1º Trabalho Prático de Programação Concorrente 2019

Desenvolva uma aplicação concorrente com espaço de endereçamento compartilhado, em C/OpenMP no Linux e que resolva um sistema linear (***Ax=b***), segundo o Método Iterativo de Jacobi-Richardson (também conhecido como Gauss-Jacobi).

Para execução, a aplicação deve ser iniciada da seguinte forma:

jacobipar <***N***> <***T***>

onde, ***jacobipar*** é o nome do arquivo binário para execução, <***N***> determina a ordem da matriz quadrada usada para armazenar as ***N*** variáveis das ***N*** equações existentes no sistema linear, e <***T***> determina a quantidade de threads que serão usadas ao todo na aplicação concorrente.

Os valores usados no sistema linear, representados pela matriz ***A*** e vetor ***b*** serão determinados de maneira pseudoaleatória. Fixe a semente para a geração dos números aleatórios, assim uma nova execução fornecerá o mesmo resultado. Ao montar a matriz ***A***, garanta a convergência do método.

O resultado da aplicação será determinado pelos valores do vetor ***x*** que satisfaçam as ***N*** equações simultaneamente. Para finalizar, a aplicação deverá perguntar ao usuário qual equação do sistema o mesmo deseja escolher para associar os resultados obtidos com o vetor ***x*** e, assim, comparar o resultado obtido com o valor correspondente no vetor ***b***.

Diferentes livros de Cálculo Numérico descrevem a solução do Método Iterativo de Jacobi-Richardson (ou Gauss-Jacobi). Em [1] há uma boa explicação do mesmo.

Desenvolva a aplicação concorrente em C/OpenMP segundo as diretrizes passadas em sala de aula para o desenvolvimento de aplicações concorrentes. Considere um projeto que maximize o ganho de desempenho em relação ao algoritmo sequencial, dados os valores de ***N*** e ***T***. O objetivo aqui é determinar e explorar a concorrência da aplicação de maneira flexível.

Execute as versões paralela e sequencial pelo menos 30 vezes cada uma, considerando três quantidades diferentes de número de threads na sua máquina. Monte uma tabela com esses resultados, contendo o número de threads testadas, médias das execuções, desvios padrão, speedups e eficiências.

Submeta no Moodle (e-disciplinas) um arquivo compactado (padrão zip) contendo: o código fonte sequencial (deve se chamar ***jacobiseq.c***), código fonte em C/OpenMP (deve se chamar ***jacobipar.c***), a descrição do projeto do algoritmo paralelo (seguindo PCAM e o arquivo deve se chamar ***pcam.pdf***) e os resultados das execuções (deve-se chamar ***resultados.pdf***) contendo a tabela dos resultados, gráficos e a explicação dos resultados obtidos. Se possível, inclua também o *makefile* que realize a compilação e execução dos códigos sequencial e paralelo.

O trabalho deverá ser entregue até as 23:55h do dia 16/10 (quarta-feira) no Moodle USP (e-disciplinas).

Questões omissas e/ou ambíguas serão fixadas pelo professor. Para saná-las, entre em contato com o professor para a orientação adequada.

Referências

[1] Ruggiero, Márcia A.G.; Lopes, Vera L.R. Cálculo Numérico: Aspectos Teóricos e Computacionais. McGraw-Hill, 1988.