가상버스 네트워크를 위한 MPI/VBus

최현진[0] 김봉완 박규호
한국과학기술원 전자전산학과(전기 및 전자공학)
hjchoi@core.kaist.ac.kr kimbw@etri.re.kr kpark@core.kaist.ac.kr

MPI/Vbus for Virtual Bus Network

Hyun Jin Choi[0] Bong Wan Kim Kyu Ho Park
Dept. of Electrical Engineering & Computer Science
(Division of Electrical Engineering)

요 약

Wave Pipelining(WP) 기법을 적용한 가상버스(Virtual Bus) 네트워크 구조는 병렬 컴퓨팅에서 중요한 메시지의 긴급성을 효과적으로 지원한다. 우리는 효율적인 통신을 지원하는 가상버스에 적합하도록 MPI를 설계, 구현(MPI/VBus)하였다. 또한, MPI/Vbus와 디바이스 드라이버가 메시지 큐들을 커널 영역에서 공유하도록 설계함으로써 사용자 레벨 통신(User Level Communication)을 쉽게 구현할 수 있도록 했다.

1. 서 론

병렬처리에 있어서, 상호 연결망의 성능은 시스템의 성능에 직접적인 영향을 미친다. 고성능 네트워크 장비는 높은 성능과 함께, 병렬처리에서의 다양한 종류의 통신 방식들을 하드웨어적으로 제공할 수 있어야 한다. 가상버스는 새로운 개념의 통신방식을 도입한 고성능 네트워크 구조로서, 병렬 프로그램에서 빈번히 사용되는 모든 노드에 동시에 메시지를 보내는 방식인 multicast와 broadcast 통신을 하드웨어적으로 한번에 처리할 수 있는 구조로서, 긴급한 성격을 갖는 방송통신을 신속히 처리할 수가 있고, 다양한 종류의 통신 유형을 지원한다. 또한, Wave Pipelining (WP)이라는 새로운 통신기술을 라우터(병렬컴퓨터의 NI, Network Interface)에 집적하여, 고속으로 메시지를 송수신할 수 있다.

네트웍의 속도가 빨라지면서 local-area 통신의 병목은 NI의 대역폭, 즉 데이터 전송속도에서, 송수신단에서의 메시지가 전송되는 소프트웨어 path쪽으로 옮겨가고 있다. 특히, 전형적인 UNIX 네트워킹 구조에서는 디바이스 드라이버와 사용자 애플리케이션 사이에는 많은 추상 계층 레벨이 존재하고, 커널이 여러 번 메시지의 복사를 수행한다. 이로 인한 오버헤드는 통신 대역폭을 제한하고, 높은 end-to-end 메시지 latency를 가지게 한다. 이의 해결을 위해 등장하게 된 것이 사용자 레벨 통신(User level communication)이다.

MPI(Message Passing Interface)의 목적은, 간략히 설명하면, 메시지 전달 프로그램을 작성하는데 있어서 효과적이면서 유연하고 널리 사용되는 표준을 개발하는 것이다.

본 논문은 가상버스 네트워크 구조의 소개 및 그 위에서 설계되고 구현된 MPI/VBus에 대해 논할 것이다. 2장에서는 WP의 개념 및 효과적인 통신 방법을 제공하기 위해 도입한 가상버스의 개념을 소개하고, 3장에서는 이러한 네트워크 구조에 적합한 사용자 레벨 통신의 구조 및 구현 내용을 설명한다. 또한, 이를 바탕으로 한 가상버스의 MPI 설계 및 구현 내용을 다룰 것이다. 4장에서 결론을 맺기로 한다.

2. WP와 가상버스 구조

메시지 전송시에, 기존의 파이프라인 방식과는 달리, 중간의 노드들에서는 신호의 통로만을 형성해주고, 끝 단에서 단 한번의 래치 만을 하는 전송방식이 WP이다. 이 WP의 기술을 사용해, 기존의 파이프라인기술보다 4배이상의 속도향상을 얻을 수가 있다. [1]

WP의 속도는 끝단에 들어오는 신호들 사이의 최대, 최소 skew(차이, 이 skew를 wave-pipelining skew라 부를 수 있고, 이 skew를 얼마나 작게 조정할 수 있느냐가 디자인 이슈가 된다.), clock의 rise/fall time과 I/O 레지스터의 setup/hold 시간에 의해 제한된다. 이 wave-pipelining skew를 해결할 수 있는 메커니즘을 SSC(Skew Sampling Circuit)에서 제공한다. [2]

메쉬 구조는 병렬계산에서 가장 많이 사용되는 구조 중 하나이다. Row와 column 버스를 가진 메쉬에서는, 다중의 독립된 버스 transaction이 동시에 존재할 수 있다. 이것의 사용 예가 [3]에 잘 나타나 있다. 그러나,

비록 이러한 dual 통신망 접근방법이 매우 낮은 latency와 collective communication facility를 제공한다고 해도, 이 방법은 내부의 대역폭 낭비의 문제점을 가지고 있다. 예를 들어 단지 지역적인 통신만을 가지는 알고리즘에서는 버스를 사용하지 않을 수도 있다. 두 통신망을 효과적으로 사용하기 위해 프로그래머는 메쉬와 버스에서의 로드의 균형을 맞춰야 하거나, 또는 컴파일러가 통신의 형태를 찾아서 그것을 메쉬와 버스에 동등하게 분배해야 한다. 또한, 로드의 균형은 항상 가능한 것이 아니다. 이러한 문제점을 해결할 수 있는 다중의 broadcasting을 가지는 새로운 네트워크 구조가 가상버스(Virtual Bus)이다.[4] 그림 1은 가상버스의 개념을 잘 보여주고 있다.

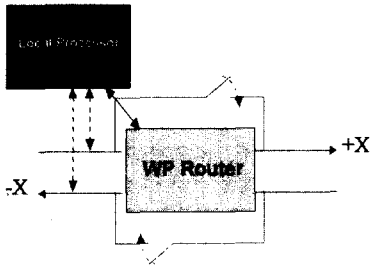

그림 1. 가상버스의 개념

가상버스는 병렬 네트웍에서 사용자의 요구에 따라 동적으로 버스망을 구성함으로써 빠른 멀티캐스팅이 가능하고 통신 대역폭을 효율적으로 활용한다. 가상버스는 방송통신 요청이 들어왔을 때, 기존의 일대일 통신을 일시 정지시키고, 가상의 버스를 네트웍 상에 만들어 통신을 하는 방식이다. 따라서 긴급한 성격을 갖는 방송통신을 신속히 처리할 수가 있고, 일대일 통신도 잠시 멈춰 섰다가 다시 진행함으로써, 다양한 종류의 통신 유형에 빠른 통신을 제공하게 된다. 가상버스 데이터는 WP 기법을 사용하여 넓은 대역폭으로 전송된다. 버스 데이터의 전송 후에, 동결됐던 메시지들은 자신의 destination을 향해 계속 진행한다. 기존의 wormhole이나 cut-through 라우터 구조와 연결구조를 변형함으로써, 가상버스의 기능을 가진 메쉬 통신망을 구현할 수 있다.

3.가상버스 네트워크를 위한 MPI/Vbus

3.1사용자 레벨 통신

사용자 레벨 통신(User Level Communication)은 송수신시에 커널의 개입없이, 사용자 영역과 네트워크 인터페이스 사이에서 직접 통신을 하는 것이다. [5]

그림 2는 가상버스 라우터 위에 설계된 DMA를 통한 사용자-레벨 통신의 동작을 설명하고 있다.

송신시의 디바이스 드라이버의 동작은 다음과 같다.

전달된 user 영역의 송신 버퍼는 아무런 조치를 취하지 않으면 커널에 의해 swap out이 될 가능성이 있다. 따라서 드라이버는 이 user 영역을 DMA로 다 전송되기전까지 page out이 일어나지 않도록 고정시킨다. 그리고 user

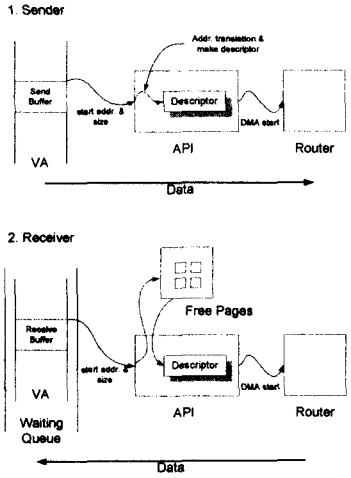

그림 2. 사용자 레벨 통신의 동작

address와 전송사이즈를 바탕으로 descriptor list를 작성하고, 라우터에 있는 DMA 제어기에 DMA 시작을 알린다. DMA 전송이 끝나면 API는 page out이 일어나지 않도록 고정시킨 user 영역을 풀고, 사용자 레벨 통신을 마친다. 수신시의 API의 동작은 다음과 같다. User 애플리케션이 수신 버퍼 영역을 넘길 때마다 API는 해당하는 descriptor list를 만들어 놓는다. 메시지가 NI를 통해 들어올 때, 그 메시지의 Mapping Key를 보고 해당 프로세스의 descriptor list를 찾은 다음, 이미 수신영역의 descriptor 리스트가 마련되어 있으면, 그 곳으로 바로 데이터를 전송하고, 아직 그 메시지의 수신버퍼가 마련되어 있지 않으면 (병렬 프로그램에서 메시지가 먼저 도착하는 경우가 있다.) 이미 할당해 놓은 free buffer로 데이터를 전송한다. free buffer에 들어온 데이터는 나중에 커널의 도움을 받아, 그 메시지를 요구하는 process의 virtual memory로 할당된다.

3.2 MPI/VBus의 설계 및 구현

그림 3은 MPI/VBus의 일반적인 구조를 보여주고 있다.

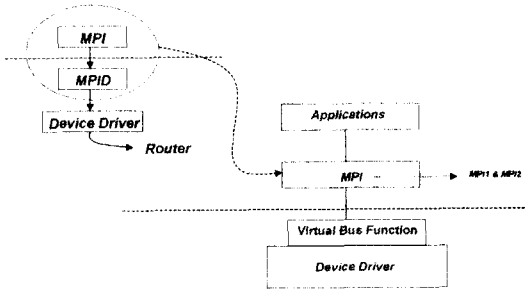

그림 3. MPI/VBus 의 일반적인 구조

MPI/VBus는 가상버스의 multicast와 broadcast scheme을 효율적으로 사용한다.

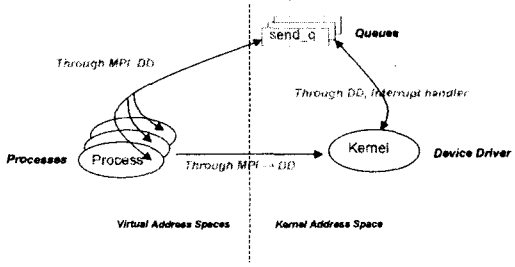

그림 4. 공동 큐를 통한 MPI/VBus1과 디바이스 드라이버의 관계

MPI 레벨은 실제로 장치 (네트워크 인터페이스)에 독립적이며, 애플리케이션 또는 컴파일러에서 사용하는 인터페이스이다. MPID 레벨은 장치에 독립적이며, 디바이스 드라이버와 연계해서 가상버스를 지원한다.

그림 4는 공동 메시지 큐들을 통한 MPI1/VBus (정확히 말하면 MPID/VBus 레벨)와 디바이스 드라이버의 관계를 설명하고 있다. 메시지 큐들은 송신(send) 큐, 수신(post) 큐와 free 큐로 구성되며, 송수신 하는 메시지들의 정보를 담고 있다. free 큐가 필요한 이유는 미리 전송되어 오는 메시지를 임시 저장할 필요가 있기 때문이다. 사용자 레벨 통신을 하기 위해서, 프로세스와 커널이 이러한 큐들을 공동으로 볼 수 있도록 설계하였다. (사용자레벨 통신을 사용하지 않는다면, 이러한 큐들은 MPI 내부에 존재할 것이고, 커널에 송수신 버퍼를 마련해서, 송수신 메시지들을 우선 커널에 저장하고, 해당 프로세스가 활성화됐을 때에, 그 큐들을 조작하기 때문에 공동 큐들은 필요가 없다. 하지만 그에 따른 오버헤드는 감수해야 한다.) MPI1 초기화 시에 이 공동 큐들은 커널상의 메모리에 만들어지며, 사용자 영역의 프로세스들은 메모리 맵핑 과정을 통해 이 큐들을 볼 수 있게 된다. MPI1/VBus는 프로세스의 요청시에 송수신 큐들에 메시지 정보를 등록하고, 시스템 호출을 통해 디바이스 드라이버의 루틴을 사용한다. 메시지의 송수신은 앞서 설명한 사용자-레벨 통신으로 이루어진다.

MPI2의 기능 중에서 one side 통신 기능을 가상버스에서 사용할 수 있도록 설계, 구현하였다. one side 통신은 Put, Get, Lock, Unlock 그리고 Fence 함수들로 이루어진다. MPI1에서는 (한쪽 노드에서의) Send 함수 호출에 대한, (상대편 노드에서의) Receive 함수 호출이라는 대응관계가 있는 반면에, MPI2는 한쪽의 노드가 송수신을 위한 모든 처리를 수행하는 구조로서, 공유메모리 프로그래밍 환경을 제공할 수 있다.

MPI/VBus는 사용자-레벨 통신을 지원하고, MPI2의 헤더에는 target 노드의 window에 대한 정보를 담고 있기 때문에, 이러한 Put/Get 요구를 디바이스 드라이버 레벨에서 처리할 수가 있게 된다. 따라서 MPID2로의 signal을 발생시키지 않아 훨씬 효율적이다. MPI2 메시지가 발생하면, 프로세스는 Put/Get/Lock 큐에 정보를 등록하고 MPI2/VBus의 함수를 호출한다.

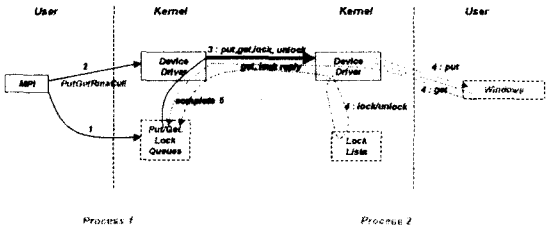

그림 5. MPI2/VBus의 동작

MPI1/VBus와 마찬가지로 MPI2/VBus도 큐가 비어 있지않다면, 즉 이전에 요청한 메시지에 대한 처리가 아직 다 끝나지 않았다면, 정보 등록 후 바로 리턴하여 소프트웨어 오버헤드를 줄일 수 있다. 디바이스 드라이버는 등록된 정보를 바탕으로 target 노드로 MPI2 메시지를 전송한다. target 노드의 디바이스 드라이버는 그림 5와 같이, Lock 요청시에는 해당 윈도우에 대한 Lock 리스트를 검사해, Lock 응답여부를 결정하고, Put 요청시에는 해당 윈도우에 DMA를 통해 데이타를 바로 업데이트하며, Get 요청시에는 해당 윈도우로부터 DMA로 데이타를 origin 노드로 전송한다.

4.결론

본 논문에서는 효율적인 통신을 지원하는 WP 및 가상버스 네트워크를 소개하였다. 또한, 효율적인 통신을 지원하는 가상버스에 적합하도록 MPI를 설계, 구현하였다. (MPI/VBus) MPI/Vbus는 사용자-레벨 통신을 사용하며, 시스템 호출의 횟수를 줄여 소프트웨어 오버헤드를 줄인다. 또한 MPI/VBus는 가상버스의 방송통신을 효율적으로 사용한다.

참고문헌

[1]J. Duato, F. Silla, and S. Yalamanchili. "A high performance router architecture for interconnection networks," In 1996 *In-ternational Conference on Parallel Processing*, pages I-1-68, 1996.

[2]B.W. Kim, H.J.Choi, K.-I.Park, J.H.Choi and K.H.Park. A Skew-Tolerant Wave-Pipelined Router on FPGA. *Proceedings of Hot Interconnect 7*, 1999

[3]D. Parkinson, D. J. Hunt, and K. S. MacQueen, "The AMT DAP 500," *Proc. of the 33rd IEEE Computer Society Int'l Conference*, pp. 196-199, 1998.

[4]J.H.Choi, B.W.Kim, K.H.Park, and K.-I.Park. "A bandwidth-efficient implementation of mesh with multiple broadcasting," *In Proceedings of Int' l Conference on Parallel Processing*, 1999.

[5]T. von Eicken, A. Basu, V Buch, and W. Vogels. "U-Net:A User-Level Network Interface for Parallel and Distributed Computing", *Proc. Of the 15th ACM Symposium on Operating Systems Principle*, 1995