Programação Paralela:

O OpenMP

Silvio Romero de Araújo Júnior
Departamento de Engenharia Elétrica
Centro Universitário da FEI
São Bernardo do Campo, Brasil
silvioromerojr@yahoo.com.br

Reinaldo A. C. Bianchi
Departamento de Engenharia Elétrica
Centro Universitário da FEI
São Bernardo do Campo, Brasil
rbianchi@fei.edu.br

Resumo — Os programadores de aplicações para Machine Learning há muito reconhecem que hardware e software escaláveis são necessários para o processamento paralelo visando melhorias no desempenho dos programas. O OpenMP foi projetado para explorar características das arquiteturas de memória e CPUs compartilhadas. Este trabalho apresenta uma aplicação simples para ilustrar os conceitos de computação paralela com memória e CPU compartilhada por meio do OpenMP.

Palavras chave – integração numérica; processamento paralelo; memória compartilhada; OpenMP

I. Introdução

O OpenMP é uma maneira fácil de converter um programa que usa uma CPU em um programa capaz de usar várias CPUs. O OpenMP foi projetado apenas para sistemas de memória compartilhada, o que significa que os aplicativos são executados em um computador e apenas neste. Se planeja-se escrever um aplicativo de alto desempenho que será executado em um cluster de muitos computadores, o MPI é uma escolha melhor. No entanto, se o sprograma for executado em um único computador, o OpenMP pode muito bem ser a melhor e mais fácil maneira de criar um programa multithread de alto desempenho. Os programas OpenMP são adequados para executar em qualquer Hardware, desde computadores de núcleo único a computadores de memória compartilhada de 24 núcleos de alto desempenho.

A. Objetivo

Utilizar a linguagem de programação C++ para implementar o método do trapézio para o cálculo de integrais definidas por meio de programação paralela baseada no OpenMP.

B. Motivação

O OpenMP (Open Multi-Processing) é uma das estruturas de programação mais comuns usadas para implementar o paralelismo de memória compartilhada [1], tópico relevante devido à necessidade de otimização de recursos computacionais. Simplifica as tarefas de distribuição de trabalho e coordenação em um programa C, C ++ ou Fortran multi-encadeado. Assim, avaliar seu funcionamento, compreender de forma sintetizada a implementação computacional de tal recurso, faz-se necessário.

C. Metodologia

Após pesquisa bibliográfica com base em livros e artigos abordando os conceitos aplicados ao relatório, aplicou-se o OpenMP para criar um programa para cálculo de integrais baseando-se no método de integração numérica conhecida como “Método do Trapézio”.

II. Revisão da Literatura

Praticamente todos os programas que exigem alto desempenho têm algum tipo de loop, principalmente do tipo FOR. Muitas vezes, uma iteração de um loop FOR não tem nada a ver com nenhuma das outras iterações. Portanto, deve ser fácil paralelizar o dito loop. O OpenMP é um padrão para a programação de sistemas de memória compartilhada. Ele usa funções especiais e diretivas de pré-processador chamadas pragmas; portanto, ao contrário de Pthreads e MPI, o OpenMP requer suporte ao compilador. Um dos recursos mais importantes do OpenMP é que ele foi projetado para que os desenvolvedores possam paralelizar incrementalmente os programas seriais existentes, em vez de precisar escrever programas paralelos do zero. Os programas OpenMP iniciam vários threads em vez de vários processos. Threads podem ter peso muito menor do que processos; eles podem compartilhar quase todos os recursos de um processo, exceto que cada encadeamento deve ter sua própria pilha e contador de programas.

O OpenMP, permite que o compilador e o sistema de tempo de execução determinem alguns detalhes do comportamento do encadeamento, para que seja mais simples codificar alguns comportamentos paralelos usando o OpenMP. O custo é que algumas interações de encadeamento de baixo nível podem ser mais difíceis de programar.

III. Implementação Proposta

Na implementação do código proposto (enviado como anexo) no presente relatório utilizou-se os seguintes recursos de Hardware:

- MacBook Pro (Retina, 15-inch, Early 2013);
- Processador: 2,4 GHz Intel Core i7;
- Memória: 8 GB 1600 MHz DDR3.

Com relação aos softwares utilizados, citam-se:

- Sistema operacional UBUNTU 18.04
- GCC 7.4

IV. Experimentos e Resultados

Para a validação do programa implementado, utilizou-se a função sqrt(1 – x^2) com os limites de integração inferior = 0 e superior igual a 1, variando-se o número de iterações conforme Tabela 1. Comparando os resultados simulados com executados sem paralelismo e com paralelismo com o OpenMP.

Tabela 1 – Resultados dos Testes

Tempo de Execução em (ms)		
Iteracoes	Sem OpenMP	Com OpenMP
1000	376,217	1,215
10000	424,307	1,517
100000	1.069,048	1,538
10000000	7.507,734	1,587

A. Resultados

Pode-se notar comparando os resultados, que com o paralelismo o tempo de execução do programa decresce de forma significativa, provando a utilidade do OpenMP como ferramenta de programação paralela.

V. Considerações Finais

O principal objetivo deste relatório foi examinar o possível ganho da utilização do paralelismo aninhado quando disponível no problema. O uso dos dois níveis de paralelismo acabou sendo imprescindível para um bom dimensionamento nos testes. Também foi mostrado como o paralelismo de dois níveis pode ser implementado no OpenMP, usando programação explícita de threads.

Referências

[1] Blikberg, R.; Sorevik, T. “Nested Parallelism: Allocation of Threads to Tasks and OpenMP Implementation”. Scientific Programming. 9. 185-194, 2001.

[2] Sen, A. “Aprendendo a usar a estrutura OpenMP com GCC”, disponível em:https://www.ibm.com/developerworks/br/aix/library/au-aix-openmp-framework/index.html .

[3] J. Li, J. Shu, Y. Chen, D. Wang and W. Zheng, "Analysis of factors affecting execution performance of openMP programs," in Tsinghua Science and Technology, vol. 10, no. 3, pp. 304-308, June 2005.