Message Passing Parallel System에서 효과적인 s-step GMRES 알고리즘

Efficient s-step GMRES Algorithm on a Message Passing Parallel System

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

병렬 컴퓨터를 사용하는 경우 하드웨어만으로 모든 것이 해결되지 않으며 병렬처리 기법의 도입이 불가피하다. 효과를 극대화하기 위하여서는 각 병렬 컴퓨터의 하드웨어적인 특징을 극대화할 수 있는 병렬 알고리즘과 병렬 프로그램 등 소프트웨어 개발이 필수적이다. GMRES(Generalized Minimal residual) 방법은 아주 큰 대칭 또는 비대칭 선형시스템의 해를 구하는 반복법 중의 하나로 일반적으로 많이 사용되고 있다. 서로 직교인 벡터를 하나씩 구하는 대신에 선형인 s개의 벡터를 구하고 각 그룹간에는 직교가 되게하는 s-step GMRES 알고리즘은 병렬적 성질을 더 많이 가지고 있다. 이 병렬 알고리즘의 전반부는 이미 개발된 s-step Arnoldi 알고리즘을 이용할 수 있다. s-step GMRES 알고리즘은 message passing 병렬 시스템에서 모든 프로세서들 사이의 자료 교환 시간을 줄임으로써 기존의 GMRES 방법에 비해 훨씬 더 병렬성을 증가시킨다. 본 논문에서는 초병렬 시스템(MPP)인 Cray T3E에서 많은 프로세서를 이용할 경우 개발된 s-step 알고리즘이 기존의 알고리즘에 비하여 얼마나 더 효과적으로 빨리 수행될 수 있는지 분석한다.

I. 서론

선형시스템 $Ax=b$ 의 해를 구할 때 A의 크기가 100이하 일 때는 가우스 소거법이나 LU 분해법등을 써서 그 해를 구하지만 A의 크기가 100보다 훨씬 커지면 여러 가지 방법중에서도 GMRES방법을 이용할 수 있다. 기존의 GMRES 알고리즘은 새롭게 제안되는 병렬처리 시스템에서 효과적이지 못하다. 병렬처리 시스템에서 효과적으로 수행하기 위해서는 각 병렬 컴퓨터의 하드웨어적인 특징을 극대화할 수 있는 병렬 알고리즘의 개발이 필수적이다. 병렬처리 컴퓨터 중에서도 프로세서들 사이의 실제적인 자료 교환(Data Communication)이 필요한 message passing 병렬 시스템에서는 각 프로세서 사이의 자료 교환시간을 줄이는 것이 중요하다. 자료 교환시간을 줄이기 위해서는 한번에 더 많은 자료를 이동하도록 기존의 알고리즘을 수정해야 한다. 즉 병렬처리(Parallelism)를 위한 입상(Granularity)을 증가시키고 동기점을 줄이는 방향으로 알고리즘을 재개발하는 것이 필요하다. 본 논문에서는 기존의 GMRES 방법과 병렬 알고리즘인 s-step GMRES 방법을 MPP인 Cray T3E에서 수행해보고 병렬 알고리즘의 효과를 분석한다.

II. 본론

2.1 GMRES 알고리즘

GMRES 방법은 Arnoldi 방법에 기초해서 만들어 진 반복법인데 $H_j = Q_j^T A Q_j$, 는 $j \times j$ Hessenberg 행렬이며 H의 각 원소인 $h_{i,l}$ 가 Arnoldi 방법에 의해서 만들어 진다. x_0를 GMRES 반복법의 초기해라고 한다면 r_0는 $b-Ax_0$ 가 되며 반복이 거듭될 때 r_i 의 크기는 반복을 종료하는 기준이 된다. 반복적인 GMRES(m) 방법은 다음과 같다.

Algorithm 1 GMRES(m) Algorithm

Compute $r_0 = b - Ax$ and

set $q_1 = \dfrac{r_0}{\| r_0 \|_2}$

For $j=1, \ldots m-1$

1. $h_{i,j} = q_i^T A q_j, \ 1 \le i \le j$
2. $\hat{q}_{j+1} = Aq_j - \sum_{i=0}^{j} h_{i,j}\, q_j$
3. $h_{j+1,j} = \| \hat{q}_{j+1} \|_2$
4. $q_{j+1} = \hat{q}_{j+1} / \| \hat{q}_{j+1} \|_2$

EndFor

From the approximate solution: $x_m = Q_m y_m$, where y_m minimizes

$$J(y) = \| \beta e_1 - G_m y \|_2 \qquad (1)$$

Here $e_1 = [1, \ldots, 0]^T$. The matrix G_m is the same as H_m except for an additional row whose only non-zero element is $h_{m+1,m}$ in the (m+1, m) position. Minimizing the error functional m-dimensional $J(y)$ is equivalent to solving:

$$\min_{x \in x_0 + K_m} \| b - Ax \|_2 \text{ where}$$

$$K_m = span\{ r_0, Ar_0, \ldots, A^{m-1}r_0 \}$$

is the Krylov subspace of dimension m.

식 (1)은 최소 선형자승 문제인데 QR 방법에 의해서 풀 수 있다. GMRES방법의 한번 반복 시에 i+2번 벡터의 곱과 i+1번의 벡터 수정 그리고 1번의 행렬과 벡터의 곱(Mv)이 필요하다..

2.2 병렬 s-step GMRES 알고리즘

병렬 s-step GMRES 알고리즘을 개발하는 방법은, s 개의 선형 독립(linearly independent)인 벡터 집합을 사용하여서 기존의 Arnoldi 알고리즘의 s 번 반복을 동시에 수행하는 s-step Arnoldi 알고리즘을 이용하는 것이다. 이때 s 개의 선형 독립인 벡터 집합 $\overline{V}_k$는 $[V_k^1, AV_k^1, \ldots, A^{s-1}V_k^1,\ V_k^1 \in IR^{n\times p}]$ 에의해 span 된다. 먼저 s-step Arnoldi 방법은 다음과 같다.

Algorithm 2 s-step Arnoldi method

Select v_1^1 and compute

$$\overline{V}_1 = [v_1^1,\ v_1^2 = Av_1^1, \ldots,\ v_1^s = A^{s-1}v_1^1]$$

For $k = 1, \ldots, m/s$

1. Call Scalar1.
2. Compute
$$v_{k+1}^1 = Av_k^s - \sum_{i=1}^{k} \overline{V}_i[\ \overline{h}_{ik}^{\ 1}].$$
3. Compute
$$v_{k+1}^2 = Av_{k+1}^1, \ldots, v_{k+1}^s = A^{s-1}v_{k+1}^1$$
4. Compute $(A^i v_k^1, v_l^j)$ for
$1 \le i, j \le s-1$ and $1 \le l \le k-1$.
5. Compute $(A^i v_k^1, A^j v_k^1)$
for $0 \le i \le s-1$ and $i \le j \le s$.
6. Call Scalar2.
7. Compute $\overline{V}_{k+1} = [v_{k+1}^1, \ldots, v_{k+1}^s]$
$$-\sum_{i=1}^{k} \overline{V}_i[0,\ t_{ik}^1, \ldots, t_{ik}^{s-1}].$$

EndFor

Scalar1: Compute and decompose
$$W_i = \overline{V}_i^{\ T}\ \overline{V}_i.$$
Solve $W_i\ \overline{h}_{ik}^{\ q} = b_{ik}^q$
for $q = 1, \ldots, s,$ where

$$b_{ik}^1 = [(v_i^1, Av_{k+1}^1), \ldots, (v_i^s, Av_{k+1}^1)]^T$$

$$, \ldots ,$$

$$b_{ik}^{s-1} = [(v_i^1, A^{s-1}v_{k+1}^1), .., (v_i^s, A^{s-1}v_{k+1}^1)]^T$$

Scalar2: Solve $W_i t_{ik}^q = b_{ik}^q$
for $1 \le q \le s-1$, where
$$[c_{ik}^1, \ldots, c_{ik}^s] = \overline{V}_i^{\ T} A\ \overline{V}_k.$$

필요한 모든 벡터의 곱은 5,6 단계에서 한꺼번에 처리될 수 있다. 이 s-step Arnoldi 방법을 이용한 s-step GMRES는 다음과 같다.

Algorithm 3 s-step GMRES method

1. Compute r_0 and set $v_1^1 = \frac{r_0}{\| r_0 \|_2}$.

2. Compute the s-step Arnoldi vectors
   $V_1, \ldots, V_m$.

3. Form the approximate solution:
   $x_m = x_0 + V_m \bar{y}_m$

4. where y_m minimizes
   $J(y) = \| [\beta e_1 - L \bar{G}_k \bar{y}] \|_2$.

5. Restart:
   Compute $r_m = b - Ax_m$ and
   stop if $\| r_m \| < e$ else set $x_0 = x_m$
   and $r_0 = r_m$ and go to 1.

기본적인 GMRES 방법에 비해서 s-step GMRES방법의 message passing 병렬처리 시스템에서의 장점은 기본적인 GMRES(m)에서 s 번의 반복동안 필요한 모든 벡터의 곱, 행렬과 벡터의 곱이 한꺼번에 처리될 수 있다.

2.3 Cray T3E 병렬처리 시스템에서 수행시간 비교

병렬 s-step 알고리즘은 s 번의 반복동안 필요한 모든 벡터 곱연산을 한꺼번에 처리할 수 있기 때문에, 통신 시간을 벡터 곱연산에 대해서만 고려하면 기존의 방법에 비하여 1/(2*s)가 된다. 또한 행렬과 벡터의 곱연산도 병렬 알고리즘에서는 s 번의 step 동안 필요한 연산이 한꺼번에 일어나므로 Cray T3E에서 수행시에 message passing 에 필요한 시간을 훨씬 줄일 수 있다. 이 절에서는 변형된 알고리즘과, 기존 알고리즘을 Cray T3E에서 수행한 결과를 분석한다. 먼저 한번의 반복시 필요한 벡터연산의 수를 표 1 에서 분석하였다. 근사해의 수렴 정도는 $\| r_i \| < 10^{-6}$ 를 이용한다. 선형시스템의 사이즈는 표 2에서 나타내고 있으며 거기에 따른 s-GMRES(m)의 반복횟수를 준다. s-step GMRES는 한번의 행렬과 벡터의 곱이 더 필요하기 때문에 s는 가급적 클수록 유리하나 s가 5보다 커지면 불안정한 성질을 가진다. 표 2는 GMRES(10)의 반복수가 5-GMRES(2)의 반복수와 같다는 것을 보여준다. 그림 1은 기본적인 방법에 비해 s-step 방법이 많은 프로세서를 쓰는 경우 더 나은 speedup을 가진다는 것을 보여주며 이것은 줄어든 자료교환 시간으로부터 나타나는 효과이다.

III. 결론

Message passing 병렬처리 시스템이 여러분야에서 많이 이용되고 있는데, 데이타 연산에 참여하는 처리기의 수가 늘어갈 수록 통신비용

표 1. GMRES와 s-step GMRES 벡터연산수

연산	GMRES	s-step GMRES
벡터곱	$(m-1)s^2+$ $[s(s+1)/2+s]$	$(m-1)s^2+$ $[s(s+1)/2+s]$
벡터수정	$(m-1)s^2+$ $[s(s+1)/2+s]$	$(m-1)s^2$
행렬벡터곱	s	$s+1$
벡터저장	$ms+1$	$(m+1)s$

표 2. 반복회수($\| r_i \| < 10^{-6}$)

vector size	GMRES(10)	5-step GMRES(2)
32	16	16
64	30	30
128	60	61
192	88	88
256	169	153

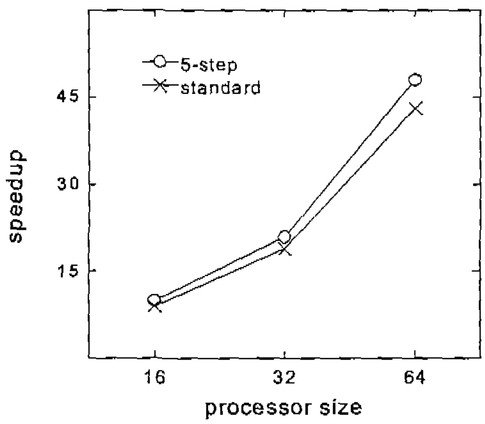

그림 1. GMRES(10)와 5-GMRES(2)의 Cray T3E speedup

의 비중이 보다 중요한 요소로 작용한다. 본 논문에서는 GMRES 알고리즘에 대하여 통신비용을 좀 더 줄일 수 있는 방법, 즉 병렬 처리에 더 적합한 병렬 알고리즘이 제안되었다. s-step Arnoldi 방법을 기반으로 한 s-step GMRES 방법을 MPP인 Cray T3E에서 기존의 GMRES 방법에 비해 더 나은 speedup을 보여 준다는 것을 증명하였다. 즉 벡터의 곱연산을 한꺼번에 많이 함으로써 분산 메모리 시스템에서 벡터의 곱 연산시에 필요한 전체 프로세서사이의 자료 전송시간을 줄일 수 있다.

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

[4] S. K. Kim and A. T. Chronopoulos, An efficient Arnoldi method implemented on parallel computers, Proc. of the 1991 Intern. Conf. on Parallel Processing, Vol. III, 167-170.

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