MINIEP6

Patr[icia da Silva Rodrigues]

Maio 2023

1 Testes

Para n = 32
Executando dgemm_0... Tempo gasto em matrix_dgemm_0: 4.720804s Resultado OK! Tempo OK!
Executando dgemm_1... Tempo gasto em matrix_dgemm_1: 1.128779s Resultado OK! Tempo OK!
Executando dgemm_2... Tempo gasto em matrix_dgemm_2: 1.054240s Resultado OK! Tempo OK!
Para n = 64 Executando dgemm_0...
Tempo gasto em
matrix_dgemm_0:
4.352568s
Resultado OK!
Tempo OK!
Executando dgemm_1...
Tempo gasto em
matrix_dgemm_1:
1.056138s
Resultado OK!
Tempo OK!
Executando dgemm_2...
Tempo gasto em
matrix_dgemm_2: 0.939202s
Resultado OK!
Tempo OK!
Para n = 128
Executando dgemm_0...
Tempo gasto em matrix_dgemm_0: 4.599323s
Resultado OK!
Tempo OK!
Executando dgemm_1...
Tempo gasto em matrix_dgemm_1:
1.051289s

Resultado OK!
Tempo OK!
Executando dgemm_2... Tempo gasto em matrix_dgemm_2: 1.008627s
Resultado OK!
Tempo OK!
Para n = 256
Executando dgemm_0...
Tempo gasto em matrix_dgemm_0:
4.599323s
Resultado OK!
Tempo OK!
Executando dgemm_1... Tempo gasto em matrix_dgemm_1: 1.051289s
Resultado OK!
Tempo OK!
Executando dgemm_2... Tempo gasto em matrix_dgemm_2:
1.008627s
Resultado OK!
Tempo OK!
N = 512
Executando dgemm_0... Tempo gasto em matrix_dgemm_0: 4.916062s Resultado OK! Tempo OK!
Executando dgemm_1... Tempo gasto em matrix_dgemm_1: 1.122107s Resultado OK! Tempo OK!
Executando dgemm_2... Tempo gasto em matrix_dgemm_2: 1.050526s Resultado OK! Tempo OK!
n = 1024
Executando dgemm_0... Tempo gasto em matrix_dgemm_0: 5.239023s Resultado OK! Tempo OK!
Executando dgemm_1... Tempo gasto em matrix_dgemm_1: 1.427932s Resultado OK! Tempo OK!
Executando dgemm_2... Tempo gasto em matrix_dgemm_2: 1.226216s Resultado OK! Tempo OK!

Conclusao: A partir dos resultados apresentados nos testes com diferentes tamanhos de matrizes, é possível observar que a técnica de blocagem utilizada para a implementação das funções dgemm_1 e dgemm_2 obteve desempenho significativamente melhor do que a implementação sem blocagem (dgemm_0).

Isso se deve ao fato de que a técnica de blocagem permite que as operações sejam realizadas de forma mais eficiente, reduzindo o número de operações de multiplicação e soma necessárias para calcular cada elemento da matriz resultante. Além disso, a técnica de blocagem também aproveita melhor a memória cache do processador, o que pode contribuir para um desempenho melhor.

Também é possível observar que a implementação dgemm_2 obteve um desempenho ligeiramente melhor do que a implementação dgemm_1, o que pode ser explicado pelo fato de que a implementação dgemm_2 utiliza blocos de tamanho maior do que a implementação dgemm_1, o que pode reduzir ainda mais o número de operações necessárias.

Executando na maquina de um amigo, obtiive:
Para n = 32:
Executando dgemm_0... Tempo gasto em matrix_dgemm_0: 4.759600s
Resultado OK!
Tempo OK!
Executando dgemm_1...
Tempo gasto em matrix_dgemm_1: 1.090879s
Resultado OK!
Tempo OK!
Executando dgemm_2...
Tempo gasto em matrix_dgemm_2: 0.943355s
Resultado OK!
Tempo OK!
Para n = 64:
Executando dgemm_0...
Tempo gasto em matrix_dgemm_0: 4.532296s
Resultado OK!
Tempo OK!
Executando dgemm_1...
Tempo gasto em matrix_dgemm_1: 1.068714s
Resultado OK!
Tempo OK!
Executando dgemm_2...
Tempo gasto em matrix_dgemm_2: 0.942884s
Resultado OK!
Tempo OK!
Para n = 128
Executando dgemm_0...
Tempo gasto em matrix_dgemm_0: 4.868790s
Resultado OK!
Tempo OK!
Executando dgemm_1...
Tempo gasto em matrix_dgemm_1: 1.073859s
Resultado OK!
Tempo OK!
Executando dgemm_2...
Tempo gasto em matrix_dgemm_2: 1.004717s
Resultado OK!
Tempo OK!

Sabemos que o tamanho das linhas de cache pode influenciar na escolha do tamanho do bloco a ser usado na técnica de blocagem, já que um tamanho de bloco que é bom para uma máquina pode não ser o melhor para outra máquina. Além disso, outras especificações do sistema. No entanto, n'ao verifiquei nenhuma mudanca significativa de desempenho entre os computadores.