2020-09-10: IODA Engines Converter Tutorial

Yannick opened the meeting by announcing the focused topic discussion of the day: A tutorial on writing a ioda-converter that is compatible with the new ioda-engines.

Before proceeding to the tutorial, Yannick had a few announcements. First, a reminder that in preparation for the release there will continue to be many pull requests that may require action from you, for example updating models. If you are asked to review a PR please do it in a timely manner. Or, if you are unable to address it soon, please ask someone from your group to address it. An example is the MPI cleaning one that was just merged but there are more coming, in particular one that will require changes for anyone using 4Denvar (this includes at least LFRic and MPAS, maybe others?).

Yannick also reminded everyone of two upcoming code sprints, one for writing tests (Sept 21-25) and one for writing documentation (Oct 5-9). Please let Yannick know if you wish to participate, if you haven't already.

Then Steve H walked us through the tutorial presented on the following slides:

IodaEnginesConv...erTutorial.pptx

The slides have detailed information and examples. Just one note of clarification here: on slide 13 Steve described how to attach attributes to variables. The JEDI team is working with the OBS team to define required attributes for different data types. But, there is at least one attribute that is required for all data: make sure you specify the units for each Variable.

When Steve completed his presentation the floor was opened for questions. Jianjun asked about the case when the input and output variables for the converters are different. Steve acknowledged that this is a common occurrence, though he did not focus on it today. All converters need to map variable names from those that are in the file to ioda naming conventions. The JEDI team is working with the OBS team to define standard naming conventions for ioda. And, they are looking into tools that will allow users to specify these translations in yaml files.

Jianjun then pointed out that, in practice, radiance data often has many locations and hundreds or even thousands of channels. He asked if ioda converters are fast enough to process such data in an operational setting. Steve explained that there is continuing work to optimize ioda-converters. This includes efforts to move selected converters from python to a compiled language (C++), parallelization of the data processing with MPI, and processing the files in chunks so all data isn't necessarily read into memory simultaneously.

Ling then asked about the limitations of specifying data types for ioda-engines. Steve responded that ioda engines has a flexible interface that can handle all the fundamental data types defined by C++, plus std::string. And, the data can have an arbitrary number of dimensions. On a related note, Wojceich later asked whether it's possible to store vectors of variable length. Steve said they had considered this capability but decided that this is most relevant for conventional data like radiosondes which is only a small fraction of the total data. It is more important to optimize the format to handle radiance data which dominates the observation data volume and which does tend to have vectors of equal length. So, they decided to handle radiance and other conventional obs by storing the data in contiguous arrays of locations with marks to note missing data.

Eric asked about the old vs new variable naming conventions described on slide 4: should we be using the old format with the "@" symbols or the new format based on slashes, "/"? Steve responded that the new format is preferable and it's possible that we will eventually deprecate the "@" format. But, he added that this is pretty ingrained and would be disruptive to make this change abruptly so it's likely that the old format will still be supported for some time.

Steve V asked if the source code for today's examples is available. Steve H said that the slides will be made available (above). There are more examples that are located in the ioda-engines repo (see the slides for the paths to the example code described in the presentation).

Chris S asked if anyone was maintaining a list of who is writing converters for what. This would be helpful to avoid redundant work. Yannick said such a list was not being maintained in JEDI and asked Dick about whether the OBS group is keeping track of this. Dick said not at the moment but it is a good idea and he made a note to address this in the future.

JJ asked if the current converters will become obsolete. Steve said yes, eventually they will be re-written based on a new API provided by the application "pybind11" which provides inter-operability between python and C++. So, the new python API will look much like the C++ API that Steve was describing today. So, the old NcWriter class will become obsolete soon. The new common ground for the converters and ioda will be the ObsGroup class now provided by ioda-engines.

Rahul asked whether there is a mechanism for read in data through ioda-engines and pass it to ufo. Steve responded yes - the examples today focused on writing but you can also specify a back end that reads data into ioda. Rahul asked if one needed to define an ObsGroup before writing data. Steve said this is not technically required but it is highly recommended to facilitate the interface. Rahul then asked if it's now possible to access ObsGroups from ufo. Steve responded that, like ioda-converters, JEDI itself still has some work to do to fully migrate to ioda-engines. So, the full functionality is not yet there but is coming soon.

Praveen asked if it was possible to access ioda-engines from fortran. Steve said yes - ioda-engines has a fortran API that is nearly done - about 95% complete.

Yannick then adjourned the meeting, encouraging people to contact Steve or Ryan directly if they had any further questions.