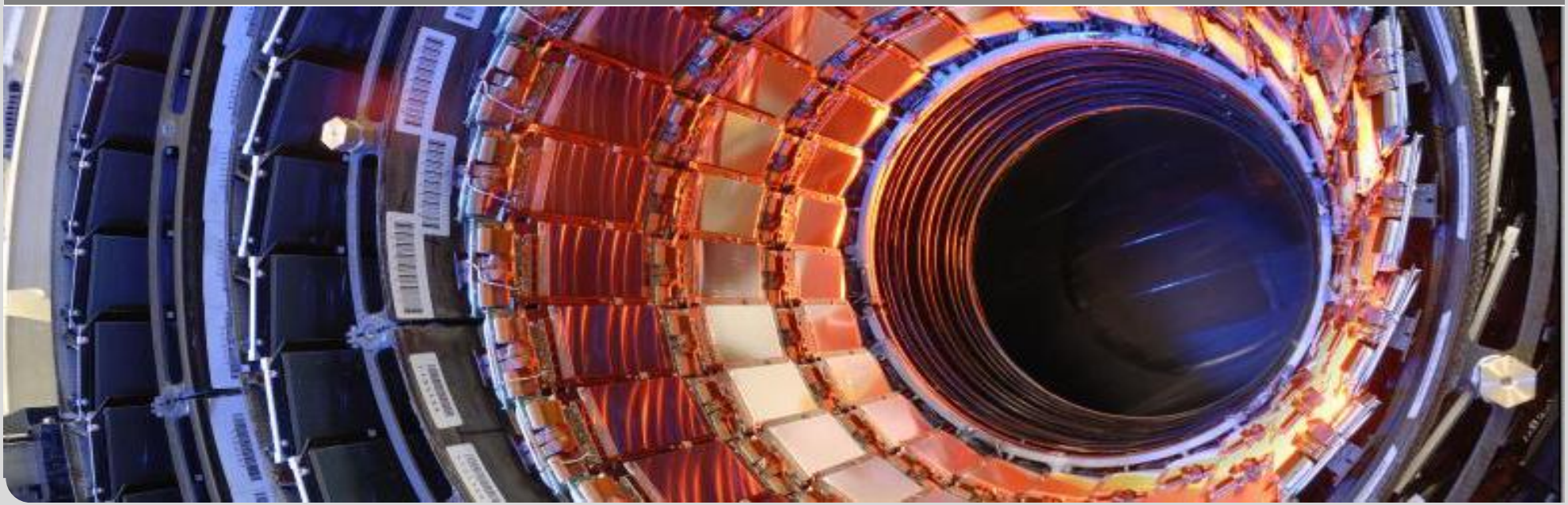
Karlsruher Institut für Technologie

Developing a pipeline for grid production with NNLOJET

Weekly QCD-Meeting

Miguel Santos Correa | 26th April 2018

INSTITUTE FOR EXPERIMENTAL PARTICLE PHYSICS (ETP) · KIT-FACULTY FOR PHYSICS



Goal

- automate workflow of NNLOJET grid production
- reusable for different processes in the future
- lightweight, small overhead
- flexible to different environments and demands

Karlsruher Institut für Technologie

Luigi is a Python package that helps you build complex pipelines of batch jobs

Features:

- dependency resolution
- workflow management
- visualization
- handling failures
- command line integration
- and much more

law
luigi analysis workflow

RWTH AACHEN UNIVERSITY

KIT
Karlsruher Institut für Technologie

law ist built on top of Luigi and adds abstractions for run locations, storage locations and software environments

Features:

- **Remote targets with automatic retries and local caching**
  WebDAV, HTTP, Dropbox, SFTP, all WLCG protocols (srm, xrootd, rfio, dcap, gsiftp, ...)
- **Automatic submission to batch systems from within tasks**
  HTCondor, LSF, gLite, ARC
- **Environment sandboxing, configurable on task level**
  Docker, Singularity, Sub-Shells

Tasks

```
import luigi

class MyTask(luigi.Task):
  param = luigi.Parameter(default = 42)

  def requires(self):
    return SomeOtherTask(self.param)

  def output(self):
    return luigi.LocalTarget(
      '/tmp/foo/bar-{}.txt'.format(self.param)
    )

  def run(self):
    f = self.output().open('w')
    print >>f, 'hello, world'
    f.close()
```

Workflows

```
import law

class MyWorkflow(law.LocalWorkflow):

  def create_branch_map(self):
    return {0: 'foo', 1: 'bar', 2: 'baz'}

  def workflow_requires(self):
    return {'common': SomeCommonDependency.req(self, ...)}

  def requires(self):
    return SomeBranchSpecificRequirement.req(self, ...)

  def output(self):
    return law.LocalFileTarget('some/file_{}.txt'.format(
                                        self.branch))

  def run(self):
    self.output().dump(self.branch_data, formatter='text')
```

Warmup Workflow

```
class Warmup(HTCondorWorkflow):
  channels = luigi.Parameter()

  def create_branch_map(self):
    branchmap = {}
    for i, channel in self.channels:
      branchmap[i] = channel
    return branchmap

  def requires(self):
    return Runcard(
      channel = self.branch_data, warmup = 'true', ...
    )

  def output(self):
    return self.remote_target(
      '{}.{}.{}.warmup.tar.gz'.format(self.process, self.
                                      branch_data, self.name)
    )
```

luigi.cfg

```
[DEFAULT]
name = ZJtriple
process = ZJ
channels = LO R V RRa RRb RV VV
wlcg_path = srm://cmssrm-kit.gridka.de:8443/srm/...
htcondor_accounting_group = cms.jet
htcondor_requirements = (TARGET.ProvidesCPU==true)
htcondor_request_cpus = 1
htcondor_request_memory = 4096
...

[Warmup]
warmup_events = 200000 10000 40000 5000 5000 10000 20000
warmup_iterations = 10 10 10 10 10 10 10
starting_seed = 0
htcondor_request_cpus = 20
htcondor_request_memory = 16384
```

Command Line Interface

```
$ law run FastWarm --print-status -1

print task status with max_depth -1 and target_depth 0

> check status of FastWarm(branch = -1, ...)
|   - check WLCGFileTarget(path=/ZJtriple/FastWarm/
                            htcondor_submission.json,
                            optional)
|     -> absent
|   - check WLCGFileTarget(path=/ZJtriple/FastWarm/
                            htcondor_status.json,
                            optional)
|     -> absent
|   - check TargetCollection(len=14, threshold=0.95)
|     -> existent (14/14)
|
|   > check status of Warmup(...)
|   |   - check ...
```

Command Line Interface

```
$ law run Warmup --local-scheduler --transfer-logs

DEBUG: Checking if Warmup(...) is complete
INFO: Informed scheduler that task
Warmup_warmup_bootstrap__1_False_74e4e0b45d has status
PENDING
INFO: [pid 646590] Worker Worker(salt=845360652, workers=1
                                , host=ekpbms2.ekp.kit.edu,
                                username=mcorrea, pid=646590)
                                 running   Warmup(...)
going to submit 7 htcondor job(s)
submitted 1/7 job(s)
submitted 7 job(s)
16:32:46: all:    7, pending:    7 (  +0),
running:    0 (  +0), finished:    0 (  +0),
retry:    0 (  +0), failed:    0 (  +0)
16:33:46: all: ...
```

Getting Started

- clone repository

```
$ git clone --recurse-submodules
  https://github.com/miguel-sc/nnlo-law-analysis.git
```

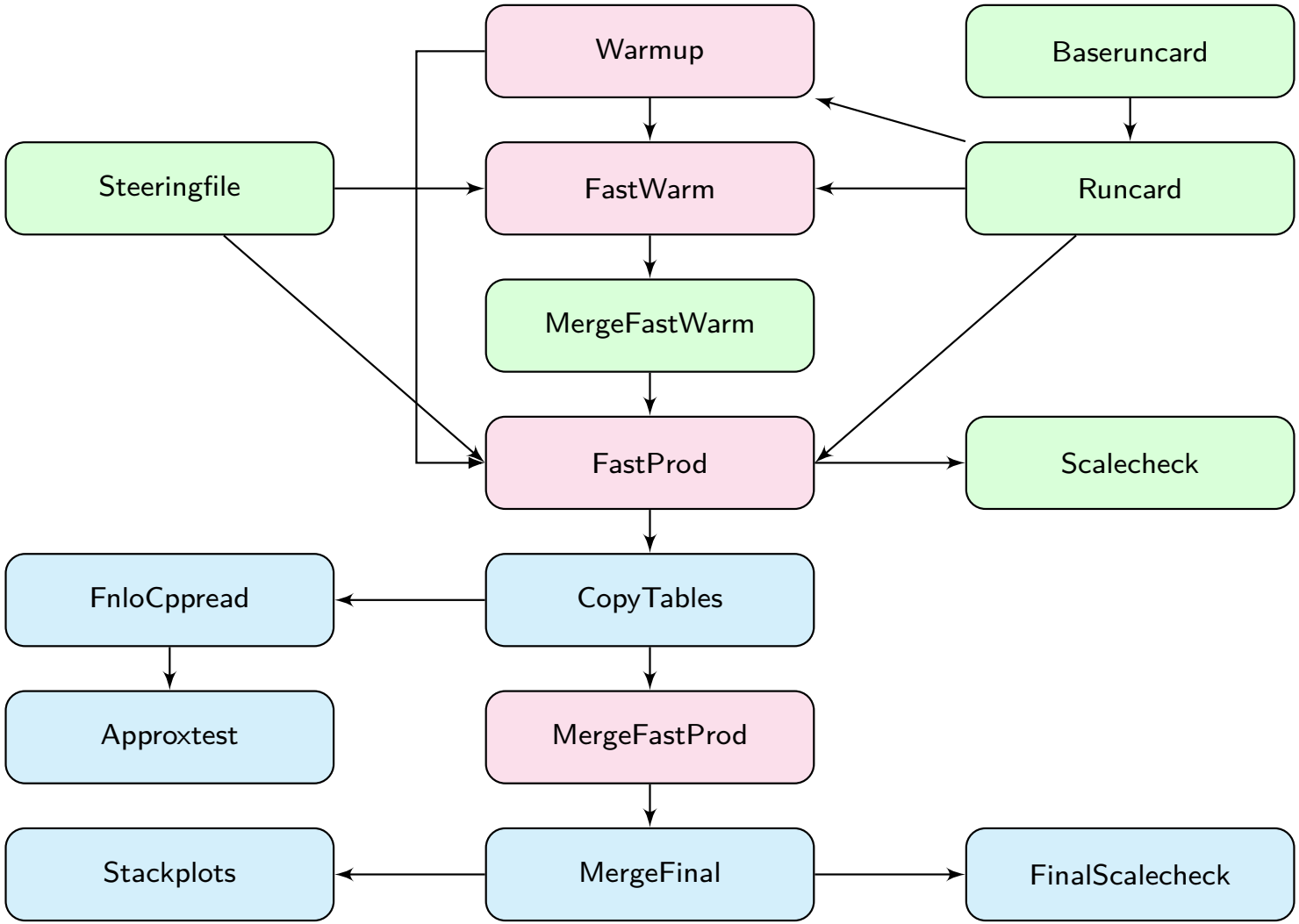
- copy NNLOJET runcard and steeringfile into repository
- insert substitution strings ('@SEED@', ...) into runcard
- edit luigi.cfg
- set environment variables

```
$ source setup.sh
```

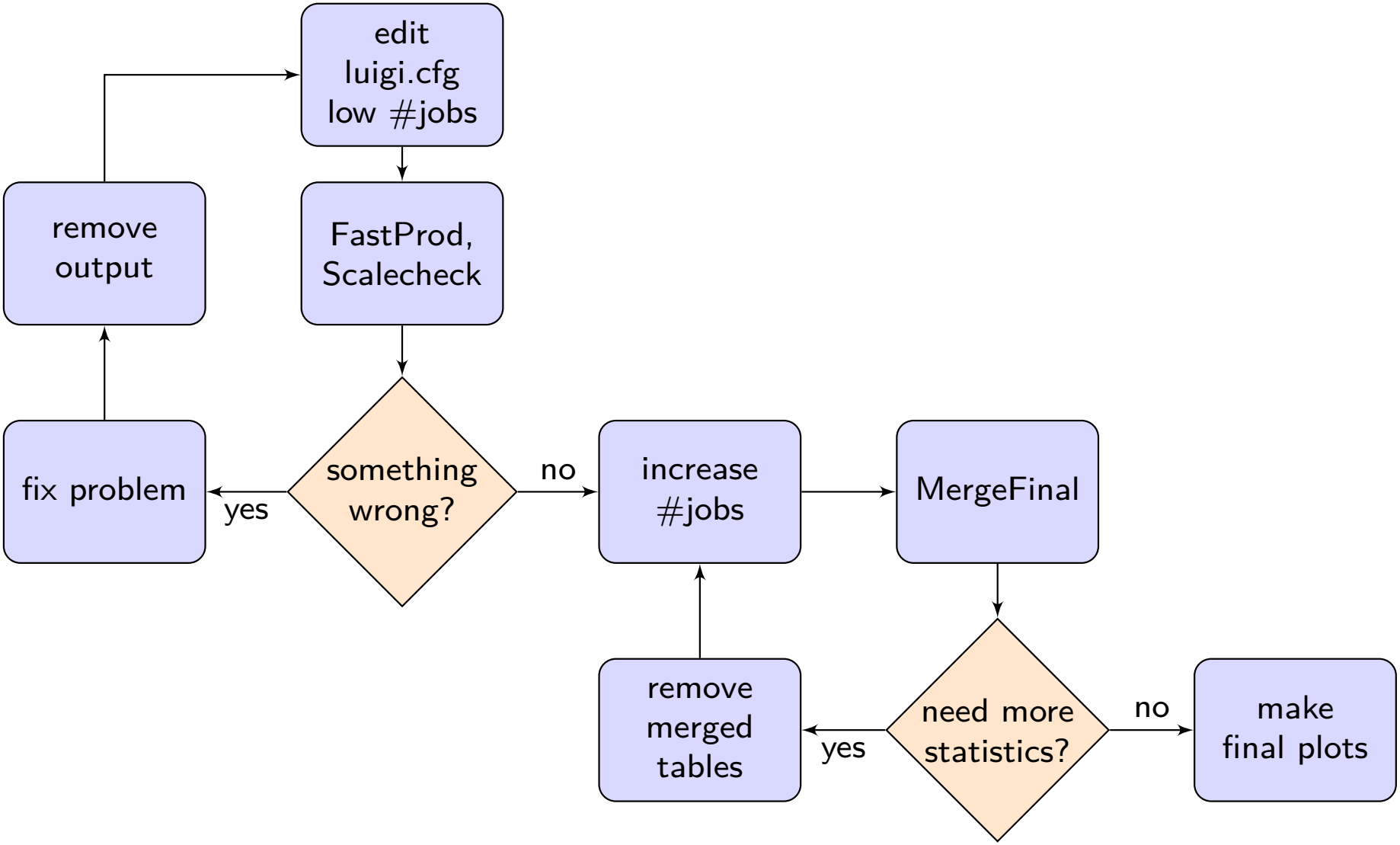
- initialize law

```
$ law db --verbose
```

Dependency Tree (simplified)



Flowchart


edit
luigi.cfg
low #jobs
FastProd,
Scalecheck
something
wrong?
yes
fix problem
remove
output
no
increase
#jobs
MergeFinal
need more
statistics?
yes
remove
merged
tables
no
make
final plots

Features

- pipeline works for any NNLOJET runcard
- run methods don't contain any external dependencies
  → works on any environment
- output of any task can still be supplied manually
- significant boost in speed through parallelization and a non-linear dependency tree
- flexible, compute only what you need
- addition of custom tasks is possible
- results are reproducible