범용 SMP 클러스터의 인터노드 통신을 위한 향상된 Push-Pull 메시지

김태훈 김성천
서강대학교 컴퓨터학과
sghun@archlab.sogang.ac.kr ksc@arqlab1.sogang.ac.kr

Advanced Push-Pull Messages for Internode Communication of Commodity SMP Clusters

Tae-Hun Kim Sung-Chun Kim
Dept. of Computer Science, Sogang University

요 약

대칭형 멀티프로세서 시스템으로 구성된 클러스터의 메시지 전송 방법은 인트라노드인 프로세서 통신과 인터노드인 시스템 통신을 동시에 수행하므로, 노드들간의 통신 성능을 위한 메모리 버퍼의 사용과 버퍼 사이의 데이터 중복 복사가 인트라와 인터노드 사이의 통신 불균형을 가져온다. 푸쉬-풀 메시지의 버퍼 사용 기법을 제한하고 메시지 전송 수행단계를 수정하여 고속 네트웍을 위한 인터노드의 통신 불균형을 감소시켰고, 주소 전환과 전송-승인 신호 중첩 기법을 고속 네트웍에 적합하도록 변형하여 기존의 푸쉬-풀 메시지 기법과 비교, 분석하였다. 제안된 기법은 인터노드 사이의 통신 지연을 약 7~18% 감소시켰다.

1. 서론

대칭형 멀티프로세서 클러스터의 성능은 하나의 객체 시스템인 인터노드(internode)와 그 객체안의 다중 프로세서인 인트라노드(intranode)들 사이의 통신 능력에 있다. 인터노드 통신을 위한 분산 환경에서의 메시지 전송은 송·수신의 비동기 문제로 버퍼링과 동기 통신을 위한 부가 단계가 필요하다. 이로 인한 메시지 전송의 통신 지연을 감소시키고 광역의 전송 대역을 확보하기 위해 푸쉬-풀(push-pull) 메시지 기법[1]이 제안되었다.

이 기법은 인트라노드의 메시지 전송에서 발생하는 버퍼간의 불필요한 메모리 복사를 제거하므로 인트라노드의 통신 지연 시간을 줄일 수 있지만 인터노드를 위한 부가적인 주소변환 오버헤드 제거가 선행되어야한다. 범용 대칭형 클러스터의 네트웍 속도가 급속히 증가하면서, 인터노드의 메시지 전송시 주소 전처리 단계가 통신 지연으로 인한 클러스터의 성능 저하 요인이 되었다.

기존 기법은 인트라노드의 통신 지연을 줄이기 위해 프로세스간, 또는 프로세스와 네트웍 인터페이스 카드의 공유 메모리를 제거하고 자료의 주소값을 갖는 제로 버퍼를 사용하였다.

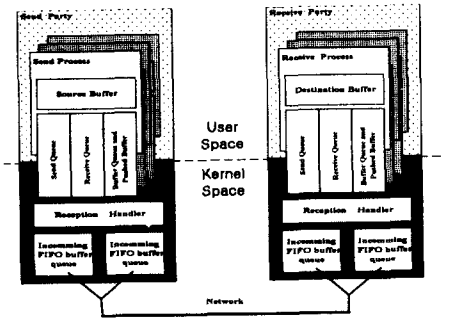
하지만 인터노드를 연결하는 네트웍의 속도가 100Mbit/s 이상일 경우에는 전처리 단계로 선행되는 주소변환 마스크 기법이 인터노드의 통신 지연을 초래하였다.

제한된 기법에서는 인트라노드와 인터노드의 통신 불균형을 제거하기 위하여 대칭형 멀티프로세서 시스템으로 구성된 클러스터의 푸쉬-풀 메시지 기법을 보완하였다. 고속 네트웍을 사용하는 클러스터에서 전처리로 수행되는 인터노드상의 주소변환 마스크 기법을 고려하여 제로 버퍼의 사용을 인터노드의 네트웍 송·수신 속도 정보에 맞춰 제한하였다. 네트웍 인터페이스 카드와 프로세스 사이의 주소 변환 시간이 고속의 네트웍 전송 시간에 비해 작아질 수 있도록 메시지 전송 단계를 수정하여 첫 단계로 보내지는 메시지 정보가 주소 변환 시간을 단축하도록 하였다.

2. 인터노드를 위한 Push-Pull 메시지 기법

송·수신 프로세스는 각각 사용자 공간에 위치한 자료 버퍼(source buffer)와 도착 버퍼(destination buffer)를 가지며, 각각의 프로세스는 세 개의 자료구조를 커널과 공유한다.

송신 큐(send queue)는 송신 오퍼레이션 정보, 수신 큐(receive queue)는 현재 수신 오퍼레이션 정보를 저장한다. 그리고 마지막 버퍼 큐와 푸쉬 버퍼(buffer queue and pushed buffer)는 현재 입력되고 있는 패킷들중 메모리의 목적지가 결정되지 못한 패킷들을 저장한다[그림 2].



[그림 2] 푸쉬-풀 메시지 기법

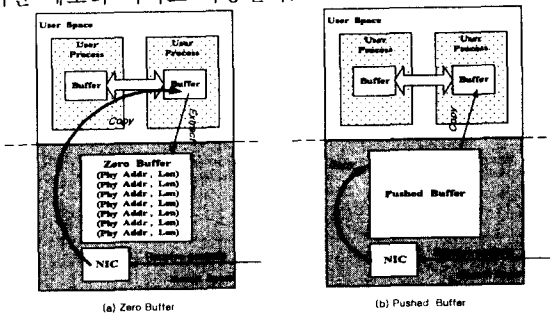
푸쉬-풀 메시지 기법에서 인터노드를 위한 전송 전략은 주소변환 오버헤드 차폐방법과 전송-승인 신호 중첩 방법이 있다. 주소변환 오버헤드 차폐는 인터노드 통신에서 주소변환 시간을 숨기기 위한 방법이다. 제로 버퍼를 사용할 때, 동일한 시스템 노드상에서 네트웍 인터페이스 카드로부터 도착 버퍼로의 자료 전송은 다른 부가적인 프로세스의 관여없이 커널에 의해서 직접 수행되지만 푸쉬-풀 메시지 기법에서는 제로 버퍼를 사용하기 위하여 전처리 단계인 주소변환 작업이 필요하다.

주소변환 오버헤드는 전송될 메시지의 크기에 따라서 선형적으로 증가한다. 주소변환 사용시간이 T_{tran}, 메시지의 크기가 M_{size} 일 때 $T_{tran} \propto M_{size}$ 이다. 일반적으로 통신 단계의 수행은 주소변환 전처리에 걸리는 시간보다 상대적으로 긴 지연 시간을 요구한다. 따라서 지연 시간 $T_{latency} \gg T_{tran}$ 이면, 오버헤드를 차폐하기 위한 주소변환이 푸쉬, 풀 단계에서 수행되는 모든 네트웍 통신 전에 이루어질 수 있다. 하지만 상대적으로 오랜 지연시간을 요구하던 인터노드의 통신이 고속 네트웍 장비로 지연 시간이 단축되고 있으며, 모든 주소변환도 안전하게 지연될 수 없다.

인터노드의 부가적인 메모리 복사를 줄이기 위해 개발된 제로 버퍼의 사용은 고속 네트웍을 사용하는 인터노드 통신의 경우 주소변환을 위한 전처리 단계의 필요로 통신 지연을 유발하는 단점이 있다. 즉 $T_{latency} \ll T_{tran}$ 이면 M_{size}에 상관없이 $T_{latency}$는 T_{tran}에 종속되어 전체 통신 성능은 주소변환 시간에 의해 지연된다. 푸쉬 단계에서 인위 전송된 메시지의 주소변환은 네트웍 전송 초기화 이전에 수행되어야 하므로, bytes_to_push을 포함한 M_{size}의 전송 지연 시간인 $T_{latencyM}$은 T_{tranM} 보다 작게 된다.

3. Advanced Push-Pull 메시지 기법

인터노드의 통신에서는 푸쉬 단계에 결정되는 메시지의 크기에 따라 전환 제로 버퍼의 사용여부와 상관없이 일정한 통신 지연 시간을 나타낸다. 고정적인 전환 제로 버퍼의 사용으로 인한 통신 지연이 발생되므로, 사용이 가변되는 전환 제로 버퍼가 필요하다. 전송되는 메시지 M_{send}의 크기가 size_of_pushed_buffer 보다 작을 경우, 주소변환 오버헤드 차폐 기법을 사용하지 않는다. 즉 제로 버퍼를 [그림 4]의 (b)와 같이 중간 버퍼링을 위한 메모리 버퍼로 사용한다.



[그림 4 가변 전환 제로 버퍼]

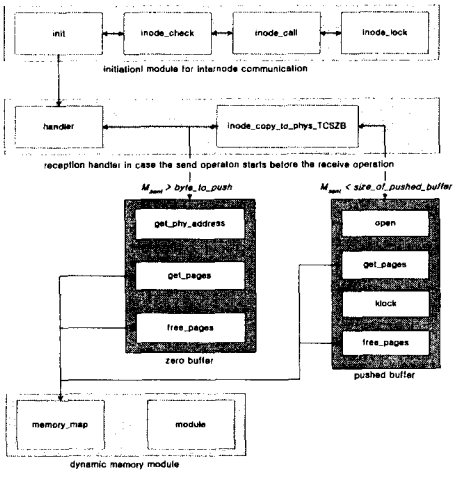
가변 전환 제로버퍼는 메시지가 bytes_to_push 메시지 크기보다 클 경우 [그림 4]의 (a)와 같이 전환 제로 버퍼의 역할을 수행한다. 이때 주소 변환 오버헤드 차폐 기법도 함께 실행된다. 전송되는 메시지가 bytes_to_push 메시지 크기보다 작을 경우에는 가변 전환 제로 버퍼가 중간 버퍼링의 역할을 수행한다. 주소변환 오버헤드 차폐기법은 차단되고 단지 푸쉬 버퍼의 역할을 수행한다. 레지스터 주소변환 오버헤드 차폐는 수신부에서 수신 오퍼레이션이 작동하고 있지 않을 경우, 메시지가 전송되면 주소 변환 오버헤드를 제거할 수 없는 치명적인 단점을 해결하기 위하여 초기 전송되는 메시지에 한하여 네트웍 인터페이스 카드에 존재하는 레지스터 사용을 허가한다. 수신 오퍼레이션이 준비되지 않은 상황에서 사용자 영역의 레시스터 사용은 메시지의 복사없이 도착 버퍼로의 직접 전송을 가능하게 한다.

4. 성능평가

성능 평가는 본 논문에서 인터노드 향상을 위해 수정된 전환 제로 버퍼 기법, 주소 변환 오버헤드 차폐 기법, 전송-승인 신호 중첩 기법이 적용된 향상된 푸쉬-풀 메시지 기법과 기존의 기법들로 구성된 푸쉬-풀 메시지 기법을 가지고 시뮬레이션을 수행하였다[3].

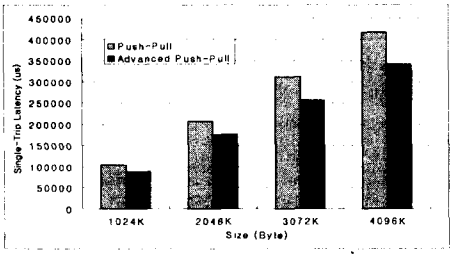
4. 성능평가

송·수신 관리 모듈은 메시지 전송에 관련한 모든 기법들이 포함되어 있으며 알고리즘은 제시된 TCSZB (Turn Cross Space Zero Buffer)로 구성된다. 즉 NIC 버퍼에 입력된 메시지의 크기(M_{sent})에 따라 TCSZB는 제로 버퍼의 역할과 푸쉬 버퍼의 역할로 각각 전환된다. 제로 버퍼의 경우(M_{sent} > *byte_to_push*)에는 get_phy_address를 통하여 NIC 버퍼와 도착 버퍼의 물리 주소를 계산한다. 물리 주소의 구조는 메모리 관리 모듈을 통하여 동적으로 할당된 구조체를 링크드 리스트로 연결하며 전송된 메시지를 저장한 키에 따라 검색한다.



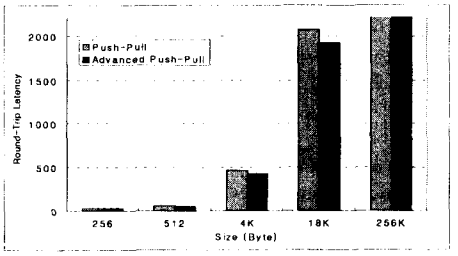
[그림 3] 시뮬레이션 알고리즘 프로그램 모듈

수행환경은 레드헷 리눅스 6.1에 500MHz Pentium III 시스템을 100Mbit/s 이더넷 카드로 연결하여 전송 지연 시간의 변화가 급격하게 변하는 512 bytes 메시지 크기가 시뮬레이션 환경의 bytes_to_push 변수값으로 설정되었다. 이 메시지의 크기를 기준으로 단방향 통신 지연 시간과 ping-pong 함수를 이용한 양방향 통신 지연 시간 측정을 수행하였다.



[그림 6] 인터노드의 단방향 통신 지연

인터노드의 양방향 통신 지연은 수신부와 송신부에서 각각 수행되는 ping-pong 함수를 통하여 측정하였다. 송신 대역폭으로 전송된 bytes_to_push 메시지와 수신 대역폭으로 전송되는 나머지 메시지의 왕복 도착시간을 이용하여, [그림 8]의 결과를 얻었다. 새롭게 제안된 기법은 초기 전송 메시지의 크기가 동일하므로 256, 512 Kbyte일 때 12%, 그 다음의 메시지 크기에 대하여는 평균 6%의 성능 향상을 가져왔다.



[그림 7] 인터노드의 양방향 통신 지연

5. 결과

고속 네트웍을 사용하는 클러스터에서 전처리로 수행되는 인터노드상의 주소변환 마스크 기법을 고려하여 초기 전송 bytes_to_push 메시지의 크기를 설정하고, 제로 버퍼의 사용을 인터노드의 네트웍 송·수신 속도 정보에 맞춰 제한하였다. 이 메시지 단계를 수정하여 기존의 방법에 비해 양방향 통신의 경우 최대 12%, 단방향의 경우 최대 18%의 지연 시간을 감소시킬 수 있었다. 주소 변환 시간이 짧아지는 고속 네트웍상의 단점과 리눅스에서 수행되는 클러스터의 인트라 노드를 위한 병렬 모듈 연구가 보안되면 보다 효율적인 메시지 기법을 제공할 수 있을 것이다.

6. 참고 문헌

[1] K.-P. Wong and C.-L. Wang. "Push-Pull Messaging: A High-Performance Communication Mechanism for Commodity SMP Clusters", *Proc. of the 1999 ICPP*, Wakamatsu, Japan, 1999

[2] C. M. Lee, A. Tam, and C. L. Wang, "Directed Point: An Efficient Communication Subsystem for Cluster Computing", *Proc. of the 10th ICPP* 1998.

[3] Bader DA and JaJa J, "SIMPLE: A methodology for programming high performance algorithms on clusters of symmetric multiprocessors (SMPs)", *Journal of Parallel & Distributed Computing*, V.58 N.1, pp.92-108, 1999. [외 13편]