Scientific Workflow Use Cases

August 21, 2019

Version 1.1

These use cases describe how researchers use workflow tools to automate complex research computing processes. There are at least two motivations for automation: first, to improve research productivity by freeing researchers from time-consuming tasks that can be automated; second, to make the research process reproducible, capturing and recording every step of the process so other researchers can inspect and validate the process behind the researcher's conclusions.

Research computing processes are extremely diverse. Consequently, there are a variety of workflow tools in use. [1][2] The public research computing community enables workflow tools by providing APIs for basic computing services, such as: job submission and management, data movement, and resource configuration and status information. Using these APIs, a workflow tool can automate the execution of a research computing process. Another important community function is coordinating the execution environments across resources: making them as similar as possible so individual research jobs can be run on a wider variety of computing resources.

Use case SW-01 describes the most important community features needed to automate research computing processes. Subsequent use cases may consider the requirements of specific workflow tools or classes of tools.

SW-01: Automate a project using a workflow manager

A **researcher** needs to automate a project to run on one or more community resources using a workflow manager so the project can be completed as quickly as possible. The project consists of a large set of application runs. We'll refer to these application runs as "jobs" and the entire set of jobs as the "project." (A project is also often known as an "ensemble," or a "workflow"). The workflow manager is an application that automates the execution of the project.

We assume the following things are true.

1. The researcher is registered with the community and belongs to a project that has received at least one allocation of community resources.
2. Any project allocations are based on an existing software application and reasonable estimates for the required resources. (CPU hours, RAM, storage, etc.)
3. The application is already installed and validated on the allocated resources. *(Use case HPC-01 describes the process of installing and validating an application on a community resource.)*
4. The researcher is using a workflow manager application to manage the project's jobs.
5. The workflow manager is configured to use the community's standard mechanisms for remote job submission (use case CAN-01), file transfer (use case CAN-02), resource information (use case CAN-08), and resource status information (use case CAN-07).

In most cases, the researcher wants to experience it as follows.

1. First, the researcher defines the jobs that need to be run.
2. Then, the researcher logs into a workflow manager using the researcher's community identity AND the researcher's identities in any other communities.
3. Then, the researcher enters the project allocation information, specifying which resources have been allocated and the allocation ID for each resource.
4. Then, the researcher enters the job descriptions into the workflow manager. At this point, the workflow manager processes the job descriptions, identifies any dependencies between the jobs, and constructs a plan for the ordering of job execution. The researcher may review this plan and iterate with the workflow manager if necessary.
5. Then, the researcher instructs the workflow manager to begin submitting jobs. At this point, the workflow manager begins monitoring the status of each resource and submitting jobs to appropriate resources following the plan from Step 3. This continues until all the project's jobs have completed. For each job:
    a. The workflow manager stages any required inputs to the resource.
    b. The workflow manager submits the job to the resource's queue.
    c. The workflow manager monitors the job status until the job completes.
    d. The workflow manager retrieves the job's exit code and gathers outputs as appropriate.
    e. If the exit code doesn't confirm successful completion of the job, the workflow manager uses diagnostic info to decide whether to resubmit the job to the same resource, resubmit to a different resource (if any are available), or alert the researcher to the issue.

6. While jobs are queued and/or running, the researcher monitors the project status using the workflow manager's status interface.
7. While jobs are queued and/or running, the researcher may pause execution, cancelling any queued or running jobs on resources and returning them to the workflow manager's set of pending jobs.
8. While jobs are queued and/or running, the researcher may cancel execution, cancelling any queued or running jobs on resources and emptying the workflow manager's set of pending jobs.
9. When all of the project's jobs have completed, the researcher uses the results to complete the researcher's research goals. The researcher will acknowledge the community, the workflow manager application, and each resource's service provider in publications resulting from the project, and will cooperate with community requests for help with reports, science stories, etc. based on the results of the project.

We'll take any solution, as long as the following are true.

1. The researcher can use the solution to process several million jobs per project.
2. In Steps 1 and 4, the workflow manager accepts job descriptions in one or more common formats, such as DRMAA (https://www.drmaa.org/) and equivalent.
3. In Steps 1 and 4, the researcher doesn't need to specify the resource on which each job will run. The workflow manager makes these decisions dynamically during Step 5 based on the allocation information provided in Step 3 and current resource status information.
4. In Step 2, the user experience for logging into the workflow manager using a community identity is as described in use case IDM-06 or IDM-07. (IDM-06 for web applications, IDM-07 for locally installed applications.)
5. In Step 2, the workflow manager can obtain credentials (as described in use case CAN-06) that allow: (1) input/output staging on each resource; (2) job submission and management on each resource; and (3) use of community information services.
6. In Step 3, the workflow manager can retrieve resource information for each allocated resource (see use case CAN-08).
7. In Step 5, the workflow manager can dynamically obtain status information from each resource as described in use case CAN-07. The status information is sufficient to enable the workflow manager to make appropriate decisions about where and when to submit each job to minimize the overall project runtime given the available resources. Ideally, this includes a queue wait time forecast for each resource with 95% confidence intervals that have width < 25% of the requested wall time.
8. In Steps 5a and 5d, the workflow manager can stage job inputs and outputs as described in use case CAN-02. The interface provides sufficient status information for the workflow manager to know when staging has completed successfully vs. completed with errors.
9. In Steps 5b, 5c, and 5d, the workflow manager can submit and manage jobs on each resource as described in use case CAN-01.
10. In Step 5e, the community's job submission interface provides sufficient job status information for the workflow manager to know when jobs have completed successfully vs. completed with errors. Error conditions are sufficiently clear that the workflow manager can decide whether to

resubmit to the same resource (transient resource issue), resubmit to another resource (inappropriate resource match), or flag the job for the researcher to fix (bad job description).

11. In Step 6, the researcher is able to determine the overall project status using the workflow manager in 5 minutes or less.

References

[1] Yu, Jia, and Rajkumar Buyya. "A taxonomy of workflow management systems for grid computing." Journal of Grid Computing 3.3-4 (2005): 171-200.

[2] Deelman, Ewa, et al. "Workflows and e-Science: An overview of workflow system features and capabilities." Future Generation Computer Systems 25.5 (2009): 528-540.

History

	Version	Date	Changes	Author
Entire Document	0.1	10/25/2012	First Version submitted to A&D	Marlon Pierce, Ravi Madduri, Suresh Marru
	0.2	10/31/2012	Iterating first use case to capture baseline requirements.	Marlon Pierce, Ravi Madduri, Suresh Marru
	0.3/1.0	04/03/2013	Significant revisions to both use cases	Marlon Pierce, Ravi Madduri, Suresh Marru
Entire Document	1.1	08/21/2019	Reformatted to XSEDE-2 use case format; added references to XSEDE enabling function use cases; removed unnecessary XSEDE terminology.	L. Liming