

Karlsruher Institut für Technologie



### The CROWN Framework **167th ROOT PPP Meeting**

Sebastian Brommer, Nils Faltermann, Olha Lavoryk, Moritz Molch, Artur Monsch, Ralf Schmieder, Nikita Shadskiy



# **CROWN NTuple Framework**



- Convert CMS NanoAOD into analysis NTuples
- > Focus on efficient and fast processing with minimal dependencies
- Automatic handling of systematics, optimised disk usage of outputs
- > Friend tree support
- Scaling via workflow tool and HTCondor batch system

Sebastian Brommer - The CROWN Framework

### **CROWN**

Histogram Framework

	\$2 Ec	dit Pins 👻 💿 Unwatch 7 👻	약 Fork 21 → 🛉 Starred 4
្រៃ main 👻 ្រៃ 9 Branches 🛇 3 Tags	Q Go to file t	Add file 👻 <> Code 👻	C++ based ROOT Workflow for tuples (CROWN)
Add working Onnxruntime Produce	r (#254) 🚥 🗸 78bd176 ·	last week 🕚 1,065 Commits	♂ crown.readthedocs.io
.github	update cmake CI commands	last month	cms analysis root wlcg
analysis_configurations	update tauID sf readout	last month	particle-physics ntuples hep-
Checks	WIP: Code cleanup ( <b>#205</b> )	8 months ago	따 Readme 해 MIT license
🖿 cmake	add_lib options	last month	-/- Activity
code_generation	Add working Onnxruntime Producer (#254)		E Custom properties
🖿 data	missed name change second try	last week	☆ 4 stars ⊙ 7 watching
docker	add unittest container for Cl	last month	មិ 21 forks
docs	Add working Onnxruntime Producer (#254)		<b>v0.3</b> Latest
include	Add working Onnxruntime Producer (#254)	last week	
profiling	adapt profiling script for new args ordering	2 years ago	Packages 1
src 🖿	Add working Onnxruntime Producer (#254)		fø crown-crown-unittest
tests	update testfiles curl command	8 months ago	Contributors 13
🗋 .clang-format	add check to test cpp formatting	3 years ago	🐣 💿 🕕 🗘 😂 🕲
🗋 .gitignore	add functionality to go from tarball to the exact o	configura last year	🕆 🕄 😃 🌒 🚯
🗋 .gitmodules	relocate RoccoR	2 years ago	
🕒 .readthedocs.yaml	update readthedocs config	last month	● C++ 66.0% ● Python 31.2% ● CMake 1.9% ● Other 0.9%
			• CMake 1.9% • Other 0.9%
			● C++ 66.0% ● Python 31.2%







- > NTuples can be used by multiple people, provided by one person
- > Defined state of selection / corrections
- > Most changes can be made on the level of the Histogram Framework
- > Expensive calculation of high level variables can be done later
- > Smaller file inputs for the histogram framework  $\rightarrow$  faster runtime and faster analysis turnarounds

# **CROWN Usage**

- > Used by the KIT CMS Group since ~2 years
- > O(10) analysis within CMS use CROWN
- > Listed as "Supported Framework" by the CMS analysis tool group (link)
- > 1065 commits and 204 pull requests by 13 Contributers

# Core Principles and Basic building blocks of CROWN



1	1		1	1	1	1	1	1	1	1	1	ľ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1



### **Core Principles** C++ and Python

- > Utilize python as a configuration language to auto-generate C++ code
- Combine simple C++ functions into one large RDataFrame
- > Executables have simple interface
- Multiple executables for different samples, and eras
- > Focus on validation of user configuration before the compilation



> ./config\_ttbar\_2018 outputfile.root /path/to/inputfiles/\*.root





### **Quantities and Producers** Basic Building Blocks

- > Quantity objects to track inputs and outputs
- > Producers to calculate new quantities and Filters to filter events
- > ProducerGroups to organise Producers easier



MET\_phi = NanoAODQuantity("PuppiMET\_phi")
MET\_pt = NanoAODQuantity("PuppiMET\_pt")

```
met_p4 = Quantity("met_p4")
met = Quantity("met")
```

### \_met = Quantity("met")

```
BuildMetVector = Producer(
    name="BuildMetVector",
    call="lorentzvectors::buildMet({df}, {input}, {output})",
    input=[
        nanoAOD.MET_pt,
        nanoAOD.MET_phi,
    ],
    output=[q.met_p4],
    scopes=["global"],
)
MetPt = Producer(
    name="MetPt",
    call="quantities::pt({df}, {output}, {input})",
    input=[q.met_p4],
    output=[q.met],
    scopes=["global"],
)
```

```
output=[q.met],
scopes=["global"]
```



# C++ Functions Basic Building Blocks

- Identical base pattern for all CROWN functions (df as first argument, df as return object)
- > Opt for simple & generic functions
- > No just-in-time compilation
- Designed to be shared among different analyses

```
BuildMetVector = Producer(
             name="BuildMetVector",
             call="lorentzvectors::buildMet({df}, {input}, {output})",
             input=[
                 nanoAOD.MET_pt,
                 nanoAOD.MET_phi,
             ١,
             output=[q.met_p4],
             scopes=["global"],
ROOT::RDF::RNode buildMet(ROOT::RDF::RNode df, const std::string &met_pt,
                         const std::string &met_phi,
                         const std::string &outputname) {
    auto construct_metvector = [](const float &pt, const float &phi) {
        // for Met, eta is zero
       auto met = ROOT::Math::PtEtaPhiEVector(pt, 0, phi, pt);
        // cast Met vector to a ROOT::Math::PtEtaPhiMVector to make latter
        // functions easier to use
       return (ROOT::Math::PtEtaPhiMVector)met;
    };
```

return df.Define(outputname, construct\_metvector, {met\_pt, met\_phi});



### Python Configuration **Basic Building Blocks**

- > The whole configuration is stored in a python object
- > Parameters are used to set cut values, working points etc.
- > Scopes for different final states, one output file per scope



Sebastian Brommer - The CROWN Framework

configuration = Configuration( era, sample, scopes, shifts, available\_sample\_types, available\_eras, available\_scopes,

configuration.add\_config\_parameters( "global", "min\_muon\_pt": 10.0, "max\_muon\_eta": 2.4,

```
"max_muon_dxy": 0.045,
"max_muon_dz": 0.2,
"muon_id": "Muon_mediumId",
"muon_iso_cut": 0.3,
```

**}**,

)

### **Python Configuration Basic Building Blocks**

User defines all **Producers** to be run and all **Quantities** to be added to the output

Gives CROWN complete knowledge

- > on what to run
- > on what to modify (see next slides)
- > to validate the user config before compiling and running
- > to optimise configuration and producer ordering
- > Minimize "magic" functions covering special cases



configuration.optimize() configuration.validate() configuration.report() return configuration.expanded\_configuration()

# Code Generation



# **Code Generation**

User

- > Based on configuration, C++ code is generated, one file per defined producer
- > Steering via cmake with extensive report on things like unused parameters, size of RDataFrame etc.
- Code generation is fast (4-5 s for a RDataFrame with 15k Defines)
- Compile times are reasonably fast (2-3 min for a 15k Defines RDataFrame with 20 cores)



# **Code Generation**

### Folder Structure





// [...] // [...]

### Auto-generated function calls

ROOT::RDF::RNode MetBasic
CorrectionManager &correc
[]
<pre>auto df2 = lorentzvect</pre>
auto df3 = lorentzvec <sup>-</sup>
"met_p4metUncluster
auto df4 = lorentzvec <sup>-</sup>
"met_p4metUncluster
auto df5 = quantities
auto df6 = quantities
auto df7 = quantities
[]
return df19;

### Repo containing a complete version of CROWNs autogenerated code (link)

#include "[…]/include/global/MetBasics.hxx"

auto df13\_global = MetBasics\_global(df12\_global, onnxSessionManager, correctionManager);

\_global (ROOT::RDF::RNode df0, OnnxSessionManager &onnxSessionManager, tionManager) {

tors::buildMet(df1, "PuppiMET\_pt", "PuppiMET\_phi", "met\_p4"); tors::buildMet(df2, "PuppiMET\_ptUnclusteredDown", "PuppiMET\_phiUnclusteredDown", edEnDown"); tors::buildMet(df3, "PuppiMET\_ptUnclusteredUp", "PuppiMET\_phiUnclusteredUp", edEnUp"); ::pt(df4, "met\_uncorrected", "met\_p4"); ::pt(df5, "met\_uncorrected\_\_metUnclusteredEnDown", "met\_p4\_\_metUnclusteredEnDown"); s::pt(df6, "met\_uncorrected\_\_metUnclusteredEnUp", "met\_p4\_\_metUnclusteredEnUp");



# Advanced Features



### **Python Configuration** Advanced Building Blocks

- Rules are used to modify the configuration for different types of samples (e.g. sample-specific corrections)
- Modifiers to change parameters based on samples or eras
- > Systematic Shifts are support for different types of shifts:
  - Different Producers
  - Different configuration parameters
  - Different input quantities

```
configuration.add_modification_rule(
    scopes,
    AppendProducer(producers=event.TopPtReweighting, samples="ttbar"),
)
configuration.add_config_parameters(
    ["et", "mt", "tt"],
    {
        "tau_sf_file": EraModifier(
        {
            "2017": "data/jsonpog-integration/P06/TAU/2017_UL/tau.json.gz",
```

```
"2017": "data/jsonpog-integration/POG/TAU/2017_UL/tau.json.gz",
"2018": "data/jsonpog-integration/POG/TAU/2018_UL/tau.json.gz",
}
),
},
}
```

```
configuration.add_shift(
    SystematicShift(
        name="vsEleBarrelDown",
        shift_config={("et", "mt"): {"tau_sf_vsele_barrel": "down"}},
        producers={("et", "mt"): scalefactors.Tau_2_VsEleTauID_SF},
    )
```





### **Python Configuration Advanced Building Blocks**

- > Shifts are propagated through the whole configuration
- > All producers affected by a shift automatically also produce shifted quantities
- Shifted quantities added to output and identified via

<quantity\_name>\_\_<shift\_name>

"met\_p4");

```
configuration.add_shift(
    SystematicShiftByQuantity(
        name="metUnclusteredEnDown",
        quantity_change={
            nanoAOD.MET_pt: "PuppiMET_ptUnclusteredDown",
            nanoAOD.MET_phi: "PuppiMET_phiUnclusteredDown",
        },
        scopes=["global"],
    ),
    exclude_samples=["data", "embedding", "embedding_mc"],
```





# **Additional Features**

- Support for the production of FriendTrees (requires a CROWN NTuple as input) including a continuation of automatic systematics tracking
- Support for the production of FriendTrees with multiple input trees (aka a CROWN NTuple and some FriendTrees)
- > Support for CMS correctionlib the standard tool in CMS standard for providing corrections
- > Support for *onnxRuntime* for ML inference
- CROWN builds contain all information to get the exact config and code that was used (e.g. to investigate mistakes in NTuples)

# Performance

### Performance

Copying and processing local files



# required for analysis

### Dataframe 15.8k Defines // 8 Threads // ROOT 6.30 Input 14,289,000 Events // 28 GB Output 6 Final States // 20 GB



Processing 5600 Events/s into 6 final states including all systematics





### Performance With recent master changes



# 30% in this example), while reducing the runtime by 10-15%

Dataframe 15.8k Defines // 8 Threads Input 14,289,000 Events // 28 GB Output 6 Final States // 20 GB



Current ROOT master significantly lowers the memory footprint (by about

### Performance CorrectionManager



New feature from yesterday (PR) significantly reduces setup time since corrections are only loaded ones and shared between functions, plus reduced memory usage

### Dataframe 15.8k Defines // 8 Threads Input 14,289,000 Events // 28 GB Output 6 Final States // 20 GB



# Workflow Management



### **Production for Analysis Workflow Management**





### 150 Samples



### Workflow management Kingmaker

- > To orchestrate CROWN, we use KingMaker, a law + Luigi workflow
- Organise samples with a sample database tool (link)
- > Kingmaker takes care of building all required tarballs, submitting jobs to HTCondor, writing outputs to grid storage
- > Allows for a turnaround cycle of < 1 day for large scale analyses (more than 100M events per era)



	🖒 Edit Pins 👻	⊙ Unwatch (3) →	양 Fork 10 - ☆ St
਼ਿੰ main 👻 ਟਿ Branches 🛇 0 Tags	Q Go to file <b>t</b> Add file	<> Code -	Workflow Management for C
larrypuuter addess deepsource comments	× 7887bd7 · last week	C 275 Commits	① Readme 좌 MIT license
.github/workflows	ping black version	last week	∿- Activity
conda_environments	Update yml files of envs to reflect current envs located o	8 months ago	■ Custom properties ☆ 0 stars
➡ law @ d067e02	update law version, add required changes in kingmaker	8 months ago	
lawluigi_configs	set default back to grid	last week	얒 <b>10</b> forks
ml_configs/example_configs	Allow relative paths for config files. Add example configs	2 years ago	Contributors 10
processor	remove tar command from build script	last week	🗟 🕂 🌐 😫
🔄 sample_database @ 64419ec	update sample_db submodule	last month	😃 😮 👘
scripts	addess deepsource comments	last week	
setup	remove explicit username from configs and use env varia	3 years ago	Python 88.2% • Shell 11.4
sm-htt-analysis @ b27f1bc	black	2 years ago	Report repository
🕒 sm-htt-analysis @ b27f1bc	black	2 years ago	Report repository

law run ProduceSamples ---analysis tau ---config config ---sample-list samples\_18.txt -production-tag 2018\_ntuples\_v10 --workers 100 --scopes mt,et,tt --shifts all





### Workflow management Kingmaker

- > Minimizes build times by reusing crownlib containing all CROWN C++ functions
- Supports FriendTree production, keeping track of all required inputs / outputs
- > Friends are always run single-core with one input file per job to maintain the correct structure







		🖒 Edit Pins 👻	⊙ Unwatch 3	- * * Fork 10 - ☆ Star
양 main → 양 5 Branches ⓒ 0 Tags	Q Go to file	t Add file 👻	<> Code -	Workflow Management for CRC
Arrypuuter addess deepsource comments		7887bd7 · last week	C 275 Commits	印 Readme 화 MIT license
.github/workflows	ping black version		last week	-\- Activity
conda_environments	Update yml files of envs to reflect	current envs located o	8 months ago	E Custom properties
► law @ d067e02	update law version, add required c	hanges in kingmaker	8 months ago	
lawluigi_configs	set default back to grid		last week	양 10 forks 
ml_configs/example_configs	Allow relative paths for config files	. Add example configs	2 years ago	Contributors 10
processor	remove tar command from build so	ript	last week	ಿ 🕂 💮 🖨 🔮
Sample_database @ 64419ec	update sample_db submodule		last month	😃 🕲 💮
scripts	addess deepsource comments		last week	
setup	remove explicit username from cor	nfigs and use env varia	3 years ago	• Python 88.2% • Shell 11.8%
sm-htt-analysis @ b27f1bc	black		2 years ago	Report repository
🕒 sm-htt-analysis @ b27f1bc	black		2 years ago	Report repository

law run ProduceSamples ---analysis tau ---config config ---sample-list samples\_18.txt -production-tag 2018\_ntuples\_v10 --workers 100 --scopes mt,et,tt --shifts all

law run ProduceFriends --analysis tau --config config --friend-config fastmtt -sample-list samples18.txt -shifts all --friend-name fastmtt\_friends\_v3 --production-tag 2018\_ntuples\_v10 --scopes mt,et,tt --workers 100



# Conclusion

- > CROWN is the main NTuple framework for multiple analysis within CMS
- > Law-based workflow manangement tool to run large-scale productions on a batch system
- > Focused on performance and efficiency
- > CROWN directly profits from all **RDataFrame improvements**



### Search docs

### ntroduction

Changelog

FriendTree Generation KingMaker

Ntuples in Postprocessing

### SETUP YOUR OWN CONFIGURATION

Writing a new producer

Writing a new C++ function

Defining a New Python Producer Best Practices for Contributions

Writing a CROWN Configuration

### DOCUMENTATION

Documentation of available python classes

Documentation of available C++ functions

### TUTORIALS

samples

How to build ROOT with CVMFS on CentOS 7 How to generate dummy NanoAOD / Welcome to The CROWN documentation

### C Edit on GitHub

### Welcome to The CROWN documentation!

The C ++-based RO OT W orkflow for N -tuples (CROWN) is a fast new way to convert NanoAOD samples into flat Trees to be used in further analysis. The main focus of the framework is to provide a fast and clean way of selecting events and calculating quantities and weights. The framework has minimal dependencies and only uses ROOT and it's Dataframe as a backend.

### Note

To get started, go here: Getting started.

### Note

To read about recent changes and new features, go here: Changelog.

### **Available Analyses**

The following analysis configurations are currently available in CROWN. If you want to add your analysis configuration, contact the developers.

### Available Analyses Configurations for CROWN

Analysis name	Repository
HTauTau	https://github.com/KIT-CMS/TauAnalysis-CROWN
earlyrun3	https://github.com/khaosmos93/CROWN-config-earlyRun3
WHTauTau	https://github.com/KIT-CMS/WHTauTauAnalysis-CROWN

### **Documentation Content**

- Introduction
- Design Idea
- Getting started
- Running the fram

CROWN: https://github.com/KIT-CMS/CROWN KingMaker: <a href="https://github.com/KIT-CMS/KingMaker">https://github.com/KIT-CMS/KingMaker</a> Documentation: https://crown.readthedocs.io/en/latest/index.html







Questions ?

# **Jitting for Snapshots**

### Currently, the snapshot is the only fitted part of CROWN

Info in <[ROOT.RDF] Info /build/jenkins/workspace/lcg\_release\_pipeline/build/projects/ROOT-6.30.02/src/ROOT/</pre> 6.30.02/tree/dataframe/src/RLoopManager.cxx:803 in void ROOT::Detail::RDF::RLoop Manager::Jit()>: Just-in-time compilation phase completed in 114.279227 seconds.

- > In principle, CROWN would be able to also track datatypes of quantities consistently and provide this information
- crashes for very large templates



>  $C_{++}$  templating is limiting factor in this case, compile times explode and even