Workshop on workflow languages for HEP analysis (3-5 April 2024)

Day 1, Wednesday 3rd April

Workshop introduction

Presenter(s): Matthew Feickert

- Focus of workshop on engaging both communities and facilitating discussion
- How do WLs fit into/together with HEP analysis culture
- Considerations for workflows:
    - Bespoke environments for analysis (e.g., containerisation support, what runtimes exist)
    - Standalone from analysis (i.e., separate repos for the workflow and analysis code)
    - Support for heterogeneous computing infrastructures (distributed computing, non-CPU architectures)
    - Graph dynamicism: can graph be computed (efficiently) without neat (i.e., known) graph size?
- REANA platform for HEP, want to leverage going forward
- Many challenges:
    - We *can* run workflows at CERN, but...
    - how do we promote organic (i.e., bottom-up rather than top-down) adoption of WLs (see Open Science pyramid)
- TODO: Link summary paper that was done on workflow languages in HEP

Snakemake overview

Presenter(s): Johannes Köster

- Analysis becomes difficult to do by hand at scale, especially difficult then to reproduce
    - Reproducibility, transparency and adaptability key goals of modern analysis
- Development
    - Accepted as a NumFOCUS sponsored project for sustained development and best dev practices
- Boilerplate-free implementation for scripts allows for closer script-Snakemake integration
    - The
- Wrappers that call out to other workflows in central repos
    - Do people in bioinformatics end up using these as centralised tooling workflows?
    - Have multiple categories that can tag things by fields
- Integration for “external” workflows enables separation of code and workflow, and extension of existing workflows

- Job grouping simplifies DAG optimisation without loss of modularity
- Conda environment integration (yaml specification of dependencies)
    - Can it use conda-lock lock files?
    - Would podman be allowed? What would be required in the future?
    - --containerize translates conda environment(s) into e.g. a Dockerfile
- HTML report generation for workflow documentation
- Service jobs
    - Could allow for Dask?
- Snakemake 8 includes *most* functionality from in 7
    - XRootD snakemake plugin

Questions

- What HPC integration is available?
    - Generic plugin for anything that receives submission scripts, additionally specific plugins for specific services (e.g., Slurm, HTCondor)
- Unit testing with Snakemake
    - Generates tests for pytest, tests per rule can then be run in pytest
    - Generated tests potentially extendable to use in ci-testing
- Snakemake built on individual solutions -> motivation for new structure and Workflow catalog

CWL overview

https://carpentries-incubator.github.io/cwl-novice-tutorial/ Teaching workflow thinking to novices

Presenter(s): Michael R. Crusoe Slides

- Workflows are “type of structured computing”
    - Focus on scaling, automation, provenance
    - Complement unstructured approaches (e.g., bridge between tools)
    - Focus is on standards
- CWL covers many tools in many languages
- *Many* workflow languages exist (at least 348) -> need a standard for workflow languages
    - Agnostic on container format, simply state what is used
    - Designed to improve workflow FAIRness
- CWL is two standards in one
    - CWL Command Line Tools (CLTs) description
        - How to run tool, allowed/required inputs, outputs and how to get them
    - CWL Workflow description
        - Connect CLTs and sub-workflows into workflow graph
- New since 2017
    - CWL paper in 2022: https://cacm.acm.org/magazines/2022/6/261172-methods-included/fulltext
    - Joined SDO (part of Software Freedom Conservancy)
    - Included in IEEE standards (referenced by FDA in US)
    - Functionality:
        - Workflow level conditionals
        - ToolTimeLimit for wall time-based schedulers

        - WorkReuse for steps that shouldn't be cached
        - InplaceUpdateRequirement adds hint that destructive data edits allowed
- Technical details:
    - Can specify file formats
    - No requirement of shared file system (quite important for HEP, where we have baked in assumption of having a shared filesystem with CVMFS)
    - Modular, reusable design, designed for HPC computing
    - Validator to catch syntax errors prior to running
- Data Model uses CLT as a basic unit
- Portability / environment extension
    - Are just generic environment files supported? Or are lock file standards (e.g. conda-lock, pixi.lock) supported?
- Questions:
    - Is CWL top down or bottom up?
        - CWL isn't target based like Snakemake
    - Charles: Interested in portable workflows. Then want to be able to run it locally, on cloud, on HPC, etc, but without having to do a lot of management and editing. Is this possible?
        - Yes. Snakemake helps you configure things in general, but CWL might need you to configure your runner (but this is ideally provided by the ops people at your resource location). (c.f. "CWL Enables Execution Portability" slide)
        - There's no CWL infrastructure, there's just the standard. Though c.f. the "CWL in the HPC ecosystem" talk after lunch on this.
    - Lukas: Can we have people write in Snakemake for usability, but then export to CWL for long term preservation (Here we're thinking explicitly of RECAST workflows)
        - Somewhat complicated to be able to export all functionality to CWL. You'd need to more port over custom Python code to CWL, which requires more work.
        - Michael: Totally worth doing

CWL in the HPC ecosystem

Presenter(s): Iacopo Colonnelli

- CWL comes with functionality for reproducibility, reusability, etc., but what about for HPC orientation?
    - Build HPC support inside (CWL4HPC WG) and around CWL (StreamFlow Workflow Management System (WMS)
- CWL4HPC WG (https://www.commonwl.org/working-groups/cwl4hpc) aims to identify workflow patterns for modelling large-scale scientific applications and implement through CWL enhancement proposals
    - Propose pattern, motivate with 2 real use cases and agree on first draft of syntax/semantics.
    - Implement as a CWL extension on cwltool (the reference implementation) and another CWL-compliant WMS (with conformance test).

    - Validate on at least two existing CWL workflows (where proposal is applied or new workflows for demonstration).
- Workflow patterns:
    - Parallel: scatter method to decompose jobs across input parameter
    - Iterative: loop extension iterated for iterative workflow support
    - Concurrent: requires streaming (Stream type not not yet supported in CWL), CWL structure regular enough to allow automatic conversions of arrays to streams (gather+scatter/loop+scatter) but not all safely optimisable.
        - Channel CWL extension proposed but needs WMS support.
        - External steering can be modelled as concurrent iteration pattern, CWL operation class can model workflow steps as custom processes.
    - Coupling (e.g., grouping in Snakemake): not supported in CWL; needs communication channels between ports of workflow steps and application-agnostic protocols.
- Large-Scale Workflows
    - Steps can require multiple intercommunicating agents (e.g. Spark or Dask cluster)
- Hybrid workflow has steps which can span multiple, heterogeneous, independent infrastructures
    - Step becomes fireable when input dependencies ready and transferred, and related location deployed
- StreamFlow WMS (https://streamflow.di.unito.it/)
    - Workflow description (e.g., CWL) and model description (e.g., Kubernetes, Docker) are separated, linked by StreamFlow file
    - Faster with hybrid workflow in federated learning example
        - How much expertise did this require? HPC workflows require expertise in general, but how much CWL expertise was required to do this multi-federated workflow?

Questions

- Lukas: how is data movement handled?
    - Iacopo: data movement depends on what you want to target: if the space is common then it is not moved; if not then it retrieves and copies
- Charles: for hybrid clustering, where does the scheduler run
    - Iacopo: within the VM, is run from

Airflow and Nextflow applications for neuroscience

Presenter(s): Erik Johnson

- Neuroscience computational operations are starting to reach maturity which now require / benefit greatly from automated workflows
- Public data archives are increasing scale
- Data in the neuroscience domain (structural neural imaging data)
    - 2 petabytes of scale for high voxel resolution
- Intern SDK standardises access to bossDB, DVID, CloudVolume data stores.
- Workflows typically include classification/detection in subvolumes, merging of identified objects and analysis of 3D objects (e.g., graph extraction, density estimates).
- Key workflow management features and limitations:

        - What functionality needs to be exposed to who?
        - What monitoring is required? (use case dependent, i.e., laptop vs cluster)
        - Scalability of execution.
        - Managing heterogeneous resources is a challenge.
- SABER workflow for neuroimaging
    - Now legacy software, but is still open source
    - Built on Galaxy; workflows specified in CWL, CWL parser written to translate to Airflow
- Integrate workflow and data management -> DataJoint+Nextflow
    - Aim to provide low-effort, easy setup integration
    - Setup script builds required docker images
    - Containerised processing steps defined, workflow built from these processes
- Future directions include:
    - Exploring intersection of workflow management and data management
    - Integration with range of archives
    - Streamline workflow and data structure creation process
        - "If you provide a barrier to entry that tends to limit adoption" <- we'll come back to this on Day 3

Questions

- Clemens:
    - Erik: done a lot of work on environments to explore the final data
- Lukas: what is the view of WL developers on integration of data management in WLs and measures to avoid reproducing? Workflow has very expensive data reduction steps
    - Johannes: independent of storage system, checksums used in past (but can be expensive across many jobs), also examined code for changes
    - : monitoring and visualisation very important for this and spotting where things are
    - Michael: some systems will give cost estimations. Should consider whether breaking apart workflows is a problem (e.g., provenance)
- Matthew: would like to circle back to discussion of provenance later.

Airflow overview

Presenter(s): Santona Tuli

- Upsolver
    - Well described by term "Computational database" -> Data-dependent pipelines
    - Based on Iceberg/Amazon S3
    - Transformations written declaratively, then automatically optimised and orchestrated
- Apache Airflow
    - Open source top-level Apache project for monitoring and data pipelines.
    - Orchestrates many technologies, providing overarching infrastructure
    - Python native -> fully programmatic workflow authoring
    - Extensible -> many open source integrations, custom operators are easy
    - Monitoring and alerting -> built in logging

    - Modular workflow architecture (doesn't care about data (data agnostic), just on the work that needs to be done)
    - Communication of data AND metadata (e.g., what version of a model is in production), dynamic generation of tasks, persisted history of runs
- Core components of airflow infrastructure
    - Web server: flask server running with Gunicorn for UI
    - Scheduler: daemon for scheduling jobs
    - Database: database for metadata storage
    - Executor: defines *how* tasks are executed
    - Worker: processes task execution
    - Triggerer: asyncio process to support scheduler
- Core concepts in airflow:
    - *Task* is a unit of work
    - Relation of tasks is a *dependency*
    - *Tasks* and dependencies together is a *DAG*
    - *DAG* instances can be run as *scheduled runs*
    - *DAG* can be triggered manually, on a schedule or by a "dataset" (i.e., waits for new data)
    - *Operators* are wrappers around tasks to define the task purpose, e.g., PythonOperator to execute python callables.
    - *Connections* are used by operators to talk to third party apps
- Model building example -> triaging of issues in Airflow github
    - Training DAG runs once a day: takes issues, extracts features and then trains model
        - Can run tasks independently when changes/fixes made
    - Deployment DAG checks for trained model every 15 minutes, validates new model, chooses and executes deployment strategy (new model if better, previous if not)
    - Prediction DAG runs every 2 minutes to serve predictions on new issues

Questions

- Clemens: environment by operator is unique. How are workflows developed interactively?
    - Santona: multiple workflow workspaces for dev, prod, etc., use dev on subsample to test. Can also develop in, e.g., annotated Jupyter notebooks and then write into DAG
    - Clemens: dev and production environments one and the same in HEP
    - Santona: possible to establish environments through e.g., Kubernetes
    - Lukas: what is the development story for workflows? Are there interfaces to run/put together operators (as it seems very low-level) and who puts them together?
    - Santona: not unlikely that airflow engineers at a company responsible for writing operators; standard open source airflow fine for some use cases. Physicist would need to run and debug DAG; writing operators is a different level of development, e.g., how CMS software is separate from physics. Whilst we want analysis process to be robust, Airflow (or similar) isn't necessarily the solution. Containerisation and code versioning better for this.
- Matthew: does Airflow take care of S3 bucket caching in the DAG?

    - Santona: DAG only runs from where it needs to run from, e.g., on fail can run from that task, otherwise exception needed to be included to pull from S3 bucket.
- Similar to the Airflow and Nextflow talk from Erik, are the people who are building these workflows and maintaining them data engineers? Or are the data science / ML engineers also in the loop on these? We want to better understand what the workflow language experience is like for a relatively new Ph.D. students working on their Ph.D. analysis.
- On slide 14 there is the example model. You mentioned that as there are versioned/timestamped outputs in an s3 bucket if the “train_model” step fails then you can pull those previous step outputs to avoid rerunning and then use them for the fixed “train_model” step. Is this something that you have to do manually, or is this something that Airflow provides as a feature?

Discussion and demos

Presenter(s):

Lukas: this is what a “typical” (?) step looks like for an ATLAS analysis (why???)

- 

```
##########################################

########## Step: histogram_stage ##########
histogram_stage:
  process:
    process_type: 'interpolated-script-cmd'
    script: |
      # Get kerberos credentials
      source /recast_auth/getkrb.sh

      # Source the environment
      source /jdm/MonoJetReader/setup_env

      # Copy the uncertainty tool to somewhere where it can be accessed by MonoJetDictMaker
      mkdir /jdm/MonoJetReader/Histograms/VjetsReweightingTool
      cp -r {input_uncertaintytool} /jdm/MonoJetReader/Histograms/VjetsReweightingTool

      # Copy the input shorttree directory to somewhere where it can be read from
      cp -r {input_shorttree}/Output_Shorttrees /jdm/Input_Shorttrees

      # Copy the listOfTrees to /jdm
      cp {list_of_trees} /jdm/listOfTrees_recast.txt

      # Get the anti-SF file, if needed
      mkdir /jdm/inputdata
      if [ -z "{antiSF_file}" ]; then
        echo "No anti-SF file specified. Proceeding sans anti-SF!"
      else
        echo 'Downloading specified anti-SF file'
        xrdcp root://eoshome.cern.ch/{antiSF_file} /jdm/inputdata/antisf_recast_lowMET.txt
        if grep -q {channel_name_antiSF} /jdm/inputdata/antisf_recast_lowMET.txt
        then
          echo "Replacing channel {channel_name_antiSF} with recast_signal"
          sed -i 's/{channel_name_antiSF}/recast_signal/g' /jdm/inputdata/antisf_recast_lowMET.txt
        else
          echo "Channel '{channel_name_antiSF}' not present in anti-SF file. Either add an entry for '{ch
          exit 1
        fi
      fi

      # Remove the reading of SF and averageIntPerXing trees, which aren't used for signal
      sed -i '/^   TTreeReaderValue<Float_t> SFs/d' /jdm/MonoJetReader/Histograms/preDictionaries.h
      sed -i '/^   TTreeReaderValue<Float_t> averageIntPerXing/d' /jdm/MonoJetReader/Histograms/preDictio

      # Run launcher to make pre-dictionaries root files for the nominal histogram and all systematics
      cd /jdm/MonoJetReader/Histograms

      # not run the EL_EFF_ID SIMPLIFIED systs which are just propagated for electron CRs
      for syst in $(seq 0 8); do
        root -l -b -q "Launcher.C($syst,0,0,1,1,\"\",\"\",\"{campaign}\")"
      done
      for syst in $(seq 43 116); do
        root -l -b -q "Launcher.C($syst,0,0,1,1,\"\",\"\",\"{campaign}\")"
      done

      # hadd all the pre-dictionary root files (for nominal and systematics) together
      cd preDict_histo
      hadd final_merged_recast.root o*recast*.root

      # Copy the python dictionary to the output file
      rm {outputfile}
      cp /jdm/MonoJetReader/Histograms/preDict_histo/final_merged_recast.root {outputfile}

      # Repeat for fullJER
      rm -rf preDict_histo
      rm -rf /jdm/Input_Shorttrees
      cp -r {input_shorttree}/Output_Shorttrees_fulljeronly /jdm/Input_Shorttrees
      rm -f /jdm/listOfTrees_recast.txt
      cp {list_of_trees_fulljeronly} /jdm/listOfTrees_recast.txt
      cd /jdm/MonoJetReader/Histograms

      for syst in $(seq 71 86); do
        root -l -b -q "Launcher.C($syst,0,0,1,1,\"\",\"\",\"{campaign}\")"
```

Matthew: provenance (discussion from after Erik’s talk)

Day 2, Thursday 4th April

Session overview

Presenter(s): Clemens Lange, Jamie Gooding

- Dinner at Luigia, table for 19:00, walking over from Building 39 at 18:35

Questions

RECAST workflows in ATLAS SUSY

Presenter(s): Ben Hodkinson

- RECAST workflows developed and used in reinterpretation efforts
    - Plugging in a SUSY signal model is “the easy bit” -> don’t want to lose the hard work!
- Re-use cases:
    - New signal points for statistical combinations
    - Apply SUSY searches to non-SUSY models and vice versa
    - Check new models not already excluded
    - Interpret in wider SUSY parameter spaces
- Analysis code is typically very specific to analysis; analysis team often disbanded after analysis complete
- RECAST preserves ATLAS software env, commands and workflow
    - Needs to include all the steps and how they chain together
    - REANA used heavily for performing reinterpretations
    - Only reprocesses the signal simulation, not other components
- Authoring a RECAST analysis workflow user experience
    - For an ATLAS analysis have a variety of inputs (e.g. dataset ID, simulation sample names, EOS file paths, XRootD paths to simulation). Outputs are generally the final statistical analysis results (CLs value)
    - When writing the workflow rely on Docker images that contain the full software environment and the source code
    - The workflow is then split between the RECAST steps file that defines the actual operations and the workflow file that drives things
    - People doing this work have some issues:
        - Analysis team is quite large, and so a single person doing the RECASTing might not know how all steps of the analysis work
        - Requires understanding multiple technical things
        - Culturally, the implementation of RECAST is more of an afterthought than something that is done concurrently
- Using RECAST for reinterpretation
    - Can design an analysis for reinterpretation for looking for new physics, that is different from what was originally looked for in the published analysis, that exists in a similar kinematic selection region as the original signal models. These won’t be maximally sensitive, but provides information quickly.
    - REANA monitoring webpage quite useful
        - CPU usage for the ATLAS pMSSM scan the is about 5 hours per job
    - Statistical combinations

        - Generate new signal points and then use RECAST
- Challenges using RECAST
    - Harmonisation of inputs
        - Each analysis works slightly differently as there is no enforced standard / schema of how things are implemented. More of a cultural issue in SUSY, but still something that could be made easier
    - Hidden dependencies in the code on the original signal samples (hard coded names)
    - Open science pyramid:
        - Make it possible [X]
        - Make it easy [in principle, but still hard at at lot of corners]
        - Make it normative [About 50 SUSY analyses that are actively useful in RECAST]
        - Make it rewarding [This is unknown still, how to make this happen]
        - Make it required [This was deeply unpopular, and probably won't happen again in the future]

Questions

- Jamie: does signal here refer to the signal contribution to the simulation histogram?
- Michael: The workflows are written in Yadage, or .. ?
    - [Matthew]: Yes, RECAST currently only supports Yadage.
- Jamie: (Slide 10) is the gitlab-registry.cern.ch the same registry discussed yesterday (ATLAS+open science)?
    - [Matthew] No, these are using the built in CERN GitLab container registry that exists automatically for each GitLab repository. The CERN Harbor container registry is separate
- Jamie: training for starting PhD students to avoid learning "cliff" when learning required technologies (Docker, CI, YAML, Kubernetes, RECAST, REANA)?
- Jamie: (slide 26) paper descoped but would the analysis have happened without RECAST?
- Charles:
    - I completely disagree that making RECAST a requirement for graduation/publication is impossible. Points out that many computational science journals require the code to be available.
    - Why do people need to generate specific Docker images?
        - The Docker images are analysis specific and also contain the original analysis source code and then it's built artefact
    - Can steps be shared and cached?
- Valeriia: In the situations where the workflow breaks due to a RECAST issue, what happens? What are the symptoms? Does it just not work? Does it work but give an invalid answer?
    - Sometimes it just fails and then gives an error message (sometimes very confusing ones).
    - Sometimes the workflow executes, but maybe a final step fails silently (exit codes are important!) or maybe merges fail.

Snakemake workflows in LHCb

Presenter(s): Mindaugas Sarpis, Valeriia Lukashenko

Part 1: Valeriia

- Time for experiment development and analysis (2+2) is greater than the average PhD lifetime (~3.5 years)
- Having a structured workflow (Snakemake) enabled better handoffs, especially useful given the turnover during the period
- Snakemake features/benefits used by all: Readability, Scalability, and Configurability
- Only Valeriia: Modularization and Transparency
- Portability was not tested/used.
- Possible dangers: People view Snakemake as a magic button and trust blindly
    - This cost 1 year of time!
- How to increase users
    - Training and promotion (LHCb starterkit)
        - Super important/critical to have people write the Snakefile while starting to write the analysis
        - Talk with Valeriia for advice as she is an HSF training working group coordinator
    - Enforce!
- LHCb snakemake template
    - Analysis Workflow Template (able to be cloned and reused)
- Can use workflows as part of upgrades for doing validation?

Part 2: Mindaugas

-

Questions

- Jamie: has Snakemake guided analyses from a collection of scripts to something more like a software project?
- Xrootd not working in Snakemake 8 because of change of file systems are included
    -

Luigi/Law workflows in CMS

Presenter(s): Marcel Rieger

- Reproducibility and portability are important for analyses
- On the complexity vs. scale problem, Luigi handles complexity and Law handles the scale
    - The scale can be quite large. Well into the millions of task operations.
- Luigi workflow language: Operates on task system
    - Able to build CLI API quickly from user defined Luigi classes
    - Execution model is make-like
- HEP concepts, constraints, and peculiarities
    - Systems designed for analysis preservation might not be the best system for a day-to-day workflow development environment
    - Might need to change the design of an analysis design well into the analysis itself, and so need the flexibility of tooling to adopt new changes
    - Remote storage is mandatory. Local storage (e.g. lab / university) is not always sufficient.
    - Analyses are large: Order 10^6 tasks in the task graph
    - Very heterogeneous compute and analysis facility infrastructure

- Law:
    - Extension on top of Luigi
    - Toolbox to follow an analysis design pattern
    -

Questions

- [Matthew] When you talk about remote storage, does this mean streaming?
- [Matthew] Mentioned on slide 22 that if you wanted to separate analysis code and workflow code, but how hard is that to do in reality? Could you use Law and have the workflow logic/control be separate?
    - Opposite: Would instead recommend keeping things separate by default
- [Jamie] When you mention the job submission, there seems to be no additional boilerplate but it is just built in at runtime. Do you have feedback from early users on interfacing with HPC resources where people wouldn't use them without this, given they would have to learn how to write these configs?
    - People seem to like to be able to work between HTCondor and Slurm easily
- [Johannes] What is confusing is that you need to be able to inherit the law.HTCondorWokrflow superclass to be able to use things, so if you wanted to use Slurm then you would need to edit the class as well.
- [Lukas] Seems that it is true that for an analysis preservation context that Law isn't designed here as one analysis could be optimised for running on a particular institute, but that won't necessarily work if you're trying to preserve analyses and rerun them elsewhere
    -

HEP-CCE

Presenter(s): Charles Leggett

- HEP-CCE explores the intersectionality of HEP software and HPC environments
    - How to exploit parallelism
    - How to run across multiple GPU architecture
    - How can we get the theory stack of simulators (e.g. MadGraph5) to run on HPC/GPU systems
- Focus on portability
    - What does it take to have a HPC workflow designed on one HPC facility and make it run on another?
- HEP workflows don't map well to traditional HPC
    - Highly non-uniform workflows (the reconstruction toolchain steps have vastly different runtimes. Some steps take seconds, other hours)
    - Need for real-time and on-demand compute resources
- Resources access challenge
    - When Rubin observatory observes a supernova it needs to be able to have access to compute on demand in short order
- Software environment challenges
    - Requests for CVMFS and EOS are non-trivial and some sites just don't support them
- How do you deal with interrupted data?

    - If data streaming is happening for a supernova event and then something goes wrong, how are you able to pivot and send the stream elsewhere without loss?
- Phase 2 HEP-CCE Plan
    - Provide experiments with validated, portable solutions
    - Deployment of portability overlay
        - Might look like using funcX
- Big questions
    - Why did all the experiments create their own tools?
    - Can we find common tools?
    - Is is possible to have real time monitoring

Questions

- Slide 15: As you mentioned solutions like funcX, are you looking at supporting a large surface area of technologies and languages? Such as using Parsl executors for just funcX and then different solutions elsewhere?
- Are there particular examples of CWL and Snakemake being used on HPCs? If so, can you point us to them?
    - CWL: iacopo.colonnelli@unito.it
- Grafana?
- [Luke] Analysis can be an iterative process, and will want to be able to know if there have been changes which affect the analysis and also the performance.
- [Lukas] In some Swiss computing centres there is a push to work on Kubernetes on HPC, as k8 is cloud native then it already provides a portable substrate. Is this something that is considered for the US HPC facilities?
    - US has quite a bit of inertia. Though the HEP-CCE IRI might push people towards adopting common APIs.
- [Clemens] On slide 17 we have issues with data provenance now for HEP where we run and then we generate ntuples which are distributed, but we don't keep provenance on this. It would be nice if we could have workflow tools that could help to add provenance information on how data were produced.

FAST-HEP: Confronting the challenges of developing a workflow language for HEP analysis

Presenter(s): Luke Kreczko

- What do we actually need?
    - Data input
        - Local files (ROOT, CSV, npy, HDF5, JSON)
        - Remote files with/without auth (xrootd, https, S3)
        - Diverse (e.g., more than one tree
        - Meadata (nEvents, dataset type, production year, provenance and workflow versioning)
    - Assumption: convenient to create configs -> easy to modify and diff
    - Regions of interest and systematic: in short, ability to clone subgraph for variations
    - Execution: delegated workflow to other software (coffea, dask, HTCondor, prefect, etc.)

- Sharing presents significant challenge:
    - Workflow sharing
    - Data sharing
    - Responsibility sharing

Questions/responses

- Michael: Sometimes we spend a lot of effort working around sharp (technology, social) corners; but really we should just fix them so we stop hurting ourselves.

Discussion and demos

Presenter(s):

-

Day 3, Friday 5th April

Session overview

Presenter(s): Clemens Lange, Jamie Gooding

REANA Overview and Demo

Presenter(s): Tibor Šimko

- The long term value of the data collected at colliders is huge. These experiments have very specific run conditions that are unique to that run in time.
- While best practices for reproducibility are known, it is hard to make it used in practice when it comes to actual science
    - Most of this is sociological
- Analysis Preservation
    - Data preservation:
        - CERN Open Data Portal
        - CERN Analyais Preservation
        - HEPData
        - Zenodo (with GitHub integration)
    - Computing environments
        - Linux containerization
    - Computational Recipes
        - Build DAGS that can be thousands of steps
        - REANA supports multiple computational workflow engines
        - How to make it actionable?
            - Need good testing and integration
            - Jupyter notebooks are popular but there is low reproducibility of them
- REANA architecture
    - Support multiple workflow engines and containers on Kubernetes
    - Installable via HELM charts
    - Have tested on Google Cloud and super computers
- REANA job controller allows for a hybrid workflow
- Reproducibility vs. preproducibility
    - https://doi.org/10.1038/d41586-018-05256-0
- REANA could be hooked up to GitLab to even run things in a CI/CD system-like workflow
- Looking at adding Dask support (Slide 32)
- Discussion topics:
    - Gherkin behaviourlal tests

Questions

- For ATLAS pMSSM basically developed against REANA, as knew this is where the execution would be. This could be slow as needed to rebuild container images, but is there a way to speed up things?
    - A few ways to do this:
        - Can develop locally

        - Can try to install REANA locally, but
        - Can export the workspace on EOS, and then using kerberos tokens to copy things over to REANA
- In the scenario where there are multiple people working against a single workflow execution, so someone on a team can change a single bit of the code but then maybe that person isn't the first who executed the workflow. So can you have co-authors on a team?
    - Currently no, but a team could user a shared service account.
        - Let's try this
- Are there experiments outside of the LHC using REANA?
    - The PHENIX experiment at BNL is
- Slide 32: What would trigger the allocation of a Dask cluster and how do you provide information on the resources? From the point of view of the workflow, nothing special happens as the Dask cluster exists already as Dask has already been provisioned.

REANA for IRIS-HEP AGC

Presenter(s): Andrii Povsten

-

Questions

Discussion: Workflow adoption roadmaps

Presenter(s): Clemens Lange, Jamie Gooding, Lukas Alexander Heinrich, Matthew Feickert

-

Questions

Discussion: Workflow adoption challenges

Presenter(s): Clemens Lange, Jamie Gooding, Lukas Alexander Heinrich, Matthew Feickert

-

Document writing

- Original ideas related to the workshop creation was that while we've had some success stories in HEP, we don't have workflow languages used in our daily experience of doing physics.
- What should we write to summarise the workshop / provide a document to encapsulate knowledge? Options:
    - Put a paper on arXiv that summarises what we as a community learned from the workshop and what we think are hurdles that need to be overcome
    - Do's and don't's-style paper (e.g., ten simple rules https://collections.plos.org/collection/ten-simple-rules/)
    - Delay paper and have a follow up workshop that is more user focused
- Thoughts:
    - Personal view is that the value of the workshop

    - Don't have to *just* pick one of the options
    - The sociological challenges are real and so important to have these things very clearly written. Shouldn't shy away from the difficulties that people have faced in trying to get these
        - Open is not enough paper
    - If involve the conveners then can enforce standards and policies
        - Though ATLAS had the soft requirement for a few years before this was demoted from a "soft requirement" to a "suggestion". Have a strong focus on user experience.
    - Should reach out to experiments *outside* of CERN. Involving DUNE would be quite important.
    - Can contribute to the HSF analysis preservation with information from this workshop
        - Currently we have an HSF REANA training lesson using Yadage; and plan to make a Snakemake version: https://hsf-training.github.io/hsf-training-reana-webpage/
        - HSF snakemake tutorial isn't very standalone at the moment, so can try to make this more modular into the future. We can propose new tools.
    - Training is good, but we also need to make sure that the user experience even after training is good.
    - Seems that a new workshop that is more user focused is important, so we should do that
        - In terms of what is written, perhaps a **short** whitepaper that summarises the difficulties and the variety of what were discussed.
        - The 10 simple rules has the advantage of being a clear document of what people should do.
        - Charles: Each experiment should put together a few non-trivial examples of what is needed to actually
            - Kinda already have this, but they aren't public and so need to have them be in collections
        - If you have whitepapers that show that there are areas of work then

"Ten simple rules for making a software tool workflow-ready"
https://doi.org/10.1371/journal.pcbi.1009823

Things that we [physics analysts] need from workflow tools

- to be able to set container cache dirs globally (snakemake won't let you control this even with `envvars`, e.g. how can we set environment variables such as APPTAINER_CACHEDIR?)
    - Johannes: Set this on the system, though Snakemake should be respecting this.
    - Matthew will open an GitHub Issue.

- Podman support to be able to have local docker experience
    - `cwltool` and `toil-cwl-runner` have this supported today with `--podman`
    - Snakemake support coming with plugin system https://github.com/snakemake/snakemake/issues/2414
    - Alternative: install REANA locally (using Docker or Podman works fine) and run CWL, Snakemake, Yadage workflows via REANA rather than via using vanilla cwltool/snakemake commands. (This will use Docker or Podman even if the workflow engine does not directly support the technology yet. Ditto for Krb5, VOMS, Rucio and other such technologies that are plugged into REANA and may not be fully plugged yet into upstream workflow engines.)
- Workflow tools need “to get out of the way”, i.e. not be installed into development environment since they might not be compatible (mainly Python versions and packages)
- Storage plugins to work with EOS, XRootD, CVMFS(?)…
    - Auth?
    - Incorporate file system knowledge into plugins, e.g. don’t check individual file existence if parent directory does not exist (done in some snakemake plugins)
- site-dependent configurations (e.g. XRootD vs. webDAV, HTCondor vs. Slurm)
- For tool dependency resolution, `cwltool` has a configuration option available to map software names/identifiers to site-specific providers: https://cwltool.readthedocs.io/en/latest/#leveraging-softwarerequirements-beta
- Resubmission logic/option
- Control filesystem mounts in containers at a per-task level (-v /cvmfs:/cvmfs …)
- Enable authentication inside a container (-v …:/tmp/kerberos…)
- Useful to have example workflows for HEP as a example for wflow languages?
    - “AGC-for-workflows”:
    - https://github.com/reanahub/reana-demo-bsm-search was one attempt
    - *The* AGC *with* workflows: https://github.com/columnflow/agc_cms_ttbar
    - Make sure it is discoverable by snakemake and makes it into the “standardised usage” category
- How to work with workflows that include scale out systems (Dask, Spark, Hadoop, … service jobs?)
    - Control / Submit / Monitor workflows from within Jupyter ?

What do we need to do to adapt to workflow languages/structured computing?

- “Workflow thinking” (M. Crusoe), small tools, clean APIs, ..
    - http://cacm.acm.org/magazines/2022/6/261172/fulltext?doi=10.1145%2F3486897
- “Ten simple rules for making a software tool workflow-ready”
    - https://doi.org/10.1371/journal.pcbi.1009823
- More shared tools, better standardized data access

Scratch Area

https://github.com/common-workflow-language/cwltool/blob/main/cwltool/singularity.py#L166
johannes.koester@uni-due.de

Johannes comments:

- thought for REANA client:
    - One could offer the ability to mount a workspace to the local system via FUSE (e.g. sshfs), so that people can browse and access it like a local volume