(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년10월14일
(11) 등록번호 10-1072807
(24) 등록일자 2011년10월06일

(51) Int. Cl.
G06F 12/12 (2006.01) *G06F 11/30* (2006.01)
G06F 12/06 (2006.01) *G06F 12/08* (2006.01)
(21) 출원번호 10-2010-0095443
(22) 출원일자 2010년09월30일
심사청구일자 2010년09월30일
(56) 선행기술조사문헌
KR1020060117869 A
KR1020060120638 A
KR1020060130081 A

(73) 특허권자
한국과학기술원
대전 유성구 구성동 373-1
(72) 발명자
박규호
대전 유성구 구성동 한국과학기술원 6-3208
황우민
대전 유성구 구성동 한국과학기술원 서측 생활관 6228호
(뒷면에 계속)
(74) 대리인
김성호

전체 청구항 수 : 총 12 항 심사관 : 권오성

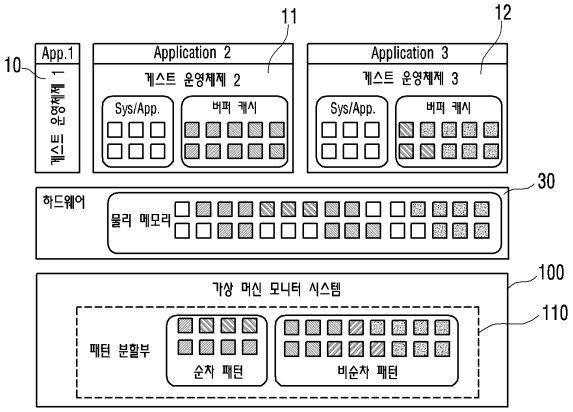
(54) 가상 머신 모니터 시스템

(57) 요 약

실시예는 가상 머신 모니터 시스템에 관한 것이다.

실시예에 따른 가상 머신 모니터 시스템은, 복수의 게스트 운영체제에 할당된 페이지 프레임 중 버퍼 캐시로 사용되는 페이지 프레임들을 추출하고, 데이터 접근에 대한 연속성을 기준으로 추출된 페이지 프레임들을 순차 패턴 및 비순차 패턴으로 분할(partition)하는 패턴 분할부, 순차 패턴 및 비순차 패턴에 속하는 페이지 프레임들을 LRU 정책에 따라 관리하되, 순차 패턴에 속하는 페이지 프레임들은 연속 페이지 프레임 단위로 관리하고, 비순차 패턴에 속하는 페이지 프레임은 페이지 프레임 단위로 관리하는 패턴 관리부, 및 복수의 가상 머신 간의 메모리 재할당을 위하여 순차 패턴에 속하는 연속 페이지 프레임을 우선적으로 선택하고, 선택된 페이지 프레임을 메모리를 필요로 하는 가상 머신으로 할당하는 메모리 선택 처리부를 포함한다.

대 표 도 - 도3



(72) 발명자
박영우
경기 성남시 중원구 은행2동 708번지

박기웅
서울 노원구 월계4동 500-11 8/5

특허청구의 범위

청구항 1

단일 하드웨어를 갖는 컴퓨터 시스템 상에서 복수의 가상 머신에 할당된 메모리를 조정하기 위한 시스템에 관한 것으로,

복수의 게스트 운영체제에 할당된 페이지 프레임 중 버퍼 캐시로 사용되는 페이지 프레임들을 추출하고, 데이터 접근에 대한 연속성을 기준으로 상기 추출된 페이지 프레임들을 순차 패턴 및 비순차 패턴으로 분할(partition)하는 패턴 분할부;

상기 순차 패턴 및 상기 비순차 패턴에 속하는 페이지 프레임들을 LRU 정책에 따라 관리하되, 상기 순차 패턴에 속하는 페이지 프레임들은 연속 페이지 프레임 단위로 관리하고, 상기 비순차 패턴에 속하는 페이지 프레임은 페이지 프레임 단위로 관리하는 패턴 관리부; 및

복수의 가상 머신 간의 메모리 재할당을 위하여 상기 순차 패턴에 속하는 연속 페이지 프레임을 우선적으로 선택하고, 상기 선택된 페이지 프레임을 메모리를 필요로 하는 가상 머신으로 할당하는 메모리 선택 처리부를 포함하는 가상 머신 모니터 시스템.

청구항 2

제1항에 있어서,

상기 패턴 분할부는,

게스트 운영체제의 버퍼 캐시 관리 코드에서 하이퍼콜 함수(hypercall function)을 호출하도록 하여 게스트 운영체제에 할당된 페이지 프레임들에 대한 정보를 전달받고,

상기 페이지 프레임들에 대한 정보에 기초하여 복수의 게스트 운영체제에 할당된 페이지 프레임 중 버퍼 캐시로 사용되는 페이지 프레임들을 추출하는, 가상 머신 모니터 시스템.

청구항 3

제1항에 있어서,

상기 패턴 분할부는,

복수의 게스트 운영체제에서 동작하는 어플리케이션 레벨(application level)에서, 단일 파일에 대한 단일 프로세스의 데이터 접근이 연속적인지를 기준으로, 상기 추출된 페이지 프레임들을 순차 패턴 및 비순차 패턴으로 분할하는, 가상 머신 모니터 시스템.

청구항 4

제3항에 있어서,

상기 패턴 분할부는,

각 프로세스의 페이지 테이블을 가리키는 레지스터 값과, 각 프로세스가 접근한 페이지 프레임이 담고 있는 데이터가 저장된 디스크의 블록 번호를 이용하여 버퍼 캐시에 속한 페이지 프레임들에 대한 각 프로세스의 접근 순서를 추적하고,

상기 페이지 프레임들에 대한 각 프로세스의 접근 순서에 기초하여 기설정된 개수 이상의 페이지 프레임들이 연속적으로 접근될 경우 해당 페이지 프레임들을 순차 패턴으로 분류하는, 가상 머신 모니터 시스템.

청구항 5

제4항에 있어서,

상기 패턴 분할부는,

순차 패턴의 분류를 위해, 원하는 데이터가 캐시에 존재하지 않을 경우에 접근하고자 하는 대상 데이터를 읽어 들이기 위한 I/O 리퀘스트를 모니터링하여 대상 페이지 프레임의 정보 및 해당 페이지 프레임에 저장된 데이터에 대한 정보를 파악하는, 가상 머신 모니터 시스템.

청구항 6

제1항에 있어서,

상기 패턴 분할부는,

복수의 게스트 운영체제에서 동작하는 파일 레벨(file level)에서, 단일 파일에 대한 단일 프로세스의 데이터 접근이 연속적인지를 기준으로, 상기 추출된 페이지 프레임들을 순차 패턴 및 비순차 패턴으로 분할하는, 가상 머신 모니터 시스템.

청구항 7

제6항에 있어서,

상기 패턴 분할부는,

사용자 레벨에서 파일 접근 시스템 호출(file read/write system call)이 발생될 경우 상기 시스템 호출에 포함된 함수 인자를 얻고,

상기 시스템 호출의 실행 중에 각각의 프로세스에 의해 접근된 페이지 프레임들의 주소를 포함하는 페이지 프레임 접근 정보를 얻고,

각 게스트 운영체제 내부의 파일 접근 정보와 상기 페이지 프레임 접근 정보를 결부시켜, 각 파일의 특정 위치의 블록 데이터가 각 게스트 운영체제 내부의 페이지 프레임의 어느 위치에 저장되어 있는지에 대한 상관 정보를 얻고,

상기 상관 정보를 이용하여 태스크에 의해 접근된 파일에 대한 연속성을 파악하는, 가상 머신 모니터 시스템.

청구항 8

제7항에 있어서,

상기 패턴 분할부는,

상기 파일 접근 시스템 호출에 의해 발생된 함수 인자에 대한 정보와, 데이터가 저장된 물리 페이지 프레임의 주소 정보를 연결하기 위하여, 각 태스크 아이디에 대해서 사용된 태스크 아이디, 파일 아이디, 파일 접근 시작 위치, 파일 접근 종료위치, 파일 접근 구간, 물리 페이지 프레임 주소로 구성되는 테이블 엔트리를 구성하고,

임의의 태스크가 특정 파일에 접근할 경우, 상기 테이블 엔트리를 이용하여 상기 물리 페이지 프레임의 접근 연속성을 파악하는, 가상 머신 모니터 시스템.

청구항 9

제1항에 있어서,

상기 패턴 관리부는,

상기 순차패턴 및 상기 비순차 패턴에 속하는 페이지 프레임들의 재배치가 이루어지는 경우, 상기 재배치될 페이지 프레임의 데이터를 영구 저장공간에 저장한 후 재배치를 수행하며,

상기 재배치된 페이지 프레임으로 게스트 운영체제의 접근이 발생될 경우, 새로운 빈 페이지 프레임을 할당하고 상기 영구 저장공간에 저장된 데이터를 상기 빈 페이지 프레임으로 복사한 후 페이지 테이블의 매핑을 수정하여 데이터를 메모리로 복원하는, 가상 머신 모니터 시스템.

청구항 10

제1항에 있어서,

상기 패턴 관리부는,

연속 페이지 프레임에 속한 페이지 프레임이 접근될 경우, 해당 연속 페이지 프레임을 MRU 위치로 이동시키고,

상기 메모리 선택 처리부는,

상기 MRU 위치에 있는 연속 페이지 프레임 중 가장 최근에 접근된 페이지 프레임부터 필요한 개수만큼의 페이지 프레임을 선택하는, 가상 머신 모니터 시스템.

청구항 11

제1항에 있어서,

상기 패턴 관리부는,

연속 페이지 프레임에 속한 페이지 프레임이 접근될 경우, 해당 연속 페이지 프레임을 MRU 위치로 이동시키고,

상기 메모리 선택 처리부는,

상기 MRU 위치에 있는 연속 페이지 프레임 중 가장 최근에 접근된 페이지 프레임부터 필요한 개수만큼의 페이지 프레임을 선택하되, 일정 개수의 마진(margin)을 두고 상기 마진에 포함되지 않은 가장 최근에 접근된 페이지 프레임부터 LRU 방향으로 선택하는, 가상 머신 모니터 시스템.

청구항 12

제1항에 있어서,

상기 메모리 선택 처리부는,

연속 페이지 프레임이 없는 경우, 상기 비순차 패턴에 속하는 페이지 프레임 중에서 LRU 위치에 있는 페이지 프레임을 선택하는, 가상 머신 모니터 시스템.

명 세 서

기 술 분 야

[0001] 실시예는 가상 머신 모니터 시스템에 관한 것이다.

배 경 기 술

[0002] 가상화(virtualization)는 하드웨어 리소스(hardware resource)를 멀티 플레싱(multiplexing)함으로써 하나의 물리 컴퓨터 상에서 다수의 OS(operating system)을 실행시킬 수 있도록 한다. CPU 및 I/O 장치는 시간 단위로 나누어 해당 시간만큼 사용하는 방식으로 여러 게스트 OS를 공유할 수 있다.

[0003] 그러나, 메모리와 같은 리소스는 해당 자원을 상기와 같은 방식으로 공유하기에는 오버헤드(overhead)가 매우

크기 때문에 시간 단위로 배분하는 것이 아니라 용량적 측면에서의 배분을 통해 자원을 공유한다. 따라서 적절한 순간에 적절한 양의 메모리를 할당하는 것이 시스템 전체적인 측면에서 성능에 중요한 역할을 하게 된다.

[0004] 가상화 환경을 제어하는 하이퍼바이저(hypervisor) 또는 가상 머신 모니터(virtual machine monitor, VMM)는 하드웨어 리소스를 멀티플레싱하여 각 게스트 OS에게 자신이 컴퓨터 리소스를 독점한 것과 같은 일루전(illusion)을 제공한다. 따라서 게스트 OS의 구동 시, QoS(Quality of Service)를 보장하기 위해 해당 게스트 OS에게 필요한 자원을 SLA(Service Level Agreement)를 준수하는 한도 내에서 자원의 시간적 공유를 가능하게 한다.

[0005] 그러나, 물리 메모리의 경우, 약속된 용량의 메모리를 각 게스트 OS에게 분배하기 때문에, 전체 메모리 용량이 각 게스트 OS들의 메모리 양의 합을 넘어서는 경우에는, 하나의 게스트 OS에 대해서 필요한 용량만큼의 물리 메모리를 공급할 수 없는 경우가 발생하게 된다. 게스트 OS들 중 하나가 함께 실행되지 못하여 유휴 자원이 발생되기 때문에, 하이퍼바이저(hypervisor)는 모든 게스트 OS들에게 약속한 용량의 메모리를 사용하고 있는 일루전(illusion)을 제공하면서 메모리를 오버커밋(overcommit)한다. 즉, 하이퍼바이저(hypervisor)는 각 게스트 OS에게 할당된 메모리를 사용할 때와 근접한 성능을 보이도록 하는 범위 내에서 메모리 자원의 일부를 VM들끼리 시간적으로 공유하도록 하여, 각 게스트 OS가 감내할 수 있는 성능적 페널티(penalty) 내에서 메모리 리소스 사용의 효율성을 극대화한다.

[0006] 하이퍼바이저(hypervisor)는 메모리 오버커밋(memory overcommit)을 수행하기 위하여, 디맨드 페이징(demand paging)을 통해, 사용중인 메모리를 선택적으로 리클래임(reclaim)한 후, 페이지 프레임(page frame)을 재할당한다. 정적 할당 정책(static allocation policy)에 따르는 경우, 단일 VM에게는 고정된 양의 메모리 페이지 프레임이 할당되기 때문에, 메모리 사용 효율성이 떨어지므로, 동적 재할당을 통하여, 해당 VM이 사용하는 메모리를 하이퍼바이저가 리클래임하여 다시 공급함으로써 메모리 재사용을 숨긴다.

[0007] 보편적으로 사용되는 동적 재할당 접근(dynamic reallocation approach)은, 정적 분할(static partitioning)보다 더 높은 수준의 서버 통합(server consolidation)을 가능하게 한다. 그러나, 동적 재할당 접근은 재할당 속도 및 'semantic gap' 에 의한 더블 페이징 문제 및 워킹 셋 샘플링(working set sampling)의 한계에 의한 문제점을 가지고 있다. 벌루닝(ballooning)만을 이용할 경우 하나 이상의 희생 VM(victim virtual machine)에게서 메모리를 명시적으로 리클래임해야 한다. 그러므로, victim VM이 하이퍼바이저의 스케줄러(scheduler)에 의해서 모두 실행되어야 하며, 일정 시간(time quantum) 안에 각 게스트 OS의 리클래임 절차가 완료되어 벌룬 드라이버(balloon driver)로의 메모리 할당이 이루어져 하이퍼바이저가 메모리의 소유권을 가져가야 한다. 따라서 동적 재할당 접근의 경우, 메모리의 재할당이 완료되기 전까지의 시간이 상당히 소모될 뿐만 아니라 할당된 CPU 시간(CPU time quantum)을 메모리 조정에 소비해야 하는 문제점이 존재하게 된다.

[0008] 디맨드 페이징(demand paging)을 사용할 경우, 게스트 OS에 대한 내부 정보를 알 수 없는 하이퍼바이저가 어느 페이지를 리클래임해야 하는지에 대한 결정을 내려야 한다. 그러므로 이러한 경우 하이퍼바이저는 어떤 페이지 프레임이 가장 가치 없는 것인지 결정 내리기가 쉽지 않다.

[0009] 이에 대한 대응방안으로, 하나 이상의 희생 VM을 선택하기 위해 워킹 세트 샘플링(working set sampling) 방식이 사용되기도 한다. 그러나, 긴 순차적 접근(long sequential access)은 워킹 세트의 크기를 과대평가하도록 하여 메모리 크기와 워킹 세트 크기와의 연관관계를 약화시킬 수 있다. 뿐만 아니라, VM에게 할당된 전체 메모리 크기, 힙(heap) 영역 및 버퍼 캐시(buffer cache)의 작용을 감안하지 않은 워킹 세트의 사이즈 추정은 해당 상태에서의 변화만을 알 수 있다는 점에서 어떤 페이지 프레임이 가장 가치 없는 것인지 결정하는 것과 큰 연관관계가 없다.

발명의 내용

해결하려는 과제

[0010] 실시예는 블록 접근 패턴에 의거하여 정확한 메모리 재할당이 가능한 가상 머신 모니터 시스템을 제공함에 목적이 있다.

[0011] 실시예는 메모리 재할당 시 필요한 메모리를 선택하기 위한 가상 머신 별 계산이 필요하지 않은 가상 머신 모니터 시스템을 제공함에 목적이 있다.

과제의 해결 수단

[0012] 실시예에 따른 가상 머신 모니터 시스템은, 단일 하드웨어를 갖는 컴퓨터 시스템 상에서 복수의 가상 머신에 할당된 메모리를 조정하기 위한 시스템에 관한 것으로, 복수의 게스트 운영체제에 할당된 페이지 프레임 중 버퍼 캐시로 사용되는 페이지 프레임들을 추출하고, 데이터 접근에 대한 연속성을 기준으로 추출된 페이지 프레임들을 순차 패턴 및 비순차 패턴으로 분할(partition)하는 패턴 분할부, 순차 패턴 및 비순차 패턴에 속하는 페이지 프레임들을 LRU 정책에 따라 관리하되, 순차 패턴에 속하는 페이지 프레임들은 연속 페이지 프레임 단위로 관리하고, 비순차 패턴에 속하는 페이지 프레임은 페이지 프레임 단위로 관리하는 패턴 관리부, 및 복수의 가상 머신 간의 메모리 재할당을 위하여 순차 패턴에 속하는 연속 페이지 프레임을 우선적으로 선택하고, 선택된 페이지 프레임을 메모리를 필요로 하는 가상 머신으로 할당하는 메모리 선택 처리부를 포함한다.

[0013] 패턴 분할부는,

[0014] 게스트 운영체제의 버퍼 캐시 관리 코드에서 하이퍼콜 함수(hypercall function)을 호출하도록 하여 게스트 운영체제에 할당된 페이지 프레임들에 대한 정보를 전달받고,

[0015] 페이지 프레임들에 대한 정보에 기초하여 복수의 게스트 운영체제에 할당된 페이지 프레임 중 버퍼 캐시로 사용되는 페이지 프레임들을 추출하는 것이 바람직하다.

[0016] 패턴 분할부는,

[0017] 복수의 게스트 운영체제에서 동작하는 어플리케이션 레벨(application level)에서, 단일 파일에 대한 단일 프로세스의 데이터 접근이 연속적인지를 기준으로, 추출된 페이지 프레임들을 순차 패턴 및 비순차 패턴으로 분할하는 것이 바람직하다.

[0018] 패턴 분할부는,

[0019] 각 프로세스의 페이지 테이블을 가리키는 레지스터 값과, 각 프로세스가 접근한 페이지 프레임이 담고 있는 데이터가 저장된 디스크의 블록 번호를 이용하여 버퍼 캐시에 속한 페이지 프레임들에 대한 각 프로세스의 접근 순서를 추적하고,

[0020] 페이지 프레임들에 대한 각 프로세스의 접근 순서에 기초하여 기설정된 개수 이상의 페이지 프레임들이 연속적으로 접근될 경우 해당 페이지 프레임들을 순차 패턴으로 분류하는 것이 바람직하다.

[0021] 패턴 분할부는,

[0022] 순차 패턴의 분류를 위해, 원하는 데이터가 캐시에 존재하지 않을 경우에 접근하고자 하는 대상 데이터를 읽어들이기 위한 I/O 리퀘스트를 모니터링하여 대상 페이지 프레임의 정보 및 해당 페이지 프레임에 저장된 데이터에 대한 정보를 파악하는 것이 바람직하다.

[0023] 패턴 분할부는,

[0024] 복수의 게스트 운영체제에서 동작하는 파일 레벨(file level)에서, 단일 파일에 대한 단일 프로세스의 데이터 접근이 연속적인지를 기준으로, 추출된 페이지 프레임들을 순차 패턴 및 비순차 패턴으로 분할하는 것이 바람직하다.

[0025] 패턴 분할부는,

[0026] 사용자 레벨에서 파일 접근 시스템 호출(file read/write system call)이 발생될 경우 시스템 호출에 포함된 함

수 인자를 얻고,

[0027] 시스템 호출의 실행 중에 각각의 프로세스에 의해 접근된 페이지 프레임들의 주소를 포함하는 페이지 프레임 접근 정보를 얻고,

[0028] 각 게스트 운영체제 내부의 파일 접근 정보와 페이지 프레임 접근 정보를 결부시켜, 각 파일의 시작 정보가 각 게스트 운영체제 내부의 페이지 프레임들 중 어느 페이지 프레임에 저장되어 있는지에 대한 상관 정보를 얻고,

[0029] 상관 정보를 이용하여 태스크에 의해 접근된 파일에 대한 순차성을 파악하는 것이 바람직하다.

[0030] 패턴 분할부는,

[0031] 파일 접근 시스템 호출에 의해 발생된 함수 인자에 대한 정보와, 물리 페이지 프레임의 주소를 가리키는 함수 인자에 대한 정보를 연결하기 위하여, 각 태스크 아이디에 대해서 사용된 태스크 아이디, 파일 아이디, 파일 접근 시작위치, 파일 접근 종료위치, 파일 접근 구간, 물리 페이지 프레임 주소로 구성되는 테이블 엔트리를 구성하고,

[0032] 임의의 태스크가 특정 파일에 접근할 경우, 테이블 엔트리를 이용하여 물리 페이지 프레임의 접근 연속성을 파악하는 것이 바람직하다.

[0033] 패턴 관리부는,

[0034] 순차패턴 및 비순차 패턴에 속하는 페이지 프레임들의 재배치가 이루어지는 경우, 재배치될 페이지 프레임의 데이터를 영구 저장공간에 저장한 후 재배치를 수행하며,

[0035] 재배치된 페이지 프레임으로 게스트 운영체제의 접근이 발생될 경우, 새로운 빈 페이지 프레임을 할당하고 영구 저장공간에 저장된 데이터를 빈 페이지 프레임으로 복사한 후 페이지 테이블의 매핑을 수정하여 데이터를 메모리로 복원하는 것이 바람직하다.

[0036] 패턴 관리부는,

[0037] 연속 페이지 프레임에 속한 페이지 프레임이 접근될 경우, 해당 연속 페이지 프레임을 MRU 위치로 이동시키고,

[0038] 메모리 선택 처리부는,

[0039] MRU 위치에 있는 연속 페이지 프레임 중 가장 최근에 접근된 페이지 프레임부터 필요한 개수만큼의 페이지 프레임을 선택하는 것이 바람직하다.

[0040] 패턴 관리부는,

[0041] 연속 페이지 프레임에 속한 페이지 프레임이 접근될 경우, 해당 연속 페이지 프레임을 MRU 위치로 이동시키고,

[0042] 메모리 선택 처리부는,

[0043] MRU 위치에 있는 연속 페이지 프레임 중 가장 최근에 접근된 페이지 프레임부터 필요한 개수만큼의 페이지 프레임을 선택하되, 일정 개수의 마진(margin)을 두고 마진에 포함되지 않은 가장 최근에 접근된 페이지 프레임부터 LRU 방향으로 선택하는 것이 바람직하다.

[0044] 메모리 선택 처리부는,

[0045] 연속 페이지 프레임이 없는 경우, 비순차 패턴에 속하는 페이지 프레임 중에서 LRU 위치에 있는 페이지 프레임을 선택하는 것이 바람직하다.

발명의 효과

[0046] 실시예에 따르면 블록 접근 패턴에 의거하여 정확한 메모리 재할당이 가능한 가상 머신 모니터 시스템을 제공할 수 있다.

[0047] 실시예에 따르면 메모리 재할당 시 필요한 메모리를 선택하기 위한 가상 머신 별 계산이 필요하지 않은 가상 머신 모니터 시스템을 제공할 수 있다.

도면의 간단한 설명

[0048] 도 1은 실시예에 따른 가상 시스템의 구성을 나타낸 도면.

도 2는 실시예에 따른 가상 머신 모니터 시스템의 구성을 나타낸 도면.

도 3은 실시예에 따른 패턴 분할부의 구성을 설명하기 위해 나타낸 도면.

도 4는 실시예에 따른 패턴 관리부의 구성을 설명하기 위해 나타낸 도면.

도 5 내지 도 8은 실시예에 따른 메모리 재할당 과정을 설명하기 위해 나타낸 도면.

도 9는 실시예에 따른 순차 패턴 내에서 희생 페이지 프레임을 선택하는 방법을 설명하기 위해 나타낸 도면.

도 10은 프로세스의 파일 수준에서 순차 패턴을 추출하기 위한 방법을 설명하기 위해 나타낸 도면.

도 11은 태스크 기반의 패턴 탐지 기법을 설명하기 위해 나타낸 도면.

발명을 실시하기 위한 구체적인 내용

[0049] 실시예에 대한 상세한 설명에 앞서 실시예의 기술적 특징에 대하여 개략적으로 설명한다. 실시예의 기술적 특징을 보다 명확히 설명하기 위해 공개된 문헌을 이용하여 설명하도록 한다.

[0050] 실시예의 기술분야와 관련된 종래의 문헌 중에는 1) “USENIX ATC 2007: Virtual Machine Memory Access Tracing with Hypervisor Exclusive Cache” 및 2) ACM VEE 2009: Dynamic memory balancing for virtual machines”가 있다.

[0051] 첫 번째 1) 문헌의 경우, 메모리 오버커밋(memory overcommit)된 두 개 이상의 가상 머신(Virtual Machine, VM)이 하나의 하드웨어 머신에서 함께 동작할 때, 각각의 가상 머신으로 실제 할당되는 물리 메모리 페이지 프레임(physical memory page frame)의 양을 각각의 가상 머신에게 할당되는 캐시 사이즈(cache size)로 변경함으로써 조정하는 기술에 관하여 개시되어 있다. 여기서, 할당된 물리 메모리 페이지 프레임(physical memory page frame)의 양을 조정하려 할 때, 하이퍼바이저로 각 가상 머신의 캐시 일부분을 옮겨 배타적 캐시(exclusive cache)로 관리하고, 배타적 캐시에 속한 데이터들에 대한 접근(access)을 추적함으로써 각 가상 머신의 배타적 캐시에 대한 실패율 곡선(miss ratio curve, MRC)를 얻어낸다.

[0052] MRC를 이용하여 현재보다 메모리 크기가 줄어들 경우, 각 가상 머신의 실패율(miss rate)의 변화를 추정하고, 이를 이용하여 모든 가상 머신의 실패율(miss rate)의 기하평균(geometric mean)이 최소가 되는 각 가상 머신의 적정 메모리 크기를 결정한다.

[0053] 각 가상 머신의 새로운 메모리 할당량이 정해지면, 희생 가상 머신(victim VM)의 배타적 캐시에서 지정된 양만큼의 페이지 프레임을 수혜 가상 머신(beneficiary VM)으로 할당하여, 발룬 드라이버(balloon driver)를 통해 가상 머신 내부에서 사용되도록 한다. 이때, 하이퍼바이저에서 유지되던 배타적 캐시는 오버헤드의 수준에 따라 모든 내용을 제거하고 빈 페이지 프레임을 원래 속했던 가상 머신으로 되돌리거나 그대로 하이퍼바이저에서 유지한다.

[0054] 실시예의 기술적 특징 중 첫 번째는, 1) 문헌의 기술과 달리, 하이퍼바이저가 별도의 캐시 영역을 관리하지 않으며 가상 머신 별로 MRC를 추적하지 않고서도 블록 접근 패턴(block access pattern)에 의거하여 정확한 메모리 재할당(memory reallocation)이 가능한 것이다.

[0055] 두 번째 2) 문헌의 경우, 상술한 시스템 환경과 동일한 환경에서, 하이퍼바이저에서 사용하는 메모리들의 추적 정보를 활용하여 메모리 재할당을 수행하는 것이 특징이다. 이를 위해 각 가상 머신의 메모리의 일부에 페이지

폴트(page fault)가 일어나도록 특권 보호(privilege protection을 설정하여 해당 페이지로의 접근을 하이퍼바이저가 알 수 있도록 함으로써, LRU(least recently used) 리스트의 끝부분에 대해 LRU 히스토그램을 얻고, 게스트 OS 내부에 교환 사이즈(swap size)의 모니터링(monitoring)을 위한 프로세스를 실행시켜서 교환(swap)되는 메모리의 양을 모니터링한다. LRU 히스토그램을 이용하는 경우는, 메모리가 줄어들 경우의 캐시 미스 카운트(cache miss count)값을 추정하고 교환(swap)되는 메모리의 크기를 이용하여 늘려야 하는 메모리의 양을 추정한다.

[0056] 이와 동시에 수행되는 모든 가상 머신에 대해서 메모리의 양을 늘리거나 줄일 때 전체적인 페이지 미스 카운트(page miss count)를 줄이는 것을 기준으로 하고, 이것을 최소화할 수 있는 각 가상 머신의 메모리 할당량을 결정하여 각 가상 머신이 메모리를 줄여야 할지 아니면 늘여야 할지와, 각각의 경우의 조정량을 결정한다.

[0057] 실시예의 기술적 특징 중 두 번째는, 상술한 2) 문헌의 기술과 달리, 모든 가상 머신들의 버퍼 캐시(buffer cache)만을 대상으로 해당 메모리들이 사용되는 순서를 가지고 LRU 정책으로 통합 관리함으로써, 필요한 메모리를 선택하기 위한 가상 머신 별로 계산을 필요로 하지 않는 것이 기술적 특징이다. 또한, 모든 가상 머신들의 순차적 접근 페이지(sequentially accessed page)들을 추출하여 이들을 다른 페이지들보다 먼저 희생시키는 것이다.

[0058] 이하 실시예에 대하여 첨부한 도면을 참조하여 상세하게 설명하기로 한다. 단, 첨부된 도면은 실시예의 내용을 보다 쉽게 개시하기 위하여 설명되는 것일 뿐, 본 발명의 범위가 첨부된 도면의 범위로 한정되는 것이 아님은 이 기술분야의 통상의 지식을 가진 자라면 용이하게 알 수 있을 것이다.

[0059] 도 1은 실시예에 따른 가상 시스템의 구성을 나타낸 도면이다. 도 2는 실시예에 따른 가상 머신 모니터 시스템(100)의 구성을 나타낸 도면이다.

[0060] 실시예에 따른 가상 머신 모니터 시스템(100)은, 패턴 분할부(110), 패턴 관리부(120)및 메모리 선택 처리부(130)를 포함한다.

[0061] 실시예에 따른 가상 머신 모니터 시스템(100)이 적용되는 환경은, 단일 하드웨어(30) 상에서 복수의 가상 머신이 구동되는 가상 시스템이다. 각 가상 머신의 초기화 시에 할당 받은 메모리의 합이 실제 물리 메모리의 용량보다 크고 이를 보완해주기 위해 가상 머신 모니터 시스템(100)은 가상 머신에서의 메모리 요청에 온디맨드(on-demand)로 대응하면서 각 가상 머신의 메모리 할당량을 조정한다.

[0062] 도 3은 실시예에 따른 패턴 분할부(110)의 구성을 설명하기 위해 나타낸 도면이다.

[0063] 패턴 분할부(110)는 복수의 게스트 운영체제(10, 11)에 할당된 페이지 프레임 중 버퍼 캐시(buffer cache)로 사용되는 페이지 프레임(page frame)들을 추출하고, 데이터 접근에 대한 연속성을 기준으로 추출된 페이지 프레임들을 순차 패턴(sequential pattern) 및 비순차 패턴(unclassified pattern)으로 분할할 수 있다.

[0064] 패턴 분할부(110)가 가상 머신이 사용하는 메모리들 중에서 버퍼 캐시로 사용하는 페이지 프레임들만 추출하는 방법에는 두 가지 방법이 있다.

[0065] 첫 번째 방법은 게스트 운영체제(10, 11)에서 하이퍼콜(hypercall function)를 통하여 페이지 캐시(page cache)로의 페이지 프레임의 삽입, 삭제 및 재사용을 가상 머신 모니터 시스템(100)으로 직접 알려주는 방식이다.

[0066] 보다 구체적으로, 상기 방식은 게스트 운영체제의 버퍼 캐시 관리 코드에서 버퍼 캐시로 페이지 프레임을 추가, 삭제, 재사용하는 부분에 각각의 명령이 발생되었음을 가상 머신 모니터 시스템(100)에게 알려주는 하이퍼콜 함수를 호출하도록 코드를 삽입하여 게스트 운영체제를 수정함으로써, 명시적으로 이벤트의 발생 및 해당 페이지 프레임의 정보를 전달 받는다. 이때, 해당 게스트 운영체제는 페이지 테이블(page table)의 주소와 가상 주소(virtual address)를 함께 전달하여 물리 주소(physical address)를 얻어 낼 수 있도록 한다. 패턴 분할부(110)는 페이지 프레임들에 대한 정보에 기초하여 각 게스트 운영체제에 할당된 페이지 프레임 중 버퍼 캐시로 사용되는 페이지 프레임을 추출한다.

[0067] 두 번째 방법으로는, “ASPLOS 2006: Geiger: monitoring the buffer cache in a virtual machine environment”의 논문을 통해 공개된 페이지 프레임 추출 방법을 이용할 수 있다.

[0068] 기존의 방식은, 먼저 각 가상 머신에서 디스크 읽기(disk read)가 발생하면, 데이터를 스토리지(storage)로부터

읽어들어 버퍼 캐시에 속한 메모리에 복사한다. 이때 접근되는 메모리는 해당 스토리지의 데이터를 캐싱(caching)하게 되는 버퍼 캐시에 속한 메모리이므로, 가상 머신 모니터 시스템(100)은 이 메모리의 주소를 알아내어 관리하게 된다. 관리되고 있는 페이지 프레임으로 새로운 데이터가 들어오게 되면 축출(eviction) 및 재사용으로 인식하거나, 디스크 읽기/쓰기 없이 사용된 페이지 프레임이 재사용되는 경우에도 축출(eviction)로 인식한다.

[0069] 패턴 분할부(110)에서 순차 패턴 및 비순차 패턴을 분리하여 구분 관리하는 방법은 다음과 같다.

[0070] 먼저, 실시예에서 순차 패턴은 한번만 접근(access)되는 연속적인 블록(block)(혹은 페이지 프레임)을 의미하는 것으로 정의한다. 순차 패턴으로 판별되지 않은 패턴들은 비순차 패턴(unclassified pattern)으로 정의한다.

[0071] 순차 패턴의 탐지는 각 가상 머신에서 동작하는 어플리케이션(application) 또는 파일(file) 레벨에서의 연속성을 기준으로 한다.

[0072] 먼저, 패턴 분할부(110)는 어플리케이션 레벨에서의 순차 접근의 탐지를 위해서 각 가상 머신에서 실행되는 프로세스(process)를 식별한다. 가상 머신에서 실행되는 프로세스의 구분은 각 프로세스의 페이지 테이블을 가리키는 레지스터 값(Intel Architecture의 경우 CR3 레지스터 값)의 변화를 기준으로 하며, 이를 통하여 현재 어떤 프로세스가 메모리 레퍼런스(memory reference)를 수행하고 있는지 알 수 있다. 여기서 메모리 레퍼런스란 프로세스가 접근 명령을 내려 메모리로 쓰기 또는 메모리로부터의 읽기를 수행하는 것을 의미한다.

[0073] 패턴 분할부(110)는 프로세스의 구별 기준인 CR3 레지스터의 값과 해당 프로세스가 접근한 페이지 프레임이 담고 있는 디스크의 페이지 프레임 번호를 한 쌍으로 하여 수집하고, 이 정보에 기초하여 각 프로세스가 버퍼 캐시에 속한 페이지 프레임들을 접근하는 순서를 추적함으로써, 해당 프로세스가 접근하는 디스크 블록의 접근 순서를 알 수 있다. 이러한 접근 순서에 기초하여 기설정된 개수 이상의 페이지 프레임들이 연속적으로 접근될 경우 해당 페이지 프레임들을 순차 패턴으로 분류한다.

[0074] 또한, 패턴 분할부(110)는 버퍼 캐시에 올라와 있는 페이지의 접근, 그리고 접근하고자 하는 페이지 프레임이 원하는 데이터를 담고 있지 않은 경우에 생성되는 I/O 리퀘스트(request)를 가상 머신 모니터 시스템(100)을 통하여 모니터링 하여 블록의 연속성을 체크한다.

[0075] I/O 리퀘스트의 모니터링에 대하여 설명하면 다음과 같다.

[0076] 예를 들어, 임의의 한 페이지 프레임이 어느 한 게스트 운영체제(#1)의 버퍼 캐시에 소속되어 있다고 가정한다. 그러나, 실제로 해당 페이지 프레임은 다른 게스트 운영체제(#2)에 할당되어 사용되는 경우가 발생할 수 있다. 이러한 경우, 해당 게스트 운영체제(#1)는 자신에게 할당되어 있다고 생각하는 페이지 프레임으로 접근한 것이지만, 실제로 해당 페이지 프레임에 저장된 데이터는 다른 게스트 운영체제(#2)에 저장된 것이다. 따라서 해당 페이지 프레임에는 해당 게스트 운영체제(#1)를 통해 저장된 데이터가 아닌 다른 게스트 운영체제(#2)에 의해 저장된 데이터가 존재하며, 해당 게스트 운영체제(#1)를 통해 저장된 데이터는 가상 머신 모니터 시스템(100)에 의해서 교체(swap)된다. 이에 따라, 교체 입출력(swap in/out)시 발생하는 I/O 리퀘스트도 함께 모니터링하여 페이지 프레임에 저장된 데이터에 대한 정보를 파악함으로써, 해당 페이지 프레임의 연속성을 파악하도록 한다.

[0077] 다음, 패턴 분할부(110)가 파일 레벨에서 순차 패턴을 추출하는 방법은 다음과 같다. 도 10은 프로세스의 파일 수준에서 순차 패턴을 추출하기 위한 방법을 설명하기 위해 나타낸 도면이다. 도 11은 태스크 기반의 패턴 탐지 기법을 설명하기 위해 나타낸 도면이다.

[0078] 먼저, 각 프로세스의 파일 수준에서 순차 패턴을 추출하기 위해서는 어느 파일의 어느 위치를 접근하는지에 대한 정보를 페이지 프레임과 연결하기 위한 이벤트 상관관계(event correlation) 방법을 이용할 수 있다. 즉, 사용자 레벨에서 읽기 시스템 호출(read system call)이 발생될 경우, 패턴 분할부(110)는 읽기 시스템 호출에 포함된 함수 인자를 얻어낼 수 있다.

[0079] 이후에 메모리 접근이 발생되면 해당 메모리 프레임의 주소를 얻어 낼 수 있다. 즉, 패턴 분할부(110)는 읽기 시스템 호출의 끝남을 탐지하면 읽기 시스템 호출이 수행 중에 각 프로세스에 의해 접근된 메모리 페이지 프레임의 주소를 포함하는 페이지 프레임 접근 정보를 얻어 낼 수 있다.

[0080] 이러한 방식으로 각 게스트 운영체제 내부의 파일 접근 정보와 상기 페이지 프레임 접근 정보를 결부시켜, 각 파일의 특정 위치의 블록 데이터가 각 게스트 운영체제 내부의 페이지 프레임들 중 어느 페이지 프레임의 어느 위치에 저장되어 있는지에 대한 상관 정보를 얻을 수 있다.

[0081] 패턴 분할부(110)는 상기 상관 정보를 이용하여 태스크 기반의 패턴 탐지 기법을 수행할 수 있다. 즉 파일의 어느 오프셋(offset)에 해당하는 정보가 어느 메모리 페이지 프레임의 어느 위치에 저장되어 있는지 알 수 있으므로, 각 태스크가 실행될 때 어느 파일의 어느 오프셋 위치를 접근하였고, 실제 저장된 페이지 프레임의 주소를 연관시킬 수 있으므로, 도 11에 도시된 테이블과 같은 정보를 파악할 수 있다. 이를 통하여 각 태스크 및 해당 태스크가 접근한 각 파일에 대한 연속성을 탐지할 수 있다.

[0082] 이러한 연속성은, 파일 분할부에서 각 프로세스 또는 스레드(thread) 등의 태스크가 각각 접근하는 파일들을 내부 컨텍스트(context)의 파일 디스크립터(file descriptor)를 가지고 파일을 구분하고, 오프셋을 가지도록 파일 내부의 상대적 위치를 비교하여 판단함으로써 얻어 낼 수 있다.

[0083] 또한, 패턴 분할부는(110)는 파일 접근 시스템 호출에 의해 발생된 함수 인자에 의해 지시되는 접근 대상 데이터의 파일 내 위치와, 대상 데이터가 실제로 디스크에서 읽혀진 이후에 위치하게 되는 물리 페이지 프레임의 주소를 연결할 수 있다. 이를 위하여 패턴 분할부(110)는 도 11에 도시된 바와 같이, 테이블 엔트리를 구성할 수 있다. 테이블 엔트리는 각 태스크 아이디에 대해서 사용된 태스크 아이디, 태스크 내부 파일 아이디, 파일 접근 시작위치, 파일 접근 종료위치, 파일 접근 구간, 물리 페이지 프레임 주소로 구성될 수 있다. 파일 접근 시작위치는 시작 파일 오프셋을 의미할 수 있다. 파일 접근 종료위치는 종료 파일 오프셋을 의미할 수 있다. 파일 접근 구간은 파일 내에서 순차적으로 접근된 파일 데이터의 길이를 의미할 수 있다. 물리 페이지 프레임 주소는 파일 시작 위치로부터 파일 종료 위치까지의 파일 내 데이터를 담고 있는 실제 페이지 프레임의 주소를 의미할 수 있다. 이러한 구성이 각 태스크 아이디에 대하여 사용된 함수 인자 세트가 되며 하나의 테이블 엔트리 구성이 된다. 테이블 엔트리 정보는 엔트리의 태스크 아이디를 갖는 태스크가 접근한 파일에 따라서 즉, 파일 아이디를 기준으로 데이터의 접근 패턴을 판단할 때, 임의의 태스크가 특정 파일을 순차적, 루프 형태 혹은 비순차적으로 접근된다는 것을 파악하게 된다. 이러한 방식을 통해 파일 수준에서 데이터 접근 패턴을 물리 페이지 프레임 간의 접근 패턴으로 변환할 수 있다.

[0084] 파일 접근 시스템 호출에 의해 발생된 함수 인자에 대한 정보는 게스트 운영체제의 정보이므로 가상 머신 모니터 시스템(hypervisor)에서는 알 수 없는 정보이다. 그러나, 상술한 바와 같은 방식을 이용하면, 가상 머신 모니터 시스템에서 게스트 운영체제 내부의 각 파일들에 대해서 데이터 접근 시의 순차성 또는 비순차성에 대한 패턴 정보를 파악할 수 있으므로, 응용 프로그램 측면에서 보다 정확한 패턴 파악을 할 수 있다.

[0085] 실시예에 따른 가상 머신 모니터 시스템(100)은 순차 패턴에 속한 페이지 프레임들에 대하여 계속적인 모니터링을 수행하지만, 상기 정의한 바에 의해 순차 레퍼런스 패턴은 1회밖에 접근되지 않는 것이므로, 이로 인해 야기되는 오버헤드(overhead)는 낮다. 만일 순차 패턴이 루프 패턴이나 기타 다른 접근 패턴으로 인해서 순차 패턴이 아닌 것으로 판별될 경우에는 해당 정보를 리스트에서 삭제하고 관련 페이지 프레임들에 대한 정보를 비순차 파티션(unclassified partition)으로 이관한다. 또한, 순차 패턴에 속하지 않은 다른 버퍼 캐시 메모리들은 비순차 파티션에서 관리되도록 한다.

[0086] 도 4는 실시예에 따른 패턴 관리부(120)의 구성을 설명하기 위해 나타낸 도면이다.

[0087] 패턴 관리부(120)는 패턴 분할부(110)를 통해 추출된 순차 패턴 및 비순차 패턴에 속하는 페이지 프레임들을 각각 LRU 정책(Least Recently Used strategy)에 따라 관리한다. 이때, 순차 패턴은 여러 개의 시퀀스(sequence) 즉, 연속 페이지 프레임을 이루고, 하나의 연속 페이지 프레임을 하나의 엘리먼트(element)로 보고 연속 페이지 프레임 단위로 관리한다. 예를 들어, 임의의 페이지 프레임들이 접근될 경우 해당 연속 페이지 프레임 전체가 LRU 정책에 따라 MRU(Most Recently Used) 위치로 이동하게 된다.

[0088] 또한, 패턴 관리부(120)는 비순차 패턴에 속하는 페이지 프레임을 페이지 프레임 단위로 관리한다.

[0089] LRU 정책은 가장 오래 전에 접근되었던 연속 페이지 프레임에 속한 페이지 프레임들을 재사용 대상으로 가장 먼저 사용하기 위한 관리 방법이다. 즉, 버퍼 캐시가 페이지 프레임의 재사용을 위해 페이지 프레임을 선택할 때, 가장 오래 전에 접근되었던 연속 페이지 프레임을 대상으로 우선 선택되도록 하는 것이다. 또한, 관리 오버 헤드를 줄이기 위하여, LRU 정책은 클록 알고리즘(clock algorithm)과 같은 다른 알고리즘으로 대체될 수 있으며, 실제 가상 머신 모니터 시스템에서 각 연속 페이지 프레임은 하나의 구조체의 시퀀스에 속한 모든 페이지 프레임에 대한 정보만을 담아 관리되도록 할 수 있다.

[0090] 가상 머신 모니터 시스템(100)은 오버커밋(overcommit)된 메모리의 재할당(reallocation)을 위해서, 기본적으로 요구 페이징(demand paging)을 사용한다. 실시예에 따른 가상 머신 모니터 시스템(100)은 재할당 대상을 버퍼 캐시에 속한 메모리로만 한정하여 두 개의 패턴으로 구분하고 이에 속하는 물리 메모리 페이지에 대해서 요구

페이징 기법을 일부 차용한다.

[0091] 각 패턴에 속하는 페이지 프레임들이 재배치(relocation)되면, 재배치된 페이지 프레임에 저장되어 있던 데이터는 영구 저장공간(permanent storage)로 저장되어 보관된다. 이후, 재배치된 페이지 프레임으로 해당 프레임을 소유하고 있던 게스트 운영체제의 접근이 발생되면, 새로운 빈 페이지 프레임을 할당하고, 영구 저장공간에 저장된 데이터를 빈 페이지 프레임으로 복사한 후, 페이지 테이블의 매핑을 수정하여 데이터를 메모리로 복원함으로써 게스트 운영체제 접근에 응답하게 된다. 이때 게스트 운영체제는 자신이 해당 페이지 프레임을 접근하는 않는 동안에는 게스트 운영체제가 해당 페이지 프레임을 가지고 있다고 알고 있게 된다. 즉, 해당 페이지 프레임이 재할당되더라도 게스트 운영체제에서는 이러한 사실을 인식하지 못한다.

[0092] 재배치된 페이지 프레임을 가리키고 있는 페이지 테이블의 엔트리(entry)의 물리 주소 항목은 무효화(invalidate)되며, 가상 머신 모니터 시스템(100)만 접근 권한을 갖도록 페이지 테이블 엔트리에 대한 접근 권한을 변경하여 해당 페이지 테이블 엔트리로의 접근을 가상 머신 모니터 시스템(100)이 알도록 한다.

[0093] 따라서, 재배치된 페이지 프레임의 주소를 담고 있던 페이지 테이블의 엔트리에는 가상 주소 항목에 대응하는 물리 주소의 값이 무효화되어 있다가 어플리케이션 등에 의해서 해당 페이지 테이블 엔트리가 접근되면 패턴 관리부(120)가 이를 복원하도록 하면서 해당 위치에 새롭게 할당된 페이지 프레임의 물리 주소를 기록하여 매핑을 변경한다.

[0094] 요구 페이징의 대상을 버퍼 캐시에 속한 메모리 페이지 프레임들로 한정하였기 때문에, 버퍼 캐시에 속한 메모리에 기록되어 있던 데이터가 클린 페이지(clean page)인 경우, 원 데이터는 디스크에 이미 기록되어 있는 것을 읽어들인 것이므로, 교체(swapping) 과정을 통해서 데이터를 저장하지 않고, 해당 페이지가 원래 담고 있던 데이터의 위치를 기록한 정보만을 남기고 메모리를 재사용하게 된다. 더티 페이지(dirty page)의 경우에 있어서도, 데이터가 기록되어야 할 위치가 결정되어 있기 때문에 데이터를 교체 영역이 아니라 저장 위치에 플러시(flush)함으로써 데이터를 보정하고 사용되었던 메모리를 재활용한다. 따라서 교체 과정을 수행하지 않으면서 요구 페이징을 수행하게 된다.

[0095] 실시예에 따른 가상 머신 모니터 시스템(100)은, 메모리의 재할당이 필요하면 게스트 운영체제에서 메모리의 리클레임(reclaim)을 수행한 이후에 메모리를 확보하는 것이 아니라, 게스트 운영체제의 개입 없이 게스트 운영체제로 하여금 메모리를 아직 가지고 있다고 인식되도록 하고 메모리 선택 처리부(130)에서 재사용될 페이지들을 선택하여 필요한 만큼을 리클레임(reclaim)한다. 이때, 게스트 운영체제의 개입이 없으므로, 게스트 운영체제는 해당 메모리를 자신이 버퍼 캐시로 사용하는 것으로 인식하고 있지만, 이후에 해당 게스트 운영체제가 해당 페이지를 접근하지 않는다면 가상 머신 모니터 시스템(100)에 의해 메모리를 재사용하였다고 하더라도 게스트 운영체제에는 아무런 영향이 없다. 차후에 해당 페이지가 접근되면 강제 페이지 부재(page fault)가 발생되도록 하여 가상 머신 모니터 시스템(100)이 새 메모리에 해당 버퍼 캐시에 들어있던 데이터를 다시 읽어드려 복구하고, 이 페이지 부재 처리 과정이 모두 끝난 이후에 게스트 운영체제의 실행을 계속한다. 또한, 다시 접근되지 않아서 게스트 운영체제의 리클레임 정책에 의해 축출(eviction)되는 것으로 탐지된 경우에도, 상술한 방법과 동일하게 방법을 통해 메모리를 확보하여 게스트 운영체제가 인식하지 못하도록 한다.

[0096] 게스트 운영체제에서 사용하고 있음에도 불구하고 재사용된 버퍼 캐시에 저장되어 있던 데이터에 대한 정보는 교체 저장공간(swap storage)과 같은 자료구조를 영구 저장공간 영역에 마련하여 관리하고, 강제 페이지 부재가 발생되었을 때 영구 저장공간 영역을 참조하여 데이터를 복구한다.

[0097] 메모리 선택 처리부(130)는 복수의 가상 머신 간의 메모리 재할당을 위하여 상기 순차 패턴에 속하는 연속 페이지 프레임을 우선적으로 선택하고, 상기 선택된 페이지 프레임을 메모리를 필요로 하는 가상 머신으로 할당한다.

[0098] 먼저, 순차 패턴과 비순차 패턴에서 재할당을 위한 페이지 프레임 선택 방법은 다음과 같다.

[0099] 순차 패턴의 속성은 한번 접근된 데이터 블록들은 다시 접근되지 않는 것이므로, 이것을 재사용하더라도 이것을 사용하고 있던 게스트 운영체제에게는 강제 페이지 부재가 발생하지 않으므로 아무런 영향을 주지 않게 된다. 따라서, 순차 패턴에 속한 페이지 프레임들을 먼저 리클레임하고, 그 이후에 비순차 패턴에 속한 페이지 프레임들을 선택한다.

[0100] 도 9는 실시예에 따른 순차 패턴 내에서 희생 페이지 프레임을 선택하는 방법을 설명하기 위해 나타낸 도면이다.

[0101] 순차 패턴에서의 재사용 페이지 프레임의 선택은 다음과 같이 이루어질 수 있다.

[0102] 각각의 연속 페이지 프레임들은 순차 패턴 부분에서 LRU 정책에 의해 관리되며, 재사용 페이지 프레임은 MRU 위치에 있는 연속 페이지 프레임(901)을 재사용 연속 페이지 프레임으로 선택된다.

[0103] 선택된 연속 페이지 프레임 내에서 재사용 페이지 프레임의 선택은 동일 연속 페이지 프레임을 비슷한 순간에 접근하는 다른 접근(905)을 위하여, 해당 연속 페이지 프레임에서 가장 최근에 접근되었던 블록(906)이 위치한 MRU 위치(903)로부터 일정량의 마진(904)에 포함된 블록들을 제외한 가장 최근에 접근된 순서의 역순으로 필요한 개수만큼을 선택한다.

[0104] 즉, 임의의 단일 순차 패턴이 복수의 태스크에 의해서 순차적으로 접근되는 경우, 메모리 소유권 변경을 위해 리클레임의 대상이 될 페이지 프레임을 해당 순차 패턴 내의 MRU 위치에서 선택하게 되면, 다른 태스크에서의 순차 접근이 이루어지기 위해 해당 메모리의 복원이 이루어져야 한다. 이는 시스템의 성능을 저하시키는 원인이 되므로 이를 방지하기 위해, MRU 위치에서부터 일정 개수의 마진(margin)을 두고, 그 마진에 포함된 블록들을 제외한 페이지 프레임들을 LRU 위치 방향으로 리클레임 대상 페이지를 선택해 나가는 것이다.

[0105] 재사용 연속 페이지 프레임에 속한 페이지들이 모두 리클레임된 경우, 다음 MRU 위치에 존재하는 연속 페이지 프레임을 재사용 연속 페이지 프레임으로 선택하고, 페이지 프레임을 획득한다. 연속 페이지 프레임에 속한 페이지 프레임이 없는 경우, 비순차 패턴에서 재사용 페이지 프레임을 선택할 수 있다. 이때, 재사용 페이지 프레임의 선택은 LRU 정책에 준하여 결정될 수 있다.

[0106] 각 가상 머신이 필요로 하는 메모리의 양은 다음과 같은 방법으로 결정될 수 있다.

[0107] 각 게스트 운영체제에서 사용하는 교체 저장공간(swap storage)의 양을 가지고 결정할 수 있다. 고스트 버퍼(ghost buffer)를 이용하여 I/O 바운드 잡(bound job)에서와 같이 디스크 캐시에서의 메모리 필요성이 대두될 때 이를 알아내도록 하여 대응하는 방법이 병행될 수 있다. 즉, 각 가상 머신의 버퍼 캐시에 대해서 이미 모니터링이 이루어지고 있으나, 게스트 운영체제 내부에서 버퍼 캐시 축출(buffer cache eviction)되는 데이터들의 내용이 아닌 저장위치와 파일에서의 위치 등의 메타 정보만을 담은 고스트 버퍼를 LRU 정책으로 관리하고, 해당 고스트 버퍼에 대해서 해당 정보들을 고스트 운영체제가 다시 접근하는 횟수를 탐지하여 재접근 될 경우, 접근된 정보의 고스트 버퍼의 리스트에서 리스트 헤드로부터의 위치를 파악하여 해당 위치 별로 LRU 히스토그램을 생성하여 관리한다.

[0108] 가상 머신 모니터 시스템(100)은 LRU 히스토그램 정보를 통하여 어느 가상 머신의 버퍼 캐시가 얼마만큼의 메모리 페이지 프레임을 필요로 하는지 파악할 수 있다. 고스트 버퍼는 디스크 캐시로부터 축출(eviction)되는 메타 정보만을 담고 있으므며, 재접근되면 고스트 버퍼로부터 해당 메타 데이터를 제거한다.

[0109] LRU 히스토그램은 해당 메타 데이터가 있던 위치(리스트의 헤드 즉 버퍼 캐시에서 가까운 쪽을 기준으로 함)로부터의 거리에 해당하는 엔트리 카운트를 증가시킴으로써 얻을 수 있다.

[0110] 도 5, 도 7, 및 도 8은 실시예에 따른 메모리 재할당 과정을 설명하기 위해 나타낸 도면이다. 도 6은 기존의 메모리 재할당 과정을 나타낸 도면이다.

[0111] 첨부된 도면을 참조하여 가상 머신 간의 메모리 재할당 과정을 설명하면 다음과 같다.

[0112] 우선, 도 6을 참조하여 기존의 메모리 재할당 과정을 설명한다. 먼저, 필요한 양의 메모리를 얻기 위해서 희생 가상 머신(victim VM)들의 게스트 운영체제 리클레임 정책이 모두 실행되어 자유 메모리(free memory)를 얻고(601), 이것을 발룬 드라이버(balloon driver)가 할당 받게 되면(602), 발룬 드라이버는 이 메모리 정보를 메모리 선택 처리부(130)로 알려준다(603).

[0113] 기존의 가상 머신 모니터 시스템은 이 메모리 정보를 필요한 메모리를 얻게 될 수혜 가상 머신(beneficiary VM)의 발룬 드라이버로 전달하여(606) 해당 발룬 드라이버가 이를 해제함으로써 게스트 운영체제가 해재된 자유 페이지 프레임을 사용하도록 한다(605).

[0114] 따라서 도 5에 도시된 타임라인(507)에서와 같이 기존 방법은 희생 가상 머신이 하이퍼바이저에 의해 스케줄링되어(505) 실행된 이후에 필요한 양의 메모리를 확보할 수 있다(504).

[0115] 그러나, 실시예에 따르면, 변형된 요구 페이징을 이용하여 리클레임 작업을 게스트 운영체제가 아닌 가상 머신 모니터 시스템(100)에서 먼저 수행하고(501), 확보된 메모리를 발루밍(ballooning)을 통해 수혜 가상 머신으로

공급한다(503). 이를 본 실시예에서는 리버스 오더 발루밍(reverse-ordered ballooning)이라 지칭한다.

[0116] 보다 구체적으로, 수혜 가상 머신이 스케줄링되기(503) 이전에 가상 머신 모니터 시스템(100)이 실행(501)되기만 하면, 가상 머신 모니터 시스템(100)은 필요한 양만큼의 메모리를 바로 확보하고(707) 수혜 가상 머신의 발룬 드라이버로 할당함(706)으로써 수혜 가상 머신이 스케줄링되는 즉시 발룬 드라이버의 해제(705)에 의해 사용될 수 있도록 한다.

[0117] 이 때문에 희생 가상 머신의 스케줄링과 재할당 과정은 직접적인 관계는 없다. 수혜 가상 머신은 현재 메모리가 부족한 상황이기 때문에 I/O를 빈번하게 발생시키고, 블록(block)되는 경향이 있으므로, 다른 가상 머신들에 비해서 높은 빈도로 스케줄링되어 메모리 공급까지의 시간에서 희생 가상 머신 및 수혜 가상 머신의 스케줄링에 의한 지연을 줄일 수 있다.

[0118] 이로써 메모리 재할당 과정이 완료되며, 게스트 운영체제의 메모리 관리 정보와 실제 물리 메모리 할당 간의 불일치는, 요구 페이징 방식과 동일하게 리클레임된 메모리로의 접근이 발생할 때만 처리하거나, 게스트 운영체제의 리클레임 정책을 통해 결정된 시간에 리클레임을 수행하거나, 일정 스레시홀드(threshold) 이상 불일치 량이 증가하는 경우, 또는 단위 시간당 리클레임된 페이지 프레임으로의 접근이 스레시홀드를 넘어가는 경우에 해당 게스트 운영체제에 운용되는 발룬 드라이버로 하여금 발룬 드라이버가 메모리 할당을 요구(701)하도록 하여, 강제로 리클레임 정책의 실행을 요구하고, 빈 메모리를 얻어내고(702), 이를 발룬 드라이버를 통해서 가상 머신 모니터 시스템(100)으로 이관하여(703), 가상 머신 모니터 시스템(100)으로 하여금 메모리를 확보한 해당 게스트 운영체제의 강제 페이지 부재에 의한 메모리 필요에 대응(704)하도록 한다.

[0119] 즉, 도 8에 도시된 바와 같이, 발룬 드라이버에 의해 명시적으로 수거(801 내지 804)한 빈 메모리로 가상 머신 모니터 시스템(100)에 의해 강제 리클레임된 메모리를 대체(805 내지 806)하도록 하여 게스트 운영체제가 알고 있는 메모리 양과 가상 머신 모니터 시스템(100)에서 관리하는 메모리 양의 차이를 줄이도록 한 것이다.

[0120] 확보된 메모리를 공급받는 수혜 가상 머신은 버퍼 캐시 미스(buffer cache miss)에 의한 I/O 발생 횟수를 카운트하여, 이 값을 최소화시킬 수 있는 배분값을 찾거나, 사용된 시간에 비해서 캐시 미스의 횟수가 기준값 이상으로 빠르게 증가하는지를 모니터링하여 결정하며, 이는 캐시 미스로 인해 발생되는 I/O 리퀘스트를 카운트함으로써 추가적인 오버헤드 없이 얻을 수 있다. 이때 최적의 배분값을 찾는 것은 오랜 시간이 소요되며 항상 해법을 찾을 수 있는 것은 아니므로, 미리 지정된 개수만큼을 주는 방식을 통해 근사하도록 하고, 주어진 메모리의 양은 가상 머신이 구동되는 시점에서 할당된 물리 메모리의 용량을 넘지 않도록 한다.

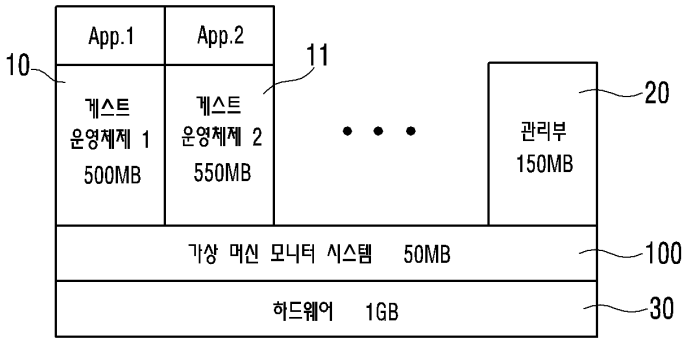
[0121] 실시예에 따르면, 물리 메모리의 용량보다 많은 양의 메모리를 다수의 가상 머신에게 할당하여 동작하는 시스템에 있어서, 부족한 메모리를 다른 가상 머신으로 이동시킬 때 이를 최소의 시간 지연으로 즉시 조정할 수 있도록 함으로써, 전체 시스템 성능을 각 가상 머신에 대해 체결되어 있는 SLA(service level agreement)를 준수하면서도 극대화시킬 수 있으므로, 시스템의 자원 활용도를 높이고, 보다 적은 수의 하드웨어를 이용하여 보다 많은 수의 가상 머신을 실행시킬 수 있다.

[0122] 이상에서 보는 바와 같이, 본 발명이 속하는 기술 분야의 당업자는 본 발명이 그 기술적 사상이나 필수적 특징을 변경하지 않고서 다른 구체적인 형태로 실시 될 수 있다는 것을 이해할 수 있을 것이다.

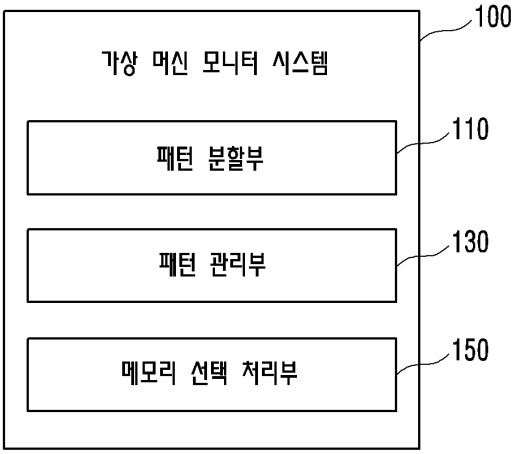
[0123] 그러므로, 이상에서 기술한 실시 예는 모든 면에서 예시적인 것이며 한정적인 것이 아닌 것으로 이해해야만 하고, 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 등가개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

도면

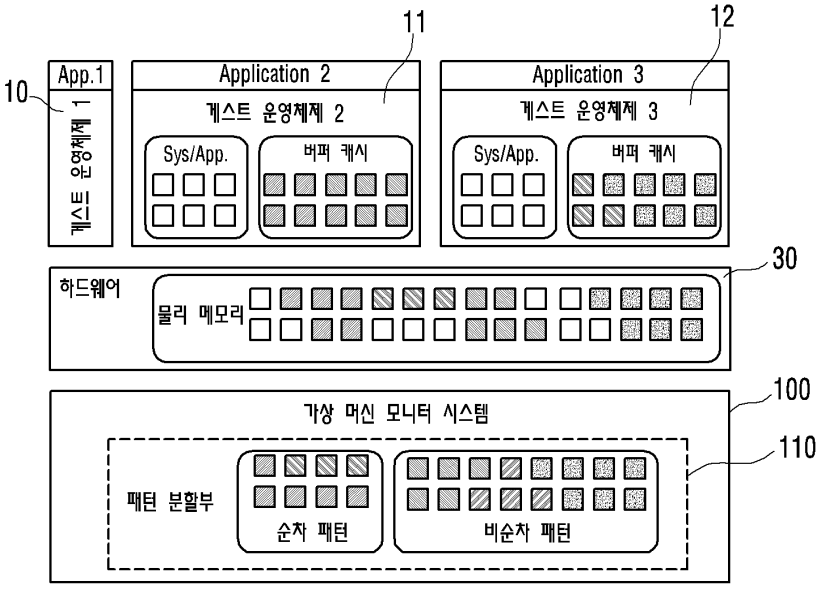
도면1



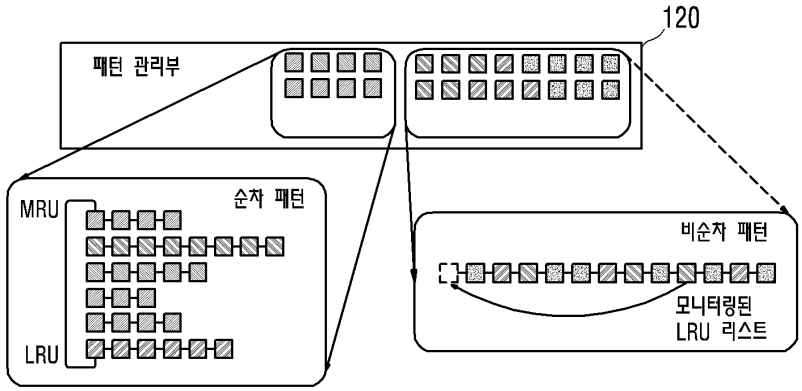
도면2



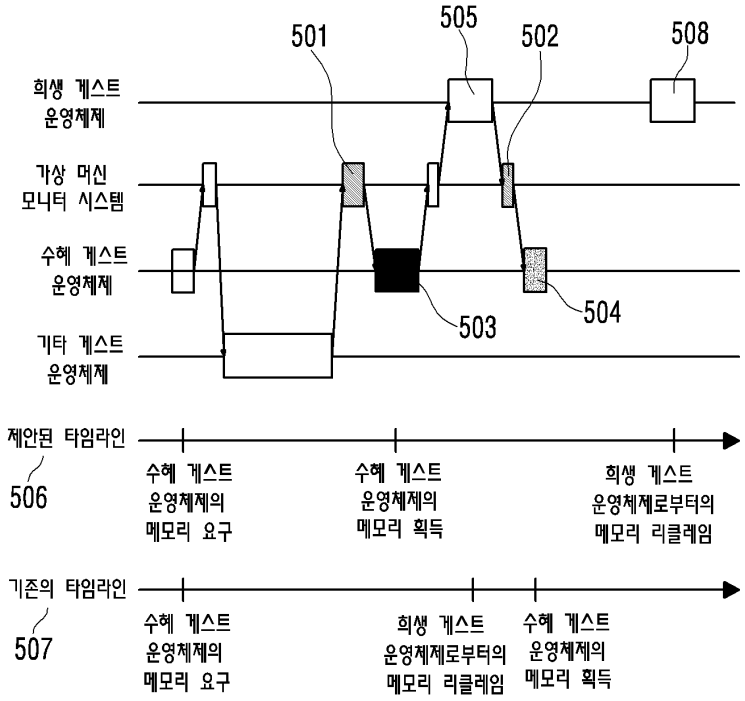
도면3



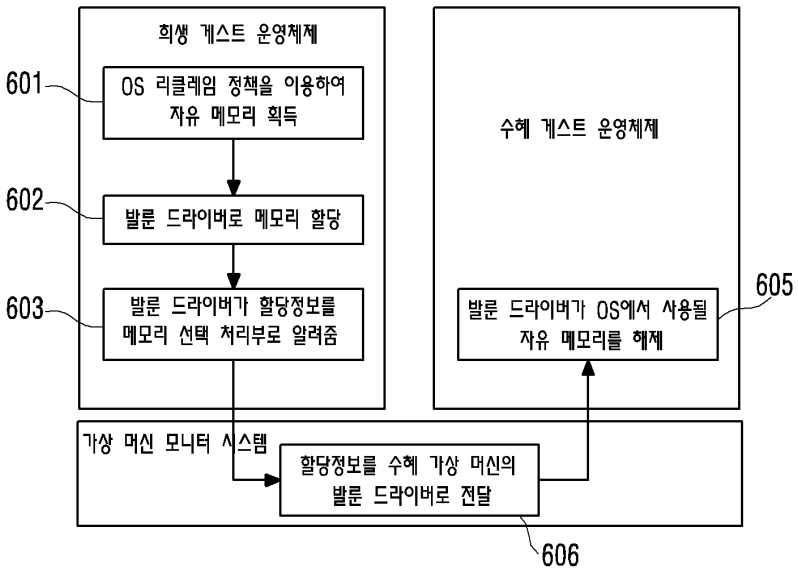
도면4


120
패턴 관리부
MRU
LRU
순차 패턴
비순차 패턴
모니터링된
LRU 리스트

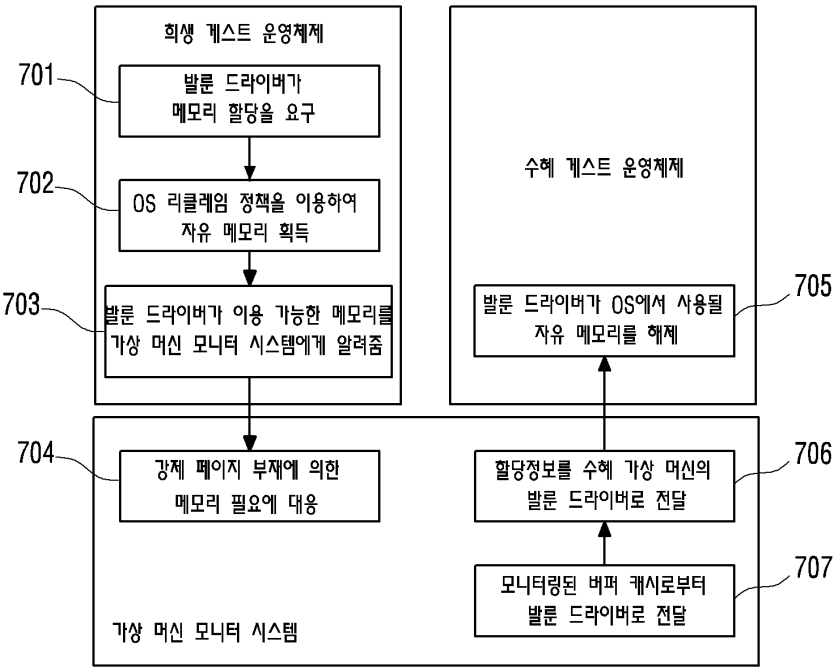
도면5


501
505
502
508
희생 게스트
운영체제
가상 머신
모니터 시스템
수혜 게스트
운영체제
기타 게스트
운영체제
503
504
제안된 타임라인
506
수혜 게스트
운영체제의
메모리 요구
수혜 게스트
운영체제의
메모리 획득
희생 게스트
운영체제로부터의
메모리 리클레임
기존의 타임라인
507
수혜 게스트
운영체제의
메모리 요구
희생 게스트
운영체제로부터의
메모리 리클레임
수혜 게스트
운영체제의
메모리 획득

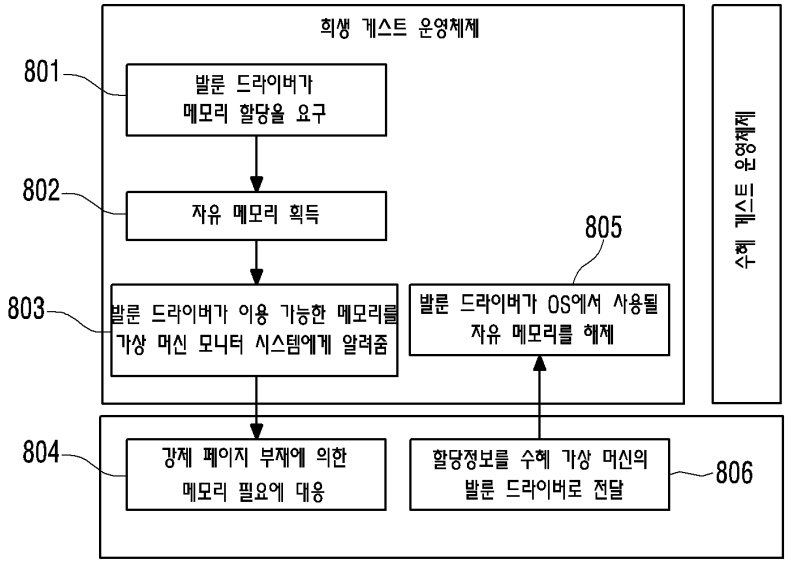
도면6


희생 게스트 운영체제
601
OS 리클레임 정책을 이용하여 자유 메모리 획득
602
발룬 드라이버로 메모리 할당
603
발룬 드라이버가 할당정보를 메모리 선택 처리부로 알려줌
수혜 게스트 운영체제
605
발룬 드라이버가 OS에서 사용될 자유 메모리를 해제
가상 머신 모니터 시스템
할당정보를 수혜 가상 머신의 발룬 드라이버로 전달
606

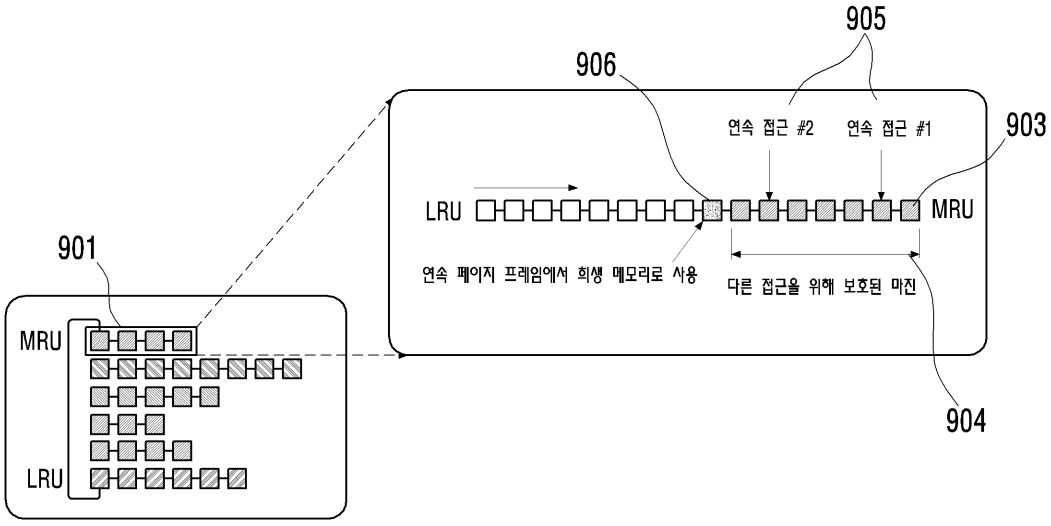
도면7


희생 게스트 운영체제
701
발룬 드라이버가 메모리 할당을 요구
702
OS 리클레임 정책을 이용하여 자유 메모리 획득
703
발룬 드라이버가 이용 가능한 메모리를 가상 머신 모니터 시스템에게 알려줌
704
강제 페이지 부재에 의한 메모리 필요에 대응
수혜 게스트 운영체제
705
발룬 드라이버가 OS에서 사용될 자유 메모리를 해제
706
할당정보를 수혜 가상 머신의 발룬 드라이버로 전달
707
모니터링된 버퍼 캐시로부터 발룬 드라이버로 전달
가상 머신 모니터 시스템

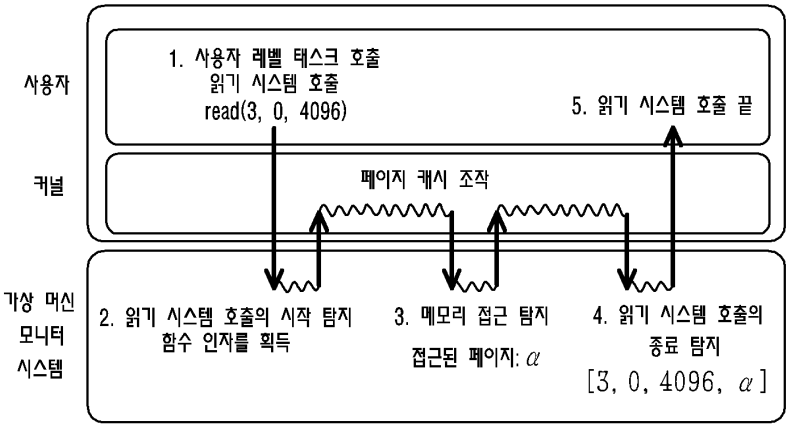
도면8


희생 게스트 운영체제
801
발룬 드라이버가
메모리 할당을 요구
802
자유 메모리 획득
805
803
발룬 드라이버가 이용 가능한 메모리를
가상 머신 모니터 시스템에게 알려줌
발룬 드라이버가 OS에서 사용될
자유 메모리를 해제
수혜 게스트 운영체제
804
강제 페이지 부재에 의한
메모리 필요에 대응
할당정보를 수혜 가상 머신의
발룬 드라이버로 전달
806

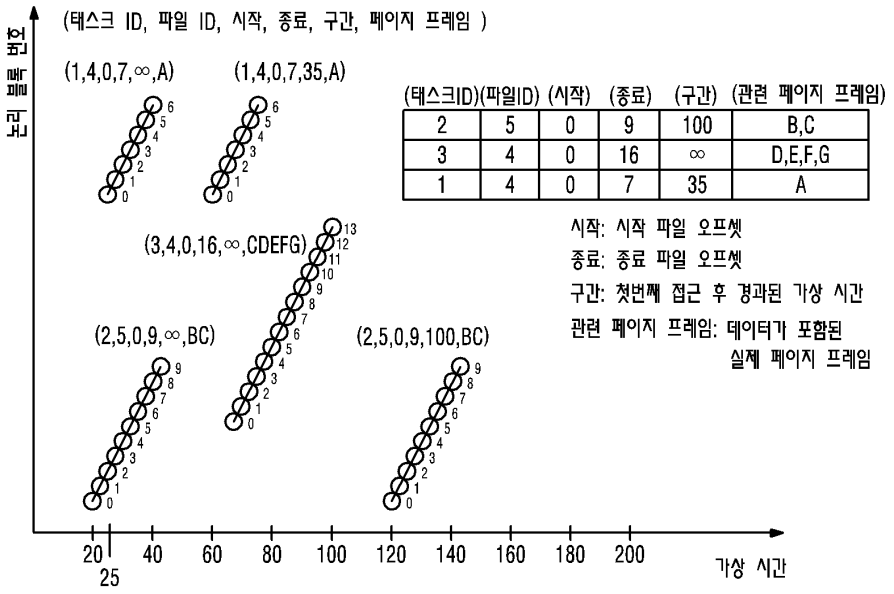
도면9


905
906
903
연속 접근 #2
연속 접근 #1
LRU
MRU
901
연속 페이지 프레임에서 희생 메모리로 사용
다른 접근을 위해 보호된 마진
MRU
LRU
904

도면10


사용자
1. 사용자 레벨 태스크 호출
읽기 시스템 호출
read(3, 0, 4096)
5. 읽기 시스템 호출 끝
커널
페이지 캐시 조작
가상 머신
모니터
시스템
2. 읽기 시스템 호출의 시작 탐지
함수 인자를 획득
3. 메모리 접근 탐지
접근된 페이지: α
4. 읽기 시스템 호출의
종료 탐지
[3, 0, 4096, α]

도면11



(태스크ID)	(파일ID)	(시작)	(종료)	(구간)	(관련 페이지 프레임)
2	5	0	9	100	B,C
3	4	0	16	∞	D,E,F,G
1	4	0	7	35	A

시작: 시작 파일 오프셋
종료: 종료 파일 오프셋
구간: 첫번째 접근 후 경과된 가상 시간
관련 페이지 프레임: 데이터가 포함된 실제 페이지 프레임