GaudiAna

A lightweight analysis framework on top of Gaudi

Martin Errenst
Attila Krasznahorkay

Gaudi Workshop
September 25, 2017

SPONSORED BY THE Federal Ministry of Education and Research

What is GaudiAna?

GaudiAna is a demonstrator for a lightweight analysis framework on top of Gaudi.

How?

- Gaudi in standalone mode
- Eventloop-like usage
- Minimal overhead for starting with simple examples

Why?

- R&D for run 3 analysis workflow
- Playground for AthenaMT development
- Keep analysis developers close to AthenaMT
- Platform independence

REPOSITORIES

GaudiAna:
`https://gitlab.cern.ch/akraszna/GaudiAnalysis/tree/master/`

GaudiAnalysisExternals:
`https://gitlab.cern.ch/akraszna/atlasexternals/tree/GaudiAnalysisTests-master-20170626`

Dependencies

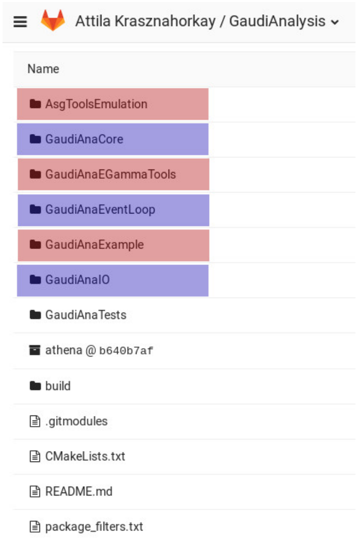
$$\begin{tikzcd}
 & \text{GaudiAna} & \\
\text{Athena} \arrow[ur] & & \text{Gaudi} \arrow[ul] \\
 & \text{GaudiAnalysisExternals} \arrow[ul] \arrow[uu] \arrow[ur] &
\end{tikzcd}$$

external dependencies:

Boost	🟥🟩🟦
Python	🟥🟩🟦
ROOT	🟥🟩🟦
TBB	🟩🟦
HEPUtils	🟥
MCUtils	🟥
Eigen	🟥
Googletest	🟦

Tested on SLC6, CentOS7 and macOS

Core Components of GAna



Attila Krasznahorkay / GaudiAnalysis

Name
- AsgToolsEmulation
- GaudiAnaCore
- GaudiAnaEGammaTools
- GaudiAnaEventLoop
- GaudiAnaExample
- GaudiAnaIO
- GaudiAnaTests
- athena @ b640b7af
- build
- .gitmodules
- CMakeLists.txt
- README.md
- package_filters.txt

- GAna Core:
  - GAna::AnalysisJob
  - GAna::EventLoopMgr
  - GAna::WhiteBoard
  - GAna::Messaging
- Necessary for example Analysis

The user only has to interact with the **`AnalysisJob`**.

Core Components

GAna::AnalysisJob

- Interaction between User & Gaudi
- Holding ApplicationMgr + default jobOptions
- Handles parameters (`--thread` or input files)

GAna::Messaging

- Providing the usual macros (`ATH_MSG_INFO() ...`)
- Common baseclass for `GAna::AlgTool`, `GAna::Algorithm` and `GAna::Service`

GAna::EventLoopMgr

Based on

- Gaudis MinimalEventLoopMgr
- AthenaHiveEventLoopMgr (simplified)

GAna::WhiteBoard

- Threadsafe & parallelized
- One `xAOD::TEvent` & `xAOD::TStore` instances per thread
- DataStoreLoaderAlg for inputfiles
- Dependency declaration via Read-/Writehandles & Keys

In more detail: WhiteBoard

```
class WhiteBoard : public extends< Service, IHiveWhiteBoard, GAna::IDataStore,
                                   GAna::IDataStoreInspector > {
    ...
};
```

- Implementing the **`IHiveWhiteBoard`** interface for the Gaudi scheduler
- Implementing the **`GAna::IDataStore`** interface defined by GaudiAnalysis
- Deals with data in memory (through **`xAOD::TStore`**) and file IO (through **`xAOD::TEvent`**) $\Rightarrow$ no converters involved
- Parallelized through **`THREAD_TLS`** pointers to partitions containing:
  - **`xAOD::TEvent`** & **`xAOD::TStore`**
  - current event number
  - & more

In more detail: Read-/WriteHandles

in header:

```
Gaudi::Property< GAna::ReadHandleKey > m_CEinputKey{ this,
   "CalibratedElectrons_InputKey",
   GAna::ReadHandleKey( "CalibratedElectrons", *this ),
   "Key␣to␣read␣an␣IParticle␣container␣with" };
```

in .cxx:

```
// Initialise the input key in initialize()
ATH_CHECK( m_CEinputKey.initialize() );
ATH_MSG_INFO( "Reading␣input␣with:␣" << m_CEinputKey );
...
// Retrieve the calibrated electrons in execute()
GAna::ReadHandle<xAOD::ElectronContainer> electrons( m_CEinputKey.value() );
// Loop over the calibrated electrons:
for( const xAOD::Electron* el : *electrons ) { ... }
```

Read-/WriteHandleKey's are based on `GAna::DataHandleKey`

GAna::DataHandleKey

```
class GAna::DataHandleKey : public Gaudi::DataHandle
```

- Constructor takes `IDataPropertyHolder` (same as Gaudi)
- Holds `ServiceHandle<GAna::IDataStore>` for IO with WhiteBoard
- The DataHandleKey is used in `Gaudi::Property` instead of **being** one itself
    - `owner()->declare( *this )` in constructor
    - `owner()->renounce( *this )` in destructor
    - `operatior<<()`, `operator=()` and copy constructor overloaded to handle `declare()` & `renounce()` correctly
- Parser implemented to parse string property to `GAna::DataHandleKey` object

DataHandleKey vs. VarHandleKey (Athena)

`DataHandleKey` is the GAna equivalent to the Athena **`VarHandleKey`**

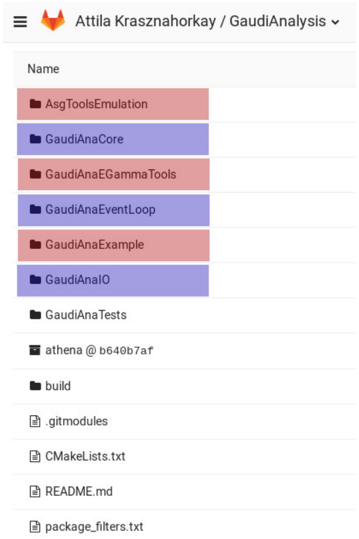
GAna

- **`DataHandleKey`** is acting as the content of a **`Gaudi::Property`**
- **`declare()`** and **`renounce()`** are called in **`DataHandleKey`**
- $\Rightarrow$ no template specializations necessary
- Read-/WriteHandle don't inherit from a common base class

Athena

- **`VarHandleKey`** is acting as a **`Gaudi::Property`** itself
- **`declare()`** and **`renounce()`** are called in **`declareProperty<T>()`**
- $\Rightarrow$ template specializations are necessary
- Read-/WriteHandle inherit from **`SG::VarHandleBase`**

Current usage example


Attila Krasznahorkay / GaudiAnalysis

Name
AsgToolsEmulation
GaudiAnaCore
GaudiAnaEGammaTools
GaudiAnaEventLoop
GaudiAnaExample
GaudiAnaIO
GaudiAnaTests
athena @ `b640b7af`
build
.gitmodules
CMakeLists.txt
README.md
package_filters.txt

- GAna Core:
  - GAna::AnalysisJob
  - GAna::EventLoopMgr
  - GAna::WhiteBoard
  - GAna::Messaging
- Necessary for example Analysis
  - AsgToolsEmulation
  - GaudiAnaEGammaTools
  - GaudiAnaExample

INGREDIENTS FOR THE EXAMPLE

GaudiAnaExample

- Standalone program
- ECalibration Algorithm
- ESelection Algorithm
- ZCandidate selection Algorithm

AsgToolsEmulation

- wrapping Asg namespace with GAna messaging and correct base classes

GaudiAnaEGammaTools

quick and dirty copy of

- ECalibration Tool
- ESelection Tool
- GoodRunsSelectionTool

Necessary for compatibiltiy
(not using Athena base classes)

THE ANALYSIS EXECUTABLE – GANA_ZANA.CXX

```
// Gaudi analysis include(s):
#include "GaudiAnaCore/AnalysisJob.h"

int main( int argc, char *argv[] ) {
   // Instantiate the analysis job
   GAna::AnalysisJob job;
   CHECK( job.configure( argc, argv ) );
   CHECK( job.setProperty( "EvtMax", "1000" ) );
   CHECK( job.setProperty( "MessageSvc", "OutputLevel",
                           std::to_string(MSG::INFO) ) );
   CHECK( job.setProperty( "EventLoopMgr", "EventPrintoutInterval", "100" ) );
   // set private tool properties
   CHECK( job.setProperties( "ECalibAlg.ElectronCalibTool",
                             { {"ESModel", "es2016data_mc15c"},
                               {"randomRunNumber", "297730"  } } ) );
   CHECK( job.setProperties( "ESelectionAlg.ElectronSelectionTool",
                             { {"WorkingPoint", "MediumLHElectron"} } ) );
```

```
    const std::string grls = "[␣\"GoodRunsLists/data16_13TeV/20161101/"
          "data16_13TeV.periodAllYear_DetStatus-v83-pro20-15_DQDefects"
          "-00-02-04_PHYS_StandardGRL_All_Good_25ns.xml\"␣]";
    CHECK( job.setProperties( "ZAnAlg.GoodRunsSelectionTool",
                              { {"PassThrough", "false"},
                                {"GoodRunsListVec", grls } } ) );

    // Add the algorithm(s) to the job:
    CHECK( job.addAlgorithm( "ZAna::ElectronCalibAlg", "ECalibAlg" ) );
    CHECK( job.addAlgorithm( "ZAna::ElectronSelectionAlg", "ESelectionAlg" ) );
    CHECK( job.addAlgorithm( "ZAna::ZAnalysisAlg", "ZAnAlg" ) );

    // And now run it:
    CHECK( job.run() );

    // Return gracefully:
    return 0;
}
```

Future plans & Next steps

- (multithreaded) Histogram Service
  1. Multithreaded THistSvc by Charles Leggett
     - Example works with `--thread 1`
     - Uncovered issue with Service initialization in GAna
  2. Own implementation with `AthHistogramming`-Interface
     - Not fully functional at the moment
     - `tbb::enumerable_thread_specific` with merge in `GAna::HistSvc::finalize()`
- Algorithm to write xAODs directly
- General improvements
  - Better testcoverage
  - More convenience implementations

BACKUP

Building is easy

Similar to building Athena, helper scripts in ./build

1. `./GAna/build/build_GaudiAnalysisExternals.sh`
2. `./GAna/build/build_Gaudi.sh`
3. `cd GAnabuilddir`
4. `source /path/to/GAna/build/build_env.sh`
5. `cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo -DCMAKE_INSTALL_PREFIX=/InstallArea/x86_64-centos7-gcc62-opt /path/to/GAna/`
6. `make`
7. `DESTDIR=/path/to/build/install/GaudiAnalysis/22.0.0 make install/fast`
8. `asetup GaudiAnalysis,22.0.0 --releasesarea=/path/to/build/install/`

IN MORE DETAIL: EVENTLOOPMGR

- Inherits form Gaudi::MinimalEventLoopMgr
- simplified version of AthenaHiveEventLoopMgr
  - just to handle nextEvent
  - no incidents
  - no metadata
- Uses GAna::WhiteBoard
- Uses the AvalancheSchedulerSvc