Solução Paralela Para Sistemas de Equações Integrais Usando RPC Assíncrona

Daniela Saccol Peranconi, Márcia Cristina Cera,
Marcelo Pasin

Universidade Federal de Santa Maria - Laboratório de Sistemas de Computação
Faixa de Camobi, Km 9 - CEP 97105-900 - Santa Maria - RS - Fone/Fax: (55) 220 8523
{dany, cera, pasin}@inf.ufsm.br

Introdução

Este trabalho apresenta a paralelização de uma aplicação que encontra soluções para sistemas de equações integrais. Tal aplicação consiste na resolução de um grande conjunto de tarefas de integração numérica. A escolha deste problema deu-se devido a demanda de tempo de processamento ser relativamente grande para encontrar as soluções do sistema. Além disso, o algoritmo seqüencial da aplicação possui pouca dependência de dados, gerando muitas oportunidades de paralelização.

O cálculo do problema numérico pode ser dividido em duas partes, a resolução das equações pelo método do Ponto Fixo [BUR 97] e a resolução das integrais pelo método de Simpson [LEI 82]. Optou-se por realizar a paralelização da resolução das equações por motivos que serão especificados no decorrer deste texto.

Método Numérico de Solução

A aplicação a ser paralelizada foi originalmente desenvolvida para encontrar as transições de fases das propriedades de supercondutores em condições ambiente [PAS 01]. O problema ao qual o algoritmo se destina a resolver, consiste em um sistema de quatro equações e quatro variáveis, sendo que todas as equações são integrais múltiplas. As mesmas podem ser visualizadas na figura 1. As equações dependem dos valores de quatro variáveis, denominadas η, Qz, Rz e Rr, bem como de H, T e G que representam, respectivamente, campo magnético, temperatura e constante de acoplamento do sistema físico em questão.

Este problema caracteriza-se por apresentar um comportamento irregular. Tal comportamento deve-se a nem sempre ser encontrada uma transição de fase em um intervalo de cálculo e às quatro funções integrais serem diferentes. Além disso, o número de iterações e de intervalos de integração necessários para resolver o sistema variam de acordo com os valores iniciais dos parâmetros de cálculo.

Algoritmo Seqüencial

O algoritmo seqüencial [PAS 01] foi decomposto em partes fundamentais. A figura 2 mostra o procedimento principal.

$$\eta = \int_{-\infty}^{+\infty} D(w)\frac{\sinh(\beta\mu')}{I_\beta}$$

$$Q_z = K_q \int_{-\infty}^{+\infty} D(w)\left[\frac{\int_0^{+\infty} uD(u)\int_{-\infty}^{+\infty} D(v)S_\beta\theta}{I_\beta}\right]^2$$

$$R_z = \frac{1}{K_r}\int_{-\infty}^{+\infty} D(w)\frac{\int_0^{+\infty} uD(u)\int_{-\infty}^{+\infty} vD(v)S_\beta\theta}{I_\beta}$$

$$Rr = \frac{1}{4}\int_{-\infty}^{+\infty} D(w)\frac{\int_0^{+\infty} u^3D(u)\int_{-\infty}^{+\infty} D(v)S_\beta}{I_\beta}$$

Figura 1: Sistema de equações integrais.

```
main procedure:
        i = 0;
        while not enough point
                find - transition η[i], Qz[i], Rz[i], R[i]
                i = i + 1
        plot surfaces
```

Figura 2: Procedimento principal.

```
find - transition procedure
        k = 0
        for H = Hi to Hf step sH
                for T = Ti to Tf step sT
                        for G = Gi to Gf step sG
                                solve η[k], Qz[k], Rz[k], R[k]
                                k = k + 1
        find j such as there is a phase transitions in η[j], Qz[j], Rz[j], R[j]
        return η[j], Qz[j], Rz[j], R[j]
```

Figura 3: Cálculo da transição de fase.

```
solve procedure
        repeat
                η  = ∫ fη( η , Qz, Rz, R, H, T, G)
                Qz = ∫ fQz( η , Qz, Rz, R, H, T, G)
                Rz = ∫ fRz( η , Qz, Rz, R, H, T, G)
                Rr = ∫ fR( η , Qz, Rz, R, H, T, G)
        until η, Qz, Rz e R converge
```

Figura 4: Cálculo das integrais.

Para encontrar uma transição de fase (figura 3), o sistema de equações é calculado milhares de vezes para diferentes valores de um intervalo de H, T e G. A fim de calcular os valores das quatro equações integrais, são feitas várias iterações, até as equações convergirem para uma solução (figura 4).

Implementação

A paralelização do método seqüencial foi implementada utilizando a CE [CER 02] como biblioteca de comunicação, por esta apresentar características que atendem às necessidades da aplicação. As facilidades que esta bilbioteca oferece são: um conjunto reduzido de funções, facilidade de programação, eficiência e comunicação através de chamadas remotas de procedimentos (RPC's) [BLO 92] assíncronas [WIL 99].

O código mostrado na seção anterior soluciona um problema específico, formado por um sistema de quatro equações com quatro variáveis e permite a variação de somente três coeficientes. Generalizou-se a aplicação de forma a possibilitar o cálculo, tanto de um número qualquer de coeficientes como de um número qualquer de variáveis.

A fim de proporcionar a generalização do número de coeficientes, substituiu-se os três laços *for* (mostrados na figura 3) por chamadas recursivas da função intervalo (mostrada na figura 5). Já, para possibilitar que um número qualquer de variáveis pudesse ser calculado, foi incluído ao algoritmo mostrado na figura 4, um laço *for*, delimitado por *n*, que representa o número de variáveis especificado pelo usuário, como apresentado na figura 6.

```
intervalo ( i )
        for k_i = k_imin to k_imax
            if i < n°. de coeficientes
                    intervalo (i + 1)
            else
                    pontofixo (k_0, ... , k_n)
```

Figura 5: Função intervalo.

```
pontofixo ( )
        repete enquanto erro > tolerância
                for i = 0 to n
                    v_i = ∫f_i (v_0, ... , v_n, k_0, ... , k_n)
```

Figura 6: Função pontofixo.

Os cálculos necessários para solucionar o problema da aplicação possuem comportamento irregular, ou seja, a demanda de tempo de processamento varia conforme o conjunto de parâmetros iniciais e tal demanda não pode ser prevista de antemão. Procurando fornecer um melhor equilíbrio da carga de processamento foi implementado um banco de processadores. Através deste, cada um dos processadores disponíveis no inicío da execução recebe uma tarefa. Enquanto existirem tarefas a serem executadas, elas serão distribuídas aos processadores a medida que estes tornarem-se disponíveis novamente.

Avaliação de Desempenho

Na execução dos testes da aplicação foram utilizadas máquinas biprocessadas Pentium III de 1 GHz, com 786 Mbytes de memória principal, 512 kbytes de memória cache, adaptador Gigabit Ethernet, modelo 3Com 3C996-T e sistema operacional Red Hat Linux 7.2.

A análise foi feita sobre os resultados de testes executados em um determinado conjunto de dados iniciais que compreendem um intervalo de cálculo crítico do problema da aplicação. Para estes dados, obteve-se um tempo de execução seqüencial de 1 hora e 58 minutos (118 minutos). Pode ser observado na figura 7 que, já com dois nós, o tempo de execução reduziu para 1 hora e 20 minutos (80 minutos). À medida em que foram sendo incluídos nós, o tempo total de execução continuou decrescendo, atingindo-se o objetivo pelo qual paralelizou-se a aplicação.

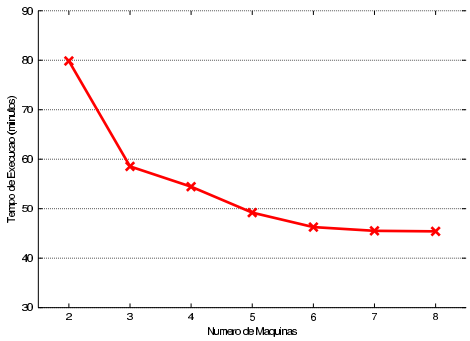

Figura 7: Gráfico do tempo de execução com CE.

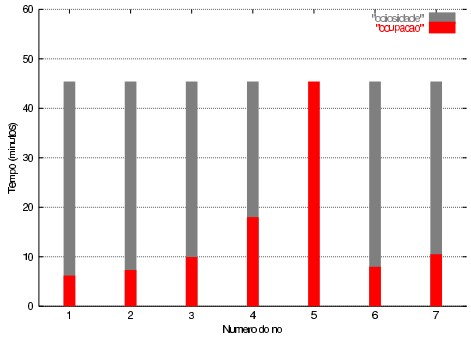

Figura 8: Gráfico da ociosidade X ocupação com 8 máquinas.

A distribuição da carga de processamento da aplicação com CE pode ser vista no gráfico da figura 8. Nele está representada uma execução com oito máquinas, sendo discri-

minada a carga em cada um dos nós, com exceção do nó 0 (zero) por este responsabilizar-se, apenas, pelo gerenciamento da execução dos outros nós. Para este caso, o tempo de ocupação do nó 5 (cinco) prevalece sobre os demais em virtude da irregularidade dos cálculos do problema e faz com que os outros nós permaneçam ociosos aguardando sua conclusão. Para amenizar o tempo de ociosidade, uma redivisão de cargas poderia ser empregada. A partir do momento que um nó ficasse ocioso, ele receberia uma parcela da carga de processamento de um outro nó que ainda estivesse executando. No caso desta aplicação, a solução seria paralelizar a execução do método de *Simpson* assim que existissem nós ociosos. Por falta de tempo, esta paralelização não foi totalmente implementada e não serão apresentados, aqui, os resultados.

Conclusão e trabalhos futuros

O desenvolvimento deste trabalho objetivou a paralelização e generalização de um método numérico que soluciona sistemas de equações integrais. A generalização possibilitou encontrar solução para sistemas de equações integrais com qualquer número de variáveis e coeficientes. Com a paralelização do cálculo das equações, foi possível reduzir o tempo total de processamento da aplicação.

No entanto, constatou-se que a maioria dos nós encontrava-se ociosa por um tempo significativamente grande. Este tempo de ociosidade poderia ser melhor aproveitado com a implementação da paralelização do método de integração, onde os nós ociosos receberiam parte do trabalho dos nós mais sobrecarregados.

Como alternativa para um trabalho futuro, há a possibilidade da realização de um estudo de métodos eficazes para amenizar o problema do balanceamento da carga de processamento. Também pode-se substituir o método de *Simpson* por outros métodos de integração que possam tornar a aplicação mais eficiente.

Referências

[BLO 92] BLOOMER, J. **Power Programming with RPC**. 1.ed. O'Reilly & Associates, Inc. 1992.

[BUR 97] BURDEN, R. L.; FAIRES, J.D. **Numerical Analysis**. ITP, 1997.

[CER 02] CERA, M. C. **libce: Uma Biblioteca de Comunicação Eficiente**. Trabalho de Graduação. Universidade Federal de Santa Maria, 2002.

[LEI 82] LEITHOLD, L. **O Cálculo com Geometria Analítica**. Vol.1. 2.ed. Harbra, 1982.

[PAD 01] PADOIN, E. L. **Estudo da Paralelização de Uma Aplicação com Alto Custo Computacional Utilizando Sistemas Distribuídos**. Dissertação (Mestrado). Universidade Federal de Santa Maria, 2001.

[PAS 01] PASIN, M.; PADOIN, E.L. **A parallel solution for systems of integral equations**. Anais SBAC-PAD, 2001.

[WIL 99] WILKINSON, B.; ALLEN, M. **Parallel Programming - Techniques and Applications Using Networked Workstations and Parallel Computers**. 1.ed. Prentice Hall, 1999.