# Automated production and evaluation of interpolation grids for cross sections at NNLO

G. Quast, **K. Rabbertz**, M. Santos Correa

Karlsruhe Institute of Technology

Institut für Experimentelle Teilchenphysik

# Motivation

- Interpretation of experimental data (strong coupling constant, PDFs, …) requires reasonably fast theory
- Often: Repeated computation of same cross section:
  - Different PDF sets; PDF uncertainties
  - Variation of renormalisation & factorisation scales $\mu_R$, $\mu_F$
  - ...
- Cross section calculations at NNLO are very costly in CPU time
- Existing technique of interpolation grids avoids repetition of time-consuming integrations, but require validation
  - Tedious workflow with many steps, error prone
  - How to automatise in reusable & reproducible way?

# The Components

- **NNLOJET:**
  Theory program providing NNLO predictions
- **APPLfast:**
  Common interface of APPLgrid & fastNLO to pQCD theory programs
- **APPLgrid, fastNLO:**
  Creation of fast interpolation grids
- **Luigi & Luigi Analysis Workflow (LAW):**
  Design of distributed, pipelined analysis workflow with inter-dependencies
- **NNLO LAW Analysis & Website:**
  Example application to NNLO calculations with integrated plot server

## *NNLOJET*

NNLOJET, A. Gehrmann-de Ridder et al., Z+jet: JHEP07 (2016) 133,
Framework: T. Gehrmann et al., PoS RADCOR2017 (2018) 074.

APPLfast, D. Britzger et al., ch. I.3 in arXiv:1803.07977.

the APPLgrid project fastNLO

APPLgrid, T. Carli et al., EPJC66 (2010) 503,
FastNLO, D. Britzger et al., arXiv:1208.3641.

luigi law luigi analysis workflow

Luigi, Spotify, https://github.com/spotifi/luigi,
LAW, M. Erdmann et al. J.Phys.Conf.Ser. 898 (2017) 072047.

https://github.com/riga/law
https://github.com/miguel-sc/nnlo-law-analysis
https://github.com/miguel-sc/nnlo-law-website

M. Santos Correa, Master Thesis, ETP-KA-2018-26, 2018,
https://ekp-invenio.physik.uni-karlsruhe.de/record/49067/files/EKP-2019-00003.pdf

# A first workflow

- **1. Preprocessing:** Check of interpolation quality
  - Short test jobs to check interpolation settings (& optimise if necessary) O(10 h)
- **2. NNLOJET Warm-up:** Vegas integration optimisation
  - 1 long (multi-core) job per process O(100 h)
- **3. APPLgrid/fastNLO Warm-up:** Adapt x- and scale-grids to accessed phase space (exact strategy differs between APPLgrid & fastNLO)
  - Only phase space provided from NNLOJET → significant speed-up O(100 h)
- **4. Interpolation grid production:**
  - Thousands of parallel jobs O(250 kh)
- **5. Postprocessing:** Statistical evaluation and combination of all produced grids …
  - Job to combine all grids and estimate statistical uncertainty O(100 h)
- **6. Validate, validate, and validate** O(? h)
- **7. Present final results :-)** 30 min :-)

# Luigi and Spotify


## Python package to build complex pipelines of batch jobs:

- **Features:**
  - **Dependency resolution**
  - **Workflow management**
  - **Visualisation**
  - **Handling failures**
  - **Command line integration**
  - **...**

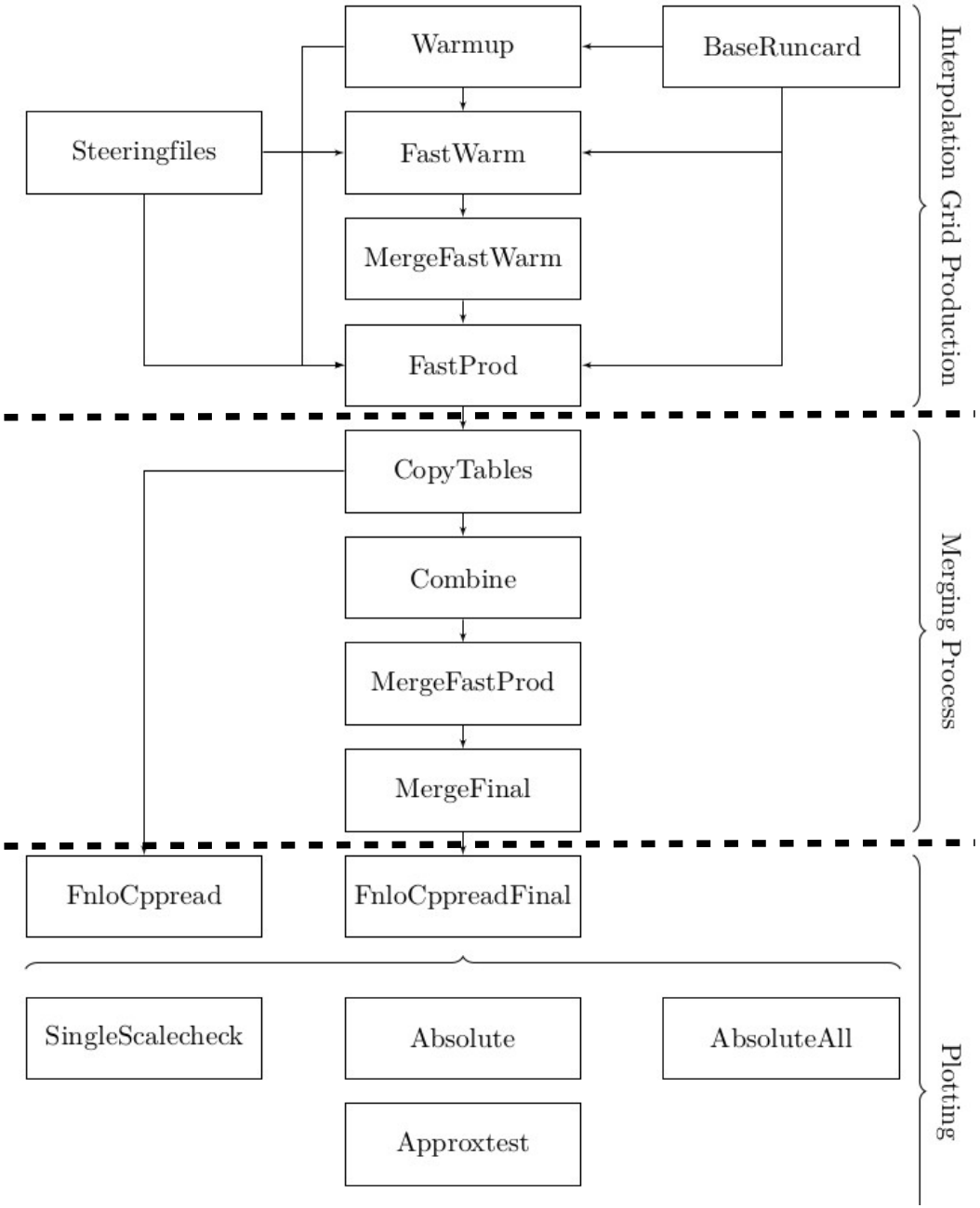
# Luigi analysis workflow



**Built on top of Luigi, adds abstraction of run locations, storage locations, and software environments:**

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

# Dependency graph

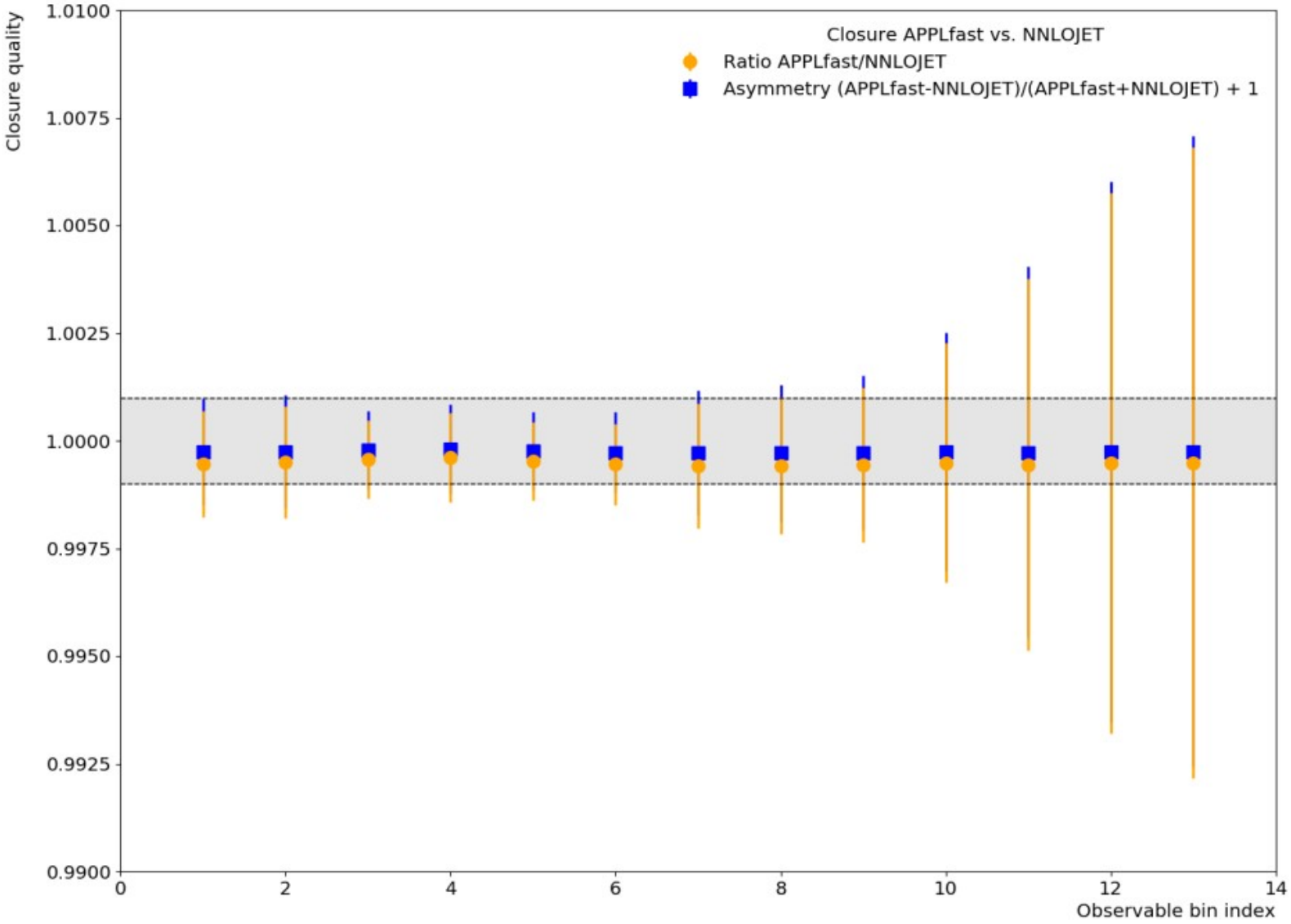



- **Interpolation grid production:**
  - NNLOJET Vegas integrations (Warmup)
  - Interpolation grid phase space optimisation (Fast warmup)
  - Mass production on large computing clusters
- **Merging process:**
  - Result harvesting to local storage space
  - Weighted merging of grids
  - Evaluation of final results
- **Plotting:**
  - Produce crosscheck & result plots
  - Provide web interface to plot collection

# Closure

## Ratio (or asymmetry) of interpolated cross section vs. NNLOJET original

**Z+jet**
**X section**



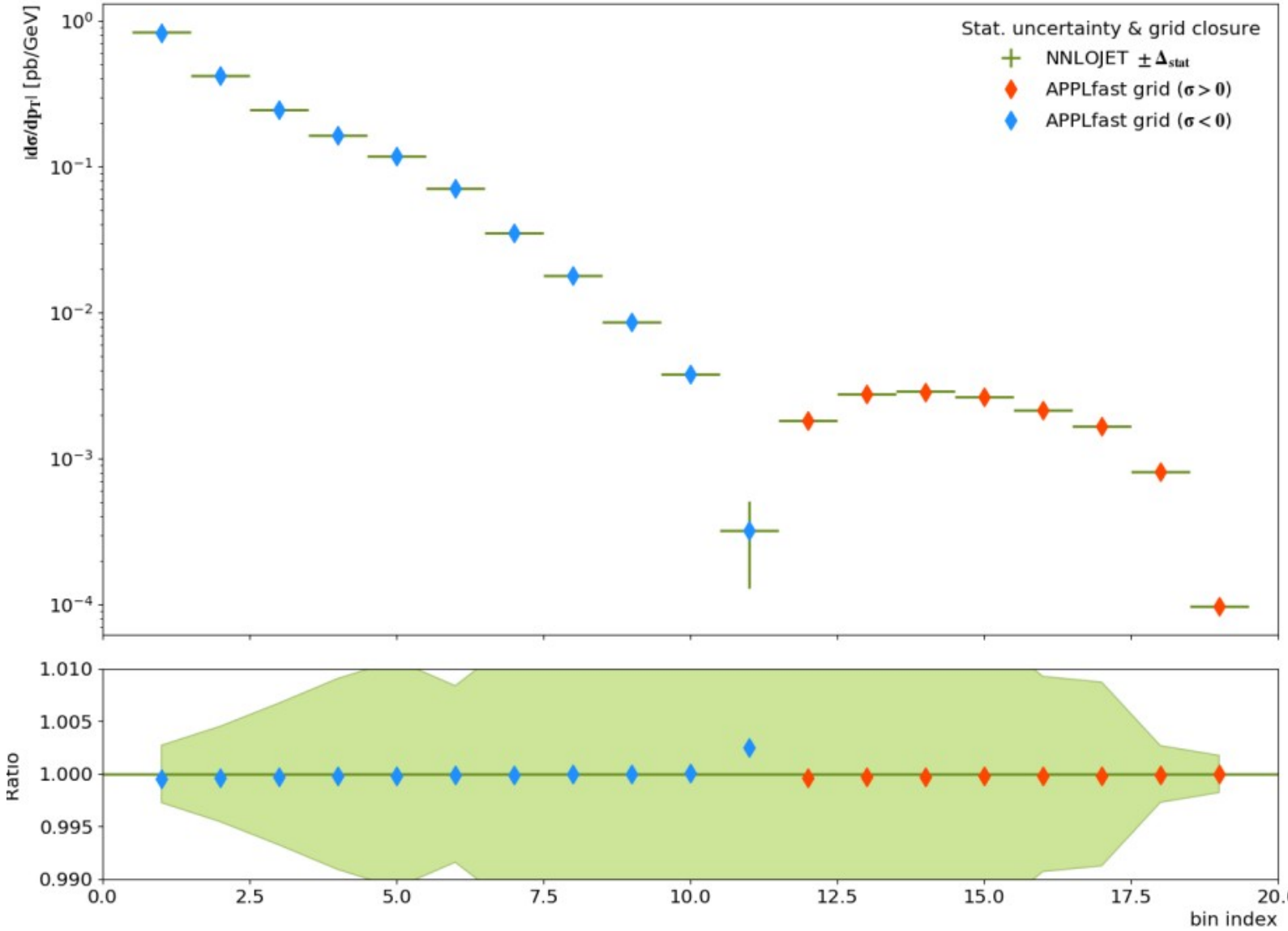

**±1‰**

**$P_{T,Z}$ bins**

# Closure in subprocesses

## Deviations with large uncertainties sometimes occur at zero(!) crossings

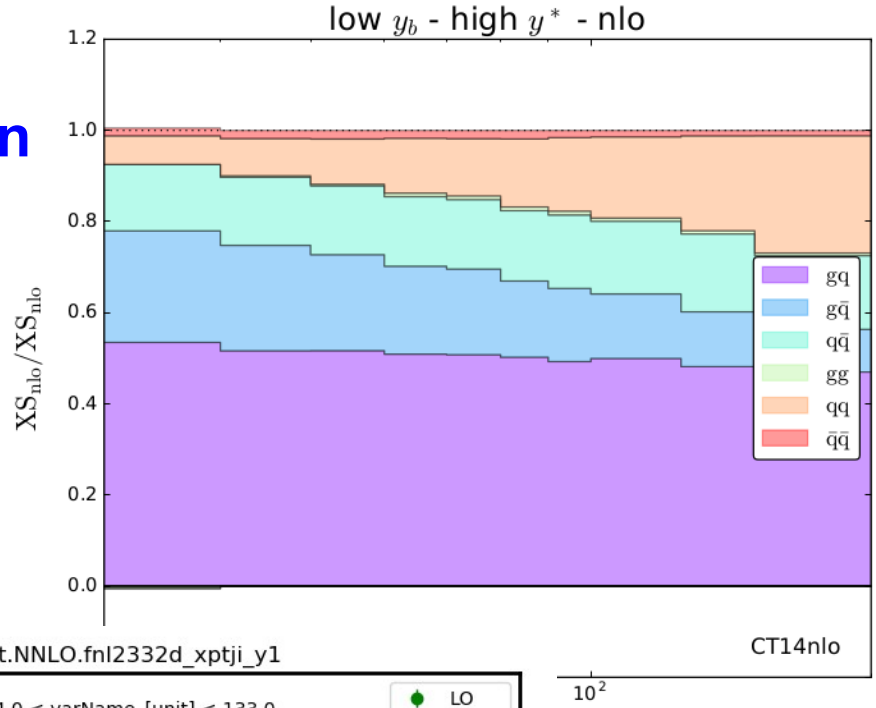
**Z+jet X section**

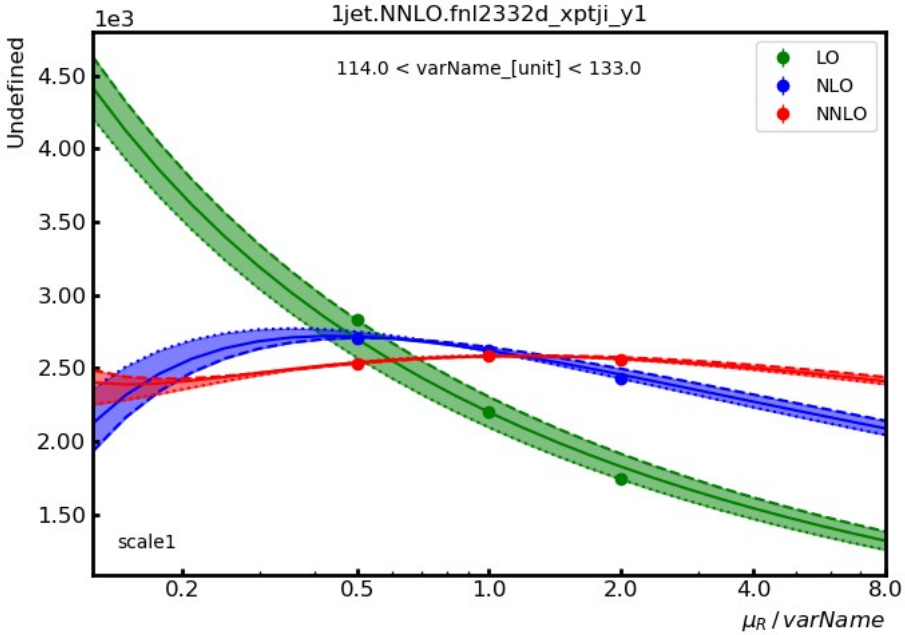



**$P_{T,Z}$ bins**

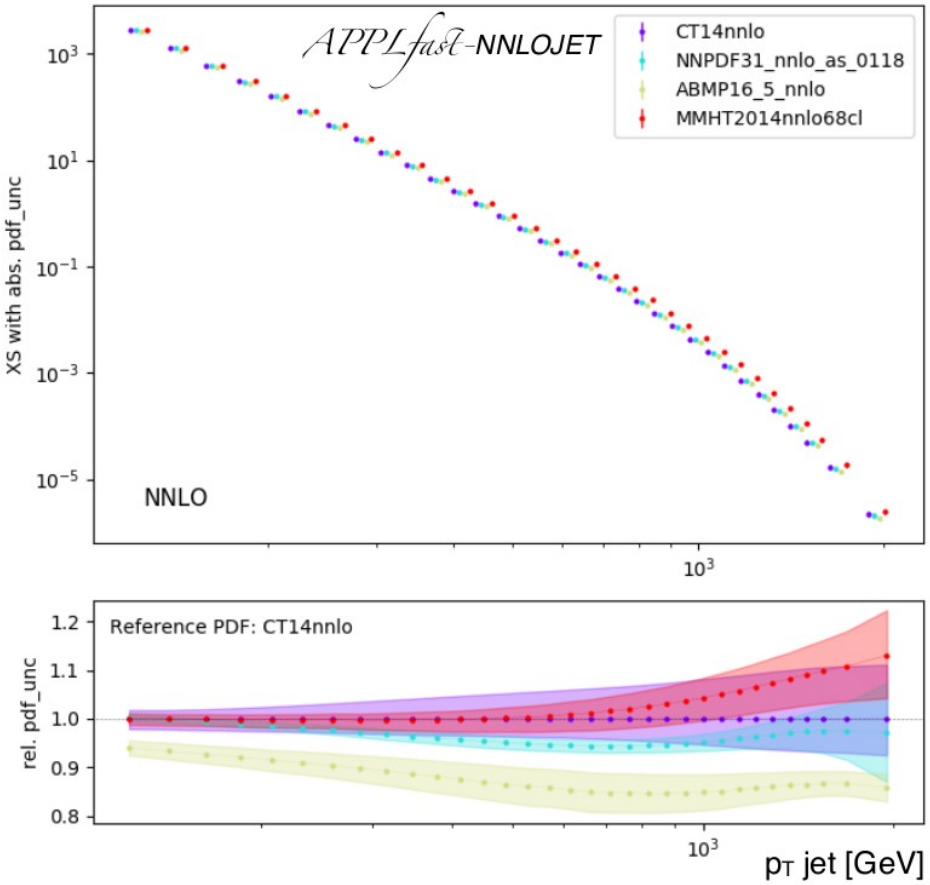
# Physics plots

## Subprocess decomposition



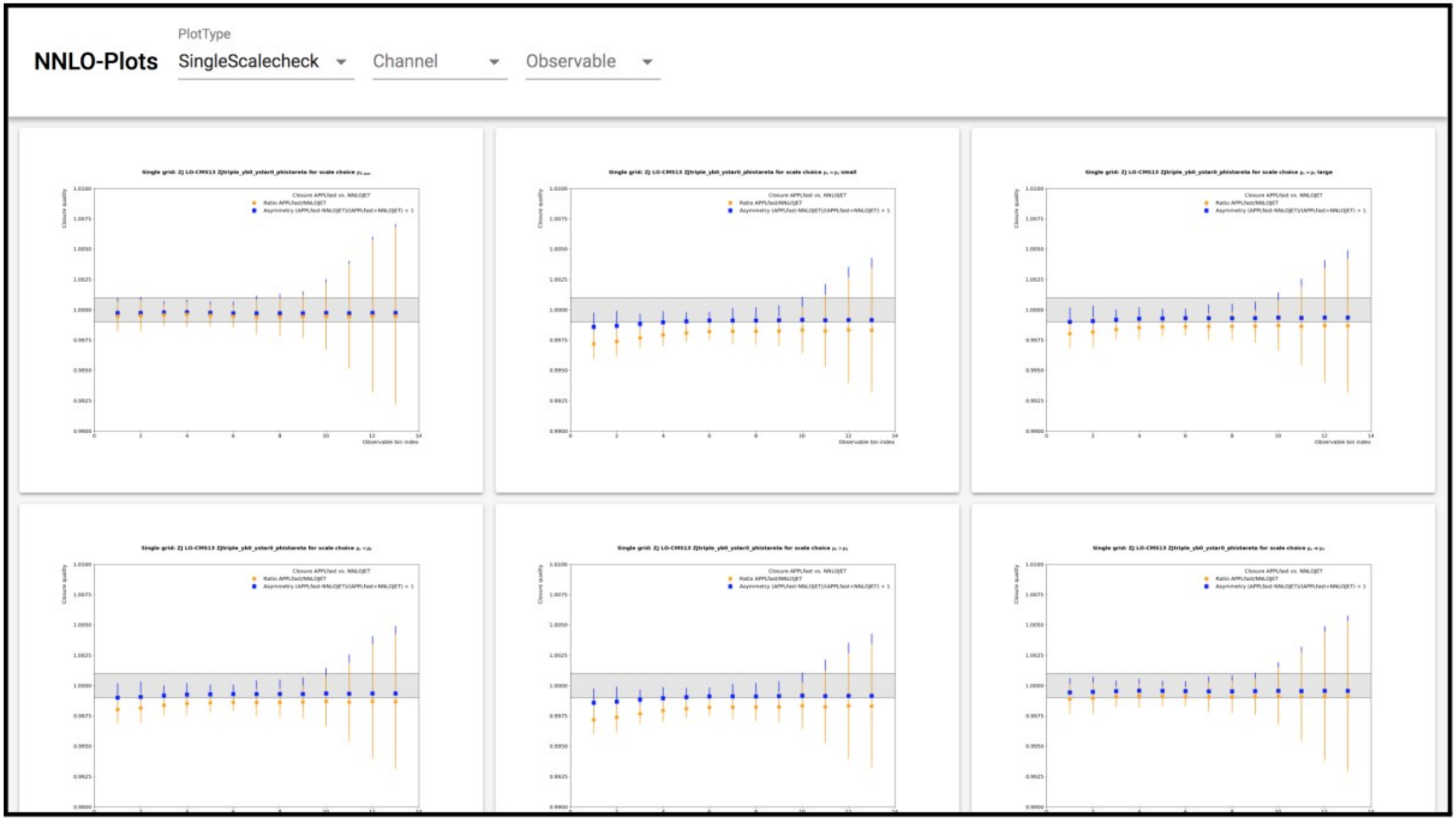




## PDF uncertainties




## Scale dependence & K factors

# Web UI for generated plots



# Summary

- Production pipeline with dependency resolution set up for NNLOJET with APPLfast interface
- One configuration file to control entire pipeline
- Output of any task can still be supplied manually, e.g. Vegas integrations from dedicated multicore compute clusters
- Expandable with custom tasks, e.g.
- Automated generation of crosscheck plots
- An additional web UI is provided to skim through generated plots
- Further improvements and developments ongoing

**Thank you for your attention and staying till the end!**

# Backup

# Interpolation concept

## Implemented in APPLgrid & fastNLO

APPLgrid, Carli et al., Eur. Phys. J. C, 2010, 66, 503.
fastNLO, Britzger et al., arXiv:0609285, 1208.3641.

## Use interpolation kernel

- Introduce set of n discrete x-nodes, $x_i$'s being equidistant in a function f(x)
- Take set of Eigenfunctions $E_i(x)$ around nodes $x_i$

→ Interpolation kernels

- Actually a rather old idea, see e.g.

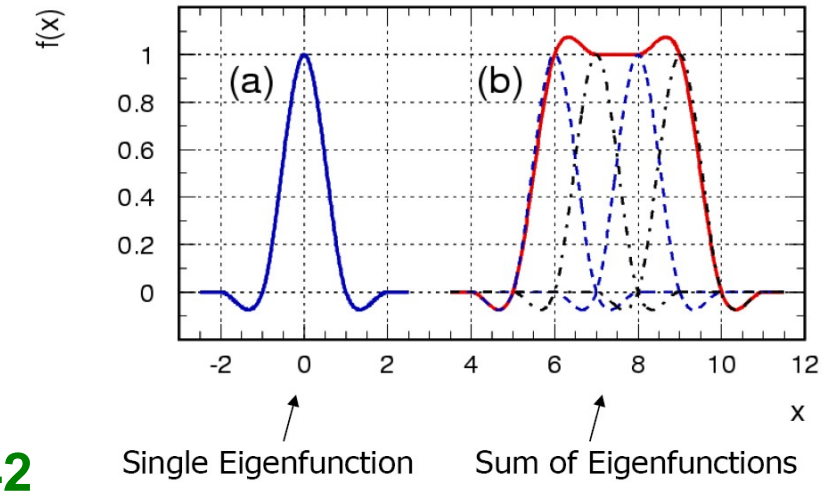
C. Pascaud, F. Zomer (Orsay, LAL), LAL-94-42

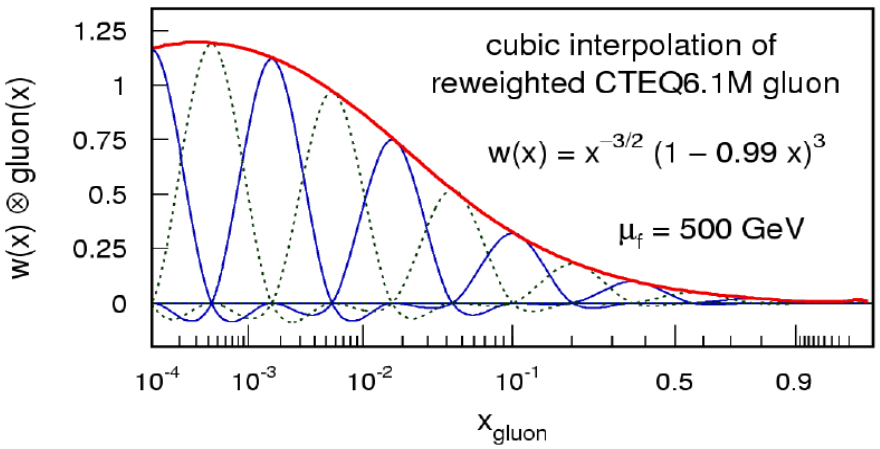
→ Single PDF is replaced by a linear combination of interpolation kernels

$$f_a(x) \cong \sum_i f_a(x_i) \cdot E^{(i)}(x)$$

→ Then the integrals are done only once

→ Afterwards only summation required to change PDF




Tabulate the convolution of the perturbative coefficients with the interpolation kernel

# Flexible-scale tables

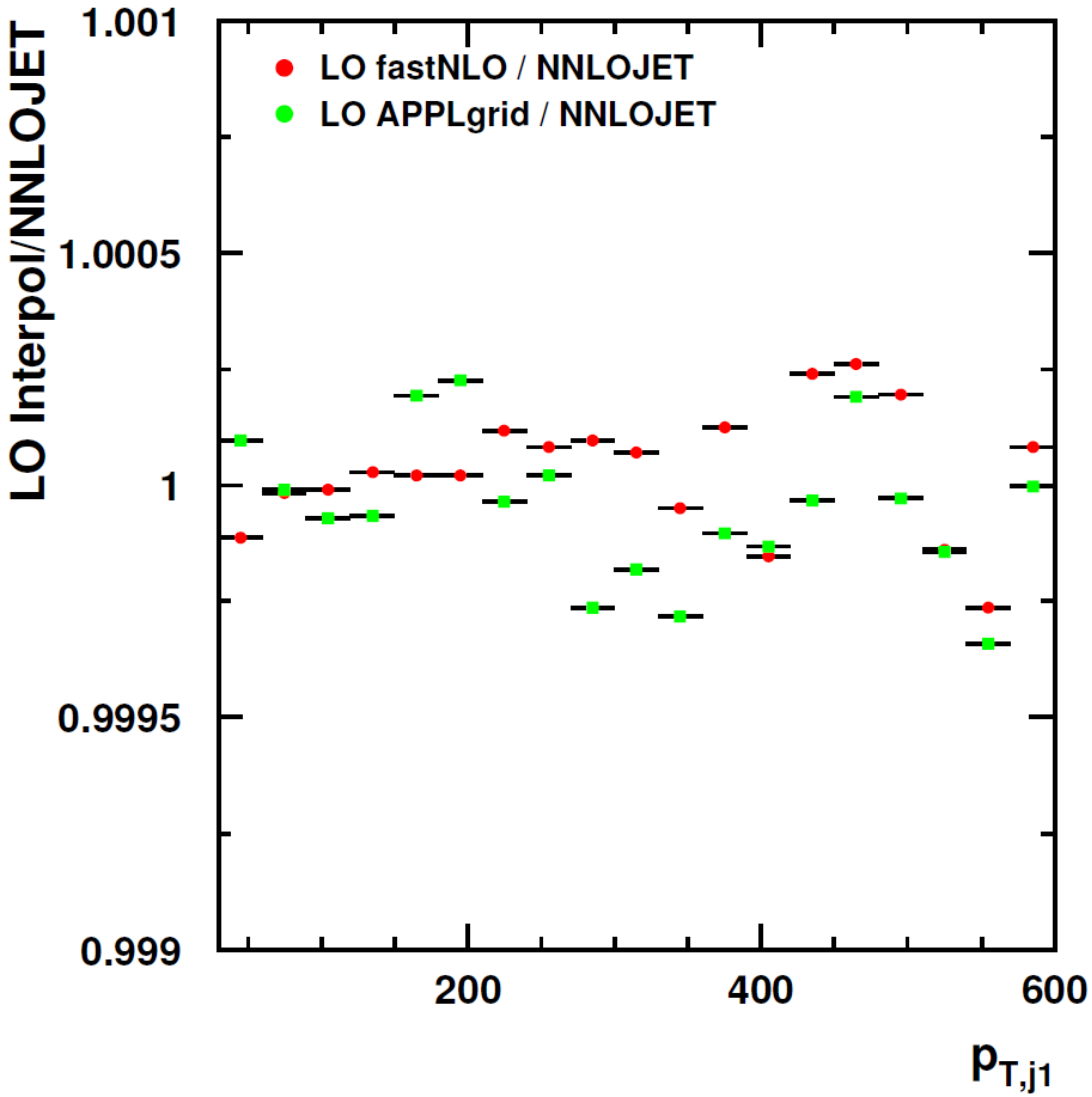
- Storage of scale-independent weights enable full scale flexibility also in NNLO
  - Additional logs in NNLO

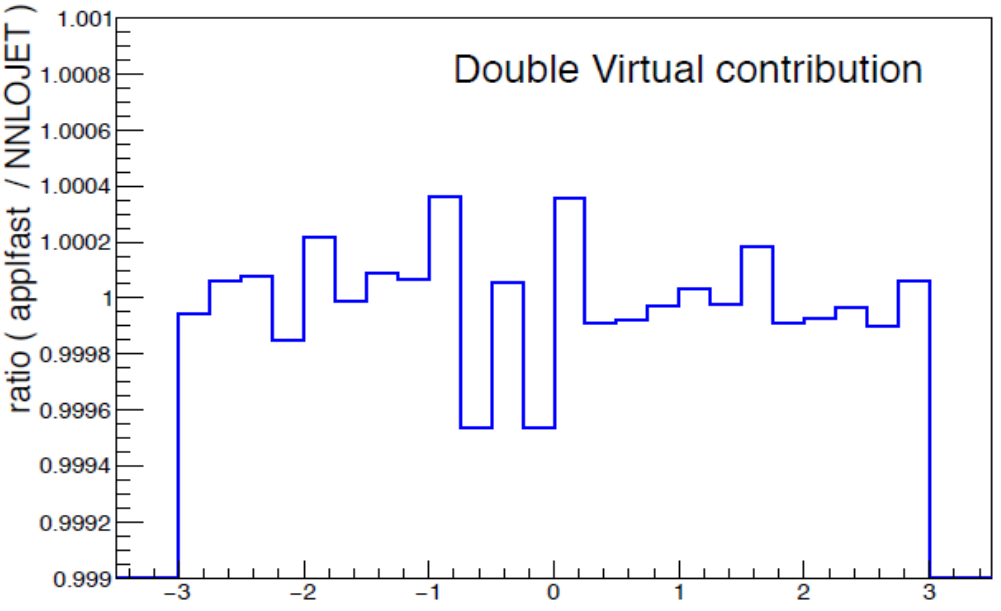
$$\omega(\mu_R,\mu_F) = \underbrace{\omega_0 + \log(\mu_R^2)\omega_R + \log(\mu_F^2)\omega_F}_{\text{log's for NLO}} + \underbrace{\log^2(\mu_R^2)\omega_{RR} + \log^2(\mu_F^2)\omega_{FF} + \log(\mu_R^2)\log(\mu_F^2)\omega_{RF}}_{\text{additional log's in NNLO}}$$

  - Store weights: $w_0$, $w_R$, $w_F$, $w_{RR}$, $w_{FF}$, $w_{RF}$ for order $\alpha_s^{n+2}$ contributions
- Advantages
  - Renormalization and factorization scale can be varied *independently* and by *any* factor
    - No time-consuming 're-calculation' of splitting functions in NLO necessary
  - Only small increase in amount of stored coefficients
- Implementation
  - *Two* different observables can be used for the scales
    - e.g.: $H_T$ and $p_{T,max}$
    - or e.g.: $p_T$ and $|y|$
    - …
  - *Any function* of those *two observables* can be used for calculating scales

# Step 1: Preprocessing

## Z+jet LO Approximation Test
## Similar performance at subpermille level




M. Sutton at QCD@LHC

**Z+jet Test Setup:** $p_{T,j1}$, $\eta_{j1}$, $y_Z$

- $E_{cms}$ = 8 TeV
- $p_{T,jet}$ > 30 GeV
- $|y_{jet}| < 3$
- $|y_{l+,l-}| < 5$
- $80 < M_{l+l-} < 100$ GeV
- $\mu_r = \mu_f = M_Z$ = fixed

(→ no scale interpolation in this test!)

# Step 2: Vegas Integrations

- **NNLOJET Warm-up:**
  - **Must be one job per process type**
  - **Multi-threading possible**

| Job Type | # Jobs | Threads / Job | Events / Job | Runtime / Job | Total Runtime |
|---|---|---|---|---|---|
| LO | 1 | 16 | 32 M | 0.35 h | 0.35 h |
| NLO-R | 1 | 16 | 16 M | 1.0 h | 1.0 h |
| NLO-V | 1 | 16 | 16 M | 1.0 h | 1.0 h |
| NNLO-RRa | 1 | 32 | 5 M | 17.5 h | 17.5 h |
| NNLO-RRb | 1 | 32 | 5 M | 20.7 h | 20.7 h |
| NNLO-RV | 1 | 16 | 8 M | 22.4 h | 22.4 h |
| NNLO-VV | 1 | 16 | 8 M | 24.6 h | 24.6 h |
| Total | 7 | - | - | - | 87.6 h |

# Step 3: Phase Space Exploration

- **APPLfast Warm-up:**
  - NNLOJET is run without CPU-time expensive weight calculation
  - At least 1 job per process needed to determine phase space limits individually
  - Grids created and optimised during warm-up (APPLgrid)
  - Grids created in production step from optimised x and Q-scale limits (fastNLO)
  - Warm-up can be parallelised, if necessary (fastNLO)
  - Presented table used for extensive testing; overkill for normal use

| Job Type | # Jobs | Events / Job | Runtime / Job | # Events | Total Runtime |
|---|---|---|---|---|---|
| LO | 5 | 500 M | 12 h | 2.5 G | 60 h |
| NLO-R | 5 | 300 M | 18 h | 1.5 G | 90 h |
| NLO-V | 5 | 500 M | 13 h | 2.5 G | 65 h |
| NNLO-RRa | 10 | 50 M | 13 h | 0.5 G | 130 h |
| NNLO-RRb | 10 | 50 M | 15 h | 0.5 G | 150 h |
| NNLO-RV | 5 | 300 M | 19 h | 1.5 G | 90 h |
| NNLO-VV | 5 | 500 M | 12 h | 2.5 G | 60 h |
| Total | 45 | --- | --- | 11.5 G | 645 h |

In this setup most $x_{min}$ limits from LO runs, 3 from higher-order runs.

# Step 4: Mass Production

- **NNLOJET + APPLfast**
  - **Massive parallelised computing on Virtual Machines with 24h lifetime**
  - **Example with fastNLO, APPLgrid example in progress**

| Job Type | # Jobs | Events / Job | Runtime / Job | # Events | Total Output | Total Runtime |
|---|---|---|---|---|---|---|
| LO | 10 | 140 M | 20.6 h | 1.4 G | 24 MB | 206 h |
| NLO-R | 200 | 6 M | 19.0 h | 1.2 G | 1.3 GB | 3800 h |
| NLO-V | 200 | 5 M | 21.2 h | 1.0 G | 1.2 GB | 4240 h |
| NNLO-RRa | 5000 | 60 k | 22.5 h | 0.3 G | 26 GB | 112500 h |
| NNLO-RRb | 5000 | 40 k | 20.3 h | 0.2 G | 27 GB | 101500 h |
| NNLO-RV | 1000 | 200 k | 19.8 h | 0.2 G | 6.4 GB | 19800 h |
| NNLO-VV | 300 | 4 M | 20.5 h | 1.2 G | 2.0 GB | 6150 h |
| Total | 11710 | --- | --- | 5.5 G | 64 GB | 248196 h |

**3 times 11710 grids/tables + all NNLOJET output!**
**Final 3 files for analysis are O(10 MB) each.**

# Step 5: Postprocessing

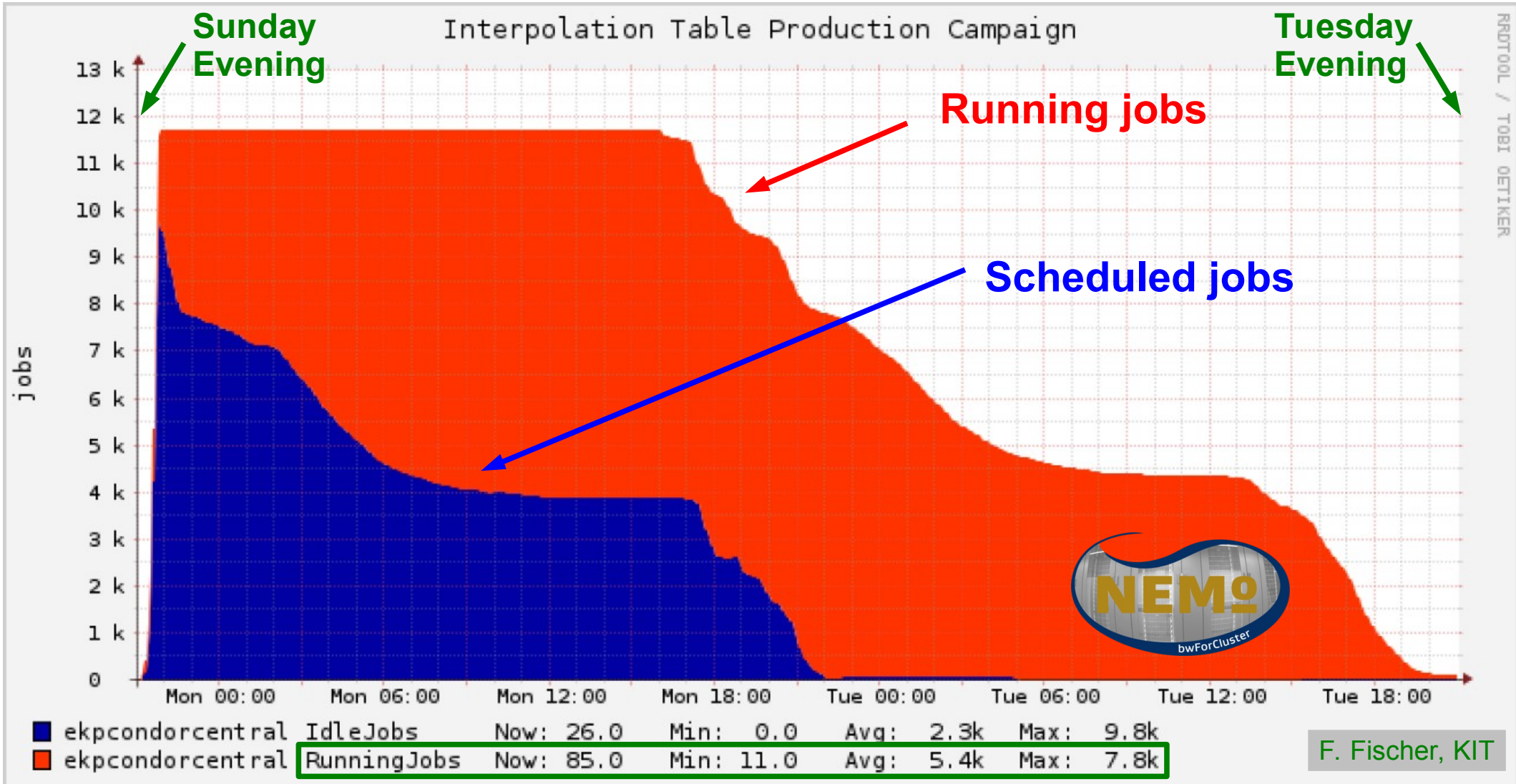
- **Checking, purging, combining:**
  - Check again interpolation quality for individual grids
  - Run NNLOJET combination script → weight tables
  - Weighted merging of grids
  - Check and treat potential remaining unsuppressed fluctuations
  - Do some nice physics

# Step 6: Validation

- **Check every aspect you can or you will be hit by Murphy's law!**
  - Check each contribution (LO, R, V, RRa, RRb, RV, VV) separately
  - Check interpolation in x-space for single grids
  - Check interpolation in scales for single grids
  - Compare merged grids to NNLOJET for each contribution
  - Compare final merged grids for each order to NNLOJET
  - … more checks/comparisons, e.g. to other programs
  - ?

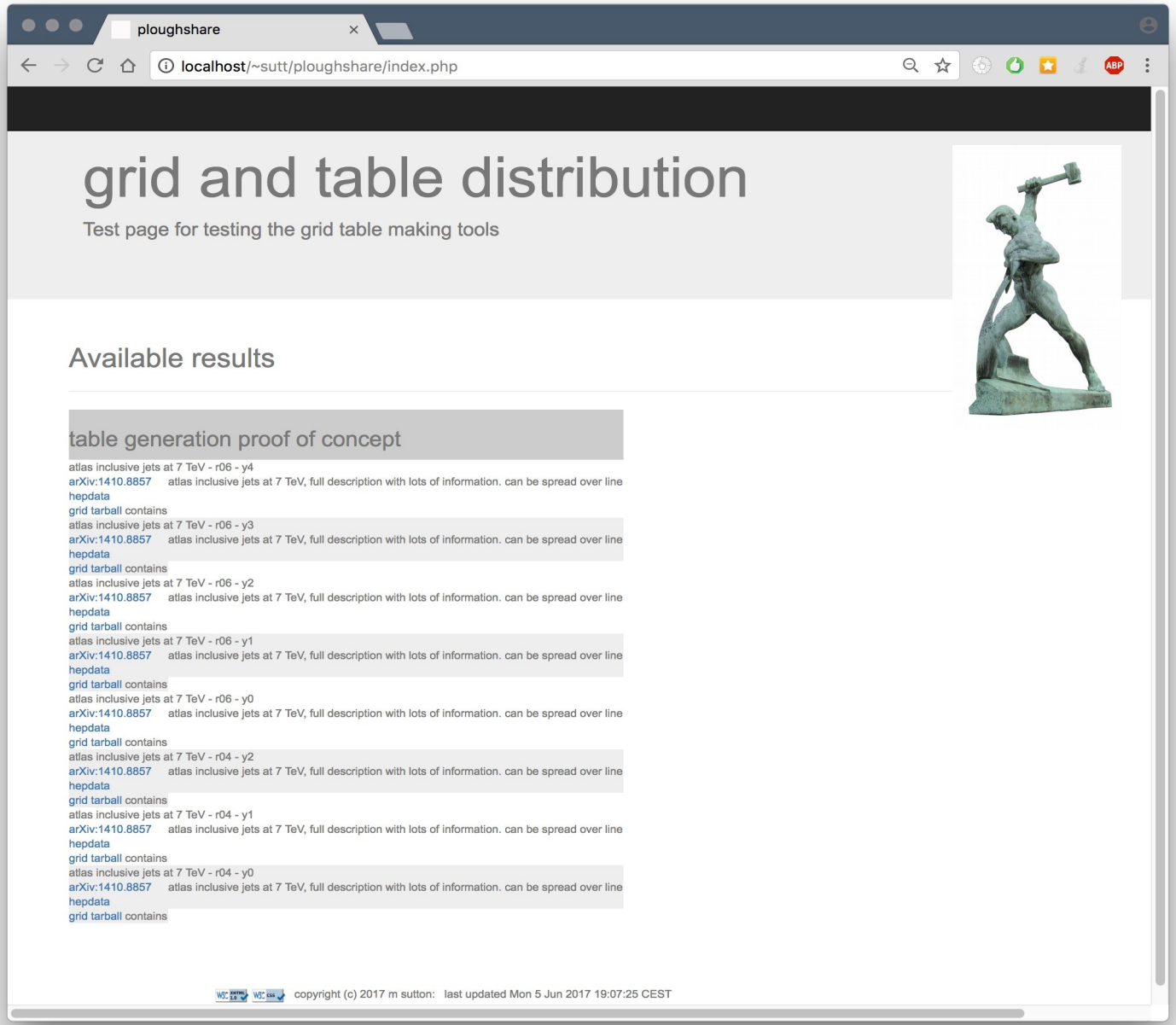
# Production Campaign

**Optimised scenario: Finished in two days with 7800 parallel jobs at maximum!**




**Thanks to bwHPC and the NEMO HPC cluster team in Freiburg!**

# Grid distribution – Ploughshare




- **Where to find the grids?**
  **Idea: New hep forge package where registered users upload grids with some documentation**
- **Registered(!) user gets FAME or BLAME**
- **Automated job treats the upload:**
  - **Add to the appropriate location in the file system**
  - **Generate relevant lists, and display web pages**
- **Should provide a user interface for automated download with a simple line of code**
- **Have expression of interest from many stakeholders …**
  - **ATLAS, CMS, HERA (H1 + ZEUS), MMHT, CTEQ, NNPDF, xFitter, APPLgrid, fastNLO …**
- **Simple proof of concept is there, development stalled a bit lately by lack of time. HELP is WELCOME!**