UNIVERSIDADE FEDERAL DE SÃO PAULO
CAMPUS SÃO JOSÉ DOS CAMPOS
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA (DCT)

Lista 1 - Programação Concorrente e Distribuída

UC: **Programação Concorrente e Distribuída**

Aluna: **Thauany Moedano**

RA: **92486**

Entrega: **29/08/2016**

Resumo

Exercício 1. Faça um relatório medindo Speedup e Eficiência para todos os casos de teste da lista 1.

Vamos construir tabelas e analisar o Speedup e eficiência para cada um dos casos de testes.

Programa A - Média dos elementos de um vetor

pThreads

Tabela 1: Speedup

N/Número de Threads	1	2	4	8
10^5	1	1.3	2	2
10^6	1	1.6	2.2	2.2
10^7	1	1.5	2.16	2.10

Tabela 2: Eficiência

N/Número de Threads	1	2	4	8
10^5	1	0.65	0.5	0.25
10^6	1	0.8	0.55	0.275
10^7	1	0.75	0.54	0.26

OpenMP

Tabela 3: Speedup

N/Número de Threads	1	2	4	8
10^5	1	1.1	1.1	1.1
10^6	1	1.08	1.18	1.24
10^7	1	1.05	1.05	0.9

Tabela 4: Eficiência

N/Número de Threads	1	2	4	8
10^5	1	0.55	0.275	0.137
10^6	1	0.54	0.295	0.155
10^7	1	0.525	0.2625	0.1125

Java Threads

Tabela 5: Speedup

N/Número de Threads	1	2	4	8
10^5	1	1.5	1	0.8
10^6	1	0.7	0.35	0.23
10^7	1	0.4	0.2	0.2

Tabela 6: Eficiência

N/Número de Threads	1	2	4	8
10^5	1	0.75	0.25	0.1
10^6	1	0.35	0.08	0.02
10^7	1	0.2	0.05	0.025

Os melhores ganhos de speedup para este caso foi utilizando pThreads. Para todas as versões o tamanho do problema não influenciou muito na perda de eficiência tendo números bem próximos para todos os casos. Em alguns casos o speedup ficou abaixo de 1 mostrando que houve piora no aumento de número de threads.

Programa B - Soma de uma matriz

pThreads

Tabela 7: Speedup

N/Número de Threads	1	2	4	8
10^2	1	1	1	1
10^3	1	1.2	1.2	1.5

Tabela 8: Eficiência

N/Número de Threads	1	2	4	8
10^2	1	0.5	0.25	0.125
10^3	1	0.6	0.3	0.18

OpenMP

Tabela 9: Speedup

N/Número de Threads	1	2	4	8
10^2	1	1.25	1.25	1.25
10^3	1	1.2	1.5	1.5

Tabela 10: Eficiência

N/Número de Threads	1	2	4	8
10^2	1	0.625	0.3125	0.15625
10^3	1	0.6	0.375	0.1875

Java Threads

Tabela 11: Speedup

N/Número de Threads	1	2	4	8
10^2	1	1	2	2
10^3	1	1.2	1	1

Tabela 12: Eficiência

N/Número de Threads	1	2	4	8
10^2	1	0.5	0.5	0.25
10^3	1	0.6	0.25	0.125

Estes testes mostram bem que o aumento do númer de testes não significa em um aumento de desempenho proporcional. Como evidencia o caso com Java Threads, somente com 4 threads o speedup sobe de um para dois mas se mantém em dois mesmo com oito threads.

Exercício 2. Considerando apenas o problema b (soma de matrizes) da lista 1 com o código desenvolvido em OpenMP, execute o programa com uma única thread e determine a fração do tempo total gasto com o trecho sequencial do programa (que não pode ser executado concorrentemente) e do trecho concorrente, baseado na Lei de Amdahl. A partir destes dados medidos, determine a estimativa de Speedup através da Lei de Amdahl e compare com um speedup real medido para 2 e 4 threads para N=10^2 e 10^3. Obs:

Para medir o speedup real meça o tempo total de processamento e não apenas o trecho concorrente.

Para 10^2: O tempo total do programa foi de 3 ms sendo que a parte serial foi executada em 2.93 ms para uma thread. Ou seja, a fração serial corresponde a 97.6% enquanto a paralelizável corresponde a 2.4%. Executando a lei de Amdahl para duas e quatro threads, teríamos:

- duas threads: 1.012 (real: 1.25)
- quatro threads: 1.018 (real: 1.25)

Para 10^3: O tempo total do programa foi de 36 ms sendo que a parte serial foi executada em 24.99 para uma thread. Ou seha, a fração serial corresponde a 69.4% e a parte paralelizável corresponde a 29.6%. Executando a lei de Amdahl para duas e quatro threads, teríamos:

- duas threads: 1.18 (real: 1.2)
- quatro threads: 1.29 (real: 1)

Os tempos previstos na Lei de Amdahl não foram tão próximos quanto o tempo medido. Isso porque a lei de Amdahl não leva em considerações fatores externos ao programa como arquitetura do computador que pode prejudicar o desempenho do programa.