Estudo Comparativo na Exploração de Paralelismo em Ambientes de Memória Compartilhada

Marcelo C. Zembrzuski, Lisandro L. Trarbach, André L. Martinotto, Delcino Picinin, Tiarajú A. Diverio

Instituto de Informática - Universidade Federal do Rio Grande do Sul
CP 15064, 91501-970, Porto Alegre, RS, Brasil
{mczembrzuski, lltrarbach, almartin, picinin, diverio} @inf.ufrgs.br

Introdução

A forma mais comum de exploração de paralelismo em máquinas com memória compartilhada é o uso de múltiplas *threads*. Dentro deste contexto, foram desenvolvidas diversas bibliotecas que oferecem suporte para criação e manipulação de *threads*. Entre essas as mais utilizadas para desenvolvimento de aplicações paralelas são a biblioteca Pthreads (POSIX *threads*) e a biblioteca OpenMP.

O objetivo deste trabalho é realizar um estudo comparativo entre essas bibliotecas. Para tanto, foram escolhidas duas aplicações. A primeira delas um algoritmo de convolução de imagens utilizado para a filtragem de imagens. A segunda, o método do GC, um método iterativo muito utilizado em aplicações científicas.

O artigo está estruturado da seguinte forma. Primeiramente são apresentas as ferramentas utilizadas no desenvolvimento deste trabalho. Em seguida, é feita uma breve descrição das aplicações desenvolvidas. E por fim, são apresentados os resultados obtidos.

Ferramentas Utilizadas

Pthreads e OpenMP são duas das bibliotecas mais utilizadas no desenvolvimento de aplicações paralelas para a exploração do paralelismo em máquinas SMP. Nessa seção são analisadas essas duas bibliotecas.

Pthread

Desenvolvido pelo comitê POSIX (*Portable Operating System Interface*), a Pthread é um padrão adotado para o desenvolvimento de bibliotecas de *threads* [NIC 96].

Na programação com a Pthreads na linguagem C, uma *thread* é um procedimento que ao ser invocado executa paralela ou concorrentemente aos outros fluxos já existentes. A concorrência ou paralelismo depende do número de processadores disponíveis e do tipo de *thread* utilizada. Esta nova *thread* ou fluxo tem acesso a todas as variáveis globais ou declaradas dentro de seu escopo [LEW 98].

A abordagem adotada na paralelização com múltiplas *threads* consiste na criação de uma segunda *thread*, semelhante a já existente. As múltiplas *threads*, quando necessário, são sincronizadas através de *mutexes* e variáveis de condição.

OpenMP

OpenMP é uma API (*Aplication Program Interface*) para desenvolvimento de aplicações paralelas que fazem uso de memória compartilhada. Essa foi idealizada por um grupo de grandes fabricantes de software e hardware e fornece um conjunto de primitivas para criação, manipulação e gerenciamento de threads.

Em OpenMP o paralelismo pode ser obtido em dois diferentes níveis. Em um primeiro nível, através da paralelização de laços (*loop level*). Neste nível basta informar ao compilador o início e o fim do código (laços) a ser paralelizado e o número de *threads* que executarão o laço. Esta forma de paralelização é trivial, uma vez que a geração do código paralelo é feita pelo próprio compilador.

O segundo nível (*parallel region*) é mais abrangente, e pode ser utilizado para a execução concorrente de um trecho genérico de código. Nesse nível, tem-se uma programação semelhante a biblioteca Pthreads (embora facilitada pelas diretivas do OpenMP) cabendo ao programador o gerenciamento de seções críticas, cuidados com condições de corrida, etc [OPE 2003].

Nesse trabalho para a paralelização do algoritmo de convolução, utilizou-se o primeiro nível do OpenMP (*loop level*). Já na paralelização do algoritmo do Gradiente Conjugado, utilizou-se o segundo nível do OpenMP (*parallel region*). A escolha da adoção dos níveis deve-se as características intrínsecas dos algoritmos paralelizados.

Aplicações Desenvolvidas

Nessa seção são introduzidas as aplicações utilizadas no desenvolvimento desse trabalho. Primeiramente, é feita uma descrição do algoritmo de convolução de imagem e, por fim, é descrito o método do GC.

Convolução de Imagens

A convolução de imagens é um método que permite o processamento de uma imagem de modo a satisfazer um objetivo específico, tal como um filtragem, realce ou restauração. Ela é definda sobre o domínio espacial e as abordagens nesse domínio são baseadas na manipulação direta dos *pixels*.

O algoritmo de convolução consiste basicamente em um produto escalar entre a vizinhança de um *pixel* (x,y) e uma máscara que varia conforme o objetivo do processamento. A máscara é uma matriz bidimensional com o mesmo tamanho que a vizinhança dos pixels da imagem original. Assim, o algoritmo de convolução baseia-se em quatro *for*'s aninhados, dois dos quais percorrem a imagem original e os outros dois percorrem a máscara de convolução [JAH 95]. Para o desenvolvimento deste trabalho utilizou-se uma máscara para a remoção do ruído de uma imagem.

Tamanho da Imagem	Seqüencial	Pthreads	OpenMP
640x480	0.4577674	0.2316300	0.3464998
800x600	0.8142230	0.4080632	0.5610098
1024x768	1.1787292	0.6102776	0.8903654
1600x1200	2.8728978	1.4554254	2.1556102
2048x2048	6.3899028	3.2047970	4.7185460

Tabela 1: Convolução de Imagens - OpenMP X Pthreads

Gradiente Conjugado (CG)

O GC é um método para a solução de sistemas de equações, iterativo não estacionário, pertencente à classe de métodos do subespaço de Krylov. Esse método é um dos mais eficientes para a solução de sistemas de grande porte e esparsos, onde a matriz de coeficiente é SDP.

O GC parte do princípio de que o gradiente, que é um campo vetorial, aponta sempre na direção mais crescente da função quadrática. O algoritmo do CG busca minimizar uma determinada função quadrática, de modo que se a matriz for SDP o gradiente dessa função $f(x)$ resume-se a $\bigtriangledown f(x) = b - Ax$. Então, como se deve minimizar o $\bigtriangledown f(x)$ deve-se determinar x tal que $\bigtriangledown f(x) = b - A.x = 0$, ou seja, $Ax = b$, significando que ao encontrar a solução do GC encontra-se a solução do sistema de equações.

O algoritmo do GC é constituído basicamente de operações matriciais, como por exemplo, a soma e o produto escalar de vetores, e a multiplicação matriz por vetor. Essa última operação é a de maior custo computacional [SHE 94].

Testes e Resultados Obtidos

Nesta secão são apresentados os resultados obtidos com as paralelizações desenvolvidas neste trabalho. Essas foram implementadas em linguagem C, utilizando o compilador *gcc* 2.95.4 sobre o sistema operacional Debian Linux com *kernel* 2.4.21. Para testes utilizou-se a biblioteca e o pré-compilador Omni 1.4a [OMN 2003]. Esses foram realizados utilizando um máquina Dual Pentium III 1.1GHz, com 1 GBytes de RAM e cache de 512KBytes. Os resultados apresentados representam a média de dez execuções.

Na tabela 1 podem ser vistos os tempos de execução do algoritmo de convolução para diferentes tamanhos de imagens. Observa-se que o desempenho obtido com Pthreads nessa implementação é muito superior ao obtido com a biblioteca OpenMP. Isto porque na versão que faz uso do OpenMP utilizou-se apenas o *loop level*. Nesse nível a geração do código paralelo é feita pelo próprio compilador.

Na tabela 2 podem ser vistos os resultados obtidos com as implementações desenvolvidas. Observa-se que os desempenhos obtidos com OpenMP utilizando o *parallel region* são similares aos obtidos com a biblioteca Pthreads.

Nº de Equações	Seqüencial	Pthreads	OpenMP
11.506	0.030048	0.014837	0.013507
46.024	0.155209	0.099174	0.099445
184.096	0.689793	0.445179	0.456486
736.384	2.811246	1.806116	1.865284

Tabela 2: Gradiente Conjugado - OpenMP X Pthreads

Conclusões

A partir dos resultados obtidos em testes, observou-se que a biblioteca OpenMP apresentou um baixo desempenho na aplicação de convolução de imagens em relação a mesma implementação com a biblioteca Pthreads. A forma de paralelização adotada com o OpenMP, nesse algoritmo, foi a *loop level*, onde a paralelização é feita de forma implícita pelo pré-compilador, e não é garantido o melhor uso dos recursos disponíveis.

Já os resultados obtidos com o algoritmo do GC foram satisfatórios. Neste caso utilizou-se o *parallel region*, onde a exploração do paralelismo é feita explicitamente pelo programador. Apesar do desenvolvimento da aplicação ser mais complexo, em relação ao *loop level*, o desempenho obtido é significativamente superior.

Um fator importante na comparação entre as bibliotecas OpenMP e Pthreads é a facilidade de utilização. O desenvolvimento de aplicações paralelas em OpenMP, mesmo no *parallel region*, é consideravelmente mais simples que na biblioteca Pthreads. Além de uma maior facilidade de programação o código fonte em OpenMP é muito mais legível e compacto. Conseqüentemente, em aplicações de grande porte facilita não só o desenvolvimento como, também, a depuração e manutenção do código.

Referências

[JAH 95] JAHNE, B. **Digital Image Processing**. [S.l.]: Springer-Verlag, 1995.

[LEW 98] LEWIS, B.; BERG, D. J. **Multithreaded Programming With Pthreads**. [S.l.]: Sun Microsystems Press, 1998. 382p.

[NIC 96] NICHOLS, B.; BUTTLAR, D.; FARRELL, J. **Pthreads Programming, A POSIX Standard for Better Multiprocessing**. [S.l.]: O'Reilly & Associates, 1996.

[OMN 2003] OMNI. **Omni OpenMP Compiler Project**. Disponível em: http://phase.hpcc.jp/Omni. Acesso em: out. 2003.

[OPE 2003] OPENMP. **OPENMP**: Simple, Portable, Scalable SMP Programming. Disponível em: http://www.openmp.org/. Acesso em: out. 2002.

[SHE 94] SHEWCHUK, J. R. **An Introduction to the Conjugate Gradient Method without the Agonizing Pain**. Disponível em: www.cs.cmu.edu/jrs/jrspapers. Acesso em: abr. 2001.