Hypercube 네트워크를 이용한 PC 클러스터

홍준표, 홍성범, 김영태, 이형봉*
강릉대학교 컴퓨터공학과

e-mail : hblee@kangnung.ac.kr

A Cluster of PC's using Hypercube-based Network

Joon-Pyo Hong, Sung-Bum Hong, Youngtae Kim, Hyung-Bong Lee*
Dept. of Computer Science & Engineering, Kangnung National University

요 약

본 논문에서는 네트워크 장비를 사용하지 않고 hypercube topology 를 기반으로 PC 들을 직접 연결하여 PC 클러스터를 구현하고 그 실용성을 검증한다. Hypercube 방식의 네트워크는 통신 장비를 사용하지 않기 때문에 저렴하고 보다 안정적이며, 네트워크 부하가 적어 높은 성능의 클러스터 시스템을 구축할 수 있는 것으로 알려져 있다. Hypercube 네트워크의 성능을 알아보기 위하여 다양한 성능 분석 툴 및 병렬 프로그램을 통하여 Gigabit 스위치를 사용한 네트워크와 비교/분석 한다.

1. 서론

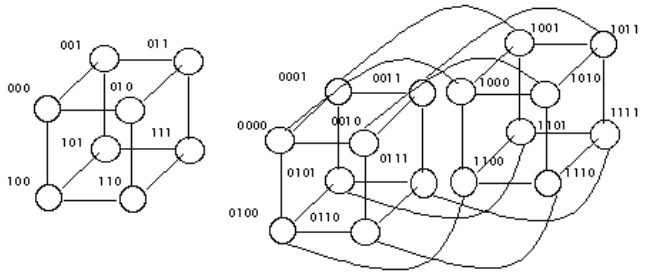
본 연구에서는 네트워크 장비를 사용하지 않고 hypercube topology 를 이용하여 PC 들을 직접 연결한 PC 클러스터를 구현하고, 그 성능의 효율을 네트워크 장비를 사용한 경우와 비교·분석한다. 이 때, 성능 분석을 위해 ping[1], netperf[1,2] 등의 통신 벤치마크 프로그램과 표준 병렬 라이브러리인 MPICH[3]를 사용한 병렬 벤치마크 프로그램을 사용하였으며, 실제 병렬 프로그램의 효율을 알아보기 위하여 두 개의 행렬을 곱하는 Cannon 의 알고리즘[4,5,6]을 사용하였다.

2. Hypercube 네트워크의 구현

2.1 Hypercube 네트워크

현실적으로 용이하게 hypercube 네트워크를 적용할 수 있는 PC 클러스터 형태는 (그림 1)과 같이 8 노드 혹은 16 노드로 구성되고, 각 노드의 ID 는 gray 코드를 따라 부여한다..

2.2 Hypercube 네트워크를 사용한 PC 클러스터의 구현



(a) 8 노드 (b) 16 노드

(그림 1) Hypercube 네트워크 연결 예

본 논문에서는 3 개의 동일한 LAN 카드를 장착한 8 대의 PC 를 사용하여 (그림 1)의 (a)와 같이 8 노드의 연결 형태를 적용하였고, 이 때, 각 노드의 ID 로는 gray 코드를 십진수로 변환한 값을 부여하였다. [표 1]에 각 노드(PC)의 사양을 요약하였다.

<표 1> Hypercube 기반 PC 클러스터 노드의 사양

항 목	사 양
CPU	Intel Pentium IV 3.0GHz
LAN Card	Gigabit PCI Adapter(Netgear) 10/100/1000 Mpbs Gigabit Ethernet GA311
운영체제	Linux 2.6.9-1.667

■ IP 설정

각 노드는 직접 연결된 3 대의 노드의 ID 를 사용하여 각각 다른 3 가지의 고유 IP 를 갖도록 설정하였다. 예를 들어, 노드 0 은 노드 1, 노드 2, 노드 4 와 직접 연결되어 있기 때문에, 노드 1 에서는 노드 0 의 IP 주소가 192.168.100.101, 노드 2 에서는 노드 0 의 IP 주소가 192.168.100.102, 그리고 노드 4 에서는 노드 0 의 IP 주소가 192.168.100.104 로 인식되도록 설정하였다.

거리가 2 이상 떨어진 노드의 경우에는 병렬 프로그램을 실행하기 위하여 도착 노드가 출발 노드의 주소를 인식해야 하기 때문에 도착 노드에서는 출발 노드의 3 개의 주소 중에 해당 경로의 주소를 사용하였다. 예를 들어, 노드 0 에서 노드 3 으로의 경로는 노드 1 을 통하도록 설정하였기 때문에 노드 3 에서의 노드 0 의 주소는 192.168.100.101 이다.

거리가 3 인 노드 7 에서의 노드 0 의 주소는 노드 1 과 3 을 중간 경로 노드로 지정하였기 때문에 노드 0 에 직접 연결된 노드 1 을 통한 주소인 192.168.100.

101 이 된다. (그림 2)에 노드 0 에서 설정한 다른 노드들의 설정 내용의 예를 보였다.

```
# cat /etc/hosts
127.0.0.1               localhost
192.168.0.100           node00
192.168.101.100         node01
192.168.102.100         node02
192.168.103.102         node03
192.168.104.100         node04
192.168.105.101         node05
192.168.106.104         node06
192.168.107.106         node07
```

(그림 2) 노드 0 에서 다른 노드의 설정 예

■ Gateway(라우팅 테이블) 설정

Hypercube 네트워크에서 노드 간의 거리가 2 이상의 경우에 중간의 노드들이 데이터를 전달할 수 있는 게이트웨이 역할을 하도록 라우팅 테이블을 설정해야 한다[10]. 예를 들어, 노드 0 에서 거리가 2 이상인 경로는 (그림 3)과 같이 설정하였다. 이 경로는 데이터를 보낼 때이며 받을 때는 통신의 효율을 위하여 다른 경로를 설정할 수 있다.

0 →1 →3, 0 →4 →5, 0 →2 →6, 0 →1 →5 →7

(그림 3) 노드 0 에서 라우팅 테이블의 설정 예

3. 성능 분석

3.1 성능분석 방법

Hypercube 네트워크 기반의 PC 클러스터의 성능을 분석하기 위하여 동일한 PC 와 Gigabit 스위치를 기반으로 하는 PC 클러스터와 성능을 비교하였다. 성능분석 프로그램은 ping[3,4], netperf[4,5]와 표준 병렬 프로그램 라이브러리인 MPICH[6]를 사용하는 병렬프로그램을 사용하였다. <표 2>에 Gigabit 스위치 기반 클러스터에 사용된 PC 의 사양을 요약하였다.

<표 2> Gigabit 스위치 PC 클러스터 노드의 사양

항 목	사 양
CPU	Intel Pentium IV 3.0GHz
LAN Card	Intel Pro/10000 MT PWLA8390MT-LP
스위치	Gigabit Ethernet Switch Series JGS500
운영체제	Linux 2.6.9-1.667

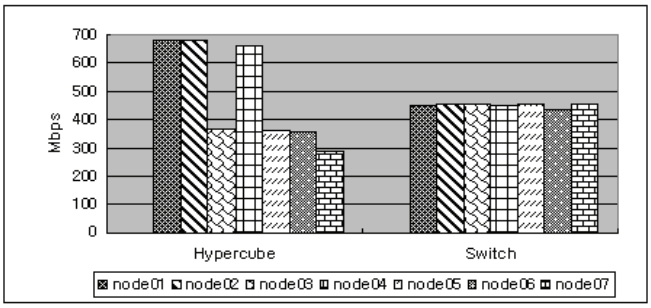
3.2 성능분석 결과

■ Point-to-point 간의 단순 전송 속도 성능

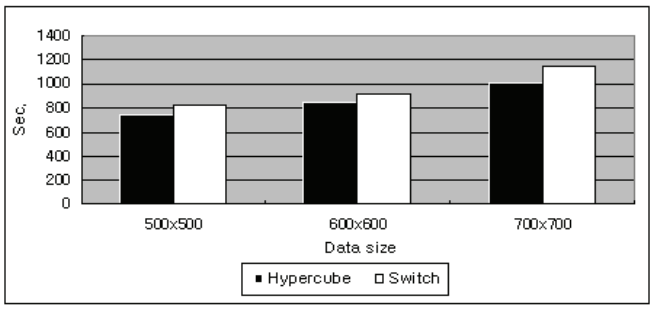
Ping, netperf, MPI_Send/Recv 에 의한 노드간 단순 전송 성능 특성은 (그림 3)과 같이 전송 거리에 따라 좌우되고, 스위치 클러스터의 경우는 거리가 일정하기 때문에 모든 노드간의 성능이 균일하다.

■ 병렬 프로그램 Cannon 의 실행 성능

병렬 프로그래밍이 적용된 Canon 의 경우 (그림 4)와 같이 hypercube 클러스터가 우수하다.



(그림 3) 노드 0 에서 ping 에 의한 성능 측정 결과



(그림 4) 병렬프로그램 Canon 의 실행 성능

4. 결 론

일반적으로 PC 클러스터의 네트워크를 구현하기 위해서 별도의 네트워크 장비를 사용하는데 이들 장비로는 Myrinet, Gigabit 스위치 등이 많이 사용되고 있다. 이 네트워크 장비들은 비교적 쉽게 구축할 수 있고 효율적인 성능을 보이고 있다. 하지만 이러한 네트워크 장비를 통하여 통신을 할 경우에는 장비에서의 최소한의 제어 과정이 필요하기 때문에 PC 간에 직접 연결한 경우보다는 성능이 낮아진다. 또한 장비에 대한 의존도가 높아지게 되어 장비에 문제가 발생할 경우에 클러스터의 안정성이 낮아지게 된다.

이 논문은 별도의 네트워크장비를 사용하지 않으면서도 보다 나은 성능을 얻을 수 있는 hypercube 네트워크에 의한 PC 클러스터의 실용 가치를 재확인했다는 점에 서 의의가 있다고 본다.

참고문헌

[1] 김재열, 강동재, 김수영, 차규일, 리눅스 Bonding 드라이버의 성능분석, 한국통신학회, KNOM Review Vol.8, No.1, pp. 57-75, Aug. 2005.

[2] Hewlett-Packard Company, Information Networks Division : A Network Performance Benchmark, 1995.

[3] Pacheco P., Parallel Programming with MPI, San Francisco, CA: Morgan Kaufmann, 1997.

[4] Fox, G. C., Johnson, M. A., Lyzenga, G. A., Otto, S. W., Salmon, J. K. and Walker, D. W., Solving Problems On Concurrent Processors Volume I: General Techniques and Regular Problem, Engelwood Cliffs: Prentice-Hall 1998.

[5] Kim, Y., Towards a Fair Comparison of Parallel Machines, Journal of KISS(A): Computer Systems and Theory (정보 과학회논문지(A)), Vol. 26, No. 1, pp. 43-52, Jan. 1999.

[6] Cannon, L. E., A cellular computer to implement the Kalman Filter Algorithm. Ph.D. thesis, Montana State University, 1969.