



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0172938
(43) 공개일자 2024년12월10일

(51) 국제특허분류(Int. Cl.)
G06F 9/48 (2018.01) G06F 9/30 (2018.01)
G06F 9/50 (2018.01) H04L 67/12 (2022.01)
(52) CPC특허분류
G06F 9/4893 (2013.01)
G06F 9/30189 (2013.01)
(21) 출원번호 10-2023-0071621
(22) 출원일자 2023년06월02일
심사청구일자 2023년06월02일

(71) 출원인
이화여자대학교 산학협력단
서울특별시 서대문구 이화여대길 52 (대현동, 이화여자대학교)
(72) 발명자
반효경
서울특별시 관악구 관악로30길 12, 107동 2001호 (봉천동, 봉천우성아파트)
기수민
서울특별시 서초구 서운로 221, 103동 2504호 (서초동, 래미안서초스위트아파트)
(74) 대리인
특허법인 무한

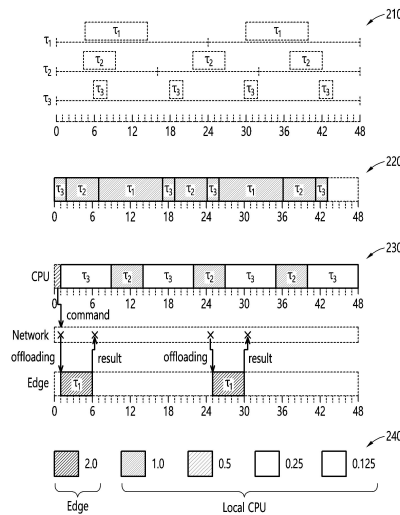
전체 청구항 수 : 총 17 항

(54) 발명의 명칭 **태스크 오프로딩, CPU 전압조절, 메모리 배치를 통합한 태스크 집합의 스케줄링 최적화 방법 및 장치**

(57) 요약

태스크 집합의 스케줄링 최적화 방법 및 장치가 개시된다. 태스크 집합의 스케줄링 최적화 방법은 태스크 집합에 포함된 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 기초로 초기 해들을 생성하는 단계; 상기 초기 해들로 구성된 해집합에서 복수의 해들을 부모해들로 선택하는 단계; 상기 부모해들을 기초로 자식해들을 생성하는 단계; 상기 자식해들을 이용하여 해집합을 업데이트하는 단계; 및 업데이트된 해집합에 포함된 해들이 수렴하는 경우, 수렴한 해에 기초하여 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 스케줄링 하는 단계를 포함할 수 있다.

대표도 - 도2



(52) CPC특허분류

G06F 9/4837 (2013.01)

G06F 9/4862 (2013.01)

G06F 9/5016 (2013.01)

G06F 9/5077 (2013.01)

H04L 67/12 (2022.05)

(72) 발명자

변규리

서울특별시 은평구 수색로 217, 102동 603호 (수색동, 디엠씨 자이1단지)

조경운

서울특별시 서초구 방배천로2안길 118, 3층 (방배동)

이 발명을 지원한 국가연구개발사업

과제고유번호 1711182079

과제번호 2019R1A2C1009275

부처명 과학기술정보통신부

과제관리(전문)기관명 한국연구재단

연구사업명 개인기초연구(과기정통부)

연구과제명 범용 OS의 혼합형 실시간 워크로드 지원을 위한 NVM 기반 end-to-end 시스템 관리 기술

기 여 율 1/3

과제수행기관명 이화여자대학교

연구기간 2023.03.01 ~ 2024.08.31

이 발명을 지원한 국가연구개발사업

과제고유번호 1711179344

과제번호 00155966

부처명 과학기술정보통신부

과제관리(전문)기관명 정보통신기획평가원

연구사업명 인공지능융합혁신인재양성(R&D)

연구과제명 인공지능융합혁신인재양성(이화여자대학교)

기 여 율 1/3

과제수행기관명 이화여자대학교산학협력단

연구기간 2023.01.01 ~ 2023.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호 1711152635

과제번호 2018-0-00549-005

부처명 과학기술정보통신부

과제관리(전문)기관명 정보통신기획평가원

연구사업명 SW컴퓨팅산업원천기술개발

연구과제명 매니코어 초대용량 메모리를 위한 확장형 순서보장 운영체제 개발

기 여 율 1/3

과제수행기관명 한국과학기술원

연구기간 2022.01.01 ~ 2022.12.31

명세서

청구범위

청구항 1

태스크 집합에 포함된 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 기초로 초기 해들을 생성하는 단계;

상기 초기 해들로 구성된 해집합에서 복수의 해들을 부모해들로 선택하는 단계;

상기 부모해들을 기초로 자식해들을 생성하는 단계;

상기 자식해들을 이용하여 해집합을 업데이트하는 단계; 및

업데이트된 해집합에 포함된 해들이 수렴하는 경우, 수렴한 해에 기초하여 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 스케줄링 하는 단계

를 포함하는 태스크 집합의 스케줄링 최적화 방법.

청구항 2

제1항에 있어서,

상기 초기 해들은,

태스크 별로 오프로딩 여부를 나타내는 문자열, CPU의 클럭 빈도를 나타내는 문자열, 및 메모리 위치를 나타내는 문자열로 구성되는 태스크 집합의 스케줄링 최적화 방법.

청구항 3

제2항에 있어서,

상기 자식해들을 생성하는 단계는,

두개의 부모해들 각각에 포함된 문자열들 각각의 일부를 교환하여 두개의 자식해들을 생성하는 태스크 집합의 스케줄링 최적화 방법.

청구항 4

제2항에 있어서,

상기 자식해들을 생성하는 단계는,

부모해들 각각에 포함된 문자열들의 값을 랜덤하게 변경하여 자식해들을 생성하는 태스크 집합의 스케줄링 최적화 방법.

청구항 5

제1항에 있어서,

스케줄링에 따른 전력 소모량을 기초로 상기 해집합에 포함된 해들 각각의 적합도를 평가하는 단계;

를 더 포함하고,

상기 해집합을 업데이트하는 단계는,

해집합의 크기 및 적합도에 따라 상기 해집합에 포함된 해들과 상기 자식해들을 선택하여 상기 해집합을 업데이트하는 태스크 집합의 스케줄링 최적화 방법.

청구항 6

제5항에 있어서,

상기 부모해들로 선택하는 단계는,

상기 초기 해들 각각의 적합도에 기초하여 기 설정된 숫자의 초기 해들을 상기 부모해들로 선택하는 태스크 집합의 스케줄링 최적화 방법.

청구항 7

제1항에 있어서,

상기 태스크들 각각의 스케줄 가능성을 테스트하는 단계; 및

상기 태스크들 중에서 스케줄 가능성 테스트를 통과하지 못하는 태스크가 발생하는 경우, 상기 테스트를 통과하지 못하는 태스크를 기초로 생성한 해의 적합도에 페널티를 부여하는 단계

를 더 포함하는 태스크 집합의 스케줄링 최적화 방법.

청구항 8

제7항에 있어서,

상기 스케줄 가능성을 테스트하는 단계는,

태스크가 엡지 서버에서 실행되는 경우의 최악 실행시간, 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간, 태스크들 각각의 업로드 시간 및 다운로드 시간을 기초로 임계 주기를 결정하고, 태스크의 실행 주기를 상기 임계 주기와 비교하여 엡지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트하는 태스크 집합의 스케줄링 최적화 방법.

청구항 9

제7항에 있어서,

상기 스케줄 가능성을 테스트하는 단계는,

태스크가 로컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행시간과 태스크의 실행 주기, DRAM과 NVM 간의 읽기/쓰기 연산 및 읽기/쓰기 지연 시간 및 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간에 기초한 이용률 테스트를 수행하여 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트하는 태스크 집합의 스케줄링 최적화 방법.

청구항 10

태스크 집합에 포함된 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 기초로 초기 해들을 생성하고,

상기 초기 해들로 구성된 해집합에서 복수의 해들을 부모해들로 선택하며,

상기 부모해들을 기초로 자식해들을 생성하고,

상기 자식해들을 이용하여 해집합을 업데이트하며,

업데이트된 해집합에 포함된 해들이 수렴하는 경우, 수렴한 해에 기초하여 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 스케줄링 하는 프로세서

를 포함하는 태스크 집합의 스케줄링 최적화 장치.

청구항 11

제10항에 있어서,

상기 초기 해들은,

태스크 별로 오프로딩 여부를 나타내는 문자열, CPU의 클럭 빈도를 나타내는 문자열, 및 메모리 위치를 나타내는 문자열로 구성되는 태스크 집합의 스케줄링 최적화 장치.

청구항 12

제11항에 있어서,

상기 프로세서는,

두개의 부모해들 각각에 포함된 문자열들 각각의 일부를 교환하여 두개의 자식해들을 생성하는 태스크 집합의 스케줄링 최적화 장치.

청구항 13

제11항에 있어서,

상기 프로세서는,

부모해들 각각에 포함된 문자열들의 값을 랜덤하게 변경하여 자식해들을 생성하는 태스크 집합의 스케줄링 최적화 장치.

청구항 14

제10항에 있어서,

상기 프로세서는,

스케줄링에 따른 전력 소모량을 기초로 상기 해집합에 포함된 해들 각각의 적합도를 평가하고,

해집합의 크기 및 적합도에 따라 상기 해집합에 포함된 해들과 상기 자식해들을 선택하여 상기 해집합을 업데이트하는 태스크 집합의 스케줄링 최적화 장치.

청구항 15

제10항에 있어서,

상기 프로세서는,

상기 태스크들 각각의 스케줄 가능성을 테스트하고, 상기 태스크들 중에서 스케줄 가능성 테스트를 통과하지 못하는 태스크가 발생하는 경우, 상기 테스트를 통과하지 못하는 태스크를 기초로 생성한 해의 적합도에 페널티를 부여하는 태스크 집합의 스케줄링 최적화 장치.

청구항 16

제15항에 있어서,

상기 프로세서는,

태스크가 엡지 서버에서 실행되는 경우의 최악 실행시간, 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간, 태스크들 각각의 업로드 시간 및 다운로드 시간을 기초로 임계 주기를 결정하고, 태스크의 실행 주기를 상기 임계 주기와 비교하여 엡지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트하는 태스크 집합의 스케줄링 최적화 장치.

청구항 17

제15항에 있어서,

상기 프로세서는,

태스크가 로컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행시간과 태스크의 실행 주기, DRAM과 NVM 간의 읽기/쓰기 연산 및 읽기/쓰기 지연 시간 및 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간에 기초한 이용률 테스트를 수행하여 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트하는 태스크 집합의 스케줄링 최적화 장치.

발명의 설명

기술 분야

본 발명은 태스크 집합의 스케줄링 최적화 방법 및 장치에 관한 것으로, 보다 구체적으로는 태스크 별 실행 위

치, CPU 모드, 메모리 배치를 최적화하는 방법 및 장치에 관한 것이다.

배경 기술

- [0002] 사물인터넷, 사이버 물리 시스템 등의 기술 발전으로 모바일 실시간 시스템의 전력소모 절감 방법이 요구되고 있다.
- [0003] 모바일 기기의 전력 소모를 절감하기 위하여 CPU의 처리 속도를 조절하는 DVFS(dynamic voltage frequency scaling) 기술이 개발되었다. DVFS는 태스크의 작업 부하가 낮을 경우 클럭 빈도와 공급 전압을 낮추어 CPU의 소모 전력을 줄이도록 태스크 집합을 스케줄링 하는 방법이다.
- [0004] 그러나, DVFS는 CPU의 처리 속도만을 제어하고 있으므로, 메모리 및 네트워크에서 소모되는 전력은 절감할 수 없다는 한계가 있었다.
- [0005] 모바일 기기의 전력 소모는 CPU, 메모리, 네트워크에서 모두 발생하고 있으므로, CPU, 메모리, 네트워크 모두의 전력 소모를 고려하여 전력 소모가 최적화되도록 태스크를 스케줄링 하는 방법이 요청되고 있다.

발명의 내용

해결하려는 과제

- [0006] 본 발명은 태스크별로 오프로딩 여부, CPU의 클럭 빈도, 및 메모리 위치로 구성된 해들의 해집합을 생성하고, 해집합에 진화 연산을 적용함으로써, 태스크 별 실행 위치, CPU 모드, 메모리 배치를 최적화하여 태스크들을 실행하는 타겟시스템의 전력 절감을 극대화하는 방법 및 장치를 제공한다.

과제의 해결 수단

- [0007] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법은 태스크 집합에 포함된 태스크들 각각의 실행 위치, CPU 모드 및 메모리 배치를 기초로 초기 해들을 생성하는 단계; 상기 초기 해들로 구성된 해집합에서 복수의 해들을 부모해들로 선택하는 단계; 상기 부모해들을 기초로 자식해들을 생성하는 단계; 상기 자식해들을 이용하여 해집합을 업데이트하는 단계; 및 업데이트된 해집합에 포함된 해들이 수렴하는 경우, 수렴한 해에 기초하여 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 스케줄링 하는 단계를 포함할 수 있다.
- [0008] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법의 상기 초기 해들은, 태스크 별로 오프로딩 여부를 나타내는 문자열, CPU의 클럭 빈도를 나타내는 문자열, 및 메모리 위치를 나타내는 문자열로 구성될 수 있다.
- [0009] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법의 상기 자식해들을 생성하는 단계는, 두개의 부모해들 각각에 포함된 문자열들 각각의 일부를 교환하여 두개의 자식해들을 생성할 수 있다.
- [0010] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법의 상기 자식해들을 생성하는 단계는, 부모해들 각각에 포함된 문자열들의 값을 랜덤하게 변경하여 자식해들을 생성할 수 있다.
- [0011] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법은 스케줄링에 따른 전력 소모량을 기초로 상기 해집합에 포함된 해들 각각의 적합도를 평가하는 단계;를 더 포함하고, 상기 해집합을 업데이트하는 단계는, 해집합의 크기 및 적합도에 따라 상기 해집합에 포함된 해들과 상기 자식해들을 선택하여 상기 해집합을 업데이트할 수 있다.
- [0012] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법의 상기 부모해들로 선택하는 단계는, 상기 초기 해들 각각의 적합도에 기초하여 기 설정된 숫자의 초기 해들을 상기 부모해들로 선택할 수 있다.
- [0013] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법은 상기 태스크들 각각의 스케줄 가능성을 테스트하는 단계; 및 상기 태스크들 중에서 스케줄 가능성 테스트를 통과하지 못하는 태스크가 발생하는 경우, 상기 테스트를 통과하지 못하는 태스크를 기초로 생성한 해의 적합도에 페널티를 부여하는 단계를 더 포함할 수 있다.
- [0014] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법의 상기 스케줄 가능성을 테스트하는 단계는, 태스크가 옛터 서버에서 실행되는 경우의 최악 실행시간, 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간, 태스크들 각각의 업로드 시간 및 다운로드 시간을 기초로 임계 주기를 결정하고, 태스크의 실행 주기를 상

기 임계 주기와 비교하여 엡지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트할 수 있다.

[0015] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법의 상기 스케줄 가능성을 테스트하는 단계는, 태스크가 로컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행시간과 태스크의 실행 주기, DRAM과 NVM 간의 읽기/쓰기 연산 및 읽기/쓰기 지연 시간 및 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간에 기초한 이용률 테스트를 수행하여 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트할 수 있다.

[0016] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치는 태스크 집합에 포함된 태스크들 각각의 실행 위치, CPU 모드 및 메모리 배치를 기초로 초기 해들을 생성하고, 상기 초기 해들로 구성된 해집합에서 복수의 해들을 부모해들로 선택하며, 상기 부모해들을 기초로 자식해들을 생성하고, 상기 자식해들을 이용하여 해집합을 업데이트하며, 업데이트된 해집합에 포함된 해들이 수렴하는 경우, 수렴한 해에 기초하여 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 스케줄링 하는 프로세서를 포함할 수 있다.

[0017] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치의 상기 초기 해들은, 태스크 별로 오프로딩 여부를 나타내는 문자열, CPU의 클럭 빈도를 나타내는 문자열, 및 메모리 위치를 나타내는 문자열로 구성될 수 있다.

[0018] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치의 상기 프로세서는, 두개의 부모해들 각각에 포함된 문자열들 각각의 일부를 교환하여 두개의 자식해들을 생성할 수 있다.

[0019] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치의 상기 프로세서는, 부모해들 각각에 포함된 문자열들의 값을 랜덤하게 변경하여 자식해들을 생성할 수 있다.

[0020] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치의 상기 프로세서는, 스케줄링에 따른 전력 소모량을 기초로 상기 해집합에 포함된 해들 각각의 적합도를 평가하고, 해집합의 크기 및 적합도에 따라 상기 해집합에 포함된 해들과 상기 자식해들을 선택하여 상기 해집합을 업데이트할 수 있다.

[0021] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치의 상기 프로세서는, 상기 태스크들 각각의 스케줄 가능성을 테스트하고, 상기 태스크들 중에서 스케줄 가능성 테스트를 통과하지 못하는 태스크가 발생하는 경우, 상기 테스트를 통과하지 못하는 태스크를 기초로 생성한 해의 적합도에 페널티를 부여할 수 있다.

[0022] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치의 상기 프로세서는, 태스크가 엡지 서버에서 실행되는 경우의 최악 실행시간, 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간, 태스크들 각각의 업로드 시간 및 다운로드 시간을 기초로 임계 주기를 결정하고, 태스크의 실행 주기를 상기 임계 주기와 비교하여 엡지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트할 수 있다.

[0023] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치의 상기 프로세서는, 태스크가 로컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행시간과 태스크의 실행 주기, DRAM과 NVM 간의 읽기/쓰기 연산 및 읽기/쓰기 지연 시간 및 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간에 기초한 이용률 테스트를 수행하여 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트할 수 있다.

발명의 효과

[0024] 본 발명의 일실시예에 의하면, 태스크별로 오프로딩 여부, CPU의 클럭 빈도, 및 메모리 위치로 구성된 해들의 해집합을 생성하고, 해집합에 진화 연산을 적용함으로써, 태스크 별 실행 위치, CPU 모드, 메모리 배치를 최적화하여 태스크들을 실행하는 타겟 시스템의 전력 절감을 극대화할 수 있다.

[0025] 또한, 본 발명의 일실시예에 의하면, 가변적인 네트워크 상황에서 오프로딩 없이 로컬 자원만으로 가능한 스케줄을 탐색하며, 변화하는 상황에 맞게 오프로딩, DVFS, 메모리 배치 등을 최적화하여 전력 절감을 극대화할 수 있다.

도면의 간단한 설명

[0026] 도 1은 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치를 도시한 도면이다.

도 2는 본 발명의 일실시예에 따른 스케줄링 결과의 일례이다.

도 3은 본 발명의 일실시예에 따라 생성한 초기 해의 일례이다.

도 4는 본 발명의 일실시예에 따라 부모해들로부터 자식해들을 생성하는 과정의 일례이다.

도 5는 본 발명의 일실시예에서 시스템 구성들 각각의 알고리즘에 따른 전력 소비량의 일례이다.

도 6은 본 발명의 일실시예에서 태스크 집합의 부하에 따른 알고리즘별 전력 소비량의 변화의 일례이다.

도 7은 본 발명의 일실시예에서 태스크 집합의 부하에 따른 시스템 구성들 각각의 전력 소비량의 변화의 일례이다.

도 8은 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법을 도시한 플로우차트이다.

발명을 실시하기 위한 구체적인 내용

- [0027] 이하에서, 첨부된 도면을 참조하여 실시예들을 상세하게 설명한다. 그러나, 실시예들에는 다양한 변경이 가해질 수 있어서 특허출원의 권리 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 실시예들에 대한 모든 변경, 균등물 내지 대체물이 권리 범위에 포함되는 것으로 이해되어야 한다.
- [0028] 실시예에서 사용한 용어는 단지 설명을 목적으로 사용된 것으로, 한정하려는 의도로 해석되어서는 안된다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 명세서 상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0029] 또한, 첨부 도면을 참조하여 설명함에 있어, 도면 부호에 관계없이 동일한 구성 요소는 동일한 참조부호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다. 실시예를 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 실시예의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다.
- [0030] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.
- [0031] 도 1은 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치를 도시한 도면이다.
- [0032] 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 장치(100)는 도 1에 도시된 바와 같이 입력부(110), 프로세서(120), 및 출력부(130)를 포함할 수 있다.
- [0033] 입력부(110)는 타겟 시스템에서 수행될 태스크와 관련된 정보를 입력받을 수 있다. 예를 들어, 입력부(110)는 유무선 통신으로 정보를 수신하는 통신 인터페이스, 저장 매체와 결합되어 저장 매체에 저장된 정보를 로딩하는 포트, 및 사용자가 프로세서(120)에 정보를 입력하기 위하여 사용하는 입력 인터페이스 중 하나일 수 있다. 예를 들어, 타겟 시스템은 DVFS(dynamic voltage frequency scaling) 기능을 가진 CPU와, DRAM과 NVM으로 구성된 메모리, 및 네트워크 어댑터를 포함하는 모바일 디바이스일 수 있다. 또한, 타겟 시스템은 오프로딩이 가능한 엣지 서버들과 유무선 통신으로 연결될 수 있다.
- [0034] 프로세서(120)는 태스크들의 집합인 태스크 집합을 정의할 수 있다. 프로세서(120)가 정의하는 태스크 집합 Γ 은 수학식 1과 같이 나타낼 수 있다.

수학식 1

$$\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$$

[0035]

- [0036] 태스크는 로컬 CPU 또는 엣지 서버에서 실행될 수 있다. 따라서, 태스크 집합 Γ 에 포함된 태스크들 각각은 실행되는 위치를 기준으로 로컬 CPU에서 실행되는 태스크들의 집합인 LOCAL과 엣지 서버에서 실행되는 태스크들의 집합인 REMOTE 중 하나로 분류될 수 있다.

[0037] 또한, 태스크는 DRAM과 NVM 중 하나의 메모리에 위치할 수 있다. 따라서, 태스크 집합 Γ 에 포함된 태스크들 각각은 메모리를 기준으로 DRAM에 위치한 태스크들의 집합과 NVM에 위치한 태스크들의 집합 중 하나로 분류될 수 있다.

[0038] 그리고, 태스크 τ_i 는 수학적식 2와 같이 정의될 수 있다.

수학적식 2

[0039]
$$\tau_i = \langle WCET_i, Period_i, Data_i \rangle$$

[0040] 이때, $WCET_i$ 는 τ_i 가 로컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행 시간을 나타낼 수 있다. 최악 실행 시간(WCET: worst-case execution time)은 태스크를 대상 하드웨어에서 수행하는데 걸리는 가장 긴 시간일 수 있다. 즉, $WCET_i$ 는 τ_i 가 로컬 CPU에서 기본 클럭 빈도로 수행하는데 걸리는 시간들 중 가장 긴 시간일 수 있다.

[0041] 또한, $Period_i$ 는 τ_i 의 실행주기를 나타내고, $Data_i$ 는 τ_i 의 데이터 특성을 나타낼 수 있다. 예를 들어, $Data_i$ 는 수학적식 3과 같이 정의될 수 있다.

수학적식 3

[0042]
$$Data_i = \langle Size_i, Input_i, Output_i, Rd_i, Wr_i \rangle$$

[0043] 이때, $Size_i$ 는 τ_i 의 메모리 풋프린트이고, $Input_i$ 는 τ_i 실행을 위한 입력 데이터이며, $Output_i$ 는 τ_i 가 실행된 후 τ_i 의 실행 결과로 출력하는 출력 데이터 크기일 수 있다. 또한, Rd_i 는 τ_i 실행 시의 메모리 읽기 쓰기 연산의 수이고, Wr_i 는 τ_i 실행 시의 메모리 쓰기 연산의 수일 수 있다.

[0044] 로컬 CPU는 $\tau_i \in \text{REMOTE}$ 인 태스크를 수행하기 위하여 타겟 시스템의 네트워크 어댑터에게 $Size_i$ 를 엡지 서버로 전송하도록 요청할 수 있다. 네트워크 어댑터는 요청에 따라 $Size_i$ 를 엡지 서버로 전송할 수 있다. 그리고, 엡지 서버에서 τ_i 의 수행이 종료되면, 타겟 시스템의 네트워크 어댑터가 엡지 서버로부터 $Output_i$ 를 수신할 수 있다.

[0045] 또한, 데드라인은 $Period_i$ 에 의해 묵시적으로 결정될 수 있다. 그리고, 모든 태스크 $\tau_i \in \Gamma$ 에 대해

$Epoch_{\Gamma}$ 는 $Period_i$ 의 최소 공배수를 의미할 수 있다.

[0046] 프로세서(120)는 태스크 집합에 포함된 태스크들 각각의 스케줄 가능성을 테스트할 수 있다.

[0047] 프로세서(120)는 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 다음과 같이 테스트할 수 있다.

[0048] 먼저, 프로세서(120)는 태스크가 로컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행 시간과 태스크의 실행

주기에 기초한 이용률 테스트를 수행할 수 있다. 구체적으로, 프로세서(120)는 태스크 τ_i 들 중 로컬 CPU에서 수행되는 태스크가 엡지 서버의 도움 없이 로컬 CPU에서 최대 클럭 빈도로 가동하는 경우, 수학식 4와 같은 이용률 테스트를 만족하는지 여부를 확인할 수 있다.

수학식 4

$$Util_{CPU} = \sum_{\tau_i \in \Gamma} \frac{WCET_i}{Period_i} \leq 1$$

[0049]

[0050] 로컬 CPU에서 수행되는 태스크가 수학식 4를 만족하는 경우, 프로세서(120)는 태스크 집합에 선점형 스케줄링인 EDF(earliest deadline first) 알고리즘을 사용하여 태스크 집합에서 가능한 스케줄을 결정할 수 있다. 또한, 프로세서(120)가 DVFS 기술을 이용하여 CPU 처리 속도를 제어하는 경우, 태스크들 각각의 최악 실행 시간은 CPU 클럭 빈도에 따라 재계산될 수 있다. 따라서, 프로세서(120)는 수학식 5와 같은 이용률 테스트를 만족하는지 여부를 확인할 수 있다.

수학식 5

$$Util_{CPU} = \sum_{\tau_i \in \Gamma} \frac{DVFS_Scaled(WCET_i)}{Period_i} \leq 1$$

[0051]

[0052] 이때, $DVFS_Scaled(WCET_i)$ 는 DVFS를 적용한 태스크 τ_i 의 최악 실행 시간일 수 있다.

[0053] 또한, 프로세서(120)는 DRAM과 NVM 간의 읽기/쓰기 연산 및 읽기/쓰기 지연 시간에 기초한 이용률 테스트를 수행할 수 있다. 프로세서(120)가 PM(persistent memory) 기술을 사용하여 메모리를 배치하는 경우, 프로세서

(120)는 PM을 적용한 태스크 τ_i 의 최악 실행 시간인 $DVFS_PM_Scaled(WCET_i)$ 를 결정할

수 있다. 예를 들어, 프로세서(120)는 수학식 6을 이용하여 $DVFS_PM_Scaled(WCET_i)$ 를 결정할 수 있다.

수학식 6

$$\begin{aligned}
 DVFS_PM_Scaled(WCET_i) \\
 = DVFS_Scaled(WCET_i) + \\
 \Delta_{Read} * Read_i + \Delta_{Write} * Write_i
 \end{aligned}$$

이때, Δ_{Read} 는 DRAM과 NVM 간의 읽기 지연 시간차이고, Δ_{Write} 는 DRAM과 NVM 간의 쓰기 지연 시간차일 수 있다. 그리고, 프로세서(120)는 수학식 7과 같은 이용률 테스트를 만족하는지 여부를 확인할 수 있다.

수학식 7

$$\begin{aligned}
 Util_{CPU} &= \sum_{\tau_i \in DRAM} \frac{DVFS_Scaled(WCET_i)}{Period_i} \\
 &+ \sum_{\tau_i \in PM} \frac{DVFS_PM_Scaled(WCET_i)}{Period_i} \\
 &\leq 1
 \end{aligned}$$

그리고, 프로세서(120)는 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간에 기초한 이용률 테스트를 수행할 수 있다. 타겟 시스템은 태스크를 엷지 서버로 오프로드할 수 있으므로, 프로세서(120)는 수학식 8과 같은 이용률 테스트를 만족하는지 여부를 확인할 수 있다.

수학식 8

$$\begin{aligned}
 Util_{CPU} &= \sum_{\tau_i \in LOCAL \cap DRAM} \frac{DVFS_Scaled(WCET_i)}{Period_i} \\
 &+ \sum_{\tau_i \in LOCAL \cap PM} \frac{DVFS_PM_Scaled(WCET_i)}{Period_i} \\
 &+ \sum_{\tau_i \in REMOTE} \frac{Network_Command_i}{Epoch_{\Gamma}} \\
 &\leq 1
 \end{aligned}$$

이때, $Network_Command_i$ 는 네트워크 모듈이 매 $Epoch_{\Gamma}$ 마다 엷지 서버에 배정된 태스크 τ_i 의 오프로딩을 진행하도록 CPU가 지시하는 커맨드 시간일 수 있다.

즉, 프로세서(120)는 태스크가 수학식 8을 만족하는 경우, 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성 테스트를 통과한 것으로 판단할 수 있다.

태스크의 실행위치와 CPU 모드가 결정된 경우, 프로세서(120)는 태스크들을 스케줄링할 수 있다. 이때, 프로세

서(120)는 $Epoch_{\Gamma}$ 이 시작될 때 REMOTE에 속한 태스크에 대해 CPU가 가장 먼저 네트워크 모듈에 오프로딩 지시

를 내려서 로컬과 원격 태스크 실행의 병렬성을 최대한 높이도록 태스크를 스케줄링할 수 있다. 스케줄링 결과에 따라 REMOTE로 분류되는 태스크의 주기가 시작되는 경우, 타겟 시스템의 네트워크 모듈은 해당 태스크의 오프로딩을 시작할 수 있다. 또한, 스케줄링 결과에 따라 LOCAL로 분류되는 태스크의 주기가 시작되는 경우, 로컬 CPU는 EDF 알고리즘에 의해 해당 태스크들을 실행할 수 있다.

[0062] 프로세서(120)는 엣지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 다음과 같이 테스트할 수 있다.

[0063] 프로세서(120)는 태스크가 엣지 서버에서 실행되는 경우의 최악 실행 시간을 결정할 수 있다. 다음으로, 프로세서(120)는 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간을 결정할 수 있다. 그 다음으로, 프로세서(120)는 최악 실행 시간, 커맨드 시간, 태스크들 각각의 업로드 시간 및 다운로드 시간을 기초로 임계 주기를 결정할 수 있다. 마지막으로 프로세서(120)는 태스크의 실행 주기를 임계 주기와 비교하여 스케줄 가능성 테스트를 수행할 수 있다.

[0064] 예를 들어, 프로세서(120)는 수학적 식 9에 따라 임계 주기 $Turnaround_Time_i$ 를 결정할 수 있다.

수학적 식 9

$$Turnaround_Time_i = Upload_time_i + Scaled_Edge(WCET_i) + Download_time_i$$

[0065]

[0066] 이때, $Scaled_Edge(WCET_i)$ 는 τ_i 가 엣지 서버에서 실행될 경우의 최악 실행 시간을 나타낼 수 있다. 프로세서(120)는 타겟 시스템의 전력 소모를 최소화할 수 있도록 스케줄링하고 있으므로, 타겟 시스템에 포함되지 않은 엣지 서버의 전력 소모는 고려하지 않을 수 있다. 예를 들어, 프로세서(120)는 로컬 CPU가 최대의 클럭 빈도를 사용하지 않도록 스케줄링하며, 엣지 서버가 최대의 클럭 빈도를 사용하도록 스케줄링할 수

있다. 따라서, 프로세서는 기 설정된 임의의 시간, 또는 엣지 서버가 최대의 클럭 빈도를 사용하는 시간을 τ_i 가 엣지 서버에서 실행될 경우의 최악 실행 시간으로 설정할 수 있다. 기 설정된 임의의 시간은 엣지 서버가 최소의 클럭 빈도를 사용하는 시간 내지 엣지 서버가 최대의 클럭 빈도를 사용하는 시간 사이의 시간 중 하나일 수 있다.

[0067] 그리고, 프로세서(120)는 수학적 식 10에 따라 업로드 시간 $Upload_time_i$ 를 결정할 수 있다.

수학적 식 10

$$Upload_time_i = \begin{cases} \frac{Task_size_i}{Network_{up}} & \text{if first execution} \\ \frac{Input_size_i}{Network_{up}} & \text{otherwise} \end{cases}$$

[0068]

[0069] 이때, $Network_{up}$ 은 업링크의 대역폭일 수 있다.

[0070] 프로세서(120)는 수학적 식 11에 따라 다운로드 시간 $Download_time_i$ 를 결정할 수 있다.

수학식 11

$$Download_time_i = \frac{Output_size_i}{Network_{down}}$$

[0071]

[0072] 이때, $Network_{dn}$ 은 다운링크의 대역폭일 수 있다.

[0073] 그리고, 프로세서(120)는 태스크가 수학식 12를 만족하는 경우, 엣지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성 테스트를 통과한 것으로 판단할 수 있다.

수학식 12

$$\forall \tau_i \in \text{REMOTE}, Turnaround_time_i < Period_i$$

[0074]

[0075] 즉, 프로세서(120)는 태스크의 실행 주기가 임계 주기를 초과하는 경우, 엣지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성 테스트를 통과한 것으로 판단할 수 있다.

[0076] 프로세서(120)는 태스크 집합에 포함된 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 기초로 초기 해들을 생성할 수 있다. 이때, 프로세서(120)가 생성하는 초기 해들은, 태스크 별로 오프로딩 여부를 나타내는 문자열, CPU의 클럭 빈도를 나타내는 문자열, 및 메모리 위치를 나타내는 문자열로 구성될 수 있다.

[0077] 프로세서(120)는 스케줄링에 따른 전력 소모량을 기초로 해집합에 포함된 해들 각각의 적합도를 평가할 수 있다. 이때, 프로세서(120)는 태스크들 중에서 스케줄 가능성 테스트를 통과하지 못하는 태스크가 발생하는 경우, 테스트를 통과하지 못하는 태스크를 기초로 생성한 해의 적합도에 페널티를 부여할 수 있다. 또한, 프로세서(120)는 해의 CPU 이용률이 1을 초과하는 경우에도 해당 해의 적합도에 페널티를 부여할 수 있다. 예를 들어, 프로세서(120)는 수학식 13에 따라 적합도 $Cost(i)$ 를 결정할 수 있다.**수학식 13**

$$Cost(i) = Energy(i) + \alpha Penalty(i)$$

[0078]

[0079] 이때, $Energy(i)$ 는 해 i 에 의해 스케줄링된 태스크 집합의 전력 소비량, α 는 가중치 파라미터, $Penalty(i)$ 는 해 i 가 스케줄 가능성 테스트를 통과하지 못한 경우에 대한 페널티 함수일 수 있다. 예를 들어, 페널티 함수는 수학식 14와 같이 정의될 수 있다.**수학식 14**

$$Penalty(i) = (Util_{CPU} - 1) + \beta * \sum_{\tau_j \in \text{REMOTE}} (Turnaround_time_j - Period_j)$$

[0080]

[0081] 이때, β 는 태스크의 기한 초과에 대한 가중치 파라미터일 수 있다.

[0082] 프로세서(120)는 초기 해들로 구성된 해집합에서 복수의 해들을 부모해들로 선택할 수 있다. 이때, 프로세서(120)는 초기 해들 각각의 적합도에 기초하여 기 설정된 숫자의 초기 해들을 부모해들로 선택할 수 있다. 예를 들어, 부모해를 4개 선택하도록 설정된 경우, 프로세서(120)는 초기 해들 중에서 적합도가 높은 순서로 4개의

초기 해들을 선택하여 부모해들로 결정할 수 있다. 또한, 프로세서(120)는 적합도가 높을수록 부모해로 선택될 확률이 증가하도록 해집합에 포함된 해들의 부모해 선택 확률을 결정할 수 있다.

[0083] 프로세서(120)는 부모해들을 기초로 자식해들을 생성할 수 있다. 이때, 프로세서(120)는 두개의 부모해들 각각에 포함된 문자열들 각각의 일부를 교환하는 교환 연산을 수행하여 두개의 자식해들을 생성할 수 있다. 또한, 프로세서(120)는 부모해들 각각에 포함된 문자열들의 값을 랜덤하게 변경하는 돌연 변이 연산을 수행하여 자식해들을 생성할 수 있다.

[0084] 프로세서(110)는 자식해들을 이용하여 해집합을 업데이트할 수 있다. 이때, 프로세서(110)는 해집합의 크기 및 적합도에 따라 해집합에 포함된 해들과 자식해들을 선택하여 해집합을 업데이트할 수 있다. 예를 들어, 프로세서(110)는 자식해들의 개수에 따라 해집합에서 적합도가 낮은 순서로 기존의 해들을 제거하고, 자식해들을 해집합에 추가하여 해집합을 업데이트할 수 있다.

[0085] 또한, 프로세서(110)는 해집합에 포함되어 있던 기존의 해들과 자식해들을 적합도에 따라 정렬하고 가장 적합도가 높은 해부터 해집합의 크기에 해당하는 해들을 선택하여 해집합을 업데이트할 수 있다. 예를 들어, 해집합에 포함되어 있던 해 a, b, c, d의 적합도가 각각 0.1, 0.2, 0.3, 0.4이고, 해 a, b를 사용하여 생성한 자식해 e, f의 적합도가 각각 0.5, 0.15인 경우, 프로세서(120)는 해 d를 삭제하고, 자식해 e를 포기하여 해집합이 해 a, b, c, f를 포함하도록 업데이트할 수 있다.

[0086] 프로세서(120)는 업데이트된 해집합에 포함된 해들이 수렴하는지 여부를 확인할 수 있다. 구체적으로 프로세서(120)는 업데이트된 해집합에 포함된 해들 각각의 적합도가 기 설정된 범위 안에 모두 포함된 경우, 해들의 다양성이 임계 수준 이하로 감소하여 수렴한 것으로 판단할 수 있다. 예를 들어, 업데이트된 해집합에 포함된 해들 최대 적합도와 최저 적합도 간의 차이가 0.01인 경우, 프로세서(120)는 해들이 수렴하는 것으로 판단할 수 있다.

[0087] 업데이트된 해집합에 포함된 해들이 수렴하지 않은 경우, 프로세서(120)는 업데이트된 해집합에 포함된 해들이 수렴할 때까지 업데이트된 해집합에 포함된 해들 중에서 부모해를 선택하고, 선택한 부모해를 사용하여 자식해들을 생성하며, 자식해들을 이용하여 해집합을 업데이트하는 과정을 반복할 수 있다.

[0088] 업데이트된 해집합에 포함된 해들이 수렴한 경우, 프로세서(120)는 총 전력 소비량인 E_{total} 이 최소화되도록 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 스케줄링할 수 있다. 총 전력 소비량인 E_{total} 은 수학적 식 15에

도시된 바와 같이 CPU 전력 소비량인 E_{CPU} , 메모리 전력 소비량인 E_{MEMORY} , 및 네트워크 전력 소비량인 $E_{NETWORK}$ 에 따라 결정될 수 있다.

수학적 식 15

[0089]
$$E_{total} = E_{CPU} + E_{MEMORY} + E_{NETWORK}$$

[0090] CPU 전력 소비량인 E_{CPU} 는 수학적 식 16과 같이 정의될 수 있다.

수학적 식 16

[0091]
$$E_{CPU} = \sum_{\tau_i \in LOCAL} CPU_Power(i) * DVFS_Scaled(WCET_i) + \sum_{\tau_i \in REMOTE} CPU_Power(i) * Network_command_i$$

[0092] 이때, $CPU_Power(i)$ 는 수학적식 17과 같이 정의될 수 있다.

수학적식 17

[0093]
$$CPU_Power(i) = c V_i^2 f_i$$

[0094] 이때, f_i 는 태스크 τ_i 를 실행하기 위한 클럭 주파수이고 V_i 는 태스크 τ_i 의 공급 전압이며 c 는 유효 스위치 커패시턴스일 수 있다.

[0095] 네트워크 전력 소비량인 $E_{NETWORK}$ 는 수학적식 18과 같이 정의될 수 있다.

수학적식 18

[0096]
$$E_{NETWORK} = \sum_{\tau_i \in REMOTE} Network_Power(i) * (Upload_time_i + Download_time_i)$$

[0097] 메모리 전력 소비량인 E_{MEMORY} 는 수학적식 19와 같이 정의될 수 있다.

수학적식 19

[0098]
$$E_{MEMORY} = E_{M_dynamic} + E_{M_static}$$

[0099] 이때, 동적 전력 소비량 $E_{M_dynamic}$ 은 메모리의 읽기 동작, 또는 쓰기 동작에 따라 소비되는 전력량이며, 수학적식 20과 같이 정의될 수 있다. 또한, 정적 전력 소비량 E_{M_static} 은 메모리의 읽기 동작, 및 쓰기 동작에 무관하게 지속적으로 소모되는 전력량이며, 수학적식 21과 같이 정의될 수 있다.

수학적식 20

[0100]
$$E_{M_dynamic} = \sum_{\tau_i \in DRAM} (read_i * DRAM_Energy_{read} + write_i * DRAM_Energy_{write}) + \sum_{\tau_i \in PM} (read_i * PM_Energy_{read} + write_i * PM_Energy_{write})$$

[0101] 이때, $DRAM_Energy_{read}$ 는 DRAM의 복수의 기본 액세스 단위에 대한 읽기 전력 소비량이고, $DRAM_Energy_{write}$ 는 DRAM의 복수의 기본 액세스 단위에 대한 쓰기 전력 소비량일 수 있다. 또

한, PM_Energy_{read} 는 PM의 복수의 기본 액세스 단위에 대한 읽기 전력 소비량이고, PM_Energy_{write} 는 PM의 복수의 기본 액세스 단위에 대한 쓰기 전력 소비량일 수 있다.

수학식 21

$$E_{M_static} = \sum_{\tau_i \in DRAM} (DRAM_power * Task_size_i * T) + \sum_{\tau_i \in PM} (PM_power * Task_size_i * T)$$

[0102]

[0103] $DRAM_power$ 는 용량당 DRAM의 정적 전력이고, PM_power 는 용량당 PM의 정적 전력이며, T는 작업 세트의 총 실행 시간일 수 있다.

[0104] 구체적으로, 프로세서(120)는 수렴한 해에 기초하여 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 스케줄링할 수 있다. 해들은 앞서 설명한 바와 같이 태스크들 각각이 로컬 CPU와 엣지 서버 중 어디서 실행되는지, 태스크들 각각이 어떤 메모리에 배치되는지, 및 태스크들 각각의 CPU의 클럭 빈도를 나타내는 문자열들로 구성되어 있으므로, 프로세서(120)는 수렴한 해에 포함된 문자열들을 이용하여 태스크들의 실행 위치, CPU 모드 및 태스크들이 배치되는 메모리를 결정할 수 있다.

[0105] 출력부(130)는 스케줄링한 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 출력하여 사용자에게 제공할 수 있다. 예를 들어, 출력부(130)는 유무선 통신으로 정보를 전송하는 통신 인터페이스, 저장 매체와 결합되어 저장 매체에 정보를 저장하는 포트, 및 디스플레이 중 하나일 수 있다.

[0106] 태스크 집합의 스케줄링 최적화 장치(100)는 태스크별로 오프로딩 여부, CPU의 클럭 빈도, 및 메모리 위치로 구성된 해들의 해집합을 생성하고, 해집합에 진화 연산을 적용함으로써, 태스크 별 실행 위치, CPU 모드, 메모리 배치를 최적화하여 태스크들을 실행하는 타겟 시스템의 전력 절감을 극대화할 수 있다.

[0107] 태스크 집합의 스케줄링 최적화 장치(100)는 가변적인 네트워크 상황에서 오프로딩 없이 로컬 자원만으로 가능한 스케줄을 탐색하며, 변화하는 상황에 맞게 오프로딩, DVFS, 메모리 배치 등을 최적화하여 전력 절감을 극대화할 수 있다.

[0108] 도 2는 본 발명의 일실시예에 따른 스케줄링 결과의 일례이다.

[0109] 태스크 집합(210)에 3개의 태스크 τ_1 , τ_2 , τ_3 가 포함되고, 태스크들 각각의 최악 실행 시간이 각각 10, 5, 2이고 실행 주기가 각각 24, 16, 12일 수 있다.

[0110] 이때, 수학식 4에 따르면, $Util_{cpu} = 10/24 + 5/16 + 2/12 = 0.896$ 이므로 1 미만이 되어 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성 테스트를 통과할 수 있다.

[0111] 종래의 태스크 스케줄 최적화 장치에 의한 스케줄링 결과(220)에 따르면, 모든 태스크들이 데드라인 내에 처리

가능한 것을 확인할 수 있다. 또한, 스케줄링 결과(220)에 따르면, 매 $Epoch_T$ 마다 5 단위시간만큼의 빈 슬롯이 발생하는 것을 확인할 수 있다.

[0112] 일정 크기 이상의 빈 슬롯은 DVFS 기술을 사용해서 감소시킬 수 있으나, 스케줄링 결과(220)에서 발생한 빈 슬롯은 크기가 너무 작아 DVFS 적용이 어려울 수 있다. CPU의 기본 클럭 빈도를 1.0이라 하고 이를 0.5, 0.25,

0.125의 4단계로 조절이 가능하다고 할 때 가장 실행시간이 짧은 τ_3 에 클럭 빈도 0.5를 적용하더라도

$DVFS(WCET_3)$ 는 4가 되어 CPU 이용률은 $Util_{CPU} = 10/24 + 5/16 + 4/12 = 1.0625$, 즉 1이 넘어 스케줄링이 불가능할 수 있다.

[0113] 태스크 집합의 스케줄링 최적화 장치(100)는 오프로딩과 DVFS를 함께 사용함으로써, 스케줄링 결과(230)를 출력할 수 있다. 스케줄링 결과(230)는 오프로딩 이후 생긴 CPU의 빈 슬롯에 DVFS를 적용한 것을 나타낼 수 있다.

[0114] 도 2의 실시예에서 엣지 서버의 클럭 빈도는 로컬 CPU의 2배인 2.0일 수 있다. 도 2에 도시된 바와 같이 태스

크 τ_1 이 오프로딩 되면, 태스크 τ_3 의 수행을 위한 CPU 클럭 빈도(240)가 0.25로 낮아지며 스케줄링 결과(230)에서 발생하던 빈 CPU 슬롯이 제거되어 전력 소모를 감소시킬 수 있다.

[0115] 오프로딩 비율이 증가할수록 로컬 CPU의 전력 소모가 감소하지만, 네트워크 상태가 좋지 않은 경우 CPU와 네트워크의 전력 소모 간에 상쇄 효과가 발생할 수 있다. 또한, 일부 태스크는 그 고유의 특성 때문에 오프로딩 자체가 불가능하여 반드시 로컬 CPU에서 처리되어야 할 수 있다. 따라서, 태스크 집합의 스케줄링 최적화 장치(100)는 서로 다른 오프로딩 비율을 가지는 해들 각각의 적합도를 평가하고, 해들의 해집합을 업데이트하여 수렴된 해들 각각에 저장된 오프로딩 비율에 따라 태스크 집합을 스케줄링함으로써, 스케줄링 결과에 포함되는 오프로딩 비율을 최적화할 수 있다.

[0116] 도 3은 본 발명의 일실시예에 따라 생성한 초기 해의 일례이다.

[0117] 해집합에 포함되는 해들은 태스크 별로 오프로딩 여부를 나타내는 문자열(310), 메모리 위치를 나타내는 문자열(320), 및 CPU의 클럭 빈도를 나타내는 문자열(330)로 구성될 수 있다.

[0118] 프로세서(120)는 태스크의 오프로딩을 단일계층 엣지 서버로 정의하고, 태스크의 오프로딩 여부에 따라 태스크의 상태를 {0, 1}로 정의하여 문자열(310)을 생성할 수 있다.

[0119] 프로세서(120)는 메모리 위치가 DRAM인 경우, 0으로 표시하고, 메모리 위치가 NVM인 경우 1로 표시하여 문자열(320)을 생성할 수 있다.

[0120] 프로세서(120)는 클럭 빈도에 근거하여 CPU의 수행 속도를 {1, 0.5, 0.25, 0.125} 4가지로 정의하여 문자열(330)을 생성할 수 있다. 예를 들어, CPU의 수행 속도가 1인 경우, 프로세서(120)는 문자열(330)에 0을 표시하고, CPU의 수행 속도가 0.5인 경우, 프로세서(120)는 문자열(330)에 1을 표시할 수 있다. 또한, CPU의 수행 속도가 0.25인 경우, 프로세서(120)는 문자열(330)에 2를 표시하고, CPU의 수행 속도가 0.125인 경우, 프로세서(120)는 문자열(330)에 3을 표시할 수 있다.

[0121] 도 4는 본 발명의 일실시예에 따라 부모해들로부터 자식해들을 생성하는 과정의 일례이다.

[0122] 프로세서(120)는 해집합에 포함된 해들 중에서 2개의 해들을 부모해(410)와 부모해(420)로 선택할 수 있다.

[0123] 부모해(410)는 문자열(411), 문자열(412), 문자열(413)로 구성되고, 부모해(420)는 문자열(421), 문자열(422), 문자열(423)로 구성될 수 있다. 이때, 문자열(411), 문자열(421)은 도 3의 문자열(310)과 같이 오프로딩 여부를 나타내는 문자열일 수 있다. 또한, 문자열(412), 문자열(422)은 도 3의 문자열(320)과 같이 메모리 위치를 나타내는 문자열일 수 있다. 그리고, 문자열(413) 문자열(423)은 도 3의 문자열(330)과 같이 CPU의 클럭 빈도를 나타내는 문자열일 수 있다.

[0124] 단계(431)에서 프로세서(120)는 부모해(410)에 포함된 문자열(411)과 부모해(420)에 포함된 문자열(421)을 추출하고, 랜덤하게 절단점을 설정할 수 있다.

[0125] 단계(432)에서 프로세서(120)는 문자열(411)의 절단점 아래 부분을 문자열(421)의 절단점 아래 부분과 교체하여 문자열(461)과 문자열(471)을 생성할 수 있다. 이때, 문자열(461)은 문자열(411)의 절단점 위부분과 문자열(421)의 절단점 아랫부분이 결합된 문자열일 수 있다. 또한, 문자열(471)은 문자열(421)의 절단점 위부분과 문자열(411)의 절단점 아랫부분이 결합된 문자열일 수 있다.

[0126] 단계(441)에서 프로세서(120)는 부모해(410)에 포함된 문자열(412)과 부모해(420)에 포함된 문자열(422)을 추출

하고, 랜덤하게 절단점을 설정할 수 있다. 이때, 프로세서(120)가 설정하는 절단점의 위치는 단계(431)에서 설정한 절단점의 위치와 동일할 수도 있고 다를 수도 있다.

- [0127] 단계(442)에서 프로세서(120)는 문자열(412)의 절단점 아래 부분을 문자열(422)의 절단점 아래 부분과 교체하여 문자열(462)과 문자열(472)을 생성할 수 있다. 이때, 문자열(462)은 문자열(412)의 절단점 위부분과 문자열(422)의 절단점 아랫부분이 결합된 문자열일 수 있다. 또한, 문자열(472)은 문자열(422)의 절단점 위부분과 문자열(412)의 절단점 아랫부분이 결합된 문자열일 수 있다.
- [0128] 단계(451)에서 프로세서(120)는 부모해(410)에 포함된 문자열(413)과 부모해(420)에 포함된 문자열(423)을 추출하고, 랜덤하게 절단점을 설정할 수 있다. 이때, 프로세서(120)가 설정하는 절단점의 위치는 단계(431)에서 설정한 절단점의 위치 및 단계(431)에서 설정한 절단점의 위치와 동일할 수도 있고 다를 수도 있다.
- [0129] 단계(452)에서 프로세서(120)는 문자열(413)의 절단점 아래 부분을 문자열(423)의 절단점 아래 부분과 교체하여 문자열(463)과 문자열(473)을 생성할 수 있다. 이때, 문자열(463)은 문자열(413)의 절단점 위부분과 문자열(423)의 절단점 아랫부분이 결합된 문자열일 수 있다. 또한, 문자열(473)은 문자열(423)의 절단점 위부분과 문자열(413)의 절단점 아랫부분이 결합된 문자열일 수 있다.
- [0130] 마지막으로 프로세서(120)는 문자열(461), 문자열(462), 및 문자열(463)을 결합하여 자식해(460)를 생성할 수 있다. 또한, 프로세서(120)는 문자열(471), 문자열(472), 및 문자열(473)을 결합하여 자식해(470)를 생성할 수 있다.
- [0131] 실시예에 따라서는 단계(431, 441, 451)에서 복수의 절단점을 설정할 수도 있고, 설정하는 절단점의 개수를 랜덤하게 결정할 수도 있다. 이때, 단계(432, 442, 452)에서 프로세서(120)는 문자열(411, 412, 413)의 가장 위부터 아래로 내려오면서 절단점을 만날 때마다 문자열(411, 412, 413)의 내용을 문자열(421, 422, 423)와 교체하여 자식해들을 생성할 수 있다.
- [0132] 즉, 프로세서(120)가 교배 연산을 수행하는 경우, 도 4에 도시된 바와 같이 자식해(460, 470)들은 부모해(410, 420)들의 문자열들 각각의 일부를 포함하여 생성된 해일 수 있다.
- [0133] 또한, 실시예에 따라 프로세서(120)는 부모해의 문자열들의 값들 중 일부를 랜덤하게 변경하는 돌연 변이 연산을 이용하여 자식해를 생성할 수도 있다. 예를 들어, 부모해가 도 3에 도시된 문자열들로 구성된 경우, 자식해는 문자열(320)의 마지막 숫자가 1에서 0으로 변경된 해일 수 있다.
- [0134] 그리고, 프로세서(120)는 교배 연산과 돌연변이 연산 중 하나만 수행할 수도 있고, 모두 수행할 수도 있다.
- [0135] 도 5는 기본 스케줄링에 따라 스케줄링한 태스크들의 전력 소비량(510), DVFS만 적용하여 스케줄링한 태스크들의 전력 소비량(520), 오프로딩만 적용하여 스케줄링한 태스크들의 전력 소비량(530), 및 태스크 집합의 스케줄링 최적화 장치(100)가 스케줄링한 태스크들의 전력 소비량(540)의 일례를 도시하고 있다.
- [0136] 도 5에 도시된 바와 같이 태스크 집합의 스케줄링 최적화 장치(100)가 스케줄링한 태스크들의 전력 소비량(540)은 전력 소비량(510), 전력 소비량(520), 및 전력 소비량(530)에 비해 절감될 수 있다.
- [0137] 도 6은 태스크 집합의 부하가 0.2에서 0.9까지 증가하는 태스크들의 전력 소비량(Relative energy)의 변화를 도시하고 있다. 이때, 태스크 집합의 스케줄링 최적화 장치(100)가 스케줄링한 태스크들의 전력 소비량(640)은 도 6에 도시된 바와 같이 태스크 집합의 부하가 증가하더라도, 기본 스케줄링에 따라 스케줄링한 태스크들의 전력 소비량(610), DVFS만 적용하여 스케줄링한 태스크들의 전력 소비량(620), 및 오프로딩만 적용하여 스케줄링한 태스크들의 전력 소비량(630) 보다 낮을 수 있다.
- [0138] 도 7은 태스크 집합의 부하에 따른 CPU 전력 소비량(710), 네트워크 전력 소비량(720), 메모리 전력 소비량(730), 및 토탈 전력 소비량(740)의 전력 소비량의 변화의 일례이다. 도 7에 도시된 바와 같이 부하가 낮을 경우에는 메모리 전력 소비량(730)이 가장 높으나, 부하가 증가할수록 오프로딩의 비율이 증가하여 네트워크 전력 소비량(720)이 증가할 수 있다.
- [0139] 도 8은 본 발명의 일실시예에 따른 태스크 집합의 스케줄링 최적화 방법을 도시한 플로우차트이다.
- [0140] 단계(810)에서 프로세서(120)는 태스크들의 집합인 태스크 집합을 정의할 수 있다. 이때, 태스크 τ_i 는 τ_i 가 로

컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행 시간, t_i 의 실행주기 및 t_i 의 데이터 특성으로 정의될 수 있다.

[0141] 단계(820)에서 프로세서(120)는 태스크 집합에 포함된 태스크들 각각의 스케줄 가능성을 테스트할 수 있다. 이때, 프로세서(120)는 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트할 수 있다. 구체적으로, 프로세서(120)는 태스크가 로컬 CPU에서 기본 클럭 빈도로 실행될 경우의 최악 실행 시간과 태스크의 실행 주기, DRAM과 NVM 간의 읽기/쓰기 연산 및 읽기/쓰기 지연 시간, 및 태스크의 오프로딩을 진행하도록 지시하는 커맨드 시간에 기초한 이용률 테스트를 수행할 수 있다. 프로세서(120)는 수학적 8을 만족하는 경우, 로컬 CPU에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성 테스트를 통과한 것으로 판단할 수 있다.

[0142] 또한, 프로세서(120)는 엣지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성을 테스트할 수 있다. 구체적으로, 프로세서(120)는 최악 실행 시간, 커맨드 시간, 태스크들 각각의 업로드 시간 및 다운로드 시간을 기초로 임계 주기를 결정할 수 있다. 그리고, 태스크의 실행 주기가 임계 주기를 초과하는 경우, 프로세서(120)는 엣지 서버에서 수행되는 태스크의 데드 라인을 보장하기 위한 스케줄 가능성 테스트를 통과한 것으로 판단할 수 있다.

[0143] 단계(830)에서 프로세서(120)는 태스크 집합에 포함된 태스크들 각각의 실행위치, CPU 모드 및 메모리 배치를 기초로 초기 해들을 생성할 수 있다. 이때, 프로세서(120)가 생성하는 초기 해들은, 태스크 별로 오프로딩 여부를 나타내는 문자열, CPU의 클럭 빈도를 나타내는 문자열, 및 메모리 위치를 나타내는 문자열로 구성될 수 있다.

[0144] 단계(840)에서 프로세서(120)는 스케줄링에 따른 전력 소모량을 기초로 해집합에 포함된 해들 각각의 적합도를 평가할 수 있다. 이때, 프로세서(120)는 태스크들 중에서 스케줄 가능성 테스트를 통과하지 못하는 태스크가 발생하는 경우, 테스트를 통과하지 못하는 태스크를 기초로 생성한 해의 적합도에 페널티를 부여할 수 있다. 또한, 프로세서(120)는 해의 CPU 이용률이 1을 초과하는 경우에도 해당 해의 적합도에 페널티를 부여할 수 있다.

[0145] 단계(850)에서 프로세서(120)는 초기 해들로 구성된 해집합에서 복수의 해들을 부모해들로 선택할 수 있다. 이때, 프로세서(120)는 초기 해들 각각의 적합도에 기초하여 기 설정된 숫자의 초기 해들을 부모해들로 선택할 수 있다. 예를 들어, 부모해를 4개 선택하도록 설정된 경우, 프로세서(120)는 초기 해들 중에서 적합도가 높은 순서로 4개의 초기 해들을 선택하여 부모해들로 결정할 수 있다. 또한, 프로세서(120)는 적합도가 높을수록 부모해로 선택될 확률이 증가하도록 해집합에 포함된 해들의 부모해 선택 확률을 결정할 수 있다.

[0146] 단계(860)에서 프로세서(120)는 단계(850)에서 선택한 부모해들을 기초로 자식해들을 생성할 수 있다. 이때, 프로세서(120)는 두개의 부모해들 각각에 포함된 문자열들 각각의 일부를 교환하는 교환 연산을 수행하여 두개의 자식해들을 생성할 수 있다. 또한, 프로세서(120)는 부모해들 각각에 포함된 문자열들의 값을 랜덤하게 변경하는 돌연 변이 연산을 수행하여 자식해들을 생성할 수 있다.

[0147] 단계(870)에서 프로세서(110)는 자식해들을 이용하여 해집합을 업데이트할 수 있다. 이때, 프로세서(110)는 해집합의 크기 및 적합도에 따라 해집합에 포함된 해들과 자식해들을 선택하여 해집합을 업데이트할 수 있다. 예를 들어, 프로세서(110)는 자식해들의 개수에 따라 해집합에서 적합도가 낮은 순서로 기존의 해들을 제거하고, 자식해들을 해집합에 추가하여 해집합을 업데이트할 수 있다.

[0148] 또한, 프로세서(110)는 해집합에 포함되어 있던 기존의 해들과 자식해들을 적합도에 따라 정렬하고 가장 적합도가 높은 해부터 해집합의 크기에 해당하는 해들을 선택하여 해집합을 업데이트할 수 있다.

[0149] 단계(880)에서 프로세서(120)는 업데이트된 해집합에 포함된 해들이 수렴하는지 여부를 확인할 수 있다. 구체적으로 프로세서(120)는 업데이트된 해집합에 포함된 해들 각각의 적합도가 기 설정된 범위 안에 모두 포함된 경우, 해들의 다양성이 임계 수준 이하로 감소하여 수렴한 것으로 판단할 수 있다.

[0150] 업데이트된 해집합에 포함된 해들이 수렴하지 않은 경우, 프로세서(120)는 업데이트된 해집합에 포함된 해들이 수렴할 때까지 업데이트된 해집합에 포함된 해들 중에서 부모해를 선택하고, 선택한 부모해를 사용하여 자식해들을 생성하며, 자식해들을 이용하여 해집합을 업데이트하는 과정을 반복할 수 있다.

[0151] 업데이트된 해집합에 포함된 해들이 수렴한 경우, 프로세서(120)는 수렴한 해에 포함된 문자열들을 이용하여 태

스크들의 실행 위치, CPU 모드 및 태스크들이 배치되는 메모리를 결정할 수 있다.

- [0153] 본 발명은 태스크별로 오프로딩 여부, CPU의 클럭 빈도, 및 메모리 위치로 구성된 해들의 해집합을 생성하고, 해집합에 진화 연산을 적용함으로써, 태스크 별 실행 위치, CPU 모드, 메모리 배치를 최적화하여 태스크들을 실행하는 타겟 시스템의 전력 절감을 극대화할 수 있다.
- [0154] 또한, 본 발명은 가변적인 네트워크 상황에서 오프로딩 없이 로컬 자원만으로 가능한 스케줄을 탐색하며, 변화하는 상황에 맞게 오프로딩, DVFS, 메모리 배치 등을 최적화하여 전력 절감을 극대화할 수 있다.
- [0156] 한편, 본 발명에 따른 발화 데이터에 기초한 태스크 집합의 스케줄링 최적화 장치 또는 발화 데이터에 기초한 태스크 집합의 스케줄링 최적화 방법은 컴퓨터에서 실행될 수 있는 프로그램으로 작성되어 마그네틱 저장매체, 광학적 판독매체, 디지털 저장매체 등 다양한 기록 매체로도 구현될 수 있다.
- [0157] 본 명세서에 설명된 각종 기술들의 구현들은 디지털 전자 회로조직으로, 또는 컴퓨터 하드웨어, 펌웨어, 소프트웨어로, 또는 그들의 조합들로 구현될 수 있다. 구현들은 데이터 처리 장치, 예를 들어 프로그램가능 프로세서, 컴퓨터, 또는 다수의 컴퓨터들의 동작에 의한 처리를 위해, 또는 이 동작을 제어하기 위해, 컴퓨터 프로그램 제품, 예를 들어 기계 판독가능 저장 장치(컴퓨터 판독가능 매체)에서 유형적으로 구체화된 컴퓨터 프로그램으로서 구현될 수 있다. 상술한 컴퓨터 프로그램(들)과 같은 컴퓨터 프로그램은 컴파일된 또는 인터프리트된 언어들을 포함하는 임의의 형태의 프로그래밍 언어로 기록될 수 있고, 독립형 프로그램으로서 또는 모듈, 구성요소, 서브루틴, 또는 컴퓨팅 환경에서의 사용에 적절한 다른 유닛으로서 포함하는 임의의 형태로 전개될 수 있다. 컴퓨터 프로그램은 하나의 사이트에서 하나의 컴퓨터 또는 다수의 컴퓨터들 상에서 처리되도록 또는 다수의 사이트들에 걸쳐 분배되고 통신 네트워크에 의해 상호 연결되도록 전개될 수 있다.
- [0158] 컴퓨터 프로그램의 처리에 적절한 프로세서들은 예로서, 범용 및 특수 목적 마이크로프로세서들 둘 다, 및 임의의 종류의 디지털 컴퓨터의 임의의 하나 이상의 프로세서들을 포함한다. 일반적으로, 프로세서는 판독 전용 메모리 또는 랜덤 액세스 메모리 또는 둘 다로부터 명령어들과 데이터를 수신할 것이다. 컴퓨터의 요소들은 명령어들을 실행하는 적어도 하나의 프로세서 및 명령어들과 데이터를 저장하는 하나 이상의 메모리 장치들을 포함할 수 있다. 일반적으로, 컴퓨터는 데이터를 저장하는 하나 이상의 대량 저장 장치들, 예를 들어 자기, 자기-광 디스크들, 또는 광 디스크들을 포함할 수 있거나, 이것들로부터 데이터를 수신하거나 이것들에 데이터를 송신하거나 또는 양쪽으로 되도록 결합될 수도 있다. 컴퓨터 프로그램 명령어들과 데이터를 구체화하는데 적절한 정보 캐리어들은 예로서 반도체 메모리 장치들, 예를 들어, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM(Compact Disk Read Only Memory), DVD(Digital Video Disk)와 같은 광 기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-Optical Media), 롬(ROM, Read Only Memory), 램(RAM, Random Access Memory), 플래시 메모리, EPROM(Erasable Programmable ROM), EEPROM(Electrically Erasable Programmable ROM) 등을 포함한다. 프로세서 및 메모리는 특수 목적 논리 회로조직에 의해 보충되거나, 이에 포함될 수 있다.
- [0159] 또한, 컴퓨터 판독가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 가용매체일 수 있고, 컴퓨터 저장매체를 모두 포함할 수 있다.
- [0160] 본 명세서는 다수의 특정한 구현물의 세부사항들을 포함하지만, 이들은 어떠한 발명이나 청구 가능한 것의 범위에 대해서도 제한적인 것으로서 이해되어서는 안되며, 오히려 특정한 발명의 특정한 실시형태에 특유할 수 있는 특징들에 대한 설명으로서 이해되어야 한다. 개별적인 실시형태의 문맥에서 본 명세서에 기술된 특정한 특징들은 단일 실시형태에서 조합하여 구현될 수도 있다. 반대로, 단일 실시형태의 문맥에서 기술한 다양한 특징들 역시 개별적으로 혹은 어떠한 적절한 하위 조합으로도 복수의 실시형태에서 구현 가능하다. 나아가, 특징들이 특정한 조합으로 동작하고 초기에 그와 같이 청구된 바와 같이 묘사될 수 있지만, 청구된 조합으로부터의 하나 이상의 특징들은 일부 경우에 그 조합으로부터 배제될 수 있으며, 그 청구된 조합은 하위 조합이나 하위 조합의 변형물로 변경될 수 있다.
- [0161] 마찬가지로, 특정한 순서로 도면에서 동작들을 묘사하고 있지만, 이는 바람직한 결과를 얻기 위하여 도시된 그 특정한 순서나 순차적인 순서대로 그러한 동작들을 수행하여야 한다거나 모든 도시된 동작들이 수행되어야 하는 것으로 이해되어서는 안 된다. 특정한 경우, 멀티태스킹과 병렬 프로세싱이 유리할 수 있다. 또한, 상술한 실시형태의 다양한 장치 컴포넌트의 분리는 그러한 분리를 모든 실시형태에서 요구하는 것으로 이해되어서는

안되며, 설명한 프로그램 컴포넌트와 장치들은 일반적으로 단일의 소프트웨어 제품으로 함께 통합되거나 다중 소프트웨어 제품에 패키징 될 수 있다는 점을 이해하여야 한다.

[0162] 한편, 본 명세서와 도면에 개시된 본 발명의 실시 예들은 이해를 돕기 위해 특정 예를 제시한 것에 지나지 않으며, 본 발명의 범위를 한정하고자 하는 것은 아니다. 여기에 개시된 실시 예들 이외에도 본 발명의 기술적 사상에 바탕을 둔 다른 변형 예들이 실시 가능하다는 것은, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 자명한 것이다.

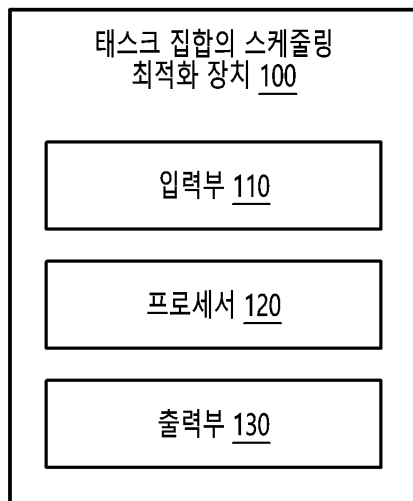
부호의 설명

[0163] 100: 태스크 집합의 스케줄링 최적화 장치

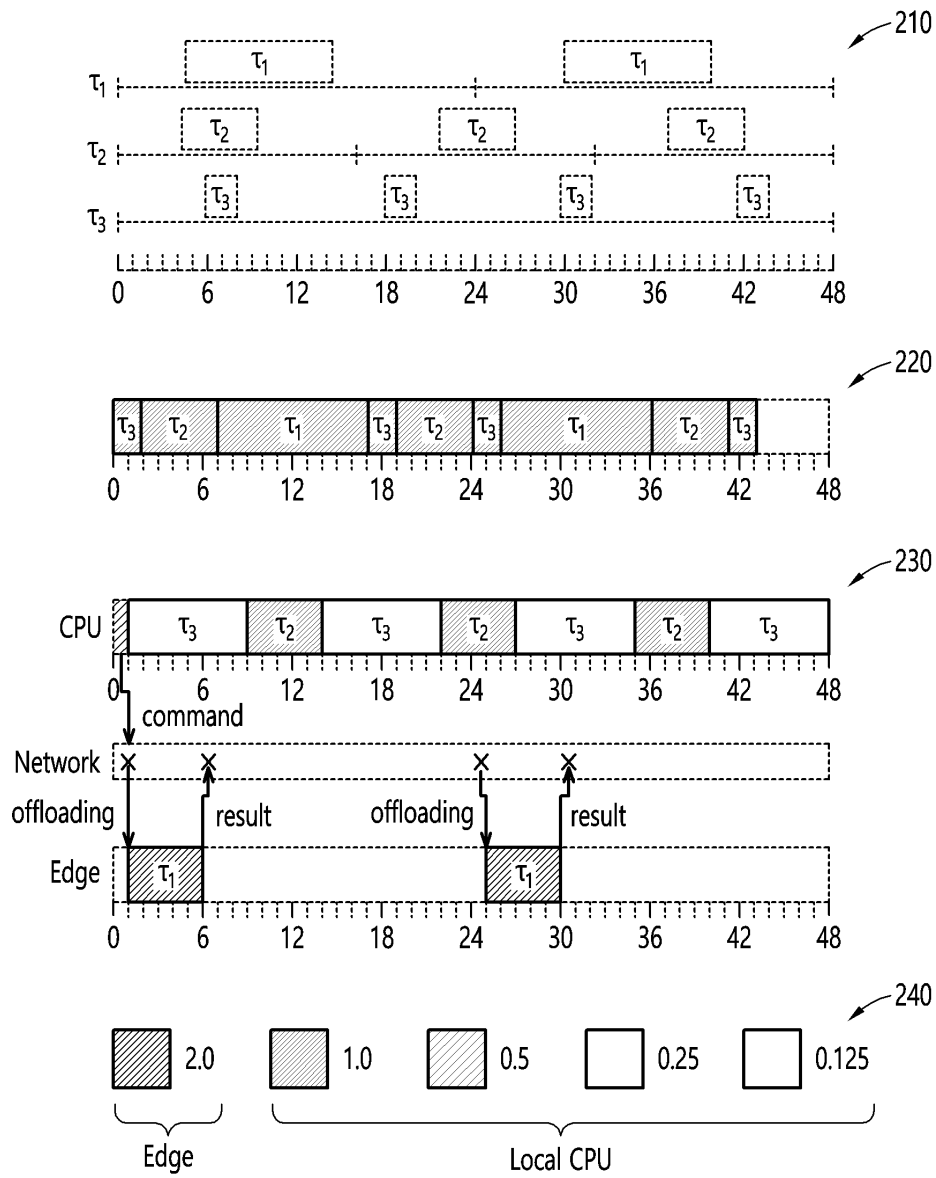
120: 프로세서

도면

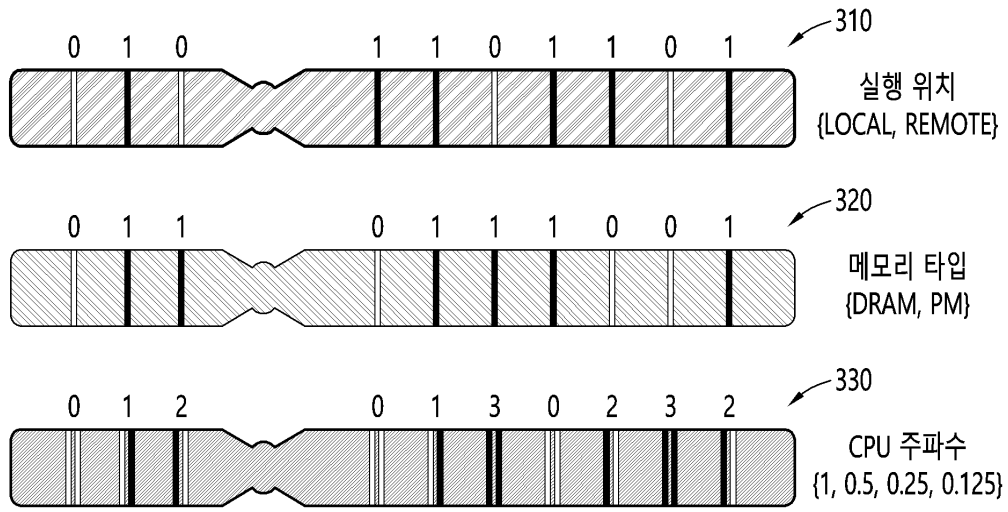
도면1



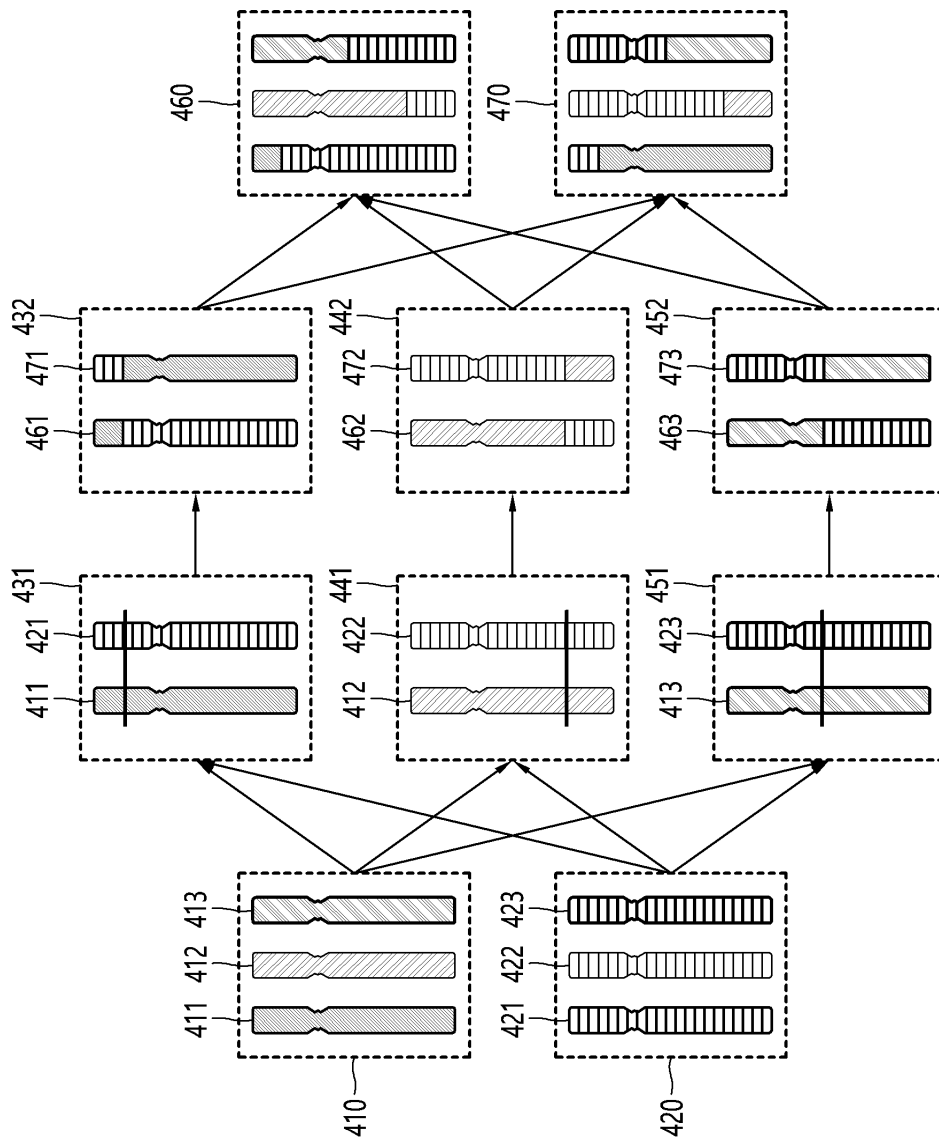
도면2



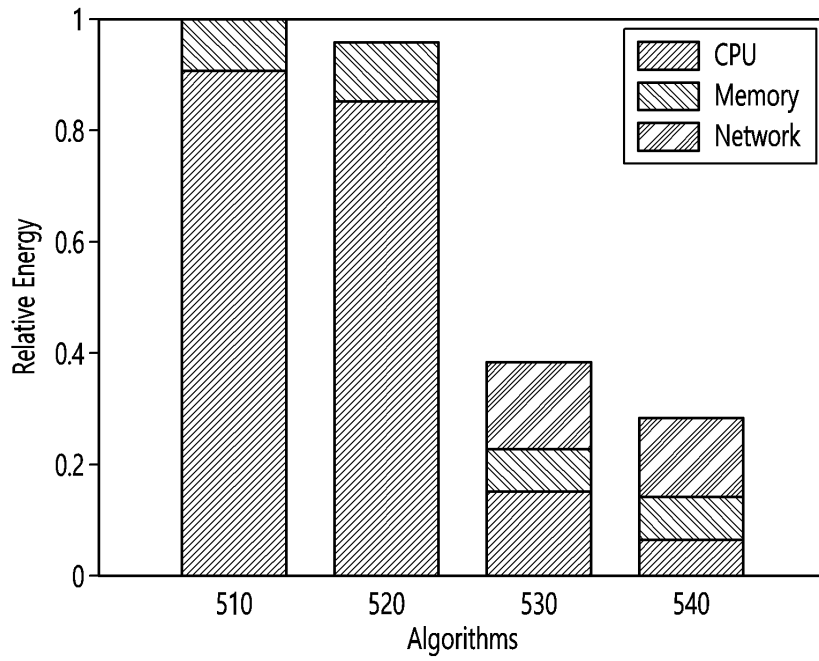
도면3



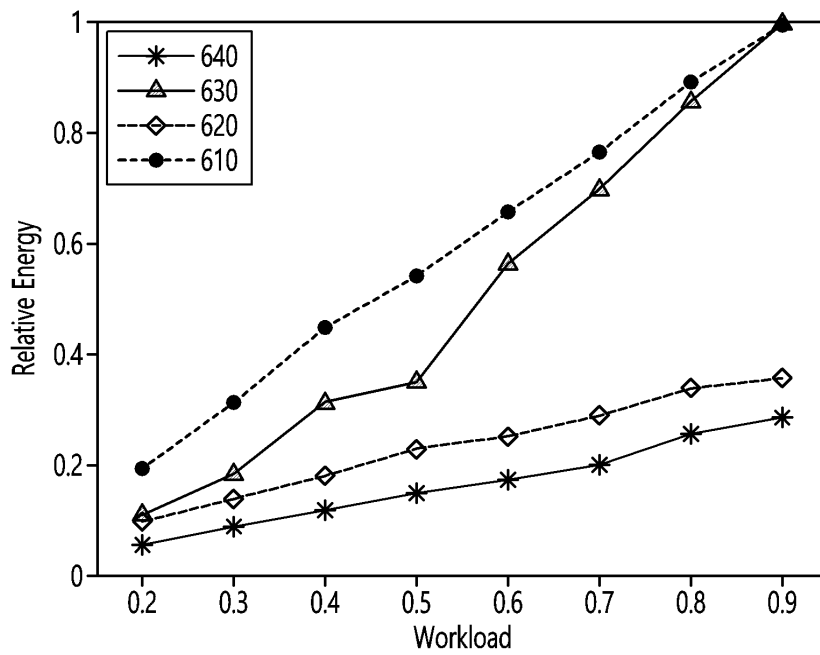
도면4



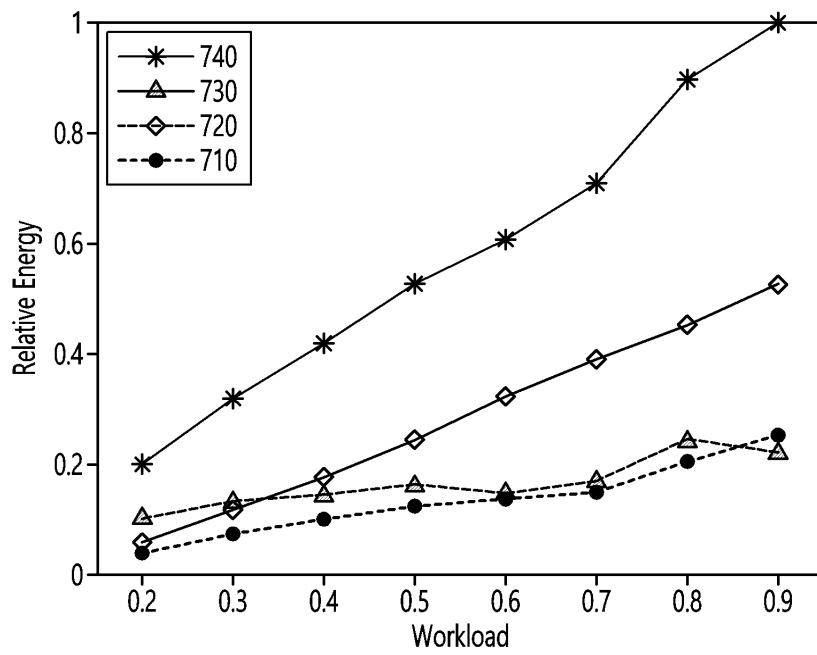
도면5



도면6



도면7



도면8

