2001 년 한국정보처리학회 춘계 학술발표논문집 제 8 권 제 1 호

프락시 서버 관리를 위한 확장된 모니터링 도구

김세주*, 이금석*
*동국대학교 컴퓨터공학과
e-mail : starwars@dgu.ac.kr
e-mail : kslee@dgu.ac.kr

An extended monitoring tool for proxy server management

Se-ju kim*, keum-suk lee*
*Dept. of Computer Engineering, dong-guk University

요 약

현재의 인터넷 사용자들은 하루가 다르게 증가하고 있는 실정이다. 사용자들의 증가로 인하여 많은 네트워크의 부하가 걸리게 되는데 이는 결국 네트워크 속도 감소로 이어지고 있다. 따라서 네트워크의 속도 문제를 해결하기 위한 방법으로 많은 연구가 진행 되어 왔는데 그 중의 하나가 웹 캐쉬 방법중의 하나인 프락시(proxy) 서버를 이용하는 방법이다. 이러한 프락시 서버 관리를 위한 모니터링 도구들이 많이 상용화 되었고 기존의 프락시 서버를 관리하는 모니터링 연구에서는 프락시 서버의 캐쉬 히트율(hit ratio)이나 바이트 히트율을 높이는 방안을 제시하고자 했다. 그러나 이러한 모니터링으로는 프락시 서버의 부하(load)가 얼마나 걸리는지를 파악 할 수가 없다. 따라서 본 논문에서는 기존의 프락시 서버 모니터링 방식과 더불어 부하의 정보를 제공함으로서 부하 균등 정책이나 사용자의 성향을 파악하여 프락시 서버를 좀 더 효율적으로 관리하고자 하는 방안을 제시하고자 한다.

1. 서론

현재의 웹 환경은 하루가 다르게 발전하고 있다. 인터넷을 이용하는 사용자들의 숫자도 하루가 다르게 증가하고 있으며 이러한 환경으로 인하여 기존의 웹 환경에서 미래의 웹 환경을 예측한다는 것 조차 거의 힘들게 되었다. 현재 인터넷을 사용하는 사용자들은 급속하게 늘어나는데 반해 기존의 네트워크 방법들은 아직도 현재의 웹 사용자들의 서비스 욕구를 만족시켜 줄 만큼의 네트워크 환경을 구축하고 있지 못한 실정이다.

인터넷을 이용하는 사용자들은 한 사이트에 방문할 때 마다 6 초안에 어떤 페이지의 변화가 없으면 바로 다른 사이트로 이동하는 경향이 있는데,[1] 이러한 사용자들의 서비스 요구를 충족시켜 주지 못하게 되는 사이트는 곧 소비자의 외면을 받게 된다.

이러한 사용자들의 서비스 요구를 충족 시켜주기 위하여 네트워크 트래픽(traffic)을 감소시키는 여러 가지 방법들이 나오게 되었는데 그 중의 한 방법이 바로 웹 캐쉬(cache)를 이용한 방법이다. 이러한 웹 캐싱(caching)으로는 크게 두 가지로 나눌 수가 있는데 브라우저를 이용한 캐쉬 방법과 프락시 서버를 이용한 캐쉬 방법이 있다.[2] 이러한 방법들 중 프락시 서버를 이용한 캐쉬 방법에는 클라언트 쪽에 위치하는 프락시 서버와 서버쪽에 위치하는 프락시 서버를 이용하는 캐쉬방법으로 나누어 진다.

웹 캐쉬의 한 방법인 프락시 서버를 이용함으로써 얻을 수 있는 장점으로는 사용자들의 지연 시간을 줄일 수 있고 네트워크의 트래픽을 감소 시킬 수 있으며 사용자들의 서비스 요청에 대한 응답시간을 줄 일

수 있다.[3] 이러한 네트워크의 트래픽을 줄일 수 있는 프락시 서버를 이용한 많은 사이트들도 등장하였으며 프락시 서버를 구축한 회사나 학교도 많이 등장하였다. 이렇게 구축된 프락시 서버를 제대로 이용하기 위해서는 프락시 서버를 관리하기 위한 프락시 서버 모니터링 도구가 필요하며 상용화된 프락시 서버 모니터링 제품들도 많이 출시되었다.[4][5] 이러한 모니터링에는 기본적으로 프락시 서버에 캐쉬된 웹 페이지들에 대한 히트율 등을 서버 관리자에게 보여주며 기타 부수적으로는 자주 다운로드 받는 파일들 , 사용자들이 자주 요청하는 웹 사이트 및 웹 페이지 , 사용자들이 자주 접속하는 시간이나 요일 , 웹 서비스를 자주 요청하는 사용자 등의 부수적인 모니터링 결과를 서버 관리자에게 보여준다.[6] 본 논문에서는 기존의 프락시 서버 모니터링 방식에 작업부하를 분석하는 기능을 추가함으로써 프락시 서버들간의 각 서버별 부하 분석 및 부하 균등을 위한 정보와 사용자들의 행태 분석에 대한 정보를 제공하는 모니터링을 구현 한다.

본 논문의 구성은 다음과 같다. 2 절에서는 본 논문과 관련된 기존 연구를 살펴보고, 3 절에서는 부하 분석을 위한 모니터링 도구 설계 및 구현 방법을 살펴보고, 4 장에서는 결론 및 향후연구에 관해 살펴보도록 하겠다.

2. 관련 연구

웹 캐쉬의 한 방법으로 사용되어 지는 프락시 서버의 기능은 네트워크의 트래픽을 줄여 웹 서버로 어떠한 서비스를 요청한 클라이언트들에게 보다 빠른 서비스 응답시간을 제공하기 위한 방법으로 고안이 되었다. 이런 프락시 서버는 두 가지 방식으로 나뉘게 되는데 클라이언트쪽에 있는 프락시 서버와 서버쪽에 있는 프락시 서버 둘로 나뉜다.

클라이언트쪽에 위치한 프락시 서버는 클라이언트들이 자주 접속하는 페이지들의 내용들을 프락시 서버에 미리 저장해 놓는다. 이럴 경우 클라이언트들은 직접 웹 서버에게 어떠한 자료를 요청하지 않고 클라이언트쪽에 위치한 프락시 서버로 자료를 요청하게 된다. 만일 프락시 서버에 자료가 담겨 있다면 캐쉬 히트가 일어나게 되며 그렇지 않다면 캐쉬 히트가 일어나지 않고 바로 웹 서버로 어떠한 자료를 요청하게 된다. 그러므로 캐쉬의 히트율이 높을수록 프락시의 서버 성능은 높아지게 되고 네트워크의 트래픽도 감소하게 되며 이와 더불어 클라이언트들이 요청한 자료를 빠르게 받아 볼 수가 있다.

서버쪽에 위치한 프락시 서버도 클라이언트가 자주 요청하는 페이지들의 내용들을 프락시 서버에 미리 저장해 놓는다. 이럴 경우 클라이언트가 요청한 일들을 직접 웹 서버가 처리하지 않고 프락시 서버가 처리하게 된다. 클라이언트들이 자주 요청한 자료들을 서버쪽에 위치한 프락시 서버에 저장해 둠으로써 요청한 자료는 웹 서버를 거치지 않고 프락시 서버에서 클라이언트로 보내준다. 이럴 경우 웹 서버의 서비스 응답 시간을 단축 시키는 장점과 웹 서버의 부하를 줄여주는 효과를 가져온다.[2] 그림 1)은 웹 캐쉬에 대한 일반적인 구성도다.[7] 웹 캐쉬는 다음의 순서대로 처리된다. 브라우저에 담겨있는 쿠키 정보에 찾고자 하는 자료가 없을 경우 클라이언트에 있는 프락시 서버에 자료를 요청한다. 이럴 경우 프락시 서버에 자료가 있다면 클라이언트에게 바로 전달되고 자료가 없을 경우에는 웹 서버로 자료를 요청하게 된다. 웹 서버 앞에 있는 프락시 서버에 요청한 자료가 있다면 클라이언트에 자료가 전달되고 프락시 서버에 없다면 웹 서버에서 자료를 가져오게 된다.

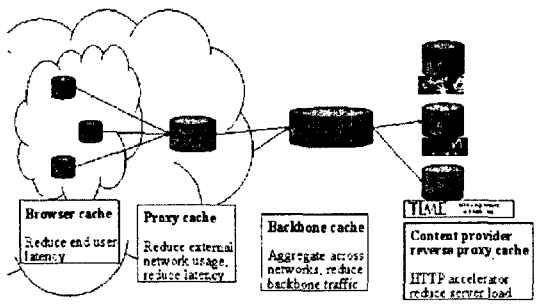

그림 1) 웹 캐쉬의 구성도

현재 프락시 서버에 걸린 부하가 얼마인지를 보여주는 모니터링 툴은 존재하지 않는다. 프락시 서버도 하나의 서버라는 관점에서 본다면 부하가 얼마나 걸리는지를 보는 것도 프락시 서버 관리에 있어서 매우 중요하다. 본 논문에서는 보다 효율적인 프락시 서버 관리를 위해 기존의 모니터링에 부하 분석까지 추가함으로써 좀 더 효율적인 프락시 서버 관리를 위한 모니터링을 만들고자 한다.

3. 확장된 모니터링 도구 설계 및 구현 환경

3.1 개요

현재 상용화되어 있는 모니터링 도구들은 프락시 서버를 이용하는 사용자들의 웹 사용 행태는 파악할 수 있지만 프락시 서버의 부하와 관련해서는 어떠한 내용도 알 수가 없다. 프락시 서버도 결국은 웹 서버와 같은 서버의 개념이므로 각 서버마다 작업 부하량을 모니터링 함으로써 프락시 서버 관리를 보다 효율적으로 다룰 수 있다. 본 논문에서는 기존의 프락시 서버 모니터링 도구에서 보여지는 기본적인 모니터링 정보와 더불어 부하 분석 정보를 보여줌으로써 프락시 서버 관리를 하는데 있어 보다 효율적으로 대처할 수 있다.

3.2 부하 분석을 추가한 배경

기존의 프락시 서버 관리 모니터링과 비교하여 볼

때 프락시 서버의 부하를 분석함으로써 각 프락시 서버간의 부하 균등이 제대로 이루어 지는지를 볼 수가 있다. 만일 한 쪽의 서버에만 부하가 집중된다면 기존의 프락시 서버에서 이루어지는 부하 균등 정책은 잘못 되어 졌다고 볼 수 있다. 모니터링 정보를 이용하면 차후에 사용자들의 웹 사용 경향에 맞게 부하 균등 정책을 바꿀 수 있도록 정책 방향 수립을 도와 줄 수 있다. 이는 인터넷을 이용하는 사용자들의 경향이 일정한 것이 아니라 사용자 집단에 따라 인터넷을 이용하는 경향이 제 각각 다르기 때문이다. 예를 들어 대학생들은 주로 연구 목적이나 학문을 위한 정보를 찾는 반면 회사원들은 주로 증권정보나 재테크 정보 등에 더 많은 인터넷 사용을 하게 된다. 그러므로 이러한 사용자들의 경향에 맞게 프락시 서버 정책도 바뀌어야만 한다.

또한 기존의 웹 서버나 프락시 서버에서의 부하는 주로 request 의 수를 부하의 주 요소(factor)로 잡았지만 본 논문에서는 request 수와 프락시 서버의 디스크에 캐쉬된 파일의 바이트를 부하의 주 요소로 보고자 한다. 표 1)은 부하를 분석한 예를 나타낸 것이다.

	서버 1	서버 2	서버 3
전체 Request 수	500	200	300
요청한 파일들이 디스크에 캐쉬된 전체 크기	15 MB	100 MB	30MB
요청한 파일들의 개별 request 수	HTML : 400 IMAGE : 90 멀티미디어: 10	HTML: 100 IMAGE: 50 멀티미디어: 50	HTML : 200 IMAGE : 80 멀티미디어: 20
요청한 파일들의 디스크에 캐쉬된 개별 크기	HTML: 400 KB IMAGE: 3MB 멀티미디어: 11.6M	HTML: 100KB IMAGE: 5MB 멀티미디어: 94.9M	HTML : 200 KB IMAGE: 10MB 멀티미디어:19.8M

표 1)부하를 분석한 표

위의 표에서 보면 1 번 서버에서와 2 번 서버를 비교해 봤을 때 request 수는 1 번 서버가 약 2.5 배 많다. 하지만 실제 데이터를 요청한 파일 크기는 오히려 2 번 서버가 더 많음을 알 수 있다. 이럴 경우 request 수와 상관없이 실제 프락시 서버에서는 오히려 2 번 서버가 더 많은 부하가 걸릴 수가 있다. 따라서 서버의 부하를 요청한 자료 중 디스크에 담긴 파일의 크기로 봤을 경우에 좀 더 정확한 부하라고 볼 수 있다.

3.3 구현 환경 및 구현 방법

본 논문에서는 부하 분석의 요소를 디스크에 캐쉬된 파일의 전체 크기와 기존 분석 요소인 request 수를 같이 보여준다. 어떤 서버 관리자에게는 request 수를 부하의 요소로 생각할 수 있으므로 서버 관리자는 request 수를 서버의 부하로 보고 부하 균등을 하면 되겠다. 반면 서버에 담긴 디스크에 캐쉬된 파일들의 바이트를 부하로 생각하는 서버 관리자라면 디스크에 캐쉬된 바이트를 부하로 보면 된다. 확장된 모니터링 도구에서는 프락시 서버의 로그(log)를 분석하여 기존의 프락시 서버 관리 모니터링 툴에서 보여주는 기본 정보들과 부하 분석 정보들까지 보여준다. 구현 환경에서 사용하는 로그는 실제 웹 환경에서 사용된 로그를 이용하여 분석하고자 한다. 이 로그는 프락시 서버 도구로 유명한 squid 를 만든 nlanr(National Lab of Applied Network Research)라는 회사에서 실제 squid 서버에 접속한 native log 들을 저장하여 보관하고 있는데 이 로그를 이용하여 모니터링을 구현 한다.[8][9]

표 2)는 본 논문에서 모니터링을 구현할 때의 환경을 나타낸다.

구현 환경 운영체제	알짜 리눅스 6.2
프락시 서버	Squid version 2
확장된 프락시 서버 모니터링 도구 개발 언어	PERL 스크립트 5
서버의 개수 (2개의 그룹)	Psa 1,2,3 (캐쉬된 데이터가 같다) Psb 1,2,3 (캐쉬된 데이터가 같다)
부하 분석 간격	10분 , 30분 , 1시간 , 24시간

표 2) 구현 환경 및 설계

본 논문에서 구현하는 모니터링의 부하 분석을 하기위해 리눅스 6.2 에서 프락시 서버는 squid 를 사용하며 모니터링 도구 개발 언어는 펄 스크립트를 사용한다. 서버는 두개의 그룹으로 나누어진 6 대의 서버를 두고 구현한다. 각각의 그룹은 같은 캐쉬 데이터를 갖고 있으며 부하 분석 시간은 4 가지의 시간별로 나누어서 부하를 분석한다. 두 개의 그룹으로 나눈 이유는 하나의 프락시 서버에 모든 웹 사이트의 정보를 담을 수 없기 때문이다. 부하 분석은 같은 캐쉬 정보를 갖고 있는 두 개의 그룹별로 나누어서 분석한다. 부하를 분석하는 방법은 우선 request 수를 각각의 서버별로 psa1 의 request 수, psa2 의 request 수 이런 방식으로 6 대 서버별로 request 수를 계산하고 각각의 서버 6 대별로 다시 디스크에 캐쉬된 파일들의 바이트 전체 크기를 보여준다. 그림 2)는 모니터링 하는 기본 구성을 나타낸다.

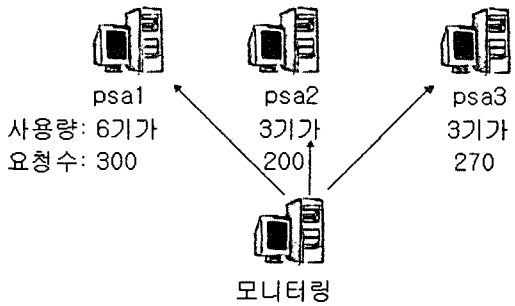

그림 2) 모니터링 구성

위의 그림과 같이 request 수와 디스크에 캐쉬된 데이터들의 바이트 총합을 보여준다. 그리고 각각의 서버별로 다시 디스크에 캐쉬된 유사한 파일들끼리 html, 이미지(gif,jpg), 멀티미디어(video, audio), 기타 나

머지 파일별로 캐쉬된 파일들중 요청된 전체 바이트 크기와 전체 request 수를 보여준다. 다시 각각의 파일별로 request 수와 디스크에 캐쉬된 바이트 크기를 보여준다. 표 3)은 부하를 분석한 결과 값이 나오는 예를 나타낸다.

	Psa 1
전체 request 수	300
요청한 파일들이 디스크에 캐쉬된 전체 크기	125.3 MB
요청한 파일들의 개별 request 수	*Html: 200 , Image : 80* 멀티미디어 : 15, 기타 다른 파일들: 5
요청한 유사 파일들이 디스크에 캐쉬된 개별 크기	Html: 300 KB , Image : 10MB 멀티미디어 : 100 MB 기타 다른 파일들 : 15 MB
개별 유사 파일들의 분석 (html 에 대한 분석)	aa.html => request:2 바이트 : 2KB ab.html => request:3 바이트 : 3KB .....
각각의 파일들 분석	aa.html => request: 2 바이트 : 2 KB ab.gif => request: 1 바이트 : 30 KB ab.html => request: 3 바이트 : 3 KB ac.mpeg => request: 1 바이트 : 3MB .....

표 3) 부하 분석한 후의 결과 예

표 3)은 psa 1 을 모니터링 했을 때 부하의 정보를 나타낸 결과로 전체 서버에 request 된 수와 디스크에 캐쉬된 전체 크기를 나타낸다. 유사한 파일별로 다시 request 수와 캐쉬된 데이터 크기를 보여준다. 그런 후 다시 각각의 파일별로 request 수와 바이트 크기를 보여준다.

이러한 모니터링 분석을 통하여 각각의 유사한 파일들끼리 또는 각각의 개별 파일별로 분석하여 각각의 프락시 서버에 가중되는 부하를 파악하고 부하 중에서 어느 요소가 서버에 많은 과부하를 일으키는지를 파악 하게 된다. 파악한 결과는 차후에 각 서버끼리의 부하 균등 이나 사용자들의 성향 파악에 많은 도움을 줄 수 있다.

4. 결론 및 향후연구

본 논문에서는 기존의 프락시 서버 모니터링 방식에 부하 분석 기능을 추가하였다. 프락시 서버의 캐쉬율을 올리기 위한 정보를 보여주는 기존의 모니터링 방식에서 벗어나 각각의 프락시 서버에서 발생되는 부하의 정보를 제공함으로써 차후에 부하의 균등정책이나 각 서버별로 부하의 특성을 파악 할 수 있다. 이와 더불어 사용자들의 경향을 파악함으로써 좀 더 개선된 프락시 서버를 운용 할 수 있도록 정보를 제공하는 확장된 개념의 모니터링 도구 설계를 하였다.

향후 연구 과제로는 현재 확장된 기능의 모니터링 도구에 대한 설계만 되어 있지만 실제 모니터링 툴을 구현 해 봄으로써 부하 분석을 통한 유저들의 성향 분석이나 각 서버들의 부하 및 프락시 서버들간의 부하 균등 정책을 위한 정보를 제공하는지를 구현해 볼 예정이다.

참고문헌

[1] daniel a. menasce, virgilio a.f. almeida ,"capacity planning for web performance" , 1997 prentice hall

[2] Xiaohui Zhang, "Cachability of Web Objects", Boston University, 1999

[3] Ghaleb Abdulla, Edward A.Fox,Marc Abrams, and Stephen Williams "WWW Proxy Traffic Characterization with Application to Caching" 1997

[4] web trends, http://www.webtrends.co.kr

[5] calamaris, http://core.de/tools/squid/calamaris/

[6]http://www.webtrends.co.kr/sample/Proffessional/PROXY/report.htm

[7] John dilley, Martin Arlitt, and Stephane Perret. "Enhancement and Validation of Squid's Cache Replacement Policy", HP Laboratories, 1999

[8] National Lab of Applied Network Research, sanitized access log, (ftp://ircache.nlanr.net/Traces/)

[9] Squid Internet Object Cache, http://squid.nlanr.net/Squid/