Cálculo de determinante de matriz NxN utilizando o teorema de Laplace

O método de Laplace

O método de Laplace calcula o determinante de uma matriz a partir do cálculo da determinante de matriz menores, chamadas de cofatores. Assim, temos para uma matriz B:

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Aplicando o teorema de Laplace na primeira linha, temos:

$$|B| = 1 \cdot \begin{vmatrix} 5 & 6 \\ 8 & 9 \end{vmatrix} - 2 \cdot \begin{vmatrix} 4 & 6 \\ 7 & 9 \end{vmatrix} + 3 \cdot \begin{vmatrix} 4 & 5 \\ 7 & 8 \end{vmatrix}$$

$$= 1 \cdot (-3) - 2 \cdot (-6) + 3 \cdot (-3) = 0$$

De forma que a matriz do cofactor é formado pelos elementos restantes quando se remove a linha e a coluna do termo a ser calculado.

O cálculo

Para esta implementação, utilizamos o cálculo do cofator de forma recursiva, reduzindo o cálculo até o caso base, ou seja, N = 1, onde N é o tamanho da matriz.

Assim, este código resume-se em:

1) Confere se N=1, se sim, retorna matriz[0][0]
2) Caso contrário:
    a) Monta a matriz de tamanho N-1 correspondente ao valor que está sendo calculado
    b) Calcula o cofator de cada termo da primeira linha da matriz N-1

Otimização utilizando o OpenMPI

Para tornar o código mais otimizado, distribuem-se o cálculo dos cofatores igualmente por todos os cores disponíveis na máquina, contando que o core 0 designa o que cada um dos seus workers (todos os cores com id > 0) fará.

Desta forma, para uma matriz de tamanho N = 8 e 4 cores disponíveis, teremos:

Id	Jobs	Posições a serem calculadas
0	0	-
1	2	0, 1
2	3	2, 3, 4
3	3	5, 6, 7

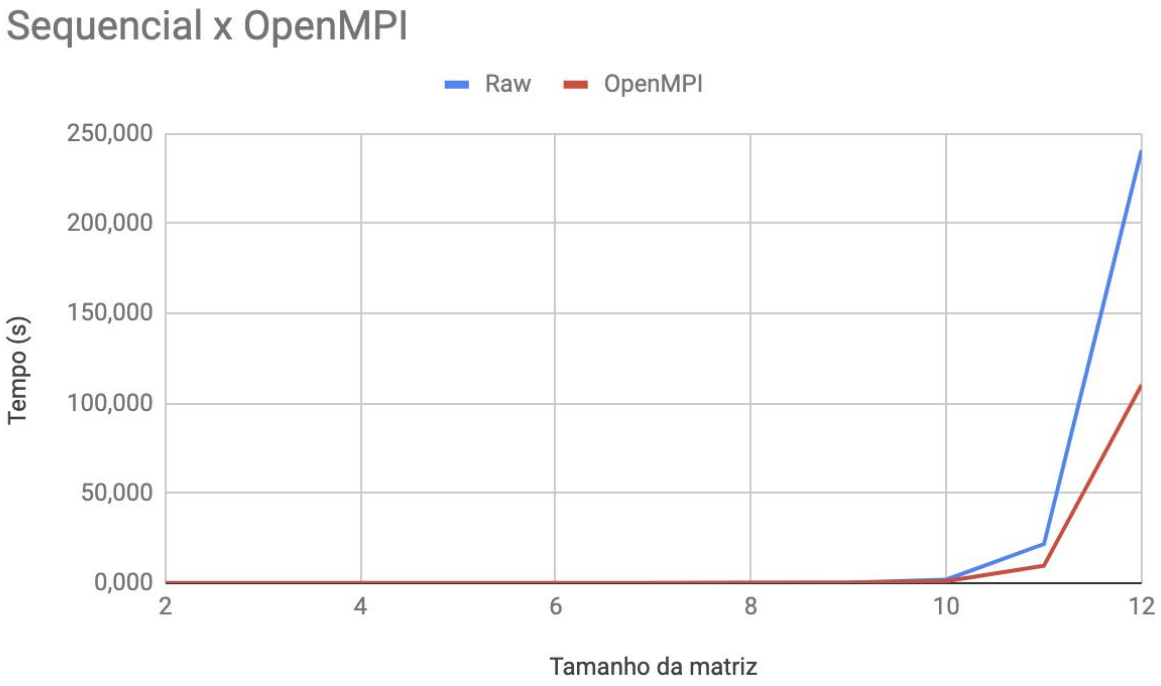
A lógica principal baseia-se em, dado uma matriz M:

1) Efetua-se o cálculo de quantos jobs cada coda worker fará e quais as posições que devem ser calculadas por ele;
2) O master aloca e preenche a matriz M;
3) Ao término do passo acima, o Master envia cada linha da matriz para cada worker pela instrução MPI_Broadcast. Cada worker recebe o broadcast pela mesma instrução, de forma que só começam o cálculo quando terminam de receber todas as linhas da matriz;
4) Cada worker faz o cálculo de todos os jobs que lhes foi designado, armazenando em uma variável local chamada det_local;
5) Em seguida, cada worker envia sua váriavel det_local para o processo master via instrução MPI_Reduce, fazendo a soma de cada det_local na váriavel det_total, que representa o valor final do determinante da matriz M;

Comparação Sequencial X Distribuída

	2	3	4	5	6	7	8	9	10	11	12
Raw (s)	0,004	0,004	0,004	0,004	0,004	0,007	0,032	0,211	1,958	21,739	240,800
OpenMPI (s)	0,117	0,116	0,114	0,118	0,116	0,121	0,164	0,216	1,074	9,634	110,360

Tempo de execução, em segundos, para cada tipo de implementação para um tamanho de matriz.



Como pode ser observado no gráfico acima, para matrizes relativamente pequenas, o OpenMPI é menos eficiente, já que o Overhead de mensagens trocados pelos processos passam a demorar mais que o próprio cálculo da determinante. Porém, a partir de matriz 10x10, a distribuição do cálculo passa a ser vantajosa, chegando a um ganho de 55% no tempo para uma matriz 12x12.