A common workflow registry of compute endpoints and applications

Daniel S. Katz (UIUC), Rosa M. Badia (BSC), Kyle Chard (ANL/UC), Jorge Ejarque (BSC)

11th JLESC (virtual) Workshop
10 September 2020

Thousands of workflow systems

- There are thousands or more workflow systems today
  - Circa 2010, Jennifer Schopf estimated NSF had supported ~1000 systems
  - CWL list (https://s.apache.org/existing-workflow-systems) currently shows 286
  - Awesome Pipeline (https://github.com/pditommaso/awesome-pipeline) has about 200
- Including ours
  - Parsl (https://parsl-project.org) and PyCOMPSs/COMPSs (http://compss.bsc.es)
- Why?
  - "I'll just write some code to do what I want, rather than looking to see if any existing tools can solve my problem"
  - "Those systems are all too complicated; I just need to do something simple" … "now I need to add some more features" … "now that I've done all this work, maybe someone else will want to use my system"
  - "The funding agency won't fund me to use someone else's system, but I can argue that my problem is different, I need to be funded to develop a new system for it"

Workflow systems help users address common problems

- High-level programming, using existing components, tools, libraries
  - Set up parameter sweeps & campaigns
  - Assemble data analysis pipelines
- Organize data
  - Data from one task that needs to be used by another
  - Output data from multiple tasks
- Efficiently use computational resources
  - Typically local system, CPUs, GPUs, clusters, HPC, cloud, edge, …
  - Ideally separate program logic from resource logic

Workflow systems themselves have common problems

- High-level programming, using existing components, tools, libraries
  - Set up parameter sweeps & campaigns
  - Assemble data analysis pipelines
  - How to address these components, tools, libraries?
- Organize data
  - Data from one task that needs to be used by another
  - Output data from multiple tasks
  - Mostly internal to the workflow system
- Efficiently use computational resources
  - Typically local system, CPUs, GPUs, clusters, HPC, cloud, edge, …
  - Ideally separate program logic from resource logic
  - How to address these resources?

Common problems

- How to address application components, tools, libraries?
  - Inputs, outputs, arguments, how to run
- How to address resources?
  - Access, authentication/authorization, queue system
- Today, workflow systems address these uniquely
- And then ask the user to find info about the apps and resources, then translate this into the right terms for the workflow system
- The user also must keep this info and the translation up to date

Proposed: a registry as a common solution

A registry of compute end points and applications, where an entry could be automatically brought in to a workflow system

Requires:

1. The registry itself
   - Including a means for adding and editing entries, potentially along with curation, or perhaps community curated, using WikiData
2. A means to use entries for a given workflow system

Who adds & maintains entries

1. Resource providers
   - Compute resource providers enter their systems
   - Application developers enter their applications
2. Workflow system providers
   - Enter systems and applications that they support
3. Workflow developers (users & those building workflows for others)
   - Enter system and applications that they & their workflow users need

Maintenance/curation to be determined

Initial work

- Define the schema for the registry, and implement it as a REST service
- Build some test elements, and enter them manually
- Build software for Parsl and PyCOMPSs/COMPSs to import and use registry entries
- Originally planned (pre-COVID-19) for summer 2020
- Now
    - Focus initially on defining the registration schema & collecting resource documents in a GitHub repository during 2020
    - Early 2021, gain experience with using them in an ad hoc fashion
    - Summer 2021, students will develop the REST API and integrate with Parsl
    - 2021, consider use of registry & integration with PyCOMPSs within EU eFlows4HPC project
    - Then evaluate progress & next steps: likely publish & bring in more workflow systems

Open questions & collaboration

Open questions:

- How to best represent computational end points and applications
- How to best build a registry of these elements, and curate the information in it
- How to bring this information into workflows specific to different workflow systems

Looking for collaborators:

- Application developers whose applications are used in multiple workflow systems
- Owners/managers of HPC systems that are used by multiple workflow systems
- Information science/systems experts
- Workflow system developers