Analysis Streamlining

D Costanzo, K Cranmer, L Heinrich, B Kersevan, A Krasznahorkay, K Suruliz on behalf of the ATLAS collaboration
Analyses at the end of Run 2 of the LHC

The LHC and its experiments are entering an evolutionary era:

- no significant increase in $\sqrt{s}$
- focus is on collecting luminosity
  - invest R&D → reduce systematic uncertainties
- We have 10 years experience with analyzing collisions data — take stock of lessons learned
- right time to streamline the way we do analyses
Motivations:

- **make analysis activity as smooth as possible**
  - focus on physics instead of mechanics of running the analysis
  - open up resources for R&D
  - not everyone needs to write their own data analysis framework
  - use well known (industry) tools where possible

- **Easier Coordination**
  - integrate technical aspects of analyses (code, data, etc) into publication progress tracking

- **Analysis Preservation:**
  - Analyses are valuable — preserved them to reuse them for new physics and if needed reproduce original results
  - need infrastructure to make it easy.
After Run-1, ATLAS signiﬁcantly redesigned its analysis model:

- **Goal:** simplify analysis by centralizing large portions, upstream work: creating ntuples, applying cuts & calibrations
  - highly successful **Derivation Framework**
  - works due to common software stack, well-deﬁned operations
  - analysis independent.
- Next Goal: How can we harmonize the downstream, fundamentally analysis-dependent pieces of an analysis.
**Typical Analysis Code**

**ATLAS** made big transition from custom tools to **git & cmake**. **CERN** phased out SVN — very good on-premise GitLab deployment

- Main reconstruction code in gitlab.cern.ch
- Many users naturally started using GitLab for their analysis as well
  - some experimenting with continuous integration
- **Clear Recommendation in ATLAS on what an analysis repository should look like:**
  - pull in dependencies as submodules (e.g. frameworks)

recursive clone == better experience than checking out needed packages by hand

- have as few “toplevel” repos as possible (e.g. if s/w stack is incompatible / very different)
Enabling Preservation

- ATLAS has multiple flavors of software releases
  - Full Release: the entire offline software
    - mostly needed for reco / production jobs
  - Analysis Release: **slimmed down release**
    - mainly ROOT + Event Data Model + CP/Physics Tools

- ATLAS provides **official Docker images** of its analysis releases
  - small size helps drive adoption
  - provides easy way to get reproducible software environment
  - Integration into **GitLab CI**
  - Preservation becomes **easy!**:

```Dockerfile
FROM atlas/athanalysis:21.2.23
ADD . /xampp/XAMPPmonoH
WORKDIR /xampp/build
RUN source ~/release_setup.sh && \
    sudo chown -R atlas /xampp && \
    cmake ../XAMPPmonoH && \
    make -j1
```
Your Analysis

Event Selection Code
- main
- submodule

Stat Analysis Code
- main
- submodule

Data
- BkgA
- BkgB
- Sig1
- Sig2

Repositories
Capture into self-consistent runtimes

Your Analysis

Event Selection Code

Stat Analysis Code

docker

Data BkgA BkgB Sig1 Sig2
Enabling Preservation — The Repo Template

Setting up CI and Preservation by hand is still a barrier: **Use project templates to seed initial repository with analysis preservation built-in**

- uses popular template tool: **cookiecutter**
- **soon: GitLab integration for custom templates**
- nice default CI pipeline (build matrix, Image building, test jobs)
- building integration with internal Analysis Database (Glance)

**cookiecutter atlas-asg/AnalysisRepositoryTemplate.git**

**Continuous Analysis Preservation — from the first commit**
Preserving Workflows

Just building images is not enough: need to preserve the workflows (what to run & in what order)

• Users becoming more familiar with the concept thanks to e.g. CI pipelines
• Workflow as graph of containerized analysis jobs
• Complex workflows (10k) jobs possible
• ...but for many applications partial analysis preservation sufficient & easier
• RECAST

```
# Example of RECAST workflow

statanalysis:
  environment: reanahub/reana-demo-atlas-recast-statanalysis
  cmd: |
    source /home/atlas/release_setup.sh
    source /analysis/build/x86*/setup.sh
    cat << 'EOF' > recast_xsecs.txt
    id/I:name/C:xsec/F:knac/F:eff/F:relunc/F
    {inp[init.did]} name {inp[init.xsec]} 1.0 1.0 1.0
    EOF
    echo {inp[init.url]} > recast_inputs.txt
    myEventSelection {workdir[submit]} recast_inputs.txt recast_xsecs.txt 30 #={lumi}
  output:
    signalfile: workdir['submit/hist-sample.root']

statanalysis:
  environment: reanahub/reana-demo-atlas-recast-statanalysis
  cmd: |
    source /home/atlas/release_setup.sh
    python /code/make_ws.py /code/data/data.root {inp[eventselection.signalfile]} /code/data/background.root {inp[eventselection.signalfile]}
    resultdir={workdir[fitresults]}
    mkdir -p $resultdir
    python /code/plot.py /code/results/meas_combined_meas_model.root $resultdir/pre.png $resultdir/post.png
    python /code/set_limit.py results/meas_combined_meas_model.root \      $resultdir/limit.png $resultdir/limit_data.json \      $resultdir/limit_data_nomsignal.json
  output:
    limit: workdir['fitresults/limit_data.json']
```
Integration into REANA

Result of analysis run on REANA

Close collaboration with REANA to provide service to users to run preserved analysis in the cloud.

- Leverage Workflows and **auto-built** Images
- cloud-native analysis platform
- multiple workflow languages
Integration into CERN Analysis Preservation
LHC-wide archive for preserved analyses. **Streamline ingestion.**

```json
analysis: {
}
```
Integration into CERN Analysis Preservation

LHC-wide archive for preserved analyses. **Streamline ingestion.**

```json
analysis: {
  metadata: {
    ...,
  },
}
```

auto-ingest from existing DB

**Metadata**
- authors
- used methods
- ...

Glance Record
Integration into CERN Analysis Preservation

LHC-wide archive for preserved analyses. **Streamline ingestion.**

```json
analysis: {
  metadata: {
    ...
  },
  data: {
    ...
  },
}
```

**Data**
- ntuples
- fixed aux meas.
- UFO models
- ...

Ingest from group share (EOS)
Integration into CERN Analysis Preservation

LHC-wide archive for preserved analyses. **Streamline ingestion.**

```json
analysis: {
  metadata: {
    ...
  },
  data: {
    ...
  },
  implementation: {
    ...
  }
}
```

**Implementation**
- workflow
- docker images
- source code

[Diagram of GitLab and CERN Analysis Preservation portal]

**NEW YORK UNIVERSITY**
Result: New Physics from old analyses

Reinterpretation enabled by containerized, streamlined analyses
The Future

Streamline Distributed Computing Infrastructure for analyses

- develop code locally → Merge Request to repo
- image built in continuous integration system
- run container-based jobs on the grid

- ongoing work to enable containers on the grid
  - Hot idea: federated Kubernetes (= cloud-native grid)
  - accelerate commit → build → run cycle to be painless

user laptop
local dev
unit & integration
tests (CI)
image building

GitLab
registry
GRID