SAN 기반 공유파일 시스템을 사용한 웹 서버에서의 효율적인 웹 분배 방식에 대한 연구

안철우[0] 백광호 황주영 김경호 이철 박규호
한국과학기술원 전자전산학과 전기및전자공학전공 컴퓨터공학연구실
(cwahn, khbaik, jyhwang, kyhkim, chullee, kpark)@core.kaist.ac.kr

A Cache-Aware Request Dispatching on the Storage Area Network based Shared File System

Chul Woo Ahn[0], Kwang-Ho Baik, Joo-Young Hwang, Chul Lee, Kyu-Ho Park,
Dept. of Electrical Engineering and Computer Science, KAIST

요 약

본 논문에서는 SAN(Storage Area Network) 기반의 공유파일시스템인 Asphodel 파일 시스템을 이용하여 클러스터 웹 서버를 구성하였다. 그리고 Asphodel 공유 파일 시스템이 가지는 락(lock) 서버를 이용하여 효율적인 웹 분배 정책인 CARD(Cache-Aware Request Dispatch)를 제안하고 이를 설계 구현하였다. 그 결과 후위 서버의 메모리 캐쉬의 적중률을 높임으로써 보통의 분배 정책인 라운드로빈(Round-Robin) 방식에 비해 웹 서버의 throughput 과 latency에서의 성능 향상을 보였다

1. 개 요

폭발적인 인터넷 사용자의 증가에 따라 인터넷 시스템의 웹 서버의 병목 현상이 두드러지고 있다. 이러한 병목 현상을 효율적으로 해결하고자 지난 몇 년간 PC서버를 이용하여 웹 서버 클러스터를 구성하는데 많은 연구가 이루어 왔다. 보통 클러스터 웹 서버의 구성은 2계층 구조를 많이 사용한다. 후위(Back End)에 실제 서비스를 담당하는 웹 서버를 놓고 전위(Front End)에는 후위의 웹 서버에 웹 요청을 효율적으로 분배하기 위한 분배기(dispatcher)가 존재하게 된다.

클러스터 웹 서버 환경 하에서 발생할 수 있는 주요한 문제점으로는 크게 2가지가 존재 할 수 있다. 첫번째는 분배기의 효율적이지 못한 분배 정책으로 인해 후위의 특정 한 웹 서버에 웹 요청이 집중적으로 몰리게 되어 클러스터 자원을 효율적으로 사용하지 못하는 경우가 있을 수 있다. 두 번째로는 후위의 웹 서버와 연결되어 있는 파일서버나 데이터 베이스 서버와의 데이터 송수신에 있어 입출력 병목 현상(I/O bottleneck)이 발생할 수 있다.

기존의 분산 파일 시스템의 경우는 후위 서버의 모든 파일에 대한 엑세스가 파일서버를 통하기 때문에 서버에서 병목 현상이 일어나게 된다. 이러한 입출력 병목 현상을 줄이기 위해서는 높은 대역폭을 가지는 SAN(Storage Area Network) 기반의 공유 파일 시스템이 효과적인 해결방안이 될 수 있다.

본 논문에서는 Fiber Channel을 이용한 SAN기반의 SANfs 공유 파일 시스템을 이용하여 공유 파일 시스템 기반의 클러스터 웹 서버를 구성하였다. 그리고 이를 바탕으로 공유 파일 시스템의 락 서버를 이용한 효율적인 CARD(Cache-Aware Request Dispatch) 웹 요청 분배정책을 제안하고 이를 실제 설계하고 구현하였다.

2. 관련연구

클러스터 웹 서버에서 후위 서버로의 웹 요청 분배에는 크게Round-Robin 방식과 LARD (Locality-Aware Request Distribution)방식이 있다[1].

라운드 로빈 방식은 들어오는 웹 요청을 순서에 따라서 차례로 후위로 분배하는 방식으로 같은 웹 요청이 다른 후위로 분배될 확률이 높아진다. 이는 웹 요청이 요구하는 데이터를 디스크에서 읽어오게 되는 경우를 증가시키므로 전체 웹 서버의 성능을 떨어뜨리게 된다. 또한 여러 웹 요청들을 처리하는 비용이 후위 노드에 따라 다를 수가 있는데, 라운드 로빈 방식은 이 점을 분배에 반영을 할 수 없는 단점이 있다.

이러한 문제를 극복하기 위한 부하 분배 방법이 콘텐트 기반 분배 방식(content-based distribution)이다. 콘텐트 기반 분배 방식은 웹 요청이 요구하는 데이터에 따라서 분배를 결정하는 방식으로 요구되는 데이터가 메모리에 있을 확률을 높이는 방식이다. 이러한 콘텐트기반 분배 방식의 대표적인 경우가 LARD 방식이다. LARD 방식에서는 웹 요청의 종류에 따라서 해당 후위 노드를 미리 정해놓는다. 따라서 같은 웹 요청은 같은 후위 노드에서 처리하게 되므로 메모리에 웹 요청이 요구하는 데이터가 있을 확률이 높아진다. 하지만 웹 요청의 빈도에 의해서 각 후위 노드의 부하가 불균등해질 수 있다. 따라서 전위 노드는 후위 노드의 상황을 계속 모니터링(monitoring)하면 필요하면서 재분배를 한다[1].

이러한 콘텐트 기반 부하 분배 방식은 전위 노드의 과부하를 초래할 수도 있다. 일반적으로 부하의 분배문제 때문에 전위 노드는 후위 노드보다 성능이 높은 하드웨어를 사용한다. 이는 그만큼 전위 노드에 부하가 집중된다는 의미이기도 하다. 전위 노드의 과부하는 전체 클러스터 시스템의 확장성에 많은 제약을 주기 때문에 바람직하지 않다. 따라서 비교적 간단한 방법의 분배 방식으로 전위 노드의 과부하를 해결하는 것이 바람직하다.

3. SANfs 공유 파일 시스템기반의 웹서버

공유 파일 시스템은 시스템내의 모든 클라이언트 노드가 네트웍을 통해서 직접 공유 디스크를 엑세스(access) 할 수 있는 파일 시스템이다. SANfs 공유 파일 시스템은 본 논문에 서 제안하는 CARD(Cache Aware Request Dispatch)을 적용하여 실험을 한 SAN 기반의 공유 파일

시스템이다. 1999년부터 개발되어온 SANfs 공유 파일 시스템은 인텔 펜티엄(Intel pentium)을 기반으로 하는 리눅스(linux) 플랫폼에서 구현되었다[2]. SANfs 공유 파일 시스템의 락 서버는 하나의 전용 노드가 아니고, 임의의 클라이언트에서 유저 영역(user level) 프로세스로 존재하면서 락 서버의 역할을 한다. 만약 파일 시스템에 연결된 클라이언트와 디스크의 용량이 늘어난다면 이 락 서버를 여러 클라이언트에 분산시킴으로써 확장성을 유지할 수 있다. SANfs 공유파일 시스템의 장점은 다음과 같다.

첫 번째로 디스크 용량을 효율적인 이용 할 수 있다. 클러스터 웹 서버에서 각각의 노드가 지역 디스크(local disk)를 이용하고 이를 공유하지 않는다면, 웹 요청 처리를 위해서 모든 노드가 같은 데이터를 디스크에 저장하고 있어야 한다. 이는 디스크의 용량을 비효율적으로 이용하는 것이 된다. 공유 파일 시스템은 클러스터의 모든 노드가 디스크를 공유하게 되므로 불필요한 중복 데이터를 보유할 필요가 없어지게 된다. 따라서 디스크 용량을 효율적으로 사용할 수 있게 된다.

두 번째로 가용성 측면에서 장점을 얻을 수 있다. 클라이언트 중의 하나가 고장이 난다고 해도 전체 시스템에는 아무런 영향이 없다. 모든 클라이언트가 직접 디스크를 액세스할 수 있기 때문이다. 기존의 분산 파일 시스템에서는 서버가 데이터의 공유, 메타 데이터의 관리 등을 하기 때문에 서버의 고장은 전체 시스템의 고장으로 이어지는 단점이 있다. 공유 파일 시스템에서는 서버가 없기 때문에 이러한 문제가 존재하지 않는다.

웹 서버로의 요청을 처리할 때 필요한 데이터는 디스크에 있을 수도 있고, 서버의 메모리에 있을 수도 있다. 서버의 메모리가 전체 파일 시스템의 크기에 비해서 작지만, 데이터의 시간적 공간적인 지역성(locality) 때문에 디스크로 요청이 되는 경우는 20%정도이다[4]. 하지만 요청된 데이터가 디스크에서 서비스될 때는 메모리에서 서비스 되는 경우보다 100~1000 배 차이가 나기 때문에[5], 디스크로의 요청이 줄어들 수록 전체 웹 서버의 성능은 높아진다고 볼 수 있다

메모리의 데이터에 대한 적중률을 높일 수 있는 한가지 방법으로는 전위 노드가 전체 디스크 액세스를 줄일 수 있는 부하 분배를 하는 것이다. 본 논문에서는 이러한 디스크 액세스를 줄일 수 있는 부하분배 방식을 제안한다.

5. CARD (Cache-Aware Request Dispatch)

5.1. CARD 부하분배방식

CARD는 부하 분배를 SANfs 공유 파일 시스템의 락 서버 프로세스의 정보를 바탕으로 분배를 결정한다 (그림 1참조). 이러한 방식을 사용하면 분배 결정이 간단해지는 장점이 있다.

전위 노드가 웹 요청을 받아들이면, 웹 요청에 필요한 데이터를 알아내게 된다. 그리고 락 서버 프로세스에게 가장 최근에 해당하는 데이터에 대해 락 요청(Lock Request)한 후위 노드에 대해서 물어보게 된다. 최근에 락 요청을 한 후위 노드일수록 해당 데이터를 메모리에 가지고 있을 확률이 높기 때문이다. 왜냐하면, 락 요청을 했다는 것은 해당 데이터를 디스크로부터 읽어오거나 썼다는 의미이기 때문이다. 최근에 디스크로부터 해당 데이터를 가지고 온 노드일수록 메모리에 데이터가 있을 확률이 높게 된다.

물론 이상적인 경우는 모든 후위 노드의 메모리에 저장되어 있는 데이터에 대해서 모니터링을 하고 있다가 이를 이용해서 분배를 하는 것이다. 하지만, 이런 경우 전위 노드가 관리해야 할 정보의 양이 너무 많아지고 이는 전위 노드에 과부하를 주게 되어서 앞에서 지적한 확장성의 제한을 가져오게 된다. 따라서 CARD는 트레이드오프(Trade off)를 통해서 전위 노드의 부담을 줄이면서 동시에 낮은 디스크 액세스를 추구하는 분배 방식이라고 할 수 있다. CARD는 기존의 락 서버 프로세스의 정보를 이용하기 때문에 구현이 간단한 장점이 있다. 후위 노드는 전혀 고칠 필요가 없기 때문이다

5.2 CARD의 동작

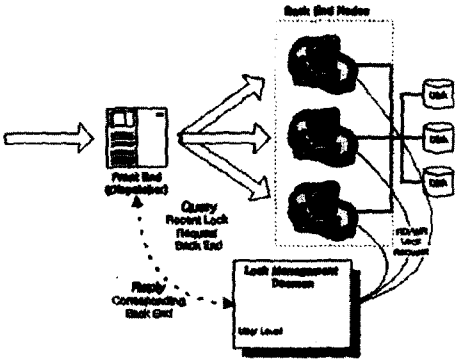

그림1. CARD의 구조

CARD에서는 전위 노드에서 들어오는 웹 요청을 락 관리 프로세스를 참조해서 후위 노드로 분배한다. 락 관리 프로세스는 후위 노드에서 읽기/쓰기에 대한 락 요청이 있을 때마다 해당 데이터(file's inode)에 대한 정보를 갱신한다. 전위 노드가 웹 요청 분배 시에 참조하게 되는 락 요청 테이블(recent lock request node table, 그림 2는 락 관리 프로세스에 의해서 최신의 정보로 계속적인 업데이트(update)가 이루어진다.

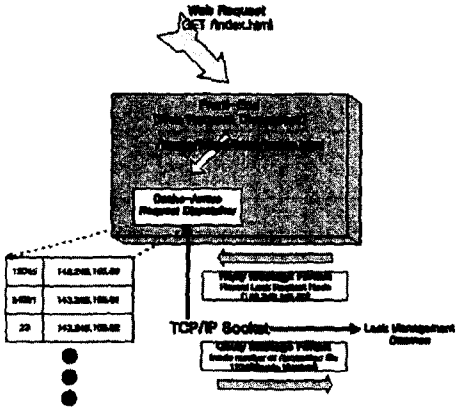

그림2. CARD의 동작

만일 분배기 테이블에 등록되어 있는 lock이 오래 전

에 등록된 후 한번도 교체가 안된 채 후위 노드가 이러한 락을 가지고 있는 파일을 교체한다면 이는 miss 상태가 될 것이다. 따라서 이러한 현상을 방지하기 위해 후위 노드가 락을 풀 때에 분배기에게 알려 분배기에 존재하는 해당 락을 제게 하도록 한다.

락 관리 프로세스가 유지하고 있는 락 요청 테이블은 일종의 후위 노드의 메모리에 대한 힌트(hint)라고 볼 수 있다. 정확하게 메모리의 데이터 캐싱(data caching) 상황을 관리하는 것은 앞에서 말한 바와 같이 너무 많은 비용이 들기 때문에 적중률이 비교적 높은 힌트 정보를 이용하는 것은 좋은 해결 방안이 될 수 있다.

5.3 CARD의 구현

앞에서 설명한 바와 같이 SANfs 공유 파일 시스템은 락에 대한 관리를 백-엔드의 노드의 사용자 영역의 전용 락 관리 프로세스에 의해서 이루어진다. 따라서 CARD와 락 관리 프로세스 사이의 통신이 필요로 하게 된다. 앞에서 설명한 바와 같이 CARD와 락 관리 프로세스 사이의 통신은 TCP/IP 프로토콜을 이용한다. CARD가 시작할 때 락 관리 프로세스와 TCP/IP 소켓을 생성하여서 초기화를 하고, 이를 이용하여 통신을 수행한다

CARD는 락 관리 프로세스로부터 얻은 정보를 따로 메모리에 저장한다. 이는 같은 요청이 반복될 때에 TCP/IP 통신을 통하지 않고 저장된 값에 따라서 웹 요청을 분배하게 된다. 이는 TCP/IP 통신의 비용을 줄이기 위해서이다. 그림 2에서 보이는 테이블은 CARD가 락 관리 프로세스로부터 받은 정보를 메모리에 저장한 것이다.

락 관리 프로세스는 기존의 락 관리 작업 외에 CARD의 요청에 응답해야 한다. 후위 노드로부터 락 요청을 받아들일 때 이를 기록해야 한다. 하나의 inode에 대해서 가장 최근에 락 요청을 한 후위 노드를 항상 업데이트 해야 한다.

6. 실험결과 및 분석

CARD의 성능을 평가하기 위해서 SPECWeb 99 벤치마크[6]를 이용하였다. SPECWeb 99에서 웹 요청을 발생시키고 이를 SANfs 공유 파일 시스템을 기반으로 하는 리눅스 아파치 웹 서버에 가했다. 클러스터 웹 서버의 각 노드는 256MB의 메인 메모리를 사용하고 인텔 펜티엄 칩을 사용하는 리눅스 머신이다. 클러스터 노드들 간의 연결은 100Mbps 이더넷을 사용했고 노드와 디스크 사이의 연결은 Fibre-Channel을 이용하였다.

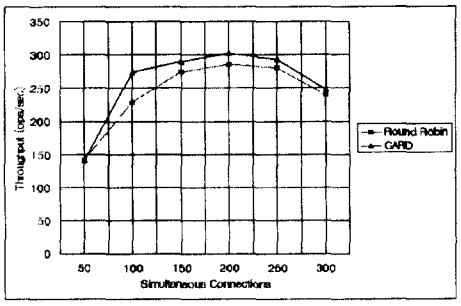

그림3. Throughput의 비교

SPECWeb 99에서 동시 연결 수(Simultaneous Connections)는 SPECWeb의 클라이언트가 웹 서버와 맺고 있는 연결의 개수이다. 동시 연결 수가 300이 넘어가게 되면 웹 서버는 포화상태에 이르게 되었다.

그림 3은 throughput을 나타낸 것인데, CARD가 라운드 로빈 방식에 비해서 최고 25%정도의 성능 향상이 있었다. CARD의 성능이 뛰어난 것은 메모리 캐쉬 적중률(memory cache hit ratio)이 올라가서 이기 때문이다.
그림 4는 응답시간을 비교한 것인데, 전체적으로 CARD가 라운드 로빈 방식보다 응답 시간이 작았다. 메모리 캐쉬 적중률이 증가로 인해서 디스크 액세스 가 줄어들어서 latency에서도 이득을 보았기 때문이다.

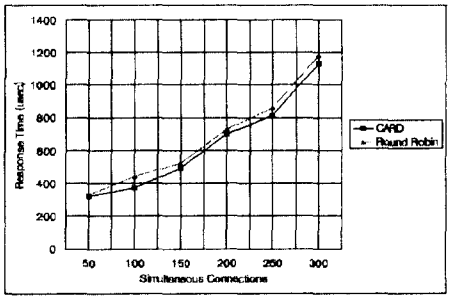

그림4. Latency 의 비교

7. 결론 및 추후과제

본 논문에서는 공유 파일 시스템 기반의 클러스터 웹 서버에서의 메모리 캐쉬 적중률(memory cache hit ratio)을 높이는 방법에 대해서 제안하였다. CARD(Cache Aware Request Distribution)은 공유 파일 시스템의 락 정보를 이용하여 웹 서버의 부하 분배를 메모리 캐쉬 적중률을 높일 수 있는 방향으로 하는 것이다. 벤치마크 결과에 따르면 CARD는 웹 서버의 throughput과 response Time을 향상시켜주었다. 추후과제로는 여러 가지 실험을 통해 분배기의 최적화를 위한 튜닝(Tuning) 및 아울러 동일한 조건에서의 LARD를 적용한 시스템을 구현한 후 제안하는 CARD와의 성능 비교 분석이 있어야 할 것이다.

참고문헌

[1] G. Banga V. Pai, M. Aron. Locality-aware request distribution in cluster based network servers. ASPLOS-VIII, October 1998.
[2] SANfs Shared File System. http://core.kaist.ac.kr
[3] Minix file system http://www.minix.org
[4] M. Arlitt, C. Williamson. "Internet Web Servers: Workload Characterization and Performance Implications" IEEE Transactions on Networking. 5(5), October 1997.
[5] L. McVoy, C. Staelin. "Imbench: Portable Tools for Performance Analysis". Proceedings of the USENIX 1996 Annual Technical Conference, January 1996
[6] Specweb http:// www.specbench.org