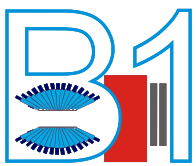
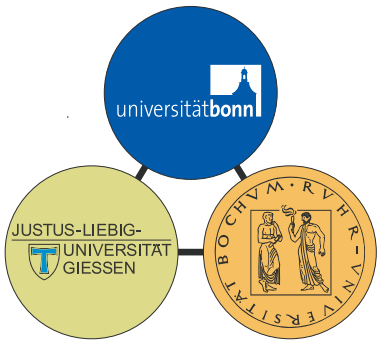
Object oriented data analysis at the BGO–OD experiment

Oliver Freyermuth
on behalf of the BGO–OD collaboration

PI Universität Bonn

18th September, 2015

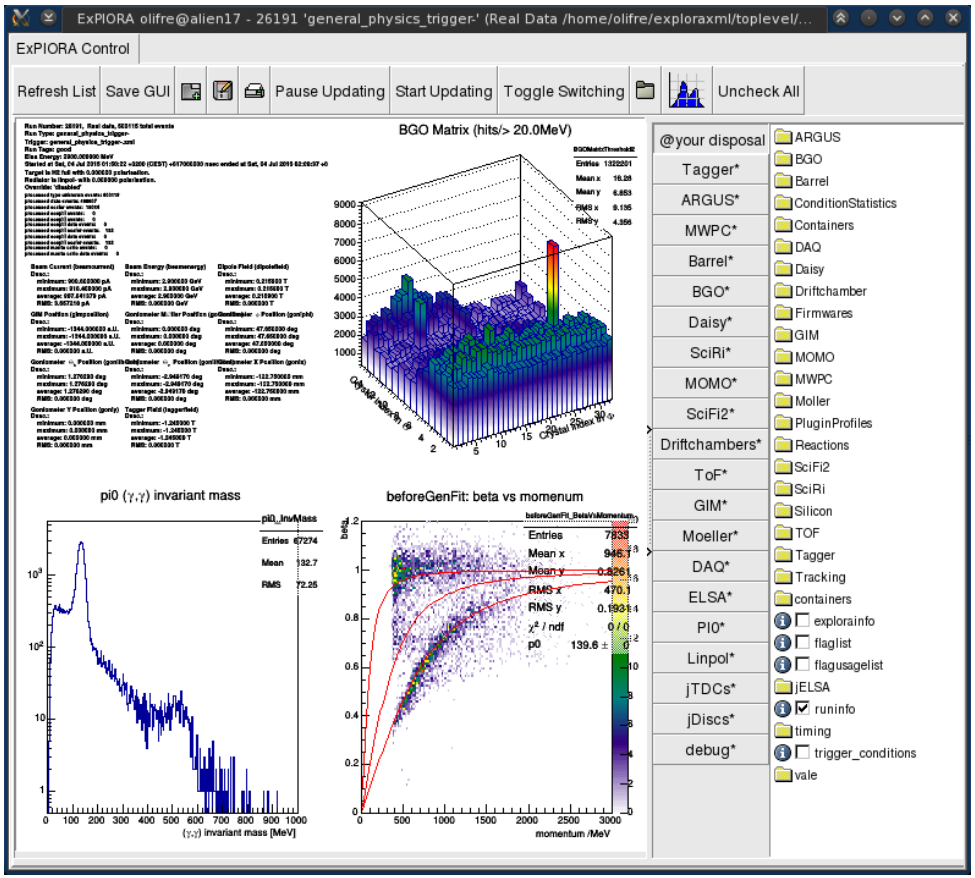




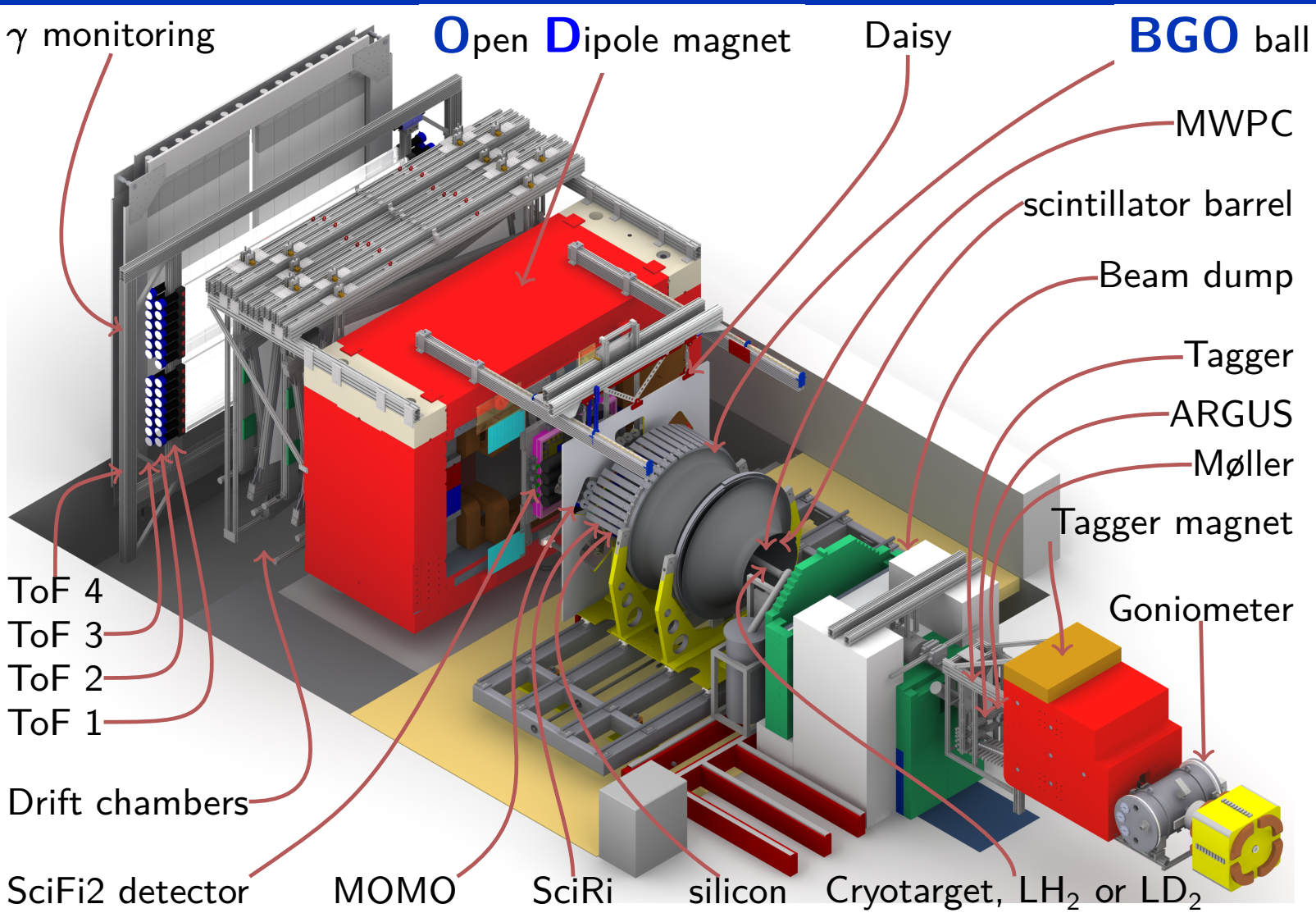
Supported by the

Overview

1. Introduction
2. Analysis Framework
   - Configuration via XML
   - ROOT Object I/O
   - Generalized functionality
   - EVEnt Display
   - Database Interfaces
3. Tools
   - Code quality
   - Qt ROOT
4. Conclusion



The BGO–OD Experiment



Challenge: Large number of very diverse detectors

universitätbonn

The BGO–OD Collaboration

74 members from 18 institutes

- 27 people ever contributed to software (incl. many Bachelor and Master students)
- Diverse levels of C++ knowledge
- No full-time developers

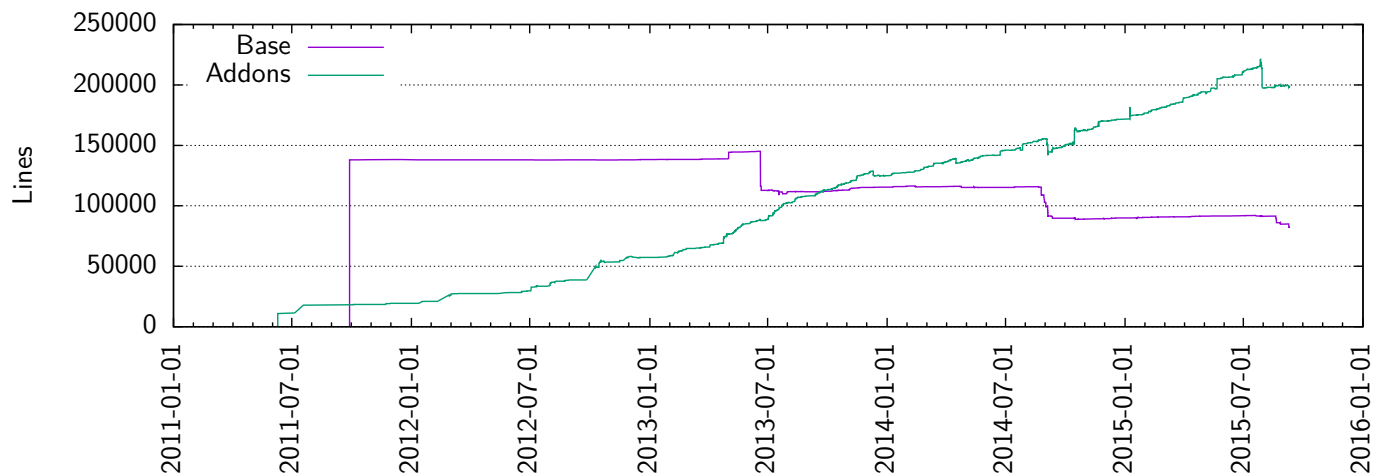
Starting point:

Analysis software based on **Ex**tended **Pl**uggable **O**bjectoriented **R**OOT(ified) **A**nalysis developed in Bonn by the CBELSA/TAPS collaboration (version from 2011):

C. Schmidt, A. Suele, J. Hartmann, U. Thoma et al.

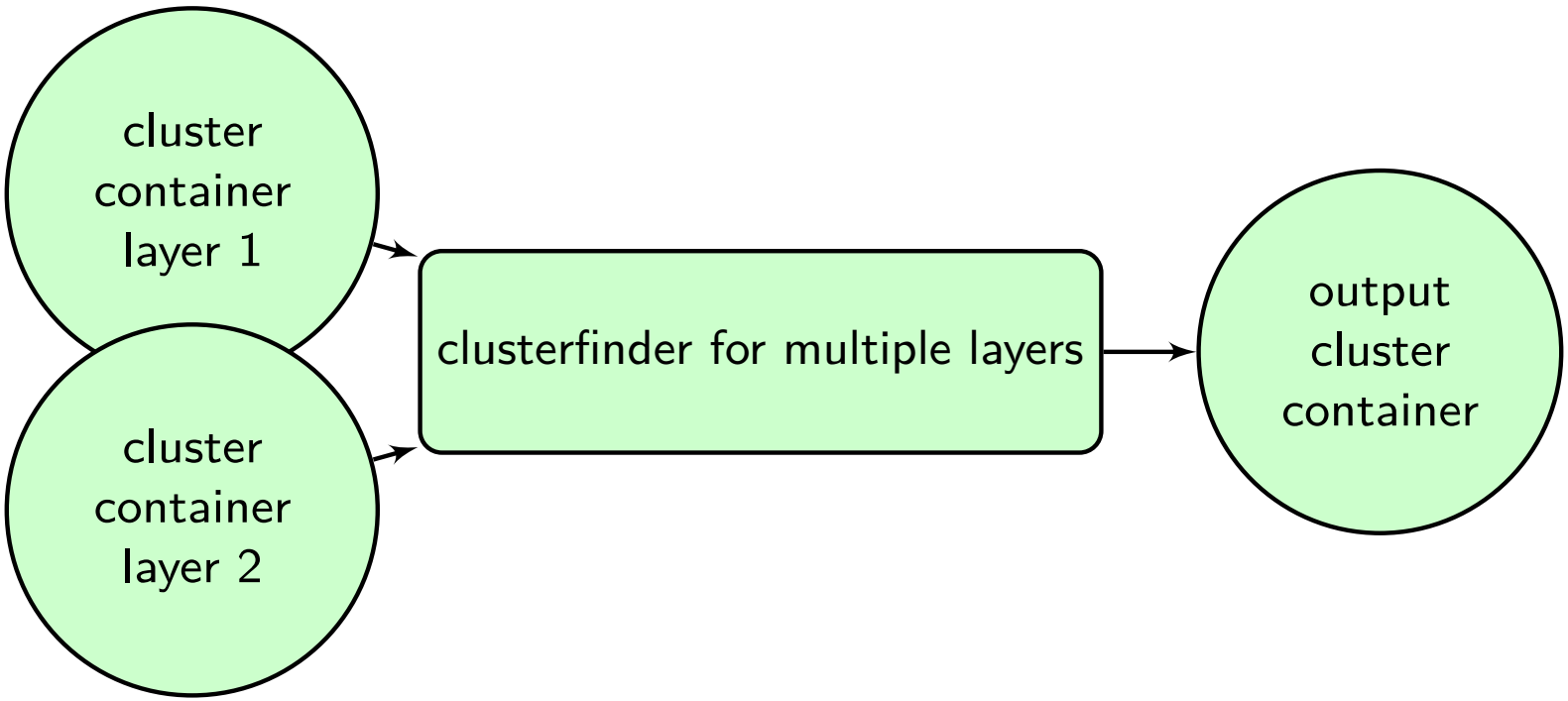
Analysis Framework ExPIORA

- Strongly relying on **ROOT**, bindings to **Geant4**, **Geant-VMC** and **VGM** for simulation, interface to **Genfit2**
- **Pluggable**: Plugins for decoding, preanalysis,. . .
- Easy configuration via **XML** files
- Documentation: **Doxygen** (+ in-repo LaTeX 2_ε documents)
- Build system: **CMake**
- Compatibility: ROOT 5 and 6 (mostly) on Linux, MacOS X



ExPIORA: Plugin and container system

- Analysis structured in plugins with small, defined tasks
- Data objects in `TClonesArray` containers
- Configuration parameters via XML



Dynamic XML configuration

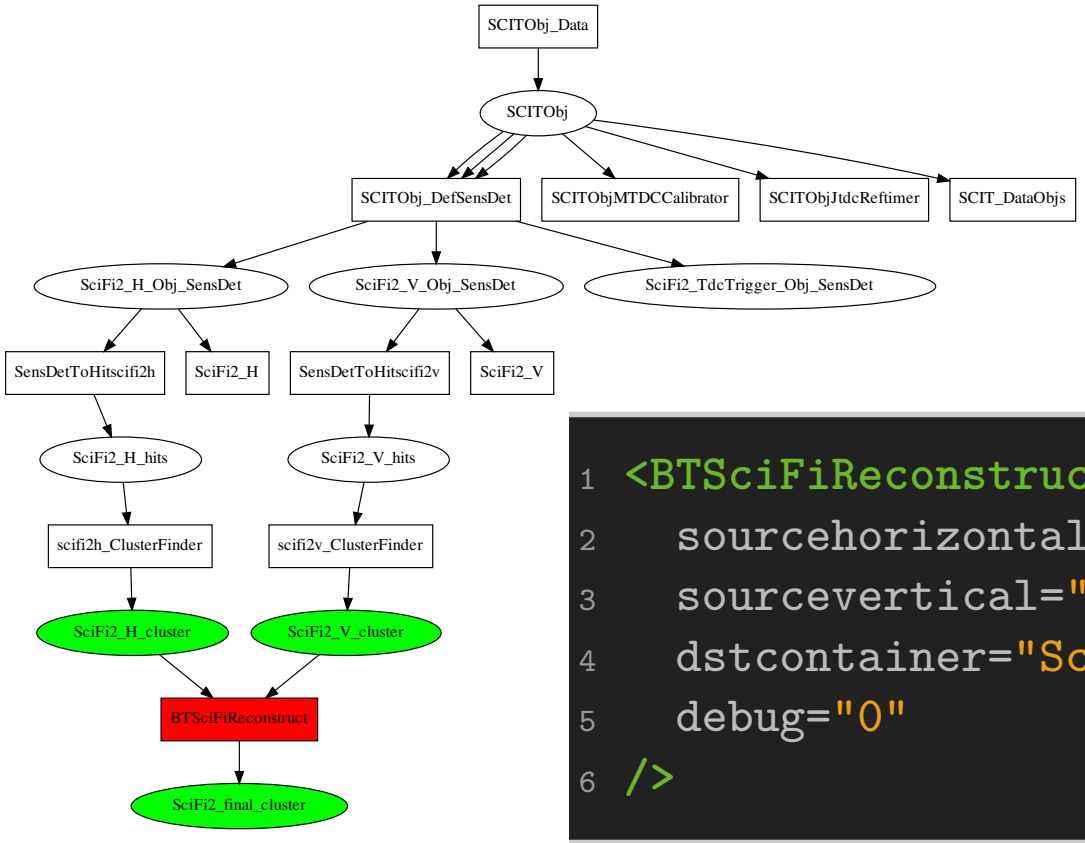
- XML-parsing: `TSAXParser` (callback parser)
- Tags handled with `TClass::New()` (autoloading)

```
<BTSciFiReconstruct
  parseonfulfilled="${RunNumber}&gt;24041"
  sourcehorizontal="SciFi2_H_cluster"
  sourcevertical="SciFi2_V_cluster"
  dstcontainer="SciFi2_final_cluster"
  debug="0"
/>
```

Heavily extended

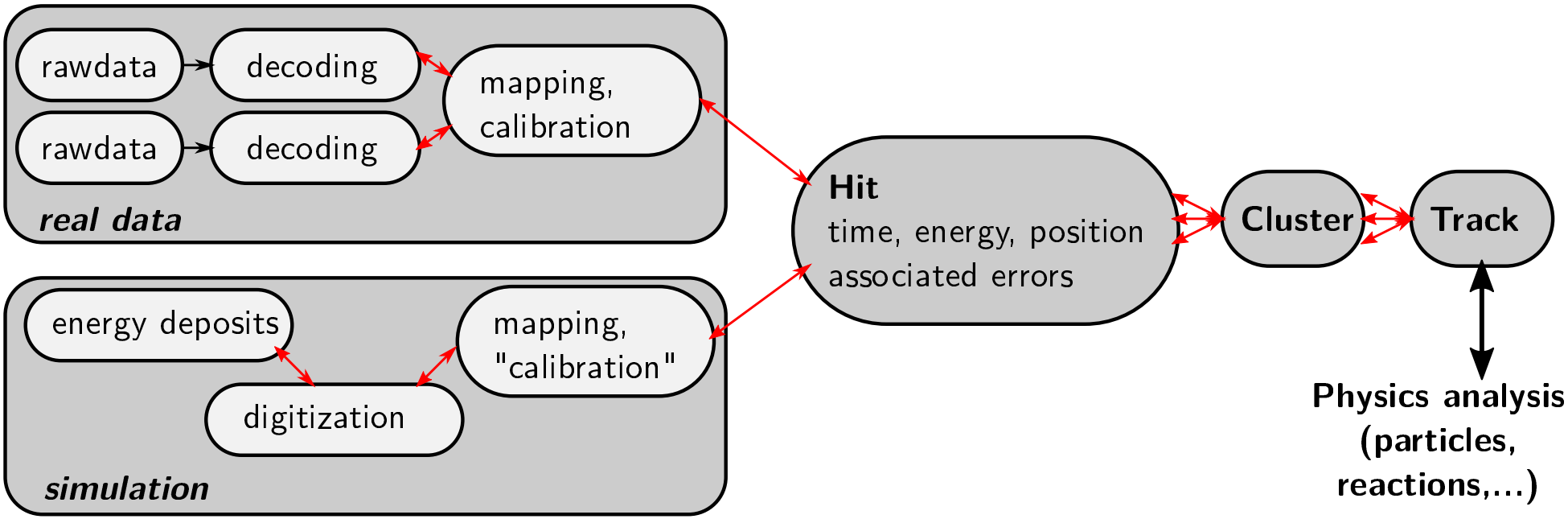
- scoped variables
- parameter evaluation (`TFormula`, `TInterpreter`, shell)
- (conditional) *include* functionality

Plugin and container graph

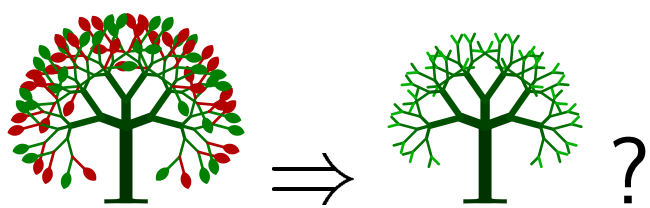



```
<BTSciFiReconstruct
  sourcehorizontal="SciFi2_H_cluster"
  sourcevertical="SciFi2_V_cluster"
  dstcontainer="SciFi2_final_cluster"
  debug="0"
/>
```

Stepwise analysis



- During reconstruction: All references needed both ways
- Streaming to disk: Need to "cut off" information, e.g. only tracks as input for physics analysis



Selective writeout of data objects

Naïve approach: Write out tracks only, `Clear()` the `TClonesArrays` of hits & clusters before.

But. . . ☹

- `TObject` will not remove itself from `TRefTable` on `Clear` (`ROOT-7249`)!
- We reset the object count in `TProcessID` per event
  ⇒ object-UIDs are only unique per-event (which is ok)

⇒ On readback, not all object-UIDs will come back:
references may point to objects from previous events! ☹

Solution

Manual pointer pruning and regular runtime checks (custom classes inheriting `TRef`, `TRefArray`).

ROOT object I/O

Good experiences

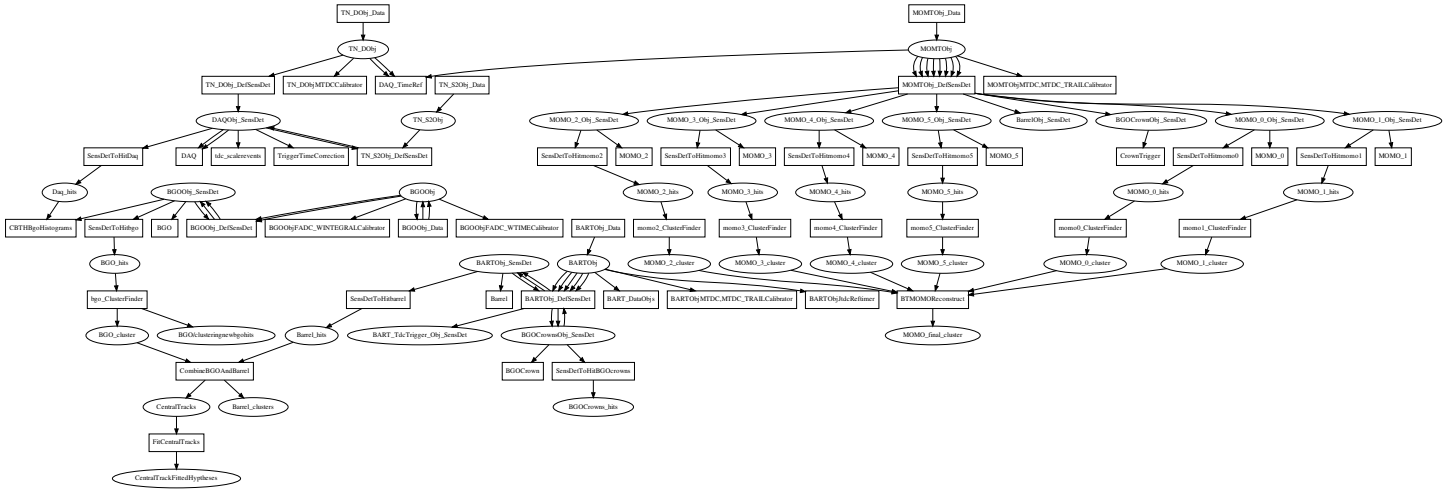
- Good performance of `TClonesArray`'s `ConstructedAt` when used consistently.
- Streaming and schema evolution work extremely well.

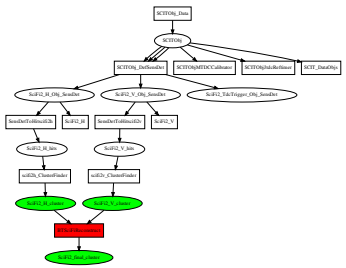
Bad experiences

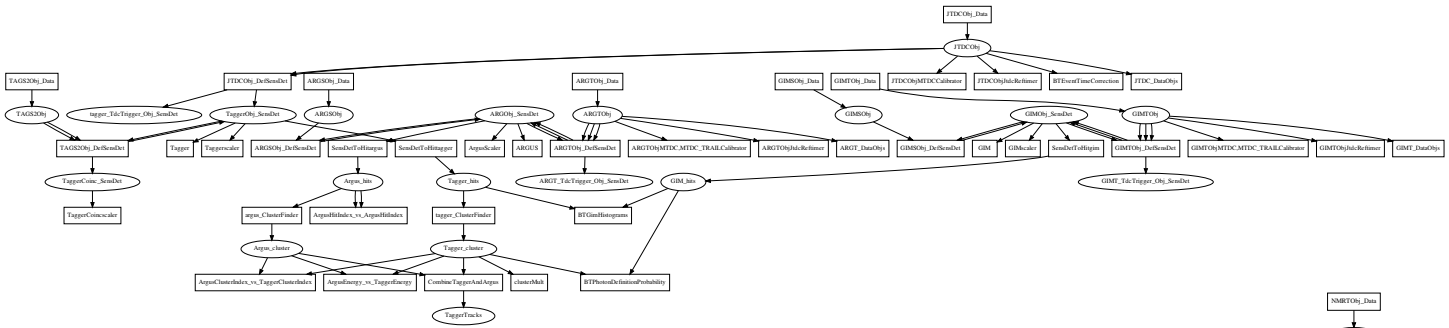
- Different implementations of `Clear` needed depending on usage of `TClonesArray` (`ConstructedAt` or placement-`new`), even classes which are part of ROOT mix them!
- `TTree::Draw` fails on referenced objects (`ROOT-7076`)
- `EXEC:`-feature can only be used on classes named `TRef.*` (`ROOT-7052`)

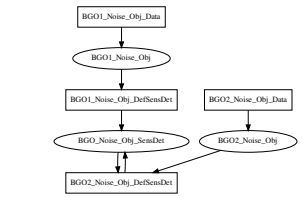
⇒ Examples for efficient use of ROOT object I/O could help.

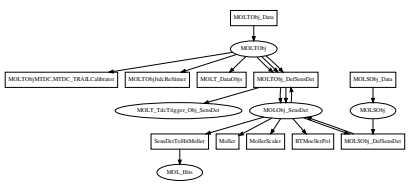
Plugin and container graph (*very incomplete*)

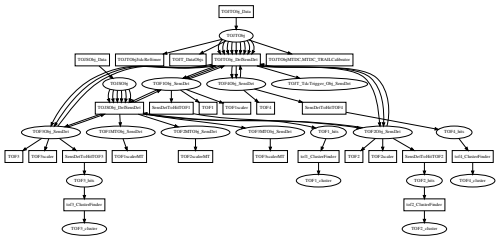


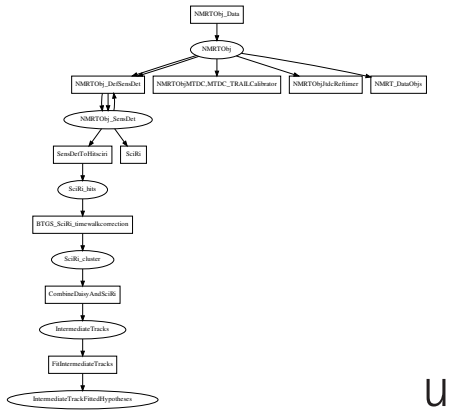


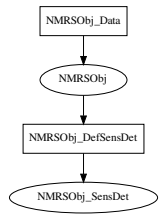












Generalized functionality: Interpreted Conditions

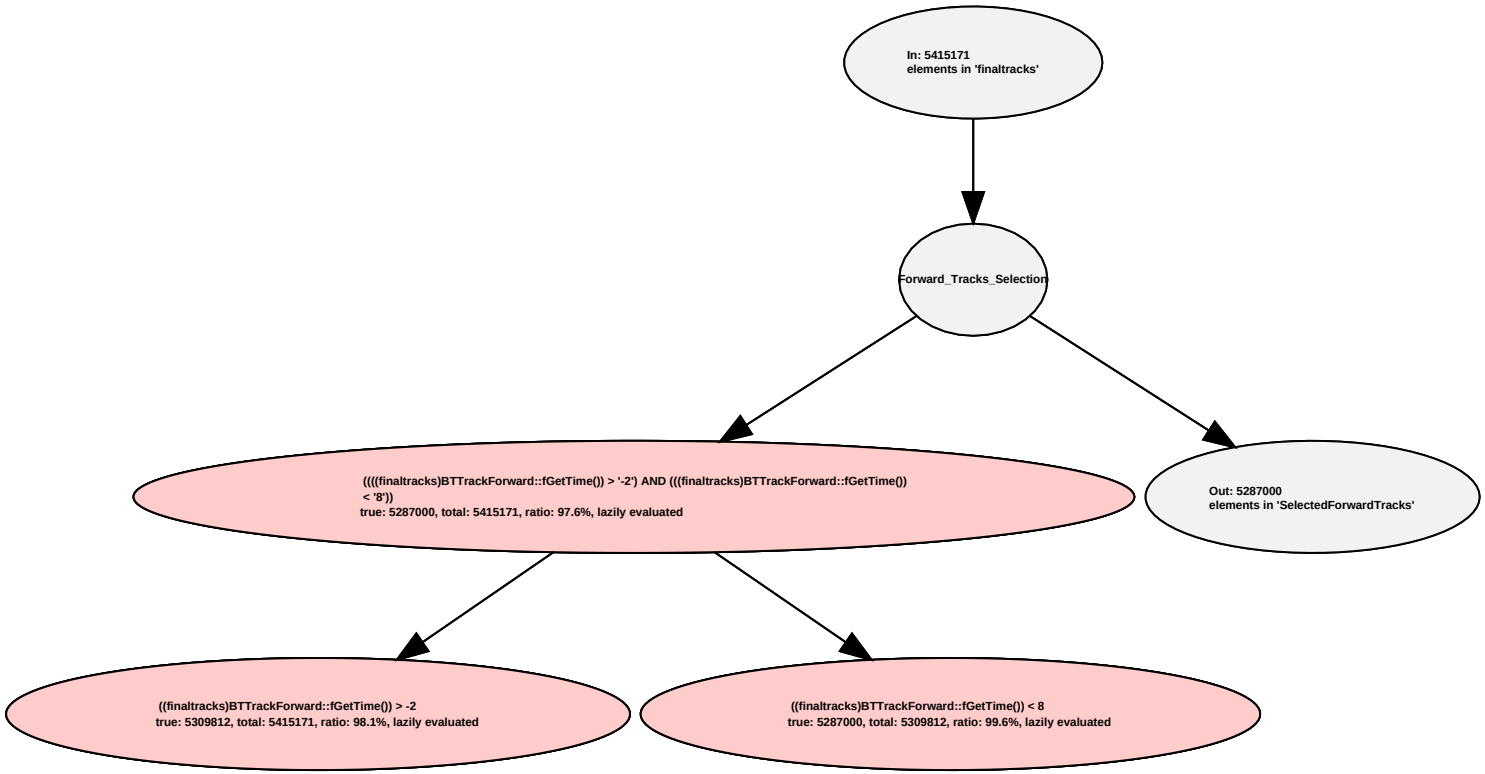
Many plugins just filter data and / or fill histograms.

```
<BTFilterContainer
  name="Forward_Tracks_Selection"
  srccontainer="ForwardTracks"
  dstcontainer="SelectedForwardTracks">
  <BTInterpretedConditionRange
    method="fGetTime"
    valuefrom="-2"
    valueto="8"
    />
</BTFilterContainer>
```

BTInterpretedCondition ***impossible without reflection!***

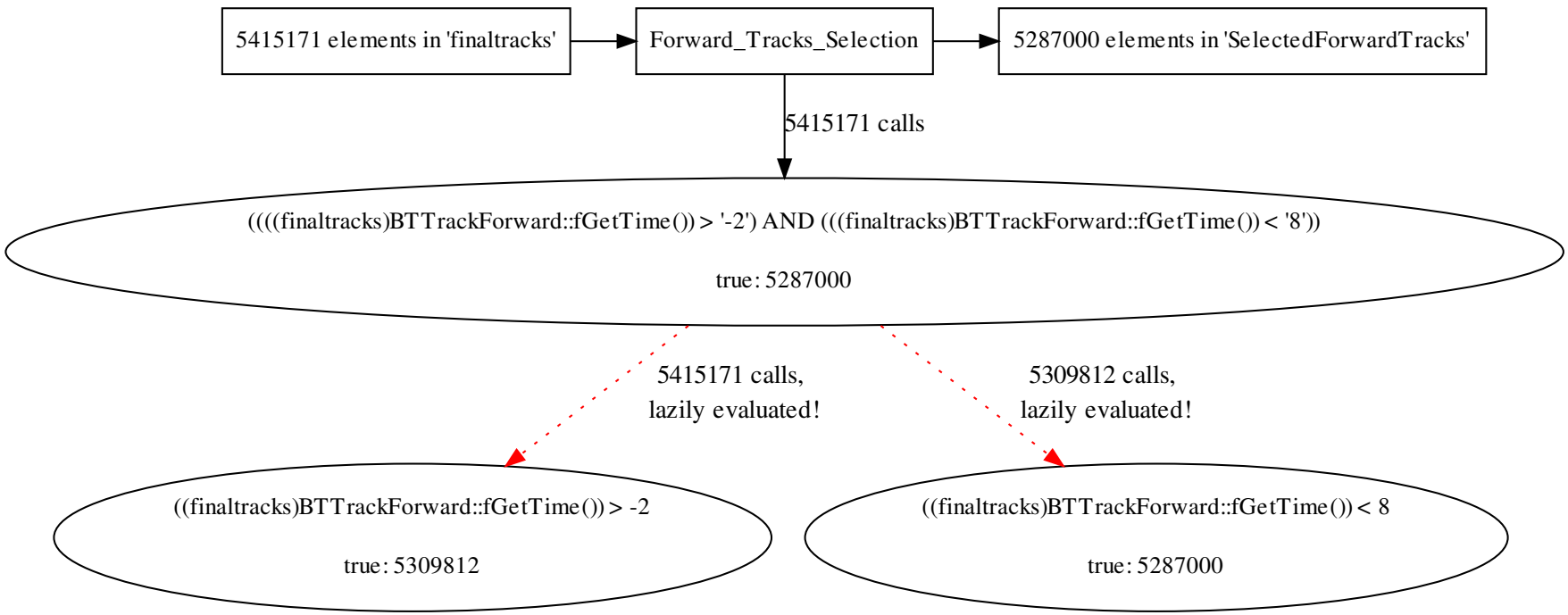
- wraps TMethodCalls and custom operators
- Tree-like condition-structure possible (*lazy evaluation*)
- **Statistics collection**

Generalized functionality: Data filtering



- Statistics stored in a dataobject (`Draw()`able, `Merge()`able)
- Implemented via `TGraphStruct` / `TGraphNode` (readability? ☹)
- Export to graphviz

Generalized functionality: Data filtering



- Statistics stored in a dataobject (`Draw()`able, `Merge()`able)
- Implemented via `TGraphStruct` / `TGraphNode` (readability? ☹)
- Export to graphviz

Generalized functionality: Histogramming

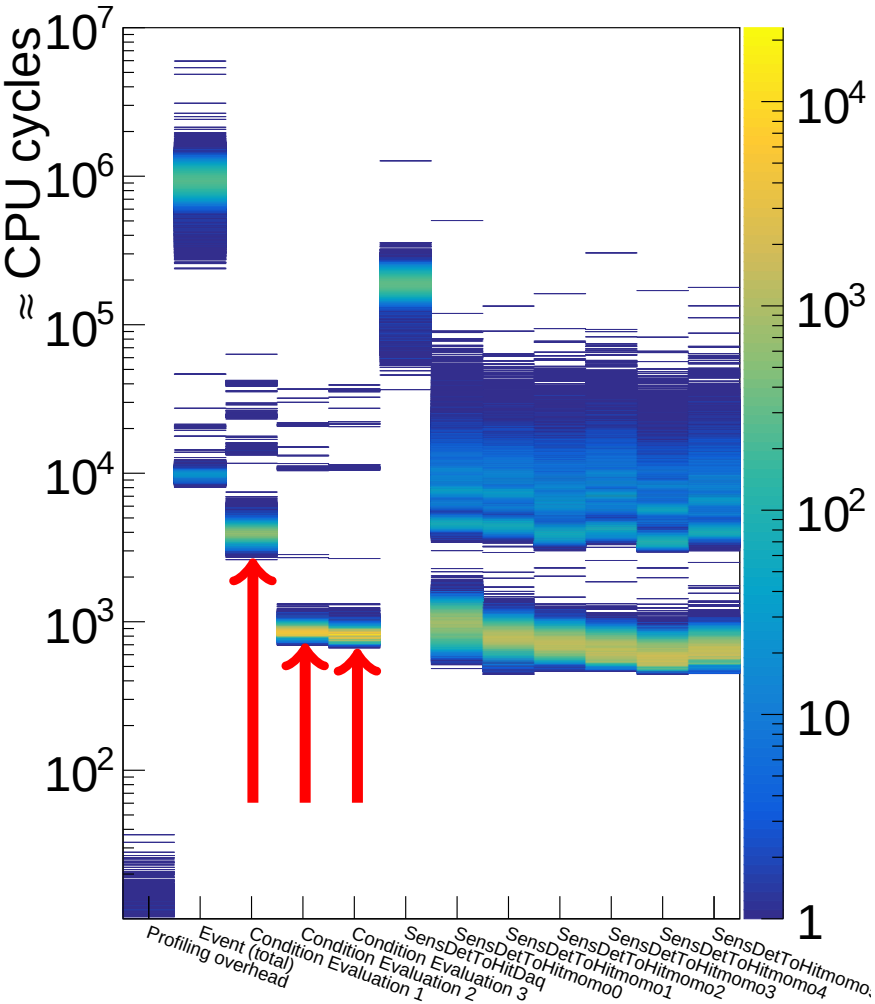
Histogram definition and filling via XML

- Most `TH1`-derived classes wrapped (Interfaces inconsistent ☹)
- Filtering, scaling, calculations, custom iterations possible
- Set of axis-classes / named axises for easy variable binning (e.g. from geometry)

Especially for multidimensional histograms...

Heavily relies on `TMethodCall` $\Rightarrow$ **Performance**?

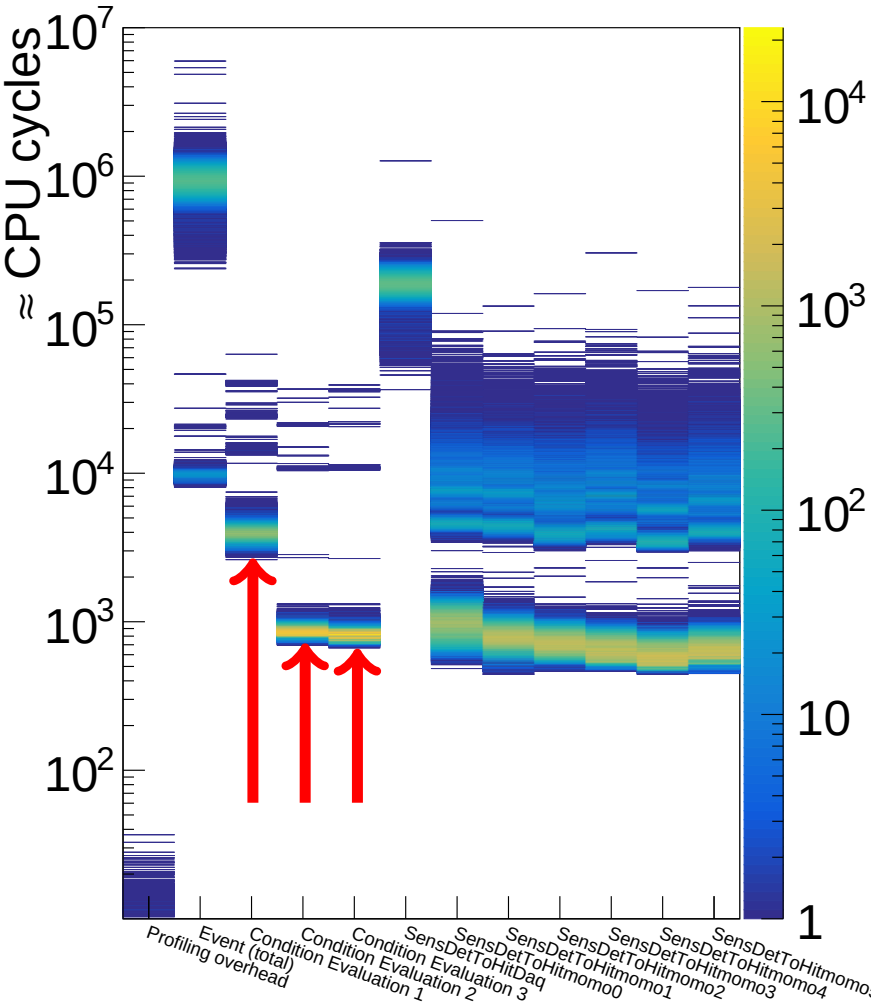
TMethodCall performance



Profiling Measurement

- CPU cycles estimated using `RDTSCP` instruction (fallback to `CPUID` & `RDTSC`)
- profiling overhead subtracted
- Agrees with `clock_gettime()`

TMethodCall performance



TMethodCall

- slowdown on first call (instruction cache?)
- performance could be better

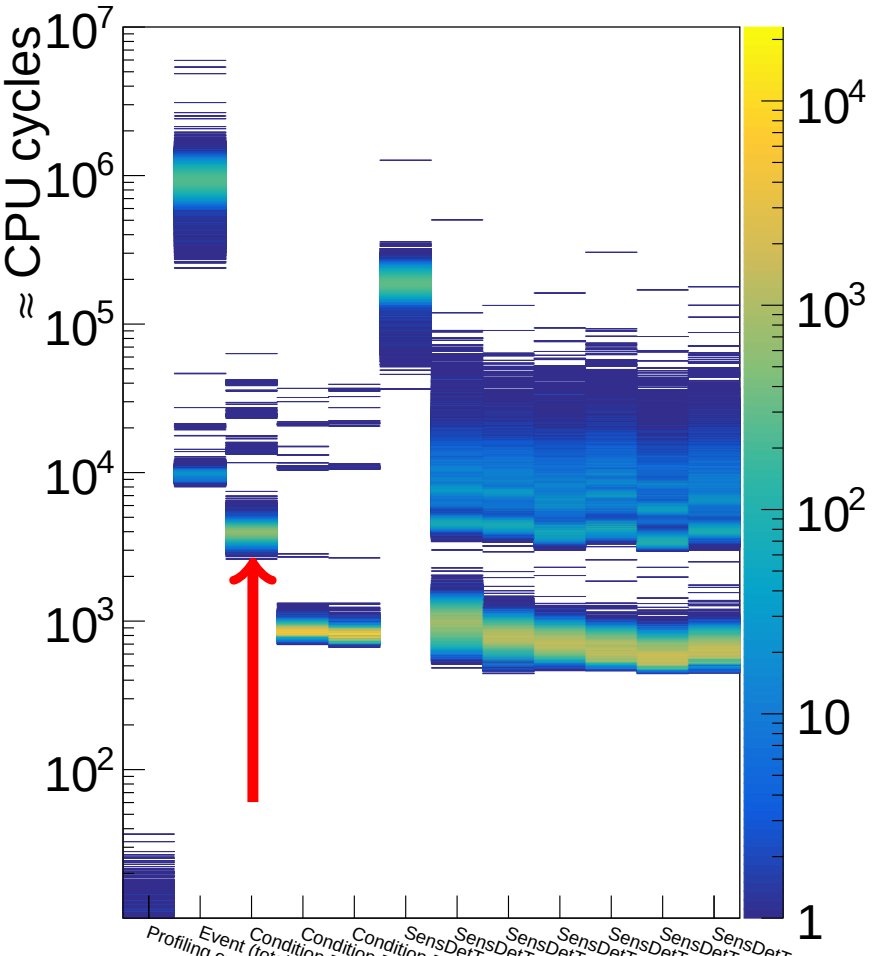
Instead: Function Dictionary

- Limited set of compiled autogenerated function pointer wrappers
- Only functions without arguments handled
- Return types: `Long_t`, `Double_t`, `const char*`

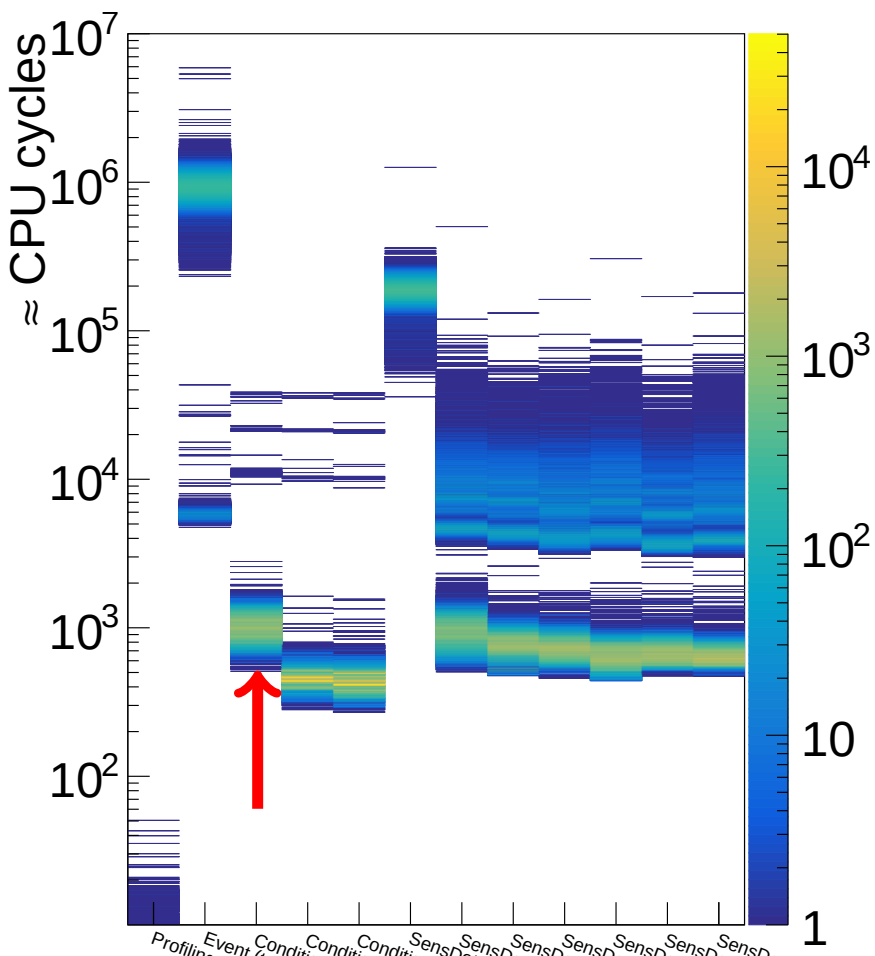
universitätbonn

Performance comparison (ROOT 5)

`TMethodCall`



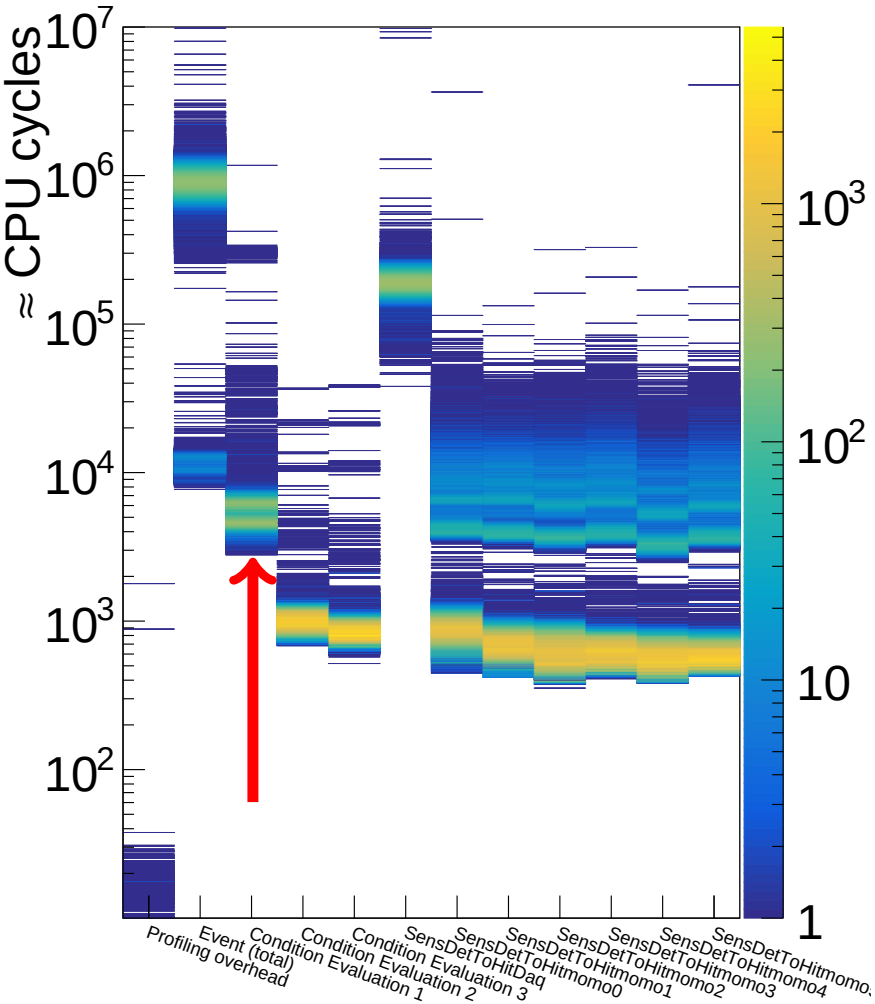
`functionDictionary`



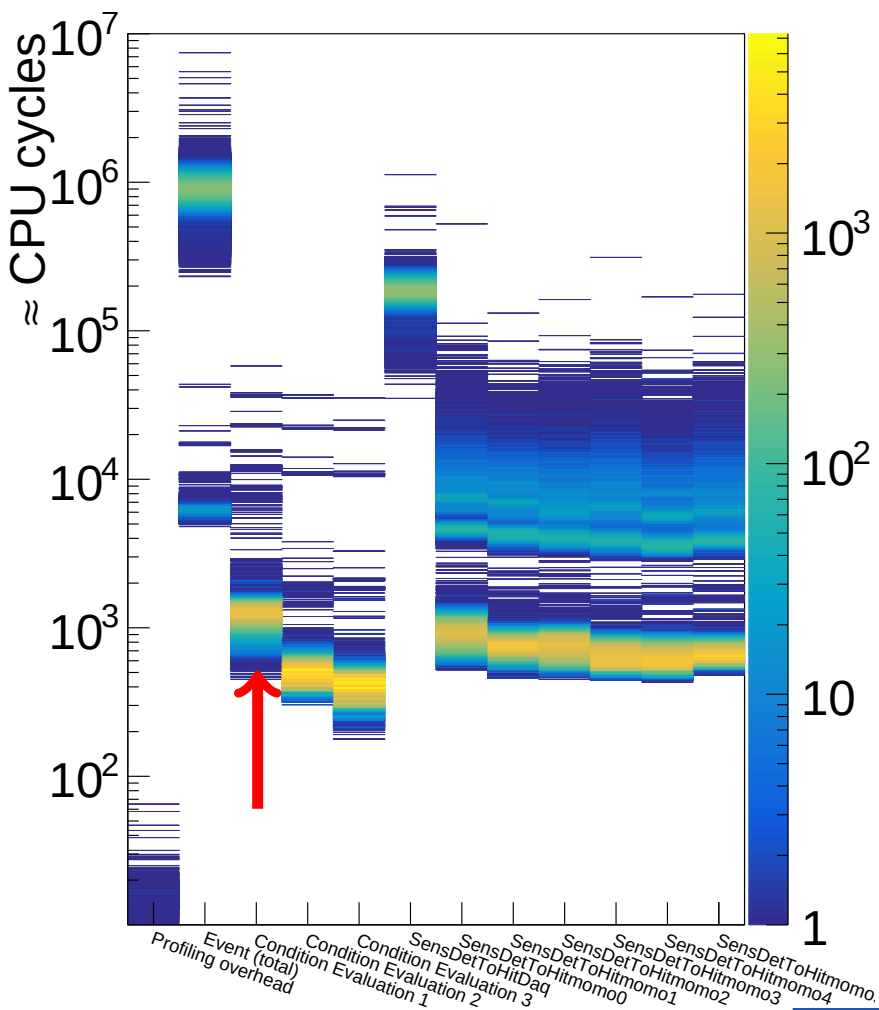
But: ROOT 6 (with **JIT**ed wrappers) should perform equal to the `functionDictionary`!

Performance comparison (ROOT 6)

TMethodCall



functionDictionary



universitätbonn

Performance comparison (ROOT 6)

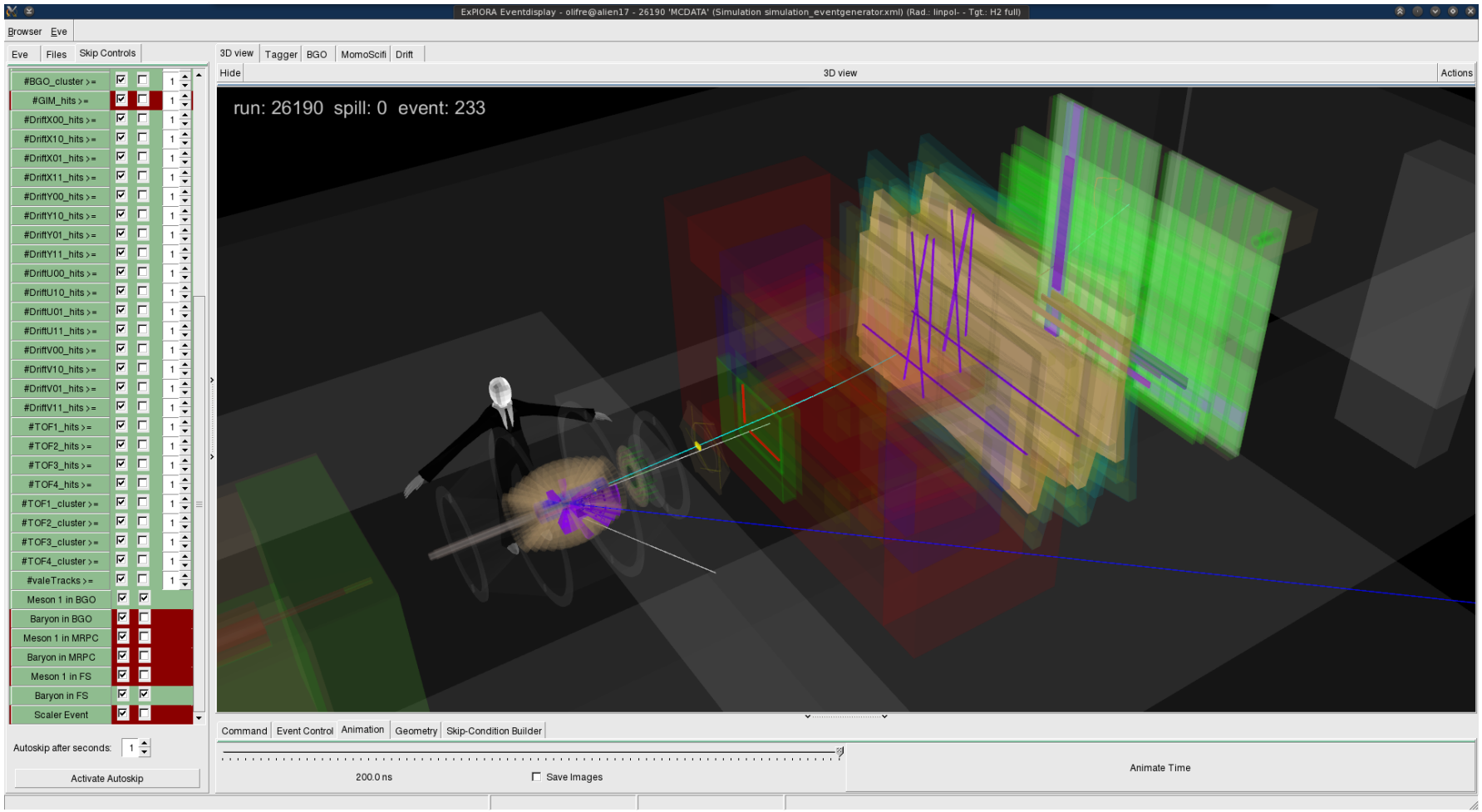
Feature-duplication should be removed (error-prone)...
...ask callgrind!

ROOT 6: Function call itself is less than 50 % of cycles!

- During `TMethodCall::Execute`, a new `cling::Value` is constructed and destructed
- `cling::Value::determineStorageType` eats most of the time (*re-guessing* return value type)

$\Rightarrow$ This could be improved! (not reported yet on Jira)

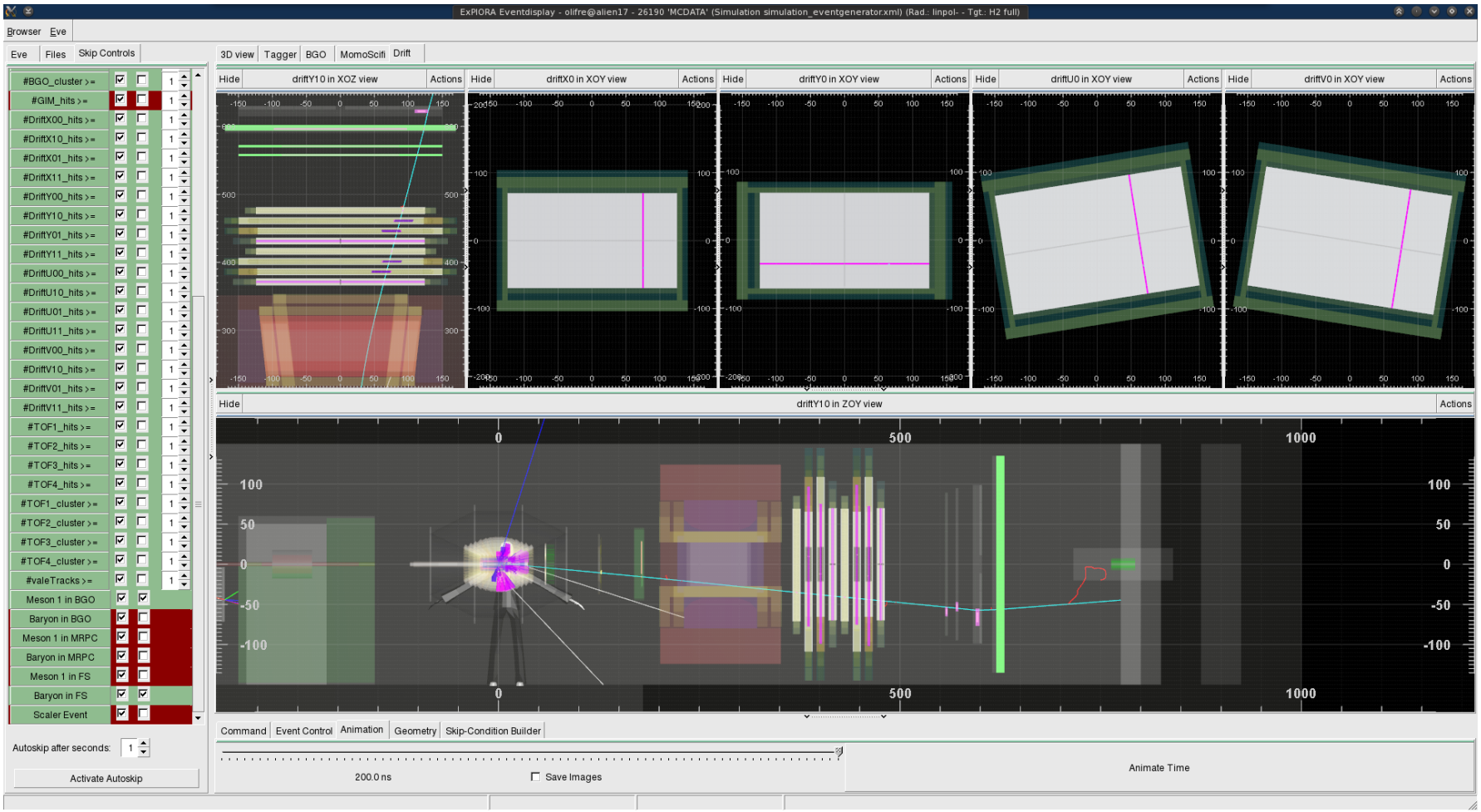
EVEnt display for simulated and real data



- Event selection with GUI-elements autocreated from interface functions or conditions
- Accurate `TGeo`-based geometry description

universitätbonn

EVEnt display for simulated and real data



- GUI elements created via XML
- Can be used online during datataking

Abstract database interfaces

Online databases: PostgreSQL

- Slowcontrol Database
- Run Database
- Analysis Database (Calibrations, time-dependent constants,...)

⇒ During analysis, parts of all databases are needed!

But: Database replication is complex, 18 institutes, no expert.

Solution: Offline Databases: SQLite (all upstreamed)

- `TPgSQLServer` extended with large object support
- Implemented `TSQLiteServer` sharing `TSQLServer` interface
- Customized tool converting parts of online-DBs to SQLite-databases

Drop-in replacement for external collaborators!

Code Quality: Portability

Upstream as much as possible!

Packaging

Packaged (or helped out) for

1. genfit2
2. VGM
3. geant-vmc

for **openSUSE** (T. Zimmermann) and **Gentoo Linux** (myself).
Internally: Experimental MacPorts portfiles.

ROOT 6 migration

Upstreamed ROOT 6 support for build systems of these packages.

$\Rightarrow$ All our dependencies can be installed via distribution package managers.

Code Quality

Unit tests

- `check_explora` analyzes chosen set of runs, histograms compared (differences shown ordered by `Kolmogorov` distance)
  $\Rightarrow$ *Idea: Parallel `TBrowser`?*
- static analyzers: clang-static, cppcheck
- coverage testing
- CDash on 2 Linux distributions and with gcc & clang

Login All Dashboards Thursday, September 10 2015 18:46:02 CES

Explorabgoaddons

Dashboard Calendar Previous Current Next Project

No file changed as of **Monday, September 07 2015 - 03:00 CEST**

19 minutes ago: 2 warnings introduced on Linux-4.2.0-gentoo-x86_64-gcc-4.9.3-root6
20 minutes ago: 3 tests failed on Linux-4.2.0-gentoo-x86_64-clang-3.7.0-root6
20 minutes ago: 1 warning introduced on Linux-4.2.0-gentoo-x86_64-clang-3.7.0-root6
20 minutes ago: 50 errors introduced on Linux-4.2.0-gentoo-x86_64-clang-3.7.0-root6
29 minutes ago: 3 warnings introduced on Linux-3.11.10-164-hsag-x86_64-gcc-4.8-v4.50.0

See full feed

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
lisa1	Linux-4.2.0-gentoo-x86_64-clang-3.7.0-root6	1	0	0	0	2	0	0	382	Sep 07, 2015 - 15:20 CEST
lisa1	Linux-4.2.0-gentoo-x86_64-gcc-4.9.3-root6	1	0	0	0	2	0	0	382	Sep 07, 2015 - 15:20 CEST
b1daq7	Linux-3.11.10-21-pae-i686-gcc-4.8-v4.44.0	1	0	0	0	0	0	0	469	Sep 07, 2015 - 15:00 CEST
b1daq9	Linux-3.11.10-164-hsag-x86_64-gcc-4.8-v4.50.0	1	0	0	0	3	0	0	382	Sep 07, 2015 -
b1daq8	Linux-3.11.6-4-default-x86_64-gcc-4.8-v4.44.0	1	0	0	0	0	0	0	471	Sep 07, 2015 -
lisa1	Linux-4.2.0-gentoo-x86_64-clang-3.7.0-root6	1	0	0	0	2	0	0	382 +1	Sep 07, 2015 - 12:20 CEST
lisa1	Linux-4.2.0-gentoo-x86_64-gcc-4.9.3-root6	1	0	0	0	2	0	0	382 +1	Sep 07, 2015 - 12:20 CEST
b1daq7	Linux-3.11.10-21-pae-i686-gcc-4.8-v4.44.0	1	0	0	0	0	0	0	469 +1	Sep 07, 2015 - 10:00 CEST
b1daq9	Linux-3.11.10-164-hsag-x86_64-gcc-4.8-v4.50.0	1	0	0	0	3	0	0	382 +1	Sep 07, 2015 - 10:00 CEST

Reflection-based checks

Post-compile time warnings

- Class-version of dataobjects
- Common naming conventions
- Force reimplementation of specific interface methods
- Header guards
- XML parameter initialization
- Existence of `ClassImp`
- Uniqueness of enum values

. . . and more!

```
vmc/include/LinkDef.hh:42: warning: trailing + missing in LinkDef for class BIVmcElectronBeamParameters

BTForwardSpectrometerMissingMass.cpp:72: warning: suspicious number 938.27 in attr->fSetDefault(938.27);
        probably proton mass meant, so use BTParticlePDG::proton.Mass()

BTReactionFinalState.cpp:192: error: Class 'BTReactionFinalState' has a property \
        'maxignorablextratracks' which is not initialized!
```

Qt ROOT applications

Qt ROOT applications

Qt ROOT works mostly well, but... ☹ (ROOT-4932)

at least 3 diverging codebases exist (commits in 2014)!

1. ROOT **in-tree** (packaged for most distributions),
   no Qt 5 support
2. "primary" **sourceforge** repository, unpackaged
   seems to have some bugfixes, Qt 5 support
3. **github** fork, unpackaged
   mixture of the former two

What should be used?

Repackaging is (almost) unmaintainable, file collisions guaranteed!

Summary and Conclusion

ROOT facilitates physics with limited manpower!

ExPIORA is an extremely ROOT-centric analysis framework

- Flexible XML configuration
- ROOT object I/O
- Interpreted-once approach allows generalized functionality
- "Education" of users with custom warnings

$\Rightarrow$ All this would be **impossible** without reflection.

Even though I used many tiny ☹ ...
...**ROOT is fun to work with!** ☺

Thanks to the developers for integrating patches upstream!

Thank you for your attention! ☕

Data acquisition

- Local Event Builders read out raw data
- Arrays of bytes sent over the network
- Event is assembled into “byte arrays” which are *streamed*
  - to a ROOT-file
  - over network for online processing

We store our rawdata in slightly compressed ROOT files:

- Tree with event-based rawdata (objects in `TClonesArray`)
- “Tableevent” objects with common information by run.

ExPIORA: Plugin and container system

- Analysis structured in plugins with small, defined tasks
- Data objects in `TClonesArray` containers
- Configuration parameters via XML

```
<BTSciFiReconstruct
  sourcehorizontal="SciFi2_H_cluster"
  sourcevertical="SciFi2_V_cluster"
  dstcontainer="SciFi2_final_cluster"
  debug="0"
/>
```

Selective writeout: Pointer pruning

Derived classes `CBTRef`, `CBTRefArray`

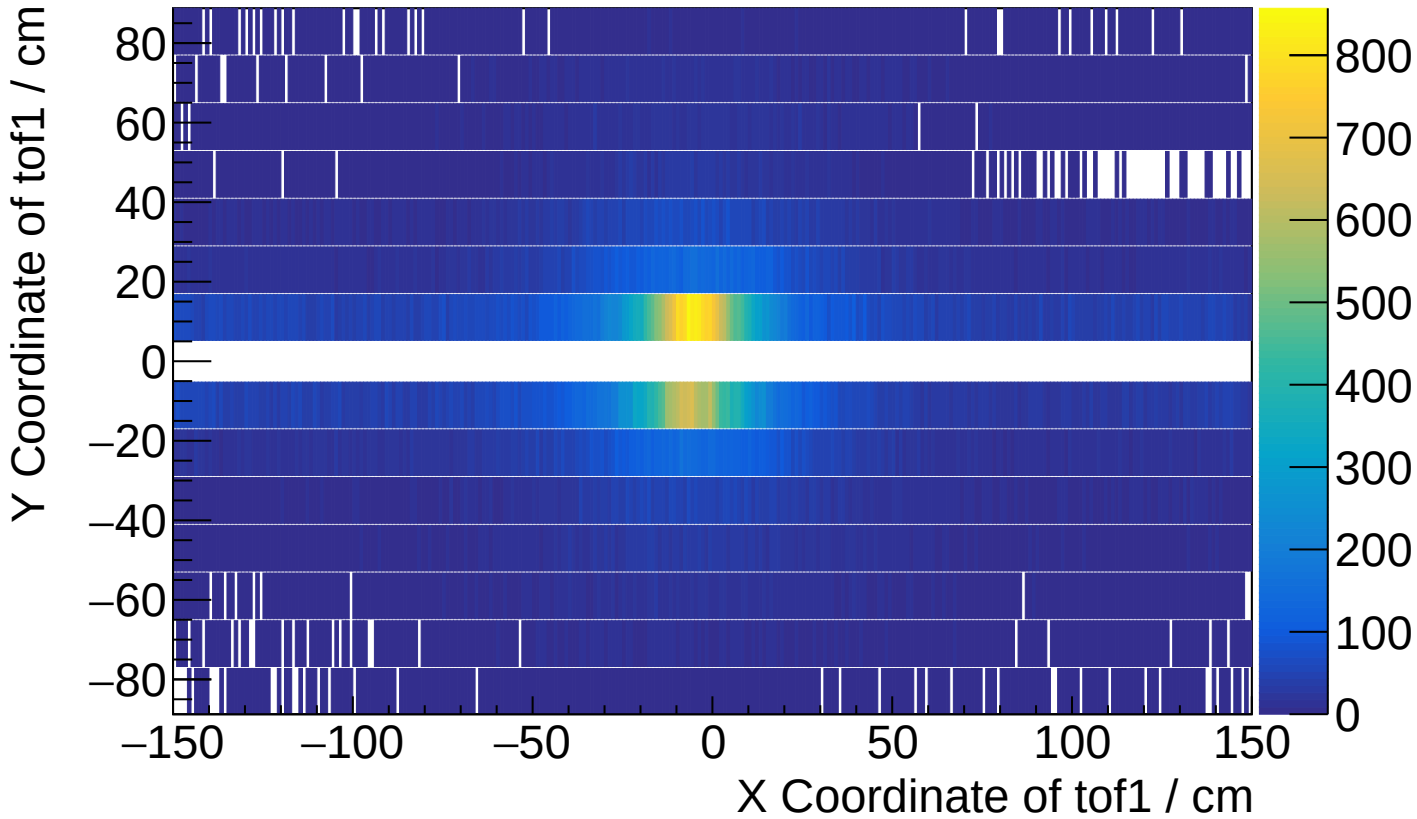
- Regular intervals, between all plugins:
  - Check for cross-event references
  - Check all referenced objects still exist before writeout
  - Streaming of all containers to memory, checksumming of buffers to check for unwanted modifications
- Before streaming: Ref'fed objects are checked for existence.
- `Clear`'ing unneeded `TClonesArray`.
- All `CBTRef` / `CBTRefArray` re-check their reference targets ⇒ Pruning (UID set to 0).
- After streaming: Check that all references have been `Clear`'ed.

Problem solved: User can select a storage-level (e.g. TrackLevel), `TRef` cut off automatically: ⇒

Generalized functionality: Histogramming

```
<BTHFlexiHisto
  parseonset="ToF1_Present"
  prefix="ToF1"
  histoname="ClusterPosition"
  folder="TOF/clustering"
  histotitle="cluster position"
  histotype="TH2D"
  x1container="TOF1_cluster"
  x1method="fGetPosition.X"
  y1container="TOF1_cluster"
  y1method="fGetPosition.Y"
  xaxis="BTHAxisDetectorCoordinates_tof1,X"
  yaxis="BTHAxisDetectorCoordinates_tof1,Y"
/>
```

Generalized functionality: Histogramming



Histogram

binning automatically created from geometry description

Performance comparison (ROOT 6) ☹

Naïve expectation

ROOT 6 (with **JIT**ed wrappers) should perform equal to the `functionDictionary`!
(*want to get rid of my extra implementation*)

#	CEst	Hex		Assembly Instructions
55 C669		0f 84 a1 00 00 00	je	55c710 <TClingCallFunc::exec_with_valref_return...
...				...
55 C6B2		e9 63 ff ff ff	jmpq	55c61a <TClingCallFunc::exec_with_valref_return...
55 C6B7	0.67	48 8b 55 00	mov	0x0(%rbp),%rdx
55 C6BB	0.38	4c 8d bc 24 70 01 00	lea	0x170(%rsp),%r15
55 C6C2		00		
55 C6C3	0.09	4c 89 ee	mov	%r13,%rsi
55 C6C6	0.09	4c 89 ff	mov	%r15,%rdi
55 C6C9	1.64	e8 c2 39 f7 ff	callq	4d0090 <cling::Value::Value(clang::QualType, cl
	32.82			465061 call(s) to 'cling::Value::Value(clang::QualType, cling::Interpreter&)' (libCling.so.6.05.01)
55 C6CE	0.09	4c 89 fe	mov	%r15,%rsi
55 C6D1	0.09	4c 89 e7	mov	%r12,%rdi
55 C6D4	1.05	e8 77 f1 f5 ff	callq	4bb850 <cling::Value::operator=(cling::Value&&)...
	2.95			465061 call(s) to 'cling::Value::operator=(cling::Value&&)' (libCling.so.6.05.01)
55 C6D9	0.09	4c 89 ff	mov	%r15,%rdi
55 C6DC	1.05	e8 4f 2e f5 ff	callq	4af530 <cling::Value::~Value()@plt>
	0.55			465061 call(s) to 'cling::Value::~Value()' (libCling.so.6.05.01)
55 C6E1	0.09	48 85 db	test	%rbx,%rbx
55 C6E4	0.38	74 0c	je	55c6f2 <TClingCallFunc::exec_with_valref_return...
55 C6E6		48 8b 03	mov	(%rbx),%rax
55 C6E9		48 89 df	mov	%rbx,%rdi
55 C6EC		ff 90 e0 01 00 00	callq	*0x1e0(%rax)
55 C6F2	0.09	48 8d 54 24 1c	lea	0x1c(%rsp),%rdx
55 C6F7	0.09	4c 89 f6	mov	%r14,%rsi
55 C6FA	0.09	48 89 ef	mov	%rbp,%rdi
55 C6FD	1.34	e8 3e 9e f5 ff	callq	4b6540 <TClingCallFunc::exec(void*, void*) cons...
	38.47			465061 call(s) to 'TClingCallFunc::exec(void*, void*) const' (libCling.so.6.05.01)
55 C702	0.09	48 63 44 24 1c	movslq	0x1c(%rsp),%rax
55 C707	0.09	49 89 04 24	mov	%rax,(%r12)

Performance comparison (ROOT 6)

Naïve expectation

ROOT 6 (with **JIT**ed wrappers) should perform equal to the `functionDictionary`!
(*want to get rid of my extra implementation*)

Inside `TClingCallFunc::exec_with_valref_return`:

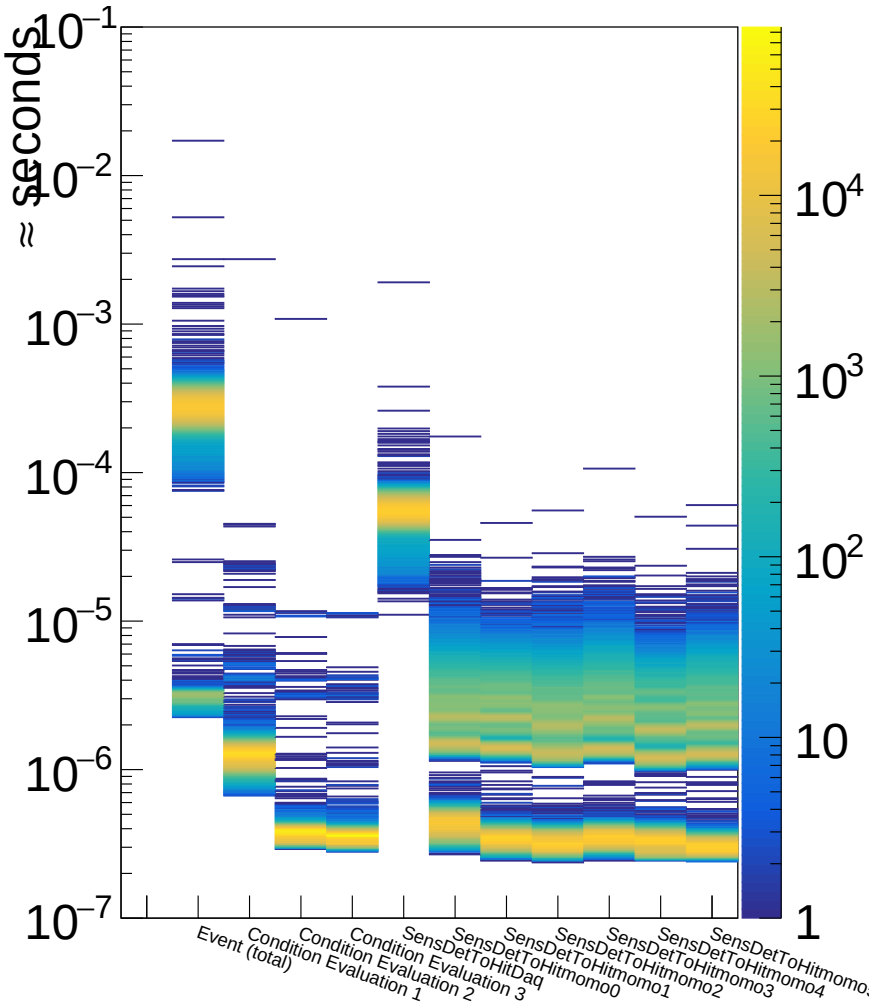
```
*ret = cling::Value(QT, *fInterp);
```

- New object constructed, moved, destructed per `TMethodCall`
  $\Rightarrow$ As expensive as the function call itself!
- Most time actually spent in
  `cling::Value::determineStorageType(clang::QualType)`

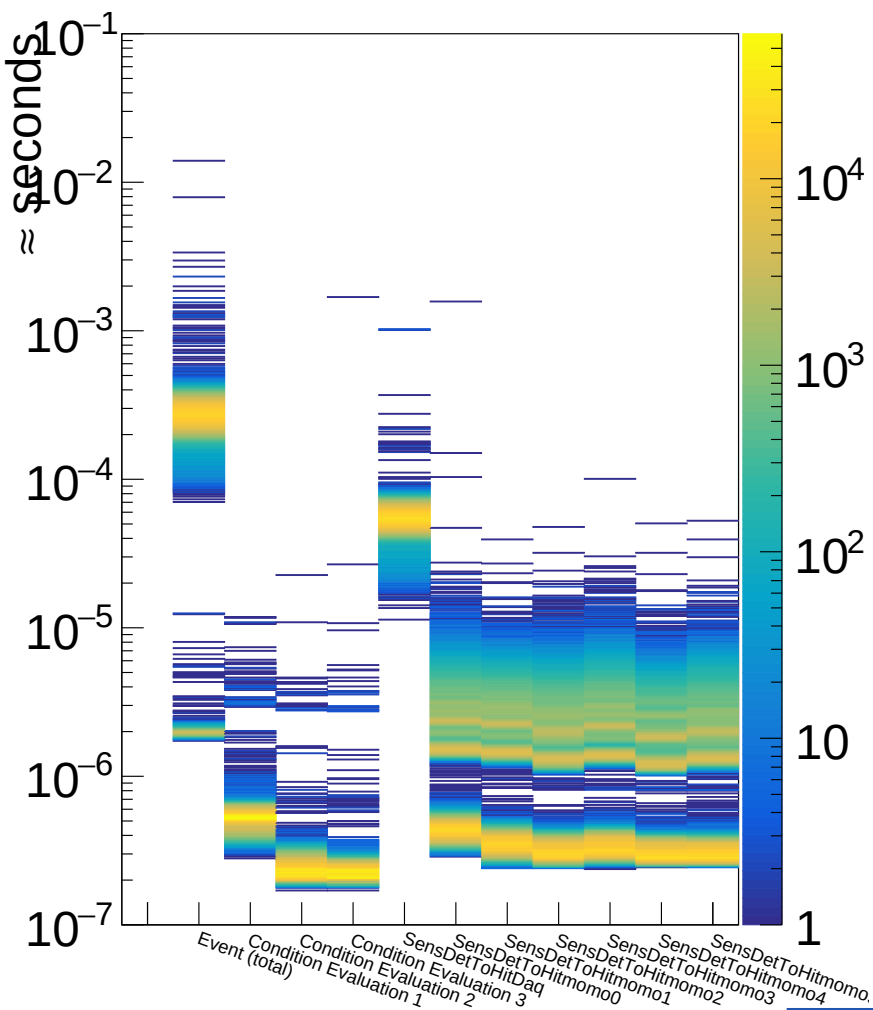
$\Rightarrow$ There is hope left!

Performance comparison (ROOT 5)

TMethodCall



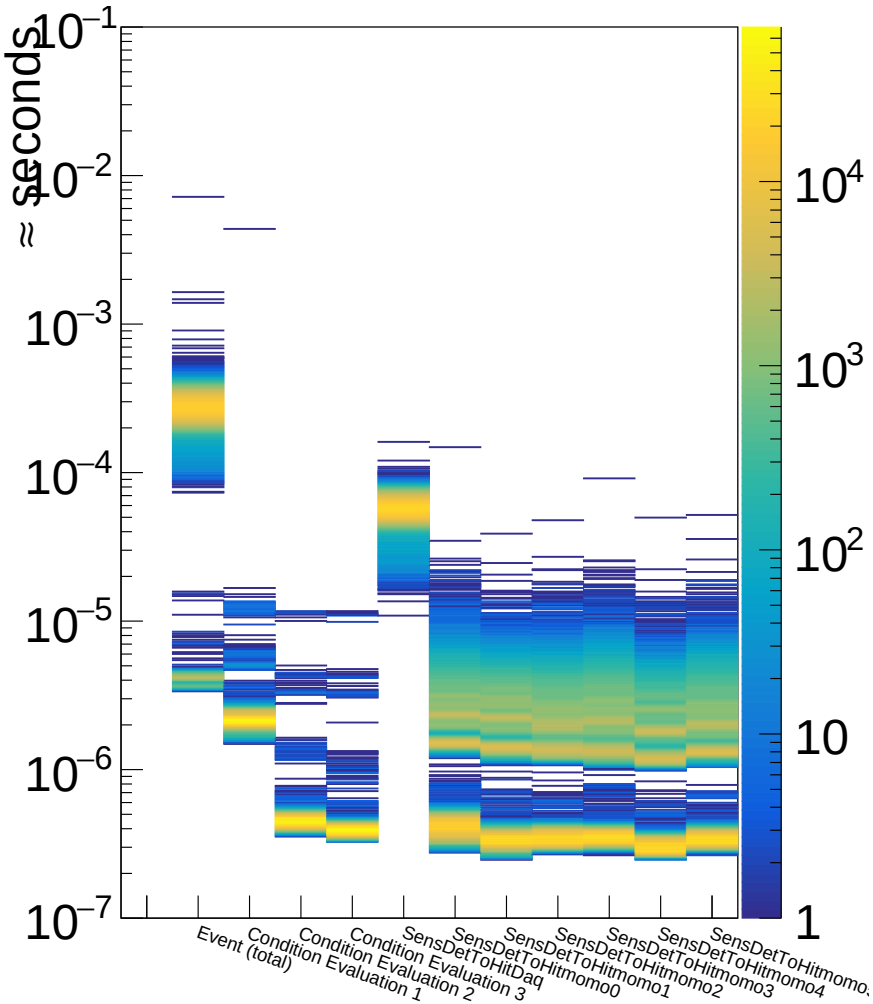
functionDictionary



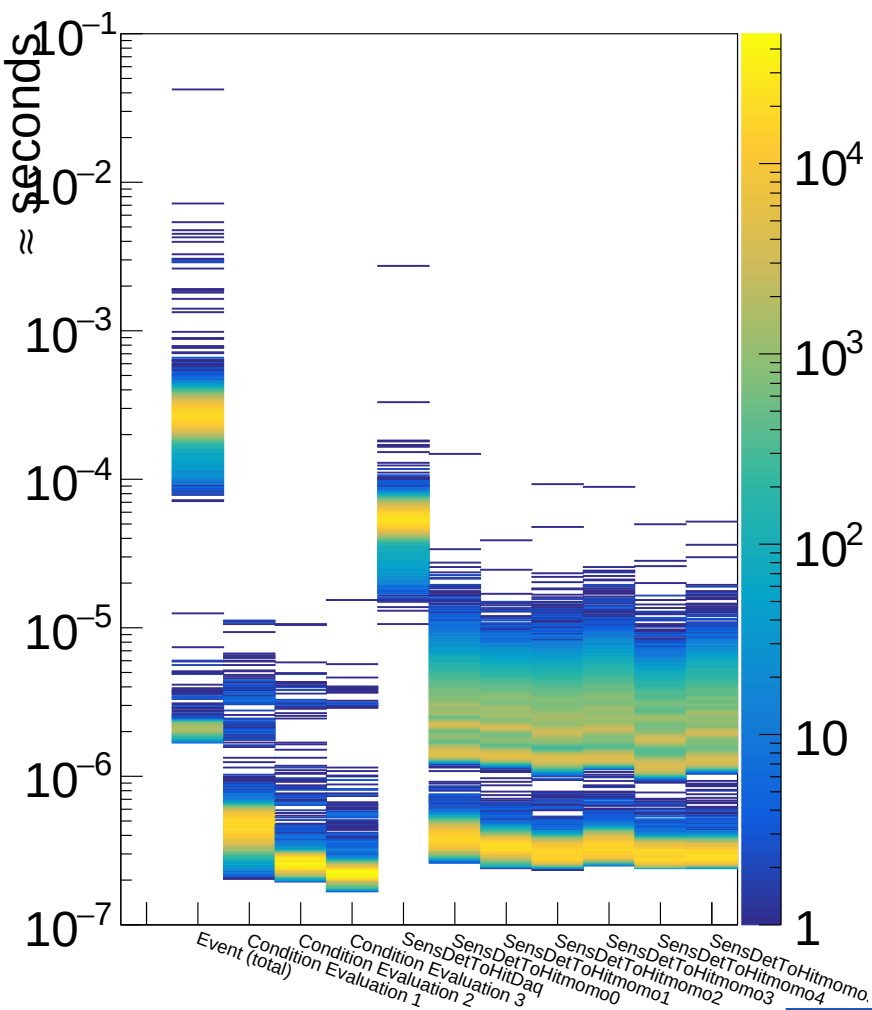
universitätbonn

Performance comparison (ROOT 6)

TMethodCall



functionDictionary

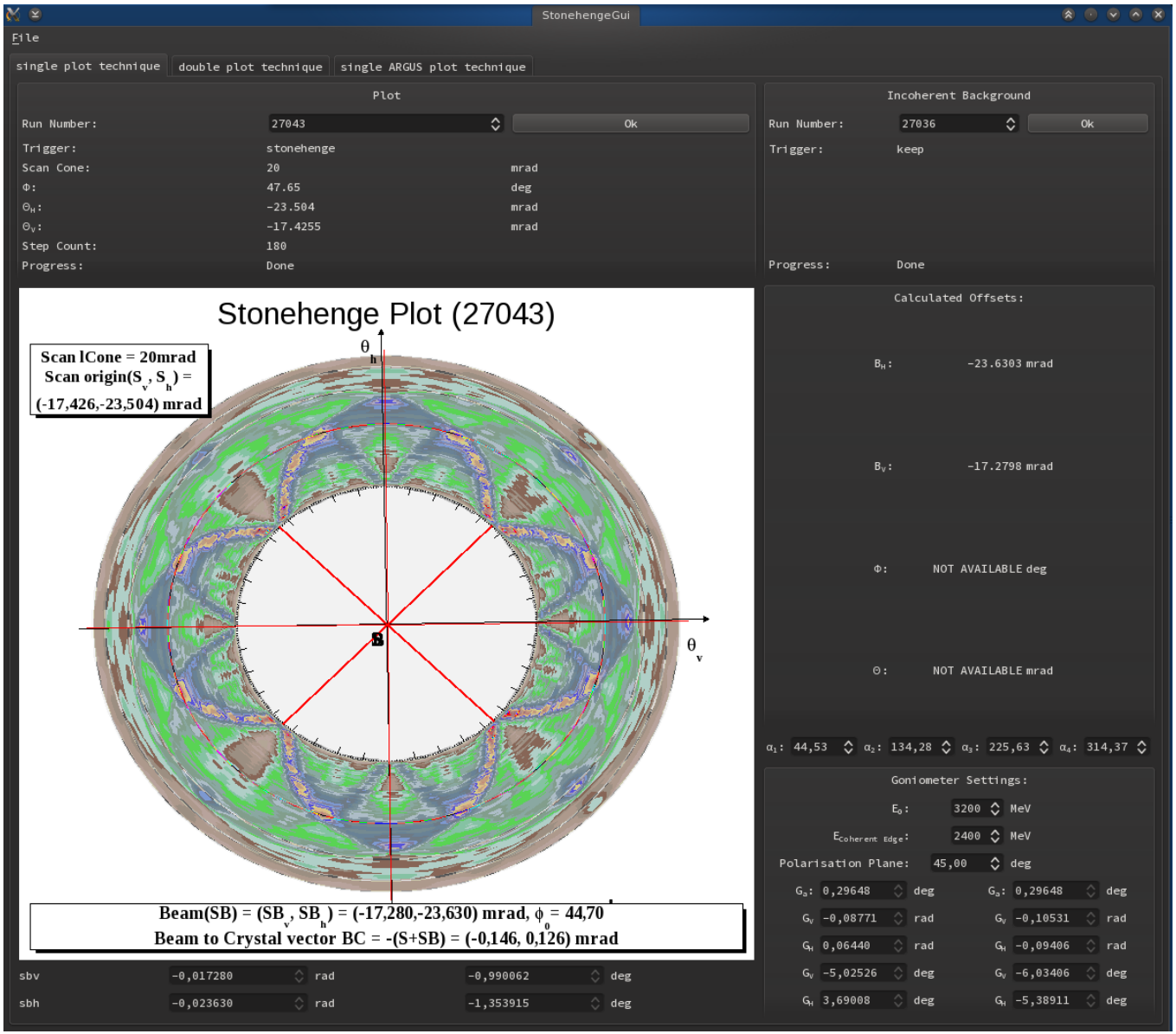
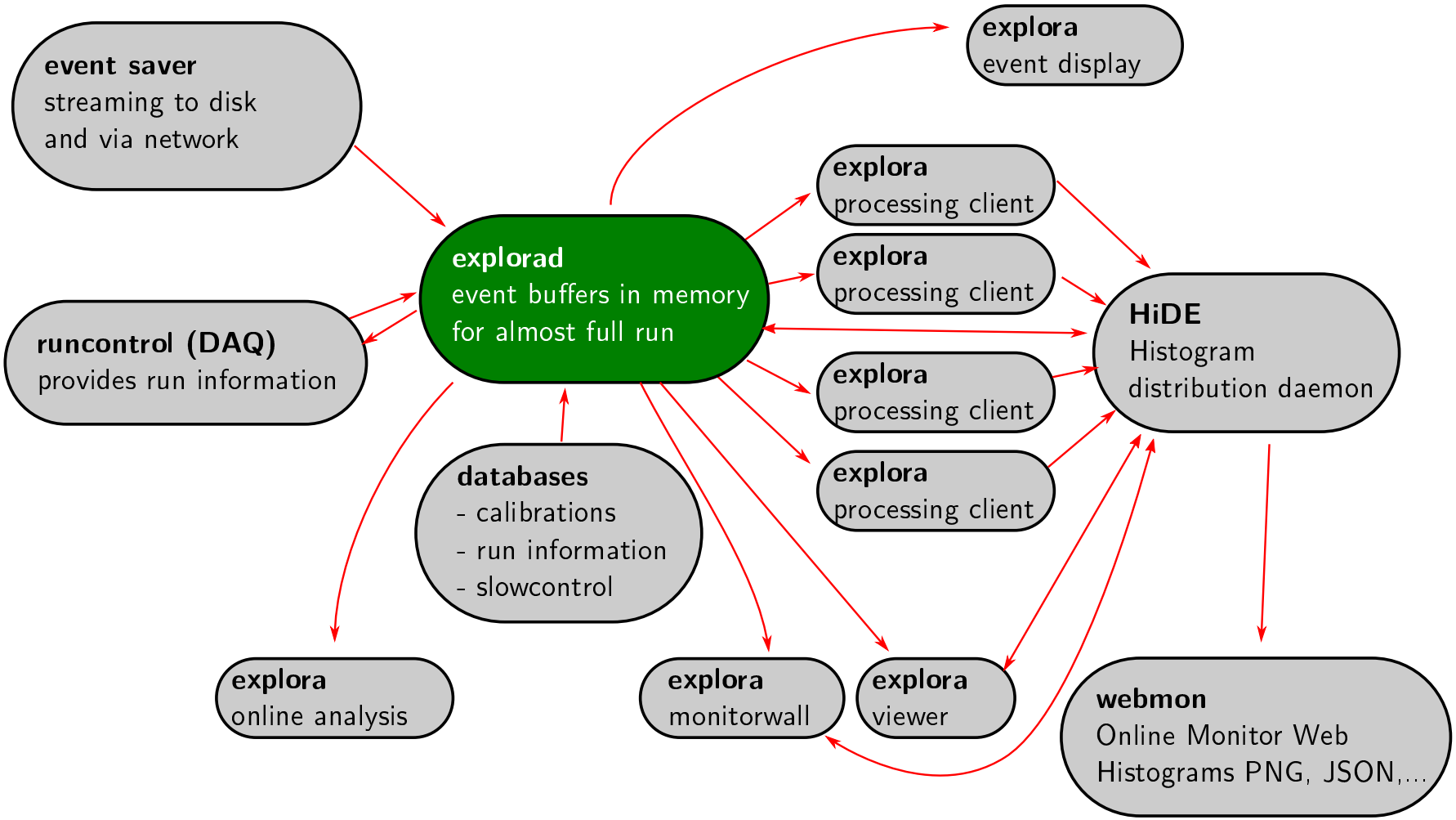


universitätbonn

Generalization of common functionality

```
<BTHFlexiHisto
  name="TOF12_cluster_pos_diff"
  xmin="-110" xmax="110" xbins="110"
  ymin="-110" ymax="110" ybins="110"
  x1container="TOF2_cluster" x1method="fGetPosition.X"
  xoperator="-"
  x2container="TOF1_cluster" x2method="fGetPosition.X"
  y1container="TOF2_cluster" y1method="fGetPosition.Y"
  yoperator="-"
  y2container="TOF1_cluster" y2method="fGetPosition.Y"
  histotype="TH2D"
  histofolder="TOF_TEST"
  histoname="TOF12_cluster_pos_diff"
  histotitle="TOF2 cluster pos - TOF1 cluster pos"
/>
```

Qt ROOT applications



Online Monitor