다중쓰레드 프로그래밍을 위한 분산공유메모리 관리 기법

서 대 화†

요 약

본 논문에서는 대규모 다중처리기 시스템에서 다중쓰레드를 지원하는 기법에 관하여 다룬다. 분산공유메모리에서의 주소번역표 관리, 블록 일관성 유지 방법, 그리고 블록대치 정책에 대하여 쓰레드 프로그래밍 환경에 적합한 새로운 기법을 제안한다. 이 기법은 분산공유메모리에서 일반적으로 발생하는 문제점들인 거짓 공유, 불필요한 중복, 블록 바운싱, 그리고 주소 엘리어싱 등을 효율적으로 해결한다. 그리고 응용프로그램의 투명성을 제공하고, 시스템의 확장과 구현 용이하도록 해주며, 다중쓰레드 환경을 사용자에게 제공한다.

Distributed Shared Memory Scheme for Multi-thread programming

Dae-wha Seo†

ABSTRACT

In this paper, we discuss a distributed shared memory management scheme based on multi-threaded programming model for a large-scale loosely coupled multiprocessor system. The scheme covers three major issues in the distribued shared memory; the address translation table management, the block coherence maintenance, and the block placement policy. The scheme efficiently resolves the general problems occurred in the distributed shared memory such as a false sharing, an unnecessary replication, a block bouncing, and an address aliasing phenomenon. It also provides the application transparency, good scalability, easy implementation, and multithreaded programming model to users.

1. 서 론

분산 공유 메모리는 다중 처리기 시스템에서 모든 처리기 사이에 공유되는 단일 가상 주소 공간을 제공하는 것을 목적으로 한다[5]. 응용 프로그램의 관점에서 볼 때, 분산 공유 메모리는 쓰레드들이 여러개의 처리기에서 병렬로 수행된다는 점을 제외하고는 일반적인 운영체제에서의 가상 메모리와 동일하다. 시스템의 메모리 관점에서 볼 때는 기존의 가상 메모리에서 다루던 메모리와 보조 메모리인 디스크의 계층구조에 원격 메모리 계층이 추가된 형태이다. 따라서 모든 처리기는 자신의 주소 공간에 속하는 어떠한 메모리 주소도 직접 접근할 수 있다. 처리기마다 존재하는 분산 공유 메모리 관리자가 지역 메모리들과 공유된 가상 주소 공간 사이를 사상한다. 이 관리자는 내부적으로는 자료를 지역 메모리로 이주(migration) 또는 중복(replication)시켜 사상할 수 있으며, 공유된 주소 공간의 일관성을 항상 유지한다.

이러한 분산 공유 메모리 기법은 분산 시스템에서의 메시지 전달 방식에 비해서 다음과 같은 장점들을 가진다[7]. 첫째, 병렬 프로그래밍을 쉽게 할 수 있고

† 정 회 원:경북대학교 전자·전기공학부 조교수
논문접수:1996년 2월 8일, 심사완료:1996년 5월 15일

프로그램의 이식성도 높다. 둘째, 리스트와 같은 복잡한 자료구조의 전달이 용이하다. 즉 공통의 공유 주소 공간에서 메모리 읽기 또는 쓰기로서 프로세스간 통신이 이루어지므로 단순히 자료 구조의 포인터만 전달하면 된다. 셋째, 프로세스의 이주시 주소 번역표를 비롯한 기본적인 컨텍스트만 옮기면 되므로 프로세스 이주의 부담이 적다.

하지만 분산 공유 메모리 시스템에서는 원격 메모리의 접근 지연 시간을 줄이기 위하여 처리기마다 필요한 자료를 자신의 지역 메모리로 이주 또는 중복시켜 사용하므로 효율적으로 공유 자료의 일관성을 유지하는 방법이 필요하다. 이를 위하여 먼저 거짓 공유를 최소화해야 하고[1], 둘째 불필요한 중복과 바운싱 현상을 최소화할 필요가 있다[3]. 마지막으로 주소 앨리어싱을 지원해야 한다. 즉 각각 다른 주소 공간을 갖는 프로세스들이 공유하고 있는 동일한 물리적 메모리 부분을 서로 다른 가상 주소로서 사상이 가능해야 한다.

그래서 본 논문에서는 분산 공유 메모리의 특성을 보장하면서 효율적인 자료의 일관성을 유지 시킬 수 있는 다중 쓰레드 프로그래밍 모델 기반의 분산공유 메모리 기법(Distributed Shared Memory Scheme based on Multi-threaded Programming Model : DSMP)을 제안한다. DSMP에서의 주소 번역표는 4 단계의 계층 구조로 되어 있으며, 동적 메모리 할당이 가능하여 일반적인 상용의 처리기 및 운영체제에서 구현이 용이하도록 하였다. 그리고 주소 앨리어싱을 지원하며, 페이지 또는 블록이 동적으로 사용되며, 공유의 기본 단위를 블록으로 하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 분산 공유 메모리 기법의 관련 연구들을 살펴본다. 3장에서는 본 논문에서 제안하고 있는 효율적인 분산 공유 메모리 관리 기법에 대한 설계의 기본 방향, 주소 번역표의 구조 및 관리, 블록 일관성 관리, 그리고 동적 배치 정책에 대하여 기술한다. 4장에서는 기존 연구에 비하여 본 연구에서 개선한 사항들에 대한 비교 평가를 한다. 마지막 결론에서 본 연구의 성과 및 앞으로 더 연구되어야 할 내용을 살펴본다.

2. 관련 연구

분산 공유 메모리 관리 방식은 Kai Li에 의하여 처음 제안된 후 많은 연구가 이루어지고 있다[4, 5, 6, 7]. 특히 주소 변역표의 관리, 공유자료의 일관성 유지, 자료의 배치 정책 등이 주된 연구 분야이다. 주소 변역표 관리는 가상 주소를 물리적 주소로 사상하는 방법에 대한 것이고, 공유자료의 일관성 유지는 동일한 자료를 여러 프로세스에서 동시에 접근을 요청할 때 이들 상호간의 동기화를 담당하는 부분이다. 자료 배치 정책은 처리기 사이에 자료를 언제 어디로 이동할 것인가를 결정하는 것으로 원격 자료의 접근 시간을 최소화하고 응용프로그램에게 자료접근의 투명성을 제공하는 방법에 대한 연구이다.

2.1 주소 번역표 관리

주소 사상과 공유의 최소 단위가 페이지 또는 블록인가에 따라 주소 번역표 구성의 마지막 단계는 페이지표 또는 블록표로 나누어진다. 그래서 주소 번역표는 마지막 단계로 페이지 또는 블록 중의 하나만 지원하는 단층 구조와 페이지와 블록 두가지 모두를 동적으로 혼합 사용할 수 있는 계층적 복층 구조로 나눌 수 있다.

페이지 단위의 단층 구조의 주소 번역표는 K. Li에 의해서 처음 사용된 방식으로 기존의 대부분 메모리 관리 하드웨어와 운영체제에서 지원하고 있는 페이징 방법을 그대로 사용이 가능하다[5]. 이 방법은 거짓 공유 현상으로 인하여 일관성 유지시 불필요한 자료 이동이 발생되기 때문에 페이지 크기가 작을수록 유리한 특징을 가지고 있다.

블록 단위의 단층 구조의 주소 번역표는 디렉토리에 기반을 둔 하드웨어 캐쉬의 일관성 유지에 사용되고 있다. 이 방법은 주소 번역표를 위한 메모리 요구량이 페이지 방법에 비하여 수십배 정도로 많아지는 문제를 가지고 있다. 그래서 Y.Tamir는 메모리 요구량을 줄이기 위하여 역표를 사용하는 단층 주소 번역표를 제시하였다[8]. 하지만 이 기법 역시 대부분의 상용 처리기에서는 지원하지 않는 역 페이지표 및 비트맵 형식의 블록표들을 인식하는 전용메모리 관리 하드웨어를 필요로 하는 단점을 가지고 있다.

그리고 Tamir는 블록 단계의 단층 주소 번역표를 메모리 요구량의 측면에서 최적화할 수 있는 대안으로 페이지와 블록의 계층적 복층 구조를 제안하였다[8]. 이 기법에서는 페이지 단계에서 주소 번역표를

역표로 사용하여 메모리 요구량을 줄였다. 하지만 이 기법 역시 전용 메모리 관리 하드웨어가 필요하고, 멀티 태스킹 환경에서 메모리가 낭비가 크며, 주소 앨리어싱 문제를 해결하지 못 했다.

2.2 자료 일관성 관리

자료의 일관성 유지와 관련된 연구 분야는 자료의 소유주 정보 관리 방법과 사본 디렉토리 구조가 있다.

소유주 정보 관리 기법에는 페이지 소유권을 고정 또는 동적으로 관리하는 방법이 있다[5]. 고정 소유권 방식은 항상 동일 처리기에 의하여 소유되는 것으로서, 다른 처리기의 쓰기마다 접근가 발생되는 부담이 있다. 동적 관리는 다시 중앙 집중형과 분산형으로 나눌 수 있다. 중앙 집중형 관리는 중앙 관리자가 소유자 정보를 가지는 방법으로 페이지 접근실패마다 중앙 관리자가 소유주한테 요청을 포워딩해야 하는 단점이 있다. 분산형 관리에서는 모든 처리기가 페이지의 소유주가 될 권한이 있는 것으로 블록 소유주의 변경 시마다 사본을 가진 모든 처리기에게도 알려야 하는 오브헤드를 가진다.

사본 디렉토리 구조는 사본 디렉토리에 기록할 수 있는 항목의 갯수를 제한하는 구조와 제한하지 않는 구조로 나눌 수 있다[2]. 전자는 하나의 자료 블록을 동시에 공유하는 처리기의 갯수가 적다는 가정하에서 사용된다. 후자는 처리기를 비트 벡트로 표시하는 완전사상 디렉토리, 단일 또는 이중 링크 구조를 사용하는 연결리스트 디렉토리, 트리 형식의 디렉토리 등이 있다.

2.3 자료 배치 정책

배치 정책에는 자료의 이주 또는 중복 여부에 따라 네 종류로 분류할 수 있다[7]. 이주와 중복이 일어나지 않고 중앙 서버가 모든 공유 자료를 가지고 접근을 서비스하는 형태, 접근을 요구하는 처리기로 항상 자료가 이주하는 형으로 SRSW(Single Reader/Single Writer) 형태, 읽기 접근 요청시 중복을 허용하는 형으로 MRSW (Multiple Reader/Single Writer) 형태, 마지막으로 읽기 및 쓰기 접근 요청시 모두 중복을 허용하는 형으로 MRMW (Multiple Reader/Multiple Writer) 형태가 있다[5].

3. 다중쓰래드 프로그래밍 기반의 분산 공유 메모리 관리

본 논문에서는 분산공유메모리 시스템에서의 기본 요구 사항인 단일 가상 주소 공간의 제공, 응용 프로그램 투명성 제공, 우수한 확장성 제공, 상용 처리기 및 상용 운영체제하에서 구현 용이성 지원 등을 만족하면서, 표준 다중쓰레드 프로그래밍 모델에 적합한 분산 공유 메모리 관리 기법(Distributed Shared Memory Scheme based on Multithreaded Programming Model: DSMP)을 제안한다.

이 장에서는 DSMP의 주소 번역표 관리 기법, 블록의 일관성 유지 기법, 그리고 동적 배치 정책에 대하여 기술하도록 한다.

3.1 주소 번역표

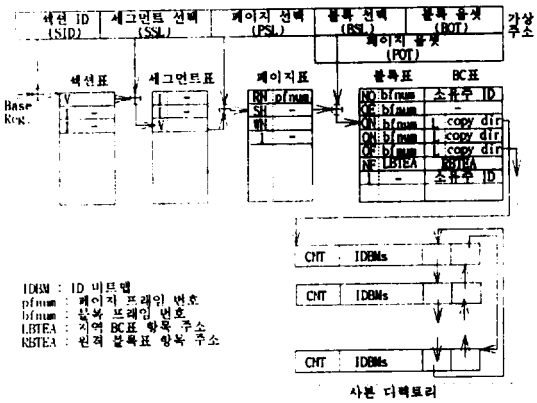
3.1.1 주소번역표의 구조

DSMP의 주소 번역표 구조는 32-비트 프로세서의 메모리 관리 하드웨어를 기본으로, 각 프로세스에 대하여 독립적으로 존재하며, 32 비트의 가상주소를 최대 4 단계까지의 계층적 복층 구조를 갖는다. 계층 구조의 마지막 단계표는 페이지표 또는 블록표 중에서 동적으로 선택할 수 있다.

유닉스 시스템 관점에서 볼 때, 텍스트, 자료와 공유라이브러리들은 세그먼트의 단위에서 주소 공간이 분할되며, 스택은 대부분 주소가 감소하는 방향으로 사용되는 특성 때문에 섹션의 단위에서 분할되어야 할 것이다. 특히 주소 번역표의 할당은 각 쓰레드가 실제 사용하는 주소 공간에 대하여 이루어진다. 그 공간은 프로세스 영역중 몇 개의 연속된 부분들로 구성되므로 적절한 세그먼트 및 섹션 크기의 선택은 효율적인 메모리 관리 차원에서 매우 중요하다.

세그먼트 및 섹션의 크기는 가상 주소내의 섹션 식별자(SID), 세그먼트 선택자(SSL), 페이지 선택자(PSL) 필드의 비트 수를 결정한다. 비선형 정수 계획법을 활용한 결과 세그먼트의 크기는 512K바이트, 섹션의 크기는 32M 바이트, 블록의 크기는 256 바이트, 그리고 페이지는 8K 바이트로 하였다. (그림 1)은 DSMP의 주소 변역표 구조이다. 각 항목에 표시된 상태 정보는 표의 할당과 메모리 일관성 유지에 필요한 것만 표시하고 있다. BC(block coherence)표와 블록표를

각 항목끼리 물리적으로 연속되는 것처럼 표현하였지만, 실제로는 각 표 단위에서 물리적으로 연속된 쌍으로 존재한다. 그리고 블록표와 BC표는 프로그램의 수행중 페이지내 어떤 블록이 공유될 때에 동적으로 표를 구축한다.



(그림 1) 주소번역표 구조
(Fig. 1) Address translation table

(그림 1)에 표시된 페이지 상태 및 블록 상태의 의미는 다음과 같다.

- 페이지 상태
  - RN : 읽기 전용의 페이지; 공유가 일어나면 SH 상태로 변경
  - WN : 쓰기가 허용된 페이지; 공유가 일어나면 SH 상태로 변경; 블록표를 할당
  - SH : 소속 블록중의 하나 이상이 공유된 페이지; 블록표가 할당
- 블록 상태
  - NO : 읽기 전용의 공유블록; 자신은 소유주가 아님
  - OE : 공유블록이 아님; 자신이 소유주; 상위 페이지가 읽기 전용이 아니면 쓰기 허용
  - ON : 읽기 전용의 공유 블록; 자신이 소유주
  - OF : 중복 또는 이주를 불허, 원격접근만 허용하는 블록; 자신이 소유주
  - NF : 원격 접근로서 자료에 접근할 수 있는 블록; 자신은 소유주

섹션표와 세그먼트표의 각 항목은 유효한(V) 상태이면 다음 단계표에 대한 시작 주소를 가진다. 페이지표 항목은 무효(I) 상태가 아니면 유효하며, RN 또는 WN 상태에서는 페이지표가 마지막 단계로서 페이지 프레임 번호를 가진다. SH 상태의 페이지는 다음 단계의 블록표의 시작 주소를 가진다. 블록표 항목은 NF 상태일 때만 원격 블록표의 시작 주소를 가지고, 나머지 상태에서는 블록 프레임 번호를 가진다.

3.1.2 주소 번역표 할당 방법

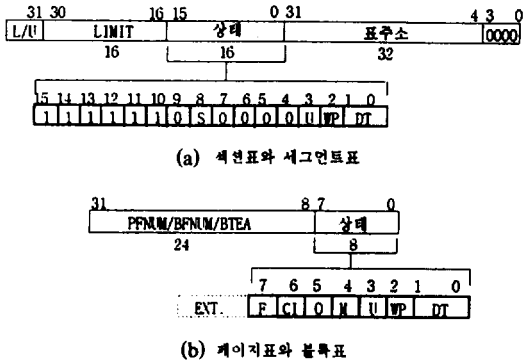
다중쓰레드 프로그래밍 모델에서, 각 쓰레드는 프로세스의 주소 공간중 일부만 공유하므로 쓰레드의 주소 번역표는 최대한 할당을 지연시키는 요구 할당 방법을 사용한다. 즉 프로세스에서는 블록표만 요구에 의한 할당을 하고, 쓰레드에서는 블록표 뿐아니라 세그먼트표와 페이지표도 요구에 의해서만 할당을 한다. 주소 번역표 할당 방식을 프로세스와 쓰레드로 나누어 기술하면 다음과 같다.

- 프로세스
  - 생성시 할당 : 섹션표, 세그먼트표, 페이지표
  - 요구시 할당 : 블록표(페이지내 블록이 공유될 때 할당)
- 쓰레드
  - 생성시 할당 : 섹션표
  - 요구시 할당 : 세그먼트표와 페이지표 (주소 번역표 접근실패시 할당)
    블록표 (페이지내 블록이 공유될 때 할당)

쓰레드에서 세그먼트표 또는 페이지표의 접근실패가 발생되었을 때, 자신의 부모 프로세스(디폴트 소유주)에게 현재의 접근를 인증하도록 요청한다. 프로세스는 정당한 접근으로 확인되면, 주소 번역표의 구축에 필요한 정보와 자료를 요청자 쓰레드로 전송한다. 쓰레드는 프로세스로부터 받은 응답의 내용에 따라 접근실패의 사후 처리를 수행한다. 쓰레드에서 요구시 할당 중에 프로세스와 공유되지 않는 스택 영역을 사상하는 표들은 예외이다.

3.1.3 페이지의 상태와 블록의 상태

소프트웨어적으로 서비스가 필요한 상태의 페이지



(그림 2) 메모리 관리 하드웨어의 주소 번역표 항목
(Fig. 2) Address translation table entry

또는 블록에 접근할 때, 하드웨어적인 접근실패가 발생되어 운영체제가 제어를 넘겨 받을 수 있어야 한다.

메모리 관리 하드웨어의 주소 번역표 항목은 (그림 2)와 같다. (그림 2) (a)는 섹션표와 세스먼트표의 항목이고 (b)는 페이지표와 블록표 항목이다.

(그림 2) (b)의 F 및 O 비트는 블록 접근실패 발생시 해당 블록의 상태를 식별하는데 사용된다. 각 항목들의 의미는 다음과 같다.

- DT(Descriptor Type): 표 항목의 형(types)
  - 0(Invalid): 무효 항목
  - 1(PD): 페이지 또는 블록 번호를 가지는 항목
  - 2(4B): 4 바이트 항목의 표를 가르키는 항목
  - 3(8B): 8 바이트 항목의 표를 가르키는 항목
- WP(Write Protect): 1이면 쓰기 불허
- U(Used): 접근마다 하드웨어적으로 1로 기록
- M(Modified): 쓰기마다 하드웨어적으로 1로 기록
- O(Owned): 1이면 블록의 소유주
- CI(Cache Inhibit): 1이면 캐쉬에 자료 저장 금지
- F(Freezing): 1이면 결빙 또는 원격 접근만 허용된 블록
- S(Supervisor): 1이면 수퍼바이저 모드 접근만 허용
- L/U, LIMIT: 다음 표에서 항목 범위의 하한 또는 상한값

접근실패는 WP 또는 DT 필드에 의해서 결정되어진다. WP 비트가 1(읽기 전용)인 블록 또는 페이지에 쓰기 접근를 시도하면 접근실패가 발생한다. 그리고 DT 비트가 0(무효 항목)인 것에 접근 시도는 읽기 또는 쓰기에 관계없이 접근실패가 발생하며, 페이지 또는 블록의 부재를 의미한다.

페이지 상태들은 하드웨어 비트만으로 명시할 수 있으며, RN 상태의 페이지에서 쓰기 접근시 페이지 접근실패가 발생한다.

- RN: WP==1, DT==1
- WN: WP==0, DT==1
- SH: (WP==1 혹은 0), DT==2

블록 상태들은 하드웨어 비트들과 2개의 소프트웨어 비트의 조합으로 명시할 수 있다. OE 또는 OF 상태의 블록에 대한 쓰기 접근에서는 WP 비트의 값 때문에 블록 접근실패가 발생된다.

- NO: O==0, WP==1, DT==1
- OE: O==1, WP==0, DT==1
- ON: O==1, WP==1, DT==1
- OF: O==1, WP==0, DT==1, F==1
- NF: O==0, WP==(1 혹은 0), DT==1, F==1

3.2 블록 일관성 관리 기법

DSMP에서는, 처리기 사이에 공유되는 블록들에 대한 일관성 유지를 위하여 동일 블록의 사본을 가지는 쓰레드들에게만 선택적으로 무효화 메시지를 보내는 방법을 사용한다. 이 때 사용되는 사본 디렉토리는 블록의 소유주가 갖고 있게 한다. 블록 소유주는 무효화 메시지 전송의 주체로서 일관성 유지뿐만 아니라 블록 부재시 접근실패 처리와 동일 블록에 대한 다중 접근실패의 동기화 역할을 하는 블록 단위의 관리자이다.

소유주는 블록 또는 페이지마다 유일하게 하나씩 존재하며, 소유권은 쓰레드 사이에 동적으로 이동될 수 있다. 모든 관리가 페이지 또는 블록 단위의 소유주를 통하여 일어나므로 소유주가 아닌 쓰레드에서 발생되는 접근실패를 처리하기 위해서는 원하는 자료의 소유주를 효과적으로 찾는 방법이 요구된다. 그리고 쓰레드에서 직접 소유주를 찾지 못하는 경우는 참조되는 것으로서 디폴트 소유주를 참조한다.

3.2.1 소유주 정보 관리

세그먼트 또는 페이지 단계 주소 번역표의 페이지 부재로 인하여 접근실패(페이지부재실패)가 발생한 쓰레드는 해당 페이지의 소유주를 직접 찾을 수 없다. 블록부재실패에서도 소유주를 직접 찾지 못하는 경우가 있는데, 이때 참조하는 대상을 일반적으로 디폴트 소유주라 부른다.

DSMP에서의 각 쓰레드는 자신의 부모 프로세스를 디폴트 소유주로 한다. 쓰레드는 자신의 고유 제어블록에 기록되어 있는 프로세스 번호로써 디폴트 소유주를 쉽게 찾을 수 있다. 그리고 다중쓰레드 프로그래밍 모델에서 대부분의 프로세스는 쓰레드들의 병렬 수행이 완료될 때까지 기다리는 역할만 하고 있으므로, 쓰레드들로부터 소유주 관련 서비스 요청을 다른 희생없이 충분히 처리할 수 있다. 디폴트 소유주는 자신의 주소공간에 속한 모든 페이지와 블록마다 실제 소유주를 항상 기록하고 있어야 한다. 디폴트 소유주에서 페이지 소유주의 기록은 페이지가 쓰레드에 의하여 사용될 때 행해지며, 프로세스의 페이지 표마다 존재하는 페이지 이주표(migration table)를 사용한다. 그리고 블록 소유주의 기록은 공유된 페이지에 속한 블록들에 대해서만 하며, 자신의 블록표마다 존재하는 BC표를 사용한다.

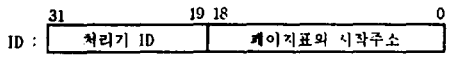
페이지 소유주는 해당 페이지를 처음으로 취한 쓰레드가 되며, 공유가 일어나거나 비공유 상태로 그 페이지의 사용을 포기할 때까지 계속 소유권을 갖는다. 블록 소유주는 페이지내 블록중의 하나 이상이 공유될 때 블록표와 BC표의 할당과 함께 지정된다. 각 블록의 최초 소유주는 페이지 소유주와 동일하지만, 다른 쓰레드에서 쓰기를 지역적으로 수행했다면 소유주는 그 쓰레드로 변경된다. 공유 블록에 대한 쓰기 수행으로 인하여 무효화 메시지를 받은 쓰레드들은 새로운 소유주를 BC표 항목에 기록한다. 모든 쓰레드는 페이지 또는 블록의 소유자가 변경될 때마다 항상 디폴트 소유주에게 알려야 한다.

페이지 또는 블록의 부재와 공유된 블록에 쓰기로 인하여 접근실패가 발생하면, 먼저 소유주를 찾기 위해서 디폴트 소유주인 프로세스에게 요청 메시지를 보낸다. 프로세스는 요청이 그 페이지에 대한 최초의 접근실패이면 프로세스에서 직접 처리하여 응답 메시지를 보내준다. 다른 쓰레드가 이미 가진 비공유 페이지이면 이주 표에 기록된 페이지 소유주한테 그 요청을 넘겨주고, 공유 페이지이면 BC표에서 접근실패가 발생한 주소에 해당되는 블록의 소유주를 찾아서 응답한다.

블록 접근실패에서는 해당 BC표 항목이 소유주 정보를 가지고 있기 때문에 그 소유주에게 요청 메시지를 보낸다. 그러나 해당 블록이 무효화되고 난 후, 소유주가 아닌 쓰레드에서 쓰기로 인하여 소유주가 다시 변경된 경우에는 그 정보는 허위 정보가 된다. 따라서 요청 메시지를 받은 처리기는 해당 블록의 상태에 따라 처리 방법이 달라진다. 첫째, 공유 블록에 쓰기 실패인 경우로 진짜 소유주일 때 직접 응답한다. 둘째, 소유주는 아니지만 유효한 블록을 가지고 있는 경우에는 해당 BC표에 기록된 진짜 소유주한테 그 요청을 전달한다. 셋째, 소유주도 아니며 유효한 블록도 가지고 있지 않은 경우에는 디폴트 소유주인 프로세스에게 그 요청을 전달하여 프로세스가 자신의 BC표 항목에 기록된 진짜 소유주에게 요청을 다시 전달한다.

동일 블록에 대한 다중 접근실패를 동기화하기 위해 모든 요청은 해당 블록의 소유주가 처리하도록 한다. 즉 블록의 사본 제공 및 무효화 메시지의 전송은 소유주의 제어하에서 모두 이루어진다. 소유주는 잠금 비트에 대한 TAS(Test And Set) 명령어를 사용하여, 블록을 잠건 후 하나의 요청만 원자적으로 처리해서, 모든 접근실패들을 요청 순서에 따라 순차적으로 처리한다.

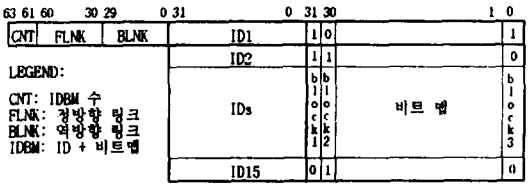
소유주 정보의 형식은 (그림 3)과 같이 주소 앨리어싱의 사용이 가능하도록 되어 있다. 즉, 어느 처리기에 존재하는, 어느 프로세스 주소 공간의 어떤 가상 주소를 사상하는 블록표 또는 페이지표 항목인지를 찾을 수 있도록 소유주 정보를 명시한다. 소유주 정보 형식은 (그림 1)에서의 소유주 ID에 해당된다. 소유주 정보 사용의 장점은 블록 또는 페이지 접근실패의 처리를 요청 받은 처리기가 소유주 쓰레드가 아니더라도 현재 수행중인 쓰레드에게 서비스 요청을 할 수 있다. 그리고 요청 인터럽트를 받은 쓰레드는 페이지표 및 블록표 항목의 주소를 계산하여 해당 잠금 비트를 통하여 블록표 항목의 접근을 동기화하면서 접근실패를 처리할 수 있다. 따라서 접근실패 처리를 신속히 할 뿐만 아니라 컨텍스트 스위칭의 부담이 없다.



(그림 3) 소유주 정보의 형식
(Fig. 3) Owner information format

3.2.2 사본 디렉토리 구조 및 관리

사본 디렉토리는 공유 블록의 사본들이 있는 위치 정보를 가진 자료구조이다. DSMP의 사본 디렉토리는 이중연결 리스트 구조로 (그림 4)와 같고, 연결 형태는 (그림 1)과 같다. ID는 소유주 정보로 사본 디렉토리에서도 주소 앨리어싱 문제를 해결할 수 있다. 각 노드의 크기는 버디 방법을 사용한 메모리 할당에 적합한 128 바이트이고, 사본 디렉토리의 각 노드는 128 바이트 크기로 최대 15개의 항목을 가질 수 있다. 범람(overflow)이 발생되면 하나의 노드를 더 할당받아 이중연결로 연결하여 사용한다.



(그림 4) 사본 디렉토리 노드의 형식
(Fig. 4) Copy directory node format

사본 디렉토리는 공유 페이지마다 하나씩 존재하며, 노드 항목(IDBM)은 특정 ID가 공유하고 있는 모든 블록들을 비트맵으로 표시한다. 즉 비트가 1이면 대응되는 블록의 사본을 ID가 가지고 있음을 의미한다. 따라서 사본 디렉토리의 효율성은 하나의 페이지에 속한 블록 중에서 소유주인 블록의 갯수에 의존적이다. 이러한 사본 디렉토리 구조는 제한 디렉토리의 메모리 절약과 간단한 관리등의 장점도 가지면서 성능 손실없이 범람을 처리할 수 있는 장점이 있다.

다음 알고리즘은 엔트리의 추가, 삭제, 추출하는 절차를 작업도 효율적으로 수행할 수 있다.

· 추가(add) 알고리즘:
IF HEAD.BLNK.CNT < MAXCNT(==15)
THEN HEAD.BLNK.ID[CNT++]=ID
ELSE 새로운 노드(P)를 할당하고 리스트에 연결
P.ID[0]=ID
P.CNT=1

· 추출(fetch all) 알고리즘:
HEAD부터 FLNK를 따라서 CNT개의 ID를 추출
Flink가 HEAD에 도달될 때까지 수행

· 삭제(delete) 알고리즘:
순차적 검색에 의하여 요청자 ID를 찾는다(P).
HEAD.BLNK의 마지막 ID(HEAD.BLNK.ID[--CNT])를 P로 이동

3.3 동적 배치 정책

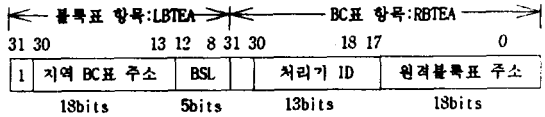
처리기 사이에 자료를 언제 어디로 이동할 것인가를 결정하는 배치 정책은 자료 접근 지연 시간과 상호연결망 교통량을 최소화하는 면에서 고려되어야 한다. DSMP의 배치 정책은 접근실패마다 자료를 중복 또는 이주하여 지역접근도 허용하고, 필요에 따라 자료를 결빙 및 해빙하도록 하여 원격접근도 허용한다.

3.3.1 원격 접근을 위한 하드웨어 모듈

원격 메모리의 접근을 처리하기 위하여 처리기 또는 메모리 뱅크의 식별자를 해당 지역 메모리 주소 앞에 첨가하는 방법을 사용한다. 각 처리기에서 이 방법을 지원해 주기 위한 하드웨어 모듈을 MMT(Memory-to-Messages/Messages-to-Memory)라고 한다[8].

요청자 입장의 MMT는 메모리 관리 하드웨어로부터 생성된 물리적 주소가 원격접근 요청인가를 인식하여, 원격 처리기에 소속된 MMT에게 이를 메시지로서 전달하는 기능을 수행한다. MMT는 물리적 주소의 최상위 비트가 1인 것을 자신에 대한 접근 요청으로 인식하고 그 주소를 가로챔으로서 이루어진다.

MMT에서 원격접근 요청을 위한 주소 정보는, 원격자료의 주소 명시 기능과 동적 해빙 결정용 누적 접근 횟수들의 정확도를 높이기 위하여, 처리기 식별



(그림 5) 원격 접근용 항목의 형식
(Fig. 5) Remot access entry format

자와 블록표 항목의 주소를 사용한다(그림 5).

MMT는 받아들인 물리적 주소로부터 BC표의 시작 주소, 블록 선택자, 블록 陝셋을 얻을 수 있다. 이러한 정보로부터 BC표 항목 주소를 계산한 후, 처리기 식별자와 원격 블록표 주소를 읽고 해당 항목의 주소를 계산한다. 처리기 식별자의 노드에 속한 MMT에게 블록표 항목의 주소와 블록 陝셋이 포함된 메시지로서 원하는 자료접근를 요청한다. 원격 응답자 MMT는 블록 표 항목의 bfnum 필드와 블록 陝셋으로 물리적 주소를 계산하여 자료접근를 완료한 후 결과를 요청자 MMT에게 전달함으로서 원격접근이 완료된다.

3.3.2 접근실패 발생시 배치

페이지 부재실패 발생시 처리절차는 다음과 같다.

디폴트 소유주인 프로세스에게 서비스 요청
IF (페이지가 사용중)
  THEN IF (비공유 페이지)
    THEN 소유주에게 요청을 포워딩
      ELSE 블록의 소유주에게 요청을 포워딩
ELSE 페이지 전체를 보내고 페이지 이주표에 요청자를 소유주

페이지 부재실패 요청을 받은 페이지 또는 블록 소유주는 해당 페이지내 블록들 중에서 자신이 소유주이면서 공유 가능한 모든 블록의 자료와 나머지 블록의 상태 및 소유주 정보를 모두 요청자에게 전송함으로서 그 다음 블록 부재 실패의 발생을 방지한다. 만약 쓰기에서 부재실패이면, 소유주는 실패 블록에 대하여 무효화도 함께 수행한다.

블록 부재실패가 발생하면 블록 소유주를 찾아 서비스하도록 하며, 접근실패 블록의 자료만 전송한다. 읽기이면 블록의 중복이 일어나고, 쓰기이면 무효화 처리 후 이주가 일어난다. 하지만 페이지 또는 블록 부재실패에서 원하는 블록이 결빙되어 있으면 그 블록은 중복 또는 이주되지 않는다. 쓰기 실패가 발생한 블록은 소유주가 되면서 즉시 결빙된다.

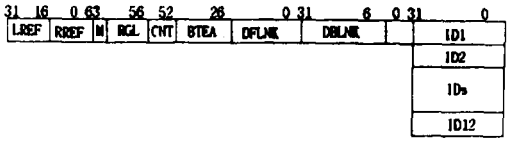
다른 처리기에서 결빙된 블록에 대한 읽기 또는 쓰기의 시도로 실패가 발생하여 소유주에게 서비스를 요청하면, 소유주 처리기는 해당 블록이 결빙되었다는 정보와 함께 블록표의 시작 주소를 응답한다. 이때 소유주는 요청자를 자신의 결빙 블록에 대한 원격접근를 하는 처리기로 인정하여 관련 BC표 항목에서 지정하는 원격접근 디렉토리에 기록한다. 실패 블록이 결빙되었다는 응답을 받은 요청자는 (그림 5)의 형식에 맞게 블록표와 BC표의 항목을 구축한다. 실패 처리를 마치고 복귀하면, 그 이후의 읽기 또는 쓰기 요구는 접근실패의 발생없이 MMT에 의하여 자동적으로 원격접근로 처리된다.

3.3.3 블록 스캐너 데몬

블록의 해빙을 동적으로 결정하는 스캐너 데몬을 각 처리기마다 두어 동적인 배치 정책을 지원한다. 그리고 해빙이 결정된 블록에 대하여 원격 사상을 끊어주는 무효화 메시지의 전송을 책임지는 서버를 따로 둔다. 해빙의 결정은 동결 블록에 대한 지역접근와 원격접근 횟수의 비교로 결정한다.

스캐너 데몬의 오버헤드를 최소화하기 위해서는 원격 접근의 부담을 줄이는 것이 가장 중요하다. DSMP에서는 MMT를 이용하여 동결 블록이 있는 처리기에서 원격접근없이 직접 원격접근 횟수를 누적할 수 있는 방법을 사용하고, 스캐너 데몬은 결빙된 블록들만 검사할 수 있도록 결빙된 블록들을 연결하는 자료구조를 사용한다. 해빙이 결정된 블록에 대한 원격접근을 무효화하기 위하여 요청하는 메시지의 전송 기능을 갖는 해빙 서버를 처리기마다 하나씩 둔다. 그리고 각 결빙 블록마다 원격접근를 하는 쓰레드의 위치를 기록하는 원격접근 디렉토리를 두어 해빙시 효과적인 무효화가 가능하다.

원격 접근 디렉토리는 (그림 6)과 같이 12개의 ID까지 기록할 수 있는 제한디렉토리 형태이며 지역접근 횟수를 기록하는 LREF와 원격접근 횟수를 기록하는 RREF는 MMT에 의하여 원격 접근마다 각 횟수의 누적에 사용된다.



(그림 6) 원격 접근 디렉토리의 형식
(Fig. 6) Remote access directory format

원격 접근를 수행하기 전에 주어진 블록표 항목의 주소로부터 관련 BC표 항목에 연결된 원격접근 디렉토리를 찾아서 지역접근 및 원격접근 횟수를 먼저 누적한다. 지역접근 횟수의 누적은 블록표 항목의 하드웨어 비트 U(그림 2)를 이용한다. 이때 지역블록이 수정되었으면 해당 블록이 재결빙되어야 하기 때문에 접근 횟수들은 모두 0으로 한다. 블록 단계에서 하드웨어 비트 U는 변경하더라도 앞 단계의 페이지표 항목의 비트값은 남아있으므로 기존의 페이지 대치 알고리즘에는 영향을 미치지 않는다.

해빙 결정 데몬은 지정된 주기마다 수행되어 결빙된 페이지만 검사하여 해빙 여부를 결정한다. 이를 위하여 (그림 6)에 표시된 DFLNK과 DBLNK로서 각 처리기내의 모든 결빙 블록에 대응되는 디렉토리들을 이중연결 리스트로 연결하여 둔다. 그리고 디렉토리에서 대응되는 블록표 항목을 찾을 수 있는 포인터로서 BTEA 필드도 둔다.

지역 블록이 수정되었으면 해빙 결정의 검사에서도 해당 블록이 재결빙되어야 하기 때문에 MMT에서 접근 횟수의 누적시와 마찬가지로 접근 횟수들은 모두 0으로 한다. 서버는 해빙 조건이 만족되는 블록을 찾은 스캐너 또는 원격접근 디렉토리의 범람으로 인하여 실패 처리를 하던 쓰레드로부터 요청을 받을 수 있다. 디렉토리를 해빙 서버의 큐에 연결시키고 관련 BC표 항목의 잠금 비트(L)를 통하여 블록의 배타적 사용권을 얻으면 즉시 요청자에게 제어권을 넘기게 하므로써 스캐너의 오버헤드를 덜 수 있다. 서버에서 무효화 처리는 큐에 도착 순서대로 수행하는데, BC표 항목의 잠금 비트를 통하여 잠금 권한을 획득했기 때문에 각 블록의 소유주 입장에서 이루어진다. 이때 RACDE의 M값은 해당 블록표 항목의 M으로 다시 옮겨야 한다. 원격 사상을 하고 있던 블록은 무효화된 후의 첫 접근에서 블록 실패가 발생하게 되어 실패시의 블록 배치 정책에 따라 블록의 상태가 바뀌게 된다.

4. 비교 평가

DSMP의 효율성을 기존의 기법들과 비교 분석을 하였다. 비교 항목은 수행 프로그램의 평균 크기와 실패 접근 유형의 통계 자료들을 근거한 자료 구조의 메모리 요구량, 소유주 관리용 메시지 갯수, 동적 배치 정책의 개선안에 대하여 평가하였다.

4.1 메모리 요구량

DSMP에서는 거짓 공유를 줄이기 위해서 페이지 크기의 수십분의 일인 블록을 사상과 공유의 최소 단위로 사용하였다. Tamir가 제안한 주소 번역표는 메모리 요구량은 비교적 효과적으로 줄일 수 있지만, 특별한 메모리 관리 하드웨어가 필요하고 주소 앨리어싱을 해결하지 못하는 문제점을 가지고 있다. DSMP에서의 주소 번역표 구조와 관리 방식은 이러한 문제점을 해결하면서 다양한 종류의 응용 프로그램들이 멀티태스킹으로 수행되는 환경에서의 메모리 요구량을 더 줄일 수 있다.

하나의 프로세스를 병렬 수행하는데 필요한 주소 번역표와 사본 디렉토리의 메모리 요구량을 다음의 가상 환경하에서 Tamir의 방식과 비교 분석하였다.

- 처리기 갯수: 최대 8,192
- 지역 메모리 크기: 최대 64 MB
- 최대 가상 주소 공간: 4 GB
- 섹션 크기: 32 MB
- 세그먼트 크기: 512 KB
- 페이지 크기: 8 KB
- 블록 크기: 256 바이트
- 사본 디렉토리 항목의 최대 갯수: 30
- 공유 비율: 15%
- 병렬 수준(프로세스당 쓰레드 갯수): 512
- Tamir의 구조에서 해쉬표의 크기: 역표 항목의 1/4
- 수행 프로그램의 평균 크기: 512개 처리기로 수행한 경우들만 고려함
  - 프로세스의 크기: 817 MB
  - 쓰레드의 크기: 2019 KB

〈표 1〉은 Tamir가 제안한 단층 구조 및 계층적 복층 구조의 메모리 요구량을 각 처리기 단위의 주소 번역표와 시스템 전체적인 사본 디렉토리의 크기를 나타내고 있다.

〈표 2〉는 DSMP의 주소 번역표에서 요구하는 메모리 요구량이다. 주소 번역표의 할당은 항목 단위가 아니라 각 단계의 표 단위로 이루어진다. 따라서 〈표 2〉에서 크기의 단위는 각 표의 할당 단위이며, 갯수는 표의 갯수이다. 섹션표는 주소 번역표의 첫 단계

〈표 1〉 Tamir의 주소 번역표의 메모리 요구량
〈Table 1〉 Memory requirement of tamir's address translation table

방 식	표의 종류	크기의 단위	갯 수	표의 크기
단층 주소 번역표	HAT	4 B	32 K	128 KB
	PBT	10 B	128 K	1280 KB
	(합계)			1408 KB
	사본 디렉토리	60 B	490.2 K	28.723 MB
계층적 복층 주소 번역표	HAT	4 B	1 K	4 KB
	PFT	10 B	4 K	40.0 KB
	SPT	17 B	0.6 K	10.2 KB
	(합계)			54.2 KB
	사본 디렉토리	192 B	15687 K	2.872 MB

〈표 2〉 DSMP 주소 번역표의 메모리 요구량
〈Table 2〉 Memory requirement of DSMP's address translation table

프로세스/처리기	표의 종류	크기의 단위	갯수	표의 크기
프로세스 주소공간	섹션 표	1024B	1	1KB
	세그먼트 표	512B	26	13KB
	페이지 표	256B	1634	408.5KB
	블록 및 BC 표	256B	15687	3921.75KB
	(합계)			4.242MB
쓰레드 주소공간	섹션 표	1024B	1	1KB
	세그먼트 표	512B	3	1.5KB
	페이지 표	256B	7	1.75KB
	블록 및 BC 표	256B	38	9.5KB
	(합계)			13.75KB
	사본 디렉토리	256B	15687	3.830MB

이므로 주소 공간의 크기에 관계없이 한개만 필요하다. 나머지 세 단계의 각 표의 갯수는 프로세스와 쓰레드 주소 공간으로 분리하여 계산해야 된다.

프로세스 주소 공간을 사상하는 세그먼트표 및 페이지표의 갯수는 817 MB를 각 표가 사상할 수 있는 크기로 나누므로서 계산되며, 블록표의 갯수는 Tamir의 복층 구조에서 사본 디렉토리의 갯수와 같다.

하나의 프로세스를 병렬 수행하는데 필요한 주소 번역표의 전체 메모리는 〈표 1〉에서는 모든 번역표의 메모리를 512로 곱하며, 〈표 2〉에서는 쓰레드의 번역표 합계에 512로 곱한 값에 프로세스의 번역표의 크기를 더하여 계산된다. DSMP에서의 주소 번역표가 필요로 하는 메모리 크기를 Tamir 방식과 비교한 결과는 〈표 3〉과 같다.

〈표 3〉에서처럼 DSMP의 주소 번역표 구조가 Tamir

〈표 3〉 주소 번역표의 메모리 요구량 비교
〈Table 3〉 Comparision of memory requirement

	주소 번역표		사본 디렉토리		합 계	
	크 기	비율	크 기	비율	크 기	비율
DSMP	11.117MB	1.00	3.830MB	1.00	14.947MB	1.00
Tamir의복층 구조	27.100MB	2.44	2.872MB	0.75	29.972MB	2.01
Tamir의단층 구조	704.000MB	63.33	23.723MB	7.50	732.723MB	34.63

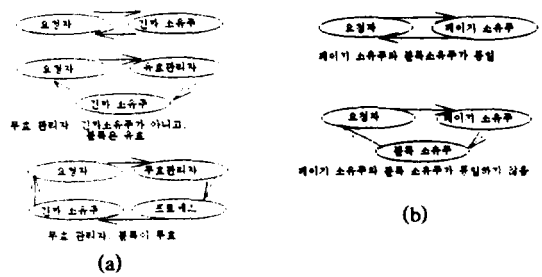
의 계층적 복층 구조보다 2.44배 우수하며, 사본 디렉토리의 메모리까지 합한 경우에도 2.01배 우수하다. 그러나 주소 앨리어싱의 문제를 해결하기 위하여 Tamir의 방식에도 메모리를 추가하면 차이가 훨씬 많이 생길 것이다. 또한 DSMP의 구조에서는 블록 단계의 효과적인 소유주 관리와 동적 블록 배치 정책에 사용되는 BC표에 메모리의 비교적 많은 부분이 사용되고 있다.

4.2 소유주 관리용 메시지 갯수

분산 공유 메모리의 효율적인 일관성 유지를 위하여 페이지 또는 블록의 부재 실패 발생의 최종 서비스 제공자는 해당 블록의 소유주이다. 이러한 소유주를 찾기 위한 메시지의 갯수는 네트워크 교통량과 직접적인 관계를 가지고 확장성에도 영향을 미친다.

Li 방식의 가상 소유주 방식은 전체의 처리기 갯수(q)와 동일 페이지를 공유할 수 있는 처리기 갯수(p)에 영향을 받는다[5]. 이 방식에서의 복잡도는 $O(q + Klogp)$이다. DSMP 방식과 Tamir의 방식에서 블록 부재실패시 소유주를 찾는 가능한 경로들을 (그림 7)의 (a)와 (b)와 같다.

DSMP는 Li 방식과는 달리 처리기 갯수에는 영향



(그림 7) 블록 소유주 검색 경로
(Fig. 7) Block owner access path

을 받지 않는다. 최악의 경우도 평균 메시지의 갯수 면에서는 DSMP 방식이 우수하다. 왜냐하면 블록 부재실패는 다른 처리기에서 쓰기의 수행으로 인하여 무효화를 당한 후 발생되며, 무효화 요청자를 실패 블록의 소유주로 기록해 두고 있다. 그리고 쓰기시 블록을 한동안 결빙하는 정책과 접근의 시간 국부성 때문에 실패 블록의 BC표에 있는 소유주가 진짜일 가능성은 아주 크다. Tamir 방식에서 페이지 소유주는 그 페이지의 사용을 포기할 때까지 변경되지 않으나 블록 소유주는 쓰기 실패마다 이동되기 때문에 페이지 소유주와 블록 소유주가 다를 확률은 아주 크다. 따라서 DSMP에서 평균 메시지 갯수는 1보다 약간 큰 값이 되고 Tamir의 방식은 2보다 약간 작은 값이 된다. Tamir 방식은 시뮬레이션 결과 1024개의 처리기하에서 평균 5.26(최악의 환경 가정) 또는 3.64(무작위)였다[5].

소유주 정보의 관리에서 검색 메시지 외에 소유주가 변경될 때 이를 알리는 메시지도 필요하다. DSMP는 블록 소유주가 변경되면 디폴트 소유주인 프로세스에 알려야 하며, Tamir 방식에서는 페이지 소유주에게 알려야 한다. 그러나 페이지의 소유주가 변경되면 DSMP가 Tamir의 것보다 우수하다. DSMP에서는 비공유 페이지인 경우에만 디폴트 소유주의 이주표를 수정하면 된다. Tamir의 방식은 페이지의 공유 여부에 관계없이 디폴트 소유주뿐만 아니라 그 페이지의 블록들을 사용하고 있는 모든 처리기에게 알려야 하므로 많은 메시지가 한꺼번에 발생되는 부담이 있다.

4.3 동적 배치 정책 효율성

DSMP의 배치 정책은 실패가 발생한 블록이 시스템 내 어느 처리기에도 존재하지 않으면 그 블록을 이주하고, 이미 존재하는 블록이면 최근에 쓰기가 일어나지 않았을 경우에만 실패의 유형에 따라 중복 또는 이주시키는 방법을 사용하였다. 그리고 접근 횟수를 사용하여 결빙을 결정한다.

DSMP의 동적 결빙 정책에서는 스캐너 데몬의 오버헤드를 줄이기 위해서 지역 메모리마다 결빙된 블록들의 리스트를 구축하여 두고, 스캐너는 그 리스트의 블록들만 검사하도록 하였다. 원격 접근 횟수의 누적을 위하여 원격 메모리의 블록 표들을 찾을 필요가 없으므로 각 스캐너 수행 시간을 LaRowe의 방식[3]에 비하여 무시할 수 있을 정도로 줄일 수 있었다. 스캐너 데몬의 수행 주기를 100ms로 하더라도 수행 오버헤드는 1%보다 작으므로 LaRowe의 방식(1 또는 10초의 수행 주기)보다 정확한 동적 해빙 결정을 할 수 있다. 또한 MMT 하드웨어 모듈에서 원격 및 지역접근 횟수를 누적함으로 결정에 사용되는 정보의 정밀성을 훨씬 높일 수 있다. 그리고 원격 사상을 하고 있는 쓰레드들의 리스트를 이용하여 해빙이 필요한 곳에만 무효화 메시지를 보낼 수 있다.

5. 결 론

본 논문에서는 주소 번역표 관리, 블록 일관성 유지, 블록 배치 정책을 종합적으로 고려한 효율적인 분산 공유 메모리 관리 방식인 DSMP를 제안하였다. 분산공유메모리 기법의 기본적인 요구 사항인 사용자 투명성, 우수한 확장성, 구현 용이성, 다중 쓰레드에 의한 병렬 프로그래밍 모델의 지원 등을 달성하면서 분산 공유 메모리 기법의 일반적인 구현 문제점인 거짓 공유, 불필요한 중복, 블록 바운싱의 현상들을 줄일 수 있었다. 그리고 주소 앨리어싱의 지원이 필요한 점을 제기하고, 주소 번역표를 통한 사상과 자료 위치 정보의 관리에서 이를 해결할 수 있음을 보였다.

주소 번역표는 프로세스의 주소 공간마다 존재하는 형태이지만, 쓰레드의 주소 번역표는 다중 쓰레드 프로그래밍 모델의 특성에 기반을 두고 실제 사용되는 주소 영역에 대해서만 요구 할당하는 방식외에 비선형 정수 계획법에 의한 최적의 4 단계 구조의 설계로 메모리의 요구량을 최소로 하였다. 주소 번역표의 구조 설계에서는 블록의 일관성 유지와 동적 블록 배치 정책에 사용되는 페이지 및 블록의 상태와 여러 자료구조들을 효율적으로 표시하는 것도 함께 고려하였다.

주소 번역표를 비롯한 여러 자료 구조를 MC68030 메모리 관리 하드웨어에서 실현할 수 있도록 설계하였으며, C와 같은 프로그래밍 언어로 간단히 정의할 수 있도록 하였다. 그리고 기존의 운영체제 커널에서 주소공간 관리 및 페이지 접근실패 처리 방식을 확장하여 구현할 수 있다.

참 고 문 헌

[1] W. Boloski, R. Fitzgerald, and M. L. Scott, "Simple But Effective Techniques for NUMA Memory Management," Proc. 12th Symp. on Operating Systems Principles, pp. 19-31, 1989.

[2] D. Chaiken, C. Fields, K. Kurihara, and A. Agarawal, "Directory-Based Cache Coherence in Large-Scale Multiprocessors," IEEE Computer, Vol. 23, No. 6, pp. 49-58, June 1990.

[3] R. LaRowe, Jr., 'Page Placement for Nonuniform Memory Access Time(NUMA) Shared Memory Multiprocessors,' Ph.D. dissertation, Duke University, Mar. 1991.

[4] K. Li and P. Hudak, "Memory Coherence in shared virtual memory systems," Proc. the 5th Annual ACM Symposium on Principles of Distributed Computing, pp. 229-239, Aug. 1986.

[5] K. Li and P. Hudak, "Memory Coherence in Shared Virtual Memory Systems," ACM Trans. on Computer Systems, Vol. 7, No. 4, pp. 321-359, Nov. 1989.

[6] B. Nitzberg and V. Lo, "Distributed Shared Memory: A Survey of Issues and Algorithms," IEEE Computer, pp. 52-60, Aug. 1991.

[7] M. Stumm and S. Zhou, "Algorithms Implementing Distributed Shared Memory," IEEE Computer, pp. 54-64, May 1990.

[8] Y. Tamir and G. Janakiraman, "Hierarchical Coherency Management for Shared Virtual Memory Multicomputers," Journal of Parallel and Distributed Computing 15, pp. 408-419, Aug. 1992.

서 대 화

1981년 경북대학교 전자공학과 졸업(학사)

1983년 한국과학기술원 전산학과(석사)

1993년 한국과학기술원 전산학과(박사)

1981년~1995년 한국전자통신연구소 시스템 S/W연구실 실장

1995년~현재 경북대학교 전자·전기공학부 조교수

관심분야: 병렬/분산 컴퓨터 구조, 병렬운영체제, 멀티미디어 서버