(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0032670
(43) 공개일자 2021년03월25일

(51) 국제특허분류(Int. Cl.)
G06F 17/16 (2006.01)
(52) CPC특허분류
G06F 17/16 (2013.01)
(21) 출원번호 10-2019-0113981
(22) 출원일자 2019년09월17일
심사청구일자 2019년09월17일

(71) 출원인
연세대학교 산학협력단
서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)
한국과학기술정보연구원
대전광역시 유성구 대학로 245 (어은동)
(72) 발명자
최정일
경기도 고양시 일산서구 대산로 142, 307동 1302호(주엽동, 문촌마을3단지아파트)
김기하
서울특별시 송파구 송파대로 567, 530동 404호(잠실동, 잠실주공아파트)
강지훈
세종특별자치시 대평로 34, 405동 2102호(대평동, 해들마을4단지)
(74) 대리인
민영준

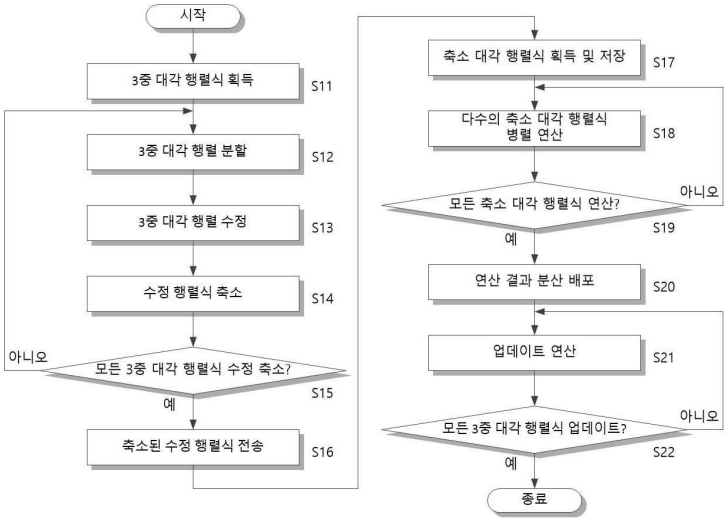
전체 청구항 수 : 총 6 항

(54) 발명의 명칭 **3중 대각행렬식 연산 방법**

(57) 요 약

본 발명은 멀티 코어 분산 메모리 시스템을 위한 3중 대각행렬식 연산 방법으로서, 다수의 코어 각각이 연속으로 인가되는 다수의 3중 대각행렬식 각각에서 코어의 개수에 대응하는 개수의 행단위로 분할된 분할 행렬을 획득하고, 기지정된 방식에 따라 수정하여 수정 행렬식을 획득하고, 수정 행렬식의 제1 행 및 마지막 행을 추출하여 축소 수정 행렬식을 획득하는 단계, 다수의 3중 대각행렬식 각각에서 획득된 축소 수정 행렬식을 다수의 코어의 기지정된 순서에 따라 다수의 코어 중 대응하는 코어로 전송하는 단계, 다수의 코어 각각이 전송된 축소 수정 행렬식을 결합하여 축소 대각행렬식을 획득 및 저장하는 단계, 다수의 코어 각각이 병렬로 획득된 축소 수정 행렬식의 해를 연산하는 단계 및 축소 대각행렬식의 해를 이용하여 3중 대각행렬식의 나머지 해를 연산하는 단계를 포함하여, 멀티 코어 분산 메모리 시스템의 연산 효율성을 극대화 할 수 있다.

대 표 도



이 발명을 지원한 국가연구개발사업

과제고유번호	1711076842
부처명	과학기술정보통신부
과제관리(전문)기관명	한국과학기술정보연구원
연구사업명	슈퍼컴퓨팅 활용 확대를 위한 과학, 공학 분야 거대문제 해결 지원
연구과제명	고병렬 열유동해석자를 활용한 거대규모 자연대류 열전달 해석에 관한 연구
기 여 율	1/1
과제수행기관명	한국과학기술정보연구원
연구기간	2019.03.01 ~ 2019.10.31

명 세 서

청구범위

청구항 1

멀티 코어 분산 메모리 시스템을 위한 3중 대각행렬식 연산 방법에 있어서,

다수의 코어 각각이 연속으로 인가되는 다수의 3중 대각행렬식 각각에서 코어의 개수에 대응하는 개수의 행단위로 분할된 분할 행렬을 획득하고, 기지정된 방식에 따라 수정하여 수정 행렬식을 획득하고, 상기 수정 행렬식의 제1 행 및 마지막 행을 추출하여 축소 수정 행렬식을 획득하는 단계;

상기 다수의 3중 대각행렬식 각각에서 획득된 축소 수정 행렬식을 상기 다수의 코어의 기지정된 순서에 따라 다수의 코어 중 대응하는 코어로 전송하는 단계;

다수의 코어 각각이 전송된 상기 축소 수정 행렬식을 결합하여 축소 대각행렬식을 획득 및 저장하는 단계;

다수의 코어 각각이 병렬로 획득된 축소 수정 행렬식의 해를 연산하는 단계; 및

축소 대각행렬식의 해를 이용하여 3중 대각행렬식의 나머지 해를 연산하는 단계를 포함하는 3중 대각행렬식 연산 방법.

청구항 2

제1 항에 있어서, 상기 코어로 전송하는 단계는

상기 다수의 3중 대각 행렬식이 인가된 순서에 따라 다수의 3중 대각 행렬식에서 획득된 상기 축소 수정 행렬식을 상기 다수의 코어에 기지정된 순서로 전송하는 3중 대각행렬식 연산 방법.

청구항 3

제1 항에 있어서, 상기 축소 수정 행렬식의 해를 연산하는 단계는

저장된 축소 수정 행렬식의 개수가 코어 개수를 초과하면, 코어 개수에 대응하는 개수의 축소 수정 행렬식의 해를 병렬로 연산하고, 나머지 축소 수정 행렬식의 해를 이후 코어 개수 단위로 병렬로 연산하는 3중 대각행렬식 연산 방법.

청구항 4

제1 항에 있어서, 상기 수정 행렬식을 획득하는 단계는

상기 분할 행렬을 수정 토마스 알고리즘에 따라 수정하여 수정 행렬식을 획득하고,

상기 축소 대각행렬식의 해를 연산하는 단계는

토마스 알고리즘에 따라 연산을 수행하는 3중 대각행렬식 연산 방법.

청구항 5

제1 항에 있어서, 상기 나머지 해를 연산하는 단계는

연산된 축소 대각행렬식의 해를 다수의 코어로 분산 전송하는 단계; 및

상기 축소된 대각행렬식의 해와 대응하는 상기 수정 행렬식을 토마스 알고리즘의 업데이트 알고리즘에 대입하여 연산하는 단계를 포함하는 3중 대각행렬식 연산 방법.

청구항 6

제5 항에 있어서, 상기 분산 전송하는 단계는

상기 축소된 대각행렬식의 해를 대응하는 축소된 수정 행렬식을 전송한 코어로 전송하는 3중 대각행렬식 연산 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 3중 대각행렬식을 연산하는 연산 방법에 관한 것으로, 멀티 코어 분산 메모리 시스템에서 하나 또는 다수의 3중 대각행렬식을 효율적으로 병렬 연산할 수 있도록 하는 연산 방법에 관한 것이다.

배 경 기 술

[0002] 3중 대각행렬식은 선형 연립 방정식의 하나로 행렬의 형태가 대각행렬의 주대각선을 포함해 대각성분이 3중 구조인 경우로써, 유체역학, 열전달, 양자역학, 전자기학 등에서 수치해석으로 특정 문제에 대한 해를 구할 때 빈번하게 나타나는 형태이다.

[0003] 3중 대각행렬식의 해를 구하는 알고리즘 중 널리 사용되는 토마스 알고리즘(Thomas algorithm)은 가우시안 소거법의 특수한 형태로써, 순차 연산 처리 방식으로 단순히 연산 처리 관점에서는 가장 효율적인 방법이다. 그러나 토마스 알고리즘은 그 계산과정이 순차적으로 진행되기 때문에 병렬화가 불가능하다. 즉 멀티 코어 분산 메모리 시스템과 같이 고성능의 연산 시스템에 적용 시에 병렬 연산을 제공할 수 없어 효율성이 크게 낮아진다.

[0004] PCR 알고리즘(parallel cyclic reduction algorithm)은 3중 대각행렬식에 대한 병렬 연산 처리가 가능하도록 고안된 방법이다. 이 방법은 재귀적 알고리즘으로써 방정식 3개씩 한 묶음으로 미지수를 소거해 병렬적으로 처리가 가능 하다는 장점이 있지만 토마스 알고리즘보다 기본적 효율이 좋지 않다. 이러한 효율성의 차이로 인한 문제는 해결해야하는 3중 대각행렬식의 크기가 크고, 수가 많을 수록 증가된다. 또한 PCR 알고리즘은 분산 메모리 시스템에 적용하기 적합하지 않다.

[0005] 한편 3중 대각행렬식을 분산메모리 시스템에서 병렬적으로 계산하는 알고리즘도 고안된 바가 있다(Mattor et al 1995). 이 방법은 먼저 각 계산 노드에서 정리된 원소 값을 모아 작은 크기의 하위 3중 대각행렬식을 만들어 그 해를 구한다. 그 다음 하위 3중 대각행렬식의 해를 이용해 각 계산 노드에서 병렬적으로 원래 3중 대각행렬식의 해를 구한다. 하지만 하나의 3중 대각행렬식을 계산할 때 모든 계산 노드에서 하위 3중 대각행렬식의 해를 구하는 과정이 불필요하게 중복되어 여전히 효율성이 낮다는 한계가 있다.

선행기술문헌

특허문헌

[0006] (특허문헌 0001) 미국 공개 특허 2019/0153824(2019.05.23 공개)

발명의 내용

해결하려는 과제

[0007] 본 발명의 목적은 멀티 코어 분산 메모리 시스템에서 효율적으로 3중 대각행렬식을 해결할 수 있는 3중 대각행렬식 연산 방법을 제공하는데 있다.

과제의 해결 수단

[0008] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따른 3중 대각행렬식 연산 방법은 멀티 코어 분산 메모리 시스템을 위한 3중 대각행렬식 연산 방법에 있어서, 다수의 코어 각각이 연속으로 인가되는 다수의 3중 대각행렬식 각각에서 코어의 개수에 대응하는 개수의 행단위로 분할된 분할 행렬을 획득하고, 기지정된 방식에 따라 수정하여 수정 행렬식을 획득하고, 상기 수정 행렬식의 제1 행 및 마지막 행을 추출하여 축소 수정 행렬식을 획득하는 단계; 상기 다수의 3중 대각행렬식 각각에서 획득된 축소 수정 행렬식을 상기 다수의 코어의 기지정된 순서에 따라 다수의 코어 중 대응하는 코어로 전송하는 단계; 다수의 코어 각각이 전송된 상기 축소 수정 행렬식을 결합하여 축소 대각행렬식을 획득 및 저장하는 단계; 다수의 코어 각각이 병렬로 획득된 축소 수정 행렬식의 해를 연산하는 단계; 및 축소 대각행렬식의 해를 이용하여 3중 대각행렬식의 나머지 해를 연산하는 단계를 포함한다.

[0009] 상기 코어로 전송하는 단계는 상기 다수의 3중 대각 행렬식이 인가된 순서에 따라 다수의 3중 대각 행렬식에서

획득된 상기 축소 수정 행렬식을 상기 다수의 코어에 기지정된 순서로 전송할 수 있다.

[0010] 상기 축소 수정 행렬식의 해를 연산하는 단계는 저장된 축소 수정 행렬식의 개수가 코어 개수를 초과하면, 코어 개수에 대응하는 개수의 축소 수정 행렬식의 해를 병렬로 연산하고, 나머지 축소 수정 행렬식의 해를 이후 코어 개수 단위로 병렬로 연산할 수 있다.

[0011] 상기 수정 행렬식을 획득하는 단계는 상기 분할 행렬을 수정 토마스 알고리즘에 따라 수정하여 수정 행렬식을 획득하고, 상기 축소 대각행렬식의 해를 연산하는 단계는 토마스 알고리즘에 따라 연산을 수행할 수 있다.

[0012] 상기 나머지 해를 연산하는 단계는 연산된 축소 대각행렬식의 해를 다수의 코어로 분산 전송하는 단계; 및 상기 축소된 대각행렬식의 해와 대응하는 상기 수정 행렬식을 토마스 알고리즘의 업데이트 알고리즘에 대입하여 연산하는 단계를 포함할 수 있다.

[0013] 상기 나머지 해를 연산하는 단계는 연산된 축소 대각행렬식의 해를 다수의 코어로 분산 전송하는 단계; 및 상기 축소된 대각행렬식의 해와 대응하는 상기 수정 행렬식을 토마스 알고리즘의 업데이트 알고리즘에 대입하여 연산하는 단계를 포함할 수 있다.

[0014] 상기 분산 전송하는 단계는 상기 축소된 대각행렬식의 해를 대응하는 축소된 수정 행렬식을 전송한 코어로 전송할 수 있다.

발명의 효과

[0015] 따라서, 본 발명의 실시예에 따른 3중 대각행렬식 연산 방법은 멀티 코어3중 대각행렬식을 해결함에 있어 병렬 확장성을 향상시켜 멀티 코어 분산 메모리 시스템에 최적화된 연산 성능을 제공할 수 있으며, 코어간 통신량과 유휴 시간을 최소화하여 부하를 저감할 수 있으며, 중복 연산을 방지하여 부하 균등성을 향상시킬 수 있어 연산 효율성을 극대화할 수 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일 실시예에 따른 3중 대각행렬식 연산 방법을 나타낸다.

도 2는 3중 대각행렬식의 일예를 나타낸다.

도 3은 도 1의 3중 대각행렬식 단계에서 수정된 3중 대각행렬식의 일예를 나타낸다.

도 4는 도 1의 수정 대각행렬식 축소 단계에서 축소된 대각행렬식의 일예를 나타낸다.

도 5는 축소된 대각행렬식을 분산 연산하는 예를 나타낸다.

도 6은 도 1의 3중 대각행렬식 연산 방법의 전체 연산 과정을 시각적으로 나타낸다.

도 7은 다수의 3중 대각행렬식 연산에서 코어 사이에 전송되는 데이터를 시각적으로 나타낸 도면이다.

도 8은 본 발명의 일 실시예에 따른 3중 대각행렬식 연산 방법의 성능을 비교 시뮬레이션한 결과를 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0017] 본 발명과 본 발명의 동작상의 이점 및 본 발명의 실시에 의하여 달성되는 목적을 충분히 이해하기 위해서는 본 발명의 바람직한 실시예를 예시하는 첨부 도면 및 첨부 도면에 기재된 내용을 참조하여야만 한다.

[0018] 이하, 첨부한 도면을 참조하여 본 발명의 바람직한 실시예를 설명함으로써, 본 발명을 상세히 설명한다. 그러나, 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며, 설명하는 실시예에 한정되는 것이 아니다. 그리고, 본 발명을 명확하게 설명하기 위하여 설명과 관계없는 부분은 생략되며, 도면의 동일한 참조부호는 동일한 부재임을 나타낸다.

[0019] 명세서 전체에서, 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라, 다른 구성요소를 더 포함할 수 있는 것을 의미한다. 또한, 명세서에 기재된 "...부", "...기", "모듈", "블록" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어나 소프트웨어 또는 하드웨어 및 소프트웨어의 결합으로 구현될 수 있다.

[0020] 도 1은 본 발명의 일 실시예에 따른 3중 대각행렬식 연산 방법을 나타내고, 도 2는 3중 대각행렬식의 일예를 나타내며, 도 3은 도 1의 3중 대각행렬식 단계에서 수정된 3중 대각행렬식의 일예를 나타낸다. 그리고 도 4는 도

1의 수정 대각행렬식 축소 단계에서 축소된 대각행렬식의 일예를 나타내고, 도 5는 축소된 대각행렬식을 분산 연산하는 예를 나타낸다.

[0021] 도 1 내지 도 5를 참조하면, 본 실시예에 따른 3중 대각행렬식 연산 방법은 우선 연산이 수행되어야 하는 다수의 3중 대각행렬식을 획득한다(S11).

[0022] 여기서 3중 대각행렬식은 Ax = d의 형태로 표현되는 행렬식으로, A가 N × N 크기의 3중 대각행렬이고, x와 d는 각각 길이 N을 갖는 열 벡터이다. 이러한 3중 대각행렬식은 행렬 원소 a_i, b_i, c_i (여기서 i = 1, …, N)로 구성된 3중 대각행렬(A)과 우변의 d_i를 원소로 갖는 열 벡터(d)에 대해 x_i를 원소로 갖는 미지의 열 벡터(x)를 구하는 선형 연립 방정식이다.

[0023] 즉 3중 대각행렬식은 수학식 1로 표현될 수 있다.

수학식 1

[0024] $$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, i = 1, \ldots, N$$

[0025] 여기서 a_1 과 c_N 의 값은 0이다.

[0026] 다수의 3중 대각행렬식이 획득되면, 연산을 수행하는 다수의 코어의 개수에 대응하여 획득된 3중 대각행렬식을 도 2에 도시된 바와 같이 행단위로 균등하게 분할한다(S12).

[0027] 본 실시예에 따른 3중 대각행렬식 연산 방법은 멀티 코어 분산 메모리 시스템에 적용되어 수행되는 것을 전제로 한다. 멀티 코어 분산 메모리 시스템은 각각의 코어가 개별적으로 연산을 수행할 수 있으므로, 3중 대각행렬식 연산을 병렬화하면 연산 효율성을 크게 높일 수 있다. 다만 분산 메모리 시스템에서 각각의 코어는 메모리를 공유하지 않으므로, 각각의 코어에서 연산된 결과는 코어간 통신을 통해 상호 전달되어야 하며, 이 과정에서 대량의 통신 트래픽을 유발하게 된다. 뿐만 아니라 다수의 3중 대각행렬식 연산을 연속하여 수행하는 경우, 다수의 코어에 분산되어 연산된 결과를 특정 코어가 인가받아 최종 연산을 수행하며, 이 과정에서 나머지 코어들이 유휴 상태로 유지됨으로써 효율성을 극대화할 수 없다는 한계가 있다. 이에 본 실시예에서는 3중 대각행렬식 연산을 병렬화하고 시분할 다중화 기법으로 병렬 연산이 수행되도록 함으로써, 연산 효율성이 극대화되도록 한다.

[0028] 이를 위해, 본 실시예에서는 다수의 코어에서 병렬적으로 동시에 연산을 수행할 수 있도록 3중 대각행렬식을 코어의 개수에 대응하여 분할한다.

[0029] 연산 코어의 개수가 P개인 경우, N × N 크기의 3중 대각행렬(A)을 행단위로 N/P = m 개씩 나누어 분할하여 P개의 코어로 제공할 수 있다. 일예로 도 2에 도시된 바와 같이, 3중 대각행렬이 12 × 12 크기(N = 12)의 3중 대각행렬이고, 연산 코어의 개수가 3개인 경우, 3중 대각행렬을 4행씩 균등하게 분할할 수 있다. 도 2의 행렬에서 여백은 0값을 갖는 원소이다. 3중 대각행렬을 m행씩 균등 분할되면, 이에 대응하여 열 또한 m열씩 균등 분할된다. 즉 m × m 크기의 P × P개의 분할 행렬로 분할된다.

[0030] 그리고 분할된 행렬식 각각은 수학식 2와 같이 표현될 수 있다.

수학식 2

[0031] $$a_i^p x_1^p + b_i^p x_i^p + c_i^p x_m^p = d_i^p, i = 1, \ldots, m$$

[0032] 여기서 p는 분할 행렬의 인덱스로서 0 ≤ p ≤ P-1 이고, x_0^p 와 x_{m+1}^p 는 각각 x_m^{p-1} 와 x_1^{p+1} 에 대응한다.

[0033] 행렬식이 분할되면, 분할 행렬식을 인가받은 각 코어는 분할된 행렬식에 대해 표 1의 수정 토마스 알고리즘을 적용하여, 분할된 행렬식을 수정한다(S13).

표 1

Algorithm : The modified Thomas algorithm

input: a^p, b^p, c^p, d^p

$d_1^p \leftarrow d_1^p / b_1^p$;
$c_1^p \leftarrow c_1^p / b_1^p$;
$a_1^p \leftarrow a_1^p / b_1^p$;

$d_2^p \leftarrow d_2^p / b_2^p$;
$c_2^p \leftarrow c_2^p / b_2^p$;
$a_2^p \leftarrow a_2^p / b_2^p$;

for $i = 3, ..., m$ do
  $r \leftarrow 1/(b_i^p - a_i^p c_{i-1}^p)$;
  $d_i^p \leftarrow r(d_i^p - a_i^p d_{i-1}^p)$;
  $c_i^p \leftarrow r c_i^p$;
  $a_i^p \leftarrow -r a_i^p a_{i-1}^p$;
end
for $i = m-2, ..., 2$ do
  $d_i^p \leftarrow d_i^p - c_i^p d_{i+1}^p$;
  $c_i^p \leftarrow -c_i^p c_{i+1}^p$;
  $a_i^p \leftarrow a_i^p - c_i^p a_{i+1}^p$;
end
$r \leftarrow 1/(1_i - a_2^p c_1^p)$;
$d_1^p \leftarrow r(d_i^p - a_i^p d_{i-1}^p)$;
$c_1^p \leftarrow -r c_1^p c_2^p$;
$a_1^p \leftarrow r a_1^p$;

[0034]

[0035] 수정 토마스 알고리즘은 3중 대각행렬식을 계산하는 기법으로 알려진 알고리즘으로 수정 토마스 알고리즘에 따라 3중 대각행렬을 변환하면, 도 3에 도시된 바와 같이, 3중 대각행렬의 대각선 원소가 모두 1로 변환된다. 또한 각행의 첫번째 원소, 즉 분할된 행렬식의 첫번째 원소가 0이 아닌 값으로 변환된다.

[0036] 이에 각 코어는 수정된 행렬식에서 제1 행 및 제m 행에 대한 방정식을 수학식 3과 같이 변환할 수 있다.

수학식 3

$$a_1^{\dagger,p} x_0^p + x_1^p + c_1^{\dagger,p} x_m^p = d_1^{\dagger,p}$$

[0037]
$$a_m^{\dagger,p} x_1^p + x_m^p + c_m^{\dagger,p} x_{m+1}^p = d_m^{\dagger,p}$$

[0038] 그리고 제2 행 내지 제 m-1 행에 대한 방정식은 수학식 4와 같이 변환할 수 있다.

수학식 4

[0039] $$a_i^{\dagger,p}x_m^{p-1}+x_i^p+c_i^{\dagger,p}x_m^p=d_i^{\dagger,p}$$

[0040] 수학식 3 및 4에서 †는 수정 토마스 알고리즘으로 수정된 원소 계수를 나타낸다.

[0041] 한편, x_0^p 및 x_{m+1}^p 를 x_m^{p-1} 및 x_1^{p+1} 로 대체하면, 수학식 3은 수학식 5로 표현된다.

수학식 5

[0042] $$a_1^{\dagger,p}x_m^{p-1}+x_1^p+c_1^{\dagger,p}x_m^p=d_1^{\dagger,p}$$

$$a_m^{\dagger,p}x_1^p+x_m^p+c_m^{\dagger,p}x_1^{p+1}=d_m^{\dagger,p}$$

[0043] 그리고 각각의 코어는 대응하는 분할된 행렬식이 수정되면, 수정된 행렬식에서 제1 행 및 제m 행만을 추출하여 수정 행렬식을 축소한다(S14).

[0044] 도 3 및 도 4에서는 3개의 코어가 각각 분할된 행렬식을 수정 및 축소하여 획득하는 것을 시각적으로 표시하기 위해, 서로 다른 코어에서 연산되는 원소에 대해 서로 다른 색상으로 표시하였다.

[0045] 수정 행렬식이 축소되면, 획득된 모든 3중 대각행렬식을 수정 및 축소하였는지 판별한다(S15). 만일 획득된 3중 대각행렬식 중 수정 및 축소되지 않은 3중 대각행렬식이 존재하면 다음 연산되어야 하는 다음 3중 대각행렬을 분할하고 수정 및 축소하여 축소된 수정 행렬을 획득한다. 여기서 축소된 수정 행렬식 각각은 각 코어에 대응하는 메모리에 임시 저장될 수 있다.

[0046] 그러나 모든 3중 대각행렬식이 수정 및 축소된 것으로 판별되면, P개의 코어 각각은 저장된 다수의 축소된 수정 행렬식을 서로 다른 코어로 전송한다(S16). 여기서 P개의 코어 각각은 축소된 수정 행렬이 획득된 순서에 기반하여 동일한 시간 구간에 획득된 축소된 수정 행렬을 기지정된 순서에 따라 하나의 코어로 전송하고, 다음 시간 구간에 획득된 축소된 수정 행렬식을 다음 지정된 코어로 전송한다.

[0047] 여기서 축소 대각행렬식을 획득하는 코어는 다른 코어들로부터 2개의 행을 갖는 축소된 수정 행렬식을 인가받으므로, 각각의 코어로부터 m개의 행을 모두 인가받는 경우에 비해, 통신량이 크게 줄어들게 된다. 즉 코어간 통신 효율성을 크게 높일 수 있다.

[0048] 그리고 다른 코어들로부터 축소된 수정 행렬식을 인가받은 코어는 축소된 수정 행렬식을 결합하여 도 4와 같이 축소된 대각행렬식을 획득하여 저장한다(S17).

[0049] 만일 획득된 3중 대각행렬식의 개수가 코어의 개수보다 많으면, 기지정된 순서에 따라 반복적으로 축소된 수정 행렬식을 인가받아 결합하여 축소된 대각행렬식을 획득하고 저장할 수 있다.

[0050] 획득된 모든 3중 대각행렬식에 대한 축소된 대각행렬식이 획득되면, 다수의 코어 각각이 다수의 축소 대각행렬식에 대해 병렬로 동시에 연산을 수행한다(S18).

[0051] 도 4에 도시된 바와 같이 축소된 대각행렬식 또한 3중 대각행렬식의 형태로 획득되며, P개의 코어 각각은 표 2 및 표 3의 토마스 알고리즘을 이용하여 축소된 대각행렬식의 해를 연산한다.

표 2

Algorithm : The Thomas algorithm

input : a,b,c,d
output: d
$d_1 \leftarrow d_1/b_1$; $c_1 \leftarrow c_1/b_1$;
for $i = 2, ..., N$ **do**
  $r \leftarrow 1/(b_i - a_i c_{i-1})$;
  $d_i \leftarrow r(d_i - a_i d_{i-1})$;
  $c_i \leftarrow rc_i$;
end
for $i = N - 1, ..., 1$ **do**
  $d_i \leftarrow d_i - c_i d_{i+1}$;
end

[0052]

[0053] 토마스 알고리즘에 따라 축소된 대각행렬식의 해가 연산되면, 모든 축소 대각행렬식에 대해 연산이 수행되었는지 판별한다(S19). 상기한 바와 같이 획득된 3중 대각행렬식의 개수가 코어의 개수보다 많은 경우, 다수의 코어는 저장된 모든 축소 대각행렬식을 동시에 연산할 수 없다. 즉 한번에 병렬로 연산을 수행할 수 있는 축소 대각행렬식의 개수는 코어의 개수로 한정된다. 이에 모든 축소 대각행렬식에 대해 연산이 수행되었는지 판별하고 연산되지 않은 축소 대각행렬식이 존재하면, 다시 다수의 코어 각각이 동시에 병렬로 서로 다른 축소 대각행렬식에 대한 연산을 수행한다(S18).

[0054] 그러나 모든 축소 대각행렬식에 대한 연산을 수행된 것으로 판별되면, 축소된 대각행렬식의 해를 다시 P개로 분할하여 P개의 코어로 분산 배포한다(S20). 토마스 알고리즘에 따라 해지는 해는 m개의 행을 갖도록 분할된 행렬식 각각에서 제1 행 및 제m 행에 대한 해로서, 3중 대각행렬식에 대한 완전한 해를 구하기 위해서는 제2 행 내지 제m -1 행에 대한 해가 추가로 계산되어야 한다.

[0055] 이에 제2 행 내지 제m -1 행에 대한 연산 또한 병렬로 수행되도록 축소된 대각행렬식의 해를 P개로 분할하여 P개의 코어로 분산 배포한다. 이때, 분산 배포되는 축소된 대각행렬식의 해는 대응하는 축소된 수정 행렬식이 전송된 코어로 전달될 수 있다.

[0056] 그리고 P개의 코어 각각은 이전 계산한 수정된 행렬식에 인가된 P개로 분할된 축소된 대각행렬식의 해를 대입하여 표 3으로 나타나는 병렬로 업데이트 알고리즘에 따라 반복 연산을 수행하여 3중 대각행렬식의 제2 행 내지 제m -1 행에 대한 해를 획득한다(S21).

표 3

Algorithm : Update algorithm

for $i = 2, ..., m - 1$ **do**
  $d_i^p \leftarrow d_i^p - a_i^p d_1^p - c_i^p d_m^p$;
end

[0057]

[0058] 도 2에서는 12 × 12 크기의 3중 대각행렬에 대한 행렬식을 3개의 코어가 병렬로 연산을 수행하는 것으로 가정하였으므로, 3개의 코어 각각은 3개의 분할된 행렬식의 제2 행 및 제3 행에 대한 해를 도 5에서와 같이 획득할 수 있다.

[0059] 그리고 획득된 모든 축소된 대각행렬식에 대한 업데이트 연산이 수행되었는지 판별한다(S22). 만일 모든 축소된 대각행렬식에 대한 업데이트 연산이 수행된 것으로 판별하면, 3중 대각행렬식 연산을 종료한다. 그러나 업데이트 연산이 수행되지 않은 축소된 대각행렬식이 존재하면 다시 P개의 코어는 병렬로 업데이트 연산을 수행한다(S21).

[0060] 수치해석과 같이 3중 대각행렬식을 해석해야 하는 분야에서는 3중 대각행렬식 하나만 나타나는 경우는 매우 드물며, 대부분 대량의 3중 대각행렬식을 연산해야 하는 경우가 빈번하게 발생한다. 즉 연속하여 다수의 3중 대각행렬식을 연산해야 하는 경우가 빈번하게 발생한다. 이에, 만일 축소된 대각행렬식을 획득한 코어는 곧바로 축소된 대각행렬식에 대한 해를 연산하게 되면, 나머지 코어는 축소된 대각행렬식을 획득한 코어가 해를 연산하는 동안 유휴 상태에 놓이게 된다. 즉 연산 효율성을 크게 떨어뜨리는 결과를 초래한다.

[0061] 이에 본 실시예에서는 다수의 3중 대각행렬식 각각에 대한 축소된 수정 행렬식을 병렬로 획득하고, 획득된 다수의 3중 대각행렬식에 대한 축소된 수정 행렬식을 일괄로 전송하도록 함으로써, 코어간 통신 시간을 줄일 수 있다. 뿐만 아니라, 다수의 코어 각각이 전송된 축소된 수정 행렬식을 결합한 축소 대각 행렬식을 병렬로 연산하고, 연산 결과를 다시 다수의 코어에 분산 배포하여 업데이트 연산을 수행하도록 함으로써 다수의 코어의 유휴 시간을 최소화할 수 있다.

[0062] 결과적으로 코어간 통신량과 코어의 유휴 시간을 최소화하여 도 2에 도시된 3중 대각행렬식 전체에 대한 해를 계산할 수 있다.

[0063] 도 6은 도 1의 3중 대각행렬식 연산 방법의 전체 연산 과정을 시각적으로 나타내고, 도 7은 다수의 3중 대각행렬식 연산에서 코어 사이에 전송되는 데이터를 시각적으로 나타낸 도면이다.

[0064] 도 1 내지 도 5를 참조하여 도 6의 전체 연산 과정을 다시 살펴보면, 3중 대각행렬식이 획득되면 다수개의 코어 각각이 3중 대각행렬식에서 분할된 행렬식을 인가받고, 인가된 분할 행렬식을 수정 토마스 알고리즘에 따라 수정하여 수정 행렬식을 획득하고, 수정 행렬식에서 제1 행 및 마지막 행을 추출하여 축소된 수정 행렬식을 획득한다. 그리고 모든 3중 대각행렬식에 대한 축소된 수정 행렬식을 획득되면, 다수의 축소된 수정 행렬식을 기지정된 순서로 서로 다른 코어로 전달한다.

[0065] 축소된 수정 행렬식을 인가받은 다수의 코어 각각은 축소된 수정 행렬식을 결합하여 축소된 대각행렬식을 획득하여 저장하고, 각각의 코어가 토마스 알고리즘을 이용하여 축소된 대각행렬식의 해를 병렬로 연산한다. 즉 분할된 행렬식 각각의 제1 행 및 마지막 행의 해를 연산한다.

[0066] 저장된 모든 축소된 대각행렬식의 해가 획득되면, 연산 결과를 다시 코어의 개수에 대응하여 분할하여 다수의 코어로 분산 배포한다. 이때, 다수의 코어는 일예로 MPI_Alltoall 방식으로 통신을 수행할 수 있다.

[0067] 분산 배포된 축소된 대각행렬식의 분할 해는 다수의 코어 각각에서 이전 계산된 수정된 행렬식과 함께 업데이트 알고리즘에 적용되어, 분할된 행렬식의 제1 행 및 마지막 행을 제외한 나머지 행에 대한 해를 연산하여 3중 대각행렬식의 전체 해를 획득한다.

[0068] 다수의 코어는 획득된 모든 3중 대각행렬식에 대한 해가 획득될 때까지 병렬로 반복적으로 업데이트 알고리즘을 수행한다.

[0069] 도 8은 본 발명의 일 실시예에 따른 3중 대각행렬식 연산 방법의 성능을 비교 시뮬레이션한 결과를 나타낸다.

[0070] 도 8에서 A와 B는 기존의 3중 대각행렬식 연산 방법으로 A는 3중 대각행렬식 전체를 다수의 코어로 전송하여 연산하는 기법을 나타내고, B는 단일 코어의 시분할 다중화 방식으로 연산하는 기법을 나타낸다. 그리고 C는 본 실시예에 따른 3중 대각행렬식 연산 방법인 PaScal TDMA(Parallel and Scalable Library for TDMA) 기법을 나타낸다. 도 8은 코어당 격자 크기를 512^2으로 고정하고, 코어를 8개에서 4096개까지 증가시키며 다수의 3중 대각행렬식을 연산하는 경우, 각 코어별 데이터 통신 시간을 시뮬레이션한 결과이다.

[0071] 도 8에 도시된 바와 같이, 본 실시예에 따른 3중 대각행렬식 연산 방법은 기존에 비해 코어간 통신 시간이 크게 저감되었을 뿐만 아니라, 코어의 개수가 증가될수록 기존에 비해 코어간 통신 시간이 더 크게 저감되었음을 확인할 수 있다.

[0072] 본 발명에 따른 방법은 컴퓨터에서 실행시키기 위한 매체에 저장된 컴퓨터 프로그램으로 구현될 수 있다. 여기서 컴퓨터 판독가능 매체는 컴퓨터에 의해 액세스 될 수 있는 임의의 가용 매체일 수 있고, 또한 컴퓨터 저장 매체를 모두 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는

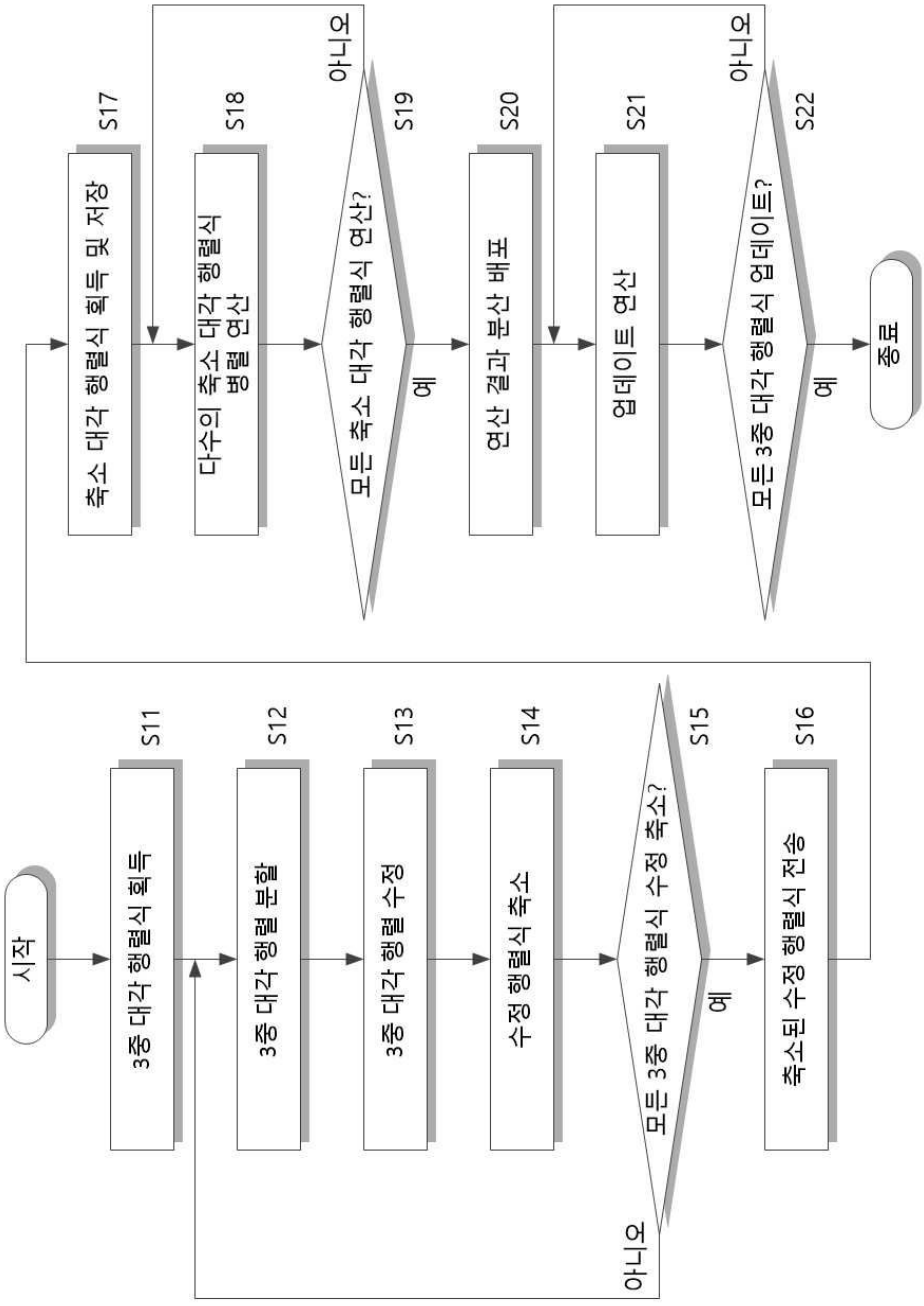
기타 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현된 휘발성 및 비휘발성, 분리형 및 비분리형 매체를 모두 포함하며, ROM(판독 전용 메모리), RAM(랜덤 액세스 메모리), CD(컴팩트 디스크)-ROM, DVD(디지털 비디오 디스크)-ROM, 자기 테이프, 플로피 디스크, 광데이터 저장장치 등을 포함할 수 있다.

[0073] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다.

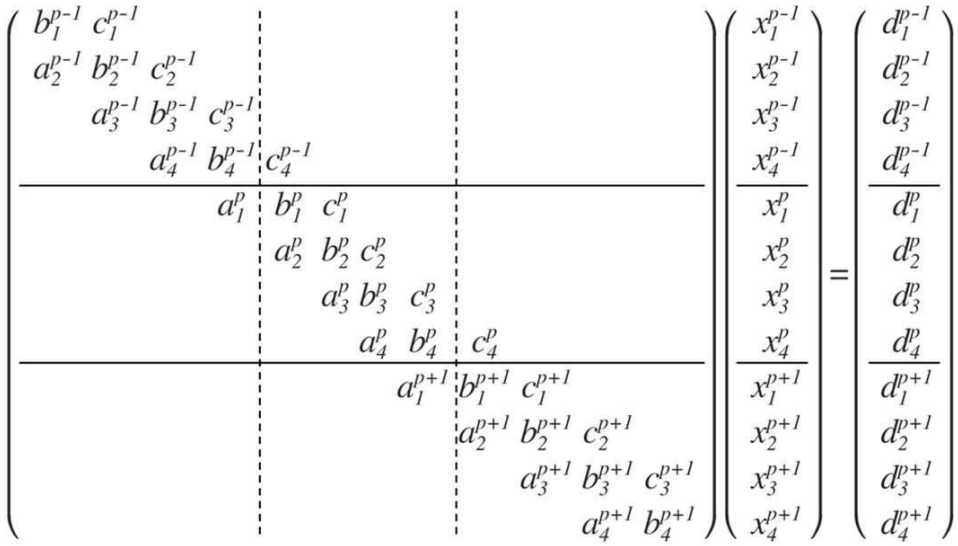
[0074] 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

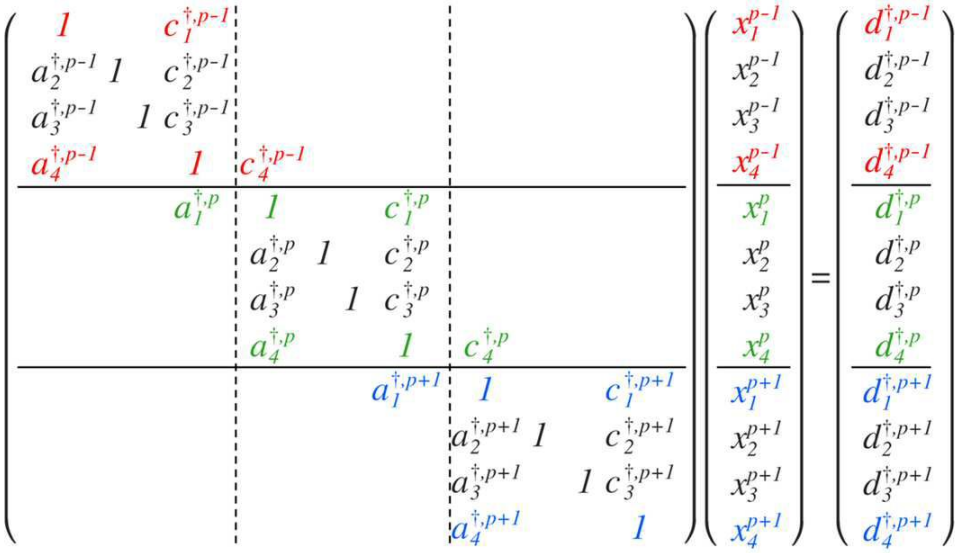
도면1



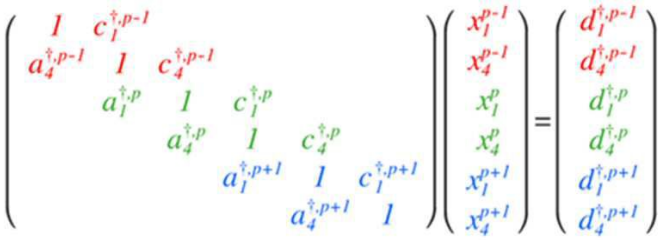
도면2



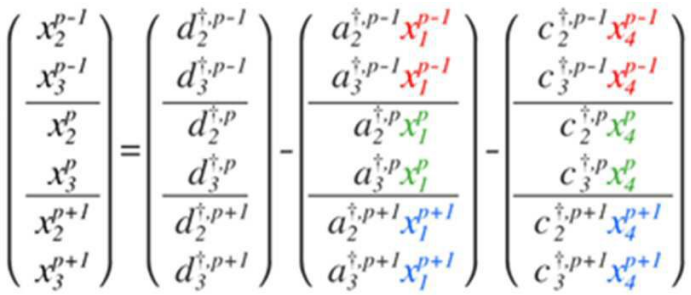
도면3



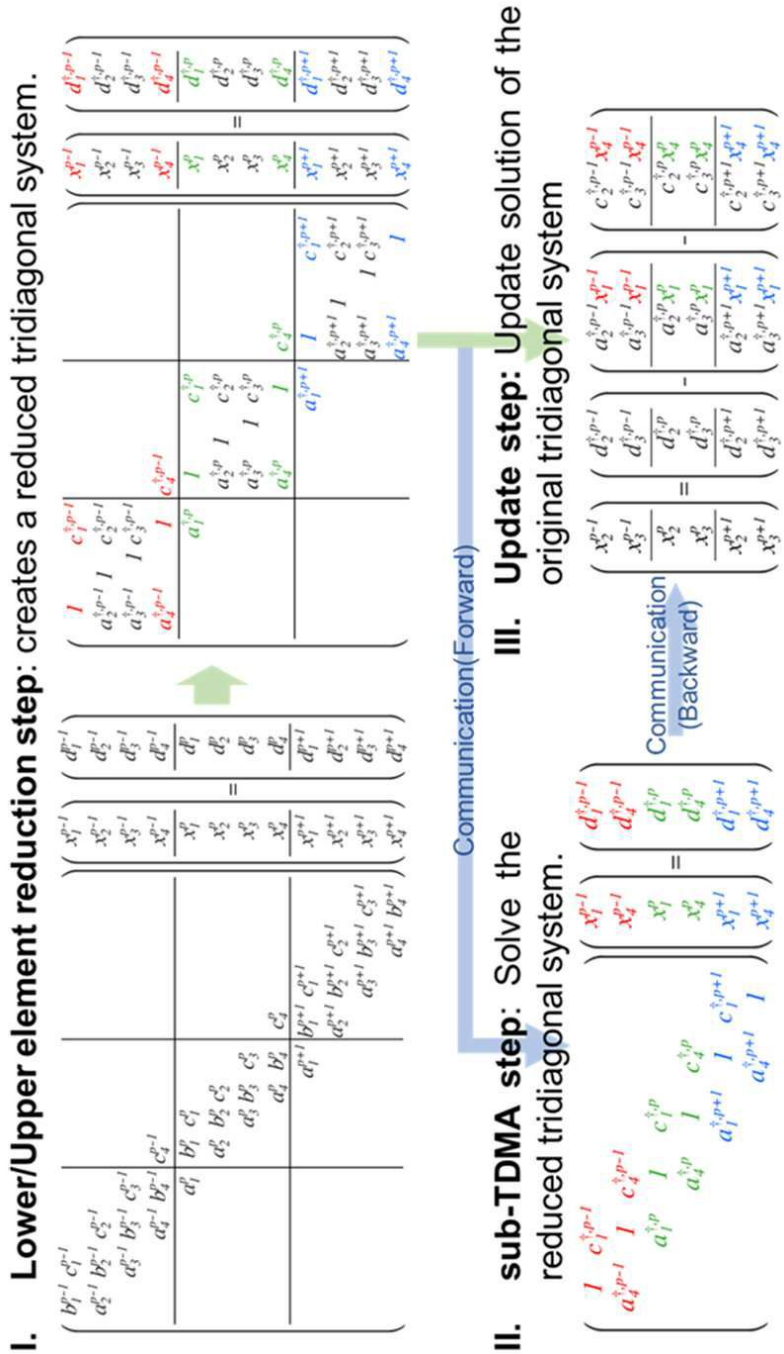
도면4



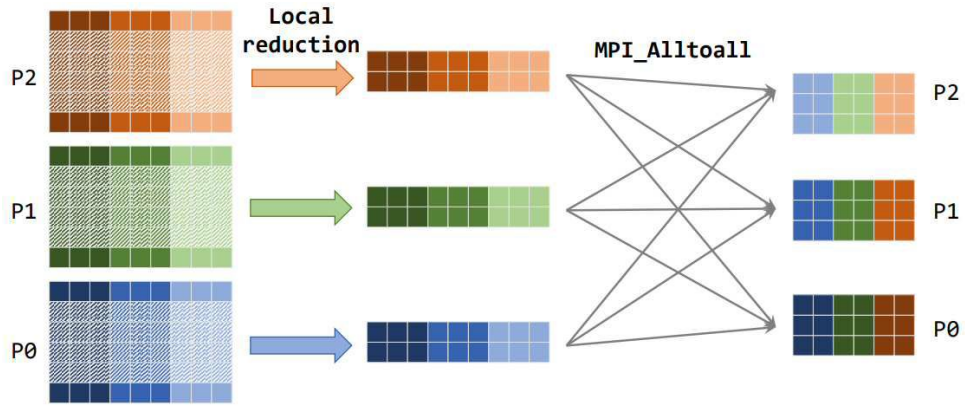
도면5



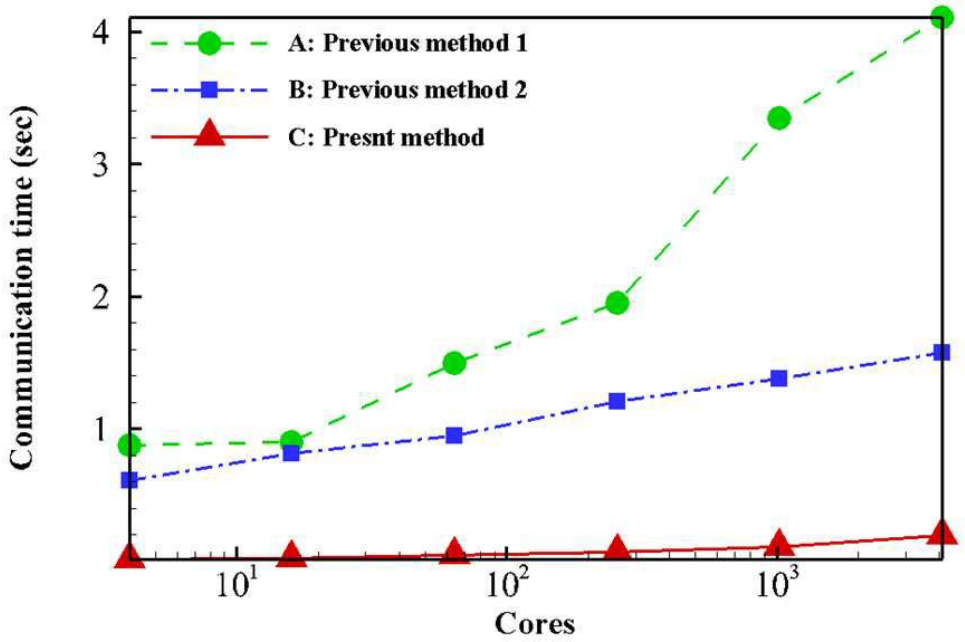
도면6


I. Lower/Upper element reduction step: creates a reduced tridiagonal system.
II. sub-TDMA step: Solve the reduced tridiagonal system.
III. Update step: Update solution of the original tridiagonal system
Communication(Forward)
Communication (Backward)

도면7


Local
reduction
MPI_Alltoall
P2
P1
P0
P2
P1
P0

도면8


A: Previous method 1
B: Previous method 2
C: Presnt method
Communication time (sec)
Cores
4
3
2
1
10^1
10^2
10^3