IRIS-HEP proposal

Snakemake workflow engine for HEP analyses

Fellow: Kyrylo Filonenko

Mentors: Clemens Lange, Katharina Schaeuble, Matthew Feickert

1 Motivation

HEP analyses run as multi-stage pipelines across Slurm, HTCondor, grid, and cloud resources, and several different workflow engines are already in use across IRIS-HEP groups. Law, built on Luigi, encodes an analysis as Python task/target classes and underlies production CMS analyses [2]. yadage instead encodes a workflow declaratively in YAML, linking parametrized processing steps through a dynamic DAG [3, 4]. Snakemake takes a different approach: rules declare input and output files, and the dependency DAG is inferred automatically [1]. Since v8 it ships executor plugins that allow Snakemake workflows to run on computing clusters commonly used in HEP, such as Slurm and HTCondor [5]. This project ports an existing workflow engine based analyses to Snakemake, compares the resulting user experience, and validates distributed execution through the Slurm and HTCondor executor plugins.

2 Workflow engines in HEP

law structures an analysis as a hierarchy of task classes; each task declares parameters, required upstream tasks, and output targets, and is complete once those targets exist. luigi resolves the DAG, while law adds remote storage, HTCondor/Slurm/LSF submission, and Docker/Singularity sandboxing [2].

Yadage separates a workflow's stage topology from the parametrized "packtivities" it invokes, producing a dynamic DAG that is executed by the yadage reference implementation [3,4]; this backend has, for example, scheduled containerized jobs on a Kubernetes cluster for the RECAST re-interpretation framework and the CERN Analysis Preservation Portal [3].

Snakefile defines rules with shell or Python bodies, wildcards generalize rules over many files, and Snakemake builds the DAG and manages per-rule conda/container environments [1]. Cluster execution is delegated to executor plugins selected with a single executor flag [5].

3 Project objectives and tasks

Port existing enginge-based analysis workflows to Snakemake, compare the resulting user experience between the engines, and validate distributed execution through Snakemake's Slurm and HTCondor executor plugins. The specific tasks are:

1. Learn Snakemake (rules, wildcards, environments) and break down the reference law workflow's tasks, targets, and dependencies.
2. Reimplement each stage of the example workflow as Snakemake rules with identical inputs, outputs, and physics logic.
3. Configure and test the Slurm and HTCondor executor plugins on real HTC/HPC resources and the REANA platform.
4. Compare workflow engines against criteria that will be defined.

4 Timeline

Start date: 2026-07-06

- **Week 1–2:** Learn Snakemake basics; study the reference law workflow's structure.
- **Week 3–4:** Design the mapping from law and Yadage tasks/targets to Snakemake rules and wildcards.
- **Week 5–7:** Implement the Snakemake workflow; set up Slurm and HTCondor execution; check output equivalence with existing workflows.
- **Week 8–9:** Compare workflow engines on the selected criteria.
- **Week 10–11:** Benchmark implementations under realistic workloads and document results as a MyST Markdown website.
- **Week 12:** Write the final report and presentation.

References

[1] F. Mölder et al., Sustainable data analysis with Snakemake, F1000Research 10, 33 (2021). DOI:10.12688/f1000research.29032.1

[2] M. Rieger, End-to-End Analysis Automation over Distributed Resources with Luigi Analysis Workflows, EPJ Web Conf. 295, 05012 (2024). arXiv:2402.17949

[3] K. Cranmer and L. Heinrich, Yadage and Packtivity – analysis preservation using parametrized workflows, J. Phys. Conf. Ser. 898, 102019 (2017). arXiv:1706.01878

[4] yadage software repository, https://github.com/yadage/yadage

[5] Snakemake documentation, Using executor plugins, https://snakemake.readthedocs.io/en/stable/executing/executors.html