Thoughts on Data Management Tools for Physics Simulation Codes

David Mikkelsen, PPPL, dmikkelsen@pppl.gov

This 'white paper' is directed to the sub-panel on "Data Management, Analysis and Assimilation", with responses to the query "*What approaches to data management, visualization or analysis would make you more productive, and improving the utility of data derived from modeling by better integrating it into the overall scientific processes in fusion energy?*"

Three types of data management are discussed below: archival storage of each calculation's provenance (data and code), a searchable database of code inputs and results, and a flexible system for saving and organizing user comments about runs (and post-processed results, figures, …). Several types of requirements affect the method of implementation and complexity of the software that could meet these needs; some issues would not be encountered early in a development path that starts with a basic set of requirements, but it is vital to keep longer range goals in mind when starting out so that large economies of scale may be achievable as the framework becomes a production system with many users. If the limitations prevent large-scale adoption the benefits may not repay the development cost.

Calculation provenance and archival storage

Methods are needed for permanently archiving all inputs and outputs; this should overcome loss of provenance arising from overwritten data or other input files by storing a copy of what was read and written by each calculation. Code provenance is also essential, especially for complicated frameworks like the Integrated Plasma Simulator, and this capability needs to include run-time scripts such as GYRO/TGYRO uses as well as the compiled-language source files and their 'make' files. The framework needs to also provide support for post-processors written in python, IDL, Matlab, … so these codes can store their inputs and outputs in the archives and the searchable database, too.

Searchable database for inputs and outputs

Efficiently finding calculations that meet specific criteria is another important capability. Database queries are an effective way to do this, and this is an important way to improve utilization of the results by collecting groups of related calculations. It is probably best to link the methods for database entry to the run provenance tools. It would be easy to automate the database entry by using namelist and other input data methods directly. The default values for input variables need to be saved, too; these may not be found in input files so the user code needs to report these directly to the archiving and database systems. This should be done after default setting, namelist reading, and processing of the input variables (some variables may be reset to ensure 'subsidiary' variables are consistent with 'primary' variable choices). Users could define additional lists of non-input variables to be sent to the database; using a namelist of output variables set up specifically for this purpose would be convenient. It is very desirable to include text comments described in the next paragraph in the searchable database, and to use a "controlled vocabulary" that will search for equivalent terms such as "plasma current", "Ip", "Iplasma", …

Run notes

Managing text comments describing the motivation for each calculation, the anticipated results, and comments about the results would be extremely useful. If the capabilities are comprehensive users will have strong reasons to use the system. It is

very desirable to have convenient (automatic?) methods that link calculations with experimental discharges or specific postulated plasma conditions (standard scenarios for future devices, for instance). Standard naming conditions exist at each facility for experimental discharges, but a system should be developed for theoretical or postulated plasma conditions. It is very desirable to have a meta-comment facility for comments about multiple runs, including a simple syntax for inserting (or quickly accessing) comments from individual calculations at user-defined places in the meta-comment text. Links to and rendering of figure files are important, especially in meta-comments.

User buy–in

A low entry-level learning barrier is esssential to achieve a significant user base and justify continued support of the framework. Basic archiving of inputs, outputs and comments must be easily accomplished without learning much syntax. Complex features can require more complex syntax, and be developed later. Sharing with other users is desirable, but this should be optional where use of the framework is not mandated by the user's employer. It should be easy to allow individuals, a 'group', or 'world' access to individual runs, groups of runs, or all runs.

Framework longevity

Users will invest a lot of effort on the comments, so these must not be vulnerable to loss in 10-20 years. Development of foundational software is expensive, risky and should be avoided. For instance, the BASIS system at LLNL is no longer actively supported at LLNL or NERSC; it used the local PDB format from the PACT project, both of which are also now inactive. The archival and database framework should allow easy extraction of data files and comments to enable user exit from the framework and to facilitate migration by developers to new underlying software structures.

Multiple platforms

Major codes are run from multiple institutions and by multiple users (even at one institution). Even a single user will run on multiple computer systems (even at a single instituion), and at multiple computer centers. *All of a single user's runs should be accessible from a single point of contact*; and queries across users of the same code are very useful, as are queries across codes for runs based on the same experimental discharge or postulated plasma conditions. The post-processing may be done on a computer that is different from the one that ran the major physics calculation.

Implementation of the data management software

It is simplest to use a repository for each user, but better to encompass all users of each major code. A central repository for each research institution or computer center would be more efficent, and it may be feasible to have national coverage from a single repository. The economies of scale will not be apparent until after the system is developed and has multiple beta users, but it is essential to have the ultimate scale of service in mind when designing the framework. It is important to integrate the code provenance tools with version control systems so code sources are properly included in the provenance; GIT, for instance, has a unique identifier that is sufficient to store with each run. It is probably best for users to integrate calls to a management library within their physics code and/or run scripts. It is less useful to base the system on a GUI requiring a lot of specifications for each run. Any GUI should be able to use a script or state file to automate the process of run creation.