네트워크 입출력 처리 성능 향상 기술

박경* 김성운**

인터넷의 보급과 웹 기반 서비스의 등장은 서버의 역할을 연산 및 자료 처리에서 TCP/IP 기반의 네트워크 서비스 제공으로 변화시켰다. 인터넷의 폭발적인 성장과 네트워크 전송 기술의 발전에 따라 서버의 네트워크 입출력 부하는 크게 증가하여 성능 한계점에 이르렀으며, 이로 인하여 다양한 방식의 네트워크 입출력 성능 개선 기술들이 개발되어왔다. 10 Gbps 급 이더넷이 현실화되고 있는 지금 서버 시스템은 날로 증가하는 네트워크 입출력 부하를 처리하기 위하여 연산 능력 향상을 목적으로 추진 하던 기술 개발 방향을 입출력 처리 성능 향상 기술로 전환하고 있으며, 인텔의 I/OAT, 마이크로소프트의 Chimney 등이 이러한 방향을 주도하고 있다. 본 논문에서는 서버의 네트워크 입출력 처리 성능 향상을 위하여 활용되는 기술들을 설명하고 아울러서 네트워크 입출력 처리 성능 개선을 목적으로 개발되는 TCP Onloading 시스템 기술에 대하여 기술한다.

목 차

Ⅰ. 서 론
Ⅱ. TCP/IP 프로토콜 부하 분석
Ⅲ. 네트워크 입출력 처리 성능 향상 기술
Ⅳ. TCP/IP 온로딩 시스템 기술
Ⅴ. 결 론

* 서버플랫폼연구팀/선임연구원
** 서버플랫폼연구팀/팀장

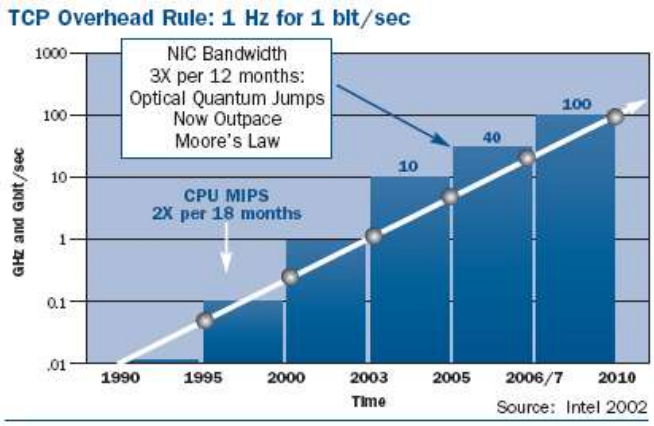
Ⅰ. 서 론

인터넷의 보급과 웹 기반 서비스의 등장은 웹서버, 응용 서버, 데이터베이스 서버, 스토리지 서버 등으로 구성되는 다중티어 방식의 인터넷 서버 구조를 탄생키 켰다. 인터넷을 통한 웹 기반 서비스를 제공하는 인터넷 서버는 현재 단순 웹 서비스는 물론, 전자상거래, 전자정부, ERP(Enterprise Resource Planning), CRM(Customer Relation Management), SCM(Supply Chain Management) 등의 기업전산화 등 다양한 분야에 걸쳐서 사용자에게 서비스를 제공하는 수단으로 활용되고 있다[1].

초고속 인터넷의 보급 및 고도화와 더불어 다양한

웹 서비스의 지속적인 발굴, 웹 기반 기업전산환경의 확산에 따라 인터넷을 통한 서비스 수요가 급증하고 있으며, 이에 따라 서비스 제공을 위한 서버의 성능 요구도 나날히 증가하고 있다. IDC 의 자료[1]에 따르면 2004 년 기준으로 인터넷을 통한 일일 정보 유통량은 약 300 테라바이트에 이르며, 2012 년이 되면 약 10 페타바이트까지 증가할 것으로 예측하고 있다. 이러한 인터넷 트래픽의 증가는 인터넷을 통한 서비스 수요의 증가와 이에 따른 서버의 성능 향상에 대한 요구를 의미하며, 특히 서버의 네트워크 처리 성능은 서비스의 안정성과 품질을 결정짓는 주요한 요소로 작용하게 된다.

인터넷과 웹기술 기반의 서비스를 제공하는 서버의 경우, 현재 네트워크 정합을 위하여 기가비트 이더넷(Gbe)을 사용하며 인터넷 트래픽 처리를 위하여 TCP/IP 프로토콜을 사용하고 있다. 이더넷 기술은 불과 몇 년전까지만 해도 100 Mbps 급을 사용하였으나 , 인터넷의 폭발적인 성장에 따른 고속화 요구와 트래픽의 증가로 인하여 현재 1 Gbps 급을 주로 사용하고 있으며 10 Gbps 규격이 제정된 이후로는 점차 10 Gbps 환경으로 전환되고 있다[2].



(그림 1) 이더넷 성능 발전과 프로세서 성능 발전의 비교

이더넷 전송 기술의 급격한 발전에 따라 이더넷 정합을 통하여 인터넷 기반 웹 서비스를 제공하는 서버 시스템은 증가되는 네트워크 트래픽 만큼 TCP/IP 프로토콜 처리 성능도 이에 발맞추어 향상되어야만 이더넷 고속화에 따른 장점을 얻을 수 있다.

TCP/IP 프로토콜 스택은 전통적으로 운영체제의 일부로서 소프트웨어로 구현되어왔으며, 디바이스 드라이버를 통하여 NIC(Network Interface Controller) 하드웨어와 정합하고 소켓

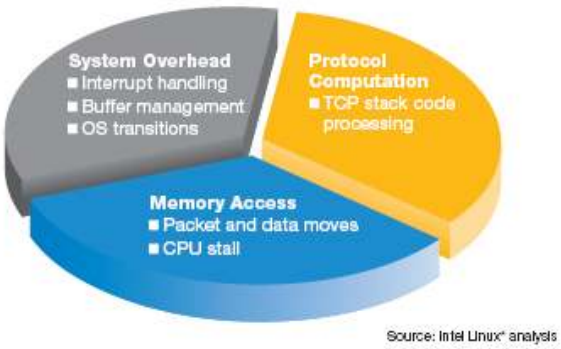
[1] 2003. 6. 2. 디지털타임즈 기사 "서버시장 회복, 리눅스·윈도우가 이끈다" 참조

API(Application Program Interface)를 통하여 응용 프로그램과 정합된다. 운영체제의 일부로서 동작하는 TCP/IP 프로토콜 스택은 현재의 반도체 기술을 기준으로 1 비트의 데이터를 전송하기위해서 1Hz 의 프로세서 성능을 요구된다고 보고되고 있으며, 기가비트 이더넷을 통하여 1 기가비트 데이터를 전송하기 위해서는 1 GHz 의 프로세서 성능을 소비하게 된다[1,3,4]. 현재 표준화가 완료되어 제품 출시가 시작된 10 기가비트 이더넷을 서버에 장착한다면 현재의 반도체 기술로는 구현이 불가능한 10 GHz 급의 프로세서를 사용하거나 여러 개의 프로세서를 사용하는 다중처리형 서버만이 10 Gbps 전송능력을 보장할 수 있다.

본 고에서는 리눅스 운영체제를 토대로 TCP/IP 프로토콜 처리 방식에 대한 간략한 설명과 이더넷 기술의 고속화에 따라 인터넷 기반 서비스에 필수적인 TCP/IP 프로토콜 스택의 성능 향상 기술에 대해서 설명한다.

II. TCP/IP 프로토콜 부하 분석

TCP/IP 프로토콜 처리는 TCP 연결제어, 송신, 수신 등 크게 3 가지의 일로 구분할 수 있다. TCP 연결제어는 통신을 위한 시스템(서버와 클라이언트)간의 연결 설정(Establishment) 및 연결 해제(Teardown) 기능을 처리한다. 통신을 위하여 연결 설정이 완료되면 데이터 전송 기능을 통해서 정보를 교환하게되며, 데이터 전송이 완료되면 연결 해제를 수행하게된다[1].

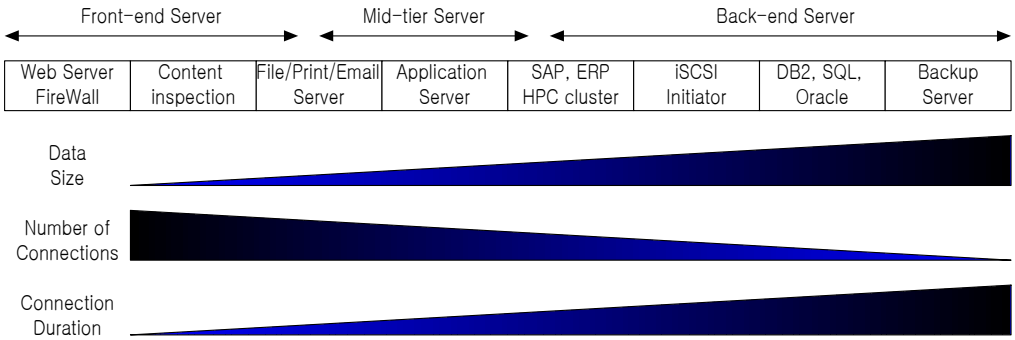


(그림 2) TCP/IP 프로토콜 처리 부하 분석

TCP/IP 프로토콜 처리에 있어서 발생하는 부하를 시스템의 기능별로 분류하면, 인터럽트 및 프로세스(혹은 쓰레드) 관리에서 발생하는 시스템 오버헤드, 프로토콜 처리를 위한 코드를 수행하는 프로토콜 처리 부하, 그리고 페이로드 처리를 위한 메모리 접근 부하로 구분할 수 있다. 패킷 사이즈가

작고 패킷의 개수가 증가할수록 시스템 오버헤드와 프로토콜 처리 부하가 증가하며, 패킷 사이즈가 클수록 페이로드 처리를 위한 메모리 접근 부하가 증가하게 된다[1,5].

인터넷을 기반으로 하는 응용 서비스의 다양화에 따라 응용별로 네트워크 트래픽의 특성이 다르며, 다중티어의 전단부에 위치한 시스템일수록 많은 사용자와 빈번한 접속을 통하여 대량의 소형 패킷을 사용하여 서비스 접근 요청을 처리하며, 후단부에 위치한 시스템의 경우 소량의 연결을 지속적으로 유지하면서 대량의 데이터 전송 기능을 수행하는 것으로 분석된다. 따라서 전단부 시스템일수록 시스템 오버헤드와 프로토콜 처리 부하를 경감하거나 가속하는 기술이 요구되며, 후단부 시스템일수록 메모리 접근 부하를 경감하거나 데이터패스를 가속하는 기술이 요구되어 진다[5,6].



(그림 3) 응용 서비스별 네트워트 부하의 특성 비교

TCP/IP 프로토콜 처리에 있어서 발생하는 시스템 오버헤드는 TCP/IP 프로토콜 스택이 운영체제의 각종 커널 프로세스(또는 쓰레드)와 프로세서, 메모리, 입출력장치 등의 시스템 자원을 공유함에 기인한다. 운영체제의 스케줄링, 자원 관리, 인터페이스 정책등을 준수하여야 함에 따라 인터럽트, 시스템콜, 드라이버, 버퍼관리 등의 기능과 연동하여 처리되어야 하며 이로인하여 프로토콜 처리라는 본연의 부하외에 오버헤드가 발생하게된다[1,7].

TCP/IP 프로토콜 처리에 의해서 발생하는 부하는 프로토콜 스택 코드를 수행하는 부하로서 연결관리, 헤더 정보 처리 등과 같은 원천적인 부하이다. TCP/IP 프로토콜 처리를 위한 코드 처리에 있어서 헤더 정보 및 페이로드 처리를 위한 빈번한 메모리 접근이 이루어지며, 이로 인하여 야기되는 부하를 메모리 접근 부하로 정의한다. 메모리 접근 부하는 메모리 접근 지연 문제와 데이터의 버퍼링 및 복제 문제로 구분된다[1,7].

메모리 접근 지연 문제는 TCP/IP 프로토콜 처리를 위해서 사용되는 헤더 데이터와 페이로드 데이터 접근에 있어서 발생하는 캐쉬 미스에 의해서 발생한다. 수신되는 패킷의 프로토콜 처리에 있어

서 필요한 헤더 데이터 및 페이로드 데이터는 시스템에 새로이 입력되는 데이터이므로 시스템의 데이터 영역에서 시간적 지역성을 상실하게 된다. 따라서 프로세서에 의해서 데이터 접근이 시도될 경우 캐쉬 미스를 야기하게 되며 이로인하여 프로세서의 스톨 현상이 발생하게 되고 결과적으로 성능 저하의 원인이 된다[1,7].

데이터 버퍼링 및 복제는 송신처리 보다는 수신처리에 있어서 심각성이 대두된다. 패킷이 수신되면 수신된 데이터는 응용 프로그램이 수신 동작을 수행하기전까는 커널의 버퍼 영역에 임시 저장된다. 응용 프로그램이 데이터 수신 처리를 시작하게 되면 커널 버퍼의 내용은 다시 응용 프로그램의 버퍼 영역으로 복제되어야 한다. 송신시에는 응용 프로그램의 버퍼에서 NIC 하드웨어로 DMA 를 통하여 데이터를 이동하는 무복제 전송기법이 활용되고 있어서 복제에 의한 부하를 경감하고 있으나, 이 역시 전통적인 BSD 소켓 인터페이스를 통해서는 구현이 불가능하다. 데이터에 대한 버퍼링 및 복제는 단순한 데이터의 이동을 위하여 프로세서가 과도한 로드와 스토어를 수행하게 되고, 이때 캐쉬 미스까지 발생시키게 됨에 따라 시스템의 성능저하의 주요한 원인이 되며, 상용화가 시작된 10 Gbps 급 이더넷 정합에 있어서 병목 현상의 원인이 된다[1,7].

III. 네트워크 입출력 처리 성능 향상 기술

TCP/IP 프로토콜 처리에 있어서 발생하는 부하를 경감하기 위한 기술들은 서버의 성능 향상은 물론 서비스의 품질 향상을 위해서 필연적으로 요구되어지는 기술이다. 이에 여러가지 방식의 TCP/IP 프로토콜 가속을 위한 기술들이 개발되고 또 연구되고 있다. 본 절에서는 TCP/IP 프로토콜 처리에 있어서 성능 향상을 위해서 NIC 하드웨어 및 프로토콜 스택 변경을 통하여 구현되는 기술과 TOE(TCP offloading Engine)기술에 대해서 설명한다.

1. NIC 하드웨어 기술 및 스택 구조 개선 기술

NIC 하드웨어의 기능 추가를 통한 TPC/IP 프로토코콜 처리 성능 향상은 프로토콜 스택 및 운영체제의 변경을 최소화하여 성능을 향상시키고자 하는 기법들이다. TCP/IP 프로토콜 처리에 있어서 발생하는 시스템 오버헤드와 데이터 버퍼링 및 복제 부하를 경감하기 위한 부가기능들이 NIC 하드웨어에 포함되고 이를 지원하는 기능을 운영체제에 부가하는 형식을 취하고 있으며, 현재 시장에 출시되는 대부분의 NIC 하드웨어에 탑재되어 있는 기술로서 그 기능은 다음과 같다[1,8].

인터럽트 통합처리 (Interrupt Coalescing)

패킷 수신에 있어서 발생하는 인터럽트 오버헤드를 경감하기 위한 기술로서 NIC 하드웨어가 수신되는 패킷들을 임시 저장하고, 여러 개의 패킷을 모아서 한번의 인터럽트로 운영체제에 전달하는 방법이다. 패킷이 수신할 때마다 발생하는 인터럽트를 일정 시간 단위 또는 도착한 패킷의 개수 단위로 모아서 인터럽트를 요청함으로서 시스템의 인터럽트를 줄이고 이로인한 오버헤드를 경감한다.

체크섬 오프로딩 (Checksum Offloading)

TCP/IP 프로토콜은 전송중에 발생하는 에러 탐지를 위하여 체크섬 방식을 사용한다. 체크섬 연산은 프로세서에서 의해서 수행되는 누적 가산 연산으로 모든 패킷에 대해서 생성 및 검사가 이루어져야 한다. 체크섬 연산은 프로토콜 처리에 있어서 프로토콜 스택과 독립적으로 연산이 가능하며 또한 하드웨어 구현에 있어서 구현 비용이 적고 NIC 하드웨어의 성능 저하 없이 구현이 가능하다. 따라서 NIC 하드웨어가 패킷의 송신 및 수신에 있어서 체크섬의 생성 및 검사를 전담하여 프로세서가 수행하여야 하는 누적 가산 연산의 부담을 제거할 수 있다.

대용량 세그먼트 지원 (Large Segment Offloading)

데이터 전송에 있어서 큰 데이터를 작은 패킷으로 나누어서 보내고, 또한 작은 패킷을 모아서 하나의 큰 데이터를 재조립하는 작업은 소프트웨어 입장에서 여러 개의 TCP/IP 헤더를 처리하여야 하는 연산 부담을 유발한다. MSS(Maximun Segment Size)를 초과는 크기의 데이터를 전송함에 있어서 연결설정이 이루어진 양단간에 협상을 통하여 대형 패킷 전송이 가능한 경우 패킷 분할 및 재조립 절차없이 데이터를 전송하도록 하는 것으로 NIC 하드웨어의 대용량 버퍼 지원을 통하여 비교적 간단하게 구현된다.

비동기 입출력 기능 (Asynchronous I/O)

네트워크 응용 프로그램의 경우 소켓을 이용하여 프로그램이 개발되며, 소켓의 송수신 요청은 TCP/IP 프로토콜 스택에 의해서 패킷 전송으로 변환된다. 비동기 입출력 처리 기능은 소켓 계층의 전송 요청을 여러 개씩 한꺼번에 요청하고 추후에 처리 결과를 확인함으로서 순차적 처리에서 발생하는 대기 시간 및 프로세스(또는 쓰레드) 스위칭 오버헤드를 최소화한다.

헤더 분리 처리 (Header Splitting)

패킷 처리에 있어서 헤더와 패이로드 부분을 분할하고 각각을 시스템내의 별도의 메모리 영역에 저장하여 처리하며, 경우에 따라서 별도의 쓰레드가 처리하는 기법이다. NIC 하드웨어는 수신된 패킷으로부터 헤더와 데이터를 분리하여 시스템 메모리의 지정된 영역에 전달하는 기능을 가져야 한다. 패킷 처리에 있어서 헤더와 데이터를 동시에 처리하는 병렬처리가 가능하고, 또한 헤더 처리에 있어서 메모리 접근에 대한 지역성을 보장할 수 있어서 선확보(Prefetch)와 같은 소프트웨어 성능 향상 기법을 동시에 활용할 수 있는 장점이 있다.

수신 성능 확장 구조 (Receive Side Scaling)

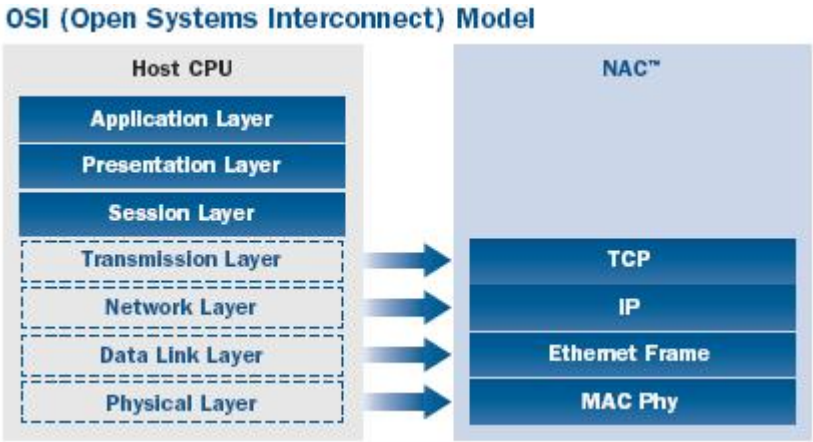
다중프로세서 시스템에서 대량의 패킷 수신을 처리하기위한 네트워크 입출력 처리 구조이다. NIC 하드웨어는 여러 개의 패킷 버퍼를 내장하고 각각의 패킷 버퍼는 시스템의 각 프로세서에게 할당되어 수신된 패킷을 병렬로 처리한다. TCP 연결별 병렬처리를 가능하게 하여 시스템의 전체적인 성능 및 네트워크 응답시간 단축에도 효과가 크다. 하지만 NIC 하드웨어의 복잡도가 많이 증가하며, 특히 NIC 하드웨어 내부에 여러 개의 패킷 버퍼를 내장하여야 하므로 구현 비용이 상대적으로 높다. 현재 시장에 출시되는 NIC 하드웨어에서는 지원하지 않는 기술이지만, 하드웨어의 저자격화에 따라 조만간에 NIC 하드웨어의 필수 기능으로 자리잡을 것으로 예측된다.

현재 시장에 출시되는 고성능 NIC 하드웨어는 대부분 상기와 같은 기능들을 적극 활용하여 1 Gbps 이하의 환경에서 최적의 성능을 보장해주고 있다. 하지만, 여러 개의 1 Gbps 급 NIC 을 사용하거나 10 Gbps 급 NIC 을 사용하여 수 Gbps 전송 성능을 제공하기에는 여전히 시스템의 성능이 부족한 상황이다.

2. TOE(TCP Offloading Engine) 기술

수 Gbps 급 이상의 네트워크 입출력 처리 속도을 제공하가 위하여 산업계를 주축으로 시스템의 TCP/IP 프로토콜 처리 기능을 하나의 하드웨어 장치로 독립시키는 노력이 진행되어 왔으며, 그 결과 TOE 라는 하드웨어 기술이 등장하였다[4, 9]..

TOE 는 (그림 4)와 같이 운영체제 내부에서 소프트웨어로 수행되는 TCP/IP 프로토콜 스택을 별도의 전용 하드웨어로 구현한 것으로 시스템과 분리된 하드웨어 프로토콜 스택으로 정의할 수 있으며, 시스템의 공유 자원과 분리되는 독립적인 메모리 공간을 사용한다[1,9].



(그림 4) TOE 하드웨어의 개념도

TOE 하드웨어는 TCP/IP 프로토콜 스택 처리에 사용되는 프로세서 부하를 시스템으로부터 제거할 수 있는 장점이 있지만 시스템의 전체적인 성능 및 기능면에서 여러가지 단점을 내포하고 있다 [1,9]. TOE 하드웨어는 SoC(System on a Chip) 형태로 구현되며 주로 32 비트급 RISC(Reduce Instruction Set) 프로세서를 내장한다. 시스템에서 사용되는 수 GHz 급 고성능 프로세서에 비해 현저히 성능이 낮은 200 ~ 300 MHz 프로세서가 내장 프로세서로 사용됨에 따라 상대적으로 시스템에서 처리할 때 보다 처리 속도가 늦어지며, 서버급 프로세서에 비해 메모리 접근을 위한 버스 속도가 느려서 메모리 접근 지연 시간을 발생시킨다. 따라서 패킷에 대한 처리 지연 시간면에서 장점을 가지지 못한다.

TOE 하드웨어는 기존의 운영체제에 존재하는 프로토콜 스택을 별도의 전용 하드웨어 스택으로 구현하는 것이므로 TOE 를 위한 전용 인터페이스를 운영체제가 제공하여야 한다. 이로 인하여 기존의 네트워크 응용 소프트웨어 사용에 있어서 호환성 문제를 야기시키며, 이는 TOE 하드웨어의 범용성 및 유연성에 큰 약점으로 지적된다.

TOE 하드웨어를 이용한다하여도 시스템은 TOE 하드웨어와 응용 프로그램간의 인터페이스를 제공하여야하며 이러한 인터페이스를 통하여 연결 제어 및 전송 명령을 TOE 하드웨어로 전달한다. 시스템과 TOE 하드웨어간의 인터페이스 처리에 발생하는 부하는 제거가 불가능하며, 연결 설정 및 해제 그리고 송수신 동작의 빈도가 높을수록 인터페이스를 통한 메시지 전달량이 늘어난다. 따라서 짧은 연결을 통하여 이벤트와 같은 소형 패킷을 처리하는 일반적인 인터넷 응용 서비스에서는 성능 개선 효과가 낮은 단점이 있다.

IV. TCP/IP 온로딩 시스템 기술

이더넷의 고속화와 더블어 인터넷 기반 응용 서비스의 증가에 따른 네트워크 트래픽의 기하급수적 증가로 인하여 서버의 네트워크 입출력처리 성능에 대한 요구가 나날히 증가하고 있으나, 시스템의 성능 향상은 네트워크 속도 향상에 비해서 현저히 느리게 발전하고 있다[1,3]. 이로인하여 현존하는 서버 시스템의 경우 수 Gbps 내지는 향후 시장의 주도품목인 10 Gbps 급 NIC 하드웨어 정합을 지원하는데 있어서 성능 한계에 직면하게 되었다[1,3,4].

서버의 네트워크 입출력 처리 성능 개선을 위하여 인텔, 마이크로소프트 등 선진기업들은 네트워크 입출력 처리 구조를 개선하는 연구를 진행해왔으며, 그 결과 인텔은 ETA(Embeded Transport Acceleration) 기술을 기반으로 하는 I/OAT(Intel I/O Acceleration Technologies) [5,6,8]를 발표하였으며, 마이크로소프트의 Chimney 구조를 기반으로 하는 새로운 NDIS(Nerwork Driver Interface Specification) [10,11] 발표하였다.

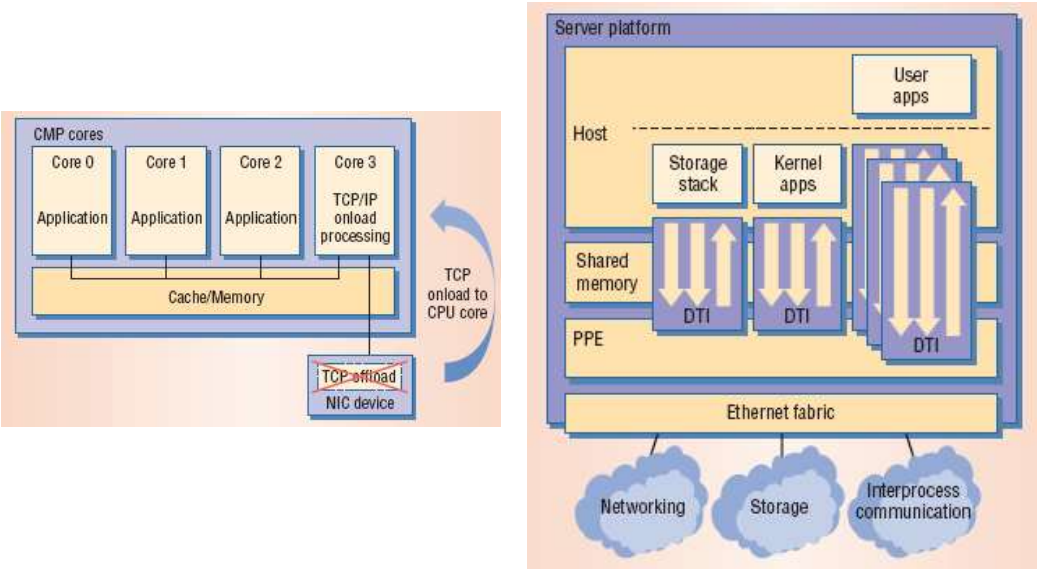
I/OAT 기술은 네트워크 프로토콜 처리에 있어서 발생하는 부하를 시스템 전반에 걸쳐서 구조와 기능을 개선하여 지원하는 이른바 Onloading 기술을 기반으로 하며. Chimney 기술은 TOE 하드웨어를 기반으로 윈도우 플랫폼에서 프로토콜 스택을 변형하는 기술이다. 본 절에서는 Onloading 개념을 통하여 서버의 네트워크 입출력 처리 구조를 새롭게 구현하는 주요 기술에 대해서 설명한다.

1. CMP 프로세서와 ETA 구조

반도체 기술의 발전에 따라 하나의 프로세서 칩에 다수의 프로세서 코어를 내장하는 CMP(Chip MultiProcessor)가 등장하였다. 인텔은 차세대 프로세서 구조로 CMP 를 제안하고 있으며, CMP 를 사용하는 서버 시스템에서 CMP 내부의 프로세서 코어를 TCP/IP 프로토콜 처리 전용 프로세서로 할당하는 구조를 제안하고 있다[7].

CMP 프로세서를 기반으로 하는 ETA(Embeded Transport Acceleration) 구조는 프로세서에 내장된 여러 개의 코어중 하나 또는 그 이상을 TCP/IP 프로토콜을 전담 처리하는 PPE(Protocol Processing Engine)으로 할당하고 나머지 코어는 일반 응용 프로그램을 수행한다. 응용 프로그램을 수행하는 코어는 PPE 코어와 DTI(Direct Transport Interface) 라는 큐 인터페이스를 통하여 정합된다. DTI 는 과거의 VIA(Virtual Interface Architecture) 또는 InfiniBand 등과 같은 고성능 클러스

터 연결망[2]에서 사용하는 큐기반 메시지 전송기법을 모태로 한다. DTI는 응용 프로그램을 수행하는 코어와 PPE 코어간에 공유하는 시스템 메모리에 구현되는 비동기 큐 모델로서 제어 큐, 데이터 큐, 동기화 큐로 구성되며 소켓기반의 명령(Connect, listen, accept 등)을 TCP/IP 프로토콜 처리하는 전용 인터페이스를 제공한다. (그림 5)는 CMP 프로세서 구조와 이를 이용한 ETA 구조의 구성도이다[1,7].



(A) CMP를 이용한 TCP/IP Onloading (B) ETA S/W 구조

(그림 5) CMP 프로세서와 이를 이용한 ETA 구조

2. 메모리 접근 지연 시간 개선

TCP/IP 프로토콜 처리에 있어서 헤더 및 데이터 처리를 위해 반복적으로 발생하는 메모리 접근 및 데이터 복제 오버헤드 최소화를 위하여 MARS(Memory Aware Reference Stack)이라는 기술을 사용한다. MARS는 TCP/IP 프로토콜 처리를 위해서 사용되는 데이터를 프로세서에서 가까운 하드웨어에 위치시키며 아울러서 메모리 접근 동작과 프로세서의 연산을 중첩시킬 수 있도록 메모리 사용 모델을 개선한 것이다.

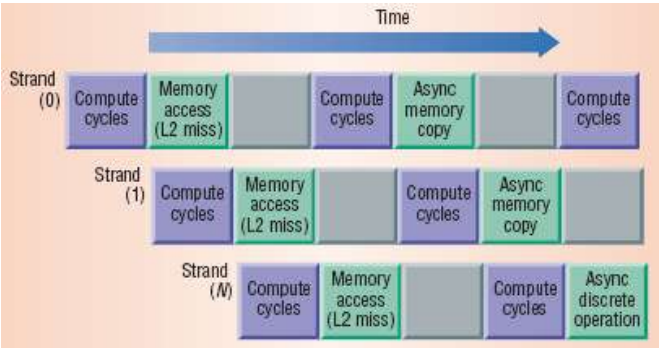
MARS 기능 구현을 위해서 사용되는 기술은 경량 쓰레드(Lightweight Threading), DCA(Direct

Cache Access), AMC(Asynchronous Memory Copy) 등이다.

경량 쓰레드 (LightWeight Threading)

메모리 접근 지연 시간에 의한 프로세서의 싸이클 소비를 최소화하기 위해서 하나의 쓰레드를 더 작은 단위로 나눈 Strand 라는 초경량 쓰레드를 사용한다. Strand 는 운영체제의 커널 쓰레드와는 달리 프로세스와 독립적으로 동작하는 것으로 하나의 연결을 통하여 처리되는 패킷 처리와 같이 매우 작은 단위의 네트워크 입출력 처리 동작으로 볼수있다. 따라서 운영체제의 스케줄링 정책에 따라 움직이는 커널 쓰레드와는 달리 Strand 는 하드웨어 상황에 따라 처리되는 소규모의 일련의 고정된 소프트웨어 코드라고 볼 수 있다.

Strand 는 프로세서의 캐쉬미스에 의해서 스위칭이 발생하는 미세한 단위의 초경량 쓰레드로서 스위칭 오버헤드가 매우 적으며, 메모리 접근 지연 시간과 패킷 처리를 중첩하여 프로세서의 싸이클 소비를 최소화하고 아울러서 연결 단위 수준의 패킷 병렬 처리를 가능하게 한다. (그림 6)은 Starnd 를 이용한 경량 쓰레드 파이프라인을 보여준다.

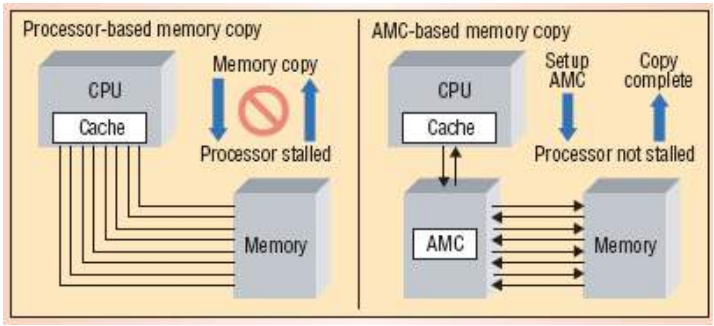


(그림 6) Strand 를 이용한 경량 쓰레드 파이프라인

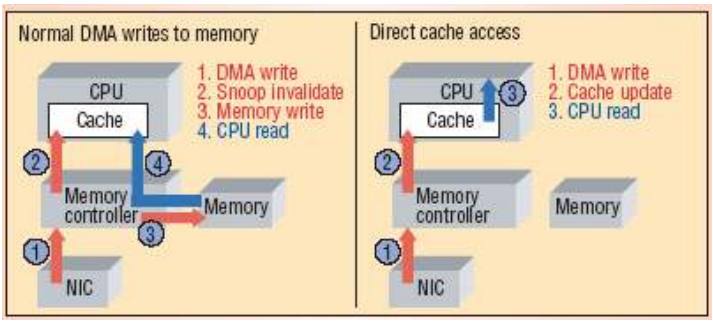
DCA (Direct Cache Access)

DCA 는 프로세서와 NIC 하드웨어간의 데이터 전송을 가속하기 위한 기법으로 프로세서와 입출력장치간의 연결을 지원하는 브릿지 칩셋의 기능 개선이 요구되는 기술이기도 하다. 일반적으로 패킷 수신이 이루어지면 NIC 하드웨어는 소켓에 대한 디스크립터와 헤더, 페이로드 등 패킷 처리에 필요한 모든 데이터를 메모리이 위치시킨후에 프로세서로 패킷 도착을 알린다. 프로세서는 패킷 처리를 위하여 필요한 데이터를 내부로 읽어들이며 이때 캐쉬미스를 야기하게 되어 메모리 읽기를 위한 접근 시간은 증가하게 된다. DCA 는 네트워크 입출력 데이터 처리에 있어서 발생하는 캐쉬미스 오버헤

드를 제거하기 위한 기술로서, 프로세서로 패킷 도착을 알리기전에 NIC 하드웨어에서 프로세서 캐쉬로 데이터를 미리 옮겨놓는 기술이다. (그림 7)은 NIC 하드웨어에서 프로세서까지 DMA(Direct Memory Access)와 DCA 를 사용할 때의 차이점을 도식화 한 것이다.



(그림 7) DCA 를 이용한 데이터 전송



(그림 8) AMC 를 이용한 데이터 복제

AMC (Asynchronous Memory Copy)

데이터 복제는 TCP/IP 프로토콜 처리에 있어서 발생하는 가장 큰 오버헤드로서 프로세서의 로드-스토어 명령에 의해서 수행될 경우 프로세서의 부하는 크게 증가하게 된다. AMC 는 데이터 복제를 위하여 스케줄링 개념이 가미된 개선 DMA 기능으로 정의될 수 있으며, 경량 쓰레드 파이프라인과 연동하여 Strand 수준의 병렬 처리를 지원하는 도구로 활용된다. 하나의 패킷을 처리하는 Starnd 가 데이터 복제가 필요하게 되면 AMC 하드웨어에 복제 명령을 전달하고 스위치 아웃된다. 그리고 다른 Strand 가 프로세서에서 수행되며, AMC 는 입력된 복제 명령을 수행하여 복제 명령을 발생한 Strand 가 다시 스위치 인 될 때 지장없이 패킷을 처리할 수 있게 한다. (그림 8)은 프로세서에 의한 데이터 복제와 AMC 를 이용한 데이터 복제를 비교하여 도식화한 그림이다.

V. 결론

인터넷의 보급과 웹 기반 서비스의 등장은 서버의 역할을 연산 및 자료처리에서 TCP/IP 기반의 네트워크 서비스 제공으로 변화시켰다. 인터넷의 폭발적인 성장과 네트워크 전송 기술의 발전에 따라 서버의 네트워크 입출력 부하는 크게 증가하여 성능 한계점에 이르렀으며, 이로 인하여 다양한 방식의 네트워크 입출력 성능 개선 기술들이 개발되어왔다. 10 Gbps 급 이더넷이 현실화되고 있는 지금 서버 시스템은 날로 증가하는 네트워크 입출력 부하를 처리하기 위하여 연산 능력 향상을 목적으로 추진 하던 기술 개발 방향을 입출력 처리 성능 향상 기술로 전환하고 있으며, 인텔의 II/OAT, 마이크로소프트의 Chimney 등이 이러한 방향을 주도하고 있다.

인터넷의 확산 및 고도화와 더블어 서비스의 고품질화는 서버의 입출력 부하를 더욱 가중시킬것으로 예상되며, 이에 따라 고전적인 서버의 구조는 네트워크 입출력 처리에 특화된 기술을 수용함으로서 점차 네트워크 서버 형태로 발전할 것으로 예측된다.

<참 고 문 헌>

[1] G. Regnier et. al., “TCP Onloading for Data Center Server”, IEEE Computer, Vol. 37, Issue 11, Nov. 2004, pp 48-58.

[2] Renato John Recio, “Server I/O Networks Past, Present, and Future”, Proc. ACM SIGCOMM 2003 Workshop, Aug. 2003, pp. 163-178.

[3] Annie P. Foong et. al., “TCP Peformance Re-Visited” : http://www.cs.duke.edu/~jaidev/papers/ispass03.pdf

[4] Andy Currid, “TCP Offloading to the Rescue”, Queue, Vol. 2, No. 2, Issue 3, May 2004, pp. 58-65.

[5] Intel I/O Acceleration Technology, : http://www.intel.com/technology/ioacceleration/306484.pdf

[6] Accelerating High-Speed Networking with Intel I/O Acceleration Technology, : http://www.intel.com/technology/ioacceleration/306517.pdf

[7] G. Regnier et. al., “ETA: Experience with an Intel Xeon Processor as a Packet Processing Engine”, IEEE Micro, Vol. 24, Issue 1, Jan./Feb. 2004, pp. 24-31.

[8] Keith Lauritzen, et. al., “Intel I/O Technology Improve Network Performance, Reliability and Efficiently”, Technology@Intel Magazine, May 2005. : http://www.intel.com/technology/magazine/communications/Intel-IOAT-0305.pdf

[9] J. Mogul, “TCP Offloading Is a Dumb Idea Whose Time Has Come”, Proc. 9th Workshop on Hot

Topics in Operaing Systems, Usenix Assoc., 2003 :
http://www.usenix.org/events/hotos03/tech/full_papers/mogul.pdf
[10] Scalable Networking woth RSS, :
http://www.microsoft.com/whdc/device/network/NDIS_RSS.mspx
[11] NDIS 6.0 Roadmap and Value Statement, :
http://www.microsoft.com/whdc/device/network/ndis/default.mspx