소프트웨어 분산공유 메모리를 위한 향상된 하이브리드 프로토콜

(An Improved Hybrid Protocol for Software Distributed Shared Memory)

이 성 우 † 김 현 철 † 유 기 영 †† 하 금 숙 †††

(Sung-Woo Lee)(Hun-Chul Kim)(Kee-Young Yoo)(Kum-Sook Ha)

요 약 최근 물리적으로 분산 메모리 하드웨어 상에서 공유메모리 프로그래밍 모델을 제공하는 3소프트웨어 분산 공유 메모리(Distributed Shared Memory, DSM) 시스템을 위해 여러 프로토콜이 등장하고 있다. 본 논문에서는 기존의 동적 복원 프로토콜인 하이브리드 프로토콜[11]의 성능향상을 제한하는 두 가지 문제를 밝혀내고 이를 개선하기 위한 향상된 하이브리드 프로토콜을 제안한다. 이 프로토콜은 동기화 시점에서 기존 프로토콜과 같이 과거에 어떤 페이지를 이미 접근한 프로세스에 대해서 복원 프로토콜을 적용할 뿐만 아니라 그 페이지에 접근한 프로세스의 수가 선택된 파라미터 값 이상이면 모든 프로세스에 대해 복원 프로토콜을 적용한다. 제안한 프로토콜을 DSM 시스템인 CVM에 구현하고, 100Mbps인 Ethernet으로 연결된 8대의 Sun ultra1상에서 6개의 응용 프로그램에 대해 성능평가를 수행하였다. 그 결과 원격 프로세스에 대한 수정정보 요구 메시지의 수를 평균 16% 감소시켰고, 4개의 응용프로그램에서 2-5%의 성능향상을 얻었다.

Abstract Recently, many different protocols have been proposed for software Distributed Shared Memory (DSM) that provides a shared-memory programming model on distributed memory hardware. This paper identifies two problems that deteriorate the performance of a hybrid protocol, an adaptive invalidate/update protocol, and then proposes an improved hybrid protocol to solve these problems. At the point of synchronization, the proposed protocol applies an update protocol not only to the processors previously accessing the page, as in the existing protocol, but also to all processors if the number of processors previously accessing the page is larger than a given parameter value. The proposed protocol was implemented in CVM, a software DSM system, and evaluated on eight Sun-ultra1 workstations. For these applications we obtained 16% reduction on the average in the number of remote modification-data request messages, which results in 2-5% improvement in execution time for four applications.

1. 서 론

최근에 높은 속도의 네트워크의 출현으로 네트워크로 연결된 워크스테이션(Network Of Workstation, NOW)은 병렬 컴퓨팅을 위한 효율적인 환경으로 인식되었다. 이들 NOW상에서의 프로그래머들은 주로 PVM[1] 혹은 MPI[2] 등과 같은 메시지 패싱 라이브러리를 이용하였다. 그러나 메시지 패싱 모델 상에서는 언제 통신을 시작해야 할지, 누구에게 해야 할지, 무엇을 해야 할지를 프로그래머가 명시를 해줘야 한다. 이런 프로그래밍의 어려움을 극복하고자 물리적으로 분산 메모리 하드웨어 상에서 공유메모리 프로그래밍 모델을 제공하는 소프트웨어 분산 공유 메모리(Distributed Shared Memory, DSM) 시스템이 등장하게 되었다[3][4].

† 학생회원 : 경북대학교 컴퓨터공학과
swlee@purple.knu.ac.kr
hskim@purple.knu.ac.kr
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
yook@bh.knu.ac.kr
††† 종신회원 : 구미1대학 컴퓨터정보전공 교수
ksha@www.kumi.ac.kr
논문접수 : 2000년 1월 13일
심사완료 : 2000년 7월 10일

DSM을 위한 이론적 모델들 중 해제일관성(Release Consistency, RC) 모델[7]은 동기화 지점까지 공유메모리의 변화들을 연기시켰다가 한꺼번에 전송하는 프로토콜이 가능하게 한다. 또한 같은 메모리 페이지에 대해 여러 프로세스들이 동시에 쓰기를 가능하게 하여 거짓 공유의 악영향을 줄일 수 있는 다중쓰기 프로토콜(Multiple-writer protocol)[8]을 사용할 수 있게 한다.

RC 모델을 지원하기 위한 대표적인 프로토콜에는 빠른 무효화 일관성(eager invalidate, EI) 프로토콜[9], 늦은 무효화(lazy invalidate, LI) 프로토콜[10]과 늦은 하이브리드(lazy hybrid, LH) 프로토콜[11] 등이 있다. EI 프로토콜은 동기화 변수가 **해제**(release)될 때 일관성 유지를 위한 작업이 발생하도록 한다. 그에 반해 LI 프로콜은 동기화 변수가 **요구**(acquire)될 때 일관성이 유지되도록 한다. EI와 LI는 둘 다 수정된 데이타의 다른 프로세스의 복사본을 무효화시키는데 반해, 하이브리드 프로토콜은 무효화 기법과 복원(update)기법을 복합해서 사용하는 동적 복원 프로토콜이다[11][12]. Keleher[11]가 제안한 LH 프로토콜은 전체적으로는 무효화 프로토콜인 LI를 사용하지만 실시간에 이전의 접근 기록을 보고 특정 페이지들에 대해서 복원 프로토콜을 사용함으로써 성능 향상을 꾀하였다. 이들을 비교해 보면 응용프로그램에 따라 차이가 있지만 하이브리드 프로토콜이 두 프로토콜에 비해 상당히 우수함이 밝혀졌다[11].

본 논문에서는 LH 프로토콜의 성능향상을 제한하는 두 가지 문제점을 밝혀내고 이에 대한 해결책으로 다수의 프로세스에 의해 접근되는 페이지들에 대한 복원을 강조하는 동적복원 프로토콜을 제안한다.

LH 프로토콜의 두 가지 문제점은 첫 번째, 다른 프로세스의 페이지 접근 기록을 유지하는 복사집합(copyset)의 부정확성으로 인해 복원 프로토콜을 적용할 기회를 상실하는 것과 두 번째, 한 페이지에 대해 대부분의 프로세스가 접근했지만 소수의 접근하지 않은 프로세스들이 끼어있어 복원 프로토콜의 적용이 단절되는 것이다. 이와 같은 현상들은 락 동기화(lock synchronization)를 사용하는 응용프로그램에서 빈번하게 발생한다.

제안한 동적복원 프로토콜은 LH 프로토콜에서처럼 각 프로세스들은 페이지 x에 대해 다른 프로세스들이 복사본을 갖고 있는지의 정보, 즉 복사집합(copyset)를 유지하여, 동기화시점에서 x에 대한 쓰기정보를 전송할 때 복사집합에 있는 프로세스들에 대해서는 무효화 프로토콜 대신에 복원 프로토콜을 적용한다. 그러나, 본 프로토콜은 페이지 x의 복사집합에 있는 프로세스의 수가 α보다 크게 되면 x에 대해서는 무조건 복원 프로토콜을 적용한다. 이 아이디어의 근거는 첫 번째, 복사집합의 크기가 α이상이라면 복사집합에 포함되지 않은 다른 프로세스들이 이미 접근했거나 앞으로 접근할 가능성이 매우 높고, 두 번째, 만일 복원정보를 받은 프로세스 p가 페이지 x를 이후에도 자신은 접근하지 않는다 하더라도 이후에 복사집합에 속한 프로세스 q와 동기화가 발생하면 q에게 x의 복원 정보를 전달할 수 있다는 점이다.

본 연구에서는 실험용 DSM시스템인 CVM[13]에 LI, LH 그리고 제안한 프로토콜들을 구현하고 100Mbps로 연결된 8대의 Sun Ultra_1 워크스테이션들 상에서 성능을 평가한다.

본 논문의 2장에서는 DSM의 구현을 위한 모델과 기존에 연구된 프로토콜들을 소개하고 3장에서는 LH 프로토콜의 한계를 설명하고 4장에서 새로운 프로토콜을 제안한다. 그리고 5장에서 구현환경과 성능평가에 사용된 응용프로그램들을 설명하고 각 응용 프로그램에 대해서 프로토콜들의 성능을 평가, 분석한다. 6장에서는 성능에 중요한 영향을 미치는 파라미터 α의 값의 변이에 따른 성능의 변화에 대해 논의하고 7장에서는 결론과 향후 연구과제에 대해 설명한다.

2. 관련연구

2.1 늦은 무효화(LI) 프로토콜

LI 프로토콜[10]에서는 공유변수에 대한 수정의 전파가 **요구**시점까지 연기된다. **요구**시점에서 **요구**하는 프로세스는 RC의 정의에 따라 그것이 필요한 수정정보가 어느 것인지를 결정한다. 각 프로세스의 수행은 구간 인덱스로 표시되는 구간(interval)으로 나뉘어 진다. 프로세스가 **해제나 요구** 작업을 수행할 때마다 새로운 구간은 생성되고 구간 인덱스는 증가된다. LI 프로토콜에서는 모든 프로세스들의 구간들 사이의 순서는 부분 순서화(partial ordering)에 의해 결정된다[14]. 우선 한 프로세스 상에서 구간들은 프로그램 순서에 의해 결정된다. 그리고 다른 프로세스들에 속한 구간들끼리의 순서에 대해서는, 프로세스 q상의 한 구간이 프로세스 p상의 한 구간을 끝맺는 **해제**에 대응하는 **요구**로 시작한다면 p의 구간이 q의 구간보다 앞선다고 본다.

RC는 프로세스 p가 프로세스 q로부터 **요구**를 끝내기 전에, p가 부분순서에 의해 앞선 모든 프로세스들의 구간들 내에 발생한 수정사실을 알도록 요구한다. 그러므로 p가 **요구** 메시지를 q에게 보내면, q는 필요한 모든

구간들의 쓰기통보(write-notice)들을 수집하여 **해제**-**요구** 메시지에 실어 p에게 보낸다. 쓰기통보는 한 페이지가 특정 구간 내에서 수정되었음을 알려주지만 실제 수정을 포함하지 않는다. 쓰기통보를 받은 p는 해당 페이지를 무효화시킨다.

무효화된 페이지에 대해 접근하게 되면 페이지 실패(page fault)를 발생한다. 이때 페이지 실패를 발생한 프로세스는 현재 구간보다 부분순서에 의해 앞서는 구간들 동안 생성된 모든 차이본(diff)[8]들을 모아 그 페이지에 적용(apply)시켜야 한다. 이를 위해 프로세스 p는 차이본을 가지고 있지 않은 쓰기통보들을 모아 가장 최근에 도착한 쓰기통보를 생성한 다른 프로세스들에게 이들에 대한 차이본들을 요청하는 메시지를 보낸다.

2.2 늦은 하이브리드(LH) 프로토콜

LH는 어떤 페이지들에 대해 **해제**-**요구**시점에서 쓰기통보를 보내 무효화시키는 대신에 차이본을 보내 복원시키는 점을 제외하고는 LI와 같은 프로토콜이다[11]. 이 프로토콜은 한 프로세스가 특정 페이지에 대해 과거에 접근했다면 미래에도 다시 접근할 것이라는 가정에 의해 시간적 국부성을 이용한다. **요구**하는 프로세스 p가 과거에 접근했다고 알려진 페이지들에 대해서는 무효화하기 보다 복원해야 한다. 따라서 어느 정도 정적인 공유 패턴(sharing pattern)을 가지는 응용 프로그램은 이 프로토콜에 의해 통신량을 줄일 수 있다.

각 프로세스는 다른 프로세스들의 페이지 접근들을 추적하기 위해 복사집합(copyset)을 사용한다. 그 복사집합은 주어진 차이본을 원격프로세스에게 보내야 할지를 결정하는 기준이 된다. 프로세스 p가 프로세스 q에게 락허가(lock grant) 메시지를 전달할 때, 페이지 x의 쓰기통보들을 전달해야 하고 x의 복사집합에 q가 포함되어져 있다면, p는 자신이 갖고 있는 그 쓰기통보들의 차이본들을 락허가 메시지에 붙여보낸다.

배리어에 도착했을 때는 각 프로세스는 다른 프로세스들에게 보여지지 않았고 자신이 갖고 있는 쓰기통보들의 리스트를 만든다. 프로세스 p_i에서 생성된, 프로세스 p_j에 대한 리스트는 p_j에게 알려지지 않은 p_i가 생성한 모든 쓰기통보들로 구성되어 있다. p_i는 이 리스트에 속하는 쓰기통보들에 대응하는 차이본들을 모아 p_j에게 보낸다.

3. LH 프로토콜의 한계

본 연구에서는 LH 프로토콜을 사용하여 여러 가지 응용 프로그램의 동작을 분석한 결과, 프로토콜의 성능을 저하시키는 두 가지 문제점을 발견하였다. 첫 번째는 락 동기화 시점에서 **해제** 프로세스의 복사집합은 **요구** 프로세스가 발생시킨 가장 최근 구간(interval)의 접근 패턴을 반영할 수 없다. LRC는 부분 순서에 의한 현재 **요구**보다 앞서는 메모리 수정의 사건들만 인식하도록 요구하기 때문에 **해제** 프로세스는 **요구** 프로세스의 최근 쓰기 사건들을 알 수 없다. 그림 1에서 보듯이 **해제** 프로세스 p는 **요구** 프로세스 q의 I_c 에 속하는 구간내의 쓰기사건들을 알 수 없다. 그러므로 LH 프로토콜은 가장 주요한 최근에 접근된 페이지들에 대해 복원 프로토콜을 적용할 기회를 상실한다.

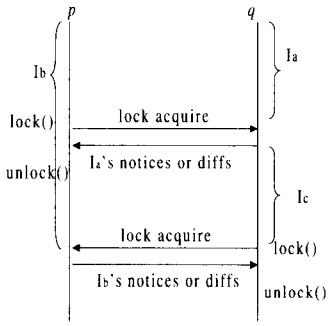

그림 1 락 동기화에 따른 쓰기 통보와 차이본의 이동

이런 복사집합의 부정확성은 락 동기화를 사용하는 응용프로그램에서 나타난다. 특정 응용프로그램이 배리어에 도달하면 모든 페이지들을 일관성을 유지시키므로 복사집합도 일치되는데 반해 락 위주의 응용 프로그램에서는 복사집합이 정확하게 되기 위해서는 상당한 시간이 걸린다. 또한 전체 프로세스의 수가 커질수록 부정확성은 오랫동안 나타난다. 따라서, 기존의 하이브리드 프로토콜은 복사집합의 부정확성으로 인해 복원 프로토콜을 적용할 기회를 상실할 수 있다.

두 번째 문제는 차이본 전송의 단절현상이다. 이것은 특정 페이지를 접근하지 않은 프로세스가 일련의 동기화들이 연속될 때, 그 사이에 그 페이지를 접근하지 않은 프로세스가 존재하면 그 페이지에 대한 복원 프로토콜의 적용이 중단되는 현상을 말한다. 아래 그림 2의 예를 통하여 이를 설명한다. 4개의 프로세스 중 p_3를 제외한 나머지 프로세스들은 이전 동기화 구간에서 이미 페이지 x를 접근한 상태이다. 따라서 각 프로세스가 갖고 있는 x에 대한 복사집합은 그림과 같다. 이때, p_1이 쓰기접근 wf_1을 발생한 후 p_2에게 락을 전송한다면 p_1의 복사집합에 p_2가 존재함으로 해당 페이지에 대한 차이본을 락과 함께 전달한다. 이 후에 p_2가 p_3에게 락을 전

달할 때, p_2의 복사집합에 p_3가 없으므로 쓰기통보(notice(p_1))를 전달한다.

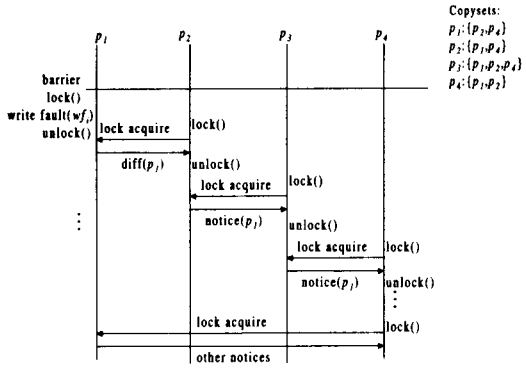

그림 2 차이본 전송의 단절현상의 예

그리고 다시 p_3가 p_4에게 락을 전달 할 때는, p_3의 복사집합에 p_4가 있으므로 차이본(diff(p_1))을 보내야 하지만 가지고 있지 않으므로 쓰기 통보만 전달한다. 이후에 p_4가 p_1과 동기화 할지라도, p_1는 부분 순서화에 따라 그 쓰기접근 wf_i을 포함한 구간보다 이후에 발생한 구간의 쓰기통보나 차이본을 p_4에게 전달한다. 그 결과로 복원 프로토콜 적용으로 인한 성능향상을 감소시킨다.

4. 제안한 향상된 늦은 하이브리드(Improved Lazy Hybrid, ILH) 프로토콜

4.1 프로토콜

본 논문에서 제안하는 향상된 늦은 하이브리드 프로토콜(improved lazy hybrid, ILH)은 차이본들이 락 허용 메시지, 배리어 도착메시지 혹은 배리어 출발메시지 등에 붙여 전달될 수 있다는 점을 제외하고는 다중쓰기를 사용하는 LI 프로토콜과 동일하다. 또한, 3.1에서 설명한 것처럼 LH와 같은 방법으로 다른 프로세스들의 접근기록을 추적하기 위해 복사집합을 유지한다.

ILH는 LI 프로토콜을 기본 프로토콜로 사용하고 있으므로 동기화가 발생하는 시점에서만 복원 프로토콜이 적용된다. 따라서 아래에서 락 동기화와 배리어 동기화로 나누어 본 프로토콜을 설명한다.

4.1.1 락 동기화

프로세스 p가 프로세스 q에게 락 허용 메시지를 보낼 때, 부분순서에 의해 보내야할 구간(interval)들의 집합, 전송집합(send_set)을 구한다. 이 전송집합에 속한 구간들 내의 쓰기통보들을 모아 락 허용 메시지에 포함시킨다. 이 때, 쓰기통보와 연관된 페이지 x의 복사집합에 q가 속해있거나 복사집합의 크기가 α를 넘는다면, p가 가지고 있거나 생성할 수 있는 x의 차이본도 메시지에 포함시킨다. 만약 이렇게 구해진 메시지의 양이 최대 메시지 길이를 초과한다면, 락 허용메시지에 쓰기통보들을 우선적으로 최대한 실어보내고 이후에 남은 차이본들을 별도의 메시지를 통해 보낸다. 이때 이 별도의 메시지들은 잃어버리더라도 일관성 유지에는 문제가 없으므로 수신측이 받았다는 메시지를 송신측에 보낼 필요는 없다.

4.1.2 배리어 동기화

락 동기화와 같이 배리어에 도착했을 때 각 프로세스 p_i들은 배리어 마스터 프로세스 p_m에게 보낼 전송집합을 구한다. 이 집합에 속한 쓰기통보들을 배리어 도착 메시지에 포함시킨다. 이때, 그 쓰기통보와 연관된 페이지 x의 복사집합에 자신 외에 하나이상의 프로세스가 있으면 페이지 x에 대해 자신이 생성했거나 생성가능한 차이본도 메시지에 포함시킨다. 이것을 받은 마스터 프로세스 p_m은 이들을 통합한 후, 다시 각 프로세스 p_i에 대해 락 동기화에서와 같이 쓰기통보들과 차이본들을 구하여 배리어 출발메시지에 실어보낸다.

락 동기화에서처럼 배리어 도착 메시지나 배리어 출발 메시지는 여러 개의 메시지로 전송될 수 있는데, 첫 번째 전송되는 메시지를 제외한 메시지는 수신측이 송신측에 확인메시지를 보낼 필요 없다.

4.2 프로토콜의 상충요소(Tradeoff)

본 논문에서 제시하는 하이브리드 프로토콜은 LH를 바탕으로 한 프로토콜이다. 이 프로토콜에서도 역시 동기를 이루는 다른 프로세스 p에게 보내야할 구간에 속한 페이지 x에 대한 쓰기통보에 대해서, p가 x의 복사집합에 속해 있다면 차이본을 동기화 메시지에 실어 보낸다. 그러나, p가 복사집합에 속해 있지 않더라도, 해당 복사집합에 속한 프로세스들의 수가 α개이라면 무조건 보낸다. 그 이유는 다음과 같다.

(1) 현재의 복사집합의 부정확성으로 인해 프로세스 p가 실질적으로는 그 페이지를 접근했지만 복사집합에는 포함되어 있지 않은 경우가 있다.

(2) p가 과거에 접근하지 않았다 하더라도 α개 이상 여러 프로세스들이 이미 접근했다면 그 프로세스도 미래에 접근할 가능성이 높다.

(3) 이렇게 전달된 차이본이 p에는 적용(apply)되지 않더라도 이후에 동기를 이루는 다른 프로세스에게 전달해 줄 수 있어, 차이본의 단절 현상을 줄일 수 있다.

이렇게 전달된 차이본이 적중한다면, 즉 이 후에 그

페이지를 해당 프로세스가 접근한다면 페이지 실패에 발생되는 원격 차이본 요구 메시지(remote diff request)를 상당부분 줄일 수 있으므로 수행속도를 향상시킬 수 있을 것이다. 그러나 이렇게 전달된 차이본이 적용되지도 않고, 전달도 되지 않는다면 이것은 불필요한 복원 프로토콜의 적용으로 인한 차이본 계산량과 통신량 증가를 초래할 것이다.

여기서 α는 전체 프로세스 수보다 작은 양수 값이다. 이 파라미터 α은 프로토콜의 성능에 많은 영향을 미친다. 응용 프로그램들의 공유 메모리 접근 패턴이 서로 다르기 때문에 최적의 α값은 응용 프로그램마다 서로 다르다. 6장에서 실험을 통해 α의 서로 다른 값이 성능에 미치는 영향을 논의할 것이다.

5. 구현 및 성능평가

앞에서 설명한 LI 프로토콜과 두 가지의 하이브리드 프로토콜을 CVM[13] DSM시스템상에서 구현하고 100Mbps의 fast Ethernet으로 연결된 8대의 Sun ultra_1을 사용하였다. 이들은 모두 Sunos 5.6을 탑재하고 있다. 다음 표1은 응용 프로그램들의 입력값과 수행 패턴을 요약한다. 배리어 수행횟수와 락 수행횟수는 본 실험환경에서 수행한 결과값들이다. 락 수행횟수는 락 **요구**(acquire)의 총 횟수가 아니라, 락 **요구**메시지를 다른 프로세스에게 보낸 횟수이다.

표 1 실험에 사용된 응용 프로그램들

프로그램	입력 데이타	배리어	락
tsp	19 cities	0	698
water-Nsqaured	512 molecules	12	700
Water-Spatial	512 molecules	9	212
Raytrace	teapot	0	46746
Cholesky	tk29.O	3	5803
barnes	4096 particle	8	45493

본 장에서는 LI 프로토콜, LH 프로토콜 그리고 ILH 프로토콜에 대해 각 응용프로그램들을 수행하여 성능을 분석한다. ILH는 α=6으로 해서 수행하였다. 제시된 실험결과들은 각 응용프로그램마다 100번을 수행한 후, 수행시간이 가장 짧은 5%와 가장 느린 5%를 버리고 나머지 결과들을 고려한다.

그림 3는 전체 응용 프로그램의 수행속도를 3가지 프로토콜별로 비교하고 있다. 그림에서 보듯이 ILH는 LH에 비해 최대 5%의 성능향상 보이고 있고 성능이 감소하는 것은 없다. TSP와 Raytrace이 5%, Water-spatial이 4%, Barnes가 2%의 성능향상을 보이고 Water-Nsquare와 Cholesky는 성능향상을 보이지 않는다. 표 2는 성능에 영향을 미치는 중요한 요소들에 대한 데이타들을 요약, 비교하고 있다. Total_messages는 총 전송된 메시지 수이고, Data는 총 전송된 데이타 량이다. Diff_request는 페이지 실패가 일어났을 때 다른 프로세스에게 차이본을 요구하는 메시지를 보낸 수로써 페이지 실패 비용에 가장 중요한 영향을 미친다. 그리고 Diffs_created는 총 차이본을 생성한 수를 의미한다.

그림 5는 각 응용 프로그램의 공유 페이지에 대한 접근 패턴을 추적한 결과로써 다음에 나오는 각 응용 프로그램의 성능분석에 이용된다. 그래프 내에서 막대그래프는 접근한 프로세스 수에 따른 페이지 수를 나타내고 있고(왼쪽 세로축) 직선그래프는 그 페이지들에 대한 페이지 실패횟수의 총합을 나타낸다(오른쪽 세로축).

이 절의 나머지 부분은 각 응용 프로그램의 접근 패턴을 설명하고 이와 관련하여 성능을 분석한다.

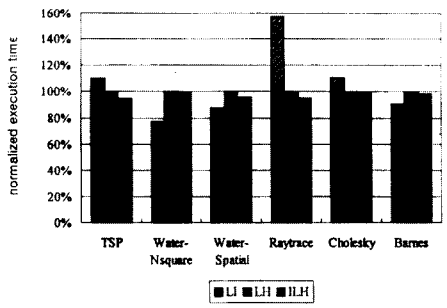

그림 3 LI, LH 그리고 ILH의 수행시간 비교

표 2 LI, LH 그리고 ILH의 실험결과 요약

Program	Protocol	Run time(s)	Total messages	Data(Kbyte)	Diff request	Diffs created
TSP	LI	25.18	7657	6884	6102	3208
	LH	22.89	2806	6119	1263	3200
	ILH	21.67	2632	6100	891	3203
Water-Nsquare	LI	13.08	4649	13991	2984	2012
	LH	17.01	3831	18917	2080	2301
	ILH	16.99	3758	19110	2050	2304
Water-Spatial	LI	22.76	6614	24703	5469	2673
	LH	25.86	4910	26860	3623	2742
	ILH	24.86	4895	26888	3457	2743
Raytrace	LI	165	161007	58576	63525	62412
	LH	104.72	89470	46919	6936	68871
	LH	99.69	86815	46787	5025	68856
Cholesky	LI	105.99	56904	122276	26611	21673
	LH	95.85	23870	117706	5824	22234
	ILH	95.55	23677	117380	5756	22076
Barnes	LI	49.7	108971	40625	28553	23856
	LH	54.82	88515	45156	9366	43908
	ILH	53.84	85482	45369	6590	45057

TSP(Traveling Salesman Problem): 대부분의 공유메모리를 차지하는 중앙 큐에 대해서 모든 프로세스가 접근하므로 각 공유페이지에 대해 모든 프로세스들이 불규칙적으로 접근하므로 그림 5 에서 보듯이 대

부분의 페이지가 모든 프로세스에 의해 접근되어진다. 따라서 ILH가 LH보다 페이지들을 좀 더 빨리 복원의 대상으로 인식하므로 Diff_request을 약 29%정도 감소시킨다. Diff_request의 수가 감소됨에 따라 Total_messages 또한 감소된다. 본 문제의 경우 복원 프로토콜의 적용이 거의 완벽하게 적중하므로, 즉 불필요한 차이본 전송이 일어나지 않으므로 Diffs_created는 ILH가 LH에 비해 증가하지 않는다. 결과적으로 수행시간은 ILH가 LH에 비해 약 5%정도 향상되었다.

Water-Nsquared: LH의 경우 LI에 비해 Diff_request가 감소되지만 Diff_created와 Data가 늘어나 전체수행 속도는 향상되지 않는다. 또한, 매우 많은 수의 동기화가 발생하는데, 각 동기화의 지연시간이 LH가 크므로 이 또한 수행속도에 악영향을 미친다. 한편, 페이지 접근 패턴을 보면 α(=6)를 넘는 페이지 수와 그들에 대한 접근 빈도가 다른 응용프로그램들에 비해 낮음을 볼수 있다. 이로 인해 ILH는 LH에 비해 Diff_request수를 그다지 감소시키지 못하고 그 외의 요소들도 비슷한 결과를 보임으로 인해 전체 성능은 향상을 보이지 못한다.

Water-Spatial: 페이지 접근 빈도가 α를 넘는 접근의 수가 대부분을 차지하지만 락 동기화에 비해 배리어 동기화의 수가 비교적 많으므로 복사집합의 오류가 적다. 따라서, LH에 비해 약 5%정도의 Diff_request수를 감소시킨다. 그러나 복원 프로토콜의 높은 적중률로 인해 전체 성능은 4%정도 향상되었다.

Raytrace: 원본 이미지가 차지하는 페이지들은 대부분 소수의 프로세스들만이 접근하고, α개 이상의 프로세스가 접근하는 페이지는 일부에 지나지 않지만 이들에 대해 페이지 실패 빈도가 대해 집중되어 있으므로 ILH로 인해 성능향상을 얻을 수 있다. Diff_request는 LH보다 약 28%, Totol_messages는 약 3% 감소하였다. 나머지 부정적인 요소는 나타나지 않고 전체 수행시간은 약 5%정도 감소하였다.

Cholesky: 한 페이지에 소수의 열 블록들이 할당되므로 대부분의 페이지들의 접근 프로세스 수는 α를 넘지 않는다. 또 다른 중요 자료구조로써, 부하균등을 위해 분산 태스크 큐를 사용하는데 이것이 할당된 페이지들에 대해서는 접근 프로세스 수가 α가 넘는다. 그러나, 이들에 대한 접근빈도는 매우 높으나 페이지 수가 그리 많지 않아 ILH가 LH에 비해 Diff_request를 거의 감소시키지 못한다. 따라서 Cholesky는 락 동기화 위주의 프로그램이지만 ILH를 적용해서 성능향상을 얻지 못한다.

Barnes: 이 프로그램은 크게 힘계산 부분과 트리 구축 부분으로 나눌 수 있다. 힘계산 부분은 상당한 규칙성과 국부성을 보이지만 트리 구축 부분은 많은 동기화와 이주적 패턴을 보인다. 따라서, 트리 구축 부분으로 인해 페이지 접근 빈도가 α를 넘는 페이지의 수가 전체 유효 페이지들의 80%이상을 차지하고 그들의 페이지에 대한 실패의 수는 전체의 96%에 달한다. 또한 배리어에 비해 락 동기화가 많으므로 복사집합의 오류가 크게 나타난다. 따라서, ILH를 적용하면 Diff_request수가 LH보다 30%나 감소시킬 수 있다. 그러나 동기화 지연시간의 증가와 부하 불균형으로 인해 성능향상은 그리 높지 않다.

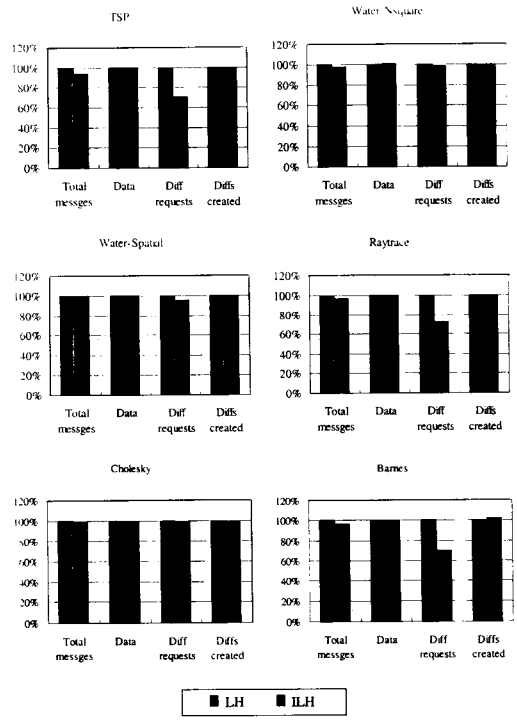

그림 4 페이지 접근 패턴 분석.

6. 파라미터 α 의 결정

동적 복원 프로토콜인 LH 프로토콜은 무효화 프로토콜인 LI과 비교해 보면, 차이본 요구 메시지의 감소로 인해 페이지 실패 시점에 걸리는 시간이 감소하고 전체 메시지 수가 줄어들어 각 프로세스가 통신에 드는 부가적인 시간을 줄일수 있다. 반면 차이본 생성량과 메시지 전송량이 많아지므로 이들에 대한 처리시간이 증가하게

되고, 다량의 차이본 전송이 동기화 시점에 발생하게 되므로 동기화 지연시간이 증가하게 된다.

이런 상관관계는 ILH 프로토콜에서 사용되는 파라미터 α의 값의 변화에 대해 꼭 같이 나타난다. 즉 α의 값을 감소할 수록 페이지 실패 비용을 상당히 줄일 수 있으나, 위의 부정적인 요소가 증가하게 됨을 예측할 수 있다.

본 연구에서는 여러 응용 프로그램에서 비교적 적절히 수행되는 파라미터 α값을 얻기 위해 모든 응용프로그램에 대해 α값을 3에서 6까지 변화시켜가며 실험해 보았다. 그 결과 TSP를 제외한 모든 프로그램들이 α=6에서 가장 우수한 결과를 얻을 수 있었다.[1)]

그림 6와 그림 7은 raytrace에 대해 α의 값을 3에서 6까지 변화시켜가며 얻은 결과이다. 그림 7에서 α가 작을수록 Diff_request가 급격히 감소하고 Total_messages의 수도 상당히 감소된다. 그에 반해 Data이나 Diff_created의 증가량은 비교적 적다. 이것은 이 응용 프로그램에서 동적 복원의 적중률이 상당히 높다는 것을 말한다. 그럼에도 불구하고 그림 6에서 수행시간을 보면 α=6일 때 전체 수행시간이 가장 우수한 이유는 α가 작을수록 동기화 시간이 증가한다는 점이다. 이렇게 증가된 동기화 지연 시간은 다량의 동적 복원으로 인한 페이지 실패를 처리하는데 걸리는 시간보다 크게 된다.

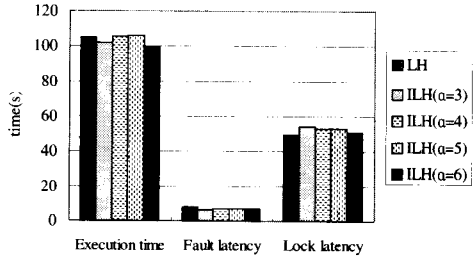

그림 6 α값 변화에 대한 수행시간, 페이지 실패 지연시간과 락 지연시간들의 비교

TSP를 제외한 다른 응용프로그램의 실험에서도 raytrace와 비슷한 양상을 보인다. TSP는 α값이 감소할수록 좋은 성능을 보이는데 그 이유는 동적 복원의 적중률이 거의 100%에 가까워 차이본생성과 메시지의 증가가 전혀 나타나지 않기 때문이다. 또한 전체 공유 페이지 수가 다른 응용 프로그램에 비해 많지 않아 동기화 지연시간의 증가가 미미한 것도 또 다른 이유이다.

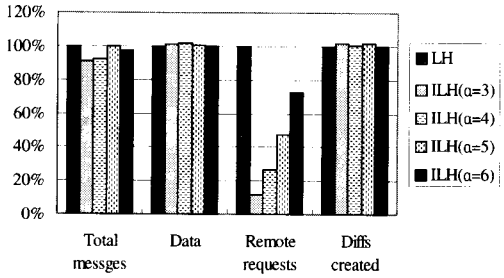

그림 7 α값 변화에 대한 주요 성능 결정요소들의 비교

7. 결 론

본 논문에서는 기존의 동적 복원 프로토콜인 늦은 하이브리드 프로토콜(LH)의 두 가지 문제점을 밝혀내고 이에 대응하여 다중접근 페이지들에 대한 복원을 강조하는 동적복원 프로토콜(ILH)을 제안하였다. 제안된 프로토콜의 성능평가를 위해, 소프트웨어 분산 공유 메모리 시스템인 CVM에 기존 프로토콜과 함께 구현하고 100Mbps에 연결된 8대의 Sun Ultra_1상에서 6개의 응용프로그램을 실행해 보았다. 그 결과, 파라미터 α=6으로 한 실험결과를 보면 제안한 프로토콜이 LH에 비해 6개의 응용프로그램 중 4개의 응용프로그램에서 수행시간을 2%-5%정도 향상시켰고 나머지 2개의 응용 프로그램은 차이가 없었다. 향상된 응용 프로그램들에서 페이지 실패에서 발생하는 차이본 요구 메시지의 수를 최대 30%까지 감소시킬 수 있었고 부정적 요소인 전송 메시지량의 증가나 차이본 생성수의 증가 등은 미미하게 나타났다. 그 이유는 이들 응용 프로그램들은 α개 이상의 프로세스들이 접근한 공유 페이지가 전체 공유 페이지의 절반이상을 차지하고 있고 이들에 대한 접근의 비율이 전체 접근의 대부분을 차지하고 있었다. 그럼에도 불구하고 이렇게 수행시간의 향상이 적은 이유는 제안한 프로토콜이 LH에 비해 동기화 지연시간을 증가시키기 때문임을 알 수 있었다.

제안한 프로토콜과 LH에서는 메시지 증가를 막기 위해 동기화 메시지에 복원정보를 같이 실어보낸다. 따라서 복원량을 증가시키게 되면 이 복원정보의 적중률에 높다하더라도 동기화 지연시간이 길어짐으로 인해 전체 수행시간의 향상은 제한되게 된다. 앞으로의 연구에서는 동적 복원 프로토콜을 적용할 때, 동기화 메시지에 실어보내는 복원정보의 량을 제한시켜 동기화 지연시간의 증가를 막고 이후 다른 메시지에 실어보내는 방안을 개발해야 한다.

1) α=7 은 모든 응용 프로그램에 대해 ILH가 LH와 거의 유사한 실험결과를 보이므로 실험에서 제외하였다.

참 고 문 헌

[1] M. Snir et al., *MPI: The Complete Reference*, The MIT Press, 1996.

[2] V.S. Sunderam, "PVM: A framework for parallel distributed computing," *Concurrency: Practice and experience*. Vol 2(4), pp 315-339, 1990.

[3] J. Protic, M. Tomasevic, and V. Milutinovic, "Distributed Shared Memory: Concepts and Systems," *IEEE Parallel & Distributed Technology*, pp. 63-79, Summer 1996.

[4] K. Li and P. Hudak, "Memory coherence in shared virtual memory systems," *ACM Transaction of Computer Systems* 7(4), 321-359, Nov 1989.

[5] A. S. Tanenbaum, *Distributed Operating Systems*. Prentice-Hall, pp. 289-375, 1995.

[6] P. Keleher et al., "TreadMarks: Shared Memory Computing on Networks of Workstations," *IEEE Computer*, pp. 18-28, Feb. 1996.

[7] K. Gharachooloo, D. Lenoski, J. Laudon, P. Gibbons, A. Gupta and J. Hennessy, "Memory consistency and event ordering in scalable shared-memory multiprocessors," *Proceedings of the 17th Annual Interbational Symposium on Computer Architecture*, pp. 15-26, May 1990.

[8] J. B. Carter, J. K. Bennett, and W. Zwaenepoel, "Implementation and Performance of Munin," *Proc. the 13th ACM Symposium on Operating Systems Principles*, pp. 152-164, Oct. 1991.

[9] J. B. Carter, J. K. Bennett, and W. Zwaenepoel, "Techniques for Reducing Consistency-Related Information in Distributed Shared Memory Systems," *ACM Trans. Computer Systems*, Vol. 13, No. 3, pp. 205-243, October 1995.

[10] P. Keleher, "Distributed Shared Memory Using Lazy Release Consistency," PhD dissertation, Rice University, Tech. Report Rice Comp-TR-240, ftp cs.rice.edu under public/TreadMarks/papers, 1994.

[11] P. Keleher, A. L. Cox, S. Dwarkadas, and W. Zwaenepoel, "An evaluation of software based release consistent protocols," *Journal of Parallel and Distributed Computing*, Vol 29, pp. 126-141, October 1995.

[12] C. Amza, A. L. Cox, S. Dawrkadas, L. Jin, K. Rajamani, W. Zwaenepoel, "Adaptive Protocols for Softwater Distributed Shared Memory," *Proceedings of IEEE Special Issue on Distributed Shared Memory*, pp. 467-475, March 1999.

[13] P. Keleher, "CVM:The Coherent Virtual Machine," http://www.cs.umd.edu/projects/ cvm, November 1996.

[14] S. Adve and M. Hill, "Weak ordering: A new definition," *Proceedings of 17th Annual International Symposium on Computer Architecture*, pp. 2-14, May 1990

이 성 우

1994년 경북대학교 컴퓨터공학과 졸업(공학사). 1996년 경북대학교 컴퓨터공학과 졸업(공학석사). 1996년 ~ 현재 경북대학교 컴퓨터공학과 박사과정. 관심분야는 어레이 프로세서 설계, 병렬 및 분산처리, 객체지향 프로그래밍 등.

김 현 철

1995년 경일대학교 컴퓨터공학과 졸업(공학사). 1997년 경북대학교 컴퓨터공학과 졸업(공학석사). 1997년 ~ 현재 경북대학교 컴퓨터공학과 박사과정. 관심분야는 어레이 프로세서 설계, 병렬 및 분산처리, 암호화 등.

유 기 영

1976년 경북대학교 수학교육학과 졸업(이학사). 1978년 한국과학기술원 전산학과 졸업(공학석사). 1992년 미국 Rensselaer Polytechnic Insitute 졸업(이학박사). 1978년 ~ 현재 경북대학교 컴퓨터공학과에 재직. 관심분야는 병렬처리, DSP array porcessor 설계, 병렬 컴파일러 등임.

하 금 숙

1983년 2월 경북대학교 전자공학과(전산전공) 학사. 1990년 2월 경북대학교 컴퓨터공학과 석사. 1997년 2월 경북대학교 컴퓨터공학과 박사과정 수료. 1986년 5월 ~ 1992년 2월 경북대학교 전자공학과 조교. 1992년 2월 ~ 현재 구미1대학 컴퓨터정보전공 부교수. 관심분야는 Race Detection, DSM, 운영체제.