효과적인 웹서비스를 위한 메시지 캐싱의 설계 및 구현

이득룡° 송하주
부경대학교 컴퓨터공학과
{drlee°, hajusong}@pknu.ac.kr

Design and Implementation of message caching for efficient Web services

Deuk-Ryong Lee°, Ha-Joo Song
Dept. of Computer Eng. Pukyong National University

요 약

현재 웹서비스는 전 세계적으로 광범위 하게 사용되어지고 있다. 그러나 인터넷 이용의 급격한 증가로 웹서비스 서버와 네트워크의 부하가 급격히 증가하였으며, 사용자들이 요청한 웹서비스의 응답 지연시간이 점차 증가하게 되었다. 웹서비스는 그 기능과 의미에 따라 웹서비스 객체의 생명 주기가 다르다. 이 때 일정 기간의 생명 주기를 가지는 객체를 클라이언트에서 캐싱을 하여 성능을 증진시킬 수 있을 것이다. 본 논문에서는 웹서비스 메시지 캐싱을 통한 성능향상을 위하여 웹서비스 메시지 캐시를 설계하고 구현하였으며, 실험평가를 통해 캐싱 되었을 경우와, 캐싱 되지 않았을 경우에 대해 성능을 평가한다.

1. 서 론

웹의 등장 이후, 사용자는 시간과 장소에 구애받지 않고 인터넷을 통하여 다양한 정보를 실생활 및 비즈니스에 이용하기를 원하고 있다. 이에 산업체에서는 기존의 시스템을 e-비즈니스화 하면서 다양한 분야에 개방과 통합의 작업을 하고 있다. 이같이 산업체에서 주도하고 있는 재통합 움직임으로 대표되는 것이 웹서비스이다. 웹서비스(Web Service)란 인터넷을 이용한 오픈 네트워크를 통해 단일한 비즈니스 또는 다수의 비즈니스 업체 간의 기존 컴퓨터 시스템 프로그램을 결합시키는 표준화된 소프트웨어 기술로서 이러한 표준 기술을 이용해 모든 비즈니스를 가능케 하는 활동을 일컫는다.[1]

현재 웹서비스는 전 세계적으로 광범위하게 사용되어지고 있다. 그러나 인터넷 이용의 급격한 증가로 웹서버와 네트워크의 부하가 급격히 증가하였으며, 사용자들이 요청한 웹서비스의 응답 지연시간이 점차 증가하게 되었다. 웹서비스의 성능 향상을 위한 방법으로 [2,3,4,5,6] 등이 있다. 웹서비스는 그 기능과 의미에 따라 웹서비스 객체의 생명 주기가 다르다. 이 때 일정 기간의 생명 주기(lifetime)를 가지는 객체를 클라이언트에서 캐싱(caching)하여 성능을 증진시킬 있다. 본 논문에서는 서비스 제공자에게 매번 결과를 요청하지 않고 웹서비스 메시지 캐시를 사용하여 성능을 향상시키고자 한다.

본 논문은 다음과 같은 내용으로 구성된다. 제 2장에서는 시스템 모델을 설명하고 제 3장에서는 성능평가를 보이고 마지막으로 제 4장에서는 결론과 향후 연구 방향에 대해 기술한다.

2. 설계 및 구현

본 논문에서는 웹서비스 메시지를 캐싱하여 성능을 높이기 위해 서버의 응답 메시지(SOAP 메시지)를 캐싱하는 기법을 사용한다. 또한 캐시 시스템과 상호작용하는 클라이언트 스텁(stub) 파일을 WSDL문서의 주소 입력만으로 스텁 파일을 생성할 수 있도록 한다. 일반적으로 웹서비스를 호출하기 위해서는 클라이언트 측에서 자바 스텁 클래스를 사용한다. 스텁 클래스는 WSDL문서를 이용하여 생성한다. 응용프로그램은 클라이언트 스텁을 통해 웹서비스를 호출하고 서버로 요청메시지를 전달한다.

그림 1은 웹서비스 캐시 시스템의 구성과 스텁 파일을 생성하는 것을 보여주고 있다.

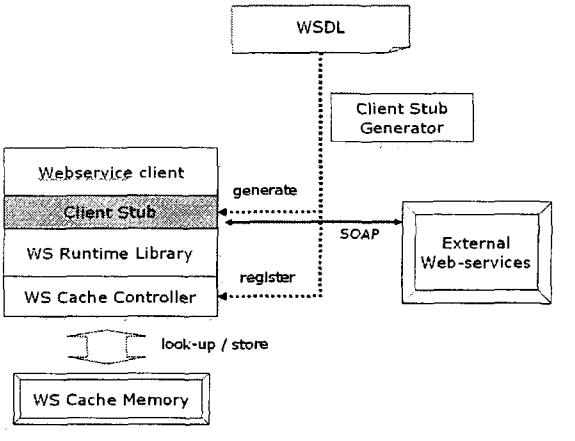

그림 1 웹서비스 메시지 캐시 시스템 구성도

본 연구에서는 웹서비스에서 메시지 캐싱을 하기 위하여 클라이언트 스텁이 서버 측으로 서비스 요청을 보내는 것을 웹서비스 캐시 시스템으로 요청 메시지를 보내도록 처리하기 위하여 스텁 생성모듈을 설계하고 웹서비스 응답 메시지를 캐싱 하기위하여 캐시 시스템을 설계하였다.

이 논문은 2005년도 두뇌한국21사업에 의하여 지원되었음

2.1 웹서비스 캐시 및 스텁 생성모듈 설계 및 구현

스텁코드 생성모듈(Client Stub Generator)은 AXIS의 WSDL2Java를 기반으로 한다. 웹서비스에서 사용되는 클라이언트 스텁은 클라이언트가 서비스를 요청하면 매번 서버로 요청 메시지를 보낸다. 스텁의 이러한 기능을 서비스 요청 시 캐시 시스템으로 서비스 요청 메시지를 보내기 위해 WSDL2Java를 비롯한 스텁생성 관련 클래스를 상속받아 캐시 시스템에 필요한 파일로 재정의(overriding)한다.

그림 2는 웹서비스 메시지의 호출을 캐시를 통해 처리하는 과정을 개념적으로 보인 것이다.

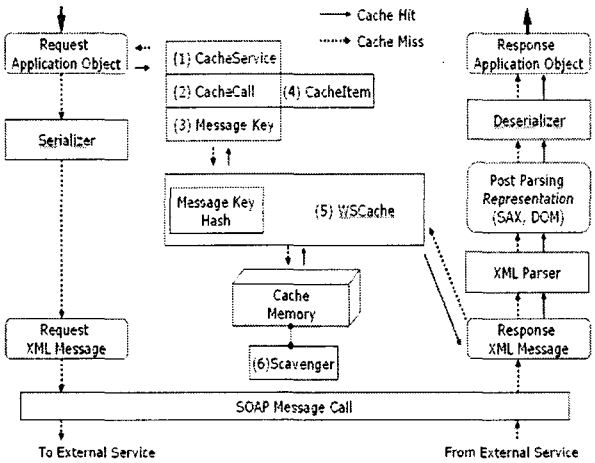

그림 2 웹서비스 메시지 캐시 시나리오

본 연구에서 사용할 캐시 시스템은 기존의 AXIS 클래스를 상속하도록 한다. 캐시 시스템에서 각 클래스의 역할은 다음과 같다.

(1)CacheService 클래스: AXIS의 Call 객체를 호출하는 대신 CacheCall 객체가 호출되도록 Locator의 동작을 조작하는 클래스이다.

(2)CacheCall 클래스: 스텁에 의해 호출되며 응용프로그램 객체를 SOAP메시지로 마샬링(marshalling)하고 호출결과 값을 언마샬링(unmarshalling)하는 것 등의 작업을 수행하며, Invoke()메소드는 호출메시지에 대한 키를 생성한다. 키 값의 구성방법에 대해서는 MessageKey 클래스에서 상세히 다루어진다. 생성된 메시지키를 이용하여 WSCache에 해당 메시지에 대한 캐시 데이터가 존재하는지를 검사하고 없을 경우 실제 웹서비스 메시지를 호출한다.

(3)MessageKey 클래스: 캐싱된 메시지를 구분하기 위한 클래스로 서비스URL, 호출연산 정보, 호출 인자 값들을 유지하는 클래스이다. 메시지 키 값은 WSCache내의 메시지 해시에 의해 사용되므로 equals 메소드를 오버라이드하여 키 값들 간의 비교가 가능하도록 한다.

(4)CacheItem 클래스: 캐싱된 메시지 키와 해당 메시지의 생명주기를 유지하는 클래스이다.

(5)WSCache 클래스: 캐시 관리자로서 Hashtable를 이용하여 메시지를 캐싱하고 캐시와 관련된 작업을 수행한다.

(6)Scavenger 클래스: 일정시간마다 캐싱된 메시지의 생명주기를 체크하여 생명주기가 경과한 메시지에 대해 삭제를 수행한다.

웹서비스 메시지 캐시 시나리오는 다음과 같이 수행된다.

(1) 클라이언트로부터 서비스 요청 메시지를 받으면 캐시 시스템에서 동일한 요청 메시지가 있는지 확인한다.

(2-1) 동일한 메시지가 있을 경우 캐시 저장소에서 응답 메시지를 클라이언트에게로 되돌려준다.

(2-2) 동일한 메시지가 없을 경우 캐시 시스템은 서버로 요청 메시지를 전달한다.

(3) 서버의 응답 메시지를 캐시 저장소에 저장하고 클라이언트에게 응답 메시지를 전달한다.

3. 성능평가

3.1 실험환경

본 실험을 위해 웹서비스를 제공하는 서버와 서비스를 사용하는 클라이언트로 구성한다. 서버 컴퓨터의 사양은 CPU는 Intel Pentium4 3.0GHz를 사용하고, RAM은 1GB를 사용하였다. 클라이언트 컴퓨터의 사양은 CPU는 Intel Pentium4 2.4GHz를 사용하고, RAM은 512MB를 사용하여 성능평가를 하였다.

3.2 실험조건

실험은 전송 메시지 크기가 1Kbyte, 100Kbyte, 1Mbyte, 2Mbyte의 데이터 호출에 대한 캐시(생명주기 10초)를 사용하였을 때와 캐시를 사용하지 않았을 때의 서비스호출에 대한 평균응답 시간을 비교하고, 전송 메시지 크기가 1Mbyte인 데이터를 사용하였을 경우 캐시의 생명주기에 따른 평균응답 시간을 비교한다.

3.2 실험결과

그림 3는 각각의 전송 메시지 크기에 따라 캐시를 사용한 것과 사용하지 않은 것에 대한 평균응답 시간을 비교한 것이다.

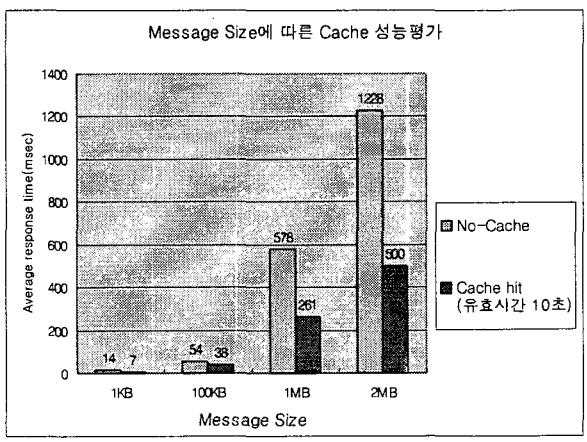

그림 3 Message Size에 따른 캐시 성능평가

그림에서 보여주듯이 전송 메시지의 크기가 증가할수록 평균응답시간이 늘어나게 된다. 캐시를 사용하였을 경우에는 전송 메시지의 크기가 증가할수록 평균응답시간이 크게 줄어드는 것을 볼 수 있다. 이는 캐시를 사용함으로 네트워크 딜레이(delay)와 프로세싱 타임(processing time)이 응답시간에 포함되지 않으므로 더 좋은 성능을 보여주는 것이다.

그림 4는 1Mbyte의 전송 메시지에 대해 캐시 생명주기에 따른 평균 응답시간을 비교한 것이다.

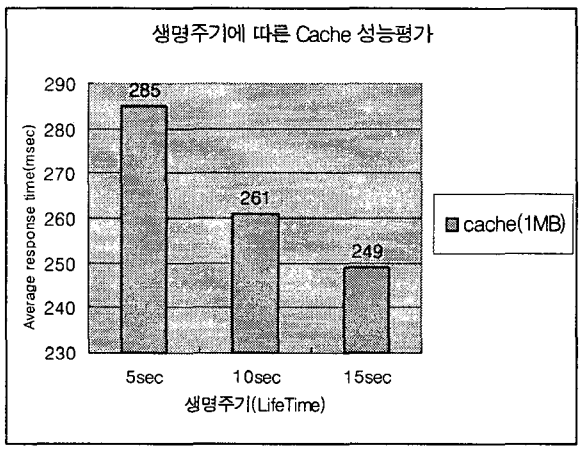

그림 4 생명주기에 따른 Cache 성능평가

그림에서 보여주듯이 생명주기가 증가할수록 평균응답시간이 줄어들게 된다. 이는 생명주기가 길 경우 생명주기가 짧을 때보다 캐싱을 자주하지 않아도 됨으로 이에 대한 오버헤드(overhead)가 줄어들기 때문이다.

4. 결론

본 논문에서 제안한 웹서비스 메시지 캐시 시스템은 클라이언트의 서비스 요청 시 캐시 시스템으로 요청 메시지를 보내고 캐시되어진 메시지가 없을 경우 실제 웹서비스를 제공하는 서버로 요청 메시지를 보낸다. 웹서비스의 응답 메시지(SOAP 메시지)는 캐시 저장소에 저장을 하여 다음번에 동일한 메시지를 요청하게 되면 서버에서 읽어오지 않고 캐시 시스템에서 읽어 클라이언트에게 전달한다. 캐싱을 함으로써 클라이언트의 서비스처리 속도가 향상된다. 실험 결과 적은 양의 데이터를 캐시 하였을 경우는 캐시를 사용하지 않았을 때 보다 미미한 성능 개선을 보였지만, 큰 데이터를 캐시 하였을 경우에는 캐시를 사용하지 않았을 때 보다 높은 성능향상을 보였고, 캐시의 생명주기가 길수록 짧을 때 보다 높은 성능향상을 보였다. 따라서 생명주기를 가지는 대용량 웹서비스에서 본 논문에서 제안한 캐시 시스템을 사용할 경우 웹서비스의 성능을 효율적으로 개선할 수 있다.

참고문헌

[1] 정부연, "웹서비스의 개념과 관련 기업에 미치는 영향", 정보통신정책 제 14권 7호 통권 299호, 2002.

[2] D. Davis ant M, Parashar. "Latency performance of SOAP Implementations". In Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 407-412, 2002.

[3] C. Kohlhoff and R. Steele. "Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems". In Alternate proceedings of the Twelfth International World Wide Web Conference, pages 262-270, 2003.

[4] T. Takase and M. Tatsubori. "Efficient Web Services Response Caching by Selecting Optimal Data Representation". 24th International Conference on Distributed Computing Systems(ICDCS 2004), pages 188-197, 2004.

[5] R. Tewari, M. Dahlin, H. Vin, and J. Kay. "Design considerations for distributed caching on the Internet". In Proc. IEEE 19th Int. Conf. on Distributed Computing Systems, page 273-284, 1999.

[6] J. Challenger, A. Iyengar, and P. Dantzig. "A Scalable System for Consistently Caching Dynamic Web Data". In Proc. INFOCOM '99, 1999.