Poseidon: A Source-to-Source Translator for Holistic HPC Optimizations of Ocean Models on Regular Grids

Maurice Brémond, Hugo Brunie, Laurent Debreu, Rupert W Ford, Florian Lemarié, Anna Mittermair, Andrew R Porter, Julien Rémy, Philippe Rosales, Martin Schreiber, et al.

Poseidon: A Source-to-Source Translator for Holistic HPC Optimizations of Ocean Models on Regular Grids

Maurice Brémond[1], Hugo Brunie[1,3], Laurent Debreu[1,3], Rupert W. Ford[2] †, Florian Lemarié[1], Anna Mittermair[4], Andrew R. Porter[2], Julien Rémy[1,3], Philippe Rosales[1,3], Martin Schreiber[1,3], Martin Schulz[4], Sergi Siso[2], Arthur Vidard[1,3]

[1]AIRSEA team, Inria - LJK, Grenoble, France - [2]Hartree Centre, STFC, Warrington, United Kingdom

[3]Univ. Grenoble Alpes, Grenoble, France - [4]TUM School of Computation, Information and Technology, Munich, Germany

SC24 ATLANTA NOV 17–22

Main objectives of Poseidon

1. Utilization of **existing Fortran code** (the **main asset**).
2. Numerics experts can still write numerics code in Fortran **without hand-written HPC optimization**.
3. Develop **method to automatically analyze, optimize and parallelize such numerics code**.
4. Output is **source code** which is **compatible to existing ocean models** including used parallel programming models.
5. First goal: **Models** with **regular grid structure**

Abstract

- **Ocean simulation** models play a **crucial role in weather** forecasts and **climate** change simulations.
- However, they are often **far from the top performance** of computer architectures due to legacy programming styles.
- Often, **entire codes are rewritten** which is often **too costly, time**-consuming and **error** prone.
- In addition, **highly performing ocean models** require a **co-design**, including **MPI**, **automatic differentiation**, and many other features.

This raises the research question "Whether it is possible to **translate existing Fortran code** to reach **top performance** without rewriting of the entire code?" motivating "Poseidon".

Overview of the Poseidon approach

Stage 1: Input - PDE solvers written in Fortran

Input: Long-term goal is the application to the Nucleus for European Modelling of the Ocean (NEMO) [1], Coastal and Regional Ocean Community Ocean (CROCO) [2] model and to other developments (e.g., Flash-X).
Case study: Research code using the numerics of the barotropic solver of the CROCO model
Example: Multi-step extrapolation and computation of total height (in Croco model)

$$\zeta^{m+\frac{1}{2}} = \left(\frac{3}{2}+\beta\right)\zeta^{m} - \left(\frac{1}{2}+2\beta\right)\zeta^{m-1} + \beta\zeta^{m-2}$$
$$D^{m+\frac{1}{2}} = \zeta^{m+\frac{1}{2}} + h$$

```
do j = jmin, jmax
    do i = imin, imax
        Drhs(i,j) =   cff1 * zeta(i,j,kstp) &
                    + cff2 * zeta(i,j,kbak) &
                    + cff3 * zeta(i,j,kold) &
                    + h(i,j)
    end do
end do
```

Stage 2: Uplift through a PDE model-specific connector

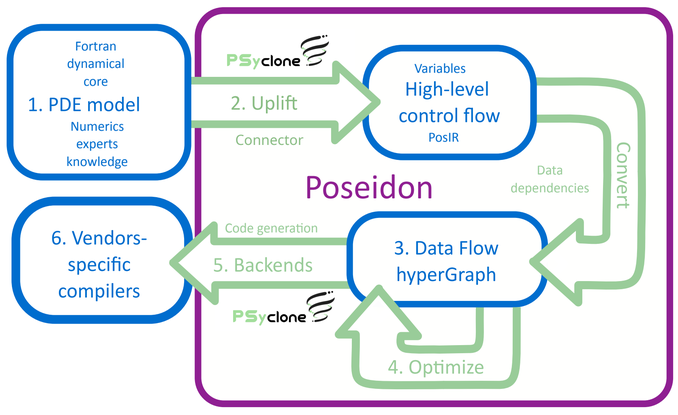
We **extract** variables, computations, and their semantics with a model-specific connector based on PSyclone [3]. After this step, **all relevant information** is known about every variable and every access in the dynamical core.

2.a Variables scope:
High-level information and meta data about **all variables** inferenced from the code with the **PDE model-specific uplifter**.

2.b Control Flow: Nodes of the following types with a closure describing their input and output data dependencies. Based again on PDE model-specific uplifter.

- Kernel Loops: DIM **nested loops** performing **stencil-like** operations, based on **PosIR**
- Boundary Conditions: (DIM-1) **nested loops on boundaries**, with conditions, based on **PosIR**
- Computations: All other computations, e.g., precomputed friction coefficients, based on **PosIR**
- Communications: High-level abstraction of distributed communication
- Black Boxes: Anything we do not yet support but can extract the closure from (I/O, printing, calls)

Poseidon Intermediate Representation (PosIR):
- List of assignment operations
- Assignment = numerical expressions
- Stencil-oriented
- Branches expressed as conditions



Stage 3: Convert to a Data Flow Graph

- Convert to a **data flow graph** (see image to the right) with **hyperedges** between nodes representing **data flow**.
- This results in a **high-level intermediate representation** for further **graph-based operations** applicable in a **safe way.**

Stage 4: Graph-based operations

E.g. on kernel loops:
- Loop **splitting**
- Kernel **fusion**

Works in progress:
- Automatic differentiation
- Injection of communications

Stage 5: Code generation backends

Create code for PDE model
- Pure Fortran with PSyclone
- Parallel programming models for now: OpenMP, optional offloading to GPU

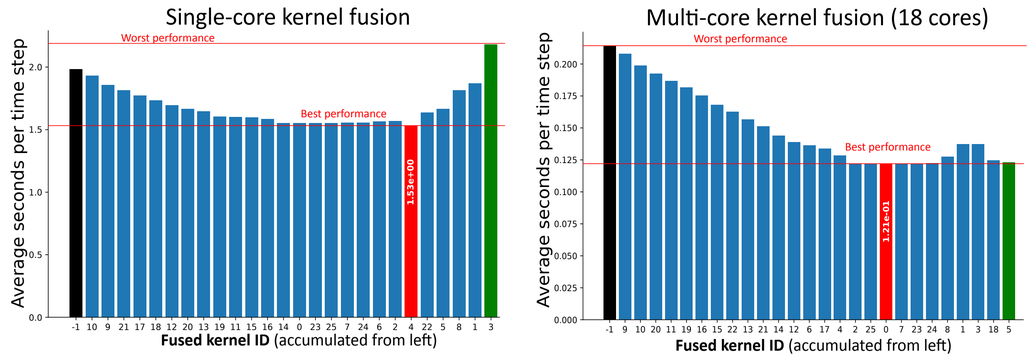
Future work: Other parallel prog. models, IRs and DSLs

Stage 6: Leverage vendors-specific compilers

- Poseidon is **independent of a specific compiler** as the next step in the compilation chain.
- Consequently, Poseidon can **leverage further optimizations** as performed by existing low-level and eventually hardware-specific compilers and communication backends.

Preliminary kernel fusion results

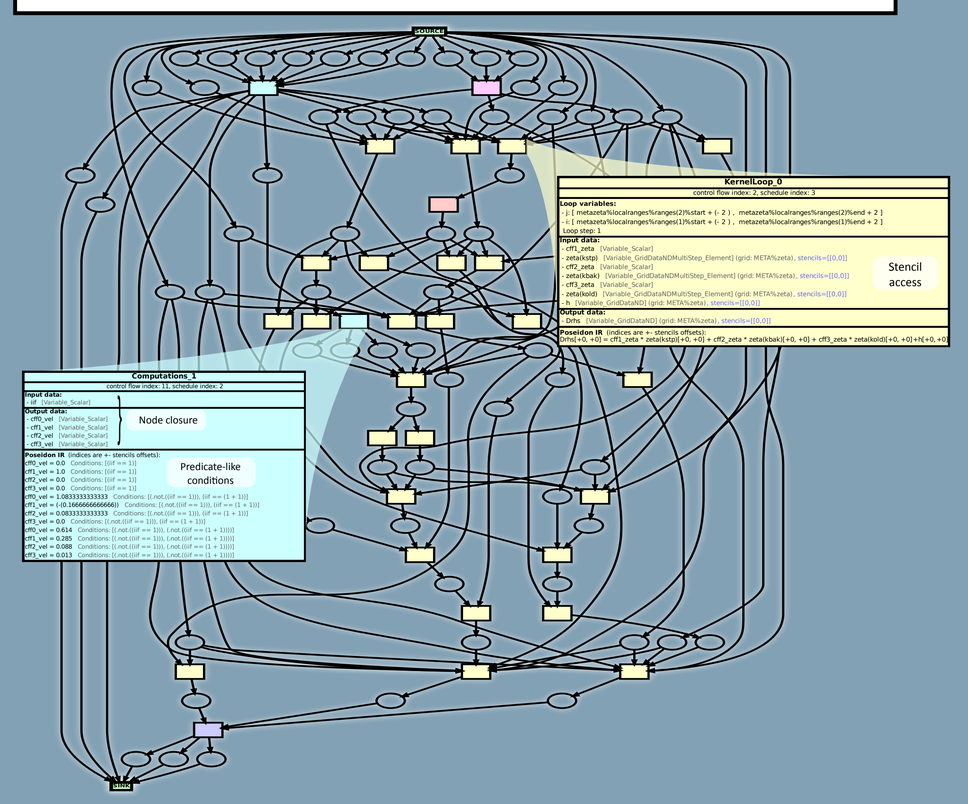
PDE **model**: **2D nonlinear barotropic solver** using an AB3/AM4 multi-step time integration scheme with all hand-optimizations undone.
Experiment: Naive **autotuning of kernel fusions** using Poseidon, see below.
Hardware: Intel® Xeon® W-2295 CPU @ 3.00GHz. / **Compiler**: nvfortran version 24.5-1.
Scalability: Single-core and multi-core with OpenMP directives on 18 cores, without MPI, single CPU.
Correctness: Tested against output of original source code.



Kernel fusion by autotuning:
- Each "Fused kernel ID" is based on a kernel fusing all kernels of ID from left to this kernel ID. E.g., in left image, "21" refers to fusing kernels 10, then 9 and finally 21. No fusion is denoted by "-1".
- Selection of next kernel ID to fuse based on **steepest descend offline autotuning**.
- The last bar on the right shows the results of a **fusion of all possible kernels**.

Discussion of results:
- Single-core vs. multi-core kernel fusion: **Kernel fusion provides higher speedups on multi-core** systems due to effects of roofline model.
- Speedup of x12 reached between the **best version on 1 core** (left graph) versus **best version on 18 cores** (right graph).
- Speedup of x16 reached between fully fused single core run (left graph) and best multi-core fusion.

Example of data flow hypergraph: Uplifted from a non-linear 2D barotropic multi-stepping solver with two of its Poseidon IR high-level nodes detailed.



Poseidon limitations

Uplifting:
- Requires to be **adapted to each model** (but Poseidon is python3 code, easy to get involved).
- Requires **correct extraction** of the numerics and control flow.

Reliability:
- Failure of correct uplifting can lead to wrong results.
- 60% of the code is currently covered with unit tests (pytest).
- Continuous integration tests on 2 research code mini-applications.

Summary

- **Poseidon**: A new uplifting approach to a data-flow-oriented intermediate representation.
- **Target**: **Holistic HPC optimization** of PDE models on regular grids written in Fortran.
- **Avoids**
  - **hand-tuned codes** (making code often less understandable),
  - **entire rewrites of code** in another domain-specific language (often not accepted by numerics experts),
  - **introducing errors in numerics** by rewriting code, e.g., in new domain-specific language
- **Approach:** Uplifting brings back the semantics of the numerical model through PDE model-specific connectors. Further steps allow performance optimizations on data flow graph.
- **Retains the original Fortran code** which is considered to be a **highly-valuable asset**.

Future research directions

Near-term goals:
- Developing uplifting connectors to **production codes**, including NEMO ocean model.
- **Reduce arithmetic intensity** by simplifying kernel numerics after kernel fusion.
- Implementing **more graph operations**, e.g., array buffer reuse and loop fusion.
- Automatic **distributed-memory communication** and **latency hiding strategies**.
- **Automatic differentiation** of data flow graph followed by graph-based optimization.

Long-term goals:
- Support other **programming models** (OpenCL, OpenACC, etc.) and **hardware** (e.g., FPGAs)
- Extension to **unstructured grids**

Related work

Usage of DaCE [5] to optimize FV3, an Earth System Model, originally written in Fortran. Optimizations are similar to those aimed with Poseidon, but the PDE solver has been entirely rewritten in Python3.

Psyclone [3] is a source-to-source Fortran compiler written in Python. It allows us to easily parse Fortran code and create our own Intermediate Representation based on a Data Flow Graph in Poseidon.

In [4] Nick Brown et al. present an extension of the Flang LLVM MLIR Compiler to extract stencil patterns from Fortran source code. Writing their framework as a compiler pass, they are able to leverage existing MLIR dialects and transformations. Contrary to this approach, e.g., our source-to-source code transformation framework allows us to use any general purpose compiler.

References

[1] Gurvan Madec, et al. "NEMO Ocean Engine Ref. Manual", Scient. Notes of IPSL Clim. Mod. Center, 20.6.2023
[2] Francis Auclair, et al. Coastal and Regional Ocean Community Model. 2.0, 22 April 2024
[3] Ford, R., et al. Psyclone. 2.5.0, GitHub, 14 Feb. 2024, doi:10.5281/zenodo.11190458.
[4] Brown, Nick, et al. "Fortran perf. optimisation and auto-parallelisation by leveraging MLIR-based domain specific abstr. in Flang.", Proc. of the SC'23 Workshops of the Int. Conf. on HPC, Network, Storage, and Analysis. 2023 (SC'23)
[5] Tal Ben-Nun, et al. "Productive performance engineering for weather and climate modeling with Python". In Proceedings of the International Conference on HPC, Networking, Storage and Analysis (SC '22).

Acknowledgements: We thank "Initiatives de recherche à Grenoble Alpes" **(IRGA) Poseidon project** for partial funding of this work. We also received fundings from **Inria associate team "Crocodiles"** (DRI-012415). This work made use of the **Grid'5000** testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations.