Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2019-28-RC1, 2019

Geoscientific Model Development Discussions

Interactive comment on "OpenArray v1.0: A Simple Operator Library for the Decoupling of Ocean Modelling and Parallel Computing" *by* Xiaomeng Huang et al.

Anonymous Referee #1

Received and published: 6 March 2019

OpenArray v1.0: A Simple Operator Library for the Decoupling of Ocean Modelling and Parallel Computing by Huang et al. describes a first implementation of a set of operators in a C++ library for writing a 3D ocean model. The basic concepts behind OpenArray are described and explained, and an initial set of operators is used to program a 3D ocean model to mimic the numerics of the Princeton Ocean Model (POM).

The technical aspects of this discussion paper are of broader interest to the climate modelling community, at least to those colleagues that are working more on the software engineering site rather than in pure geosciences. Despite my remarks and suggestions below, I find the OpenArray approach attractive, and the GOMO use case

makes the paper complete for a first release of the software and its publication.

I recommend a revised version of the paper for publication in Geoscientific Model Development.

P.S.: I like your conclusion, especially the last sentence.

1 General Comments

While the discussion paper is well structured and clearly written I am missing some parts which I outline below. In my opinion the publication as a whole does not need to be restructured or rewritten but I suggest to extent/rewrite/restructure the introduction by describing the state of the art in somewhat more detail. I wished the authors had mentioned their solution for IO and provided a discussion section to share their experiences and opinion about the pros and cons of their approach.

Unfortunately, I did not succeed to install OpenArray on an OSX system (macOS 10.14.3, Armadillo 9.2, Boost 1.66, LLVM 7.0, gcc/g++/gfortran 8.3, openmpi 3.0)

1.1 Introduction

While the general motivation to start the OpenArray approach is made clear in this paper I am missing a more complete discussion of the state of the art. A few approaches (ATMOL, ICON DSL, STELLA) are listed but the text does not provide useful hints in how far OpenArray really goes beyond existing approaches. I am missing ATLAS (DOI: 10.1016/j.jcp.2017.07.006) . I am not an expert in this field, but to me ATLAS seems to cover several design aspects, in particular operators, support for parallelism, and support for different grid types, and seems to be in these aspects similar to OpenArray. The ESCAPE project and its follow-up ESCAPE2 worked or will work in this direction. The

authors cite Lawrence et al., 2018 but only as a reference for a trend towards the usage of “heterogeneous and advanced computing platforms”, even though Lawrence et al. also discuss software approaches to address these challenges, including concepts like those used by OpenArray.

1.2 IO

As a model without IO is pretty useless it would have been nice to have read a few lines about how the (parallel) IO is approached. It is included in OpenArray, so why not spending one paragraph on such an important issues as well, perhaps with some graph showing the IO performance.

1.3 Discussion

You are convincing in the sense that your approach is valid and takes major burden from the oceanographer who “only” wants to run and modify an ocean model. On the other hand the complexity has not magically disappeared but it is moved from GOMO (in this example) to OpenArray. When porting the whole software onto a new system the major effort now goes into OpenArray - which is fine, but it has to be done. How complex is this? How flexible is the OpenArray approach in this respect.

If the unlucky oceanographer comes up with the idea to try out yet another (perhaps higher-order) advection or any other scheme which is not yet supported by OpenArray, how difficult is it to extend OpenArray? Does this require expert knowledge and support from OpenArray developers?

How seamless is - in your opinion - the integration of other stencils into the OpenArray library?

I am not insisting on answering these questions line by line but rather take these as

suggestions for what could be addressed in a thorough discussion. You as authors may wish to stress different - and in your opinion more important - points.

2 Specific Comments

Line 39 and elsewhere: consider to replace “climate model” by “Earth system model”, as the latter is now mainly used when talking about multi-component models in the context of Earth system and climate modelling efforts.

Line 41 ff: Please rephrase the sentence, as computing platforms are not applied but used.

Line 55: Which gap do you precisely have in mind? Do you really mean climate science in general or rather climate modelling (aka Earth system modelling)?

Line 70 ff: What is the former, what is the latter language? Could you briefly explain to the non-experts amongst the readers the difference between source-to-source and DSL? Perhaps this whole paragraph needs some restructuring (see remarks in my section 1.1).

Line 90: Please introduce the reader to the heterogeneity you have in mind here. What makes TaihuLight different in terms of heterogeneity? From the system specification further down in your text, TaihuLight does not look that heterogeneous.

Line 100: I would say that you solved the problem for one ocean model or a particular class of ocean models but not yet for ocean models in general.

Line 109: Is it really valid Fortran code, or shouldn’t it better be classified as pseudo Fortran code as GOMO cannot really live without the OpenArray library?

Line 111: Is it really meant like that you compile and link one executable which can then be executed on any computing platform? Or are you talking about the intermediate C++

code? But this would require compiling and linking on the target system before it can be executed.

Line 120: True but only if the OpenArray has been ported to and is available on the Sunway platform. There is probably no free lunch when moving to a new hardware or software environment.

Line 148 and elsewhere: Equation 1 is probably taken from the POM user manual, but nevertheless should not expressions like $\frac{\partial DU}{\partial x}$ rather be written as $\frac{\partial}{\partial x}(DU)$?

Line 152: Could provide some hint on how to arrive at the discrete expression (2)

Line 211: I thought your current implementation of the operators only supports uniform (=equidistant) grids? What am I missing here?

Line 214 and elsewhere: I suggest to avoid the phrase “automatic”. Nothing is done automatically but every effect has a cause. Here, something is happening because you programmed it that way and some conditions are coming together to trigger an action.

Line 238 ff: Could you clarify how the Arakawa grid type, the jumping rules and the differential operators are linked together? Let us assume I have formulated my ocean model on an Arakawa C grid and now for curiosity would like to run it on an A grid (not because it would really makes sense but to demonstrate the effect of discretisation on the numerical solution) what would I have to change in my ocean model code?

Line 247 ff: What is the motivation for the list of ocean model codes you provide in this paragraph? There are other codes around, e.g. FESOM (see https://fesom.de and the list of publication there) or an unstructured grid model for global ocean dynamics by P. Korn (see https://doi.org/10.1016/j.jcp.2017.03.009) and several more.

Line 274 section 3.1:

Could you add a few lines to describe the handling of lateral and vertical boundary conditions within your operators?

Line 334: When speaking of subgraphs and the kernel function, can the individual advection and diffusion terms be accessed for diagnostic purposes?

Line 352: I am not sure what is meant here and perhaps the sentence should be rephrased. Such a function needs to be programmed once for a particular ocean model, but once it is there it can be used, see e.g. ESMF_FieldBundleHalo contained in the Earth System Modeling Framework (ESMF). To my knowledge, other ocean models use similar approaches for the halo exchange as well. But no doubt, it is a relief to have it.

Line 391: Is the mode splitting algorithm inherited from POM, if so, this should be mentioned.

Line 422: Could you provide the number of lines for OpenArray as well? I could calculate it myself but . . .

Line 425: You raise the impression that porting is not an issue anymore. While this is certainly true for GOMO (which is of course very valuable) the porting still has to be done for OpenArray. Maybe this could be clarified somewhere (perhaps in the discussion).

Line 469 section 5.3: Which hardware do you use for these tests? What sets the upper bound of 4096 processes?

Line 481 section 5.4: What does this mean for a reasonable local domain size? 32X32 points as in sec. 5.3 is still on the good side, while 9x9 as used on TaihuLight does shows some performance degradation.

Line 490 ff: As I understand the steps up to compiling the JIT are done only once at the beginning of a job. If you run a longer experiment (in terms of wallclock time or number of timesteps) the initialisation phase should be negligible when compared to the total run time. Why don’t you provide two numbers, one for the initialisation and one for the integration within the time loop?

3 Technical Corrections/Suggestions

L39 climate model → climate models

L42 climate community → climate modelling community

L43 model program needs → model programs need

L68 inefficiency → inefficiently

L83 the sentence needs to be reordered. It is not clear (to me) to which part "at the product level" is referring to.

L106 change to : . . . is similar to the original but manually optimized parallel program.

L108 support → supports

L125 → The implementation

L139 → In traditional ocean models . . .

L143 → When using the OpenArray library . . .

L211 → we propose

L220 → . . . the horizontal velocity components Array(U) and Array(V) are . . .

or

. . . the horizontal velocity Array(U, V) is . . .

L238 can be used → are used

L257 different → difference ; operator → operators

L289 will be concealed by → is hidden behind

L290 → . . . and can escape . . .

L303 → to implement a so-called lazy expression . . .

L318 . . . AYF are the interpolated functions → . . . AYF are the average functions.

L373 computing processing elements : aren't these Central Processing Units (CPUs)

L384 a practical ocean model → a numerical ocean model

L404 rephrase, the TKE and alike can be calculated but not the submodel.

L505 a practical ocean model → a numerical ocean model

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2019-28, 2019.