# I/O 시간 중첩에 의한 웹 캐쉬 서버 성능 향상에 관한 연구

이대성° 김기창
인하대학교 전자계산 공학과
{blue, kchang}@super.inha.ac.kr

## A Study on Improving Cache Server Performance through I/O time Overlapping

Dae-Seong Lee Ki-Chang Kim
Dept of Computer Science and Engineering, In-Ha University

요 약

인터넷 사용자의 급격한 증가로 인한 네트웍 응답 시간의 지연이 가속화되고 있으며 이에 대한 대책으로 캐쉬 서버(프락시 서버)의 사용이 대두되고 있다. 그러나 캐쉬 서버의 사용은 처리 요청의 집중에 따른 또 하나의 병목 현상을 일으킬 수 있다. 이에 따라 다중 프락시 서버를 이용하는 연구들이 활발히 진행되어 왔으나 기존의 연구들은 분산 알고리즘을 수행하는 다중 프락시 서버에 편중되어 있으며 캐쉬 서버가 일반 웹 서버와 다른 점(웹 오브젝트를 디스크에 저장하는 일)을 간과하여 캐쉬 서버 자체의 성능을 효과적으로 개선하지 못하고 있다. 따라서, 요청 폭주 시에 캐쉬 미스 등의 처리에 있어서 비효율성을 나타내고 있다. 본 연구에서는 병목 현상을 일으키는 요인을 분석하고 이를 바탕으로 프락시 서버를 재구성하여 실험을 통해 이 시스템의 효율성을 분석한다.

## 1. 서론

1990년대 이후로 인터넷 사용자는 꾸준히 증가하고 있으며 특히 최근 몇 년 동안 급격히 증가하는 추세이다. 이에 따른 네트웍 트래픽의 증가로 패킷 교환 시 일어나는 충돌에 의한 속도 저하와 서버 시스템의 오버헤드로 인한 속도 저하가 발생하게 되어 결국 인터넷 서비스의 정체 현상이 가속화 되고 있다.[1]

웹 캐싱 기술은 이러한 문제를 해결하는 경제적이고 효율적인 방법중의 하나이다. 웹 캐싱 기술을 사용함으로써 사용자의 서비스 요청에 대한 응답시간을 줄일 수 있고 실제 서버로의 요청을 지역 네트웍 안에서 해결함으로써 서버에 대한 부하감소와 전체 네트웍 트래픽 또한 줄일 수가 있다.[2]

또한 최근의 연구분석에 따르면 서비스 요청은 일부 핫 오브젝트에 편중되고 있으며 요청 페이지의 사이즈도 증가하고 있는 추세이다. 이는 웹 캐싱 기술의 개발을 더욱 부각시키고 있다.[3]

이에 발맞추어 웹 캐쉬 서버의 개발이 활발히 진행되고 있으나 단일 웹 캐쉬 서버를 사용할 경우 저장 용량에 한계가 있을 뿐만 아니라 서비스 가능한 클라이언트의 수 또한 제한을 받게 된다.

이러한 문제를 해결하기 위해 많은 연구들이 수행되어 왔으며 그 대표적인 예로는 ICP(Internet Cache Protocol), CARP(Cache Array Routing Protocol), Summary Cache 그리고 리다이렉션 메시지를 이용하는 방법 등이 있다.[6,8,9]

위의 연구들은 저장 용량의 한계를 극복하고는 있으나 분산 알고리즘을 수행하는데 비중을 두고 있기 때문에 캐쉬 서버 자체의 성능을 개선하고 있지는 않다. 이는 캐쉬 서버가 일반 웹 서버와 다른점(웹 오브젝트를 디스크에 저장하는 일)을 간과하고 있기 때문이다.

본 연구에서는 캐쉬 서버가 병목 현상을 일으키는 요인을 분석하고 이를 바탕으로 I/O 시간 중첩에 의한 프락시 서버를 구성하여 높은 요청률과 캐쉬 미스시에도 효과적으로 대처할 수 있는 시스템을 제안하고 실험을 통해 효율성을 분석한다

## 2. 관련 연구

캐쉬 용량이 무한하다면 캐쉬 서버의 적중률은 그 서버가 서비스하는 사용자수에 비례한다는 연구 보고가 있다.[4,5] 이러한 캐쉬 용량 문제를 해결하기 위해 많은 연구들이 진행되어 왔으며, 그 중 대표적인 연구들은 다음과 같다.

### 2.1 ICP(Internet Cache Protocol)

ICP는 현재 Squid 서버, CERN 서버 및 상용 서버에서 실제로 사용되는 프로토콜로 계층적인 구조를 갖는다.[6,7]

이 프로토콜은 요청에 대해 캐쉬 미스가 발생하면 부모 캐쉬나 형제 캐쉬에 해당 데이터가 있는지 묻게 되고 일정 시간이 지난 후 응답이 없으면 원격 호스트에 데이터를 요청하게 된다. ICP는 캐쉬 용량 문제를 해결하고는 있지만 캐쉬 데이터의 중복 가능성이 높고 계층적인 구조로 되어 있기 때문에 신선도가 떨어지는 자료들이 신선한 자료로 오인될 수 있는 단점을 가지고 있다.

### 2.2 Summary Cache

Summary Cache는 ICP에서의 네트웍 트래픽 증가를 감소시키기 위해 착안된 방법으로 같은 디렉토리내에 포함된 모든 프락시 서버들이 각 프락시 서버들의 데이터 유지 정보인 Summary(일종의 테이블)를 갖고 있다. 이 경우는 캐

쉬 미스가 발생하면 갖고 있는 Summary를 통해 요청 데이터가 어느 프락시 서버에 있는지 찾아내고 해당 프락시 서버에서 데이터를 가져오게 된다. Summary Cache는 Summary의 내용을 매번 갱신하지 않고 일정 비율만큼의 내용 변화가 발생했을 때 갱신하기 때문에 false misses와 false hits이 발생하고 불필요한 inter-proxy 트래픽을 유발하는 단점이 있다.[8]

### 2.3 CARP(Cache Array Routing Protocol)

CARP는 저장장치의 효율적인 이용에 초점을 맞춘 방법이다. CARP는 약 결합(loosely coupled)된 프락시 서버들의 리스트와 해쉬 함수로 구성이 된다. CARP를 지원하는 서버는 요청이 들어오면 URL과 각 프락시 서버의 식별자를 해쉬 함수를 통해 해당 URL에 대해 각 서버들의 우선권을 계산하게 되고 우선권이 가장 높은 프락시 서버에서 해당 URL을 처리하게 한다.[9]

CARP의 경우에는 요청 빈도수가 높은 데이터를 갖는 프락시 서버로 보다 많은 요청이 가기 때문에 해당 프락시 서버에 오버헤드가 발생할 수 있으며 각 프락시 서버들이 약 결합으로 구성되어 있기 때문에 데이터의 관리가 그다지 용이하지 못하다.

위의 세 가지 방법들은 기존에 존재하는 프락시 서버들을 연계하는 방법이기 때문에 각 프락시 서버마다 사용자의 수와 이용빈도가 다르다. 따라서 동시에 많은 데이터를 요청하는 경우 특정 서버에 상당한 오버헤드가 발생하게 된다. 결국 프락시 서버 자체의 성능을 효과적으로 개선하지 않는다면 높은 요청률에 대해 능동적으로 대처할 수 없다.

본 연구에서는 이러한 문제점을 해결할 수 있는 방법을 제시하고자 한다

## 3. I/O 중첩을 이용한 프락시 서버

I/O 중첩을 이용해 시스템의 성능을 향상시키고자 하는 연구가 진행되고 있다. 일반적인 서버들의 공통적인 특징은 다수의 사용자 요청을 신속히 처리해야 하고 시스템의 안정성을 제공해야 한다. 그러나 다수의 사용자 요청이 폭주할 경우 디스크 I/O, 네트웍 I/O에 의한 병목 현상이 발생하게 되어 시스템의 성능이 저하된다.[10]

본 연구에서는 이러한 병목 현상을 개선하는데 주력하였으며 Squid(Proxy Server Program)와 Polygraph(Cache Server Benchmarking Program)을 이용해 실험을 하였다.

### 3.1 Squid(Proxy Server Program)

Squid는 Harvest Project의 일환으로 개발된 소프트웨어로 현재 무료로 배포되고 있다. Squid의 주요 동작원리와 문제점을 살펴보면 다음과 같다. 아래의 알고리즘은 Squid-2.3.STABLE3를 분석한 내용이다.

#### 3.1.1 Squid 알고리즘

```
while(1) {
 if (client의 service 요청 도착){
  • 요청 데이터의 URL과 METHOD를 MD5 해쉬를 이용해 key
    값 추출
  • 추출된 key값을 갖고 있는 StoreEntry를 검색
    if (StoreEntry exists) {  // Cache Hit
       if (in Memory)
         • Memory Read후 client에게 service
       else if (in Disk)
①        • Disk Read후 client에게 service
    }
    else {  // Cache Miss
       • origin server에게 해당 데이터 요청
       do {
        • origin server로부터 해당 데이터 도착
        • 해당 data를 Memory에 load
              • Memory에 load된 양만큼 client에게 service
②      • 해당 데이터를 Disk에 Write
       } while(해당 데이터 도착 미완료)
    } // else
  } // if
} // while
```

#### 3.1.2 Squid의 문제점

Squid의 문제점은 클라이언트의 요청이 폭주할 경우 Squid 알고리즘의 ①, ②에서 Disk I/O에 병목 현상이 발생하게 되어 시스템의 성능을 저하 시키고 있다. 이는 공식적인 프락시 성능 테스트 프로그램인 Polygraph를 통하여 확인할 수 있었다. Disk I/O에 걸리는 병목 현상은 Squid뿐만이 아니라 일반적인 서버에서 나타나는 시스템 성능 저하의 주 요인이다.

본 연구에서는 Disk I/O에 걸리는 시간을 중첩하여 시스템의 성능을 향상시키는데 주력하였다.

#### 3.1.3 Disk I/O 시간 중첩에 의해 개선된 알고리즘

알고리즘1. MASTER

```
while(1) {
 if (client의 service 요청 도착){
  • 요청 데이터의 URL과 METHOD를 MD5 해쉬를 이용해 key
    값 추출
  • 추출된 key값을 갖고 있는 StoreEntry를 검색
    if (StoreEntry exists) {  // Cache Hit
       • Memory Read후 client에게 service
    }
    else {  // Cache Miss
       • SLAVE server에게 해당 데이터 요청
       do {
        • SLAVE server로부터 해당 데이터 도착
        • 해당 data를 Memory에 load
        • Memory에 load된 양만큼 client에게 service
       } while(해당 데이터 도착 미완료)
    } // else
 } // if
} // while
```

알고리즘2. SLAVE

```
while(1) {
 if (MASTER의 service 요청 도착){
  • 요청 데이터의 URL과 METHOD를 MD5 해쉬를 이용해 key
    값 추출
  • 추출된 key값을 갖고 있는 StoreEntry를 검색
    if (StoreEntry exists) {  // Cache Hit
       if (in Memory)
         • Memory Read후 MASTER에게 service
       else if (in Disk)
①        • Disk Read후 MASTER에게 service
    }
    else {  // Cache Miss
       • origin server에게 해당 데이터 요청
       do {
        • origin server로부터 해당 데이터 도착
        • 해당 data를 Memory에 load
        • Memory에 load된 양만큼 MASTER에게 service
②      • 해당 데이터를 Disk에 Write
       } while(해당 데이터 도착 미완료)
    } // else
 } // if
} // while
```

위 알고리즘의 핵심은 Squid의 병목 현상을 유발하는 Disk I/O를 SLAVE에게 위임하여 MASTER의 CPU 시간을 중첩함으로써 시스템의 효율성을 개선하고자 하는 것이다.

### 3.2 Polygraph

Polygraph는 캐쉬 서버의 성능을 테스트하기 위해 개발된 공인된 프로그램이다. Polygraph는 실제 인터넷 현황에 유사하도록 만들어 졌으며 클라이언트가 응답을 받으면 데이

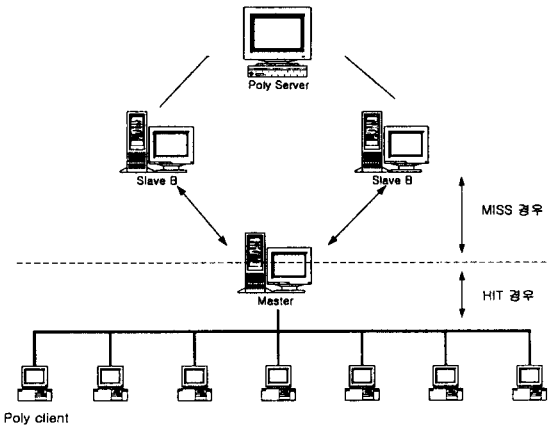
터를 요청하는 Best-Effort Model과 응답을 받지 않더라도 일정 시간 동안 지속적으로 데이터를 요청하는 Constant-Rate Model을 제공한다.[11]

## 4. 실험 및 평가

본 장에서는 실험을 통해 Disk I/O 시간 중첩에 의한 시스템의 성능 향상을 검증한다.

### 4.1 시스템의 구성

3.1.3절에서 언급한 개선된 Squid의 실험을 위한 시스템 구성은 [그림 1]과 같다.

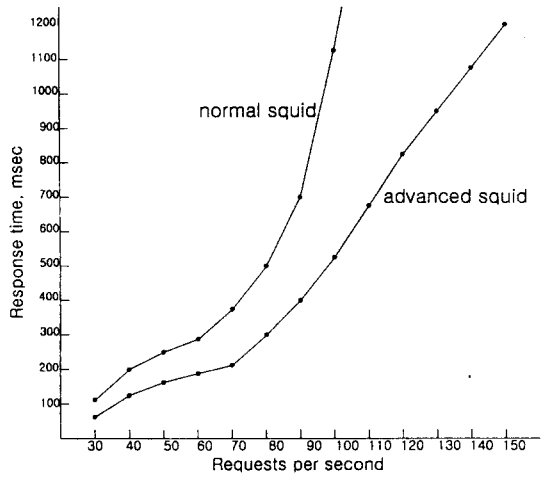



[그림1] 시스템 구성도

MASTER의 시스템 사양은 PENTIUM-III 600MHz, 512MB MEMORY이었고 SLAVE A는 PENTIUM-III 700MHz, 768MB MEMORY, SLAVE B는 PENTIUM-II 400MHz, 256MB MEMORY였다. 모든 실험은 100Mbps의 대역폭에서 실행하였다.

### 4.2 실험 결과

[그림 1]과 같이 구성된 시스템상에서 공식적인 캐쉬 서버 성능 테스트 프로그램인 Polygraph를 사용하였으며 트래픽 모델은 Constant-Rate Model을 적용하였다. 이 실험의 측정 결과는 [그림 2]와 같다.




[그림2] 성능 테스트 비교

이 실험의 결과를 살펴 보면 정상적인 Squid는 [그림2]에서 알 수 있듯이 요청률이 증가함에 따라 응답 시간이 현저하게 지연되고 있으나 Disk I/O 시간 중첩을 이용한 Squid는 정상적인 Squid에 비해 성능이 향상되고 있음을 보여주고 있다.

## 5. 결론 및 향후 연구 과제

본 논문에서는 Disk I/O 시간 중첩에 의한 캐쉬 서버의 성능 향상을 제안하였다. 제안한 Squid는 병목 현상을 유발하는 Disk I/O를 SLAVE에게 위임함으로써 시스템이 블록 상태에 빠지는 일이 없도록 유도하였다. 이로 인해 다수의 클라이언트 요청이 폭주하더라도 응답 시간의 지연을 개선할 수 있었다.

향후 연구 과제로는 동일 시스템 내에서 Disk I/O 시간을 중첩할 수 있는 방안이 필요하며 캐쉬 서버뿐 아니라 일반적인 서버들에게서 나타나는 Network I/O 블록 현상을 해결할 수 있는 연구들이 필요 할 것이다.

## 참고 문헌

[1] Radhika Malpani, Jacob Lorch, and David. Berger University of California at Berkeley, "Making World Wide Web Caching Servers Cooperate". http://bmrc.berkeley.edu/papers/1995/138/paper-59.html, 1995

[2] Van Jacobson. "How to kill the internet", In SIGCOMM'95 Middleware Workshop, August 1995. ftp://ftp.ee.lhl.gov/talks/vj-webflame.ps.Z

[3] John Dilley. Hewlett-Packard Laboratories, " Web Server Workload Characterization"

[4] Bradley M. Duska, David Marwood, and Michael J. Feeley. "The measured access characteristics of world-wid-web client proxy caches", In Proceedings of USENIX Symposium on Internet Technology and Systems, December 1997

[5] Steve Gribble and Eric Brewer. "System design issues for internet middleware service : Deduction from a large client trace", In Proceedings of USENIX Symposium on Internet Technology and Systems, December 1997

[6] Duane Wessels and k claffy. "ICP and the Squid Web Cache", August 13, 1997

[7] Anawat Chankhunthod, Peter B. Danzig and Chuck Neerdaels. Computer Science Department Univesity of Southern California, "A Hierarchical Internet Object Cache"

[8] Li Fan, Pei Cao, Jussara Almeida and Andrei Z. Broder. "Summary Cache : A Scalable Wide-Area Web Caching Sharing Protocol". http://www.cs.wisc.edu/~cao/papers/summary-cache/sharc.html, 1998

[9] Microsoft Corporation, CARP White Paper, http://www.microsoft.com/proxy/beta/moreinfo.html, 1997

[10] Sachin More and Alok Choudhary. "MTIO, A MULTI-THREADED PARALLEL I/O SYSTEM", In Proceedings of 11th International Parallel Processing Sysposium, pages 368-373, April 1997

[11] Alex Rousskov and Duane Wessels. "High Performance Benchmarking With Web Polygraph" http://polygraph.ircache.net/doc/papers/paper01.ps.gz, June 2000