병렬 컴퓨터 시스템에서의 Multi-drop 방식을 사용한 하드웨어 장벽 동기화

(A Hardware Barrier Synchronization using Multi-drop Scheme in Parallel Computer Systems)

이 준 범 † 김 성 천 ††

(June-Bum Lee) (Sungchun Kim)

요 약 대규모의 업무처리나 복잡한 연산을 요구하는 응용 분야에서는 프로그램의 병렬화를 이용하는 병렬 컴퓨터 시스템이 요구되고 있다. 이 병렬 컴퓨터 시스템의 핵심 작업 중 하나가 동기화이다. 동기화 작업 중 가장 대표적인 방법이 장벽 동기화인데 이 방법은 동기화에 참여하는 모든 프로세서들이 모두 장벽에 도달할 때까지 다음 작업을 진행시킬 수 없게 만드는 것이다.

장벽 동기화는 소프트웨어에 의한 방법, 하드웨어에 의한 방법, 그리고 그 두 가지가 결합된 방법 등이 있는데 이 중에서 하드웨어에 의한 방법이 가장 빠르고 start-up overhead가 적다는 장점으로 인하여 널리 쓰이는 추세이다.

본 논문에서는 하드웨어에 의 한 방법 중에서 간단한 오류를 해결할 수 있고 보다 빠른 동기화를 가능하게 하는 새로운 스위치 모듈을 제안한다. 새로운 스위치 모듈과 더불어 제안하는 장벽 동기화는 기존에 제안되었던 방법에 비해서 스위치에 모든 것을 의존하는 방식이 아니라 프로세서에 의해 동작되는 부분이 많기 때문에 적은 하드웨어 비용을 들이고서 효과적인 장벽 동기화를 실행한다.

새로 제안하는 장벽 동기화는 어떠한 망의 구조에서도 구현될 수 있게 설계되었다. 본 논문에서는 MIN에서의 장벽 동기화에 대한 비교만을 성능 평가하였는데 24.6% ~ 24.8%의 평균 지연 시간의 감소를 보였다. 하지만 임의의 망인 비정규적인 망에서 보다 나은 성능 향상을 보일 것을 기대한다.

Abstract The parallel computer system that uses parallel program on the application such as a large scale business or complex operation is required. One of crucial operation of parallel computer system is synchronization. A representative method of synchronization is barrier synchronization. A barrier forces all process to wait until all the process reach the barrier and then releases all of the processes.

There are software schemes, hardware scheme, or combinations of these mechanism to achieve barrier synchronization which tends to use hardware scheme. Besides, barrier synchronization lets parallel computer system fast because it has fewer start-up overhead.

In this paper, we propose a new switch module that can implement fast and fault-tolerant barrier synchronization in hardware scheme. A proposed barrier synchronization is operated not in full-switch-driven method but in processor-driven method. An effective barrier synchronization is executed with inexpensive hardware supports.

Therefore, a new proposed hardware barrier synchronization is designed that it is operated in arbitrary network topology. In this paper, we only show comparison of barrier synchronization on Multistage Interconnection Network. This research results in 24.6-24.8% reduced average delay. Through this result, we can expect lower average delay in irregular network.

† 비 회 원 : 서강대학교 컴퓨터학과
bluesky@arglab1.sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
ksc@arglab1.sogang.ac.kr
논문접수 : 1999년 2월 24일
심사완료 : 2000년 3월 29일

1. 서 론

최근 대규모의 업무 처리나 복잡한 연산을 요구하는 응용분야에서는 프로그램 수행의 병렬화를 이용하는 병렬 컴퓨터 시스템 (parallel computer system)이 요구되고 있다. 이 시스템은 이전에는 한 대형 컴퓨터에 수

백 혹은 수천 개의 독립적인 프로세서(processor)와 메모리를 갖는 노드들을 고속의 상호연결 망(interconnection network)방식을 이용해 동작하여 내부적으로 병렬화 작업을 할 수 있는 것이었지만 컴퓨터 기술의 발달로 인하여 워크스테이션이나 개인용 컴퓨터 사이에서도 병렬화 작업을 할 수 있게 되었다.

동기화 작업 중 가장 대표적인 방법은 장벽(barrier) 동기화 방법이다. 이 방법은 동기화에 참여하는 모든 프로세서들이 모두 장벽에 도달할 때까지 다음 작업을 진행시킬 수 없게 만드는 방법으로 장벽에 도달하는 프로세서들의 개수를 센다든지 하는 방법 등으로 모두 동기화가 이루어졌는지를 확인하는 방법이다[1]. 이 장벽 동기화는 메시지 전달 시스템 (message-passing system), 특히 공유 메모리 병렬 시스템 (shared-memory parallel system)들에게 가장 핵심적인 집합적 통신 작업(collective communication operation)이다[2].

장벽 동기화는 소프트웨어, 하드웨어, 그리고 그 둘이 결합된 방법등 세 가지 방법으로 나뉘어지는데 IBM SP2, Intel Paragon등 대부분의 워크스테이션들의 네트워크에서는 전체적으로 소프트웨어에 근거한 방법을 채택하고 있다. 이러한 방법은 새로운 시스템에 쉽게 장착 가능하지만, 느리다는 점과 심각한 start-up 오버헤드가 발생한다는 단점이 발생하게 된다[2].

이러한 단점을 보완하려는 노력 중 하나가 하드웨어로 처리하는 방법이다. 이 방법은 소프트웨어에 의한 방법에 비하여 빠르다는 장점이 있지만 수백, 수천의 노드들을 가지고 있는 큰 시스템에서는 하드웨어 비용이 증가하는 단점을 가지게 된다. 하지만 최근 하드웨어 비용이 줄어드는 경향과 빠른 처리 속도를 추구하는 업무가 많아지는 것에 비추어 볼 때 하드웨어를 이용하는 방법이 더욱 유용하게 사용될 것이다.

하드웨어에 의한 방법은 초창기 AND 게이트를 사용하는 방법[3]에서부터 2개의 NAND 게이트를 사용하는 방법[4], 여러 개의 레지스터(register)와 AND 게이트, 그리고 zero detect logic등을 사용하는 방법[5], RAM(Random Access Memory)을 사용하는 방법[2] 등 많은 방법이 발표되었다. 하지만 이런 방법들은 소프트웨어에 의한 방법보다는 빠르지만 전체 시스템 중 모든 프로세서가 동기화에 참여하지 않는 경우에 적용할 수 없다는 점, 수백, 수천 개의 프로세서가 동기화에 참여하는 경우에 많은 레지스터가 필요하다는 점, 그리고 스위치에서 너무 많은 시간이 소요된다는 점등이 단점으로 나타났다.

본 논문에서는 위에서 제기된 문제점들을 해결하고 보다 빠른 동기화를 위하여 RAM을 사용하지 않고 확인 신호를 빠르게 받을 수 있는 멀티 드롭 방식을 사용한 새로운 스위치 모듈과 그 스위치 모듈에서 사용되어지는 동기화 유닛 모듈을 제안하고 오류가 발생하였을 경우에도 보완할 수 있는 방법을 포함한 새로운 동기화 방법을 제안하고자 한다. 본 논문에서 새로 제안하는 방법은 기존에 연구되었던 방법에 비해서 프로세서들에 의하여 동작하는 부분(processor-driven)이 많고 스위치에 의하여 동작하는 부분(switch-driven)은 상대적으로 적다. 그렇기 때문에 이것은 다단계 상호연결망(Multi-stage Interconnection Networks :MIN)등 정규망(regular networks)에서 뿐 아니라 멀티 드롭 경로에 근거한 다중 전송 웜을 사용한 멀티캐스팅(multi-casting)방법[6]을 사용하여 비정규망(irregular networks)에서도 사용할 수 있도록 제안한다.

본 논문에서 제안하는 구조에서는 오류가 발생하였을 때에 대해서는 자세히 설명하지 않았다. 하지만 프로세서 상호간의 통신 중에 생기는 메시지를 잃어버리는 것 같은 단순한 오류가 발생하였을 때 대처할 수 있는 방법은 스위치나 루트 노드, 혹은 루트가 아닌 노드들의 유한 상태 기계 다이어그램(finite state machine diagram)을 통하여 설명하였다. 또한 이 유한 상태 기계 다이어그램에서 루트, 혹은 루트 이외의 프로세서들이나 스위치의 동작 기능을 설명하여 전체적인 작동 상황을 알 수 있게 하였다.

본 논문의 구성은 다음과 같다. 1장 서론에 이어 2장에서는 장벽 동기화가 무엇인지 그리고 어떤 방법으로 이루어지는 지 살펴보고 기존에 연구되었던 하드웨어 장벽 동기화에 대해서 살펴본다. 또 3장에서는 본 논문에서 제안하는 데 필요한 구조, 즉 패킷과 스위치 구조 그리고 장벽 유닛에 대해서 살펴보고 오류가 발생하였을 때 사용되는 프로토콜(protocol)에 대해서 살펴본다. 4장에서는 기존의 MIN에서 사용되었던 하드웨어 장벽 동기화와 본 논문에서 제안하는 장벽 동기화를 시뮬레이션을 통해 비교한다. 그리고 마지막으로 5장에서 결론을 맺는다.

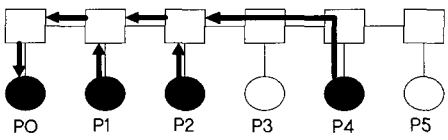
2. 하드웨어 장벽 동기화

하드웨어 장벽 동기화는 Harry Jordan[3]에 의해서 처음 제안되었다. 그후로 이것은 MIMD(multiple instruction stream, multiple data stream) 병렬 프로세스(process)들의 중요한 메커니즘이 되어왔다. 그리하여 AND 트리 하드웨어를 사용하는 Burroughs FMP, delayed firing 등 더욱 복잡한 하드웨어를 사용한 퍼지

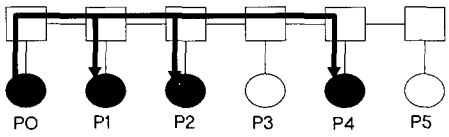
장벽 등으로 발전되어 왔고 CM-5나 Cray T3D 등에서도 FMP와 비슷한 장벽 동기화를 사용하고 있다[7].

그러나 AND 트리를 사용하는 방법은 수백, 수천 개의 노드들로 이루어지는 동기화의 경우, 게다가 연속적으로 동기화가 이루어져야 하는 경우에 느려진다는 단점이 나타나게 되었다. 그리하여 노드들 중 일부만 참여할 수 있고 더 빠른 동기화를 위한 방법이 제안되었다. <그림 1>은 선형 배열에서의 장벽 동기화[8]를 나타낸 그림이다.

<그림 1>에서 나타난 장벽 동기화는 개더(gather) 과정과 브로드캐스트(broadcast) 과정, 두 단계로 이루어지게 되는데 각 단계는 개더 웜들과 브로드캐스트 웜들의 명령을 사용하여 초기에 제안된 장벽 동기화의 단점을 극복하면서 빠른 장벽 동기화를 구현한 것이다. 초기에 나타났던 단점이란 망에 연결된 프로세서들 중에 일부만 동기화에 참여하는 경우, 특히 연속되는 동기화 과정이 있어서 동기화에 참여하는 프로세서가 달라지는 경우에는 사용하지 못하기 때문에 모든 프로세서가 장벽 동기화에 참여하지 않으면 사용할 수 없다는 점이다. <그림 1>에서는 이러한 단점을 보완한 동기화를 나타내고 있다.



(a) 보고과정



(b) 브로드캐스트 과정

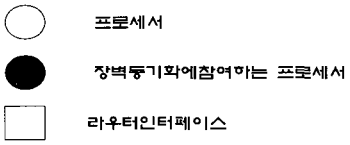


그림 1 선형 배열에서 장벽의 두 단계 구현

동작하는 단계를 자세히 살펴보면 <그림 1>의 a)에서 가장 오른쪽의 프로세서인 P4에서 장벽에 도착한 후에 다중 목적지의 초기화를 시작하고 이 웜의 헤더에는 장벽에 참여하는 프로세서들이 목적지의 순서대로 차례로 구성이 되어 있어서 이 개더 웜이 차례로 (P2, P1, P0의 순서로) 방문하면서 그 순간의 프로세서가 장벽에 도달할 때까지 기다리다가 도착하는 즉시 다음 목적지로 가는 방법을 이용하여 최종 목적지인 P0에 도착하면 개더 웜, 즉 보고 과정이 끝나게 된다.

위의 보고 과정이 아무런 오류 없이 끝나게 되면 최종 목적지인 P0에서는 장벽에 참여하였던 모든 프로세서들에게 각각 기다리고 있던 다음 작업을 계속하여도 좋다는 일깨움(wake-up) 과정이 필요하게 된다. 이 과정은 <그림 1>의 b)에서 나타난 것처럼 개더 웜의 역으로 순서가 진행이 되어 P1, P2, P4의 순서로 다음 작업을 진행해도 좋다는 신호를 주게되는 일깨움을 하게 된다. 이 과정이 브로드캐스트 과정이다. 이 구조의 단점은 장벽에 모든 프로세서들이 도착할 때까지 많은 시간이 걸린다는 점이다. 즉 보고하는 과정에 있어서 가장 오른쪽에 있는 프로세서가 제일 먼저 장벽에 도착하는 경우, 가장 늦게 다음 작업을 진행시켜도 좋다는 신호를 받기 때문에 전체 장벽 동기화 과정의 시간이 오래 걸린다. 그리고 프로세서가 선형 배열, 또는 메쉬(mesh) 등 규칙적으로 배열되어 있는 망에서만 유효하다는 단점도 존재한다.

위의 연구에서 본 장벽 동기화는 오류가 발생할 때에는 전혀 고려하지 않았지만 R. Sivaram[2]가 연구한 장벽 동기화는 위의 예와 같이 두 단계가 아니라 세 단계의 작업을 보여주면서 오류가 발생할 때 오류 허용(fault tolerant)이 가능하게 해주고 있다. 물론 위의 예보다 더 느려진다는 단점이 있지만 장벽 동기화의 방법의 차이에 의해 시간을 단축하여 선형 배열에서의 동기화[8] 연구보다는 빠른데다가 연속적인 장벽 동기화를 가능하게 해주고 있음을 알 수 있다.

Sivaram[2]의 연구에서는 기존의 하드웨어 장벽 동기화와는 달리 기존의 스위치에 약간의 추가 하드웨어만 추가함으로서 값싼 장비만을 이용하여 빠른 장벽 동기화의 구현을 제안하고 있다.

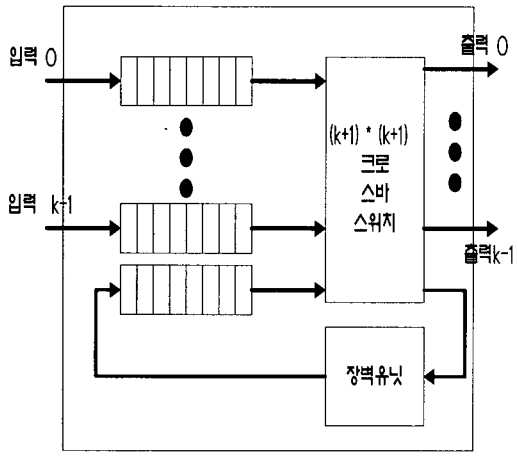
각 스위치에서의 단계는 세 단계로 이루어진다. 첫 번째 단계는 결합(combine) 단계로서 장벽에 도달한 각 프로세서들은 멀티캐스트 트리의 루트로서 동작하는 프로세서(이하 루트 프로세서라고 한다)로 장벽에 도달하였다는 메시지인 결합 패킷(packet)을 보낸다. 두 번째 단계는 루트 프로세서가 마지막으로 장벽에 도달하는 프로세서로부터 결합 패킷을 받은 즉시 (대부분의 경우 루트에 해당하는 프로세서이다) 장벽 동기화에 참여한 모든 프로세서들(이하 참여 프로세서라고 한다)에게 결합 패킷을 받은 역 순서로 멀티캐스트 패킷을 보낸다.

세 번째 단계는 멀티캐스트 패킷을 받은 프로세서들은 받은 즉시 다음 작업을 수행하는 동시에 아무런 오류 없이 메시지를 받았다는 확인(acknowledgement)패킷을 루트 프로세서에게 보내는 것이다. 루트 프로세서가 오류 없이 참여 프로세서들로부터 확인 패킷을 받으면 루트 프로세서는 다음 작업을 진행하게 된다.

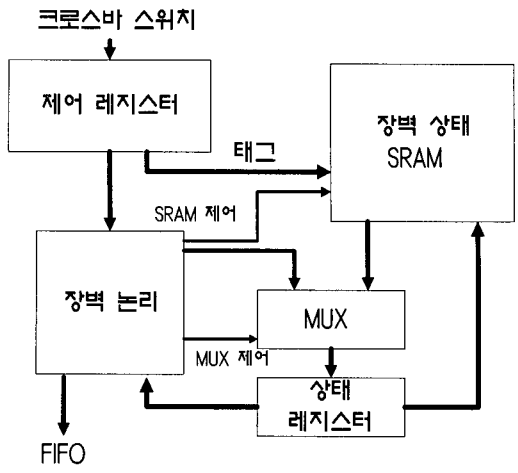
<그림 2>는 R. Sivaram의 연구[2]에서 제안된 장벽 동기화의 스위치 모듈과 장벽 유닛의 구조를 보여주고 있다. <그림 2>의 (a)의 그림은 기존의 (k x k) 크로스바 스위치(crossbar switch)에 한 개의 입력과 한 개의 출력을 더 추가하여 장벽 유닛으로부터의 입력과 출력을 담당하고 있는 스위치 모듈을 보여주고 있다.

<그림 2.>의 (b)의 그림은 네 단계의 사이클로 동작하고 있는 장벽 유닛을 보여주고 있다. 첫 번째 사이클은 크로스바 스위치에서 들어온 패킷은 제어(control) 레지스터로 들어가는 데 소요되고, 두 번째 사이클 동안에 장벽 태그로써 올바른 장벽 상태를 페치(fetch)하는 데 소요되며 동시에 패킷의 에러 감지 코드(error detection code)는 에러를 발견하는 데 사용되고 에러가 발견된 패킷은 즉시 폐기한다. 세 번째 사이클에는 상태(state)를 수정하고 네 번째 사이클에서 상태를 SRAM(static RAM)에 기록하면서 장벽 유닛 동작의 4 사이클이 완료된다.

SRAM에 쓰이는 기재사항(entry)에는 그 패킷이 결합 상태, 멀티캐스트 상태, 확인 상태 중 어느 것인지를 나타내는 상태의 종류와 지금 현재의 장벽 동기화인지 다음 장벽 동기화인지를 나타내는 순열(sequence number) 비트, 트리 구조의 상위 부모 포트를 나타내는



(a) 스위치 모듈



(b) 장벽 유닛

그림 2 하드웨어 장벽 동기화에서의 스위치와 장벽 유닛 모듈

비트, 트리 구조를 나타내는 트리 정보(tree information) 비트, 결합 상태를 나타내는 비트, 그리고 확인 상태를 나타내는 비트 등이 기재되어 있다.

R. Sivaram의 연구[2]에서는 또 오류가 발생할 때에도 처리하는 방법을 프로토콜(protocol)을 통하여 설명하고 있는데 각 프로세서나 스위치에서는 일정 시간을 두어서 각 단계의 패킷이 도착하지 않으면 다시 요청을 하는 방법과 중간단계가 생략되고 다음 단계의 패킷이 도착할 경우 처리하는 방법을 사용하여 오류가 발생했을 때에도 정상적으로 처리할 수 있게 한다.

3. 멀티드롭방식을 사용한 장벽 동기화

3.1 필요한 하드웨어

다음 <그림 3>은 일반적인 장벽 동기화에 필요한 패킷의 구조이다. 이 패킷은 장벽 동기화의 정보만을 전달할 뿐 다른 데이타를 포함하여 전송하지는 않는다. 이 패킷의 구성요소로는 어떤 패킷인가를 알려주는 패킷의 종류와 스위치의 주소를 가리키는 필드, 해당 스위치에서 목적지와 연결된 출력포트를 나타내는 필드, 그리고 해당 스위치와 이전 스위치의 연결된 (즉 입력포트) 링크를 나타내는 필드로 구성되어 있다. 여기서 스위치의 주소와 출력포트, 이전 입력포트의 필드는 연결되는 스위치의 개수에 따라 더 연결될 수 있다. 이것이 정규망에서 사용된다면 그 규칙에 따라 (MIN이면 MIN의 종

류에 따라, 메쉬 구조이면 연속적인 순서로) 스위치가 여러 개가 연결되는 데 만약 이것이 다단계 상호 연결망에서 사용이 된다면 트리 구조에서 루트 프로세서가 있는 스위치에서 가장 최하위의 자식 스위치에 연결된 프로세서가 있는 스위치까지의 단계가 더 연결된다.

그러나, 이 패킷은 전체 시스템의 크기와 비례하여 커지게 되어 오버헤드(overhead)를 초래할 가능성이 크다. 따라서 전 단계를 가리키는 주소필드, 입출력 포트 필드, 이전 입출력 필드는 제거하고 다음 단계의 주소필드부터 패킷의 단계 필드에 연결하여 전송한다.

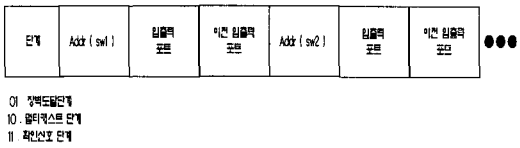


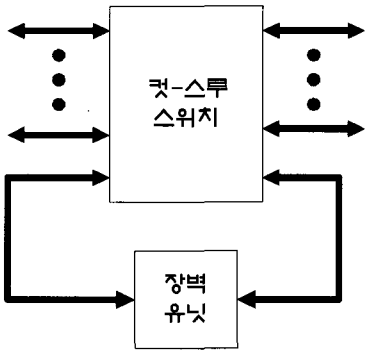
그림 3 패킷의 구조

단계는 2 비트로 상태를 표시한다. 즉, '01'일 때는 그 스위치에 연결된 모든 목적 프로세서에서 장벽 동기화가 이루어졌음을 나타내는 장벽 도달을 나타내고 '10'일 때에는 동기화에 참여하는 모든 프로세서들에게 다음 작업을 진행시켜도 좋다는 것을 나타내는 멀티캐스트 단계임을 나타내고 '11'일 때에는 멀티캐스트 패킷을 받은 스위치가 아무런 오류 없이 받았다는 뜻을 나타내는 확인 패킷임을 나타낸다. 따라서 '01'과 '11'일 때에는 루트 프로세서로 향하는 패킷이며 '10'일 때에는 다른 자식 목적지로 향하는 패킷임을 나타낸다.

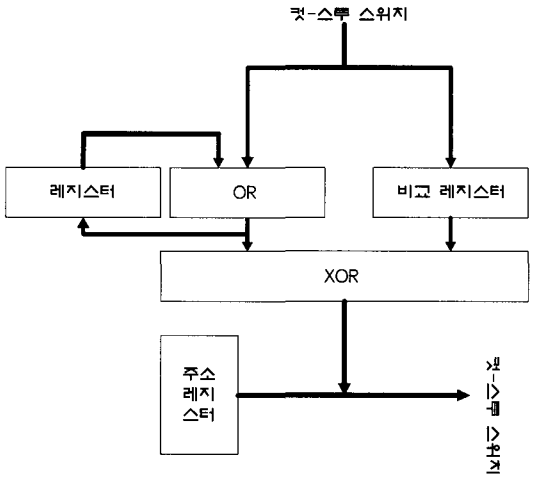
Addr(sw1)은 현재 단계의 스위치의 주소를 가리키며 Addr(sw2)는 다음 단계의 스위치의 주소를 가리킨다. 즉 장벽 동기화가 몇 단계를 거쳐서 이루어지는 가에 따라 스위치 주소의 개수가 결정된다. 본 논문의 정규망에서는 4비트로 정의하고 비정규 망에서는 3비트로 정의하였다

입출력포트를 나타내는 필드는 스위치의 종류에 따라 비트 수가 결정이 된다. 즉 (4 x 4) 스위치이면 8 비트, (8 x 8) 스위치일 경우에는 16 비트이다. 이것은 예를 들어 '10010001'이라면 (4 x 4) 스위치의 첫번째, 다섯번째, 여덟번째 출력포트가 목적지임을 가리키는 것이다.

이전 입출력 포트를 나타내는 필드는 해당 스위치와 이전 스위치와의 연결된 포트를 가리킨다. 이것도 입출력 포트를 나타내는 필드와 마찬가지로 스위치의 크기에 비례하여 결정 된다. 예를 들어 '00001000'이라면 이전 스위치의 네 번째 포트에서 해당 스위치로 연결이 된다는 것을 나타낸다.



(a) 스위치 모듈



(b) 장벽 유닛

그림 4 멀티드롭방식을 사용한 장벽 동기화를 위한 스위치와 장벽 유닛 모듈

위의 <그림 4>의 (a)는 본 논문에서 제안하는 스위치 모듈이다. 이것은 여러 개의 메시지를 복사할 수 있는 컷-스루 스위치를 사용하고 각 링크는 양방향(bi-directional)이 되는 것을 사용한다. 이것이 연속되는 장벽 동기화를 위한 것이라면 스위치 내부에 버퍼를 추가하여 사용하게 되겠지만 복잡한 내부 동작을 필요로 하므로 본 논문에서는 다루지 않았다.

<그림 4>의 (b)는 [그림 3.2]의 (a)에서 가리키는 장벽 유닛 모듈의 상세한 그림이다. 장벽 유닛 모듈의 구

성은 모두 3개의 레지스터와 OR 게이트, exclusive-OR(XOR) 게이트가 사용된다. 레지스터는 컷-스루 스위치에서 들어온 장벽 동기화의 단계 중 출력포트를 나타내는 비트를 저장하는 레지스터와 장벽 동기화에 참여하는 스위치에 종속된 모든 프로세서들을 나타내는 필드를 저장하는 비교 레지스터(compare register), 그리고 다음 단계의 스위치 주소를 저장하는 주소 레지스터(address register)가 필요하다. 이 주소 레지스터에는 패킷의 구조에서도 보았듯이 다음 단계의 스위치 주소 필드가 포함된다. 또 OR 게이트에서는 해당 스위치에 연결된 모든 프로세서들이 장벽 동기화에 이를 때까지, 즉 비교 레지스터에 저장된 값과 같아 질까지 그 신호를 계속 OR를 하는 데에 이용된다. XOR 게이트는 OR하는 신호가 비교 레지스터에 저장된 값과 일치하는지 비교하는 데 사용된다. 그리하여 이 값은 주소 레지스터에 저장되어 있는 값이 컷-스루 스위치로 진행이 이루어지게 하는 클럭(clock)으로 작용한다. 즉 '0'이면 주소 레지스터에 있는 값을 내보내고, 그렇지 않으면 계속 그 값을 가지고 있는 상태가 반복이 되는 것이다.

결과적으로 이 장벽 유닛에서는 3 사이클이 소요된다. 즉, 첫 번째 사이클에서는 컷-스루 스위치에서 들어온 패킷을 분해하여 비교 레지스터에 들어가거나 또는 OR 게이트에서 기존의 레지스터에 있는 값과 OR를 한다. 두 번째 사이클에서는 OR한 값들과 비교레지스터의 값을 XOR 게이트에서 비교한다. 그리고 마지막 사이클에서 다음 주소를 가지고 있는 주소 레지스터의 클럭으로 작동하여 주소 레지스터에 있는 주소로 메시지를 전송하게 된다.

3.2 오류를 방지하는 프로토콜

이 장에서는 성공적으로 장벽 동기화를 끝내기 위해서 필요한 스위치와 루트 프로세서, 그리고 참여 프로세서들에 의해서 사용되는 프로토콜에 대해서 설명을 한다. 우리는 먼저 어떠한 오류없이 정상적으로 장벽 동기화가 이루어지는 과정을 유한 상태 기계 다이어그램으로 설명하고 다음에는 오류가 발생하였을 때 처리하는 방법을 설명한다.

본 논문에서 제안하는 장벽 동기화는 세 가지 단계로 이루어져 있다. 보통의 경우 두 단계로 이루어지는 것과는 달리 추가된 마지막 단계는 메시지를 잃어버리거나 와전되는 등의 오류가 발생하였을 경우를 대비하여 이루어지는 과정이다.

첫번째 단계는 장벽 동기화에 참여하는 모든 프로세서들이 장벽에 도달하였다는 신호를 루트 프로세서에게 보내는 장벽 도달(barrier reached) 단계이다. 이 단계에서 루트 프로세서도 물론 장벽 동기화에 참여할 수 있으며 참여하지 않을 수도 있다. 이것은 정규 망 뿐 아니라 비정규 망에서도 적용된다. 또 각 스위치에서는 그 스위치에 연결된 참여 프로세서들이 모두 장벽에 도달하기 전까지 장벽 도달 신호를 상위 단계에 해당하는 스위치나 루트 프로세서에게 보내지 않는다.

두번째 단계는 참여 프로세서들로부터 장벽에 이르렀다는 신호를 받은 루트 프로세서가 나머지 루트가 아닌 참여 프로세서들에게 다음 작업을 진행시켜도 좋다는 신호를 보내는 멀티캐스트 단계이다. 본 논문의 프로토콜에서 멀티캐스트라는 의미는 헤더에 메시지를 포함해서 같이 전송을 하는 일반적인 의미의 멀티캐스트가 아니라 장벽 동기화 과정 중이라는 의미를 갖는 헤더만을 보내는 특별한 의미를 갖는다. 이 신호를 받은 루트가 아닌 참여 프로세서들은 다음 작업을 실행시킬 수 있다. 그러나 루트 프로세서가 장벽 동기화에 참여하고 있다면 모든 루트가 아닌 참여 프로세서들에게서 정상적으로 멀티캐스트 신호를 받았다는 신호를 받기 전에는 다음 작업을 실행할 수 없다.

세번째 단계는 참여 프로세서들이 루트 프로세서에게 멀티캐스트 신호를 아무런 오류 없이 잘 받았다는 확인 신호를 보내는 확인(acknowledgement) 단계이다. 이 단계에서는 루트프로세서가 장벽 동기화에 참여하지 않는다면 필요하지 않은 단계이다. 여기서 확인 신호는 루트가 아닌 참여 프로세서가 직접 루트 프로세서에게 보내는 것이 아니고 두 번째 단계인 멀티캐스트 단계에서 멀티캐스트 신호를 받은 중간 단계의 스위치에서 다음 단계의 스위치나 루트가 아닌 참여 프로세서로 멀티캐스트 신호를 전송하는 동시에 전 단계의 스위치나 루트 프로세서에게로 확인 신호를 전송하는 것이다. 따라서 루트 프로세서는 좀 더 빠르게 확인 신호를 받을 수 있으므로 전체의 장벽 동기화는 빠르게 이루어지는 것이다.

3.2.1 정상 동작

이 절에서는 정상적으로 아무런 오류 없이 장벽 동기화가 이루어질 때의 장벽 동기화에 참여하는 프로세서와 스위치의 프로토콜에 대해서 설명한다. 이것은 장벽 동기화의 순서대로 루트가 아닌 참여 프로세서로부터 스위치, 루트에 해당하는 순서로 설명한다.

<그림 5>는 루트가 아닌 참여 프로세서의 프로토콜을 나타내고 있다. 처음에 프로세서는 휴지(idle) 상태로 존재한다. 물론 해당 작업을 하고 있을 때에도 마찬가지이다. 그 상태에서 작업이 끝난 후 장벽에 도달하게 되면 장벽에 도달하였다는 장벽 도달 패킷을 상위 스위치

로 전송하게 된다. 그리고 프로세서는 멀티캐스트 신호를 기다리는 멀티캐스트 준비 상태가 된다. 이 상태에서 어느 정도의 시간이 지나도 멀티캐스트 신호가 도착하지 않으면 중간에 장벽 도달 패킷을 잃어버렸거나 와전된 것으로 해석하여 다시 장벽 도달 신호를 전송하게 된다. 이 패킷은 스위치의 레지스터 속에 계속 존재하고 있으므로 재전송이 가능하다. 멀티캐스트 준비 상태에 있던 프로세서로 멀티캐스트 신호가 도착한다면 확인 신호를 스위치로 전송을 하게 된다. 그 후에 다시 휴지 상태로 돌아가서 다음 작업을 할 수 있게 된다.

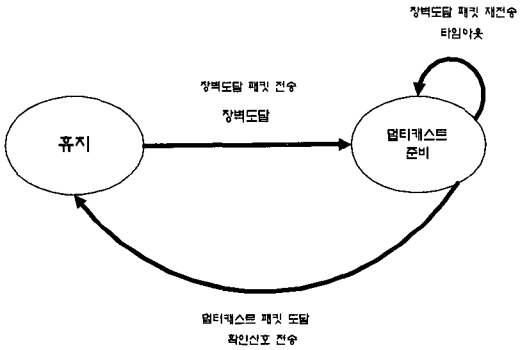


그림 5 루트가 아닌 참여하는 프로세서에 의해 수행되는 알고리즘을 위한 유한 상태 기계 다이어그램

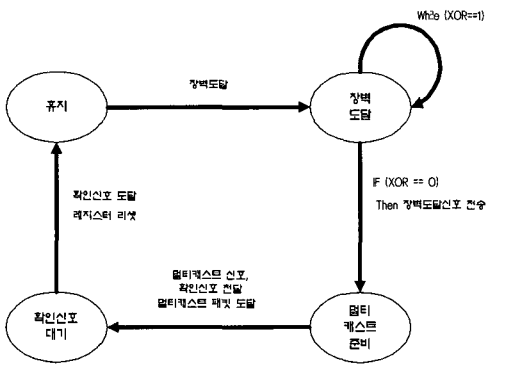


그림 6 스위치의 함수를 나타내는 유한 상태 기계 다이어그램

<그림 6>은 스위치의 기능을 나타내는 프로토콜을 보여주고 있다. 처음에 스위치는 휴지상태로 존재한다. 이 상태는 장벽 동기화가 진행되지 않는 상태를 포함한다. 즉 장벽 동기화가 아닌 다른 메시지를 전송하고 있는 상태도 휴지 상태이다. 그 상태에서 스위치에 연결된 프로세서들 중 하나가 장벽에 도달하게 되면 장벽 도달(barrier reached) 상태가 된다. 이 상태는 XOR 게이트의 출력 값이 '1'일 동안 계속되다가, XOR 게이트의 출력 값이 '0'이 되었을 때에 장벽 도달 신호를 상위 스위치나 루트에 해당하는 프로세서에게 전송하는 동시에 멀티캐스트 신호를 기다리는 멀티캐스트 준비 상태가 된다. 물론 스위치에 연결된 장벽 동기화에 참여하는 프로세서가 오직 한 개라면 멀티캐스트 준비 상태로 곧바로 되어야한다.

멀티캐스트 준비 상태에서 멀티캐스트 신호가 도착하면, 하위 단계의 스위치나 루트가 아닌 장벽 동기화에 참여하는 프로세서에게로 멀티캐스트 신호를 전송하는 것과 동시에 상위 단계의 스위치나 루트에 해당하는 프로세서에게로 확인 신호를 보낸다. 그리고 나서 확인 신호를 기다리는 확인 대기 상태가 된다. 이 상태에서 하위 단계의 스위치나 루트가 아닌 참여 프로세서로부터 확인 신호를 받으면 모든 레지스터의 값을 리셋(reset)시키면서 휴지상태로 돌아간다. 그리하여 모든 장벽 동기화 과정이 끝난다. 위의 모든 작업 중에서 스위치가 멀티캐스트 신호나 확인 신호를 기다리고 있는 상태에 있을 때, 얼마동안의 시간이 지나도 신호가 오지 않으면 중간에서 신호를 잃어버리거나 와전되었다고 인식하여 신호를 재 전송한다.

<그림 7>은 루트 프로세서에 의해서 수행되는 알고리즘의 프로토콜을 나타낸 유한 상태 기계 다이어그램이다. 이것은 처음에 휴지 상태에 있다가 두 가지 상태로 상황에 따라 바뀌게 된다. 즉 루트 프로세서가 장벽 동기화에 참여하는 경우, 루트 프로세서가 먼저 장벽에 도달하는 경우와 루트 프로세서보다 나머지 루트가 아닌 참여 프로세서들이 먼저 장벽에 도달하는 경우의 두 가지 경우로 나뉘게 된다. 전자의 경우 장벽에 도달하게 되면 다른 프로세서들이 장벽에 도달할 때까지 기다리게 되는 전 장벽 도달 대기 상태가 된다. 이 상태에서 장벽 동기화에 참여하는 다른 모든 프로세서들이 장벽에 도달하였을 때, 멀티캐스트 신호를 전송하고 확인 신호를 기다리는 확인 신호 대기 상태가 된다. 또 후자의 경우 다른 장벽 동기화에 참여하는 모든 프로세서들이 장벽에 도달하였다는 신호를 받은 루트 프로세서는 자신이 장벽에 도달하기를 기다리는 장벽 도달 대기 상태가 된다. 이 상태에서 자신이 장벽에 도달하게 되면 멀티캐스트 신호를 스위치나 다른 프로세서에게 전송하게 되고 확인 신호를 기다리는 확인 신호 대기 상태가 된

다. 확인 신호 대기 상태에서도 얼마동안의 시간이 지난 후에도 확인 신호가 도착하지 않으면 멀티캐스트 신호를 재전송하게 된다. 그런 다음에 확인 신호를 받으면 휴지 상태로 바뀌게 되어 다음 작업을 진행시킬 수 있게 된다.

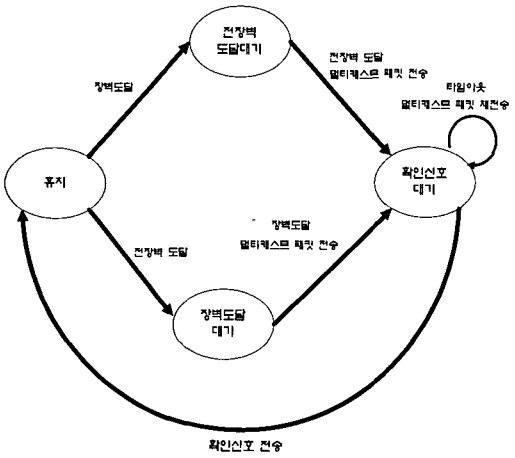


그림 7 루트에 의해 수행되는 알고리즘을 위한 유한 상태 기계 다이어그램

3.2.2 오류 발생시 동작

앞 절에서 서술한 바와 같이 본 논문에서 제안하는 장벽 동기화는 세 단계로 이루어진다. 즉, 참여 프로세서가 루트 프로세서에게 장벽에 도달하였다고 알려주는 장벽 도달 단계, 루트 프로세서가 그 외의 프로세서에게 다음 작업을 진행하여도 좋다는 멀티캐스트 단계, 그리고 멀티캐스트 메시지를 받은 스위치나 프로세서들이 그 전 단계의 루트 프로세서나 스위치에게 아무런 오류없이 받았다는 것을 알려주는 확인 단계의 세 단계로 이루어진다. 물론 각 단계에서는 메시지가 와전된다든지 중간에 소실될 여지가 있다. 이러한 문제는 장벽 동기화를 매우 느리게 하거나 다음 작업으로의 진행에 매우 큰 장애 요인이 된다. 그리하여 각 단계에서 오류가 발생하는 경우, 오류를 방지할 수 있는 프로토콜이 필요하다.

먼저 장벽 도달 단계에서의 오류가 발생하였을 경우를 살펴보면 스위치나 루트가 아닌 참여 프로세서들은 일정기간의 간격을 두고 장벽에 도달하였다는 장벽 도달 신호를 재전송한다. 그리하여 중간에 소실된 장벽 도달 신호가 있어도 다음 단계로 작업을 진행시킬 수 있게 한다. 이 때, 이런 과정이 존재하지 않는다면 장벽에 이르렀다는 신호를 기다리는 스위치에서는 언제까지나 신호를 기다리게 되어서 장벽 동기화가 이루어지지 않을 것이다. 만약 스위치에서 그 스위치에 연결된 장벽 동기화에 참여하는 프로세서들이 모두 장벽에 도달하여 다음 상위 단계의 스위치로 장벽 도달 신호를 보낸 후에도 이 장벽 도달 신호가 프로세서들로부터 오게 되면 그 신호들은 무시된다.

다음으로 멀티캐스트 단계에서 오류가 발생하였을 경우를 살펴보면 루트에 해당하는 프로세서는 나머지 모든 장벽 동기화에 참여하는 프로세서들에게 멀티캐스트 신호를 보내게 되고 이 신호를 받은 프로세서나 중간 스위치에서는 멀티캐스트 신호를 받자마자 바로 루트에 해당하는 프로세서에게 확인 신호를 보내기 때문에 장벽 도달 신호의 재전송 시간보다 결과를 빨리 알 수 있다. 그러므로 일정시간이 흐른 뒤에 멀티캐스트 신호를 재전송하여 중간에 오류가 발생하더라도 다음 동기화 작업을 진행할 수 있게 한다. 스위치에서는 멀티캐스트 패킷에 전송할 스위치나 프로세서의 주소를 가지고 있으므로 쉽게 재전송할 수 있다.

마지막으로 확인 단계에서 오류가 발생하였을 경우를 살펴보면 멀티캐스트 패킷을 받은 스위치나 장벽 동기화에 참여하는 루트가 아닌 프로세서에서는 멀티캐스트 패킷중 전의 입력포트 부분을 보고 그곳으로 확인 신호를 전송한다. 그리하여 확인 신호를 받은 전 단계의 스위치나 루트에 해당하는 프로세서에서는 바로 다음 작업을 할 수 있게 된다. 본 논문이 다른 연구에서보다 빠른 장벽 동기화를 가능하게 하는 부분이다. 이것은 일정시간이 지나도 재전송하지 않는데 그 이유는 확인 신호를 계속 기다리는 불필요함이 없이 다음 장벽 동기화가 진행되어 장벽 도달 신호를 전송하게 되면 그 전 단계의 장벽 동기화가 올바르게 끝났음을 알 수 있기 때문에 재전송하지 않는 것이다.

위와 같이 일정 시간이 지나면 같은 신호를 재전송하여 중간에 메시지를 잃어버리거나 와전되었을 때, 신뢰할 수 있는 장벽 동기화의 프로토콜이 완성됨을 알 수 있다.

4. 성능 평가 및 결과 분석

4.1 성능 평가 개요

이 장에서는 본 논문에서 제안하는 하드웨어 장벽 동기화와 기존의 하드웨어 장벽 동기화의 비교를 시뮬레이션을 통해 평가, 분석한다. 본 논문에서는 실제 시스템의 동작과 관련된 다양한 환경 혹은 시스템 내에서 존재할 수 있는 여러 가지 변화 요인들에 의하여 발생

하는 영향을 시뮬레이션 모델에 적용시켜서 이에 대한 결과를 평가 및 분석하기 위한 도구로써 컴퓨터 시뮬레이션을 수행하였다.

본 논문에서는 Scientific and Engineering Software Inc.의 SES/Workbench Release 3.0[9]을 사용하여 확률적 이산 사건 모델링(discrete event modeling)으로 시뮬레이션을 수행하였다. (8 x 8) 스위치를 사용하는 64 노드 시스템에서 각각 장벽 동기화에 참여하는 노드의 수를 달리하여 각각 시뮬레이션을 수행하였다.

4.2 성능 평가 요소

본 논문에서는 성능 평가 요소로서 각 프로세서가 장벽에 도달하고 난 후부터 모든 프로세서와 스위치가 다른 작업을 수행할 수 있는 모든 작업 동기화가 끝날 때까지의 평균 지연 시간(Average Delay : AD)에 근거하여 성능 평가를 수행하였다.

도착한 패킷의 수를 N, 장벽 동기화가 시작되는 장벽 도달 신호 패킷 j가 생성된 시간을 Gj, 장벽 동기화가 모두 끝나는 확인 신호 패킷이 스위치에 도착한 시간을 Aj라고 할 경우 장벽 동기화의 평균 지연 시간은 다음과 같다.

평균지연시간;

$$AD=\frac{\sum_{j=0}^{N}(Aj-Gj)}{N}$$

평균 지연 시간은 각 프로세서가 장벽 동기화를 수행하라는 신호를 각 장벽 동기화에 참여하는 모든 프로세서들에게 보내서 각 프로세서들이 장벽에 도달하였다는 장벽 도달 신호 패킷을 상위 스위치에 보낸 시점으로부터 모든 프로세서가 장벽 동기화를 이룬 뒤 마지막 프로세서와 스위치가 다음 작업을 진행시켜도 좋다는 확인 신호 패킷을 받았을 때까지의 평균소요시간을 의미하고 단위는 1 단위 시간(unit time)으로 정의한다. 이 시뮬레이션에서는 연속적인 장벽 동기화의 경우는 고려하지 않았다.

4.3 시뮬레이션을 위한 가정

본 논문에서 제안하는 하드웨어 장벽 동기화와 기존의 정규망에서의 하드웨어 장벽 동기화를 위한 시뮬레이션은 다음과 같은 가정 하에서 수행하였다.

가정 1 : 모든 스위치는 동기적으로 동작한다.

가정 2 : 모든 스위치는 버퍼를 갖지 않는다.

가정 3 : 장벽 동기화에 참여한 각 프로세서가 장벽에 도달하는 데 걸리는 시간은 4-12 단위시간(unit time)으로서 각 프로세서마다 임의로 시간이 정해진다.

가정 4 : 두 개 이상의 프로세서에서 동시에 장벽에 도달하여 스위치에 같은 신호를 보내는 경우 한 패킷이 스위치에서 처리되는 동안 다른 한 패킷은 기다리고 있어야 하며 그 시간은 어떤 스위치를 사용하는 가에 따라 다르지만 같은 구조 내에서는 같은 시간이 소요된다.

가정 5 : 스테이지 사이클은 임의의 패킷이 요구하는 출력 링크가 사용 가능하고 다음 스위치 소자에 공간이 있을 경우 다음 스테이지로 전송되기까지 소요되는 시간을 의미하는 데 이것은 1 단위 시간으로 한다.

가정 6 : 스위치와 프로세서간의 전송 시간은 1 단위 시간으로 한다.

가정 7 : 한 장벽 동기화 작업이 진행되는 동안 다음 장벽 동기화 작업이 진행되지 않게 한다.

4.4 시뮬레이션 결과 분석

위에서와 같은 가정 아래에서 기존에 연구되었던 하드웨어 장벽 동기화와 본 논문에서 제안하는 하드웨어 장벽 동기화를 비교하였다. 그 중 (8 x 8) 스위치를 사용하는 64 노드에서 장벽 동기화에 참여하는 프로세서의 수가 4개에서 64개까지 4개의 간격으로 변화시키면서 시뮬레이션만을 수행하였다. 그래프의 가로축과 세로축 매개변수는 평균 지연 시간과 장벽 동기화에 참여하는 프로세서, 즉 노드의 수를 나타낸다.

<그림 8>은 본 논문에서 제안하는 하드웨어 장벽 동기화와 기존에 연구되었던 하드웨어 장벽 동기화의 평균 지연 시간을 비교한 것이다. 이 그림은 64노드 시스템의 경우에 장벽 동기화에 참여하는 노드의 수의 변화에 따라 나타낸 그래프이다.

그래프에서 나타났듯이 장벽 동기화에 참여하는 노드의 수가 증가할수록 평균 지연 시간이 증가하는 것은 기존의 하드웨어 장벽 동기화와 본 논문에서 제안하는 하드웨어 장벽 동기화의 두 경우 모두 나타난 것을 알 수 있었다. 이러한 이유는 가정에서 나타났듯이 장벽에 도달한 프로세서들의 장벽 도달 신호를 동시에 입력이 들어왔을 때 처리하지 못하기 때문이다. 이것은 앞으로 연구할 과제이다.

하지만 그래프에서 나타났듯이 본 논문에서 제안하는 하드웨어 장벽 동기화의 평균 지연 시간이 기존의 하드웨어 장벽 동기화의 평균 지연 시간보다 최고 24.8%에서 최저 24.6%의 감소를 보였다. 평균적으로 24.7%로 기존의 연구에서보다 평균 지연 시간이 짧아졌음을 알 수 있었다.

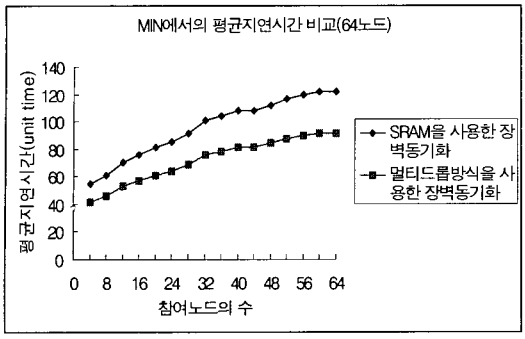


그림 8 MIN에서의 장벽 동기화의 평균지연시간 비교(64노드)

5. 결 론

본 논문은 공유 메모리 병렬 시스템에서 사용하는데 가장 핵심적인 작업중 하나인 동기화 중 가장 널리 알려진 장벽 동기화를 하드웨어적인 방법을 사용하여 빠르게 구현하기 위한 연구이다.

이 연구에서는 빠른 동기화를 구현하기 위해서 멀티 드롭방식을 사용한 새로운 스위치 구조를 제안하고 있으며 이 구조는 스위치에서 장벽 동기화의 전 과정을 작동(switch-driven)하는 방식이 아니라 프로세서에서 보내어지는 패킷과 멀티 드롭방식을 사용하여 프로세서에서 장벽 동기화의 과정을 주로 동작하고(processor-driven) 스위치는 단순히 메시지의 복사와 다음 스위치로의 진행 주소를 가르쳐주는 것만을 담당하기 때문에 정규망에서 뿐 아니라 임의의 구조에서도 통용될 수 있다. 임의의 구조인 비정규 망에서도 사용 가능하게 하려면 이 구조에 멀티 드롭 경로에 근거한 다중 전송 웜 방법을 사용하여 비정규망을 정규적인 트리 형태로 바꾼 다음에 장벽 동기화를 실행시키면 될 것이다.

이 연구에서 제안된 구조는 빠른 동기화 외에 전송도중 메시지를 잃어버리거나 와전되는 것 같은 간단한 오류가 발생하였을 때에 오류를 보완할 수 있도록 설계하였다.

성능 평가 및 분석을 통하여 볼 때, 다단계 상호 연결망과 같은 정규 망에서 64 노드 시스템의 경우 본 논문에서 제안하는 효과적인 하드웨어 장벽 동기화가 기존에 연구되었던 하드웨어 장벽 동기화보다 평균 지연 시간이 24.6% ~ 24.8% 감소하였다. 이것이 어느 정도 한계를 벗어나지 못한 이유는 정규망에서는 스위치의 단계가 일정하기 때문에 장벽 동기화의 단계 역시 일정한 데서 비롯된 것이다.

본 논문에서는 연속적인 장벽 동기화가 아닌 한 번의 장벽 동기화를 염두에 두고서 개발하여 메시지 충돌이 라든지 연속적인 장벽 동기화도 할 수 있는 버퍼를 사용한다든지 하는 부분은 고려하지 않았다. 또, 성능 평가 부분도 장벽 동기화에 참여하는 경우만 고려하였지, 다른 일반 메시지도 전송되는 일반적인 상황에서 실험하지 못하였다. 그리고 하드웨어적인 방법이 소프트웨어적인 방법보다 큰 장점을 가지고 있는 start-up overhead도 고려하지 못하였다. 앞으로 이런 부분을 고려한 연구가 필요하다.

참 고 문 헌

[1] Kai Hwang, *Advanced Computer Architecture,* McGraw Hill, International Editions, 1993.

[2] R. Sivaram, C. Stunkel, and D. K. Panda, "A Reliable Hardware Barrier Synchronization," *11th International Parallel Processing Symposium*, pp.274-280, Apr. 1997.

[3] H. F. Jordon, "A Special Purpose Architecture for Finite Element Analysis," *Proc. Int'l Conf. on Parallel Processing*, pp. 263-266, 1978.

[4] M. T. O'keefe, and H. G. Dietz, "Hardware barrier Synchronization : Static Barrier MIMD (SBM)," *Proc. Int'l Conf. on Parallel Processing*, pp. 35-42, 1990.

[5] C. J. Beckmann, and C. D. Polychronopoulos, "Broadcast Networks for Fast Synchronization," *Proc. Int'l Conf. on Parallel Processing*, pp. I:220-224, 1991.

[6] R. Kesavan, and D. K. Panda, "Multicasting on Switch-based Irregular Networks using Multi-drop path-based Multidestination Worms," *PCRCW '97* pp. 179-192, June 1997.

[7] T. Muhammad, "Hardware Barrier Synchronization for a Cluster of Personal Computer," http://garage.ecn.purdue.edu/~papers/TARIQ_SLIDES/.

[8] D. K. Panda, "Fast Barrier Synchronization in Wormhole k-ary n-cube Networks with Multi-destination Worms," *International Symposium on High Performance Computer Architecture*, pp. 200-209, 1995.

[9] *SES/Workbench Rel. 3.0*, Scientific and Engineering Software, Inc., 1995. tally

이 준 범
1997년 서강대학교 공과대학 전자계산학과(학사). 1999년 서강대학교 공과대학 전자계산학과 대학원(석사). 현재 삼성전자 컴퓨터 사업부 시스템개발그룹. 관심분야는 병렬 컴퓨터 구조, 워크스테이션 클러스터링, interconnection network

김 성 천
정보과학회논문지 : 시스템 및 이론
제 27 권 제 2 호 참조