Cray T3E에서 극한 고유치문제의 효과적인 수행

Efficient Implementation of an Extreme Eigenvalue Problem on Cray T3E

김선경
Sun Kyung Kim
대구대학교 컴퓨터정보공학부
Taegu University, Computer & Information Engineering
전화번호 : (053) 850-6582
Fax번호 : (053) 850-6589
E-mail 주소 : skkim@biho.taegu.ac.kr

ABSTRACT

공학의 많은 응용분야에서 큰 희소 행렬(Large Sparse Matrices)에 대한 가장 작거나 또는 가장 큰 고유치(Eigenvalues)들을 요구하게 되는데, 이때 많이 이용되는 것은 Krylov Subspace로의 Projection방법이다. 대칭 행렬에 대해서는 Lanczos방법을, 비대칭 행렬에 대해서는 Biorhtogonal Lanczos방법을 이용할 수 있다. 이러한 기존의 알고리즘들은 새롭게 제안되는 병렬처리 시스템에서 효과적이지 못하다. 많은 프로세서를 가지는 병렬처리 컴퓨터 중에서도 분산 기억장치 시스템(Distributed Memory System)에서는 프로세서들 사이의 Data Communication에 필요한 시간을 줄이도록 해야한다. 본 논문에서는 기존의 Lanczos 알고리즘을 수정함으로써, 알고리즘의 동기점(Synchronization Point)을 줄이고 병렬화를 위한 입상(Granularity)을 증가시켜서 MPP인 Cray T3E에서 Data Communication에 필요한 시간을 줄인다. 많은 프로세서를 사용하는 경우 수정된 알고리즘이 기존의 알고리즘에 비해 더 나은 speedup을 보여준다.

I. 서론

아주 사이즈(Size)가 큰 희소행렬에 대한 몇개의 고유치(Eigenvalue)를 구하는 것이 요구되고 또 큰 희소 선형 방정식의 해를 구해야 하는 경우가 많이 있다. 행렬이 크지 않은 경우에 고유치 구하는 문제는 QR, Jacobi 알고리즘을 이용하지만 행렬이 아주 큰 경우는 Lanczos, Arnoldi, Biorthogonal Lanczos방법같은 반복적 방법(Iterative Method)에 의해서 구해야 한다.

기존의 알고리즘들은 새롭게 제안되는 병렬처리 시스템에서 효과적이지 못하다. 수개 또는 수만개 까지의 프로세서를 가지는 병렬처리 컴퓨터 중에서도 프로세서들 사이의 실제적인 자료 교환(Data Communication)이 필요한 분산 기억장치 시스템(Distributed Memory System)에서는 각 프로세서 사이의 자료 교환시간이 많은 부분을 차지한다. 그러므로 같은 거리에 있는 프로세서 사이에서 더 많은 양의 데이터를 한꺼번에 이동하는 것이 적은 양일 때 보다 효과적이다. 그러므로 기존의 알고리즘에서 한번에 더 많은 자료를 이동하기 위해서는, 즉 병렬처리(Parallelism)를 위한 입상(Granularity)을 증가시키는 방향으로 알고리즘을 개발하기 위해서 동기점을 줄여야 한다. 본 논문에서는 기존의 Lanczos방법을 Cray T3E에서 수행할 경우 통신시간을 줄일 수있도록 알고리즘을 수정하며 그 방법에 관해 논하고 Cray T3E에서 수행한 다음 결과를 분석한다.

II. 본론

2.1 대칭행렬에 대한 방법

주어진 대칭행렬 A(사이즈 N × N) 에서 몇개의 고유치를 구하기 위해서 사용되는 Lanczos방법은, Krylov subspace $\{q_1, Aq_1, \ldots, A^{j-1}q_1\}$의 orthogonalization을 이용하여, 주어진 대칭행렬을 사이즈가 아주 적은 tridiagonal 행렬 $T_j = Q_j{}^T A Q_j$ 로 바꾼다음 고유치를 구하는 것이다. 여기에서 $Q_j = q_1, q_2, \ldots, q_j$는 Krylov Subspace 의 Orthonormal 벡터들로 Lanczos벡터라고 한다. 기본적인 Lanczos알고리즘은 다음과 같다.

Algorithm 1. The Lanczos Algorithm

Choose q_1 with $\|q_1\|_2 = 1$, $q_0 = 0$

For j=1 until Convergence Do

1. Compute and store Aq_j
2. $\alpha_j = (Aq_j, q_j)$
3. $r_j = Aq_j - \beta_{j-1}q_{j-1} - \alpha_j q_j$
4. $\beta_j = \sqrt{(r_j, r_j)}$
5. $q_{j+1} = r_j / \beta_j$

EndFor

알고리즘 1 의 j번째 반복에서 생성되는 tridiagonal 행렬 T_j 는 다음과 같다.

$$T_j = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & \beta_{j-1} & \alpha_j \end{bmatrix}$$

Algorithm 1 에서 단계 1-5 까지의 한번의 반복 수행동안, 단계 1에서 행렬과 벡터의 곱연산이 필요하고 단계 2 와 4 에서 벡터의 곱연산이 필요하며 단계 3 와 5 에서 벡터 수정연산이 필요하다.

2.2 알고리즘의 재조정

병렬처리 시스템에서는 수행될 자료(Data)와 제어(Control)를 어떤식으로 각 프로세서에 분배하는가 하는 문제가 매우 중요하다. 본 연구에서는 제어보다는 많은 양의 자료를 어떻게 분배할 것인가를 고려하는 자료 병렬처리 알고리즘(Data Parallel Algorithm)에 주안점을 둔다. 편미분 방정식으로 표현되는 시스템으로부터 유한차분법에 의해서 형성되는 행렬은 다음과 같은 sparse banded 행렬이 되는데

$A = [C_{k-1}, D_k, B_k], \quad 1 \le k \le n,$

여기서 D_i 는 tridiagonal 행렬이 되고, B_i, C_i 는 대각행렬이 된다. 그리고 $C_0 = B_n = 0$ 이다.

행렬이 위와 같은 형태일 때 행렬과 벡터의 곱 연산시, Cray T3E와 같은 분산 메모리 시스템에서 이웃한 프로세서들 사이에 부분적인 자료전송으로 충분하다. 그러나 벡터의 곱 연산은 실제 연산 시간에 비해 프로세서들 사이의 자료 교환 시간의 비중이 크다. 따라서 가급적 한꺼번에 모으는 것이 알고리즘의 효과적인 병렬 처리를 위해서 필요한 데 다음과 같이 수정될 수 있다.

Algorithm 2. The restructured Lanczos Algorithm

Choose r_0 with $r_0 \neq 0$, $q_0 = 0$

For j=0 until Convergence Do

1. Compute and store Ar_j
2. $\beta_j = \sqrt{(r_j, r_j)}$
3. $\alpha_{j+1} = (Ar_j, r_j)/(r_j, r_j)$
4. $q_{j+1} = r_j/\beta_j$
5. $r_{j+1} = Ar_j/\beta_j - \beta_j q_j - \alpha_{j+1} q_{j+1}$

EndFor

2.3 실험결과

이 절에서는 2.2 절에서 수정된 알고리즘과, 2.1 절의 기존 알고리즘을 Cray T3E에서 수행한 결과를 분석한다. 먼저 한번의 반복시 필요한 연산의 수와 통신 횟수를 테이블 1 에서 분석하였다. 벡터의 곱하기 연산은 각 벡터를 프로세서의 수에 맞게 분배한 후 프로세서는 주어진 자료들에 대해 각각 연산한다. 그후 모든 프로세서에 있는 결과값을 더해야 하는데 이 과정에서 모든 프로세서들 사이의 결과값 교환이 필요하다. 그림 1 은 Cray T3E에서, 벡터의 길이가 4096일때 2번의 Inner products를 한꺼번에 수행했을 경우와 각각 다른 단계에서 수행했을 경우 프로세서의 수에 따른 speedup을 보여 주는 그래프이다. 그림 2 는 행렬크기가 4096×4096 일 때, 기존의 Lanczos 알고리즘과 재조정된 Lanczos 알고리즘에 대해, 프로세서의 수가 더 많아질수록 수정된 알고리즘의 speedup이 더 좋아지는 것을 알 수 있다.

표 1. Lanczos 알고리즘의 벡터연산과 통신횟수

	standard	restructured
벡터곱연산	2	2
벡터수정연산	3	4
행렬벡터곱연산	1	1
global communication	2	1
local communication	1	1

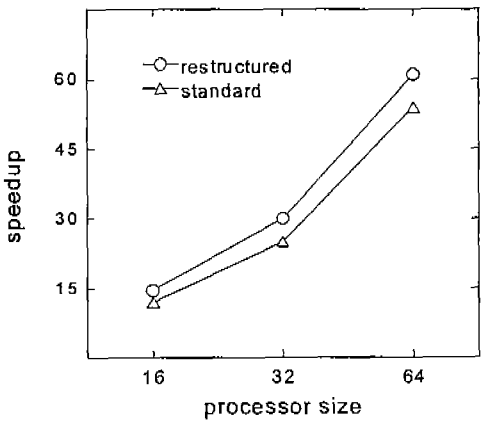

그림 1. 벡터 곱 연산의 speedup

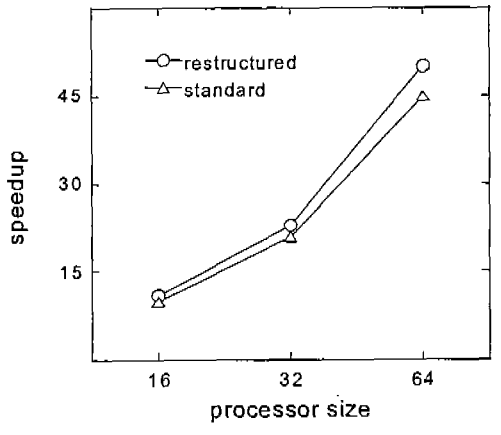

그림 2. Lanczos 알고리즘의 speedup

III. 결론

수개부터 수만개의 프로세서들을 가지고 있는 병렬처리 시스템이 여러분야에서 많이 이용되고 있는데, 데이타 연산에 참여하는 처리기의 수가 늘어갈 수록 통신비용의 비중이 보다 중요한 요소로 작용한다. 본 논문에서는 Lanczos 알고리즘에 대하여 통신비용을 좀 더 줄일 수 있는 방법, 즉 병렬 처리에 더 적합한 알고리즘으로 재구조 작업이 이루어졌다. 즉 벡터의 곱 연산을 한꺼번에 많이 함으로써 분산 메모리 시스템에서 자료 전송시간을 줄일 수 있으며, 본 논문에서 제안된 수정 알고리즘은 Cray T3E에서 많은 프로세서를 사용하는 경우 더 나은 speedup을 보여준다.

IV. 참고문헌

[1] Jane K. Cullum and Raph A. Willoughby, "Lanczos Algorithms for Large Symmetric eigenvalues Computation"., Birkhauser Boston, Inc. 1985

[2] Horoi M, Enbody R, "Efficient implementation of a Lanczos Eigenvalue Solver on a Cray T3E-900", High-Performance Computing and Networking, 1401:907-909 1998

[3] B.N.Parlett, D.R.Taylor and Z.A.Liu, "A Lookahead Algorithm for Unsymmetric Matrices", Mathematics Computation, Vol 44, No.169, January 1985, pp105-124