Geosci. Model Dev., 15, 9157–9176, 2022
https://doi.org/10.5194/gmd-15-9157-2022

Geoscientific Model Development
Open Access
EGU

Improving scalability of Earth system models through coarse-grained component concurrency – a case study with the ICON v2.6.5 modelling system

Leonidas Linardakis[1], Irene Stemmler[2,a], Moritz Hanke[3], Lennart Ramme[1], Fatemeh Chegini[1], Tatiana Ilyina[1], and Peter Korn[1]

[1]Max Planck Institute for Meteorology, Hamburg, Germany
[2]wobe-systems GmbH, Kiel, Germany
[3]Deutsches Klimarechenzentrum, Hamburg, Germany
[a]previously at: Max Planck Institute for Meteorology, Hamburg, Germany

Correspondence: Leonidas Linardakis (leonidas.linardakis@mpimet.mpg.de)

Received: 29 August 2022 – Discussion started: 15 September 2022
Revised: 25 November 2022 – Accepted: 4 December 2022 – Published: 21 December 2022

Abstract. In the era of exascale computing, machines with unprecedented computing power are available. Making efficient use of these massively parallel machines, with millions of cores, presents a new challenge. Multi-level and multi-dimensional parallelism will be needed to meet this challenge.

Coarse-grained component concurrency provides an additional parallelism dimension that complements typically used parallelization methods such as domain decomposition and loop-level shared-memory approaches. While these parallelization methods are data-parallel techniques, and they decompose the data space, component concurrency is a function-parallel technique, and it decomposes the algorithmic space. This additional dimension of parallelism allows us to extend scalability beyond the limits set by established parallelization techniques. It also offers a way to maintain performance (by using more compute power) when the model complexity is increased by adding components, such as biogeochemistry or ice sheet models. Furthermore, concurrency allows each component to run on different hardware, thus leveraging the usage of heterogeneous hardware configurations.

In this work we study the characteristics of component concurrency and analyse its behaviour in a general context. The analysis shows that component concurrency increases the "parallel workload", improving the scalability under certain conditions. These generic considerations are complemented by an analysis of a specific case, namely the coarse-grained concurrency in the multi-level parallelism context of two components of the ICON modelling system: the ICON ocean model ICON-O and the marine biogeochemistry model HAMOCC. The additional computational cost incurred by the biogeochemistry module is about 3 times that of the ICON-O ocean stand alone model, and data parallelization techniques (domain decomposition and loop-level shared-memory parallelization) present a scaling limit that impedes the computational performance of the combined ICON-O–HAMOCC model. Scaling experiments, with and without concurrency, show that component concurrency extends the scaling, in cases doubling the parallel efficiency. The experiments' scaling results are in agreement with the theoretical analysis.

1 Introduction

Since the dawn of modern computing, numerical weather prediction and climate modelling have been among the first scientific applications to make use of the new technology (Dalmedico, 2001; Washington et al., 2009; Balaji, 2013). In the decades following the creation of the first atmosphere computer models, computational power has been increasing exponentially (McGuffie and Henderson-Sellers, 2001), allowing for the development of complex Earth system mod-

els (Randall et al., 2018) running at ever higher resolutions. A first impressive result was described by Miyamoto et al. (2013), where the atmosphere model NICAM ran in a global sub-kilometre resolution for 12 simulated hours, dynamically resolving convection. Since then, other groups have followed the path of reducing parameterizations by increasing the resolution, as, for example, in global storm resolving setups described in Stevens et al. (2019). We are currently viewing the perspective of constructing the Earth's digital twin (Voosen, 2020; Bauer et al., 2021), where much of the Earth's system complexity will be captured by models at 1 km resolution.

These developments have been made possible by the availability of massively parallel computers. Since the beginning of the 21st century the focus of CPU development has switched from constructing more powerful processing units to packing more units into an integrated chip. In response, programmers had to turn much of their efforts from optimizing the code to efficiently parallelizing it (Sutter, 2005; Mattson et al., 2008). The era of exascale computing is here with the construction of machines like the Frontier at the Oak Ridge National Laboratory. While less than 15 years ago we were facing the challenge of petascale computing (Washington et al., 2009), we are now facing a new level of challenge: how to efficiently parallelize our codes for machines with millions of cores.

The parallelization backbone of Earth system models consists of domain decomposition techniques, where the horizontal grid is decomposed into subdomains, which are assigned to different processing units, and the Message Passing Interface (MPI; Walker, 1992; The MPI Forum, 1993) is used to communicate information between them. This approach has been designed primarily for distributed memory parallelization. In the last years it has become apparent that domain decomposition methods alone cannot efficiently scale when using a high number of cores placed on a shared memory board. For 2 decades, shared-memory parallelization mechanisms, such as OpenMP (Mattson, 2003), have been being developed. These have been increasingly employed for providing loop-level shared-memory parallelization, in order to exploit the new multi-core architectures. More recently, GPUs have attracted a lot of attention due to the high computing power they provide through massive parallelism while at the same time requiring lower power consumption per FLOP (floating point operations per second) than traditional CPUs. The two levels of parallelization that are currently widely used, domain decomposition with MPI for distributed memory parallelization and OpenMP shared-memory loop-level parallelization (or similar approaches, like OpenACC for GPUs), have so far been successful in yielding satisfactory performance on parallel machines. They still pose though some limitations, as their scaling efficiency typically depends on the number of grid points available for parallelization.

The concept of concurrency goes back to before parallel computing came into practice (Lamport, 2015). It refers to algorithmic dependencies and independencies and was first developed in the context of multitasking. The term has come to be synonymous to task parallelism, as independent tasks can run in parallel. In contrast to the domain decomposition and loop-level parallelization methods, concurrency is a function (or task) parallel approach. Coarse-grained component concurrency is a special case of concurrency, where the independent components are large model modules, essentially sub-models, with comparable computational workload. It has been used in climate modelling for decades in atmosphere–ocean coupled setups: the two models run in parallel and are coupled every one or more time steps. More recently, the same idea has been applied to the radiation component of the atmosphere (Mozdzynski and Morcrette, 2014; Balaji et al., 2016) and to the radiation and ocean wave components (Mozdzynski, 2018). A concurrency approach by splitting the atmosphere dynamics and physics processes is presented in Donahue and Caldwell (2020).

Increasing the grid resolution allows us to resolve smaller scales, to better approximate the physical processes, and to rely less on parameterizing unresolved processes. This increased problem size can still be effectively parallelized using data parallelism, at least up to a point. On the other hand, there is interest to include more processes into Earth system models, in order to have a more detailed representation of the Earth system. Such processes may represent the atmosphere chemistry, the cryosphere, and the ocean biogeochemistry and can have a significant impact on the Earth's climate and the biosphere.

Ocean biogeochemistry comprises a variety of chemical and biological processes in the water column and the sediments of the ocean (Sarmiento and Gruber, 2006). These processes include, for example, biological activity of phytoplankton, zooplankton and different types of bacteria, the chemical and biological cycles of carbon or nitrogen, and the dissolution of gases in seawater. Ocean biogeochemistry is therefore an important component for quantifying critical developments in the Earth system, like the oceanic uptake of anthropogenic CO_2 released from fossil fuels (Ciais et al., 2014) or ocean acidification and deoxygenation under the impact of global warming (Orr et al., 2005; Breitburg et al., 2018). However, the number of processes that could be included is extensive, and ocean biogeochemistry models are becoming ever more complex through the addition of more tracers and processes (Ilyina et al., 2013). This results in a large computational cost, which hinders their integration in high-resolution Earth system models, especially when simulating the long timescales that are crucial to investigate changes in the biogeochemical state of the ocean.

In contrast to increasing the grid size, the additional computational cost imposed by introducing new processes cannot be absorbed through grid decomposing parallel methods, as these are limited by the grid size. Component concurrency offers a way to increase the model complexity while maintaining reasonable performance.

In this paper we study the impact of component concurrency on the scaling behaviour of a model. We examine it in a general abstract context, in what manner component concurrency differs from the more traditional approaches and what its scaling characteristics are. We consider concurrency to be part of a multi-level parallelism scheme, and we examine the cases where it can improve performance and when this improvement is optimal. The ICON-O–HAMOCC ocean biogeochemistry model offers a good test case, as it is about 4 times slower than ICON-O alone. We have run two sets of experiments on two different machines. The scaling results obtained from the experiments are in good agreement with the predictions from the theoretical analysis.

In Sect. 2 we give a brief description of the ICON models. In Sect. 3 we examine the behaviour of component concurrency in a general context. Section 4 describes the basic steps to engineer concurrency for ICON-O–HAMOCC. Experiments and the results are presented in Sect. 5. A comparison of the experimental results with the theoretical analysis is given here. In Sect. 6 we give an overview of the results and future work.

2 ICON model description

ICON is an Earth system model framework developed in collaboration with the German Weather Service (DWD), the Max Planck Institute for Meteorology (MPIM), the Institute of Meteorology and Climate Research at the Karlsruhe Institute of Technology, and the German Climate Computing Centre (DKRZ). It consists of the numerical weather prediction model ICON-NWP (Zängl et al., 2015), the climate atmosphere model ICON-A (Giorgetta et al., 2018; Crueger et al., 2018), the land model JSBACH (Nabel et al., 2020), the ocean model ICON-O (Korn et al., 2022), the atmosphere aerosol and chemistry model ICON-ART (Rieger et al., 2015), and the marine biogeochemistry model HAMOCC (Ilyina et al., 2013). The ICON Earth system model ICON-ESM consists of ICON-A, JSBACH, ICON-O, and HAMOCC (Jungclaus et al., 2022).

The ICON horizontal grid consists of triangular cells constructed by recursively dividing the icosahedron (Tomita et al., 2001), and it provides near-uniform resolution on the sphere. More general non-uniform triangular grids can be used by ICON-O (Logemann et al., 2021).

The ICON framework provides common infrastructure to its components. It supplies the domain decomposition routines and model-contextual high-level communication interfaces to the Message Passing Interface (MPI). It also provides flexible interfaces for input and automatic parallel asynchronous output mechanisms. The YAC library (Hanke et al., 2016) serves as a general coupler between models. Models are registered to a simple master control module through namelists. The ICON models employ both domain decomposition and OpenMP loop-level parallelism. ICON-A and JSBACH can also run on GPUs using OpenACC directives (Giorgetta et al., 2022).

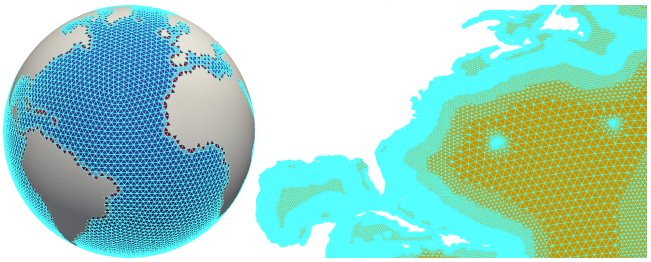

Figure 1. Left: the uniform ocean icosahedron-based grid at 160 km resolution. Right: detail from the non-uniform global coastal ocean grid with resolution 8–80 km used in Mathis et al. (2021).

2.1 The ICON-O ocean model

ICON-O is the ocean general circulation model that provides the ocean component to the ICON-ESM. Its horizontal spatial discretization is based on unstructured triangular grids, allowing for a variety of setups, from idealized basins (Korn and Danilov, 2017) to global ocean domains, where the interior land points are removed; see Fig. 1 left. In non-uniform setups, it has been tested with "telescoping" setups (Korn et al., 2022), which can produce local grid spacings of 600 m (Hohenegger et al., 2022). Furthermore, a topographic and coastal adaptive local refinement (Logemann et al., 2021) has been used for global coastal ocean simulations (see Fig. 1 right). This setup includes the HAMOCC biogeochemistry model (see Mathis et al., 2021).

ICON-O solves the oceanic hydrostatic Boussinesq equations, also referred to as the "primitive equations". The primitive equations are solved on the triangular ICON grid with an Arakawa C-type staggering, using a *mimetic* horizontal discretization, where certain conservation properties of the continuous formulation are inherited to the discretized one. The staggering necessitates reconstructions to connect variables that are located at different grid positions. This is accomplished in ICON-O by utilizing the novel concept of *Hilbert space admissible reconstructions*; for details see Korn (2017) and Korn and Linardakis (2018).

The vertical coordinate axis is given by the z coordinate, which reflects the geopotential height. The two-dimensional triangles are extended by a height-based dimension, which generates three-dimensional prisms. Alternative vertical coordinates such as the z^* coordinate are also available in ICON-O.

ICON-O is stepping forward in time with a semi-implicit Adams–Bashforth-2 scheme. The free surface equation is solved implicitly in time, using an iterative conjugate gradient solver. The remaining state variables are discretized explicitly. For details we refer to Korn (2017).

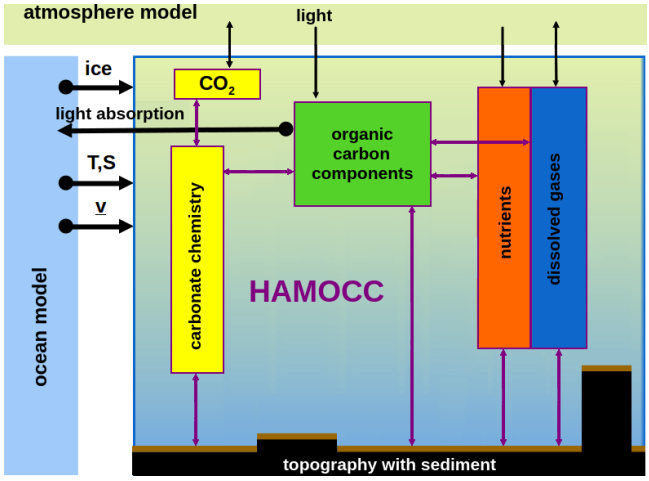

Figure 2. Schematic of the biogeochemical processes simulated by HAMOCC. HAMOCC needs to be coupled to an ocean model that provides the fields of temperature, salinity, and sea-ice cover and transports the HAMOCC tracers according to the flow field. The surface inputs can either come from a coupled atmosphere model or are prescribed. The biogeochemistry in HAMOCC can then feed back to the ocean model via the impact of light absorption on temperature and to the atmosphere model via the uptake or release of CO_2.

The ICON sea-ice model consists of a dynamic and a thermodynamic component. The sea-ice dynamics follows the elastic–viscous–plastic (EVP) rheology formulation and is based on the sea-ice dynamics component of FESIM; see Danilov et al. (2015). The thermodynamics of the sea ice follow the zero-layer formulation; see Semtner (1976).

A combination of a first-order upwind scheme and a second-order scheme are utilized for the horizontal tracer transport. The second-order method flux calculations are based on compatible reconstructions, as described in Korn (2017). The two schemes are combined through a Zalesak limiter (Zalesak, 1979), resulting in a "flux-corrected transport", which avoids the creation of new extrema (over-/undershoots). This combination results in both monotonicity and low numerical diffusion, which are essential for preserving the water density structure.

For the vertical tracer transport we use a combination of the piecewise parabolic method (PPM; see Colella and Woodward (1984)) as a high-order and upwind as a low-order method.

2.2 The HAMOCC biogeochemistry model

The Hamburg Ocean Carbon Cycle (HAMOCC) model has initially been developed in earlier work by Maier-Reimer (1984) and Maier-Reimer and Hasselmann (1987) to address the role of ocean processes driving the fate of carbon in the climate system over timescales ranging from seasons to thousands of years. To achieve a consistent evolution of the ocean biogeochemistry, the biogeochemical variables are handled as tracers on the three-dimensional grid of the ocean general circulation model. They are transported in the same manner, i.e. using the same numerical methods and time step, as salinity and temperature.

The processes simulated by HAMOCC include biogeochemistry of the water column and upper sediment, as well as interactions with the atmosphere. Figure 2 shows a schematic overview of the key components of the HAMOCC model. In the water column, the biogeochemical tracers undergo modifications by biological and chemical processes, described in detail in Ilyina et al. (2013) and Paulsen et al. (2017). At the air–sea interface, the fluxes of O_2, N_2, and CO_2 are calculated. Furthermore, dust and nitrogen deposition from the atmosphere to the ocean is accounted for. The simulation of the oceanic sediment follows the approach of Heinze et al. (1999), and biogeochemical tracers are exchanged with the upper sediment.

Marine biology dynamics connects biogeochemical cycles and trophic levels of the marine food web through the uptake of nutrients and remineralization of organic matter. It is represented by the extended NPZD approach with nutrients, i.e. dissolved inorganic nitrogen (N), phytoplankton (P), zooplankton (Z), and detritus (D) (sinking particulate matter) and also dissolved organic matter (Six and Maier-Reimer, 1996). Explicit fixation of nitrogen is performed by cyanobacteria (Paulsen et al., 2017). All organic compounds have identical nutrient and oxygen composition following the Redfield ratio concept, extended by a constant ratio for carbon and the micronutrient iron. The treatment of carbon chemistry follows the guide to best practices, as described in Dickson et al. (2007) and Dickson (2010).

The transport of biogeochemical tracers presents the most expensive computational part of the HAMOCC model. The number of advected tracers depends on the complexity of the included processes. For example, including organic matter from riverine or terrestrial sources (Lacroix et al., 2021), extending the nitrogen cycle by including ammonium and nitrite, simulating carbon isotopes or using a more realistic sinking method for particular organic matter (M4AGO scheme: Maerz et al., 2020), or incorporation of stable carbon isotope ^{13}C (Liu et al., 2021) increase the number of advected tracers from the default value of 17 and therefore increase the computational cost. Introducing concurrency enables the use of currently simulated processes and may allow for the addition of even more tracers, necessary for including more processes, while maintaining an acceptable throughput.

3 Coarse-grained component concurrency and multi-dimensional parallelism

In a coarse component concurrent setup, two or more components of the model are run in parallel. The level of "coarseness" is difficult to define; here we will understand it as being

able to run concurrently the components throughout a whole time step. The components are algorithmically independent and may only need to receive input data from other components once in each time step. For example, the ocean model in a coupled setup requires the atmosphere surface fluxes at the start of each time step and then can proceed independently from the atmosphere model. Thus, we expect only one point of communication between the components, where all the information is exchanged. Such components may be the radiation, the ocean biogeochemistry, the sea ice, and the ice sheets, etc.

From here on, we will use the term "concurrency", instead of coarse-grained component concurrency, for brevity. We will also use the term "sequential" as a synonym for "non-concurrent", in the sense that the components run sequential to each other; other types of parallelization though may still be present.

A schematic of concurrency is drawn in Fig. 3. Let A and B be two components of the model. In the case of using domain decomposition parallelism only, the domain is decomposed, and the subdomains are distributed among the processing units, while the two modules run sequential to each other, as in Fig. 3a. In the case of concurrency the two modules run on two different groups of processing units, depicted in Fig. 3b.

3.1 Levels of parallelism

We can identify three levels of parallelism:

a. *High-level* parallelism is applied over the whole model or over the whole concurrent components of the model. The most successful such technique is to decompose the horizontal grid and use MPI for communicating between the subdomain processes. Component concurrency falls into this category. The most well-known example of concurrency in climate modelling is running the atmosphere and the ocean models concurrently and coupling them every one or more time steps. An important characteristic of high-level parallelization is that it is independent of the machine architecture. It can be applied across nodes of heterogeneous machines and can facilitate hybrid setups, by running simultaneously on different types of processing units.

b. In *medium-level* parallelism we identify parallel structures on a task or loop level. These are shared-memory parallelization techniques, such as OpenMP or OpenACC. We can consider them to be "medium- or fine-grained" parallelism. In contrast to high-level parallelization, the implementation of this level of parallelization may not be independent from the type of architecture.

c. In *low-level* parallelism we identify techniques closer to the architecture, such as vectorization and out-of-order execution. These techniques depend on the particular architectures and will not be considered in the following discussion.

Another way to characterize parallelism is by the type of decomposition that is employed. In *data parallelism* the data domain is decomposed, and the same operations are applied to each sub-domain. Examples of data parallelism are the domain decomposition techniques and the loop-level parallelism. In the *function (or task) parallelism*, the algorithmic space is "decomposed". Examples are OpenMP task parallelism, out-of-order execution, and also the coarse-grained component concurrency. So, these two types of decomposition exist across the three levels of parallelism.

Domain decomposition parallelism comes with a communication and synchronization cost, typically caused from exchanging values of "halo" cells between processes. These halo cells consist of the boundary of subdomains which are replicated by their neighbour subdomains (see Fig. 4). The total number of halo cells generally increases proportional to $\sqrt{N}$, where N is the number of subdomains. In turn, the parallelization gain for halos is only proportional to $\sqrt{N}$, instead of N. This imposes a limit on how far we can use only domain decomposition as a parallel paradigm for running on massively parallel machines. We will further examine this behaviour in the experiments in Sect. 5.1.

Another type of communication is global reduction operators, such as global sums. These are typically used in matrix inversions, as is the case for ICON-O, and they also impose scaling limits. We will also observe this behaviour in the experiments in Sect. 5.2.

OpenMP parallelization provides complementary advantages to the domain decomposition. It offers dynamic load balancing and no communication cost.[1] On the other hand, performance is restricted by overheads, memory bandwidth, and, in NUMA (non-uniform memory access), machines by data locality. This last disadvantage is alleviated in the domain decomposition approach due to a smaller memory footprint per process.

Component concurrency also comes with a cost, which depends on how it is implemented. If we keep the total MPI tasks constant, equal to N, we have two options. In the case of a distributed memory parallelization, we split the MPI tasks among the two components, assigning N_1 to the first and N_2 to the second, and apply shared-memory parallelization inside each component. The cost comes from communicating between the two components.

In the case of a shared-memory implementation of concurrency, both components will run on N MPI tasks. In this case we do not have a communication cost, but the synchronization cost remains. Moreover, the performance will partially depend on how efficiently MPI can handle multiple commu-

[1] Here by "communication" direct communication between the parallel tasks is meant. The cost of it can be significant when it takes place through the network.

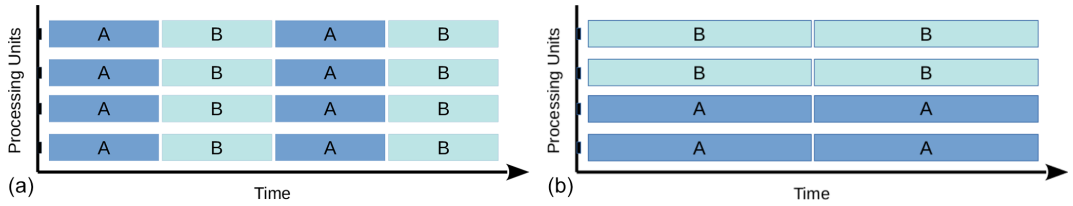

Figure 3. **(a)** Two model components A and B running on four processing units using only domain decomposition parallelism. **(b)** The two components run concurrently, each on 2 processing units.

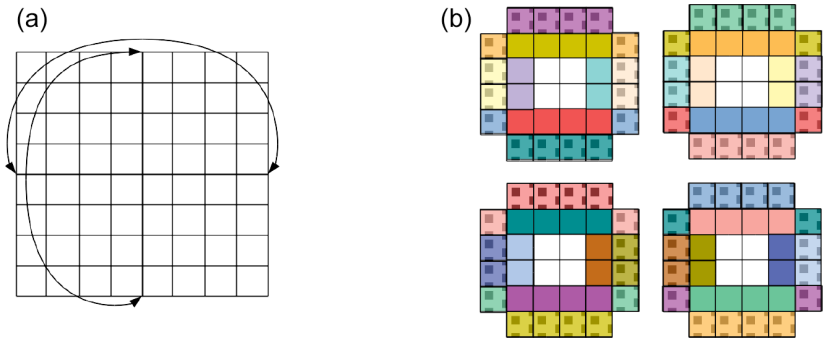

Figure 4. **(a)** A domain of 64 grid points. It is assumed double-periodic for illustration purposes. **(b)** The domain decomposed in four subdomains. Grid points with the same colour are duplicated, with halos depicted with shading.

nicators from the same MPI task concurrently. Other aspects may also prove to be significant, like input and output (I/O). In the case of distributed memory implementation, all infrastructure is automatically also distributed, including output.

The other aspect of these two options is the implementation. The distributed memory case is independent of the architecture and can even be applied in hybrid mode. While in principle the same functionality can be achieved using shared-memory parallelization, no such standard, to the authors' knowledge, is currently mature enough to be implemented across multiple architectures. Weighting the pros and cons can only be done in some context. In this work we choose to implement component concurrency as a distributed memory approach due to the flexibility it offers.

3.2 Coarse-grained component concurrency and scalability

For climate models the total computing workload is proportional to the grid size[2] and the number of operations per grid point required to solve the problem. We have $W_{\mathrm{T}} = a \cdot s$, where W_{T} is the total workload, s is the grid size, and a is the number of operations per grid point. We define the *parallel workload* as the workload inside a parallel region, that is between two synchronization points, assigned on one processing unit. For example, the total workload inside an OpenMP parallel loop, divided by the number of OpenMP threads, would constitute a parallel workload. Such a parallel region contains a constant number of operations a_{p} per grid point[3], so we have

$$W_{\mathrm{p}} = a_{\mathrm{p}} \cdot s/N.$$

Let A and B be two modules of the model using the same grid (as in Fig. 3). Let W_A be the total workload of module A and $W_B = \lambda \cdot W_A$ the total workload of module B. Proportionally, let $N_A = N$ be the number of processing units that A runs on its own and $N_B = \lambda \cdot N$ the additional number of processing units we use when adding module B. The total number of processing units is now $N_A + N_B$. Let us consider the parallel workload as the workload assigned to a parallel loop. In the typical data parallel case, where the grid space is decomposed, the parallel workload of a parallel region is $W_{AB\mathrm{p}} = a_{\mathrm{p}} \frac{s}{N_A+N_B}$, independently if this parallel region belongs to module A or B. In the concurrent case, where module A runs on N_A units and module B on N_B units, the parallel workload of this parallel region is $W_{A\mathrm{p}} = a_{\mathrm{p}} \frac{s}{N_A}$ if it belongs to module A and $W_{B\mathrm{p}} = a_{\mathrm{p}} \frac{s}{N_B}$ if it belongs to B. In both cases concurrency increases the parallel workload compared to data parallelism only. This is a main feature of concurrency compared to data parallelism; it provides another parallelism dimension by decomposing the function space a instead of the problem size s.

[2]The total 3-dimensional grid size.

[3]The number of operations per horizontal grid point may vary depending on conditionals and number of active vertical levels. These differences would create imbalance. Without loss of generality we can take the maximum workload among processes.

How does increasing the parallel workload affect the total performance? If we ignore the scaling issues, there is no effect. In the non-concurrent case the time to solution is proportional to $(W_A + W_B)/(N_A + N_B) = ((1+\lambda)\cdot W_A)/((1+\lambda)\cdot N) = W_A/N$, while in the concurrent case it is $W_A/N = (\lambda \cdot W_A)/(\lambda \cdot N) = W_B/N_B$. The performance is the same. Only when scaling is taken into account does concurrency have an impact. We will examine this impact in the following discussion.

Let $T(1) = r \cdot W_A = r \cdot a \cdot s$ be the time it takes to run module A on one processing unit, where r is a constant that characterizes the computing power of the processing unit and W_A the workload. We will only consider the homogeneous case, where all units have the same processing power. Let $T(N)$ be the time for running on N processing units. The *speed-up* is defined as $S(N) = T(1)/T(N)$ and the parallel *efficiency* as $F(N) = T(1)/N/T(N) = S(N)/N$. We have

$$T(N) = T(1)/(F(N)\cdot N).$$

Let us now add another component B to the model A as above that adds a workload of $W_B = \lambda \cdot W_A$, increasing the time cost for running on one processing unit to $T(1)\cdot(1+\lambda)$. We increase the number of processing units proportionally, from N to $N(1+\lambda)$. We will assume that our new component B has the same scaling behaviour as A, so that the same efficiency function $F(N)$ also applies to B. When using data decomposition parallelization, our new time cost is

$$\begin{aligned} T_\mathrm{d}(N\cdot(1+\lambda)) &= \frac{T(1)\cdot(1+\lambda)}{F(N\cdot(1+\lambda))\cdot N\cdot(1+\lambda)} \\ &= \frac{T(1)}{F(N\cdot(1+\lambda))\cdot N}. \end{aligned}$$

When on the other hand we run component B concurrently on $\lambda\cdot N$ nodes, with A on N nodes, then the total time cost is

$$T_\mathrm{c}(N\cdot(1+\lambda)) = \max(T_{A\mathrm{c}}, T_{B\mathrm{c}}),$$

where $T_{A\mathrm{c}} = \frac{T(1)}{F(N)\cdot N} + C(N\cdot(1+\lambda))$, $T_{B\mathrm{c}} = \frac{T(1)\cdot\lambda}{F(N\cdot\lambda)\cdot\lambda\cdot N} + C(N\cdot(1+\lambda)) = \frac{T(1)}{F(N\cdot\lambda)N} + C(N\cdot(1+\lambda))$ are the time costs for running components A and B concurrently, and $C(N\cdot(1+\lambda))$ is the cost incurred by the concurrency.

We assume that the parallel efficiency is a non-increasing function of N, that is $F'(N) \le 0$[4]. Without loss of generality we take $\lambda \le 1$ (in the opposite case we can just swap the modules A and B). Then $F(N\cdot\lambda) \ge F(N)$ and $T_\mathrm{c} = T_{A\mathrm{c}}$. Comparing T_c with T_d, we have

$$\begin{aligned} \frac{T_\mathrm{c}}{T_\mathrm{d}} &= \frac{\frac{T(1)}{F(N)\cdot N} + C(N\cdot(1+\lambda))}{\frac{T(1)}{F(N\cdot(1+\lambda))\cdot N}} = \frac{F(N\cdot(1+\lambda))}{F(N)} \\ &\quad + \frac{C(N\cdot(1+\lambda))\cdot F(N\cdot(1+\lambda))\cdot N}{T(1)}. \end{aligned}$$

[4]We take the liberty to consider N continuous whenever needed.

We set

$$L(N,\lambda) = \frac{F(N\cdot(1+\lambda))}{F(N)},$$

termed *relative efficiency*. We have

$$\frac{T_\mathrm{c}}{T_\mathrm{d}} = L(N,\lambda) + \frac{C(N\cdot(1+\lambda))\cdot L(N,\lambda)\cdot F(N)\cdot N}{T(1)}.$$

Taking into account that $T(N) = \frac{T(1)}{F(N)\cdot N}$, we have

$$\begin{aligned} \frac{T_\mathrm{c}}{T_\mathrm{d}} &= L(N,\lambda) + L(N,\lambda)\cdot\frac{C(N\cdot(1+\lambda))}{T(N)} \\ &= L(N,\lambda)\cdot\left(1 + \frac{C(N\cdot(1+\lambda))}{T(N)}\right). \end{aligned} \tag{1}$$

We seek the conditions where $\frac{T_\mathrm{c}}{T_\mathrm{d}}$ is smaller than 1 and as small as possible. In a linear or near-linear scaling regime, where the efficiency $F(N)$ is nearly constant as a function of N, we have $L(N,\lambda) \approx 1$, and concurrency will provide little, if any, benefit.

Let us examine the sub-linear scaling regime. Then $F(N)$ is a strictly decreasing function of N, and $L(N,\lambda) < 1$. Moreover $L(N,\lambda)$ is a strictly decreasing function of λ; when we keep N constant, we have

$$\frac{\partial L(N,\lambda)}{\partial\lambda} = \frac{1}{F(N)}\frac{\partial F(N\cdot(1+\lambda))}{\partial\lambda} < 0.$$

Concurrency will provide the maximum benefits when λ is maximum, that is $\lambda = 1$ (recall that $\lambda \le 1$), and the two modules have the same workload. On the other hand, if $\lambda \ll 1$, we have $L(N,\lambda) \approx 1$, and the benefits would be significantly reduced. In this case the bulk of the workload is borne by module A, and the additional parallelism for B provides little profit. The "coarse-grained" part of the concurrency does not hold, and one should consider using fine-grained parallelism.

The sub-linear scaling property is not sufficient to allow us to deduce the behaviour of $L(N,\lambda)$ as a function of N. We will further assume that the scaling behaviour follows Amdahl's law (Amdahl, 1967). In this case $T(N) = T(1)\frac{(1-\sigma)+\sigma\cdot N}{N}$, where $0 < \sigma < 1$ is the part of the code that does not scale. Then $S(N) = \frac{T(1)}{T(N)} = \frac{N}{(1-\sigma)+\sigma\cdot N}$, $F(N) = \frac{S(N)}{N} = \frac{1}{(1-\sigma)+\sigma\cdot N}$, and $L(N,\lambda) = \frac{(1-\sigma)+\sigma\cdot N}{(1-\sigma)+\sigma\cdot N\cdot(1+\lambda)}$. We have

$$\begin{aligned} &\frac{\partial L(N,\lambda)}{\partial N} = \\ &\frac{\sigma[(1-\sigma)+\sigma\cdot N\cdot(1+\lambda)] - \sigma\cdot(1+\lambda)\cdot[(1-\sigma)+\sigma\cdot N]}{[(1-\sigma)+\sigma\cdot N\cdot(1+\lambda)]^2} \\ &= \frac{-\sigma\cdot\lambda\cdot(1-\sigma)}{[(1-\sigma)+\sigma\cdot N\cdot(1+\lambda)]^2} < 0. \end{aligned}$$

L in this case is a decreasing function of N, with a lower limit $L_\mathrm{l} = \frac{1}{1+\lambda}$, which provides yet another piece of evidence of the optimality of $\lambda = 1$.

Experiment results show that in the case of ICON-O–HAMOCC, L is in general a decreasing function of N (see Sect. 5), and concurrency is effective only after a scaling threshold has been reached. Scaling tests for the ICON-A also indicate that $L(N, \lambda)$ is a decreasing function of N; see Giorgetta et al. (2022), Table 3.

Let us now examine the concurrency communication cost C. The load of the point-to-point communication is proportional to $1/N$ unlike to the halo communication cost, which is proportional to $1/\sqrt{N}$. So we do not expect the ratio $C(N \cdot (1+\lambda))/T(N)$ to change significantly as a function of N, but load imbalance, interconnect, and latency costs may influence it. In some cases concurrency may also require a halo exchange (this is the case for ICON-O–HAMOCC). Concurrency, obviously, will perform better when the relative cost of the communication C to the computational cost T_A is small. If we take $C(N \cdot (1+\lambda))/T(N) = c$, constant then $\frac{T_c}{T_d} = L(N, \lambda) \cdot (1+c)$, which implies that the scaling threshold $L(N, \lambda) < \frac{1}{1+c}$ has to be reached before concurrency is effective.

The case of super-linear scaling is rare but not unknown. Typically it occurs on cache-based architectures, when a smaller memory footprint allows for more efficient use of the cache memory. We can still use Eq. (1) to deduce some conclusions (we can switch modules A and B if necessary). In this regime we have $F(N_1) < F(N_2)$ for $N_1 < N_2$, and $L(N, \lambda) > 1$; so concurrency will result in worse performance.

Finally we examine the case of "flattening" scaling, the limit after which the speed-up does not increase. This can be the case for massively multi-core architectures, like GPUs, when the workload per node is not enough to occupy the computing units, and resources are idling. Let N be the number of nodes beyond which scaling does not increase. Then, from the previous analysis, we see that in the concurrent case scaling can still be increased up to $N \cdot (1+\lambda)$ nodes but not beyond this. This in essence underlines the fact that concurrency increases the parallel workload when compared to data parallelism.

4 Engineering concurrency for the ICON-O–HAMOCC model

Constructing coarse-grained component concurrency is a software engineering task. The candidate components have to present "natural" concurrency; this in practice means that they will present one communication point between them, while the algorithmic part of the components can run independently. Having identified the two components, the next steps are as follows:

a. encapsulating the components,

b. creating an interface between them, and

c. providing the necessary infrastructure for the two components to run independently and to communicate.

The procedure is not dissimilar to that of constructing standalone models, as described, for example, in Eastham et al. (2018), or constructing coupled setups, as described in Long et al. (2015).

The original structure and workflow of the sequential ICON-O HAMOCC process are sketched in Fig. 5a. The call to the HAMOCC biogeochemistry takes place just before the tracer transport is called (we will use the term "transport" for brevity). Upon returning, the HAMOCC tracers have been updated regarding the biogeochemistry processes, and the tracer transport routine is called. The HAMOCC tracers are transported along with the other two ICON-O tracers, temperature and salinity. In this scheme ICON-O and HAMOCC are entangled through the memory usage and the tracer transport. The HAMOCC tracers are part of the ICON-O tracer structure. Other HAMOCC variables, like tendencies and sediment, while exclusive to HAMOCC, were still created in ICON-O in order to allow for the use of the ICON infrastructure, like I/O. The ICON-O–HAMOCC interface handles the memory recasting between ICON-O and HAMOCC, as they use different memory layouts. The surface fluxes for HAMOCC are also handled in ICON-O. On the other hand, ICON-O does not have any dependencies on HAMOCC, except optionally the calculation of solar short-wave radiation absorption ratio, which is calculated in HAMOCC based on the chlorophyll's concentration.

While the two components are entangled, the basic prerequisites for concurrency exist: algorithmic independence and one-point communication. There is though a point for further consideration: most of the time when running ICON-O–HAMOCC is actually spent in the tracer transport, rather than in the HAMOCC biogeochemistry itself (see Sect. 5). HAMOCC transports 17 tracers, making it the most expensive part in the ICON-O–HAMOCC execution. Parallelizing only the HAMOCC biogeochemistry would result in only modest performance benefits, as the bulk of the execution would still be sequential (following the discussion in Sect. 3.2).

The solution that we follow is to allow the biogeochemistry to transport its own tracers, independently of the ocean. This requires the encapsulation of the tracer transport, so that it can be called by both ICON-O and HAMOCC. Two structures were created as interface to the tracer transport:

a. A tracer collection structure was created that contains the information of the tracers required for the transport.

b. A transport state structure was created that contains all the required fluxes. The transport state can be communicated from ICON-O to HAMOCC, allowing it to run the transport independently of ICON-O.

The tracer transport is an "embarrassingly" parallel process with regard to the number of tracers; every tracer can be

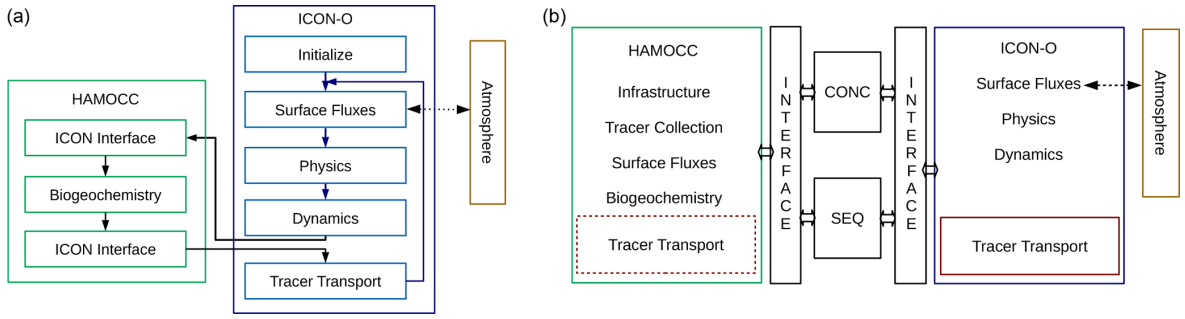

Figure 5. **(a)** Diagram of the sequential ICON-O–HAMOCC flow. **(b)** Diagram of the concurrent ICON-O–HAMOCC structure. The same interface is used for both the sequential and the concurrent mode.

transported independently of the others. This offers another level of data-parallel, medium-level parallelism, but it also presents some technical challenges, such as how to efficiently handle multiple communicators. We note that this approach does not replace concurrency, as a large part of the code, notably the dynamical core, still runs sequentially, presenting a scaling bottleneck. This level of parallelization has not been implemented in this project.

The next task was to disentangle the HAMOCC memory from ICON-O. The memory management of HAMOCC was moved into the HAMOCC component, and references to common global memory between the two components were removed. The surface flux calculations for the biogeochemistry were also moved from the ICON-O surface module to HAMOCC.

In the next step an interfacing mechanism between ICON-O and HAMOCC was created. This mechanism passes as parameters the information required for the two models. To HAMOCC the ocean and atmosphere variables are passed as described in Sect. 2.2; in addition the transport state is passed to HAMOCC to be used by the HAMOCC tracer transport. To ICON-O the short-wave penetration is passed and, in the case of a coupled setup, the CO_2 fluxes, which in turn are passed to the atmosphere through the coupler.

The final task was to provide HAMOCC with the necessary infrastructure to run autonomously. ICON provides a simple mechanism through namelists for registering the components that run concurrently, by defining the component and the group of MPI processes assigned to it. The calls to the infrastructure setup, such as domain decomposition, setting the communicators, the I/O, and the coupler, are done in the initialization phase in each of the components. While this mechanism does not provide the sophistication and power of more complex infrastructure frameworks, like the Earth System Modeling Framework (Hill et al., 2004; Collins et al., 2005), and it requires us to partly duplicate the code of setting up the infrastructure, it provides high-level infrastructure interfaces, and it is serviceable.

The final construction is presented in Fig. 5b. Two interfaces are constructed to send and receive information between ICON-O and HAMOCC on each side. The communication takes place just before the tracer transport for ICON-O while for HAMOCC at the beginning of each time step. The information communicated from ICON-O to HAMOCC includes the temperature, salinity, and pressure, used in the chemistry processes, surface fluxes, such as the total surface water flux, the solar radiation flux, the CO_2 concentration, and the wind stress. For the tracer transport, HAMOCC receives the fluxes and velocities from ICON-O, as well as the sea surface height. On the other side, ICON-O receives optionally the solar radiation absorption ratio and, in the case of a coupled setup with the atmosphere, the CO_2 ocean–atmosphere fluxes, which in turn are communicated to ICON-A.

The interfaces can serve two modes: sequential or concurrent. Both modes are transparent; the two components are "unaware" of the mode they run, as this is handled within the interfacing mechanism. This process also works in the coupled ICON-O–HAMOCC ICON-A setup, so the three components can run concurrently.

The interfacing mechanism serves only to communicate information between the two components on the same grid, without providing any further functionality that general couplers may provide. In the concurrent mode we use the communication library YAXT (Yet Another eXchange Tool; https://swprojects.dkrz.de/redmine/projects/yaxt, last access: 16 December 2022) developed at DKRZ. It provides a flexible interface that allows us to define both 2D and 3D communication patterns and can also aggregate the communication into one call. YAXT provides an abstraction for defining communication without any explicit MPI-message-passing calls. The communication scheme is automatically derived from descriptions of locally available data on each process. Thus, very different communication patterns, like transpositions or boundary exchanges, can be generated in a user-friendly manner, independently of the complexity of the domain decomposition. This leads to a significantly reduced and less error-prone programming effort. YAXT can also be used to generate the communication patterns for the redistribution of data between two sets of processes that use different

domain decompositions but the same grid. This is an essential functionality for implementing concurrency communication between components, that do not necessarily share the same decomposition but do share the same grid.

The new ICON-O–HAMOCC implementation gives bit-identical results with the original one, both in the sequential and concurrent mode, when the HAMOCC feedbacks (the ocean solar radiation absorption ratio and the ocean–atmosphere CO_2 fluxes) are disabled. Bit-identical results cannot be obtained in the case these feedbacks are activated in the concurrent mode, due to the different workflow from the sequential mode. It has been technically checked though for correctness when running with these feedbacks activated. The impact of concurrency on the results in this case still needs to be evaluated.

A final step was taken to introduce OpenMP directives in HAMOCC. While this is not directly related to coarse-grained concurrency, it provides the shared-memory level of parallelization. The memory layout currently used in ICON is suboptimal regarding the performance on CPUs, due to poor data locality (especially for stencil operators). This also has a negative impact on the OpenMP scaling, and thus the results do not represent the true potential of OpenMP parallelization.

A first study on the impact of the memory layout on performance and direct vs indirect (unstructured grid) indexing is presented in MacDonald et al. (2011) but without a discussion on its impact on shared-memory parallelization and scaling.

5 Experiments and performance results

We have performed two sets of experiments, one at a low horizontal resolution of 160 km on a 36 cores-per-node machine and another at a medium resolution of 40 km on a 128 cores-per-node machine. The two setups were measured for strong scaling, both in sequential and concurrent mode. Each of the runs was repeated three times, and the best of the three, in terms of the total time, was selected for the analysis.

We study the behaviour of the combined MPI, OpenMP parallelization, and the coarse-grained concurrency. Most of the ICON-O code is OpenMP-parallelized but not all. In particular, the sea-ice dynamics is not OpenMP-parallelized, and it was disabled in order not to distort the scaling behaviour. The Gent–McWilliams and the Redi parameterizations have not yet been included in the concurrent version, and they were also disabled. As we focus on the scaling behaviour, with and without concurrency, rather than the performance itself, these two modules would not change our conclusions on the effect of the concurrency.[5] All output was disabled in these runs. The time measures do not include the initialization phase of the models, as this is a one-time cost and would distort the scaling analysis for short runs.

[5] In fact, we expect that concurrency would be improved by including these two modules, as λ would get closer to 1 (see also Table 1).

Figure 6. The 160 km grid decomposed into 12 subdomains.

In fine-tuning a setup for performance we would typically calculate the number of OpenMP threads, the vector size, and the MPI tasks, so that shared-memory parallelization is balanced. No such effort was taken in these setups. We did examine though the effect of different vector sizes on the low-resolution experiment.

A discussion on scaling bottlenecks for a similar ocean-biogeochemistry setup is presented in Epicoco et al. (2016).

5.1 Low-resolution experiment, 160 km

The 160 km grid consists of 14 298 horizontal ocean cells and 40 vertical levels. This setup was run on the Mistral compute2 partition at DKRZ. Each node is equipped with two Intel Broadwell CPUs, providing a total of 36 cores. The experiments ran for 5 simulated years.

For the domain decomposition, a recursive weighted medial decomposition is employed. Each subdomain is assigned the number of total further "cuts" and is bisected in a weighted manner across the longest axis of weighted longitudes or latitudes. For example, if a subdomain is to be decomposed into five subdomains, the longest axis is found and is bisected with weights two and three. The two child subdomains are assigned two and three cuts respectively, and the process continues recursively. An example of the domain decomposition is given in Fig. 6.

First, we examine the behaviour of the sequential experiments. In the case that the domain decomposition would produce perfectly balanced square subdomains, the number of halo cells per subdomain would be proportional to $4\sqrt{A/N}$, where A is the total number of grid cells and N the number of subdomains. The total number of halo cells would be proportional to $4\sqrt{N \cdot A}$. In Fig. 7a, the ratio of halo cells to the grid cells is depicted, as well as the ratio of $4\sqrt{N \cdot A}$ to A. The actual halo ratio is significantly larger than the ideally calculated one, due to the imperfect decomposition with regard to the number of halo points. The relative cost of the halo communication is also depicted, which only partly reflects the increase of halos. In ICON, whenever possible, halo values are computed, instead of communicated, unless

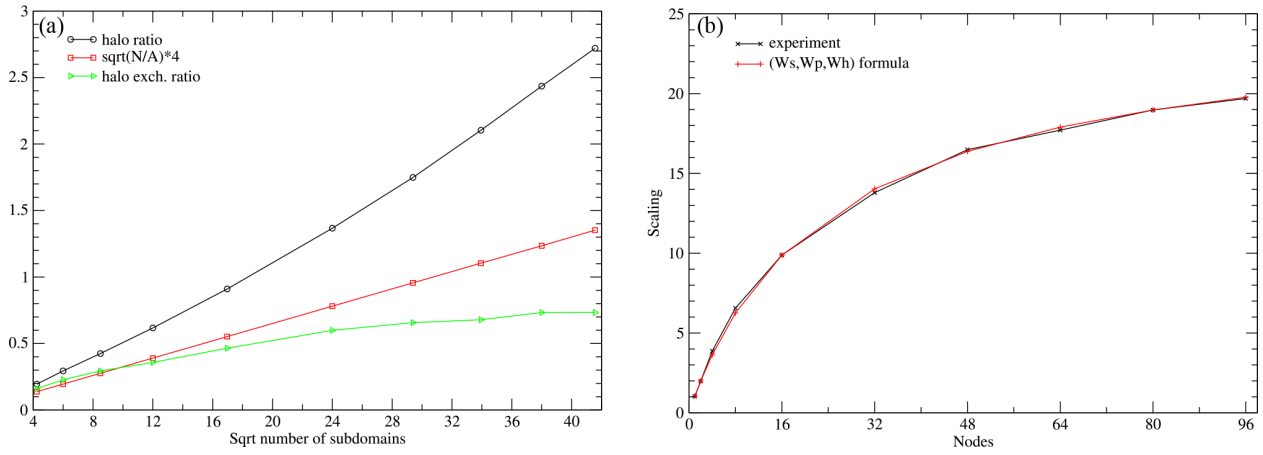

Figure 7. (a) The ratio of halo to non-halo cells. **(b)** Scaling results from the 160 km experiment, and scaling computed from formula (2).

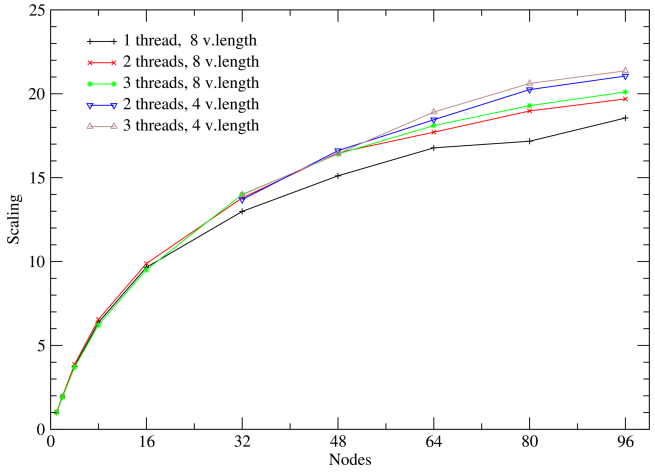

Figure 8. Scaling for the sequential 160 km setups for different numbers of OpenMP threads and vector lengths.

the computation is expensive. So the total computation cost is increased as well.

We can further approximate the scaling behaviour of the experiment by considering three types of workload: a sequential part W_s, a purely parallel part W_p, and a halo part W_h. We have for the total workload $W_t = W_s + W_p + W_h$, where we measure it in terms of time cost. The time cost when running on N MPI processes (i.e. subdomains) is

$$T_N = W_s + W_p/N + W_h/\sqrt{N}. \quad (2)$$

We estimated the workloads based on the sequential runs on 2, 16, and 80 nodes as $W_s = 243$ s, $W_p = 101\,535$ s, and $W_h = 774$ s. In Fig. 7b, the scaling results from the experiments and the above formula are shown. The formula captures the scaling behaviour of the experiments well, highlighting the importance of the non-scaling parts of the code.

For this experiment we tested the scaling for one, two, three, six, and nine OpenMP threads. More than three threads performed worse, and we do not include them in the results. We also checked the impact of a vector length 8 and 4 on the scaling behaviour. The results are presented in Fig. 8. The OpenMP parallelization becomes more important for a higher number of nodes. The vector length of 4 provides finer granularity and better balancing in higher number of nodes, and so it improves performance. We note that even as the number of halo cells exceeds the number of the original grid cells, we still get some scaling. As discussed above, the theoretical scaling in the case of the halo computation being dominant would still be proportional to $\sqrt{N}$.

Next we will only consider the best of the runs from Fig. 8 (that is the ones with the smallest total time). The relative costs and the times for the major components of the model are presented in Fig. 9. The highest cost comes from the HAMOCC tracer transport, while HAMOCC itself incurs relatively small cost. The halo communication cost imposes the most important restriction to scaling. This also includes load imbalance costs, which increase with the number of subdomains. The dycore (dynamical core) process consists of the calculation of the horizontal velocities and the sea surface height. ICON-O uses the iterative conjugate gradient method (CG) for inverting the matrix required by the implicit sea surface height calculation. In each iteration the computation of a global sum is required for calculating the magnitude of the residual. The time cost for the global sum used by the CG solver remains constant[6], imposing a sequential scaling limit, as described by Amdahl's law. Its relative cost though remains small in these runs due to the large cost of the tracer transport.

In the concurrent setup we kept the number of OpenMP threads constant, equal to 2. The vector length was set to 8, except for the experiments on 80 and 96 nodes, where it was set to 4. The MPI tasks are organized into two contiguous groups, the first containing the ICON-O tasks and the sec-

[6]This cost is included in the dycore cost.

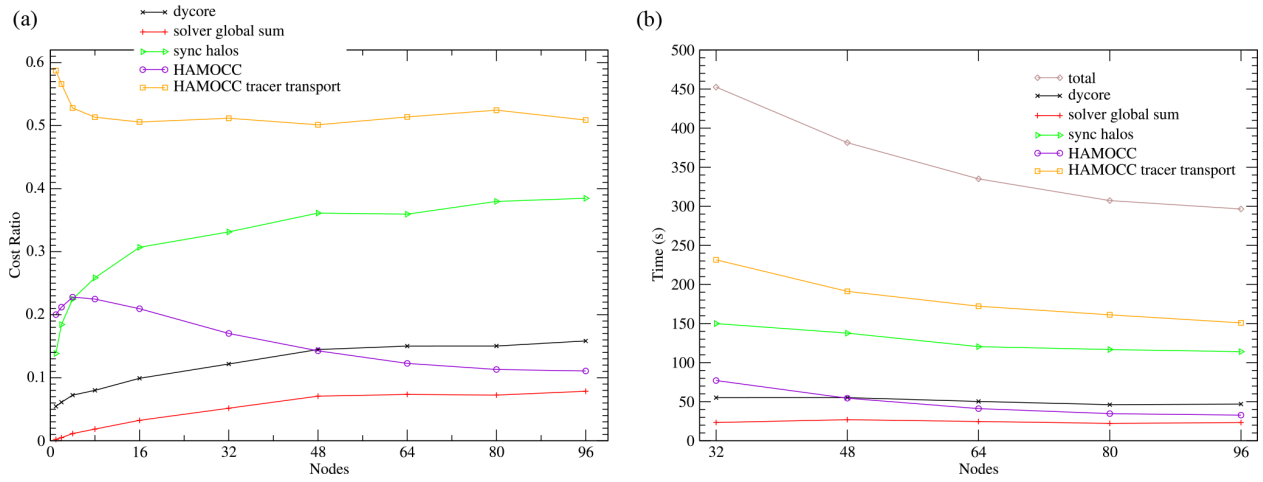

Figure 9. (a) Relative cost ratio of the ICON-O–HAMOCC components in the sequential 160 km setup. **(b)** Run time of the components.

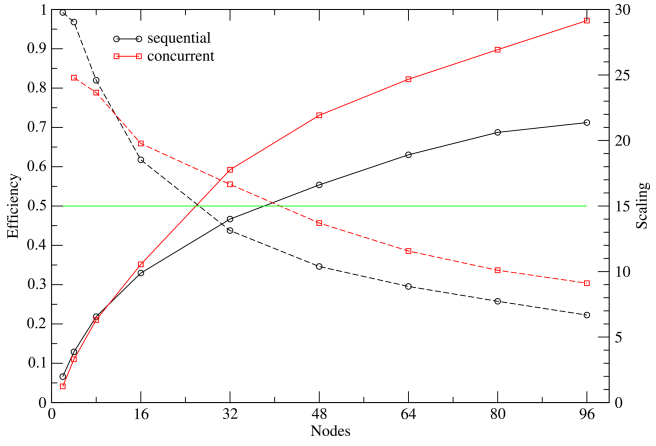

Figure 10. Scaling (solid line) and parallel efficiency (dashed line) for the 160 km sequential and concurrent setups. The 0.5 efficiency line is marked.

ond the HAMOCC tasks. The scaling behaviour of both the concurrent and the sequential case is depicted in Fig. 10. The performance is improved only after the sequential parallel efficiency has gone down to about 60 %. The limit of 50 % efficiency is reached in the sequential case at about 24 nodes while in the concurrent at about 40 nodes.

In Figs. 11 and 12 the relative costs and timers for the components of ICON-O and HAMOCC respectively are given for the concurrent experiments.

The different scaling characteristics of the two modules create additional imbalance; some care has been taken to improve this by reducing the number of ICON-O nodes (see Table 1). The imbalance becomes apparent in the cost of exchanging data and synchronizing ICON-O and HAMOCC. This cost in ICON-O is large, reaching more than 20 %, while in HAMOCC it is negligible. This indicates that the actual communication cost is small, while the cost of imbalance is borne by ICON-O. This imbalance though does not affect the total performance significantly, as on a total of 96 nodes for example, 76 are used for HAMOCC, and increasing this number by a few nodes would result in little improvement.

In Sect. 3.2 an analysis was provided regarding the scaling characteristics of concurrency in relation to the sequential parallel efficiency. We expect that concurrency would be beneficial after a parallel efficiency threshold has been reached, and this is confirmed by the experimental results. In Table 1 we compare the values of T_c/T_d, as computed by formula (1) in Sect. 3.2 and the ones given by the experiments. In Sect. 3.2 we defined the workload as the total number of operations and λ as the ratio of the workload between component A and B; both of them were considered to be constant numbers, independent of the number of nodes. We further assumed that the scaling behaviour of A and B is the same. In our experiment setup though the situation is different. The scaling behaviour of the two models is different, with the ICON-O scaling being worse that HAMOCC. In the sequential runs, the total relative cost of HAMOCC (including the HAMOCC tracer transport) is about 80 % on two nodes, while on 96 nodes this cost is about 62 %. The relative cost of ICON-O almost doubles in this range. In order to account for this difference, we will consider λ not to be constant but dependent on the number of nodes, so that it reflects the relative cost of HAMOCC in the sequential experiments.

In the experiment case, taking ICON-O as component A and HAMOCC as component B, we have $\lambda \geq 1$. In this case we have $L(N,\lambda) = \frac{F(N\cdot(1+\lambda))}{F(N\cdot\lambda)}$ and $T_c/T_d = L(N,\lambda)\cdot(1 + C(N\cdot(1+\lambda))/T(N\cdot\lambda))$. We set $C(N\cdot(1+\lambda))/T(N\cdot\lambda) = 0.1$ as a first guess. We calculate $L(N,\lambda)$ from the sequential experiments for $\lambda = 1, 2, 3$, and the corresponding ratio T_c/T_d is given in columns 6, 5, and 4 respectively. We note that $\lambda \geq 1$, so we have $L(N,\lambda) = \frac{F(N\cdot(1+\lambda))}{F(N\cdot\lambda)}$. In column 3 we give an estimation of λ_{exp} based on the sequential runs, computed as the ratio of the HAMOCC time to the ICON-O time.

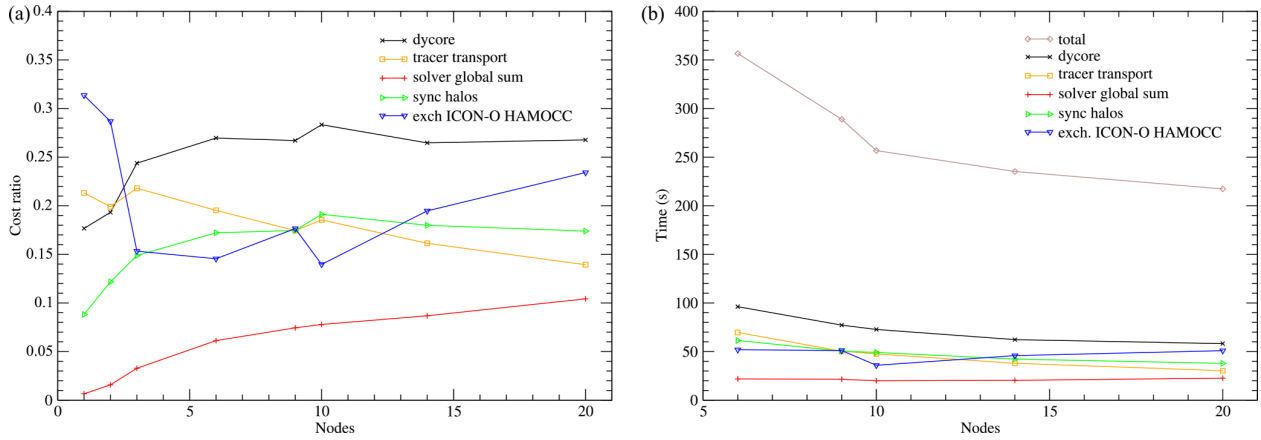

Figure 11. **(a)** Relative costs of the ICON-O components in the concurrent 160 km setup. **(b)** Run times of the ICON-O components.

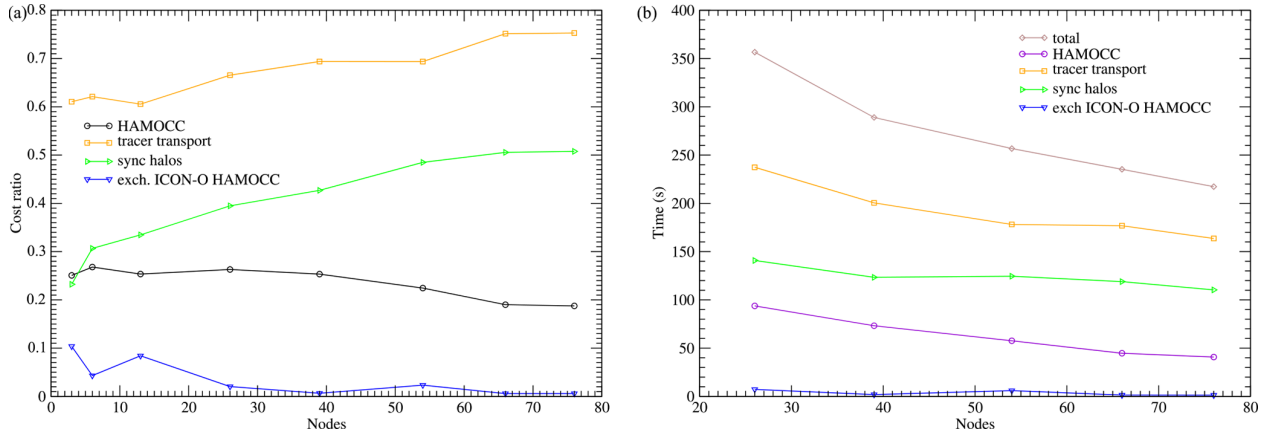

Figure 12. **(a)** Relative costs of the HAMOCC components in the concurrent 160 km setup. **(b)** Run times of the HAMOCC components.

In column 7 the estimation of the concurrent to the sequential time ratio T_c/T_d is given by linearly interpolating the T_c/T_d values from the $\lambda = 1, 2, 3$ parameter to the λ_{exp}. In the last column the actual ratio T_c/T_d is given from the experiments. While formula (1) is a great simplification of the real model behaviour, it still gives a reasonable approximation of the scaling behaviour of the concurrent setup, based on the behaviour of the sequential one. Furthermore, by comparing columns 4, 5, and 6, we observe that the predicted concurrency efficiency declines when λ deviates from the optimal value of 1, as expected, and the experiments T_c/T_d similarly improve as λ_{exp} approaches 1.

5.2 Medium-resolution experiment, 40 km

The 40 km grid consists of 230 124 horizontal ocean cells and 64 vertical levels. The experiments ran for 1 simulated year. They were run on the Levante machine at DKRZ, and each node is equipped with two AMD 7763 CPUs, giving a total of 128 cores per node. The number of OpenMP threads is constant, equal to 4, and the vector length is equal to 8. The MPI tasks are placed in a cyclic way in groups of eight across the nodes, so that the first eight MPI tasks occupy half of the first CPU of the first node, the second group occupies half of the first CPU of the second node, and so on. In the concurrent case the group of eight tasks consists of two ICON-O tasks and six HAMOCC tasks. This placement alleviates load imbalance, as it allows for improved memory bandwidth for the slower processes.

In Fig. 13 the sequential setup time costs are presented. The picture differs from the 160 km setup in the time cost of the CG global sum, which here takes 25 % of the time on 128 nodes. The total communication cost, including the global sum and the halo exchange, on 128 nodes exceeds 60 % of the total time, while for the 160 km on 96 nodes this cost is about 47 %. This underlines the communication bottlenecks when using machines equipped with powerful multi-core nodes.

The scaling of the sequential and the concurrent setup is shown in Fig. 14. Scaling is measured against the sequential run on two nodes. The first thing to observe is the super-linear scaling of both the sequential and the concurrent setups when

Table 1. The first two columns describe the nodes used for the ICON-O–HAMOCC concurrent setup. Columns 4–6 describe the estimated ratio of concurrent to the sequential time T_c/T_d calculated using Eq. (1) and the relative efficiency for different values of λ from the sequential runs. In column 3 $\lambda = \lambda_{exp}$ is computed from the sequential runs and in column 7 the estimation of the T_c/T_d ratio for this λ through interpolation. In the last column the actual T_c/T_d is computed from the runs.

Total nodes	ICON-O nodes	λ_{exp}	T_c/T_d, $\lambda = 3$	T_c/T_d, $\lambda = 2$	T_c/T_d, $\lambda = 1$	T_c/T_d, $\lambda = \lambda_{exp}$	T_c/T_d exp.
4	1	3.10	1.07	1.11	1.07	1.07	1.17
8	2	2.82	1.01	1.01	0.93	1.01	1.04
16	3	2.51	0.97	0.94	0.83	0.96	0.94
32	6	2.14	0.94	0.91	0.78	0.92	0.79
48	9	1.81	0.92	0.90	0.74	0.86	0.76
64	10	1.75	0.94	0.87	0.74	0.84	0.77
80	14	1.77	0.98	0.83	0.74	0.81	0.77
96	20	1.63	0.95	0.82	0.71	0.78	0.73

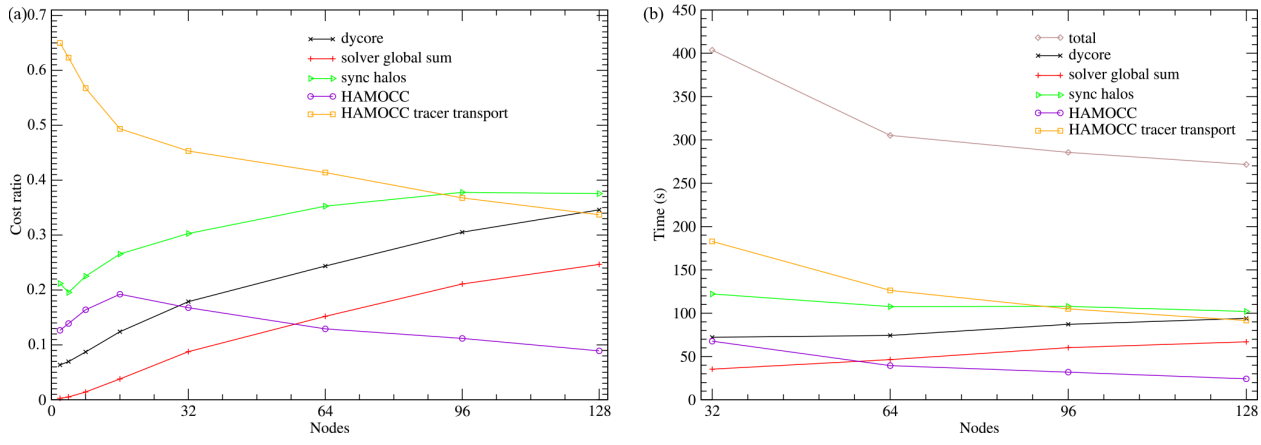

Figure 13. (a) Relative costs of the ICON-O–HAMOCC components in the sequential 40 km setup. **(b)** Run times for each component.

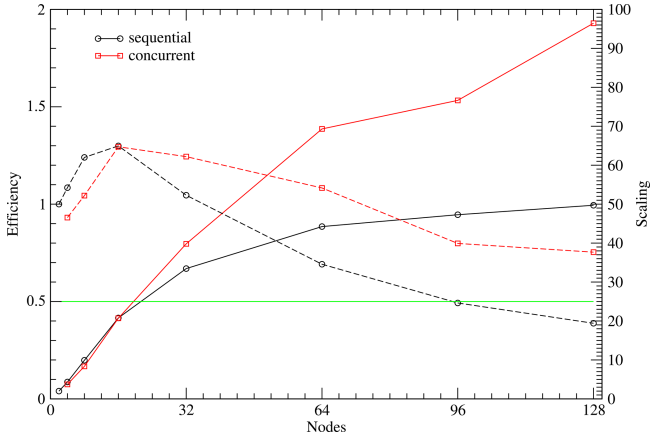

Figure 14. Scaling (solid line) and parallel efficiency (dashed line) for the 40 km sequential and concurrent setups. The 0.5 efficiency line is marked.

using up to 16 nodes. This behaviour typically is caused by more efficient cache usage in NUMA machines. An indication towards this inference is provided from the behaviour of the HAMOCC tracer transport in Fig. 13a. The tracer transport process is memory-intensive due to the large number of stencil operations, which, as we have noted, are sub-optimal due to poor data locality. Its cost drops sharply when using up to 16 nodes, which indicates better relative memory efficiency. Another related consequence is that the parallel efficiency in the sequential run reaches the 0.5 mark at a relatively high count of nodes, 96, while in the concurrent case it never reaches this limits and stays above 0.7. This is due to the poor performance on the reference number of nodes of two.

We note that the assumptions for formula (2) do not apply to this experiment. This is partly due to the super-linear behaviour of this setup in the low count of nodes but also because this setup exhibits a part that its cost increases with the number of nodes, namely the global sum (see Fig. 13b). This cost is significant and has not been accounted for in formula (2).

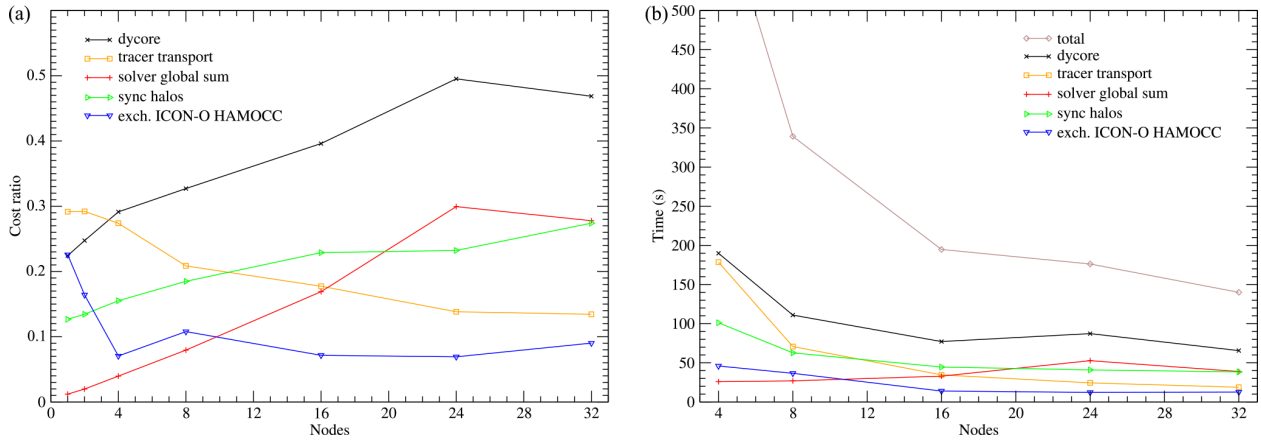

Figure 15. **(a)** Relative costs of the ICON-O components in the concurrent 40 km setup. **(b)** Run times for each component.

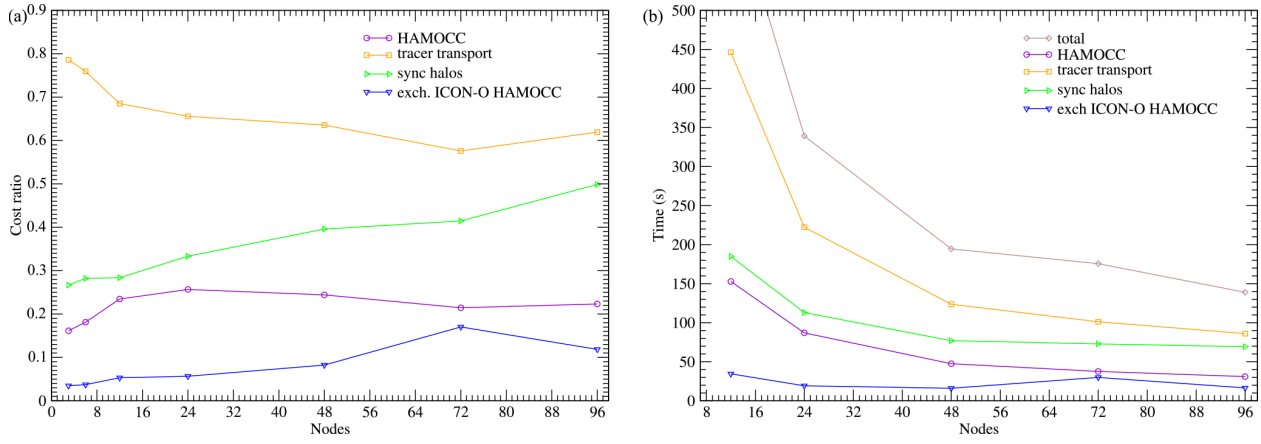

Figure 16. **(a)** Relative costs for the HAMOCC components in the concurrent 40 km setup. **(b)** Run times for each component.

The relative costs and times for ICON-O and HAMOCC in the concurrent setup are given in Figs. 15 and 16 respectively. We again see that the communication costs within each module are significant, while the coupling cost between ICON-O and HAMOCC remains relatively small.

In Table 2 we perform the same calculations for the 40 km setup, as we did in Table 1 for the 160 km setup. We see a similar picture, except for the first two rows, where both the predicted and the actual concurrency efficiency declines as a function of N and as a function of λ. This a result of the super-linear scaling. Only after the threshold of 16 nodes does concurrency become beneficial, and thereafter its efficiency, relatively to the sequential setup, increases. While the theoretical prediction underestimates the effect of concurrency (this is due to the different scaling behaviour of the two components, which is not accounted in the formula), it still gives a reasonable estimation.

We note that the concurrency efficiency is significantly better in this setup, compared to the 160 km. The ratio T_c/T_d reaches 0.52, while for the 160 km it is 0.73. This behaviour reflects the limitations of the data parallel approaches when using highly parallel architectures. Figure 13 reveals the issue: the cost of communicating the halos dominates the performance at 128 nodes, along with the solver global sum.

The two setups of the 160 and the 40 km present a scenario where the resolution is increased, in combination with the transition to a more powerful and parallel machine. We examine the impact of concurrency in this scenario. Two experiments were selected from the two setups, so that they have approximately the same number of 2D cells per core, and at the same time they are at a similar scaling limit in the sequential case of each setup, where the parallel efficiency drops below 50 %. The details of these two experiments are given in Table 3.

In row 6 the number of 3D cells per core, normalized for one simulated minute, is given. Their ratio suggests the relative workload per core for the two experiments: the 40 km has 2.3 times more load per core.[7] In rows 7 and 8 the times

[7] Some details are not taken into account here, such as the number of iterations the CG solver requires, which is higher for the 40 km.

Table 2. As Table 1, for the 40 km experiment. The ratio of the ICON-O to the HAMOCC cores is constant, 1/3, and is omitted.

Total nodes	λ_{exp}	T_c/T_d, $\lambda = 3$	T_c/T_d, $\lambda = 2$	T_c/T_d, $\lambda = 1$	T_c/T_d, $\lambda = \lambda_{exp}$	T_c/T_d exp.
4	3.20	1.17	1.12	1.19	1.17	1.17
8	2.73	1.21	1.14	1.26	1.19	1.19
16	2.18	1.11	1.14	1.15	1.14	1.00
32	1.64	0.97	1.01	0.89	0.96	0.84
64	1.19	0.90	0.89	0.73	0.76	0.64
96	0.92	0.81	0.87	0.64	0.64	0.62
128	0.74	0.87	0.78	0.61	0.61	0.52

Table 3. Detailed numbers for the two experiments selected from the two setups of 160 and 40 km. The description is provided in the text. A total of 1152 Mistral cores correspond to 32 nodes, while 16 384 Levante cores correspond to 128 nodes.

		160 km, 1152 Mistral cores	40 km, 16 384 Levante cores	Ratio 40 km / 160 km
(1)	2D cells	14 298	230 124	16.10
(2)	3D cells	481 402	11 960 498	24,85
(3)	2D cells/core	12.41	14.05	1.13
(4)	3D cells/core	417.88	730.01	1.75
(5)	Time step (min)	60	45	0.75
(6)	3D cells/core/sim.min.	6.97	16.22	2.33
(7)	Seq. time (s), 1 simulated year	90.50	271.63	3.00
(8)	Conc. time (s), 1 simulated year	71.22	140.00	1.97

for simulating 1 year are given. In the sequential case, the 40 km is 3 times slower than the 160 km, a value that is above the estimated 2.3. On the other hand, in the concurrent case, the 40 km experiment is less than 2 times slower, which provides another indication of the increased impact of concurrency when moving to highly parallel machines.

6 Discussion and outlook

By decomposing the algorithmic space, coarse-grained component concurrency offers another parallelization dimension, in addition to the existing data parallel approaches. It improves scalability when certain scaling limits have been reached through the data parallel methods. It is more effective in the regimes where the relative parallel efficiency drops. In the regimes of linear scaling it results in little, if any, improvements. It produces higher parallel workload when compared to data parallel approaches. This makes it suitable, and probably indispensable, for efficiently utilizing massively parallel machines. Our experiments show that the concurrency effectiveness increased from 1.4 times improvement on a 36 cores-per-node machine to 2 times improvement on a 128 cores-per-node machine. We also see that the "coarseness" is an important factor to this approach. It gives the best results when the two components incur approximately the same cost, while its effectiveness deteriorates when moving away from this balance. Both our theoretical analysis and our experimental results concur with the above conclusions.

It is clear that coarse-grained concurrency does not make the code faster, and the more traditional code optimization and scaling improvement procedures are still an important part of the process of getting the models to run efficiently. We expect for example that improving data locality for ICON-O–HAMOCC would have a higher performance impact on Levante than concurrency. These more intricate optimization processes require a high level of expertise and in cases extensive code restructuring. In comparison, engineering coarse-grained component concurrency is in general a simpler process and requires only a modest effort when good software engineering practices were already in place.

Coarse-grained concurrency can always be applied on top of the other optimizations, allowing us to use more computational resources and to use these more efficiently. This implies that enough parallel computational resources should be available in order to be practically useful. On the other hand, the new machines are massively parallel, with nodes equipped with hundreds or thousands of cores, providing computing power that has reached the exaflop level. The most effective way to make use of this massively parallel computing power is through multi-level and multi-dimensional parallelism. Highly parallel architectures, like

GPUs, require a minimum of parallel workload to provide the best efficiency. Experiments show that we need millions of 3D grid points per GPU to make full use of them (Leutwyler et al., 2016; Giorgetta et al., 2022). This limits the extend that data parallelism can be used on such machines. We expect that coarse-grained concurrency will provide an effective leverage for making use of these architectures.

Coarse-grained concurrency is a high-level parallelization and thus independent of architectures. This makes it applicable on different architectures, without the need to reimplement the concurrency mechanism. Furthermore, it can be applied on heterogeneous environments in hybrid mode, as on CPUs and GPUs, providing the potential to make use of all the resources of heterogeneous machines.

A positive side effect of coarse-grained component concurrency is that it naturally bequeaths concurrency to the infrastructure attached to these components, like I/O and real-time post-processing. In particular, I/O can pose a significant performance bottleneck, and parallel asynchronous I/O approaches have already been developed; see, for example, Brown et al. (2020), Yepes-Arbós et al. (2022), and Hohenegger et al. (2022). The naturally inherited concurrency to the components' infrastructure can further enhance the performance of such schemes. While these side effects have not been the subject of this paper, we expect them to help increase the efficiency of the existing approaches.

The applicability of coarse-grained concurrency seems plausible to other components of Earth system models. One such component can be the sea-ice model. The coarse-grained concurrency effectiveness will depend how much "coarseness" the components present and how tightly they are connected, which will reflect on the coupling cost. Questions related to the handling of the feedbacks between the components have to be considered. We have not addressed this question regarding the concurrent ICON-O–HAMOCC when the interactive carbon cycle is activated in a coupled setup. This would be a subject of future work.

Code availability. The ICON code is available under licenses; see https://mpimet.mpg.de/en/science/modeling-with-icon/code-availability (Max-Planck-Institut für Meteorologie, 2022). The experiments were performed with ICON v.2.6.5. The code is also documented in the provided dataset.

Data availability. The code, scripts, and results are available in the dataset provided in Linardakis (2022, https://doi.org/10.17617/3.FGFQZG).

Author contributions. IS integrated the HAMOCC code into the ICON framework and contributed to the concurrent implementation. MHe provided the YAXT library and its interface for the concurrent ICON-O–HAMOCC communication and contributed the YAXT description. LR, FC, and TI contributed the biogeochemistry and HAMMOCC description. PK provided the ICON-O description. LL contributed the rest. All co-authors contributed to the text formulation.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Acknowledgements. The reviewers Peter Düben and Mauro Bianco provided useful suggestions that helped improve this paper and clarify some points. Marco Giorgetta and Daniel Klocke offered insightful comments that helped clarify some of the concepts presented in this paper. Reiner Schnur read a first draft of this paper and offered points for improvement. Kai Logemann and Moritz Mathis provided the ICON-O–HAMOCC coastal ocean grid for illustration purposes. Jan Frederik Engels and Panos Adamidis at DKRZ have helped us with the technical aspects of running the experiments on Mistral and Levante. Finally, the text formulation has benefited from the meticulous work by the *GMD* typesetting and editing team.

Financial support. The article processing charges for this open-access publication were covered by the Max Planck Society.

Review statement. This paper was edited by Chanh Kieu and reviewed by Mauro Bianco and Peter Düben.

References

Amdahl, G. M.: Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, in: Proceedings of the April 18–20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring), 483–485, Association for Computing Machinery, New York, NY, USA, https://doi.org/10.1145/1465482.1465560, 1967.

Balaji, V.: Scientific Computing in the Age of Complexity, XRDS, 19, 12–17, https://doi.org/10.1145/2425676.2425684, 2013.

Balaji, V., Benson, R., Wyman, B., and Held, I.: Coarse-grained component concurrency in Earth system modeling: parallelizing atmospheric radiative transfer in the GFDL AM3 model using the Flexible Modeling System coupling framework, Geosci. Model Dev., 9, 3605–3616, https://doi.org/10.5194/gmd-9-3605-2016, 2016.

Bauer, P., Stevens, B., and Hazeleger, W.: A digital twin of Earth for the green transition, Nat. Clim. Change, 11, 80–83, https://doi.org/10.1038/s41558-021-00986-y, 2021.

Breitburg, D., Levin, L. A., Oschlies, A., Grégoire, M., Chavez, F. P., Conley, D. J., Garçon, V., Gilbert, D., Gutiérrez, D., Isensee, K., Jacinto, G. S., Limburg, K. E., Montes, I., Naqvi, S. W. A., Pitcher, G. C., Rabalais, N. N., Roman, M. R., Rose, K. A.,

Seibel, B. A., Telszewski, M., Yasuhara, M., and Zhang, J.: Declining oxygen in the global ocean and coastal waters, Science, 359, eaam7240, https://doi.org/10.1126/science.aam7240, 2018.

Brown, N., Weiland, M., Hill, A., Shipway, B., Maynard, C., Allen, T., and Rezny, M.: A highly scalable Met Office NERC Cloud model, CoRR, ArXiv [preprint], https://doi.org/10.48550/arXiv.2009.12849, 2020.

Ciais, P., Sabine, C., Bala, G., Bopp, L., Brovkin, V., Canadell, J., Chhabra, A., DeFries, R., Galloway, J., Heimann, M., Jones, C., Le Quéré, C., Myneni, R. B., Piao, S., and Thornton, P.: Carbon and other biogeochemical cycles, in: Climate change 2013: the physical science basis, Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change, Cambridge University Press, ISBN 978-1-107-05799-1, 465–570, 2014.

Colella, P. and Woodward, P. R.: The Piecewise Parabolic Method (PPM) for gas-dynamical simulations, J. Comput. Phys., 54, 174–201, https://doi.org/10.1016/0021-9991(84)90143-8, 1984.

Collins, N., Theurich, G., DeLuca, C., Suarez, M., Trayanov, A., Balaji, V., Li, P., Yang, W., Hill, C., and da Silva, A.: Design and Implementation of Components in the Earth System Modeling Framework, Int. J. High Perform. C., 19, 341–350, https://doi.org/10.1177/1094342005056120, 2005.

Crueger, T., Giorgetta, M., Brokopf, R., Esch, M., Fiedler, S., and Hohenegger, S.: ICON-A, the atmosphere component of the ICON Earth system model. II: Model evaluation, J. Adv. Model. Earth Sy., 10, 1638–1662, https://doi.org/10.1029/2017MS001233, 2018.

Dalmedico, A. D.: History and Epistemology of Models: Meteorology (1946–1963) as a Case Study, Arch. Hist. Exact Sci., 55, 395–422, 2001.

Danilov, S., Wang, Q., Timmermann, R., Iakovlev, N., Sidorenko, D., Kimmritz, M., Jung, T., and Schröter, J.: Finite-Element Sea Ice Model (FESIM), version 2, Geosci. Model Dev., 8, 1747–1761, https://doi.org/10.5194/gmd-8-1747-2015, 2015.

Dickson, A. G.: The carbon dioxide system in seawater: equilibrium chemistry and measurements, Guide to best practices for ocean acidification research and data reporting, 1, 17–40, https://doi.org/10.2777/66906, 2010.

Dickson, A. G., Sabine, C. L., and Christian, J. R.: Guide to best practices for ocean CO_2 measurements, PICES Special Publication 3, 191 pp., ISBN 1-897176-01-5, 2007.

Donahue, A. S. and Caldwell, P. M.: Performance and Accuracy Implications of Parallel Split Physics-Dynamics Coupling in the Energy Exascale Earth System Atmosphere Model, J. Adv. Model. Earth Sy., 12, e2020MS002080, https://doi.org/10.1029/2020MS002080, 2020.

Eastham, S. D., Long, M. S., Keller, C. A., Lundgren, E., Yantosca, R. M., Zhuang, J., Li, C., Lee, C. J., Yannetti, M., Auer, B. M., Clune, T. L., Kouatchou, J., Putman, W. M., Thompson, M. A., Trayanov, A. L., Molod, A. M., Martin, R. V., and Jacob, D. J.: GEOS-Chem High Performance (GCHP v11-02c): a next-generation implementation of the GEOS-Chem chemical transport model for massively parallel applications, Geosci. Model Dev., 11, 2941–2953, https://doi.org/10.5194/gmd-11-2941-2018, 2018.

Epicoco, I., Mocavero, S., Macchia, F., Vichi, M., Lovato, T., Masina, S., and Aloisio, G.: Performance and results of the high-resolution biogeochemical model PELAGOS025 v1.0 within NEMO v3.4, Geosci. Model Dev., 9, 2115–2128, https://doi.org/10.5194/gmd-9-2115-2016, 2016.

Giorgetta, M., Brokopf, R., Crueger, T., Esch, M., Fiedler, S., and Helmert, J.: ICON-A, the atmosphere component of the ICON Earth system model. I: Model description, J. Adv. Model. Earth Sy., 10, 1613–1637, https://doi.org/10.1029/2017MS001242, 2018.

Giorgetta, M. A., Sawyer, W., Lapillonne, X., Adamidis, P., Alexeev, D., Clément, V., Dietlicher, R., Engels, J. F., Esch, M., Franke, H., Frauen, C., Hannah, W. M., Hillman, B. R., Kornblueh, L., Marti, P., Norman, M. R., Pincus, R., Rast, S., Reinert, D., Schnur, R., Schulzweida, U., and Stevens, B.: The ICON-A model for direct QBO simulations on GPUs (version icon-cscs:baf28a514), Geosci. Model Dev., 15, 6985–7016, https://doi.org/10.5194/gmd-15-6985-2022, 2022.

Hanke, M., Redler, R., Holfeld, T., and Yastremsky, M.: YAC 1.2.0: new aspects for coupling software in Earth system modelling, Geosci. Model Dev., 9, 2755–2769, https://doi.org/10.5194/gmd-9-2755-2016, 2016.

Heinze, C., Maier-Reimer, E., Winguth, A. M., and Archer, D.: A global oceanic sediment model for long-term climate studies, Global Biogeochem. Cy., 13, 221–250, 1999.

Hill, C., DeLuca, C., Balaji, Suarez, M., and Da Silva, A.: The architecture of the Earth System Modeling Framework, Comput. Sci. Eng., 6, 18–28, https://doi.org/10.1109/MCISE.2004.1255817, 2004.

Hohenegger, C., Korn, P., Linardakis, L., Redler, R., Schnur, R., Adamidis, P., Bao, J., Bastin, S., Behravesh, M., Bergemann, M., Biercamp, J., Bockelmann, H., Brokopf, R., Brüggemann, N., Casaroli, L., Chegini, F., Datseris, G., Esch, M., George, G., Giorgetta, M., Gutjahr, O., Haak, H., Hanke, M., Ilyina, T., Jahns, T., Jungclaus, J., Kern, M., Klocke, D., Kluft, L., Kölling, T., Kornblueh, L., Kosukhin, S., Kroll, C., Lee, J., Mauritsen, T., Mehlmann, C., Mieslinger, T., Naumann, A. K., Paccini, L., Peinado, A., Praturi, D. S., Putrasahan, D., Rast, S., Riddick, T., Roeber, N., Schmidt, H., Schulzweida, U., Schütte, F., Segura, H., Shevchenko, R., Singh, V., Specht, M., Stephan, C. C., von Storch, J.-S., Vogel, R., Wengel, C., Winkler, M., Ziemen, F., Marotzke, J., and Stevens, B.: ICON-Sapphire: simulating the components of the Earth System and their interactions at kilometer and subkilometer scales, Geosci. Model Dev. Discuss. [preprint], https://doi.org/10.5194/gmd-2022-171, in review, 2022.

Ilyina, T., Six, K. D., Segschneider, J., Maier-Reimer, E., Li, H., and Núñez-Riboni, I.: Global ocean biogeochemistry model HAMOCC: Model architecture and performance as component of the MPI-Earth system model in different CMIP5 experimental realizations, J. Adv. Model. Earth Sy., 5, 287–315, https://doi.org/10.1029/2012MS000178, 2013.

Jungclaus, J. H., Lorenz, S. J., Schmidt, H., Brovkin, V., Brüggemann, N., Chegini, F., Crüger, T., De-Vrese, P., Gayler, V., Giorgetta, M. A., Gutjahr, O., Haak, H., Hagemann, S., Hanke, M., Ilyina, T., Korn, P., Kröger, J., Linardakis, L., Mehlmann, C., Mikolajewicz, U., Müller, W. A., Nabel, J. E. M. S., Notz, D., Pohlmann, H., Putrasahan, D. A., Raddatz, T., Ramme, L., Redler, R., Reick, C. H., Riddick, T., Sam, T., Schneck, R., Schnur, R., Schupfner, M., von Storch, J.-S., Wachsmann, F., Wieners, K.-H., Ziemen, F., Stevens, B., Marotzke, J., and Claussen, M.: The ICON Earth System Model Ver-

sion 1.0, J. Adv. Model. Earth Sy., 14, e2021MS002813, https://doi.org/10.1029/2021MS002813, 2022.

Korn, P.: Formulation of an Unstructured Grid Model for Global Ocean Dynamics, J. Comput. Phys., 339, 525–552, 2017.

Korn, P. and Danilov, S.: Elementary dispersion analysis of some mimetic discretizations on triangular C-grids, J. Comput. Phys., 330, 156–172, https://doi.org/10.1016/j.jcp.2016.10.059, 2017.

Korn, P. and Linardakis, L.: A conservative discretizations of the shallow-water equations on triangular grids, J. Comput. Phys., 375, 871–900, 2018.

Korn, P., Brüggemann, N., Jungclaus, J. H., Lorenz, S. J., Gutjahr, O., Haak, H., Linardakis, L., Mehlmann, C., Mikolajewicz, U., Notz, D., Putrasahan, D. A., Singh, V., von Storch, J.-S., Zhu, X., and Marotzke, J.: ICON-O: The Ocean Component of the ICON Earth System Model – Global Simulation Characteristics and Local Telescoping Capability, J. Adv. Model. Earth Sy., 14, e2021MS002952, https://doi.org/10.1029/2021MS002952, 2022.

Lacroix, F., Ilyina, T., Laruelle, G. G., and Regnier, P.: Reconstructing the preindustrial coastal carbon cycle through a global ocean circulation model: was the global continental shelf already both autotrophic and a CO_2 sink?, Global Biogeochem. Cy., 35, e2020GB006603, https://doi.org/10.1029/2020GB006603, 2021.

Lamport, L.: Turing Lecture: The Computer Science of Concurrency: The Early Years, Commun. ACM, 58, 71–76, https://doi.org/10.1145/2771951, 2015.

Leutwyler, D., Fuhrer, O., Lapillonne, X., Lüthi, D., and Schär, C.: Towards European-scale convection-resolving climate simulations with GPUs: a study with COSMO 4.19, Geosci. Model Dev., 9, 3393–3412, https://doi.org/10.5194/gmd-9-3393-2016, 2016.

Linardakis, L.: Dataset for: Improving scalability of Earth System Models through coarse-grained component concurrency – a case study with the ICON v2.6.5 modelling system, Edmond [data set], https://doi.org/10.17617/3.FGFQZG, 2022.

Liu, B., Six, K. D., and Ilyina, T.: Incorporating the stable carbon isotope ^{13}C in the ocean biogeochemical component of the Max Planck Institute Earth System Model, Biogeosciences, 18, 4389–4429, https://doi.org/10.5194/bg-18-4389-2021, 2021.

Logemann, K., Linardakis, L., Korn, P., and Schrum, C.: Global tide simulations with ICON-O: testing the model performance on highly irregular meshes, Ocean Dynam., 21, 43–57, 2021.

Long, M. S., Yantosca, R., Nielsen, J. E., Keller, C. A., da Silva, A., Sulprizio, M. P., Pawson, S., and Jacob, D. J.: Development of a grid-independent GEOS-Chem chemical transport model (v9-02) as an atmospheric chemistry module for Earth system models, Geosci. Model Dev., 8, 595–602, https://doi.org/10.5194/gmd-8-595-2015, 2015.

MacDonald, A. E., Middlecoff, J., Henderson, T., and Lee, J.-L.: A general method for modeling on irregular grids, Int. J. High Perform. C., 25, 392–403, https://doi.org/10.1177/1094342010385019, 2011.

Maerz, J., Six, K. D., Stemmler, I., Ahmerkamp, S., and Ilyina, T.: Microstructure and composition of marine aggregates as co-determinants for vertical particulate organic carbon transfer in the global ocean, Biogeosciences, 17, 1765–1803, https://doi.org/10.5194/bg-17-1765-2020, 2020.

Maier-Reimer, E.: Towards a global ocean carbon model, in: Interactions between climate and biosphere, Swets & Zeitlinger, 295–310, ISBN-10 9026505272, ISBN-13 978-9026505270, 1984.

Maier-Reimer, E. and Hasselmann, K.: Transport and storage of CO_2 in the ocean – an inorganic ocean-circulation carbon cycle model, Clim. Dynam., 2, 63–90, 1987.

Mathis, M., Logemann, K., Maerz, J., Lacroix, F., Hagemann, S., Chegini, F., Ramme, L., Ilyina, T., Korn, P., and Schrum, C.: Seamless integration of the coastal ocean in global marine carbon cycle modeling, J. Adv. Model. Earth Sy., 14, e2021MS002789, https://doi.org/10.1029/2021MS002789, 2022.

Mattson, T. G.: How Good is OpenMP, Sci. Programm., 11, 124373, https://doi.org/10.1155/2003/124373, 2003.

Mattson, T. G., Hwu, W., and Keutzer, K.: The Concurrency Challenge, IEEE Des. Test Comput., 25, 312–320, https://doi.org/10.1109/MDT.2008.110, 2008.

Max-Planck-Institut für Meteorologie: ICON, https://mpimet.mpg.de/en/science/modeling-with-icon/code-availability, last access: 16 December 2022.

McGuffie, K. and Henderson-Sellers, A.: Forty years of numerical climate modelling, Int. J. Climatol., 21, 1067–1109, https://doi.org/10.1002/joc.632, 2001.

Miyamoto, Y., Kajikawa, Y., Yoshida, R., Yamaura, T., Yashiro, H., and Tomita, H.: Deep moist atmospheric convection in a subkilometer global simulation, Geophys. Res. Lett., 40, 4922–4926, https://doi.org/10.1002/grl.50944, 2013.

Mozdzynski, G.: Report outlining a strategic approach for efficiency savings based on concurrency and accuracy (D2.6), Zenodo, https://doi.org/10.5281/zenodo.1453858, 2018.

Mozdzynski, G. and Morcrette, J.-J.: Reorganization of the radiation transfer calculations in the ECMWF IFS, ECMWF Technical Memoranda, 721, https://doi.org/10.21957/pxjpl93ov, 2014.

Nabel, J. E. M. S., Naudts, K., and Pongratz, J.: Accounting for forest age in the tile-based dynamic global vegetation model JSBACH4 (4.20p7; git feature/forests) – a land surface model for the ICON-ESM, Geosci. Model Dev., 13, 185–200, https://doi.org/10.5194/gmd-13-185-2020, 2020.

Orr, J. C., Fabry, V. J., Aumont, O., Bopp, L., Doney, S. C., Feely, R. A., Gnanadesikan, A., Gruber, N., Ishida, A., Joos, F., Key, R. M., Lindsay, K., Maier-Reimer, E., Matear, R., Monfray, P., Mouchet, A., Najjar, R. G., Plattner, G. K., Rodgers, K. B., Sabine, C. L., Sarmiento, J. L., Schlitzer, R., Slater, R. D., Totterdell, I. J., Weirig, M.-F., Yamanaka, Y., and Yool, A.: Anthropogenic ocean acidification over the twenty-first century and its impact on calcifying organisms, Nature, 437, 681–686, 2005.

Paulsen, H., Ilyina, T., Six, K. D., and Stemmler, I.: Incorporating a prognostic representation of marine nitrogen fixers into the global ocean biogeochemical model HAMOCC, J. Adv. Model. Earth Sy., 9, 438–464, 2017.

Randall, D. A., Bitz, C. M., Danabasoglu, G., Denning, A. S., Gent, P. R., Gettelman, A., Griffies, S. M., Lynch, P., Morrison, H., Pincus, R., and Thuburn, J.: 100 Years of Earth System Model Development, Meteorol. Monogr., 59, 12.1–12.66, https://doi.org/10.1175/AMSMONOGRAPHS-D-18-0018.1, 2018.

Rieger, D., Bangert, M., Bischoff-Gauss, I., Förstner, J., Lundgren, K., Reinert, D., Schröter, J., Vogel, H., Zängl, G., Ruhnke, R., and Vogel, B.: ICON–ART 1.0 – a new online-coupled model

system from the global to regional scale, Geosci. Model Dev., 8, 1659–1676, https://doi.org/10.5194/gmd-8-1659-2015, 2015.

Sarmiento, J. L. and Gruber, N.: Ocean Biogeochemical Dynamics, Princeton University Press, ISBN 9780691017075, 2006.

Semtner, A. J.: A model for the thermodynamic growth of sea ice in numerical investigations of climate, J. Phys. Oceanogr., 6, 379–389, 1976.

Six, K. D. and Maier-Reimer, E.: Effects of plankton dynamics on seasonal carbon fluxes in an ocean general circulation model, Global Biogeochem. Cy., 10, 559–583, 1996.

Stevens, B., Satoh, M., Auger, L., Biercamp, J., Bretherton, C. S., Chen, X., Düben, P., Judt, F., Khairoutdinov, M., Klocke, D., Kodama, C., Kornblueh, L., Lin, S.-J., Neumann, P., Putman, W. M., Röber, N., Shibuya, R., Vanniere, B., Vidale, P. L., and Wedi, N.: DYAMOND: the DYnamics of the Atmospheric general circulation Modeled On Non-hydrostatic Domains, Prog. Earth Planet. Sci., 6, 61, https://doi.org/10.1186/s40645-019-0304-z, 2019.

Sutter, H.: The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software, Dr. Dobb's Journal, 30, 2005.

The MPI Forum: MPI: A Message Passing Interface, in: Proceedings of the 1993 ACM/IEEE Conference on Supercomputing, Supercomputing '93, p. 878–883, Association for Computing Machinery, New York, NY, USA, https://doi.org/10.1145/169627.169855, 1993.

Tomita, H., Tsugawa, M., Satoh, M., and Goto, K.: Shallow Water Model on a Modified Icosahedral Geodesic Grid by Using Spring Dynamics, J. Comput. Phys., 174, 579–613, 2001.

Voosen, P.: Europe builds digital twin of Earth to hone climate forecasts, Science, 370, 16–17, https://doi.org/10.1126/science.370.6512.16, 2020.

Walker, D. W.: Standards for message-passing in a distributed memory environment, Tech. Rep. ORNL/TM-12147, Oak Ridge National Lab., TN (United States), Center for Research on Parallel Computing (CRPC), https://technicalreports.ornl.gov/1992/3445603661204.pdf (last access: 16 December 2022), 1992.

Washington, W. M., Buja, L., and Craig, A.: The computational future for climate and Earth system models: on the path to petaflop and beyond, Philos. T. Roy. Soc. A, 367, 833–846, https://doi.org/10.1098/rsta.2008.0219, 2009.

Yepes-Arbós, X., van den Oord, G., Acosta, M. C., and Carver, G. D.: Evaluation and optimisation of the I/O scalability for the next generation of Earth system models: IFS CY43R3 and XIOS 2.0 integration as a case study, Geosci. Model Dev., 15, 379–394, https://doi.org/10.5194/gmd-15-379-2022, 2022.

Zalesak, S.: Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys., 31, 335–362, 1979.

Zängl, G., Reinert, D., Ripodas, P., and Baldauf, M.: The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core, Q. J. Roy. Meteor. Soc., 141, 563–579, 2015.