고속의 채널기반 네트웍에서 SDP 프로토콜 성능평가

김영환, 박창원, 전기만
전자부품연구원 유비쿼터스 컴퓨팅 센터

A Performance Evaluation for SDP(Socket Direct Protocol) in Channel based Network

Young Hwan Kim, Chang Won Park, Ki Man Jeon
Ubiquitous Computing Research Center, Korea Electronics Technology Institute

Abstract - 네트워크 사용자의 급속한 증가로 네트워크 내의 부하를 감당하기에는 많은 어려움을 가져왔다. 이와 같은 이유로 기존의 TCP/IP에서 세션을 통하여 노드들 간의 통신을 연결하는 방식에서 현재는 하나의 채널을 통해 고속의 I/O가 가능하도록 하는 기술이 많이 연구되고 있다. 그 대표적인 것으로 인피니밴드가 있다. 인피니밴드는 프로세싱 노드와 입출력 장치 사이의 통신, 프로세스간 통신에 대한 산업 표준이 되고 있고 프로세싱 노드와 입출력 장치를 연결하기 위해 스위치 기반의 상호 연결은 전통적인 버스 입출력을 대체하는 새로운 입출력 방식이 사용된다. 또한 인피니밴드에서는 현재 많은 이슈가 되고 있는 RDMA 방식을 이용해 원격지 서버들 간에 직접 메모리 접근 방식을 통해 CPU와 OS의 로드를 최소화하고 있다. 본 논문에서는 RDMA를 적용한 새로운 채널 기반 네트웍의 프로토콜인 SDP(Socket Direct Protocol)를 구현하여 SDP_STREAM의 패킷 처리량에 대한 성능을 평가한다. 그리고 이에 대한 성능 평가를 위해서 Netperf 툴을 이용했다. 특히 Zero-Copy방식을 사용하지 않는 일반적인 소켓 API을 이용한 TCP_STREAM과 Zero-Copy방식을 이용한 SDP_STREAM의 패킷 처리량을 비교했으며 성능 평가 결과는 기존의 TCP_STREAM 패킷 처리량에 비해 약 3배 이상 향상된 결과를 나타냈다.

1. 소 개

네트워크 기술의 급속한 발달과 정보의 효율적 공유에 대한 요구의 증가로 거의 모든 컴퓨팅 장비들이 네트워크에 연결되어 있다. 따라서 복잡하게 구성되어 있는 전체 네트워크의 상태 및 장비들을 효율적으로 파악하고, 관리하기 위한 필요성이 대두되고 있다. 특히 대용량 스토리지와 서버사이 입출력 분야에서는 한 개의 프로세서와 여러 개의 입출력 장치를 가진 소규모의 서버에서 수백개의 프로세서와 수천개의 입출력 장치를 가진 대규모의 슈퍼 컴퓨터까지 사용 가능한 인피니밴드는 더욱 더 중요성이 대두 되고 있다.

현재 컴퓨터 시스템에서 사용되고 있는 입출력 버스 방식은 디스크 접근, 특히 고 성능의 서버에 있어서 병목 현상의 주요원인으로 나타나고 있다. 이러한 버스 방식은 구조가 단순하다는 큰 이점을 가지고 있어 지금까지 산업 전반적으로 사용되어 왔지만 버스 기반의 입출력 시스템은 현재의 디바이스 장치들이 요구하는 데이터 전송 대역폭을 처리할 수 있을 만큼의 시스템 입출력 성능을 가지고 있지 않다. 또한 현재 대부분의 네트워크 제품들은 최고의 패킷 처리량과 최소의 전송 지연, 그리고 전송 대역폭에 대한 보장을 요구해 왔고, 이는 하드웨어와 커널 바이패싱, Zero-Copy 네트워킹에서 신뢰성 있는 전송 프로토콜 사용을 통해 가능하게 되었다. 이들 메커니즘은 전통적인 TCP/IP방식에서는 불가능했던 초고속의 네트웍 데이터 프로세싱을 가능하게 한다[1-2].

인피니밴드에는 SDP, IPOIB(IP Over Infiniband), SRP(SCSI RDMA Protocol) 등과 같은 여러 응용 프로토콜이 있다. 여기서SDP는 인피니밴드와 같은 새로운 네트웍 기술이 가진 특징을 기존 소켓 어플리케이션이 사용 가능하도록 하기 위해 개발 되었고, 원격지 DMA버퍼간에 읽기와 쓰기를 통해 데이터를 전송하는 Zero-Copy방식을 사용한다. 또한 SDP를 사용하기 위해서는 신뢰성 연결(RC: Reliable Connection) 메세지 기반의 전송 프로토콜 계층에서 ULP(User Level Protocol) 인터페이스를 통해 접근하는데 이에 대한 더 자세한 내용은 다음의 [11] SDP 명세서를 참고하기 바란다.

기존 이더넷에서는 TCP/IP 계층에서 프로토콜 프로세싱를 위해 하나의 패킷을 처리하기 위해 커널 영역과 사용자 영역의 메모리에 읽기/쓰기를 반복한다. 이 결과로 전송 지연이 커지게 되는 단점이 있다. 그러나 인피니밴드에서는 커널 바이패싱이나 RDMA 기법을 사용하여 이 CPU에 대한 부하와 전송 지연을 최소화 하고 있다. 본 논문에서는 인피니밴드에서 SDP를 구현하고 이를 이용했을 때 패킷 처리량과 CPU utilization을 기존의 TCP/IP을 이용했을 때와 성능을 비교 분석할 것이다[4, 7].

본 논문의 구조는 다음과 같다. 2장에서는 인피니밴드 프로토콜 스택 구조와 SDP구조에 대해 설명하며 3장에서는 Netperf을 이용해서 SDP의 성능을 평가 한다. 마지막으로 4장에서는 요약과 함께 성능 평가를 통해 얻은 결과로 결론을 내린다.

2. 인피니밴드

2.1 시스템 구조

인피니밴드는 고속의 점대점 링크에 스위치를 기반으로 노드가 서로 연결된 망으로 설계되었다. 인피니밴드망은 하나 또는 그 이상의 스위치와 프로세싱 노드, 입출력 디바이스 장치로 구성된 서브넷이 라우터에 의해서 상호 연결되어 있다. 인피니밴드에서 라우팅은 각 스위치에 저장된 포워딩 테이블를 기반으로 하며, 유연성과 확장성을 제공하기 위하여, 사용자에 의해 정의된 어떠한 토폴로지도 지원한다.

인피니밴드 링크는 양방향 점대점 통신 채널로 되어 있고, 링크의 신호 발생율은 현재 2.5GHz이며 물리적 링크 레벨에서는 보다 높은 대역폭을 얻기 위해 병렬적으로 사용되는데, 가장 낮은 대역폭은 1X로서 2.5Gbps이고 최대 12X인 30Gbps까지 지원한다.

인피니밴드 연결망은 스위치 기반 비정형 연결망으로 종단 노드(프로세서 노드, 입출력 노드)가 연결되는 여러 개의 서브넷으로 구성된다. 서브넷은 종단 노드와 스위치, 라우터로 구성되며 서브넷 간 연결은 라우터를 통해 이루어 진다. 하나의 서브넷에는 최대 6만5536(216)개의 종단 노드를 연결할 수 있는 고확장성이 제공된다. 각 종단 노드는 인피니밴드 연결망 접속을 위한 채널 어댑터를 가지며, 프로세서 노드 쪽에서는 호스트 채널 어댑터(HCA:Host Channel Adapter)를 사용하고 입출력 처리 노드 및 입출력 장치 쪽에서는 타rpt 채널 어댑터(TCA:Target Channel Adapter)를 사용한다.

각 노드는 메세지 전송을 위해 사전에 호스트 메모리에 메세지 전송을 위해서 사용할 영역을 가상 주소를 사용·지정해야 한다. 사용자 프로그램은 메세지 전송 요구 및 메세지 데이터를 지정된 영역에 위치시키게 되고 HCA 또는 TCA는 운용 체계의 간섭 없이 지정된 영역에서 메세지 전송 요구와 해당 데이터를 가상 주소를 사용해 읽어 내어 인피니밴드 연결망으로 전송한다. 인피니밴드 전송 메커니즘은 종단노드 간에 여러 가지 타입의 통신 서비스를 제공하며 이와 같은 타입은 연결 지향성과 데이터 그램 서비스 그리고 신뢰성과 비 신뢰성으로 분류 될 수 있다. 또한 대역폭 보장과 최대 지연 마감과 같은 QoS요구조건을 만족하도록 하기 위해 어플리케이션은 자원 할당을 할 수 있도록 신뢰성 있는 연결을 사용해야 한다.

통신 방식으로는 Send/Receive와 RDMA 그리고 Atomic 의 세가지 방식을 사용한다. 여기서 SDP는 Send/Receive와 RDMA방식을 사용하는데 Send는 어플리케이션이 자신의 버퍼에 저장된 데이터를 원격지의 어플리케이션에게 보내는 것이고 Receive는 일단 원격지 어플리케이션이 Send를 동작을 하기 전에 로컬의 큐에 저장된다. 큐에는 저장될 메모리 위치에 대한 정보를 갖고 있다.

RDMA방식은 어플리케이션이 원격지 어플리케이션의 메모리에 데이터를 직접 쓰거나 읽을 수 있는데 이를위해 RDMA 메세지 내에 원격지의 가상 메모리 주소를 포함하고 있다. 따라서 원격지의 어플리 케이션은 RDMA를 위해 어떤한 동작도 필요하지 않으며 RDMA는 원격지 노드의 CPU 간섭 없이 데이터 전송이 수행된다. 결국, 그 데이터는 커널 버퍼로의 복사 없이 사용자 레벨의 버퍼간에 복사를 하게 된다 [3].

2.2 SDP

SDP는 TCP와 유사한 기능을 가진 스트림 전송 프로토콜이다. 따라서 표준 소켓 API는 SDP위에서 동작하며 기존의 소켓 어플리케이션에 대해 어떠한 수정도 없이도 사용이 가능하다. 또한, SDP는 Zero-Copy 전송과 멀티케스팅 방식을 이용한 다중 전송이 가능하다.

SDP의 주요 목적은 Zero-Copy 네트워킹과 신뢰성 있는 전송 프로토콜 사용이 가능한 새로운 네트워킹 기술에 의해 제공되는 확장성을 이용하면서 기존의 소켓 스트림 기능을 유지하기 위함이다. 인피니밴드는 SDP에 의해 사용되는 신뢰성 연결를 제공한다. SDP는 스트림 기반의 프로토콜인 반면에 신뢰성 연결은 메세지 기반의 프로토콜이다. 그래서 SDP는 하드웨어 상에서 수행되는 신뢰성있는 메세지 패싱 프로토콜을 기반으로 실행되는 스트림 프로토콜이다.

SDP는 데이터 전송을 위해 Private버퍼 와 RDMA 버퍼 두가지 타입의 버퍼를 사용한다. Private 버퍼는 Send/Receive를 수행하는 버퍼 복사 데이터 전송 메커니즘에 의해 사용되고 그 데이터는 Private 버퍼와 어플리케이션 버퍼 간에 복사한다. RDMA버퍼는 Zero-Copy데이터 전송 메커니즘에 의해 사용되고 어플리케이션 버퍼 간에 데이터가 복사된다.

2.3 SDP 데이터 전송 메커니즘

SDP는 Bcopy, Read Zcopy, Write Zcopy 그리고 Transaction의 네가지 데이터 전송 방식을 제공하고 있다. Bcopy는 데이터 전송을 위해 private 버퍼를 사용하는 반면에, Read Zcopy는 RDMA 읽기를 이용한 zero-copy전송을 하고, Write Zcopy는 RDMA 쓰기를 통한 zero-copy전송을 한다. 마지막으로 Transaction은 원격지 노드에 간단한 명령 메세지를 전송하는 작업에 대해 Write Zcopy기반으로 한 최적화된 메커니즘이다.

Bcopy메커니즘에 있어서 SDP는 원격지 노드간의 통신을 위한 연결을 맺었을 때 Private버퍼의 풀(pool)을 생성하고 유지한다. 이 풀은 원격지에서 send verb을 사용한 데이터 전송 시 로컬 노드가 이 데이터를 잠시 저장하기 위해 수신 버퍼로서 사용된다. SDP도 기존의 TCP와 같이 데이터 수신측에서 수신 버퍼에 대한 공간이 부족할 경우 데이터 송신측에서 더 이상 데이터를 보내지 못 하도록 흐름 제어에 대한 메커니즘을 가지고 있다. 이에 대한 흐름 정보는 역방향에서 오는 데이터 메세지 내에 포함되거나 빈 데이터 페이로드 내에 이 정보를 넣어서 전송하기도 한다. 그리고 SDP는 신뢰성 기반으로 한 데이터 전송 방식을 사용하므로 Acknowledgements에 대한 걱정이 없다.

Write Zcopy 데이터 전송을 위해 우선적으로 취해야 일은 RDMA 쓰기 동작에 의해 채워질 RDMA 버퍼가 유용한지를 데이터 송신측에 알리기 위해 데이터 수신측에 보내질 데이터 수신 가능 메세지를 받는 것이다. 이 수신 가능 메세지 내에는 해당 버퍼의 크기와 가상 주소, RDMA 쓰기를 수행하도록 데이터 송신측에 허락하는 정보를 포함하고 있다. 이 과정이 끝나면, 데이터 송신측은 데이터 수신 가능 메세지에 의해 알려진 데이터 수신 버퍼 내에 RDMA 쓰기를 할 수 있다. RDMA 쓰기가 완료되면 데이터 송신측은 전송된 데이터의 크기를 담고 있는 RDMA 쓰기 완료 메세지를 수신측에 보내게 된다.

Read Zcopy 데이터 전송 방식은 Write Zcopy의 정반대이다. Read Zcopy는 우선 RDMA 읽기에 의해 전송된 데이터가 저장될 버퍼 공간이 유용함을 데이터 수식측에 알리기 위해 데이터 송신 가능 메세지를 보내는 것으로 데이터 송신측은 시작한다. 이 데이터 송신 가능 메세지에는 RDMA 읽기를 수행하는 대 있어 데이터 수신측에서 필요한 정보를 담고 있다. 데이터 수신측은 이제 송신 가능 메세지에 의해 알려진 데이터 송신 버퍼로부터 RDMA 읽기가 가능하다. RDMA 읽기가 완료되면 데이터 수신측은 RDMA 읽기 완료 메세지를 데이터 송신측에 보낸다.

데이터 전송 메커니즘의 네 번째 방식은 Transaction이다. 이 전송방식은 간단한 명령 메세지를 원격지 노드에게 전손하는 방식으로 수신 가능 메세지에 그 명령어를 포함시켜 보낸다. 이 방식은 통신을 할려는 두 노드 사이에 필요한 메세지의 수를 줄일 수 있다[3, 5, 6, 9, 11].

3. SDP 성능 평가 및 분석 결과

성능 평가를 위해 사용한 서버, HCA와 기가빗 네트웍 카드에 대한 스펙은 다음의 표 1과 같다.

표 1. 성능 평가 장비 규격

	서버 1	서버 2
CPU	듀얼 Xeon 2.0GHz	듀얼Xeon 3.0GHz
RAM	3G	3G
버스	133MHz PCI-X	133MHz PCI-X
HCA	Mellanox MT23108	Mellanox MT23108
NIC	Intel Pro/1000 XT	Intel Pro/1000 XT

그리고, 성능 평가를 위해 사용된 OS는 Red Hat Linux 9.0이다.

성능 평가를 위해 사용된 인피니밴드 소프트웨어로는 SDP 미들웨어 드라이버가 포함되어 있는 Openib 인피니밴드 스택을 사용했다. Openib는 인피니밴드 개발을 위해 여러 관련 기업이 공동 작업을 위해 만든 인피니밴드 프로젝트 단체이다. 인피니밴드 표준화 단체인 IBTA(InfiniBand Trade Association)에서는 인피니밴드 관련 표준화를 진행하고, 실제 개발은 Openib에서 이뤄지고 있다. 그림 1은 OpenIB에서 배포한 프로토콜 계층 구조에 대한 그림이다. 응용 계층에는 VIPL(Virtual Interface Provider Library), MPI(Message Passing Interface) 그리고 소켓이 있고, 미들웨어 드라이버로는 SDP, SRP, IPoIB 등이 있다. 커널을 통해 HCA에 접근하는 방식과 RDMA를 이용한 커널을 by-passing하여 HCA에 접근하는 방식을 이용해 메세지를 전송한다.

성능평가를 위한 테스트 툴은 HP에서 만든 Netperf를 사용했다. 두가지 버전의 Netperf를 사용했는데 하나는 기가빗 이더넷을 측정하기 위한 버전과 또 다른 하나는 Netperf를 수정하여 SDP를 측정하기 위한 NetPerf_IB 버전이다. 전자는 원격지 노드와 연결을 시도하기 위해 소켓 구성시 프로토콜 패밀리 가운데 AF_INET 24를 사용하도록 하고, 후자는 소켓을 구성할 때 새로운 프로토콜 패밀리를 추가하였는데 AF_INET_SDP 26을 사용하도록 수정했다[8].

성능 비교를 위해 기가빗 이더넷과 인피니밴드에서 또 다른 미들웨어 드라이버인 IPOIB를 포함 시켰다. 현재 IPOIB는 IETF에서 표준화를 진행하고 있는데 이 단체에서는 인피니밴드 망에서 IPv4/v6을 통한 데이터 전송이 가능하도록 프로토콜과 인캡슐레이션 방식에 대해 정의하고, 인피니밴드 망 자체의 관리가 가능하도록 MIB 객체에 대한 표준화를 진행하고 있다[12].

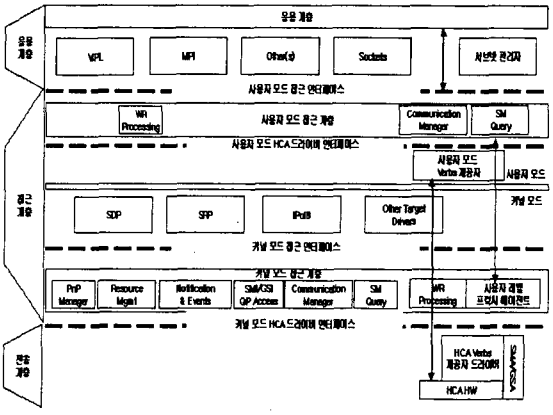

그림 1. OpenIB 인피니밴드 프로토콜 스택

3.1 패킷 처리량

그림 2-3은 SDP, IPOIB 그리고 기가빗 이더넷에 의해 얻어진 패킷 처리량 결과이다. 성능 평가 방법을 위해 메세지를 크기별로 구분하였는데 절대적인 방식으로 구분한 것이 아니고 임의적으로 패킷 처리량의 변화에 따라 작은 사이즈의 메세지, 중간 사이즈의 메세지 그리고 큰 사이즈의 메시지와 같이 세 가지로 구분하여 측정하였다.

메세지 크기에 따른 패킷 처리량에 대해 두드러진 결과를 보이는 것은 SDP이다. 작은 사이즈의 패킷 처리량에서는 메세지 크기가 16바이트에서 128바이트까지는 기가빗 이더넷에 비해 패킷 처리량이 적지만 256바이트 부터는 기가빗에 비해 패킷 처리량이 많다는 것을 알 수 있다. 그 이유는 기가빗 이더넷 비해 수신 가능 메세지나 송신 가능 메세지 등에 의해 처리 해야할 패킷이 많기 때문이다. 중간 사이즈의 패킷 처리량에서는 4096바이트에서 최고의 패킷 처리량을 보이고 있는 데, 그 이유는 x86의 리눅스 기반에서는 메모리의 한 페이지 크기가 4096이므로 한 패킷의 크기가(MTU: Maximum Transfer Unit) 4096일 때 가장 효율적으로 세그먼테이션이 가능하기 때문이다. 큰 사이즈의 메세지 패킷 처리량에서는 기가빗 이더넷과 IPOIB에 비해 3배 이상의 패킷 처리량을 보이고 있다. 그리고, 기가빗 이더넷에 비해 IPOIB는 메세지 크기가 32K바이트 이상에서는 패킷 처리량이 많다는 것을 알수 있다.

기가빗 이더넷, IPOIB 그리고 SDP의 패킷 처리량에 대한 최적의 메세지 사이즈 는 그림 2에서 기가빗 이더넷이 256바이트 가장 많은 패킷 처리량을 보이고

있고, IPOIB는 그림 4에서 알수 있듯이 32K바이트, 그리고 SDP는 메세지 크기가 4K바이트 일 때 가장 효율적인 패킷 처리량을 보이고 있다.

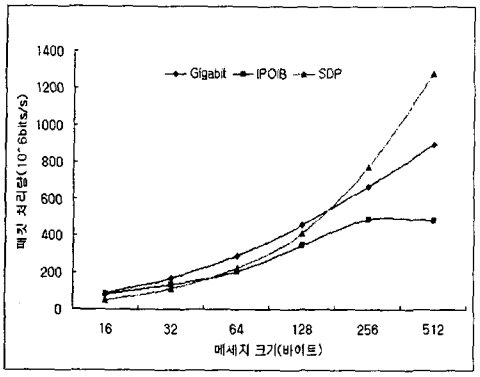

그림 2. 작은 사이즈 메세지에 대한 패킷 처리량

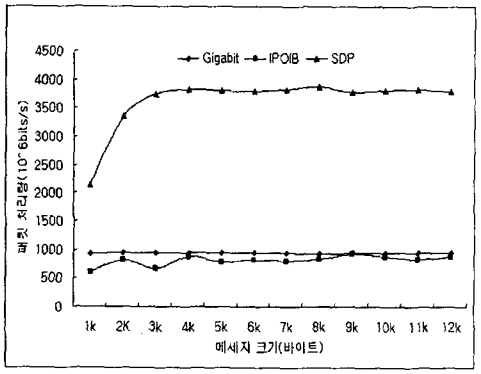

그림 3. 중간 사이즈 메세지에 대한 패킷 처리량

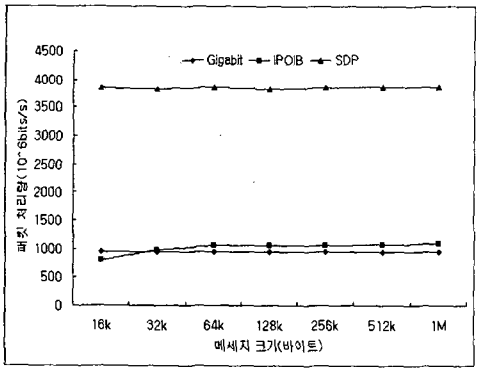

그림 4. 큰 사이즈 메세지에 대한 패킷 처리량

3.2 CPU Utilization

네트웍 트레픽에 대한 처리는 결국 호스트의 CPU 부하와 연관된 것이다. 기가빗 이더넷, IPOIB 그리고 SDP의 프로토콜 프로세싱을 위한 CPU의 부하를 3.1절에서 측정한 패킷 처리량에 따라 각 메세지의 크기별로 CPU 부하를 측정하였다. CPU에 대한 부하를 최소화시켜 CPU상에서 동작하는 어플리케이션에게 가능한 더 많은 CPU time을 부여하는 것이 다. 그림 5와 6은 메세지 크기에 따른 CPU Utilization에 대한 측정 결과로 작은 사이즈의 메세지에 대해서 SDP가 상대적으로 높은 CPU 사용도를 보이고 있다. 그리고, 큰 사이즈의 메세지에 대해서도 기가빗 이더넷에 비해 다소 높은 CPU 사용도를 보이고 있다.

메세지 크기에 따른 CPU Utilization을 1%의 CPU Utilization 당 패킷 처리량으로 나타냈을 때 그림 7에서 볼 수 있듯이 IPOIB나 SDP에 비해 기가빗 이더넷이 CPU 1% Utilization에 대한 패킷 처리량이 높은 것으로 나왔다.

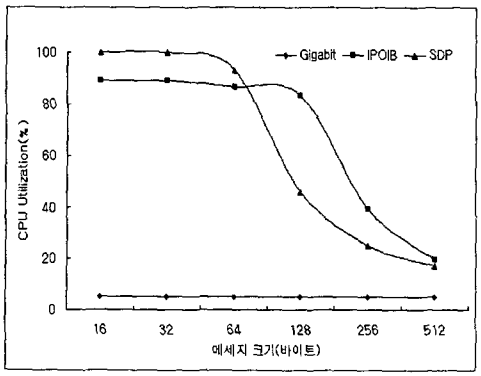

그림 5. 작은 사이즈의 메세지에 대한 CPU Utilization

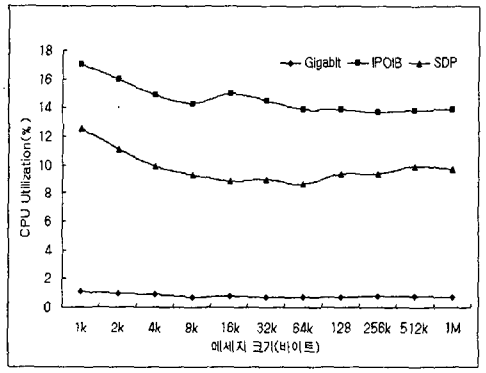

그림 6. 큰 사이즈의 메세지에 대한 CPU Utilization

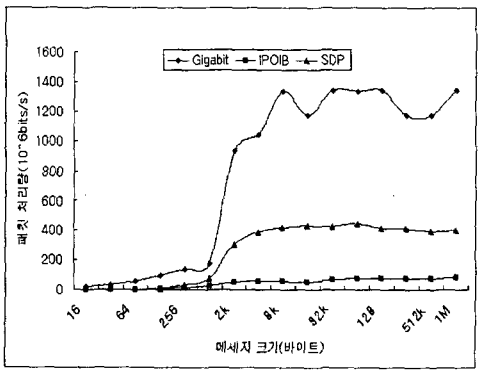

그림 7. CPU 1% Utilization 당 패킷 처리량

4. 결론

본 논문에서는 현재 개발 단계에 있는 인피니밴드가 고성능의 네트웍에서 어떠한 장점과 단점을 가지고 있는지를 알아보는 것이었다. 특히, 인피니밴드의 목적은 데이터 입출력이 많은 IDC(Internet Data Center)에 사용될 계획으로 있어 사용자 요구에 따른

데이터 처리에 중점을 두고 있다[10]. 여기서는 인피니밴드 프로토콜 스택에서 SDP 미들웨어 드라이버를 이용함으로서 다른 프로토콜에 비해 상대적으로 높은 패킷 처리량을 보였다. 그러나 1%의 CPU Utilization 당 패킷 처리량에서는 다소 높은 수치를 보이고 있어 이부분에 대한 디버깅이 좀더 이뤄 져야 할 것으로 본다.

[참 고 문 헌]

[1] Infiniband Architecture Specification, Release 1.1 Infiniband Trade Association, 2002.

[2] S. Bhattacharya, S. Pratt, B. Pulavarty, and J. Morgan. Asynchromous I/O Support inLinux 2.5. In Proceedings of the Linux Symposium, pp.371-386, July 2003

[3] *Cohen, A*, "A performance analysis of the sockets direct protocol (SDP) with asynchronous I/O over 4x infiniband", 2004 IEEE International Conference , pp. 241-246, April 2004.

[4] S.N. Damianakis, C. Dubnicji, and E.W. Felten. "Stream Sockets on SHRIMP". In Lecture Notes in Computer Science 1199, pp.16-30, 1997.

[5] T. von Eiken, A Basu, V.Buch, and W. Vogels. "U-Net: A User-Level Network Interface for parallel and Distributed Computing". In Proceedings of the ACM Symposium on Operating Systems Principles, PP. 40-53, December 1995.

[6] P. Balaji, S. Narravula, K. Vaidyanathan, S. Krishnamoorthy, J. Wu, and D. K. Panda. "Socket Direct Protocol over infiniband: Is it Beneficial?" Technical Report OSU-CISRC-10/03-TR54, The Ohio State University, 2003.

[7] P. Balaji, P. Shivam, P. Wyckoff, and D.K. Panda, and J. Saltz. Impact of High Performance Sockets on Gigabit Ethernet. In Cluster Computing, September 2002.

[8] The Public Netperf, http://www.netperf.org

[9] J. Wu, P. Wyckoff, and D. K. Panda, PVFS over Infiniband: Design and Performace Evaluation, In ICPP, 2003.

[10] S. Narravula, P. Balaji, K. Vaidynathan, S. Krishnamoorthy, J, Wu, and D. K. Panda. Supporting Strong Cohency for Active Caches in Multi-tier Data-Center over Infiniband. In SAN, 2004

[11] Socket Direct Protocol. http://www.infinibanta.com

[12] IP over Infiniband(ipoib), http://www.ietf.org/html.charters/ipoib-charter.html