GPU와 CRC테이블을 이용한 IP패킷 라우팅 구조

(IP Packet Routing Architecture with GPUs and CRC-table)

권 기 둥†, 안 우 근‡, 이 신 형‡, 유 혁‡
†고려대학교 임베디드소프트웨어학과
‡고려대학교 컴퓨터·전파통신공학과
(Ki-doong Kwon, Woo-geun Ahn, Shin-Hyoung Lee, Chuck Yoo)
(†Department of Embedded Software. Korea Univ.)
(‡Department of Computer and Radio Communications Engineering. Korea Univ.)

Abstract : Recent research such as cloud computing, future internet architectures and network virtualization issues requires programmability to their routers. Software router supports more programmability as the router supports familiar programming environments on general-purpose operating system. In addition to that, the fact that the cost of commodity hardware is going down makes it possible to realize off-the-shelf programmable routers. Although software routers have advantages in terms of programmability and low cost, they show lower performance than hardware router. Existing software routers report near 10 Gbps forwarding performance even on a single machine, but the CPU quickly becomes the bottleneck for more compute-intensive applications. The IP address lookup is one of applications which raise the CPU bottleneck problem as the router needs to find the longest matching prefix for the address. In out paper, we present IP forwarding architecture with Graphics Processing Units (GPUs) and CRC-table inside the software router to address the CPU bottleneck. The data-parallel processing model of GPUs fits nicely with inherent parallelism in IP address lookup process so that computing needs can move to GPUs for high-throughput packet processing. We have implemented a CUDA application to process table lookup and some kernel modules to deliver huge number of packets to the application effectively. Adding to that, we utilize hash-based design using CRC-table, in order to implement one-time memory access to lookup the routing table. Because, in modern routers, routing table update takes place frequently, we also have considered table update issue. Our architecture partitions the lookup table into the smaller ones for prefixes and the access to each partitioned tables is allowed through CRC hash table. Partitioning for prefixes effectively reduces the complexity of finding “the longest prefix match” problem to “the first prefix match”. By finding the first matching prefix for addresses with one-time memory access inside hundreds of slim cores in parallel, our architecture shows significantly higher throughput performance over the CPU-only implementation.

Keywords : GPU, routing, CRC, packet, forwarding

Ⅰ. 서 론

미래인터넷에 대한 관심이 높아지면서, 현재 인터넷에 유연성을 제공하고자 하는 노력들이 이루어지고 있다[1-4][7-11]. 기존에 사용되고 있는 상용 라우터에 비해 소프트웨어 라우터는 상대적으로 적은 비용으로 확장성와 프로그래머빌리티를 제공한다[2][3][7-10]. 또한 상용 하드웨어에서의 가격 하락이 계속 되면서 성능을 높이기 위한 비용이 줄어들게 되어, 적은 비용의 상용 하드웨어를 사용하

여 소프트웨어 라우터를 구성할 수 있다는 것 또한 소프트웨어 라우터의 장점이 된다[3][4].

이러한 소프트웨어의 라우터의 장점에도 불구하고 높은 성능을 내기 힘들다는 점이 소프트웨어 라우터의 큰 문제점으로 지적되고 있다[3]. 기존에 이루어졌던 연구들을 보면, 패킷 당 처리해야될 작업들이 많아질수록 CPU 병목현상이 유발되고, 전체적인 성능저하가 일어나는 것으로 나타난다[3][4]. 이러한 CPU 병목현상으로 인한 성능저하를 막는 방법으로 라우터에 더 많은 CPU를 장착하는 방법이 있을 수 있다. 하지만 추가되는 비용당 성능을 고려했을 때 효율적이지 못하다.

본 논문은 Graphic Processing Unit(GPU)에서 제공하는 거대한 병렬성을 이용하여 라우팅 테이블 룩업 작업을 CPU에서 GPU로 가져와 수행하여, CPU의 부하를 줄이고 결론적으로 소프트웨어라우터의 전체적인 성능을 높이는 방법을 연구하였다. 본 논문은 IP 패킷 라우팅에서 검색 성능만을 고려한 것이 아니라 업데이트에 대한 이슈 또한 고려하였다. 라우팅 성능을 측정하여 리눅스에서 기본적으로 제공하는 라우팅 성능과 비교하고, 더 나아가 기존에 연구된 다른 소프트웨어 라우터와의 성능을 비교하여 더 좋은 성능을 보이도록 시스템을 설계하고 구현하는 것을 연구 목표로 하고 있다. 이를 통해, 다중 10G 소프트웨어 라우터 환경에서 GPU가 높은 성능을 달성 할 수 있는 큰 기회를 제공할 수 있음을 보여준다.

II. 관련 연구

1. RouteBricks

RouteBricks[3]는 Intel Research Berkeley에서 제안된 소프트웨어 라우터 디자인으로, 상용 하드웨어 상에서 다중 서버간 및 한 서버내의 다중 코어간에 라우터 기능을 병렬화하여 분산적으로 처리하는 방식을 가지고 있다. 또한 RouteBricks는 서버의 숫자를 증가하여 선형적인 처리능력 확장성을 달성한다. RouteBricks의 중요한 디자인 요소는 Direct VLB 즉, 패킷 처리 및 스위칭을 병렬화하는 로드밸런싱 라우팅 알고리즘을 사용했다는 것이다. 이 알고리즘은 메모리 접근에서의 병렬성 이용, 다중 수신 및 전송 NIC 큐 그리고 패킷의 일괄처리등과 같은 구현 전략들과 같이 사용된다. 그 결과, 각각 하나의 10Gbps NIC가 장착된 4개의 서버로 구성된 라우터가 35Gbps까지의 성능을 지원할 수 있음 보여준다. RouteBricks에서 성능을 저하시키는 원인으로 지목되는 것이 CPU인데, 이는 패킷당 처리해야 될 작업량이 많아질수록 CPU 병목현상이 일어나기 때문이다. RouteBricks는 앞으로 차세대 서버들이 코어의 숫자가 증가하고 많은 수의 PCIe 슬롯을 제공함으로써 성능의 큰 향상을 이룰 수 있을것으로 기대하고 있다.

2. PacketShader

PacketShader[4]는 카이스트에서 제안된 소프트웨어 라우터 플랫폼으로서, GPU를하여 성능의 향상을 이루었다는 것이 다른 소프트웨어 라우터들과의 차별화 요소이다. PacketShader는 GPU의 거대한 병렬 처리 능력을 이용하여 RouteBricks의 성능 한계를 극복하였다. PakcetShader는 4개의 NIC 장치를 장착한 단일 상용 PC에서 64B IPv4 패킷을 39Gbps의 속도로 전달할 수 있음을 보여주었다. PacketShader는 IP 패킷 라우팅 알고리즘으로 DIR-24-8 알고리즘[5]을 사용하였는데, 이를 통해 거의 한번의 메모리 접근으로 라우팅을 수행할 수 있게 하였다. 하지만 PakcetShader는 라우팅 테이블 업데이트에 대한 이슈는 고려하지 않았는데, DIR-24-8 알고리즘의 특성상 업데이트 과정이 복잡하고 그에 따른 비용이 크기 때문에 라우팅 테이블의 업데이트가 자주 발생하는 네트워크에서는 사용하기 힘들다는 점이 PacketShader의 단점이 된다[5].

III. 설 계

1. 프리픽스에 기반한 라우팅 테이블 분할관리

CIDR환경에서 라우팅 테이블 검색은 Longest Prefix Matching방식으로 이루어졌다. 테이블에서 매칭되는 주소를 찾더라도 더 길게 매칭되는 주소가 있을 수 있기 때문에 거대한 테이블 검색을 여러번 수행해야 한다는 점이 오버헤드로 지적되고 있다. 본 논문에서는 이러한 오버헤드를 줄이기 위해 라우팅 테이블을 프리픽스에 기반하여 분할 관리하는 방식을 채택하였다. 이렇게 분리된 테이블내에서의 검색은 Longest prefix matching 방식에서 first prefix matching 방식으로 변경되기 때문에 하나의 주소에 대한 결과를 얻기 위해 한 번의 검색만 수행하면 된다[6]. 또한 GPU에서 이렇게 나누어진 테이블에 대해 각각의 코어가 병렬적으로

검색을 수행하게 하여 CPU에서의 검색에 비해 훨씬 높은 성능으로 결과를 얻을 수 있다.

2. CRC-table를 이용한 테이블 구성 및 접근

앞서 언급했던 테이블을 구성하고 접근하는 것은 해싱에 기반하여 이루어진다. 해싱방식을 적용하게 되면 한번의 메모리 접근으로 일치하는 테이블 엔트리에 접근할 수 있다는 장점이 있다. 본 논문에서는 해싱 키로서 기존의 해싱 함수중에서 거의 완벽한 해싱 성능을 낸다고 평가되는 CRC 값을 사용한다. 또한 본 논문에서는 CRC 값을 얻기 위해 CRC 테이블을 이용한다. CRC 계산을 소프트웨어적으로 구현하는 것이 많은 CPU 계산을 요구하게 되는데, 미리 계산된 CRC값들을 테이블에 저장하여 사용하게 되면, 한번의 메모리 접근으로 원하는 라우팅테이블 엔트리에 대한 인덱스를 얻을 수 있다 [12].

IV. 구 현

본 논문을 구현하기 위해 Intel ixgbe 드라이브 수정 및 CUDA 응용프로그램 제작이 이루어졌다. ixgbe 드라이브는 NIC 카드에 수신된 패킷들을 NAPI 인터페이스를 통해 일괄적으로 모은 다음 CUDA 어플리케이션에 전달한다. CUDA 어플리케이션은 시작시 라우팅을 위한 라우팅 테이블 및 CRC테이블을 제작하여 GPU 메모리에 전달한다. 네트워크 드라이브로부터 패킷들이 전달되면 이를 GPU 메모리로 전송하고 GPU에 라우팅 작업 명령을 수행하도록 한다.

1. 테이블 구성

테이블을 구성할 때, 프리픽스에 해당하는 분할된 라우팅 테이블이 선택되고, IP 주소와 출력 인터페이스 정보가 획득이 되면, IP 주소를 가지고 CRC 테이블에 접근하여 해당하는 CRC값을 얻는다. 이렇게 얻어진 CRC값을 인덱스로 하여 분할된 라우팅 테이블에 엔트리로 저장한다.

2. 테이블 검색

테이블을 검색하는 과정은 CRC테이블에 접근하여 얻어진 CRC 값을 가지고 분할된 라우팅 테이블에 접근하는 것으로 각 GPU 코어에서 병렬적으로 이루어진다. 검색이 필요한 IP 주소가 들어오게 되면 프리픽스 0부터 프리픽스 32까지 모든 가능한 프리픽스에 대한 테이블에서의 검색을 프리픽스당 하나의 코어가 담당하게 된다. 본 논문은 현재 IPv4 유니캐스트만을 고려하고있기 때문에 프리픽스 31에 대해서는 GPU 검색을 수행하지 않는다. 추가적으로 해싱 기반의 테이블의 특성상 오버플로우가 발생할 수 있기 때문에 오버플로우 테이블에 대해서도 검색을 수행해야 한다. 결론적으로 하나의 IP 패킷당 32개의 GPU를 코어를 사용하여 검색을 수행하게 되는데 3072개의 코어를 가진, Geforce GTX490 그래픽 카드 사용할 경우 동시에 약 100개의 패킷을 처리할 수 있다. 그렇게 얻어진 결과값 중에서 가장 긴 프리픽스에 해당하는 테이블에서 얻어진 결과를 사용한다.

3. 테이블 변경

테이블 변경은 앞서 언급했던 구성 및 검색과 비슷하게 이루어진다. CRC테이블에 접근하여 얻어진 CRC값을 가지고 변경이 필요한 분할된 라우팅 테이블에 접근한 다음, 엔트리를 변경하게 된다.

라우팅 테이블 변경에 있어 CRC테이블 접근, 라우팅 테이블 접근, 이렇게 두 번의 테이블 접근이 필요하게 되므로 PacketShader에 비해 업데이트에 있어서 오버헤드가 작고 작업이 단순하다.

V. 결 론

본 논문은 Software router의 성능을 높이기 위해 GPU를 이용하여 라우팅 테이블 룩업을 수행하는 새로운 구조를 제안하였다. NAPI 인터페이스를 통해 패킷을 모은 뒤, 패킷을 CUDA 어플리케이션에 전송하는 것으로 패킷 당 소모되는 오버헤드를 줄였으며, 분할 테이블 및 CRC 테이블의 사용으로 주소검색에 필요한 오버헤드를 줄였다.

본 논문에서 제안하는 구조는 검색에서의 성능을 보장할 뿐 아니라 라우팅 테이블 관리에서도 용이함을 제공하게 되어 실제 라우팅 테이블 업데이트가 자주 일어나는 네트워크 환경에서도 쓰일 수 있음을 보였다.

참 고 문 헌

[1] The Internet is broken.
http://www.technologyreview.com/news/405318/the-internet-is-broken/

[2] Xie, G., He, P., Guan, H., Li, Z., Xie, Y., Luo, L., Zhang, J., Wang, Y., Salamatian, K.: PEARL: a programmable virtual router platform. IEEE Communications Magazine 49, 71?77 (2011)

[3] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. RouteBricks: exploiting parallelism to scale software routers. In SOSP, 2009.

[4] Sangjin Han, Keon Jang, KyungSoo Park, Sue Moon. "PacketShader: a GPU-Accelerated Software Router", SIGCOMM10. 2010..

[5] P.Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds", In IEEE INFOCOM, 1998

[6] Mohammad J. Akhbarizadeh and Mehrdad Nourani. "An IP Packet Forwarding Technique Based on Partitioned Lookup Table", IEEE International Conference on Communications, ICC2002, New York,

[7] Astaro: Security Gateway. http://www.astaro.com

[8] Riverbed: Application Acceleration. http://www.riverbed.com.

[9] Sourcefire: Network Security. http://www.sourcefire.com.

[10] Symantec: Data Loss Protection. http://www.vontu.com.

[11] Vyatta Series 2500. http://vyatta.com/downloads/datasheets/vyatta_2500_datasheet.pdf.

[12] Ramabadran, T.V.; Gaitonde, S.S. (1988). "A tutorial on CRC computations". IEEE Micro 8