

A Scalable Mesh and Field Data Source that is both Virtual and Tuneable

Mark C. Miller (LLNL)

Many aspects to HPC Exascale software productivity involve the same key software engineering issues of any large software project. One aspect that is somewhat unique to HPC exa-scale software productivity is the need to design, test and debug code running at the exa-scale. However, apart from having actual compute resources, a key pre-requisite for being able to run at exa-scale is having exa-scale sized data to work with.

All code teams inevitably face the problem of needing to test code and algorithms at scale and having no quality data to work with. They often invest resources to generate static file sets. But, this can be a large distraction from code development activities, the resultant file(s) are cumbersome to manage and work only for specific scale(s) and because the files typically cannot be shared between code teams the effort is often duplicated.

All code teams could benefit from having a scalable data source that is both virtual and tuneable.

By “virtual”, we mean the data never exists in file(s) anywhere. Instead it is generated on-the-fly when the application reads data from the scalable source. However, the application is completely unaware that it is interacting with anything other than the same kinds of files it is accustomed reading. The applications can continue to use the same I/O interfaces to “read” this data as they currently do to read real files. This is possible by building everything on top of a new kind of filesystem “device driver” (e.g. an alternative to ext3 or zfs) that effectively “spoofs” real files with on-the-fly data generation.

By tuneable, we mean the data can be generated in prescribed ways relevant to testing various aspects of codes at scale. Examples would be ensuring such things as the size of the mesh assigned to each MPI rank or OpenMP thread, whether the mesh is continuous, structured, unstructured or adaptive, or that the mesh has certain defects that mimic real world meshing issues or that a given field on the mesh has a prescribed gradient or boundary condition, or that the parallel decomposition and assignment to MPI ranks matches a given communication topology, etc.

Although meshing is a key capability, in this context the goal is to restrict the set of available models output by the scalable data source such that meshing part of the problem is substantially simplified. For example, maybe the initial implementation

serves up only rectangular, spherical or cylindrical models. Nonetheless, in the long term it may still be possible and useful to employ a meshing code (PMesh may be a good candidate) to drive the generation of more interesting physical models. To develop a scalable data source as described here, a few key technologies need to be combined together. These include

- Filesystem device driver(s): One or more device drivers for each unique operating system to be supported.
- Simplified Mesher: For restricted set of input models (initially just cubic, spherical and cylindrical) with later option to support more sophisticated models with a full-blown meshers such as PMesh or MeshKit.
- Tuning logic: additional logic to ensure prescribed tuning parameters of the generated output are met
- Tuning Targets Inventory: a survey of various HPC exa-scale codes for tuneable aspects of mesh and field models necessary to drive codes into particular testing regimes.
- Byte-Stream Plugins: File byte-stream generator plugins, one for each I/O interface to be supported, that can emit byte-streams according to the layout requirements of the I/O interfaces such as HDF5, Silo, netCDF or MOAB.

Note that the a scalable data source as described here has application and implications beyond that of supporting testing of codes at scale.

Such a tool can help spur the development of benchmarks and inter-code performance comparisons because it enables the identical data to be input to multiple codes. It helps developers measure and compare code performance without having it unnecessarily skewed by I/O activities. In addition, it can even help in the testing and benchmarking of real file I/O activities by using the data generated by the scalable source as an input for driving I/O interfaces and hardware.