CEFET-MG - Campus II
Departamento de Engenharia Elétrica
Computação de Alto Desempenho

Lista II

Alunos: Antônio Augusto Diniz Sousa
Professor: Eduardo Henrique da Rocha Coppoli

1 de Julho
2019

CEFET-MG - Campus II
Departamento de Engenharia Elétrica
Computação de Alto Desempenho

Lista II

Lista II apresentado à Disciplina de Computação de Alto Desempenho do Curso de Engenharia Elétrica do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para conclusão das matérias eletivas do curso de Engenharia de Computação da mesma instituição.

Alunos: Antônio Augusto Diniz Sousa

Professor: Eduardo Henrique da Rocha Coppoli

1 de Julho
2019

Sumário

Introdução

O objetivo deste documento é listar os enunciados e organizar os arquivos produzidos durante a execução da Lista II da disciplina de Computação de Alto Desempenho.

Todos os arquivos necessários para execução dos algoritmos e acompanhamento da execução do trabalho está disponível no github.

Dentro da pasta algoritmos do repositório citado, possui uma pasta para cada exercício desta lista, que conterá os códigos implementados e as observações necessárias.

1 Questão 1

1.1 Enunciado

Realize o cálculo do número PI, conforme expresso pela função (1). Utilize por exemplo integração trapezoidal para resolver o problema. Após esta etapa paralelize seu código utilizando OpenMP. Utilize uma referência como resultado para comparar sua solução.

$$\int \frac{4}{1+x^2} dx \tag{1}$$

1.2 Execução

O código implementado baseou-se em uma rotina de repetição, para simular a integração, em um intervalo finito, delimitado pelo *for*.

1.3 Resultado

Para uma iteração de 100000 vezes, o resultado obtido foi de 3.1415826536 que, quando comparado com o valor disponibilizado pela calculadora do *Google*, 3.14159265359, possui uma diferenciação apenas na 5 casa decimal, o que valida o resultado do algoritmo implementado.

2 Questão 2

2.1 Enunciado

Implemente computacionalmente o Método dos Gradientes Conjugados (GC) e resolva um sistema linear exemplo. Paralelize o solver implementado utilizando OpenMP. Verifique se houve redução no tempo de processamento e faça uma análise dos resultados.

2.2 Execução

Como o método utilizado possui uma certa complexidade, e a sua implementação não era o foco da lista, baseou-se em um código disponível de maneira aberta no repositório do github.

Na implementação utilizada, um laço de repetição foi utilizado na implementação. Fez-se então uma pequena alteração, para aplicar-se o paralelismo em um *for*.

2.3 Resultado

Por questões de tempo na implementação, utilizou-se o método manual para inserção do sistema linear, o que impossibilitou testes muito grandes. Devido a isso, utilizou-se a biblioteca time.h, que está no repositório citado, para melhorar a precisão na medição do tempo e conseguirmos um *speedup* de 1.5487, ou seja, um ganho aproximado de 50% ao rodarmos um sistema linear de grau 10 de maneira paralela nos dois núcleos disponíveis no computador utilizado.

O resultado possui uma precisão de 0.1, porém é uns dos parâmetros passados na chamada da *main*, e caso precise/deseja-se uma precisão maior, basta alterar na execução do algoritmo.

3 Questão 3

3.1 Enunciado

Escreva um algoritmo que permita a partição de um sistema linear de modo a ser processado em 4 máquinas interligadas em rede e operando em cluster com MPI. Apresente o algoritmo que realiza este procedimento.

3.2 Execução

Neste exercício, utilizou-se um algoritmo muito semelhante ao utilizado no tópico anterior, alterando apenas a maneira de otimização.

A mesma alterações feitas, foram mantidas, atentando em uma melhor divisão entre as rotinas do *for* que seriam passadas para cada máquina.

3.3 Resultado

Devido a dificuldades em fazer o MPI funcionar, não foi possível testar na prática o algoritmo implementado. As tentativas foram feitas no Cluster do DECOM, no CEFET-MG, porém, de acordo com as pesquisas feitas diante dos erros que estavam ocorrendo, o problema estava na versão do MPI que estava instalada, e como não é possível instalar versões diferentes, infelizmente esse teste ficará em pendência até uma próxima oportunidade.

4 Questão 4

4.1 Enunciado

Paralelize, utilizando OpenMP o Método de Gauss-Seidel. Explique no código o que está sendo realizado e resolva um sistema linear exemplo utilizando seu programa.

5 Questão 5

5.1 Enunciado

Explique os três níveis de Blas passados pelo professor, indicando exemplos.

5.2 Explicação

O Blas tem como objetivo otimizar os algoritmos, para torná-los mais eficientes e serem executados ainda mais rápido.

As otimizações propostas são feitas em tipos de operações pré definidas que são executadas em um domínio específico, no qual separa-se em três níveis conforme as especificaçôes deste domínio e de seu contra-domínio, conforme abaixo:

5.2.1 Nível 1

No nível 1 otimiza-se as operações que envolvem vetores unidimensionais na qual o resultado da operação também será um vetor unidimensional.

Um exemplo que pode-se utilizar desta otimização é a soma das magnitudes de todos os elementos de um vetor.

5.2.2 Nível 2

Já no nível 2, visa-se a otimização das operações que envolvem no domínio vetores bidimensionais, porém com o contra-domínio sendo um vetor unidimensional.

Um exemplo de operação que se enquadra neste tipo de otimização, seria a soma dos elementos da coluna de uma matriz, por exemplo.

5.2.3 Nível 3

Enfim, no nível 3, otimiza-se os cálculos que possuem domínio e contra-domínio sendo vetores bidimensionais, ou seja, matrizes.

Exemplos para essa operações tem aos montes, como, por exemplo, a soma de duas matrizes.