Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-75-SC3, 2020

Geoscientific Model Development Discussions

Interactive comment on "Developing a common, flexible and efficient framework for weakly coupled ensemble data assimilation based on C-Coupler2.0" *by* Chao Sun et al.

Lars Nerger

lars.nerger@awi.de

Received and published: 15 June 2020

Dear Dr. Liu,

Dear Dr Lars Nerger, Thanks a lot for your introductions, comments and suggestions.

We are sorry about the misunderstandings regarding PDAF in the manuscript. We will correct them when revising the manuscript based on more discussions with you.

My time to perform this discussion is unfortunately very limited. The essential information to correct your statements about PDAF is already in my initial comment. Thus, I will keep this second comment short.

Here, I'd like to briefly introduce the background about DAFCC and this manuscript. We began to design and develop DAFCC in 2017 according to the requirements from operational model development in China. We noted your PDAF work after someone introduced it to us in 2018. Then we know that PDAF is a pioneer of our work especially after your latest GMDD manuscript was online. To make DAFCC known by potential users and to share some "new" aspects in developing a DA framework, we submitted this manuscript to GMD for possible publication. We had to state the differences between PDAF and DAFCC because it will be easily asked what are new advancements from the state of the art in the review process.

Well, claiming that the system you describe solves apparent limitations of PDAF, which as I explained is incorrect, is an interesting strategy when you want to point out "new aspects". Obviously, there is also the significant risk to be inaccurate. To my impression there are enough differences in the strategy to use a complex coupler compared to the approach of PDAF to by-pass the coupler. This difference is also in the situation that PDAF is designed to be compatible with any model coupler (even C-Coupler), while one would need to figure out how to use the C-Coupler based system if a coupled model is already implemented with a coupler like OASIS-MCT. The obvious advancement I see is that when one has a coupled model that already uses C-Coupler, it might be easier to add the data assimilation functionality - at least if there is a data assimilation algorithm already coded for the model as in the example in your article.

There would be differences between PDAF and DAFCC regarding the generation of communicators, where PDAF relies on users' efforts while DAFCC automatically generates communicators implicitly. According to our experiences from the cooperation with Chinese model teams, we just feel that, it is not easy to develop the codes for generating the communicator of the ensemble of a component model in the ensemble run of a coupled model, especially for scientists.

This experience doesn't coincide with ours. The templates for the communicator

setup that we provide with PDAF are readily usable. Thus, there is no particular "users' effort".

We inferred that PDAF requires all model ensemble members use the same number of processes and the same parallel decomposition, and only makes the processor cores of the first ensemble member available to the DA algorithm, based on the PDAF codes (e.g., version V1.15.1). For example, in the "SUBROUTINE PDAF_get_state" in the PDAF code file "PDAF-D_get_state.F90", the root process in COMM_couple corresponding to one ensemble member gets state variables from the remaining processes in COMM_couple corresponding to other ensemble members. Based on the examples available in the code package, we know that the global communicator is generally split into a set of COMM_couple each of which corresponds to the ith process of all ensemble members. We really do not know how to organize COMM_couple and whether "PDAF_get_state" still works, under the case that model ensemble members use different numbers of processes or different parallel decompositions, or a DA algorithm uses all processes of the ensemble. Could you please show some examples about that case? Thanks.

The key is in deed in the setup of COMM_couple. I think this case is analogous to that of letting all processes perform the analysis step, which is discussed further below. (Nonetheless, I'm not aware of any situation where a different decomposition would be useful for an ensemble of equivalent model states where one does not know before the actual computation that any of them would be particularly faster or slower than the others)

We are sorry of that "... efforts should be made to enable the software compilation system of the model to compile the code of the DA methods." is incorrect. How about "... efforts should be made to enable the software compilation system of PDAF to compile the code of the DA methods."

If you also intend to write " ... efforts should be made to enable the software

compilation system of C-Coupler to compile" your alternative sentence would make sense. Obviously, also the code of C-Coupler and the assimilation code you use need to be compiled. Thus, it might be even more effort in your case since C-Coupler and the assimilation codes are really separate.

Regarding your comments "However, implementing them is a pretty trivial task. Actually, this operation is always model-specific and an analogous operation even happens when using a coupler like C-Coupler. However, in this case it's in the coupler API instead of the assimilation API.". In our opinion, it will be not a pretty trivial task but a heavy task even like developing a new coupler when the DA algorithm uses a different parallel decomposition. C-Coupler has formalized model-specific operations for data transfer among different component models or different parallel decompositions with standard APIs, like other couplers. So DAFCC that is based on C-Coupler2 does not require users to conduct such tasks.

With PDAF this is not at all a 'heavy task'. Let me show this on two examples which are in the model binding codes provided by the PDAF package:

For the ocean model FESOM in AWI-CM, we use lines like

```
DO i = 1, myDim_nod3D
   state_p(i + offset(2)) = uf(i)
END DO
```

which writes the meridional velocity into the state vector. Analogously, for the ocean model MITgcm we use

```
      DO bj=myByLo(myThid),myByHi(myThid)
         DO bi=myBxLo(myThid),myBxHi(myThid)
            DO k=1,Nr
               DO j=1,sNy
```

```
                DO i=1,sNx
                   koffset = koffset + 1
                   state_p(koffset) = uVel(i,j,k,bi,bj)
                ENDDO
             ENDDO
          ENDDO
       ENDDO
    ENDDO
```

where the two outermost loops provide support for the threading in MITgcm and the other loops run simply over the three indices of this 3D field. Actually, we have done such implementations for various models and the scheme is always the same. For me, that's 'pretty trivial'.

Regarding your comments "Thus by letting more processes compute the DA algorithm, the overall speedup will be limited. In contrast, using more processes for the analysis step requires remapping of the domain decompositions. This requires more MPI communication calls, which will take more time than just collecting ensemble information on the first ensemble task. Thus, while one gains speed in the analysis one looses time in the remapping. What is faster will be case-dependent.". We agree that what is faster will be case-dependent. That's why we try to offer a maximum number of processes to DA algorithms while a DA algorithm can only use a part of processes it can effectively use in real cases. Could you please show us a detail example how PDAF enables a DA algorithm to use all processes in the ensemble of a component model(the DA algorithm will generally use a very different parallel decomposition from the model ensemble), without modifying the PDAF codes (e.g., version V1.15.1) and with trivial efforts in developing call back functions.

I don't have a detailed example at hand, and don't have time to prepare one for this discussion about correcting statements about PDAF in your article. Anyway,

I can shortly describe what's required.
Since PDAF itself has no remapping functionality small changes to the user code are required. As global filters are not relevant for high-dimensional models, I'm considering here the localized filters like LETKF and LESTKF. In these filters, PDAF has a loop running over local analysis domains, and a finer domain decomposition will shorten these loops. Further, I'm considering here the case that each model subdomain used in the forecast is further decomposed, which should be sufficiently flexible.
In short the modification would be as follows: One modifies the parallel setup so that COMM_couple consists of a single process each, which switches off communication in COMM_couple. Then, in the call-back routines (namely in prepoststep_pdaf.F90) one would add a subroutine to collect the ensemble on the sub-subdomains of the model (thus, on chunks of the state vector) and to distribute the ensemble states again after the analysis update. Now, one adapts the number of local analysis domains for the local analysis loop and adds an offset for the loop counter when accessing the state vector elements according to the finer domain decomposition. Then, the filter runs over a smaller number of local analysis domains for each process, without further modification of the code. This code version would also be compatible wit the case that the filter is executed on a lower number of precesses or only on the processes of the first model task.

I hope that these additional explanations made my initial comments about the PDAF features even clearer.

Kind regards,
Lars Nerger

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-75, 2020.