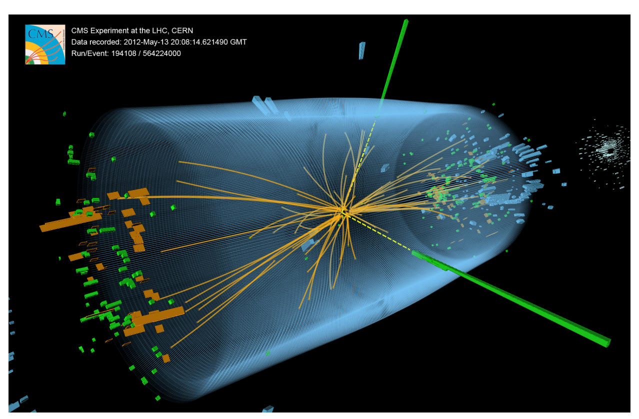
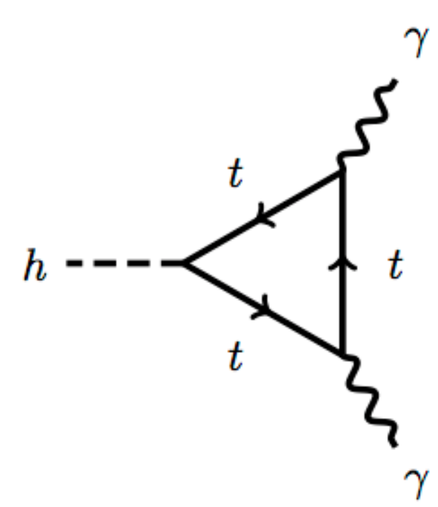
Workflow Management for User Analyses in Particle Physics



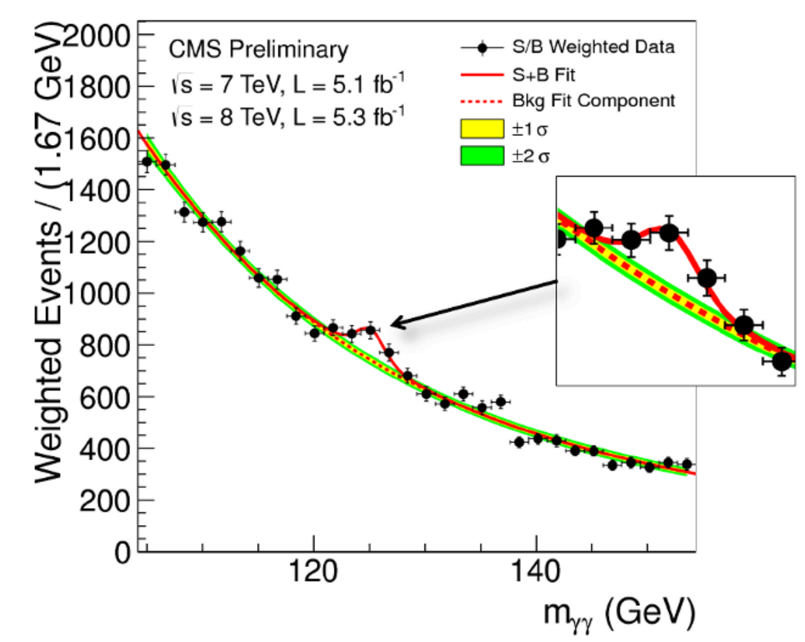
experiment

+

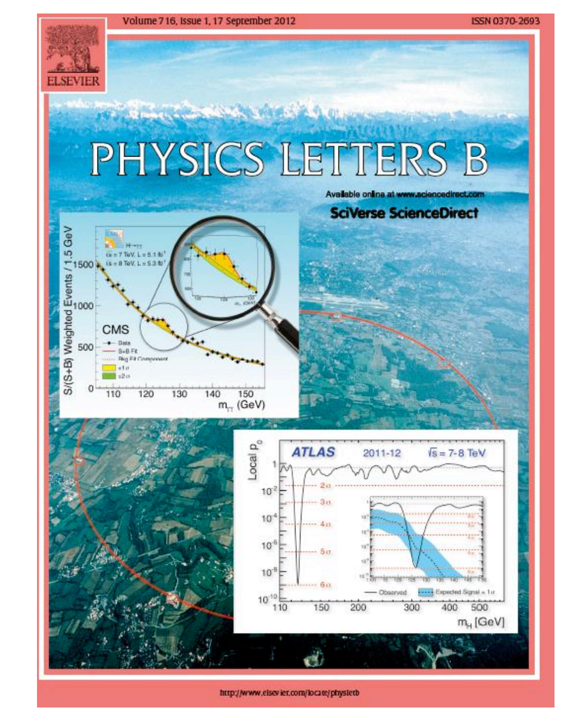


theory

→



→



Robert Fischer

Ralf Florian von Cube, Martin Erdmann, Marcel Rieger

HAP Workshop 2017

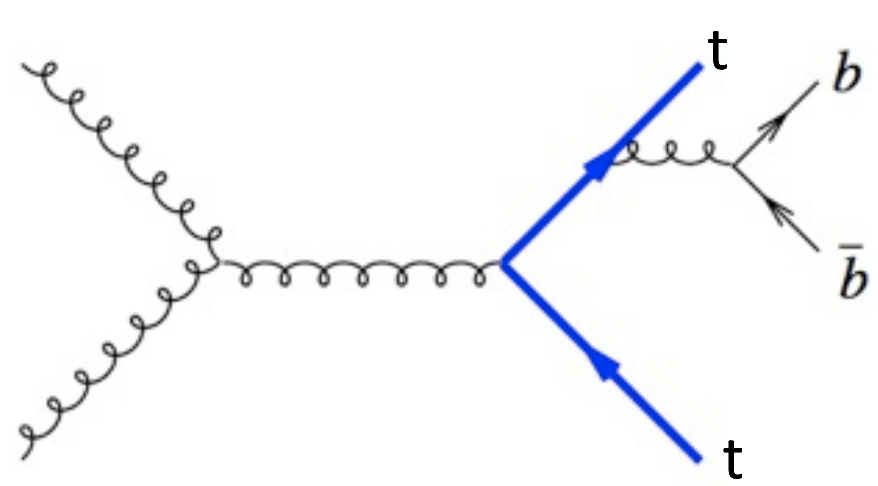
DFG

SPONSORED BY THE Federal Ministry of Education and Research

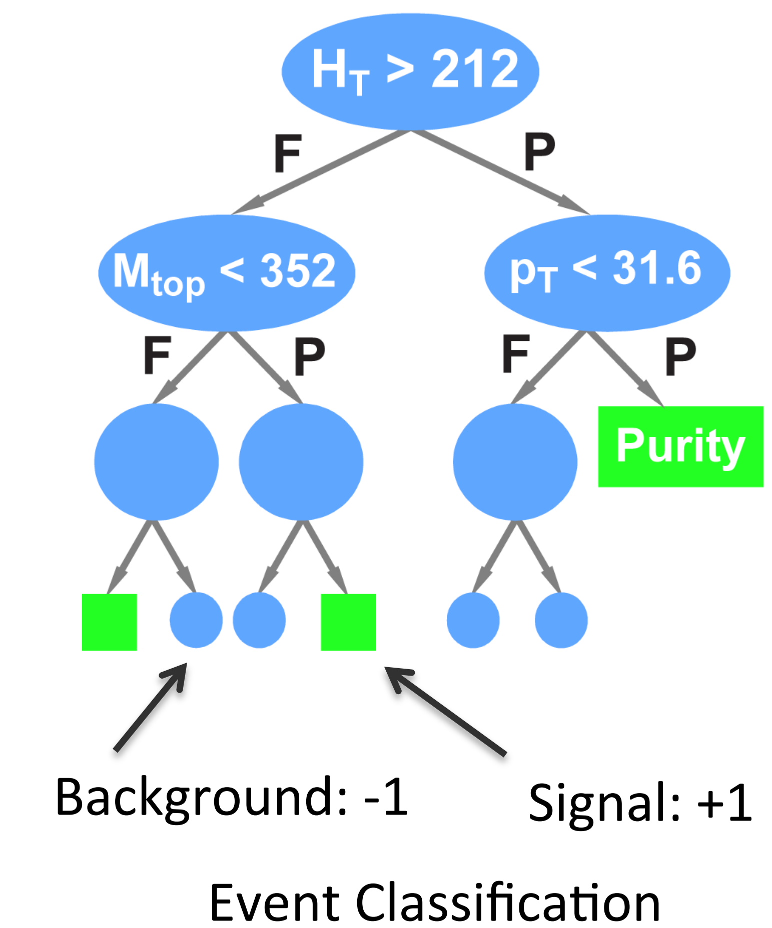
Physics Institute III A | RWTH AACHEN UNIVERSITY

Background

Member of CMS collaboration and VISPA team

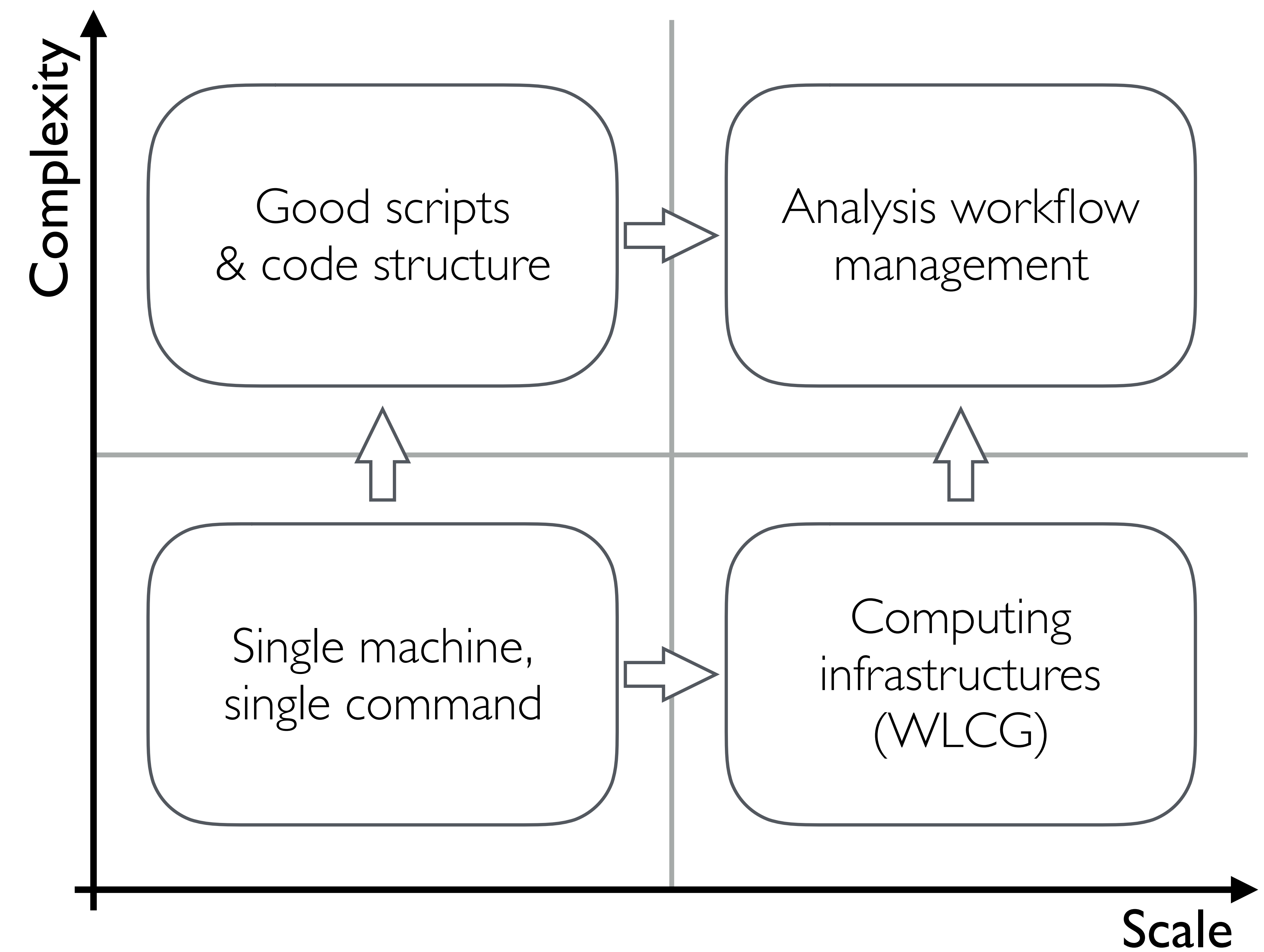


	Data	MC
Events	500 million	300 million
Files	40 k	30 k
File Size	140 TB	100 TB



Landscape of HEP Analyses

- Increasing scale and complexity
- Undocumented dependencies between workloads, only exist in the physicist's head
- Bookkeeping of data, revisions, …
- Manual execution / steering of jobs
- Error-prone & time-consuming



→ Analysis workflow management essential for future measurements

Wishlist

Analysis / Conformability

- Reproducible intermediate and final results
- Adaptable, e.g. during review process, new recipes
- Collaborative development and processing
- Arbitrary programming language

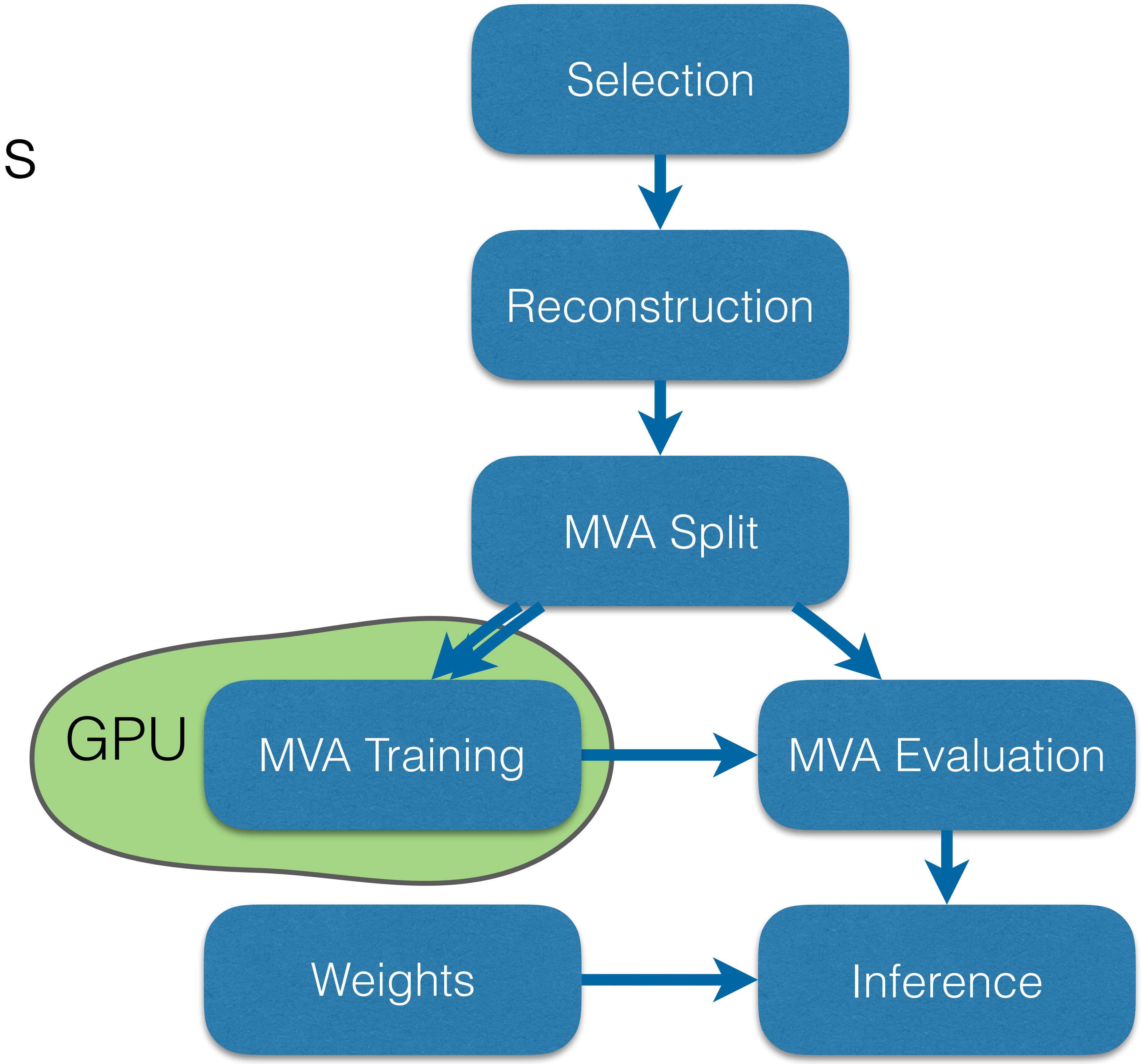
Operation / Convenience

- `make`-like distributed execution
- Opportunistic: run and storage locations
- Automatisation: bookkeeping, data retrieval, dependency resolution, etc.

- Steering in Python

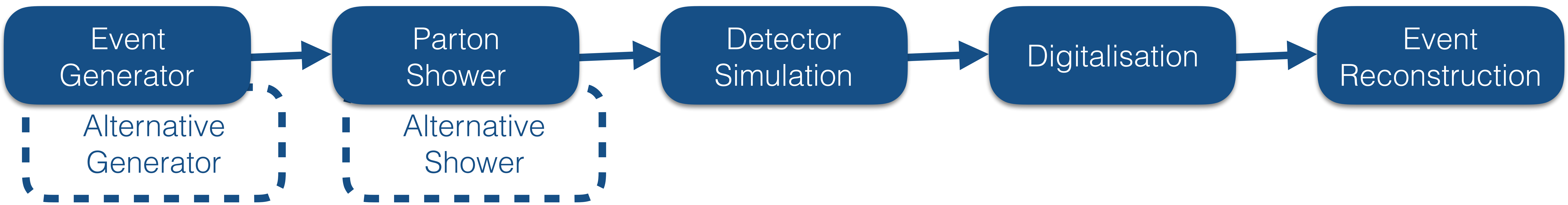
Abstraction: Particle Physics Analysis

- Workflow comprises smaller workloads
- Workloads related to each other by common interface
- Computing resources
  - Run location (CPU, GPU, grid, …)
  - Storage location (local, dCache, eos, …)
- Software environment



→ Large overlap with Workflow Management Solutions

Existing Workflows: MC Production



- Workflows are static, one-dimensional, recurring
- Homogeneous software requirements
- Special infrastructures: Databases, storage, workload management system

→ Static workflows not flexible enough for user analyses

Two Approaches

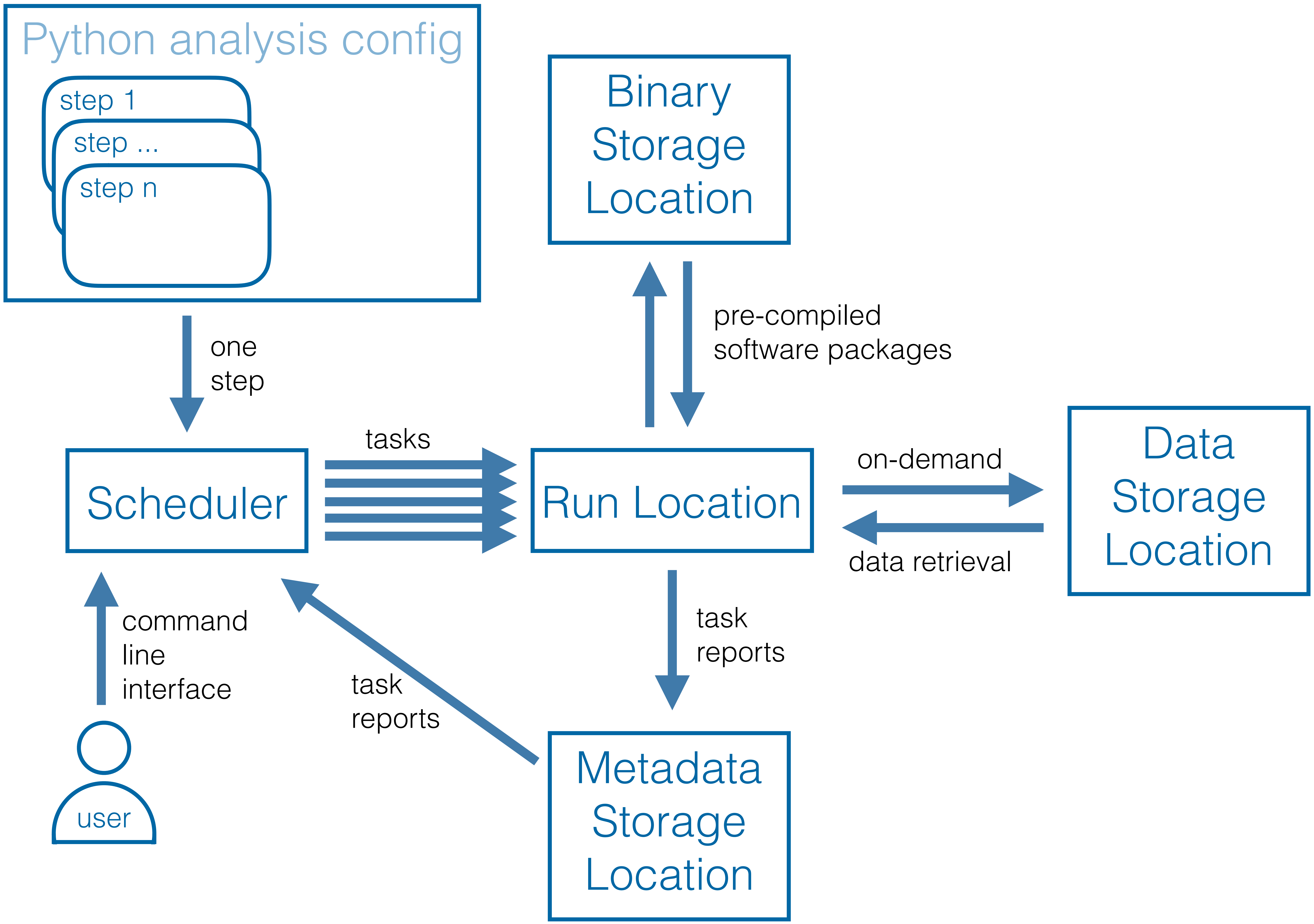
Report Based

- Tailored for HEP from scratch
- Store report file for each execution, evaluate reports of predecessors
- Should perform well if storage is slow

Target Based

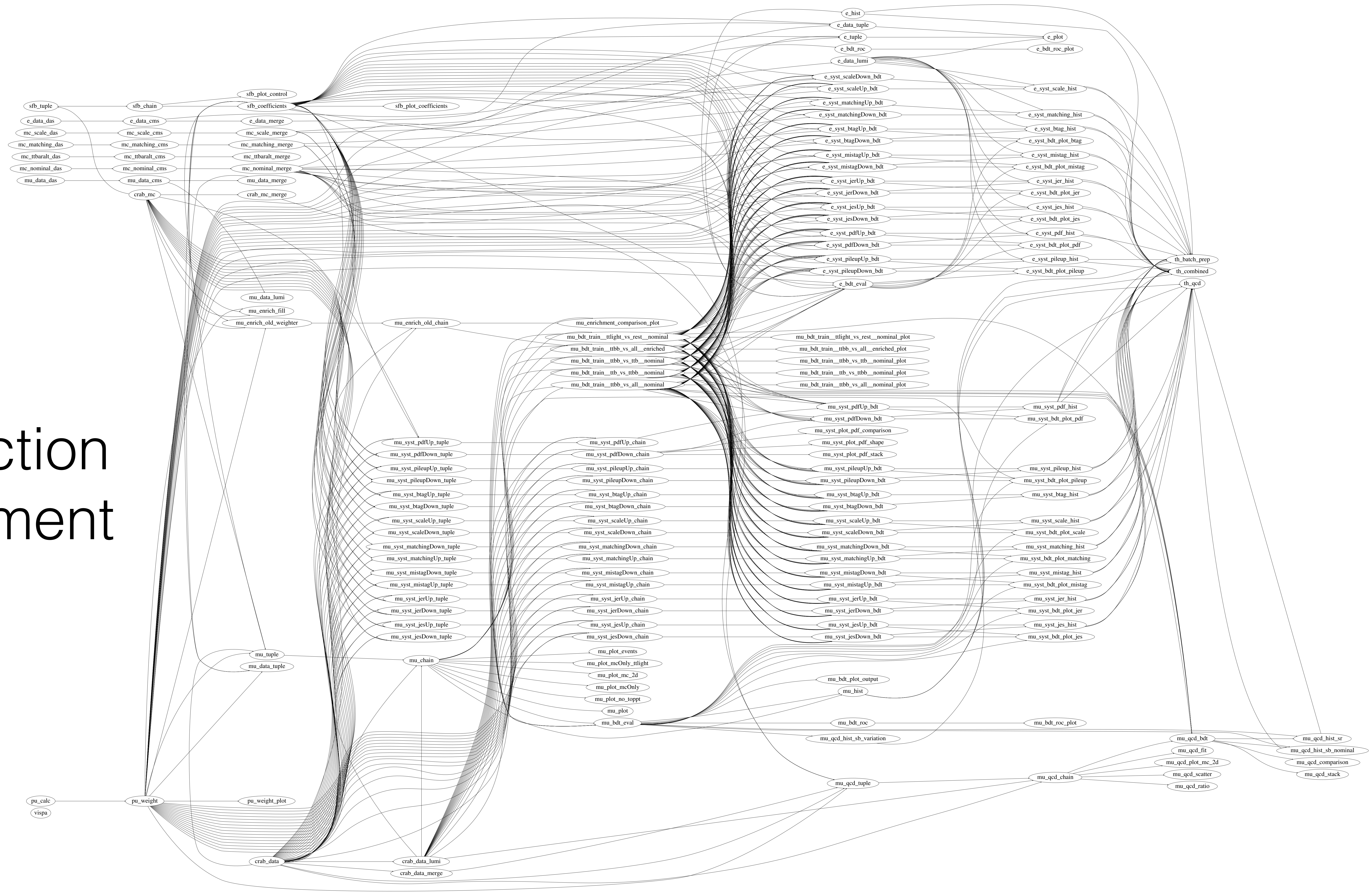
- Which community tools can be adopted?
- Check for output target of predecessors bevor execution
- Should reduce complexity if storage is fast

Report Based Approach



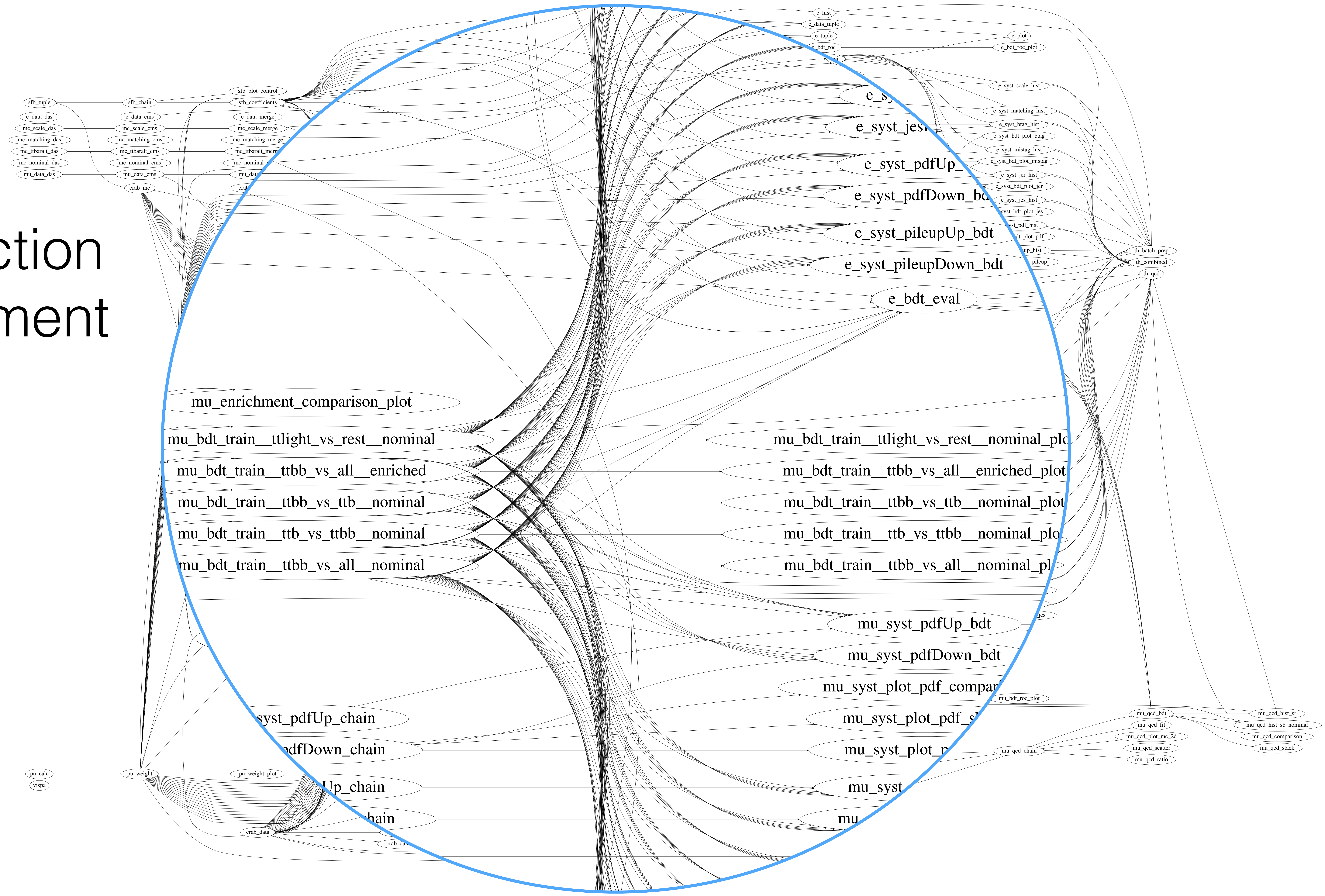
Real Workflow

ttbb
cross section
measurement



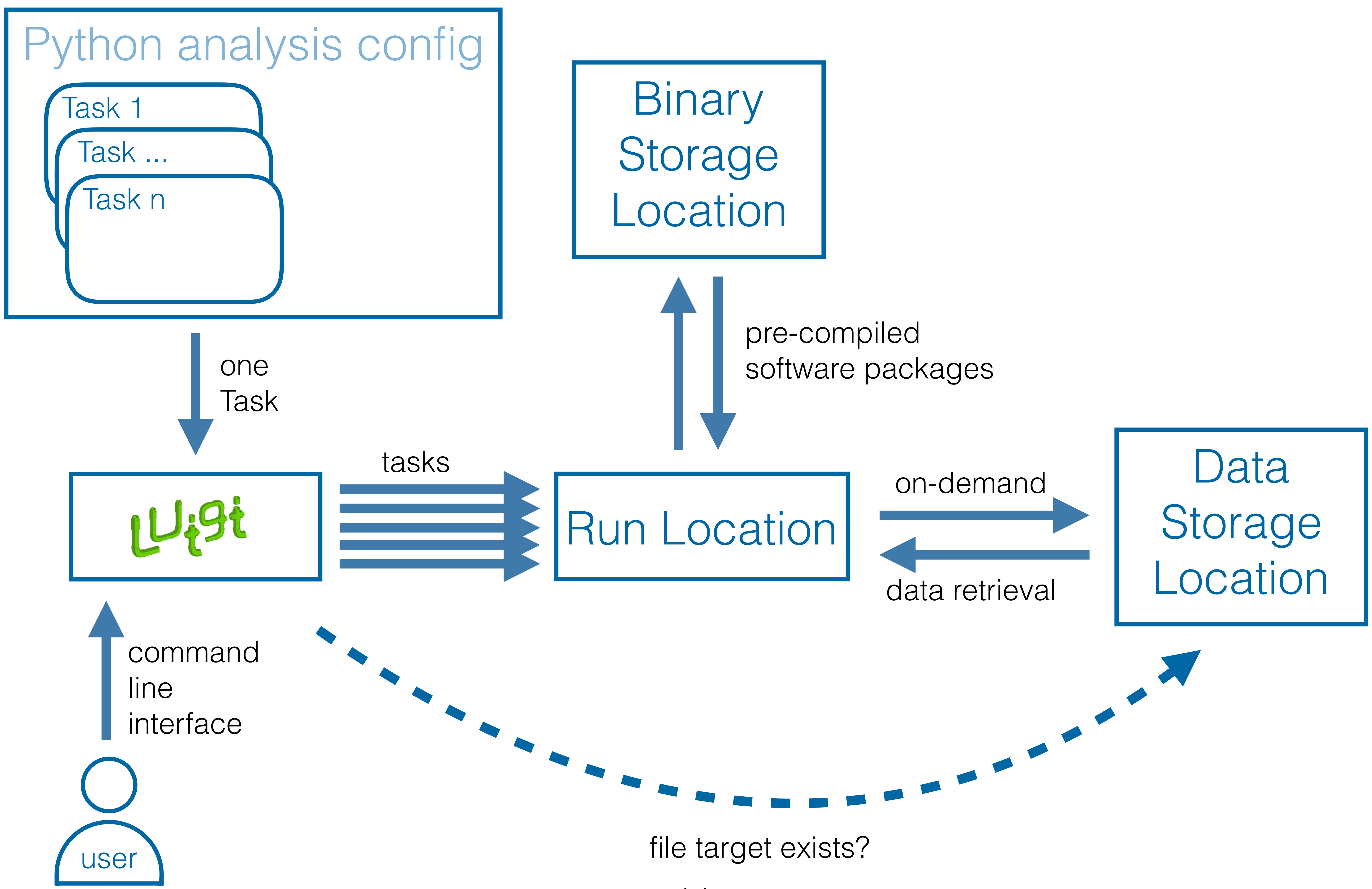
Real Workflow

ttbb
cross section
measurement



Target Based Approach

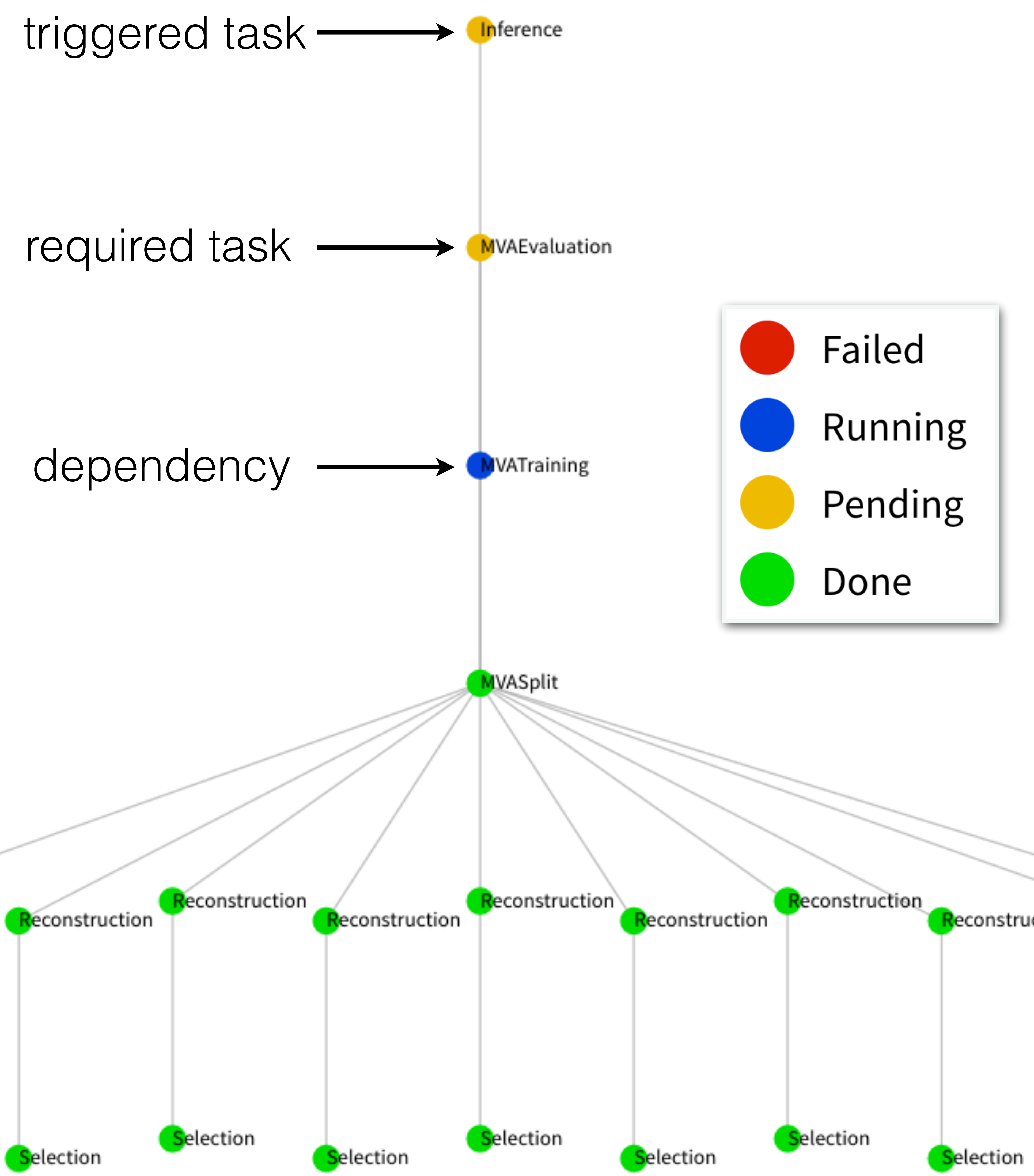
- Luigi: Package for building complex pipelines
- Initially developed at Spotify, now open-source



Luigi Execution Model

- Execution is **`make`**-like
- Trigger one task

1. Create dependency tree
2. Walk down the tree
3. Run incomplete tasks in *n* workers



Software: Environments and Sandboxes

- Opportunistic

  Define environments using existing software on run location

  (dynamic .bashrc)

- Pro-active

  Build and install user software (locally or on run location)

- Sandboxing

  Coherent isolated environment

- Future
  - Docker / containers
  - Lightweight virtual machines

Achievements

- Both approaches proofed with real data analyses

  ttbb and ttH cross section measurements

- Toolbox providing building blocks for analyses, not a software framework

  → ***Design pattern***

- All inputs and parameters transparently encoded

  → ***Reproducibility***

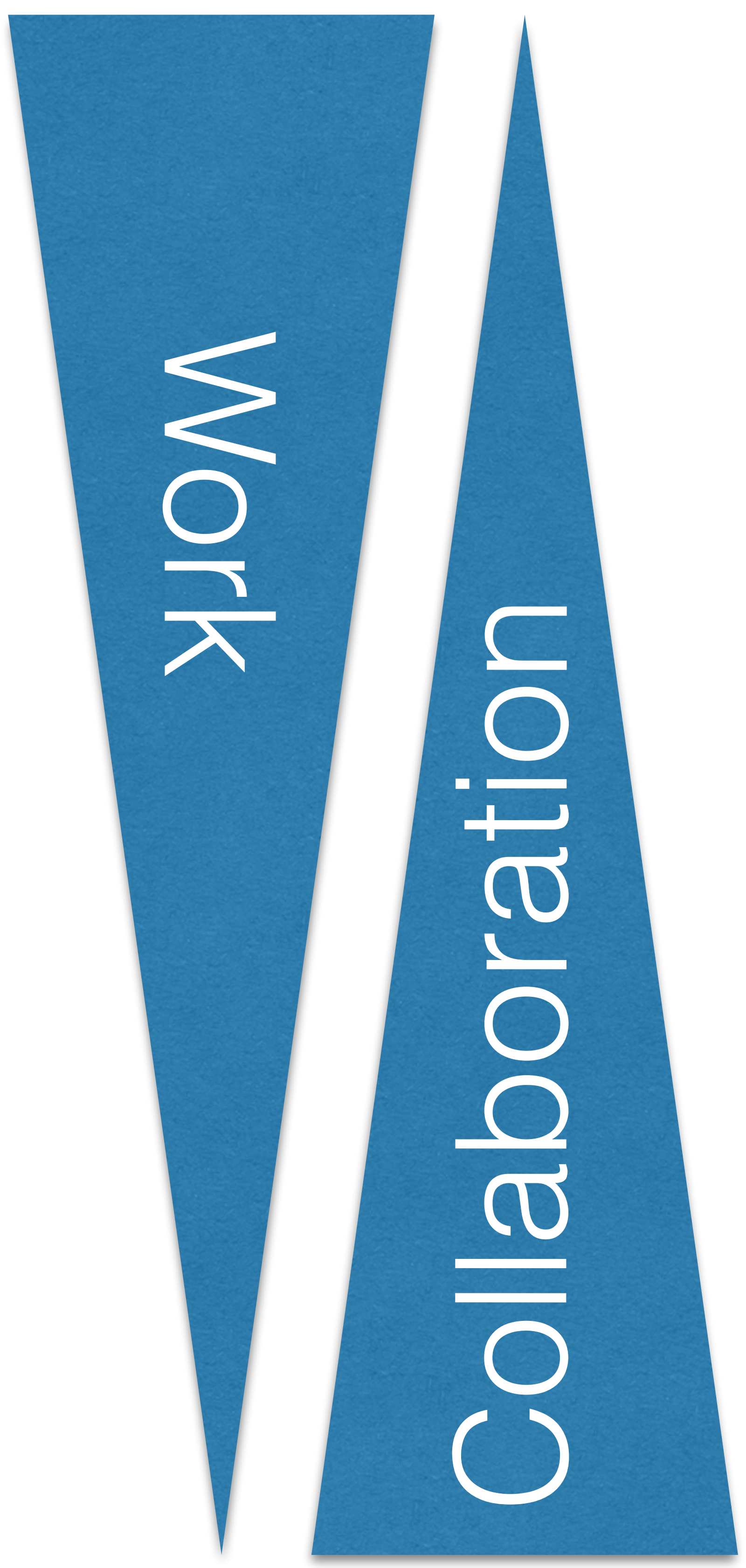
- `make`-like execution across distributed resources demonstrated

  → Reduces overhead

  → **Focus on physics**

Changed paradigm from Executing to Defining Analysis

Collaboration



Written Text	• Wiki • Paper • Slides
Files / Data	• Histograms • Tuples
Code Fragments	• E-Mail • Repository
Shared Workflow	• Same code • Same files

→ Actually work on same analysis across groups

Analysis Preservation

- DPHEP: *Tools and best practices for "adding value" to data*

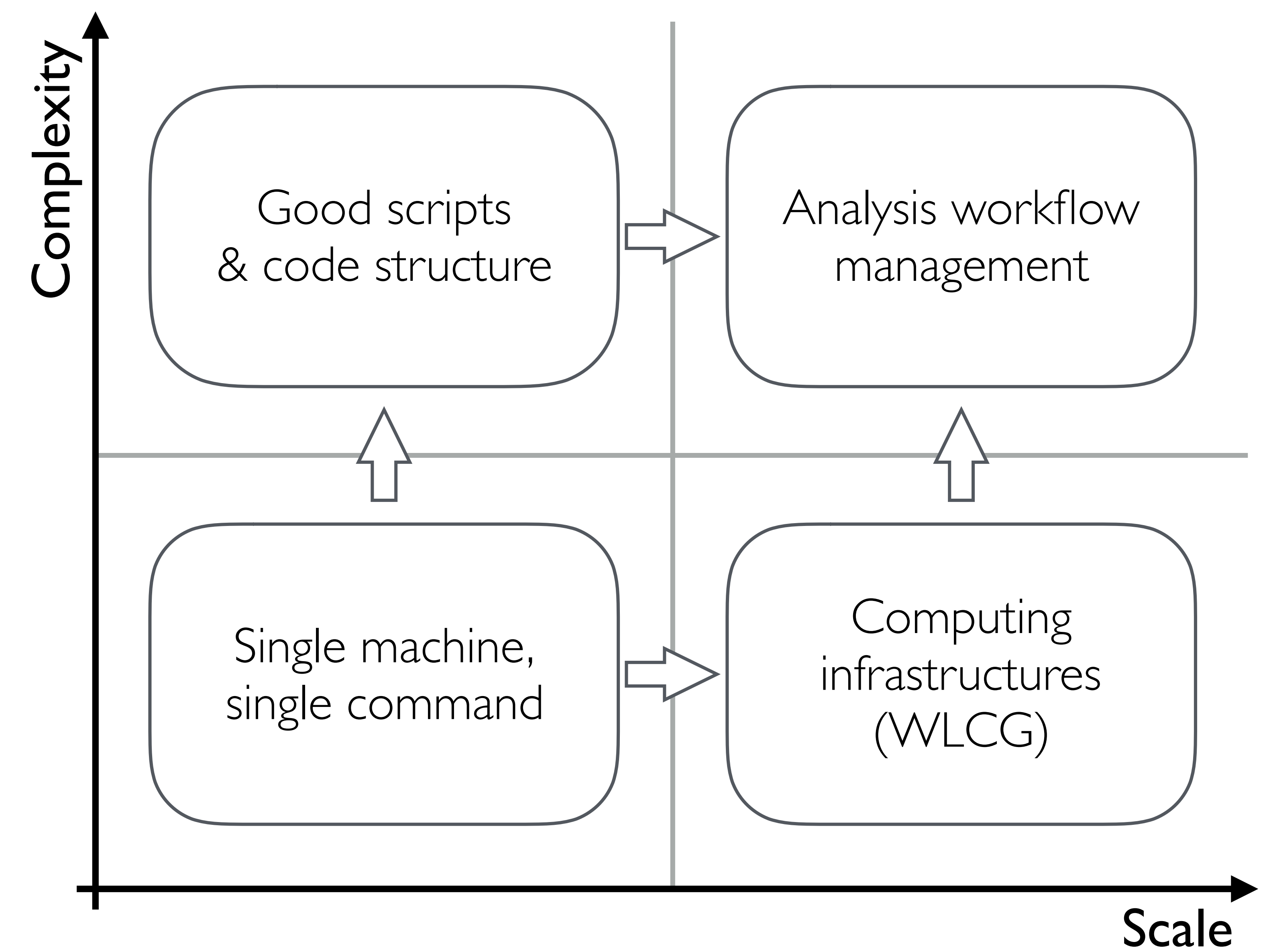
https://hep-project-dphep-portal.web.cern.ch

- HEPData: *open-access repository for scattering data from experimental particle physics*

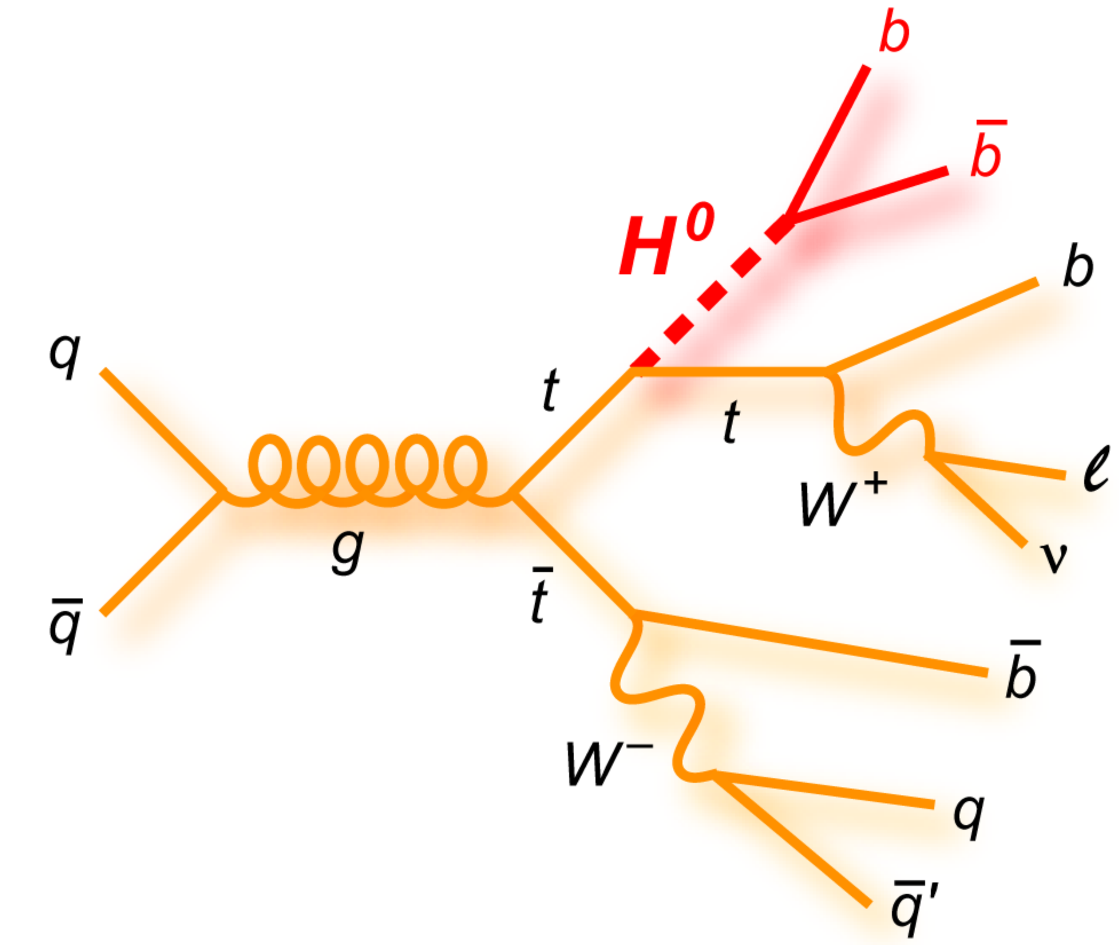
https://hepdata.net

Conclusions

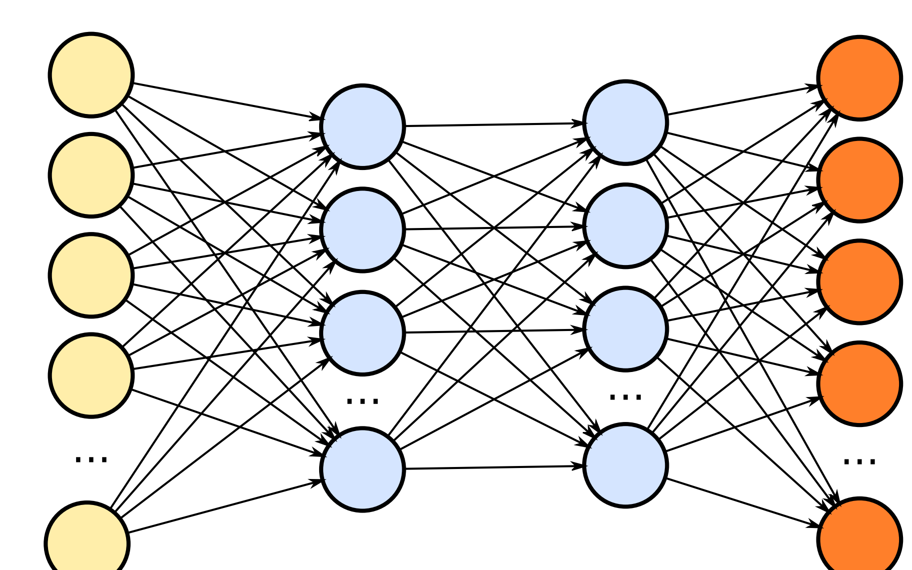
- Trend to larger, more complex analyses continues
- Workflow systems help to manage analyses
- Additional benefits
  - Foster collaboration among analysts
  - Gateway to analysis preservation



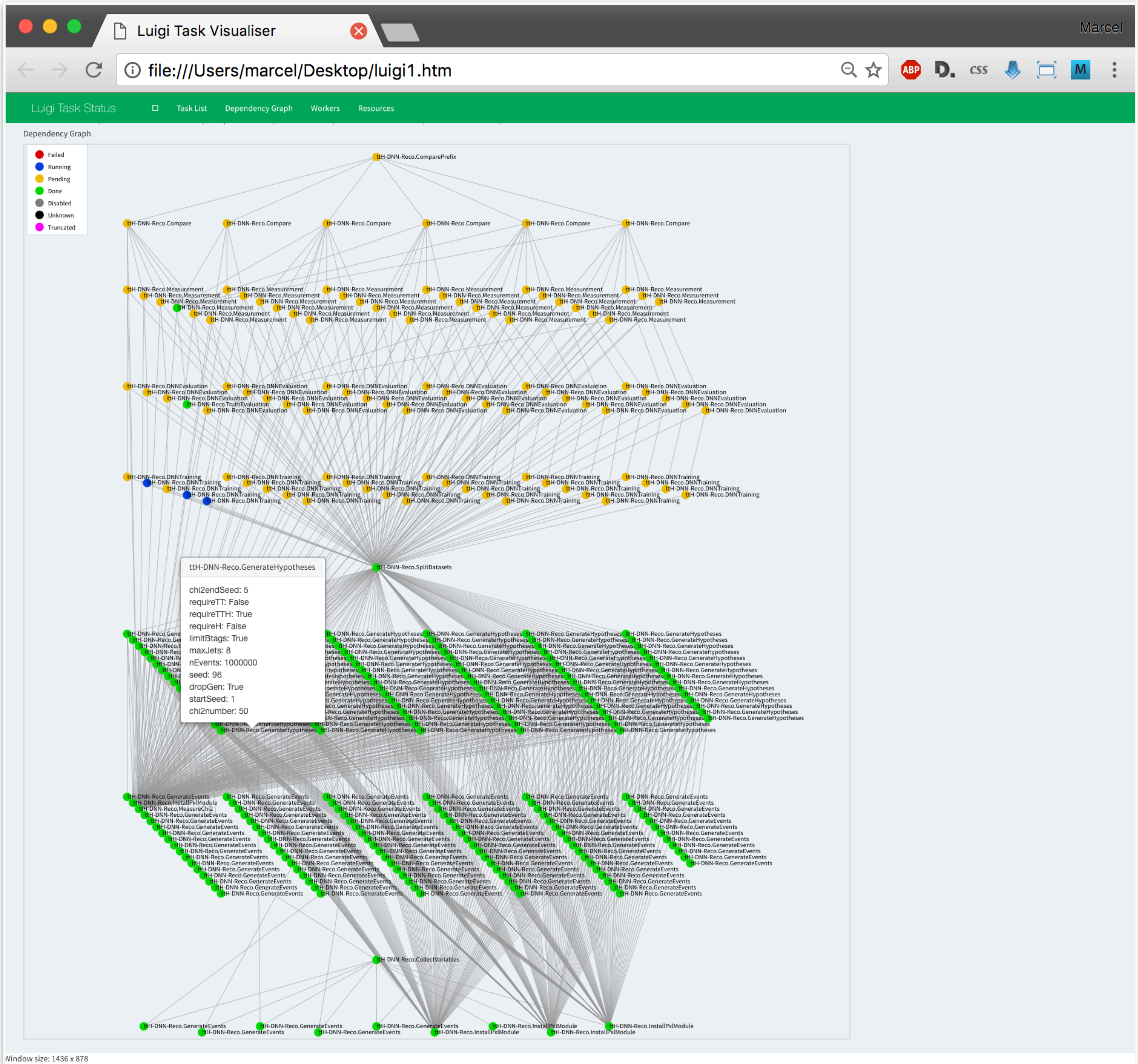
Example Application: ttH Analysis



- Large-scale:
  - ~50k files, ~50 TB of storage, ~1k unique tasks
- Complex:
  - ~40 systematic variations, DNNs/BDTs/MEM, multiple categorization schemes
- Run locations:
  - 7 CEs, local machines, GPU machines
- Storage locations:
  - 2 SEs (dCache), local disk, Dropbox, CERNBox
- Aware of entire workflow at all times, fast dev.
- Clear allocation of duties in group
- Entire analysis operable by everyone at all times



→ Successful proof of usability & suitability



Luigi Task Visualiser
Marcel
file:///Users/marcel/Desktop/luigi1.htm
Luigi Task Status
Task List
Dependency Graph
Workers
Resources
Dependency Graph
Failed
Running
Pending
Done
Disabled
Unknown
Truncated
ttH-DNN-Reco.ComparePrefix
ttH-DNN-Reco.Compare
ttH-DNN-Reco.Measurement
ttH-DNN-Reco.DNNEvaluation
ttH-DNN-Reco.DNNTraining
ttH-DNN-Reco.SplitDatasets
ttH-DNN-Reco.GenerateHypotheses
chi2endSeed: 5
requireTT: False
requireTTH: True
requireH: False
limitBtags: True
maxJets: 8
nEvents: 1000000
seed: 96
dropGen: True
startSeed: 1
chi2number: 50
ttH-DNN-Reco.GenerateEvents
ttH-DNN-Reco.CollectVariables
ttH-DNN-Reco.InstallPxlModule
Window size: 1436 x 878

Luigi in a Nutshell

```python
# reco.py

import luigi

from analyses.ttH.tasks import Selection


class Reconstruction(luigi.Task):

    dataset = luigi.Parameter(default="ttH125")

    def requires(self):
        return Selection(dataset=self.dataset)

    def output(self):
        return luigi.LocalTarget("reco_%s.root" \
                                 % self.dataset)

    def run(self):
        # do whatever a reconstruction does
        ...
```

```
> python reco.py Reconstruction --dataset ttJets
```