이중 CPU PC에서 병렬 계산을 위한 Matlab 행렬 연산 라이브러리의 구현 및 성능 측정

김 철 민 이 정 훈∪
제주대학교 컴퓨터교육학과 제주대학교 전산통계학과
E-Mail :{cmkim@educom, jhlee@venus1}.cheju.ac.kr

An implementation and performance measurement of Matlab matrix operation library for parallel computing on dual CPU PC

Cheolmin Kim, Junghoon Lee∪
Dept. of Computer Science and Statistics, Cheju National University

요 약

본 논문에서는 전기 단층 촬영 기법과 같이 많은 양의 데이터에 대해 산술 계산을 수행하는 응용의 수행속도를 개선하기 위하여 이중 CPU PC 상에서 Matlab의 기본연산, 즉 행렬 곱하기, 역행렬 계산, 의사 역행렬 계산 등을 병렬로 수행하는 라이브러리 프로그램을 구현하고 그 성능을 측정한다. 구현된 라이브러리는 행렬의 곱하기, 역행렬 계산, 의사 역행렬 계산 등 기본적인 행렬 연산에 대해 각 CPU에서 수행될 쓰레드를 생성하고 이 쓰레드에 분할 행렬을 인자로 넘겨줌으로써 병렬 계산을 실행하도록 하고 부분 결과를 합성하여 최종적인 결과를 산출하게 된다. 구현된 코드를 수행시켜 속도를 측정한 결과 행렬의 곱하기는 최대 69 %, 역행렬은 34.8 %, 의사 역행렬은 52 % 까지 수행시간을 단축시켰다. 이에 의해 전기 단층 촬영 프로그램은 한번의 전류 주입에 대해 영상 복원에 소요되는 시간을 48 %로 감소시켰다.

1. 서론 및 배경

컴퓨터의 계산 능력이 향상됨에 따라 막대한 산술 계산을 요구하는 응용들이 일반화되어 PC와 같은 환경에서 수행되고 있다[1]. 본 논문에서 수행속도를 개선하고자 하는 ET(Electrical Tomography)는 그 한 예로서 대상이 되는 물체 내부에 인위적인 전기 신호를 주입한 후 그 물체에서 나오는 신호를 분석하여 물체 내부의 상태를 영상화한다[2]. 이 응용은 그 분석과정에서 행렬에 대한 곱하기, 역행렬, 의사 역행렬(pseudo inverse) 등 수많은 행렬 연산을 수행하며 보다 정확하고 신속한 분석을 위해 기본적인 행렬 연산의 고속화를 필요로 한다. 이와 같은 응용의 계산 속도 향상 요구를 만족시키기 위하여 최근의 PC들은 최대 4 개까지의 CPU를 탑재하여 병렬 계산을 지원할 뿐 아니라 Window NT 이상의 운영체제에서는 프로그래머가 지정한 각 쓰레드를 각 CPU에게 할당하여 계산 속도의 향상을 기한다[3].

산술 계산을 효율적으로 수행할 수 있는 소프트웨어 도구들도 많이 개발되어 상용화되고 있으며 최근 Java에서도 산술 계산의 중요성을 인식하여 이에 대한 연구단체가 결성되었다[4]. 상용화된 소프트웨어 중에서 Matlab은 가장 널리 사용되는 계산 프로그램으로서 인터프리터에 기반한 명령처리 방식과 복잡한 데이터 표현방식 때문에 전반적인 수행속도가 늦어지는 단점을 갖고 있지만 행렬의 기본 연산에 대해서는 가장 수행속도가 빠르다[5]. 따라서 ET 시스템과 같이 많은 행렬 연산을 포함하는 프로그램들은 Matlab으로 작성되어 있으며 그 속도가 상당히 우수하다. 이와 아울러 효율적인 행렬 계산을 수행하는 C 라이브러리를 제공하여 프로그래머들로 하여금 새로운 기능을 추가할 수 있도록 하고 있다.

현재 Matlab은 단일 CPU 상에서 수행되도록 설계되어 있으며 이를 이중 CPU PC에서 수행시킨다 하더라도 속도 개선을 기할 수 없다. 따라서 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 구현된 라이브러리의 수행속도를 측정하여 기존의 Matlab에서의 수행속도와 비교한다.

2. 라이브러리의 구현

2.1 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 mex를 이용하여 컴파일한다.

```
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 1에서 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.

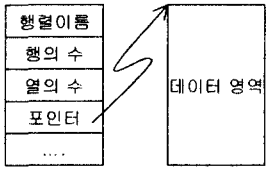

그림 1. Matlab의 mxArray 구조체

처리 부분에서는 인자로 넘어온 행렬을 분할하고 윈도우 운영체제에서 지원하는 함수를 사용하여 쓰레드를 생성한 후 각 쓰레드로 하여금 할당된 작업을 수행하도록 한다[6]. 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 과정은 mxArray의 포인터를 변경하여 낭비시간이 거의 없이 수행될 수 있다. 이중 CPU PC는 각 CPU가 버스를 통해 공유 메모리에 접근할 수 있는 다중처리기(multi-processor)이므로 쓰레드들은 전역 변수들을 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.

2.2 병렬 곱하기

A*B와 같은 행렬의 곱하기를 병렬로 수행하려면 그림 2에서 보는 바와 같이 먼저 한 행렬을 둘로 분할하여야 한다. B 행렬을 분할하는 이유는 Matlab에서 행렬을 표현할 때 열우선으로 각 항목을 저장하는데 때문이며 연속적인 항목을 나누는 것이 효율적이다. 분할 후 각 쓰레드는 부분곱을 수행하며 각 쓰레드는 mlfMtimes라는 Matlab 라이브러리 함수를 사용한다.

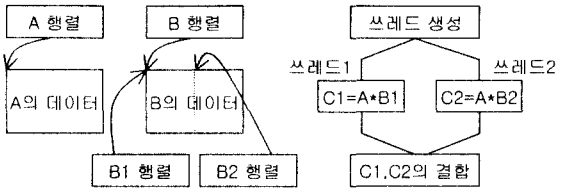

그림 2. 병렬 곱하기를 위한 부분행렬과 쓰레드

이 함수의 결과는 mxArray 타입의 임시 구조체에 저장되는데 쓰레드들이 종료하면 이를 결합하여 새로운 구조체를 생성하여야 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. 물론 Matlab 라이브러리를 사용하지 않고 C 함수를 이용하여 부분 곱을 수행한다면 메모리를 프로그래머가 원하는 대로 할당할 수 있지만 곱하기의 수행속도가 현저히 저하된다.

2.3 역행렬 계산

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 3과 같은 분할 방식을 기반으로 병렬 역행렬 계산을 구현한다[7].

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

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

그림 3. 병렬 역행렬 계산을 위한 행렬 분할

그림 3의 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 2.2에서 구현된 병렬 곱하기를 이용하여 계산 속도를 향상시킨다.

2.4 의사 역행렬 계산

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 다음과 같이 유도된다[8].

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

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 2.2와 2.3에서 구현된 함수를 사용한다.

3. 성능 측정 및 분석

그림 4, 5, 6 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=400에서 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.

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.

4. 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. 추후 과제로서 기본적인 행렬 계산과 아울러 ET 시스템에서 수행시간을 차지하는 자코비언(Jacobian)에 대해 병렬 프로그램 작성에 의해 복원의 속도를 개선할 수 있다.

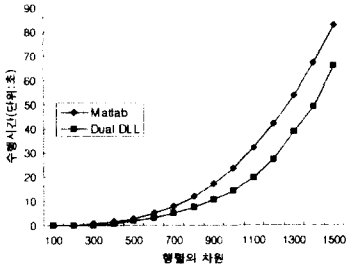

그림 4. 행렬 곱하기의 수행시간

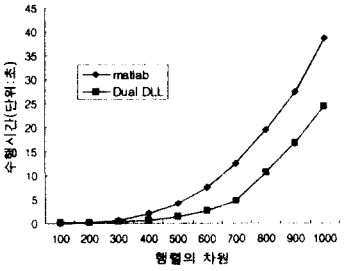

그림 5. 역행렬 계산 시간

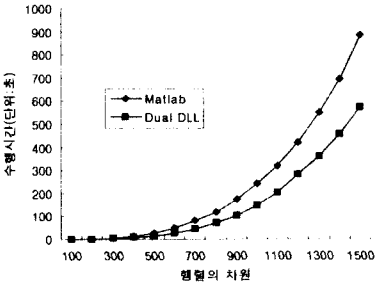

그림 6. 의사 역행렬 계산 시간

참고문헌

[1] W. Furmanski, "Petaops and exaops: supercomputing on the Web," *IEEE Internet Computing*, pp.38-46, 1997.
[2] 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.
[3] S. Matthew, *Windows 2000 Server*, Sybex, 2001.
[4] R. Voisvert, et. al., "Java and numerical computing," *Computing in Science & Engineering*, pp.18-26, 2001.
[5] H. Brian, et. al, *Guide to Matlab : For Beginners and Experienced Users*, Cambridge Univ., 2001,
[6] J. Richter, *Advanced Windows*, MS Press., 1996.
[7] S. R. Searle, *Matrix Algebra Useful for Statistics*, John Wiley & Sons, 1982.
[8] K. Gallivan, et. al., *Parallel Algorithms for Matrix Computations*, SIAM, 1990.