



(19) 대한민국특허청(KR)

(12) 등록특허공보(B1)

(45) 공고일자 2020년09월17일

(11) 등록번호 10-2157379

(24) 등록일자 2020년09월11일

(51) 국제특허분류(Int. Cl.)

H04N 21/2343 (2011.01) H04N 21/2387

(2011.01)

H04N 21/443 (2016.01)

(52) CPC특허분류

H04N 21/23439 (2013.01)

H04N 21/234309 (2013.01)

(21) 출원번호 10-2019-0034101

(22) 출원일자 2019년03월26일

심사청구일자 2019년03월26일

(56) 선행기술조사문헌

KR100869513 B1*

KR101454809 B1*

KR101871116 B1

*는 심사관에 의하여 인용된 문헌

(73) 특허권자

인하대학교 산학협력단

인천광역시 미추홀구 인하로 100(용현동, 인하대학교)

(72) 발명자

송민석

서울특별시 서초구 남부순환로 2311-12, 107동 402호(방배동, 래미안방배아트힐)

이다영

인천광역시 미추홀구 인하로119번길 7, 청호주택 406호(용현동)

(74) 대리인

양성보

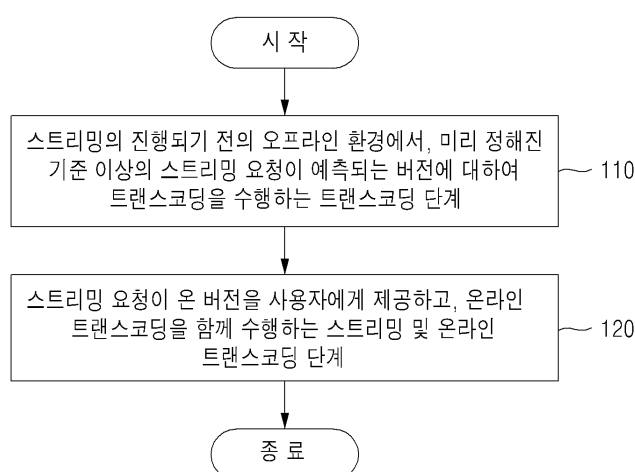
전체 청구항 수 : 총 4 항

심사관 : 김응권

(54) 발명의 명칭 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법

(57) 요약

DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법 및 장치가 제시된다. 본 발명에서 제안하는 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법은 스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩 단계 및 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계를 포함하고, 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계는, 스트리밍 요청이 올 것으로 예측되지만 미리 정해진 기준 이하의 사용자 인기도를 갖는 경우, 프로세스의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 수행한다.

대표도 - 도1

(52) CPC특허분류

H04N 21/2387 (2013.01)

H04N 21/4436 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 NRF-2017M3C4A7080248

부처명 교육부

과제관리(전문)기관명 한국연구재단

연구사업명 원천기술개발사업 (차세대정보컴퓨팅기술개발사업)

연구과제명 클라우드 환경에서 콘텐츠 및 사용자 속성을 고려한 저전력 멀티미디어 시스템 SW

개발

기 여 율 1/2

과제수행기관명 인하대학교

연구기간 2018.03.01 ~ 2020.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호 NRF-2018R1D1A1B07050614

부처명 교육부

과제관리(전문)기관명 한국연구재단

연구사업명 이공분야기초연구사업 (기본연구지원사업)

연구과제명 엣지 클러스터 기반 비디오 스트리밍 환경에서 협업 캐싱을 통한 전력 최적화 기법

에 관한 연구

기 여 율 1/2

과제수행기관명 인하대학교

연구기간 2018.06.01 ~ 2020.05.30

명세서

청구범위

청구항 1

스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩 단계; 및

스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계

를 포함하고,

스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계는,

스트리밍 요청이 올 것으로 예측되지만 미리 정해진 기준 이하의 사용자 인기도를 갖는 경우, 프로세스의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 수행하고,

사용자 인기도에 기초하는 온라인에서의 트랜스코딩 수행을 이용하여 오프라인에서의 트랜스코딩 워크로드를 계산하고, 트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화하며,

스트리밍 요청에 따라 전송할 비디오 세그먼트가 없는 경우, 온라인에서 가장 낮은 비트율 버전을 트랜스코딩하여 전송하고, 온라인 및 오프라인 워크로드를 분리하고,

트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화하기 위해 비디오 세그먼트의 가능한 트랜스코딩 버전 조합의 모든 경우의 수에 대한 집합을 구하고, 오프라인 트랜스코딩의 소요 시간과 온라인 트랜스코딩의 소요 시간을 계산하며, 전체 스트리밍 요청 횟수 중 한번 이상 요청된 가장 낮은 버전의 요청 횟수 및 요청 확률을 계산하고, 트랜스코딩할 비트율 버전의 조합에 속한 버전의 품질 이득을 모두 합한 값이 최대가 되도록 하는 비트율 버전 조합을 선택하여 트랜스코딩 작업을 수행하는

비디오 트랜스코딩을 위한 온라인 및 오프라인 작업 분배 방법.

청구항 2

제1항에 있어서,

스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩 단계는,

사용자 스트리밍 비디오 인지 품질을 최대화하고 온라인, 오프라인에서 사용되는 전력의 합이 미리 정해진 전력 한도를 넘지 않도록 사용자 인기도에 기초하여 스트리밍 요청을 예측하는

비디오 트랜스코딩을 위한 온라인 및 오프라인 작업 분배 방법.

청구항 3

삭제

청구항 4

삭제

청구항 5

제1항에 있어서,

사용자 스트리밍 비디오 인지 품질의 최대화 및 프로세서 소모 전력의 최적화를 위해 휴리스틱 기반 알고리즘 또는 동적 프로그래밍 기반 알고리즘 중 어느 하나를 이용하는

비디오 트랜스코딩을 위한 온라인 및 오프라인 작업 분배 방법.

청구항 6

스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩부; 및

스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩부

를 포함하고,

스트리밍 및 온라인 트랜스코딩부는,

스트리밍 요청이 올 것으로 예측되지만 미리 정해진 기준 이하의 사용자 인기도를 갖는 경우, 프로세스의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 수행하고,

트랜스코딩부는,

사용자 스트리밍 비디오 인지 품질을 최대화하고 온라인, 오프라인에서 사용되는 전력의 합이 미리 정해진 전력 한도를 넘지 않도록 사용자 인기도에 기초하여 스트리밍 요청을 예측하고,

사용자 인기도에 기초하는 온라인에서의 트랜스코딩 수행을 이용하여 오프라인에서의 트랜스코딩 워크로드를 계산하고, 트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화하며,

스트리밍 및 온라인 트랜스코딩부는,

사용자 인기도에 기초하는 온라인에서의 트랜스코딩 수행을 이용하여 오프라인에서의 트랜스코딩 워크로드를 계산하고, 트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화하고,

트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화하기 위해 비디오 세그먼트의 가능한 트랜스코딩 버전 조합의 모든 경우의 수에 대한 집합을 구하고, 오프라인 트랜스코딩의 소요 시간과 온라인 트랜스코딩의 소요 시간을 계산하며, 전체 스트리밍 요청 횟수 중 한번 이상 요청된 가장 낮은 버전의 요청 횟수 및 요청 확률을 계산하고, 트랜스코딩할 비트율 버전의 조합에 속한 버전의 품질 이득을 모두 합한 값이 최대가 되도록 하는 비트율 버전 조합을 선택하여 트랜스코딩 작업을 수행하는

비디오 트랜스코딩을 위한 온라인 및 오프라인 작업 분배 장치.

청구항 7

삭제

청구항 8

삭제

발명의 설명

기술 분야

[0001] 본 발명은 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법에 관한 것이다.

배경 기술

[0002] 최근 인터넷 및 스마트 기기 기술의 발전을 통해 Youtube, Netflix, Twitch 등에서 제공하는 비디오 스트리밍 서비스가 대중화되었다.

[0003] 이러한 스트리밍 업체들은 일반적으로 동적 적응 스트리밍(DASH, Dynamic Adaptive Streaming over HTTP)을 사용한다. 동적 적응 스트리밍은 비디오를 여러 개의 비디오 세그먼트들로 분할한 후, 해당 세그먼트를 여러 버전의 비트율로 트랜스코딩한 후 사용자의 네트워크 환경에 맞추어 가장 적합한 비트율인 세그먼트를 제공하는 방법이며, 이를 위해서는 다양한 비트율의 세그먼트들이 필요하다. Netflix에 의하면 총 120번의 트랜스코딩 연산이 1개의 비디오에 대해서 필요한 것으로 알려져 있다.

[0004] 트랜스코딩은 고도의 연산이 집약되어 있으므로, 비디오 스트리밍 코딩 서버는 상당한 전력을 필요로 한다. 또한 비디오 스트리밍 서비스의 인기가 점점 증가함에 따라 트랜스 코딩 작업 부하 역시 증가한다. 따라서, 전력 비용의 최적화는 서비스 제공 업체에게 있어 가장 중요한 문제 중 하나이므로 이를 위한 연구가 진행되었으나 대부분 모든 버전의 트랜스코딩을 전제로 하거나, 그렇지 않더라도 선택된 버전의 트랜스코딩을 오프라인에서 일시에 한꺼번에 트랜스코딩 할 때의 프로세서의 부하가 상당했으며, 온라인에서의 추가적인 트랜스코딩에 대한 전력이 고려되지 않아 전체 전력의 합이 원하는 전력 비용을 초과하는 모습을 보였다.

[0005] 오버헤드를 줄이기 위해 사용자 인기도를 통한 스트리밍 예측을 바탕으로 일부 트랜스코딩 작업을 미루거나 혹은 제외할 수 있다. 이를 위해서는 실제 스트리밍에 들어가기 전에 오프라인 환경에서 트랜스코딩할 버전을 선택하거나, 실제 스트리밍에 들어갔을 시 미리 트랜스코딩되지 않은 버전이 스트리밍 요청이 되었을 때, 전체적인 사용자 인지 품질을 최대화 하면서 온라인 환경에서의 트랜스코딩을 진행하는 것에 대한 고려가 필요하다.

[0006] 지금까지 트랜스코딩의 소모 전력을 줄이기 위한 연구는 많이 진행되어왔으나, 프로세서의 부하를 고려하여 워크로드를 온라인/오프라인으로 분배하면서 트랜스코딩할 비디오 비트율 버전을 선택하는 연구는 진행되지 않았다.

발명의 내용

해결하려는 과제

[0007] 본 발명이 이루고자 하는 기술적 과제는 사용자 인기도를 이용하여 스트리밍 요청을 예측함으로써 프로세서의 트랜스코딩을 온라인/오프라인으로 나누어 프로세서의 워크로드를 줄이면서, 사용자 인지 품질을 최대화하는 방법 및 장치를 제공하는데 있다. 또한, 사용자 제공 품질의 최대화 및 전체 전력의 합이 설정한 전력 한도를 넘지 않을 것을 함께 고려하여 비디오 세그먼트의 어떤 버전을 언제 트랜스코딩 할 것인가를 선택하는 2가지 기법을 제안한다.

과제의 해결 수단

[0008] 일 측면에 있어서, 본 발명에서 제안하는 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법은 스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩 단계 및 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계를 포함하고, 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계는, 스트리밍 요청이 올 것으로 예측되지만 미리 정해진 기준 이하의 사용자 인기도를 갖는 경우, 프로세서의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 수행한다.

[0009] 스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩 단계는 사용자 스트리밍 비디오 인지 품질을 최대화하고 온라인, 오프라인에서 사용되는 전력의 합이 미리 정해진 전력 한도를 넘지 않도록 사용자 인기도에 기초하여 스트리밍 요청을 예측한다.

[0010] 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계는 사용자 인기도에 기초하는 온라인에서의 트랜스코딩 수행을 이용하여 오프라인에서의 트랜스코딩 워크로드를 계산하고, 트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화한다.

[0011] 스트리밍 요청에 따라 전송할 비디오 세그먼트가 없는 경우, 온라인에서 가장 낮은 비트율 버전을 트랜스코딩하여 전송하고, 온라인 및 오프라인 워크로드를 분리한다.

[0012] 사용자 스트리밍 비디오 인지 품질의 최대화 및 프로세서 소모 전력의 최적화를 위해 휴리스틱 기반 알고리즘 또는 동적 프로그래밍 기반 알고리즘 중 어느 하나를 이용한다.

[0013] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 장치는 스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩부 및 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩부를 포함하고, 스트리밍 및 온라인 트랜스코딩부는 스트리밍 요청이 올 것으로 예측되지만 미리 정해진 기준 이하의 사용자 인기도를 갖는

경우, 프로세스의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 수행한다.

발명의 효과

[0014] 본 발명의 실시예들에 따르면 트랜스코딩 작업 시 설정된 전력 한도에 맞추어 비디오 비트율 버전을 선택하고 사용자 인기도 기반의 스트리밍 요청 예측을 통해 비디오 스트리밍 서버의 프로세서 부하를 줄이고, 전체 소모 전력을 감소시킬 수 있다.

도면의 간단한 설명

[0015] 도 1은 본 발명의 일 실시예에 따른 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법을 설명하기 위한 도면이다.

도 2는 본 발명의 일 실시예에 따른 동적 프로그래밍 기반 알고리즘을 나타내는 도면이다.

도 3은 본 발명의 일 실시예에 따른 휴리스틱 기반 알고리즘을 나타내는 도면이다.

도 4는 본 발명의 일 실시예에 따른 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 장치의 구성을 나타내는 도면이다.

도 5는 본 발명의 일 실시예에 따른 휴리스틱 기반 알고리즘과 모든 버전에 대한 트랜스코딩의 결과를 비교한 그래프이다.

도 6은 본 발명의 일 실시예에 따른 방법과 종래기술의 비트율 선택 기법과의 평균 전력 소비를 비교한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0016] 본 발명은 비디오 스트리밍 서버(Video streaming server)에서 트랜스코딩(transcoding)을 위해 프로세서의 부하를 줄이기 위해 워크로드를 온라인/오프라인으로 분리하는 기법을 제안한다. 프로세서에서 소비되는 전력을 줄이면서 사용자에게 제공되는 비디오의 사용자 인지 품질을 높이기 위해, 트랜스코딩 작업 시 전력 한도에 맞추어 비디오 비트율 버전을 선택할 뿐만 아니라 사용자 인기도를 이용하여 스트리밍 요청을 예측함으로써 온라인/오프라인 워크로드 분배를 통해 프로세서의 부하를 줄일 수 있다. 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0018] 도 1은 본 발명의 일 실시예에 따른 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법을 설명하기 위한 도면이다.

[0019] 본 발명은 사용자 인기도를 통한 스트리밍 예측을 바탕으로 트랜스코딩 작업을 온라인 및 오프라인 워크로드 분배를 통해 프로세스의 부하를 줄일 수 있다. 비디오 스트리밍 서버에서 트랜스코딩 전력을 최적화하면서 동시에 사용자에게 제공되는 스트리밍 비디오의 품질을 최대화하는 것으로, 트랜스코딩 전력 한도 내에서 사용자 인지 품질을 최대화하기 위해 비디오 비트율 버전을 선택한다. 또한, 트랜스코딩 전력 한도 내에서 사용자 인지 품질을 최대화하기 위해 온라인/오프라인에서 트랜스 코딩될 비디오 비트율 버전을 선택한다.

[0020] 제안하는 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법은 스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행하는 트랜스코딩 단계(110) 및 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행하는 스트리밍 및 온라인 트랜스코딩 단계(120)를 포함한다.

[0021] 단계(110)에서, 스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행한다.

[0022] 사용자 스트리밍 비디오 인지 품질을 최대화하고 온라인, 오프라인에서 사용되는 전력의 합이 미리 정해진 전력 한도를 넘지 않도록 사용자 인기도에 기초하여 스트리밍 요청을 예측한다.

[0023] 단계(120)에서, 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행한다. 이때, 스트리밍 요청이 올 것으로 예측되지만 미리 정해진 기준 이하의 사용자 인기도를 갖는 경우, 프로세스의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 수행한다.

[0024] 사용자 인기도에 기초하는 온라인에서의 트랜스코딩 수행을 이용하여 오프라인에서의 트랜스코딩 워크로드를 계

산하고, 트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화한다.

[0025] 스트리밍 요청에 따라 전송할 비디오 세그먼트가 없는 경우, 온라인에서 가장 낮은 비트율 버전을 트랜스코딩하여 전송하고, 온라인 및 오프라인 워크로드를 분리할 수 있다.

[0026] 이와 같이, 사용자 스트리밍 비디오 인지 품질의 최대화 및 프로세서 소모 전력의 최적화를 위해 휴리스틱 기반 알고리즘 또는 동적 프로그래밍 기반 알고리즘을 이용하여 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법을 수행할 수 있다. 도 2 및 도 3을 참조하여 아래에서 더욱 상세히 설명한다.

[0027] 더욱 상세하게는, 트랜스코딩 단계는 스트리밍이 진행되기 전의 오프라인 환경에서, 높은 사용자 인기도를 가지고 있어 스트리밍 요청이 많을 것으로 예측되는 버전에 대하여 먼저 트랜스코딩을 시행하는 과정이고, 스트리밍 및 온라인 트랜스코딩 단계는 스트리밍 요청이 온 버전을 제공해주는 세션이며 이 때 온라인 트랜스코딩이 함께 진행된다. 온라인 트랜스코딩은 요청이 올 것으로 예측되지만 비교적 인기도가 낮아 확실하지 않은 경우, 프로세서의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 하는 것이다. 이 때 사용자 인지 품질을 최대화하면서 동시에 온라인, 오프라인에서 사용되는 전력의 합이 설정한 전력 한도를 넘지 않도록 해야 하므로 사용자 인기도에 기반한 스트리밍 요청을 예측하고 이를 반영하는 것이 굉장히 중요하다.

[0029] N^{seg} 는 전체 비디오 세그먼트의 수이며, 각 비디오 세그먼트는 원본을 포함하여 N^{ver} 개의 버전을 가지고 있다.

[0030] 각 비디오 버전을 $V_{i,j}$ ($i = 1, \dots, N^{seg}$ and $j = 1, \dots, N^{ver}$) 라고 한다. 번호가 낮을수록 비트율이 낮은 버전이다. 즉 $V_{i,N^{ver}}$ 이 비디오 세그먼트 i 원본, $V_{i,1}$ 이 가장 낮은 비트율 버전이다.

[0031] N^{tran} 는 트랜스코딩의 타겟 버전의 숫자를 의미하며, $N^{ver}-1$ 이다. 당연히 원본인 N^{ver} 는 트랜스코딩의 소스로만 사용되므로 포함되지 않는다.

[0032] $p_{i,j}$ 는 $V_{i,j}$ 의 사용자 인기도 기반의 접근 확률을, $Q_{i,j}$ 는 $V_{i,j}$ 의 SSIM값에 기반한 비디오 품질을 의미한다. $G_{i,j}$ 는 품질 이득으로써, 아래와 같이 정의된다.

$$G_{i,j} = p_{i,j} Q_{i,I_{i,j}}$$

[0034] 여기서 $I_{i,j}$ 는 사용자 요청에 대해 전송하는 세그먼트를 의미한다.

[0035] 만약 서버에 요청된 세그먼트 j 가 없을 경우 그보다 비트율이 낮은 버전 중 가장 높은 버전, 다시 말해서 j 의 바로 아래 비트율 버전을 전송하며, 아래의 비트율 버전이 아무것도 없을 경우 가장 마지막 버전을 온라인 트랜스코딩하여 전송하게 된다.

[0036] S_i 는 비디오 세그먼트 i 가 가능한 트랜스코딩 버전 조합의 모든 경우의 수에 대한 집합을 의미하며, $S_{i,m}$ 은 S_i 의 m 번째 집합을 의미한다. 예를 들어 N^{tran} 가 2일 경우, 2가 원본 다음으로 가장 높은 비트율 버전일 것이며, $S_i = \{S_{i,1} = \emptyset, S_{i,2} = \{V_1\}, S_{i,3} = \{V_2\}, S_{i,4} = \{V_1, V_2\}\}$ 이다. 여기서 m 이 1일 경우는 공집합을 의미한다. 또한 N^{set} 는 S_i 의 전체 숫자를 의미하며, $N^{set} = 2^{N^{tran}}$ 이다.

[0037] 트랜스코딩 시간이 줄어들수록 소모 전력은 감소한다. 그러므로 사용자 인기도를 이용한 스트리밍 요청을 예측함으로써 온라인/오프라인 워크로드를 분리하기 위해, 트랜스코딩 세션에서의 오프라인 트랜스코딩의 소요 시간 $C_{i,m}^{offline}$ 과 스트리밍 세션에서의 온라인 트랜스코딩의 소요 시간 $C_{i,m}^{online}$ 을 정의했다.

[0038] $C_i^{i \rightarrow k}$ 는 $V_{i,j}$ 에서 $V_{i,k}$ 로 트랜스코딩하는 데에 사용되는 프로세서의 소요 시간을 의미한다. 이때 반드시 $j > k$ 여야 한다.

[0039] $C_{i,m}^{offline}$ 의 정의는 아래와 같다. ($m = 1, \dots, N^{set}$)

$$C_{i,m}^{offline} = \sum_{\forall V_{i,j} \in S_{i,m}} C_i^{H_{i,j} \rightarrow j}$$

[0040]

[0041] $C_{i,m}^{online}$ 의 정의는 아래와 같다.

$$C_{i,m}^{online} = \begin{cases} C_i^{H_{i,1}} & \sum_{\forall j < V_{i,m}^{low}} N_{i,j} \geq 1 \\ C_i^{H_{i,1}} \sum_{\forall j < V_{i,m}^{low}} N_{i,j} & otherwise \end{cases}$$

[0042]

[0043] 여기서 $V_{i,m}^{low}$ 은 1번 이상 요청된 가장 낮은 버전을 의미한다. $H_{i,j}$ 는 $V_{i,j}$ 를 트랜스코딩하기 위한 소스를 의미하며, S_i 에 존재하는 j 보다 큰 비트율 버전 중 가장 작은 버전, 다시 말해 j 의 바로 위의 비트율 버전을 의미한다. 그러므로 반드시 $H_{i,j} > j$ 이다. $N_{i,j}$ 는 전체 스트리밍 요청 횟수 L^{total} 중 $V_{i,j}$ 의 요청 횟수는 얼마 인지를 의미하며, $\lambda_{i,j}$ 는 $V_{i,j}$ 의 요청 확률을 의미한다. 그러므로 $N_{i,j}$ 는 아래와 같이 정의된다.

$$N_{i,j} = \lambda_{i,j} L^{total}$$

[0044]

[0045] $C_{i,m}^{total}$ 은 전체 트랜스코딩에 사용된 총 소요 시간이며, $C_{i,m}^{online}$ 과 $C_{i,m}^{offline}$ 의 합이다. 스트리밍 비디오의 사용자 인지 품질을 최대화하기 위한 문제를 다음과 같이 제시하였다.

$$\begin{aligned} & \text{Maximize} \quad \sum_{i=1}^{N^{seg}} G_{i,Y_i}^{sum} \\ & \text{subject to} \quad \sum_{i=1}^{N^{seg}} C_{i,m} \leq C^{limit} \\ & \quad Y_i = 1, \dots, N^{set} \end{aligned}$$

[0046]

[0047] Y_i 는 해당 문제를 해결하기 위해 선택된 $S_{i,m}$ 의 m 값을 의미하며, 다시말 해 비디오 세그먼트 i 가 가능한 트랜스코딩 버전 조합의 모든 경우의 수에 대한 집합 S_i 의 Y_i 번째 요소를 의미한다. S_{i,Y_i} 번째 집합의 버전이 오프라인에 트랜스코딩됨을 의미한다.

[0048] $G_{i,m}^{sum}$ 는 비디오 세그먼트 i 의 트랜스코딩할 비트율 버전의 조합이 m 일 경우, 해당 집합 m 에 속한 버전 j 의 품질 이득 $G_{i,j}$ 을 모두 합한 것이다.

[0049] 해당 문제는 G_{i,Y_i}^{sum} 를 최대화하는 비트율 버전 조합 Y_i 을 선택하여 트랜스코딩 작업을 진행하도록 정의하였다.

- [0050] 사용자 인기를 고려하여 온라인에서의 트랜스코딩 수행을 고려하여 오프라인에서의 트랜스코딩 워크로드를 계산하고, 서버 관리자에 의해 설정된 트랜스코딩 시간 한도 C^{limit} 내에서 스트리밍 비디오의 사용자 인지 품질을 최대화를 목적으로 했다.
- [0051] 만약 스트리밍 요청에 따라 보내줄 비디오 세그먼트가 없을 경우 온라인에서 가장 낮은 비트율 버전을 트랜스코딩해서 전송하며 이 역시 문제 정의에서 함께 고려함으로써 온라인/오프라인 워크로드를 분리하였다.
- [0052] 사용자 스트리밍 비디오 인지 품질의 최대화 및 프로세서 소모 전력의 최적화를 위해 최적값에 근접한 값을 찾고 빠른 시간 복잡도를 가진 휴리스틱 기반과 반드시 최적값을 찾는 동적 프로그래밍 기반의 두 가지 알고리즘을 개발하였으며 각각 도 2 및 도 3에 도시하였다.
- [0054] 도 2는 본 발명의 일 실시예에 따른 동적 프로그래밍 기반 알고리즘을 나타내는 도면이다.
- [0055] 도 2을 참조하여 동적 프로그래밍 알고리즘(TPS-DP)을 설명한다.
- [0056] $D_{i,n}^{gain}$ 은 트랜스코딩 시간 한도의 값이 n 일 때의 비디오 세그먼트 i 의 최대 품질 이득의 합 값이다. ($i = 1, ..., N^{seg}$ and $n = 0, ..., C^{limit}$)
- [0057] 트랜스코딩 시간 한도의 값이 n 일 때의 비디오 세그먼트 i 의 최대 품질 이득 값 $D_{i,n}^{gain}$ 을 얻을 수 있는 비트율 버전의 조합이 $S_{i,m}$ 이라면, $D_{i,n}^{back}$ 은 m 이다.
- [0058] 동적 프로그래밍 알고리즘은 도 2에 표현되었으며, 아래와 같은 3개의 단계로 구성되어있다.
- [0059] 1) 초기화: 먼저 $D_{i,n}^{gain}$ 과 $D_{i,n}^{back}$ 의 값을 모두 0으로 초기화한다(210). 이후 $D_{i,n}^{gain}$ 의 값을 각 시간 한도의 품질 이득의 합 $G_{i,m}^{sum}$ 중 최대값으로, $D_{i,n}^{back}$ 의 값은 m 으로 초기화한다(220).
- [0060] 2) 재귀: 비디오 세그먼트 i ($i = 2, ..., N^{seg}$)의 각 반복을 통해, $D_{i,n}^{gain}$ 은 아래와 같이 계산된다.
- [0061]
$$D_{i,n}^{gain} = \max(D_{i-1,n}^{gain}, \max_{m=1, ..., N^{set}} (D_{i-1,n-C_{i,m}^{sum}}^{gain} + G_{i,m}^{sum}))$$
- [0062] 또한 $D_{i,n}^{back}$ 는 위의 식을 만족하는 m 의 값이다(230).
- [0063] 3) 역추적: $D_{N^{seg}, C^{limit}}^{gain}$ 는 트랜스코딩 품질 이득의 최적값이다. 그러므로 비디오 세그먼트 N^{seg} 및 트랜스코딩 시간 한도 C^{limit} 에서 시작하여 최적의 비트율 버전 조합 Y_i 를 찾는다(240).
- [0065] 도 3은 본 발명의 일 실시예에 따른 휴리스틱 기반 알고리즘을 나타내는 도면이다.
- [0066] 도 3을 참조하여 휴리스틱 기반 알고리즘(TPS-H)을 설명한다.
- [0067] 동적 프로그래밍 알고리즘이 가진 시간, 공간적으로 많은 비용이 드는 문제로 인해 휴리스틱 기반의 알고리즘(TPS-H)를 함께 제안한다.
- [0068] 이 문제를 해결하기 위해 $P_{i,m}$ 를 정의했으며 아래와 같다.
- [0069] I_i 는 각 비디오 세그먼트 i 에 선택된 트랜스코딩할 비트율 버전의 조합이며, 처음에는 1로 초기화한 후, 알고리즘에 의하여 선택된 조합의 시간 비용의 합 C^{temp} 이 C^{limit} 을 넘기 직전까지 모든 $P_{i,m}$ 의 경우의 수를 담은

배열 A 에서 가장 큰 값인 $P_{i, high}$ 를 지속적으로 선택하면서 I_i 값을 변경해 나간다.

- [0071] 도 4는 본 발명의 일 실시예에 따른 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 장치의 구성을 나타내는 도면이다.
- [0072] DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 장치(400)는 트랜스코딩부(410) 및 스트리밍 및 온라인 트랜스코딩부(420)를 포함한다. 트랜스코딩부(410) 및 스트리밍 및 온라인 트랜스코딩부(420)는 도 1의 단계들(110~120)을 수행하기 위해 구성될 수 있다.
- [0073] 트랜스코딩부(410)는 스트리밍이 진행되기 전의 오프라인 환경에서, 미리 정해진 기준 이상의 스트리밍 요청이 예측되는 버전에 대하여 트랜스코딩을 수행한다.
- [0074] 사용자 스트리밍 비디오 인지 품질을 최대화하고 온라인, 오프라인에서 사용되는 전력의 합이 미리 정해진 전력 한도를 넘지 않도록 사용자 인기도에 기초하여 스트리밍 요청을 예측한다.
- [0075] 스트리밍 및 온라인 트랜스코딩부(420)는 스트리밍 요청이 온 버전을 사용자에게 제공하고, 온라인 트랜스코딩을 함께 수행한다. 이때, 스트리밍 요청이 올 것으로 예측되지만 미리 정해진 기준 이하의 사용자 인기도를 갖는 경우, 프로세스의 부하를 줄이기 위해 스트리밍 요청이 왔을 시 온라인에서 트랜스코딩을 수행한다.
- [0076] 사용자 인기도에 기초하는 온라인에서의 트랜스코딩 수행을 이용하여 오프라인에서의 트랜스코딩 워크로드를 계산하고, 트랜스코딩 시간 한도 내에서 스트리밍의 사용자 스트리밍 비디오 인지 품질을 최대화한다.
- [0077] 스트리밍 요청에 따라 전송할 비디오 세그먼트가 없는 경우, 온라인에서 가장 낮은 비트율 버전을 트랜스코딩하여 전송하고, 온라인 및 오프라인 워크로드를 분리할 수 있다.
- [0078] 이와 같이, 사용자 스트리밍 비디오 인지 품질의 최대화 및 프로세서 소모 전력의 최적화를 위해 휴리스틱 기반 알고리즘 또는 동적 프로그래밍 기반 알고리즘을 이용하여 DASH 서버에서의 비디오 트랜스코딩을 위한 온라인/오프라인 작업 분배 방법을 수행할 수 있다.
- [0079] 스트리밍 비디오의 사용자 인지 품질과 프로세서의 트랜스코딩에 대한 소모 전력을 예측하는 프로그램을 통해 본 발명에서 제안한 기법의 효율성을 측정했다.
- [0080] Poisson 분포 기반의 스트리밍 환경에서 비디오의 인기도가 Zipf 분포로 구성된 환경에서 진행되었으며 SSIM 틀에서 사용자 인지 품질을 추출하고 요청될 비트율 버전 예측을 통한 온라인/오프라인 워크로드 분리 및 프로세서 소모 전력에 따른 스트리밍 결과를 확인하였다.
- [0081] 트랜스코딩에 소모되는 시간당 전력 P^{tran} 은 93W로 설정하였다.
- [0082] 동적 프로그래밍 알고리즘은 반드시 최적값을 찾는 반면 실행 시간이 많이 걸리므로 이를 해결하기 위한 휴리스틱 기반 알고리즘을 함께 개발하였다. 그리고, 각 비트율 버전의 인기도 및 전력 한도의 설정을 변경하며 비교하였다. 비트율 버전의 인기도는 4가지 경우로 구성했으며 HVP는 높은 비트율 버전이 인기 있는 경우, MVP는 중간 비트율 버전이 인기 있는 경우, LVP는 낮은 비트율 버전이 인기 있는 경우, RVP는 비트율 버전의 인기도가 랜덤으로 결정되는 경우이다. 전력 한도 설정은 4가지 경우로 구성했으며 전체 비트율 버전을 트랜스코딩할 경우의 35%, 50%, 65%, 80%이다.
- [0083] 표 1은 스트리밍된 비디오의 사용자 인지 품질을, 표 2는 비디오 버전의 선택 및 분배를 위한 알고리즘을 수행하는데 걸린 시간을 나타내었으며, 휴리스틱 기반의 알고리즘이 동적 프로그래밍 알고리즘에 비해 걸린 시간은 훨씬 적으면서 사용자 인지 품질의 최적값에 근접함을 확인 할 수 있다.

[0084] <표 1>

인기도	알고리즘	사용자 인지 품질 (점수)			
		6.771Wh	9.673Wh	12.575Wh	16.444Wh
HVP	휴리스틱	4.315	4.454	4.503	4.523
	동적 프로그래밍	4.318	4.457	4.503	4.523
MVP	휴리스틱	4.102	4.273	4.331	4.354
	동적 프로그래밍	4.107	4.276	4.331	4.354
LVP	휴리스틱	4.005	4.106	4.133	4.143
	동적 프로그래밍	4.006	4.106	4.133	4.143
RVP	휴리스틱	4.181	4.302	4.34	4.355
	동적 프로그래밍	4.184	4.303	4.341	4.355

[0085]

[0086] <표 2>

인기도	알고리즘	실행 시간 (초)			
		6.771Wh	9.673Wh	12.575Wh	16.444Wh
HVP	휴리스틱	0.019	0.023	0.039	0.044
	동적 프로그래밍	12.600	44.784	74.195	123.649
MVP	휴리스틱	0.018	0.023	0.043	0.047
	동적 프로그래밍	12.466	44.950	74.888	125.420
LVP	휴리스틱	0.018	0.028	0.044	0.046
	동적 프로그래밍	12.405	44.726	71.728	127.057
RVP	휴리스틱	0.016	0.027	0.037	0.048
	동적 프로그래밍	12.610	43.414	71.299	131.001

[0087]

[0088] 각 비디오의 세그먼트가 6초인 1~3시간짜리 비디오 3000개에 대하여 비교를 진행하였으며, 이 때 비디오 세그먼트는 100만개 이상이다. 인기도 및 전력 한도의 설정 방법은 위의 비교와 같다.

[0090] 표 3과 도 5는 본 발명의 일 실시예에 따른 휴리스틱 기반 알고리즘과 모든 버전에 대한 트랜스코딩의 결과를 비교한 그래프이다.

[0091] <표 3>

인기도	TPS-ALL	TPS-H			
	11.691kWh	4.092kWh	5.846kWh	9.353kWh	4.378kWh
HVP	4.378	4.225	4.331	4.362	4.375
MVP	4.168	3.987	4.113	4.150	4.165
LVP	3.978	3.884	3.953	3.971	3.977
RVP	4.179	4.052	4.140	4.166	4.176

[0092]

[0093] 표 3은 모든 사용자 워크로드에 대한 사용자 인지 품질, 도 5는 평균 소모 전력을 표현하였다.

[0094] 모든 버전에 대한 트랜스코딩과 비교 했을 때, 소모 전력이 65% 감소 시 사용자 인지 품질 저하는 최대 4.36%만 일어나므로 인지 품질을 최대한 유지하면서 소모 전력을 줄임을 확인했다. 이는 비트율 버전 선택의 효과 및 중요성을 보여준다.

[0096] 도 6은 본 발명의 일 실시예에 따른 방법과 종래기술의 비트율 선택 기법과의 평균 전력 소비를 비교한 도면이다.

[0097] 휴리스틱 기반의 비트율 버전 선택 기법 타 인기도 기반 기법을 비교하였으며, 실험 환경과 인기도 및 전력 한도의 설정 방법은 위 비교와 같다.

[0098] 비교에 사용한 기법은 Pop-video, Pop-segment, Pop-version 총 3가지이며, Pop-video은 비디오의 인기도, Pop-segment는 비디오를 구성하는 세그먼트의 인기도, Pop-version는 각 세그먼트의 비트율 버전의 인기도가 높은 것부터 전력 한도가 찰 때까지 트랜스코딩 하는 방식이다.

[0099] 표 4와 도 6은 각 기법의 결과를 비교하였으며 표 4는 전체 워크로드에 대한 사용자 인지 품질, 도 6은 평균 소모 전력을 표현하였다.

[0100] <표 4>

인기도	알고리즘	사용자 인지 품질 (점수)			
		6.771Wh	9.673Wh	12.575Wh	16.444Wh
HVP	휴리스틱	4.225	4.331	4.362	4.375
	Pop-video	4.274	4.304	4.329	4.350
	Pop-segment	4.313	4.343	4.362	4.373
	Pop-version	4.321	4.349	4.365	4.374
MVP	휴리스틱	3.987	4.113	4.150	4.165
	Pop-video	4.044	4.080	4.109	4.136
	Pop-segment	4.091	4.127	4.149	4.163
	Pop-version	4.095	4.129	4.151	4.163
LVP	휴리스틱	3.884	3.953	3.971	3.977
	Pop-video	3.909	3.929	3.945	3.960
	Pop-segment	3.935	3.955	3.968	3.975
	Pop-version	3.935	3.955	3.968	3.975
RVP	휴리스틱	4.052	4.140	4.166	4.176
	Pop-video	4.092	4.117	4.137	4.156
	Pop-segment	4.125	4.149	4.165	4.174
	Pop-version	4.126	4.150	4.166	4.175

[0101]

[0102] 스트리밍 요청에 대해 전송할 버전이 없는 경우, 다른 인기도 기반의 기법은 온라인 트랜스코딩을 고려하지 못하여 전력 한도를 초과하나, 제안한 기법 스트리밍 요청 예측을 통한 온라인/오프라인 워크로드 분배가 이루어짐에 따라 전력 한도를 초과하지 않았다.

[0103] 모든 기법의 사용자 인지 품질의 차이는 거의 없으나, 제안한 기법은 전력 한도가 4.092kWh일 때는 다른 기법에 비해 낮은 모습을 보이지만 점점 높아질수록 다른 알고리즘 보다 높아져 전력한도가 9.353kWh일 때는 가장 높다.

[0104] 제안한 기법은 다른 인기도 기반의 기법과 달리 스트리밍 요청 예측을 통한 온라인/오프라인 트랜스코딩 워크로드 분배를 통해 전력 한도를 넘기지 않고 사용자 인지 품질을 최대한 유지시키면서, 프로세서 소모 전력을 효과적으로 줄일 수 있음을 확인할 수 있다.

[0105] 본 발명에 따르면 프로세서 소모 전력을 65% 감소시킬 때, 모든 비디오 비트율 버전을 트랜스코딩할 경우에 비해 사용자 인지 품질 저하가 평균 3.31% 인 것으로 나타났다. 또한 스트리밍 요청 시 온라인에서 추가적으로 진행되는 트랜스코딩의 전력을 고려함으로써 반드시 설정한 전력 내에서 트랜스코딩 되는 것을 확인했다.

[0107] 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 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서 (parallel processor)와 같은, 다른 처리 구성 (processing configuration)도 가능하다.

[0108] 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 장치에 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.

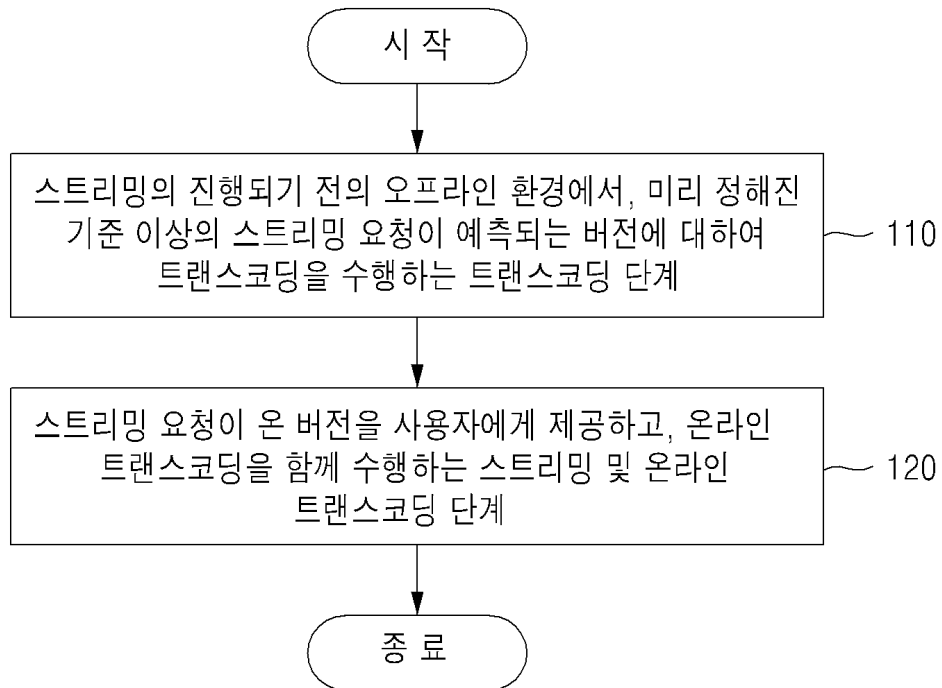
[0109] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체 (magnetic 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.

[0110] 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.

[0111] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

도면1



도면2

```

1: Temporary variables: i, m, and n;
2: for i = 1 to Nseg do
3:   for n = 0 to Climit do
4:     Di,ngain ← 0;
5:     Di,nback ← 0;
6:   end for
7: end for
8: for n = 1 to Climit do
9:   D1,ngain = maxm ∈ {m | Ci,msum ≤ n} Gi,msum;
10:  D1,nback = arg maxm=1,...,Nset D1,ngain;
11: end for
12: for i = 2 to Nseg do
13:   for n = 1 to Climit do
14:     Di,ngain ← max (Di,n-1gain, maxm=1,...,Nset (Dn-1,n-Ci,msumgain + Gi,msum));
15:     Di,nback ← arg maxm=1,...,Nset Di,ngain;
16:   end for
17: end for
18: i ← Nset;
19: n ← Climit;
20: while i > 0 do
21:  Yi ← Di,nback;
22:  i ← i - 1;
23:  n ← n - Ci,Yisum;
24: end while

```

210

220

230

240

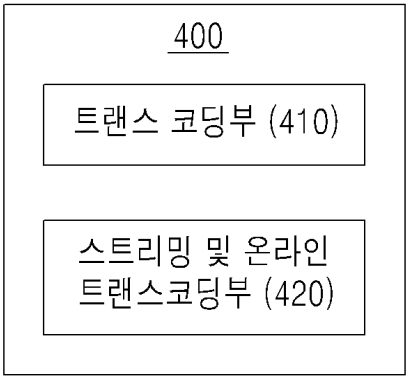
도면3

```

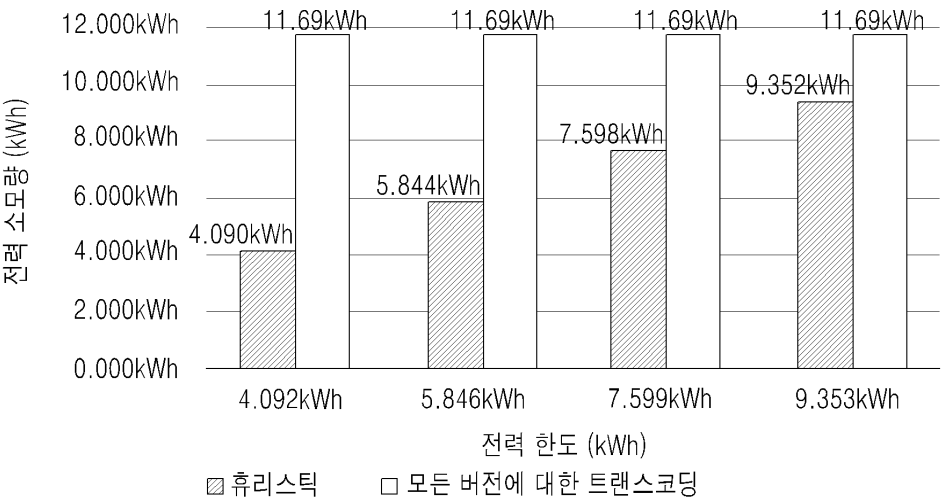
1: Temporary variable: Ctemp ← 0;
2: ∀ i, Y ← 1;
3: while Ctemp ≤ Climit do
4:   Choose the highest value of Pi,high in the set A and
   removes Pi,high from A;
5:   if high > Yi then
6:     Ctemp ← Ctemp + Ci,highsum - Ci,Yisum;
7:     Yi ← high;
8:   end if
9:   if A = ∅ then
10:    Break the loop;
11:   end if
12: end while

```

도면4



도면5



도면6

