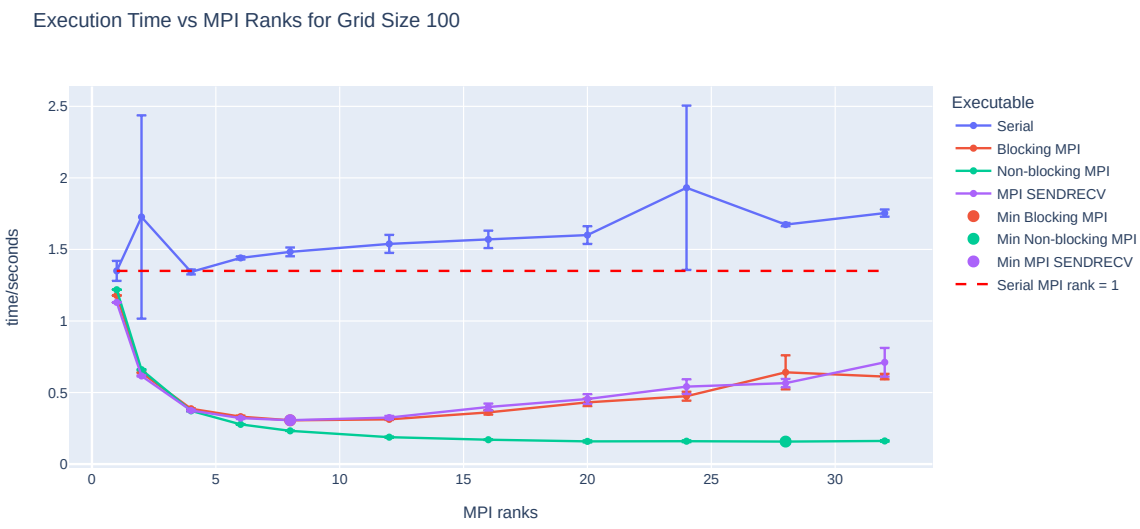
> What is the scaling limit, i.e. the number of MPI ranks corresponding to the maximum performance, for each grid size and for each MPI point-to-point call?

I ran Fortran programs for grid sizes of 100, 200, 300, and MPI ranks in the range (1 2 4 6 8 12 16 20 24 28 32). I ran them three times to find the standard deviation, which is represented as error bars on the graphs. In the graph below for a grid size of 100, there are some noticeable outlier points for Serial 2 and 24, but it's important to consider the scale on the Y-axis. Notably, even though serial is defined as running on 1 CPU, out of curiosity, I decided to also run the program with input data for other MPI ranks. Logically, one would expect to see a constant value ±, but as the graphs show, this is not the case. The likely reasons are scheduling overhead and resource sharing. This is also one of the motivations for running the programs multiple times, to check the reproducibility of the timing results.

For grid sizes of 200 and 300, the minimum occurs with Non-blocking MPI at the maximum value of 32.

Generally speaking, a border point cannot be considered a global minimum, but considering that for other parallel computations, the global minimums occur at higher time values, this does not affect the answer to "which version of MPI gives the best result." Therefore, 32 is the scaling limit only within the chosen range of MPI ranks.

For a grid size of 100, the scaling limit is at MPI rank 28, but the time saving is not as significant compared to 8, which is where the scaling limits of other MPI versions lie. It is also worth noting that the increase in time with increasing MPI ranks is associated with increased synchronization overhead. The same can be said for other grid size values (only instead of 8 -- 16).



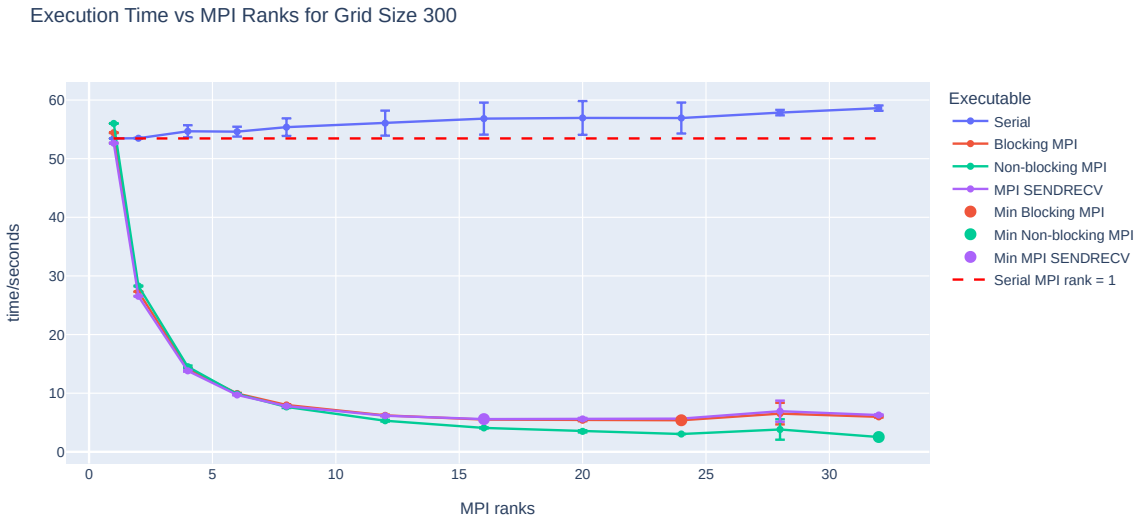


Table 1: scaling limit corresponding to grid size

Grid Size	Min time, s	Scaling limit	Executable	Serial time/Min time
100	0.157	28	Non-blocking MPI	8.6
200	0.768	32	Non-blocking MPI	17.7
300	2.522	32	Non-blocking MPI	21.2

Table 2: scaling limit corresponding to MPI rank, grid size = 100

Executable	Grid Size	MPI Ranks	Execution Time Mean, s
MPI SENDRECV	100	1	1.130
MPI SENDRECV	100	2	0.616
Non-blocking MPI	100	4	0.372
Non-blocking MPI	100	6	0.278
Non-blocking MPI	100	8	0.232
Non-blocking MPI	100	12	0.188
Non-blocking MPI	100	16	0.170
Non-blocking MPI	100	20	0.159
Non-blocking MPI	100	24	0.160
Non-blocking MPI	100	28	0.157
Non-blocking MPI	100	32	0.162

Table 3: scaling limit corresponding to MPI rank, grid size = 200

Executable	Grid Size	MPI Ranks	Execution Time Mean, s
MPI SENDRECV	200	1	13.191
MPI SENDRECV	200	2	6.719
MPI SENDRECV	200	4	3.539
Non-blocking MPI	200	6	2.606
Non-blocking MPI	200	8	2.011
Non-blocking MPI	200	12	1.464
Non-blocking MPI	200	16	1.153
Non-blocking MPI	200	20	0.987
Non-blocking MPI	200	24	0.902
Non-blocking MPI	200	28	0.862
Non-blocking MPI	200	32	0.768

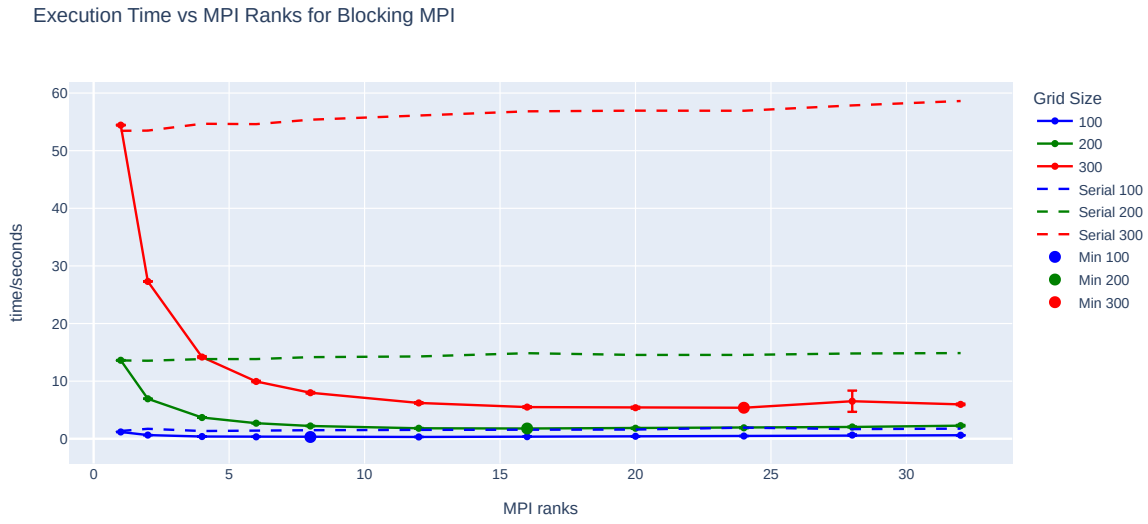
Table 4: scaling limit corresponding to MPI rank, grid size = 300

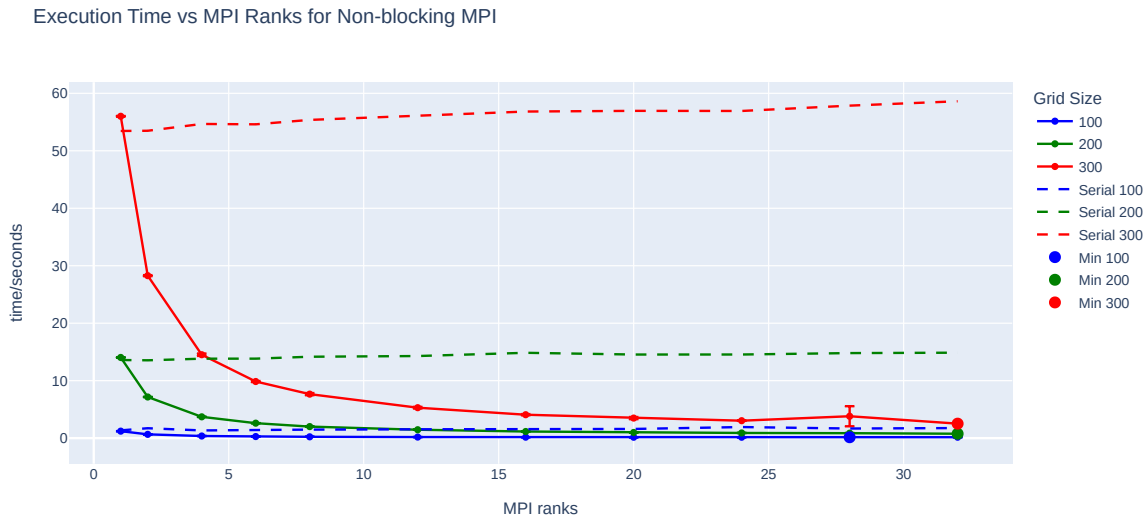
Executable	Grid Size	MPI Ranks	Execution Time Mean, s
MPI SENDRECV	300	1	52.647
MPI SENDRECV	300	2	26.531
MPI SENDRECV	300	4	13.813
MPI SENDRECV	300	6	9.696
Non-blocking MPI	300	8	7.658
Non-blocking MPI	300	12	5.300
Non-blocking MPI	300	16	4.078
Non-blocking MPI	300	20	3.491
Non-blocking MPI	300	24	3.034
Non-blocking MPI	300	28	3.811
Non-blocking MPI	300	32	2.523

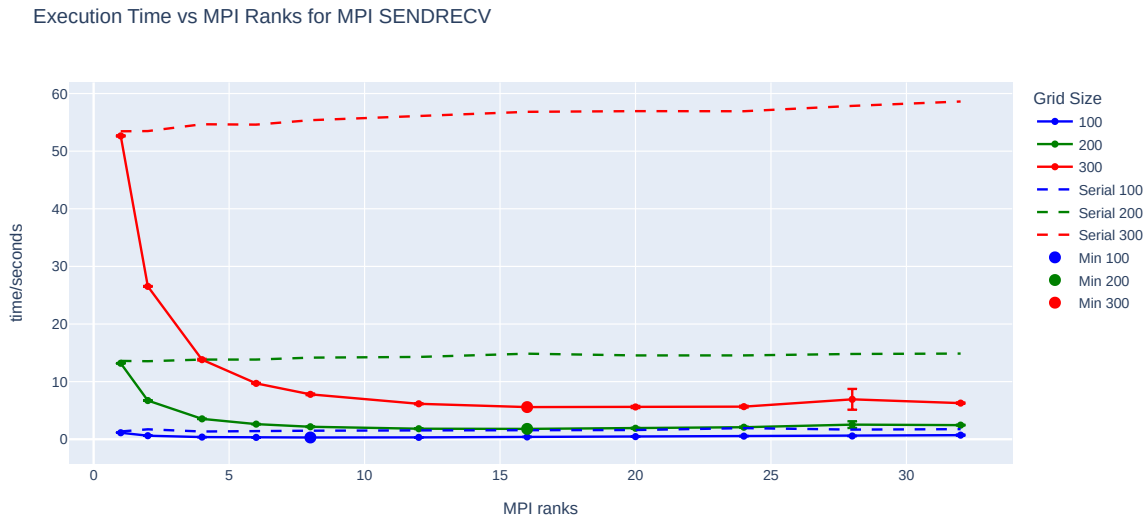
Globally, Non-blocking MPI shows better results, but when considering segments with MPI ranks ≤ 6, in some cases MPI SENDRECV demonstrates better performance. Moreover, the larger the Grid size, the wider the range of MPI ranks at which MPI SENDRECV is faster.

> How does this change as the grid gets bigger and why?

The larger the grid size, the greater the delta with the serial version. It is also obvious that the larger the grid size, the more time is required. Put simply, the larger the dimension, the more efficient it is to use parallelization (see Table 1, 'Serial time/Min time' column).







> Which MPI version gives the best performance: blocking MPI SEND and RECV, non-blocking MPI SEND and RECV or MPI SENDRECV ?

Answer: **non-blocking MPI**