Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2017-167-AC1, 2017
© Author(s) 2017. This work is distributed under
the Creative Commons Attribution 4.0 License.

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

Response to Reviewer 1: Dr Joakim Kjellsson

The manuscript presents a new framework for a Lagrangian particle model, Parcels. The new particle model is in a testing phase with only the most basic components functional. The authors describe the current workings of the model, test its accuracy, and present where they envision development going further. Overall, the manuscript is well written. The main novelty lies in presenting the new framework to the particle-modelling community and its possible future developments. However, there are very few actual results. I recommend the paper for publication, but only after addressing the

comments below.

We thank Dr Kjellsson for these kind words, and have indeed addressed all his comments, as detailed below

There is not much discussion about how exactly the new model will be more suited to cope with petascale age computing. The authors spend some time talking about how to optimise the loop over particles to improve performance, but with petascale OGCMs, where velocity fields amount to hundreds or more terabytes, reading and interpolating those fields into the particle model will be a huge bottleneck. Section 5.1 has a paragraph on how reading data from massive files could work, but there is no demonstration. In the practical example, the data file is 6Gb, which is not very large. I understand the authors have not focused on optimisation of PARCELS yet, but are there any examples, not necessarily with particle codes, where spatial indexing has given a performance improvement? I strongly recommend more discussion (in the introduction, design, and discussion sections) about how all current particle codes, e.g. CMS, Tracmass, Ariane, will hit this bottleneck in the peta-scale age, and more details about how PARCELS will overcome it.

We agree that the future performance challenges have not been addressed in sufficient detail and have made additions to the suggested sections 1 (p 2 of the track-changed pdf, lines 9-12), 2.2 and the new 2.3 (p 6 of the track-changed pdf, lines 18-32 and page 7 of the track-changed pdf, lines 1-4), 3.1 (p 8 of the track-changed pdf, lines 5-8) and 5.1 (p 14 of the track-changed pdf, lines 1-20).

The overall rationale follows the argument that, as no "silver bullet" solution is as yet known for the big-data challenges facing Lagrangian tracking applications with petascale OGCMs, we see the primary limitations of existing codes in the lack of flexibility and dynamic adaptability that is required to explore new optimisation strategies. Throughout the listed sections we explore two potential solutions for dealing with the vast volumes of field data required, for two different usage scenarios of Lagrangian tracking models: a) Coupled execution with the host model to avoid the bandwidth limitations of disk and file storage; and b) reducing the required data volume by selectively prefetching field data based on known particle positions. We aim to highlight throughout that both approaches have been considered during the design of Parcels to enable efficient exploration of such schemes during future development.

Since PARCELS is very flexible, could it be extended to work for atmospheric particles? Perhaps PARCELS should not be presented as a tool for Lagrangian ocean analysis, but rather Lagrangian particle tracking in both atmosphere and oceans? Presenting this kind of framework to the atmospheric modelling community as well could be beneficial, but would mean changing the paper quite a bit. Even if the authors decide to stick with presenting PARCELS as an ocean particle code, atmospheric particle codes still need some mentioning (MetOffice NAME model, FLEXPART) in the introduction.

This is an interesting suggestion. While our own development for now focusses on oceanographic applications, the framework could indeed in principle also be used in the atmosphere. We have now added some discussion of atmospheric particle tracking codes in the introduction (p 2 of the track-changed pdf, lines 27-30).

Throughout the paper, the authors name the model "Parcels". However, more than once I found that the name of the model could be confused with actual parcels. Why not use PARCELS, as any other model (e.g. NEMO, CESM, IFS etc.) to avoid confusion?

We thank the reviewer for this thoughtful suggestion, but we feel that the current convention is more concise and in the spirit of the Python philosophy. The suggested all-caps naming style is largely derived from Fortran90 coding conventions, where routine and package names are often capitalised, while it is preferred in the Python community to use lower-case module names (https://www.python.org/dev/peps/pep-0008/package-and-module-names). Since this is largely a matter of personal taste, we feel that we must respectfully decline.

Page 1, Line 1: "petascale age" is rather unspecific. The sentence uses future tense,

suggesting we are not there yet, even though there are already > 1petaflops computers. Please specify what is meant. OGCMs of a certain horizontal resolution, e.g. global $\equiv$ 1/50 or $\equiv$ 1/100?

We have now clarified in the abstract that by petascale age we mean output of OCGCM models exceeding the 1 petabyte storage space barrier (p 1 of the track-changed pdf, lines 2-3). This would for example be the case for a global simulation at 1/50o run for 50 years and stored at daily intervals.

Page1, Line 19: Add reference for seawater parcels: Doos 1995, Blanke Raynaud 1997.

We have now added references to Döös 1995 and Blanke Raynaud 1997 to this sentence (p 1 of the track-changed pdf, line 19)

Page 2, Line 16: How would it keep up? By being very efficient at reading in velocity data?

We have now clarified that indeed we see scalability and efficiency in reading in hydrodynamic data as key strategies to keep up with OGCM development (p 2 of the track-changed pdf, lines 21).

Page 2, Line 18: I recommend replacing "functionality such as a myriad of behaviours to the particles" for "active particle behaviours".

We have changed this phrasing following the reviewer's suggestion (p 2 of the track-changed pdf, line 23)

Section 2.2: I found the section a bit confusing, and I think it needs some rewriting to become clearer. As I understood it, these are two methods for interpolating data, e.g. velocity, onto the particle position? I'm familiar with the interpolator from SciPy, but what method does the JIT method use? Is that something the user can write himself/herself? Is the SciPy interpolator restricted to nearest-neighbour or linear interpolation methods? Also, what pre-defined macros are you referring to?

To make this clearer we have separate the details of the provided interpolation routines from the overview of the execution loop and added a new subsection 2.3 (p 6/7 of the track-changed pdf, lines 23-4). This now clearly states the current implementation details, its restrictions and potential for additional interpolation schemes in the future.

Page 6, Line 11-13: Non-compatibility with non-regular grids excludes quite a few OGCMs, which often use rotated pole, tri-polar or cubed-sphere grids. I think this is one of the most important shortcomings of v0.9 that must be addressed soon by the authors or the user community. The authors should say so.

We fully agree with the reviewer, and indeed extension to curvilinear grid is the next major development goal for Parcels. We also agree that we should state this major limitation more upfront, and have now mentioned it in the abstract too (p 1 of the track-changed pdf, line 15).

Page 7, Line 5-7: This sentence does not read well with its two parenthesis. I would split into two sentences

We have changed the phrasing following the reviewer's suggestion (p 8 of the track-changed pdf, lines2-4).

Page 8, Line 13: 6Gb is actually not a very large file. Many laptops have 16Gb RAM these days and could definitely cope with this while the user sips his/her coffee at some hip cafe.

We have limited the data set here to 6GB in order to keep the downloading of the data manageable, for users who want to run this experiment themselves. We thought some users might find it problematic to be asked to download 100s of GB. This example is meant to highlight the API, not per se to show the scalability of the code (which will be optimised in future version of Parcels anyways). The 6 GB of data was hence a compromise between 'manageable' download volume and sufficient data to conduct a meaningful experiment.

Section 3.5: Does PARCELS write CF 1.6 compliant data?

The output of Parcels is indeed CF-1.6 compliant, following the guidelines at http://cfconventions.org/cf-conventions/v1.6.0/cf-conventions.html#discrete-sampling-geometries. We have now explicitly mentioned that at the beginning of section 3.5 (p 10 of the track-changed pdf, lines 2-4).

Sections 4.2.1 - 4.2.7: The test cases need to be described a bit more. Are the fields generated within PARCELS, or generated and stored as netCDF files and then read into PARCELS? Also, please give Δx, Δy, Δt for all fields.

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).

Page 10, Line 28: "steady-state"

We have fixed the type-o (p 12 of the track-changed pdf, line 15)

Section 5.1: Are there any tests that show that the optimisations they propose would give some performance improvement? Optimising the reading of velocity fields from very large files would be one of the main strengths of this model. See major comment above.

Unfortunately, there is no hard evidence or citable publications for any of the proposed methods yet, which is why further experimentation is required. The general concepts of out-of-core streaming for large data-sets have been very skillfully addressed by a combination of two prominent Python packages recently, namely xarray and dask. This method significantly reduces the memory overhead of large files, and could possibly be extended to incorporate particle-specific information to further optimise the file reads, as outlined in the re-written parts of section 5.1 (page 14 of the track-changed pdf). The flexible design structure and native compatibility with these Python packages will allow for efficient exploration of such advanced methods in future work.

Please also note the supplement to this comment:
https://www.geosci-model-dev-discuss.net/gmd-2017-167/gmd-2017-167-AC1-supplement.pdf

---

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