Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4'888 GPUs with COSMO 5.0

Oliver Fuhrer[1], Tarun Chadha[2], Torsten Hoefler[3], Grzegorz Kwasniewski[3], Xavier Lapillonne[1], David Leutwyler[4], Daniel Lüthi[4], Carlos Osuna[1], Christoph Schär[4], Thomas C. Schulthess[5,6], and Hannes Vogt[6]

[1]Federal Institute of Meteorology and Climatology, MeteoSwiss
[2]ITS Research Informatics, ETH Zurich
[3]Scalable Parallel Computing Lab, ETH Zurich
[4]Institute for Atmospheric and Climate Science, ETH Zurich
[5]Institute for Theoretical Physics, ETH Zurich
[6]Swiss National Supercomputing Centre, CSCS

Correspondence to: Oliver Fuhrer (oliver.fuhrer@meteoswiss.ch)

Abstract. The best hope for reducing long-standing global climate model biases is through increasing the resolution to the kilometer scale. Here we present results from an ultra-high resolution non-hydrostatic climate model for a near-global setup running on the full Piz Daint supercomputer on 4888 GPUs. The dynamical core of the model has been completely rewritten using a domain-specific language (DSL) for performance portability across different hardware architectures. Physical param-
5   eterizations and diagnostics have been ported using compiler directives. To our knowledge this represents the first complete atmospheric model being run entirely on accelerators at this scale. At a grid spacing of 930 m (1.9 km), we achieve a simulation throughput of 0.043 (0.23) simulated years per day and an energy consumption of 596 MWh per simulated year. Furthermore, we propose the new memory usage efficiency metric that considers how efficiently the memory bandwidth – the dominant bottleneck of climate codes – is being used.

1   Introduction

Should global warming occur at the upper end of the range of current projections, the local impacts of unmitigated climate change would be dramatic. Particular concerns relate to the projected sea-level rise, increases in the incidence of extreme events such as heat waves and floods, and changes in availability of water resources and the occurrence of droughts (Pachauri and Meyer, 2014).

15   Currently available climate projections are mostly based on global climate models (GCMs). These models represent the coupled atmosphere-ocean-land system, and integrate the governing equations for a set of prescribed emissions scenarios. Despite significant progress during the last decades, the uncertainties are still large. For example, current estimates of the equilibrium global mean surface warming for doubled greenhouse gas concentrations range between 1.5 and 4.5 °C (Pachauri and Meyer, 2014). On regional scales and in terms of the hydrological cycle, the uncertainties are even larger. Reducing the

uncertainties of climate change projections, in order to make optimal mitigation and adaptation decisions, is thus urgent and has a tremendous economic value (Hope, 2015).

How can the uncertainties of climate projections be reduced? There is overwhelming evidence from the literature that the leading cause of uncertainty is the representation of clouds, largely due to their influence upon the reflection of incoming solar radiation (Boucher et al., 2013; Bony et al., 2015; Schneider et al., 2017). Horizontal resolutions of current global climate models are typically in the range 50-200 km. At this resolution, clouds must thus be parametrized, based on theoretical and semi-empirical considerations. Refining the resolution to kilometer-scale resolution would allow to explicitly represent deep convective clouds (thunderstorms and rain showers, e.g., Figure 1). Studies using regional climate models demonstrate that around this resolution the representation of precipitation is dramatically improved (Kendon et al., 2014; Ban et al., 2015). The representation of shallow cumulus cloud layers, which are common over significant fractions of the tropical oceans, requires even higher resolution. The US National academy of sciences has thus recommended (Bretherton, 2012) "to develop high-end global models that execute efficiently [...], enabling cloud-resolving atmospheric resolutions (2-4 km) and eddy-resolving ocean resolutions (5 km)" in the near future.

While the scientific prospects of such an undertaking are highly promising, the computational implications are appalling. Indeed, increasing the horizontal resolution from 50 to 2 km increases the computational effort by at least a factor $25^3 = 15000$. Such simulations will only be possible on future extreme-scale high performance computers. Furthermore, power constraints
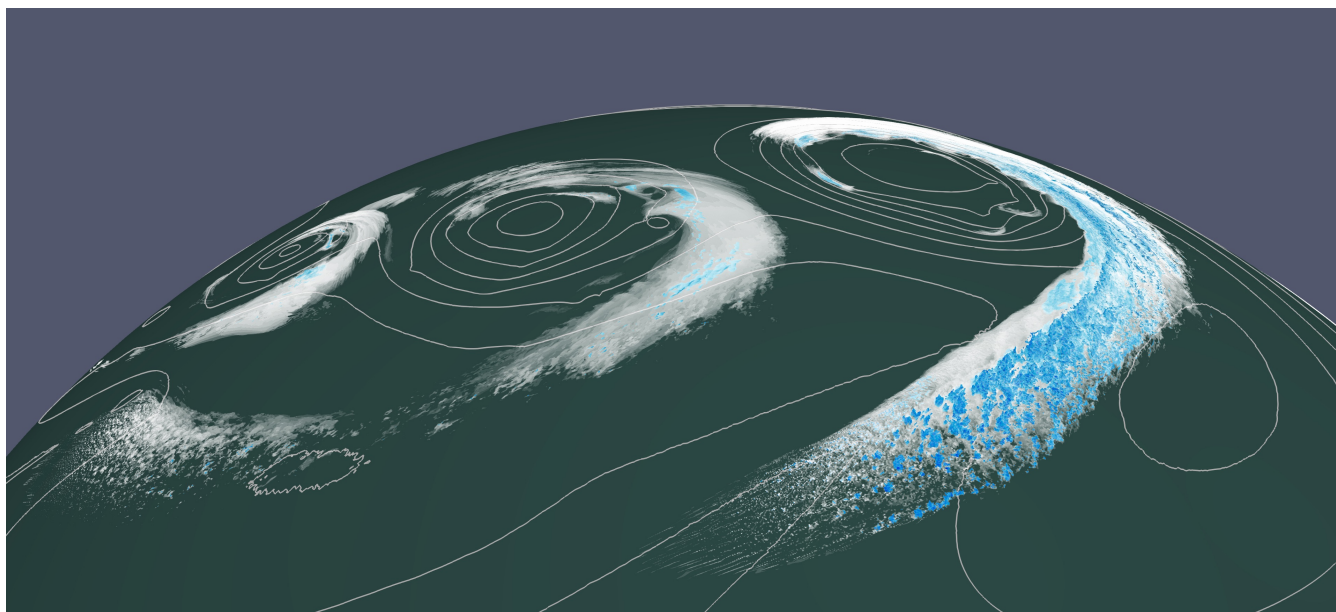
Figure 1. Visualization of the baroclinic wave at day 10 of the simulation with 930 m grid spacing. (White shading) Volume render of cloud ice, cloud water, and graupel $\geq 10^{-3}$ g/kg. (Blue shading) Isosurface of rain and snow hydrometeros $\geq 4 \cdot 10^{-2}$ g/kg. The white contours denote surface pressure.

have been driving the widespread adoption of many-core accelerators in leading edge supercomputers and the weather and climate community is struggling to migrate the large legacy simulation codes to these architectures and use them efficiently.

But what does efficient mean? While concerns of total cost of ownership of an HPC system have shifted the focus from a concern over peak floating point performance towards a concern over improving power efficiency, it is not clear what the right efficiency metric is for a full-fledged climate model. Today, floating point operations are around $100\times$ cheaper than data movement in terms of time and $1000\times$ in terms of energy, depending on where the data comes from (Borkar and Chien, 2011; Shalf et al., 2011). Thus, while focusing on floating point operations was very relevant 25 years ago, it has lost most of this relevance today. Instead, domain-specific metrics may be much more applicable to evaluate and compare application performance. An often used metric for climate models is the time-compression achieved by the simulation measured in simulated years per wall-clock day (SYPD, see Balaji et al. (2017) for a detailed discussion on metrics). For global atmospheric models, a suitable near-term target is to conduct decade-long simulations and to participate in the AMIP effort. Such simulations require a 36 year long simulation for the period 1979-2014 driven by observed atmospheric greenhouse gas concentrations and sea-surface temperatures. Within the framework of current climate modeling centers, such simulation would be feasible for a SYPD larger than about 0.2–0.3, with which the simulation would take up to several months. However, domain-specific metrics such as SYPD are very dependent on the specific problem and approximations in the code under consideration and are often hard to compare. Ideally comparisons would be performed for production quality global atmospheric models that have been extensively validated for climate simulations and cover the full (non-hydrostatic and compressible) dynamics and the entire suite of model parametrizations.

With the SYPD metric alone, it is hard to assess how efficiently a particular computing platform is used. Efficiency of use is particularly important because, at the typical scale of climate simulations, computing resources are very costly and energy intensive. Thus, running high-resolution climate simulations also faces a significant computer science problem when it comes to computational efficiency. As mentioned before, floating point efficiency is often not relevant for state-of-the-art climate codes. Not only is counting flop/s not reflecting the actual (energy) costs well, but the typical climate code has very low arithmetic intensity (the ratio of floating point operations to consumed memory bandwidth). Attempts to increase the arithmetic intensity may increase the floating point rate but it is not clear if it improves any of the significant metrics (e.g., SYPD). However, solely focusing on memory bandwidth can also be misleading. Thus, we propose *memory usage efficiency (MUE)*, a new metric that considers the efficiency of the code's implementation with respect to input/output (I/O) complexity bounds as well as the achieved system memory bandwidth.

In summary, the next grand challenge of climate modeling is refining the grid spacing of the production model codes to the kilometer-scale, as it will allow addressing long-standing open questions and uncertainties on the impact of anthropogenic effects for the future of our planet. Here, we address this grand challenge and demonstrate the first simulation of a production-level atmospheric model, delivering 0.23 (0.043) SYPD at a grid-spacing of 1.9 km (930 m), sufficient for AMIP[1]-type simulations. Further, we evaluate the efficiency of these simulations using a new memory usage efficiency metric.

---

[1]atmospheric model inter-comparison (AMIP), which is part of the climate-model inter-comparison project (CMIP6, see Eyring et al. (2016))

2 Current State of the Art

Performing global kilometer-scale climate simulations is an ambitious goal (Palmer, 2014), but a few kilometer-scale landmark simulations have already been performed. While arguably not the most relevant metric, many of the studies have reported sustained floating point performance. In 2007, Miura et al. (2007) performed a week-long simulation with a horizontal grid spacing
5 of 3.5 km with the Nonhydrostatic Icosahedral Atmospheric Model (NICAM) on the Earth Simulator, and in 2013 Miyamoto et al. (2013) performed a 12-hour-long simulation at a grid spacing of 870 m on the K-computer achieving 230 Tflop/s double precision performance. In 2014, Skamarock et al. (2014) performed a 20-day-long simulation with a horizontal grid spacing of 3 km with the Model for Prediction Across Scales (MPAS) and later, in 2015, participated in the Next Generation Global Prediction System (NGGPS) model inter-comparison project (Michalakes et al., 2015) at the same resolution and achieved
10 0.16 SYPD on the full NERSC Edison system. In 2015, Bretherton and Khairoutdinov (2015) simulated several months of an extended aquaplanet channel at a grid spacing of 4 km using the System for Atmospheric Modeling (SAM). Yashiro et al. (2016) were the first to deploy a weather code on the GPU-accelerated TSUBAME system. The fully rewritten NICAM model sustained a double-precision performance of 60 Tflop/s on 2560 GPUs of the TSUBAME 2.5 supercomputer. In 2016, Yang et al. (2016) implemented a fully implicit dynamical core at 488 m grid spacing in a β-plane channel achieving 7.95 Pflop/s on
15 the TaihuLight supercomputer.

The optimal numerical approach for high-resolution climate models may depend on the details of the target hardware architecture. For a more thorough analysis, the physical propagation of information in the atmosphere has to be considered. While many limited-area atmospheric models use a filtered set of the governing equations that suppresses sound propagation, these approaches are not precise enough for global applications (Davies et al., 2003). Thus, the largest physical group velocity to
20 face in global atmospheric models is the speed of sound. The speed of sound in the atmosphere amounts to between 280 and 360 m/s. Thus in a time span of an hour, the minimum distance across which information needs to be exchanged amounts to about 1500 km, corresponding to a tiny fraction of 1.4% of the earth's surface. However, many numerical schemes exchange information at much larger rates. For instance, the popular pseudo-spectral methodology (e.g., ECMWF (2016)) requires Legendre and Fourier transforms between the physical grid and the spherical harmonics, and thus couples globally at each time
25 step. Similarly, semi-Lagrangian semi-implicit time-integration methods require the solution of a Helmholtz-type elliptical equation (Davies et al., 2005), which implies global communication at each time step. These methods have enabled fast and accurate solutions at intermediate resolution in the past, but they are likely not suited for ultra-high resolution models, as the rate of communication typically increases proportional to the horizontal mesh size. Other approaches use time integration methods with only locally implicit solves (e.g., Giraldo et al. (2013)), where they try to retain the advantages of fully implicit
30 methods but only require nearest neighbor communication.

The main advantage of implicit and semi-implicit approaches is that they allow large acoustic Courant numbers $\alpha_c = c\Delta t/\Delta x$ where c denotes the speed of sound and Δt and Δx the time step and the grid spacing. For instance, Yang et al. (2016) use an acoustic Courant number up to 177, i.e., their time step is 177 times larger than in a standard explicit integration

(this estimate is based on the $\Delta x = 488$ m simulation with $\Delta t = 240$ s). Such a large time step may be chosen, as the sound propagation is not relevant for weather phenomena.

However, although implicit methods are unconditionally stable (stable irrespectively of the time step used), there are other limits to choosing the time step. In order to appropriately represent advective processes with typical velocities up to 100 m/s and associated phase changes (e.g., condensation and fallout of precipitation), numerical principles dictate an upper limit to the advective Courant number $\alpha_u = |u|\Delta t/\Delta x$, where $|u|$ denotes the largest advective wind speed, e.g., (Ewing and Wang, 2001). The specific limit for α_u depends on the numerical implementation and time-stepping procedures. For instance, semi-Lagrangian schemes may produce accurate results for values of α_u up to 4 or even larger. For most standard implementations, however, there are much more stringent limits, often requiring that $\alpha_u \leq 1$. For the recent study of Yang et al. (2016), who used a fully implicit scheme with a time step of 240 s, the advective Courant number reaches values up to $\alpha_u = 4.2$ and 17.2 for the $\Delta x =$2 km and 488 m simulation, respectively. Depending upon the numerical approximation, such a large Courant number will imply significant phase errors (Durran, 2010) or even a reduction in effective model resolution (Ricard et al., 2013). In order to produce accurate results, the scheme would require a significantly smaller time step and reducing the time step with decreasing grid spacing. For the NGGPS inter-comparison the hydrostatic IFS model used a time step of 120 s at 3.125 km (Michalakes et al., 2015), the regional semi-implicit, semi-Lagrangian model fully non-hydrostatic model MC2 used a timestep of 30 s at 3.0 km (Benoit et al., 2002), and MeteoFrance in their semi-implicit AROME model use a time step of 45 s and 60 s for their 1.3 km and 2.5 km implementations, respectively. *In summary, we conclude that even for fully-implicit, global, convection-resolving climate simulations at ~1–2 km grid spacing, a timestep larger than 40–60 s cannot be considered a viable option.*

In the current study we use the split-explicit time stepping scheme with an underlying Runge-Kutta time step (Wicker and Skamarock, 2002) of the COSMO model (see Section 3.1). This scheme uses sub-time-stepping for the fast (acoustic) modes with a small time step $\Delta\tau$, and explicit time stepping for all other modes with a large time step $\Delta t = n\Delta\tau$. Most of the computations are required on the large time step, with $\alpha_u \leq 2$, depending on the combination of time-integration and advection scheme. Unlike with semi-implicit, semi-Lagrangian and implicit schemes, the approach does not require solving a global equation, and all computations are local (i.e., vertical columns exchange information merely with their neighbors). The main advantage of this approach is that it exhibits – at least in theory – perfect weak scaling. This also applies for the communication load per sub-domain, when applying horizontal domain decomposition.

3 Methods

3.1 Model description

For the simulations presented in this paper, we use a refactored version 5.0 of the regional weather and climate code developed by the Consortium for Small-Scale Modeling (COSMO) (COSMO, 2017; Doms and Schättler, 1999; Steppeler et al., 2002) and – for the climate mode – the CLM-Community (CLM-Community, 2017). At kilometer-scale resolution, COSMO is used for numerical weather prediction (Richard et al., 2007; Baldauf et al., 2011) and has been thoroughly evaluated for climate
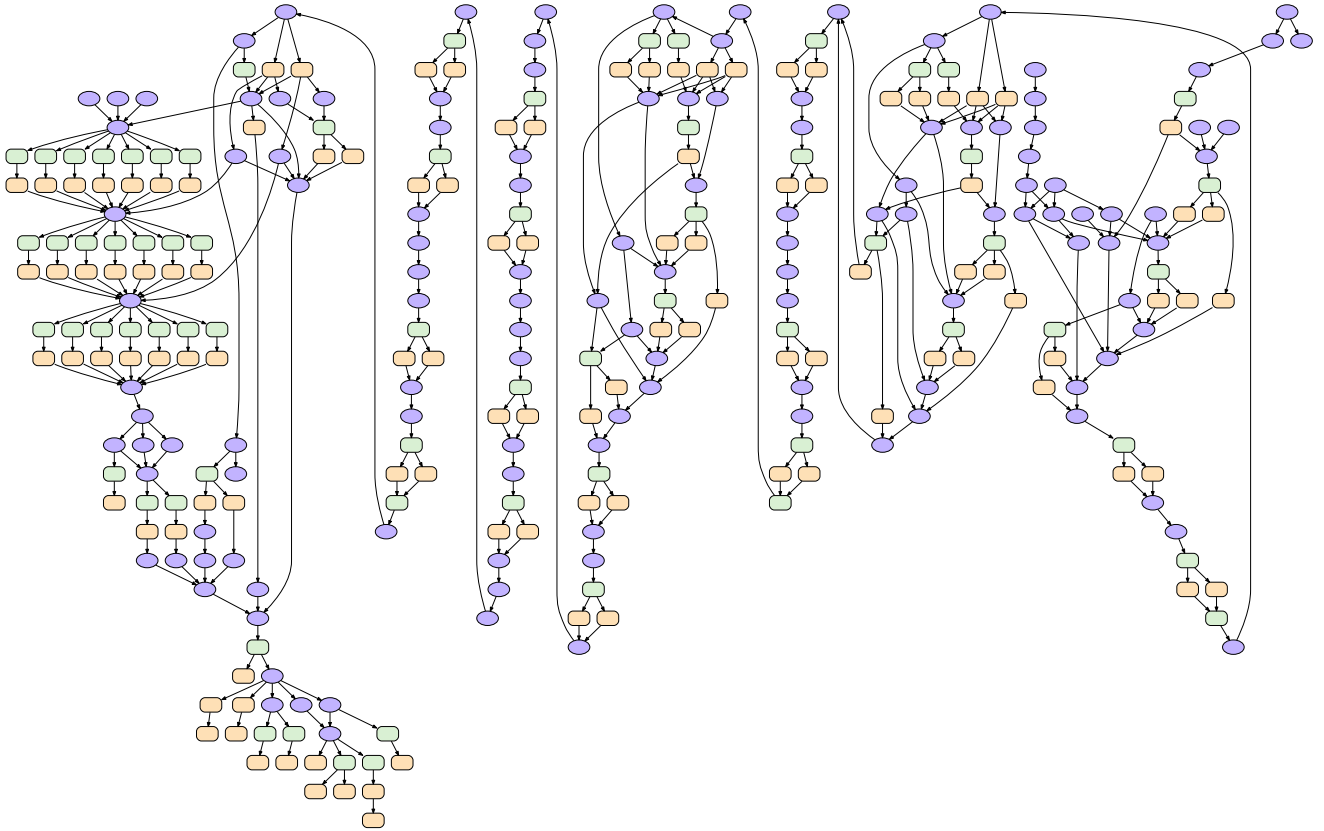
Figure 2. Computational Directed Acyclic Graph (CDAG) of the COSMO dynamical core. The CDAG shows complex stencil operations (a kernel) that produce multiple fields (blue ellipses) as well as MPI halo exchanges (green rectangles) and boundary condition operations (orange rectangles).

simulations in Europe (Ban et al., 2015; Leutwyler et al., 2017). The COSMO model is based on the thermo-hydrodynamical equations describing non-hydrostatic, fully compressible flow in a moist atmosphere. It solves the fully compressible Euler equations using finite difference discretization in space (Doms and Schättler, 1999; Steppeler et al., 2002). For time-stepping, it uses a split-explicit three-stage second-order Runge-Kutta discretization to integrate the prognostic variables forward in time

5   (Wicker and Skamarock, 2002). For horizontal advection, a fifth-order upwind scheme is used for the dynamic variables and a Bott scheme (Bott, 1989) is used for the moisture variables. The model includes a full set of physical parametrizations required for real-case simulations. For this study, we use a single-moment bulk cloud-microphysics scheme that uses five species (cloud water, cloud ice, rain, snow and graupel) described in Reinhardt and Seifert (2006). For the full physics simulations, additionally a radiation scheme (Ritter and Geleyn, 1992), a soil model (Heise et al., 2006), and a sub-grid scale turbulence scheme

10   (Raschendorfer, 2001) is switched on.

ARE

The COSMO model is a regional model and physical space is discretized in a rotated latitude / longitude / height coordinate system and projected onto a regular, structured, three-dimensional grid (IJK). In the vertical a terrain-following coordinate allows prescribing arbitrary topography. The spatial discretization applied to solve the governing equations generate so called stencil computations (operations that require data from neighbor grid points). Due to the strong anisotropy of the atmosphere, implicit methods are employed in the vertical direction, as opposed to the explicit methods applied to the horizontal operators. The numerical discretization yields a large number of mixed compact horizontal stencils and vertical implicit solvers, strongly connected via the data dependencies on the prognostic variables. Figure 2 shows the data dependency graph of the computational kernels of the dynamical core of COSMO used in this setup, where each kernel encompasses a complex set of fused stencil operations in order to maximize the data locality of the algorithms. Maximizing the data locality of these stencil computations is crucial to optimize the time-to-solution of the application.

To enable COSMO on hybrid high performance computing systems with GPU-accelerated compute nodes, we rewrote the dynamical core of the model, that implements the solution to the non-hydrostatic Euler equations, from Fortran to C++ (Fuhrer et al., 2014). This enabled us to introduce a new C++ template library-based domain specific language (DSL) we call STELLA (Gysi et al., 2015a), to provide a performance-portable implementation for the stencil algorithmic motifs by abstracting hardware dependent optimization. Specialized backends of the library produce efficient code for the target computing architecture. Additionally the DSL supports an analysis backend that records the access patterns and data dependencies of the kernels shown in Figure 2. This information is then used to determine the amount of memory accesses and assess the memory utilization efficiency. For GPUs, the STELLA backend is written in CUDA, and other parts of the refactored COSMO implementation use OpenACC directives (Lapillonne and Fuhrer, 2014).

Thanks to this refactored implementation of the model and the STELLA DSL, COSMO is the first fully capable weather and climate model to go operational on GPU accelerated supercomputers (Lapillonne et al., 2016). In the simulations we analyze here, the model was scaled to nearly 5000 GPU-accelerated nodes of the Piz Daint supercomputer[2] at the Swiss National Supercomputing Centre. To our knowledge, COSMO is still the only production-level weather and climate model available capable of running on GPU-accelerated hardware architectures.

3.2 Hardware description

The experiments were performed on the hybrid partition of the Piz Daint supercomputer, located at the Swiss National Supercomputing Centre (CSCS) in Lugano[3]. At the time of writing this supercomputer consisted of a multi-core partition of 1431 Cray XC40 nodes with two Intel Xeon E5-2695 v4 (code name Broadwell) processors, which were not used in this study, as well as the hybrid partition of 5320 Cray XC50 nodes. These hybrid nodes are equipped with an Intel E5-2690 v3 CPU (code name Haswell) and a PCIe version of the NVIDIA Tesla P100 GPU (code name Pascal) with 16 GB second generation high bandwidth memory (HBM2). The nodes of both partitions are interconnected in one fabric based on Aries technology in a Dragonfly topology (Alverson et al., 2012).

---

[2]See http://www.cscs.ch/computers/piz_daint/index.html for more information.
[3]See http://www.cscs.ch/computers/piz_daint/index.html for more information.

3.3 Energy Measurements

We measure energy to solution of our production level runs on Piz Daint using the methodology established and described in detail by Fourestey et al. (2014). The resource utilization report provided on Cray systems for a job provides the total energy (E_n) consumed by each application run on the compute nodes. The total energy is then computed using

$$E_{tot} = \frac{E_n + N/4 \times 100\,\text{W} \times \tau}{0.95}, \tag{1}$$

where τ is the wall time for the application, the $N/4 \times 100\,\text{W} \times \tau$ term accounts for the 100 W per blade contribution from the Aries interconnect and the 0.95 on the denominator adjusts for AC/DC conversion.

3.4 Simulation setup and verification

Either say
"The JW baroclinic wave testcase"
or say "a baroclinic wave"
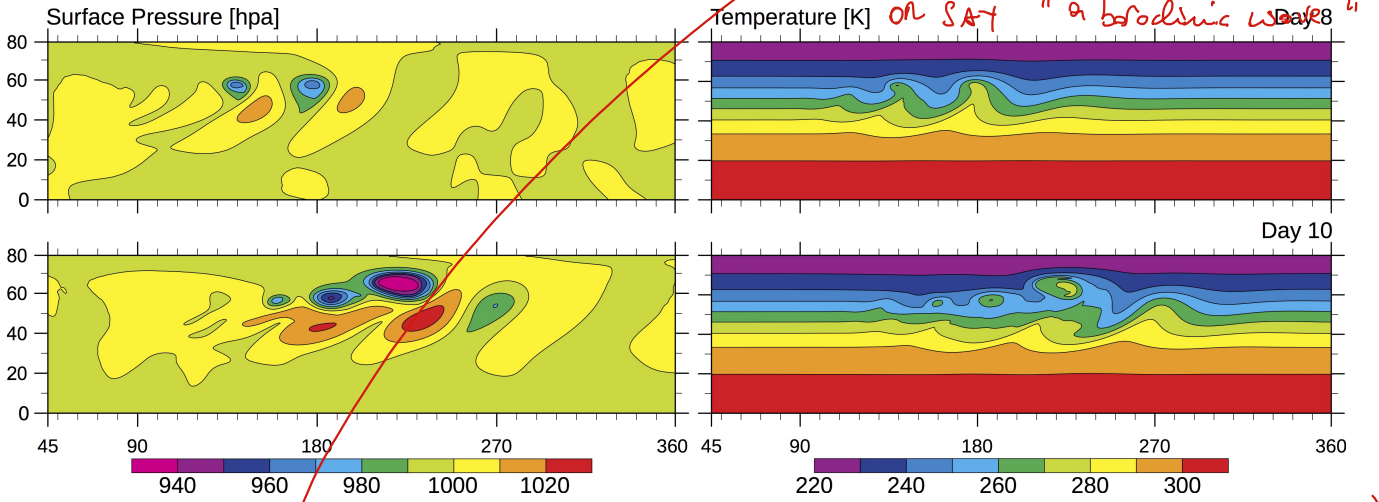
Figure 3. Evolution of the baroclinic wave in a dry simulation with 47 km grid spacing on day 8 and 10: (Left) Surface Pressure and (right) temperature on the 850 hPa pressure level (roughly 1.5 km above sea level).

When pushing ahead the development of global high-resolution climate models, there are two complementary pathways. First, one can refine the resolution of existing global climate models (Miura et al., 2007). Second, one may alternatively try to expand the computational domain of high-resolution limited-area models towards the global scale (Bretherton and Khairoutdinov, 2015). Here we choose the latter and develop a near-global model from the limited-area high-resolution model COSMO.

We perform near-global simulations for a computational domain that extends to a latitude band from 80°S to 80°N, which covers 98.4% of the surface area of planet Earth. The simulation is inspired by the testcase used by then winner of the 2016 Gordon Bell Prize (Yang et al., 2016).

The simulations are based on an idealized baroclinic wave test (Jablonowski and Williamson, 2006), which can be considered a standard benchmark for dynamical cores of atmospheric models. The test describes the growth of initial disturbances in a dynamically unstable westerly jet stream into finite-amplitude low and high-pressure systems. The development includes a

rapid transition into a nonlinear regime accompanied by the formation of sharp meteorological fronts, which in turn trigger the formation of complex cloud and precipitation systems.

The setup uses a two-dimensional (latitude-height) analytical description of a hydrostatically balanced atmospheric base state with westerly jet streams below the tropopause, in both hemispheres. On this balanced initial state, a large-scale local Gaussian perturbation is applied, which triggers the formation of a growing baroclinic wave in the northern hemisphere, evolving over the course of several days (Figure 3). To allow moist processes, the dry initial state is extended with a moisture profile (Park et al., 2013) and the parametrization of cloud-microphysical processes is activated.

The numerical problem is discretized on a latitude-longitude grid with up to 36000×16001 horizontal grid points for the 930 m simulation. In zonal direction the domain is periodic and at 80° North/South confined by boundary conditions, relaxing the evolving solution against the initial conditions in a 500 km wide zone. The vertical direction is discretized using 60 stretched model levels, spanning from the surface to the model top at 40 km. The respective layer thickness widens from 20 m at the surface to 1.5 km near the domain top.
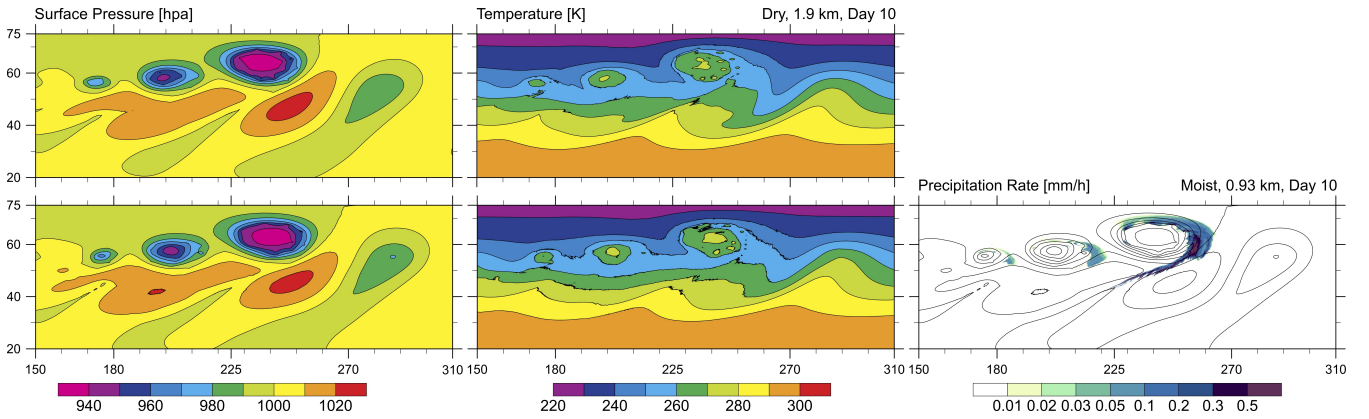
Figure 4. Output of the baroclinic wave (day 10): (Top) dry simulation with 1.9 km grid spacing, (bottom) moist simulation with 0.93 km grid spacing. (Left) surface pressure, (middle) temperature on the 850 hPa pressure level and (right) precipitation.

For the verification against previous dry simulations, a simulation at 47 km grid spacing is used. The evolution of the baroclinic wave (Figure 3) very closely follows the solution originally found by Jablonowski and Williamson (2006, see their Figure 5). At day 8 of the simulation, three low pressure systems with frontal systems have formed, and at day 10 wave breaking is evident. At this time, the surface temperature field shows cut-off warm-core cyclonic vortices.

The evolution in the moist and the dry simulation at very high resolution is shown in Figure 4. The high-resolution simulations reveal the onset of a secondary (likely barotropic) instability along the front, and the formation of small-scale warm-core vortices with a spacing of up to 200-300 km. The basic structure of these vortices is already present in the dry simulation, but they exhibit considerable intensification and precipitation in the moist case. Formation of a large series of secondary vortices is sometimes observed in maritime cases of cyclogenesis (Fu et al., 2004; Ralph, 1996), but appears to be a rather rare phenom-

ena. However, it appears that cases with one or a few secondary vortices are not uncommon and they may even be associated with severe weather (Ludwig et al., 2015).

The resulting cloud pattern is dominated by the comma-shaped precipitating cloud that forms along the cold and occluded fronts of the parent system (Figure 1). The precipitation pattern is associated with stratiform precipitation in the head of the cloud and small patches with precipitation rates exceeding 5 mm/h in their tail, stemming from small embedded convective cells. Looking closely at the precipitation field (Figure 4 right-hand panel), it can be seen that the secondary vortices are co-located with small patches of enhanced precipitation.

4 Efficiency metric

In the past, the prevalent metric to measure performance of an application was the number of floating point operations executed per second (flop/s). It used to pay off to minimize the number of required floating point operations in an algorithm and to implement the computing system in such a way that a code would execute many such operations per second. Thus, it made sense to assess application performance with the flop/s metric. In the meantime, the world of supercomputing has changed entirely. Floating point operations are cheap. They cost about a factor 1000 less if measured in terms of energy needed to move operands between memory and registers, and they execute several 100 times faster compared to the latency of memory operations. Thus, algorithmic optimization today has to focus on minimizing data movements, and it may even pay off to re-compute certain quantities if this avoids data movements. In fact, a significant part of the improvements in time to solution in the refactored COSMO code are due to re-computation of variables that were previously stored in memory—the original code was written for vector supercomputers in the 1990s. On today's architectures, it may even pay off to replace a floating point minimal sparse algorithm with a block-sparse algorithm, into which many trivial zero-operations have been introduced (Hutter et al., 2014). Using the flop/s metric to characterize application performance would be very misleading in these cases.

Assessing performance of an application simply in terms of sustained memory bandwidth, which is measured in bytes per second, would be equally deceiving. For example, by storing many variables in memory, the original implementation of COSMO introduces an abundance of memory movements that boost the sustained memory bandwidth but are inefficient on modern processor architectures, since these movements cost much more than the re-computation of these variables.

Furthermore, full bandwidth utilization on modern accelerator architectures, such as the GPUs used here, requires very specific conditions. Optimizing for bandwidth utilization can lead the application developer down the wrong path because unaligned, strided, or random accesses can be intrinsic to an underlying algorithm and severely impact the bandwidth (Lee et al., 2010). Optimizations to improve alignment or reduce randomness of memory accesses may introduce unnecessary memory operations that could be detrimental to time-to-solution or energy efficiency.

Thus, in order to properly assess the performance of an application running on today's supercomputers, one needs to directly consider data transfers. Here we propose how this can be done in practice, and present the resulting memory usage efficiency *(MUE) metric.*

Our proposal is motivated by the full-scale COSMO runs we perform here, where 74% of the total time is spent in local stencil computations. The dependencies between stencils of these computations in complex *stencil programs* can be optimized with various inlining and unrolling techniques (Gysi et al., 2015a). Thus, to be efficient, one needs to achieve maximum spatial and temporal data reuse to minimize the number of data movement operations to perform only necessary transactions at highest

5 bandwidth.

In order to assess the efficiency in memory usage of an implementation on a particular machine, one needs to compare the actual number of data transfers executed, which we denote with D, with the necessary data transfers Q of the algorithm. Q is the theoretical lower bound (Hong and Kung, 1981) of number of memory accesses required to implement the numerical algorithm.

10 The MUE can be intuitively interpreted as how well the code is optimized both for data locality and bandwidth utilization, i.e., if MUE = 1, the implementation reaches the memory movement lower bound of the algorithm *and* performs all memory transfers with maximum bandwidth. Formally:

$$MUE = \text{I/O efficiency} \cdot \text{BW efficiency} = \frac{Q}{D} \cdot \frac{B}{\hat{B}} \tag{2}$$

Where B and $\hat{B}$ represents the bandwidth achieved by an implementation and maximum achievable bandwidth, respectively.

15 In order to compute the MUE for COSMO, we developed a performance model that combines the theoretical model from Hong and Kung (1981), hypergraph properties and graph partitioning techniques to estimates the necessary data transfer, Q, from the CDAG information of COSMO. By partitioning the CDAG into subcomputations that satisfies certain conditions imposed by the architecture, this model asses the theoretical minimum amount of memory transfers by maximizing the data locality inside the partitions. To approximate the number of actual transfers D, we use the same technique to evaluate the

20 quality of current COSMO partitioning. The values B and $\hat{B}$ were measured empirically - the former one by profiling our application, the latter by a set of microbenchmarks.

The details of how to determine the MUE for COSMO are given in Appendix A. The MUE metric cannot be used to compare different algorithms, but is a measure of the efficiency of an implementation of a particular algorithm on a particular machine, i.e. how much data locality is preserved and what ist he achieved bandwidth. It thus complements other metrics such as SYPD

25 and may replace popular metrics such as flop/s or memory bandwidth.

5 Performance Results

To establish a performance baseline for global kilometer-scale simulations we here present a summary of the key performance metrics for the simulations at 930 m, 1.9 km, and 47 km grid spacing, as well as a study of weak and strong scalability.
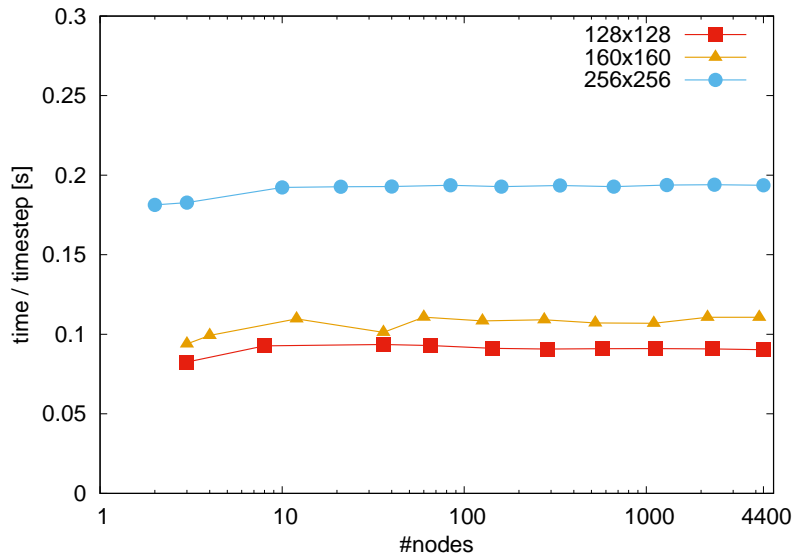
Figure 5. Weak scalability on the hybrid P100 Piz Daint nodes, per COSMO time step of the dry simulation.

5.1 Weak Scalability

had been

The GPU version of COSMO was so far only scaled up to 1000 nodes of the Piz Daint supercomputer (Fuhrer et al., 2014), while the full machine – at the time of the experiment – provides 4932 nodes. The scaling experiments (weak and strong) were performed with the dry simulation setup. Including a cloud microphysics parametrization, as used in the moist simulation
5   (Figure 4), increases time to solution by about 10%.

Figure 5 shows weak scaling for three per-node domain sizes ranging from 128×128 to 256×256 grid points in the horizontal, while keeping the size in the vertical direction fixed at 60 grid points. In comparison, on 4888 nodes, the high-resolution simulations at 930 m and 1.9 km horizontal grid spacing correspond to a domain size per node of about 346×340 and 173×170.[4] The models shows excellent weak scalability properties up to the full machine, which can at least partially be
10   explained by the next-neighbor halo-exchange pattern. This property reduces the complexity of COSMO's scalability on Piz Daint to the strong scaling behavior of the code. Essentially, the number of grid points per node determines the achievable time to solution of a given problem.

5.2 Strong Scalability

From earlier scaling experiments of the STELLA library (Gysi et al., 2015b) and the GPU version of COSMO (Fuhrer et al.,
15   2014; Leutwyler et al., 2016) it is known that, for experiments in double precision on Tesla K20x, linear scaling is achieved as long as the number of grid points per node exceeds about 64×64 to 128×128 grid points per horizontal plane. In comparison,

---

[4]The exact domain size of these simulations is slightly different on each node due to the domain decomposition.
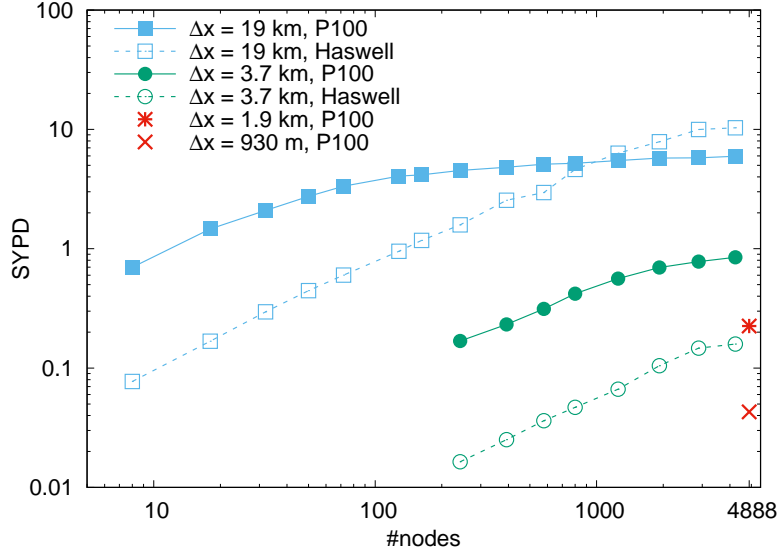
Figure 6. Strong scalability on Piz Daint. (Filled symbols) On P100 GPUs, and (empty symbols) on Haswell CPUs, using 12 MPI ranks per node.

$\langle \Delta x \rangle$	#nodes	Δt [s]	SYPD	MWh/SY	gridpoints
930 m	4888	6	0.043	596	3.46×10^{10}
1.9 km	4888	12	0.23	97.8	8.64×10^{9}
47 km	18	300	9.6	0.099	1.39×10^{7}

Table 1. Time compression (SYPD) and energy cost (MWh/SY) for three moist simulations. At 930 m grid spacing obtained with a full 10d simulation, at 1.9 km from 1000 steps, and at 47 km from 100 steps

mixed precision measurements on Tesla P100 already start to saturate at a horizontal domain size of about 200×200 grid points per node (Figure 6), corresponding to about 32 nodes for a 19 km setup and about 1000 nodes for a 3.7 km setup. Since with the 930 m and 1.9 km setup we are already in (930 m) or close to (1.9 km) the linear scaling regime on the full machine, we here chose a coarser horizontal grid spacing of 3.7 km and 19 km. The lower limit on the number of nodes is given by the GPU memory of 16 GB. In addition to the GPU benchmarks (filled symbols), we measured the performance with the CPU version of COSMO (empty symbols), using 12 MPI ranks per CPU, i.e., one MPI rank per Haswell core[5]. Exceeding 1000 nodes (38×38 grid points per node) execution on CPUs yields a shorter time to solution on CPUs than on GPUs.

5.3 Time to Solution

On 4888 nodes, a 10-day-long moist simulation at 930 m grid spacing required a wall time of 15.3 h at a rate of 0.043 SYPD (Table 1), including a disk-I/O[6] load of 5 3D-fields and 7 2D-fields, written periodically every 12h of the simulation. Short benchmark simulations at 1.9 km and 47 km, integrated for 1000 and 100 time steps respectively, yield 0.23 SYPD and 9.6 SYPD. While for the 47 km setup scalability already starts to saturate, even with the 18 nodes reported (Table 1), the high resolution simulations, with an approximate per-node domain size of $346 \times 340 \times 60$ and $173 \times 170 \times 60$ respectively, are still in a regime of good scalability. We have also conducted a 1.9 km simulation with a full set of physical parametrizations switched on and this increases the time-to-solution by 27% relative to the moist simulations presented here.

In conclusion, the results show that AMIP-type simulations at horitzontal grid spacings of 1.9 km using a full-fledged atmospheric model are already feasible on Europe's highest ranking supercomputer. In order to reach resolutions of 1 km a further reduction of time-to-solution of at least a factor 5 is required.

In the following, we attempt a comparison of our results at 1.9 km against the performance achieved by Yang et al. (2016) in their 2 km simulation (as their information for some of the other simulations is incomplete). They argue that for an implicit solver the timestep can be kept at 240 s independent of the grid spacing and report (cf. their Figures 7 and 9) values of 0.57 SYPD for a grid spacing of 2 km, on the full TaihuLight system. As argued in Section 2, such large timesteps are not admissible for global climate simulations resolving convective clouds (even when using implicit solvers), and a maximum timestep of 40-80 s would very likely be needed; this will decrease their SYPD by a factor 3 to 6. Furthermore, their simulation covers only 32% of the Earth's surface (18°N to 72°), but uses twice as many levels; this would further reduce their SYPD by a factor 1.5. Thus we estimate that the simulation of Yang et al. (2016) at 2 km would yield 0.124 to 0.064 SYPD when accounting for these differences. In comparison, our simulation at 1.9 km yields 0.23 SYPD, i.e. is faster by at least a factor 2.5. Note that this estimate does not account for additional simplifications in their study (neglection of microphysical processes, spherical shape of the planet, and topography). In summary, while a direct comparison with their results is difficult, we argue that our results can be used to set a realistic baseline for production-level GCM performance results and represent an improvement of a factor 2 or more with respect to previous results.

5.4 Energy to Solution

Based on the power measurement (cf. Section 3.3) we provide the energy cost of full scale simulations (Table 1) using the energy cost unit MWh per simulation year (MWh/SY). The 10-day-long simulation at 930 m grid spacing running on 4888 nodes requires 596 MWh/SY while the cost of the simulation at 1.9 km on 4888 nodes is 97.8 MWh/SY. For comparison the coarse resolution at 47 km simulation on a reduced number of nodes (18) requires only 0.01 MWh/SY.

---

[5]The CPU measurements were performed on the hybrid partition of Piz Daint as well, since the multi-core partition is much smaller.

[6]The standard I/O routines of COSMO require global fields on a single node, which typically do not provide enough storage to hold a global field. To circumvent this limitation, a binary I/O mode, allowing each node to write its output to the file-system, was implemented.

level	METIS Q	COSMO D	no-opt $\hat{D}$
registers	$1.51 \cdot 10^9$	$1.72 \cdot 10^9$	$2.6 \cdot 10^9$
shared memory	64800	107600	229120
L2 cache	1023	1160	2341

Table 2. Data transfer cost estimations of the dynamical core for a theoretical lower bound (METIS), dynamical core implementation using the STELLA library (COSMO) and non-optimized dynamical core implementation (no-merging).

metric	optimized	not optimized	ratio
time per step	0.16s	0.25s	**0.64**
estimated MUE	0.67	0.44	**0.65**

Table 3. Performance model verification results. Measured dynamical core time step execution time for the STELLA implementation optimized for data locality and the non-optimized implementation compared against the corresponding MUE metric.

A 30 year AMIP-type simulation (with full physics) at a horizontal grid spacing of 1 km on the Piz Daint system would take 900 days to complete, imply an energy cost of approximately 22 GWh - which approximately corresponds to the consumption of 6500 households during one year.

Again, we attempt a comparison with the simulations performed by Yang et al. (2016). The Piz Daint system reports a peak power draw of 2052 kW when running the HPL benchmark. The sustained power draw when running the 930 m simulation amounted to 1059.7 kW, thus 52% of the HPL value. The TaihuLight system reports a sustained power draw for the HPL benchmark of 15371 kW (top, 2017). While Yang et al. (2016) do not report power consumption of their simulations, we expect the simulations on Piz Daint to be at least 5 times more power efficient, even when assuming similar achieved SYPD (see above).

5.5 Simulation efficiency

In view of the energy consumption, it is paramount to consider the efficiency of the simulations executed. We do this here by considering the Memory Transfer Efficiency metric introduced in Section 4.

To estimate a solution for the optimization problem described in Appendix, Eqn. (A1), we use the METIS library (Karypis and Kumar, 2009). The results are presented in Table 2. The **METIS** Q column is the approximation of the lower bound Q obtained from the performance model using the METIS library. The **COSMO** D column is the evaluation of current COSMO partitioning in our model. The **no-opt** $\hat{D}$ column shows the amount of data transfers of COSMO if no data locality optimization techniques are applied, like in the original Fortran version of the code. Since the original CDAG is too large for the minimization problem of the performance model, three different simplified versions of the CDAG, that focus on the accesses to three different layers of the memory hierarchy of the GPU are studied: registers, shared memory, and L2 cache.

The model shows how efficient COSMO is, in terms of data transfers – it generates only 14%, 66%, and 13% more data transfers than the lower bound in the register, shared memory and L2 cache layers, respectively. The sophisticated data locality optimization techniques of the STELLA implementation of the dynamical core of COSMO result in close to optimal data reuse. Because on the P100, all memory accesses to/from DRAM go through the L2 unit, we focus on the efficiency of this unit such that

$$MUE = \frac{Q_{L2}}{D_{L2}} \cdot \frac{B}{\hat{B}} = 0.88 \cdot 0.76 = 0.67,$$

where D_{L2} and Q_{L2} stand for estimated number of main memory operations and its lower bound, respectively.

The model also can estimate the efficiency of our optimizations. Assuming that we can reach the peak achievable bandwidth $\hat{B}$ if we perform no data locality optimizations ($\hat{D}$), then:

$$MUE_{no\_opt} = \frac{Q_{L2}}{\hat{D}_{L2}} \cdot \frac{\hat{B}}{\hat{B}} = 0.44$$

It can be seen, that

$$\frac{MUE}{MUE_{no\_opt}} = \frac{0.67}{0.44} = 1.52$$

5   This shows the importance of data locality optimizations - the optimized implementation is more than 50% faster than a potential version that achieves peak bandwidth while using no data locality techniques.

To validate the model results, we have conducted single-node runs with and without data locality optimizations. The fact that the two ratios in the third column of Table 3 agree is testimony to the high precision of the performance model.

6   Conclusions

10   With the work presented here, we are setting a new baseline for full-fledged km-scale climate simulations at global scale. Our implementation of the COSMO model that is used for production-level numerical weather predictions at MeteoSwiss has been scaled to the full system on 4888 nodes of Piz Daint, a GPU accelerated Cray XC50 supercomputer at the Swiss National Supercomputing Centre (CSCS). The dynamical core has been fully rewritten in C++ using a DSL that abstracts the hardware architecture for the stencil algorithmic motifs and enables sophisticated tuning of data-movements. Optimized backends are

15   available for multi-core processors with OpenMP, GPU accelerators with CUDA, and for performance analysis.

The code shows excellent strong scaling up to the full machine size when running at a grid spacing of 4 km and below, on both the P100 GPU accelerators and the Haswell CPU. For smaller problems, e.g., at coarser grid spacing of 47 km, the GPUs run out of parallelism and strong scalability is limited to about 100 nodes, while the same problem continues to scale on multi-core processors to 1000 nodes. Weak scalability is near optimal to the full size of the machine.

20   The simulations performed here are based on the idealized baroclinic wave test (Jablonowski and Williamson, 2006), which is part of the standard procedure to test global atmospheric models. Our results validate against the original solution published by Jablonowski and Williamson (2006).

We measured time to solution in terms of simulated years per wall clock day (SYPD) for a near global simulation on a latitude band from 80°S to 80°N that covers 98.4% of planet Earth's surface. At extreme scale running on 4888 P100 GPUs of Piz Daint, we measured 0.23 SYPD at 1.9 km grid spacing. This performance is adequate for numerical weather predictions and 10-year scale climate studies. Simulations at this resolution had an energy cost of 97.8 MWh/SY.

In a moist simulation with 930 m horizontal grid spacing, we observed the formation of frontal precipitating systems, containing embedded explicitly-resolved convective motions, and additional cut-off warm-core cyclonic vortices. The explicit representation of embedded moist convection and the representation of the resolved instabilities exhibits physically different behavior from coarser-resolution simulations. This is testimony to the usefulness of the high-resolution approach, as a much expanded range of scales and processes is simulated. Indeed it appears that for the current test case, the small vortices have not previously been noted, as the test case appears to converge for resolutions down to 25 km (Ullrich et al., 2015), but they clearly emerge at kilometer-scale resolution.

These results serve as a baseline benchmark for global climate model applications. For the 930 m experiment we achieved 0.043 SYPD on 4888 GPU accelerated nodes, which is approximately one seventh of the 0.2–0.3 SYPD required to conduct AMIP-type simulations. Energy cost for simulations at this horizontal grid spacing was 596 MWh/SY.

Our work was inspired by the dynamical core solver that won the 2016 Gordon Bell award at the Supercomputing conference (Yang et al., 2016). The goal of this award is to reward high performance achieved in the context of a realistic computation. A direct comparison with the results reported there is difficult, since Yang et al. (2016) were running a research version of a dynamical core solver and COSMO is a full-fledged regional weather and climate model. Nevertheless, our analysis indicates that our benchmarks represent an improvement both in terms of time to solution and energy to solution. As far as we know, our results represent the fastest (in terms of application throughput measured in SYPD at 1 km grid spacing) simulation of a production-ready, non-hydrostatic climate model on a near-global computational domain.

In order to reach 3-5 SYPD performance necessary for long climate runs, simulations would be needed that run 100 times faster than the baseline we set here. Given Piz Daint with NVIDIA P100 GPUs was a multi-petaflops system, the 1 km-scale climate runs at 3-5 SYPD performance represent a formidable challenge for exascale systems. As such simulations are of interest to scientists around the globe, we propose that this challenge be defined as goal post to be reached by the exascale systems that will be deployed next decade.

Finally, we propose a new approach to measure the efficiency of application codes on modern supercomputing systems. It is based on the observation that typical weather and climate applications, including COSMO, are predominantly memory bound. Since data movement is the expensive commodity on modern processors, we advocate that the code's performance on a given machine be characterized in terms of data movement efficiency. We use the I/O complexity lower bound as the necessary data transfers, Q, of the algorithm. With the time to solution and the maximum system bandwidth $\hat{B}$, it is possible to determine the memory usage efficiency which captures how well the code is optimized both for data locality and bandwidth utilization. We propose that data movement efficiency be considered in future performance evaluations for weather and climate codes on high performance computing systems.

Code and data availability. The particular version of the COSMO model used in this study is based on the official version 5.0 with many additions to enable GPU-capability and available under license[7]. These developments are currently in the process of being re-integrated into the mainline COSMO version. COSMO may be used for operational and for research applications by the members of the COSMO consortium. Moreover, within a license agreement, the COSMO model may be used for operational and research applications by other national (hydro-)meteorological services, universities and research institutes. The model output data will be archived for a limited amount of time and is available on request.

---

[7]consult http://www.cosmo-model.org/content/consortium/licencing.htm for more information

Appendix A: Computing the efficiency metric

Here, we discuss how to determine the memory usage efficiency (MUE) given in Equation 2 for an application code, such as the COSMO model. We need to determine the necessary data transfers Q, the maximum system bandwidth $\hat{B}$ and measure the execution time T.

A1 Necessary Transfers Q.

A natural representation of data flow and dependencies of algorithms is a Computational Directed Acyclic Graph (CDAG). This abstraction is widely used for register allocation optimization, scheduling problems and communication minimization. A CDAG models computations as vertices (V) and communications as edges (E) between them. A CDAG can be used to develop theoretical models that reason about data-movements of an application. However, not all the edges of the CDAG account for data transfers, since the data required by a computation might be stored in fast memory (*cached*), depending on the execution schedule. Finding an execution schedule that minimizes the transaction metric is NP-hard for general CDAGs, and therefore an intractable problem (Kwok and Ahmad, 1999).

Minimizing data movement has been the subject of many studies. Two main approaches have been established: (1) finding analytical lower bounds for chosen algorithms for a given machine model (Hong and Kung, 1981; Vetter, 2001; Goodrich et al., 2010) and (2) finding optimal graph partitions (Gadde, 2013). The former one is designed for particular, highly regular small algorithms, like sorting (Vetter, 2001) or matrix multiplication (Hong and Kung, 1981) and not suitable for large-scale applications like COSMO. The latter one is mostly used for minimizing network communication (Liu et al., 2014) and has not been applied to large applications either. In our performance model, we combine the two into a novel graph cutting technique. We build on Hong & Kung's 2S-partitioning (Hong and Kung, 1981) and construct a hypergraph partitioning technique to estimate a memory movement lower bound. To the best of our knowledge, we are first to apply those techniques to a real-world parallel application.

The key concept behind estimating Q is to partition the whole CDAG $G = (V, E)$ into subcomputations (2S-partitions) $P_i : \bigcup P_i = V$, such that each P_i requires at most S data transfer operations. Then, if $H(2S)$ is the minimal number of 2S-partitions for a given CDAG, Hong and Kung (1981) showed that the minimal number Q of memory movement operations for any valid execution of the CDAG is bounded by

$$Q \geq S \times (H(2S) - 1)$$

Here we outline key steps of our modeling approach:

1. We reduce Hong and Kung's *2S-Partitioning* (Hong and Kung, 1981) definition to hypergraph cut by relaxing the constraints on the dominator and minimum set sizes. Each hyperedge contains a vertex from the original CDAG and all its successors.

2. We approximate the minimal hypergraph cut by minimizing the total communication volume.

memory level	vertex def.	$|\mathbf{V}|$	$|\mathbf{E}|$	S
registers	one IJK value	157803	984101	32
shared memory	one IJ plane	2649	12137	8
L2 cache	whole array	1912	9863	29

Table A1. COSMO CDAGs at various GPU memory hierarchy levels.

3. We then express the memory movement lower bound as:

$$min \sum\nolimits_{P_i} \sum\nolimits_{v \in P_i} w(v) \cdot (Nbr(v) - 1) \tag{A1}$$

subject to

$$\bigcup P_i = V \tag{A2}$$

$$\forall_{i \neq j} P_i \cap P_j = \emptyset \tag{A3}$$

$$\forall_{P_i} \sum\nolimits_{v \in P_i} w(v) \cdot (Nbr(v) - 1) \leq 2S \tag{A4}$$

where $Nbr(v)$ is the number of partitions that vertex v is adjacent to, $w(v)$ is the memory size of vertex v and S is the size of memory at a level for which the optimization is performed. Eqn. (A1) now minimizes the sum of the communication volume across all partitions (assuming we load partition one after the other), while constraint (A4) bounds the boundary weight for each partition to $2S$ such that it fits in fast memory.

A2   COSMO CDAG

A single timestep in COSMO accesses 781 variables, each of which is represented by a 346 x 340 x 60 array for our 930 m simulation. Some variables are updated multiple times during a timestep which results in a total number of variable accesses (CDAG vertices) of more than 10^{10}. The resulting large and complex graph makes estimating Q impractical. In order to reduce the complexity one can coarsen the CDAG by grouping multiple accesses into a single vertex. As an example, Figure 2 shows the coarsest representation of the CDAG where each vertex models a full kernel. Each kernel may read and write various output variables, compute multiple stencil operations and boundary conditions, or perform halo exchanges. Thus, in this coarsened version, valuable data dependency information is lost and one cannot argue about the optimality and possible rearrangement of the operations fused within a kernel.

We now describe how we determine coarsening strategies of the COSMO CDAG for three levels of the memory hierarchy of our target system, registers, shared memory, and L2 cache:

1. **Register (65536):** The COSMO GPU implementation assigns all computations accessing variables with the same IJ coordinate to the same GPU thread and use registers to reuse values in the K direction. To model this memory hierarchy,
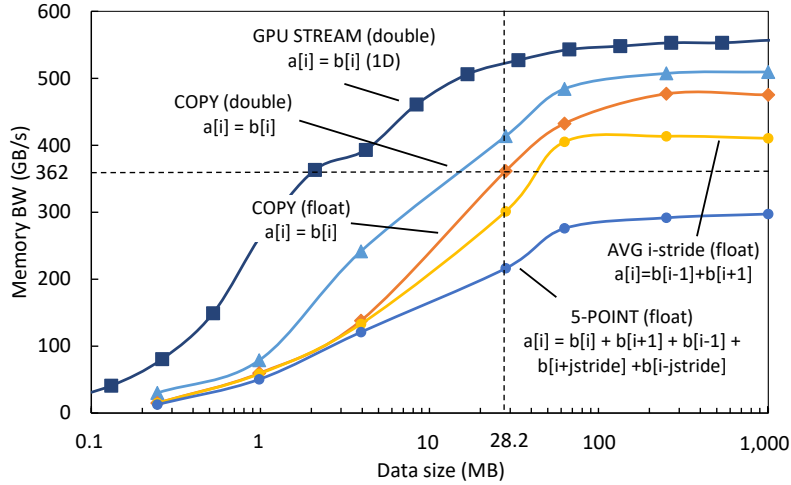
Figure A1. Bandwidth of the representative stencil benchmarks and GPU-STREAM on Tesla P100. All kernels (except of GPU STREAM) operate on a 3D domain.

it is only necessary to keep the stencil accesses in the K direction. Thus, all accesses in the IJ plane are represented as a single vertex in the CDAG, which is then simplified to 781 variables and their dependencies among all 60 levels in K.

2. **Shared memory (64 kB):** The shared memory of the GPU is used to communicate values between the different compute threads. In order to model this layer, all different accesses in the K direction of a variable are represented as a single vertex in the CDAG, while all accesses in the IJ plane are kept.

3. **L2 cache (4 MB):**. This last cache level before DRAM is used to store whole arrays (fields). In this layer all accesses to a variable in any direction are represented as a vertex in the CDAG. It keeps only the data dependencies among variables, irrespective of the offset and direction of the access.

Table **??** lists details about the CDAGs at each of our three levels. Memory capacity of the GPU for each of the three layers is then used as a constraint to derive the parameter S (see values in Table **??**). Values of the estimation of Q, obtained from the performance model for the three memory levels are shown in Table 2.

To generate the whole CDAG, we used the STELLA analysis back end (Gysi et al., 2015b) to trace all local memory accesses to all fields. Based on the information from the access offsets and order of operations, we reconstruct the read-write, write-read, and write-write dependencies.

A3   Maximum Achievable Bandwidth $\hat{B}$

We now describe how we measure the memory usage efficiency in practice. We start by describing how to determine the maximum achievable bandwidth for COSMO stencils. Even though the Tesla P100 has a theoretical peak memory bandwidth of 720 GB/s (NVIDIA, 2016), we argue that this may not be achievable for real applications. A well-established method to

measure the maximum achievable bandwidth of GPUs is the GPU STREAM benchmark (Deakin et al., 2016). Our tests show that the maximum achievable bandwidth for COPY is 557 GB/s if at least 30 MB double precision numbers are copied (Figure A1). However, stencil codes on multi-dimensional domains like COSMO require more complex memory access patterns that, even when highly tuned, cannot achieve the same bandwidth as STREAM due to architectural limitations.

5   We identified the most common patterns and designed and tuned a set of microbenchmarks that *only mimic the memory access patterns without the computation* to investigate the machine capability of handling memory accesses for stencils. They include aligned, unaligned and strided patterns in all dimensions. All benchmarks operate on a 3D domain of parametric size, on either single or double precision numbers. The results of a representative set of four chosen microbenchmarks are shown in Figure A1, together with the GPU-STREAM, which operates on 1D domain. The fastest stencil kernel (double precision

10 aligned COPY) reaches 510 GB/s. The slowdown compared to GPU-STREAM is due to the more complex access pattern in the 3D domain. Furthermore, using single precision numbers further deteriorates the bandwidth on P100 (COPY (float) reaches 475 GB/s). Our COSMO 930 m run uses pre-dominantly single precision numbers on a 346 x 340 x 60 domain, which results in 28.2 MB of data per field. Our measurements show that the maximum achievable bandwidth for this setup is 362 GB/s (in the best case of the simple COPY benchmark). We will use this upper-bound number as the maximum system bandwidth. The

15 average measured memory bandwidth across all COSMO real-world stencils is 276 GB/s which gives $\frac{B}{\hat{B}} = 0.76$.

Competing interests. TEXT

Acknowledgements. This work was supported by the Swiss National Science Foundation under Sinergia grant CRSII2_154486/1 and by a grant from the Swiss National Supercomputing Centre (CSCS). We would like to acknowledge the many contributors to the GPU-capable version of COSMO used in this study. Among others: Andrea Arteaga, Mauro Bianco, Isabelle Bey, Christophe Charpilloz, Valentin Clement, Ben Cumming, Tiziano Diamanti, Tobias Gysi, Peter Messmer, Katherine Osterried, Anne Roches, Stefan Rüdisühli, Pascal Spörri. Also, the authors would like to acknowledge the Center for Climate Systems Modeling (C2SM) and the Federal Office of Meteorology and Climatology MeteoSwiss for their support. Furthermore, we would like to thank Nils Wedi and Piotr Smolarkiewicz (ECMWF) for useful comments and discussions.

References

TOP500 Supercomputer Site, http://www.top500.org, 2017.

Alverson, B., Froese, E., Kaplan, L., and Roweth, D.: Cray XC Series Network, Tech. rep., accessed: 2017-04-03, 2012.

Balaji, V., Maisonnave, E., Zadeh, N., Lawrence, B. N., Biercamp, J., Fladrich, U., Aloisio, G., Benson, R., Caubel, A., Durachta, J., Foujols, M.-A., Lister, G., Mocavero, S., Underwood, S., and Wright, G.: CPMIP: measurements of real computational performance of Earth system models in CMIP6, Geoscientific Model Development, 10, 19–34, https://doi.org/10.5194/gmd-10-19-2017, https://www.geosci-model-dev.net/10/19/2017/, 2017.

Baldauf, M., Seifert, A., Foerstner, J., Majewski, D., Raschendorfer, M., and Reinhardt, T.: Operational Convective-Scale Numerical Weather Prediction with the COSMO Model: Description and Sensitivities, Mon. Weather Rev., 139, 3887–3905, 2011.

Ban, N., Schmidli, J., and Schär, C.: Heavy precipitation in a changing climate: Does short-term summer precipitation increase faster?, Geophysical Research Letters, 42, 1165–1172, 2014GL062588, 2015.

Benoit, R. et al.: The Real-Time Ultrafinescale Forecast Support during the Special Observing Period of the MAP, Bulletin of the American Meteorological Society, 83, 85–109, 2002.

Bony et al., S.: Clouds, circulation and climate sensitivity, Nature Geosci, 8, 261–268, 2015.

Borkar, S. and Chien, A. A.: The Future of Microprocessors, Commun. ACM, 54, 67–77, 2011.

Bott, A.: A positive definite advection scheme obtained by nonlinear renormalization of the advective fluxes, Monthly Weather Review, 117, 1006–1016, 1989.

Boucher, O., Randall, D., and et al.: Climate Change 2013: The Physical Basis, vol. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change, chap. Clouds and Aerosols, Cambridge University Press, New York, NY, USA, 2013.

Bretherton, C. S.: A National Strategy for Advancing Climate Modeling, p. 280, National Academic Press, http://www.nap.edu/, 2012.

Bretherton, C. S. and Khairoutdinov, M. F.: Convective self-aggregation feedbacks in near-global cloud-resolving simulations of an aqua-planet, Journal of Advances in Modeling Earth Systems, 7, 1765–1787, 2015.

CLM-Community: Climate Limited-area Modelling Community, http://www.clm-community.eu/, [Online; accessed 3-April-2017], 2017.

COSMO: Consortium for Small-Scale Modeling, http://www.cosmo-model.org/, [Online; accessed 4-April-2017], 2017.

Davies, T., Staniforth, A., Wood, N., and Thuburn, J.: Validity of anelastic and other equation sets as inferred from normal-mode analysis, Quarterly Journal of the Royal Meteorological Society, 129, 2761–2775, 2003.

Davies, T. et al.: A new dynamical core for the Met Office's global and regional modelling of the atmosphere, Quarterly Journal of the Royal Meteorological Society, 131, 1759–1782, 2005.

Deakin, T., Price, J., Martineau, M., and McIntosh-Smith, S.: GPU-STREAM v2.0: Benchmarking the Achievable Memory Bandwidth of Many-Core Processors Across Diverse Parallel Programming Models, pp. 489–507, Springer, Cham, 2016.

Doms, G. and Schättler, U.: The nonhydrostatic limited-area model LM (Lokal-Modell) of the DWD. Part I: Scientific documentation, Tech. rep., German Weather Service (DWD), Offenbach, Germany, http://www.cosmo-model.org/, 1999.

Durran, D. R.: Numerical Methods for Fluid Dynamics with Applications to Geophysics, vol. 32 of *Texts in Applied Mathematics*, Springer New York, 2010.

ECMWF: IFS Documentation Part III: Dynamics and numerical procedures, European Centre for Medium-Range Weather Forecasts, Shinfield Park, Reading, RG2 9AX, England, 2016.

Ewing, R. E. and Wang, H.: A summary of numerical methods for time-dependent advection-dominated partial differential equations, Journal of Computational and Applied Mathematics, 128, 423 – 445, numerical Analysis 2000. Vol. VII: Partial Differential Equations, 2001.

Eyring, V. et al.: Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, Geoscientific Model Development, 9, 1937–1958, 2016.

5 Fourestey, G. et al.: First Experiences With Validating and Using the Cray Power Management Database Tool, CoRR, abs/1408.2657, 2014.

Fu, G., Niino, H., Kimura, R., and Kato, T.: Multiple Polar Mesocyclones over the Japan Sea on 11 February 1997, Monthly Weather Review, 132, 793–814, 2004.

Fuhrer, O. et al.: Towards a performance portable, architecture agnostic implementation strategy for weather and climate models, Supercomputing frontiers and innovations, 1, http://superfri.org/superfri/article/view/17, 2014.

10 Gadde, S.: Graph partitioning algorithms for minimizing inter-node communication on a distributed system, Ph.D. thesis, The University of Toledo, 2013.

Giraldo, F. X., Kelly, J. F., and Constantinescu, E. M.: Implicit-Explicit Formulations of a Three-Dimensional Nonhydrostatic Unified Model of the Atmosphere (NUMA), SIAM Journal on Scientific Computing, 35, B1162–B1194, https://doi.org/10.1137/120876034, https://doi.org/10.1137/120876034, 2013.

15 Goodrich, M. T., Sitchinava, N., and Arge, L.: Parallel external memory graph algorithms, 2010 IEEE International Symposium on Parallel and Distributed Processing (IPDPS), 00, 1–11, 2010.

Gysi, T., Grosser, T., and Hoefler, T.: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, in: Proceedings of the 29th ACM on International Conference on Supercomputing, ICS '15, pp. 177–186, ACM, New York, NY, USA, 2015a.

20 Gysi, T., Osuna, C., Fuhrer, O., Bianco, M., and Schulthess, T. C.: STELLA: A Domain-specific Tool for Structured Grid Methods in Weather and Climate Models, in: Proc. of the Intl. Conf. for High Performance Computing, Networking, Storage and Analysis, SC '15, pp. 41:1–41:12, ACM, New York, NY, USA, 2015b.

Heise, E., Ritter, B., and Schrodin, R.: Operational implementation of the multilayer soil model, COSMO Tech. Rep., No. 9, Tech. rep., COSMO, 2006.

25 Hong, J.-W. and Kung, H. T.: I/O Complexity: The Red-blue Pebble Game, in: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, STOC '81, pp. 326–333, ACM, New York, NY, USA, 1981.

Hope, C.: The $10 trillion value of better information about the transient climate response, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 373, 2015.

Hutter, J., Iannuzzi, M., Schiffmann, F., and VandeVondele, J.: CP2K: atomistic simulations of condensed matter systems, Wiley Interdisci-
30 plinary Reviews: Computational Molecular Science, 4, 15–25, 2014.

Jablonowski, C. and Williamson, D. L.: A baroclinic instability test case for atmospheric model dynamical cores, Quarterly Journal of the Royal Meteorological Society, 132, 2943–2975, 2006.

Karypis, G. and Kumar, V.: MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, http://www.cs.umn.edu/~metis, 2009.

35 Kendon, E. J. et al.: Heavier summer downpours with climate change revealed by weather forecast resolution model, Nature Clim. Change, 4, 570–576, 2014.

Kwok, Y.-K. and Ahmad, I.: Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors, ACM Comput. Surv., 31, 406–471, 1999.

Lapillonne, X. and Fuhrer, O.: Using Compiler Directives to Port Large Scientific Applications to GPUs: An Example from Atmospheric Science, Parallel Processing Letters, 24, 1450 003, 2014.

Lapillonne, X. et al.: Operational numerical weather prediction on a GPU-accelerated cluster supercomputer, in: EGU General Assembly Conference Abstracts, vol. 18 of *EGU General Assembly Conference Abstracts*, p. 13554, 2016.

5   Lee et al., V. W.: Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU, SIGARCH Comput. Archit. News, 38, 451–460, 2010.

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, Geoscientific Model Development, 9, 3393–3412, 2016.

Leutwyler, D., Lüthi, D., Ban, N., Fuhrer, O., and Schär, C.: Evaluation of the convection-resolving climate modeling approach on continen-

10   tal scales, Journal of Geophysical Research: Atmospheres, 122, 5237–5258, https://doi.org/10.1002/2016JD026013, http://dx.doi.org/10.1002/2016JD026013, 2017.

Liu, L., Zhang, T., and Zhang, J.: DAG Based Multipath Routing Algorithm for Load Balancing in Machine-to-Machine Networks, International Journal of Distributed Sensor Networks, 10, 457 962, 2014.

Ludwig, P., Pinto, J. G., Hoepp, S. A., Fink, A. H., and Gray, S. L.: Secondary Cyclogenesis along an Occluded Front Leading to Damaging

15   Wind Gusts: Windstorm Kyrill, January 2007, Monthly Weather Review, 143, 1417–1437, 2015.

Michalakes et al., J.: AVEC Report: NGGPS Level-1 Benchmarks and Software Evaluation, Tech. rep., NGGPS Dynamical Core Test Group, 2015.

Miura, H., Satoh, M., Nasuno, T., Noda, A. T., and Oouchi, K.: A Madden-Julian Oscillation Event Realistically Simulated by a Global Cloud-Resolving Model, Science, 318, 1763–1765, 2007.

20   Miyamoto, Y. et al.: Deep moist atmospheric convection in a subkilometer global simulation, Geophysical Research Letters, 40, 4922–4926, 2013.

NVIDIA: NVIDIA TESLA P100 Technical Overview, Tech. rep., accessed: 2017-04-03, 2016.

Pachauri, R. K. and Meyer, L. A., eds.: Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovtl Panel on Climate Change, p. 151, IPCC, Geneva, Switzerland, 2014.

25   Palmer, T.: Climate forecasting: Build high-resolution global climate models, Nature, 2014.

Park, S.-H., Skamarock, W. C., Klemp, J. B., Fowler, L. D., and Duda, M. G.: Evaluation of Global Atmospheric Solvers Using Extensions of the Jablonowski and Williamson Baroclinic Wave Test Case, Monthly Weather Review, 141, 3116–3129, 2013.

Ralph, F. M.: Observations of 250-km-Wavelength Clear-Air Eddies and 750-km-Wavelength Mesocyclones Associated with a Synoptic-Scale Midlatitude Cyclone, Monthly Weather Review, 124, 1199–1210, 1996.

30   Raschendorfer, M.: The new turbulence parameterization of LM, 2001.

Reinhardt, T. and Seifert, A.: A three-category ice scheme for LMK, 2006.

Ricard, D., Lac, C., Riette, S., Legrand, R., and Mary, A.: Kinetic energy spectra characteristics of two convection-permitting limited-area models AROME and Meso-NH, Quarterly Journal of the Royal Meteorological Society, 139, 1327–1341, https://doi.org/10.1002/qj.2025, http://dx.doi.org/10.1002/qj.2025, 2013.

35   Richard, E., Buzzi, A., and Zängl, G.: Quantitative precipitation forecasting in the Alps: The advances achieved by the Mesoscale Alpine Programme, Q. J. Roy. Meteor. Soc., 133, 831–846, https://doi.org/10.1002/qj.65, 2007.

Ritter, B. and Geleyn, J.-F.: A comprehensive radiation scheme for numerical weather prediction models with potential applications in climate simulations, Monthly Weather Review, 120, 303–325, 1992.

Schneider, T. et al.: Climate goals and computing the future of clouds, Nature Clim. Change, 7, 3–5, 2017.

Shalf, J., Dosanjh, S., and Morrison, J.: Exascale Computing Technology Challenges, pp. 1–25, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-19328-6_1, https://doi.org/10.1007/978-3-642-19328-6_1, 2011.

Skamarock, W. C., Park, S.-H., Klemp, J. B., and Snyder, C.: Atmospheric Kinetic Energy Spectra from Global High-Resolution Nonhydrostatic Simulations, Journal of the Atmospheric Sciences, 71, 4369–4381, 2014.

Steppeler, J., Doms, G., Schättler, U., Bitzer, H., Gassmann, A., Damrath, U., and Gregoric, G.: Meso gamma scale forecasts using the nonhydrostatic model LM, Meteor. Atmos. Phys., 82, 2002.

Ullrich, P. A., Reed, K. A., and Jablonowski, C.: Analytical initial conditions and an analysis of baroclinic instability waves in f- and, β-plane 3D channel models, Quarterly Journal of the Royal Meteorological Society, 141, 2972–2988, 2015.

Vetter, J. S.: External Memory Algorithms and Data Structures: Dealing with Massive Data, ACM Comput. Surv., 33, 209–271, 2001.

Wicker, L. J. and Skamarock, W. C.: Time-Splitting Methods for Elastic Models Using Forward Time Schemes, Monthly Weather Review, 130, 2088–2097, 2002.

Yang et al., C.: 10M-core Scalable Fully-implicit Solver for Nonhydrostatic Atmospheric Dynamics, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC16, pp. 6:1–6:12, IEEE Press, Piscataway, NJ, USA, 2016.

Yashiro, H. et al.: Performance Analysis and Optimization of Nonhydrostatic ICosahedral Atmospheric Model (NICAM) on the K Computer and TSUBAME2.5, in: Proceedings of the Platform for Advanced Scientific Computing Conference on ZZZ - PASC '16, ACM Press, 2016.