Optimising workflow lifecycle management: development, HPC-ready containers deployment and reproducibility

Workflow provenance hands-on guide

In this hands-on, we are using an adapted version of a workflow used in CAELESTIS for mechanical testing simulation which was developed by Riccardo Cecco (previous member of the Workflows and Distributed Computing Group, BSC) with the guidance of Gerard Guillamet (Dual Technologies Research Group, BSC) and Aravind Sasikumar (AMADE Research UDG). The workflow is distributed with a singularity container described by Fernando Vazquez (Workflows and Distributed Computing Group, BSC) and created with the Image Creation Service developed by Jorge Ejarque (previous member of the Workflows and Distributed Computing Group, BSC) at eFlows4HPC project. This link[1] provides the instructions about how to create a container with the Container Image Creation tool for an HPC cluster.

The source of the workflow can be found in this link[2]. The main code of the workflow is implemented in *workflow.py*. The workflow has 2 parameters as input:

- a yaml file that configures the workflow indicating the problem to solve (variables to sample, mesh of the object to test and AI models and parameters to analyse)
- the execution directory.

The workflow has three parts:

1. A sampling part which receives the problem definition and generates the Design of Experiments (DoE) matrix (sampling.py file)
2. The simulation part that for each row of the DoE matrix launches a simulation (simulation.py file)
3. An AI part which performs a model selection using the Grid Search algorithm from the dislib and the Decision Tree Regressor from sk-learn (ai.py file). The source code of the Grid Search can be found here and the internal PyCOMPSs tasks (fit_sklearn_estimator and score_sklearn_estimator) that are invoked can be seen here.

The other files contain auxiliary functions that are not relevant for the hands-on.

[1] https://github.com/eflows4hpc/image_creation/blob/main/README.md#instructions-to-run-container-image-creation-on-your-local-computer

[2] https://github.com/eflows4hpc/workflow-registry/tree/main/tutorial/HPC_AI_training/src

Exercise 1: Generate workflow provenance and inspect it

Detailed help for this exercise is available at:
Step 1: https://compss-doc.readthedocs.io/en/latest/Sections/05_Tools/04_Workflow_Provenance.html#yaml-configuration-file
Step 2:
https://compss-doc.readthedocs.io/en/latest/Sections/05_Tools/04_Workflow_Provenance.html#recording-activation
Step 3: https://compss-doc.readthedocs.io/en/latest/Sections/05_Tools/04_Workflow_Provenance.html#resulting-crate

1. Select the parameters of your run:
    a. Select the model between:
        i. test.yaml: sklearn.tree.DecisionTreeRegressor
        ii. test_SVR.yaml: sklearn.svm.SVR
    b. Select between these suggested configurations (8 samples):
        i. Submit using 2 nodes, ALYA_PROCS = 28 (4 Alyas per node)
        ii. Submit using 4 nodes, ALYA_PROCS = 56 (2 Alyas per node)
2. Modify provenance YAML PROV_SC24.yaml:
    a. Improve the ‘name’ and ‘description’ to better describe your experiment
    b. Establish yourself as an Agent:
        i. If you don’t have an ORCID, just remove the ‘Agent’ section, the first Author will be considered the Agent. Or leave the file unchanged.
        ii. data_persistence must remain False (avoid saturating disk)
3. Submit your experiment activating provenance generation:
    a. Use run_prov.sh, that includes the --provenance=PROV_SC24.yaml flag

```
mn5$ source load_compss.sh
mn5$ sh run_prov.sh [test.yaml, test_SVR.yaml] [2, 3, 4]
```

4. The execution can take between 10 - 20 minutes, depending on the configuration
5. While your execution finishes, we inspect together existing RO-Crates from WorkflowHub (Exercise 3).
6. Once the run finishes, inspect the generated COMPSs_RO-Crate_[uuid]/ folder
    a. Using pycompss inspect

```
mn5$ pycompss inspect COMPSs_RO-Crate_8ecdcd9e-a0de-425d-9b88-c113e94c9cac/
```

    b. Also, check the content of the RO-Crate folder:

```
mn5$ cd COMPSs_RO-Crate_8ecdcd9e-a0de-425d-9b88-c113e94c9cac/
```

Exercise 2: Publish a workflow run in WorkflowHub

Detailed help for this exercise is available at:
https://compss-doc.readthedocs.io/en/latest/Sections/05_Tools/04_Workflow_Provenance.html#publish-and-cite-your-results-with-workflowhub

1. Package the result

```
mn5$ zip -r my_app_crate.zip COMPSs_RO-Crate_8ecdcd9e-a0de-425d-9b88-c113e94c9cac/
```

```
laptop$ scp nct01XXX@transfer1.bsc.es:~/CAELESTIS/my_app_crate.zip ~/Desktop/
```

2. Upload to WorkflowHub
    a. Go to WorkflowHub -> Contribute (https://workflowhub.eu/workflows/new)
        i. Upload/Import Workflow RO-Crate -> Select local file (my_app_crate.zip), click “Register”
        ii. Briefly inspect imported metadata
        iii. Select Team “COMPSs Tutorials”, check “Sharing permissions”, click “Register”

Exercise 3: Inspect a previous published execution

Detailed help for this exercise is available at:
https://compss-doc.readthedocs.io/en/latest/Sections/05_Tools/04_Workflow_Provenance.html#resulting-crate

1. Find your own published workflow
    a. My Items -> Workflows
2. Or browse for other COMPSs Workflows at WorkflowHub
    a. Browse -> Workflows
    b. Workflow type: filter by “COMPSs”
    c. Team: filter by “COMPSs Tutorials” (or don’t)

3. **(OPTIONAL):** inspect the metadata in detail
    a. Check *compss_submission_command_line.txt* (main app used, parameters passed)
    b. Check *App_Profile.json* (number of nodes, statistics per node, overall statistics)
    c. Check time and result of provenance generation (*compss_XXXX.out*)
        i. What was the longest time during provenance generation?
    d. Can you understand the metadata (*ro-crate-metadata.json*)?
        i. Identify the 3 main parts of the JSON: Root Data Entity, Data Entities, Contextual Entities
        ii. ro-crate-metadata.json interesting keywords to search for: *mainEntity, ComputationalWorkflow, WorkflowSketch, #compss, CreateAction (object, result)*
        iii. Observe the *CreateAction* in detail

4. Questionnaire to be answered:
    a. Who ran this code? Where? When? With which COMPSs version?
    b. What is the name of the main application source file?
    c. What were the inputs and outputs used or generated in this workflow run?
    d. Can you say how many cluster nodes were used for the run?
    e. Where can you find detailed profiling of the application?
    f. Are the data assets included in the package?
    g. What was the command used to run this workflow? What were the parameters passed to the application?
    h. Navigate the workflow diagram

Your answers:

Exercise 4: Reproduce an execution from an application published in WorkflowHub

1. Browse for a COMPSs Workflow at WorkflowHub
   a. Browse -> Workflows
   b. Workflow type: filter by "COMPSs"
2. Download RO-Crate from WorkflowHub (example: https://workflowhub.eu/workflows/1197)
   a. Avoid the CAELESTIS examples just uploaded, since the Reproducibility Service is not yet compatible with containers
3. Copy the crate from your laptop to MN5

```
laptop$ scp -r ~/Desktop/workflow-1197-1/ nct01XXX@transfer1.bsc.es:.
```

4. Run the COMPSs Reproducibility Service (RS) (beta version) with the copied crate

```
mn5$ cd
mn5$ source CAELESTIS/load_RS.sh
mn5$ compss_reproducibility_service workflow-1197-1/
```

5. You may have to answer if you want to generate the provenance of your new run (y/n)
   a. If yes: you need to provide Agent details.
6. See how the RS checks:
   a. With data_persistence: if all input files are included, and their *Size*
   b. Without data_persistence: if the *Modification Date* and *Size* of the input files match the ones in the metadata
7. When asked to "add more flags" say "y":
   a. Add your submission flags: --project_name=nct_312 --qos=gp_training --reservation=tutorial-sc24
8. The application is submitted to the queue
9. The execution directory is **reproducibility_service_[timestamp]/**

Exercise 5 (OPTIONAL): Reproduce MANUALLY an execution from an application published in WorkflowHub

1. Browse for a COMPSs Workflow at WorkflowHub
    a. Browse -> Workflows
    b. Workflow type: filter by “COMPSs”
2. Download RO-Crate from WorkflowHub, some examples:
    a. PyCOMPSs: Matrix multiplication with data persistence: https://doi.org/10.48546/workflowhub.workflow.838.1
    b. PyCOMPSs: Matrix multiplication without data persistence: https://doi.org/10.48546/workflowhub.workflow.839.1
    c. Java COMPSs Matrix Multiplication, out-of-core using files, reproducible example, data persistence True: https://doi.org/10.48546/workflowhub.workflow.1086.1
    d. Java COMPSs Matrix Multiplication, out-of-core using files, reproducible example, data persistence False, MareNostrum V: https://doi.org/10.48546/workflowhub.workflow.1088.1
3. Copy the crate from your laptop to MN5

```
laptop$ scp -r ~/Desktop/workflow-839-1/ nct01XXX@transfer1.bsc.es:.
```

4. Follow the instructions to re-execute the workflow manually: https://compss-doc.readthedocs.io/en/latest/Sections/05_Tools/04_Workflow_Provenance.html#re-execute-a-compss-workflow-published-in-workflowhub

Exercise 6 (OPTIONAL): Repeat all previous exercises with data_persistence=True

1. Set data_persistence to True in the corresponding YAML file (i.e.: PROV_SC24.yaml)
2. Re-execute your application
3. Publish your new run to WorkflowHub
4. Compare the runs
    a. Is there a dataset/ folder???
    b. ro-crate-metadata.json -> check how data assets are referenced now
        i. Search for a specific file you know
        ii. Or look for “CreateAction”

Exercise 7 (OPTIONAL): Repeat all previous exercises with Lysozyme in Water application

1. Code available at: /gpfs/scratch/nct_312/Tutorial_SC24/lysozyme_in_water/
2. Copy the code to your home directory

```
$ cp -r /gpfs/scratch/nct_312/Tutorial_SC24/lysozyme_in_water/ ~
```

3. You can run different versions of the code:

```
$ ./launch.sh 2 10 false config/ dataset_small/ output/
```

```
$ ./launch_full.sh 2 10 false config/ dataset_small/ output/
```

```
$ ./launch_full_no_mpi.sh 2 10 false config/ dataset_small/ output/
```

4. You can change the number of workers (instead of the above specified 2, use more)
5. You can run with a larger dataset

```
$ ./launch_full.sh 2 10 false config/ dataset/ output/
```

6. In your runs, play also with data_persistence True and False in the provenance YAML file.