Geosci. Model Dev., 9, 3483–3491, 2016
www.geosci-model-dev.net/9/3483/2016/
doi:10.5194/gmd-9-3483-2016

Geoscientific Model Development
Open Access
EGU

Earth system modelling on system-level heterogeneous architectures: EMAC (version 2.42) on the Dynamical Exascale Entry Platform (DEEP)

Michalis Christou[1], **Theodoros Christoudias**[1], **Julián Morillo**[2], **Damian Alvarez**[3], **and Hendrik Merx**[1,4]

[1]The Cyprus Institute, Nicosia, Cyprus
[2]Barcelona Supercomputing Center, Barcelona, Spain
[3]Jülich Supercomputing Centre, Jülich, Germany
[4]Max Planck Institute for Chemistry, Mainz, Germany

Correspondence to: Theodoros Christoudias (christoudias@cyi.ac.cy)

Received: 30 November 2015 – Published in Geosci. Model Dev. Discuss.: 28 January 2016
Revised: 30 May 2016 – Accepted: 15 September 2016 – Published: 29 September 2016

Abstract. We examine an alternative approach to heterogeneous cluster-computing in the many-core era for Earth system models, using the European Centre for Medium-Range Weather Forecasts Hamburg (ECHAM)/Modular Earth Submodel System (MESSy) Atmospheric Chemistry (EMAC) model as a pilot application on the Dynamical Exascale Entry Platform (DEEP). A set of autonomous coprocessors interconnected together, called Booster, complements a conventional HPC Cluster and increases its computing performance, offering extra flexibility to expose multiple levels of parallelism and achieve better scalability. The EMAC model atmospheric chemistry code (Module Efficiently Calculating the Chemistry of the Atmosphere (MECCA)) was taskified with an offload mechanism implemented using OmpSs directives. The model was ported to the MareNostrum 3 supercomputer to allow testing with Intel Xeon Phi accelerators on a production-size machine. The changes proposed in this paper are expected to contribute to the eventual adoption of Cluster–Booster division and Many Integrated Core (MIC) accelerated architectures in presently available implementations of Earth system models, towards exploiting the potential of a fully Exascale-capable platform.

1 Introduction

The ECHAM/MESSy Atmospheric Chemistry (EMAC) model is a numerical chemistry and climate simulation system that includes sub-models describing tropospheric and middle atmosphere processes and their interaction with oceans, land, and human influences (Jöckel et al., 2010). It uses the second version of the Modular Earth Submodel System (MESSy2) to link multi-institutional computer codes. The core atmospheric model is the 5th generation European Centre for Medium Range Weather Forecasts Hamburg general circulation model (ECHAM5, Roeckner et al., 2003, 2006).

The EMAC model runs on several platforms, but it is currently unsuitable for massively parallel computers, due to its scalability limitations and large memory requirements per core. EMAC employs complex Earth-system simulations, coupling a global circulation model (GCM) with local physical and chemical models. The global dynamical processes are strongly coupled and have high communication demands, while the local physical processes are inherently independent with high computation demands. This heterogeneity between different parts of the EMAC model poses a major challenge when running on homogeneous parallel supercomputers.

We test a new approach for a novel supercomputing architecture as proposed by the DEEP project (Eicker et al., 2013, 2015; Mallon et al., 2012, 2013; Suarez et al., 2011), an innovative European response to the Exascale challenge. Instead of adding accelerator cards to Cluster nodes, the DEEP

project proposes to use a set of interconnected coprocessors working autonomously (called Booster), which complements a standard Cluster. Together with a software stack focused on meeting Exascale requirements – comprising adapted programming models, libraries, and performance tools – the DEEP architecture enables unprecedented scalability. The system-level heterogeneity of DEEP, as opposed to the common node-level heterogeneity, allows users to run applications with kernels of high scalability alongside kernels of low scalability concurrently on different sides of the system, avoiding at the same time over- and under-subscription.

The Cluster–Booster architecture is naturally suited to global atmospheric circulation–chemistry models, with global components running on the Cluster nodes exploiting the high-speed Xeon processors and local components running on the highly parallel Xeon Phi coprocessors. By balancing communication vs. computation, the DEEP concept provides a new degree of freedom, allowing us to distribute the different components at their optimal parallelization.

2 Overview of application structure

The EMAC model comprises two parts, the dynamical base model ECHAM, using a non-local, spectral algorithm with low scalability, and the modular framework MESSy, linking local physical and chemical processes to the base model, with high scalability. While the number of processors used for the base model is limited by the non-local spectral representation of global dynamical processes, local physical and chemical processes described by framework submodels run independently of their neighbours and present very high scalability.

2.1 Phases

The ECHAM base model runs in parallel in the distributed-memory paradigm using the Message Passing Interface (MPI, Aoyama and Nakano, 1999) library for communication; the MESSy framework inherits the parallel decomposition defined by the base model. While ECHAM has been shown to be able to exploit the shared-memory paradigm using the Open Multi-Processing (OpenMP, 2015) library (Dagum and Menon, 1998), no such effort had been undertaken for the MESSy model so far.

It is, however, currently not possible to delegate the whole MESSy subsystem to full multi-threaded execution as some physical processes are naturally modelled in a column-based approach, and are strongly dependent on the system states at their vertically adjacent grid points. The implementation of submodels simulating these processes consequently relies on the column structure inherited from the base model, e.g. sunlight intensity at lower grid points depending on the absorption at higher grid points, and precipitation depending on the flux of moisture from vertically adjacent grid cells.

Figure 1. Phases of EMAC. Green phases run on the Cluster; blue phases run on the Booster.

Describing gas-phase chemical kinetics, the MESSy MECCA submodel (Sander et al., 2011) executes independently of its physical neighbours and is not limited by vertical adjacency relations. As more than half of the total run time is spent in MECCA for a typical model scenario, it seems adequate to concentrate on the MECCA kernel with strong algorithmic locality and small communication volume per task. As sketched in Fig. 1 the current implementation of MECCA, developed in the DEEP project, is delegated to the Booster using a task-based approach, while both ECHAM and the remaining MESSy submodels are executed on the Cluster in the distributed-memory paradigm.

2.2 Scalability dominant factors

Implementing a spectral model of the dynamical state of the atmosphere, the ECHAM phase comprises six transform and six transposition operations in each time step. The data in memory for each time step (data size scales with the square of the model horizontal resolution) are transposed in an all-to-all communication pattern, and this phase is dominated by network bandwidth.

Figure 2 displays one model cycle traced with Extrae/Paraver (Extrae, 2015; Paraver, 2015) starting with the end of the grid-point calculations of the last time step calculated – in which most processors are already idle (orange) due to load imbalance and waiting for process 14 (blue) to finish running. This is followed by the transpositions in the beginning of a new time step, and Fourier and Legendre transformations (magenta), which execute simultaneously, as further analysis showed. After the transpositions a short interval with all processors running (blue) can be identified with the time-step integration in spectral space, followed by the inverse transformations and transpositions and transport calculations in ECHAM.

While the pattern described so far repeats towards the end of the displayed interval, the major fraction of the time step is spent without communication, running (blue) or waiting (orange) in calculations in MESSy in grid space. The MESSy phase comprises some 30 submodels that are tightly coupled by exchanging the atmospheric observables using global variables. Investigations during the first phase of the project determined the load imbalance visible in Fig. 2 to be caused by chemical processes computed in the MECCA submodel.

The observed load imbalance is one of the main factors determining application scalability. It is caused by an adaptive time-step integrator solving a system of differential equations. As the stiffness of these equations representing pho-

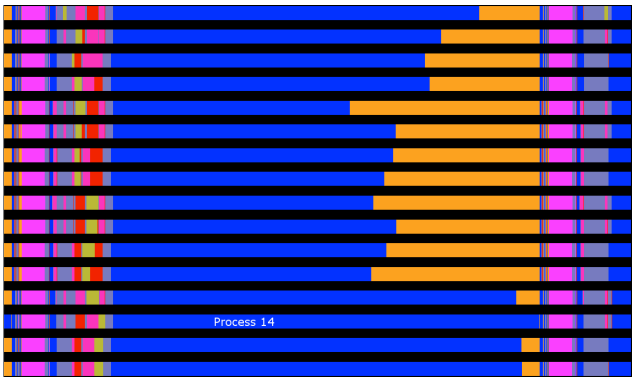

Figure 2. Paraver trace of major processor usage of one time step. Time is along the horizontal and each bar corresponds to a separate CPU core. Blue colour depicts computation; orange corresponds to idle time due to load imbalance. The grid-space transpositions and Fourier and Legendre transformations are shown in magenta.

Table 1. Experimental set-up for JUDGE scalability test out-of-the-box.

Number of columns	8192 columns with 90 levels
Number of grid points	737 280 grid points
Number of chemical species	139 species in 318 reactions
Spectral resolution	T42L90MA

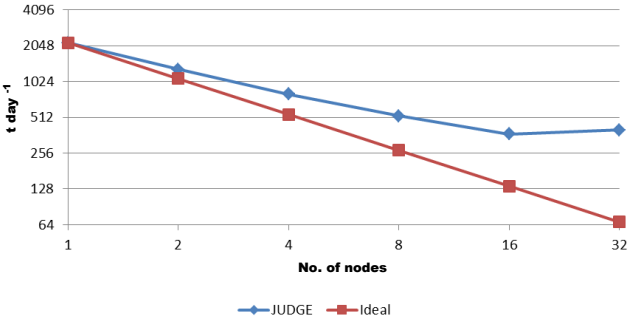

Figure 3. Wall time for 1 simulated day vs. the number of nodes on JUDGE.

tochemical reactions varies by up to 2 orders of magnitude due to changes in the intensity of sunlight, the adaptive integrator demands varying amounts of run time accordingly.

In the MECCA phase the algorithmically complex adaptive time-step differential equation integrator operates on variables representing chemical concentrations of a total data size of the order of a few kilobytes per grid point. Yet, as observed using Scalasca (Scalasca, 2015), this phase consumes the major proportion of the total execution time (76 %), and it is compute-bound and an obvious candidate for offloading to accelerators. It should be noted though that in a conventional architecture, accelerating this highly parallel phase will not eliminate the load imbalance.

To test the model scalability out-of-the-box, the EMAC application has been ported to the JUDGE cluster at the Jülich Supercomputing Centre (JSC), and a representative benchmark with a horizontal resolution of 128 grid points in the longitudinal direction and 64 grid points in the latitudinal direction with 90 vertical levels and a spin-up period of 8 simulated months has been compiled, frozen, and packaged to be used for measurements. Table 1 details the experimental set-up for the results shown in this section. The default E5M2 KPP chemistry batch option, along with model namelist set-up (NML_SETUP) E5M2/02b_qctm, without dynamics nudging and with the diagnostic submodels D14CO, DRADON, S4D, and VISO switched off, is used throughout this work. All disk output channels were turned off for timing purposes.

The EMAC strong scaling was benchmarked with different numbers of processors on JUDGE in order to determine the run-time behaviour of the total application. As shown in Fig. 3, the application scales up to 384 processes (16 nodes $\times$ 24 MPI processes each); at higher numbers, the performance decreases. Parallel execution speed is determined by the balance of three factors: computation, communication, and load imbalance. The benchmarking set-up for the JUDGE cluster can be seen in Table 2. While the computational resources increase with additional processors and therefore increase the application performance, communication demands diminish the positive effect of the additional processors. Additionally, increasing the granularity of the total workload also increases the load imbalance.

While the number of processors used for the distributed-memory part of the code is limited by the scalability of the non-local representation of global dynamical processes in ECHAM, the local processes in MESSy running independently of their neighbours scale very well. The MESSy subsystem has not been designed for multi-threaded execution, though, and contains non-local code due to characteristics of the physical processes and algorithmic design decisions. Some physical processes are naturally modelled in a column-based approach, because they are strongly dependent on the system states at vertically adjacent grid points, e.g. sunlight intensity at lower grid points depending on the absorption at higher grid points, and precipitation depending on the flux of moisture from vertically adjacent grid cells. Sub-models simulating these processes consequently rely on the column structure implemented in the current model.

In the existing distributed-memory parallel decomposition, the three-dimensional model grid is split horizontally using two run-time parameters, setting the number of processes in the latitudinal and longitudinal directions. As work is distributed independently for each direction, a rectangular decomposition is obtained. In ECHAM5, two such rectangular sets of grid points, symmetric with respect to the Equator and balancing the load distribution between the Earth's hemi-

Table 2. System set-up details for the analysis done on the JUDGE system.

Backend compiler version	Intel 13.1.3
MPI run-time version	Parastation/Intel MPI 5.0.27
Compilation flags	-O3 –fp-model source –r8 –align all
MPI processes per node	24

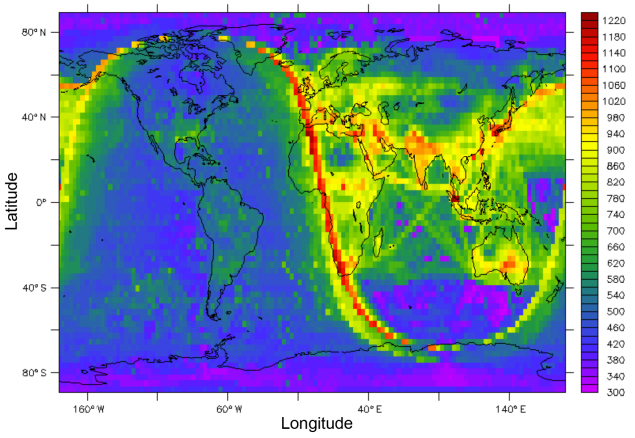

Figure 4. Column-maximum MECCA kernel execution wall time in microseconds for a single time step. The adaptive time-step integrator shows a non-uniform run time caused by stratospheric photochemistry and natural and anthropogenic emissions.

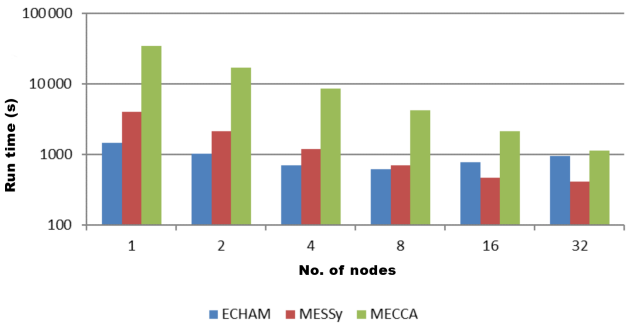

Figure 5. Run time of each phase of EMAC, when running on MareNostrum 3.

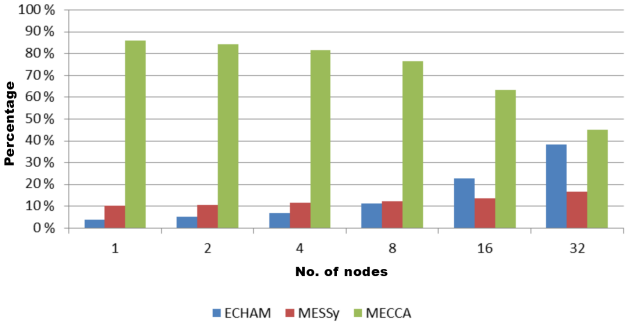

Figure 6. Percentage of run time of each phase of EMAC, when running on MareNostrum 3.

spheres, are assigned to one processor. Further, the first transposition in the dynamical core reorders the variables assigned to a processor to rectangular stripes extending along the full latitudes. Assigning the two run-time parameters mentioned above so that the rectangular sets are elongated in the latitudinal direction reduces the inter-processor communication in the first transposition.

The physical load imbalance, caused by photo-chemical processes in the lower stratosphere and natural and anthropogenic emissions (lightning, soil bacteria, fuel combustion–motor vehicles, industrial, and utility), appears in the run time spent for each grid point when examining the benchmark calculations. In Fig. 4 the maximal MECCA kernel execution wall time for one grid point in each column differs by up to a factor of 4. The load imbalance is caused by the adaptive time-step integrator solving the differential equations that describe the chemical equations computed in the MECCA submodel.

At high levels of parallelization, the load imbalance becomes a limiting factor, and the factors determining scalability in absolute numbers in Fig. 5 are both communication and computation. For the ECHAM phase (blue), when scaling to beyond eight nodes, the communication demands of the underlying spectral model involving several all-to-all communication patterns start to dominate. Computing time for MESSy is still decreasing and the computing time for MECCA decreases more strongly than MESSy (note the logarithmic scale and the distinction of MECCA from MESSy). For reference, the MareNostrum 3 heterogeneous compute nodes each have 2×8 CPU cores (MPI) and $2\times$ Xeon Phi accelerators.

In Fig. 6 the point at which communication and computation require equal times around 8 nodes is clearly apparent; this ultimately limits the maximum number of cores that can be used with high efficiency and 16 nodes (equivalent to 256 cores) are commonly used in production runs of the EMAC atmospheric model as a scientific application to balance efficiency and total required wall time.

3 Model developments

3.1 Intranode taskification

The EMAC model atmospheric chemistry code (MECCA) was taskified using OmpSs (Bueno et al., 2011, 2012; Duran et al., 2011; Sainz et al., 2014) directives. OmpSs allows the user to specify inputs and outputs for blocks of code or functions, giving enough information to the Nanos++ run-time system to construct a dependency graph. This dependency graph reflects at all moments which tasks are ready to be executed concurrently, and therefore the programmer does

not have to explicitly manage the parallelization. The concept of tasks and task dependencies has also been adopted in the OpenMP 4.0 standard (OPENMP 4.0, 2013). Since in MECCA each grid point is computed independently of its neighbours, this part of the code is in principle embarrassingly parallel, with no communication or inter-task dependencies involved. The MECCA submodel was refactored through the creation of computational kernels for intranode parallelization with shared-memory tasks. All changes are in the automatically generated MESSy MECCA KPP source code and no additional changes are needed in the MESSy submodel interface layer (SMIL) or in the core layer (SMCL). Since the source is automatically generated, the changes have to be propagated to the generator after the KP4 mechanism runs (applied in order to remove the indirect indexing and expand the original KPP code by an additional dimension to enable a better performance due to better cache usage). The propagation is still a work in progress and falls outside the scope of this paper.

The new integration subroutine performs identical functions to the kpp_integrate subroutine but on unique elements:

```
!$OMP TARGET DEVICE(SMP) COPY_IN(k,
temp, press, cair, khet_st, khet_tr,
jx, time_step_len, atol, rtol, icntrl,
rcntrl) COPY_INOUT(conc)
SUBROUTINE kpp_integrate_kernel (k,
conc, temp, press, cair, khet_st,
khet_tr, jx, dt, atol, rtol, icntrl,
rcntrl)
```

The statements COPY_INOUT and COPY_IN refer to the OmpSs specification and declare memory dependence and declare task dependencies. These dependencies are used by the Nanos++ run-time system to construct a dependency graph which ensures that the tasks are executed concurrently when their dependencies are met and there are free threads. The OmpSs pragma targets symmetric multiprocessing (SMP), symmetric multiprocessor system hardware and software architecture where each identical processor unit connects to a single, shared main memory and has full access to all I/O devices.

The new version of EMAC, running ECHAM with MPI processes and MECCA with shared-memory OmpSs tasks, outperforms the old EMAC using pure MPI, and continues to scale beyond the region where the original implementation scaling performance plateaus. This can be seen in Fig. 7, which shows the performance using multi-threading on the DEEP Cluster.

3.2 Internode taskification

In DEEP, OmpSs has been extended to support offloading tasks to remote nodes (Beltran et al., 2015). This mimics the behaviour of other accelerator APIs that move data from the host to the device, compute in the device, and return the results to the host. However, OmpSs adds two very important

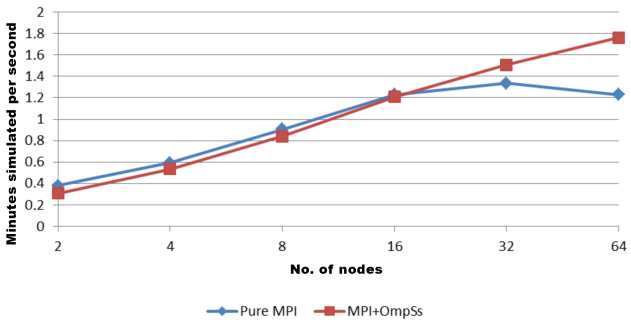

Figure 7. Performance of OmpSs threading in the DEEP Cluster.

features: (i) it allows offloading to remote nodes, not just locally available coprocessors/accelerators, which is a key functionality to effectively use the Booster; and (ii) it allows use of the Booster as a pool of coprocessors, so tasks can be offloaded to any Booster node with enough free cores. The latter enables one to eliminate the load imbalance caused by sunlight gradients in MECCA. The second source of imbalance by heterogeneous reactions is also automatically alleviated by the dynamical load balance using the massive parallelization in the Booster. Our proposed solution is agnostic to the specific origin of load imbalance in the chemistry calculation.

In a shared-memory taskification the data are already shared between threads, and no memory copies are necessary. However, in DEEP, to leverage the Booster, these data have to be copied to the Booster nodes. Keeping that in mind, the new task-based MECCA implementation was optimized and the memory and network footprint of the distributed-memory offloading were reduced by 3 orders of magnitude. To minimize the memory footprint for offloaded tasks, the number of computational grid elements issued to MESSy is further split into individual elements for each task, by rearranging the grid-point arrays in each time step to implement data locality at the grid-point level, resulting in a reduction of the total memory footprint from 2.7 MB down to 6.3 KB for each task. This was the result of refactoring both the data and code structures in MECCA.

In order to create multiple threads integrating the chemical equations simultaneously, all data describing the state of a single grid point have been identified and separated into local variables. Using these variables an integrator kernel has been created that can be offloaded onto threads running on the main processor or hardware accelerators. The integrator kernel needs the variables describing the local state of the atmosphere and the integrator parameters defining the solution of the chemical equations. As the chemical mechanism is compiled by the Kinetic PreProcessor (KPP) (Damian et al., 2002), some module variables contain the data of a set of grid points that is accessed using the index K inside the integrator. In order to maintain the code for the integration of different chemical mechanisms created by the KPP compiler,

these input variables are passed in their original size together with the column grid-point index K:

```
!OMP TASK FIRSTPRIVATE (k) INOUT (conc
(k, :) ) IN (temp, press, cair, khet_st,
khet_tr, jx, time_ step_len, atol, rtol,
icntrl, rcntrl) ONTO ()
```

The keywords INOUT and IN refer to the OmpSs specification and declare task dependencies. ONTO sets the offload target – left blank it targets any available accelerator device. The input arrays have been reduced in the dimension of NBL (see below), and now carry the memory of a single grid point. The only two-dimensional array left is conc, and only one slice (k,:) is passed to and from containing the concentrations of all chemical species. Prior to passing each element all two-dimensional arrays above are transposed in the outermost dimension, to have sequential memory regions, as in the Fortran language specification. The compiler directives are implemented as pragmas (comments in the source code), automatically controlled with definitions at compile time (being invoked only if SMP/Mercurium is present); thus, no additional user input is required. The kernel data sizes depend on parameters in the EMAC environment NBL = KPROMA × NLEV. The block length NBL corresponds to the vector length used in the ECHAM base model for grid-point calculations. A block in ECHAM comprises all grid points in a slice of the model grid extending along a virtual longitude of length KPROMA and all model levels NLEV. KPROMA is a run-time parameter and NLEV corresponds to the model vertical resolution. The vector length in the integrator kernel is defined by the user based on the capabilities of the hardware at compile time. The total size of data available to a single shared-memory task depends on the KPP control parameters and number of species (NSPEC). As an example, using the benchmark parameters for the chemical mechanism, each task has access to 2.7 Mbytes of shared main memory.

When offloading tasks, to maximize massive parallelization, the parameter NBL is reduced at run time to NBL = 1, by redefining the grid-point arrays, resulting in a total memory footprint of just 3588 bytes for each task. At the benchmark resolution of T42L90MA a total number of 737 280 independent tasks is generated in each time step, resulting in a total data size of 2.5 Gbytes. Creating the tasks requires all this memory to be transferred from the Cluster to the Booster.

At a typical parallelization of 256 processes for the less-scaling spectral parts of the model, each MPI process creates NBL = 2880 tasks and transfers 9.9 Mbytes per process in each simulation time step. The data returned from the OmpSs tasks are smaller in size as they only comprise the concentrations of each chemical species in the CONC array. For the benchmark chemical mechanism with NSPEC = 139 this amounts to 1112 bytes per task. The corresponding total amount of data per time step is 782 Mbytes, with each MPI process receiving 3.0 Mbytes.

Additionally, the distributed-memory offloading code was redesigned to exploit shared memory within the Xeon Phi many-core processors by nesting an OmpSs shared-memory region within Cluster-to-Booster tasks encompassing a variable, run-time-defined number of individual grid-point calculations. The run-time parameter can be tweaked to change the total memory required by the number of grid points per task to fit within the device shared memory, to minimize run-time overhead (avoiding many tasks spawned over the network, which has a huge penalty). Thus, the number of tasks to be sent to the Booster can be controlled and optimized for each architecture, and host-specific configuration allows for optimum task size based on bandwidth, reducing task communication overheads.

With this approach, the specifics of the DEEP system architecture, and in particular the hardware present in MIC coprocessors, are exploited by massively parallelizing the chemistry calculations at the grid-point level and offloading to the Booster, exposing a significant amount of thread parallelism. At the same time the load imbalance observed in MECCA is automatically alleviated through OmpSs' dynamic load balancing by selecting a sufficiently fine task size and decoupling the model-domain location of the grid point from the task execution on the physical CPU.

3.3 Kernel refactoring

The computational kernel of MECCA conceptually accesses several single elements of the global arrays describing the state of the atmosphere. In order to create data locality, in a first step the module variables used in the current version of MECCA to represent the model state were refactored within the mecca_physc function to be passed explicitly to the kpp_integrator routine that computes the chemical equations of one vector of grid points. Subsequently, to increase memory adjacency, some fields were transposed from the memory order optimized for vectorization along longitudes as given by the ECHAM core model to arrays with adjacent elements representing chemical concentrations and reaction rate constants. In a third step, the computation of the equations of a single grid point were encapsulated in a new kpp_integrator_kernel function that calls the functions update_rconst and integrate, which are created automatically by the KPP compiler. This innermost routine is called in a loop over all grid points in a vector along the virtual longitude defined by the ECHAM infrastructure:

```
DO k = 1, vl_glo
CALL kpp_integrate_kernel (conc (:,
k), temp (k), press (k), cair (k),
khet_st (:, k), khet_tr (:, k), jx (:,
k), time_step_len, atol, rtol, icntrl,
rcntrl, wtime (k))
END DO
```

By maintaining the interfaces both to the MESSy infrastructure and the code implementing the KPP integrator, the

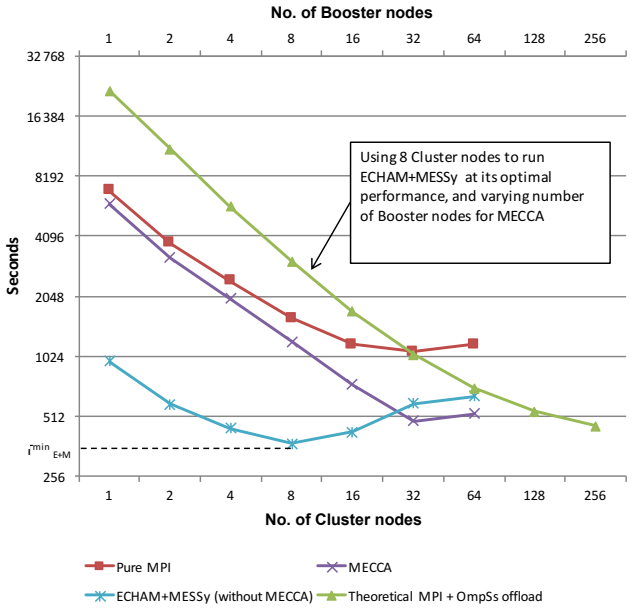

Figure 8. Time per simulated day in DEEP using a pure MPI approach, and a theoretical performance with offloading to Xeon Phi, based on the metrics collected on MareNostrum 3. The theoretical MPI + OmpSs offload data are based on a fixed configuration on the Cluster using eight nodes and scaling the number of Booster nodes.

changes to the EMAC code base were kept minimal, and a transfer of the changes to other MESSy submodels using the KPP integrator should be feasible.

4 Attainable performance

At the time of writing this paper, the DEEP Booster is in the bring-up phase, and is not available to users. In order to project the performance of the full DEEP system, Xeon-based measurements on the DEEP Cluster were combined with Xeon Phi-based measurements on MareNostrum 3. The "Pure MPI" nominal time on the DEEP Cluster for all phases (red line), the time for each phase, and the theoretical performance when offloading to the Booster are shown in Fig. 8. The DEEP Cluster reference data weighted by the relative factors for each phase derived from the metrics measurements exhibit a performance maximum for the base model (ECHAM) and MESSy (excluding MECCA) at eight nodes (light blue line), representing a good estimate for the optimal parallelization of that phase on the Cluster. This estimate of 375 s per simulated day for the low-scaling Cluster phases was used to extrapolate the attainable performance (green line), merging this result at eight nodes, with the Xeon Phi data retrieved from MareNostrum 3, where benchmarks using one node had been run with varying numbers of processing elements within one Xeon Phi processor.

While the number of Booster nodes required to attain similar performance to the original distributed-memory based implementation corresponds to regular accelerator architectures with individual boosters directly attached to cluster nodes, the projected DEEP performance (green line) scales beyond the optimal performance achieved so far. The EMAC atmospheric chemistry global climate model seems therefore well suited to exploit an architecture providing considerably more hardware acceleration than provided by regular systems. To find the attainable performance time $T_{\text{attainanle}}$ when running the ECHAM + MESSy on the Cluster and MECCA on the Booster, varying the number of Booster nodes, the following equation is used:

$$T_{\text{attainable}} = T^{\text{min}}_{\text{ECHAM+MESSy}} + T^{\text{Booster}}_{\text{MECCA}}.$$

The projected attainable performance that outperforms the pure-MPI conventional cluster paradigm at higher core count (depicted here as the number of Booster nodes, while keeping the ECHAM/MESSy MPI part on eight Cluster nodes for optimal performance $T^{\text{min}}_{\text{ECHAM+MESSy}}$) is also shown in Fig. 8.

5 Conclusions

The ECHAM/MESSy Atmospheric Chemistry (EMAC) global climate model is used to study climate change and air quality scenarios. The EMAC model is constituted by a non-local dynamical part (ECHAM) with low scalability, and local physical (MESSy) and chemical (MECCA) processes with high scalability. The model's structure naturally suits the DEEP architecture using the Cluster nodes for the non-local part and the Booster nodes for the local chemistry processes. Different implementations of the code's memory and workload divisions were developed and benchmarked to test different aspects of the achievable performance on the proposed architecture. The use of the OmpSs API largely frees the programmers from implementing the offloading logic and, given that EMAC is developed and used in a large community working on all aspects of the model, can facilitate adoption of the concept in the MESSy community.

The chemistry mechanism was taskified at the individual grid-point level using OmpSs directives. The chemistry code was refactored to allow for memory adjacency of vector elements. The OmpSs taskification with remote offload allows for massive task parallelization and the implementation of optional two-stage offload to control Cluster–Booster task memory size and optimum bandwidth utilization.

The computational load imbalance arising from a photochemical imbalance is alleviated at moderate parallelization by assigning grid points with differing run times to each process and distributing the load over all processes. Due to the physical distribution of sunlight this load balancing does not require an explicit algorithm at moderate parallelization; instead, the implicit assignment of the model grid in rectan-

gular blocks suffices for this purpose. At higher numbers of processors this implicit load-balancing decreases and the resulting load imbalance has to be solved by active balancing. The dynamic scheduling provided by the OmpSs run-time system balances the computational load without a possible, but expensive, prediction for the current time step.

With these approaches, the specifics of the DEEP system architecture, and in particular the hardware present in MIC coprocessors, can be exploited by massively parallelizing the chemistry calculations at the grid-point level and offloading to the Booster, exposing a significant amount of thread parallelism. At the same time the load imbalance observed in MECCA will be automatically alleviated through dynamic load balancing by minimizing the individual task size to one grid box and decoupling the model-domain location from the task execution on the physical CPU, and transferring it to any available core on the Booster.

Benchmark projections based on available hardware running the DEEP software stack suggest that the EMAC model requires the large numbers of Xeon Phi accelerators available in the DEEP architecture to scale beyond the current optimal performance point and exploit Amdahl's law with the highly scalable grid-point calculations while capitalizing on the high performance and fast communication for the spectral base model on Intel Xeon processors.

In the longer term, when the optimal speedup of the chemical kinetics computation is approached, the overall parallel speedup of EMAC will be limited by the remaining non-accelerated submodels. To achieve further speedup, additional computationally expensive submodels like the aerosol submode GMXe could be addressed. GMXe is column-bound and calls KPP up to four times during one time step (for grid-/subgrid-scale liquid/ice clouds), with load imbalance caused by the distribution of clouds over the model domain. In principle, the MECCA implementation is directly applicable to the case of SCAV. The actual performance gain would be dependent on the exact set-up and remains to be tested.

The changes proposed in this paper are expected to contribute to the eventual adoption of MIC accelerated architectures for production runs, in presently available implementations of Earth system models, towards exploiting the potential of a fully Exascale-capable platform.

6 Code availability

The Modular Earth Submodel System (MESSy) is continuously further developed and applied by a consortium of institutions. The usage of MESSy and access to the source code are licensed to all affiliates of institutions which are members of the MESSy Consortium. Institutions can become a member of the MESSy Consortium by signing the MESSy Memorandum of Understanding. More information can be found on the MESSy Consortium website (http://www.messy-interface.org). Changes to the source code to implement system-level heterogeneous offloading are currently specific to the Mercurium compiler, Nanos++ run time, and proprietary Parastation MPI, and are hosted on the DEEP project version control repository.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287530.

Edited by: D. Roche
Reviewed by: two anonymous referees

References

Aoyama, Y. and Nakano, J.: RS/6000 SP: Practical MPI Programming, 1999.

Beltran, V., Labarta, J., and Sainz, F.: Collective Offload for Heterogeneous Clusters, IEEE International High Performance Computing (HiPC), Bangalore, India, 2015.

Bueno, J., Martinell, L., Duran, A., Farreras, M., Martorell, X., Badia, R. M., Ayguade, E., and Labarta, J.: Productive cluster programming with OmpSs, Euro-Par 2011 Parallel Processing, Lecture Notes in Computer Science 6852, 555–566, 2011.

Bueno, J., Planas, J., Duran, A., Badia, R. M., Martorell, X., Ayguade, E. and Labarta, J.: Productive Programming of GPU Clusters with OmpSs, Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, 557–568, 2012.

Dagum, L. and Menon, R.: OpenMP: an industry standard API for shared-memory programming, IEEE Computational Science & Engineering, 5, 46–55, 1998.

Damian, V., Sandu, A., Damian, M., Potra, F., and Carmichael, G. R.: The Kinetic PreProcessor KPP – A Software Environment for Solving Chemical Kinetics, Comput. Chem. Eng., 26, 1567–1579, 2002.

Duran, A., Ayguadé, E., Badia, R. M., Labarta, J., Martinell, L., Martorell, X., and Planas, J.: OmpSs: a proposal for programming heterogeneous multi-core architectures, Parallel Processing Letters, 21, 173–193, 2011.

Eicker, N., Lippert, T., Moschny, T., and Suarez, E.: The DEEP project: Pursuing cluster-computing in the many-core era, Proc. of the 42nd International Conference on Parallel Processing Workshops (ICPPW) 2013, Workshop on Heterogeneous and Unconventional Cluster Architectures and Applications (HUCAA), Lyon, France, 885–892, 2013.

Eicker, N., Lippert, T., Moschny, T., and Suarez, E.: The DEEP Project – An alternative approach to heterogeneous cluster-computing in the many-core era, Concurrency and Computation, 2015.

Extrae: Barcelona Supercomputing Center, available at: https://www.bsc.es/computer-sciences/extrae, last access: 23 November 2015.

Jöckel, P., Kerkweg, A., Pozzer, A., Sander, R., Tost, H., Riede, H., Baumgaertner, A., Gromov, S., and Kern, B.: Development cycle 2 of the Modular Earth Submodel System (MESSy2), Geosci. Model Dev., 3, 717–752, doi:10.5194/gmd-3-717-2010, 2010.

Mallon, A. D., Eicker, N., Innocenti, M. E., Lapenta, G., Lippert, T., and Suarez, E.: On the Scalability of the Cluster-Booster Concept: a Critical Assessment of the DEEP Architecture, FutureHPC '12, Proceedings of the Future HPC Systems: the Challenges of Power-Constrained Performance, ACM New York, 2012.

Mallon, A. D., Lippert, T., Beltran, V., Affinito, F., Jaure, S., Merx, H., Labarta, J., Staffelbach, G., Suarez, E., and Eicker, N.: Programming Model and Application Porting to the Dynamical Exascale Entry Platform (DEEP), Proceedings of the Exascale Applications and Software Conference, Edinburgh, Scotland, UK, 2013.

OpenMP: Application Program Interface Version 4.0 – July 2013, OpenMP Architecture Review Board, available at: http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf, last access: 23 November 2015.

Paraver: Barcelona Supercomputing Center, available at: https://www.bsc.es/computer-sciences/performance-tools/paraver, last access: 23 November 2015.

Roeckner, E., Bäuml, G., Bonaventura, L., Brokopf, R., Esch, M., Giorgetta, M., Hagemann, S., Kirchner, I., Kornblueh, L., Manzini, E., Rhodin, A., Schlese, U., Schulzweida, U., and Tompkins, A.: The atmospheric general circulation model ECHAM 5. PART I: Model description, Report/MPI für Meteorologie, 349, 2003.

Roeckner, E., Brokopf, R., Esch, M., Giorgetta, M., Hagemann, S., Kornblueh, L., Manzini, E., Schlese, U., and Schulzweida, U.: Sensitivity of simulated climate to horizontal and vertical resolution in the ECHAM5 atmosphere model, J. Climate, 19, 3771–3791, 2006.

Sainz, F., Mateo, S., Beltran, V., Bosque, J. L., Martorell, X., and Ayguadé, E.: Leveraging OmpSs to Exploit Hardware Accelerators, International Symposium on Computer Architecture and High Performance Computing, 112–119, 2014.

Sander, R., Baumgaertner, A., Gromov, S., Harder, H., Jöckel, P., Kerkweg, A., Kubistin, D., Regelin, E., Riede, H., Sandu, A., Taraborrelli, D., Tost, H., and Xie, Z.-Q.: The atmospheric chemistry box model CAABA/MECCA-3.0, Geosci. Model Dev., 4, 373–380, doi:10.5194/gmd-4-373-2011, 2011.

Scalasca: available at: http://www.scalasca.org/, last access: 23 November 2015.

Suarez, E., Eicker, N., and Gürich, W.: Dynamical Exascale Entry Platform: the DEEP Project, inSiDE, 9, 50–51, 2011.