



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0089150
(43) 공개일자 2021년07월15일

(51) 국제특허분류(Int. Cl.)
G06F 12/1009 (2016.01) G06F 12/02 (2018.01)
G06F 12/0811 (2016.01) G06F 12/0882 (2016.01)
G06F 12/1036 (2016.01) G06F 12/1045 (2016.01)
G06F 12/109 (2016.01) G06F 12/126 (2016.01)
G06F 9/455 (2018.01)
(52) CPC특허분류
G06F 12/1009 (2013.01)
G06F 12/023 (2013.01)
(21) 출원번호 10-2021-7012320
(22) 출원일자(국제) 2019년11월14일
심사청구일자 없음
(85) 번역문제출일자 2021년04월23일
(86) 국제출원번호 PCT/US2019/061345
(87) 국제공개번호 WO 2020/106533
국제공개일자 2020년05월28일
(30) 우선권주장
16/198,620 2018년11월21일 미국(US)
16/423,137 2019년05월27일 미국(US)

(71) 출원인
마이크로소프트 테크놀로지 라이선싱, 엘엘씨
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원
마이크로소프트 웨이
(72) 발명자
마크 예브게니
미국 워싱턴주 98052-6399 레드몬드 원 마이크로
소프트 웨이 마이크로소프트 테크놀로지
라이선싱, 엘엘씨
이이균 메멧
미국 워싱턴주 98052-6399 레드몬드 원 마이크로
소프트 웨이 마이크로소프트 테크놀로지
라이선싱, 엘엘씨
키산 아룬 유
미국 워싱턴주 98052-6399 레드몬드 원 마이크로
소프트 웨이 마이크로소프트 테크놀로지
라이선싱, 엘엘씨
(74) 대리인
김태홍, 김진희

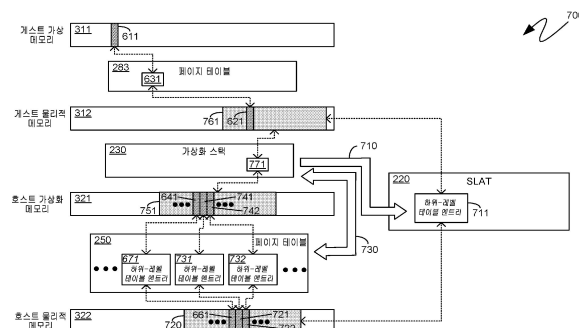
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 호스트 컴퓨팅 디바이스 가상 메모리에 의해 지원되는 가상 머신 메모리의 더 빠른 액세스

(57) 요약

호스트 컴퓨팅 디바이스 상에서 실행되는 하나 이상의 프로세스에 할당된 가상 메모리에 의해 가상 머신 환경의 게스트 물리적 메모리가 지원되는 메모리 액세스의 일부로서 제2 계층 주소 테이블(SLAT)의 계층적 레벨이 순회되는 속도를 증가시키기 위해, SLAT 내의 하나 이상의 계층적 레벨의 테이블은 스킵되거나 참조되지 않을 수 있다. SLAT는 계층적으로 상위-레벨의 테이블에서 메모리 상관 관계로 파플레이팅될 수 있지만, 가상 메모리의 호스트 컴퓨팅 디바이스의 제어를 지원하는 호스트 컴퓨팅 디바이스의 페이지 테이블은 계층적으로 최저 테이블 레벨에서 대응하는 연속적인 메모리 상관 관계의 세트를 유지할 수 있으며, 이에 의해 호스트 컴퓨팅 디바이스가 더 작은 메모리 청크를 페이징 아웃하게 하거나 조작할 수 있게 한다. 이러한 조작이 발생하면, SLAT는 계층적으로 최저 테이블 레벨에서 메모리 상관 관계로 다시 파플레이팅될 수 있다.

대표도



(52) CPC특허분류

G06F 12/0811 (2013.01)
G06F 12/0882 (2013.01)
G06F 12/1036 (2013.01)
G06F 12/1063 (2013.01)
G06F 12/109 (2013.01)
G06F 12/126 (2013.01)
G06F 9/45558 (2013.01)
G06F 2009/45583 (2019.08)
G06F 2212/1024 (2013.01)

명세서

청구범위

청구항 1

컴퓨터 메모리의 액세스 속도를 증가시키는 방법에 있어서,

가상 머신 컴퓨팅 환경에서 실행되는 제1 프로세스로부터 제1 메모리 범위로 향하는 제1 메모리 액세스를 검출하는 단계;

상기 제1 메모리 액세스를 완료하기 위한 전제 조건으로서, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리를 생성하는 단계 - 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블은 호스트 물리적 메모리 주소들을 게스트 물리적 메모리 주소들과 상관시키며, 상기 제1 엔트리는 상기 제1 엔트리에 의해 식별되는 제2 메모리 범위가 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 상기 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있고, 상기 제2 메모리 범위는 상기 제1 메모리 범위보다 큼 - ; 및

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 상기 제1 엔트리를 생성하는 것에 응답하여, 계층적으로 배열된 페이지 테이블에서 제1 복수의 엔트리를 사용된 것으로 마킹하는 단계 - 상기 계층적으로 배열된 페이지 테이블은 상기 호스트 물리적 메모리 주소들을 호스트 가상 메모리 주소들과 상관시키고, 상기 제1 복수의 엔트리는 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 상기 제1 엔트리와 동일한 상기 제2 메모리 범위를 총괄하여 참조하고, 상기 제1 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있음 - 를 포함하고;

상기 게스트 물리적 메모리 주소들은 상기 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스들에 의해 물리적 메모리에 대한 주소들인 것으로 인식되고,

상기 호스트 물리적 메모리 주소들은 상기 가상 머신 컴퓨팅 환경을 호스팅하는 호스트 컴퓨팅 디바이스의 실제 물리적 메모리에 대한 주소들이고,

상기 호스트 가상 메모리 주소들은 상기 호스트 컴퓨팅 디바이스 상에서 실행되는 메모리 관리자에 의해 제공되는 가상 메모리에 대한 주소들이고, 상기 메모리 관리자는 상기 가상 메모리를 제공하기 위해 상기 계층적으로 배열된 페이지 테이블을 활용하고,

상기 게스트 물리적 메모리 주소들에 의해 식별되는 게스트 물리적 메모리는 상기 호스트 가상 메모리 주소들의 일부에 의해 식별되는 호스트 가상 메모리의 일부에 의해 지원되는 것인, 컴퓨터 메모리의 액세스 속도를 증가시키는 방법.

청구항 2

제1항에 있어서,

상기 제1 메모리 액세스가 완료된 후, 상기 계층적으로 배열된 페이지 테이블의 제1 복수 엔트리의 제1 서브세트가 원래 대응하는 호스트 물리적 메모리 주소들에 저장되었고 후속하여 비휘발성 저장 매체로 페이징 아웃(paging out)된 데이터를 가졌음을 검출하는 단계;

상기 검출에 응답하여, 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 상기 제1 엔트리를 무효화하는 단계; 및

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 상기 제1 엔트리 대신에, 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 복수의 엔트리를 생성하는 단계 - 상기 제2 복수의 엔트리는 상기 제1 엔트리와 동일한 제2 메모리 범위의 적어도 일부를 총괄하여 참조하고, 상기 제2 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있음 - 를 더 포함하는, 컴퓨터 메모리의 액세스 속도를 증가시키는 방법.

청구항 3

제1항에 있어서,

상기 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역들을 상기 호스트 물리적 메모리의 단일의 큰 페이지 크기 영역으로 어셈블링하는 단계를 더 포함하는, 컴퓨터 메모리의 액세스 속도를 증가시키는 방법.

청구항 4

제3항에 있어서,

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 엔트리를 생성하는 단계 - 상기 제2 엔트리는 상기 제2 엔트리에 의해 식별되는 제3 메모리 범위가 상기 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 상기 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있고, 상기 제3 메모리 범위는 상기 호스트 물리적 메모리의 상기 제1 복수의 연속적인 작은 페이지 크기 영역이 어셈블링된 상기 호스트 물리적 메모리의 상기 단일의 큰 페이지 크기 영역을 참조함 - ; 및

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 상기 제2 엔트리를 생성하는 것에 응답하여, 상기 계층적으로 배열된 페이지 테이블의 제2 복수의 엔트리를 사용된 것으로 마킹하는 단계 - 상기 제2 복수의 엔트리는 상기 호스트 물리적 메모리의 상기 단일의 큰 페이지 크기 영역으로 어셈블링된 상기 호스트 물리적 메모리의 상기 제1 복수의 연속적인 작은 페이지 크기 영역을 참조함 - 를 더 포함하는, 컴퓨터 메모리의 액세스 속도를 증가시키는 방법.

청구항 5

제4항에 있어서,

상기 호스트 물리적 메모리의 제2 복수의 작은 페이지 크기 영역으로부터 상기 제1 복수의 연속적인 작은 페이지 크기 영역의 적어도 일부로 데이터를 복제하는 단계 - 상기 제2 복수의 작은 페이지 크기 영역은 적어도 파티(party) 불연속적임 - ; 및

상기 호스트 물리적 메모리의 상기 제2 복수의 작은 페이지 크기 영역을 참조한 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 복수의 엔트리를 무효화하는 단계 - 상기 제2 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있음 - 를 더 포함하고,

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 생성된 제2 엔트리는 무효화된 상기 제2 복수의 엔트리 대신 활용되는 것인, 컴퓨터 메모리의 액세스 속도를 증가시키는 방법.

청구항 6

제4항에 있어서,

상기 호스트 물리적 메모리의 하나 이상의 작은 페이지 크기 영역의 제2 세트로부터 상기 제1 복수의 연속적인 작은 페이지 크기 영역의 적어도 일부로 데이터를 복제하는 단계; 및

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 복수의 엔트리를 무효화하는 단계 - 상기 제2 복수의 엔트리는: (1) 상기 호스트 물리적 메모리의 상기 하나 이상의 작은 페이지 크기 영역의 상기 제2 세트를 참조하는 엔트리들의 제1 서브세트 및 (2) 상기 호스트 물리적 메모리의 상기 제1 복수의 연속적인 작은 페이지 크기 영역의 적어도 일부를 참조하는 엔트리들의 제2 서브세트를 모두 포함하고, 상기 제2 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있음 - 를 더 포함하고,

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 생성된 상기 제2 엔트리는 무효화된 상기 제2 복수의 엔트리 대신 활용되는, 컴퓨터 메모리의 액세스 속도를 증가시키는 방법.

청구항 7

제3항에 있어서,

상기 호스트 물리적 메모리의 상기 제1 복수의 연속적인 작은 페이지 크기 영역을 어셈블링하는 단계는 상기 제1 복수의 연속적인 작은 페이지 크기 영역의 일부로부터 상기 제1 복수의 연속적인 작은 페이지 크기 영역과 다른 상기 호스트 물리적 메모리의 다른 작은 페이지 크기 영역들로 데이터를 복제하는 단계를 포함하는, 컴퓨터

메모리의 액세스 속도를 증가시키는 방법.

청구항 8

제1항에 있어서,

상기 제2 메모리 범위의 페이징(paging)을 방지하는 단계를 더 포함하는, 컴퓨터 메모리의 액세스 속도를 증가시키는 방법.

청구항 9

실행시 컴퓨팅 디바이스로 하여금 제1항의 단계들을 수행하게 하는 컴퓨터-실행 가능 명령어들을 포함하는, 하나 이상의 컴퓨터-판독 가능 저장 매체.

청구항 10

컴퓨팅 디바이스에 있어서,

하나 이상의 중앙 처리 장치;

랜덤 액세스 메모리(RAM: random access memory); 및

하나 이상의 컴퓨터-판독 가능 매체를 포함하고, 상기 하나 이상의 컴퓨터-판독 가능 매체는:

상기 컴퓨팅 디바이스에 의해 실행될 때, 상기 컴퓨팅 디바이스로 하여금 호스트 가상 메모리 주소들을 상기 RAM 상에서 위치들을 식별하는 호스트 물리적 메모리 주소들로 트랜슬레이팅하기 위하여 계층적으로 배열된 페이지 테이블을 참조하는 메모리 관리자를 제공하게 하는 컴퓨터-실행 가능 명령어들의 제1 세트;

상기 컴퓨팅 디바이스에 의해 실행될 때, 상기 컴퓨팅 디바이스로 하여금 가상 머신 컴퓨팅 환경을 제공하게 하는 컴퓨터-실행 가능 명령어들의 제2 세트 - 상기 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스들은 게스트 물리적 메모리 주소들을 물리적 메모리에 대한 주소들로 인식하고, 상기 게스트 물리적 메모리 주소들에 의해 식별되는 추가 게스트 물리적 메모리는 상기 호스트 가상 메모리 주소들의 일부에 의해 식별되는 호스트 가상 메모리의 일부에 의해 지원됨 - ; 및

상기 컴퓨팅 디바이스에 의해 실행될 때, 상기 컴퓨팅 디바이스로 하여금:

상기 가상 머신 컴퓨팅 환경에서 실행되는 제1 프로세스로부터 제1 메모리 범위로 향하는 제1 메모리 액세스를 검출하게 하고;

상기 제1 메모리 액세스를 완료하기 위한 전제 조건으로서, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리를 생성하게 하고 - 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블은 호스트 물리적 메모리 주소들을 게스트 물리적 메모리 주소들과 상관시키며, 상기 제1 엔트리는 상기 제1 엔트리에 의해 식별되는 제2 메모리 범위가 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 상기 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있음 - ;

상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 상기 제1 엔트리를 생성하는 것에 응답하여, 계층적으로 배열된 페이지 테이블에서 제1 복수의 엔트리를 사용된 것으로 마킹하게 하는 - 상기 제1 복수의 엔트리는 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 상기 제1 엔트리와 동일한 상기 제2 메모리 범위를 총괄하여 참조하고, 상기 제1 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있음 - 컴퓨터-실행 가능 명령어들의 제3 세트를 포함하는, 컴퓨팅 디바이스.

청구항 11

제1항에 있어서,

상기 가상 머신 컴퓨팅 환경에서 실행되는 상기 제1 프로세스로부터 제2 메모리 액세스를 검출하는 단계 - 상기 제2 메모리 액세스는 상기 제1 메모리 범위와 다른 제3 메모리 범위로 함 - ; 및

상기 계층적으로 배열된 페이지 테이블의 상기 제1 복수의 엔트리의 제2 서브세트를 참조하여 상기 제2 메모리 액세스를 충족시키는 단계를 더 포함하고;

상기 제1 메모리 액세스는 상기 계층적으로 배열된 페이지 테이블의 상기 제1 복수의 엔트리의 제1 서브세트를

참조하여 충족되었고, 상기 제1 서브세트는 상기 계층적으로 배열된 페이지 테이블에서 상기 제2 서브세트와 다른 엔트리들을 포함하는, 방법.

청구항 12

제2항에 있어서,

상기 제2 복수의 엔트리는 상기 제1 프로세스에 의해 이전에 액세스되었고 페이징 아웃되지 않았던 상기 제2 메모리 범위의 부분들을 참조하는, 방법.

청구항 13

제3항에 있어서,

상기 제1 메모리 액세스를 검출한 후, 그리고 상기 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 상기 제1 엔트리를 생성하기 전에 상기 어셈블링이 발생하는, 방법.

청구항 14

제7항에 있어서,

상기 제1 복수의 연속적인 작은 페이지 크기 영역 중 일부로부터 상기 다른 작은 페이지 크기 영역들로 상기 데이터를 복제하는 단계는 상기 제1 복수의 연속적인 작은 페이지 크기 영역의 조각화가 조각화 임계값 미만인 경우에만 수행되는, 방법.

청구항 15

제8항에 있어서,

상기 제2 메모리 범위의 하나 이상의 부분이 액세스 빈도 임계값보다 덜 빈번하게 액세스되는 경우, 상기 제2 메모리 범위의 페이징의 방식을 제거하는 단계를 더 포함하는, 방법.

발명의 설명

기술 분야

배경 기술

[0001]

모뎀 컴퓨팅 디바이스는 랜덤 액세스 메모리(RAM: random access memory)의 형태로 고속 휘발성 메모리로부터 컴퓨터-실행 가능 명령을 실행함으로써 동작하며, 이러한 컴퓨터 실행 가능 명령의 실행은 종종 RAM으로부터 데이터를 읽는 것을 수반한다. 비용, 물리적 크기 제한, 전력 요건 및 다른 유사한 제약으로 인해, 컴퓨팅 디바이스는 통상적으로 이러한 컴퓨팅 디바이스 상에서 통상적으로 실행되는 프로세스에 필요한 것보다 더 적은 RAM을 포함한다. 이러한 제약을 수용하기 위해, 가상 메모리가 활용되며, 이에 의해 컴퓨팅 디바이스 상에서 실행되는 프로세스에 이용 가능한 것으로 보이는 메모리는 물리적 메모리 회로가 제공하는 것보다 크다. 가상 메모리와 물리적 메모리 간의 관계는 통상적으로 하나 이상의 메모리 관리자에 의해 관리되며, 이는 물리적 메모리 또는 일부 형태의 저장 매체 중 어느 하나에 하나 이상의 가상 메모리 주소와 대응 데이터의 위치 간의 관계를 기술하는 정보가 있는 "페이지 테이블"을 구현, 유지 및/또는 참조한다. 현대의 컴퓨팅 디바이스 및 그 상에서 실행되는 프로세스와 관련된 메모리의 양을 수용하기 위해, 현대의 페이지 테이블은 통상적으로 복수의 계층 레벨의 테이블로 구성되며, 계층적으로 상위-레벨 테이블은 각각 다른 계층적으로 하위-레벨 테이블을 식별하는 엔트리를 포함하고, 계층적으로 최저-레벨 테이블은 추가 테이블을 식별하지 않고 메모리 주소 자체를 식별하는 엔트리를 포함한다.

[0002]

컴퓨팅 디바이스에 의해 실행될 수 있는 프로세스 중에는 컴퓨팅 디바이스의 기본 하드웨어를 가상화하거나 추상화하는 프로세스가 있다. 이러한 프로세스는 이러한 가상 머신에 의해 제공되는 가상화된 컴퓨팅 컨텍스트 내에서 실행되는 프로세스에 대한 완전한 기본 컴퓨팅 디바이스를 시뮬레이팅할 수 있는 가상 머신을 포함한다. 하이퍼바이저(hypervisor) 또는 유사한 컴퓨터-실행 가능 명령 세트는 이러한 하이퍼바이저를 호스팅하는 물리적 컴퓨팅 디바이스의 기본 하드웨어를 가상화하거나 추상화함으로써 가상 머신의 제공을 용이하게 할 수 있다.

하이퍼바이저는 또한 앞서 언급한 페이지 테이블과 유사한 방식으로 계층적으로 배열될 수 있는 제2 계층 주소 테이블(SLAT: Second Layer Address Table)을 유지할 수 있다. SLAT는 기본 컴퓨팅 하드웨어의 가상화가 하이퍼바이저에 의해 용이하게 되는 가상 머신이 컨텍스트 내에서 실행되는 프로세스를 포함하여 하이퍼바이저의 탑(top) 상에서 실행되는 프로세스에 대해 물리적 메모리 위치로 보이는 하나 이상의 메모리 주소와 실제 물리적 메모리 자체의 메모리 위치 간의 관계를 기술하는 정보를 유지할 수 있다.

[0003] 예를 들어, 가상 머신의 컨텍스트 내에서 실행되는 프로세스가 메모리에 액세스할 때, 2개의 다른 룩업이 수행될 수 있다. 하나의 룩업은 요청된 가상 메모리 주소를 물리적 메모리 주소와 상관시키기 위해 가상 머신 컨텍스트 자체 내에서 수행될 수 있다. 이러한 룩업은 가상 머신 컨텍스트 자체 내에서 수행되므로, 식별된 물리적 메모리 주소는 가상 머신의 컨텍스트 내에서 실행되는 프로세스에 의해 인식되는 물리적 메모리 주소일 뿐이다. 이러한 룩업은 가상 머신의 컨텍스트 내에서 실행되는 메모리 관리자에 의해 수행될 수 있으며, 가상 머신의 컨텍스트 내에 존재하는 페이지 테이블을 참조하여 이루어질 수 있다. 그 후, 제2 룩업이 가상 머신의 컨텍스트 외부에서 수행될 수 있다. 보다 구체적으로, 제1 룩업에 의해 식별된 물리적 메모리 주소(가상 머신의 컨텍스트 내)는 실제 물리적 메모리 주소와 상관될 수 있다. 이러한 제2 룩업은 SLAT를 참조하는 컴퓨팅 디바이스의 하나 이상의 프로세싱 유닛을 수반할 수 있으며, SLAT는 인식된 물리적 메모리 주소를 실제 물리적 메모리 주소와 상관시킬 수 있다.

[0004] 나타낸 바와 같이, 페이지 테이블과 SLAT 모두는 다른 계층적 레벨의 테이블의 계층적 배열일 수 있다. 따라서, 테이블 룩업의 성능은 페이지 테이블을 참조하여 메모리 관리자에 의해 실행되든지, 또는 SLAT를 참조하여 하이퍼바이저에 의해 수행되든지간에 최저 계층적 레벨 테이블에 도달될 때까지, 최고 계층적 테이블 레벨 내에서 적절한 테이블 엔트리를 결정하고, 해당 테이블 엔트리에 의해 식별되는 계층적으로 하위-레벨 테이블을 참조하고, 계층적으로 하위-레벨 테이블 내에서 적절한 테이블 엔트리를 결정하고, 해당 테이블 엔트리에 의해 식별되는 계층적으로 여전히 하위-레벨 테이블을 참조하는 것 등을 수반할 수 있으며, 이 때 해당 최저 계층 레벨 테이블의 개별 엔트리는 여전히 추가 테이블을 식별하는 것으로 가정되는 메모리 자체의 하나 이상의 특정 주소 또는 주소 범위를 식별한다. 계층적으로 하위-레벨 테이블에 대한 각각의 참조는 프로세서 사이클을 소비하고 메모리 액세스의 지속 시간을 증가시킨다.

[0005] 예를 들어, 가상 머신의 컨텍스트 내에서 실행되는 프로세스로부터의 메모리 액세스의 경우, 이러한 메모리 액세스의 지속 시간은 가상 머신의 컨텍스트에서 메모리 관리자에 의해 수행되는 페이지 테이블의 계층적 레벨의 순회(traversing)와 또한 하이퍼바이저에 의해 수행되는 SLAT의 계층적 레벨의 순회 모두를 포함할 수 있다. SLAT를 참조하여 하이퍼바이저에 의해 수행된 룩업에 의해 도입된 추가 지연은 가상 머신의 컨텍스트 내에서 실행되는 프로세스 또는 실제로 하이퍼바이저를 통해 메모리에 액세스하는 임의의 프로세스로부터의 메모리 액세스를 보다 직접적으로 메모리에 액세스하는 프로세스에 비해 더욱 비효율적으로 만든다. 이러한 비 효율성은 사용자가 보안의 이점 및 하이퍼바이저를 통해 메모리에 액세스하여 부여되는 다른 이점을 달성하지 못하게 할 수 있다.

발명의 내용

[0006] 호스트 컴퓨팅 디바이스 상에서 실행되는 가상 머신 프로세스에 의한 메모리 활용의 효율성을 증가시키기 위해, 이러한 가상 머신 프로세스는 호스트의 가상 메모리에 의해 지원될 수 있으며, 이에 의해 이러한 가상 머신 프로세스에 의해 소비되는 메모리에 종래의 가상 메모리 효율성을 적용할 수 있다. 가상 머신 환경의 게스트 물리적 메모리가 호스트 컴퓨팅 디바이스 상에서 실행되는 하나 이상의 프로세스에 할당된 가상 메모리에 의해 지원되는 환경에서 제2 계층 주소 테이블(SLAT)의 계층적 레벨이 메모리 액세스의 일부로서 순회되는 속도를 증가시키기 위해, SLAT 내의 하나 이상의 계층적 레벨의 테이블이 스킵(skip)되거나 그렇지 않으면 참조되지 않을 수 있으며, 이에 의해 보다 효율적인 SLAT 순회 및 보다 효율적인 메모리 액세스로 귀결된다. SLAT는 계층적으로 상위-레벨의 테이블에서 메모리 상관으로 파플레이팅(populating)될 수 있지만, 호스트 컴퓨팅 디바이스의 가상 메모리 제공을 지원하는 호스트 컴퓨팅 디바이스의 페이지 테이블은 계층적으로 최저 테이블 레벨에서 대응하는 연속적인 메모리 상관 세트를 유지할 수 있으며, 이에 의해 호스트 컴퓨팅 디바이스가 더 작은 메모리 청크(chunk)를 페이징 아웃(paging out)하거나 다르게 조작할 수 있게 한다. 이러한 조작이 발생하면, SLAT는 계층적으로 최저 테이블 레벨에서 메모리 상관으로 다시 파플레이팅될 수 있다. 반대로, 호스트가 작은 페이지의 충분히 큰 연속 세트를 재어셈블링할 수 있는 경우, SLAT는 계층적으로 상위-레벨의 테이블에서 상관으로 다시 파플레이팅될 수 있으며, 이에 의해 다시 더 효율적인 SLAT 순회와 더욱 효율적인 메모리 액세스로 귀결된다. 이러한 방식으로, 호스트 컴퓨팅 디바이스의 가상 메모리로 가상 머신 프로세스를 지원하는 메모리

활용 효율성의 이점을 유지하면서 보다 효율적인 SLAT 순회가 달성될 수 있다.

[0007] 이러한 개요는 상세한 설명에서 추가로 후술되는 단순화된 형태로 개념의 선택을 소개하기 위해 제공된다. 이러한 개요는 청구된 주제의 핵심 특징이나 본질적인 특징을 식별하려고 의도된 것도 아니고, 청구된 주제의 범위를 제한하는 데 사용하려고 의도된 것도 아니다.

[0008] 추가적인 특징 및 이점은 첨부 도면을 참조하여 진행되는 이하의 상세한 설명으로부터 명백해질 것이다.

도면의 간단한 설명

[0009] 이하의 상세한 설명은 첨부된 도면과 관련하여 받아들일 때 가장 잘 이해될 수 있다.

도 1은 예시적인 컴퓨팅 디바이스의 블록도이다.

도 2는 예시적인 하드웨어 및 소프트웨어 컴퓨팅 시스템의 시스템도이다.

도 3은 본원에서 이용되는 용어를 나타내는 블록도이다.

도 4는 호스트 가상 메모리로 게스트 물리적 메모리를 지원하기 위한 예시적인 방법을 나타내는 흐름도이다.

도 5는 호스트 가상 메모리에 의해 지원되는 게스트 물리적 메모리의 데이터의 라이프 사이클에서 다양한 예시적인 액션을 나타내는 흐름도이다.

도 6은 호스트 가상 메모리로 게스트 물리적 메모리를 지원하기 위한 예시적인 메커니즘을 나타내는 블록도이다.

도 7은 호스트 가상 메모리로 게스트 물리적 메모리를 지원하면서 게스트 물리적 메모리와 호스트 물리적 메모리 간에 보다 효율적으로 트랜슬레이팅하기 위한 예시적인 메커니즘을 나타내는 블록도이다.

도 8은 호스트 가상 메모리로 게스트 물리적 메모리를 지원하면서 게스트 물리적 메모리와 호스트 물리적 메모리 간에 보다 효율적으로 트랜슬레이팅하기 위한 예시적인 방법을 나타내는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0010] 이하의 설명은 가상 머신 컴퓨팅 환경을 제공하는 하나 이상의 가상 머신 프로세스를 호스팅하는 컴퓨팅 디바이스 상에서 보다 효율적인 메모리 활용을 동시에 제공하면서 메모리 액세스의 효율성을 증가시키는 것에 관한 것이다. 호스트 컴퓨팅 디바이스 상에서 실행되는 가상 시스템 프로세스의 메모리 활용의 효율성을 증가시키기 위해, 이러한 가상 머신 프로세스는 호스트의 가상 메모리에 의해 지원될 수 있으며, 이에 의해 이러한 가상 머신 프로세스에 의해 소비되는 메모리에 종래의 가상 메모리 효율성을 적용할 수 있다. 가상 머신 환경의 게스트 물리적 메모리가 호스트 컴퓨팅 디바이스 상에서 실행되는 하나 이상의 프로세스에 할당된 가상 메모리에 의해 지원되는 환경에서 제2 계층 주소 테이블(SLAT)의 계층적 레벨이 메모리 액세스의 일부로서 순회되는 속도를 증가시키기 위해, SLAT 내의 하나 이상의 계층적 레벨의 테이블이 스킵되거나 다르게 참조되지 않을 수 있으며, 이에 의해 보다 효율적인 SLAT 순회 및 보다 효율적인 메모리 액세스로 귀결된다. SLAT는 계층적으로 상위-레벨의 테이블에서 메모리 상관으로 파플레이팅될 수 있지만, 호스트 컴퓨팅 디바이스의 가상 메모리 제공을 지원하는 호스트 컴퓨팅 디바이스의 페이지 테이블은 계층적으로 최저 테이블 레벨에서 대응하는 연속적인 메모리 상관 세트를 유지할 수 있으며, 이에 의해 호스트 컴퓨팅 디바이스가 더 작은 메모리 청크를 페이지 아웃하거나 다르게 조작할 수 있게 한다.

[0011] 이러한 조작이 발생하면, SLAT는 계층적으로 최저 테이블 레벨에서 메모리 상관으로 다시 파플레이팅될 수 있다. 반대로, 호스트가 충분히 큰 연속적인 작은 페이지 세트를 재어셈블링할 수 있는 경우, SLAT는 계층적으로 더 높은 레벨의 테이블에서 상관으로 다시 파플레이팅될 수 있으며, 이에 의해 다시 보다 효율적인 SLAT 순회 및 보다 효율적인 메모리 액세스로 귀결된다. 이러한 방식으로, 호스트 컴퓨팅 디바이스의 가상 메모리로 가상 머신 프로세스를 지원하는 메모리 활용 효율성의 이점을 유지하면서 보다 효율적인 SLAT 순회가 달성될 수 있다.

[0012] 필수적인 것은 아니지만, 이하의 설명은 컴퓨팅 디바이스에 의해 실행되는 프로그램 모듈과 같은 컴퓨터-실행 가능 명령의 일반적인 맥락에서 이루어질 것이다. 더욱 구체적으로, 설명은 달리 지시되지 않는 한, 하나 이상의 컴퓨팅 디바이스 또는 주변 기기에 의해 수행되는 행동 및 동작의 부호적 표현을 참조할 것이다. 이와 같이, 때로 컴퓨터에 의해 실행되는 것으로 언급되는 이러한 행동 및 동작은 구조화된 형태로 데이터를 나타내

는 전기 신호의 프로세싱 유닛에 의한 조작을 포함한다는 것이 이해될 것이다. 이러한 조작은 데이터를 변환하거나 이를 메모리의 위치에서 유지하며, 이는 본 기술 분야의 통상의 기술자에 의해 잘 이해되는 방식으로 컴퓨팅 디바이스 또는 주변 기기의 동작을 재구성하거나 다르게 변경한다. 데이터가 유지되는 데이터 구조는 데이터 포맷에 의해 정의된 특성의 특성을 갖는 물리적 위치이다.

[0013] 일반적으로, 프로그램 모듈은 특정 작업을 수행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 객체, 구성 요소, 데이터 구조 등을 포함한다. 또한, 본 기술 분야의 통상의 기술자는 컴퓨팅 디바이스가 통상의 퍼스널 컴퓨터로 제한될 필요가 없으며, 서버, 휴대용 디바이스, 다중-프로세서 시스템, 마이크로프로세서 기반 또는 프로그래밍 가능한 소비자 전자 기기, 네트워크 PC, 미니 컴퓨터, 메인프레임 컴퓨터 등을 포함하는 다른 컴퓨팅 구성을 포함한다는 것을 이해할 것이다. 유사하게, 메커니즘은 또한 통신 네트워크를 통해 링크된 원격 프로세싱 디바이스에 의해 작업이 수행되는 분산 컴퓨팅 환경에서 실행될 수 있기 때문에, 컴퓨팅 디바이스는 독립형 컴퓨팅 디바이스로 제한될 필요가 없다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 로컬 및 원격 메모리 저장 디바이스 모두에 위치될 수 있다.

[0014] 위에서 참조된 메모리 할당 및 액세스 메커니즘의 상세한 설명을 진행하기 전에, 아래 설명을 위한 컨텍스트를 제공하는 예시적인 호스트 컴퓨팅 디바이스의 상세한 설명이 도 1에 나타난 예시적인 컴퓨팅 디바이스(100)를 참조하여 제공된다. 예시적인 컴퓨팅 디바이스(100)는 하나 이상의 중앙 처리 장치(CPU: central processing unit)(120), 시스템 메모리(130) 및 시스템 메모리를 포함하는 다양한 시스템 구성 요소를 처리 장치(120)에 커플링하는 시스템 버스(121)를 포함할 수 있지만 이에 한정되지 않는다. 시스템 버스(121)는 메모리 버스 또는 메모리 컨트롤러, 주변 버스 및 임의의 다양한 버스 아키텍처를 사용하는 로컬 버스를 포함하는 몇몇 유형의 버스 구조 중 임의의 것일 수 있다. 컴퓨팅 디바이스(100)는 그래픽 하드웨어 인터페이스(160) 및 디스플레이 디바이스(161)를 포함하지만 이에 한정되지 않는 그래픽 하드웨어를 선택적으로 포함할 수 있으며, 이는 터치-감지, 또는 멀티-터치 가능 디스플레이 디바이스와 같은 터치-기반 사용자 입력을 수신할 수 있는 디스플레이 디바이스를 포함할 수 있다. 특성의 물리적 구현에 따라, CPU(120), 시스템 메모리(130) 및 컴퓨팅 디바이스(100)의 다른 구성 요소 중 하나 이상은 단일 칩 상에서와 같이 물리적으로 공존할 수 있다. 이러한 경우에, 시스템 버스(121)의 일부 또는 전부는 단지 단일 칩 구조 내의 실리콘 경로일 수 있으며, 도 1의 예시는 예시의 목적을 위한 단지 표기적 편의일 수 있다.

[0015] 컴퓨팅 디바이스(100)는 또한 통상적으로 컴퓨팅 디바이스(100)에 의해 액세스될 수 있는 임의의 이용 가능한 매체를 포함하는 컴퓨터 판독 가능 매체를 포함하고, 휘발성 및 비휘발성 매체와 이동식 및 비이동식 매체를 모두 포함한다. 제한이 아닌 예시의 방식으로, 컴퓨터 판독 가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터 판독 가능 명령, 데이터 구조, 프로그램 모듈 또는 다른 데이터와 같은 콘텐츠를 저장하기 위한 임의의 방법 또는 기술로 구현된 매체를 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래시 메모리 또는 다른 메모리 기술, CD-ROM, 디지털 버서타일 디스크(DVD: digital versatile disk) 또는 다른 광 디스크 스토리지, 자기 카세트, 자기 테이프, 자기 디스크 스토리지 또는 다른 자기 저장 디바이스, 또는 원하는 콘텐츠를 저장하는 데 사용될 수 있고 컴퓨팅 디바이스(100)에 의해 액세스될 수 있는 임의의 다른 매체를 포함하지만 이에 제한되지 않는다. 그러나, 컴퓨터 저장 매체는 통신 매체를 포함하지 않는다. 통신 매체는 통상적으로 컴퓨터 판독 가능 명령, 데이터 구조, 프로그램 모듈 또는 다른 데이터를 반송파 또는 다른 전송 메커니즘과 같은 변조된 데이터 신호로 구현하고 임의의 콘텐츠 전달 매체를 포함한다. 제한이 아닌 예시의 방식으로, 통신 매체는 유선 네트워크 또는 직접-유선 접속과 같은 유선 매체와 음향, RF, 적외선 및 다른 무선 매체와 같은 무선 매체를 포함한다. 상술한 임의의 조합 또한 컴퓨터 판독 가능 매체의 범위 내에 포함되어야 한다.

[0016] 시스템 메모리(130)는 판독 전용 메모리(ROM: read only memory)(131) 및 랜덤 액세스 메모리(RAM: random access memory)(132)와 같은 휘발성 및/또는 비휘발성 메모리 형태의 컴퓨터 저장 매체를 포함한다. 기동(start-up) 동안과 같이, 컴퓨팅 디바이스(100) 내의 요소들 사이의 콘텐츠 전송을 돕는 기본 루틴을 포함하는 기본 입력/출력 시스템(133)(BIOS: basic input/output system)은 통상적으로 ROM(131)에 저장된다. RAM(132)은 통상적으로 즉시 액세스 가능하고/가능하거나 처리 장치(120)에 의해 현재 동작 중인 데이터 및/또는 프로그램 모듈을 포함한다. 제한이 아닌 예시의 방식으로, 도 1은 운영 체제(134), 다른 프로그램 모듈(135) 및 프로그램 데이터(136)를 나타낸다.

[0017] 컴퓨팅 디바이스(100)는 또한 다른 이동식/비이동식, 휘발성/비휘발성 컴퓨터 저장 매체를 포함할 수 있다. 단지 예시의 방식으로, 도 1은 비이동식, 비휘발성 자기 매체로부터 읽거나 이에 쓰는 하드 디스크 드라이브(141)를 나타낸다. 예시적인 컴퓨팅 디바이스와 함께 사용될 수 있는 다른 이동식/비이동식, 휘발성/비휘발성 컴

퓨터 저장 매체는 자기 테이프 카세트, 플래시 메모리 카드, 디지털 버서타일 디스크, 디지털 비디오 테이프, 솔리드 스테이트 RAM, 솔리드 스테이트 ROM 및 위에 정의되고 기술된 다른 컴퓨터 저장 매체를 포함하지만, 이에 한정되지 않는다. 하드 디스크 드라이브(141)는 통상적으로 인터페이스(140)와 같은 비휘발성 메모리 인터페이스를 통해 시스템 버스(121)에 접속된다.

[0018] 위에서 논의되고 도 1에 나타난 드라이브 및 관련 컴퓨터 저장 매체는 컴퓨팅 디바이스(100)에 대한 컴퓨터 관독 가능 명령, 데이터 구조, 프로그램 모듈 및 다른 데이터의 저장을 제공한다. 도 1에서, 예를 들어, 하드 디스크 드라이브(141)는 운영 체제(144), 다른 프로그램 모듈(145) 및 프로그램 데이터(146)를 저장하는 것으로 나타내어져 있다. 이들 구성 요소는 운영 체제(134), 다른 프로그램 모듈(135) 및 프로그램 데이터(136)와 동일하거나 상이할 수 있음에 유의한다. 운영 체제(144), 다른 프로그램 모듈(145) 및 프로그램 데이터(146)에는 적어도 이들이 상이한 사본임을 나타내기 위해 이에 다른 번호가 부여된다.

[0019] 컴퓨팅 디바이스(100)는 하나 이상의 원격 컴퓨터에 대한 논리적 접속을 사용하여 네트워킹된 환경에서 동작할 수 있다. 컴퓨팅 디바이스(100)는 네트워크 인터페이스 또는 어댑터(150)를 통해 일반 네트워크 접속(151)(네트워크(190))에 접속되는 것으로 나타내어져 있으며, 네트워크 인터페이스 또는 어댑터(150)는 차례로 시스템 버스(121)에 접속된다. 네트워킹된 환경에서, 컴퓨팅 디바이스(100), 또는 그 일부 또는 주변 기기와 관련하여 도시된 프로그램 모듈은 일반 네트워크 접속(161)을 통해 컴퓨팅 디바이스(100)에 통신 가능하게 커플링된 하나 이상의 다른 컴퓨팅 디바이스의 메모리에 저장될 수 있다. 나타난 네트워크 접속은 예시이고 컴퓨팅 디바이스들 사이의 통신 링크를 확립하는 다른 수단이 사용될 수 있음을 이해할 것이다.

[0020] 단일 물리적 디바이스로 설명되었지만, 예시적인 컴퓨팅 디바이스(100)는 가상 컴퓨팅 디바이스일 수 있으며, 이 경우 CPU(120), 시스템 메모리(130), 네트워크 인터페이스(160) 및 다른 유사한 구성 요소와 같은 상술한 물리적 구성 요소의 기능은 컴퓨터-실행 가능 명령에 의해 제공될 수 있다. 이러한 컴퓨터-실행 가능 명령은 단일 물리적 컴퓨팅 디바이스 상에서 실행될 수 있거나, 이러한 컴퓨터-실행 가능 명령을 호스팅하는 특정의 물리적 컴퓨팅 디바이스가 필요와 이용 가능성에 의존하여 시간 경과에 따라 동적으로 변할 수 있는 동적인 방식으로 복수의 물리적 컴퓨팅 디바이스에 걸쳐 분산되는 것을 포함하여 복수의 물리적 컴퓨팅 디바이스에 걸쳐 분산될 수 있다. 예시적인 컴퓨팅 디바이스(100)가 가상화된 디바이스인 상황에서, 이러한 가상화된 컴퓨팅 디바이스를 호스팅하는 기본 물리적 컴퓨팅 디바이스는 그 자체로 상술한 것과 유사한 물리적 구성 요소를 포함할 수 있으며, 유사한 방식으로 동작한다. 또한, 가상 컴퓨팅 디바이스는 다른 가상 컴퓨팅 디바이스의 구성 내에서 실행되는 하나의 가상 컴퓨팅 디바이스와 함께 복수의 계층에서 활용될 수 있다. 따라서, 본원에서 활용되는 용어 "컴퓨팅 디바이스"는 물리적 컴퓨팅 디바이스 또는 가상 컴퓨팅 디바이스를 포함하는 가상화된 컴퓨팅 환경 중 어느 하나를 의미하며, 가상 컴퓨팅 디바이스 내에서 컴퓨터-실행 가능 명령은 물리적 컴퓨팅 디바이스에 의한 실행과 일치하는 방식으로 실행될 수 있다. 유사하게, 본원에서 활용되는 컴퓨팅 디바이스의 물리적 구성 요소를 지칭하는 용어는 동일하거나 동등한 기능을 수행하는 해당 물리적 구성 요소 또는 그 가상화 중 어느 하나를 의미한다.

[0021] 도 2에서, 여기에 나타난 시스템(200)은 컴퓨팅 디바이스 하드웨어 및 그 상에서 실행되는 프로세스 모두를 포함하는 컴퓨팅 시스템을 나타낸다. 보다 구체적으로, 도 2에 나타난 예시적인 시스템(200)은 상술한 예시적인 컴퓨팅 디바이스(100)의 일부로서 나타난 위에 언급한 처리 장치(120) 및 RAM(132)과 같은 컴퓨팅 디바이스 하드웨어 자체를 포함한다. 또한, 예시적인 시스템(200)은 예시적인 하이퍼바이저(210)와 같은 하이퍼바이저를 포함하며, 이는 컴퓨팅 디바이스(100)가 예시적인 하이퍼바이저(210)와 관련된 기능을 수행할 수 있도록 하기 위해 예시적인 컴퓨팅 디바이스(100)에 의해 실행되는 컴퓨터 실행 가능 명령으로 구성될 수 있다. 추가로 후술하는 바와 같이, 예시적인 하이퍼바이저(210)는 예시적인 SLAT(220)와 같은 제2 계층 주소 테이블(SLAT)을 포함하거나 이를 참조할 수 있다.

[0022] 예시적인 하이퍼바이저(210)는 컴퓨팅 디바이스(100)의 하드웨어에 상응할 수 있거나, 프로세서 유형 및/또는 용량의 차이, RAM의 양 및/또는 유형의 차이, 저장 매체의 양 및/유형의 차이 및 다른 유사한 차이를 포함하여 이와 상이할 수 있는 컴퓨팅 하드웨어 세트를 가상화할 수 있다. 이러한 가상화는 특히 예시적인 가상 머신 프로세스(260 및 250)와 같은 하나 이상의 가상 머신 프로세스를 가능하게 할 수 있다. 하이퍼바이저(210)는 이러한 가상 머신의 컨텍스트 내에서 실행되는 프로세스 및 하이퍼바이저의 탭 상에서 실행되는 임의의 다른 프로세스 모두에 컴퓨팅 디바이스 하드웨어(110) 상에서 직접 실행되는 외관을 제시할 수 있다. 도 2에 나타난 예시적인 시스템(200)은 예시적인 하이퍼바이저(210)의 탭 상에서 실행되는 예시적인 운영 체제(240)와 같은 운영 체제를 나타낸다. 결국, 예시적인 시스템(200)에 의해 나타난 바와 같이, 예시적인 운영 체제(240)는 위에 언급한 가상 머신 프로세스(260 및 250)와 같은 하나 이상의 컴퓨터 실행 가능 어플리케이션 프로그램을 그 위에

서 실행할 수 있다.

[0023] 예시적인 운영 체제(240)는 아래의 설명과 관련된 다양한 구성 요소, 하위 구성 요소, 또는 그 양태를 포함할 수 있다. 도 2에서는 시각적으로 구별되는 요소로 나타내었지만, 이러한 시각적 도시는 프로세스 경계, 독립 메모리 사일로(silo) 또는 다른 유사한 계산의 기술과 같은 특정 계산적인 독립성을 전달하기 위한 것이 아니다. 운영 체제(240)의 하나의 양태는 예시적인 메모리 관리자(241)와 같은 메모리 관리자일 수 있다. 보다 구체적으로, 예시적인 메모리 관리자(241)는 하나 이상의 컴퓨터 판독 가능 저장 매체로부터 코드 또는 데이터를 판독하고 이러한 코드 또는 데이터를 RAM(132)에 의해 지원될 수 있는 메모리로 복제할 수 있다.

[0024] 본원에서 활용되는 바와 같이, 용어 "가상 메모리"는 가상 머신을 지칭하는 것이 아니라, 어플리케이션 프로그램 또는 컴퓨팅 디바이스 상에서 실행되는 다른 프로세스에 대해 제시하는 개념을 지칭하며, RAM(132)에 물리적으로 존재하는 것보다 더 많은 양의 메모리에 대한 액세스를 갖는 외관이다. 따라서, 예를 들어, 운영 체제(240)에 의해 제공되는 가상 메모리 기능은 물리적 메모리(132)와 같이 궁극적으로 물리적 메모리에 의해 지원되는 메모리로의 코드 및/또는 데이터의 로딩을 허용할 수 있지만, 그 양은 물리적 메모리보다 크다. 운영 체제(240)의 일부로서 실행되는 메모리 관리자(241)는 하나 이상의 가상 메모리 주소와 하나 이상의 물리적 메모리 주소 간의 상관을 설명하기 위해 페이지 테이블(250)을 유지, 수정 및/또는 활용할 수 있다. 보다 구체적으로, 페이지 테이블(250)은 하나의 메모리 어드레싱 스킴과 다른 상이한 메모리 어드레싱 스킴 사이의 주소 트랜슬레이션을 제공한다. 이를 달성하기 위해, 페이지 테이블(250)은 하나의 메모리 어드레싱 스킴에 따른 하나 이상의 메모리 주소, 즉, 가상 메모리 주소를 다른 상이한 메모리 어드레싱 스킴에 따른 하나 이상의 메모리 주소, 즉, 물리적 메모리 주소에 상관시킬 수 있는, 추가로 상세히 후술되는 복수 계층 레벨의 테이블을 포함할 수 있다.

[0025] 도 2에 나타난 예시적인 시스템(200)은 운영 체제(240)의 탑 상에서 실행되는 어플리케이션 프로그램을 나타낸다. 특히, 이러한 어플리케이션 프로그램은 예시적인 가상 머신 프로세스(260 및 270)와 같은 가상 머신 프로세스를 인스턴스화할 수 있는 가상 머신 어플리케이션 프로그램일 수 있다. 각각의 가상 머신 프로세스는 자신의 운영 체제, 어플리케이션 프로그램 및 컴퓨팅 디바이스(100)에 의해 제시되는 것과 같은 물리적 컴퓨팅 환경에 의해 호스팅될 수 있는 다른 유사한 기능을 호스팅할 수 있는 가상 컴퓨팅 환경을 생성할 수 있다. 도 2에 나타난 예시적인 시스템(200) 내에서, 예시적인 가상 머신 프로세스(260)는 가상 머신 환경(261)을 생성하는 것으로 나타내어지고, 예시적인 가상 머신 프로세스(270)는 가상 머신 환경(271)을 생성하는 것으로 나타내어진다. 예시적인 가상 머신 환경(271)은 예시적인 운영 체제(240)와 유사할 수 있거나 다른 유형의 운영 체제일 수 있는 예시적인 운영 체제(281)의 형태로 운영 체제를 호스팅하는 것으로 나타내어진다. 예시를 위해, 예시적인 가상 머신 환경(271) 내에서 실행되는 예시적인 운영 체제(281)는 가상 머신 환경(271) 내에서 가상 메모리를 제공하기 위해 결국 페이지 테이블, 즉, 예시적인 페이지 테이블(283)을 참조하는 메모리 관리자, 즉, 예시적인 메모리 관리자(282)를 포함하는 것으로 나타내어진다. 가상 머신 환경(271) 내의 이러한 가상 메모리는 또한 예시적인 가상 머신 환경(271) 내에서 실행되는 것으로 도 2에 나타내어진 예시적인 프로세스(291 및 292)와 같은 예시적인 운영 체제(281)의 탑 상에서 실행되는 프로세스에 의해 활용될 수 있다.

[0026] 메모리 관리자(241 및 282)에 의해 구현되는 것과 유사한 방식으로, 메모리를 포함하는, 컴퓨팅 하드웨어의 가상 세트의 표현을 용이하게 하는 데 있어서 하이퍼바이저(210)는 메모리 주소를 하나의 메모리 어드레싱 스킴으로부터 다른 메모리 어드레싱 스킴으로 유사하게 트랜슬레이팅할 수 있다. 보다 구체적으로, 하이퍼바이저(210)는 물리적 메모리 주소인 것으로 하이퍼바이저(210)에 의해 제시된 메모리 주소를 예를 들어, 메모리(132) 내의 물리적 위치를 식별하는 메모리 주소와 같은 실제 물리적 메모리 주소로 트랜슬레이팅할 수 있다. 이러한 트랜슬레이션은 다른 계층적으로 조직된 테이블, 즉, SLAT(220)를 참조하여 수행될 수 있으며, 이는 또한 하나의 메모리 어드레싱 스킴, 이 경우 물리적 메모리 주소인 것으로 하이퍼바이저의 탑 상에서 실행되는 프로세스에 의해 인식되는 주소와, 또 다른 상이한 메모리 어드레싱 스킴, 즉, 이 경우 실제 물리적 메모리 주소 간의 메모리 주소 트랜슬레이션을 제공할 수 있다.

[0027] 일 양태에 따르면, 운영 체제(240)에 의해 제공되는 가상 메모리는 호스트 컴퓨팅 디바이스 상에서 페이지되지 않은 물리적 메모리 할당을 사용하지 않고 가상 머신의 물리적 메모리를 지원하는 데 활용될 수 있다. 이는 메모리 관리자(241)가 게스트 물리적 메모리와 연관된 호스트 물리적 메모리를 관리할 수 있게 한다. 특히, 메모리 관리자(241)와 같이 호스트 상에 이미 존재하는 메모리 관리 로직은 이제 게스트 가상 머신의 물리적 메모리를 관리하는 데 활용될 수 있다. 이는 예시적인 하이퍼바이저(210)와 같은 하이퍼바이저를 구현하는 데 사용되는 코드의 양의 관점에서 더 작은 하이퍼바이저의 사용을 허용할 수 있다. 악용될 수 있거나 에러를 가질 수 있는 코드가 적기 때문에 더 작은 하이퍼바이저가 더 안전할 수 있다. 또한, 가상 머신을 구현하는 데 더 적은

호스트 물리적 메모리가 활용될 수 있으므로, 이는 호스트 상의 증가된 가상 머신 밀도를 허용한다. 예를 들어, 이전에는, 가상 머신 프로세스가 페이지되지 않은 메모리를 필요로 했기 때문에, 예시적인 컴퓨팅 디바이스(100)가 3 GB의 RAM(132)을 포함한다면, 2 GB의 RAM을 갖는 가상 머신을 생성한 하나의 가상 머신 프로세스만을 호스팅할 수 있었다. 반대로, 예시적인 가상 머신 프로세스(270)와 같은 가상 머신 프로세스에 의해 소비된 메모리가 운영 체제(240)의 메모리 관리자(241)에 의해 제공되는 것과 같은 가상 메모리였다면, 4 GB의 가상 메모리 공간이 각각 2 GB의 RAM을 갖는 가상 머신을 생성하는 2개의 가상 머신 프로세스에 대한 지원을 제공할 수 있다. 이러한 방식으로, 예시적인 컴퓨팅 디바이스(100)는 단지 하나가 아닌 2개의 가상 머신 프로세스를 호스팅할 수 있으며, 이에 의해 예시적인 컴퓨팅 디바이스(100) 상에서 실행되는 가상 머신 프로세스의 밀도를 증가시킬 수 있다.

[0028] 다른 이점은 시스템(호스트 및 가상 머신) 상의 모든 메모리를 관리하는 단일 메모리 관리 코드 베이스를 포함한다. 따라서, 하나의 코드 베이스에서의 개선, 수정 및/또는 조정은 모두에게 이익이 된다. 또한, 하나의 코드 베이스만 유지하면 되므로 이는 엔지니어링 비용의 감소로 귀결될 수 있다. 다른 예시적인 이점은 가상 머신이 페이징, 페이지 공유, 작업 세트 에이징 및 트리밍, 오류 클러스터링 등과 같은 밀도 향상으로부터 즉시 이익을 얻을 수 있는 것일 수 있다. 또 다른 예시적인 이점은 가상 메모리 및 물리적 메모리 소비 제한이 임의의 다른 프로세스와 같이 가상 머신에 대해 설정될 수 있어 시스템 거동을 제어하는 관리자 능력을 허용하는 것일 수 있다. 또 다른 이점은 가상 머신에 더 많은 성능, 밀도 및 기능을 제공하기 위해 호스트 메모리 관리자에 추가적인 피치가 추가될 수 있는 것일 수 있다(그리고, 다른 비가상 머신 워크로드도 이로부터 마찬가지로 이익을 얻을 수 있을 것이다).

[0029] 도 3에 있어서, 시스템(300)은 도 2에 나타난 시스템(200)에 적용 가능한 복수 레벨의 페이지 테이블 및 대응하는 메모리 어드레싱 스킴을 참조하기 위해 본원에 활용된 용어를 나타낸다. 보다 구체적으로, 가상 머신 환경(271) 내에서, 예시적인 메모리 관리자(282)는 가상 머신 환경(271)의 컨텍스트 내에서 가상 메모리 주소를 다시 가상 머신 환경(271)의 컨텍스트 내에서 물리적 메모리 주소에 상관시키기 위해 페이지 테이블(283)을 참조할 수 있다. 이전에 나타난 바와 같이, "가상 메모리"라는 용어는 가상 머신 환경 내의 메모리를 지칭하는 것이 아니라, 물리적 메모리 용량에 의해 지원되지만 이보다 큰 컴퓨팅 디바이스 상에서 실행되는 프로세스에 이용 가능하게 된 메모리의 양을 지칭한다. 따라서, 가상 머신 환경 내의 가상 메모리와 가상화되지 않은 또는 "베어 메탈(bare metal)" 컴퓨팅 환경 내의 가상 메모리 간에 명목상으로 구별하기 위해, 형용사 "게스트" 및 "호스트"가 활용될 것이며, 형용사 "게스트"는 가상 머신 컴퓨팅 환경을 나타내고, 형용사 "호스트"는 가상 머신 환경을 제공하는 가상 머신 프로세스를 실행하는 베어 메탈 컴퓨팅 환경을 나타낸다.

[0030] 이와 같이, 그리고 시스템(301)에 나타난 바와 같이, "게스트 가상 메모리"(311) 어드레싱 스킴 내의 주소는 도 2에 나타난 가상 머신 환경(271) 내의 페이지 테이블(283)에 의해 "게스트 물리적 메모리"(312) 어드레싱 스킴 내의 주소와 상관될 수 있다. 유사하게, "호스트 가상 메모리"(321) 어드레싱 스킴 내의 주소는 비가상 컴퓨팅 환경에서 실행되는 도 2에 나타난 예시적인 운영 체제(240)의 일부인 페이지 테이블(250)에 의해 "호스트 물리적 메모리"(322) 어드레싱 스킴 내의 주소와 상관될 수 있다. 도 3의 예시적인 시스템(301)에 의해 나타내지 않았지만, 일 양태에 따라, 그리고 아래에서 추가로 설명되는 바와 같이, 가상 머신 환경의 게스트 물리적 메모리(312)는 호스트 가상 메모리(321)에 의해 지원될 수 있으며, 이에 의해 상술한 이점을 제공한다. 호스트 가상 메모리(321)로 게스트 물리적 메모리(312)를 지원하는 것은 SLAT(220)의 수정을 수반할 수 있으며, SLAT(220)는 예시적인 시스템(301)에 의해 나타난 바와 같이, "게스트 물리적 메모리"(312) 어드레싱 스킴과 "호스트 물리적 메모리"(322) 어드레싱 스킴 사이의 트랜슬레이션을 제공한다.

[0031] 또한 도 3에 나타난 시스템(302)에 있어서, 예시적인 페이지 테이블(250), 예시적인 페이지 테이블(283), 및/또는 예시적인 SLAT(220)를 구현하기 위해 활용될 수 있는 것과 같은 테이블의 계층적 배열이 나타내어진다. 현대의 컴퓨팅 디바이스 및 어플리케이션 프로그램 프로세스에 적용할 수 있는 메모리의 양을 수용하기 위해, 페이지 테이블(250, 283) 및/또는 SLAT(220)는 복수의, 통상적으로 4개 레벨의 계층적 테이블을 가질 것이다. 그러나, 예시의 용이함을 위해 시스템(302)은 단지 2개의 레벨의 계층적 테이블을 나타낸다. 계층적으로 상위-레벨 테이블(340)은 예시적인 테이블 엔트리(341, 342, 343 및 344)와 같은 복수의 테이블 엔트리를 포함할 수 있다. 예를 들어, 계층적으로 상위-레벨 테이블(340)은 4 KB의 크기일 수 있으며, 512개의 개별 테이블 엔트리를 포함할 수 있으며, 이는 각각 8 바이트의 크기일 수 있다. 각각의 테이블 엔트리는 예시적인 계층적으로 하위-레벨 테이블(350 및 360) 중 하나와 같은 계층적으로 하위-레벨 테이블의 포인터 또는 다른 유사한 식별자를 포함할 수 있다. 따라서, 예를 들어, 예시적인 테이블 엔트리(341)는 화살표(371)에 의해 나타난 바와 같이, 예시적인 계층적으로 하위-레벨 테이블(350)의 식별을 포함할 수 있다. 계층적으로 상위-레벨 테이블(340)이 4

KB의 크기이고 각각의 테이블 엔트리가 8 바이트의 크기인 경우, 이러한 8 바이트의 데이터 중 적어도 일부는 계층적으로 하위-레벨 테이블(350)의 포인터 또는 다른 유사한 식별자일 수 있다. 예시적인 테이블 엔트리(342)는 유사한 방식으로 화살표(372)에 의해 나타난 바와 같이, 계층적으로 하위-레벨 테이블(360)의 식별을 포함할 수 있다. 예시적인 계층적으로 상위-레벨 테이블(340)의 나머지 개별 테이블 엔트리의 각각은 유사하게 고유한 계층적으로 하위-레벨 테이블을 식별할 수 있다. 따라서, 계층적으로 상위-레벨 테이블(340)이 4 KB의 크기이고, 각각 8 바이트의 크기인 512개의 테이블 엔트리를 포함하는 예에서, 이러한 계층적으로 상위-레벨 테이블(340)은 예시적인 계층적으로 하위-레벨 테이블(350 및 360)과 같이, 512개의 고유한 계층적 하위-레벨의 512개의 식별을 포함할 수 있다.

[0032] 계층적으로 하위-레벨 테이블의 각각은 유사한 방식으로 개별 테이블 엔트리를 포함할 수 있다. 예를 들어, 예시적인 계층적으로 하위-레벨 테이블(350)은 예시적인 테이블 엔트리(251, 352, 353 및 354)를 포함할 수 있다. 유사하게, 예시적인 계층적으로 하위-레벨 테이블(360)은 테이블 엔트리(361, 362, 363 및 364)를 포함할 수 있다. 하나의 예로서, 계층적으로 하위-레벨 테이블의 각각은 계층적으로 상위-레벨 테이블(340)과 동등한 크기 및 구조일 수 있다. 따라서, 예시적인 계층적으로 하위-레벨 테이블(350)은 4 KB의 크기일 수 있으며, 각각 8 바이트의 크기일 수 있는, 예시적인 테이블 엔트리(351, 352, 353 및 354)와 같은 512개의 테이블 엔트리를 포함할 수 있다. 시스템(302)에 의해 나타난 예에서, 예를 들어, 예시적인 계층적으로 하위-레벨 테이블(350)의 예시적인 테이블 엔트리(351, 352, 353 및 354)와 같은 계층적으로 하위-레벨 테이블의 테이블 엔트리는 각각 연속적인 범위의 메모리 주소를 식별할 수 있다. 예를 들어, 예시적인 테이블 엔트리(351)는 화살표(391)에 의해 나타난 바와 같이, 메모리 주소(371)의 범위를 식별할 수 있다. 메모리 주소(371)의 범위는 메모리의 "페이지"를 포함할 수 있고, 메모리의 개별적으로 관리 가능한 최소의 양일 수 있다. 따라서, 설치된 물리적 메모리의 양보다 큰 가상 메모리의 양을 가능하게 하는 것의 일부로서 휘발성 메모리에 저장된 정보를 일시적으로 비휘발성 저장 매체로 이동시키거나 그 반대와 같은 액션은 페이지별로 수행될 수 있다. 또한, 액세스 권한은 페이지별로 확립될 수 있다. 예를 들어, 메모리(371)의 범위는 테이블 엔트리(351)에 의해 단독으로 식별되기 때문에, 액세스 권한은 메모리(371)의 범위 내의 모든 메모리 주소에 적용될 수 있다. 반대로, 메모리(371) 범위 내의 메모리 주소의 액세스 권한은 화살표(392)에 의해 나타난 바와 같이, 다른 테이블 엔트리, 즉, 테이블 엔트리(352)에 의해 고유하게 식별될 수 있는 메모리(372)의 범위에 대해 확립된 액세스 권한과 독립적일 수 있다.

[0033] 메모리(371)의 범위와 같은 단일 메모리 페이지의 메모리 양은 프로세서 설계 및 다른 하드웨어 요소뿐만 아니라 통신 접속 등을 포함하는 다양한 요소에 따를 수 있다. 하나의 예에서, 메모리(371)의 범위에 의해 표현되는 것과 같은 메모리의 페이지는 4 KB의 크기일 수 있다.

[0034] 하나 이상의 메모리 주소에 의해 식별된 메모리 위치에 대한 액세스 요청을 수신하면, 액세스될 메모리 주소를 포함하는 메모리 주소의 범위에 대응하는 계층적으로 상위-레벨 테이블의 테이블 엔트리가 식별될 수 있다. 예를 들어, 액세스될 메모리 주소가 메모리(371)에 의해 표현된 메모리를 포함한다면, 테이블 엔트리(341)는 계층적으로 상위-레벨 테이블(340)에서 식별될 수 있다. 테이블 엔트리(341)의 식별은 결국 계층적으로 하위-레벨 테이블(350)의 식별로 귀결될 수 있다. 계층적으로 하위-레벨 테이블(350) 내에서 액세스될 메모리 주소는 테이블 엔트리(351)에 의해 식별되는 메모리의 일부인 것으로 결정될 수 있다. 그 후, 테이블 엔트리(351) 내의 정보가 액세스하려고 했던 메모리(371)를 식별할 수 있다.

[0035] 계층적으로 하위-레벨의 페이지 테이블의 각각의 이러한 순회는 메모리 액세스 요청을 수신하고 해당 메모리에 액세스를 반환하는 것 사이의 지연을 증가시킬 수 있다. 일 양태에 따르면, 예를 들어, 계층적으로 상위-레벨 테이블(340)의 예시적인 테이블 엔트리(341)와 같은 계층적으로 상위-레벨 테이블의 테이블 엔트리는 예시적인 계층적으로 하위-레벨 테이블(350)과 같은 계층적으로 하위-레벨 테이블을 식별할 수 없고, 오히려 계층적으로 하위-레벨 테이블의 테이블 엔트리에 의해 식별되었을 메모리의 전체 범위에 대응할 수 있는 메모리의 페이지를 식별할 수 있다. 따라서, 예를 들어, 계층적으로 하위-레벨 테이블(350)의 개별 테이블 엔트리(351, 352 및 353)에 의해 식별된 메모리(371, 372 및 373)의 각각의 페이지가 4 KB의 크기인 경우, 예시적인 계층적으로 하위-레벨 테이블(350)은 각각 4 KB 크기의 메모리의 범위를 식별하는 512개의 테이블 엔트리를 포함할 수 있으므로, 2 MB 크기의 메모리의 범위는 계층적으로 하위-레벨 테이블(350)의 모든 개별 테이블 엔트리의 결합된 양에 의해 식별될 수 있다. 이러한 예에서, 모든 2 MB의 메모리가 단일 2 MB 메모리 페이지로 취급된다면, 이러한 단일의 큰 메모리 페이지는 계층적으로 상위-레벨 테이블(340)의 예시적인 테이블 엔트리(341)에 의해 직접 식별될 수 있다. 이러한 경우, 예시적인 테이블 엔트리(341)는 예시적인 2 MB 메모리 페이지(370)와 같은 단일 2 MB 메모리 페이지를 직접 식별할 것이고 계층적으로 하위-레벨 테이블을 식별하지 않을 것이기 때문에, 임의의

계층적으로 하위-레벨 테이블을 참조할 필요가 없을 것이다. 예를 들어, 페이지 테이블 또는 SLAT가 4개의 계층적 레벨을 포함하는 경우, 예시적인 ms 메모리 페이지(370)와 같은 대형 메모리 페이지를 활용하는 것은 이러한 계층적 레벨의 테이블 중 하나가 참조되지 않고 스킵될 수 있게 할 것이고, 이에 의해 4개가 아닌 3개의 계층적 레벨의 테이블만을 참조함으로써 가상 메모리와 물리적 메모리 간 메모리 주소 트랜슬레이션을 제공하여, 보다 효율적으로, 즉, 25% 더욱 효율적으로 메모리 액세스를 제공한다. 페이지는 개별적으로 관리할 수 있는 최소량의 메모리이므로, 2 MB 크기의 메모리 페이지가 사용되는 경우, 메모리 액세스 권한은 예를 들어, 이러한 큰 메모리 페이지의 모든 2 MB 메모리에 대해 동일할 수 있다.

[0036] 추가적인 예로서, 계층적으로 예시적인 테이블(340) 위에 있는 계층적으로 배열된 테이블이 있고 엔트리(341, 342, 343 및 344)와 같은 예시적인 테이블(340)의 각각의 엔트리가 예시적인 2 MB 메모리 페이지(370)와 같은 2 MB 메모리 페이지에 어드레싱할 수 있다면, 이러한 512개의 개별 테이블 엔트리는 함께 통합될 때 1 GB의 메모리를 어드레싱할 수 있다. 단일 1 GB 페이지의 메모리가 사용된 경우, 테이블(340)에 대한 참조 및 예시적인 테이블(350)과 같은 계층적으로 하위-레벨 테이블에 대한 후속 참조는 모두 회피될 수 있다. 페이지 테이블 또는 SLAT가 4개의 계층적 레벨을 포함하는 예로 돌아가면, 이러한 거대한 메모리 페이지를 활용하는 것은 이러한 계층적 레벨의 테이블 중 2개가 스킵될 수 있게 하고 참조되지 않게 할 수 있으며, 이에 의해 4개가 아닌 계층적 레벨의 테이블만을 참조함으로써 가상 메모리와 물리적 메모리 간의 메모리 주소 트랜슬레이션을 제공하여 대략 절반의 시간에 메모리 액세스를 제공한다. 다시, 페이지는 개별적으로 관리할 수 있는 최소량의 메모리이므로, 1 GB 크기의 메모리 페이지가 사용하는 경우, 메모리 액세스 권한은 예를 들어, 이러한 거대한 메모리 페이지의 모든 1 GB 메모리에 대해 동일할 수 있다.

[0037] 본원에서 활용되는 용어 "큰 메모리 페이지"는 계층적으로 최저 레벨에서 단일 테이블에 의해 식별되었을 메모리의 모든 범위를 포함하고, 계층적으로 최저 레벨의 테이블 위의 하나의 계층 레벨에 있는 테이블의 단일 테이블 엔트리에 의해 고유하고 직접 식별될 수 있도록 크기가 정해진 연속적인 메모리 범위를 의미한다. 위에 제공된 특정 크기 조정 예에서, 본원에서 해당 용어가 정의되는 "큰 메모리 페이지"는 2 MB 크기의 메모리 페이지일 것이다. 그러나, 나타난 바와 같이, "큰 메모리 페이지"라는 용어는 특정 크기를 지칭하는 것이 아니라, "큰 메모리 페이지"의 메모리 양이 페이지 테이블 자체의 계층적 설계와 계층적으로 최저-레벨의 페이지 테이블에서 각각의 테이블 엔트리에 의해 참조되는 메모리의 양에 의존한다는 것을 지칭한다.

[0038] 유사한 방식으로, 본원에서 활용되는 "거대 메모리 페이지"라는 용어는 계층적으로 두번째-최저-레벨에서 단일 테이블에 의해 식별되고 계층적으로 두번째로-최저-레벨의 테이블 위의 하나의 계층 레벨에서 테이블의 단일 테이블 엔트리에 의해 고유하고 직접 식별될 수 있는 메모리의 모든 범위를 포함하도록 크기가 정해지는 메모리의 연속 범위를 의미한다. 위에 제공된 특정 크기 조정 예에서, 해당 용어가 본원에서 활용되는 "거대 메모리 페이지"는 1 GB 크기의 메모리 페이지일 것이다. 다시, "거대 메모리 페이지"라는 용어는 특정 크기를 지칭하는 것이 아니라, "거대 메모리 페이지"의 메모리 양이 페이지 테이블 자체의 계층적 설계와 계층적으로 최저-레벨의 페이지 테이블에서 각각의 테이블 엔트리에 의해 참조되는 메모리의 양에 의존한다는 것을 지칭한다.

[0039] 다시 도 2로 돌아가서, (가상 머신의) 게스트 물리적 메모리를 지원하기 위해 (호스트 컴퓨팅 디바이스의) 호스트 가상 메모리가 활용될 수 있는 메커니즘에 대한 설명이 도 2에 나타난 예시적인 가상화 스택(230)과 같은 가상화 스택을 참조하여 제공된다. 사용자-모드 프로세스가 게스트 가상 머신을 지원하기 위한 가상 메모리를 제공하기 위해 호스트 컴퓨팅 디바이스 상에서 실행될 수 있다. 이러한 하나의 사용자-모드 프로세스가 각각의 게스트 머신에 대해 생성될 수 있다. 대안적으로, 단일 사용자-모드 프로세스가 복수의 가상 머신에 대해 사용될 수 있거나, 복수의 프로세스가 단일 가상 머신에 대해 사용될 수 있다. 대안적으로, 가상 메모리는 아래에 설명되는 바와 같이 사용자-모드 프로세스를 사용하는 것 이외의 다른 방식으로 구현될 수 있다. 설명을 위해, 가상 머신 프로세스(270)에 할당된 가상 메모리는 가상 머신 환경(271)에서 가상 머신 프로세스(270)에 의해 제시된 게스트 물리적 메모리를 지원하는 데 활용될 수 있으며, 유사하게, 가상 머신 프로세스(260)에 할당된 가상 메모리는 가상 머신 환경(261)에서 가상 머신 프로세스(260)에 의해 제시된 게스트 물리적 메모리를 지원하는 데 활용될 수 있다. 도 2의 예시적인 시스템(200)에서 이와 같이 나타내었지만, 앞서 언급한 사용자-모드 프로세스는 예시적인 가상 머신 프로세스(260 및 270)와 같은 가상 머신 프로세스로부터 분리될 수 있다.

[0040] 일 양태에 따르면, 예시적인 가상화 스택(230)과 같은 가상화 스택은 예시적인 가상 머신 프로세스(260 또는 270)와 같은 가상 머신을 호스팅할 지정된 사용자-모드 프로세스의 주소 공간에 호스트 가상 메모리를 할당할 수 있다. 호스트 메모리 관리자(241)는 이 메모리를 임의의 다른 가상 할당으로 취급할 수 있으며, 이는 페이지징될 수 있으며, 이를 지원하는 물리적 페이지가 시스템의 다른 곳에서 연속적인 메모리 할당을 충족하는 목적을 위해 변경될 수 있고, 물리적 페이지가 다른 프로세스에서 다른 가상 할당과 공유될 수 있음을 의미한다(이

는 결국 다른 가상 머신 지원 할당 또는 시스템 상의 임의의 다른 할당일 수 있음). 동시에, 호스트 메모리 관리자가 가상 할당을 지원하는 가상 머신을 필요에 따라 특별히 취급하도록 많은 최적화가 가능하다. 또한, 가상화 스택(230)이 밀도보다 성능을 우선하는 것으로 선택하면, 가상 머신이 해당 부분에 대한 페이지를 경험하지 않도록 보장하기 위하여 메모리에 페이지를 락킹하는 것과 같이 운영 체제 메모리 관리자(241)에 의해 지원되는 많은 동작을 수행할 수 있다. 마찬가지로, 아래에서 상세히 설명하는 바와 같이, 가상 머신에 대해 더욱 더 많은 성능을 제공하도록 큰 페이지가 사용될 수 있다.

[0041] 주어진 가상 머신은 호스트 가상 메모리에 의해 지원되는 게스트 물리적 메모리에 모든 게스트 물리적 메모리 주소를 가질 수 있거나, 호스트 가상 메모리에 의해 지원되는 게스트 물리적 메모리에 게스트 물리적 메모리 주소의 일부를 가질 수 있으며, 호스트 물리적 메모리로부터 이루어진 페이지되지 않은 물리적 메모리 할당과 같은 레거시 메커니즘에 의해 일부를 가질 수 있다.

[0042] 새로운 가상 머신이 생성될 때, 가상화 스택(230)은 게스트 물리적 메모리를 지원하기 위해 가상 메모리 할당을 호스팅하는 사용자-모드 프로세스를 사용할 수 있다. 이는 새로 생성된 빈 프로세스, 복수의 가상 머신을 호스팅하는 기존 프로세스 또는 가상 머신 자체에 보이지 않는 다른 가상 머신-관련 가상 할당도 포함하는 가상 머신 당 프로세스(예를 들어, 가상화 스택 데이터 구조)일 수 있다. 커널 가상 주소 공간을 사용하여 가상 머신을 지원할 수도 있다. 이러한 프로세스가 발견되거나 생성되면, 가상화 스택(230)은 가상 머신이 가져야 하는 게스트 물리적 메모리의 양에 대응하는 주소 공간에서 프라이빗 메모리 가상 할당(또는 섹션/파일 매핑)을 이룰 수 있다. 구체적으로, 가상 메모리는 프라이빗 할당, 파일 매핑, 페이지파일-지원 섹션 매핑 또는 호스트 메모리 관리자(241)에 의해 지원되는 임의의 다른 유형의 할당일 수 있다. 아래에 상세히 설명하는 바와 같이, 이러한 할당이 액세스 속도의 이점을 제공할 수 있지만, 하나의 연속 할당일 필요는 없다. 대신 이러한 할당은 임의의 개수의 할당일 수 있다.

[0043] 가상 메모리 할당이 이루어지면, 예시적인 가상 머신 환경(271)과 같은 가상 머신의 물리적 주소 공간을 관리할 구성 요소와 함께 등록되고, 호스트 메모리 관리자(241)가 가상 메모리 할당을 지원하도록 선택할 호스트 물리적 메모리 페이지와 동기화를 유지할 수 있다. 이러한 구성 요소는 호스트 커널 및/또는 드라이버의 일부로서 구현될 수 있는 하이퍼바이저(210) 및 가상화 스택(230)일 수 있다. 하이퍼바이저(210)는 SLAT(220)를 활용하여 게스트 물리적 메모리 주소 범위와 대응하는 호스트 물리적 메모리 주소 범위 간의 트랜슬레이션을 관리할 수 있다. 특히, 가상화 스택(230)은 대응하는 게스트 물리적 메모리 페이지를 지원하는 호스트 물리적 메모리 페이지로 SLAT(220)를 업데이트할 수 있다. 실제 메모리 페이지. 하이퍼바이저(210)는 특정 액세스 유형이 예시적인 가상 머신(271)과 같은 게스트 가상 머신에 의해 수행될 때 가상화 스택(230)이 인터셉트를 수신하는 능력을 주어진 게스트 물리적 메모리 주소에 노출할 수 있다. 예를 들어, 가상화 스택(230)은 게스트 가상 머신(271)에 의해 특정 물리적 주소가 쓰여질 때 인터셉트 수신을 요청할 수 있다.

[0044] 일 양태에 따르면, 예시적인 가상 머신(271)과 같은 가상 머신이 처음 생성될 때, 이러한 가상 머신의 게스트 물리적 메모리 주소를 지원하기 위해 호스트 물리적 메모리 주소가 할당되지 않았기 때문에, SLAT(220)는 이러한 가상 머신에 대응하는 임의의 유효한 엔트리를 포함하지 않을 수 있다(아래에 나타내었지만, 일부 실시예에서, SLAT(220)는 가상 머신(271)의 생성과 동시에 또는 거의 동시에 일부 게스트 물리적 메모리 주소에 대해 사전에 파플레이팅될 수 있음). 하이퍼바이저(210)는 가상 머신(27)에 의해 활용될 게스트 물리적 메모리 주소 범위를 알 수 있지만, 이 지점에서 이들 중 어느 것도 임의의 호스트 물리적 메모리에 의해 지원될 필요가 없다. 가상 머신 프로세스(270)가 실행을 시작할 때, 그(게스트) 물리적 메모리 페이지에 액세스하기 시작할 수 있다. 각각의 새로운 물리적 메모리 주소에 액세스될 때, 대응하는 SLAT 엔트리가 대응하는 호스트 물리적 메모리 주소로 아직 파플레이팅되지 않았으므로, 적절한 유형(읽기/쓰기/실행)의 인터셉트를 생성할 수 있다. 하이퍼바이저(210)는 게스트 액세스 인터셉트를 수신하고 이를 가상화 스택(230)으로 전달할 수 있다. 차례로, 가상화 스택(230)은 요청된 게스트 물리적 메모리 주소 범위(및 그 가상 주소 공간 지원이 예시적인 가상 머신 프로세스(270)와 같은 것으로부터 할당된 호스트 프로세스)에 대응하는 호스트 가상 메모리 주소 범위를 찾기 위해 가상화 스택(230)에 의해 유지되는 데이터 구조를 참조할 수 있다. 이 지점에서, 가상화 스택(230)은 인터셉트를 생성한 게스트 물리적 메모리 주소에 대응하는 특정 호스트 가상 메모리 주소를 알 수 있다.

[0045] 그 후, 가상화 스택(230)은 가상 머신 프로세스(270)와 같은 가상 주소 범위를 호스팅하는 프로세스의 컨텍스트에서 호스트 메모리 관리자(241)에 가상 결함을 발행할 수 있다. 가상 결함은 가상 머신(271)이 게스트 물리적 메모리의 물리적 주소에 액세스했을 때 발생한 원래 인터셉트의 대응하는 액세스 유형(읽기/쓰기/실행)으로 발행될 수 있다. 가상 결함은 특정된 가상 주소를 유효하게 하고 호스트 CPU에 의해 액세스할 수 있도록 하기 위해 정규 페이지 결함이 취하는 것과 유사하거나 동일한 코드 경로를 실행할 수 있다. 하나의 차이점은 가상

결합 코드 경로는 메모리 관리자(241)가 가상 주소를 유효하게 만드는 데 사용한 물리적 페이지 번호를 반환할 수 있다는 것일 수 있다. 이러한 물리적 페이지 번호는 호스트 가상 주소를 지원하는 호스트 물리적 메모리 주소일 수 있으며, 이는 차례로 하이퍼바이저(210)에서 원래 액세스 인터셉트를 생성한 게스트 물리적 메모리 주소를 지원한다. 이 지점에서, 가상화 스택(230)은 호스트 물리적 메모리 주소와 호스트에서 가상 주소를 유효하게 만드는 데 사용되었던 액세스 유형(읽기/쓰기/실행)으로 인터셉트를 생성한 원래 게스트 물리적 메모리 주소에 대응하는 SLAT(220)에서 테이블 엔트리를 생성하거나 업데이트할 수 있다. 이것이 수행되면, 게스트 가상 머신(271)에 대한 액세스 유형으로 게스트 물리적 메모리 주소에 즉시 액세스할 수 있다. 예를 들어, 게스트 가상 머신(271)의 병렬 가상 프로세서는 인터셉트를 히팅(hitting)하지 않고 이러한 주소에 즉시 액세스할 수 있다. 원래 인터셉트 처리가 완료될 수 있으며, 인터셉트를 생성한 원래 가상 프로세서는 명령을 재시도하고 메물 액세스를 진행하여 이제 SLAT는 요청된 게스트 물리적 주소를 대응하는 호스트 물리적 메모리 주소와 상관시킬 수 있는 테이블 엔트리를 포함한다.

[0046] 호스트 메모리 관리자(241)가 가상 결합을 통해 유효하게 된 호스트 가상 주소의 호스트 물리적 주소 지원을 변경할 수 있거나 변경할 임의의 액션을 수행하기로 결정한 경우 및/또는 그러한 때, 해당 호스트 가상 주소에 대해 트랜슬레이션 버퍼(TLB: translation buffer) 플러시(flush)를 수행할 수 있다. 호스트 메모리 관리자(241)가 호스트 상의 하드웨어 CPU와 가질 수 있는 기존 계약을 준수하기 위해 이미 이러한 액션을 수행할 수 있다. 가상화 스택(230)은 이러한 TLB 플러시를 인터셉팅하고 임의의 가상 머신에서 임의의 게스트 물리적 메모리 주소를 지원하는 플러싱된 임의의 호스트 가상 주소의 대응하는 SLAT 엔트리를 무효화할 수 있다. TLB 플러시 호출은 플러싱되는 가상 주소 범위를 식별할 수 있다. 그 후, 가상화 스택(230)은 주어진 호스트 가상 주소에 의해 지원될 수 있는 게스트 물리적 범위를 찾기 위해 호스트 가상 주소에 의해 인덱싱될 수 있는 데이터 구조에 대해 플러싱되는 호스트 가상 주소를 조회할 수 있다. 이러한 임의의 범위가 발견되면, 해당 게스트 물리적 메모리 주소에 대응하는 SLAT 엔트리가 무효화될 수 있다. 또한, 호스트 메모리 관리자는 (예를 들어, SLAT 무효화 시간, 후속 메모리 인터셉트 등을 감소시키기 위해) TLB 플러시 거동을 최적화하기 위해 필요하거나 원하는 경우 가상 메모리를 다르게 지원하는 가상 할당을 처리할 수 있다.

[0047] 가상화 스택(230)은 (메모리 관리자(241)에 의해 발행된) 호스트에 의해 수행된 TLB 플러시에 대해 가상 결합 (메모리 관리자(241)에 의해 서비스됨)으로부터 반환된 호스트 물리적 메모리 페이지 번호와 SLAT(220)의 업데이트를 주의 깊게 동기화할 수 있다. 이는 호스트 메모리 관리자(241)와 가상화 스택(230) 사이에 복잡한 동기화를 추가하는 것을 피하기 위해 수행될 수 있다. 가상 결합에 의해 반환된 물리적 페이지 번호는 가상화 스택(230)으로 반환될 때 무효일 수 있다. 예를 들어, 가상 주소가 이미 무효화되었을 수 있다. 호스트 메모리 관리자(241)로부터 TLB 플러시 호출을 인터셉팅함으로써, 가상화 스택(230)은 이러한 레이스가 언제 발생했는지를 알 수 있고 업데이트된 물리적 페이지 번호를 획득하기 위해 가상 결합을 재시도할 수 있다.

[0048] 가상화 스택(230)이 SLAT 엔트리를 무효화할 때, 가상 머신(271)에 의한 해당 게스트 물리적 메모리 주소에 대한 후속 액세스는 다시 하이퍼바이저(210)에 대한 인터셉트를 생성할 것이며, 이는 결국 가상화 스택(230)으로 전달되어 상술한 바와 같이 해결될 것이다. 게스트 물리적 메모리 주소를 먼저 읽기 위해 액세스된 다음 나중에 쓸 때 동일한 프로세스가 반복될 수 있다. SLAT 엔트리는 "읽기" 액세스 유형으로만 유효하게 만들어질 수 있기 때문에, 쓰기는 별도의 인터셉트를 생성할 것이다. 해당 인터셉트는 통상과 같이 가상화 스택(230)으로 전달될 수 있고 "쓰기" 액세스를 갖는 가상 결합이 적절한 가상 주소를 위해 호스트 메모리 관리자(241)에 발행될 수 있다. 호스트 메모리 관리자(241)는 호스트 물리적 메모리 페이지가 이제 더러움을 나타내기 위해 내부 상태(통상적으로 페이지 테이블 엔트리(또는 "PTE"))를 업데이트할 수 있다. 이는 가상 시스템이 게스트 물리적 메모리 주소에 쓰기를 허용하기 전에 수행될 수 있으며, 이에 의해 데이터 손실 및/또는 손상을 피할 수 있다. 호스트 메모리 관리자(241)가 (TLB 플러시를 수행하고 결과로서 대응하는 SLAT 엔트리를 무효화할) 가상 주소를 트리밍하기로 결정한 경우 및/또는 그러한 때, 호스트 메모리 관리자(241)는 페이지가 더럽고 용도가 바뀌기 전에 하드 디스크 드라이브의 페이지파일과 같은 비휘발성 컴퓨터 관독 가능 저장 매체에 쓰여질 필요가 있음을 알 수 있다. 이러한 방식으로, 상술한 시퀀스는 호스트 컴퓨팅 디바이스(100) 상에서 실행되는 임의의 다른 프로세스에 대한 정규의 프라이빗 가상 할당에 대해 발생하는 것과 다르지 않다.

[0049] 하나의 최적화로서, 호스트 메모리 관리자(241)는 모든 메모리(120)에 대해 페이지 코밍(combining) 패스를 수행하도록 선택할 수 있다. 이는 호스트 메모리 관리자(241)가 모든 프로세스에 걸쳐 동일한 페이지를 찾아서 이를 모든 프로세스가 공유하는 페이지의 하나의 관독-전용 사본으로 결합하는 동작일 수 있다. 결합된 가상 주소들 중 임의의 것이 쓰여지는 경우 및/또는 이러한 때, 메모리 관리자(241)는 쓰기가 진행될 수 있도록 쓰기-시-복사 동작을 수행할 수 있다. 이러한 최적화는 이제 예시적인 가상 머신(261 및 271)과 같은 가상 머신에

걸쳐 투명하게 작동할 수 있어, 가상 머신에 걸쳐 동일한 페이지를 결합하여 주어진 컴퓨팅 디바이스 상에서 실행되는 가상 머신의 밀도를 증가시키고, 이에 의해 메모리 소비를 감소시킨다. 페이지 결합이 발생할 때, 호스트 메모리 관리자(241)는 영향을 받은 가상 주소를 매핑하는 PTE를 업데이트할 수 있다. 이러한 업데이트 중에, 호스트 물리적 메모리 주소가 고유한 프라이빗 페이지로부터 공유된 페이지로의 해당 가상 주소에 대해 변경할 수 있으므로, TLB 플러시를 수행할 수 있다. 이의 일부로서, 상술한 바와 같이, 가상화 스택(230)은 대응하는 SLAT 엔트리를 무효화할 수 있다. 가상 주소가 공유된 페이지를 가리키도록 결합된 게스트 물리적 메모리 주소를 읽는 경우 및/또는 읽을 때, 가상 결합 해결책은 인터셉트 처리 중에 공유된 페이지의 물리적 페이지 번호를 반환할 수 있으며 SLAT(220)은 공유된 페이지를 가리키도록 업데이트될 수 있다.

[0050] 결합된 게스트 물리적 메모리 주소 중 임의의 주소가 가상 머신에 의해 쓰여지는 경우, 쓰기 액세스를 갖는 가상 결합은 쓰기-시-복사 동작을 수행할 수 있으며 새로운 프라이빗 호스트 물리적 메모리 페이지 번호가 반환되고 SLAT(220)에서 업데이트될 수 있다. 예를 들어, 가상화 스택(230)은 페이지 결합 패스를 수행하도록 호스트 메모리 관리자(241)에 지시할 수 있다. 일부 실시예에서, 가상화 스택(230)은 메모리의 어느 부분이 코딩을 위해 스캔되어야 하는지 또는 어떤 프로세스가 스캔되어야 하는지를 특정할 수 있다. 예를 들어, 가상화 스택(230)은 호스트 가상 메모리 할당이 결합을 위해 스캔되어야 하는 프로세스인 것으로 대응하는 가상 머신, 즉, 예시적인 가상 머신(261 및 271)의 게스트 물리적 메모리를 지원하는 예시적인 가상 머신 프로세스(260 및 270)와 같은 프로세스를 식별할 수 있다.

[0051] 쓰기를 위해 수행되는 가상 결합으로 인해 쓰기 액세스를 허용하도록 SLAT(220)이 업데이트되는 경우에도, 하이퍼바이저(210)는 가상화 스택(230)에 의해 요청되는 경우 이러한 SLAT 엔트리에 대한 쓰기에 대한 인터셉트 트리거링을 지원할 수 있다. 가상화 스택(230)은 이러한 쓰기가 발생하기 위해 호스트 메모리 관리자(241)에 수용 가능하다는 사실에 관계없이 쓰기가 발생하는 때를 알고 싶어할 수 있으므로, 이는 유용할 수 있다. 예를 들어, 가상 머신의 라이브 마이그레이션(migration) 또는 가상 머신 스냅샷팅(snapshotting)은 쓰기를 모니터링하는 가상화 스택(230)을 수반할 수 있다. 가상화 스택(230)은 예를 들어, PTE가 이미 더러운 것으로 마킹된 경우와 같이 쓰기를 위해 그에 따라 호스트 메모리 관리자의 상태가 이미 업데이트된 경우에도 쓰기가 발생할 때를 여전히 통지받을 수 있다.

[0052] 호스트 메모리 관리자(241)는 임의의 다른 프로세스 주소 공간에 할당된 정규 호스트 가상 페이지에 대해 하는 것처럼 게스트 물리적 메모리 주소 공간을 지원하는 각각의 호스트 가상 페이지에 대한 정확한 액세스 이력을 유지할 수 있다. 예를 들어, PTE의 "액세스된 비트"는 메모리 인터셉트 처리의 일부로서 수행되는 가상 결합 중에 업데이트될 수 있다. 호스트 메모리 관리자가 임의의 PTE 상에서 액세스된 비트를 클리어할 때, 이는 메모리 손상을 피하기 위해 이미 TLB를 플러싱할 수 있다. 상술한 바와 같이, 이러한 TLB 플러시는 대응하는 SLAT 엔트리를 무효화할 수 있으며, 이는 결국 가상 머신이 게스트 물리적 메모리 주소에 다시 액세스하는 경우 액세스 인터셉트를 생성할 수 있다. 인터셉트 처리의 일부로서, 호스트 메모리 관리자(241)의 가상 결합 프로세싱은 액세스된 비트를 다시 설정하여, 페이지에 대한 적절한 액세스 이력을 유지한다. 대안적으로, 하이퍼바이저(210)에서의 액세스 인터셉트를 가능한 한 피하는 것과 같은 성능상의 이유로, 호스트 메모리 관리자(241)는 기본 하드웨어에 의해 지원되는 경우 SLAT 엔트리로부터 수집된 하이퍼바이저(210)로부터 직접 페이지 액세스 정보를 소비할 수 있다. 호스트 메모리 관리자(241)는 가상화 스택(230)과 협력하여 (게스트 물리적 메모리 주소에 의해 조직된) SLAT(220)의 액세스 정보를 어떠한 주소가 액세스되었는지를 알기 위해 해당 게스트 물리적 메모리 주소를 지원하는 호스트 가상 메모리 주소로 트랜슬레이팅할 수 있다.

[0053] 페이지에 대한 정확한 액세스 이력을 가짐으로써, 호스트 메모리 관리자(241)는 프로세스의 작업 세트의 일반적인 지능형 에이징 및 트리밍 알고리즘을 실행할 수 있다. 이는 호스트 메모리 관리자(241)가 전체 시스템의 상태를 조사할 수 있게 하고, 필요한 경우 또는 다른 이유로 메모리 압력을 완화하기 위해 디스크로 트리밍 및/또는 페이지 아웃할 주소 또는 다른 유사한 동작에 대한 지능적인 선택을 할 수 있다.

[0054] 일부 실시예에서, 가상 및 물리적 메모리 제한은 시스템 상의 임의의 다른 프로세스와 마찬가지로 호스트 메모리 관리자(241)에 의해 가상 머신에 부과될 수 있다. 이는 시스템 관리자 샌드박스를 돕거나, 예시적인 가상 머신(261 및 271)과 같은 가상 머신을 다르게 제한하거나 활성화할 수 있다. 호스트 시스템은 네이티브 프로세스에 대한 것과 같이 이를 달성하기 위해 동일한 메커니즘을 사용할 수 있다. 예를 들어, 더 높은 성능이 요구되는 일부 실시예에서, 가상 머신은 호스트 물리적 메모리에 의해 직접 지원되는 게스트 물리적 메모리 전체를 가질 수 있다. 대안적으로, 일부는 호스트 가상 메모리에 의해 지원될 수 있는 반면, 다른 부분은 호스트 물리적 메모리(120)에 의해 지원된다. 더 낮은 성능이 허용될 수 있는 또 다른 예로서, 가상 머신은 주로 호스트 가상 메모리에 의해 주로 지원될 수 있으며, 호스트 가상 메모리는 호스트 물리적 메모리에 의한 전체 지원보다

적게 제한될 수 있다. 예를 들어, 예시적인 가상 머신(271)과 같은 가상 머신은 4 GB 크기의 게스트 물리적 메모리를 제시할 수 있으며, 이는 결국 상술한 프로세스를 활용하는 것이 4 GB의 호스트 가상 메모리에 의해 지원될 수 있다. 그러나, 호스트 가상 메모리는 2 GB의 호스트 물리적 메모리에 의해서만 지원되도록 제한될 수 있다. 이는 디스크로의 페이징 또는 다른 성능 장애를 야기할 수 있지만, 관리자가 서비스 레벨에 기초하여 제한하거나 가상 머신 배포에 대해 다른 제어를 행하는 방법이 될 수 있다. 마찬가지로, 특정한 일관된 레벨의 성능을 제공하기 위해 호스트 메모리 관리자에 의해 지원되는 것처럼 가상 머신에 특정량의 물리적 메모리가 보장될 수 있다(한편, 여전히 가상 머신이 호스트 가상 메모리에 의해 지원될 수 있게 함).

[0055] 가상 머신 환경 내에서 제시되는 게스트 물리적 메모리의 양은 동적으로 변할 수 있다. 게스트 물리적 메모리를 추가될 필요가 있는 경우, 상술한 바와 같이 다른 호스트 가상 메모리 주소 범위가 할당될 수 있다. 가상화 스택(230)이 메모리에 대한 액세스 인터셉트를 처리할 준비가 되면, 게스트 물리적 메모리 주소 범위가 가상 머신 환경에 추가될 수 있다. 게스트 물리적 메모리 주소를 제거할 때, 제거된 게스트 물리적 메모리 주소를 지원하는 호스트 가상 주소 범위의 부분은 호스트 메모리 관리자(241)에 의해 해제될 수 있다(그리고 가상화 스택(230) 데이터 구조에서 그에 따라 업데이트됨). 대안적으로 또는 추가적으로, 다양한 호스트 메모리 관리자 API가 이를 위해 호스트 가상 주소 공간을 해제하지 않고 호스트 가상 주소 범위의 해당 부분 상에서 호출되어 호스트 물리적 메모리 페이지를 해제할 수 있다. 대안적으로, 일부 실시예에서, 호스트 메모리 관리자(241)가 결국 이들 페이지를 작업 세트로부터 트리밍하고 더 이상 실제로 액세스되지 않을 것이기 때문에 페이지파일과 같은 디스크에 결국 이를 쓸 수 있으므로 아무것도 하지 않는 것이 가능하다.

[0056] 일 양태에 따르면, SLAT(220)는 물리적 메모리 주소 매핑을 호스트하기 위한 게스트 물리적 메모리 주소의 일부 또는 전부로 사전에 파플레이팅될 수 있다. 이는 가상 머신 초기화에서 수행되는 결합 처리 동작의 수를 줄일 수 있다. 그러나, 가상 머신이 동작함에 따라 SLAT(220)의 엔트리는 다양한 이유로 무효화될 수 있으며, 상술한 결합 처리가 사용되어 게스트 물리적 메모리 주소를 호스트 물리적 메모리 주소에 다시 한 번 상관시킨다. SLAT 엔트리는 가상 컴퓨팅 환경의 부팅 전에 또는 런타임에 사전에 파플레이팅될 수 있다. 또한, 전체 SLAT(220) 또는 일부만 사전에 파플레이팅될 수 있다.

[0057] 또 다른 최적화는 호스트 가상 메모리의 추가 부분을 호스트 물리적 메모리로 사전-페칭(fetching)하고, 이전에 페이징 아웃된 게스트 물리적 메모리를 지원하는 것일 수 있어, 후속 메모리 인터셉트가 도착할 때, 디스크로 이동하여 데이터를 읽을 필요 없이 충족될 수 있으므로 가상 결합이 더욱 신속하게 충족될 수 있다.

[0058] 도 4에 있어서, 위에서 상세히 설명된 메커니즘의 개요를 제공하는 방법(400)이 예시된다. 본 방법(400)은 호스트 가상 메모리로 게스트 물리적 메모리를 지원하는 동작을 포함할 수 있다. 본 방법은 호스트 컴퓨팅 디바이스 상에서 실행되는 가상 머신으로부터 게스트 물리적 메모리 액세스를 사용하여 게스트 물리적 메모리 액세스를 시도하는 것을 포함한다(동작 402). 예를 들어, 도 2에 나타난 가상 머신 환경(271)은 가상 머신 환경(271) 내에서 실행되는 프로세스에 실제 물리적 메모리로 보이는 게스트 물리적 메모리에 액세스할 수 있다.

[0059] 본 방법(400)은 게스트 물리적 메모리 액세스가 게스트 물리적 메모리 주소를 호스트 물리적 메모리 주소와 상관시키는 데이터 구조에서 유효한 엔트리를 갖지 않는 게스트 물리적 메모리 주소를 참조하는 것으로 결정하는 것을 추가로 포함할 수 있다(동작 404). 예를 들어, 도 2에 나타난 바와 같이, SLAT(220)에 유효한 엔트리가 없는 것으로 결정이 이루어질 수 있다.

[0060] 그 결과, 본 방법(400)은 게스트 물리적 메모리 주소에 대응하는 호스트 가상 메모리 주소를 식별하고 호스트 가상 메모리 주소에 대응하는 호스트 물리적 메모리 주소를 식별하는 것을 포함할 수 있다(동작 406). 예를 들어, 도 2에 나타난 가상화 스택(230)은 게스트 물리적 메모리 주소에 대응하는 호스트 가상 메모리 주소를 식별할 수 있으며, 또한 도 2에 나타난 메모리 관리자(241)는 호스트 가상 메모리 주소에 대응하는 호스트 물리적 메모리 주소를 식별할 수 있다.

[0061] 본 방법(400)은 게스트 물리적 메모리 주소와 식별된 호스트 물리적 메모리 주소의 상관 관계로 호스트 물리적 메모리 주소와 게스트 물리적 메모리 주소를 상관시키는 데이터 구조를 업데이트하는 것을 추가로 포함할 수 있다(동작 408). 예를 들어, 도 2에 나타난 가상화 스택(230)은 또한 도 2에 나타난 메모리 관리자(241)로부터 호스트 물리적 메모리 주소를 획득할 수 있고, 게스트 물리적 메모리 주소와 식별된 호스트 물리적 메모리 주소의 상관 관계로 (도 2에 나타난) SLAT(220)를 업데이트할 수 있다.

[0062] 본 방법(400)은 인터셉트를 유발함으로써 실시될 수 있다. 인터셉트는 호스트 상의 가상화 스택으로 전달될 수 있다. 이로 인해 가상화 스택이 게스트 물리적 메모리 주소에 해당하는 호스트 가상 메모리 주소를 식별하고

호스트 가상 메모리 주소에 대응하는 호스트 물리적 메모리 주소를 얻기 위해 메모리 관리자에 결함을 발행할 수 있다. 그 후, 가상화 스택은 게스트 물리적 메모리 주소와 식별된 호스트 물리적 메모리 주소의 상관 관계로 게스트 물리적 메모리 주소를 호스트 물리적 메모리 주소와 상관시키는 데이터 구조를 업데이트할 수 있다.

[0063] 본 방법(400)은 게스트 물리적 메모리 액세스에 대한 유형을 결정하고 게스트 물리적 메모리 주소와 식별된 호스트 물리적 메모리 주소에 상관된 결정된 유형으로 게스트 물리적 메모리 주소를 호스트 물리적 메모리 주소와 상관시키는 데이터 구조를 업데이트하는 것을 추가로 포함할 수 있다. 예를 들어, 게스트 물리적 메모리 액세스가 읽기인 경우, SLAT가 업데이트되어 이를 나타낼 수 있다.

[0064] 본 방법(400)은 호스트 가상 메모리 주소를 지원하는 호스트 물리적 메모리 주소를 변경할 수 있는 액션을 수행하는 것을 추가로 포함할 수 있다. 그 결과, 본 방법은 게스트 물리적 메모리 주소를 호스트 물리적 메모리 주소와 상관시키는 데이터 구조에서 호스트 물리적 메모리 주소와 게스트 물리적 메모리 주소를 상관시키는 엔트리를 무효화하는 것을 포함할 수 있다. 이로 인해 게스트 물리적 메모리 주소에 대한 후속 액세스가 게스트 물리적 메모리 주소를 지원하는 호스트 가상 메모리에 대한 올바른 상관 관계로 게스트 물리적 메모리 주소를 호스트 물리적 메모리 주소와 상관시키는 데이터 구조를 업데이트하는 데 사용할 수 있는 결함을 생성할 수 있다. 예를 들어, 액션은 페이지 결함 동작을 포함할 수 있다. 페이지 결함은 호스트 상의 가상 머신의 밀도를 증가시키는 데 사용될 수 있다.

[0065] 본 방법(400)은 게스트 가상 머신을 초기화하는 것을 포함할 수 있다. 게스트 가상 머신 초기화의 일부로서, 본 방법(400)은 게스트 가상 머신에 대한 호스트 물리적 메모리 주소 매핑에 대한 게스트 물리적 메모리 주소의 일부 또는 전부로 게스트 물리적 메모리 주소를 호스트 물리적 메모리 주소와 상관시키는 데이터 구조의 적어도 일부를 사전에 파플레이팅하는 것을 포함할 수 있다. 따라서, 예를 들어, 호스트 물리적 메모리는 가상 머신 및 SLAT로 입력된 적절한 상관 관계에 대해 사전-할당될 수 있다. 이는 게스트 가상 머신을 초기화하는 데 필요한 더 적은 예외로 귀결될 것이다.

[0066] 이제 도 5를 참조하면, 예시적인 게스트 물리적 메모리 주소(이하 "GPA"로 약칭) 0x1000에서 일부 데이터의 수명의 일부에 걸쳐 발생할 수 있는 다양한 동작을 나타내는 예시적인 흐름(500)이 나타내어진다. 502에 나타난 바와 같이, GPA 0x1000에 대한 SLAT 엔트리는 유효하지 않으며, 이는 해당 특정 GPA에 대해 SLAT 엔트리가 없음을 의미한다. 504에 나타난 바와 같이, 가상 머신은 GPA 0x1000에서 읽기를 수행하려고 시도하여 가상 머신(이하 "VM"으로 약칭) 읽기 인터셉트를 야기한다. 506에서 나타난 바와 같이, 하이퍼바이저는 호스트 상의 가상화 스택에 인터셉트를 전달한다. 508에서, 가상화 스택은 GPA 0x1000에 대응하는 호스트 가상 메모리 주소(VA)에 대한 가상화 룩업을 수행한다. 룩업은 VA 0x8501000을 산출한다. 510에서, VA 0x8501000에 대한 읽기 액세스에 대한 가상 결함이 생성된다. 메모리 관리자로부터의 가상 결함 처리는 GPA 0x1000의 데이터가 물리적으로 위치한 시스템 메모리의 주소를 정의하는 0x88000의 시스템 물리적 주소(이하 "SPA"라고 약칭)를 반환한다. 따라서, 514에 나타난 바와 같이, SLAT는 업데이트되어 GPA 0x1000과 SPA 0x88000를 상관시키고 데이터 액세스를 "읽기 전용"으로 마킹한다. 516에서, 가상화 스택은 읽기 인터셉트 처리를 완료하고 하이퍼바이저는 게스트 가상 머신 실행을 재개한다.

[0067] 518에 나타난 바와 같이, 시간이 경과한다. 520에서 가상 머신은 GPA 0x1000에서 쓰기 액세스를 시도한다. 522에서, 하이퍼바이저는 호스트 상의 가상화 스택에 쓰기 액세스를 전달한다. 524에서, 가상화 스택은 GPA 0x1000에 대한 호스트 VA에 대한 가상화 룩업을 수행한다. 이전에 언급한 바와 같이, 이는 VA 0x8501000에 있다. 526에서, VA 0x8501000 상의 쓰기 액세스에 대해 가상 결함이 발생한다. 528에서, 가상 결함은 물리적 메모리에 SPA 0388000을 반환한다. 530에서, GPA 0x1000에 대한 SLAT 엔트리가 업데이트되어 데이터 액세스가 "읽기/쓰기"임을 나타낸다. 532에서, 가상화 스택은 쓰기 인터셉트 처리를 완료한다. 하이퍼바이저는 게스트 가상 머신 실행을 재개한다.

[0068] 534에 나타난 바와 같이, 약간의 시간이 경과한다. 536에서, 호스트 메모리 관리자는 페이지 결함 패스를 실행하여 기능적으로 동일한 호스트 물리적 메모리의 임의의 페이지를 결함한다. 538에서, 호스트 메모리 관리자는 VA 0x8501000 및 다른 프로세스의 다른 가상 주소에 대한 결함 후보를 찾는다. 540에서, 호스트는 VA 0x8501000에 대해 TLB 플러시를 수행한다. 542에서, 가상화 스택은 TLB 플러시를 인터셉팅한다. 544에서, GPA 0x1000에 대한 SLAT 엔트리는 유효하지 않다. 546에서, GPA 0x1000에 대해 가상 머신 인터셉트가 수행된다. 548에서, VA 0x8501000에 대한 읽기 액세스에 대한 가상 결함이 발생한다. 550에서, 가상 결함은 SPA 0x52000을 반환하며, 이는 536에서 페이지 결함 패스로부터의 N 프로세스들 간의 공유된 페이지이다. 552에서, GPA 0x1000에 대한 SLAT 엔트리가 업데이트되어 SPA 0x52000과 상호되고, 액세스는 "읽기 전용"으로 설정된다.

- [0069] 554에 나타난 바와 같이, 약간의 시간이 경과한다. 556에서, GPA 0x1000에 대해 가상 머신 쓰기 인터셉트가 발생한다. 558에서, VA 0x8501000에 대한 쓰기 액세스에 대한 가상 결함이 발생한다. 560에서, 호스트 메모리 관리자는 VA 0x8501000에서 쓰기-시-복제를 수행한다. 562에서, 호스트는 VA 0x850100에 대해 TLB 플러시를 수행한다. 564에 나타난 바와 같이, 이로 인해 GPA 031000에 대한 SLAT 엔트리가 무효화된다. 566에서, 가상 결함은 쓰기-시-복제 후의 프라이빗 페이지인 SPA 0x11000을 반환한다. 568에서, GPA 0x1000에 대한 SLAT 엔트리가 SPA 0x1000으로 업데이트되고 액세스는 "읽기/쓰기"로 설정된다. 570에서, 가상화 스택은 읽기 인터셉트 처리를 완료하고, 하이퍼바이저는 가상 머신 실행을 재개한다.
- [0070] 따라서, 위에 나타난 바와 같이, 가상 머신 물리적 주소 공간은 호스트 가상 메모리(통상적으로 호스트 프로세스의 사용자 주소 공간에 할당됨)에 의해 지원되며, 이는 호스트 메모리 관리자에 의해 정규 가상 메모리 관리를 받는다. 가상 머신의 물리적 메모리를 지원하는 가상 메모리는 호스트 메모리 관리자(118)에 의해 지원되는 임의의 유형일 수 있다(프라이빗 할당, 파일 매핑, 페이지파일-지원 섹션 매핑, 큰 페이지 할당 등). 호스트 메모리 관리자는 기존 동작을 수행할 수 있으며 가상 메모리에 정책을 적용하고/적용하거나 필요에 따라 가상 메모리가 가상 머신의 물리적 주소 공간을 지원하고 있음을 알고 있는 특수 정책을 적용할 수 있다.
- [0071] 도 6에 있어서, 그 내부에 나타난 예시적인 시스템(600)은 상술한 바와 같이, 도 4에 개요되고 도 5에 상세하게 설명된 메커니즘의 블록 표현을 포함한다. 보다 구체적으로, 게스트 가상 메모리(311), 게스트 물리적 메모리(312), 호스트 가상 메모리(321) 및 호스트 물리적 메모리(322)는 폭이 메모리의 범위를 나타내는 직사각형 블록으로 나타내어진다. 설명을 간단하게 하기 위해, 직사각형 블록은 대략 동등한 폭을 갖지만, 실제 동작에서, 가상 메모리의 양이 이러한 가상 메모리를 지원하는 물리적 메모리의 양을 초과할 가능성이 높다.
- [0072] 위에서 상세히 설명된 바와 같이, 도 2에 나타난 예시적인 가상 머신 컴퓨팅 환경(271)과 같은 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스가 이러한 가상 머신 컴퓨팅 환경 내에서 실행되는 운영 체제에 의해 제시된 가상 메모리의 일부에 액세스할 때, 예시적인 페이지 테이블(283)과 같은 해당 운영 체제의 페이지 테이블은 적어도 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스에 의해 인식되는 액세스된 가상 메모리 주소를 물리적 메모리 주소에 상관시킬 수 있는 PTE를 포함할 수 있다. 따라서, 예를 들어, 영역(611)으로 표현되는 가상 메모리가 액세스되면, 페이지 테이블(283)의 일부일 수 있는 PTE(631)는 게스트 가상 메모리(311)의 해당 영역(611)을 게스트 물리적 메모리(312)의 영역(621)과 상관시킬 수 있다. 위에서 추가로 상세히 설명된 바와 같이, 게스트 물리적 메모리(312)는 호스트 가상 메모리(321)의 일부에 의해 지원될 수 있다. 따라서, 상술한 가상화 스택(230)은 게스트 물리적 메모리(312)의 영역(621)을 호스트 가상 메모리(321)의 영역(641)과 상관시킬 수 있는 엔트리(651)를 포함할 수 있는 데이터 구조를 포함할 수 있다. 상술한 페이지 테이블(250)을 포함하는 호스트 컴퓨팅 디바이스 상에서 실행되는 프로세스는 그 후 호스트 가상 메모리(321)의 영역(641)을 호스트 물리적 메모리(322)의 영역(661)과 상관시킬 수 있다. 또한, 위에 상세히 설명한 바와 같이, SLAT(220)는 게스트 물리적 메모리(312)의 영역(621)을 호스트 물리적 메모리(322)의 영역(661)과 상관시킬 수 있는 테이블 엔트리(691)로 업데이트될 수 있는 계층적으로 배열된 테이블 세트를 포함할 수 있다. 보다 구체적으로, 가상화 스택(230)에 의해 구현된 것과 같은 메커니즘은 영역(621)과 영역(661) 사이의 상관 관계를 검출할 수 있고, 액션(680)에 의해 나타난 바와 같이, SLAT(220) 내의 엔트리(691)와 같은 테이블 엔트리를 업데이트 또는 생성하여 이러한 상관 관계를 포함할 수 있다. 이러한 방식으로, 게스트 물리적 메모리(312)는 호스트 가상 메모리(321)에 의해 지원될 수 있으며, SLAT(220)를 포함한 기존 메커니즘이 종래의 방식으로 기능하도록 계속 허용할 수 있다. 예를 들어, 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스의 게스트 가상 메모리(311)의 영역(611)의 후속 액세스는 통상적인 방식으로 2개의 페이지 테이블 룩업을 수반할 수 있다: (1) 요청된 영역(611)을 게스트 물리적 메모리(312)의 대응 영역(621)과 상관시킬 수 있는 엔트리(631)와 같은 엔트리를 포함할 수 있는 페이지 테이블(283)을 참조하여 가상 머신 컴퓨팅 환경 내에서 수행되는 페이지 테이블 룩업, 및 (2) 게스트 물리적 메모리(312)의 영역(621)을 호스트 물리적 메모리(322)의 영역(661)과 상관시킬 수 있는 엔트리(691)와 같은 엔트리를 포함할 수 있는 SLAT(220)를 참조하여 호스트 컴퓨팅 디바이스 상에서 수행되고, 이에 의해 호스트의 물리적 메모리에 저장된 관련 데이터에 대한 경로를 완성하는 페이지 테이블 룩업.
- [0073] 또한, 위에 상세히 설명한 바와 같이, 가상 메모리 컴퓨팅 환경의 예시적인 페이지 테이블(283)과 같은 다양한 페이지 테이블, 호스트 컴퓨팅 디바이스 상의 예시적인 페이지 테이블(250) 및 예시적인 SLAT(220)는 각각의 페이지 테이블을 활용하는 프로세스에 의해 개별적으로 그리고 이산적으로 유지 및 동작될 수 있는 메모리 주소의 최소 범위일 수 있는, 해당 용어가 본원에 명확하게 정의되는 메모리의 "작은 페이지"를 계층적으로 하위 계층의 테이블의 엔트리가 식별할 수 있도록 계층적으로 배열된 테이블의 계층을 포함할 수 있다. 도 6에 나타난 예시적인 시스템(600)에서, 다양한 페이지 테이블 엔트리(PTE)(631, 671 및 691)는 영역(611, 621, 641 및

661)이 작은 메모리 페이지가 될 수 있도록 테이블의 최하위 계층에 있을 수 있다. 위에 상세히 설명한 바와 같이, 하나의 공통 마이크로프로세서 아키텍처에서, 해당 용어가 본원에서 명시적으로 정의된 바와 같은 이러한 작은 메모리 페이지는 4 KB의 연속 메모리 주소를 포함할 수 있다.

[0074] 일 양태에 따르면, 물리적 메모리가 호스트 컴퓨팅 디바이스의 가상 메모리에 의해 지원되는 가상 머신 컴퓨팅 환경으로부터의 메모리 액세스의 속도 및 효율성을 증가시키기 위해, SLAT(220)는 테이블의 계층적으로 상위 계층에서 상관 관계를 유지할 수 있다. 즉, SLAT(220)는 해당 용어가 본원에서 명시적으로 정의된 바와 같은 메모리의 "큰 페이지" 또는 메모리의 "거대 페이지"를 상관시킬 수 있다. 하나의 공통 마이크로프로세서 아키텍처에서, 해당 용어가 본원에서 명시적으로 정의된 바와 같은, 큰 메모리 페이지는 2 MB의 연속 메모리 주소를 포함할 수 있고, 해당 용어가 본원에서 명시적으로 정의된 바와 같은 거대 메모리 페이지는 1 GB의 연속 메모리 주소를 포함할 수 있다. 다시, 앞서 나타난 바와 같이 "작은 페이지", "큰 페이지" 및 "거대 페이지"라는 용어는 계층적으로 배열된 페이지 테이블의 세트 내에서 계층적 계층을 참조하여 명시적으로 정의되며, 이러한 양의 메모리는 상이한 마이크로프로세서 아키텍처, 운영 체제 아키텍처 등 간에 달라질 수 있기 때문에, 메모리의 특정 양에 기초하여 정의되지 않는다.

[0075] 도 7에 있어서, SLAT(220)에서 큰 페이지 엔트리를 생성하여 호스트 컴퓨팅 디바이스의 가상 메모리에 의해 물리적 메모리가 지원되는 가상 머신 컴퓨팅 환경에서 메모리 액세스의 속도와 효율성을 증가시키기 위한 하나의 메커니즘이 나타내어진다. 도 6에서와 같이, 도 7에 나타난 시스템(700)에서, 게스트 가상 메모리(311), 게스트 물리적 메모리(312), 호스트 가상 메모리(321) 및 호스트 물리적 메모리(322)는 폭이 메모리 범위를 나타내는 직사각형 블록으로 나타내어진다. 또한, 영역(611)의 액세스는 위에서 상세히 설명한 방식으로, 그리고 도 6에 나타난 바와 같이 진행될 수 있다. 그러나, SLAT(220) 내에서 계층적으로 최하위-레벨의 테이블 엔트리를 생성하는 대신, 여기에 설명된 메커니즘은 예시적인 테이블 엔트리(711)와 같은 상위-레벨의 테이블 엔트리를 생성할 수 있다. 위에 상세히 설명한 바와 같이, 이러한 상위-레벨 테이블 엔트리는 메모리의 큰 페이지 또는 거대 페이지를 상관시킬 수 있다. 따라서, 가상화 스택(230)이 테이블 엔트리(711)가 SLAT(220)에 입력되게 할 때, 테이블 엔트리(711)는 게스트 물리적 메모리(312)의 큰 페이지 크기의 영역(761)을 호스트 물리적 메모리(322)의 큰 페이지 크기의 영역(720)과 상관시킬 수 있다. 그러나, 이하에서 상세히 설명되는 바와 같이, 이러한 큰 페이지 크기의 영역은 호스트 컴퓨팅 디바이스 상에서 실행되고 페이지 테이블(250)을 활용하는 메모리 관리자의 관점에서 대형 페이지가 아니며, 결과적으로, 이러한 큰 페이지 크기의 영역은 예를 들어, 단일의 큰 페이지로서 페이징 아웃되지 않거나, 아니면 호스트 컴퓨팅 디바이스 상에서 실행되는 메모리 관리자에 의해 단일의 분할 불가능한 큰 페이지로서 처리되지 않는다.

[0076] 반대로, 예시적인 게스트 물리적 메모리(312)와 같은 게스트 물리적 메모리가 예시적인 호스트 가상 메모리(321)와 같은 호스트 가상 메모리에 의해 지원될 수 있게 하는 위에서 상세히 설명된 메커니즘에 의해 제공되는 효율성, 밀도의 이점 및 다른 이점을 유지하기 위해, 호스트 상에서 실행되는 페이지 테이블(250)은 SLAT(220)의 상위-레벨 테이블 엔트리(711)와 유사한 상위-레벨 테이블 엔트리가 아니라, 도 7에 나타난 예시적인 하위-레벨 테이블 엔트리(671, 731 및 732)와 같은 하위-레벨 테이블 엔트리를 단지 포함할 수 있다. 보다 구체적으로, 예시적인 하위-레벨 테이블 엔트리(671, 731, 732 등)와 같은 페이지 테이블(250)의 특정 하위-레벨 테이블 엔트리 세트는 페이지 테이블(250)에서 하위-레벨 테이블 엔트리의 전체 등가 시퀀스에 의해 식별되는 호스트 물리적 메모리(322)의 전체 연속 범위가 SLAT(220)의 단일 상위-레벨 테이블 엔트리(711)와 동일한 메모리 범위를 포함하도록 호스트 물리적 메모리(322)의 연속 범위를 식별할 수 있다. 따라서, 하나의 예로서, 작은 페이지가 4 KB의 크기이고 큰 페이지가 2 MB의 크기인 마이크로프로세서 아키텍처에서, 페이지 테이블(250)의 512개의 연속하는 작은 페이지 엔트리는 총괄적으로 호스트 물리적 메모리(322)의 연속적인 2 MB를 식별할 수 있다.

[0077] 큰 페이지 크기의 메모리 영역 미만에 대해 가상 결함이 생성되었을 수 있기 때문에, 예시적인 하위-레벨 테이블 엔트리(671, 731 및 732)와 같은 하위-레벨 테이블 엔트리의 특정 엔트리만이 활용될 수 있으며, 나머지 하위-레벨 테이블 엔트리는 자유 메모리로서 마킹된 채로 유지될 수 있다. 예를 들어, 게스트 물리적 메모리(312)를 활용하는 가상 머신 컴퓨팅 환경 내에서 실행되는 어플리케이션은 초기에 하나의 작은 페이지 크기의 메모리 양만을 필요로 할 수 있다. 이러한 경우에, 단일 상위 레벨 테이블 엔트리(711)는 활용되는 큰 페이지 크기의 영역(720)과 같은 큰 페이지 크기의 메모리 영역을 식별할 수 있다. 그러나, 페이지 테이블(250)을 참조하는 호스트 컴퓨팅 디바이스의 메모리 관리자의 관점에서, 예시적인 하위 레벨 테이블 엔트리(671)와 같은 하위 레벨 테이블 엔트리 중 하나만이 활용되는 것으로 표시될 수 있으며, 나머지는 활용 가능한 것으로 표시된다. 다른 엔트리가 이용 가능한 것으로 처리되는 것을 방지하기 위해, 가상화 스택(230)은 예시적인 하위-레벨 테이블 엔트리(731, 732) 등과 같은 나머지 하위 레벨 테이블 엔트리도 사용되는 것으로 마킹할 수 있다. 가상

머신 컴퓨팅 환경 내에서 실행되는 어플리케이션이 추가적인 작은 페이지 크기의 메모리 양을 활용해야 할 필요가 있을 때, 가상화 스택(230)은 어플리케이션의 추가 메모리 수요를 충족시키기 위해, 이전에 사용되는 것으로 표시되었지만 실제로는 가상 머신 컴퓨팅 환경 내에서 실행되는 어플리케이션에 의해 아직 활용되지 않은 메모리 위치를 활용할 수 있다.

[0078] 일 양태에 따르면, 앞서 언급한 마킹과 같은 페이지 테이블(250)의 하위-레벨 테이블 엔트리들의 증가 시퀀스의 추적은 도 7의 액션(730)에 의해 나타낸 바와 같이, 가상화 스택(230)에 의해 조정될 수 있다. 이러한 경우, 호스트 상에서 실행되는 메모리 관리자가 이러한 하위-레벨 테이블 엔트리 중 하나 이상을 변경하면, 가상화 스택(230)은 상위-레벨 테이블 엔트리(711)와 같은 SLAT(220)의 대응 엔트리에 대한 적절한 변경을 트리거링할 수 있다.

[0079] SLAT(220)의 상위-레벨 테이블 엔트리(711)와 등가인 페이지 테이블(250)의 하위-레벨 테이블 엔트리의 시퀀스는 예시적인 범위(641, 741 및 742)와 같은 호스트 가상 메모리 범위의 시퀀스를 각각 예시적인 범위(661, 721 및 722)와 같은 호스트 물리적 메모리 범위와 상관시킬 수 있다. 결과적으로, 예시적인 큰 페이지 크기 영역(751)과 같은 호스트 가상 메모리(321)의 큰 페이지 크기 영역은 페이지 테이블(250)의 하위-레벨 테이블 엔트리, 즉, 예시적인 하위-레벨 테이블 엔트리(671, 731, 732 등)의 시퀀스를 통해, 예시적인 큰 페이지 크기 영역(720)과 같은 호스트 물리적 메모리(322)의 큰 페이지 크기 영역과 상관된다. 가상화 스택(230)은 호스트 가상 메모리 위치를 게스트 물리적 메모리 위치에 대해 조정할 수 있는 계층적으로 배열된 테이블과 같은 데이터 구조를 유지할 수 있다. 따라서, 가상화 스택은 호스트 가상 메모리(321)의 큰 페이지 크기 영역(751)을 게스트 물리적 메모리(312)의 대응하는 큰 페이지 크기 영역(761)과 상관시킬 수 있는 예시적인 데이터 엔트리(771)와 같은 하나 이상의 데이터 엔트리를 유지할 수 있다. 데이터 엔트리(771)가 단일 엔트리로서 나타내어져 있지만, 가상화 스택(230)은 작은 페이지로서 크기가 정해진 하위-부분을 포함하여, 영역(761)의 하위-부분을 영역(751)에 개별적으로 상관시키는 것과 같이, 단일 엔트리로서, 복수의 엔트리로서, 또는 그 조합으로 앞서 언급한 상관 관계를 유지할 수 있다. 사이클을 완료하기 위해, SLAT(220)의 앞서 언급한 상위-레벨 테이블 엔트리(711)는 그 후 게스트 물리적 메모리(312)의 큰 페이지 크기 영역(761)을 호스트 물리적 메모리(322)의 큰 페이지 크기 영역(720)과 상관시킬 수 있다. 다시, 단일 상위-레벨 테이블 엔트리(711)만을 갖는 SLAT(220)의 관점에서, 큰 페이지 크기 영역(720 및 761)은 (단일 상위-레벨 테이블 엔트리(711)에 따라) SLAT에 의해 단일로 처리되는 큰 페이지이며, 호스트, 페이지 테이블(250) 및 실제로 가상화 스택(230) 상에서 실행되는 메모리 관리자의 관점에서, 이러한 영역은 단일이 아니라 예를 들어, 페이지 테이블(250)의 하위-레벨 테이블 엔트리에 의해 식별되는 개별적으로 관리 가능한 작은 페이지 영역으로 구성된다.

[0080] 해당 용어가 본원에 정의된, 하나 이상의 큰 또는 거대 메모리 페이지와 동등한 게스트 물리적 메모리(312)의 연속적인 작은 메모리 페이지의 가용성은 게스트 물리적 메모리(312)가 물리적 메모리로서 대응하는 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스에 의해 인식되기 때문에 간단할 수 있다. 결과적으로, 아래의 설명은 호스트 물리적 메모리(322)의 연속성에 초점을 맞춘다. 또한, 게스트 물리적 메모리(312)가 물리적 메모리인 것으로 인식되기 때문에, ("읽기/쓰기/실행" 권한과 같은) 전체 메모리 액세스 권한은 게스트 물리적 메모리(312)의 전체 범위에 걸쳐 있고, 따라서, 페이지 테이블(250)의 하위-레벨 테이블 엔트리들 사이에 메모리 액세스 권한의 불연속성이 있을 가능성은 낮다.

[0081] 호스트 컴퓨팅 디바이스의 페이지 테이블(250)에 초점을 맞추면, 호스트 물리적 메모리(322)의 하나 이상의 큰 페이지 크기 영역이 사용되지 않은 상태로 남아있는 경우, 일 양태에 따라, 위에서 상세히 설명한 바와 같이, 가상 결합이 사용되지 않고 남아 있는 큰 페이지 크기 영역으로 기술된 메모리 범위 내의 작은 페이지에 대해 트리거링될 때, 큰 페이지 크기 영역 내의 나머지 작은 페이지, 예시적인 상위-레벨 테이블 엔트리(711)와 같은 단일 상위-레벨 테이블 엔트리는 SLAT(220)에서 생성될 수 있다.

[0082] 그러나, 상위-레벨 테이블 엔트리(711)에 대응하는 큰 페이지(720)는 페이지 테이블(250)의 관점에서, 예시적인 하위-레벨 테이블 엔트리(671, 731, 732 등)와 같은 하위-레벨 테이블 엔트리의 시퀀스로 인해, 예시적인 작은 페이지(661, 721, 722 등)와 같은 작은 페이지의 시퀀스이므로, 이러한 페이지 테이블(250)을 활용하는 메모리 관리 메커니즘은 각각의 작은 페이지를 개별적으로 처리할 수 있으며, 이는 이러한 작은 페이지를 하드 디스크 드라이브와 같은 비휘발성 저장 매체로 페이징하는 것을 포함할 수 있다. 이러한 경우에, 일 양태에 따르면, SLAT(220)의 상위-레벨 테이블 엔트리(711)는 하위-레벨 테이블 엔트리의 동등한 시퀀스로 대체될 수 있다. 보다 구체적으로, SLAT(220) 내에서 생성된 하위-레벨 테이블 엔트리는 연속적일 수 있으며, 메모리 관리자에 의해 페이징 아웃된 페이지에 대응하는 하위-레벨 테이블 엔트리 중 특정한 것이 누락되거나 생성되지 않을 수 있

다는 예외를 갖고, 총괄적으로 상위-레벨 테이블 엔트리(711)와 동일한 범위의 메모리 주소를 참조할 수 있다.

[0083]

다른 양태에 따르면, 큰 페이지 또는 거대 페이지 크기의 호스트 물리적 메모리(322)의 영역이 위에 상세히 설명한 바와 같이 가상 결합이 트리거링될 때 이용 가능하지 않은 경우, 이용 가능한 큰 페이지, 또는 거대 페이지 크기의 호스트 물리적 메모리의 영역은 연속하는 작은 페이지의 적절한 양을 어셈블링함으로써 인터셉트 시기에 구성될 수 있다. 예를 들어, 큰 페이지 크기 영역은 "자유 목록"에서 이용할 수 있는 연속적인 작은 페이지 또는 이용 가능한 호스트 물리적 메모리 페이지의 다른 유사한 열거로 구성될 수 있다. 또 다른 예로서, 충분한 양의 작은 페이지를 하나의 큰 페이지 크기 영역으로 어셈블링하는 데 필요한 연속성을 확립하기 위해 다른 프로세스에 의해 현재 활용 중인 작은 페이지가 필요한 경우, 이러한 작은 페이지는 이러한 다른 프로세스에서 얻을 수 있으며, 그 안에 포함된 데이터는 다른 작은 페이지로 전송되거나 디스크로 페이지징 아웃될 수 있다. 이러한 큰 페이지 크기 영역이 구성되면, SLAT(220)에서 예시적인 상위-레벨 테이블 엔트리(711)와 같은 상위-레벨 테이블 엔트리의 생성을 포함하여, 위에서 상세히 설명된 방식으로 가상 결합의 프로세싱이 진행될 수 있다.

[0084]

일부 경우에, 페이지 테이블(250)을 활용하는 데 있어서 호스트 컴퓨팅 디바이스 상에서 실행되는 메모리 관리자는 이용 가능한 큰 페이지 또는 거대한 페이지 크기 영역이 연속된 작은 페이지의 충분한 양으로 구성되는 배경 또는 기회적인 메커니즘을 구현할 수 있다. 예를 들어, 가상 결합이 트리거링될 때 이러한 메커니즘이 이용 가능한 큰 페이지 크기 영역으로 되는 정도까지, 프로세싱은 위에 나타난 방식으로 진행될 수 있다. 그러나, 이러한 메커니즘이 아직 이용 가능한 큰 페이지 크기 영역의 어셈블링을 완료하지 않은 정도까지, 이러한 큰 페이지 크기 영역의 구성은 더 높은 우선 순위로 인터셉트 시간에 완료될 수 있다. 예를 들어, 이러한 큰 페이지 크기 영역을 구성하는 프로세스는 더 높은 실행 우선 순위를 부여받을 수 있거나, 배경에서 전경으로 이동될 수 있거나, 그렇지 않으면 보다 적극적으로 실행될 수 있다.

[0085]

대안적으로, 가상 결합이 트리거링될 때 호스트 물리적 메모리의 큰 페이지 크기 영역을 이용할 수 없는 경우, 프로세싱은 복수의 작은 페이지를 활용하여 위에서 상세하게 설명한 대로 진행될 수 있다. 후속하여, 큰 페이지 크기 영역을 이용할 수 있게 될 때, (당시 큰 페이지 크기 영역을 사용할 수 없었기 때문에) 이전에 활용된 복수의 작은 페이지로부터의 데이터가 큰 페이지로 복제될 수 있으며, 이전에 활용된 작은 페이지는 후속적으로 해제될 수 있다. 이러한 경우에, SLAT(220)의 하위-레벨 테이블 엔트리는 예시적인 상위-레벨 테이블 엔트리(711)와 같은 하나의 상위-레벨 테이블 엔트리로 대체될 수 있다.

[0086]

또한, 큰 페이지 크기 영역이 구성되는 파라미터는 인터셉트 시간에 또는 기회적으로 SLAT 록업 동안 테이블의 계층적 계층의 스킵과 작은 페이지로부터 큰 페이지 크기 영역을 구성하는 데 있어 초기 지연으로 인해 더 빠른 메모리 액세스 간의 균형에 따라 변할 수 있다. 이러한 하나의 파라미터는 전체의 큰 페이지 크기 영역에 걸쳐 연속성을 완료하는 데 필요한 다른 작은 페이지의 트레이딩 또는 페이지징 아웃을 트리거링하기에 충분한 연속하는 작은 페이지의 양일 수 있다. 예를 들어, 큰 페이지를 구성하는 데 512개의 연속적인 작은 페이지가 필요하고 200개의 작은 페이지와 311개의 작은 페이지의 연속된 범위가 존재하고, 그 사이에 하나의 작은 페이지가 현재 다른 프로세스에 의해 활용되고 있는 경우, 이러한 작은 페이지는 페이지징 아웃되거나, 이용 가능한 다른 작은 페이지와 트레이딩될 수 있어, 512개의 연속하는 작은 페이지가 하나의 큰 페이지 크기 영역으로 구성되는 데 이용 가능할 수 있다. 이러한 예에서, 이용 가능한 작은 페이지의 조각화는 매우 낮을 수 있다. 반대로, 현재 다른 프로세스에 의해 활용 중인 다른 작은 페이지에 의해 지속적으로 중단되는 20개 내지 30개의 작은 페이지의 연속 범위는 매우 높은 조각화를 갖는 작은 페이지 범위를 설명할 수 있다. 이러한 고도로 조각화된 작은 페이지 세트는 여전히 트레이딩, 페이지징 아웃 또는 다르게 개편될 수 있어 큰 페이지 크기 영역과 동등한 연속적인 범위의 작은 페이지를 생성할 수 있으며, 이러한 노력은 상당히 더 오래 걸릴 수 있다. 따라서, 일 양태에 따르면, 조각화 임계값이 설정될 수 있으며, 이는 더 높거나 더 낮은 조각화를 갖는 작은 페이지 범위로부터 큰 페이지 크기 영역을 구성하려는 시도가 수행되는지를 기술할 수 있다.

[0087]

일 양태에 따르면, 큰 페이지 크기 영역의 구성은 메모리가 가상 머신 컴퓨팅 환경의 게스트 물리적 메모리를 지원하기 위해 활용되는 프로세스에 의한 작은 페이지의 기존 활용을 이용할 수 있다. 간단한 예로서, 메모리가 게스트 물리적 메모리(312)를 지원하는 프로세스에 의해 200개의 작은 페이지와 311개의 작은 페이지의 연속된 범위가 이미 활용되었고, 이러한 2개의 연속 범위 사이의 단지 하나의 작은 페이지가 큰 페이지와 동등한 작은 페이지의 연속 범위를 확립하는 데 필요한 경우, 페이지징 아웃 또는 트레이딩되는 것과 같이 작은 페이지가 해제되면, SLAT(220)의 대응하는 하위-레벨 테이블 엔트리가 무효화되고 동일한 메모리 범위를 포함하는 단일 상위-레벨 테이블 엔트리로 대체될 수 있음에도 불구하고, 복제될 필요가 있는 유일한 데이터는 하나의 작은 페

이지에 대응하는 데이터일 수 있으며, 나머지 데이터는 제자리에 머물 수 있다.

[0088] 작은 페이지의 연속 범위로부터 큰 페이지 크기의 메모리 영역을 재구성할 필요가 없도록 하기 위해, SLAT(220)의 단일 상위-레벨 테이블 엔트리(711)와 동등한 메모리 범위를 총괄하여 포함하는 페이지 테이블(250)의 연속한 하위-레벨 테이블 엔트리의 기존의 세트는 페이지징 아웃되는 것을 피할 수 있도록 락킹될 수 있다. 따라서, 예를 들어, 가상화 스택(230)은 호스트 물리적 메모리(661, 721, 722 등)의 대응 범위가 디스크로 페이지징되지 않고 저장된 데이터를 유지할 수 있도록 하위-레벨 테이블 엔트리(671, 731, 732 등)가 락킹되도록 요청할 수 있다. 일 양태에 따르면, 가상화 스택(230)에 의한 것과 같이, 이러한 엔트리가 락킹되도록 요청하는 결정은 다양한 요소에 의존할 수 있다. 예를 들어, 이러한 하나의 요소는 이러한 메모리 범위 내에 저장된 데이터의 활용 빈도일 수 있으며, 더 자주 활용되는 데이터는 엔트리가 락킹되는 것으로 귀결되고, 덜 자주 활용되는 데이터는 엔트리가 후속적으로 락킹되지 않는 것으로 귀결된다. 전체 시스템의 메모리 조각화의 양, 호스트 컴퓨팅 디바이스에서 가상 머신 컴퓨팅 환경의 더 큰 밀도에 대한 수요 및 다른 유사한 요소와 같은 다른 요소는 해당 페이지 테이블 엔트리를 요청할지 여부를 결정할 때 동일하게 참조될 수 있으며, SLAT 내의 상위-레벨 테이블 엔트리가 락킹되는 것을 지원한다.

[0089] 도 8에 있어서, 여기에 나타난 흐름도(800)는 예시적인 일련의 단계로서 상술한 메커니즘을 나타낸다. 초기에, 단계 810에서, 위에서 상세히 설명된 바와 같이 가상 결합이 발생할 수 있다. 후속적으로, 단계 815에서, 단계 810의 가상 결합이 큰 페이지 크기의 메모리 영역 내의 메모리를 참조하는지 여부에 대한 결정이 이루어질 수 있으며, 여기서 큰 페이지 크기 영역 내의 다른 메모리는 가상 결합에 의해 이전에 참조되지 않았다. 특히, 호스트 물리적 메모리는 큰 페이지 크기의 세그먼트로 기술될 수 있다. 하나의 큰 페이지 크기 영역 내의 메모리를 참조하는 제1 가상 결합은 단계 815에서 긍정적인 결정을 초래할 수 있으며, 이에 의해 프로세싱은 단계 835로 진행할 수 있다.

[0090] 단계 835에서, 단일 상위-레벨 테이블 엔트리가 SLAT에서 생성될 수 있으며, 이에 대응하여, 단계 840에서, 복수의 하위-레벨 테이블 엔트리가 호스트 페이지 테이블에서 활용될 수 있으며, 여기서 복수의 하위-레벨 테이블 엔트리의 관련 양은 SLAT에서 생성된 단일 상위-레벨 테이블 엔트리와 동일한 메모리 범위를 포함하는 것이며, 즉, 단계 810에서 액세스가 가상 결합을 트리거링한 메모리 위치를 포함하는 메모리의 큰 페이지 크기를 기술하는 메모리 영역에 대응하는 메모리 범위이다. 이전에 나타난 바와 같이, 호스트 페이지 테이블은 단일 상위-레벨 테이블 엔트리와 반대로 복수의 하위-레벨 테이블 엔트리를 포함하기 때문에, 호스트 컴퓨팅 디바이스의 메모리 관리는 호스트 페이지 테이블의 이러한 하위-레벨 테이블 엔트리에 의해 식별되는 디스크, 하나 이상의 메모리 영역 또는 페이지로 해당 콘텐츠를 복제하는 것과 같이 페이지징 아웃하도록 선택할 수 있다. 이러한 페이지징 아웃이 발생하면, 단계 845에서 결정된 바와 같이, 프로세싱은 단계 850으로 진행할 수 있고, SLAT의 상위-레벨 테이블 엔트리는 페이지징 아웃되지 않은 호스트 페이지 테이블의 테이블의 하위-레벨 테이블 엔트리에 의해 참조되는 메모리 영역에 대응하는 복수의 하위-레벨 테이블 엔트리로 대체될 수 있다(실제로 페이지징 아웃된 호스트 페이지 테이블의 하위-레벨 테이블 엔트리에 의해 참조된 메모리 영역에 대응하는 해당 레벨 테이블 엔트리를 SLAT로부터 남겨둠).

[0091] 일 양태에 따르면, 큰 페이지 크기의 메모리 영역으로부터의 작은 페이지가 단계 845에서 페이지징 아웃되고, SLAT의 상위-레벨 테이블 엔트리가 무효화되고 대응하는 하위-레벨 테이블 엔트리의 시퀀스로 대체되는 경우(페이지징 아웃된 작은 페이지를 참조하는 임의의 하위-레벨 테이블 엔트리를 예외로 함), 선택적으로, 프로세싱은 단계 855로 진행하여 해제되는 이용 가능한 큰 페이지 크기의 메모리 영역을 계속 확인할 수 있다. 위에서 상세히 설명한 바와 같이, 예를 들어, 호스트 컴퓨팅 디바이스 상에서 실행되는 메모리 관리 프로세스는 이용 가능한 메모리의 큰 페이지 크기 영역을 기회적으로 재구성할 수 있다. 이러한 큰 페이지 크기의 메모리 영역이 단계 855에서 결정된 바와 같이 이용 가능해지면, 프로세싱은 단계 860으로 진행할 수 있으며, 여기서 SLAT의 하위-레벨 테이블 엔트리는 SLAT의 단일 상위-레벨 테이블 엔트리로 대체될 수 있으며, 적절한 경우, 이전 하위-레벨 테이블 엔트리에 의해 참조된 메모리 영역으로부터 단일 상위-레벨 테이블 엔트리에 의해 이제 참조된 메모리 영역으로 데이터가 복제할 수 있다(두 메모리 영역이 다른 경우).

[0092] 단계 815로 돌아가서, 단계 810의 가상 결합이 이전에 기술된 큰 페이지 크기 영역 내에서 메모리를 참조하는 제1 결합이 아니라면, 프로세싱은 단계 820으로 진행할 수 있다. 단계 820에서, 위에서 상세히 설명한 방식과 같이, 인터셉트 시간에 큰 페이지가 생성되어야 하는지 여부에 대해 결정이 이루어질 수 있다. 보다 구체적으로, 단계 835 및 840이 이전에 큰 페이지 크기의 메모리 영역에 대해 수행되었고 SLAT는 이제 단일 상위-레벨 테이블 엔트리를 포함하는 한편, 페이지 테이블은 SLAT의 단일 상위-레벨 테이블 엔트리에 의해 참조된 메모리 영역의 하위-부분을 참조하는 동등한 하위-레벨 테이블 엔트리를 포함하는 경우, SLAT의 단일 상위-레벨 테이블

엔트리에 의해 참조된 메모리 영역 내의 임의의 메모리 위치에 대한 후속 메모리 액세스는 위에 상세히 설명된 바와 같이, 가상 결함을 트리거링하지 않을 것이다. 따라서, 가상 결함이 단계 810에서 트리거링되고, 단계 815에서 가상 결함이 이전에 기술된 큰 페이지 크기의 메모리 영역 내의 메모리 위치로 처음 지시되지 않는 것으로 결정되면, 단계 835 및 840이 큰 페이지 크기 영역 내의 메모리 위치에 대해 이전 수행되었기 때문에 단계 810에서 트리거링된 가상 결함이 트리거링될 가능성이 있었지만, 후속하여 큰 페이지 크기의 메모리영역을 포함하는 하나 이상의 작은 페이지가 단계 845에서 결정된 바와 같이 페이징 아웃되었고, 후속하여 단계 850이 이전에 수행되었다. 따라서, 단계 820에서, 큰 페이지 크기의 메모리 영역이 인터셉트 시간에 구성되어야 하는지 여부에 대한 결정이 이루어질 수 있다. 이러한 큰 페이지 크기 영역이 구성되어야 하는 경우, 프로세싱은 단계 830으로 진행할 수 있고, 위에서 상세히 설명된 바와 같이, 적절한 양의 연속적인 작은 페이지가 어셈블링될 수 있다. 그 후 프로세싱은 이전과 같이 단계 835로 진행될 수 있다. 반대로, 단계 830에서 이용 가능한 메모리의 큰 페이지 크기 영역의 온 디맨드 구성이 요청되지 않은 경우, 도 4 및 도 5를 참조하여 위에 상세히 설명된 메커니즘에 따라 호스트 페이지 테이블 및 SLAT의 하위-레벨 테이블 엔트리를 활용하도록 프로세싱이 진행될 수 있다.

[0093] 위의 설명은 제1 예로서 컴퓨터 메모리의 액세스 속도를 증가시키는 방법을 포함하며, 본 방법은: 가상 머신 컴퓨팅 환경에서 실행되는 제1 프로세스로부터 제1 메모리 범위로 향하는 제1 메모리 액세스를 검출하는 단계; 제1 메모리 액세스를 완료하기 위한 전제 조건으로서, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리를 생성하는 단계로서, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블은 호스트 물리적 메모리 주소들을 게스트 물리적 메모리 주소들과 상관시키며, 제1 엔트리는 제1 엔트리에 의해 식별되는 제2 메모리 범위가 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있고, 제2 메모리 범위는 제1 메모리 범위보다 큰, 생성하는 단계; 및 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 제1 엔트리를 생성하는 것에 응답하여, 계층적으로 배열된 페이지 테이블에서 제1 복수의 엔트리를 사용된 것으로 마킹하는 단계로서, 계층적으로 배열된 페이지 테이블은 호스트 물리적 메모리 주소들을 호스트 가상 메모리 주소들과 상관시키고, 제1 복수의 엔트리는 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리와 동일한 제2 메모리 범위를 총괄하여 참조하고, 제1 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있는, 마킹하는 단계를 포함하고; 게스트 물리적 메모리 주소들은 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스들에 의해 물리적 메모리에 대한 주소들인 것으로 인식되고; 호스트 물리적 메모리 주소들은 가상 머신 컴퓨팅 환경을 호스팅하는 호스트 컴퓨팅 디바이스의 실제 물리적 메모리에 대한 주소들이고; 호스트 가상 메모리 주소들은 호스트 컴퓨팅 디바이스 상에서 실행되는 메모리 관리자에 의해 제공되는 가상 메모리에 대한 주소들이고, 메모리 관리자는 가상 메모리를 제공하기 위해 계층적으로 배열된 페이지 테이블을 활용하고; 게스트 물리적 메모리 주소들에 의해 식별되는 게스트 물리적 메모리는 호스트 가상 메모리 주소들의 일부에 의해 식별되는 호스트 가상 메모리의 일부에 의해 지원된다.

[0094] 제2 예는 제1 예의 방법이며, 가상 머신 컴퓨팅 환경에서 실행되는 제1 프로세스로부터 제2 메모리 액세스를 검출하는 단계로서, 제2 메모리 액세스는 제1 메모리 범위와 다른 제3 메모리 범위로 향하는, 검출하는 단계; 및 계층적으로 배열된 페이지 테이블의 제1 복수의 엔트리의 제2 서브세트를 참조하여 제2 메모리 액세스를 충족시키는 단계를 더 포함하고; 제1 메모리 액세스는 계층적으로 배열된 페이지 테이블의 제1 복수의 엔트리의 제1 서브세트를 참조하여 충족되었고, 제1 서브세트는 계층적으로 배열된 페이지 테이블에서 제2 서브세트와 다른 엔트리들을 포함한다.

[0095] 제3 예는 제1 예의 방법이며: 제1 메모리 액세스가 완료된 후, 계층적으로 배열된 페이지 테이블의 제1 복수 엔트리의 제1 서브세트가 원래 대응하는 호스트 물리적 메모리 주소들에 저장되었고 후속하여 비휘발성 저장 매체로 페이징 아웃(paging out)된 데이터를 가졌음을 검출하는 단계; 검출에 응답하여, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리를 무효화하는 단계; 및 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리 대신에, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 복수의 엔트리를 생성하는 단계로서, 제2 복수의 엔트리는 제1 엔트리와 동일한 제2 메모리 범위의 적어도 일부를 총괄하여 참조하고, 제2 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있다.

[0096] 제4 예는 제3 예의 방법이며, 제2 복수의 엔트리는 제1 프로세스에 의해 이전에 액세스되었고 페이징 아웃되지 않았던 제2 메모리 범위의 부분들을 참조한다.

[0097] 제5 예는 제1 예의 방법이며: 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역들을 호스트

물리적 메모리의 단일의 큰 페이지 크기 영역으로 어셈블링하는 단계를 더 포함한다.

- [0098] 제6 예는 제5 예의 방법이며, 제1 메모리 액세스를 검출한 후, 그리고 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 제1 엔트리를 생성하기 전에 어셈블링이 발생된다.
- [0099] 제7 예는 제5 예의 방법이며: 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 엔트리를 생성하는 단계로서, 제2 엔트리는 제2 엔트리에 의해 식별되는 제3 메모리 범위가 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있고, 제3 메모리 범위는 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역이 어셈블링된 호스트 물리적 메모리의 단일의 큰 페이지 크기 영역을 참조하는, 생성하는 단계; 및 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 엔트리를 생성하는 것에 응답하여, 계층적으로 배열된 페이지 테이블의 제2 복수의 엔트리를 사용된 것으로 마킹하는 단계로서, 제2 복수의 엔트리는 호스트 물리적 메모리의 단일의 큰 페이지 크기 영역으로 어셈블링된 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역을 참조하는, 마킹하는 단계를 더 포함한다.
- [0100] 제8 예는 제7 예의 방법이며: 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 엔트리를 생성하는 단계로서, 제2 엔트리는 제2 엔트리에 의해 식별되는 제3 메모리 범위가 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있고, 제3 메모리 범위는 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역이 어셈블링된 호스트 물리적 메모리의 단일의 큰 페이지 크기 영역을 참조하는, 생성하는 단계; 및 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 엔트리를 생성하는 것에 응답하여, 계층적으로 배열된 페이지 테이블의 제2 복수의 엔트리를 사용된 것으로 마킹하는 단계로서, 제2 복수의 엔트리는 호스트 물리적 메모리의 단일의 큰 페이지 크기 영역으로 어셈블링된 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역을 참조하는, 마킹하는 단계를 더 포함한다.
- [0101] 제9 예는 제7 예의 방법이며: 호스트 물리적 메모리의 하나 이상의 작은 페이지 크기 영역의 제2 세트로부터 제1 복수의 연속적인 작은 페이지 크기 영역의 적어도 일부로 데이터를 복제하는 단계; 및 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 복수의 엔트리를 무효화하는 단계로서, 제2 복수의 엔트리는: (1) 호스트 물리적 메모리의 하나 이상의 작은 페이지 크기 영역의 제2 세트를 참조하는 엔트리들의 제1 서브세트 및 (2) 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역의 적어도 일부를 참조하는 엔트리들의 제2 서브세트를 모두 포함하고, 제2 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있는, 무효화하는 단계를 더 포함하고, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 생성된 제2 엔트리는 무효화된 제2 복수의 엔트리 대신 활용된다.
- [0102] 제10 예는 제5 예의 방법이며, 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역을 어셈블링하는 단계는 제1 복수의 연속적인 작은 페이지 크기 영역의 일부로부터 제1 복수의 연속적인 작은 페이지 크기 영역과 다른 호스트 물리적 메모리의 다른 작은 페이지 크기 영역들로 데이터를 복제하는 단계를 포함한다.
- [0103] 제11 예는 제10 예의 방법이며, 제1 복수의 연속적인 작은 페이지 크기 영역 중 일부로부터 다른 작은 페이지 크기 영역들로 데이터를 복제하는 단계는 제1 복수의 연속적인 작은 페이지 크기 영역의 조각화가 조각화 임계값 미만인 경우에만 수행된다.
- [0104] 제12 예는 제1 예의 방법이며, 제2 메모리 범위의 페이징(paging)을 방지하는 단계를 더 포함한다.
- [0105] 제13 예는 제12 예의 방법이며, 제2 메모리 범위의 페이징을 방지하는 단계는 제2 메모리 범위의 하나 이상의 부분이 액세스 빈도 임계값보다 더욱 빈번하게 액세스되는 경우에만 수행된다.
- [0106] 제14 예는 제12 예의 방법이며, 제2 메모리 범위의 하나 이상의 부분이 액세스 빈도 임계값보다 덜 빈번하게 액세스되는 경우, 제2 메모리 범위의 페이징의 방지를 제거하는 단계를 더 포함한다.
- [0107] 제15 예는 제1 예의 방법이며, 제2 메모리 범위는 2 MB의 크기이다.
- [0108] 제16 예는 컴퓨팅 디바이스이며, 컴퓨팅 디바이스는: 하나 이상의 중앙 처리 장치; 랜덤 액세스 메모리(RAM); 및 하나 이상의 컴퓨터-판독 가능 매체를 포함하고, 하나 이상의 컴퓨터-판독 가능 매체는: 컴퓨팅 디바이스에 의해 실행될 때, 컴퓨팅 디바이스로 하여금 호스트 가상 메모리 주소들을 RAM 상에서 위치들을 식별하는 호스트 물리적 메모리 주소들로 트랜슬레이팅하기 위하여 계층적으로 배열된 페이지 테이블을 참조하는 메모리 관리자를 제공하게 하는 컴퓨터-실행 가능 명령어들의 제1 세트; 컴퓨팅 디바이스에 의해 실행될 때, 컴퓨팅 디바이스

로 하여금 가상 머신 컴퓨팅 환경을 제공하게 하는 컴퓨터-실행 가능 명령어들의 제2 세트로서, 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스들은 게스트 물리적 메모리 주소들을 물리적 메모리에 대한 주소들로 인식하고, 게스트 물리적 메모리 주소들에 의해 식별되는 추가 게스트 물리적 메모리는 호스트 가상 메모리 주소들의 일부에 의해 식별되는 호스트 가상 메모리의 일부에 의해 지원되는, 컴퓨터-실행 가능 명령어들의 제2 세트; 및 컴퓨팅 디바이스에 의해 실행될 때, 컴퓨팅 디바이스로 하여금: 가상 머신 컴퓨팅 환경에서 실행되는 제1 프로세스로부터 제1 메모리 범위로 향하는 제1 메모리 액세스를 검출하게 하고; 제1 메모리 액세스를 완료하기 위한 전제 조건으로서, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리를 생성하게 하고, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블은 호스트 물리적 메모리 주소들을 게스트 물리적 메모리 주소들과 상관시키며, 제1 엔트리는 제1 엔트리에 의해 식별되는 제2 메모리 범위가 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있고, 제2 메모리 범위는 제1 메모리 범위보다 크고; 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 제1 엔트리를 생성하는 것에 응답하여, 계층적으로 배열된 페이지 테이블에서 제1 복수의 엔트리를 사용된 것으로 마킹하게 하고, 제1 복수의 엔트리는 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리와 동일한 제2 메모리 범위를 총괄하여 참조하고, 제1 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있는, 컴퓨터-실행 가능 명령어들의 제3 세트를 포함한다.

[0109] 제17 예는 제16 예의 컴퓨팅 디바이스이며, 컴퓨터-실행 가능 명령어들의 제3 세트는 컴퓨팅 디바이스에 의해 실행될 때 컴퓨팅 디바이스로 하여금, 제1 메모리 액세스가 완료된 후, 계층적으로 배열된 페이지 테이블의 제1 복수 엔트리의 제1 서브세트가 원래 대응하는 호스트 물리적 메모리 주소들에 저장되었고 후속하여 비휘발성 저장 매체로 페이징 아웃된 데이터를 가졌음을 검출하게 하고; 검출에 응답하여, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리를 무효화하게 하고; 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리 대신에, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제2 복수의 엔트리를 생성하게 하는 컴퓨터-실행 가능 명령어들을 더 포함하고, 제2 복수의 엔트리는 제1 엔트리와 동일한 제2 메모리 범위의 적어도 일부를 총괄하여 참조하고, 제2 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있다.

[0110] 제18 예는 제16 예의 컴퓨팅 디바이스이며, 컴퓨터-실행 가능 명령어들의 제3 세트는 컴퓨팅 디바이스에 의해 실행될 때 컴퓨팅 디바이스로 하여금, 호스트 물리적 메모리의 제1 복수의 연속적인 작은 페이지 크기 영역들을 호스트 물리적 메모리의 단일의 큰 페이지 크기 영역으로 어셈블링하게 하는 컴퓨터-실행 가능 명령어들을 더 포함하고; 제1 메모리 액세스를 검출한 후, 그리고 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 제1 엔트리를 생성하기 전에 어셈블링이 발생된다.

[0111] 제19 예는 제16 예의 컴퓨팅 디바이스이며, 컴퓨터-실행 가능 명령어들의 제3 세트는 컴퓨팅 디바이스에 의해 실행될 때 컴퓨팅 디바이스로 하여금 제2 메모리 범위의 페이징을 방지하게 하는 컴퓨터-실행 가능 명령어들을 더 포함한다.

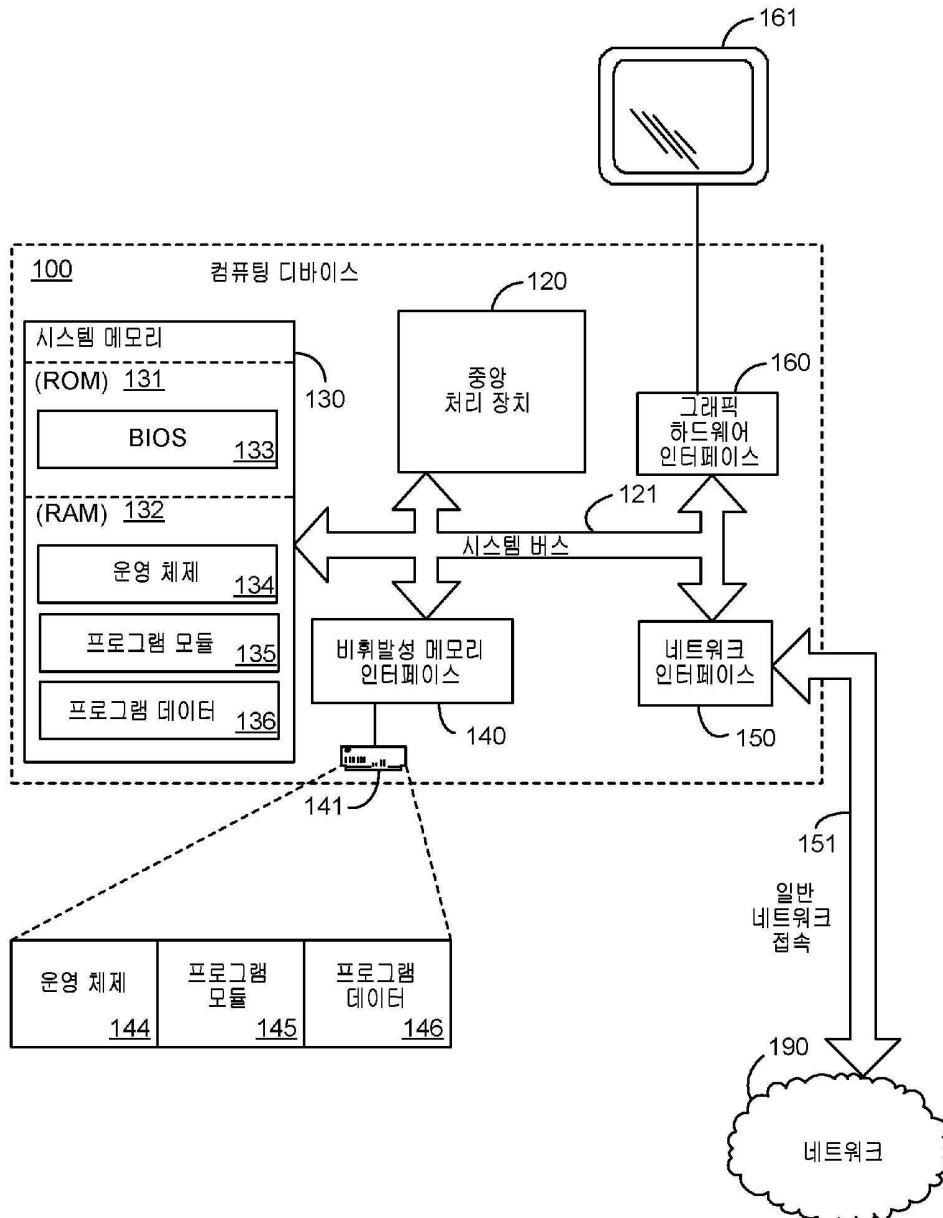
[0112] 제20 예는 컴퓨터-실행 가능 명령어들을 포함하는 하나 이상의 컴퓨터-판독 가능 저장 매체이며, 컴퓨터-실행 가능 명령어들은 실행될 때, 컴퓨팅 디바이스로 하여금, 가상 머신 컴퓨팅 환경에서 실행되는 제1 프로세스로부터 제1 메모리 범위로 향하는 제1 메모리 액세스를 검출하게 하고; 제1 메모리 액세스를 완료하기 위한 전제 조건으로서, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리를 생성하게 하고, 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블은 호스트 물리적 메모리 주소들을 게스트 물리적 메모리 주소들과 상관시키며, 제1 엔트리는 제1 엔트리에 의해 식별되는 제2 메모리 범위가 계층적으로 최저 레벨의 테이블들의 어떠한 테이블도 참조하지 않고 식별될 수 있도록 계층적으로 최저 레벨의 테이블들 위의 적어도 하나의 계층적 레벨에 있고, 제2 메모리 범위는 제1 메모리 범위보다 크고; 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블에서 제1 엔트리를 생성하는 것에 응답하여, 계층적으로 배열된 페이지 테이블에서 제1 복수의 엔트리를 사용된 것으로 마킹하게 하고, 계층적으로 배열된 페이지 테이블은 호스트 물리적 메모리 주소들을 호스트 가상 메모리 주소들과 상관시키고, 제1 복수의 엔트리는 계층적으로 배열된 제2 계층 주소 트랜슬레이션 테이블의 제1 엔트리와 동일한 제2 메모리 범위를 총괄하여 참조하고, 제1 복수의 엔트리의 엔트리들은 계층적으로 최저 레벨의 테이블들에 있고; 게스트 물리적 메모리 주소들은 가상 머신 컴퓨팅 환경 내에서 실행되는 프로세스들에 의해 물리적 메모리에 대한 주소들인 것으로 인식되고; 호스트 물리적 메모리 주소들은 가상 머신 컴퓨팅 환경을 호스팅하는 호스트 컴퓨팅 디바이스의 실제 물리적 메모리에 대한 주소들이고; 호스트 가상 메모리 주소들은 호스트 컴퓨팅 디바이스 상에서 실행되는 메모리 관리자에 의해 제공되는 가상 메모리에 대한 주소들이고, 메모리 관리자는 가상 메모리를 제공하기 위해 계층적으로 배열된 페이지 테이블을 활용하고; 게스트 물리적 메모리

주소들에 의해 식별되는 게스트 물리적 메모리는 호스트 가상 메모리 주소들의 일부에 의해 식별되는 호스트 가상 메모리의 일부에 의해 지원된다.

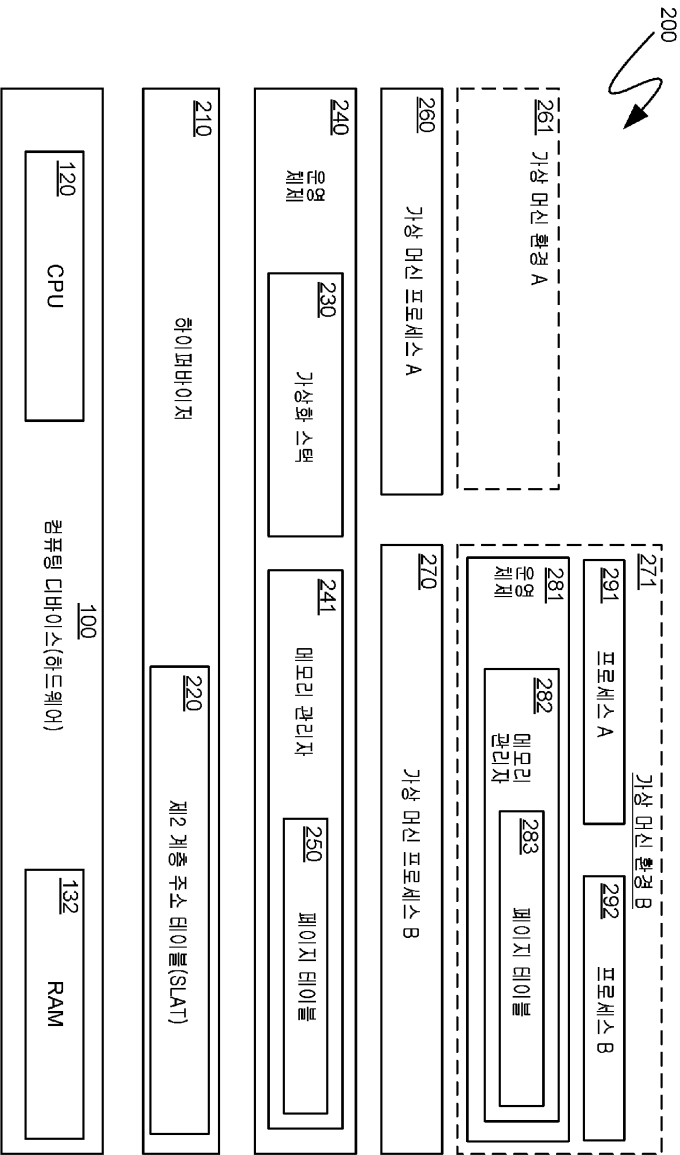
[0113] 위의 설명으로부터 알 수 있는 바와 같이, SLAT를 통한 메모리 액세스가 가속화될 수 있는 메커니즘이 설명되었다. 본원에 설명된 주제의 다수의 가능한 변형의 관점에서, 이하의 청구항 및 그 동등물의 범위 내에 있을 수 있는 이러한 모든 실시예를 본 발명으로 주장한다.

도면

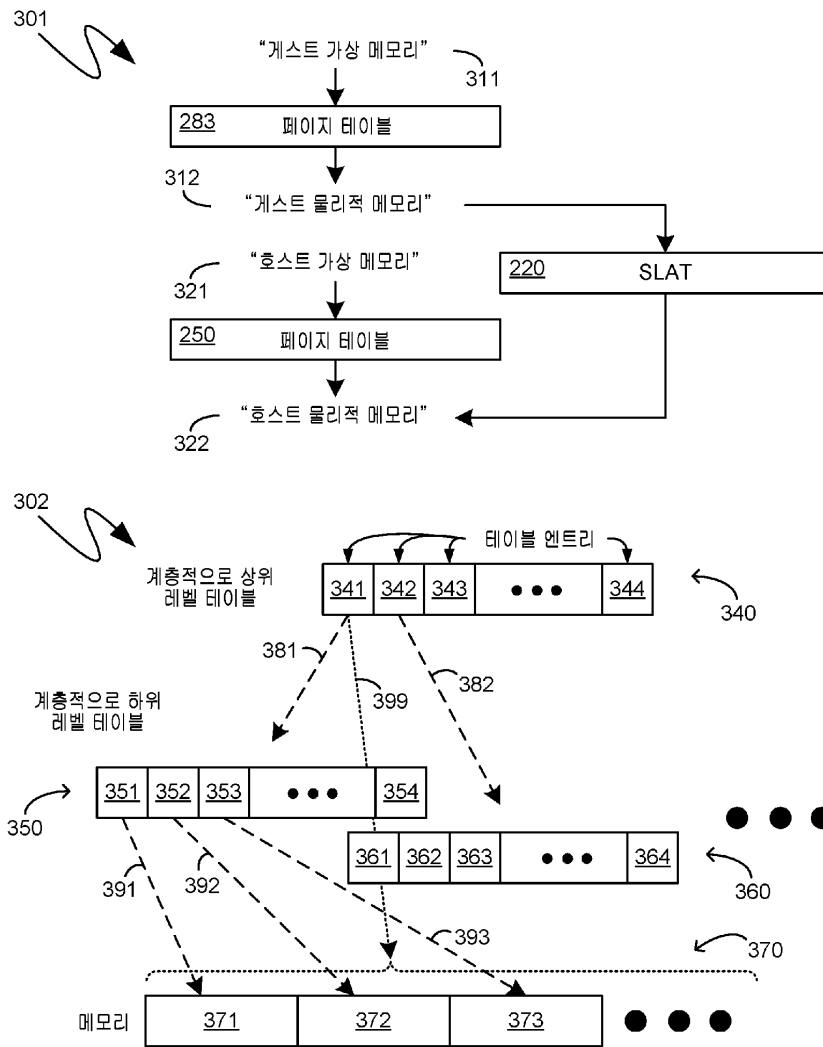
도면1



도면2

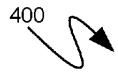
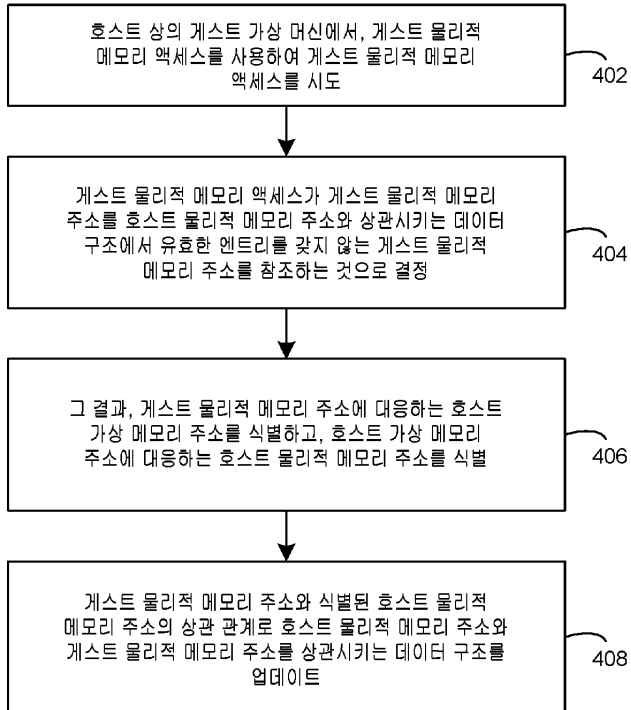


도면3

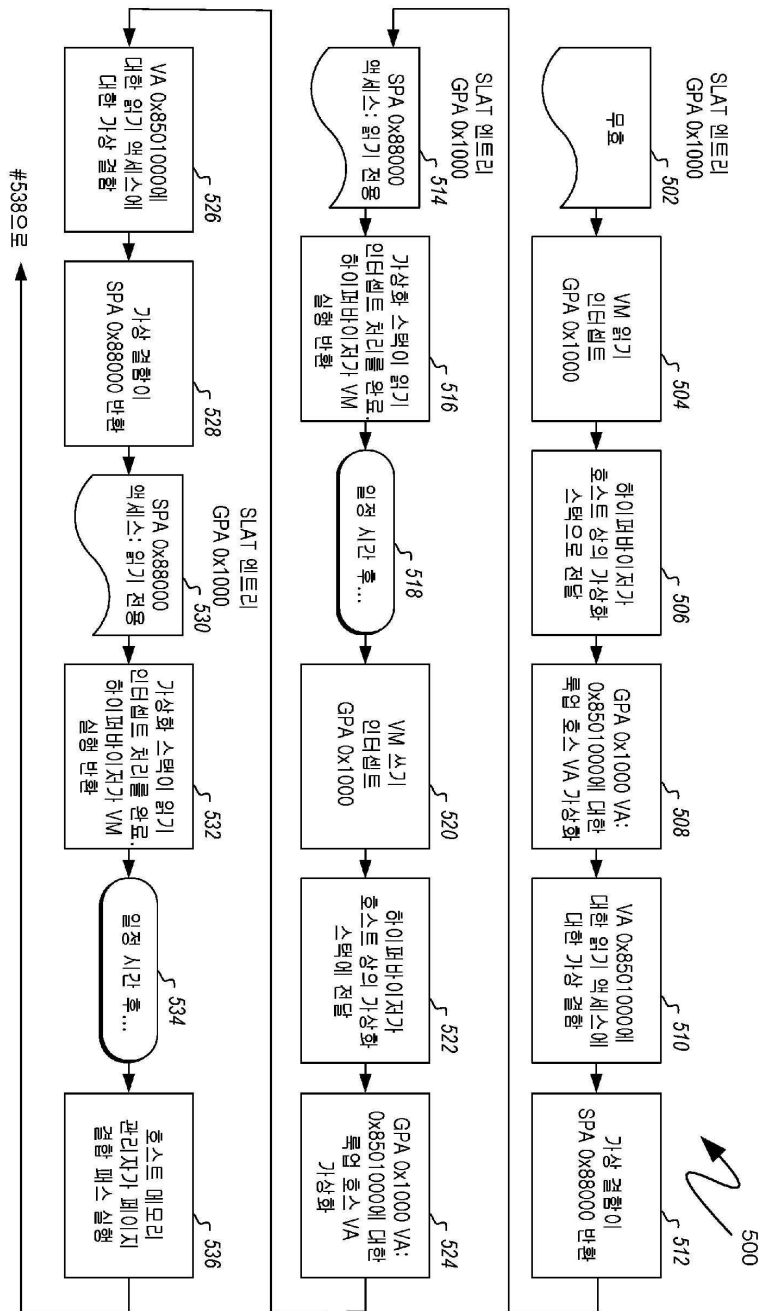


도면4

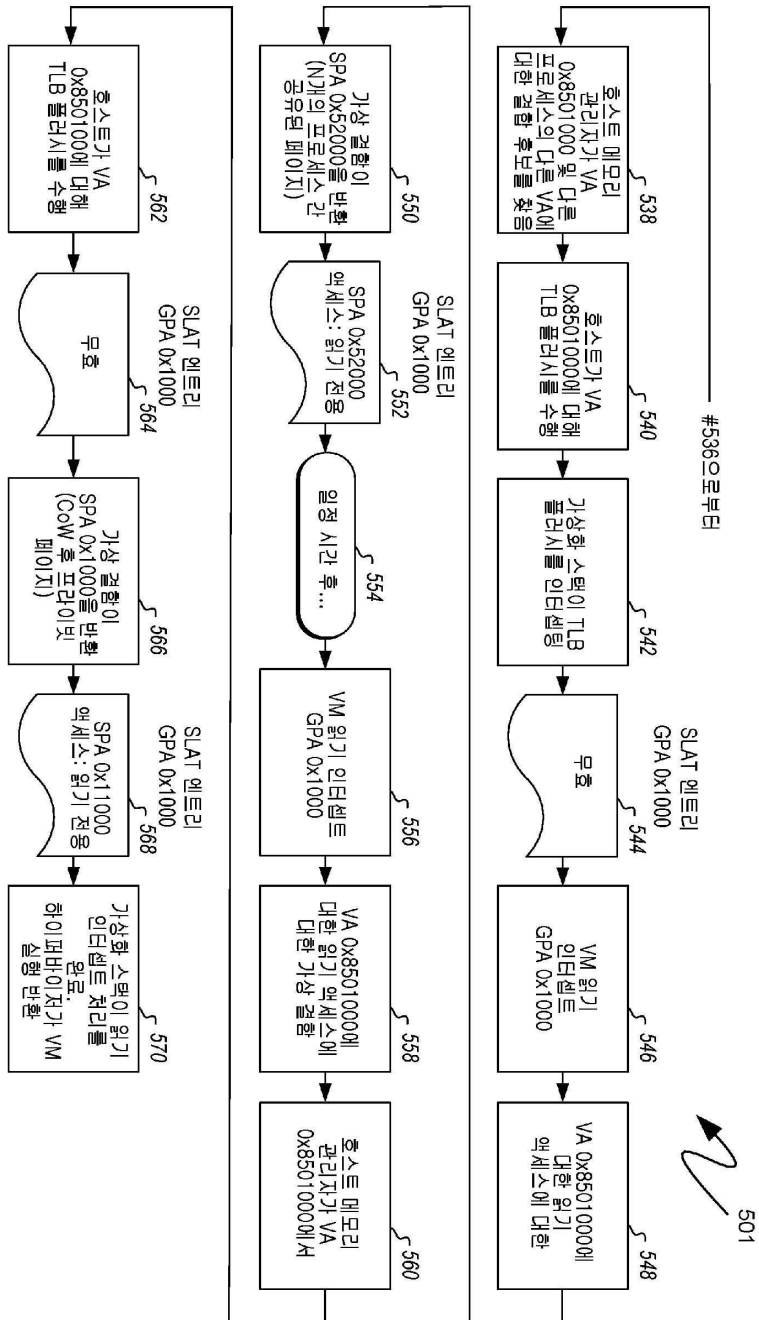
400

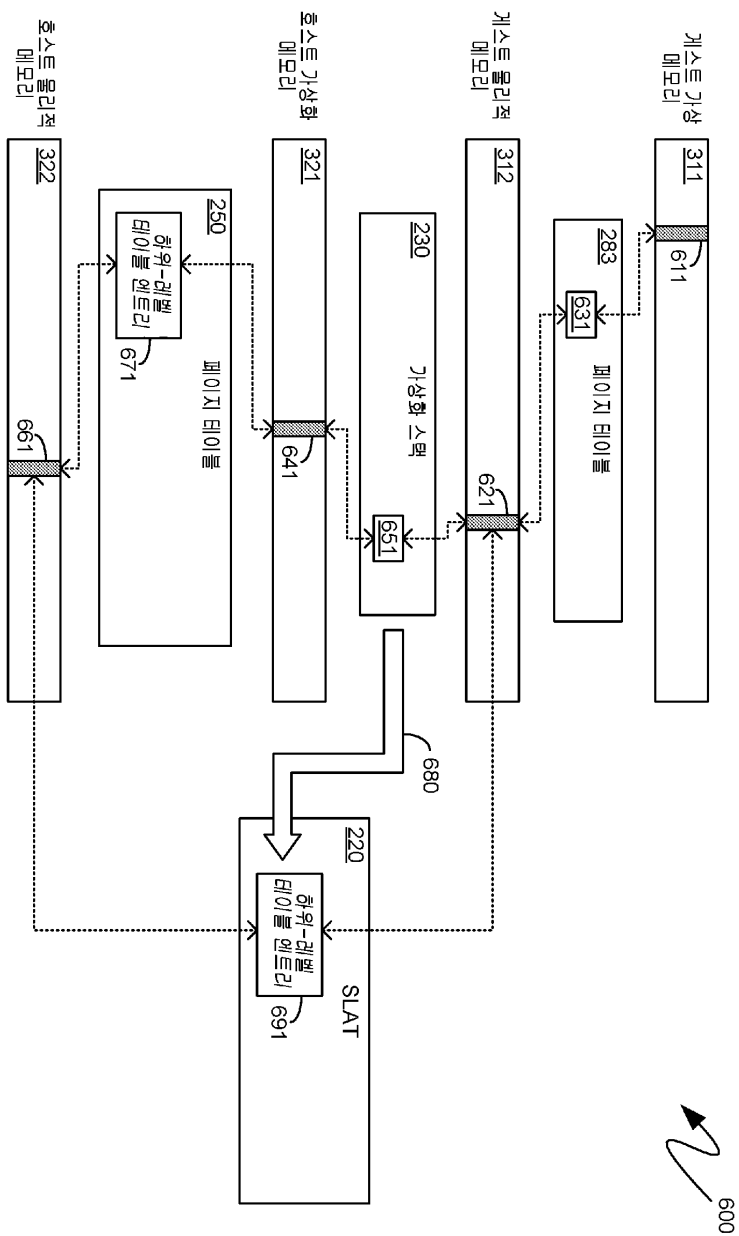
도면5a



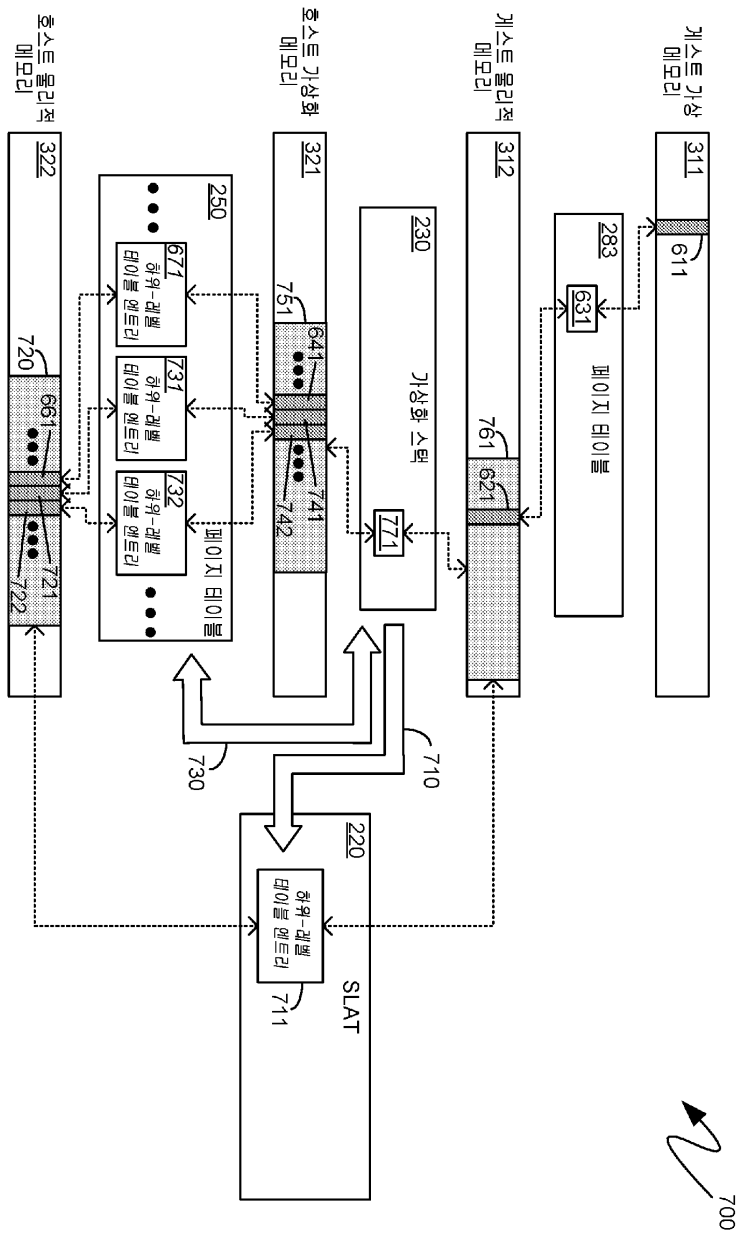
도면5b



도면6



도면7



도면8

