



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0060421
(43) 공개일자 2020년05월29일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2018.01) G06F 9/48 (2018.01)
G06N 20/00 (2019.01)
(52) CPC특허분류
G06F 9/5038 (2013.01)
G06F 9/4881 (2013.01)
(21) 출원번호 10-2020-7011136
(22) 출원일자(국제) 2018년10월11일
심사청구일자 2020년04월27일
(85) 번역문제출일자 2020년04월17일
(86) 국제출원번호 PCT/CN2018/109753
(87) 국제공개번호 WO 2019/072200
국제공개일자 2019년04월18일
(30) 우선권주장
201710953233.7 2017년10월13일 중국(CN)

(71) 출원인
후아웨이 테크놀로지 컴퍼니 리미티드
중국 518129 광둥성 셴젠 룡강 디스트릭트 반티안
후아웨이 어드미니스트레이션 빌딩
(72) 발명자
첸 쿼올린
중국 518129 광둥성 셴젠 룡강 디스트릭트 반티안
후아웨이 어드미니스트레이션 빌딩
첸 한빙
중국 518129 광둥성 셴젠 룡강 디스트릭트 반티안
후아웨이 어드미니스트레이션 빌딩
강 지
중국 518129 광둥성 셴젠 룡강 디스트릭트 반티안
후아웨이 어드미니스트레이션 빌딩
(74) 대리인
제일특허법인(유)

전체 청구항 수 : 총 33 항

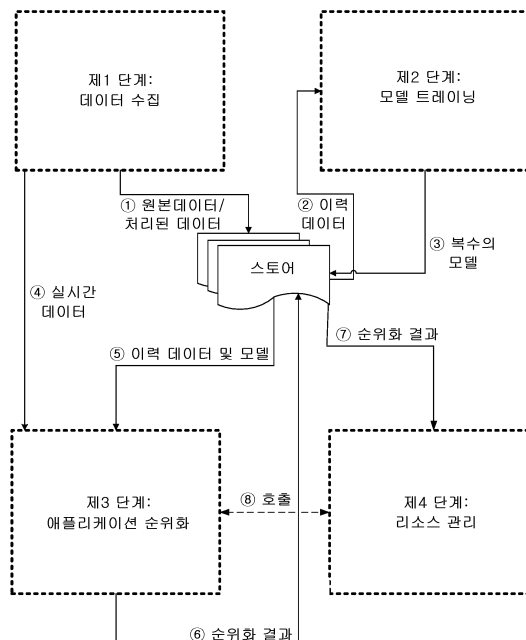
(54) 발명의 명칭 리소스 관리 방법 및 단말 장치

(57) 요약

본원은 컴퓨터 시스템에서 리소스를 관리하기 위한 방법 및 단말 장치를 제공한다. 방법은, 데이터를 취득- 데이터는 현재의 포그라운드 애플리케이션과 관련되는 애플리케이션 시퀀스 특징 데이터를 포함하고, 또한 데이터는 이하의 실시간 데이터, 즉 컴퓨터 시스템의 시스템 시간, 컴퓨터 시스템의 현재 스테이더스 데이터, 및 컴퓨터

(뒷면에 계속)

대표도 - 도4



터 시스템의 현재 위치 데이터 중 적어도 하나를 더 포함함 -하는 단계; 실시간 데이터 중 적어도 하나의 데이터에 기초하여 복수의 머신 러닝 모델로부터, 실시간 데이터와 일치하는 타깃 머신 러닝 모델을 선택하는 단계; 취득된 데이터를 타깃 머신 러닝 모델에 입력하여, 컴퓨터 시스템에 인스톨된 복수의 애플리케이션의 중요도를 순위화하는 단계; 및 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계를 포함한다. 이 방법에 있어서는, 애플리케이션의 중요도를 식별하는 정확도가 향상되므로, 리소스 할당/예비할당 또는 리소스 재활용의 대상이 더욱 정확하고, 이로써 리소스 관리의 효율성이 향상된다.

(52) CPC특허분류

G06N 20/00 (2019.01)

명세서

청구범위

청구항 1

컴퓨터 시스템에서 리소스를 관리하기 위한 방법으로서,

데이터를 취득— 상기 데이터는 현재의 포그라운드 애플리케이션과 관련되는 애플리케이션 시퀀스 특징 데이터를 포함하고, 이하의 실시간 데이터, 즉 상기 컴퓨터 시스템의 시스템 시간, 상기 컴퓨터 시스템의 현재 스테이 터스 데이터, 및 상기 컴퓨터 시스템의 현재 위치 데이터 중 적어도 하나를 더 포함함 —하는 단계;

상기 실시간 데이터 중 적어도 하나의 데이터에 기초하여 복수의 머신 러닝 모델로부터, 상기 실시간 데이터와 일치하는 타깃 머신 러닝 모델을 선택하는 단계;

상기 취득된 데이터를 상기 타깃 머신 러닝 모델에 입력하여, 상기 컴퓨터 시스템에 인스톨된 복수의 애플리케이션의 중요도를 순위화하는 단계; 및

상기 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계를 포함하는 방법.

청구항 2

제1항에 있어서,

상기 취득된 데이터는 상기 컴퓨터 시스템의 시스템 시간을 포함하고;

상기 실시간 데이터와 일치하는 타깃 머신 러닝 모델을 선택하는 단계는:

상기 컴퓨터 시스템의 시스템 시간에 기초하여, 상기 컴퓨터 시스템이 현재 위치된 기간을 결정하는 단계; 및

상기 컴퓨터 시스템이 현재 위치된 상기 기간에 기초한 대응관계로부터, 상기 컴퓨터 시스템이 현재 위치된 상기 기간에 대응하는 타깃 머신 러닝 모델을 결정— 상기 대응관계는 복수의 기간 및 상기 복수의 기간에 제각기 대응하는 복수의 머신 러닝 모델을 포함함 —하는 단계를 포함하는

방법.

청구항 3

제1항에 있어서,

상기 취득된 데이터는 상기 컴퓨터 시스템의 현재 위치 데이터를 포함하고;

상기 실시간 데이터와 일치하는 타깃 머신 러닝 모델을 선택하는 단계는:

상기 컴퓨터 시스템의 현재 위치 데이터에 기초하여, 상기 컴퓨터 시스템이 현재 위치된 의미론적 위치를 결정하는 단계; 및

상기 컴퓨터 시스템이 현재 위치된 상기 의미론적 위치에 기초한 대응관계로부터, 상기 컴퓨터 시스템이 현재 위치된 상기 의미론적 위치에 대응하는 타깃 머신 러닝 모델을 결정— 상기 대응관계는 복수의 의미론적 위치 및 상기 복수의 의미론적 위치에 제각기 대응하는 복수의 머신 러닝 모델을 포함함 —하는 단계를 포함하는

방법.

청구항 4

제1항 내지 제3항 중 어느 한 항에 있어서,

애플리케이션 데이터 및 상기 컴퓨터 시스템의 관련 데이터를 수집 및 저장— 상기 애플리케이션 데이터는 상기 애플리케이션의 식별자 및 상기 애플리케이션이 사용된 시간을 포함하고, 상기 컴퓨터 시스템의 관련 데이터는 이하의 데이터, 즉 상기 애플리케이션이 사용된 시간에 상기 컴퓨터 시스템의 시간, 스테이터스 데이터, 및 위치 데이터 중 적어도 하나를 포함함 —하는 단계를 더 포함하는

방법.

청구항 5

제4항에 있어서,

과거의 기간에 수집 및 저장된 상기 애플리케이션 데이터에 기초하여, 계산을 통해, 복수의 애플리케이션의 애플리케이션 시퀀스 특징 데이터를 취득하는 단계;

상기 애플리케이션 데이터를, 또는 상기 애플리케이션 데이터 및 상기 컴퓨터 시스템의 관련 데이터를 분류 모델에 입력하여, 상기 애플리케이션을 사용하기 위한 규칙과 관련되는 복수의 카테고리를 취득— 상기 복수의 카테고리는 일차원 카테고리 또는 다차원 카테고리이고, 임의의 두 카테고리는 제각기 두 가지의 상이한 규칙에 대응하고 있음 —하는 단계; 및

상기 복수의 카테고리 각각에 대한 머신 러닝 모델을 트레이닝— 상기 머신 러닝 모델은 애플리케이션들의 중요도를 순위화하는 데 사용되고, 상기 트레이닝의 입력은 상기 애플리케이션이 사용된 시간 및 상기 애플리케이션 시퀀스 특징 데이터를 포함하고, 상기 트레이닝의 입력은 상기 컴퓨터 시스템의 관련 데이터 중 적어도 하나를 더 포함함 —하는 단계를 더 포함하는

방법.

청구항 6

제4항 또는 제5항에 있어서,

상기 수집 단계는 이하의 이벤트들, 즉 포그라운드-백그라운드 전환 이벤트, 애플리케이션 설치 이벤트, 애플리케이션 제거 이벤트, 상기 컴퓨터 시스템의 상기 스테이터스 데이터의 변경에 의해 야기되는 통지 이벤트, 또는 상기 컴퓨터 시스템의 상기 위치 데이터의 변경에 의해 야기되는 통지 이벤트 중 하나 이상의 이벤트가 검출될 경우에 개시되는

방법.

청구항 7

제5항 또는 제6항에 있어서,

상기 복수의 카테고리는 시간 차원으로 분류된 복수의 카테고리인

방법.

청구항 8

제7항에 있어서,

상기 복수의 카테고리는 근무 기간 및 비-근무 기간을 포함하는

방법.

청구항 9

제1항 내지 제8항 중 어느 한 항에 있어서,

이하의 이벤트들, 즉 포그라운드-백그라운드 전환 이벤트, 애플리케이션 설치 이벤트, 애플리케이션 제거 이벤트, 상기 컴퓨터 시스템의 상기 스테이터스 데이터의 변경에 의해 야기되는 통지 이벤트, 또는 상기 컴퓨터 시스템의 상기 위치 데이터의 변경에 의해 야기되는 통지 이벤트 중 하나 이상의 이벤트가 검출될 경우에 상기 데이터를 취득하는 단계를 개시하는 단계를 포함하는

방법.

청구항 10

제1항 내지 제9항 중 어느 한 항에 있어서,

상기 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계 전에, 상기 방법은:

보호될 필요가 있는 애플리케이션의 수량 N 을 취득— N 의 값은 이하의 조건, 즉 일정 기간에 모든 애플리케이션을 사용한 횟수의 합에 대한 과거에 가장 빈번하게 사용된 N 개의 애플리케이션을 사용한 횟수의 비가 미리 설정된 제1 임계치보다 크고, N 은 0보다 크고 M 이하의 정수인 조건을 충족함 —하는 단계를 더 포함하고;

상기 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계는 N 및 상기 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계를 포함하는

방법.

청구항 11

제10항에 있어서,

상기 N 및 상기 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계는:

상기 중요도 순위화의 결과로부터, 상위에 순위화된 $N1$ 개의 애플리케이션을 식별하는 단계, 및 상기 $N1$ 개의 애플리케이션 및 다른 나머지 애플리케이션에 대한 리소스 관리를 수행— $N1$ 은 N 이하의 양의 정수임 —하는 단계를 포함하는

방법.

청구항 12

제10항에 있어서,

새롭게インストール된 애플리케이션들의 가중치에 기초하여 상기 새롭게インストール된 애플리케이션들의 중요도를 순위화하는 단계, 및 상기 새롭게インストール된 애플리케이션들로부터 상위에 순위화된 $N2$ 개의 새롭게インストール된 애플리케이션을 선택— 상기 새롭게インストール된 애플리케이션들이 상기 컴퓨터 시스템에インストール된 시간은 미리 설정된 제2 임계치보다 작음 —하는 단계를 더 포함하고;

그에 상응하여, 상기 N 및 상기 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계는:

상기 중요도 순위화의 결과로부터, 상위에 순위화된 N 개 내지 $N2$ 개의 애플리케이션을 식별하는 단계, 및 상기 N 개 내지 $N2$ 개의 애플리케이션 및 상기 $N2$ 개의 새롭게インストール된 애플리케이션에 대한 리소스 관리를 수행하거나, 또는 다른 나머지 애플리케이션에 대한 리소스 관리를 수행하는 단계를 포함하는

방법.

청구항 13

제12항에 있어서,

상기 새롭게 인스톨된 애플리케이션들의 가중치에 기초하여 상기 새롭게 인스톨된 애플리케이션들의 중요도를 순위화하는 단계는:

사용 가능성 가중치 및 시간 감쇠 가중치에 기초하여 각각의 새롭게 인스톨된 애플리케이션의 스코어를 계산— 높은 스코어를 갖는 새롭게 인스톨된 애플리케이션의 중요도는 낮은 스코어를 갖는 새롭게 인스톨된 애플리케이션의 중요도보다 높음 —하는 단계를 포함하고,

상기 사용 가능성 가중치는 상기 새롭게 인스톨된 애플리케이션이 최근에 사용되었는지의 여부를 반영하는 데 사용되고, 상기 시간 감쇠 가중치는 현재 시간과 상기 애플리케이션이 인스톨된 시간 사이의 시간차를 반영하는 데 사용되는

방법.

청구항 14

제1항 내지 제13항 중 어느 한 항에 있어서,

상기 중요도 순위화의 결과에 기초하여 리소스 관리를 수행하는 단계는, 이하의 관리 조치들, 즉

높은 중요도를 갖는 식별된 애플리케이션에 대하여 리소스를 예비할당하는 조치;

특정 시간이 종료될 때까지 낮은 중요도를 갖는 식별된 애플리케이션을 일시적으로 정지시키는 조치; 및

각각의 CPU에 대하여 VIP 큐를 생성— 상기 VIP 큐는 높은 중요도를 갖는 애플리케이션의 태스크를 포함하고, 상기 VIP 큐에서의 각각의 태스크의 실행은 상기 CPU의 다른 실행 큐보다 우선함 —하는 조치 중 하나 이상의 조치를 포함하는

방법.

청구항 15

제1항 내지 제14항 중 어느 한 항에 있어서,

상기 애플리케이션 시퀀스 특징 데이터는 k1개의 최근에 사용된 애플리케이션, 상기 포그라운드 애플리케이션 중에서 k2개의 가장 가능성 있는 전단(pre-order) 애플리케이션, 및 상기 포그라운드 애플리케이션 중에서 k3개의 가장 가능성 있는 후단(post-order) 애플리케이션을 포함하고, 상기 k1, k2, 및 k3은 모두 양의 정수인

방법.

청구항 16

제1항 내지 제15항 중 어느 한 항에 있어서,

상기 현재 위치 데이터는 의미론적 위치 데이터인

방법.

청구항 17

제1항 내지 제16항 중 어느 한 항에 있어서,

상기 현재 스테이터스 데이터는 이하의 데이터, 즉 네트워크 연결 또는 네트워크 연결해제를 나타내는 데이터, 헤드셋 연결 또는 연결해제를 나타내는 데이터, 충전 케이블 연결 또는 연결해제를 나타내는 데이터, 및 블루

투스 연결 또는 연결해제를 나타내는 데이터 중 하나 이상의 데이터인 방법.

청구항 18

프로세서 및 메모리를 포함하는 단말 장치로서, 상기 메모리는 컴퓨터 판독 가능 명령어를 저장하도록 구성되고, 상기 프로세서는 상기 메모리에 저장된 상기 컴퓨터 판독 가능 명령어를 판독하여 제1항 내지 제17항 중 어느 한 항에 기재된 방법을 수행하도록 구성되는 단말 장치.

청구항 19

애플리케이션을 일시적으로 정지시키기 위한 방법으로서,

특정 이벤트가 검출될 경우에 일부 애플리케이션들을 일시적으로 정지시키는 단계; 및

특정 기간이 종료될 경우 또는 특정 이벤트의 종료가 검출될 경우에 상기 정지된 애플리케이션들의 전부 또는 일부를 정지해제하거나, 또는 상기 정지된 애플리케이션들 중 하나 이상의 애플리케이션의 실행 환경의 변경이 검출될 경우에 상기 정지된 애플리케이션들 중 하나 이상의 애플리케이션을 정지해제하는 단계를 포함하는

방법.

청구항 20

제19항에 있어서,

상기 특정 이벤트는 필요한 리소스의 양이 증가됨을 나타내는 이벤트인

방법.

청구항 21

제19항 또는 제20항에 있어서,

상기 특정 이벤트는 이하의 이벤트, 즉 애플리케이션 기동 이벤트, 촬영 이벤트, 갤러리 줌 작업 이벤트, 슬라이딩 이벤트, 및 스크린-온/오프 이벤트 중 적어도 하나를 포함하는

방법.

청구항 22

제19항 내지 제21항 중 어느 한 항에 있어서,

상기 일시적으로 정지시키는 단계는 타이머를 설정하는 것에 의해 구현되고, 상기 타이머의 지속기간은 상기 특정 기간으로 설정되는

방법.

청구항 23

제19항 내지 제22항 중 어느 한 항에 있어서,

상기 일시적으로 정지되는 일부 애플리케이션들은 모든 백그라운드 애플리케이션을 포함하거나, 또는 상기 일시적으로 정지되는 일부 애플리케이션들은 낮은 중요도를 갖는 애플리케이션을 포함하고, 애플리케이션의 중요도

는 해당 애플리케이션의 사용 이력, 머신 러닝 알고리즘, 및 시스템의 현재 시나리오 데이터에 기초하여 취득되는 방법.

청구항 24

제19항 내지 제23항 중 어느 한 항에 있어서,

상기 정지된 애플리케이션들 중 하나 이상의 애플리케이션의 실행 환경의 변경이 검출되는 것은 이하의 이벤트, 즉 상기 정지된 애플리케이션이 포그라운드로 전환되는 이벤트, 상기 정지된 애플리케이션이 종료되는 이벤트, 상기 정지된 애플리케이션이 비동기 메시지를 수신하는 이벤트, 및 상기 정지된 애플리케이션의 통지 메시지가 사용자에게 의해 탭핑되는 이벤트 중 어느 하나 이상의 이벤트를 검출하는 것을 포함하는

방법.

청구항 25

프로세서 및 메모리를 포함하는 단말 장치로서, 상기 메모리는 컴퓨터 판독 가능 명령어를 저장하도록 구성되고, 상기 프로세서는 상기 메모리에 저장된 상기 컴퓨터 판독 가능 명령어를 판독하여 제19항 내지 제24항 중 어느 한 항에 기재된 방법을 수행하도록 구성되는 단말 장치.

청구항 26

컴퓨터 시스템에서 태스크를 실행하기 위한 방법으로서, 상기 방법은 컴퓨터 시스템에 적용되고, 상기 컴퓨터 시스템은 복수의 물리적 코어를 포함하고, 각각의 물리적 코어는 하나의 제1 큐 및 하나의 제2 큐에 대응하고, 상기 제1 큐 및 상기 제2 큐는 각각 상기 물리적 코어에 의해 실행될 하나 이상의 태스크를 포함하고, 적어도 하나의 물리적 코어는 이하의 단계, 즉,

상기 제1 큐에서의 모든 태스크의 실행이 완료될 때까지 상기 제1 큐에서의 태스크를 취득 및 실행하고 나서, 상기 제2 큐에서의 태스크를 취득 및 실행하는 단계를 수행하는

방법.

청구항 27

제26항에 있어서,

대기 시간이 특정 임계치를 초과한 태스크가 상기 제1 큐에 존재하는지의 여부를 모니터링하고, 대기 시간이 상기 특정 임계치를 초과한 태스크가 존재하면, 해당 태스크를 다른 물리적 코어에 대응하는 제1 큐로 이동시키는 단계를 더 포함하는

방법.

청구항 28

제26항 또는 제27항에 있어서,

상기 태스크를 다른 물리적 코어에 대응하는 제1 큐로 이동시키는 단계는, 상기 태스크가 이동 가능하다고 결정된 후에, 상기 태스크를 상기 다른 물리적 코어에 대응하는 제1 큐로 이동시키는 단계를 포함하는

방법.

청구항 29

제26항 내지 제28항 중 어느 한 항에 있어서,

상기 제1 큐에서의 태스크는 중요한 태스크 및 상기 중요한 태스크가 의존하는 태스크를 포함하는 방법.

청구항 30

제29항에 있어서,

상기 중요한 태스크는 사용자 경험에 영향을 미치는 태스크인 방법.

청구항 31

제29항 또는 제30항에 있어서,

상기 중요한 태스크는 높은 중요도를 갖는 애플리케이션의 태스크이고, 애플리케이션의 중요도는 해당 애플리케이션의 사용 이력, 머신 러닝 알고리즘, 및 시스템의 현재 시나리오 데이터에 기초하여 취득되는 방법.

청구항 32

제29항 내지 제31항 중 어느 한 항에 있어서,

종속성은 이하의 종속성 관계, 즉 데이터 종속성, 잠금 종속성, 및 바인더 서비스 종속성 중 적어도 하나를 포함하는 방법.

청구항 33

프로세서 및 메모리를 포함하는 단말 장치로서, 상기 메모리는 컴퓨터 판독 가능 명령어를 저장하도록 구성되고, 상기 프로세서는 상기 메모리에 저장된 상기 컴퓨터 판독 가능 명령어를 판독하여 제26항 내지 제32항 중 어느 한 항에 기재된 방법을 수행하도록 구성되는 단말 장치.

발명의 설명

기술 분야

[0001] 본원은 컴퓨터 운영 체제 분야에 관한 것으로, 특히 운영 체제에 배치되는 애플리케이션들의 중요도를 순위화하고, 중요도 순위에 기초하여 리소스 관리를 수행하기 위한 방법, 장치, 시스템 등에 관한 것이다.

배경 기술

[0002] 스마트폰 운영 체제를 포함하는 컴퓨터 운영 체제의 주된 과제는 컴퓨터 운영 체제에서 실행되는 다양한 애플리케이션에 대한 리소스 할당을 유지하는 것이다. 컴퓨터 운영 체제는, 타임 슬라이스 형태로 표현되는 프로세서(예를 들어, 중앙 처리 장치, CPU) 컴퓨팅 리소스, 메모리 페이지 형태로 표현되는 메모리 리소스, 및 대역폭 형태로 표현되는 입력/출력(Input/Output, I/O) 리소스와 같은 복수 타입의 리소스를 포함한다. 사용자의 현재의 작업을 지원하는 리소스가 제시간에 공급될 수 없을 경우에는, 운영 체제가 중지된다. 따라서, 운영 체제의 중지 여부에 영향을 미치는 핵심 요인이 리소스 스케줄링 정책이고, 리소스 스케줄링 정책이 적절한지의 여부를

결정하는 가장 중요한 조건은 리소스 스케줄링 정책이 중요한 애플리케이션과 중요하지 않은 애플리케이션을 식별해서 리소스를 가능한 한 적절하게 할당하는지의 여부이다.

- [0003] 스마트폰 운영 체제인 안드로이드(Android)가 예시로서 사용된다. 애플리케이션들은 포그라운드 애플리케이션, 백그라운드 애플리케이션, 및 기동되지 않은 애플리케이션으로 분류될 수 있다. 일반적으로, 사용자는 포그라운드 애플리케이션을 사용하는 과정에서 중지된다는 느낌을 받을 수 있다. 따라서, 포그라운드 애플리케이션이 상대적으로 더욱 중요하다. 따라서, 사용자의 관점에서, 안드로이드 시스템이 중지되는 주된 이유는, 애플리케이션들이 순차적으로 실행되지 않고, 그 결과 포그라운드 애플리케이션 또는 서비스가 사용될 때 필요한 리소스가 보장될 수 없다는 데 있다. 기존의 방법에 있어서는, 포그라운드 애플리케이션과 백그라운드 애플리케이션이 식별될 수 있고, 더 많은 리소스가 포그라운드 애플리케이션에 적절히 할당되지만, 이러한 할당은 상대적으로 고정되어 있다. 그러나, 실시간 시나리오에 있어서는, 특히, 리소스가 불충분할 경우, 보다 많은 자원을 포그라운드 애플리케이션 또는 중요한 애플리케이션에 일시적으로 어떻게 제공하여 이들 중요한 애플리케이션의 실행을 보장할지가 현재 중요하지 않은 애플리케이션에 의해 점유되는 일부 리소스를 어떻게 해제할지의 문제와 관련되어 있다. 이러한 문제를 해결하는 과정에서, 긴급하게 해결해야 할 문제는 중요한 애플리케이션을 일시적으로 가능하게 하기 위해 리소스가 해제되어야 하는 애플리케이션을 결정하는 것이다.

발명의 내용

- [0004] 본원은 리소스 관리 방법, 이 방법이 적용되는 단말 장치 등을 제공한다. 이 방법은, 현재의 시나리오에서 애플리케이션의 중요도를 식별하고, 또한 그에 상응하여 리소스를 스케줄링해서 더욱 중요한 애플리케이션에 대한 리소스 공급을 보장하도록, 애플리케이션 중요도 순위화 방법 및 리소스 스케줄링 방법을 포함하고, 이로써 시스템 중지가 어느 정도 회피되고, 사용자 경험이 향상된다.
- [0005] 제1 양태에 따르면, 본원은 리소스 관리 방법을 제공한다. 이 방법은 컴퓨터 시스템, 예를 들어, 단말 장치에 적용될 수 있다. 단말 장치는 데이터를 취득하고, 데이터는 현재의 포그라운드 애플리케이션들과 관련되는 애플리케이션 시퀀스 특징 데이터 및 이하의 실시간 데이터, 즉 컴퓨터 시스템의 시스템 시간, 컴퓨터 시스템의 현재 스테이터스 데이터, 및 컴퓨터 시스템의 현재 위치 데이터 중 적어도 하나를 포함한다. 단말 장치는, 실시간 데이터 중 적어도 하나의 데이터에 기초하여 복수의 머신 러닝 모델로부터, 실시간 데이터와 일치하는 타깃 머신 러닝 모델을 선택하고, 본 명세서에서 복수의 머신 러닝 모델은 상이한 애플리케이션 사용 규칙들에 대응하고 있다. 단말 장치는 취득된 모든 데이터를 타깃 머신 러닝 모델에 입력하여, 타깃 머신 러닝 모델을 사용함으로써, 컴퓨터 시스템에 인스톨된 복수의 애플리케이션의 중요도를 순위화한다. 중요도 순위화의 결과는 단말 장치가 리소스 관리를 수행하기 위한 의사결정 요인들 중 하나로서 사용될 수 있다. 타깃 머신 러닝 모델은 대안적으로 실시간 데이터 이외의 취득된 데이터를 사용해서 결정될 수 있다.
- [0006] 복수의 머신 러닝 모델이 설정되고, 수집된 데이터에 기초하여 복수의 머신 러닝 모델 중에서 단말 장치의 현재의 시나리오와 가장 관련된 머신 러닝 모델이 결정되고, 해당 머신 러닝 모델에 기초하여 애플리케이션 중요도 순위화가 결정되므로, 애플리케이션 중요도 순위화의 결과는 단말 장치의 현재의 시나리오에 더 부합하게 되고, 즉, 중요도 순위화의 실시간 정확도가 더 높아진다.
- [0007] 또한, 단일 데이터 또는 소량의 데이터가 수집되는 종래기술과 비교하면, 본원에서 제공되는 애플리케이션 순위화는 단말 장치의 사용과 관련되는 복수 타입의 수집된 데이터에 기초하고, 데이터 다양성이 또한 애플리케이션 순위화의 정확도를 향상시킬 수 있다.
- [0008] 일부 실시형태에 있어서, 애플리케이션 시퀀스 특징 데이터는 복수의 애플리케이션을 사용하기 위한 시간 시퀀스의 데이터를 나타내는 데 사용된다. 구체적으로, 애플리케이션 시퀀스 특징 데이터는 k1개의 최근에 사용된 애플리케이션, 포그라운드 애플리케이션들 중에서 k2개의 가장 가능성 있는 전단(pre-order) 애플리케이션, 및 포그라운드 애플리케이션들 중에서 k3개의 가장 가능성 있는 후단(post-order) 애플리케이션을 포함할 수 있고, k1, k2, 및 k3은 모두 양의 정수이다.
- [0009] 일부 실시형태에 있어서, 단말 장치는, 컴퓨터 시스템의 시스템 시간에 기초하여, 컴퓨터 시스템이 현재 위치한 기간을 결정하고; 컴퓨터 시스템이 현재 위치한 기간에 기초한 대응관계로부터, 컴퓨터 시스템이 현재 위치한 기간에 대응하는 타깃 머신 러닝 모델을 결정하고, 대응관계는 복수의 기간 및 복수의 기간에 제각기 대응하는 복수의 머신 러닝 모델을 포함한다.
- [0010] 일부 실시형태에 있어서, 단말 장치는, 컴퓨터 시스템의 현재 위치 데이터에 기초하여, 컴퓨터 시스템이 현재 위치한 의미론적 위치를 결정하고; 이어서, 컴퓨터 시스템이 현재 위치한 의미론적 위치에 기초한 대응관계로부터

터, 컴퓨터 시스템이 현재 위치한 의미론적 위치에 대응하는 타깃 머신 러닝 모델을 결정하고, 대응관계는 복수의 의미론적 위치 및 복수의 의미론적 위치에 제각기 대응하는 복수의 머신 러닝 모델을 포함한다.

- [0011] 전술한 내용은 타깃 머신 러닝 모델을 결정하는 두 가지 방식이다. 복수의 타깃 머신 러닝 모델이 제각기 사용자에게 의해 애플리케이션을 사용하기 위한 복수의 규칙에 대응하고 있다. 복수의 사용 규칙은 하나의 차원, 예를 들어, 시스템 시간 또는 현재 위치 데이터에 기초하여 분류될 수 있거나, 또는 복수의 차원에 기초하여 분류될 수 있다.
- [0012] 일부 실시형태에 있어서, 단말 장치는 적어도 2개의 실시간 데이터에 기초하여 타깃 머신 러닝 모델을 결정한다. 전술한 두 가지 구현예와 달리, 이들 실시형태에 있어서, 타깃 머신 러닝 모델에 대응하는 사용 규칙은 복수의 차원에 기초하여 수행된 분류의 결과이다. 예를 들어, 사용자의 사용 규칙들은 시간 차원 및 지리적 위치 차원으로 4가지 타입으로, 즉 근무 시간 및 회사, 근무 시간 및 출장(회사는 아님), 비-근무 시간 및 집, 그리고 비-근무 시간 및 여흥 장소(집은 아님)로 분류된다. 4가지 타입의 사용 규칙은 제각기 상이한 특징을 나타내고, 그에 따라 각각이 하나의 머신 러닝 모델에 대응하고 있다. 단말 장치는, 실시간 데이터에 기초하여, 현재의 시나리오에 가장 부합하는 머신 러닝 모델을 결정한다.
- [0013] "사용 규칙(use regularity)"을 측정하는 방법은 본원에서 제한되지 않는다. 사용 규칙들이 상이하다는 것은 현재의 측정 방법의 관점에서 상이한 사용 규칙들이 상이한 특징들을 나타낸다는 것을 의미한다.
- [0014] 일부 실시형태에 있어서, 단말 장치는, 애플리케이션 사용 이력에 기초하여, 현재의 시나리오에서 사용자에게 의해 빈번하게 사용된 애플리케이션들의 수량 N 을 더 예측하고 나서, 중요도 순위에 기초하여, 상위에 순위화된 N 개의 애플리케이션을 결정할 수 있다. 이렇게 해서, 단말 장치는 리소스 관리를 수행할 때 N 개의 애플리케이션을 중요한 애플리케이션으로서 사용할 수 있다. 경우에 따라, 단말 장치는 N 개의 애플리케이션에 대하여 리소스를 예비할당할 수 있거나, 또는 N 개의 애플리케이션을 보호하기 위한 일부 조치를 취할 수 있다.
- [0015] 모든 애플리케이션에 대한 중요도 순위화의 결과가 존재하더라도, 단말 장치가 리소스 관리 동안 일부 애플리케이션을 선택할 때 선택 대상 애플리케이션들의 수량을 결정하는 것은 어려운 일이다. 사용자에게 의해 빈번하게 사용되는 애플리케이션의 수량 N 을 예측함으로써, 단말 장치는 리소스 관리를 더욱 합목적적이고 명확하게 수행할 수 있다. 또한, N 개의 애플리케이션은 실제로 중요도가 매우 높은 애플리케이션이고, 이는 리소스 관리를 더 적합하게 만든다.
- [0016] 일부 실시형태에 있어서, 단말 장치는 중요도가 상위에 순위화된 N 개의 애플리케이션(또는 더 많거나 적은 애플리케이션)을 결정하고, 결정된 애플리케이션들에 대하여 리소스를 예비할당하거나, 또는 나머지 다른 애플리케이션을 일시적으로 정지시키거나, 또는 각각의 CPU에 대한 VIP 큐를 생성한다. VIP 큐는 이들 결정된 애플리케이션의 태스크(프로세스 또는 스레드)를 포함하고, VIP 큐에서의 각각의 태스크의 실행은 CPU의 다른 실행 큐보다 우선한다.
- [0017] 본원에서 제공되는 방법을 사용해서 애플리케이션들이 순위화될 경우, 해당 애플리케이션들의 사용에 관한 이력 데이터가 수집될 필요가 있다. 그러나, 새롭게 인스톨된 애플리케이션의 경우, 해당 애플리케이션은 애플리케이션의 사용에 관한 이력 데이터의 양이 너무 적기 때문에 하위에 순위화될 수 있다. 그러나, 이는 새롭게 인스톨된 애플리케이션의 실제 중요도를 정확하게 나타낼 수 없다.
- [0018] 제2 양태에 따르면, 본원은 새롭게 인스톨된 애플리케이션들의 중요도를 순위화하기 위한 방법을 더 제공하고, 이는 새롭게 인스톨된 애플리케이션들의 중요도를 보상하는 것에 상당한다. 단말 장치는 새롭게 인스톨된 애플리케이션들의 가중치에 기초하여 새롭게 인스톨된 애플리케이션들의 중요도를 순위화하고, 새롭게 인스톨된 애플리케이션들로부터 상위에 순위화된 N_2 개의 새롭게 인스톨된 애플리케이션을 선택하고, 새롭게 인스톨된 애플리케이션들이 컴퓨터 시스템에 인스톨된 시간은 미리 설정된 제2 임계치보다 작다. 그에 상응하여, 리소스 관리를 수행할 경우, 단말 장치는 새롭게 인스톨된 애플리케이션들의 순위를 또한 고려할 수 있다. 예를 들어, 리소스를 예비할당할 경우, 애플리케이션들의 수량이 제한되면, 단말 장치는 중요도 순위화 결과에서 상위에 순위화된 일부 애플리케이션을 고려할 뿐만 아니라, 새롭게 인스톨된 애플리케이션들의 순위화 결과에서 상위에 순위화된 새롭게 인스톨된 애플리케이션도 고려한다. 다른 타입의 리소스 관리도 유사하다.
- [0019] 이는 새롭게 인스톨된 애플리케이션들 중에서 사용자에게 상대적으로 중요한 애플리케이션이 리소스 관리 동안 무시되는 것을 회피하고, 리소스 관리의 효율성이 더욱 향상된다.
- [0020] 일부 실시형태에 있어서, 단말 장치는 사용 가능성 가중치 및 시간 감쇠 가중치에 기초하여 각각의 새롭게 인스톨된 애플리케이션의 스코어를 계산하고, 높은 스코어를 갖는 새롭게 인스톨된 애플리케이션의 중요도는 낮은

스코어를 갖는 새롭게 인스톨된 애플리케이션의 중요도보다 높다. 사용 가능성 가중치는 새롭게 인스톨된 애플리케이션이 최근에 사용되었는지의 여부를 반영하는 데 사용되고, 시간 감쇠 가중치는 현재 시간과 애플리케이션이 인스톨된 시간 사이의 시간차를 반영하는 데 사용된다.

- [0021] 제3 양태에 따르면, 본원은 데이터 수집 방법 및 수집된 데이터에 기초하여 모델을 트레이닝하기 위한 방법을 더 제공하고, 이들 방법은 다른 실시형태에서 제공되는 애플리케이션 순위화, 리소스 관리 등을 지원하는 데 사용될 수 있다.
- [0022] 단말 장치는 애플리케이션 데이터 및 컴퓨터 시스템의 관련 데이터를 수집 및 저장하고, 애플리케이션 데이터는 애플리케이션의 식별자 및 애플리케이션이 사용된 시간을 포함하고, 컴퓨터 시스템의 관련 데이터는 이하의 데이터 중 적어도 하나, 즉 애플리케이션이 사용된 시간에 컴퓨터 시스템의 시간, 스테이터스 데이터, 및 위치 데이터 중 적어도 하나를 포함한다.
- [0023] 또한, 단말 장치는, 계산을 통해, 과거의 기간에 수집 및 저장된 애플리케이션 데이터에 기초하여 복수의 애플리케이션의 애플리케이션 시퀀스 특징 데이터를 취득하고, 애플리케이션 데이터를, 또는 애플리케이션 데이터 및 컴퓨터 시스템의 관련 데이터를 엔트로피 생성 모델과 같은 분류 모델에 입력하여, 애플리케이션을 사용하기 위한 규칙과 관련되는 복수의 카테고리를 취득— 임의의 2가지 카테고리가 제각기 2가지 상이한 규칙에 대응함—하고; 복수의 카테고리 각각에 대한 머신 러닝 모델을 트레이닝— 머신 러닝 모델은 애플리케이션들의 중요도를 순위화하는 데 사용되고, 트레이닝의 입력은 애플리케이션이 사용된 시간, 애플리케이션 시퀀스 특징 데이터, 및 컴퓨터 시스템의 관련 데이터 중 적어도 하나를 포함함—한다.
- [0024] 일부 실시형태에 있어서, 모델 트레이닝 프로세스는 서버 측에서 수행될 수 있다. 단말 장치는 수집된 데이터를 서버에 송신하고, 서버는 모델을 트레이닝한다. 서버는 트레이닝된 모델을 로컬에 저장할 수 있거나, 또는 트레이닝된 모델을 단말 장치에 반환할 수 있다. 모델이 서버 측에 저장되면, 단말 장치는 중요도 순위화를 수행할 때 모델을 서버에 신청할 수 있다. 또한, 중요도 순위화는 대안적으로 서버 측에서 수행될 수 있으며, 단말 장치는 단지 순위화 결과를 저장하거나 또는 순위화 결과를 사용할 때 순위화 결과를 서버에 신청하면 된다.
- [0025] 제4 양태에 따르면, 본원은 리소스 관리 방법을 더 제공하고, 이 방법은 컴퓨터 시스템, 예를 들어, 단말 장치에 적용될 수 있다. 특정 이벤트를 검출할 경우, 단말 장치는 특정 기간이 종료될 때까지 일부 애플리케이션을 일시적으로 정지시키고 나서, 정지된 애플리케이션들의 전부 또는 일부를 정지해제한다. 특정 이벤트는 필요한 리소스량이 증가한다는 것을 나타내는 이벤트, 예를 들어, 애플리케이션 기동 이벤트, 촬영 이벤트, 갤러리 줌 작업 이벤트, 슬라이딩 이벤트, 및 스크린 온/오프 이벤트이다.
- [0026] 특정 기간이 종료된다는 것은 단지 하나의 정지해제 조건일 뿐이고, 다른 조건은 특정 이벤트가 종료됨이 검출된다는 것일 수 있다.
- [0027] 또한, 일부 긴급 이벤트가 발생할 수 있으며, 이들 이벤트는 특정 기간 및 특정 이벤트가 종료되기 전에 발생한다. 이러한 긴급 이벤트가 발생하면, 긴급 이벤트와 관련되는 애플리케이션은 스케줄에 앞서 정지해제될 필요가 있다.
- [0028] 필요한 순간적인 리소스의 양이 많을 경우, 사용자에 의해 감지된 상대적으로 많은 리소스량을 필요로 하는 애플리케이션들에 대한 리소스 공급을 보장하기 위해, 중요도가 상대적으로 낮은 일부 애플리케이션이 일시적으로 정지되어 일부 리소스를 해제하고, 이로써 이들 애플리케이션에 대한 중지가 회피되고, 사용자 경험이 향상된다. 일시적인 정지는 상대적으로 짧은 기간 동안 애플리케이션을 정지하고 나서, 애플리케이션을 해제하는 것이다. 그러나, 종래기술에 있어서, 장기간 동안 사용되지 않은 애플리케이션은 일반적으로 장기간 동안 정지되고 나서, 애플리케이션이 요청될 때 정지해제된다. 전술한 내용은 모두 정지 방식이기는 하지만, 이 정지 방식들은 적어도 사용 시나리오 및 구체적인 정지 방식에 관해서는 상이하다.
- [0029] 일부 실시형태에 있어서, 단말 장치는 타이머를 설정함으로써 일시적인 정지를 구현하고, 타이머의 지속기간은 특정 기간으로 설정된다. 이렇게 해서, 코드 변경량이 비교적 작다.
- [0030] 일부 실시형태에 있어서, 일시적으로 정지된 일부 애플리케이션은 모든 백그라운드 애플리케이션 또는 사용자에게 의해 감지될 수 없는 모든 백그라운드 애플리케이션을 포함한다. 일부 다른 실시형태들에 있어서, 일시적으로 정지된 일부 애플리케이션은 중요도가 낮은 애플리케이션을 포함한다. 애플리케이션의 중요도는 애플리케이션의 사용 이력, 머신 러닝 알고리즘, 및 시스템의 현재 시나리오 데이터에 기초하여 취득된다. 구체적으로, 애플리케이션의 중요도는 위에 제공된 중요도 순위화 방법에 따라 취득될 수 있으며, 중요도가 하위에 순위화된

애플리케이션이 일시적으로 정지되도록 선택된다.

- [0031] 제5 양태에 따르면, 본원은 다른 리소스 관리 방법을 더 제공하고, 이 방법은 컴퓨터 시스템, 예를 들어, 단말 장치에 적용될 수 있다. 단말 장치는 복수의 물리적 코어를 포함하고, 각각의 물리적 코어는 하나의 제1 큐 및 하나의 제2 큐에 대응하고, 제1 큐 및 제2 큐는 각각 물리적 코어에 의해 실행될 하나 이상의 태스크를 포함한다. 적어도 하나의 물리적 코어는 이하의 방법, 즉 제1 큐에서의 모든 태스크의 실행이 완료될 때까지 제1 큐에서의 태스크를 취득 및 실행하고 나서, 제2 큐에서의 태스크를 취득 및 실행하는 방법을 수행한다.
- [0032] 단말 장치는 중요한 태스크들을 실행 우선순위가 더 높은 추가적인 큐에 배치하고, 물리적 코어는 이들 중요한 태스크를 먼저 실행하여 중요한 태스크들에 대한 리소스 공급을 보장한다.
- [0033] 일부 실시형태에 있어서, 단말 장치는 대기 시간이 특정 임계치를 초과한 태스크가 제1 큐에 존재하는지의 여부를 모니터링하고, 대기 시간이 특정 임계치를 초과한 태스크가 존재하면, 단말 장치가 해당 태스크를 다른 물리적 코어에 대응하는 제1 큐로 이동시킨다.
- [0034] 리눅스(Linux) 운영 체제의 제한으로 인해, 실시간 태스크는 일반적으로 하나의 물리적 코어에서 다른 물리적 코어로 이동될 수 없다. 본 명세서에서의 중요한 태스크는 비-실시간 태스크이다. 대기 시간이 만료된 중요한 태스크가 검출될 경우, 대기 시간이 만료된 중요한 태스크는, 중요한 태스크의 지나치게 긴 대기 시간에 의해 야기되는 중지를 회피하기 위해, 다른 물리적 코어의 유휴 제1 큐로 이동된다.
- [0035] 일부 실시형태에 있어서, 제1 큐에서의 태스크는 중요한 태스크(또는 핵심 태스크라고도 함) 및 중요한 태스크가 의존하는 태스크를 포함한다. 중요한 태스크는 사용자 경험에 영향을 미치는 태스크, 또는 사용자에게 의해 감지될 수 있는 태스크, 또는 중요도가 높은 애플리케이션의 태스크이다. 애플리케이션의 중요도는 애플리케이션의 사용 이력, 머신 러닝 알고리즘, 및 시스템의 현재 시나리오 데이터에 기초하여 취득된다. 예를 들어, 태스크들 사이의 종속성 관계는 데이터 종속성, 잠금 종속성, 또는 바인더 서비스 종속성이다.
- [0036] 중요한 태스크의 실행은 중요한 태스크가 의존하는 태스크의 실행에 의존하기 때문에, 중요한 태스크 및 중요한 태스크가 의존하는 태스크는 모두 중요한 태스크의 실행 속도를 더 높이기 위해 실행 우선순위가 더 높은 제1 큐에 배치된다.
- [0037] 제6 양태에 따르면, 본원은 본원에서 제공되는 각각의 방법에 대응하는 장치를 더 제공하고, 이 장치는 해당 방법의 단계들을 구현하도록 구성되는 모듈을 포함한다. 모듈은 소프트웨어에 의해 구현될 수 있거나, 소프트웨어 및 하드웨어의 조합에 의해 구현될 수 있거나, 또는 하드웨어에 의해 구현될 수 있다.
- [0038] 제7 양태에 따르면, 본원은 프로세서 및 메모리를 포함하는 단말 장치를 더 제공한다. 메모리는 컴퓨터 판독 가능 명령어를 저장하도록 구성되고, 프로세서는 메모리에 저장된 컴퓨터 판독 가능 명령어를 판독하여 본원에서 제공되는 어느 하나 이상의 방법을 구현하도록 구성된다.
- [0039] 제8 양태에 따르면, 본원은 저장 매체를 더 제공한다. 저장 매체는 구체적으로 비휘발성 저장 매체일 수 있고, 컴퓨터 판독 가능 명령어를 저장하도록 구성된다. 하나 이상의 프로세서가 컴퓨터 판독 가능 명령어를 실행할 경우, 본원에서 제공되는 어느 하나 이상의 방법이 구현된다.
- [0040] 제9 양태에 따르면, 본원은 컴퓨터 프로그램 제품을 더 제공한다. 제품은 컴퓨터 판독 가능 명령어를 포함한다. 하나 이상의 프로세서가 컴퓨터 판독 가능 명령어를 실행할 경우, 본원에서 제공되는 어느 하나 이상의 방법이 구현된다.
- [0041] 일부 실시형태에 있어서, 당업자라면 상이한 경우들에 상이한 요건들이 존재한다는 것을 이해할 수 있을 것이기 때문에, 중요도의 구체적인 정도는 본원에서 특별히 제한되지 않는다. 본원에서 제공되는 순위화 방법의 경우, 애플리케이션의 "중요도(importance)"는 애플리케이션이 사용자에게 의해 사용될 가능성이고, 가능성이 클수록 중요도가 높다는 것을 나타낸다. 그러나, 일부 다른 실시형태들에 있어서, 예를 들어, 애플리케이션 리소스 관리 및 제어에 있어서, 애플리케이션의 "중요도"는 현재 리소스 관리 및 제어 스테이터스에 기초하여 결정될 수 있으며, 예를 들어, 애플리케이션의 중요도는 사용자가 애플리케이션을 감지하는지의 여부 등과 관련될 수 있다. 또한, 본원에서 제공되는 두 가지 리소스 관리 방법은 본원에서 제공되는 순위화 방법에 의존할 수 있거나 또는 의존하지 않을 수 있다.
- [0042] 컴퓨터 운영 체제에서 리소스들의 무질서한 스케줄링을 해결하는 열쇠는, 시스템 리소스들이 충분히 사용되는 것을 보장하기 위해, 시스템이 애플리케이션의 중요도를 실시간으로 정확하게 감지하고, 애플리케이션 중요도 순위화에 기초하여 최적의 리소스 관리 정책을 구현할 수 있도록 하는 것이다. 요컨대, 운영 체제는 사용자의

관점에서 사용자의 사용 요건을 완전히 식별하고, 해당 요건에 따라 리소스를 공급할 필요가 있다. 예를 들어, 운영 체제는 사용자에게 의해 현재 사용되는 애플리케이션에 대한 리소스를 충분히 보장하고, 사용자에게 의해 사용될 애플리케이션에 대한 리소스를 준비하고, 비효율적인 자기-기동 또는 연관된 기동을 통해 기동되는 애플리케이션과 같이, 사용자에게 의해 현재 가장 적은 관심을 받는 애플리케이션에 대한 리소스를 충분히 재활용할 필요가 있다.

[0043] 본원에서 제공되는 애플리케이션 순위화 방법에 따르면, 컴퓨터 장치(예를 들어, 지능형 단말)의 복수 타입의 정보가 실시간으로 수집되고, 애플리케이션 사용 규칙들과 관련되는 상이한 카테고리들에서의 복수의 머신 러닝 모델이 별도로 트레이닝된다. 중요도 순위화 동안, 사용자의 사용 규칙과 가장 부합하는 머신 러닝 모델이 선택되어 애플리케이션의 중요도를 실시간으로 예측하고, 이로써 애플리케이션 중요도의 식별 정확도가 향상된다.

[0044] 또한, 애플리케이션 중요도의 식별에 기초하여 리소스 관리가 적절하고 효과적으로 수행되므로, 중요한 애플리케이션에 대한 리소스 공급이 보장되고, 컴퓨터 장치를 사용하는 능숙도가 향상되고, 이로써 사용자의 사용 경험이 향상된다.

도면의 간단한 설명

[0045] 본원에서 제공되는 기술적인 해법을 보다 명료하게 설명하기 위해, 하기에서는 첨부 도면을 간략히 설명한다. 명백히, 하기의 설명에서의 첨부 도면은 단지 본원의 일부 실시형태만을 도시한다.

- 도 1은 단말 장치의 논리적 구조의 개략도이고;
- 도 2는 단말 장치에 배치되는 운영 체제의 논리적 구조의 개략도이고;
- 도 3은 지각(perception) 관리 장치의 일부 모듈의 논리적 구조의 개략도이고;
- 도 4는 리소스 관리 방법의 일반적인 개략도이고;
- 도 5는 리소스 관리 방법에서의 데이터 수집 방법의 개략적인 흐름도이고;
- 도 6은 리소스 관리 방법에서의 모델 트레이닝 방법의 개략적인 흐름도이고;
- 도 7은 하나 이상의 차원에 기초하여 애플리케이션 사용 규칙들을 분류하는 원리 및 효과의 개략도이고;
- 도 8은 리소스 관리 방법에서의 실시간 애플리케이션 순위화 방법의 개략적인 흐름도이고;
- 도 9는 리소스 관리 방법에서의 일시적인 정지 방법의 개략적인 흐름도이고;
- 도 10은 준비 상태 큐의 제1 상태의 예시적인 도면이고;
- 도 11은 준비 상태 큐의 제2 상태의 예시적인 도면이고;
- 도 12는 각각의 CPU에 대응하는 2개의 큐의 예시적인 도면이고;
- 도 13은 태스크 이동 방법의 개략적인 흐름도이고;
- 도 14는 태스크 이동과 관련되는 큐 변경의 예시적인 도면이다.

발명을 실시하기 위한 구체적인 내용

[0046] 본원의 실시형태들에 대한 이해를 돕기 위해, 본원의 실시형태들의 설명에서 사용되는 몇몇 요소를 여기서 먼저 설명한다.

[0047] 운영 체제(operating system): 운영 체제는 컴퓨터 하드웨어 및 소프트웨어 리소스를 관리하는 컴퓨터 프로그램이고, 컴퓨터 시스템의 핵심이며 초석이기도 하다. 운영 체제는 기본 트랜잭션, 예를 들어, 메모리 관리 및 구성, 시스템 리소스 공급 및 수요의 우선순위 결정, 입력 및 출력 장치 제어, 네트워크 운영, 및 파일 시스템 관리를 취급해야 한다. 운영 체제는 또한 사용자가 시스템과 상호작용하기 위한 조작 인터페이스를 제공한다. 운영 체제는 매우 다양한 형태를 갖는다. 상이한 기계들에 인스톨되는 운영 체제들은 단순하거나 복잡할 수 있으며, 휴대폰의 내장 시스템 또는 슈퍼컴퓨터의 대형 운영 체제일 수 있다. 많은 운영 체제 제조자들은 운영 체제의 범위 대하여 서로 다른 정의를 갖는다. 예를 들어, 일부 운영 체제는 그래픽 사용자 인터페이스(graphic user interface, GUI)를 통합하지만, 일부 운영 체제 각각은 커맨드-라인 인터페이스만을 사용하고, GUI를 필수적이지 않은 애플리케이션 프로그램으로서 간주한다. 일반적으로, 단말 운영 체제는 휴대폰, 태블릿

컴퓨터, 또는 POS(point of sale)와 같은 단말 상에서 실행되는 운영 체제, 예를 들어, 현재 주류인 안드로이드 또는 iOS로 간주된다.

[0048] 애플리케이션(application): 애플리케이션은 애플리케이션 소프트웨어 또는 애플리케이션 프로그램이라고도 하며, 사용자를 위해 연관 기능들, 태스크들, 또는 활동들의 그룹을 구현하도록 설계된 컴퓨터 프로그램이다. 애플리케이션은 운영 체제에 배치된다. 구체적으로, 애플리케이션은 배치를 위해 운영 체제의 시스템 소프트웨어 (예를 들어, 운영 체제)에, 예를 들어, 시스템-레벨 애플리케이션(또는 시스템 서비스라고 함)에 구속될 수 있거나; 또는 독립적으로 배치될 수 있는, 예를 들어, 현재 일반적인 텍스트 처리 애플리케이션(예를 들어, 워드(Word) 애플리케이션), 웹 브라우저 애플리케이션, 멀티미디어 플레이 애플리케이션, 또는 게임 애플리케이션일 수 있다.

[0049] 시스템 리소스(system resource): 본원에서의 시스템 리소스는 컴퓨터 시스템에서의 리소스이고, 메모리 리소스, 처리 리소스, 및 I/O 리소스 중 어느 하나 이상을 포함하지만, 이들로 제한되는 것은 아니다. 리소스는 많은 구현들에서 관리된다. 예를 들어, 리소스는 일부 애플리케이션을 종료, 정지, 또는 압축함으로써 재활용되거나, 리소스는 애플리케이션의 기동을 거부함으로써 유지되거나, 또는 리소스는 애플리케이션을 미리 로딩함으로써 애플리케이션에 예비할당된다.

[0050] 포그라운드 애플리케이션, 백그라운드 애플리케이션, 및 애플리케이션의 포그라운드-백그라운드 전환 (foreground application, background application, and foreground-background switching of an application): 포그라운드 애플리케이션은 포그라운드에서 실행되는 애플리케이션의 줄임말이다. 백그라운드 애플리케이션은 백그라운드에서 실행되는 애플리케이션의 줄임말이다. 예를 들어, 워드 애플리케이션 및 웹 브라우저 애플리케이션이 현재 기동되고 있지만, 사용자가 현재 일부 운영 체제를 작성하여 2개의 목록을 사용해서 2개의 애플리케이션을 별도로 관리한다. 리눅스 시스템에서, 애플리케이션이 포그라운드에서 백그라운드로 또는 백그라운드에서 포그라운드로 전환될 경우, 포그라운드-백그라운드 전환 이벤트가 트리거된다. 시스템은 포그라운드-백그라운드 전환 이벤트를 모니터링함으로써 애플리케이션의 포그라운드-백그라운드 전환을 감지할 수 있다. 일부 실시형태에 있어서, 백그라운드 애플리케이션들은 둔감한 백그라운드 애플리케이션 및 민감한 백그라운드 애플리케이션으로 더 분류될 수 있다. 민감한 백그라운드 애플리케이션은 일부 애플리케이션, 예를 들어, 음악 재생 애플리케이션 또는 내비게이션 애플리케이션이 백그라운드에서 실행되더라도, 해당 애플리케이션이 사용자에게 의해 여전히 감지될 수 있음을 의미한다. 사용자는 이들 두 애플리케이션이 백그라운드에서 실행되더라도 여전히 음악 또는 내비게이션 음성을 들을 수 있다.

[0051] 전단 애플리케이션 및 후단 애플리케이션(pre-order application and post-order application): 애플리케이션의 전단 애플리케이션은 해당 애플리케이션이 사용되기 전(애플리케이션이 포그라운드로 전환되기 전으로 이해될 수 있음)에 사용되는 애플리케이션이고, 애플리케이션의 후단 애플리케이션은 해당 애플리케이션이 사용된 후에 사용되는 애플리케이션이다. 애플리케이션이 사용된다는 것은 애플리케이션이 기동되거나 또는 애플리케이션이 백그라운드에서 포그라운드로 전환된다는 것을 의미하고, 애플리케이션이 기동된다는 것은 애플리케이션이 (종료로부터) 포그라운드로 전환된다는 것으로 이해될 수도 있다.

[0052] 애플리케이션 시퀀스 특징 데이터(application sequence feature data): 애플리케이션 시퀀스 특징 데이터는 복수의 애플리케이션을 사용하기 위한 시간 시퀀스를 나타내는 데 사용되는 데이터, 예를 들어, 어느 애플리케이션이 최근에 사용되었는지, 어느 애플리케이션이 애플리케이션의 전단 애플리케이션일 수 있는지, 또는 어느 애플리케이션이 애플리케이션의 후단 애플리케이션일 수 있는지의 데이터이다.

[0053] 시스템 스테이터스 데이터(system status data): 시스템 스테이터스 데이터는 줄여서 스테이터스 데이터라고도 하며, 컴퓨터 장치 또는 운영 체제의 스테이터스에 관한 정보를 나타내는 데 사용된다. 보다 구체적으로, 시스템 스테이터스 데이터는 장치의 내장 컴포넌트 또는 외장 컴포넌트의 스테이터스 데이터, 예를 들어, 네트워크 연결 스테이터스, 또는 헤드셋 또는 충전 케이블과 같은 외부 장치의 연결 스테이터스로 이해될 수 있다.

[0054] 위치 데이터 및 의미론적 위치 데이터(location data and semantic location data): 위치 데이터는 넓은 의미에서의 개념이고, 위치를 나타내는 임의의 정보는 위치 데이터, 예를 들어, 경도 및 위도로 간주될 수 있다. 경도 및 위도와 같은 상대적으로 정확한 위치 데이터에 실질적인 의미, 예를 들어, "집(home)", "회사(company)", 및 "여흥 장소(entertainment place)"가 부여될 경우, 이 위치 데이터가 의미론적 위치 데이터이다. 의미론적 위치 데이터도 일종의 위치 데이터이다.

[0055] 위치 데이터 및 시스템 스테이터스 데이터를 총괄하여 장치의 시나리오 데이터로서 이해할 수도 있다.

- [0056] 애플리케이션 중요도(application importance): 애플리케이션 중요도는 애플리케이션을 사용할 확률이거나, 또는 애플리케이션을 포그라운드로 전환할 가능성으로 이해된다. 본원에서, 애플리케이션 중요도의 순위는 애플리케이션을 사용할 확률의 순위이다. 이러한 순위의 기준은 애플리케이션을 사용할 확률의 예측이며, 애플리케이션의 사용 이력 및 일부 다른 정보에 기초한 예측이다.
- [0057] 달리 명시되지 않는 한, 본원에서 "복수(a plurality of)"는 2개 이상을 의미한다. 본원에서 언급되는 "데이터(data)" 또는 "정보(information)"의 저장 방식 또는 포맷은 제한되지 않는다.
- [0058] 본원에서 제공되는 방법은 주로 단말 장치에 적용되고, 단말 장치(일반적으로 이동 단말임)는 사용자 장비(user equipment, UE), 이동국(mobile station, MS), 이동 단말(mobile terminal) 등이라고 할 수도 있다. 선택적으로, 단말은 무선 액세스 네트워크(radio access network, RAN)를 통해 하나 이상의 코어 네트워크와 통신할 수 있다. 예를 들어, 단말은 휴대폰("셀룰러(cellular)" 폰이라고 함) 또는 이동성 특성을 갖는 컴퓨터일 수 있다. 예를 들어, 단말은 휴대폰, 소형, 핸드헬드, 컴퓨터 내장형, 또는 차량용 이동 장치일 수도 있다. 예를 들어, 단말 장치는 셀룰러 폰, 스마트폰, 랩톱 컴퓨터, 디지털 브로드캐스트 단말, 개인용 정보 단말, 휴대폰 멀티미디어 플레이어, 또는 내비게이션 시스템일 수 있다. 본원의 임의의 실시형태에서 제공되는 방법은, 이동 단말에 더하여, 퍼스널 컴퓨터, POS(point of sale), 또는 현금 자동 입출금기와 같은 고정 단말에 적용될 수도 있거나; 또는 서버와 같은 비-단말 타입의 컴퓨터 시스템에 적용될 수 있다는 점을 이해해야 한다.
- [0059] 본원에서의 단말 장치는 첨부 도면을 참조하여 더 상세하게 설명된다. "유닛(unit)" 및 "모듈(module)" 같은 후치어는 단지 설명의 편의를 위한 것이고, 이들 후치어는 서로 구별되는 의미 또는 기능을 갖지 않는다는 점에 유의해야 한다.
- [0060] 도 1은 실시형태에 따른 단말 장치의 개략적인 구성도이다. 도 1에 도시된 바와 같이, 단말 장치(100)는 무선 통신 모듈(110), 센서(120), 사용자 입력 모듈(130), 출력 모듈(140), 프로세서(150), 오디오/비디오 입력 모듈(160), 인터페이스 모듈(170), 메모리(180), 및 전원 장치(190)를 포함한다.
- [0061] 무선 통신 모듈(110)은 단말 장치(100)와 무선 통신 시스템 사이 또는 단말 장치(100)와 단말 장치(100)가 위치된 네트워크 사이에서 무선 통신을 구현할 수 있는 적어도 하나의 모듈을 포함할 수 있다. 예를 들어, 무선 통신 모듈(110)은 브로드캐스트 수신 모듈(115), 이동 통신 모듈(111), 무선 인터넷 모듈(112), 로컬 에어리어 통신 모듈(113), 및 위치(또는 위치결정) 정보 모듈(114)을 포함할 수 있다.
- [0062] 브로드캐스트 수신 모듈(115)은 외부 브로드캐스트 관리 서버로부터 브로드캐스트 채널을 통해 브로드캐스트 신호 및/또는 브로드캐스트-관련 정보를 수신할 수 있다. 브로드캐스트 채널은 위성 채널 및 지상파 채널을 포함할 수 있다. 브로드캐스트 관리 서버는 브로드캐스트 신호 및/또는 브로드캐스트-관련 정보를 생성 및 송신하는 서버일 수 있거나, 또는 사전-생성된 브로드캐스트 신호 및/또는 브로드캐스트-관련 정보를 수신하고 사전-생성된 브로드캐스트 신호 및/또는 브로드캐스트-관련 정보를 단말 장치(100)에 송신하는 서버일 수 있다. 브로드캐스트 신호는 텔레비전 브로드캐스트 신호, 라디오 브로드캐스트 신호, 및 데이터 브로드캐스트 신호 뿐만 아니라, 텔레비전 브로드캐스트 신호와 라디오 브로드캐스트 신호의 조합 형태의 신호도 포함할 수 있다. 브로드캐스트-관련 정보는 브로드캐스트 채널, 브로드캐스트 프로그램, 또는 브로드캐스트 서비스 공급자에 관한 정보일 수 있으며, 심지어 이동 통신 네트워크를 사용해서 제공될 수 있다. 브로드캐스트-관련 정보가 이동 통신 네트워크를 사용해서 제공되는 경우, 브로드캐스트-관련 정보는 이동 통신 모듈(111)에 의해 수신될 수 있다. 브로드캐스트-관련 정보는 다양한 형태로 존재할 수 있다. 예를 들어, 브로드캐스트-관련 정보는 디지털 멀티미디어 브로드캐스팅(digital multimedia broadcasting, DMB) 시스템의 전자 프로그램 가이드(electronic program guide, EPG) 형태로, 또는 디지털 비디오 브로드캐스팅-핸드헬드(digital video broadcast-handheld, DVB-H) 시스템의 전자 서비스 가이드(electronic service guide, ESG) 형태로 존재할 수 있다. 브로드캐스트 수신 모듈(115)은 다양한 브로드캐스팅 시스템을 사용해서 브로드캐스트 신호를 수신할 수 있다. 보다 구체적으로, 브로드캐스트 수신 모듈(115)은 멀티미디어 브로드캐스팅-지상파(multimedia broadcasting-terrestrial, DMB-T), 디지털 멀티미디어 브로드캐스팅-위성(digital multimedia broadcasting-satellite, DMB-S), 미디어플로(media forward link only, MediaFLO), DVB-H, 및 통합 서비스 디지털 브로드캐스팅-지상파(integrated services digital broadcasting-terrestrial, ISDB-T)와 같은 디지털 브로드캐스팅 시스템을 사용해서 브로드캐스트 신호를 수신할 수 있다. 전술한 디지털 브로드캐스팅 시스템에 더하여, 브로드캐스트 수신 모듈(115)은 브로드캐스트 신호를 제공하는 브로드캐스팅 시스템으로부터 신호를 수신할 수 있다. 브로드캐스트 수신 모듈(115)에 의해 수신되는 브로드캐스트 신호 및/또는 브로드캐스트-관련 정보는 메모리(180)에 저장될 수 있다.
- [0063] 이동 통신 모듈(111)은 무선 신호를 이동 통신 네트워크 내의 기지국, 외부 단말, 및 서버 중 적어도 하나에 송

신할 수 있거나, 또는 기지국, 외부 단말, 및 서버 중 적어도 하나로부터 무선 신호를 수신할 수 있다. 텍스트/멀티미디어 메시지의 수신 및 송신에 기초하여, 신호는 음성 호출 신호, 영상통화 호출 신호, 및 다양한 포맷의 데이터를 포함할 수 있다.

[0064] 무선 인터넷 모듈(112)은 무선 액세스에 사용되는 모듈에 대응할 수 있고, 단말 장치(100)에 포함될 수 있거나 또는 외부적으로 단말 장치(100)에 연결될 수 있다. 무선 LAN(WLAN 또는 Wi-Fi), 와이맥스(world interoperability for microwave access, WiMAX), 고속 다운링크 패킷 액세스(high speed downlink packet access, HSDPA) 등이 무선 인터넷 기술로서 사용될 수 있다.

[0065] 로컬 에어리어 통신 모듈(113)은 로컬 에어리어 통신에 사용되는 모듈에 대응할 수 있다. 또한, 블루투스(Bluetooth), 무선 주파수 식별(radio frequency identification, RFID), 적외선 통신 규격(infrared data association, IrDA), 초광대역(ultra wide band, UWB), 및/또는 ZigBee가 로컬 에어리어 통신 기술로서 사용될 수 있다.

[0066] 위치 정보 모듈(114)은 이동 단말(100)의 위치를 결정 또는 취득할 수 있다. 위치 정보 모듈(114)은 범지구 내비게이션 위성 시스템(global navigation satellite system, GNSS)을 사용해서 위치 정보를 취득할 수 있다. GNSS는, 무선 내비게이션 수신기들이 지구 표면 상의 또는 그 부근의 무선 내비게이션 수신기들의 위치를 결정할 수 있도록, 지구 주위를 회전하면서 기준 신호를 소정 타입의 무선 내비게이션 수신기들에 송신하는 무선 내비게이션 위성 시스템이다. GNSS는 미국의 범지구 위치결정 시스템(global positioning system, GPS), 유럽의 갈릴레오 시스템(Galileo system) 시스템, 러시아의 범지구 궤도 내비게이션 위성 시스템(global orbit navigation satellite system), 중국의 컴퍼스 시스템(compass system), 일본의 준-천정 위성 시스템(quasi-zenith satellite system) 등을 포함할 수 있다.

[0067] GPS 모듈은 위치 정보 모듈(114)의 대표적인 예이다. GPS 모듈(114)은, 계산을 통해, 하나의 지점 또는 대상과 적어도 3개의 위성 사이의 거리에 관한 정보 및 측정을 통해 거리 정보가 취득되는 시간에 관한 정보를 취득할 수 있고, 취득된 거리 정보에 삼각 측량법을 적용해서, 경도, 위도, 및 고도에 기초하여 소정의 시간에 해당 지점 또는 대상에 관한 3차원 위치 정보를 취득할 수 있다. 대안으로서, GPS 모듈(114)은, 계산을 통해, 3개의 위성을 사용해서 위치 및 시간 정보를 취득하고, 계산을 통해 취득된 위치 및 시간 정보를 다른 위성을 사용해서 수정할 수 있다. 또한, GPS 모듈(114)은 지속적으로 현재 위치를 실시간으로 계산하고, 위치결정 또는 위치 정보를 사용해서 속도 정보를 계산할 수 있다.

[0068] 센서(120)는 단말 장치(100)의 개방/폐쇄 상태와 같은 단말 장치(100)의 현재 상태를 감지할 수 있고, 단말 장치(100)의 위치, 사용자가 단말 장치(100)와 접촉하고 있는지의 여부, 단말 장치(100)의 방향, 및 단말 장치(100)의 가속/감속을 감지할 수 있다. 또한, 센서(120)는 단말 장치(100)의 동작을 제어하는 데 사용되는 감지 신호를 생성할 수 있다. 예를 들어, 슬라이드 폰의 경우, 센서(120)는 슬라이드 폰이 개방 또는 폐쇄되는지의 여부를 감지할 수 있다. 또한, 센서(120)는 전원 장치(190)가 전력을 공급하는지의 여부 및/또는 인터페이스 유닛(170)이 외부 장치에 연결되는지의 여부를 감지할 수 있다. 센서(120)는 구체적으로 자세 검출 센서, 근접도 센서 등을 포함할 수 있다.

[0069] 사용자 입력 모듈(130)은, 입력되는 디지털 정보, 문자 정보, 또는 접촉 터치 동작/비-접촉 제스처를 수신하고, 단말 장치(100)의 사용자 설정 및 기능 제어 등과 관련되는 신호 입력을 수신하도록 구성된다. 터치스크린이라고도 하는 터치 패널(131)은, 터치 패널(131) 상에서 또는 그 부근에서 사용자에게 의해 수행되는 터치 동작(예를 들어, 터치 패널(131) 상에서 또는 터치 패널(131) 부근에서 손가락 또는 스타일러스와 같은 임의의 적절한 물체 또는 액세서리를 사용해서 사용자에게 의해 수행되는 동작)을 수집하고, 미리 설정된 프로그램에 기초하여 상응하는 연결 장치를 구동할 수 있다. 선택적으로, 터치 패널(131)은 두 부분, 즉 터치 검출 장치 및 터치 제어기를 포함할 수 있다. 터치 검출 장치는 사용자의 터치 방향을 검출하고, 터치 동작에 기인하는 신호를 검출하고, 신호를 터치 제어기에 전달한다. 터치 제어기는 터치 검출 장치로부터 터치 정보를 수신하고, 터치 정보를 접촉 좌표로 변환하고, 이어서 접촉 좌표를 프로세서(150)에 송신한다. 또한, 터치 제어기는 프로세서(150)에 의해 송신되는 커맨드를 수신 및 실행할 수 있다. 예를 들어, 사용자는 터치 패널(131) 상에서 애플리케이션의 아이콘을 손가락으로 탭핑한다. 터치 검출 장치는 탭핑에 기인하는 신호를 검출하고 나서, 신호를 터치 제어기에 전달한다. 이어서, 터치 제어기는 신호를 좌표로 변환하고, 좌표를 프로세서(150)에 송신한다. 프로세서(150)는 좌표 및 신호의 타입(단일 탭핑 또는 이중 탭핑)에 기초하여 애플리케이션을 기동한다.

[0070] 터치 패널(131)은 저항 타입, 정전 타입, 적외선 타입, 및 표면 탄성과 타입과 같은 복수의 타입을 사용해서 구현될 수 있다. 터치 패널(131)에 더하여, 입력 장치(130)는 다른 입력 장치(132)를 더 포함할 수 있다. 다른

입력 장치(132)는 물리적 키보드, 기능 키(예를 들어, 음량 제어 키 또는 온/오프 키), 트랙볼, 마우스, 조이스틱, 멤브레인 스위치(돔 스위치), 스크롤 휠(jog wheel), 및 조그 스위치(jog switch) 중 하나 이상을 포함할 수 있지만, 이들로 제한되는 것은 아니다.

[0071] 출력 모듈(140)은 사용자에게 의해 입력되는 정보, 사용자에게 제공되는 정보, 단말 장치(100)의 다양한 메뉴 인터페이스 등을 디스플레이하도록 구성되는 디스플레이 패널(141)을 포함한다. 선택적으로, 디스플레이 패널(141)은 액정 디스플레이(liquid crystal display, LCD), 유기 발광 다이오드(organic light-emitting diode, OLED) 등의 형태로 구성될 수 있다. 일부 다른 실시형태들에 있어서, 터치 패널(131)은 디스플레이 패널(141)을 덮어서 터치 디스플레이 스크린을 형성할 수 있다.

[0072] 또한, 출력 모듈(140)은 오디오 출력 모듈(142), 알람(143), 촉각 모듈(144) 등을 더 포함할 수 있다.

[0073] 오디오 출력 모듈(142)은, 호출 신호 수신 모드, 전화 호출 모드, 녹음 모드, 음성 인식 모드, 및 방송 수신 모드에서, 무선 통신 모듈(110)로부터 수신되는 오디오 데이터 또는 메모리(180)에 저장된 오디오 데이터를 출력할 수 있다. 오디오 출력 모듈(142)은 단말 장치(100)에서 수행되는 기능과 관련되는, 호출 신호 수신 톤 또는 메시지 수신 톤과 같은 오디오 신호를 출력할 수 있다. 오디오 출력 모듈(142)은 수신기, 스피커, 버저 등을 포함할 수 있다. 오디오 출력 모듈(142)은 헤드셋 잭을 통해 사운드를 출력할 수 있다. 사용자는 헤드셋을 헤드셋 잭에 연결함으로써 사운드를 청취할 수 있다.

[0074] 알람(143)은 단말 장치(100)의 이벤트의 발생을 나타내는 데 사용되는 신호를 출력할 수 있다. 예를 들어, 알람(143)은 호출 신호, 메시지, 입력 키 신호, 또는 입력 터치를 수신할 때 알람을 생성할 수 있다. 알람(143)은 비디오 신호 또는 주파수 신호와는 다른 형태의 신호, 예를 들어, 진동을 통해 이벤트의 발생을 나타내는 신호를 더 출력할 수 있다.

[0075] 촉각 모듈(144)은 사용자에게 의해 감지될 수 있는 다양한 촉각 효과를 생성할 수 있다. 진동이 촉각 효과의 예이다. 촉각 모듈(144)에 의해 생성되는 진동의 강도 및/또는 모드가 더 제어될 수 있다. 예를 들어, 상이한 진동들이 조합하여 또는 순차적으로 출력될 수 있다. 촉각 모듈(144)은 복수의 촉각 효과를 생성할 수 있다. 진동에 더하여, 촉각 효과는 또한 다음과 같은 복수의 효과, 즉 피부 표면에 대하여 수직으로 이동하는 바늘 어레이의 자극 효과, 스프레이 구멍 또는 흡입 구멍을 사용해서 형성되는 공기 스프레이 효과 또는 공기 흡입 효과, 피부-마찰 자극 효과, 전극-접촉 자극 효과, 정전기력을 사용한 자극 효과, 열을 흡수하거나 열을 방출할 수 있는 요소를 사용해서 재현될 수 있는 고온 또는 저온 효과 등의 하나 이상일 수 있다. 촉각 모듈(144)은 직접 접촉을 통해 촉각 효과를 보낼 수 있을 뿐만 아니라, 사용자가 사용자의 손가락 또는 팔의 근육 감각을 사용해서 촉각 효과를 느끼게 할 수도 있다. 단말 장치(100)는 복수의 촉각 모듈(144)을 포함할 수 있다.

[0076] 프로세서(150)는 하나 이상의 프로세서를 포함할 수 있고, 예를 들어, 프로세서(150)는 하나 이상의 중앙 처리 장치를 포함할 수 있거나, 또는 하나의 중앙 처리 장치 및 하나의 그래픽 처리 장치를 포함할 수 있다. 프로세서(150)가 복수의 프로세서를 포함하는 경우, 복수의 프로세서가 동일한 칩에 통합될 수 있거나, 또는 복수의 프로세서가 각각 독립적인 칩일 수 있다. 하나의 프로세서는 하나 이상의 물리적 코어를 포함할 수 있으며, 물리적 코어는 최소 처리 모듈이다.

[0077] 오디오/비디오 입력 모듈(160)은 오디오 신호 또는 비디오 신호를 입력하도록 구성된다. 오디오/비디오 입력 모듈(160)은 카메라(161) 및 마이크로폰(162)을 포함할 수 있다. 카메라(161)는, 정지 화상 또는 동화상으로 이루어지며 영상통화 모드 또는 촬영 모드에서 이미지 센서에 의해 취득되는 이미지 프레임을 처리할 수 있다. 처리된 이미지 프레임은 디스플레이 패널(141)에 디스플레이될 수 있다.

[0078] 카메라(161)에 의해 처리되는 이미지 프레임은 메모리(180)에 저장될 수 있거나, 또는 무선 통신 모듈(110)을 사용해서 외부 장치로 송신될 수 있다. 단말 장치(100)는 복수의 카메라(161)를 더 포함할 수 있다.

[0079] 마이크로폰(162)은 호출 모드, 녹음 모드, 또는 음성 인식 모드에서 외부 오디오 신호를 수신하고, 수신한 오디오 신호를 전자 오디오 데이터로 처리할 수 있다. 이어서, 오디오 데이터는 이동 통신 모듈(111)을 사용해서 이동 통신 기지국에 송신될 수 있는 형태로 변환되어 호출 모드에서 출력될 수 있다. 마이크로폰(162)은 다양한 노이즈 제거 알고리즘(noise cancellation algorithms)을 사용하여 외부 오디오 신호가 수신될 때 생성되는 노이즈를 제거 또는 감소시킬 수 있다.

[0080] 인터페이스 모듈(170)은 단말 장치(100)에 연결되는 외부 장치의 경로로서 사용될 수 있다. 인터페이스 모듈(170)은 데이터 또는 전력을 외부 장치로부터 수신하고 데이터 또는 전력을 단말 장치(100)의 내부 컴포넌트에 송신할 수 있거나, 또는 단말 장치(100)의 데이터를 외부 장치에 송신할 수 있다. 예를 들어, 인터페이스 모듈

(170)은 유선/무선 헤드-마운트형 헤드셋 포트, 외부 충전기 포트, 유선/무선 데이터 포트, 메모리 카드 포트, 가입자 식별 모듈을 갖는 장치들을 연결하는 데 사용되는 포트, 오디오 I/O 포트, 비디오 I/O 포트, 및/또는 헤드셋 포트를 포함할 수 있다.

- [0081] 인터페이스 모듈(170)은 가입자 식별 모듈에 더 연결될 수 있으며, 가입자 식별 모듈은 단말 장치(100)의 사용 권한을 검증하는 데 사용되는 정보를 저장하는 칩이다. 가입자 식별 모듈을 포함하는 식별 장치는 스마트 카드 형태로 제조될 수 있다. 따라서, 식별 장치는 인터페이스 모듈(170)을 사용해서 단말 장치(100)에 연결될 수 있다.
- [0082] 인터페이스 모듈(170)은 또한 단말 장치(100)가 외부 트레이에 연결될 때 외부 트레이로부터 전력을 단말 장치(100)에 제공하는 데 사용되는 경로일 수 있거나, 또는 트레이를 사용해서 사용자에게 의해 입력되는 다양한 커맨드 신호를 단말 장치(100)에 송신하는 데 사용되는 경로일 수 있다. 트레이로부터 입력되는 다양한 커맨드 신호 또는 전력은 단말 장치(100)가 트레이에 정확하게 설치되어 있는지를 확인하기 위한 신호로 사용될 수 있다.
- [0083] 메모리(180)는 컴퓨터 프로그램을 저장하고, 컴퓨터 프로그램은 운영 체제 프로그램(182), 애플리케이션 프로그램(181) 등을 포함한다. 예를 들어, 일반적인 운영 체제는 Microsoft의 Windows 또는 Apple의 MacOS와 같이, 데스크톱 컴퓨터 또는 노트북 컴퓨터에 사용되는 시스템; 또는 Google에 의해 개발된 Linux-기반 안드로이드(Android) 시스템과 같이, 이동 단말에 사용되는 시스템이다. 프로세서(150)는 메모리(180)로부터 컴퓨터 프로그램을 판독하고 나서, 컴퓨터 프로그램에 의해 정의된 방법을 수행하도록 구성된다. 예를 들어, 프로세서(150)는 운영 체제 프로그램(182)을 판독하여 단말 장치(100) 상에서 운영 체제를 실행하고 운영 체제의 다양한 기능을 구현하거나; 또는 하나 이상의 애플리케이션 프로그램(181)을 판독하여 단말 장치 상에서 애플리케이션을 실행한다.
- [0084] 운영 체제 프로그램(182)은 본원의 임의의 실시형태에서 제공되는 방법을 구현할 수 있는 컴퓨터 프로그램을 포함하므로, 프로세서(150)가 운영 체제 프로그램(182)을 판독하고 운영 체제를 실행한 후에, 운영 체제가 본원에서 제공되는 실시간 애플리케이션 순위화 기능 및/또는 리소스 관리 기능을 가질 수 있다.
- [0085] 메모리(180)는 컴퓨터 프로그램과는 상이한 다른 데이터(183), 예를 들어, 본원에서 수집을 통해 취득되는 애플리케이션 정보와 같은 정보, 트레이닝을 통해 취득되는 모델, 및 실시간 순위화의 결과를 더 저장한다. 다른 예로서, 메모리(180)는 입력/출력 데이터(예를 들어, 전화번호부 데이터, 메시지, 정지 화상, 및/또는 동화상), 및 터치스크린에 터치 입력이 적용될 때 출력되는 다양한 진동 및 사운드 모드에 관한 데이터를 일시적으로 저장한다.
- [0086] 메모리(180)는 다음과 같은 타입, 즉 플래시(flash) 메모리, 하드 디스크-타입 메모리, 마이크로 멀티미디어 카드 메모리, 카드 메모리(예를 들어, SD 메모리 또는 XD 메모리), 랜덤 액세스 메모리(random access memory, RAM), 정적 랜덤 액세스 메모리(static RAM, SRAM), 리드-온리 메모리(read only memory, ROM), 전기적 소거 및 프로그램 가능 리드-온리 메모리(electrically erasable programmable read-only memory, EEPROM), 프로그램 가능 리드-온리 메모리(programmable ROM, PROM), 자기 메모리, 자기 디스크, 또는 광학 디스크 중 하나 이상일 수 있다.
- [0087] 일부 다른 실시형태들에 있어서, 메모리(180)는 대안적으로 인터넷 상의 네트워크 저장 장치일 수 있고, 단말 장치(100)는 인터넷 상의 메모리(180) 상에서 업데이트 동작, 판독 동작 등을 수행할 수 있다.
- [0088] 전원 장치(190)는 프로세서(150)의 제어 하에서 외부 전력 및 내부 전력을 공급받고, 단말 장치(100)의 각각의 컴포넌트의 동작에 필요한 전력을 제공할 수 있다.
- [0089] 모듈들 사이의 연결 관계는 단지 예시일 뿐이다. 본원의 임의의 실시형태에서 제공되는 방법은 다른 연결 방식으로 단말 장치에 적용될 수도 있으며, 예를 들어, 모든 모듈들은 버스를 사용해서 연결된다.
- [0090] 본원에서 제공되는 방법은 하드웨어 또는 소프트웨어를 사용해서 구현될 수 있다. 하드웨어 구현에 있어서, 주문형 집적 회로(application specific integrated circuit, ASIC), 디지털 신호 프로세서(digital signal processor, DSP), 프로그램 가능 로직 장치(programmable logic device, PLD), 필드 프로그램 가능 게이트 어레이(field programmable gate array, FPGA), 프로세서, 컨트롤러, 마이크로컨트롤러, 및/또는 마이크로프로세서와 같은 전자 장치들 중 적어도 하나가 본원의 구현예를 구현하기 위해 사용될 수 있다. 소프트웨어 구현에 있어서, 프로세스, 기능 등의 구현은 적어도 하나의 기능 및 동작을 수행하는 소프트웨어 모듈을 사용해서 구현될 수 있다. 소프트웨어 모듈은 임의의 적절한 소프트웨어 언어로 작성된 소프트웨어 프로그램에 의해 구현될 수 있다. 소프트웨어 프로그램은 메모리(180)에 저장될 수 있고, 프로세서(150)는 소프트웨어 프로그램을

관독 및 실행한다.

- [0091] 본원에서 제공되는 방법의 구현에는 안드로이드(Android) 시스템을 예로서 사용해서 도 2에서 설명된다. 도면에 도시된 바와 같이, 일반적인 안드로이드 시스템(200)은 애플리케이션(210), 애플리케이션 프레임워크(220), 시스템 런타임 라이브러리 및 안드로이드 런타임(230), 및 커널(240)을 포함한다. 안드로이드는 리눅스를 기반으로 개발되었기 때문에, 커널(240)은 리눅스 커널이다.
- [0092] 애플리케이션(210)은 브라우저, 미디어 플레이어, 게임 애플리케이션, 및 텍스트 처리 애플리케이션과 같은 다양한 애플리케이션을 포함한다. 일부 애플리케이션은 시스템의 고유 애플리케이션이고, 일부 애플리케이션은 요청에 따라 사용자에게 의해 인스톨된다. 이 애플리케이션은 사용자에게 의한 이들 애플리케이션의 사용 스테이터스에 중점을 두어, 사용자에게 의해 가장 많이 사용될 것 같은 애플리케이션에 대한 리소스 공급을 보장하고, 사용자의 관점에서 애플리케이션이 중지되지 않도록 한다.
- [0093] 애플리케이션 프레임워크(220)는 안드로이드 서비스(224)를 포함하고, 안드로이드 서비스는 다른 모듈에 의한 사용을 위해 안드로이드 시스템에 의해 제공되는 복수의 시스템 서비스, 예를 들어, 전력 매니저(224-a), 통지 매니저(224-b), 연결 매니저(224-c), 패키지 매니저(224-d), 위치 매니저(224-e), 유선 액세스 매니저(224-g), 블루투스 장치(224-h), 및 뷰 시스템(224-i)을 포함한다. 모든 매니저의 구현에 대해서는, 종래기술에서의 안드로이드 시스템에 의해 제공되는 모듈을 참조하고, 본원에서는 상세하게 설명하지 않는다.
- [0094] 시스템 런타임 라이브러리 및 안드로이드 런타임(230)은 시스템 런타임 라이브러리(231) 및 안드로이드 런타임(232)을 포함한다. 시스템 런타임 라이브러리(231)는 프로그램 라이브러리라고도 하며, 일부 C/C++ 라이브러리를 포함하고, 이들 라이브러리는 안드로이드 시스템의 상이한 컴포넌트들에 의해 사용될 수 있다.
- [0095] 안드로이드 런타임(232)은 코어 라이브러리 및 달빅(Dalvik) 가상 머신을 포함한다. 코어 라이브러리는 자바(Java) 프로그래밍 언어 코어 라이브러리의 대부분의 기능을 제공한다. 각각의 안드로이드 애플리케이션 프로그램은 안드로이드 애플리케이션 프로그램의 프로세스에서 실행되고, 독립적인 달빅 가상 머신 인스턴스를 구비한다. 달빅은 복수의 가상 시스템을 동시에 효율적으로 실행할 수 있는 장치로 설계된다. 달빅 가상 머신은 스레드 메커니즘 및 기본 메모리 관리 메커니즘과 같은 리눅스 커널(240)의 일부 기능에 의존한다.
- [0096] 보안, 메모리 관리, 프로세스 관리, 네트워크 프로토콜 스택, 및 다양한 구동 모델과 같은, 안드로이드의 코어 시스템 서비스는 리눅스 커널(240)에 의존한다. 리눅스 커널(240)은 하드웨어와 소프트웨어 사이의 추상 계층으로서도 사용된다. 또한, 안드로이드는 리눅스 커널을 부분적으로 더 수정하고, 수정은 주로 다음 두 부분에 관련된다:
- [0097] 바인더 (IPC) 드라이버: 바인더 (IPC) 드라이버는 효과적인 프로세스간 통신을 제공한다. 리눅스 커널이 이미 기능을 제공했다고 해도, 안드로이드 시스템의 많은 서비스들은 이 기능을 사용할 필요가 있다. 구체적인 이유로, 안드로이드 시스템이 프로세스간 통신 메커니즘을 구현한다.
- [0098] 전원 관리: 이는 주로 전력을 절약하는 데 사용된다. 안드로이드 시스템이 이동 단말, 예를 들어, 스마트폰 용으로 설계되기 때문에, 저전력 소비가 중요한 목적이다.
- [0099] 안드로이드 시스템은 소프트웨어 코드 형태로 메모리(180)에 저장되고, 프로세서(150)는 소프트웨어 코드를 관독 및 실행하여, 단말 장치(100) 상에서, 이 실시형태에서 제공되는 기능들을 포함하여 시스템에 의해 제공되는 기능들을 구현한다.
- [0100] 이 실시형태에서의 개량은 주로 애플리케이션 프레임워크(220)에 있다. 도면에 도시된 바와 같이, 원래의 안드로이드 시스템 서비스에 더하여, 이 실시형태에서의 애플리케이션 프레임워크(220)는 지각 관리 장치를 더 포함한다. 지각 관리 장치는 데이터 수집 모듈(221), 애플리케이션 순위화 모듈(222), 및 의사결정 실행 모듈(223)을 포함한다.
- [0101] 지각 관리 장치는 다른 컴포넌트에 의한 사용을 위해 안드로이드 시스템 서비스로서 제공될 수도 있다는 점을 이해해야 한다. 모듈(221 내지 223)은 제각기 3가지 시스템 서비스로서 사용될 수 있거나, 또는 적절히 조합 또는 세분될 수 있다.
- [0102] 데이터 수집 모듈(221)은 단말 장치(100)의 시나리오 데이터를 수집한다. 시나리오 데이터는 장치의 위치 데이터(예를 들어, GPS 위치 데이터), 장치의 내장 컴포넌트 또는 외부 컴포넌트의 스테이터스 데이터 등을 포함한다. 예를 들어, 스테이터스 데이터는 디스플레이 패널 레이아웃 스테이터스, 네트워크 연결 스테이터스, 헤드셋 또는 충전 케이블의 연결 스테이터스, 카메라의 스테이터스, 오디오/비디오 컴포넌트, 또는 센서 등일 수

있다. 단말 장치(100)의 복수의 컴포넌트가 도 1을 참조하여 이해될 수 있다. 본 명세서에서 "컴포넌트(component)"는 하드웨어 및 소프트웨어를 모두 포함한다.

- [0103] 데이터 수집 모듈(221)은 애플리케이션의 사용과 관련된 애플리케이션 데이터, 예를 들어, 애플리케이션의 타입, 애플리케이션의 이름, 및 애플리케이션이 사용된 시간을 더 수집한다. 수집된 데이터는 바로 메모리(180)에 저장되거나 또는 다른 모듈에 송신되거나, 또는 처리된 후에 메모리(180)에 저장되거나 또는 다른 모듈에 송신된다.
- [0104] 애플리케이션 순위화 모듈(222)은 데이터 수집 모듈(221)에 의해 실시간으로 수집된 데이터 및 머신 러닝 모델에 기초하여 실시간 시나리오로 모든 애플리케이션의 중요도 순위화 결과를 결정한다. 실시간 중요도 순위화가 구현되기 전에, 애플리케이션 순위화 모듈(222)은 데이터 수집 모듈(221)에 의해 수집된 이력 데이터에 기초하여 트레이닝을 더 수행해서 전술한 머신 러닝 모델을 취득한다.
- [0105] 의사결정 실행 모듈(223)은, 애플리케이션 순위화 모듈(222)에 의해 출력되는 실시간 애플리케이션 중요도 순위에 기초하여, 시스템 리소스를 어떻게 관리할지에 대하여 의사결정을 하고, 바로 다른 모듈을 실행 또는 호출하여 상응하는 리소스 관리 조치를 수행한다. 특정 구현예에 있어서, 의사결정 실행 모듈(223)은 모든 리소스 관리 의사결정을 할 때 반드시 실시간 애플리케이션 중요도 순위를 필요로 하는 것은 아니라는 점을 이해해야 한다.
- [0106] 전술한 모듈들이 중간 데이터 또는 최종 데이터를 저장할 필요가 있을 경우에는, 저장 기능이 필요해진다는 점에 유의해야 한다. 이들 저장 기능은 독립적인 저장 모듈에 의해 구현될 수 있거나, 또는 상기 3개의 모듈에 병합됨으로써 별도로 구현될 수 있다.
- [0107] 일부 다른 실시형태들에 있어서, 모듈(221 내지 223)은 시스템 런타임 라이브러리 및 안드로이드 런타임(230) 또는 커널(240)에서 구현될 수 있거나; 또는 이들 모듈 중 일부는 애플리케이션 프레임워크(220)에서 구현될 수 있고, 이들 모듈 중 일부는 다른 계층에서 구현될 수 있다.
- [0108] 도 3에 도시된 바와 같이, 애플리케이션 순위화 모듈(222)은 2개의 서브모듈, 즉 모델 트레이닝 모듈(310) 및 실시간 순위화 모듈(320)을 포함한다.
- [0109] 모델 트레이닝 모듈(310)은, 데이터 수집 모듈(221)에 의해 수집되는 이력 데이터에 기초하여, 실시간 애플리케이션 순위에 적용되는 모델을 트레이닝하고, 실시간 순위화 모듈(320)이 실시간 애플리케이션 순위화를 수행할 때 실시간 순위화 모듈(320)에 의한 호출을 위해 해당 모델을 메모리(180)에 저장한다. 구체적으로, 모델 트레이닝 모듈(310)은 주로 3가지 모델, 즉 차원-기반 분류 모델, 순위화 모델, 및 N-값 예측 모델을 트레이닝한다. 본원에서의 "모델(model)"은 넓은 의미의 개념이고, 파라미터, 수식, 알고리즘, 대응관계 등이 모델로 간주될 수 있다.
- [0110] 차원-기반 분류 모델은 애플리케이션 사용 규칙들을 분류하는 데 사용되고, 분류는 일차원 분류 또는 다차원 분류일 수 있다. 시간 또는 공간과 같은 "차원(dimension)"은 사용 규칙 분류를 위한 기준이다. 시간이 예로서 사용되며, 하루 24시간은 애플리케이션 사용 규칙을 2가지 카테고리, 즉 근무 기간 및 비-근무 기간으로 분류하기 위한 분류 기준으로 사용된다. 구체적으로, 단말이 현재 근무 기간에 속하는지 또는 비-근무 기간에 속하는지의 여부는 현재 시스템 시간 및 트레이닝을 통해 취득되는 차원-기반 분류 모델에 기초하여 결정된다. 본 명세서에서, 이는 실제로 사용자가 현재 근무 상태인지 또는 비-근무 상태인지의 여부를 반영한다. 이들 두 상태에 있어서, 사용자는 애플리케이션을 사용하기 위한 규칙에서 상이한 특징들을 제시한다. 또한, 차원은 다차원, 예를 들어, 시간 차원 및 위치 차원일 수 있다. 예를 들어, 근무 기간은 "회사 근무(working at a company)" 또는 "출장 근무(working on a business trip)"로 더 분류되고, 비-근무 기간은 "집(home)", "여행(travelling)" 등으로 더 분류된다. 이해의 편의상, 본 명세서의 실시예에서는 의미론적 설명, 예를 들어, "근무(working)" 및 "여행(travelling)"이 사용된다. 이는 특정 구현예로 제한되지 않는다는 점을 이해해야 한다.
- [0111] N-값 예측 모델은 특정 분류에서 사용자에게 의해 빈번하게 사용되는 애플리케이션의 수량을 결정하는 데 사용된다. N의 값은 양의 정수이다. 예를 들어, 분류는 시간 차원에 기초하여 수행된다. N의 값은 특정 기간에 사용자에게 의해 빈번하게 사용되는 애플리케이션의 수량을 반영한다.
- [0112] 실시간 순위화 모듈(320)은 실시간 처리 모듈이고, 데이터 수집 모듈(221)에 의해 실시간으로 수집되는 데이터 및 모델 트레이닝 모듈(310)에 의해 트레이닝되는 모델에 기초하여 실시간 애플리케이션 중요도 순위를 결정할 수 있다. 실시간 순위화 모듈(320)은 모델 트레이닝 모듈(310)에 의해 제공되는 기능을 사용해서 3가지 기능,

즉 차원-기반 분류 예측(321), 애플리케이션 순위 예측(322), 및 N-값 취득(323)을 수행한다.

- [0113] 모델 트레이닝 모듈(310) 및 실시간 순위화 모듈(320) 각각에서의 각 기능은 기능 모듈 또는 유닛으로 간주될 수 있다는 점을 이해해야 한다.
- [0114] 도 3에 도시된 바와 같이, 데이터 수집 모듈(221)은 수집된 데이터의 전부 또는 일부에 대한 정리/처리와 같은 처리를 수행하고 나서, 처리된 데이터를 메모리(180)에 저장하거나 또는 처리된 데이터를 실시간 순위화 모듈(320)에 제공할 수 있다.
- [0115] 전술한 모듈들 및 기능들이 전부 모든 실시형태에 적용될 필요는 없다는 점에 유의해야 한다. 본원의 일부 실시형태에 있어서, 구현을 위해 일부 기능만이 선택될 수 있다.
- [0116] 실시형태들에서 제공되는 애플리케이션 순위화 방법 및 리소스 관리 방법을 아래에서 설명하고, 즉, 전술한 모듈들 또는 유닛들의 구체적인 구현예를 설명한다.
- [0117] 도 4는 실시형태에 따른 애플리케이션 순위화 방법 및 리소스 관리 방법의 포괄적 해법의 개략도이다. 도면은 제1 단계에서의 데이터 수집, 제2 단계에서의 모델 트레이닝, 제3 단계에서의 애플리케이션 순위화, 및 제4 단계에서의 리소스 관리를 포함하는 4개의 주된 방법 절차를 도시한다. 설명 과정에서, 해법의 포괄적 개요를 제공하기 위해, 해법을 전술한 순서로 설명한다. 그러나, 해법의 구체적인 구현에 있어서는, 4개의 단계 전부가 연속적으로 수행되는 것은 아니라는 점을 이해해야 한다.
- [0118] 단말 장치가 실행된 후에, 단말 장치는 먼저 제1 단계, 즉, 데이터 수집 프로세스를 수행하고, 주로 사용자의 애플리케이션 사용에 관한 데이터, 및 단말 장치의 시스템 스테이터스 데이터 및 위치 데이터와 같은 정보를 수집한다. 정보는 사용자에게 의해 애플리케이션을 사용하기 위한 규칙을 반영할 수 있고, 규칙은 장래에 사용자에게 의해 애플리케이션을 사용할 확률을 예측하는 데 사용된다. 수집된 데이터 중 일부는 바로 저장되고, 수집된 데이터 중 일부는 저장되기 전에 처리되어야 할 수도 있다(①). 제1 단계는 데이터 수집 모듈(221)에 의해 수행된다.
- [0119] 일정 기간 동안 데이터 수집이 수행된 후에, 제2 단계에서의 모델 트레이닝이 수행된다. 본원에서는, 머신 러닝 트레이닝 방법이 모델 순위화에 사용된다. 특정 수량의 이력 데이터(②)가 입력되어야 한다. 그러나, 제1 트레이닝이 시작되기 전에 수집을 수행하는 데 특별히 필요한 시간은 본원에서 제한되지 않는다. 머신 러닝 모델 트레이닝 프로세스는 모델 설정 프로세스이고, 본질적으로 파라미터 학습 및 최적화 프로세스이다. 모델을 트레이닝하는 것은 모델 파라미터를 학습 및 업데이트하는 것이다. 트레이닝된 머신 러닝 모델 또는 모델 파라미터는 제3 단계에서의 순위화에서 사용하기 위해 저장된다.
- [0120] 순위화 모델이 트레이닝되기 전에, 수집된 이력 데이터는 머신 러닝 알고리즘 또는 다른 알고리즘에 기초하여 하나 이상의 차원에서 복수의 카테고리로 분류된다. 복수의 카테고리에서 애플리케이션을 사용하기 위한 규칙은 상이한 특징들을 제시하고, 각각의 순위화 모델은 복수의 카테고리에 대하여 별도로 트레이닝된다. 따라서, 제2 단계에서 중요한 출력은 복수의 카테고리 및 복수의 카테고리에 제각기 대응하는 복수의 머신 러닝 모델(또는 모델 파라미터)(③)이다. 제2 단계는 모델 트레이닝 모듈(310)에 의해 수행된다. 또한, 모델 트레이닝 모듈(310)은, N-값 예측 알고리즘에 기초하여, 특정 카테고리에서 사용자에게 의해 빈번하게 사용되는 애플리케이션의 수량(NB)을 계산하고, 해당 수량을 후속 리소스 관리를 위해 저장한다. N-값 예측 프로세스는 제3 단계 또는 제4 단계에 배치될 수도 있다.
- [0121] 단말 장치는 주기적으로 트레이닝 프로세스를 시작할 수 있거나, 또는 지정된 시점에 트레이닝 프로세스를 시작할 수 있거나, 또는 수집 데이터가 해당 데이터가 단순 데이터이면 일치되지 않는다는 원리에 따라 트레이닝 프로세스를 시작할 수 있거나 하는 등이다.
- [0122] 제2 단계에서의 모델 트레이닝이 완료된 후에, 제3 단계에서의 애플리케이션 순위화가 수행될 수 있다. 제3 단계에서, 애플리케이션들은 실시간 데이터(④) 및 제2 단계에서 출력된 머신 러닝 모델(⑤)에 기초하여 순위화된다. 두 아이টে 더하여, 일부 이력 데이터(⑤)가 또한 지원을 위해 필요해진다. 실시간 데이터는 실시간으로 수집된 현재 애플리케이션, 현재 애플리케이션의 사용 시간, 단말 장치의 현재 시간(즉, 현재 시스템 시간), 단말 장치의 현재 스테이터스 데이터, 단말 장치의 현재 위치 데이터 등을 하나 이상 포함한다. 복수의 애플리케이션을 사용하기 위한 시간 시퀀스와 같은 일부 데이터는 이력 데이터에 기록된 복수의 애플리케이션의 사용 시간을 사용해서 계산을 통해 더 취득될 필요가 있다. 제3 단계는 실시간 순위화 모듈(320)에 의해 수행되고, 출력이 단말 장치 상에 인스톨된 모든 애플리케이션의 순위화 결과(⑥)이다.

- [0123] 제3 단계에서의 애플리케이션 순위화는 주기적으로 기동될 수 있거나, 또는 이벤트-트리거될 수 있다. 트리거 이벤트는 애플리케이션 전환, 애플리케이션 설치, 애플리케이션 제거, 단말 장치의 스테이터스 변경 등일 수 있다. 트리거 이벤트들은 전부 사용자에게 애플리케이션 중요도 변경을 야기할 수 있다. 이 경우, 순위화 결과는, 순위화 결과가 제4 단계에서의 리소스 관리에 필요해질 때 사용하기 위해 저장될 수 있다. 또한, 트리거 이벤트는 제4 단계에서의 리소스 관리에 의해 트리거되는 요청 이벤트일 수도 있으며, 즉, 애플리케이션 순위화는 리소스 관리에 의한 호출 하에서 수행된다.
- [0124] 제4 단계에서의 리소스 관리 동안, 메모리에 저장된 최신 순위화 결과(㉑)가 액세스 및 사용될 수 있거나, 또는 순위화 결과는 요청에 따라 실시간으로 제3 단계에서의 애플리케이션 순위화를 호출(㉒)함으로써 취득될 수 있다. 이어서, 애플리케이션 중요도 순위에 기초하여 리소스가 관리된다.
- [0125] 4개의 단계의 상세한 구현예들은 아래에서 실시예들을 사용해서 별도로 설명된다.
- [0126] 도 5는 제1 단계에서의 데이터 수집 프로세스의 개략도이다. 데이터 수집 모듈(221)은 먼저 이벤트 모니터링 메커니즘을 설정(S501)하여 키 이벤트의 발생 여부를 모니터링한다. 키 이벤트가 검출(S502)되면, 데이터 수집 프로세스가 트리거(S503)된다. 수집된 데이터는 바로 저장(S505)될 수 있거나, 또는 저장되기 전에 정리/처리(S504)될 수 있다.
- [0127] 데이터 정리(Data cleaning)는, 컴퓨터가 데이터를 재-검사하고 재-확인해서, 반복되는 정보를 삭제하고, 기존 에러를 수정하고, 데이터 일관성을 제공하는 프로세스이다. 데이터 처리는, 컴퓨터가 데이터의 타입, 포맷 등을 변경하거나, 또는 데이터에 대한 수학적 변환과 같은 처리를 수행하는 것을 의미한다.
- [0128] 일 양태에 있어서, 데이터 수집 모듈(221)은 단말 장치(100)의 시나리오 데이터 및 애플리케이션 사용과 관련되는 이력 애플리케이션 데이터를 수집하고, 수집된 데이터를 정리/처리하고 나서 정리된/처리된 데이터를 메모리(180)에 저장하고, 정리된/처리된 데이터를 이력 데이터로서 모델 트레이닝 모듈(310)에 제공하여 트레이닝을 통해 다양한 모델을 생성한다. 구체적으로, 데이터 수집 모듈(221)은 단말 장치의 스테이터스, 애플리케이션 사용 스테이터스 등을 실시간으로 모니터링하고; 조건이 충족되면 데이터를 수집 및 저장한다. 예를 들어, 데이터 수집 모듈(221)은 사용자 행동 스테이터스(단말 장치의 행동 스테이터스, 예를 들어, 이동 상태 또는 정지 상태로서 이해될 수도 있음), 애플리케이션 사용 스테이터스, 시스템 스테이터스, 및 위치 스테이터스의 변경을 모니터링 및 버퍼링한다. 애플리케이션 포그라운드-백그라운드 전환 또는 다른 타입의 키 이벤트가 발생한 후에, 포그라운드로 전환된 애플리케이션의 패키지 이름, 현재 시간, 및 전술한 스테이터스들의 최신 데이터와 같은 정보는 영구 저장 매체에 기록된다.
- [0129] 다른 양태에 있어서, 데이터 수집 모듈(221)은 순위화가 필요해질 때 실시간으로 데이터를 더 수집한다. 수집된 데이터가 실시간 데이터로서 사용되고 실시간 순위화 모듈(320)의 입력이 되어, 실시간 순위화 모듈(320)이 애플리케이션들을 실시간으로 순위화한다.
- [0130] 예를 들어, 데이터 수집 모듈(221)에 의해 수집된 데이터 및 수집 방식은 이하의 표에 도시된다. 일부 실시형태에서, 일부 데이터는 요건에 따라 대응되게 감소 또는 추가될 수 있다는 점을 이해해야 한다. 이 실시형태는 단지 예시일 뿐이며, 이 표의 데이터로 제한되게 하려는 것이 아니다.
- [0131] 표 1

데이터 카테고리	수집 방식
시간	현재 시스템 시간이 취득된다.
애플리케이션이 포그라운드로 전환된 시간	이 시간은 현재 시스템 시간을 수집함으로써 또는 애플리케이션이 전환될 때 타임스탬프를 취득함으로써 취득된다.
포그라운드 애플리케이션의 패키지 이름	애플리케이션의 패키지 이름은 패키지 매니저(224-d) (PackageManager)에 의해 제공되는 getApplicationInfo 방법을 사용해서 취득된다. 수집-관련 모듈: PackageManager
블루투스 연결 스태이터스	장치에 구속되는 블루투스 장치 목록이 취득되고, 연결된 상태의 블루투스 장치가 존재하는지의 여부가 결정된다. 대안으로서, 블루투스 연결 스태이터스는 BluetoothDevice.ACTION_BOND_STATE_CHANGED 를 모니터링함으로써 수동적으로 취득될 수 있다. 수집-관련 모듈: 블루투스 장치(224-h) (BluetoothDevice)
네트워크 연결 스태이터스	현재 네트워크 스태이터스는 CONNECTIVITY_SERVICE 시스템 서비스에 의해 제공되는 getActiveNetworkInfo 방법을 사용해서 취득된다. 대안으로서, 현재 네트워크 스태이터스는 동작: ConnectivityManager.CONNECTIVITY_ACTION 을 등록함으로써 수동적으로 취득될 수 있다. 수집-관련 모듈: 연결 매니저(224-c) (ConnectivityManager)
통지 목록 스태이터스	스태이터스 바에 있는 현재 통지 정보 목록은 시스템 서비스 NOTIFICATION_SERVICE 에 의해 제공되는 getActiveNotifications 방법을 사용해서 취득되고, 해당 목록은, getPackageName 방법을 사용함으로써, 통지 정보와 연관되는 애플리케이션 패키지 이름을 취득하도록 자세히 검토된다. 대안으로서, 통지 정보 목록은 NotificationListenerService 방식으로 수동적으로 취득될 수 있다. 수집-관련 모듈: 통지 매니저(224-b) (NotificationManager)

[0132]

충전 케이블 연결 스테이터스	특정 의도(ACTION_BATTERY_CHANGED)가 전력 매니저(BatteryManager)에 송신되어, 단말 장치가 현재 충전 케이블에 연결되어 있는지의 여부를 확인한다. 대안으로서, 충전 케이블 연결 스테이터스는 android.intent.action.ACTION_POWER_CONNECTED 또는 android.intent.action.ACTION_POWER_DISCONNECTED 를 등록함으로써 수동적으로 취득될 수 있다. 수집-관련 모듈: 전력 매니저(224-a)(BatteryManager)
헤드셋 케이블 연결 스테이터스	시스템 브로드캐스트(ACTION_HEADSET_PLUG)가 모니터링되고, 연결 스테이터스 변경시에 최신 상태가 버퍼링된다. 수집-관련 모듈: 유선 액세스 매니저(224-g) (WiredAccessoryManager)
GPS 정보	위치 변경이 모니터링되고(LocationListener), 위치 변경시에 최신 위치 데이터가 버퍼링된다. 수집-관련 모듈: 위치 매니저(224-e)(LocationManager)
애플리케이션 타입	애플리케이션 패키지 이름의 속성 플래그 및 hwFlags 가 시스템 서비스의 PackageManager 에 의해 제공되는 getApplicationInfo 방법을 사용해서 취득되고, 두 속성을 결합함으로써 애플리케이션 타입(내장 타입이라고도 함)이 결정된다. 수집-관련 모듈: PackageManager
애플리케이션 레이아웃 정보	데스크톱 레이아웃 데이터베이스는 패키지 이름에 대응하는 애플리케이션의 데스크톱 레이아웃 정보에 대하여 질의하기 위해 액세스된다. 주: 사용자가 제 3 자 데스크톱 애플리케이션을 사용할 경우, 데스크톱 레이아웃 정보가 본 방법을 사용해서 취득될 수 없으므로, 현재 시스템 데스크톱 애플리케이션의 패키지 이름이 결정되어야 한다. 수집-관련 모듈: 뷰 시스템(224-i)(View System)

[0133]

[0134]

안드로이드 시스템은 기능 인터페이스를 제공하여 일부 데이터를 취득한다. 표 1은 데이터를 취득하기 위한 방법과 관련되는 모듈(예를 들어, PackageManager) 또는 기능(예를 들어, getActiveNotifications)을 설명한다. 도 2를 참조하면, 당업자가 요건에 따라 다른 취득 방식을 사용할 수 있다. 이는 본원에서 제한되지 않는다. 안드로이드 시스템과는 상이한 다른 시스템들에서 호출된 모듈들 및 기능들은 상이할 수 있으며, 본원은 안드로이드 시스템으로 제한되지 않는다.

[0135]

표 1에서의 포그라운드 애플리케이션은 현재 포그라운드에서 실행되는 애플리케이션이다. 이러한 애플리케이션은 현재 사용자에게 의해 사용되고 있고 사용자에게 상대적으로 중요한 애플리케이션으로서 간주된다. 사용자는 또한 백그라운드에서 실행되는 일부 애플리케이션, 예를 들어, 음악 재생 애플리케이션을 사용한다. 이 타입의 애플리케이션의 수집 시간은 이 타입의 애플리케이션이 기동된 순간, 즉, 이 타입의 애플리케이션이 처음으로 포그라운드로 전환된 순간으로 설정될 수 있다.

[0136]

전술한 데이터 수집 프로세스는 키 이벤트가 검출될 때 트리거된다. 본 명세서에서 키 이벤트는 이하의 이벤트, 즉 포그라운드-백그라운드 전환 이벤트, 애플리케이션 설치 이벤트, 애플리케이션 제거 이벤트, 시스템 스테이터스의 변경에 의해 야기되는 통지 이벤트, 또는 지리적 위치 정보의 변경에 의해 야기되는 통지 이벤트 중 하나 이상을 포함할 수 있다. 지리적 위치 정보가 변경된다는 것은 의미론적 지리적 위치가 변경되는지의 여부가 먼저 식별된다는 것, 예를 들어, 의미론적 지리적 위치가 집에서 사무실 위치로 변경되는지의 여부일 수도 있다. 데이터 수집은 의미론적 지리적 위치가 변경되면 시작된다. 키 이벤트는 전술한 실시예들로 제한되지 않는다. 요컨대, 키 이벤트는 주로 수집될 필요가 있는 정보에 의존한다. 수집될 필요가 있는 정보가 변경

될 수 있을 것으로 검출되면, 전술한 데이터 수집 프로세스가 시작되어야 할 수 있다.

- [0137] 수집된 데이터에 있어서, 애플리케이션이 포그라운드로 전환되는 시간은 현재 시스템 시간일 수 있다. 예를 들어, 데이터 수집은 포그라운드-백그라운드 전환 이벤트가 검출될 경우에 수행된다. 이 경우, 수집된 현재 시스템 시간은 현재 포그라운드 애플리케이션의 전환 시간으로 간주될 수 있다. 다른 이벤트의 트리거 하에서 데이터 수집이 수행되면, 전환 시간을 나타내는 데 사용되며 현재 포그라운드 애플리케이션이 전환될 때 기록되는 타임스탬프는 애플리케이션이 포그라운드로 전환된 시간을 취득하는 데 사용될 수 있다. 애플리케이션이 포그라운드로 전환될 경우, 타임스탬프는 그때의 시스템 시간을 사용해서 기록된다.
- [0138] 본원에서, "애플리케이션이 포그라운드로 전환되는 시간(a time when an application is switched to a foreground)"은 때때로 "애플리케이션이 사용되는 시간(a time when an application is used)"이라고도 한다.
- [0139] 수집된 데이터는 최종 사용 방식에 관하여 두 가지 타입으로 더 분류될 수 있다: 즉, (1) 직접 사용될 수 있는 데이터, 예를 들어, 블루투스/네트워크/헤드셋/충전 케이블 연결 스테이터스, 통지 목록 스테이터스, 및 표 1에 도시된 애플리케이션 타입과 같은 데이터— 데이터는, 특징 추출을 통해, 모델 트레이닝을 위한 입력 파라미터로서 직접 사용될 수 있음 —; (2) 주로 두 가지 타입, 즉, 애플리케이션 시퀀스 특징과 관련되는 정보 및 GPS 위치 정보를 포함하는, 처리될 필요가 있는 데이터— 데이터는 파라미터 입력을 위해 다른 형태로 변환되도록 더 처리될 필요가 있음 —. 타입 (2)는 주로 아래에서 상세하게 설명된다.
- [0140] GPS 위치 정보의 경우, 집 및 사무실 위치와 같이, 의미론적 위치는 군집화될 필요가 있다. 두 지리적 위치가 교환되면, 시스템은 의미론적 위치 정보가 변경되었음을 나타내는 브로드캐스트를 푸시한다. 일부 다른 실시형태들에 있어서는, GPS 위치 정보가 직접 사용될 수 있다.
- [0141] 데이터를 처리하는 프로세스는 데이터가 저장되기 전에 발생할 수 있거나, 또는 데이터가 모델 트레이닝에 사용되기 전에 데이터가 처리될 수 있다는 점을 이해해야 한다.
- [0142] 애플리케이션 시퀀스 특징과 관련되는 정보는 주로 포그라운드 애플리케이션의 패키지 이름 및 애플리케이션이 포그라운드로 전환된 시간을 포함한다. 두 가지 타입의 정보의 이력 기록(사용자에 의해 애플리케이션을 사용한 트랙 기록으로서 간주될 수 있음)을 수집함으로써, 이하의 세 가지 타입의 정보가 통계 수집을 통해 더 취득될 수 있다: (a) k_1 개의 최근에 사용된 애플리케이션; (b) 현재 포그라운드 애플리케이션들 중에서 k_2 개의 가장 가능성 있는 전단 애플리케이션; 및 (c) 현재 포그라운드 애플리케이션들 중에서 k_3 개의 가장 가능성 있는 후단 애플리케이션. k_1 , k_2 , 및 k_3 의 값은 같거나 또는 같지 않을 수 있다. 세 가지 타입의 정보, 즉, (a), (b), 및 (c)를 총괄하여 애플리케이션 시퀀스 특징 데이터라고 한다.
- [0143] (a)는 현재 포그라운드 애플리케이션이 사용되기 전에 사용자에게 의해 최근에 사용된 k_1 개의 애플리케이션을 포함하고, k_1 개의 애플리케이션은 저장된 이력 데이터에 있는 각각의 애플리케이션 및 각각의 애플리케이션의 사용 시간에 기초하여 직접 취득될 수 있다. 예를 들어, $k_1=2$ 이고, 현재 애플리케이션 사용 시간이 18:00이라고 가정하고, 각각의 애플리케이션의 사용 시간에 기초하여, 현재 애플리케이션 사용 시간 이전에 애플리케이션을 사용한 최근 시간이 15:45 및 15:30이라고 결정한다. 이 경우, k_1 개의 애플리케이션은 15:45 및 15:30에 사용된 2개의 애플리케이션이다. 일부 다른 실시형태들에 있어서, k_1 은 또한 현재 포그라운드 애플리케이션을 포함할 수 있다.
- [0144] (b) 및 (c)의 경우, 애플리케이션들 사이의 다단(multi-order) 연관 관계는 사용자에게 의해 애플리케이션을 사용하는 이력 트랙으로부터 통계 수집을 통해 수학적 방법을 사용해서 취득될 필요가 있다. 다단 연관 관계의 통계적 모델은 아래에서 설명된다.
- [0145] 이 실시형태에 있어서, 매트릭스 $U[M \times M]$ 은 단말 장치(100)에 인스톨된 M 개의 애플리케이션들 사이의 연관 관계를 나타내는 데 사용되고, $U[i \times j]$ 는 애플리케이션 i 에서 애플리케이션 j 로 직접 전환하는 횟수— i 및 j 는 각각 M 이하의 양의 정수임 —를 나타낸다. 예를 들어, 애플리케이션 i 를 애플리케이션 j 로 전환하는 동작이 특정 순간에 발생하면, 기록 $U[i \times j]$ 가 1 증가된다.
- [0146] 이는 포그라운드 애플리케이션에 대한 것이라는 점에 유의해야 한다. 구체적으로, 현재 포그라운드에 위치된 애플리케이션은 애플리케이션 i 에서 애플리케이션 j 로 직접적으로 변경된다. 다시 말해, 사용자는 애플리케이션 i 를 사용하고 나서 애플리케이션 j 를 사용하고, 즉, 애플리케이션 i 는 애플리케이션 j 로 직접적으로 전환된다.
- [0147] 또한, 애플리케이션간 점프의 수량도 고려될 수 있다. 애플리케이션 i 가 애플리케이션 j 로 직접적으로 전환되

면, 두 애플리케이션은 밀접하게 상관될 수 있음을 나타낸다. 그러나, 애플리케이션 i 가 몇 번 점프한 후에 애플리케이션 i 가 애플리케이션 j 로 전환되면, 두 애플리케이션은 가능한 연관성을 가질 수 있음을 또한 반영한다. 이러한 구현예에 기초하여, 점프 수량 파라미터가 추가될 수 있다. 예를 들어, 애플리케이션 i 가 d 번 점프한 후에 애플리케이션 i 가 애플리케이션 j 로 전환되면, 기록 $U[i,j][d]$ 가 1 증가된다. 점프의 수량이 지나치게 크면, 애플리케이션들 사이의 연관 관계가 매우 약하고 무시될 수 있음을 나타낸다. 따라서, D 점프의 최대치가 설정될 수 있다(예를 들어, D 의 값은 5임).

[0148] 이 경우, 애플리케이션 i 및 애플리케이션 j 의 근접 매트릭스에서 각각의 요소 $U'[i,j]$ 는 다음과 같이 정의될 수 있다:

$$U'[i,j] = \alpha U[i,j][0] + \beta U[i,j][1] + \dots + \gamma U[i,j][D] \quad (1)$$

[0150] 전술한 방법에 따르면, $U'[i,j]$ (매트릭스의 열 j)는 다른 애플리케이션에서 애플리케이션 j 로 점프할 가능성을 나타낼 수 있으며, $U'[i,j]$ (매트릭스의 행 i 를 나타냄)는 애플리케이션 i 에서 다른 애플리케이션으로 점프할 가능성을 나타낼 수 있다. 이 매트릭스로부터, 애플리케이션의 k_2 개의 가장 가능한 전단 애플리케이션 및 애플리케이션의 k_3 개의 가능 가능한 후단 애플리케이션이 취득될 수 있다. 예를 들어, 현재 포그라운드 애플리케이션이 애플리케이션 v 일 경우, k_2 최대값은 $M'[v]$ 로부터 연속적으로 선택될 수 있고, k_2 값들에 대응하는 행 값들은 애플리케이션 v 의 k_2 개의 가장 가능한 전단 애플리케이션이고; k_3 최대값은 $U'[v,]$ 로부터 연속적으로 선택되고, k_3 값들에 대응하는 열 값들은 애플리케이션 v 의 k_3 개의 가장 가능한 후단 애플리케이션이다.

[0151] 애플리케이션 시퀀스 특징 데이터는 다음 단계에서의 모델 트레이닝을 위해 다른 수집된 데이터와 함께 메모리(180)에 저장된다. 실시간 애플리케이션 순위화 프로세스에 있어서, 전술한 방법은 포그라운드 애플리케이션에 대응하는 애플리케이션 시퀀스 특징 데이터를 취득하기 위해서도 사용된다.

[0152] 도 6은 제2 단계에서의 모델 트레이닝 프로세스의 개략도이다. 일정 기간 동안 데이터 수집이 수행된 후에, 일부 데이터는 이미 메모리(180)에 저장되고, 해당 데이터는 애플리케이션을 사용하기 위한 규칙을 반영한다. 이러한 규칙은 머신 러닝을 통해 재사용 가능한 모델을 형성할 필요가 있다. 이는 모델 트레이닝 프로세스에서 수행된다. 도면에 도시된 바와 같이, 데이터가 메모리(180)로부터 판독되고(S601), 특징 추출(S602)과 같은 동작이 데이터에 대하여 수행되고, 이어서 차원-기반 분류 모델이 트레이닝되고, 즉 사용자에게 의한 애플리케이션 사용 습관이 하나 이상의 차원에 기초하여 분류된다(S603). 분류 이후, 순위화 모델들은 상이한 카테고리들에 대하여 별도로 트레이닝되고(S604), 트레이닝 결과는 제3 단계에서의 애플리케이션 순위화에서 사용하기 위해 메모리에 저장된다. 또한, 이 단계에서, 사용자에게 의해 빈번하게 사용되는 애플리케이션들의 수량 N 이 예측될 수 있다(S605). 세 가지의 관련 모델 알고리즘을 별도로 후술한다.

[0153] (a) 차원-기반 분류 모델의 트레이닝(S603)

[0154] 사용자에게 의한 애플리케이션 사용 습관은 몇몇 특정 차원에 있어서는 상이할 수 있다. 예를 들어, 시간 차원에서, 근무 기간 및 비-근무 기간(휴무 기간이라고 할 수도 있음)에 사용자에게 의한 애플리케이션 사용 습관은 현저하게 상이할 수 있다. 이 경우, 순위화 모델이 일반적으로 애플리케이션들을 순위화하도록 구성되면, 매우 큰 편차가 발생할 수 있다. 차원-기반 분류 모델의 트레이닝은 몇몇 차원에서 사용자의 습관의 세분화된 분류를 탐색하는 것이므로, 상기와 같은 차이에 의해 야기되는 에러가 제거되고, 순위화 정확도가 더욱 향상된다.

[0155] 하나 이상의 차원에 기초한 상이한 분류에 대하여, 분류 결과는 일부 분류 알고리즘을 원본 데이터에 적용함으로써 취득될 수 있다. 도 7에 도시된 바와 같이, 원본 데이터는 차원 X 에서 세 가지 카테고리로 분류된다. 또한, 각각의 카테고리는 차원 Y 에서 더 많은 카테고리로 더 분류될 수 있고, 세부 내용을 설명하지는 않는다. 이는 또한 후술하는 엔트로피 생성 알고리즘의 기본 원리이다.

[0156] 이 실시형태에 있어서, 시간 차원이 예로서 사용되고, 시간은 두 가지 카테고리, 즉, 근무 기간 및 비-근무 기간으로 분류된다. 하루의 시간에 기초하여, 방법은 하나 또는 두 개의 시점을 찾아서 사용자의 근무 시간과 비-근무 시간을 구별하는 것이다. 이하의 두 분류 방식은 특정 구현예에 포함될 수 있다: (1) 단일-라인 분류, 즉, $[0:00-x]$ 및 $[x-24:00]$ 형태의 분류, 여기서는 단 하나의 변수 x 가 존재함; 및 (2) 이중-라인 분류, 즉, $[0:00-x_1]$, $[x_1-x_2]$, 및 $[x_2-24:00]$ 형태의 분류, 여기서는 두 개의 변수 x_1 및 x_2 가 존재함. 이중-라인 분류는 아래의 실시예로서 사용된다.

[0157] 데이터가 먼저 메모리(180)로부터 판독된다. 본 명세서에서 사용되는 데이터는 주로 애플리케이션의 패키지 이름 및 애플리케이션이 사용된 시간을 포함한다.

[0158] 하루의 시간은 24개의 기간으로 분할되고, 각 기간은 1시간을 포함한다. 판독된 데이터에 기초하여, 사용자에게 의한 애플리케이션의 사용 스테이터스에 관한 통계가 모든 기간에 수집된다. 이 실시형태에 있어서, 임의의 기간에 애플리케이션의 사용 스테이터스에 관한 통계가 2차원 매트릭스 S를 사용해서 수집된다. 예를 들어, S[h][i]는 기간 h에 애플리케이션 i를 사용한 총 횟수를 나타낸다. 따라서, 기간 [h₁,h₂]에 애플리케이션 i를 사용한 총 횟수 Sum_{[h₁,h₂](i) 및 모든 애플리케이션의 사용 횟수에 대한 기간 [h₁,h₂]에 애플리케이션 i를 사용한 횟수의 비 f_{[h₁,h₂](i)}는 이하의 수식 (2) 및 (3)을 사용해서 계산을 통해 제각기 취득될 수 있다:}

$$Sum_{[h_1, h_2]}(i) = \sum_{h \in [h_1, h_2]} S[h][i] \quad (2)$$

$$f_{[h_1, h_2]}(i) = Sum_{[h_1, h_2]}(i) / \sum_i Sum_{[h_1, h_2]}(i) \quad (3)$$

[0160] 여기서, f_{[h₁,h₂](i)}를 기간 [h₁,h₂]에 애플리케이션 i를 사용하기 위한 빈도라고 할 수도 있다.

[0161] 따라서, 기간 [h₁,h₂]에 사용자에게 의해 애플리케이션을 사용하는 정보 엔트로피 E_[h₁,h₂]가 다음과 같이 정의된다:

$$E_{[h_1, h_2]} = \sum_i f_{[h_1, h_2]}(i) * (-1) * \log_2 f_{[h_1, h_2]}(i) \quad (4)$$

[0162] 이렇게 해서, x₁ 및 x₂에 기초한 하나의 이중-라인 분류에 의해 생성되는 엔트로피가 다음과 같이 계산된다:

$$E(x_1, x_2) = f(0, x_1) * E_{[0, x_1]} + f(x_1, x_2) * E_{[x_1, x_2]} + f[x_2, 23] * E_{[x_2, 23]} \quad (5)$$

[0163] 여기서, f(x₁,x₂)는 하루 종일 모든 애플리케이션을 사용한 횟수에 대한 기간 [x₁,x₂]에 모든 애플리케이션을 사용한 횟수의 비이고, 다음과 같이 계산된다:

$$f(x_1, x_2) = \sum_{h \in [x_1, x_2]} \sum_i S[h][i] / \sum_{h \in [0, 23]} \sum_i S[h][i] \quad (6)$$

[0164] 따라서, 문제의 해결은 이 분류의 엔트로피 값을 최소화하기 위해 마지막으로 두 분류 시간 x₁ 및 x₂를 찾는 것으로 변환된다, 즉:

$$\arg \min E(x_1, x_2) \quad (7)$$

[0165] 예를 들어, 마지막으로 취득되는 결과는 x₁ 및 x₂가 제각기 9:00 및 18:00이라는 것일 수 있다. 세 기간은 비-근무 기간 [0:00-9:00], 근무 기간 [9:00-18:00], 및 비-근무 기간 [18:00-24:00]이다. 이는 다음과 같이 표 2에 의해 표시된다:

[0166] 표 2

[0:00-9:00]	비-근무
[9:00-18:00]	근무
[18:00-24:00]	비-근무

[0167] 이 실시형태에 있어서, [0:00-9:00] 및 [18:00-24:00]는 모두 비-근무 기간으로 간주되고, 순위화 모델은 비-근무 기간 동안 트레이닝된다. 일부 다른 실시형태들에 있어서, 순위화 모델은 [0:00-9:00] 및 [18:00-24:00] 각각에 대하여 별도로 트레이닝될 수 있다. 또한, "근무(working)" 및 "비-근무(non-working)"의 의미론적 의미는 대부분의 사람들의 현재의 생활 습관에 기초하여 주어진 것이라는 점에 유의해야 한다. 실제로, 이 방법의 의미는 두 가지 상이한 규칙이 애플리케이션 사용 이력에 표시됨을 반영하는 것이다. "근무" 및 "비-근무"와 같은 의미론적 분류가 본원에서 반드시 사용되어야 하는 것은 아니다.

[0168] (b) 순위화 모델의 트레이닝(S604)

[0169] 단말 장치 상의 애플리케이션의 중요도는 애플리케이션의 사용 확률과 관련되고, 사용 확률이 높을수록 중요도

가 더 높다는 것을 나타낸다. 애플리케이션의 사용 확률은 시스템의 상태에 따라 상이할 수 있다. 예를 들어, 네트워크 연결 상태에서의 각각의 애플리케이션의 사용 확률은 네트워크 연결해제 상태에서의 애플리케이션의 사용 확률과 다르다. 이 경우, 네트워크 연결 경우의 애플리케이션 중요도 순위는 네트워크 연결해제 경우의 애플리케이션 중요도 순위와 다르다. 따라서, 애플리케이션들이 순위화될 때 현재 시스템 스테이터스가 고려될 필요가 있다.

[0176] 네트워크 연결 상태 또는 네트워크 연결해제 상태는 단지 예시일 뿐이며, 단말 장치와 관련되는 다른 상태, 예를 들어, 헤드셋이 연결되는지의 여부, 블루투스가 연결되는지의 여부, 충전 케이블이 연결되는지의 여부, 또는 단말 장치의 현재 위치가 또한 고려될 수 있다는 점에 유의해야 한다.

[0177] 요컨대, 순위화 모델 트레이닝은 주로 이하의 몇 가지 단계를 포함한다.

[0178] 데이터 판독(data read): 데이터가 메모리(180)로부터 판독된다. 본 명세서에서 트레이닝에 사용될 필요가 있는 데이터는 주로 애플리케이션 사용 시간, 애플리케이션 시퀀스 특징 정보, 시스템 스테이터스 정보, 및 의미론적 위치 정보를 포함한다. 애플리케이션 시퀀스 특징 정보의 내용 및 취득 프로세스는 위에서 상세하게 설명되었다. 시스템 스테이터스 정보는 주로 헤드셋, 충전 케이블, 및 네트워크의 연결 스테이터스를 포함한다. 예를 들어, 의미론적 위치 정보는 집 및 사무실을 포함할 수 있다.

[0179] 특징 추출(feature extraction): 통계 수집, 식별, 벡터화 등을 통해, 판독된 데이터가 머신 러닝 알고리즘에 의해 처리될 수 있는 특징 벡터/특징 매트릭스로 변환된다. 이 단계는 구체적으로 다음을 포함한다: (s1) 애플리케이션 사용 시간 및 애플리케이션 시퀀스 특징 정보에 대하여, 애플리케이션 사용 시간 및 애플리케이션 시퀀스 특징 정보는 벡터화, 타임 슬라이스, 및 근무일/비-근무일 식별을 통해 벡터로 변환된다. 예를 들어, 애플리케이션 사용 시간은 두 부분, 즉 날짜 및 시간으로 분할된다. 월요일, 화요일, ..., 및 일요일은 제각기 0, 1, ..., 및 6에 맵핑된다. 시간들은 세그먼트에 기초하여 맵핑된다. 예를 들어, [9:00-18:00]은 세그먼트이고, 해당 시간 내의 모든 시간은 동일한 번호에 맵핑된다. (s2) 시스템 스테이터스 정보에 대하여, 판독된 시스템 스테이터스 정보로부터 특징이 추출되고, 시스템 스테이터스 정보는 이산법/열거법을 사용해서 벡터로 변환된다. (s3) 의미론적 위치 정보에 대하여, 판독된 의미론적 위치 정보가 인코딩을 통해 벡터로 변환된다. (s4) 정규화 처리가 수행된다. [0, 1] 정규화는 최대값/최소값 방법을 사용해서 값에 대하여 수행되고, 정규화는 백색화(whitening) 방법을 사용해서 값의 분산에 대하여 수행된다. (s6) 특징 매트릭스 조합이 수행된다. 모든 차원의 데이터가 결합되어 특징 매트릭스를 형성하고, 특징 매트릭스가 알고리즘 트레이닝 모듈에 제공된다.

[0180] 모델 트레이닝 및 저장(model training and storage): 데이터가 복수의 그룹으로 분할되고, 복수의 데이터 그룹 각각에 대하여 모델을 트레이닝하여 복수의 모델을 취득한다. 데이터는 대안적으로 특징 추출 전에 분할될 수 있고, 특징 추출은 데이터가 분할된 후에 별도로 수행된다. 예를 들어, 본원의 전술한 실시형태에서 제공되는 시간 차원에 기초한 분류에 따르면, 월요일부터 금요일까지 [9:00-18:00]의 데이터 및 월요일부터 금요일까지 [0:00-9:00] 및 [18:00-24:00]의 데이터에 대하여 2개의 모델이 별도로 트레이닝되고, 생성된 순위화 모델(머신 러닝 모델의 파라미터 정보) 및 상응하는 기간이 데이터베이스에 저장된다. 토요일 및 일요일 전체의 데이터는 비-근무 기간의 데이터로서 트레이닝될 수 있다.

[0181] 저장 결과의 예는 다음과 같다:

[0182] 표 3

[9:00-18:00] (근무 시간)	머신 러닝 모델 1의 파라미터 정보
[0:00-9:00] [18:00-24:00] (비- 근무 시간)	머신 러닝 모델 2의 파라미터 정보

[0183]

[0184] 아래에서는 머신 러닝 알고리즘 소프트맥스(Softmax)를 예로서 사용해서 트레이닝 프로세스를 상세하게 설명한다. 소프트맥스 알고리즘은 해당 알고리즘이 증분식 학습 알고리즘이기 때문에 선택된다. 증분식 학습 알고리즘은, 트레이닝 프로세스에서 새로운 트레이닝 샘플의 특징 데이터가 원래 학습된 모델 파라미터에 기초하여 병합되어, 파라미터를 더 최적화하는 것, 즉, 원본 샘플 데이터가 유지될 필요가 없음이 보장될 수 있다는 것을 의미한다. 이렇게 해서, 데이터의 저장량이 감소될 수 있고, 저장 공간이 절약될 수 있다.

[0185] 통계 수집, 식별, 벡터화 등을 통해, 애플리케이션 시퀀스 특징 정보 및 시스템 스테이터스 정보와 같은 취득된 데이터가 소프트웨어 알고리즘에 의해 처리될 수 있는 특징 벡터 $s=(s_1, s_2, \dots, s_W)$ 로 변환— W 는 특징 벡터의 수량(또는 길이)임—된다. 애플리케이션 세트는 벡터 $a=(a_1, a_2, \dots, a_M)$ 를 사용해서 표현되고, a_j 는 j 번째 애플리케이션을 나타내고, M 은 휴대폰에インストール된 애플리케이션의 수량을 나타낸다. 목적은 순위화 함수(즉, 순위화 모델) $h_\theta(s)$ 를 계산하는 것이다. 분명히, 순위화 함수를 계산하는 전제는 파라미터 θ 를 계산하는 것이다.

[0186] 따라서, 소프트웨어 알고리즘에 기초하여, 순위화 함수가 다음과 같이 정의된다:

$$h_\theta(s) = \frac{1}{\sum_{i=1}^M e^{\theta_i s}} \begin{bmatrix} e^{\theta_1 s} \\ \vdots \\ e^{\theta_M s} \end{bmatrix} \quad (8)$$

[0188] 이 모델의 비용 함수는 다음과 같다:

$$J(\theta) = -\frac{1}{k} \left[\sum_{i=1}^k \sum_{j=1}^M 1\{y^{(i)} = j\} \log \frac{e^{\theta_j x^{(i)}}}{\sum_{l=1}^M e^{\theta_l x^{(i)}}} \right] + \frac{\lambda}{2} \sum_{i=1}^M \sum_{j=1}^W \theta_{ij}^2 \quad (9)$$

[0190] 여기서, k 는 트레이닝을 위해 입력된 샘플의 수량을 나타내고; M 은 카테고리의 수량, 즉, 애플리케이션의 수량을 나타내고; W 는 특징 벡터 s 의 길이를 나타내고; λ 는 가중치($\lambda > 0$)를 나타내고; $1\{.\}$ 은 특성 함수이고, 특성 함수의 값 규칙은 다음과 같다:

[0191] $1\{\text{표현식이 참}\}=1$

[0192] $1\{\text{표현식이 거짓}\}=0$

[0193] $y^{(i)} = j$ 는 i 번째 샘플에서의 애플리케이션 패키지 이름의 식별자가 벡터 a 에서 j 인 것을 식별한다. 따라서, i 번째 샘플에서의 애플리케이션 패키지 이름의 식별자가 벡터 a 에서 j 인 것이 참이면, $1\{y^{(i)} = j\}=1$ 이거나; 또는 i 번째 샘플에서의 애플리케이션 패키지 이름의 식별자가 벡터 a 에서 j 인 것이 거짓이면, $1\{y^{(i)} = j\}=0$ 이다.

[0194] 모델 트레이닝 동안 기울기 하강 방법 또는 뉴턴 방법을 사용해서 $J(\theta)$ 를 최소화하여 θ 를 계산한다. 기울기 하강 방법은 다음과 같다:

$$\theta'_j = \theta_j - \alpha \nabla_{\theta_j} J(\theta) \quad (10)$$

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{k} \sum_{i=1}^k \left[x^{(i)} (1\{y^{(i)} = j\} - \frac{e^{\theta_j x^{(i)}}}{\sum_{l=1}^M e^{\theta_l x^{(i)}}}) \right] + \lambda \theta_j \quad (11)$$

[0197] 여기서 α 는 고정 파라미터이고, j 는 반복 수량을 나타낸다.

[0198] 이 실시형태에 있어서, 식별 모델이 사용자의 실제 사용 습관에 점점 가까워지도록, 24시간마다 트레이닝이 수행되어 식별 모델을 지속적으로 업데이트한다.

[0199] 파라미터 θ 가 전술한 모델 트레이닝을 통해 계산된 후에, 순위화 함수 $h_\theta(s)$, 즉, 실시간 순위화 동안 사용되는 순위화 함수를 알게 된다. 계산을 통해 취득되는 정보는 실시간 순위화 동안 사용하기 위해 표 3의 형태로 메모리(180)에 저장된다.

[0200] (c) N-값 예측 알고리즘(S605)

[0201] N의 값은 양의 정수이고, 특정 분류에서 사용자에게 의해 빈번하게 사용되는 애플리케이션의 수량, 예를 들어, 특정 기간(예를 들어, 근무 기간)에 사용자에게 의해 빈번하게 사용되는 애플리케이션의 수량을 반영한다. N-값 예측 알고리즘은 N의 값을 결정하는 데 사용된다. N의 값은 리소스 관리에 사용될 수 있다. N개의 애플리케이션이 사용자에게 의해 빈번하게 사용되기 때문에, N개의 애플리케이션의 리소스는 리소스 관리 동안 적절히 보호될 수 있다.

[0202] 전술한 설명에 기초하면, 이 실시형태에 있어서, 사용자의 사용 규칙은 표 3에 도시된 바와 같이 시간을 차원으로 사용해서 2가지 카테고리로 분류된다. 이 경우, 시간을 차원으로 사용해서 N의 값을 계산할 때에도 동일한 분류가 수행될 수 있다. 출력 결과는 표 4에 도시되고: N의 값들은 근무 기간 및 비-근무 기간에 대하여 별도로 계산된다. 이렇게 해서, N의 값의 예측이 더욱 정확하다.

[0203] 일부 다른 실시형태들에 있어서, 모든 카테고리에 대하여 단 하나의 N의 값이 계산될 수 있거나, 또는 N의 값을 계산하기 위한 분류 방식은 표 3에서의 분류 방식과 다르다. 일부 다른 실시형태들에 있어서, 위치 차원과 같은 비-시간 차원을 사용해서 N의 값을 계산할 수 있다. 예를 들어, N의 값은 집과 관련된 데이터에 대하여 계산되고, N의 값은 사무실 위치와 관련된 차원에 대하여 계산된다.

[0204] N의 값은 사전에 계산 및 저장될 수 있고, 예를 들어, 표 4에 저장될 수 있거나; 또는 N의 값은, N의 값이 리소스 관리에 필요해질 때, N-값 예측 알고리즘을 실시간으로 호출해서 취득될 수 있다.

[0205] 표 4

[9:00-18:00] (근무 기간)	머신 러닝 모델 1 의 파라미터 정보	N의 값
[0:00-9:00] [18:00-24:00] (비-근무 기간)	머신 러닝 모델 2 의 파라미터 정보	N의 값

[0206]

[0207] N-값 예측 알고리즘의 구현예에 있어서, 전날의 또는 여러 날의 동일한 기간에 애플리케이션들의 사용 기록이 취득되고, 각각의 애플리케이션의 사용 시간의 양에 관한 통계가 수집되고, 이어서 최대량의 사용 시간을 갖는 V개의 애플리케이션이 사용 시간의 내림차순의 양에 기초하여 연속으로 확인된다. V개의 애플리케이션의 사용 시간의 총량이 해당 기간에 모든 애플리케이션의 사용 시간의 총량의 90%에 도달할 수 있으면, N=V이다.

[0208] N-값 예측 알고리즘의 다른 구현예는 이차 이동 평균이다. t번째 날의 특정 기간(또는 다른 차원의 카테고리)에 보호될 필요가 있는 애플리케이션의 수량 N_t 는, 수식 (12)에 도시된 바와 같이, (t-1)번째 날 및 (t-2)번째 날의 동일한 기간에 보호될 필요가 있는 애플리케이션들의 수량의 평균에, 애플리케이션들의 수량의 분산 가중치를 더한 것이다.

$$N_t = \frac{(N_{t-1} + N_{t-2})}{2} + \alpha * \sigma \quad (12)$$

[0209]

[0210] 여기서, α 는 가중 계수이고, σ 는 데이터에서의 모든 N_i 의 분산 값이다. 예를 들어, α 의 값은 0.005이다. N의 값은 전술한 구현예를 사용해서 첫날 및 둘째 날 동안 계산될 수 있다.

[0211] N의 값이 수식 (12)에 기초하여 계산될 때마다, 모든 이력 N_i 의 분산 값 σ 가 계산될 필요가 있으므로, 모든 이력 N_i 가 저장될 필요가 있다. 이 실시형태에 있어서, 이력 N_i 의 저장량을 줄이기 위해, 증분 계산 방법을 사용해서 모든 N_i 의 분산 값을 평가한다:

$$\sigma^2 = \frac{P[\sigma_H^2 + (\bar{X} - \bar{H})^2] + Q[\sigma_A^2 + (\bar{X} - \bar{A})^2]}{P + Q} \quad (13)$$

[0212]

[0213] 여기서, $\bar{X}$ 는 모든 데이터의 평균이고, σ_H^2 는 이력 데이터의 분산이고, $\bar{H}$ 는 이력 데이터의 평균이고, P는 이력 데이터의 수량이고, σ_A^2 는 새롭게 추가된 데이터의 분산이고, $\bar{A}$ 는 새롭게 추가된 데이터의 평균이고, Q는 새롭게 추가된 데이터의 수량이다.

[0214] 도 8은 제3 단계에서의 실시간 애플리케이션 순위화 프로세스의 개략도이다.

[0215] 실시간 순위화 모듈(320)은 실시간 순위화를 트리거하는 키 이벤트를 검출하기 위해 이벤트 모니터링 메커니즘을 설정한다(S801). 키 이벤트를 검출(S802)한 후에, 실시간 순위화 모듈(320)은 데이터 수집 모듈(221)을 트리거해서 실시간 데이터를 수집한다(S803). 실시간 데이터는 단말 장치의 시스템 시간, 단말 장치의 현재 스테

이터스 데이터, 및 단말 장치의 현재 위치 데이터를 포함할 수 있다.

- [0216] 이하의 몇몇 키 이벤트가 포함된다: (1) 포그라운드-백그라운드 전환; (2) 애플리케이션 설치 및 제거; (3) 시스템 스테이터스의 변경, 즉, 헤드셋/네트워크의 연결 스테이터스의 변경; (4) 지능형 시나리오 브로드캐스트 통지, 즉, 의미론적 지리적 위치(회사/집)가 변경됨을 나타내는 통지의 수신 등.
- [0217] 시간 도메인에서 단말 장치가 현재 위치된 카테고리, 즉, 근무 기간 또는 비-근무 기간은 단말 장치의 수집된 시스템 시간 및 차원-기반 분류 모델의 트레이닝 결과(표 2)에 기초하여 결정된다(S804). 관련된 분류의 순위화 모델이 선택되거나 또는 순위화 모델 트레이닝 결과에 기초하여 모델 파라미터만이 취득되고(표 3), 이어서 실시간 시나리오에서 모든 애플리케이션의 중요도 순위가 이력 데이터, 실시간으로 수집된 데이터 등에 기초하여 예측된다(S805). 본 명세서에서 이력 데이터는 이력 수집 및 저장된 데이터이다. 상세하게는, 표 1을 참조한다.
- [0218] 순위화 모델에 입력될 데이터가 사전 처리되어야 할 수도 있음을 이해해야 한다. 예를 들어, 현재 포그라운드 애플리케이션과 관련되는 시퀀스 특징 데이터는 수집된 실시간 데이터 및 일부 이력 데이터에 대한 통계를 수집함으로써 취득될 필요가 있다. 의미론적 위치 데이터가 모델에 입력될 필요가 있으면, 실시간으로 수집된 단말 장치의 위치 정보가 군집화를 통해 의미론적 위치 데이터로 변환될 필요가 있거나, 또는 상응하는 의미론적 위치 데이터가 GPS 위치 범위와 의미론적 위치 사이의 미리 설정된 대응관계에 기초하여 취득된다. 사전 처리가 필요한지의 여부, 및 모델 트레이닝에 맞게 어떠한 사전 처리가 수행될 필요가 있는지는 본원에서 구체적으로 제한되지 않는다.
- [0219] 표 2 및 표 3은 단지 이해를 돕기 위한 예시일 뿐이라는 점에 유의해야 한다. 구체적인 구현 프로세스에 있어서, 두 테이블은 하나의 테이블, 즉 하나의 대응관계로 결합될 수 있다. 또한, 순위화 모델 또는 모델 파라미터는 시스템의 현재 시간에 기초하여 한 번에 결정될 수 있다.
- [0220] 이 실시형태에 있어서, 실시간으로 수집된 시스템 시간은 S804에서 분류에 사용된다는 점에 유의해야 한다. 다른 실시형태에 있어서, 다른 차원이 분류에 사용되면, 분류는 실시간으로 수집되는 다른 데이터 및/또는 이력 데이터를 사용해서 요건에 따라 수행될 수 있다.
- [0221] 실시간 순위화 모듈(320)은 마지막으로 제4 단계에서의 리소스 관리 동안 질의를 위한 순위화 결과(S806)를 저장한다. 일부 다른 실시형태들에 있어서, 제3 단계에서의 순위화는 제4 단계에서 실시간으로 순위화가 호출되고 나서, 순위화 결과가 실시간으로 반환될 경우에만 발생할 수 있다.
- [0222] 또한, 순위화가 수행된 후에, 해당 기간에 사용자에게 의해 빈번하게 사용된 애플리케이션의 수량 N 이 N -값 예측 알고리즘 또는 미리 저장된 대응관계(예를 들어, 표 4)를 사용해서 취득될 수 있고, 이는 도 8에는 도시되지 않는다. 이 단계는 선택 사항이고, N 의 값은, 리소스 관리에 N 의 값이 필요해질 경우에, 실시간으로 계산될 수 있다.
- [0223] 애플리케이션들이 전술한 방법을 사용해서 순위화될 때, 사용자에게 의해 최근에 휴대폰에 인스톨된 새로운 애플리케이션의 경우에는, 매우 소량의 데이터가 수집되기 때문에 해당 애플리케이션에 대하여 평가된 중요도는 매우 낮을 수 있다. 따라서, 본원은 새롭게 인스톨된 애플리케이션에 보충 메커니즘을 더 제공하고, 새롭게 인스톨된 애플리케이션들은 스코어(score)를 계산함으로써 순위화된다.
- [0224] 새롭게 인스톨된 애플리케이션들의 순위화에는 두 가지 팩터가 주로 고려된다: (1) 사용 가능성 가중치, 사용 가능성 가중치는 본 실시형태에서는 LRU 가중치라고 할 수도 있고; 현재 새롭게 인스톨된 애플리케이션이 LRU (가장 최근에 사용된) 목록에 나타나는지의 여부를 검사해서, 새롭게 인스톨된 애플리케이션이 최근에 사용되었는지의 여부를 결정하고; 새롭게 인스톨된 애플리케이션이 LRU 목록에 나타나면, LRU 는 1이거나, 또는 새롭게 인스톨된 애플리케이션이 LRU 목록에 나타나지 않으면, LRU 는 0이고; (2) 시간 감쇠 가중치, 현재 시간과 새롭게 인스톨된 애플리케이션이 인스톨된 시간 사이의 시간차의 감쇠 가중치가 계산된다. LRU 목록은 최근에 사용된 복수의 애플리케이션의 식별자를 저장한다. 예를 들어, 목록의 길이가 8이면, 최근에 사용된 8개의 애플리케이션이 시간 시퀀스로 저장되고, 새롭게 사용된 애플리케이션이 삽입될 때마다, 가장 오래 전에 사용된 애플리케이션이 삭제된다.
- [0225] 두 팩터에 기초하여, 새롭게 인스톨된 애플리케이션의 스코어가 다음과 같이 정의된다:

$$Score = \alpha_1 \times LRU + \alpha_2 \times e^{-t} \quad (14)$$

- [0227] 여기서 α_1 은 LRU 가중 계수이고, α_2 는 시간 감쇠 가중 계수이고, t 는 현재 시간과 애플리케이션이 인스톨된 시간 사이의 시간차의 이산 값이다. 새롭게 인스톨된 모든 애플리케이션의 스코어가 계산되고 순위화된다.
- [0228] 일부 다른 실시형태들에 있어서, 새롭게 인스톨된 애플리케이션이 최근에 사용되었는지의 여부를 결정하기 위해 다른 형태가 사용될 수 있다. 예를 들어, 새롭게 인스톨된 애플리케이션의 최종 사용 시간이 기록되고, 최종 사용 시간과 현재 시간 사이의 시간차가 임계치보다 작은지의 여부가 결정된다.
- [0229] 사용자에게 의해 빈번하게 사용된 N 개의 애플리케이션이 추천될 필요가 있을 경우, 상위에 순위화된 $N-x$ 개의 애플리케이션이 소프트웨어 알고리즘을 사용해서 취득되는 순위화 결과에 기초하여 추천되고, x 개의 애플리케이션이 새롭게 인스톨된 애플리케이션들의 순위화 결과에 기초하여 추천되므로, 두 가지 추천 방식을 사용해서 N 개의 애플리케이션이 모두 추천된다. N 개의 애플리케이션은 시스템에서 현재 가장 중요한 애플리케이션으로서 사용되고, 후속 리소스 관리를 위해 사용된다.
- [0230] x 개의 새롭게 인스톨된 애플리케이션은 최대 스코어를 가진 연속적으로 선택되는 x 개의 애플리케이션일 수 있으며, x 의 값은 요건에 따라 선택될 수 있고, 예를 들어, 2일 수 있다. 대안으로서, 다음과 같은 조건이 설정된다: 즉, 새롭게 인스톨된 애플리케이션의 추천된 스코어는 임계치(예를 들어, 0.5)보다 커야 하고, 및/또는 추천된 애플리케이션의 최대 수량은 2이다. 조건을 구체적으로 어떻게 설정할지는 본원에서 제한되지 않는다.
- [0231] 제4 단계에서의 시스템 리소스 관리의 실시형태를 아래에서 설명한다. 아래에서 설명되는 방법은 의사결정 실행 모듈(223)에 의해 부분적으로 또는 완전히 구현될 수 있다.
- [0232] 시스템 리소스 관리의 주된 목적은 포그라운드 애플리케이션 또는 높을 중요도를 가진 애플리케이션에 대한 리소스 공급을 보장하는 것이다. 이 실시형태는, 필요한 리소스 수량이 상대적으로 클 경우에 중요하지 않은 일부 프로세스를 일시적으로 정지시키기 위해, 애플리케이션 프로세스를 일시적으로 정지(순간적으로 정지하는 것이라고도 할 수 있음)시키는 방법을 제공하고, 이로써 중요한 프로세스를 사용하기 위한 더욱 충분한 리소스가 제공되고, 불충분한 리소스에 의해 야기되는 애플리케이션 중지로 인해 사용자 경험에 영향을 미치는 것이 회피된다.
- [0233] 프로세스 정지(process freezing)는 일반적으로 프로세스가 중단되지 않는 슬립(sleep) 상태 또는 멈춤(stop) 상태로 설정되고 대기 큐(wait queue)에 배치되는 절차이다. "프로세스"는 실행되는 애플리케이션의 인스턴스로서 이해될 수 있으며, 이 둘은 일부 설명에서 동일하게 이해될 수 있다.
- [0234] 종래의 데스크톱 컴퓨터 운영 체제에서, 정지 기술은, 시스템이 동면(hibernate) 또는 일시정지(suspend)되는 경우에 주로 프로세스 목록의 각각의 프로세스의 스테이터스를 슬립 상태 또는 멈춤 상태로 설정하고, 모든 프로세스의 맥락을 하드 디스크에 저장하는 데 사용된다. 이렇게 해서, 시스템이 동면 상태 또는 일시정지 상태에서 복원될 경우, 시스템은 모든 정지된 프로세스를 정지해제함으로써 이전의 실행 상태를 복원할 수 있다.
- [0235] 상이한 단말 운영 체제들은 상이한 정지 기술들을 사용한다. 일부 단말 운영 체제는 의사 백그라운드 기술을 사용한다. 구체적으로, 애플리케이션 프로세스가 백그라운드로 전환되고 일정 기간 동안 백그라운드에서 실행된 후에, 프로세스가 정지되고, 사용자가 애플리케이션을 포그라운드로 다시 전환한 후에만 프로세스가 정지해제되고 계속해서 실행된다. 안드로이드 운영 체제의 메모리가 부족할 경우, OOM(out of memory) 모듈 또는 LMK(low memory killer) 모듈이 메모리 재이용 기능을 트리거하여 프로세스를 강제 종료해서 메모리를 재이용한다. 그러나, 결과적으로, 강제 종료된 애플리케이션을 재기동하는 시간이 더 길어진다. 또한, 사용자의 원래의 상태는 저장될 수 없고, 사용자 경험이 저하된다. 따라서, 정지 기술이 도입되어 이 문제를 해결한다. 구체적으로, 시스템의 메모리가 부족할 경우(예를 들어, 잔여 메모리 용량이 임계치보다 적음), 프로세스가 먼저 정지되고, 프로세스의 맥락이 ROM으로 전환되고, 메모리가 해제된다. 이렇게 해서, 애플리케이션이 다음에 재기동될 때, 프로세스가 정지해제되고 계속 실행된다. 이는, 애플리케이션이 강제 종료되는 것으로 인해 원래의 상태가 손실되는 것을 방지할 뿐만 아니라, 애플리케이션의 재기동 시간을 감소시키고, 그에 따라 사용자 경험이 향상된다.
- [0236] 결론적으로, 기존의 정지 기술은 주로 장기간 동안 사용되지 않은 프로세스를 처리하거나, 또는 장기간 동안 리소스를 해제하는 데 사용되고, 프로세스는 사용자가 프로세스를 필요로 할 때에만 정지해제된다.
- [0237] 이 실시형태에서는 순간적인 정지 기술이 제공된다. 다른 애플리케이션 프로세스는 일시적으로 정지되어, 중요한 애플리케이션 또는 중요한 시나리오에 필요한 리소스가 제 시간에 공급되는 것을 일시적으로 보장한다. 또한, 필요한 리소스의 양이 감소된 후에, 이러한 정지된 애플리케이션 프로세스가 다시 자동으로 복원되고 정지

해제된다. 이렇게 해서, 일부 다른 애플리케이션들이 영구적으로 정지된 후에 야기되는 경험 저하를 방지하면서 애플리케이션의 처리 및 응답 속도가 증가되고(예를 들어, 장기간 동안 다운로드가 정지된 후에, 다운로드가 계속 수행될 수 없음), 그에 따라 사용자 경험이 향상된다.

[0238] 상대적으로 다량의 리소스를 순간적으로 필요로 하는 특정 이벤트(예를 들면, 애플리케이션 기동, 활동 전환, 촬영, 슬라이딩, 스크린-온/오프, 및 갤러리 줌 작업)가 트리거될 경우, 중요하지 않은 애플리케이션의 실행에 의해 야기되는 이들 특정 이벤트에 대한 영향을 피하기 위해, 중요하지 않은 애플리케이션은 일시적으로 강제로 정지되고, 키 이벤트가 완료된 후에, 일시적으로 정지된 일부 애플리케이션의 실행이 애플리케이션의 중요도 순위에 기초하여 복원된다.

[0239] 본 명세서에서 "특정 이벤트"는 일반적으로 중요한 애플리케이션과 관련된다. 예를 들어, 중요한 애플리케이션은 포그라운드 애플리케이션 또는 민감한 백그라운드 애플리케이션이다. 예를 들어, 민감한 백그라운드 애플리케이션은 백그라운드에서 실행되는 음악 재생 애플리케이션이다. 일시적으로 정지될 중요하지 않은 애플리케이션은 요건에 따라 선택될 수 있다. 예를 들어, N개의 제1 애플리케이션 이외의 모든 애플리케이션은 전술한 실시형태에서 제공되는 애플리케이션 중요도 순위화 결과에 기초하여 선택되고 일시적으로 정지되거나(본 명세서에서 N은 전술한 실시형태에서 계산된 사용자에게 의해 빈번하게 사용된 애플리케이션의 수량임), 또는 모든 백그라운드 애플리케이션은 일시적으로 정지되도록 선택되거나, 또는 요건에 따라 선택되고 결정에 기초한 현재 중요하지 않은 다른 애플리케이션이거나 또는 예측에 기초한 일정 기간 내에 중요하지 않은 애플리케이션이다.

[0240] 도 9에 도시된 바와 같이, 검출 기능은 특정 이벤트를 모니터링하고(S901), 활동 매니저 서비스(Activity Manager Service, AMS)의 도팅(dotting) 보고 기능을 사용해서 특정 이벤트를 보고하고(S902), 일부 애플리케이션을 일시적으로 정지시킨다(S903). 일시적인 정지 함수가 호출될 경우에, 정지 시간, 예를 들어, 1.5s가 설정된다. 정지 시간은 미리 설정될 수 있고 변경될 수 없거나, 또는 사용자에게 의해 설정될 수 있다. 정지 시간은 타이머를 사용해서 구현되고, 애플리케이션은 타이머가 만료된 것이 검출(S904)된 후에 정지해제된다(S907).

[0241] 대안으로서, 특정 이벤트의 실행이 모니터링될 수 있고, 특정 이벤트의 종료 검출(S905)된 후에 애플리케이션이 정지해제된다. 본 명세서에서 특정 이벤트의 종료를 검출은 활동 매니저 서비스의 도팅 보고 기능의 사용에 의한 보고를 통해 구현될 수도 있다. 단계(S904) 및 단계(S905)는 두 가지의 일반적인 정지해제 조건으로서 이해될 수 있고, 애플리케이션은 두 가지 조건 중 어느 하나 또는 둘 모두가 충족될 때 정지해제될 수 있다.

[0242] 정지해제가 수행될 경우, 모든 애플리케이션이 또는 일부 애플리케이션만이 정지해제될 수 있다. 정지해제될 필요가 있는 애플리케이션이 선택될 경우, 애플리케이션 중요도 순위에 기초하여 중요도가 상대적으로 높은 일부 애플리케이션이 선택되어 정지해제될 수 있다. 다른 애플리케이션은 해당 애플리케이션의 중요도가 매우 낮고 장기간 동안 사용자에게 의해 사용되지 않을 수 있기 때문에 계속해서 정지될 수 있다.

[0243] 특정 이벤트가 완료되기 전에, 그리고 지정된 타이밍 지속기간이 도달하지 않았을 경우에, 정지된 애플리케이션의 실행 환경의 변경을 나타내는 이벤트가 발생(S906)하면, 예를 들어, 정지된 애플리케이션이 포그라운드로 전환되고, 정지된 애플리케이션이 종료되고, 정지된 애플리케이션이 바인더 비동기 메시지를 수신하고, 정지된 애플리케이션의 통지 메시지가 사용자에게 의해 탭핑되면, 정지된 애플리케이션이 스케줄에 앞서 정지해제될 수 있다(S907). 본 명세서에서, "정지된 애플리케이션의 실행 환경의 변경을 나타내는 이벤트(the event indicating a change in a running environment of a frozen application)"는 일반적으로 정지된 애플리케이션과 관련된다. 이벤트의 발생은 일반적으로 정지된 애플리케이션의 중요도를 급작스럽게 높이고, 이윽고 정지된 애플리케이션이 정지해제되어야 한다. 유사하게, 정지된 애플리케이션의 실행 환경의 변경을 나타내는 이벤트의 모니터링이 또한 활동 매니저 서비스의 도팅 보고 기능에 의해 구현될 수 있다. 정지된 애플리케이션의 실행 환경의 변경을 나타내는 이벤트는 이하의 이벤트들 중 어느 하나일 수 있다: (1) 애플리케이션이 포그라운드로 전환됨; (2) 애플리케이션이 종료됨; (3) 애플리케이션이 재-인스톨되거나 애플리케이션의 버전이 업데이트됨; (4) 동적 월페이퍼 애플리케이션이 동적 월페이퍼로 설정됨; (5) 애플리케이션 위젯이 데스크톱에 추가됨; (6) 네트워크 데이터 패킷이 인스턴트 메시징(instance message, IM) 애플리케이션, 단문 메시지 서비스(short message service, SMS) 애플리케이션, 또는 이메일(Email) 애플리케이션으로 도달함; (7) 네트워크 연결해제 상태에서 IM/SMS/Email 애플리케이션이 네트워크에 연결됨; (8) 다른 애플리케이션이 정지된 애플리케이션의 공급자(provider) 또는 서비스(service)에 액세스함; (9) 시스템 또는 다른 애플리케이션이 바인더를 사용해서 정지된 프로세스를 동시에 호출함; (10) 데스크톱 상에 위젯을 갖는 애플리케이션이 잠금해제 후에 검출됨; (11) GPS 및 센서를 사용하는 애플리케이션이 이동 개시 전에 정지됨; (12) 정지된 애플리케이션이 헤드셋 키를 처리함; (13) 일시적으로 정지된 애플리케이션이 바인더 비동기 메시지를 수신함; (14) 정지된 애플리케이션에 액

세스하도록 알림 표시줄이 탭핑됨; (15) 시스템이 스크린-오프 상태에 진입함 등.

- [0244] 이벤트 (1) 내지 (15) 중 일부는 특정 애플리케이션과 관련된다는 점에 유의해야 하고, 이는 해당 애플리케이션이 정지해제될 필요가 있음을 의미한다(해당 애플리케이션이 정지된 상태일 경우). 그러나, 일부 이벤트는 특정 애플리케이션과 관련이 없으며, 이 경우, 정지된 애플리케이션들 중 일부 또는 전부가 정지해제되도록 선택된다. 예를 들어, (15)에서, 시스템이 스크린-오프 상태에 진입하는 것이 검출된다. 이는 사용자에게 보이는 포그라운드 애플리케이션이 보호될 필요가 없음을 의미한다. 따라서, 모든 정지된 애플리케이션이 정지해제될 수 있으므로, 애플리케이션들은 가능한 한 빨리 계속해서 실행될 수 있다.
- [0245] 두 가지의 일반적인 정지해제 방식에 더하여, 이 실시형태는 긴급 정지해제 방식 또는 스케줄에 앞서 정지해제를 수행하는 방식을 더 제안한다는 것을 알 수 있다. 특정 이벤트가 완료되기 전에, 그리고 지정된 타이머 지속기간에 도달하지 않았지만, 정지된 애플리케이션의 실행 환경의 변경을 나타내는 이벤트가 발생했을 경우, 정지된 애플리케이션은, 정지로 인한 애플리케이션의 정상 사용에의 영향을 피하기 위해, 스케줄에 앞서 정지해제될 수 있다.
- [0246] 사용자의 지속적인 동작(예를 들어, 사용자가 지속적으로 애플리케이션을 슬라이드 또는 기동함)은 일시적인 정지를 지속적으로 트리거한다. 일부 백그라운드 애플리케이션이 지속적인 순간 정지 동작 하에서 실행될 기회를 갖게 하기 위해, 이 실시형태는 주기적인 정지 방법을 더 제공한다. 애플리케이션에서 정지 동작이 수행되기 전에, 애플리케이션이 정지해제된 후의 애플리케이션의 실행 시간이 먼저 검출된다. 일시적인 정지는 애플리케이션이 정지해제된 후 누적된 실행 지속기간이 t_1 초(t_1 은 미리 설정된 지속기간, 예를 들어, 10s임)에 도달한 애플리케이션에서 지속적으로 수행될 수 있다. 그러나, 일시적인 정지는 애플리케이션이 정지해제된 후 누적된 실행 지속기간이 t_1 보다 작은 애플리케이션에서 주기적으로 수행될 수 있다. 애플리케이션은 먼저 t_2 초 동안 정지되고 나서, 지속적인 정지 조건이 충족될 때까지 t_3 초 동안 정지해제된다. 본 명세서에서, t_2 및 t_3 은 미리 설정된 지속기간 값($t_2 < t_1$, 및 $t_3 < t_1$), 예를 들어, $t_2 = 1.5s$, 및 $t_3 = 3s$ 이다. 이 주기적인 정지해제 방법에서는, 일부 백그라운드 애플리케이션이 장기간 동안 백그라운드 및 포그라운드에서 교대로 동작할 때 어느 정도 실행될 기회를 얻을 수 있게 하는 것이 잘 보장될 수 있으므로, 장기간에 걸쳐 일부 중요한 백그라운드 애플리케이션이 실행될 수 없는 경우가 회피되고, 심지어 일부 애플리케이션이 지속적으로 정지되기 때문에 발생하는 예외(예를 들어, 장기간 동안 다운로드가 정지된 후에, 다운로드가 계속해서 수행될 수 없음)가 회피된다. 애플리케이션의 정지해제 여부와 관계없이, 정지된 애플리케이션의 실행 환경의 변경을 나타내는 이벤트가 검출된 후에, 애플리케이션이 정지해제된 후 누적된 실행 지속기간이 0으로 재설정된다. 이는 실행 환경의 변경을 나타내는 이벤트가 수신된 후 일정 기간 동안 모든 백그라운드 애플리케이션이 실행될 수 있는 것을 보장할 수 있으므로, 포그라운드 애플리케이션의 높은 순간적인 리소스 요건을 우선적으로 보장하면서, 백그라운드 애플리케이션의 실행 및 경험이 보장된다.
- [0247] 본원의 다른 실시형태는 중요한 태스크를 스케줄링하기 위한 방법을 제공한다. 본 명세서에서의 태스크는 프로세스 또는 스레드일 수 있다.
- [0248] 이상적인 상태에 있어서는, 각각의 태스크가 CPU로부터 동일한 시간 슬라이스를 취득할 수 있고, 모든 태스크가 CPU에서 동시에 실행된다. 그러나, 실제로는, 하나의 CPU에서 동시에 실행될 수 있는 태스크는 하나 뿐이다. 다시 말해, 하나의 태스크가 CPU를 점유하면, 다른 태스크는 대기해야 한다. 완전 공정 스케줄러(completely fair scheduler, CFS) 스케줄링 알고리즘은 기존의 리눅스 시스템에서 비교적 일반적인 스케줄링 알고리즘이다. 공정성을 달성하기 위해, CFS 스케줄링 알고리즘은 대기중인 프로세스가 다음으로 스케줄링되도록 현재 실행되는 태스크를 처분할 필요가 있다. 구체적인 구현예에 있어서, CFS는, 각각의 태스크의 가상 실행 시간(virtual run time, vruntime)을 사용해서, 가장 스케줄링할 가치가 있는 태스크를 측정한다. CFS에서의 준비 상태 큐는 vruntime을 키 값으로 사용하는 레드-블랙 트리(red-black tree)이다. 프로세스는 vruntime이 작을수록 전체 레드-블랙 트리의 가장 좌측 끝에 가까워진다. 따라서, 스케줄러는, 매번, 레드-블랙 트리의 가장 좌측 끝에 위치한 태스크를 선택하고, 태스크의 vruntime은 가장 작다.
- [0249] 이 실시형태에서 언급되는 "CPU"는 컴퓨터 장치에서 최소 처리 유닛이며, 처리 코어 또는 간략히 코어라고 할 수도 있다는 점에 유의해야 한다.
- [0250] 도 10에 도시된 바와 같이, 이진 트리 및 이진 트리 상의 태스크 T1 내지 태스크 T6의 경우, CPU가 태스크를 실행하는 시퀀스는 T1-T3-T5-T2-T4-T6이다. 태스크 T1은 중요한 태스크라고 가정한다.
- [0251] vruntime은 태스크의 실제 실행 시간 및 태스크의 가중치(weight)를 사용해서 계산되고, 누적되는 값이다. CFS

스케줄러에서는, 태스크 우선순위의 개념이 약화되지만, 태스크의 가중치가 강조된다. 태스크의 가중치가 크면, 태스크가 더욱 실행될 필요가 있음을 나타낸다. 따라서, 태스크의 가상 실행 시간이 더 작고, 태스크를 스케줄링할 더 많은 기회가 존재한다.

[0252] CFS 스케줄링 알고리즘의 공정성 때문에, 중요한 태스크의 가중치가 매우 큰 값으로 설정되더라도, 중요한 태스크가 한 번 실행된 후에는, 중요한 태스크도 역시 적어도 하나의 다른 태스크가 실행될 때까지 대기할 필요가 있다. 태스크 T1이 한 번 실행된 후에는, 태스크 T1의 vruntime 값이 더 커진다. CFS가 태스크 T3을 스케줄링한 후에, 태스크 T1은 vruntime 값이 커지기 때문에 도 11에 도시된 바와 같이 큐의 끝에 삽입된다. 이 경우, 태스크 T1은 태스크 T3-T5-T2-T4-T6이 각각 태스크 T1이 다시 실행되기 전에 한 번 실행될 때까지 대기할 필요가 있다. 도 11은 예시이다. 태스크 T1이 한 번 실행된 후에, 태스크 T1은 다른 위치에서 다시 큐에 더해질 수 있다. 그러나, 태스크 T1의 위치에 관계없이, 공정성을 구현하기 위해, CFS는 항상 태스크 T1을 대기시킨다. 태스크 T1은 사용자에게 중요한 태스크이기 때문에, 이러한 대기는 시스템 중지를 야기할 수 있다.

[0253] 이 실시형태는 중요한 태스크를 스케줄링하기 위한 방법을 제공한다. 전술한 준비 상태 큐에 더하여, 각각의 CPU에 대하여 새로운 실행 큐(아래에서는 VIP 큐라고 함)가 생성되고, 새로운 실행 큐의 구조는 전술한 준비 상태 큐의 구조와 유사하다. 중요한 태스크는 VIP 큐에 배치된다.

[0254] 중요한 태스크는 사용자 경험에 미치는 영향이 비교적 큰 태스크이고, 전술한 실시형태에서 언급된 중요도 순위가 상위에 있는 N개의 애플리케이션의 모든 스레드(또는 프로세스)를 포함할 수 있거나; 또는 중요한 태스크는 N개의 애플리케이션의 스레드들 중 UI 스레드 및 렌더링 스레드와 같은 키 스레드를 포함하거나; 또는 중요한 태스크는 UI 스레드 및 렌더링 스레드와 같은 현재 포그라운드 애플리케이션의 키 스레드를 포함한다.

[0255] 중요한 태스크들은 두 가지 타입, 즉 정적인 중요한 태스크 및 동적인 중요한 태스크로 분류된다. 정적인 중요한 태스크는 일반적으로 사용자 모드에서 식별되며, 예를 들어, 포그라운드 애플리케이션의 사용자 인터페이스(user interface, UI) 스레드 및 렌더링(render) 스레드이다. 정적인 중요한 태스크의 중요도는 일반적으로 애플리케이션 중요도가 변경될 때에만 취소된다. 동적인 중요한 태스크는 정적인 중요한 태스크가 의존하는 중요한 태스크이고, 일반적으로 커널 모드에서 식별된다. 종속성이 해제되면, 동적인 중요한 태스크의 중요도가 취소된다. 동적인 중요한 태스크는 정적인 중요한 태스크가 직접적으로 의존하는 태스크를 포함하고, 정적인 중요한 태스크가 간접적으로 의존하는 태스크를 또한 포함할 수 있다.

[0256] 본 명세서에서 종속성은 데이터 종속성, 잠금 종속성, 바인더 서비스 종속성, 제어 흐름 종속성 등일 수 있다. 데이터 종속성은 태스크 B의 실행이 태스크 A의 출력에 의존할 필요가 있음을 의미한다. 잠금 종속성은 태스크 B의 실행이 태스크 A에 의해 해제되는 잠금을 필요로 함을 의미한다. 제어 흐름 종속성은 실행 논리의 관점에서, 태스크 A가 실행된 후에만 태스크 B가 실행될 수 있음을 의미한다. 바인더 서비스 종속성은 제어 흐름 종속성의 특정 인스턴스이고, 태스크 A가 바인더 기능을 호출(원격 절차 호출과 유사함)하여 태스크 B에게 기능을 완료하고 실행 결과를 반환하도록 요구해서, 태스크 A가 태스크 B에 대한 바인더 서비스 종속성을 생성함을 의미한다.

[0257] 리눅스 시스템은 종속성 검출 관계의 특정한 구현 프로세스를 설명하기 위한 예로서 사용된다. 태스크 제어 블록 task_struct에 두 개의 필드 static_vip 및 dynamic_vip가 추가되고, 이들 필드는 제각기 정적인 중요한 태스크 및 동적인 중요한 태스크의 플래그 값들을 나타내는 데 사용된다. static_vip=1이면, 태스크가 정적인 중요한 태스크임을 나타내거나; 또는 dynamic_vip가 0이 아니면, 태스크가 동적인 중요한 태스크임을 나타낸다. 동적인 중요한 태스크의 경우에는, 복수의 다른 태스크들이 동시에 하나의 태스크에 의존할 수 있고, 해당 종속성의 이유는 동일하거나 또는 상이할 수 있으며, 예를 들어, 뮤텁스(mutex) 잠금 종속성, rwsem 관독/기입 세마포어(semaphore) 종속성, 및 바인더 서비스 종속성일 수 있다. 따라서, dynamic_vip 필드는 제각기 뮤텁스 잠금 종속성, rwsem 종속성, 및 바인더 종속성을 나타내는 세 가지 타입으로 분류되거나; 또는 그 이상의 타입으로도 분류될 수 있지만, 여기서 그 이상의 타입을 자세하게 설명하지는 않는다. 몇 가지 타입이 장래의 확장을 위해 예비될 수도 있다. 각각의 타입은 8비트를 사용해서 저장된다. 이렇게 해서, 상응하는 종속성 함수가 호출될 때마다, 필드에 대응하는 블록 값이 1씩 증가되고, 종속성 함수가 완료될 때마다, 모든 필드가 0이 될 때까지, 그에 상응하여 값이 1씩 감소된다. 이어서 태스크의 중요도 속성이 취소되고, 태스크가 실행을 위한 준비 상태 큐에 재-배치된다.

[0258] 뮤텁스 잠금 종속성 및 바인더 종속성은 아래에서 뮤텁스 종속성 및 바인더 종속성의 단계들을 상세하게 설명하기 위한 예로서 사용된다.

- [0259] mutex_lock 기능은 뮤텍스 잠금을 취득하기 위해 중요한 태스크 A에서 호출되고, 해당 기능에서 잠금 상태가 검출된다. 잠금의 취득에 실패하면, 다른 태스크에 의해 잠금이 취득되었음을 나타낸다. 현재 태스크 A는 일시정지되고 슬립 상태에 진입한다. 잠금의 현재 소유자(태스크 B)를 취득하기 위해 코드가 추가되고, 태스크의 구조의 dynamic_vip 필드의 값에 대응하는 비트가 1씩 증가되고, 이어서, 태스크 B가 스케줄링 및 실행을 위해 VIP 큐로 이동된다. 태스크 B가 잠금을 해제(mutex_unlock)할 경우, 태스크 B의 task_struct 내의 dynamic_vip의 상응하는 값이 결정되고 1씩 감소된다. 상응하는 값이 0이면, 해당 태스크가 VIP 큐에서 제거된다. 다른 잠금의 실행 프로세스는 유사하다.
- [0260] 중요한 태스크 A가 보통의 태스크 B의 바인더 기능을 호출할 경우, 커널의 바인더 드라이버에서의 binder_thread_write 기능에서 태스크 B의 태스크 구조(task_struct)가 먼저 확인되고, 태스크 구조의 dynamic_vip의 값이 설정된다. 이어서, 상응하는 기능 ID 및 파라미터가 태스크 B에 전달되고, 태스크 B가 실행을 위해 기동된다. 이어서, 중요한 태스크 A는 태스크 B가 실행 결과를 반환할 때까지 대기한다. 태스크 B가 binder_thread_read 기능을 사용해서 상응하는 기능을 완료한 후에, 실행 결과를 태스크 A에 반환하기 전에, 태스크 B는 dynamic_vip의 값을 검사한다. dynamic_vip의 값이 0이면, 태스크가 VIP 큐에서 제거된다.
- [0261] 복수의 중요한 태스크가 전술한 방식으로 식별된 후에, 이들 태스크가 VIP 큐에 삽입된다. 삽입 원리는 전술한 준비 상태 큐에 대한 원리와 동일할 수 있으며, 즉, 각각의 태스크는 해당 태스크의 vruntime의 값에 기초하여 삽입된다. 도 12에 도시된 바와 같이, VIP 큐는 두 개의 중요한 태스크 T1 및 T7을 포함하고, T7의 vruntime은 현재 T1의 vruntime보다 작다.
- [0262] 실행될 필요가 있는 다음 태스크를 CPU가 취득할 때마다, CPU는 먼저 VIP 큐에서의 태스크가 실행될 필요가 있는지의 여부를 검사한다. VIP 큐에서의 태스크가 실행될 필요가 있으면, CPU는 해당 큐로부터 태스크를 선택하여 실행한다. 해당 큐가 비어 있으면, CPU는 준비 상태 큐로부터 태스크를 선택하여 실행한다. 이를 통해, 모든 중요한 태스크가 다른 중요하지 않은 태스크보다 먼저 실행될 수 있고, 중요한 태스크에 대하여 큐 삽입 기능과 유사한 기능이 구현된다.
- [0263] 또한, VIP 큐에서의 태스크의 수량이 비교적 클 경우, VIP 큐에서의 태스크가 큐에서 대기하는 것을 방지하기 위해, 각각의 CPU의 VIP 큐에서의 태스크가 현재 지연되는지의 여부가 적절한 시간에 검사될 수 있다. 지연된 태스크가 있으면, 태스크가 이동 가능한지의 여부가 결정된다. 태스크가 이동 가능하면, 태스크는 다른 CPU의 유휴 VIP 큐로 이동된다. 이렇게 해서, 중요한 태스크의 마이그레이션(또는 이동이라고도 함)이 구현되고, 이로써 중요한 태스크의 적시 실행이 보장되고, 중지가 회피되고, 사용자 경험이 더 향상된다.
- [0264] 구체적으로, 도 13을 참조하면, 클록 인터럽트에 도달할 경우(S1301), 커널은 대기 기간이 임계치(예를 들어, 10 ms)를 초과한 태스크가 현재의 CPU에 대응하는 VIP 큐에 존재하는지의 여부를 검사하고, 즉, 지연된 태스크가 있는지의 여부를 결정한다(S1302). 대기 기간이 임계치를 초과한 태스크가 존재하면, 커널은 태스크의 데이터 및/또는 명령어가 여전히 캐시(cache)에 존재하는지의 여부를 더 검사하고, 즉, 태스크가 이동 가능한지의 여부를 결정한다(S1303). 태스크의 데이터 및/또는 명령어의 전부 또는 일부가 캐시에 존재하지 않으면(캐시가 핫 캐시인지의 여부), 커널은 현재 CPU가 위치한 CPU 클러스터(cluster)로부터 대상 CPU를 선택— 대상 CPU의 VIP 큐에서 태스크가 대기하고 있지 않고 대상 CPU의 VIP 큐에 실시간 태스크가 없음 —하고(S1304); 태스크를 대상 CPU에 마이그레이션한다(S1305). 태스크는 커널에 의해 제공되는 마이그레이션 기능을 호출함으로써 마이그레이션될 수 있다. 캐시가 핫 캐시인지의 여부는 커널에 의해 제공되는 task_hot 기능을 호출함으로써 검출될 수 있다.
- [0265] 특정 구현예에 있어서, 마이그레이션될 수 있는 모든 태스크가 한 번에 식별될 수 있고, 이어서 마이그레이션이 수행된다. 대안으로서, VIP 큐에서의 태스크들은 순차적으로 처리될 수 있다.
- [0266] 태스크가 대기 기간 동안 실행된 적이 없다면, 태스크의 대기 기간은 태스크의 현재 시간과 태스크의 대기 시간 사이의 시간차이다. 대안으로서, 태스크의 대기 기간은 태스크의 현재 시간과 마지막 실행 시간 사이의 시간차이다.
- [0267] 일반적으로 큐에서의 태스크의 대기 시간이 임계치를 초과하는지의 여부를 검출하기 위해 큐의 잠금이 먼저 취득될 필요가 있다. 클록 인터럽트에 도달하기 전에 전술한 방법이 구현되므로, 교착 상태(deadlock)가 어느 정도 회피될 수 있다.
- [0268] 태스크가 이동될 경우, 태스크는 원래의 CPU보다 처리 효율이 높은 CPU로 이동될 수 있다. 이렇게 해서, 중요한 태스크의 처리 효율이 더욱 향상될 수 있다. 예를 들어, 현재의 단말 장치의 8개의 코어(CPU 0 내지 CPU

7)는 일반적으로 두 가지 레벨, 즉 4개의 소형 코어 및 4개의 대형 코어로 분류된다. 마이그레이션 동안 원래의 CPU가 소형 코어이면, 대형 코어가 대상 CPU로서 선택될 수 있고, 중요한 태스크는 도 14에 도시된 바와 같이 대상 CPU로 마이그레이션된다.

[0269] VIP 큐는 각각의 CPU에 대하여 생성되고, 중요한 태스크가 해당 큐에 배치되므로, 중요한 태스크는 준비 상태 큐에서 다른 태스크보다 먼저 실행되고, 이로써 중요한 태스크의 실행 효율이 보장된다. 중요한 태스크는 시스템의 중지 경험과 관련되기 때문에, 사용자에게 의해 감지될 수 있는 시스템 중지가 어느 정도 회피될 수 있고, 이로써 단말 장치의 사용자 경험이 향상된다.

[0270] 전술한 실시형태들에서 제안되는 모듈들 또는 모듈 구분은 단지 예시로서 도시되며, 모듈들의 설명된 기능들은 단지 설명을 위한 예시일 뿐이라는 점에 유의해야 한다. 본원은 이에 제한되지 않는다. 프로그래머는 2개 이상의 모듈의 기능들을 요건에 따라 결합하거나, 또는 하나의 모듈의 기능을 분할하여 보다 세분화된 모듈을 취득할 수 있거나, 또는 다른 변환 방식을 사용할 수 있다.

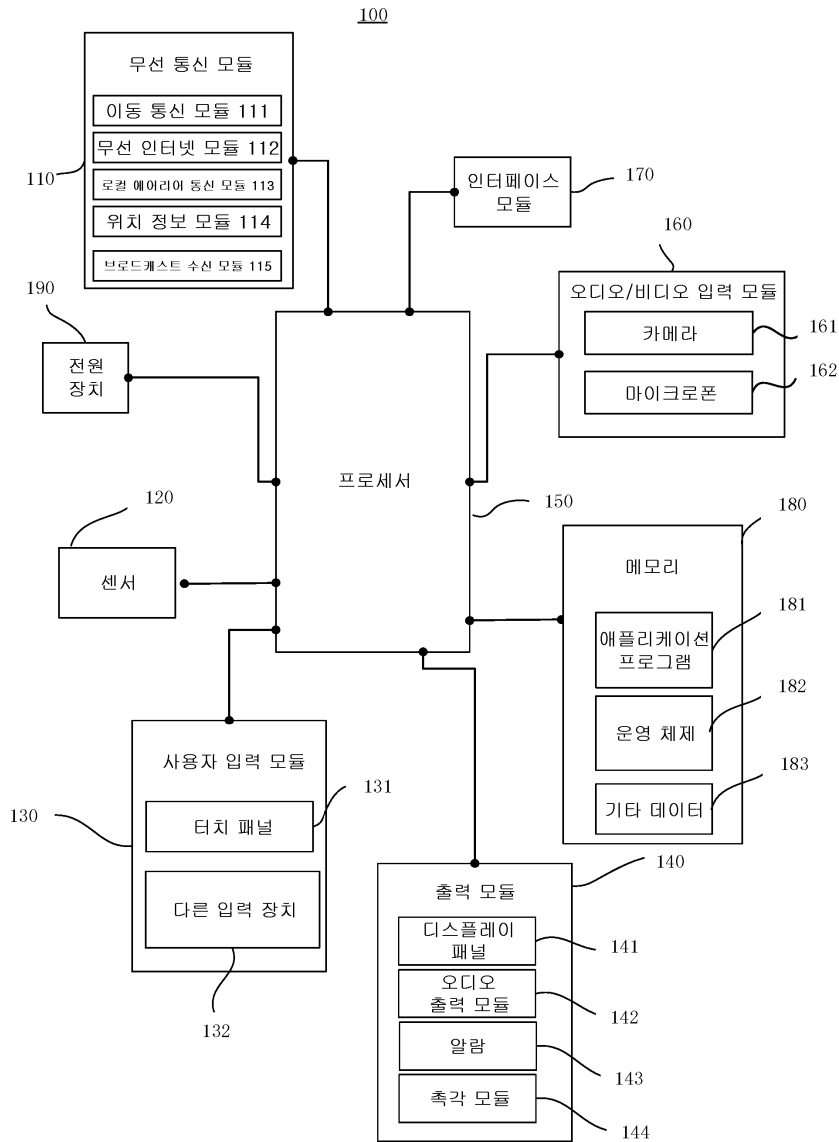
[0271] 전술한 실시형태들에서 동일한 또는 유사한 부분들에 대해서는, 이들 실시형태를 참조할 수 있다.

[0272] 설명된 장치 실시형태들은 단지 예시일 뿐이다. 별도의 부분들로서 설명되는 모듈들은 물리적으로 분리될 수도, 아닐 수도 있으며, 모듈들로서 도시된 부분들은 물리적인 모듈일 수도, 아닐 수도 있고, 한 자리에 위치될 수 있거나, 또는 복수의 네트워크 모듈 상에 분산될 수도 있다. 모듈들의 일부 또는 전부는 실시형태들의 해법의 목적을 달성하기 위해 실제 요구에 따라 선택될 수 있다. 또한, 본원에서 제공되는 장치 실시형태들의 첨부 도면에서, 모듈들 사이의 연결 관계는 모듈들이 서로 통신 연결을 갖는 것을 나타내고, 이는 하나 이상의 통신 버스 또는 신호 케이블로서 구체적으로 구현될 수 있다. 당업자라면, 창조적인 노력 없이 본원의 실시형태들을 이해 및 구현할 수 있을 것이다.

[0273] 전술한 설명은 단지 본원의 구체적인 일부 구현예들이지, 본원의 보호 범위를 제한하려는 것은 아니다.

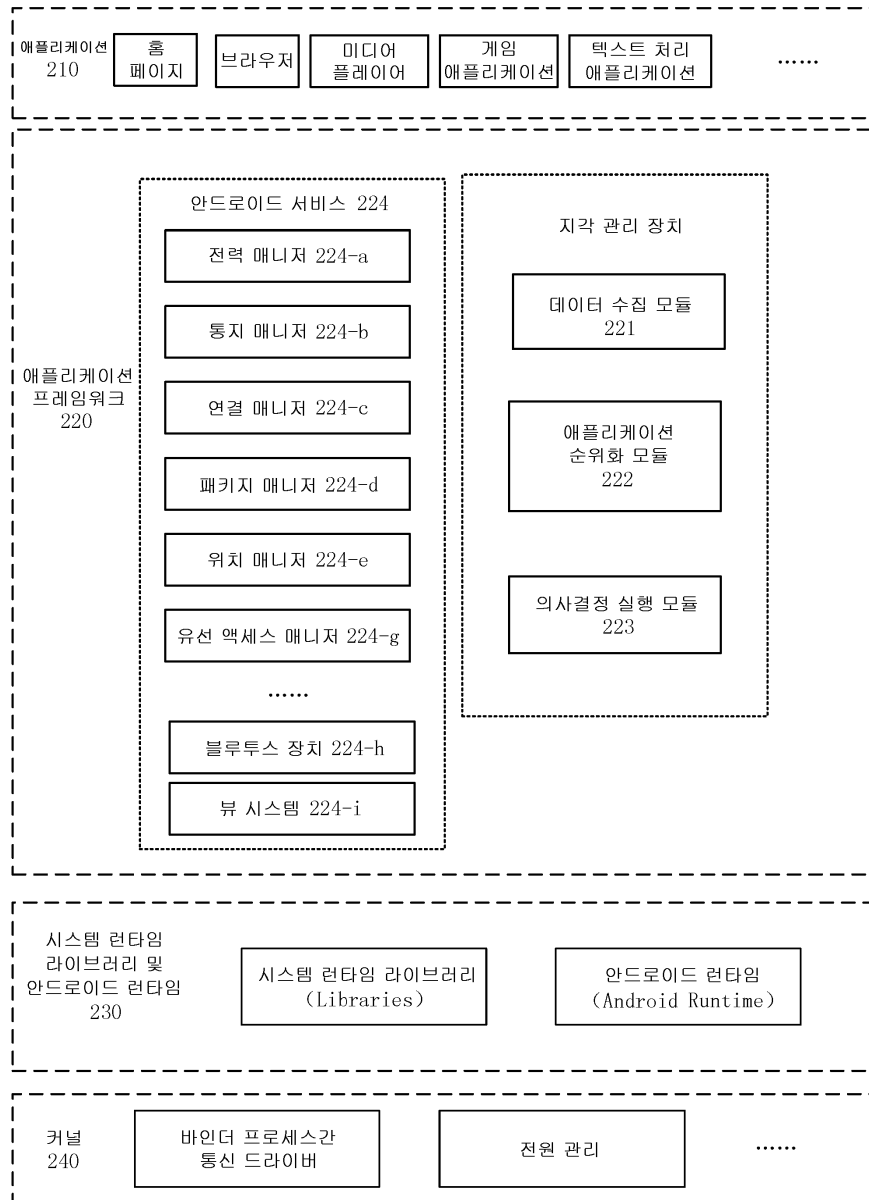
도면

도면1

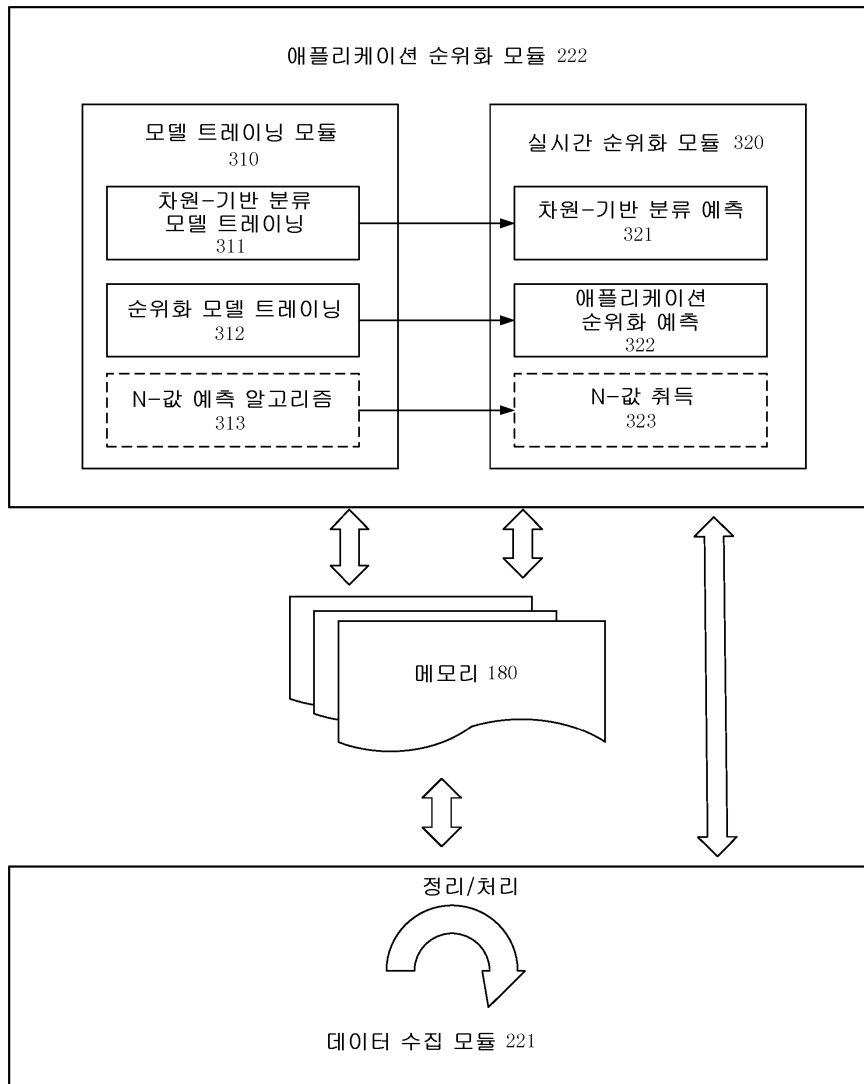


도면2

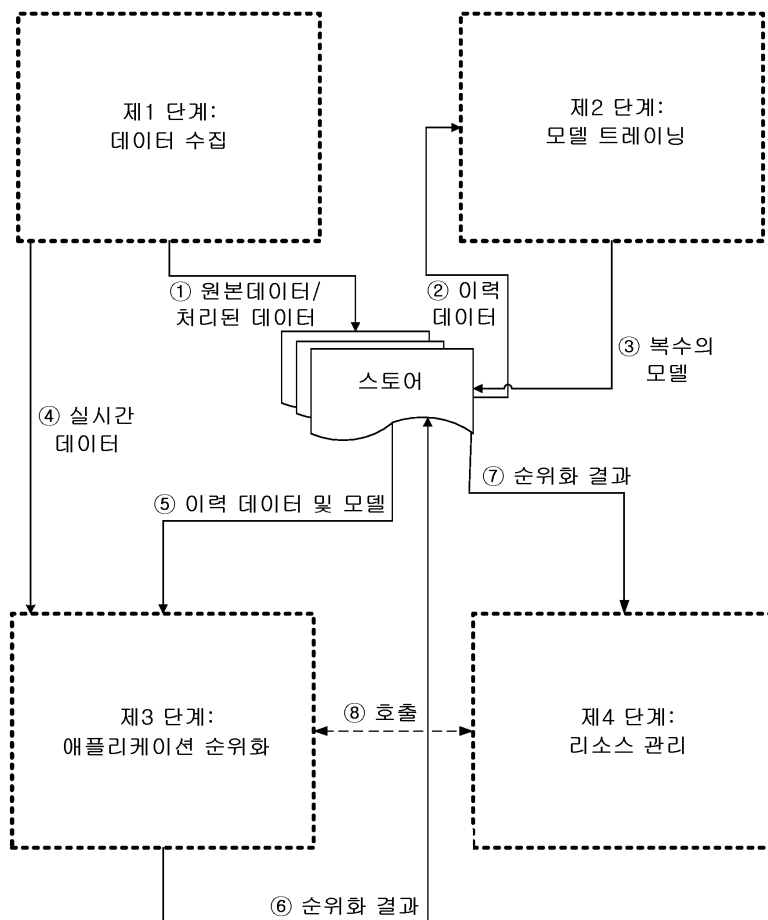
200



도면3

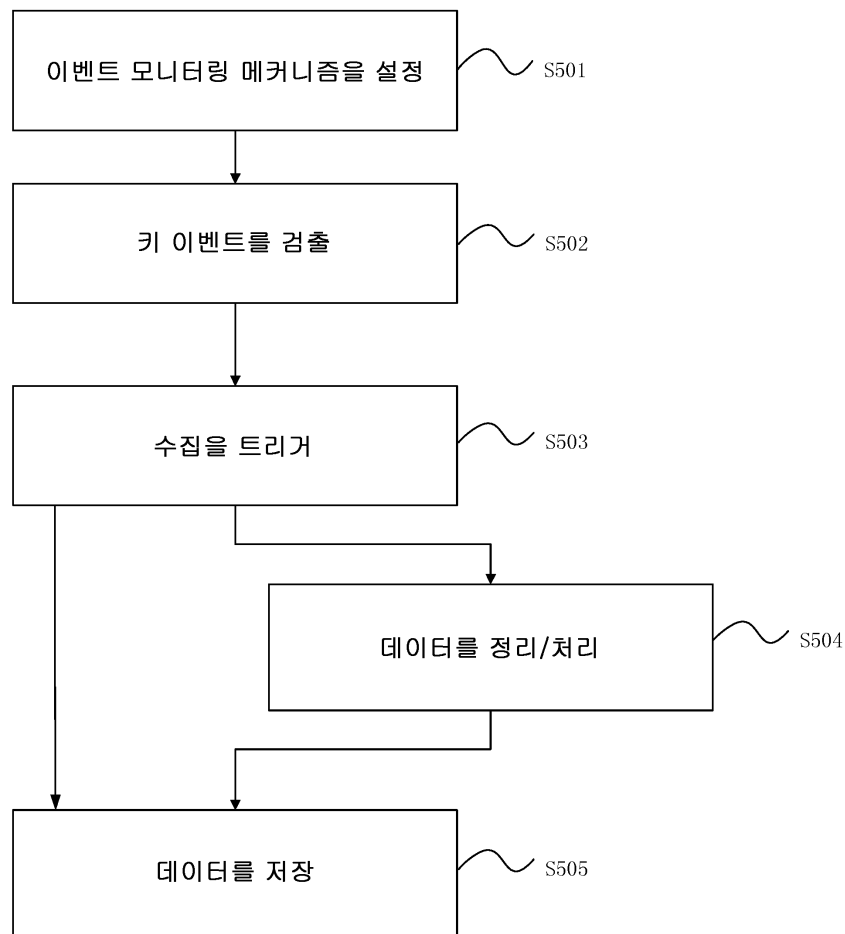


도면4

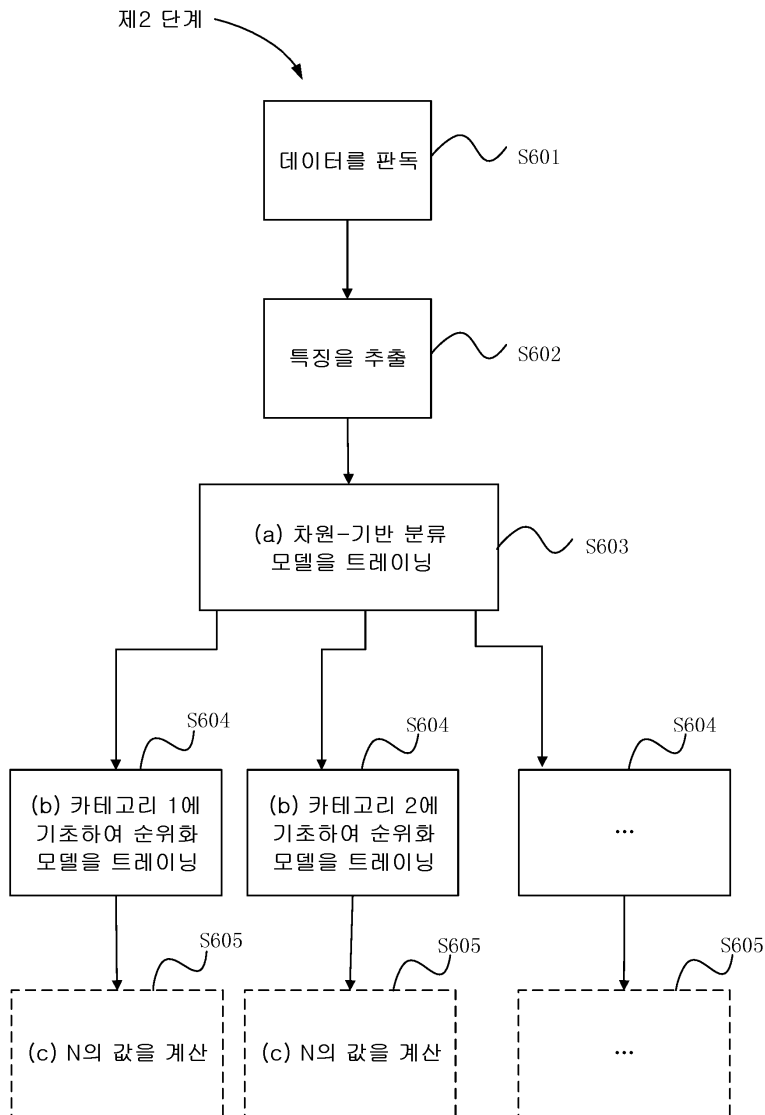


도면5

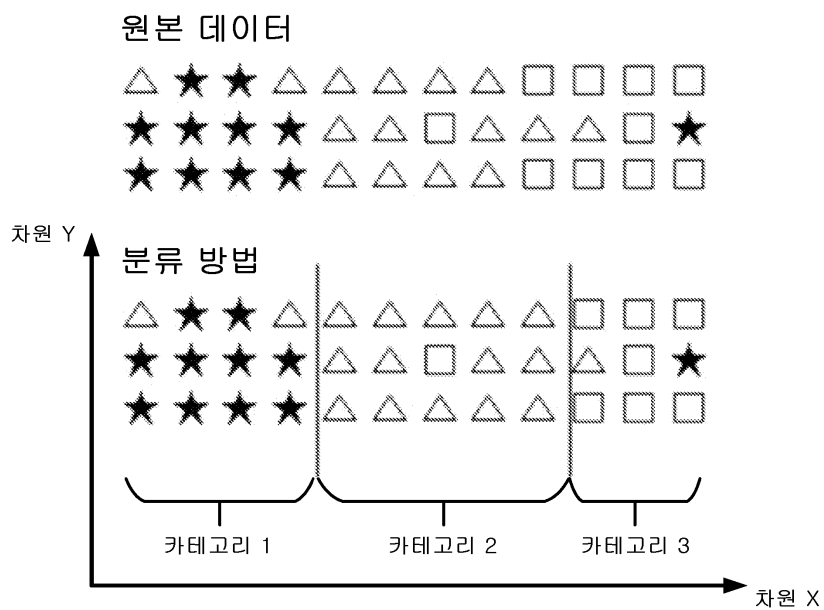
제1 단계



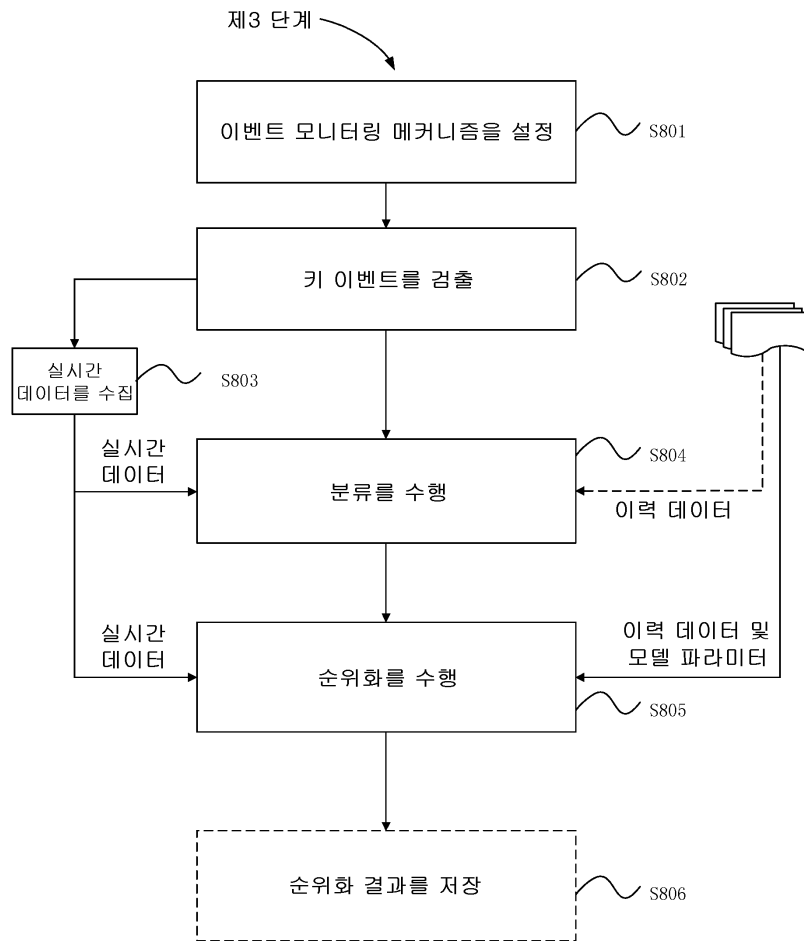
도면6



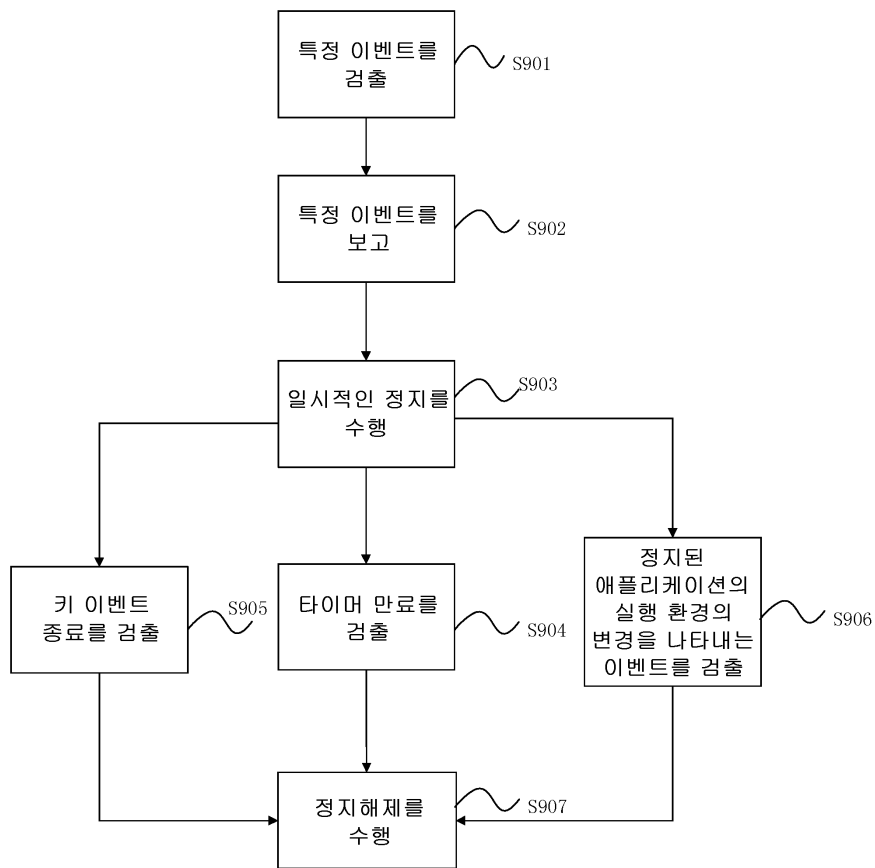
도면7



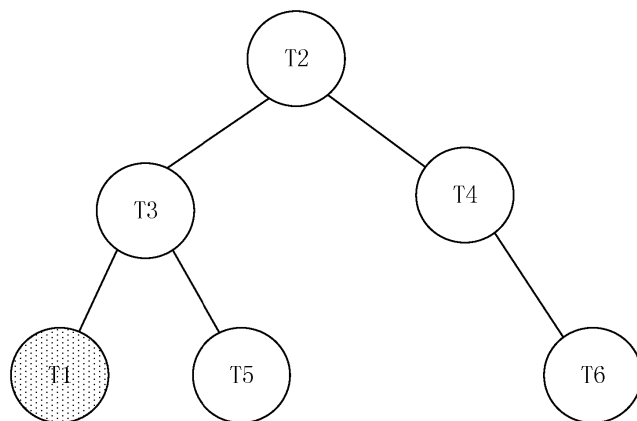
도면8



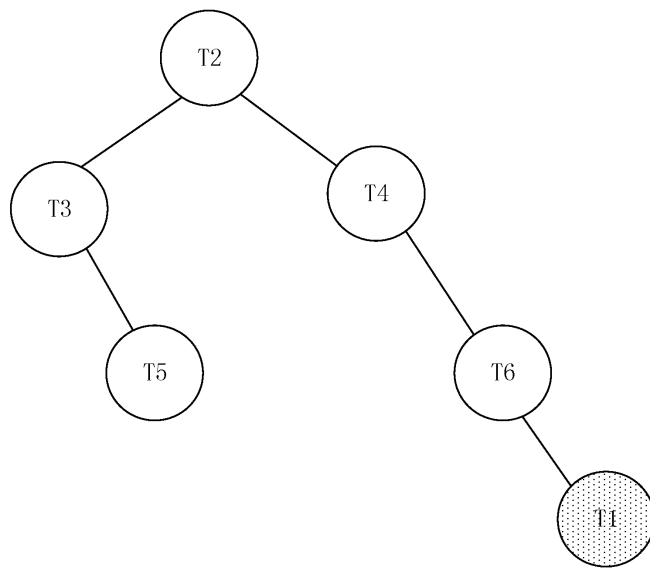
도면9



도면10

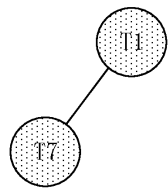


도면11

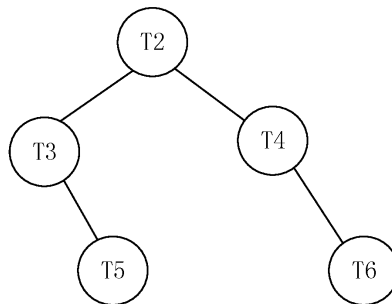


도면12

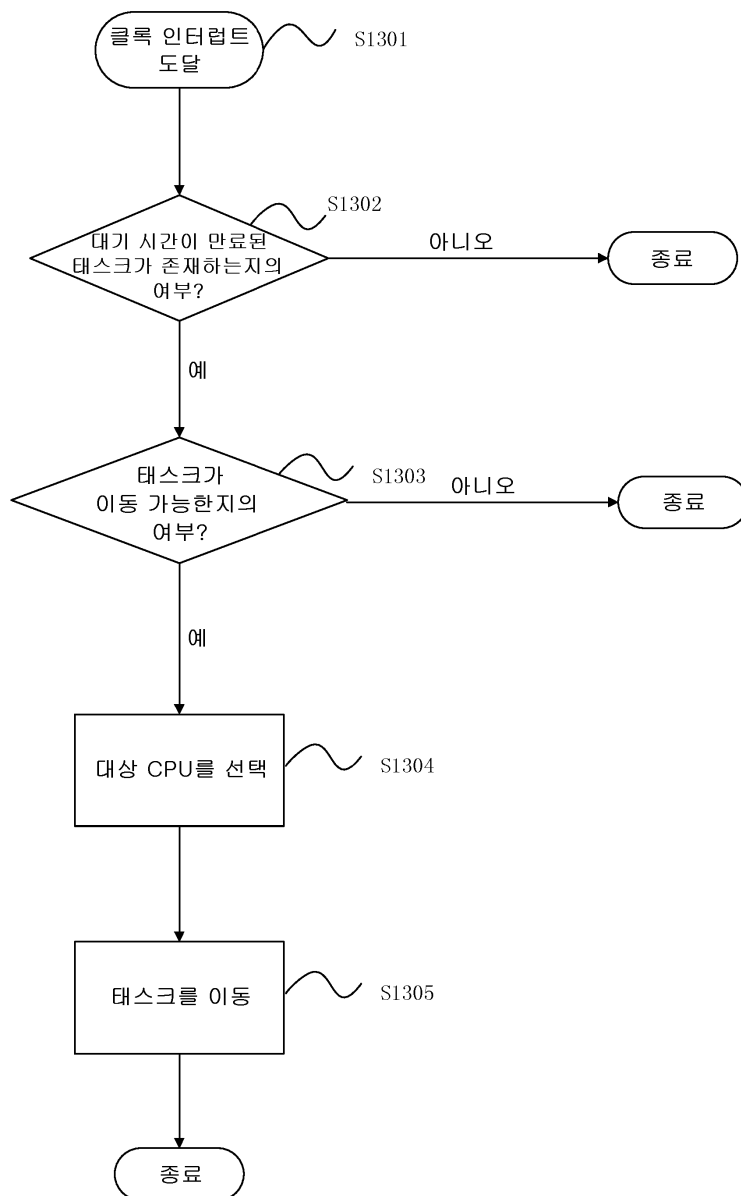
VIP 큐



준비 상태 큐



도면13



도면14

