Bridges-2 Webinar

*Data-driven Computational Pipelines on Bridges-2*

Ivan Cao-Berg
Pittsburgh Supercomputing Center

June 26, 2023
The Biomedical Applications Group pursues cutting-edge research in high performance computing in the biomedical sciences. They foster collaboration between PSC experts in computational science and biomedical researchers nationwide.
Motivation

• FAIR data stands for **Findable, Accessible, Interoperable, and Reusable** data.

• Overall, FAIR data principles aim to improve the discoverability, accessibility, **interoperability**, and **reusability** of research data.

• By adhering to these principles, researchers and organizations can maximize the value and impact of their data.
What are workflows?

- In computational workflows, individual tasks or steps are organized in a logical order, where the output of one task serves as the input for the subsequent task.
- This allows for the creation of **reproducible** process that can be executed reliably.
- Workflows can be designed to handle a wide range of tasks, including **data processing**, **analysis**, simulation, modeling, and decision-making.
Workflow management systems

• Workflow Management Systems (WMS) are software tools or platforms that facilitate the design, execution, and management of workflows.

• Bioinformatics workflow management systems are specialized software tools or platforms designed for managing and automating bioinformatics workflows.

• These systems cater to the unique needs and challenges of bioinformatics research and analysis, which involve handling large-scale biological data, executing diverse computational tasks, and integrating various bioinformatics tools and resources.
Bioinformatics workflow management systems

- Galaxy
- Taverna
- **Nextflow**
- **Snakemake**
- Cromwell
- Pegasus
- Toil
- GenePattern

- BioBlend
- Terra
- SeqWare
- Kepler
- .... and more
Types of Computational Workflows

There are different types of computational workflows, including procedural workflows, data-driven workflows, and model-driven workflows

- **Procedural workflows.** These workflows follow a predefined sequence of steps or procedures.
- **Model-driven workflows.** These workflows incorporate mathematical or computational models as the core components.
- **Data-driven workflows.** These workflows focus on the flow and manipulation of data. They utilize data dependencies to determine the order in which tasks should be executed. *Data-driven workflows are common in data analysis and data exploration pipelines.*
Model driven-workflows example in Simulink

https://www.mathworks.com
# Procedure driven-workflows example using Airflow

## DAGs

### DAG Details

<table>
<thead>
<tr>
<th>DAG Name</th>
<th>Owner</th>
<th>Runs</th>
<th>Schedule</th>
<th>Last Run</th>
<th>Recent Tasks</th>
<th>Actions</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>example_bash_operator</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:08:11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_branch_dop_operator_v3</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-23, 14:09:17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_branch_operator</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:08:04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_complex</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:07:33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_external_task_marker_child</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:08:34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_external_task_marker_parent</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:08:34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_kubernetes_executor</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:07:40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_nested_branch_dag</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:07:37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>example_passing_params_via_test_command</td>
<td>airflow</td>
<td></td>
<td></td>
<td>2020-10-26, 21:07:37</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[https://airflow.apache.org](https://airflow.apache.org)
Data-driven workflows vs other types

• The main difference between data-driven workflows and other types of workflows lies in the primary focus and driving factor behind their execution.

• Data-driven workflows are primarily driven by the availability and characteristics of data.

• The tasks and operations within the workflow are often determined by the input data and the desired output.
Nextflow

• Nextflow supports various programming languages and uses a domain-specific language (DSL) for workflow creation and configuration.

• Key features include **scalability for large datasets**, reproducibility through tracking of data and software versions, and portability across computing environments.

• Nextflow leverages **container technologies** for easy packaging and deployment of software dependencies.

• It offers robust error handling, ensuring reliable execution of workflows.
Pros of using Nextflow

- Scalability.
- Reproducibility (within reason).
- Portability.
- Containerization. Nextflow integrates with container technologies like Docker and Singularity, facilitating the packaging and deployment of software dependencies.
- Error Handling.
- Active Community.
Cons of using Nextflow

- Learning Curve: Nextflow's domain-specific language (DSL) may require some learning for users unfamiliar with the syntax and concepts.
- Debugging Complexity.
- On Bridges-2 the setup for RM-shared requires additional configuration.
Nextflow community

https://nf-co.re/
Parts of a Nextflow workflow

- Main Workflow Script.
- Configuration File.
- Input Data Files: These are the input data files required by your workflow. Input files can be of any type, such as FASTA files, CSV files, or any other data format relevant to your workflow.
- Additional Scripts/Functions (Optional): Depending on the complexity of your workflow, you may include additional scripts or functions to perform custom operations, data processing, or other tasks.
- Output Directory (Optional).
Nextflow binary is available as a module.
Additional configuration is needed to run a workflow on Bridges2.
nf-core/rnaseq is a bioinformatics pipeline that can be used to analyse RNA sequencing data obtained from organisms with a reference genome and annotation. It takes a samplesheet and FASTQ files as input, performs quality control (QC), trimming and (pseudo-)alignment, and produces a gene expression matrix and extensive QC report.

(https://github.com/nf-core/rnaseq)
An example pulling a workflow from nf-core
Snakemake is a powerful and flexible workflow management system that simplifies the execution of complex data analysis pipelines.

Yet an alternative to Nextflow (or is Snakemake an alternative to Nextflow?).
Pros of using Snakemake

• Reproducibility.
• Scalability.
• Flexibility.
• Portability.
• Integration. Seamlessly integrates with various software tools, libraries, and computing clusters, providing a unified framework for managing and executing diverse analysis tasks.
• Community and Support.
Cons of using Snakemake

• Learning Curve.

• DSL Limitations. While Snakemake's domain-specific language (DSL) is powerful, it may have certain limitations or constraints when compared to more general-purpose programming languages.

• Complex Workflows.

• Debugging.

• Community Support.
Snakemake community

https://workflowhub.eu/
Snakemake can be installed using pip in your home directory.

(1) module load anaconda3
(2) pip install snakemake --user
Differences between Snakemake and Nextflow

• **Language and Syntax.** Nextflow is based on a custom domain-specific language (DSL) inspired by Groovy, while Snakemake uses a Python-like DSL.

• **Ease of Use.** Snakemake is often considered easier to learn and use, especially for users already familiar with Python.

• **Portability.** Nextflow places a strong emphasis on portability, enabling workflows to be executed on different computing platforms and containerization technologies (e.g., Docker and Singularity).

• **Community and Ecosystem.**
• Ultimately, the choice between Nextflow and Snakemake depends on various factors, including
  • user preferences,
  • familiarity with the programming languages used,
  • specific workflow requirements,
  • and the existing community and ecosystem support available for the intended application domain.
Common Workflow Language

- Common Workflow Language (CWL) is an open standard and specification for describing computational workflows in a portable and platform-agnostic manner.

- Some features
  - Declarative Syntax.
  - Inputs and Outputs. CWL allows the specification of input parameters, their types, and default values.
  - Validation and Verification. CWL provides validation mechanisms to ensure the correctness and integrity of workflow descriptions.
  - Tool and Process Descriptions.
  - Data and File Handling.
  - Portability and Interoperability.
Pros of using Common Workflow Language

- Portability.
- Reproducibility.
- Interoperability.
- **Flexibility.** CWL supports a wide range of programming languages and tools, allowing users to work with their preferred languages and tools within their workflows.
- Community and Ecosystem.
Cons of using Common Workflow Language

- Learning Curve.

- **Complexity.** As workflows become more complex, expressing intricate control flow or advanced programming logic within CWL can be challenging.

- Limited Tool Support. While CWL has gained popularity, not all existing tools and software have native support for CWL.

- Development and Maintenance. Creating and maintaining CWL workflows requires considerable effort and attention to detail.
cwltool can be installed using pip in your home directory.
If you have installed your own Conda distribution on Bridges2, you can alternatively run:

```bash
(base) → ~ conda install -c conda-forge cwltool
```
In another nutshell...

- Nextflow, Snakemake and CWL can be used to run workflows on Bridges2.
- There are many other popular workflow manager, including some that are cloud based, like Terra.
- The benefits of sharing your workflows, e.g. for a publication, outweighs the cons of using a system like this.
In another nutshell... (cont.)

Benchmarking a trivial example.

```
(base) → fortune hyperfine --warmup 10 'fortune fortunes' 'snakemake --cores 1 --printshellcmds' -i --export-json fortune.json

Benchmark 1: fortune fortunes  
  Time (mean ± o):  56.7 ms ± 1.2 ms  [User: 34.3 ms, System: 14.1 ms]  
  Range (min ... max):  55.0 ms ... 63.1 ms  46 runs

Benchmark 2: snakemake --cores 1 --printshellcmds  
  Time (mean ± o):  556.4 ms ± 13.8 ms  [User: 327.1 ms, System: 142.7 ms]  
  Range (min ... max):  547.8 ms ... 594.0 ms  10 runs

Summary  
fortune fortunes ran 9.81 ± 0.32 times faster than snakemake --cores 1 --printshellcmds
```

(Based) → fortune
Examples scripts for Bridges2

- A set of examples will be made available at this link
  - https://github.com/pscedu/workflow-examples