제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

캐싱 기법을 적용한 푸시 에이전트 모델

최은실*, 김광종, 이연식
*군산대학교 컴퓨터과학과
e-mail : es800322@kunsan.ac.kr

Push Agent Model Appling Caching Technique

Eun-Sil Choi*, Kwang-Jong Kim, Yeon-Sik Lee
*Dept. of Computer Science, Kun-San University

요 약

기존 푸시 기술의 제품들이 가지는 상호 운용성을 해결하기 위한 CORBA 기반의 푸시 에이전트 모델(CPAM:CORBA-based Push Agent Model)은 사용자 요구에 적합한 정보만을 수집한 후, 이를 사용자에게 전송하여 주는 에이전트 프레임웍(Framework)이다. 이러한 푸시 에이전트는 푸시 에이전트 서버가 제공하는 정보의 양과 종류, 그리고 이를 이용하는 동시 사용자 수가 증가함에 따라 서버 시스템의 과부하 및 네트워크 트래픽의 증가가 초래된다.

이를 위해 본 논문에서는 푸시 에이전트 모델에 캐싱 기법을 적용함으로써 자주 요청되어지는 동일한 데이터에 대한 서버 시스템의 과부하를 감소시키도록 한다. 또한, 클라이언트 푸시 에이전트의 상태 정보와 주제 분류에 따른 검색 키워드를 포함하고 있는 사용자 프로파일을 통해 효율적인 캐쉬 메모리 접근을 제공하도록 한다.

1. 서론

오늘날 인터넷 기술의 발전 및 웹의 보편화로 인해 수많은 정보의 확산과 획득이 용이해졌으나 웹 상의 정보량이 급속히 증가함에 따라 사용자들은 필요로 하는 정보에 대해 직접적인 컨텐츠 요구 방식이 아닌 정보 제공자에 의한 수동적인 컨텐츠 제공 방식을 요구하고 있다. 그러나 기존 풀 방식의 인터넷 서비스는 이러한 사용자 요구를 충족시키지 못함으로써, 이를 해결하기 위한 새로운 인터넷 서비스 방식으로서 푸시 방식이 대두되었다[1].

푸시 기술은 분산 컴퓨팅 환경에서 사용자가 원하는 작업 목표만을 명시하면 사용자의 온/오프라인에 상관없이 프로그램이 작업을 수행하여 해당하는 정보들을 사용자가 원하는 시간에 지속적으로 제공하는 서비스 기술이다. 사용자들은 이러한 푸시 기술을 사용함으로써 웹 상에 존재하는 수많은 정보들을 지속적으로 제공받을 수 있으며, 사용자가 원하는 정보들을 찾기 위해 들이는 시간과 노력을 감소시킬 수 있다[2].

그러나 이러한 푸시 방식은 사용자가 원하는 때에만 정보 제공을 수행함으로써 사용자의 정보 요구가 발생할 때까지 제공되어질 정보들은 서버 혹은 클라이언트에 저장되어 있어야만 한다. 이는 멀티미디어 데이터나 수집된 대량의 데이터를 제공하고자 할 때 네트워크 트래픽 증가와 서버 과부하의 문제점을 발생시킨다. 또한, 사용자로부터 동일한 정보의 중복된 요청, 혹은 동시 사용자 수의 증가에 대해서도 서버 과부하의 문제점은 대두된다.

따라서 본 논문에서는 사용자에게 능동적인 정보 제공을 수행하는 푸시 에이전트의 서버 과부하 및 네트워크 트래픽 증가를 해결하기 위한 방안을 제안한다. 본 논문에서 제안하고자 하는 푸시 에이전트 모델은 기존 푸시 기술들이 가지고 있는 상호 호환성의 문제점을 해결하기 위해 이기종간의 상호 운용성을 제공하는 CORBA 를 기반으로 하며, 이를 통해 사용자들은 폭 넓은 정보 서비스를 제공받도록 한다.

또한, 푸시 에이전트 모델에 캐싱을 적용함으로써 동시 사용자 수 증가에 따른 서버 과부하의 문제점을 해결하도록 하며, 클라이언트 푸시 에이전트의 사용자 프로파일을 이용하여 중복된 데이터 요청과 동일한 사용자 요구에 따른 네트워크 트래픽 증가를 해결하였다. 이를 통해 푸시 에이전트 사용자는 대량의 데이터에 대해 대기 지연 시간 없이 정보를 제공받을 수 있다.

2. 관련 연구

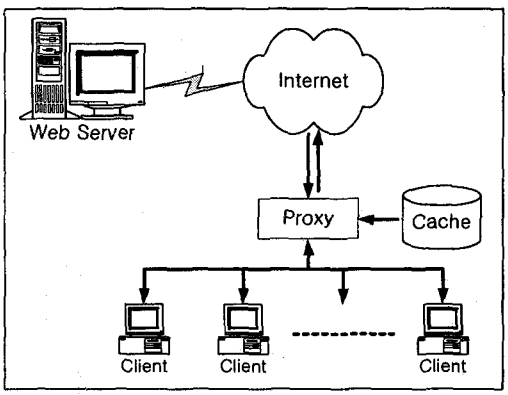
본 장에서는 서버 시스템의 과부하와 네트워크 트래픽 증가 문제를 해결하기 위해 지금까지 제안되어 온 캐싱 기법들과 캐쉬 메모리의 효율적인 관리를 위한 캐쉬 메모리 교체 전략에 대해 기술하였다.

2.1 기존의 캐싱 기법들

웹 서버의 처리 능력과 네트워크 트래픽 감소를 위한 방법으로는 하드웨어적인 기능 향상과 서버 구조의 개선, 그리고 캐쉬를 이용한 방법이 사용된다. 이 중 현재 사용되고 있는 방법으로는 하드웨어적인 기능 향상보다는 서버 구조 변경과 캐쉬를 이용한 연구가 진행되고 있다[3].

서버 구조의 개선 방법은 다중 서버를 둠으로써 확장성은 뛰어난 반면 비용과 복잡성 면에서의 단점을 가진다. 따라서 네트워크 상의 대역폭을 줄이기 위한 방법으로 캐쉬가 많이 이용된다. 이 방법은 사용자의 반복적인 요구를 흡수하기 위해 인터넷의 각 요소마다 프록시 캐쉬를 배치함으로써 데이터들을 지리적으로 가까운 곳에 위치하도록 하는 것이다. 이러한 캐쉬를 사용하는 방법은 사용자 요구에 대한 서버 응답 시간을 줄임으로써 사용자가 정보 요청을 시행한 후 응답을 받을 때까지의 시간을 줄이기 위한 매우 효과적인 방법이다. 그러나, 이 방법은 메모리 내에 캐쉬를 구성하기 때문에 캐쉬의 크기가 제한적이며 이는 대용량의 데이터들을 효율적으로 제공하기 위해서는 한계를 가지게 된다[3,4,5].

다음 [그림 1]은 웹 서버와 클라이언트, 웹 프록시 캐쉬의 구성도이다.



[그림 1] 웹 상에서의 프록시 캐쉬 구성도

이러한 구조의 프록시 캐쉬는 사용자와 웹 서버의 중간에서 사용자의 요청을 처리하는 중간자 역할을 수행한다. 사용자의 정보 요청이 발생하면 프록시 서버는 캐쉬를 검색하여 해당 데이터가 있을 경우, 이를 요청한 사용자에게 제공한다. 만약 사용자가 요청한 문서가 프록시 서버의 캐쉬에서 발견되지 않을 경우, 웹 서버로 사용자 요청이 전송된다. 웹 서버는 사용자 요청에 대해 적절하게 처리하여 사용자에게 처리 결과를 전송한다. 서버로부터의 처리 결과는 프록시 서버에 캐싱된다.

캐쉬 구조를 이용한 대표적인 소프트웨어인 Harvest 시스템은 여러 개의 프록시 서버들을 계층적으로 구성함으로써 웹 캐싱의 성능을 향상시키는 계층적 캐싱 서브 시스템이다.

Harvest 시스템의 동작 흐름은 클라이언트가 지역 Harvest 캐쉬를 통해 웹 객체를 요구하면 캐싱 서브 시스템이 웹 프록시처럼 작동하여 필요한 객체를 클라이언트에게 넘겨주고, 만약 객체가 캐쉬에 없으면 지역 캐쉬는 그 이웃과 부모, 그리고 객체의 최초 호스트를 차례로 방문한다.

이러한 계층 구조의 Harvest 캐쉬는 계층 구조상에서의 위치를 수동으로 조정해 주어야 하며, 프록시 서버들간의 통신 경로에 있어서 유연성을 제공하지 못하고 있다[5,6,7,9].

캐쉬를 이용하는 또 다른 방법으로는 클라이언트가 서버에 문서를 요청했을 때 서버가 가까운 미래에 요청할 것으로 추론되는 문서들을 한꺼번에 클라이언트에게 전송하는 방법도 있다. 이 방법은 서버가 관리하는 각 문서의 통계 정보에 기초한다. 이 방법은 특정 문서가 요청되었을 때 그 다음에 요청되는 문서의 확률을 계산해서 사용함으로써, 문서의 크기가 작을 경우에는 빠른 응답과 서버 로드 감소의 장점을 보인다. 그러나 추론되어 전송되는 데이터가 멀티미디어 데이터와 같은 대용량의 데이터일 경우, 네트워크 트래픽 증가의 문제점을 가진다[7,8,9].

2.2 캐쉬 메모리 교체 전략

캐쉬를 이용하여 네트워크 상의 요청 수와 데이터 양을 감소시킴으로써 사용자에게 빠른 응답을 제공할 수 있다. 그러나 이러한 캐쉬는 제한된 크기를 가짐으로써 캐쉬내의 데이터를 효율적으로 관리해야 할 필요성이 있다. 또한, 사용자들의 요청에 대해 서버로부터 캐쉬에 전달된 데이터는 일정 시간이 지난 후에 수정P삭제됨으로써 계속적으로 증가하는 웹 상의 정보들 중 유효한 정보들만을 사용자에게 제공할 수 있도록 해야 한다[10,11]. 따라서, 캐쉬 내의 데이터는 다음 [표 1]과 같은 교체 전략을 가진다.

[표 1] 캐쉬 교체 전략과 데이터 조건

교체 전략	교체되는 데이터의 조건
LRU (Least-Recently-Used)	최근 접근 시간
LRU-k-TH	최근 k번째 접근 시간
LRU-SIZE	문서의 크기
LFU (Least-Frequently-Used)	참조의 수, 최근 접근 시간
Log(Size)+LRU	문서의 크기, 최근 접근 시간
Hyper-G	마지막 검색 시간, 데이터 크기
Pitkow/Recker	최근 접근 시간, 최근 접근된 문서 크기
Lowest-Latency-First	전송 지연 시간
Static caching	참조수/문서크기

표에서 보여지는 것과 같이 캐쉬에서의 교체 정책 중 기본으로 많이 사용되는 것은 데이터의 참조 수와

데이터 크기, 최근 접근 시간이다. 즉, 데이터의 참조 수가 적은 데이터를 먼저 교체하고, 데이터의 크기가 큰 데이터를 제거하도록 한다.

그러나 캐쉬 기법을 적용하는 시스템은 사용하고자 하는 데이터와 시스템의 목적에 따라 알맞은 교체 전략을 선택하여야 한다.

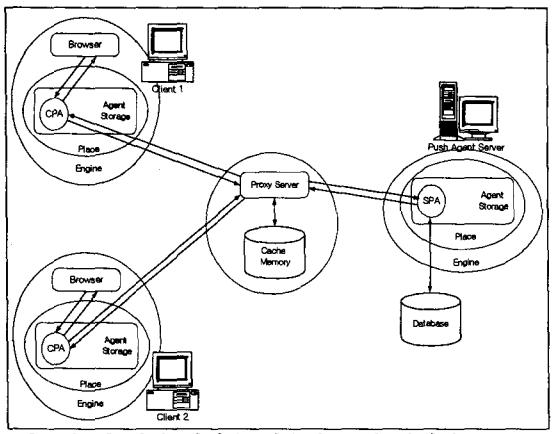
본 논문에서 제안하는 캐싱을 적용한 푸시 에이전트 모델은 웹 상의 데이터를 사용자에게 실시간으로 제공하기 위한 모델로서, 정보의 빠른 성장과 변화를 수용하는 웹의 특성을 수용하기 위해 LFU(Least-Frequently-Used) 교체 전략을 적용한다.

3. 캐싱 기법을 적용한 푸시 에이전트 모델

본 장에서는 CORBA 기반의 푸시 에이전트 모델을 통해 사용자에게 능동적으로 정보를 제공하도록 하며, 서버로부터 클라이언트들에게 정보를 제공할 때 발생하게 될 네트워크 트래픽 증가와 서버 과부하를 해결하기 위해 사용자 프로파일을 이용한 캐싱 기법을 제안한다.

3.1 캐싱 기법을 적용한 푸시 에이전트 모델 설계

푸시 에이전트 모델은 CORBA 이벤트 서비스의 객체 간 서비스 방식에 기반하여 에이전트를 생성하고, 생성된 에이전트는 클라이언트 푸시 에이전트의 상태 정보를 포함한다. 이러한 클라이언트 푸시 에이전트의 상태 정보는 사용자의 정보 요청이 발생할 때, 캐쉬 메모리에 저장된다. 다음 [그림 2]는 캐싱 기법을 적용한 푸시 에이전트 모델의 구조이다.



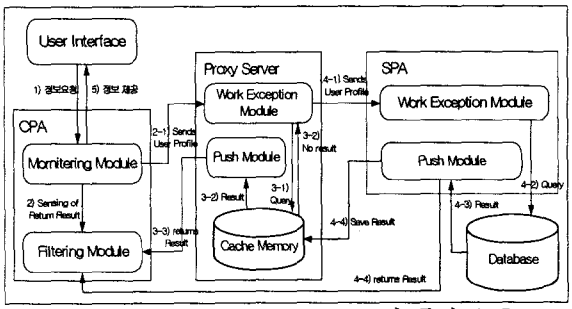
[그림 2] 캐싱 기법을 적용한 푸시 에이전트 모델

위의 그림과 같이 푸시 에이전트 모델은 서버 푸시 에이전트(SPA:Server Push Agent)를 포함하는 푸시 에이전트 서버(Push Agent Server)와 클라이언트 푸시 에이전트(CPA:Client Push Agent) 및 클라이언트 브라우저(Client Browser)를 가진 클라이언트로 구성된다.

클라이언트 푸시 에이전트는 각 클라이언트 시스템 상에 위치하며, 사용자가 정보를 요청하게 되면 클라이언트 푸시 에이전트의 상태 정보와 사용자 정보 검색 키워드를 포함하는 사용자 프로파일을 프록시 서버로 전달한다. 프록시 서버는 사용자 프로파일 내에 포함되어 있는 사용자 키워드를 검색하여 캐쉬 메모리를 접근한다. 캐쉬 메모리에 해당 정보가 있을 경우, 이를 클라이언트 푸시 에이전트로 전송하며 클라이언트 푸시 에이전트는 전달받은 데이터를 사용자에게 제공한다. 만약, 사용자 프로파일의 키워드에 해당하는 정보가 캐쉬 메모리 내에 존재하지 않을 경우 서버 푸시 에이전트를 구동시킨다.

서버 푸시 에이전트는 푸시 에이전트 서버에 위치하며, 클라이언트 푸시 에이전트로부터 사용자 프로파일을 전달받아 데이터베이스를 검색하여 추출된 정보를 클라이언트 푸시 에이전트에 전송한다.

다음 [그림 3]은 사용자의 정보 요청에 대한 클라이언트 푸시 에이전트와 프록시 서버, 서버 푸시 에이전트의 통신 과정이다.



[그림 3] SPA, CPA, Proxy Server 의 통신 흐름도

사용자가 정보 요청을 위한 사용자 프로파일을 작성하면, 클라이언트 푸시 에이전트의 모니터링 모듈은 이를 감지하여 프록시 서버의 실행 모듈을 호출하고, 사용자가 작성한 사용자 프로파일을 전송한다. 실행 모듈은 클라이언트 푸시 에이전트로부터 전송되어진 사용자 프로파일에 포함되어진 검색 키워드와 클라이언트 푸시 에이전트의 상태 정보를 분석하여 캐쉬 메모리를 접근한다. 해당 데이터가 존재할 경우 푸시 모듈로 결과를 전송하고, 푸시 모듈은 이를 클라이언트 푸시 에이전트의 필터링 모듈로 전송한다. 필터링 모듈은 전달 받은 데이터의 중복성을 제거하고, 필터링 모듈을 실시간으로 감지하는 모니터링 모듈은 필터링된 데이터를 사용자에게 제공한다.

만약 캐쉬 메모리에서 적합한 정보를 얻지 못했을 경우, 프록시 서버의 실행 모듈은 서버 푸시 에이전트의 실행 모듈을 호출하고, 사용자 프로파일을 전달한다. 서버 푸시 에이전트의 실행 모듈은 데이터베이스를 검색하여 검색 키워드에 해당하는 데이터들을 서버 푸시 에이전트 내에 위치한 푸시 모듈로 전송한다. 푸시 모듈은 전달받은 데이터들을 클라이언트 푸시 에이전트의 필터링 모듈로 전송하고 또한, 클라이언트 푸시 에이전트의 캐쉬 메모리에 전송함으로써 동일한 데이터가 요청되었을 때 캐쉬 메모리에서 검색 결과를 반환할 수 있도록 한다.

3.2 사용자 프로파일에 기반한 캐싱 기법

제안된 사용자 프로파일은 사용자에 의해 입력된 주제 분류에 따른 우선 순위와 클라이언트 푸시 에이전트의 상태 정보를 포함한다.

주제 분류에 따른 우선 순위는 푸시 에이전트가 검색된 결과를 사용자에게 제공할 때 사용자에게 우선 순위별 정보 서비스를 가능하도록 하며, 데이터베이스 검색 시 대량의 정보들로부터 보다 정확한 정보 수집을 수행할 수 있도록 한다. 또한, 주제 분류에 따른 캐쉬 메모리를 구성함으로써 캐쉬 메모리의 접근 효율성을 가져온다.

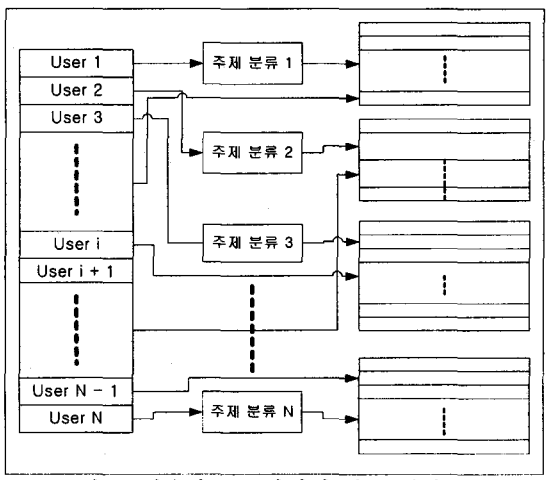
사용자 프로파일의 구성 요소 중 하나인 상태 정보는 정보를 요청한 클라이언트 푸시 에이전트들을 통해서 구성되어진다. 이는 한번 이상 정보 검색을 수행했던 사용자가 동일한 검색을 수행하고자 할 때, 상태 정보가 포인터하고 있는 메모리만을 접근함으로써 검색 수행 시간을 단축할 수 있다.

이러한 사용자 프로파일은 다음 [표 2]와 같다. 사용자 프로파일은 클라이언트 푸시 에이전트의 상태 정보를 통해 다른 사용자 프로파일과 구별할 수 있도록 해주며, 키워드는 각 주제 분류의 공통 키워드이기 때문에 한번만 저장된다.

[표 2] 사용자 프로파일의 구조

CPA State Information	주제분류 1	주제분류 2	.....	주제분류 N
Search Keyword	우선순위 1	우선순위 2	.....	우선순위 N

주제 분류별 사용자 프로파일은 하나의 키워드로 인해 무분별하게 검색되는 기존의 검색 방법의 단점을 보안하며, 사용자 프로파일에 기반한 캐쉬 메모리를 구성하고 접근함으로써 사용자에게 빠른 응답을 제공한다. 다음 [그림 4]는 이러한 사용자 프로파일에 따른 캐쉬의 구조이다.



[그림 4] 사용자 프로파일에 따른 캐쉬 구조

클라이언트 푸시 에이전트 상태 정보는 정보 검색이 요청되어지면 캐쉬 메모리에 상태 정보를 등록함과 동시에 사용자 프로파일의 주제 분류와 키워드를 포인터 하도록 한다. 만약, 상태 정보는 동일하나 검색하고자 하는 키워드가 다른 경우에는 저장되어져 있는 상태 정보에 새로운 검색 키워드를 포인터 하도록 한다.

4. 결론 및 향후 연구

본 논문에서는 사용자에게 능동적으로 정보를 제공하는 푸시 에이전트 모델에 캐싱을 적용함으로써 동시 사용자 수 증가와 지속적인 정보 서비스로 인해 발생하는 서버 과부하 및 네트워크 트래픽 증가의 문제점을 해결하였다. 또한, 클라이언트 푸시 에이전트로부터 전송되어진 사용자 프로파일을 이용해 캐쉬를 구성하며, 사용자 프로파일의 구성 요소인 상태 정보는 각 주제 분류와 키워드를 포인터 함으로써 동일한 정보 요청에 대해 빠른 정보 검색 수행을 제공하도록 한다.

향후 연구 과제로는 다양한 데이터에 대한 캐쉬의 구조가 요구된다. 웹 상에 존재하는 데이터의 종류는 다양하며, 이러한 데이터를 캐쉬 메모리에 저장할 경우의 저장 방법과 저장되어진 데이터의 교체 시간에 대한 연구가 요구된다.

참고문헌

[1] OMG, "Agent Technology Green Paper", Agent Platform Special Interest Group, http://www.objs.com, 2000.

[2] 이윤정, "Push 기술과 정보 배포 기술의 분석 및 동향 연구", 한국정보처리학회 추계학술발표논문집, 제 7 권, 제 2 호, 2000.

[3] 임은지, "연속미디어 데이터의 재생량에 기반한 프락시 캐싱 기법", 부산대학교 전자계산학과 석사학위논문, 2001.

[4] Mohammad Raunak, "Implications of Proxy Caching for Provisioning Networks and Servers", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL.20, NO.7, SEPTEMBER 2002.

[5] 김평수, "웹에서 지연 시간 개선을 위한 푸시 캐싱", 서울산업대학교 산업대학원 석사학위논문, 2001,2.

[6] 최승교, "멀티미디어 데이터를 위한 웹 서버 성능 개선에 대한 연구", 컴퓨터공학연구, 2000.

[7] 백윤철, "Prefetch 하는 웹 캐쉬 프록시의 성능에 대한 연구" , 컴퓨터산업교육기술학회 논문 VOL.02, NO.11, pp. 1453~1464, 2001,11.

[8] Sandra G. Dykes, "Limitations and Benefits of Cooperative Proxy Caching", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL.20, NO.7, SEPTEMBER 2002.

[9] 유해영, "선호도 기반 웹 캐싱 전략", 한국정보과학회 논문지 A, VOL.29, NO.10, pp.530~538, 2000,10.

[10] 최승락, "웹 프락시 서버를 위한 적응형 캐시 교체 정책", 정보과학회논문지, 제 29 권, 제 6 호, 2002,6.

[11] 이명종, "웹에서 멀티미디어 데이터를 위한 푸시 캐싱", 서울대학교대학원 석사학위논문, 1999.