

2020-07-16: Atlas

Yannick opened the meeting by announcing the special topic, which is the use of Atlas in JEDI. Atlas is currently used to provide a generic data structure for the saber/bump interface but we are looking into other ways it may be used in the future, including possibly for increments and interpolation.

Yannick then introduced our featured speaker of the day, Willem Deconinck, who is the lead developer of Atlas for ECMWF. After some technical difficulties, Willem proceeded to present the following slides



The following is a brief summary - please refer to the slides for further details.

Atlas was originally developed to explore and support new numerical schemes for the IFS forecast system at ECMWF, particularly finite volume methods based on unstructured meshes. It was introduced in 2017 and is now available as an open source project distributed by ECMWF on Github. Development continues at ECMWF, with ongoing contributions from JCSDA, Met Office and MeteoFrance. It is written in C++ with Fortran interfaces; each C++ class has a corresponding derived type in Fortran.

Atlas functionality includes structured and unstructured grid generation, parallel partitioning (mesh generation), distributed computations (halo exchange, divergence, gradient, laplacian), interpolation, and spherical harmonic transforms. The spherical harmonic transforms are based on two distinct implementations. The first uses the ECMWF transi library while the second uses the open-source FFTW library and is entirely native to Atlas. Currently, transi is still proprietary but ECMWF is considering granting public access. The native implementation is public access but is currently limited to inverse transforms (spectral to spatial) and serial applications (as opposed to parallel).

Grids options are currently categorized into Structured and Unstructured abstract classes. Structured grids have the characteristic that latitudes are aligned (each latitude has a number of longitude points, which can vary). As Dan would later explain, this classification does not include cubed sphere grids, which are technically structured but not in this sense. So, Dan and collaborators are working to introduce cubed sphere grids as a third abstract grid option.

Once a grid is defined, the next step is to define a mesh, which includes a parallel data decomposition and connectivity information that is used for halo exchange. The halo generation algorithm can be called recursively to extend a previous halo; for example, from 1 to 2 points. The functionspace class defines how data is discretized on a mesh and can be used to define Field objects that hold the data. Data can be stored on nodes or edges. Fields are abstract containers that hold data and can be created by specifying the data type and dimensions. The data in fields is accessed through array views. The functionspace controls halo exchange for Fields. Each concrete instantiation of a Grid class includes a projection method that will convert internal coordinates (e.g. x,y) into lat, lon.

Using the NEMO ocean grid as an example, Willem described how new functionality can be incorporated into Atlas through plugins (slide 14). These plugins are rendered as dynamic libraries that can be loaded at run time with no need to change any atlas code. One advantage of this is that proprietary code can be kept private.

Atlas provides tools to help the user with actions such as grid parameter specification (see the atlas-grids tool example on slides 12 and 13). Willem added that by adding and registering new functionality through the plugin feature, the built-in tools will become accordingly aware of the plugin feature. For example, the atlas-grids tool will display grid specifications for the NEMO ocean grid when enabled through the plugin mechanism.

Grids of different resolution can be created with overlapping domain decompositions to minimize communication during interpolation (slides 16, 22, 23). Interpolation for grids that do not have overlapping domain decompositions is less efficient but can be useful. They are working on this capability but it is not yet available.

At the conclusion of Willem's presentation, Rahul asked about the capability to change domain decompositions at will. As an example, he asked if one could rearrange data so that a particular MPI task has an entire 2D slice of data (lat/lon), with data distributed over vertical level. This could be used to do a local operation like an FFT, for example, and then redistribute data back the way it was. Willem responded that there is currently a scatter/gather operation that can collect all of a 3D field onto single PE/task but they're still working on the slice capability that Rahul described. An example that can be done now is to undo a tiling decomposition for output into a single file.

Guillaume asked if the differential operators (gradient, divergence, laplacian) have adjoints. Willem said not yet but this is planned - also for interpolation.

Jake asked if atlas can support an unstructured grid of any type. Willem responded that a grid is just a collection of points but interpreted the question in terms of a mesh. Atlas can currently support a triangular or quadrilateral mesh or a combination of both. He said this could be extended as needed.

Tom asked about the efficiency of the GPU performance that Willem described. Willem answered that atlas is not doing computations itself - it enables GPU usage but benchmarking efficiency would depend on the application, which they have not yet done. Willem added that atlas does take steps to optimize for GPU efficiency such as padding arrays to 32 elements to align with GPU caches.

Andrew Lorenc asked how atlas handles ensembles. Willem said that the object-oriented approach will let you create as many objects as you wish to represent an ensemble. But, Yannick clarified the question and described how some applications may benefit from the ensemble member as the innermost dimension of a field. Willem said that one could create a functionspace that did this but then there was some debate on how generic this would be.

Then Dan presented some work in progress on adding a cubed sphere grid to atlas - see slides for details:



After Dan's presentation, Mark M asked about how easy it would be to incorporate a user-defined mesh into this framework. For example, if a model in JEDI were using a cubed sphere grid with a different parallel decomposition, would the modeler have to define a new mesh subclass in atlas to handle this? Willem answered that one could create a hand-crafted partitioner to create a custom decomposition but if the decomposition was created elsewhere it should in principle be possible for the mesh generator to define a mesh based on the grid decomposition that is passed to it.

Guillaume asked about the possibility of applying the interpolation in lat-lon space versus non-projected xy coordinates for some grids. Willem responded that it should be possible to do either provided that there was no data loss such as smoothing applied in the projection, and provided that there is a regular analytic mapping from x,y to lat-lon (which is not the case for cubed sphere). Dan also added that the fv3 developers are considering using the gridtools library that is now used optionally by atlas to improve efficiency in mesh operations.

Jake asked if Dan's work with the cubed sphere grid would eventually be incorporated into the Atlas code base. For now, the cubed sphere grid code will remain in JEDI, and folding it back into Atlas will be considered in the future.

Since we had exceeded our 1-hour meeting time by this point, the meeting was adjourned