APROP - Entrega 1

trabalho realizado por

Miguel Oliveira nº1210445

Jorge Silva nº1210443

docente

Prof. Tiago Carvalho

19 de novembro de 2022

1 Introdução

Este trabalho foi desenvolvido numa máquina virtual usando o VirtualBox, que é um software de virtualização, onde foi criada uma máquina virtual com um sistema operativo Ubuntu server de 64-bit. A nível de especificações, a mesma tem um processador AMD Ryzen 5 5600X, com 4 cores e 4Gb de RAM.

2 Ex7 - Blocked Matrix Processing

2.1 Alínea a

De modo a comparar os tempos de execução das duas diferentes abordagens, foram executadas 50 vezes ambos os programas de modo a analisar os valores obtidos. As execuções tiveram os seguintes outputs:

GCC

2.1.1 Versão em Célula - GCC

- Nº de vezes executado = 50
- Tempo máximo = 0.516423
- Tempo mínimo = 0.443153
- Média dos tempos = 0.474433
- Desvio Padrao = 0.019862
- Tempo de execução sequencial = 0.008172

2.1.2 Versão em Bloco - GCC

- Nº de vezes executado = 50
- Tempo máximo = 0.003462
- Tempo mínimo = 0.002283
- Média dos tempos = 0.002382
- Desvio Padrao = 0.000175
- Tempo de execução sequencial = 0.008205

Analisando os valores, concluímos que a versão em bloco é mais rápida do que a versão em célula e até mesmo do que a versão sequencial. A versão em célula acaba por ser mais lenta do que a versão sequencial, pois a atribuição de uma tarefa por cada célula acaba por causar um overhead, o que faz com que o seu tempo de execução seja maior. Por outro lado, a versão em bloco atribui tarefas por vários blocos, ou seja, por conjuntos de células, portanto acaba por ser mais eficiente.

Foram também executadas 50 vezes as versões com o LLVM:

LLVM

2.1.3 Versão em Célula - LLVM

- Nº de vezes executado = 50
- Tempo máximo = 1.516308
- Tempo mínimo = 1.306070
- Média dos tempos = 1.481892
- Desvio Padrao = 0.029625
- Tempo de execução sequencial = 0.007958

2.1.4 Versão em Bloco - LLVM

- Nº de vezes executado = 50
- Tempo máximo = 0.004935
- Tempo mínimo = 0.002289
- Média dos tempos = 0.002482
- Desvio Padrao = 0.000407
- Tempo de execução sequencial = 0.007966

Analisando os valores obtidos, a versão com o LLVM é mais lenta do que a versão com o GCC no que toca às versões paralelas, em relação à versão sequencial acaba por ser um pouco mais rápida. Os tempos de execução das versões GCC e LLVM da versão em bloco acabam por ser semelhantes, mas as versões em célula possuem uma diferença de 1 segundo no tempo de execução, o que é bastante.

2.2 Alínea b

Comparando as estratégias de execução das tarefas, a estratégia mais eficiente é a versão por blocos, porque tal como já foi referido anteriormente, esta versão divide melhor o trabalho pelas várias tarefas,

evitando assim um overhead que podería causar um maior tempo de execução, pois no caso da versão por célula, todo o processo de criação das tarefas demora demasiado tempo devido ao facto de serem criadas tarefas para cada célula.

3 Ex7 - Cholesky decomposition

3.1 Alínea a

Tal como no exercício anterior, foi também executado 50 vezes o programa de modo a analisar os valores. A matriz usada teve um tamanho de 1000x1000, o tamanho do bloco foi 10x10 e o número de threads usadas foi 4. O output foi o seguinte:

GCC

3.1.1 Versão Sequencial - GCC

- Tempo Máximo = 0.091352
- Tempo Mínimo = 0.069226
- Média dos tempos = 0.070744
- Desvio Padrão dos tempos = 0.003115
- Média gflops = 4.719171

3.1.2 Versão com Worksharing Constructs - GCC

- Tempo Máximo = 0.070369
- Tempo Mínimo = 0.025443
- Média dos tempos = 0.043802
- Desvio Padrão dos tempos = 0.009796
- Média gflops = 8.014524

3.1.3 Versão com Tasks sem dependências - GCC

- Tempo Máximo = 0.128767
- Tempo Mínimo = 0.062692
- Média dos tempos = 0.084300
- Desvio Padrão dos tempos = 0.013962
- Média gflops = 4.055223

3.1.4 Versão com Tasks com dependências - GCC

- Tempo Máximo = 0.151233
- Tempo Mínimo = 0.092219
- Média dos tempos = 0.106999
- Desvio Padrão dos tempos = 0.013139
- Média gflops = 3.155778

LLVM

3.1.5 Versão Sequencial - LLVM

- Tempo Máximo = 0.087971
- Tempo Mínimo = 0.065450
- Média dos tempos = 0.070189
- Desvio Padrão dos tempos = 0.005682
- Média gflops = 4.777509

3.1.6 Versão com Worksharing Constructs - LLVM

- Tempo Máximo = 0.080500
- Tempo Mínimo = 0.028880
- Média dos tempos = 0.051261
- Desvio Padrão dos tempos = 0.010542
- Média gflops = 6.765160

3.1.7 Versão com Tasks sem dependências - LLVM

- Tempo Máximo = 0.055785
- Tempo Mínimo = 0.038295
- Média dos tempos = 0.042270
- Desvio Padrão dos tempos = 0.003150
- Média gflops = 7.926155

3.1.8 Versão com Tasks com dependências - LLVM

- Tempo Máximo = 0.220652
- Tempo Mínimo = 0.117083
- Média dos tempos = 0.137503
- Desvio Padrão dos tempos = 0.020795
- Média gflops = 2.463127

Analisando todos os valores obtidos, concluímos que a versão com worksharing constructs usando o GCC é a que possui melhor performance, seguida da versão sequencial, depois a versão de tasks sem dependências, e por último a versão com tasks com dependências. No caso do LLVM, a versão com melhor performance é a versão com tasks sem dependências, seguida da versão com worksharing constructs, depois a versão sequencial, e por último a versão de tasks com dependências. Comparando o GCC com o LLVM, concluímos que o GCC é melhor a gerir as worksharing constructs, mas o LLVM é melhor a gerir as tasks sem dependências.

3.2 Alínea b

Comparando as estratégias de implementação, como já foi visto anteriormente, a que possui melhor performance é a versão com worksharing constructs no caso do GCC, pois esta implementação paraleliza os ciclos "for", o que faz com que os cálculos sejam mais rápidos. Em relação às versões com tasks, a versão com dependências acaba por ser mais lenta, pois são criadas várias tasks, e o trabalho de gerir as dependências destas mesmas tasks acaba por causar um overhead, o que faz com que seja mais lento.