A parallel algorithm for solving tridiagonal linear systems on coarse grained multicomputer

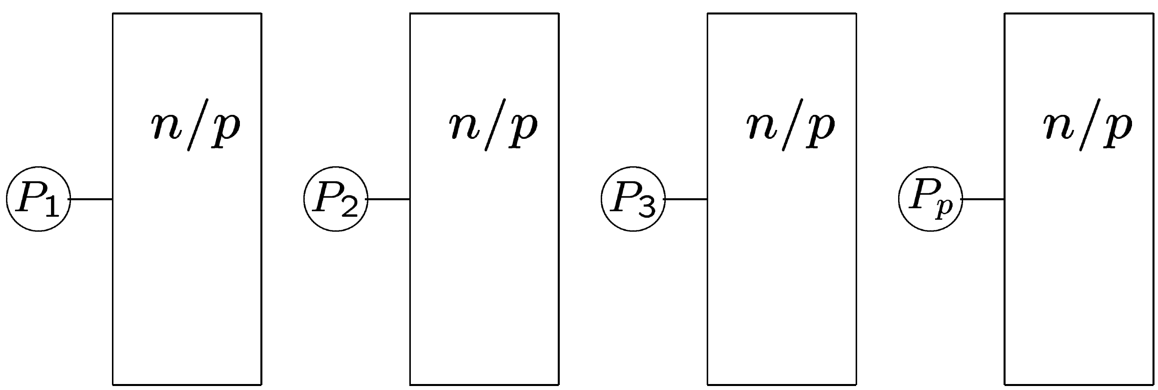
E. L. G. Saukas and S. W. Song

Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

9/outubro/1997

Modelo CGM (Coarse-Grained Multicomputer)

- Problema de tamanho n
- Arquitetura de memória distribuída
- p processadores cada um c/ mem. local $\frac{n}{p}$ (em geral $n >> p$)



Algoritmo: (tenta minimizar R)

repetir R vezes:

Computação: cada proc.
computa independentemente

Rodada comunicação: cada proc.
envia $\frac{n}{p}$ dados
recebe $\frac{n}{p}$ dados

Projeto de algoritmos CGM

Meta: projetar algoritmos que minimizam o número de rodadas de comunicação visando escalabilidade.

Alguns resultados ($p =$ num. de processadores):

Problema	Rodadas com.
Visibilidade segmentos	Constante
Casco convexo 2-D	Constante
Casco convexo 3-D	Constante
Seleção	$O(\log p)$
Seleção simultânea	$O(\log p)$
Ordenação	$O(\log p)$
List ranking	$O(\log p)$
Euler tour	$O(\log p)$
Componentes conexas	$O(\log p)$
Ancestral comum inferior	$O(\log p)$
Contração de árvore	$O(\log p)$
Decomposição em orelhas	$O(\log p)$
Biconectividade	$O(\log p)$
NESTE TRABALHO:	
Sistemas tridiagonais	Constante

Sistema Linear Tridiagonal

$Ax = b$ onde A é uma matriz $n \times n$ tridiagonal:

$$A = \begin{pmatrix} d_1 & u_1 & 0 & 0 & \ldots & 0 \\ l_2 & d_2 & u_2 & 0 & \ldots & 0 \\ 0 & l_3 & d_3 & u_3 & \ldots & 0 \\ 0 & 0 & l_4 & d_4 & \ldots & 0 \\ & & & \ldots & & \\ 0 & 0 & 0 & 0 & \ldots & d_n \end{pmatrix}$$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \ldots \\ x_n \end{pmatrix} \text{ e } \; b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \ldots \\ b_n \end{pmatrix}$$

Supor A pertence à classe de matrizes simétricas definidas positivas ou diagonalmente dominantes.

Algoritmo básico: Redução par-ímpar

1. Cada x_i com i ímpar substituída por uma função de x_{i-1} e x_{i+1}, ambas com índices pares.

2. O sistema resultante consiste de variáveis com índices pares. Sendo também tridiagonal podemos aplicar recursivamente mesma idéia até reduzir a somente uma equação em x_n.

3. Obtida x_n, trabalhamos de trás para frente obtendo $x_{n/2}$, depois $x_{n/4}$, $x_{n/8}$ etc.

4. Tendo todas as variáveis de índices pares, resolvemos para x_i de índices ímpares.

Um exemplo

$$\begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & -4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 5 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 3 \\ \frac{18}{23} \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Substituir x_1, x_3, x_5 e x_7 por

$$\begin{aligned} x_1 &= \frac{1}{4}(1 - x_2) \\ x_3 &= -\frac{1}{4}(1 - 2x_2 - x_4) \\ x_5 &= \frac{1}{3}(\frac{18}{23} - x_4 - 2x_6) \\ x_7 &= \frac{1}{5}(-2x_6 - 2x_8) \end{aligned}$$

Novo sistema tridiagonal:

$$\begin{pmatrix} \frac{15}{4} & \frac{1}{4} & 0 & 0 \\ 0 & \frac{5}{3} & -\frac{2}{3} & 0 \\ 0 & 0 & \frac{13}{5} & -\frac{2}{5} \\ 0 & 0 & -\frac{2}{5} & \frac{18}{5} \end{pmatrix} \begin{pmatrix} x_2 \\ x_4 \\ x_6 \\ x_8 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{63}{23} \\ 1 \\ 0 \end{pmatrix}$$

Substituir x_2 e x_6 por

$$\begin{aligned} x_2 &= \frac{1}{15}(2 - x_4) \\ x_6 &= \frac{1}{13}(5 + 2x_8) \end{aligned}$$

resulta em:

$$\begin{pmatrix} \frac{5}{3} & \frac{4}{39} \\ 0 & \frac{46}{13} \end{pmatrix} \begin{pmatrix} x_4 \\ x_8 \end{pmatrix} = \begin{pmatrix} \frac{2687}{897} \\ \frac{2}{13} \end{pmatrix}$$

Finalmente subsituir x_4 por

$$x_4 = \frac{1}{5}(\frac{2687}{299} + \frac{4}{13}x_8)$$

Resolver p/ índices pares e ímpares

Temos uma única equação e resolvemos para obter x_8, depois para x_6, x_4, x_2:

$$\begin{aligned} x_8 &= \frac{1}{23} \\ x_4 &= \frac{9}{5} \\ x_2 &= \frac{1}{75} \\ x_6 &= \frac{9}{23} \end{aligned}$$

Obtemos depois os valores de índices ímpares:

$$\begin{aligned} x_1 &= \frac{37}{150} \\ x_3 &= \frac{31}{150} \\ x_5 &= -\frac{3}{5} \\ x_7 &= -\frac{4}{23} \end{aligned}$$

O algoritmo para memória compartilhada tem complexidade de tempo $O(\log n)$.

Algoritmo CGM proposto

Cada processador contém blocos horizontais de n/p linhas da matriz A e vetor b.

Veja o exemplo para $n = 8$ e $p = 2$:

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ -1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 2 & -4 & 1 & 0 & 0 & 0 & 0 & \Rightarrow P_1 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & \\ -- & -- & -- & -- & -- & -- & -- & -- & \\ 0 & 0 & 0 & 1 & 3 & 2 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 3 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 2 & 5 & 2 & \Rightarrow P_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & \end{pmatrix}$$

$$b = \begin{pmatrix} 1 & \\ 0 & \\ 1 & \Rightarrow P_1 \\ 3 & \\ --- & \\ \frac{18}{23} & \\ 1 & \\ 0 & \Rightarrow P_2 \\ 0 & \end{pmatrix}$$

Algoritmo:

1. Cada processador aplica redução par-ímpar para eliminar todas as linhas exceto a primeira e última (sobram 4 incógnitas).

2. Cada processador P_i envia suas duas equações (com 4 incógnitas) ao processador 1. (Processador 1 recebe um sistema tridiagonal com $2p$ equações e $2p$ incógnitas.)

3. Processador 1 resolve o sistema localmente por redução par-ímpar obtendo as $2p$ incógnitas.

4. Processador 1 envia a cada processador P_i as 4 incógnitas computadas.

5. Cada processador realiza localmente um processo inverso da redução par-ímpar, usando a solução das duas equações para resolver as demais equações.

Implementação

Parsytec PowerXplorer de 16 nós cada nó com:

. um processador PowerPC601 e um processador de comunicação T805

. 32 Mbytes de memória local

Implementação feita com PVM e também interface nativa Parix (dando resultados semelhantes).

Resultados

Tempos de execução (em micro-segundos):

n	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$
1024	8727	5767	3372	3080	4714
2048	18896	11665	5887	3885	4794
4096	39032	23901	11730	6457	5903
8192	78618	47190	23654	12358	8077
16384	155944	95979	46693	24191	14005
32768	308148	189749	93867	47397	25552
65536	613099	375037	185122	94915	48508
131072	1223813	746283	365247	187116	95927
262144	2429358	1496103	727908	369437	187960
524288	4840124	2968610	1457379	736184	369990

Conclusão

. Algoritmo usa apenas um número constante de rodadas de comunicação com a transmissão de $O(p)$ dados por rodada.

. A complexidade de tempo do algoritmo é de $O(\log n/p)$.

. O ganho é quase linear para valores grandes de n.

Conclusão

. Algoritmo usa apenas um número constante de rodadas de comunicação com a transmissão de $O(p)$ dados por rodada.

. O ganho é quase linear para valores grandes de n.