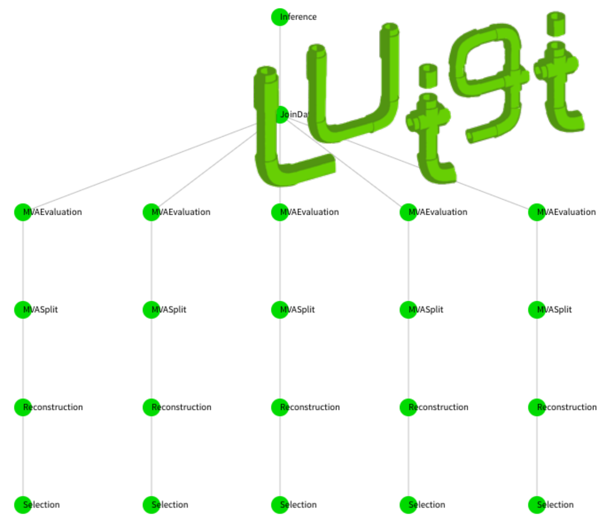


– law –

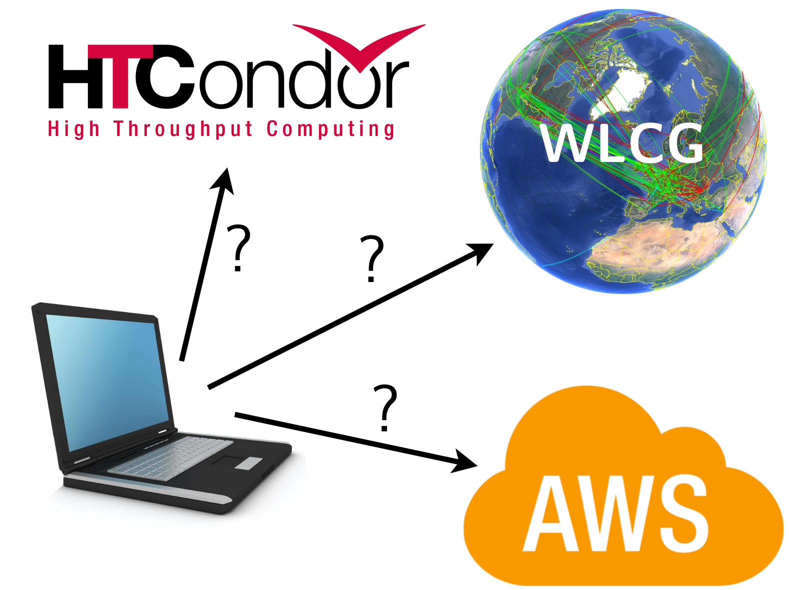
Distributed make-like Analyses on the Grid based on Spotify's Pipelining Package luigi

Marcel Rieger

RWTH AACHEN UNIVERSITY

GEFÖRDERT VOM

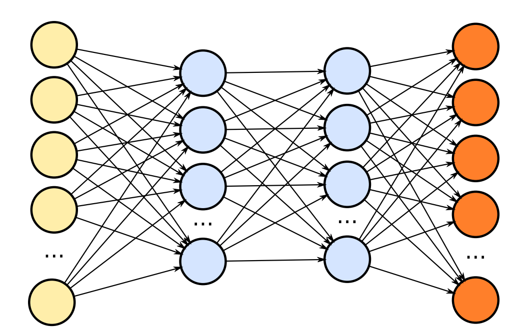
Bundesministerium für Bildung und Forschung



- **Portability**: Does the analysis depend on ...
  - where it runs?
  - where it stores data?
    - ▹ Execution/storage should not dictate code design!

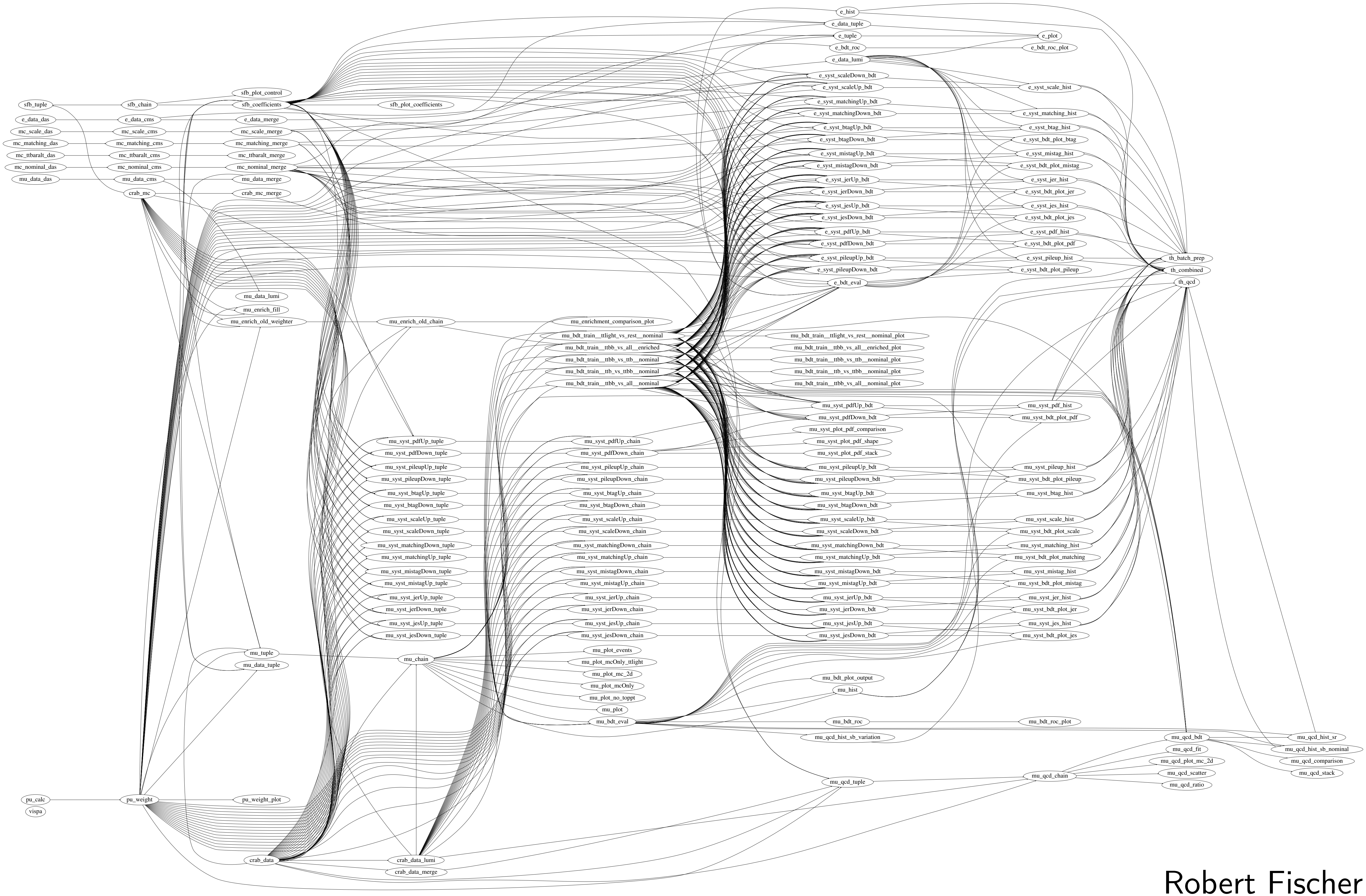
- **Reproducibility**: When a M.Sc. / PhD / Postdoc leaves, ...
  - can someone else run the analysis?
  - is there a loss of information? Is a new *framework* required?
    - ▹ Dependencies often only exist in the physicists head!

- Our background: ttH, ttbb, HH analyses at CMS
  - Large & complex analysis workflows
  - Multiple MVA techniques (DNNs, BDTs, Matrix element method)



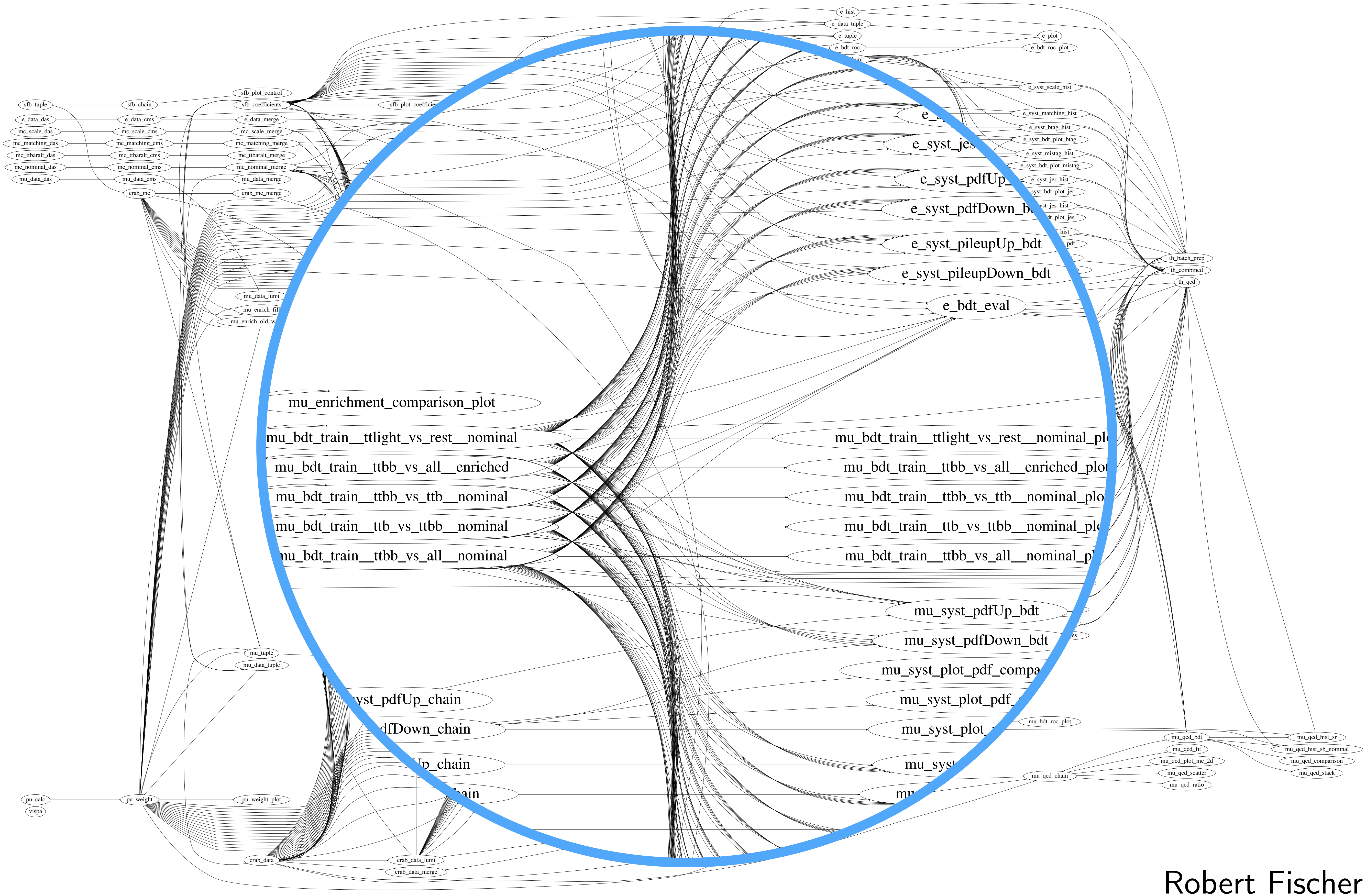
Example: ttbb cross section measurement

Marcel Rieger



Robert Fischer

RWTHAACHEN UNIVERSITY
Marcel Rieger



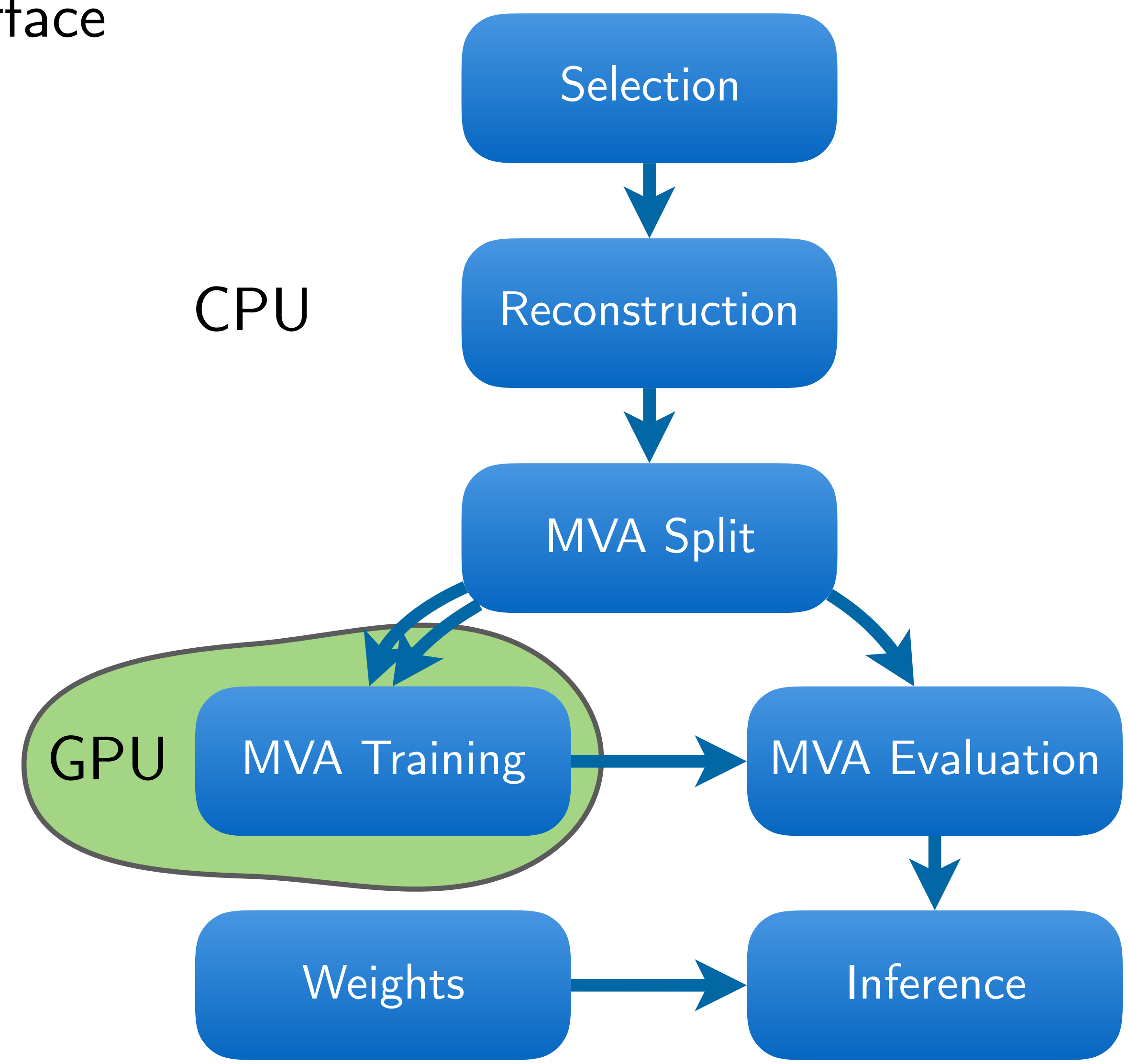
Robert Fischer

RWTH AACHEN UNIVERSITY
Marcel Rieger

Abstraction: analysis workflows

- Workflow, decomposable into particular workloads
- Workloads related to each other by common interface
  - In/outputs define directed data flow
- Alter default behavior via parameters
- Computing resources
  - Run location (CPU, GPU, WLCG, ...)
  - Storage location (local, dCache, EOS, ...)
- Software environment
- Collaborative development and processing
- Reproducible intermediate and final results

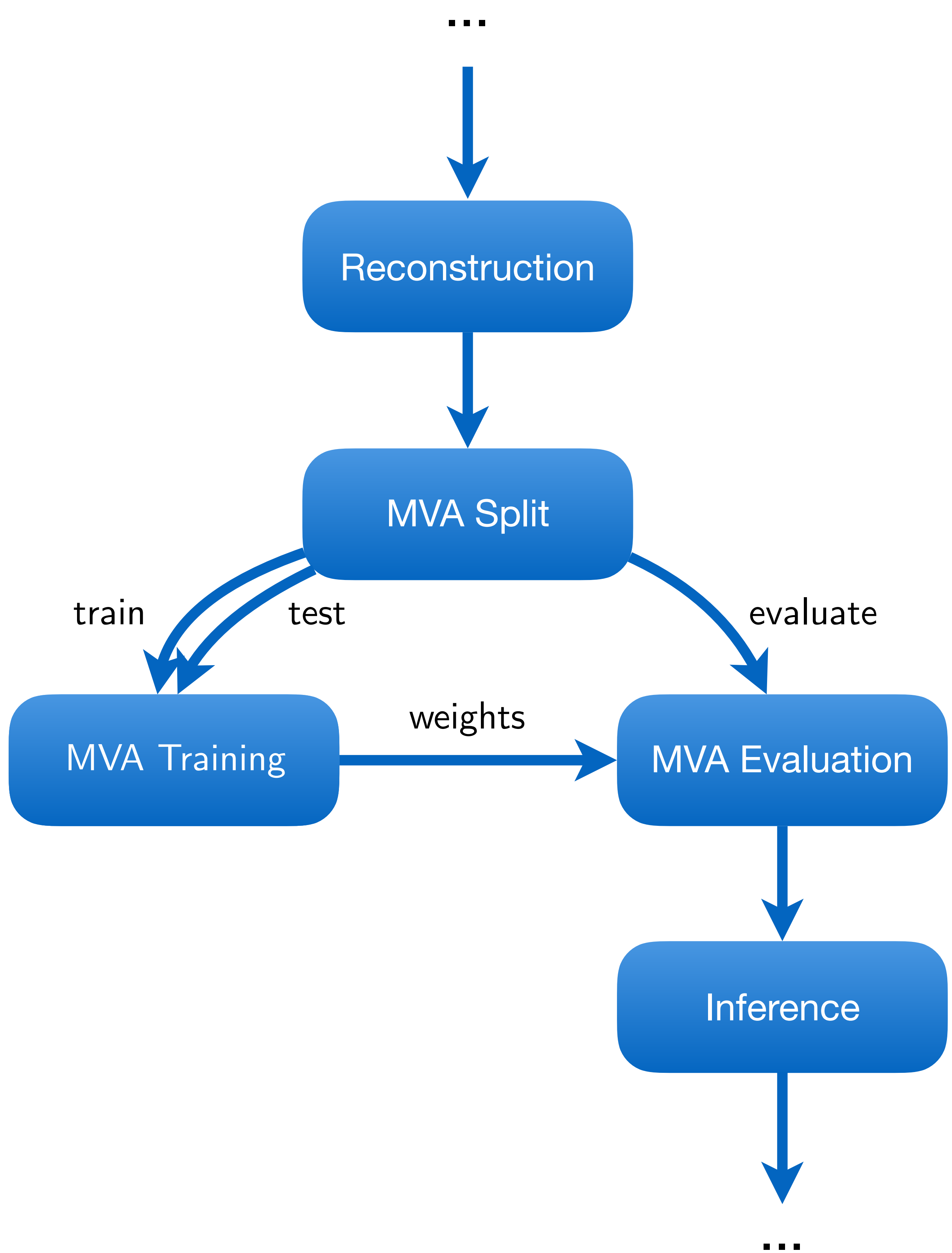
Example



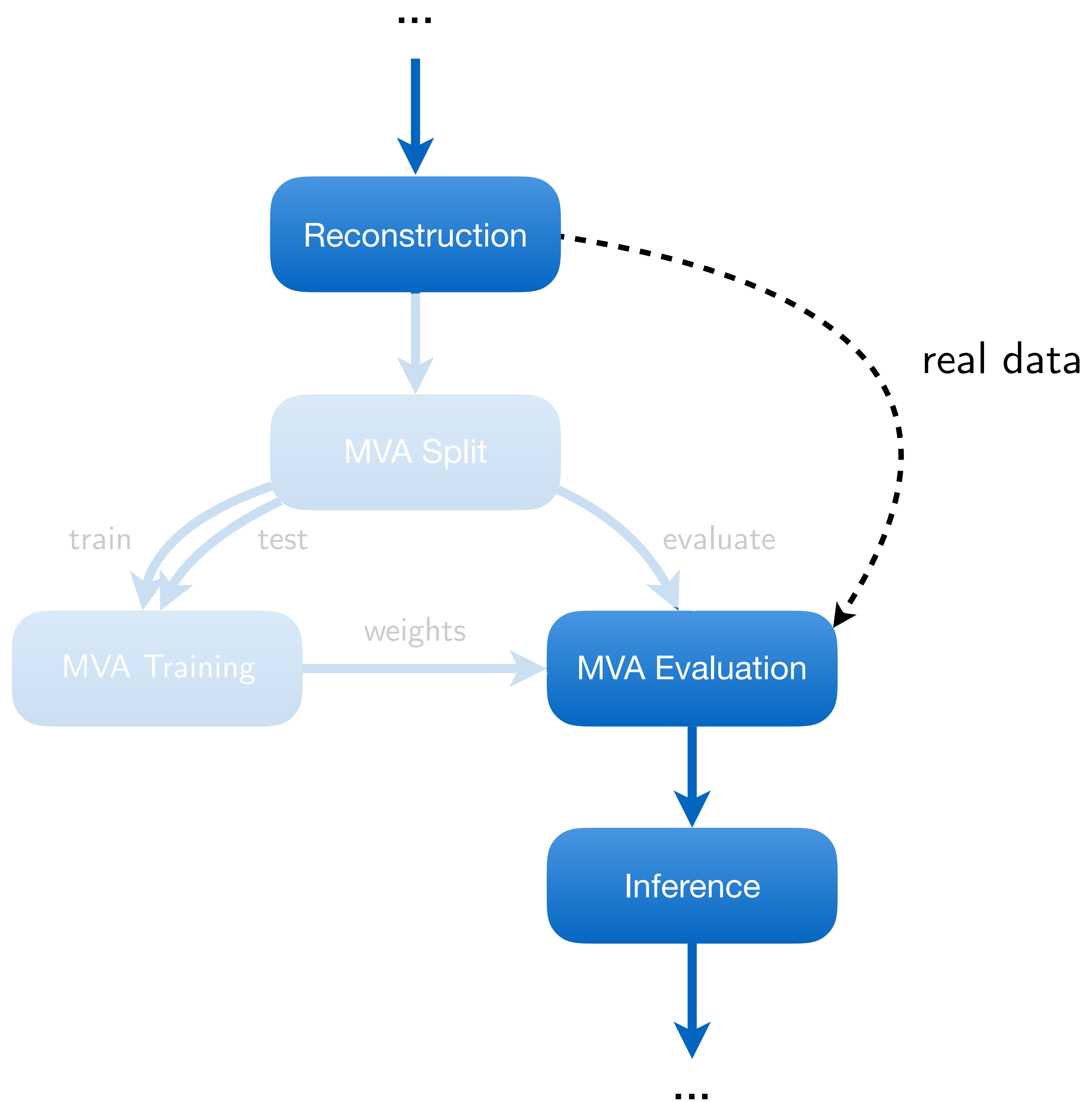
→ Reads like a checklist for analysis workflow management

A typical example: deep learning workflow

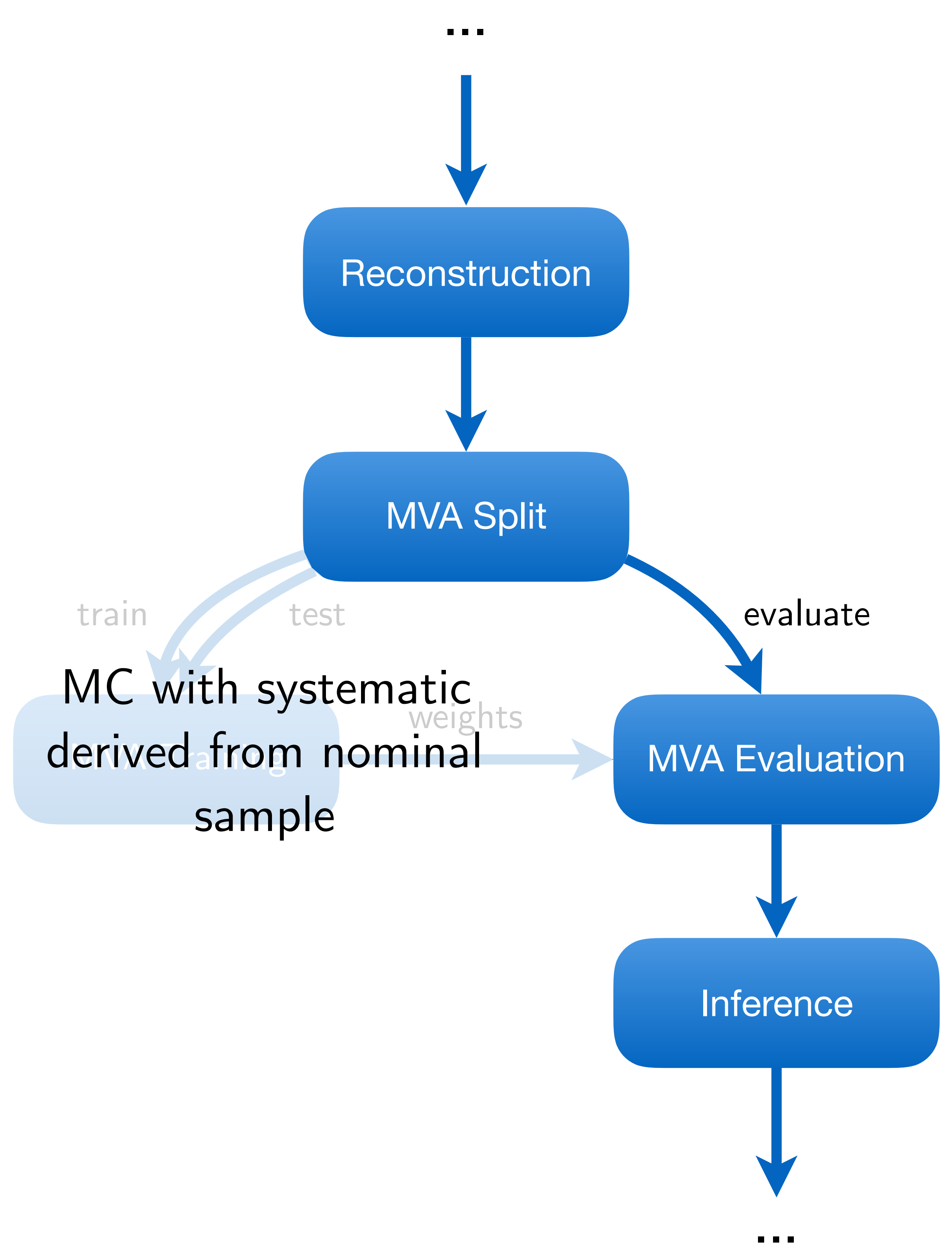
RWTHAACHEN UNIVERSITY
Marcel Rieger



Marcel Rieger

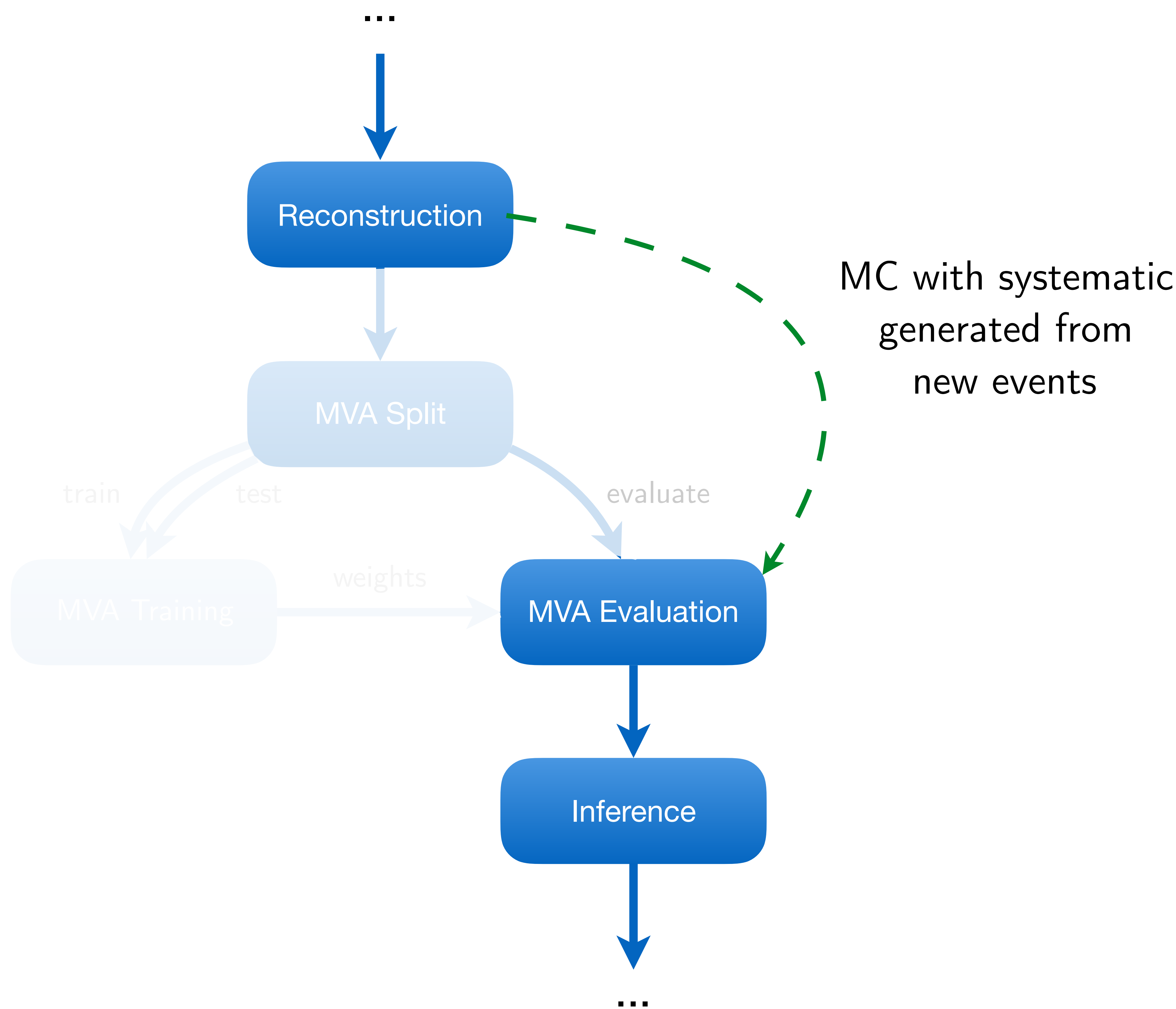


RWTHAACHEN UNIVERSITY
Marcel Rieger



A typical example: deep learning workflow

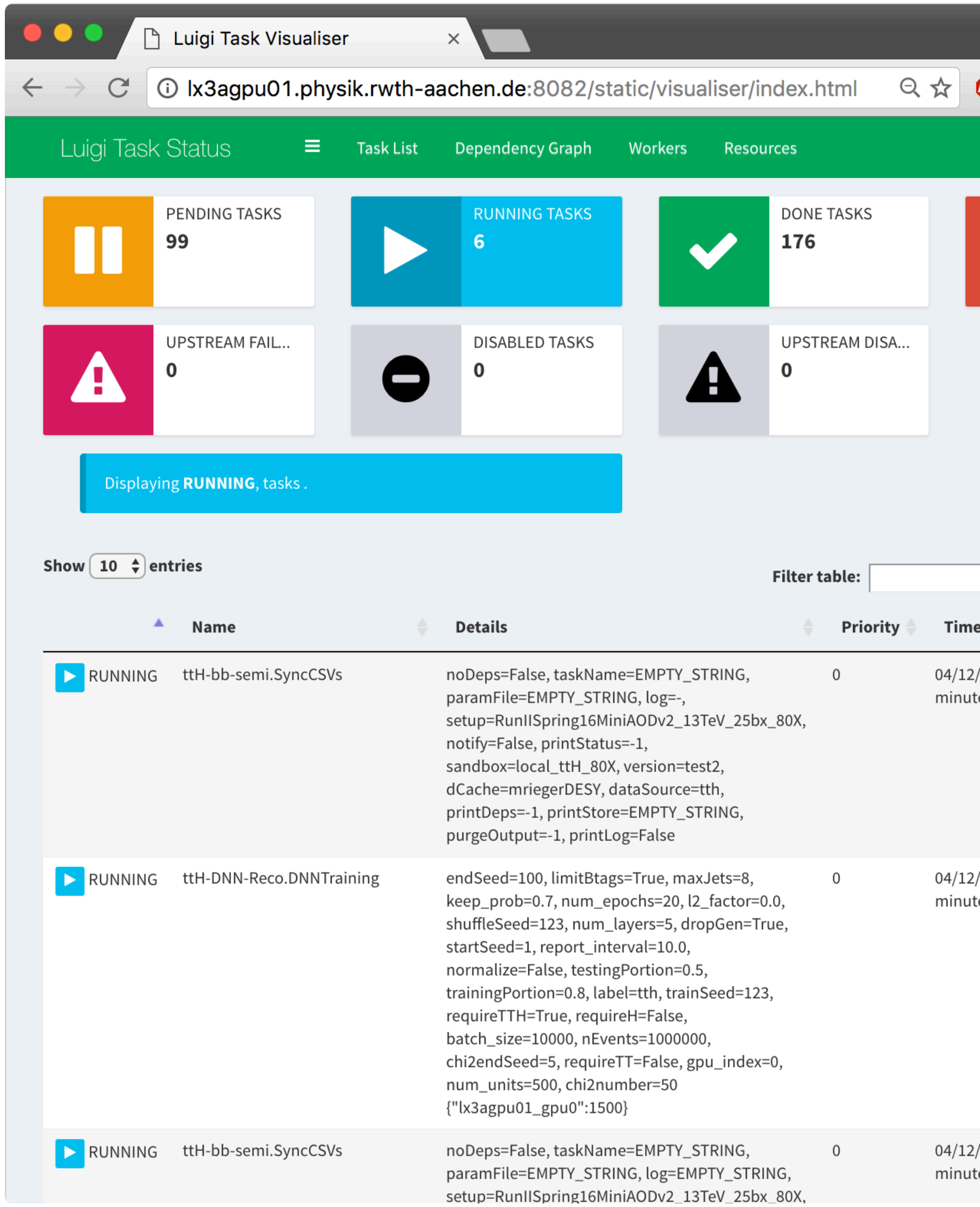
RWTH AACHEN UNIVERSITY
Marcel Rieger



Luigi

RWTH AACHEN UNIVERSITY
Marcel Rieger

- Python package for building complex pipelines
- Development started at Spotify, now open-source and community-driven



<u>Building blocks</u>

1. Workloads defined as **Task** classes
2. Tasks **require** other tasks & output **Targets**
3. **Parameters** customize tasks and control behavior

- Web interface, error handling, command line tools, task history, collaborative features, ...
- github.com/spotify/luigi

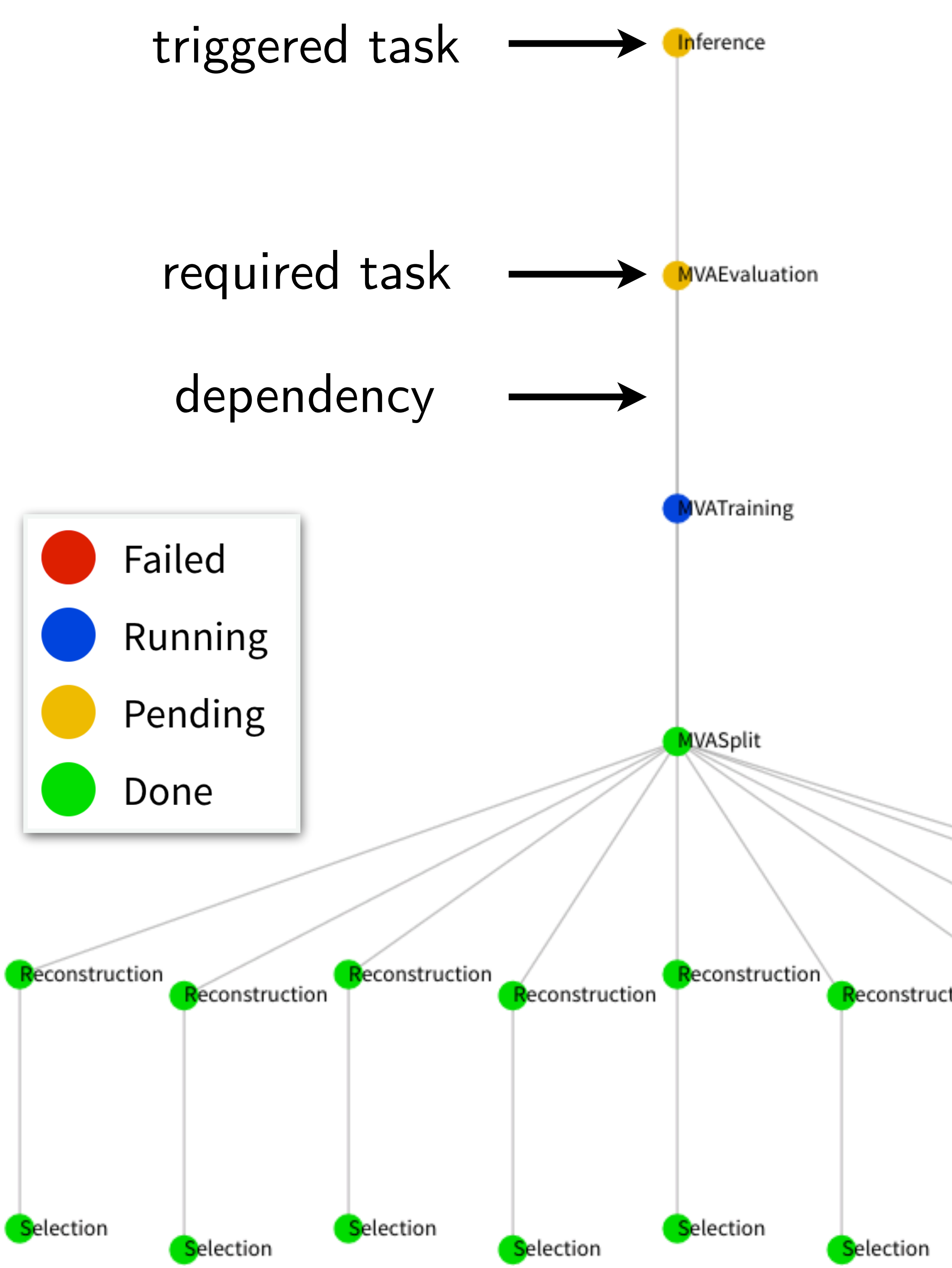
```python
# reco.py

import luigi

from analysis.ttH.tasks import Selection

class Reconstruction(luigi.Task):

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


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

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

    def run(self):
        inp = self.input()  # this is the "output()" of Selection

        # do whatever a reconstruction does
```

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

RWTH AACHEN UNIVERSITY
Marcel Rieger

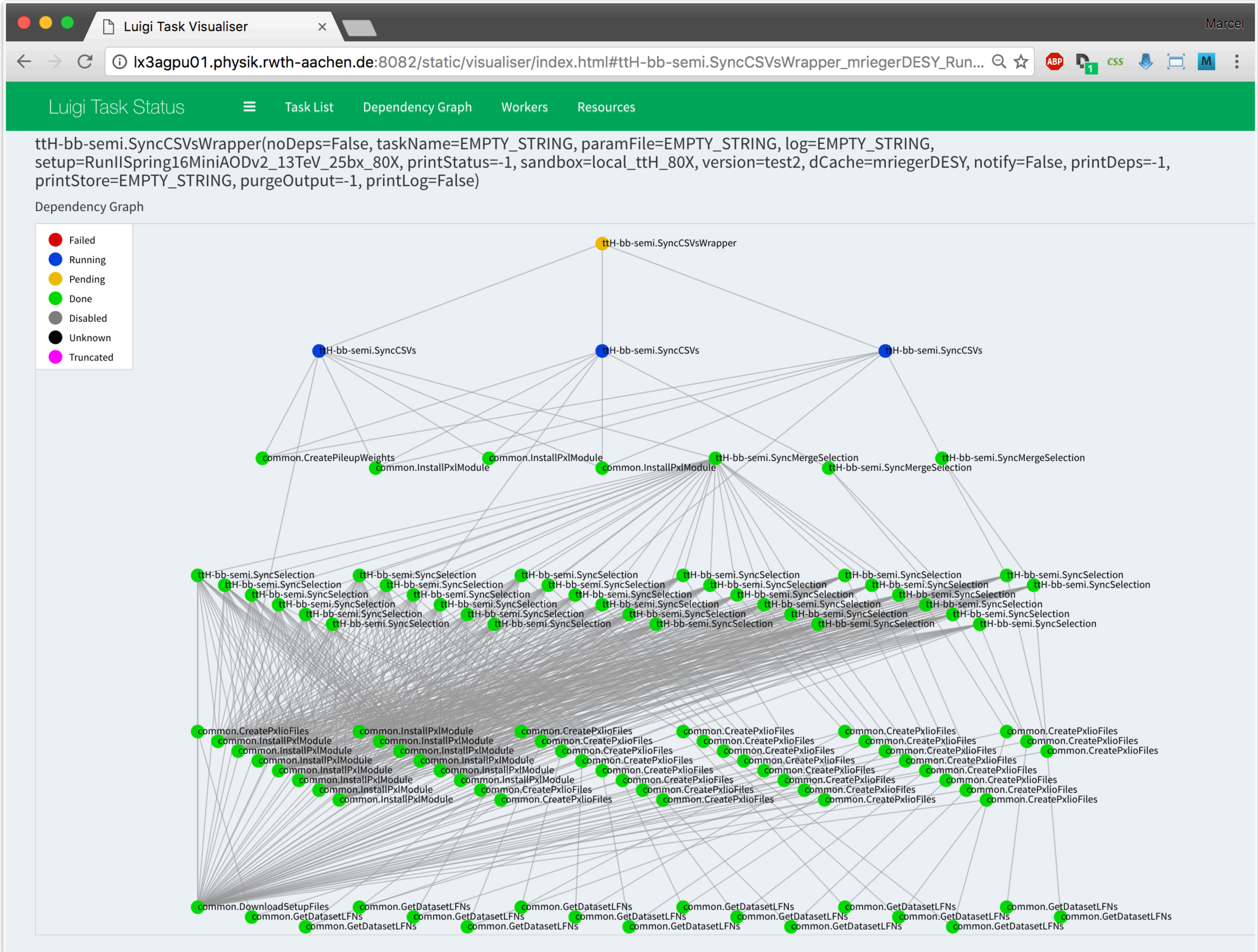
- Luigi's execution model is make-like

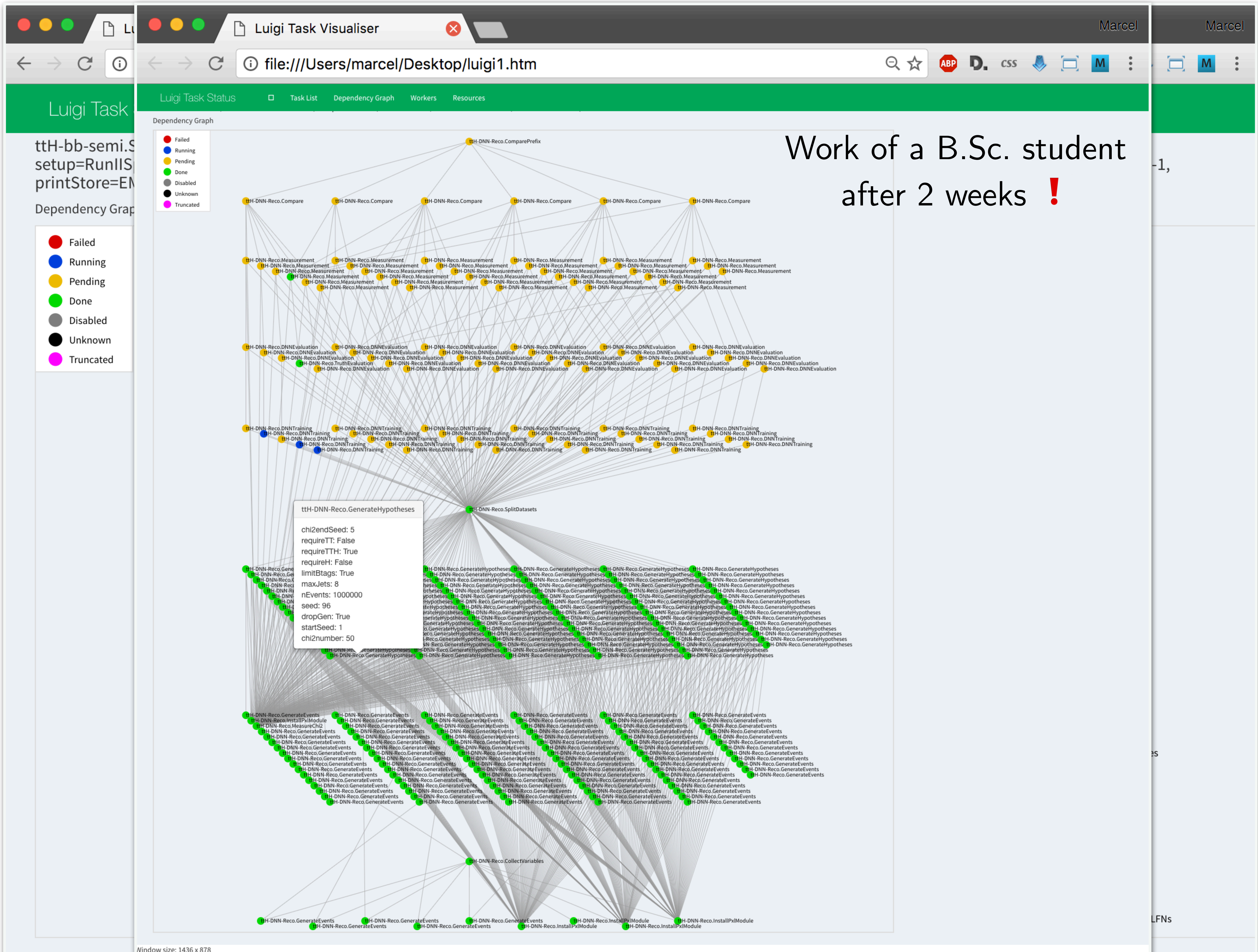
1. Create dependency tree for triggered task
2. Determine tasks to actually run:
   - Walk through tree (top-down)
   - For each path, stop when all output targets of a task exist



- Only processes what is really necessary
- Error handling & automatic re-scheduling
- Clear & scalable through simple structure

RWTHAACHEN UNIVERSITY
Marcel Rieger





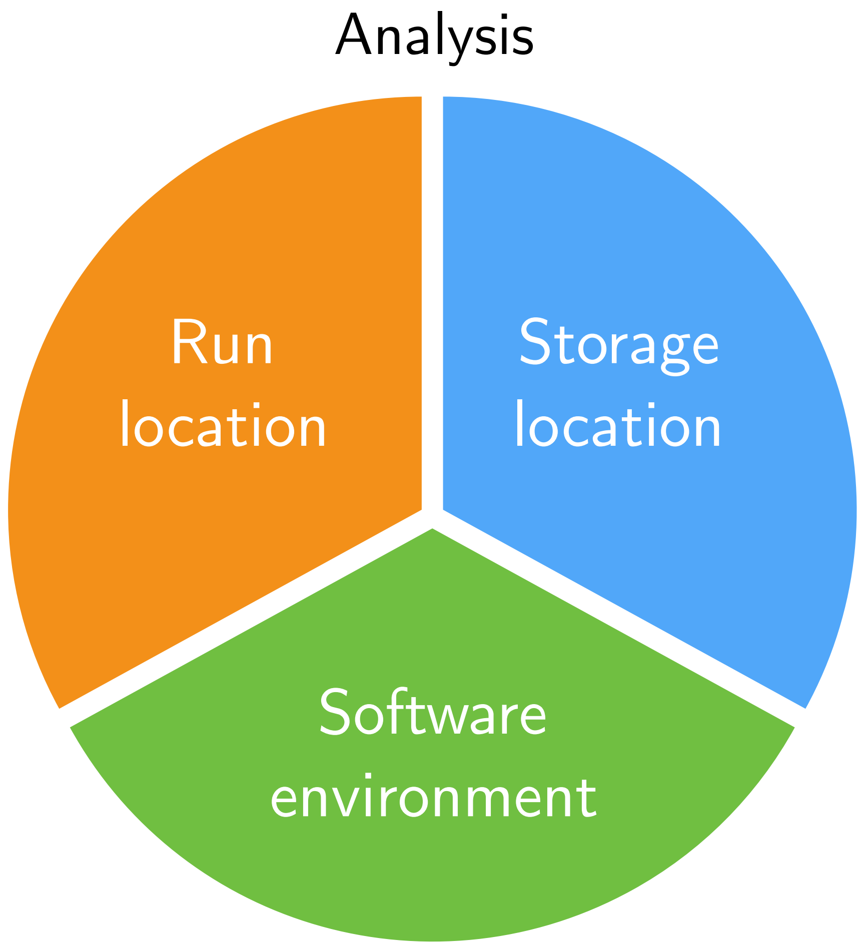
Work of a B.Sc. student after 2 weeks !

- law: layer **on top** of *luigi* (i.e. it does not replace *luigi*)

- Development follows 2 main goals:

  1. Scalability on HEP infrastructure (but not limited to)

  2. Decoupling of **run locations**, **storage locations** & **software environments**
     - ▹ No fixation on dedicated resources
     - ▹ All components interchangeable



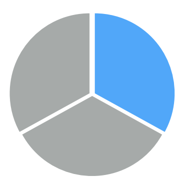
- Provides a toolbox to follow an **analysis design pattern**
  - ▪ No constraint on language or data structures
  - → Not a framework!

law features (1)

1. Job submission

- Idea: submission built into tasks, **no need to write extra code**
- Currently supported job systems: HTCondor, LSF, gLite, ARC
  - Backend not hard-coded, selectable at runtime
- Mandatory features
  - Automatic resubmission, dashboard interface

2. Remote targets



- Idea: work with remote files **as if they were local**
- Build on top of GFAL2 Python bindings
  - Supports all WLCG protocols (dCache, EOS, XRootD, CERNBox, ...) + Dropbox
- Mandatory features
  - Automatic retries, local caching

example

```
target = DCacheTarget("/path/to/file.txt")

with target.open("w") as f:
    f.write("some result")
```

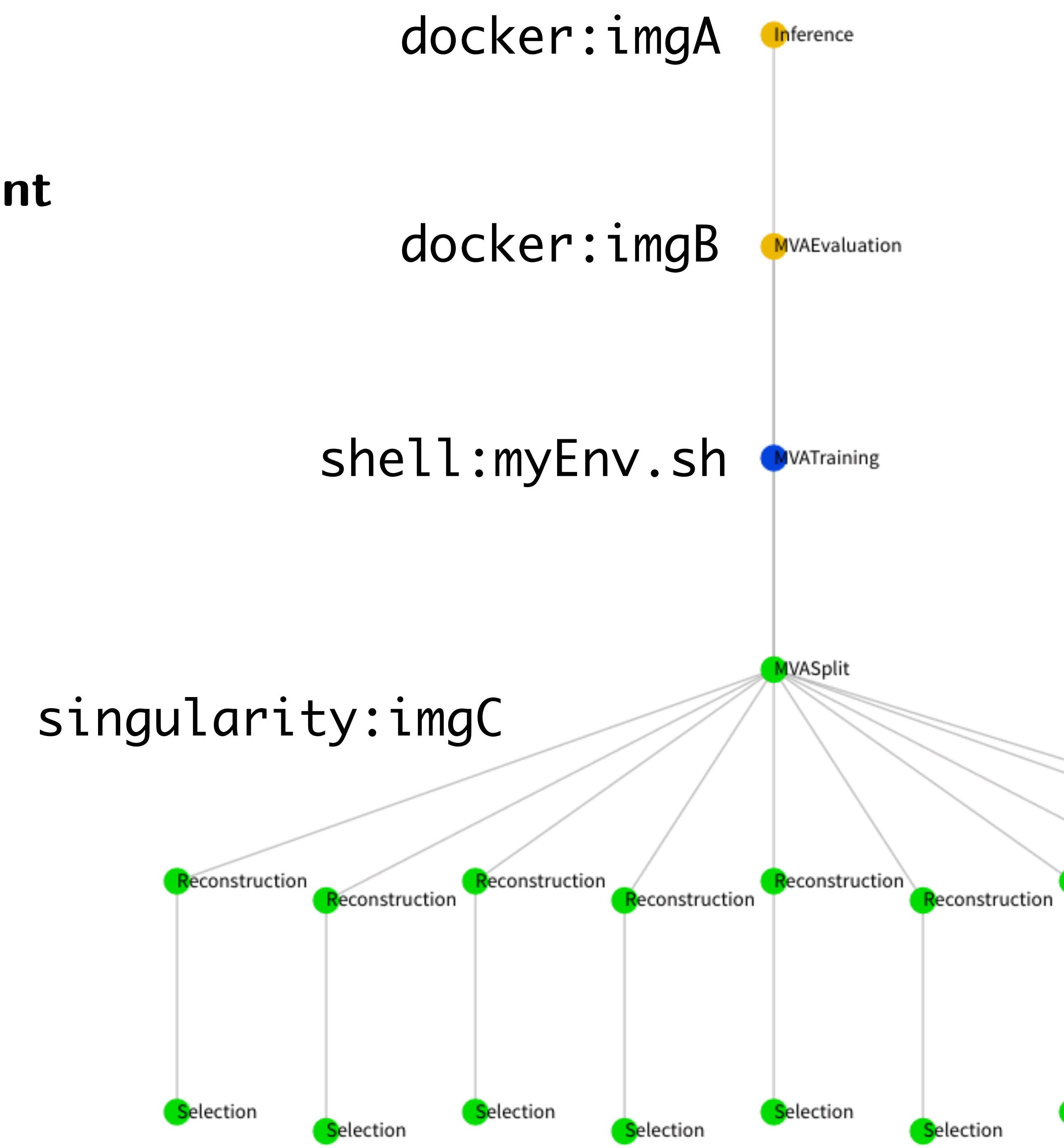
Marcel Rieger

3. Environment sandboxing

- Diverging software requirements between typical workloads is a great feature / challenge / problem

- Introduce sandboxing:
  - Run entire task in **different environment**

- Existing sandbox implementations:
  - Sub-shell with init file
  - Docker images
  - Singularity images



Singularity

RWTH AACHEN UNIVERSITY
Marcel Rieger

```python
# reco.py

import luigi

from analysis.ttH.tasks import Selection

class Reconstruction(luigi.Task):

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


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

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

    def run(self):
        inp = self.input()  # this is the "output()" of Selection

        # do whatever a reconstruction does
```

- [x] luigi task
- [ ] law task
- [ ] Run on grid CE
- [ ] Store on grid SE
- [ ] Run in docker

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

RWTH AACHEN UNIVERSITY
Marcel Rieger

```python
# reco.py

import luigi
import law
from analysis.ttH.tasks import Selection

class Reconstruction(law.Task):

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


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

    def output(self):
        return law.LocalFileTarget("reco_%s.root" % self.dataset)

    def run(self):
        inp = self.input()  # this is the "output()" of Selection

        # do whatever a reconstruction does
```

- [x] luigi task
- [x] law task
- [ ] Run on grid CE
- [ ] Store on grid SE
- [ ] Run in docker

```
> law run Reconstruction --dataset ttJets
```

RWTH AACHEN UNIVERSITY
Marcel Rieger

```python
# reco.py

import luigi
import law
from analysis.ttH.tasks import Selection

class Reconstruction(law.Task, law.GLiteWorkflow):

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


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

    def output(self):
        return law.LocalFileTarget("reco_%s.root" % self.dataset)

    def run(self):
        inp = self.input()  # this is the "output()" of Selection

        # do whatever a reconstruction does
```

- ☑ luigi task
- ☑ law task
- ☑ Run on grid CE
- ☐ Store on grid SE
- ☐ Run in docker

```
> law run Reconstruction --dataset ttJets --workflow glite
```

RWTH AACHEN UNIVERSITY
Marcel Rieger

```python
# reco.py

import luigi
import law
from analysis.ttH.tasks import Selection

class Reconstruction(law.Task, law.GLiteWorkflow):

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


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

    def output(self):
        return law.WLCGFileTarget("reco_%s.root" % self.dataset)

    def run(self):
        inp = self.input()  # this is the "output()" of Selection

        # do whatever a reconstruction does
```

- ☑ luigi task
- ☑ law task
- ☑ Run on grid CE
- ☑ Store on grid SE
- ☐ Run in docker

```
> law run Reconstruction --dataset ttJets --workflow glite
```

RWTH AACHEN UNIVERSITY
Marcel Rieger

```python
# reco.py

import luigi
import law
from analysis.ttH.tasks import Selection

class Reconstruction(law.Task, law.GLiteWorkflow):

    dataset = luigi.Parameter(default="ttH_bb")
    sandbox = "docker::rootproject/root-ubuntu16"

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

    def output(self):
        return law.WLCGFileTarget("reco_%s.root" % self.dataset)

    def run(self):
        inp = self.input()  # this is the "output()" of Selection

        # do whatever a reconstruction does
```

- ☑ luigi task
- ☑ law task
- ☑ Run on grid CE
- ☑ Store on grid SE
- ☑ Run in docker

```
> law run Reconstruction --dataset ttJets --workflow glite
```

Summary

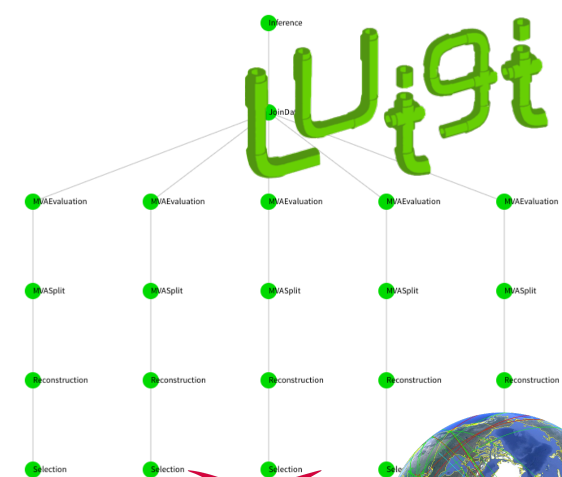
- HEP analyses likely to increase in scale and complexity
  - Analysis workflow management **essential** for success of future measurements
  - Need for toolbox providing a design pattern, **not a framework**

- Luigi is able to model even complex workflows
- Law adds convenience & scalability in the HEP context

- **All** information transparently encoded in tasks, targets & dependencies
- Analysis preservation out-of-the-box

- github.com/riga/law, law.readthedocs.io

law
luigi analysis workflow



HTCondor High Throughput Computing

WLCG

Singularity

docker

Backup

- What is a *framework*?
  - → Bash scripts, python tools, crab configs, CMSSW modules, magic
  - → Connections mostly exist in the physicists head

- Documentation?
  - → Not the most beloved hobby in the physics community

- When a M.Sc. / PhD / Postdoc leaves ...
  - → Can someone else run the analysis?
  - → Is this information lost? Is a new framework required?

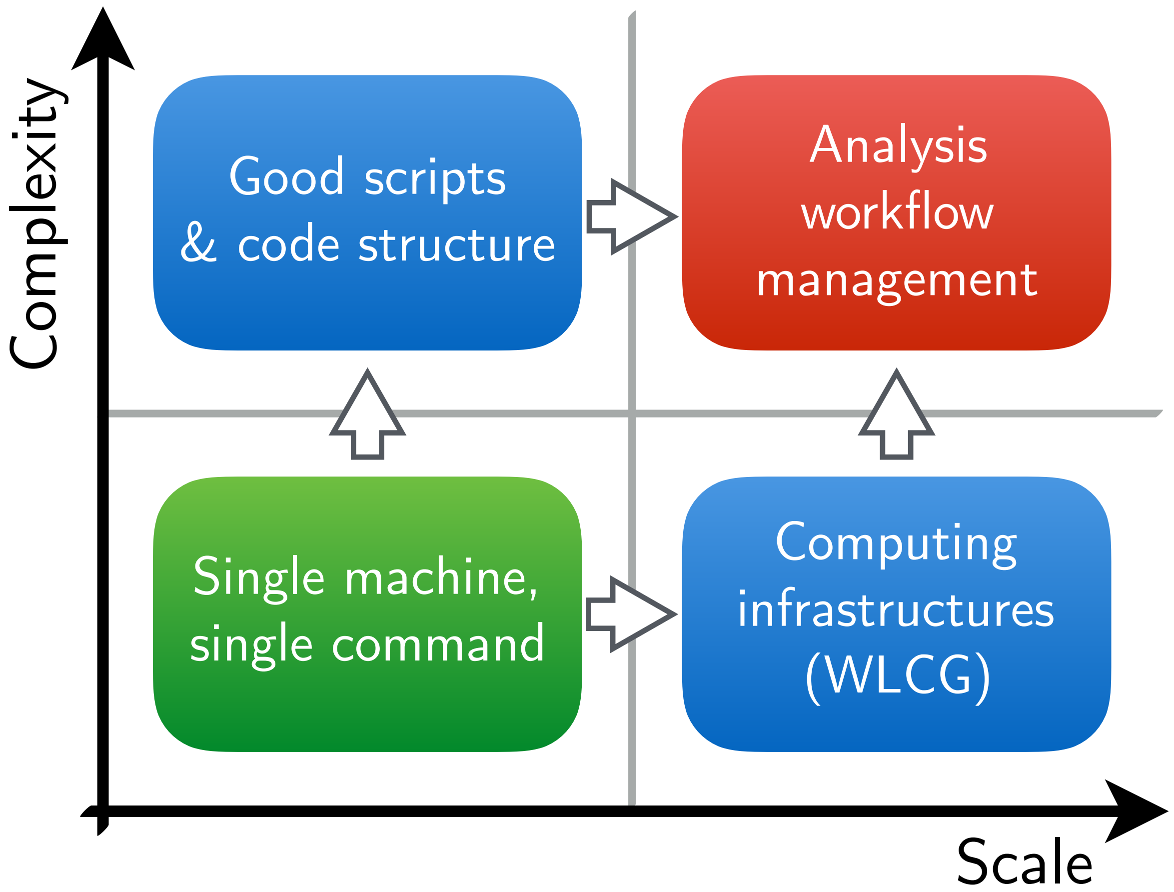
- Does execution dictate code design?
  - → Does the analysis depend on where it runs?

- From *my* experience: ⅔ of time required for technicalities, ⅓ for physics
  - → Physics output doubled if it was the other way round?

Marcel Rieger

- Scale: measure of resource consumption and amount of data
- Complexity: measure of granularity and inhomogeneity of workloads

- Future analyses likely to be large and complex, bottlenecks:
  - Undocumented structure & requirements between workloads, only exists in the physicist's head
  - Bookkeeping of data, versions, ...
  - Manual execution/steering of jobs
  - Error-prone & time-consuming



→ Analysis workflow management essential for future measurements!

Existing WMS: MC production

Marcel Rieger

Tailored systems

- Structure known in advance
- Workflows static & recurring
- One-dimensional design
- Special infrastructures
- Homogeneous software requirements

→ Requirements for HEP analyses mostly orthogonal

Existing WMS: MC production

RWTH AACHEN UNIVERSITY
Marcel Rieger

Tailored systems

- Structure known in advance
- Workflows static & recurring
- One-dimensional design
- Special infrastructures
- Homogeneous software requirements

Wishlist for end-user analyses

- Structure "iterative", a-priori unknown
- Dynamic workflows, fast R&D cycles
- Tree design, arbitrary dependencies
- Incorporate existing infrastructure
- Use custom software, everywhere

→ Requirements for HEP analyses mostly orthogonal

RWTH AACHEN UNIVERSITY
Marcel Rieger

	Existing WMS e.g. MC Management	Generic Analysis WMS
Development Process	final objective known in advance	iterative, final composition a priori unknown
Workflow Structure	chain structure, mostly one-dimensional	tree structure, arbitrarily branched
Evolution	static over time, recurrent execution	dynamic, fast R&D cycles
Infrastructure	specially tailored, e.g. storage systems, DBs	incorporate existing, quickly adapt to changes
Applicability	tuned to particular use case	flexible, able to model every possible workflow

→ Existing WMS highly specialized for designated use case

→ Requirements for HEP analyses mostly orthogonal

order: structure external HEP data

Marcel Rieger

- Pythonic class collection to order "soft", external HEP data
  - physics processes & cross sections
  - campaigns & datasets
  - channels & categories
  - systematics & statistical models
- Some data could be centrally managed, some is analysis specific
- More info in the intro.ipynb notebook
- Use as data backend:

```
> law run Selection --dataset ttH125_bb --...
```

```
dataset_ttH = Dataset("ttH125_bb", 100,
    keys     = "/ttHTobb_M125_TuneCUETP8M2_.../.../MINIAODSIM",
    nFiles   = 119,
    nEvents  = 3845992
)

process_ttH125 = Process("ttH125", 100,
    label  = r"$t\bar{t}H$",
    xsecs  = { 13: Number(0.5071, (0.058, 0.092)) }
)

dataset_ttH.add_process(process_ttH125)
```

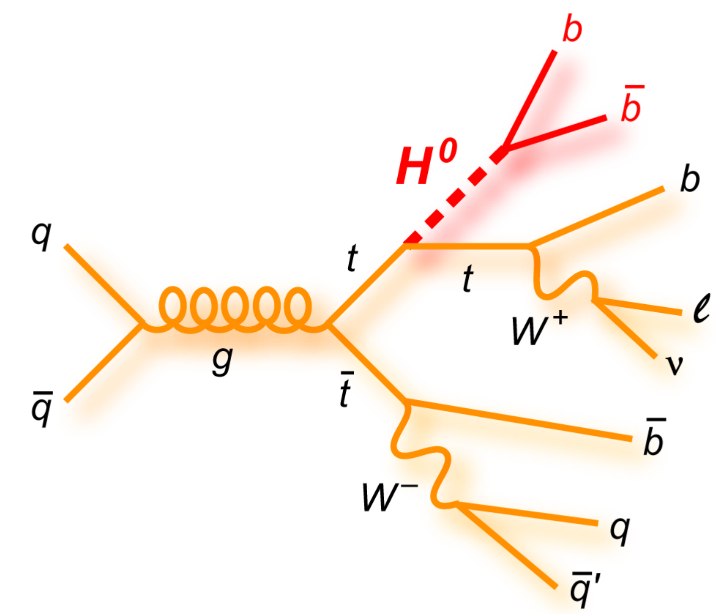
Marcel Rieger

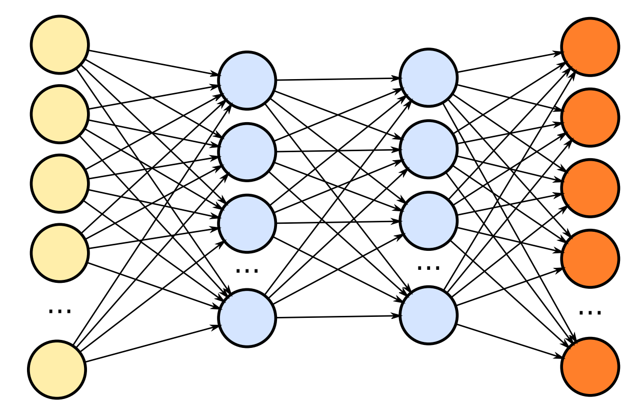
1. Toolbox providing building blocks for analyses
   - → Design pattern, **not a framework** (no constraint on language or data structure)
   - → Full decoupling of run location, storage location and software environment
2. **All** information transparently encoded in tasks, targets & dependencies
   - → Results **reproducible** by developer, groups, collaboration, ...
   - → Analysis preservation out-of-the-box
3. `make`-like execution across distributed resources
   - → Reduces overhead of manual management
   - → Improves cycle times & error-proneness

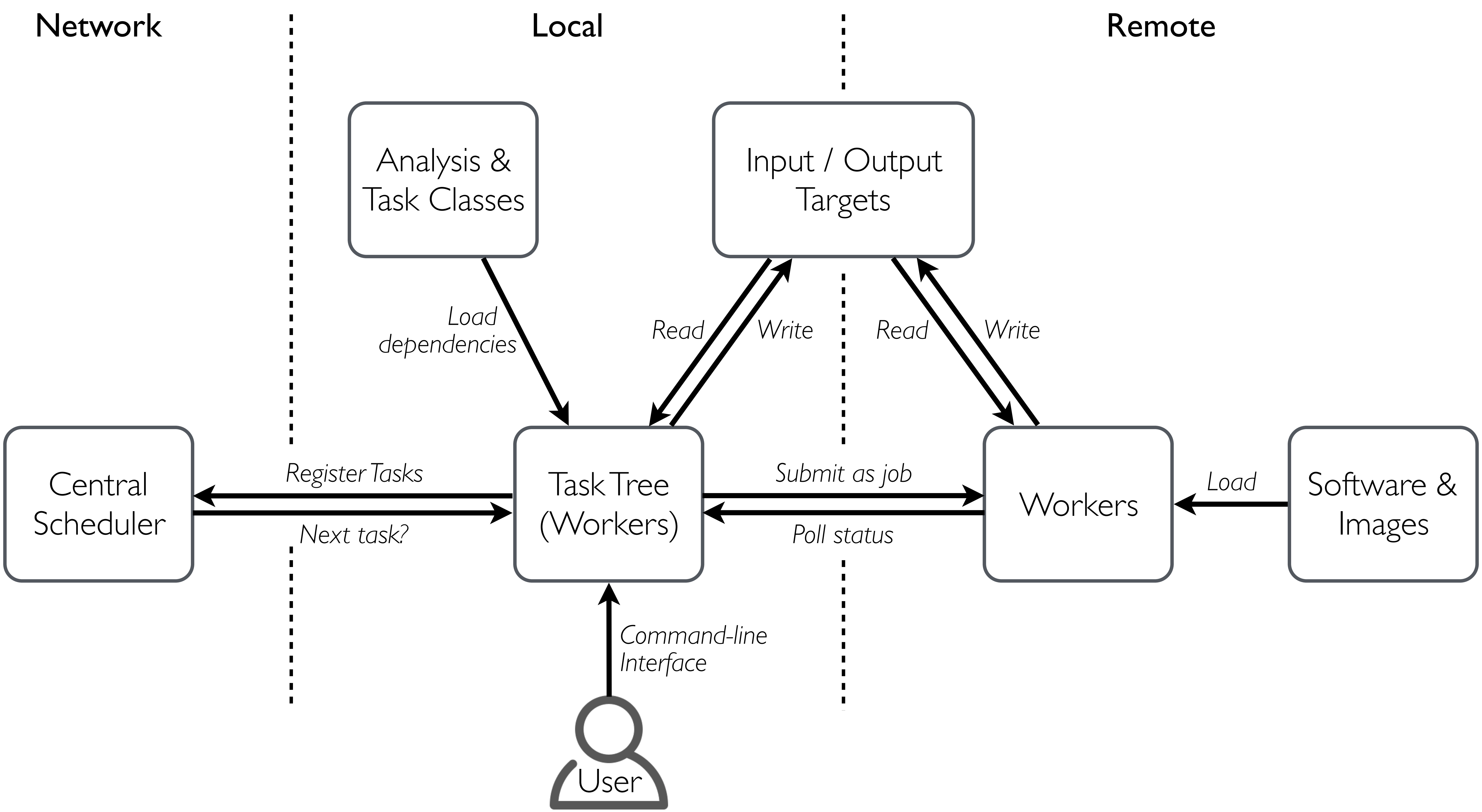
→ Changed paradigm from executing to defining an analysis
→ Move focus back to physics

RWTH AACHEN UNIVERSITY
Marcel Rieger

- Large-scale:
  - ~80 TB of storage, ~500k tasks
- Complex:
  - DNNs/BDTs/MEM, ~70 systematic variations
- Run locations:
  - 7 CEs, local machines, GPU machines
- Storage locations:
  - 2 SEs (dCache), local disk, Dropbox, CERNBox
- Clear allocation of duties in group
- Entire analysis operable by everyone

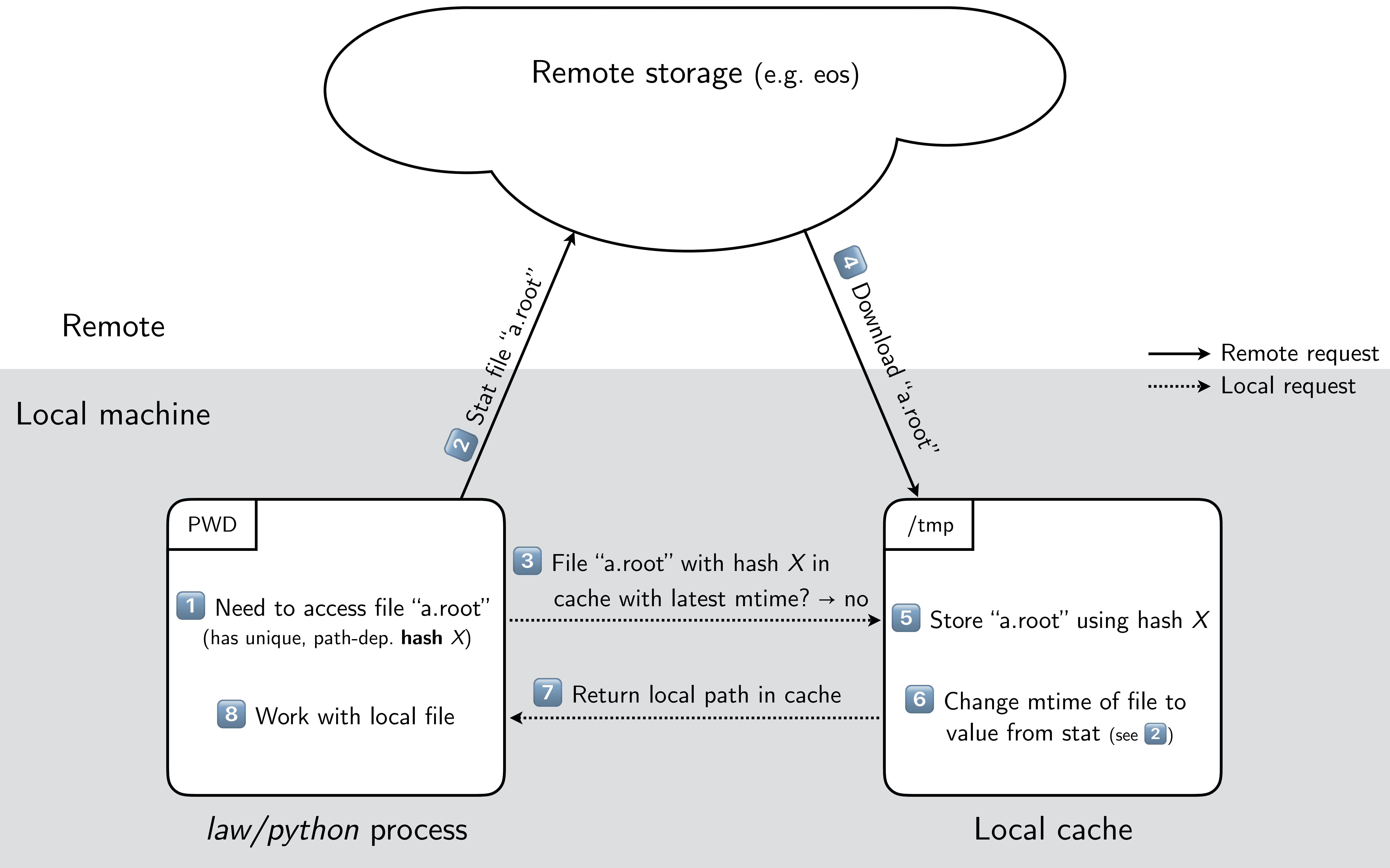





Network
Local
Remote
Analysis & Task Classes
Input / Output Targets
Load dependencies
Read
Write
Read
Write
Central Scheduler
Register Tasks
Next task?
Task Tree (Workers)
Submit as job
Poll status
Workers
Load
Software & Images
Command-line Interface
User

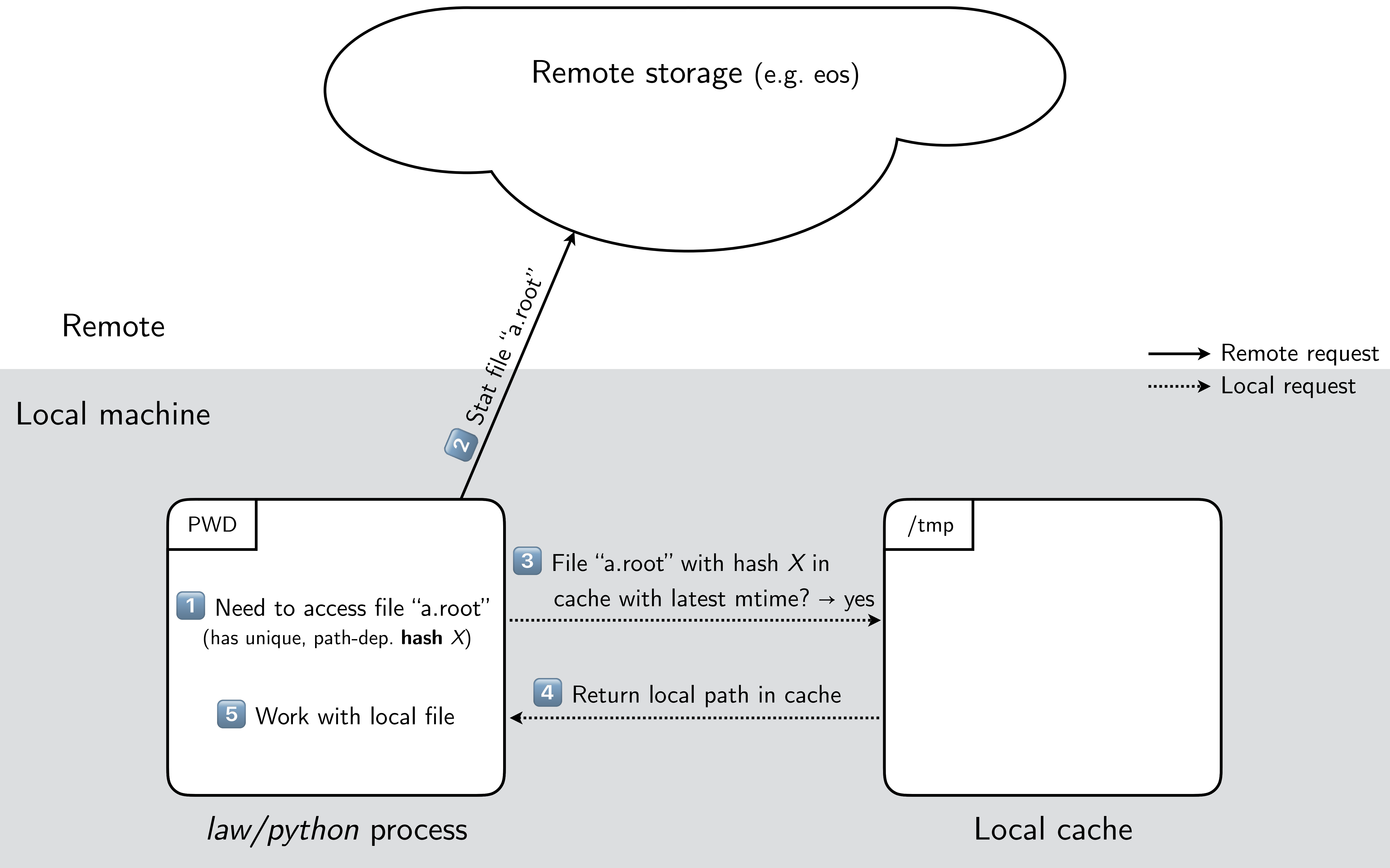
Marcel Rieger

Scenario A: file not cached *yet*



Marcel Rieger

Scenario B: file *already* cached



RWTH AACHEN UNIVERSITY
Marcel Rieger

```
> check status of ttH-bb-semi.Selection(taskName=EMPTY_STRING,
|   - check DCacheFileTarget(path=/analyses/ttH_bb_semi/Select
|     -> absent
|   - check DCacheFileTarget(path=/analyses/ttH_bb_semi/Select
|     -> absent
|   - check SiblingTargetCollection(len=1, threshold=1.0, 0x7f
|     -> absent (0/1)
|   > check status of common.CreatePxlioFiles(taskName=EMPTY_S
|   |   - check DCacheFileTarget(path=/analyses/ttH_bb_semi/Cr
|   |     -> absent
|   |   - check DCacheFileTarget(path=/analyses/ttH_bb_semi/Cr
|   |     -> absent
|   |   - check SiblingTargetCollection(len=1, threshold=1.0,
|   |     -> existent (1/1)
|   |   > check status of common.GetDatasetLFNs(taskName=EMPTY
|   |   |   - check DCacheFileTarget(path=/analyses/ttH_bb_sem
|   |   |     -> existent
|   |   > check status of common.DownloadSetupFiles(taskName=E
|   |   |   - check SiblingTargetCollection(len=7, threshold=1
|   |   |     -> existent (7/7)
|   |   > check status of common.UploadRepo(dCache=marcelDESY,
|   |   |   - check SiblingTargetCollection(len=10, threshold=
|   |   |     -> absent (0/10)
|   |   |   > check status of common.BundleRepo(taskName=EMPTY
|   |   |   |   - check LocalFileTarget(path=/user/public/anal
|   |   |   |     -> absent
|   |   > check status of common.UploadSoftware(dCache=marcelD
|   |   |   - check SiblingTargetCollection(len=10, threshold=
```

law
luigi analysis workflow
github.com/riga/law

- Completeness: 90%
- Missing: documentation, unit tests

order
github.com/riga/order

- Completeness: 95%
- Missing: datacards and luminosity helpers
- Optional: centrally managed processes/campaigns/datasets

scinum
github.com/riga/scinum

- Completeness: 100%
- (scientific numbers w/ uncertainties & gaussian propagation)