High Performance Computing for Weather and Climate Performance model for halo-updates

Nora Zilibotti, Paul Farron, Jonas Grütter
zilnora@student.ethz.ch, pfarron@student.ethz.ch, jgruette@student.ethz.ch

September 25, 2022

Abstract

We present a performance analysis of a 3-dimensional MPI implementation of the diffusion equation on a 2-dimensional worker grid for different numbers of ranks and halo-points, as well as different total domain sizes. Our model is divided into a description of MPI-communication and one for the stencil-operations on each rank. While our model takes into consideration the most relevant aspects that affect performance, some assumptions are specific to our domain size, both with respect to the total field, as well as with respect to the number of workers and the number of halo-points.

1 Introduction

In weather and climate models PDEs are integrated numerically on a discretized grid and compute evolution of the field on a certain point is computed from a compact neighborhood of gridpoints in it's vicinity. This repetitive task has to be done for all points in the domain, which is usually very time consuming if done by one worker in models with many gridpoints (such as a weather and climate model). One solution to obtain a speed up is to divide the domain into several parts and assign the corresponding computations for each different part to different workers while communicating the boundary conditions (halo-update, see Section 2.2). However, the speed up obtained by domain decomposition varies a lot depending on certain parameters such as the number of workers used or the halo size. The goal of this project is therefore to construct a simple performance model which will allow us to predict the performance given the run parameters of the halo-update.

2 Implementation

2.1 Model

This project aims to solve the high-order monotonic diffusion equation introduced in [2]:

$$\frac{\partial \phi}{\partial t} = S + (-1)^{n/2+1} \alpha_n \nabla^n \phi \tag{1}$$

where ϕ is any prognostic variable, S is the processes or sources and α_n the diffusion coefficient. This project focuses on the 4^{th}-order ($n = 4$) non-monotonic diffusion. Other processes are ignored ($S = 0$) and no limitors are taken into account. Equation (1) is therefore simplified into:

$$\frac{\partial \phi}{\partial t} = -\alpha_4 \nabla^4 \phi \tag{2}$$

The discretization of Equation (2) is performed using the finite difference method. As we are implementing a 3 dimensional field, a 3 dimensional grid is used with spatial resolution h in all directions and time resolution Δt. Let $\phi^n_{i,j,k} = \phi(x_i, y_j, z_k, t_n)$ where (x_i, y_j, z_k) is the coordinate of the grid point (i, j, k) and t_n is the time at the n^{th} step. The time derivative is therefore approximated by

$$\frac{\partial \phi^n_{i,j,k}}{\partial t} \approx \frac{\phi^{n+1}_{i,j,k} - \phi^n_{i,j,k}}{\Delta t} \tag{3}$$

And the 3-dimensional laplacian by:

$$\nabla^2 \phi^n_{i,j,k} \approx \frac{-6\phi^n_{i,j,k} + \phi^n_{i+1,j,k} + \phi^n_{i-1,j,k} + \phi^n_{i,j+1,k} + \phi^n_{i,j-1,k} + \phi^n_{i,j,k+1} + \phi^n_{i,j,k-1}}{h^3} := \Delta_h \phi^n_{i,j,k} \tag{4}$$

Using Equations (3) and (4), the update looks like the following explicit scheme:

$$\phi^{n+1}_{i,j,k} \approx \phi^n_{i,j,k} - \alpha_4 \Delta_h (\Delta_h \phi^n_{i,j,k}) \tag{5}$$

In order to deal with the grid points located at the boundary of the domain, halo-points are introduced. These are additional grid points which are updated in a periodic way by taking the value of the cells located at the opposite extreme part of the domain as shown in Figure 1 for a 2D representation. In this project, the field is 3 dimensional, we therefore also have to use halo points and periodic boundary conditions in the z direction as shown in Fig.2.

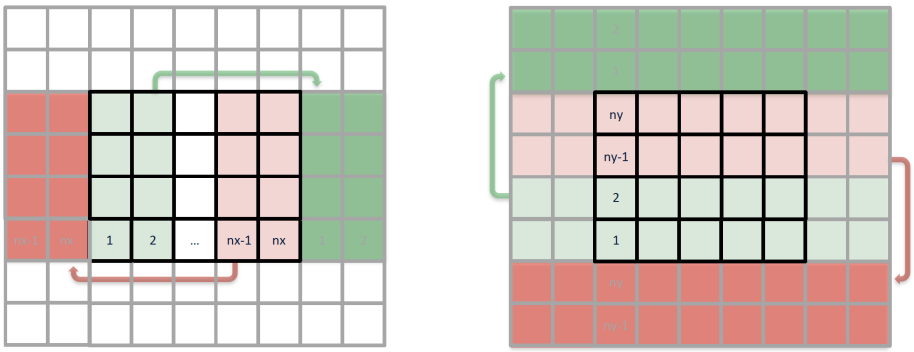

Figure 1: 2 dimensional Halo update. Taken from [1].

2.2 2 Dimensional Domain Decomposition

This project is focused on the speed up that can be achieved by using domain decomposition: By splitting the domain into several sub-domains, it is possible to assign each part of the

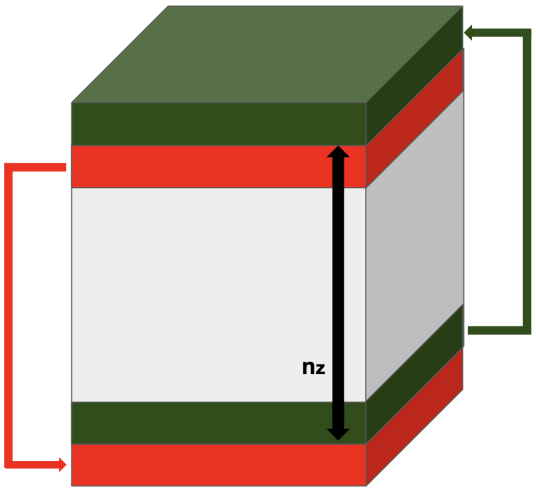

Figure 2: Halo update in z direction.

calculations corresponding to a certain part of the domain to different workers. However, the calculations must be coordinated using a communication interface between the different workers. This interface is called MPI, and the different workers correspond to MPI ranks.

In this case, we are interested in a performance model of a 2 dimensional halo update of a 3 dimensional field. The 3-dimensional field is thus decomposed in the x-y-plane as shown in Figure 3.

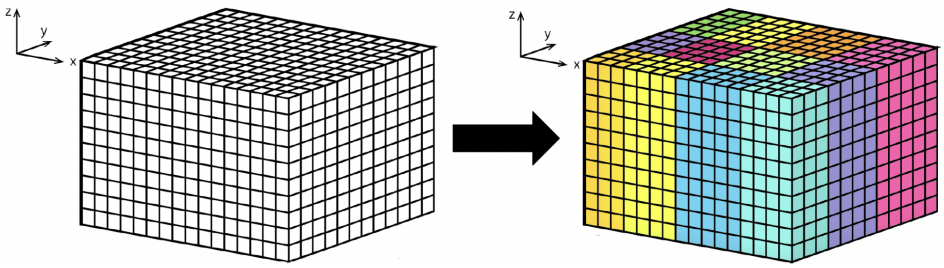

Figure 3: The computational domain (left) is 2d decomposed into several subdomains in x and y direction (right). Taken from [1] and modified.

The number of subdomains resulting from the decomposition is the number of workers (MPI ranks) available. Each sub-domain (i.e. colour on Fig.3) is therefore treated by a different rank. As solving the diffusion equation for the field at one point requires the field in the adjacent points, each subdomain has its own halo points. The communication in x and y direction takes place as follows: for each k-level, the update of the halo points is done according to Fig.4 where the update is illustrated for a domain decomposed into 4 MPI ranks. In the z-direction we assume periodic boundary conditions and thus define the halo-points according to Figure 2. These points are updated at the end of each iteration within each rank.

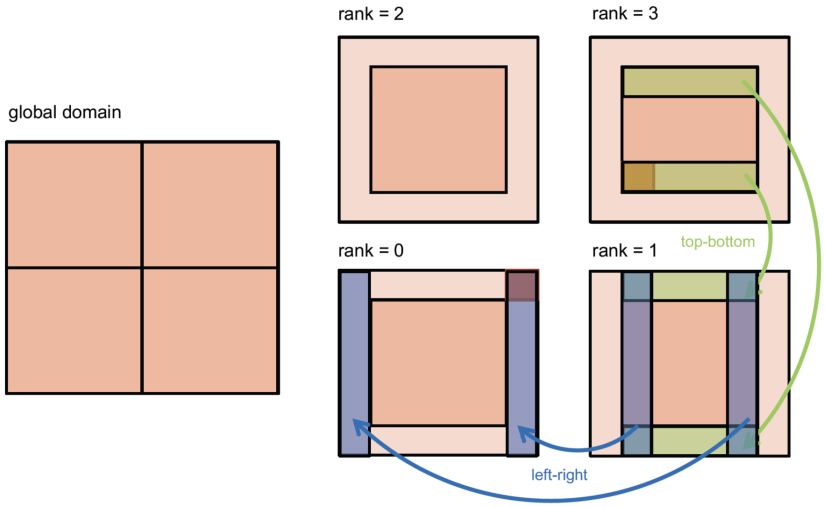

Figure 4: Halo update in x and y direction for one k-level. It is illustrated for a domain decomposed into 4 ranks. Taken from [1] and modified.

3 Performance model

We implement the 2-dimensional halo-update of 3-dimensional fields in Fortran and test the performance on the Piz Daint supercomputer of the Swiss National Supercomputing Center (CSCS) for different setups by varying the number of ranks on two nodes as well as the number of halo-points of the ranks in the x-y plane. Using theoretical considerations we fit a model that is generalized for any domain size to our experiment with a total domain size of $512 \times 512 \times 32$ gridpoints and then test our fitted model on a smaller domain size. We run each combination of number of ranks (between 2 and 24) and number of halo points (between 2 and 14) 10 times and analyse the average runtimes of different components of the code.

3.1 Model

3.1.1 MPI-communication

Our model for the MPI-communication only includes the point-to-point communication of the halo-updates, and neglects the scatter and gather communications, as these are only performed once in the entire program. The model is based on the considerations of [3] and [4] that the time t_{comm} needed for each communication between MPI ranks can be written as

$$t_{comm} = \alpha + b\beta \tag{6}$$

where b denotes the number of Bytes transmitted. $\beta = 1/w$ where w is the bandwidth of communication between ranks and $\alpha = L + 2o$ where L is the latency (the time needed for a word or a short message to travel from its source rank to its target ranks), and o is the overhead, defined as the time needed for a processor to receive or transmit the messages [3].

We approximate the number of gridpoints in the x- and y- direction assigned to each worker to be $n_x/\sqrt{p}$ and $n_y/\sqrt{p}$. This leads to the following number of Bytes being sent and received by each worker:

$$b = 4(\frac{n_x}{\sqrt{p}} + 2(\frac{n_y}{\sqrt{p}} + n_{halo}))2n_{halo}(n_z + 2) \tag{7}$$

where n_{halo} is the number of halo-points and p is the number of workers.

To first order this equation predicts a linear dependence of the number of halo points, which is what we observe in the data. Equation 7 describes the communication time per iteration if we assume that all workers can send and receive one message at the same time. From our measurements we find this assumption to hold true only for small numbers of workers. Figure 5 shows the runtime for the halo update as a function of the number of ranks for 2, 4, 8, 10 and 14 halo points.

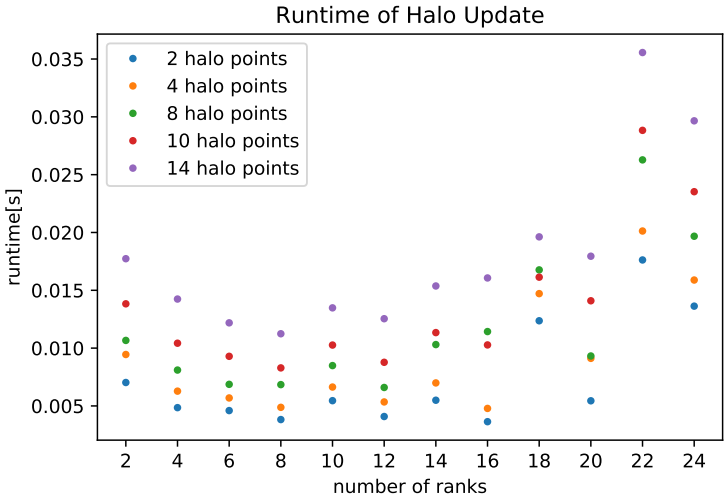

Figure 5: Halo update runtime as a function of the number of ranks for 2, 4, 8, 10 and 14 halo points

We can see that the runtime only decreases with the number of workers up to around 12 ranks and then increases again. However, Equation 6 and 7 only explains a decrease of runtime with increasing number of ranks. In addition we observe an increasing slope and offset of the runtimes as a function of halo points with increasing number of workers. We find that this behaviour is relatively well approximated with a quadratic dependence on p and add p^2 and bp^2 as predictors to our model. We thus conclude that with increasing number of ranks the simultaneous transmission of messages is more and more inhibited and more time is spent in the MPI-Waitall function, as can also be seen in the performance statistics report. We then perform a linear regression with the predictors (1,b,p^2,bp^2) to find estimates for α and β. Figure 6 shows the modelled curves against the measured values for 2, 8 and 14 halo points. We find that our model explains 78% of the variance of the data and a residual analysis shows that the residuals are reasonably normally distributed.

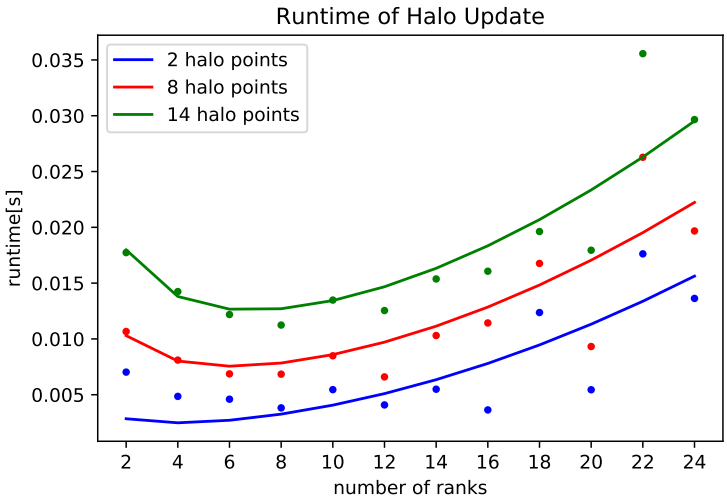

Figure 6: Halo update runtime as a function of the number of ranks for 2, 8, and 14 halo points with fitted curve

3.1.2 Runtime of Diffusion Equation

For the estimate of the runtime of the diffusion equation without the MPI communication we neglect computation times, given that the program is memory-bound and specifically we only take into account latency, as the performance statistics show that the performance is limited by latency. With a field of $512 \times 512 \times 32$, single precision floating point numbers, and given that in each part of the diffusion equation we access the entire field at least once, we find that we are bound by L3 accesses and that we can essentially neglect the effect of different caching for different numbers of workers. In fact, even for smaller field sizes we find that adding the effect of caching does not help to better predict runtimes. Figure 7 shows the total runtime of the program as a function of number of ranks for different numbers of halo points. While for small numbers of ranks the runtime decreases with increasing workers as could be expected due to division of work, the runtime stays approximately constant for numbers of ranks larger than 12, and even increases slightly for certain of the setups. This has to do with two effects: On the one hand the MPI communication becomes increasingly important for an increasing number of ranks, as shown in the left panel of Figure 8. On the other hand if we plot the runtime per gridpoint of the diffusion without the halo update as shown on the right panel of Figure 8, we find that it increases approximately linearly with an increasing number of ranks. We capture this in our model by assuming a fixed overhead for each rank and capture it in the following equation for the runtime of the diffusion:

$$t_{diffusion} = o_{diffusion} + a\frac{n_x n_y n_z}{p} L_{diffusion} \tag{8}$$

where $o_{diffusion}$ denotes an overhead to the computation, a denotes the number of slow accesses, and $L_{diffusion}$ is the upper bound of the latency of memory accesses. Dividing this by the number of gridpoints per rank ($\frac{n_x n_y n_z}{p}$) we find the linear relationship observed

in Figure 8. The two paramaters $o_{diffusion}$ and $L_{diffusion}$ are again determined with linear regression. Figure 9 shows the obtained model curves and the measured values of the runtime of the diffusion equation for different numbers of halo points. In theory it should be possible to estimate $L_{diffusion}$ with the upper threshold of the latency for memory accesses and the amount of memory accesses to the smallest cache that fits the entire field. However, such an estimation does not work for our simple model.

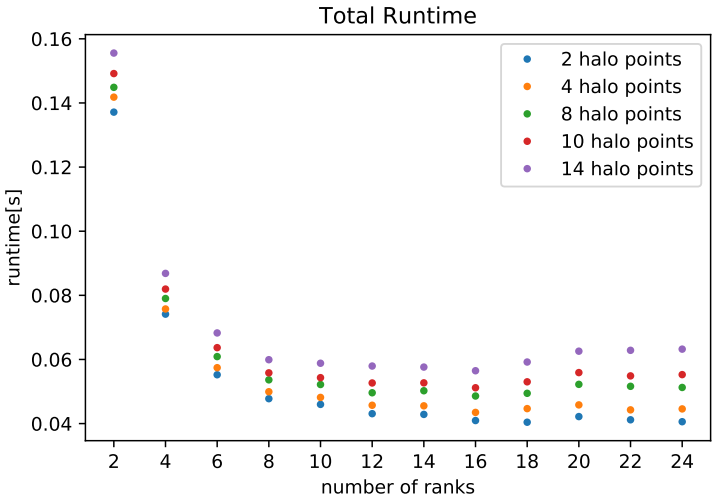

Figure 7: Total runtime as a function of number of ranks for different numbers of halo points

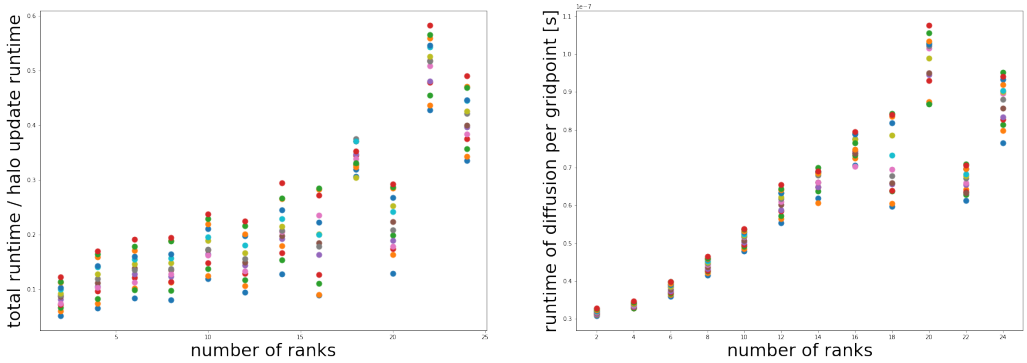

Figure 8: Total runtime of diffusion - halo update runtime, as a function of number of ranks

3.1.3 Validation

Since our model has been specifically tailored for a $512 \times 512 \times 32$ field and in particular the parameters have been fitted to that data, showing that it fits the data for experiments with that field size is not a conclusive validation. We therefore also apply our model with the parameters estimated for $512 \times 512 \times 32$ to a $128 \times 128 \times 32$ field. Figure 10 shows the measured values for this field and the predicted curves. We see that while the character of the dependence is well captured, our model seems to be slightly overfitted to the field it was

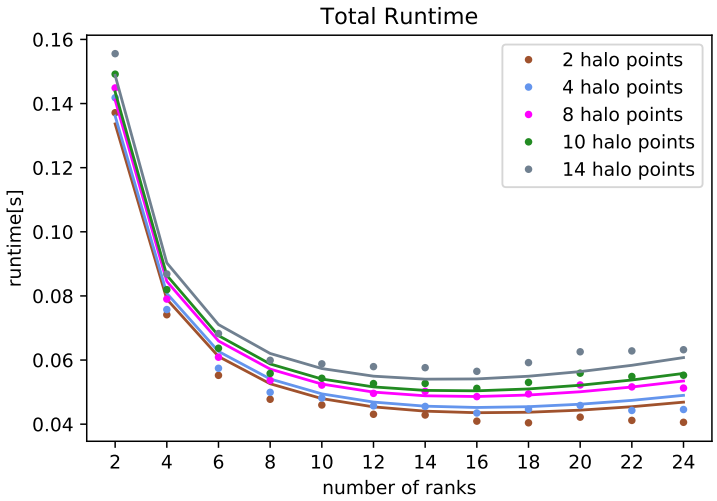

Figure 9: Total runtime as a function of number of ranks for different numbers of halo points and the corresponding modelled values

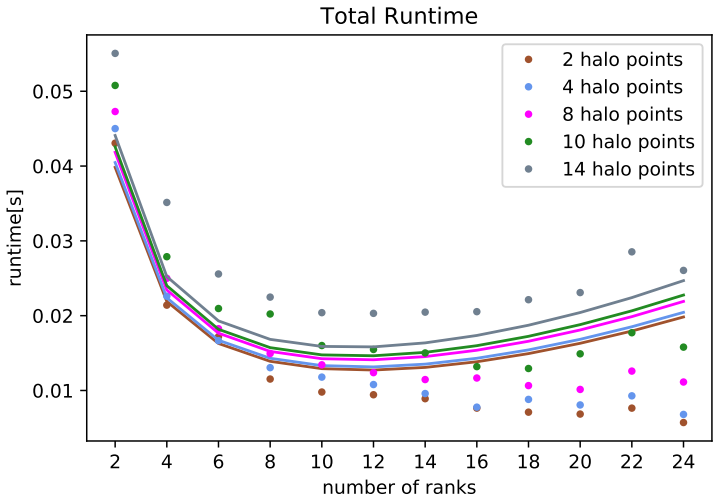

Figure 10: Total runtime as a function of number of ranks for different numbers of halo points for a 128 x 128 x 32 field and the corresponding modelled values

estimated on. In particular the halo-update runtime was estimated with four predictors that were chosen with the data in mind, and which in addition were correlated.

4 Conclusion

We have constructed a model that predicts the runtime of a 3-dimensional implementation of the diffusion equation on a 2-dimensional worker grid based on the number of workers, the number of halo-points and the total field size. While we expect our model to capture the most relevant theoretical aspects contributing to the program's performance, our model

is strongly based on our specific experiment and the domains (of the field as well as the MPI-parameters) analysed in it. It is therefore useful to understand the time-consuming processes but should only be extrapolated to other domains carefully. A possible extention of this analysis would be to further investigate the performance for different field sizes and for a larger amount of nodes and ranks. Further, it would be interesting to determine a range of parameters where caching effects become more important.

References

[1] High Performance Computing for Weather and Climate (Lecture Slides), Tobias Wicky & Oliver Fuhrer, *ETHZ*, Spring semester 2022

[2] High-Order Monotonic Numerical Diffusion and Smoothing, MING XUE, *Center for Analysis and Prediction of Storms, University of Oklahoma, Norman, Oklahoma*, 17 September 1999

[3] Culler, David Karp, Richard Patterson, David Sahay, Abhijit Santos, Eunice Schauser, Klaus Subramonian, Ramesh von Eicken, Thorsten. (1993). LogP: towards a realistic model of parallel computation. 10.1145/155332.155333.

[4] Thakur R, Rabenseifner R, Gropp W. Optimization of Collective Communication Operations in MPICH. The International Journal of High Performance Computing Applications. 2005;19(1):49-66. doi:10.1177/1094342005051521