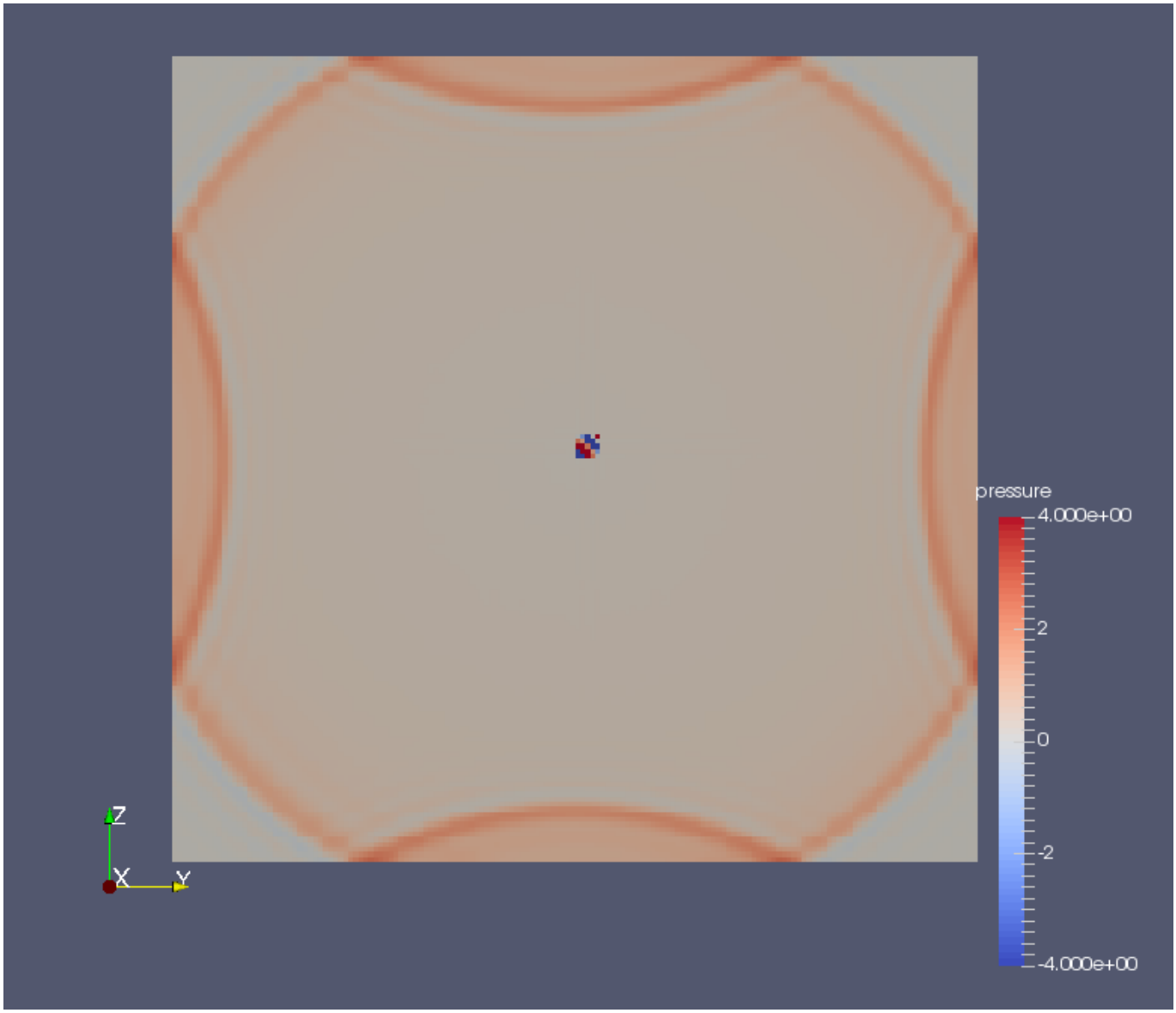
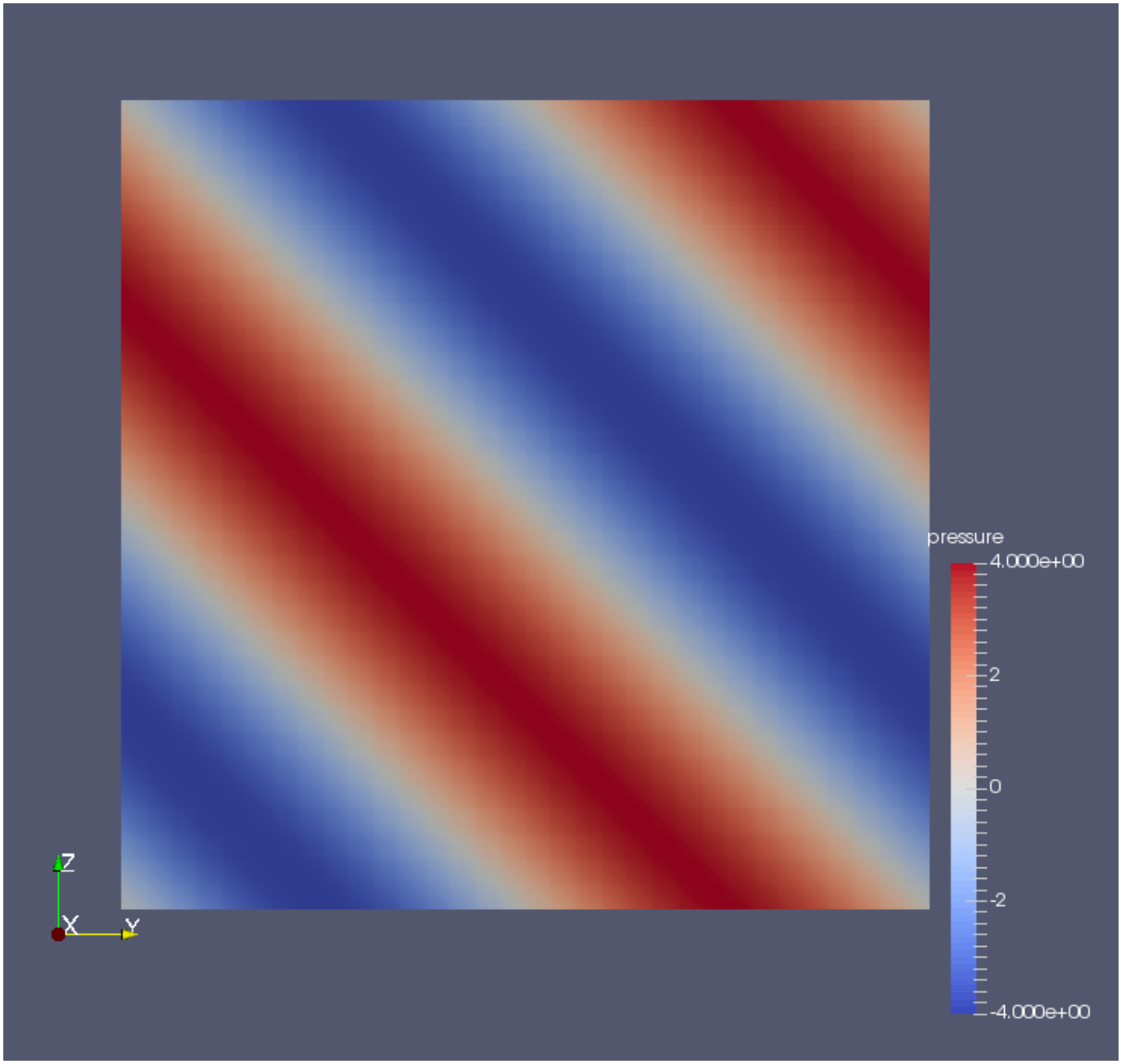
Project Report

The generated output data of our program looks correct. Samples for orders 6 and than can be found in the `output` folder. The tests were executed on a single node using a 32x32 grid with 16 MPI tasks and 4 OMP threads per task. The log files for these tests are also located in the `output` folder.



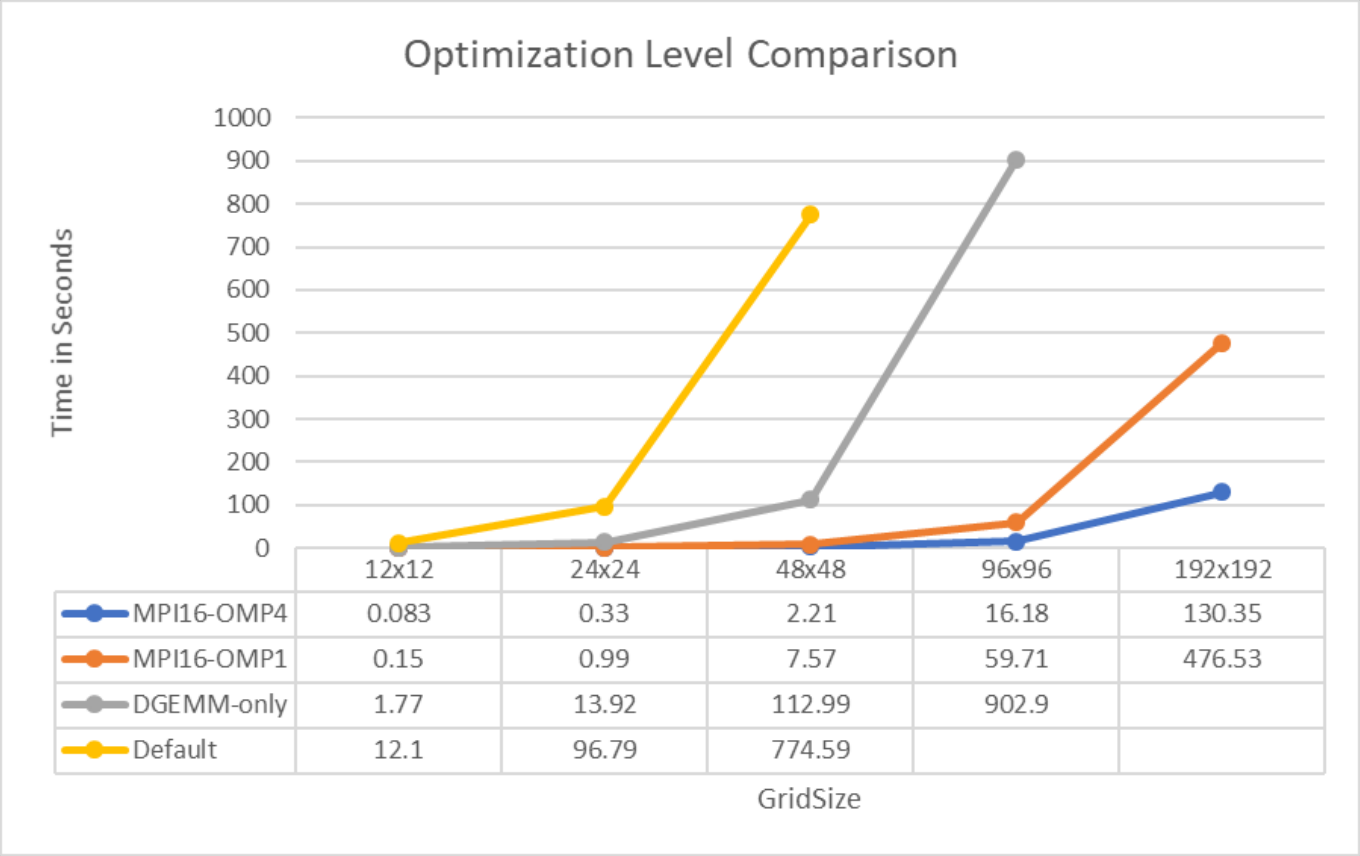


We tried to execute Vtune Amplifier profiling on the final code, but sadly all the batch nodes of the cluster were occupied and our job scripts didn't get executed in time.
Executing the profiler from the interactive shell does not seem to work as it can be seen in the results in the `logs/final code/Vtune amplifier/useless results` folder.

Execution Time Comparisons

For execution time comparisons the `Stopwatch.h` that was used in the assignments is used again.

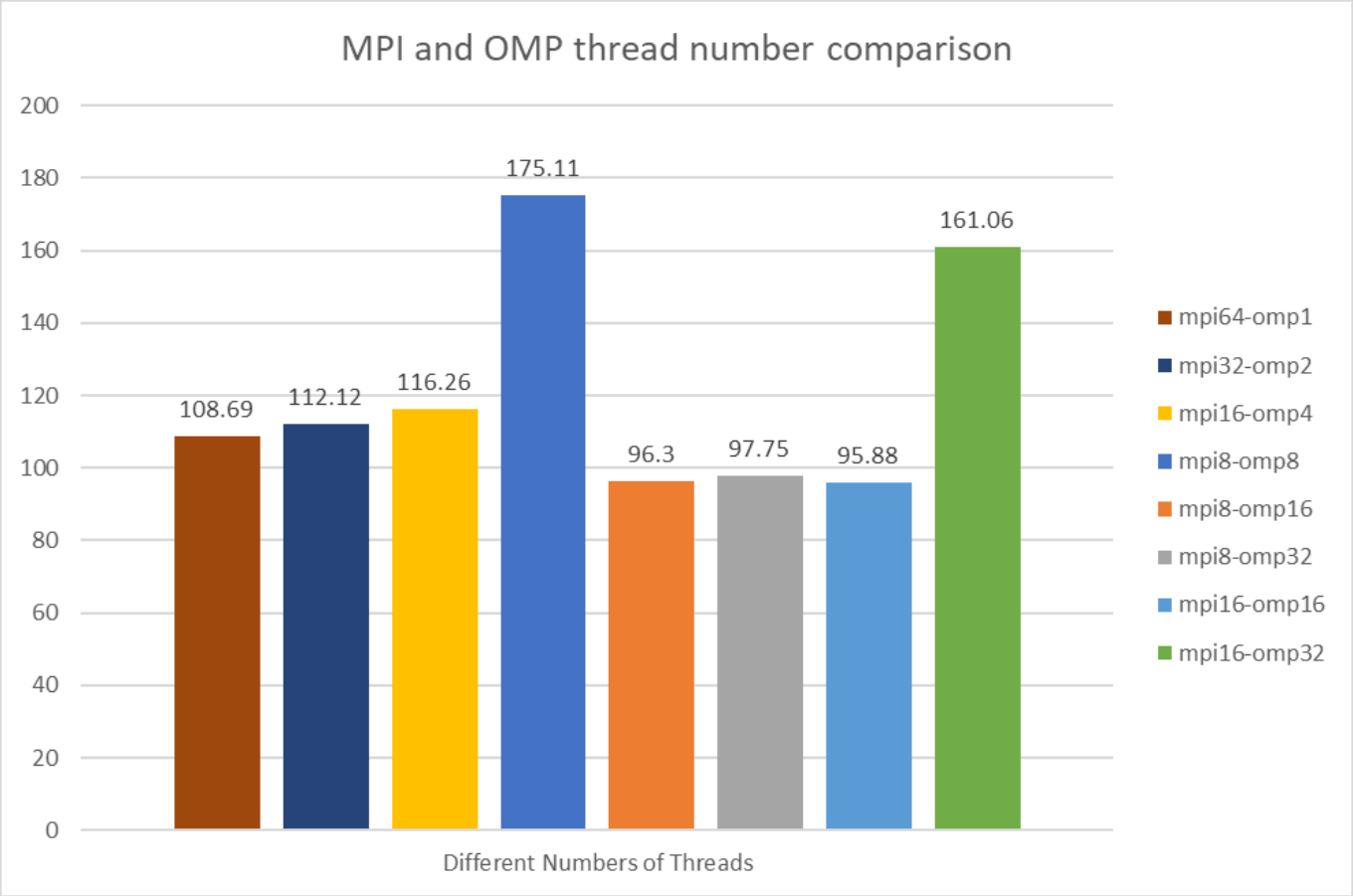
The difference in execution time for different levels of optimization was tested for Order 6. When using MPI 16 Tasks were used. The application was executed without output. The results can be found in `logs/Optimization comparison`.



	12x12	24x24	48x48	96x96	192x192
MPI16-OMP4	0.083	0.33	2.21	16.18	130.35
MPI16-OMP1	0.15	0.99	7.57	59.71	476.53
DGEMM-only	1.77	13.92	112.99	902.9	
Default	12.1	96.79	774.59		

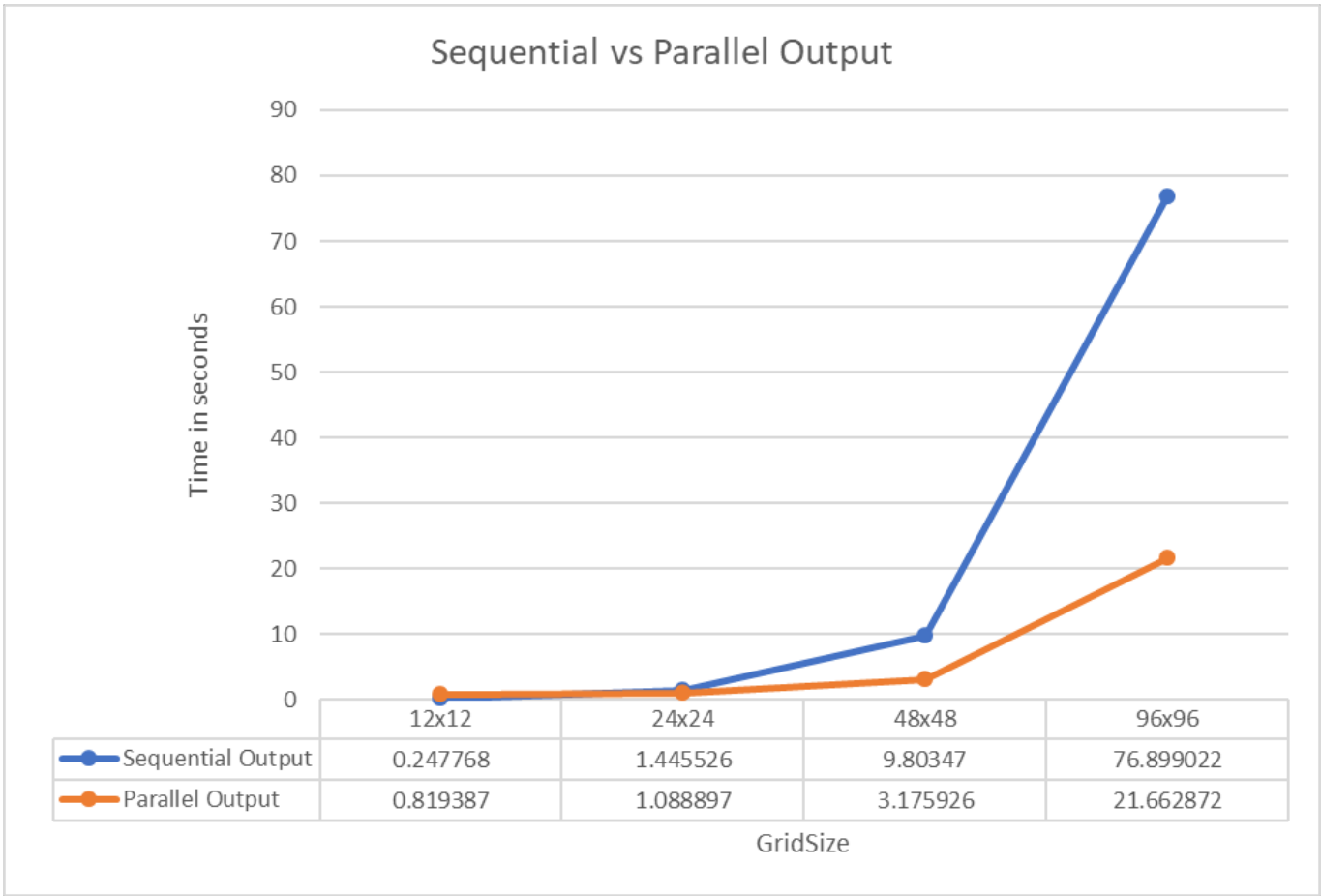
The default implementation, a version only using DGEMM optimization, a version using MPI+DGEMM and a version using MPI+OMP+DGEMM were tested.
The final optimized version using MPI and OpenMP is by far the fastest one. For the first two results of the larger grid size were omitted due to really long execution times.
For a grid of 48x48 the final optimized version is 350 times faster than the original implementation.
The results show that every implemented optimization achieves a reasonable speedup. The speedup between OMP and no OMP is in this case around 3.5 which can kind of be expected considering it uses 4 times more cores in this configuration.

Using Order 10 and a Grid Size of 96x96 it was tested which combination of MPI and OMP threads is the fastest.



When utilizing all 64 cores of a KNL Xeon Phi the execution time slightly increases when lowering the MPI tasks and even spikes when using only 8 MPI tasks.
Using hyperthreading helps the performance up to a certain point. It seems that the optimal performance can be achieved when using hyperthreading with 256 threads running in total. This is also the maximum supported thread count per node. The logs can be found in `logs/Thread number comparison`.

The improved execution time for the parallelized output was tested for Order 6, 16 MPI Tasks and 4 OMP threads per task.
The code without HDF5 parallelization can still be found in the `combined-no-hdf5` branch. The results can be found in `logs/Output parallelization`.



It is clear that the parallelized output is improving the performance greatly. The improvement becomes larger for larger grids.
For a 96x96 grid the parallel output is already about 3.65 times faster.