Desenvolvimento de Código Otimizado

Alyson Matheus Maruyama Nascimento - 8532269

Atividade 6

Blocagem (Aula 9)

Universidade de São Paulo - São Carlos

Programa Escolhido

Assim como na atividade passada, os programa utilizados nessa atividade encontram-se nos arquivos *matmul.c* e *matmul_block.c*. Ambos são responsáveis por realizar uma multiplicação de matrizes, entretanto o primeiro realiza de uma forma convencional enquanto que o segundo utiliza a técnica de "blocagem" para tentar melhorar o desempenho. Para isso, os programas recebem por argumento (*argv*) um único valor inteiro (quantidade de linhas e colunas das matrizes), popula cada uma das matrizes com valores aleatórios sorteados, e realizam a multiplicação. Vale notar que ambas as matrizes a serem multiplicadas possuem mesma dimensão e são quadradas.

Compilação e Execução

Tanto o programa "convencional" de multiplicação de matrizes quanto o "otimizado por blocagem" foram compilados e executados diversas vezes mantendo a mesma arquitetura. Para cada vez, os programas foram executados com o mesmo parâmetro (1000 linhas/colunas por matriz) através do *perf* e as estatísticas retornadas foram salvas na pasta *results/*, entregue juntamente com o arquivo comprimido dessa atividade.

Todos os comandos de compilação e execução utilizados encontram-se dentro do arquivo *compile_exec.sh*, que corresponde ao script criado para simplificar o processo de compilação e execução para essa atividade e que também foi entregue no arquivo comprimido junto à esse documento.

Resultados

Como citado anteriormente, as estatísticas retornadas pelo comando *perf stat* encontram-se na pasta *results/*. Os gráficos abaixo resumem as informações coletadas e demonstra as diferenças de tempo de execução, *cache-hits* e *cache-misses* entre cada uma das compilações.

Para visualizar as saídas em formato bruto do comando *perf*, basta acessar o diretório *results* no arquivo compactado.

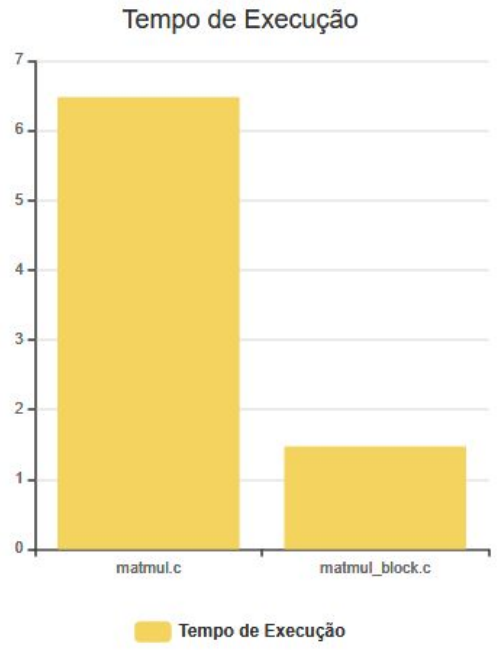

Figura 1: diferenças de tempo de execução

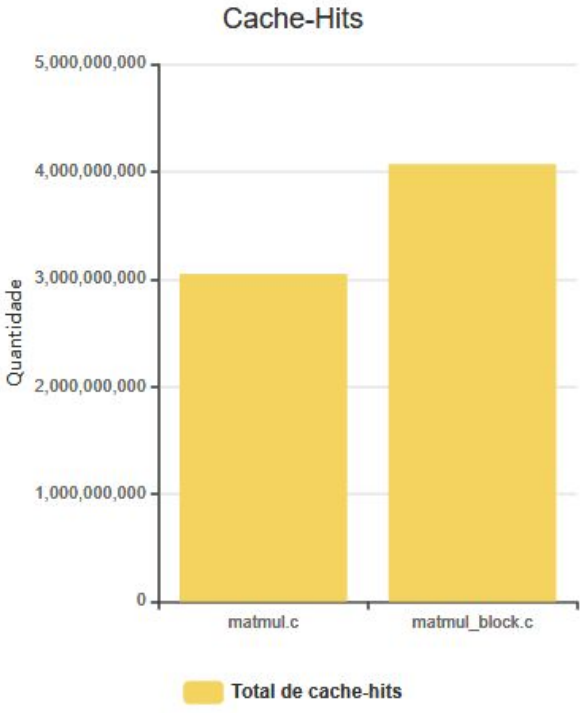

Figura 2: total de cache hits por programa

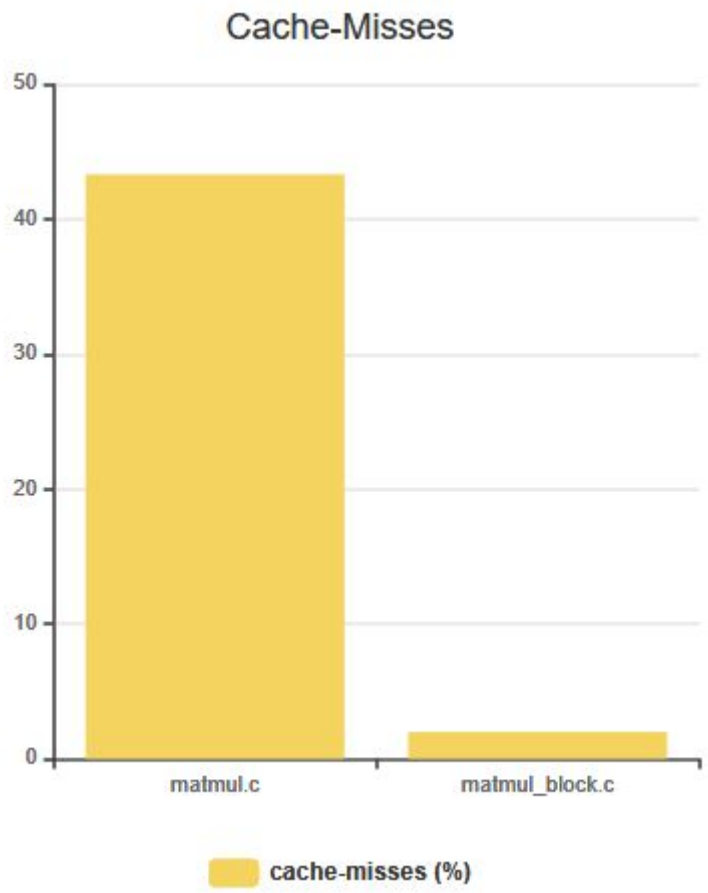

Figura 3: porcentagem de cache misses

Dentre os dados coletados, podemos notar que a técnica de blocagem obteve um grande ganho em todas as métricas coletadas (tempo de execução, *cache-hits* e porcentagem de *cache-misses*). Como o programa em questão possui grande uso de matrizes e o tamanho das matrizes são relativamente grandes (1000 x 1000) o uso da blocagem apresenta um grande ganho de desempenho.

O tempo de redução foi reduzido para praticamente 25% do tempo do código original, ao mesmo tempo que a quantidade de *cache-hits* aumentou substancialmente, consequentemente causando a redução no percentual de *cache_misses*. Esses dois últimos pontos se devem ao fato de a técnica de blocagem ser capaz de aproveitar de maneira eficaz os valores da matriz carregados na cache, reduzindo a quantidade de vezes que estes são retirados e depois inseridos de volta à cache quando ocorrer um *miss*.

É importante notar, entretanto, que conforme diminuirmos o tamanho

das matrizes, a blocagem se torna menos eficiente, ao ponto que se determinarmos matrizes de tamanhos menores que o tamanho do bloco (*BLOCK_SIZE*), provavelmente não teremos ganhos significativos na execução do programa.