Cray T3E에서 효과적인 과학계산의 수행

Efficient Scientific Computation on Cray T3E

김선경
Sun Kyung Kim

대구대학교 컴퓨터정보공학부
Taegu University, Computer & Information Engineering

주소 : 경상북도 경산시 진량면 내리리 대구대학교 컴퓨터정보공학부
우편번호 : 712-714
전화번호 : (053) 850-6582
Fax번호 : (053) 850-6589
E-mail 주소 : skkim@biho.taegu.ac.kr
발표분야 : 기타 관련 분야 (병렬처리)

Abstract : 슈퍼컴퓨터는 여러 분야에서 많이 이용되고 있으며 특히 과학과 공학 분야에서 해결하려는 응용 문제들은 더욱 빠른 컴퓨터에 대한 요구가 보다 많아지고 있다. 이미 단일 프로세서로는 그 요구를 충족시킬 수 없으며 따라서 병렬처리 기법의 도입이 불가피하다. 컴퓨터는 하드웨어만으로 모든것이 해결되지 않는다. 하드웨어적인 특징을 극대화할 수 있는 알고리즘과 프로그램 등 소프트웨어 개발이 필수적이다. 본 논문에서는 아주 큰 행렬의 극한의 고유치(extreme eigenvalue)를 구하는 란초스(Lanczos) 알고리즘, 또한 아주 큰 선형시스템의 해를 구하는 GMRES방법에 대하여 병렬알고리즘을 제안하고 message-passing 병렬처리 컴퓨터에서 얼마나 효과적으로 수행할 수 있는지 분석한다. 초병렬 컴퓨터(MPP)인 Cray T3E는 128개의 PE(Processing Element)로 구성되어 있는데 사용하는 PE의 수에 따라 병렬알고리즘의 성능분석을 하였다.

I. 서론

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 교환시간을 줄이는 것이 중요하다. 그러므로 같은 거리에 있는 프로세서 사이에서 더 많은 양의 데이터를 한꺼번에 이동하는 것이 적은 양일 때 보다 효과적이다. 아주 사이즈(Size)가 큰 희소행렬에 대한 몇개의 고유치(Eigenvalue)를 구하는 것이 요구되는 경우 Lanczos, Arnoldi, Biorthogonal Lanczos 알고리즘같은 반복적 방법(Iterative Method)에 의해서 구해야 한다. 기존의 알고리즘들은 새롭게 제안되는 병렬처리 시스템에서 효과적이지 못하다. 기존의 알고리즘에서 한번에 더 많은 자료를 이동하기 위해서는, 즉 병렬처리(Parallelism)를 위한 입상(Granularity)을 증가시키는 방향으로 알고리즘을 개발하기 위해서는 동기점을 줄여야 한다. 본 논문에서는 기존의 Lanczos방법을 분산 기억장치 시스템에서 수행할 경우 통신시간을 줄일 수있도록 병렬 s-step Lanczos 알고리즘을 개발한다. 기존의 알고리즘과 수정된 알고리즘을 각각 MPP인 Cray T3E에서 수행한 다음 병렬적 효과를 분석한다.

II. 본론

2.1 기본적인 Lanczos 알고리즘

주어진 대칭행렬 A(N×N)에서 몇 개의 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 백터들로 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. 아주 정확한 결과를 얻기 위해서는 reorthogonalization을 이용한 iterative Lanczos방법을 사용해야 하지만 결국 연산구조는 기본적인 Lanczos 알고리즘과 비슷하기 때문에 본 논문에서는 알고리즘 1 을 가지고 Cray T3E에서 수행하도록한다. 이 경우 고유치만 구하는 경우 길이 N 의 두개 벡터가 들어갈 장소가 필요하다.

2.2 병렬 s-step Lanczos 알고리즘

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 이런점을 고려해서 병렬 s-step Lanczos 알고리즘을 개발한다.

병렬 s-step GMRES 알고리즘을 개발하는 방법은, s 개의 선형 독립(linearly independent)인 벡터 $\overline{V_k}$, 즉 $[v_k^1, Av_k^1, \ldots, A^{s-1}v_k^1, v_k^1 \in IR^{n\times 1}]$을 이용하여 Lanczos 알고리즘의 s 번 반복을

동시에 수행하는 것이며 다음과 같이 나타낼 수 있다.

Algorithm 2. s-step Lanczos Algorithm

Select v_1^1

Compute $\overline{V_1} = v_1^1, A v_1^1, \ldots, A^{s-1} v_1^1$

Compute 2s inner products

For k=1 **until** Convergence **Do**

1. Call Scalar1
2. Compute $v_{k+1}^1 = A v_k^s - \overline{V_{k-1}} \gamma_{k-1}^s - \overline{V_k} \alpha_k^s$
3. Compute $A v_{k+1}^1, A^2 v_{k+1}^1, \ldots, A^s v_{k+1}^1$
4. Compute the 2s inner products
5. Call Scalar2
6. Compute $v_{k+1}^j = A^{j-1} v_{k+1}^1 - \overline{V_k} t_k^j$ for $j=2, \ldots, s$

EndFor

Scalar1 : Decompose W_k and

solve $W_{k-1} \gamma_{k-1}^i = c_{k-1}^i$, $W_k \alpha_k^i = d_k^i$ for $i=1, \ldots, s$

Scalar2 : Solve $W_k t_k^j = b_k^j$, for $j=1, \ldots, s$

$$\overline{T_k} = \begin{bmatrix} G_1 & E_1 & & & \\ F_1 & G_2 & E_2 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & F_{k-1} & G_k \end{bmatrix}$$

여기에서 G_k, E_k 그리고 F_k 는 s x s 행렬인데, 특히 F_k는 (1,s) 위치에 있는 원소만이 0이 아닌 행렬이다. 또한 $W_k = \overline{V_k}^T \overline{V_k} = (v_k^i, v_k^j)$, $1 \le i, j \le s$ 이며 대칭인 행렬이다. 그리고 위의 알고리즘에서 필요한 모든 벡터의 곱은 행렬 A와 관련한 벡터 v_k^1의 앞부분 2s moments로 모두 대치될 수 있다. 기본적인 Lanczos 알고리즘에서의 T_j와 마찬가지로 s-step Lanczos 알고리즘에서 A의 극한 고유치는 $\overline{T_k}$의 고유치 λ_k를 이용해서 구할 수 있다.

2.3 Cray T3E 병렬처리 시스템에서 수행시간 비교

기본적인 Lanczos 알고리즘의 s번 반복 동안 필요한 벡터 연산과 통신 횟수 또한 s-step Lanczos 알고리즘의 한번의 반복 동안 필요한 벡터 연산과 통신 횟수를 비교하면 표 1과 같다. s-step 알고리즘이 약간 더 많은 연산을 요구하지만 통신 횟수는 확연히 줄어든 것을 알 수 있다. 표 2는 개발된 s-step 알고리즘의 정확도를 나타내 주는 실험 결과이며 기본적인 Lanczos 알고리즘과 결국 같은 고유치를 계산해 낸다는 것을 보여 준다. 행렬 A는 다음 PDE 문제로부터 얻어진 것이다. γ와 β를 모두 0으로 취하면 행렬이 대칭이 된다.

문제 1 : $-(bu_x)_x-(cu_y)_y+(du)_x+(eu)_y+fu=g$

on the unit square, where

$b(x, y)=e^{-xy}$, $c(x, y)=e^{xy}$, $d(x, y)=\beta(x+y)$,

$e(x, y)=\gamma(x+y)$ and $f(x, y)=1/(1+x+y)$

subject to the Dirichlet boundary conditions u(x,y) equals the solution on the boundary.

다음은 분산 기억장치 시스템이고 MPP인 Cray T3E에서 s-step lanczos 알고리즘의 병렬성이 얼마나 더 효과적인지 그림 1에서 보여 준다. 병렬 s-step 알고리즘은 s 번의 반복동안 필요한 모든 벡터 곱연산을 한꺼번에 처리할 수 있기 때문에 표 1에서 나타나 있듯이, 통신 시간을 벡터 곱연산에 대해서만 고려하면 기존의 방법에 비하여 1/(2*s)가 된다. 다음으로 행렬과 벡터의 곱연산도 병렬 알고리즘에서는 s 번의 step 동안 필요한 연산이 한꺼번에 일어나므로 Cray T3E에서 수행시는 이웃한 프로세서 사이에 자료 전송시간이 줄어들 수 있다.

표 1. 벡터연산과 통신회수

	standard Lanczos	s-step Lanczos
벡터곱연산	2s	2s
벡터수정연산	5s	2s(s+1)
행렬벡터곱연산	s	s+1
global communication	2s	1
local communication	s	1

표 2. 문제 1의 가장 큰 고유치

T_j,T_k의 크기	standard Lanczos	5-step Lanczos
10×10	0.10704428E+02	0.10704427E+02
20×20	0.11083956E+02	0.11083955E+02
30×30	0.11086467E+02	0.11086460E+02
40×40	0.11086467E+02	0.11086460E+02

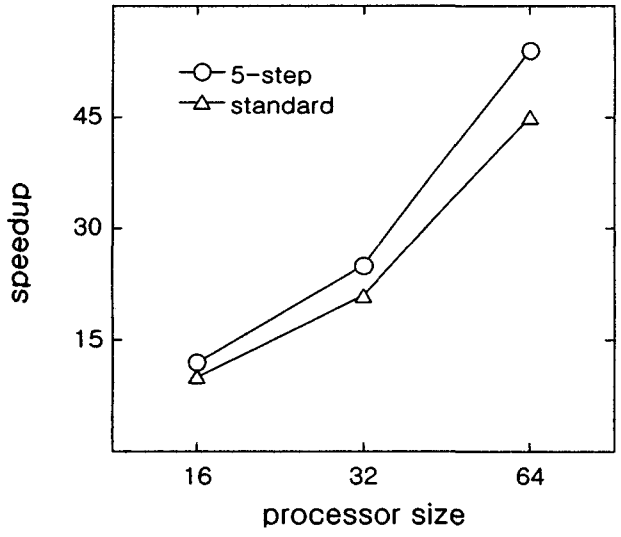

그림 1. Cray T3E에서의 speedup

III. 결론

많은 프로세서들을 가지고 있는 병렬처리 시스템이 여러분야에서 많이 이용되고 있는데, 데이타 연산에 참여하는 처리기의 수가 늘어갈수록 통신비용의 비중이 보다 중요한 요소로 작용한다. 특히 message passing 병렬처리 시스템에서는 데이타 연산에 참여하는 처리기의 수가 늘어갈수록

통신비용의 비중이 보다 중요한 요소로 작용한다. 본 논문에서는 Lanczos 알고리즘에 대하여 통신비용을 좀 더 줄일 수 있는 방법, 즉 병렬 처리에 더 적합한 병렬 알고리즘이 제안되었다. 기존의 Lanczos 방법이 만드는 축소 행렬에 similar하고 같은 고유치를 가지는 행렬을 만들며 병렬 처리 시스템에서 더 효과적인 s-step Lanczos 알고리즘이다. s-step Lanczos 방법은 MPP인 Cray T3E에서 기존의 Lanczos 방법에 비해 더 나은 speedup을 보여 준다는 것을 증명하였다. 즉 벡터의 곱연산을 한꺼번에 많이 함으로써 분산 메모리 시스템에서 자료 전송시간을 줄일 수 있기 때문이다. 본 논문에서 제안된 병렬 알고리즘의 문제점으로는 연산의 수가 부분적으로 좀 더 많아지는 것이며 s가 5보다 커지면 s-step 알고리즘이 약간 불안정하 된다는 점이다..

IV. 참고문헌

[1] A. T. Chronopoulos, s-step iterative methods for (non)symmetric (in)definite linear systems, SIAM J. on Num. Analysis 28, 6 (1991).

[2] A. T. Chronopoulos and C. W. Gear, s-step iterative methods for symmetric linear systems, J. of Comput. and Appl. Math. 25, 153-168, 1989

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

[4] Gupta A, Joshi M, Kumar V, "A high-performance serial and parallel symmetric sparse linear solver" Applied Parallel Computing, 1541:182-194 1998

[5] 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

[6] 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

[7] P. E. Saylor, Leapfrog variants of iterative methods for linear algebraic equations, J. Comput. and Appl. Math. 24, 169-193, 1988.