J. of Basic Sciences, Cheju Nat'l Univ.
15(1), 81~89, 2002

기초과학연구 제주대학교
15(1), 81~89, 2002

ET 영상 복원을 위한 병렬 및 분산 라이브러리의 구현 및 성능 측정

이 정 훈
제주대학교 전산통계학과
e-mail: *jhlee@cheju.ac.kr*

요 약

본 논문은 막대한 양의 계산을 필요로 하는 ET 영상 복원의 속도를 향상시키기 위하여 Matlab을 중심으로 하는 클러스터를 구축하고 위한 병렬 및 분산 계산 라이브러리를 작성한다. 병렬 계산 라이브러리는 이중 CPU 구조에 의해 곱하기, 역행렬 등 행렬 연산 속도를 증가시키도록 쓰레드와 동기화 기법을 사용하였으며 분산 계산 라이브러리는 운영체제에서 제공하는 프로세스간 통신 기능을 이용하여 구현된다. 이 라이브러리와 이상 유동장 특성에 기반하여 각 노드들이 효율적으로 메쉬 요소들을 그루핑하는 기법을 제시하고 그 성능을 평가한다. ET에서 주로 사용되는 776*776 차원의 행렬 연산에 대해 병렬 계산 함수들은 그 수행시간을 48% 까지 감소시킬 수 있어서 전체 복원 시간을 최대 42 %로 단축시킬 수 있다. 분산 계산은 물체 내부 이미지에 따라 속도 개선이 다르기는 하지만 주어진 영상에 대해 영상 복원 시간을 노드 수에 따라 기존 수행시간의 6 % 이내로 단축시킨다.

1. 서 론

ET(Electric Tomography), 즉 전기적 단층촬영 기법은 대상이 되는 물체에 인위적인 전기신호를 주입한 후 그 물체에서 나오는 신호를 분석하여 물체의 내부를 영상화한다[1]. 이 기법은 CT(Computerized Tomography)나 NMR (Nuclear Magnetic Resonance) 등 비하여 가격이 저렴하고 데이터의 취득속도가 빠르기 때문에 최근 새로운 단층촬영 기법으로 주목을 받고 있다. 그러나 영상 복원 과정에 있어서 행렬에 대한 곱하기, 역행렬, 의사 역행렬(pseudo inverse) 등 수많은 행렬 연산을 수행하기 때문에 계산량이 막대하여 복원 알고리즘을 수행하는데 있어서 상당한 시간이 소요된다. 이를 개선하기 위해서는 일차적으로 하드웨어 환경의 성능 증대와 아울러 효율적인 산술 계산 소프트웨어의 사용이 필수적이다[2].

하드웨어 기술의 발달은 컴퓨터들의 계산 속도를 현저히 증가시키고 있으며 현재 출시되는 PC들은 최대 4 개까지의 CPU를 탑재할 수 있디서 병렬처리를 위한 토대로 제공하고 있을 뿐 아니라 네트워크의 성능 향상은 클러스터 컴퓨팅과 같은 분산 계산을 가능하게 한다[3]. 그러나 ET 영상 복원 기법을 수행하는데 있어서는 하나의 정적 이미지를 복원하는데 현재의 PC에서 수십분 정도 소요되므로 하드웨어만의 기술 향상은 한계가 있으며 클러스터 컴퓨팅에 의해 속도를 개선한다 하더라도 하부 계산 라이브러리와 아울러 영상 복원 기법의 특성을 고려한 효율적인 분산 알고리즘의 개발이 절실히 요구된다.

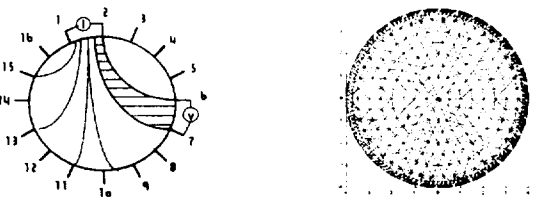
산술 계산을 효율적으로 수행할 수 있는 소프트웨어 도구들도 많이 개발되어 상용화되고 있으

며 ET와 같이 계산량이 막대한 응용에 사용되고 있다. 상용화된 소프트웨어 중에서 Matlab은 가장 널리 사용되는 계산 프로그램으로서 인터프리터에 기반한 명령처리 방식과 복잡한 데이터 표현방식 때문에 전반적인 수행속도가 늦어지는 단점을 갖고 있지만 곱하기, 역행렬 등 행렬의 기본 연산에 대해서는 가장 수행속도가 빠르다[4]. 따라서 ET 시스템과 같이 많은 행렬 연산을 포함하는 프로그램들은 Matlab으로 작성되어 있으며 그 속도가 상당히 우수하다. Matlab은 기본 명령어 이외에도 프로그래머들로 하여금 C 언어로 새로운 동적 링크 라이브러리를 작성할 수 있도록 하여 새로운 기능을 확장시킬 수 있다. 현재 Matlab은 단일 CPU 상에서 수행되도록 설계되어 있으며 이를 이중 CPU PC에서 수행시킨다 하더라도 속도 개선을 기할 수 없는데 병렬 및 분산 계산 기능을 구현함이 바람직하다.

본 논문에서는 ET와 같이 많은 양의 산술 계산을 수반하는 응용의 수행속도를 향상시키기 위해 이중 CPU PC 상에서 Matlab의 기본연산, 즉 행렬 곱하기, 역행렬 계산, 의사 역행렬 계산 등을 병렬로 수행하는 라이브러리 프로그램을 구현한다. 이와 아울러 분산 계산을 지원하기 위하여 여러 노드에 분산되어 수행되는 Matlab 응용들이 서로 통신하여 자료구조를 교환하는 프로그램을 작성하여 ET 계산을 위한 협력계산 환경을 구현한다. 즉 Matlab 계산 소프트웨어를 중심으로 클러스터를 구축함으로써 기존의 Matlab 프로그램들이 쉽게 분산 프로그램으로 작성 혹은 이식되도록 한다. 이를 기반으로 그루핑 기법을 제안함으로써 각 노드들이 협력 계산에 의해 행렬 원소의 수를 감소시키도록 하여 영상 복원의 속도를 향상시킨다.

본 논문의 구성은 다음과 같다. 1장에서 문제를 제기한 후 2장에서는 ET 기법의 특성과 계산 속도 향상을 위한 기존의 연구를 소개한다. 3장에서는 Matlab 라이브러리의 작성에 관련된 구현 내역을 상세히 설명한 후 4 장에서는 ET 영상 복원 속도의 향상도를 측정한다. 마지막으로 5장에서는 본 논문을 정리하고 추후 연구과제를 도출한다.

2. 연구 배경 및 관련 연구



(a) 외부 전극 (b) 메쉬 요소

그림 1. ET의 원리

ET는 비파괴 단층촬영 기법으로서 그림 1(a)에서 보는 바와 같이 물체 외부에 배치된 전극을 통해 전류를 순차적으로 주입하고 각 주입된 전류에 대해 다른 전극에서 전압을 측정한다[5].

이 과정에서 그림 1(b) 에서처럼 물체 내부를 가상적인 요소(element)들로 분할한 후 측정된 전압을 기준으로 각 요소들의 저항값을 계산하여 물체 내부의 상태를 파악한다. 각 요소들이 미지의 변수가 되며 전극에서 측정된 데이터들은 변수의 값을 계산하는데 필요한 방정식을 제공한다. 그림에서는 16개의 전극을 사용하는 예를 보이고 있는데 전극의 수가 많아지면 방정식이 많아져서 요소의 수를 증가시킬 수 있으므로 복원 영상의 해상도를 향상시킬 수 있다. 그림 1(b)는 776개의 요소들로 구성된 메쉬 분할을 보이고 있는데 결국 ET 영상복원 프로그램은 776개의 미지수에 대한 연립방정식을 푸는 문제로 귀착된다. 이 과정에서 요소의 개수는 계산 시간에 큰 영향을 주며 외부 전극의 개수와 해상도, 속도 요구사항에 의해 결정된다. Newton-Raphson, 동적 칼만 필터와 같은 영상 복원 알고리즘은 요소의 수를 n이라 할 때 $n \times n$ 2차원 행렬에 대한 자코비언, 행렬 곱하기 및 역행렬 계산 등으로 구성된다.

따라서 ET 시스템에 있어서는 이러한 산술 계산을 효율적으로 수행할 수 있는 하드웨어 혹은 계산 소프트웨어가 필수적이다.

영상 복원 알고리즘은 반복적인 루프로 구성되며 있으며 한 루프를 프레임이라고 한다. 한 프레임은 하나의 전극에서 주입된 신호를 다른 전극에서 측정한 결과를 기반으로 각 요소의 값을 결정하는 것이며 한 프레임의 계산 결과가 다음 프레임의 계산에 인자로 사용된다. 따라서 계산 시간을 단축시키려면 CPU의 성능을 개선하거나 요소의 수를 감소시켜야 한다. 요소의 수를 감소시키려면 루프의 초기화 혹은 진행에 따라 같은 특성을 갖는 요소들을 추출하여 하나의 요소로 결합하는 그루핑 기법이 가장 효율적인데 같은 이의 오버헤드가 전체 성능에 영향을 주게 되지만 분산 계산에 의해 이를 최소화할 수 있다. 본 논문에서 대상으로 하고 있는 원자력 발전소의 열수력 시스템의 경우 물체 내부는 물과 기포, 오직 두 상태만 갖는데 이와 같이 물체 내부가 오직 두 상태만 갖는 경우를 이상 유동 모델(two-phase flow model)이라고 한다[6]. 이러한 모델에서 각 프레임 계산에서 모든 요소의 값을 계산하기보다는 사전 분석에 의해 물로 판명된 요소들을 동일 그룹으로 결합하여 하나의 요소인 것으로 간주한다면 계산속도를 개선할 수 있다. 이 사전분석 과정의 수행은 추가적인 낭비시간이지만 데이터와 처리 과정의 독립성이 강하므로 분산 계산에 의해 그 시간을 단축시킬 수 있으며 전체적인 복원 시간을 단축할 수 있다.

3. 구 현

3.1 구현환경

Matlab은 프로그래머로 하여금 새로운 라이브러리를 구현하도록 하기 위하여 mex 도구를 지원하고 있으며 이는 Matlab의 명령과 DLL (Dynamic Link Library) 사이의 인터페이스를 담당하여 인자들이 전달되고 수행 결과를 받을 수 있도록 한다. Matlab에서 DLL을 작성하기 위해서는 다음과 같은 함수를 C 언어로 작성한 후 mex 유틸리티를 이용하여 컴파일한다. 이후 새로운 명령이 추가되어 인터프리터 혹은 M-file[3]에서 새로운 함수를 호출할 수 있다.

```
void mexFunction ( int nlhs, mxArray *plhs[],
                   int nrlhs, mxArray *prhs[]) {
        ..... 처리부분 .....
}
```

mexFunction의 인자들은 입력 및 출력 인자의 수와 더불어 Matlab에서 정의하고 생성한 mxArray 타입의 행렬을 포함한다. mxArray 구조체는 그림 2에서 보는 바와 같이 일반적인 크기의 행렬을 저장하기 위하여 행의 수, 열의 수 및 데이터에 대한 포인터 등을 갖고 있다. 데이터 영역에는 행렬의 각 원소들이 열 우선(column major) 방식으로 저장되어 있다.

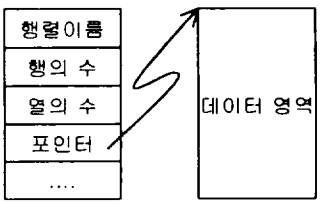


그림 2. Matlab의 mxArray 구조체

3.2 병렬 계산 라이브러리의 구현

인자로 넘어온 행렬을 이중 CPU를 이용하여 계산하려면 먼저 행렬을 분할하고 윈도우 운영체제에서 지원하는 함수를 사용하여 쓰레드를 생성한 후 각 쓰레드로 하여금 할당된 작업을 수행하도록 한다[7]. 쓰레드는 병렬 프로그램의 실행 단위로서 프로그램은 다수의 프로그램 흐름을 갖게 된다. 쓰레드 프로그램을 작성하는데 있어서 windows.h 헤더 파일에 정의되어 있는 CreateThread 등의 함수를 사용한다. 분할 과정

3) Matlab의 스크립트 파일

은 mxArray의 포인터를 변경하여 낭비시간이 없이 수행될 수 있다. 이중 CPU PC는 각 CPU가 버스를 통해 공유 메모리에 접근할 수 있는 다중처리기(multiprocessor)이므로 쓰레드들은 전역변수들을 공유할 수 있어서 쓰레드간 통신에 필요한 오버헤드는 최소화된다. 그러나 쓰레드 생성과 동기화에 따르는 낭비시간은 불가피하게 발생하므로 계산이 단순한 행렬의 더하기와 빼기 등은 낭비시간 때문에 병렬 수행을 하더라도 속도가 개선되지 않는다.

각 CPU에 할당된 계산의 수행을 완료하면 이 중간결과들은 하나의 행렬로 결합되어야 한다. 쓰레드의 종료를 기다리기 위해 윈도우에서 제공하는 WaitforSingleObject 동기화 함수를 이용하여 쓰레드의 종료를 기다린다. 결과를 합성하는 과정에서는 Matlab 라이브러리의 오버헤드를 최소화하기 위하여 memcpy와 같은 C 언어 함수를 사용하여 고속의 메모리 복사를 수행한다.

A*B와 같은 행렬의 곱하기를 병렬로 수행하려면 그림 3에서 보는 바와 같이 먼저 한 행렬을 둘로 분할하여야 한다. B 행렬을 분할하는 이유는 Matlab에서 행렬을 표현할 때 열우선으로 각 항목을 저장하는기 때문이며 연속적인 항목을 나누는 것이 효율적이다. 이는 B1, B2 두 개의 mxArray 구조체를 생성한 후 이들의 포인터가 각각 B 행렬의 처음 부분과 중간 부분을 가리키도록 하면 된다. 분할 후 각 쓰레드는 부분곱을 수행하며 각 쓰레드는 mlfMtimes라는 Matlab 라이브러리 함수를 사용한다. 이 함수는 곱하기를 효율적으로 수행하는 함수로서 그 결과를 C1, C2 새로운 행렬을 위한 메모리를 할당하여 저장한다. 이들을 결합하기 위해서는 C 행렬을 위한 공간을 A의 행, B의 열을 인자로 mxCreateMatrix 함수를 호출하여 할당한 후 C1과 C2의 데이터 부분을 C의 데이터 영역에 복사한다. 결국 병렬 곱하기는 그 계산 과정에서 많은 메모리를 필요로 하며 데이터 복사가 성능저하 요인으로 작용할 수 있지만 데이터의 독립성이 강하므로 병렬 계산의 효율을 기대할 수 있다.

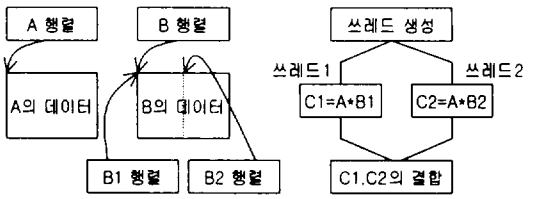


그림 3. 병렬 곱하기를 위한 부분행렬과 쓰레드

역행렬을 병렬로 계산하려면 우선 주어진 행렬을 분할하여야 하는데 이 과정에서 병렬 수행의 장점을 극대화하려면 연속된 공간의 분할 즉, 사각형 형태의 분할이 바람직하다. 더욱이 데이터가 공유되므로 데이터의 의존성을 효율적으로 처리할 수 있다. 따라서 그림 4와 같은 분할 방식을 기반으로 병렬 역행렬 계산을 구현한다[8].

$$A=\begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix} \quad A^{-1}=\begin{bmatrix} G & -G\,A12\,A22' \\ -A22'\,A21\,G & A22'+A22'\,A21\,G\,A12\,A22' \end{bmatrix}$$

$$G=(A11-A12*A22'*A21)'$$

그림 4. 병렬 역행렬 계산을 위한 행렬 분할

그림 4의 분할 방식은 하나의 역행렬 계산을 행렬의 차원이 반으로 감소한 행렬에 대해 7 번의 역행렬, 10 번의 곱하기 및 2 번의 더하기 혹은 빼기 연산으로 구성되는데 중복된 계산이 많으므로 연산의 수는 줄어든다. A22에 대한 역행렬을 계산한다면 임시적으로 생성된 행렬을 각 쓰레드가 공유하므로 중복 계산을 감소시킬 수 있다. 즉, A22'를 한번 계산해 놓으면 이후 계속 사용할 수 있으며 A22'*A21*G의 결과도 추후 사용할 수 있는 장점이 있다. 따라서 두 번의 역행렬 계산과 5 번의 곱하기로 줄어든다. 행렬 연산의 수행 속도는 행렬의 원소 수에 크게 영향을 받는데 행렬의 차원이 반감되었기 때문에 분할의 효과만으로도 속도의 향상을 기할 수 있다. 부분 행렬의 역행렬 계산은 Matlab의 mlfInv 함수를 사용하는 반면 행렬의 곱하기는 구현된 병렬 곱하기를 이용하여

계산 속도를 향상시킨다.

의사 역행렬은 정방행렬이 아닌 일반 행렬 A에 대해 AMA=A, MAM=M을 만족시키는 M 행렬을 구하는 것으로서 무어와 펜로즈에 의해 존재성과 유일성이 증명되었다. ET에서 사용되는 행렬들은 모두 full-rank인데 이 경우는 계산식이 식(1)과 같이 유도된다[9].

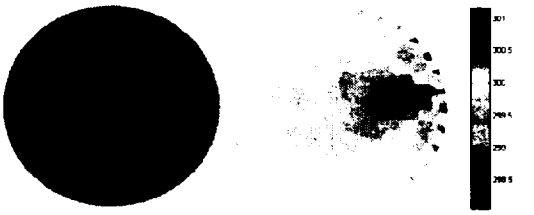
$$A=[\ X:\ Y\ ] \qquad L=X^{*}\ [\ I\ \ X'^{*}Y\ ]$$
$$M=L^{T}{}^{*}(A^{*}L^{T})' \qquad (1)$$

위 식에서 보는 바와 같이 의사 역행렬 계산은 정방 행렬 X와 $(A^{*}L^{T})$에 대한 역행렬 계산과 곱하기로 분할되어 계산된다. 전치행렬 계산은 Matlab의 mlfTranspose를 사용하는 반면 곱하기와 역행렬 계산은 앞에서 구현된 병렬 라이브러리를 이용하여 의사역행렬을 구한다.

3.3 분산 계산의 구현

네트워크로 연결된 노드, 즉 Matlab을 수행하는 PC 혹은 CPU[4]들이 서로 행렬들을 교환할 수 있으려면 운영체제에서 사용하는 통신 라이브러리와 함께 컴파일하여야 하는데 송신자는 전송할 데이터 영역을 추출하는 반면 수신자는 미리 데이터 영역을 할당한 후 그 영역에 네트워크를 통해 수신된 데이터를 채워 넣는다. 이를 위해 수신자 측의 DLL 루틴은 입력으로 수신될 Matlab 행렬의 차원을 받아야 한다.

Windows 계열의 운영체제들은 다양한 통신 함수들을 제공하는데 ET 시스템은 그 특성상 신뢰성있는 통신을 요구하므로 NT 운영체제에서 사용되는 Pipe 기능을 사용한다[10]. 이 통신 DLL 함수는 연결의 초기화 및 연결을 통한 읽기, 쓰기 및 연결 해제 등의 기능을 제공한다. 한 노드가 여러 노드와 연결되기 위해서는 각 연결에 대한 자료구조를 Matlab 내에 관리하여야 하며 통신을 원하는 노드에 해당하는 연결을 선택하여 통신한다. 구현된 Matlab 명령어와 인자는 다음과 같이 사용된다. 한 연결에서 클라이언트로 동작하는 노드는 그 연결을 이용한 통신에서 계속 클라이언트 함수를 호출하여야 한다.



(a) 대상 물체 (b) 초기 루프의 계산

그림 5. 그룹 형성의 기본 특성

동적 칼만 필터 알고리즘은 요소의 수에 따라 결정되는 여러 2차원 배열을 자료구조로 갖고 있

표1. 함수 사용의 예

클라이언트 함수	서버 함수	비고
p1 = cpipe(1, 컴퓨터, 연결 번호)	p1 = spipe(1)	서버는 블로킹 호출
cpipe(2, p1, mxArray)	spipe(2, p1, mxArray)	전송
cpipe(3, p1, mxArray, row, col)	spipe(3, p1, mxArray, row, col)	수신시 차원을 넘김
cpipe(4)	spipe(4)	연결 해제

4) 다중 CPU를 탑재한 PC에서 Matlab을 동시에 수행시키는 경우를 포함

으며 이들은 루프가 진행되는 동안 이전 루프에서 계산된 값, 해당 루프에서의 값의 변화, 그리고 다양한 중간 계산 결과들을 포함한다[11,12]. 처음 루프는 이전 계산결과가 없기 때문에 수행시간이 가장 빠를 뿐 아니라 그룹을 정하는데 있어서 공정한 데이터를 제공할 수 있다.반면 두 번째 루프이후는 이전 프레임 계산의 잔영이 남아있게 된다. 실제 구현된 프로그램의 수행속도를 측정한 바에 의하면 초기 루프와 이후 루프의 수행시간은 약 3~4배 정도 차이가 난다.

그림 5(a)는 대상 물체 내부의 한 가운데에 기포가 하나 있는 경우이며 그림 5(b)는 처음 루프를 수행한 후의 결과이다. 그림 5(b)의 오른쪽 표면에서부터 적색이 퍼지고 있는데 이 곳에 1번 전극이 위치하고 있다. 초기 루프를 수행하고 나면 1번 전극부터 기포를 연결하는 가상의 직선(사실은 곡률이 작은 곡선) 부근은 어떤 물체가 있음을 나타내고 있으며 그 이외의 부분은 물로 보여지고 있다. 이 부분은 이후 루프에서 다른 값으로 변했다가 최종적으로 물의 값으로 수렴하는데 이 과정을 배제하고 물 그룹에 속하도록 하면 속도가 개선될 수 있다. 각 프레임 계산을 위한 데이터는 고속으로 수집되며 분석 과정에서 어는 프레임을 먼저 계산하여도 무방하다.

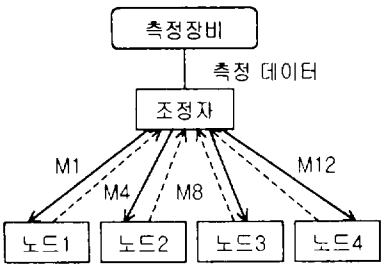
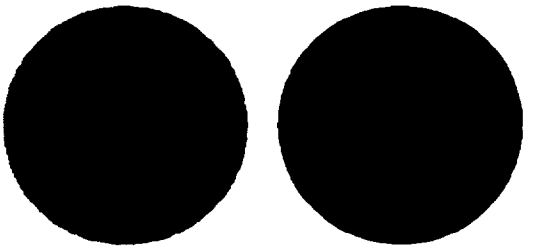
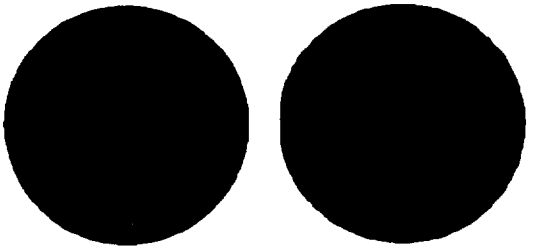


그림 6. 분산 계산에 의한 사전 분석

그룹의 크기를 증가시키려면 여러 프레임에 대해 초기 루프를 수행시켜야 하는데 이는 전체적인 영상복원 시간을 증가시킬 수 있다. 따라서 그림 6에서 보는 바와 같이 조정자가 수집된 계산 결과를 구현된 통신 DLL을 통해 각 노드에게 전송하고 각 노드는 자신에게 할당된 프레임을 초기 루프로써 수행한 다음 이에 의해 물로 판명된 요소들을 보고하도록 한다. 조정자 노드가 수합하여 하나의 그룹으로 결합한다면 계산될 요소의 수가 감소한다. 이후 조정자 노드에서 영상 복원 알고리즘을 수행한다면 그룹 과정 없이 수행하는 것보다 속도의 향상을 기할 수 있다. 특히 대부분의 연산이 복잡도가 $O(n^3)$인 행렬 곱하기, 역행렬 계산이므로 n 값은 조금만 감소하여도 수행시간의 차이는 크게 나타난다. 이때 통신 오버헤드는 단지 한 프레임의 특정 데이터이므로 각 조정자와 노드간 2 Kbit의 정보를 교환하면 되는데 전체적인 수행시간에 비해 그 크기가 상대적으로 작다. 또 노드의 수가 증가하여도 통신의 오버헤드만 약간 추가되므로 사전 분석 과정의 오버헤드 증가는 없다.



(a) 그룹 1(노드=1)　(b) 그룹 2(노드=2)



(c) 그룹 4(노드=4)　(d) 그룹 4(노드=8)

그림 7. 결합된 그룹의 크기

그림 7은 그룹의 결과를 보이고 있는데 그룹 1은

그림 7(a)에서 보는 바와 같이 전체 16개의 전극 중 1번 전극의 프레임에 대해서만 초기루프를 수행하여 물로 판명된 요소들만 표현한 것이다. 그룹 2는 2 개의 노드가 1, 8번 프레임을, 그룹 3은 4 개의 노드가 1, 4, 8, 12 프레임을, 그룹 4는 8 개의 노드가 8 개의 프레임에 분산 계산에 의해 물을 하나의 그룹으로 묶은 것이다. 그림에서 짙은 부분이 그룹으로 묶인 부분이며 그림 7(d)의 그룹 크기가 그림 7(a)보다 크다. 원래 776 개이던 요소의 수는 각 그룹 계산에 의해 507, 267, 235, 196 개로 감소하며 이에 의해 영상 복원 알고리즘의 수행속도는 표 1에서 보는 바와 같이 개선된다. 물론 내부 데이터의 특성에 따라 본 논문에서 제시하는 기법의 성능은 영향을 받는다. 예를 들어 여러 기포들이 전극 주변에 위치하고 있다면 오히려 그룹을 계산하는데 필요한 시간이 낭비 시간으로 작용할 수 있다.

4. 성능 측정 및 분석

그림 8, 9, 10 은 각각 Matlab과 병렬 라이브러리의 곱하기, 역행렬 및 의사 역행렬에 대한 수행시간을 보여주고 있다. 프로그램이 수행된 컴퓨터는 2 개의 Pentium III CPU와 128 M의 메모리를 갖고 있다. n을 차원이라 할 때 n은 100부터 1500까지 변화한다. 각 행렬은 난수를 이용하여 발생시켰으며 의사 역행렬인 경우 $(n-1)*n$의 행렬이 생성된다. 구현된 라이브러리는 곱하기의 경우 두 개의 n=400에서 69 %로, 역행렬의 경우 역시 n=400 에서 34.8 %로, 의사 역행렬의 경우에는 n=500, 즉 499*500에서 52 %로 수행시간을 최대로 감소시켰다. 또 ET 시스템에서 주로 사용되는 800*800 행렬에 대해서는 곱하기와 역행렬의 수행시간이 63.2 %와 54.4 %로 단축된다. 곱하기의 경우에는 데이터 복사에 따르는 낭비시간 때문에 50 % 이하로 감소되지는 않는 반면 역행렬과 의사 역행렬인 경우에는 병렬 계산과 아울러 분할 계산의 효과도 성능향상에 기여한다.

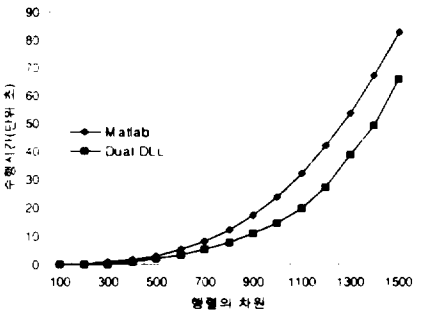


그림 8. 행렬 곱하기의 수행시간

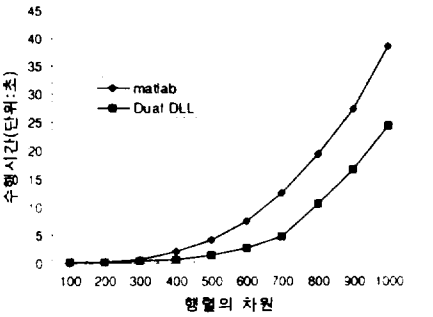


그림 9. 역행렬 계산 시간

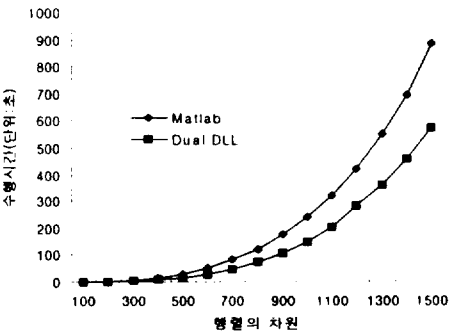


그림 10. 의사 역행렬 계산 시간

본 논문에서 구현된 병렬 곱하기 방식은 Matlab과 동일한 계산 결과를 수행하므로 정확성에 오차가 없다. 또 역행렬과 의사 역행렬인 경우에는 A22 부분행렬이 역행렬을 가질 수 있어야 해가 존재하며 피보팅이 부분 행렬 내에서만 이루어지기 때문에 오차를 포함할 수 있다. 그러나 주어진 행렬과 ET 시스템에서 계산하는 행렬에 대해 기존의 Matlab 코드와 정확성 분석 결과는

$O(10^{-16})$으로 계산 결과의 정확성에 영향을 주지 않는다[13].

표 2. 병렬 계산에 의한 성능 향상

루프	기존방식	성능 개선	수행시간비
1	25.140	8.719	34.68%
2	70.313	29.75	42.31%
3	65.719	29.578	45.01%
4	65.453	29.578	45.19%
5	65.422	29.563	45.19%
6	65.359	28.991	44.36%
7	65.531	30.025	45.82%
8	65.391	29.687	45.40%
9	65.422	29.625	45.28%
10	65.187	29.266	44.90%

표 3. 분산 계산에 의한 성능향상 (단위:초)

	비그룹	그룹4	그룹3	그룹2	그룹1
루프 1	8.313	1.985	2.453	2.661	4.156
루프 2	32.812	1.937	2.469	3.359	15.344
루프 3	35.329	1.938	2.485	3.375	15.328
루프 4	29.796	1.937	2.484	3.375	15.344
루프 5	29.118	1.953	2.469	3.360	15.328

5. 결론

본 논문에서 구현된 행렬의 곱하기, 역행렬 및 의사 역행렬은 막대한 산술 계산을 수반하는 응용의 수행 속도를 증가시킬 수 있다. 영상 복원에 있어서 이들 연산을 많이 사용하는 ET 시스템의 경우 영상 복원 시간이 48 %로 감소하였다. 이러한 계산 구조는 CPU가 3 개 이상으로 확장된 경우에도 쉽게 적용될 수 있을 뿐 아니라 네트워크를 통해 여러 개의 컴퓨터가 연결된 경우에도 효율적으로 계산을 수행할 수 있다. 또 Matlab을 수행하는 각 노드들이 분산 계산을 수행할 수 있는 환경을 구축하고 프레임의 사전 분석을 각 노드에서 수행하고 조정자 노드로 하여금 이를 수합하도록 함으로써 영상 복원의 속도를 개선하였다. 향후 다양한 기포 분포에 대한 실험이 진행될 것이며 구축된 분산 환경을 기반으로 효율적인 영상복원을 위한 분산 알고리즘을 구현하여 영상 복원 속도를 증진시킬 것이다.

추후 과제로서는 Matlab에 의하지 않는 클러스터를 구축하고 MPI, CLAPACK과 같은 라이브러리를 이용하여 보다 많은 수의 노드를 영상 복원 계산에 참여시키는 방안과 아울러 이 클러스터 구조에 적합한 분산 계산 알고리즘의 개발이 진행될 예정이다.

참고문헌

[1] 과학기술부, 「이상유동장 가시화를 위한 ET 기법 개발에 관한 연구」, 2001년 7월

[2] W. Furmanski, "Petaops and exaops: supercomputing on the Web," *IEEE Internet Computing*, pp.38-46, 1997.

[3] R. Buyya, *High Performance Cluster Computing*, Vol.2, PTR PH, 1999.

[4] http://www.mathworks.com

[5] K. H. Cho, S. Kim, Y. J. Lee., "Impedance imaging of two-phase flow field with mesh grouping algorithm," *Nuclear Eng. & Design*, pp.18-26, 2001.

[6] Minchan Kim, Sin Kim, Kyeongyeon Kim, Junghoon Lee, Yoonjoon Lee, "Reconstruction of particle concentration distribution in annular Coutte flow using electrical impedance tomography," *Journal of Industrial Engineering Chemistry*, Oct. 2001.

[7] S. Matthew, *Windows 2000 Server*, Sybex, 2001.

[8] S. R. Searle, *Matrix Algebra Useful for Statistics*, John Wiley & Sons, 1982.

[9] K. Gallivan, et. al., *Parallel Algorithms for Matrix Computations*, SIAM, 1990.

[10] J. Richter, *Advanced Windows*, MS Press., 1996.

[11] M. Cheney, et. al., "Electrical impedance tomography," *SIAM Review*, No. 41, pp.85-101, 1999.

[12] M. Vauhkonen, at. al, "A Kalman filter approach to track fast impedance changes in eletrical impedance tomography," I*EEE Trans. on Biomedical Engineering*, pp.486-493, 1998.

[13] 김철민, 이정훈, "이중 CPU PC에서 병렬 계산을 위한 Matlab 행렬 연산 라이브러리의 구현 및 성능 측정," 『정보과학회 추계학술대회(Ⅲ)』, pp.871-873, 2001년 10월.

An implementation and performance measurement of parallel and distributed computing libraries for image reconstruction based on the electrical tomography

Junghoon Lee
Dept. of Computer Science and Statistics,
Cheju National University, Jeju, 690-756

This paper builds a Matab-based cluster and implements parallel and distributed computing libraries, aiming at enhancing the execution speed of ET image reconstructions algorithm which requires an enormous amount of CPU computation. The parallel computing functions are implemented using thread and its synchronization utilities provided by Windows operating system on top of a dual CPU PC platform on order to improve the computation speed of primitive matrix operations such as multiplication, inverse and so on. Additionally, the distributed computing function enables each Matlab application performing its job on a discrete computer to exchange its data. With these libraries and the characteristics of two phase model of ET, a cooperative grouping scheme is proposed and its performance is also measured. For the 776*776 two dimension arrays, the typical array used in ET, the parallel function can reduce the computation time upto 48 % at maximum, resulting in the final 42 % reduction of overall reconstruction time. In addition, for the given internal structure, the cooperative distributed grouping scheme achieves the time reduction upto 6 %.