Memory Mapped File을 이용한 웹 서비스 성능향상 연구

이형봉°, 최형진, 차홍준
호남대학교 소프트웨어공학과°, 강원대학교 컴퓨터과학과
hblee°@honam.ac.kr, {choihj, tchahj}@kangwon.ac.kr

A Study on Performance Improvement of Web Service Using Memory Mapped File

Hyung Bong Lee°, Hyung Jin Choi, Hong Jun Tcha
Dept. of Software Engineering, Honam Univ.°, Dept. of Computer Science, Kangwon Univ.

요 약

웹 서버의 처리 용량을 향상시키기 위한 가장 보편적인 접근방안은 서버 시스템을 중복하여 여러 개 설치하는 것이다. 하지만 이러한 물리적인 증설은 확장이 용이한 반면 비용이나 공간 등의 측면에서 큰 부작용을 동반하기 마련이다. 이와 같은 물리적인 확장 외에 운영체제 환경이나 응용 프로그램의 작은 개선에 의해서도 뜻밖의 큰 성능효과를 얻을 수 있는 경우가 있다. 이 논문에서는 이미지나 데이터 파일 등 고정된 파일을 내려 받을 때 웹 서버에서 일반 파일 대신 Memory Mapped File 입·출력을 사용한 웹 서비스 성능 향상 방법을 설계·구현하고 이를 통해서 얻어지는 이득을 실 서비스 환경에서 검증한다.

1. 서 론

웹 서비스는 시간과 공간을 초월하여 거의 무한대의 사용자를 대상으로 이루어지기 때문에 이를 지원하는 정보시스템의 처리용량 또한 끝없는 증설요구에 직면하는 경우가 많다. 이런 문제를 해결하기 위한 가장 보편적인 접근방안은 서버 시스템을 중복하여 여러 개 설치하는 것이다. 그러나 이러한 물리적 더하기식의 확장 방안은 설치자체는 간편해 보이지만, 부하분산, 데이터 일관성, 설치비용, 설치공간 등 많은 부작용을 동반한다[1,2]. 반면에 운영체제, 데이터베이스, 응용 프로그래밍 기법 등 소프트웨어 측면에서의 작은 개선으로 의외의 큰 효과를 얻는 경우도 있다[3,4]. 이는, 사소한 부주의나 습관에 의한 고정관념으로 인해 막대한 컴퓨팅 파워가 허비될 수도 있다는 것을 의미하기도 한다.

Memory Mapped File(MMF)은 간과되기 쉬운 대표적인 웹 서비스 성능향상 방안 중의 하나이다. MMF는 이미지나 데이터 파일 등 고정된 파일을 웹으로 서비스할 때 파일과 네트워크에 대한 입·출력을 담당하는 운영체제가 차지하는 커널 영역과 응용 프로그램인 웹 서버가 점유하는 사용자 공간 사이에서의 메모리 복사를 줄임으로써 컴퓨팅 파워를 절약하는데 적용될 수 있다[5].

이 논문에서는 MMF를 이용한 웹 서버의 성능향상 가능성을 모색한다. 이를 실증적으로 접근하기 위하여, 웹 서버를 직접 수정하지는 않고, 단순 CGI를 개선한 Fast CGI[6] 파일서버를 구현하여 MMF의 효용성을 검증한다. 즉, 파일서버를 보편적인 파일 입·출력과 MMF에 의한 입·출력 등 두 가지 방법으로 구현하여 각각의 웹 서비스 성능을 측정하고 그 결과를 분석한다.

2. Memory Mapped File(MMF) 고찰

2.1 MMF의 개념

MMF는 그림 1과 같이 디스크 등 저장장치에 저장된 파일의 공간을 사용자 프로그램의 가상 주소공간으로 사상시켜, read()나 write() 등 입·출력 시스템 콜 대신 대응된 주소부위를 참조함으로써 실질적인 입·출력 효과를 얻을 수 있는 운영체제 서비스 중의 하나이다[5].

그림 1에서 mmap 영역은 사용자 영역에서 주소공간만 차지할 뿐 물리 메모리는 할당되지 않고 단지 커널 영역의 디스크 공간(디스크 캐시)으로의 대응 역할만 담당한다. 따라서 mp 부분에서 buf 부분으로의 복사는 커널 영역의 파일내용을 사용자 영역의 배열로 읽는 효과와 동일하다. 이 과정을 copyout이라 하는데 보통의 메모리 복사보다 더 많은 시간을 필요로 한다(반대 과정은 copyin 임).

Mmap 부분을 공유메모리 영역(shared memory region)에 설정하고, 그 공유메모리를 프로세스들이 공유할 경우 여러 프로세스가 파일을 공유하는 효과를 얻을 수 있기 때문에 다중 파일서버에 의한 파일 서비스 환경구축에도 편리하다.

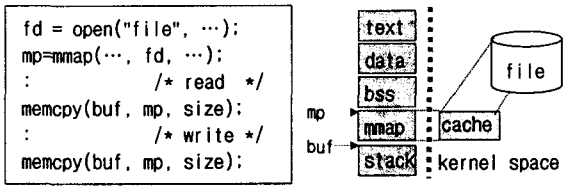

그림 1 Memory Mapped File(MMF)의 개념

2.2 일반 파일 서버와 MMF 서버의 비교

일반적인 파일 서비스에서는 그림 2와 같이 커널 영역의 파일 버퍼 캐시로부터 사용자 영역으로 데이터 복사가 일어난 후(① copyout), 다시 사용자 영역으로부터 소켓 버퍼가 있는 커널 영역으로의 데이터 복사가 일어난다(②copyin). 이는 웹 서버가 일단 파일 데이터를 얻은 후 그것을 네트워크 쪽으로 보내야 하는 전달자 역할을 하기 때문에 불가피하다.

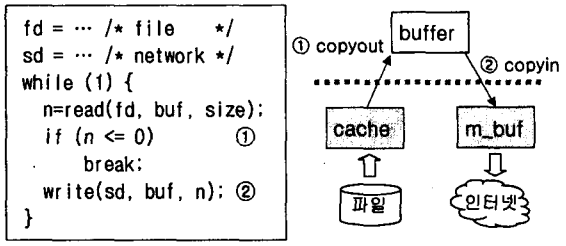

그림 2 일반 파일 서비스에서의 데이터 흐름

반면에 MMF를 사용할 경우에는 그림 3과 같이 사용자 영역으로의 데이터 복사가 필요하지 않다. 즉, MMF는 사용자 프로그램에게 커널 영역의 데이터에 대한 포인터(① mp)를 알려주는 형식이기 때문에, 웹 서버는 커널 영역의 데이터를 복사하여 전달하는 대신, 데이터에 대한 포인터만 전달하면 된다. 따라서 실질적인 데이터 복사는 커널 영역에서 단 한 번만 일어난다(② bcopy). 이는 복사 회수도 줄어들 뿐 아니라, 사용자와 커널 영역 사이를 건너야 하는 별도의 부담도 절약할 수 있는 이득을 준다.

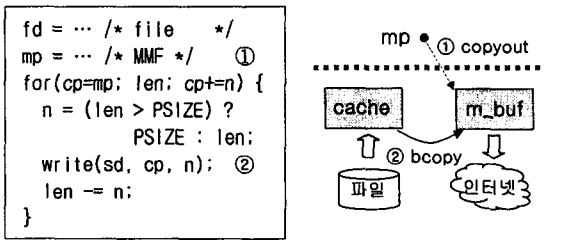

그림 3 MMF 서비스에서의 데이터 흐름

3. MMF를 이용한 웹 파일 서버 구현

그림 5에 MMF를 사용한 웹 서버의 파일 서비스 루틴을 제시하였다. 이 루틴은 FTABSIZE 만큼의 파일에 대한 MMF를 설정하고 있다가 파일 요청이 있을 때 사용자 영역의 버퍼를 경유하지 않고 곧바로 커널 내 복사에 의해 파일 내용이 네트워크 버퍼로 이동될 수 있도록 한다.

이 루틴은 FastCGI API에 의해 파일 서버로 독립되어 디몬으로 동작하면서 그림 4와 같이 웹 서버와 연동한다.

그림 4 FastCGI 파일 서버 개념

```
#include <sys/fcntl.h>
#define PACKSIZE  1024
#define FTABSIZE  32
struct ftable {
    char          fname[32];
    int           fd;
    char          *mp;
    int           size;
}  Ftab[FTABSIZE];
char     *Mmap();

srv_file(int sd, char fname[])
{
  int    fd, fn, ft, size, n;
  char   *mp, *cp;
  write(sd, "Content-Type: text/plain; "
                    "charset=euc-krWnWn", 42);
  for (fn = 0, ft = -1; fn < FTABSIZE; fn++) {
    if (Ftab[fn].fd >= 0) {
      if (!strcmp(Ftab[fn].fname, fname))
        break;
    }
    else if (ft < 0)       /* empty slot */
      ft = fn;
  }
  if (fn < FTABSIZE) {
    mp = Ftab[fn].mp;
    size = Ftab[fn].size;
  }
  else {
    if ((fd=open(fname,O_RDONLY)) < 0) { /*error*/ }
    Ftab[ft].fd = fd;
    Ftab[ft].mp = Mmap(fd, &size);
    Ftab[ft].size = size;
    strcpy(Ftab[ft].fname, fname);
  }
  for (cp = mp; size > 0; size -= n, cp += n) {
    n = size > PACKSIZE ? PACKSIZE : size;
    write(sd, cp, n);
  }
  close(sd);
}

#include <sys/mman.h>
#include <sys/stat.h>
char *Mmap(int fd, int *len)
{
  struct  stat st;  char   *mp;
  fstat(fd, &st);
  mp = mmap(NULL, st.st_size, PROT_READ,
          MAP_FILE | MAP_VARIABLE | MAP_PRIVATE,
             fd, 0);
  if (mp == (caddr_t)-1) {  /* error */  }
  *len = st.st_size;
  return(mp);
}
```

그림 5 MMF를 이용한 파일 서비스 루틴

4. MMF를 이용한 웹 서버의 성능 분석

4.1 시험환경 및 성능측정 방법

이 논문에서 MMF를 사용한 웹 서비스 성능측정은 표 1의 환경에서 이루어졌다. 이 표에서 FastCGI[6]는 앞의 그림 4와 같이 단순 CGI를 개선하여 특정 게이트웨이를 디몬으로 생성할 수 있는 인터페이스를 제공한다.

그림 3과 5의 MMF 파일 서비스 프로그램과 그림 2의 일반 파일 서비스 프로그램을 FastCGI의 게이트웨이로 등록하여 파일 서버로 동작시킨 후 각각에 대한 서비스 성능을 측정하여 비교하였다.

파일을 내려 받는 시간을 측정하기 위하여 소켓 프로그램으로 작성된 HTTP 1.1 요구 패킷 발생기를 사용하였고, 1K~1M의 다양한 크기의 파일에 대하여 각각 1000번의 요구 패킷을 쉬지 않고 보내도록 하였다. 이러한 시도를 5회 실시하여 그 평균값을 최종 측정값으로 사용하였다. 또한 이 논문의 주요 목적이 메모리 복사에 따른 미세한 시간 차이를 측정하는데 있으므로 네트워크의 영향을 배제하기 위하여 로컬 네트워크를 사용하였다.

표 1 MMF 웹 파일 서비스 성능측정 환경

항 목	사 양
시스템 모델	COMPAQ DS20
CPU	Alpha Ev67 667MHz
메모리	512M
운영체제	Digital Tru64UNIX 4.0F
웹 서버	Apache version 2.0.47
FastCGI	Version 2.4.0

4.2 측정결과 분석

그림 6에 일반 파일 서비스와 MMF에 의한 파일 서비스에 대한 성능측정 결과를 보였고, 그림 7에는 그림 6의 미세한 부분을 확대하여 보였다. 이들 두 그림으로부터 다음의 두 사실을 확인할 수 있다.

- 측정 대상인 모든 크기의 데이터에 대하여 MMF를 이용한 파일 서비스 성능이 우수하다.
- 요구 파일의 크기가 클수록 얻어지는 시간적 성능 이득은 커진다.

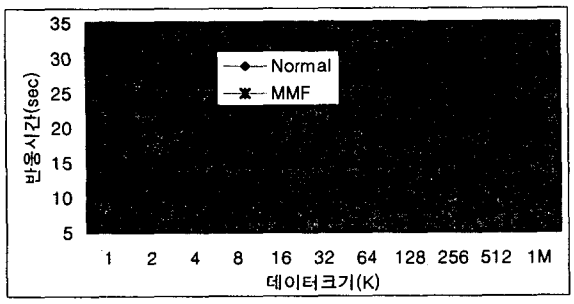

그림 6 일반 파일과 MMF에 의한 파일 서비스의 성능

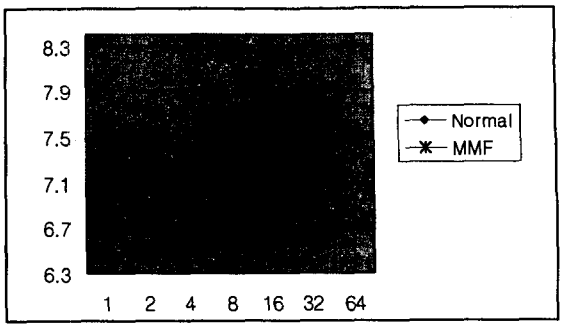

그림 7 그림 6의 미세한 부분의 확대

5. 결 론 및 향후 연구

그림 6과 7로부터 일반 파일 서비스보다 MMF에 의한 파일 서비스의 성능이 우수하고, 그 주된 요인이 메모리 복사의 절약에서 기인함을 보였다. 이러한 성능상의 이득은 파일 서비스가 이루어질 때마다 누적될 것이기 때문에 이것이 시스템의 처리능력에 미치는 영향이 크지 않을 수 없다. 예를 들어 그림 6에서 크기 1M인 파일의 경우 하루 동안 MMF 파일 서비스에 의해 321178번을 더 내려 받을 수 있다.

이번 연구는 단일 요구 발생기에 의한 1:1 모델이라는 다소 제한적인 환경에서 이루어졌다. 그런데, MMF의 공유에 의한 다중 서버 환경에서 얻어지는 한계 이득은 더욱 커질 것으로 예상되기 때문에, 실 환경에 보다 더 가까운 다중 사용자 모델에서의 파일 서버 구축 방안에 대한 연구가 계속될 것이다.

참고문헌

[1] 김성수, 정지영, "웹 서버 클러스터를 위한 효율적인 부하 분배 알고리즘", 정보과학회 논문지: 정보통신 제28권 제4호,pp.550-558, 2001.12

[2] Arun Iyengar, Jim Challenger, Daniel Dias, Paul Dantzig, "High-Performance Web Site Design Techniques", IEEE Internet Computing, March April 2000

[3] Erich Nahum, Tsipora Barzilai, Dilip Kandlur, "Performance Issues in WWW Servers", IEEE Transactions on Networking,10(2):2-11, Feb 2002

[4] 이기용, 곽태영, 서정현, 김명호, "대규모 온라인 검색 요구를 효율적으로 처리하기 위한 KRISTAL-II 웹 게이트웨이의 설계 및 구현", 정보과학회 논문지: 컴퓨팅의 실제 제6권 제5호, pp.496-504, 2000.10

[5] Digital, "mmap()", Digital UNIX Reference Page Section 2:System Calls, pp.243-249, March 1996

[6] Open Market, "FastCGI", http://www.fastcgi.com