Automatic port to OpenACC/OpenMP for Climate & Weather Code using the CLAW Compiler

Valentin Clement[1], Philippe Marti[1], Xavier Lapillonne[2], Oliver Fuhrer[2], William Sawyer[3]

[1]ETH Zurich, Center for Climate Systems Modeling (C2SM), Zurich, Switzerland
[2]Federal Office of Meteorology and Climatology MeteoSwiss, Zurich, Switzerland
[3]CSCS Swiss National Supercomputing Centre, Lugano, Switzerland

1. Porting the ICON model to hybrid architecture

In order to prepare for heterogenous supercomputer architectures, the global climate model ICON [4, 2] is being ported to accelerators. The major part of the porting is achieved using OpenACC [5] compiler directives. For time-critical components such as physical parameterizations, code restructuring and optimizations are necessary to obtain optimal performance.
In some cases, these GPU-optimizations may have a negative impact when running the same code on a CPU architecture. In order to address such performance portability [3] issues without imposing disturbing changes to the code base, the CLAW Single Column Abstraction (SCA) [1] was introduced. It is designed to address the physical parameterizations of atmospheric models which are horizontally independent so each column can be computed separately. With this approach, the physical parameterizations are written in Fortran only considering the vertical dependencies. The CLAW Compiler can transform the code for a specific target architecture and insert compiler directives such as OpenMP or OpenACC.
In this poster, we introduce a special case of the CLAW SCA dedicated to the ELEMENTAL functions and subroutines applied to the JSBACH soil model.

2. ICON JSBACH soil model and ELEMENTALs

JSBACH is the soil model of ICON in global configuration. The code takes advantage of ELEMENTAL function or subroutine and vector notation. In its current shape, the code is not suited to be ported easily to OpenACC or OpenMP as compilers are not so permissive with PURE or ELEMENTAL functions and subroutines. Nevertheless, an ELEMENTAL function or subroutine can be seen as a specific case of the CLAW SCA. Indeed, the soil model is a point-wise computation or a column-wise computation with a single vertical level. Code transformation allows to achieve performance portability from a single source code.

Some numbers about the JSBACH soil model:

- Heavy use of Fortran 2003/2008
- ~30 tasks pipelined
- ~30'000 LOC
- 35 ELEMENTAL functions / 33 ELEMENTAL subroutines
- 1 to N kernels generated by function/subroutine depending their size

3. Automatic port to OpenACC/OpenMP

Lines 3 and 7 of Figure 1 are the only additional CLAW directives inserted into the code to drive the transformation. Those specify which fields coming from the whole model have additional dimensions.

```
1 ELEMENTAL SUBROUTINE calc_radiation_surface(swvis_down, swnir_down, alb_vis, alb_nir, lw_down, t, rad_net, &
2       swvis_net, swnir_net, sw_net, lw_net)
3   !$claw model-data
4   REAL(wp), INTENT(in) :: swvis_down, swnir_down, alb_vis,  alb_nir, lw_down, t
5   REAL(wp), INTENT(out) :: rad_net
6   REAL(wp), INTENT(out), OPTIONAL :: swvis_net, swnir_net, sw_net, lw_net
7   !$claw end model-data
8   REAL(wp) :: zswvis_net, zswnir_net, zsw_net, zlw_net
9
10  ! Compute net SW radiation from downward SW and albedo
11  zswvis_net = swvis_down * (1._wp - alb_vis)
12  zswnir_net = swnir_down * (1._wp - alb_nir)
13  zsw_net    = zswvis_net + zswnir_net
14  ! Compute LW net radiation from incoming and the thermal radiation
15  zlw_net = lwnet_from_lwdown(lw_down, t)
16  ! Compute net radiation
17  rad_net = zsw_net + zlw_net
18
19  IF (PRESENT(swvis_net)) swvis_net = zswvis_net
20  IF (PRESENT(swnir_net)) swnir_net = zswnir_net
21  IF (PRESENT(sw_net)) sw_net = zsw_net
22  IF (PRESENT(lw_net)) lw_net = zlw_net
23 END SUBROUTINE calc_radiation_surface_net
```

Figure 1: Typical JSBACH ELEMENTAL subroutine

The CLAW SCA transformation for ELEMENTAL is only triggered for the GPU target. It performs the following steps:

- Change attributes of the function/subroutine signature or duplicate it if needed.
- Insert DO statements to iterate over domain dimensions.
- Perform data dependencies analysis and apply promotion on variables that needs it.
- Insert OpenACC or OpenMP compiler directives for parallelization and data movement based on the data dependencies analysis.
- Data movement strategy and parallelization strategy are configurable from the command line or the CLAW configuration file.

The dimensions information of the model are stored in a TOML configuration file. The same file is used by all the transformation in JSBACH.

4. CLAW Compiler

Open-source source-to-source compiler for Fortran 2008 code. Transforms CLAW directives to produce **optimized code** for **specific target architecture** and **compiler directives**. Based on the **OMNI Compiler Project**[6]. **Modular** and **easily extensible** to new transformations with a plug-in.

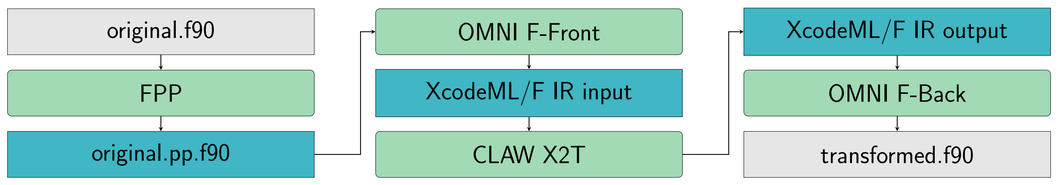

Figure 2: CLAW Compiler workflow

Fortran code is pre-processed and then parsed to the XcodeML/F[7] IR. This IR represented as an AST is then manipulated by CLAW X2T to produce the different version of the code for a specific target with inserted directives. Finally, the IR is decompiled to standard Fortran code before being compiled by default compilers.

```
1 $ clawfc --target=gpu --directive=acc -o transformed.f90 original.f90
```

Figure 3: Call the CLAW Compiler for a GPU target with OpenACC directives

5. Current results

Early results of our approach are shown here. Figure 4A shows the execution time of three tasks (T1, T2 and T3) representative of the computation patterns and size found in JSBACH. All results are obtained with Cray Compiler CCE 8.7.3 on Piz Daint.

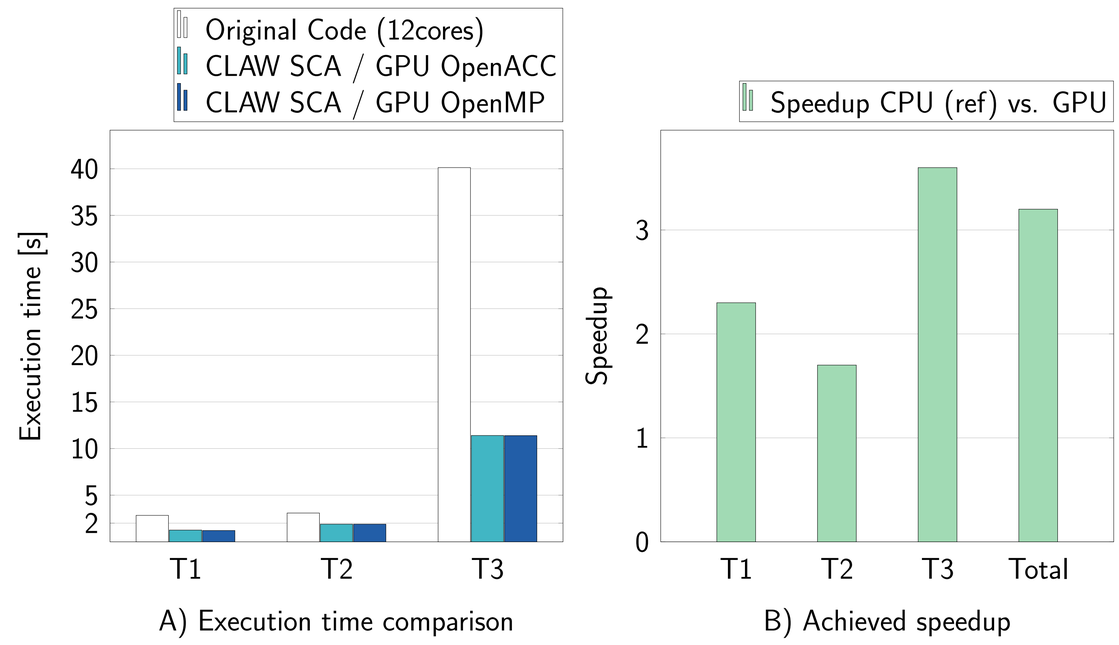

Figure 4: Performance comparison (socket to socket) of JSBACH tasks on Intel Haswell E5-2690v3 and NVIDIA P100. Domain size (number of horizontal grid points × vertical levels) = 20480×47.

- The code in the repository contains **zero OpenACC/OpenMP directives**. Versions are **automatically generated**.
- **3.6x** speedup on the longest task and **3.2x** speedup achieved for the total execution with no change in the original code besides the introduction of 1 CLAW block directives as shown in Figure 1.
- OpenACC and OpenMP versions of the generated code give similar performance.
- Hand-written version would require **20 times more** directives to achieve the same results.
- **Maintenance of the code is simplified** since CLAW re-generates the OpenACC/OpenMP directives when new changes come in. Data movement are always up-to-date.
- The source code stays **100% Standard Fortran**.

This approach is currently applied to the rest of soil model used in ICON and will be the only accelerated available version.

References

[1] V. Clement, S. Ferrachat, O. Fuhrer, X. Lapillonne, C. E. Osuna, R. Pincus, J. Rood, and W. Sawyer.
The claw dsl: Abstractions for performance portable weather and climate models.
In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '18, pages 2:1–2:10, New York, NY, USA, 2018. ACM.

[2] T. Crueger, M. A. Giorgetta, R. Brokopf, M. Esch, S. Fiedler, C. Hohenegger, L. Kornblueh, T. Mauritsen, C. Nam, A. K. Naumann, K. Peters, S. Rast, E. Roeckner, M. Sakradzija, H. Schmidt, J. Vial, R. Vogel, and B. Stevens.
Icon-a, the atmosphere component of the icon earth system model: Ii. model evaluation.
Journal of Advances in Modeling Earth Systems, 10(7):1638–1662, 2018.

[3] O. Fuhrer, C. Osuna, X. Lapillonne, T. Gysi, B. Cumming, M. Bianco, A. Arteaga, and T. C. Schulthess.
Towards a performance portable, architecture agnostic implementation strategy for weather and climate models.
Supercomputing frontiers and innovations, 1(1):45–62, 2014.

[4] M. A. Giorgetta, R. Brokopf, T. Crueger, M. Esch, S. Fiedler, J. Helmert, C. Hohenegger, L. Kornblueh, M. Köhler, E. Manzini, T. Mauritsen, C. Nam, T. Raddatz, S. Rast, D. Reinert, M. Sakradzija, H. Schmidt, R. Schneck, R. Schnur, L. Silvers, H. Wan, G. Zängl, and B. Stevens.
Icon-a, the atmosphere component of the icon earth system model: I. model description.
Journal of Advances in Modeling Earth Systems, 10(7):1613–1637, 2018.

[5] X. Lapillonne and O. Fuhrer.
Using compiler directives to port large scientific applications to gpus: An example from atmospheric science.
Parallel Processing Letters, 24(1), Mar. 2014.

[6] Omni Compiler Project - An Infrastructure for Source-to-Source Transformation, 2013-2019.
http://omni-compiler.org.

[7] XcalableMP Specification Working Group.
XcodeML/Fortran Specification.
Language specification, RIKEN AICS, Kobe, Japan, July 2017.