DSM시스템에서 통신 부하의 가중치를 고려한 경쟁적인 갱신 프로토콜

임 성 화[†] · 백 상 현[†] · 김 재 훈[††] · 김 성 수[††]

요 약

분산 공유 메모리(Distributed Shared Memory)시스템은 사용자에게 간단한 공유메모리 개념을 제공하기 때문에 노드 사이의 데이터 이동에 관여할 필요가 없다. 각 노드는 프로세서, 메모리, 그리고 네트워크 연결장치 등으로 이루어져 있다. 메모리는 페이지 단위로 구분되며 페이지는 여러 노드에 복제본을 소유할 수 있다. 이들간 일치성을 유지하기 위하여 무효화 방식(invalidate protocol)과 갱신 방식(update protocol)이 전통적으로 많이 사용되었다. 이 두 가지 프로토콜의 성능은 시스템 변수 또는 응용 프로그램의 공유 메모리 사용 형태에 따라 좌우된다. 메모리 사용 형태에 적응하기 위하여 경쟁적 갱신(competitive update) 프로토콜은 가까운 장래에 사용되어질 복제본을 갱신시키는 반면, 다른 복제본은 무효화 시킨다. 본 논문에서는 노드 사이의 통신 비용이 동일하지 않은 구조를 감안한 가중치를 고려한*(weighted)* 경쟁적 갱신 프로토콜을 제안하였다. 시뮬레이션에 의한 성능 측정 결과 가중치를 고려한 경쟁적 갱신 프로토콜의 성능 향상을 보였다.

Weighted Competitive Update Protocol for DSM Systems

Sung-Hwa Lim[†] · Sang-Hyun Baek[†] · Jai-Hoon Kim[††] · Sungsoo Kim[††]

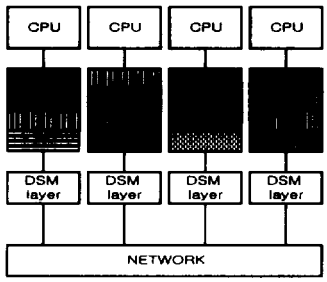
ABSTRACT

Since DSM provides a user a simple shared memory abstraction, the user does not have to be concerned with data movement between hosts. Each node in DSM systems has processor, memory, and connection to a network. Memory is divided into pages, and a page can have multiple copies in different nodes. To maintain data consistency between nodes, two conventional protocols are used: write-update protocol and invalidate protocol. The performance of these protocols depends on the system parameters and the memory access patterns. For adapting to memory access patterns, *competitive update protocol* updates those copies of a page that are expected to be used in the near future, while selectively invalidating other copies. We present *weighted* competitive update protocols that consider different communication bandwidth for each connection of two nodes. Test results by simulation show that the weighted competitive update protocol improves performance.

1. 서 론

분산 시스템의 주된 목적은 여러 개의 독립적인 시스템들을 결합하여 시스템 전체의 성능을 점진적으로 높이고, 자원을 공유하며, 시스템간의 통신 수단을 효과적으로 이용하는 것이다. 사용자의 입장에서 볼 때 복수개의 독립된 시스템을 한 개의 커다란 시스템처럼 느끼며 이용하는 것이 분산 시스템의 궁극적인 목적이다. 이를 제공하기 위한 한 방편으로 분산 공유 메모리(DSM: Distributed Shared Memory)시스템이 개발되었다[12].

※ 본 연구는 아주대학교 1998년도 정착연구비 지원으로 진행되었음.
† 비 회 원 : 아주대학교 대학원 컴퓨터공학과
†† 정 회 원 : 아주대학교 정보및컴퓨터공학부 교수
논문접수 : 1999년 4월 1일, 심사완료 : 1999년 7월 23일



(그림 1) DSM내에서의 노드간 **블록** 복제

(그림 1)은 DSM의 구조를 나타낸다. 분산 공유 메모리 시스템을 이루는 각 노드는 프로세서와 지역 메모리로 이루어지며 각 노드는 네트워크로 연결된다. 메모리는 다수개의 블록(또는 페이지)으로 분할되며 각 블록은 복수개의 복제를 허용한다. (그림 1)에서 같은 모양으로 표현된 메모리 영역은 여러 노드에 복제된 동일 블록을 의미한다. 이렇게 복제를 허용할 경우 어떤 노드에서 복제된 블록에 대해 쓰기(write) 동작이 일어날 경우, 다른 복제되었던 블록들과의 일치성을 유지해 주어야 한다. 이를 위한 전통적인 일치성 유지 방식에는 무효화 프로토콜(invalidate protocol)과 갱신 프로토콜 (update protocol)이 있다[12].

- 무효화 프로토콜 (invalidate protocol): 어떤 노드에서 쓰기 동작이 일어날 경우 다른 노드들에 복제된 블록들을 모두 무효화시킨다. 따라서, 쓰기 이후에는 단일 복제만 존재하게 된다.
- 갱신 프로토콜 (update protocol): 어떤 노드에서 쓰기 동작이 일어날 경우 다른 노드들에 복제되어있는 모든 블록들을 쓰기를 한 내용과 동일하게 갱신시킨다. 쓰기 이후에도 복수개의 복제가 존재할 수 있다.

이러한 일치성 유지를 위한 무효화 프로토콜 또는 갱신 프로토콜을 일관적으로 적용할 경우 분산 환경의 다양함과 응용 프로그램의 공유 메모리에 대한 이용 형태를 반영하기가 어렵다. 따라서 일치성 유지방식은 프로그램 수행 도중에 보다 효율적인 프로토콜을 선택적으로 적용하는 방법을 필요로 하게 된다[1,3,5,6,7,8,9,10]. 프로그램 수행 도중에 효율적인 프로토콜을 선택적으로 적용하는 적응성을 가진 프로토콜로써 제안된 경쟁적인 갱신 프로토콜(Competitive update protocol)[6]은 복제된 블록에 대한 사용 형태를 반영한다. 즉, 지역 사용(local access)이 상대적으로 자주 일어나는 노드의 복제는 다른 노드로부터의 갱신을 받아 들여 지역에 복제를 보유하게 하여 페이지 오류(page fault)를 줄임으로써 빠른 응답성을 가지게 한다. 반면 지역 사용(local access)이 상대적으로 자주 일어나지 않는 블록은 무효화 시키어 다른 노드로 부터의 갱신에 따르는 통신 부하를 줄임으로써 결과적으로 전체 시스템의 성능을 높이기 위한 프로토콜이다.

본 논문에서 제시하는 가중치를 고려한 경쟁적인 갱신 프로토콜은 프로그램 수행 도중에 경쟁적인 갱신 프로토콜과 같이 복제된 블록에 대한 사용 형태를 반영하는 것은 물론 노드간에 서로 다른 통신 비용을 갖는 네트워크 환경을 동시에 반영한다. 프로그래머의 관여 없이 복제된 블록에 대한 사용 형태에 따라 무효화 프로토콜 또는 갱신 프로토콜 중 효율적인 프로토콜을 적용하고 또한 통신 환경을 반영하여 통신 비용을 줄일 수 있는 방향으로 프로토콜이 선택된다. 이는 기본 프로토콜을 크게 변형하지 않고, 구현이 용이하면서도 공유 데이터의 일치성 유지에 필요한 통신 오버헤드를 줄이는 것을 목적으로 하는 프로토콜이다.

2. 관련연구

DSM에서 일치성 유지를 효율적으로 하기 위한 방법들이 많이 연구 되어왔는데, 이중 사용 형태를 반영하기 위한 방법에는 복제된 블록마다 다른 노드로부터의 갱신 횟수에 한계값을 가지게 하는 경쟁적 갱신 프로토콜[6], 일정한 시간이 지나도록 사용 형태가 일어나지 않는 블록을 무효화시키는 갱신 타임아웃(update timeout) 방식[3], 실행중 연속되는 쓰기 동작의 수를 측정하여, 앞으로도 유사한 형태로 사용을 한다는 가정하에 쓰기 동작이 연속으로 일어날 것으로 예측되는 구간에서는 다른 복제된 블록들을 무효화시키는 write-run 모델의 적응적 프로토콜 (adaptive protocol on write-run model)[1], 두 지역 사용 사이에 다른 노드로부터의 갱신수를 고려하는 거리-적응 갱신 프로토콜 (distance-adaptive update protocol) 방법[10], 그리고 응용 프로그램에서 데이터들이 노드들을 옮겨가면서 처리되는 이주(migratory) 형태인 경우 옮기기 전의 노드에서 스스로-무효화(self-invalidation)함으로써 무효화 비용을 줄이는 이주성 공유 적응(adapt to migratory sharing) 방법이 있다[4]. 또, 유사한 방법으로 노드 스스로 주어

진 주건에 따라 자동으로 무효화됨으로써 무효화 하는 데 필요한 통신비용을 줄이도록 하는 가변적 스스로-무효화 (dynamic self-invalidation) 방법이 있다[7]. 특히 write-run 모델의 적응적 프로토콜[1]은 전통적인 경쟁적 갱신 프로토콜[6]을 개선했다는 점에서 본 논문과 맥락을 같이 하지만 경쟁적 갱신 프로토콜의 단점을 해결하고자 하는 접근 방법이 다르다. [6]에서는 갱신 한계값을 고정시킬 경우의 문제점을 해결하고자 가변적으로 하였고, 본 연구에서는 노드간 통신비용이 다를 경우 이를 고려하였다. 따라서 두 연구의 성능 향상을 비교하자면 어느 시스템 파라메터를 변동시키는지에 따라 다르다. (물론 두 방법을 동시에 수용하였을 때 더욱 좋은 성능 향상이 기대된다.)

프로그래머가 관여하여 응용 프로그램, 프로그램 루틴, 또는 자료구조에 따라 적합한 방법을 지정하도록 하는 방법으로써 응용 프로그램 자체의 사용 형태를 분석하여 프로그래머가 분석된 형태에 따라 적절한 방법을 적용하는 애플리케이션에 따른 프로토콜(application specific protocol)[5], 분산환경에서 통신환경에 투명하게 읽기, 쓰기를 통해 사용하되, 부분적으로 필요한 경우 성능향상을 위해 메시지 전송을 통한 송신(send), 수신(receive) 동작으로 통신을 하게 하는 명시적 통신(explicit communication) 방법[9], 프로그래머가 자료구조 특성에 따라 자료구조 별로 효과적으로 적용될 프로토콜을 선언하는 Munin의 다중 일치성 프로토콜 (multiple consistency protocol) 방법[3] 등의 연구가 있다.

3. 가중치를 고려한 경쟁적인 갱신 프로토콜

기존의 경쟁적 갱신 프로토콜은 사용 형태를 반영하기 위해 가장 최근의 지역 노드에 의한 메모리 사용 이후에 다른 노드로부터의 갱신 횟수만을 고려하여 갱신 한계값(update limit) 이내에서는 지역 복제를 갱신시키고 한계값을 넘으면 무효화 시킨다[6]. 이것은 노드간의 단위 메시지를 전송하는데 필요한 통신 비용이 동일하다는 가정에서는 올바른 방법이지만 노드간 통신비용이 다를 경우 이를 고려하기 위한 방법이 필요하다.

가중치를 고려한 경쟁적 갱신 프로토콜은 노드간 동일 메시지를 전송하는데 필요한 통신비용이 서로 다른 경우를 고려하여 노드간 통신 비용에 따라 갱신 가능 횟수를 차등하게 적용함으로써 전체적인 일치성 유지를 위한 통신 부하를 줄이도록 한다. 예를 들면, 통신환경에 따라 통신비용이 큰 노드로부터 갱신을 받을 때에는 지역 메모리는 무효화 시키고 통신비용이 적은 노드로부터 갱신을 받을 때에는 지역 메모리를 갱신하는 것이 전체 성능 향상에 유리할 수 있다.

가중치를 고려한 경쟁적 갱신 프로토콜의 적용 방법을 요약하면 다음과 같다.

- 노드사이의 메시지 전송비용은 각기 차이가 있으므로 이를 반영하기 위해 정의된 메시지 전송비용에 따라 가중치(weight)를 부여한다. 즉, 노드 i에서 노드 j로 단위 메시지를 보내기 위한 통신 비용은 C(i,j)로 주어지며 노드 i, j에 따라 C(i,j) 값은 다를 수 있다.
- 경쟁적 갱신 프로토콜에서는 지역 노드가 공유메모리를 사용할 때 마다 갱신 횟수를 0으로 초기화하고 다른 노드에 의해 갱신될 때 마다 갱신 횟수를 1씩 증가 시키게 되는데, 가중치를 고려한 경쟁적 갱신 프로토콜에서는 갱신 횟수의 증가 값을 메시지 전송 비용에 따라 부여된 가중치에 따라 다르게 증가시킨다.

경쟁적 갱신 프로토콜의 구현 방법에 따라 (1) 갱신 횟수를 초기에 0으로 하고 갱신을 받을 때 마다 증가시키어 갱신 한계값이 넘으면 무효화하거나, (2) [1]에서와 같이 카운터를 갱신 한계값으로 초기화 시킨 후 원격 노드로부터 갱신을 받을 때 마다 카운터를 감소시키어 0까지 갱신을 수용하는 방법이 있는 데 기본 개념은 동일하다.

결과적으로 공유메모리 사용 형태에 따라 효율적인 프로토콜을 적용함은 물론, 통신 환경도 고려해 갱신을 위한 메시지 전송비용이 적은 곳은 갱신 기회를 많이 주고, 전송 비용이 큰 곳은 갱신 기회를 적게 주어 일치성 유지를 위한 평균 메시지 전송 비용을 줄이게 된다. 제안된 가중치를 고려한 경쟁적 갱신 프로토콜의 장점을 요약하면 다음과 같다.

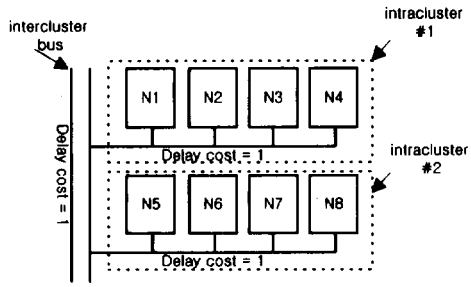
- 노드간 불균등한 통신비용을 반영하여 일치성 유지를 위한 비용을 줄인다.
- 사용자의 관여 없이 사용자에 투명하게 일치성을 유지한다.
- 기존의 무효화 프로토콜과 갱신 프로토콜을 큰 변환 없이 적용이 가능하다.

4. 성능분석 모델

가중치를 고려한 경쟁적 갱신 프로토콜의 성능을 다양한 통신 환경에서 검증하기 위해 다음과 같은 세가지 종류의 시스템 구조를 모델링하였다.

4.1 계층적 구조

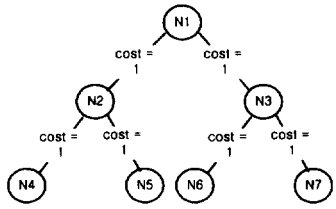
(그림 2)는 계층 버스 구조의 네트워크 환경으로써 Dash 시스템과 유사한 구조이다. 각 노드들이 "intrabus" 내에서 연결되어있고 또 이러한 "intrabus"는 "interbus"를 통해 연결되어있는 구조이다. 한 노드에서 다른 노드로 메시지를 보낼 때 한 개의 목적 노드에게만 메시지가 전달되고, 브로드캐스트 기능은 없는 것으로 가정한다. "delay cost"는 단위 메시지를 보내는 데 필요한 통신 비용을 나타낸다. 단위 메시지란 제어 신호(무효화, 응답 등)와 같이 데이터 크기가 작은 메시지를 의미한다. 이와 같은 구조에서는 같은 클러스터에 속해 있는 노드 사이에서는(intracluster) 단위 메시지를 보낼 때 1 단위의 통신 비용이 필요하고, 다른 클러스터 사이의 노드간에는(intercluster) 3 단위의 통신 비용이 필요하다. 전통적인 경쟁적 갱신 프로토콜에서는 갱신 횟수만을 고려하여 모두 같은 비용으로 처리하지만 본 논문에서 제안 하고 있는 가중치를 고려한 경쟁적인 갱신 프로토콜에서는 동일 클러스터내의 노드사이의 단위 통신 비용은 1로 하고, 다른 클러스터의 노드 사이의 통신 비용은 3으로 차등하게 계산함으로써 성능 향상을 기대할 수 있다.



(그림 2) 계층 구조

4.2 트리 구조

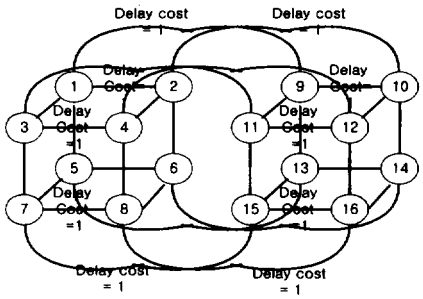
(그림 3)과 같이 최상위 노드 아래에 하위 계층의 노드들이 연결되어있고, 다시 이 노드들에 하위 계층의 노드들이 연결 되어있는 구조이다. 부모-자식간(인접한 노드사이)에는 단위 통신 비용이 1이고 그 밖의 노드 사이는 거리에 비례하여 통신 비용이 증가 한다. 예를 들어 N2와 N7사이의 단위 통신 비용은 3이 된다.



(그림 3) 트리 구조

4.3 큐빅 구조

(그림 4)와 같은 n-cube 구조이다. 인접한 노드간 단위 통신 비용은 1이고 그 밖의 노드사이의 통신 비용은 노드간 거리에 비례한다. 예를 들면 노드 1과 노드 16사이의 통신 비용은 4이다.



(그림 4) 큐빅 구조

위에서 설명한 모델을 기반으로 다음과 같은 대상 시스템을 가정하였다.

- 공유 메모리 사용에는 읽기와 쓰기가 있다.
- Copyset(복제를 보유한 노드의 집합)을 얻기 위한 비용은 고려하지 않는다. (본 논문에서 가중치를 고려한 경쟁적인 갱신 프로토콜을 단순화 하기 위해 copyset을 얻기 위한 비용을 고려하지 않았지만 이를 포함한 방식도 쉽게 구할 수 있다.)
- 노드간 통신 부하는 동일하지 않다.
- 한 노드에서 다른 노드로 메시지를 보낼 때 한 개의 목적 노드에게만 메시지가 전달되고, 브로드캐스트 기능은 없는 것으로 가정한다.
- 각 구조에서의 노드의 수, 연결환경, 통신 부하를 반영하는 가중치는 (그림 2), (그림 3), (그림 4)와 같다.

- 최근에 공유 메모리를 자주 사용한 노드는 시간적 지역성에 의해(time locality) 다음에도 자주 사용될 가능성이 크다.

대상 시스템의 환경 변수는 다음과 같다.

- 노드 번호(node number): 시뮬레이션 중에 읽기 또는 쓰기 동작이 일어날 노드의 번호를 선택하는 수로써 노드마다 정의된 공유메모리 사용 비율에 의해 랜덤하게 결정된다. 한 순간에 한 노드만 이 공유 메모리를 사용할 수 있다.
- 동작(operation): 선택된 노드가 요청할 명령의 종류로써 정의된 확률에 의해 읽기 또는 쓰기 동작이 결정된다.
- 가중치(weight): 두 노드간 단위 메시지를 전송하는데 필요한 통신 비용.
- 페이지 오류 비용 (page fault cost): 지역 노드에 블록의 복제가 없어 다른 노드로부터 블록을 전송받는데 요구되는 통신 비용.
- 갱신 비용(update cost): 복제를 가진 다른 노드의 쓰기에서 변경된 데이터를 전송 받고 이에 응답하는 데 요구되는 통신 비용.
- 무효화 비용 (invalidate cost): 무효화 신호를 전송하고 이에 대한 응답을 받는데 필요한 통신 비용.
- 갱신 횟수(update count): 경쟁적 갱신 프로토콜과 가중치를 고려한 경쟁적 갱신 프로토콜에서 사용되며, 원격노드의 쓰기에서 갱신 횟수가 정의된 한계값(갱신 한계값)을 넘지 않은 블록들은 갱신하고, 갱신 한계값을 넘은 블록들은 무효화시킨다. 이 값은 다른 노드의 쓰기에서 갱신 메시지를 받을 때마다 가중치만큼 증가한다. (경쟁적 갱신 프로토콜에서 가중치는 항상 1이다.)

갱신 한계값(update limit): 복제된 블록이 지역에서의 사용 없이 계속해서 다른 노드로부터 갱신을 받을 수 있는 갱신 횟수의 한계값이다.

갱신 한계값은 (가중치를 고려한) 경쟁적 갱신 프로토콜의 성능을 좌우하는 주요 파라메터가 된다. [8]에서 제시한 "세그먼트"당 비용 모델을 바탕으로 적절한 갱신 한계값을 고려해 보자. "세스먼트"란 한 노드에서 한 페이지를 엑세스 하는 데 필요한 통신 비용을 계산하는 기본단위가 된다. 새로운 "세그먼트"의 시작은 원격(remote) 노드가 해당 공유메모리의 페이지에 쓰기를 수행한 후 첫 번째로 지역(local) 노드가 그 페이지를 사용 (읽기 또는 쓰기)하는 시점이다. 즉, 세그먼트는 지역 노드의 메모리 사용과 그 사이의 일련의 원격 노드의 쓰기들로 이루어져있다. 세그먼트당 지역 노드의 일치성 유지를 위한 비용을 계산하면 다음과 같다. (비용 계산시 지역 노드에서 원격 노드로 갱신 메시지 전송은 원격 노드의 비용에만 포함된다.

- 무효화 프로토콜: 세그먼트가 시작될 때 페이지 오류(page fault)가 발생되어 페이지를 다른 노드로부터 전송 받는데 페이지 오류 비용이 필요하다. 또한, 원격 노드에서 첫 번째의 쓰기 발생시 한 개의 갱신 비용이 필요하지만 원격 노드의 두 번째 쓰기 발생시 지역 노드에는 복제가 없으므로 추가 메시지가 필요없다. 따라서 세그먼트당 필요한 비용은 [페이지오류 비용+갱신 비용]이다.
- 갱신프로토콜: 원격 노드에서 쓰기가 발생될 때 마다 한 개의 갱신 비용이 필요하다. 따라서, 세그먼트에서 총 U개의 갱신 메시지를 받았다면 비용은 [갱신비용*U]이다.
- 경쟁적 갱신 프로토콜: 원격 노드에서 쓰기가 발생될 때 마다 한 개의 갱신 비용이 필요하다. 따라서, 세그먼트에서 총 U개의 갱신 메시지를 받았다면 비용은 [갱신비용*U]이다. 그러나 U가 갱신 한계값을 넘으면 (한계값+1)개의 갱신 메시지를 받을 때 지역 복제가 무효화되고 나중에 페이지 오류가 발생되므로 비용은 [(갱신한계값+1)*갱신비용+페이지오류비용]이 된다.

위의 분석에서 볼 때 갱신 한계값이 너무 크면 지역의 공유메모리 사용 빈도가 낮을 때 불필요한 갱신 비용이 낭비가 되며, 갱신 한계값이 너무 작으면 지역의 공유메모리 사용이 일정 빈도 이상 일 때 너무 이른 무효화 때문에 페이지 오류 비용을 부담하게 된다. 적절한 갱신 한계값은 메모리 사용 형태(지역노드가 해당 세그먼트에서 받은 갱신 메시지의 수)에 따라 다르지만, 무효화 프로토콜과 갱신 프로토콜의 비용이 동일하게 되는 세그먼트 당 갱신 횟수 즉, [(페이지오류 비용+갱신비용)/갱신비용] 또는 이보다 약간 적은 수가 적정값이다[8].

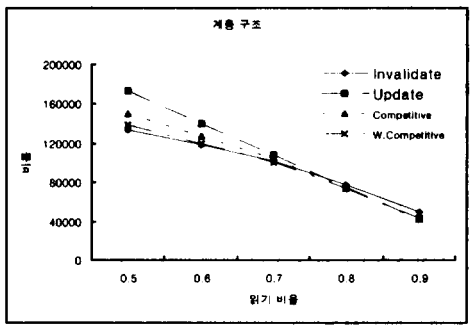
5. 성능 평가 및 분석

가중치를 고려한 경쟁적 갱신 프로토콜의 성능을 평

가하기 위하여 4장에서 설명한 모델을 시뮬레이션하고 성능을 측정하였다. 애플리케이션 프로그램 마다 서로 다른 메모리 이용 형태에 따라 DSM의 성능에 큰 차이를 보인다. 본 연구에서 성능 평가시 실제의 애플리케이션 대신 다양한 메모리 사용 형태에 따른 성능을 비교하기 위하여 읽기 비율의 변화에 따른 비용(cost)을 비교하였다. 성능은 메시지수, 데이터량, 노드간 통신 지연 시간 등이 될 수 있는데, 본 연구에서는 노드간 통신 지연 시간을 가정하였다. 참고로 8노드로 구성된 워크스테이션(Sun Sparc 4) 클러스터(10Mbps Ethernet 연결)에서 측정한 결과 제어 신호용 메시지를 보내고 응답 받는데 약 3.8msec., 페이지 오류시 페이지를 받는 데 약 30msec. 정도의 지연시간이 소요되는 것으로 측정되었다. 갱신 비용은 갱신된 데이터의 크기에 따라 3.8msec.와 20msec. 사이로 측정되었다. 이와 같은 측정값을 바탕으로 본 논문에서는 다음과 같은 시스템 환경 변수를 가정하였다.

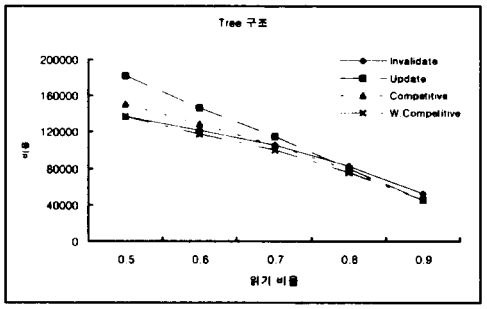
- 페이지 오류 비용: 10
- 갱신 비용: 3
- 무효화 비용: 1
- 갱신 한계값: 3 ((페이지오류비용+갱신비용)/갱신비용=13/3 보다 적은 3으로 결정)
- 총 공유메모리 사용횟수: 10000회

읽기 비율에 따른 각 프로토콜의 성능을 각 구조별로 측정하였다. (그림 5)는 계층 구조, (그림 6)은 트리 구조, 그리고 (그림 7)은 큐빅 구조의 성능 비교를 나타낸다.



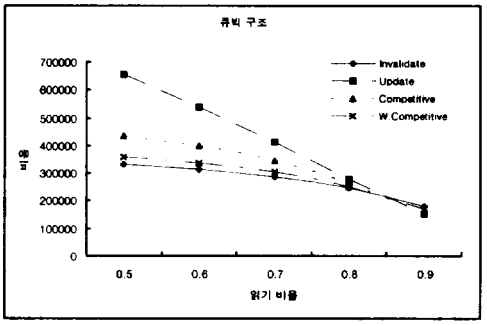
노드 개수: 8
노드 1, 2, 3, 4의 사용 비율: 각각 15%
노드 5, 6, 7, 8의 사용 비율: 각각10%

(그림 5) 계층 구조에서의 성능 비교



노드 개수: 7
노드 2, 4, 5의 사용 비율: 각각20%
노드 1, 3, 6, 7의 비율: 각각10 %

(그림 6) 트리 구조에서의 성능 비교



노드 개수: 16
노드 1,2,3,4,5,6,7,8의 사용 비율: 각각 8%
노드 9,10,11,12,13,14,15,16의 사용 비율: 각각4.5%

(그림 7) 큐빅 구조에서의 성능 비교

세가지 구조를 시뮬레이션한 결과의 공통점은 무효화 프로토콜과 갱신 프로토콜의 성능을 상대적으로 비교하면 예상했던 대로 무효화 프로토콜은 읽기 비율이 적을수록, 갱신 프로토콜은 읽기 비율이 많을수록 성능이 좋다. 또한, 경쟁적 갱신 프로토콜은 지역의 공유메모리 사용 빈도가 낮을 때 불필요한 갱신으로 인하여 무효화 프로토콜보다 성능이 떨어지기 때문에, 읽기 비율이 적을 때 무효화 프로토콜 보다 성능이 약간 떨어진다. 그러나, 경쟁적 갱신 프로토콜은 일반적으로 안정적인 성능을 보였다.

가중치를 고려한 경쟁적 갱신 프로토콜은 노드간의 통신 비용이 서로 다른 환경을 반영하므로 이를 감안하지 않는 전통적인 경쟁적 갱신 프로토콜 보다 전반적으로 우수한 성능을 보인다. 특히, 계층적 구조(그림 5)보다 노드간 통신 비용의 차이가 더욱 큰 트리 구조

(그림 6) 및 큐빅 구조(그림 7)에서 큰 성능 향상을 보였다. 기존의 경쟁적 갱신 프로토콜은 주로 버스를 기반으로 하여 노드간 통신 비용이 동일한 구조에서 유용하게 사용될 수 있었지만, 노드간 통신 구조가 복잡하고 분산된 환경에서는 적합하지 않다. 본 논문에서 제안된 가중치를 고려한 경쟁적 갱신 프로토콜은 복잡한 네트워크 또는 이종의 네트워크 및 분산 환경에 적합한, 공유메모리의 일치성을 위한 프로토콜이 될 수 있음을 시뮬레이션을 통하여 확인하였다.

6. 결론 및 향후 과제

시뮬레이션 결과에서 보듯이 노드간 통신 비용이 동일하지 않은 환경에서는 통신 비용의 차이를 감안한 가중치를 고려한 경쟁적 갱신 프로토콜이 통신 비용의 차이를 감안하지 않는 전통적인 경쟁적 갱신 프로토콜보다 우수한 성능을 나타내었다. 즉 가중치를 고려한 경쟁적 갱신 프로토콜은 경쟁적 갱신 프로토콜의 특징인, 공유메모리의 사용 형태에 따라 무효화 프로토콜 또는 갱신 프로토콜을 선택하는 적응성을 가지는 것은 물론, 노드간 서로 다른 통신 비용을 반영하는 장점을 가진다.

본 논문에서 제시한 가중치를 고려한 경쟁적 갱신 프로토콜의 성능을 실제로 검증하기 위해 자주 사용되는 응용프로그램을 여러 종류의 구조를 가진 다양한 시스템에서 구현을 통하여 실험할 필요가 있다. 또한, 프로그램 수행도중 변할 수 있는 공유메모리 사용 형태와 통신 부하를 프로그램 수행 도중에 반영하는 방법, 및 가중치 및 갱신 한계값의 최적화 문제, 그리고 고유메모리 사용 형태와 통신 부하 이외의 또 다른 시스템 특성을 반영하는 방법을 향후 연구에서 고려할 계획이다.

참 고 문 헌

[1] C. Anderson and A. Karlin, "Two adaptive hybrid cache coherency protocols," in Proc. of the International Symposium on High-Performance Computer Architecture, pp.303-313, Feb. 1996.

[2] J.B. Carter, J. K. Bennett, and W. Zwaenepoel, "Implementation and Performance of Munin," in Proc. of the 13th ACM Symp. On Operating Systems Principles (SOS P'91), pp.152-164, Oct. 1991.

[3] J. B. Carter, Efficient Distributed Shared Memory Based on Multi-Protocol Release Consistency, Ph.D. dissertation, Rice University, Sept. 1993.

[4] A. Cox and R. Fowler, "Adaptive cache coherency for detecting migratory shared data," in Proc. of the 20th Annual International Symposium on Computer Architecture, pp.98-108, May 1993.

[5] B. Falsafi, A. Lebeck, S. Reinhart, I. Schoinas, M. Hill, J. Larus, A. Rogers, and D. Wood, "Application-specific protocols for user-level shared memory," in Proc. of Supercomputing'94, pp. 308-389, Nov. 1994.

[6] H. Grahn, P.Stenstrom, and M. Dubois, "Implementaiton and evaluation of update-based cache protocols under relaxed memory consistency models," Future Generation Computer Systems, Vol.11, pp.247-271, jun. 1995.

[7] A. Lebeck and D. Wood, "Dynamic self-invalidation: reducing coherence overhead in shared-memory multiprocessors," in Proc. of the 22nd Annual International Symposium on Computer Architecture, pp. 48-59

[8] J.H. Kim and N. H. Vaidya, "A cost-comparison approach for adaptive distributed shared memory," in Proc. of 1996 ACM International Conference on Super computing, pp.44-51, May 1996.

[9] U. Ramachandran, G. Shah, A. Sivasubramaniam, A. Singla, and I. Yanasak, "Architectural mechanisms for explicit communication in shared memory multiprocessors,"in Proc. of Supercomputing'95, Dec. 1995.

[10] A. Raynaud, Z. Zhang, and J. Torrellas, "Distance-adaptive update protocols for scalable shared-memory multiprocessors," in Proc. of the International Symposium on High-Performance Computer Architecture, pp.323-334, Feb. 1996.

[11] W. Stallings, operating systems, prentice hall, 1998.

[12] M. Stumm and S. Zhou, "Algorithms imple-

menting distributed shared memory," IEEE Computer, Vol.23, pp.54-64, May 1990.

임 성 화

e-mail : holyfire@madang.ajou.ac.kr

1999년 아주대학교 정보 및 컴퓨터공학부(학사)

1999년 현재 아주대학교 컴퓨터공학과 석사과정

관심분야 : 분산시스템, 결함허용 시스템, 이동컴퓨팅 등

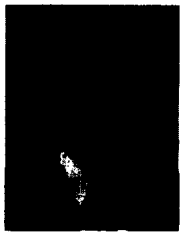
백 상 현

e-mail : swimski@cesys.ajou.ac.kr

1999년 아주대학교 정보 및 컴퓨터공학부(학사)

1999년 현재 아주대학교 컴퓨터공학과 석사과정

관심분야 : 분산시스템, 실시간시스템 등



김 재 훈

e-mail : jaikim@madang.ajou.ac.kr

1984년 서울대학교 제어계측공학과(학사)

1993년 Indiana University, Computer Science(석사)

1997년 Texas A&M University, Computer Science(공학박사)

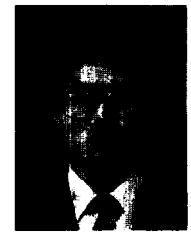
1984년~1991년 대우통신㈜ 컴퓨터연구실 대리

1995년~1997년 Texas A&M University, Graduate Research Assistant

1997년~1998년 삼성전자㈜ 컴퓨터시스템팀 수석연구원

1998년~현재 아주대학교 정보및컴퓨터공학부 조교수

관심분야 : 분산시스템, 실시간시스템



김 성 수

e-mail : sskim@madang.ajou.ac.kr

1982년 서강대학교 전자공학과(공학사)

1984년 서강대학교 전자공학과(공학석사)

1995년 Texas A&M University, 전산학과(공학박사)

1983년~1986년 삼성전자(주) 종합연구소 컴퓨터연구실(주임연구원)

1986년~1996년 삼성종합기술원 수석연구원

1991년~1992년 Texas Transportation Institute 연구원

1993년~1995년 Texas A&M University, 전산학과, T.A.

1997년~1998년 한국정보처리학회, 한국정보과학회 논문지 편집위원

1996년~현재 아주대학교 정보통신대학 정보및컴퓨터공학부 교수

관심분야 : 멀티미디어, 결함 허용, 이동 컴퓨팅, 성능평가, 시뮬레이션