웹 서버 성능 가속기

조 준 우, 최 현 진, 박 규 호
한국과학기술원 전자전산학과

Web-server accelerator

- CDA (Contents Delivery Accelerator)

Joon-Woo Cho, Hyun-Jin Choi, Kyu-Ho Park
School of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology
e-mail : {jwc, yhchoi, khpark}@core.kaist.ac.kr

Abstract

Current web-server deals a multimedia data as well as text data. But dealing a multimedia data is high burden to web-server. So it can degrade web-server response. We introduce H/W feature CDA (Contents Delivery Accelerator). Main function of this H/W is transferring data between SCSI disk and NIC by direct path, and TCP offloading. These 2 functions can accelerate web-server performance. In this paper we will explain problem of current web-server and suggest our new architecture and say various implementation issues.

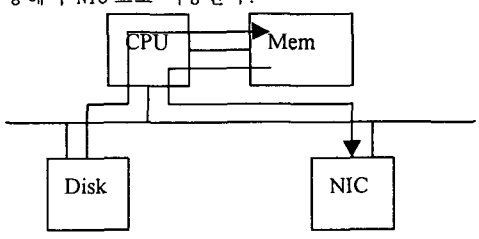
I. 서론

현재의 웹 서버는 텍스트 데이터 뿐만 아니라 많은 용량의 멀티미디어 데이터까지 다루고 있다. 이로 인해 host CPU bus 에 많은 부하가 걸리게 되고 결과적으로 서버 시스템의 응답 속도를 떨어뜨리게 된다. 현재의 웹 서버가 멀티미디어 데이터를 효과적으로 다루지 못하는 이유는 disk 와 NIC 사이의 많은 중복된 복사와 TCP processing overhead 이다.

이러한 문제를 해결하기 위해 CDA 라는 보조 HW 를 제안한다. 이 HW 는 host computer 의 PCI slot 에 꽂는 형태이며 다음의 두 기능을 행한다. 첫째로 disk 와 NIC 사이에 direct 한 path 를 제공함으로서 불필요한 데이터 복사를 줄이고 둘째로 TCP offloading 기능을 제공함으로서 host CPU 의 부담을 줄인다. 또한 이 HW 를 사용하더라도 OS 의 내부만 고쳐서 현재의 application 을 그대로 사용할 수 있도록 하였다.

II. 현재의 서버 시스템

보통의 멀티 미디어 서버는 범용 OS 를 사용해서 동작하고 있다. 그러나 이런 범용 OS 는 멀티 미디어 데이터를 효과적으로 처리해 주지 못하고 있다. 범용 OS 에서 disk 의 데이터는 아래 그림과 같은 경로를 통해서 NIC 으로 이동한다.



본 논문은 국가지정연구실(NRL) 사업의 지원 하에 이루어졌음.

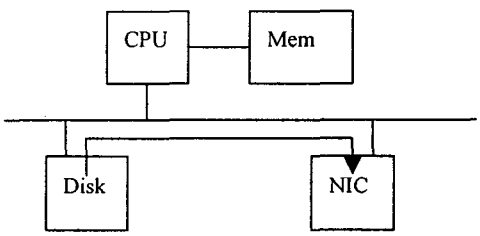
Host CPU가 disk에 데이터를 요청하면, 데이터는 PCI bus 와 local bus 를 거쳐서 Memory 로 전달되고 이 데이터는 다시 PCI bus 와 local bus 를 거쳐서 NIC 에 전달되게 된다.[1] 이런 중복된 동작이 전체 데이터 전송이 끝날 때까지 계속된다. 보통 멀티미디어 데이터를 다루는 서버는 working-set 이 크기 때문에 캐쉬 메커니즘의 이득을 텍스트 데이터를 주로 다룰 때처럼 볼 수 없음을 생각한다면 이는 서버의 성능에 큰 영향을 주는 요인이 될 수 있다.

또한 TCP protocol processing 도 서버의 성능을 저하시키는 요인이 된다. TCP processing 은 많은 부하가 걸리는 작업이므로 CPU clock 을 많이 소모하게 되어 web-server 의 전체 performance 를 떨어뜨리게 된다. 앞으로의 Gigabit 환경에서는 이것이 더욱 큰 문제가 될 것이다.[2]

III. CDA(Contents Delivery Accelerator)

3.1 CDA Overview

앞에서 기술한 문제들을 해결하기 위해 CDA 구조를 제안한다. 성능향상을 이루는 동시에 현재의 application 이 수정 없이 돌아갈 수 있도록 호환성을 유지하기 위해 노력하였다. CDA 의 주된 성능 향상은 disk 와 NIC 사이의 direct path 에서 이루어진다. 전체적인 operation 은 아래 그림과 같다.



여기서 host CPU 는 처음에 데이터 전송을 초기화 시켜 주는 역할만 행한다. 이후 데이터는 host CPU 와 local bus 의 관여 없이 motherboard 의 DMA engine 에 의해 disk 에서 NIC 으로 전송된다. 또한 PCI bus 의 사용량도 앞의 operation 과 비교해서 반으로 줄어든다. 이런 direct path 를 실제 시스템에서 적용하기 위해 'sendfile' system call 을 고친다. 'sendfile' call 은 디스크에서 읽힌 데이터가 결국 NIC 으로 가게 되는 것을 의미한다. 따라서 이 system call 을 사용하면 host 는 데이터 흐름을 명확하게 알 수 있으므로 이때 CDA 는 direct path 와 TCP offloading 기능을 적용할 수 있다.

3.2 Direct data path

CPU 가 data 를 disk 에서 NIC 으로 보내라는 명령을 내리면, OS 는 motherboard DMA 엔진에 이 전송을 위한 정보(address translation, file pointer)를 제공하고 DMA engine 은 CPU 와는 독립적으로 데이터를 디스크에서 ■서 NIC 으로 보내고 이 데이터는 CDA 에 의해 결국 네트웍을 통해 바깥으로 보내진다.

이와 같이 direct path 를 사용하면 host CPU 와 local bus 의 부담을 줄어들지만 data 가 하부의 I/O 기기 사이에서만 교환되기 때문에 데이터가 host system cache(main memory, CPU cache)에 저장되지 않는다. 만약 서버의 working-set 이 작다면 이것은 서버의 성능을 크게 감소시키는 요인이 될 수 있다. 따라서 CDA 자체에 일시적으로 데이터를 저장할 수 있도록 memory 를 가져야 한다. 이 memory 는 OS 의 메커니즘을 유지하면서 접근되고 관리되어야 한다. 이에 대한 자세한 사항은 4.2 에서 서술한다.

3.3 TCP- offloading

CDA 의 또 하나의 중요한 기능은 TCP offloading 이다. Direct path 를 사용하면 host CPU 가 외부로 나가는 데이터에 대해 TCP processing 을 해 주지 않으므로 host CPU 를 대신해서 CDA 가 TCP processing 을 수행해야 한다. 이를 위해 OS 가 특정 file 을 disk 에서 읽은 후에 protocol processing 을 하지 않은 상태로 바로 CDA HW 로 전송될 수 있도록 OS 를 고쳐주어야 한다. 이렇게 전송된 raw data 는 CDA 에 장착된 고속의 TCP-offloading chip 에 의하여 빠르게 처리된 후 외부로 전송된다.

IV. 구현

최종적인 CDA system 은 고유의 HW 를 사용해서 구현될 것이다. 현재는 HW 를 제작하기 전에 CDA 구조를 Intel IXP1200[3] 평가보드를 통해서 구조를 검증하고, SW 영역(OS 메커니즘, application)에서 필요한 내용을

적용하는 작업을 진행하는 중이다. IXP 보드에는 IXP1200 network processor, 100/1000 Ethernet, PCI bus connector, memory unit 등이 포함되어 있으므로 CDA 의 동작을 검증하는 용도로는 충분한 기능을 가지고 있다. Target OS 로는 Linux 를 사용하였으며, 커널 버전은 2.4.18 이다.

4.1. Device driver

IXP 보드를 Linux kernel 에 인식시키기 위해 device driver 가 필요하다. 이 driver 의 주된 기능은 IXP 보드의 memory 를 kernel 이 자유롭게 사용할 수 있도록 kernel 의 address space 에 mapping 하는 것이다. 이렇게 mapping 을 하고 나면, kernel 은 IXP 의 32MB memory 를 bus address 를 통해 access 할 수 있게 된다.

4.2. Direct file-transfer

Multimedia application 에서 disk 안의 data 접근은 결국 file 단위로 이루어진다. Linux kernel version 2.4 이후로는 모든 file 에 대한 접근은 disk I/O access 속도를 높이기 위해 page 를 통해 이루어진다.[4] CDA 에서도 빠른 disk access 를 위해 CDA 의 file access 는 Linux 의 page 메커니즘을 사용해서 구현되었다. 이를 위해서 IXP 의 메모리는 페이지 단위로 관리가 되며 디스크에서 나온 데이터는 host memory 의 page 에 들어가는 것처럼 IXP memory 의 page 로 들어가게 된다.

4.3 Address conflict

현재의 computer system 에서 large data 전송은 DMA 를 통해서 이루어지게 된다. DMA 가 되기 위해서는 OS 에서 독립적으로 사용하는 virtual address 에 대응되는 bus address 를 가지고 있는 address table 을 DMAC(DMA Controller)에 보내주어야 한다. Original Linux 에서는 address table 에 IXP memory 의 bus address 를 잘못된 값으로 넣어서 DMAC 가 올바른 데이터 전송을 못하게 된다. 따라서 address table 을 작성하는 함수를 수정하여 address table 에 IXP memory 의 올바른 bus address 를 집어넣을 수 있도록 하였다.

4.4. TCP offloading

IXP 보드에 TCP offloading 을 구현하는 것은 어려운 문제이다. TCP offloading 은 시간이 많이 걸리는 작업이므로 IXP 보드에서 SW 적으로 TCP offloading 기능을 구현하면 서버의 전체적인 성능이 떨어지게 된다. 따라서 이것은 보통 전용의 HW chip 을 통해 구현되며 CDA HW 는 이 chip 을 사용해서 TCP offloading 기능을 추가할 것이다. 그 이전에는 실험을 위해 raw data 와 그것에 대응되는 TCP/IP header 를 disk 에 정적으로 미리 넣어 두었다가 IXP 에 모두 전송시킨 후에 그것들을 사용해서 TCP processing 없이 외부 네트웍으로 전송하는 방법을 택해서 구현 중이다.

V. 해야 할 일

현재 host OS 에서의 코딩 작업은 끝나있고, IXP 보드에 들어갈 firmware 프로그램(패킷 스케줄링, 패킷 출력)을 작성하는 중이다. 이 작업 후에는 CPU, bus utilization 과 max throughput 을 측정해서 제시한 구조의 효용성을 검증하고 문제점을 파악한다. 이러한 SW 작업이 끝나면 현재 제작하고 있는 고유의 CDA HW 를 CDA SW 와 통합시켜서 완전한 시스템을 구현할 계획이다.

VI. 결론

이 논문에서는 멀티미디어 데이터를 다루는 서버의 성능을 높이기 위해 CDA 구조를 제시하였다. 이 구조는 현재 application 과의 호환성을 유지하면서 부하가 많이 걸리는 서버의 성능을 향상시키는 것을 목표로 한다. CDA 의 주된 기능은 disk 와 NIC 사이의 direct path 를 만들어 주는 것과 TCP offloading 이다. 이 두 기능의 효용성을 검증하기 위해 host computer 에 IXP 보드를 붙여서 SW 적으로 구현하였다. 사용된 target OS 는 Linux 커널 버전 2.4.18 이다. 현재 IXP1200 보드에 들어가는 firmware 프로그램을 작성 중이고 이 작업 후에는 고유의 HW 를 사용해서 CDA 의 기능을 완전히 사용할 수 있도록 하려고 한다.

참고문헌

[1] M. Weeks, H. Bataia, R. Sotudeh, "Improved Multimedia Server I/O subsystems", Proceedings of 24th Euromicro Conference, 1998

[2] Evangelos P. Markatos, "Speeding up TCP/IP: Faster Processors are not Enough", IEEE International Performance, Computing, and Communication Conference (IPCCC'02), 2002

[3] Intel Corporation, Intl IXP1200 Network Processor (white paper),2000.
http://developer.intel.com/design/network/products/npfamily/ixp1200.html

[4] Daniel P. Bovet, Marco Cesati, Understating the Linux Kernel 2nd edition, O'Reilly, 2003