시스톨릭 구조 기반의 효율적인 양방향 중앙 병렬 가우스 소거법

이연규° 김학원 이광희 이충세
충 북 대 학 교 전 산 학 과
{iguana,hakwon,cleo,csrhee}@algo.chungbuk.ac.kr

An Efficient Central Two-Sided Parallel Gaussian Elimination Method Based on Systolic Structure

Yearn-Gui Yi Hak-Won Kim Kwang-Hee Lee Chung-Sei Rhee
Dept. of Computer Science, Chungbuk University

요 약

이 논문에서는 새로운 중앙 소거 방식과 행렬 이분법의 개념을 시스톨릭한 구조 위에 결합시켜 $O(N^3)$의 문제를 해결한다. 새로운 중앙 소거 방식은 주어진 시스톨릭한 구조의 병렬성을 최대한 증가시키는 것을 가능하게 해주며, 행렬 이분법은 기존의 가우스 소거 방식 상에서 나타나는 $O(N^2)$의 복잡도를 요구하는 후진 대입을 효과적으로 제거 시켜준다. 새로운 소거 방법은 독립적인 선형방정식으로 이루어진 시스템의 차수를 N이라 할 때 $2N(N+1)$의 저장 공간과 $4N+2\log_2 N-4$의 시간 복잡도를 갖는다. 제안 한 새로운 소거 방식은 단순한 구조와 연결 방식을 가진 그물 구조의 시스톨릭 병렬 시스템에 적용되기에 충분히 적합한 단순한 알고리즘을 사용하면서도 이전의 방법과 동일한 수준의 저장공간을 요구하고 동시에 훨씬 우수한 시간 성능을 나타내는 것이 가능하다.

1. 서론

$AX=B$의 형태를 이루는 독립적인 선형방정식들로 이루어진 시스템의 해를 구하는 문제는 자연과학과 공학 전반에 걸쳐 광범위하게 대두되는 중요한 문제 중의 하나이다. 그러나 이 문제에 대한 고전적인 해결 방법이라고 할 수 있는 피벗을 활용한 가우스 소거 방법은 N을 시스템의 차수라고 할 경우 $O(N^3)$의 상당히 높은 복잡도를 갖는 것으로 알려져 왔다[4]. 따라서 이러한 높은 복잡도를 줄이기 위한 수많은 연구가 이전에 행해졌으며 이러한 방법들 중의 하나가 가우스 소거 상에 존재하는 계산상의 특성과 우선 순위를 최대한 반영해 줄 수 있도록 특별하게 구성된 시스톨릭 구조의 병렬 프로세싱 시스템을 이용하는 것이다[4]. 이러한 시스톨릭 시스템과 결합된 알고리즘은 일반적인 시스템을 가정한 알고리즘보다 높은 성능을 나타내는 것이 가능하며 더욱이 발전된 현대의 소자 기술은 이러한 시스톨릭 시스템의 실현과 그 응용에 타당성을 제공해주고 있다.

이 논문에서는 시스톨릭 병렬 프로세싱 시스템의 병렬성을 최대한 살릴 수 있는 효과적인 수단으로서 새로운 중앙 소거 방식을 제안하고, 또한 가우스 소거 방식 상에 존재하는 $O(N^2)$의 복잡도를 요구하는 순차적인 후진 대입의 필요성을 제거하는 효율적인 해결책으로서 양방향으로 진행되는 행렬 이분법의 개념을 사용한다[1]. 새로운 소거 방법은 단순한 구조의 연결 형태를 갖는 그물 구조의 병렬 프로세싱 시스템과 결합되기에 충분히 적합할 정도로 단순한 기본 연산과 적은 저장 공간을 요구하면서도 이전의 연구에서 제시 된 방법들 보다 우수한 성능을 나타냄을 직접 확인할 수 있다.

이 논문은 다음과 같이 구성된다. 제2절에서는 새로운 알고리즘이 효과적으로 구현될 수 있는 시스톨릭 시스템의 구조를 제시하고 제3절에서는 새로운 중앙 소거 방식과 후진 대입을 제거하는 양방향으로 진행되는 행렬 이분법의 개념을 소개한다. 제4절에서는 앞선 절에서 제시된 논의들을 결합하여 발전된 새로운 가우스 소거 알고리즘을 제시하며 제5절에서는 새로운 알고리즘의 성능을 평가한다. 끝으로 제6절에서 결론을 맺는다.

2. 시스톨릭 구조

시스톨릭 구조와 결합 된 알고리즘이 일반적인 순차처리 구조 위에서의 알고리즘보다 좋은 성능을 나타내

기 위해서는 시스톨릭 시스템이 주어진 알고리즘의 특성을 적절히 반영할 수 있는 구조를 갖추어야만 한다. 따라서 이 절에서는 다음절에서 소개 할 새로운 가우스 소거 방법이 최대의 효과를 얻는 것이 가능하게 해줄 수 있는 시스톨릭 시스템의 단위 셀의 구조와 단위 셀 간의 연결 형태를 제안한다. 물론 단위 셀 간의 통신 부하를 줄이기 위해서는 단위 셀 당 많은 저장 공간과 복잡한 연결 구조를 갖는 것이 유리하지만 이것은 곧바로 비용 상승과 구조상의 복잡성을 유발하므로 결국 이 두 가지 요소 사이에 적절한 타협점이 존재한다는 것을 알 수 있다. (그림 1)은 일반적인 형태의 시스톨릭 시스템의 단위 셀의 구조를 나타낸 것이고 흰 삼각형은 처리 요소를 검은 사각형은 저장 요소를 추상화한 것이다.

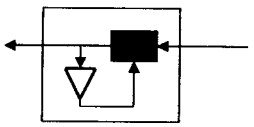


그림 1. 시스톨릭 시스템의 단위 셀 $p_{i,j}$의 구조

이 논문에서 제안하는 단위 셀의 구조는 그림 1의 일반적인 형태의 단위 셀의 구조상의 검은 사각형 즉 저장 요소에 다음과 같은 다섯 가지의 요소에 대한 저장 공간을 필요로 한다. 여기서 $p_{i,j}$는 행렬 요소 $a_{i,j}$와 일대일로 대응된다.

. $a_{i,j}$: 행렬 요소의 값을 저장

. $up_{i,j}$: 소거 시 바로 위에 인접한 행의 요소 값을 저장

. $\pi_{i,j}$: 소거 기준열의 피벗 비율 값을 저장

. $V_{i,j}$: 프로세서 $p_{i,j}$에서 수행한 수직 연산의 개수를 저장

. $H_{i,j}$: 프로세서 $p_{i,j}$에서 수행한 수평 연산의 개수를 저장

3. 중앙 소거와 양방향의 행렬 이분법

3.1. 양방향의 행렬 이분법

일반적인 가우스 소거 방식 하에서 나타나는 순차적인 $O(N^2)$의 복잡도를 요구하는 후진 대입을 제거하기 위한 효과적인 수단은 양방향으로 진행되는 행렬 이분법을 적용하는 것이다. 이것의 이론적 근거는 일반적인 선형 대수 이론에 의하면 동일한 해집합을 갖는 두 개의 독립적인 선형방정식으로 이루어진 시스템은 동일한 해집합을 갖는 하나의 선형방정식의 시스템과 동등하다는 것이다[1]. 따라서 주어진 N의 차수를 갖는 독립된 선형방정식들의 집합을 $N/2$의 차수를 갖는 동등한 두 개의 방정식들의 집합으로 적절한 소거 방식을 사용하여 연속적으로 분리해 나가는 것이다. (그림 2)는 임의의 소거 방식을 사용하여 4×5 행렬에 대하여 양방향으로 진행되는 행렬 이분법의 소거 과정을 단계적으로 나타낸 것이다. 최초에 주어진 선형방정식의 집합을 하나 더 복사한 후 독립적으로 전-후 양방향으로 소거를 진행시키고, 그 크기가 반으로 줄어들면 다시 복사와 소거를 독립적으로 반복하는 것이다. 따라서 일반성이 결여됨이 없이 시스템의 차수 N을 2의 거듭제곱 형태라고 가정한 경우 (그림 2)와 같은 양방향으로 진행되는 행렬 이분법에 의한 소거 방법은 $\log_2 N$ 번의 단계만에 끝나게 됨을 쉽게 알 수 있다.

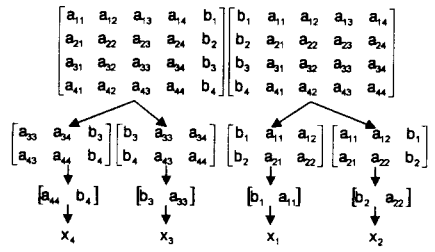


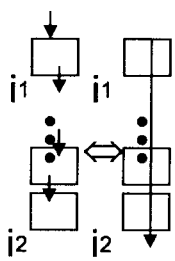
그림 2. 4×5행렬에 대한 양방향 행렬 이분법의 예

3.2. 중앙 소거법

앞 절에서 제안한 양방향의 행렬 이분법을 수행하는 과정에서 주어진 행렬을 반으로 줄이는데 사용하게 되는 양방향 행렬 이분법에 적절하게 결합될 수 있는 새로운 중앙 소거 방식을 소개한다. 새로운 중앙 소거 방식은 일반적인 소거 방식과는 다르게 피벗이 되는 기준열을 행렬의 중앙에서부터 좌-우측으로 이동시켜 가는 것이다. 이러한 피벗 기준열의 이동의 결과는 각 단계의 소거 마지막에 자연스럽게 행렬이 분리될 수 있게 해주며 동시에 다음에 제시되는 2가지의 기본 소거 연산이 최대한 파이프라인 방식으로 처리될 수 있게 해준다.

. $V(i_1, i_2, j)$: j열상의 i_1행과 i_2행 사이의 인접 행 들 사이에 병렬적으로 $a_{i-1,j}$ 를 $up_{i,j}$로 수직 복사하는 연산

. $H(i_1, i_2, j_1, j_2)$: i_1행과 i_2행 사이에 존재하는 피벗 비율 $\pi_{i,j}$를 j_1열에서 j_2열로 병렬적으로 수평 복사하는 연산



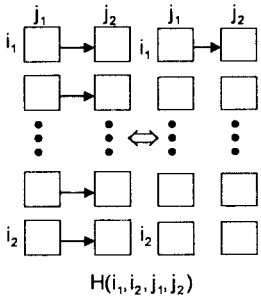


그림 3. 2가지 중요 기본 연산에 대한 예와 단순화 된 표현

(그림 4)에 행렬에 대해 중앙 소거 방식을 적용한 예를 단순화 된 표현을 사용하여 나타내었다.

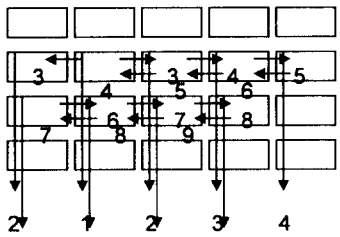


그림 4. 4×5 행렬에 대한 중앙 소거 방식의 예

4. 새로운 소거 알고리즘

이 절에서는 앞에서 제안한 중앙 소거 방식을 양방향 행렬 이분법과 결합시킨 효율적인 시스톨릭 알고리즘을 제시한다.

ALGORITHM1. 양방향 중앙 소거 방법

//여기서 m은 각 stage 에서의 시스템의 차수

for all processors p_{ij} **where** $1 \le i \le m, 1 \le j \le m+1$

```
for ( t=1; t ≤ 2m+1; t++ )
{ if ( ( t== | j-m/2+V_{i,j}-1 | +3V_{i,j}-2 )
       && ( i> V_{i,j} ) && ( V_{i,j} ≤ m/2 ) )
       { 수직 방향의 복사 연산 실행; }
   if ( ( t== | j-m/2+H_{i,j}-1 | +3H_{i,j}-1 )
       && ( i> H_{i,j} ) && ( m/2-H_{i,j}+1 ≠ 0 ) )
       { 수평 방향의 복사 연산 실행; }
   if ( ( t== j-m/2+4H_{i,j}-4 ) && ( i> H_{i,j}-1 )
       && ( H_{i,j} ≠ 1 )
       && ( m/2-H_{i,j}+1 < j ≤ m-H_{i,j}+2 ) )
       { 자리 이동을 위한 수평 방향의 복사 연산
         실행; }
)
```

5. 성능 평가

이 절에서는 앞 절에서 제시한 새로운 소거 알고리즘의 성능을 평가한다. 먼저 시간 복잡도를 구하기 위해 제안한 알고리즘의 특성을 살펴보면 다음과 같다.

. N을 시스템의 차수라고 할 경우 전체 소거 과정은 $\log_2 N$의 단계를 거친다.

.단계 s 에서의 차수는 $N/2^{s-1}$이 된다.

.단계 s 에서의 차수를 m이라 할 경우 단계 s에서의 총 처리 시간은 $2m+1$ 이 된다.

. 최초를 제외하고 각 단계마다 행렬 원소의 복사가 일어나므로 총 $\log_2 N-1$의 복사 연산이 수행된다.

. 마지막 단계에서 해를 구하기 위한 한 번의 나누기 연산이 수행된다.

이들 시간 요소를 종합하면 새로운 소거 방식에 의한 시간 복잡도는 다음과 같다.

$$\sum_{s=1}^{\log_2 N} (2\frac{N}{2^{s-1}}+1)+(\log_2 N-1)+1$$
$$=4N+2\log_2 N-4$$

알고리즘의 공간 복잡도는 독립적인 양방향 소거를 위해서 행렬의 복사를 해야하므로 시스템의 차수 N에 대하여 $2N(N+1)$의 공간 복잡도를 갖는 것이 자명하다.

6. 결론

이 논문에서는 가우스 소거 과정에서 나타나는 순차적인 후진대입을 제거시킬 수 있는 양방향으로 진행되는 행렬 이분법과 새로운 중앙 소거 방식을 결합하여 시스톨릭한 구조의 병렬 프로세싱 시스템에 적용시켰다. 새로운 소거 방법은 병렬적인 행과 열 단위의 단순한 기본 연산만을 수행하였기 때문에 기존의 $5N-\log N-4$ [1]보다 개선된 $4N+2\log N-4$의 시간 복잡도와 동일한 수준의 $2N(N+1)$의 공간을 요구함을 확인할 수 있었다.

참고문헌

[1] K.N. Balasubramanya Murthy, K.Bhuvaneswari, and C.Siva Ram Murthy, "A New Algorithm Based on Givens Rotations for Solving Linear Equations on Fault-Tolerant Mesh-Connected Processors," *IEEE Trans. Parallel and Distributed Systems*,vol.9, pp.825-832, Aug, 1998.

[2] Thomas Lippert, Armin Seyfried, Achim Bode, and Klaus Schilling, "Hyper-Systolic Parallel Computing," *IEEE Trans. Parallel and Distributed Systems*, vol.9, pp.97-108, Feb, 1998.

[3] Vipin Kumar et el., *Introduction to Parallel Computing*, The Benjamin/Cummings, 1994.

[4] Ellis Horowitz, Sartaj Sahni, and Sanguthevar Rajasekaran, *Computer Algorithms C++*, Computer Science Press., 1996