(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。Int. Cl.
G06F 12/00 (2006.01)
G06F 12/08 (2006.01)
G06F 12/12 (2006.01)

(11) 공개번호 10-2007-0024552
(43) 공개일자 2007년03월02일

(21) 출원번호 10-2006-7025218
(22) 출원일자 2006년11월30일
심사청구일자 없음
번역문 제출일자 2006년11월30일
(86) 국제출원번호 PCT/US2005/015493
국제출원일자 2005년05월04일
(87) 국제공개번호 WO 2006/007043
국제공개일자 2006년01월19일

(30) 우선권주장 10/881,508 2004년06월30일 미국(US)

(71) 출원인 인터내셔널 비지네스 머신즈 코포레이션
미국 10504 뉴욕주 아몬크 뉴오차드 로드

(72) 발명자 잰, 죠폰
미국, 뉴욕 10562, 오시닝, 반스 스트리트 213
패트네일, 프라탭, 챤드라
미국, 뉴욕 10562, 오시닝, 반스 스트리트 213
부르큘라, 르만쟈네야, 사르마
미국, 뉴욕 10520, 크로톤 온 허드슨, 아파트 유, 시딕 드라이브24

(74) 대리인 신영무
윤혜진

전체 청구항 수 : 총 14 항

(54) 컴퓨터 운영 시스템의 가상 메모리 서브시스템 자동 조정

(57) 요약

컴퓨터 시스템의 주메모리를 그 운용 시스템 인스턴스에서 실행하는 어플리케이션 사이에 효율적으로 분산하기 위한 방법, 정보 처리 시스템 및 컴퓨터 판독 가능 매체. 더욱 구체적으로, 가상 메모리 관리기의 페이지 교체 알고리즘에 의해 이용되는 임계치는 컴퓨터 시스템의 메모리 상의 부하에 응답하여 자동 조정된다. 이 임계치는 메모리 상의 부하의 함수로 변경되는 빈 메모리의 저 임계치이다. 예를 들어, 이 부하는 정의된 간격 동안 대기 큐에 부가된 스레드의 수를 그 기간 내의 클럭 틱의 수로 분할한 것으로 나타낼 수 있다. 이것을 스레드 대기 비율이라고 한다. 이 대기 비율은 타겟의 대기 비율과 비교되어 저 임계치가 변경되어야 하는지를 판단한다. 빈 메모리 공간이 저 임계치 이하로 떨어지면, 페이지 교체 데몬은 더욱 많은 메모리 공간을 이용 가능하게 만들기 위해 메모리를 페이지아웃하는 데에 이용된다.

대표도

도 2

특허청구의 범위

청구항 1.

컴퓨터 시스템의 메모리 이용 가능성을 관리하기 위한 방법에 있어서:

빈 메모리 공간의 저 임계치를 메모리 부하의 함수로 자동 변경시키는 단계; 및

빈 메모리 공간이 상기 저 임계치 이하일 때 더 많은 메모리 공간을 이용 가능하게 하는 단계

를 포함하는 방법.

청구항 2.

컴퓨터 시스템의 메모리 이용 가능성을 관리하기 위한 방법에 있어서:

스레드 대기 비율이 타겟의 스레드 대기 비율과 다를 때 저 임계치를 자동으로 변경시키는 단계 - 상기 스레드 대기 비율은 단위 시간 당 빈 메모리 대기 리스트에서 대기하는 스레드의 평균 개수임 - ; 및

빈 메모리 공간이 상기 저 임계치 이하일 때 더 많은 메모리 공간을 이용 가능하게 하는 단계

를 포함하는 방법.

청구항 3.

제2항에 있어서, 상기 저 임계치는 상기 스레드 대기 비율이 상기 타겟의 스레드 대기 비율 보다 더 커질 때 상승되는 방법.

청구항 4.

제2항에 있어서, 상기 저 임계치는 상기 스레드 대기 비율이 상기 타겟의 스레드 대기 비율 보다 더 작아질 때 감소되는 방법.

청구항 5.

제2항에 있어서, 고 임계치는 상기 스레드 대기 비율이 상기 타겟의 스레드 대기 비율 보다 더 커질 때 상승되고, 상기 고 임계치는 페이지 교체 데몬이 실행될 때 이용 가능하게 되는 메모리 공간양을 결정하는 데에 이용되는 방법.

청구항 6.

제2항에 있어서, 고 임계치는 상기 스레드 대기 비율이 상기 타겟의 스레드 대기 비율 보다 더 낮아질 때 감소되고, 상기 고 임계치는 페이지 교체 데몬이 실행될 때 이용 가능하게 되는 메모리 공간양을 결정하는 데에 이용되는 방법.

청구항 7.

제2항에 있어서, 상기 스레드 대기 비율은 상기 빈 메모리 대기 리스트에서 대기한 모든 스레드에 의해 소모되는 누적된 클럭 틱의 수를 계수하고 상기 누적된 수를 상기 페이지 교체 데몬의 두 연속되는 실행 사이의 클럭 틱의 총수로 나누어 연산될 수 있는 방법.

청구항 8.

제7항에 있어서, 상기 스레드는 현재 상기 빈 메모리 대기 리스트에 있는 제1 스레드와 상기 페이지 교체 데몬의 상기 두 연속되는 실행 중 처음 실행 이후 상기 빈 메모리 대기 리스트에 있었던 제2 스레드를 포함하고, 상기 제2 스레드는 상기 빈 메모리 대기 리스트에 더 이상 있지 않은 방법.

청구항 9.

제2항에 있어서, 상기 스레드 대기 비율은 상기 빈 메모리 대기 리스트 내의 상기 현재 스레드 수를 일정 수로 나누어 연산될 수 있는 방법.

청구항 10.

제9항에 있어서, 상기 일정 수는 정수 2인 방법.

청구항 11.

제2항에 있어서, 페이지 교체 데몬은 빈 메모리 공간이 상기 저 임계치 이하로 떨어질 때 실행되고, 상기 페이지 교체 데몬은 더 많은 메모리 공간을 이용 가능하게 하는 방법.

청구항 12.

제11항에 있어서, 상기 페이지 교체 데몬은 빈 메모리 프레임의 수가 저 임계치 이하로 떨어지면 실행되고, 상기 페이지 교체 데몬은 빈 프레임의 수가 상기 고 임계치에 이르도록 다수의 프레임을 비우는 것을 포함하는 방법.

청구항 13.

디지털 처리 장치에 의해 판독 가능하며 상기 저장 장치에서 실제 구현되며 컴퓨터 시스템의 메모리 이용 가능성을 관리하는 방법을 실행하도록 상기 처리 장치에 의해 실행될 수 있는 명령의 프로그램을 갖는 프로그램 저장 장치에 있어서, 상기 방법은:

빈 메모리 공간의 저 임계치를 메모리 부하의 함수로 자동 변경시키는 단계; 및

빈 메모리 공간이 상기 저 임계치 이하일 때 더 많은 메모리 공간을 이용 가능하게 하는 단계

를 포함하는 프로그램 저장 장치.

청구항 14.

컴퓨터 시스템의 메모리 이용 가능성을 관리하기 위한 장치에 있어서:

빈 메모리 공간의 저 임계치를 메모리 부하의 함수로 자동 변경시키기 위한 수단; 및

빈 메모리 공간이 상기 저 임계치 이하일 때 더 많은 메모리 공간을 이용 가능하게 하기 위한 수단

을 포함하는 장치.

명세서

기술분야

본 발명은 컴퓨터 운영 시스템의 가상 메모리 관리 구성 요소에 관한 것으로, 더욱 특히 운영 시스템의 가상 메모리 관리기의 페이지 교체 알고리즘에 의해 이용되는 임계치의 조정에 관한 것이다.

배경기술

기기 상에서 실행되는 운영 시스템 (OS)의 가상 메모리 관리기 (VMM) 구성 요소는 그 OS 인스턴스에서 실행되는 어플리케이션 중에서 기기의 주메모리를 효율적으로 분산시키는 작용을 한다. VMM의 주요 작용은 디스크 상의 페이징 공간으로 이용되지 않은 주메모리 블럭 ("프레임"이나 "페이지 프레임"으로 불림)의 컨텐츠를 페이지아웃하고, 주메모리를 필요로 하는 다른 어플리케이션에 이 프레임을 재할당하는 것이다. 이것은 통상 "페이지 교체 데몬(daemon)" (또한 대부분의 UNIX 운영 시스템에서 "LRU 데몬"으로도 불림)으로 불리는 데몬 프로세스의 도움으로 이루어진다.

프레임을 비우는 프로세스 (즉, 디스크가 빈 프레임이 되도록 그 컨텐츠를 모두 이동시키는 동작)는 요청 어플리케이션 (빈 프레임의 소모자)에게 빈 프레임을 할당하는 프로세스 보다 더 길게 걸리기 때문에, 페이지 교체 데몬은 OS에서 부가의 빈 프레임의 필요성을 미리 예상하고, 통상 OS의 빈 프레임의 수가 제로가 되기 전에 보통 프레임을 페이지아웃하기 시작한다. VMM은 두 조정 가능한 파라미터 min_free와 max_free를 이용하여, 페이지 교체 데몬을 언제 시작할지와 각 실행시 얼마나 많은 페이지가 비워져야 하는지를 결정할 수 있다. 페이지 교체 데몬은 빈 프레임의 수가 min_free 이하가 되자 마자 시작되고, 각 실행시 충분한 페이지를 비워서 빈 프레임의 수가 종료시 max_free가 되게 한다.

현재 이들 파라미터는 OS에서 실행되는 어플리케이션의 조건에 만족하도록 VMM의 성능을 조정하기 위해 시스템 관리자에 의해 입력되어야 한다. 이 조정은 사람의 수동 입력을 필요로 하기 때문에, 이들 파라미터는 거의 조정되고 있지 않으며, 그 결과 VMM 및 이에 따라 OS의 성능은 준최적이 되게 된다. 이런 조정의 결여로 IT 기구에 대해 더 많은 비용을 지불하게 만든다.

발명의 상세한 설명

본 발명의 목적은 시스템 관리자에 의한 VMM의 수동적 조정의 필요성을 없애어 시스템의 성능을 개선하는 것이다. 이 발명의 중요한 장점은 작업 부하의 변경에 대한 응답성/적응성이 더욱 좋아진다는 것이다. 더욱 구체적으로 설명하면, 본 발명은 VMM 시스템 파라미터의 조정을, 이들의 값을 OS에서의 변경 메모리 부하에 응답하여 자동적으로 변하게 함으로써 자동적으로 만든다.

본 발명은 OS의 메모리 부하에 응답하여 저 임계치로 알려진 파라미터를 자동으로 변경시켜 OS에서의 메모리 이용성을 개선하기 위한 방법을 제공한다. 현재 빈 메모리 공간이 저 임계치 이하가 될 때 더욱 많은 빈 메모리 공간이 형성된다.

본 발명의 더욱 바람직한 실시예는 빈 메모리 공간의 저 임계치를 초기 값으로 설정하고 현재의 "스레드 대기 비율 (thread wait rate)"이 타겟의 "스레드 대기 비율"과 다를 때 이 저 임계치를 자동으로 변경시켜 OS의 메모리 관리기를 자동으로 조정하는 방법을 제공한다. 이 때 "스레드 대기 비율"은 특정한 시간에 걸쳐 단위 시간 당 대기하는 스레드의 개수이다. 메모리 관리기는 빈 메모리 공간이 저 임계치 이하로 떨어지면 더욱 많은 메모리 공간을 이용 가능하게 하기 위한 동작을 초기화하게 된다.

실시예

도 1을 참조하여, 통상의 운영 시스템 커널(10)은 몇개의 구성 요소 - 가상 메모리 관리기 (VMM; 20), 프로세서 관리기 (30), 파일 시스템(40), 네트워킹 서브시스템(50) 등으로 이루어진다. 파일 시스템(40)은 하드 디스크 드라이브에 구조적 액세스를 제공한다. 프로세스 관리기는 프로세서에 대해 프로세스 및 스레드를 스케줄하는 한편, 네트워킹 서브시스템은 어플리케이션이 다른 기기나 컴퓨터와 통신할 수 있게 한다. 본 발명은 운영 시스템(OS)의 가상 메모리 관리기 구성 요소에 관한 것이고, 그 외 구성 요소는 본 기술에 잘 알려져 있기 때문에 이 출원에서는 설명하지 않았다. 도 2는 시스템 메모리를 관리하기 위해 VMM(20)이 보유한 데이터 구조를 나타낸다. VMM(20)은 실제 메모리의 빈 페이지 프레임(22)을 추적하도록 하나 이상의 빈 리스트(21)을 보유한다. VMM(20)은 또한 변수 nfree(63)에 빈 프레임의 총수를 보유한다. 빈 페이지의 요청이 들어오면, VMM(20)은 nfree>0인지 체크하고, 그렇다면, 빈 리스트(21)를 검색하여 빈 페이지를 불러낸다. nfree(63)의 값이 저 임계치 min_free(61) 이하가 되면, VMM(20)은 페이지 교체 데몬(25) (또한 LRU 데몬으로 알려짐)을 불러내어 사용중 페이지 프레임(23)의 컨텐츠를 디스크(51)로 내보내어 빈 페이지 프레임(22)을 형성하도록 페이지아웃을 초기화한다. 여기에 기재된 예시에서, 이 페이지아웃은 페이지 교체 데몬으로 불리는 개별의 프로세스로 구현된다. 페이지 교체 데몬은 그 실행 종료시 nfree가 max_free보다 크거나 동일하도록 충분한 수의 페이지 프레임을 페이지아웃하게 된다. 빈 페이지 프레임에 대해 이행되지 않은 요청들 모두는 도 2에서 나타낸 바와 같이 대기 리스트(32)에 큐된다. 요청 스레드(33)은 빈 페이지 프레임이 나중에 이용 가능하게 될 때 되살려진다.

페이지 교체 데몬에 의해 이용되는 고위 알고리즘이 도 3에 나타나 있다. 박스(10)은 nfree<min_free일 때 페이지 교체 데몬을 불러내는 것이다. 102에서, 페이지 교체 데몬은 max_free에서 nfree(63)를 감하여 페이지아웃될 페이지의 수를 연산한다. 103에서, 내보낼 적당한 후보를 찾기 위해 사용중 페이지 프레임 리스트(23)을 조사한다. 페이지 교체 데몬이 내보낼 후보 페이지를 선택하기 위해 메모리를 조사하기 시작하면, 어떤 특정 페이지가 페이지아웃될지를 결정하는 데에는 몇가지 가능한 폴리시(policy)가 있다. 본 발명은 내보낼 후보 페이지를 선택하기 위해 이용되는 특정 폴리시에 따라 좌우된다. 102에서 연산된 페이지수를 내보낸 후, 페이지 교체 데몬은 104에서 nfree가 max_free 보다 여전히 작은지에 대해 다시 검사한다. 이것은 페이지 프레임이 비워진 바로 후에 소모되고 있으면 발생할 수 있다. 104에서 술어가 예라고 평가되면, 분기(106)로 가서 생겨 단계(102)를 재시작하게 된다. 아니면, 분기(105)로 가서 107에서 페이지 교체 데몬이 다시 슬립 상태가 된다.

상기 기재에 의하면, VMM의 페이지 교체 데몬의 목표는 대기 리스트(32) 상에 너무 많은 빈 페이지 요청자(33)을 갖는 비용과 너무 많은 사용중 페이지(24)를 너무 이르게 내보내는 비용을 균형 맞추는 것이다. 본 발명은 이 문제를 파라미터 min_free(61)과 max_free(62)에 대한 최적의 값을 결정하고 진행에 따른 파라미터를 빈 프레임 변경에 대한 대기자의 수로서 조정하는 기구를 제공함으로써 해결한다.

도 4는 본 발명을 구현하는 데에 필요한 바람직한 실시예의 데이터 구조를 나타낸다. 우리는 요청자의 수와 각 요청자가 빈 프레임을 대기하는 대기 리스트에서 소모하는 시간을 연속하여 모니터할 필요가 있다. 이는 변수 thrd_wait(35)를 보유하는 것으로 행해진다. thrd_wait의 값은 다음과 같이 갱신된다. 스레드(33)가 대기 리스트(32)로 큐될 때 마다, 운영 시스템은 스레드가 스레드당 변수 waitlist_enque_time(36)에 큐되는 시간을 기록한다. 빈 페이지가 이용 가능하여 스레드가 마침내 디큐(dequeu)될 때, OS는 현재 시간에서 waitlist_enque_time(36)을 감하여 스레드가 대기 리스트에서 소모한 총 시간을 구한다. OS는 다음에 이 총시간을 클럭 틱 (clock tic)으로 전환하여, 이를 thrd_wait(35)의 카운터에 부가하고, 이는 모든 스레드가 함께 대기 리스트에서 소모한 클럭 틱의 총 시간을 포함한 것이다. 이 카운터는 도 5에서 설명하는 바와 같이 페이지 교체 데몬에 의해 리세트되게 된다. thrd_wait(35) 카운터는 페이지 교체 데몬의 두 실행 사이의 몇 지점에서 대기 리스트에 큐되거나 이로부터 디큐되는 모든 스레드에 대한 총 대기 시간을 포함한다. 페이지 교체 데몬은 다른 변수 thrd_wait_rate(37)을 보유하며, 이는 클럭 틱 당 대기하는 평균 스레드수가 된다. 이것은 clock_tic 당 카운트의 단위로 보유되므로, 이 값은 여러 CPU 주파수를 갖는 시스템 마다 표준화되게 된다. thrd_wati_rate(37)의 값은 페이지 교체 데몬이 시작될 때 마다 모든 스레드가 마지막 실행 이후 대기한 누적 총 시간과 두 실행 간에 포함되는 클럭 틱수의 비율로서 연산된다. 페이지 교체 데모은 또한 상술된 두 실행 사이에 포함되는 클럭 틱의 수를 연산하는 데에 이용되는 다른 변수 strt_time(39)를 보유한다. 마지막으로, 도 4에서 주어진 thrd_wait_rate_tgt(38) 값이 페이지 교체 데몬에 의해 이용되어 min_free와 max_free가 얼마나 변경되어야 하는지 연산한다.

도 5에 나타낸 플로우챠트는 페이지 교체 데몬의 새로운 방식을 요약한 것이다. 박스(201)는 nfree<min_free일 때 페이지 교체 데몬을 불러오는 것이다. 202에서, 페이지 교체 데몬은 마지막 실행 이후 경과 시간과 마지막 실행 이후 모든 스레드가 대기 리스트에서 소모한 클럭 틱의 총시간을 연산한다. 경과 시간은 현재 시스템 시간을 판독하고 이전에 기록된 strt_time(38) 값을 감하는 것으로 연산된다. 모든 스레드가 마지막 실행 이후 대기한 총 시간은 두 부분으로 이루어진다. 제1 부분은 현재 대기 리스트에 있지 않는 스레드의 총 대기 시간이다. 이는 OS에 의해 thrd_wait 카운터에 보유된다. 제2 부분은 대기 리스트에 여전히 있는 스레드가 큐에서 소모한 총 시간이다. 페이지 교체 데몬(25)은 이 제2 부분을, 대기 리

스트(32) 전체에 걸쳐, 현재 시간에서 각 스레드의 waitlist_queue_time(36)을 감하고, 모든 대기 시간을 더하는 것으로 연산한다. 대기 리스트 내 또는 외의 모든 스레드의 총 대기 시간은 상기 값을 thrd_wait 카운터(35)에 더하는 것으로 연산된다.

모든 스레드의 총 대기 시간을 연산하는 다른 기구로는 대기 리스트의 스레드의 수를 계수하고 이를 thrd_wait 카운터에 더하여 매 클럭 틱마다 한번 대기리스트를 폴(poll)하는 것이 있다. 이 경우, 페이지 교체 데몬은 각 실행 시작시 대기 리스트 전체를 볼 필요가 없다. 단점은 매 클럭 틱 마다 부가의 작업을 행해야 하는데, 이는 너무 많은 오버헤드가 될 수 있다.

페이지 교체 데몬이 202에서 경과 시간과 총 스레드 대기 시간을 연산하므로, 시간 스탬프 값 strt_time과 스레드 당 waitlist_enque_time을 이들 변수를 판독한 바로 후의 현재 시스템 시간으로 리세트한다. 또한 thrd_wait_time을 0으로 리세트하여 이 카운터가 지금부터 대기 리스트 전체에 걸친 모든 스레드에 대한 대기 시간을 포함하게 된다. 203에서 페이지 교체 데몬은 총 스레드 대기 시간을 경과 시간으로 나누어 thrd_wait_rate(36)을 연산한다. 204에서는 아래와 같이, min_free(61)와 max_free(62) 값을 상기 연산된 thrd_wait_rate와 미리 설정된 타겟 값 thrd_wait_rate_tgt의 차이에 기초하여 재연산한다.

원하는 min_free 값은 thrd_rate가 thrd_wait_rate_tgt 보다 더 큰 경우 증가하고 thrd_rate가 thrd_wait_rate_tgt 보다 더 작은 경우 감소한다. 이 실시예에서, 원하는 min_free는 아래와 같이 연산된다.

desired min_free=min_free*thrd_wait_rate/thrd_wait_rate_tgt

min_free 파라미터는 원하는 min_free 값과 현재 값의 평균으로 갱신된다. 이 평균화는 작업 부하의 상승으로 인한 변동에 대해 약간의 감쇄를 제공한다. 심하나 변동을 피하기 위해서, 페이지 교체 데몬에 의해 갱신될 수 있는 min_free 값에 상한치를 둘 수 있다.

max_free 파라미터는 min_free의 갱신 이전과 같이 동일한 max_free와 min_free 간에 동일한 차이를 유지하도록 갱신된다.

예: thrd_wait_rate_tgt가 1로 설정되고, thrd_wait_rate는 1.5로 연산되고, min_free와 max_free는 각각 120과 128이라고 가정한다. 원하는 min_free는 120*1.5/1=180이 된다. 페이지 교체 데몬은 min_free를 원하는 min_free와 현재 min_free의 평균값으로 변경하게 되고, 이는 120+ 180/2=150이다. max_free의 새로운 값은 150+ (128-120)=158이 된다.

몇 다른 기구들은 상기 주어진 더욱 간단한 선형 근사화 대신에, thrd_wait_rate의 편차로부터 원하는 min_free 값을 연산하는 데에 이용될 수 있다. 원하는 min_free 값을 연산하는 데에 이용되는 기구는 thrd_wait_rate>thrd_wait_rate_tgt인 경우 증가되고, thrd_wait_rate<thrd_wait_rate_tgt인 경우 감소된다는 일반 원리를 고수한다.

min_free와 max_free 값의 재연산 후에, 페이지 교체 데몬의 나머지 단계 - 205, 206, 207, 208, 209 및 120- 는 도 3의 단계 - 101, 103, 104, 105, 106 및 107- 과 각각 유사하다. 더욱 구체적으로, 205에서, 페이지 교체 데몬은 nfree(63)을 max_free(61)으로부터 감하여 페이지아웃되는 페이지수를 연산한다. 206에서, 사용중 페이지 프레임을 조사하여 내보낼 적당한 후보를 구한다. 페이지 교체 데몬인 내보낼 후보 페이지를 선택하기 위해 메모리를 조사하기 시작하게 되면, 어느 특정 페이지가 페이지아웃 될지를 결정하는 몇 가지 가능한 폴리시가 있다. 본 발명은 내보낼 후보 페이지를 선택하는 데에 이용되는 특정 폴리시에 좌우되지 않는다. 205에서 연산된 바와 같이 페이지 수를 내보낸 후, 페이지 교체 데몬은 207에서 nfree가 max_free 보다 작은지를 다시 검사한다. 이는 페이지 프레임이 비어진 바로 후에 소모되고 있는 경우 발생할 수 있다. 207에서의 술어가 예이면, 분기(209)로 가서 단계 205가 다시 시작된다. 아니면 분기(208)로 가서 페이지 교체 데몬은 201에서 다시 슬립 상태가 된다.

도 6은 타임라인을 이용하여, thrd_wait_rate가 연산되는 방법을 나타낸다. 화살표(381)는 벽시계 시간의 기준선으로 작용한다. 381에서, 3 개의 시간 스탬프 ts_1, ts_2 및 ts_3은 페이지 교체 데몬의 3개의 실행의 시작을 나타낸다. 시간 축에 대해 나타낸 이중 머리 화살표는 페이지 교체 데몬(25)의 각 실행이 완성하는 데에 드는 시간을 나타낸다. 도면으로부터 페이지 교체 데몬의 각 실행은 완성하는 데에 다른 시간이 소요되는 것을 알 수 있다. 또한, 페이지 교체 데몬의 2 연속 인스턴스 사이의 경과 시간은 고정되어 있지 않다. 각 시간 스탬프에서의 TW_n 값은 페이지 교체 데몬의 마지막 실행 이후 모든 스레드가 대기 큐에서 소모하는 총 시간을 나타낸다. TW_n 값은 이전 단락에서 기재한 바와 같이, 대기 리스트의 각 스레드의 thrd_wati 카운터 및 watilist_enque_time을 이용하여 연산된다.

예시의 실시예에서, 여러 파라미터는 아래와 같이 시스템 초기화 시간에 초기화되게 된다:

·min_free(61) 및 max_free(62)는 디폴트 값으로 설정됨.

·thrd_wati 카운터(35) 및 thrd_wait_rate(37)는 0으로 초기화됨

·thrd_wait_rate_tgt(38)은 특정 값으로 초기화됨.

·strt_time(39)는 현재 시간으로 초기화됨.

상기 예시의 실시예에서 기재된 바와 같이 정밀한 thrd_wait_rate 값을 보유하지 않고도 본 발명을 구현할 수 있다는 것이 이해될 것이다. 이 출원에서 기재된 바람직한 실시예에서, 페이지 교체 데몬은 불러낼 때 마다 전체 대기 리스트에 걸쳐 정밀한 값 thrd_wait_rate를 연산한다. thrd_wait_rate의 대충 예상치를 연산하여 본 발명을 구현할 수 있으며, 이는 메모리 이용 가능성에 미치는 영향을 상당히 저하시키지 않고 구현시의 복잡성을 감소시킬 수 있다. 다음의 단락에서는 thrd_wait_rate를 연산하는 두 방법을 설명한다.

1) 시스템은 thrd_wait 카운터에 부가하여 2개의 변수 nthrds_waited 및 nthrds_waiting를 보유할 수 있다. nthrds_waited는 thrd_wait의 값에 기여한 스레드의 수를 포함한다. nthds_waiting은 현재 대기 리스트의 드레스의 수를 포함한다. 이들 두 변수는 스레드가 대기 리스트를 나갈 때 마다 갱신되고; nthrds_waiting은 또한 스레드가 대기 리스트상에 큐될 때 갱신된다. 이들 변수가 주어지면 페이지 교체 데몬은 다음과 같이 thrd_wait_rate를 연산할 수 있다.

thrd_wait_rate=(thrd_wait+ ((thrd_wait)/nthrds_waited)*nthrds_waiting))/(current time_str_time)

이를 이용하여 각 스레드에 대한 waitlist_enque_time을 보유할 필요를 없애준다.

2) 대기 리스트에서 벗어난 스레드를 무시하는 것으로 예측을 더욱 간략화할 수 있다. 현재 대기 리스트에 있지 않은 스레드를 무시하고 대기 리스트 상의 스레드가 균일한 시간격에서 큐된다고 가정하면, thrd_wait_rate는 단순히 nthrds_waiting/2으로 연산될 수 있다. 이는 다음과 같이 유도된다.

대기 리스트 상의 제1 스레드가 시간 T1에서 큐되고, 현재 시간은 T2라고 가정한다. 스레드가 균일한 시간격으로 대기 리스트 상에 큐되었다고 가정했기 때문에, 각 스레드는 평균 (T2-T1)/2의 시간 대기하게 된다.

현재 대기 리스트의 모든 스레드의 총 대기 시간=(nthrds_waiting*(T2-T1)/2)

경과 시간=(T2-T1)

thrd_wait_rate=(nthrds_waiting*(T2-T1)/2)/(T2-T1)=nthrds_waiting/2

도 7은 본 발명의 일 실시예를 구현하기에 유용한 정보 처리 시스템을 나타내는 고위 블럭도이다. 컴퓨터 시스템은 프로세서(704)와 같은 하나 이상의 프로세서를 포함한다. 프로세서(704)는 통신 인프라구조(702) (예를 들어, 통신 버스, 크로스오버 바, 또는 네트워크)에 연결된다. 여러 소프트웨어 실시예가 이 예시의 컴퓨터 시스템에 관하여 기재된다. 당업자에게는 이 설명을 읽고 나서 본 발명을 다른 컴퓨터 시스템 및/또는 컴퓨터 아키텍처를 이용하여 구현하는 방법이 명백하게 될 것이다.

컴퓨터 시스템은 디스플레이 유닛(710) 상에서의 표시를 위해 통신 인프라구조(702)로부터 (또는 도시하지 않은 프레임 버퍼로부터) 그래픽, 텍스트 및 그 외 데이터를 전달하는 디스플레이 인터페이스(708)를 포함한다. 컴퓨터 시스템은 또한 메인 메모리(706), 바람직하게 랜던 액세스 메모리(RAM)을 포함하고, 또한 보조 메모리(712)를 포함한다. 보조 메모리(712)는 예를 들어, 하드 디스크 드라이브(714) 및/또는 플로피 디스크 드라이브, 자기 테이프 드라이브, 광 디스크 드라이브 등을 나타내는 착탈 가능 저장 드라이브(716)을 포함한다. 착탈 가능 저장 드라이브(716)은 당업자에게는 잘 알려진 방식으로 착탈 가능 저장 유닛(718)으로부터 판독하거나 여기에 기록한다. 착탈 가능 저장 유닛(718)은 플로피 디스크, 컴팩트 디스크, 자기 테이프, 광 디스크 등을 나타내며, 이는 착탈 가능 저장 드라이브(716)에 의해 판독되거나 기록된다. 이해되는 바와 같이, 착탈 가능 저장 유닛(718)은 내부에 컴퓨터 소프트웨어 및/또는 데이터가 저장된 컴퓨터 판독 가능 매체를 포함한다.

다른 실시예에서, 보조 메모리(712)는 컴퓨터 프로그램이나 그 외 명령이 컴퓨터 시스템에 로딩되게 하기 위한 그 외 유사한 수단을 포함한다. 이런 수단은 예를 들어, 착탈 가능 저장 유닛(722) 및 인터페이스(720)을 포함한다. 이의 예는 프로그램 카트리지와 카트리지 인터페이스 (비디오 게임 장치에서 볼 수 있음), 착탈 가능 메모리 칩 (EPROM 또는 PROM 등)과 관련 소켓, 및 그 외 소프트웨어와 데이터가 착탈 가능 저장 유닛(722)으로부터 컴퓨터 시스템으로 전달되게 하는 착탈 가능한 저장 유닛(722)과 인터페이스(720)를 포함한다.

컴퓨터 시스템은 또한 통신 인터페이스(724)를 포함한다. 통신 인터페이스(724)는 소프트웨어와 데이터가 컴퓨터 시스템과 외부 장치 간에 전달되게 한다. 통신 인터페이스(724)의 예로는 모뎀, 네트워크 인터페이스 (이더넷 카드 등), 통신 포트, PCMCIA 슬롯과 카드 등이 있다. 통신 인터페이스(724)를 통해 전달되는 소프트웨어 및 데이터는 예를 들어, 전자, 전자기, 광학 신호나, 그 외 통신 인터페이스(724)에 의해 수신될 수 있는 신호의 유형일 수 있다. 이들 신호는 통신 경로 (즉, 채널; 726)를 통해 통신 인터페이스(724)에 제공된다. 이 채널(726)은 신호를 이송하며 와이어나 케이블, 광 섬유, 전화선, 셀룰러폰 링크, RF 링크 및/또는 그 외 통신 채널을 이용하여 구현될 수 있다.

이 서류에서, 용어 "컴퓨터 프로그램 매체", "컴퓨터 이용 가능 매체" 및 "컴퓨터 판독 가능 매체"는 일반적으로 주메모리(706)와 보조 메모리(712) 등의 매체, 착탈 가능 저장 드라이브(716), 하드 디스크 드라이브(714)에 설치된 하드 디스크, 및 신호를 말하는 데에 이용된다. 이들 컴퓨터 프로그램 제품은 컴퓨터 시스템에 소프트웨어를 제공하기 위한 수단이다. 컴퓨터 판독 가능 매체는 컴퓨터 시스템이 데이터, 명령, 메시지나 메시지 패킷, 및 그 외 컴퓨터 판독 가능 정보를 컴퓨터 판독 가능 매체로부터 판독할 수 있게 한다. 컴퓨터 판독 가능 매체는 플로피 디스크, ROM, 플래시 메모리, 디스크 드라이브 메모리, CD-ROM 및 그 외 영구 저장소 등의 비휘발성 메모리를 포함한다. 이것은 예를 들어, 컴퓨터 시스템 사이에 데이터와 컴퓨터 명령 등의 정보를 전송하는 데에 유용하다. 더욱, 컴퓨터 판독 가능 매체는 컴퓨터가 이 컴퓨터 판독 가능 정보를 판독할 수 있게 하는 유선망이나 무선망을 포함하여, 네트워크 링크 및/또는 네트워크 인터페이스 등의 임시 상태 매체 내의 컴퓨터 판독 가능 정보를 포함할 수 있다.

컴퓨터 프로그램 (또한 컴퓨터 제어 로직으로 불림)은 주메모리(706) 및/또는 보조 메모리(712)에 저장된다. 컴퓨터 프로그램은 또한 통신 인터페이스(724)를 통해 수신된다. 이 컴퓨터 프로그램은 실행시, 컴퓨터 시스템이 본 발명의 특성을 여기 기재된 바와 같이 실행할 수 있게 한다. 특히, 컴퓨터 프로그램은 실행시, 프로세서(704)가 컴퓨터 시스템의 특성을 실행할 수 있게 한다. 따라서, 이런 컴퓨터 프로그램은 컴퓨터 시스템의 제어기를 나타낸다.

본 발명의 특정 실시예가 기재되었지만, 당업자라면 본 발명의 정신 및 영역에서 벗어나지 않고 특정 실시예에 변경을 행할 수 있다는 것이 이해될 것이다. 따라서, 본 발명의 영역은 특정 실시예에만 제한되는 것이 아니다.

더욱, 첨부한 청구범위는 본 발명의 정신 내의 모든 응용, 수정 및 실시예를 포괄하는 것이다.

도면의 간단한 설명

본 발명의 요지는 상세 설명 이후 청구범위에서 지적되어 명백하게 청구되고 있다. 본 발명의 상술 및 그 외 특성과 장점들은 첨부한 도면을 참조한 다음 상세 설명으로부터 명백하게 될 것이다.

도 1은 통상의 컴퓨터 운영 시스템의 주요 서브시스템을 나타낸다.

도 2는 현재 페이지 교체 알고리즘을 구현하는 데에 이용되는 데이터 구조를 나타낸다.

도 3은 현재 페이지 교체 알고리즘의 플로우 챠트를 나타낸다.

도 4는 페이지 교체 데몬의 동적인 자동 조정을 구현하기 위해 바람직한 실시예에서 이용되는 데이터 구조를 나타낸다.

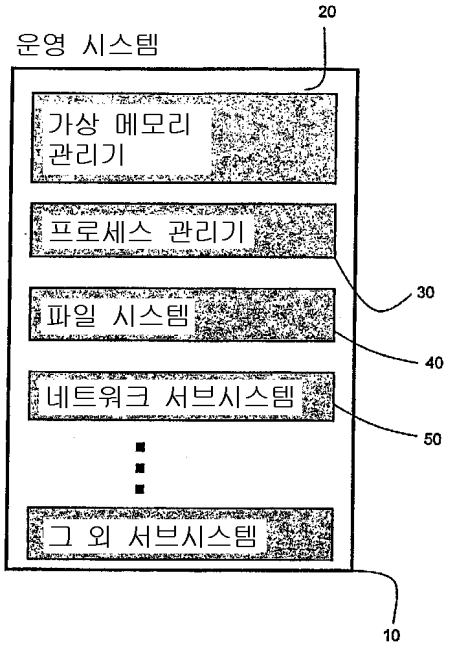
도 5는 본 발명에서의 새로운 페이지 교체 알고리즘의 플로우챠트를 나타낸다.

도 6은 min_free와 max_free가 LRU 데몬에 의해 시간에 따라 어떻게 변경되는지를 나타낸다.

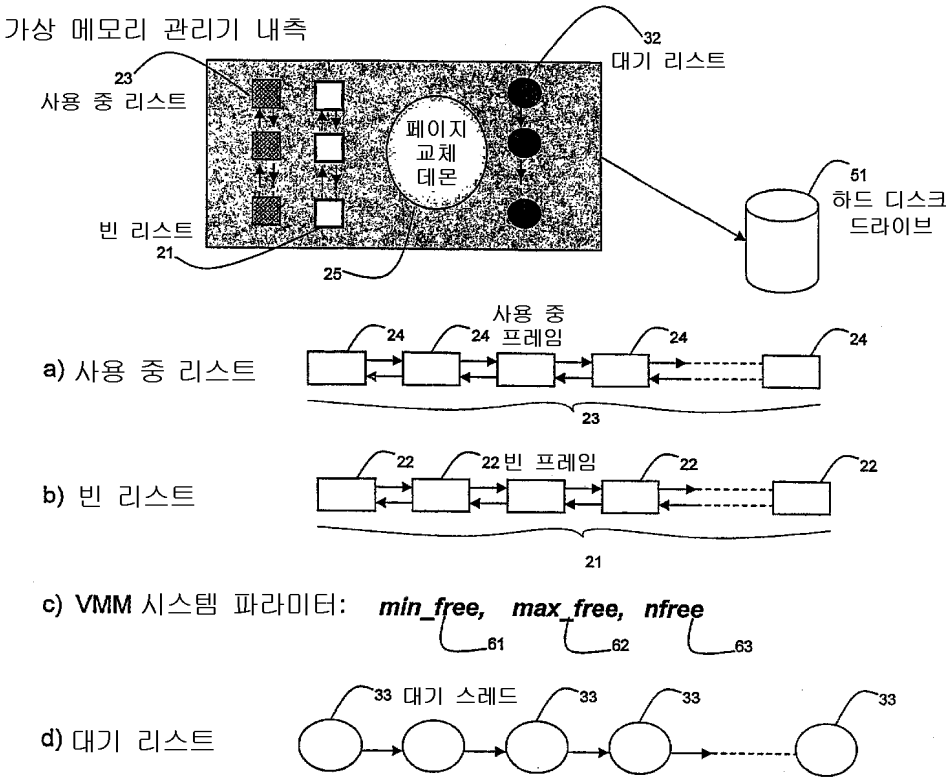
도 7은 본 발명의 하나 이상의 구성 요소/방법들이 구현되는 것에 따라서 컴퓨팅 시스템의 하드웨어 구현의 설명도이다.

도면

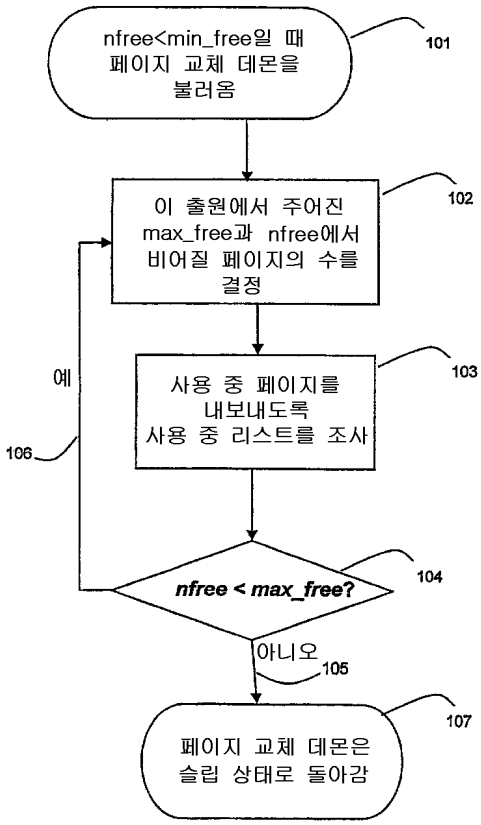
도면1


운영 시스템
20
가상 메모리 관리기
프로세스 관리기
30
파일 시스템
40
네트워크 서브시스템
50
그 외 서브시스템
10

도면2


가상 메모리 관리기 내측
23
사용 중 리스트
32
대기 리스트
페이지 교체 데몬
51
하드 디스크 드라이브
빈 리스트
21
25
사용 중 프레임
a) 사용 중 리스트
24
23
b) 빈 리스트
22
22 빈 프레임
21
c) VMM 시스템 파라미터: min_free, max_free, nfree
61
62
63
d) 대기 리스트
33 대기 스레드
33

도면3



도면4

a) *thrd_wait* 35

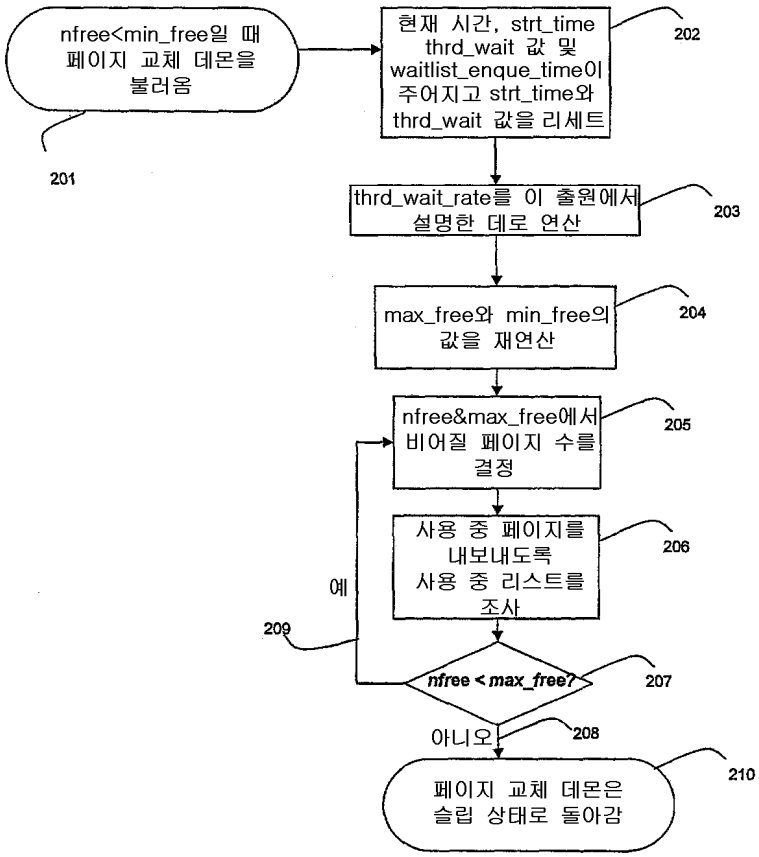
b) 각 스레드에 대한 *waitlist_enque_time* 36

c) *thrd_wait_rate* 37

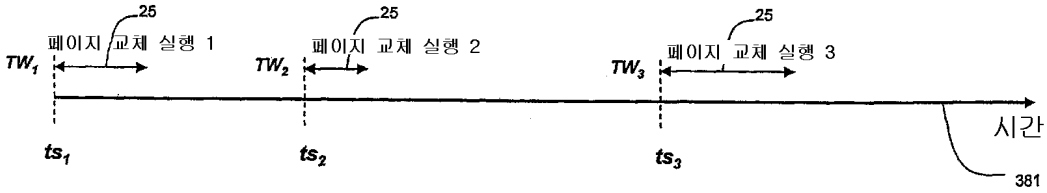
d) *thrd_wait_rate_tgt* 38

e) *strt_time* 39

도면5

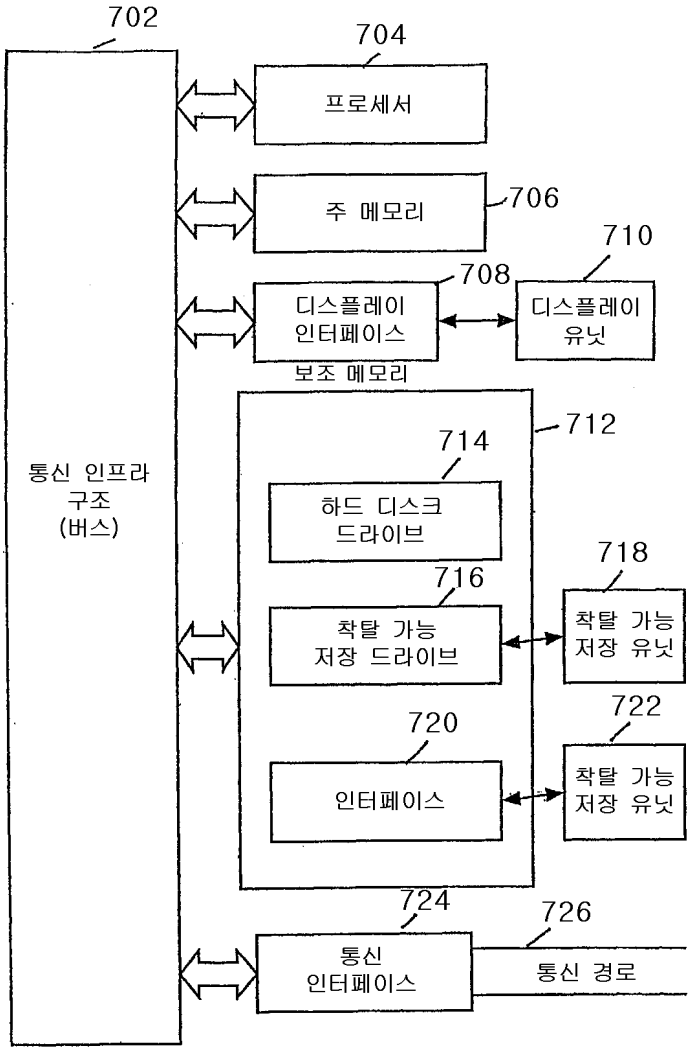


도면6



ts_n	페이지 교체 실행 'n'의 시작시의 타임 스탬프
TW_n	시간 ts_{n-1}과 ts_n 사이의 "빈프레임 대기" 상태의 모든 스레드에 의해 소모한 총시간, 현재 대기 리스트에 있는 모든 스레드에 대해 TW_n = *thrd_wait* + Sum of (ts_n – *waitlist_enque_time*)
thrd_wait_rate	시간 ts_n = TW_n / (ts_n – ts_{n-1})에서

도면7


702
704
프로세서
주 메모리
706
710
708
디스플레이 인터페이스
디스플레이 유닛
보조 메모리
712
714
하드 디스크 드라이브
718
716
착탈 가능 저장 드라이브
착탈 가능 저장 유닛
720
722
인터페이스
착탈 가능 저장 유닛
724
726
통신 인터페이스
통신 경로
통신 인프라 구조 (버스)