스에 대한 접근을 줄임으로 다른 프로세스들이 데이타베이스 서버에 접속할 수 있게 한다.

공유 객체는 주 쓰레드와 다른 쓰레드에서 수행되며, 주 쓰레드또는 클라이언트 세션에서 실행가능 하다. 분산환경에서 공유 객체는 서버의 주 세션 또는 클라이언트 세션에서만 접근이 가능하다.

그림 2.8은 각각의 세션에서 공유 객체를 접근하는 것을 보여준다.

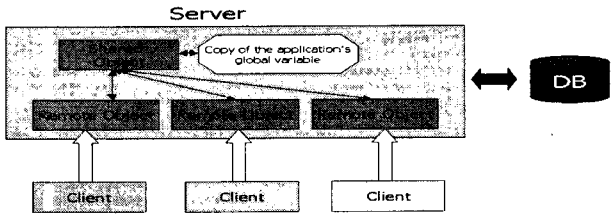

그림 2.8 공유 객체로의 접근

2.9 테스트

기존방법(방법1)과 cache backward 방법을 이용한 테스트 결과는 아래 표와 같다.

사용table명	방법1	방법2 (backward사용)	Backward사용시 성능평가
customer	35.592초	18.186초	48.90%향상
product	41.390초	20.930초	49.43%향상
department	37.444초	22.853초	38.97%향상

3. 결 론

본 논문에서는 클라이언트에게 보다 신속히 서비스를 제공하기 위해서 캐쉬 forward/backward라는 소프트웨어적인 방법을 제시하였다. 물론 서버의 용량을 늘리고 네트워크의 대역폭을 늘리면 이보다 훨씬 성능은 뛰어나겠지만 클라이언트 수가 증가할 때마다 고비용을 들여서 네트워크 장비, 서버 컴퓨터을 구입한다는 것은 많은 무리가 따를 것이다.

본 논문에서 제시하는 캐쉬 forward/backward 방법은 이러한 하드웨어의 고비용을 소프트웨어적으로 보완한 것이다. 기존 캐쉬 서버가가지고 있는 데이타를 클라이언트로 하여금 로컬 캐쉬를 이용하여 가지고 있게 함으로 서버에 부하가 많이 걸린 경우는 클라이언트에서 서버로 데이타 서비스를 하게 한다. 그렇게 함으로써 데이타베이스 서버의 부하로 인한 지연을 줄이자는 것인데, 이것은 좀더 클라이언트에게 신속한 서비스를 제공할 수 있는 방법이 될 것이다.

서버는 많은 클라이언트의 접속으로 인해 신속한 서비스를 제공하는 것이 불가능하다고 판단될 경우 기존 접속한 클라이언트에게 로컬 캐쉬에 보관하고 있는 데이타를 backward로 서버에 보내주도록 요청함으로서 대기 상태에 있는 클라이언트들에게 서비스를 제공할 수 있게 된다.

향후 더 연구되어야 할 과제로는 클라이언트에 관한 정보를 좀더 구체적이고 효과적으로 관리해야 할 것이며, 클라이언트에 대한 접속비용과 데이타베이스 접속에 대한 비용을 산출하여 저 비용으로 접근할 수 있는 서비스 알고리즘을 연구하는 일이다.

[참 고 문 헌]

[1] Chandak, R., and Chandak, P., Advanced Powerbuilder7 Technique, Wiley, 2000.

[2] Clausen, G., Writing Multithreaded Powerbuilder Applications, Sybase PowerLine, 1998.

[3] Howard, B., Millard, B., Boris, G., William, G., and Andy, T.,
Powerbuilder Foundation Class Library Professional Reference, McRraw-Hill, 1998.

[4] Larman, C., Applying UML and Patterns, Prentice Hall, 1998.

[5] Li, K., and Hudak, P., "Memory Coherence in Shared Virtual Memory Systems," ACM Trans. on Computer Systems, Vol.7, No.4, pp.321-359, Nov. 1989.

[6] Myers, A.C., and Liskov, B., "Parameterized Types for Java," Proceeding of the 24th ACM Symposium on Principles of Programming Languages, Paris, France, January 1997.

[7] Powerbuilder, Powerbuilder Foundation Class Library, http://www.pfcguide.com.

[8] Powerbuilder, Shared Object in Powerbuilder, http://www.pdbr.com.

[9] Richard, B., PFC Programmer's Reference Manual, Manning, 1998.

[10] Stumm, M., and Zhou, S., "Algorithms Implementing Distributed Shared Memory," IEEE Computer, pp.54-64,

May 1990.

[11] Sybase, Multi-thread in Powerbuilder, http://www.sybase.com

[12] Sybase, Powerbuilder Application Techniques, 2000.

[13] Sybase, Powerbuilder User Guide, 2000.

[14] Willis, M.A., Ansynchronous Processing in Powerbuilder 6, Sybase PowerLine, 1998.

[15] 김구수, 엄영익, "분산 시스템에서의 부하 공유 기법 설계 및 성능평가," 한국정보처리학회 정보처리학회논문지, Vol.4, No.8, pp.2092-2105, 1997.

[16] 데이텍주식회사, 오브젝트스토어 기술세미나, 1998.

[17] 박 석, 데이타베이스 설계, 홍릉과학출판사, 1996.

[18] 서대화, "다중쓰레드 프로그래밍을 위한 분산공유메모리 관리 기법," 한국정보처리학회 정보처리학회논문지, Vol.3, No.4, pp.791-792, 1996.

[19] 서무영, 홍현술, 한성국, "Push 서버를 이용한 원격교육시스템의 구현," 공업기술개발연구지, Vol.17, No.2, pp.73-77, 1997.

[20] 신원, 김재현, 이경현, "Pull 및 Push 기술을 이용한 원격강의의 제안과 구현," 한국멀티미디어학회 춘계학술발표논문집, Vol.0, No.0, pp.303-304, 1998.

[21] 임혜정, 김명, "워크스테이션 클러스터 상에서 분산공유메모리 인터페이스로 배열 데이타의 공유를 지원하는 Java 패키지의 설계와 구현," 한국멀티미디어학회 멀티미디어학회논문지, Vol.2, No.3, pp.355-365, 1999.