Accelerated Chemical Kinetics in the EMAC Chemistry–Climate Model

Theodoros Christoudias
The Cyprus Institute
PO Box 27456
1645 Nicosia
Cyprus
Email: *christoudias@cyi.ac.cy*

Michail Alvanos
The Cyprus Institute
PO Box 27456
1645 Nicosia
Cyprus
Email: *m.alvanos@cyi.ac.cy*

***Abstract*—The global climate model ECHAM/MESSy Atmospheric Chemistry (EMAC) is used to study climate change and air quality scenarios. The EMAC model is constituted by a nonlocal dynamical part with low scalability, and local physical/chemical processes with high scalability. The EMAC chemistry–climate model does not benefit from the support of accelerators which are nowadays installed in many HPC systems. We study strategies to offload the calculation of the atmospheric chemistry to accelerator technologies (GPU and Intel MIC), as in typical model configurations this is the most computational resource-demanding subtask. The proposed solutions extend the Kinetic Pre Processor (KPP) general purpose open-source software tool used in atmospheric chemistry.**

I. 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 [1]. It uses the second version of the Modular Earth Submodel System (MESSy2) to link multi-institutional computer codes.

EMAC is a distributed memory application exclusively relying on the Message Passing Interface (MPI) for parallelisation. 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.

Several trends in the usage of EMAC will push this concept to its limit and require to exploit new technologies:

- The atmosphere is represented with increasing resolution
- The complexity of chemical and physical processes considered is increased
- Ensembles of model configurations are run in parallel, multiplying the required computational resources

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

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. Describing homogeneous gas phase chemical kinetics, the MESSy submodel MECCA executes independently of its physical neighbours and is not limited by vertical adjacency relations. For the benachmark model scenario, using the EMAC 2.42 `E5M2/02b_qctm` standard namelist over one month, after restart to avoid spinup effects, more than half of the total run-time is spent in MECCA (Fig. 1a), and up to 90% for some configurations (Fig. 1b).

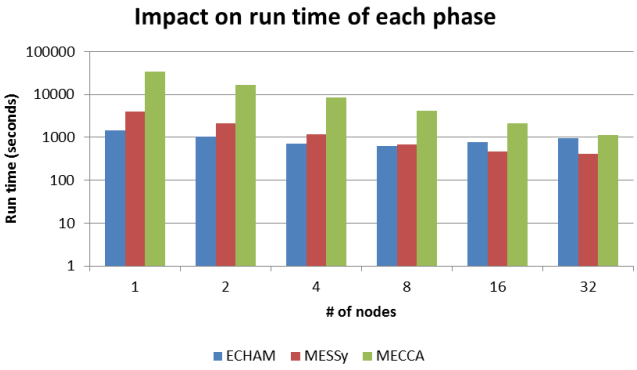
The chemical mechanism is compiled by the Kinetic Preprocessor (KPP) [2] implementing a domain-specific language for chemical kinetics. KPP is an open-source software tool used in atmospheric chemistry. Taking a set of chemical reactions and their rate coefficients as input, KPP generates code of the resulting ordinary differential equations (ODEs). Solving the ODEs allows the temporal integration of the kinetic system. Efficiency is obtained by exploiting the sparsity structures of the Jacobian and of the Hessian. A comprehensive suite of stiff numerical integrators is also provided.

Therefore, at first we concentrate the effort to support accelerator technologies on the MECCA kernel with strong algorithmic locality and small communication volume per task.

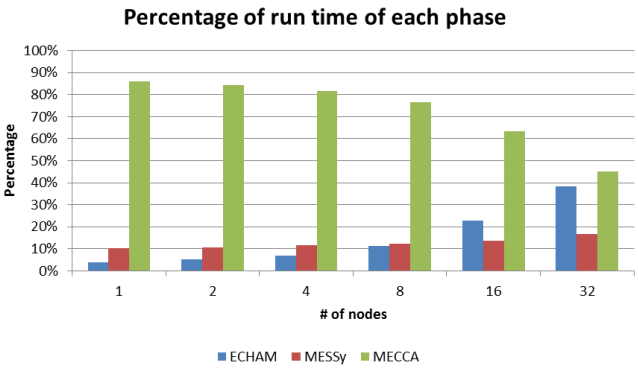
II. Scalability Considerations

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 latitudinal and longitudinal direction, to obtain a rectangular decomposition. A physical load-imbalance, caused by photochemical processes in the lower stratosphere and natural and anthropogenic emissions, appears in the run time spent for each grid point.

As can be seen in Fig. 2, the maximum MECCA kernel execution wall-time for one grid point in each column differs by up to a factor of four. The load imbalance is caused by the adaptive time-step integrator solving the differential



(a) Run Time



(b) Run Time Percentage

Fig. 1. Scaling of impact on run time and percentage of total run time for each phase of EMAC. Results obtained for climate-chemistry benchmark month with `E5M2/02b_qctm` standard namelist on the MareNostrum 3 supercomputer.

equations that describe the chemical equations computed in the MECCA submodel. The strongly varying light intensity at sunrise and sunset and night-time emissions lead to stiff differential equations that require more intermediate time steps for derivative function evaluations and increase the local computational load by up to one order of magnitude.

The major factor leading to dynamic load imbalance is abrupt change in light intensity (sunrise, sunset) combined with concentrations of precursors and gases (such as NO_x, O_3) undergoing photochemical reactions (mostly in the stratosphere over mid-latitudes) that heavily alter the stiffness of the chemical kinetics ODEs.

The ECHAM+MESSy phases only scale up to approximately a few hundred cores (blue line in Fig. 5 minimum at 8 nodes), due to the heavy all-to-all communication overhead of the spectral decomposition. At higher levels of parallelism, at or beyond approximately 1000 cores (magenta line flattening at 32 nodes in Fig. 5), the MECCA load imbalance due to the photochemistry also becomes a limiting factor.

III. Node-level heterogeneity

We have developed a source-to-source compiler that generates a CUDA [6] solver, compatible with NVIDIA GPUs, by

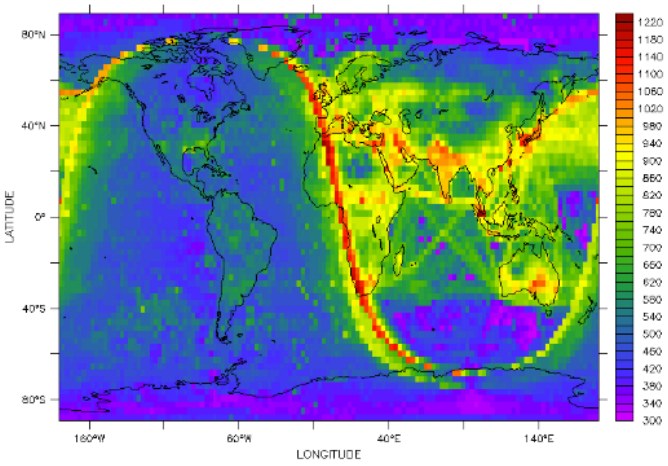


Fig. 2. Column maximum MECCA kernel execution wall-time in microseconds. The adaptive time-step integrator shows a non-uniform run time caused by stratospheric photochemistry and natural and anthropogenic emissions.

parsing the auto-generated FORTRAN code that is produced from the KPP domain specific language. Thus, the compiler is applicable on any and all presently available and future chemical mechanisms as specified by the users for each application. The CUDA compiler parses the produced source code and the FORTRAN compiler links the object file with the rest of the application.

As in the default implementation, the computation is subdivided in runtime-specified arrays of columns, with the memory of each array transferred to the GPU global memory and each grid box calculated on each GPU core. The global GPU memory size suffices for offloading the full dataset, and it is not a limiting factor for the present and future foreseen chemistry mechanism complexity. The performance is also not limited by the overuse of any function unit. Global shared memory is used to store all required data, and local register memory is used for each gridbox during computation, with cache transfers and misses being the current performance limiting factor. Each Streaming Multiprocessor (SM) contains a small amount of memory that can be used as shared or for L1. Thus, the application sets the configuration to use the most of the on-chip memory as cache rather shared memory (`cudaFuncCachePreferL1`). There is a small utilisation of GPU constant memory to store the concentrations of chemical species that remain unchanged by chemical kinetics.

Two challenges that limit the performance must be addressed. First, the naive implementation does not allow more than two CPU cores (MPI processes) from one node to offloaded to the node GPUs, thus limiting the attainable performance. The programmer either rewrite the application to achieve better scheduling and usage of the GPUs either use the *MULTI-PROCESS SERVICE* [7] that allows concurrent execution of kernels and memory transfers from different processes on the same node. Unfortunately, the last approach requires a GPU accelerator with computation capability 3.5 or higher that was not available during the experiments.

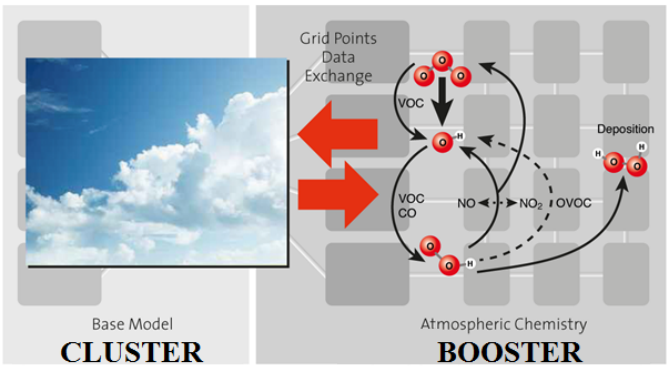


Fig. 3. Distribution of the Earth System Model components on the Cluster-Booster architecture.

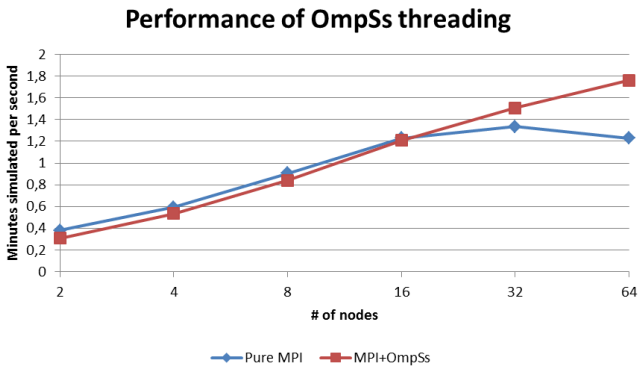


Fig. 4. Performance of OmpSs threading in the DEEP Cluster (8 MPI tasks per node instantiated and 3 threads per MPI task).

Second, the high register pressure limits the occupancy, the number of concurrent threads per Streaming Multiprocessor (SM), and increases the overall execution time of the kernel. To address this challenge we limit the number of registers per thread to 128. Unfortunately, this approach increases the usage of the stack for spill loads and stores, creating additional and possible off-chip memory traffic. Thus, the application execution time is dominated from the global memory access latency.

Work is underway to optimise the performance through task subdivision, by tiling the domain for each MPI sub-domain, and splitting the work of each gridbox to multiple GPU cores, to limit cache misses and paging and take advantage of shared memory and the specifics of the memory hierarchy on GPUs. The FORTRAN to CUDA code-to-code compiler is developed in C and Python and will be contributed upstream to the EMAC model development consortium under an open license, to be made available to the developer community. It automatically parses the automatically generated MECCA KPP solver code and produces a CUDA library that can be linked to and called directly from within the MESSy FORTRAN code at the time of compilation.

IV. System-level heterogeneity

We examine an alternative approach to heterogeneous cluster-computing in the many-core era for Earth System models on the Dynamical Exascale Entry Platform (DEEP). A set of autonomous Intel MIC coprocessors interconnected together, called Booster, complements a conventional HPC Cluster and increases its performance, offering extra flexibility to expose multiple levels of parallelism and achieve better scalability (Fig. 3).

In the same EMAC model used for the GPU case, in the MECCA submodel an integrator kernel has been created that can be offloaded onto worker threads running on the main processor or hardware accelerators [3].

A. Intranode taskification

MECCA was taskified using OmpSs [4] directives. The OmpSs Programming Model extends OpenMP with new directives to support asynchronous parallelism and heterogeneity. However, it can also be understood as new directives extending other accelerator based APIs like CUDA or OpenCL. The DEEP OmpSs environment is built on top of the Mercurium compiler and the Nanos++ runtime system.

The MECCA submodel was refactored through the creation of computational kernels for intranode parallelisation with shared-memory tasks. 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. 4, which shows the performance using multi-threading on the DEEP Cluster (with 8 MPI tasks per node instantiated and 3 threads per MPI task). Since in MECCA each gridpoint is completely independent of its neighbours, this part of the code is in principle embarrassingly parallel, with no communication or inter-task dependencies involved.

The code continues to scale beyond 32 nodes because extra threads are utilised in the gridpoint embarrassingly parallel region, without increasing the spectral component (inter-process) communication overhead.

B. Internode taskification

In the DEEP prototype hardware system, OmpSs has been extended to support offloading tasks to remote nodes [5]. Using the Booster as a pool of coprocessors, and dynamically offloading to any Booster node with enough free cores enables to eliminate the load-imbalance caused by sunlight gradients in MECCA.

The grid point arrays are rearranged in each time step to implement data locality at the gridpoint level. and minimise the memory footprint for offloaded tasks. All of the KPP integrator component of MECCA is offloaded. The time needed tranfer memory is folded in the time to launch and execute Booster tasks.

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

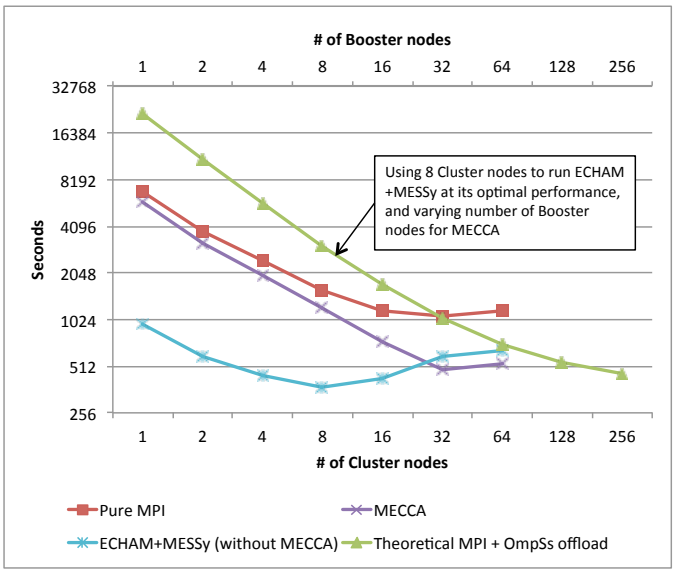


Fig. 5. Time per simulated day using a pure MPI approach, and a theoretical performance with offloading to Xeon Phi, based on the metrics collected in MareNostrum 3. The theoretical MPI + OmpSs offload data (green) is based on a fixed configuration on the Cluster using 8 nodes, combined with a number of Booster nodes between 1–256.

region within Cluster-to-Booster tasks with a variable number of individual gridpoint calculations, defined at runtime. Thus, the number of tasks to be sent to the Booster can be controlled and optimised for each architecture, based on bandwidth, to reduce task communication overheads.

C. Performance Evaluation

The DEEP Booster is in the bring-up phase, and not available to users. In order to project the performance on a full system, Xeon-based measurements on the DEEP Cluster can be combined with Xeon Phi-based measurements on MareNostrum 3. 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 8 nodes, representing a good estimate for the optimal parallelisation of that phase on the Cluster.

The projected DEEP performance scales beyond the optimal performance achieved so far. The projected attainable performance outperforms the pure-MPI conventional cluster paradigm at high core counts (number of Booster nodes) while keeping the ECHAM/MESSy MPI part on 8 Cluster nodes for optimal performance (Fig. 5).

V. Conclusion

The global climate model ECHAM/MESSy Atmospheric Chemistry (EMAC) is used to study climate change and air quality scenarios. The EMAC model is constituted by a nonlocal dynamical part with low scalability, and local physical/chemical processes with high scalability.

We have developed different approaches to enable acceleration (GPU and MIC) of the KPP chemical kinetics solver in the EMAC chemistry-climate model.

The EMAC atmospheric chemistry global climate model is well suited to exploit a system-level heterogeneous architecture, benefiting considerably from hardware acceleration.

In the long 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 (going beyond the scope of the present project), additional computationally expensive submodels like the aerosol submodels SCAV and GMXe could be addressed similarly.

The major performance bottleneck stemming from the spectral part of the model, and the increasing cost of inter-node communications can be alleviated by substitution of the dynamical core model.

Acknowledgment

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 and from the European Union's Horizon 2020 research and innovation programme under grant agreement No 675121.

References

[1] P., Jöckel, A., Kerkweg, A., Pozzer, R., Sander, H., Tost, H., Riede, A., Baumgaertner, S., Gromov and B., Kern, *"Development cycle 2 of the Modular Earth Submodel System (MESSy2)"*, Geoscientific Model Development 3, 717–752, 2010.

[2] V., Damian, A., Sandu, M., Damian, F., Potra and G.R., Carmichael, *"The Kinetic PreProcessor KPPA Software Environment for Solving Chemical Kinetics"*, Computers and Chemical Engineering 26, 11, 1567–1579, 2002.

[3] M., Christou, T., Christoudias, J., Morillo, D.A., Mallon and H., Merx, *" Earth System Modelling on System-level Heterogeneous Architectures: EMAC (version 2.42) on the Dynamical Exascale Entry Platform (DEEP)"*, Geosci. Model Dev. Discuss., doi:10.5194/gmd-2015-262, in review, 2016.

[4] S., Florentino, S., Mateo, V., Beltran, J.L., Bosque, X., Martorell and E., Ayguadé, *"Leveraging OmpSs to Exploit Hardware Accelerators"*, International Symposium on Computer Architecture and High Performance Computing, 112–119., 2014.

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

[6] Programming guide, Nvidia, CUDA, 2015

[7] Multi-Process Service, Nvidia, 2015