Control Workflow Project Proposal

Christine Sweeney and Pat McCormick
October 14, 2015

Workflow for scientific computing (collection, generation and processing of numerical data) is defined as "sequencing and orchestrating operations, along with the attendant tasks related to these operations"[1]. For example, workflow can include moving data between processing stages. The specification of input data, control flow and output data is critical to many aspects of the scientific workflow. For the purposes of this document, we will call this the **control workflow**. It is a workflow that sits at a higher level than the workflow that exists within application **components**. By components we mean executables, packages or libraries that compose an entire scientific application. This proposal concerns researching and prototyping novel mechanisms for control workflows for new scientific application codes currently being developed. Ideas for this proposal are based on two meetings with Next Generation Code project leaders, David Daniel and Aimee Hungerford.

A preliminary investigation [2] of the control workflow for some LANL codes reveals a rich history of control workflow mechanisms that have evolved and improved throughout the years. Currently, an **input deck** is a common expression for the specification of input data and control for a given computational science experiment, and is often in the form of scripts. These scripts indicate inputs to the system, control of the application, and specification of outputs and output frequency. Developers use input decks to run smaller scientific codes or smaller subsets of scientific codes and do their own verification and validation. Production users typically run large scientific simulations, parameter studies, verification and validation, and regression testing. Both types of users are interested in increased scientist productivity and in provenance. Scientist productivity can be adversely affected when the control workflow process of specifying and recording the scientific experiment is cumbersome and not well integrated with the application components. Input deck scripts (and for some also the input files they generate) and the associated scientific input data currently serve as provenance regarding how applications were run and are used to be able to reproduce results, to compare runs and as a starting point for new experiments. Current input deck mechanisms such as InGen use Python scripts to generate one or more input files that are in formats custom to different applications. The application components read these input files at startup.

Control workflows for some applications at LANL have evolved based on legacy applications that have stand-alone executable components that parse their own custom-format input files. New codes being written now have the option of making their functionality available via libraries, which opens up many possibilities to improve and extend the control workflow for these applications. Input and control can be done via a program that makes calls to application libraries, without generating input files in custom formats for each application component. Furthermore, provenance for the experiment can be recorded in a flexible manner and can possibly even be domain-specific. Control workflow technology can additionally provide customized code to application components; this code can be run within the application to do specialized in situ visualization, analysis and/or provenance recording. Control workflow technology can also be instrumental in enabling compilation of codes to use compile-time constants so that the machine code can be optimized for a

particular experiment. There may be other control workflow possibilities that we are not yet aware of until these new application codes are designed and written.

We propose to study how control workflows are managed in existing scientific codes, study how that may differ in new codes being developed, identify the constraints of applications and users and design and build novel prototype control flow technology. We will investigate the idea of control workflow software written in a high-level programming language that is simple, yet has capabilities that are useful for research in this area. One language we plan to research is Lua [3]. Originally designed for gaming, this simple, compact language has many agile features that make it ideal for our research direction. Although lacking in the number of scientific support modules and user base (compared to Python), Lua is a promising research vehicle to prototype some of these control workflow designs. It can be embedded and is small enough to be contained in scientific application components. This can enable shipping Lua code to application components in order to run it within that component, even if the component was not written in Lua. Application components can embed LuaJIT , a version of Lua that does just-in-time (JIT) compilation to compile to machine code at runtime, even though it is an interpreted language. LuaJIT creates highly performant code [4] and is easy to embed due to its small footprint. Having a flexible control workflow software prototype can help determine what kinds of application component library interfaces are necessary for this kind of coordination. It can also allow for layering of control workflow functionality, such that the system can be easily configured for developers, who may have simpler control workflows, as well as for production users, who may want to create a suite of various coordinated control workflows.

Another benefit of novel control workflow technology is that it can address dynamic system behavior and science needs. It can allow for steering of analysis and visualization as well as adapting the computation execution to dynamic system conditions. For example, the workflow control software could dynamically add tracer particles in order to increase tracking of quantities, or switch from recomputing quantities to storing them if enough disk space and bandwidth is available. We will investigate analysis and/or visualization steering as examples of dynamic workflows. This will provide a basis for experimentation with dynamic workflows that might someday extend to science aspects, although science aspects are currently not in the scope of this proposal.

[1] Deelman, E. and Peterka, T. "ASCR Workshop on the Future of Scientific Workflows." Slides. April 2015. Report coming out soon.
[2] Conversations with Aimee Hungerford, David Daniel and Brian Jean.
[3] Lua website. www.lua.org
[4] Lua Benchmarks. https://attractivechaos.github.io/plb/