

Simplifying Scientific Workflow Creation and Management
Jay Lofstead, Sandia National Laboratories, gflfst@sand.gov

1 Introduction

Scientific Workflow Management Systems (WMS) currently offer support for connecting various processing components including simulations, analysis, and visualization tools. Each engine has a different technique for linking these components and managing the workflow execution. While these systems can work well, they require an expert user to make use effectively of them.

Two particular workflow aspects stand out as unsolved problems. First, a missing component in WMSs is standardized “glue” for linking individual components. No matter if the components use the file system or direct communication or something in between to communicate, independently developed components do not necessarily expect data organized in the same way. To compensate, workflow creators write specialized intermediate steps that convert the output of one step into acceptable input for the next step. These glue components are one-off scripts that can rarely be reused. Creating these glue components is one of the hurdles for WMS adoption. Having to create these additional components each time reduces the value gained by learning and using a standard engine. The amount of benefit for simpler to general workflows by using these engines does not exceed the threshold for encouraging use.

The second aspect is effective runtime management addressing rightsizing workflow components based on the data flow and system characteristics that may change over the workflow lifetime. WMSs should monitor the data flow status between components and, where possible, change resource allocations to minimize the end-to-end time. These techniques should also be able to manage either failures or cases where insufficient resources are available to address data flow needs. Some initial work in this area¹ shows promise, but it must be expanded, formalized, and incorporated into WMSs to have impact

2 Component Connections

Consider this example. The LAMMPS² molecular dynamics simulation generates an atom position list that can be examined to determine when features, such as a fracture, appear. In this crack detection workflow, LAMMPS runs forward generating data using a coarse timestep progression. There is no need to use a fine-grained timestep progression until just before the fracture appears. By using an analysis routine for in flight data processing to detect the fracture, LAMMPS can be restarted using the appropriate state just before the crack appears, but with the fine-grained timestep progression. To accomplish this workflow a few general concepts are necessary. First, the atoms list from each process should be *aggregated into a single list*. Second, these aggregated atom lists should be buffered so that they decouple LAMMPS from the analysis routine. These two “glue” operator types are common with many different scientific workflows. If we can make the aggregation and buffering into

¹J. Dayal and et. al, I/O containers: Managing the data analytics and visualization pipelines of high end codes, in HPDIC 13, pp. 20152024.

²<http://lammmps.sandia.gov/>

standard components that work across a variety of different workflow components, application scientists can assemble this or similar workflows simply by selecting the correct operator (e.g., aggregate) rather than having to write custom code for each workflow created.

Buffering can be a more interesting example. Consider how a buffer may be used within a workflow. More traditional scientific workflows would use buffering to queue output from one component for processing by the next downstream component. While this is useful in itself, big data and streaming applications suggest other interesting uses. For example, a buffer may be used as a sliding window on a stream allowing analysis generating insights about current stream properties.

While these two components are obvious, they are not the complete frequently used operator set. Additional candidate operators must be identified and investigations on how to make these sorts of generic operators possible is required. Truly generic operators that require no additional user input may be impossible, but with a little additional information from the end user, we should be able to offer significant savings for workflow creation time, reducing potential errors, and encouraging using these WMSs affording portability between platforms.

3 Runtime Management

Efficient resource allocation across all workflow components is a challenging task. Users make a best guess and hope it is sufficient to achieve the end-to-end throughput required. Unfortunately, dynamic system conditions, such as interconnect or file system contention, or dynamic simulation state, such as the time between outputs for fine-grained vs. coarse-grained timesteps, can require a resource rebalance to maintain workflow throughput. A link management system (LMS) should address these kinds of management considerations to ease the guesswork and trial-and-error required to ensure workflow throughput and offer additional opportunities for workflow management.

While this may seem straightforward, these kinds of management operations require more detailed component knowledge to operate properly. For example, increasing or decreasing a workflow component resource allocation may require restarting a component or simply adding or removing additional parallel instances. There are also input and output queue management issues that must be addressed. For example, when increasing resources for a component, will this increase throughput for a single entity or does it offer an opportunity to process additional data sets in parallel. If it is additional data sets, how are they sequenced to downstream consumers to ensure ordering is maintained? Is strict ordering even required? If a component is reduced, what should happen to the work queue currently assigned to that instance? Should it be interleaved, prepended, or appended to a companion work queue? Should an unrecoverable failure occur or insufficient resources be available for required processing how should the work queues be handled? Is this a fatal error or should the data be redirected to storage? Can some of the messages be dropped? If so, how to decide? All of these issues prompt a deeper investigation in how to think about managing the end-to-end requirements for scientific workflows.

Acknowledgement: Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.