멀티미디어 데이터 전송을 위한 EPVM의 기능 개선

윤종원, 심재홍, 최경희, 김재훈
아주대학교 정보 및 컴퓨터 공학부

Functional Improvement of EPVM for Transferring Multimedia Data

Jong-Won Yun, Jae-Hong Shim, Kyung-Hee Choi, Jai-Hoon Kim
Division of Information & Computer Engineering, Ajou University

요 약

복잡한 메시지 패싱 경로를 가지고 있는 PVM의 성능을 개선하기 위하여 공유 메모리를 이용해 패싱 경로를 간소화하고, 사용자 수준의 통신을 이용해 리모트 호스트와의 통신 성능을 향상하고자 EPVM이 설계되었다.[1] 기존 EPVM은 좋은 성능을 보이지만 공유 메모리 크기보다 작은 메시지만 전송된다. 이는 멀티 미디어 데이터 같이 용량이 큰 데이터를 전송하는데 적절하지 않다. 본 논문에서는 EPVM의 전송 기능을 멀티미디어 데이터 같은 대용량의 데이터에 적절하게 개선하였다.

1.서론

PVM은 병렬 프로그래밍을 위한 미들 웨어로 많이 사용되어 왔다. 그러나 기존의 PVM은 리모트 호스트로의 메시지 전달 시에 메시지 복사가 여러 번 발생한다. 그래서, 기존연구에서[6] PVM을 이용한 병렬 컴퓨팅 시스템의 성능을 향상시키기 위해, 공유 메모리를 사용해서 메시지 복사의 횟수를 감소 시켰다. 그리고, 더 나아가 커널에서 발생하는 메시지 복사를 없앴다. 이렇게 하여 개선된 EPVM(Enhanced Parallel Virtual Machine)이 설계되었고, EPVM은 일반적으로 좋은 성능을 보여주고 있다. 그러나, 메시지의 크기가 공유 메모리의 용량을 초과하면, 메시지가 전송되지 않는 문제점이 있다. 본 논문에서는 이러한 단점을 개선시키는 방법을 제시하고, 구현하였다.

2. EPVM

서두에서 밝힌 것처럼, EPVM은 공유 메모리를 이용해 PVM을 개선시킨 것이다. 기존의 PVM은 상당히 복잡한 메시지 전송경로를 가지고 있고, 여기서 발생하는 부하 때문에 성능이 그리 뛰어나지 못하다. 그래서 EPVM은 이런 PVM의 복잡한 메시지 전달경로를 공유 메모리를 이용해 간략화 해서 성능을 향상시켰다. 그리고, 리모트 호스트로 메시지 전송시 커널의 스택 영역에 메시지가 복사되는 부하마저 줄이기 위해, Unet에서 사용하는 Bigphysarea(그림 1)를 적용시켰다. [2]

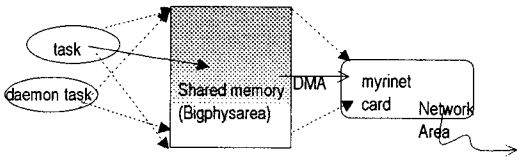

그림 1. EPVM의 공유 메모리

EPVM에서 sender는 전송하려는 메시지를 자신의 공유 메모리에 packing 시킨다. pvm_send라는 함수는 receiver의 수신박스에 packing된 메시지가 어디에 있는지 알려준다. 그러면, receiver는 pvm_recv를 이용해 sender의 공유 메모리에 packing된 메시지를 자신의 로컬 메모리로 가지고 오면서 unpaking 시킨다. (그림 2)

EPVM은 기존의 PVM이 가지고 있는 API를 변형시키지 않고 그대로 이용이 가능하게 되어있다. 그래서 이전에 만들어 놓은 PVM application이 변경없이 이용가능하고, 기존의 PVM에 비해 상당히 우수한 성능을 보여준다[6].

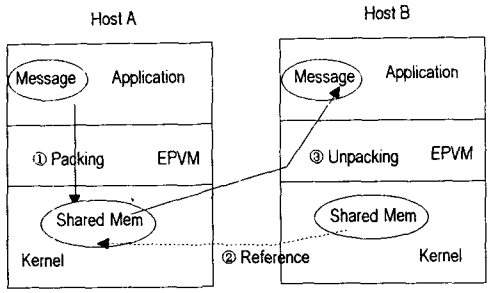

그림 2. EPVM 의 메시지 전송

3. EPVM 의 기능 개선

3.1 EPVM 의 문제점

위의 표에서 처럼, EPVM 은 기존의 PVM 에 비해, 좋은 성능을 보이지만 개선이 필요한 부분이 있다. EPVM 은 각 호스트에 할당된 공유 메모리의 크기보다 큰 메시지의 전송이 되지 않는다. 현재 EPVM 은 메시지를 전송하기 위해 미리 필요한 양의 공유 메모리를 할당을 받은 뒤에 메시지를 packing 해서 공유 메모리에 위치시키고 전송을 한다. 만일 공유 메모리가 모자란 경우 자신에게 할당된 공유 메모리를 모두 free 시켜서 다른 태스크가 사용할 수 있게 한다. 그리고 자신은 요청한 양의 공유 메모리가 할당되기를 기다리면서 blocking 상태로 들어가게 된다. 그러므로 공유 메모리보다 큰 메시지를 전송하려고 하면 요구한 메모리를 할당 받지 못한 채 계속 blocking 상태로 빠져 있게 된다. 물론 application 프로그램에서 메시지를 공유 메모리의 크기보다 작게 잘라서 전송을 한다면 이것은 문제가 되지 않겠지만, 기존의 PVM 과 완전한 호환성을 유지 한다는 원칙과 어긋나게 된다. 그래서 기존의 PVM 어플리케이션과의 호환성을 유지하기 위해 공유 메모리의 크기보다 큰 경우에도 전송이 되도록 수정이 필요하다.

3.2 EPVM 의 개선 방법

EPVM 이나 PVM 은 모두 메시지를 전송하기 전에 메시지를 XDR 포맷으로 packing 을 한다. 그 과정에서 PVM 은 자신의 로컬영역에, EPVM 은 공유 메모리 영역에 메시지를 packing 시킨다. 메시지를 packing 시키면서, EPVM 은 페이지 단위로 메시지를 잘라서 리스트를 구성해 관리한다. 그러므로 한번에 전송하기 위해 메시지 전체 크기에 해당하는 공유 메모리를 할당 할 필요는 없다. 그리고 메시지의 일부를 먼저 packing 해서 전송하고 receiver task 가 받으면 다시 나머지부분을 받는 방식으로 구현할 수 있다.

그러나 여기서 문제는 packing 하는 것과 전송하는 부분이 나누어져 있다는 것이다. 즉 메시지의 일부를 packing 한 뒤 전송하고 다시 메시지의 일부를 packing 하고 전송하는 방식으로 구현이 되면 기존의 PVM 의 프로그램 방식과 달라진다. 왜냐하면, PVM 에서는 메시지를 send 하는 pvm_send 를 call 하기 전에 pvm_pk*계열의 함수를 호출해서 먼저 packing 을 한다. EPVM 에서도 packing 하는 함수를 호출하면, 메시지를 packing 하여 공유 메모리에 위치시킨다. 다음에 pvm_send 로 receiver task 에게 전송하려는 메시지의 위치등과 같은 정보를 준다. 그래서, 그림 2 에서 보여주는 것처럼 receiver task 는 pvm_recv 함수에서 한번에 unpacking 하면서 메시지를 자신의 로컬 영역에 가지고 온다.

기존 PVM 과의 호환성을 유지하고, 공유 메모리보다 큰 메시지를 전송하도록 기능을 개선하기위해서는 우선 PVM 태스크의 로컬영역에 메시지를 packing 을 해야 한다. 그리고, packing 된 메시지를 일정단위로 나누어서 공유 메모리로 전송하고 전송이 모두 되면, 다시 나머지 메시지를 다시 공유 메모리로 가져오는 것을 반복한다.

물론 이렇게 PVM 태스크의 로컬 메모리로 먼저 packing 을 하고, 일부를 잘라서 공유 메모리로 가져가 전송하면, 구현된 EPVM 의 성능을 저하시킬 수 있지만, 이전에 구현된 모든 프로그램과의 호환성을 유지하기 위해서 필요하다.

3.3 EPVM 의 개선

EPVM 의 통신 기능의 어떤 부분이 개선 되어야 하는지는 앞에서 설명했다. 앞에서 설명한 것을 구현하기 위해 수정된 것은 두 부분이다. 하나는 packing 하는 부분이고, 나머지 하나는 pvm_send 와 pvm_recv 함수이다.

우선 packing 하는 부분은 위에서 말한 것 처럼, 메시지의 크기를 측정해 공유 메모리보다 큰 경우, 로컬 영역에 packing 을 시킨다. 그리고 이것을 공유 메모리로 옮기고 전송하는 것을 반복하는데, 공유 메모리로 옮길 때는 memory mapped IO 를 이용한다.

메시지가 로컬 메모리에 packing 될 때, EPVM 은 메시지를 메모리의 페이지단위로 된 fragment 로 나눈다. 이렇게 packing 된 메시지를 전송하기 위해서 일정단위로 grouping 을 하고, 한 group 씩 전송한다. 그리고, receiver 태스크는 한 group 을 받아 자신의 로컬영역에 unpacking 을 한다.

이러한 방법으로 메시지를 전송하기 위해서는 send 태스크와 receive 태스크사이에 간략한 프로토콜이 필요하다.

Sender 태스크는 전송하려는 메시지를 보고, 몇 개의 group 으로 나뉘어지고, 마지막에 남는 fragment 가 얼마나 되

는지 계산해서 receiver 태스크에게 알려준다.

Receiver 태스크는 group count 와 남은 fragment 수를 receiver 쪽의 공유 메모리에 기록을 한다. Sender 는 처음에 하나의 group 에 포함된 모든 fragment 를 전송한다. 그리고, receiver 쪽에 있는 group count 가 감소되기까지 기다린다. Receiver 는 한 group 의 모든 fragment 를 받아내면, group count 를 감소 시켜서 sender 에게 받았음을 알린다. Sender 는 group count 가 감소 된 것을 확인하고, 다음 번 전송할 group 에 포함된 모든 fragment 를 전송한다. 이런 작업을 group count 가 0 이 될 때까지 수행한다. 0 이 되면, 하나의 메시지 전송이 종료된 것이므로 sender 는 group count 영역을 수정할 수 있는 권한을 갖게 된다. 그리고 더 전송할 메시지가 있다면, 다시 group count 를 기록해 주면서 위의 작업을 sender 와 receiver 간에 반복한다. (그림 3)

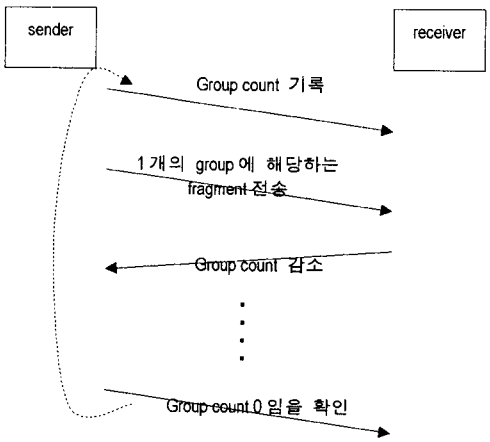

그림 3. 메시지 전송 프로토콜

4. 결론 및 향후 과제

EPVM 에서 대용량의 데이터 전송을 이루려면, 기존의 PVM 과는 약간 다른 방식을 적용해야 했다. 본 논문에서는 기존의 PVM 과 호환성을 가지면서 멀티미디어 데이터 같은 대용량의 데이터 전송에 적절한 기능 개선의 방법을 제시했다. 물론 로컬 영역으로의 packing 으로 공유 메모리보다 작은 메시지를 전송할 때와 같은 성능을 유지하지는 못한다.

위에서 설명한 것처럼 구현은 되었지만, 아직 추가로 결정해야 할 것들이 남아 있다. 우선은 하나의 group 을 몇 개의 fragment 로 구성하는 것이 효율적인지 분석적으로 또는 실험을 통해 결정되어져야 한다.

그리고 공유 메모리 할당에 대한 문제도 고려가 되어야 한다. 즉 여러 task 가 동시에 공유 메모리를 요구하여 공유 메모리가 모자란 경우, 어떤 task 를 선택해서 공유 메모리를 할당해 주는 것이 가장 효율적인지 연구할 계획이다.

Reference

[1] Geist, A. Beguelim, J. Dongara, W.Jiang, R. Mancheck, and V. Sunderam, *"PVM 3 User's Guide and Reference manual."* Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, Sept.1994

[2] Thorsten von Eicken, Anindya Basu, Vineet Buch, and Werner Vogels *"U-Net: A User-Level Network Interface for Parallel and Distributed Computing."* Proc. of the 15th ACM Symposium on Operating Systems Principles, Copper Mountain, Ccolorado, December 3-6, 1995

[3] Geist, G.A., Sunderam, V.S., *"Network-Based Concurrent computing on the PVM System."*, Concurrency: Practice and Experience, 4 (4):293-311, June 1992.

[4] M. Welsh and A. Basu and T. von Eicken, *"Low-Latency Communication over Fast Ethernet."*, Proc. EUROPAR 96, August 1996.

[5] M. Welsh and A. Basu and T. von Eicken, *"Incorporating Memory Management into User-Level Network Interfaces."*, Proc. Hot Interconnects V, August 1997.

[6] 윤종원, 심재홍, 문경덕, 최경희, 김재훈, 정기현, *"공유 메모리와 사용자 수준 통신을 이용한 PVM 성능 개선."*, 한국 정보과학회 98 추계가을 학술 발표 논문집 III, 제 25 권 2 호, 1998