Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2017-167-AC2, 2017

Geoscientific Model Development Discussions

# *Interactive comment on* "Parcels v0.9: prototyping a Lagrangian Ocean Analysis framework for the petascale age" *by* Michael Lange and Erik van Sebille

**Michael Lange and Erik van Sebille**

e.vansebille@uu.nl

Received and published: 26 September 2017

Reviewer 2: Dr Yu Cheng

*The authors introduce a new Lagrangian Ocean Analysis framework, Parcels that is built with flexibility, scalability, and ease-of-use in mind. This manuscript serves well as a proof-of-concept and demonstrates its user-friendly interfaces and the accuracy of the codes. However, the claimed improvement of performance, scalability for large datasets and ease to integrated with different OGCMs are yet to be seen.*

*The authors' effort to build Parcels following modern software engineering practice is*

*much appreciated. From my experience, none of the existing tools, although gradually migrated to Github, use CI tools, making them difficult to be implemented to and tested on different machines. This could be one of the reasons that each of community only reaches a certain size. Positioning itself as a framework, together with its modular design, Parcels has much more potentials than any existing tools to attract more users from different fields. At its early stage of development, Parcels urgently needs the input from a larger community, and this paper is a well-constructed invitation. I recommend the paper to be published once some of following points are addressed. Also, some of my questions and comments are also listed:*

We thank Dr Cheng for these very kind words. Below we have listed our responses to his comments

*P.2 line 6-8: Is there a limit that you expect each of three codes will not be computationally efficient anymore? From my own experience, instead of computation, the bottle neck of CMS is in the "output to NetCDF" step. Opening a huge NetCDF file and dumping particle information into it drain the system memory (32G/16core node) and dragged down the whole machine.*

This is a very good point. While it is beyond the scope of this paper to speculate when the other code to not be computationally efficient anymore (that also depends on development by their teams), we do agree that particle trajectory writing can be a major bottleneck too. We have now highlighted this in the introduction (p 2 of the track-changed pdf, lines 9).

*P.2 line 28: It is not clear to me what "generic particle-mesh interaction computations." is.*

The sentence has been rewritten to make the intention clearer (p 3 of the track-changed pdf, lines 3-4).

*P.5 line 29: Is this limited to a particular JIT package? How much faster is using*

*JITParticle than using ScipyParticle? It'd be good to see a benchmark comparison. If I want to use a new Interpolation scheme, how can I implement it into both modes?*

The JIT components used by Parcels are tightly integrated into the code generation engine and are internal to Parcels. The primary advantage of using code generation and JIT processing is that the low-level interpolation routines can be tightly coupled with the processing kernels via inlining, which is orders of magnitude faster than the pure Python mode, due to the large overhead of calling native Python functions repeatedly in tight loops. This is now explained in more detail in the newly added section 2.3 (p 6 and 7 of the track-changed pdf, lines 23-4), and an "anecdotal" performance benchmark has been added to illustrate the performance improvement due to JIT mode.

*P.8 line 15 and the codes. Please correct me if my understanding is incorrect: The "snapshots" was defined in the code to load the data of initial three time steps. The for-loop was so designed because the particles are back tracked. The pset.execute uses the loaded filedset to advect particles in these three days (runtime) with time step (5min) and output every one day, and then "fldset.advancetime" is called to replace the last snapshot by concatenating the newly loaded snapshot to the front.*

This is indeed correct; we have now further clarified the use of the advancetime function in sections 3.1 and 3.4 (p 9 of the track-changed pdf, lines 19-21).

*P.7 line 8, and P.8 line 15: It is confusing to me that "Snapshots" was first a list of three members. Within the loop, it got updated to a single member list, which is the condition to trigger advancetime. Also, it might be better to point out that the "set_ofes_fieldset" is a user-defined function, not included in the Parcels package.*

In the revised version of the manuscript, we now explicitly state that set_ofes_fieldset is indeed a user-defined function (p 8 of the track-changed pdf, lines 9-11). We have also clarified that this function accepts a list of any length.

*Is there a particular reason that field.py uses the native Netcdf4 API to load in fields,*

*but uses Xarray to write to output files? Performance concerned?*

The use of the native netCDF API is largely an overhead from earlier development stages and will be addressed in future releases. The plan is to incorporate Xarray with the scheduling library Dask to facilitate efficient out-of-core file reads that complement the optimised particle computation. The primary work on this is planned for the upcoming performance optimization and parallelization phases.

*Section 4.2: How did you generate the idealized flow fields? It might be trivial, but the grid-size information is missing for some cases.*

The reviewer was right that not all of the test cases provided sufficient information on how the fields were generated, and at what resolution. We have now added that information (p 11 of the track-changed pdf, lines 8-10, 12-15 and 23; p 12 of the track-changed pdf, lines 8-9, 18-19 and 29; and p 13 of the track-changed pdf, line 12).

*Section 4.2.7: I am just curious why $K_h = 100m^2/s$? From my experience, this value depends on resolution. Let's say if I want to use Parcels with 1/10 deg OFES, what value should I use? Is there a test case that I can tune this value against some observations?*

The $K_h = 100m^2/s$ here was purely chosen as an illustrative example. The reviewer raises a very good question about appropriate values for $K_h$, but we feel that (or even if Brownian motion is a good model of diffusion in the ocean anyway) is beyond the scope of the paper here. We know that quite a few groups, including for example Arne Biastoch's in Kiel, are actively investigating the question raised here.

*P.12 line 16: Not exactly clear what "Spatial indexing methods" indicates. Also, this sentence sounds not clear to me: How will something "impacted" become promising in the future? How are you planning to do the "close integration of such scheduling techniques with the particle loop?"*

Section 5.1 has been reworded to explain potential performance optimizations in more detail, including the use of spatial indexing methods to reduce the overall field data volume. Additional references have also been added for background on geospatial indexing (p 13 of the track-changed pd).

*Section 5.1.1: How are you planning to proceed on this front? I know there is a "Parallel NetCDF" project, but I don't think its Python API will be available anytime soon. How much of performance gain could be achieved once the second point is addressed in version 1.0?*

Various projects exist that improve on the current shortcomings of the netCDF Python wrappers. The most popular and established package is probably Xarray, which provides parallelisation (via Dask) and out-of-core computation, and could potentially be used to implement grid parallelisation based on spatial decomposition methods.

*Section 5.3: This is exciting! It would be wonderful to be able to couple Parcels to many ocean models. However, to do so, Parcels need to work on their native grids. It is possible to run a remapping layer in between, but I think that will significantly impact the performance. Just curious, could you briefly suggest how to modify the field.py to handle the non-regular grids, for example, the tri-polar grid in POP2, or the unstructured triangular meshes in FESOM? What's your current plan to support multiple grids? Or will that be left to the users?*

The newly added section 2.3 adds a few details about the current assumptions made in the code about field interpolation, and its potential for future extensions (p 6 and 7 of the track-changed pdf, lines 23-24). The primary idea is that the current Field class acts as an abstract base class for future field types with individual field-specific interpolation routines provided as macros to the code-generation engine to include at runtime. While this will require developer input and expertise, such additions would be relatively concise and should not affect existing user models, allowing a relatively easy transition between offline and online models.

Please also note the supplement to this comment:
https://www.geosci-model-dev-discuss.net/gmd-2017-167/gmd-2017-167-AC2-supplement.pdf