(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년07월22일
(11) 등록번호 10-2280443
(24) 등록일자 2021년07월16일

(51) 국제특허분류(Int. Cl.)
G06F 16/2455 (2019.01) *G06F 16/22* (2019.01)
H04L 29/08 (2006.01)
(52) CPC특허분류
G06F 16/24552 (2019.01)
G06F 16/2228 (2019.01)
(21) 출원번호 10-2019-0173762
(22) 출원일자 2019년12월24일
심사청구일자 2019년12월24일
(65) 공개번호 10-2021-0081619
(43) 공개일자 2021년07월02일
(56) 선행기술조사문헌
KR101587631 B1*
US08805951 B1*
US20060026154 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
주식회사 티맥스티베로
경기도 성남시 분당구 황새울로258번길 29(수내동)
(72) 발명자
이정우
경기도 성남시 분당구 정자일로 1, B동 802호(금곡동)
박상영
서울특별시 강남구 삼성로 212, 5동 110호(대치동, 은마아파트)
(뒷면에 계속)
(74) 대리인
이대호, 박건홍

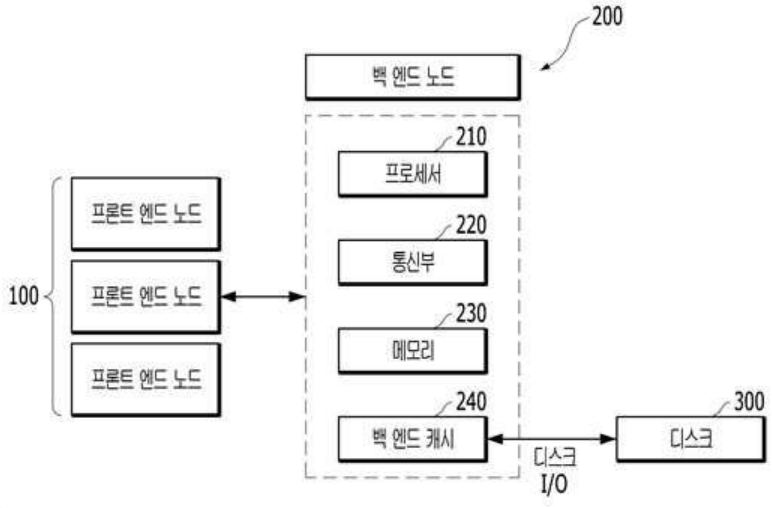
전체 청구항 수 : 총 25 항 심사관 : 이현중

(54) 발명의 명칭 **셀렉트 쿼리 처리 시 네트워크 비용 절감을 위해 멀티 캐시를 구비한 클라우드 데이터베이스 시스템**

(57) 요 약

전술한 과제를 해결하기 위한 본 개시의 몇몇 실시예에 따라, 클라우드 데이터베이스 시스템에서의 백 엔드 노드가 개시된다. 상기 백 엔드 노드는, 통신부; 버퍼 캐시 데이터(buffer cache data) 및 메타 데이터 정보를 저장하는 백 엔드 캐시(back end cache)-상기 버퍼 캐시 데이터 및 상기 메타 데이터 정보는 데이터베이스 시스템에 저장된 데이터 블록에 대응함-; 프로세서; 를 포함하고, 상기 메타 데이터 정보는, 상기 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 정보를 포함할 수 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 16/24557 (2019.01)
H04L 67/288 (2013.01)
(72) 발명자
이학용
경기도 성남시 분당구 성남대로171번길 8, 106동 1401호(금곡동, 청솔마을서광영남아파트)
김태경
경기도 성남시 분당구 성남대로171번길 8, 104동 603호(금곡동, 청솔마을서광영남아파트)
김보배
경기도 용인시 수지구 동천로99번길 8-6, 307호(동천동)

이 발명을 지원한 국가연구개발사업

과제고유번호	2018-0-00600
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기술진흥센터
연구사업명	SW컴퓨팅산업원천기술개발사업 GCS 과제
연구과제명	Heterogeneous Data Type을 통합 지원하는 Smart Storage기능과 클라우드 환경에서 서버 무한 확장 기능이 강화된, AI 기반의 Zero-DBA를 가능하게 하는 DBMS 플랫폼 개발
기 여 율	1/1
과제수행기관명	㈜티맥스데이터
연구기간	2018.05.01 ~ 2020.04.30

명 세 서

청구범위

청구항 1

통신부;

버퍼 캐시 데이터(buffer cache data) 및 메타 데이터 정보를 저장하는 백 엔드 캐시(back end cache)-상기 버퍼 캐시 데이터 및 상기 메타 데이터 정보는 데이터베이스 시스템에 저장된 데이터 블록에 대응함-; 및

프로세서;

를 포함하고,

상기 메타 데이터 정보는,

상기 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 정보를 포함하고,

상기 프로세서는,

제 1 프론트 엔드 노드로부터 제 3 데이터 블록이 상기 제 1 프론트 엔드 노드에서 인밸리데이트되었다는 신호를 수신한 경우, 제 3 메타 데이터 정보를 탐색하고,

상기 제 3 메타 데이터 정보에서 상기 제 3 데이터 블록에 대한 제 1 공유 정보를 삭제하는 - 상기 제 3 데이터 블록에 대한 제 1 공유 정보는 상기 제 1 프론트 엔드 노드에 상기 제 3 데이터 블록이 저장되어있다는 정보임 -,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 2

제 1 항에 있어서,

상기 백 엔드 캐시는 상기 버퍼 캐시 데이터에 대응되는 버퍼 헤더를 더 포함하고,

상기 버퍼 헤더는 상기 메타 데이터 정보를 포함하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 3

제 1 항에 있어서,

상기 백 엔드 캐시는 상기 버퍼 캐시 데이터에 대응되는 버퍼 헤더를 더 포함하고,

상기 메타 데이터 정보는 상기 버퍼 캐시 데이터와 동일한 검색 구조를 공유하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 4

제 3 항에 있어서,

상기 버퍼 헤더와 상기 메타 데이터 정보는,

동일한 해시 함수(hash function)을 이용해 상기 백 엔드 캐시에 저장되고,

상기 버퍼 헤더와 상기 메타 데이터 정보와 관련된 해시 함수의 입력 값은 동일하고,

상기 검색 구조 내에서 상기 버퍼 헤더와 상기 메타 데이터 정보가 저장된 공간은 상기 입력 값을 상기 해시 함수에 입력하여 얻어진 결과 값과 연관됨으로써,

상기 검색 구조를 공유하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 5

제 1 항에 있어서,

상기 메타 데이터 정보는,

비트맵(bitmap), 배열로 표현된 노드 ID 정보 또는 공유 메모리 공간 상에 B+ 트리로 표현된 노드 ID 정보를 가리키는 포인터 중 어느 하나를 포함하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 6

제 1 항에 있어서,

상기 프로세서는,

제 1 프론트 엔드 노드로부터 제 1 데이터 블록에 대한 전송 요청을 상기 통신부를 통해 수신한 경우, 제 1 버퍼 캐시 데이터 또는 제 1 메타 데이터 정보 중 적어도 하나를 상기 백 엔드 캐시에서 탐색하고,

상기 제 1 버퍼 캐시 데이터가 탐색된 경우 또는 탐색된 상기 제 1 메타 데이터 정보에 공유 정보가 존재하는 경우, 상기 제 1 버퍼 캐시 데이터 또는 상기 제 1 메타 데이터 정보 중 적어도 하나에 기초하여 제 1 데이터 블록을 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어하고,

제 1 데이터 블록에 대한 제 1 공유 정보를 제 1 메타 데이터 정보에 저장하는-상기 제 1 데이터 블록에 대한 제 1 공유 정보는 상기 제 1 프론트 엔드 노드에 상기 제 1 데이터 블록이 저장되었다는 정보임-,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 7

제 6 항에 있어서,

상기 프로세서는,

상기 제 1 메타 데이터 정보에 공유 정보가 존재하는 경우, 상기 제 1 메타 데이터 정보에 포함된 상기 제 1 데이터 블록에 대한 제 2 공유 정보를 탐색하고-상기 제 1 데이터 블록에 대한 제 2 공유 정보는 제 2 프론트 엔드 노드에 상기 제 1 데이터 블록이 저장되었다는 정보임-,

상기 통신부를 통하여 상기 제 2 프론트 엔드 노드로부터 상기 제 1 데이터 블록을 수신하고,

수신한 상기 제 1 데이터 블록을 상기 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 8

제 6 항에 있어서,

상기 제 1 메타 데이터 정보가 상기 백 엔드 캐시에 존재하지 않을 경우,

제 1 메타 데이터 정보를 수용할 수 있는 제 1 데이터 구조체를 생성 또는 로드(load)하고,

상기 제 1 데이터 구조체에 상기 제 1 데이터 블록의 정보를 저장하고,

상기 제 1 데이터 구조체를 상기 제 1 메타 데이터 정보로서 결정하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 9

제 6 항에 있어서,

상기 프로세서는,

상기 제 1 버퍼 캐시 데이터가 상기 백 엔드 캐시에 존재할 경우, 상기 제 1 버퍼 캐시 데이터를 상기 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 10

제 6 항에 있어서,

상기 프로세서는,

상기 제 1 버퍼 캐시 데이터가 존재하지 않고, 탐색된 상기 제 1 메타 데이터에 상기 공유 정보가 존재하지 않을 경우, 디스크(disk)에 상기 제 1 데이터 블록에 대한 요청 신호를 전송하도록 상기 통신부를 제어하고,

상기 디스크로부터 상기 제 1 데이터 블록을 수신하면, 상기 제 1 데이터 블록을 상기 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 11

제 1 항에 있어서,

상기 프로세서는,

상기 통신부가 제 2 데이터 블록에 대한 업데이트 요청을 제 1 프론트 엔드 노드로부터 수신한 경우, 제 2 메타 데이터 정보를 탐색하고,

상기 제 2 메타 데이터 정보를 참조하여 상기 제 2 데이터 블록에 대한 업데이트를 수행하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 12

제 11 항에 있어서,

상기 프로세서는,

상기 제 2 메타 데이터 정보가 상기 백 엔드 캐시에 존재할 경우, 상기 제 2 메타 데이터 정보에 포함된 상기 제 2 데이터 블록에 대한 제 1 공유 정보를 탐색하고-상기 제 2 데이터 블록에 대한 제 1 공유 정보는 제 1 프론트 엔드 노드에 상기 제 2 데이터 블록이 저장되었다는 정보임-,

상기 통신부를 통하여 상기 제 1 프론트 엔드 노드로부터 상기 제 2 데이터 블록을 수신하고,

상기 제 2 데이터 블록을 상기 백 엔드 캐시에 적재하고,

상기 제 2 데이터 블록에 대한 업데이트를 수행하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 13

제 11 항에 있어서,

상기 프로세서는,

상기 제 2 메타 데이터 정보를 이용하여 상기 제 2 데이터 블록을 저장한 하나 이상의 프론트 엔드 노드를 인식하고,

상기 제 2 데이터 블록을 상기 하나 이상의 프론트 엔드 노드 각각의 프론트 엔드 캐시에서 인밸리데이트(invalidate)하도록 하는 인밸리데이트 신호를 전송하도록 상기 통신부를 제어하고,

인밸리데이트 신호를 전송하도록 통신부를 제어하는 것과 동기적 또는 비동기적으로 상기 제 2 데이터 블록에 대한 상기 업데이트를 수행하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 14

제 13 항에 있어서,

상기 제 2 데이터 블록에 대한 상기 업데이트를 상기 인밸리데이트 신호를 전송하는 것과 비동기적으로 수행하는 경우, 상기 프로세서는,

상기 제 2 데이터 블록에 대한 상기 업데이트를 수행하고,

상기 하나 이상의 프론트 엔드 노드 모두로부터 인밸리데이트 신호에 대한 완료 신호를 수신했는지 여부를 인식하고,

상기 하나 이상의 프론트 엔드 노드 모두로부터 인밸리데이트 신호에 대한 완료 신호를 수신한 경우, 상기 제 2 데이터 블록에 대한 업데이트 완료 신호를 상기 제 1 프론트 엔드 노드로 전송하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 15

제 11 항에 있어서,

상기 프로세서는,

상기 제 2 메타 데이터 정보가 존재하지 않는 경우, 상기 제 2 데이터 블록에 대한 업데이트를 수행하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 16

삭제

청구항 17

제 1 항에 있어서,

상기 프로세서는,

상기 통신부가 제 1 프론트 엔드 노드로부터 제 4 데이터 블록이 상기 제 1 프론트 엔드 노드에서 캐시 아웃되었다는 캐시 아웃 신호를 수신한 경우, 상기 캐시 아웃 신호에 기초하여 상기 제 4 데이터 블록에 대한 제 4 메타 데이터 정보를 업데이트하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 18

제 2 항에 있어서,

상기 프로세서는,

백 엔드 캐시에서 제 5 데이터 블록이 캐시 아웃될 경우, 상기 제 5 데이터 블록에 대한 제 5 메타 데이터 정보를 백 엔드 노드의 메모리에 저장하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 19

제 18 항에 있어서,

상기 프로세서는,

상기 제 5 데이터 블록이 다시 상기 백 엔드 노드로 읽어 들여지면, 상기 제 5 메타 데이터 정보를 다시 로드(load)하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 20

제 19 항에 있어서,

상기 제 5 메타 데이터 정보를 다시 로드(load)하는 경우, 상기 프로세서는,

상기 제 5 데이터 블록이 상기 백 엔드 캐시로 읽어들여졌다고 인식한 경우, 상기 제 5 메타 데이터 정보를 인식하고,

상기 제 5 메타 데이터를 제 5 데이터 블록과 관련된 버퍼 헤더에 기록하는,

클라우드 데이터베이스 시스템에서의 백 엔드 노드.

청구항 21

제 1 항에 있어서,

상기 프로세서는,

제 6 데이터 블록에 대응하는 제 6 메타 데이터 정보를 인식하고,

상기 제 6 메타 데이터 정보에 기초하여 상기 제 6 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 적어도 하나의 프론트 엔드 노드를 인식하고,

상기 적어도 하나의 프론트 엔드 노드로 상기 제 6 데이터 블록에 대한 인밸리데이트 신호를 전송하도록 통신부를 제어하고,

상기 적어도 하나의 프론트 엔드 노드 모두로부터 상기 제 6 데이터 블록에 대한 인밸리데이트 신호에 대한 완

료 신호를 수신한 경우, 상기 제 6 메타 데이터 정보를 상기 백 엔드 캐시에서 삭제하는,

클라우드 데이터 베이스 시스템에서의 백 엔드 노드.

청구항 22

하나 이상의 데이터 블록을 저장하는 프론트 엔드 캐시;

셀렉트 쿼리와 관련된 데이터 블록을 백 엔드 노드(back end node)로부터 수신하는 통신부; 및

데이터 블록을 상기 프론트 엔드 캐시에 저장하기로 결정한 경우, 상기 데이터 블록을 상기 프론트 엔드 캐시에 저장하는 프로세서;

를 포함하고,

상기 프로세서는,

상기 통신부가 상기 백 엔드 노드로부터 상기 프론트 엔드 캐시에서 인밸리데이트(invalidate)될 제 3 데이터 블록에 대한 인밸리데이트 신호를 수신한 경우, 상기 제 3 데이터 블록을 상기 프론트 엔드 캐시로부터 인밸리데이트하고,

상기 프론트 엔드 캐시에 커런트 상태로 저장되어 있는 제 3 데이터 블록을 CR(Consistent Read) 상태로 변경하는,

클라우드 데이터베이스 시스템에서의 프론트 엔드 노드.

청구항 23

제 22 항에 있어서,

스캔 종류, 대상 세그먼트(segment)의 크기, 필터링 정도 또는 접근 빈도 중 적어도 하나를 이용하여 상기 데이터 블록을 상기 프론트 엔드 캐시에 저장할지 여부를 결정하는 옵티마이저(optimizer);

를 더 포함하는,

클라우드 데이터베이스 시스템에서의 프론트 엔드 노드.

청구항 24

제 22 항에 있어서,

상기 프로세서는,

상기 백 엔드 노드로부터 제 1 데이터 블록에 대한 요청 신호를 수신한 경우, 상기 제 1 데이터 블록을 상기 프론트 엔드 캐시에서 탐색하고,

상기 제 1 데이터 블록이 상기 프론트 엔드 캐시에 존재하면, 상기 제 1 데이터 블록을 상기 백 엔드 노드로 전송하는,

클라우드 데이터베이스 시스템에서의 프론트 엔드 노드.

청구항 25

제 22 항에 있어서,

상기 프로세서는,

상기 제 3 데이터 블록을 인밸리데이트하였다는 상기 인밸리데이트 신호에 대한 완료 신호를 상기 백 엔드 노드

로 전송하도록 상기 통신부를 제어하는,

클라우드 데이터베이스 시스템에서의 프론트 엔드 노드.

청구항 26

삭제

청구항 27

제 22 항에 있어서,

상기 프로세서는,

상기 프론트 엔드 캐시에서 캐시 아웃(cache out)될 제 4 데이터 블록을 인식하고,

상기 제 4 데이터 블록이 상기 프론트 엔드 캐시로부터 캐시 아웃되도록 상기 프론트 엔드 캐시를 제어하고,

상기 제 4 데이터 블록이 상기 프론트 엔드 캐시에서 캐시 아웃되었다는 캐시 아웃 신호를 상기 백 엔드 노드로 전송하도록 상기 통신부를 제어하는,

클라우드 데이터베이스 시스템에서의 프론트 엔드 노드.

발명의 설명

기 술 분 야

[0001] 본 개시는 클라우드 시스템에 관한 것으로서, 구체적으로 셀렉트 쿼리의 성능을 향상시키기 위한 시스템에 관한 것이다.

배 경 기 술

[0002] 기존의 데이터베이스 시스템은 사용자로부터 쿼리를 수신하여 쿼리 플랜을 세우는 프론트 엔드(front end) 노드와 프론트 엔드 노드로부터 데이터 블록에 대한 요청을 받으면 실제 디스크 상에서 버퍼 캐시(즉, 백 엔드 캐시)를 통해 데이터 블록을 읽어 프론트 엔드 노드로 전송해주는 백 엔드(back end) 노드로 구성된다.

[0003] 이러한 환경에서, 자주 접근되는 데이터 블록들에 대하여 매번 프론트 엔드 노드가 백 엔드 노드에게 데이터 블록을 요청할 경우, 시스템을 운영하는 데 요구되는 네트워크 비용이 커지게 되어 전체 데이터베이스 시스템의 성능 저하를 야기할 수 있다.

[0004] 따라서, 데이터베이스 시스템 운영상의 네트워크 비용을 절감하기 위한 방법에 관한 수요가 당업계에 존재할 수 있다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) 대한민국 특허 공개 공보 2015-0103477호

(특허문헌 0002) 미국 특허 공개 공보 2007-0143344호

발명의 내용

해결하려는 과제

[0006] 본 개시는 전술한 배경기술에 대응하여 안출된 것으로, 셀렉트 쿼리 처리시 네트워크 비용을 절감하기 위한 시스템을 제공하고자 한다.

[0007] 본 개시의 기술적 과제들은 이상에서 언급한 기술적 과제로 제한되지 않으며, 언급되지 않은 또 다른 기술적 과제들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0008] 클라우드 데이터베이스 시스템에서의 백 엔드 노드로서, 상기 백 엔드 노드는: 통신부; 버퍼 캐시 데이터(buffer cache data) 및 메타 데이터 정보를 저장하는 백 엔드 캐시(back end cache)-상기 버퍼 캐시 데이터 및 상기 메타 데이터 정보는 데이터베이스 시스템에 저장된 데이터 블록에 대응함-; 프로세서;를 포함하고, 상기 메타 데이터 정보는, 상기 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 정보를 포함할 수 있다.

[0009] 또한, 상기 백 엔드 캐시는 상기 버퍼 캐시 데이터에 대응되는 버퍼 헤더를 더 포함하고, 상기 버퍼 헤더는 상기 메타 데이터 정보를 포함할 수 있다.

[0010] 또한, 상기 백 엔드 캐시는 상기 버퍼 캐시 데이터에 대응되는 버퍼 헤더를 더 포함하고, 상기 메타 데이터 정보는 상기 버퍼 캐시 데이터와 동일한 검색 구조를 공유할 수 있다.

[0011] 또한, 상기 버퍼 헤더와 상기 메타 데이터 정보는, 동일한 해시 함수(hash function)을 이용해 상기 백 엔드 캐시에 저장되고, 상기 버퍼 헤더와 상기 메타 데이터 정보와 관련된 해시 함수의 입력 값은 동일하고, 상기 검색 구조 내에서 상기 버퍼 헤더와 상기 메타 데이터 정보가 저장된 곳은 상기 입력 값을 상기 해시 함수에 입력하여 얻어진 결과 값과 연관됨으로써, 상기 검색 구조를 공유할 수 있다.

[0012] 또한, 상기 메타 데이터 정보는, 비트맵(bitmap), 배열로 표현된 노드 ID 정보 또는 공유 메모리 공간 상에 B+ 트리로 표현된 노드 ID 정보를 가리키는 포인터 중 어느 하나를 포함할 수 있다.

[0013] 또한, 상기 프로세서는, 제 1 프론트 엔드 노드로부터 제 1 데이터 블록에 대한 전송 요청을 상기 통신부를 통해 수신한 경우, 제 1 버퍼 캐시 데이터 또는 제 1 메타 데이터 정보 중 적어도 하나를 상기 백 엔드 캐시에서 탐색하고, 상기 제 1 버퍼 캐시 데이터가 탐색된 경우 또는 탐색된 상기 제 1 메타 데이터 정보에 공유 정보가 존재하는 경우 , 상기 제 1 버퍼 캐시 데이터 또는 상기 제 1 메타 데이터 정보 중 적어도 하나에 기초하여 제 1 데이터 블록을 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어하고, 제 1 데이터 블록에 대한 제 1 공유 정보를 제 1 메타 데이터 정보에 저장할 수 있다-상기 제 1 데이터 블록에 대한 제 1 공유 정보는 상기 제 1 프론트 엔드 노드에 상기 제 1 데이터 블록이 저장되었다는 정보임-.

[0014] 또한, 상기 프로세서는, 상기 제 1 메타 데이터 정보에 공유 정보가 존재하는 경우, 상기 제 1 메타 데이터 정보에 포함된 상기 제 1 데이터 블록에 대한 제 2 공유 정보를 탐색하고-상기 제 1 데이터 블록에 대한 제 2 공유 정보는 제 2 프론트 엔드 노드에 상기 제 1 데이터 블록이 저장되었다는 정보임-, 상기 통신부를 통하여 상기 제 2 프론트 엔드 노드로부터 상기 제 1 데이터 블록을 수신하고, 수신한 상기 제 1 데이터 블록을 상기 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어할 수 있다.

[0015] 또한, 상기 제 1 메타 데이터 정보가 상기 백 엔드 캐시에 존재하지 않을 경우, 제 1 메타 데이터 정보를 수용할 수 있는 제 1 데이터 구조체를 생성 또는 로드(load)하고, 상기 제 1 데이터 구조체에 상기 제 1 데이터 블록의 정보를 저장하고, 상기 제 1 데이터 구조체를 상기 제 1 메타 데이터 정보로서 결정할 수 있다.

[0016] 또한, 상기 프로세서는, 상기 제 1 버퍼 캐시 데이터가 상기 백 엔드 캐시에 존재할 경우, 상기 제 1 버퍼 캐시 데이터를 상기 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어할 수 있다.

[0017] 또한, 상기 프로세서는, 상기 제 1 버퍼 캐시 데이터가 존재하지 않고, 탐색된 상기 제 1 메타 데이터에 상기 공유 정보가 존재하지 않을 경우, 디스크(disk)에 상기 제 1 데이터 블록에 대한 요청 신호를 전송하도록 상기 통신부를 제어하고, 상기 디스크로부터 상기 제 1 데이터 블록을 수신하면, 상기 제 1 데이터 블록을 상기 제 1 프론트 엔드 노드로 전송하도록 상기 통신부를 제어할 수 있다.

[0018] 또한, 상기 프로세서는, 상기 통신부가 제 2 데이터 블록에 대한 업데이트 요청을 제 1 프론트 엔드 노드로부터 수신한 경우, 제 2 메타 데이터 정보를 탐색하고, 상기 제 2 메타 데이터 정보를 참조하여 상기 제 2 데이터 블록에 대한 업데이트를 수행할 수 있다.

[0019] 또한, 상기 프로세서는, 상기 제 2 메타 데이터 정보가 상기 백 엔드 캐시에 존재할 경우, 상기 제 2 메타 데이터 정보에 포함된 상기 제 2 데이터 블록에 대한 제 1 공유 정보를 탐색하고-상기 제 2 데이터 블록에 대한 제 1 공유 정보는 제 1 프론트 엔드 노드에 상기 제 2 데이터 블록이 저장되었다는 정보임-, 상기 통신부를 통하여

상기 제 1 프론트 엔드 노드로부터 상기 제 2 데이터 블록을 수신하고, 상기 제 2 데이터 블록을 상기 백 엔드 캐시에 적재하고, 상기 제 2 데이터 블록에 대한 업데이트를 수행할 수 있다.

[0020] 또한, 상기 프로세서는 상기 제 2 메타 데이터 정보를 이용하여 상기 제 2 데이터 블록을 저장한 하나 이상의 프론트 엔드 노드를 인식하고, 상기 인밸리데이트 신호를 전송하도록 통신부를 제어하는 것과 동기적 또는 비동기적으로 상기 제 2 데이터 블록에 대한 상기 업데이트를 수행하고, 상기 제 2 데이터 블록을 상기 하나 이상의 프론트 엔드 노드 각각의 프론트 엔드 캐시에서 인밸리데이트(invalidate)하도록 하는 인밸리데이트 신호를 전송하도록 상기 통신부를 제어할 수 있다.

[0021] 또한, 상기 제 2 데이터 블록에 대한 상기 업데이트를 상기 인밸리데이트 신호를 전송하는 것과 비동기적으로 수행하는 경우, 상기 프로세서는, 상기 제 2 데이터 블록에 대한 상기 업데이트를 수행하고, 상기 하나 이상의 프론트 엔드 노드 모두로부터 인밸리데이트 신호에 대한 완료 신호를 수신했는지 여부를 인식하고, 상기 하나 이상의 프론트 엔드 노드 모두로부터 인밸리데이트 신호에 대한 완료 신호를 수신한 경우, 상기 제 2 데이터 블록에 대한 업데이트 완료 신호를 상기 제 1 프론트 엔드 노드로 전송할 수 있다.

[0022] 또한, 상기 프로세서는, 상기 제 2 메타 데이터 정보가 존재하지 않는 경우, 상기 제 2 데이터 블록에 대한 업데이트를 수행할 수 있다.

[0023] 또한, 상기 프로세서는, 제 1 프론트 엔드 노드로부터 제 3 데이터 블록이 상기 제 1 프론트 엔드 노드에서 인밸리데이트되었다는 신호를 수신한 경우, 제 3 메타 데이터 정보를 탐색하고, 상기 제 3 메타 데이터 정보에서 상기 제 3 데이터 블록에 대한 제 1 공유 정보를 삭제할 수 있다-상기 제 3 데이터 블록에 대한 제 1 공유 정보는 상기 제 1 프론트 엔드 노드에 상기 제 3 데이터 블록이 저장되어있다는 정보임-.

[0024] 또한, 상기 프로세서는, 상기 통신부가 제 1 프론트 엔드 노드로부터 제 4 데이터 블록이 상기 제 1 프론트 엔드 노드에서 캐시 아웃되었다는 캐시 아웃 신호를 수신한 경우, 상기 캐시 아웃 신호에 기초하여 상기 제 4 데이터 블록에 대한 제 4 메타 데이터 정보를 업데이트할 수 있다.

[0025] 또한, 상기 프로세서는, 백 엔드 캐시에서 제 5 데이터 블록이 캐시 아웃될 경우, 상기 제 5 데이터 블록에 대한 제 5 메타 데이터 정보를 백 엔드 노드의 메모리에 저장할 수 있다.

[0026] 또한, 상기 프로세서는, 상기 제 5 데이터 블록이 다시 상기 백 엔드 노드로 읽어 들여지면, 상기 제 5 메타 데이터 정보를 다시 로드(load)할 수 있다.

[0027] 또한, 상기 제 5 메타 데이터 정보를 다시 로드(load)하는 경우, 상기 프로세서는, 상기 제 5 데이터 블록이 상기 백 엔드 캐시로 읽어들여졌다고 인식한 경우, 상기 제 5 메타 데이터 정보를 인식하고, 상기 제 5 메타 데이터를 상기 제 5 데이터 블록과 관련된 버퍼 헤더에 기록할 수 있다.

[0028] 또한, 제 1 항에 있어서, 상기 프로세서는, 제 6 데이터 블록에 대응하는 제 6 메타 데이터 정보를 인식하고, 제 6 메타 데이터 정보에 기초하여 상기 제 6 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 적어도 하나의 프론트 엔드 노드를 인식하고, 상기 적어도 하나의 프론트 엔드 노드로 상기 제 6 데이터 블록에 대한 인밸리데이트 신호를 전송하도록 통신부를 제어하고, 상기 적어도 하나의 프론트 엔드 노드 모두로부터 상기 제 6 데이터 블록에 대한 인밸리데이트 신호에 대한 완료 신호를 수신한 경우, 상기 제 6 메타 데이터 정보를 상기 백 엔드 캐시에서 삭제할 수 있다.

[0030] 전술한 과제를 해결하기 위한 클라우드 데이터베이스 시스템에서의 프론트 엔드 노드가 개시된다. 상기 프론트 엔드 노드는: 하나 이상의 데이터 블록을 저장하는 프론트 엔드 캐시; 셀렉트 쿼리와 관련된 데이터 블록을 백 엔드 노드(back end node)로부터 수신하는 통신부; 및 데이터 블록을 상기 프론트 엔드 캐시에 저장하기로 결정한 경우, 상기 데이터 블록을 상기 프론트 엔드 캐시에 저장하는 프로세서;를 포함할 수 있다.

[0031] 또한, 스캔 종류, 대상 세그먼트(segment)의 크기, 필터링 정도 또는 접근 빈도 중 적어도 하나를 이용하여 상기 데이터 블록을 상기 프론트 엔드 캐시에 저장할지 여부를 결정하는 옵티마이저(optimizer); 를 더 포함할 수 있다.

[0032] 또한, 상기 프로세서는, 상기 백 엔드 노드로부터 제 1 데이터 블록에 대한 요청 신호를 수신한 경우, 상기 제 1 데이터 블록을 상기 프론트 엔드 캐시에서 탐색하고, 상기 제 1 데이터 블록이 상기 프론트 엔드 캐시에 존재하면, 상기 제 1 데이터 블록을 상기 백 엔드 노드로 전송할 수 있다.

[0033] 또한, 상기 프로세서는, 상기 통신부가 상기 백 엔드 노드로부터 상기 프론트 엔드 캐시에서 인밸리데이트

(invalidate)될 제 3 데이터 블록에 대한 인밸리데이트 신호를 수신한 경우, 상기 제 3 데이터 블록을 상기 프론트 엔드 캐시로부터 인밸리데이트하고,상기 제 3 데이터 블록을 인밸리데이트하였다는 상기 인밸리데이트 신호에 대한 완료 신호를 상기 백 엔드 노드로 전송하도록 상기 통신부를 제어할 수 있다.

[0034] 또한, 상기 프로세서는, 상기 제 3 데이터 블록을 상기 프론트 엔드 캐시로부터 인밸리데이트하는 경우, 상기 프론트 엔드 캐시에 커런트 상태로 저장되어 있는 제 3 데이터 블록을 CR(Consistent Read) 상태로 변경할 수 있다.

[0035] 또한, 상기 프로세서는, 상기 프론트 엔드 캐시에서 캐시 아웃(cache out)될 제 4 데이터 블록을 인식하고, 상기 제 4 데이터 블록이 상기 프론트 엔드 캐시로부터 캐시 아웃되도록 상기 프론트 엔드 캐시를 제어하고, 상기 제 4 데이터 블록이 상기 프론트 엔드 캐시에서 캐시 아웃되었다는 캐시 아웃 신호를 상기 백 엔드 노드로 전송하도록 상기 통신부를 제어할 수 있다.

발명의 효과

[0036] 본 개시에 따라 셀렉트 쿼리의 성능이 향상될 수 있다.

[0037] 본 개시에서 얻을 수 있는 효과는 이상에서 언급한 효과로 제한되지 않으며, 언급하지 않은 또 다른 효과들은 아래의 기재로부터 본 개시가 속하는 기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0038] 다양한 양상들이 이제 도면들을 참조로 기재되며, 여기서 유사한 참조 번호들은 총괄적으로 유사한 구성요소들을 지칭하는데 이용된다. 이하의 실시예에서, 설명 목적을 위해, 다수의 특정 세부사항들이 하나 이상의 양상들의 총체적 이해를 제공하기 위해 제시된다. 그러나, 그러한 양상(들)이 이러한 구체적인 세부사항들 없이 실시될 수 있음은 명백할 것이다.

도 1은 본 발명의 몇몇 실시예에 따른 프론트 엔드 노드 및 백 엔드 노드를 포함하는 예시적인 클라우드 데이터베이스 시스템에 대한 개략도를 도시한다.

도 2는 본 개시의 몇몇 실시예에 따른 프론트 엔드 노드의 구성을 나타낸 블록도이다.

도 3a는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 백 엔드 캐시에 버퍼 헤더, 메타 데이터 정보 및 버퍼 캐시 데이터가 저장된 양상을 나타낸 도면이다.

도 3b는 검색 구조를 공유하기 위한 구체적인 방법의 일례를 나타낸 도면이다.

도 4는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록을 프론트 엔드 노드로 전달하는 것을 나타낸 순서도이다.

도 5는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록에 대한 업데이트를 수행하는 과정을 나타낸 순서도이다.

도 6은 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 프론트 노드에서 인밸리데이트 된 데이터 블록의 정보를 업데이트하는 과정을 나타낸 순서도이다.

도 7은 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록이 프론트 엔드 캐시에서 캐시 아웃된 경우 이를 업데이트 하는 과정을 나타낸 순서도이다.

도 8은 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 캐시 아웃된 데이터 블록이 다시 백 엔드 캐시에 저장된 경우 메타 데이터 정보를 로드하는 과정을 나타낸 순서도이다.

도 9는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록의 메타 데이터 정보를 백 엔드 캐시로부터 삭제하는 과정을 나타낸 순서도이다.

도 10은 본 개시의 몇몇 실시예에 따른 프론트 엔드의 프로세서가 프론트 엔드 캐시에 데이터 블록을 저장하는 과정을 나타낸 순서도이다.

도 11은 본 개시의 몇몇 실시예들이 구현될 수 있는 예시적인 컴퓨팅 환경에 대한 간략하고 일반적인 개략도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0039] 다양한 실시예들 및/또는 양상들이 이제 도면들을 참조하여 개시된다. 하기 설명에서는 설명을 목적으로, 하나 이상의 양상들의 전반적 이해를 돕기 위해 다수의 구체적인 세부사항들이 개시된다. 그러나, 이러한 양상(들)은 이러한 구체적인 세부사항들 없이도 실행될 수 있다는 점 또한 본 개시의 기술 분야에서 통상의 지식을 가진 자에게 감지될 수 있을 것이다. 이후의 기재 및 첨부된 도면들은 하나 이상의 양상들의 특정한 예시적인 양상들을 상세하게 기술한다. 하지만, 이러한 양상들은 예시적인 것이고 다양한 양상들의 원리들에서의 다양한 방법들 중 일부가 이용될 수 있으며, 기술되는 설명들은 그러한 양상들 및 그들의 균등물들을 모두 포함하고자 하는 의도이다. 구체적으로, 본 명세서에서 사용되는 "실시예", "예", "양상", "예시" 등은 기술되는 임의의 양상 또는 설계가 다른 양상 또는 설계들보다 양호하다거나, 이점이 있는 것으로 해석되지 않을 수도 있다.

[0040] 이하, 도면 부호에 관계없이 동일하거나 유사한 구성 요소는 동일한 참조 번호를 부여하고 이에 대한 중복되는 설명은 생략한다. 또한, 본 명세서에 개시된 실시예를 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 본 명세서에 개시된 실시예의 요지를 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다. 또한, 첨부된 도면은 본 명세서에 개시된 실시예를 쉽게 이해할 수 있도록 하기 위한 것일 뿐, 첨부된 도면에 의해 본 명세서에 개시된 기술적 사상이 제한되지 않는다.

[0041] 본 명세서에서 사용된 용어는 실시예들을 설명하기 위한 것이며 본 발명을 제한하고자 하는 것은 아니다. 본 명세서에서, 단수형은 문구에서 특별히 언급하지 않는 한 복수형도 포함한다. 명세서에서 사용되는 "포함한다(comprises)" 및/또는 "포함하는(comprising)"은 언급된 구성요소 외에 하나 이상의 다른 구성요소의 존재 또는 추가를 배제하지 않는다.

[0042] 비록 제1, 제2 등이 다양한 소자나 구성요소들을 서술하기 위해서 사용되나, 이들 소자나 구성요소들은 이들 용어에 의해 제한되지 않음은 물론이다. 이들 용어들은 단지 하나의 소자나 구성요소를 다른 소자나 구성요소와 구별하기 위하여 사용하는 것이다. 따라서, 이하에서 언급되는 제1 소자나 구성요소는 본 발명의 기술적 사상 내에서 제2 소자나 구성요소 일 수도 있음은 물론이다.

[0043] 다른 정의가 없다면, 본 명세서에서 사용되는 모든 용어(기술 및 과학적 용어를 포함)는 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 공통적으로 이해될 수 있는 의미로 사용될 수 있을 것이다. 또 일반적으로 사용되는 사전에 정의되어 있는 용어들은 명백하게 특별히 정의되어 있지 않는 한 이상적으로 또는 과도하게 해석되지 않는다.

[0044] 더불어, 용어 "또는"은 배타적 "또는"이 아니라 내포적 "또는"을 의미하는 것으로 의도된다. 즉, 달리 특정되지 않거나 문맥상 명확하지 않은 경우에, "X는 A 또는 B를 이용한다"는 자연적인 내포적 치환 중 하나를 의미하는 것으로 의도된다. 즉, X가 A를 이용하거나; X가 B를 이용하거나; 또는 X가 A 및 B 모두를 이용하는 경우, "X는 A 또는 B를 이용한다"가 이들 경우들 어느 것으로도 적용될 수 있다. 또한, 본 명세서에 사용된 "및/또는"이라는 용어는 열거된 관련 아이템들 중 하나 이상의 아이템의 가능한 모든 조합을 지칭하고 포함하는 것으로 이해되어야 한다.

[0045] 더불어, 본 명세서에서 사용되는 용어 "정보" 및 "데이터"는 종종 서로 상호교환 가능하도록 사용될 수 있다.

[0046] 이하, 도면 부호에 관계없이 동일하거나 유사한 구성 요소는 동일한 참조 번호를 부여하고 이에 대한 중복되는 설명은 생략한다. 또한, 본 명세서에 개시된 실시예를 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 본 명세서에 개시된 실시예의 요지를 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다. 또한, 첨부된 도면은 본 명세서에 개시된 실시예를 쉽게 이해할 수 있도록 하기 위한 것일 뿐, 첨부된 도면에 의해 본 명세서에 개시된 기술적 사상이 제한되지 않는다.

[0047] 비록 제1, 제2 등이 다양한 소자나 구성요소들을 서술하기 위해서 사용되나, 이들 소자나 구성요소들은 이들 용어에 의해 제한되지 않음은 물론이다. 이들 용어들은 단지 하나의 소자나 구성요소를 다른 소자나 구성요소와 구별하기 위하여 사용하는 것이다. 따라서, 이하에서 언급되는 제1 소자나 구성요소는 본 개시의 기술적 사상 내에서 제2 소자나 구성요소 일 수도 있음은 물론이다.

[0048] 다른 정의가 없다면, 본 명세서에서 사용되는 모든 용어(기술 및 과학적 용어를 포함)는 본 개시가 속하는 기술 분야에서 통상의 지식을 가진 자에게 공통적으로 이해될 수 있는 의미로 사용될 수 있을 것이다. 또 일반적으로 사용되는 사전에 정의되어 있는 용어들은 명백하게 특별히 정의되어 있지 않는 한 이상적으로 또는 과도하게 해석되지 않는다.

[0049] 본 명세서에서의 컴퓨터 판독가능 매체는 컴퓨터 시스템에 의해서 판독될 수 있도록 프로그램 및 데이터가 저장되는 모든 종류의 저장 매체를 포함할 수 있다. 본 개시내용에서의 컴퓨터 판독가능 매체는, 컴퓨터 판독가능 저장 매체 및 컴퓨터 판독가능 전송 매체를 포함할 수 있다. 본 발명의 일 양상에 따르면, 컴퓨터 판독가능 저장 매체는: ROM(판독 전용 메모리), RAM(랜덤 액세스 메모리), CD(컴팩트 디스크)-ROM, DVD(디지털 비디오 디스크)-ROM, 자기 테이프, 플로피 디스크, 광 데이터 저장장치 등을 포함할 수 있다. 또한, 컴퓨터 판독가능 전송 매체는 캐리어 웨이브(예컨대, 인터넷을 통한 전송)의 형태로 구현되는 임의의 전송 가능한 형태의 매체를 포함할 수 있다. 추가적으로, 이러한 컴퓨터 판독가능 매체는 네트워크로 연결된 시스템에 분산되어, 분산 방식으로 컴퓨터가 판독가능한 코드들 및/또는 명령들을 저장할 수도 있다.

[0050] 본 발명의 실시를 위한 구체적인 내용을 설명하기에 앞서, 본 발명의 기술적 요지와 직접적 관련이 없는 구성에 대해서는 본 발명의 기술적 요지를 흐뜨리지 않는 범위 내에서 생략하였음에 유의하여야 할 것이다. 또한, 본 명세서 및 청구범위에 사용된 용어 또는 단어는 발명자가 자신의 발명을 최선의 방법으로 설명하기 위해 적절한 용어의 개념을 정의할 수 있다는 원칙에 입각하여 본 발명의 기술적 사상에 부합하는 의미와 개념으로 해석되어야 할 것이다.

[0051] 본 개시내용에서, 쿼리(query)는 백 엔드 노드에서의 처리를 요청하는 임의의 요청 또는 명령을 의미하며, 예를 들어, DML(Data Manipulation Language), DDL(Data Definition Language) 및/또는 PL/SQL 등을 포함할 수 있다. 또한, 본 개시내용에서의 쿼리는 사용자/개발자 등으로부터 발행되는 임의의 요청을 의미할 수 있다. 또한, 쿼리는 프론트 엔드 노드 및/또는 백 엔드 노드에 인입되고 프론트 엔드 노드 및/또는 백 엔드 노드에서 처리되는 임의의 요청을 의미할 수 있다.

[0053] 도 1은 본 발명의 몇몇 실시예에 따른 프론트 엔드 노드 및 백 엔드 노드를 포함하는 예시적인 클라우드 데이터베이스 시스템에 대한 개략도를 도시한다.

[0054] 본 개시에 따른 클라우드 컴퓨팅 환경에서의 프론트 엔드 노드 및 백 엔드 노드는 클라우드 컴퓨팅 환경 하에서의 데이터베이스 시스템을 구성한다. 클라우드 컴퓨팅은 정보가 백 엔드 또는 디스크에 영구적으로 저장되고, 데스크톱, 태블릿 컴퓨터, 노트북, 넷북, 스마트폰 등의 IT 기기 등과 같은 프론트 엔드에는 일시적으로 보관되는 컴퓨터 환경을 뜻한다. 즉 이용자의 모든 정보를 서버에 저장하고, 이 정보를 각종 IT 기기를 통하여 언제 어디서든 이용할 수 있다는 개념이다.

[0055] 다시 말하면 서로 다른 물리적인 위치에 존재하는 컴퓨팅 자원(즉, 프론트 엔드 노드(100) 및 백 엔드 노드(200))를 가상화 기술로 통합해 제공하는 기술을 말한다.

[0056] 이러한 클라우드 컴퓨팅 환경 하에서의 데이터베이스 시스템은 사용자로부터 쿼리를 수신하여 쿼리 플랜을 세우는 프론트 엔드(front end) 노드(100)와 프론트 엔드 노드로부터 데이터 블록에 대한 요청을 받으면 실제 디스크 상에서 버퍼 캐시(즉, 백 엔드 캐시)를 통해 데이터 블록을 읽어 프론트 엔드 노드로 전송해주는 백 엔드(back end) 노드(200)로 구성된다.

[0057] 일반적인 클라우드 컴퓨팅 환경 하에서의 데이터베이스 시스템은 프론트 엔드에 별도의 캐시를 구비하고 있지 않다. 따라서, 자주 접근되는 데이터 블록들에 대하여 매번 프론트 엔드 노드가 백 엔드 노드에게 데이터 블록을 요청할 경우, 시스템을 운영하는 데 요구되는 네트워크 비용이 커지게 되어 전체 데이터베이스 시스템의 성능 저하를 야기할 수 있다.

[0058] 이러한 문제를 해결하기 위하여, 본 개시는 프론트 엔드 노드(100)에 프론트 엔드 캐시(140)를 추가함으로써 데이터베이스 시스템 운영에 요구되는 네트워크 비용을 절감하고, 이에 따라 전체 시스템의 성능 향상을 꾀할 수 있다. 이에 관하여는 이하 도 2에서 자세히 후술한다.

[0059] 도 1에서 도시되는 바와 같이, 시스템은 프론트 엔드 노드(100) 및 백 엔드 노드(200)를 포함할 수 있다. 프론트 엔드 노드(100) 및 백 엔드 노드(200)는 임의의 네트워크(미도시)에 의해 서로 연결될 수 있다.

[0060] 도 1에서 도시되는 바와 같이, 프론트 엔드 노드(100)는 네트워크를 통하여 통신하기 위한 매커니즘을 갖는 클라우드 데이터베이스 시스템에서의 임의의 형태의 노드(들)를 의미할 수 있다. 예를 들어, 프론트 엔드 노드(100)는 PC, 랩탑 컴퓨터, 워크스테이션, 단말 및/또는 네트워크 접속성을 갖는 임의의 전자 디바이스를 포함할 수 있다. 또한, 프론트 엔드 노드(100)는 에이전트(Agent), API(Application Programming Interface) 및 플러그-인(Plug-in) 중 적어도 하나에 의해 구현되는 임의의 서버를 포함할 수도 있다. 또한, 프론트 엔드 노드(100)는 애플리케이션 소스 및/또는 클라이언트 애플리케이션을 포함할 수 있다.

[0061] 프론트 엔드 노드(100)는 프로세서 및 메모리를 포함하여, 임의의 데이터를 처리 및 저장할 수 있는 임의의 엔티티일 수 있다. 또한, 도 1에서의 프론트 엔드 노드(100)는 백 엔드 노드(200)를 사용하거나 백 엔드 노드(200)와 통신하는 사용자와 관련될 수 있다. 이러한 예시에서, 프론트 엔드 노드(100)는 백 엔드 노드(200)로 쿼리를 발행할 수 있다. 일 예시에서, 프론트 엔드 노드(100)가 백 엔드 노드(200)로 쿼리를 발생시키는 경우, 백 엔드 노드(200) 내에서 해당 쿼리는 트랜잭션의 형태로 처리될 수 있다. 즉, 프론트 엔드 노드(100)는 백 엔드 노드(200)에서 트랜잭션이 발생 및 처리되도록 할 수 있다. 프론트 엔드 노드(100)는 개발자 등에 의해 프로그래밍 언어로 작성된 애플리케이션 소스를 수신할 수도 있다. 또한, 예를 들어, 프론트 엔드 노드(100)는 애플리케이션 소스를 컴파일링하여 클라이언트 애플리케이션을 생성할 수 있다. 예를 들어, 생성된 클라이언트 애플리케이션은 백 엔드 노드(200)로 전달된 후 최적화되어 실행될 수 있다.

[0062] 백 엔드 노드(200)는, 예를 들어, 마이크로프로세서, 메인프레임 컴퓨터, 디지털 프로세서, 휴대용 디바이스 및 디바이스 제어기 등과 같은 임의의 타입의 컴퓨터 시스템 또는 컴퓨터 디바이스를 포함할 수 있다. 이러한 백 엔드 노드(200)는 데이터베이스 시스템(미도시), 프로세서(210), 통신부(220), 메모리(230) 및 백 엔드 캐시(240)를 포함할 수 있다. 또한 백 엔드 노드(200)는 외부 디스크(300)에 대한 디스크 입출력을 통해 데이터를 전송 및 수신할 수 있다. 도 1에서는 1개의 백 엔드 노드 및 3개의 프론트 엔드 노드를 예시적으로 도시하고 있으나, 이보다 많은 백 엔드 노드들(관리 장치들) 및 프론트 엔드 노드들 또한 본 발명의 범위에 포함될 수 있다는 점이 당해 출원 분야에 있어서 통상의 지식을 가진 자에게 명백할 것이다.

[0063] 도 1에서 도시되는 바와 같이, 백 엔드 노드(200)는 버퍼 캐시를 포함하는 하나 이상의 메모리(230)를 포함할 수 있다. 또한, 도 1에서 도시되는 바와 같이, 백 엔드 노드(200)는 하나 이상의 프로세서(210)를 포함할 수 있다. 따라서, 클라우드 데이터베이스 시스템은 상기 메모리(230) 상에서 상기 프로세서(210)에 의하여 동작될 수 있다.

[0064] 프로세서(210)는 하나 이상의 코어로 구성될 수 있으며, 컴퓨팅 장치의 중앙 처리 장치(CPU: central processing unit), 범용 그래픽 처리 장치 (GPGPU: general purpose graphics processing unit), 텐서 처리 장치(TPU: tensor processing unit) 등의 임의의 형태의 프로세서를 포함할 수 있다. 프로세서(210)는 메모리(230)에 저장된 컴퓨터 프로그램을 판독하여 본 개시의 일 실시예에 따른 셀렉트 쿼리 성능 향상 방법을 수행할 수 있다.

[0065] 프로세서(210)는 프론트 엔드 노드(100)로부터 백 엔드 노드(200)로 인입되는 쿼리와 관련되어 저장되는 임의의 데이터 및 로그 정보를 임시적으로 또는 영구적으로 저장할 것을 결정할 수 있다. 프로세서(210)는 데이터 테이블 및/또는 인덱스 테이블 등을 저장할 것을 결정할 수 있다. 프로세서(210)는 저장되는 데이터 및/또는 로그 정보의 메모리(230) 상에서의 저장 위치 혹은 디스크(300) 상에서의 저장 위치를 결정할 수 있다.

[0066] 메모리(230)는 프로세서(210)의 동작을 위한 프로그램을 저장할 수 있고, 입/출력되는 데이터들(예를 들어, 업데이트 요청, 데이터 블록, 버퍼 헤더(400) 등)을 임시 또는 영구 저장할 수도 있다. 메모리(230)는 플래시 메모리 타입(flash memory type), 하드디스크 타입(hard disk type), 멀티미디어 카드 마이크로 타입(multimedia card micro type), 카드 타입의 메모리(예를 들어 SD 또는 XD 메모리 등), 램(Random Access Memory, RAM), SRAM(Static Random Access Memory), 롬(Read-Only Memory, ROM), EEPROM(Electrically Erasable Programmable Read-Only Memory), PROM(Programmable Read-Only Memory), 자기 메모리, 자기 디스크, 광디스크 중 적 어도 하나의 타입의 저장매체를 포함할 수 있다. 이러한 메모리(230)는 프로세서(210)의 제어에 의하여 동작 될 수 있다.

[0067] 본 개시의 몇몇 실시예에 따른 메모리(230)는 데이터 값을 포함하는 데이터 블록을 저장할 수 있다. 상기 데이터 블록은 데이터 값을 포함할 수 있으며, 본 개시내용의 일 실시예에서 상기 데이터 블록의 데이터 값은 메모리(230)로부터 디스크(300)에 기록될 수 있다.

[0068] 추가적인 양상에서, 본 개시의 몇몇 실시예에 따른 백 엔드 노드(200)는 백 엔드 캐시(240)를 포함할 수 있다.

[0069] 도 2는 본 개시의 몇몇 실시예에 따른 프론트 엔드 노드의 구성을 나타낸 블록도이다.

[0070] 프론트 엔드 노드(100)는, 예를 들어, 마이크로프로세서, 메인프레임 컴퓨터, 디지털 프로세서, 휴대용 디바이스 및 디바이스 제어기 등과 같은 임의의 타입의 컴퓨터 시스템 또는 컴퓨터 디바이스를 포함할 수 있다.

[0071] 프론트 엔드 노드(100)는 데이터베이스 시스템(미도시), 프로세서(110), 통신부(120), 옵티마이저(130) 및 프론트 엔드 캐시(140)를 포함할 수 있다. 다만, 상술한 구성 요소들은 프론트 엔드 노드(100)를 구현하는데 있어서

필수적인 것은 아니어서, 프론트 엔드 노드(100)는 위에서 열거된 구성요소들 보다 많거나, 또는 적은 구성요소들을 가질 수 있다. 여기서, 각각의 구성 요소들은 별개의 칩이나 모듈이나 장치로 구성될 수 있고, 하나의 장치 내에 포함될 수도 있다.

[0073] 도 2에서 도시되지 않았지만, 프론트 엔드 노드(100)는 하나 이상의 메모리(미도시)를 포함할 수 있다. 또한, 도 2에서 도시되는 바와 같이, 프론트 엔드 노드(100)는 하나 이상의 프로세서(110)를 포함할 수 있다. 따라서, 클라우드 데이터베이스 시스템은 상기 메모리(미도시) 상에서 상기 프로세서(110)에 의하여 동작될 수 있다.

[0074] 프로세서(110)는 하나 이상의 코어로 구성될 수 있으며, 컴퓨팅 장치의 중앙 처리 장치(CPU: central processing unit), 범용 그래픽 처리 장치 (GPGPU: general purpose graphics processing unit), 텐서 처리 장치(TPU: tensor processing unit) 등의 임의의 형태의 프로세서를 포함할 수 있다. 프로세서(110)는 메모리에 저장된 컴퓨터 프로그램을 판독하여 본 개시의 일 실시예에 따른 셀렉트 쿼리 성능 향상 방법을 수행할 수 있다.

[0075] 프로세서(110)는 외부에서 프론트 엔드 노드(100)로 인입되는 쿼리와 관련되어 저장되는 임의의 데이터 및 로그 정보를 임시적으로 또는 영구적으로 저장할 것을 결정할 수 있다. 프로세서(110)는 데이터 테이블 및/또는 인덱스 테이블 등을 저장할 것을 결정할 수 있다. 프로세서(110)는 저장되는 데이터 및/또는 로그 정보의 메모리 상에서의 저장 위치 혹은 디스크(300) 상에서의 저장 위치를 결정할 수 있다.

[0076] 한편, 프로세서(110)는 프론트 엔드 노드(100)의 전반적인 동작을 제어할 수 있다.

[0077] 본 개시에 있어 옵티마이저(130)는 사용자가 요청한 SQL문을 가장 효율적이고 빠르게 수행할 수 있는 최적(최저 비용)의 처리 경로를 선택해주는 구성요소 또는 모듈로 정의될 수 있다.

[0078] 본 개시에 따른 몇몇 실시예에서 옵티마이저(130)는 스캔 종류, 대상 세그먼트(segment)의 크기, 필터링 정도 또는 접근 빈도 중 적어도 하나를 이용하여 상기 데이터 블록을 상기 프론트 엔드 캐시에 저장할지 여부를 결정할 수 있다.

[0079] 옵티마이저(130)는 본 개시에 따른 컴퓨터 판독가능 저장매체에 저장된 프로그램의 일부일 수도 있고, 해당 프로그램이 구동되는 프로세서를 의미할 수도 있다.

[0080] 옵티마이저(130)가 결정한 셀렉트 쿼리와 관련된 스캔의 종류는, 예를 들면, 테이블 풀 스캔(Table full scan) 또는 인덱스 스캔(Index scan)일 수 있다. 다만 이에 한정되는 것은 아니다.

[0081] 테이블 풀 스캔은 테이블에 존재하는 모든 로우를 읽어내는 방법으로 정의될 수 있다. 반면 인덱스 스캔은 특정 테이블과 관련된 인덱스를 이용하여 데이터를 읽어내는 방법으로 정의될 수 있다.

[0082] 인덱스 스캔이 수행된 경우에는 프론트 엔드 캐시(140)에 저장된 특정 데이터 블록이 자주 접근될 확률이 높고 접근하는 데이터 블록의 개수도 적다. 따라서 프론트 엔드 캐시(140)를 탐색하면 셀렉트 쿼리와 관련된 제 1 데이터 블록이 탐색될 가능성이 높으므로, 프론트 엔드 캐시(140)를 사용함에 의한 성능 향상을 기대할 수 있다. 즉, 옵티마이저(130)는 셀렉트 쿼리와 관련된 스캔의 종류가 인덱스 스캔이라고 인식한 경우, 셀렉트 쿼리와 관련된 데이터 블록을 프론트 엔드 캐시(140)에서 저장하기로 결정할 수도 있다. 다만 이에 한정되는 것은 아니다.

[0083] 그러나 테이블 풀 스캔이 사용된 경우에는 접근하는 데이터 블록의 개수가 많고, 스캔에 의하여 얻은 데이터 블록에 대한 필터링(필터링은 사용자가 정의한 조건을 만족하는 데이터만 선택하는 기능을 의미함)이 많이 일어나 실제 사용하는 데이터 블록이 적은 경우, 프론트 엔드 캐시(140)를 사용함으로써 얻을 수 있는 이익이 크지 않다. 즉, 테이블 풀 스캔이 사용된다면, 작은 테이블에 대하여 결과 필터링이 적은 경우에만 프론트 엔드 캐시(140)에 대한 저장을 수행하는 것이 적절할 수 있다. 즉, 옵티마이저(130)는 셀렉트 쿼리와 관련된 스캔의 종류가 테이블 풀 스캔이고 작은 테이블에 대하여 결과 필터링이 적다고 인식한 경우, 셀렉트 쿼리와 관련된 데이터 블록을 프론트 엔드 캐시(140)에 저장하기로 결정할 수 있다. 다만 이에 한정되는 것은 아니다.

[0084] 본 개시에 따른 세그먼트는, 디스크 저장공간을 사용하는 객체를 의미할 수 있다. 즉, 저장공간을 가지고 있는 임의의 오브젝트(object)는 세그먼트일 수 있다.

[0085] 세그먼트의 크기가 큰 경우, 셀렉트 쿼리와 관련된 데이터 블록이 아닌 잉여 블록이 많이 존재할 수 있다. 따라서 당해 세그먼트 전체를 프론트 엔드 캐시에 저장하는 것은 캐시 공간을 비효율적으로 사용할 수 있다. 따라서, 일반적으로, 옵티마이저(130)는 세그먼트의 크기가 큰 경우 프론트 엔드 캐시(140)에 당해 세그먼트를

저장하지 않도록 결정할 수 있다.

[0086] 나아가, 본 개시에 따른 접근 빈도는 셀렉트 쿼리와 관련된 임의의 데이터 블록에 대한 단위 시간 당 접근 요청 횟수를 의미할 수 있다.

[0087] 예를 들어, 옵티마이저(130)는 셀렉트 쿼리와 관련된 임의의 데이터 블록에 대한 과거의 접근 빈도가 높을수록 프론트 엔드 캐시(140)에 당해 데이터 블록을 저장하도록 설정될 수 있다. 반대로, 옵티마이저(130)는 셀렉트 쿼리와 관련된 임의의 데이터 블록에 대한 과거의 접근 빈도가 낮을수록 프론트 엔드 캐시(140)에 당해 데이터 블록을 저장하도록 설정될 수 있다.

[0088] 본 개시의 몇몇 실시예에 따른 프론트 엔드 캐시(140)는 프로세서(110)에 의하여 프론트 엔드 노드의 메모리로부터 할당될 수 있다(즉, 프론트 엔드 캐시(140)는 메모리의 일부일 수 있다).

[0089] 본 개시의 몇몇 실시예에 따르면, 옵티마이저(130)는 스캔(scan)의 종류(즉, 테이블 풀 스캔인지, 인덱스 스캔인지 등), 대상 세그먼트의 크기, 필터링 정도, 접근 빈도 등을 고려해 셀렉트 쿼리와 관련된 데이터 블록을 반환하기 위해 프론트 엔드 캐시를 먼저 탐색할 것인지 여부를 결정할 수 있다.

[0090] 프론트 엔드 캐시(140)는 백 엔드 노드(200)로부터 입/출력되는 데이터들(예를 들어, 업데이트 요청, 데이터 블록, 버퍼 헤더(400) 등)을 임시적으로 저장할 수 있다. 프론트 엔드 캐시(140)는 플래시 메모리 타입(flash memory type), 하드디스크 타입(hard disk type), 멀티미디어 카드 마이크로 타입(multimedia card micro type), 카드 타입의 메모리(예를 들어 SD 또는 XD 메모리 등), 램(Random Access Memory, RAM), SRAM(Static Random Access Memory), 롬(Read-Only Memory, ROM), EEPROM(Electrically Erasable Programmable Read-Only Memory), PROM(Programmable Read-Only Memory), 자기 메모리, 자기 디스크, 광디스크 중 적 어도 하나의 타입의 저장매체를 포함할 수 있다. 이러한 프론트 엔드 캐시(140)는 프로세서에 제어에 의하여 동작 될 수 있다.

[0091] 다만, 본 개시에 따른 멀티 캐시 구성에 있어서, 데이터 블록의 업데이트를 포함하는 데이터 블록에 대한 수정 작업은 읽기/쓰기(read/write) 캐시를 구비한 백 엔드 캐시(240)에서 일어날 수 있다. 따라서, 바람직한 실시예에서 프론트 엔드 캐시(140)는 읽기 전용 캐시(read-only cache)일 수 있다.

[0092] 상술한 바와 같이 프론트 엔드 노드에 캐시를 추가함으로써 자주 사용되는 블록들을 프론트 엔드 캐시에서 바로 찾을 수 있어, 쿼리 처리에 소요되는 네트워크 비용이 줄어들 수 있다.

[0093] 도 3a는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 백 엔드 캐시에 버퍼 헤더(400), 메타 데이터 정보 및 버퍼 캐시 데이터가 저장된 양상을 나타낸 도면이다.

[0094] 본 개시의 몇몇 실시예에 따른 버퍼 헤더(400)는, 임의의 데이터 블록을 프론트 엔드 캐시(140)에 저장하고 있는 프론트 엔드 노드(100)에 관한 메타 데이터 정보를 포함할 수 있다.

[0095] 예를 들어, 제 1 데이터 블록이 제 1 프론트 엔드 노드 및 제 2 프론트 엔드 노드에 저장된 경우, 제 1 데이터 블록의 버퍼 헤더(400)는 제 1 프론트 엔드 및 제 2 프론트 엔드의 노드 ID 또는 이에 상응하는 식별 정보와 관련될 수 있다. 이는 메타 데이터 정보에 관한 일례에 불과하므로, 메타 데이터 정보의 구체적인 형태는 이에 한정되지 않는다.

[0096] 본 개시에 따른 버퍼 헤더(400)는 데이터 블록 마다 존재할 수 있다. 버퍼 헤더(400)는, 당해 버퍼 헤더(400)와 연관된 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 하나 또는 하나 이상의 프론트 엔드 노드 정보를 관리하는 데이터로 정의될 수 있다. 버퍼 헤더(400)는 프론트 엔드 캐시(140)와 백 엔드 캐시(240)에 저장된 데이터 블록의 관리, 프론트 엔드와 프론트 엔드 간의 데이터 블록의 전송을 위해 이용될 수 있다.

[0097] 본 개시의 몇몇 실시예에 따른 메타 데이터 정보는 특정 데이터 블록이 공유된 프론트 엔드에 관한 정보를 의미할 수 있고, 이러한 메타 데이터 정보는 공유된 프론트 엔드 정보를 저장하는 임의의 데이터 구조체에 저장될 수 있다

[0098] 본 개시에 있어서, 제 1 메타 데이터 정보(500)는 제 1 데이터 블록을 프론트 엔드 캐시에 적재하고 있는 프론트 엔드 노드들의 정보를 의미할 수 있다. 또, 제 2 메타 데이터 정보는 제 2 데이터 블록을 프론트 엔드 캐시에 적재하고 있는 프론트 엔드 노드들의 정보를 의미할 수 있다. 본 개시에 있어서는 설명의 편의 상 하나의 메타 데이터 정보가 하나의 데이터 블록에 대응하는 것으로 표현하나, 메타 데이터 정보와 데이터 블록간의 관계는 이에 한정되지 않는다.

[0099] 구체적으로, 본 개시의 몇몇 실시예에 따른 임의의 버퍼 헤더(400)는 당해 버퍼 헤더(400)에 대응하는 데이터

블록을 커런트(Current)로 가져간 프론트 엔드 노드(100)의 정보(예를 들면, 사전에 결정된 관리번호 또는 일련번호 등)를 메타 데이터 정보로서 포함할 수 있다. 이러한 메타 데이터 정보로서, 비트맵(bitmap), 배열 또는 B+ 트리 형태로 표현된 노드 ID 정보를 가리키는 포인터가 이용될 수 있다.

[0100] 본 개시의 몇몇 실시예에 따른 공유 정보는, 임의의 데이터 블록이 임의의 프론트 엔드 캐시에 저장되어 있는지 여부를 나타내는 정보를 의미할 수 있다.

[0101] 일례로, 공유 정보가 비트 형식일 수 있다. 여기서 제 1 데이터 블록에 대한 제 1 공유 정보의 비트 값이 1인 경우를 가정하자. 이 경우, 프로세서(210)는 제 1 데이터 블록에 대한 제 1 공유 정보가 존재한다고 인식할 수 있다. 반대로, 제 1 데이터 블록에 대한 제 1 공유 정보의 비트 값이 0이라면 프로세서(210)는 제 1 데이터 블록에 대한 제 1 공유 정보가 존재하지 않는다고 인식할 수 있다.

[0102] 또 다른 일례로, 공유 정보가 체인 형태로 표현될 수 있다. 이 경우, 프로세서(210)는 체인 상에 제 1 데이터 블록에 대한 제 1 공유 정보가 존재하는지 여부에 따라, 제 1 공유 정보의 존재 여부를 인식할 수 있다.

[0103] 또한, 공유 정보는 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 수에 관한 정보를 포함할 수 있다. 예를 들어, 제 1 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 수에 관한 정보의 값이 0이라면, 프로세서(210)는 제 1 데이터 블록에 대한 공유 정보가 존재하지 않는다고 인식할 수 있다.

[0104] 이는 프로세서(210)가 임의의 데이터 블록에 대한 공유 정보를 인식하는 방법에 관한 예시에 불과하다. 공유 정보의 형식 및 그 인식 방법은 상술한 바에 한정되지 않으며, 본원 발명에서 나타내는 공유 정보 및 공유 정보로 구성되는 메타 데이터 정보의 형식에 따라 적절히 변형될 수 있다.

[0105] 즉, 제 1 데이터 블록에 대한 제 1 공유 정보는 제 1 프론트 엔드 노드의 프론트 엔드 캐시에 제 1 데이터 블록이 저장되었다는 정보를 의미할 수 있다.

[0106] 또 다른 일례로, 제 2 데이터 블록에 대한 제 1 공유 정보는 제 1 프론트 엔드 노드의 프론트 엔드 캐시에 제 2 데이터 블록이 저장되었다는 정보를 의미할 수 있다.

[0107] 본 개시에 따른 메타 데이터 정보는 개별 공유 정보를 포함할 수 있으며, 임의의 데이터 블록에 대한 개별 공유 정보를 종합한 것은 메타 데이터 정보를 구성할 수 있다.

[0108] 예를 들어, 본 개시의 일 실시예에 따른 프론트 엔드 노드가 3개인 경우에, 제 1 데이터 블록에 대한 제 1 메타 데이터 정보(500)는 제 1 데이터 블록에 대한 제 1 공유 정보, 제 1 데이터 블록에 대한 제 2 공유 정보, 제 1 데이터 블록에 대한 제 3 공유 정보를 포함할 수 있다.

[0109] 본 개시에 따른 몇몇 실시예에 있어서, 메타 데이터 정보는 버퍼 헤더(400)에 포함될 수 있다.

[0110] 비트 맵 정보는, 임의의 프론트 엔드 노드에 대응되는 필드에 0 또는 1을 이용하여 임의의 프론트 엔드 노드의 캐시에 임의의 데이터 블록이 저장되었는지를 나타내는 정보를 의미할 수 있다.

[0111] 예를 들면, 제 1 프론트 엔드 노드(100)가 제 1 데이터 블록을 프론트 엔드 캐시(140)에 저장하고 있을 경우, 제 1 데이터 블록의 버퍼 헤더(400)에 포함된 비트맵 정보는 비트맵 정보에 포함된 임의의 필드에 0 또는 1 이 입력되어 있는 형태일 수 있다. 이 때, 임의의 필드 각각은 임의의 프론트 엔드 노드에 대응된다.

[0112] 본 개시의 몇몇 실시예에 따른 메타 데이터 정보는 버퍼 캐시 데이터와 동일한 검색 구조를 공유할 수 있다. 여기서 동일한 검색 구조를 가진다는 것의 구체적인 의미는, 도 3a에 도시되는 바와 같이 하나의 해시 버켓 안에 버퍼 헤더와 메타 데이터 정보가 같이 들어간다는 것일 수 있다. 따라서, 프로세서(210)는 한 번의 락(lock)을 통해 버퍼 헤더와 메타 데이터 정보를 모두 찾아낼 수 있다. 따라서, 데이터 블록(또는 버퍼 캐시 데이터)의 검색이 효율적으로 수행될 수 있다.

[0113] 구체적으로, 버퍼 헤더(400)와 메타 데이터 정보는 동일한 해시 함수를 이용해 백 엔드 캐시에 저장될 수 있다. 버퍼 헤더와 메타 데이터 정보를 해시 함수를 이용해 백 엔드 데이터에 저장하기 위해 해시 함수에 입력되는 해시 함수의 입력 값은 동일할 수 있다. 이 때, 검색 구조 내에서 버퍼 헤더와 메타 데이터 정보가 저장된 공간은 입력 값을 해시 함수에 입력하여 얻어진 결과 값과 연관되어, 검색 구조를 공유할 수 있다.

[0114] 예를 들어, 프로세서(210)는 캐시에서 찾고자하는 블록의 데이터 파일 번호와, 파일 내에서 당해 데이터 블록이 몇 번째 블록인지를 해시 함수의 입력 값으로 하여 해시 결과 값을 구할 수 있다. 이 해시 결과 값을 이용하여,

프로세서(210)는 버퍼 헤더와 메타 데이터 정보를 특정 해시 버켓(300a)과 연결하여 백 엔드 캐시(240)내에 저장할 수 있다. 추후 프로세서(210)가 상기 데이터 블록(버퍼 캐시 데이터)에 접근하고자 할 경우, 동일하게 당해 블록의 데이터 파일 번호 및 당해 데이터 블록이 몇 번째 블록인지를 해시 함수의 입력 값으로 하여 해시 버켓을 찾을 수 있다. 그 후 그 버켓의 락을 잡고 리스트를 검색하면, 단 한번의 락을 이용하여 메타 데이터 정보, 버퍼 헤더 및 버퍼 캐시 데이터를 찾아낼 수 있다. 따라서, 버퍼 캐시 데이터를 찾기 위한 검색의 효율화가 달성되어 네트워크 비용이 감소할 수 있다.

[0115] 추가적으로, 프로세서(210)는 해시 버켓에 메타 데이터 정보와 버퍼 헤더를 저장할 때, 메타 데이터 정보를 위한 리스트 및 버퍼 헤더를 위한 리스트를 별도로 생성할 수 있다. 이에 관한 자세한 내용은 도 3b에서 후술한다.

[0117] 본 개시의 몇몇 실시예에 따른 프론트 엔드 노드의 노드 ID(식별 정보)는 2바이트의 크기를 가질 수 있다. 따라서, 임의의 데이터 블록 하나에 대한 메타 데이터 정보는 노드의 수가 늘어날수록 커진다. 따라서, 메타 데이터 정보로 인해 저장 공간의 비효율적 사용이 일어날 수 있다.

[0118] 본 개시의 몇몇 실시예에 따른 백 엔드 노드(200)의 프로세서(210)가 메타 데이터 정보를 관리하는 또 다른 일례에 대해 설명한다.

[0119] 본 개시의 몇몇 실시예에 따른 백 엔드 노드(200)의 프로세서(210)에 의하여, 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 ID 정보는 클라우드 환경의 전체 프론트 엔드 노드의 수가 기 설정된 수 미만이라면 메타 데이터 정보에 비트맵(bitmap) 형태로 표현되어 포함될 수 있다.

[0120] 클라우드 환경의 전체 프론트 엔드 수가 기 설정된 수 이상이면서, 임의의 데이터 블록을 프론트 엔드 캐시에 저장한 프론트 엔드 노드가 제 1 값 이하인 경우에 프로세서(210)는 메타 데이터 정보에 포함된 노드 ID와 관련된 필드를 배열로 변환하여, 당해 데이터 블록을 프론트 엔드 캐시에 저장한 프론트 엔드 노드의 노드 ID를 저장할 수 있다.

[0121] 반대로, 클라우드 환경의 전체 프론트 엔드 수가 기 설정된 수 이상이면서, 임의의 데이터 블록을 프론트 엔드 캐시에 저장한 프론트 엔드 노드가 제 1 값을 초과하는 경우 프로세서(210)는 메타 데이터 정보에 포함된 노드 ID와 관련된 필드를 포인터로 변환할 수 있다. 변환된 포인터는 노드 ID와 관련된 공유 메모리 공간을 가리킬 수 있다. 이 때, 공유 메모리 공간에서, 임의의 데이터 블록을 가져간 노드의 노드 ID는 공유 메모리 공간 상에서 B+ 트리 형태로 표현되어 저장될 수 있다.

[0122] 메타 데이터 정보 내에 포함된 노드 ID와 관련된 필드의 크기를 8바이트(64비트)라고 가정하고, 상술한 메타 데이터 정보 관리 방법의 일 예시를 설명한다. 클라우드 환경의 전체 프론트 엔드 수가 64개 미만인 경우, 메타 데이터 정보는 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드들의 노드 ID를 비트맵 형태로 포함할 수 있다.

[0123] 클라우드 환경의 전체 프론트 엔드 수가 64개 이상인 경우에, 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드가 4개 이하인 경우, 프로세서(210)는 메타 데이터 정보에 포함된 노드 ID와 관련된 필드를 배열 형태로 구성할 수 있다. 이에 따라 프로세서(210)는 상기 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 노드 ID(당해 노드 ID는 2바이트의 크기를 가질 수 있다)를 상기 배열에 기록할 수 있다.

[0124] 클라우드 환경의 전체 프론트 엔드 수가 64개 이상인 경우에, 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드가 4개를 초과하는 경우, 프로세서(210)는 메타 데이터 정보에 포함된 노드 ID와 관련된 필드를 포인터로 변환할 수 있다. 상기 포인터는 공유 메모리 공간에 B+ 트리 형태로 저장된 노드 ID 정보를 가리킬 수 있다.

[0125] 이는 메타 데이터를 관리하는 방법의 일례에 불과하므로 메타 데이터의 관리 방법 및 그 형태는 이에 한정되지 않는다.

[0126] 본 개시의 몇몇 실시예에 따른 버퍼 캐시 데이터는, 디스크에 저장된 데이터 블록이 복사된 데이터일 수 있다. 즉, 버퍼 캐시 데이터는 디스크 또는 프론트 엔드에 저장된 데이터 블록에 대응될 수 있다. 본 개시의 몇몇 부분에서 버퍼 캐시 데이터와 데이터 블록은 혼용되어 사용될 수 있다.

[0127] 또, 버퍼 캐시 데이터는 버퍼 헤더(400) 및 메타 데이터 중 적어도 하나와 연결되어, 백 엔드 캐시(240)내에 저

장될 수 있다.

[0129] 메타 데이터를 위와 같이 관리할 경우, 임의의 데이터 블록을 프론트 엔드 캐시에 저장하는 프론트 엔드 노드(100) 정보를 효율적으로 저장할 수 있다.

[0130] 도 3b는 검색 구조를 공유하기 위한 구체적인 방법의 일례를 나타낸 도면이다.

[0131] 도 3a에서 상술한 바와 같이, 프로세서(210)는 하나의 해시 버킷 내에서 버퍼 헤더(400)의 리스트와 메타 데이터 정보의 리스트를 통합하거나 분리하여 백 엔드 캐시(240)에 저장할 수 있다.

[0132] 도 3b에서 도시되는 바와 같이, 하나의 해시 버킷(300a)에 연결되는 리스트의 노드들은 버퍼 헤더 구조체(400a), 메타 데이터 정보 구조체(500a), 일반 구조체(700) 중 어느 하나일 수 있다.

[0133] 만약 해시 버킷(300a)에 버퍼 헤더(400)의 리스트와 메타 데이터 정보의 리스트가 분리되어 연결된 경우에, 프로세서(210)는 데이터 파일 번호 및 데이터 블록이 몇 번째인지를 이용해 해시 버켓을 찾고, 이를 통해 버퍼 헤더, 메타 데이터 정보 및 버퍼 캐시 데이터를 찾을 수 있다.

[0134] 해시 버켓(300a)에 연결되는 버퍼 헤더(400)의 리스트와 메타 데이터 정보의 리스트가 통합된 경우에, 프로세서(210)는 먼저 일반 구조체(700)를 이용해 리스트에 접근한 후, 버퍼 헤더 구조체(400a) 및 메타 데이터 정보 구조체(500a)에 포함된 유형(type) 요소를 인식하여 리스트의 해당 구조체가 버퍼 헤더(400)에 해당하는지, 메타 데이터 정보에 해당하는지를 인식할 수 있다. 해당 구조체의 유형이 인식되면, 프로세서(210)는 해당 유형에 알맞은 처리를 수행할 수 있다.

[0135] 더 구체적으로, 일반 구조체(700) 타입으로 리스트에 포함된 버퍼 헤더 구조체(400a) 및 메타 데이터 정보 구조체(500a)가 접근되려면, 두 구조체의 앞 부분은 통일되어야 한다. 이를 위해, 예시적으로 도 3a에서 list_link_t bucket_link; 라는 구조체의 요소가 존재한다. 프로세서(210)는, 우선 일반 구조체(700) 타입을 이용해 리스트에 접근한 후, 버퍼 헤더 구조체(400a) 및 메타 데이터 정보 구조체(500a)에 포함된 유형 요소를 인식할 수 있다. 유형 요소가 인식되면, 프로세서(210)는 상기 일반 구조체(700)를 당해 유형 요소로 타입 형 변환(type casting)을 수행할 수 있다. 당해 유형 요소로 타입 형 변환된 리스트의 구조체는 그에 알맞은 로직에 따라 처리될 수 있다. 버퍼 헤더, 메타 데이터 정보 및 버퍼 캐시 데이터 간 검색 구조를 공유하기 위한 방법의 일례에 불과하므로, 권리범위는 상술한 예시에 한정되지 아니한다.

[0136] 상술한 방법에 의하여, 프로세서(210)는 하나의 해시 버켓에 대한 락(lock) 만으로 버퍼 헤더, 메타 데이터 정보 및 버퍼 캐시 데이터에 모두 접근할 수 있다. 따라서, 복수의 락을 잡을 필요가 없으므로 프로세서(210)의 탐색에 관한 처리 비용이 감소하고, 네트워크 비용 또한 절감될 수 있다.

[0138] 도 4는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록을 프론트 엔드 노드로 전달하는 것을 나타낸 순서도이다.

[0139] 도 4를 참조하면, 프로세서(210)는 제 1 버퍼 캐시 데이터(600) 또는 제 1 메타 데이터 정보(500) 중 적어도 하나를 백 엔드 캐시에서 탐색할 수 있다(S110).

[0140] 프로세서(210)는, 제 1 버퍼 캐시 데이터(600)가 존재하는 경우 또는 탐색된 제 1 메타 데이터 정보(500)에 공유 정보가 존재하는 경우 (S120, Yes), 제 1 데이터 블록에 대한 제 1 공유 정보를 제 1 메타 데이터 정보(500)에 저장할 수 있다(S150).

[0141] 상술한 단계(S150)은 제 1 버퍼 캐시 데이터(600)가 존재하고 탐색된 제 1 메타 데이터 정보(500)에 공유 정보가 존재하는 경우에도 수행된다.

[0142] 본 개시의 몇몇 실시예에 따른 공유 정보는, 임의의 데이터 블록이 임의의 프론트 엔드 캐시에 저장되었다는 정보를 의미할 수 있다.즉, 제 1 데이터 블록에 대한 제 1 공유 정보는 제 1 프론트 엔드 노드의 프론트 엔드 캐시에 제 1 데이터 블록이 저장되었다는 정보를 의미할 수 있다.

[0143] 또 다른 일례로, 제 2 데이터 블록에 대한 제 1 공유 정보는 제 1 프론트 엔드 노드의 프론트 엔드 캐시에 제 2 데이터 블록이 저장되었다는 정보를 의미할 수 있다.

[0144] 일례로, 공유 정보가 비트 형식일 수 있다. 여기서 제 1 데이터 블록에 대한 제 1 공유 정보의 비트 값이 1인 경우를 가정하자. 이 경우, 프로세서(210)는 제 1 데이터 블록에 대한 제 1 공유 정보가 존재한다고 인식할 수 있다. 이 때, 프로세서(210)는 제 1 데이터 블록에 대한 공유 정보가 존재한다고 인식할 수 있다.

[0145] 반대로, 제 1 데이터 블록에 대한 제 1 공유 정보의 비트 값이 0이라면 프로세서(210)는 제 1 데이터 블록에 대한 제 1 공유 정보가 존재하지 않는다고 인식할 수 있다. 이때, 프로세서(210)는 제 1 데이터 블록에 대한 공유 정보가 존재하지 않는다고 인식할 수 있다.

[0146] 또 다른 일례로, 공유 정보가 체인 형태로 표현될 수 있다. 이 경우, 프로세서(210)는 체인 상에 제 1 데이터 블록에 대한 제 1 공유 정보가 존재하는지 여부에 따라, 제 1 공유 정보의 존재 여부를 인식할 수 있다.

[0147] 또한, 공유 정보는 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 수에 관한 정보를 포함할 수 있다. 예를 들어, 제 1 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드의 수에 관한 정보의 값이 0이라면, 프로세서(210)는 제 1 데이터 블록에 대한 공유 정보가 존재하지 않는다고 인식할 수 있다.

[0148] 특정 데이터 블록에 대한 공유 정보가 존재하지 않을 경우, 프로세서(210)는 "특정 데이터 블록이 어떤 프론트 노드 엔드에도 공유되지 않았다"고 인식할 수 있다.

[0149] 이는 프로세서(210)가 임의의 데이터 블록에 대한 공유 정보를 인식하는 방법에 관한 예시에 불과하다. 공유 정보의 형식 및 그 인식 방법은 상술한 바에 한정되지 않으며, 본원 발명에서 나타내는 공유 정보 및 공유 정보로 구성되는 메타 데이터 정보의 형식에 따라 적절히 변형될 수 있다.

[0150] 따라서, 프로세서(210)는 제 1 메타 데이터 정보를 백 엔드 캐시(240)에서 탐색한 경우에도, 제 1 메타 데이터 정보에 공유 정보가 존재하지 않는 경우에는, 제 1 버퍼 캐시 데이터가 백 엔드 캐시(240)에 존재하지 않는 한, 다른 프론트 엔드 노드로부터 제 1 데이터 블록을 받아 프론트 엔드 노드로 전달할 수 없다. 따라서 이 경우 프로세서(210)는 디스크로부터 제 1 데이터 블록을 받아, 이를 제 1 프론트 엔드 노드로 전달하게 된다.

[0151] 본 개시에 따른 메타 데이터 정보는 개별 공유 정보를 포함할 수 있으며, 임의의 데이터 블록에 대한 개별 공유 정보를 종합한 것은 메타 데이터 정보를 구성할 수 있다.

[0152] 예를 들어, 본 개시의 일 실시예에 따른 프론트 엔드 노드가 3개인 경우에, 제 1 데이터 블록에 대한 제 1 메타 데이터 정보(500)는 제 1 데이터 블록에 대한 제 1 공유 정보, 제 1 데이터 블록에 대한 제 2 공유 정보, 제 1 데이터 블록에 대한 제 3 공유 정보를 포함할 수 있다.

[0153] 본 개시에 따른 몇몇 실시예에 있어서, 메타 데이터 정보는 버퍼 헤더(400)에 포함될 수 있다.

[0154] 예를 들어, 제 1 버퍼 캐시 데이터(600)가 백 엔드 캐시(240)에서 탐색된 경우, 프로세서(210)는 제 1 버퍼 캐시 데이터(600)(제 1 데이터 블록)에 대한 제 1 공유 정보를 제 1 메타 데이터 정보(500)에 저장한 후, 제 1 버퍼 캐시 데이터(600)(제 1 데이터 블록)를 제 1 프론트 엔드 노드로 전송할 수 있다.

[0155] 또 다른 일례로, 탐색된 제 1 메타 데이터 정보(500)에 공유 정보가 존재할 경우, 프로세서(210)는 제 1 메타 데이터 정보(500)에 포함된 제 1 데이터 블록에 대한 제 2 공유 정보를 탐색하여, 제 2 프론트 엔드 노드로부터 제 1 데이터 블록을 수신할 수 있다. 그 후 프로세서(210)는 제 1 데이터 블록에 대한 제 1 공유 정보를 제 1 메타 데이터 정보(500)에 저장한 뒤, 제 1 데이터 블록을 제 1 프론트 엔드 노드로 전송하도록 통신부를 제어할 수 있다.

[0156] 이 때, 백 엔드 노드(200)로부터 제 1 데이터 블록에 대한 요청 신호를 수신한 임의의 프론트 엔드 노드의 프로세서(110)는, 제 1 데이터 블록을 프론트 엔드 캐시에서 탐색하고, 제 1 데이터 블록이 프론트 엔드 캐시에 존재하면, 제 1 데이터 블록을 백 엔드 노드로 전송할 수 있다.

[0157] 상술한 바에 의하여, 백 엔드 캐시에 메타 데이터 정보 또는 버퍼 캐시 데이터 중 하나만 존재하더라도 디스크에 대한 접근 없이 제 1 데이터 블록에 대한 셀렉트 쿼리 요청을 처리할 수 있다. 따라서 네트워크 비용의 감소 및 처리 속도의 향상이 달성될 수 있다.

[0158] 프로세서(210)는, 제 1 버퍼 캐시 데이터(600) 및 공유 정보가 존재하는 및 제 1 메타 데이터 정보(500)가 백 엔드 캐시에 존재하지 않는 경우(S120, No), 디스크에 제 1 데이터 블록에 대한 요청 신호를 전송할 수 있다(S130).

[0159] 프로세서(210)는, 디스크로부터 제 1 데이터 블록을 수신할 수 있다(S140).

[0160] 본 개시의 몇몇 실시예에 따른 프로세서(210)는, 백 엔드 캐시(240)에 제 1 버퍼 캐시 데이터가 탐색되지 않으면서, 제 1 메타 데이터 정보가 탐색되지 않거나, 제 1 메타 데이터 정보가 탐색되더라도 공유 정보가 그 안에

존재하지 않는 경우에는, 디스크(300)에 제 1 데이터 블록에 대한 요청 신호를 전송하도록 통신부(220)를 제어할 수 있다. 프로세서(210)는 디스크(300)로부터 제 1 데이터 블록을 수신하면, 제 1 데이터 블록을 제 1 프론트 엔드 노드로 전송하도록 통신부(220)를 제어할 수 있다.프로세서(210)는 디스크(300)로부터 수신한 제 1 데이터 블록을 제 1 프론트 엔드로 전송함으로써, 제 1 프론트 엔드의 제 1 데이터 블록에 대한 요청을 처리할 수 있다.

[0161] 이와 별도로, 프로세서(210)가 공유 정보가 존재하는 제 1 메타 데이터 정보의 탐색 과정에서 제 1 메타 데이터 정보 자체(공유 정보의 존재 여부와 무관함)가 백 엔드 캐시(240)에 존재하는 지 여부를 인식할 수 있음은 자명하다. 따라서, 제 1 메타 데이터 정보(500)가 백 엔드 캐시(240)에 존재하지 않을 경우, 프로세서(210)는 제 1 메타 데이터 정보(500)를 수용할 수 있는 제 1 데이터 구조체를 생성 또는 로드(load)하고, 제 1 데이터 구조체에 제 1 데이터 블록의 정보를 저장하고, 제 1 데이터 구조체를 제 1 메타 데이터 정보(500)로서 결정할 수 있다. 여기서, 제 1 데이터 구조체는 메타 데이터 정보와 동일한 형식을 가지는 구조체일 수 있다.

[0162] 프로세서(210)는, 제 1 데이터 블록을 제 1 프론트 엔드 노드로 전송하도록 통신부를 제어할 수 있다(S160).

[0163] 상술한 바와 같이, 백 엔드 노드(200)의 프로세서(210)는 백 엔드 캐시(240)에 제 1 버퍼 캐시 데이터(600)가 존재하는 경우에는 제 1 버퍼 캐시 데이터(600)를 제 1 프론트 엔드 노드로 전송할 수 있다. 제 1 버퍼 캐시 데이터(600)는 디스크 또는 제 1 메타 데이터에 기록된 프론트 엔드 노드들에 저장된 제 1 데이터 블록과 동일할 수 있다.

[0164] 제 1 데이터 블록을 전송받은 제 1 프론트 엔드 노드는 프론트 엔드 노드에 포함된 옵티마이저(130)를 이용하여 이를 프론트 엔드 캐시(140)에 저장할지 여부를 결정할 수 있다.

[0165] 도 5는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록에 대한 업데이트를 수행하는 과정을 나타낸 순서도이다.

[0166] 본 개시에 있어 데이터 블록에 대한 업데이트는, 데이터 블록에 대한 데이터의 추가적인 입력, 데이터의 수정 또는 데이터의 삭제 등을 통해 일어난 데이터 블록의 변화를 백 엔드 캐시(240)에 기록하는 것을 의미할 수 있다.

[0167] 본 개시에 따른 클라우드 데이터베이스 시스템에 있어 업데이트는 백 엔드 노드(200)에서 발생할 수 있으므로, 백 엔드 캐시(240)에 업데이트를 기록하기 전 업데이트가 발생한 데이터 블록들을 프론트 엔드 캐시에서 제거하는 과정이 요구된다.

[0168] 도 5를 참조하면, 프로세서(210)는 제 2 데이터 블록에 대한 업데이트 요청을 수신할 수 있다(S210).

[0169] 본 개시에 따른 백 엔드 캐시(240)는 읽기 및 쓰기가 모두 가능할 수 있다. 따라서, 임의의 데이터 블록에 대한 업데이트 요청은 백 엔드 노드(200)의 프로세서(210)에 의해 수행될 수 있다.

[0170] 프로세서(210)는 제 2 메타 데이터 정보를 탐색할 수 있다(S220).

[0171] 이 때, 본 개시의 몇몇 실시예에 따른 프로세서(210)는 제 2 메타 데이터 정보가 존재하지 않는 경우 제 2 데이터 블록에 대한 업데이트를 수행할 수 있다. 제 2 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드가 존재하지 않으므로, 이후의 단계가 더 이상 필요하지 않을 수 있다.

[0172] 프로세서(210)는 제 2 메타 데이터 정보를 이용하여 제 2 데이터 블록을 저장한 하나 이상의 프론트 엔드 노드를 인식할 수 있다(S230).

[0173] 단계(S230)를 통해 제 2 데이터 블록이 프론트 엔드 캐시에서 삭제되어야 하는 프론트 엔드 노드(100)의 정보를 파악할 수 있다.

[0174] 프로세서(210)는 제 2 데이터 블록에 대한 업데이트를 수행할 수 있다(S240).

[0175] 도 5에서, 단계(S250)는 단계(S240)의 수행 후에 수행되는 것으로 도시되었다. 그러나 이는 설명의 편의를 위한 것일 뿐, 단계(S240)와 단계(S250)의 순서는 이에 구속되지 아니한다. 즉, 인밸리데이트 신호는 백 엔드 노드(200)에서 프로세서(210)가 제 2 데이터 블록에 대한 업데이트의 수행 시점과 무관하게 프론트 엔드 노드로 전송될 수 있다.

[0176] 또, 본 개시의 몇몇 실시예에 따른 프로세서(210)는 제 2 데이터 블록에 대한 업데이트를 동기적으로뿐만 아니라 비동기적으로도 수행할 수 있다.

[0177] 프로세서(210)는 제 2 데이터 블록을 저장한 하나 이상의 프론트 엔드 노드에 인밸리데이트 신호를 전송할 수 있다(S250).

[0178] 프로세서(210)는 인밸리데이트 신호를 전송한 하나 이상의 프론트 엔드 노드 모두로부터 인밸리데이트 완료 신호를 수신하면, 업데이트를 요청한 제 1 프론트 엔드로 업데이트 완료 신호를 전송할 수 있다.

[0179] 예를 들어, 본 개시의 몇몇 실시예에 따른 프로세서(210)는, 제 2 데이터 블록에 대한 업데이트를 수행한 후, 인밸리데이트 신호를 전송한 하나 이상의 프론트 엔드 노드 모두로부터 인밸리데이트 신호에 대한 완료 신호를 수신했는지 여부를 인식할 수 있다. 프로세서(210)는 인밸리데이트 신호를 전송한 하나 이상의 프론트 엔드 노드 모두로부터 인밸리데이트 신호에 대한 완료 신호를 수신한 경우, 제 2 데이터 블록에 대한 업데이트 완료 신호를 제 1 프론트 엔드 노드로 전송할 수 있다.

[0180] 상술한 바에 의하여, 다수의 프론트 엔드 노드(100)와 백 엔드 노드(200)에 데이터 블록이 각각 저장되어있는 경우에도, 프론트 엔드 노드(100)와 백 엔드 노드(200) 간의 데이터 블록 간의 일치성을 유지하며 업데이트를 수행할 수 있다.

[0182] 도 6은 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 프론트 노드에서 인밸리데이트 된 데이터 블록의 정보를 업데이트하는 과정을 나타낸 순서도이다.

[0183] 프로세서(210)는, 제 1 프론트 엔드 노드로부터 제 3 데이터 블록이 제 1 프론트 엔드 노드에서 인밸리데이트되었다는 신호를 수신할 수 있다(S310).

[0184] 본 개시에 따른 데이터 블록의 인밸리데이트(invalidate)는 캐시에 저장된 데이터 블록을 캐시에서 삭제하는 것을 의미할 수 있다.

[0185] 프로세서(210)는 프론트 엔드 노드(100)로부터, 프론트 엔드 캐시(140)에서 제 3 데이터 블록이 인밸리데이트되었다는 신호를 수신한 후 이를 메타 데이터 정보에 반영할 수 있다.

[0186] 즉, 본 개시의 몇몇 실시예에 따른 프로세서(110)는 제 3 데이터 블록에 대한 인밸리데이트 신호를 수신한 경우, 제 3 데이터 블록을 프론트 엔드 캐시로부터 인밸리데이트할 수 있다.

[0187] 또한, 프로세서(110)는 프론트 엔드 캐시에 커런트(Current) 상태로 저장되어있는 제 3 데이터 블록을 CR(Consistent Read) 상태로 변경함으로써, 인밸리데이트를 수행할 수 있다.

[0188] 따라서, CR로 변경된 데이터 블록은 추후 재사용될 수 있다. 예를 들어, CR로 변경된 블록에 대해서 쿼리가 요청될 경우에, 프로세서(110)는 CR로 변경된 블록과 스냅샷에 기록된 블록을 비교할 수 있다. CR로 변경된 블록과 스냅샷에 기록된 블록이 동일한 경우, 프로세서(110)는 CR로 변경된 블록을 반환할 수 있다.

[0189] 프로세서(210)는, 제 3 데이터 블록과 관련된 제 3 메타 데이터 정보를 탐색할 수 있다(S320).

[0190] 여기서, 제 3 메타 데이터 정보는 제 3 데이터 블록을 프론트 엔드 캐시에 포함하고 있는 프론트 엔드들의 정보를 의미할 수 있다.

[0191] 프로세서(210)는, 제 3 메타 데이터 정보에서 제 3 데이터 블록에 대한 제 1 공유 정보를 삭제할 수 있다(S330).

[0192] 프로세서(210)는 제 3 메타 데이터에 포함된, 제 3 데이터 블록에 대한 제 1 공유 정보를 삭제함으로써 제 1 프론트 엔드 노드의 프론트 엔드 캐시에 더 이상 제 3 데이터 블록이 존재하지 않음을 나타낼 수 있다.

[0193] 상술한 바에 의하여, 임의의 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 프론트 엔드 노드들에 대한 지속적인 추적이 가능하다. 따라서, 프로세서는 추후 데이터 블록의 탐색 요청이 있을 시 당해 데이터 블록을 가지고 있는 프론트 엔드 노드들을 빠르게 찾아낼 수 있어, 네트워크 비용 및 셀렉트 쿼리 처리 속도가 빨라질 수 있다.

[0195] 도 7은 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록이 프론트 엔드 캐시에서 캐시 아웃된 경우 이를 업데이트 하는 과정을 나타낸 순서도이다.

[0196] 도 7을 참조하면, 프로세서(210)는 제 4 데이터 블록이 제 1 프론트 엔드 노드에서 캐시 아웃되었다는 캐시 아웃 신호를 수신할 수 있다(S410).

[0197] 본 개시의 몇몇 실시예에 따른 캐시 아웃은 프론트 엔드 노드(100)의 프로세서(110)에 의해 프론트 엔드 캐시

(140)에 커런트 상태로 저장되어 있는 임의의 데이터 블록이 CR(Consistent Read) 상태로 변경되는 것을 의미할 수 있다. 그러나 프론트 엔드 노드(100)는 당해 데이터 블록을 메모리 상에서 삭제할 수도 있다. 즉, 캐시 아웃의 수행은 상술한 방법들에 한정되지 않는다.

[0198] 본 개시의 몇몇 실시예에 따른 프론트 엔드 노드(100)의 프로세서(110)는 기 설정된 방법을 이용하여 캐시 아웃될 데이터 블록을 결정할 수 있다.

[0199] 예를 들면, 프론트 엔드 노드(100)의 프로세서(110)는 페이지 교체 알고리즘의 일종인 LRU(Least Recently Used)를 이용하여 프론트 엔드 캐시(140)에서 특정 데이터 블록을 캐시 아웃시킬 수 있다. 즉, 임의의 프론트 엔드 노드(100)는 프론트 엔드 캐시(140)에 저장된 데이터 블록 중 가장 오랫동안 셀렉트 쿼리가 수행되지 않은 데이터 블록을 프론트 엔드 캐시(140)에서 캐시 아웃시킬 수 있다.

[0200] 이는 캐시 아웃될 데이터 블록을 결정하는 일례에 불과하며, 다양한 규칙이 캐시 아웃될 데이터 블록을 결정하는데 사용될 수 있다.

[0201] 백 엔드 노드(200)의 프로세서는 메타 데이터 정보만을 조정함으로써 임의의 프론트 엔드 캐시(140)에서의 데이터 블록 저장 상태를 관리할 수 있다. 따라서 프론트 엔드 캐시(140)를 유지하면서 동시에 프론트 엔드 캐시(140) 관리에 요구되는 네트워크 비용을 절감할 수 있다.

[0202] 프로세서(110)는, 프론트 엔드 캐시(140)에서 캐시 아웃될 제 4 데이터 블록을 인식할 수 있다. 프론트 엔드 캐시(140)에서 캐시 아웃 될 제 4 데이터 블록을 인식한 후, 프로세서(110)는 제 4 데이터 블록을 프론트 엔드 캐시(140)로부터 캐시아웃 시킬 수 있다. 그 후, 제 4 데이터 블록이 프론트 엔드 캐시(140)에서 캐시 아웃되었다는 캐시 아웃 신호를 백 엔드 노드(200)로 전송할 수 있다.

[0203] 프로세서(210)는 제 4 데이터 블록에 대한 제 4 메타 데이터 정보를 업데이트할 수 있다(S420).

[0204] 여기서 메타 데이터 정보의 업데이트는, 제 4 데이터 블록이 캐시 아웃된 프론트 엔드 노드에 관한 공유 정보를 제 4 메타 데이터 정보로부터 제거하는 것을 의미할 수 있다.

[0205] 상술한 바에 의하여, 프론트 엔드 노드(100)에서 특정 데이터 블록이 캐시 아웃된 경우에 이를 백 엔드 캐시(240)에 포함된 메타 데이터 정보에 반영할 수 있다. 따라서, 당해 데이터 블록이 삭제된 프론트 엔드 노드에 대한 불필요한 접근을 줄일 수 있어, 네트워크 비용이 감소될 수 있다.

[0206] 도 8은 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 캐시 아웃된 데이터 블록이 다시 백 엔드 캐시에 저장된 경우 메타 데이터 정보를 로드하는 과정을 나타낸 순서도이다.

[0207] 도 8을 참조하면, 프로세서(210)는 제 5 데이터 블록에 대한 제 5 메타 데이터 정보를 백 엔드 노드의 메모리에 저장할 수 있다(S510).

[0208] 프로세서(210)는, 제 5 데이터 블록이 백 엔드 캐시(240)에서 캐시 아웃된 경우, 이에 대응하는 제 5 메타 데이터 정보를 백 엔드 노드의 메모리에 저장할 수 있다.

[0209] 본 개시의 몇몇 실시예에 따른 캐시 아웃은 백 엔드 노드(200)의 프로세서(210)에 의해 프론트 엔드 캐시(240)에 커런트 상태로 저장되어 있는 임의의 데이터 블록이 CR(Consistent Read) 상태로 변경되는 것을 의미할 수 있다. 그러나 프론트 엔드 노드(200)는 당해 데이터 블록을 메모리 상에서 삭제할 수도 있다. 즉, 캐시 아웃의 수행은 상술한 방법들에 한정되지 않는다.

[0210] 본 개시의 몇몇 실시예에 따른 백 엔드 노드(200)의 프로세서(210)는 기 설정된 방법을 이용하여 캐시 아웃될 데이터 블록을 결정할 수 있다.

[0211] 예를 들면, 백 엔드 노드(200)의 프로세서(210)는 페이지 교체 알고리즘의 일종인 LRU(Least Recently Used)를 이용하여 백 엔드 캐시(240)에서 특정 데이터 블록을 캐시 아웃시킬 수 있다. 즉, 임의의 백 엔드 노드(200)는 백 엔드 캐시(240)에 저장된 데이터 블록 중 가장 오랫동안 셀렉트 쿼리가 수행되지 않은 데이터 블록을 백 엔드 캐시(240)에서 캐시 아웃시킬 수 있다.

[0212] 이는 캐시 아웃될 데이터 블록을 결정하는 일례에 불과하며, 다양한 규칙이 캐시 아웃될 데이터 블록을 결정하는데 사용될 수 있다.

[0213] 특히, 프로세서(210)는 제 5 데이터 블록이 캐시 아웃 되는 경우라도, 제 5 메타 데이터 정보를 백 엔드 캐시(240)에 남겨둘 수 있다.

[0214] 프로세서(210)는 제 5 데이터 블록이 다시 백 엔드 노드로 읽어들여지면, 제 5 메타 데이터 정보를 다시 로드(load)할 수 있다(S520).

[0215] 그 후, 프로세서(210)는 제 5 데이터 블록이 다시 버퍼 캐시 데이터의 형태로 백 엔드 캐시(240)로 읽어들여질 경우, 메모리(또는 백 엔드 캐시(240))에 기 저장되어 있던 제 5 메타 데이터 정보를 다시 읽어들일 수 있다.

[0216] 구체적으로, 본 개시의 몇몇 실시예에 따른 프로세서(210)는, 제 5 데이터 블록이 백 엔드 노드(200)로 다시 읽어 들여지면, 제 5 메타 데이터 정보를 다시 로드할 수 있다.

[0217] 특히, 제 5 메타 데이터 정보를 다시 로드하는 것은: 제 5 데이터 블록이 백 엔드 캐시로 읽어들여졌다고 인식한 경우, 제 5 메타 데이터 정보를 인식하고, 제 5 메타 데이터를 제 5 데이터 블록과 관련된 버퍼 헤더(400)에 기록함으로써 이루어질 수 있다.

[0218] 상술한 바에 의하여, 특정 데이터 블록이 백 엔드 캐시(240)로부터 삭제되더라도 백 엔드 노드(200)의 프로세서(210)는 여전히 당해 데이터 블록을 프론트 엔드 캐시(140)에 저장하고 있는 프론트 엔드 노드(100) 정보에 접근할 수 있다. 따라서, 디스크(300)에 대한 접근 없이 프론트 엔드 노드(100)에 접근함으로써 데이터 블록을 불러올 수 있고, 이에 따라 네트워크 비용이 감소될 수 있다.

[0220] 도 9는 본 개시의 몇몇 실시예에 따른 백 엔드 노드의 프로세서가 데이터 블록의 메타 데이터 정보를 백 엔드 캐시로부터 삭제하는 과정을 나타낸 순서도이다.

[0221] 도 9를 참조하면, 프로세서(210)는 제 6 데이터 블록에 대응하는 제 6 메타 데이터 정보를 인식할 수 있다(S610).

[0222] 여기서, 제 6 메타 데이터 정보는 백 엔드 캐시(240)에서 삭제될 메타 데이터 정보를 의미할 수 있다.

[0223] 이에 의하여, 프로세서(210)는 제 6 데이터 블록을 프론트 엔드 캐시(140)에 저장하고 있는 프론트 엔드 노드(100)들의 정보를 인식할 수 있다.

[0224] 프로세서(210)는 제 6 메타 데이터 정보에 기초하여 제 6 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 적어도 하나의 프론트 엔드 노드를 인식할 수 있다(S620).

[0225] 제 6 메타 데이터 정보를 삭제하면 백 엔드 캐시(240) 상에서 제 6 데이터 블록을 프론트 엔드 캐시(140)에 저장하고 있는 프론트 엔드 노드(100) 정보에 접근할 수 없게된다. 따라서, 제 6 메타 데이터 정보의 삭제에 맞추어 프론트 엔드 노드(100)의 프론트 엔드 캐시(140)에서 제 6 데이터 블록을 삭제하는 것이 요구된다.

[0226] 프로세서(210)는 적어도 하나의 프론트 엔드 노드로 제 6 데이터 블록에 대한 인밸리데이트 신호를 전송하도록 통신부를 제어할 수 있다(S630).

[0227] 본 개시에 따른 데이터 블록의 인밸리데이트(invalidate)는 캐시에 저장된 데이터 블록을 캐시에서 삭제하는 것을 의미할 수 있다.

[0228] 프로세서(210)는 적어도 하나의 프론트 엔드 노드 모두로부터 제 6 데이터 블록에 대한 인밸리데이트 신호에 대한 완료 신호를 수신할 수 있다(S640).

[0229] 프로세서(210)는 제 6 메타 데이터 정보를 백 엔드 캐시(240)에서 삭제할 수 있다(S650).

[0230] 상술한 바에 의하여, 제 6 메타 데이터 정보를 백 엔드 캐시(240)에서 삭제하면서, 이에 맞추어 적어도 하나의 프론트 엔드 노드(100)들에서 제 6 데이터 블록이 삭제되도록 할 수 있다.

[0231] 따라서, 백 엔드 캐시(240)에서의 저장 공간 확보가 요구되는 경우 필요성이 낮은 메타 데이터 정보부터 차례로 삭제할 수 있다. 이에 따라, 저장 공간의 효율적인 이용이 가능하다.

[0233] 도 10은 본 개시의 몇몇 실시예에 따른 프론트 엔드의 프로세서가 프론트 엔드 캐시에 데이터 블록을 저장하는 과정을 나타낸 순서도이다.

[0234] 도 10을 참조하면, 프로세서(110)는 수신된 셀렉트 쿼리와 관련된 데이터 블록이 프론트 엔드 캐시에 존재하는지 여부를 인식할 수 있다(S710).

[0235] 본 개시내용에서, 쿼리(query)는 백 엔드 노드에서의 처리를 요청하는 임의의 요청 또는 명령을 의미하며, 예를 들어, DML(Data Manipulation Language), DDL(Data Definition Language) 및/또는 PL/SQL 등을 포함할 수 있다. 또한, 본 개시내용에서의 쿼리는 사용자/개발자 등으로부터 발행되는 임의의 요청을 의미할 수 있다. 또

한, 쿼리는 프론트 엔드 노드 및/또는 백 엔드 노드에 인입되고 프론트 엔드 노드 및/또는 백 엔드 노드에서 처리되는 임의의 요청을 의미할 수 있다.

[0236] 예를 들면, 프로세서(110)는, 수신된 셀렉트 쿼리와 관련된 제 1 데이터 블록이 프론트 엔드 캐시(140)에 존재하는지 여부를 결정하기 위해, 프론트 엔드 캐시(140)를 탐색할 수 있다. 프로세서(110)는 프론트 엔드 캐시(140)에 제 1 데이터 블록이 존재하는 경우, 프론트 엔드 캐시에 제 1 데이터 블록이 존재한다고 결정할 수 있다. 한편, 프로세서(110)는 프론트 엔드 캐시(140)에 제 1 데이터 블록이 존재하지 않는 경우, 프론트 엔드 캐시에 제 1 데이터 블록이 존재하지 않는다고 결정할 수 있다.

[0237] 이는 프로세서(110)가 수신된 셀렉트 쿼리와 관련된 데이터 블록이 프론트 엔드 캐시(140)에 존재하는지 여부를 결정하는 일례이므로, 권리범위는 이에 한정되지 않는다.

[0238] 프로세서(110)는 셀렉트 쿼리와 관련된 데이터 블록이 프론트 엔드 캐시에 존재하지 않는 경우(S710, No), 백 엔드 노드로부터 수신된 셀렉트 쿼리와 관련된 데이터 블록을 수신할 수 있다(S720).

[0239] 상술한 바와 같이, 이 셀렉트 쿼리와 관련된 데이터 블록은 다른 프론트 엔드 노드(100) 또는 백 엔드 노드(200)로부터 수신된 것일 수 있다.

[0240] 이 때, 옵티마이저(130)는 수신된 셀렉트 쿼리에 관하여 프론트 엔드 캐시(140)의 저장 여부를 결정할 수 있다.

[0241] 프로세서(110)는, 셀렉트 쿼리와 관련된 데이터 블록이 프론트 엔드 캐시에 존재할 경우(S710, Yes), 수신된 셀렉트 쿼리와 관련된 데이터 블록을 반환할 수 있다(S730).

[0242] 전술한 바와 같이, 데이터 블록의 반환은 본 개시에 따른 셀렉트 쿼리를 요청하는 사용자에게 당해 셀렉트 쿼리와 관련된 데이터 블록의 정보를 제공하는 것을 의미할 수 있다.

[0243] 프로세서(110)는, 수신된 셀렉트 쿼리와 관련된 데이터 블록을 프론트 엔드 캐시에 저장할지 여부를 결정할 수 있다(S740).

[0244] 이 경우, 옵티마이저(130)는 셀렉트 쿼리와 관련된 스캔의 종류, 대상 세그먼트(segment)의 크기, 필터링 정도, 접근 빈도 등을 고려해 프론트 엔드 캐시(140)의 사용 여부를 결정할 수 있다.

[0245] 본 개시에 따른 세그먼트는, 디스크 저장공간을 사용하는 객체를 의미할 수 있다. 즉, 저장공간을 가지고 있는 임의의 오브젝트(object)는 세그먼트일 수 있다.

[0246] 세그먼트의 크기가 큰 경우, 셀렉트 쿼리와 관련된 데이터 블록이 아닌 잉여 블록이 많이 존재할 수 있다. 따라서 당해 세그먼트 전체를 프론트 엔드 캐시에 저장하는 것은 캐시 공간을 비효율적으로 사용할 수 있다. 따라서, 일반적으로, 옵티마이저(130)는 세그먼트의 크기가 큰 경우 프론트 엔드 캐시(140)에 당해 세그먼트를 저장하지 않도록 결정할 수 있다.

[0247] 필터링 정도는 사용자가 정의한 조건을 만족하는 데이터만 선택하는 기능을 의미할 수 있다. 옵티마이저(130)는 셀렉트 쿼리와 관련된 스캔의 종류가 테이블 풀 스캔이고 작은 테이블에 대하여 결과 필터링이 적다고 인식한 경우, 셀렉트 쿼리와 관련된 데이터 블록을 프론트 엔드 캐시(140)에 저장하기로 결정할 수 있다. 다만 이에 한정되는 것은 아니다.

[0248] 접근 빈도는 셀렉트 쿼리와 관련된 임의의 데이터 블록에 대한 단위 시간 당 접근 요청 횟수를 의미할 수 있다.

[0249] 예를 들어, 옵티마이저(130)는 셀렉트 쿼리와 관련된 임의의 데이터 블록에 대한 과거의 접근 빈도가 높을수록 프론트 엔드 캐시(140)에 당해 데이터 블록을 저장하도록 설정될 수 있다. 반대로, 옵티마이저(130)는 셀렉트 쿼리와 관련된 임의의 데이터 블록에 대한 과거의 접근 빈도가 낮을수록 프론트 엔드 캐시(140)에 당해 데이터 블록을 저장하도록 설정될 수 있다.

[0250] 상술한 바와 같이 프론트 엔드 캐시(140)를 유지함으로써, 자주 접근되는 데이터 블록에 대해서는 백 엔드 노드(200)에 데이터 블록을 요청할 필요 없이 바로 셀렉트 쿼리를 처리할 수 있다. 또한, 옵티마이저로 하여금 다양한 요소들을 고려함으로써 프론트 엔드 캐시에 저장되는 데이터 블록을 결정하도록 하면, 프론트 엔드 캐시가 더 효율적으로 사용될 수 있다. 따라서, 셀렉트 쿼리를 수행하는 데 필요한 네트워크 비용을 절감할 수 있다.

[0251] 도 11은 본 개시의 몇몇 실시예들이 구현될 수 있는 예시적인 컴퓨팅 환경에 대한 간략하고 일반적인 개략도를 도시한다.

[0252] 도 11에서 도시되는 컴퓨터(1102)는, 프론트 엔드 노드(100), 백 엔드 노드(200), 디스크(300) 및 통신 네트워

크(미도시)를 구성하는 컴퓨팅 장치 중 적어도 하나에 대응될 수 있다.

[0253] 본 개시내용이 일반적으로 하나 이상의 컴퓨터 상에서 실행될 수 있는 컴퓨터 실행가능 명령어와 관련하여 전술되었지만, 당업자라면 본 개시내용 기타 프로그램 모듈들과 결합되어 및/또는 하드웨어와 소프트웨어의 조합으로서 구현될 수 있다는 것을 잘 알 것이다.

[0254] 일반적으로, 본 명세서에서의 모듈은 특정의 태스크를 수행하거나 특정의 추상 데이터 유형을 구현하는 루틴, 프로시져, 프로그램, 컴포넌트, 데이터 구조, 기타 등등을 포함한다. 또한, 당업자라면 본 개시의 방법이 단일-프로세서 또는 멀티프로세서 컴퓨터 시스템, 미니컴퓨터, 메인프레임 컴퓨터는 물론 퍼스널 컴퓨터, 핸드헬드 컴퓨팅 장치, 마이크로프로세서-기반 또는 프로그램가능 가전 제품, 기타 등등(이들 각각은 하나 이상의 연관된 장치와 연결되어 동작할 수 있음)을 비롯한 다른 컴퓨터 시스템 구성으로 실시될 수 있다는 것을 잘 알 것이다.

[0255] 본 개시의 설명된 실시예들은 또한 어떤 태스크들이 통신 네트워크를 통해 연결되어 있는 원격 처리 장치들에 의해 수행되는 분산 컴퓨팅 환경에서 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 로컬 및 원격 메모리 저장 장치 둘다에 위치할 수 있다.

[0256] 컴퓨터는 통상적으로 다양한 컴퓨터 판독가능 매체를 포함한다. 컴퓨터에 의해 액세스 가능한 매체 로서, 휘발성 및 비휘발성 매체, 일시적(transitory) 및 비일시적(non-transitory) 매체, 이동식 및 비-이동식 매체를 포함한다. 제한이 아닌 예로서, 컴퓨터 판독가능 매체는 컴퓨터 판독가능 저장 매체 및 컴퓨터 판독가능 전송 매체를 포함할 수 있다.

[0257] 컴퓨터 판독가능 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보를 저장하는 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성 매체, 일시적 및 비-일시적 매체, 이동식 및 비이동식 매체를 포함한다. 컴퓨터 판독가능 저장 매체는 RAM, ROM, EEPROM, 플래시 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital video disk) 또는 기타 광 디스크 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 기타 자기 저장 장치, 또는 컴퓨터에 의해 액세스될 수 있고 원하는 정보를 저장하는 데 사용될 수 있는 임의의 기타 매체를 포함하지만, 이에 한정되지 않는다.

[0258] 컴퓨터 판독가능 전송 매체는 통상적으로 반송파(carrier wave) 또는 기타 전송 메커니즘(transport mechanism)과 같은 피변조 데이터 신호(modulated data signal)에 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터등을 구현하고 모든 정보 전달 매체를 포함한다. 피변조 데이터 신호라는 용어는 신호 내에 정보를 인코딩하도록 그 신호의 특성들 중 하나 이상을 설정 또는 변경시킨 신호를 의미한다. 제한이 아닌 예로서, 컴퓨터 판독가능 전송 매체는 유선 네트워크 또는 직접 배선 접속(direct-wired connection)과 같은 유선 매체, 그리고 음향, RF, 적외선, 기타 무선 매체와 같은 무선 매체를 포함한다. 상술된 매체들 중 임의의 것의 조합도 역시 컴퓨터 판독가능 전송 매체의 범위 안에 포함되는 것으로 한다.

[0259] 컴퓨터(1102)를 포함하는 본 개시의 여러가지 측면들을 구현하는 예시적인 환경(1100)이 나타내어져 있으며, 컴퓨터(1102)는 처리 장치(1104), 시스템 메모리(1106) 및 시스템 버스(1108)를 포함한다. 시스템 버스(1108)는 시스템 메모리(1106)(이에 한정되지 않음)를 비롯한 시스템 컴포넌트들을 처리 장치(1104)에 연결시킨다. 처리 장치(1104)는 다양한 상용 프로세서들 중 임의의 프로세서일 수 있다. 듀얼 프로세서 및 기타 멀티프로세서 아키텍처도 역시 처리 장치(1104)로서 이용될 수 있다.

[0260] 시스템 버스(1108)는 메모리 버스, 주변장치 버스, 및 다양한 상용 버스 아키텍처 중 임의의 것을 사용하는 로컬 버스에 추가적으로 상호 연결될 수 있는 몇 가지 유형의 버스 구조 중 임의의 것일 수 있다. 시스템 메모리(1106)는 판독 전용 메모리(ROM)(1110) 및 랜덤 액세스 메모리(RAM)(1112)를 포함한다. 기본 입/출력 시스템(BIOS)은 ROM, EPROM, EEPROM 등의 비휘발성 메모리(1110)에 저장되며, 이 BIOS는 시동 중과 같은 때에 컴퓨터(1102) 내의 구성요소들 간에 정보를 전송하는 일을 돕는 기본적인 루틴을 포함한다. RAM(1112)은 또한 데이터를 캐싱하기 위한 정적 RAM 등의 고속 RAM을 포함할 수 있다.

[0261] 컴퓨터(1102)는 또한 내장형 하드 디스크 드라이브(HDD)(1114)(예를 들어, EIDE, SATA) - 이 내장형 하드 디스크 드라이브(1114)는 또한 적당한 섀시(도시 생략) 내에서 외장형 용도로 구성될 수 있음 - , 자기 플로피 디스크 드라이브(FDD)(1116)(예를 들어, 이동식 디스켓(1118)으로부터 판독을 하거나 그에 기록을 하기 위한 것임), 및 광 디스크 드라이브(1120)(예를 들어, CD-ROM 디스크(1122)를 판독하거나 DVD 등의 기타 고용량 광 매체로부터 판독을 하거나 그에 기록을 하기 위한 것임)를 포함한다. 하드 디스크 드라이브(1114), 자기 디스크 드라이브(1116) 및 광 디스크 드라이브(1120)는 각각 하드 디스크 드라이브 인터페이스(1124), 자기 디스크 드라이브 인

터페이스(1126) 및 광 드라이브 인터페이스(1128)에 의해 시스템 버스(1108)에 연결될 수 있다. 외장형 드라이브 구현을 위한 인터페이스(1124)는 예를 들어, USB(Universal Serial Bus) 및 IEEE 1394 인터페이스 기술 중 적어도 하나 또는 그 둘다를 포함한다.

[0262] 이들 드라이브 및 그와 연관된 컴퓨터 판독가능 매체는 데이터, 데이터 구조, 컴퓨터 실행가능 명령어, 기타 등등의 비휘발성 저장을 제공한다. 컴퓨터(1102)의 경우, 드라이브 및 매체는 임의의 데이터를 적당한 디지털 형식으로 저장하는 것에 대응한다. 상기에서의 컴퓨터 판독가능 저장 매체에 대한 설명이 HDD, 이동식 자기 디스크, 및 CD 또는 DVD 등의 이동식 광 매체를 언급하고 있지만, 당업자라면 집 드라이브(zip drive), 자기 카세트, 플래쉬 메모리 카드, 카트리지, 기타 등등의 컴퓨터에 의해 판독가능한 다른 유형의 저장 매체도 역시 예시적인 운영 환경에서 사용될 수 있으며 또 임의의 이러한 매체가 본 개시의 방법들을 수행하기 위한 컴퓨터 실행가능 명령어를 포함할 수 있다는 것을 잘 알 것이다.

[0263] 운영 체제(1130), 하나 이상의 애플리케이션 프로그램(1132), 기타 프로그램 모듈(1134) 및 프로그램 데이터(1136)를 비롯한 다수의 프로그램 모듈이 드라이브 및 RAM(1112)에 저장될 수 있다. 운영 체제, 애플리케이션, 모듈 및/또는 데이터의 전부 또는 그 일부분이 또한 RAM(1112)에 캐싱될 수 있다. 본 개시가 여러가지 상업적으로 이용가능한 운영 체제 또는 운영 체제들의 조합에서 구현될 수 있다는 것을 잘 알 것이다.

[0264] 사용자는 하나 이상의 유선/무선 입력 장치, 예를 들어, 키보드(1138) 및 마우스(1140) 등의 포인팅 장치를 통해 컴퓨터(1102)에 명령 및 정보를 입력할 수 있다. 기타 입력 장치(도시 생략)로는 마이크, IR 리모콘, 조이스틱, 게임 패드, 스타일러스 펜, 터치 스크린, 기타 등등이 있을 수 있다. 이들 및 기타 입력 장치가 종종 시스템 버스(1108)에 연결되어 있는 입력 장치 인터페이스(1142)를 통해 처리 장치(1104)에 연결되지만, 병렬 포트, IEEE 1394 직렬 포트, 게임 포트, USB 포트, IR 인터페이스, 기타 등등의 기타 인터페이스에 의해 연결될 수 있다.

[0265] 모니터(1144) 또는 다른 유형의 디스플레이 장치도 역시 비디오 어댑터(1146) 등의 인터페이스를 통해 시스템 버스(1108)에 연결된다. 모니터(1144)에 부가하여, 컴퓨터는 일반적으로 스피커, 프린터, 기타 등등의 기타 주변 출력 장치(도시 생략)를 포함한다.

[0266] 컴퓨터(1102)는 유선 및/또는 무선 통신을 통한 원격 컴퓨터(들)(1148) 등의 하나 이상의 원격 컴퓨터로의 논리적 연결을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(들)(1148)는 워크스테이션, 서버 컴퓨터, 라우터, 퍼스널 컴퓨터, 휴대용 컴퓨터, 마이크로프로세서-기반 오락 기기, 피어 장치 또는 기타 통상의 네트워크 노드일 수 있으며, 일반적으로 컴퓨터(1102)에 대해 기술된 구성요소들 중 다수 또는 그 전부를 포함하지만, 간략함을 위해, 메모리 저장 장치(1150)만이 도시되어 있다. 도시되어 있는 논리적 연결은 근거리 통신망(LAN)(1152) 및/또는 더 큰 네트워크, 예를 들어, 원거리 통신망(WAN)(1154)에의 유선/무선 연결을 포함한다. 이러한 LAN 및 WAN 네트워킹 환경은 사무실 및 회사에서 일반적인 것이며, 인트라넷 등의 전사적 컴퓨터 네트워크(enterprise-wide computer network)를 용이하게 해주며, 이들 모두는 전세계 컴퓨터 네트워크, 예를 들어, 인터넷에 연결될 수 있다.

[0267] LAN 네트워킹 환경에서 사용될 때, 컴퓨터(1102)는 유선 및/또는 무선 통신 네트워크 인터페이스 또는 어댑터(1156)를 통해 로컬 네트워크(1152)에 연결된다. 어댑터(1156)는 LAN(1152)에의 유선 또는 무선 통신을 용이하게 해줄 수 있으며, 이 LAN(1152)은 또한 무선 어댑터(1156)와 통신하기 위해 그에 설치되어 있는 무선 액세스 포인트를 포함하고 있다. WAN 네트워킹 환경에서 사용될 때, 컴퓨터(1102)는 모뎀(1158)을 포함할 수 있거나, WAN(1154) 상의 통신 서버에 연결되거나, 또는 인터넷을 통하는 등, WAN(1154)을 통해 통신을 설정하는 기타 수단을 갖는다. 내장형 또는 외장형 및 유선 또는 무선 장치일 수 있는 모뎀(1158)은 직렬 포트 인터페이스(1142)를 통해 시스템 버스(1108)에 연결된다. 네트워크화된 환경에서, 컴퓨터(1102)에 대해 설명된 프로그램 모듈들 또는 그의 일부분이 원격 메모리/저장 장치(1150)에 저장될 수 있다. 도시된 네트워크 연결이 예시적인 것이며 컴퓨터들 사이에 통신 링크를 설정하는 기타 수단이 사용될 수 있다는 것을 잘 알 것이다.

[0268] 컴퓨터(1102)는 무선 통신으로 배치되어 동작하는 임의의 무선 장치 또는 개체, 예를 들어, 프린터, 스캐너, 데스크톱 및/또는 휴대용 컴퓨터, PDA(portable data assistant), 통신 위성, 무선 검출가능 태그와 연관된 임의의 장비 또는 장소, 및 전화와 통신을 하는 동작을 한다. 이것은 적어도 Wi-Fi 및 블루투스 무선 기술을 포함한다. 따라서, 통신은 종래의 네트워크에서와 같이 미리 정의된 구조이거나 단순하게 적어도 2개의 장치 사이의 애드혹 통신(ad hoc communication)일 수 있다.

[0269] Wi-Fi(Wireless Fidelity)는 유선 없이도 인터넷 등으로의 연결을 가능하게 해준다. Wi-Fi는 이러한 장치, 예를

들어, 컴퓨터가 실내에서 및 실외에서, 즉 기지국의 통화권 내의 아무 곳에서나 데이터를 전송 및 수신할 수 있게 해주는 셀 전화와 같은 무선 기술이다. Wi-Fi 네트워크는 안전하고 신뢰성 있으며 고속인 무선 연결을 제공하기 위해 IEEE 802.11(a,b,g, 기타)이라고 하는 무선 기술을 사용한다. 컴퓨터를 서로에, 인터넷에 및 유선 네트워크(IEEE 802.3 또는 이더넷을 사용함)에 연결시키기 위해 Wi-Fi가 사용될 수 있다. Wi-Fi 네트워크는 비인가 2.4 및 5 GHz 무선 대역에서, 예를 들어, 11Mbps(802.11a) 또는 54 Mbps(802.11b) 데이터 레이트로 동작하거나, 양 대역(듀얼 대역)을 포함하는 제품에서 동작할 수 있다.

[0270] 본 개시의 기술 분야에서 통상의 지식을 가진 자는 여기에 개시된 실시예들과 관련하여 설명된 다양한 예시적인 논리 블록들, 모듈들, 프로세서들, 수단들, 회로들 및 알고리즘 단계들이 전자 하드웨어, (편의를 위해, 여기에서 "소프트웨어"로 지칭되는) 다양한 형태들의 프로그램 또는 설계 코드 또는 이들 모두의 결합에 의해 구현될 수 있다는 것을 이해할 것이다. 하드웨어 및 소프트웨어의 이러한 상호 호환성을 명확하게 설명하기 위해, 다양한 예시적인 컴포넌트들, 블록들, 모듈들, 회로들 및 단계들이 이들의 기능과 관련하여 위에서 일반적으로 설명되었다. 이러한 기능이 하드웨어 또는 소프트웨어로서 구현되는지 여부는 특정한 애플리케이션 및 전체 시스템에 대하여 부과되는 설계 제약들에 따라 좌우된다. 본 개시의 기술 분야에서 통상의 지식을 가진 자는 각각의 특정한 애플리케이션에 대하여 다양한 방식들로 설명된 기능을 구현할 수 있으나, 이러한 구현 결정들은 본 개시의 범위를 벗어나는 것으로 해석되어서는 안 될 것이다.

[0271] 여기서 제시된 다양한 실시예들은 방법, 장치, 또는 표준 프로그래밍 및/또는 엔지니어링 기술을 사용한 제조 물품(article)으로 구현될 수 있다. 용어 "제조 물품"은 임의의 컴퓨터-판독가능 장치로부터 액세스 가능한 컴퓨터 프로그램, 캐리어, 또는 매체(media)를 포함한다. 예를 들어, 컴퓨터-판독가능 저장 매체는 자기 저장 장치(예를 들면, 하드 디스크, 플로피 디스크, 자기 스트립, 등), 광학 디스크(예를 들면, CD, DVD, 등), 스마트 카드, 및 플래쉬 메모리 장치(예를 들면, EEPROM, 카드, 스틱, 키 드라이브, 등)를 포함하지만, 이들로 제한되는 것은 아니다. 용어 "기계-판독가능 매체"는 명령(들) 및/또는 데이터를 저장, 보유, 및/또는 전달할 수 있는 무선 채널 및 다양한 다른 매체를 포함하지만, 이들로 제한되는 것은 아니다.

[0272] 제시된 프로세스들에 있는 단계들의 특정한 순서 또는 계층 구조는 예시적인 접근들의 일례임을 이해하도록 한다. 설계 우선순위들에 기반하여, 본 개시의 범위 내에서 프로세스들에 있는 단계들의 특정한 순서 또는 계층 구조가 재배열될 수 있다는 것을 이해하도록 한다. 첨부된 방법 청구항들은 샘플 순서로 다양한 단계들의 엘리먼트들을 제공하지만 제시된 특정한 순서 또는 계층 구조에 한정되는 것을 의미하지는 않는다.

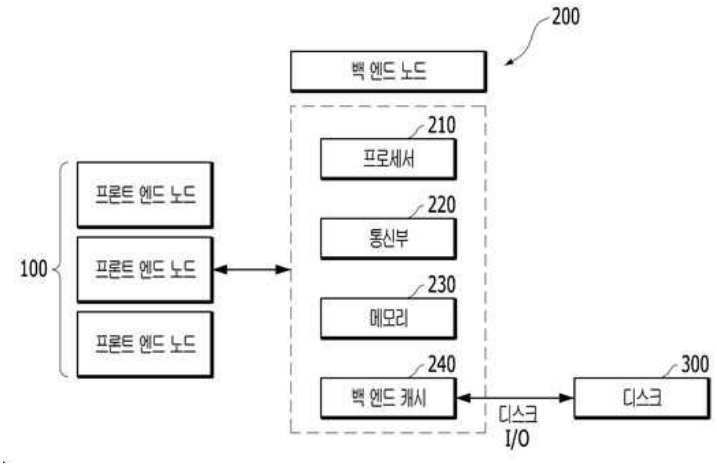
[0273] 본 개시의 목적 및 효과, 그리고 그것들을 달성하기 위한 기술적 구성들은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 본 개시를 설명하는데 있어서 공지 기능 또는 구성에 대한 구체적인 설명이 본 개시의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 그리고 후술되는 용어들은 본 개시에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다.

[0274] 그러나 본 개시는 이하에서 개시되는 실시예들에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있다. 단지 본 실시예들은 본 개시가 완전하도록 하고, 본 개시가 속하는 기술분야에서 통상의 지식을 가진 자에게 개시의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 개시는 청구항의 범주에 의해 정의될 뿐이다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

[0275] 제시된 실시예들에 대한 설명은 임의의 본 개시의 기술 분야에서 통상의 지식을 가진 자가 본 개시를 이용하거나 또는 실시할 수 있도록 제공된다. 이러한 실시예들에 대한 다양한 변형들은 본 개시의 기술 분야에서 통상의 지식을 가진 자에게 명백할 것이며, 여기에 정의된 일반적인 원리들은 본 개시의 범위를 벗어남이 없이 다른 실시예들에 적용될 수 있다. 그리하여, 본 개시는 여기에 제시된 실시예들로 한정되는 것이 아니라, 여기에 제시된 원리들 및 신규한 특징들과 일관되는 최광의의 범위에서 해석되어야 할 것이다.

도면

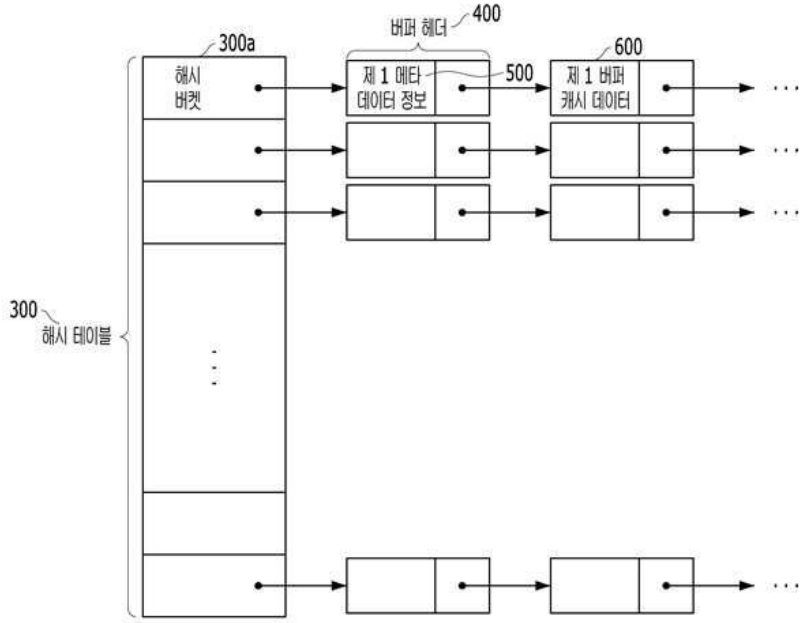
도면1


200
백 엔드 노드
210
프로세서
220
통신부
230
메모리
240
백 엔드 캐시
300
디스크
디스크
I/O
100
프론트 엔드 노드
프론트 엔드 노드
프론트 엔드 노드

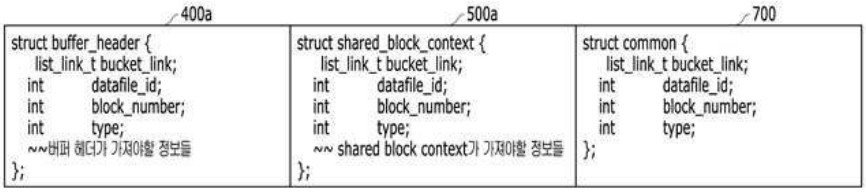
도면2


100
프론트 엔드 노드
110
프로세서
120
통신부
130
옵티마이저
140
프론트 엔드 캐시

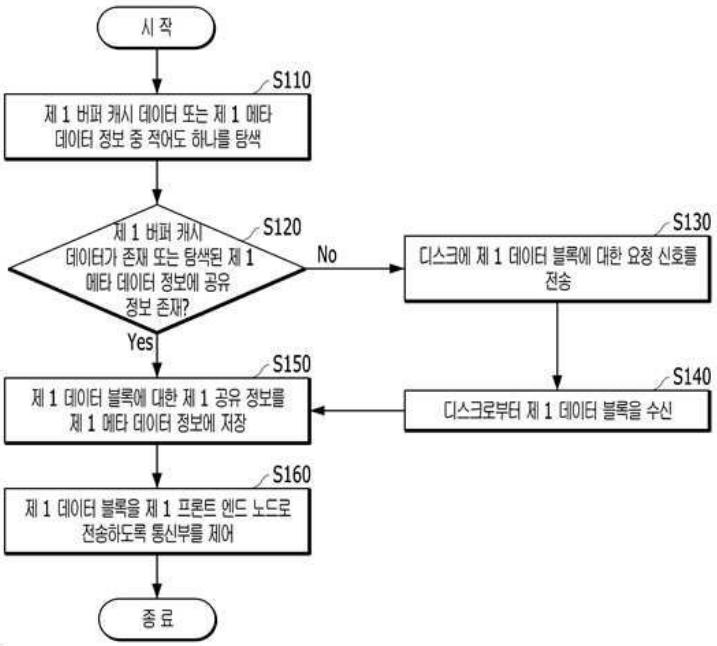
도면3a



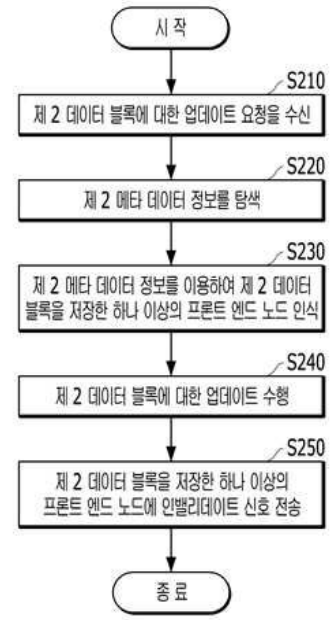
도면3b



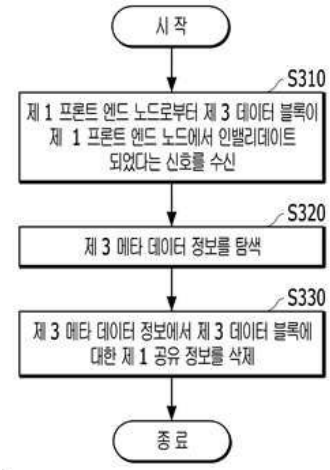
도면4



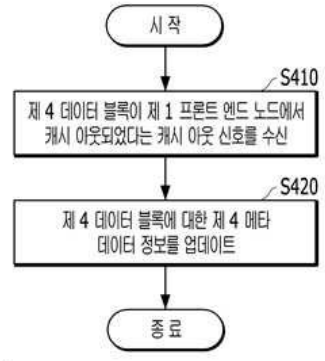
도면5


시작
S210
제 2 데이터 블록에 대한 업데이트 요청을 수신
S220
제 2 메타 데이터 정보를 탐색
S230
제 2 메타 데이터 정보를 이용하여 제 2 데이터 블록을 저장한 하나 이상의 프론트 엔드 노드 인식
S240
제 2 데이터 블록에 대한 업데이트 수행
S250
제 2 데이터 블록을 저장한 하나 이상의 프론트 엔드 노드에 인밸리데이트 신호 전송
종료

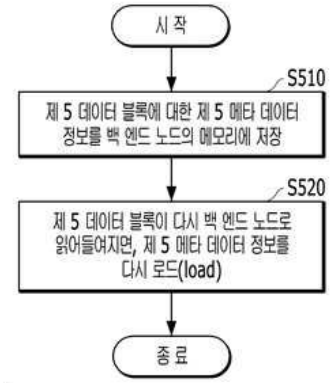
도면6


시작
S310
제 1 프론트 엔드 노드로부터 제 3 데이터 블록이 제 1 프론트 엔드 노드에서 인밸리데이트 되었다는 신호를 수신
S320
제 3 메타 데이터 정보를 탐색
S330
제 3 메타 데이터 정보에서 제 3 데이터 블록에 대한 제 1 공유 정보를 삭제
종료

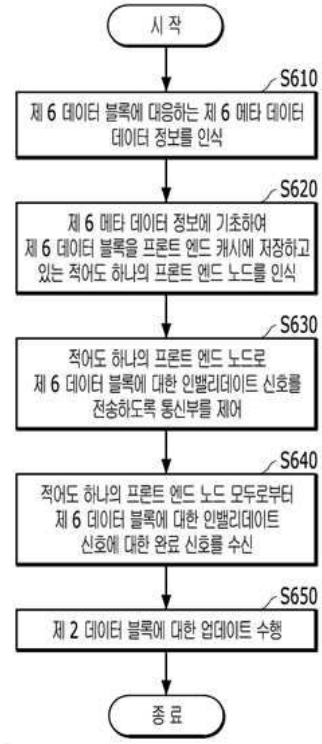
도면7


시작
S410
제 4 데이터 블록이 제 1 프론트 엔드 노드에서 캐시 아웃되었다는 캐시 아웃 신호를 수신
S420
제 4 데이터 블록에 대한 제 4 메타 데이터 정보를 업데이트
종료

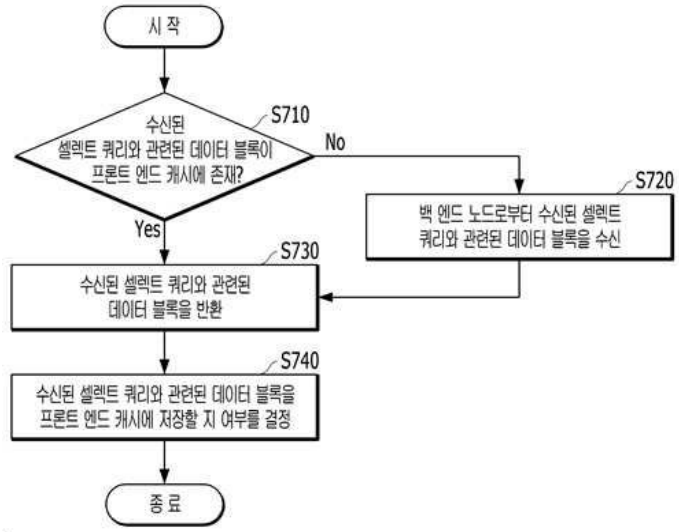
도면8


시작
S510
제 5 데이터 블록에 대한 제 5 메타 데이터 정보를 백 엔드 노드의 메모리에 저장
S520
제 5 데이터 블록이 다시 백 엔드 노드로 읽어들여지면, 제 5 메타 데이터 정보를 다시 로드(load)
종료

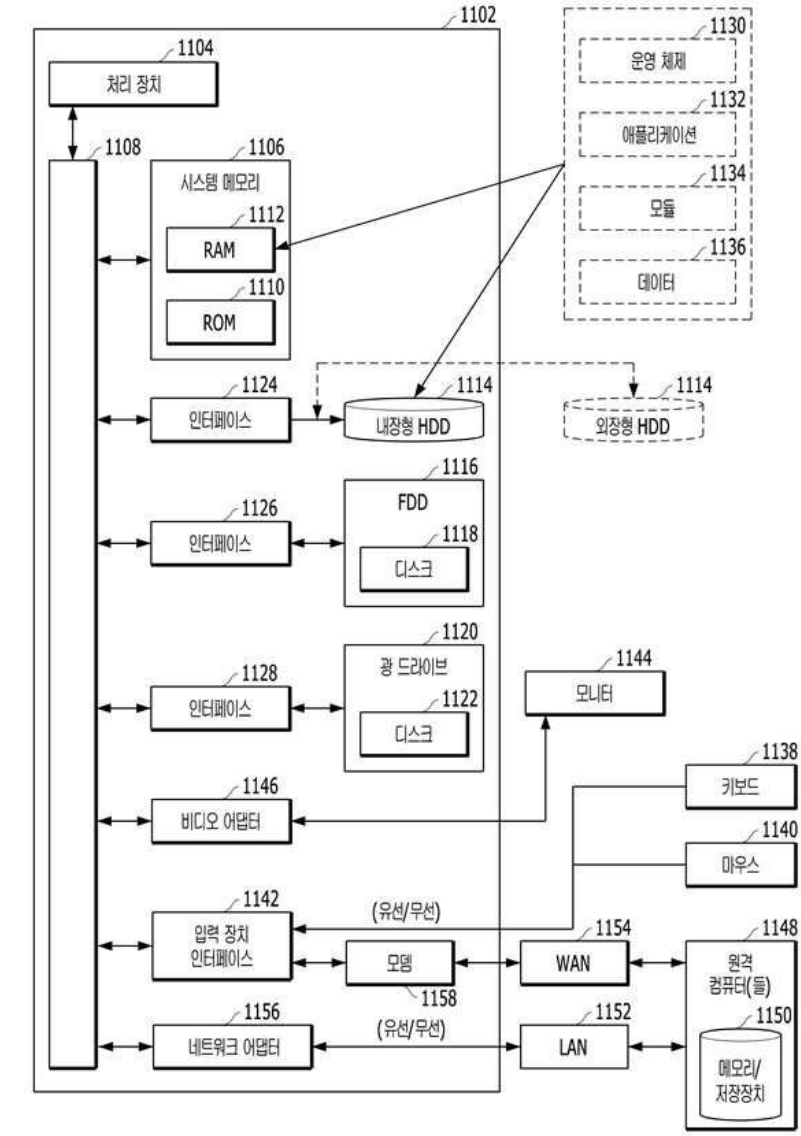
도면9


시작
S610
제 6 데이터 블록에 대응하는 제 6 메타 데이터 데이터 정보를 인식
S620
제 6 메타 데이터 정보에 기초하여 제 6 데이터 블록을 프론트 엔드 캐시에 저장하고 있는 적어도 하나의 프론트 엔드 노드를 인식
S630
적어도 하나의 프론트 엔드 노드로 제 6 데이터 블록에 대한 인밸리데이트 신호를 전송하도록 통신부를 제어
S640
적어도 하나의 프론트 엔드 노드 모두로부터 제 6 데이터 블록에 대한 인밸리데이트 신호에 대한 완료 신호를 수신
S650
제 2 데이터 블록에 대한 업데이트 수행
종료

도면10


시 작
수신된
셀렉트 쿼리와 관련된 데이터 블록이
프론트 엔드 캐시에 존재?
S710
No
Yes
S720
백 엔드 노드로부터 수신된 셀렉트
쿼리와 관련된 데이터 블록을 수신
S730
수신된 셀렉트 쿼리와 관련된
데이터 블록을 반환
S740
수신된 셀렉트 쿼리와 관련된 데이터 블록을
프론트 엔드 캐시에 저장할 지 여부를 결정
종 료

도면11


1102
1104
처리 장치
1108
1106
시스템 메모리
1112
RAM
1110
ROM
1130
운영 체제
1132
애플리케이션
1134
모듈
1136
데이터
1124
인터페이스
1114
내장형 HDD
1114
외장형 HDD
1116
FDD
1126
인터페이스
1118
디스크
1120
광 드라이브
1128
인터페이스
1122
디스크
1144
모니터
1138
키보드
1146
비디오 어댑터
1140
마우스
1142
입력 장치 인터페이스
(유선/무선)
모뎀
1158
1154
WAN
1148
원격 컴퓨터(들)
1156
네트워크 어댑터
(유선/무선)
1152
LAN
1150
메모리/ 저장장치