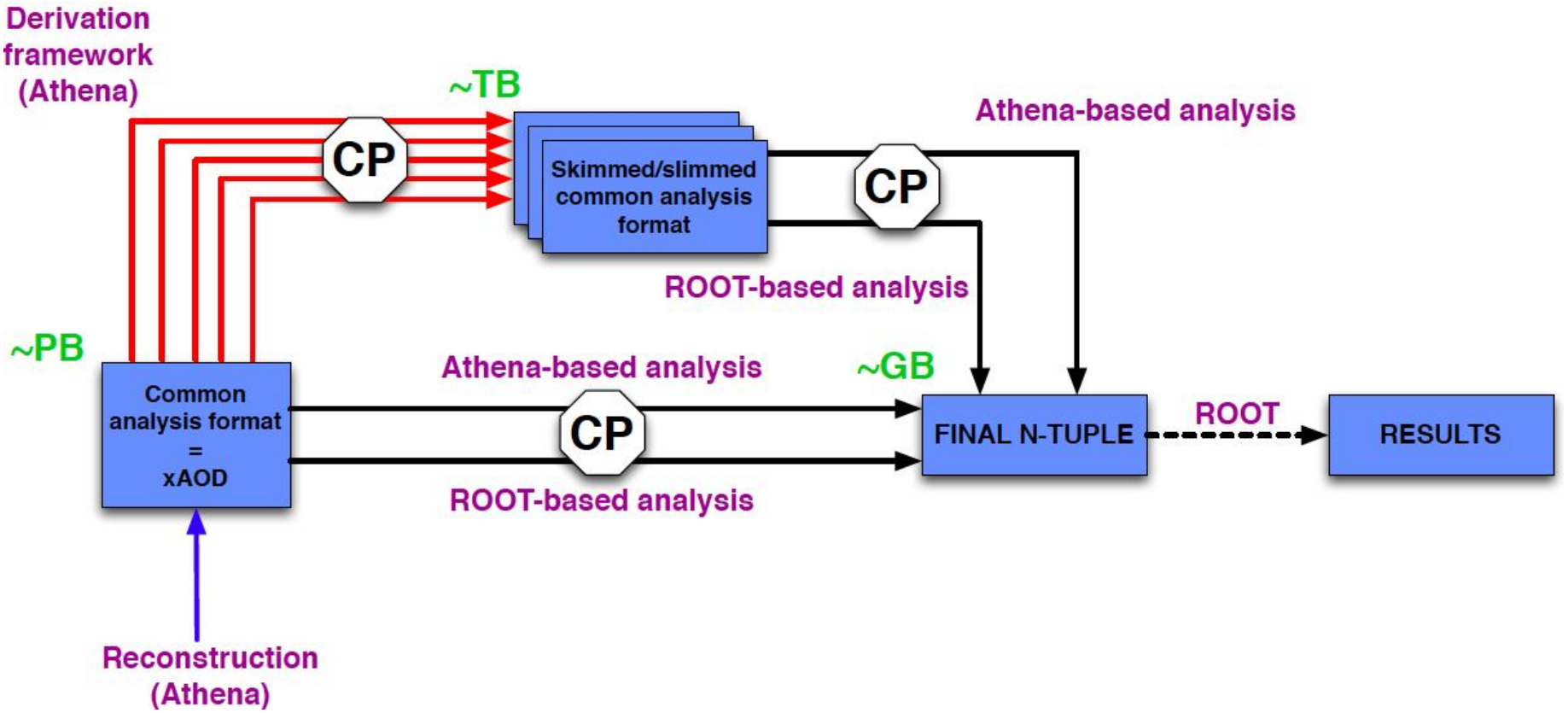
The RunII Analysis Model



Ulla Blumenschein, Göttingen university

ATLAS data analysis

1) Root analysis

compilation for example with RootCore
using e.g. MakeClass method or EventLoop package to create loop over events:
lighter, faster, less powerful

→ 75% of end user analyses in ATLAS

In RunI using ntuple format → RunII: xAOD format

2) Athena analysis

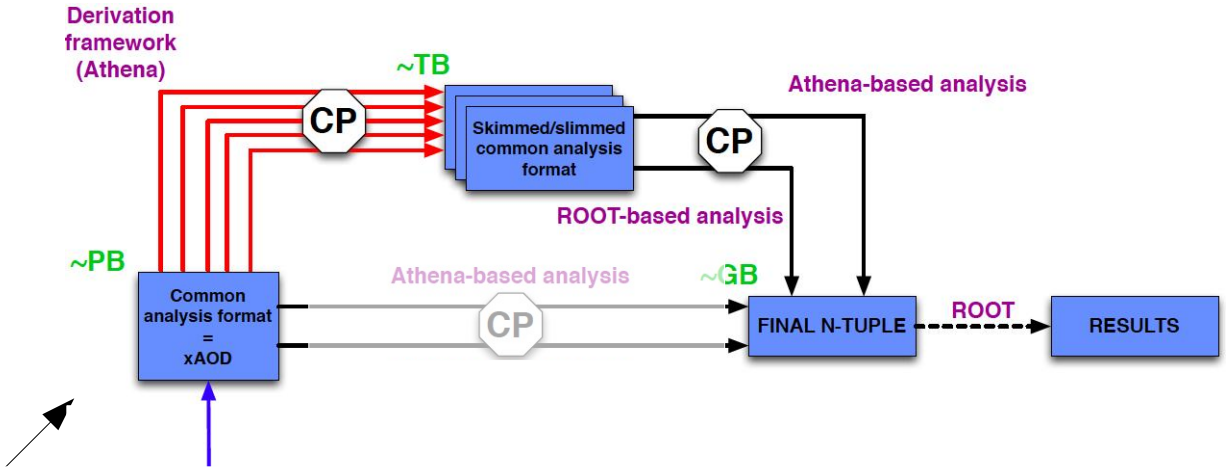
python (+ some C++ wrappers), compiled with cmt, using the Athena Framework to loop over events: heavy but powerful

→ detector simulation, reconstruction, 25% of end user analysis

In RunI producing or using AOD format → RunII: xAOD format

→ Common data format and analysis tool packages for both types of analysis frameworks

RunII analysis model: more details



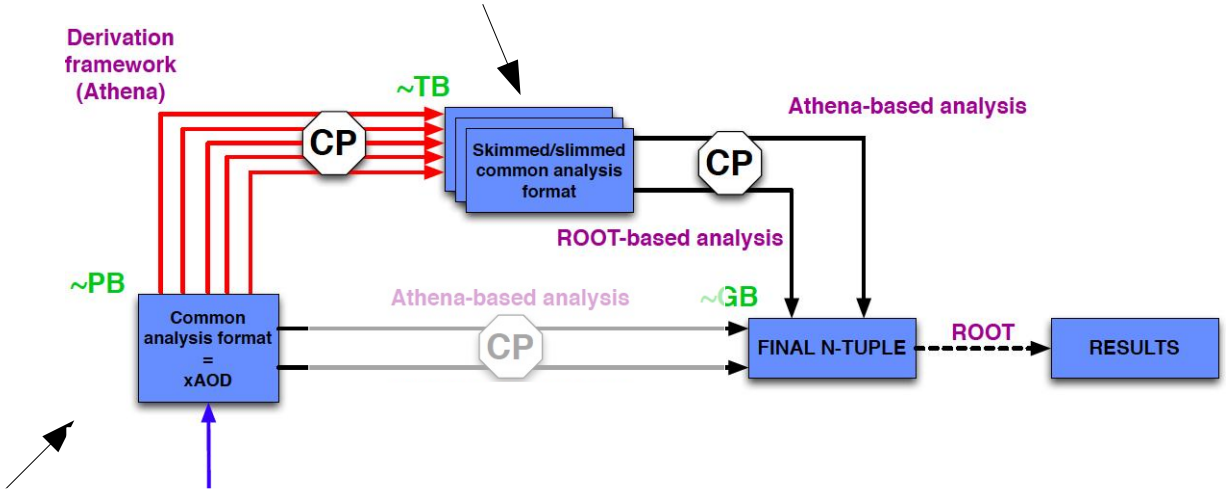
♦ **Common data format: xAOD**

- Main part readable in athena and ROOT
- Compact
- Uniform across objects
- user ↔ interfaces ↔ auxiliary storage

RunII analysis model: more details

♦ **Derivations:**

- Designed/managed by CP / PA groups
- Centrally produced (upon group requests)
- Size ~1% of original xAOD
- Skimmed/slimmed/thinned xAOD
- Augmented by user data
- Apply xAOD fixes between reprocessings



♦ Common data format: xAOD

- Main part readable in athena and ROOT
- Compact
- Uniform across objects
- user ↔ interfaces ↔ auxiliary storage

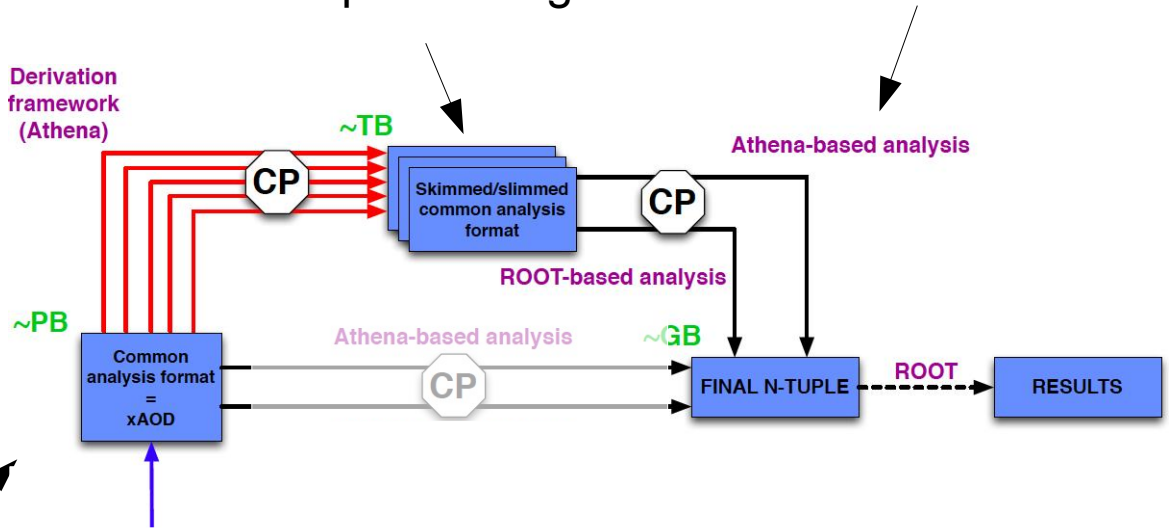
RunII analysis model: more details

♦ **Derivations**:

- Designed/managed by CP / PA groups
- Centrally produced (upon group requests)
- Size ~1% of original xAOD
- Skimmed/slimmed/thinned xAOD
- Augmented by user data
- Apply xAOD fixes between reprocessings

♦ **Subgroup specific analysis frameworks**

- Athena or Root (or PyRoot) based
- Using versioned Analysis Release
- Dual-use Combined Performance tools:
  Same tools as in derivation framework
  Common interface, operate on objects



♦ **Common data format: xAOD**

- Main part readable in athena and ROOT
- Compact
- Uniform across objects
- user ↔ interfaces ↔ auxiliary storage

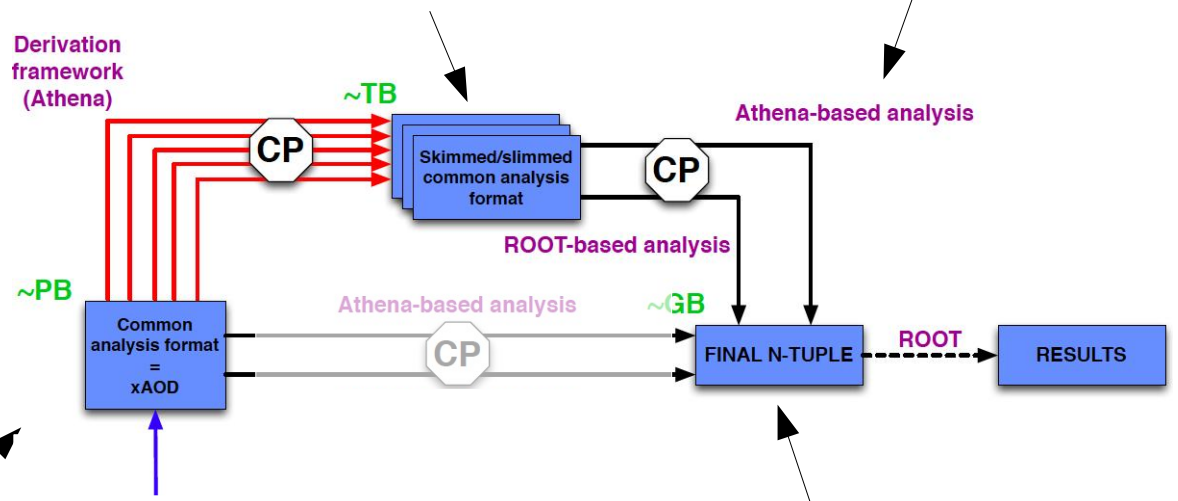
RunII analysis model: more details

♦ **Derivations:**

- Designed/managed by CP / PA groups
- Centrally produced (upon group requests)
- Size ~1% of original xAOD
- Skimmed/slimmed/thinned xAOD
- Augmented by user data
- Apply xAOD fixes between reprocessings

♦ **Subgroup specific analysis frameworks**

- Athena or Root (or PyRoot) based
- Using versioned Analysis Release
- Dual-use Combined Performance tools:
  Same tools as in derivation framework
  Common interface, operate on objects



♦ **Common data format: xAOD**

- Main part readable in athena and ROOT
- Compact
- Uniform across objects
- user ↔ interfaces ↔ auxiliary storage

♦ **Final Ntuple:**

- Tiny
- Mini xAOD (→ CP tools)
  or flat mini ntuple

The xAOD data format

EventData Model (EDM)

- Collection of classes, interfaces and concrete objects
- Provide a representation of an event detected by ATLAS and ease its manipulation

→ *How Electrons, Muons, Jets etc are stored and how to use them*

Documentation of object classes (automated doxygen documentation)

https://twiki.cern.ch/twiki/bin/view/AtlasProtected/PhysicsAnalysisWorkBookRel19

Two generations

- DC14: currently in use, used for preparation of the RunII analysis,
  Does not yet contain the final layout of the RunII detector

  Samples: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/DC14DataMCSampleInfo

- MC15 / data15: final RunII format, currently in production

  Validation samples: https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerValidation#Validation_samples

The xAOD::EventInfo

Documentation:

http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDocDirectory/xAODEventInfo/html/index.html

→ "Classes“ → xAOD::EventInfo_v1

→ Run number, event number, Luminosity block number,

→ One object per event

Basic event information

uint32_t	**runNumber** () const The current event's run number.
void	**setRunNumber** (uint32_t value) Set the current event's run number.
unsigned long long	**eventNumber** () const The current event's event number.
void	**setEventNumber** (unsigned long long value) Set the current event's event number.
uint32_t	**lumiBlock** () const The current event's luminosity block number.
void	**setLumiBlock** (uint32_t value) Set the current event's luminosity block number.
uint32_t	**timeStamp** () const POSIX time in seconds from 1970. January 1st.
void	**setTimeStamp** (uint32_t value) Set the POSIX time of the event.
uint32_t	**timeStampNSOffset** () const Nanosecond time offset wrt. the time stamp.
void	**setTimeStampNSOffset** (uint32_t value) Set the nanosecond offset wrt. the time stamp.
uint32_t	**bcid** () const The bunch crossing ID of the event.

The xAOD::IParticle

Documentation:

http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDocDirectory/xAODBase/html/index.html

→ "Classes“ → "IParticle"

Base interface class for all objects (electrons, muons, jets, tracks, ...)

Functions describing the 4-momentum of the object

typedef TLorentzVector	**FourMom_t** Definition of the 4-momentum type.
virtual double	**pt** () const =0 The transverse momentum (p_T) of the particle.
virtual double	**eta** () const =0 The pseudorapidity (η) of the particle.
virtual double	**phi** () const =0 The azimuthal angle (ϕ) of the particle.
virtual double	**m** () const =0 The invariant mass of the particle.
virtual double	**e** () const =0 The total energy of the particle.
virtual double	**rapidity** () const =0 The true rapidity (y) of the particle.
virtual const **FourMom_t** &	**p4** () const =0 The full 4-momentum of the particle.

Particles are stored in Containers, e.g.
- xAOD::MuonContainer
- xAOD::JetContainer

The actual information is stored in an auxiliary container

The user sees only the interfaces

An object can have Element links to objects in other containers, e.g. a muon has an element link to its track

The xAOD::IParticle

Documentation:

http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDocDirectory/xAODBase/html/index.html

→ "Classes“ → "IParticle"

Base interface class for all objects (electrons, muons, jets, tracks, ...)

Functions for getting and setting user properties

template<class T >

T & **auxdata** (const std::string &name, const std::string &clsname="")
Fetch an aux data variable, as a non-const reference.

template<class T >

const T & **auxdata** (const std::string &name, const std::string &clsname="") const
Fetch an aux data variable, as a const reference.

template<class T >

bool **isAvailable** (const std::string &name, const std::string &clsname="") const
Check if a user property is available for reading or not.

template<class T >

bool **isAvailableWritable** (const std::string &name, const std::string &clsname="") const
Check if a user property is available for writing or not.

Analysis releases

Content:

- Tools which calibrate, perform efficiency corrections in simulation, calculate systematic uncertainties, etc ("Combined Performance“)
- Packages which contain the object classes (electrons, muons, jets etc.)
- Other useful packages e.g. the EventLoop package (loop over events)

Precompiled binaries available via cvmfs → also availble on Tier3, Grid sites
Source files can also be downloaded and built locally (eg. on Ubuntu, MacOS)

Two branches: 2.1.X for intermediate DC14 data format, 2.3.X for final RunII format

Provide releases for Root analysis and for Athena analysis, I will focus on Root analysis

Simple setup from cvmfs

- setupATLAS → general setup of ATLAS software
- rcSetup, Base,2.1.28 → produces a RootCore directory with the links to the header files and the precompiled binaries on cvmfs
- rc clean, rc find_packages, rc compile → to compile local analyis package

Release Documentation: https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/AnalysisBase
Tool documentation: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/PhysicsAnalysisWorkBookRel19
Support: hn-atlas-PATHelp@cern.ch

Today's program

xAOD EDM exercises:

https://twiki.cern.ch/twiki/bin/view/AtlasComputing/SoftwareTutorialxAODEDM

→ Browsing the xAOD with the TBrowser

→ PyROOT with the xAOD

XAOD analysis in ROOT:

https://twiki.cern.ch/twiki/bin/view/AtlasComputing/SoftwareTutorialxAODAnalysisInROOT

→ Creating your analysis package and algorithm

→ Accessing xAOD quantities

→ Creating and running our steering macro

The exercises can be caried out on pcatlas or on lxplus