HPCSE1
Robin Worreby, 16-921-298

Exercise 06
MPI IO, Hybrid MPI + OpenMP,
ADI scheme, and PSE method

2020-12-23
rworreby@student.ethz.ch

1. Diffusion: Parallel I/O

a) Figure 1 shows the written data from the parallel MPI IO code and Figure 2 show how the modules on EULER were loaded.

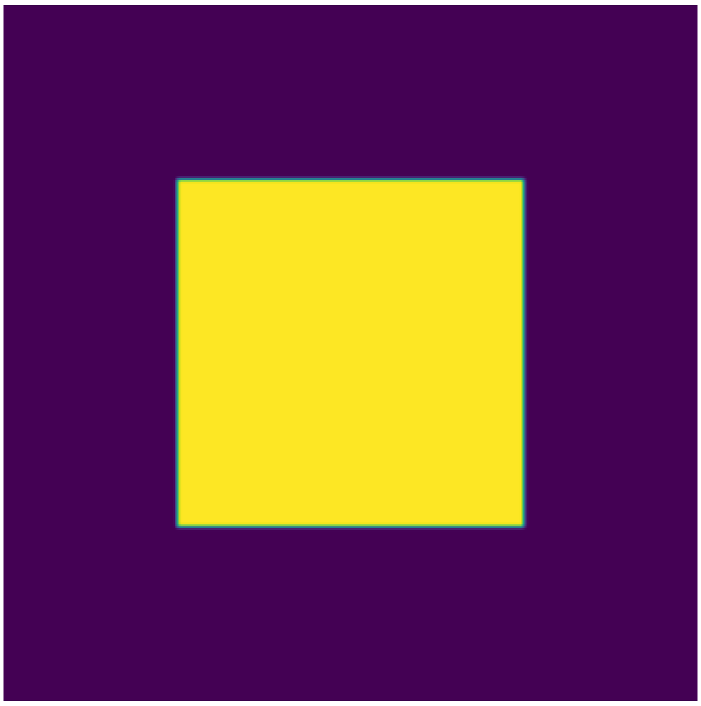
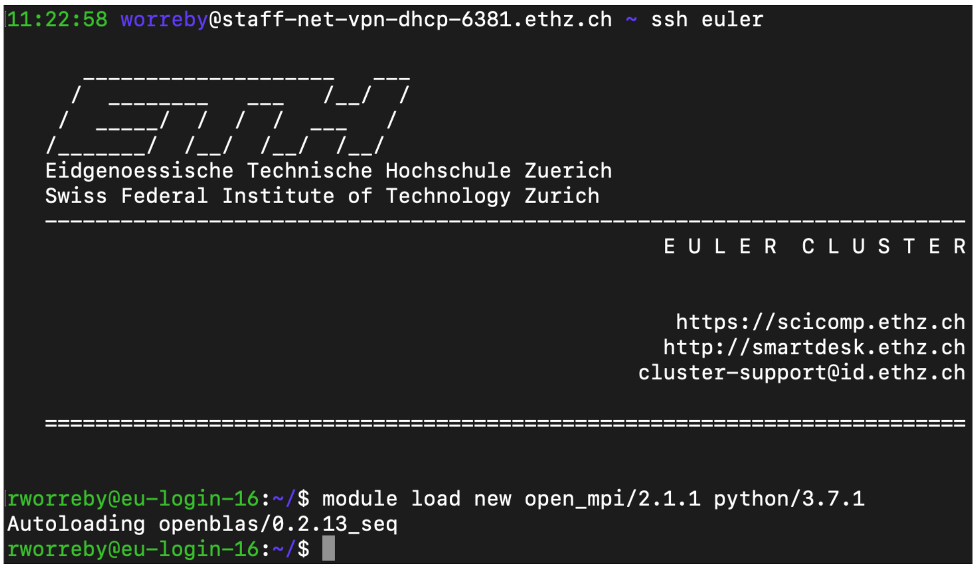


Figure 1: Plot of the data written by the parallel MPI IO function. Parameters: $D = 1, L = 2, N = 1024, ranks = 16$



```
11:22:58 worreby@staff-net-vpn-dhcp-6381.ethz.ch ~ ssh euler


      ____________________   ___
     /  ________   ___   /__/  /
    /  _____/  /  /  /  ___   /
   /_______/  /__/  /__/  /__/
   Eidgenoessische Technische Hochschule Zuerich
   Swiss Federal Institute of Technology Zurich
   -------------------------------------------------------------------------
                                                   E U L E R   C L U S T E R


                                                     https://scicomp.ethz.ch
                                                    http://smartdesk.ethz.ch
                                                  cluster-support@id.ethz.ch

   =========================================================================


rworreby@eu-login-16:~/$ module load new open_mpi/2.1.1 python/3.7.1
Autoloading openblas/0.2.13_seq
rworreby@eu-login-16:~/$ 
```

Figure 2: Screenshot showing the loading of the modules on EULER.

b) We use 16 ranks to check the performance of our parallel MPI IO implementation. When plotting the results, as seen in Figure 3, we can see that the parallel version demonstrates a better scaling performance. For a gridsize $N \times N$ with $N = 6144$ we can see that the execution time of the file write is roughly $3x$ lower for the parallel version. As we use 16 ranks our max achievable speedup is $S = 16$. We assume that the reason for not reaching this is because of the overhead from the MPI operations, like calculating the file pointer locations and opening the files on all ranks. Additionally, the amount of data we're writing is still rather small and we expect the gap to further increase as the data size increases. The presumed bottleneck in the sequential version is the sending of the data to the main rank, which is omitted for the parallel version. Another thing to note is that for very small sizes the sequential implementation outperforms the parallel version. This is, however, no surprise as the overhead described above is computationally more expensive than sending the data to the main rank for small data.

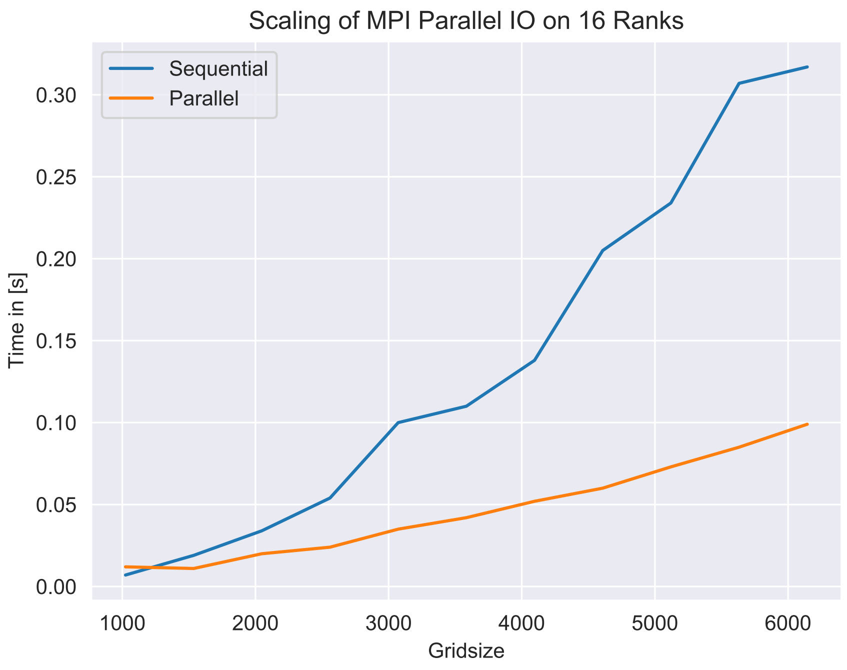


Figure 3: Plot of the parallel MPI IO scaling.