(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2011-0049957
(43) 공개일자 2011년05월13일

(51) Int. Cl.
G06F 9/445 (2006.01) *G06F 9/32* (2006.01)
G06F 9/46 (2006.01) *G06F 9/38* (2006.01)
(21) 출원번호 10-2009-0106744
(22) 출원일자 2009년11월06일
심사청구일자 2009년11월06일

(71) 출원인
서강대학교산학협력단
서울 마포구 신수동 1-1 서강대학교
(72) 발명자
박수용
서울특별시 마포구 상수동 369 두산위브아파트 101동 1001호
김동선
서울특별시 마포구 신수동 AS관 712B
김석환
서울특별시 마포구 신수동 81-21 302호
(74) 대리인
특허법인충현

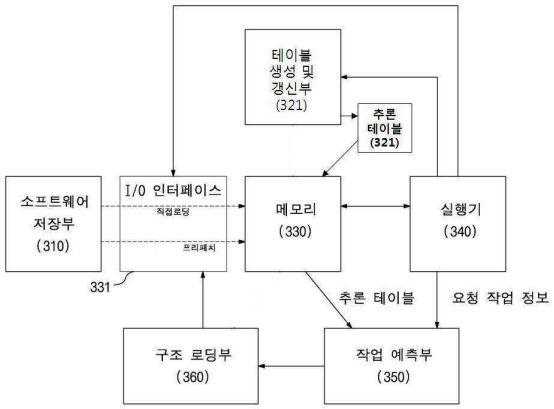
전체 청구항 수 : 총 12 항

(54) 사용기록 기반의 동적 스케줄링 방법, 장치 및 그 기록 매체

(57) 요 약

사용기록 기반의 동적 스케줄링 방법, 장치 및 그 기록 매체가 개시된다. 본 발명의 일 실시 예에 따른 사용기록 기반의 동적 스케줄링 방법은 사용자가 실행한 작업들의 순서 및 빈도가 누적되어 기록된 추론 테이블을 로딩하는 단계; 상기 추론 테이블 및 사용자에 의해 현재 요청된 작업을 기준으로 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정하는 단계; 및 상기 어플리케이션의 컴포넌트들 중 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 미리 로딩하는 단계를 포함한다. 본 발명의 실시 예들에 의하면, 제한된 메모리 내에서 어플리케이션의 구성 요소인 컴포넌트들의 로딩을 동적으로 처리하여 어플리케이션 실행 시의 대기 시간을 줄일 수 있고, 자원 효율성을 높일 수 있다.

대 표 도 - 도3



이 발명을 지원한 국가연구개발사업
과제고유번호 3-5
부처명 지식경제부
연구관리전문기관
연구사업명 인간기능 생활지원 지능로봇 기술개발사업
연구과제명 안정적 서비스를 위한 자가 치유 및 동적 재배치 기술
기여율
주관기관 KIST
연구기간 2004년 7월 1일 ~ 2013년 3월 31일

특허청구의 범위

청구항 1

복수의 컴포넌트들로 구성되는 어플리케이션의 동적 스케줄링 방법에 있어서,

사용자가 실행한 작업들의 순서 및 빈도가 누적되어 기록된 추론 테이블을 로딩하는 단계;

상기 추론 테이블 및 사용자에 의해 현재 요청된 작업을 기준으로 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정하는 단계; 및

상기 어플리케이션의 컴포넌트들 중 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 미리 로딩하는 단계를 포함하는, 사용기록 기반의 동적 스케줄링 방법.

청구항 2

제 1 항에 있어서,

상기 추론 테이블은

상기 어플리케이션의 컴포넌트들의 마지막 사용 시간을 기록하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 방법.

청구항 3

제 1 항에 있어서,

상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 미리 로딩하는 단계는

상기 어플리케이션의 컴포넌트들 중 최근 가장 적게 사용된 컴포넌트를 상기 메모리에서 언로딩한 후, 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 로딩하는 단계인 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 방법.

청구항 4

제 1 항에 있어서,

상기 어플리케이션의 종료시까지 실행된 작업들의 순서 및 빈도를 누적하여 상기 추론 테이블을 갱신하는 단계를 더 포함하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 방법.

청구항 5

제 1 항에 있어서,

상기 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정하는 단계는

상기 추론 테이블을 기준으로, 상기 현재 요청된 작업에 연속하여 가장 자주 실행된 작업들을 상기 스케줄링 작업으로서 선정하는 단계를 포함하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 방법.

청구항 6

제 1 항에 있어서,

상기 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정하는 단계는

상기 메모리의 크기를 고려하여 상기 스케줄링 작업의 수를 결정하는 단계를 포함하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 방법.

청구항 7

제1항 내지 제6항 중 어느 한 항의 방법을 컴퓨터 시스템에서 실행하기 위한 프로그램이 기록된, 컴퓨터 시스템이 판독할 수 있는 기록매체.

청구항 8

복수의 컴포넌트들로 구성되는 어플리케이션을 저장하는 소프트웨어 저장부;

상기 어플리케이션의 종료시마다 사용자가 실행한 작업들의 순서 및 빈도에 따라 추론 테이블을 갱신하는 테이블 생성 및 갱신부;

상기 복수의 컴포넌트들 중 적어도 하나를 일시적으로 저장하는 메모리;

상기 추론 테이블 및 사용자에 의해 현재 요청된 작업을 기준으로 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정하는 작업 예측부; 및

상기 어플리케이션의 컴포넌트들 중 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 미리 로딩하는 구조 로딩부를 포함하는, 사용기록 기반의 동적 스케줄링 장치.

청구항 9

제 8 항에 있어서,

상기 추론 테이블은

상기 어플리케이션의 컴포넌트들의 마지막 사용 시간을 기록하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 장치.

청구항 10

제 8 항에 있어서,

상기 구조 로딩부는

상기 어플리케이션의 컴포넌트들 중 최근 가장 적게 사용된 컴포넌트를 상기 메모리에서 언로딩한 후, 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 로딩하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 장치.

청구항 11

제 8 항에 있어서,

상기 테이블 생성 및 갱신부는

상기 어플리케이션의 종료시까지 실행된 작업들의 순서 및 빈도를 누적하여 상기 추론 테이블을 갱신하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 장치.

청구항 12

제 8 항에 있어서,

상기 작업 예측부는

상기 추론 테이블을 기준으로, 상기 현재 요청된 작업에 연속하여 가장 자주 실행된 작업들을 상기 스케줄링 작업으로서 선정하는 것을 특징으로 하는, 사용기록 기반의 동적 스케줄링 장치.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 본 발명은 어플리케이션 소프트웨어의 실행 및 운영에 관련된 것으로, 특히, 사용기록 기반의 동적 스케줄링 방법, 장치 및 그 기록 매체에 관한 것이다.

배 경 기 술

[0002] 최근, 컴퓨터의 보급이 늘어나면서 사용자들이 다기능화 및 고속화를 요구하면서 컴퓨터 관련 소프트웨어도 발전되고 있다. 이때, 소프트웨어의 다기능화에 따라 크기도 증가하고 있다.

[0003] 그런데, 일반적으로 컴퓨터 시스템에서 소프트웨어를 실행시키기 위하여는, 먼저 실행시키고자 하는 소프트웨어를 주기억장치에 로딩(loading)한 후 중앙처리장치(CPU)를 통해 처리하고 있다.

[0004] 도 1a를 참조하여 설명하면, 외부로부터 사용자에 의한 제어신호를 입력받기 위한 키보드 및 마우스 등과 같은 입력수단(1)과, 상기 입력수단(1)의 제어신호에 따라 외부에서 입력된 다수의 프로그램이 저장되어 있는 하드디스크(2)와, 컴퓨터 시스템에서 처리되고 있는 상황을 사용자에게 알려주기 위하여 디스플레이 시키는 모니터(3)와, 컴퓨터 시스템에서 실행하고자 하는 특정 프로그램이 로딩되는 메인메모리(RAM)(4)와, 사용자에 의해 소정의 입력수단을 통해 실행하고자 하는 프로그램의 선택이 이루어지면 해당 프로그램을 상기 하드디스크(2)에서 메인메모리(4)로 로딩시키는 동작을 제어하는 중앙 처리부(CPU)(5)를 포함하여 이루어진다.

[0005] 상기와 같은 구성을 갖는 컴퓨터 시스템에서의 소프트웨어 실행동작을 설명하면 다음과 같다.

[0006] 먼저, 사용자가 입력수단의 키조작을 통해 하드디스크(2)에 저장되어 있는 프로그램들의 리스트를 모니터(3)에 디스플레이시켜 확인하고, 특정 프로그램의 실행을 선택하면 상기 중앙처리부(5)에서는 선택된 프로그램을 하드디스크(2)로부터 읽어서 메인메모리(4)에 로딩한다.

[0007] 그후, 상기 메인메모리(4)로부터 프로그램의 실행에 필요한 기능들을 부분적으로 읽어오면서 프로그램의 실행을 수행하게 된다.

[0008] 그런데, 상기와 같은 종래의 소프트웨어 실행동작에서는 선택된 프로그램을 완전히 로딩한 후 실행을 수행하기 때문에 프로그램의 로딩에 많은 시간이 필요하게 되는 문제점이 발생한다. 결국, 컴퓨터 시스템의 속도를 저하시키게 되는 문제점이 발생한다.

발명의 내용

해결 하고자하는 과제

[0009] 본 발명이 이루고자 하는 첫 번째 기술적 과제는 어플리케이션의 구성 요소인 컴포넌트들의 로딩을 동적으로 처리하여 어플리케이션 실행 시의 대기 시간을 줄일 수 있는 사용기록 기반의 동적 스케줄링 방법을 제공하는 데 있다.

[0010] 본 발명이 이루고자 하는 두 번째 기술적 과제는 제한된 메모리 내에서 컴포넌트들의 로딩을 동적으로 처리하여 어플리케이션 실행 시의 대기 시간을 줄일 수 있는 사용기록 기반의 동적 스케줄링 장치를 제공하는 데 있다.

[0011] 본 발명이 이루고자 하는 세 번째 기술적 과제는 상기 사용기록 기반의 동적 스케줄링 방법을 컴퓨터 시스템에서 실행하기 위한 프로그램이 기록된 매체로서, 컴퓨터 시스템이 판독할 수 있는 기록매체를 제공하는 데 있다.

과제 해결수단

[0012] 본 발명의 일 실시 예에 따른 사용기록 기반의 동적 스케줄링 방법은 사용자가 실행한 작업들의 순서 및 빈도가 누적되어 기록된 추론 테이블을 로딩하는 단계; 상기 추론 테이블 및 사용자에 의해 현재 요청된 작업을 기준으로 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정하는 단계; 및 상기 어플리케이션의 컴포넌트들 중 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 미리 로딩하는 단계를 포함한다.

[0013] 또한, 본 발명의 일 실시 예에 따른 사용기록 기반의 동적 스케줄링 장치는 복수의 컴포넌트들로 구성되는 어플리케이션을 저장하는 소프트웨어 저장부; 상기 어플리케이션의 종료시마다 사용자가 실행한 작업들의 순서 및 빈도에 따라 추론 테이블을 갱신하는 테이블 생성 및 갱신부; 상기 복수의 컴포넌트들 중 적어도 하나를 일시적으로 저장하는 메모리; 상기 추론 테이블 및 사용자에 의해 현재 요청된 작업을 기준으로 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정하는 작업 예측부; 및 상기 어플리케이션의 컴포넌트들 중 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 미리 로딩하는 구조 로딩부를 포함한다.

[0014] 또한, 상기의 사용기록 기반의 동적 스케줄링 방법을 컴퓨터에서 실행시키기 위한 프로그램을 컴퓨터로 읽을 수 있는 기록매체에 기록하여 제공한다.

효 과

[0015] 본 발명의 실시 예들에 의하면, 제한된 메모리 내에서 어플리케이션의 구성 요소인 컴포넌트들의 로딩을 동적으로 처리하여 어플리케이션 실행 시의 대기 시간을 줄일 수 있고, 자원 효율성을 높일 수 있다.

발명의 실시를 위한 구체적인 내용

[0016] 이하에서는 도면을 참조하여 본 발명의 바람직한 실시 예를 설명하기로 한다. 그러나, 다음에 예시하는 본 발명의 실시 예는 여러 가지 다른 형태로 변형될 수 있으며, 본 발명의 범위가 다음에 상술하는 실시 예에 한정되는 것은 아니다.

[0017] 도 1b는 어플리케이션의 초기 사용까지의 대기 시간을 도시한 것이다.

[0018] 도 1b의 결과는 포토샵과 이클리스라는 어플리케이션의 초기 실행 시에 대기 시간을 10회 측정한 후 평균값을 구한 것이다. 컴포넌트를 수정하여 대기 시간을 줄이는 방법은 추가 비용이 들고 항상 가능한 것도 아니다.

[0019] 본 발명의 실시 예들에서는 작업 실행 및 컴포넌트 로딩을 병력적으로 수행한다. 어플리케이션이 사용자의 작업을 예측하고, 사용자의 요청에 앞서 필요한 컴포넌트를 프리패치(prefetch)할 수 있다면 전체 대기 시간은 줄어들 수 있다.

[0020] 도 2a는 특정 어플리케이션에 대해 작업 선호도가 다른 사용자들을 위한 기능 집합의 예를 도시한 것이다.

[0021] 일반적으로, 사용자들은 하나의 어플리케이션 내에서도 각자 다른 기능을 선호하게 된다. 예를 들어, 사용자 A가 포토샵의 기본 그림 툴만을 선호하는 반면, 사용자 B는 기본 그림 툴과 효과 툴을 동시에 선호하고, 사용자 C는 기본 그림 툴, 효과 툴, 그래픽 필터, 기타 다른 플러그인을 선호할 수 있다. 사용자마다 선호하는 작업이 상이한 상황은 워드 프로세서나 기타 다른 어플리케이션에서도 발생할 수 있다. 본 발명의 실시 예들은 특정 어플리케이션 내에서 선호도의 차이를 고려한다.

[0022] 도 2b는 동일한 작업 선호도를 가지는 사용자들을 위한 서로 다른 순서로 작업을 수행하는 예를 도시한 것이다.

[0023] 사용자마다 각 작업에 대한 선호도가 동일하다고 하더라도 사용자들은 각자 다른 순서로 작업들을 수행할 수 있다. 어떤 사용자는 특정한 시간에 특정한 작업만을 수행하기를 선호할 수 있다. 본 발명의 실시 예들은 사용자의 작업 순서에 대한 선호도를 고려한다.

[0024] 본 발명의 실시 예들에서는 사용자의 작업 히스토리를 누적하여 사용자가 주로 사용하는 컴포넌트의 빈도 및 순서에 따라 사용자의 작업을 예측하고, 예측 결과에 따라 컴포넌트들을 로딩한다.

[0025] 도 3은 본 발명의 일 실시 예에 따른 사용기록 기반의 동적 스케줄링 장치의 블록도이다.

[0026] 소프트웨어 저장부(310)는 복수의 컴포넌트들로 구성되는 어플리케이션을 저장한다. 소프트웨어 저장부(310)는 하드 디스크 드라이브, 광 디스크 및 광 디스크 드라이브 등 소프트웨어의 저장 및/또는 저장된 소프트웨어를 읽어들일 수 있는 수단을 포함한다.

[0027] 테이블 생성 및 갱신부(320)는 어플리케이션의 종료시마다 사용자의 작업 순서 및 빈도에 따라 추론 테이블(321)을 갱신한다. 보다 구체적으로, 테이블 생성 및 갱신부(320)는 어플리케이션의 종료시까지 실행된 작업들의 순서 및 빈도를 누적하여 추론 테이블(321)을 갱신한다. 갱신된 추론 테이블(321)은 어플리케이션의 실행 초기에 메모리(330)에 로딩된다. 본 발명의 다른 실시 예에서, 추론 테이블(321)은 어플리케이션의 컴포넌트들의 마지막 사용 시간이 기록될 수 있다.

[0028] 메모리(330)는 복수의 컴포넌트들 중 적어도 하나를 일시적으로 저장한다. 메모리(330)에는 휘발성 메모리 소자인 D램이 사용될 수 있으나, 본 발명이 여기에 한정되는 것은 아니다.

[0029] I/O 인터페이스(331)는 소프트웨어 저장부(310)에 저장된 데이터들을 메모리(330)에 전달하는 역할을 한다. I/O 인터페이스(331)는 실행기(340) 또는 구조 로딩부(360)의 제어에 따라 직접 로딩 또는 프리패치를 수행할 수 있다. 직접 로딩은 사용자의 요청이 있을 때에만 요청된 컴포넌트를 로딩하는 것을 의미하고, 사용자의 요청이 없어도 사용될 것으로 예측되는 컴포넌트를 미리 로딩하는 것을 의미한다.

[0030] 실행기(340)는 사용자의 어플리케이션 실행 요청을 입력받고, 요청된 어플리케이션이 소프트웨어 저장부(310)에서 메모리(330)에 로딩되도록 제어한다. 일반적인 PC에서는 사용자가 원하는 어플리케이션의 아이콘을 마우스 등의 입력 수단으로 클릭하면 실행기(340)가 동작하게 된다. 어플리케이션의 로딩 이후에 실행기(340)는 메모리(330)를 이용하여 사용자의 작업 진행을 위해 필요한 연산을 수행한다.

[0031] 작업 예측부(350)는 추론 테이블 및 사용자에 의해 현재 요청된 작업을 기준으로 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정한다. 본 발명의 다른 실시 예에서, 작업 예측부(350)는 추론 테이블을 기준으로, 현재 요청된 작업에 연속하여 가장 자주 실행된 작업들을 스케줄링 작업으로서 선정한다.

[0032] 구조 로딩부(360) 어플리케이션의 컴포넌트들 중 작업 예측부(350)에서 결정한 스케줄링 작업에 대응하는 컴포넌트들을 메모리(330)에 미리 로딩한다. 구조 로딩부(360)에는 어플리케이션의 컴포넌트들 중 최근 가장 적게 사용된 컴포넌트를 메모리(330)에서 언로딩한 후, 스케줄링 작업에 상응하는 컴포넌트들을 메모리(330)에 로딩하는 방법이 적용될 수 있다.

[0033] 도 4는 본 발명의 일 실시 예에 따른 사용기록 기반의 동적 스케줄링 방법의 흐름도이다.

[0034] 먼저, 사용자로부터 어플리케이션의 실행 요청이 있으면, 추론 테이블을 로딩한다(S410). 추론 테이블은 과거에 사용자가 실행한 작업들의 순서 및 빈도가 누적되어 기록된 데이터이다. 예를 들어, 추론 테이블은 어플리케이션의 컴포넌트들의 마지막 사용 시간을 기록해 놓은 데이터일 수 있다.

[0035] 다음, 추론 테이블 및 사용자에 의해 현재 요청된 작업을 기준으로 가장 빈도가 높은 연속 작업들을 스케줄링 작업으로서 결정한다(S420). 본 발명의 일 실시 예에서, 스케줄링 과정(S420)은 추론 테이블을 기준으로, 현재 요청된 작업에 연속하여 가장 자주 실행된 작업들을 스케줄링 작업으로서 선정하는 과정을 포함한다. 본 발명의 다른 실시 예에서는 메모리의 크기를 고려하여 스케줄링 작업의 수를 결정한다.

[0036] 어플리케이션의 컴포넌트들 중 상기 스케줄링 작업에 상응하는 컴포넌트들을 메모리에 미리 로딩한다(S430). 본 발명의 실시 예들에서, 작업 예측 다음에는 구조 로딩이 이어진다. 즉, 작업 예측 과정에 의해 예측된 작업들에 기초하여 어플리케이션은 상기 작업들에 대응하는 작업을 로딩한다. 이는 어플리케이션이 상기 대응하는 작업 구조로 컴포넌트들을 로딩하는 것을 의미한다. 예를 들어, 현재 로딩된 컴포넌트 집합 Ccur이 {C2, C4, C5, C6, C7, C13}이고, 요청된 컴포넌트 집합 Creq가 {C4, C8, C9, C12}인 경우를 가정한다. 이러한 요청에 대해, C4는 미리 로딩되고, C8, C9, C12는 새롭된 로딩될 필요가 있다, 결국, {C2, C4, C5, C6, C7, C8, C9, C12, C13}가 요청된 컴포넌트 집합이다.

[0037] 이후, 사용자가 어플리케이션을 종료하면(S440) 지금까지 실행된 작업들의 순서 및 빈도를 정리하여 상기 추론 테이블을 갱신한다(S450). 반면, 어플리케이션을 종료하지 않고 사용자가 작업을 계속하면 사용자의 현재 작업에 이어질 것으로 예상되는 다음 작업의 컴포넌트들을 예측하는 과정(S420)부터 반복한다.

[0038] 만약 추론 테이블이 갱신된 후에 어플리케이션의 재실행 요청이 있으면(S450), 갱신된 추론 테이블을 메모리에 로딩한 후에 추론 테이블을 기준으로 사용자가 어플리케이션의 실행 초기에 수행할 작업이 무엇인지 예측하는 과정(S420)부터 순차적으로 수행한다. 재실행 요청이 없으면 모든 절차를 종료한다.

[0039] 동적 구조 스케줄링의 필요성은 어플리케이션이 사용할 수 있는 메모리 공간의 최대 양과 관련이 있다. 메모리 공간에 요청된 모든 컴포넌트를 로딩할 수는 없다. 이러한 공간 제약을 극복하기 위해, 대체 알고리즘(Replacement algorithm}이 사용된다. 대체 알고리즘에는 적응적 대체 캐쉬(Adaptive Replacement Cache; ARC), 최근에 가장 사용되지 않은 것을 우선(Least Recently Used; LRU), 그리디 듀얼 사이즈(GreedyDual Size; GDS), 가장 적게 사용되는 것을 우선(Least Frequently Used), 최근에 가장 많이 사용되는 것을 우선(Most Recently Used), 및 유사-LRU 등이 있다. 이하에서는 LRU 알고리즘을 기준으로 구조 로딩을 설명한다.

[0040] LRU 알고리즘을 적용할 경우, 최근에 가장 적게 사용되는 컴포넌트는 특정 메모리 공간에서 제거된다. 어플리케이션은 모든 컴포넌트의 마지막 사용 시간을 기록하는 테이블을 관리한다. 컴포넌트의 대체가 필요하면, 어플리케이션은 메모리 제약 조건이 충족될때가지 컴포넌트들을 언로딩한다. 위 예에서, C4, C8, C9, C12는 새롭게 요청된 컴포넌트이므로 MRU 컴포넌트로 식별된다. 그리고나서 컴포넌트 집합 {C2, C4, C5, C6, C7, C8, C9, C12, C13}로부터 최근에 가장 적게 사용되는 컴포넌트를 제거한다. 최근에 가장 적게 사용되는 컴포넌트를 결정하는 과정은 대기 시간의 증가를 가져오지 않는다.

[0041] 메모리에 없는 컴포넌트가 요청되는 경우(즉, 예측이 잘못된 경우), 어플리케이션은 컴포넌트를 로딩해야 한다. 이는 불가피하게 대기 시간을 증가시킨다. 따라서, 예측의 정확성이 성능에 영향을 준다. 예측 오류를 줄이는 것이 중요하다.

[0042] 도 5는 도 4의 어플리케이션 종료 여부를 판단하는 과정(S440) 이후에 추가될 수 있는 컴포넌트 언로딩 과정의 상세 흐름도이다.

[0043] 먼저, 지금까지 사용자에 의해 수행된 작업들을 기준으로 다시 수행될 확률이 떨어지는 작업의 컴포넌트들의 목

록을 정리한다(S541). 이 과정(S541)에는 과거의 어플리케이션 활용 행태를 반영하기 위해 추론 테이블이 참고 데이터로 사용될 수도 있다. 특히, 다시 수행될 확률이 떨어지는 작업의 컴포넌트들은 최근 가장 적게 사용된 컴포넌트들일 수 있다.

[0044] 다음, 위에서 정리된 목록의 컴포넌트들은 다시 사용될 확률이 적으므로, 이를 컴포넌트들을 메모리에서 언로딩하여 메모리 공간을 확보한다(S542).

[0045] 도 6은 도 4의 스케줄링 작업 결정 과정(S420)에 사용되는 추론 테이블을 생성하는 과정의 예를 도시한 것이다.

[0046] 본 발명의 일 실시 예에서, 어플리케이션은 사용자의 다음 작업을 추론하기 위해 사용자의 작업 실행 기록을 처음부터 끝까지 기록한다. 이 과정에서, 어플리케이션은 실행 중인 모든 작업을 기록하고, 각 작업에 이어지는 작업들의 빈도를 카운트한다. 예를 들어, 어플리케이션이 사용자의 작업을 끝마칠 때, 어플리케이션은 도 6에 도시된 연속적인 작업들의 수행 횟수를 각각 카운트한다. 추론 테이블의 갱신 과정은 현재 작업의 이전 작업 기록(바로 이전에 실행된 작업 기록 포함)을 추론 테이블에 반영하는 과정이다.

[0047] 빠른 스케줄링을 위해 어플리케이션의 종료 후에 추론 테이블이 생성된다. 테이블 생성은 이전의 모든 작업 실행 기록을 검사하므로, 시간이 소모되는 작업이다. 그러나, 이 작업은 어플리케이션 실행 이후에 수행되고, 전체 대기 시간에 영향을 주지 않는다.

[0048] 도 7a 및 7b는 도 4의 스케줄링 작업 결정 과정(S420)의 예를 도시한 것이다.

[0049] 작업 실행 및 컴포넌트 로딩을 병렬적으로 수행하기 위해, 사용자의 과거 작업 기록에 기반하여 다음의 연속적인 작업들을 예측한다. 이를 위해, 사용자의 작업 실행 기록을 추론 테이블에 누적하고, 추론 테이블에 근거하여 가능한 작업들을 추론한다. 동적 구조 스케줄링의 목적이 대기 시간을 줄이는 것이므로, 이 추론은 빠르게 수행되어야 하고 가능한 정확해야 한다. 이를 위해, 추론 테이블에 기초하여 다음 작업을 예측할 때, 주어진 추론 테이블 및 사용자에 의해 요청된 현재 작업에 연속하여 가장 자주 출현하는 작업들을 프리패치될 작업으로서 선정할 수 있다.

[0050] 프리패치될 작업의 수는 윈도우 크기 w에 의해 결정된다. 예를 들어, 도 7a에서, 현재 작업이 작업 A이고, 윈도우 크기가 3(w=3)인 경우, 현재 주어진 추론 테이블에 기초하여 어플리케이션은 작업 B, C 및 D를 프리패치하도록 결정한다.

[0051] 도 7b는 각 시퀀스의 2번째 및 3번째의 연속 작업을 반영하기 보다는 연속될 확률이 높은 작업 순으로 프리패치하는 예를 보여준다.

[0052] 즉, A, B, C, D가 하나의 시퀀스이고, B, D, B, C가 두 번째 시퀀스이며, D, H, D, E가 세 번째 시퀀스이고, H, G, C, K가 네 번째 시퀀스인 경우, 현재 작업이 작업 A라면, 이어지는 작업을 B, D, H의 순으로 예측하는 것이다.

[0053] 이러한 예측 방법은 테이블 룩업에 따라 미래 작업들을 예측하는 것이다. 이 방법이 최적의 스케줄링을 보장하는 것은 아니지만, 단순한 테이블 룩업은 빠른 스케줄링을 보장한다.

[0054] 도 7a의 예측 방법은 사용자의 작업 순서가 최초 작업이 무엇인지에 크게 의존하는 경우에 효과가 증가한다. 도 7b의 예측 방법은 최초 작업에 관계없이 계속해서 확률이 가장 큰 작업을 프리패치할 수 있다.

[0055] 도 8은 본 발명의 일 실시 예에 따른 동적 스케줄링을 적용한 결과를 그래프로 도시한 것이다.

[0056] 도 8에서 시퀀스의 평균 길이는 55이고, 윈도우 크기는 1~22까지이다. 그래프는 윈도우 크기가 4 이상부터는 히트율이 80% 이상이 됨을 보여준다. 그리고, 윈도우 크기가 커질 수록 히트율은 100%에 근접한다. 그러나, 윈도우 크기가 너무 크면 컴포넌트의 대체가 필요한 경우가 많아지므로 적절한 윈도우 크기를 설정하는 것이 바람직하다.

[0057] 도 9는 본 발명의 일 실시 예에 따른 동적 스케줄링을 적용하여 대기 시간을 줄어든 것을 나타내는 그래프이다.

[0058] 몇 가지 어플리케이션을 이용하여 대기 시간을 측정한 결과, 본 발명의 일 실시 예에 따를 때, 기존 대기 시간보다 짧은 대기 시간을 구현할 수 있음을 알 수 있다. 구체적으로, C.E.L Pad의 대기 시간은 22% 감소했고, JDraw의 대기 시간은 41% 감소했다.

[0059] 본 발명은 소프트웨어를 통해 실행될 수 있다. 바람직하게는, 본 발명의 실시 예들에 따른 사용기록 기반의 동적 스케줄링 방법을 컴퓨터에서 실행시키기 위한 프로그램을 컴퓨터로 읽을 수 있는 기록매체에 기록하여 제공

할 수 있다. 소프트웨어로 실행될 때, 본 발명의 구성 수단들은 필요한 작업을 실행하는 코드 세그먼트들이다. 프로그램 또는 코드 세그먼트들은 프로세서 판독 가능 매체에 저장되거나 전송 매체 또는 통신망에서 반송파와 결합된 컴퓨터 데이터 신호에 의하여 전송될 수 있다.

[0060] 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 장치의 예로는 ROM, RAM, CD-ROM, DVD±ROM, DVD-RAM, 자기 테이프, 플로피 디스크, 하드 디스크(hard disk), 광데이터 저장장치 등이 있다. 또한, 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 장치에 분산되어 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다.

[0061] 본 발명은 도면에 도시된 일 실시 예를 참고로 하여 설명하였으나 이는 예시적인 것에 불과하며 당해 분야에서 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 실시 예의 변형이 가능하다는 점을 이해할 것이다. 그리고, 이와 같은 변형은 본 발명의 기술적 보호범위 내에 있다고 보아야 한다. 따라서, 본 발명의 진정한 기술적 보호범위는 첨부된 특허청구범위의 기술적 사상에 의해서 정해져야 할 것이다.

산업이용 가능성

[0062] 본 발명은 어플리케이션 소프트웨어의 실행 및 운영에 관련된 것으로, 개인용 컴퓨터, 노트북 컴퓨터, 각종 휴대용 전자기기에 적용되거나 소프트웨어의 구동 모듈, 라이브러리, 플러그인 등에 사용될 수 있다.

도면의 간단한 설명

[0063] 도 1a는 일반적인 컴퓨터 시스템의 블록도이다.

[0064] 도 1b는 어플리케이션의 초기 사용까지의 대기 시간을 도시한 것이다.

[0065] 도 2a는 특정 어플리케이션에 대해 작업 선호도가 다른 사용자들을 위한 기능 집합의 예를 도시한 것이다.

[0066] 도 2b는 동일한 작업 선호도를 가지는 사용자들을 위한 서로 다른 순서로 작업을 수행하는 예를 도시한 것이다.

[0067] 도 3은 본 발명의 일 실시 예에 따른 사용기록 기반의 동적 스케줄링 장치의 블록도이다.

[0068] 도 4는 본 발명의 일 실시 예에 따른 사용기록 기반의 동적 스케줄링 방법의 흐름도이다.

[0069] 도 5는 도 4의 어플리케이션 종료 여부를 판단하는 과정(S440) 이후에 추가될 수 있는 컴포넌트 언로딩 과정의 상세 흐름도이다.

[0070] 도 6은 도 4의 스케줄링 작업 결정 과정(S420)에 사용되는 추론 테이블을 생성하는 과정의 예를 도시한 것이다.

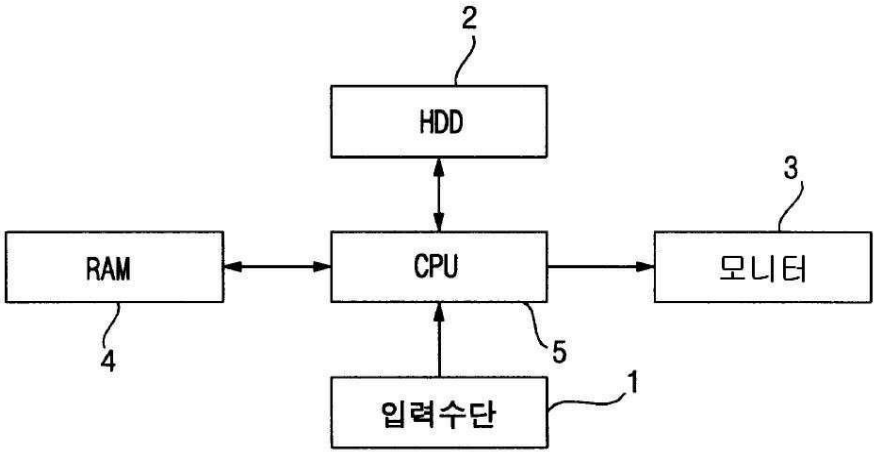
[0071] 도 7a 및 7b는 도 4의 스케줄링 작업 결정 과정(S420)의 예를 도시한 것이다.

[0072] 도 8은 본 발명의 일 실시 예에 따른 동적 스케줄링을 적용한 결과를 그래프로 도시한 것이다.

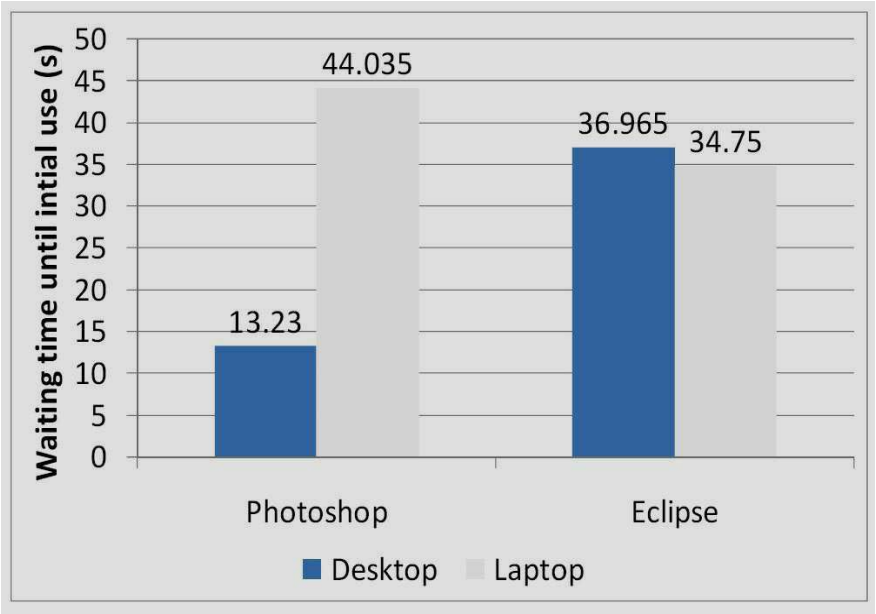
[0073] 도 9는 본 발명의 일 실시 예에 따른 동적 스케줄링을 적용하여 대기 시간을 줄어든 것을 나타내는 그래프이다.

도면

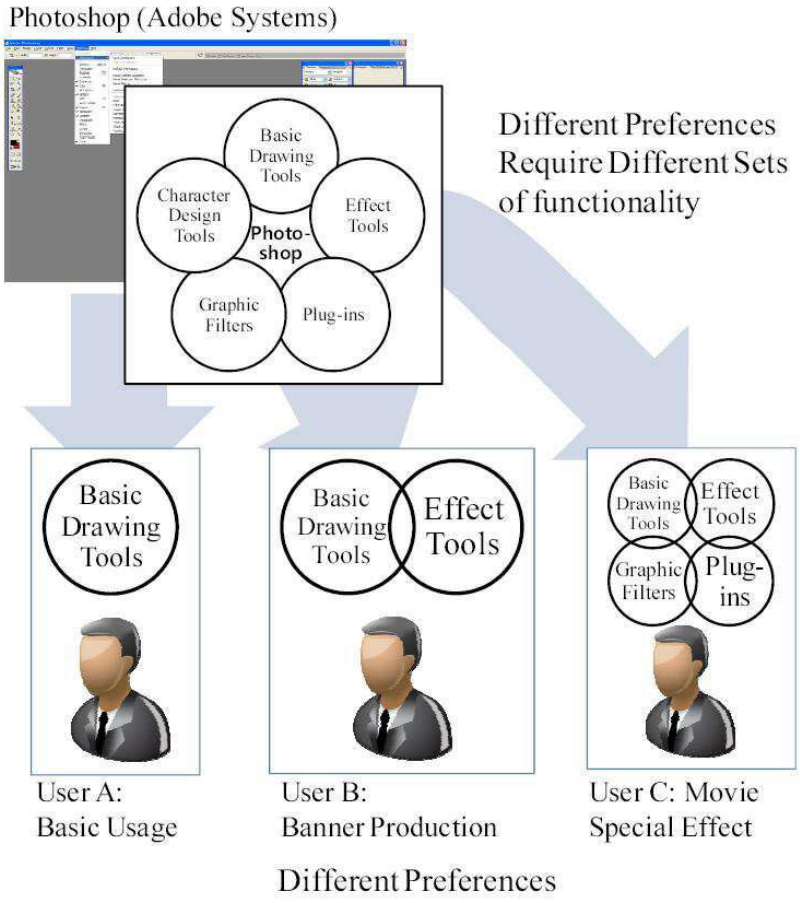
도면1a


2
HDD
3
RAM
CPU
모니터
4
5
1
입력수단

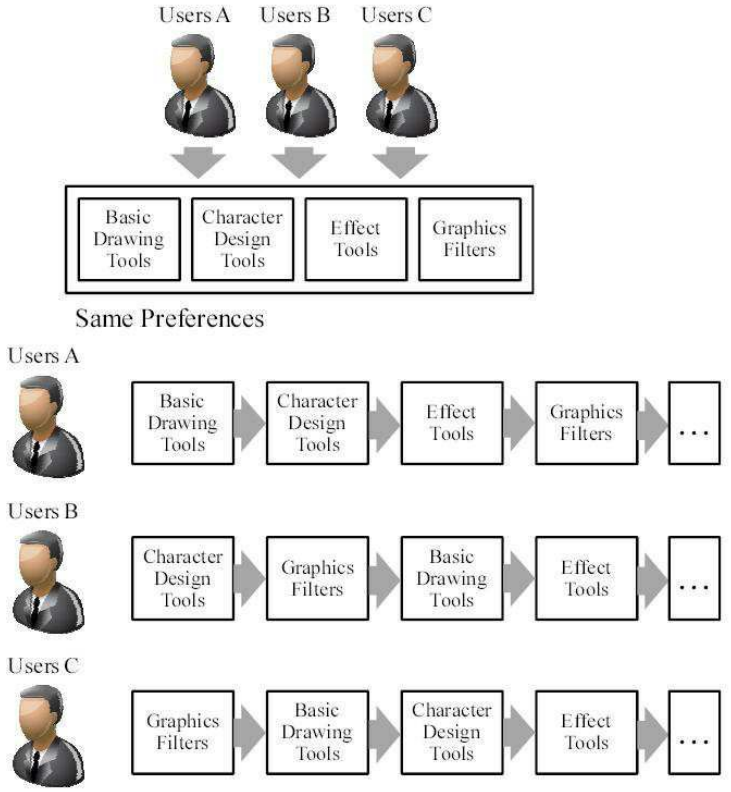
도면1b


Waiting time until intial use (s)
50
45
40
35
30
25
20
15
10
5
0
44.035
36.965
34.75
13.23
Photoshop
Eclipse
Desktop
Laptop

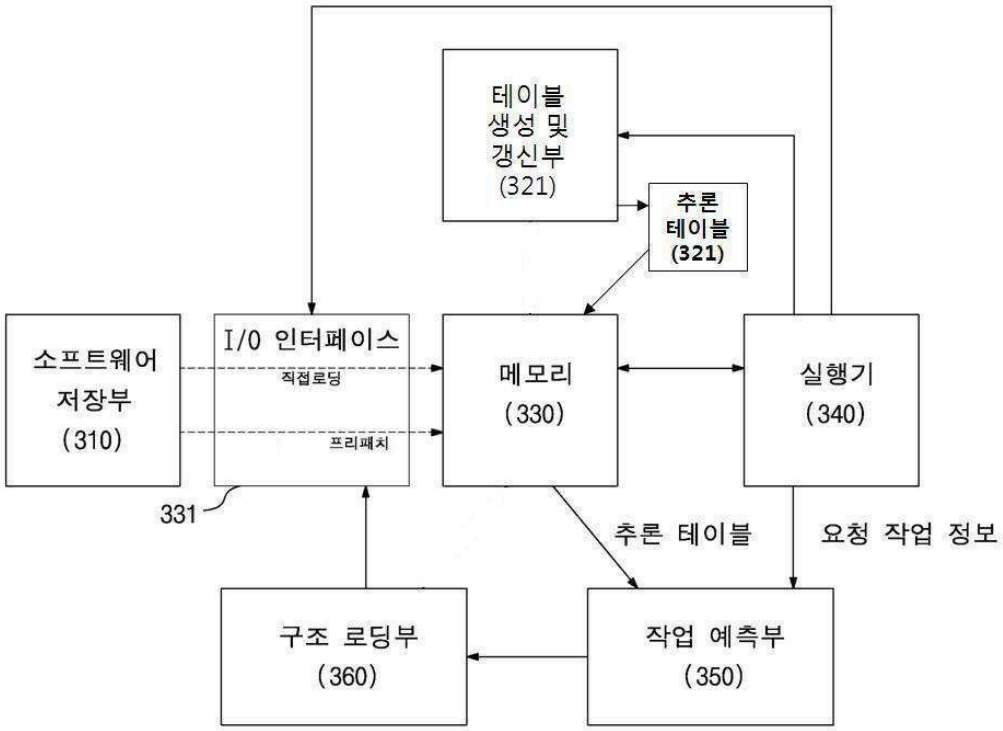
도면2a


Photoshop (Adobe Systems)
Basic Drawing Tools
Character Design Tools
Photo-shop
Effect Tools
Graphic Filters
Plug-ins
Different Preferences Require Different Sets of functionality
Basic Drawing Tools
Basic Drawing Tools
Effect Tools
Basic Drawing Tools
Effect Tools
Graphic Filters
Plug-ins
User A: Basic Usage
User B: Banner Production
User C: Movie Special Effect
Different Preferences

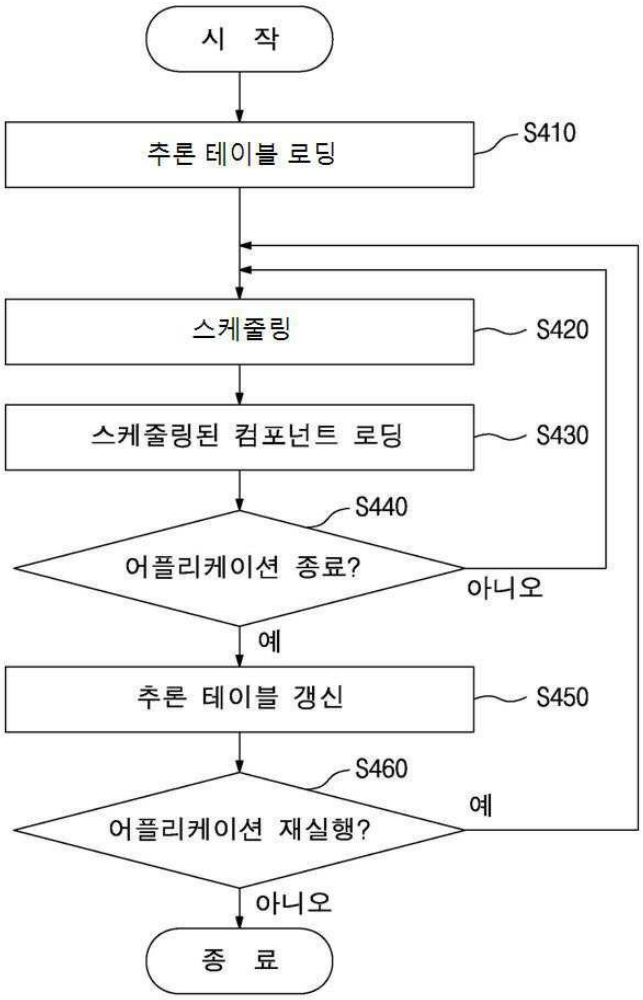
도면2b


Users A
Users B
Users C
Basic Drawing Tools
Character Design Tools
Effect Tools
Graphics Filters
Same Preferences
Users A
Basic Drawing Tools
Character Design Tools
Effect Tools
Graphics Filters
...
Users B
Character Design Tools
Graphics Filters
Basic Drawing Tools
Effect Tools
...
Users C
Graphics Filters
Basic Drawing Tools
Character Design Tools
Effect Tools
...

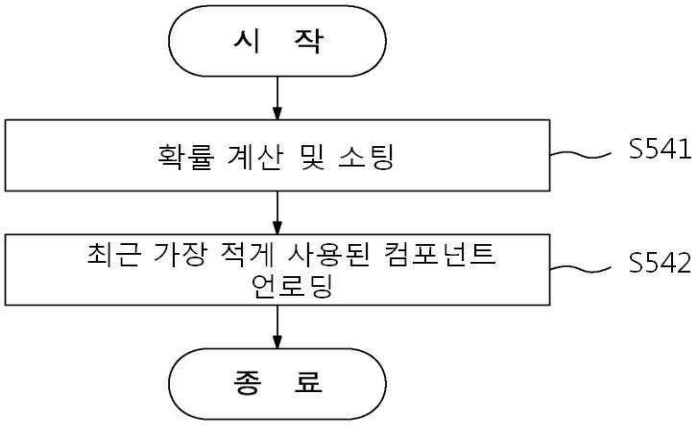
도면3


테이블 생성 및 갱신부 (321)
추론 테이블 (321)
소프트웨어 저장부 (310)
I/O 인터페이스
직접로딩
프리패치
331
메모리 (330)
실행기 (340)
추론 테이블
요청 작업 정보
구조 로딩부 (360)
작업 예측부 (350)

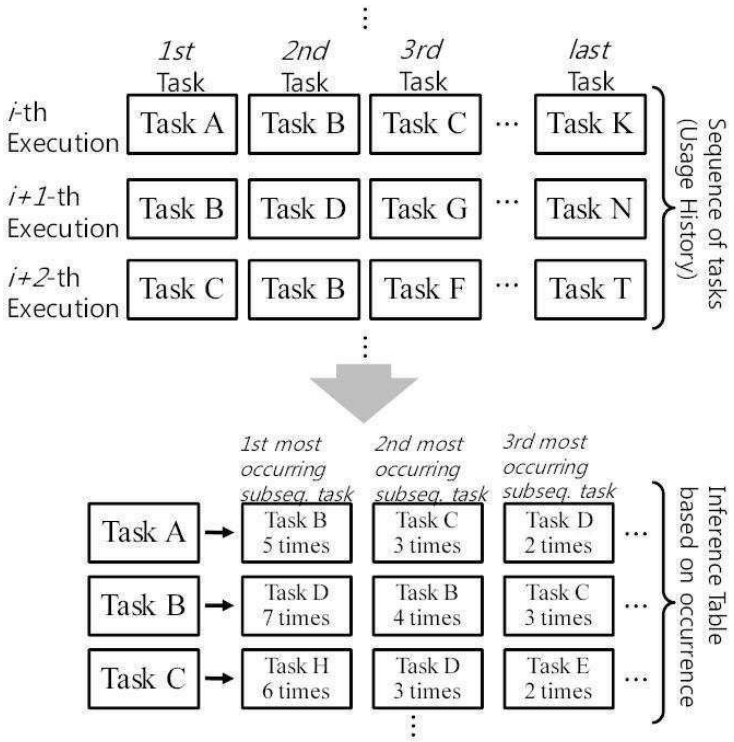
도면4


시 작
추론 테이블 로딩
S410
스케줄링
S420
스케줄링된 컴포넌트 로딩
S430
S440
어플리케이션 종료?
아니오
예
추론 테이블 갱신
S450
S460
어플리케이션 재실행?
예
아니오
종 료

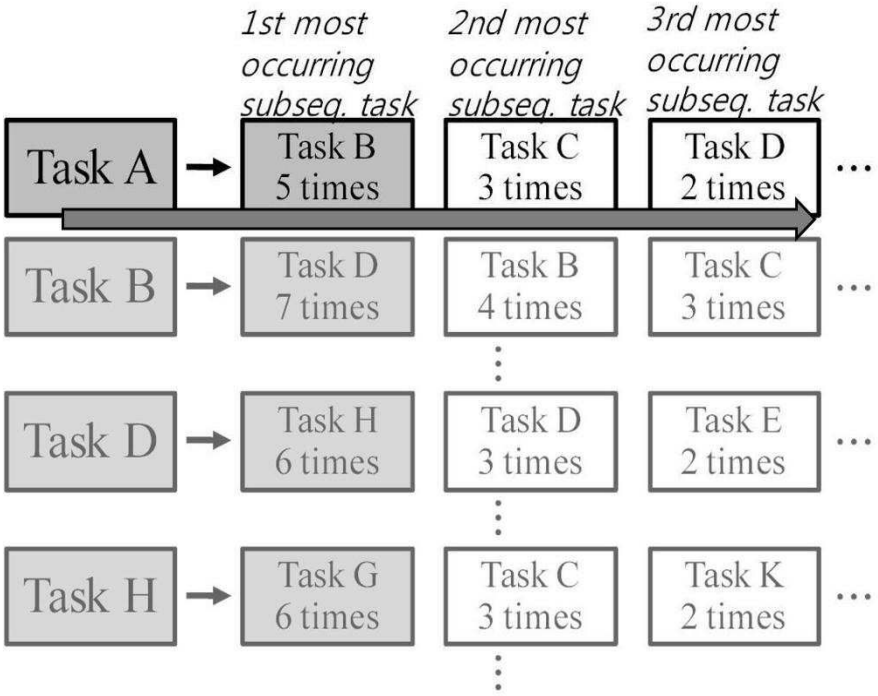
도면5


시 작
확률 계산 및 소팅
S541
최근 가장 적게 사용된 컴포넌트 언로딩
S542
종 료

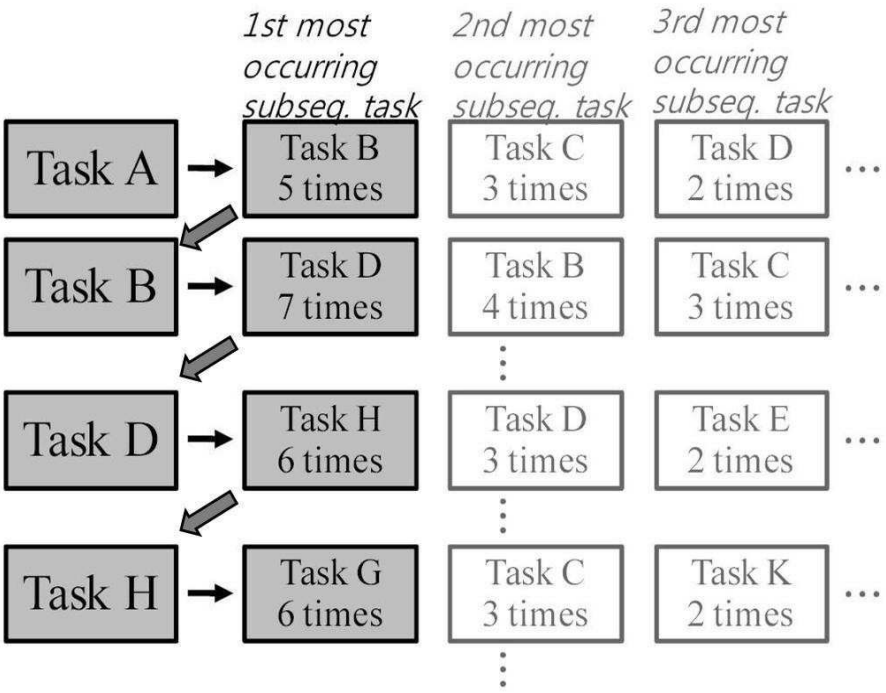
도면6


1st Task
2nd Task
3rd Task
last Task
i-th Execution
Task A
Task B
Task C
Task K
i+1-th Execution
Task B
Task D
Task G
Task N
i+2-th Execution
Task C
Task B
Task F
Task T
Sequence of tasks (Usage History)
1st most occurring subseq. task
2nd most occurring subseq. task
3rd most occurring subseq. task
Task A
Task B 5 times
Task C 3 times
Task D 2 times
Task B
Task D 7 times
Task B 4 times
Task C 3 times
Task C
Task H 6 times
Task D 3 times
Task E 2 times
Inference Table based on occurrence

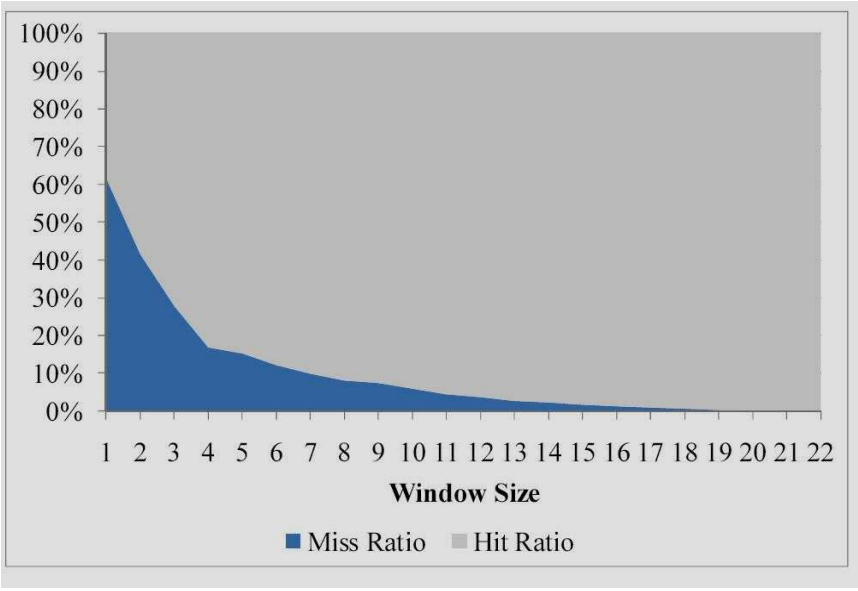
도면7a


1st most occurring subseq. task
2nd most occurring subseq. task
3rd most occurring subseq. task
Task A
Task B 5 times
Task C 3 times
Task D 2 times
Task B
Task D 7 times
Task B 4 times
Task C 3 times
Task D
Task H 6 times
Task D 3 times
Task E 2 times
Task H
Task G 6 times
Task C 3 times
Task K 2 times

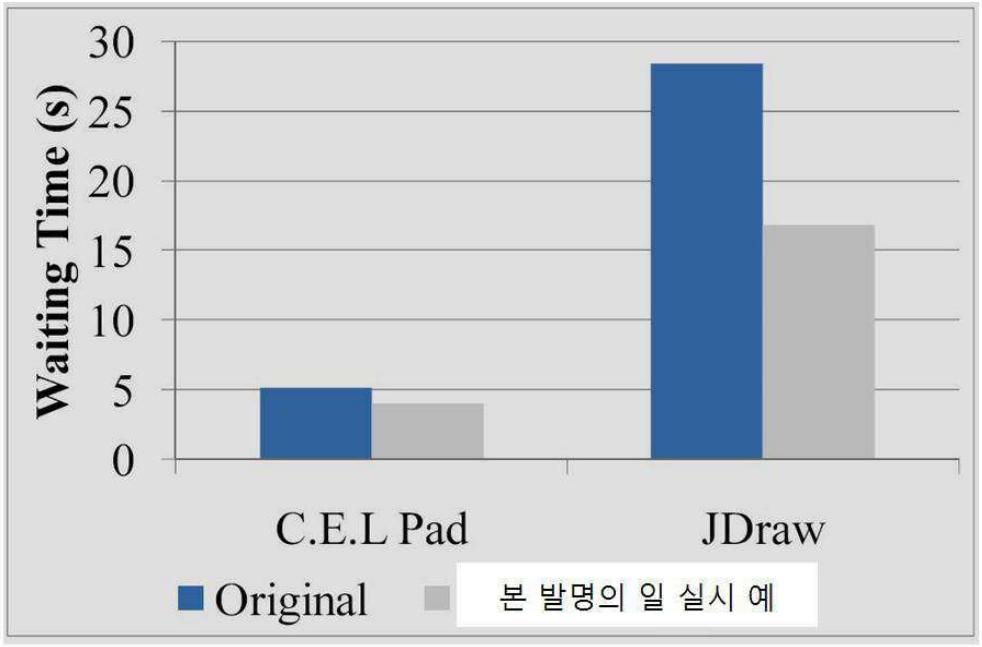
도면7b


1st most occurring subseq. task
2nd most occurring subseq. task
3rd most occurring subseq. task
Task A
Task B 5 times
Task C 3 times
Task D 2 times
Task B
Task D 7 times
Task B 4 times
Task C 3 times
Task D
Task H 6 times
Task D 3 times
Task E 2 times
Task H
Task G 6 times
Task C 3 times
Task K 2 times

도면8


100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Window Size
Miss Ratio
Hit Ratio

도면9


Waiting Time (s)
30
25
20
15
10
5
0
C.E.L Pad
JDraw
Original
본 발명의 일 실시 예