

	(19) 대한민국특허청(KR) (12) 공개특허공보(A)	(11) 공개번호 10-2014-0049204 (43) 공개일자 2014년04월25일
(51) 국제특허분류(Int. Cl.) G06F 9/28 (2006.01) G06F 9/38 (2006.01) (21) 출원번호 10-2012-0115118 (22) 출원일자 2012년10월17일 심사청구일자 2012년10월17일		(71) 출원인 인하대학교 산학협력단 인천광역시 남구 인화로 100, 인하대학교 (용현동) (72) 발명자 최원익 인천 연수구 해돋이로84번길 10, 607동 504호 (송도동, 송도풍림아이원6단지아파트) 김민철 인천 남구 토금남로 79, 101동 811호 (용현동, 우성아파트) (74) 대리인 양성보

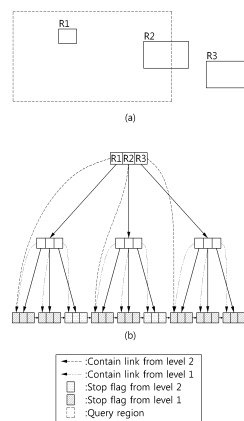
전체 청구항 수 : 총 5 항

(54) 발명의 명칭 **질의 영역과 노드 간의 포함 관계를 이용한 R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법**

(57) 요약

질의 영역과 노드 간의 포함 관계를 이용한 R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법이 개시된다. R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법은 검색 노드와 질의 영역 간의 포함(contain) 관계를 이용하여 R-트리 계열의 계층형 색인 구조에서 범위 질의를 처리하는 단계를 포함할 수 있으며, 이때 상기 R-트리 계열의 계층형 색인 구조에는, 상기 검색 노드가 질의 영역과 포함 관계에 있을 때 상기 검색 노드에 포함된 자식 노드를 검색하지 않고 바로 이동할 단말 노드에 대한 링크를 나타내는 컨테인 링크(contain link)가 포함될 수 있다.

대표도 - 도1



이 발명을 지원한 국가연구개발사업

과제고유번호 2011- 0026987

부처명 교육과학기술부

연구사업명 기초연구사업 일반연구자지원사업

연구과제명 GPU를 이용한 시공간데이터베이스 가속화 기법

기 여 율 1/1

주관기관 인하대학교 산학협력단

연구기간 2010.09.01 ~ 2015.08.31

특허청구의 범위

청구항 1

검색 노드와 질의 영역 간의 포함(contain) 관계를 이용하여 R-트리 계열의 계층형 색인 구조에서 범위 질의를 처리하는 단계

를 포함하고,

상기 R-트리 계열의 계층형 색인 구조에는,

상기 검색 노드가 질의 영역과 포함 관계에 있을 때 상기 검색 노드에 포함된 자식 노드를 검색하지 않고 바로 이동할 단말 노드에 대한 링크를 나타내는 컨테인 링크(contain link)가 포함되는 것

을 특징으로 하는 R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법.

청구항 2

제1항에 있어서,

상기 컨테인 링크는 상기 검색 노드의 자식 노드들이 포함하는 단말 노드들 중 가장 앞에 위치한 노드를 가리키며, 검색 레벨에 대한 정보를 포함하는 것

을 특징으로 하는 R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법.

청구항 3

제1항에 있어서,

상기 R-트리 계열의 계층형 색인 구조에는,

상기 컨테인 링크를 통해 상기 단말 노드로 이동한 검색에 대해 객체 아이디의 검색을 중단해야 하는 위치를 나타내는 정지 플래그(stop flag)가 더 포함되는 것

을 특징으로 하는 R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법.

청구항 4

제1항에 있어서,

상기 범위 질의를 처리하는 단계는,

CPU(central processing unit)에서 상기 R-트리 계열의 레벨에 대응되는 개수의 쓰레드(thread)를 생성하는 단계;

GPU(graphic processing unit)에서 상기 검색 노드와 질의 영역 간의 관계를 판단하여 상기 질의 영역과 포함 관계에 있는 노드들을 제1 리스트에 저장하고 상기 질의 영역과 겹침(intersect) 관계에 있는 노드들을 제2 리스트에 저장하는 단계; 및

상기 GPU에서 상기 제1 리스트를 이용하여 다음 레벨의 검색을 수행하고 상기 CPU에서 상기 제2 리스트를 이용하여 상기 쓰레드를 통해 상기 단말 노드의 객체 아이디를 검색하는 단계

를 포함하는 R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법.

청구항 5

제1항에 있어서,

상기 범위 질의를 처리하는 단계는,

상기 R-트리 계열의 계층형 색인 구조에서 너비 우선 검색(breadth first search) 방식을 적용하여 상기 범위 질의를 처리하는 것

을 특징으로 하는 R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법.

명세서

기술분야

[0001] 본 발명의 실시예들은 R-트리 계열의 계층형 색인 구조에서 범위 질의를 처리하는 방법에 관한 것이다.

배경기술

[0002] R-tree는 공간자료 데이터베이스 시스템에서 가장 널리 사용되는 자료구조로서, 다차원의 자료를 관리하는데 매우 적합하며, 공간 데이터에 대한 삽입, 삭제, 검색 등의 작업을 효율적으로 수행할 수 있는 구조를 가지고 있는 색인 구조이다.

[0003] 한국공개특허 제10-2011-0068578호(공개일 2011년 06월 22일)에는 GPU(Graphics Processing Units)를 트리 형태의 색인 구조에 적용하여 R-tree에서 범위 질의(Range Query)를 처리하는 기술이 개시되어 있다.

[0004] 질의 수행과정에서 검색 대상 노드들은 질의 영역에 포함(contain)된 노드와 겹침(intersect)된 노드를 구별할 수 있다. 그러나, 포함 관계에 있는 노드들의 경우 해당 노드에 포함되는 자식 노드들이 모두 질의 영역에 포함되어 있음에도 불구하고 자식 노드들을 모두 검색하는 불필요한 연산이 존재하기 때문에 R-tree에서의 검색 성능이 느려지는 문제가 있다.

발명의 내용

해결하려는 과제

[0005] R-트리 계열의 계층형 색인 구조의 범위질의 처리과정에서 불필요한 연산을 제거할 수 있는 방법을 제공한다.

[0006] 질의 영역과 노드 간의 관계를 나타내는 노드 구조를 이용한 범위 질의 처리 방법을 제공한다.

과제의 해결 수단

[0007] 본 발명의 실시예에 따르면, R-트리 계열의 계층형 색인 구조의 범위질의 병렬 가속화 방법은 검색 노드와 질의 영역 간의 포함(contain) 관계를 이용하여 R-트리 계열의 계층형 색인 구조에서 범위 질의를 처리하는 단계를 포함할 수 있으며, 이때 상기 R-트리 계열의 계층형 색인 구조에는, 상기 검색 노드가 질의 영역과 포함 관계에 있을 때 상기 검색 노드에 포함된 자식 노드를 검색하지 않고 바로 이동할 단말 노드에 대한 링크를 나타내는 컨테인 링크(contain link)가 포함될 수 있다.

[0008] 일 측면에 따르면, 상기 컨테인 링크는 상기 검색 노드의 자식 노드들이 포함하는 단말 노드들 중 가장 앞에 위치한 노드를 가리키며, 검색 레벨에 대한 정보를 포함할 수 있다.

[0009] 다른 측면에 따르면, 상기 R-트리 계열의 계층형 색인 구조에는, 상기 컨테인 링크를 통해 상기 단말 노드로 이동한 검색에 대해 객체 아이디의 검색을 중단해야 하는 위치를 나타내는 정지 플래그(stop flag)가 더 포함될 수 있다.

[0010] 또 다른 측면에 따르면, 상기 범위 질의를 처리하는 단계는, CPU(central processing unit)에서 상기 R-트리 계열의 레벨에 대응되는 개수의 스레드(thread)를 생성하는 단계; GPU(graphic processing unit)에서 상기 검색 노드와 질의 영역 간의 관계를 판단하여 상기 질의 영역과 포함 관계에 있는 노드들을 제1 리스트에 저장하고 상기 질의 영역과 겹침(intersect) 관계에 있는 노드들을 제2 리스트에 저장하는 단계; 및 상기 GPU에서 상기 제1 리스트를 이용하여 다음 레벨의 검색을 수행하고 상기 CPU에서 상기 제2 리스트를 이용하여 상기 스레드를 통해 상기 단말 노드의 객체 아이디를 검색하는 단계를 포함할 수 있다.

[0011] 또 다른 측면에 따르면, 상기 범위 질의를 처리하는 단계는, 상기 R-트리 계열의 계층형 색인 구조에서 너비 우선 검색(breadth first search) 방식을 적용하여 상기 범위 질의를 처리할 수 있다.

발명의 효과

[0012] 본 발명의 실시예에 따르면, 질의 영역에 포함되는 노드의 경우 해당 노드에 포함되는 자식 노드들을 검색하지 않고 바로 검색 노드에 포함된 단말 노드로 이동해 객체 아이디를 읽어옴으로써 연산량을 줄일 수 있다.

[0013] 본 발명의 실시예에 따르면, 각 노드가 질의 영역과 포함 관계를 가질 때 이동해야 하는 단말 노드에 대한 링크와 함께, 단말 노드로 이동한 검색에 대해 객체 아이디 읽기를 중단해야 하는 위치를 표시하는 정지 플래그를 이용함으로써 범위 질의 처리 과정에서 불필요한 연산을 효과적으로 제거하여 검색 성능을 향상시킬 수 있다.

도면의 간단한 설명

[0014] 도 1은 본 발명의 일 실시예에 있어서, 컨테인 링크를 포함한 노드 구조를 설명하기 위한 도면이다.
 도 2는 본 발명의 일 실시예에 있어서, 컨테인 링크를 이용한 GPU에서의 검색 과정을 설명하기 위한 도면이다.
 도 3은 본 발명의 일 실시예에 있어서, G-tree의 전체 검색 과정을 도시한 흐름도이다.
 도 4는 CPU의 메모리 내에 저장되어 있는 R-tree의 노드 분포를 나타낸 것이다.
 도 5는 배열을 활용한 노드 저장 방식을 설명하기 위한 도면이다.
 도 6은 도 5의 노드 저장 방식에 대한 GPU의 접근 패턴을 나타낸 도면이다.
 도 7은 깊이 우선 검색 방식과 너비 우선 검색 방식을 설명하기 위한 도면이다.
 도 8 내지 도 10은 본 발명에서 제안한 G-tree와 GPU에서의 선형검색, R-tree 계열 중에서 저장된 인덱싱에 많이 사용되는 R*-tree의 성능을 비교한 그래프이다.
 도 11 내지 도 13은 도 8 내지 도 10에서 G-tree와 GPU에서의 선형검색부분만을 확대하여 그린 그래프이다.

발명을 실시하기 위한 구체적인 내용

[0015] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0016] 본 실시예들은 다차원의 공간 데이터를 저장하는 색인 자료구조인 R-tree 계열의 계층형 색인 구조에서 범위 질의를 처리하는 검색 시스템에 적용될 수 있다. 본 명세서에서, 'R-tree'는 R*-tree, X-tree, SS-tree 등 모든 R-tree 변종 계열을 통칭한 것이다.

[0017] 본 실시예에서는 범위 질의를 처리하는 과정에서 검색 양을 줄이기 위한 목적으로 컨테인 링크(contain link)를 이용한다.

[0018] 컨테인 링크와 이를 이용한 검색 과정에 대해서 상세히 설명하기로 한다.

[0019] 컨테인 링크

[0020] R-tree에서는 질의 영역과 검색 노드 간 MBR(minimum bounding rectangle)의 관계가 겹침(intersect) 과 포함(contain)으로 나누어질 수 있다. 이때, 질의 영역과 검색 노드가 겹치는 경우에는 자식 노드들 중에서 실제로 질의 영역에 포함되지 않는 노드들이 존재하기 때문에 자식 노드들에 대한 질의 영역과의 관계를 검사해야만 한다. 하지만, 질의 영역이 검색 노드를 포함하는 경우에는 검색 노드의 자식 노드들이 모두 질의 영역에 포함되므로 질의 영역과 검색 노드가 겹칠 때와는 다르게 자식 노드들을 검사할 필요가 없다. 따라서, 포함 관계에 있는 노드의 경우에는 자식 노드들을 검사하지 않고 바로 검색 노드에 포함된 단말 노드로 이동해 객체 아이디만을 읽어오는 것으로 연산량을 줄일 수 있다.

[0021] 이러한 과정을 수행하기 위해서 R-tree에 컨테인 링크와 정지 플래그를 추가한다. 이때, 컨테인 링크는 각 노드가 질의 영역과 포함 관계를 가질 때 이동해야 하는 단말 노드에 대한 링크를 나타낸다. 또한, 각 단말 노드는 컨테인 링크를 통해 단말 노드로 이동한 검색에 대해 객체 아이디 읽기를 중단해야 하는 위치를 표시하고 있는 정지 플래그를 포함하고 있다.

[0022] 도 1에서 (a)는 질의 예시를 도시한 것이고, (b)는 (a)의 질의에 대한 컨테인 링크와 정지 플래그를 포함하고 있는 노드 구조를 도시한 것이다. 각 노드의 컨테인 링크는 자신의 자식 노드들이 포함하는 단말 노드들 중 가장 앞에 있는 노드를 가리킨다. 이러한 컨테인 링크에는 검색이 어느 레벨에서 링크를 타고 내려왔는지에 대한 정보를 포함하고 있다. 도 1의 (b)에서 각 레벨 별로 링크가 다르게 도시되어 있음을 알 수 있다. 이런 링크의 정보를 이용해서 단말 노드 검색에서는 검색이 정지해야 하는 정보를 담고 있는 정지 플래그를 이용해서 검색을 진행한다.

- [0023] 도 1의 (a)에서 노드 R1은 질의 영역에 포함되어 있는 노드이므로 자식 노드에 대한 검색을 중단하고, 컨테인 링크를 따라서 단말 노드로 바로 이동한다. 이 경우 레벨 2로부터 이동한 검색이기 때문에 단말 노드로 이동한 검색은 연한 회색으로 음영 처리된 노드를 만날 때까지 단말 노드를 따라서 이동하며 단말 노드에 기록된 객체 아이디를 읽는다. 도 1의 (a)에서 노드 R2는 질의 영역과 겹치기 때문에 단말 노드로 바로 이동할 수가 없고, 기존의 R-tree와 마찬가지로 자식 노드에 대한 검색을 계속해서 수행해야 한다.
- [0024] 검색 과정
- [0025] 도 2는 컨테인 링크를 이용한 검색 과정을 설명하기 위한 예시 화면이다.
- [0026] GPU에서의 검색은 도 2의 (a)와 같이 시작한다. 도 2의 (a)에는 현 단계에서 검색해야 하는 노드를 GPU의 질의 결과 인덱스 리스트로부터 읽어서 가져오는 과정이 도시되어 있다. GPU의 질의 결과 인덱스 리스트로부터 검색할 노드들을 읽어온 후, 도 2의 (b)에서 GPU의 검색 결과에 따라서 GPU의 질의결과 인덱스 리스트에 검색을 수행해야 할 인덱스를 기록하고 질의 결과 겹침 플래그(Intersect Flag) 리스트와 질의 결과 포함 플래그(Contain Flag) 리스트에 각각 자식 노드의 겹침과 포함의 관계를 기록한다.
- [0027] 이런 결과를 바탕으로 도 2의 (c)에 도시한 바와 같이 겹침 관계에 있는 노드들은 다시 GPU로 보내 도 2의 (a)를 통해 설명한 단계부터 다시 검색을 수행하고, 포함 관계에 있는 노드들은 컨테인 링크를 이용해서 단말 노드로 이동한다. 이동한 단말 노드에서부터 CPU를 이용하여 결과 노드들의 객체 아이디를 결과 배열에 저장한다.
- [0028] 도 3은 전체 검색과정을 나타낸다.
- [0029] 단계(S310)에서 CPU는 검색을 시작하기 위해서 G-tree 레벨의 숫자만큼 쓰레드를 생성한다. 이 쓰레드들은 각 레벨에서 포함 관계에 있는 노드를 처리하기 위해서 사용된다. 쓰레드를 생성한 후 리스트 N에는 루트 노드를 저장하고 리스트 C는 초기화를 한다. 또한 변수 i에는 현재 검색하는 레벨을 저장한다.
- [0030] 이어, 단계(S320)에서 GPU는 자식 노드들의 포함 관계를 계산하고 결과를 저장한다. 저장된 결과를 이용해서 리스트 N에 겹침 관계에 있는 노드들을 저장하고 리스트 C에는 포함 관계에 있는 노드들을 저장한다. 이때 리스트 N은 GPU 메모리에 있는 리스트이고 리스트 C는 CPU에 있는 리스트이다.
- [0031] 단계(S330)에서는 리스트 N을 이용해서 GPU에서 다음 레벨의 검색을 수행하고, 리스트 C를 이용해서 미리 만들어 놓은 쓰레드에서 단말 노드의 객체 아이디를 찾아낸다. 이 과정을 단말 노드까지 반복해서 수행한다.
- [0032] 앞서 설명한 컨테인 링크를 이용한 검색 과정을 GPU에서 병렬 처리함으로써 추가적인 성능 향상을 이룰 수 있다.
- [0033] G-tree의 기본적인 구조와 범위 질의 병렬화 방법에 대해서 설명하면 다음과 같다.
- [0034] G-tree의 노드 구조
- [0035] R-tree는 노드를 삽입할 때 메모리에 새로운 공간을 할당하고 이곳에 노드를 저장한다. 이러한 노드 저장방식은 R-tree의 노드들이 메모리 곳곳에 퍼져있게 만든다. 도 4는 이와 같은 CPU의 메모리 내에 저장되어 있는 R-tree의 노드 분포를 나타낸 것이다. 하지만 이와 같은 CPU의 메모리 저장 방식은 GPU 메모리 접근 특성에 적합하지 않다. 따라서, GPU를 위해서 R-tree를 구성할 때는 노드들이 메모리에 선형적으로 정렬되어 있어야 한다. 이와 같은 저장 방식을 이루기 위해서 R-tree의 노드들을 배열에 선형적으로 저장한다. 그러나 단순히 배열에 노드들을 저장하는 것만으로는 최적의 메모리 읽기 성능을 이룰 수 없다.
- [0036] 도 5는 배열을 활용한 두 가지 방식의 차이를 보여준다. 도 5의 (a)와 같은 구조체를 배열에 저장할 때는 도 5의 (b)와 (c) 같은 방식으로 저장할 수 있다. 도 5의 (b)는 일반적으로 구조체를 배열 형태로 저장한 방식이다.
- [0037] 이와 같은 방식으로 저장할 경우 메모리에서 선형적인 접근은 가능하지만 메모리 트랜잭션 내에 하나의 워프가 정렬되어 있지 못하다. 따라서, 추가적인 메모리 접근이 필요하게 된다. 하지만 도 5의 (c)와 같은 배열의 구조체 방식은 다수의 쓰레드들이 동시에 접근하는 구조체의 요소들(min_x, min_y 등)이 인접한 메모리에 존재한다. 그러므로, 이와 같은 저장 구조를 이용할 경우 선형적으로 정렬된 메모리 접근이 가능하게 되어 최적의 메모리 읽기가 가능하게 된다.
- [0038] 도 6은 도 5에 대한 GPU의 접근 패턴을 나타낸 그림이다. 도 6의 (a)는 도 5의 (b)에 대한 접근 패턴이다. 이와 같은 접근 패턴에서는 구조체의 각 요소들이 메모리에 퍼져있기 때문에 한번의 메모리 트랜잭션으로는 하나

의 워프 내의 쓰레드들이 모두 필요한 데이터를 읽어 올 수 없다. 따라서 도 5의 (c)를 접근하는 방식인 도 6의 (b)와 같은 접근 패턴을 사용해야 한다. 그러나 도 6의 (b)에는 첫 번째 요소(min_x)에 대해서는 최적의 읽기를 이룰 수 있지만, 두 번째 요소(min_y)에 대해서는 2번의 읽기를 수행해야만 워프 내의 모든 쓰레드들이 필요한 데이터를 읽어 올 수 있게 된다. 그러므로 도 6의 (b)는 도 6의 (c)와 같이 최적화를 수행할 있다. 도 6의 (c)는 디스크 기반의 R-tree에서 주로 사용되는 디스크 블록크기(4KB)로 노드의 크기를 결정하는 것과는 다르게 각 노드 별로 요소의 개수를 32의 배수로 결정함으로써 최적의 메모리 읽기 성능을 얻을 수 있다.

[0039] 범위 질의 병렬화 방법

[0040] 일반적으로 R-tree에서는 깊이 우선 검색(Depth-First-Search, DFS)을 주로 사용한다. 깊이 우선 검색 방식은 질의 영역과 겹치는 노드를 발견하면 바로 자식 노드로 이동하여 검색을 진행한다. 이와 같은 검색 방식은 하위 레벨의 노드와 상위 레벨의 노드 사이의 이동이 빈번하게 일어나 상위 레벨 노드와 하위 레벨 노드 간의 데이터 의존성이 생긴다. 이와 같은 의존성은 GPU에서 병렬로 데이터를 처리하는데 방해요소로 작용한다.

[0041] 따라서 데이터 의존성을 제거하기 위해서 G-tree에서는 깊이 우선 검색 대신 너비 우선 검색(Breadth-First-Search, BFS)을 사용한다. 너비 우선 검색은 각 레벨의 검색이 끝나기 전까지는 하위 레벨로 이동하지 않기 때문에 데이터 의존성을 크게 줄일 수 있다.

[0042] 도 7은 깊이 우선 검색 방식과 너비 우선 검색 방식의 차이점을 도시하고 있다. 도 7의 (a)는 깊이 우선 검색 방식을 나타내는 그림으로, 괄호의 숫자는 검색 순서를 나타낸다. 도 7의 (b)는 너비 우선 검색을 나타내는 그림으로, 위에서와 마찬가지로 괄호의 숫자는 검색 순서를 나타낸다. 도 7의 (b)에서는 각 레벨을 동시에 처리함으로써 GPU에서의 병렬화가 가능함을 보여준다.

[0043] 아래 표 1의 환경에서 범위 질의 처리 과정을 실험하였다.

표 1

CPU	i7-960 3.2GHz	
RAM	12GB	
GPU	GTX 580	
	코어	512
	RAM	1.5GB
	Compute Capability	2.0

[0045] 표 1과 같이, CPU는 인텔사의 i7-960으로 3.2GHz의 클럭을 가지고 있다. 사용된 PC의 메모리는 총 12GB이며, GPU는 Nvidia사의 GTX 580이다. GTX 580은 512개의 코어와 1.5GB의 메모리를 가지고 있으며, compute capability는 2.0이다. 이에, 이미지에서 추출한 SURF 특징점 데이터로부터 2~4차원을 뽑아 120~590만개를 사용하였다.

[0046] 도 8, 9, 10은 본 발명에서 제안한 G-tree와 GPU에서의 선형검색, R-tree계열 중에서 저차원 인덱싱에 많이 사용되는 R*-tree의 성능을 비교한 그래프이다. 도 8은 120만개의 데이터, 도 9는 350만개의 데이터, 도 10은 590만개의 데이터에 대한 각 기법의 성능을 나타낸다. 각 그래프 상의 최상위의 3개의 선이 R*-tree의 성능으로 G-tree, GPU에서의 선형검색에 비해서 나쁜 성능을 보여준다.

[0047] 도 11, 12, 13은 도 8, 9, 10에서 G-tree와 GPU에서의 선형검색부분만을 확대하여 그린 그래프이다. 각 그래프에서 상단의 3개의 선이 GPU에서 선형 검색을 의미하며, G-tree에 비해서 성능이 나쁘다는 것을 확인할 수 있다. G-tree와 GPU의 선형 검색은 모두 선택도에 대해서 선형적인 시간 증가를 보이지만 G-tree가 그 증가폭이 더 낮다는 것을 도 11, 12, 13을 통해서 확인할 수 있다.

[0048] 이와 같이, 본 발명의 실시예에 따르면, 각 노드가 질의 영역과 포함 관계를 가질 때 이동해야 하는 단말 노드에 대한 링크와 함께, 단말 노드로 이동한 검색에 대해 객체 아이디 읽기를 중단해야 하는 위치를 표시하는 정지 플래그를 이용함으로써 범위 질의 처리 과정에서 불필요한 연산을 효과적으로 제거하여 검색 성능을 향상시킬 수 있다.

[0049] 본 발명의 실시예에 따른 방법들은 다양한 컴퓨터 시스템을 통하여 수행될 수 있는 프로그램 명령(instruction) 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은

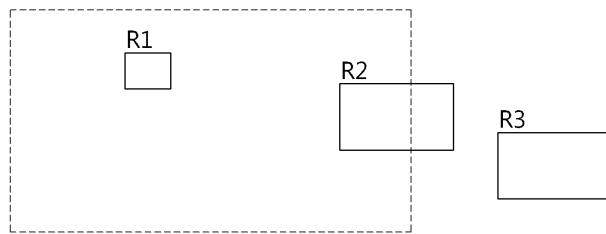
본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것 일 수도 있다. 또한, 상술한 파일 시스템은 컴퓨터 판독이 가능한 기록 매체에 기록될 수 있다.

[0050] 이상과 같이 본 발명은 비록 한정된 실시예와 도면에 의해 설명되었으나, 본 발명은 상기의 실시예에 한정되는 것은 아니며, 본 발명이 속하는 분야에서 통상의 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형이 가능하다.

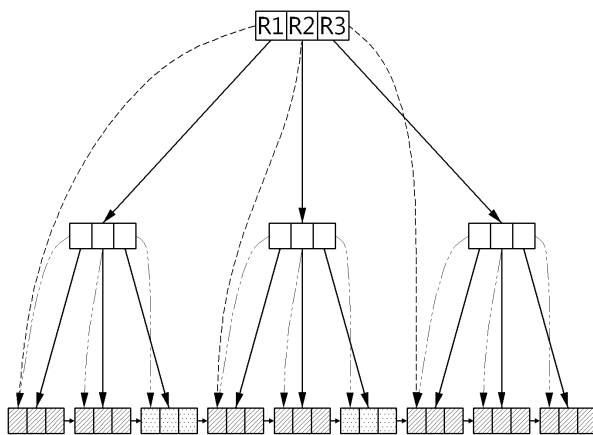
[0051] 그러므로, 본 발명의 범위는 설명된 실시예에 국한되어 정해져서는 아니 되며, 후술하는 특허청구범위뿐 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

도면

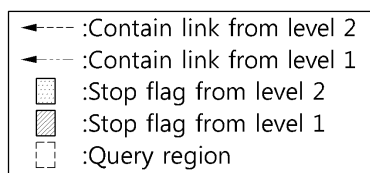
도면1



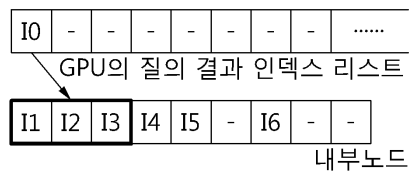
(a)



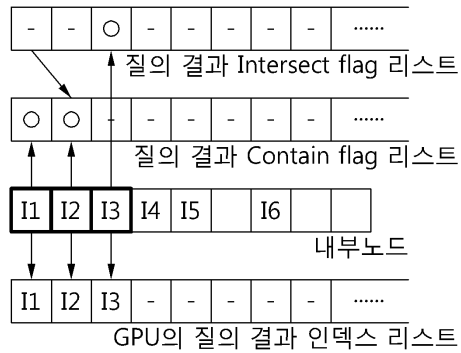
(b)



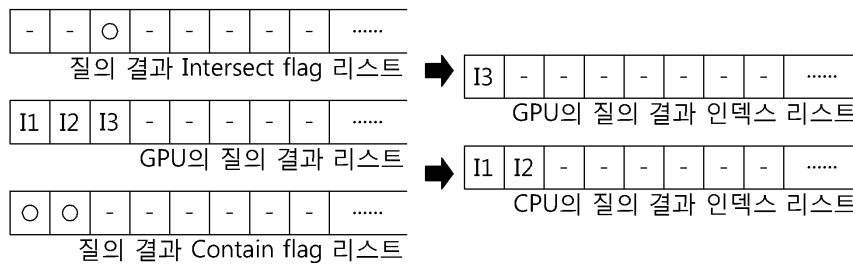
도면2



(a)

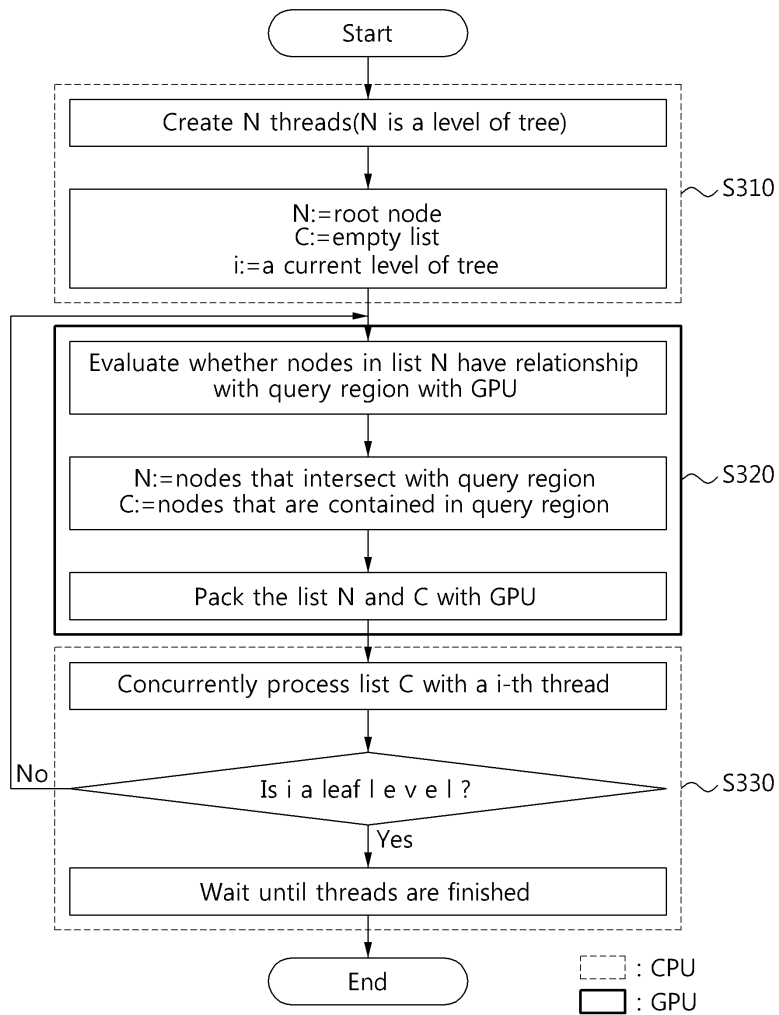


(b)

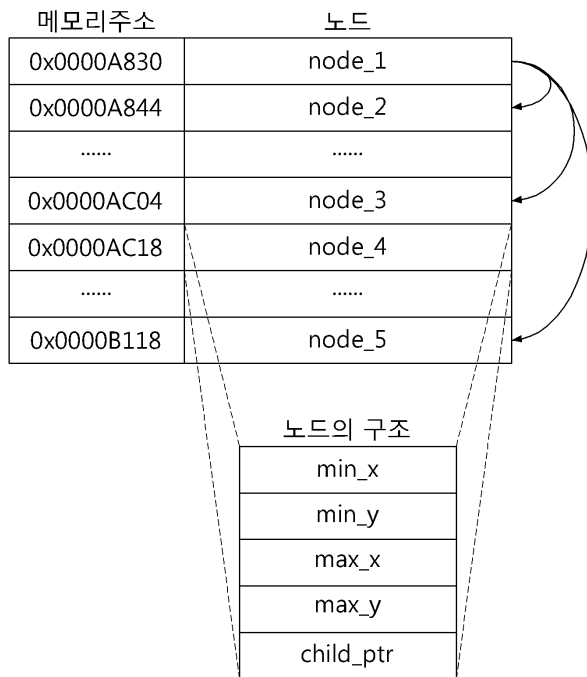


(c)

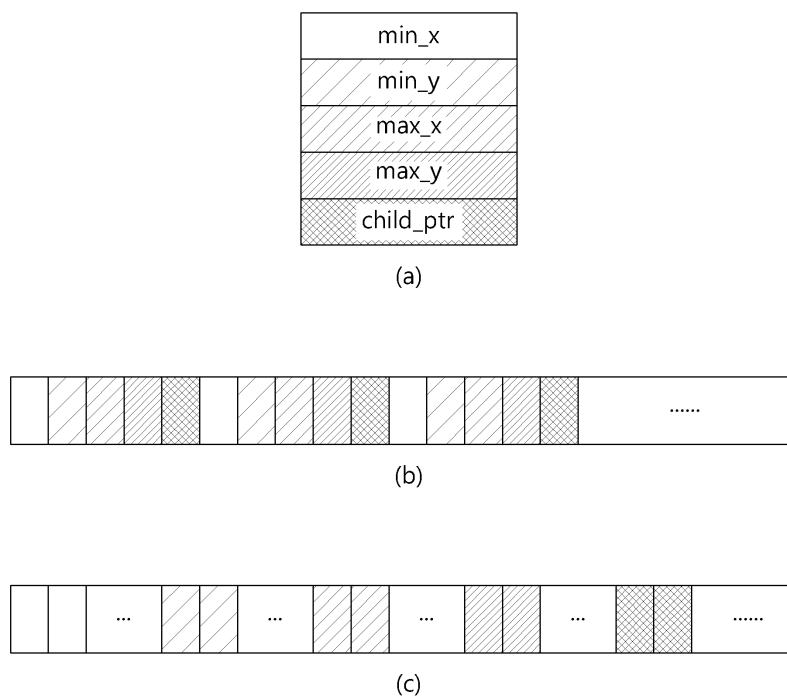
도면3



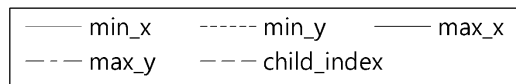
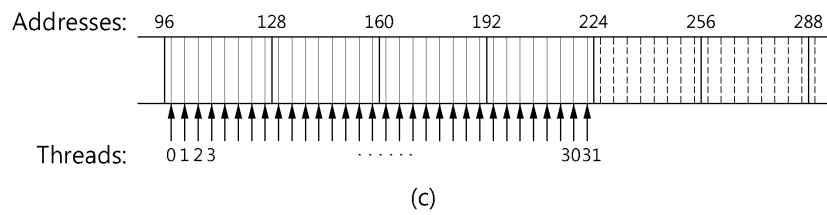
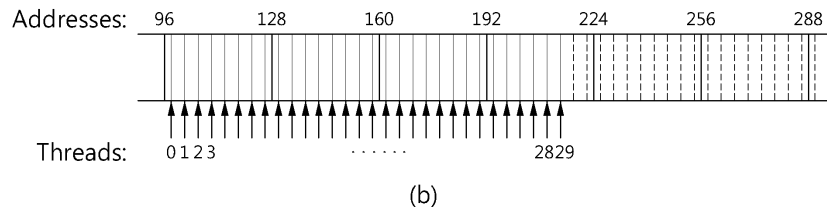
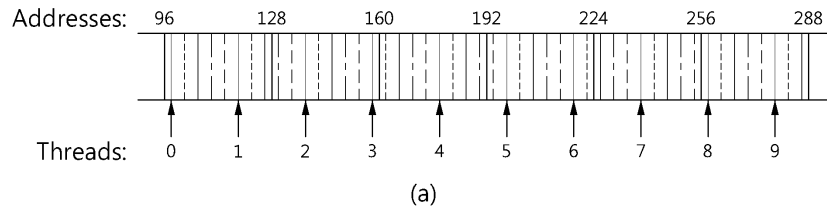
도면4



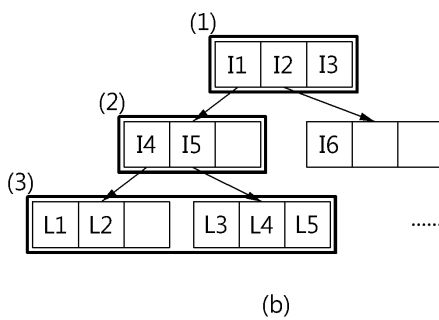
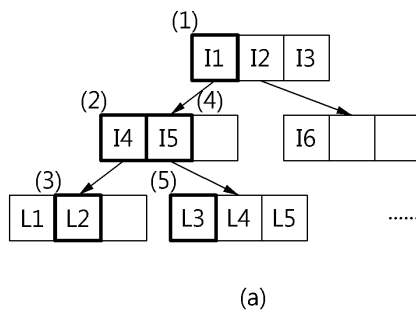
도면5



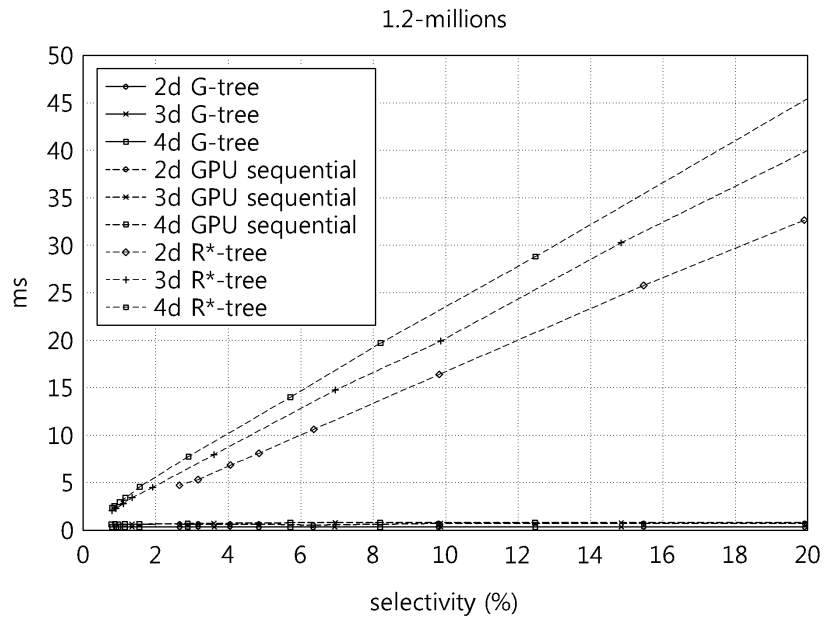
도면6



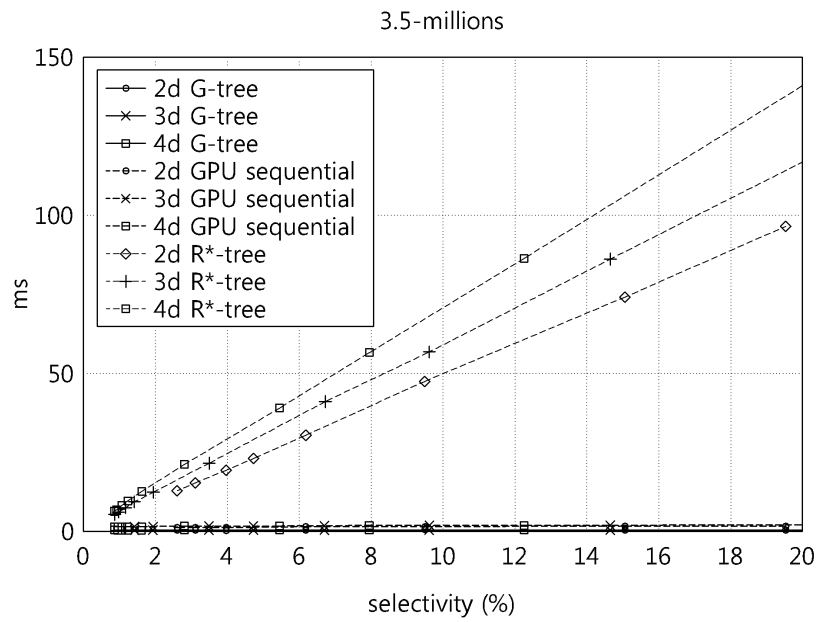
도면7



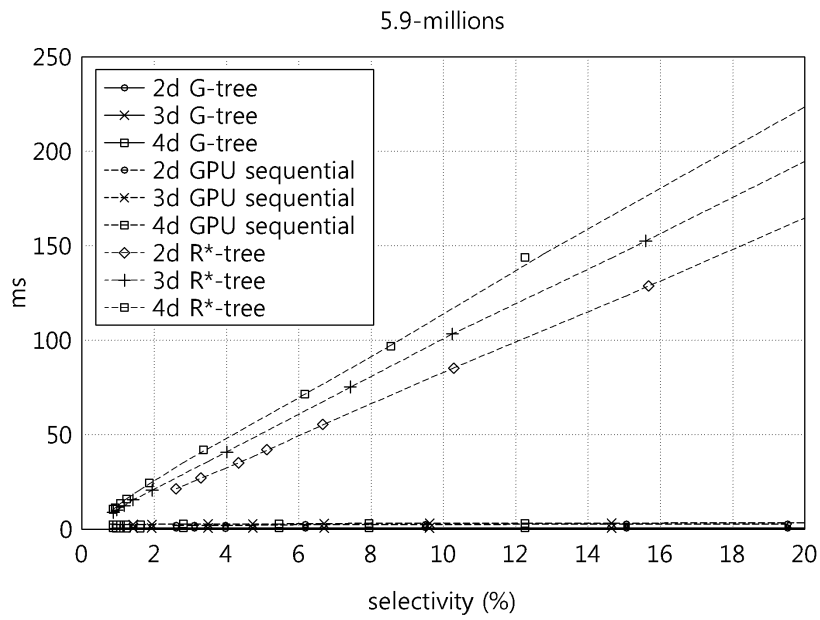
도면8



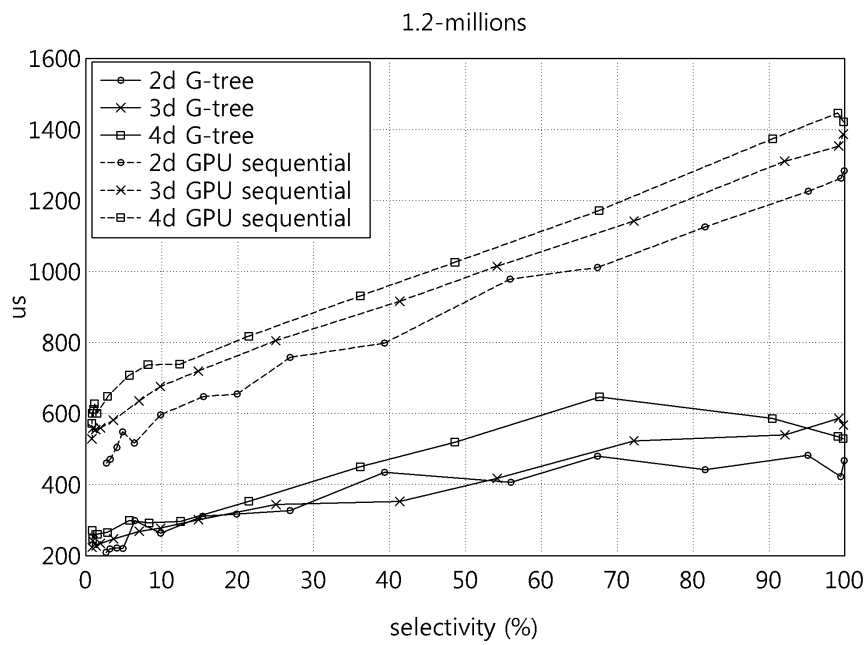
도면9



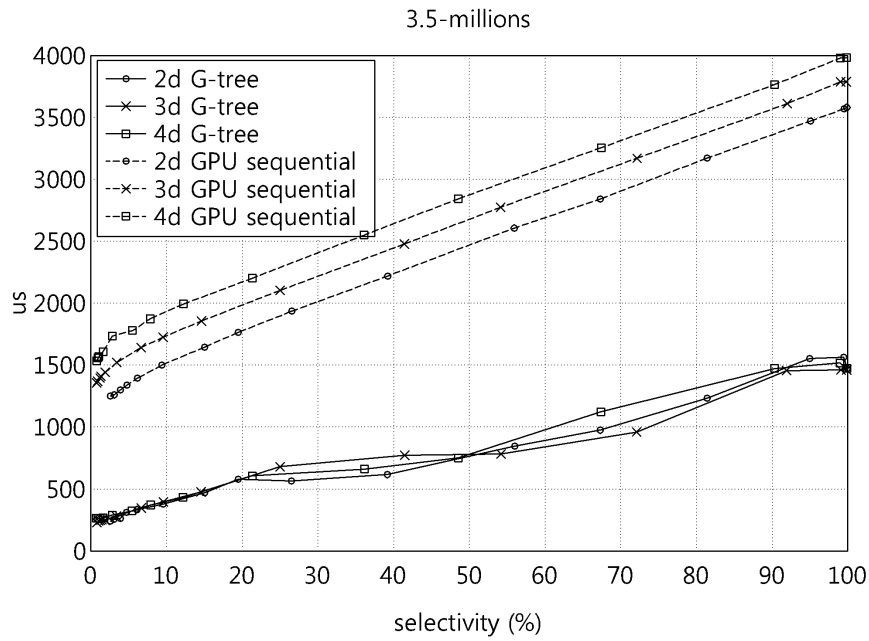
도면10



도면11



도면12



도면13

