Resolução Paralela de Sistemas de Equações Lineares Algébricos

Alessandro Copetti[1], Edson Luiz Padoin[12], Marlon Possani[1], Manuel Binelo[1], Oleg Khatchatourian[2]

[1]UNICRUZ - Departamento de Informática
PPD - Grupo de Processamento Paralelo e Distribuído
Rua Andrade Neves, 308 – 98025-810 – Cruz Alta, RS
[2]UNIJUI - Departamento de Física, Estatística e Matemática
Rua São Francisco, Cx. P. 560 – 98700-000 – Ijuí, RS
{copetti, padoin, possani, manoelb, olegkha}@unicruz.edu.br

Introdução

Os algoritmos seqüenciais para os métodos de *Householder* e de *Givens* [EGE 95] são muito utilizados nas área de matemática aplicada conforme sua descrição detalhada nas obras de [KHA 95] e [ORT 88]. Este trabalho descreve a paralelização desses métodos, aplicações de alto custo computacional utilizadas na resolução de sistemas lineares. Procura-se desenvolver uma solução de baixo custo que explore os recursos da computação paralela com o objetivo de melhor desempenho para resolução de SELA[1] com matriz densa e de grande porte.

A seguir são descritos brevemente os algoritmos na sua versão seqüencial e paralela e os resultados obtidos.

Aspectos do Algoritmo Seqüencial

O grupo PPD implementou a versão seqüencial para as transformações ortogonais de *Householder* (método de reflexão) e as transformações de *Givens* para resolução de SELA com matriz quadrada não singular $A\boldsymbol{x}=\boldsymbol{b}$ onde $A \in M_n(\Re)$, $\mathbf{x} \in \Re^n$, $\mathbf{b} \in \Re^n$.

A aplicação de *Householder* pode ser definida como um vetor-coluna unitário $\mathbf{w} \in \Re^n$ e uma matriz $\boldsymbol{U}=\boldsymbol{I}-2\boldsymbol{w}\boldsymbol{w}^T$, onde chama-se transformação de *Householder* a matriz de reflexão. É possível demonstrar que U é ortogonal, simétrica e que $\mathbf{w}$ é um autovetor desta matriz associado com autovalor $\lambda = -1$.

Para ampliar a capacidade do aplicativo de resolver os sistemas de ordem maior utilizando somente memória operacional, o algoritmo de multiplicação da matriz U pela matriz aumentada B foi transformado da seguinte forma:

$$UB=(I-2w\cdot w^T)B=B-2w(w^TB)$$

Isto permitiu operar somente com uma matriz, calculando inicialmente o vetor-linha $\mathbf{w}^T\mathrm{B}$ e subtraindo os elementos do produto dos vetores $\mathbf{w}$ e $\mathbf{w}^T\mathrm{B}$ dos elementos da matriz B diretamente sem formação de matriz intermediária.

[1] sistemas de equações lineares algébricos

Os detalhes dos métodos de *Householder* e *Givens* podem ser encontrados na íntegra em [COP 02].

Divisão e distribuição de tarefas no algoritmo paralelo

O algoritmo consiste em um grande conjunto de tarefas de integração numérica significativamente independentes. Sua paralelização é um problema de seqüenciamento, que consiste em alocar n tarefas a m processadores paralelos idênticos, que será efetuado baseado na utilização de bibliotecas para troca de mensagens no sentido de aumentar a eficiência dos algoritmos[PAD 01].

Durante o processo de paralelização foram exploradas diferentes formas para se alcançar a melhor performance sendo apresentado neste trabalho a melhor solução encontrada até então. As transformações de *Householder* e *Givens* podem ser perfeitamente utilizadas como objeto de estudo por apresentarem um grande volume de processamento com elevado tempo de computação, além de apresentarem uma forte iteração durante o processamento. O paradigma mestre – escravo foi o modelo de programação utilizado na divisão de processamento.

No caso do *Householder* paralelo, a matriz é dividida em linhas e distribuída aos escravos. No *Givens*, a divisão da matriz ocorre em colunas. O algoritmo automaticamente diminui a quantidade de processos na medida que o sistema vai diminuindo, já que a cada iteração a ordem da matriz diminui.

No *Householder* apenas os resultados são passados de processo para processo, num modelo em *pipeline* [AND 00], sem que haja necessidade de que um único, conheça toda a matriz. Já no *Givens*, há a necessidade de todos os escravos enviarem suas porções da matriz, a fim de o mestre calcular o vetor resultado.

Ambiente de programação e execução

Para a execução da aplicação utilizou-se os computadores do Laboratório de Ciência da Computação composto de 20 máquinas Athlon de 1.6 Ghz com 128 MB de RAM rodando Sistema Operacional GNU/Linux-Conectiva 7.0, versão kernel 2.2.14-5.0, linguagem de programação C – compilador GCC e a biblioteca de programação por troca de mensagens PVM [BEG 91]. A rede utilizada foi uma Fast Ethernet (100 base T).

Resultados obtidos

O algoritmo paralelo foi executado para matrizes de ordem 512, 1024 e 2048. Abaixo, na tabela 1 (*Householder*) e tabela 2 (*Givens*) são apresentados os tempos em segundos, sendo que os mesmos representam uma média de cinco execuções, o *speedup* e a eficiência.

	nodos	1	2	3	4	5	6	7	8	9	10	11	12	13
512x512	Tempo Médio	8,3200	6,2685	4,7205	4,8585	4,8856	5,3305	5,4105	6,1578	6,6942	7,1842	7,9002	8,5322	10,0042
	Speedup	1	1,3273	1,7625	1,7125	1,7030	1,5608	1,5378	1,3511	1,2429	1,1581	1,0531	0,9751	0,8316
	Eficiência	100%	66%	59%	43%	34%	26%	22%	17%	14%	12%	10%	8%	6%
1024x102	Tempo Médio	67,1200	34,1140	36,3340	31,7180	26,2040	26,8240	22,6260	24,1120	25,0800	26,8000	27,2780	27,7860	32,5860
	Speedup	1	1,96752	1,84731	2,11615	2,56144	2,50224	2,9665	2,78368	2,67624	2,50448	2,46059	2,4156	2,05978
	Eficiência	100%	98%	62%	53%	51%	42%	42%	35%	30%	25%	22%	20%	16%
2048x204	Tempo Médio	590,830	383,154	245,358	190,422	142,546	112,556	128,390	110,100	122,140	120,762	142,504	144,088	162,880
	Speedup	1	1,54202	2,40803	3,10274	4,14484	5,24921	4,60184	5,3663	4,83732	4,89252	4,14606	4,10048	3,62739
	Eficiência	100%	77%	80%	78%	83%	87%	66%	67%	54%	49%	38%	34%	28%

Tabela 1 . Tempos de execução, *speedup* e eficiência do algoritmo *Householder*

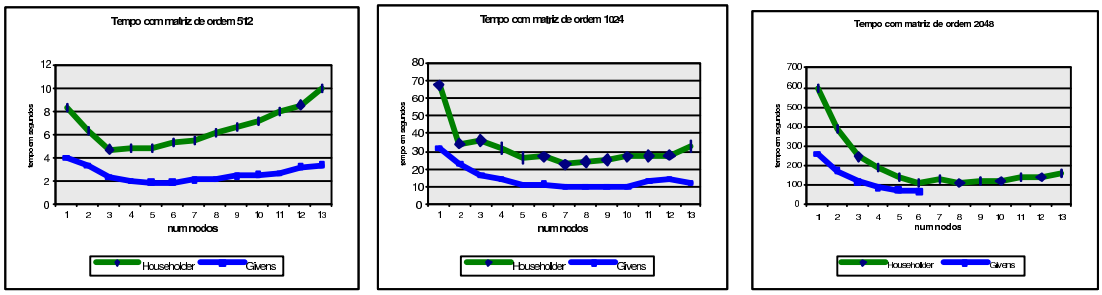
	nodos	1	2	3	4	5	6	7	8	9	10	11	12	13
512x512	Tempo Médio	3,9600	3,3300	2,3480	1,9880	1,8960	1,9260	2,1100	2,2020	2,4500	2,5660	2,7080	3,2600	3,4100
	Speedup	1	1,1892	1,6865	1,9920	2,0886	2,0561	1,8768	1,7984	1,6163	1,5433	1,4623	1,2147	1,1613
	Eficiência	100%	59%	56%	50%	42%	34%	27%	22%	18%	15%	13%	10%	9%
1024x102	tempo médio	31,6700	22,7960	16,4880	14,0980	11,0680	11,3560	10,1680	9,8500	9,9460	10,0940	13,0360	14,3180	12,0320
	Speedup	1	1,3893	1,9208	2,2464	2,8614	2,7888	3,1147	3,2152	3,1842	3,1375	2,4294	2,2119	2,6321
	Eficiência	100%	69%	64%	56%	57%	46%	44%	40%	35%	31%	22%	18%	20%
2048x204	tempo médio	258,449	174,588	117,708	89,1160	75,9640	69,8920							
	Speedup	1	1,48034	2,19568	2,90014	3,40226	3,69783							
	Eficiência	100%	74%	73%	73%	68%	62%							

Tabela 2 . Tempos de execução, *speedup* e eficiência do algoritmo *Givens*

Analisando tais resultados, pode-se observar que na execução paralela do primeiro método, utilizando-se 3, 7 e 8 nodos respectivamente para matrizes de ordem 512, 1024 e 2048 obteve-se os menores tempos de execução. Sendo que, a partir desse ponto aumentando-se o número de nodos não conseguiu-se diminuir o tempo e sim houve um aumento acentuado. Na execução seqüencial do método para uma matriz de ordem 2048 transcorreu 590 segundos, já com o método paralelo se alcançou o melhor desempenho, reduzindo-se o tempo para 110 segundos, menos da quinta parte do tempo seqüencial, ou seja um *speedup* de 5,30 e uma uma eficiência de 67%. Já no segundo método, obteve-se os melhores resultados com 5 e 8 nodos. Deve-se ressaltar que o método de *Givens* ainda não foi explorado em sua totalidade.

Pode-se concluir que quanto maior a ordem da matriz, mais se justifica o acréscimo de nodos processadores com relação ao tempo. Porém, se analisarmos a eficiência, não se pode inferir da mesma forma, pois aumentando o número de nodos diminui-se a eficiência.

Observando-se os gráficos abaixo (comparativo de desempenho entre os dois métodos) pode-se observar que o método de *Givens* sempre apresentou um menor tempo de execução. O mesmo, a partir de seu ponto ótimo (melhor *speedup*), sofreu uma pequena elevação de tempo. Enquanto que o método *Householder* apresentou uma elevação significativa.



(a) ordem 512 (b) ordem 1024 (c) ordem 2048

Figura 1 : Gráficos de desempenho dos métodos paralelos de *Householder e Givens*

Conclusões e trabalhos futuros

Apresentou-se neste trabalho os resultados da paralelização das transformações de *Householder* e *Givens* para solução de SELAs usando PVM em aglomerados de PC. Implementou-se um modelo paralelo que reduz o número de troca de mensagens alcançando-se assim um bom equilíbrio de cargas entre os processos.

Com os métodos paralelos pode-se explorar matrizes de grande porte justificando-se um número ainda maior de nodos devido ao fato que apenas uma parte da matriz é alocada na memória de cada nodo. Estes códigos paralelos ainda não apresentam um bom balanceamento de cargas, pois quando um nodo conclui o seu processamento, ele permanece inativo. Pretende-se como trabalho futuro investigar o balanceamento de carga, bem como aumentar a ordem das matrizes e explorar outros métodos numéricos.

Referências

[AND 00] ANDREWS, G., **Foundations of Multithread, Parallel, and Distributed Programming**. Adisson Wesley, 2000.

[COP 02] COPETTI, A., PADOIN, E. L., KHATCHATOURIAN, O. **Transformações de Householder e Givens para Resolução de Sistemas Lineares em Clusters de PC.** ERMAC 2002 - Encontro Reg. de Matemática Aplicada e Comp., POA, 2002.

[BEG 91] BEGUELIN, A., DONGARRA, J.J., GEIST, G.A., MANCHEK, R., SUNDERAM, V.S., **A Users' Guide to PVM parallel virtual machine**, ORNL/TM11826, 1991.

[EGE 95] EGECIOGLU Ö., SRINIVASAN, A., **Givens and Householder Reductions for Linear Least Squares on a Cluster of Workstation**, Proc. Int. Conf. on High Performance Computing (HiPC), New Delhi, pp. 734-739, 1995

[KHA 95] KHATCHATOURIAN, O., BORGES P.A., **Métodos Numéricos da Álgebra.** Ed. UNIJUÍ, 1995.

[ORT 88] ORTEGA, J. M., **Introduction to Parallel and Vector Solution of Linear Systems**, Plenum Press, New York, 1988.

[PAD 01] PADOIN, E. L. **Estudo da paralelização de aplicações com alto custo computacional utilizando sistemas distribuídos,** Dissertação(Mestrado em PPGEP) - UFSM - Universidade Federal de Santa Maria, Santa Maria, 2001.