



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0092396
(43) 공개일자 2018년08월20일

(51) 국제특허분류(Int. Cl.)

G06F 17/11 (2006.01)

(52) CPC특허분류

G06F 17/11 (2013.01)

(21) 출원번호 10-2017-0018007

(22) 출원일자 2017년02월09일

심사청구일자 2017년02월09일

(71) 출원인

울산과학기술원

울산광역시 울주군 언양읍 유니스트길 50

(72) 발명자

정원기

울산광역시 울주군 언양읍 유니스트길 50 울산과학기술대학교

홍수민

울산광역시 울주군 언양읍 유니스트길 50 울산과학기술대학교

(74) 대리인

제일특허법인(유)

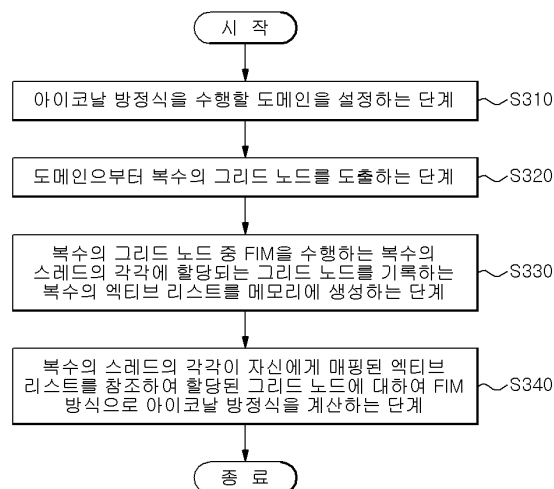
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 락프리 기반 고속 반복 계산 방법 및 장치

(57) 요약

본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 방법은 아이코날 방정식을 수행할 도메인을 설정하는 단계, 도메인으로부터 복수의 그리드 노드(grid node)를 도출하는 단계, FIM(fast iterative method)을 수행하는 복수의 스레드(thread)의 각각에 할당되는 그리드 노드를 기록하는 복수의 액티브 리스트(active list)를 메모리에 생성하는 단계 및 복수의 스레드의 각각이 매핑된 액티브 리스트를 참조하여 할당된 그리드 노드에 대하여 FIM 방식으로 아이코날 방정식을 계산하는 단계를 포함한다.

대표도 - 도3



이 발명을 지원한 국가연구개발사업	
과제고유번호	R0190-15-2012
부처명	교육과학기술부
연구관리전문기관	정보통신기술진흥센터
연구사업명	SW컴퓨팅산업원천기술개발사업
연구과제명	빅데이터 처리 고도화 핵심 기술개발 사업 총괄 및 고성능 컴퓨팅 기술을 활용한 성능 가
속화 기술 개발	
기 여 율	1/1
주관기관	울산과학기술원
연구기간	2016.06.01 ~ 2017.05.31

명세서

청구범위

청구항 1

컴퓨터 상에서 수행되는 락프리 기반 고속 반복 계산 방법으로서,

아이코날 방정식을 수행할 도메인을 설정하는 단계;

상기 도메인으로부터 복수의 그리드 노드(grid node)를 도출하는 단계; 및

상기 복수의 그리드 노드 중 FIM(fast iterative method)을 수행하는 복수의 스레드(thread)의 각각에 할당되는 그리드 노드를 기록하는 복수의 액티브 리스트(active list)를 메모리에 생성하는 단계를 포함하되, 상기 복수의 스레드는 상기 복수의 액티브 리스트와 하나씩 매핑되어 있고,

상기 방법은,

상기 복수의 스레드의 각각이 자신에게 매핑된 액티브 리스트를 참조하여 할당된 그리드 노드에 대하여 FIM 방식으로 아이코날 방정식을 계산하는 단계를 더 포함하는

락프리 기반 고속 반복 계산 방법.

청구항 2

제1항에 있어서,

상기 방법은,

상기 복수의 스레드 중 제1 스레드와 상기 복수의 스레드 중 제2 스레드를 매핑하는 단계를 더 포함하고,

상기 아이코날 방정식을 계산하는 단계는,

상기 제1 스레드에 매핑된 제1 액티브 리스트에 할당된 그리드 노드의 개수와 상기 제2 스레드에 매핑된 제2 액티브 리스트에 할당된 그리드 노드의 개수가 1개 이하의 차이가 나도록 그리드 노드를 분배하는 단계를 포함하는

락프리 기반 고속 반복 계산 방법.

청구항 3

제2항에 있어서,

상기 매핑하는 단계는,

상기 복수의 스레드의 각각에 인덱스가 1부터 n의 인덱스가 있다고 할 때,

[수학식 1]

$$j = (i + \text{offset}) \bmod n$$
 (i는 1부터 n 사이의 홀수 번째 인덱스, offset은 소정의 홀수, n은 상기 복수의 스레드의 개수)

의 상기 i번째 인덱스의 스레드와 상기 j번째 인덱스의 스레드를 매핑시키는

락프리 기반 고속 반복 계산 방법.

청구항 4

제1항에 있어서,

상기 아이코날 방정식을 계산하는 단계는,

상기 복수의 그리드 노드 중 제1 그리드 노드가 상기 복수의 액티브 리스트 중 어느 하나의 액티브 리스트에 할

락되거나 제외되는 경우, 상기 제1 그리드 노드의 할당 여부를 메모리에 기록하는 단계를 포함하고,

상기 제1 그리드 노드가 상기 메모리에 기록된 할당 여부에 따라 상기 어느 하나의 액티브 리스트에 할당되어 있는 것으로 기록되어 있는 경우, 상기 제1 그리드 노드는 상기 할당되어 있는 액티브 리스트를 제외한 다른 액티브 리스트에 할당되지 않는

락프리 기반 고속 반복 계산 방법.

청구항 5

락프리 기반 고속 반복 계산 장치에 있어서,

아이코날 방정식을 수행할 도메인을 설정하는 도메인 설정부;

상기 도메인으로부터 복수의 그리드 노드(grid node)를 도출하는 그리드 노드 도출부; 및

상기 복수의 그리드 노드 중 FIM(fast iterative method)을 수행하는 복수의 스레드(thread)의 각각에 할당되는 그리드 노드를 기록하는 복수의 액티브 리스트(active list)를 메모리에 생성하는 액티브 리스트 생성부를 포함 하되, 상기 복수의 스레드는 상기 복수의 액티브 리스트와 하나씩 매핑되어 있고,

상기 장치는,

상기 복수의 스레드의 각각이 자신에게 매핑된 액티브 리스트를 참조하여 할당된 그리드 노드에 대하여 FIM 방식으로 아이코날 방정식을 계산하게 하는 FIM 계산부를 더 포함하는

락프리 기반 고속 반복 계산 장치.

청구항 6

제1항 내지 제4항의 방법을 프로세서가 수행하도록 하는 컴퓨터 판독 가능 기록매체에 저장된 프로그램.

청구항 7

제1항 내지 제4항의 방법을 프로세서가 수행하게 하는 명령어를 포함하는 프로그램이 기록된 컴퓨터 판독 가능 기록매체.

발명의 설명

기술 분야

[0001] 본 발명은 락프리 기반 고속 반복 계산 방법 및 장치에 관한 것으로서, 보다 자세하게는 아이코날 방정식(eikonal equation)을 컴퓨터 상에서 수행함에 있어서 복수의 스레드가 각각의 액티브 리스트(active list)를 관리하게 하여 FIM(fast iterative method)을 수행하게 하는 락프리 기반 고속 반복 계산 방법 및 장치에 관한 것이다.

배경 기술

[0002] 아이코날 방정식은 두 점간 광학적 최단 거리 또는 기하학적 최단 거리를 도출할 수 있도록 하는 방정식이다.

[0003] 아이코날 방정식을 컴퓨터 상에서 수행하기 위해 가장 널리 쓰이던 FMM(fast marching method)은 아이코날 방정식을 풀기 위하여 그리드 노드(grid node) 간의 종속성에 따라 계산을 수행함으로써, 중복 계산의 가능성을 원천 봉쇄하였지만 종속성에 따라 그리드 노드를 순차적으로 계산하여야 한다는 특성상 복수의 스레드가 병렬적으로 계산을 수행하도록 하기 어려운 문제가 있었다.

[0004] 이를 해결하기 위하여 FIM(fast iterative method)은 GPU와 같은 SIMD (Single Instruction Multiple Data) 하드웨어를 사용하여 액티브 리스트(Active list)에 속한 그리드 노드에 대하여 아이코날 방정식을 병렬적으로 계산할 수 있는 방법을 제시하였다. (Won-Ki Jeong and Ross T. Whitaker. A fast iterative method for eikonal equations. SIAM J. Sci. Comput. Vol.30 No.5, pages 2512-2534, 2008.)

[0005] 그러나 FIM 모델에서는 글로벌 메모리(global memory)에 저장된 하나의 액티브 리스트를 복수의 스레드가 참조하여 아이코날 방정식의 연산을 수행하였기 때문에 race condition이 발생할 수 있다는 문제가 있다.

[0006] 이를 해결하기 위하여, 락 동기화(lock synchronization)를 사용하여 race condition을 해결할 수는 있지만, 락 동기화를 사용하는 경우 복수의 스레드가 병렬적으로 계산을 수행하는 FIM 모델에서는 치명적인 성능 저하를 가져올 수 있다는 문제가 있다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 미국등록특허 제8,762,442호(2011.02.24 공고)

발명의 내용

해결하려는 과제

[0008] 본 발명의 실시예에서 해결하고자 하는 과제는 FIM에 따라 병렬적으로 계산을 수행하는 복수의 스레드가 락 동기화 없이 액티브 리스트를 참조하여도 race condition이 발생하지 않도록 하는 기술을 제공하는 것이다.

[0009] 다만, 본 발명의 실시예가 이루고자 하는 기술적 과제는 이상에서 언급한 과제로 제한되지 않으며, 이하에서 설명할 내용으로부터 통상의 기술자에게 자명한 범위 내에서 다양한 기술적 과제가 도출될 수 있다.

과제의 해결 수단

[0010] 본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 방법은 아이코날 방정식을 수행할 도메인을 설정하는 단계, 상기 도메인으로부터 복수의 그리드 노드(grid node)를 도출하는 단계 및 상기 복수의 그리드 노드 중 FIM(fast iterative method)을 수행하는 복수의 스레드(thread)의 각각에 할당되는 그리드 노드를 기록하는 복수의 액티브 리스트(active list)를 메모리에 생성하는 단계를 포함하되, 상기 복수의 스레드는 상기 복수의 액티브 리스트와 하나씩 매핑되어 있고, 상기 방법은 상기 복수의 스레드의 각각이 자신에게 매핑된 액티브 리스트를 참조하여 할당된 그리드 노드에 대하여 FIM 방식으로 아이코날 방정식을 계산하는 단계를 더 포함한다.

[0011] 이때 상기 방법은 상기 복수의 스레드 중 제1 스레드와 상기 복수의 스레드 중 제2 스레드를 매핑하는 단계를 더 포함하고, 상기 아이코날 방정식을 계산하는 단계는 상기 제1 스레드에 매핑된 제1 액티브 리스트에 할당된 그리드 노드의 개수와 상기 제2 스레드에 매핑된 제2 액티브 리스트에 할당된 그리드 노드의 개수가 1개 이하의 차이가 나도록 그리드 노드를 분배하는 단계를 포함할 수 있다.

[0012] 또한 상기 매핑하는 단계는 상기 복수의 스레드의 각각에 인덱스가 1부터 n의 인덱스가 있다고 할 때, $[수학식 1] \quad j = (i + offset) \bmod n$ (i는 1부터 n 사이의 홀수 번째 인덱스, offset은 소정의 홀수, n은 상기 복수의 스레드의 개수)의 상기 i번째 인덱스의 스레드와 상기 j번째 인덱스의 스레드를 매핑시킬 수 있다.

[0013] 더불어 상기 아이코날 방정식을 계산하는 단계는 상기 복수의 그리드 노드 중 제1 그리드 노드가 상기 복수의 액티브 리스트 중 어느 하나의 액티브 리스트에 할당되거나 제외되는 경우, 상기 제1 그리드 노드의 할당 여부를 메모리에 기록하는 단계를 포함하고, 상기 제1 그리드 노드가 상기 메모리에 기록된 할당 여부에 따라 상기 어느 하나의 액티브 리스트에 할당되어 있는 것으로 기록되어 있는 경우, 상기 제1 그리드 노드는 상기 할당되어 있는 액티브 리스트를 제외한 다른 액티브 리스트에 할당되지 않을 수 있다.

[0014] 본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 장치는 아이코날 방정식을 수행할 도메인을 설정하는 도메인 설정부, 상기 도메인으로부터 복수의 그리드 노드(grid node)를 도출하는 그리드 노드 도출부 및 상기 복수의 그리드 노드 중 FIM(fast iterative method)을 수행하는 복수의 스레드(thread)의 각각에 할당되는 그리드 노드를 기록하는 복수의 액티브 리스트(active list)를 메모리에 생성하는 액티브 리스트 생성부를 포함하되, 상기 복수의 스레드는 상기 복수의 액티브 리스트와 하나씩 매핑되어 있고, 상기 장치는 상기 복수의 스레드의 각각이 자신에게 매핑된 액티브 리스트를 참조하여 할당된 그리드 노드에 대하여 FIM 방식으로 아이코날 방정식을 계산하게 하는 FIM 계산부를 더 포함한다.

발명의 효과

[0015] 본 발명의 실시예에 따르면, 복수의 스레드들이 독립적으로 액티브 리스트를 관리함으로써 락 동기화 없이 FIM

을 수행하여도 race condition이 발생하지 않는다.

[0016] 또한 복수의 액티브 리스트간에 동일한 그리드 노드가 중복되어 추가되지 않도록 하여 복수의 스레드가 각각의 액티브 리스트를 효율적이면서 안전하게 접근 및 관리할 수 있도록 한다.

도면의 간단한 설명

[0017] 도 1a 내지 도 1c는 기존 FIM의 계산 과정을 설명하기 위한 예시도이다.

도 2는 본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 장치의 기능 블록도이다.

도 3은 본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 방법의 프로세스를 도시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0018] 본 발명의 목적과 기술적 구성 및 그에 따른 작용 효과에 관한 자세한 사항은 본 발명의 명세서에 첨부된 도면에 의거한 이하의 상세한 설명에 의해 보다 명확하게 이해될 것이다. 첨부된 도면을 참조하여 본 발명에 따른 실시예를 상세하게 설명한다.

[0019] 본 명세서에서 개시되는 실시예들은 본 발명의 범위를 한정하는 것으로 해석되거나 이용되지 않아야 할 것이다. 이 분야의 통상의 기술자에게 본 명세서의 실시예를 포함한 설명은 다양한 응용을 갖는다는 것이 당연하다. 따라서, 본 발명의 상세한 설명에 기재된 임의의 실시예들은 본 발명을 보다 잘 설명하기 위한 예시적인 것이며 본 발명의 범위가 실시예들로 한정되는 것을 의도하지 않는다.

[0020] 도면에 표시되고 아래에 설명되는 기능 블록들은 가능한 구현의 예들일 뿐이다. 다른 구현들에서는 상세한 설명의 사상 및 범위를 벗어나지 않는 범위에서 다른 기능 블록들이 사용될 수 있다. 또한, 본 발명의 하나 이상의 기능 블록이 개별 블록들로 표시되지만, 본 발명의 기능 블록들 중 하나 이상은 동일 기능을 실행하는 다양한 하드웨어 및 소프트웨어 구성들의 조합일 수 있다.

[0021] 또한, 어떤 구성 요소들을 포함한다는 표현은 개방형의 표현으로서 해당 구성 요소들이 존재하는 것을 단순히 지칭할 뿐이며, 추가적인 구성 요소들을 배제하는 것으로 이해되어서는 안 된다.

[0022] 나아가 어떤 구성 요소가 다른 구성 요소에 연결되어 있다거나 접속되어 있다고 언급될 때에는, 그 다른 구성 요소에 직접적으로 연결 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 한다.

[0023] 또한 '제1, 제2' 등과 같은 표현은 복수의 구성들을 구분하기 위한 용도로만 사용된 표현으로써, 구성들 사이의 순서나 기타 특징들을 한정하지 않는다.

[0024] 이하에서는 도면들을 참조하여 본 발명의 실시예들에 대해 설명하도록 한다.

[0025] 아이코날 방정식은 두 점간 광학적 최단 거리 또는 기하학적 최단 거리를 도출할 수 있도록 하는 방정식으로 아래 수학적 1과 같다.

수학적 1

$$[0026] \quad H(\mathbf{x}, \nabla \phi) = |\nabla \phi(\mathbf{x})|^2 - \frac{1}{f^2(\mathbf{x})} = 0 \quad \forall \mathbf{x} \in \Omega,$$

[0027] 이때 Ω 는 계산을 수행할 대상이 되는 도메인을 의미하며, $\emptyset$ 는 도메인 상에서 특정 지점까지의 거리를 계산할 기준이 되는 지점이고, $\mathbf{x}$ 는 도메인 상에서의 특정 지점으로, $\nabla \emptyset(\mathbf{x})$ 는 지점 $\emptyset$ 로부터 지점 $\mathbf{x}$ 까지의 최단 거리를 의미하며, $f(\mathbf{x})$ 는 Ω 에서의 속도 함수이고, $H(\mathbf{x}, \nabla \emptyset(\mathbf{x}))$ 를 0으로 두었을 때 위 방정식의 해 $\nabla \emptyset(\mathbf{x})$ 를 구하여 지점 $\emptyset$ 로부터 지점 $\mathbf{x}$ 까지의 최단 거리를 구할 수 있다.

[0028] 이러한 아이코날 방정식을 컴퓨터 상에서 수행하기 위해서는 연속적으로 이루어진 도메인을 불연속적으로 나누어야 하기 때문에, 예를 들면 기존에 사용되는 Godunov upwind difference scheme을 통해 도메인을 불연속적으

로 이루어진 그리드 노드(grid node)로 분할할 수 있다.

[0029] Godunov upwind difference scheme을 통해 도메인이 불연속적인 그리드 노드로 분할된 경우 도메인이 3차원이라면 아이코날 방정식은 아래 수학식 2와 같이 나타낼 수 있다.

수학식 2

[0030]

$$g(\mathbf{x}) = \left[\frac{(U(\mathbf{x}) - U(\mathbf{x})^{x\min})^+}{h_x} \right]^2 + \left[\frac{(U(\mathbf{x}) - U(\mathbf{x})^{y\min})^+}{h_y} \right]^2 + \left[\frac{(U(\mathbf{x}) - U(\mathbf{x})^{z\min})^+}{h_z} \right]^2 - \frac{1}{f(\mathbf{x})^2},$$

[0031] 이때 $U(\mathbf{x})$ 는 도메인 상에서 거리를 계산할 기준이 되는 지점 $\emptyset$ 로부터 그리드 노드 $\mathbf{x}=(i, j, k)$ 까지의 최단 거리를 의미하며, $U(\mathbf{x})^{p\min}$ 은 p 축($p \in \{x, y, z\}$)상에서 $U(\mathbf{x})$ 와 인접한 두 개의 그리드 노드 중 더 짧은 거리에 있는 그리드 노드의 $U(\mathbf{x})$ 값을 의미한다. 또한 h_p 는 p 축에 있어서 그리드 노드 간의 공간이며, $f(\mathbf{x})$ 는 속도 함수이고, $(n)^+ = \max(n, 0)$ 을 의미한다.

[0032] FIM(fast iterative method)은 아이코날 방정식을 컴퓨터 상에서 병렬적으로 계산하기 위해 액티브 리스트(active list)를 생성한다. 액티브 리스트에는 아이코날 방정식을 계산하고 있거나 계산해야 할 그리드 노드의 정보를 글로벌 메모리(global memory)에 저장하여, 복수의 스레드들이 액티브 리스트를 참조하여 액티브 리스트에 포함된 그리드 노드에 대해 아이코날 방정식을 계산함으로써 병렬적인 연산이 가능하도록 한다.

[0033] 도 1a 내지 도 1c는 기존 FIM의 계산 과정을 설명하기 위한 예시도이다.

[0034] 도 1a에 도시된 바와 같이, FIM 모델은 도메인 상에서 거리를 계산할 기준이 되는 씨드 노드(seed node)(101)와 인접한 그리드 노드(102, 103)를 액티브 리스트에 포함시켜 첫번째 연산을 통해 액티브 리스트에 포함된 그리드 노드의 아이코날 방정식을 계산하고, 도 1b와 같이 첫번째 연산이 끝나면 첫번째 연산에서 계산된 그리드 노드(102, 103)와 인접한 그리드 노드(104, 105, 106)를 액티브 리스트에 추가하여 액티브 리스트에 있는 그리드 노드에 대하여 아이코날 방정식을 계산하도록 한다. 이때 아이코날 방정식의 계산을 통해 특정 값에 수렴하게 되는 그리드 노드는 액티브 리스트에서 제외시키며, 이 과정을 모든 그리드 노드가 수렴하게 될 때까지 반복하여 아이코날 방정식의 계산을 완료한다.

[0035] 아래 표 1은 아이코날 방정식을 컴퓨터 상에서 계산하기 위한 기존 FIM의 수도 코드를 나타낸 것이다.

표 1

Algorithm 1: FIM

Input: Grid Ω , Solution U , Active list L

/ Initialization */*

```

1 forall  $x \in \Omega$  do
2   if  $x$  is a source node then
3      $U(x) \leftarrow 0$ 
4     add  $x$  to  $L$ 
5   else
6      $U(x) \leftarrow \infty$ 
7   /* Compute new solutions for  $L$  */
8   while  $L$  is not empty do
9     forall  $x \in L$  do
10       $p \leftarrow U(x)$ 
11       $q \leftarrow \text{solution of } g(x) = 0$ 
12      /* If not converged */
13      if  $p > q$  then
14         $U(x) \leftarrow q$ 
15      /* If converged */
16      else
17        /* Check adjacent neighbor nodes for reactivation */
18        forall  $x_{nb}$  adjacent to  $x$  do
19          if  $U(x_{nb}) > U(x)$  and  $x_{nb} \notin L$  then
20             $p \leftarrow U(x_{nb})$ 
21             $q \leftarrow \text{solution of } g(x_{nb}) = 0$ 
22            if  $p > q$  then
23               $U(x_{nb}) \leftarrow q$ 
24              add  $x_{nb}$  to  $L$ 
25      remove  $x$  from  $L$ 

```

[0036]

[0037]

표 1에 표기된 source node는 도메인 Ω 에서 특정 그리드 노드까지의 거리를 계산할 기준이 되는 씨드 노드(seed node)를 의미한다. 따라서 씨드 노드 자체의 $U(x)$ 는 0이며, 그 외의 그리드 노드의 아이코날 방정식의 해를 ∞ 로 초기 설정한 후, 씨드 노드로부터 인접한 그리드 노드를 액티브 리스트에 추가하면서 액티브 리스트에 포함된 그리드 노드에 대한 아이코날 방정식을 수행한다. 이때 이전에 계산한 아이코날 방정식의 해인 p 와 가장 최근 계산한 아이코날 방정식의 해 q 를 비교하여 특정 그리드 노드에서 $q < p$ 인 경우 해당 그리드 노드에서의 아이코날 방정식의 해가 수렴한 것으로 간주하여 액티브 리스트에서 해당 그리드 노드를 제외하는 방식으로 모든 그리드 노드가 수렴할 때까지 반복하여 아이코날 방정식을 계산한다.

[0038]

한편, 기존 FIM 모델에서는 글로벌 메모리에 저장된 하나의 액티브 리스트를 복수의 스레드가 참조하여 아이코날 방정식의 연산을 수행하기 때문에 race condition이 발생할 수 있다는 문제가 있다. 예를 들어, 표 1의 19, 20 라인의 수도 코드를 참조하면, 그리드 노드 x_{nb} 에 대하여 아이코날 방정식의 해를 업데이트 하는 과정과 그리드 노드 x_{nb} 를 액티브 리스트에 추가하는 과정을 각각 다른 스레드가 할당받아 수행할 수 있기 때문에 이때 race condition이 발생할 수 있다.

[0039]

이에, 본 발명의 실시예는 병렬적으로 계산을 수행하는 복수의 스레드가 락 동기화(lock synchronization) 없이 액티브 리스트를 참조하여도 race condition이 발생하는 않도록 하는 기술로서 아래 도 2 내지 도 3과 함께 설명하도록 한다.

[0040]

도 2는 본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 장치의 기능 블록도이다.

[0041]

도 2에 도시된 바와 같이, 본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 장치는 도메인 설정부(210), 그리드 노드 도출부(220), 액티브 리스트 생성부(230) 및 FIM 계산부(240)를 포함한다.

[0042]

도메인 설정부(210)는 아이코날 방정식을 계산할 도메인을 설정한다. 예를 들어, 특정 지역에서 지진파가 전달

되는 거리 또는 속도를 알기 위해 특정 지역의 지리학적 정보를 도메인으로 설정할 수 있다.

[0043] 그리드 노드 도출부(220)는 도메인으로부터 복수의 그리드 노드를 도출한다. 예를 들어, 그리드 노드 도출부(220)는 Godunov upwind difference scheme을 통해 도메인을 불연속적으로 이루어진 그리드 노드로 분할할 수 있다.

[0044] 액티브 리스트 생성부(230)는 FIM을 수행하는 복수의 스레드의 각각에 할당되는 그리드 노드를 기록하는 복수의 액티브 리스트를 메모리에 생성할 수 있고, 예를 들어 메모리는 글로벌 메모리를 포함할 수 있다. 이때 복수의 스레드는 각 스레드만이 접근할 수 있는 고유한 액티브 리스트와 매핑될 수 있다.

[0045] FIM 계산부(240)는 복수의 스레드의 각각이 자신에게 매핑된 액티브 리스트를 참조하여 할당된 그리드 노드에 대하여 FIM 방식으로 아이코날 방정식을 계산하게 한다.

[0046] 이때 그룹 순서 기반 고속 반복 계산 장치(200)의 각 구성이 동작하는 구체적인 과정을 도 3과 함께 설명하기로 한다.

[0047] 한편 상술한 실시예가 포함하는 도메인 설정부(210), 그리드 노드 도출부(220), 액티브 리스트 생성부(230) 및 FIM 계산부(240)는 이들의 기능을 수행하도록 프로그램된 명령어를 포함하는 메모리, 및 이들 명령어를 수행하는 마이크로프로세서를 포함하는 연산 장치에 의해 구현될 수 있다.

[0048] 도 3은 본 발명의 일 실시예에 따른 락프리 기반 고속 반복 계산 방법의 프로세스를 도시하는 흐름도이다. 도 3에 따른 락프리 기반 고속 반복 계산 방법의 각 단계는 도 2를 통해 설명된 락프리 기반 고속 반복 계산 장치에 의해 수행될 수 있으며, 각 단계를 설명하면 다음과 같다.

[0049] 우선, 도메인 설정부(210)는 아이코날 방정식을 계산할 도메인을 설정하고(S310), 그리드 노드 도출부(220)는 도메인으로부터 복수의 그리드 노드를 도출한다(S320).

[0050] 다음으로, 액티브 리스트 생성부(230)는 FIM을 수행하는 복수의 스레드의 각각에 할당되는 그리드 노드를 기록하는 복수의 액티브 리스트를 메모리에 생성한다(S330). 즉, 기존의 FIM 모델에서는 글로벌 메모리에 저장된 하나의 액티브 리스트를 복수의 스레드가 참조하지만, 본 발명의 실시예에 따르면 복수의 스레드는 각 스레드가 계산할 그리드 노드를 기록하는 특정 액티브 리스트와 매핑될 수 있다.

[0051] 이후, FIM 계산부(240)는 복수의 스레드의 각각이 매핑된 액티브 리스트를 참조하여 할당된 그리드 노드에 대하여 FIM 방식으로 아이코날 방정식을 계산한다(S340).

[0052] 이때 각 스레드가 FIM 방식에 따라 아이코날 방정식을 계산하는 과정에서 각 스레드에 매핑된 액티브 리스트가 할당받은 그리드 노드의 개수가 달라질 수 있다. 이러한 경우 액티브 리스트에 그리드 노드가 상대적으로 많이 할당된 스레드에 비하여, 액티브 리스트에 그리드 노드가 상대적으로 적게 할당된 스레드는 연산이 빠르게 완료되기 때문에, 다른 스레드의 연산 과정이 완료될 때까지 대기하여 효율이 저하될 수 있다.

[0053] 이러한 점을 개선하기 위해, S340 단계에서는 FIM 연산에서 액티브 리스트를 업데이트하는 과정이 일어나기 전 또는 후에, 각 액티브 리스트에 포함된 그리드 노드의 개수를 비교하여 동기화할 수 있다.

[0054] 예를 들어, 복수의 스레드 중 제1 스레드에 매핑된 제1 액티브 리스트에 할당된 그리드 노드의 개수와 복수의 스레드 중 제2 스레드에 매핑된 제2 액티브 리스트에 할당된 그리드 노드의 개수가 동일하거나 1개의 차이가 나도록 그리드 노드를 분배할 수 있다.

[0055] 이때 제1 스레드와 상기 제2 스레드는 각각 매핑되어 있을 수 있다. 가령, 복수의 스레드의 각각에 인덱스가 1부터 n의 인덱스가 있다고 할 때, 아래 수학적 식 3과 같이 i번째 인덱스의 스레드와 j번째 인덱스의 스레드를 매핑시켜, 복수의 스레드끼리 상호의 액티브 리스트에 포함된 그리드 노드의 개수를 참조하여, 각각의 액티브 리스트에 할당된 그리드 노드의 개수가 동일하거나 적어도 1개의 차이가 나도록 그리드 노드를 분배할 수 있다.

수학적 식 3

$$j = (i + \text{offset}) \bmod n$$

[0056]

[0057] (i는 1부터 n 사이의 홀수 번째 인덱스, offset은 소정의 홀수, n은 복수의 스레드의 개수)

- [0058] 한편, 복수의 액티브 리스트의 각각에 할당된 최초의 그리드 노드는 서로 중복되어 할당되어 있지 않다고 하더라도, FIM 계산 과정에서 동일한 그리드 노드가 서로 다른 액티브 리스트에 추가될 가능성이 있다. 예를 들어, 서로 다른 액티브 리스트에 특정 그리드 노드가 FIM 계산 과정에서 서로 다른 액티브 리스트에서 수렴한 것으로 판단된 그리드 노드와 인접하고 있을 경우 그리드 노드의 중복 추가가 발생할 수 있다.
- [0059] 이러한 점을 개선하기 위해, S340 단계에서는 특정 그리드 노드가 어느 하나의 액티브 리스트에 포함되어 있는지를 체크하기 위해, 그리드 노드에 표시기(indicator)를 붙일 수 있다.
- [0060] 예를 들어, 특정 그리드 노드가 복수의 액티브 리스트 중 어느 하나의 액티브 리스트에 할당되거나 제외되는 경우, 아래 표 2와 같이 해당 그리드 노드의 할당 여부를 글로벌 메모리에 기록할 수 있다.

표 2

Listing 1. Lock-free code using a fetch-and-modify atomic operator

```
// idx : node index
// F[idx] : true if idx is in the list, false otherwise

if(__sync_lock_test_and_set(&(F[idx]), 1) == 0)
{
    // insert idx into active list
    . . . . .
}
```

- [0061]
- [0062] 따라서 특정 그리드 노드가 어느 하나의 액티브 리스트에 할당되게 되면, 글로벌 메모리에 어느 하나의 액티브 리스트에 할당되어 있다는 표시기가 붙게 되므로, 그리드 노드는 현재 할당되어 있는 액티브 리스트를 제외한 다른 액티브 리스트에 할당되지 않게 되어 서로 다른 액티브 리스트에 중복되어 추가되는 문제가 발생하지 않는다.
- [0063] 상술한 실시예에 따른 단계를 수도코드로 나타내면 아래 표 3와 같다.

표 3

Algorithm 2: LOCK-FREE PARALLEL FIM	
Input: Grid Ω , Solution U , Active list L	
<i>/* Initialization */</i>	
1 $n \leftarrow$ number of threads	
2 $N = \{0, 1, 2, \dots, n-1\}$	
3 Initialize U and L as line 1 to 6 in Algorithm 1	
4 Split L into disjoint sublists L_i for all $i \in N$ so that $L = \cup_{i \in N} L_i$	
5 $F =$ flag array, initialized 0	
<i>/* Flag array for node state */</i>	
6 foreach $i \in N$ do in parallel	
7 forall $x \in L_i$ do	
8 $F(x) \leftarrow 1$	
<i>/* Parallel update of sublists L_i */</i>	
9 while L_i is not empty for some $i \in N$ do	
<i>/* Load balancing */</i>	
10 offset $\leftarrow$ a randomly selected odd number	
11 foreach $i \in N$ do in parallel	
12 if i is odd number then	
13 $j \leftarrow (i + \text{offset}) \% n$	
14 Make the size of L_i and L_j equal by stealing nodes from the larger list	
15 Barrier synchronization	
<i>/* Solve for L_i */</i>	
16 foreach $i \in N$ do in parallel	
17 forall $x \in L_i$ do	
18 $p \leftarrow U(x)$	
19 $q \leftarrow$ solution of $g(x) = 0$	
<i>/* If not converged */</i>	
20 if $p > q$ then	
21 $U(x) \leftarrow q$	
<i>/* If converged */</i>	
22 else	
<i>/* Check adjacent neighbor */</i>	
23 forall x_{nb} adjacent to x do	
24 if $U(x_{nb}) > U(x)$ and $F(x_{nb}) == 0$ then	
25 $p \leftarrow U(x_{nb})$	
26 $q \leftarrow$ solution of $g(x_{nb}) = 0$	
27 if $p > q$ then	
28 $U(x_{nb}) \leftarrow q$	
<i>/* atomic operator */</i>	
29 if $F(x_{nb}) == 0$ then	
30 $F(x_{nb}) \leftarrow 1$	
31 add x_{nb} to L_i	
32 Remove x from L_i	
33 $F(x) \leftarrow 0$	
34 Barrier synchronization	

[0064]

[0065]

이처럼 본 발명의 일 실시예에 따르면, 복수의 스레드들이 독립적으로 액티브 리스트를 관리함으로써 락 동기화 없이 FIM을 수행하여도 race condition이 발생하지 않으며, 복수의 액티브 리스트간에 동일한 그리드 노드가 중복되어 추가되지 않도록 하여 복수의 스레드가 각각의 액티브 리스트를 효율적이면서 안전하게 접근 및 관리할 수 있도록 한다.

[0066]

상술한 본 발명의 실시예들은 다양한 수단을 통해 구현될 수 있다. 예를 들어, 본 발명의 실시예들은 하드웨어, 펌웨어(firmware), 소프트웨어 또는 그것들의 결합 등에 의해 구현될 수 있다.

[0067]

하드웨어에 의한 구현의 경우, 본 발명의 실시예들에 따른 방법은 하나 또는 그 이상의 ASICs(Application Specific Integrated Circuits), DSPs(Digital Signal Processors), DSPDs(Digital Signal Processing

Devices), PLDs(Programmable Logic Devices), FPGAs(Field Programmable Gate Arrays), 프로세서, 컨트롤러, 마이크로 컨트롤러, 마이크로 프로세서 등에 의해 구현될 수 있다.

[0068] 펌웨어나 소프트웨어에 의한 구현의 경우, 본 발명의 실시예들에 따른 방법은 이상에서 설명된 기능 또는 동작들을 수행하는 모듈, 절차 또는 함수 등의 형태로 구현될 수 있다. 소프트웨어 코드는 메모리 유닛에 저장되어 프로세서에 의해 구동될 수 있다. 상기 메모리 유닛은 상기 프로세서 내부 또는 외부에 위치하여, 이미 공지된 다양한 수단에 의해 상기 프로세서와 데이터를 주고 받을 수 있다.

[0069] 이와 같이, 본 발명이 속하는 기술분야의 당업자는 본 발명이 그 기술적 사상이나 필수적 특징을 변경하지 않고서 다른 구체적인 형태로 실시될 수 있다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적인 것이 아닌 것으로서 이해해야만 한다. 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 등가개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

부호의 설명

[0070] 200: 그룹 순서 기반 고속 반복 계산 장치

210: 도메인 설정부

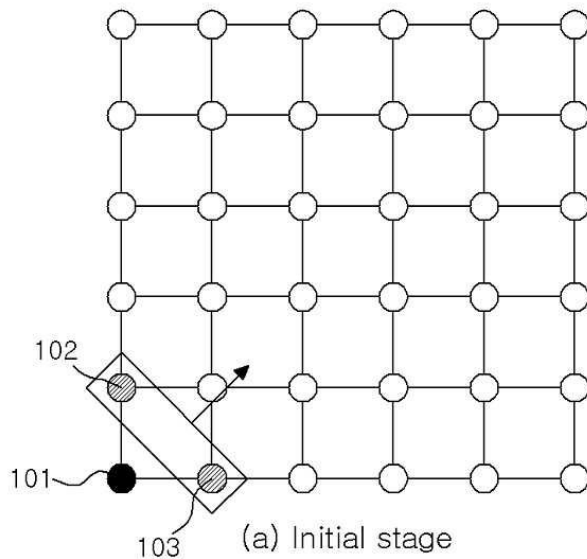
220: 그리드 노드 도출부

230: 액티브 리스트 생성부

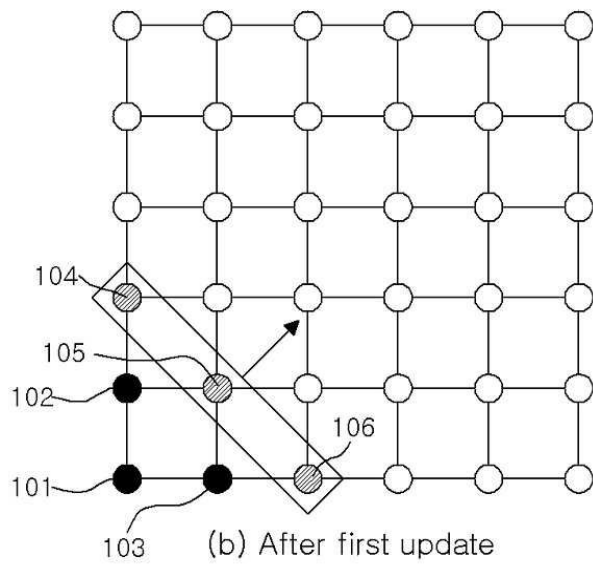
240: FIM 계산부

도면

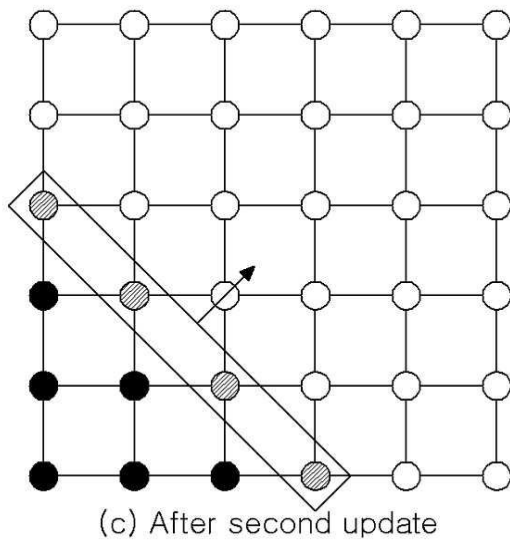
도면1a



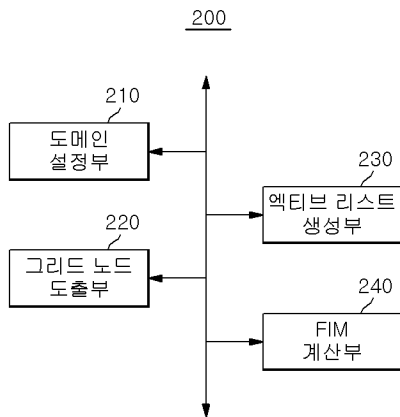
도면1b



도면1c



도면2



도면3

