

Alexandre Gabriel Angelo de Souza Blandino, Murilo Caetano de
Melo Pereira e Samuel da Rocha Villela

Avaliação de Desempenho da Multiplicação de Matrizes Utilizando PAPI

Nova Iguaçu, RJ

2025



Avaliação de Desempenho da Multiplicação de Matrizes Utilizando PAPI

Alunos: Alexandre Gabriel Angelo de Souza Blandino, Murilo Caetano de
Melo Pereira e Samuel da Rocha Villela

Docente: Prof. Marcelo Zamith

Nova Iguaçu
2025

Resumo

Este trabalho apresenta uma avaliação de desempenho da multiplicação de matrizes quadradas utilizando a biblioteca PAPI (Performance API) para a coleta de métricas de hardware. Foram analisados diferentes níveis de otimização do compilador (-O0 e -O3), bem como o impacto do hardware, comparando uma máquina moderna e uma máquina mais antiga.

As métricas avaliadas incluem tempo de execução, número de instruções executadas, ciclos de clock e falhas de cache nos níveis L1 e L2. Os resultados evidenciam ganhos significativos com o uso de otimizações de compilação e demonstram a forte influência da hierarquia de memória no desempenho computacional.

O relatório foi elaborado em LaTeX, utilizando o editor Overleaf.

Sumário

1	INTRODUÇÃO	5
2	METODOLOGIA	6
2.1	Compilação	6
2.2	Métricas Coletadas	6
2.3	Ambiente Experimental e Reprodutibilidade	6
2.3.1	Código-Fonte	6
2.3.2	Dados e Geração de Gráficos	7
2.3.3	Especificações dos Computadores	7
3	AMBIENTE EXPERIMENTAL	8
4	RESULTADO E ANÁLISES	9
4.1	Tempo de Execução	9
4.1.1	Comparação entre PC 1 e PC 2	9
4.1.2	Comparação entre os níveis de otimização	10
4.2	Instruções Totais	11
4.3	Ciclos Totais	13
4.4	IPC (Instrução por ciclo)	15
4.5	Falhas de Cache	17
4.6	Speedup	19
4.7	Comparação entre -O3 e -O3 com Código Otimizado	19
5	CONCLUSÃO	21

6	REFERÊNCIAS	22
----------	------------------------------	-----------

1 Introdução

A multiplicação de matrizes é uma operação fundamental em diversas áreas da computação, como computação científica, aprendizado de máquina e processamento de sinais. Por ser computacionalmente intensiva e apresentar forte dependência da hierarquia de memória, essa operação é amplamente utilizada como benchmark para avaliação de desempenho.

Neste trabalho, utilizamos a biblioteca PAPI (Performance API) para instrumentar um código de multiplicação de matrizes e coletar métricas de hardware. O objetivo é analisar o impacto das otimizações do compilador, do hardware e do acesso à memória no desempenho da aplicação.

2 Metodologia

Os experimentos foram realizados a partir do código de multiplicação de matrizes disponibilizado pelo professor, instrumentado com a biblioteca PAPI para coleta de métricas de desempenho.

Foram utilizadas matrizes quadradas de dimensões 1000×1000 , 2000×2000 , 3000×3000 e 4000×4000 , sempre realizando a multiplicação da matriz por ela mesma ($C = A \times A$).

2.1 Compilação

O código foi compilado utilizando o compilador `gcc`, com os níveis de otimização `-O0` (sem otimização) e `-O3` (máxima otimização). Além disso, foi desenvolvida uma versão adicional do código com otimizações manuais, executada com a flag `-O3`.

2.2 Métricas Coletadas

As métricas coletadas por meio da biblioteca PAPI foram:

- Tempo de execução;
- Número total de instruções executadas (`PAPI_TOT_INS`);
- Número total de ciclos de clock (`PAPI_TOT_CYC`);
- Falhas de cache L1 de dados (`PAPI_L1_DCM`);
- Falhas de cache L2 de dados (`PAPI_L2_DCM`).

A partir dessas métricas, foram derivadas métricas adicionais, como o IPC (Instruções por Ciclo) e o speedup.

2.3 Ambiente Experimental e Reprodutibilidade

2.3.1 Código-Fonte

O código-fonte utilizado para a implementação dos algoritmos, execução dos experimentos e coleta dos dados está disponível publicamente no repositório GitHub:

<https://github.com/Samuel-Villela/Desempenho-da-multiplica-o-de-matrizes->

O repositório contém os códigos em linguagem C, scripts auxiliares e instruções para compilação e execução dos experimentos.

2.3.2 Dados e Geração de Gráficos

Os dados coletados durante os experimentos foram organizados e analisados por meio do Google Planilhas. Os gráficos apresentados neste trabalho foram gerados a partir dessas planilhas, disponíveis no link a seguir:

https://docs.google.com/spreadsheets/d/1tWa5Rndf8PmRnzM2VMLfBw7BCvk5_6k04sWNUrU83Lk/edit?usp=sharing

2.3.3 Especificações dos Computadores

Os experimentos foram executados em dois computadores distintos, denominados PC1 e PC2. A Tabela 1 apresenta uma comparação entre as principais especificações de hardware e software de cada máquina, considerando apenas características comuns a ambos os ambientes.

Tabela 1 – Comparação das especificações dos computadores utilizados

Componente	PC1	PC2
Processador	AMD Ryzen 5 5600G	Intel Core i3-2120
Microarquitetura	Zen 3	Sandy Bridge (32 nm)
Núcleos / Threads	6 / 12	2 / 4
Frequência Base	3.9 GHz	3.30 GHz
Cache L2	3 MB (512 KB por núcleo)	512 KB (256 KB por núcleo)
Cache L3	16 MB (compartilhado)	3 MB (compartilhado)
Memória RAM	16 GB	8 GB
Arquitetura	x86_64	x86_64
Sistema Operacional	Ubuntu 22.04.5 LTS (64 bits)	Linux (64 bits)
Compilador	GCC	GCC 14.2.0

3 Ambiente Experimental

Os experimentos foram realizados em duas máquinas distintas, uma considerada moderna e outra mais antiga, com o objetivo de analisar o impacto do hardware no desempenho.

O sistema operacional utilizado foi Linux, e o compilador utilizado foi o GCC. Todas as execuções foram realizadas em ambiente nativo, sem o uso de máquinas virtuais, devido às limitações da biblioteca PAPI.

4 Resultado e Análises

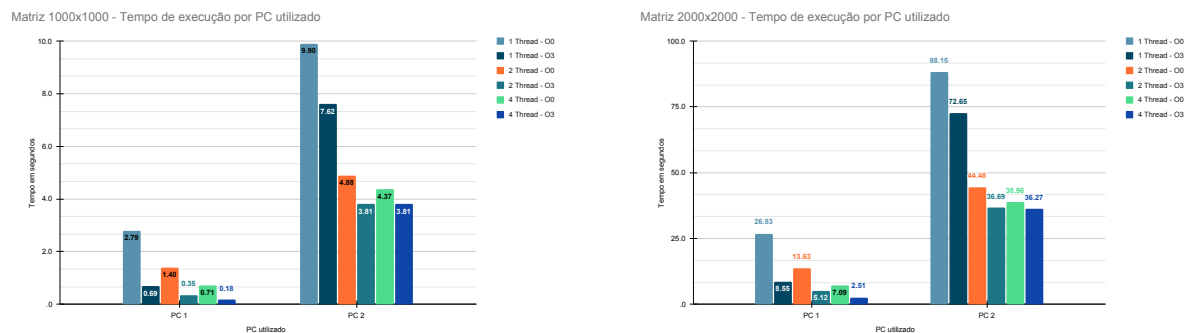
Nesta seção são apresentados os resultados obtidos a partir dos experimentos, organizados em gráficos e tabelas, seguidos de uma análise comparativa.

4.1 Tempo de Execução

Nesta seção são apresentados os resultados referentes ao tempo de execução da multiplicação de matrizes quadradas. Os gráficos foram organizados de forma a evidenciar separadamente o impacto do hardware e do nível de otimização do compilador, considerando matrizes de diferentes dimensões.

4.1.1 Comparação entre PC 1 e PC 2

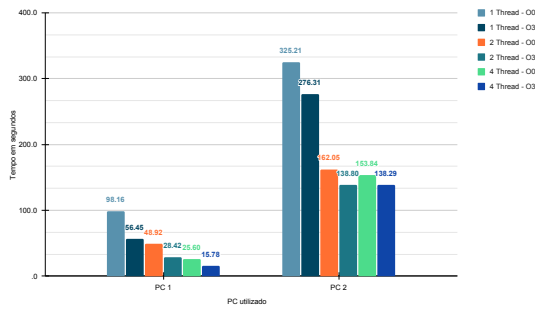
Nesta subseção, o tamanho da matriz é mantido fixo, variando-se o ambiente de execução entre PC 1 (máquina mais moderna) e PC 2 (máquina mais antiga), bem como o número de threads e o nível de otimização.



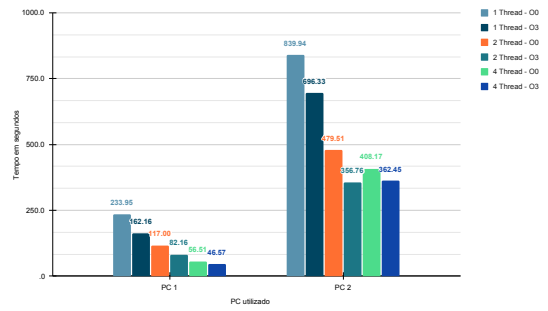
(a) Tempo de execução para matriz 1000×1000 . (b) Tempo de execução para matriz 2000×2000 .

Figura 1 – Comparação do tempo de execução entre PC 1 e PC 2 para diferentes números de threads e níveis de otimização.

Matriz 3000x3000 - Tempo de execução por PC utilizado



Matriz 4000x4000 - Tempo de execução por PC utilizado



(a) Tempo de execução para matriz 3000×3000 .

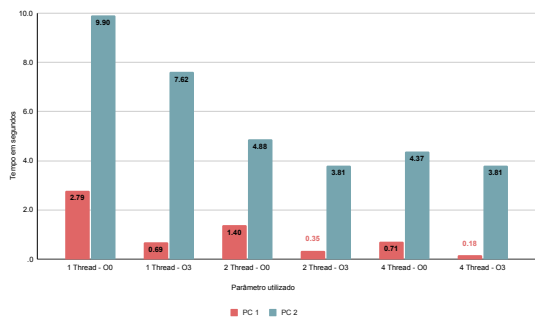
(b) Tempo de execução para matriz 4000×4000 .

Figura 2 – Comparação do tempo de execução entre PC 1 e PC 2 para matrizes de maior dimensão.

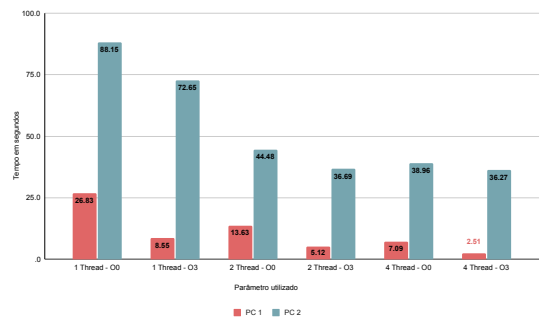
4.1.2 Comparação entre os níveis de otimização

Nesta subseção, o tamanho da matriz é mantido fixo, comparando-se diretamente os níveis de otimização -O0 e -O3, considerando diferentes números de threads e ambientes de execução.

Matriz 1000x1000 - Tempo de execução por parâmetro utilizado



Matriz 2000x2000 - Tempo de execução por parâmetro utilizado

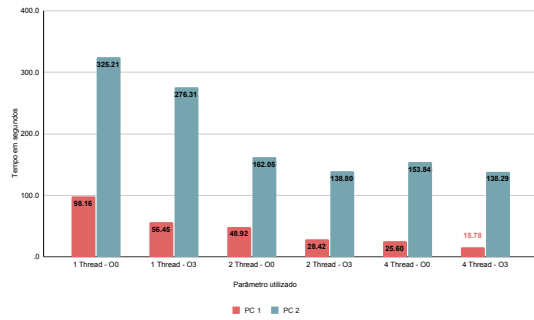


(a) Tempo de execução para matriz 1000×1000 .

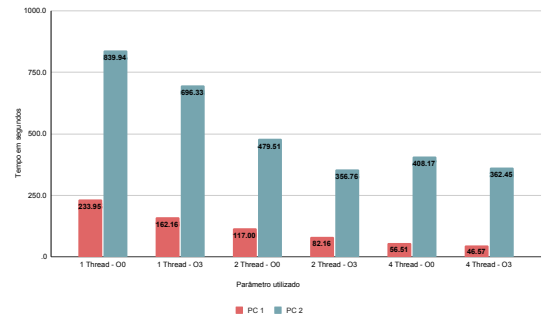
(b) Tempo de execução para matriz 2000×2000 .

Figura 3 – Comparação do tempo de execução entre os níveis de otimização -O0 e -O3.

Matriz 3000x3000 - Tempo de execução por parâmetro utilizado



Matriz 4000x4000 - Tempo de execução por parâmetro utilizado



(a) Tempo de execução para matriz 3000×3000 .

(b) Tempo de execução para matriz 4000×4000 .

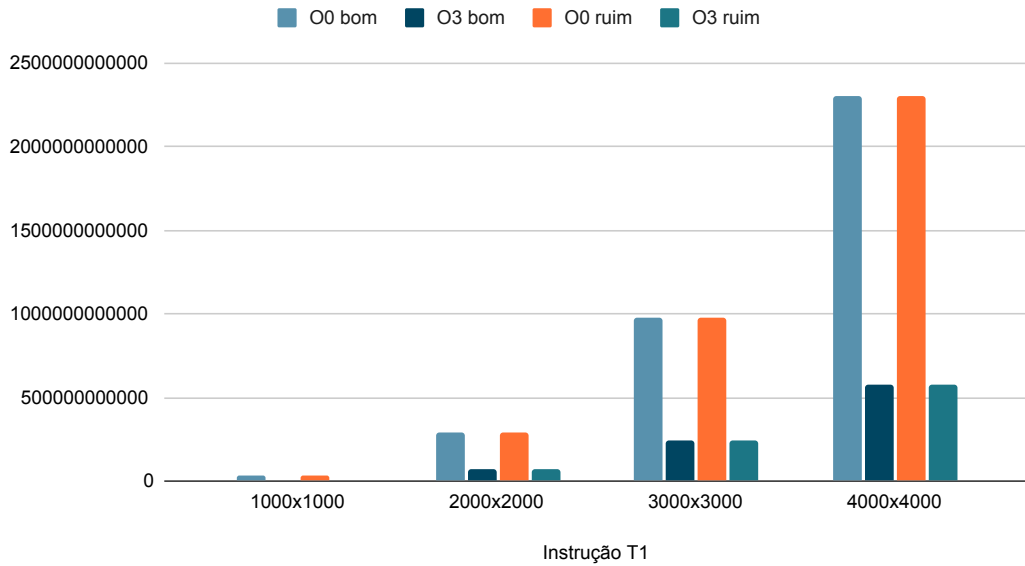
Figura 4 – Comparação do tempo de execução entre os níveis de otimização para matrizes de maior dimensão.

Os resultados mostram que, para um mesmo tamanho de matriz, o PC 1 apresenta menor tempo de execução em todas as configurações analisadas, refletindo sua arquitetura mais moderna. Além disso, a utilização da otimização `-O3` reduz significativamente o tempo de execução em relação à versão `-O0`, sendo o ganho mais expressivo à medida que o tamanho da matriz aumenta.

4.2 Instruções Totais

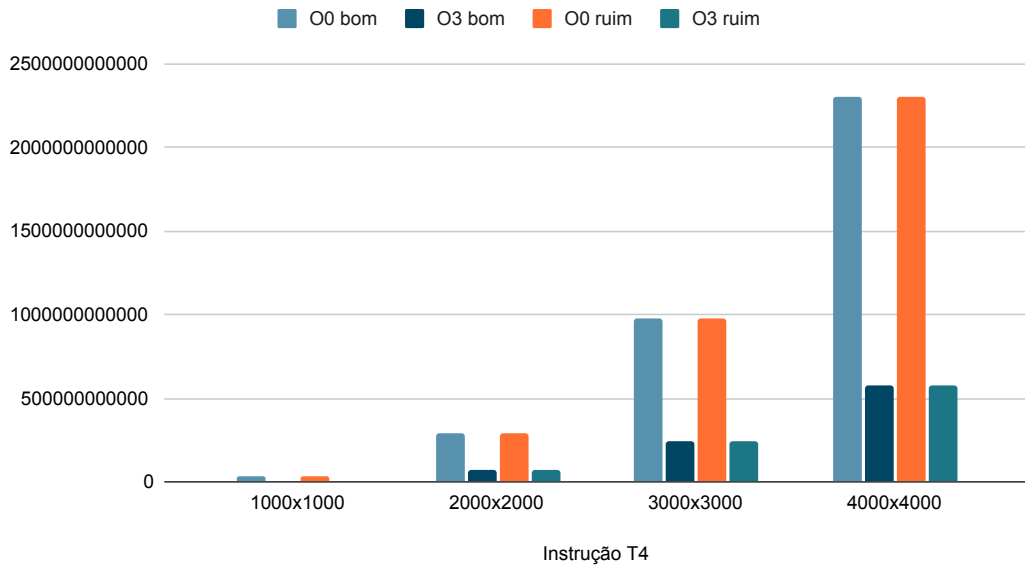
Nesta seção são apresentados os gráficos referentes ao número total de instruções executadas em função do tamanho da matriz, considerando execuções com 1, 2 e 4 threads.

Instruções totais por tamanho da matriz com 1 thread



(a) Instruções totais com 1 thread.

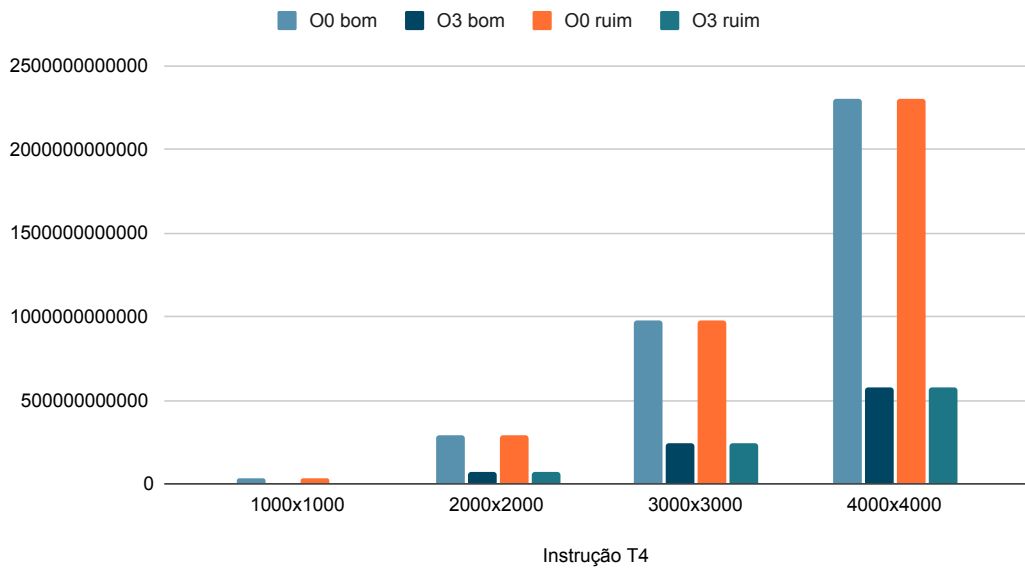
Instruções totais por tamanho da matriz com 2 thread



(b) Instruções totais com 2 threads.

Figura 5 – Número total de instruções executadas em função do tamanho da matriz para 1 e 2 threads.

Instruções totais por tamanho da matriz com 4 thread



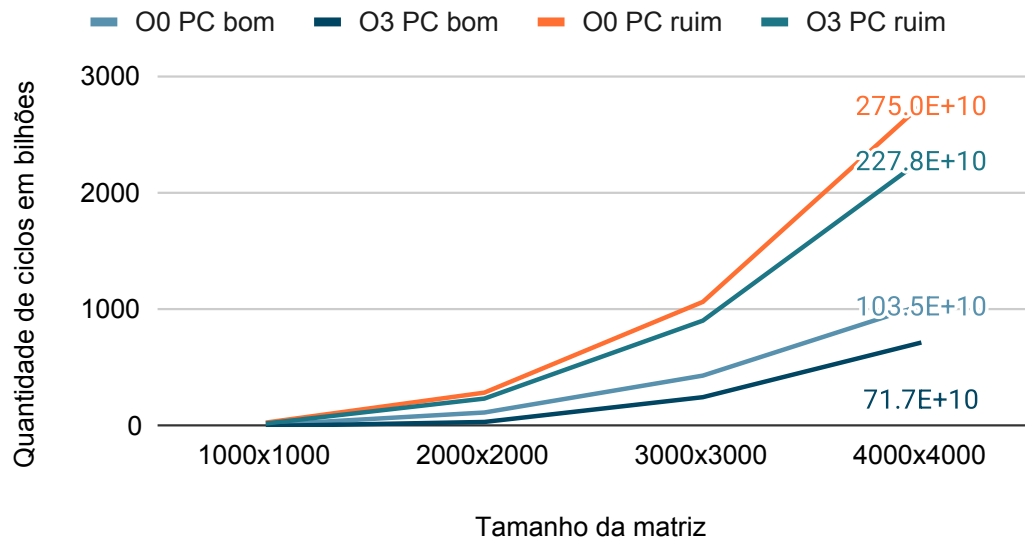
(a) Instruções totais com 4 threads.

Figura 6 – Número total de instruções executadas em função do tamanho da matriz para 4 threads.

4.3 Ciclos Totais

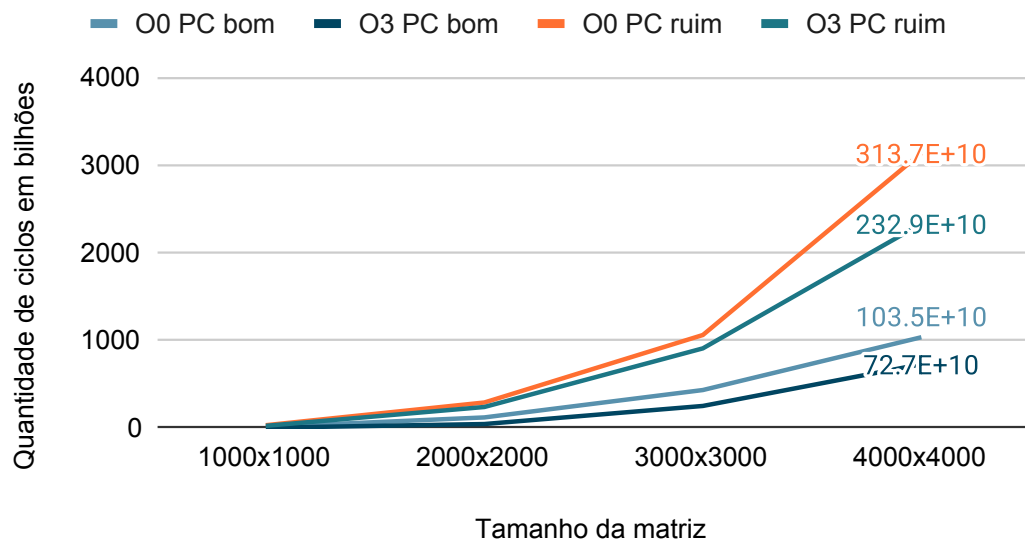
Nesta seção são apresentados os gráficos referentes ao número total de ciclos de clock em função do tamanho da matriz, considerando execuções com 1, 2 e 4 threads.

Ciclos totais por tamanho da matriz em 1 thread



(a) Ciclos totais com 1 thread.

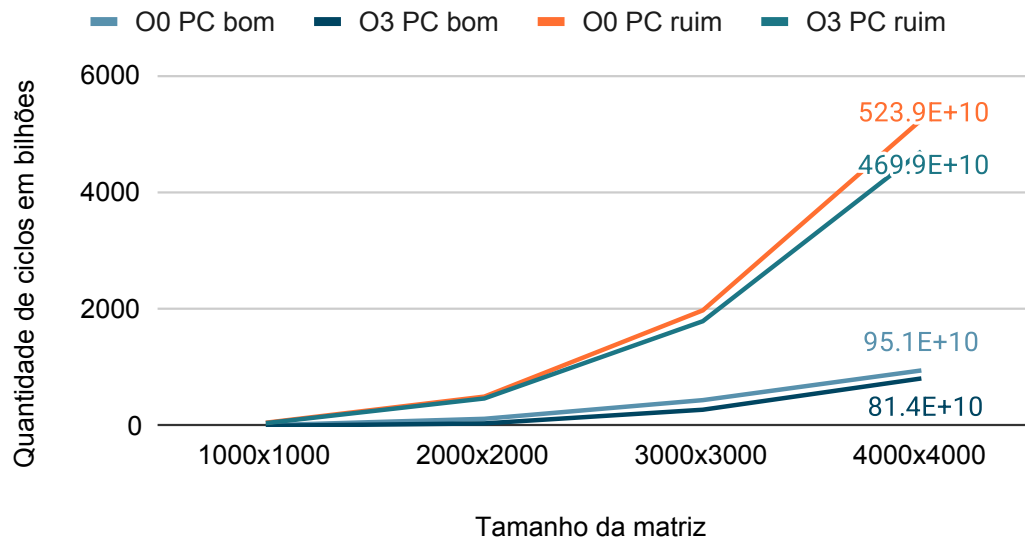
Ciclos totais por tamanho da matriz em 2 threads



(b) Ciclos totais com 2 threads.

Figura 7 – Número total de ciclos de clock em função do tamanho da matriz para 1 e 2 threads.

Ciclos totais por tamanho da matriz em 4 threads



(a) Ciclos totais com 4 threads.

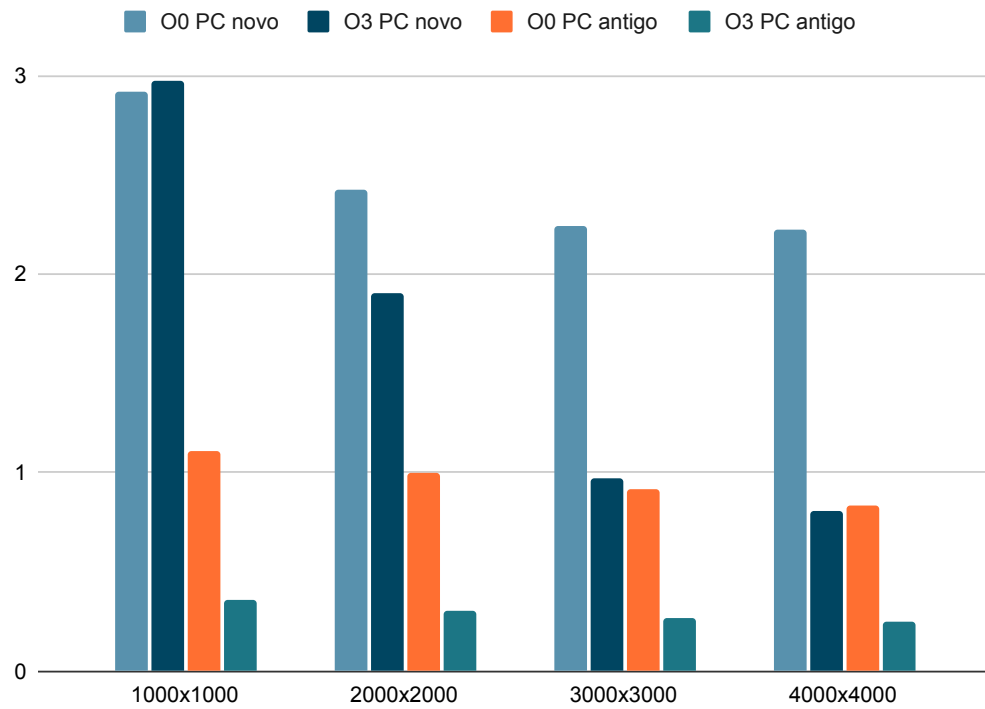
Figura 8 – Número total de ciclos de clock em função do tamanho da matriz para 4 threads.

Conforme apresentado nas Figuras 7 e 8, observa-se que o número total de ciclos cresce com o aumento do tamanho da matriz. A utilização de múltiplas threads reduz o número de ciclos necessários para a execução, evidenciando o ganho de paralelismo proporcionado pela execução concorrente.

4.4 IPC (Instrução por ciclo)

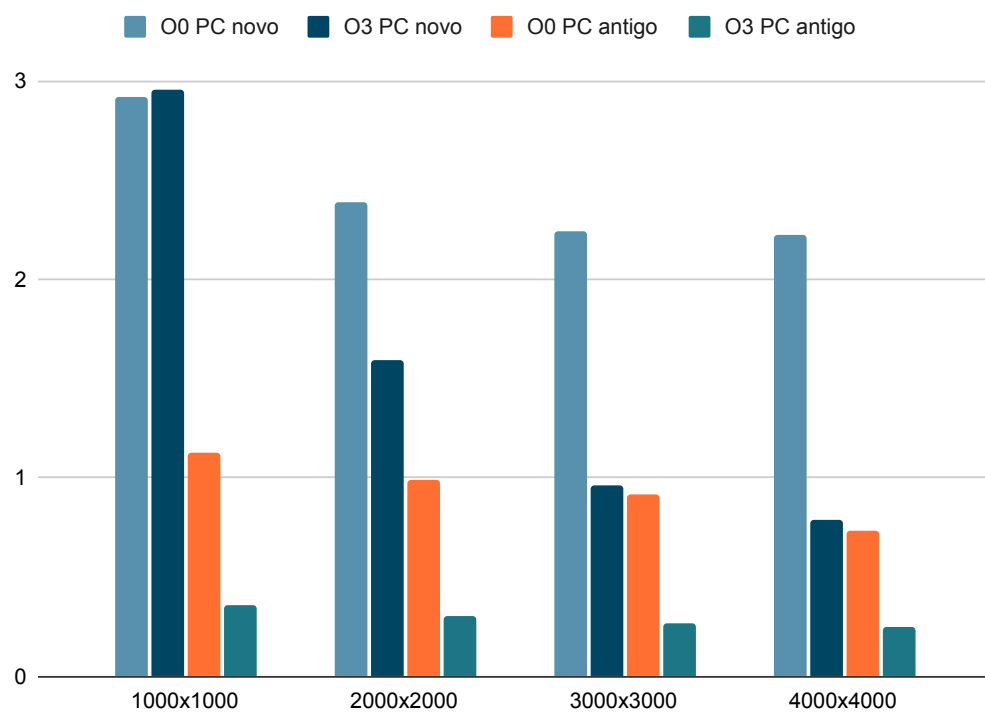
Nesta seção são apresentados os gráficos do IPC médio em função do tamanho da matriz, considerando execuções com 1, 2 e 4 threads.

IPC com 1 thread



(a) IPC médio com 1 thread.

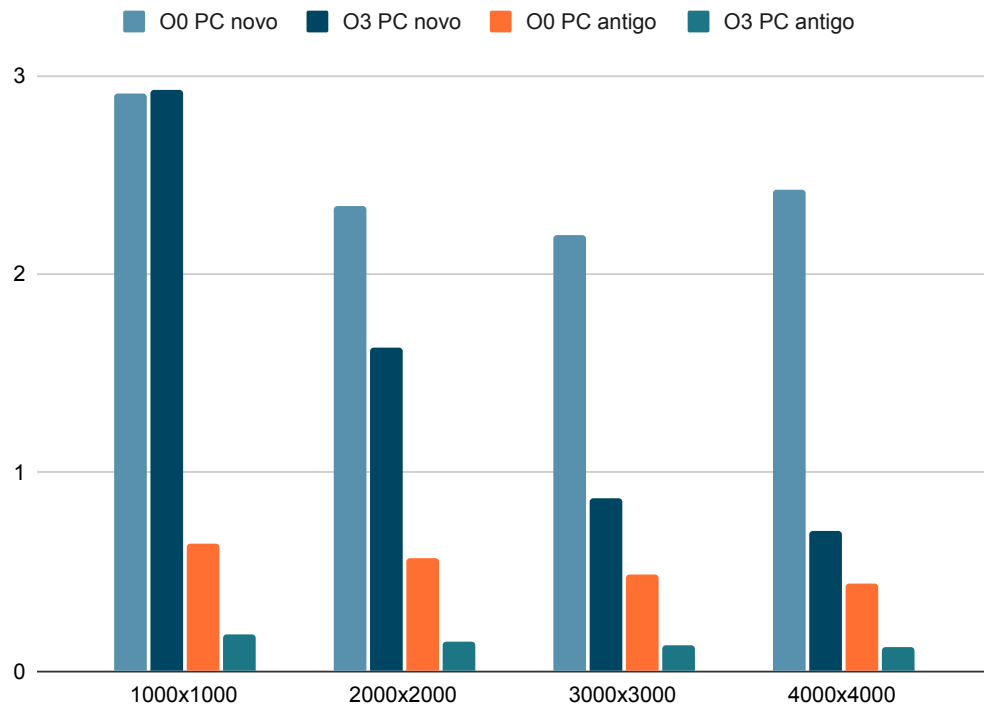
IPC com 2 threads



(b) IPC médio com 2 threads.

Figura 9 – IPC médio em função do tamanho da matriz para execuções com 1 e 2 threads.

IPC com 4 threads



(a) IPC médio com 4 threads.

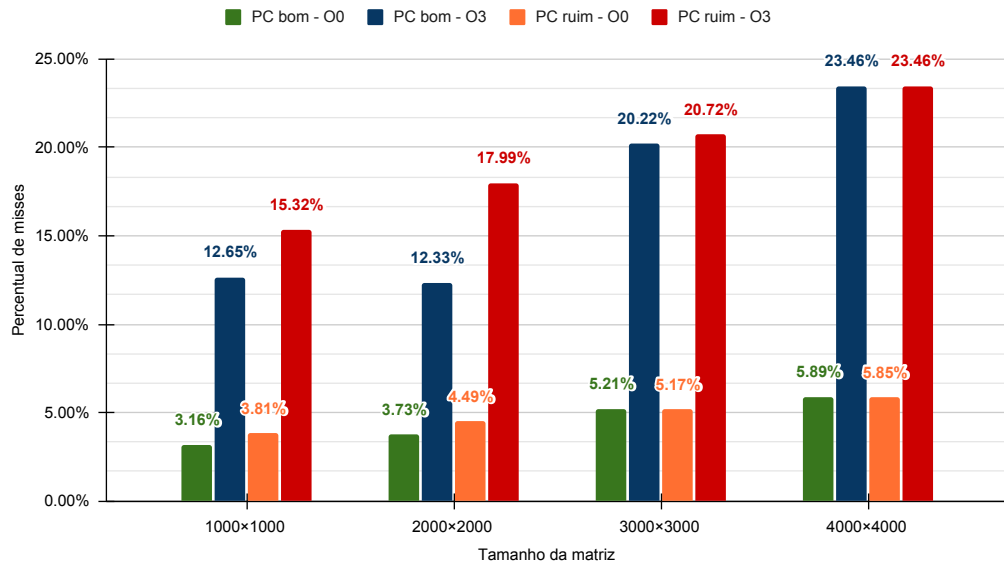
Figura 10 – IPC médio em função do tamanho da matriz para execuções com 4 threads.

Conforme apresentado nas Figuras 9 e 10, observa-se que o IPC médio tende a aumentar com o uso de otimizações e maior paralelismo. Entretanto, para matrizes maiores, o IPC apresenta estabilização ou leve queda, indicando que o desempenho passa a ser limitado principalmente por acessos à memória, e não pela capacidade de execução da CPU.

4.5 Falhas de Cache

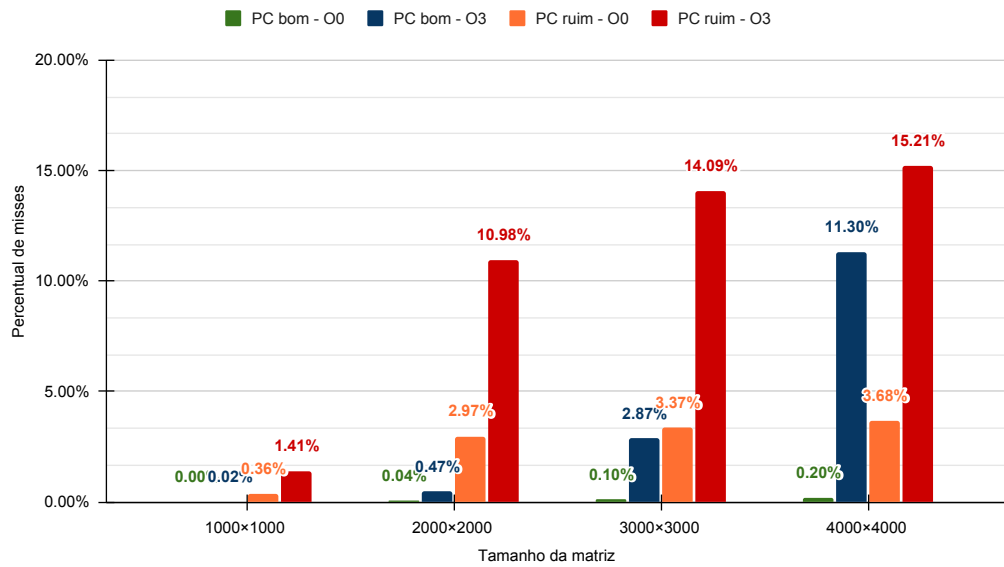
Nesta seção são apresentados os gráficos das falhas de cache normalizadas nos níveis L1 e L2 em função do tamanho da matriz. As falhas foram normalizadas pelo número total de acessos à memória, permitindo uma comparação mais justa entre diferentes configurações.

Percentual de miss em L1 por instrução executada em 1 thread



(a) Falhas de cache L1 normalizadas com 1 thread.

Percentual de miss em L2 por instrução executada em 1 thread



(b) Falhas de cache L2 normalizadas com 1 thread.

Figura 11 – Falhas de cache normalizadas nos níveis L1 e L2 em função do tamanho da matriz para execução com 1 thread.

4.6 Speedup

Nesta seção é apresentado o gráfico de speedup da multiplicação de matrizes em função do tamanho da matriz, considerando os níveis de otimização do compilador -O0 e -O3. O speedup foi calculado como a razão entre o tempo de execução da versão compilada sem otimizações (-O0) e o tempo da versão compilada com otimizações (-O3).

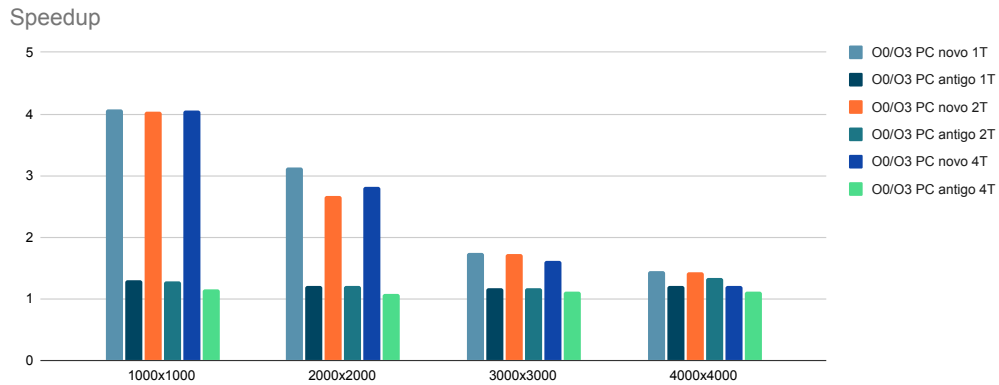
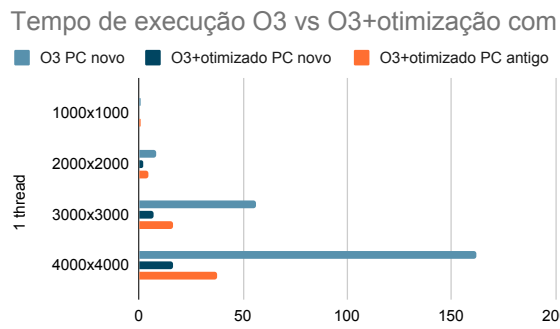


Figura 12 – Speedup da multiplicação de matrizes em função do tamanho da matriz obtido com a otimização -O3 em relação à versão -O0.

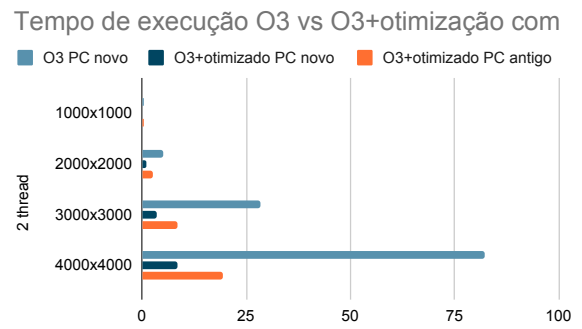
Conforme apresentado na Figura 12, observa-se que o speedup aumenta com o crescimento do tamanho da matriz, indicando que os benefícios das otimizações de compilação tornam-se mais evidentes para cargas de trabalho maiores. Para matrizes de menor dimensão, o ganho é mais limitado, uma vez que o tempo de execução é dominado por overheads que não são significativamente afetados pelas otimizações do compilador.

4.7 Comparação entre -O3 e -O3 com Código Otimizado

Nesta seção é apresentada a comparação entre a versão do código compilada com a flag de otimização -O3 e uma versão adicional contendo otimizações manuais no código-fonte, também compilada com -O3. O objetivo é avaliar o impacto das otimizações manuais no desempenho da multiplicação de matrizes, considerando diferentes números de threads.

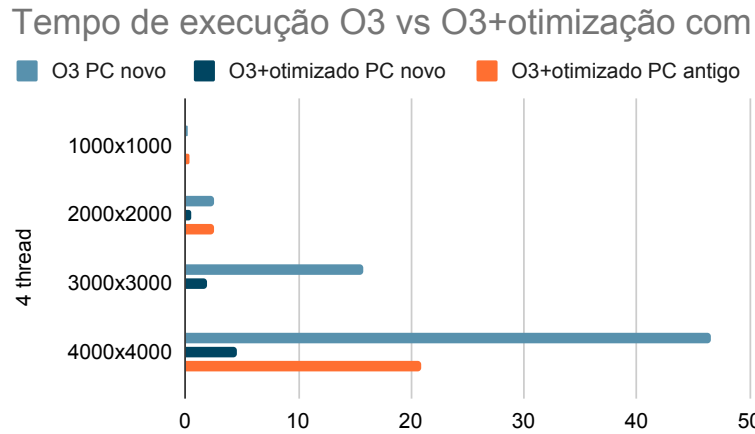


(a) Comparação com 1 thread.



(b) Comparação com 2 threads.

Figura 13 – Comparação entre -O3 e -O3 com código otimizado para diferentes tamanhos de matriz.



(a) Comparação com 4 threads.

Figura 14 – Comparação entre -O3 e -O3 com código otimizado para execuções com 4 threads.

Os resultados indicam que as otimizações manuais no código proporcionam ganhos adicionais de desempenho em relação à versão compilada apenas com -O3, especialmente para matrizes de maior dimensão. Observa-se que o impacto das otimizações torna-se mais evidente à medida que o número de threads aumenta, sugerindo melhor aproveitamento do paralelismo e da hierarquia de memória.

5 Conclusão

Neste trabalho foi realizada uma avaliação detalhada do desempenho da multiplicação de matrizes utilizando métricas de hardware coletadas com a biblioteca PAPI. Os resultados demonstram que as otimizações de compilação e as otimizações manuais no código têm impacto significativo no desempenho, reduzindo o tempo de execução e melhorando o uso dos recursos da CPU.

Além disso, foi possível observar a influência da hierarquia de memória, evidenciada pelo aumento das falhas de cache para matrizes maiores. Como trabalhos futuros, sugere-se a implementação de técnicas como blocking e análise do uso de vetorização explícita.

6 Referências

PAPI. Performance Application Programming Interface. Disponível em: <https://icl.utk.edu/papi/>

ZAMITH, M. Código de multiplicação de matrizes. Disponível em: <https://github.com/mzamith-ufrrj/Computa-o-de-Alto-Desempenho/>

OpenMP Architecture Review Board. OpenMP Application Programming Interface. Disponível em: <https://www.openmp.org/specifications/>

GNU Project. GCC Optimization Options. Disponível em: <https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html>