

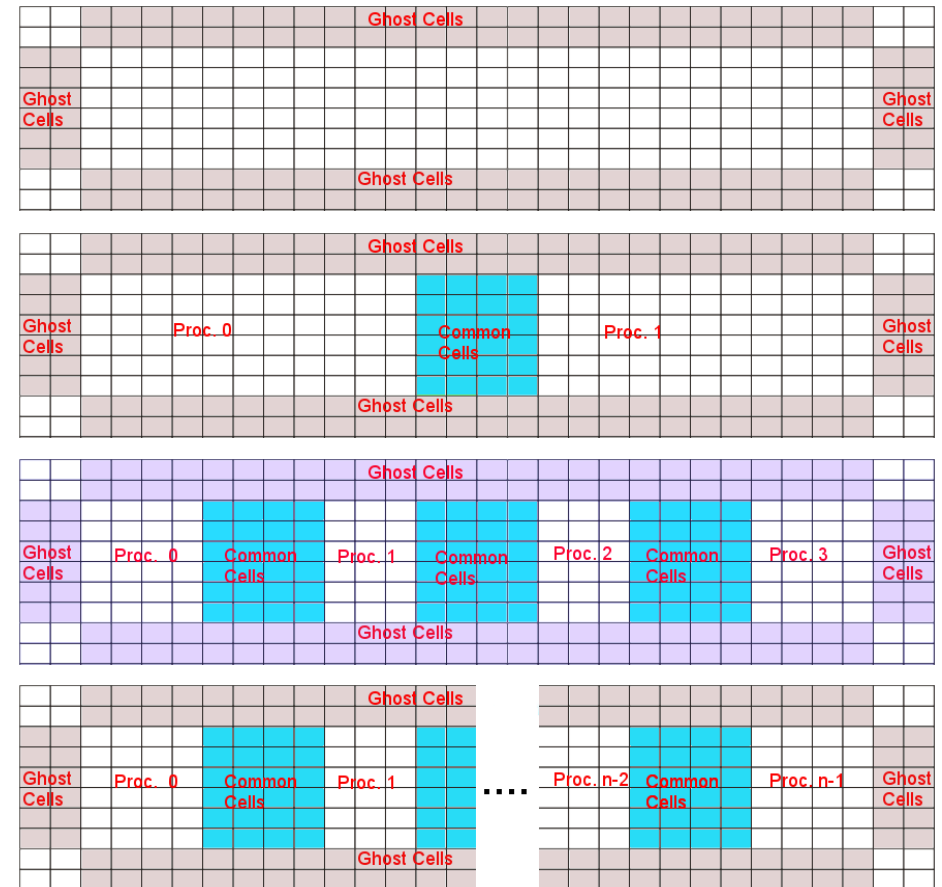
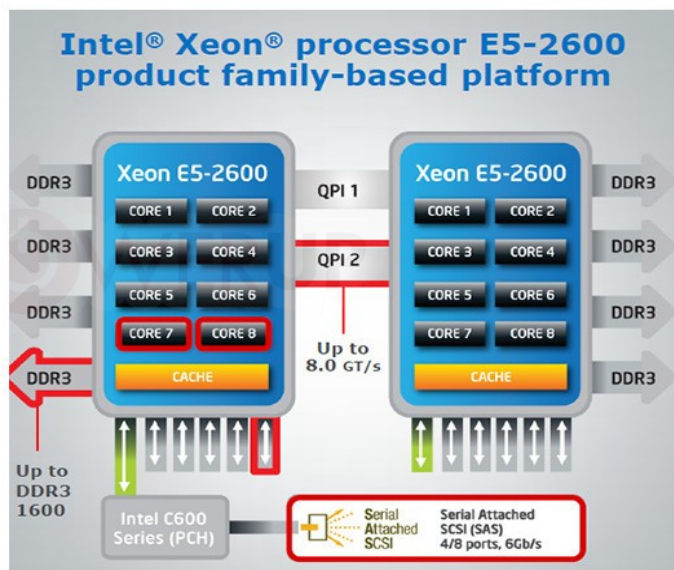
Introduction

Parallelization of a 2D Hydro code using MPI

Course: ESC401 - HPC1b: Parallel Computing

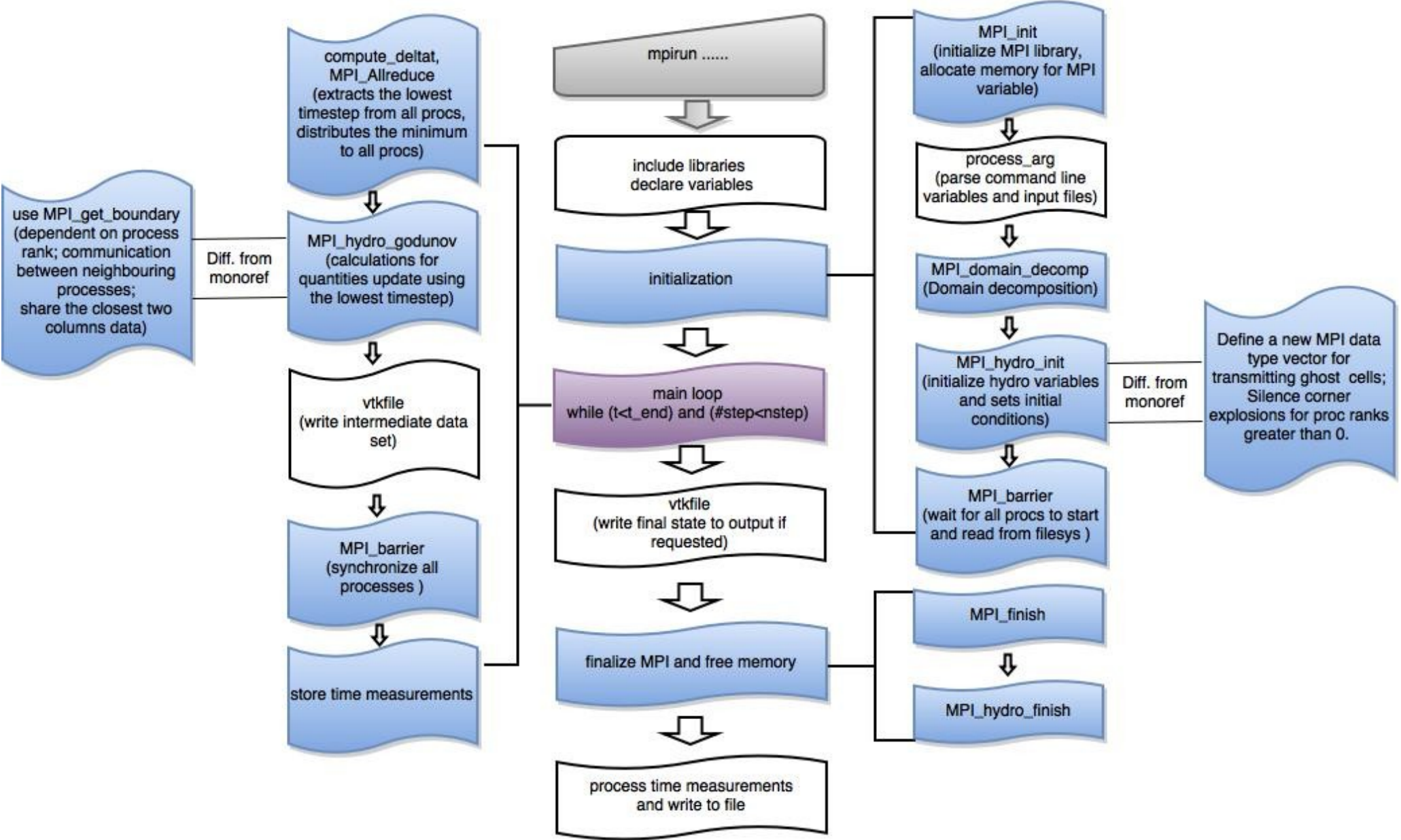
Authors: Rafael Küng, Christian Reinhardt and Mihai Tomozeiu

- The original code simulates an explosion in the lower left corner (right images) of a 2D wind tunnel using a single process
- Number of individual proper cells $n_x(\text{rows}) * n_y(\text{columns})$ and $2 * (n_x + n_y)$ ghost cells for reflective boundaries
- Approach:
 - general idea: modify as little code as possible
 - slicing the domain along the x-axis
- Domain distribution in right figures panels:
 - 2nd: 2 cores; 3rd: 4 cores; 4th: n cores
- Each neighboring processes need to communicate the closest two columns of proper cells to each other
- Minimum time step found among processes will become the time step of all processes

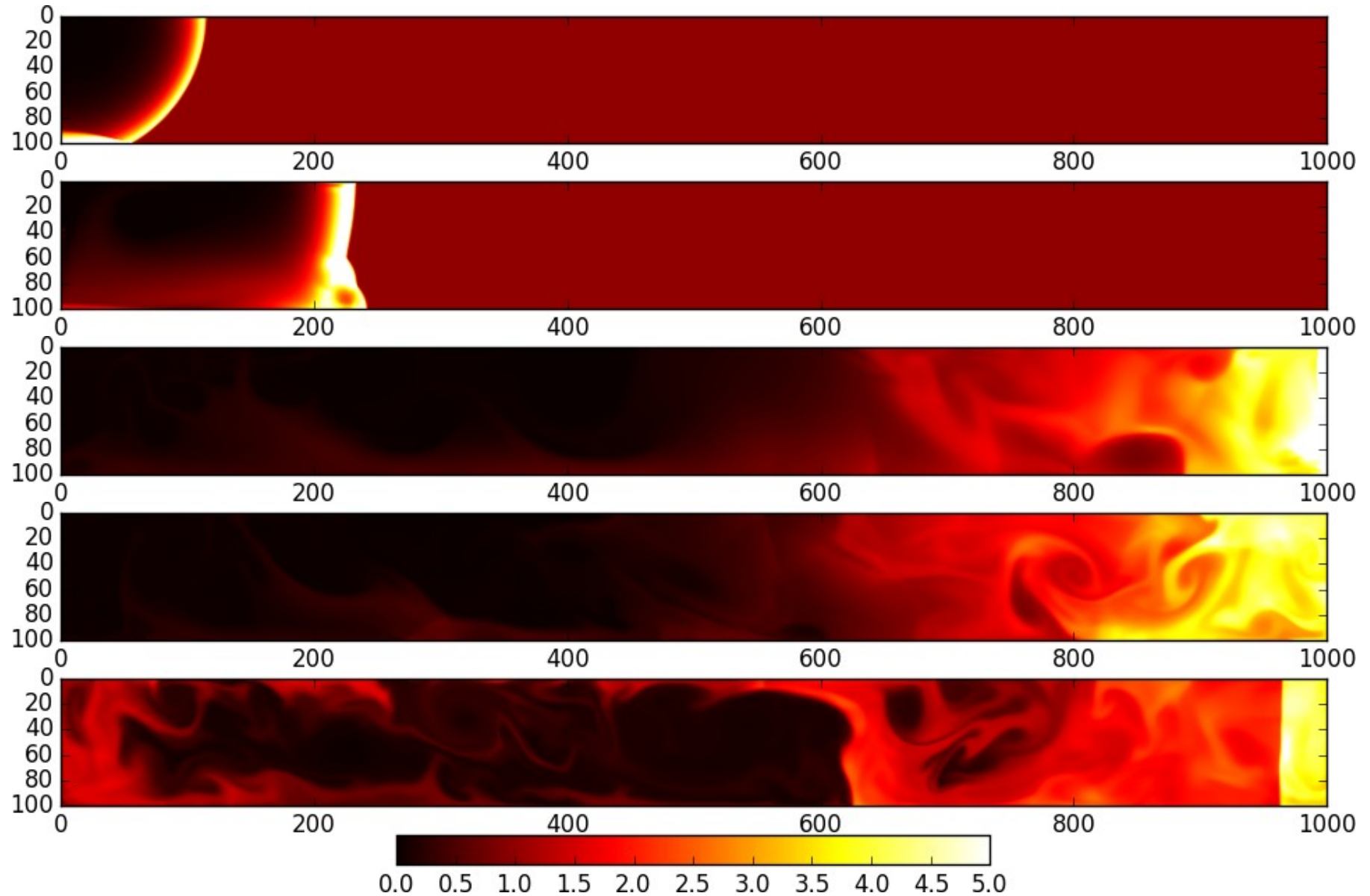


- The simulations were run on the local zbox4
- Intel Xeon E5-2660 (family platform sketch in the left image)
- 8 cores per socket
- 2 sockets per node
- Internode communication with QDR Infiniband

Code Description



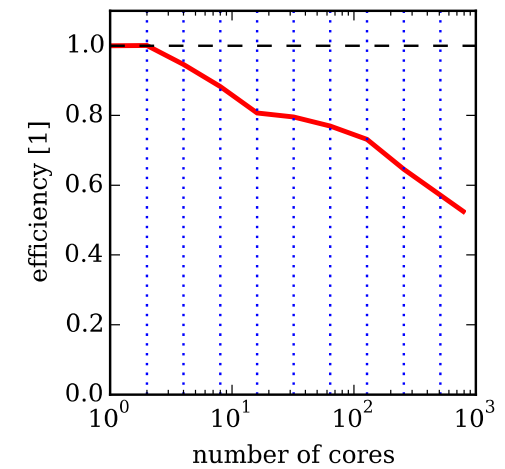
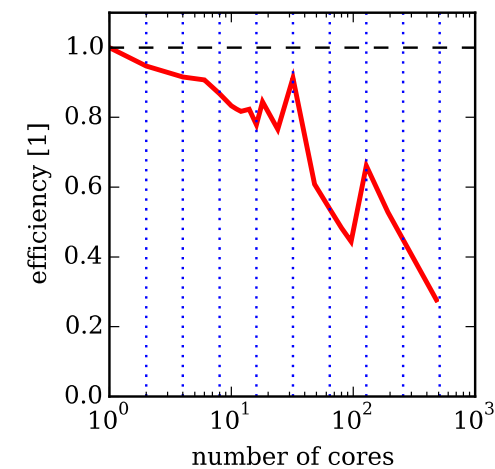
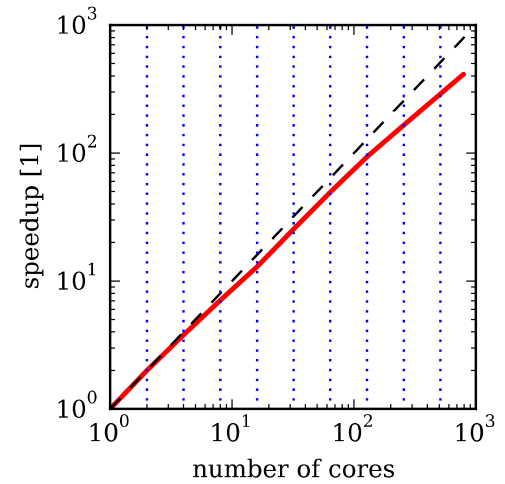
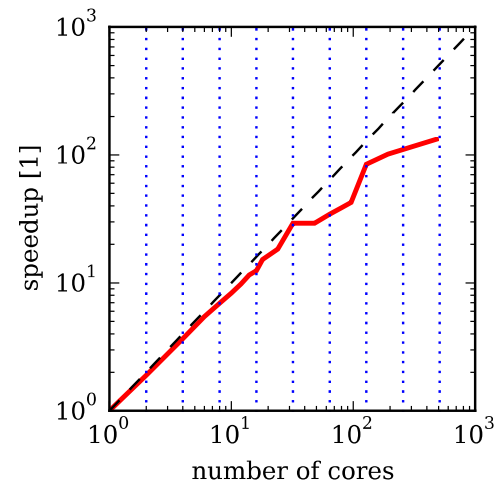
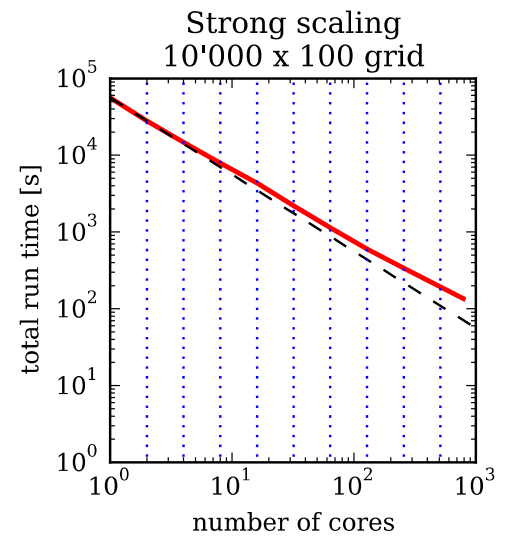
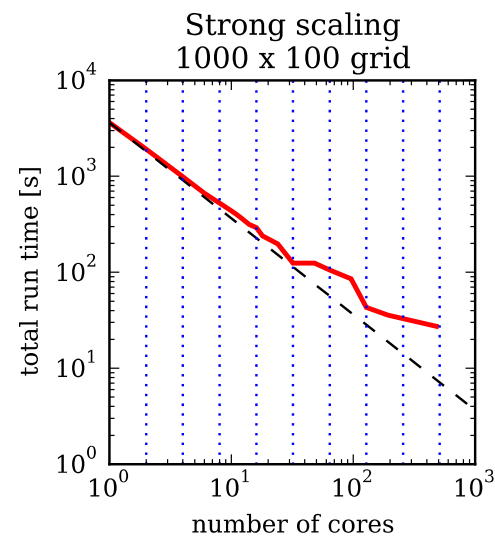
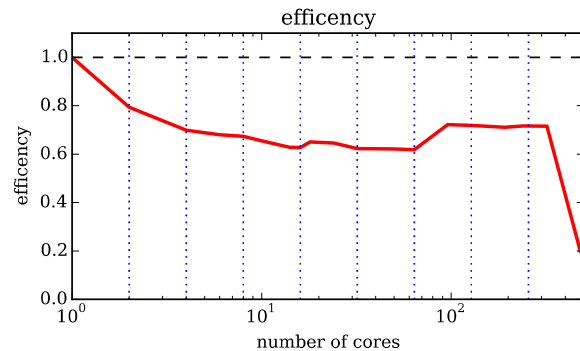
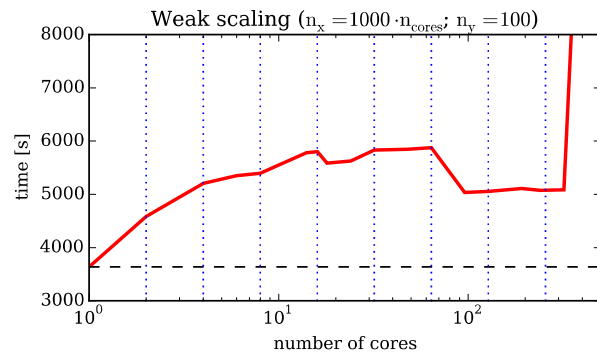
Results / Output



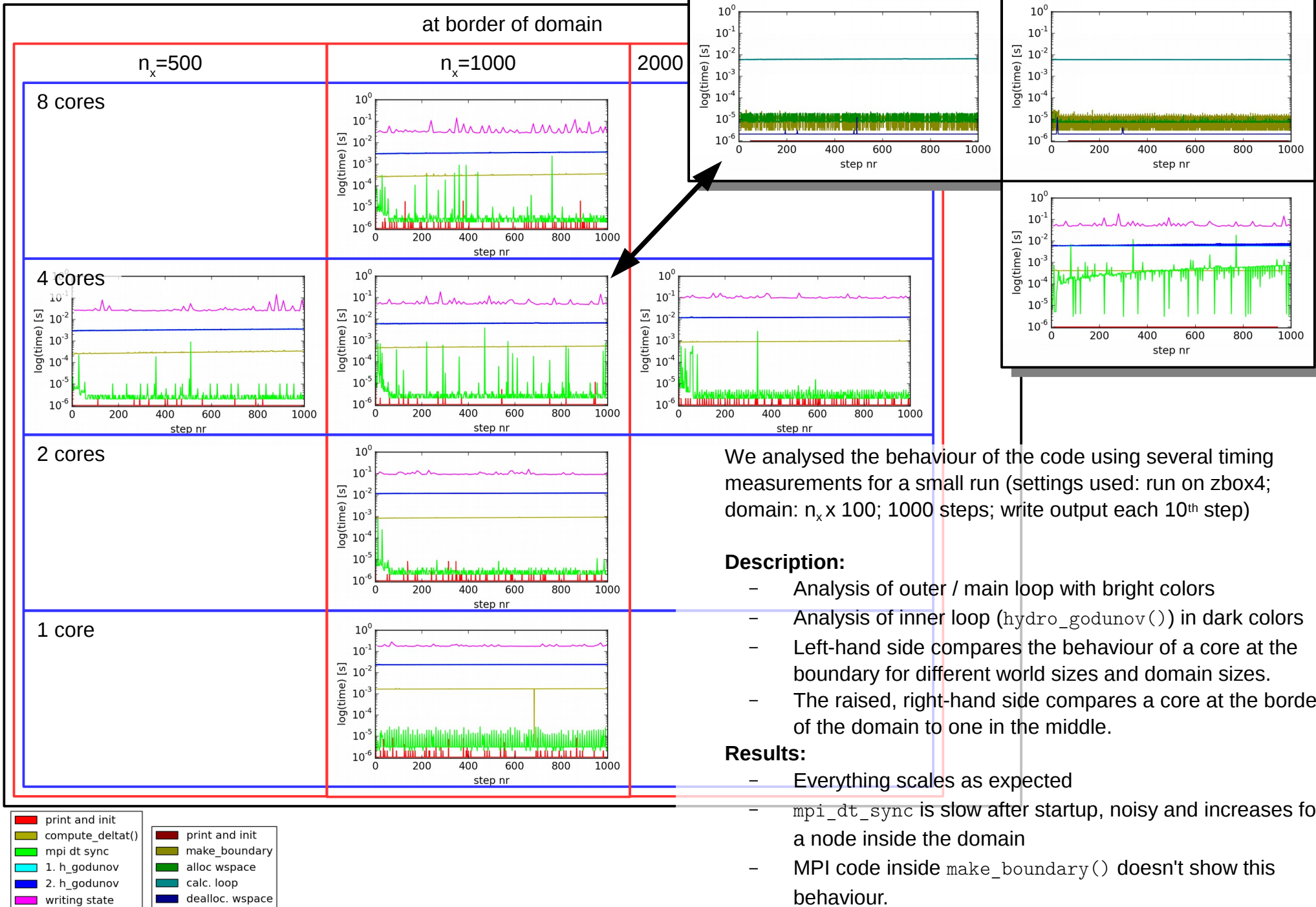
Example of a run (1000x100 grid): An initial overpressure in the top left cell causes a blast wave to propagate through the computational domain. The walls are reflecting so the shock bounces back at the domain boundaries and interferes with the original wave. This causes an oscillating bulk motion that generates a lot of turbulence in the inner part of the domain. The colors represent the fluid density where black is zero and white is at least five.

Scaling Behaviour

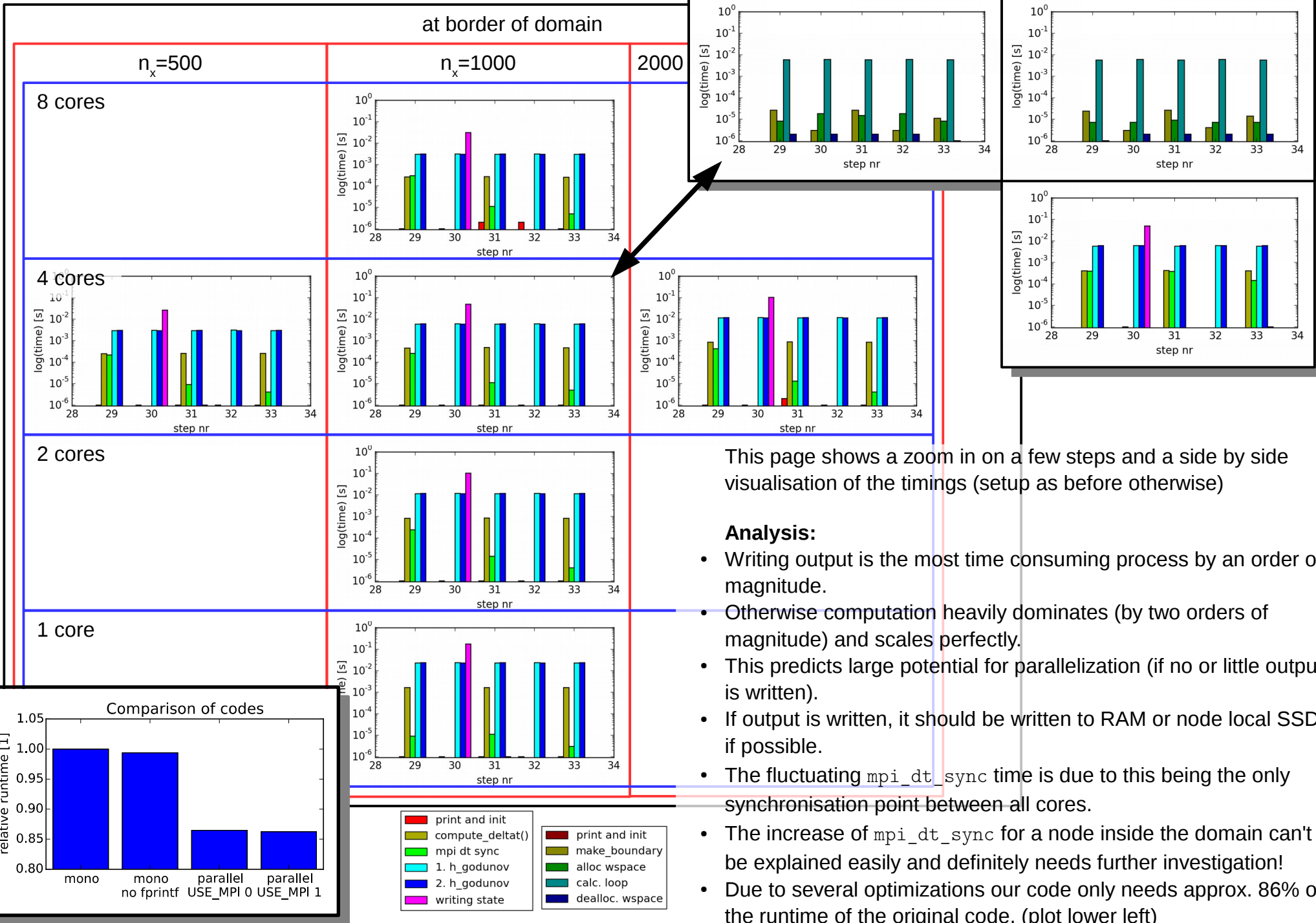
- Strong Scaling (right figures)
 - At large numbers of cores (approaching 500) the great majority of the cells have to be shared among neighbouring processes.
 - Above 500 cores the code breaks down since the number of columns per process becomes less than 2. A minimum of 2 columns are needed for proper data transfer between neighbouring processes.
 - Please note: for the 10'000 x 100 grid: The value for 1 core is interpolated
- Weak Scaling (left figures)
 - n_x scaled with the number of cores, n_y constant
 - The 16 cores point marks the transition to internode communication showing a core performance worsening
 - The last run on 480 cores showed an anomaly. Possibly because we saturate the LAN or the file server.
- Legend: blue dotted lines show powers of 2 black dashed line shows ideal scaling behaviour



Code Diagnostics – Overview



Code Diagnostics – Details



Conclusions

- We successfully parallelized the existing serial code with MPI.
- Using our code we ran several simulations with different grid sizes on a varying number of CPU cores. Both the hard and the weak scaling plots show a clear performance improvement when run on multiple cores.
- The analysis shows that hard scaling up to 480 cores works (but deviates slightly from ideal)
- Outputs are - as expected - a bottleneck and reducing outputs does indeed increase performance.
- In general we are limited in the maximum number of cores to $n_x/2$ because we need at least 2 columns in the inner part of the computational domain.
- We tried to add OpenMP, but didn't invest much time:
 - We failed to parallelize the outer loop, probably because of faulty declaration of variables.
 - Parallelizing the inner loops succeeded, but performance was inferior, it almost halved
 - This is to be expected, the overhead of spawning a thread for each small loop inside the main loop is too much.