Patterns and Anti-Patterns in Migrating from Legacy Workflows to Workflow Management Systems

Daniela Cassol, Jeff Froula, Edward Kirton, Seung-Jin Sul, Mario Melara, Ramani Kothadia, Elais Player, Setareh Sarrafan, Stephen Chan, Kjiersten Fagnan

Joint Genome Institute
Lawrence Berkeley National Lab
Background

• Who are we?
  • DOE Joint Genome Institute ([https://jgi.doe.gov/](https://jgi.doe.gov/)) - established as part of the Human Genome Project, provides sequencing and analysis services for scientists all over the world
  • Over 300 staff members
  • We use hundreds of thousands of CPU hours every year on computational biology workflows
  • Dozens (perhaps hundreds) of workflows, growing, evolving since the founding of JGI in the late 90’s
  • Moving towards a consensus architecture based on Workflow Description Language (WDL), Cromwell and Containers

• What is JAWS?
  • JGI Analysis Workflow Service (JAWS) is a geographical distributed workflow execution platform
  • Unifying workflow across JGI Groups
  • Uses Cromwell to execute workflows in a common Workflow Description Language (WDL) with Globus file transport to run computational workflows across multiple HPC facilities.
Pattern: Build A Community Around New Standard

- Work to establish a community that shares knowledge and practices for the new platform
  - **Hackathons**: Facilitate meetups and collaborations
    - Invite newcomers and partner them up with experienced members
  - **Code Sharing**
    - Create a shared repository for the organization for workflows, sub-workflows, and containers
  - **Documentation**
    - Provide a space where users can share and discuss experiences. This facilitates the exchange of insights on finding and modifying solutions

- Create online communities
  - Slack
  - Mailing lists
  - Try to keep the community “flat” and avoid having it all driven by an individual or tiny cohort
    - Have members of the community feel a sense of common ownership

- **Anti-Pattern**: Superheroes who do all the work
  - Not sustainable in the long run
  - Leaves many people as consumers/observers instead of being producers and having agency
Anti-Pattern: Inappropriate Parallelism

- Task parallelism involves distributing tasks across independent compute nodes, primarily when no data dependencies exist between tasks.
- Example of sub-sub-workflow:

  - Sub-workflow Modular
  - Reuse Containers
  - Parallelism
Anti-Pattern: Inappropriate Parallelism

- Task parallelism involves distributing tasks across independent compute nodes, primarily when no data dependencies exist between tasks.
- Example of sub-sub-workflow:

  - Execution time: less than one minute/tasks
  - I/O filesystem overhead
  - > 17,000 tasks
Anti-Pattern: Inappropriate Parallelism

- Task parallelism involves distributing tasks across independent compute nodes, primarily when no data dependencies exist between tasks.

Example of sub-sub-workflow:

- Execution time: less than one minute/tasks
- I/O filesystem overhead
- > 17,000 tasks

-71% shards/tasks
-73% execution → Reduce I/O filesystem overhead
Comparisons

- **Porting Legacy workflows to WDL - Execution time:**

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Legacy workflow</th>
<th>Using JAWS/WDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Reference Database (450M genes)</td>
<td>13 hrs</td>
<td>6 hrs (using large memory single node)</td>
</tr>
<tr>
<td>Horizontal Transfer (5M new genes)</td>
<td>2 hrs</td>
<td>1.3 hrs (single thread)</td>
</tr>
<tr>
<td>Phylogenetic Distribution (5M new genes)</td>
<td>6.5 hrs</td>
<td>1.3 hrs (using 10 shards)</td>
</tr>
</tbody>
</table>

- **Improving Existing Workflows - Using /tmp for some I/O intensive tasks:**

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Pass Before</th>
<th>Pass After</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAP-seq</td>
<td>76%</td>
<td>99%</td>
</tr>
</tbody>
</table>
Summary

Strategies for Effective Migration to Workflow Management Systems

- **Modularization & Scalability**: Simplify and enhance efficiency
- **Containerization**: Promote consistency across different environments
- **Parallelism**: Balance between task execution and overhead
- **Complex Workflow Migration**: Favor phased, systematic approach
- **Performance Metrics Collection**: Optimize resource use
- **Version Control**: Utilize robust systems like Git
- **Documentation**: Facilitate community use and adaptation
- **Testing and Validation**: Essential for reliable transitions
Thank you!
Questions?
dcassol@lbl.gov