Programação de Alto Desempenho

Atividade 2 - Otimizando o desempenho de códigos para afinidade de memória

Lucas Santana Lellis - 69618
PPGCC - Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo

I. Introdução

Nesta atividade foram realizados experimentos relacionados com a otimização do desempenho de algoritmos quanto à afinidade de memória.
Cada experimento foi realizado 5 vezes, e os resultados apresentados são a média dos resultados obtidos em cada um deles.

Todos os programas foram feitos em C, utilizando a biblioteca PAPI para estimar o tempo total de processamento, quantidade de chache misses em memória cache L2, e o total de operações de ponto flutuante.

As especificações da máquina utilizada estão disponíveis na Tabela I.

Tabela I: Especificações da Máquina

CPU	Intel Core i5 - 3470
Cores	4
Threads	4
Clock	3.2 GHz
Cache L3	6144 KB
Cache L2	256 KB * 4
Hardware Counters	11
RAM	8 Gb
SO	Fedora 24
Kernel	4.7.2-201.fc24.x86_64
GCC	6.1.1 20160621

II. Experimento 1 - Multiplicação Trivial

Nesse experimento foi implementado o algoritmo tradicional para multiplicação de matrizes, sem blocagem, para verificar a diferença no desempenho causada pela mudança da hierarquia dos laços: IJK, IKJ, JIK, JKI, KIJ e KJI.

Na Tabela II temos um resumo dos experimentos realizados, onde percebemos a clara vantagem que as hierarquias IKJ e KJI têm sobre as demais. Desses resultados podemos perceber que o modelo KJI é mais eficiente para matrizes de 128x128, enquanto o modelo IKJ se sobressai nas demais.
Além disso, percebemos que os experimentos com menores tempos de execução também são os com o menor número de cache misses.

III. Experimento 2 - Multiplicação com Blocagem

Nesse experimento foi implementado o algoritmo para multiplicação de matrizes com blocagem, para verificar a diferença no desempenho causada pela mudança do tamanho do bloco para 2, 4, 16 e 64. A implementação foi feita com base na hierarquia de laços IKJ, que obteve os melhores resutlados em matrizes maiores.

Tabela II: Multiplicação de matrizes trivial comutando hierarquia de laços.

Size	Mode	Tempo(μs)	L2_DCM	MFLOPS	CPI
128	IJK	2328.4	23315.6	2505.984	0.45
128	IKJ	1148.0	60899.4	4924.554	0.43
128	JIK	2275.4	73107.4	2416.602	0.46
128	JKI	2888.4	68470.0	2009.924	0.47
128	KIJ	1140.4	43748.0	4227.710	0.43
128	KJI	2897.0	50055.8	1951.854	0.47
512	IJK	1013288.0	17067141.6	1937.120	3.30
512	IKJ	66925.8	2476257.6	5170.584	0.41
512	JIK	1020823.2	2600910.6	1659.448	3.22
512	JKI	1960534.6	19045678.8	2126.452	4.99
512	KIJ	75300.2	7942061.2	4770.238	0.46
512	KJI	1950122.6	18215106.2	2124.414	4.97
1024	IJK	9527464.4	91163781.6	1620.666	3.80
1024	IKJ	627804.0	23053915.6	4077.232	0.48
1024	JIK	9323631.2	47415128.6	1401.120	3.73
1024	JKI	18096589.8	295045291.8	2111.022	5.85
1024	KIJ	844605.2	79738619.8	3346.942	0.67
1024	KJI	18088030.8	377899075.8	2018.388	5.83

Na tabela III também percebe-se uma forte relação entre os melhores tempos e o número de cache misses. Neste caso, o bloco de tamanho 16x16 se mostrou mais eficiente para matrizes de tamanho 128x128, enquanto blocos de tamanho 64 foram mais eficientes para as demais.

Tabela III: Multiplicação de matrizes IKJ com blocagem, variando tamanho dos blocos.

Size	Block	Tempo(μs)	L2_DCM	MFLOPS	CPI
128	2	1214.2	21063.6	3588.478	0.41
128	4	2271.6	11271.0	1917.092	0.54
128	16	1057.0	12158.4	4214.226	0.37
128	64	1070.4	5600.6	4900.386	0.37
512	2	74317.8	697431.0	3785.532	0.40
512	4	142427.8	356271.0	1969.946	0.53
512	16	63909.2	459286.0	4457.264	0.37
512	64	60182.0	533064.6	5920.236	0.34
1024	2	594527.6	3915184.8	3770.678	0.41
1024	4	1140751.8	1529148.6	1968.784	0.54
1024	16	514722.8	2758772.8	4387.404	0.37
1024	64	478965.6	2779402.4	5468.596	0.33

IV. Experimento 3 - Multiplicação de Strassen

Nesse experimento foi implementado o algoritmo de Strassen, de forma que a matriz é particionada e realocada em matrizes menores utilizando uma técnica de divisão e conquista. Nessa implementação específica o algoritmo possui uma otimização, pois quando se obtém uma matriz suficientemente pequena, é realizada uma multiplicação trivial IKJ.

Neste experimento, variamos o tamanho da matriz do caso base, que corresponde à segunda coluna da Tabela IV. Percebemos então um empate técnico entre a matriz de tamanho 32x32 e a de tamanho 64x64. Dessa vez, não necessariamente os

melhores resultados foram os com o menor número de cache misses, uma vez que cada chamada recursiva requer alocação dinâmica de matrizes, que pode provocar tais variações nos resultados.

Tabela IV: Multiplicação de Matrizes de Strassen, variando tamanho da matriz do caso base.

Size	Block	Tempo(μs)	L2_DCM	MFLOPS	CPI
128	16	3019.8	50676.0	1148.756	0.41
128	32	1330.6	33608.8	2768.332	0.40
128	64	1443.8	21108.8	3020.316	0.47
512	16	154615.6	3778285.0	1136.444	0.43
512	32	72525.6	2768841.4	2565.314	0.43
512	64	72308.2	2078839.2	2997.040	0.48
1024	16	1089220.4	27908371.8	1137.370	0.43
1024	32	537118.6	20777161.6	2444.004	0.45
1024	64	533280.4	15909049.0	2864.332	0.49

V. Experimento 4 - BLAS

Nesse experimento foi utilizada a função cblas_dgemm do BLAS para realizar a multiplicação de duas matrizes, e obtemos assim os resultados da Tabela V.

Tabela V: Multiplicação de matrizes da biblioteca BLAS

Size	Tempo(μs)	L2_DCM	MFLOPS	CPI
128	66215.4	6893.8	4962.292	0.43
512	25526.6	329320.6	10435.756	0.37
1024	181173.4	691752.8	11727.154	0.35

Comparando os melhores resultados dos 4 experimentos (considerando os testes com matrizes de tamanho 1024x1024), fazemos então a comparação da Tabela VI, onde percebemos a clara superioridade do BLAS/ATLAS sobre todas as outras tentativas. Tal fato fica ainda mais evidente na Figura 1.

Tabela VI: Comparação dos melhores resultados em multiplicação de matrizes de tamanho 1024x1024

Algorithm	Tempo(μs)	L2_DCM	MFLOPS	CPI
Trivial	593359.8	19241602.8	0.010	0.47
Blocking	465177.8	2731991.4	0.000	0.33
Strassen	495581.0	20617179.2	0.002	0.44
BLAS	185065.6	730013.4	281.374	0.35

VI. Experimento 5 - Fusão de Laços

Nesse experimento foi feita a comparação do desempenho da técnica de fusão de laços. Na Tabela VII, a peimeira linha representa o algoritmo sem fusão de laços, e a segunda, o algoritmo com fusão de laços. Nesse experimento, percebemos uma leve melhora no desempenho do algoritmo.

Tabela VII: Técnica de fusão de laços.

Size	Mode	Tempo(μs)	L2_DCM	MFLOPS	CPI
1000000	Sem Fusao	136321.6	17590.0	501.082	2.96
1000000	Com fusao	133880.8	11508.0	432.718	2.96

VII. Experimento 6 - Estruturas de dados

Nesse experimento foi feita a comparação do desempenho da técnica de fusão de laços trabalhando com diferentes estruturas de dados. Na tabela VIII, a primeira linha representa o

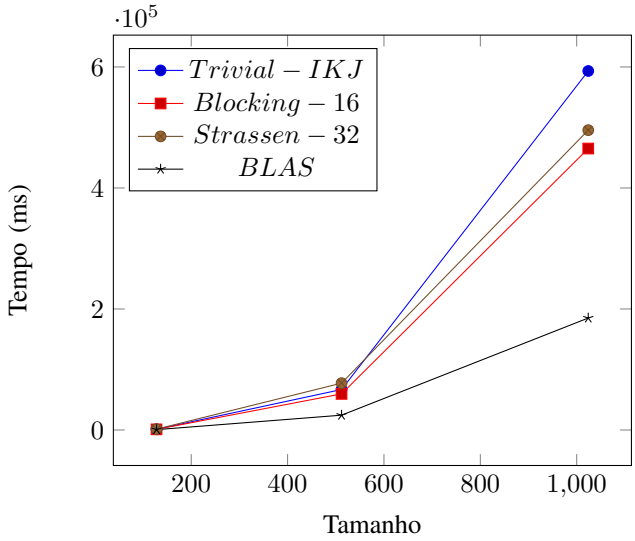

Figura 1: Comparação entre os melhores resultados dos quatro experimentos com relação ao tempo.

formato **double abc[?][3]**, a segunda linha o formato **double abc[3][?]**, e a terceira um array do tipo de dados **struct {double a, b, c; } est_abc** . Nesse experimento percebemos que embora possua o menor número de cache misses, o terceiro modo não é o mais rápido, perdendo por pouco do primeiro modo. Muito embora ambos sejam virtualmente equivalentes, pois em ambos os casos, os valores a, b e c estão contíguos na memória.

Tabela VIII: Desempenho obtido no exp 5

Size	Mode	Tempo(μs)	L2_DCM	MFLOPS	CPI
1000000	abc[?][3]	134765.2	33154.4	443.484	2.96
1000000	abc[3][?]	139954.6	11377.2	441.758	3.09
1000000	struct	137169.8	2774.8	444.124	2.96

VIII. Conclusão

Os experimentos realizados demonstraram a grande vantagem computacional obtida por meio da utilização de técnicas que favorecem a afinidade de memória, adequando-se a hierarquia dos laços, e também as estruturas de dados de forma que se faça o uso mais eficiente do processador, evitando que este permaneça ocioso.

Porém, isso não é o suficiente para se obter um desempenho ótimo do algoritmo, uma vez que os resultados variam de acordo com a complexidade dos problemas, e uma vez que desconhecemos todos os recursos adicionais presentes na arquitetura do processador, que o BLAS certamente faz uso.

Isso também evidencia a superioridade e a importância da utilização de bibliotecas otimizadas de operações de álgebra linear