클러스터링 기반의 무선 인터넷 프록시 서버

우재용 곽후근[0] 정윤재 박홍주 김동승* 정규식
숭실대학교 정보통신전자공학부 *고려대학교 공과대학 전기공학과
{tad572, gobarian[0], jgyver, phj0520, kchung}@q.ssu.ac.kr *dkim@classic.korea.ac.kr

A Clustering based Wireless Internet Proxy Server

Jaeyong Woo Hukeun Kwak[0] Yunjae Jung Hongjoo Park Dongseung Kim* Kyusik Chung
School of Electronics Engineering, Soongsil University
*Department of Electrical Engineering, Korea University

요 약

기존 유선 인터넷과 달리 무선 인터넷은 낮은 대역폭, 빈번하게 접속이 끊기는 현상, 단말기내의 낮은 컴퓨팅 파워 및 작은 화면, 사용자의 이동성 등의 특성에 따른 많은 제약점들을 갖고 있다. 또한 무선 인터넷 서버는 급증하는 사용자에 따른 대용량 트래픽을 처리할 수 있도록 확장성이 있어야 한다. 이에 위의 문제를 캐싱(Caching)과 압축(Transcoding, Distillation)으로 해결하는 방법으로 무선 프록시 서버를 사용한다.

TranSend는 클러스터링 기반의 무선 프록시 서버로 제안된 것이나 시스템적인(Systematic) 방법으로 확장성을 보장하지 못하는 단점을 가진다. 이에 본 논문에서는 시스템적인 방법으로 확장성을 보장하는 클러스터링 기반의 무선 인터넷 프록시 서버를 제안한다. 16대의 컴퓨터를 사용하여 실험을 수행하였고 실험 결과 TranSend 시스템에 비해 32.17%의 성능 향상을 보였다.

1. 서론

무선 인터넷의 사용이 증가하고 있는 현실에서 무선 인터넷의 본질적인 문제 역시 무시할 수 없는 요소로 부각되고 있다. 현재까지 나와 있는 무선 인터넷의 근본적인 문제점과 고려해 볼 수 있는 해결책은 다음과 같다.

- 낮은 대역폭 : 압축(Distillation), 캐싱(Caching)
- 빈번하게 연결이 끊기는 현상 : 쿠기(Cookie), 캐싱
- 단말기내의 컴퓨팅 파워 및 작은 화면 : 압축
- 단말기 사용자의 이동성 : Mobile IP

이에 위의 문제를 감소시킬 수 있는 무선 프록시 서버[1-4]가 필요로 되고 있다. 기본적으로 무선 프록시는 캐싱(Caching), 압축(Distillation), 대용량 트래픽에 대한 확장성(Scalability)를 고려하여야 한다.

본 논문에서는 무선 인터넷 상의 대용량 트래픽에 대한 확장성을 고려한 TranSend 무선 인터넷 프록시[2]의 문제점을 분석하고 이를 해결할 새로운 구조를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 클러스터링 기반의 무선 프록시 서버인 TranSend 시스템 및 이 시스템이 가지는 문제점을 소개한다. 3장에서는 TranSend가 가지는 문제점을 해결할 새로운 구조를 설명하고, 4장에서는 제안된 방법의 실험 및 토론을, 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 연구 배경

2.1 클러스터링 무선 프록시 서버[2]

그림 1은 TranSend 프록시 시스템의 전체적인 구조를 나타낸다.

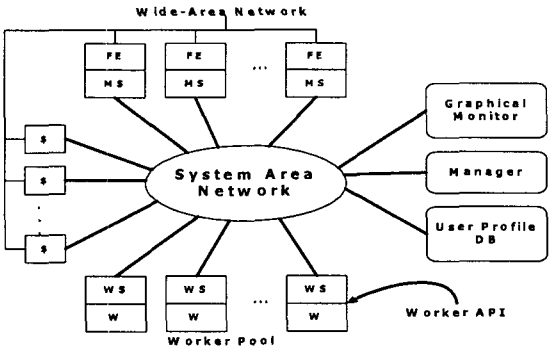


그림 1 TranSend 프록시 시스템 구조

각 모듈로써 Front End(FE, MS:Manager Stub), User Profile DB, Cache($), Worker(W, Worker API, WS:Worker Stub), Manager, Graphical Monitor가 구성된다. FE는 Client 요청에 대한 외부 인터페이스를 담당하며, User Profile DB는 사용자와 관련된 정보(Preference)를 저장한다. Cache는 Client의 요청을 처리하며, Worker(Datatype-Specific Distiller)는 데이터에 대한 압축을 수행한다. Manager는 Distiller를 관리하고, Graphical Monitor는 시스템 전체의 상태를 볼 수 있게 해준다.

Client 요청을 FE가 받고 Cache 서버에 요청하여 존재하면 해당 데이터를 받고, 존재하지 않으면 Cache 서버가 웹 서버로부터 요청하여 받아온다. FE는 그 데이터를 Worker(Distiller)에

게 보내 압축을 요청하며 압축된 데이터를 Client에게 보내는 방법으로 동작한다.

2.2 TranSend 무선 프록시의 문제점

(1) 확장성 (Scalability)

그림 1 구조에서 보면 FE, Cache, Distiller는 각각 여러개의 노드(Node) 들로 구성가능하다. 프록시 서버를 확장성있게 만들려면 노드들을 추가해야하지만 어느 분류(FE 그룹, Cache 그룹, Distiller 그룹)의 노드들을 언제 추가해야 하는지 시스템적인(Systematic) 방법이 없다. 즉, 실험 결과에 의존하여 특정 모듈 그룹이 병목 현상이 발생하면 그룹 모듈을 추가하는 방식으로 하게 된다.

(2) 부하 분산 (Load Balancing)

FE의 경우 다수개의 FE를 위한 부하 분산 방식이 고려되어 있지 않고, Cache의 경우 MD5 Hash를 사용하나 동일 주소로 요청이 몰릴 경우 부하 분산이 제대로 되지 않는 문제점이 존재한다.

(3) 오류 복구 (Fault Tolerance)

TranSend는 Manager(FE가 복구)와 Distiller(Manager가 복구)를 제외하고 FE와 Cache는 오류 복구가 안되는 문제점을 가지고 있다.

3. 제안된 프록시 서버

그림 2는 2.2절에서 분석된 TranSend 무선 프록시의 문제점을 기반으로 이를 해결할 수 있도록 제안된 구조이다. 제안된 구조는 TranSend에 사용된 모듈들을 모두 하나의 호스트에 넣고(이하 All-in-one) 이 호스트들을 LVS(Linux Virtual Server)[5]를 사용하여 부하 분산을 하는 것이다. TranSend에서는 각각의 모듈들(FE, Cache, Distiller) 각각이 클러스터링 되어 있는 반면에 본 논문에서는 각 모듈들을 하나의 호스트들에 포함하고 이러한 호스트를 클러스터링하는 구조로 되어 있다.

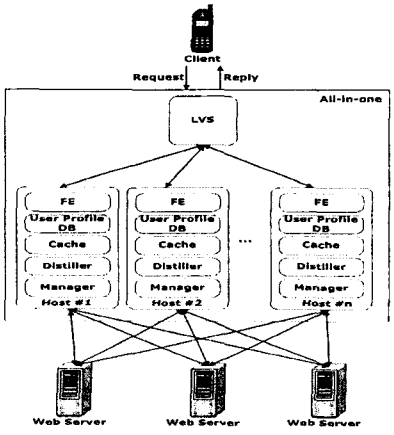


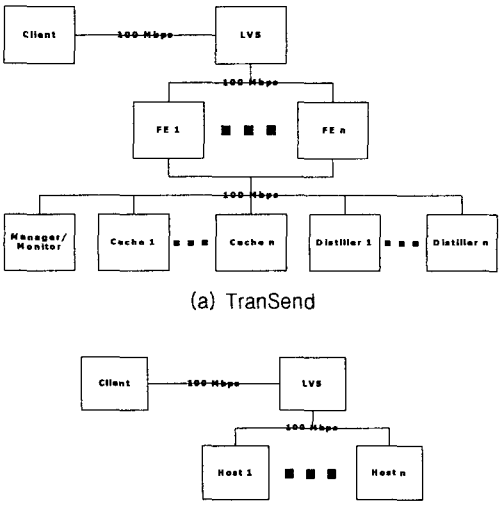
그림 2 제안된 구조(All-in-one)

4. 실험 및 토론

4.1 실험 환경

실험에는 Client 1대, LVS 1대, 동일 사양의 Host 16대로 구성되어 있고 TranSend 시스템에서 FE의 부하 분산과 제안된 시스템의 부하 분산을 위하여 LVS라는 Load Balancer를 사용하였다. Client는 AB(Apache Bench), Cache는 Squid, Distiller는 JPEG-6b 라이브러리를 사용하였다. 클라이언트의 요청 개수는 약 200초 동안 프록시를 처리할 수 있는 최대 개수이고, 요청 이미지는 JPEG, 요청 크기는 300 bytes, 1 K, 10 K, 100 Kbytes, Variation(1~10 Kbytes)[6]이다.

그림 3는 실험에 사용된 TranSend와 All-in-one의 구성도를 나타낸다. 그림 3(a)는 All-in-one과 TranSend를 확장성 측면에서 비교하기위해 TranSend 기본 구조에 FE를 클러스터링하기위해 LVS를 추가하였다.



(a) TranSend

(b) All-in-one

그림 3 구성도

4.2 실험 결과

그림 4는 TranSend에서 이미지 크기를 달리 요청했을 때 호스트 개수에 따른 초당 요청수를 나타낸다.

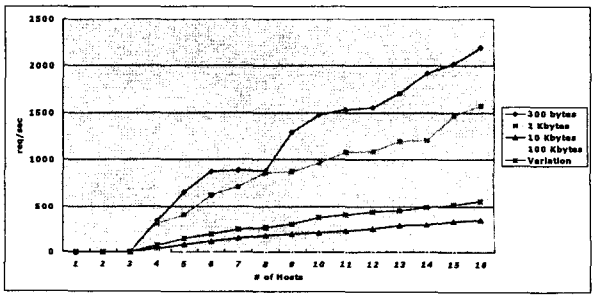


그림 4 호스트 개수에 따른 초당 요청수 (TranSend)

그림 5는 All-in-one에서 이미지 크기를 달리 요청했을 경우 호스트 개수에 따른 초당 요청수를 나타낸다.

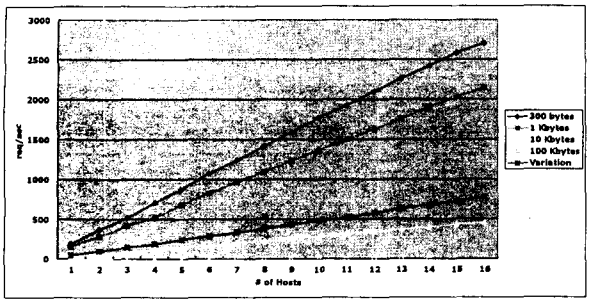


그림 5 호스트 개수에 따른 초당 요청수 (All-in-one)

표 1은 16대의 호스트를 기준으로 TranSend에 대한 All-in-one의 성능 향상률을 나타낸다. 평균 향상률(TranSend 1 기준)은 32.17%이고, Distiller의 추가(spawn) 시간을 고려한 평균 향상률(TranSend 2 기준)은 26.55%이다. Distiller의 추가 시간을 고려한 것은 TranSend의 구조로 인해 모든 Distiller가 동작하려면 어느 정도의 시간이 소요됨으로 이러한 상황을 고려하여 사용자 요청 시간(약 200초)을 2배로 하여 실험을 수행한 것이다.

표 1 All-in-one 성능 향상률 (16개의 호스트 기준)

	300 bytes	1 Kbytes	10 Kbytes	100 Kbytes	Variation	Avg.
TranSend 1	2190.27	1575.90	350.29	26.10	553.61	
TranSend 2	2217.78	1634.77	360.86	28.50	581.98	
All-in-one	2699.92	2148.53	458.43	34.08	773.90	
Improve 1	23.27	36.34	30.87	30.58	39.79	32.17
Improve 2	21.74	31.43	27.04	19.58	32.98	26.55

4.3 토론

제안된 시스템이 기존 시스템에 비해 성능이 좋아진 이유는 구조적 관점에서 호스트들의 CPU 활용도로 설명할 수 있다. 예를 들면, TranSend 시스템은 호스트의 개수가 4개일 때 FE를 제외하고 나머지 모듈들의 CPU 활용도가 낮다 (Manager /Monitor = 7.3%, FE1 = 100%, Cache1 = 39.8%, Distiller1 = 36.7%). 반면 제안된 시스템은 모든 모듈이 하나의 호스트에 들어있고, 요청된 이미지의 크기에 따라 필요한 모듈의 CPU 점유율이 상대적으로 높게 배정되는 방식으로 운영된다. 그래서 항상 CPU 점유율이 100%가 됨으로 구조적 관점에서 CPU 활용도가 높다.

제안된 구조의 단점은 다음과 같다.

- LVS : 제안된 구조는 FE와 Cache의 오류 복구 문제를 해결하였으나 이의 해결을 위해 새롭게 추가된 LVS가 새로운 단일장애점(a single point of failure)이 되는 문제를 가지고 있다. 이의 해결을 위해서는 백업(backup) LVS를 두어 오류 복구를 하거나 L4/L7 switch를 사용하는 방법이 있다.
- Cache : 제안된 구조는 Cache간의 협동(Cooperation)이 없어 다른 호스트에 사용자 요청과 동일한 데이터가 Cache되어 있어도 매번 실제 웹 서버로 요청하고 이를 자신의 로컬 Cache에 두어서 Cache된 데이터의 합이 커지는 문제가 발생한다. 이의 문제를 해결하기 위해서는 Cache 라우팅 테이블(Cache Routing Table)이나 CARP(Cache Array Routing Protocol)[7]의 적용을 고려해 볼 수 있다.

5. 결론

본 논문에서는 무선 인터넷의 근본적인 문제점 중 일부를 해결 할 수 있도록 제안된 TranSend 프록시 서버의 문제점을 지적하고, 구조 및 성능 개선을 제안하였다. TranSend 프록시 서버의 문제점을 확장성, 부하 분산, 오류 복구 관점에서 분석하였고, 이를 해결하기 위해 새로운 구조를 제안하였다. 제안된 구조를 실험을 통해 성능 향상에 기여했음을 확인하였다.

향후 연구 방향을 요약하면 다음과 같다.

- 대용량 이미지나 동영상에 대한 압축기(Distiller)들의 수행 시간 단축
- 클라이언트의 환경을 고려한 QoS(Quality of Service)

참고문헌

[1] B. Housel, G. Samaras and D. Lindquist, "WebExpress: A client/intercept based system for optimizing web browsing in a wireless environment", Mobile Networks and Applications, ACM, pp. 419-431, 1998.

[2] A. Fox, "A Framework For Separating Server Scalability and Availability From Internet Application Functionality", Ph. D. dissertation, U. C. Berkeley, 1998.

[3] K. Kim, H. Lee and K. Chung, "A Distributed Proxy Server System for Mobile Web Service", Proceedings of the 15th International Conference on Information Networking, IEEE, pp. 8A 749-754, 2001

[4] A. Maheshwari, A. Sharma, K. Ramamritham and P. Shenoy, "TranSquid: transcoding and caching proxy for heterrogeneous e-commerce environments", Proceedings of 12th International Workshop on RIDE-2EC, IEEE, pp. 50-59, 2002

[5] Linux Vitual Server, http://www.linuxvirtualserver.org

[6] S. Chandra, A. Gehani, C. Ellis and A. Vahdat, "Transcoding Characteristics of Web Images", Proceedings of the SPIE Multimedia Computing and Networking Conference, 2001.

[7] V. Valloppillil and K. Ross, "Cache array routing protocol v1.0", 1998