Alvanos, M and Christoudias, T 2017 MEDINA: MECCA Development in Accelerators – KPP Fortran to CUDA source-to-source Pre-processor. *Journal of Open Research Software,* 5: 13, DOI: https://doi.org/10.5334/jors.158

SOFTWARE METAPAPER

MEDINA: MECCA Development in Accelerators – KPP Fortran to CUDA source-to-source Pre-processor

Michail Alvanos and Theodoros Christoudias
The Cyprus Institute, PO Box 27456, 1645 Nicosia, CY
Corresponding author: Michail Alvanos
(malvanos@gmail.com)

The global climate model ECHAM/MESSy Atmospheric Chemistry (EMAC) is a modular global model that simulates climate change and air quality scenarios. The application includes different sub-models for the calculation of chemical species concentrations, their interaction with land and sea, and the human interaction. The paper presents a source-to-source parser that enables support for Graphics Processing Units (GPU) by the Kinetic Pre-Processor (KPP) general purpose open-source software tool. The requirements of the host system are also described. The source code of the source-to-source parser is available under the MIT License [1].

Keywords: GPU; CUDA; Chemical Kinetics; Climate modeling; Atmospheric Chemistry
Funding Statement: The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 675121 and grant agreement No 676629. This work was also supported by the Cy-Tera Project, which is co-funded by the European Regional Development Fund and the Republic of Cyprus through the Research Promotion Foundation.

(1) Overview

Introduction

The numerical global atmosphere-chemistry model EMAC (ECHAM/MESSy Atmospheric Chemistry) is a modular application used for climate modeling simulations [5]. EMAC uses the chemical kinetic module MECCA [8], utilizing the Kinetic Pre-Processor (KPP) general purpose open-source software tool [4] to calculate the concentrations and the interactions between different chemical species in the atmosphere. Numerically, solving atmospheric chemical kinetics is one of the most computationally intensive tasks in atmospheric chemical transport simulations.

The MECCA sub-model uses KPP to numerically solve ordinary differential equations (ODE) describing atmospheric chemical kinetics. KPP takes as input chemical reactions written in a domain-specific language and produces C and FORTRAN compatible code. The output ODE solver allows for the temporal integration of the full kinetic system. KPP utilizes the sparsity of the Jacobian matrices to increase the efficiency of the solver [9].

In typical climate simulations, chemical kinetics can take up to 90% of execution time [2]. To address this computational challenge, this paper presents a source-to-source parser that transforms the output of KPP from FORTRAN to GPU accelerated code by generating a CUDA [7] compatible solver [3]. The goal is to significantly improve the performance of numerical chemical kinetics (in terms of time-to-solution and problem complexity) in climate simulation models using GPU accelerators.

The software provides a purpose-built acceleration pathway for EMAC, rather than the general closed source solution KPP-A [6], which generates GPU-accelerated code from KPP-language directly. There have also been similar efforts in the WRF-Chem model [6].

Implementation and architecture

The implementation uses a source-to-source parser written in the Python programming language to generate a CUDA [7] compatible solver, by parsing the KPP preprocessor auto-generated FORTRAN code. Each GPU thread calculates the chemical concentrations of an individual cell. The solver uses the temporary arrays between different steps and they are allocated in the stack memory with the exception of the RCONST array that is stored in global memory. The memory required for temporary arrays depends on the configuration of the chemistry and can go up to 50 KB per CUDA thread. The intermediate values are stored in temporary arrays using the double representation. The accelerated code uses 3 KB of shared memory and 0.5 KB of constant memory when indirect arrays are not used. The parser allocation additional constant memory up to 2 KB when indirect accesses are used. All the methods that are available in the KPP numerical library under MECCA are supported.

The computation data structures are subdivided in runtime-specified arrays of columns in the atmosphere, with the memory of each array transferred to the GPU global memory and each grid box calculated on a separate GPU core to achieve massive parallelization, as shown in **Figure 1**. The CUDA chemical kinetics solver comprises three steps, also presented diagrammatically as a flow chart in **Figure 2**:

1. The first step is the calculation of the reaction rate coefficients. The variable values are stored in a global array inside the GPU and used in the computational kernels.
2. The second step is the most computationally demanding, including mostly linear algebra functions for the ODE solvers. The kernel selects the variation of the Rosenbrock solver method inside the GPU using an array of constant values in the memory.
3. The third step kernel is used for statistical reduction, and demands limited computational time compared with other kernels.

There are two files required to enable the GPU utilization: i) `f2c_alpha.py` and ii) `kpp_integrate_cuda_prototype.cu`. The pre-processor is executed by running `python f2c_alpha.py` in the `messy/util` directory.

When offloading to GPUs, the number of cells must not exceed 12288. The application calculates the number of cells by multiplying the number of columns by the number of levels for the atmosphere. The user can specify the number of columns by using the `NVL[1] (NPROMA)` runtime parameter in the configuration of the EMAC.

Quality control

To ensure the quality of the code, we conduct unit testing by comparing the GPU accelerated with a pure Fortran simulation for one model year, using 155 species and 310 reactions. We compare the output of chemical element concentrations between the CPU only and accelerated version after the first time-step to calculate the relative error. The results show a median difference of 0.00005% with the maximum difference value of 0.54% using identical inputs, ensuring the high accuracy of the chemical solver. This is well within the accuracy criterion, asserting the numerical correctness of the GPU kernel.

Finally, we compare the results of aggregated mass of the CPU-only and GPU-accelerated version, over one year of simulated time. This test aggregates the error created over time using the L1-norm method and investigates the stability of the model. The results show that the median value of the difference in aggregated mass is less than 5%, well within the expected margin of differences stemming from architecture and compiler implementations.

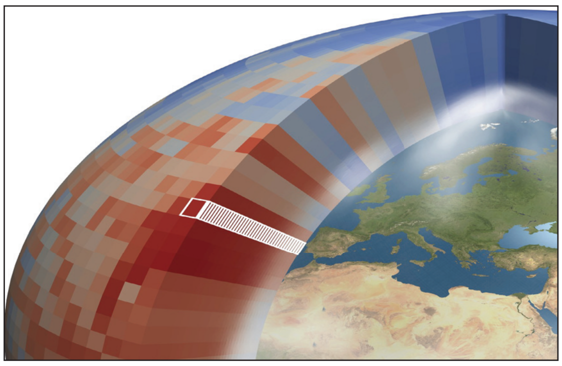

Figure 1: Grid partitioning of the atmopshere in the global model. Each GPU thread is assigned the calculation for one cell, stacked in arrays of columns.

(2) Availability

Operating system

The software is compatible with any operating system that supports the CUDA SDK.

Programming language

The script is written in the Python language and it has been tested with Python version 2.7. The script produces code that supports the the CUDA programming model version 2.0 or later.

Additional system requirements

Each CPU process that offloads to GPU requires a chunk of the GPU VRAM memory, whose size is dependent on the number of species and reaction constants in the MECCA mechanism. The number of GPUs per node and VRAM memory available in each GPU dictates the total number of CPU cores that can run simultaneously. We strongly recommend that at least 2.5 GB of VRAM per CPU process should be available on each GPU. Note that if not enough memory is available, the CUDA runtime will silently fail – without any warning.

Dependencies

The script requires Python version 2.6 or 2.7. The source code produced requires the CUDA SDK 6.5 or newer.

List of contributors

- Michail Alvanos – Active maintenance and optimization.
- Theodoros Christoudias – Conception, integration with the ECHAM/MESSy climate model, correctness/accuracy testing.
- Giannis Ashiotis – Developed an initial version of the parser.

Software location

Archive

Name: MECCA – KPP Fortran to CUDA source-to-source pre-processor.

Persistent identifier: DOI: 10.5281/zenodo.546811

Licence: MIT License.

Publisher: The Cyprus Institute.

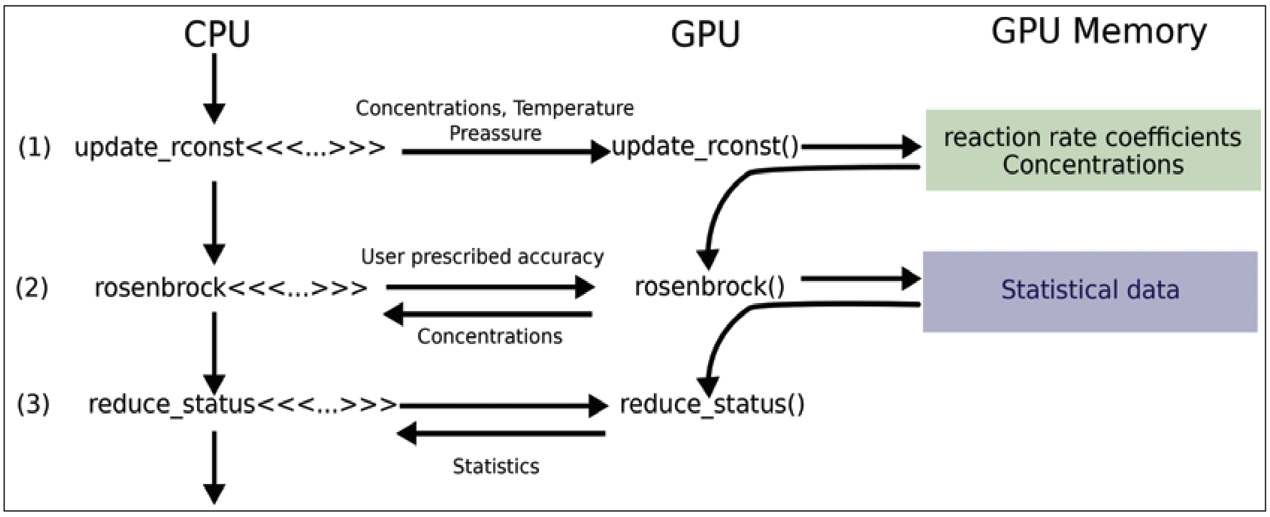

Figure 2: Tasks offload execution GPU.

Version published: 1.1.
Date published: 18/04/17.

Code repository
Name: GitHub.
identifier: https://github.com/CyIClimate/medina.
Licence: MIT License.
Date published: 12/12/16.

Language
English.

(3) Reuse potential

The source-to-source parser core can be used by other researchers to transform FORTRAN 90 code to CUDA accelerated code. The methodology can be used as template to create high performance computing capabilities on GPUs for scientific code that is currently written in FORTRAN 90.

The CUDA compatible ODE solvers comprises the Runge Kutta methods and the linearly-implicit Rosenbrock family of solvers, and may be used for chemical kinetics and numerical integration outside the field. The software is supported by the developers. Any inquiries should be addressed by e-mail.

Acknowledgements
The authors would like to thank Patrick Jöckel for his support during the packaging and integration of the source-to-source parser and Klaus Klingmueller for the graphics in **Figure 1**.

Competing Interests
The authors have no competing interests to declare.

References

1. **Alvanos, M** and **Christoudias, T** 2016 'Medina: KPP Fortran to CUDA source-to-source pre-processor'.
2. **Christou, M Christoudias, T, Morillo, J Alvarez, D** and **Merx, H** 2016 'Earth system modelling on system-level heterogeneous architectures: EMAC (version 2.42) on the Dynamical Exascale Entry Platform (DEEP)', *Geoscientific Model Development* 9(9), 3483. DOI: https://doi.org/10.5194/gmd-9-3483-2016
3. **Christoudias, T** and **Alvanos, M** 2016 Accelerated chemical kinetics in the EMAC chemistry-climate model, In *'High Performance Computing & Simulation (HPCS), 2016 International Conference on'*, IEEE, pp. 886–889. DOI: https://doi.org/10.1109/hpcsim.2016.7568427
4. **Damian, V, Sandu, A, Damian, M, Potra, F** and **Carmichael, G R** 2002 'The kinetic preprocessor KPP – a software environment for solving chemical kinetics', *Computers & Chemical Engineering* 26(11), 1567–1579. DOI: https://doi.org/10.1016/S0098-1354(02)00128-X
5. **Jöckel, P, Kerkweg, A, Pozzer, A, Sander, R, Tost, H, Riede, H, Baumgaertner, A, Gromov, S** and **Kern, B** 2010 'Development cycle 2 of the modular earth submodel system (messy2)', *Geoscientific Model Development* 3(2), 717–752. DOI: https://doi.org/10.5194/gmd-3-717-2010
6. **Linford, J C, Michalakes, J, Vachharajani, M** and **Sandu, A** 2009 Multi-core acceleration of chemical kinetics for simulation and prediction, In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ACM, p. 7. DOI: https://doi.org/10.1145/1654059.1654067
7. **Nvidia, C** 2015 'Programming guide'.
8. **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** 2011 'The atmospheric chemistry box model CAABA/MECCA-3.0', *Geoscientific Model Development* 4(2), 373–380. DOI: https://doi.org/10.5194/gmd-4-373-2011
9. **Zhang, H, Linford, J C, Sandu, A** and **Sander, R** 2011 'Chemical mechanism solvers in air quality models', *Atmosphere* 2(3), 510–532. DOI: https://doi.org/10.3390/atmos2030510

How to cite this article: Alvanos, M and Christoudias, T 2017 MEDINA: MECCA Development in Accelerators – KPP Fortran to CUDA source-to-source Pre-processor. *Journal of Open Research Software,* 5: 13, DOI: https://doi.org/10.5334/jors.158

Submitted: 09 December 2016 **Accepted:** 20 April 2017 **Published:** 28 April 2017