제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

클러스터 시스템에 대한 DRACO 구조의 적용성 연구

서효중
가톨릭대학교 컴퓨터정보공학부
e-mail : hjsuh@catholic.ac.kr

A Study on DRACO Architecture Applied to the Cluster Systems

Hyo-Joong Suh
School of Computer Science and Information Engineering
The Catholic University of Korea

요 약

DRACO 구조는 이중 링 연결형태에 대비하여 노드간 접근 경로를 단축할 수 있는 구조로, CC-NUMA 시스템에 대하여 그 적용성과 프로토콜이 시험되어 그 효율성이 입증되었다. 본 논문은 이러한 DRACO 구조가 보다 많은 프로세서를 수용할 수 있는 클러스터 시스템에 대하여 동일한 경로 단축과 대역폭 확장을 적용할 수 있음에 기반하여, 이중 연결 링크를 가지고 있는 점대 점 연결 형태의 클러스터 시스템에 대한 적용성을 연구하였다. 노드 내의 네트워크 인터페이스 및 소프트웨어만을 이용한 DRACO 구조 적용은 많은 수의 노드를 필요로 하는 시스템에 적합하고, 하드웨어를 이용한 연결 경로를 이용할 경우 상대적으로 적은 수의 노드를 채용하고 고속의 링크 성능을 필요로 하는 시스템에 적합할 것이다.

1. 서론

클러스터 시스템은 높은 수준의 병렬성과 계산능력을 요구하는 응용에 적합한 시스템으로 단위 시스템을 기가비트 이더넷, Myrinet [1] 등의 상호연결망을 이용하여 여러 대의 컴퓨터를 연결함으로써 구현되어 왔다. 특히 MPI(Message Passing Interface)[2], VIA(Virtual Interface Architecture)[3]등 표준화된 인터페이스 방법을 이용한 PC 클러스터 시스템은 높은 가격대 성능비와 표준화에 따른 높은 활용성을 나타냄으로써 학계 및 산업계에서 널리 연구되고 있다[4]. 상호연결망의 성능은 클러스터 시스템의 성능에 직접 연관되는 요소로써, 높은 대역폭과 적은 지연을 나타내는 연결망을 사용할 때, 클러스터 시스템 전체의 성능 또한 보다 높아지게 된다.

DRACO 구조는 이중 연결망을 낮은 지연을 갖도록 배치한 형태로써 CC-NUMA 시스템에 대한 적용성이 연구된 바 있다[5]. 이 구조는 이중 연결 링 형태에 대비하여 적은 지연과 효율적인 연결망을 사용을 나타내며, 이러한 연결 형태는 동일한 특성을 요구하는 다양한 응용에 적용될 수 있다. 본 연구는 DRACO 연결 구조에 대한 적용성을 클러스터 시스템으로 확장함으로써 클러스터 시스템에서 필요로 하는 일대일 통신 형태의 응용에 대한 적용성을 연구하였다.

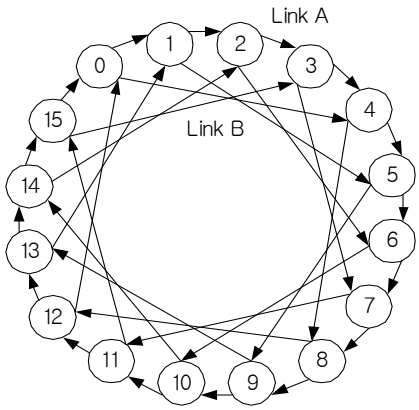
2. DRACO 구조의 연결 경로 길이

다음 그림 1 은 DRACO 연결 구조의 한 형태이다. DRACO 구조는 방송 트랜잭션과 일대일 트랜잭션에 대하여 각각 지연을 축소할 수 있는 형태로 제시되었으며, 건너뜀 경로로 인한 경로 단축을 얻을 수 있다.

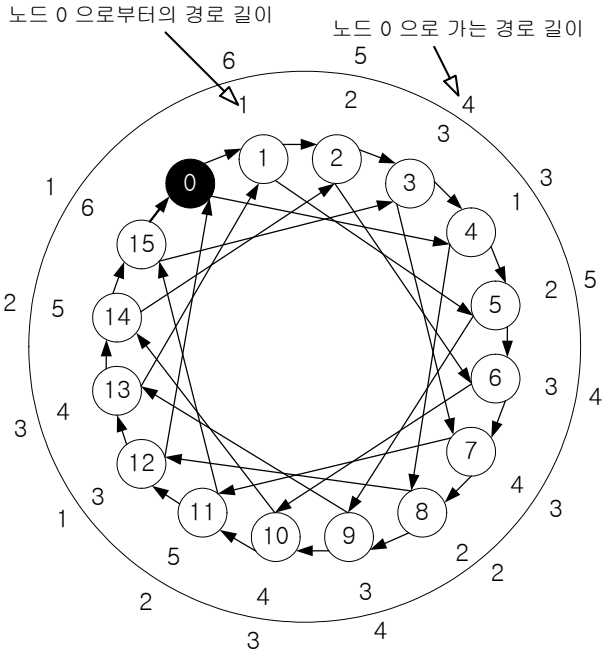
건너뜀 연결로 인한 경로 단축 효과 중 일대일 통신을 통한 양방향 연결 단계의 경로 길이는, 다음 그림 2 와 같이 나타나게 된다.

즉 DRACO 구조 연결형태를 이용할 경우, 각 노드간 양방향 통신을 할 때, 왕복 경로 길이의 합은 각 노드간에 비교적 균일하게 형태로 나타나게 되며, 분산 프로그램에서 각 노드간에 비교적 균등한 통신 부

하가 나타날 경우 상당히 효율적인 통신 비용을 나타내게 됨을 알 수 있다.



(그림 1) DRACO 구조의 연결 예, 16 노드, 4 건너뜀



(그림 2) DRACO 구조의 경로 길이

3. 왕복 경로 길이 비용

클러스터 시스템이 ***N*** 개의 노드로 구성되고, DRACO 구조의 건너뜀 수가 ***S*** 일때, 모든 노드간 일대일 조합에 대한 모든 왕복 연결 경로 길이의 총 합은 다음과 같이 계산된다.

$$2N\sum_{i=0}^{(N/S-1)}\sum_{j=0}^{(S-1)}(i+j)=2N^2\frac{\frac{N}{S}+S-2}{2}$$

또한 각 노드 조합에 대한 왕복 경로 길이는 ***N/S*** 와 ***S+N/S-1*** 중의 한 가지로 균일하게 나타나게 되므로, 노드간 경로 길이의 단축과 균일하게 나타나는 왕복 경로 특성은 클러스터 시스템 구성시 고성능의 연결 구조로 활용될 수 있다.

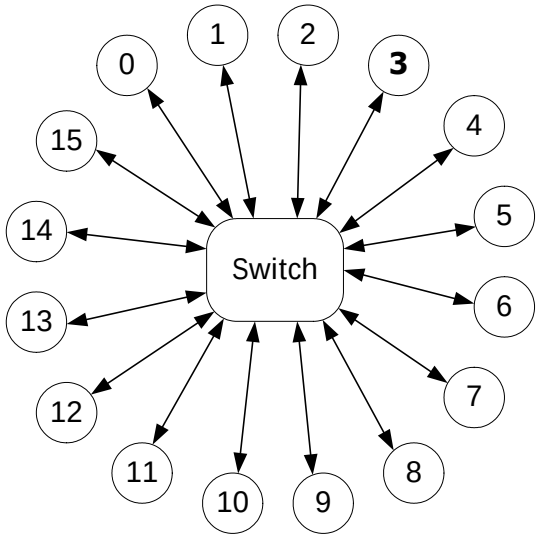
모든 조합에 대하여 작은 왕복 경로 길이인 ***N/S*** 가 나타나는 횟수는 총 ***N(N/S-1)*** 회 이므로, 건너뜀 수 ***S*** 가 적어질수록 짧은 경로는 보다 많이 나타나게 되나, 두 가지로 나타나는 왕복 경로 길이간의 차이는 보다 적어지게 된다.

4. 클러스터 시스템

클러스터 시스템에 DRACO 구조를 적용할 경우, 링크 연결은 SCI(Scalable Coherent Interface)[6], 기가비트 이더넷, Myrinet 등 제한되지 않으며, 어떠한 링크를 사용하여도 쉽게 구현될 수 있다.

Beowulf[7] 와 같은 PC 클러스터 시스템으로 구현할 경우, 가장 용이한 방법은 기가비트 이더넷 인터페이스를 각 시스템에 두 장씩 장착하여 직접 PC 간 DRACO 구조와 같은 형태로 연결하여 구현할 수 있다. 1000BASE-T 이더넷 인터페이스의 경우 단위 링크에 TX 와 RX 가 각각 존재하므로, 각 pair 에 대하여 별개의 경로로 구성되어야 하며, 각 PC 에서 동작하는 운영체제는 이더넷 인터페이스에 대하여 DRACO 구조에서 제시하는 경로에 따라 라우팅을 제공할 수 있어야 하므로, 인터페이스를 통하여 들어오는 패킷과 나가는 패킷을 직접 제어할 수 있는 디바이스 드라이버의 제작이 필요하다.

기가비트 이더넷을 이용한 PC 클러스터 시스템은 그림 3 과 같이 고속 스위치를 이용하여 star 형태로 구성된다.



(그림 3) 스위치로 연결된 클러스터 시스템

Star 형태로 만들어진 클러스터 시스템의 경우, 각 노드간의 조합에서 명시적으로 나타나는 경로 길이는 2 단계로 균일하다. 그러나 스위치 내부 에서 링크간 연결을 위한 네트워크망의 구성은 필수적이며, 결과적으로 스위치의 대역폭 및 구성 형태에 따라서 전송 효율이 결정된다. 기가비트 이더넷 인터페이스를 이용하여 클러스터 시스템이 구성될 경우, 스위치는 ***N*** 노드에서 최대 ***N*** 기가 비트의 처리를 필요로 하며, 클러스터 시스템이 많은 노드를 수용하게 될 경우, 단위

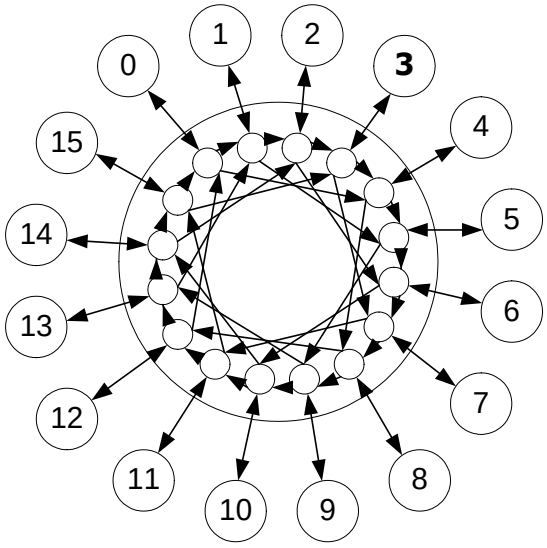
스위치로 이와 같은 대역폭을 해결할 수 없으므로 계층형 구성을 필요로 하게 된다. 결국 스위치의 성능과 확장성에 의하여 시스템의 확장성이 제한된다.

반면 DRACO 구조의 경우 노드의 추가에 따른 연결 재구성이 필요할 수 있으나, 스위치와 같은 부가적인 하드웨어를 필요로 하지 않으며, 스위치 내부의 구현 형태와 대역폭 제한에 따른 성능 저하가 발생하지 않는다. 또한 DRACO 구조는 두 배의 링크를 가지게 되므로, 늘어난 대역폭으로 인한 이득이 나타난다.

5. 성능 제한 요소 및 확장성

스위치를 이용한 클러스터 시스템에 대비하여 DRACO 구조가 가지는 성능 제한 요소와 문제점은 두 가지가 있다. 첫째. DRACO 구조에서는 링크를 통하여 패킷을 전달하는 형태이므로, 연결된 각 시스템이 인터페이스를 통하여 받은 패킷에 대한 처리와 라우팅을 결정하는 부가적인 처리를 필요로 하며, 둘째로, 결국 패킷이 노드간의 연결을 통하여 전달되므로, 링크에 대해 보다 많은 패킷 전달이 발생한다.

이 두 가지 문제점은 DRACO 구조 적용시 중요한 성능 저하 요소로 대두될 수 있으므로, 이와 같은 문제점을 해결하기 위하여 DRACO 구조 형태의 연결을 STAR 형태에서 스위치 부분에 대응하여 구현할 수 있다. 다음 그림 4 는 DRACO 연결형을 중앙 집중 형태로 변형한 것이다.



(그림 4) DRACO 구조의 별도 인터페이스 구현형태

DRACO 연결 링크 구조에 대하여 그림 4 와 같이 별도의 하드웨어를 이용하여 구현할 경우, 일반적인 스위치를 이용한 구조와 동일한 확장성 제한 문제가 대두될 수 있다. 그러나 이 경우, 패킷의 전달에 대한 처리를 고속의 하드웨어를 이용할 수 있으므로, 보다 빠른 처리를 기대할 수 있으며, 각 노드에서 별다른 드라이버 제작 등을 필요로 하지 않고 표준 드라이버를 이용한 응용 수준의 시스템 구성이 가능하다.

6. DRACO 구조 적용에 대한 고찰

두 개의 네트워크 인터페이스를 통한 연결 형태와 별도의 하드웨어 구성을 이용한 두 가지 DRACO 구조 적용은 각각 장단점을 나타낸다. 앞서 논의된 바와 같이 첫번째 구조는 별도의 하드웨어를 필요로 하지 않는 반면 확장성에 있어서 star 형 구조에 대비하여 장점을 나타내나 라우팅 처리부가 각 노드에 포함되므로 부가적인 드라이버 처리 등을 필요로 하게 되고 이로 인한 성능 저하가 발생할 수 있으며, 두 번째 유형은 별도의 라우팅 하드웨어를 사용하게 되므로, 고속 처리가 가능하고 각 노드에서 드라이버 수준의 처리가 필요로 하지 않으나, 스위치를 이용한 경우와 마찬가지로 확장성에 대한 제한이 발생하게 된다.

이 두 가지 유형은 각각 규모에 대한 적용성 및 혼합한 형태로 활용할 수 있을 것으로 기대한다. 즉 적은 규모의 클러스터 시스템에서는 하드웨어 구현과 라우팅 드라이버 구현을 필요로 하지 않으면서 고속처리가 가능하도록 적용하는 것이 적절할 것으로 생각하며, 부가적인 하드웨어 없이 소프트웨어 수준에서의 라우팅 처리로 구현한 형태의 경우 많은 노드에서 나타나는 DRACO 구조의 경로단축 성질과 유연한 확장성이 장점으로 나타날 것으로 생각한다.

DRACO 구조를 이용하여 클러스터 시스템을 구현할 경우 또다른 문제로 대두될 수 있는 것은 DRACO 와 같은 연결형을 이용할 경우 특정 노드에 문제가 생겼을 때, 정적 연결 경로만을 적용할 경우 특정 노드간에 통신이 불가능해지는 것이다. 이 문제는 특히 내결함성적 측면으로서의 클러스터 시스템의 장점을 가지지 못하게 되므로 보다 깊게 고려되어야 하며, 각 드라이버에서 연결에 대한 감시를 통하여 문제가 발생할 때, 동적으로 라우팅 경로를 재구성하여야만 하므로 부가적인 관리 소프트웨어가 구현되어야만 한다.

7. 결론

본 연구는 DRACO 구조에서 제안된 연결을 클러스터 시스템 형태로 적용할 때 가능한 연결형에 대하여 제시하고, 각 연결형의 장점과 단점에 대하여 제시하였다. DRACO 구조를 이용하여 클러스터 시스템을 구현할 때, 부가적인 하드웨어를 사용하지 않고 소프트웨어 드라이버 수준의 처리를 이용한 구성할 경우 많은 수의 노드를 이용한 시스템 구현이 용이하고 단축경로의 장점을 나타내나, 소프트웨어 드라이버의 수정 및 내결함성을 보완하기 위한 동적 경로 변경 관리를 필요로 한다. 별도의 하드웨어 내에서 DRACO 연결 구조를 구현하여 노드를 연결할 경우, 소프트웨어 처리를 필요로 하지 않고 고속으로 처리할 수 있는 반면, 스위치를 이용한 클러스터 시스템과 동일한 확장성의 문제를 나타내게 된다. 차후 연구로써 이와 같은 두 가지 형태의 구성이 star 형태와 같은 클러스터 시스템에 대비하여 나타나는 성능 비교가 진행중이며, 첫 단계로 각 시스템의 성능을 시뮬레이션을 통한 성능평가가 진행될 것이고, 다음 단계로 시뮬레이션의 결과에 따라 링크의 성능과 노드 수에 따른 적절한 시스템 유형을 선택하여 구현할 예정이다.

참고문헌

[1] http://www.myri.com/myrinet
[2] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, UT-CS-94-230, 1994.
[3] Intel, Compaq, and Microsoft Corporations, Virtual Interface Architecture specification Version 1.0, December 1997, http://www.viarch.org/
[4] T. Anderson and D. Culler and D. Patterson, “A Case for NOW (Networks of Workstations)", IEEE Micro, Vol.15, no.1, pp.54-64, Feb. 1995.
[5] 서효중, “이중 연결 CC-NUMA 시스템의 효율적인 상호 연결망 구성 기법”, 정보처리학회논문지, 제 11-A 권 제 1 호, 2004.
[6] IEEE Computer Society, IEEE Standard for Scalable Coherent Interface(SCI), Institute of Electrical and Electronics Engineers, August 1993.
[7] http://www.beowulf.org/

서 효 중
e-mail : hjsuh@catholic.ac.kr
1991 년 서울대학교 이학사
1994 년 서울대학교 공학석사(컴퓨터공학)
2000 년 서울대학교 공학박사(컴퓨터공학)
2002 년 지씨티 리서치 선임연구원
2003 년~현재 서울대학교 컴퓨터연구소 객원연구원
2003 년~현재 가톨릭대학교 컴퓨터정보공학부 전임강사
관심분야 : 컴퓨터 구조, 병렬처리 시스템, 내장형시스템, 클러스터 시스템