



**UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

NICOLAS VON DOLINGER MOREIRA ROCHA

**ESTRUTURA DE DADOS
RELATÓRIO PRÁTICA 4**

BELO HORIZONTE

2023

Introdução

Este relatório descreve a caracterização da localidade de referência no código fornecido para análise. O objetivo é avaliar qualitativamente o comportamento do código em termos de acesso à memória e identificar as estruturas de dados e segmentos de código críticos que afetam o desempenho. O código fornecido é uma aplicação de exemplo que realiza operações matriciais, incluindo multiplicação, soma e transposição de matrizes. O código foi escolhido para análise devido à sua complexidade de acesso à memória, que oferece uma oportunidade valiosa para caracterizar a localidade de referência.

Análise Qualitativa

1 - Acessos de Memória Esperados:

O código realiza intensivos acessos à memória devido às operações em matrizes, especialmente durante a multiplicação de matrizes de grande porte. Além das operações de matrizes, o código também pode acessar a memória para outras operações, como alocação de recursos e manipulação de estruturas de dados complexas.

2 - Localidade de Referência:

O código provavelmente exibirá uma localidade de referência espacial, uma vez que as operações em matrizes geralmente envolvem acesso a elementos próximos na memória. A localidade de referência temporal pode ser observada em iterações sobre matrizes, onde os mesmos elementos de matriz são acessados repetidamente.

3 - Estruturas de Dados Críticas:

As estruturas de dados cruciais que têm o potencial de ter um impacto substancial no desempenho incluem as matrizes de entrada e saída, que representam os principais elementos de acesso à memória. No contexto deste código, a estrutura de dados crítica é denominada 'mat_tipo'. Essa estrutura contém um arranjo bidimensional de números em ponto flutuante e é considerada crítica, visto que

desempenha um papel central em todas as operações de acesso à memória realizadas ao longo da execução do código.

4 - Segmentos de Código Críticos:

Os loops aninhados que lidam com a inicialização aleatória da matriz e o acesso aos seus elementos desempenham um papel crucial na execução do programa. Eles são responsáveis por percorrer todos os elementos da matriz, o que pode resultar em uma demanda intensa de memória e tempo de execução. Essa parte do código é especialmente sensível ao tamanho da matriz, pois o número de iterações aumenta quadraticamente com o tamanho da matriz.

A função 'transpoeMatriz' é outro ponto crítico no código devido ao seu elevado número de operações de acesso à matriz e à complexidade de efetuar trocas de elementos para realizar a transposição. A transposição de uma matriz envolve a troca de elementos entre linhas e colunas, e isso requer um número significativo de operações. A função 'multiplicaMatrizes' apresenta uma estrutura complexa com três loops aninhados e inúmeras operações de acesso à matriz. A multiplicação de matrizes é uma operação computacionalmente intensiva, e o desempenho dessa função pode ser crítico, especialmente para matrizes grandes.

Plano de Caracterização de Localidade de Referência

O objetivo deste plano é avaliar a localidade de referência no código fornecido para identificar oportunidades de otimização em relação ao acesso à memória e ao desempenho geral do programa.

Etapa 1: Compreensão do Código e Identificação de Pontos Críticos

Nesta etapa, foi feita uma revisão do código-fonte para entender sua estrutura, operações principais e estruturas de dados relevantes. Foram identificados os pontos críticos do código que podem afetar o acesso à memória. É importante ter uma compreensão clara do programa e das áreas potencialmente problemáticas antes de iniciar a caracterização.

Etapa 2: Seleção de Cenários de Execução

Nesta etapa, foram escolhidos os cenários de execução representativos que envolvam operações críticas de acesso à memória. Isso pode incluir operações de multiplicação de matrizes, soma de matrizes ou outras operações significativas. A escolha de cenários de execução representa o comportamento real do programa e direciona a caracterização para áreas críticas de acesso à memória.

Etapa 3: Execução do Cachegrind para Caracterização

Usaremos a ferramenta Cachegrind (parte do Valgrind) para caracterizar a localidade de referência durante a execução do programa nos cenários selecionados. Executaremos o programa com os seguintes comandos:

```
valgrind --tool=cachegrind ./MeuPrograma <cenário_de_execução>
```

O Cachegrind fornece informações detalhadas sobre o comportamento do acesso à memória, incluindo métricas de cache e taxa de acertos, ajudando a identificar falhas de cache e áreas de otimização.

Etapa 4: Análise dos Resultados do Cachegrind

Após a execução do programa com o Cachegrind, analisaremos os resultados gerados pelo Cachegrind usando a ferramenta **cg_annotate**. Isso nos permitirá entender o comportamento de acesso à memória e identificar áreas críticas que requerem otimização. A análise dos resultados nos fornece insights sobre onde ocorrem os principais acessos à memória e onde estão os gargalos de desempenho.

O mesmo será feito com o Callgrind.

Parâmetros

Nesta seção do relatório, descreveremos os parâmetros selecionados para a caracterização do código, conforme definido no Makefile fornecido. As seguintes execuções foram realizadas:

1. Multiplicação de Matrizes (500x500):

Comando: **\$(EXE) -m -p /tmp/mult500.out -x 500 -y 500**

Descrição: Esta execução representa uma multiplicação de matrizes de dimensão 500x500. É uma operação computacionalmente intensiva que envolve o acesso a muitos elementos da matriz.

2. Soma de Matrizes (500x500):

Comando: **\$(EXE) -s -p /tmp/soma500.out -x 500 -y 500**

Descrição: Esta execução representa uma operação de soma de matrizes de dimensão 500x500. Também envolve acesso intensivo à memória para adição de elementos de matriz.

3. Transposição de Matrizes (500x500):

Comando: **\$(EXE) -t -p /tmp/transp500.out -x 500 -y 500**

Descrição: Nesta execução, a transposição de matrizes de dimensão 500x500 é realizada. A transposição envolve reorganização de dados na memória e é crítica para a localidade de referência.

4. Multiplicação de Matrizes (100x100) com Registro de Acesso:

Comando: **\$(EXE) -m -p /tmp/mult100log.out -l -x 100 -y 100**

Descrição: Esta execução é semelhante à primeira, mas também inclui registro de acesso (-l). Isso permitirá uma análise detalhada dos padrões de acesso à memória durante a multiplicação de matrizes menores de 100x100.

Justificativa para a Seleção dos Parâmetros:

Os parâmetros selecionados incluem operações críticas de acesso à memória, como multiplicação, soma e transposição de matrizes. Essas operações envolvem leitura e escrita intensivas na memória, tornando-as relevantes para a caracterização de localidade de referência. A escolha de matrizes de dimensão 500x500 e 100x100 permite avaliar o comportamento do programa em diferentes escalas, abrangendo operações de grande e pequeno porte.

A execução com registro de acesso (opção -l) na multiplicação de matrizes de 100x100 permitirá uma análise detalhada dos padrões de acesso à memória

durante essa operação específica. As operações selecionadas refletem situações comuns em programas que lidam com manipulação de matrizes, e os resultados obtidos ajudarão a identificar áreas críticas para otimização em termos de localidade de referência.

Nesse sentido, esses parâmetros foram escolhidos com base na complexidade de acesso à memória que eles representam e na importância de entender como o programa se comporta durante essas operações críticas.

Cachegrind

Soma de Matrizes

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ valgrind --tool=cachegrind ./bin/matop -s -p
/tmp/soma500.out -x 500 -y 500
==8222== Cachegrind, a cache and branch-prediction profiler
==8222== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==8222== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==8222== Command: ./bin/matop -s -p /tmp/soma500.out -x 500 -y 500
==8222==
--8222-- warning: L3 cache found, using its data for the LL simulation.
matop
  -s      (somar matrizes)
  -m      (multiplicar matrizes)
  -t      (transpor matriz)
  -c <arq> (cria matriz e salva em arq)
  -x <int> (primeira dimensao)
  -y <int> (segunda dimensao)
==8222==
==8222== I  refs:      139,110
==8222== I1 misses:    1,194
==8222== LLi misses:   1,174
==8222== I1 miss rate: 0.86%
==8222== LLi miss rate: 0.84%
==8222==
==8222== D  refs:      42,738 (31,673 rd + 11,065 wr)
==8222== D1 misses:    1,599 ( 1,275 rd +   324 wr)
==8222== LLd misses:   1,351 ( 1,057 rd +   294 wr)
==8222== D1 miss rate:  3.7% (  4.0% +  2.9% )
==8222== LLd miss rate: 3.2% (  3.3% +  2.7% )
==8222==
==8222== LL refs:      2,793 ( 2,469 rd +   324 wr)
==8222== LL misses:    2,525 ( 2,231 rd +   294 wr)
==8222== LL miss rate:  1.4% (  1.3% +  2.7% )
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ cg_annotate cachegrind.out.8222
```

Ir	I1mr	I1Lmr	Dr	D1mr	D1Lmr	Dw	D1mw	D1Lmw
file: function								
23,714 (17.05%)	16 (1.34%)	16 (1.36%)	7,578 (23.93%)	170 (13.33%)	151 (14.29%)	3,002 (27.13%)	7 (2.16%)	4 (1.36%)
./elf/./elf/dl-lookup.c:do_lookup_x								
22,377 (16.09%)	6 (0.50%)	6 (0.51%)	4,280 (13.51%)	62 (4.86%)	62 (5.87%)	56 (0.51%)	0	0
./elf/./elf/dl-tunables.c:_GI__tunables_init								
15,513 (11.15%)	25 (2.09%)	25 (2.13%)	2,620 (8.27%)	311 (24.39%)	281 (26.58%)	739 (6.68%)	2 (0.62%)	0
./elf/./elf/dl-reloc.c:_dl_relocate_object								
9,054 (6.51%)	10 (0.84%)	10 (0.85%)	2,605 (8.22%)	1 (0.08%)	0	2,059 (18.61%)	7 (2.16%)	3 (1.02%)
./elf/./elf/dl-lookup.c:_dl_lookup_symbol_x								
8,372 (6.02%)	3 (0.25%)	3 (0.26%)	1,380 (4.36%)	78 (6.12%)	77 (7.28%)	0	0	0
./elf/./sysdeps/generic/dl-new-hash.h:_dl_lookup_symbol_x								
7,528 (5.41%)	44 (3.69%)	44 (3.75%)	1,290 (4.07%)	56 (4.39%)	47 (4.45%)	0	0	0
./string/./sysdeps/x86_64/multiarch/./multiarch/strcmp-sse2.S:strcmp								
5,158 (3.71%)	6 (0.50%)	6 (0.51%)	1,714 (5.41%)	62 (4.86%)	49 (4.64%)	587 (5.31%)	1 (0.31%)	0
./elf/./elf/dl-lookup.c:check_match								
4,264 (3.07%)	19 (1.59%)	19 (1.62%)	1,247 (3.94%)	114 (8.94%)	105 (9.93%)	379 (3.43%)	0	0
./elf/./elf/do-rel.h:_dl_relocate_object								
3,995 (2.87%)	17 (1.42%)	17 (1.45%)	927 (2.93%)	15 (1.18%)	15 (1.42%)	215 (1.94%)	23 (7.10%)	18 (6.12%)
./elf/./sysdeps/x86_64/dl-machine.h:_dl_relocate_object								
3,730 (2.68%)	1 (0.08%)	1 (0.09%)	556 (1.76%)	1 (0.08%)	1 (0.09%)	0	0	0
./elf/./elf/dl-tunables.h:_GI__tunables_init								
3,354 (2.41%)	9 (0.75%)	9 (0.77%)	499 (1.58%)	7 (0.55%)	7 (0.66%)	78 (0.70%)	0	0
./elf/./sysdeps/x86_64/dl-cacheinfo.h:intel_check_word.constprop.0								
2,570 (1.85%)	25 (2.09%)	25 (2.13%)	850 (2.68%)	32 (2.51%)	30 (2.84%)	262 (2.37%)	21 (6.48%)	21 (7.14%)
./elf/./elf/dl-version.c:_dl_check_map_versions								
2,413 (1.73%)	2 (0.17%)	2 (0.17%)	0	0	0	0	0	0
./elf/./bits/stdlib-bsearch.h:intel_check_word.constprop.0								
1,558 (1.12%)	39 (3.27%)	39 (3.32%)	322 (1.02%)	40 (3.14%)	40 (3.78%)	161 (1.46%)	12 (3.70%)	12 (4.08%)
./elf/./elf/dl-load.c:_dl_map_object_from_fd								

Multiplicação de Matrizes

```

nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ valgrind --tool=cachegrind ./bin/matop -m -p
/tmp/mult500.out -x 500 -y 500
==7960== Cachegrind, a cache and branch-prediction profiler
==7960== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==7960== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==7960== Command: ./bin/matop -m -p /tmp/mult500.out -x 500 -y 500
==7960==
--7960-- warning: L3 cache found, using its data for the LL simulation.
matop
-s      (somar matrizes)
-m      (multiplicar matrizes)
-t      (transpor matriz)
-c <arq>      (cria matriz e salva em arq)
-x <int>      (primeira dimensao)
-y <int>      (segunda dimensao)
==7960==
==7960== I   refs:      139,110
==7960== I1  misses:      1,193
==7960== LLi misses:      1,173
==7960== I1  miss rate:    0.86%
==7960== LLi miss rate:    0.84%
==7960==
==7960== D   refs:      42,738 (31,673 rd + 11,065 wr)
==7960== D1  misses:      1,600 ( 1,276 rd +   324 wr)
==7960== LLD misses:      1,352 ( 1,058 rd +   294 wr)
==7960== D1  miss rate:    3.7% (  4.0% +  2.9% )
==7960== LLD miss rate:    3.2% (  3.3% +  2.7% )
==7960==
==7960== LL refs:        2,793 ( 2,469 rd +   324 wr)
==7960== LL  misses:      2,525 ( 2,231 rd +   294 wr)
==7960== LL  miss rate:    1.4% (  1.3% +  2.7% )
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ cg_annotate cachegrind.out.7960

```

Ir	I1mr	I1mr	Dr	D1mr	DLmr	Dw	D1mw	DLmw
file:function								
23,714 (17.05%)	16 (1.34%)	16 (1.36%)	7,578 (23.93%)	170 (13.32%)	151 (14.27%)	3,002 (27.13%)	7 (2.16%)	4 (1.36%)
./elf/./elf/dl-lookup.c:do_lookup_x								
22,377 (16.09%)	6 (0.50%)	6 (0.51%)	4,280 (13.51%)	62 (4.86%)	62 (5.86%)	56 (0.51%)	0	0
./elf/./elf/dl-tunables.c:_GI__tunables_init								
15,513 (11.15%)	25 (2.10%)	25 (2.13%)	2,620 (8.27%)	311 (24.37%)	281 (26.56%)	739 (6.68%)	2 (0.62%)	0
./elf/./elf/dl-reloc.c:_dl_relocate_object								
9,054 (6.51%)	10 (0.84%)	10 (0.85%)	2,605 (8.22%)	1 (0.08%)	0	2,059 (18.61%)	7 (2.16%)	3 (1.02%)
./elf/./elf/dl-lookup.c:_dl_lookup_symbol_x								
8,372 (6.02%)	3 (0.25%)	3 (0.26%)	1,380 (4.36%)	78 (6.11%)	77 (7.28%)	0	0	0
./elf/./sysdeps/generic/dl-new-hash.h:_dl_lookup_symbol_x								
7,528 (5.41%)	44 (3.69%)	44 (3.75%)	1,290 (4.07%)	56 (4.39%)	47 (4.44%)	0	0	0
./string/./sysdeps/x86_64/multiarch/./multiarch/strcmp-sse2.S:strcmp								
5,158 (3.71%)	6 (0.50%)	6 (0.51%)	1,714 (5.41%)	62 (4.86%)	49 (4.63%)	587 (5.31%)	1 (0.31%)	0
./elf/./elf/dl-lookup.c:check_match								
4,264 (3.07%)	19 (1.59%)	19 (1.62%)	1,247 (3.94%)	114 (8.93%)	105 (9.92%)	379 (3.43%)	0	0
./elf/./elf/do-rel.h:_dl_relocate_object								
3,995 (2.87%)	17 (1.42%)	17 (1.45%)	927 (2.93%)	15 (1.18%)	15 (1.42%)	215 (1.94%)	23 (7.10%)	18 (6.12%)
./elf/./sysdeps/x86_64/dl-machine.h:_dl_relocate_object								
3,730 (2.68%)	1 (0.08%)	1 (0.09%)	556 (1.76%)	1 (0.08%)	1 (0.09%)	0	0	0
./elf/./elf/dl-tunables.h:_GI__tunables_init								
3,354 (2.41%)	9 (0.75%)	9 (0.77%)	499 (1.58%)	7 (0.55%)	7 (0.66%)	78 (0.70%)	0	0
./elf/./sysdeps/x86_64/dl-cacheinfo.h:intel_check_word.constprop.0								
2,570 (1.85%)	25 (2.10%)	25 (2.13%)	850 (2.68%)	32 (2.51%)	30 (2.84%)	262 (2.37%)	21 (6.48%)	21 (7.14%)
./elf/./elf/dl-version.c:_dl_check_map_versions								
2,413 (1.73%)	2 (0.17%)	2 (0.17%)	0	0	0	0	0	0
./elf/./bits/stdlib-bsearch.h:intel_check_word.constprop.0								
1,558 (1.12%)	39 (3.27%)	39 (3.32%)	322 (1.02%)	40 (3.13%)	40 (3.78%)	161 (1.46%)	12 (3.70%)	12 (4.08%)
./elf/./elf/dl-load.c:_dl_map_object_from_fd								
1,167 (0.84%)	64 (5.36%)	64 (5.46%)	282 (0.89%)	31 (2.43%)	20 (1.89%)	117 (1.06%)	6 (1.85%)	6 (2.04%)

Transposição de Matrizes

```

nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/ativ4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/ativ4/pa04$ valgrind --tool=cachegrind ./bin/matop -t -p
/tmp/transp500.out -x 500 -y 500
==8394== Cachegrind, a cache and branch-prediction profiler
==8394== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==8394== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==8394== Command: ./bin/matop -t -p /tmp/transp500.out -x 500 -y 500
==8394==
--8394-- warning: L3 cache found, using its data for the LL simulation.
matop
-s      (somar matrizes)
-m      (multiplicar matrizes)
-t      (transpor matriz)
-c <arq>      (cria matriz e salva em arq)
-x <int>      (primeira dimensao)
-y <int>      (segunda dimensao)
==8394==
==8394== I refs:      139,110
==8394== I1 misses:    1,195
==8394== LI misses:    1,175
==8394== I1 miss rate: 0.86%
==8394== LI miss rate: 0.84%
==8394==
==8394== D refs:      42,738 (31,673 rd + 11,065 wr)
==8394== D1 misses:    1,599 ( 1,275 rd +   324 wr)
==8394== LD misses:    1,351 ( 1,057 rd +   294 wr)
==8394== D1 miss rate:  3.7% (  4.0% +  2.9% )
==8394== LD miss rate:  3.2% (  3.3% +  2.7% )
==8394==
==8394== LL refs:      2,794 ( 2,470 rd +   324 wr)
==8394== LL misses:    2,526 ( 2,232 rd +   294 wr)
==8394== LL miss rate:  1.4% (  1.3% +  2.7% )
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/ativ4/pa04$ cg_annotate cachegrind.out.8394

```


nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/ativ4/pa04

file:function	Ir	I1mr	I1Lmr	Dr	D1mr	D1Lmr	Dw	D1mw	D1Lmw
23,714 (17.05%)	16 (1.34%)	16 (1.36%)	7,578 (23.93%)	170 (13.33%)	151 (14.29%)	3,002 (27.13%)	7 (2.16%)	4 (1.36%)	
./elf/./elf/dl-lookup.c:do_lookup_x									
22,377 (16.09%)	6 (0.50%)	6 (0.51%)	4,280 (13.51%)	62 (4.86%)	62 (5.87%)	56 (0.51%)	0	0	
./elf/./elf/dl-tunables.c:_GI__tunables_init									
15,513 (11.15%)	25 (2.09%)	25 (2.13%)	2,620 (8.27%)	311 (24.39%)	281 (26.58%)	739 (6.68%)	2 (0.62%)	0	
./elf/./elf/dl-reloc.c:_dl_relocate_object									
9,054 (6.51%)	10 (0.84%)	10 (0.85%)	2,605 (8.22%)	1 (0.08%)	0	2,059 (18.61%)	7 (2.16%)	3 (1.02%)	
./elf/./elf/dl-lookup.c:_dl_lookup_symbol_x									
8,372 (6.02%)	3 (0.25%)	3 (0.26%)	1,380 (4.36%)	78 (6.12%)	77 (7.28%)	0	0	0	
./elf/./sysdeps/generic/dl-new-hash.h:_dl_lookup_symbol_x									
7,528 (5.41%)	44 (3.68%)	44 (3.74%)	1,290 (4.07%)	56 (4.45%)	0	0	0	0	
./string/./sysdeps/x86_64/multiarch/./multiarch/strcmp-sse2.S:strcmp									
5,158 (3.71%)	6 (0.50%)	6 (0.51%)	1,714 (5.41%)	62 (4.86%)	49 (4.64%)	587 (5.31%)	1 (0.31%)	0	
./elf/./elf/dl-lookup.c:check_match									
4,264 (3.07%)	19 (1.59%)	19 (1.62%)	1,247 (3.94%)	114 (8.94%)	105 (9.93%)	379 (3.43%)	0	0	
./elf/./elf/do-rel.h:_dl_relocate_object									
3,995 (2.87%)	17 (1.42%)	17 (1.45%)	927 (2.93%)	15 (1.18%)	15 (1.42%)	215 (1.94%)	23 (7.10%)	18 (6.12%)	
./elf/./sysdeps/x86_64/dl-machine.h:_dl_relocate_object									
3,730 (2.68%)	1 (0.08%)	1 (0.09%)	556 (1.76%)	1 (0.08%)	1 (0.09%)	0	0	0	
./elf/./elf/dl-tunables.h:_GI__tunables_init									
3,354 (2.41%)	9 (0.75%)	9 (0.77%)	499 (1.58%)	7 (0.55%)	7 (0.66%)	78 (0.70%)	0	0	
./elf/./sysdeps/x86/dl-cacheinfo.h:intel_check_word.constprop.0									
2,570 (1.85%)	25 (2.09%)	25 (2.13%)	850 (2.68%)	32 (2.51%)	30 (2.84%)	262 (2.37%)	21 (6.48%)	21 (7.14%)	
./elf/./elf/dl-version.c:_dl_check_map_versions									
2,413 (1.73%)	2 (0.17%)	2 (0.17%)	0	0	0	0	0	0	
./elf/./bits/stdlib-bsearch.h:intel_check_word.constprop.0									
1,558 (1.12%)	39 (3.26%)	39 (3.32%)	322 (1.02%)	40 (3.14%)	40 (3.78%)	161 (1.46%)	12 (3.70%)	12 (4.08%)	
./elf/./elf/dl-load.c:_dl_map_object_from_fd									
1,167 (0.84%)	64 (5.36%)	64 (5.45%)	282 (0.89%)	31 (2.43%)	20 (1.89%)	117 (1.06%)	6 (1.85%)	6 (2.04%)	

Multiplicação de Matrizes com Registro de Acesso

nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/ativ4/pa04

```
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/ativ4/pa04$ valgrind --tool=cachegrind ./bin/matop -m -p /tmp/multi00log.out -x 100 -y 100
==8548== Cachegrind, a cache and branch-prediction profiler
==8548== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==8548== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==8548== Command: ./bin/matop -m -p /tmp/multi00log.out -x 100 -y 100
==8548==
--8548-- warning: L3 cache found, using its data for the LL simulation.
matop
-s      (somar matrizes)
-m      (multiplicar matrizes)
-t      (transpor matriz)
-c <arq>      (cria matriz e salva em arq)
-x <int>      (primeira dimensao)
-y <int>      (segunda dimensao)
==8548==
==8548== I refs:      139,096
==8548== I1 misses:    1,193
==8548== LLi misses:    1,173
==8548== I1 miss rate:  0.86%
==8548== LLi miss rate: 0.84%
==8548==
==8548== D refs:      42,730 (31,665 rd + 11,065 wr)
==8548== D1 misses:  1,600 ( 1,276 rd +   324 wr)
==8548== LLd misses: 1,352 ( 1,058 rd +   294 wr)
==8548== D1 miss rate: 3.7% ( 4.0% + 2.9% )
==8548== LLd miss rate: 3.2% ( 3.3% + 2.7% )
==8548==
==8548== LL refs:      2,793 ( 2,469 rd +   324 wr)
==8548== LL misses:  2,525 ( 2,231 rd +   294 wr)
==8548== LL miss rate: 1.4% ( 1.3% + 2.7% )
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/ativ4/pa04$ cg_annotate cachegrind.out.8548
```

Ir	I1mr	ILmr	Dr	D1mr	DLmr	Dw	D1mw	DLmw
file:function								
23,714 (17.05%)	16 (1.34%)	16 (1.36%)	7,578 (23.93%)	170 (13.32%)	151 (14.27%)	3,002 (27.13%)	7 (2.16%)	4 (1.36%)
./elf/./elf/dl-lookup.c:do_lookup_x								
22,377 (16.09%)	6 (0.50%)	6 (0.51%)	4,280 (13.52%)	62 (4.86%)	62 (5.86%)	56 (0.51%)	0	0
./elf/./elf/dl-tunables.c:___GI___tunables_init								
15,513 (11.15%)	25 (2.10%)	25 (2.13%)	2,620 (8.27%)	311 (24.37%)	281 (26.56%)	739 (6.68%)	2 (0.62%)	0
./elf/./elf/dl-reloc.c:_dl_relocate_object								
9,054 (6.51%)	10 (0.84%)	10 (0.85%)	2,605 (8.23%)	1 (0.08%)	0	2,059 (18.61%)	7 (2.16%)	3 (1.02%)
./elf/./elf/dl-lookup.c:_dl_lookup_symbol_x								
8,372 (6.02%)	3 (0.25%)	3 (0.26%)	1,380 (4.36%)	78 (6.11%)	77 (7.28%)	0	0	0
./elf/./sysdeps/generic/dl-new-hash.h:_dl_lookup_symbol_x								
7,528 (5.41%)	44 (3.69%)	44 (3.75%)	1,290 (4.07%)	56 (4.39%)	47 (4.44%)	0	0	0
./string/./sysdeps/x86_64/multiarch/./multiarch/strcmp-sse2.S:strcmp								
5,158 (3.71%)	6 (0.50%)	6 (0.51%)	1,714 (5.41%)	62 (4.86%)	49 (4.63%)	587 (5.31%)	1 (0.31%)	0
./elf/./elf/dl-lookup.c:check_match								
4,264 (3.07%)	19 (1.59%)	19 (1.62%)	1,247 (3.94%)	114 (8.93%)	105 (9.92%)	379 (3.43%)	0	0
./elf/./elf/do-rel.h:_dl_relocate_object								
3,995 (2.87%)	17 (1.42%)	17 (1.45%)	927 (2.93%)	15 (1.18%)	15 (1.42%)	215 (1.94%)	23 (7.10%)	18 (6.12%)
./elf/./sysdeps/x86_64/dl-machine.h:_dl_relocate_object								
3,730 (2.68%)	1 (0.08%)	1 (0.09%)	556 (1.76%)	1 (0.08%)	1 (0.09%)	0	0	0
./elf/./elf/dl-tunables.h:___GI___tunables_init								
3,354 (2.41%)	9 (0.75%)	9 (0.77%)	499 (1.58%)	7 (0.55%)	7 (0.66%)	78 (0.70%)	0	0
./elf/./sysdeps/x86_64/dl-cacheinfo.h:intel_check_word.constprop.0								
2,570 (1.85%)	25 (2.10%)	25 (2.13%)	850 (2.68%)	32 (2.51%)	30 (2.84%)	262 (2.37%)	21 (6.48%)	21 (7.14%)
./elf/./elf/dl-version.c:_dl_check_map_versions								
2,413 (1.73%)	2 (0.17%)	2 (0.17%)	0	0	0	0	0	0
./elf/./bits/stdlib-bsearch.h:intel_check_word.constprop.0								
1,558 (1.12%)	39 (3.27%)	39 (3.32%)	322 (1.02%)	40 (3.13%)	40 (3.78%)	161 (1.46%)	12 (3.70%)	12 (4.08%)
./elf/./elf/dl-load.c:_dl_map_object_from_fd								
1,167 (0.84%)	64 (5.36%)	64 (5.46%)	282 (0.89%)	31 (2.43%)	20 (1.89%)	117 (1.06%)	6 (1.85%)	6 (2.04%)

Callgrind

Soma de Matrizes

```

nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ valgrind --tool=callgrind ./bin/matop -s -p
/tmp/soma500.out -x 500 -y 500
==9189== Callgrind, a call-graph generating cache profiler
==9189== Copyright (C) 2002-2017, and GNU GPL'd, by Josef Weidendorfer et al.
==9189== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==9189== Command: ./bin/matop -s -p /tmp/soma500.out -x 500 -y 500
==9189==
==9189== For interactive control, run 'callgrind_control -h'.
matop
-s      (somar matrizes)
-m      (multiplicar matrizes)
-t      (transpor matriz)
-c <arq>      (cria matriz e salva em arq)
-x <int>      (primeira dimensao)
-y <int>      (segunda dimensao)
==9189==
==9189== Events      : Ir
==9189== Collected : 137811
==9189==
==9189== I   refs:      137,811
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ callgrind_annotate callgrind.out.8189

```

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ callgrind_annotate callgrind.out.9189
-----
Profile data file 'callgrind.out.9189' (creator: callgrind-3.19.0)
-----
I1 cache:
D1 cache:
LL cache:
Timerange: Basic block 0 - 30981
Trigger: Program termination
Profiled target: ./bin/matop -s -p /tmp/soma500.out -x 500 -y 500 (PID 9189, part 1)
Events recorded: Ir
Events shown: Ir
Event sort order: Ir
Thresholds: 99
Include dirs:
User annotated:
Auto-annotation: on
-----
Ir
```

Multiplicação de Matrizes

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ valgrind --tool=callgrind ./bin/matop -m -p /tmp/mult500.out -x 500 -y 500
==9741== Callgrind, a call-graph generating cache profiler
==9741== Copyright (C) 2002-2017, and GNU GPL'd, by Josef Weidendorfer et al.
==9741== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==9741== Command: ./bin/matop -m -p /tmp/mult500.out -x 500 -y 500
==9741==
==9741== For interactive control, run 'callgrind_control -h'.
matop
  -s      (somar matrizes)
  -m      (multiplicar matrizes)
  -t      (transpor matriz)
  -c <arq> (cria matriz e salva em arq)
  -x <int> (primeira dimensao)
  -y <int> (segunda dimensao)
==9741==
==9741== Events      : Ir
==9741== Collected : 137811
==9741==
==9741== I   refs:      137,811
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$
```

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ callgrind_annotate callgrind.out.9741
-----
Profile data file 'callgrind.out.9741' (creator: callgrind-3.19.0)
-----
I1 cache:
D1 cache:
LL cache:
Timerange: Basic block 0 - 30981
Trigger: Program termination
Profiled target: ./bin/matop -m -p /tmp/mult500.out -x 500 -y 500 (PID 9741, part 1)
Events recorded: Ir
Events shown: Ir
Event sort order: Ir
Thresholds: 99
Include dirs:
User annotated:
Auto-annotation: on
-----
Ir
```

Transposição de Matrizes

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ valgrind --tool=callgrind ./bin/matop -t -p /tmp/transp500.out -x 500 -y 500
==10051== Callgrind, a call-graph generating cache profiler
==10051== Copyright (C) 2002-2017, and GNU GPL'd, by Josef Weidendorfer et al.
==10051== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==10051== Command: ./bin/matop -t -p /tmp/transp500.out -x 500 -y 500
==10051==
==10051== For interactive control, run 'callgrind_control -h'.
matop
  -s      (somar matrizes)
  -m      (multiplicar matrizes)
  -t      (transpor matriz)
  -c <arq> (cria matriz e salva em arq)
  -x <int> (primeira dimensao)
  -y <int> (segunda dimensao)
==10051==
==10051== Events      : Ir
==10051== Collected : 137811
==10051==
==10051== I   refs:      137,811
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ callgrind_annotate callgrind.out.9741
```

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ callgrind_annotate callgrind.out.10051
-----
Profile data file 'callgrind.out.10051' (creator: callgrind-3.19.0)
-----
I1 cache:
D1 cache:
LL cache:
Timerange: Basic block 0 - 30981
Trigger: Program termination
Profiled target: ./bin/matop -t -p /tmp/transp500.out -x 500 -y 500 (PID 10051, part 1)
Events recorded: Ir
Events shown: Ir
Event sort order: Ir
Thresholds: 99
Include dirs:
User annotated:
Auto-annotation: on
-----
Ir
```

Multiplicação de Matrizes com Registro de Acesso

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ valgrind --tool=callgrind ./bin/matop -m -p /tmp/multi00log.out -x 100 -y 100
==10334== Callgrind, a call-graph generating cache profiler
==10334== Copyright (C) 2002-2017, and GNU GPL'd, by Josef Weidendorfer et al.
==10334== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==10334== Command: ./bin/matop -m -p /tmp/multi00log.out -x 100 -y 100
==10334==
==10334== For interactive control, run 'callgrind_control -h'.
matop
  -s      (somar matrizes)
  -m      (multiplicar matrizes)
  -t      (transpor matriz)
  -c <arq> (cria matriz e salva em arq)
  -x <int> (primeira dimensao)
  -y <int> (segunda dimensao)
==10334==
==10334== Events      : Ir
==10334== Collected : 137797
==10334==
==10334== I   refs:      137,797
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ callgrind_annotate callg
```

```
nicolasvondolinger@nicolasvondolinger-82MF: ~/Codes/data_structure/atv4/pa04
nicolasvondolinger@nicolasvondolinger-82MF:~/Codes/data_structure/atv4/pa04$ callgrind_annotate callgrind.out.10334
-----
Profile data file 'callgrind.out.10334' (creator: callgrind-3.19.0)
-----
I1 cache:
D1 cache:
LL cache:
Timerange: Basic block 0 - 30980
Trigger: Program termination
Profiled target: ./bin/matop -m -p /tmp/multi00log.out -x 100 -y 100 (PID 10334, part 1)
Events recorded: Ir
Events shown: Ir
Event sort order: Ir
Thresholds: 99
Include dirs:
User annotated:
Auto-annotation: on
-----
Ir
```

Análise das Saídas

Cachegrind

1. Quão bem o programa se comporta em termos de memória?

O relatório referente ao código mostra que a função `do_lookup_x` e `__GI__tunables_init` estão consumindo uma quantidade significativa de recursos. No geral, o programa executou 139.110 instruções (Ir) e teve 31.673 leituras de dados (Dr). A maioria dos eventos está associada a instruções de leitura (I1mr, ILmr) e leituras de dados (Dr, D1mr, DLmr).

Essas funções são responsáveis por uma parcela substancial do tempo de execução do programa, com `do_lookup_x` representando 17,05% das instruções executadas (Ir) e `__GI__tunables_init` contribuindo com 16,09%. Outras funções, como `_dl_relocate_object` e `_dl_lookup_symbol_x,` também são relevantes em termos de instruções executadas e leituras de dados. Com base nesses dados, é evidente que as funções `do_lookup_x` e `__GI__tunables_init` são as principais áreas a serem otimizadas em termos de consumo de recursos.

Em geral, com base nas informações fornecidas, o programa parece se comportar bem em termos de memória. A alta porcentagem de acertos na memória cache sugere que o programa está aproveitando eficientemente o uso da cache para reduzir os acessos à memória principal, o que é fundamental para o desempenho de muitas aplicações.

2. Quais estruturas de dados devem ser caracterizadas para melhor entendimento?

Hierarquia de Cache: Entender a hierarquia de cache, incluindo o tamanho e a associatividade de cada nível de cache (L1, L2, L3, etc.), ajuda a determinar como os dados estão sendo armazenados e acessados em diferentes camadas de cache.

Acessos à Memória: Analisar as métricas de leitura (read) e escrita (write) para cada nível de cache, bem como os acertos (hits) e falhas (misses), fornece informações sobre a eficiência do uso da memória cache. Isso inclui o número absoluto de acessos e a porcentagem de acertos em cada nível.

Padrões de Acesso: Identificar os padrões de acesso à memória, como localidade espacial (acessos a dados próximos na memória) e localidade temporal (reutilização de dados recentemente acessados), é crucial para entender como o programa está acessando os dados.

Funções Críticas: Analisar as estatísticas de acesso à memória para funções específicas no código-fonte pode ajudar a identificar quais partes do programa estão contribuindo mais para o tráfego de memória e onde podem ocorrer oportunidades de otimização.

Tamanho dos Dados: Avaliar o tamanho dos dados usados pelo programa pode ajudar a determinar se os dados cabem eficientemente na memória cache disponível ou se há necessidade de estratégias de otimização, como particionamento de dados.

Padrões de Escrita: Entender como as operações de escrita afetam a memória cache e a memória principal é importante para otimizar o programa. Pode ser útil identificar padrões de escrita, como escrita única ou escrita múltipla em um mesmo local de memória.

Dependência de Dados: Identificar dependências de dados entre diferentes partes do programa pode ajudar a determinar a ordem de execução das operações e a minimizar os conflitos na memória cache.

Políticas de Substituição e Escrita: Conhecer as políticas de substituição de cache (por exemplo, LRU, FIFO) e as políticas de escrita (por exemplo, write-through, write-back) usadas pelo hardware de cache é importante para entender como os dados são gerenciados na memória cache.

Tamanho dos Blocos de Cache: Compreender o tamanho dos blocos de cache (também conhecidos como linhas de cache) é importante para entender como os dados são agrupados e carregados na memória cache.

Instruções vs. Dados: Distinguir entre acessos à memória para instruções e dados é fundamental, pois os padrões de acesso podem variar significativamente entre os dois tipos de acesso.

3. Quais segmentos de código devem ser instrumentados para suportar a caracterização?

Para suportar a caracterização do desempenho de um programa, é importante instrumentar os segmentos de código que são críticos para a análise de cache e memória. Aqui estão alguns segmentos de código que devem ser instrumentados:

Pontos de Entrada: como a função `main()` ou outras funções principais que iniciam a execução. Isso permite rastrear o comportamento do programa desde o início.

Funções Críticas: Isso pode incluir funções que realizam cálculos intensivos, operações de E/S (entrada e saída), ou funções que manipulam grandes volumes de dados.

Laços de Processamento: laços que iteram sobre estruturas de dados grandes.

Acessos à Memória: acessos à memória, incluindo leituras e escritas em variáveis e estruturas de dados. Isso ajudará a rastrear onde os dados estão sendo acessados e modificados.

Pontos Críticos de Cache: pontos críticos onde os dados são frequentemente carregados na memória cache e onde os acertos ou falhas de cache têm um impacto significativo no desempenho. Isso pode incluir operações de busca ou acesso a estruturas de dados compartilhadas.

Acesso a Estruturas de Dados Complexas: como matrizes multidimensionais, listas encadeadas ou árvores, que podem ter padrões de acesso à memória distintos.

Callgrind

1. Quão bem o programa se comporta em termos de memória?

Com base apenas no perfil do Callgrind fornecido, não se pode tirar conclusões definitivas sobre o comportamento do programa em termos de memória. Para avaliar o uso de memória do programa, deveria ser realizada uma análise de perfil de memória usando as ferramentas apropriadas. No entanto, para todos os códigos de manipulação de matrizes, o número total de instruções retiradas é 137.811.

As estruturas de dados e os segmentos de código são os mesmos da análise feita do cachegrind.

Conclusão

O código fornecido apresenta uma complexidade de acesso à memória significativa devido às operações em matrizes e ao potencial uso de estruturas de dados complexas. A análise qualitativa revelou a presença esperada de localidade de referência espacial e temporal, bem como a identificação das estruturas de dados e segmentos de código críticos que podem afetar o desempenho.

A caracterização de localidade de referência é fundamental para a otimização do código, permitindo a identificação de oportunidades de melhoria no acesso à memória e, conseqüentemente, no desempenho geral. Este relatório fornece uma visão geral inicial da complexidade de acesso à memória no código e serve como ponto de partida para análises mais aprofundadas e otimizações futuras.