



- (51) 국제특허분류: *G06F 17/30* (2006.01) *G06F 9/48* (2006.01)
- (21) 국제출원번호: PCT/KR2015/005851
- (22) 국제출원일: 2015 년 6 월 10 일 (10.06.2015)
- (25) 출원언어: 한국어
- (26) 공개언어: 한국어
- (71) 출원인: 울산과학기술원 (UNIST(ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY))
[KR/KR]; 44919 울산광역시 울주군 언양읍 유니스트길 50, Ulsan (KR).
- (72) 발명자: 남범석 (NAM, Beom Seok); 689-798 울산시 울주군 언양읍 유니스트길 50, 국립대학법인 울산과학기술대학교 산학협력단, Ulsan (KR).
- (74) 대리인: 한상민 (HAN, Sang Min) 등; 08502 서울시 금천구 가산디지털1로 196 303 호, Seoul (KR).
- (81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,

CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ,
LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK,
MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA,
PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD,
SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR,
TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

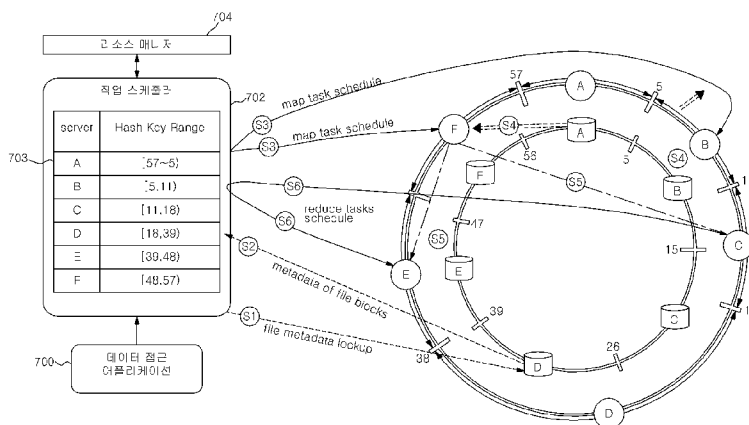
- (84) **지정국** (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

공개:

— 국제조사보고서와 함께 (조약 제 21 조(3))

- (54) Title:** CODE DISPERSION HASH TABLE-BASED MAP-REDUCE SYSTEM AND METHOD

- (54) 발명의 명칭 : 코드 분산 해시테이블 기반의 맵리듀스 시스템 및 방법



```

700 ... Date access application
702 ... Task scheduler
704 ... Resource manager
S1 ... File metadata lookup
S2 ... Metadata of file blocks
S3 ... Map task schedule
S6 ... Reduce tasks schedule

```

(57) Abstract: According to the present invention, a code dispersion hash table-based map-reduce system can achieve load balancing and increase a cache hit rate by managing data in a ring structure of dual-layers including an in-memory cache layer and a file system layer on the basis of a code dispersion hash table, predicting probability distribution of a data access request according to a data access request frequency of a user, adjusting a hash-key range of the code dispersion hash table of the in-memory cache layer on the basis of the predicted probability distribution, and scheduling a task.

(57) 요약서: 본 발명에 따르면, 코드 분산 해쉬 테이블 기반의 맵리듀스 시스템에서, 코드 분산 해시 테이블 기반의 파일 시스템 계층과 인메모리 캐시 계층의 이중계층의 링구조로 데이터를 관리하고 사용자의 데이터 접근 요청 빈도에 따라 데이터 접근 요청의 확률분포를 예측한 후, 예측된 확률분포를 기반으로 인메모리 캐시 계층의 코드 분산해시 테이블의 해시키 범위를 조정하고 테스트를 스케줄링함으로써 로드 밸런싱이 가능하며, 캐시 히트율을 높일 수 있다.

명세서

발명의 명칭: 코드 분산 해시테이블 기반의 맵리듀스 시스템 및 방법

기술분야

- [1] 본 발명은 분산 파일 시스템에 관한 것으로, 특히 코드 분산 해시테이블 기반의 파일 시스템 계층과 인메모리 캐시 계층의 이중계층 링구조(double-layered ring structure)로 데이터를 관리하고 사용자의 데이터 접근 요청 빈도에 따라 데이터 접근 요청의 확률분포를 예측한 후, 예측된 확률분포를 기반으로 인메모리 캐시 계층의 코드 분산해시테이블(chord distributed hash table)의 해시키 범위(hash key range)를 조정하고 테스크(task)를 스케줄링함으로써 로드 밸런싱(load balancing)이 가능하며, 캐시 히트율을 높일 수 있는 코드 분산 해시테이블 기반의 맵리듀스 시스템 및 방법에 관한 것이다.

배경기술

- [2] 클라우드 컴퓨팅(cloud computing)이란 다수의 컴퓨터를 하나의 집합체로 묶어 가상의 연산집단인 클라우드를 구성하는 것으로, 데이터의 저장이나 연산을 개개의 컴퓨터에게 맡기는 것이 아니라 컴퓨터들의 집합체인 클라우드에게 위임하는 개념을 뜻한다. 클라우드 컴퓨팅은 여러 분야에서 활용되고 있으며 최근 가장 큰 쓰임새로 주목 받고 있는 분야는 '빅데이터' 분야이다.
- [3] 이러한, 빅데이터는 대량의 테라바이트(Tera Byte)를 넘어 페타바이트(Peta Byte)나 그 이상의 크기를 가지는 데이터를 의미하는 것으로, 단일 컴퓨터로 처리하는 것이 불가능하기 때문에 클라우드 컴퓨팅은 빅데이터 처리의 기본 플랫폼으로 여겨지고 있다.
- [4] 한편, 빅데이터를 위한 클라우드 컴퓨팅 환경 중 대표적으로 Java 언어를 기반으로 한 다중 컴퓨터 클라우드 환경인 아파치(Apache)의 하둡(Hadoop) 시스템 등이 있다.
- [5] 이러한 하둡은 입력되는 데이터를 나누어 처리할 수 있도록 하는 하둡 분산 파일 시스템(Hadoop Distributed File System : HDFS)을 포함하며, 분산 저장된 데이터들은 대용량 데이터를 클러스터 환경에서 고속 병렬 처리하기 위해 개발된 맵리듀스(mapreduce)과정에 의해 처리된다.
- [6] 그러나, 하둡에서 적용되고 있는 하둡 분산 파일 시스템은 중앙 파일 시스템으로써 중앙에 디렉토리를 관리하는 매니저가 있으며, 디렉토리를 관리하는 매니저가 각 서버에서 어떠한 데이터를 가지고 있는지에 대한 모든 관리를 수행하도록 하고 있어 너무 많은 데이터를 관리함에 따라 성능이 떨어지는 문제점이 있다.
- [7] 또한, 하둡 분산 파일 시스템에서는 파일안에 어떠한 내용이 있는지는 전혀 고려하지 않고 파일을 적당히 분할하여 여러 개의 서버에 분산 저장하도록 하고

있어 특정 서버에만 필요한 데이터가 몰려서 저장되어 있을 수 있고, 어떤 서버는 저장된 데이터가 아예 없을 수도 있다.

- [8] 따라서, 하둡에서 맵리듀스 작업 요청이 수신되는 경우 맵리듀스 작업 요청 처리에 필요한 특정 데이터가 많이 몰려 있는 서버에서만 맵이라는 함수가 여러개 돌아가게 되어 로드 밸런스가 맞지 않게 됨으로써 성능이 저하되는 문제점이 있다.

발명의 상세한 설명

기술적 과제

- [9] 따라서, 본 발명에서는 코드 분산 해시테이블 기반의 파일 시스템 계층과 인메모리 캐시 계층의 이중계층의 링구조로 데이터를 관리하고 사용자의 데이터 접근 요청 빈도에 따라 데이터 접근 요청의 확률분포를 예측한 후, 예측된 확률분포를 기반으로 인메모리 캐시 계층의 코드 분산해시테이블의 해시키 범위를 조정하고 테스크를 스케줄링함으로써 로드 밸런싱이 가능하며, 캐시 히트율을 높일 수 있는 코드 분산 해시테이블 기반의 맵리듀스 시스템 및 방법을 제공하고자 한다.

과제 해결 수단

- [10] 상술한 본 발명은 코드 분산 해시테이블 기반의 맵리듀스 시스템으로서, 코드 분산 해시테이블을 기반으로 데이터를 저장하는 파일 시스템과 인메모리 캐시를 구비하는 다수의 서버와, 상기 파일 시스템과 인메모리 캐시에 저장된 데이터를 이중계층의 링구조로 관리하며, 외부로부터 특정 파일에 대한 데이터 접근 요청 수신 시 상기 다수의 서버 중 상기 데이터 접근 요청된 파일을 저장하고 있는 서버들로 맵리듀스 테스크를 할당하여 상기 맵리듀스 테스크 수행된 결과값을 상기 데이터 접근 요청에 대한 응답으로 출력하는 작업 스케줄러를 포함한다.
- [11] 또한, 상기 작업 스케줄러는, 상기 데이터 접근 요청 수신 시, 상기 파일의 이름을 통해 해시키를 추출하고 각 서버의 인메모리 캐시에 할당된 해시키 범위를 확인하여 상기 파일을 저장하고 있는 서버를 검색하며, 상기 검색된 서버로부터 상기 파일에 대한 메타데이터를 수신하여 상기 파일이 저장된 서버들로 상기 맵리듀스 테스크를 할당하는 것을 특징으로 한다.
- [12] 또한, 상기 작업 스케줄러는, 상기 파일에 대한 데이터 블록 구조와 각 데이터 블록이 분산 저장된 서버의 정보를 상기 메타데이터로 수신하며, 상기 각 데이터 블록이 저장된 서버들로 상기 맵리듀스 테스크를 할당하는 것을 특징으로 한다.
- [13] 또한, 상기 인메모리 캐시는, 상기 코드 분산 해시테이블을 이용하여 데이터에 대응되는 해시키를 저장하고 있으며, 상기 해시키의 저장에 있어서 상기 인메모리 캐시에 할당된 해시키 범위에 해당하는 해시키와 상기 해당 해시키에 대응하는 데이터를 저장하고 있는 것을 특징으로 한다.
- [14] 또한, 상기 작업 스케줄러는, 각 서버의 상기 인메모리 캐시의 해시키 범위를 각

- 서버에 대한 데이터 접근 요청 빈도에 따라 서버별로 동적으로 변경 설정하는 것을 특징으로 한다.
- [15] 또한, 상기 작업 스케줄러는, 상기 파일의 각 데이터 블록에 대한 상기 맵리듀스 태스크 처리 과정에서 발생하는 중간계산결과를 상기 파일 시스템에 저장하는 것을 특징으로 한다.
- [16] 또한, 상기 중간계산결과는, 상기 각 데이터 블록에 따라 서로 다른 해시키를 가지도록 생성되며, 각각 다른 서버로 분산되는 것을 특징으로 한다.
- [17] 또한, 상기 중간계산결과는, 상기 인메모리 캐시의 중간 결과물 재사용 캐시 영역에 저장되는 것을 특징으로 한다.
- [18] 또한, 상기 시스템은, 상기 작업 스케줄러와 연동되며, 새로운 서버의 추가, 제거, 복구를 관리하거나 파일의 업로드를 관리하는 리소스 매니저를 더 포함하는 것을 특징으로 한다.
- [19] 또한, 본 발명은 파일 시스템과 인메모리 캐시를 구비하는 다수의 서버와 상기 서버로 맵리듀스 태스크를 할당하는 작업 스케줄러를 포함하는 코드 분산 해시테이블 기반의 맵리듀스 시스템에서 맵리듀스 태스크 수행방법에 있어서, 상기 작업 스케줄러에서 상기 파일 시스템과 인메모리 캐시에 저장된 데이터를 이중계층의 링구조로 관리하는 단계와, 외부로부터 특정 파일에 대한 데이터 접근 요청을 수신하는 단계와, 상기 파일에 대한 해시키를 추출하여 상기 파일을 저장하고 있는 파일 시스템의 서버를 검색하는 단계와, 상기 검색된 서버로부터 상기 파일에 대한 데이터 블록 구조와 각 데이터 블록의 분산 저장된 서버의 정보를 메타데이터로 수신하는 단계와, 상기 각 데이터 블록이 저장된 서버들로 맵리듀스 태스크를 할당하는 단계와, 상기 맵리듀스 태스크 수행된 결과값을 상기 작업 요청에 대한 응답으로 출력하는 단계를 포함한다.
- [20] 또한, 상기 파일 시스템과 인메모리 캐시는, 코드 분산 해시테이블을 기반으로 상기 데이터를 저장하는 것을 특징으로 한다.
- [21] 또한, 상기 인메모리 캐시는, 상기 코드 분산 해시테이블을 이용하여 데이터에 대응되는 해시키를 저장하고 있으며, 기설정된 해시키 범위를 할당받아, 상기 해시키 범위에 해당하는 해시키와 상기 해당 해시키에 대응하는 데이터를 저장하고 있는 것을 특징으로 한다.
- [22] 또한, 상기 해시키 범위는, 각 서버에 대한 데이터 접근 요청 빈도에 따라 서버별로 동적으로 변경 설정되는 것을 특징으로 한다.
- [23] 또한, 상기 맵리듀스 태스크는, 상기 각 데이터 블록이 저장된 서버들에서 처리되며, 상기 맵리듀스 태스크의 처리 과정에서 발생하는 중간계산결과가 상기 파일 시스템에 저장되는 것을 특징으로 한다.
- [24] 또한, 상기 중간계산결과는, 상기 각 데이터 블록에 따라 서로 다른 해시키를 가지도록 생성되며, 각각 다른 서버로 분산되는 것을 특징으로 한다.
- [25] 또한, 상기 중간계산결과는, 상기 인메모리 캐시의 중간 결과물 재사용 캐시 영역에 저장되는 것을 특징으로 한다.

발명의 효과

- [26] 본 발명에 따르면, 코드 분산 해시테이블 기반의 파일 시스템 계층과 인메모리 계층의 이중계층의 링구조로 데이터를 관리하고 사용자의 데이터 접근 요청 빈도에 따라 데이터 접근 요청의 확률분포를 예측한 후, 예측된 확률분포를 기반으로 인메모리 계층의 코드 분산해시테이블의 해시키 범위를 조정하고 태스크를 스케줄링함으로써 로드 밸런싱이 가능하며, 캐시 히트율을 높일 수 있는 이점이 있다.
- [27] 또한, 본 발명에 따르면, 중앙제어식 분산파일시스템 대신에 코드 분산파일시스템(Chord distributed file system)을 사용하고, 코드 분산파일시스템에서 각 서버는 코드 라우팅 테이블을 관리하면서 중앙에서 관리되는 메타데이터(metadata)를 통하지 않고 바로 원격(remote)의 파일에 접근할 수 있기 때문에 확장성이 보장되는 이점이 있다.
- [28] 또한, 분산메모리 환경을 적극적으로 활용하는 인메모리 캐시를 사용하고, 코드 분산해시테이블을 이용하여 키밸류 데이터를 인덱싱하며, 입력 데이터뿐만 아니라 맵태스크의 결과로써 생성된 중간계산결과도 인메모리 캐시에 저장함으로써 캐시 히트율을 높일 수 있다.
- [29] 또한, 인메모리 캐시의 인덱싱은 파일 시스템을 관리하는 코드 분산해시테이블과 독립적으로 관리되어, 해시키 범위를 데이터 요청의 빈도에 따라 유동적으로 조절함으로써 서버마다 고른 데이터 접근이 이루어질 수 있도록 할 수 있는 이점이 있다.
- [30] 또한, 지역성인식 페어 스케줄링 알고리즘을 적용하여 작업 스케줄러가 분산 해시키 범위를 토대로 어떤 서버의 인메모리 캐시에 필요한 데이터가 저장되어 있을지를 확인하여 데이터들을 재사용할 수 있도록 스케줄링하고, 만약 데이터 요청이 특정 데이터에 편향되어 있는 경우, 해시키 범위를 조절함으로써 모든 서버에 데이터 접근이 고루 이루어질 수 있도록 하는 이점이 있다.

도면의 간단한 설명

- [31] 도 1은 본 발명의 실시예에 따른 코드 분산 파일 시스템의 동작 개념도,
 [32] 도 2는 본 발명의 실시예에 따른 코드 분산 해시테이블 기반의 파일 시스템과 인메모리의 이중계층 링구조로 데이터를 관리하는 개념도,
 [33] 도 3 내지 도 5는 본 발명의 실시예에 따른 데이터 접근 확률 분포 그래프 예시도,
 [34] 도 6은 본 발명의 실시예에 따른 누적 확률분포 그래프 예시도,
 [35] 도 7은 본 발명의 실시예에 따른 코드 분산 해시테이블 기반의 맵리듀스 시스템에서 맵리듀스 태스크 수행 동작 개념도.

발명의 실시를 위한 최선의 형태

- [36] 이하, 첨부된 도면을 참조하여 본 발명의 동작 원리를 상세히 설명한다. 하기에서 본 발명을 설명함에 있어서 공지 기능 또는 구성에 대한 구체적인

설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 그리고 후술되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

- [37] 먼저, 본 발명에서는 하둡 등과 같은 종래 중앙제어식 분산 파일 시스템 대신에 코드 분산 파일 시스템(chord distributed file system)을 사용하였다. 이러한 코드 분산 파일 시스템에서 각 서버는 코드 라우팅 테이블을 관리하면서 중앙에서 관리되는 메타데이터(metadata)를 통하지 않고 바로 원격의 파일에 접근할 수 있기 때문에 확장성이 보장될 수 있다.
- [38] 도 1은 본 발명의 실시예에 따른 코드 분산 해시테이블(chord distributed hash table : DHT) 기반의 맵리듀스(mapreduce) 시스템에 적용된 파일 시스템의 개념을 도시한 것이다.
- [39] 이하, 위 도 1을 참조하여 본 발명의 코드 분산 파일 시스템에서의 동작을 보다 상세히 설명하기로 한다.
- [40] 종래 중앙 제어식 분산파일 시스템에서는 중앙 디렉토리에 각각의 노드에서 가지고 있는 데이터에 대한 정보를 모두 저장하고 있는 반면에 본 발명의 코드 분산 파일 시스템에서는 각각의 노드에서 이웃 노드에 대한 정보를 가지도록 구현된다.
- [41] 즉, 각각의 노드는 도 1에 도시된 바와 같이 화살표로 표시된 바로 옆의 인접 노드와 인접 노드로부터 다시 2씩 곱한 이웃 노드들에 대한 정보를 가지도록 구현될 수 있다. 예를 들어 해시키가 0번(100)인 노드 바로 옆의 이웃 노드인 해시키가 1인 노드(102)와 해시키가 2인 노드(104), 해시키가 4인 노드(106), 해시키가 8인 노드(114)에 대한 정보를 알 수 있다.
- [42] 이때, 위와 같이 구현되는 본 발명의 코드 분산 파일 시스템에서 해시키가 0인 노드(100)로 예를 들어 해시키 7에 대응되는 데이터에 대한 사용자 요청 메시지가 수신되는 경우, 해시키가 0인 노드(100)에서는 해시키 7를 가지고 있는 노드(112)에 대한 정보를 알지 못하므로, 자신이 알고 있는 도 1에서 도시된 바와 같은 화살표로 연결되어 있는 해시키가 1인 노드(102), 해시키가 2인 노드(104), 해시키가 4인 노드(106), 해시키가 8인 노드(114) 중, 해시키가 4인 노드(106)로 사용자 요구 메시지를 전송한다. 이때, 해시키가 0인 노드(100)는 자신이 알고 있는 해시키 중 해시키 8은 7을 넘어버리므로 7보다 작은 해시키 중 가장 큰 해시키 4를 가지고 있는 노드(106)로 사용자 요구 메시지를 전송하게 되는 것이다.
- [43] 그러면, 해시키가 4인 노드(106)에서는 다시 자신이 알고 있는 화살표로 연결되어 있는 해시키가 5인 노드(108), 해시키가 6인 노드(110), 해시키가 8인 노드(114), 해시키가 12인 노드(116) 중 해시키가 6인 노드(110)로 사용자 요구 메시지를 전송한다. 이때, 해시키가 4인 노드(106)는 자신이 알고 있는 해시키중

- 해시키 12는 7을 넘어버리므로 7보다 작은 해시키 중 가장 큰 해시키 6을 저장하고 있는 노드(110)로 사용자 요구 메시지를 전송하게 되는 것이다.
- [44] 이어, 다시 사용자 요구 메시지를 수신한 해시키가 6인 노드(110)에서도 자신이 알고 있는 노드 중 사용자 요구 메시지를 전송할 노드를 검색하게 되는데, 이때, 해시키가 6인 노드(110)에서는 사용자 요구 메시지에 대응되는 해시키 7을 저장하고 있는 노드(112)를 알고 있으므로, 해시키 7을 저장하고 있는 노드(112)로 사용자 요구 메시지를 전송하게 된다.
- [45] 위와 같은 방식을 통해 해시키 7에 대응되는 데이터를 요구하는 사용자 요구 메시지를 최초로 수신한 해시키 0을 저장하고 있는 노드(100)에서 데이터를 읽어올 수 있게 되는 것이다.
- [46] 도 2는 본 발명의 실시예에 따른 코드 분산 해시테이블 기반의 파일 시스템과 인메모리 캐시의 이중계층 링구조로 데이터를 관리하는 개념을 도시한 것이다.
- [47] 위 도 2를 참조하면, 본 발명에서는 하둡 등에서의와 같은 중앙 디렉토리가 존재하지 않으며, 각각의 서버에서 자신의 피어(peer)들에 대한 정보만 가지고서 파일 관리를 하게 되며, 이러한 파일 관리 기법을 파일 시스템에만 적용한 것이 아니라 파일 시스템과 연동되는 캐시메모리인 인메모리 캐시에도 적용하여 이중계층의 링구조로 파일을 관리하게 된다.
- [48] 이하, 도 2를 참조하여 코드 분산 해시테이블 기반의 코드 분산 파일 시스템에서의 동작을 보다 상세히 설명하기로 한다.
- [49] 먼저, 각각의 서버에서 관리하는 분산 파일 시스템은 도 2에서 보여지는 바와 같이 파일 시스템(202, 206, 210, 214, 218, 222)과 인메모리 캐시(in memory)(200, 204, 208, 212, 216, 220)로 구성될 수 있으며, 본 발명에서는 이러한 다수의 서버의 파일 시스템과 인메모리 캐시를 이중계층의 링구조로 연결하여 사용자로부터의 데이터 접근(access) 요청을 처리할 수 있다. 이때, 이러한 파일 시스템은 하드 디스크(hard disk)의 등의 대용량 저장장치를 말할 수 있으며, 인메모리 캐시는 캐시 메모리(cache memory) 등을 말할 수 있으나, 이에 한정되는 것은 아니다.
- [50] 또한, 본 발명에서는 각 서버의 파일 시스템과 인메모리 캐시상에 해시키 범위(hash key range)를 할당하고, 이러한 해시키에 대한 정보를 후술하는 작업 스케줄러(job scheduler)에서 관리하도록 하여 사용자로부터의 특정 파일에 대한 데이터 접근 요청 메시지를 처리하도록 할 수 있다.
- [51] 각각의 서버는 자신이 저장하는 관리하는 해시키의 범위가 설정될 수 있으며, 이러한 해시키의 범위는 파일 시스템과 인메모리 캐시에 모두 설정될 수 있다.
- [52] 예를 들어 A 서버의 파일 시스템(202)은 56~5의 해시키 범위에 해당하는 해시키를 저장하도록 해시키 범위가 설정될 수 있으며, 인메모리 캐시(200)는 57~5의 해시키 범위를 가지도록 설정될 수 있다. 또한, B 서버의 파일 시스템(206)은 5~15의 해시키 범위를 가지며, 인메모리 캐시(204)는 5~11의 해시키 범위를 가지도록 설정될 수 있다. 또한, C 서버의 파일 시스템(210)은

15~26의 해시키 범위를 가지며, 인메모리 캐시(208)는 11~18의 해시키 범위를 가지도록 설정될 수 있다. 또한, D 서버의 파일 시스템(214)은 26~39의 해시키 범위를 가지며, 인메모리 캐시(212)는 18~39의 해시키 범위를 가지도록 설정될 수 있다. 또한, E 서버의 파일 시스템(218)은 39~47의 해시키 범위를 가지며, 인메모리 캐시(216)는 39~48의 해시키 범위를 가지도록 설정될 수 있다. 또한, F 서버의 파일 시스템(222)은 47~56의 해시키 범위를 가지며, 인메모리 캐시(220)는 48~57의 해시키 범위를 가지도록 설정될 수 있다.

- [53] 이때, 본 발명에서는 파일 시스템과 달리 인메모리 캐시의 해시키 범위 설정에 있어서 사용자로부터의 각 서버에 대한 데이터 접근(data access) 요청의 회수를 기반으로 해시키 범위를 변경시킬 수 있도록 한다. 예를 들어 B 서버의 경우 파일 시스템(206)의 해시키 범위는 5~15인 반면에 인메모리 캐시(204)의 해시키 범위는 5~11로 변경하여 설정할 수 있다.
- [54] 이와 같이 인메모리 캐시의 해시키 범위를 변경 설정하는 이유는 파일 시스템에 존재하는 해시키는 다른 서버의 파일 시스템으로 이동하는 경우 오버헤드(over head)가 크게 되지만, 인메모리 캐시에 존재하는 해시키는 다른 서버의 인메모리 캐시로 이동하는 경우 상대적으로 오버헤드가 크지 않기 때문이며, 인메모리 캐시에서 데이터를 못 찾는 경우 파일 시스템에서 해당 데이터를 찾으려면 되므로 크게 문제가 되지 않기 때문이다.
- [55] 따라서, 본 발명에서는 인메모리 캐시에 할당되는 해시키 범위를 사용자로부터의 데이터 접근 요청 회수를 기반으로 변경 설정시킴으로 데이터 검색 효율을 높일 수 있도록 한다.
- [56] 이때, 위와 같이 인메모리 캐시에 할당되는 해시키 범위를 변경 설정에 대해 좀더 자세히 설명하기로 한다.
- [57] 먼저, 각각의 서버에서 관리되고 있는 데이터에 대해 최근에 n개의 맵리듀스 테스트(mareduce task)들이 해시키에 접근한 분포가 도 3에서 보여지는 히스토그램(histogram)과 같다면, 도 3에서와 같은 히스토그램을 올드 히스토리(old history)로 관리하고, 그 다음 n개의 맵리듀스 테스트들이 들어왔을 때 해시키에 접근한 분포에 대해 도 4에서와 같은 히스토그램을 산출한 후, 무빙 에버리지(moving average) 공식을 이용하여 합친다.
- [58] 도 5는 해시키에 접근한 분포에 따른 도 3의 기존 히스토그램과 도 4의 가장 최근의 히스토그램을 무빙 에버리지 공식을 이용하여 합친 그래프를 도시한 것이며, 이러한 도 5에 도시된 그래프는 해시키 범위에서 데이터 접근이 발생하는 확률분포를 나타낼 수 있다.
- [59] 이어, 다시 위 도 5에서와 같은 확률분포를 이용하여 도 6에서와 같은 누적 확률분포에 대한 그래프를 산출할 수 있다.
- [60] 이때, 위 도 6에서와 같이 누적 확률 분포상에서 각 서버의 구간들이 같은 확률분포를 갖도록 표시할 수 있고, 이렇게 표시하는 경우 상대적으로 많은 접근(access)이 발생하는 데이터를 가지고 있는 서버(server 2, server4)의 해시키

범위는 줄어들고, 상대적으로 적게 접근되는 데이터를 가지고 있는 서버(server 1, server 3, server 5)의 해시키 범위는 넓어질 수 있다. 이에 따라 위와 같은 해시키 범위를 기반으로 각 서버의 인메모리 캐시상 할당되는 해시키 범위를 각 서버의 데이터 접근에 따른 해시키 범위로 변경 설정할 수 있는 것이다.

- [61] 위와 같이 해시키 범위를 변경하는 경우 원래 파일 시스템에 할당된 해시키 범위와 다르게 서버별로 데이터 접근이 발생하는 확률 분포에 따라서 인메모리 캐시의 해시키 범위를 동적으로 변화시킬 수 있고, 이렇게 함으로써 서버별 로드 밸런스(load balance)를 맞출 수 있어 파일 시스템의 성능을 높일 수 있다.
- [62] 다시 도 2를 참조하면, 예를 들어 D 서버의 경우 D 서버의 인메모리 캐시(212)는 상대적으로 많이 데이터 접근이 이루어지지 않으므로 D 서버의 인메모리 캐시(212)에는 해시키 범위를 넓게 설정하여 많은 해시키를 관리하도록 할 수 있고, B 서버의 경우 B 서버의 인메모리 캐시(204)는 상대적으로 많이 데이터 접근이 이루어지고 있으므로 B 서버의 인메모리 캐시(204)에는 해시키 범위를 좁게 설정하여 파일 시스템의 해시키 범위보다 적은 해시키를 관리하도록 할 수 있다.
- [63] 즉, 종래의 하둡 등과 같은 파일 시스템에서는 예를 들어 해시키 5에서 15 사이에 들어오는 데이터 접근 요청에 대해 모두 다 B 서버에서 수행하도록 되어 있어서 5부터 15까지 데이터 접근 요청이 굉장히 많이 발생하는 경우 B 서버만 바쁘게 되고 나머지 서버는 여유 자원이 있음에도 불구하고 활용되지 않게 되어 성능이 나빠지게 된다.
- [64] 반면에 본 발명에서는 도 2에서 보여지는 바와 같이 상대적으로 데이터 접근 요청이 많은 B 서버에 대해서는 B 서버의 인메모리 캐시(204)에 할당되는 해시키 범위를 줄여줌으로써 B 서버가 담당하는 로드를 일정 부분 줄여주고, 나머지 부분은 C 서버에서 동작이 이루어지도록 함으로써 서버간 로드 밸런스가 맞춰져서 성능을 높일 수 있게 되는 것이다.
- [65] 도 7은 본 발명의 실시예에 따른 코드 분산 해시테이블 기반의 맵리듀스 시스템에서 맵리듀스 태스크 수행 동작 개념을 도시한 것이다.
- [66] 위 도 7을 참조하면, 코드 분산 해시테이블 기반의 맵리듀스 시스템은 작업 스케줄러(job scheduler)(702)와 리소스 매니저(resource manager)(704)와 이중계층의 링구조로 연결되는 다수의 서버의 파일 시스템(202, 206, 210, 214, 218, 222)과 인메모리 캐시(in memory cache)(200, 204, 208, 212, 216, 220) 등을 포함할 수 있다.
- [67] 먼저, 작업 스케줄러(702)는 코드 분산해시테이블 기반의 파일 시스템 계층과 인메모리 캐시 계층의 이중계층의 링구조로 데이터를 관리하며, 맵리듀스 태스크들을 각 서버에 할당하는 역할을 수행한다. 리소스 매니저(704)는 새로운 서버의 추가, 제거, 복구 등을 관리하며, 파일의 업로드도 관리한다.
- [68] 작업 스케줄러(702)에서 맵리듀스 태스크를 각 서버에 할당하면, 각 서버는 분산된 방법으로 분산 해시테이블 파일 시스템의 데이터 블록에 접근한다. 이때,

중앙에서 동작하는 작업 스케줄러(702)와 리소스 매니저(704)는 시스템의 확장성을 위하여 최소한의 기능만 하도록 구현될 수 있다.

- [69] 또한, 작업 스케줄러(702)에서는 해시함수에 의해 각 데이터 블록들의 해시키를 결정하며, 이 해시키를 통해 각 데이터 블록을 어느 서버에 저장할지 결정한다.
- [70] 또한, 각 서버는 최대 m 개의 서버를 분산해시테이블에 저장하여 관리할 수 있으며, m 의 값은 시스템의 관리자가 선택할 수 있고, 이때 m 의 값은 S 를 전체 서버의 개수라고 했을 때, $2^m - 1 > S$ 를 충족해야한다.
- [71] 이때, 클러스터의 전체 서버 개수가 천 개 이상이 되지 않는다면, m 을 서버의 개수와 같게 하여 한 번 만에 데이터에 접근할 수 있게 한다. 또한, 서버의 추가, 제거가 많은 동적인 종래 peer-to-peer 시스템과 달리 본 발명의 클러스터 구성은 상대적으로 정적이기 때문에 많은 개수의 서버를 해시테이블에 저장하는 것이 확장성에 크게 지장을 주지 않으며, 데이터 접근 성능을 높이는 효과가 있다.
- [72] 이러한 모든 서버의 정보를 가지고 있는 분산해시테이블을 통해 각 파일의 키 범위를 고려하여 파일에 바로 접근할 수 있으며 중앙제어식인 하둡 등의 종래 시스템과 비교하여 확장성이 우수하다. 특히 각 서버에 저장되는 해시테이블은 모든 파일들의 메타데이터를 저장하는 것이 아니라 서버들이 담당하는 해시키 범위만을 저장하도록 함으로써 해시테이블이 차지하는 용량은 매우 작고 오버헤드가 거의 없도록 할 수 있다.
- [73] 예를 들어 한 서버가 데이터 접근 요청을 받으면 데이터의 해시키가 해당 서버의 해시키 범위 안에 있는지 체크한 후, 범위 안에 있으면 해당 서버의 파일 디렉토리에서 데이터 접근을 제공하며, 다른 서버의 범위 안에 있으면 요청을 해당 서버로 전달하도록 하는 것이다.
- [74] 또한, 분산해시테이블은 다른 서버들의 정보를 서버의 추가, 제거 등이 일어날 때만 업데이트하며, 하트비트(heartbeat) 메시지를 주기적으로 주고 받으면서 서버의 오작동(failure)을 감지한다. 만약 한 서버가 오작동하게 되면 리소스 매니저(704)가 상실된 파일 블록들을 다른 서버의 복제된 블록들로부터 복구하도록 할 수 있다. 한편 이러한 파일 블록들의 복제는 서로 다른 k 개의 해시 함수를 통해 k 개만큼 복제되어 서로 다른 서버에 저장되며, k 값은 시스템 관리자에 의해 조절될 수 있다.
- [75] 또한, 인메모리 캐시(200, 204, 208, 212, 216, 220)의 캐시 계층은 dCache 와 iCache 의 두 개 파티션으로 나뉘어 관리될 수 있으며, dCache 는 입력 데이터 블록이 저장되는 파티션이고, iCache 는 중간 결과물을 재사용하는 캐시 영역으로 구분되는 파티션으로, 맵테스크의 결과인 중간계산결과 데이터가 저장될 수 있다. 즉, 본 발명의 코드 분산해시테이블 인메모리 캐시에서는 작업의 중간계산결과까지 저장하여 재사용할 수 있도록 하며, 또한 memcached 와 같이 다른 서버의 캐시에 저장되어 있는 데이터도 읽어올 수 있게 하여 재사용도를 더욱 높일 수 있다. 이때, 특히 코드 분산해시테이블의 해시키를

이용하여 다른 서버에 저장되어 있는 데이터를 찾기 때문에 효율적일 수 있다. 또한 입력 데이터 블록들이 고루 분산되지 않은 상황에서도 다른 서버의 캐시에 접근할 수 있도록 함으로써 로드 밸런싱의 불균형 문제도 해결될 수 있다.

- [76] 일반적으로 계산중심의 작업들은 같은 어플리케이션(application)이 같은 입력 데이터를 요청하는 경우가 많다. 데이터베이스의 질의(query)들이 같은 테이블을 여러 번 참조하는 것이 그 예에 해당한다. 몇몇 연구에서 맵리듀스 작업의 30% 이상이 반복적으로 수행되고 있음이 알려짐에 따라 중간계산결과를 캐시에 저장하는 본 발명의 인메모리 캐시를 통해 불필요한 중복 계산 과정을 대폭 줄여줄 수 있다.
- [77] 이때, iCache에서 중간계산결과를 검색하기 위하여, 본 발명에서는 어플리케이션 ID, 입력 데이터 블록, 어플리케이션 매개변수 등의 해당 중간계산결과를 구분할 수 있도록 메타데이터를 구성하여 관리할 수 있다. 따라서, 새로운 작업이 들어오면 이러한 메타데이터를 탐색하여 중간계산결과를 재사용할 수 있는지 여부를 판단하고, 재사용할 수 있는 메타데이터가 발견되면 맵 과정을 거치지 않고 저장된 중간계산결과를 리듀스 과정에서 바로 처리할 수 있도록 한다. 또한, iCache에서 중간계산결과를 재사용할 수 없는 경우, dCache에서 입력 데이터 블록의 사용 가능여부를 판단하여 사용 가능하면 바로 캐시에서 입력 데이터에 접근한다. dCache 에서도 입력 데이터 블록을 사용할 수 없는 경우, 분산해시테이블 파일시스템 계층을 통해 입력 데이터에 접근하도록 할 수 있다.
- [78] 또한, 전술한 바와 같이 파일시스템 계층과 다르게, 인메모리 캐시의 해시키 범위에 대해서는 중앙의 작업 스케줄러(702)가 관리하면서 해시키 영역을 각 서버마다 분할 후 할당한다. 이때 각 서버마다 할당하는 해시키 범위는 동적으로 조절되기 때문에 접근 요청이 많은 입력 데이터의 경우 다수의 서버에 저장되게 된다. 예를 들어 많은 작업들이 하나의 입력 데이터에 접근 요청을 하는 경우, 본 발명에서는 작업 스케줄러가 지역성 인식 페어(fair) 스케줄링을 통해서 해당 입력 데이터가 여러 서버로 복제되도록 함으로서 다수의 서버에서 입력 데이터를 처리할 수 있도록 한다. 따라서 연속된 작업들의 캐시 사용률을 극대화하면서 로드 밸런싱을 보장할 수 있게 된다.
- [79] 이하에서는 다시 위 도 7을 참조하여 코드 분산 해시테이블 기반의 맵리듀스 시스템에서 맵리듀스 태스크를 수행하는 동작에 대해 보다 상세히 설명하기로 한다.
- [80] 도 7을 참조하면, 먼저 작업 스케줄러(702)는 도 2에 도시된 바와 같이 각 서버의 인메모리 캐시(200, 204, 208, 212, 216, 220)의 해시키 범위를 할당하고, 서버별 파일 시스템(202, 206, 210, 214, 218, 222)의 해시키 범위의 정보를 관리하고 있다.
- [81] 이러한 상태에서 특정 데이터 접근 어플리케이션(application)(700)을 통해 특정 파일에 대한 사용자로부터의 데이터 접근 요청이 수신될 수 있다.

- [82] 이와 같이 데이터 접근 요청이 수신되는 경우 작업 스케줄러(702)는 다수의 서버 중 사용자에게 의해 데이터 접근 요청된 파일을 저장하고 있는 서버들로 맵리듀스 테스트를 할당하여 맵리듀스 테스트 수행된 결과값을 데이터 접근 요청에 대한 응답으로 출력할 수 있다.
- [83] 즉, 작업 스케줄러(702)는 먼저 사용자로부터 데이터 접근 요청된 파일의 이름을 가지고 해시 함수를 돌려서 요청된 데이터에 해당하는 해시키를 얻을 수 있다.
- [84] 이때 만일, 요청된 데이터의 해시키가 38번이라고 가정하는 경우 작업 스케줄러(702)는 파일 시스템에 대해 할당된 해시키 범위 정보를 참조하여 해시키 38번이 해시키 26번과 39번 사이를 저장하고 있는 D 서버에 저장되어 있는 것을 알게 되고, 해시키 38번에 해당하는 파일에 대한 정보를 얻기 위해 D 서버의 파일 시스템(214)에 접근한다(S1).
- [85] 여기서, 해시키 38번에 해당하는 파일은 하나의 파일이나 이 파일은 여러개로 쪼개져서 여러개의 서버에 분산되어 있을 수 있다. 그러나 해시키 38번에 해당하는 파일이 여러개로 쪼개져서 분산되어 있는 경우라 하더라도 D 서버는 해시키 38번에 해당하는 파일에 대한 모든 정보를 가지고 있게 된다.
- [86] 이때, 이러한 해시키 38번에 해당하는 파일이 예를 들어 두 개의 데이터 블록으로 나누어져 분산되어 있고 각 데이터 블록의 해시키가 5번과 56번인 경우 D 서버는 이러한 파일의 메타데이터 정보를 작업 스케줄러(702)에게 제공한다(S2).
- [87] 그러면, 작업 스케줄러(702)는 메타데이터 정보를 이용하여 해시키 38번에 해당하는 파일이 두 개의 데이터 블록으로 분리되어 다른 서버에 분산되어 있음을 알게 되고, 각각의 데이터 블록에 대한 해시키 5번과 해시키 56번에 해당되는 해시키 범위를 가지는 인메모리 캐시에 해당하는 서버를 확인한다.
- [88] 이때, 디스크 파일 시스템에 할당된 해시키 범위와 인메모리 캐시에 할당된 해시키 범위가 서로 다를 수 있으므로 작업 스케줄러(702)는 서버별 인메모리 캐시에 할당된 해시키 범위 정보 테이블(703)를 이용하여 해시키 5번과 해시키 56번에 해당하는 인메모리 캐시를 확인하게 된다.
- [89] 여기서, 다시 도 7을 참조하면, B 서버의 인메모리 캐시(204)가 해시키 5번을 가지고 있고, F 서버의 인메모리 캐시(220)가 해시키 56번을 가지고 있음을 알 수 있다.
- [90] 따라서, 작업 스케줄러(702)는 데이터 접근 요청된 파일이 두 개의 데이터 블록으로 쪼개져 서로 다른 서버에 분산 저장되어 알 수 있고, 두 개의 맵퍼가 각각 B 서버와 F 서버에서 실행되어야 하는 등과 같은 맵테스크 스케줄링(map task schedule)을 수행하게 된다(S3).
- [91] 이에 따라 맵리듀스를 수행하는 해당 서버는 작업 스케줄러(702)로부터 맵테스크 스케줄 정보를 수신하여 액세스 요청된 파일의 데이터 블록이 분산 저장되어 있는 B 서버와 F 서버에서 각각 맵함수를 실행시켜 키, 값 데이터를

생성하게 된다.

- [92] 한편, 작업 스케줄러(702)는 해시키 5번과 해시키 56번의 데이터가 B 서버의 인메모리 캐시(204)와 F 서버의 인메모리 캐시(220)에 존재하는 경우 해당 인메모리 캐시에서 바로 읽어오면 되나, 만약에 해당 인메모리 캐시에 해시키 5번과 해시키 56번의 데이터가 존재하지 않는 경우에는 파일 시스템에서 데이터를 읽어올 수 있다.
- [93] 이때, 도 7에서 보여지는 바와 같이 해시키 56번의 데이터는 인메모리 캐시(220)상에서는 F 서버이지만 파일 시스템(202)상에서는 A 서버의 파일 시스템에 저장되어 있으므로 캐시 미스(cache miss)가 발생할 수 있으며, 캐시 미스가 발생하는 경우 작업 스케줄러(702)는 A 서버의 파일 시스템(202)으로부터 해시키 56번의 데이터를 읽어오게 된다(S4).
- [94] 또한, 마찬가지로 해시키 5번의 데이터에 대해서도 B 서버의 인메모리 캐시(204)상에 존재하지 않아 캐시 미스가 발생하는 경우, 해시키 5번의 데이터를 저장하고 있는 B 서버의 파일 시스템(206)으로부터 해시키 5번의 데이터를 읽어오게 된다. 해시키 5번의 경우에는 서로 비슷하게 인메모리 캐시(204)와 파일 시스템(206)에 할당된 해시키 범위가 비슷하게 맞아서 동일한 서버의 파일 시스템에서 읽어오는 케이스가 될 수 있다.
- [95] 이어, 해당 서버에서 위와 같이 B 서버와 F 서버에 대해서 맵함수를 실행하고 나면, 그 중간계산결과가 나오게 되며, 이러한 중간계산결과는 다시 데이터 블록에 따라 서로 다른 해시키로 도출될 수 있다.
- [96] 따라서, 작업 스케줄러(702)는 맵테스크 과정에서 출력된 중간계산결과로서 출력된 해시키가 몇 번인지를 확인하고 해당하는 해시키 범위를 가지는 서버의 인메모리 캐시에 저장할 수 있다. 이때, 예를 들어 두 개의 출력값에 해당하는 해시키가 각각 E 서버의 인메모리 캐시(216)에서 관리하는 해시키 범위에 매핑(mapping)되고, C 서버의 인메모리 캐시(208)에서 관리하는 해시키 범위에 매핑이 된다고 하면, 작업 스케줄러(702)는 해당 결과값의 해시키를 각각 E 서버의 인메모리 캐시(216)와 C 서버의 인메모리 캐시(208)에 저장하게 된다(S5).
- [97] 이어, 작업 스케줄러(702)는 해당 서버로 맵테스크가 실행된 중간계산결과값이 E 서버와 C 서버상에 저장되어 있음을 알려 주어 리듀스 작업이 각각 E 서버와 C 서버에서 실행되어야 하는 등과 같은 리듀스 테스크 스케줄링(reduce task schedule)을 해주게 된다(S6).
- [98] 이에 따라, E 서버와 C 서버에서 리듀스 함수를 돌려 최종 결과물 즉 데이터 액세스 요청에 대한 출력 파일을 생성하고 이를 사용자에게 제공하게 된다.
- [99] 상기한 바와 같이, 본 발명에 따르면, 코드 분산 해쉬테이블 기반의 맵리듀스 시스템에서, 코드 분산 해쉬테이블 기반의 파일 시스템 계층과 인메모리 캐시 계층의 이중계층의 링구조로 데이터를 관리하고 사용자의 데이터 접근 요청 빈도에 따라 데이터 접근 요청의 확률분포를 예측한 후, 예측된 확률분포를

기반으로 인메모리 캐시 계층의 코드 분산해시테이블의 해시키 범위를 조정하고 테스크를 스케줄링함으로써 로드 밸런싱이 가능하며, 캐시 히트율을 높일 수 있다.

- [100] 한편 상술한 본 발명의 설명에서는 구체적인 실시예에 관해 설명하였으나, 여러 가지 변형이 본 발명의 범위에서 벗어나지 않고 실시될 수 있다. 따라서 발명의 범위는 설명된 실시 예에 의하여 정할 것이 아니고 특허청구범위에 의해 정하여져야 한다.

청구범위

- [청구항 1] 코드 분산 해시테이블을 기반으로 데이터를 저장하는 파일 시스템과 인메모리 캐시를 구비하는 다수의 서버와,
상기 파일 시스템과 인메모리 캐시에 저장된 데이터를 이중계층의 링구조로 관리하며, 외부로부터 특정 파일에 대한 데이터 접근 요청 수신 시 상기 다수의 서버 중 상기 데이터 접근 요청된 파일을 저장하고 있는 서버들로 맵리듀스 태스크를 할당하여 상기 맵리듀스 태스크 수행된 결과값을 상기 데이터 접근 요청에 대한 응답으로 출력하는 작업 스케줄러
를 포함하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.
- [청구항 2] 제 1 항에 있어서,
상기 작업 스케줄러는,
상기 데이터 접근 요청 수신 시, 상기 파일의 이름을 통해 해시키를 추출하고 각 서버의 인메모리 캐시에 할당된 해시키 범위를 확인하여 상기 파일을 저장하고 있는 서버를 검색하며, 상기 검색된 서버로부터 상기 파일에 대한 메타데이터를 수신하여 상기 파일이 저장된 서버들로 상기 맵리듀스 태스크를 할당하는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.
- [청구항 3] 제 2 항에 있어서,
상기 작업 스케줄러는,
상기 파일에 대한 데이터 블록 구조와 각 데이터 블록이 분산 저장된 서버의 정보를 상기 메타데이터로 수신하며, 상기 각 데이터 블록이 저장된 서버들로 상기 맵리듀스 태스크를 할당하는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.
- [청구항 4] 제 1 항에 있어서,
상기 인메모리 캐시는,
상기 코드 분산 해시테이블을 이용하여 데이터에 대응되는 해시키를 저장하고 있으며, 상기 해시키의 저장에 있어서 상기 인메모리 캐시에 할당된 해시키 범위에 해당하는 해시키와 상기 해당 해시키에 대응하는 데이터를 저장하고 있는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.
- [청구항 5] 제 4 항에 있어서,
상기 작업 스케줄러는,
각 서버의 상기 인메모리 캐시의 해시키 범위를 각 서버에 대한 데이터 접근 요청 빈도에 따라 서버별로 동적으로 변경 설정하는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.
- [청구항 6] 제 1 항에 있어서,

상기 작업 스케줄러는,
상기 파일의 각 데이터 블록에 대한 상기 맵리듀스 태스크 처리 과정에서 발생하는 중간계산결과를 상기 파일 시스템에 저장하는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.

[청구항 7] 제 6 항에 있어서,
상기 중간계산결과는,
상기 각 데이터 블록에 따라 서로 다른 해시키를 가지도록 생성되며, 각각 다른 서버로 분산되는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.

[청구항 8] 제 6 항에 있어서,
상기 중간계산결과는,
상기 인메모리 캐시의 중간 결과물 재사용 캐시 영역에 저장되는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.

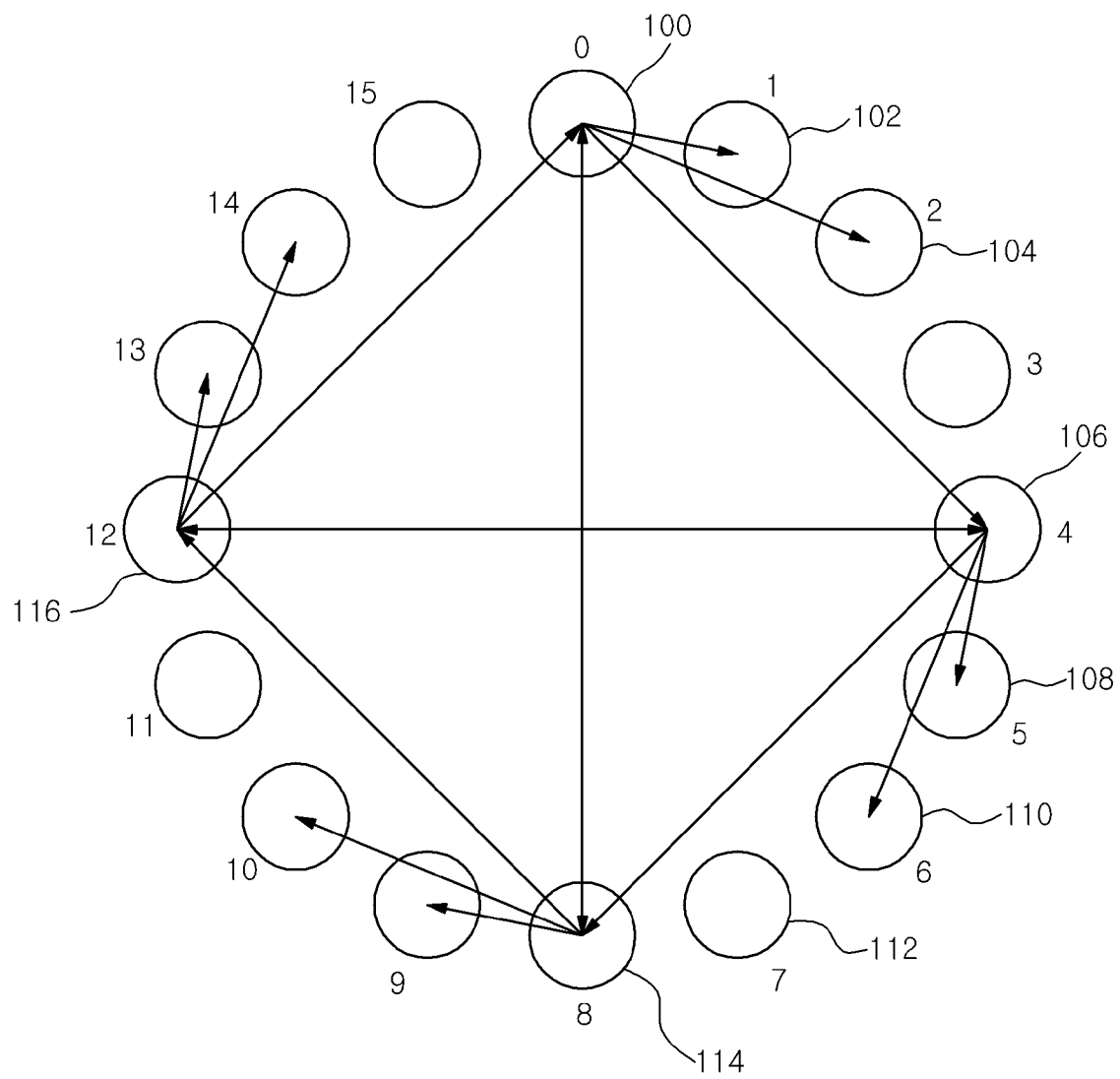
[청구항 9] 제 1 항에 있어서,
상기 시스템은,
상기 작업 스케줄러와 연동되며, 새로운 서버의 추가, 제거, 복구를 관리하거나 파일의 업로드를 관리하는 리소스 매니저를 더 포함하는 것을 특징으로 하는 코드 분산 해시테이블 기반의 맵리듀스 시스템.

[청구항 10] 파일 시스템과 인메모리 캐시를 구비하는 다수의 서버와 상기 서버로 맵리듀스 태스크를 할당하는 작업 스케줄러를 포함하는 코드 분산 해시테이블 기반의 맵리듀스 시스템에서 맵리듀스 태스크 수행방법에 있어서,
상기 작업 스케줄러에서 상기 파일 시스템과 인메모리 캐시에 저장된 데이터를 이중계층의 링구조로 관리하는 단계와,
외부로부터 특정 파일에 대한 데이터 접근 요청을 수신하는 단계와,
상기 파일에 대한 해시키를 추출하여 상기 파일을 저장하고 있는 파일 시스템의 서버를 검색하는 단계와,
상기 검색된 서버로부터 상기 파일에 대한 데이터 블록 구조와 각 데이터 블록의 분산 저장된 서버의 정보를 메타데이터로 수신하는 단계와,
상기 각 데이터 블록이 저장된 서버들로 맵리듀스 태스크를 할당하는 단계와,
상기 맵리듀스 태스크 수행된 결과값을 상기 작업 요청에 대한 응답으로 출력하는 단계
를 포함하는 방법.

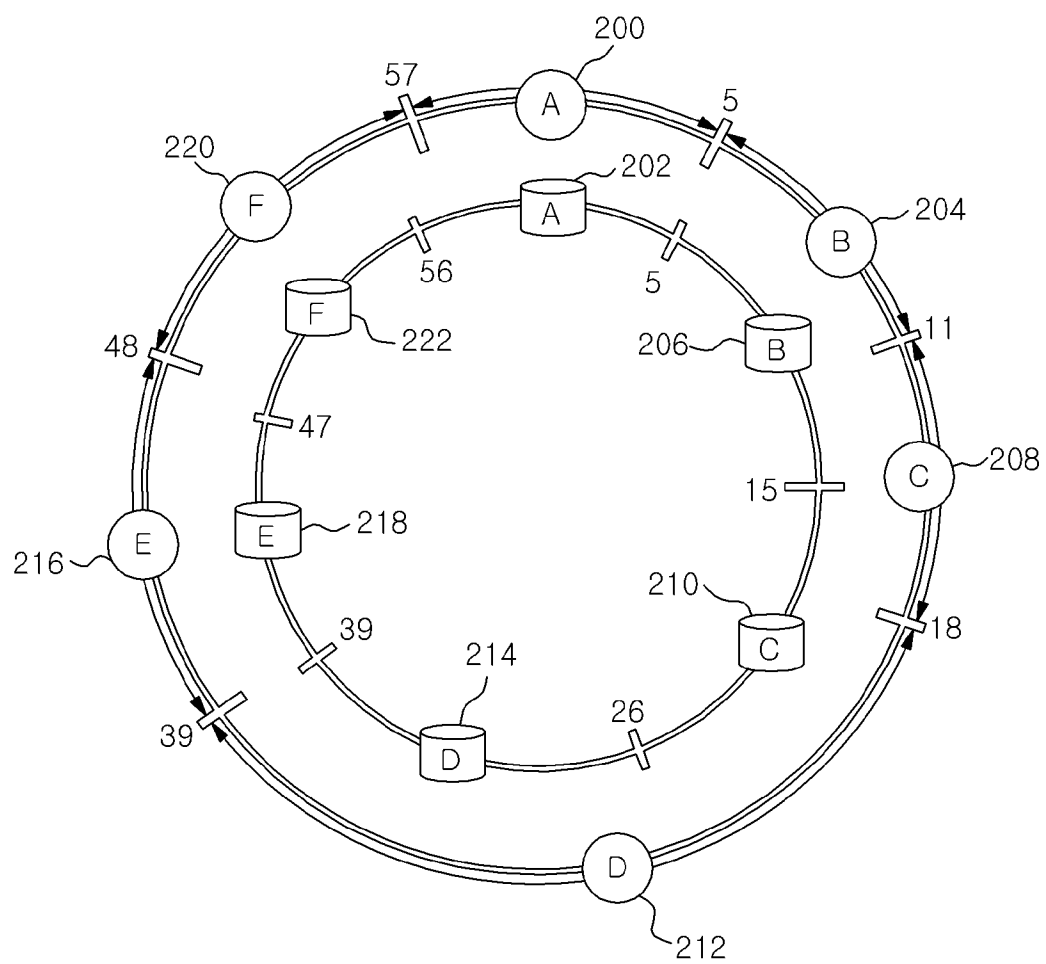
[청구항 11] 제 10 항에 있어서,
상기 파일 시스템과 인메모리 캐시는,
코드 분산 해시테이블을 기반으로 상기 데이터를 저장하는 것을 특징으로 하는 방법.

- [청구항 12] 제 10 항에 있어서,
상기 인메모리 캐시는,
상기 코드 분산 해시데이터블을 이용하여 데이터에 대응되는 해시키를
저장하고 있으며, 기설정된 해시키 범위를 할당받아, 상기 해시키 범위에
해당하는 해시키와 상기 해당 해시키에 대응하는 데이터를 저장하고
있는 것을 특징으로 하는 방법.
- [청구항 13] 제 12 항에 있어서,
상기 해시키 범위는,
각 서버에 대한 데이터 접근 요청 빈도에 따라 서버별로 동적으로 변경
설정되는 것을 특징으로 하는 방법.
- [청구항 14] 제 10 항에 있어서,
상기 맵리듀스 태스크는,
상기 각 데이터 블록이 저장된 서버들에서 처리되며, 상기 맵리듀스
태스크의 처리 과정에서 발생하는 중간계산결과가 상기 파일 시스템에
저장되는 것을 특징으로 하는 방법.
- [청구항 15] 제 14 항에 있어서,
상기 중간계산결과는,
상기 각 데이터 블록에 따라 서로 다른 해시키를 가지도록 생성되며, 각각
다른 서버로 분산되는 것을 특징으로 하는 방법.
- [청구항 16] 제 14 항에 있어서,
상기 중간계산결과는,
상기 인메모리 캐시의 중간 결과물 재사용 캐시 영역에 저장되는 것을
특징으로 하는 방법.

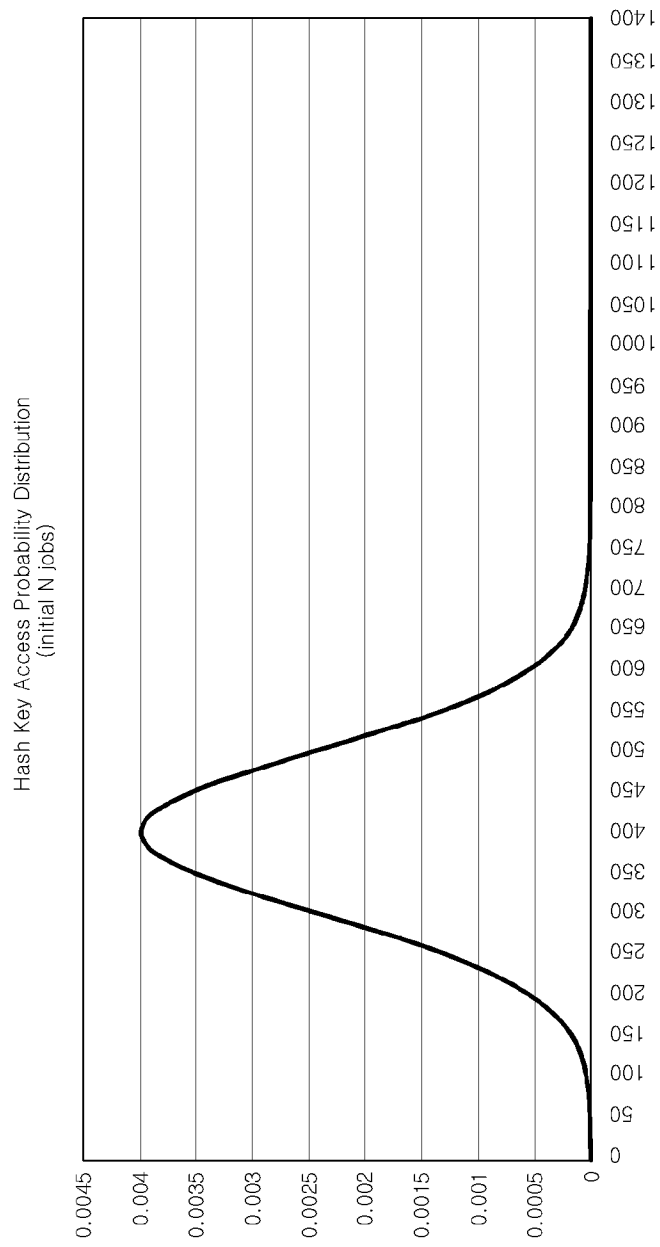
[도1]



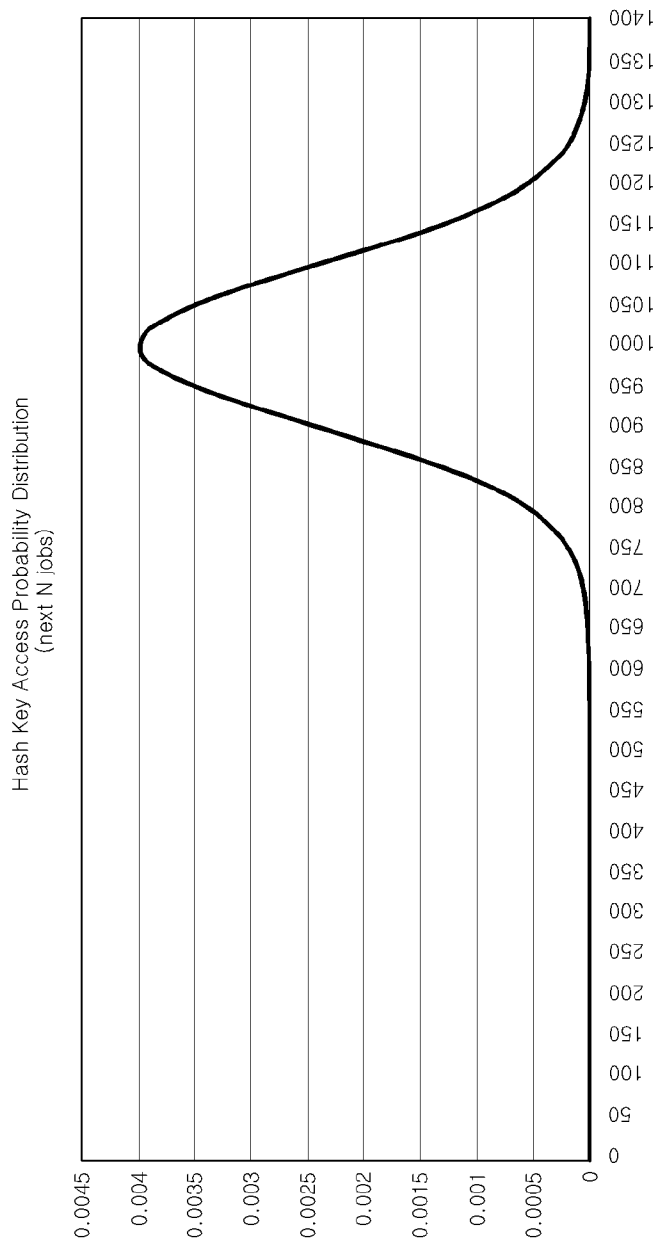
[도2]



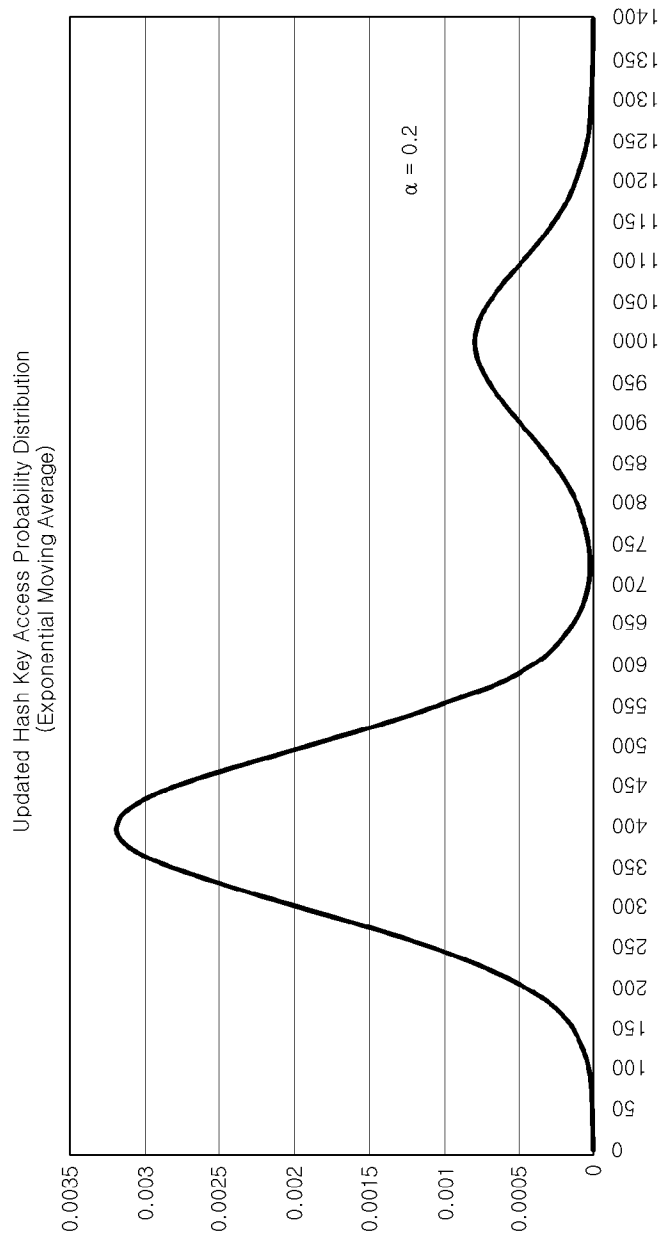
[도3]



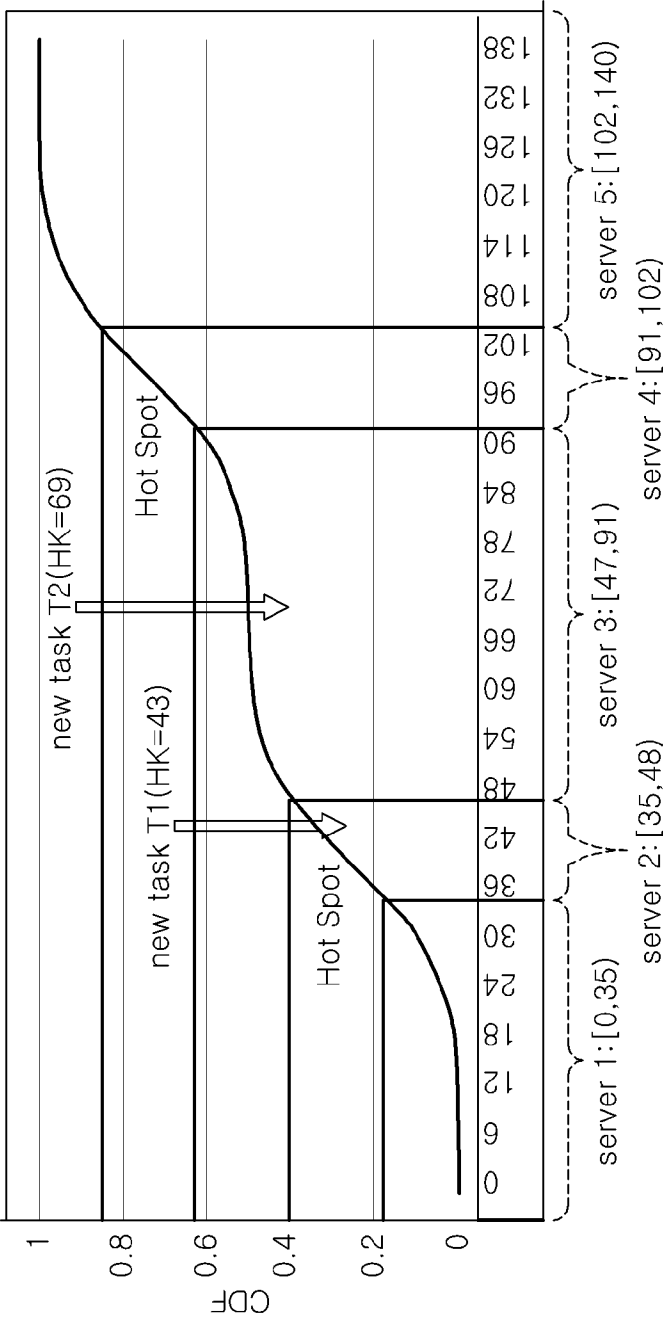
[도4]



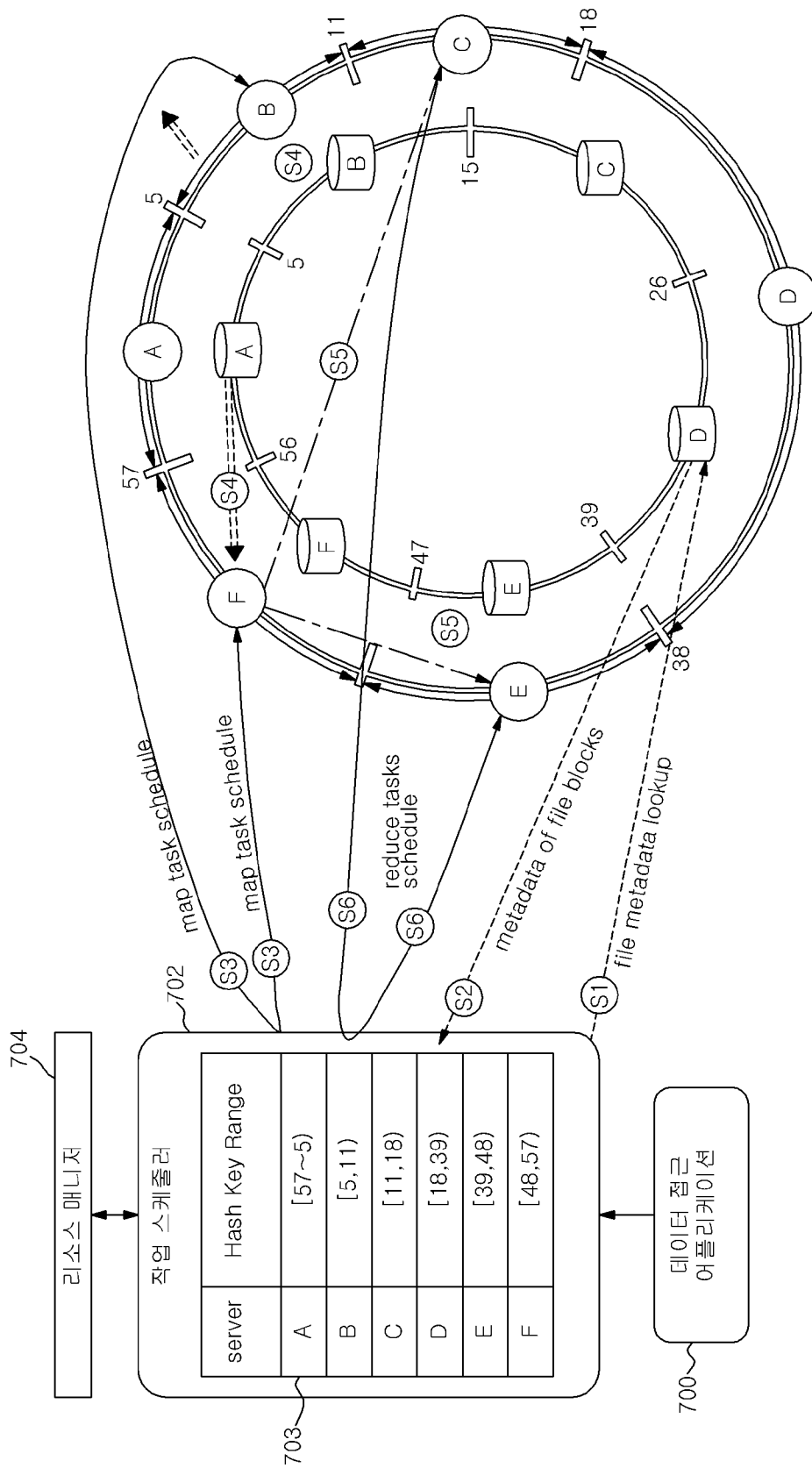
[도5]



[도6]



[도7]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/KR2015/005851**A. CLASSIFICATION OF SUBJECT MATTER****G06F 17/30(2006.01)i, G06F 9/48(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 17/30; G06F 15/16; H04L 29/08; G06F 21/78; G06F 7/00; H04L 29/06; G06F 9/50; G06F 9/48

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean Utility models and applications for Utility models: IPC as above

Japanese Utility models and applications for Utility models: IPC as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS (KIPO internal) & Keywords: chord distributed hash table, file system, in memory cache, server, double-layered ring structure, mapreduce, job scheduler.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 2634997 A1 (TELEFONAKTIEBOLAGET L M ERICSSON AB. (PUBL)) 04 September 2013 See paragraphs [0032] and [0043]; and figures 2-3.	1-16
A	KR 10-2014-0096936 A (SOMANSA CO., LTD.) 06 August 2014 See paragraphs [0049], [0051], [0055] and [0061]-[0064]; and figure 3.	1-16
A	US 2012-0278323 A1 (CHATTOPADHYAY, Biswapesh et al.) 01 November 2012 See paragraphs [0012]-[0013] and figures 1-2.	1-16
A	KR 10-2014-0119090 A (MICROSOFT CORP.) 08 October 2014 See paragraphs [0089]-[0090] and figure 1.	1-16
A	US 2008-0120314 A1 (YANG, Hung - Chih et al.) 22 May 2008 See paragraphs [0014]-[0016] and figure 2.	1-16



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

29 FEBRUARY 2016 (29.02.2016)

Date of mailing of the international search report

29 FEBRUARY 2016 (29.02.2016)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 189 Seonsa-ro, Daejeon 302-701,
Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/KR2015/005851

Patent document cited in search report	Publication date	Patent family member	Publication date
EP 2634997 A1	04/09/2013	CA 2724406 A1	26/11/2009
		CN 102037704 A	27/04/2011
		CN 102037704 B	27/08/2014
		EP 2289217 A1	02/03/2011
		JP 2011-521573 A	21/07/2011
		JP 5684110 B2	11/03/2015
		US 2011-110269 A1	12/05/2011
		US 8526331 B2	03/09/2013
		WO 2009-141013 A1	26/11/2009
KR 10-2014-0096936 A	06/08/2014	KR 10-1435789 B1	29/08/2014
US 2012-0278323 A1	01/11/2012	AU 2012-250238 A1	14/11/2013
		CN 103620601 A	05/03/2014
		EP 2702510 A1	05/03/2014
		US 8924426 B2	30/12/2014
		WO 2012-148616 A1	01/11/2012
KR 10-2014-0119090 A	08/10/2014	CN 104081353 A	01/10/2014
		EP 2810163 A1	10/12/2014
		JP 2015-506523 A	02/03/2015
		US 2013-0204991 A1	08/08/2013
		US 8868711 B2	21/10/2014
		WO 2013-116664 A1	08/08/2013
US 2008-0120314 A1	22/05/2008	US 7523123 B2	21/04/2009

A. 발명이 속하는 기술분류(국제특허분류(IPC))

G06F 17/30(2006.01)i, G06F 9/48(2006.01)i

B. 조사된 분야

조사된 최소문헌(국제특허분류를 기재)

G06F 17/30; G06F 15/16; H04L 29/08; G06F 21/78; G06F 7/00; H04L 29/06; G06F 9/50; G06F 9/48

조사된 기술분야에 속하는 최소문헌 이외의 문헌

한국등록실용신안공보 및 한국공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

일본등록실용신안공보 및 일본공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

국제조사에 이용된 전산 데이터베이스(데이터베이스의 명칭 및 검색어(해당하는 경우))

eKOMPASS(특허청 내부 검색시스템) & 키워드: 코드 분산 헤시테이블, 파일 시스템, 인메모리 캐시, 서버, 이중계층의 링 구조, 맵리듀스, 작업 스케줄러.

C. 관련 문헌

카테고리*	인용문헌명 및 관련 구절(해당하는 경우)의 기재	관련 청구항
A	EP 2634997 A1 (TELEFONAKTIEBOLAGET L M ERICSSON AB (PUBL)) 2013.09.04 단락 [0032]와 [0043]; 및 도면 2-3 참조.	1-16
A	KR 10-2014-0096936 A ((주)소만사) 2014.08.06 단락 [0049], [0051], [0055]과 [0061]-[0064]; 및 도면 3 참조.	1-16
A	US 2012-0278323 A1 (BISWAPESH CHATTOPADHYAY 등) 2012.11.01 단락 [0012]-[0013] 및 도면 1-2 참조.	1-16
A	KR 10-2014-0119090 A (마이크로소프트 코퍼레이션) 2014.10.08 단락 [0089]-[0090] 및 도면 1 참조.	1-16
A	US 2008-0120314 A1 (HUNG-CHIH YANG 등) 2008.05.22 단락 [0014]-[0016] 및 도면 2 참조.	1-16

☐ 추가 문헌이 C(계속)에 기재되어 있습니다.

☒ 대응특허에 관한 별지를 참조하십시오.

* 인용된 문헌의 특별 카테고리:

“A” 특별히 관련이 없는 것으로 보이는 일반적인 기술수준을 정의한 문헌

“T” 국제출원일 또는 우선일 후에 공개된 문헌으로, 출원과 상충하지 않으며 발명의 기초가 되는 원리나 이론을 이해하기 위해 인용된 문헌

“E” 국제출원일보다 빠른 출원일 또는 우선일을 가지나 국제출원일 이후에 공개된 선출원 또는 특허 문헌

“X” 특별한 관련이 있는 문헌. 해당 문헌 하나만으로 청구된 발명의 신규성 또는 진보성이 없는 것으로 본다.

“L” 우선권 주장에 의문을 제기하는 문헌 또는 다른 인용문헌의 공개일 또는 다른 특별한 이유(이유를 명시)를 밝히기 위하여 인용된 문헌

“Y” 특별한 관련이 있는 문헌. 해당 문헌이 하나 이상의 다른 문헌과 조합하는 경우로 그 조합이 당업자에게 자명한 경우 청구된 발명은 진보성이 없는 것으로 본다.

“O” 구두 개시, 사용, 전시 또는 기타 수단을 언급하고 있는 문헌

“&” 동일한 대응특허문헌에 속하는 문헌

“P” 우선일 이후에 공개되었으나 국제출원일 이전에 공개된 문헌

국제조사의 실제 완료일

2016년 02월 29일 (29.02.2016)

국제조사보고서 발송일

2016년 02월 29일 (29.02.2016)

ISA/KR의 명칭 및 우편주소

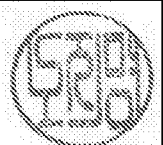
대한민국 특허청
(35208) 대전광역시 서구 청사로 189,
4동 (둔산동, 정부대전청사)

팩스 번호 +82-42-472-7140

심사관

노지명

전화번호 +82-42-481-8528



국제조사보고서에서 인용된 특허문헌	공개일	대응특허문헌	공개일
EP 2634997 A1	2013/09/04	CA 2724406 A1 CN 102037704 A CN 102037704 B EP 2289217 A1 JP 2011-521573 A JP 5684110 B2 US 2011-110269 A1 US 8526331 B2 WO 2009-141013 A1	2009/11/26 2011/04/27 2014/08/27 2011/03/02 2011/07/21 2015/03/11 2011/05/12 2013/09/03 2009/11/26
KR 10-2014-0096936 A	2014/08/06	KR 10-1435789 B1	2014/08/29
US 2012-0278323 A1	2012/11/01	AU 2012-250238 A1 CN 103620601 A EP 2702510 A1 US 8924426 B2 WO 2012-148616 A1	2013/11/14 2014/03/05 2014/03/05 2014/12/30 2012/11/01
KR 10-2014-0119090 A	2014/10/08	CN 104081353 A EP 2810163 A1 JP 2015-506523 A US 2013-0204991 A1 US 8868711 B2 WO 2013-116664 A1	2014/10/01 2014/12/10 2015/03/02 2013/08/08 2014/10/21 2013/08/08
US 2008-0120314 A1	2008/05/22	US 7523123 B2	2009/04/21