Data and Workflow Solutions for Fusion using NERSC

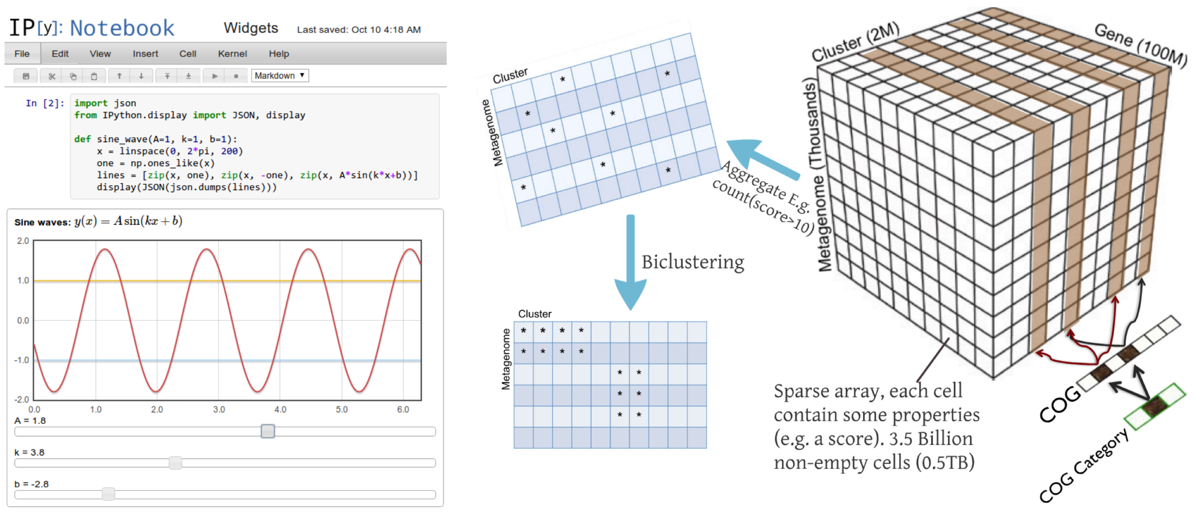
Alice Koniges, Shreyas Cholia, Prabhat, Yushu Yao
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
Corresponding Author: aekoniges@lbl.gov

In this white paper, we advocate three elements; the NERSC data partition, a connection of workflow elements using open source tools such as an IPython Notebook, and the Scientific Data Base (SciDB) infrastructure. A majority of the fusion scientists in the US, and a large component from outside the US use the NERSC as their primary compute platform. The center (originally started as a fusion-only centralized computing facility) is making several new features and hardware available to users. Users of NERSC benefit from the scale of the facility, since the needs of a wide range of scientific users are fairly similar when it comes to data and workflows. NERSC funds full-time computer scientists who develop and maintain essential software and infrastructure. It is critical that FES scientists partner closely with NERSC to assure that the developing systems address the needs of fusion community. The next generation NERSC machine, Cori, which is starting to be assembled, is primed to support the increasingly data-intensive computing needs of NERSC users. Critical to the data performance of Cori is Cray DataWarp technology, which accelerates application I/O and addresses the growing performance gap between compute engine and storage. This technology, or “Burst Buffer,” is a layer of NVRAM that serves to move data quickly between processor and disk. Cori will include a number of advanced features designed to accelerate data-intensive applications:

- Large number of login/interactive nodes to support applications with advanced workflows
- Immediate access queues for jobs requiring real-time data ingestion or analysis
- Connectivity that allows compute nodes to interact with external databases and workflow controllers

For a scientific workflow platform, one possibility is a “glue” based on the IPython Notebook. This gives a full generality to the workflow and allows one to plug and play different workflow components and target resources. The IPython Notebook was designed to enable researchers to move fluidly between all the phases of the research life-cycle and has gained rapid adoption. It provides an integrated environment for all computation, without locking scientists into a specific tool or format: Notebooks can always be exported into regular scripts and IPython supports the execution of code in other languages such as R, Octave, bash, etc. The IPython Notebook Server is an application that runs a web server to which the user’s browser connects. The web browser is the user interface where code is executed and the notebook document is edited. IPython was created as a system for interactive and parallel computing that is the *de facto* environment for scientific Python. The IPython Notebook, a web-based *interactive computational notebook* can combine Linux system commands to download data, a scripting capability to provide input to massive multicore simulations, the ability to diagnose and visualize simulations in-situ, and many other workflow necessities in codes, into a single reusable document format (see see left image of Fig. 1). Notebook workflow files can be easily shared among users and can be used to find bottlenecks and optimization paths.

NERSC is planning on launching a pilot iPython notebook service (ipython.nersc.gov) to enable interactive access to NERSC systems. This will allow users to login to a service with their NERSC credentials and initiate a live iPython notebook session, with access to NERSC resources (like global filesystems, libraries and APIs). This will provide a powerful platform for performing analytics and interacting with data products from jobs at NERSC, while removing the burden of managing such a service from the end-users. A user would simply log into the web interface and interact with NERSC via the notebook.



```
In [2]: import json
        from IPython.display import JSON, display

        def sine_wave(A=1, k=1, b=1):
            x = linspace(0, 2*pi, 200)
            one = np.ones_like(x)
            lines = [zip(x, one), zip(x, -one), zip(x, A*sin(k*x+b))]
            display(JSON(json.dumps(lines)))
```

Sine waves: $y(x) = A\sin(kx + b)$

A = 1.8

k = 3.8

b = -2.8

Figure 1: On left is the typical Interactive widgets in the IPython Notebook. On right is a workflow for Meta-Genome Analysis.

The SciDB Pilot Project workflows are quite diverse and we give two examples below. Fusion workflows can benefit from seeing how different communities are starting to use these emerging technologies.

In genomics, the growth of sequencing using next generation sequencing technologies has led to the generation of enormous amounts of sequence per genome/metagenome study resulting in Terabyte-sized metagenomic datasets containing hundreds of millions to billions of genes each. Traditional gene-centric methods of comparative analysis of genomes and especially metagenomes, are no longer computationally sustainable. One sample workflow is illustrated in the right image of Figure 1. It consists of subselection, aggregation and machine learning algorithm (bi-clustering in this case) on very large dataset (TB). Throughout this workflow, Python is used to orchestrate the process, including the data injection into the database (SciDB), driving the subselection aggregation inside the database, piping the data into and out from the bi-clustering algorithm. Depending on the input condition, the complexity of this workflow can vary greatly. This provides an excellent opportunity to test the surrogate-based modeling.

As another example workflow, in astrophysics, large telescopes such as the Sloan Digital Sky Survey produces a large number (millions) of spectral data of astronomical objects; the data is on the order of terabytes. Scientists often need to sub-select the objects based on some condition (type, color, position, etc) and perform some analysis on the selected objects. This workflow consists of two major parts. First, the workflow includes sub-selecting the objects based on conditions. This step is normally done with the help of some database technology (in our case, we use SciDB) to cut through the feature space and reduce the number of interested objects to tens of thousands. Second, the workflow calculates aggregate values from the selected objects. These steps are normally I/O intensive because of the non-continuous pattern of the data reads. Similar to the first example, the complexity of this workflow can vary greatly depending on the input conditions. Optimal conditions can be predicted and tuned with surrogate-based modeling.

Despite the differences in actual codes used for modeling fusion devices and other experimental facilities, there are a number of similarities in the requirements. We feel that it is critical coordinate with other modeling programs. One specific example is to learn from and possibly partner with the accelerator community, where we are also trying to optimize the design and operation of a range of accelerators including short-pulse laser driven ones and produce real-time feedback. Optimal use of tools developed not just through the fusion community, but also through collaboration with other large-scale experimental facilities is key to economical code development and dependance on software that will be maintained and improved as computing technology improves.