Myrinet 대역폭 향상을 위한 채널 본딩 기반 VI-GM 통신 모듈 개발

장기성*, 김상형**, 최현호**, 유원경***, 유관종*
*충남대학교 컴퓨터공학과
**충남대학교 컴퓨터과학과
***성신여자대학교 컴퓨터공학부
e-mail:zephy17@empal.com

A Channel Bonding based Vi-GM Communication Module Development for Myrinet Bandwidth Enhancement

Gi-Sung Jang*, Sang-Hyong Kim**, Hyun-Ho Choi**,
Weon-Kyung Yoo***, Kwan-Jong, Yoo*
*Dept of Computer Engineering, Chung-Nam University
**Dept of Computer Science, Chung-Nam University
***School of Computer Science & Engineering, Sung-Shin Women's University

요 약

클러스터 파일 시스템의 성능은 노드 내부 연산의 성능 뿐만 아니라 노드간의 통신 성능이 전체 시스템의 성능에 큰 영향을 미친다. 최근의 클러스터 파일 시스템에는 Myrinet, ServerNet, QNet, SCI(Scalable Coherent Interface) 등의 고속 인터페이스를 통해 연결하는 것이 일반화되어 있다 본 논문에서는 노드간의 통신 성능을 높이기 위해서 Myrinet 환경에서 제공해 주는 사용자 수준의 통신 프로토콜인 VI-GM(Virtual Interface Architecture over GM)을 사용하여 2개 이상의 네트워크 장치를 하나처럼 보이게 해서 Redundancy와 대역폭을 증가시키는 채널 본딩 기법을 기반으로 통신 모듈을 개발하였다. 그리고 성능 실험을 통해 제안된 모듈의 우수함을 보였다.

1. 서론

인터넷의 급속한 성장으로 대용량의 데이터를 효과적으로 관리하기 위해 가장 많이 사용되는 방법이 클러스터 파일 시스템을 활용하는 것이다. 기존의 TCP/IP를 이용한 통신 모듈은 여러 네트워크 계층을 거치면서 생기는 오버헤드와 데이터의 전송 및 수신 과정에서 커널이 직접 개입으로 System Buffer로의 잦은 데이터 복사와 시스템 콜 등의 클러스터 파일 시스템의 전체 성능 저하를 가져왔다. 시스템의 성능을 증가시키기 위한 새로운 통신 모듈이 필요하게 되었다. 그 결과로 사용자 수준 통신 모델들이 제시되었다. 본 논문에서는 SAN 환경의 클러스터 파일 시스템에서 많이 사용되는 Myrinet의 사용자 수준 프로토콜인 VI-GM을 사용하였다. 한편, 최근의 데이터가 대용량의 멀티미디어 데이터로 변화하면서 서버간의 통신 대역폭 확장의 필요성이 대두되었다. 대역폭 확장을 위한 방법으로 채널 본딩 메커니즘이 소개되었으며 이것은 여러 개의 네트워크 카드를 병렬로 구성하여 하나의 고속 네트워크 카드처럼 사용하는 방법이다. 본 논문에서는 채널 본딩 기반 VI-GM 통신 모듈을 구현함으로써 전체 성능 향상을 꾀하였다.

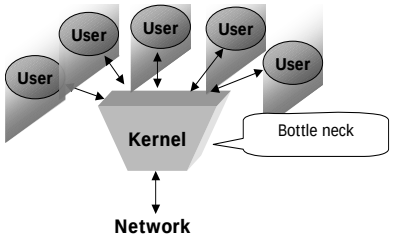
본 논문의 구성은 다음과 같이 이루어져 있다. 2장에서는 관련 연구로써 VIA, 채널 본딩의 특징과 구조를 살펴본다. 3장에서는 기존 클러스터 파일 시스템과 TCP/IP 문제점을 분석한다. 그리고 VI-GM 통신 프로토콜에 채널 본딩 개념을 도입하여 클러스터 파일 시스템의 통신 모듈을 설계 및 구현한다. 4장에서는 기존 시스템과 제안한 시스템의 성능을 실험하고 결과를 분석한다. 마지막으로 5장에서는 결

론을 맺는다.

2. 관련 연구

2.1 VIA

지속적인 네트워크 기술의 발달로 인하여 데이터 송·수신 과정에서 네트워크의 대역폭은 높아지고 지연 시간은 짧아짐에 따라, 네트워크로부터 전달받은 데이터를 어플리케이션의 버퍼에 복사하는 경로나 어플리케이션의 데이터를 네트워크로 전달하는 경로인 호스트 내부의 소프트웨어 구간이 새로운 병목 구간으로 등장하게 되었다. 즉, 어플리케이션 입장에서는 사용자 영역이나 커널 영역에 존재하는 여러 네트워크 계층으로 인하여 하부 네트워크가 제공하는 대역폭을 충분히 활용하지 못하는 새로운 문제가 발생했다. 또한, 네트워크 프로토콜은 일반적으로 운영 체제 내부에서 구현되므로 모든 데이터의 송·수신은 운영체제를 거쳐야 하며 그에 따른 사용자 공간(user space)과 커널 공간(kernel space)의 데이터 복사 또한 성능 저하의 또 다른 원인이 된다.



[그림 1] 새로운 병목 구간

VIA는 소프트웨어 또는 하드웨어로 구현되어 있으며, [그림 1]에서 볼 수 있듯이 NIC(Network Interface Card)가 사용자 영역의 버퍼로 직접 접근할 수 있게 하여, 메시지 복사와 문맥 교환이 TCP/IP와 비교하면 현저하게 감소되고, 기존의 통신 프로토콜이 가지는 전송 지연 현상을 줄였다[1].

2.2 채널 본딩

Beowulf 클러스터는 1994년 미국 NASA의 계약사인 CESDIS(Center of Excellence in Space Data and Information Sciences)에서 제작하였다. 비용도 절감하고 특정 벤더에 종속되지 않기 위하여 Beowulf 개발자들은 주로 리눅스를 운영체제로 채택하고 클러스터 내의 메시지 패싱은 표준 프로토콜을 이용한다.

Beowulf 채널 본딩 기술의 경우에는 가상의 디바이스 드라이버를 추가하여 여러 개의 네트워크 카드를 제어할 수 있도록 한 기술로서 TCP/IP 계층 아래에 구현된 것이다[2].

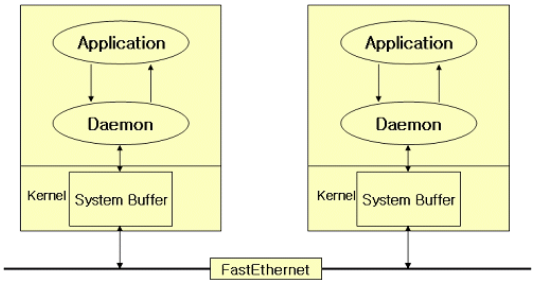
3. 채널 본딩 기반 VI-GM 통신 모듈

3.1 기존 통신 모듈

기존의 TCP/IP와 VI-GM을 기반으로 구현된 클러스터 파일 시스템의 문제점과 제안한 채널 본딩 기반 Vi-GM 전송 모듈에 대해 살펴본다.

3.1.1 TCP/IP 클러스터 파일 시스템

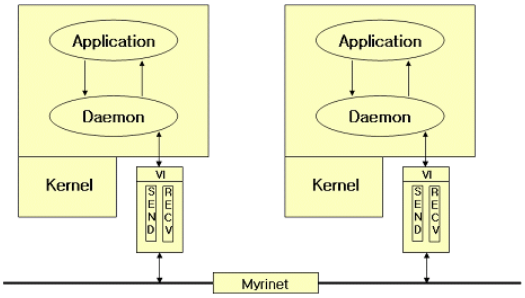
클러스터 파일 시스템에서는 노드간의 통신이 빈번히 일어나는데 기존의 TCP/IP를 이용한 통신은 여러 계층의 프로토콜 스택을 거치면서 생기는 오버헤드와 시스템 호출로 인해 생기는 오버헤드로 [그림 2]와 같이 네트워크 장치로 데이터를 전송하기 위해서 커널의 시스템 버퍼로 데이터 복사가 이루어져야 하는 오버헤드가 발행한다. 또한, 빈번한 시스템 콜이 발생하여 클러스터 파일 시스템 전체 성능의 저하를 가져온다[3].



[그림 2] TCP/IP 클러스터 파일 시스템

3.1.2 VI-GM 클러스터 파일 시스템

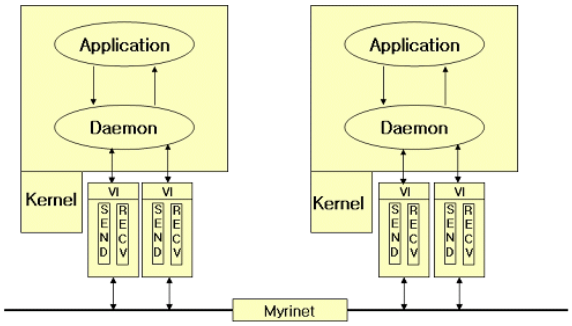
본 논문에서는 노드간의 초고속 통신을 지원해주는 Myrinet 환경에서 제공해 주는 사용자 수준의 통신 프로토콜인 VI-GM을 사용하여 클러스터 파일 시스템의 노드간 통신 모듈을 구현하였다. [그림 3]은 VI-GM 클러스터 파일 시스템을 나타낸 것이다. 네트워크를 통해 전달되는 데이터의 특성이 대용량으로 변하고 동시에 클러스터 파일 시스템을 동시에 접속하여 같은 작업을 요청하는 사용자도 늘어나게 되었다. 따라서 클러스터 파일시스템의 대역폭 확장의 필요성이 대두되었다[4].



[그림 3] 채널 본딩 기반 VI-GM 클러스터 파일 시스템

3.2 제안 통신 모듈

본 절에서는 제시된 기존 시스템의 문제점을 해결하고자 채널 본딩 메커니즘을 적용한 VI-GM 통신 모듈을 설계 및 구현하였다. [그림 4]는 VI-GM 통신 모듈에 채널 본딩 메커니즘을 결합한 채널 본딩 기반 VI-GM 클러스터 파일 시스템을 나타내는 그림이다.



[그림 4] 채널 본딩 기반 VI-GM 클러스터 파일 시스템

3.2.1 채널 본딩 전송 모듈 설계

본 논문에서는 트래픽의 측정으로 인한 오버헤드가 큰 Adaptive Distribution 방법과 특정 factor에 따른 불공정 분배가 발생하는 Static Distribution 방법을 채택하지 않고 동작 과정의 간단함과 데이터의 불공정 분배가 일어나지 않는 Round Robin 방식을 Data Distribution 정책으로 채택하였다.

· Round Robin Data Distribution의 첫 번째 단계는 본 논문의 시스템에서 네트워크 카드를 2개를 사용하였기 때문에 가장 먼저 VipOpenNic API를 이용하여 통신망 인터페이스 장치에 대한 접근권한을 얻어온다.

· 두 번째 단계는 Open한 통신망 인터페이스 장치에 대한 속성 정보들을 VipQueryNic 함수를 통해 얻어온다.

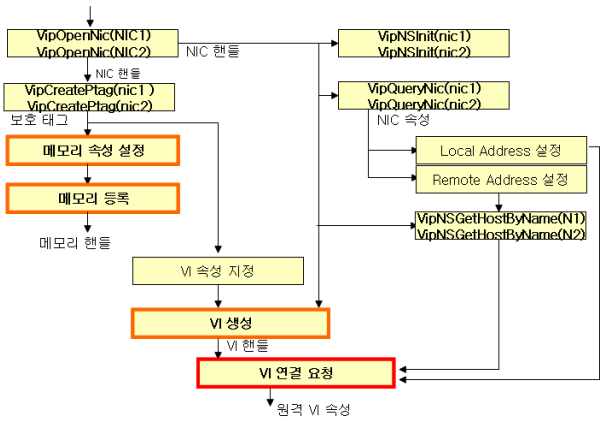
· 통신을 위해 필요한 주소 정보를 해당 변수에 메모리 할당하여 저장한다.

· VipNSGetHostByName API를 사용해서 Node1 NIC1- to -Node2 NIC1, Node1 NIC2- to -Node2 NIC2 연결을 설정한다.

· Round Robin 정책을 메모리 등록 과정에서 적용하는데 Node 1 NIC1- to -Node 2 NIC1 데이터 통신을 위한 메모리 영역을 할당하고 Node 1 NIC2 - to -Node 2 NIC2 데이터 통신을 위한 전용 메모리 영역을 연속되게 할당한다.

· 각각의 주소 정보와, 길이 정보, 속성 정보를 가지고 NIC1과 NIC2에 가상의 연결 통로인 VI를 VipCreateVi API를 통해 생성하여 D1은 NIC1의 VI에 D2는 NIC2의 VI에 각각 할당한다.

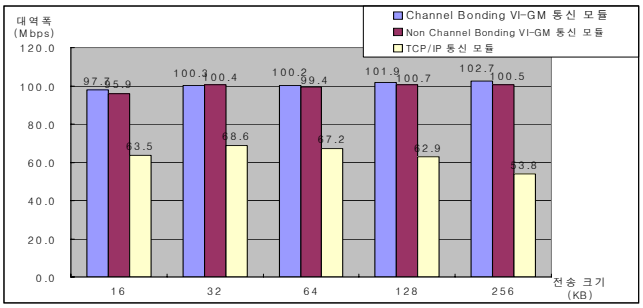
위와 같은 과정으로 Round Robin Data Distribution 과정이 일어난다. 아래 [그림 5]는 채널 본딩 전송 흐름을 나타낸 그림이다.



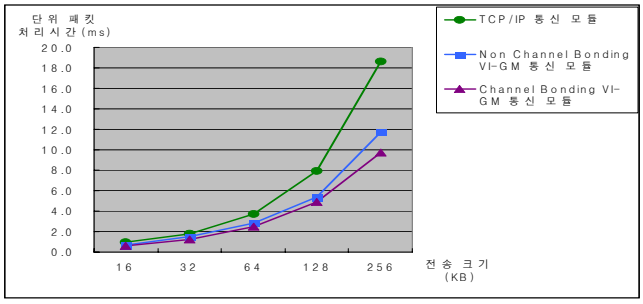
[그림 5] Round Robin Data Distribution 과정

4. 실험 및 결과 분석

제안된 모듈의 성능 측정 실험은 클러스터 파일 시스템의 특성으로 인해 총 5 구간(16KB, 32KB, 64KB, 128KB, 256KB)의 패킷 사이즈별로 128MB 파일을 import 하는 실험을 통해 성능을 측정하였다. 기존의 TCP/IP, VI-GM 그리고 본 논문에서 제안한 채널 본딩 기반 VI-GM 통신 모듈을 각각 대역폭 및 지연 시간을 측정하였다. [그림 6]과 [그림 7]은 각각 TCP/IP, VI-GM, 채널 본딩 메커니즘을 적용한 VI-GM으로 통신 모듈을 구현한 클러스터 파일 시스템의 대역폭과 단위 패킷 처리 시간을 측정한 결과이다.

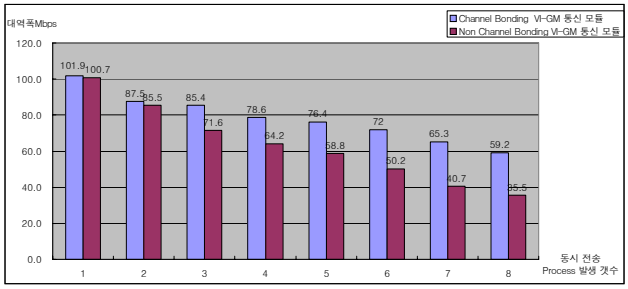


[그림 6]단일 전송 Process 발생 시 대역폭

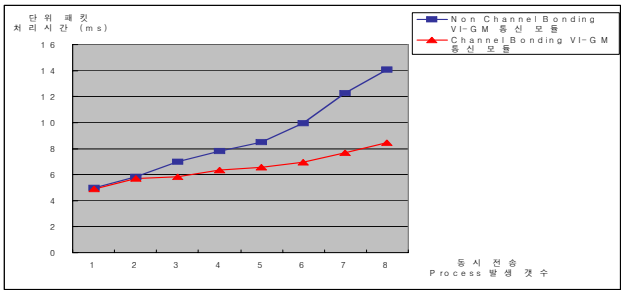


[그림 7] 단일 전송 Process 발생 시 단위 패킷처리시간

[그림 8]과 [그림 9]는 importing 서비스를 동시에 사용하는 Process를 늘려감에 따라 각각 채널 본딩 메커니즘을 적용한 VI-GM 통신 모듈과 적용하지 않은 통신 모듈의 대역폭과 단위 패킷 처리시간을 측정한 결과이다.



[그림 8]동시 전송 Process 증가에 따른 대역폭



[그림 9] 동시 전송 Process 증가에 따른 단위 패킷 처리시간

[표 1]에서 볼 수 있듯이 실험을 통해 채널 본딩 메커니즘을 적용한 VI-GM 통신 모듈이 성능이 우수함을 보인다.

[표 1] 동시 전송 Process 증가에 따른 대역폭 향상

동시 전송 Process 수	1	2	3	4	5	6	7	8
채널 본딩 메커니즘 적용한 경우	101.9	87.5	85.4	78.6	76.4	72	65.3	59.2
채널 본딩 메커니즘 적용하지 않은 경우	100.7	85.5	71.6	64.2	58.8	50.2	40.7	35.5
향 상 폭	1%	2%	19%	22%	29%	43%	60%	67%

5. 결 론

본 논문에서는 기존의 클러스터 파일 시스템의 노드간 통신 모듈이 소켓 기반의 TCP/IP를 사용함으로써 생기는 문제점으로 전체의 성능 저하를 해결하기 위해 Myrinet 환경에서 제공해 주는 사용자 수준 통신 프로토콜인 VI-GM에 비교적 쉽고 저렴한 비용으로 대역폭을 향상시킬 수 있는 채널 본딩 메커니즘을 적용하여 통신 모듈을 구현하였다. 본 논문에서 제안한 통신 모듈은 기본적으로 TCP/IP로 구현된 통신 모듈에 비해 뛰어난 성능 향상을 보였고, 채널 본딩 메커니즘을 적용한 통신 모듈의 경우는 동시에 전송 Process가 많이 발생하는 경우에 채널 본딩 메커니즘을 적용하지 않은 통신 모듈 보다 좋은 성능을 보여 1% ~ 67% 까지 클러스터 파일 시스템의 대역폭 성능 향상을 가져왔다. 이로써 본 논문에서 제안한 통신 모듈은 전송량이 동시에 많이 발생하고 대용량의 서버에 적합한 통신 모듈임을 증명하였다.

참고 문헌

[1]"Intel Virtual Interface (VI) Architecture Developer's Guide," Version 1.0, December 1997, http://developer.intel.com/design/servers/vi/developer/ia_imp_guide.htm

[2] http://www.beowulf.org/intro.htm

[3] D. A. Rusling. The Linux Kernel. Linux LDP, 1998.

[4] 장기성, 이주열, 유관종, 박의수, 최현호, 유원경, "Myrinet 상에서 VI-GM을 이용한 CFS의 효율적인 통신기법" 한국통신학회 하계종합학술발표회 논문초록집, 2006.