CC-NUMA 시스템에서의 동기화 기법에 대한 성능 비교

(Performance Comparison of Synchronization Methods for CC-NUMA Systems)

문 의 선 † 장 성 태 †† 전 주 식 †††

(Eui Sun Moon) (Seong Tae Jhang) (Chu Shik Jhon)

요 약 동기화는 병렬 프로그램의 수행이 정확하게 이루어지도록 하기 위해 공유 데이타나 프로그램 상의 임계구간(critical section)에 대해 배타적인 수행을 보장하는 것을 목적으로 한다. 배타적인 프로그램의 수행은 병렬 프로그램의 병렬성을 제한하므로 효율적인 동기화는 높은 성능의 병렬 프로그램 수행을 위해 반드시 필요하다. 이런 필요에 의해 응용 프로그램이나 시스템의 특성을 이용하여 동기화의 성능을 높이는 기법들이 고안되었다. 본 논문에서는 모의실험을 통해 캐시에 기반을 둔 NUMA(Non-Uniform Memory Access) 시스템에서 나타나는 기존 동기화의 비효율성을 분석하여 제시하고, 이 비효율성을 제거할 수 있는 Freeze&Melt 동기화 기법과의 성능을 비교한다. 제시된 결과를 통해 Test-and-Test&Set 동기화는 동기화 과정에서 발생하는 방송(broadcast) 작업에 의해 비효율이 발생하고, QOLB(Queue-On-Lock-Bit) 동기화는 공유 데이타나 임계구간을 수행할 프로세서의 순서가 미리 정해져 있다는 점에 의해 비효율이 발생함을 확인할 수 있다. 이와 같은 단점들을 극복하고자 제안된 Freeze&Melt 동기화를 이용하여 임계구간을 수행하기까지 대기하는 시간과 임계구간을 수행하는 시간을 줄이고, 클러스터간의 통신량(traffic)을 감소시킴으로써 성능의 향상을 이룰 수 있다.

Abstract The main goal of synchronization is to guarantee exclusive access to shared data and critical sections, and then it makes parallel programs work correctly and reliably. Exclusive access restricts parallelism of parallel programs, therefor efficient synchronization is essential to achieve high performance in shared-memory parallel programs. Many techniques are devised for efficient synchronization, which utilize features of systems and applications. This paper shows the simulation results that existing synchronization methods have inefficiency under CC-NUMA(Cache Coherent Non-Uniform Memory Access) system, and then compares the performance of Freeze&Melt synchronization that can remove the inefficiency. The simulation results present that Test-and-Test&Set synchronization has inefficiency caused by broadcast operation and the pre-defined order of Queue-On-Lock-Bit (QOLB) synchronization to execute a critical section causes inefficiency. Freeze&Melt synchronization, which removes these inefficiencies, has performance gain by decreasing the waiting time to execute a critical section and the execution time of a critical section, and by reducing the traffic between clusters.

† 정 회 원 : (주)NextecNCS CIO 연구원
moon@comp.snu.ac.kr
†† 종신회원 : 수원대학교 전자계산학과 교수
stjhang@mail.suwon.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학과 교수
csjhon@riact.snu.ac.kr
논문접수 : 1999년 7월 2일
심사완료 : 2000년 3월 4일

1. 서 론

병렬 프로그램은 프로그램 내의 종속성(dependency)이 보장되도록 병렬적으로 수행되는 각 스레드(thread)의 동작을 조정해야 한다. 이로 인해 프로그램에 의해 수행되는 작업이 묵시적으로 의도하고 있는 순서만으로 종속성의 보장이 불충분할 경우에는 명시적인 동기화 명령이 필요하게 된다. 동기화는 프로그램의 수행이 배타적으로 이루어지도록 함으로써 병렬 프로그램이 정확

하게 동작하도록 하는 것을 목적으로 한다. 따라서, 동기화는 병렬 프로그램의 병렬성(parallelism)을 제한하는 요소일 수밖에 없으므로 전체 성능에 많은 영향을 끼치게 되며, 결과적으로 높은 계산능력에 대한 요구가 많아질수록 효율적인 동기화에 대한 필요성이 증대된다. 이와 같은 이유로 다중프로세서(multiprocessor) 시스템은 동기화에 대한 여러 가지 여러 기법들을 제공한다. Dijkstra[1]와 Knuth[2]는 원자적(atomic)인 읽기와 쓰기 기능만으로도 배타적인 수행을 보장할 수 있음을 증명하였으나, 일반적으로 효율적인 동기화를 위해 전용의 명령어(primitive)가 하드웨어적으로 제공되어 여러 알고리즘을 이용하여 동기화를 구현할 수 있도록 한다.

동기화 명령어는 compare&swap 명령어[3], fetch&Φ류의 명령어[4], LL(Load-Linked)과 SC(Store-Conditional) 명령어[5], 그리고 QOLB[6] 등을 예로 들 수 있다. 이 명령어들은 시스템이나 응용 프로그램의 특성에 따라 서로 다른 장점과 단점을 보인다. 따라서 시스템 설계자는 처리능력의 향상을 위해 시스템의 구조와 수행될 응용 프로그램의 특성을 고려하여 사용자 요구에 부합하는 동기화 명령어를 제공해야 한다. 한편, 동기화 명령어는 동기화를 위한 기본적인 기능만을 제공하므로 성능의 향상을 위해서는 동기화 알고리즘과 연동이 되어 구현된다. 이와 같은 예는 Test-and-Test&Set 동기화[7], LH와 M 동기화[8], wait-free 동기화[9], non-blocking 동기화[10], MCS 동기화[11] 등을 들 수 있다.

한편, 처리 능력에 대한 사용자의 요구가 높아지면서 시스템의 성능을 향상시키려는 노력은 부단히 이어졌다. 그 결과 시스템을 구성하는 각 소자의 성능을 향상시키려는 노력과는 별도로 여러 개의 프로세서를 이용하여 커다란 문제를 해결하기 위한 병렬 컴퓨터가 고안되기 시작하였다. 특히 공유메모리 다중프로세서 시스템은 각 프로세서에서 수행되는 작업들이 하나의 주소 공간을 공유하므로 프로그램을 작성하기 쉽고, 다중 프로세스 기반의 응용 프로그램들을 재사용하기에 용이하다는 장점[12]으로 인해 중요한 의미를 갖는 컴퓨터 구조 중의 하나로 취급된다. 초기에는 비교적 적은 수의 프로세서를 이용한 간단한 구조의 UMA(Uniform Memory Access) 시스템이 개발되었으나, 처리 능력에 대한 높은 요구로 인해 보다 많은 수의 프로세서가 필요하게 되어 NUMA 시스템에 대한 연구가 주류를 이루게 되었다.

이와 같이 시스템의 구조가 복잡해짐에 따라 비교적 단순한 구조의 시스템을 위해 고안된 동기화 기법들은 이전에는 나타나지 않았던 문제점을 드러내게 되었고, 이에 따라 새로운 동기화 기법을 필요로 하게 되었다. 본 논문에서는 가장 단순한 구조의 Test-and-Test&Set 동기화와 일반적으로 가장 좋은 성능을 보이는 것으로 인정받는 QOLB 동기화를 이용하여 기존 동기화 기법이 NUMA 시스템에서 보이는 문제점을 파악하고, NUMA 시스템을 위해 고안된 Freeze&Melt 동기화[13]에 대해 간략한 설명한 이후에, 각 동기화 기법간의 성능과 특징을 모의실험을 통해 비교한다.

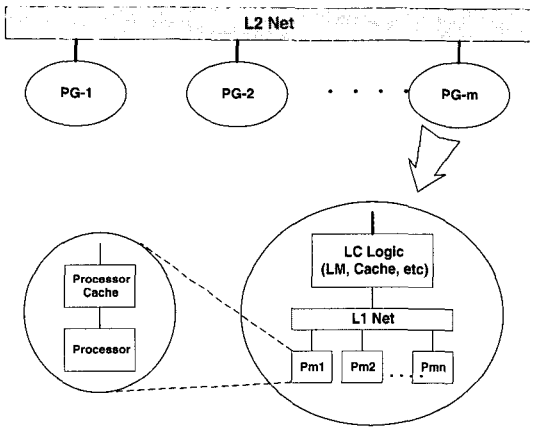


그림 1 CC-NUMA 시스템의 개념적 구조

2. 시스템 모델 및 모의실험 환경

그림 1은 클러스터 구조의 CC-NUMA 시스템의 개념적인 구조도를 나타낸 것이다. 본 논문에서 진행되는 논의를 위해 반드시 클러스터 구조를 가정할 필요는 없으나, 클러스터 구조의 NUMA 시스템은 각 프로세서의 메모리 참조에 대한 지연(access latency)이 확연히 구별되는 특징이 있으므로 논의의 편의성이 높고, 특히 최근들어 다중프로세서 시스템을 위한 기능이 지원되는 *Intel Pentium* 프로세서와 같은 상용(commercial off-the-shelf)의 프로세서를 이용하여 클러스터 구조의 NUMA 시스템을 개발하는 추세가 증가되고 있으므로 그림 1의 구조를 가정하였다. PG(Processor Group)라고 명명된 각 클러스터간의 메모리 참조는 L2 Net이라는 이름의 상호연결망을 통하여 이루어진다. 각 클러스터는 네 개의 프로세서 모듈, 지역 메모리 그리고 원격 메모리에 대한 참조 시간을 줄이기 위한 원격 캐시로 구성된다. 각각의 프로세서 모듈은 하나의 프로세서와 하나의 프로세서 캐시로 구성되며, 클러스터 내에서의

데이타 전송은 L1 Net을 통해 이루어진다. 본 논문에서는 L2 Net과 L1 Net이 각각 점대점 링크를 이용한 링(ring)과 버스 구조를 가진다고 가정한다. 이 외에 모의실험에 사용된 시스템 모델은 표 1과 같은 특성을 갖는다.

표 1 시스템 환경 변수

변수 종류	변수 값	변수 종류	변수 값
클러스터당 프로세서의 수	4	프로세서 동작 클럭	500 MHz
클러스터의 수	2, 4, 8, 16	L1 Net 클럭	100 MHz
프로세서 캐시의 크기	512 KB	L2 Net 클럭	100 MHz
원격 캐시	2048 KB	상호연결망 폭	64 bit
캐시 라인의 크기	32 B	캐시 연관성	1

본 논문에서는 프로그램 구동 방식의 모의실험 도구인 Mint[14]를 이용하여 그림 1에 나타난 시스템 구조에서 SPLASH-2[15] 벤치마크 프로그램을 수행되는 것을 모의실험 하였다.

3. Test-and-Test&Set 동기화

Test-and-Test&Set(TTS) 동기화는 동기화 변수의 획득(acquire)과 방출(release)을 위해 각각 Test&Set 명령과 Unset 명령을 사용한다. TTS 동기화에서는 대부분의 다른 동기화에서와 마찬가지로 동기화 변수의 획득을 요구하는 프로세서들간의 경쟁을 통하여 임계구간을 수행할 프로세서가 결정된다. 이 사실은 방출된 상태의 동기화 변수에 가장 먼저 획득 요구를 수행하는 프로세서가 그 동기화 변수를 획득하게 됨을 의미한다. 따라서, 동기화 변수에 대한 참조지연이 가장 작은 프로세서가 그 동기화 변수를 획득하고 임계구간을 수행하게 된다. 한편, TTS 동기화에서는 동기화 변수도 일반 공유 데이타와 마찬가지로 동기화 변수의 값에 변동이 생길 때마다, 즉 획득과 방출이 발생할 때마다 방송을 통해 해당 동기화 변수를 공유하고 있는 모든 프로세서에게 반영이 된다. 이로 인해 획득과 방출 작업이 수행완료(perform)되는 시점은 방송 작업이 완료된 이후가 된다. 방송 작업은 방송이 아닌 작업에 비해 많은 시간을 소요하므로, 획득과 방출이 수행되는 시간을 증가시키는 요인이 된다[16]. 그림 2는 TTS 동기화를 사용하는 경우, 각 벤치마크를 수행하는 과정에서 방송에 의해 지연시간이 증가하는 경우가 발생할 확률을 나타낸다. 한번의 임계구간 수행마다 획득과 방출이 각각 한번씩 수행되는 것을 고려하면, 그림 2에 나타난 확률은 많은 지연이 유발될 것임을 쉽게 예상할 수 있다.

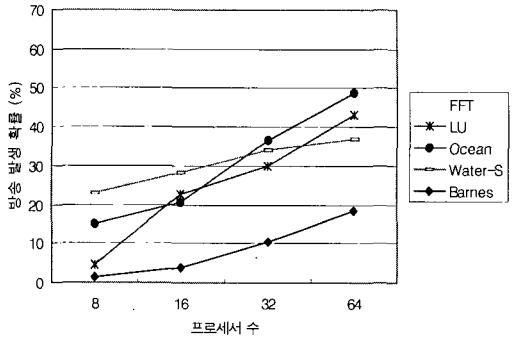


그림 2 방송에 의해 지연이 발생할 확률

4. QOLB 동기화

QOLB 동기화는 동기화 변수와 일반 공유 데이타에 대한 처리를 다르게 함으로써 TTS 동기화에서 나타나는 방송에 의한 문제를 아주 훌륭히 해결한 동기화 기법이다. QOLB 동기화는 TTS 동기화에서 방출이 일어날 때마다 발생하는 프로세서들간의 경쟁을 일으키지 않는다. 대신에 각 프로세서들이 획득을 요구한 순서에 의해 미리 동기화 변수를 획득할 차례를 결정하여 큐(queue)로 구성한다. 따라서, 방출이 일어나면 방송에 의해 다른 프로세서들에게 알려지는 대신, QOLB 큐 상의 다음 차례에 해당하는 프로세서에게만 전달되도록 함으로써 방송 작업이 발생하지 않도록 한다. 이로써 TTS 동기화에서 발생하는 방송에 의한 지연을 줄일 수가 있다.

반면에 QOLB 동기화는 오직 획득을 요구한 순서에 의해 프로세서를 결정하게 되므로 동기화 변수를 가장 빨리 획득할 수 있는 프로세서를 선정할 기회는 사라지고 만다. 그림 1의 시스템 모델에서 동기화 변수를 방출하는 프로세서와 다시 그 동기화 변수를 획득하는 프로세서가 같은 클러스터에 존재하는 경우를 내부전달(intra-transfer)이라 하고, 그렇지 않은 경우를 외부전달(inter-transfer)이라 하면, 내부전달은 외부전달에 비해 획득과 방출에 소요되는 시간이 더 작다. 뿐만 아니라, 임계구간을 수행하는 과정에서 참조되는 데이타 또한 동기화 변수를 방출한 프로세서의 클러스터 내의 캐

시에 저장되어 있을 확률이 높으므로, 임계구간의 수행에 소요되는 시간도 내부전달의 경우가 외부전달에 비해 더 작다. 이와 같이 내부전달에 비해 많은 지연을 유발하는 외부전달이 발생할 확률을 측정하면 그림 3과 같이 나타난다. 그림 3에 나타난 결과에 따르면 성능 개선의 여지가 많음을 알 수 있다.

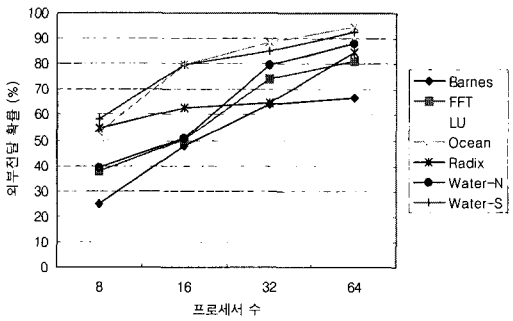


그림 3 외부전달 발생 확률

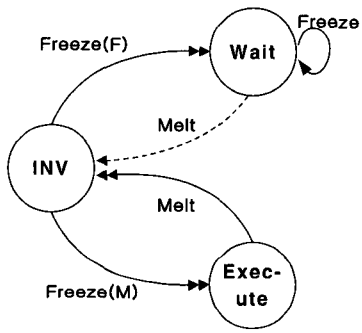
5. Freeze&Melt 동기화

Freeze&Melt 동기화[13]는 동기화 변수의 획득과 방출에 각각 Freeze 명령과 Melt 명령을 사용한다. 또한 이 두 명령어는 동기화 변수의 값을 변경하는 과정에 있어서 TTS 동기화의 Test&Set 명령과 Unset 명령의 기능과 매우 유사하다. 이는 Freeze&Melt 동기화가 TTS 동기화에 기반을 두고 있어서 프로세서간의 경쟁을 이용하여 동기화 변수를 획득할 프로세서를 선정하는 과정이 유사하기 때문이다. 이와 동시에 QOLB 동기화에서와 같이 동기화 변수를 일반 공유 데이타와 다르게 처리를 함으로써 획득과 방출 과정에서 발생하는 비효율성을 제거하는 것을 목적으로 하고 있다. 이 세 동기화 기법을 동기화 변수를 획득하는 프로세서를 선정하는 방식, 공유되는 동기화 변수를 처리하는 방식, 그리고 동기화를 위해 별도로 유지하는 부가 정보의 유무 측면에서 비교한 결과가 표 2에 나타나 있다.

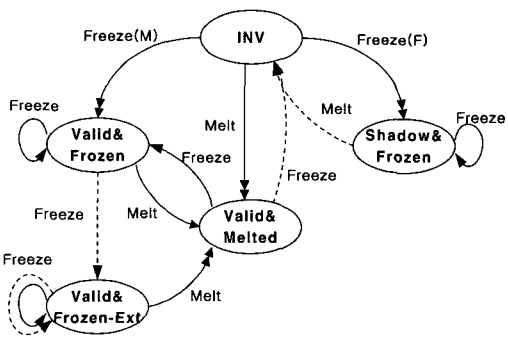
Freeze&Melt 동기화가 TTS 동기화에 비해 더 나은 성능을 보일 수 있는 이유는 QOLB 동기화에서와 같이 동기화 변수에 대해서는 일반 공유 데이타에 적용되는 것과 같은 캐시의 일관성(cache consistency)을 유지하지 않아도 된다는 것이다. 즉, QOLB 동기화에서 동기화 변수가 방출됨에도 불구하고 한 프로세서에게만 이런 사실이 반영되는 것과 마찬가지로 동기화 변수에 대

표 2 동기화 기법의 비교

	Test-and-Test&Set	Freeze&Melt	QOLB
선정 방식	경쟁	경쟁	FIFO (First In First Out)
공유 처리	일반 데이타와 동일	별도	별도
부가 정보	없음	없음	순서 정보



(a) 프로세서 캐시



(b) 원격 캐시 및 지역 메모리

그림 4 상태 전이도

한 캐시 일관성은 무시한다는 것이다. 즉, TTS 동기화에서는 동기화 변수가 획득되거나 방출될 때마다 이 변수를 공유하는 모든 프로세서에게 동시에 반영되어야 하지만, Freeze&Melt에서는 이 규칙을 느슨하게 적용시켜 동기화 변수의 변화를 먼저 반영한 프로세서가 아직 변화를 반영하지 않은 프로세서가 존재하는 시점에도 동기화 변수를 획득할 수 있도록 한다. 반면에 QOLB 동기화에서와 같이 미리 정해진 순서에 의해 동기화 변

수를 획득하는 방식을 택하는 대신, 경쟁에 의해 동기화 변수를 획득할 프로세서를 선정하도록 함으로써 동기화 변수에 대한 접근지연이 가장 작은 프로세서로 하여금 획득하게 한다. 이로 인해서 동기화 변수의 획득과 방출 과정에서 발생하는 비효율성을 제거함과 동시에 임계구간을 수행하는데 소요되는 시간도 줄일 수 있다. 결과적으로는 QOLB 동기화와 TTS 동기화의 장점을 채용하여 성능의 향상을 도모하는 것이다. Freeze&Melt 동기화를 지원하기 위해서는 프로세서 캐시, 원격 캐시 그리고 지역 메모리에 그림 4와 같은 상태의 전이가 추가되어야 한다.

Freeze&Melt 동기화를 사용함에 의해 두 가지 긍정적인 효과를 기대할 수 있다. 첫째는 동기화 변수를 획득하거나 방출하는데 소요되는 시간을 줄일 수 있다는 것이다. 둘째는 임계구간에 대한 캐시 적중률을 높임으로써 결과적으로는 임계구간을 수행하는데 소요되는 시간을 줄일 수 있다는 것이다.

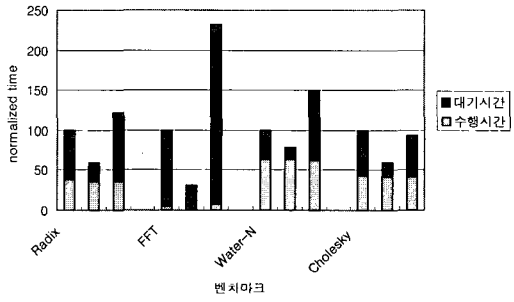
6. 실험 결과

응용 프로그램은 임계구간에 대한 경쟁률과 임계구간의 수행 과정에서 발생하는 메모리 참조의 수에 따라 성능상의 특징이 다르게 나타난다[17]. 따라서, 벤치마크를 경쟁률과 메모리 참조의 수에 의해 네 부류로 분류할 수 있으며, 표 3에 나타난 네 개의 벤치마크는 각각의 부류에서 하나씩 선정이 된 것이다. 이와 같이 각기 다른 특징을 나타내는 네 개의 벤치마크를 이용한 것은 특정 동기화 기법에 유리한 결과를 산출할 수 있는 벤치마크의 선정을 회피함과 동시에 보다 폭넓은 상황에 대해 고려하기 위함이다. 실험 결과에서는 이 네 벤치마크를 모의실험한 결과를 제시한다.

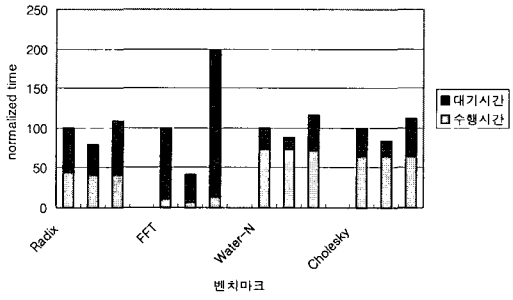
표 3 벤치마크 프로그램 환경

응용 프로그램	문제의 크기
Cholesky	tk15.O
FFT	65536 complex doubles
RADIX	1M keys
Water-NS	512 molecules

동기화 변수에 대한 획득 요구가 발생한 다음부터 임계구간의 수행이 종료될 때까지의 과정을 살펴보면, 첫 번째 단계는 동기화 변수의 획득에 성공하기까지의 대기단계이고, 두 번째 단계는 동기화 변수를 획득한 후에 임계구간을 수행하는 단계이다. 따라서 동기화 기간(synchronization period)를 크게 대기시간과 수행시간으로 구분할 수 있다. 그림 5는 각 동기화 기법을 이용하여 네 벤치마크를 수행한 경우의 동기화 기간을 QOLB 동기화의 동기화 기간과 비교하여 나타내고 있다. 각 벤치마크에 해당하는 세 개의 막대는 왼쪽부터 각각 QOLB 동기화, Freeze&Melt 동기화, TTS 동기화의 결과를 나타낸다. 이 결과를 살펴보면 Freeze&Melt 동기화는 네 개의 벤치마크 모두에서 가장 성능이 뛰어남을 알 수 있고, 그 성능의 향상은 프로세서의 수가 많아질수록 커짐을 확인할 수 있다.



(a) 64개 프로세서

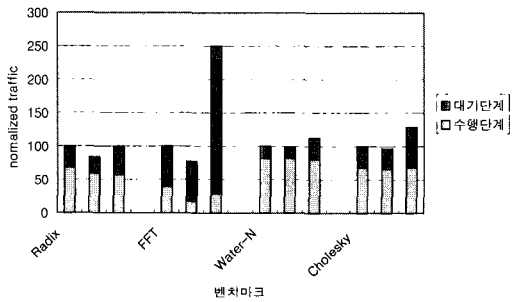


(b) 32개 프로세서

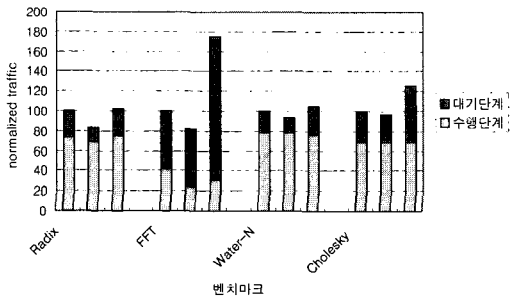
그림 5 동기화 기간(synchronization period)

클러스터간에 이루어지는 메모리 참조는 클러스터 내에서 이루어지는 메모리 참조에 비해 훨씬 많은 지연시간을 필요로 함은 자명하다. 따라서, 동기화가 진행되는 과정에서 발생하는 클러스터간 통신량은 동기화 성능에 많은 영향을 끼치게 된다. 그림 6은 동기화가 진행되는

과정에서 발생한 클러스터간 통신량을 대기단계와 수행단계로 구분하여 QOLB 동기화의 통신량과 비교한 결과이다. 그림 5에서와 마찬가지로 각 벤치마크에 해당하는 세 개의 막대는 왼쪽부터 차례로 QOLB 동기화, Freeze&Melt 동기화, TTS 동기화를 의미한다. 클러스터간 통신량을 비교한 결과에 의하면 Freeze&Melt 동기화가 다른 두 동기화 기법이 비해 우수한 특징을 지니고 있음을 알 수 있다.



(a) 64개 프로세서



(b) 32개 프로세서

그림 6 클러스터간 통신량

7. 결 론

이 논문에서 서로 대비되는 동기화 기법인 TTS 동기화와 QOLB 동기화의 성능을 Freeze&Melt의 것과 비교하였다. 모의실험의 결과 Freeze&Melt 동기화는 동기화 과정에서 소요되는 시간과 클러스터간 통신량 측면에서 우수함을 알 수 있었다. 이와 같은 성능의 향상은 임계구간에 대한 경쟁이 심할수록, 프로세서의 수가 늘어날수록 더 크게 나타난다. Freeze&Melt 동기화가 NUMA 시스템에서 더 나은 성능을 보이는 것은 TTS 동기화와 QOLB 동기화가 보이는 단점을 해결함으로써 가능했다. 즉, TTS 동기화에서와 같이 프로세서간의 경쟁을 이용해 동기화 변수에 대한 접근지연이 가장 작은 프로세서로 하여금 획득에 성공하도록 함으로써 동기화 변수를 획득하는 과정에서 발생하는 비효율성을 개선하고, 결과적으로 임계구간을 수행하는 과정에서 발생하는 메모리 참조에 대한 지연시간도 줄일 수 있었다. 그 결과 지연시간이 큰 클러스터간의 통신량을 줄임으로써 프로그램의 수행에 유리한 장점도 얻을 수 있었다. 이와 함께 QOLB 동기화에서와 같이 동기화 변수의 획득에 실패하는 프로세서로 인해 발생하는 불필요한 지연을 제거함으로써 동기화 변수의 방출과 획득 과정에서 소요되는 지연시간을 감소시킬 수 있었다. 이와 같은 동기화의 성능 향상은 병렬 프로그램의 병렬성이 제한되는 동기화 과정을 효율적으로 처리할 수 있도록 함으로써 전체 프로그램의 처리 능력이 향상되는 결과를 가져오게 된다.

반면에 Freeze&Melt 동기화에서 사용하는 동기화 명령어인 Freeze 명령은 QOLB 명령을 제외한 다른 동기화 명령과 같이 동기화 변수를 획득하고자 하는 경쟁 방식을 택한다. 즉, 가장 먼저 수행된 Freeze 명령에 의해 동기화 변수가 획득된다. 이와는 달리 QOLB 명령은 획득을 요구하는 프로세서들로 하여금 선입선출의 순서를 지키도록 한다. 동기화에서의 형평성(fairness)은 획득 요구가 발생한 순서대로 프로세서들이 동기화 변수를 획득할 수 있는지의 여부를 나타내므로 QOLB 동기화는 형평성을 지원하는 반면에, Freeze&Melt 동기화와 TTS 동기화는 그렇지 못하다는 특징이 있다. 그러나, 모의실험에 의하면 Freeze&Melt 동기화의 형평성은 TTS 동기화에 비해 큰 차이를 보이지 않음이 나타나므로 우려할만한 단점은 아니라고 판단되며, 반드시 형평성이 제공되어야 하는 경우에 있어서는 다른 동기화에서와 마찬가지로 동기화 알고리즘을 이용해서 해결할 수 있다.

Freeze&Melt 동기화는 링 구조의 PANDA 시스템[18]에 적용될 계획이다. 그러기 위해서는 먼저 형평성 문제에 대한 다양한 실험과 다양한 시스템 환경에 대한 실험이 선행되어야 하고, 이와는 별도로 Test-and-Test&Set이 채용된 현재의 PANDA 시스템에서 실제적인 실행궤적(real trace)을 추출하여 보다 실제적인 검증이 이루어져야 한다. 이들 각각에 대한 결과는 별도의 논문에서 자세히 언급될 것이다.

참 고 문 헌

[1] E. W. Dijstra, Solution of a Problem in Concurrent Programming Control, Communications of the ACM 8(9), 1965

[2] D. E. Knuth, Additional Comments on a problem in Concurrent Programming Control, Communication of the ACM 9(5), 1966

[3] R. P. Case, and A. Padegs, Architecture of the IBM System 370. Communication of the ACM, 21(1):73-76, 1978

[4] P. J. Woest, and J. R. Goodman, An Analysis of Synchronization Mechanisms in Shared-Memory Multiprocessors. Technical Report TR1005, University of Wisconsin-Madison, 1991

[5] E. H. Jensen, G. W. Hagensen, and J. M. Brouchton, A New Approach to Exclusive Data Access in Shared Memory Multiprocessors. Technical Report UCRL-97663, Lawrence Livermore National Lab, 1987

[6] J. R. Goodman, M. K. Vernon, and P. J. Woest. "Efficient synchronization primitives for large-scale cache-coherent shared-memory multiprocessors," In Proceedings of the 3rd Symposium on Architectural Support for Programming Languages and Operating Systems. 1989

[7] L. Rudolph, and Z. Seagall. "Dynamic decentralized cache schemes for MIMD parallel processors," In Proceedings of the 11th Annual International Symposium on Computer Architecture. 1984

[8] P. Magnusson, A. Landin, and E. Hagersten. Efficient Software Synchronization on Large Cache Coherent Multiprocessors. Technical Report T94:07, Swedish Institute of Computer Science, Feburuary 1994

[9] M. Herlihy. "Wait-free synchronization," ACM Transactions on Programming Language and Systems, 11(1), 1991

[10] S. Prakash, Y. Lee, and T. Johnson. "Non-blocking algorithms for concurrent data structures," Technical Report TR91-002, Univ. of Florida, 1991

[11] J. M. Mellor-Crummey, and M. L. Scott. "Algorithms for scalable synchronization on shared-memory multiprocessors," ACM Transactions on Computer Systems. 1991

[12] J. Laudon and D. Lenoski. The SGI origin: A CC-NUMA highly scalable server. In Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA'97), pages 241-251, June 1997

[13] E. S. Moon, S. T. Jhang, and C. S. Jhon. "Adjacency preferred hardware synchronization method for CC-NUMA systems," In Proceedings of International Conference on Electronics, Informations and Communications. 1998

[14] J. E. Veenstra and R. J. Fowler. "MINT: a front end for efficient simulation of shared-memory multiprocessors," In Proceedings of the 2nd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. 1994

[15] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. "Methodological considerations and characterization of the SPLASH-2 parallel application suite," In Proceedings of the 22th Annual International Symposium on Computer Architecture. 1995

[16] D. E. Culler and J. P. Singh, Parallel Computer Architecture, pp538-541, Morgan Kaufmann Publishers, INC, San Francisco, 1998

[17] E. S. Moon, S. T. Jhang, and C. S. Jhon, Analysis of the Relation of Synchronization Algorithm and Parallel Programs in Shared-Memory Multiprocessor Systems, Will be appear in Proceedings of High Performance Computing Symposium 2000, April 2000

[18] S. W. Chung, S. T. Jhang, and C. S. Jhon, "PANDA : Ring-Based Multiprocessor System using New Snooping Protocol," In Proceedings of International Conference on Parallel And Distributed Systems. 1998.

문 의 선

190년 서울대학교 컴퓨터공학과 학사. 1992년 서울대학교 컴퓨터공학과 석사. 2000년 서울대학교 컴퓨터공학과 박사. 1999년 ~ 현재 서울대학교 컴퓨터신기술공동연구소 특별연구원. 1999년 ~ 현재 (주) NextecNCS CIO. 관심분야는 병렬처리, ASIC, 하드웨어 설계 및 인터넷 통신

장 성 태

정보과학회논문지 : 시스템 및 이론 제 27 권 제 2 호 참조

전 주 식

정보과학회논문지 : 시스템 및 이론 제 27 권 제 2 호 참조