



(19) 대한민국특허청(KR)  
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0030177  
(43) 공개일자 2020년03월20일

(51) 국제특허분류(Int. Cl.)  
G06F 3/06 (2006.01)

(52) CPC특허분류  
G06F 3/0604 (2013.01)  
G06F 12/023 (2013.01)

(21) 출원번호 10-2018-0108659

(22) 출원일자 2018년09월12일  
심사청구일자 2018년09월12일

(71) 출원인  
고려대학교 산학협력단

서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)

(72) 발명자  
정연돈

서울특별시 중구 청계천로 400, 105-812 (황학동, 롯데캐슬베네치아)

서일현

서울특별시 종로구 종로 347, 지동 1805호 (송인동, 롯데캐슬천지인)

(74) 대리인

윤귀상

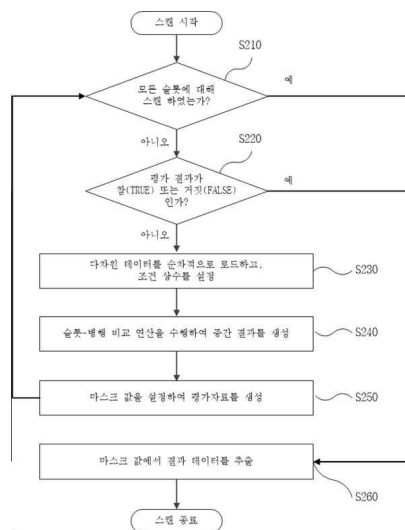
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 인-메모리 기반의 다차원 데이터 관리 장치 및 방법, 이를 수행하기 위한 기록 매체

### (57) 요약

메인 메모리에 다차원 데이터를 저장하고 검색하는 인-메모리 기반의 다차원 데이터 관리 장치에 있어서, 복수의 다차원 데이터가 저장되는 저장부, 저장 대상이 되는 상기 다차원 데이터를 분할하여 슬롯을 생성하고, 동일한 차원의 슬롯들은 기둥(column) 형태로 각각 다른 주소로 상기 저장부에 분배하며, 각 차원별로 동일한 순번의 슬롯들이 연속적으로 배치되는 행을 형성하도록 상기 저장부에 저장하는 저장 제어부, 상기 저장부의 다차원 데이터의 슬롯과 비교 대상 데이터의 슬롯을 비교하여 대소 여부에 대한 중간 결과를 생성하는 비교부 및 상기 비교부의 중간 결과와 마스크에 설정된 데이터에 대해 검색식에서 지정된 조건의 만족 여부를 평가하는 스캔부를 포함하는 인-메모리 기반의 다차원 데이터 관리 장치를 개시하고 있다.

대표도 - 도10



(52) CPC특허분류

*G06F 3/0613* (2013.01)

*G06F 3/0626* (2013.01)

*G06F 3/0631* (2013.01)

*G06F 3/0658* (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 2018019262

부처명 과학기술정보통신부

연구관리전문기관 한국연구재단

연구사업명 (이공)중견연구자-핵심(후속)

연구과제명 CCTV 빅데이터 기반 지능형 범죄추적/예방 및 사회안전시스템

기 여 율 1/1

주관기관 고려대학교

연구기간 2018.03.01 ~ 2019.02.28

---

## 명세서

### 청구범위

#### 청구항 1

메인 메모리에 다차원 데이터를 저장하고 검색하는 인-메모리 기반의 다차원 데이터 관리 장치에 있어서,  
복수의 다차원 데이터가 저장되는 저장부;

저장 대상이 되는 상기 다차원 데이터를 분할하여 슬롯을 생성하고, 동일한 차원의 슬롯들을 기둥(column) 형태로 각각 다른 주소로 상기 저장부에 분배하여, 각 차원별로 동일한 순번의 슬롯들이 연속적으로 배치되는 बैं크를 형성하도록 상기 저장부에 저장하는 저장 제어부;

상기 저장부의 다차원 데이터의 슬롯과 비교 대상 데이터의 슬롯을 비교하여 대소 여부에 대한 중간 결과를 생성하는 비교부; 및

상기 비교부의 중간 결과와 마스크에 설정된 데이터에 대해 검색식에서 지정된 조건의 만족 여부를 평가하는 스캔부;를 포함하는 인-메모리 기반의 다차원 데이터 관리 장치.

#### 청구항 2

제 1 항에 있어서,

상기 저장부에 저장된 다차원 데이터와 상기 비교 대상 데이터를 비교하기 위해 레지스터에 임시로 저장하는 레지스터부;를 더 포함하는, 인-메모리 기반의 다차원 데이터 관리 장치.

#### 청구항 3

제 2 항에 있어서, 상기 저장 제어부는,

상기 बैं크들이 연속적으로 배치되는 워드(word)의 크기를 상기 레지스터부의 크기와 일치하도록 선택하는, 인-메모리 기반의 다차원 데이터 관리 장치.

#### 청구항 4

제 1 항에 있어서, 상기 저장 제어부는,

상기 बैं크의 크기를 명령어에서 지원하는 크기들 중에서 선택하는, 인-메모리 기반의 다차원 데이터 관리 장치.

#### 청구항 5

제 1 항에 있어서, 상기 비교부는,

상기 저장부에 저장된 다차원 데이터와 상기 비교 데이터에 대해 동일한 위치에 있는 슬롯의 최우선 비트(MSB, Most Significant Bit)에 대한 비트 연산 결과와 상기 슬롯의 최우선 비트(MSB)가 아닌 비트에 대해 감산 연산 결과에 대해 비트 연산을 수행하여 대소를 비교하여 중간 결과를 생성하는, 인-메모리 기반의 다차원 데이터 관리 장치.

#### 청구항 6

제 1 항에 있어서, 상기 스캔부는,

상위 순번 슬롯에서 하위 순번 슬롯의 순서로 검색식에서 지정된 조건을 순차적으로 평가하는, 인-메모리 기반의 다차원 데이터 관리 장치.

## 청구항 7

제 6 항에 있어서, 상기 스캔부는,

상위 순번 슬롯에 대해 검색식에서 지정된 조건의 만족 여부가 참(TRUE) 또는 거짓(FALSE)으로 평가되면, 하위 순번 슬롯에 대해서는 조건의 만족 여부의 평가를 더 이상 수행하지 않고 종료하는, 인-메모리 기반의 다차원 데이터 관리 장치.

## 청구항 8

제 1 항에 있어서, 상기 스캔부는,

상기 저장부에 저장된 각각의 다차원 데이터들은 단일 명령어 복수 데이터(SIMD, Single Instruction Multiple Data)기술을 이용하여 상기 비교부의 차원 데이터와 비교 대상 데이터의 대소의 비교 및 상기 스캔부의 검색식에서 지정된 조건의 만족 여부 평가를 병렬적으로 수행하는, 인-메모리 기반의 다차원 데이터 관리 장치.

## 청구항 9

컴퓨터 프로세서에 의해 수행되는 메인 메모리에 다차원 데이터를 저장하고 검색하는 인-메모리 기반의 다차원 데이터 관리 방법에 있어서,

저장 대상이 되는 상기 다차원 데이터를 분할하여 슬롯을 생성하고, 동일한 차원의 슬롯들을 기둥(column) 형태로 각각 다른 주소로 분배하여, 각 차원별로 동일한 순번의 슬롯들이 연속적으로 배치되는 बैं크를 형성하도록 저장하는 단계;

상기 다차원 데이터의 슬롯과 비교 대상 데이터의 슬롯을 비교하여 대소 여부에 대한 중간 결과를 생성하는 단계; 및

상기 중간 결과와 마스크에 설정된 데이터에 대해 검색식에서 지정된 조건의 만족 여부를 평가하는 단계;를 포함하는 인-메모리 기반의 다차원 데이터 관리 방법.

## 청구항 10

제 9 항에 있어서,

상기 비교하는 단계를 수행하기 위해 다차원 데이터를 레지스터에 임시로 저장하는 단계;를 더 포함하는, 인-메모리 기반의 다차원 데이터 관리 방법.

## 청구항 11

제 9 항에 있어서, 상기 중간 결과를 생성하는 단계는,

상기 다차원 데이터와 상기 비교 데이터에 대해 동일한 위치에 있는 슬롯의 최우선 비트(MSB, Most Significant Bit)에 대한 비트 연산을 수행하는 단계;

상기 슬롯의 최우선 비트(MSB)가 아닌 비트에 대해 감산을 수행하는 단계; 및

상기 비트 연산 결과와 상기 감산 연산 수행 결과에 대해 비트 연산을 수행하는 단계;를 더 포함하는, 인-메모리 기반의 다차원 데이터 관리 방법.

## 청구항 12

제 9 항에 있어서, 상기 평가하는 단계는,

상위 순번 슬롯에서 하위 순번 슬롯의 순서로 검색식에서 지정된 조건을 순차적으로 평가하는, 인-메모리 기반의 다차원 데이터 관리 방법.

## 청구항 13

제 12 항에 있어서, 상기 평가하는 단계는,

상위 순번 슬롯에 대해 검색식에서 지정된 조건의 만족 여부가 참(TRUE) 또는 거짓(FALSE)으로 평가되면, 하위 순번 슬롯에 대해서는 조건의 만족 여부의 평가를 더 이상 수행하지 않고 종료하는, 인-메모리 기반의 다차원 데이터 관리 방법.

## 청구항 14

제 9 항에 있어서, 상기 평가하는 단계는,

상기 다차원 데이터는 단일 명령어 복수 데이터(SIMD, Single Instruction Multiple Data)기술을 이용하여 상기 차원 데이터와 상기 비교 대상 데이터의 대소의 비교 및 검색식에서 지정된 조건의 만족 여부 평가를 병렬적으로 수행하는, 인-메모리 기반의 다차원 데이터 관리 장치.

## 청구항 15

제 9 항 내지 제 14 항 중 어느 한 항에 따른 인-메모리 기반의 다차원 데이터 관리 방법을 실행하기 위한 컴퓨터 프로그램이 기록된 컴퓨터로 판독 가능한 저장 매체.

## 발명의 설명

### 기술 분야

[0001] 본 발명은 인-메모리 기반의 다차원 데이터 관리 장치 및 방법, 이를 수행하기 위한 기록 매체에 관한 것으로서, 더욱 상세하게는 메인 메모리상에 다차원 데이터를 배치하는 방법 및 배치된 다차원 데이터에 대한 스캔 방법에 관한 것이다.

### 배경 기술

[0003] 인-메모리 데이터베이스(In-memory Database)는 데이터 저장 장치의 메인 메모리에 데이터베이스를 설치하여 운영되는 방식의 데이터베이스 관리 시스템으로 디스크에 접근할 필요가 없기 때문에 데이터베이스가 디스크에 구축된 경우보다 빠른 검색 성능을 제공한다. 전원이 소실될 경우나 고의적인 상황 등에서 램(RAM)의 휘발성 문제가 되었으나, 비휘발성 램(RAM)기술의 도입으로 인-메모리 데이터베이스는 전력 손실에도 완전한 속도로 데이터를 유지할 수 있다.

[0004] 다차원 데이터(Multidimensional data)는 과학 영역에서 자연 현상이나 인공적인 시뮬레이션을 측정하여 수집된 데이터를 다차원 배열 구조로 모델링하여 다차원 조직 구조를 가지는 데이터를 의미한다.

[0005] SIMD(Single Instruction Multiple Data)는 병렬 프로세서의 한 종류로, 기존의 SISD(Single Instruction Multiple Data)는 한번에 하나의 명령어에 대해 하나의 데이터에 대해서만 연산이 가능했으나, SIMD는 하나의

명령어로 여러 개의 데이터를 동시에 연산하는 방식이다.

- [0006] 기존의 메인 메모리 데이터 레이아웃에서 검색은 한 번에 하나씩 각 차원에서 순차적으로 수행하였기 때문에 다차원 데이터 전체에 대해 검색을 수행해야만 하는 문제점이 있었다.

## 선행기술문헌

### 특허문헌

- [0008] (특허문헌 0001) 한국등록특허 제 10-1881676호  
(특허문헌 0002) 한국공개특허 제 10-2016-0014860호  
(특허문헌 0003) 미국등록특허 제 USRE38514호  
(특허문헌 0004) 한국등록특허 제 10-1331383호

## 발명의 내용

### 해결하려는 과제

- [0009] 이에, 본 발명의 기술적 과제는 이러한 점에서 착안된 것으로 본 발명의 목적은, 다차원 데이터의 빠른 스캔을 제공하기 위해 새로운 구조를 가진 인-메모리 기반의 다차원 데이터 관리 장치를 제공하는 것이다.
- [0010] 본 발명의 다른 목적은 다차원 데이터의 빠른 스캔을 제공하기 위해 새로운 구조를 가지는 인-메모리 기반의 다차원 데이터 관리 방법을 제공하는 것이다.
- [0011] 본 발명의 또 다른 목적은 다차원 데이터의 빠른 스캔의 제공하기 위해 새로운 구조를 가지는 인-메모리 기반의 다차원 데이터 관리 방법을 수행하기 위한 기록 매체를 제공하는 것이다.
- [0012] 본 발명의 기술적 과제는 이상에서 언급한 기술적 과제로 제한되지 않으며, 언급되지 않은 또 다른 기술적 과제들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

### 과제의 해결 수단

- [0014] 상기한 본 발명의 목적을 실현하기 위한 일 실시 예에 따른 메인 메모리에 다차원 데이터를 저장하고 검색하는 인-메모리 기반의 다차원 데이터 관리 장치는, 복수의 다차원 데이터가 저장되는 저장부; 저장 대상이 되는 상기 다차원 데이터를 분할하여 슬롯을 생성하고, 동일한 차원의 슬롯들을 기둥(column) 형태로 각각 다른 주소로 상기 저장부에 분배하여, 각 차원별로 동일한 순번의 슬롯들이 연속적으로 배치되는 बैं크를 형성하도록 상기 저장부에 저장하는 저장 제어부; 상기 저장부의 다차원 데이터의 슬롯과 비교 대상 데이터의 슬롯을 비교하여 대소 여부에 대한 중간 결과를 생성하는 비교부; 및 상기 비교부의 중간 결과와 마스크에 설정된 데이터에 대해 검색식에서 지정된 조건의 만족 여부를 평가하는 스캔부;를 포함한다.
- [0015] 본 발명의 실시 예에서, 상기 인-메모리 기반의 다차원 데이터 관리 장치는, 상기 저장부에 저장된 다차원 데이터와 상기 비교 대상 데이터를 비교하기 위해 레지스터에 임시로 저장하는 레지스터부를 더 포함할 수 있다.
- [0016] 본 발명의 실시 예에서, 상기 저장 제어부는, 상기 बैं크들이 연속적으로 배치되는 워드(word)의 크기는 상기 레지스터부의 크기와 일치하도록 선택할 수 있다.
- [0017] 본 발명의 실시 예에서, 상기 저장 제어부는, 상기 बैं크의 크기를 명령어에서 지원하는 크기들 중에서 선택할 수 있다.
- [0018] 본 발명의 실시 예에서, 상기 비교부는, 상기 저장부에 저장된 다차원 데이터와 상기 비교 데이터에 대해 동일한 위치에 있는 슬롯의 최우선 비트(MSB, Most Significant Bit)에 대한 비트 연산 결과와 상기 슬롯의 최우선 비트(MSB)가 아닌 비트에 대해 감산 연산 결과에 대해 비트 연산을 수행하여 대소를 비교하여 중간 결과를 생성

할 수 있다.

- [0019] 본 발명의 실시 예에서, 상기 스캔부는, 상위 순번 슬롯에서 하위 순번 슬롯의 순서로 검색식에서 지정된 조건을 순차적으로 평가할 수 있다.
- [0020] 본 발명의 실시 예에서, 상기 스캔부는, 상위 순번 슬롯에 대해 검색식에서 지정된 조건의 만족 여부가 참(TRUE) 또는 거짓(FALSE)으로 평가되면, 하위 순번 슬롯에 대해서는 조건의 만족 여부의 평가를 더 이상 수행하지 않고 스캔을 종료할 수 있다.
- [0021] 본 발명의 실시 예에서, 상기 저장부에 저장된 각각의 다차원 데이터들은 단일 명령어 복수 데이터(SIMD, Single Instruction Multiple Data)기술을 이용하여 상기 비교부의 차원 데이터와 비교 대상 데이터의 대소의 비교 및 상기 스캔부의 검색식에서 지정된 조건의 만족 여부 평가를 병렬적으로 수행할 수 있다.
- [0022] 상기한 본 발명의 목적을 실현하기 위한 다른 실시 예에 따른 컴퓨터 프로세서에 의해 수행되는 메인 메모리에 다차원 데이터를 저장하고 검색하는 인-메모리 기반의 다차원 데이터 관리 방법은, 저장 대상이 되는 상기 다차원 데이터를 분할하여 슬롯을 생성하고, 동일한 차원의 슬롯들을 기둥(column) 형태로 각각 다른 주소로 분배하여, 각 차원별로 동일한 순번의 슬롯들이 연속적으로 배치되는 행을 형성하도록 저장하는 단계; 각각의 순번에 대해 형성된 행들을 각각 다른 위치에 저장하는 단계; 상기 다차원 데이터의 슬롯과 비교 대상 데이터의 슬롯을 비교하여 대소 여부에 대한 중간 결과를 생성하는 단계; 및 상기 중간 결과와 마스크에 설정된 데이터에 대해 검색식에서 지정된 조건의 만족 여부를 평가하는 단계;를 포함한다.
- [0023] 본 발명의 실시 예에서, 인-메모리 기반의 다차원 데이터 관리 방법은 상기 비교하는 단계를 수행하기 위해 다차원 데이터를 레지스터에 임시로 저장하는 단계;를 더 포함할 수 있다.
- [0024] 본 발명의 실시 예에서, 상기 중간 결과를 생성하는 단계는, 상기 다차원 데이터와 상기 비교 데이터에 대해 동일한 위치에 있는 슬롯의 최우선 비트(MSB, Most Significant Bit)에 대한 비트 연산을 수행하는 단계; 상기 슬롯의 최우선 비트(MSB)가 아닌 비트에 대해 감산을 수행하는 단계; 및 상기 비트 연산 결과와 상기 감산 연산 수행 결과에 대해 비트 연산을 수행하는 단계;를 더 포함할 수 있다.
- [0025] 본 발명의 실시 예에서, 상기 평가하는 단계는, 상위 순번 슬롯에서 하위 순번 슬롯의 순서로 검색식에서 지정된 조건을 순차적으로 평가할 수 있다.
- [0026] 본 발명의 실시 예에서, 상기 평가하는 단계는, 상위 순번 슬롯에 대해 검색식에서 지정된 조건의 만족 여부가 참(TRUE) 또는 거짓(FALSE)으로 평가되면, 하위 순번 슬롯에 대해서는 조건의 만족 여부의 평가를 더 이상 수행하지 않고 종료할 수 있다.
- [0027] 본 발명의 실시 예에서, 상기 다차원 데이터는 단일 명령어 복수 데이터(SIMD, Single Instruction Multiple Data)기술을 이용하여 상기 차원 데이터와 상기 비교 대상 데이터의 대소의 비교 및 검색식에서 지정된 조건의 만족 여부 평가를 병렬적으로 수행할 수 있다.
- [0028] 상기한 본 발명의 목적을 실현하기 위한 다른 실시예에 따른 컴퓨터로 관독 가능한 저장 매체에는, 인-메모리 기반의 다차원 데이터 관리 방법을 실행하기 위한 컴퓨터 프로그램이 기록되어 있다.

## 발명의 효과

- [0030] 본 발명의 일 측면에 따른 인-메모리 기반의 다차원 데이터 관리 장치에 의해 제공되는 저장 방법은 기존의 다차원 데이터 레이아웃을 사용하는 경우보다 높은 밀도로 데이터를 저장할 수 있고, 별도의 구분자(identifier)를 필요로 하지 않기 때문에 요구되는 데이터 장치의 용량을 감소시킬 수 있고, 데이터베이스를 구축할 때 고가의 부속품인 메인 메모리의 용량을 줄일 수 있으므로 장비 단가 감소의 효과가 있다.
- [0031] 또한, 인-메모리 기반의 다차원 데이터 관리 장치에 의해 제공되는 슬롯 병행 비교 알고리즘(Slot-parallel comparison)은 단일 명령어 복수 데이터(SIMD, Single Instruction Multiple Data)를 이용하여 병행 연산이 가능하면서도 슬롯 단위로 순차적으로 연산하여 조건의 평가를 조기에 종료(early stopping)할 수 있어서 전체적인 연산 비용을 낮추는 효과가 있다.
- [0032] 본 발명에서 얻을 수 있는 효과는 이상에서 언급한 효과로 제한되지 않으며, 언급하지 않은 또 다른 효과들은 아래의 기재로부터 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이

다.

### 도면의 간단한 설명

- [0034] 도 1은 본 발명의 일 실시 예에 따른 인-메모리 기반의 다차원 데이터 관리 장치의 개략적인 블록도이다.
- 도 2는 본 발명에 따른 인-메모리 기반의 다차원 데이터의 레이아웃을 나타낸 구성도이다.
- 도 3은 본 발명에 따른 인-메모리 기반의 다차원 데이터의 저장 방법을 나타낸 흐름도이다.
- 도 4 내지 도 6은 각각 본 발명에 따른 슬롯-병행 비교의 첫번째 단계, 두번째 단계, 세번째 단계를 나타낸 메모리 순서도들이다.
- 도 7은 본 발명에 따른 스캔 방법의 다차원 데이터와 조건 상수를 나타낸 메모리 구성도이다.
- 도 8 및 도 9는 본 발명에 따른 스캔 방법의 첫번째 순환과 두번째 순환을 나타낸 메모리 순서도들이다.
- 도 10은 본 발명에 따른 인-메모리 기반의 다차원 데이터의 스캔 방법을 나타낸 흐름도이다.
- 도 11 내지 도 13은 본 발명에 의한 각 실험 예에서 연산 결과를 비교하는 그래프들이다.

### 발명을 실시하기 위한 구체적인 내용

- [0035] 후술하는 본 발명에 대한 상세한 설명은, 본 발명이 실시될 수 있는 특정 실시 예를 예시로서 도시하는 첨부 도면을 참조한다. 이들 실시 예는 당업자가 본 발명을 실시할 수 있기에 충분하도록 상세히 설명된다. 본 발명의 다양한 실시 예는 서로 다르지만 상호 배타적일 필요는 없음이 이해되어야 한다. 예를 들어, 여기에 기재되어 있는 특정 형상, 구조 및 특성은 일 실시 예와 관련하여 본 발명의 정신 및 범위를 벗어나지 않으면서 다른 실시 예로 구현될 수 있다. 또한, 각각의 개시된 실시 예 내의 개별 구성요소의 위치 또는 배치는 본 발명의 정신 및 범위를 벗어나지 않으면서 변경될 수 있음이 이해되어야 한다. 따라서, 후술하는 상세한 설명은 한정적인 의미로서 취하려는 것이 아니며, 본 발명의 범위는, 적절하게 설명된다면, 그 청구항들이 주장하는 것과 균등한 모든 범위와 더불어 첨부된 청구항에 의해서만 한정된다. 도면에서 유사한 참조부호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭한다.
- [0036] 이하, 도면들을 참조하여 본 발명의 바람직한 실시 예들을 보다 상세하게 설명하기로 한다.
- [0037] 도 1은 본 발명의 일 실시 예에 따른 인-메모리 기반의 다차원 데이터 관리 장치(100)의 개략적인 블록도이다.
- [0038] 도 1을 참조하면, 본 발명의 일 실시 예에 따른 데이터 관리 장치(100)는 저장부(110), 저장 제어부(120), 비교부(130) 및 스캔부(140)를 포함할 수 있고, 본 발명의 다른 실시 예에 따른 데이터 관리 장치(100)는 레지스터부(150)를 더 포함할 수 있다.
- [0039] 저장부(110), 저장 제어부(120), 비교부(130), 스캔부(140) 및 레지스터부(150)의 구성은 통합 모듈로 형성되거나, 하나 이상의 모듈로 이루어질 수 있다. 그러나, 이와 반대로 각 구성은 별도의 모듈로 이루어질 수도 있다.
- [0040] 저장부(110)는 관리의 대상이 되는 데이터 베이스가 저장되고 구축되는 데이터 스토리지이며, 일반적으로는 데이터 관리 장치(100)의 메인 메모리일 수 있다.
- [0041] 저장 제어부(120)는 데이터의 레이아웃을 분할하고 분배하여 저장부(110)에 저장되도록 하며, 상세하게는 다차원 데이터를 장치의 레지스터 크기, 다차원 데이터의 수, 각각의 다차원 데이터의 크기 등을 고려하여 데이터를 분할하여 슬롯을 생성하고, 각각의 차원의 동일한 순번을 가지는 슬롯들을 연속적으로 배치한 벡터를 생성하여 저장부(110)에 저장할 수 있다.
- [0042] 비교부(130)는 두 개의 데이터에 대해 비트 연산 및 감산 연산을 수행하여 대소를 비교한 중간 결과를 생성할 수 있다.
- [0043] 스캔부(140)는 상위 순번 슬롯에 대해 비교부(130)에서 생성한 중간 결과와 새로 설정한 마스크 값에 대해 비트 연산을 수행하여 조건을 만족하였는지 여부를 판단하고, 판단 결과가 참(true)이면 조건의 만족 여부 평가를 조기에 종료하고, 거짓(false)이면 다음 순번 슬롯에 대해 조건의 만족 여부를 평가를 수행한다.
- [0044] 레지스터부(150)는 상기 비교부(130)의 슬롯-병행 알고리즘 및 상기 스캔부(140)에 의한 스캔 알고리즘을 수행

하기 위해 다차원 데이터와 비교 대상 데이터를 임시로 저장한다.

[0045] 도 2는 본 발명에 따른 인-메모리 기반의 다차원 데이터의 레이아웃을 나타낸 구성도이다.

[0046] 도 2를 참조하면, 디멘션 슬라이스(DimensionSlice)의 구조를 알 수 있다. 레지스터(register)의 크기(R)은 simd\_register 크기(예: AVX2의 128, 256)를 의미하며, 워드(word)의 크기(W)는 메모리상 R의 크기를 가진 연속 비트로 정의한다. 뱅크(bank)의 크기(b)는 simd\_instructions에서 사용할 수 있는 뱅크 크기들(8, 16, 32, 64)중 에서 선택된다(M110, M150, M150). 슬롯(S, slot)은 b비트 뱅크를 n개의 부분으로 나눈 것을 의미하며(M111, M113, M115 ~ M155), n은 다차원 데이터 세트의 차원의 수를 의미한다.

[0047] 각 뱅크는 전역 슬롯 형식 S를 사용하여 논리적으로 n개의 슬롯으로 분할된다. 슬롯 형식  $S = (S_1, \dots, S_n)$ 은 b 비트를 통해 정의되며, 각 요소  $S_i$ 는 i번째 슬롯의 길이를 나타낸다. 예를 들어  $S = (S_1, \dots, S_n) = (|slot_1|, \dots, |slot_n|)$ ,  $\sum S_i = b$ 를 나타낸다. i번째 슬롯은 i번째 차원 값의 부분을 저장하는데 사용된다.

[0048] i번째 차원의 차원 값을 나타내는데 필요한 최소 비트 수는  $K_i$  이다. 각 차원 값  $d_i$ 의 비트 시퀀스는  $\lceil K_i / S_i \rceil$ 개의 부분으로 분할되며, 이 부분은 다른 워드의 동일한 뱅크 위치에 있는 i번째 슬롯으로 분배된다. 이와 같이  $R/b$  차원 코드의 일부분은 워드에 배열되고 뱅크의 각 슬롯에는 해당 차원 값의 일부분이 포함된다. 사용되지 않는 비트 공간은 0으로 채워진다.  $d_j^{[m]}$  ( $d_j$ 의 m번째 순서로 읽음)은 연결된 슬롯 값으로, 각 i번째 슬롯은  $d_j^i$ 의 m번째 최우선  $S_i$ 크기 부분을 저장한다.

[0049] 도 2는 simd\_register의 크기(R)가 256이고 simd\_instructions에서 사용할 수 있는 뱅크의 크기 8인 경우에 아래 표 1의 3차원인 샘플 데이터를 32개의 뱅크에 저장하는 예를 보여준다.

표 1

|     | $d_1$   |       |           | $d_2$   |       |           | ... | $d_3$      |       |           |
|-----|---------|-------|-----------|---------|-------|-----------|-----|------------|-------|-----------|
| [m] | 표현형     | 10진 값 | 2진 값      | 표현형     | 10진 값 | 2진 값      |     | 표현형        | 10진 값 | 2진 값      |
| 1st | $d_1^1$ | 202   | 011001010 | $d_2^1$ | 141   | 010001101 | ... | $d_{32}^1$ | 369   | 101110001 |
| 2nd | $d_1^2$ | 120   | 001111000 | $d_2^2$ | 379   | 101111011 | ... | $d_{32}^2$ | 66    | 001000010 |
| 3rd | $d_1^3$ | 27    | 011011    | $d_2^3$ | 5     | 000101    | ... | $d_{32}^3$ | 1     | 000001    |

[0050]

[0051] 이 예에서 simd\_register의 크기(R)는 256이므로 W는 연속적으로 구성된 256비트를 가지게 되며, M210, M230, M250은 각각의 워드  $W_1, W_2, W_3$ 를 나타낸다. simd\_instructions에서 사용할 수 있는 뱅크의 크기에 따라  $b = 8$ 이다.

[0052] 총 32개의 차원 값(예.  $d_1, d_2, \dots, d_{32}$ )중 하나의 차원 값(예.  $d_1$ )을 나타내는 데 필요한 최소한의 비트 수는 각각  $d_1^1$ 이 9 비트,  $d_1^2$ 가 9 비트 및  $d_1^3$ 가 6 비트이다. 각 차원 값은 3 부분으로 분할된다. 차원 코드의 부분은 3개의 워드의 동일한 주어진 뱅크 위치에 저장된다. 예를 들어,  $W_1$  워드(M210)의 첫번째 뱅크(M110)의 3개의 슬롯(M111, M113, M115)에는 첫 번째 차원 코드( $d_1$ )인  $d_1^1$ 의 첫번째 3-비트 011(M111),  $d_1^2$ 의 첫번째 3-비트 001(M113) 및  $d_1^3$ 의 첫번째 2-비트인 01부분(M115)이 저장된다. 마찬가지로  $W_2$  워드(M230)의 첫번째 뱅크(M110)에는 첫 번째 차원 코드( $d_1$ )인  $d_1^1$ 의 두번째 3-비트인 001(M111),  $d_1^2$ 의 두번째 3-비트인 111(M113) 및  $d_1^3$ 의 두번째 2-비트인 10(M115)이 저장된다.

[0053] 같은 방식으로  $d_2$ 는  $d_1$ 이 저장된 바로 다음 영역 ( $M200 \times M130$ )에 저장되고,  $d_{32}$ 는 마지막 영역( $M200 \times M150$ )에 저장된다.

[0054] 다차원 코드를 저장하는 데 필요한 워드의 수는  $I$ 로 표시되며, 아래 수학적 1과 같이 표현된다.

# 수학식 1

$$l = \max_{1 \leq i \leq n} \left( \left\lceil \frac{k^i}{S_i} \right\rceil \right)$$

[0055]

[0056] 최소한의  $l$  값은 스캔시 사용되지 않는 비트와 메모리 대역폭 소비를 최소화하는데 바람직하다.  $l$  을 최소화하기 위해서, 총합에 대해 각  $k^i$  값의 비율에 따라 뱅크를 슬롯으로 나눈다.

[0057] 도 3은 본 발명에 따른 인-메모리 기반의 다차원 데이터의 저장 방법을 나타낸 흐름도이다.

[0058] 도 3을 참조하면, 다차원 데이터가 입력이 되면 simd\_register의 크기(R)와 simd\_instructions의 크기(b)를 고려하여 필요한 워드의 수  $l$  을 계산한 후, 각 차원 데이터를 분할하여 슬롯을 생성한다(S110). 동일한 차원의 슬롯들은 기둥(column) 형태로 각각 다른 주소에 분배하고(s130), 각 차원별로 동일한 순번의 슬롯들을 연속적으로 배치한 뱅크를 형성하도록 저장한다(s150).

[0059] 상기 표 1과 도 2를 참조하여 예를 들면  $R = 256$ 과  $b = 8$ 이 주어지고 개별적인 다차원 데이터의 크기가  $24(=9+9+6)$ 이므로  $l$  의 값은  $3(=24/8)$ 이 된다. 각 차원 데이터는 3개의 슬롯( $S_1, S_2, S_3$ )으로 분할되고,  $d_1^1, d_1^2$  및  $d_1^3$  의  $S_1$ 은 하나의 뱅크를 형성하여 첫번째 워드(M210)의 첫번째 뱅크(M110)에 저장되고,  $d_1^1, d_1^2$  및  $d_1^3$  의  $S_1$ 은 다른 주소를 가지는 두번째 워드(M230)의 첫번째 뱅크(M110)에 저장된다. 이와 같은 과정을 순차적으로 수행하여 다차원 데이터 전체를 메인 메모리에 저장할 수 있게 된다.

[0060] 도 4 내지 도 6은 각각 본 발명에 따른 슬롯-병행 비교의 첫번째 단계, 두번째 단계, 세번째 단계를 나타낸 메모리 순서도들이다.

[0061] 도 4 내지 도 6을 참조하면, 두 뱅크 간의 슬롯-병행 비교를 나타낸 것이다. 다차원 데이터의 각 차원의 동일한 순번의 슬롯들이 모인 뱅크 단위로 슬롯-병행 비교하며, 비교의 목적은 슬롯으로 분할된 차원 데이터를 저장한 구조를 유지한 채 두 뱅크에서 동일한 위치에 있는 슬롯 쌍의 비교 결과를 유도하는 것이다. 이하에서는 3차원 데이터의 3비트, 3비트, 2비트 슬롯들로 형성된 뱅크들을 비교하는 것으로 0과 1의 메모리 안쪽의 숫자들은 이해를 돕기 위한 예시이다.

[0062] 도 4를 참조하면, 비교 대상 데이터가 있는 왼쪽 뱅크(M311)는 LEFT, 오른쪽 뱅크(M321)는 RIGHT라고 하고, 비트 연산 수행결과가 표시되는 왼쪽 뱅크(M315)는 Lm, 오른쪽 뱅크(M325)는 Rm이라고 한다. 슬롯-병행 비교의 첫번째 단계는 LEFT(M311)와 RIGHT(M321)에 있는 각 슬롯의 최우선비트(MSB)에 대해서만 비트 연산을 수행하여 각 슬롯 값의 크기를 비교하는 것이다. Lm(M315)은 RIGHT에 대해 NOT연산을 수행한 후 LEFT와 AND연산(M313)을 수행한 결과이고, Rm(M325)은 LEFT에 대해 NOT연산을 수행한 후 RIGHT와 AND연산(M313)을 수행한 결과이다. 의사 코드(pseudo code)로 표현하면 아래와 같다.

[0063]  $Lm(M315) \leftarrow \sim RIGHT(M321) \wedge LEFT(M311)$

[0064]  $Rm(M325) \leftarrow \sim LEFT(M311) \wedge RIGHT(M321)$

[0065] Rm(M325)의 각 슬롯의 최우선비트(MSB)는 0, 0, 1이므로 최우선비트(MSB)만 비교한 결과로는 RIGHT(M321)의 세번째 슬롯의 값(11)이 LEFT(M311)의 세번째 슬롯의 값(01)보다 크다는 것을 의미한다.

[0066] 도 5를 참조하면, 슬롯-병행 비교 알고리즘의 두번째 단계를 나타낸다. 비교 대상 데이터가 있는 왼쪽 뱅크(M331)는 도 4의 LEFT(M311)이고, 오른쪽 뱅크(M341)는 도 4의 RIGHT(M321)이다. 감산 연산 수행결과가 표시되는 왼쪽 뱅크(M337)는 Lrest, 오른쪽 뱅크(M347)는 Rrest이라고 한다. 슬롯-병행 비교의 두번째 단계는 LEFT(M311)와 RIGHT(M321)에 대해 감산 연산을 수행하여 각 슬롯의 최우선비트(MSB)를 제외한 나머지 비트의 크기를 비교하는 것이다. 각각의 슬롯은 특별한 경계없이 연속적으로 배치되어 있으므로 작은 값에서 큰 값을 감산할 때 발생할 수 있는 언더플로우(underflow)를 방지하기 위해 첫 번째 피연산자의 각 슬롯의 최우선 비트(MSB)를 1로 설정(slotMSB1)하고(S310, S320), 두번째 피연산자의 각 슬롯의 최우선 비트를 0으로 설정( $\sim slotMSB1$ )한다(S330, S340). 그 후 첫번째 피연산자(M333, M343)에서 두번째 피연산자(M335, M345)를 감산한다. 의사 코드(pseudo code)로 표현하면 다음과 같다.

- [0067]  $Lrest(M337) \leftarrow (slotMSB1 \vee LEFT)(M333) - (\sim slotMSB1 \wedge RIGHT)(M335)$
- [0068]  $Rrest(M347) \leftarrow (slotMSB1 \vee RIGHT)(M343) - (\sim slotMSB1 \wedge LEFT)(M345)$
- [0069] Rrest(M347)의 각 슬롯의 최우선비트(MSB)는 0, 1, 1이므로 최우선 비트(MSB)를 제외한 나머지 부분에 대해서는 RIGHT(M341)의 첫번째 슬롯의 값(010)이 LEFT(M331)의 첫번째 슬롯의 값(011)보다 작다는 것을 의미한다.
- [0070] 도 6을 참조하면, 상기 ANDNOT 연산 결과인 Lm(M351)과 Rm(M361), 상기 감산 연산 결과인 Lrest(M353)과 Rrest(M363), 그리고 최종 연산 수행 결과로 왼쪽 뱅크(M355)는 ge(크거나 같음, GREATER-THAN-OR-EQUAL-TO)이고, 오른쪽 뱅크(M365)는 le(작거나 같음, LESS-THAN-OR-EQUAL-TO)이다. 슬롯-병행 비교의 마지막 세번째 단계는 중간 결과(Lm(M351), Rm(M361), Lrest(M353), Rrest(M363))에 대해 비트연산(M370, M380)을 수행하여 ge와 le를 구하는 것이다
- [0071] 동일한 위치에서 LEFT와 RIGHT의 한 쌍의 슬롯을 고려할 때 Rm(M361)의 슬롯의 최우선비트(MSB)가 1로 설정되면 LEFT의 슬롯 값이 RIGHT의 슬롯 값보다 작다고 결정할 수 있고, Rrest(M363)의 슬롯의 최우선비트(MSB)가 1이면 LEFT의 슬롯 값이 RIGHT의 슬롯 값보다 작거나 동일한 것으로 결정할 수 있다. 이 사실을 이용하면 작거나 같음(le)은 Rm(M361)과 Rrest(M363)에 대해 OR연산을 수행한 후에 NOT연산을 수행한 Lm(M351)과 AND 연산을 수행함으로써 유도할 수 있다(M370). ge를 유도하는 것은 le를 유도하는 것과 대칭적이다. 의사 코드(pseudo code)로 표현하면 다음과 같다.
- [0072]  $le(M365) = \sim Lm(M351) \wedge (Rm(M361) \vee Rrest(M363))$
- [0073]  $ge(M355) = \sim Rm(M361) \wedge (Lm(M351) \vee Lrest(M353))$
- [0074] ge(M355)의 각 슬롯의 최우선비트(MSB)는 1, 1, 0이고, le(M365)의 각 슬롯의 최우선비트(MSB)는 0, 1, 1이다. 이는 LEFT의 첫번째 슬롯에 대해 ge(M355)는 1이고, le는 0이므로 LEFT의 첫번째 슬롯의 값(011)이 RIGHT의 첫번째 슬롯의 값(010)보다 크다는 것을 의미하고, LEFT의 두번째 슬롯에 대해 ge(M355)는 1이고, le(M365)도 1이므로 LEFT의 두번째 슬롯의 값(001)이 RIGHT의 두번째 슬롯의 값(001)과 같다는 것을 의미한다.
- [0075] 슬롯-병행 비교 알고리즘은 단일 연산을 나타낸 것으로 데이터의 비교 및 감산을 위해 비트 단위 연산만 있고, 데이터를 분할 이전 상태로 되돌리는 등의 연산이 없으므로 SIMD(Single Instruction Multiple Data)에서 연산이 가능하도록 변형할 수 있다.
- [0076] 도 7은 본 발명에 따른 스캔 방법의 다차원 데이터와 조건 상수를 나타낸 메모리 구성도이다.
- [0077] 도 7을 참조하면, 다차원 데이터(M410, data)와 검색 대상이 되는 조건 상수(M430, predicate constant)를 메모리 상에 나타낸 것이다. 차원 데이터의 뱅크의 크기(M420)와 조건 상수의 뱅크의 크기(M440)는 동일해야 하고, 각 차원의 슬롯의 위치도 동일 해야 한다. 다차원 데이터의 첫번째 차원의 첫번째 슬롯(M421×M451)과 조건 상수의 첫번째 차원의 첫번째 슬롯(M441×M451)을 비교하고 조건을 만족하였는지를 평가하게 된다.
- [0078] 이하에서는 다차원 데이터의 첫번째, 두번째, 세번째 차원은 각각  $d_1(M421)$ ,  $d_2(M423)$ ,  $d_3(M425)$ 라 하고, 조건 상수의 첫번째, 두번째, 세번째 차원을 각각  $pc_1(M441)$ ,  $pc_2(M443)$ ,  $pc_3(M445)$ 라고 할 때,  $(d_1 > pc_1) \wedge (d_2 > pc_2) \wedge (d_3 > pc_3)$ 의 조건을 만족하는지 확인하는 과정으로 0과 1의 메모리 안쪽의 숫자들은 이해를 돕기 위한 예시이다.
- [0079] 도 8 및 도 9는 본 발명에 따른 스캔 방법의 첫번째 순환과 두번째 순환을 나타낸 메모리 순서도들이다.
- [0080] 도 8 및 도 9를 참조하면, 두 뱅크 간의 스캔 절차를 나타낸 것이다.
- [0081] 우선 검사된 슬롯의 결과를 저장하기 위해 마스크 Mgt(M511)을 사용한다. 처음에는 Mgt(M511)의 각 슬롯의 최우선비트(MSB)가 0으로 설정된다. 초기화 후 다차원 코드의 첫번째 뱅크(M531, 도 7의 M420×M451)와 조건 상수의 첫번째 뱅크(M551, 도 7의 M440×M451)가 로드된다. 그 다음 슬롯-병행 비교를 수행하여 구한 ge와 le에 NOT연산을 추가로 수행(M513, M613)하여 lt(M533, M633)와 gt(M553, M653)를 구한다.
- [0082] Mgt(M511)와 lt(M533, M633)는 ANDNOT연산을 수행한다(M515, M615). 이는 이전 순환에서 조건을 이미 충족한 슬롯을 평가에서 제외하기 위한 것으로 슬롯의 최우선비트(MSB) 중 하나라도 1로 설정되면 특정 차원의 조건이 거짓(FALSE)으로 평가된 것이기 때문에 평가를 종료하고 거짓(FASLE)을 반환하게 된다.
- [0083] Mgt(M511)와 gt(M553, M653)는 OR연산(M517, M617)을 수행한다. 이는 이전 순환을 조건을 이미 충족한 슬롯까

지 포함하여 평가하기 위한 것으로 모든 슬롯의 최우선비트(MSB)가 1로 설정되면 모든 차원의 조건이 참(TRUE)으로 평가된 것이기 때문에 평가를 종료하고 참(TRUE)을 반환하게 된다.

[0084] 참(TRUE) 또는 거짓(FALSE)의 결과를 평가할 수 없다면 OR연산(M517, M617)의 결과를 Mgt(M511)에 반영하고, 해당 बैं크에 대한 순환을 종료하고, 다음 बैं크에 대한 순환을 진행한다.

[0085] LESS-THAN은 GREATER-THAN에 사용된 피연산자를 교환하여 구현할 수 있다. 부호 없는 정수(unsigned integer)인 것으로 차원 코드를 가정하면, 상수 값을 증가시키거나 감소시켜서 GREATER-THAN-OR-EQUAL-TO와 LESS-THAN-OR-EQUAL-TO를 평가할 수 있다. EQUALITY의 경우, 해당 슬롯 형식 상수와 동일한 1을 बैं크로 유지하고, 1로 채워지지 않은 बैं크가 없는 경우 순환을 정지하는 부분을 수정하여 구현할 수 있다. BETWEEN의 경우, 더 낮은 상수와 높은 상수에 대한 비교를 수행 한 후 비트 AND를 적용하고 마스크를 업데이트하여 두 결과에 모두 반영한다.

[0086] 도 8을 참조하면, 도 7의 다차원 데이터의 बैं크(M420×M451)과 조건 상수의 बैं크(M440×M451)가  $(d_1 > pc_1) \wedge (d_2 > pc_2) \wedge (d_3 > pc_3)$  조건을 만족하였는지를 각 차원 데이터의 첫번째 बैं크에 대해 평가하는 단계이다. ANDNOT연산(M515)의 결과(M535)에서 모든 슬롯의 최우선비트(MSB)가 모두 0이므로 거짓(FALSE)으로 평가할 수 없고, OR연산(M517)의 결과(M555)에서 모든 슬롯의 최우선 비트(MSB)에서 0이 있으므로 참(TRUE)으로도 평가할 수 없다. 참(TRUE) 또는 거짓(FALSE)으로 평가할 수 없으므로 OR연산(M517)의 결과(M555)를 Mgt(M511)에 반영하고 첫번째 순환을 종료한다.

[0087] 도 9을 참조하면, 도 7의 다차원 데이터의 बैं크(M420×M453)과 조건 상수의 बैं크(M440×M453)가  $(d_1 > pc_1) \wedge (d_2 > pc_2) \wedge (d_3 > pc_3)$  조건을 만족하였는지를 각 차원 데이터의 두번째 बैं크를 기준으로 평가하는 단계이다. ANDNOT연산(M615)의 결과(M635)에서 모든 슬롯의 최우선비트(MSB)가 모두 0이므로 거짓(FALSE)을 반환할 수 없고, OR연산(M617)의 결과(M655)에서 모든 슬롯의 최우선 비트(MSB)가 1이므로 참(TRUE)으로 평가된다. 정지 조건이 참(TRUE) 충족되었으므로 참(TRUE)을 반환하고, 스캔 절차를 종료한다.

[0088] 도 10은 본 발명에 따른 인-메모리 기반의 다차원 데이터의 스캔 방법을 나타낸 흐름도이다.

[0089] 도 10을 참조하면, 스캔 알고리즘의 개략적인 절차는 다음과 같다.

[0090] 우선, 모든 슬롯에 대해 스캔을 하였는지 확인하고, 모든 슬롯에 대해 스캔을 한 경우에는 순환을 완료하고 그렇지 않은 경우에는 평가 결과 판단을 진행한다(S210). 마스크 값을 참고하여 참(TRUE) 또는 거짓(FALSE)으로 평가되면 스캔을 정지하고, 그렇지 않으면 스캔 절차를 수행하는 것으로 결정한다(S220). 이 두 가지 종료 조건은 오버 헤드를 줄이기 위해 단일 조건으로 통합할 수 있다. 의사 코드(pseudo code)로 표현하면 아래와 같다.  $I$  은 슬롯의 수이다.

[0091] for  $m = 1 \rightarrow I$  do

[0092] if simd\_test\_zero(Mpassed) then

[0093] break

[0094] end if

[0095] 다차원 데이터를 슬롯 기준으로 순차적으로 로드하고, 조건 상수도 슬롯 기준으로 순차적으로 설정한다(S230).

의사 코드(pseudo code)로 표현하면 아래와 같다.  $W_m$ 은 다차원 데이터( $d_1^{[m]}, \dots, d_{R/b}^{[m]}$ )가 로드되는 레지스터이고,  $W_{pcm}$ 은 조건 상수가 설정되는 레지스터이다.

[0096]  $W_m \leftarrow \text{simd\_load}(d_1^{[m]}, \dots, d_{R/b}^{[m]})$

[0097]  $W_{pcm} \leftarrow \text{simd\_set}(pc[m])$

[0098] 슬롯-병행 비교 연산을 수행하여 중간 결과를 생성하고(S240), 마스크 값을 설정하여 평가자료를 생성(S250)한다. 의사 코드(pseudo code)로 표현하면 아래와 같다. slotMSB1는 슬롯의 최우선 비트(MSB)가 1로 설정된 마스크이고, ge와 le에 슬롯-병행 비교 결과를 저장한다. Mzeros는 모든 값이 1로 설정된 마스크이고, Mpassed는 알고리즘 결과를 결정하는 차원 코드를 저장하는 마스크, Mgt는 해당 बैं크에서 true인 1로 평가된 코드를 저장하

는 마스크이다.

[0099] `simd_slotparallelcomparison(Wm, Wpcm, SlotMSB1, ge, le)`

[0100] `Mpassed ← simd_and(Mpassed, simd_cmpeq(Mzeros, simd_andnot(Mgt, ~ge)))`

[0101] `Mgt ← simd_or(Mgt, simd_andnot(le, Mpassed))`

[0102] `Mpassed ← simd_andnot(Mgt, Mpassed)`

[0103] 그리고 나서 다시 처음으로 돌아가 다음 순번 슬롯에 대해 스캔을 하여야 하는 것인지 판단하는 과정(S210, S220)을 수행한다. 순환을 완료하거나 순환을 조기에 종료하면, 마스크 값에서 결과 데이터를 추출하고(S260), 스캔 절차를 종료한다. 의사 코드(pseudo code)로 표현하면 아래와 같다. r은 결과 비트 벡터이다.

[0104] `r ← simd_movemask(simd_cmpeq(Mones, Mgt))`

[0105] 도 11 내지 도 13은 본 발명에 의한 각 실험 예에서 연산 결과를 비교하는 그래프들이다.

[0106] 도 11 내지 도 13을 참조하면, 스캔 알고리즘의 효율을 알 수 있다. 다차원 데이터를 스캔하는 동안 평가 결과가 결정되면 추가 검사가 필요하지 않으나, 스캔이 여러 बैं크에서 동작할 때 모든 बैं크의 평가 결과가 결정되어야 순환을 정지할 수 있다.

[0107] x와 y가 k비트로 표현되는 정수이고, 둘 다  $[0, 2^k)$ 에서 균일하게 랜덤하다면  $x < y$ 와  $x > y$ 의 확률은 모두  $(2^k - 1/2^{k+1})$ 이고,  $x = y$ 의 확률은  $1/2^k$ 이다. n차원 값이 도메인에서 균일하게 분포되고 임의의 슬롯 형식  $S = (S_1, \dots, S_n)$ 이 사용된다고 가정한다. 차원 코드의 첫번째 बैं크와 조건 상수의 첫번째 बैं크를 비교하면 모든 슬롯 값의 확률이 상수보다 큰 확률은 아래 수학적 식 2와 같다.

### 수학적 식 2

$$\prod_{i=1}^n \left( \frac{2^{s_i} - 1}{2^{s_i} + 1} \right)$$

[0109] 슬롯 값 중 적어도 하나가 조건 상수의 대응 슬롯보다 작을 확률은 아래 수학적 식 3과 같다.

### 수학적 식 3

$$1 - \prod_{i=1}^n \left( \frac{2^{s_i} + 1}{2^{s_i} + 1} \right)$$

[0111] GREATER-THAN을 평가할 때 차원 값 및 조건 상수의 분포가 도메인에서 균일하면 t번째 순환 이후에 조기에 정지할 확률은 아래 수학적 식 4와 같다.

### 수학적 식 4

$$P_{DS}(b, S, t) = \left( \left( 1 - \prod_{i=1}^n \frac{2^{s_i \times t} + 1}{2^{s_i \times t} + 1} + \prod_{i=1}^n \frac{2^{s_i \times t} - 1}{2^{s_i \times t} + 1} \right) \right)^{\frac{R}{b}}$$

[0113] 이하 도 11 내지 13에서는 성능 평가 결과를 나타내고 있다. 성능 평가를 위해 장치는 16GB DDR4 메모리와 각 코어에는 32KB L1 명령어 캐시, 32KB L1 데이터 캐시 및 256KB L2 통합 캐시가 있고, 6MB L3 캐시는 모든 코어

에서 공유되는 Intel Core i5-7400 쿼드 코어 프로세서로 구성되어 있으며, 모든 메소드는 C++로 구현되었으며, 최적화 플래그가 -O3으로 설정된 g++ 5.4.0 컴파일러를 사용하여 컴파일 하였다. 모든 메소드가 동일한 루프 구조를 사용하였다. 모든 실험은 단일 스레드 내에서 실행되었다. 모든 실험에 대해 레지스터의 크기(R)로 256을 사용하였다.

- [0114] 바이트 슬라이스(ByteSlice)는 데이터를 메인 메모리에 수직으로 분배하는 저장 구조로 뱅크의 크기(b)가 8비트 이고, 데이터의 크기가 11비트인 경우 첫번째 뱅크에 앞의 8비트를 저장하고, 남은 3비트는 아래 쪽 다른 뱅크에 저장하며, 남은 5비트는 0으로 채우게 된다. 이 구조의 저장 방식은 8비트를 비교하고 참(TRUE) 또는 거짓(FALSE)을 평가할 수 없으면 다음의 8비트에 대해 비교를 수행한다.
- [0115] SIMDNaive는 차원 별로 차원 코드를 검사하는 방법으로 동일한 차원의 차원 값을 레지스터에 로드한 다음 동일한 조건 상수 벡터로 비교 연산을 수행하고 모든 차원을 검사할 때까지 점진적으로 결과를 업데이트하는 방법이다.
- [0116] DS8와 DS16은 디멘션 슬라이스(DimensionSlice)를 사용한 실험에서 뱅크의 크기가 8 비트인 경우와 16 비트인 경우를 각각 의미한다.
- [0117] DS16HBP은 단일 프로세서 워드에 여러 개의 컬럼형 값을 저장하는 수평 비트 병행(HBP, Horizontal Bit-Parallel) 구조로 수정된 DS16의 결과를 나타내는데 사용된다.
- [0118] 도 11을 참조하면, GREATER-THAN을 평가할 때 뱅크 크기(b)와 차원의 수(n)를 변경하여 순환한 후 초기에 종료할 확률을 알 수 있다. 가능한 한 공평하게 각 슬롯 크기를 구성한다고 가정하면, 슬롯 수(즉, 차원의 수)를 증가시키는 것은 초기 정지의 가능성을 약간 감소시킨다는 것을 알 수 있다. 한편, 뱅크 크기(b)와 슬롯 크기를 두 배로 늘리면 확률이 크게 증가한다. 그러나 뱅크 크기를 두 배로 늘리면 SIMD의 데이터 병렬 처리가 절반으로 감소한다.
- [0119] 도 12 내지 도 13을 참조하면, 합성 데이터를 사용했을 때 실험 수행 결과를 알 수 있다. 먼저  $2^{30}$  차원 코드로 구성된 합성 데이터 세트에 대한 실험을 수행한다. 각 도메인 내에서 차원 값이 일정하다고 가정하자. 차원 수를 늘릴 때 스캔을 수행하는 데 소비된 주기를 비교하였다. 차원 수를 늘리는 동안 각 차원을 나타내는데 사용되는 비트 수를 정의하는 두 가지 전략을 고려해본다.
- [0120] 도 12를 참조하면, 차원 수를 늘리는 동안 각 차원을 나타내는데 사용되는 비트 수를 정의하는 첫번째 전략을 고려한 수행 결과를 알 수 있다.
- [0121] 첫 번째 전략은 각 차원에 대한 비트 수를 변경하는 것이다. 이 경우 n이 증가하면 차원 코드를 나타내는 데 사용되는 총 비트 수도 증가한다. 각 측정 기준의 도메인 간격은  $[0, 2^{10})$ 이라고 가정하자. 따라서 n이 1씩 증가함에 따라 차원 코드에 사용되는 비트 수는 10만큼 증가한다. 조건 상수는 임의로 선택되어 10%의 선택도를 갖는다. 도 12는 차원 수를 늘릴 때의 스캔 성능을 보여준다. 당연히 SIMDNaive는 최악의 상황을 보였다. DS8과 DS16이 바이트 슬라이스(ByteSlice)와 DS16HBP보다 우위에 있음을 알 수 있다. 이 결과를 불필요한 simd\_loads 및 검사를 제거하는 초기 정지 전략에서 보여준 효과로 해석한다. DS16HBP는 각 비교에서 더 적은 연산을 사용하지만, (1) 각 슬롯의 추가 비트가 순환 횟수를 증가시키고, (2) 각 순환에서 검사된 비트를 줄이기 때문에 DS8 및 DS16보다 성능이 떨어진다.
- [0122] 그 다음 DS8과 DS16의 성능을 비교하여 뱅크 크기 b를 늘리는 효과를 평가한다.  $n < 5$ 일 때, DS8이 DS16보다 성능이 우수하다. 그러나 n이 4에서 5로 증가할 때, DS8의 성능 저하가 현저하고, DS16은 이 시점에서 DS8보다 성능이 좋아진다. 조사한 결과 DS8에서 이 시점에서 단일 비트 크기의 슬롯이 나타나기 시작하여 각 차원을 수직으로 저장하는 데 필요한 뱅크의 불균형이 크게 증가한다는 것을 알았다. 특히, DS8에서  $n = 5$ 인 경우 슬롯 형식은 (2, 2, 2, 1, 1)로 구성되며, 두 차원은 10개의 수직 뱅크를 갖지만 나머지는 5개만 필요하다. 이 불균형은 순환 횟수와 사용되지 않는 비트 수의 증가를 야기한다. 일반적으로 n슬롯으로 나누기에 충분한 사이즈를 가지지 못하는 뱅크에 의해 야기되는 각 차원에 필요한 수직 뱅크의 과도한 불균형이 없다면, 디멘션 슬라이스(DimensionSlice)에서 더 작은 뱅크 크기를 사용하는 것이 권장되는 것으로 결론지을 수 있다. 이 결론은 실제 데이터 세트를 사용하여 다음 실험에서 다시 확인된다.
- [0123] 도 13을 참조하면, 차원 수를 늘리는 동안 각 차원을 나타내는데 사용되는 비트 수를 정의하는 두번째 전략을 고려한 수행 결과를 알 수 있다.

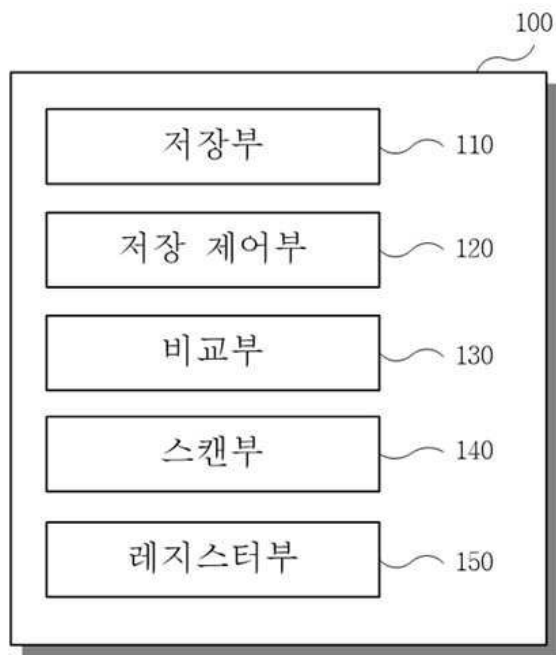
- [0124] 두 번째 전략은  $n$ 을 늘리는 동안 코드 크기를 변경하는 것이다. 코드를 표현하는데 사용되는 총 비트 수는 24로 고정된 것으로 가정하자.  $n$ 을 증가시키면  $n$ 개의 차원을 나타내는 데 사용되는 비트 수는 가능한 한 공평하고 합계는 24이다. 예를 들어,  $n = 5$  인 경우 각 차원을 나타내는 데 사용되는 비트 수는 4, 4, 4, 4 및 5이다. 선택도는 10 %로 고정되어 있다. 도 13은 코드를 나타내는 데 사용된 비트 수를 고정시키고,  $n$ 을 증가시킬 때의 스캔 성능을 보여준다. 도 13에서 볼 때 DS8 및 DS16은 바이트 슬라이스(ByteSlice) 및 DS16HBP보다 우수한 성능을 보인다. DS8과 DS16의 코드당 사이클(cycle/code)은  $n$ 이 증가함에 따라 약간 증가한다는 것을 알 수 있다. 이것은 도 11에 제시된 예상 조기 정지 확률의 결과와 일치한다. 바이트 슬라이스(ByteSlice)의 코드 당 사이클은 디멘션 슬라이스(DimensionSlice)보다 더 가파른 경향이 있다. 이는 바이트 슬라이스(ByteSlice)가 각 차원을 순차적으로 검사하기 때문에 순환 횟수가 선형 증가하는 반면 디멘션 슬라이스(DimensionSlice)는 코드에 사용된 총 비트 수가 고정되어 있으면 "거의 일정한" 순환이 필요하기 때문이다. 이전에 언급한 바와 같이 각 차원을 수직으로 저장해야 하는 बैं크의 불균형으로 인해 순환 횟수가 증가할 수 있으므로 "거의 일정한"이라는 용어를 사용한다. 또한,  $n = 5$ 에서 DS8의 성능 저하는 이러한 측면을 설명한다.
- [0125] 디멘션 슬라이스(DimensionSlice)는 레지스터의 크기와 명령어를 고려하여 다차원 데이터를 슬롯단위로 분할하고, 각 차원에서 동일한 순번을 가지는 슬롯들로 बैं크를 형성하여 저장하는 구조와 이러한 구조를 이용하여 다차원 데이터와 조건 상수와 순차적으로 슬롯 단위로 비교하며, 비교 과정은 슬롯-병행 비교로 중간 결과를 생성하고, 설정된 마스크 값과 비트연산을 수행한 결과가 참인지 거짓인 평가하여 참 또는 거짓이 확정되면 다음 순번의 슬롯에 대한 스캔을 하지 않고 조기 종료를 가능하게 하는 스캔 알고리즘을 개시하고 있다.
- [0126] 이와 같은, 인-메모리 기반의 다차원 데이터 관리 방법은 어플리케이션으로 구현되거나 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다.
- [0127] 상기 컴퓨터 판독 가능한 기록 매체에 기록되는 프로그램 명령어는 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수도 있다.
- [0128] 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD 와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다.
- [0129] 프로그램 명령어의 예에는, 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.
- [0130] 이상에서는 실시 예들을 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

## 부호의 설명

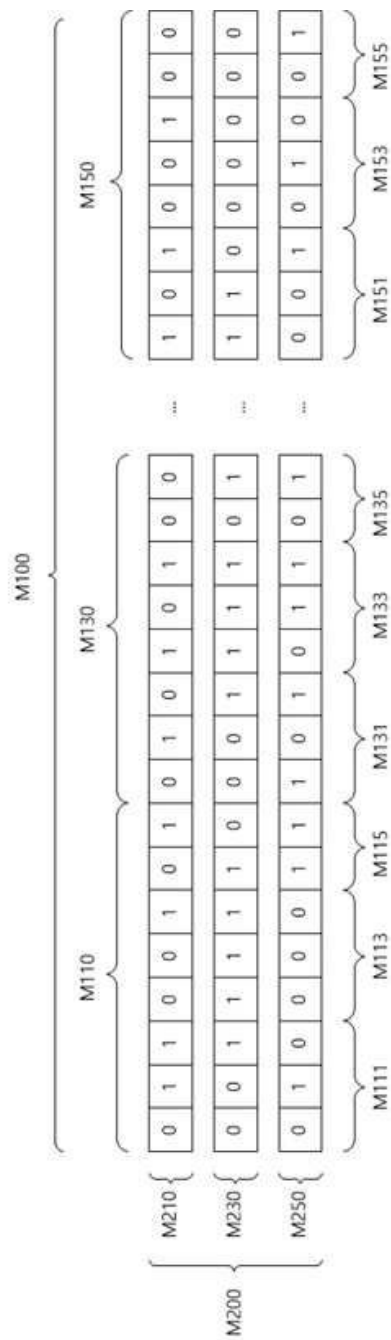
- [0132] 100: 인-메모리 기반의 다차원 데이터 관리 장치  
110: 저장부  
120: 저장 제어부  
130: 비교부  
140: 스캔부  
150: 레지스터부

도면

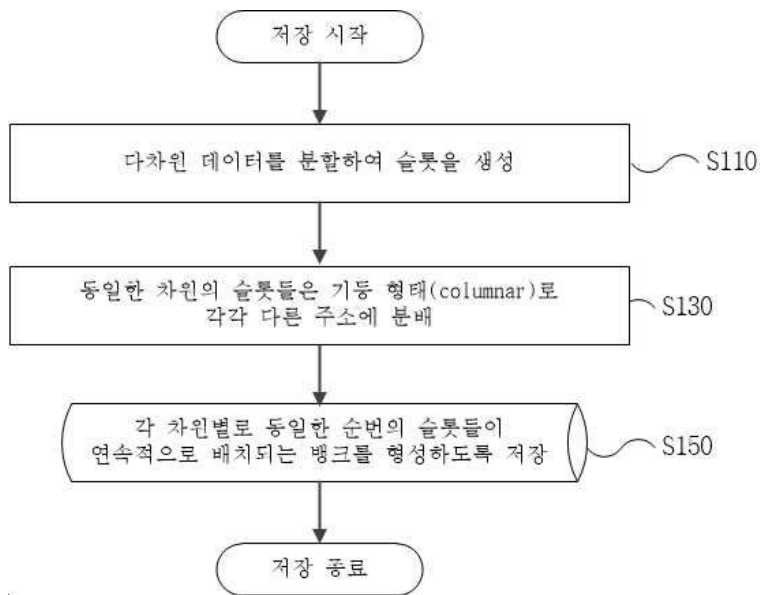
도면1



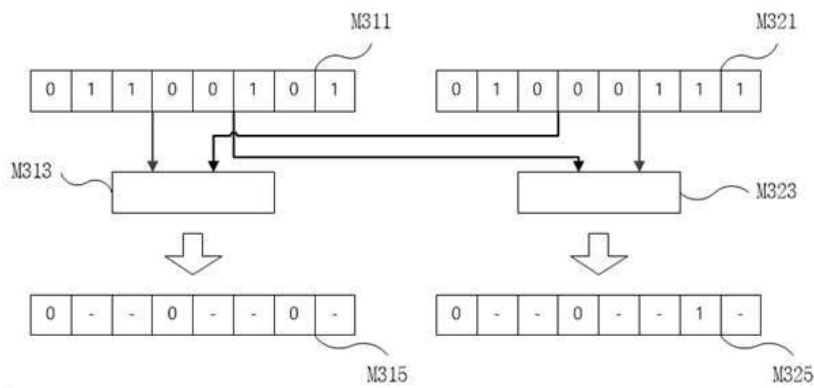
도면2



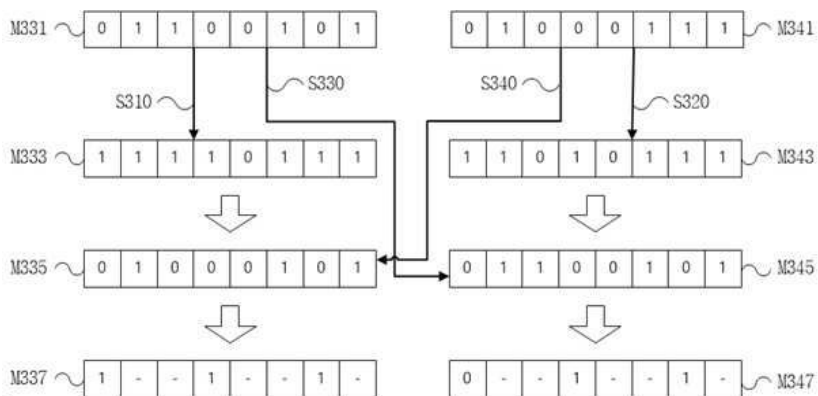
도면3



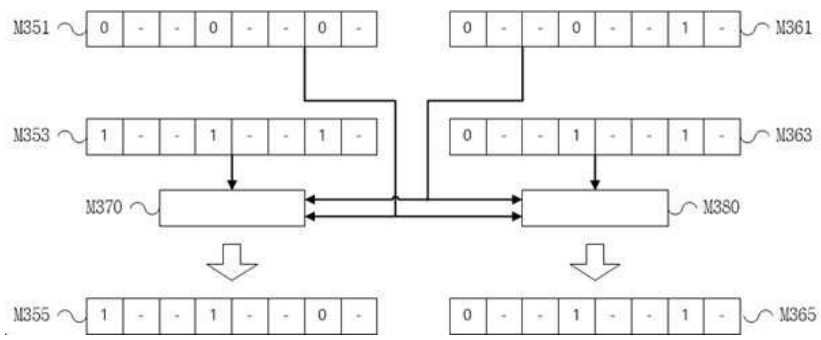
도면4



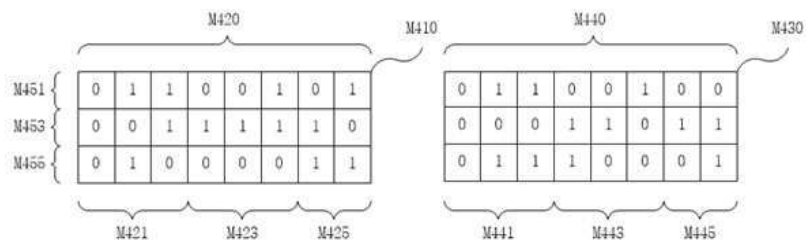
도면5



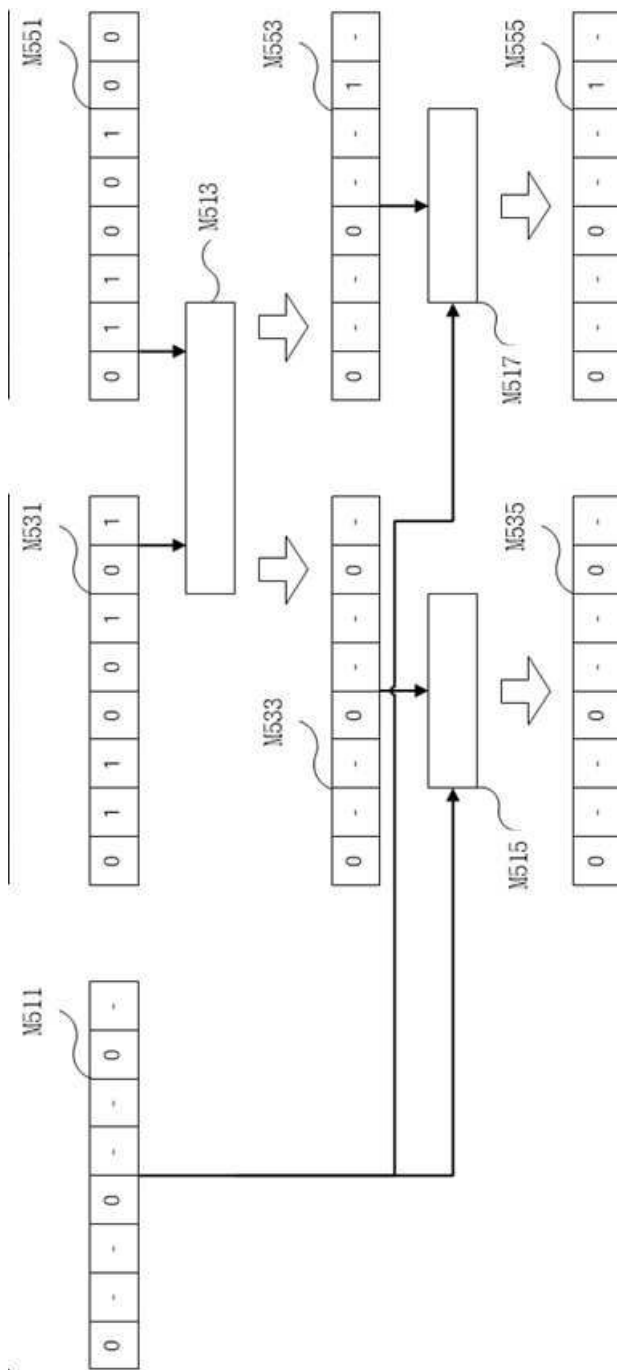
도면6



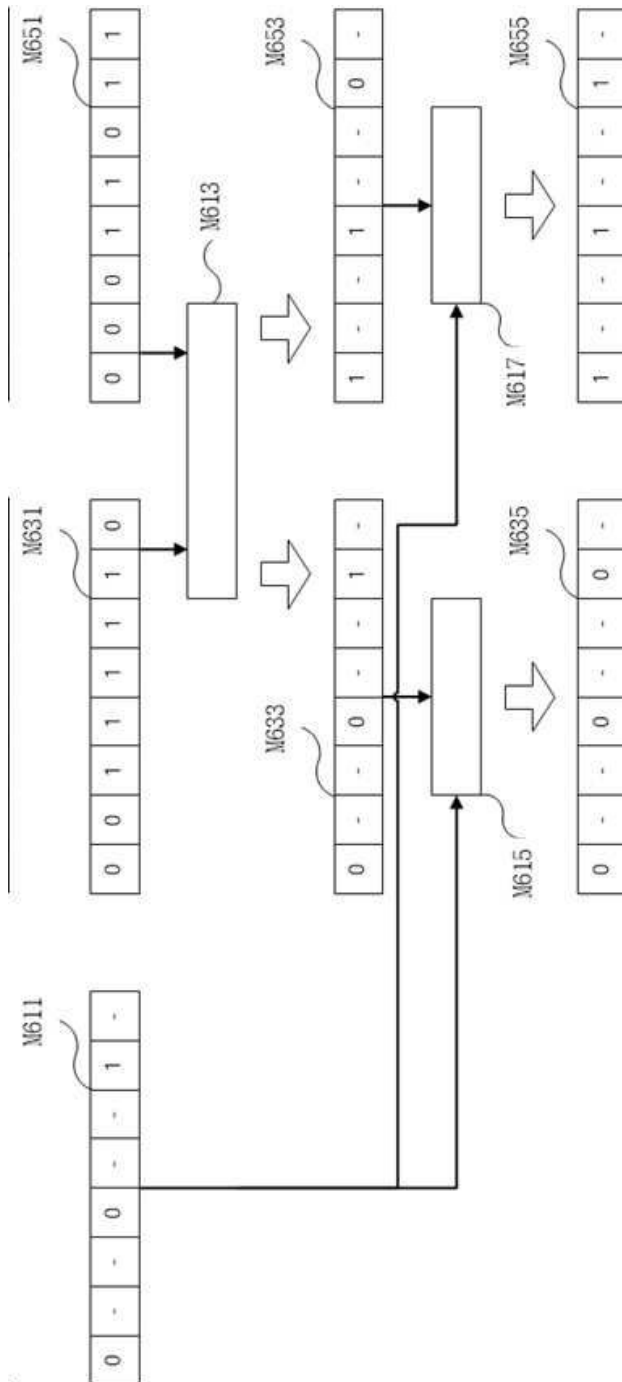
도면7



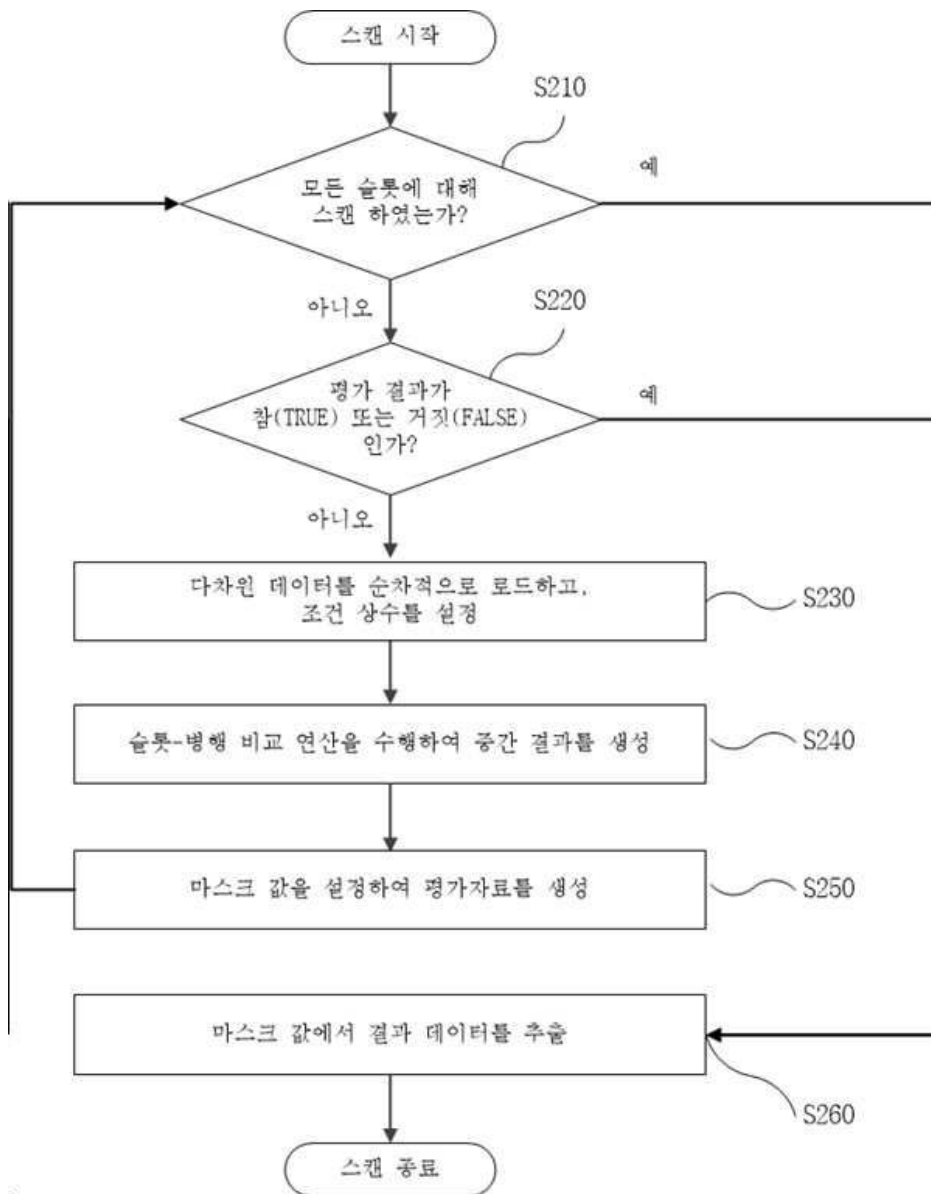
도면8



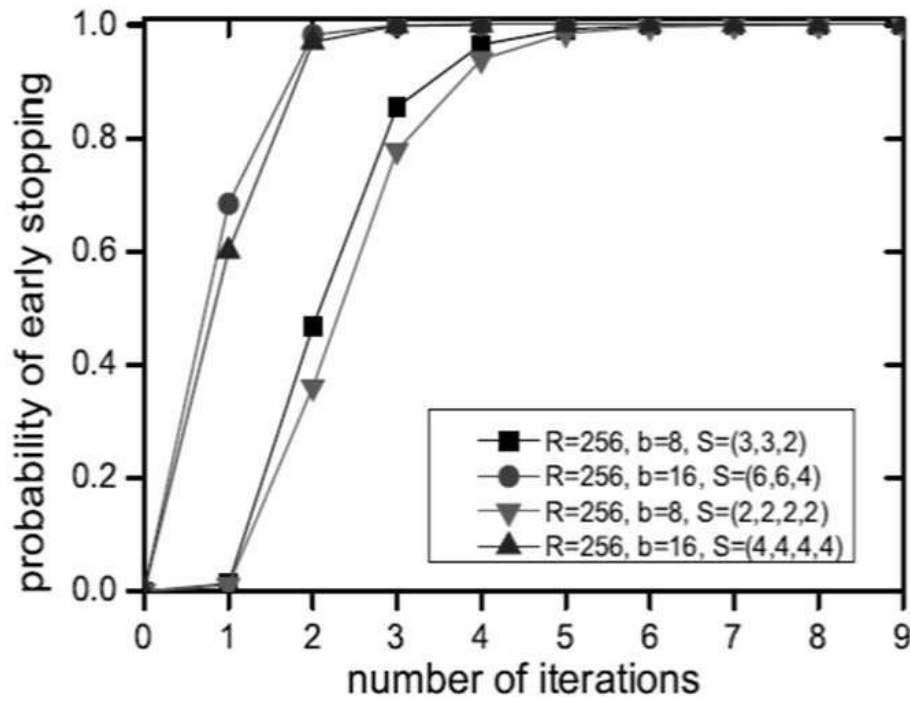
도면9



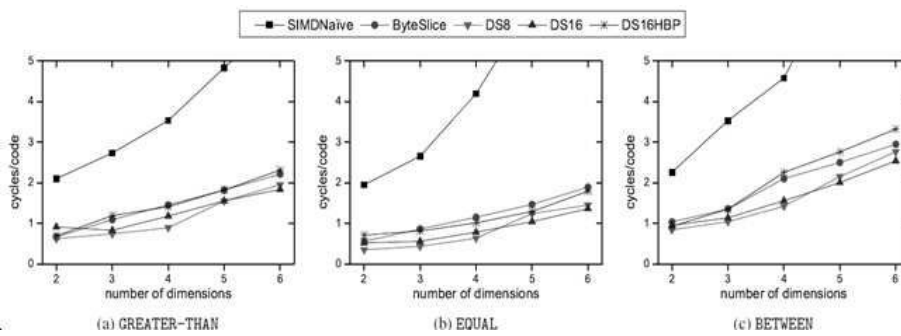
도면10



도면11



도면12



도면13

