클러스터링 기반의 무선 인터넷 프록시 서버 성능 개선

곽후근° 한경식 정규식
숭실대학교 정보통신전자공학부
{gobarian°, ihanks, kchung}@q.ssu.ac.kr

The Performance Improvement of a Clustering based Wireless Internet Proxy Server

Hukeun Kwak° Kyungsik Han Kyusik Chung
School of Electronics Engineering, Soongsil University

요 약

TranSend는 클러스터링 기반의 무선 프록시 서버로 제안된 것이나 시스템적인(Systematic) 방법으로 확장성을 보장하지 못하고 불필요한 모듈간의 통신구조로 인해 복잡하다는 단점을 가진다. 기존 연구에서 시스템적인 방법으로 확장성을 보장하는 All-in-one 이라는 구조를 제안하였으나 이 역시 모듈간의 통신 구조가 복잡하다는 단점을 가진다. 이에 본 논문에서는 시스템적으로 확장성을 보장하고 모듈간의 단순한 통신 구조를 가지는 클러스터링 기반의 무선 인터넷 프록시 서버를 제안한다. 16대의 컴퓨터를 사용하여 실험을 수행하였고 실험 결과 TranSend 시스템과 All-in-one 시스템에 비해 각각 105.69%, 39.93%의 성능 향상을 보였다.

1. 서 론

무선 인터넷의 사용이 증가하고 있는 현실에서 무선인터넷의 본질적인 문제 역시 무시할 수 없는 요소로 부각되고 있다. 현재까지 나와 있는 무선 인터넷의 근본적인 문제점과 고려해 볼 수 있는 해결책은 표 1과 같다.

표 1 무선 인터넷의 문제점과 가능한 해결책

문제점	가능한 해결책
낮은 대역폭	압축(Distillation), 캐싱(Caching)
빈번하게 연결이 끊기는 현상	쿠키, 캐싱
단말기내의 낮은 컴퓨팅 파워 및 작은 화면	압축
단말기 사용자의 이동성	Mobile IP, TCP Migration
네트워크 프로토콜	Wireless TCP
보안	프록시 서버

이에 위의 문제를 캐싱과 압축으로 해결하는 방법으로 무선 프록시를 사용한다. 기본적으로 무선 프록시는 캐싱, 압축, 대용량 트래픽에 대한 확장성을 고려하여야 한다.

기존 무선 프록시들[1-3]은 캐싱 및 압축 기능을 제공하고 있다. 확장성 관점에서는 일부[2]는 확장성을 고려하고 있지만 대부분[1, 3]은 고려하고 있지 않다. 구조 관점에서는 대부분[1, 3]은 캐싱과 압축 기능이 하나의 호스트에 통합되어 있는 반면에 일부[2]는 다른 호스트로 분리되어 있다.

1.1. TranSend[2]

그림 1은 TranSend 프록시 시스템의 전체적인 구조를 나타낸다. 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)에게 보내 압축을 요청하며 압축된 데이터를 Client에게 보내는 방법으로 동작한다.

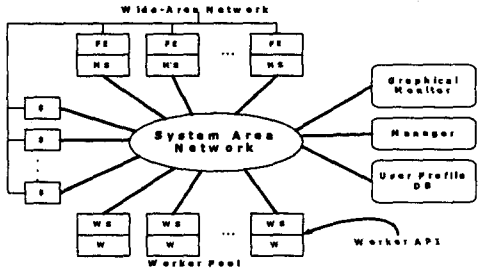


그림 1 TranSend 프록시 시스템 구조

1.2 All-in-one[4]

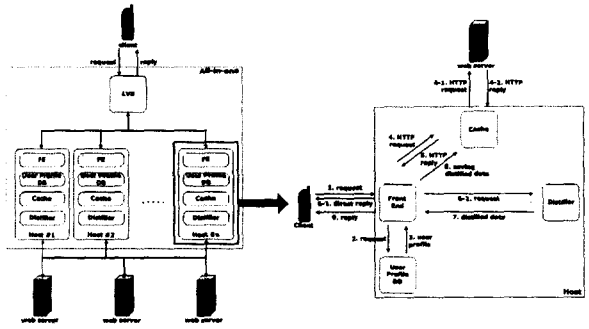


그림 2 All-in-one

All-in-one은 TranSend에 사용된 모듈들을 모두 하나의 호스트에 넣고(이하 All-in-one) LVS(Linux Virtual Server)[5]를 사용하여 호스트들의 부하 분산을 하는 것이고 그림 2는 이를

나타낸다. TranSend에서는 모듈들(FE, Cache, Distiller) 각각이 클러스터링 되어 있는 반면에 All-in-one에서는 각 모듈들을 하나의 호스트에 포함하고 이러한 호스트를 클러스터링하는 구조로 되어 있다.

1.3 접근 방식

TranSend 무선 프록시 서버 및 이의 개선 구조인 All-in-one의 문제점을 정리하면 표 2와 같다.

표 2 기존 구조의 문제점

기존 구조	문제점
TranSend	• 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 해당 병목 모듈을 추가하는 방식으로 하게 된다. • 복잡성(Complexity) : TranSend는 FE, Cache, Distiller로 구성되어 FE를 중심으로 서로간에 통신(Communication)을 하도록 구성되어 있다. 이러한 구조는 FE로 모든 통신이 편중되어 있고 Cache와 Distiller가 분리되어 있어 불필요한 통신을 하는 단점을 가진다.
All-in-one	• 복잡성(Complexity) : All-in-one 구조는 TranSend를 구성하는 모듈을 하나의 호스트에 넣은 구조임으로 TranSend와 마찬가지로 모듈간의 통신이 복잡하다는 단점을 가진다.

본 논문에서는 TranSend와 All-in-one 구조의 단점인 복잡성(Complexity)을 단순화(Simplification)한 새로운 구조를 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 기존 무선 프록시가 가지는 문제점을 해결하는 새로운 구조를 설명하고, 3장에서는 실험 및 토론을, 4장에서는 결론 및 향후 연구 방향을 제시한다.

2. 제안된 프록시 서버(CD-A)

그림 3은 1.3절에서 분석된 TranSend 및 All-in-one 무선 프록시의 문제점을 기반으로 이를 해결할 수 있도록 제안된 구조이다. 제안된 구조는 TranSend에 사용된 기본 모듈에서 Distiller를 없애고 Cache에 압축(Distillation) 기능을 추가(이하 CD: Cache & Distiller)한 것이다. 그리고 이 모듈들(FE, Cache)을 하나의 호스트에 넣고 LVS를 사용하여 부하 분산을 한 것(CD-A: CD & All-in-one)이다. 제안된 구조에서 Cache에 압축 기능을 추가한 이유는 전체 구조에서 Distiller를 제거하여 불필요하고 복잡한 통신 구조를 단순화하기 위해서이다. 그리고 모듈들을 하나의 호스트에 넣은 이유는 시스템적으로 확장하는 구조를 만들기 위해서이다. 즉, TranSend는 새로운 모듈을 추가시에 동작과정중의 병목을 찾아 그 모듈을 추가해야하는(No Systematic) 반면에, CD는 병목에 상관없이 새로운 호스트를 추가하면(Systematic) 그 호스트내에 모듈중에 필요한 모듈이 상대적으로 많이 사용된다.

제안된 구조의 동작원리는 다음과 같다.

• 사용자(Client)가 제안된 프록시 구조(CD-A)에 데이터를 요청한다.

• 요청을 받은 LVS가 라운드 로빈(Round Robin) 방식으로 임의의 호스트를 선택한다.

• 선택된 호스트는 실제 웹 서버에 데이터를 요청한다.

• 호스트는 웹 서버로부터 받은 데이터를 LVS를 통해 사용자에게 전송한다.

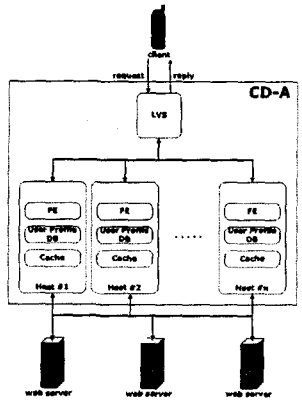
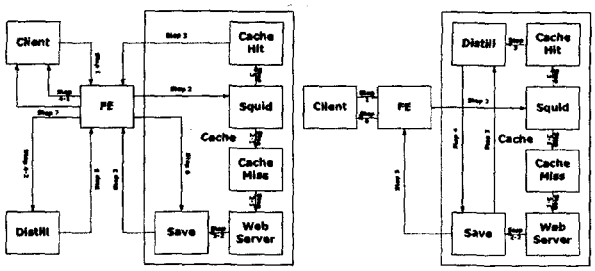


그림 3 제안된 구조(CD-A)

그림 4는 Cache 관점에서 TranSend와 제안된 구조의 사용자 요청 처리 순서를 나타낸 것이다. All-in-one은 TranSend 모듈을 한 호스트에 집어넣은 것으로 처리 순서는 TranSend와 거의 동일하다.



(a) TranSend (b) CD-A

그림 4 Client 요청 처리 순서

표 3은 TranSend와 All-in-one을 제안된 시스템(CD-A)과 구조적으로 비교한 표이다.

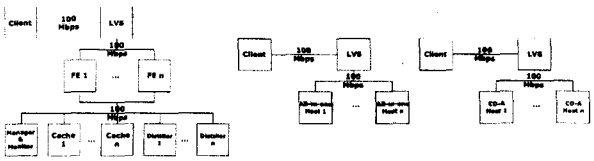
표 3 기존 구조 vs 제안된 구조(CD-A)

	TranSend	All-in-one	CD-A
Scalability	No Systematic	Systematic	Systematic
Simplification	X	X	O

3. 실험 및 토론

실험 환경은 Client 1대, LVS 1대, 동일 사양의 Host 16대로 구성되어 있고 TranSend 시스템에서 FE의 부하 분산, All-in-one과 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)이다.

그림 5는 실험에 사용된 구조들의 구성도를 나타낸다.

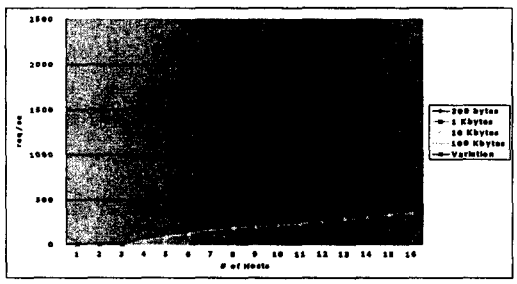


(a) TranSend (b) All-in-one (c) CD-A

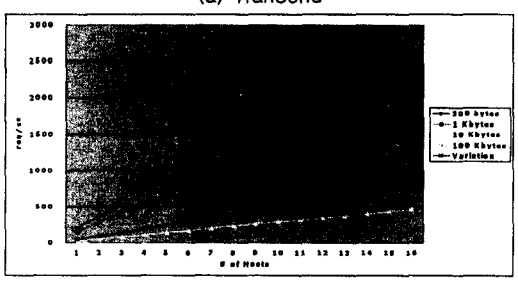
그림 5 실험에 사용된 구성도

3.1 각 구조의 초당 요청수

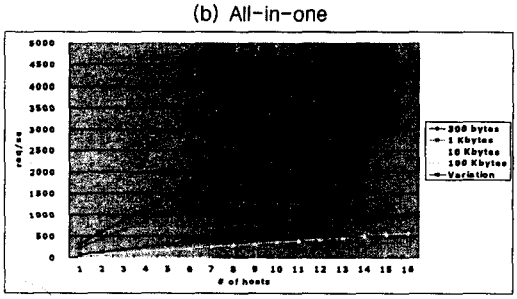
그림 6은 각 구조의 초당 요청수를 나타낸다.



(a) TranSend



(b) All-in-one



(c) CD-A

그림 6 각 구조의 초당 요청수

3.2 TranSend, All-in-one vs. CD-A

표 4은 TranSend와 All-in-one에 대한 CD-A의 평균 성능 향상률을 나타낸 것이다. 제안된 시스템(CD-A)은 TranSend와 All-in-one에 비해 각각 평균 105.69%와 39.93%의 성능 향상을 가진다. 작은 크기의 이미지는 큰 크기의 이미지보다 상대적으로 모듈간의 통신에서 많은 시간이 소요됨으로 제안된 시스템에서 높은 성능 향상률을 가짐을 알 수 있다.

표 4 평균 성능 향상률 (CD-A)

%	300 bytes	1 Kbytes	10 Kbytes	100 Kbytes	Vari.	Avg.
TranSend vs. CD-A	125.02	130.84	88.76	87.30	96.53	105.69
All-in-one vs. CD-A	64.51	58.96	27.30	16.59	32.29	39.93

3.3 토론

제안된 시스템이 기존 시스템에 비해 성능이 좋아진 이유는 구조적 관점에서 복잡한 구조를 단순한 구조로 바꾼 결과로 설명할 수 있다. 단순화 과정은 다음 2가지로 요약할 수 있다.

• 단순화 1 : Cache에서 FE, FE에서 Distiller로 통신하는 구조를 Cache와 Distiller가 직접 통신하는 구조로 단순화

• 단순화 2 : Cache와 Distiller가 소켓 통신을 이용하지 않고 Cache에서 압축을 직접 수행하는 구조로 단순화

위의 단순화된 구조는 모듈(FE, Cache, Distiller)간의 통신이 상대적으로 빈번한 작은 크기의 이미지 요청 실험에서 더 좋은 성능을 보인다.

제안된 구조의 단점은 다음과 같다.

• 캐시간 협동 : 제안된 구조는 캐시간 협동(Cooperation)이 없다. 그래서 다른 호스트에 사용자 요청과 동일한 데이터가 캐시되어 있어도 매번 실제 웹 서버로 요청하고 이를 자신의 로컬 캐시에 두어서 캐시된 데이터의 합이 커지는 문제가 발생한다. 이의 문제를 해결하기 위해서는 LVS에서 호스트로 부하 분산시에 Destination Hashing Scheduling을 적용하면 된다.

4. 결론

본 논문에서는 무선 인터넷의 근본적인 문제점 중 일부를 해결 할 수 있도록 제안된 TranSend 및 이의 개선된 구조인 All-in-one 프록시 서버의 문제점을 복잡성 관점에서 지적하고, 구조 및 성능 개선을 제안하였다. 그리고 실험을 통해 제안된 구조가 성능 향상에 기여했음을 확인하였다.

향후 연구 방향을 요약하면 다음과 같다.

• 부하 분산 시에 라운드 로빈 대신에 호스트의 처리 능력에 따른 부하 분산 방식으로 변환 : 부하 분산에서 라운드 로빈을 적용할 경우 부하가 동일하게 분산됨으로 최종 결과는 처리 능력이 가장 적은 호스트에 종속되는 단점을 가진다.

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

[4] 우재용, 곽후근, 정윤재, 박홍주, 김동승, 정규식, "클러스터링 기반의 무선 인터넷 프록시 서버", 한국정보과학회 가을 학술발표논문집, Vol. 30, No. 2, pp. 76-78, 2003.

[5] LVS, http://www.linuxvirtualserver.org