Relatório de Desempenho de Programas

Filipe dos Santos Schuler e Rafael Benjamin Bombach

Introdução

Este relatório visa fornecer uma análise aprofundada do desempenho de três programas de multiplicação de matrizes, sendo eles P1(com matrizes estáticas), P2(com matrizes alocadas dinamicamente) e P3(com a matriz sendo representada por um vetor gigante). A análise é baseada em duas tabelas que fornecem informações adquiridas pela ferramenta "perf" do linux sobre ciclos, instruções, acessos à memória, misses e hits em TLB (Tabela 1) e misses e hits em caches L1, L2 e L3 (Tabela 2).

Especificações da máquina

Para a realização do trabalho foi utilizada o seguinte hardware:

- Modelo hardware: Hewlett-Packard HP Pavilion 14 Notebook PC
- Memória RAM: 8,0 GiB
- Processador: Intel® Core™ i7-4500U CPU @ 1.80GHz × 4
- Gráficos: HAINAN (, LLVM 15.0.7, DRM 2.50, 6.2.0-39-generic) / Mesa Intel® HD Graphics 4400 (HSW GT2)
- Capacidade Disco: 1,0 TB
- Nome SO: Ubuntu 22.04.3 LTS
- Tipo SO: 64 bits
- GNOME versão: 42.9

Lista de eventos utilizados na análise de desempenho

- cycles
- instructions

- L1-dcache-load-misses
- L1-dcache-loads
- l2_rqsts.code_rd_miss
- l2_rqsts.code_rd_hit
- LLC-load-misses
- LLC-loads
- dTLB-load-misses
- dTLB-loads
- mem-loads

Tabela 1 - Ciclos, Instruções, Misses e Hits em TLB, IPC e Miss Rate em TLB.

Programa	Ciclos	Instruções	IPC	TLB Misses	TLB Hits	Miss Rate em TLB
P1 32	2.859.507	2.248.718	0,786	1.355	757.521	0,178
P1 64	6.545.686	8.861.510	1,355	2.001	3.485.817	0,057
P1 128	34.234.668	54.290.937	1,587	3.952	27.365.940	0,014
P1 256	301.814.739	442.764.974	1,467	16.553	195.024.357	0,008
P1 512	2.819.743.322	3.518.870.087	1,247	1.255.498	1.434.981.418	0,087
P2 32	1.943.135	2.987.551	1,536	1.123	1.027.799	0,109
P2 64	6.393.029	14.891.843	2,329	1.254	6.048.934	0,020
P2 128	40.386.364	113.679.146	2,819	2.560	51.016.652	0,005
P2 256	373.119.603	851.102.301	2,282	18.493	363.590.570	0,005
P2 512	3.439.640.481	6.553.111.553	1,905	172.516	2.791.823.520	0,006
P2 1024	42.419.023.134	52.863.536.995	1,245	218.768.448	22.467.998.021	0,001
P2 2048	407.285.713.861	422.613.535.753	1,039	8.751.133.744	178.925.698.392	0,005
P2 4096	3.918.893.751.502	3.374.105.556.805	0,861	71.454.959.156	1.349.092.163.610	0,005
P2 8192	28.653.849.313.854	26.968.895.604.300	0,940	561.225.136.816	10.900.065.926.784	0,005
P3 32	2.548.499	2.772.111	1,086	1.232	886.948	0,138
P3 64	8.935.101	13.734.011	1,536	1.157	5.092.319	0,022
P3 128	35.891.030	96.361.804	2,685	2.346	44.260.663	0,005
P3 256	327.129.287	688.633.296	2,105	4.908	311.691.311	0,002
P3 512	3.084.262.413	6.214.005.767	2,014	71.520	2.327.496.732	0,003

P3 1024	68.358.665.457	48.516.955.915	0,709	125.163.538	18.335.470.700	0,007
P3 2048	672.956.894.610	387.903.134.390	0,576	8.663.376.789	145.723.908.744	0,006
P3 4096	5.764.743.789.686	3.101.256.683.218	0,537	69.679.644.295	1.168.660.960.814	0,006
P3 8192	51.548.320.955.881	24.876.463.052.944	0.48	555.282.734.122	8.829.064.220.410	0,006

Análise da Tabela 1

Com os valores de ciclos e instruções da Tabela 1 é possível calcular o IPC (Instruções Por Ciclo). O IPC mede quantas instruções são executadas por ciclo do processador. Valores mais altos indicam uma execução mais eficiente.

$$IPC = \frac{instruções}{ciclos}$$

Calculando o IPC de cada programa, chegamos à conclusão de que o P1 é menos eficiente se comparado aos outros programas considerando o mesmo tamanho, pois apresenta um IPC menor. A diferença entre P2 e P3 em questão de eficiência é pouco notória, mas P2 tem uma leve vantagem para um número maior de dados.

Já o TLB (Translation Lookaside Buffer) é uma estrutura de cache que acelera a tradução de endereços virtuais para endereços físicos, melhorando a eficiência do acesso à memória em sistemas que utilizam memória virtual. O aumento nos TLB misses significa uma maior complexidade do acesso à memória, e sugere que otimizações no algoritmo podem ser implementadas para melhorar a localidade da memória e reduzir o número de acessos à TLB.

É possível analisar o Miss Rate de TLB e perceber que o P1 possui uma complexidade maior de acesso a memória e por isso acaba sendo um programa menos otimizado. Por outro lado as aplicações P2 e P3 apresentam um Miss rate semelhante, mas com a P2 levando uma certa vantagem em relação a P3.

Tabela 2 - Misses e Hits em Caches e Acessos à Memória

Programa	L1 Misses	L1 Hits	L2 Misses	L2 Hits	L3 Misses	L3 Hits	Acessos à memória
P1 32	25.538	730.396	15.895	28.424	3.728	6.704	0
P1 64	53.458	3.474.766	15.570	41.808	4.581	12.244	0
P1 128	2.199.964	27.547.648	66.232	50.166	32.757	49.910	0
P1 256	17.321.565	197.403.979	183.182	49.486	278.780	12.120.462	0
P1 512	182.716.692	2.861.881.392	575.301	242.527	13.332.273	123.605.317	52.424
P2 32	15.812	1.039.076	19.100	18.742	4.049	10.164	0
P2 64	20.814	6.044.882	17.197	21.430	4.814	6.115	0
P2 128	196.378	49.610.669	70.524	24.774	10.463	100.047	0
P2 256	3.343.951	367.872.652	42.203	15.874	220.871	860.414	0
P2 512	182.716.692	2.861.881.392	215.989	123.987	5.925.814	7.752.679	0
P2 1024	1.339.359.557	22.469.697.600	3.012.281	1.324.041	73.291.323	63.226.255	249.539
P2 2048	12.464.078.514	179.226.711.227	37.878.350	13.080.691	658.857.934	5.982.284.996	2.291.697
P2 4096	116.915.472.738	1.429.327.897.595	411.687.947	135.555.422	5.919.490.572	102.574.596.606	7.569.680
P2 8192	1.208.374.590.368	11.537.256.800.996	1.434.844.160	317.072.480	42.475.752.155	831.829.799.914	11.189.634
P3 32	19.258	888.743	1.232	21.306	4.427	4.690	0
P3 64	12.242	4.954.433	1.157	5.092.319	5.198	7.044	0
P3 128	19.461	17.764.725	2.346	9.868	4.892	14.569	0
P3 256	56.633	321.352.515	4.908	23.156	56.633	9.887.933	0
P3 512	3.256.869	2.204.008.512	71.520	159.195	3.256.869	130.606.292	0
P3 1024	661.050.023	18.297.573.549	125.163.538	2.636.704	661.050.023	429.063.914	0
P3 2048	11.270.666.135	146.136.996.943	27.893.788	43.718.428	7.425.468.955	1.653.272.843	0
P3 4096	107.734.263.307	1.167.393.404.504	576.718.843	279.890.035	64.273.502.438	13.686.585.687	0
P3 8192	1.144.485.199.128	8.244.342.297.713	4.750.218.084	1.777.123.569	567.172.459.820	205.740.894.775	15.913.378

Análise da Tabela 2

À medida que o tamanho da matriz aumenta, há um aumento no número de misses em todas as camadas de cache para todos os programas. Isso era esperado, pois matrizes maiores podem não caber nas caches, resultando em mais misses, e em alguns casos, em buscas na memória caso os dados desejados não sejam encontrados nas caches.

A quantidade de misses está diretamente relacionada com a eficiência do programa, visto que quanto mais misses o programa possuir, pior é a sua distribuição de dados na memória. Assim, agrupando os programas de mesmo tamanho é possível observar que de maneira geral P2 e P3 são mais eficientes que P1. Além disso, para matrizes de grande tamanho, é possível observar que P2 não contém a matriz inteira nas caches e necessita fazer buscas pela memória, o que acaba sendo uma desvantagem, pois o acesso à memória é mais lento, o que impacta na eficiência do programa.

Resultados e Discussões

Para realizar o experimento, começamos desenvolvendo os programas com auxílio de funções “printf” até que tivéssemos certeza de que os programas executassem como desejado, então retiramos essas funções para que não interferissem nos resultados.

Como o esperado, o programa P1 (matrizes estáticas) não foi capaz de realizar todas execuções desejadas, apresentando falha de segmentação quando recebeu como entrada matrizes de tamanho 1024. Já os programas P2 (matrizes alocadas dinamicamente) e P3 (matrizes representadas por vetores gigantes) obtiveram sucesso nos testes realizados até entradas de 8192, mas para valores maiores o tempo de execução se tornou muito grande e a execução teve que ser cancelada após 10 horas.

Conclusão

Com base na análise detalhada, foi possível concluir que o tamanho do programa influencia diretamente o desempenho, com um aumento proporcional nas métricas de desempenho. Deste modo, aplicações maiores apresentam maiores números de misses em TLB e cache.

O P1, por ser alocado estaticamente, não suporta tamanhos muito grandes de dados e isso é evidenciado pela leitura de seu IPC que apresenta sua ineficiência e pelo seu alto número de Miss Rate em TLB em comparação com P2 e P3. Por outro lado, estes dois últimos programas, por serem alocados dinamicamente apresentam um comportamento mais estável e desse modo são mais recomendados para sua utilização. Ainda assim, estes programas ainda sofrem com o aumento do número de dados e a partir de certo número de dados seus desempenhos caem notavelmente.

Em comparação entre o P2 e P3 é possível perceber certa vantagem favorecendo o P2 devido seu modo de alocação de memória diferente do P3. Mesmo assim, estratégias de otimização específicas podem ser implementadas para melhorar o desempenho geral dessas aplicações, como ajustes na gestão de TLB e otimizações na hierarquia de caches.