Trabalho de MPI - Programação Distribuída e Paralela

Nome: Igor Martins Silva (00333069)
Data:07/01/2025
24/2 - INF01008
Prof: Arthur Lorezzon

Introdução

O trabalho consiste em simulações de multiplicação de matriz utilizando a biblioteca de MPI, usando as diretivas de comunicação bloqueantes, não bloqueantes e coletivas.

A biblioteca MPI é uma biblioteca responsável por criar comunicação entre processos e paralelizar códigos em ambientes distribuídos. Ela é extremamente útil, portável e fácil de ser implementada e utilizada. Questões como multiplicação de matrizes, que é extremamente utilizável na computação pode ser altamente paralelizável.

No trabalho iremos utilizar o openMP para mensurar o impacto em diferentes multiplicações de matrizes quadradas e analisar qual a melhor diretiva para a nossa implementação.

Metodologia

Minha metodologia consiste em 3 principais abordagens:

- Mudança no tamanho do bloco da matriz, com valores de 2.048, 8.192 e 16.384.
- Mudança no número de máquinas, sendo feito com 2,3 e 4 máquinas simultaneamente.
- Mudança no número de tasks(processos), de 20, 40, 80, 120 e 160.

Os experimentos foram realizados nas máquinas “hype” do Parque Computacional de Alto Desempenho(PCAD) do instituto de informática da UFRGS. As configurações das máquinas utilizadas são as seguintes:

Nome	Partição	CPU	RAM	Acelerador	Disco	Placa-mãe
hype1	hype	2 x Intel(R) Xeon(R) E5-2650 v3, 2.30 GHz, 40 threads, 20 cores	128 GB DDR4		558.9 GB HDD	HP ProLiant DL380 Gen9
hype[2,3]	hype	2 x Intel(R) Xeon(R) E5-2650 v3, 2.30 GHz, 40 threads, 20 cores	128 GB DDR4		558.9 GB HDD	HP ProLiant XL170r Gen9
hype5	hype	2 x Intel(R) Xeon(R) E5-2650 v3, 2.30 GHz, 40 threads, 20 cores	128 GB DDR4	2 x NVIDIA Tesla K80	558.9 GB HDD	HP ProLiant XL190r Gen9

Fonte: Parque Computacional de Alto Desempenho (PCAD)

E as simulações basicamente consistem em fazer as matrizes 2.048, 8.192 e 16.384:

- 2 máquinas e 40 tasks
- 2 máquinas e 80 tasks
- 3 máquinas com 120 tasks
- 4 máquinas com 160 tasks

E teve um teste que realizei exclusivamente com a comunicação não bloqueante com sempre 20 tasks e a matriz de tamanho 2048 mas mudando o número de máquinas para ver o quanto impacto dá a quantidade de computadores se comunicando.

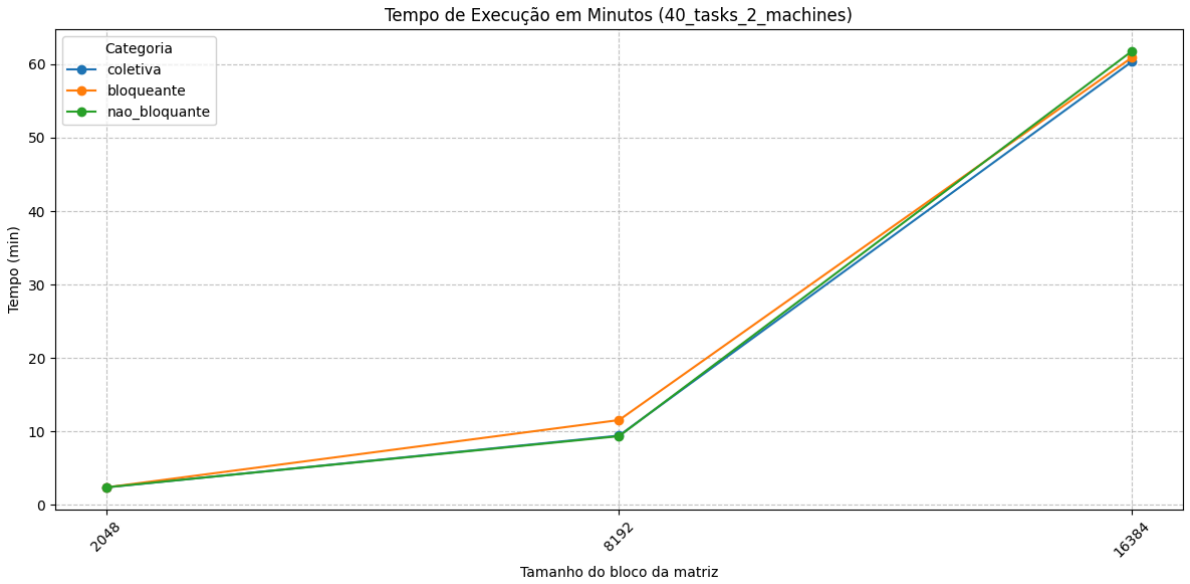
Resultados:

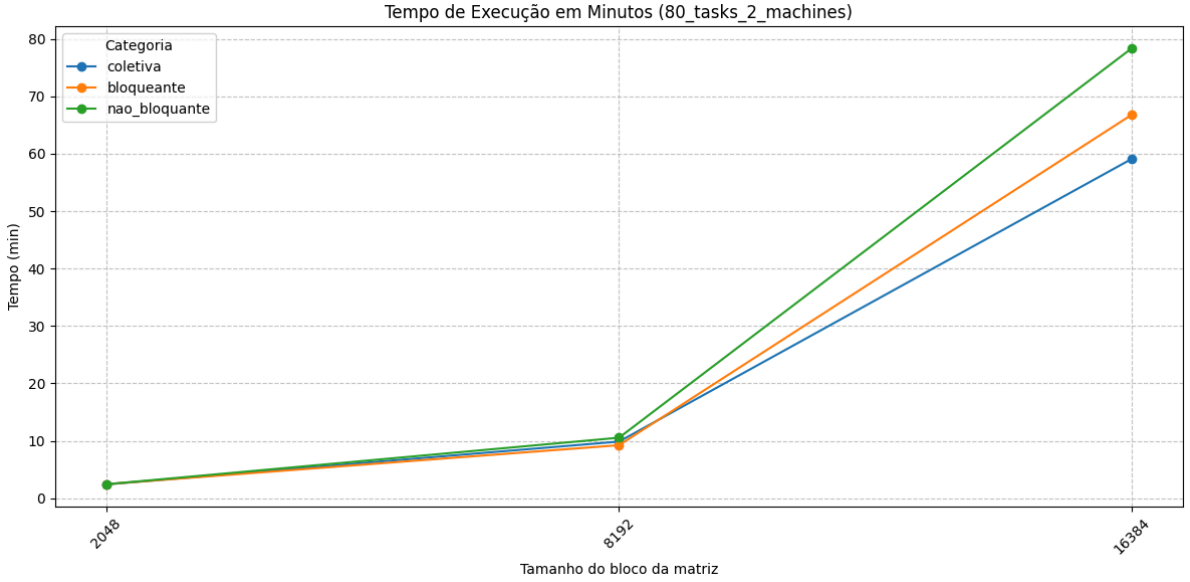
Tempo de execução

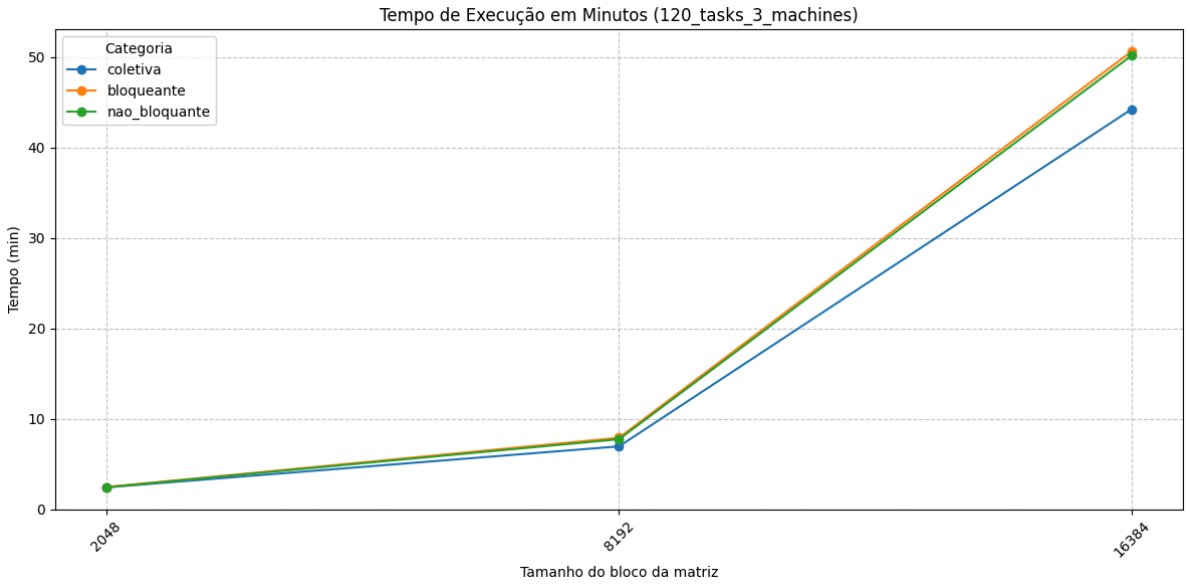
Como pode-se notar nos gráficos abaixo, a comunicação não bloqueante atingiu resultados piores em dois contextos: 80 tasks em 2 máquinas e 160 tasks em 4 máquinas, sendo que nesse primeiro caso chegou a ser 20 minutos a mais que a comunicação coletiva. Nos outros experimentos ele foi extremamente parecido com com os resultados das outras categorias.

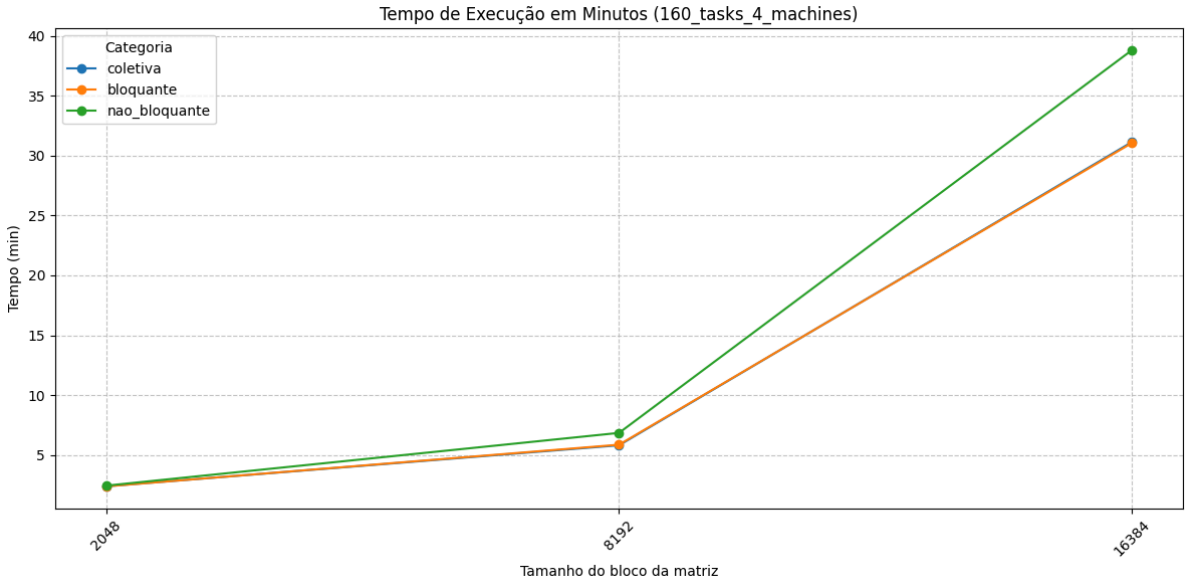
Outra tendência que se notou é que nos casos de 40 e 80 tasks com 2 máquinas foram os casos que mais demoraram para executar, sendo que no caso das 80 tasks levou quase 80 minutos, tempo superior ao caso de 160 tasks e 4 maquinas, que levaram no máximo 40 minutos.

Possivelmente essa demora a mais tem haver com a comunicação não bloqueante ter um rotina de comunicação específica no código chamada de *MPI_Wait* que faz com bloqueie uma rotina específica até o fim da comunicação, fazendo que ele demore mais tempo para terminar de executar. se tornando **um gargalo** no código, que poderia ser trocado por *MPI_Test* que testa se a operação de send e receive terminaram, mas não fica esperando isso ser finalizado.


Tempo de Execução em Minutos (40_tasks_2_machines)
Categoria
coletiva
bloqueante
nao_bloquante
Tempo (min)
0
10
20
30
40
50
60
2048
8192
16384
Tamanho do bloco da matriz


Tempo de Execução em Minutos (80_tasks_2_machines)
Categoria
coletiva
bloqueante
nao_bloquante
Tempo (min)
0
10
20
30
40
50
60
70
80
2048
8192
16384
Tamanho do bloco da matriz

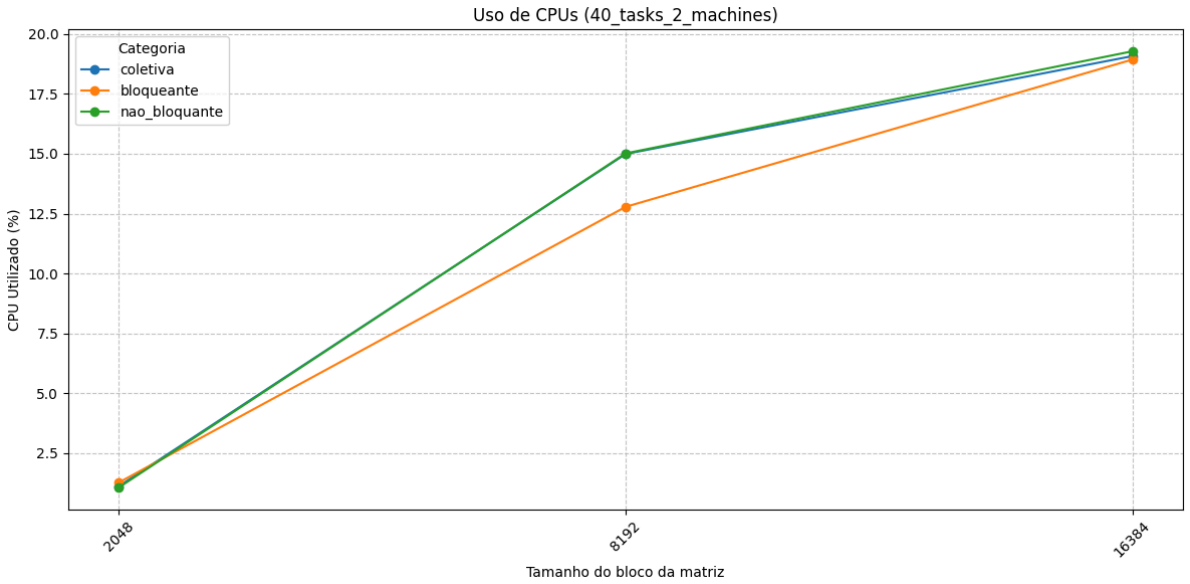


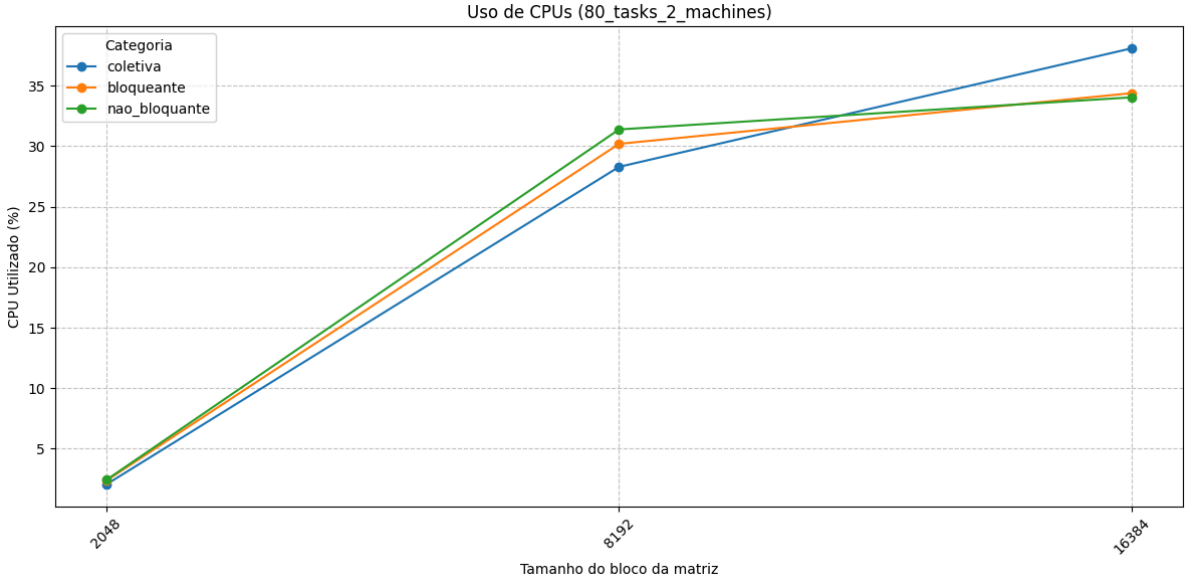


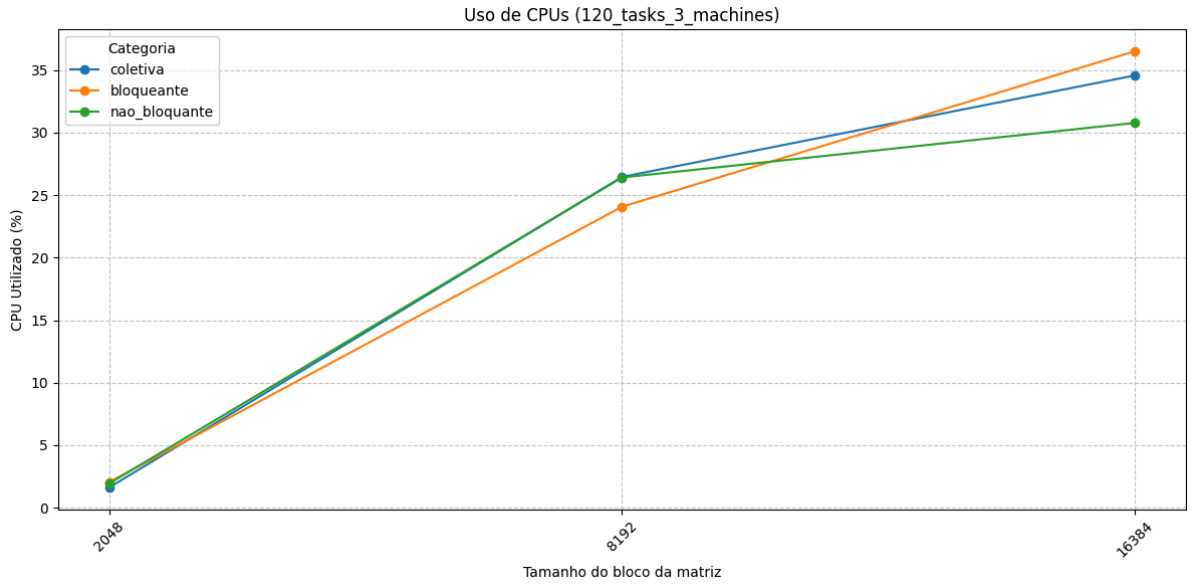
Uso de CPUs

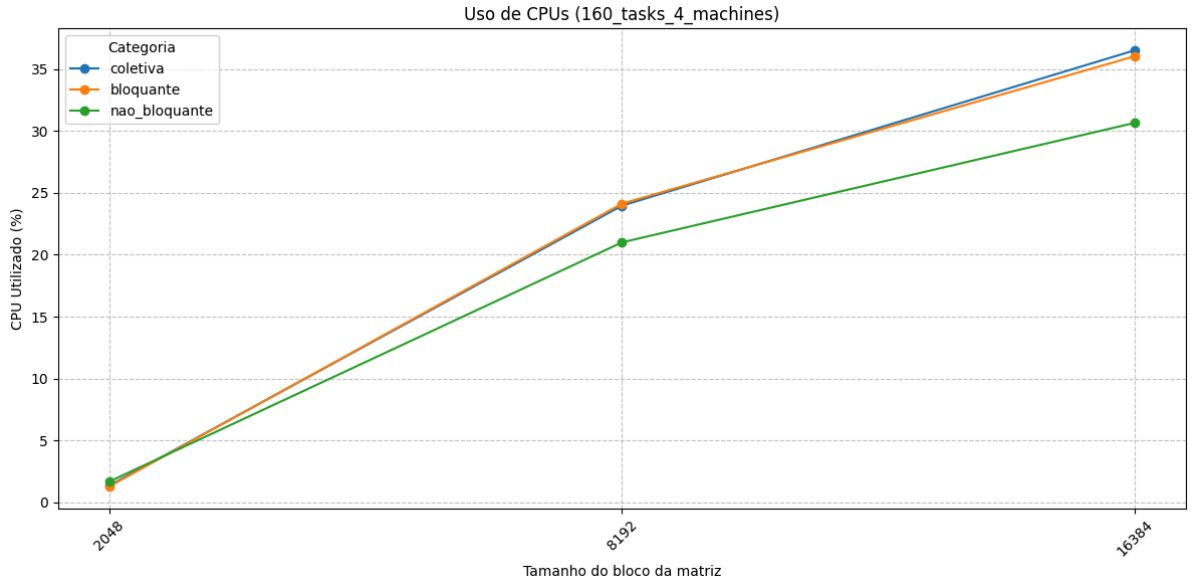
Sobre o uso da CPU podemos afirmar que a comunicação coletiva usa um pouco mais que os outros, mas com pouca ou nenhuma diferença quase. Em blocos de matrizes até 8192 o padrão se mantém do uso da CPU, com resultados bem parecidos, mas nos blocos com 16384 e nos casos de 80 e 120 tasks há troca de qual comunicação usa mais a CPU. Sendo que no primeiro caso a coletiva usa, em média, 38% de CPU, enquanto a comunicação bloqueante que estava embaixo das outras no caso das 120 tasks, tem um uso maior que as outras com 36% de uso no bloco maior.

Possivelmente esse uso maior da coletiva se dá por causa que necessita de maior organização para fazer uma comunicação coletiva, então há mais custo computacional envolvido.


Uso de CPUs (40_tasks_2_machines)
Categoria
coletiva
bloqueante
nao_bloquante
CPU Utilizado (%)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
2048
8192
16384
Tamanho do bloco da matriz

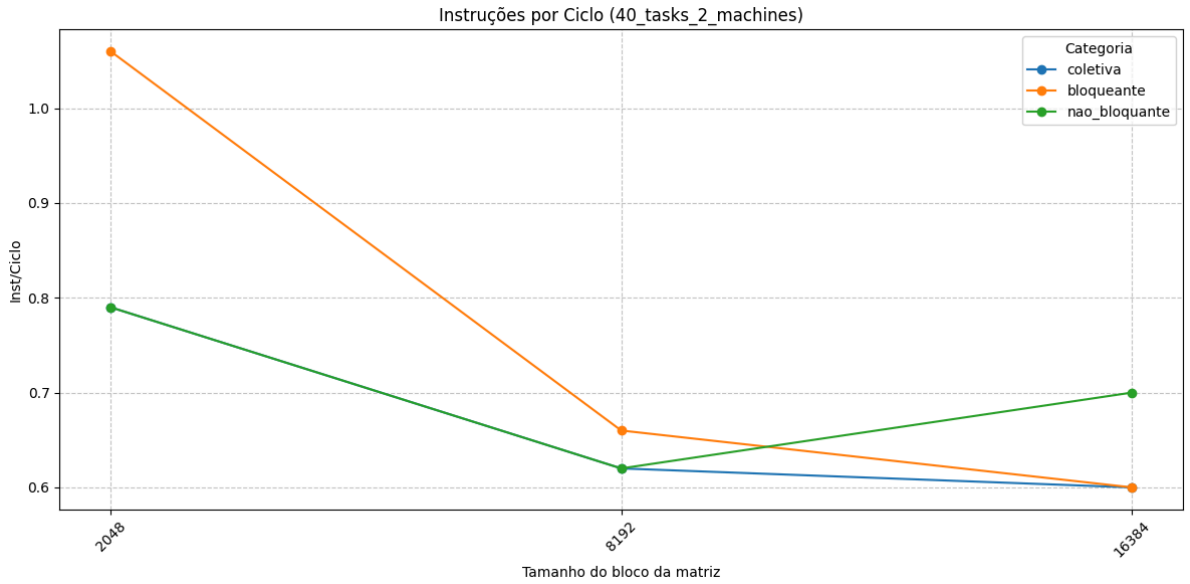

Uso de CPUs (80_tasks_2_machines)
Categoria
coletiva
bloqueante
nao_bloquante
CPU Utilizado (%)
35
30
25
20
15
10
5
2048
8192
16384
Tamanho do bloco da matriz

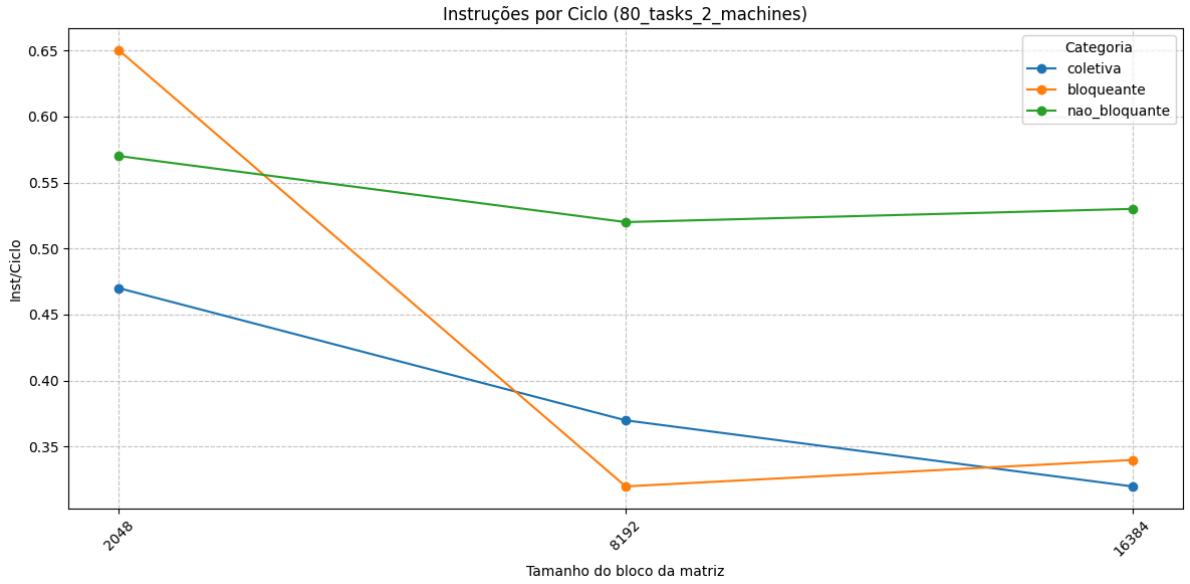


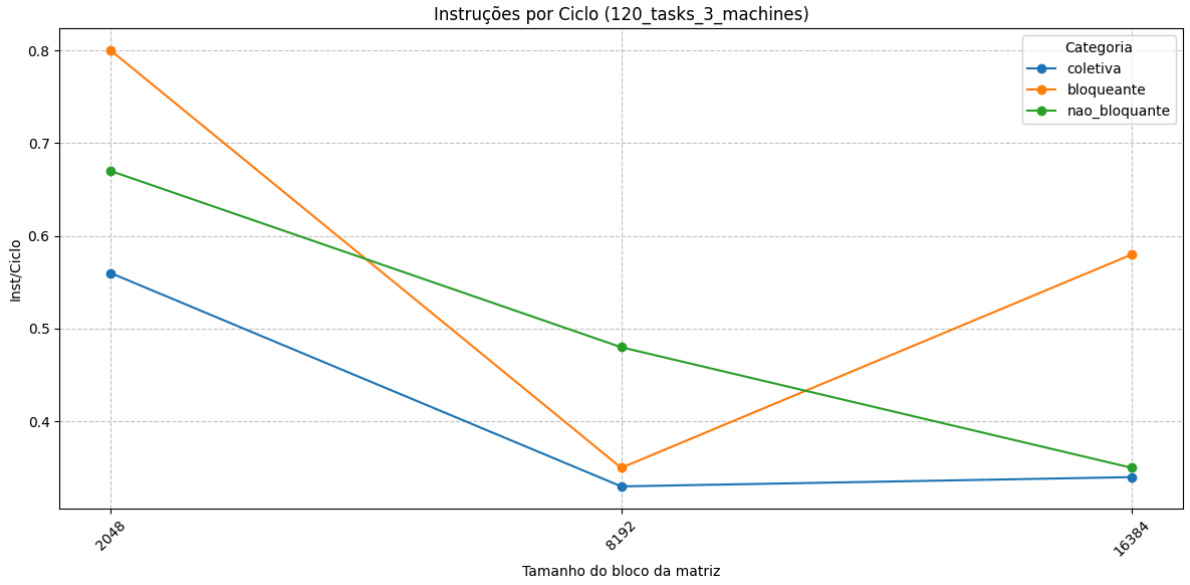


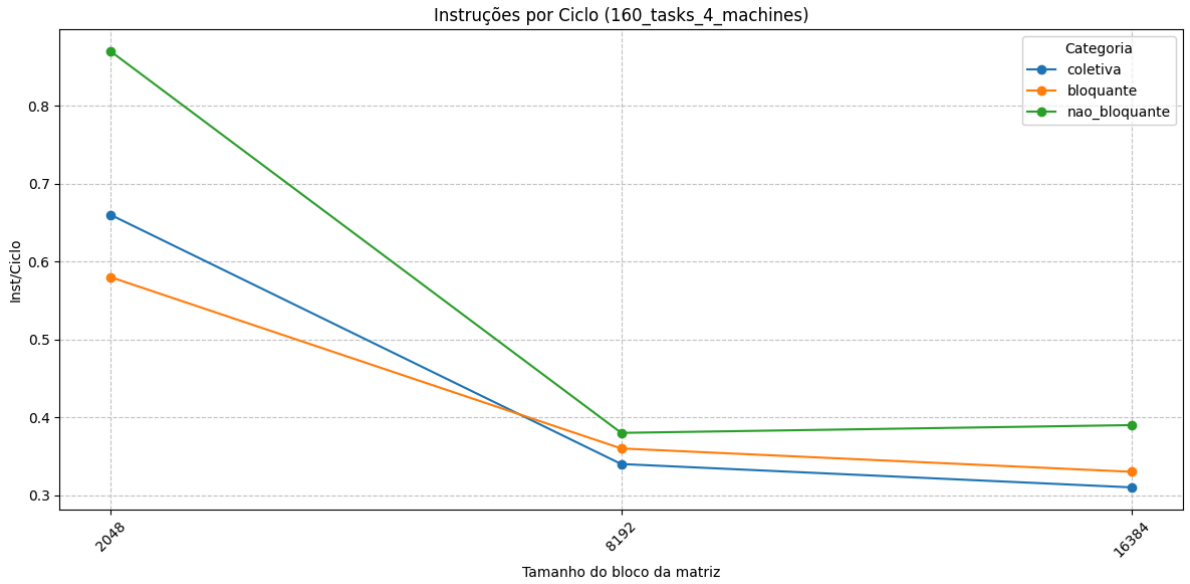
Instruções Por ciclo

Já neste teste, podemos observar que em matrizes de bloco pequeno temos bem mais instruções por ciclo, mas depois todas caem, exceto no caso das 120 task do não bloqueante, que teve um aumento quando aumento o tamanho do bloco. Em casos de blocos menores a comunicação bloqueante se prova mais eficiente por ter mais instruções por ciclo mas em blocos maiores a comunicação não bloqueante se provou melhor nesse aspecto.


Instruções por Ciclo (40_tasks_2_machines)
Categoria
coletiva
bloqueante
nao_bloquante
Inst/Ciclo
1.0
0.9
0.8
0.7
0.6
2048
8192
16384
Tamanho do bloco da matriz


Instruções por Ciclo (80_tasks_2_machines)
Categoria
coletiva
bloqueante
nao_bloquante
Inst/Ciclo
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2048
8192
16384
Tamanho do bloco da matriz

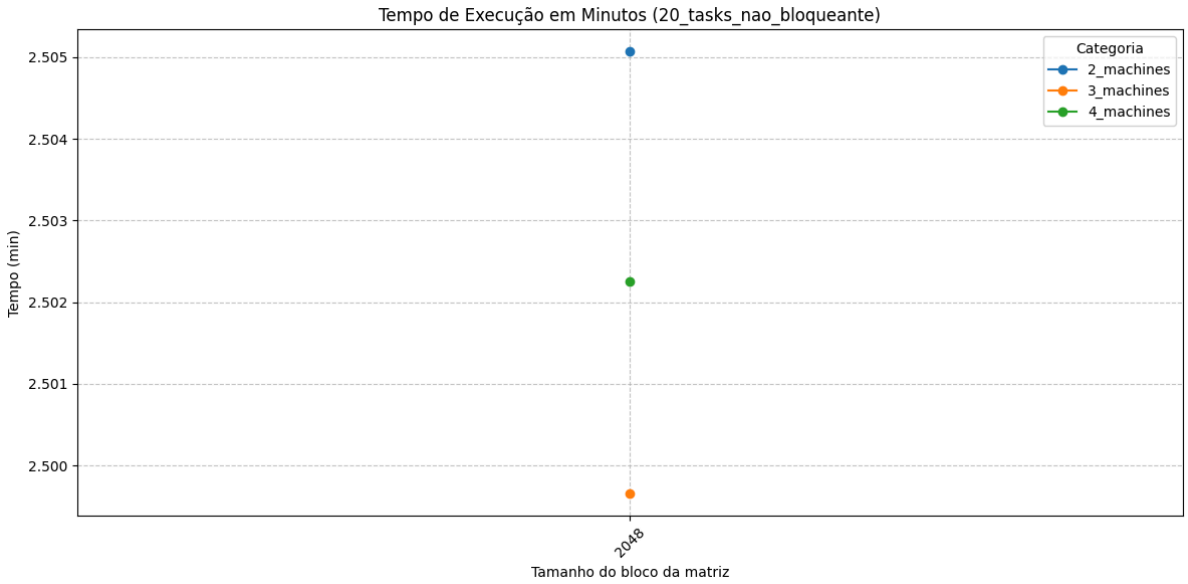

Instruções por Ciclo (120_tasks_3_machines)
Categoria
coletiva
bloqueante
nao_bloquante
Inst/Ciclo
0.8
0.7
0.6
0.5
0.4
2048
8192
16384
Tamanho do bloco da matriz

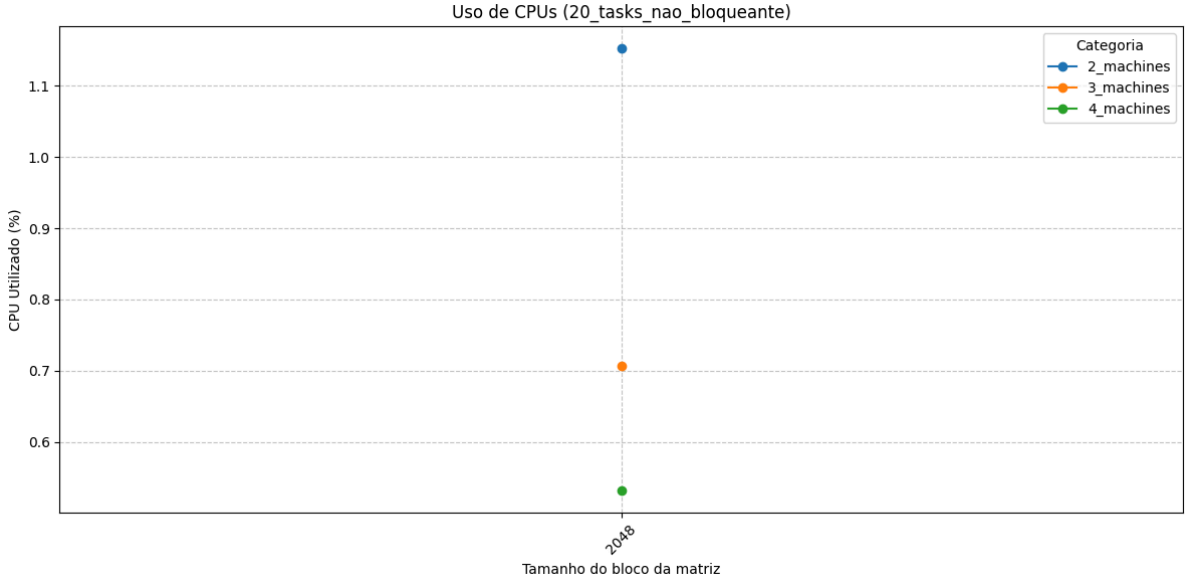


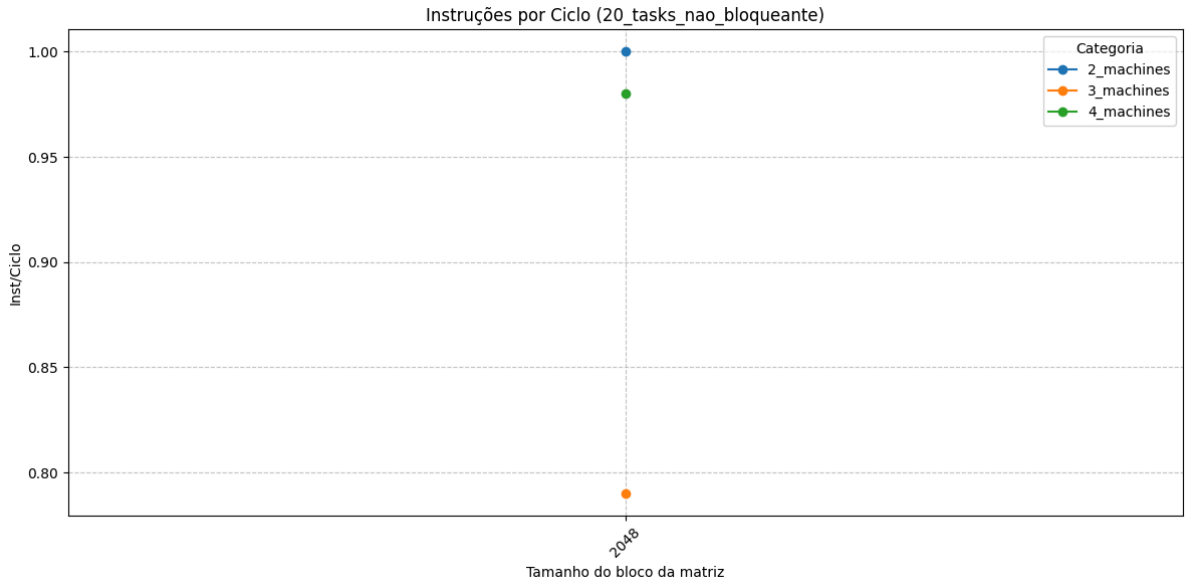
Teste pelo número de máquinas

O último teste realizado foi para ver se o número de máquinas impacta na performance da comunicação, mantendo o tamanho do bloco e de tasks e o mesmo tipo de comunicação. Também foi feito o experimento com 20 tasks e deixar o bloco em 16 mil, mas passou de uma hora e meia o experimento, sendo então que foi trocado para um bloco menor para mensurar o impacto da quantidade de computadores se comunicando. Foi escolhido a não bloqueante pois ela em tese deve ser melhor que as outras.

Como pode-se observar pelos gráficos abaixo, o tempo de execução foi praticamente o mesmo em todos os casos. Já o uso de CPU teve um certa diferença com o uso maior com duas máquinas e menor com 4 máquinas, mostrando que o custo computacional se dilui entre as máquinas, sendo que cada máquina a mais diminui em média em 25% o uso da CPU. E as instruções por ciclo temos que duas máquinas fazem cerca 0.2 instruções a mais que quando se tem 3 máquinas, mas tem resultado parecido com 4 máquinas.


Tempo de Execução em Minutos (20_tasks_nao_bloqueante)
Tempo (min)
2.505
2.504
2.503
2.502
2.501
2.500
2048
Tamanho do bloco da matriz
Categoria
2_machines
3_machines
4_machines


Uso de CPUs (20_tasks_nao_bloqueante)
CPU Utilizado (%)
1.1
1.0
0.9
0.8
0.7
0.6
2048
Tamanho do bloco da matriz
Categoria
2_machines
3_machines
4_machines


Instruções por Ciclo (20_tasks_nao_bloqueante)
Inst/Ciclo
1.00
0.95
0.90
0.85
0.80
2048
Tamanho do bloco da matriz
Categoria
2_machines
3_machines
4_machines

Conclusão

Observando os resultados obtidos, temos que em termos gerais a comunicação coletiva foi mais rápida cerca de até 20 minutos que a não bloqueante, mas com maior custo computacional. Percebe-se também que todas se beneficiam do maior número de máquinas, mesmo com o aumento do número de tasks e do tamanho de bloco. Para blocos pequenos a comunicação não bloqueante é uma boa opção, mas para tamanhos grandes ela é pior. E a bloqueante se mostrou um meio termo entre tempo e uso de CPU.

Referências

Comunicação Não Bloqueante. Blog do Kuniga
https://kuniga.wordpress.com/2010/12/10/comunicacao-nao-bloqueante/ Acessado 08/01/2025.

MPI - Message Passing Interface. Bate Byte
https://www.batebyte.pr.gov.br/Pagina/MPI-Message-Passing-Interface Acessado 08/01/2025

GWDG – Kurs Parallel Programming with MPI MPI Collective Operations
https://moodle.ufrgs.br/pluginfile.php/7124088/mod_resource/content/1/pchpc_mpi_collective_slides.pdf

MPI Non-blocking Communication
https://moodle.ufrgs.br/pluginfile.php/7124087/mod_resource/content/1/NonblockingCommunication.pdf

Parque Computacional de Alto Desempenho (PCAD)
https://gppd-hpc.inf.ufrgs.br/#orgaee39cb