복제에 기반한 고가용성 푸쉬 서버

노진홍° 홍영식
동국대학교 컴퓨터공학과
{jhno°, hongys}@dgu.ac.kr

Highly-Available Push Servers based on Replication

Jin-Hong No° Young-Sik Hong
Dept. of Computer Engineering, Dongguk Univ.

요 약

인터넷의 정보가 많아짐에 따라 인터넷에서 원하는 정보를 얻어 효과적으로 사용하기는 더욱 힘들어지고 있다. 그러므로 관련된 정보에 쉽게 접근하기 위해 얻고자 하는 정보를 인터넷에서 자동적으로 전달해 주는 푸쉬 기술은 중요한 인터넷 기술로 주목받고 있다. 하지만 푸쉬 기술은 정보 전달 방식으로 인해 네트웍 대역폭을 너무 많이 소요하기 때문에 푸쉬 서버의 가용성이 급격히 떨어질 수 있다는 단점을 가지고 있다. 따라서 본 논문에서는 메시지 폭주로 인한 시스템 고장에 대처하고 부하를 분산시킬 수 있는 복제에 기반한 푸쉬 서버를 설계하고, Erlang/OTP로 구현하였다. 성능을 분석하기 위하여 단일 푸쉬 서버를 사용한 결과와 본 논문에서 제안된 구조를 사용한 결과를 비교·분석하였다.

1. 서 론

인터넷이 발달함에 따라 정보량은 그에 비례해서 증가하고 있다. 이에 따라 점차 인터넷상에서 사용자가 원하는 정보만을 선별하여 얻기가 어려워지고 있는 실정이다. 그러므로 사용자가 얻고자 하는 정보만을 효과적으로 전달해주는 푸쉬 기술은 중요한 인터넷 기술로 주목받고 있다. 이전까지는 인터넷에 접속되어 있는 수많은 정보를 이용하기 위해서는 수동적으로 데이터를 끌어내야 했지만, 푸쉬(push) 기술은 이용자에게 필요한 정보를 보내는 기술이다. 즉, 푸쉬 기술은 얻고 싶은 정보를 인터넷이 자동적으로 자신의 컴퓨터에 가져다주는 장점을 가지고 있다. 그러나 푸쉬 기술은 수많은 정보 송신과 최신 정보 수신으로 인해 네트웍 대역폭을 너무 많이 소모한다는 단점을 가지고 있다. 이와 같은 정보 전달 방식으로 인해 푸쉬 기능을 제공하는 서버의 요청을을 급격하게 증가시키고 심각한 통신정체를 가져올 수 있으며 쉽게 고장이 발생할 수 있다.

이에 본 논문에서는 메시지 폭주로 인한 시스템 고장에 대처하며 메시지 집중을 분산 시킬 수 있는 복제(replication)에 기반한 분산 푸쉬 서버를 개발하였다. 시스템 고장을 처리 하기 위하여 푸쉬 서버는 분산 복제되어 실행되며 서버간의 일치된 상태를 유지한다. 또한 메시지 집중을 처리하기 위하여 부하 분배기(load balancer)를 사용하여 푸쉬 서버들에게 정보 수신과 송신 메시지를 분산하였다. 푸쉬 서버는 분산 병렬 언어인 Erlang/OTP[1]로 구현되었고, 성능을 분석하기 위해 단일 푸쉬 서버를 사용한 결과와 비교하였다.

2장에서는 관련 연구로 푸쉬 기술에 대하여 설명하고, 3장에서는 본 논문에서 구현된 복제에 기반한 푸쉬 서버에 관하여 기술한다. 4장에서는 실험 결과를 검토하고 5장에서는 본 논문의 결론과 향후 과제를 제시한다.

본 연구는 정보통신부 대학 IT연구센터 육성•지원사업의 연구결과로 수행되었습니다.

2. 관련 연구

기존의 인터넷 환경에서는 사용자들이 스스로 원하는 내용이 있는 곳을 검색하고, 또 원하는 사이트를 직접 찾아가서 정보를 내려 받는 방식이었다. 이것은 특정 장소에서 정보를 꺼내온다는 의미에서 풀(pull) 기술이라고 할 수 있다. 이에 비해서 푸쉬 기술이란 정보를 가지고 있는 서버에서 사용자에게 원하는 정보를 밀어내 준다는 점에서 정 반대의 개념을 가진다. 즉, 푸쉬 기술은 이미 등록이 되어 있는 서버에서 원하는 시간에 원하는 내용을 자신의 컴퓨터로 정기적으로 배달되도록 하는 방식이다[2]. 다음은 기존의 풀 기술과 푸쉬 기술을 비교한 내용이다.

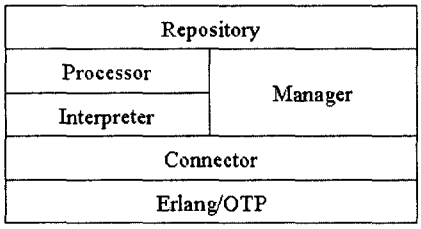
[표 1] 풀과 푸쉬의 비교

항 목	풀 기술	푸쉬 기술
정보 전달	사용자가 서버에서 가져옴	서버에서 사용자에게 정보를 보내줌
공급자	수동적 모형	능동적 모형
사용자 입장	원하는 정보의 위치와 정보의 갱신 시기를 알 수 없어 해당 서버를 직접 방문해야 한다.	요구에 따라 가공되고 갱신된 정보를 서버로부터 자동으로 받는다.
작동 방식	▫ 실시간 방식 ▫ Foreground 방식	▫ Off-line 방식 ▫ Background 방식

그러나 푸쉬 기술은 여러 장점에도 불구하고 다량의 정보 송신과 수신으로 인해 푸쉬 기능을 제공하는 서버의 요청율을 급격하게 증가시켜 심각한 통신 정체를 가져올 수 있으므로 가용성이 떨어질 수 있다는 단점을 가지고 있다.

3. 복제에 기반한 푸쉬 서버

본 논문에서는 통신 정체를 해결하고 서버 고장을 처리하기 위한 푸쉬 서버를 개발하였다. 다음은 개발된 푸쉬 서버의 구조도이다.



[그림 1] 푸쉬 서버 구조도

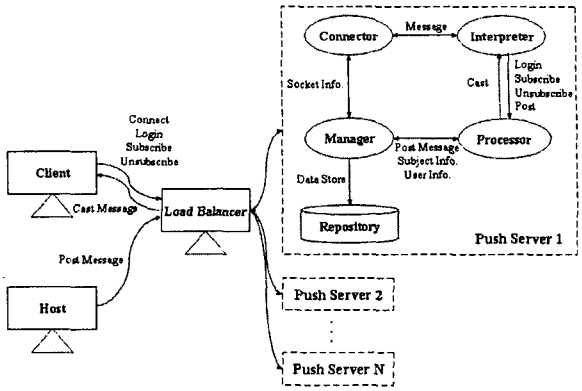
푸쉬 서버는 Erlang/OTP 언어를 사용하여 개발되었다. 이때 프로그램은 OTP 설계 원칙 중 감독 트리 행위(supervision trees behaviour)와 일반 서버 행위(generic server behaviour) 구조를 가지도록 작성하거나 일반 모듈로 작성하였다. 행위는 일반적인 패턴(pattern)을 형식화한 것으로서 일반적인 부분과 특수한 부분으로 프로세스를 위한 코드를 분할하는 방법이다. 연결자(connector)는 TCP 방식으로 클라이언트의 연결과 메시지 송·수신을 담당한다. 해석기(interpreter)는 클라이언트의 메시지 해석을 담당하고 해석된 메시지에 따라 처리기(processor)에게 전달한다. 처리기는 해석기로부터 수신된 메시지에 따라 다음과 같은 연산을 처리한다.

[표 2] 처리기 연산 종류

연산	내용
Login	DB에 클라이언트 정보 등록
Subscribe	DB에 존재하는 항목에 클라이언트를 등록
Unsubscribe	DB에 존재하는 항목에서 클라이언트를 해제
Cast	Post 연산 후 클라이언트로 메시지 전송
Post	최신 정보를 수신하여 메시지 저장

관리기(manager)는 처리기를 통해 수신된 메시지나 연결자를 통해 생성된 클라이언트 연결 정보를 데이터베이스에 저장한다. 저장소(repository)는 Erlang/OTP에서 사용하는 Mnesia DBMS[1]를 사용하고 메시지와 소켓 정보를 저장한다.

다음은 푸쉬 서버가 동작하는 모습을 보여주고 있다.



[그림 2] 푸쉬 서버 동작도

클라이언트와 호스트는 메시지들을 분배하는 부하 분배기를 통하여 현재 연결이 가장 적게 되어 있는 푸쉬 서버에 접속한다. 본 연구에서 부하 분배기는 LVS(Linux Virtual Server)[5]를 사용하고, 포워딩 방식은 DR(Direct Routing)을 사용하며 분배 알고리즘으로는 WLC(weighted least-connection)을 사용하였다. 푸쉬 서버는 다른 푸쉬 서버의 고장을 처리하고 부하를 균등하게 하기 위해 임의의 N개로 복제되어 실행할 수 있다. 연결자는 클라이언트나 호스트가 처음으로 접속할 때마다 새로운 프로세스로 생성되며 클라이언트 정보와 프로세스 ID를 저장소에 저장한다. 이를 통해 다른 푸쉬 서버에서도 연결을 유지하며 메시지를 송신할 수 있게 된다. 해석기는 메시지를 해석하여 처리기에게 전달하고, 처리기는 연산 종류에 따라 관리자를 통해 사용자 정보, 항목 정보, 메시지 등을 저장한다. Cast 연산 시에는 전송할 메시지의 항목에 등록된 사용자들을 찾아 연결된 연결자 프로세스가 존재하면 메시지를 전달하여 푸쉬하도록 한다. 그렇지 않으면 보낼 메시지가 있음을 repository에 저장한다.

클라이언트와 호스트가 푸쉬 서버의 가상 주소로 접속하면 부하 분배기는 푸쉬 서버들 중 하나로 접속을 연결함으로써 복제 투명성을 유지한다. 그리고 로그인하여 접속을 유지하고 있는 도중에 접속된 푸쉬 서버에 고장이 발생한다면 연결이 끊어지게 되고, 마지막에 로그인했던 ID와 비밀번호를 사용하여 자동으로 가상 주소로 재접속을 한다. 이때는 부하 분배기가 다른 푸쉬 서버로 연결 요청을 전달하게 된다. 이와 같이 푸쉬 서버에 고장이 발생하더라도 클라이언트는 끊임없이 메시지를 수신할 수 있다.

푸쉬 서버간의 Repository는 항상 동일한 내용을 가지도록 복제되어 있어 푸쉬 서버간의 일치성을 유지할 수 있다. Mnesia는 여러 노드에 램(ram_copies)이나 디스크(disc_only_copies), 혹은 램과 디스크를 혼용(disc_copies)하여 복제된 테이블들을 유지할 수 있으며, 일련의 테이블 처리 연산이 하나의 원자적 트랜잭션(atomic transaction)으로 그룹화 되어 처리함으로써 일관성을 유지하는 기능을 기본적으로 제공하고 있다[3].

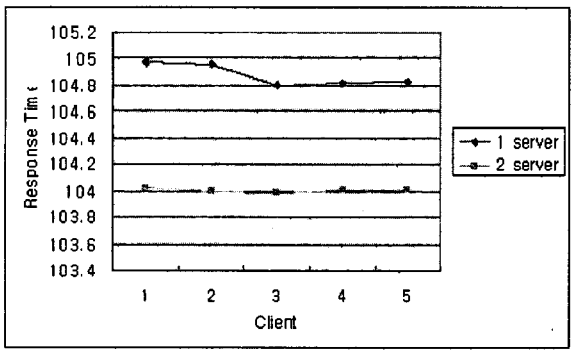
4. 실험 및 성능 분석

본 연구에서 제안된 구조가 메시지 집중을 효율적으로 처리하는지 성능을 분석하기 위하여 단일 푸쉬 서버인 경우와 본 연구에서 제안된 구조에서의 클라이언트의 메시지 응답 시간을 측정하여 비교하였다. 실험에서 사용한 파라미터는 다음과 같다.

[표 3] 실험 파라미터

항목		값
Num. of Push message		1000
Period per message		2 msec
Num. of Process	Client	5
	Host	1
	Load balancer	1
	Push Server	2

본 실험에서는 메시지를 받는 클라이언트들과 메시지를 보내는 호스트를 같은 노드에서 실행하고 분배기와 푸쉬 서버들은 각각 분리된 노드에서 실행하며 응답 시간을 측정하였다. 또한 푸쉬 서버들의 repository는 disc_only_copies 방식으로 두 서버에 복제하였다.



[그림 3] 클라이언트 당 평균 응답 시간 측정

[표 4] 실험 결과

subject / server	average(ms)	maximum(ms)	minimum(ms)
1	104.9	270	9
2	104.0	200	9

[그림 3]과 [표 4]에서 보듯이 두 개의 서버를 사용한 경우의 평균 응답시간이 약간 개선된 결과를 알 수 있다. 성능이 월등히 개선되지 않은 이유는 푸쉬 서버간의 repository 복제로 발생하는 오버헤드 때문으로 분석된다. 하지만 최고 응답 시간을 측정한 결과 두개의 푸쉬 서버를 사용한 경우가 35% 정도의 성능이 개선된 것을 알 수 있다. 이는 분배기를 사용하여 메시지 처리를 분산하고 푸쉬 서버의 과부하를 방지하였기 때문이다. 최소 응답 시간은 두 경우가 모두 비슷한 것을 알 수 있다.

실험 결과로 푸쉬 서버가 복제된 경우 통신 정체를 해결하여 가용성을 증가시킨다는 것을 알 수 있다. 또한 푸쉬 서버의 개수를 증가한다면 최고 응답시간은 더 감소할 것으로 예상되지만, 평균 응답 시간은 repository의 복제로 인한 오버헤드로 오히려 증가할 수도 있다는 점을 알 수 있다.

5. 결론 및 향후과제

푸쉬 기술은 주목받는 인터넷 기술 중 하나임에 틀림없지만, 푸쉬 서버의 요청을을 급격하게 증가시키고 심각한 통신 정체를 가져올 수 있으며 쉽게 고장이 발생할 수 있다는 단점이 있다. 본 논문에서는 이를 해결하고자 부하 분배기와 함께 Erlang/OTP에서 지원하는 Mnesia DBMS를 사용한 복제에 기반한 고가용성 푸쉬 서버를 구축하였다. 그리고 단일 푸쉬 서버와의 성능을 비교한 결과 통신 정체를 효율적으로 처리하여 가용성이 증가하였음을 알 수 있었다.

향후 과제로는 부하 분배기에 푸쉬 서버의 고장을 탐지하는 기능을 추가하여 고장 처리 기능을 향상시키고, repository의 복제로 인해 발생하는 오버헤드를 감소시킬 방법을 개발할 것이다. 또한 안전한 메시지 처리를 위하여 보안 기능을 추가할 예정이다.

6. 참고문헌

[1] Erricson, Open Source Erlang, http://www.erlang.org.

[2] Netscape Communications Corporation, An Exploration of Dynamic Objects, http://home.netscape.com/assist/net_sites/pushpull.html.

[3] H. Mattsson, H. Nilsson, C. Wikström, Mnesia - A Distributed Robust DBMS for Telecommunications Applications, First International Workshop on Practical Aspects of Declarative Languages, January 1999.

[4] R. Baldoni, S. Bonamoneta, C. Marchetti, Implementing Highly-Available WWW Servers Based on Passive Object Replication , Second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, May 1999.

[5] M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, P. Shenoy, Adaptive Push-Pull: Disseminating Dynamic Web Data, IEEE Trans. on Computers, vol. 51, no. 6, page 652-668 , June 2002.

[6] W. Zhang. Linux Virtual Server for Scalable Network Services, Linux Symposium, Ottawa, 2000.