패킷단위 병렬데이터 전송을 통한 MPICH-G2 집합통신 성능향상

이정희○ 한동수
한국정보통신대학원
{lake○, dshan }@icu.ac.kr

The Performance Improvement of Collective Communication in MPICH-G2 through Packet Parallel Data Transfer

Junghee Lee○ Dongsoo Han
Dept. of Engineering, Information and Communication University

요 약

본 논문에서는 MPICH-G2의 집합 통신 중 가장 많이 이용되고 있는 브로드캐스트 함수의 통신 속도를 개선함으로써, 분산 자원들의 통신속도를 높여 컴퓨팅 능력을 증가시켜주는 효과를 도모한다. 성능향상의 방법으로 패킷단위의 부분 데이터 송수신과 한 노드에서 여러 노드로 동시에 전송하는 두 가지 방법을 이용하였다. 제안된 방식을 사용했을 경우의 얻어지는 효과를 분석한 결과 데이터가 나뉘어진 패킷 수 p, 노드 수 n일 때, $O(p \times n)$ 에서 $O(p+n)$ 로의 성능향상을 나타냈고, bandwidth가 작은 노드로 인해 발생하는 병목현상이 감소되는 것을 확인하였다.

1. 서 론

대용량 데이터와 컴퓨팅 자원을 제공하는 그리드 환경은 WAN과 LAN환경이 혼재되어 있고, 자원마다 컴퓨팅 능력도 다를 수 있다. 따라서 그리드의 컴퓨팅 자원의 접근성을 높이기 위해서는 통신속도가 중요한 요소로 작용한다. 일반적으로 WAN환경에서는 네트워크 변동이 비교적 심하므로, latency와 같은 네트워크 정보를 이용한 스케쥴링이 전체 통신속도에 큰 영향을 준다. 또한 자원의 bandwidth와 같은 항목은 한번에 데이터를 보낼 수 있는 비율을 결정지으므로 통신 스케쥴링에 있어서 중요한 요소로 고려해야 한다. MPICH-G2 통신 함수는 통신하려는 프로세스가 같은 머신에 있지 않으면 TCP/IP를 사용한다[3]. 본 논문에서는 통신 컨트롤을 TCP/IP에 전담시키지 않는, MPICH-G2 라이브러리에서 본 논문에서 제시하는 방법으로 사용함으로써 통신성능 향상을 도모한다.

현재 MPI_Bcast는 송신자가 통신그룹에 참가한 나머지 수신자들에게 데이터 전송이 완료되면 끝나게 된다. 여기서 노드별 전송 과정은 한 프로세스가 데이터를 수신이 완료된 후 다른 노드로 전송을 시작한다. 시간이 많이 소요되는 비효율성을 극복하기 위해, 데이터를 패킷단위로 수신하면서 송신을 하는 방법과 한 노드에서 여러 노드로 동시에 전송을 하는 두 가지 방법을 이용해 MPICH-G2의 집합통신이 가지고 있는 문제점의 보완을 시도하였다. 또한 MPICH-G2에서 사용하는 기본 통신프로토콜인 eager protocol은 작은 데이터에는 효과적이지만, 긴 데이터에는 불리하게 작용한다. 패킷단위 브로드캐스트 방법은 이러한 점을 극복하도록 해준다. 전체 데이터를 수신 완료한 후 송신을 시작하는 방법이 $O(p \times n)$ 비용을 갖는 반면, 패킷 병렬 데이터 전송방법이 $O(p+n)$ 의 비용을 보여주었다. 데이터 크기가 클수록 더 큰 효과를 보여주었고, bandwidth가 작은 노드로 가는 전송을 다른 노드에도 동시 전송시킴으로써 bandwidth가 작은 노드로 인해 발생하는 병목현상을 감소시켜주었다.

2. 패킷단위 병렬 데이터 전송

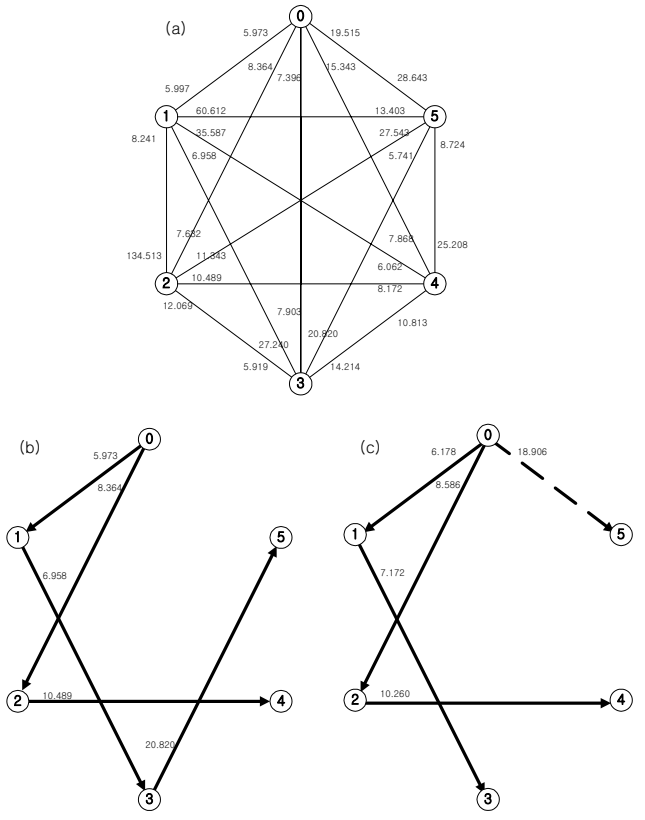
MPICH-G2의 집합 통신은 네트워크 Topology-aware 스케쥴링으로 성능을 향상시키고자 했다. 그러나 상세한 네트워크 정보를 이용한 통신트리 생성이 아니기 때문에, WAN처럼 네트워크 정보를 가변적이고 상대적으로 긴 latency를 가진 환경에서는 통신 시간이 많이 걸릴 수 있다. 따라서, 정확한 네트워크 정보를 이용해 스케쥴링하는 것이 전체 성능에 큰 영향을 미칠 수 있다. 이러한 점을 고려하여 latency를 이용한 통신트리 생성 방법인 ECEF(Earliest Complete Edge First)[3]와 ECEF를 이용해 두 개의 트리로 데이터를 전송하는 TTCC(Two-Tree Collective Communication)[2]의 휴리스틱 알고리즘이 제안되었다. 또한 MPICH-G2에서는 데이터 전송 컨트롤을 OS의 TCP/IP 스택에 맡기고 있다. 여기서는 효과적인 통신 컨트롤을 위해 MPICH-G2가 패킷 단위의 브

로드캐스트하는 방법과 네트워크 정보를 이용해 1:n 통신을 하는 방법을 제안한다.

2.1 패킷단위 브로드캐스트

현재 MPICH-G2의 MPI_Bcast의 구현은 각 호스트에서 전체 데이터 수신이 완료된 후 다음 호스트로 전송을 시작한다. MPI_Bcast의 전송속도를 빠르게 하기 위해 상대적으로 긴 전체 데이터의 수신이 완료된 후 다음 호스트로 전송을 시작하는 것이 아니라, 패킷 단위의 수신이 완료되면 바로 다음 호스트로 패킷을 송신하는 방법을 채택한다.[1] 보통 작은 크기보다는 큰 크기의 데이터 송신 세그먼트가 더 효과적이라는 것은 이미 알려진 사실이다. 한번에 보내지는 데이터 크기를 MTU 크기로 보낸다면, 네트워크 성능의 overhead없이 전체 데이터 송신 속도를 높일 수 있다.

예를 들어, ECEF 및 TTCC와 비교하기 위해 그림 1과 같은 데이터를 이용한다. 그림 1(a)는 latency정보를 나타내고, 그림 1(b)는 ECEF 알고리즘에 의해 생성된 통신 트리이고, 1(c)는 TTCC에 의해 생성된 트리를 보여준다 [2]. 그림 1(b)의 통신트리로 데이터 전송 과정을 살펴보자.



[그림 1] (a):latency정보 (b): ECEF트리 (3)TTCC트리

전체 데이터가 w,x,y,z의 4개의 패킷으로 나뉜다면, 전송되는 방법은 그림 2처럼 나타나게 된다. 각 노드는 각 패킷 w, x, y, z의 수신이 완료될 때마다 바로 다음 호스트로 전송을 시작한다. 데이터를 4개의 패킷으로 나누어 전송할 경우 Time 9만에 전송이 완료된다. 전체 데이터를 수신 후 전송할 경우 Time 16만에 완료하게 될 것이다.

각 패킷마다 Datatype의 Packing에 드는 비용이 크지 않다고 가정할 경우, 전체 데이터 크기를 M이라고 하고, MTU 크기를 m이라 하고 트리의 depth를 d라고 하자. 가장 극단적인 방법으로 linear 트리를 보면, 전체 메시지의 수신이 완료된 후 전송을 시작하는 방법의 비용은 latency가 아니라 단위 α 시간으로만 고려할 때 간단히 $\left\lceil \frac{M}{m} \right\rceil \times n \times \alpha$ 이고, 패킷 단위의 송수신 방법은 $(\left\lceil \frac{M}{m} \right\rceil + n - 2) \times \alpha$ 이 된다. 패킷단위의 통신에서는 통신 트리의 depth가 높을수록 유리하고, width가 넓어질수록 소요되는 시간이 길어진다. 이는 데이터 크기가 클수록 얻는 효과는 커지게 된다. 데이터가 MTU 크기의 4배라고 가정하고, 그림 1의 latency 정보를 고려해 송수신이 동시에 발생하는 방법으로 타임 테이블을 그려보면 그림 3(a)와 같게 나타난다. ECEF의 방법은 33.750M의 시간이 소요되는 반면, 패킷단위의 송신을 했을 경우 24.054M으로 9.696M의 시간이 절약된다.

	Node 0	Node 1	Node 3	Node 5	Node 0	Node 2	Node 4
Time 0	wxyz						
Time 1		w					
Time 2		wx	w				
Time 3		wxy	wx	w			
Time 4		wxyz	wxy	wx	wxyz		
Time 5			wxyz	wxy		w	
Time 6				wxyz		wx	w
Time 7						wxy	wx
Time 8						wxyz	wxy
Time 9							wxyz

[그림 2] 패킷 단위 통신

그러나 MPICH에서는 Datatype의 Packing에 드는 비용을 고려해, 패킷마다 packing 후 송신을 하지 않고, 전체 데이터를 packing한 후 송신한다. 따라서 브로드캐스트에서 패킷단위의 송수신을 할 때 얻을 수 있는 장점을 사용하지 못하고 있다. 또한 이 경우 한 노드에서 송수신이 동시에 발생할 수 있으므로 이로 인한 오버헤드를 고려해야 한다.

2.2 병렬 데이터 전송

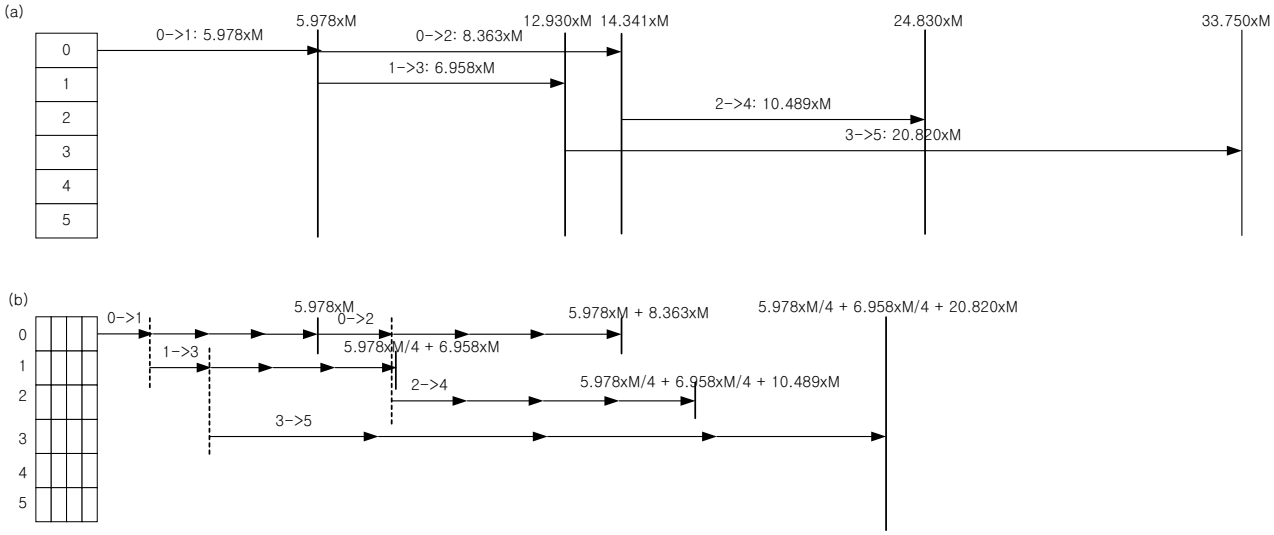
WAN 환경에서 네트워크 상황의 변동은 빈번하게 발생한다. 지연(latency) 정보만으로는 정확한 통신 트리를 만드는데 어려움이 있다. 실제 데이터 전송이 일어날 때

[1] worm hole routing 개념

bandwidth에 따라 전송률이 달라지기 때문에 통신속도에 영향을 미칠 수 있다. 호스트의 bandwidth를 고려해 송신 능력은 많지만 수신측에서 bandwidth를 허락하지 않는 문제가 있다. 이러한 문제를 극복하기 위한 방법으로, 1:1 송수신이 아닌, 1:n 송수신을 제안한다. 즉, 한 노드에서 한 번에 한 호스트로만 전송을 하는 것이 아니라, 여러 개의 동시에 여러 개의 호스트들에 전송하도록 하는 것이다. 이 방법에서는 여러 개의 노드에 전송할 때 사용가능한 bandwidth가 감소하게 되므로, 네트워크의 bandwidth와 latency 등을 고려해 최적의 n의 값을 구하여야 할 것이다. 예를 들어, 그림 1에서 노드 1의 bandwidth가 아주 작

[그림 4] (a): MPICH-G2 (b): 새로운 통신 방법

3. 결론 및 향후과제



고, 노드 0의 bandwidth는 충분히 클 경우, 노드 1에 병목현상이 걸릴 수가 있다. 이 때 노드 0에서 bandwidth가 허용하는 범위 내에서 노드 1과 노드 2에 동시에 데이터를 전송한다면, 노드 1로 인한 병목현상을 해소할 수 있고, 노드 2가 더 빨리 새로운 전송자가 되어 다음 노드로 전송을 할 수 있으므로, 전체적인 통신 속도를 향상시킬 수 있을 것이다. 패킷 단위로 1:n의 통신을 하는 것은 overlay 네트워크에서 사용하는 방법이기도 하다.

첫번째 방법에서는 통신 속도에 영향을 미치는 요소가 데이터 크기, latency, 통신 트리의 depth 등이다. 두번째 방법에서 중요한 요소로는 bandwidth와 latency가 될 것이다. 이 두가지 방법을 모두 이용해 각 요소를 정확히 고려한 스케쥴링을 한다면, 전체 데이터를 통신 그룹에 참가한 모든 노드들에 데이터를 송신하는데 걸리는 시간을 감소시킬 수 있다. 이 방법의 성능효과는 향후에 실험을 통해 입증할 것이다.

이를 요약하면 그림 4와 같이 간단히 표현할 수 있다. 그림 4(a)는 기존의 MPICH-G2에서 사용하는 방법으로 3단계의 송신으로 브로드캐스트를 완료한다. (b)는 본 논문에서 제안한 방법으로 패킷을 받을 때마다 전송을 하므로 (a)처럼 2가 아니라 1’로 표시하였다.

본 논문에서는 그리드 환경에서 MPI 인터페이스를 제공하는 MPICH-G2의 집합통신에 대한 성능향상에 관하여 다루었다. 대표적인 집합 통신 함수인 브로드캐스트에서 전체 데이터를 송수신하는데 걸리는 시간을 줄이기 위해 패킷단위의 브로드캐스트 방법과 병렬 데이터 전송방법을 이용해 성능향상의 가능성을 보여주었다. 2장에서는 TTCC에서 사용한 데이터를 이용해 ECEF와 TTCC를 패킷단위 브로드캐스트와 비교하였다. 또한 선행 연구인 ECEF와 TTCC알고리즘으로 생성된 트리를 이용해 본 논문에서 제시한 방법으로 통신했을 경우 나타나는 시간그래프를 살펴보았다. 이는 속도향상의 정도를 이론적으로 확인할 수 있었다. 데이터 크기가 클수록 얻을 수 있는 효과는 더욱 커질 것이다. 또한 ECEF의 경우 통신트리 생성시 bandwidth를 고려하지 않았다. TTCC의 경우 2개의 통신트리로 통신하기 때문에, 전체적으로 네트워크에 부하를 줄 수 있고, 두 개의 노드로부터 수신할 경우, 수신노드 선택의 결정을 TCP/IP가 해준다는 가정에 문제가 있다. 본 논문에서 제안한 방법은 이를 극복해준다.

앞으로의 과제는, 패킷 단위의 송수신과 1:n 통신의 특성에 맞춘 통신스케쥴링에 따라 실험을 통해 그 타당성을 검증하는 것이다. 패킷단위의 송수신을 할 경우 발생할 수 있는 datatype packing 오버헤드와 송수신을 동시에 할 경우 발생하는 오버헤드를 고려해보아야 한다. 또한 bandwidth정보와 latency를 이용해 통신 스케쥴링하는 알고리즘을 고안할 예정이다. 데이터 길이에 따른 성능분

석도 해볼 것이다. 또한 MPICH-G2에 두 가지 방법을 반영하여 성능테스트도 수행할 예정이다.

참고문헌

[1] Peter S. Pacheco, “ Parallel Programming with MPI,” Morgan Kautmann Publishers, Inc.

[2] Kwangho Cha, Dongsoo Han, and Chansu Yu, “ Two-tree collective communication”, Proceedings of the IASTED International Conference on Networks, Parallel and Distributed Processing and Applications, pp.30-35, Oct. 2003, Japan

[3] P.B. Bhat, C.S. Raghavendra, and V.K. Prasanna, “ Efficient collective communication in distributed heterogeneous systems,” 19th IEEE International Conference on Distributed Computing Systems, 1999.

[4] Thomas Dramitsch, “ Distributed Computations in a dynamic, heterogeneous grid environment,” Nov. 2002, Dissertation of university Potsdam

[5] http://www.mpi-forum.org