Overview

The ECP ExaWorks project was funded for a phase-I effort starting in August 2020 to show the feasibility of creating an extensible software development kit (SDK) for developing exascale workflows. Part of the Phase-I charge was to conduct a survey of ECP teams to understand their workflow requirements and challenges. The survey was conducted in two parts: a short online survey and targeted deep-dive interviews with a subset of teams to assess project needs.

This report details the survey questions, summarizes the responses, and compares them with two other workflow surveys: the workflow survey conducted in the first year of ECP, and a survey organized by WorkflowsRI, an NSF-funded project focused on investigating community-based research infrastructure.

Prior Surveys

ECP Workflow Survey June 2017: the first workflow surveyed carried out across the applications teams in ECP occurred between March and April 2017. The aim was to assess workflow needs of application teams across the project. The survey received 13 responses out of the 22 application projects that the survey was sent to. The interpretation of the survey lead to the following high level findings:

- Half the responses intended to combine multiple applications into workflows
- There were a wide-range of task sizes (number of processors) and durations. While most were longer (>1 hour), there were 5 projects that had many short tasks which would be challenging to scale without new software support
- Most responses indicated that a combination of MPI and files would be used to orchestrate their workflows, indicating that an ad-hoc or custom designed infrastructure would likely be used in many cases
- When asked about workflow systems and workflow managers, only 1 project listed an existing solution. Most either did not plan to use a pre-existing workflow solution, or had not reached a stage of planning in their project where a solution had been selected
- Most teams (~10 out of 13) indicated Python as the main language they planned to use for defining workflows, with the Unix command line as the preferred user interface for invoking them

The assessment of the ECP ExaWorks team is that at the time the survey was conducted, most teams were at early stages of physics and model development, and had not begun to tackle their longer term workflow requirements, preferring to rely on existing methods (Python and shell scripts) to compose workflows.

Survey Structure

The ExaWorks survey was conducted in two phases: first a questionnaire was sent to 24 ECP applications teams and the 5 co-design centers. Out of the 15 responses we identified five teams to interview in more depth. Our selection criteria emphasized teams that were performing complex workflows and that had either written or were leveraging workflow management tools. Our goal with these interviews was to broaden our understanding of these workflows and the toolsets employed by these teams. We note that ExaWorks SDK technologies are currently being leveraged by CANDLE, ExaLearn, and ExaSky, so the interviews with the selected teams were crucial in understanding additional workflows of importance to ECP.

Responses and Interviews Summarized

Figure 1: word cloud showing the key-terms received from respondents, indicating a large usage of unix scripts and ad-hoc tools.

Responses to the survey indicate that many ECP application teams are orchestrating workflows using homegrown scripts (shell, Python, Perl) and tools like Make. Some teams reported usage of workflow tools: Airflow, Cheetah, Fireworks, libEnsemble, Merlin, Nexus, Parsl, and Savannah. Note, we allowed respondents to define ‘workflow tool’ broadly, and this results in a mixture of general workflow tools and tools under development for particular sub-domains in HPC.   We will summarize the initial survey results by each question, and then go into some detail on the use-cases and learnings obtained from the in-depth team interviews.

Workflow motifs represented in the reponses:

- **Single simulations:** often scaling to extreme scale single simulations
- **Ensembles:** by far the most common motif reported was generating ensembles of runs, typically via statically defined parameter studies, parameter sweeps and convergence studies
- **Analysis:** experiment-driven workflows which involve the instantiation of a mixture of short/small jobs and larger analysis jobs.

- **Machine Learning / Dynamic:** in these workflows the set of simulations that will be run is not known at the time the work is submitted to compute resources. These are the most advanced workflows, and the most likely to utilize customized or domain-specific infrastructures.

Internal Orchestration:
This question was aimed at understanding the need for tasks in a workflow to interact with each other and also whether tasks in a single job allocation might need to interact. Overall, responses indicated very little usage of orchestration among respondents. One team does utilize streaming/service oriented workflows where task to task interaction was required.

External Orchestration:
This question was aimed at understanding the extent to which teams utilized multiple machines, or executed workflows across multiple machines. The responses were evenly divided, with about half of the respondents indicating that their workflows span systems or that they would run them in that mode if they had a workflow tool that makes it possible to do so. In most cases, the usage of multiple systems was driven by the need to access as much resource as possible and to reduce the overall time for a given workflow, rather than a differentiation based on hardware or data locality. It is also the case that some teams have ensembles that span multiple job time limits, and so must submit several batch jobs to complete a single ensemble of simulations, and they classed this as a case of external orchestration.

Homogeneous vs. Heterogeneous tasks:
In general, most respondents indicated a large dynamic range of job sizes. Reasons for this range include: scaling/convergence studies, simulation vs. analysis jobs, inclusion of ML and simulation tasks. In general, we find that the more complex and dynamic the workflow, the more likely it is that teams will report high levels of task heterogeneity.

Key take-aways from the Interviews:

Supporting complex dynamic workflows across multiple machines, and porting to new machines is expensive in terms of developer time. Each main, even those that outwardly appear similar (e.g. Linux OS, Slurm batch scheduler, etc.), require customization in the workflow scripts and projects that needed to run at multiple facilities have developed abstraction layers to support these customizations. A key take-away is that attacking the lower layers of the workflow management stack can bring increased portability and reduce porting costs for teams, which lead to ExaWorks focusing on the J/PSI portability layer for schedulers.

Related to this is that robustness is both a pain-point and oftentimes a determining factor in whether a team will adopt a third party workflow technology or create their own bespoke capability. Accordingly, ExaWorks identified a well resourced effort to build an SDK with widely deployed testing via continuous integration technologies as a key enabler for the adoption of third party workflow technologies by application teams.