윈도우를 사용한 웹 프록시 저장장치 관리 기법

서본근[0] 오승택 이준원
한국과학기술원 전산학과
bkseo@camars.kaist.ac.kr0, stoh@calab.kaist.ac.kr, joon@cs.kaist.ac.kr

Storage management of web proxy with window

Seo Bon Keun[0] Seung-taek Oh Joon-Won Lee
CS. Div. Dept. of EECS. KAIST

요 약

웹 프록시의 처리량을 향상하고자 하는 시도로 Hummingbird가 제시되었다. 그러나 Hummingbird의 복구 방식은 느리기 때문에 좀더 빠르게 웹 프록시 저장장치를 복구할 수 있는 방법이 필요하게 되었다. 본 연구에서는 빠른 복구를 위하여 디스크의 일부만 읽어서 복구할 수 있도록 메타 정보를 위한 공간을 할당하도록 하였다. 또한 메타 정보를 기록하는 비용을 줄이기 위해 윈도우를 사용한 메타 정보 관리 방식을 적용하여 윈도우 안에서만 클러스터를 할당 · 삭제할 수 있도록 하며, 윈도우의 이동에 따라 메타 정보를 모아서 기록할 수 있도록 한다. 윈도우를 사용한 메타 정보 관리 방식에서는 윈도우에 포함된 클러스터들의 메타 정보만이 잘못될 수 있도록 하여 복구 시에 윈도우에 포함된 클러스터들의 메타 정보를 삭제하고 빠르게 복구할 수 있다.

1. 서 론

웹 프록시는 웹 클라이언트들의 요청을 받아 웹 서버에 있는 파일들을 임시로 저장하고 웹 클라이언트에 전달하는 것을 통해 웹 클라이언트가 빠르게 파일들을 받을 수 있게 한다. 웹 프록시는 주로 빠른 속도로 연결된 지역 네트워크에서 외부로 나가는 요청들을 처리하며 이를 통해 비교적 느린 외부 회선의 대역폭을 확장시키는 것과 같은 효과를 얻을 수 있도록 하고 있다. 하지만 웹 프록시의 성능이 나쁜 경우에는 오히려 네트워크의 병목으로서 작용하는 경우도 발생할 수 있다.[4,6]

웹 프록시가 네트워크 성능에 영향을 미치게 되는 첫 번째 요소로 명중률을 들 수 있다. 웹 프록시의 명중률은 주로 저장장치의 용량과 파일 대체 알고리즘에 의해 결정되는데, 근래에 지속적인 하드디스크 드라이브의 용량 향상과 가격저하로 인해 웹 프록시 저장장치의 용량이 증가함에 따라 명중률이 증가하는 추세에 있다.

또한 웹 프록시가 처리할 수 있는 대역폭의 한계인 처리량도 네트워크의 성능에 영향을 미친다. 웹 프록시의 처리량이 네트워크의 대역폭보다 작은 경우 네트워크의 성능을 떨어뜨리는 효과를 일으킨다. 더욱이 네트워크의 대역폭이 지속적으로 증가하고 있으며 웹의 사용이 늘어남에 따라 웹 프록시의 처리량의 향상은 지속적으로 요구되고 있다. 웹 프록시의 처리량을 향상시키기 위해서 가장 중요한 것 중의 하나가 웹 프록시 저장장치의 성능 향상이다. 웹 프록시의 응답 시간의 30%정도가 웹 프록시 저장장치에 영향을 받으며[1] 일반적으로 하드디스크 드라이브는 느린 장치에 속하기 때문에 웹 프록시의 처리량은 웹 프록시 저장장치의 처리량에 큰 영향을 받게 된다.

웹 프록시에 저장된 파일들은 웹 서버에 있는 파일들의 사본이기 때문에 삭제되어도 웹 서버로부터 다시 받아와서 저장할 수 있다. 따라서 웹 프록시 저장장치의 복구는 중요하게 여겨지지 않았다. 그러나 웹 프록시의 용량이 증가함에 따라 웹 프록시 저장장치를 채워 최대값의 명중률을 얻는 데는 많은 시간이 걸리며[3] 그동안에 네트워크에 추가적으로 걸리는 부하는 네트워크의 성능을 떨어뜨리는 요인이 된다. 따라서 웹 프록시 저장장치의 복구가 필요하게 되었다.

이러한 필요에 부합하기 위하여 Shriver등은 Hummingbird라는 사용자 수준 파일시스템 라이브러리를 구현하였다.[3] Hummingbird에서는 지역성이 있는 파일들을 클러스터라는 단위로 모아서 기록하고, 파일을 읽어야 하는 경우 클러스터를 통째로 디스크에서 읽어오는 것을 통해 디스크 접근을 줄이도록 한다. 또한 저장장치의 복구를 위하여 클러스터에 포함된 파일들의 메타 정보를 클러스터에 함께 기록하여 디스크에 기록하도록 하여, 웹 프록시가 비정상 종료된 경우에도 디스크에 기록된 클러스터들의 메타 정보를 읽어서 저장장치의 내용을 복구해 낼 수 있도록 하고 있다.

하지만 Hummingbird에서는 저장장치를 모두 복구하기 위해 디스크 전체를 읽어야 하므로 복구에 많은 시간이 걸린다는 단점이 있다. 본 연구에서는 이를 개선하기 위하여 디스크의 일부만을 읽어 저장장치의 복구가 가능하도록 하며, 이를 위하여 별도의 복구에 필요한 내용을 기록하는 공간을 디스크에 할당하도록 하는 방법을 제안한다. 또한 이 경우에 발생하는 디스크 접근을 줄이기 위해 윈도우를 사용하여 메타 정보를 기록하는 방법을 사용하고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 Hummingbird의 구체적인 아이디어와 그 경우에 발생하는 문제점을 알아보고 3장에서 이러한 Hummingbird의 문제를 해결하기 위해 본 연구에서 제시한 윈도우를 사용한 메타 데이터 관리 기법에 대하여 설명하며 4장에서 실험을 통한 실제적인 성능 향상에 대해서 보이고 있다. 마지막으로 5장에서 전체를 정리하고 결론을 내리도록 한다.

2. Hummingbird의 구조와 문제점

2.1. 클러스터화

웹에 있는 파일들은 한두 개의 html파일에 여러 그림 파일들이 포함되어 한 페이지를 이루는 경우가 대부분이며 이러한 파일들은 한번에 요청되므로 파일들에는 지역성이 있게 된다. Hummingbird에서는 웹 프록시가 제공하는 지역성 정보를 사용하여 지역성이 있는 파일들을 클러스터라는 정해진 크기의 비교적 큰 단위로 모아서 기록하는 것을 통해 파일들을 디스크에 기록하는 비용을 줄이고 있다. 또한 디스크에 기록된 파일

을 읽을 필요가 있는 경우 클러스터를 통째로 읽어 한번에 여러 파일을 읽을 수 있도록 하며, 이러한 파일들은 빠른 시간 내에 사용될 가능성이 높으므로 같은 수의 파일을 읽는데 필요한 디스크 접근을 줄이는 효과를 준다. Shriver등은 이 방식을 사용하여 Squid의 경우보다 2.3~13배 정도의 성능 향상을 얻었음을 보고하고 있다.

Hummingbird에서는 파일이 어느 클러스터에 포함되어 있는지를 찾기 위해 메모리에 상주하는 해시 테이블을 사용하여 파일이 포함된 클러스터를 찾도록 하고 있다. 그러나 이 해시 테이블이 메모리에만 존재하므로 웹 프록시가 재시작된 경우 이 해시 테이블의 내용을 재구성할 필요가 있다. 이러한 작업을 웹 프록시 저장장치의 복구라고 할 수 있다.

2.2. 저장장치 복구의 문제

Hummingbird에서는 클러스터에 클러스터가 포함하고 있는 파일의 내용과 함께 각 파일들에 대한 메타 정보를 함께 기록하도록 하여 비정상적으로 웹 프록시가 종료된 경우에 디스크에 기록된 모든 클러스터들을 읽어와 파일이 포함된 클러스터를 알게 하는 해시 테이블을 구성해 낼 수 있도록 하고 있다. 그러나 이 과정에서 디스크 전체를 읽어야 하므로 전체 복구에는 많은 시간이 걸리며 저장장치의 용량이 늘어날수록 복구에 걸리는 시간은 길어진다.

따라서 Hummingbird에서는 주기적으로 자주 사용되는 클러스터에 대한 로그를 남겨서 전체 복구 과정을 수행하기 전에 이 클러스터들에 대해 복구를 먼저 수행하고 이 클러스터들을 읽을 수 있게 하여 전체 복구 과정이 수행되는 긴 시간동안에도 어느 정도의 명중률을 낼 수 있도록 하고 있다. 그러나 디스크 전체를 읽어야 하는 전체 복구 과정이 진행되는 동안 복구되지 않은 클러스터들에 대한 접근이 불가능하며, 새로운 파일들을 기록하지 못한다는 문제점이 있다. 이것은 복구가 끝났을 때 명중률을 떨어뜨리는 이유로 작용할 수 있다. 따라서 이러한 Hummingbird의 느린 복구를 빠르게 할 수 있도록 하여 디스크 전체를 사용하는데 짧은 시간이 걸리도록 하여야 한다.

3. 윈도우를 사용한 메타 정보 관리

3.1. 윈도우를 사용한 클러스터 할당 및 메타 정보 기록

Hummingbird에서는 디스크 전체를 읽기 때문에 복구에 많은 시간이 걸리는 문제가 발생한다. 따라서 본 연구에서는 이를 해결하기 위하여 디스크에 파일의 메타 정보만을 기록하는 부분을 할당하여 각 클러스터마다 어떠한 파일이 어느 위치에 기록되어 있는지를 기록하도록 하고 있다. 따라서 이 부분만을 읽으면 복구가 가능하게 되며 이 부분이 전체 디스크보다는 매우 작기 때문에 복구 시간을 줄일 수 있다. 그러나 이 경우에는 메타 정보를 기록하기 위한 추가적인 디스크 접근이 발생하여 이러한 추가적인 부하는 최소화하여야 한다. 메타 정보를 기록하는 디스크 접근은 SoftUpdate[2]와 같이 여러 파일의 변경을 모아서 기록하는 것을 통해 줄일 수 있다.

웹 프록시는 파일 대체가 일어나는 지점이 매우 불규칙하며 넓은 범위에서 일어나기 때문에, 메타 정보가 변경되는 범위가 매우 넓다. 따라서 메타정보를 모아서 기록하면 디스크의 메타 정보가 갱신되는데 많은 시간이 걸리게 되어 디스크에 기록된 메타 정보의 신뢰성이 떨어진다.

본 연구에서는 메타 정보의 효과적인 메모리 캐싱을 위해 메타 정보가 변경되는 범위를 한정시키는 방법을 적용하였다. 메타 정보의 변경을 일으키는 새 클러스터를 기록하는 작업에 대해 새로운 클러스터를 할당받는 범위를 제한하였으며, 이를 위해 윈도우를 만들어 윈도우에 해당하는 클러스터들에 대해서만 할당 및 기록될 수 있도록 하였다. 그리고 클러스터들이 기록되어감에 따라 윈도우를 이동하고 윈도우가 지나간 클러스터들에 대해서 변경된 메타 정보들을 디스크에 한번에 기록하도록 하였다. 이렇게 하여 메타 정보들을 모아서 기록할 수 있게 되었다.

3.2. 윈도우를 사용한 파일 삭제 방법

윈도우를 사용하여 클러스터를 할당하면 클러스터를 할당받는 범위가 윈도우에만 국한하기 때문에 윈도우에 포함되지 않는 파일들을 삭제하는 것은 공간의 낭비를 일으키게 된다. 따라서 새 클러스터를 기록하기 위한 공간을 만들기 위해 파일들을 삭제하는 것을 윈도우 안에서만 일어날 수 있도록 하였다.

기존의 웹 프록시에서 파일을 삭제하는 것은 파일별로 우선순위를 매기고 그에 따라 파일을 삭제하였다. 그러나

이와는 반대로 클러스터별로 우선순위를 매기고 클러스터에 포함된 모든 파일을 지워 새 클러스터를 기록하기 위한 공간을 생성하는 클러스터별 삭제 방식을 적용할 수 있다. 이 경우에는 사용될 가능성이 비교적 높은 파일이 삭제될 수 있기 때문에 명중률이 감소하지만, 디스크 접근 당 읽을 수 있는 파일 수를 늘려 웹 프록시 저장장치의 처리량을 향상시킬 수 있다.

이 방법에서는 LRU를 사용하며 대체 순위의 일정 비율에 해당하는 클러스터들을 대체 대상으로 지정하여 두었다가 윈도우 안에 포함된 경우 이 클러스터에 포함된 파일을 모두 삭제하여 빈 공간에 새 클러스터를 기록할 수 있도록 하였다.

이와 같이 윈도우를 사용하여 클러스터들을 할당하고 파일들을 삭제하면 윈도우가 지나가면서 메타 정보가 기록되고 윈도우에 포함되기 전에는 클러스터의 내용 및 메타 정보의 내용이 바뀌지 않으므로 윈도우에 해당하지 않는 영역은 디스크에 기록된 메타 정보의 내용이 올바르다.

3.3. 저장장치 복구 알고리즘

본 연구에서 사용된 윈도우를 사용한 방식에서 저장장치의 복구란 디스크에 기록된 메타 정보의 내용을 기초로 디스크에 기록된 메타 정보의 내용을 실제 클러스터의 내용과 일치할 수 있도록 보정하는 작업과 웹 프록시에서 사용하는 메모리 자료구조를 종료되기 전의 상태로 재구축하는 작업으로 이루어진다. 앞에서 말한바와 같이 메타 정보가 실제 디스크에 기록된 클러스터의 내용과 일치하지 않는 부분은 시스템이 비정상 종료되기 직전의 윈도우에만 해당하므로 다음과 같은 간단한 방법으로 복구를 할 수 있게 된다.

1) 슈퍼 블록을 읽어 정상 종료된 경우인지 판단하고, 실제 사용된 공간들의 크기를 결정한다.

2) 디스크의 메타 정보 영역에 해당하는 내용을 메모리에 읽어서 메타 정보의 내용을 바탕으로 윈도우의 범위를 찾아낸다. 이는 각 메타 정보에 기록한 시간을 포함시켜 기록하여 기록된 시간이 과거로 돌아가는 지점이 윈도우의 시작부분임을 판별할 수 있다.

3) 정상적으로 종료되는 경우에는 메타데이터를 모두 기록할 수 있으므로 디스크의 메타데이터가 올바르다. 그러나 비정상 종료의 경우는 복구할 때 메타 정보를 보정해야 한다. 따라서 읽어온 내용 중 윈도우에 해당하는 영역의 메타 정보를 모두 삭제하고 디스크에 기록하여 클러스터의 내용과 메타 정보 영역의 내용을 일치시킨다.

4) 3)의 결과 생성된 메타 정보를 바탕으로 메모리 상주 해시 테이블을 재구성한다.

5) 윈도우를 종료 전의 윈도우의 위치로 설정하고 저장장치를 사용할 수 있도록 한다.

이후 웹 프록시 저장장치에 저장되어 있는 파일들을 읽을 수 있으며 클러스터의 할당 및 기록이 가능하게 된다.

4. 성능 평가

이 연구에서 제안한 윈도우를 사용한 메타 정보 관리 방식의 효과를 파악하기 위하여 Hummingbird의 방식과 윈도우를 사용한 방식에 대해서 시뮬레이터를 구현하여 성능을 비교하여 보았다. 이 시뮬레이터는 웹 프록시의 로그를 사용하여 실제 네트워크의 상황을 재현하였다. 또한 실제 디스크를 사용하지 않고 디스크 I/O에 걸리는 시간을 측정할 수 있도록 하기 위해 DiskSim[5]을 사용하였다. 모든 방식에서 메모리 대체 알고리즘으로는 파일별 LRU를 적용하도록 하였다.

4.1. 평가 환경

본 연구의 성능 평가에서는 한국과학기술원의 투명 웹 프록시 서버의 2002년 8월 22일의 로그를 사용했는데 이 로그는 평균 12킬로바이트의 크기에 약 18,000,000개의 작업이 기록되어 있다. 여기에 시뮬레이터로 16GB의 저장장치 공간에 128MB의 메모리 캐시를 사용하는 시스템에 대해 실험하되 클러스터의 크기는 32KB, 64KB, 128KB, 256KB를 사용하여 각각의 경우에 대해서 성능을 측정해 보았으며 파일들의 지역성 정보는 클라이언트들의 IP를 주어 같은 클라이언트가 요청한 파일들을 모으도록 하였다.

4.2. 실험결과

각 방식에서의 명중률의 차이는 표 1과 같이 나타났다. 이 결과는 윈도우를 사용하는 방식을 적용하면 윈도우 안에서만 파일들을 삭제하기 때문에 삭제 우선순위가 낮은 파일이 먼저 지워지는 경우가 많아 명중률이 감소하는 것을 보여준다. 그러나 그 차이는 1~2%정도로 비교적 작기 때문에 네트워크 성능에 큰 영향을 미치지 않는다.

클러스터 크기	32KB	64KB	128KB	256KB
Hummingbird	79.4	79.4	79.5	79.5
윈도우	77.4	77.2	77.2	77.1

표 1 클러스터의 크기별 명중률의 차이 (단위 : %)

각 방식에서의 처리량은 표 2와 같았다. 이 결과를 통하여 윈도우를 사용한 메타 정보 관리 방식은 처리량을 떨어뜨리지 않는 방식임을 알 수 있다.

클러스터 크기	32KB	64KB	128KB	256KB
Hummingbird	172	283	332	301
윈도우	392	625	743	794

표 2 사용한 방식에 따른 처리량 차이 (단위 : req/sec)

본 연구에서 파일 시스템의 복구에 필요한 시간을 비교하기 위해 Hummingbird의 복구 방식을 구현하여 비교하는 것은 무리가 있다. 따라서 Hummingbird에서 복구에 걸리는 시간에 가장 크게 영향을 미치는 전체 디스크 읽기에 걸리는 시간과 윈도우를 사용한 복구 방식에서 실제로 걸린 시간을 비교하여 복구에 걸리는 시간을 비교하였다.

복구에 걸리는 시간에 대한 실험 결과 Hummingbird는 16GB 하드 전체를 읽는데 8분 33초가 걸렸으며, 디스크의 대역폭을 줄이기 클러스터의 앞부분 4KB씩만을 읽는 방법을 사용한 경우 7분 56초가 걸렸다. 그러나 윈도우를 사용한 방식에서는 불과 5초만에 복구과정이 완료되었다. 이 결과에 의하면 최소한 100배 정도 빠르게 복구가 가능함을 알 수 있다. 또한 [3]에 의하면 Hummingbird의 자주 사용되는 클러스터들의 복구에 8GB의 용량인 경우 최소 30초가 걸리며 완전한 복구 과정을 마치기 위해서는 이보다 훨씬 많은 시간이 걸린다는 사실을 알 수 있다. 그러나 이것은 심지어 윈도우를 사용한 방식에 비해 6배 이상 느리다.

5. 결론

본 연구에서는 Hummingbird의 복구가 느리다는 문제를 해결하기 위하여 윈도우를 사용한 메타 정보 관리 방식을 제안하였다. 이 방식은 디스크의 일부분에 메타 정보를 기록하기 위한 공간을 할당하고 이 공간에 효율적으로 메타 정보를 기록할 수 있도록 하였다. 또한 이 방식은 새로운 클러스터 할당 및 삭제, 그리고 복구 방식을 제시하는데 이를 통해 효과적인 웹 프록시 저장장치 복구가 가능하게 되었다.

제안한 방식의 성능에 대해 평가하기 위하여 각 방식들에 대해 시뮬레이터를 구현하고 실험한 결과 윈도우를 사용하여 메타 데이터를 관리하는 방식을 사용하여 Hummingbird보다 100배 이상 빠르게 복구가 가능함을 보였다.

References

[1] Alex Rousskov and Valery Soloviev, "On performance of caching proxies," In Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '98/PERFORMANCE '98), 1998.

[2] Marshall K. McKusick and Gregory R. Ganger, "Soft updates: a technique for eliminating most synchronous writes in the Fast Filesystem," In Proceedings of the FREENIX Track, 1999 USENIX Annual Technical Conference, 1999.

[3] Elizabeth Shriver, Eran Gabber, Lan Huang, and Christopher A. Stein, "Storage management for web proxies," In Proceedings of the Annual USENIX Technical Conference, 2001.

[4] Evangelos P. Markatos, Manolis G.H. Katevenis, Dionisis Pnevmatikatos, and Michail Flouris, "Secondary storage management for Web proxies," In Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems (USITS99), 1999.

[5] Gregory R. Ganger, Bruce L. Worthington, and Yale N. Patt, "The DiskSim Simulation Environment Version 1.0 Reference Manual," Technical report CSE-TR-358-98. Department of Computer Science and Engineering, University of Michigan, February 1998.

[6] Jeffrey C. Mogul, "Speedier Squid: A Case Study of an Internet Server Performance Problem," ;login: The USENIX Association Magazine, 24(1):50-58, 1999.