(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년12월03일
(11) 등록번호 10-1468357
(24) 등록일자 2014년11월26일

(51) 국제특허분류(Int. Cl.)
G06F 1/32 (2006.01) *G06F 9/50* (2006.01)
(21) 출원번호 10-2014-0017861
(22) 출원일자 2014년02월17일
심사청구일자 2014년02월17일
(56) 선행기술조사문헌
KR1020120013986 A
KR1020050011787 A
KR1019970060745 A
KR1020050012225 A

(73) 특허권자
인하대학교 산학협력단
인천광역시 남구 인하로 100, 인하대학교 (용현동)
(72) 발명자
송민석
서울 서초구 남부순환로 2311-12, 107동 402호 (방배동, 래미안방배아트힐)
이영주
인천 남동구 백범로227번길 69, B동 402호 (만수동, 삼용아파트)
박진한
경기 부천시 원미구 장말로 71, 1504동 101호 (상동, 한아름아파트)
(74) 대리인
양성보

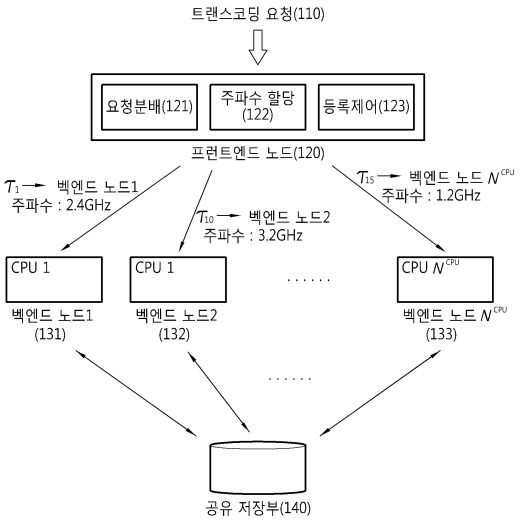
전체 청구항 수 : 총 5 항 심사관 : 김곤희

(54) 발명의 명칭 **트랜스 코딩 서버의 CPU 전력 관리 방법**

(57) 요 약

트랜스 코딩 서버의 CPU 전력 관리 방법이 제시된다. 본 발명에서 제안하는 트랜스 코딩 서버의 CPU 전력 관리 방법은 백엔드 노드의 주파수를 결정하기 위한 파라미터들을 초기화 하는 단계, 미리 정해진 복수의 주파수 리스트 중 가장 낮은 주파수를 예비 주파수로 결정하는 단계, 프런트엔드 노드가 사용자 요청에 따른 트랜스코딩 작업을 상기 백엔드 노드에 할당하는 단계, 상기 선택된 예비 주파수로 상기 트랜스코딩 작업의 수행이 가능한지 확인하는 단계, 상기 트랜스코딩 작업의 수행이 가능할 경우, 상기 예비 주파수를 상기 트랜스코딩 작업을 수행할 때 사용할 최종 주파수로 결정하는 단계를 포함할 수 있다.

대 표 도 - 도1



이 발명을 지원한 국가연구개발사업
과제고유번호 10041971
부처명 미래창조과학부
연구관리전문기관 한국산업기술평가관리원
연구사업명 산업융합원천기술개발사업
연구과제명 상황대응형 분산트랜스코딩 기술을 이용한 저전력 고성능 멀티미디어 콘텐츠관리기술 개발
기 여 율 1/2
주관기관 인하대학교 산학협력단
연구기간 2012.06.01 ~ 2014.05.31
이 발명을 지원한 국가연구개발사업
과제고유번호 10035243
부처명 미래창조과학부
연구관리전문기관 한국산업기술평가관리원
연구사업명 산업원천기술개발사업
연구과제명 CPS(Cyber Physical System)를 위한 콤포넌트 기반 설계이론 및 제어커널 개발
기 여 율 1/2
주관기관 서울대학교 산학협력단
연구기간 2011.03.01 ~ 2015.02.28

특허청구의 범위

청구항 1

트랜스코딩 서버의 CPU 전력 관리 방법에 있어서,

백엔드 노드의 주파수를 결정하기 위한 파라미터들을 초기화 하는 단계;

미리 정해진 복수의 주파수 리스트 중 가장 낮은 주파수를 예비 주파수로 결정하는 단계;

프런트엔드 노드가 사용자 요청에 따른 트랜스코딩 작업을 상기 백엔드 노드에 할당하는 단계;

상기 결정된 예비 주파수로 상기 트랜스코딩 작업의 수행이 가능한지 확인하는 단계; 및

상기 트랜스코딩 작업의 수행이 가능할 경우, 상기 예비 주파수를 상기 트랜스코딩 작업을 수행할 때 사용할 최종 주파수로 결정하는 단계

를 포함하는 트랜스 코딩 서버의 CPU 전력 관리 방법.

청구항 2

제1항에 있어서,

상기 트랜스코딩 작업의 수행이 불가능할 경우, 상기 결정된 예비 주파수가 상기 주파수 리스트 중 가장 높은 주파수인지 확인하는 단계; 및

상기 결정된 예비 주파수가 가장 높은 주파수가 아닐 경우, 상기 결정된 예비 주파수를 증가시키는 단계

를 더 포함하고,

상기 예비 주파수를 증가시키는 단계가 완료된 후, 할당되지 않은 트랜스코딩 작업을 상기 백엔드 노드에 할당하고, 상기 최종 주파수를 결정하는 단계를 반복하는

하는 트랜스 코딩 서버의 CPU 전력 관리 방법.

청구항 3

제1항에 있어서,

상기 미리 정해진 복수의 주파수 리스트는 각각의 주파수에 따른 상기 트랜스코딩 작업의 마감기한 및 최소의 전력 소비를 만족하는 CPU의 주파수를 매칭한 주파수 리스트인

트랜스 코딩 서버의 CPU 전력 관리 방법.

청구항 4

제1항에 있어서,

상기 결정된 예비 주파수로 상기 트랜스코딩 작업의 수행이 가능한지 확인하는 단계는,

CPU에 할당된 모든 작업의 이용률의 합이 상기 결정된 예비 주파수에서의 최대 이용률 보다 작은 경우, 마감기한 내에 상기 트랜스코딩 작업의 수행이 가능한

트랜스 코딩 서버의 CPU 전력 관리 방법.

청구항 5

제1항에 있어서,

상기 프런트엔드 노드는 새로운 트랜스코딩 요청을 받을 때마다 새로운 트랜스코딩 작업을 포함한 모든 트랜스코딩 작업에 필요한 마감기한을 계산하고, 상기 주파수 리스트를 업데이트하는

트랜스 코딩 서버의 CPU 전력 관리 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 멀티미디어 스트리밍(multimedia streaming) 서버에서 트랜스코딩(transcoding) 시 소비되는 에너지를 줄일 수 있는 방법에 관한 것이다.

배 경 기 술

[0002] 최근의 멀티미디어와 네트워크 기술의 발전은 전자 도서관, 주문형 교육(education-on-demand), 원격교육(distance learning) 과 UCC(user-created content) 같은 응용 프로그램들을 주문형 비디오(video-on-demand) 서비스로의 즉각적인 이용을 가능하게 하였다. 이러한 응용 프로그램들의 다양성과 인기는 데이터 센터(data center)의 많은 에너지 소비로 이어졌다. 따라서 전력 소비를 줄이는 것이 비디오 서비스 제공자들에게 굉장히 중요한 문제가 되었다.

[0003] TV 에브리웨어(TV Everywhere)와 N-스크린 플랫폼(N-screen platform)과 같이 다양한 장치에 콘텐츠를 제공할 수 있는 제품들이 상용화됨에 따라 언제 어디서나 무선 비디오 스트리밍(streaming) 서비스를 이용 할 수 있게 되었다. 그러나 이러한 장치의 종류들이 다양해지고 각 장치들이 가지는 해상도, 비트율(bit rate)과 같은 고유 성질이 다르기 때문에, 비디오 콘텐츠를 서비스하기 위해서는 각 장치에 맞는 적절한 형식으로 변환이 필요하다. 이러한 변환 과정을 트랜스 코딩(transcoding)이라 하고 많은 수요를 보이고 있다. 최근 도입된 일반 서버에서 스트리밍 비디오 스트리밍을 가능하게 하는 기술인 DASH(dynamic adaptive streaming over HTTP)에서는 특히 더 필요로 한다. DASH 는 비디오를 부분(segment)로 나누고, 이 부분을 다양한 형식으로 저장하여둔다. 그리고 사용자들은 이용 가능한 대역폭 대에서 알맞은 버전으로 받는다. 이것은 트랜스코딩 과정에 대한 중요한 요구사항을 생성한다.

[0004] 트랜스 코딩은 비트율, 공간 해상도 및 시간 해상도의 감소뿐만 아니라 하나의 비디오를 다른 형식의 비디오로의 변환하는 것도 가능하다. 이런 종류의 작업은 집약적인 계산을 수반하고 그 결과 상당한 전력 소비가 발생하게 된다.

[0005] 서버의 전력 소비를 감소시키는 효과적인 방법은 작업량의 변화에 따라 CPU의 전압 및 주파수를 변화시키는 DVFS(dynamic voltage and frequency scaling)를 사용하는 것이다. CPU에 의해 낭비되는 에너지는 공급 전압에 따라 차이가 있기 때문에, 전압을 줄이는 것은 많은 에너지 소비 감소 효과를 낼 수 있다. 하지만 CPU의 주파수를 낮추는 것은 프로그램의 수행속도가 느려지는 결과를 초래한다. 그러므로 전력 소비에 따라 균형을 맞추기 위해 실행시간이 필요하고, 실행시간을 알 수 있다면 DVFS를 효과적으로 사용 할 수 있다. 트랜스코딩의 경우 작업은 각각의 마감기한이 존재하고, 생성된 시간 요건은 전원 관리 프로세스에 통합되어야 한다.

발명의 내용

해결하려는 과제

[0006] 본 발명이 이루고자 하는 기술적 과제는 트랜스코딩 작업량에 따라 가장 적은 전력을 소비하면서 모든 작업을 마감기한 안에 완료할 수 있는 CPU 주파수를 선택하여, 해당 주파수에 트랜스코딩 작업을 할당하는 새로운 DVFS 방법을 제공하는데 있다.

과제의 해결 수단

[0007] 일 측면에 있어서, 본 발명에서 제안하는 트랜스 코딩 서버의 CPU 전력 관리 방법은 백엔드 노드의 주파수를 결정하기 위한 파라미터들을 초기화 하는 단계, 미리 정해진 복수의 주파수 리스트 중 가장 낮은 주파수를 예비 주파수로 결정하는 단계, 프런트엔드 노드가 사용자 요청에 따른 트랜스코딩 작업을 상기 백엔드 노드에 할당하는 단계, 상기 선택된 예비 주파수로 상기 트랜스코딩 작업의 수행이 가능한지 확인하는 단계, 상기 트랜스코딩 작업의 수행이 가능할 경우, 상기 예비 주파수를 상기 트랜스코딩 작업을 수행할 때 사용할 최종 주파수로 결정하는 단계를 포함할 수 있다.

[0008] 상기 트랜스코딩 작업의 수행이 불가능할 경우, 상기 선택된 예비 주파수가 상기 주파수 리스트 중 가장 높은 주파수인지 확인하는 단계, 상기 선택된 예비 주파수가 가장 높은 주파수가 아닐 경우, 상기 선택된 예비 주파수를 증가시키는 단계를 더 포함하고, 상기 예비 주파수를 증가시키는 단계가 완료된 후, 할당되지 않은 트랜스

코딩 작업을 상기 백엔드 노드에 할당하고, 상기 최종 주파수를 결정하는 단계를 반복할 수 있다.

[0009] 상기 미리 정해진 복수의 주파수 리스트는 각각의 주파수에 따른 상기 트랜스코딩 작업의 마감기한 및 CPU 최적 이용률을 매칭한 주파수 리스트이다.

[0010] 상기 선택된 예비 주파수로 상기 트랜스코딩 작업의 수행이 가능한지 확인하는 단계는 CPU의 이용률의 합이 상기 선택된 예비 주파수에 따른 최대 이용률보다 클 경우, 상기 트랜스코딩 작업의 수행이 가능할 수 있다.

[0011] 상기 프런트엔드 노드는 새로운 트랜스코딩 요청을 받을 때마다 새로운 트랜스코딩 작업을 포함한 모든 트랜스코딩 작업에 필요한 마감기한을 계산하고, 상기 주파수 리스트를 업데이트할 수 있다.

발명의 효과

[0012] 본 발명의 실시예들에 따르면 트랜스 코딩 서버의 CPU 전력 관리 방법은CPU가 전력 최소한의 전력을 소비하면서 트랜스코딩 작업을 마감기한 내에 완료할 수 있는 주파수를 결정하고 각 CPU에 작업을 최적화하여 할당할 수 있다. 또한 작업 이동이 안 되는 시스템에 대한 알고리즘을 제안함으로써, 실제 트랜스코딩 작업의 전력소비 측정을 바탕으로 하는 시뮬레이션(simulation)결과에서 기존 리눅스의 DVFS 기법과 비교하여 10% ~ 16%의 전력 소비 감소 효과를 확인하였다.

도면의 간단한 설명

[0013] 도 1은 본 발명의 일 실시예에 따른 트랜스 코딩 서버의 CPU 전력 관리의 전체 과정을 설명하기 위한 도면이다.

도 2는 본 발명의 일 실시예에 따른 트랜스 코딩 서버의 CPU 전력 관리 방법을 설명하기 위한 흐름도이다.

도 3 및 도 4는 본 발명의 일 실시예에 따른 멀티미디어 서버와 기존의 멀티미디어 서버의 에너지 소비를 비교한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0014] 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0015] 도 1은 본 발명의 일 실시예에 따른 트랜스 코딩 서버의 CPU 전력 관리의 전체 과정을 설명하기 위한 도면이다.

[0016] 트랜스코딩 서버는 도 1과 같이 복수의 노드(node)로 구성되어 있다. 예를 들어, 복수의 노드는 프런트엔드(front-end)(120) 및 백엔드(back-end) 노드들(131, 132, 133)을 포함할 수 있다. 프런트엔드 노드(120)는 사용자로부터 트랜스코딩 요청(110)을 받아 백엔드 노드들(131, 132, ..., 133)로 할당할 수 있다. 트랜스코딩 요청(110)에는 요청분배(121), 주파수 할당(122), 등록 제어(123) 등을 포함할 수 있다. 백엔드 노드들(131, 132, ..., 133)은 요청 받은 트랜스코딩 요청(110)을 처리할 수 있다. 각 백엔드 노드들(131, 132, ..., 133)은 독립적인 주파수와 전압에서 동작하는 DVFS가 가능한 CPU를 사용할 수 있다. 예를 들어, 백엔드 노드 1(131)은 주파수 2.4GHz에서 동작하고, CPU 1을 사용하고, 백엔드 노드 2(132)은 주파수 3.2GHz에서 동작하고, CPU 2을 사용하고, 백엔드 노드 N^{CPU}(133)은 주파수 1.2GHz에서 동작하고, CPU N^{CPU}를 사용할 수 있다. 그리고 프런트엔드 노드(120)가 백엔드 노드들로 트랜스코딩 작업을 할당하면 백엔드 노드에서는 해당 작업을 수행할 때 사용할 주파수를 결정할 수 있다. 이때 주파수는 트랜스코딩 작업량에 따라 가장 적은 전력을 소비하면서 모든 작업을 마감기한 안에 완료할 수 있는 CPU 주파수를 선택할 수 있다. 이러한 방법으로 선택된 주파수를 이용하여 벤엔드 노드들은 공유 저장부(140)를 통해 요청된 트랜스코딩 작업을 수행할 수 있다.

[0017] 도 2는 본 발명의 일 실시예에 따른 트랜스 코딩 서버의 CPU 전력 관리 방법을 설명하기 위한 흐름도이다.

[0018] 트랜스 코딩 서버의 CPU 전력 관리 방법은 백엔드 노드의 주파수를 결정하기 위한 파라미터들을 초기화 하는 단계(210), 미리 정해진 복수의 주파수 리스트 중 가장 낮은 주파수를 예비 주파수로 결정하는 단계(220), 프런트엔드 노드가 사용자 요청에 따른 트랜스코딩 작업을 상기 백엔드 노드에 할당하는 단계(230), 상기 선택된 예비 주파수로 상기 트랜스코딩 작업의 수행이 가능한지 확인하는 단계(240), 상기 트랜스코딩 작업의 수행이 가능할 경우, 상기 예비 주파수를 상기 트랜스코딩 작업을 수행할 때 사용할 최종 주파수로 결정하는 단계(250), 트랜스코딩 작업의 수행이 불가능할 경우, 상기 선택된 예비 주파수가 상기 주파수 리스트 중 가장 높은 주파수인지

확인하는 단계(260), 선택된 예비 주파수가 가장 높은 주파수가 아닐 경우, 상기 선택된 예비 주파수를 증가시키는 단계(270), 할당되지 않은 트랜스코딩 작업을 상기 백엔드 노드에 할당하고, 상기 최종 주파수를 결정하는 단계를 반복하는 단계(280)를 포함할 수 있다. 트랜스 코딩 서버의 CPU 전력 관리 방법에 대하여 예를 들어 더욱 상세히 설명한다.

[0019] 단계(210)에서, 백엔드 노드의 주파수를 결정하기 위한 파라미터들을 초기화 할 수 있다. 새로운 작업 요청이 왔을 경우 새 트랜스코딩 작업에 대해 $x_{i,j}$ 값을 0으로 초기화 할 수 있다. $x_{i,j}$ 는 작업 τ_i 가 CPU_j 에 할당된 것을 나타내고, CPU_j 할당된 주파수를 F_j 로 나타낸다고 가정할 때, 단계(210)에서, $x_{i,j}$ 및 F_j 백엔드 노드의 주파수를 결정하기 위한 파라미터들을 초기화 할 수 있다.

[0020] 단계(220)에서, 미리 정해진 복수의 주파수 리스트 중 가장 낮은 주파수를 예비 주파수로 결정할 수 있다. 미리 정해진 복수의 주파수 리스트는 각각의 주파수에 따른 트랜스코딩 작업의 마감기한 및 CPU 최적 이용률을 매칭한 주파수 리스트를 나타낼 수 있다. 이러한 복수의 주파수 리스트 중 가장 낮은 주파수를 예비 주파수로 결정할 수 있다. 여기에서, 예비 주파수란 요청된 트랜스코딩 작업을 마감시간 내에 수행하고, CPU 전력도 최소로 할 수 있는 주파수 값을 찾기 위한 과정에서 최초로 설정되는 주파수를 의미한다. 예를 들어, 각 CPU는 여러 독립된 주파수 단계(level) 에서 실행 가능할 수 있다. CPU의 종류가 서로 다르다면 CPU가 가지는 주파수 단계 또한 다를 수 있다. $f_j(k)$를 CPU_j 의 단계의 주파수라고 가정하면, 제안하는 방법에서는 전체 백엔드 노드에서 사용하는 모든 CPU의 주파수 중 가장 높은 주파수를 기준으로 하고 이를 f^{base}로 나타낼 수 있다. f^{base}를 수학식1과 같이 나타낼 수 있다.

$$f^{base} = \max_{j=1,\ldots,N^{CPU}} f_j\left(N_j^{freq}\right)$$

[0021] 수학식1

[0022] 프런트엔드 노드는 새로운 트랜스코딩 요청을 받을 때마다 새로운 트랜스 코딩 작업을 포함한 모든 트랜스코딩 작업에 필요한 계산 시간을 추산할 수 있다. 각 작업 $\tau_i(i = 1, \ldots, N^{CPU})$은 (D_i, C_i)와 연관될 수 있다. D_i는 절대적인 마감기한과 현재 시간과의 차이인 상대적인 마감기한이고, C_i는 f^{base}에서 τ_i 을 계산하는데 필요한 시간이다. 이 때 작업 τ_i 의 이용률(utilization factor) U_i는 수학식2와 같이 나타낼 수 있다.

$$U_i = \frac{C_i}{D_i}$$

[0023] 수학식2

[0024] 제안하는 방법의 목적은 서버의 총 에너지 소비를 줄이는 것이고, 측정을 통해 백엔드 노드에서 요구하는 CPU 주파수와 에너지의 실제 관계를 얻을 수 있다. 예를 들어, 표 1은 인텔 코어 I5-3470 프로세서(processor)의 가장 낮은 주파수에서 높은 주파수까지 각 주파수 당 트랜스코딩 작업을 수행하는데 소비되는 에너지를 측정하고, 가장 높은 주파수를 기준으로 정규화 한 것이다. 표 1을 바탕으로 CPU 주파수와 에너지 소비의 관계가 단순하다고 가정할 수 있다.

[0025] <표1>

주파수 (GHz)	3.2	3.1	3.0	2.9	2.7	2.6	2.5	2.4	2.3	2.2	2.1	1.9	1.8	1.7	1.6
정규화 에너지	1	0.989	0.968	0.956	0.933	0.926	0.918	0.909	0.898	0.894	0.891	0.872	0.872	0.869	0.866

[0026]

[0027] 단계(230)에서, 프런트엔드 노드가 사용자 요청에 따른 트랜스코딩 작업을 상기 백엔드 노드에 할당할 수 있다. 그리고 단계(240)에서, 선택된 예비 주파수로 상기 트랜스코딩 작업의 수행이 가능한지 확인할 수 있다. 트랜스

코딩 작업의 수행이 가능할 경우, 단계(250)에서 예비 주파수를 상기 트랜스코딩 작업을 수행할 때 사용할 최종 주파수로 결정할 수 있다. 반면에, 트랜스코딩 작업의 수행이 불가능할 경우, 단계(260)에서 선택된 예비 주파수가 상기 주파수 리스트 중 가장 높은 주파수인지 확인할 수 있다. 실시예에 따르면, CPU의 이용률의 합이 상기 선택된 예비 주파수에 따른 최대 이용률보다 클 경우, 상기 트랜스코딩 작업의 수행이 가능하다고 판단할 수 있다. 그리고 단계(270)에서, 선택된 예비 주파수가 가장 높은 주파수가 아닐 경우, 상기 선택된 예비 주파수를 증가시키고, 단계(280)에서, 할당되지 않은 트랜스코딩 작업을 상기 백엔드 노드에 할당하고, 상기 최종 주파수를 결정하는 단계를 반복할 수 있다. 또한, 프런트엔드 노드는 새로운 트랜스코딩 요청을 받을 때마다 새로운 트랜스코딩 작업을 포함한 모든 트랜스코딩 작업에 필요한 마감기한을 계산하고, 주파수 리스트를 업데이트할 수 있다.

[0028] 예를 들어, 제안하는 방법은 높은 우선순위를 가진 작업에 마감기한을 부여하는 EDF 스케줄링(EDF scheduling)을 사용할 수 있다. EDF 스케줄링은 각 작업이 마감기한을 만족하면서 CPU 자원을 효과적으로 사용할 수 있도록 한다. CPU_j 에서 수행되는 모든 트랜스코딩 작업이 마감기한 안에 끝나기 위해서는 CPU의 이용률의 합이 CPU_j 의 k단계 주파수에서의 최대 이용률(utilization bound) $U_j^{bound}(k)$보다 작거나 같아야 한다. $U_j^{bound}(k)$를 수학식3과 같이 나타낼 수 있다.

$$U_j^{\text{bound}}(k) = \frac{f_j(k)}{f^{\text{base}}}$$

[0029] 수학식3

[0030] CPU_j 에 할당된 모든 작업의 이용률 합이 $U_j^{bound}(k)$보다 작거나 같다면 모든 작업은 마감기한 안에 트랜스코딩 될 수 있고, 트랜스코딩 스케줄(schedule)이 적합한 것이다. 수학식 3을 통해 주파수 단계가 오를수록 최대 이용률이 증가한 다는 것을 알 수 있다. 최대 이용률이 증가할수록 마감기한 내에 완료할 수 있는 트랜스코딩 작업이 많아지지만, 표 1을 통해 알 수 있듯이 최대 이용률과 에너지 소비는 트레이드오프(trade-off)의 관계이다. 작업 이동의 허용 여부도 고려해야 할 부분이다. 작업 이동이 가능 할 경우 트랜스코딩 작업을 CPU에 할당하는 것에 영향을 주기 때문이다. 제안하는 방법에서는 작업 이동이 가능한 알고리즘을 개발하고, 작업 이동이 허락되지 않은 상황에 맞춰 수정할 수 있다.

[0031] 또한, 제안하는 방법에서는 $x_{i,j}$ 라는 이진 변수를 도입할 수 있다. 예를 들어, $x_{i,j}$ = 1은 작업 τ_i 가 CPU_j 에 할당되었다는 것을 의미할 수 있다. CPU 에서 선택한 주파수 단계를 F_j(F_j = 1, ..., N_j^{freq})라고 가정한다. 새로운 트랜스코딩 요청이 들어왔을 때, CPU_j 의 이용률의 합이 $U_j^{bound}(k)$보다 작거나 같은지 검사하기 위해 프런트엔드 노드에서 모든 트랜스코딩 요청의 우선순위를 계산할 수 있다. 이 작업은 모든 트랜스코딩 작업이 마감기한 안에 끝나는지 여부를 알기 위해 필요하다. 그러나 제안하는 방법의 목적은 전력 소모를 최소화 하는 것이고, 따라서 각 CPU당 가장 낮은 주파수 단계를 선택해야 한다. 백엔드 노드에서 사용하는 CPU_j 의 주파수 단계 k에서의 전력 소모량을 $P_j(k)$라고 하자. F_j 와 $x_{i,j}$ 를 결정하는 방법은 수학식4를 이용할 수 있다.

$$Minimize \sum_{i=1}^{N^{task}} \sum_{j=1}^{N^{CPU}} x_{i,j} P_j(F_j)$$

$$s.t. \quad F_j = \arg\min_{k=1,\ldots,N_j^{freq}} \sum_{i=1}^{N^{task}} x_{i,j} U_i \le U_j^{bound}(k) \ , \ x_{i,j} \in 0,1.$$

$$\sum_{j=1}^{N^{CPU}} x_{i,j} = 1$$

[0032] 수학식4

[0033] 각 CPU에 대한 주파수 단계 F_j와 각 작업 τ_i 와 각 CPU에 대한 2진 변수 $x_{i,j}$ 를 결정해야 한다. 다음 세 단계의 알고리즘을 통해 F_j와 $x_{i,j}$ 를 결정한다. 첫째, 각 CPU에 대한 예비 주파수 F_j를 결정 하는 단계. 둘째, 각 작업

τ_i 와 CPU 에 대한 $x_{i,j}$ 를 결정하는 작업 할당 단계. 마지막으로, 마감기한을 만족하지 못하는 CPU에 대해 마감기한을 만족 하는 주파수 선택하기 위해 주파수 단계적 증가(escalation) 단계이다. 주파수 결정 단계의 첫 번째는 총 이용률 U^{demand}를 구하는 식은 수학식5와 같이 나타낼 수 있다.

$$U^{\mathrm{demand}} = \sum_{i=1}^{N^{\mathrm{task}}} \frac{C_i}{D_i}$$

[0034] 수학식5

[0035] 모든 CPU의 최대 이용률 합은 U^{demand}를 넘어야 한다. 그렇지 않으면 마감기한 내에 작업을 끝낼 수 없다. 그러나 F_j는 전력 소모를 최소로 하는 주파수 여야 한다. 이러한 F_j 와 CPU_j 를 결정하는 식은 수학식6을 이용할 수 있다.

$$\begin{aligned} &Minimize && \sum_{j=1}^{N^{\mathrm{CPU}}} P_j(F_j) \\ &s.t. && \sum_{j=1}^{N^{\mathrm{CPU}}} U_j^{\mathrm{bound}}(F_j) \geq U^{\mathrm{demand}},\ F_j = 1,\ldots,f_{N_j^{\mathrm{freq}}} \end{aligned}$$

[0036] 수학식6

[0037] 수학식6은 이익을 극대화를 위해 가중치에 따라 여러 클래스(class)의 각 항목을 하나 선택할 수 있는 다중 선택 배낭(multiple-choice knapsack)문제를 변형한 것이다. 주파수 결정 단계는 각 CPU당 $\sum_{j=1}^{N^{\mathrm{CPU}}} U_j^{\mathrm{bound}}(F_j) \geq U^{\mathrm{demand}}$ 을 만족하면서 전력 절감 효과가 최대인 하나의 주파수를 선택한다. 그리디 알고리즘(greedy algorithm)이 다중 선택 배낭문제에서 좋은 성능을 보여주기 때문에, 제안하는 방법에서는 최대 이용률의 비에 기반을 둔 그리디 알고리즘을 제안한다.

[0038] 제안하는 방법에서는 $r_j(k)$, ($\forall$j, k= 2, ..., N_j^{freq}) 를 CPU_j 의 1단계 주파수에 해당하는 비율로 정규화한 Pj(k)와 $U_j^{bound}(k)$의 비율로 정의 하였고, 식으로 표현하면 수학식7과 같이 나타낼 수 있다.

$$r_j(k) = \frac{P_j(k) - P_j(1)}{U_j^{\mathrm{bound}}(k) - U_j^{\mathrm{bound}}(1)}$$

[0039] 수학식7

[0040] 예를 들어, 초기화 시 모든 값 리스트(list) L을 만 수 있다. 그리고 모든 F_j를 가장 낮은 $U_j^{bound}(k)$값을 갖지만 가장 낮은 전력을 소비하는 주파수 단계인 1단계로 설정할 수 있다. 주파수 증가가 필요할 때는 리스트 의 최저값을 고를 수 있다. 이것은 전력 소비에 있어서 이용률 증가하는 동안 최소한의 증가를 요청하기 때문이다. 따라서 알고리즘은 CPU 색인 c와 $r_c(1)$이 최저인 주파수 단계 l로 구성된 한 쌍을 찾고, 리스트 L에서 찾은 $r_c(1)$ 값을 지울 수 있다. 현재 주파수 단계인 F_c보다 높은 주파수 단계 l을 선택했다면, F_c는 l로 증가할 수 있다. 이 과정은 $\sum_{j=1}^{N^{\mathrm{CPU}}} U_j^{\mathrm{bound}} \geq U^{\mathrm{demand}}$ 를 만족할 때까지 반복할 수 있다. 리스트 L은 이 과정이 끝난 후에도 유지되고, 주파수 단계적 증가 단계에서 사용될 수 있다.

[0041] 작업 할당 단계에서는 각 작업을 적절한 CPU에 할당할 수 있다. CPU_j 의 총 이용률은 최대 이용률 $U_j^{bound}(k)$을 넘지 않아야 한다. 이 문제는 상자 채우기(bin packing) 문제의 변형한 것이다. 변수 $x_{i,j}$ 를 결정하기 위해

worst-fit 경험적 해결책을 변형하여 사용할 수 있다. 이러한 해결책은 모든 프로세서의 작업량 균형을 맞추며 실행될 수 있다. 모든 작업을 할당하기 위해서, $x_{i,j}$ = 0인 작업의 색인 배열 A를 유지해야 한다. 그 후, CPU_j의 남은 이용률을 계산할 수 있다. 그리고 가장 많은 이용률이 남은 CPU를 첫 번째로 지정할 수 있다.

[0042] 모든 작업이 $\sum_{i=1}^{N^{task}} x_{i,j} U_i \leq U_j^{bound}$ 조건을 만족하면서 CPU에 할당되는지를 나타내기 위해 불 방식(boolean) 매개변수 feasible을 사용할 수 있다.

[0043] 만약 작업 할당 단계 후 매개변수 feasible이 거짓(FALSE) 이라면, 조건을 만족하지 않은 CPU의 주파수를 증가시켜서 U^{bound}를 증가 시켜야 한다. 주파수 결정 단계에서 만들어진 $r_{j,k}$ 의 리스트 L을 이용한다. 주파수 결정 단계와 비슷하게 $l>F_c$를 만족하는 CPU 색인 c와 $r_c(l)$의 가장 낮은 주파수 단계 l이 선택될 수 있다.

[0044] 주파수 단계적 증가 단계가 완료된 후, 이전 작업 할당 단계에서 할당되지 않은 작업에 대해 작업 할당 단계가 반복될 수 있다. 작업 할당 단계에서 가능한 해결책을 만들 때까지 이 두 과정이 반복될 수 있다. 위의 알고리즘은 작업 이동이 가능한 경우의 알고리즘으로 본 발명에서 이것을 FTA(Frequency and task allocation)라고 명한다. 작업 이동이 불가능한 경우, 기존 작업의 $x_{i,j}$ 값을 변경하는 것이 불가능하다. 따라서 알고리즘의 수정이 필요하고, 제안하는 방법에서 수정된 알고리즘을 FTA-no라고 명한다. 이 알고리즘은 트랜스코딩 작업이 요청되거나 종료될 때 기존의 알고리즘과 다르게 수행될 수 있다.

[0045] 새로운 작업 요청이 왔을 경우, 기존 모든 작업에 대한 는 모두 사용 가능한 것이므로, 새 트랜스코딩 작업에 대해 $x_{i,j}$ 값을 0으로 초기화 할 수 있다. 주파수 결정단계를 실행하는 대신에 F_j 의 값을 $\sum_{\forall i,j\ x_{i,j}=1} U_i \leq U_j^{bound}(F_j)$ 의 조건을 만족하는 가장 낮은 주파수로 초기화 한다. 다른 단계는 이전 알고리즘과 같다.

[0046] 트랜스 코딩 작업이 끝났을 경우, 기존 모든 작업에 대한 $x_{i,j}$ 는 모두 사용 가능한 것이므로, 각 CPU의 주파수 단계가 다시 선택될 필요가 있다. 그리고, F_j의 값을 $\sum_{\forall i,j\ x_{i,j}=1} U_i \leq U_j^{bound}(F_j)$ 조건을 만족하는 최저 주파수 단계로 설정할 수 있다. $x_{i,j}$ 와 F_j를 결정한 후, 각 CPU에 할당된 작업들에 대해 마감기한이 이른 순서대로 높은 우선순위를 다시 부여할 수 있다.

[0047] 기존의 트랜스 코딩 작업의 마감 시간 요구 사항을 충족하기 위해, 새로운 트랜스 코딩 요청의 입장을 제어 할 필요가 있다. 따라서 새로운 트랜스코딩 요청이 도착할 때마다, 요청 받은 트랜스코딩이 입장된다는 가정을 하고 알고리즘이 실행된다. 만약 작업 할당 단계의 결과가 가장 높은 주파수에서도 feasible변수가 거짓(FALSE)이면, 그 요청은 받아들여지지 않을 수 있다.

[0048] 도 3 및 도 4는 본 발명의 일 실시예에 따른 멀티미디어 서버와 기존의 멀티미디어 서버의 에너지 소비를 비교한 도면이다.

[0049] 제안하는 방법을 이용하여 네 종류의 CPU를 사용하는 백엔드 노드로 구성된 트랜스코딩 서버의 시뮬레이션을 통해 트랜스 코딩 서버의 CPU 전력 관리 방법의 효율성을 측정하였다. VLC 트랜스코딩 프로그램을 통해 트랜스코딩을 하는 동안 각 CPU의 시스템 전체의 전력 소비를 측정했다. 또한 3.2GHz의 CPU 주파수에서 같은 프로그램으로 트랜스코딩 작업 시 걸리는 시간을 측정하였다. 트랜스코딩 작업의 정확한 시간을 얻기 위해 백그라운드(background) 작업보다 작업에 가장 높은 우선순위를 부여하는 SCHED_FIFO 클래스로 설정되었다. 표 2은 다른 종류의 트랜스코딩 작업은 각기 다른 수행시간이 걸리는 것을 보여준다.

[0050] <표 2>

[0051]

드라마		애니메이션	
트랜스코딩 종류	소요 시간(초)	트랜스코딩 종류	소요 시간(초)
FULL HD -> HD	202	wmv -> h.264	120
FULL HD -> VGA	103	HD -> VGA	62
4163kbps -> 3000kbps	286	3303kbps -> 3000kbps	119
4163kbps -> 2000kbps	249	3303kbps -> 2000kbps	54
4163kbps -> 1000kbps	205	3303kbps -> 1000kbps	86

[0052] 측정의 결과를 바탕으로 트랜스코딩작업의 시간을 30초에서 300초 사이 임의의 값으로 정해지도록 하였다. 각 작업 τ_i 의 이용률 U_i 또한 0.02에서 0.12의 임의의 값을 가지도록 하고, 마감기한은 식 2를 통해 계산되었다.

[0053] 트랜스코딩 요청은 α 값을 평균 요청 시간으로 하는 포아송 과정을 따른다. 총 이용률이 에너지 소비에 미치는 영향을 실험하기 위해 도착률은 $\frac{U^{\text{demand}}}{N^{\text{CPU}}}$ 값에 따라 조정되었다. T^{low} 와 T^{high}를 조정 임계값이라고 할 때 $\frac{U^{\text{demand}}}{N^{\text{CPU}}} < T^{\text{low}}$ 일 경우, 요청을 증가시키기 위해 α 값을 50초로 설정할 수 있다. $\frac{U^{\text{demand}}}{N^{\text{CPU}}} > T^{\text{low}}$ 일 경우, 요청을 감소시키기 위해 α 값을 150초로 설정할 수 있다. 그 외의 경우에는 α 값을 100초로 설정한다.

[0054] 본 발명의 기법과 다음 세 가지 방식의 기법을 비교하였다.

[0055] <표 3>

[0056]

CPU 종류	인텔 i7 3770	인텔 i5 3470	인텔 i7 2660	AMD phenom 9550
주파수 범위 (GHz)	1.6 - 3.4	1.6 - 3.2	1.6 - 3.4	1.1 - 2.2
주판수 단계 수	15	15	10	2
트랜스코딩 시 전력 소비 범위 (Watt)	60.8 - 93.6	51.4 - 78.4	83 - 136.4	108 - 154

[0057] 1. 트랜스코딩 시 항상 가장 높은 주파수를 선택하고, 트랜스코딩 요청은 라운드 로빈(round-robin)방식으로 스케줄링되는 백엔드 노드.

[0058] 2. 리눅스의 기본 DVFS 기법인 온디맨드(ondemand)를 사용하는 백엔드 노드. 온디맨드 정책은 CPU의 이용률을 측정하고 이용률에 맞춰 CPU의 주파수를 변경한다. 만약 CPU의 이용률이 30초간 0.8을 넘으면, 주파수를 최대로 올린다. 반대로 CPU 이용률이 0.4보다 낮을 경우 주파수를 1단계 내린다.

[0059] 3. 변형 온디맨드 기법. 변형 온디맨드 기법은 온디맨드 정책과 비슷하지만 다른 점은 CPU이용률 이 0.8을 넘을 경우 주파수를 1단계만 올린다.

[0060] U^{avg}시뮬레이션 하는 동안 각 CPU의 이용률의 평균이라고 하자. 에너지 소비가 U^{avg}에 어떤 영향을 받는지 조사하기 위해 T^{low}와 T^{high}를 변형하였다. 도 3은 표 3 에서 제시한 4종류의 CPU를 사용한 32개의 백엔드 노드로 구성된 시스템에 대한 T^{low}, T^{high}, T^{avg}의 상대적인 값을 보여준다. 실험 결과 다음과 같은 결과를 얻을 수 있었다.

[0061] FTA는 변형 온디맨드 보다10% ~ 16%, non-DVFS 보다 11% ~ 17%만큼의 적은 에너지를 사용하면서 최적의 성능을

나타낸다. FTA 와 FTA-no의 에너지 소비의 차이는 무시할 수 있다. 하지만 작업 이동에 걸리는 오버헤드(overhead)는 고려되지 않았기 때문에 FTA-no의 결과가 FTA의 결과 보다 더 정확하다. 제안하는 방법은 U^{avg}가 50%에서 60% 사이일 때 에너지 절약 효과가 더 뚜렷이 나타난다. 온디맨드 정책은 대부분 가장 높은 주파수를 사용하기 때문에 온디맨드 정책이 non-DVFS보다 더 좋은 성능을 보여준다.

[0062] 도 4와 같이 T^{low} = 0.4, T^{high} = 0.7인 상황에서 에너지 소비가 CPU의 수에 따라 어떻게 변하는지 실험하였다. 실험을 통해 다음과 같은 결과를 얻을 수 있었다.

[0063] <표 4>

[0064]

CPU 수	20	24	28	32	36	40
Non-DVFS	0.2071	0.1851	031731	0.1671	031541	0.1118
온디맨드	0.2073	0.1856	0.1737	0.1678	0.1549	0.1126

[0065] FTA는 변형 온디맨드 기법보다 11%~15%, non-DVFS 보다 14%~16% 적은 에너지를 사용하면서 최적의 성능을 나타낸다. CPU의 수가 많아질수록, 트랜스코딩의 부하가 줄고, FTA와 변형 온디맨드의 차이가 감소되었다. T^{low} = 0.75, T^{high} = 0.85 일 때 CPU 수에 따라 마감기한을 지키지 못하는 비율이 어떻게 변하는지 알아보았고, 표 4와 같은 결과를 얻을 수 있었다. 제안하는 방법은 마감기한을 모두 지켰지만, 다른 세 기법은 11%~21%의 확률로 마감기한을 지키지 못하였다. non-DVFS기법이 온디맨드 기법보다 조금 더 좋은 성능을 보였고 온디맨드와 변형 온디맨드는 같은 결과를 가진다. CPU의 수가 증가할수록 마감기한을 지키지 못하는 비율은 줄어들었다.

[0066] 본 발명에서는 비디오 트랜스코딩 서버에서의 전력 소비를 줄이기 위한 새로운 DVFS 기법을 제안했다. 트랜스코딩의 작업량의 특징을 조사했고, CPU 주파수가 시스템의 전력 소비에 어떻게 영향을 미치는지를 파악했다. 그리고 각 CPU가 전력 최소한의 전력을 소비하면서 트랜스코딩 작업을 마감기한 내에 완료할 수 있는 주파수를 결정하고 각 CPU에 작업을 할당하는 최적화 문제를 만들었다. 해당 문제에 대한 세 단계의 알고리즘을 제안했다. 또한 작업 이동이 안 되는 시스템에 대해 알고리즘을 하였다. 실제 트랜스코딩 작업의 전력소비 측정을 바탕으로 하는 시뮬레이션(simulation)결과는 기존 리눅스의 DVFS 기법과 비교하여 10% ~ 16%의 전력 소비 감소 효과를 보여주었다.

[0067] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0068] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

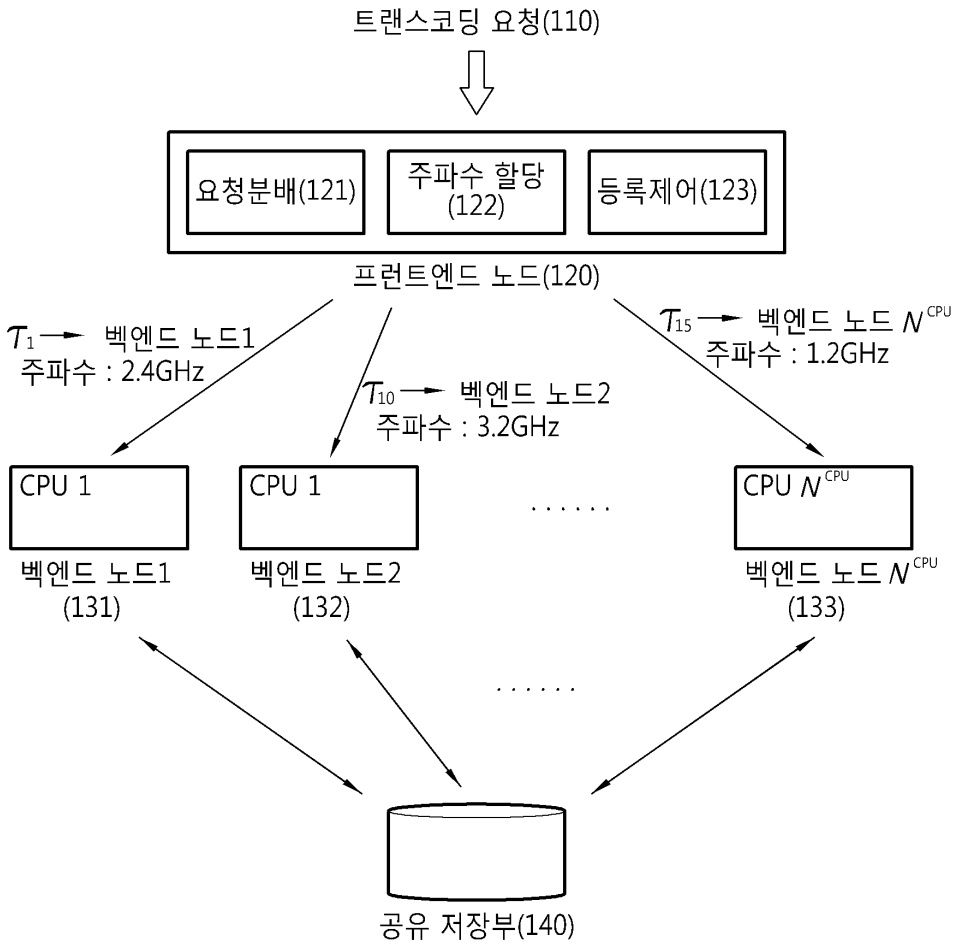
[0069] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0070] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

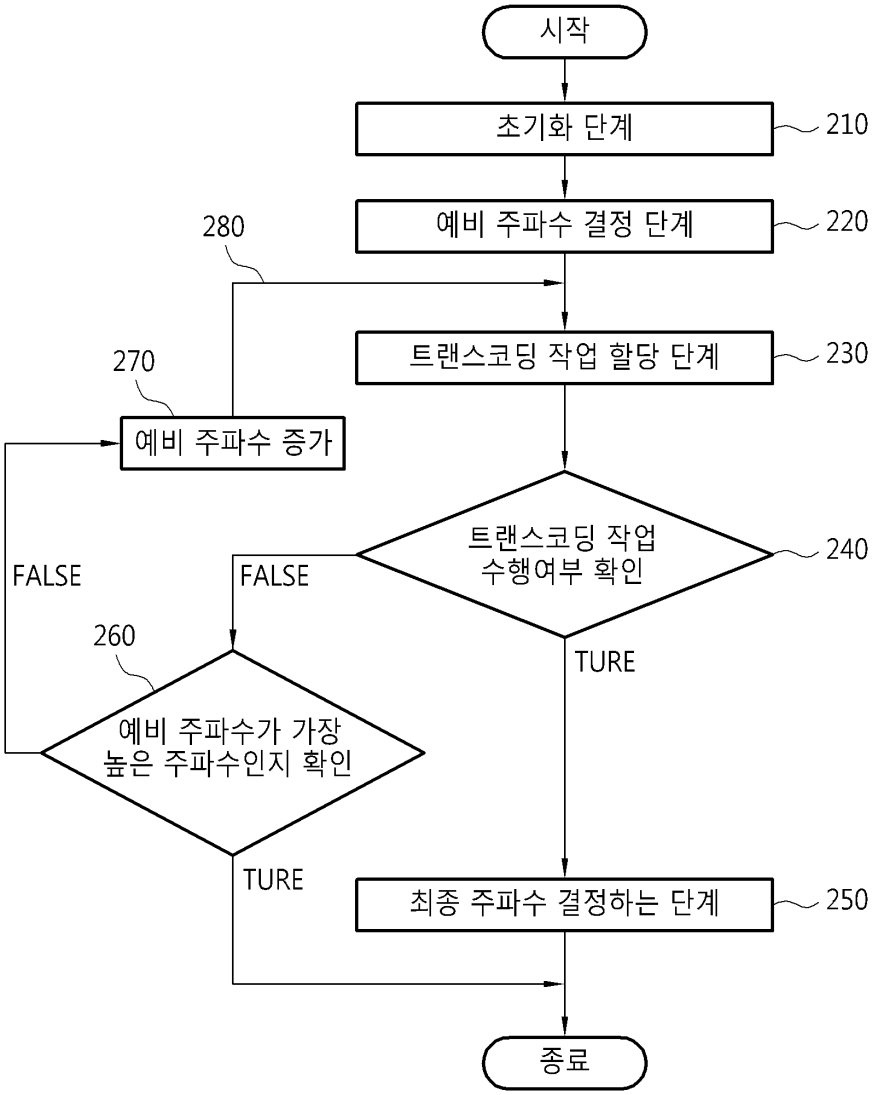
[0071] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

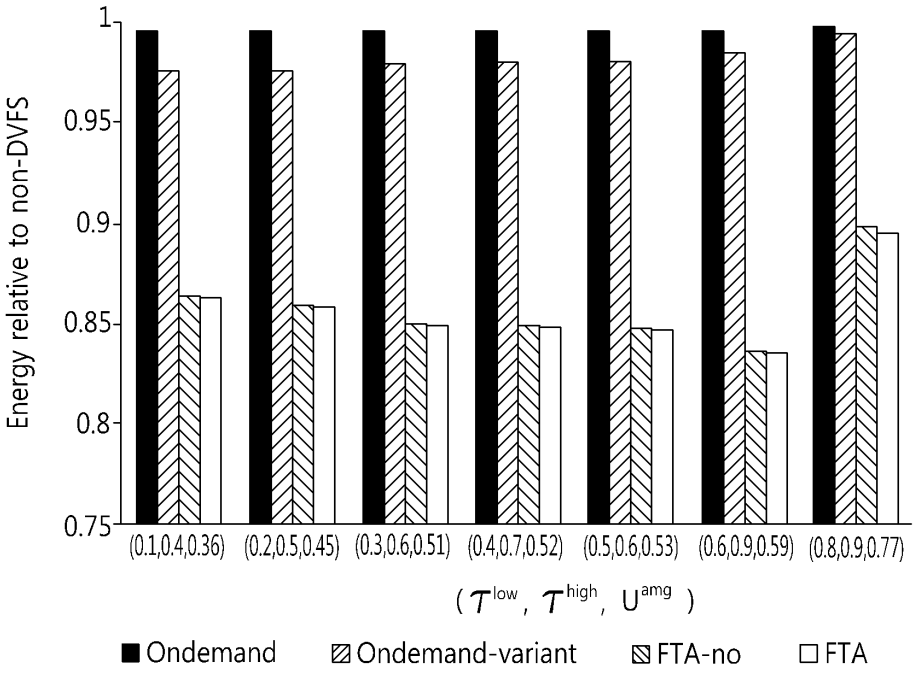
도면1



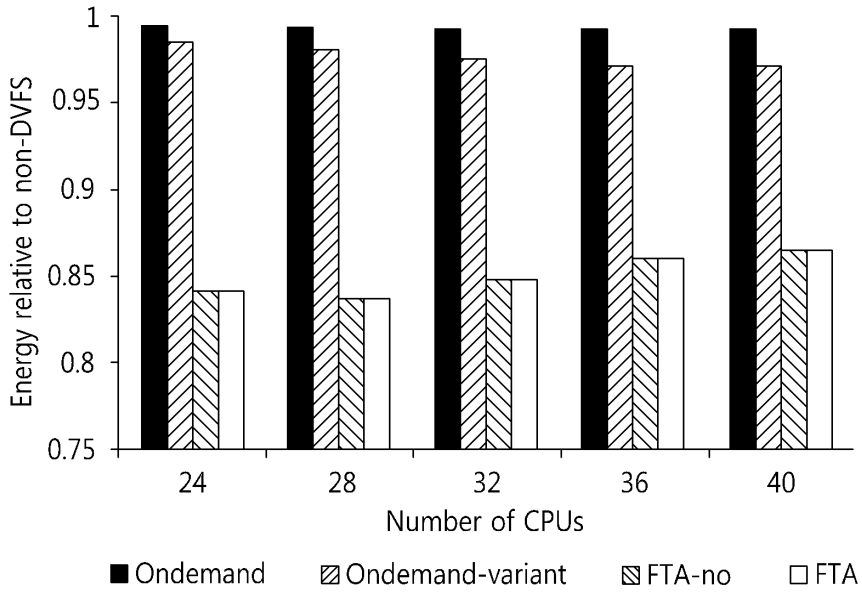
도면2


시작
초기화 단계
210
예비 주파수 결정 단계
220
280
트랜스코딩 작업 할당 단계
230
270
예비 주파수 증가
트랜스코딩 작업
수행여부 확인
240
FALSE
FALSE
TURE
260
예비 주파수가 가장
높은 주파수인지 확인
TURE
최종 주파수 결정하는 단계
250
종료

도면3


1
0.95
0.9
0.85
0.8
0.75
Energy relative to non-DVFS
(0.1,0.4,0.36)
(0.2,0.5,0.45)
(0.3,0.6,0.51)
(0.4,0.7,0.52)
(0.5,0.6,0.53)
(0.6,0.9,0.59)
(0.8,0.9,0.77)
(T^{low} , T^{high} , U^{amg})
Ondemand
Ondemand-variant
FTA-no
FTA

도면4


1
0.95
0.9
0.85
0.8
0.75
Energy relative to non-DVFS
24
28
32
36
40
Number of CPUs
Ondemand
Ondemand-variant
FTA-no
FTA