PROCESSAMENTO PARALELO ASSÍNCRONO DE UM ALGORITMO MULTINÍVEL+

*C.P. BOTTURA, *J.T. COSTA FILHO, **A.P. RIBEIRO COSTA
*Faculdade de Eng. Elétrica - DMCSI - UNICAMP
Cx. P. 6101 - 13.081 - Campinas, SP
**CPqD/TELEBRÁS

RESUMO

Neste trabalho, analisamos a concorrência mestre-escravo existente na paralelização de um algoritmo com o objetivo de alterar a estrutura de cálculo para reduzir o efeito de sincronização sobre a execução computacional de um algoritmo multinível implementado no Processador Preferencial. Uma análise de desempenho com relação a tempo de processamento, eficiência computacional e comunicação de dados é apresentada.

ABSTRACT

With the objective of changing the calculation structure, in order to reduce the synchronization effects on the computer execution of a multilevel algorithm implemented on the Preferential Processor, in this article the master - slave concurrence there is in the parallelization of a synchronous algorithm is analysed. A performance analysis with respect to processing time, computational efficiency and data communication is presented.

1. INTRODUÇÃO

A elaboração de algoritmos paralelos torna-se tão importante quanto o desenvolvimento de arquiteturas para o processamento paralelo. Uma vez que a implementação e desempenho de algoritmos paralelos dependem significativamente da arquitetura, é necessário especificá-la para computação paralela quando consideramos algoritmos paralelos. Não há no entanto nenhum método universal para o desenvolvimento de algoritmos paralelos. Uma técnica para se obter algoritmos paralelos explora o grau de paralelismo existente em algoritmos sequenciais. O emprego desta técnica tem permitido detectar o paralelismo intrínseco de algoritmos sequenciais e avaliar o grau de concorrência resultante. O paralelismo intrínseco de um algoritmo sequencial reflete a intensidade da comunicação existente no algoritmo paralelizado.

Neste contexto, analisamos a concorrência mestre-escravo (M/E) existente em algoritmos de otimização multinível de sistemas de grande porte resultante da paralelização. Objetivando obter um melhor desempenho computacional do algoritmo sobre o Processador Preferencial (PP), alteramos a estrutura de cálculo do algoritmo e reduzimos o grau de concorrência através da exploração do paralelismo intrínseco.

Na seção 2, descrevemos a estrutura do Processador Preferencial para processamento paralelo que utilizamos. Na seção 3, definimos a concorrência mestre-escravo exibida em alguns algoritmos de controle ótimo hierárquico. Na seção 4, formulamos, resumidamente, uma metodologia de otimização e o algoritmo em dois níveis. Na seção 5, adotamos um critério de partição para obtermos uma execução assíncrona do algoritmo paralelizado sobre a arquitetura especificada. Na seção 6, analisamos o desempenho.

2. ESTRUTURA DO PROCESSADOR PREFERENCIAL

As implementações descritas neste trabalho foram realizadas sobre o Processador Preferencial (PP), hardware desenvolvido pelo Centro de Pesquisa e Desenvolvimento da TELEBRÁS - CPqD.

A configuração do hardware está baseada numa arquitetura modular constituída de várias placas processadoras de 16 bits, placas de memórias e um conjunto de placas para controle de dispositivos periféricos, organizadas em torno de um barramento assíncrono dedicado de alta velocidade (10 M bytes), permitindo endereçamento de até 16 M bytes de memória. A possibilidade de operação em multiprocessamento deve-se a presença de um arbitrador de barramento em uma unidade central de processamento (UCP).

A estrutura do sistema para processamento paralelo com médio acoplamento entre as unidades processadoras consiste de: a)Unidade Central de Processamento (PP-UCP): microcomputador baseado no microprocessador iAPX286 e co-processador numérico 80287; com memória RAM local de 512 bytes e com memória EPROM de 64 K bytes; b) unidade de controle de Discos Rígidos e Flexíveis (PP-DIS); c) unidade de Rastreamento (PP-RAS) - permite monitorar as ocorrências no barramento e armazenar dados relativos a 2084 ciclos do barramento (PP-BAR); d) unidade de conversão de barramento PP-BAR ao IBM-PC (PP-CON) - permite a ligação de placas periféricas comerciais(IBM-PC) ao barramento PP-BAR.

Para a arquitetura apresentada, Fig. 1: 1) Todas as UCPs têm acesso direto com a mesma prioridade à memória RAM (128 kbytes) existente na

+ Este trabalho contou com a colaboração de pesquisadores e engenheiros do CPqD, CTI e CEPEL.

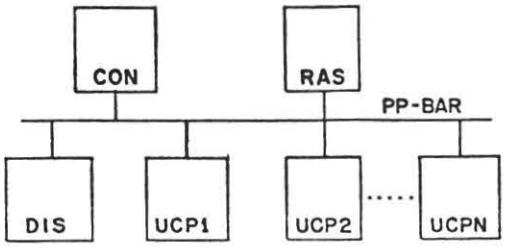

Fig. 1 - Estrutura do PP para Processamento Paralelo.

placa PP-CON, utilizada como área comum de dados (memória global); 2) memória global é utilizada apenas para comunicação de dados e para sincronização de programas; 3) A sincronização e a comunicação entre os programas independe da localização dos mesmos e são realizadas através de comandos de atribuição presentes na linguagem PASCAL; 4) Um sistema operacional monoprocessador compatível com o DOS é carregado em cada UCP.

3. CONCORRÊNCIA MESTRE-ESCRAVO

Um problema de controle ótimo de sistema de grande porte pode ser resolvido, usando a concorrência mestre-escravo, se o problema pode ser decomposto em subproblemas independentes. A independência significa que a solução de um subproblema é desconhecida de outros subproblemas. Neste caso, cada escravo pode ser designado para resolver um subproblema.

O escravo requer uma especificação do mestre para o subproblema e por sua vez informa ao mestre a solução de seu subproblema. Destas soluções dos subproblemas, o mestre determina a solução global.

3.1. Definição da Concorrência M/E

Um programa sequencial especifica uma execução sequencial de uma lista de comandos, sendo que a execução deste programa é denominada processo. Um programa concorrente especifica dois ou mais programas sequenciais que podem ser executados concorrentemente como processos paralelos.

Um programa para resolver um problema exibe uma concorrência M/E se os processos que existem durante a execução do programa podem ser particionados em processo mestre e processos escravos tal que: 1) O problema pode ser decomposto em subproblemas independentes de forma que cada escravo resolva um único subproblema e manipule somente as variáveis do programa que estão relacionadas com aquele subproblema; 2) O mestre comunica-se com cada escravo somente uma vez, antes que qualquer mudança ocorra no processo escravo; 3) Cada escravo comunica-se com o mestre somente para enviar a solução de seu subproblema.

Como consequência da definição: os escravos são disjuntos; os escravos não se comunicam uns com os outros; a ordem de execução dos escravos é irrelevante para a solução global tal que nenhuma sincronização entre os escravos é estabelecida, porém há necessidade de sincronizar os escravos e o mestre quando os escravos comunicam suas soluções ao mestre; somente o mestre determinará que a solução global foi atingida e por conseguinte os escravos têm suas atividades concluídas.

Neste trabalho, consideramos apenas um nível para os escravos, uma vez que os problemas de comunicação e sincronização já são bastante significativos.

4. O MÉTODO PREDIÇÃO DE CO-ESTADO

A metodologia de otimização aqui tratada e o algoritmo em dois níveis resultante são bastante estudados em [1,2,3].

4.1. Formulação do Problema

O problema global consiste em minimizar o critério de desempenho

$$J = \sum_{K=0}^{T-1} \frac{1}{2} \left[\|X(K)\|_Q^2 + \|U(K)\|_R^2 \right] \qquad (1)$$

sujeito as seguintes restrições: para $K = 0,\ldots, T-1$

$$X(K+1) = AX(K) + BU(K);\ X(0) = X_o \qquad (2)$$

onde $X \in R^n$, $U \in R^m$, $Q \geq 0$, $R > 0$

4.2. Solução Decomposta

O sistema dinâmico (2) é decomposto em N subsistemas descritos por: $i = 1,\ldots,N$ e $K = 0,\ldots,T-1$

$$X_i(K+1) = A_i X_i(K) + B_i U_i(K) + Z_i(K);$$

$$X_i(0) = X_{io} \qquad (3)$$

$$Z_i(K) = g_i(X(K)) \qquad (4)$$

onde $\sum_{i=1}^{N} n_i = n$, $\sum_{i=1}^{N} m_i = m$ e $Z_i(K)$ representa a interação entre os subsistemas.

Para cada subsistema, associamos o problema de controle:

$$\text{minimizar } J_i = \sum_{K=0}^{T-1} \frac{1}{2} \left[\|X_i(K)\|_{Q_i}^2 + \|U_i(K)\|_{R_i}^2 \right] \qquad (5)$$

O procedimento de solução pelo Princípio do Máximo compreende: definição do Hamiltoniano $H = \sum_{i=1}^{N} H_i$, onde o sub-Hamiltoniano é:

$$H_i = \frac{1}{2} \|X_i(K)\|_{Q_i}^2 + \frac{1}{2} \|U_i(K)\|_{R_i}^2 +$$

$$+ \lambda_i'(K+1)[A_i X_i(K) + B_i U_i(K) + Z_i(K)] +$$

$$+ \beta_i'(K)[Z_i(K) - g_i(X(K))] \qquad (6)$$

onde $\lambda_i(K)$ é o vetor de co-estado de dimensão n_i e $\lambda_i(K)$ é o vetor multiplicador de Lagrange de dimensão n_i ; bem como a satisfação das condições de otimalidade:

$$\frac{\partial H_i}{\partial X_i(K)} - \lambda_i(K) = 0 \quad ; \quad \lambda_i(T) = 0$$

$$\frac{\partial H_i}{\partial U_i(K)} = 0 \; ; \quad \frac{\partial H}{\partial \beta_i(K)} = 0 \; ; \quad \frac{\partial H_i}{\partial Z_i(K)} = 0 \qquad (7)$$

As equações (7) são resolvidas iterativamente em dois níveis, juntamente com (3), tal que para i = 1,...,N ; K = 0,...,T-1 o algoritmo toma a forma:

- NÍVEL 1 (Subproblemas)

$$X_i(K+1) = A_iX_i(K) + B_iU_i(K) + Z_i(K); \; X_i(0)=X_{io} \qquad (8)$$

$$R_iU_i(K) + B_i'\lambda_i(K+1) = 0 \qquad (9)$$

$$Z_i(K) - g_i(X(K)) = 0 \qquad (10)$$

$$\beta_i(K) + \lambda_i(K+1) = 0 \qquad (11)$$

- NÍVEL 2 (Coordenador)

$$\lambda_i(K) = Q_iX_i(K) + A_i'\lambda_i(K+1) - \frac{\partial g_i(X(K))}{\partial X_i(K)} \cdot \beta_i(K) - \sum_{j\neq i}^{N} \frac{\partial g_j'(X(K))}{\partial X_i(K)} \cdot \beta_j(K) \; ;$$

$$\lambda_i(T) = 0 \qquad (12)$$

5. IDENTIFICAÇÃO DA CONCORRÊNCIA M/E E DO PARALELISMO INTRÍNSECO

A estrutura de decomposição do problema de otimização reflete um forte procedimento interativo entre os subproblemas em cada iteração. A paralelização do algoritmo segundo a estrutura em dois níveis (Fig. 2) apresentada leva a uma execução concorrente síncrona dos subproblemas : muitos pontos de sincronização; um baixo grau de paralelismo; ociosidade das unidades de processamento e uma sobrecarga com a comunicação.

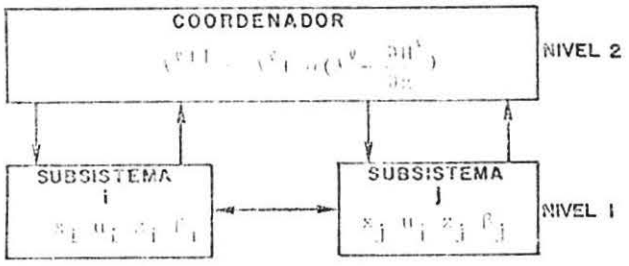

Fig. 2 - Estrutura em Dois Níveis.

Concluímos, portanto, pela necessidade de um critério de partição do algoritmo que será baseado na concorrência M/E da estrutura de otimização e na obtenção do paralelismo intrínseco, objetivando realizar o processamento paralelo assíncrono dos subproblemas bem como aumentar o grau de paralelismo.

O critério adotado compreende:

a) Obtenção da concorrência M/E: consiste da decomposição do sistema:

$$S \; : X(K+1) = AX(K) + BU(K) + Z(K) \qquad (13)$$

em N subsistemas

$$S_i \; : X_i(K+1) = A_iX_i(K) + B_iU_i(K) + Z_i(K) \; ; \quad (i = 1,...,N) \qquad (14)$$

mutuamente disjuntos, tal que: 1) $S = \cup_{i=1}^{N} S_i$; 2) Para qualquer i,j com i,j=1,...,N e $j \neq i$, S_i e S_j não tenham elementos comuns; 3) S_i é associado com o critério de desempenho (5) definindo o subproblema i.

b) Paralelismo intrínseco: consiste na distribuição da tarefa de coordenação C em N subtarefas C_i(i=1,...,N) tal que: 1) $C = \cup_{i=1}^{N} C_i$; 2) Para qualquer i,j com i,j = 1,..., N e $i \neq j$, C_i e C_j tenham somente os vetores X(K) e $\beta(K)$ comuns.

Para um entendimento mais detalhado da validação deste critério do ponto de vista algoritmico, sugerimos as referências [1,3].

A partir do critério de partição adotado, definimos, para processamento paralelo, as tarefas:

TAREFA i(i = 1,...,N)

. inicialização das trajetórias $\lambda(K)$ e Z(K)

. coletar da TAREFA j(j = 1,..., N , $j \neq i$), $X_j(K)$ e $\beta_j(K)$

. calcular $U_i(K)$, $\beta_i(K)$, $X_i(K+1)$, $\lambda_i(K)$ e $Z_i(K)$

. enviar a TAREFA j, $X_i(K)$ e $\beta_i(K)$

. testar a iteração ℓ se:

$$\left| \begin{matrix} \lambda_i^{\ell+1}(K) - \lambda_i^{\ell}(K) \\ Z_i^{\ell+1}(K) - Z_i^{\ell}(K) \end{matrix} \right| < \varepsilon .$$

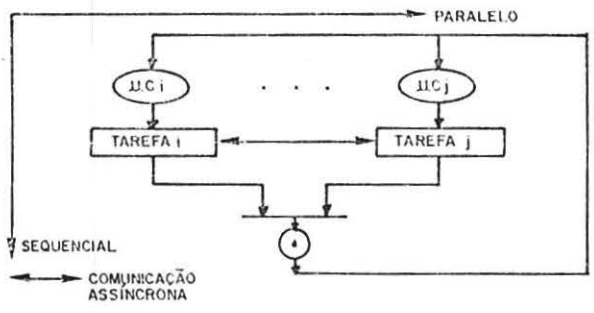

Fig. 3 - Paralelização.

A estrutura de paralelização final compreende a concorrência M/E e a distribuição da tarefa de coordenação pelos N subproblemas de otimização conforme a Fig. 4.

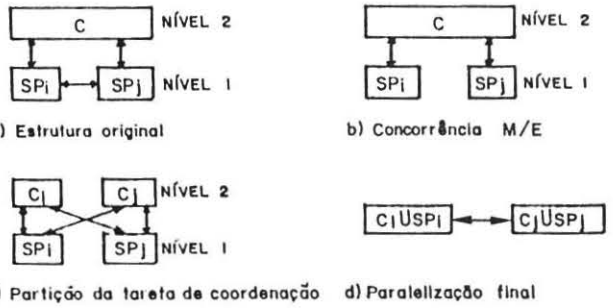

Fig. 4 - Etapas de Partição e Paralelização.

Na Fig. 4a representamos esquematicamente a estrutura de paralelização do algoritmo com : i) comunicação síncrona entre os subproblemas e o coordenador; ii) comunicação síncrona entre os subproblemas (SP_i e SP_j). A estrutura representada na Fig. 4b, resultante do critério de partição (5.a) adotado exibe efetivamente a concorrência M/E; esta estrutura tem apresentado um desempenho superior ao da estrutura da Fig. 4a. Nas Figs. 4c e 4d, etapas de partição da tarefa de coordenação e paralelização final respectivamente, levamos em consideração a distribuição da tarefa de coordenação e a conseqüente eliminação da comunicação síncrona entre SP_i e C_i e entre SP_j e C_j, obtendo a estrutura definitiva da Fig. 4d. Com esta última estrutura conseguimos reduzir os requisitos de comunicação entre as tarefas paralelas, limitando ao mínimo o procedimento interativo entre elas. Os experimentos que realizamos mostram que há ganho substancial na velocidade de processamento.

6. ANÁLISE DE DESEMPENHO COMPUTACIONAL

O tempo de computação no processamento paralelo compreende o tempo de comunicação e de sincronização e o tempo de execução. Supondo para o problema de otimização proposto e para as condições de inicialização dados que o tempo total de execução na implementação sequencial do algoritmo paralelizado T_1 seja:

$$T_1 = \sum_{\ell=0}^{L-1} \{ \sum_{i=1}^{N} F(n_i) + t_c \} d_\ell$$

onde $F(n_i)$ é uma função não linear da dimensão do i-ésimo subsistema, t_c é o tempo necessário para a execução da tarefa de coordenação, L é o número de iterações e $0 < d_\ell \leq 1$ é uma variável que depende do tempo de solução das trajetórias, e que o tempo total de execução do algoritmo paralelizado em N processadores T_N seja:

$$T_N = \sum_{\ell=0}^{L-1} \{(\max\{F(n_1),\ldots,F(n_N)\} + t_c/N\} d_\ell$$

definimos o ganho de velocidade de processamento do algoritmo paralelo como: $g_N = T_1/T_N \leq N$.

Definimos a eficiência computacional do algoritmo paralelo como: $E_N = g_N/N$. Supondo que os subsistemas têm a mesma dimensão, concluímos que idealmente $E_N = 1$.

Na arquitetura de computação especificada a comunicação entre as tarefas se realiza através da troca de dados na memória global. Os dados utilizados para sincronizar o acesso às variáveis compartilhadas consistem de somente alguns bytes e portanto não são considerados na determinação da quantidade total D de dados transferidos por iteração na implementação da estrutura da Fig. 4d e dada por:

$$D = T \cdot B \sum_{i=1}^{N} (2n_i)$$

onde B é o número de bytes necessário para armazenar uma variável.

7. EXEMPLO NUMÉRICO

$$\text{sistema dinâmico} \begin{cases} X_1(K+1) = 0,9X_1(K) + 0,1X_2(K) + 0,1U_1(K) \\ X_2(K+1) = 0,2X_1(K) + 0,1X_2(K) + \\ \qquad + 0,1U_2(K) - 0,1X_2^2(K) \end{cases}$$

critério de desempenho:

$$J = \sum_{K=0}^{50} \frac{0,1}{2} \{X_1^2(K) + X_2^2(K) + 2U_1^2(K) + U_2^2(K)\}$$

Particionamos o sistema em dois subsistemas:

Subsistema 1: $X_1(K)$ e $U_1(K)$ e

Subsistema 2: $X_2(K)$ e $U_2(K)$.

O algoritmo paralelizado convergiu em 15 iterações para $\varepsilon = 0,001$.

Tabela 1

	TP	TE	TS	MT	MG
SEQ	0,97			3,00	
PAR	0,50	29,17	36,71	5,40	13,80

SEQ: implementação sequencial (1UCP) ; PAR: implementação paralela (2UCPs); MT: memória total para armazenar os programas em kbytes: MG: quantidade total de dados compartilhados em Kbytes; TP: tempo total de computação (segundos); TE : tempo de execução da TAREFA i por iteração (milisegundos); TS: tempo total gasto com sincronização e comunicação (milisegundos).

8. CONCLUSÃO

Do critério de partição adotado, com base na

descentralização da estrutura de otimização, a paralelização obtida tornou-se computacionalmente mais atrativa para o processamento paralelo. A sobrecarga resultante da comunicação e da sincronização entre as tarefas foi significativamente reduzida. A atividade de processamento das tarefas foi aumentada e, por conseguinte, a velocidade de processamento global.

9. BIBLIOGRAFIA

[1] Xinogalas, T.C.; Dasigi, S. e Singh, M.G., "Coordination in Hierarchical Algorithms", IEEE Transactions on Systems, Man and Cybernetic, vol. SMC-13, nº 3, pág. 397-405, May/June 1983.

[2] Mahmoud, M.S., "Dynamic Multilevel Optimization for a Class on Nonlinear Systems", International Journal of Control, vol. 30, nº 6, pág. 927-948, December 1979.

[3] Bottura, C.P.; Costa Filho, J.T.; "Controle Hierárquico Via Precisão de Co-estado Utilizando um Sistema de Múltiplos Microcomputadores", 3ª Congresso Latino-Americano de Automática, Viña del Mar, Chile, Outubro 1988, (aceito para publicação).

[4] Bottura, C.P.; Costa Filho, J.T., "Processamento Paralelo de Algoritmo de Controle Hierárquico", 7º Congresso Brasileiro de Automática, Agosto 1988.

[5] Kung, H.T.; "The Structure of Parallel Algorithms", Advances in Computers", vol. 19, pp. 65-108, 1980.

[6] Bottura, G.P.; Costa Filho, J.T., "On Parallel Computing for a Multilevel Optimization Algorithm", IFAC Workshop: Control Applications of Nonlinear Programming, June 21-27/ 1988, Tbilisi, Russia.

[7] Bottura, C.P.; Costa Filho, J.T., "Computação de Algoritmo de Otimização Hierárquica Via Multiprogramação", X Congresso Nacional de Matemática Aplicada e Computacional. vol. 1, pág. 82-88, 1987.

[8] Bottura, C.P.; Costa Filho, J.T., "Programação Paralela de Algoritmo de Otimização Multinível", I Simpósio Brasileiro de Arquitetura de Computadores - Processamento Paralelo, comunicação, 1987.

[9] Zenios, S.A.; Mucvey, J.M., "A Distributed Algorithm for Convex Network Optimization Problems", Parallel Computing, vol. 6, nº 1, pp. 45-56, 1988.

[10] Kanakia, H.R.; Tobagi, F.A., "On Distributed Computations with Limited Resources", IEEE Transactions on Computers, vol. C-36, nº 5, pp. 517-528, 1987.