Foundations of HPC - Thin node gemm

Filippo Olivo

Table of contents

Size scaling

Single precision

The theoretical peak performance for a single socket on an thin node is:

$$P = n.cores \cdot frequency \cdot \frac{FLOPC}{cycle} = 12 \cdot 3.7GHz \cdot 64Flops = 2841.6GFlops$$

A thin node can deliver 64 single precision floating point operations per cycle. I have tested the math libraries with the OpenMP options `OMP_PROC_BIND=close` and `OMP_PROC_BIND=spread`

The table below shows the peak performances for the different math libraries:

Settings	OpenBlas	Size OpenBlas	MKL	Size MKL
Close	1613.81	15500	1607.68	15500
Spread	1642.47	19000	1647.67	15500

From the table we can see that the difference between the different algorithms and the OpenMP are quite small. The graphs below shows the differences beteween the different policies in size scalability (one graph for each math library). In both the cases we can see that the best OpenMP option is spread. A possible explaination to this behaviour could be a better memory usage and the fact that we have both of the L3 cache when we work on two different sockets.

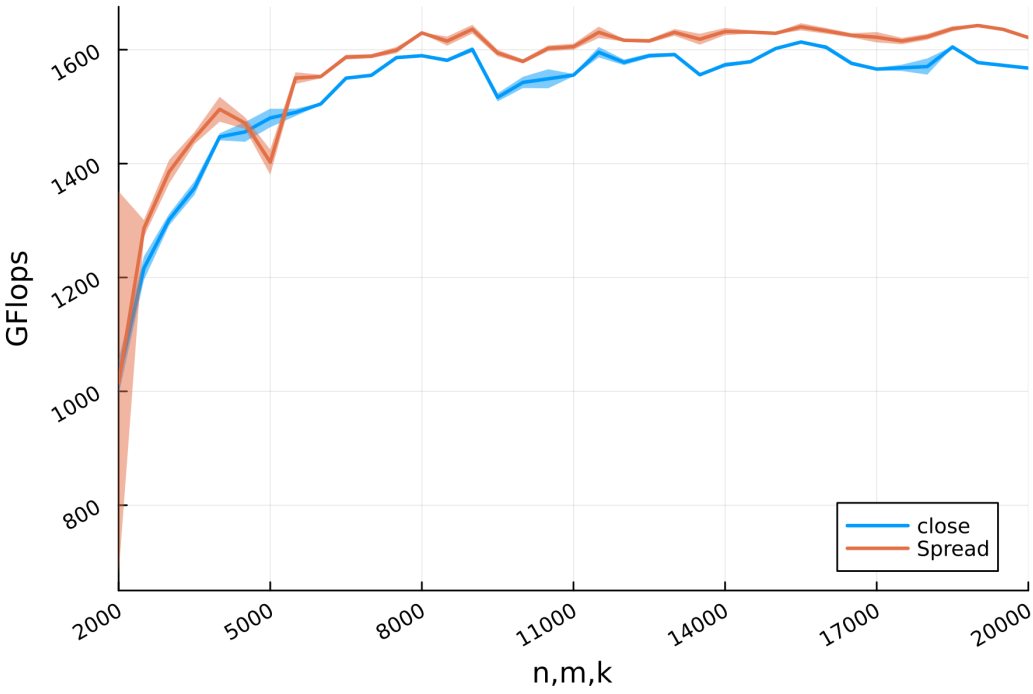

Figure 1: Comparison OpenBlas - Float

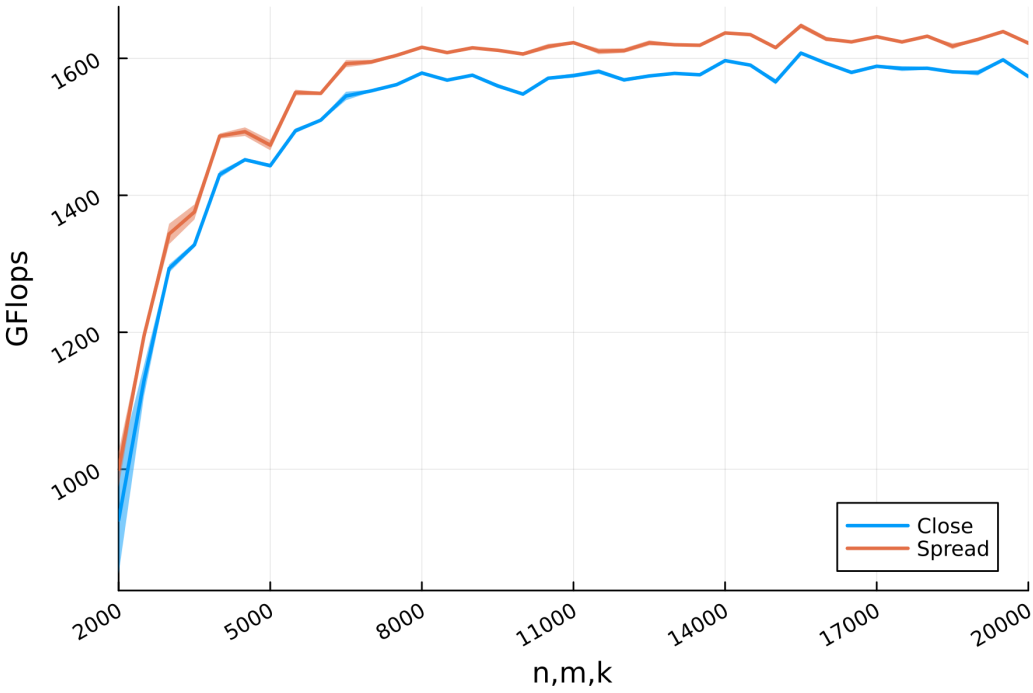

Figure 2: Comparison MKL - Float

The graphs below shows the comparison between the two math libraries. We can see that the MKL and OpenBlas are almost equivalent in both the cases:

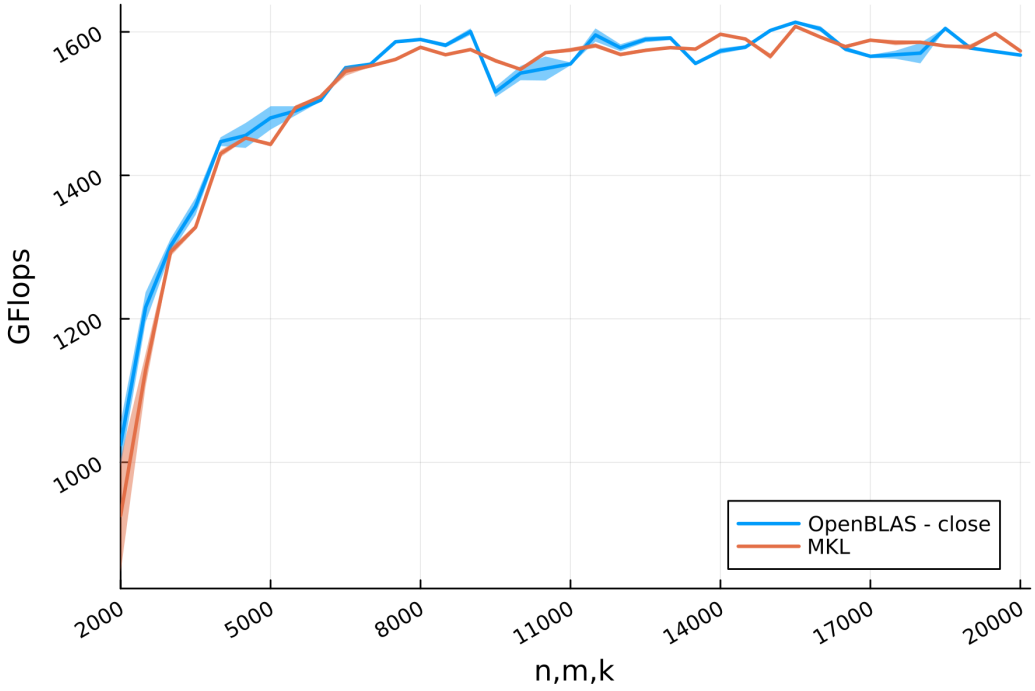

Figure 3: Comparison close - Float

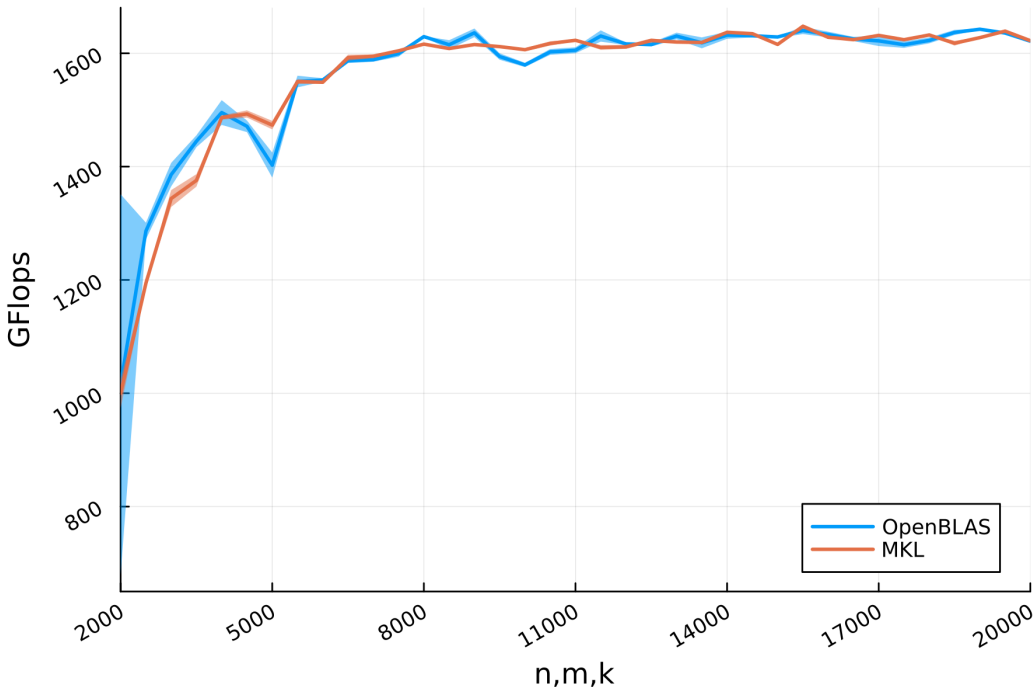

Figure 4: Comparison spread - Float

Double precision

The theoretical peak performance for a single socket on an thin node is 1420.8 GFlops

A thin node can deliver 32 double precision floating point operations per cycle. I have tested the math libraties with the same OpenMP options of the sngle precision case The table below contain the peak performance of the libraries:

Settings	OpenBlas	Size OpenBlas	MKL	Size MKL
Close	792.92	9500	813.6	9500
Spread	810.76	11500	828.96	18000

The peak performances of MKL are higher than the ones of OpenBlas. Also the difference between the two math libraries is larger than in the floating point case. The graphs below shows the differences beteween the two OpenMP options (one graph for each math library). We can see that the `OMP_PROC_BIND=spread` is slightly better than `OMP_PROC_BIND=spread`

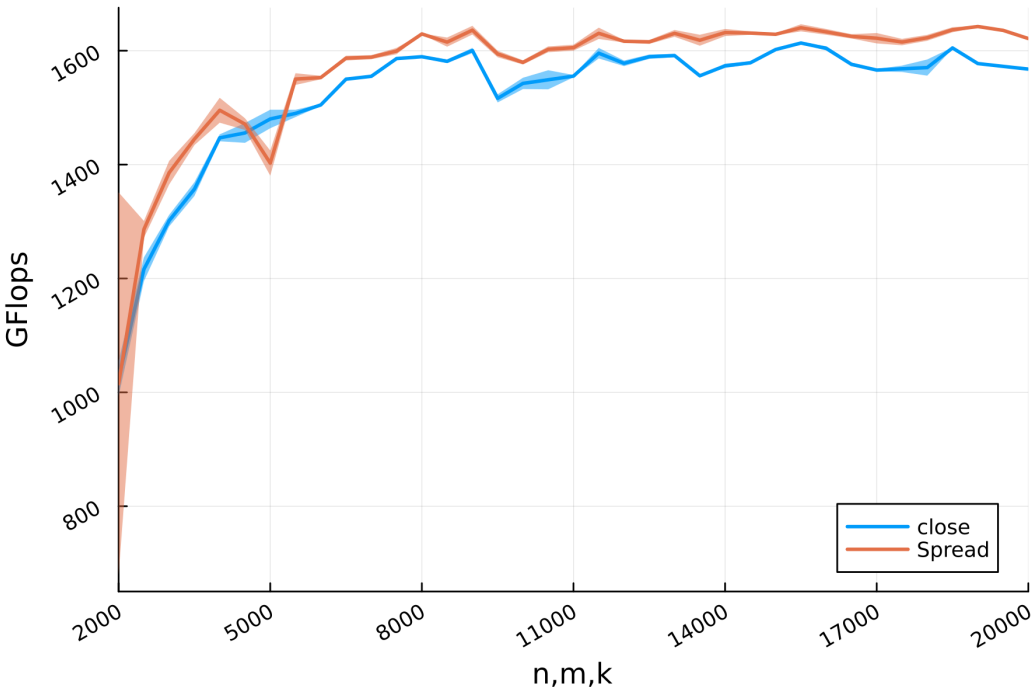

Figure 5: Comparison OpenBlas - Float

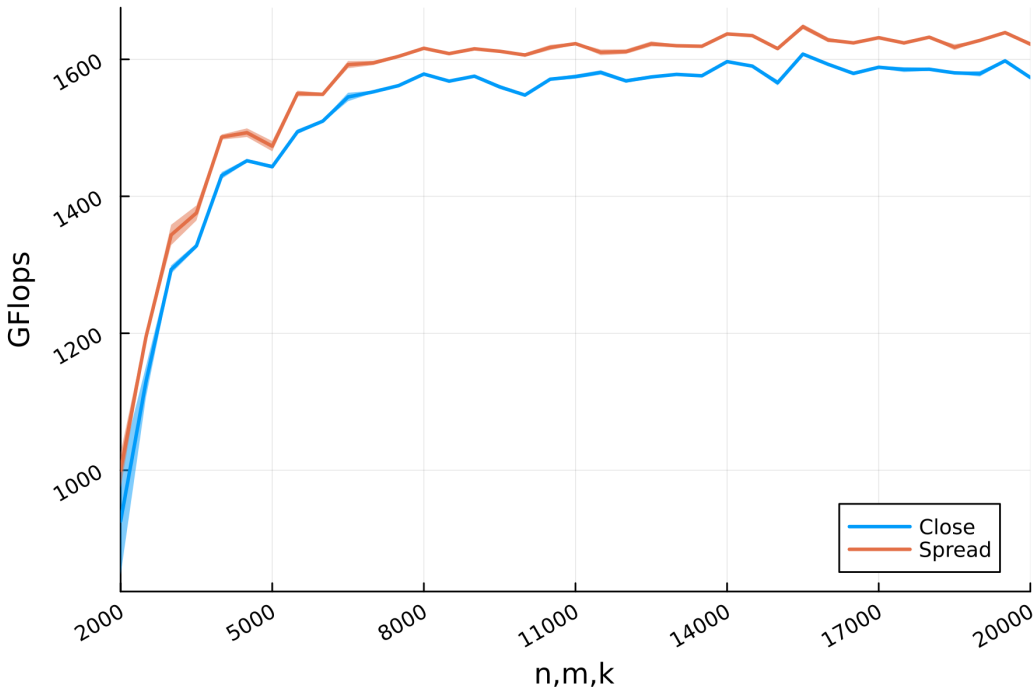

Figure 6: Comparison MKL - Float

Core scaling

In this section I analyze the behaviour of the different math libraries on a fixed value of n, m, k when the number of OpenMP threads increases. I have run the program for $n = m = k = 10000$. I have used the OpenMP options `OMP_PROC_BIND=spread` and `OMP_PROC_BIND=close`.

Single precision

The graphs below show the OpenMP scalability. We can see that MKL speed up follows amost perfectly the theoretical one. As it concern OpenBlas we can see that the speed up is a bit above the theoretical speed up.

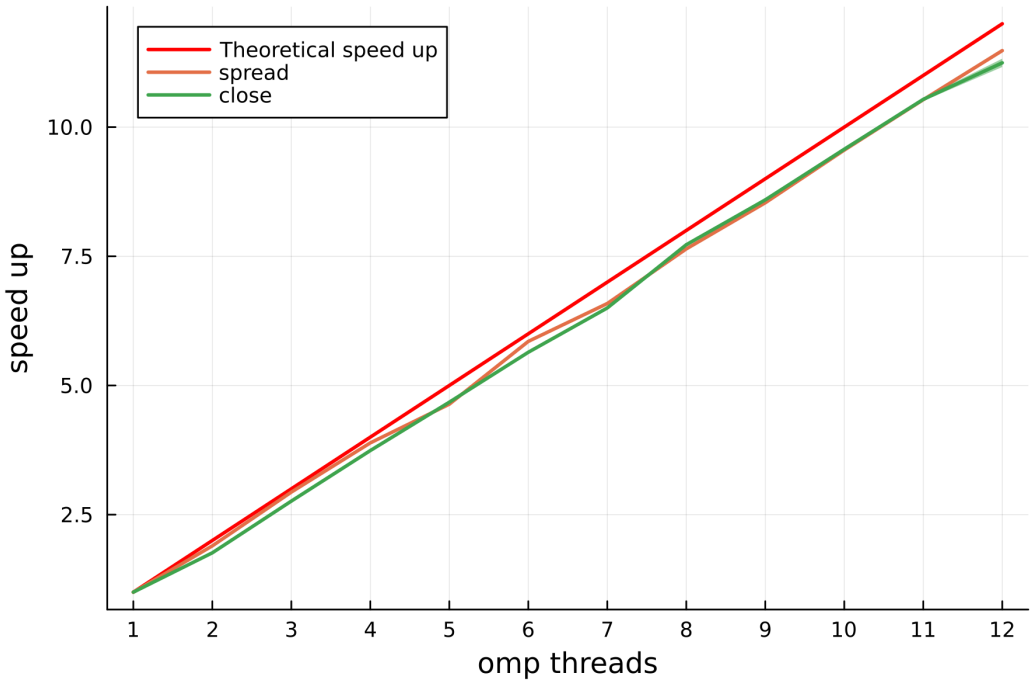

Figure 7: Scalability OpenBlas - Float

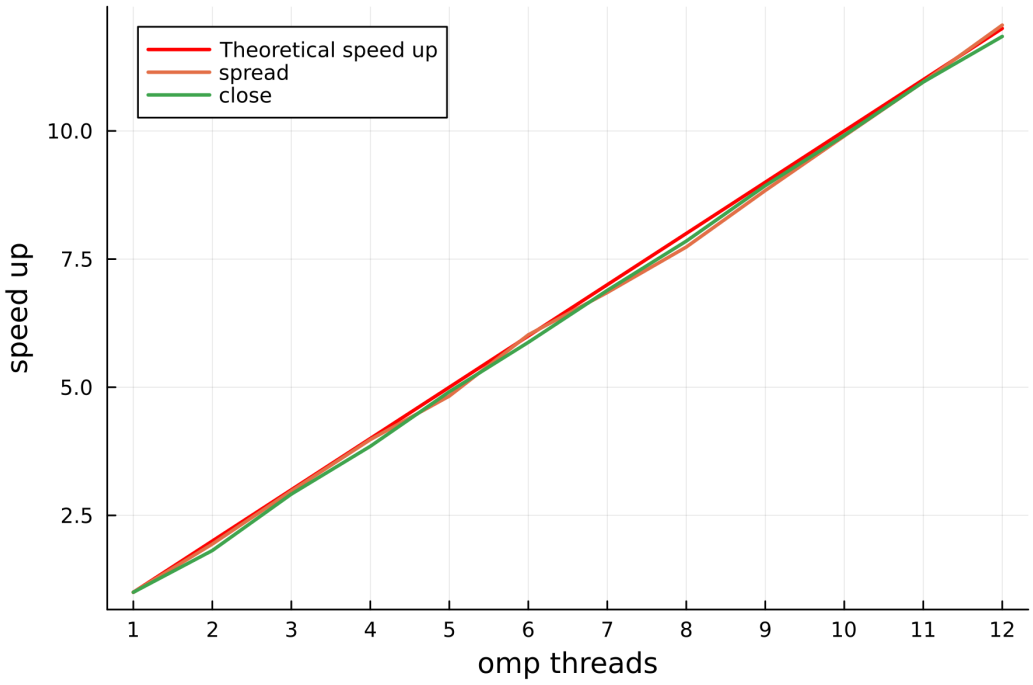

Figure 8: Scalability MKL - Float

Double precision

The graphs below show the OpenMP scalability. We can see that OpenBlas speed up with `OMP_PROC_BIND=spread` follows almost the theoretical one, while with `OMP_PROC_BIND=close` is slightly lower. From the MKL graph we can see that the speed up with `OMP_PROC_BIND=spread` is higher than the theoretical one, while with `OMP_PROC_BIND=close` it follows almost the theoretical line.

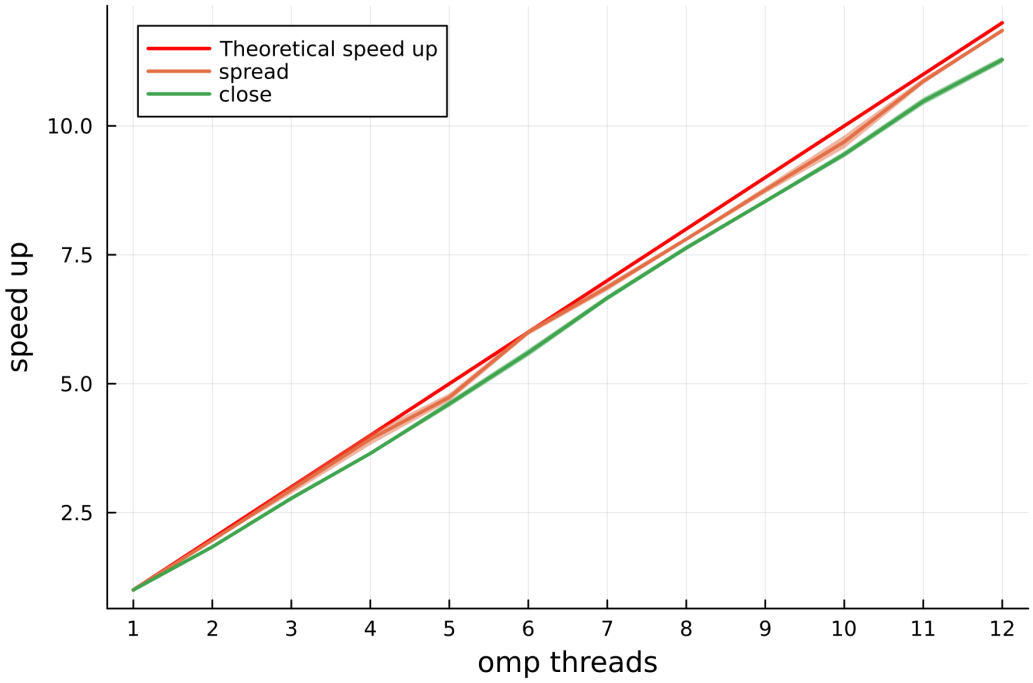

Figure 9: Scalability OpenBlas - Double

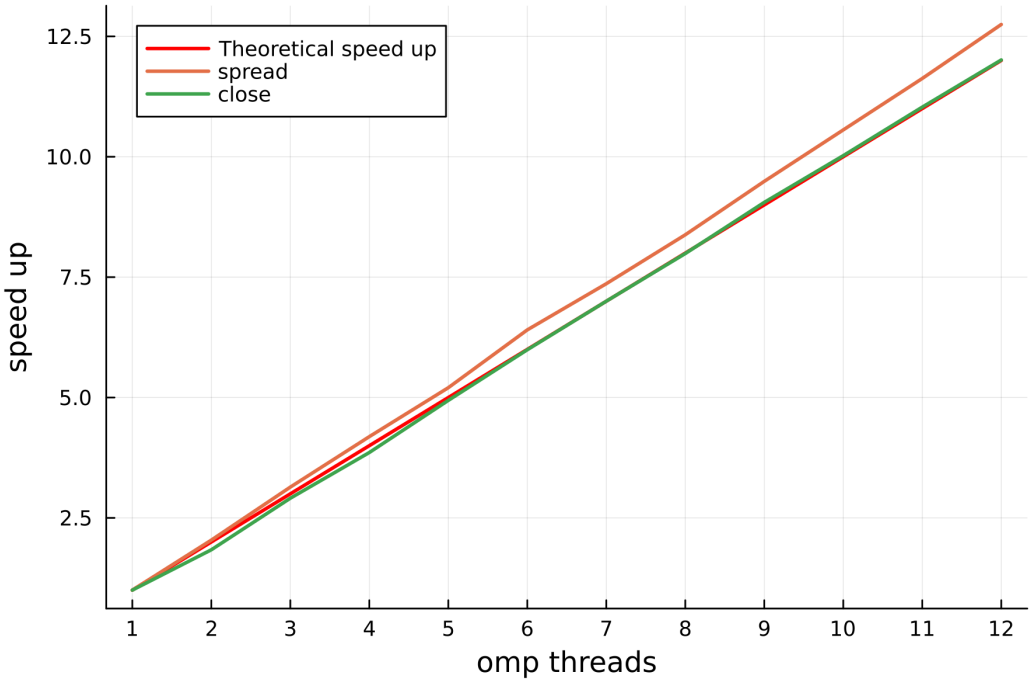

Figure 10: Scalability MKL - Double