(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년07월06일
(11) 등록번호 10-2273622
(24) 등록일자 2021년06월30일

(51) 국제특허분류(Int. Cl.)
*G06F 12/1009* (2016.01) *G06F 12/02* (2018.01)
*G06F 12/0815* (2016.01) *G06F 12/0868* (2016.01)
*G06F 12/0871* (2016.01) *G06F 12/0873* (2016.01)
*G06F 12/1027* (2016.01) *G06F 12/12* (2016.01)
(52) CPC특허분류
*G06F 12/1009* (2013.01)
*G06F 12/023* (2013.01)
(21) 출원번호 10-2019-7011367
(22) 출원일자(국제) 2017년08월25일
심사청구일자 2019년04월19일
(85) 번역문제출일자 2019년04월19일
(65) 공개번호 10-2019-0052106
(43) 공개일자 2019년05월15일
(86) 국제출원번호 PCT/US2017/048663
(87) 국제공개번호 WO 2018/057235
국제공개일자 2018년03월29일
(30) 우선권주장
15/273,433 2016년09월22일 미국(US)
(56) 선행기술조사문헌
US20090172344 A1*
WO2011002900 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
**구글 엘엘씨**
미국 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600 (우:94043)
(72) 발명자
**코번, 조엘 딜런**
미국 94043 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600
**보처스, 알버트**
미국 94043 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600
*(뒷면에 계속)*
(74) 대리인
**특허법인 남앤남**

전체 청구항 수 : 총 15 항 심사관 : 안지현

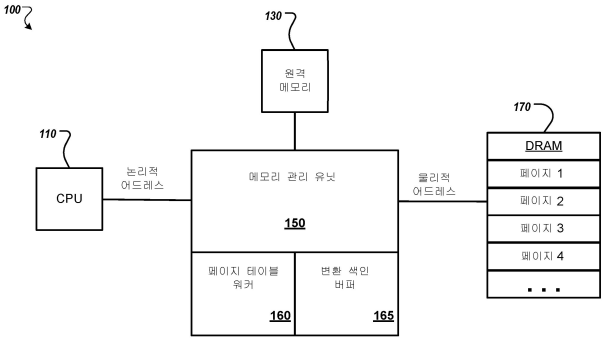
(54) 발명의 명칭 **거대한 페이지들을 지원하는 메모리 관리**

## (57) 요 약

메인 메모리로부터, 데이터의 제1 페이지의 제1 부분에 포함된 데이터에 액세스하기 위한 요청을 수신하고 – 데이터의 제1 페이지는 제1 페이지 사이즈를 가짐 –; 데이터의 제1 페이지가 메인 메모리에 저장되지 않는다고 결정하는 것에 기반하여 페이지 폴트를 개시하고; 제1 페이지 사이즈와 동등한 메인 메모리의 부분을 할당하고; 데

(뒷면에 계속)

## 대 표 도



이터의 제1 페이지 전체를 전달하지 않으면서 2차 메모리로부터 메인 메모리의 할당된 부분으로 데이터의 제1 페이지의 제1 부분을 전달하며; 그리고 데이터의 제1 페이지의 제1 부분이 전달되는 메인 메모리의 할당된 부분의 위치를 가리키도록 데이터의 제1 페이지의 제1 부분과 연관된 제1 페이지 테이블 엔트리를 업데이트하기 위한 방법들, 시스템들, 및 장치가 제공된다.

(52) CPC특허분류
***G06F 12/0815*** (2013.01)
***G06F 12/0868*** (2013.01)
***G06F 12/0871*** (2013.01)
***G06F 12/0873*** (2013.01)
***G06F 12/1027*** (2013.01)
***G06F 12/12*** (2013.01)
*G06F 2212/1024* (2013.01)
*G06F 2212/152* (2013.01)
*G06F 2212/3042* (2013.01)

(72) 발명자
**존슨, 크리스토퍼 라일**
미국 94043 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600
**스프린클, 로버트 에스.**
미국 94043 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600

# 명 세 서

## 청구범위

### 청구항 1

메인 메모리로부터, 데이터의 제1 페이지의 제1 부분에 포함된 데이터에 액세스하기 위한 요청을 수신하는 단계 - 상기 데이터의 제1 페이지는 제1 페이지 사이즈를 갖고, 상기 제1 부분은 상기 제1 페이지 사이즈보다 작은 제2 페이지 사이즈를 포함함 -;

상기 데이터의 제1 페이지가 상기 메인 메모리에 저장되어 있지 않고 2차 메모리에 저장되어 있다고 결정하는 것에 기반하여 페이지 폴트(page fault)를 개시하는 단계;

상기 페이지 폴트를 개시하는 것에 대한 응답으로, 상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분을 할당하는 단계;

상기 데이터의 제1 페이지 전체를 전달하지 않고 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 상기 데이터의 제1 페이지의 제1 부분을 전달하는 단계 - 상기 데이터의 제1 페이지의 나머지 양은 상기 2차 메모리에 저장된 채로 유지됨 -;

상기 데이터의 제1 페이지의 제1 부분이 전달된 상기 메인 메모리의 할당된 부분의 위치를 가리키도록 상기 데이터의 제1 페이지의 제1 부분과 연관된 제1 페이지 테이블 엔트리를 업데이트하는 단계; 및

상기 2차 메모리로부터 상기 메인 메모리로 상기 데이터의 제1 페이지의 나머지 양을 전달하는 단계를 포함하고,

상기 데이터의 제1 페이지의 나머지 양을 전달하는 단계는,

상기 데이터의 제1 페이지 전체가 상기 메인 메모리에 저장될 때까지, 상기 2차 메모리로부터 상기 데이터의 제1 페이지 중 상기 제2 페이지 사이즈에 대응하는 개개의 부분들을 상기 메인 메모리의 할당된 부분으로 반복적으로 전달하는 단계; 및

상기 메인 메모리 내의 상기 데이터의 제1 페이지의 상기 개개의 부분들의 각자의 위치들을 가리키도록 상기 데이터의 제1 페이지의 상기 개개의 부분들 각각에 대한 각자의 페이지 테이블 엔트리를 업데이트하는 단계를 포함하는,

컴퓨터로 구현되는 방법.

### 청구항 2

삭제

### 청구항 3

삭제

### 청구항 4

제1항에 있어서,

일단 상기 데이터의 제1 페이지 전체가 상기 메인 메모리에 저장되면, 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 전달된 상기 데이터의 제1 페이지의 개개의 부분들로부터 상기 데이터의 제1 페이지를 리어셈블리하는 단계; 및

상기 메인 메모리 내의 상기 데이터의 리어셈블리된 제1 페이지의 위치를 가리키도록, 상기 데이터의 제1 페이지와 연관된 페이지 테이블 엔트리를 업데이트하는 단계를 더 포함하는,

컴퓨터로 구현되는 방법.

### 청구항 5

제1항에 있어서,

상기 2차 메모리로부터 상기 메인 메모리로 상기 데이터의 제1 페이지의 나머지 부분을 전달하기 전에, 액세스되도록 요청되었던 상기 데이터의 제1 페이지의 제1 부분이 상기 메인 메모리로 전달되었다는 것을 표시하는 단계를 더 포함하는,

컴퓨터로 구현되는 방법.

**청구항 6**

제1항, 제4항 및 제5항 중 어느 한 항에 있어서,

페이지 테이블 스캐너를 이용한 페이지 테이블의 스캔에 기반하여, 액세스 비트가 상기 페이지 테이블의 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정하는 단계 - 상기 액세스 비트는 상기 페이지 테이블 엔트리와 연관된 페이지가 마지막 스캔 기간에서 액세스되었는지 여부를 표시하고, 상기 제1 페이지 사이즈를 갖는 적어도 하나의 페이지는 상기 제2 페이지 사이즈의 페이지들로 분할되며, 상기 페이지 테이블 내의 상기 제2 페이지 사이즈의 페이지들 각각에 대한 페이지 테이블 엔트리가 스캐닝됨 -;

상기 액세스 비트가 상기 페이지와 연관된 페이지 테이블 엔트리에 대해 세팅되지 않는다고 결정하는 것에 대한 응답으로 각각의 페이지에 대한 카운트를 증분시키는 단계;

상기 액세스 비트가 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정한 이후, 상기 액세스 비트를 리세팅하는 단계; 및

분할되었던 상기 제1 페이지 사이즈를 갖는 페이지로 상기 제2 페이지 사이즈의 페이지들을 리어셈블리하는 단계를 더 포함하는,

컴퓨터로 구현되는 방법.

**청구항 7**

제6항에 있어서,

상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분이 할당될 수 없다면, 상기 각각의 페이지에 대한 카운트에 기반하여 상기 제1 페이지 사이즈를 갖는 가장 적게 사용된 페이지를 결정하고, 상기 가장 적게 사용된 페이지를 상기 2차 메모리로 릴리즈(release)하며, 그리고 상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분을 상기 릴리즈된 가장 적게 사용된 페이지의 위치에 할당하는 단계; 및

상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분이 할당될 수 있다면, 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 상기 데이터의 제1 페이지의 제1 부분을 전달하는 단계를 더 포함하는,

컴퓨터로 구현되는 방법.

**청구항 8**

제1항, 제4항 및 제5항 중 어느 한 항에 있어서,

상기 제1 페이지 사이즈를 갖는 상기 데이터의 제1 페이지의 메모리 구조를 상기 제1 페이지 사이즈보다 작은 상기 제2 페이지 사이즈를 갖는 데이터의 복수의 페이지들로 변경시키는 단계를 더 포함하는,

컴퓨터로 구현되는 방법.

**청구항 9**

하나 이상의 프로세서들; 및

메인 메모리 및 2차 메모리를 포함하는 메모리를 포함하며,

상기 메모리는, 실행될 경우 상기 하나 이상의 프로세서들로 하여금 동작들을 수행하게 하도록 동작가능한 명령들을 저장하고,

상기 동작들은,

상기 메인 메모리로부터, 데이터의 제1 페이지의 제1 부분에 포함된 데이터에 액세스하기 위한 요청을 수신하는 동작 - 상기 데이터의 제1 페이지는 제1 페이지 사이즈를 갖고, 상기 제1 부분은 상기 제1 페이지 사이즈보다 작은 제2 페이지 사이즈를 포함함 -;

상기 데이터의 제1 페이지가 상기 메인 메모리에 저장되어 있지 않고 상기 2차 메모리에 저장되어 있다고 결정하는 것에 기반하여 페이지 폴트를 개시하는 동작;

상기 페이지 폴트를 개시하는 것에 대한 응답으로, 상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분을 할당하는 동작;

상기 데이터의 제1 페이지 전체를 전달하지 않고 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 상기 데이터의 제1 페이지의 제1 부분을 전달하는 동작 - 상기 데이터의 제1 페이지의 나머지 양은 상기 2차 메모리에 저장된 채로 유지됨 -;

상기 데이터의 제1 페이지의 제1 부분이 전달된 상기 메인 메모리의 할당된 부분의 위치를 가리키도록 상기 데이터의 제1 페이지의 제1 부분과 연관된 제1 페이지 테이블 엔트리를 업데이트하는 동작; 및

상기 2차 메모리로부터 상기 메인 메모리로 상기 데이터의 제1 페이지의 나머지 양을 전달하는 동작을 포함하고,

상기 데이터의 제1 페이지의 나머지 양을 전달하는 동작은,

상기 데이터의 제1 페이지 전체가 상기 메인 메모리에 저장될 때까지, 상기 2차 메모리로부터 상기 데이터의 제1 페이지 중 상기 제2 페이지 사이즈에 대응하는 개개의 부분들을 상기 메인 메모리의 할당된 부분으로 반복적으로 전달하는 동작; 및

상기 메인 메모리 내의 상기 데이터의 제1 페이지의 상기 개개의 부분들의 각자의 위치들을 가리키도록 상기 데이터의 제1 페이지의 상기 개개의 부분들 각각에 대한 각자의 페이지 테이블 엔트리를 업데이트하는 동작을 포함하는,

시스템.

**청구항 10**

삭제

**청구항 11**

삭제

**청구항 12**

제9항에 있어서,

상기 동작들은,

일단 상기 데이터의 제1 페이지 전체가 상기 메인 메모리에 저장되면, 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 전달된 상기 데이터의 제1 페이지의 개개의 부분들로부터 상기 데이터의 제1 페이지를 리어셈블리하는 동작; 및

상기 메인 메모리 내의 상기 데이터의 리어셈블리된 제1 페이지의 위치를 가리키도록, 상기 데이터의 제1 페이지와 연관된 페이지 테이블 엔트리를 업데이트하는 동작을 더 포함하는,

시스템.

**청구항 13**

제9항에 있어서,

상기 동작들은, 상기 2차 메모리로부터 상기 메인 메모리로 상기 데이터의 제1 페이지의 나머지 부분을 전달하기 전에, 액세스되도록 요청되었던 상기 데이터의 제1 페이지의 제1 부분이 상기 메인 메모리로 전달되었다는 것을 표시하는 동작을 더 포함하는,

시스템.

**청구항 14**

제9항, 제12항 및 제13항 중 어느 한 항에 있어서,

상기 동작들은,

페이지 테이블 스캐너를 이용한 페이지 테이블의 스캔에 기반하여, 액세스 비트가 상기 페이지 테이블의 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정하는 동작 - 상기 액세스 비트는 상기 페이지 테이블 엔트리와 연관된 페이지가 마지막 스캔 기간에서 액세스되었는지 여부를 표시하고, 상기 제1 페이지 사이즈를 갖는 적어도 하나의 페이지는 상기 제2 페이지 사이즈의 페이지들로 분할되며, 상기 페이지 테이블 내의 상기 제2 페이지 사이즈의 페이지들 각각에 대한 페이지 테이블 엔트리가 스캐닝됨 -;

상기 액세스 비트가 상기 페이지와 연관된 페이지 테이블 엔트리에 대해 세팅되지 않는다고 결정하는 것에 대한 응답으로 각각의 페이지에 대한 카운트를 증분시키는 동작;

상기 액세스 비트가 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정한 이후, 상기 액세스 비트를 리세팅하는 동작; 및

분할되었던 상기 제1 페이지 사이즈를 갖는 페이지로 상기 제2 페이지 사이즈의 페이지들을 리어셈블리하는 동작을 더 포함하는,

시스템.

**청구항 15**

제14항에 있어서,

상기 동작들은,

상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분이 할당될 수 없다면, 상기 각각의 페이지에 대한 카운트에 기반하여 상기 제1 페이지 사이즈를 갖는 가장 적게 사용된 페이지를 결정하고, 상기 가장 적게 사용된 페이지를 상기 2차 메모리로 릴리즈하며, 그리고 상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분을 상기 릴리즈된 가장 적게 사용된 페이지의 위치에 할당하는 동작; 및

상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분이 할당될 수 있다면, 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 상기 데이터의 제1 페이지의 제1 부분을 전달하는 동작을 더 포함하는,

시스템.

**청구항 16**

제9항, 제12항 및 제13항 중 어느 한 항에 있어서,

상기 동작들은, 상기 제1 페이지 사이즈를 갖는 상기 데이터의 제1 페이지의 메모리 구조를 상기 제1 페이지 사이즈보다 작은 상기 제2 페이지 사이즈를 갖는 데이터의 복수의 페이지들로 변경시키는 동작을 더 포함하는,

시스템.

**청구항 17**

하나 이상의 프로세서들에 의해 실행가능한 명령들을 저장하는 컴퓨터-판독가능 저장 디바이스로서,

상기 명령들은, 실행 시에, 상기 하나 이상의 프로세서들로 하여금 동작들을 수행하게 하며,

상기 동작들은,

메인 메모리로부터, 데이터의 제1 페이지의 제1 부분에 포함된 데이터에 액세스하기 위한 요청을 수신하는 동작 - 상기 데이터의 제1 페이지는 제1 페이지 사이즈를 갖고, 상기 제1 부분은 상기 제1 페이지 사이즈보다 작은 제2 페이지 사이즈를 포함함 -;

상기 데이터의 제1 페이지가 상기 메인 메모리에 저장되어 있지 않고 2차 메모리에 저장되어 있다고 결정하는

것에 기반하여 페이지 폴트를 개시하는 동작;

상기 페이지 폴트를 개시하는 것에 대한 응답으로, 상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분을 할당하는 동작;

상기 데이터의 제1 페이지 전체를 전달하지 않고 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 상기 데이터의 제1 페이지의 제1 부분을 전달하는 동작 - 상기 데이터의 제1 페이지의 나머지 양은 상기 2차 메모리에 저장된 채로 유지됨 -; 및

상기 데이터의 제1 페이지의 제1 부분이 전달된 상기 메인 메모리의 할당된 부분의 위치를 가리키도록 상기 데이터의 제1 페이지의 제1 부분과 연관된 제1 페이지 테이블 엔트리를 업데이트하는 동작; 및

상기 2차 메모리로부터 상기 메인 메모리로 상기 데이터의 제1 페이지의 나머지 양을 전달하는 동작을 포함하고,

상기 데이터의 제1 페이지의 나머지 양을 전달하는 동작은,

상기 데이터의 제1 페이지 전체가 상기 메인 메모리에 저장될 때까지, 상기 2차 메모리로부터 상기 데이터의 제1 페이지 중 상기 제2 페이지 사이즈에 대응하는 개개의 부분들을 상기 메인 메모리의 할당된 부분으로 반복적으로 전달하는 동작;

상기 메인 메모리 내의 상기 데이터의 제1 페이지의 상기 개개의 부분들의 각자의 위치들을 가리키도록 상기 데이터의 제1 페이지의 상기 개개의 부분들 각각에 대한 각자의 페이지 테이블 엔트리를 업데이트하는 동작을 더 포함하는,

컴퓨터-판독가능 저장 디바이스.

**청구항 18**

삭제

**청구항 19**

제17항에 있어서,

상기 동작들은,

일단 상기 데이터의 제1 페이지 전체가 상기 메인 메모리에 저장되면, 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 전달된 상기 데이터의 제1 페이지의 개개의 부분들로부터 상기 데이터의 제1 페이지를 리어셈블리하는 동작; 및

상기 메인 메모리 내의 상기 데이터의 리어셈블리된 제1 페이지의 위치를 가리키도록, 상기 데이터의 제1 페이지와 연관된 페이지 테이블 엔트리를 업데이트하는 동작을 더 포함하는,

컴퓨터-판독가능 저장 디바이스.

**청구항 20**

제17항 또는 제19항에 있어서,

상기 동작들은,

페이지 테이블 스캐너를 이용한 페이지 테이블의 스캔에 기반하여, 액세스 비트가 상기 페이지 테이블의 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정하는 동작 - 상기 액세스 비트는 상기 페이지 테이블 엔트리와 연관된 페이지가 마지막 스캔 기간에서 액세스되었는지 여부를 표시하고, 상기 제1 페이지 사이즈를 갖는 적어도 하나의 페이지는 상기 제2 페이지 사이즈의 페이지들로 분할되며, 상기 페이지 테이블 내의 상기 제2 페이지 사이즈의 페이지들 각각에 대한 페이지 테이블 엔트리가 스캐닝됨 -;

상기 액세스 비트가 상기 페이지와 연관된 페이지 테이블 엔트리에 대해 세팅되지 않는다고 결정하는 것에 대한 응답으로 각각의 페이지에 대한 카운트를 증분시키는 동작;

상기 액세스 비트가 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정한 이후, 상기 액세스 비트를

리세팅하는 동작;

분할되었던 상기 제1 페이지 사이즈를 갖는 페이지로 상기 제2 페이지 사이즈의 페이지들을 리어셈블리하는 동작;

상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분이 할당될 수 없다면, 상기 각각의 페이지에 대한 카운트에 기반하여 상기 제1 페이지 사이즈를 갖는 가장 적게 사용된 페이지를 결정하고, 상기 가장 적게 사용된 페이지를 상기 2차 메모리로 릴리즈하며, 그리고 상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분을 상기 릴리즈된 가장 적게 사용된 페이지의 위치에 할당하는 동작; 및

상기 제1 페이지 사이즈와 동등한 상기 메인 메모리의 부분이 할당될 수 있다면, 상기 2차 메모리로부터 상기 메인 메모리의 할당된 부분으로 상기 데이터의 제1 페이지의 제1 부분을 전달하는 동작을 더 포함하는,

컴퓨터-판독가능 저장 디바이스.

## 발명의 설명

### 기 술 분 야

[0001] [0001] 본 명세서는 일반적으로 메모리 시스템들에 관한 것이다.

### 배 경 기 술

[0002] [0002] 광범위하게 다양한 메모리 디바이스들은 다양한 컴퓨터들 및 유사한 시스템들에 대한 데이터 및 명령들을 유지 및 저장하는 데 사용될 수 있다. 종래의 컴퓨팅 시스템들에서, 동적 랜덤 액세스 메모리(DRAM) 기술은 통상적으로, 애플리케이션이 고속으로 동작되기 위해 컴퓨터의 동적 메모리를 동작시키는 데 이용되었다. 그러나, 컴퓨터 시스템들에서 메인 메모리로서 사용되는 DRAM은 더 이상 과거만큼 신속하게 확장되고 있지 않다. 그 결과, DRAM 저장소는 컴퓨팅 환경들에서 제한된 리소스가 되었다.

### 발명의 내용

[0003] [0003] 제2 티어(tier)의 메모리, 이를테면 디스크-기반 메모리, NAND 플래시 메모리, STT-MRAM(spin torque transfer magnetic memory), ReRAM(resistive random access memory) 등이 사용될 수 있다. 제2 티어의 메모리는 메모리 또는 IO 버스를 통해 로컬적으로 또는 고속 네트워크를 통해 원격으로 액세스될 수 있다. 그러나, 애플리케이션들은 데이터 배치를 명시적으로 관리할 필요가 있거나, 또는 시스템은 메모리 티어들 사이에서 데이터를 투명하게 이동시키는 자동 관리를 제공해야 한다. 부가적으로, 거대한 페이지(huge page)들 또는 큰 페이지들 또는 슈퍼 페이지들(이들 용어들은 상호교환가능하게 사용됨)은, 대부분의 작업로드들 및 특히 클라우드-기반 서빙 애플리케이션들에 대해 상당한 성능 증가를 제공하는 것으로 나타났으며, 여기서, 거대한 페이지들은 프로세서 아키텍처에 의존하여, 4KB일 수 있는 통상적인 페이지보다 사이즈가 큰 메모리의 블록들, 예컨대, 8KB, 64KB, 256KB, 1MB, 2MB, 4MB, 16MB, 256MB, 512MB, 또는 1GB이다. 따라서, 기존의 기법들의 부적절성들을 극복하기 위해 최소 성능 영향을 갖는 자동 관리에 대한 새로운 기법들이 필요하다.

[0004] [0004] 본 명세서에 설명된 청구 대상의 하나의 혁신적인 양상은, 메인 메모리로부터, 데이터의 제1 페이지의 제1 부분에 포함된 데이터에 액세스하기 위한 요청을 수신하는 것 - 데이터의 제1 페이지는 제1 페이지 사이즈를 갖고, 제1 부분은 제1 페이지 사이즈보다 작은 제2 페이지 사이즈를 포함함 -; 데이터의 제1 페이지가 메인 메모리에 저장되지 않고 2차 메모리에 저장된다고 결정하는 것에 기반하여 페이지 폴트(page fault)를 개시하는 것; 페이지 폴트를 개시하는 것에 대한 응답으로, 제1 페이지 사이즈와 동등한 메인 메모리의 부분을 할당하는 것; 데이터의 제1 페이지 전체를 전달하지 않으면서 2차 메모리로부터 메인 메모리의 할당된 부분으로 데이터의 제1 페이지의 제1 부분을 전달하는 것 - 데이터의 제1 페이지의 나머지 양은 2차 메모리에 저장된 채로 유지됨 -; 및 데이터의 제1 페이지의 제1 부분이 전달되는 메인 메모리의 할당된 부분의 위치를 가리키도록 데이터의 제1 페이지의 제1 부분과 연관된 제1 페이지 테이블 엔트리를 업데이트하는 것을 포함하는 시스템들 및 방법들로 구현된다.

[0005] [0005] 특정한 구현들에서, 데이터의 제1 페이지의 나머지 양은 2차 메모리로부터 메인 메모리로 전달된다. 데이터의 제1 페이지의 나머지 양을 전달하는 것은, 데이터의 제1 페이지 전체가 메인 메모리에 저장될 때까지 2차 메모리로부터 메인 메모리의 할당된 부분으로, 데이터의 제1 페이지의, 제2 페이지 사이즈에 대응하는 개개의 부분들을 반복적으로 전달하는 것; 및 메인 메모리 내의 데이터의 제1 페이지의 개개의 부분들의 개개의 위

치들을 가리키도록 데이터의 제1 페이지의 개개의 부분들 각각에 대한 개개의 페이지 테이블 엔트리를 업데이트하는 것을 포함할 수 있다.

[0006] [0006] 특정한 구현들에서, 일단 데이터의 제1 페이지 전체가 메인 메모리에 저장되면, 데이터의 제1 페이지는 2차 메모리로부터 메인 메모리의 할당된 부분으로 전달된 데이터의 제1 페이지의 개개의 부분들로부터 리어셈블리되고; 그리고 데이터의 제1 페이지와 연관된 페이지 테이블 엔트리는 메인 메모리 내의 데이터의 리어셈블리된 제1 페이지의 위치를 가리키도록 업데이트된다.

[0007] [0007] 본 명세서에 설명된 청구 대상의 다른 양상은, 2차 메모리로부터 메인 메모리로 데이터의 제1 페이지의 나머지 부분을 전달하기 전에, 액세스되도록 요청되었던 데이터의 제1 페이지의 제1 부분이 메인 메모리로 전달되었다는 것을 표시하는 것을 포함하는 시스템들 및 방법들로 구현된다.

[0008] [0008] 본 명세서에 설명된 청구 대상의 다른 혁신적인 양상은, 페이지 테이블 스캐너를 이용한 페이지 테이블의 스캔에 기반하여, 액세스 비트가 페이지 테이블의 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정하는 것 - 액세스 비트는 페이지 테이블 엔트리와 연관된 페이지가 마지막 스캔 기간에서 액세스되었는지 여부를 표시하고, 제1 페이지 사이즈를 갖는 페이지들 중 적어도 하나는 제2 페이지 사이즈의 페이지들로 분할되며, 페이지 테이블 내의 제2 페이지 사이즈의 페이지들 각각에 대한 페이지 테이블 엔트리가 스캐닝됨 -; 액세스 비트가 페이지와 연관된 페이지 테이블 엔트리에 대해 세팅되지 않는다고 결정하는 것에 대한 응답으로 각각의 페이지에 대한 카운트를 증분시키는 것; 및 액세스 비트가 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정한 이후, 액세스 비트를 리세팅하는 것을 포함하는 시스템들 및 방법들로 구현된다.

[0009] [0009] 특정한 구현들에서, 제1 페이지 사이즈와 동등한 메인 메모리의 부분이 할당될 수 없다면, 제1 페이지 사이즈를 갖는 가장 적게 사용된 페이지들 중 하나는 각각의 페이지에 대한 카운트에 기반하여 결정되고, 가장 적게 사용된 페이지들 중 하나는 2차 메모리로 릴리즈(release)되고, 제1 페이지 사이즈와 동등한 메인 메모리의 부분은 가장 적게 사용된 페이지들 중 릴리즈된 페이지의 위치에 할당되며; 그리고 제1 페이지 사이즈와 동등한 메인 메모리의 부분이 할당될 수 있다면, 데이터의 제1 페이지의 제1 부분이 2차 메모리로부터 메인 메모리의 할당된 부분으로 전달된다.

[0010] [0010] 이들 양상들의 다른 실시예들은 컴퓨터 저장 디바이스들 상에서 인코딩된 방법들의 액션들을 수행하도록 구성되는 대응하는 시스템들, 장치, 및 컴퓨터 프로그램들을 포함한다.

[0011] [0011] 본 명세서에 설명된 청구 대상의 특정한 실시예들은 다음의 장점들 중 하나 이상을 실현하기 위해 구현될 수 있다. 예컨대, 메모리 내의 페이지들에 대한 사용도 또는 액세스 통계들은, 액세스 통계들이 단지 페이지 레벨보다는 서브-페이지 레벨에서 결정될 수 있기 때문에, 소프트웨어 및 샘플링 기법들을 수반하는 현재의 방법들보다 더 정밀하고 정확할 수 있다. 추가로, 데이터의 전체 페이지보다는 액세스가 처음 요청된 페이지의 특정한 부분을 전달함으로써, 페이지에 액세스하기 위한 요청에 대한 응답으로 페이지를 전달함으로써 야기되는 지연이 감소될 수 있으며, 이는 실행되는 애플리케이션 또는 프로세스의 더 빠른 실행을 초래할 수 있다. 다른 장점은, 시스템이 거대한 페이지들의 이점들, 이를테면, 더 적은 레벨들의 페이지 테이블들 및 더 양호한 TLB(translation lookaside buffer) 커버리지로 인한 더 양호한 메모리 액세스 성능을 이용하며, 작은 페이지를 전달하는데에만 필요한 페이지 폴트를 서비스하는 감소된 레이턴시로 인해 더 양호한 페이지 폴트 성능을 제공하는 작은 페이지 입도로 페이징을 여전히 수행할 수 있다는 것이다. 따라서, 거대한 페이지들의 메모리 액세스 이점들 및 작은 페이지들의 요구 페이징 이점들 둘 모두가 달성될 수 있다. 또한, 필요한 데이터만이 작은 페이지 사이즈에 따라 전달될 수 있으며, 그 결과, 거대한 페이지들을 직접 페이징하는 것과 비교하여 핫(hot) 데이터를 메인 메모리에 그리고 콜드(cold) 데이터를 2차 메모리에 더 양호하게 유지하는 것으로 인해, 메인 메모리는 불필요한 데이터로 점유되지 않는다.

[0012] [0012] 본 발명의 하나 이상의 실시예들의 세부사항들은 첨부한 도면들 및 아래의 설명에서 기재된다. 본 발명의 다른 특징들 및 장점들은 설명, 도면들, 및 청구항들로부터 명백해질 것이다.

## 도면의 간단한 설명

[0013] [0013] 도 1은 본 개시내용의 구현들에 따른, 메모리 디바이스를 포함하는 시스템의 일 예를 도시한다.

[0014] [0014] 도 2는 본 개시내용의 구현들에 따른, 메모리 디바이스를 포함하는 시스템의 일 예를 도시한다.

[0015] [0015] 도 3a는 본 개시내용의 구현들에 따른, 가상 메모리를 물리적 메모리에 맵핑시키기 위한 페이지 테이블의 일 예를 도시한다.

[0016] 도 3b는 본 개시내용의 구현들에 따라 할당된 메모리의 일부의 일 예를 도시한다.

[0017] 도 4는 본 개시내용의 구현들에 따른, 메모리 관리를 위한 프로세스의 일 예의 흐름도를 도시한다.

[0018] 다양한 도면들 내의 유사한 참조 번호들 및 지정들은 유사한 엘리먼트들을 표시한다.

## *발명을 실시하기 위한 구체적인 내용*

[0014] [0019] 거대한 페이지들은, 대부분의 작업로드들 및 특히 클라우드-기반 서빙 애플리케이션들에 대해 상당한 성능 증가를 제공하는 것으로 나타났다. 용어 "거대한 페이지들"이 본 명세서에서 사용될 수 있지만, 그 용어는, 특정한 아키텍처가 핸들링할 수 있는 가장 작은 사이즈의 페이지, 즉 작은 페이지 또는 그의 표준 페이지 사이즈보다 큰 임의의 사이즈의 페이지에 적용된다. 예컨대, 특정한 아키텍처에 대한 가장 작은 페이지 사이즈 또는 표준 페이지 사이즈는 4KB일 수 있고, 거대한 페이지는 2MB일 수 있다. 다른 구현들에서, 예컨대, 거대한 페이지는 8KB, 64KB, 256KB, 1MB, 2MB, 4MB, 16MB, 256MB, 512MB, 또는 1GB 이상, 또는 그들 사이의 임의의 사이즈일 수 있다. 예컨대, 거대한 페이지는 4KB의 임의의 정수배, 즉 n*4KB일 수 있으며, 특정한 실시예들에서는 표준 페이지 사이즈의 2배의 임의의 거듭제곱(power)일 수 있다. 본 개시내용의 실시예들은, 제2 티어의 더 느린 메모리(종종, 2차 메모리로 지칭됨)에 페이징하기 위해 종래의 작은 페이지들을 사용하면서, 메인 메모리(예컨대, DRAM 캐시)에 액세스하기 위해 거대한 페이지들을 사용할 수 있는 새로운 방식을 도입한다. 특정한 실시예들은 더 느린 메모리에 액세스하기 위해 사용되는 상호연결의 타입에 기반하여 수정될 수 있다. 예컨대, 맞춤화된 커널 드라이버에 기반한 소프트웨어-기반 솔루션이 IO 상호연결을 위해 구현될 수 있다. 추가로, 예컨대, 거대한 페이지들을 관리하기 위한 하드웨어 솔루션이 캐시 코히런트 상호연결을 위해 구현될 수 있다.

[0015] [0020] 따라서, 본 개시내용의 실시예들은, 메모리 또는 IO 버스를 통해 로컬적으로 또는 네트워크를 통해 원격으로 이용가능한 2차 메모리의 고성능 자동 관리를 위한 시스템을 제공한다. 2차 메모리는 디스크-기반일 수 있고, 속성상 비-휘발성이고 영구적인 컴퓨터 메모리일 수 있다. 2차 메모리는 프로세서에 의해 직접 액세스되지 않을 수 있고, 1차 또는 메인 메모리보다 느릴 수 있다. 1차 메모리, 1차 저장소, 내부 메모리 또는 제1-티어 메모리로 또한 지칭되는 메인 메모리는 CPU에 직접 액세스가능할 수 있다. 아래에서 더 상세히 설명되는 바와 같이, 최적화된 커널 드라이버는, 예컨대 제2 티어의 메모리에 대한 빠른 경로를 제공하고, 메모리 관리 하드웨어와의 모든 통신을 핸들링할 수 있다. 그 프로세스는, 동기화, 메모리 관리, 및 블록 IO 전달들과 같은 것들에 많은 비용들을 발생시키는, 페이징을 위해 커널을 통하는 기존의 경로들과 비교하여 유리하다.

[0016] [0021] 이들 특징들 및 부가적인 특징들은 아래에서 더 상세히 설명된다.

[0017] [0022] 도 1은 본 개시내용의 구현들에 따른, 메모리 디바이스를 포함하는 시스템(100)의 일 예를 도시한다. 중앙 프로세싱 유닛(CPU)(110)은 DRAM(120) 및 메모리 관리 유닛(MMU)(150)의 형태의 메인 메모리와 통신할 수 있다. 시스템(100)은 네트워크를 통해 액세스될 수 있는 원격 메모리(130)의 형태의 2차 메모리를 더 포함할 수 있다. MMU(150)는 메모리의 관리에서 동작할 수 있다. 부가적으로, 페이지 테이블 워커(walker)(160) 및 TLB(translation lookaside buffer)(165)는 MMU(150)의 일부이거나 또는 그것과 함께 구현될 수 있다. 부가적으로, 시스템(100)은 물리적 메모리로서 DRAM(170)을 포함할 수 있다.

[0018] [0023] MMU(150)는, 메모리 참조들이 통과되어 가상 메모리 어드레스들의 물리적 어드레스로의 변환을 수행하고 캐시 제어를 핸들링할 수 있는 하드웨어 유닛이다. 예컨대, MMU(150)는 가상 페이지 넘버들을 메인 메모리 내의 물리적 페이지 넘버들에 맵핑시키기 위해 페이지당 하나의 페이지 테이블 엔트리(PTE)를 포함하는 메모리-내 테이블로서 페이지 테이블을 사용할 수 있다. 변환 색인 버퍼(165)는 PTE들의 연관 캐시로서, 가상 어드레스가 맵핑될 때마다 메인 메모리에 액세스할 필요성을 피하기 위해 사용될 수 있다. 예컨대, 어떠한 물리적 랜덤 액세스 메모리도 그 가상 페이지에 할당되지 않았기 때문에, PTE가 가상 페이지에 대한 액세스를 금지할 경우, MMU(150)는 페이지 폴트를 CPU(110)에게 시그널링할 수 있다.

[0019] [0024] CPU(110)는, 프로세싱 레이턴시를 감소시키기 위해 데이터의 일시적인 카피(copy)들을 포함하도록 구성될 수 있는, 프로세서에 구축된 작은 양의 빠른 메모리일 수 있는 캐시를 가질 수 있다. TLB(165)는, CPU(110)가 각각의 메모리 액세스에서 체크할 수 있는 최근에 사용된 페이지들의 고정-사이즈 어레이일 수 있다. TLB(165)는, DRAM(170) 내의 물리적 페이지들이 현재 할당된 가상 어드레스 범위들을 열거할 수 있다. 따라서, 예컨대, TLB(165)는 MMU(150)에 대한 캐시로서 기능할 수 있다. 이러한 방식으로, TLB(165)에 열거된 가상 어드레스들에 대한 액세스들은 연관된 물리적 메모리, 예컨대 DRAM(170)으로 직접 진행될 수 있다. 부가적으로, TLB(165)에 열거되지 않은 가상 어드레스들에 대한 액세스들, 즉 TLB 미스는, 하드웨어에 의해 또는 페이지 폴

트 핸들러에 의해 수행될 수 있는 페이지 테이블 룩업을 트리거링할 수 있다.

[0020] [0025] 도 2는 본 개시내용의 구현들에 따른, 메모리 디바이스를 포함하는 시스템(200)의 일 예를 도시한다. 시스템(200)은 CPU(220) 및 물리적 어드레스 공간(240)을 포함할 수 있다. MMU(230)는 가상 어드레스들을 해석하여, 대응하는 물리적 어드레스들을 식별할 수 있다. 예컨대, 가상 어드레스들에서 메모리를 판독, 기입, 또는 실행하려는 시도들은 대응하는 물리적 어드레스들로 변환될 수 있거나, 또는 인터럽트, 즉 페이지 폴트는 소프트웨어가 시도된 액세스에 응답하게 허용하도록 생성될 수 있다. 물리적 메모리 어드레스들은, 주어진 판독 또는 기입 동작과 연관된 물리적 메모리를 구성하는 저장 하드웨어의 일부분 내의 특정 메모리 셀 또는 일부를 식별할 수 있다. 가상 메모리는 메모리 어드레스들의 소프트웨어-제어된 세트, 예컨대 가상 어드레스 공간을 제공할 수 있으며, 각각의 프로세스, 예컨대, 프로세스 A(205) 및 프로세스 B(210)가 커널 공간 및 사용자 공간을 포함할 수 있는 그 자신의 가상 메모리 어드레스 범위를 갖게 허용할 수 있다. 가상 어드레스들은, 가상 어드레스 범위들을 연관된 저장된 콘텐츠에 맵핑시킬 수 있는 페이지 테이블들을 사용하여 MMU(230)에 의해 해석될 수 있다. 프로세서에 대한 가장 작은 어드레싱가능 유닛이 바이트 또는 워드일 수 있지만, MMU(230)는 페이지들로 메모리를 관리할 수 있다.

[0021] [0026] 도 3a는 본 개시내용의 구현들에 따른, 가상 메모리(310)를 물리적 메모리(330)에 맵핑시키기 위한 페이지 테이블(320)의 일 예를 도시한다. 페이지 테이블들(320)은, 프로세스에 대한 메모리 맵핑들의 리스트를 포함하는 데이터 구조들일 수 있으며, 연관된 리소스들을 추적하는 데 사용될 수 있다. 예컨대, 각각의 프로세스는 그 자신의 세트의 페이지 테이블들을 가질 수 있다. 가상 어드레스 공간, 예컨대 가상 메모리(310)는, 특정한 사이즈의 어드레스들의 인접한 범위일 수 있는 페이지들로 분할될 수 있다. 페이지들은, 페이지의 시작 어드레스가 페이지 사이즈의 배수이도록 구조화될 수 있다. 위에서 설명된 바와 같이, MMU(230)는, 가상 메모리(310)로부터 페이지들의 가상 어드레스들을 해석하고 물리적 메모리(330) 내의 페이지 프레임들의 대응하는 물리적 어드레스들을 식별하기 위해 페이지 테이블(320)을 사용할 수 있다. 부가적으로, 페이지 테이블들은 계층적 또는 멀티-레벨, 해시-기반 등일 수 있으며, 이는 거대한 페이지들에 대한 장점을 제공하여, 더 빠른 페이지 테이블 워크로 계층구조를 높인다.

[0022] [0027] 위에서 참조된 바와 같이, 2차 메모리 또는 제2 티어의 메모리, 이를테면 디스크-기반 메모리 또는 다른 제2 티어 메모리는 메인 메모리 또는 1차 메모리, 이를테면 DRAM보다 느릴 수 있다. 특정한 구현들에 따르면, 맞춤화된 커널 드라이버는 거대한 페이지들을 이용하여 제2 티어의 메모리를 관리할 수 있다. 커널 드라이버는, DRAM의 캐시에 대해 거대한 페이지들의 배수들인 인접한 구역들에 물리적 메모리를 예비할 수 있다. 애플리케이션이 부가적인 메모리를 필요로 할 경우, 커널 드라이버는 거대한 페이지 배수들에, 즉 거대한 페이지의 사이즈의 배수들에 공간을 할당할 수 있다. 커널 드라이버는 페이지 교체 정책을 구현할 수 있으며, 교체에 대한 데이터가 선택될 경우, 거대한 페이지는 제2 티어의 메모리로 페이징 아웃(page out)될 수 있다. 그 프로세스는 메모리 내의 데이터에 대한 액세스를 요청했던 구동중인 애플리케이션에 비동기적으로 발생할 수 있다.

[0023] [0028] 애플리케이션이 제2 티어의 메모리에 상주하는 데이터에 대한 액세스에 폴트가 있는 경우, 페이지 폴트 핸들러는 요청된 캐시 라인을 포함하는 단일의 작은 페이지만을 제2 티어의 메모리로부터 메인 메모리, 예컨대 DRAM으로 전달할 수 있다. 그러나, 특정한 구현들에 따르면, 거대한 페이지를 구성하는 각각의 작은 페이지의 상태가 추적될 수 있다. 따라서, 예컨대, 커널 드라이버가 거대한 페이지 내에서 모든 또는 미리 결정된 양의 작은 페이지들에 폴트가 있는 경우, 기존의 PTE들을 거대한 페이지에 대한 단일 PTE로 대체하고 TLB(165)로부터 임의의 관련있는 TLB 엔트리들을 플러싱(flush)함으로써, 임의의 나머지 작은 페이지들을 페이징-인(page-in)하고 작은 페이지들을 다시 DRAM의 거대한 페이지로 합치거나(coalesce) 리어셈블리하기 위한 결정이 이루어질 수 있다.

[0024] [0029] 따라서, DRAM에 상주하는 데이터에 대한 거대한 페이지들의 이점들이 유지될 수 있으면서, 페이지 폴트들의 비용이 또한, 작은 페이지를 전달한 이후 폴트 핸들러 프로세스를 완료함으로써 감소될 수 있다. 예컨대, 거대한 페이지들을 사용하는 것은, 더 큰 입도로 데이터를 추적하는 것이 더 적은 엔트리들을 갖는 더 작은 페이지 테이블을 가능하게 하기 때문에, 리소스 오버헤드를 감소시키는 장점을 제공할 수 있다. 그러나, 거대한 페이지들을 사용하는 것은, 시스템이 항상 제2 티어의 메모리에 거대한 페이지들을 기입하면, 전체 기입 대역폭이 증가하게 할 수 있다. 또한, "핫"한, 예컨대 빈번하게 사용되거나 또는 최근에 사용된 거대한 페이지 내의 작은 페이지들이 2차의 더 느린 메모리로 페이징 아웃되게 하여, 그 "핫" 데이터에 대한 부가적인 폴트들을 초래할 수 있는 가능성이 존재한다. 특정한 구현들에 따르면, 그 이슈들은, 거대한 페이지 및 작은 페이지 통계들에 기반하여 거대한 페이지들을 분할하거나 합칠 때를 커널 드라이버가 동적으로 결정함으로써 완화될 수 있

다. 예컨대, 아래에서 더 상세히 설명되는 바와 같이, 거대한 페이지들은, PTE 액세스 비트들을 통해 거대한 페이지 내의 작은 페이지들에 대한 통계들을 수집하기 위해 주기적으로 분할될 수 있다. 추가로, 드라이버는 거대한 페이지들 및 작은 페이지들 둘 모두를 유지할 수 있어서, 거대한 페이지 내의 미리 결정된 수의 작은 페이지들이 "핫"하거나 또는 빈번하게 또는 최근에 액세스된 경우, 작은 페이지들은 거대한 페이지로 이주되어 병합될 수 있다. 반대로, 거대한 페이지 내의 너무 많은 서브-페이지들, 즉 작은 페이지들이 "콜드"하면, 거대한 페이지는 작은 페이지들로서 분할 및 프로세싱될 수 있다.

[0025] [0030] 특정한 구현들에 따르면, 메인 메모리에 저장되지 않은 데이터에 액세스하려는 시도가 이루어져서 페이지 폴트가 발생하는 경우, 전체의 거대한 페이지가 메인 메모리로 전달되는 것이 아니라, 오히려, 액세스가 요청된 데이터를 포함하는 더 작은 덩어리의 데이터, 예컨대 작은 페이지가 2차 메모리로부터 메인 메모리로 전달될 수 있다. 따라서, 애플리케이션은 요청된 데이터에 액세스하여 계속 구동될 수 있다. 후속하여, 거대한 페이지의 나머지는 백그라운드에서 메인 메모리로 전달될 수 있으며, 페이지 테이블 엔트리는 그에 따라 업데이트될 수 있다. 이러한 방식으로, 요청된 데이터는 더 신속하게 액세스될 수 있고, 시스템은 거대한 페이지들을 관리하는 이점들을 여전히 누릴 수 있다. 다시 말하면, 2차 메모리로부터 거대한 페이지들을 판독하고 거대한 페이지들을 메인 메모리에 기입하는 데 요구되는 시간은, 작은 페이지들을 판독 및 기입하는 데 요구되는 시간보다 크며; 따라서, 액세스되도록 요청된 데이터를 포함하는 작은 페이지만을 판독하는 것은, 애플리케이션 또는 프로세싱 스레드가 중지되거나 또는 데이터가 2차 메모리로부터 메인 메모리로 전달되는 것을 대기하는 시간을 감소시킨다. 따라서, 데이터를 메인 메모리로 전달하기 위한 레이턴시 시간을 감소시키는 것은, 메인 메모리 밖으로 전달된 데이터가 통상적으로, 동작 성능에 거의 영향을 주지 않거나 또는 어떠한 영향도 주지 않으면서 백그라운드에서 전달되는 "콜드" 데이터의 페이지이기 때문에, 데이터를 다시 2차 메모리로 전달하기 위한 시간보다 성능 결정적인 것으로서 더 중요하지만, 메인 메모리로 전달되는 데이터는 애플리케이션 또는 프로세싱 스레드의 실행을 지연시킬 수 있다.

[0026] [0031] 위에서 설명된 바와 같이, 페이지 폴트는, 스레드 또는 구동중인 프로그램이 가상 어드레스 공간에 맵핑되지만 메인 메모리에 실제로는 로딩되지 않은 메모리 페이지에 액세스할 경우 발생할 수 있다. MMU(150) 또는 페이지 폴트 핸들러는 페이지 폴트를 검출할 수 있으며, 페이지 폴트가 검출되는 경우, 메모리에 사용중이지 않은 공간(free space)이 존재하는지 여부에 관한 결정이 이루어질 수 있다. 사용중이지 않은 공간이 존재하면, 페이지 데이터는 2차 저장소로부터 메모리 내의 사용중이지 않은 페이지 위치로 카피될 수 있다. 사용중이지 않은 페이지가 존재하지 않으면, 페이지는, 예컨대 큐의 메모리 내의 모든 페이지들을 추적할 수 있는 FIFO 큐로부터 가져와질 수 있으며, 가장 최근의 도착 항목이 뒤에 있고 가장 오래된 도착 항목이 앞에 있다. 그 페이지가 오염되면, 즉 수정되었다면, 시스템은 페이지를 2차 메모리에 기입할 수 있다. 메인 메모리로부터 2차 메모리로 페이지를 전달할 시에, 페이지와 연관된 페이지 테이블 엔트리는 무효화될 수 있고, 페이지와 연관된 임의의 엔트리들에 대한 TLB 슈트다운(shootdown)이 실행될 수 있으며, 예컨대, TLB 엔트리들로 하여금 다른 프로세서들에 대해 플러싱되게 한다. 그 페이지가 이제 사용중이지 않으면, 페이지 데이터는 2차 저장소로부터 사용중이지 않은 페이지 위치로 카피될 수 있다. 페이지 테이블들은, 페이지의 메인 메모리 내의 위치를 가리키도록 페이지와 연관된 PTE를 업데이트시킴으로써 유효한 PTE를 생성하기 위해 업데이트될 수 있다. 일단 페이지 폴트가 핸들링되면, 스레드 또는 구동중인 프로그램은, 그것이 액세스하도록 요청했던, 이제 메인 메모리에 있는 데이터로 재개될 수 있다.

[0027] [0032] 도 3b는 본 개시내용의 구현들에 따라 할당된 메모리(305)의 일부의 일 예를 도시한다. 요구 페이징의 경우, "콜드" 데이터, 예컨대 몇몇 임계 액세스 레이트보다 작은 레이트로 액세스되거나 또는 특정한 시간 기간 동안 액세스되지 않았던 데이터의 페이지는 2차 저장소로 기입될 필요가 있을 수 있으며, 애플리케이션이 데이터에 액세스하려고 시도하고 페이지 폴트가 발생한 경우, 데이터의 페이지는 다시 메인 메모리로 전달될 필요가 있을 수 있다. 특정한 구현들에 따르면, 메인 메모리를 페이징 인 및 페이징 아웃하는 프로세스는, 프로세서의 관점으로부터 시스템이 거대한 페이지들로만 작업하는 것처럼 발생한다. 다시 말하면, 거대한 페이지는, 콜드인 경우(즉, 빈번하게 또는 최근에 사용되지 않은 경우) 메인 메모리 밖으로 내보내질 수 있고, 그리고 페이지가 메인 메모리 내로 들어올 필요가 있는 경우, 거대한 페이지의 일부만, 예컨대 작은 페이지만 초기에 전달되더라도, 메모리의 거대한 페이지 전체가 할당될 수 있다. 따라서, 할당된 메모리(305)는 거대한 페이지에 대응할 수 있으며, 거대한 페이지에 대응하는 인접한 메모리가 물리적 메모리(325) 뿐만 아니라 가상 메모리(315)에 할당될 수 있다.

[0028] [0033] 예컨대, 페이지 폴트가 발생하고 데이터의 페이지가 메인 메모리로 전달될 필요가 있는 경우, 메모리의 거대한 페이지가 먼저 할당될 수 있다. 이어서, 액세스되도록 요청된 데이터를 포함하는 거대한 페이지 전체를

전달하기보다는, 애플리케이션에 의해 액세스되도록 요청된 데이터를 포함하는 데이터의 서브-페이지 또는 작은 페이지만이 초기에 메인 메모리로 전달될 수 있다. 예컨대, 애플리케이션은 단지 바이트 또는 워드에만 액세스할 필요가 있을 수 있어서, 애플리케이션이 계속하는 데에 거대한 페이지 전체가 요구되지 않으며, 시스템은, 애플리케이션이 계속 구동하는 데 필요한 데이터를 포함하는 데이터의 더 작은 부분, 예컨대 작은 페이지만을 메인 메모리로 전달할 수 있다. 그 작은 페이지의 전달 시에, 요청된 데이터가 메인 메모리로 전달되었거나 또는 이제 메인 메모리로부터 액세스되는 데 이용가능하다는 표시가 애플리케이션에 대해 이루어질 수 있다.

[0029] [0034] 후속하여, 거대한 페이지 전체가 메인 메모리로 전달되지 않았기 때문에, 거대한 페이지의 나머지 부분이 백그라운드에서 메인 메모리로 전달될 수 있다. 대안적으로, 예컨대, 액세스 통계들에 기반하여, 거대한 페이지를 작은 페이지들로 분해하는 것이 유리하다고 결정되며, 그에 의해, 단일의 거대한 페이지로부터 구성의 작은 페이지들로 페이징 데이터 구조들을 변경시킬 수 있다. 메인 메모리로 전달되지 않았던 나머지 작은 페이지들 중 임의의 페이지에 애플리케이션이 후속하여 액세스하면, 페이지 폴트가 발생하면서, 그 작은 페이지들은 그 때에, 즉 액세스되도록 요청된 때에 전달될 수 있다.

[0030] [0035] 거대한 페이지의 나머지 부분을 전달하여 합칠지 또는 거대한 페이지를 작은 페이지들로 분할할지에 관한 결정을 행하기 위해, 액세스 통계들은 거대한 페이지 내의 페이지들 중에서 페이지들의 "온도", 예컨대 "핫" 페이지들 및 "콜드" 페이지들을 식별하도록 수집될 수 있다. 따라서, 특정한 구현들에 따르면, 메모리가 거대한 페이지 덩어리들에 할당되고 예비되지만, 거대한 페이지는 작은 페이지들로 분할되어, 더 작은 페이지 덩어리들로 작업할 수 있다. 거대한 페이지가 작은 페이지들로 분할될 경우, 페이지 테이블은 각각의 작은 페이지에 대한 개개의 PTE로 업데이트될 수 있으며; 거대한 페이지가 리어셈블리될 경우, 페이지 테이블은 각각의 작은 페이지에 대한 개개의 PTE들을 거대한 페이지 전체에 대한 하나의 엔트리로 대체함으로써 업데이트될 수 있다.

[0031] [0036] 페이지들에 대한 액세스 통계들을 수집하는 것은, 페이지들의 액세스를 결정하는, 예컨대 "콜드" 페이지들 및 "핫" 페이지들을 결정하는 임의의 프로세스 또는 수단을 통해 달성될 수 있다. 예컨대, 프로세스는, 거대한 페이지를 작은 페이지들로 주기적으로 분할하는 것, 작은 페이지가 마지막으로 액세스되었던 때 또는 작은 페이지가 액세스되었던 빈도 또는 얼마나 최근에 작은 페이지가 액세스되었는지를 결정하기 위해 페이지들의 세트를 스캐닝하고 액세스 비트를 판독하는 것, 및 이어서, 일단 통계들이 수집되었다면 작은 페이지들을 다시 거대한 페이지로 리어셈블리하는 것을 포함할 수 있다. 이러한 방식으로, 예컨대, 거대한 페이지 내의 서브-페이지들 또는 작은 페이지들에 대한 액세스들의 샘플이 작은 페이지들에 대한 액세스에 관한 통계 데이터를 획득하기 위해 사용될 수 있다.

[0032] [0037] 더 상세하게, 특정한 구현들에서, 페이지 테이블 스캐닝, 즉 페이지 테이블을 통한 스캔은 CPU 오버헤드를 요구하는 소프트웨어보다는 하드웨어를 통해 수행될 수 있으며, 이는, 더 느린 메모리 액세스 및 프로세싱 그리고 소프트웨어를 통했다면 유용했을 캐시 정보를 폐기하는 것을 종종 초래한다. 일반적으로, 페이지 데이터의 액세스 빈도, 예컨대 어느 페이지 데이터가 데이터의 다른 페이지들에 비해 빈번하게 액세스되었는지 및 어느 페이지 데이터가 데이터의 다른 페이지들에 비해 덜 빈번하게 액세스되었는지는 페이지 테이블을 스캐닝함으로써 결정될 수 있다. 페이지 테이블에 맵핑된 각각의 페이지, 예컨대 각각의 PTE는, 페이지가 액세스될 때마다 세팅되고, 이어서, 페이지 테이블을 스캐닝한 이후 CPU에 의해 클리어(clear)될 수 있는 플래그(flag) 또는 액세스 비트를 가질 수 있다.

[0033] [0038] 이러한 하드웨어는, 하나 이상의 페이지 테이블 워커들을 포함할 수 있는 페이지 테이블 워커(160) 또는 MMU(150), 예컨대, 페이지 테이블을 판독하고, 가상-물리적 변환들을 TLB(165)에 자동으로 로딩하기 위한 빌트-인(built-in) 하드웨어를 증대시킴으로써 구현될 수 있다. 따라서, 하드웨어는 프로세서에서 페이지 테이블 스캐닝 메커니즘을 사용하는 프로세서 아키텍처의 일부일 수 있다. 예컨대, 하드웨어는, 페이지 테이블을 통해 스캐닝하여 PTE들을 스캐닝해서, 마지막 스캔 이후로 액세스 비트가 각각의 PTE에서 세팅되었는지를 결정하기 위한 루틴을 구현할 수 있다. 액세스 비트는, 액세스 비트가 세팅되었다고 결정한 이후 클리어될 수 있으며, 이어서, 동작들은 페이지 테이블의 다음 스캔까지 진행될 수 있다. 스캐닝은 주기적으로, 예컨대, 스캔들 사이의 미리 결정된 시간 기간으로 발생할 수 있거나, 또는 스캐닝은 몇몇 외부 이벤트에 의해 트리거링될 수 있다. 액세스 비트 또는 플래그가 세팅되었다고 결정될 때마다, 각각, 카운트는 각각의 페이지에 대해 증분될 수 있다. 대안적으로, 액세스 비트 또는 플래그가 세팅되지 않았다고 결정될 때마다, 각각, 카운트는 각각의 페이지에 대해 증분될 수 있다.

[0034] [0039] 시간의 경과에 따라, 프로파일은 스캐닝으로부터 생성될 수 있고, 프로파일은, 각각의 페이지가 얼마나

빈번하게 그리고/또는 얼마나 최근에 액세스되었는지를 표시할 수 있다. 예컨대, 하드웨어는 사용 통계들, 예컨대 세팅된 액세스 비트 또는 플래그의 카운트를 유지하기 위해 각각의 페이지 또는 블룸 필터(bloom filter)들에 대한 하나 이상의 카운터들을 포함할 수 있거나, 또는 결과들은, 예컨대, 가장 많이 및 가장 적게 사용된 페이지들 또는 더 빈번하게 및 덜 빈번하게 액세스된 페이지들을 분류 및 필터링을 허용하기 위해 메모리, 예컨대 2-레벨 메모리에 저장될 수 있다. 더 상세하게, 하드웨어는 페이지가 얼마나 최근에 액세스되었는지를 결정하기 위해 페이지-당 카운터를 유지할 수 있고, 각각의 카운터는 개개의 PTE가 스캐닝될 경우 업데이트될 수 있다. 페이지-당 카운터들은 빠른 액세스를 위해 온-칩(on-chip) SRAM에서 제공될 수 있다. 대안적으로, 2차 메모리의 사이즈가 커서 카운터들의 영역 비용을 더 높게 만들 수 있기 때문에, 블룸 필터들을 카운팅하는 것은 페이지들의 세트들에 관한 액세스 통계들을 유지하는 데 사용될 수 있으며, 그에 의해 영역을 절약한다. 대안적으로, 하드웨어는 작은 양의 프라이빗(private) DRAM을 사용할 수 있거나 또는 카운터들을 시스템 DRAM에 저장할 수 있다.

[0035] [0040] 따라서, 예컨대, 액세스 통계들에 기반하여, 페이지들은 가장 많이 사용된 것으로부터 가장 적게 사용된 것으로 또는 그 반대로 순서화될 수 있다. 페이지 폴트가 서비스되는 경우, 메인 메모리 DRAM(170)에 어떠한 사용중이지 않은 페이지들이 존재하지 않으면, 페이징 프로세스는, 가장 적게 사용된 페이지들 중 하나를 2차 메모리로 릴리즈하거나 다시 기입할 수 있고, 새로운 페이지를 메인 메모리로 전달하기 위해 그 가장 적게 사용된 페이지의 위치를 사용할 수 있다.

[0036] [0041] 도 4는 본 개시내용의 구현들에 따른, 메모리 관리를 위한 프로세스(400)의 일 예의 흐름도를 도시한다. 프로세스(400)는 410에서, 메인 메모리로부터, 데이터의 제1 페이지의 제1 부분에 포함된 데이터에 액세스하기 위한 요청을 수신하는 것을 포함할 수 있다. 데이터의 제1 페이지는 제1 페이지 사이즈, 예컨대 거대한 페이지를 가질 수 있고, 제1 부분은 제1 페이지 사이즈보다 작은 제2 페이지 사이즈, 예컨대 작은 페이지를 가질 수 있다. 420에서, 페이지 폴트는, 데이터의 제1 페이지가 메인 메모리에 저장되지 않고 2차 메모리에 저장된다고 결정하는 것에 기반하여 개시될 수 있다. 추가로, 430에서, 페이지 폴트를 개시하는 것에 대한 응답으로, 제1 페이지 사이즈, 예컨대 거대한 페이지와 동등한 메인 메모리의 부분이 할당될 수 있다. 440에서, 데이터의 제1 페이지의 제1 부분은 데이터의 제1 페이지 전체를 전달하지 않으면서 2차 메모리로부터 메인 메모리의 할당된 부분으로 전달될 수 있다. 따라서, 데이터의 제1 페이지의 나머지 양은 2차 메모리에 저장된 채로 유지될 수 있다. 450에서, 데이터의 제1 페이지의 제1 부분과 연관된 제1 페이지 테이블 엔트리는, 데이터의 제1 페이지의 제1 부분이 전달되는 메인 메모리의 할당된 부분의 위치를 가리키도록 업데이트될 수 있다. 후속하여, 데이터의 제1 페이지의 나머지 양은, 예컨대 애플리케이션이 계속 구동되는 동안 백그라운드에서 2차 메모리로부터 메인 메모리로 전달될 수 있다.

[0037] [0042] 데이터의 제1 페이지의 나머지 양을 전달하기 위해, 데이터의 제1 페이지의, 제2 페이지 사이즈에 대응하는 개개의 부분들은, 데이터의 제1 페이지 전체가 메인 메모리에 저장될 때까지 2차 메모리로부터 메인 메모리의 할당된 부분으로 전달될 수 있다. 추가로, 데이터의 제1 페이지의 개개의 부분들 각각에 대한 개개의 페이지 테이블 엔트리는, 메인 메모리 내의 데이터의 제1 페이지의 개개의 부분들의 개개의 위치들을 가리키도록 업데이트될 수 있다. 또한, 일단 데이터의 제1 페이지 전체가 메인 메모리에 저장되면, 데이터의 제1 페이지는 2차 메모리로부터 메인 메모리의 할당된 부분으로 전달된 데이터의 제1 페이지의 개개의 부분들로부터 합쳐지거나 또는 리어셈블리될 수 있다. 리어셈블리에 따르면, 데이터의 제1 페이지와 연관된 페이지 테이블 엔트리는 메인 메모리 내의 데이터의 리어셈블리된 제1 페이지의 위치를 가리키도록 업데이트될 수 있다.

[0038] [0043] 특정한 구현들에서, 2차 메모리로부터 메인 메모리로 데이터의 제1 페이지의 나머지 부분을 전달하기 전에, 시스템은, 액세스되도록 요청되었던 데이터의 제1 페이지의 제1 부분이 메인 메모리로 전달되었다는 것을 표시할 수 있어서, 액세스를 요청했던 애플리케이션 또는 스레드가 메인 메모리 내의 요청된 데이터에 액세스함으로써 계속 구동되게 할 수 있다.

[0039] [0044] 메모리 관리를 위한 프로세스의 일 예는 또한, 페이지 테이블 스캐너를 이용한 페이지 테이블의 스캔에 기반하여, 액세스 비트가 페이지 테이블의 각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정하는 것을 포함할 수 있다. 그러한 프로세스에서, 액세스 비트는, 페이지 테이블 엔트리와 연관된 페이지가 마지막 스캔 기간에 액세스되었는지 여부를 표시할 수 있다. 위에서 설명된 바와 같이, 제1 페이지 사이즈를 갖는 페이지들 중 적어도 하나, 예컨대 거대한 페이지는 제2 페이지 사이즈의 페이지들, 예컨대 작은 페이지들로 분할될 수 있으며, 페이지 테이블 내의 제2 페이지 사이즈의 페이지들 각각에 대한 페이지 테이블 엔트리가 스캐닝된다. 특정한 구현들에서, 각각의 페이지에 대한 카운트는, 액세스 비트가 페이지와 연관된 페이지 테이블 엔트리에 대해 세팅되지 않는다고 결정하는 것에 대한 응답으로 증분될 수 있다. 후속하여, 액세스 비트가

각각의 페이지 테이블 엔트리에 대해 세팅되는지 여부를 결정한 이후, 액세스 비트가 리세팅될 수 있다.

[0040] [0045] 특정한 구현들에서, 제1 페이지 사이즈와 동등한 메인 메모리의 부분이 할당될 수 없다면, 제1 페이지 사이즈를 갖는 가장 적게 사용된 페이지들 중 하나는 각각의 페이지에 대한 카운트에 기반하여 결정될 수 있고, 결정된 가장 적게 사용된 페이지는 2차 메모리로 릴리즈될 수 있다. 따라서, 제1 페이지 사이즈와 동등한 메인 메모리의 부분은 가장 적게 사용된 페이지들 중 릴리즈된 페이지의 위치에 할당될 수 있다. 반대로, 제1 페이지 사이즈와 동등한 메인 메모리의 부분이 할당될 수 있다면, 데이터의 제1 페이지의 제1 부분이 2차 메모리로부터 메인 메모리의 할당된 부분으로 전달될 수 있다.

[0041] [0046] 더 상세하게, 예컨대, 메인 메모리가 사용중이지 않은 페이지를 갖지 않고 페이지 전달을 수신할 수 없으면, 메인 메모리 내의 가장 적게 사용된 페이지들 중 하나는 각각의 페이지에 대한 카운트에 기반하여 결정될 수 있다. 페이지 폴트 핸들러 또는 제어기는 페이지 전달을 관리할 수 있으며, 가장 적게 사용된 페이지들 중 결정된 페이지는 2차 메모리로 릴리즈되거나 또는 다시 기입될 수 있다. 추가로, 액세스가 요청된 데이터의 페이지는 2차 메모리로부터, 가장 적게 사용된 페이지들 중 릴리즈된 페이지의 위치의 메인 메모리로 전달될 수 있다. 대안적으로, 메인 메모리가 사용중이지 않은 페이지를 갖고 페이지 전달을 수신할 수 있으면, 페이지 폴트 핸들러 또는 제어기는 2차 메모리로부터 메인 메모리로의 페이지 데이터의 전달을 관리할 수 있다.

[0042] [0047] 특정한 실시예들에서, 페이지 폴트가 개시될 경우, 위에서 설명된 바와 같이, 데이터 전달이 페이지 폴트를 서비스하도록 관리되는 동안, 스레드 또는 구동중인 프로그램의 실행이 중지될 수 있다. 후속하여, 페이지 폴트가 서비스된 이후, 스레드는 메인 메모리 내의 페이지에 액세스하도록 릴리즈될 수 있다.

[0043] [0048] 특정한 구현들에서, 2차 메모리 내의 어느 페이지들이 "핫"하게 되는지, 즉 액세스의 빈도가 증가하는지를 결정할 뿐만 아니라 메인 메모리 DRAM 내의 어느 페이지들이 "콜드"하게 되는지, 즉 액세스의 빈도가 감소하는지를 결정하는 것이 유리할 수 있다. 다시 말하면, 메인 메모리보다 느릴 수 있는 2차 메모리에서 어느 페이지들이 더 빈번하게 액세스되는지 및 메인 메모리에서 어느 페이지들이 덜 빈번하게 액세스되는지가 결정된다. 메인 메모리에서 어느 페이지들이 덜 빈번하게 액세스되는지를 결정하기 위한 하나의 프로세스는, 예컨대 액세스 비트가 세팅되는 카운트에 기반하여 메인 메모리, 예컨대 DRAM에 대한 사용도 또는 액세스 통계들을 참조하여 위에서 설명된다. 시스템은, 메인 메모리에 대해 위에서 설명된 액세스 통계들에 기반하여, 메인 메모리로부터 2차 메모리로 데이터를 이동시킬 때 및 2차 메모리로부터 메인 메모리로 데이터를 이동시킬 때를 결정할 수 있다.

[0044] [0049] 부가적으로, 더 상세히 위에서 설명된 바와 같이, 냉각되거나(cool off) 또는 덜 빈번하게 액세스되는 페이지들은 PTE들을 모니터링함으로써 결정될 수 있다. 예컨대, 페이지에 대한 액세스-간 시간이 액세스-간 시간 임계치를 충족시킬 경우, 시스템은, 페이지와 연관된 PTE를 무효화시키고, 페이지와 연관된 임의의 엔트리들에 대한 TLB 슈트다운을 실행하며, 메인 메모리로부터 2차 메모리로 페이지를 전달함으로써 메인 메모리로부터 2차 메모리로의 페이지의 전달을 개시할 수 있다.

[0045] [0050] 특정한 구현들에 따르면, 캐시 코히런트 상호연결을 이용하여, DRAM 캐시 및 제2 티어의 메모리는 하드웨어에 의해 관리될 수 있으며, 이들은 코히런트 메모리의 소유자 및 코히런트 메모리의 사용자 둘 모두로서 작동할 수 있다. 다시 말하면, DRAM은 최적의 성능을 위해, 구성가능한 입도로 페이징하기 위하여 하드웨어에 의해 관리되는 캐시로서 작동한다. 최적의 성능을 위한 구성가능한 입도는 애플리케이션 지역성(locality) 및 제2 티어의 메모리의 성능에 의존할 수 있다.

[0046] [0051] 위에서 설명된 바와 같이, 맞춤화된 커널 드라이버는 하드웨어에 의해 소유된 어드레스 공간을 거대한 페이지들과만 맵핑시킬 수 있다. 이러한 방식으로, 시스템이 메모리의 이러한 구역에 액세스할 때마다, 시스템은 거대한 페이지들의 이점들, 이를테면 더 큰 TLB 도달범위(reach)로 인한 개선된 성능을 달성할 수 있다. 하드웨어는, 페이지가 메인 메모리에 존재하는지를 체크하기 위해 캐시 룩업 구조를 유지할 수 있다. 메모리 액세스가 호스트로부터 수신될 경우, 이러한 룩업 구조가 질의(query)될 수 있다. 페이지가 존재하면, 판독 또는 기입이 메인 메모리에서 직접 수행될 수 있다. 페이지가 존재하지 않으면, 데이터는 2차 메모리로부터 메인 메모리, 예컨대 DRAM으로 페칭(fetch)될 수 있다. 특정한 구현들에서, 성능 고려사항들에 대해, 캐시는 비동기적으로 퇴출(eviction)들을 수행할 수 있으며 – 예컨대, "콜드" 데이터는 인입 페이지들을 서비스하기 위해 최소수의 페이지들을 사용중이지 않게 유지하도록 백그라운드에서 2차 저장소로 다시 기입될 수 있다. 일반적으로, 이러한 프로세스는, 페이징을 위한 캐시에 캐시 코히런트 상호연결이 제공될 수 있도록 페이지 레벨에서 캐싱 메커니즘을 제공할 수 있다.

[0047] [0052] 특정한 구현들에서, 캐시 최적화가 적용될 수 있다. 예컨대, 페이지 폴트가 발생하고 페이지가 2차 저장소로부터 메인 메모리로 전달될 경우, 시스템은 메인 메모리에 페이지를 유지할지, 또는 페이지가 비-일시적이면, 페이지를 스트리밍하고 주어진 액세스에 대해 그 페이지를 한번 판독할지를 결정하거나, 또는 현재 페칭된 페이지가 액세스되는 것에 대한 응답으로 다음 페이지를 페칭하는 것으로 결정함으로써 프리-페칭(pre-fetching)을 수행할 수 있다. 특정한 구현들에서, 플래그들은 애플리케이션이 메모리를 얼마나 사용하고 있는지에 관해 세팅될 수 있으며, 프리-페칭에 대한 결정은 이들 플래그들에 기반하여 이루어질 수 있다.

[0048] [0053] 다수의 구현들이 설명되었다. 그럼에도, 본 개시내용의 사상 및 범위를 벗어나지 않으면서 다양한 수정들이 행해질 수 있다는 것이 이해될 것이다. 예컨대, 위에서 도시된 흐름들의 다양한 형태들은 재순서화된, 부가된, 또는 제거된 단계들과 함께 사용될 수 있다.

[0049] [0054] 본 명세서에서 설명된 본 발명 및 모든 기능 동작들의 구현들은, 디지털 전자 회로, 펌웨어, 또는 본 명세서에 개시된 구조들 및 그들의 구조적 등가물들을 포함하는 하드웨어, 또는 그들 중 하나 이상의 조합들에서 구현될 수 있다. 본 발명의 구현들은, 데이터 프로세싱 장치에 의한 실행을 위해, 또는 데이터 프로세싱 장치의 동작을 제어하기 위해 컴퓨터 판독가능 매체 상에서 인코딩된 하나 이상의 컴퓨터 프로그램 제품들, 즉 컴퓨터 프로그램 명령들의 하나 이상의 모듈들로서 구현될 수 있다. 컴퓨터 판독가능 매체는, 머신-판독가능 저장 디바이스, 머신-판독가능 저장 기판, 메모리 디바이스, 또는 그들 중 하나 이상의 조합일 수 있다. 용어 "데이터 프로세싱 장치"는 프로그래밍가능 프로세서, 컴퓨터, 또는 다수의 프로세서들 또는 컴퓨터들을 예로서 포함하는, 데이터를 프로세싱하기 위한 모든 장치, 디바이스들, 및 머신들을 포함한다. 장치는 하드웨어에 부가하여, 해당 컴퓨터 프로그램에 대한 실행 환경을 생성하는 코드, 예컨대 프로세서 펌웨어, 프로토콜 스택, 데이터베이스 관리 시스템, 운영 시스템, 또는 이들 중 하나 이상의 조합을 구성하는 코드를 포함할 수 있다.

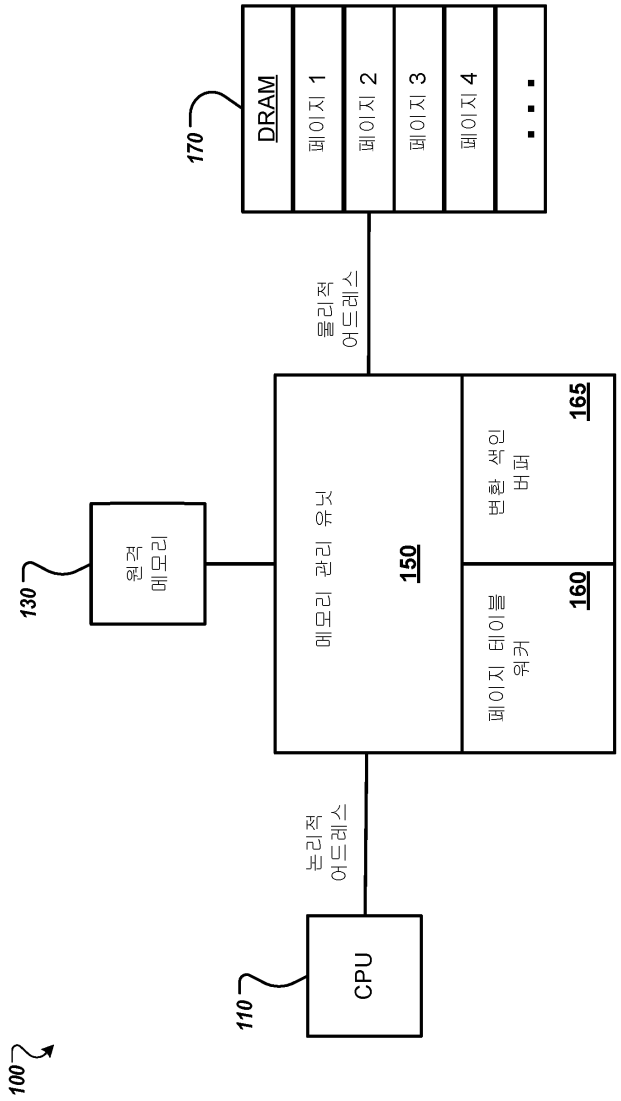
[0050] [0055] 본 개시내용이 많은 특정한 것들을 포함하지만, 이들은 본 발명의 또는 청구될 수 있는 것의 범위에 대한 제한들로서 해석되는 것이 아니라 오히려, 본 발명의 특정 구현들에 특정한 특징들의 설명들로서 해석되어야 한다. 별개의 구현들의 맥락에서 본 명세서에 설명된 특정한 특징들은 또한, 단일 구현의 조합으로 구현될 수 있다. 대조적으로, 단일 구현의 맥락에서 설명된 다양한 특징들은 또한, 다수의 구현들에서 별개로 또는 임의의 적절한 하위조합으로 구현될 수 있다. 또한, 특징들이 특정한 조합들에서 동작하는 것으로 위에서 설명되고 심지어 초기에는 그와 같이 청구될 수 있지만, 청구된 조합으로부터의 하나 이상의 특징들은 몇몇 경우들에서, 그 조합으로부터 삭제될 수 있으며, 청구된 조합은 하위조합 또는 하위조합의 변경으로 안내될 수 있다.

[0051] [0056] 유사하게, 동작들이 특정한 순서로 도면들에 도시되지만, 이것은, 바람직한 결과들을 달성하기 위해, 그러한 동작들이 도시된 특정한 순서 또는 순차적인 순서로 수행되거나, 모든 도시된 동작들이 수행될 것을 요구하는 것으로서 이해되지는 않아야 한다. 특정한 환경들에서, 멀티태스킹 및 병렬 프로세싱이 유리할 수도 있다. 또한, 위에서 설명된 구현들에서의 다양한 시스템 컴포넌트들의 분리는 모든 구현들에서 그러한 분리를 요구하는 것으로서 이해되지는 않아야 하며, 설명된 프로그램 컴포넌트들 및 시스템들이 일반적으로, 단일 소프트웨어 제품에 함께 통합되거나 다수의 소프트웨어 제품들로 패키징될 수 있음을 이해해야 한다.

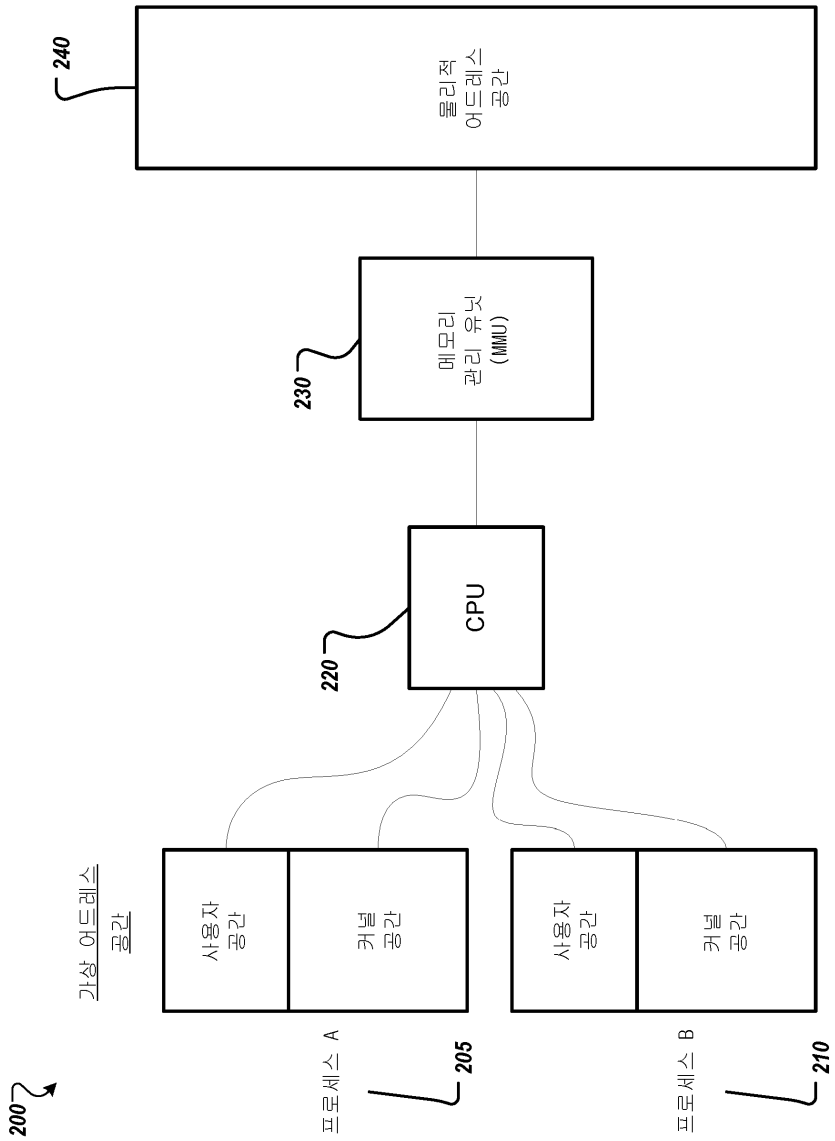
[0052] [0057] 따라서, 본 개시내용의 특정한 구현들이 설명되었다. 다른 구현들은 다음의 청구항들의 범위 내에 존재한다. 예컨대, 청구항들에서 인용된 동작들은, 상이한 순서로 수행될 수 있으며, 여전히 바람직한 결과들을 달성할 수 있다. 다수의 구현들이 설명되었다. 그럼에도, 본 개시내용의 사상 및 범위를 벗어나지 않으면서 다양한 수정들이 행해질 수 있다는 것이 이해될 것이다. 예컨대, 위에서 도시된 흐름들의 다양한 형태들은 재순서화된, 부가된, 또는 제거된 단계들과 함께 사용될 수 있다. 따라서, 다른 구현들은 다음의 청구항들의 범위 내에 존재한다.

도면

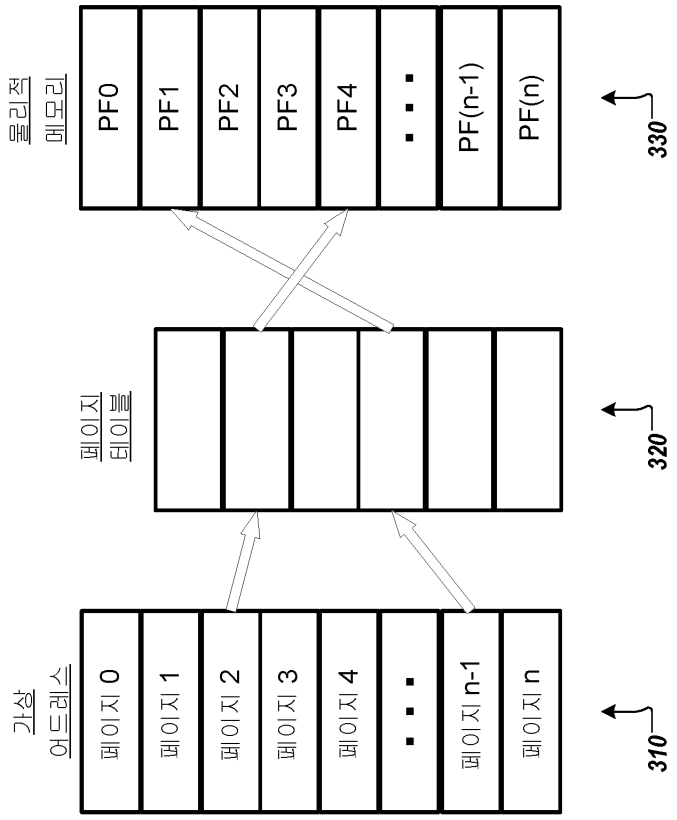
도면1


100
110
CPU
논리적 어드레스
130
원격 메모리
메모리 관리 유닛
150
변환 색인 버퍼
165
페이지 테이블 워커
160
물리적 어드레스
170
DRAM
페이지 1
페이지 2
페이지 3
페이지 4
. . .

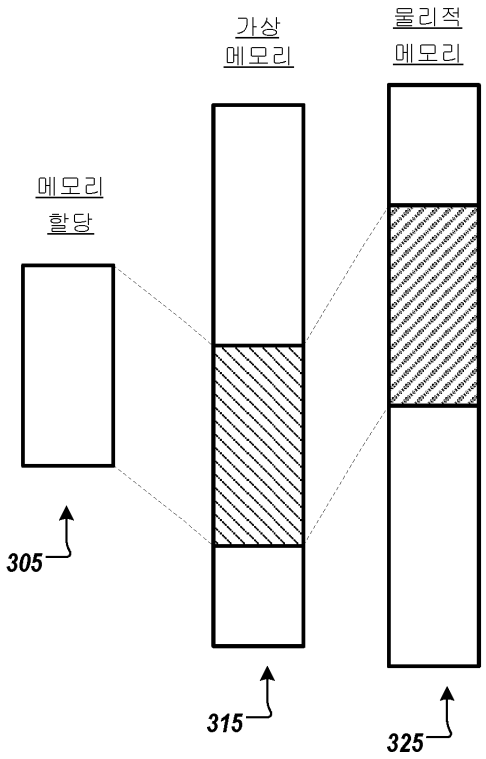
도면2


200
프로세스 A
205
프로세스 B
210
가상 어드레스 공간
사용자 공간
커널 공간
사용자 공간
커널 공간
220
CPU
230
메모리 관리 유닛 (MMU)
240
물리적 어드레스 공간

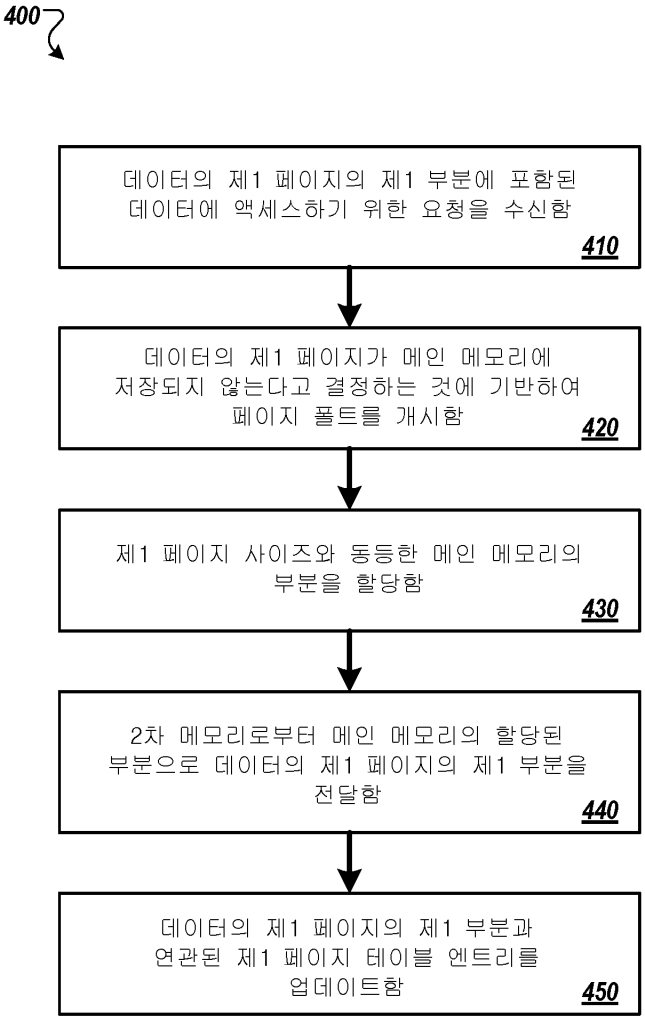
도면3a


물리적
메모리
PF0
PF1
PF2
PF3
PF4
PF(n-1)
PF(n)
330
페이지
테이블
320
가상
어드레스
페이지 0
페이지 1
페이지 2
페이지 3
페이지 4
페이지 n-1
페이지 n
310

도면3b


가상
메모리
물리적
메모리
메모리
할당
305
315
325

도면4


400
데이터의 제1 페이지의 제1 부분에 포함된 데이터에 액세스하기 위한 요청을 수신함
410
데이터의 제1 페이지가 메인 메모리에 저장되지 않는다고 결정하는 것에 기반하여 페이지 폴트를 개시함
420
제1 페이지 사이즈와 동등한 메인 메모리의 부분을 할당함
430
2차 메모리로부터 메인 메모리의 할당된 부분으로 데이터의 제1 페이지의 제1 부분을 전달함
440
데이터의 제1 페이지의 제1 부분과 연관된 제1 페이지 테이블 엔트리를 업데이트함
450