Universidade Federal do Rio de Janeiro

Universidade Federal
do Rio de Janeiro
Escola Politécnica

Computação de Alto Desempenho

COC472

Alunos	Luis Eduardo Pessoa
	Eduardo Guimarães Ribeiro
	José Roberto Cunha
Professor	Álvaro Coutinho
Monitor	Rômulo Montalvão
Horário	Sex - 10:00-12:00

Rio de Janeiro, 3 de Julho de 2019

Conteúdo

1 Objetivo

O objetivo do trabalho é verificar como a paralelização de códigos pode influenciar no desempenho. Para isso foi utilizado um código de multiplicação matriz vetor na linguagem C/C++. Conforme verificado anteriormente C/C++ pega os dados da matriz na orientação da linha, portanto o algoritmo mais eficiente (que o orientado a coluna, mais eficiente no Fortran, por exemplo) foi utilizado.

1.1 Github

Os arquivos dos resultados podem ser encontrados no seguinte repositório:

https://github.com/LEpessoa/CAD_2019.1.git

2 Método

Na Multiplicação da Matriz A[nxn] x o vetor B[n] foram utilizados 4 tamanhos de n diferentes: 10000, 20000, 30000, 40000. O computador tem 4 núcleos e 4 threads, assim foram utilizados para análise dos 4 tamanhos os 4 núcleos, ou seja, cada tamanho foi rodado em 1, 2, 3 e 4 threads. O tamanho máximo de 40000 foi escolhido porque é perto do limite teórico e prático da memória RAM do computador (7.8 GB). Além dos resultados de cada tamanho foi gerado um gráfico de speedup.

O sistema operacional utilizado foi o Ubuntu 16.04, a versão do compilador foi a 5.4 e os códigos foram compilados da seguinte forma: g++ -mcmodel=medium matVet.cpp -fopenmp. A flag -mcmodel=medium foi utilizada para poder comportar o tamanho da matriz.

https://stackoverflow.com/questions/6069695/relocation-truncated-to-fit-error-when-compiling-using-g

3 Análise de resultados

Verificamos que, quanto a complexidade do algoritmo (O(n2)), os tempos para todos os tamanhos de n e números de threads são condizentes com os valores esperados. Lembrando que a partir desse tamanho de n esse comportamento não seria mais observado pois nem tudo caberia na memória principal acarretando em um tempo muito maior que o esperado.

N / THREADS (tempos em s)	1	2	3	4
10000	0.247s	0.130s	0.098s	0.082s
20000	1.048s	0.543s	0.406s	0.335s
30000	2.337s	1.208s	0.913s	0.756s
40000	4.157s	2.145s	1.620s	1.340s

Figura 1: Performance

Em seguida vemos as curvas de speedup (LINEAR x REAL)

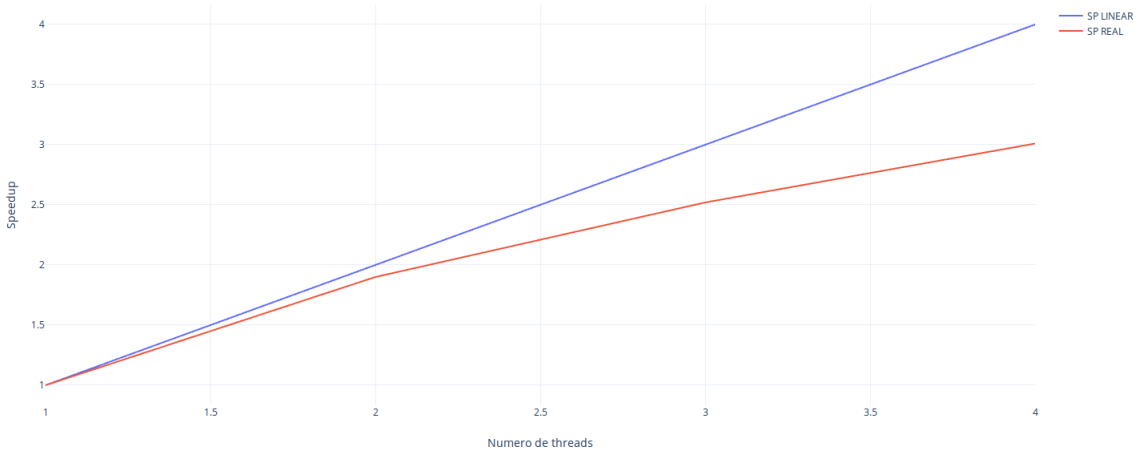

Figura 2: Gráfico speedup

De novo, como esperado, a melhora do tempo ocorre com o maior número de threads, mas essa melhora é menor que a melhora teórica (linear). Isso é o mais normal, pois nem tudo é totalmente paralelizável num código ou não

há um balanceamento correto entre as threads, mesmo assim consideramos que a melhora foi significativa.

Só um gráfico é mostrado, com o tamanho de n 10000, mas os outros 3 gráficos possíveis tem comportamento muito semelhante a esse, de forma que se conclui o mesmo para os 4 tamanhos de n.