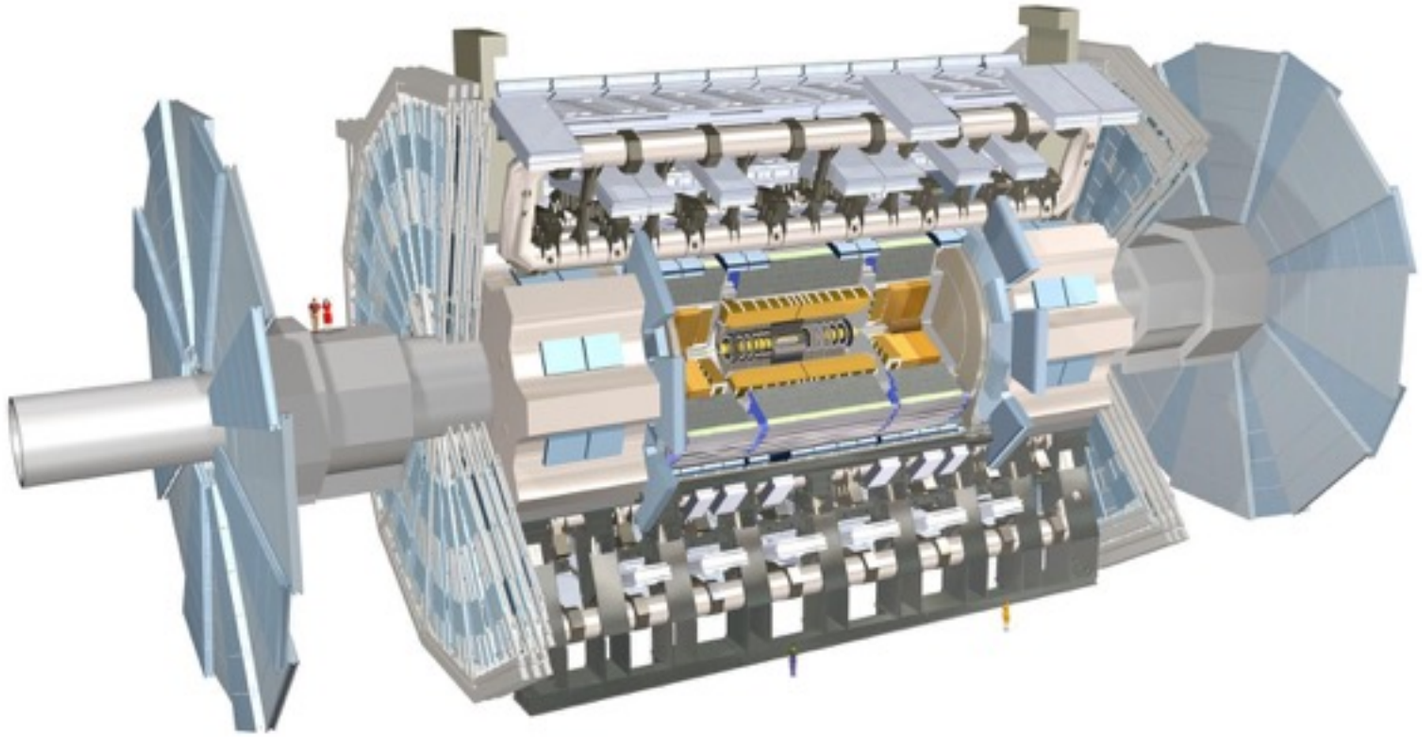
The ATLAS data processing chain: from collisions to papers

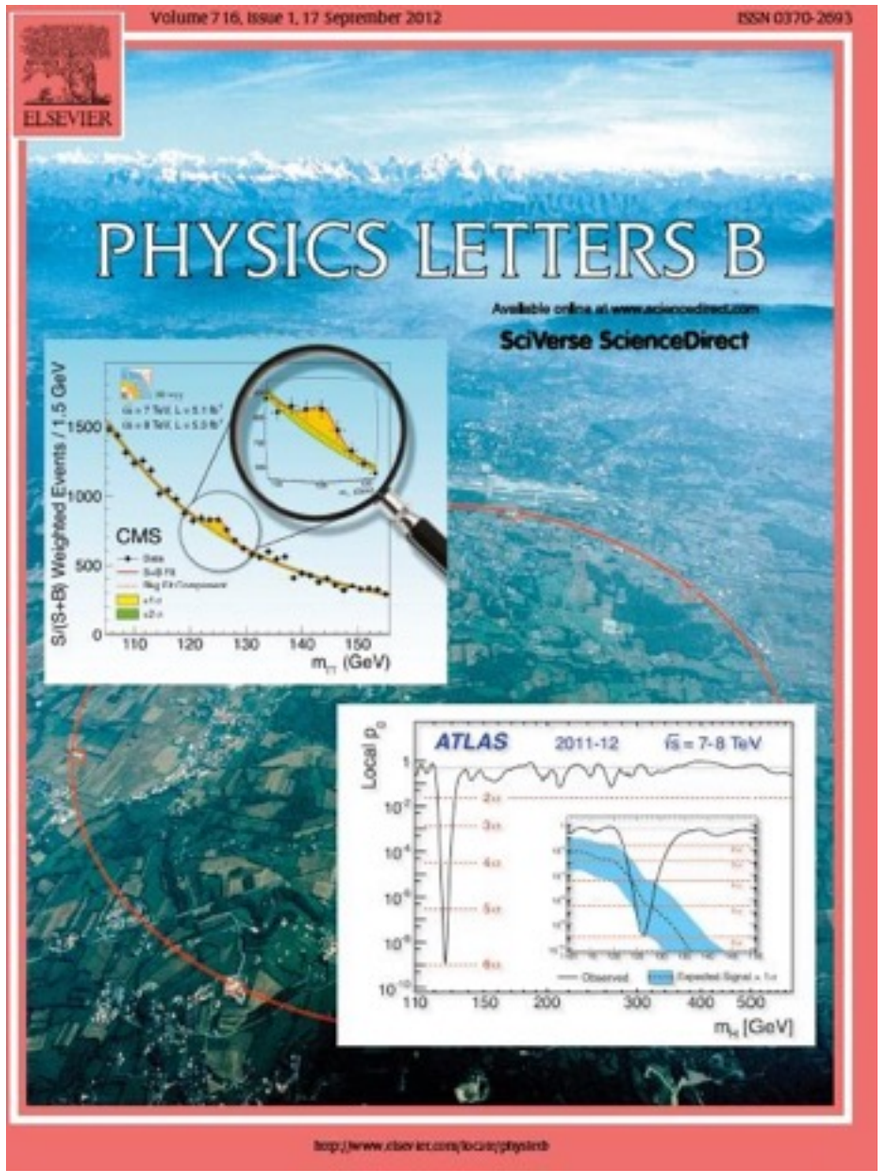
James Catmore
University of Oslo

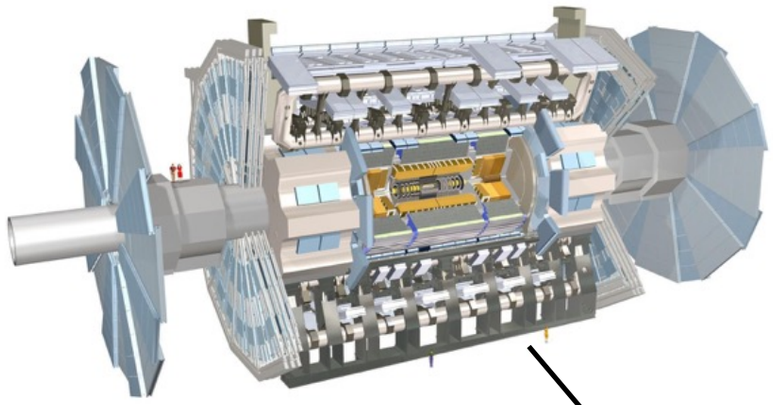
with slides purloined from:
Lukas Heinrich
Karsten Köneke
Eric Torrence

How do we get from this….



to this…?





Data

GENERATION

SIMULATION

DIGITIZATION

RECONSTRUCTION

Monte Carlo

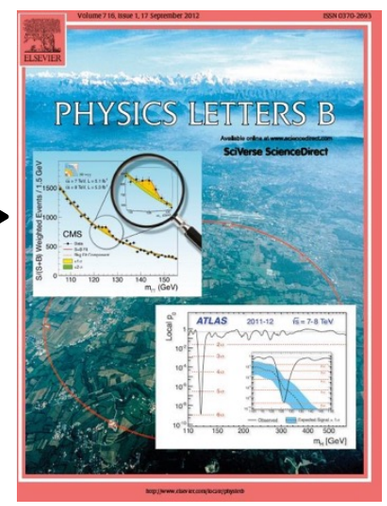
TRIGGER

RECONSTRUCTION

DERIVATION

DERIVATION

ANALYSIS

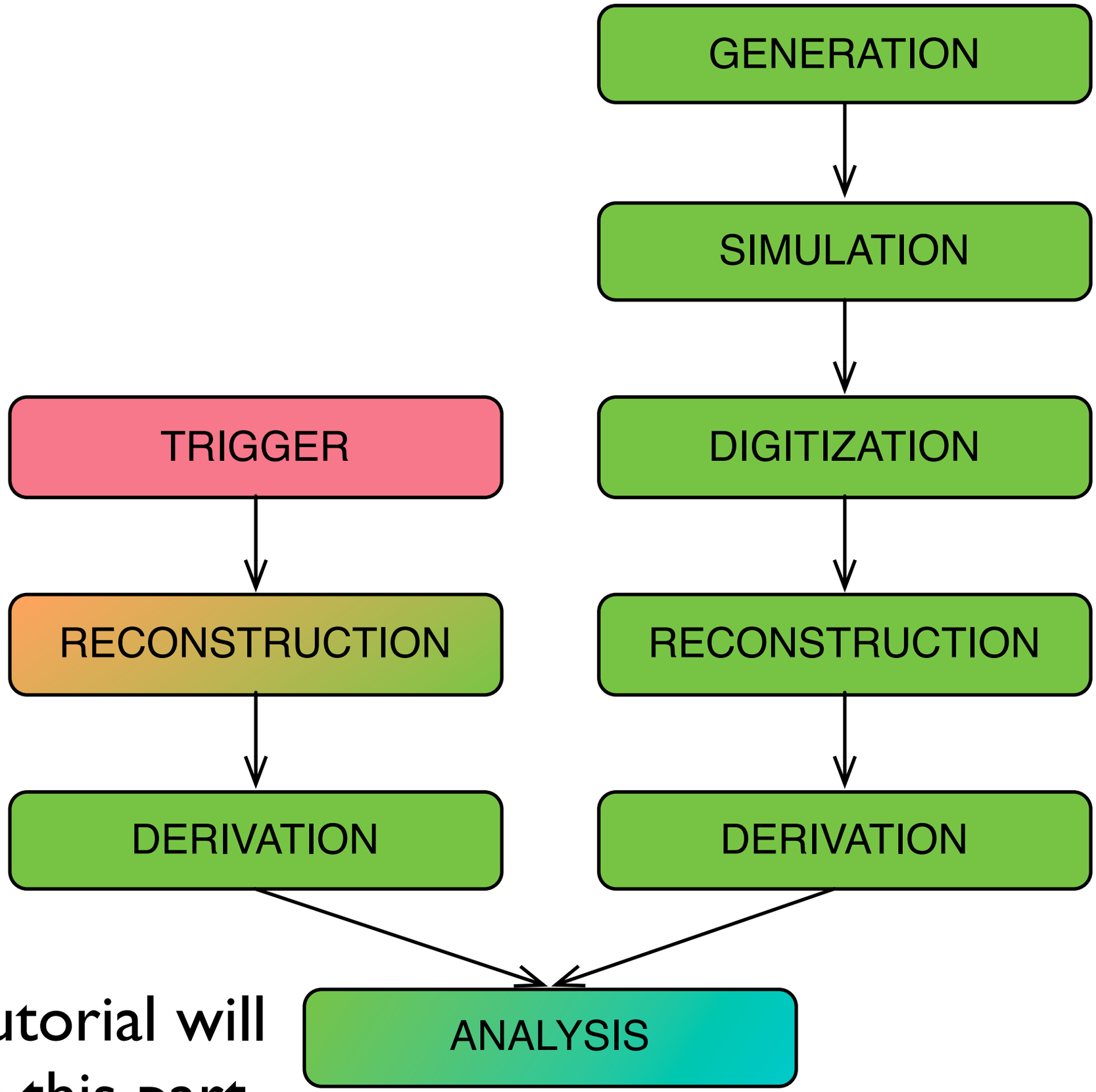


Online

Tier-0

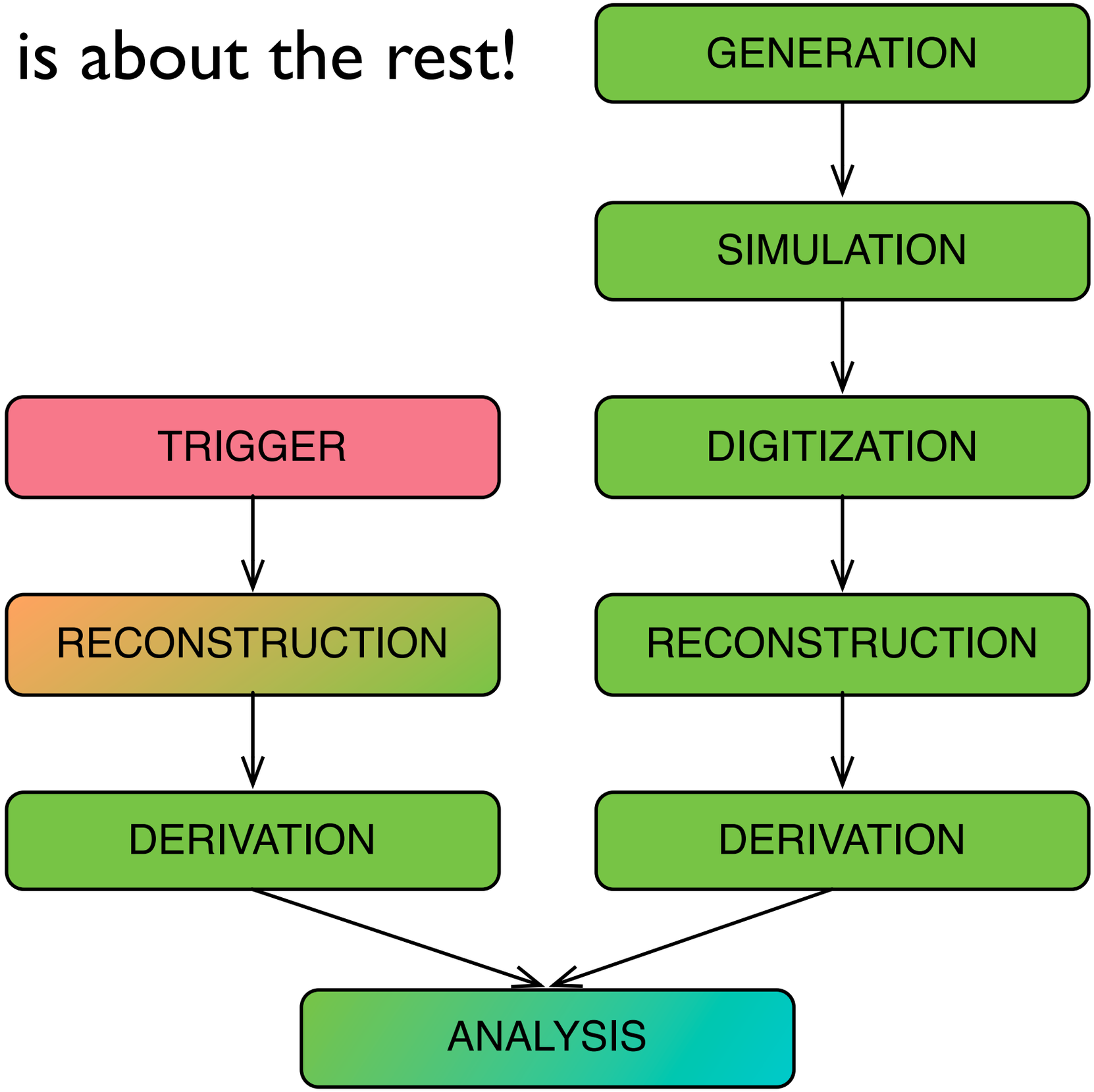
Grid

Local



The tutorial will focus on this part….

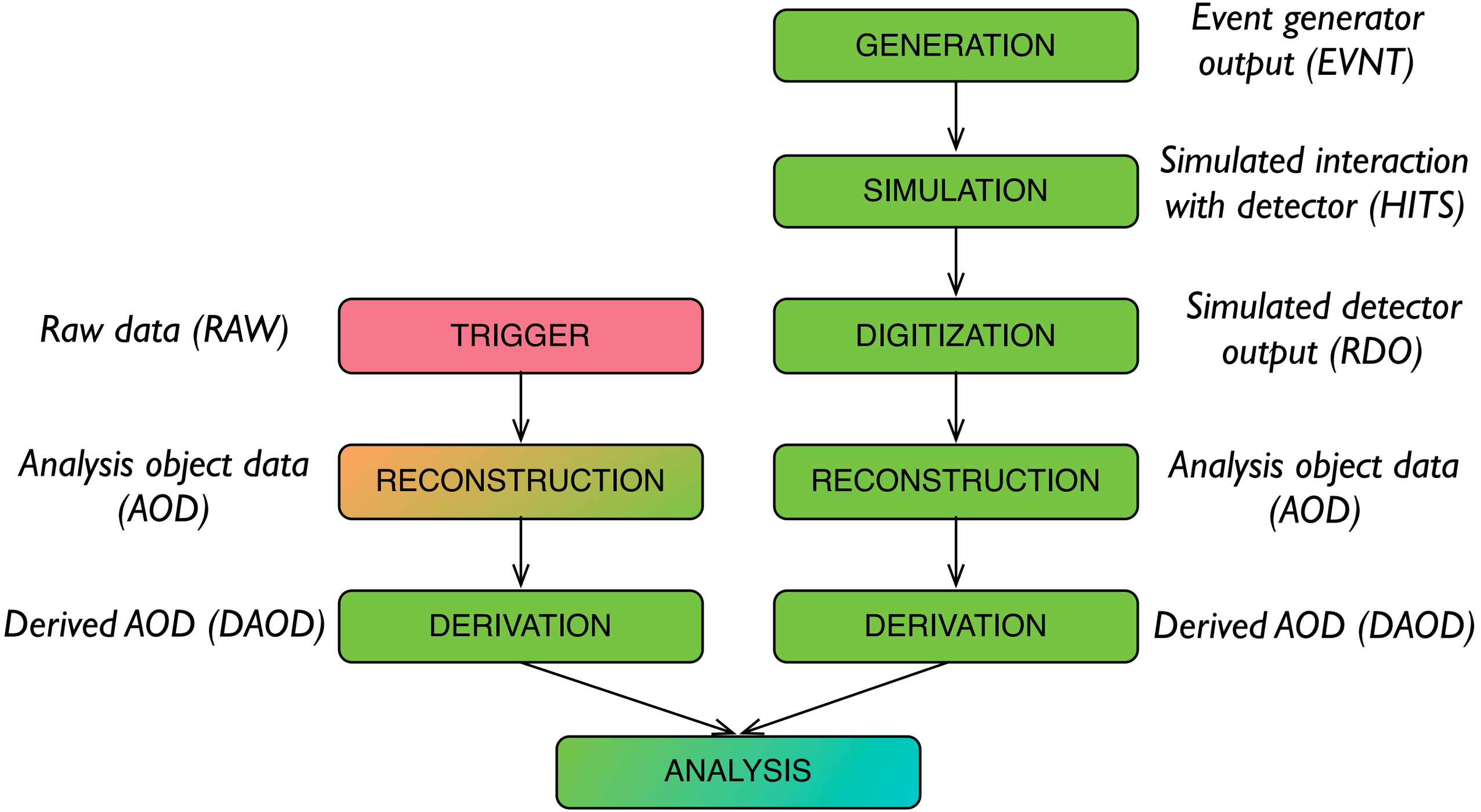
This talk is about the rest!



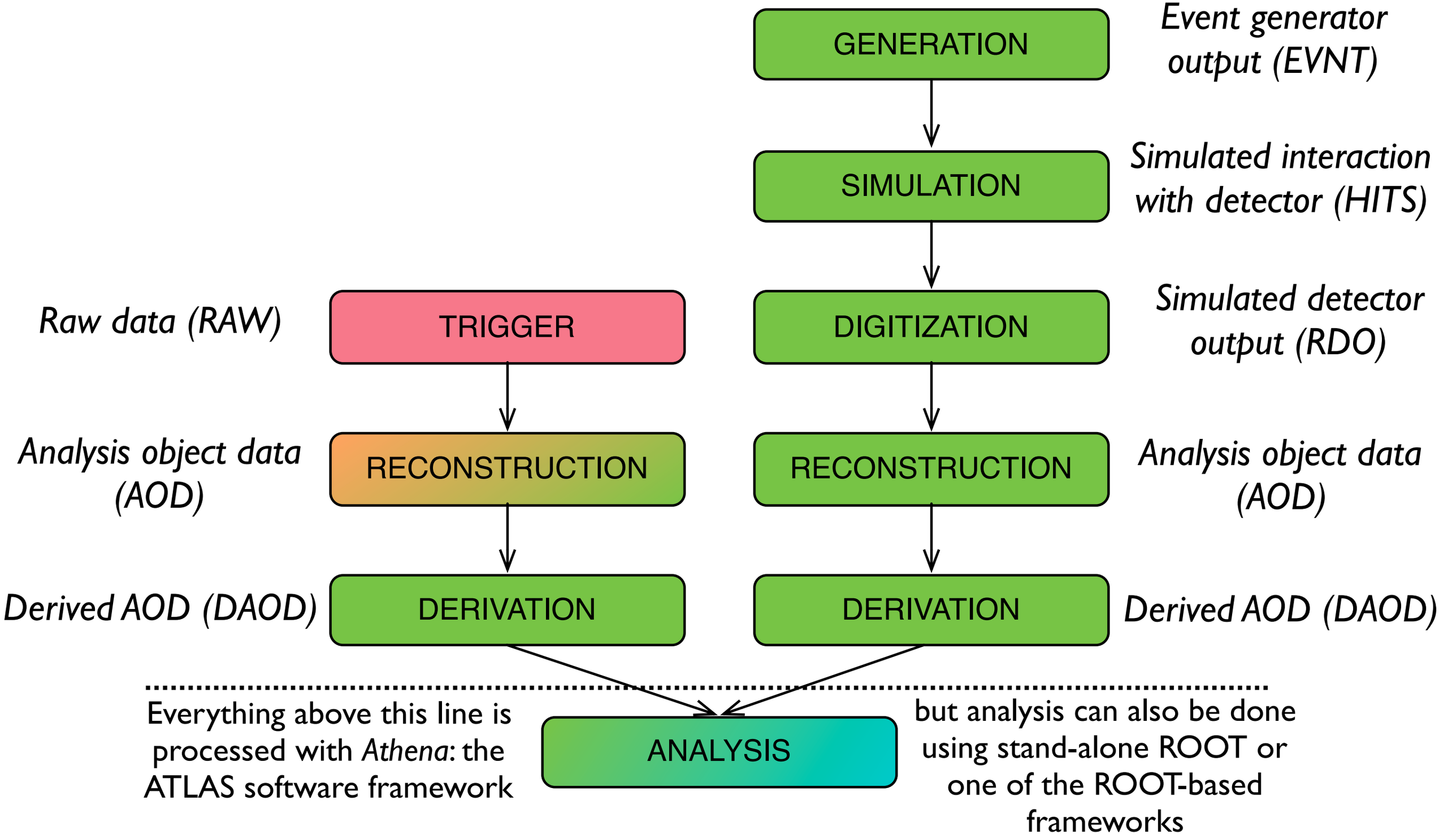
Contents

- Data formats in ATLAS
- Trigger and data preparation
  - Prompt reconstruction and reprocessing
  - Luminosity and data quality
- Monte Carlo (MC) production
- The ATLAS Analysis Model
- What's in the reconstructed data/MC?
- Tutorial overview

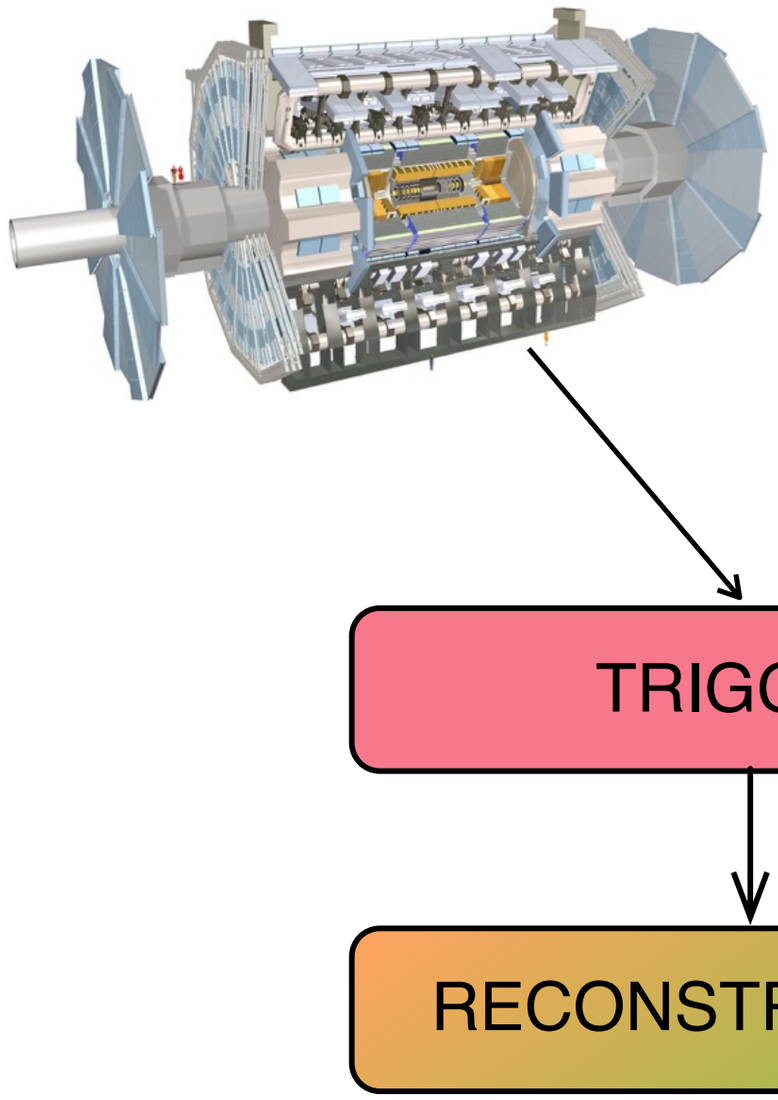
Data formats in ATLAS



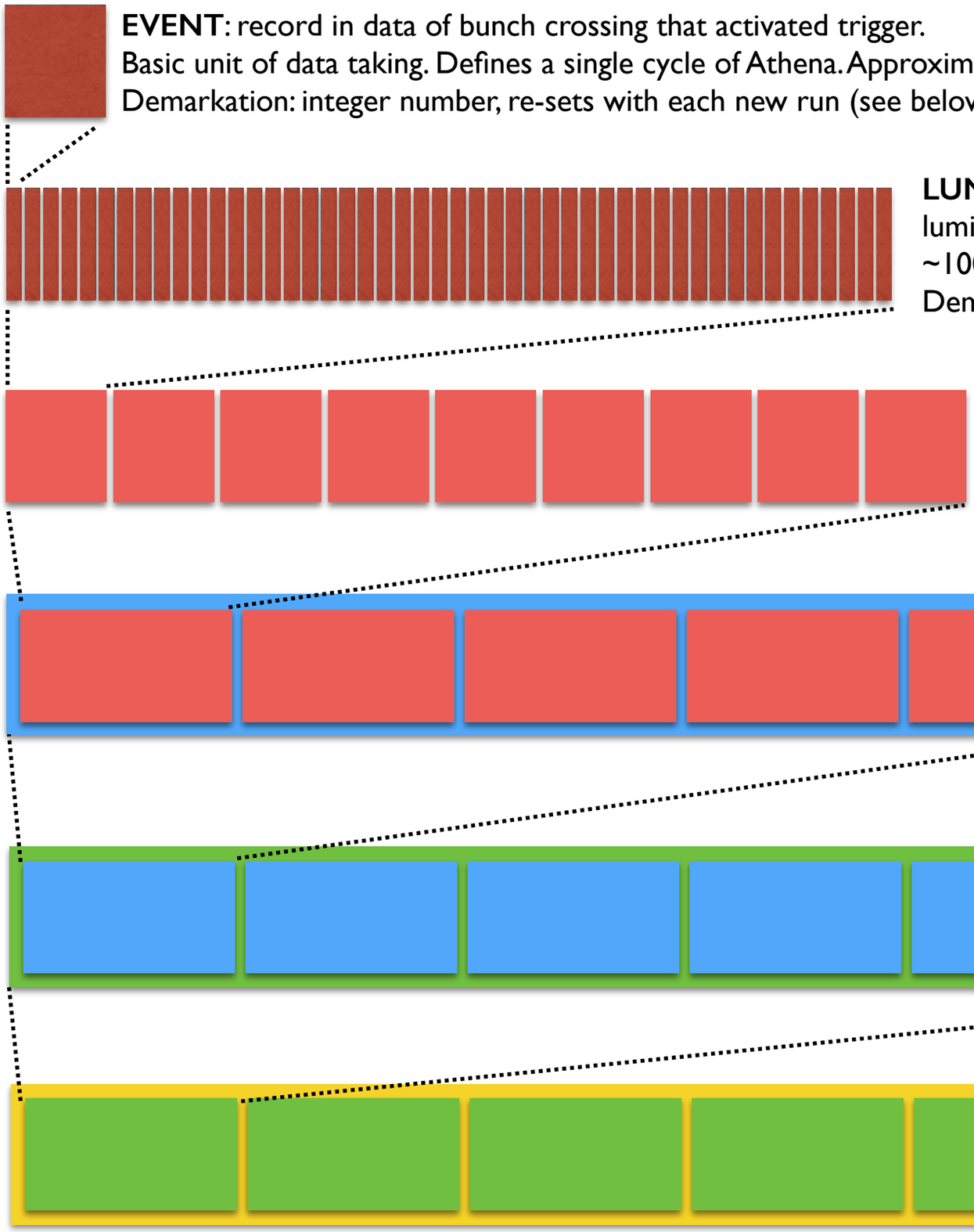
Data formats in ATLAS



Trigger and data preparation



How data is divided up in ATLAS



EVENT: record in data of bunch crossing that activated trigger.
Basic unit of data taking. Defines a single cycle of Athena. Approximately 1K per second.
Demarkation: integer number, re-sets with each new run (see below)

LUMINOSITY BLOCK: about 1 minute of data taking. Unit of known luminosity.
~100K events
Demarkation: integer number, re-sets with each new run

RUN: continuous period of ATLAS data recording. Usually corresponds to a single LHC fill, e.g. hours of data taking. ~1000 luminosity blocks.
Demarkation: integer number. Never resets → each event can be uniquely specified by its run and event number

SUB-PERIOD: group of runs taken with very similar conditions (e.g. trigger, LHC).
~5-10 runs
Demarkation: letter and number (e.g. E5)

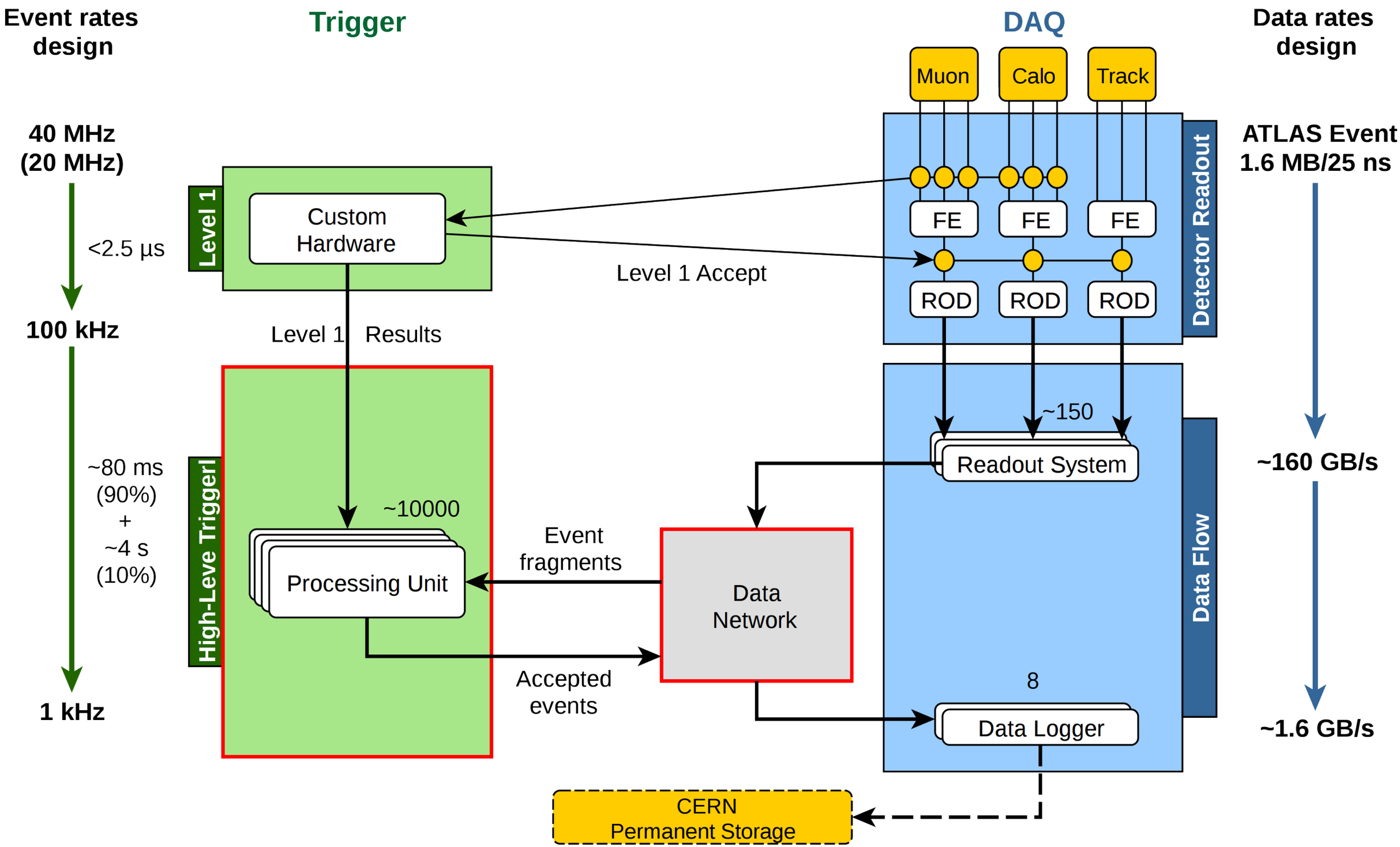
PERIOD: group of sub-periods taken with similar conditions, or during some period (e.g. increments after technical stop)
~5-10 sub periods
Demarkation: letter (e.g. E); resets each year

LHC RUN: long period of data taking, between shutdowns. 2009-2012 was Run 1, we are now in Run II, and Run III will start in 2021. Typically 20-30 periods per LHC run.

Trigger

- More details: https://indico.cern.ch/event/403126/session/10/contribution/3/attachments/1153704/1657569/SoftwareTutorialTrigger-2015-09-14.pdf
- *Trigger*: hardware and software that decides what events get written to disk
  - LHC delivers events at up to 40MHz (1 bunch crossing every 25ns)
  - We can afford to write up to 1KHz
  - This means we can keep one in 40,000 events
- The ATLAS trigger uses a two-step mechanism
  - LEVEL 1: implemented in hardware inside ATLAS - reduces rate from 40MHz to 100KHz
    - Has very little time to decide whether to keep an event or not, so can only use parts of the detector that can be quickly read out → not inner detector, only muons and calorimetry
  - HIGH LEVEL: implemented in software running on dedicated machines nearby ATLAS: reduces rate from 100KHz to 1KHz
    - Has a bit more time so can use the full detector information, including the inner detector

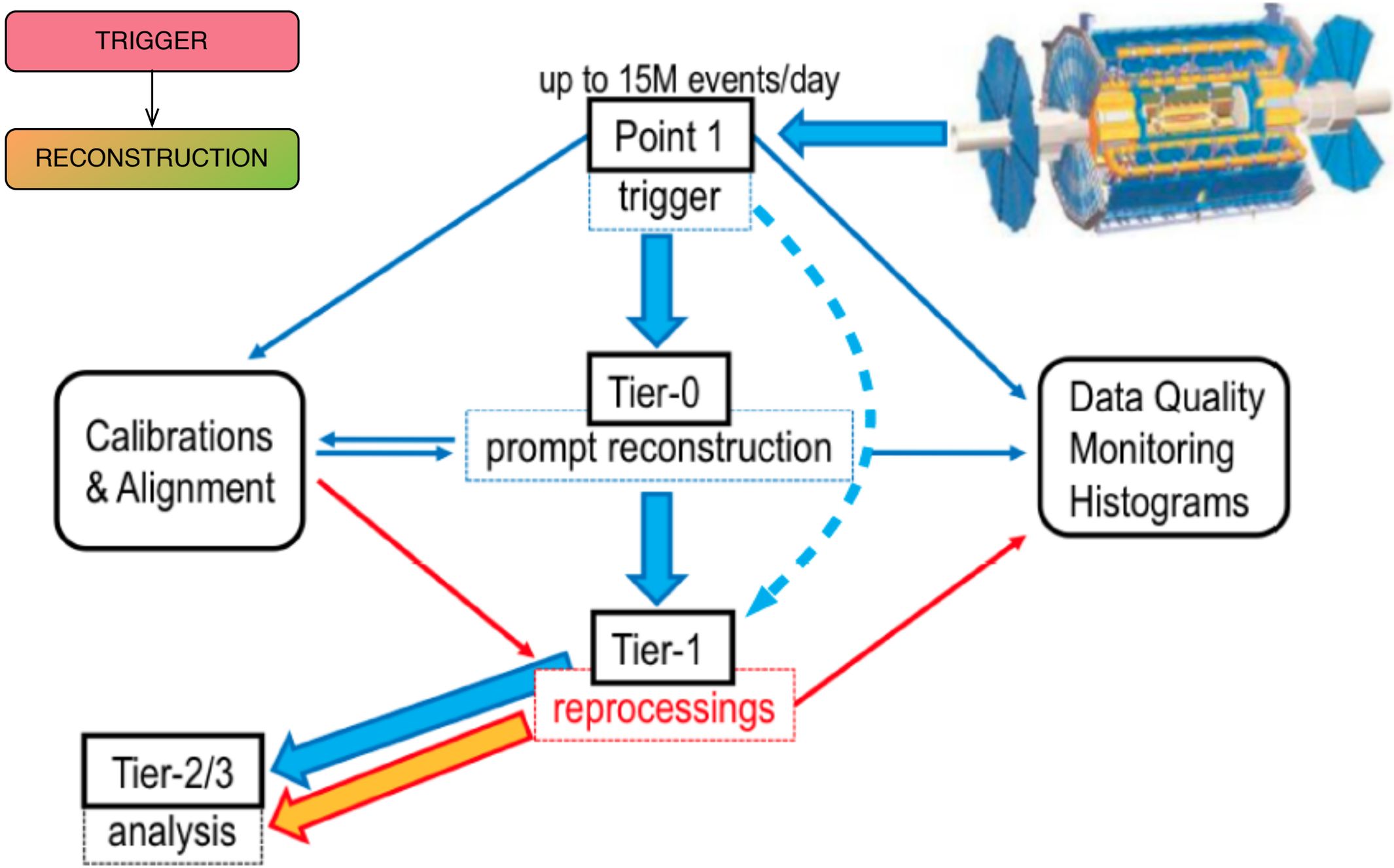
Short Recap Of The Trigger in ATLAS



Trigger in analysis

- The types of signature that the trigger accepts are encoded in the *trigger menu*
  - Example of a trigger: 2 muons with p_T greater than 40 GeV (**`HLT_2mu40`**)
  - There are several hundred such items in the trigger menu
  - Main trigger page: https://twiki.cern.ch/twiki/bin/view/Atlas/AtlasTrigger
  - Trigger menu: https://twiki.cern.ch/twiki/bin/view/Atlas/TriggerPhysicsMenu
- Some trigger items are *prescaled* (every N events passing kept, rest thrown away) to keep the rate within acceptable limits
  - Prescales must be accounted for in your analysis, especially in the luminosity calculation
  - Ideally you can find a suitable unprescaled trigger for your analysis
- The trigger is not 100% efficient for any signature, and estimating the trigger efficiency is a crucial part of any physics analysis (especially measurement)
  - Often this requires matching reconstructed objects with the equivalent trigger
- The trigger shapes the data as recorded by ATLAS, and a thorough understanding of this shaping is needed for all physics analyses
  - Often this is the hardest part of the whole analysis
  - The trigger is VERY complicated! … but there are analysis tools to help

What is Data Preparation?



- Question - What is Data Preparation?
  - One answer - “everything” between the detector and your analysis !

What is Data Preparation?

- Prompt reconstruction of the data:
  - On a Computing Farm at CERN (Tier0 Operations)
  - Using fast calibrations (Beamspot, Calibrations and Conditions)
  - And well-validated software (PROC)
- The Data Preparation group are responsible for which data:
  - How much data? (Luminosity)
  - How good is it? (Data Quality)
  - The Not-data data !!! (Non-collision backgrounds)
  - Data describing the data (Metadata)
- We also incorporate improvements in software and conditions
  - Reprocessing xAOD (Reprocessing)
  - (Derivation Framework) (in the Analysis Software Group)
- More information: https://twiki.cern.ch/twiki/bin/view/Atlas/DataPreparation

Trigger and Streams

- You'll hear more about the trigger in the next talk…

- The output of the trigger, RAW data, is organised into inclusive Streams
  - inclusive means that events can end up in one or more streams, depending on which triggers they pass
- Each trigger chain (HLT output) belongs to one stream

- In Run 1 there were three major physics streams: Egamma, Muon, JetTauEtmiss
  - In Run 2 there is one primary physics stream: physics_Main
  - Other special purpose physics streams: MinBias, physics_Late

- There are special streams critical for operations e.g. calibration, cosmics, debug…
  - calibration streams only have partial events

- …and an Express Stream: used for quick Data Quality and prompt calibration

- Can find streams produced in any given run (and many other useful things) using https://atlas-runquery.cern.ch

Express Stream & Prompt Calibrations

- Express stream contains triggers to monitor efficiencies, backgrounds and detector noise

- Reconstruction happens very quickly, even while a run is still in progress
  - Using the Tier0, identified by an AMI-tag, an *x* tag, e.g. *x353*
  - Try it !… once you have a working setup you can run the reconstruction software
  - Using the above example you would run:
    - Reco_tf.py AMI=x353 --inputBSFile=myRawData.data

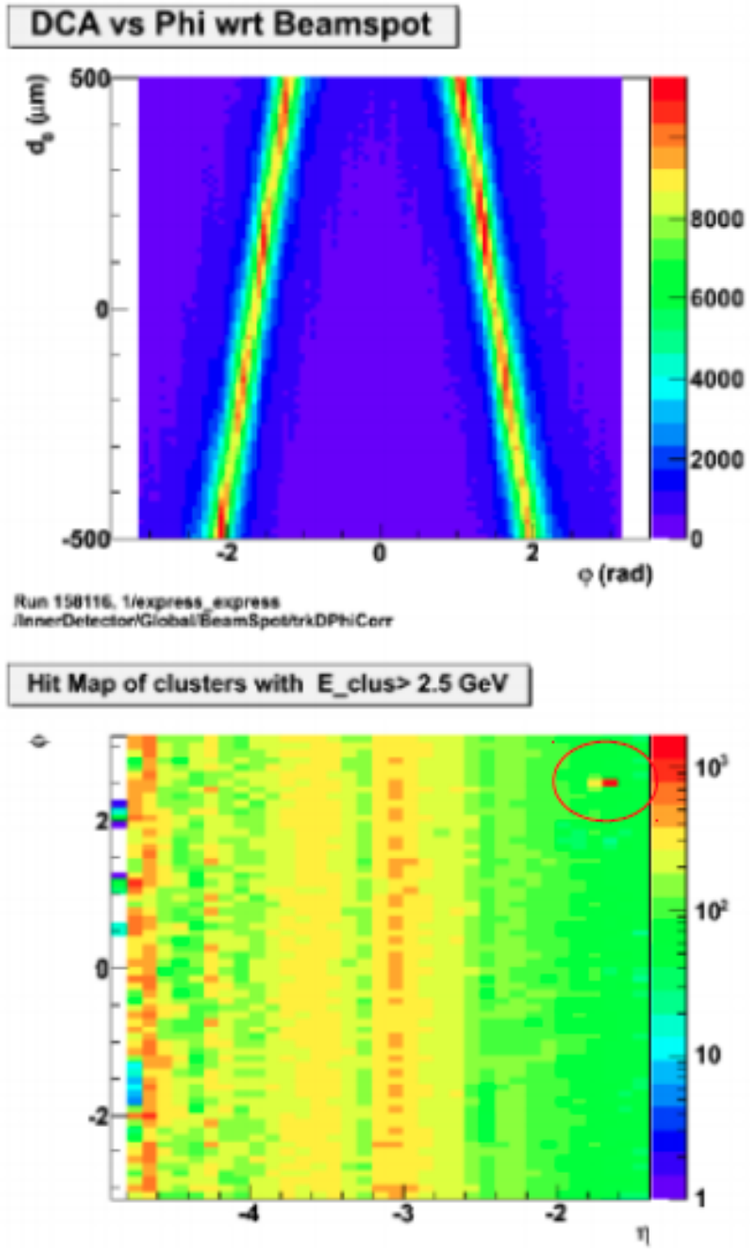
- The reconstruction job produces monitoring histograms
  - https://atlasdqm.cern.ch/webdisplay/tier0/
  - These are used for online Data Quality assessment

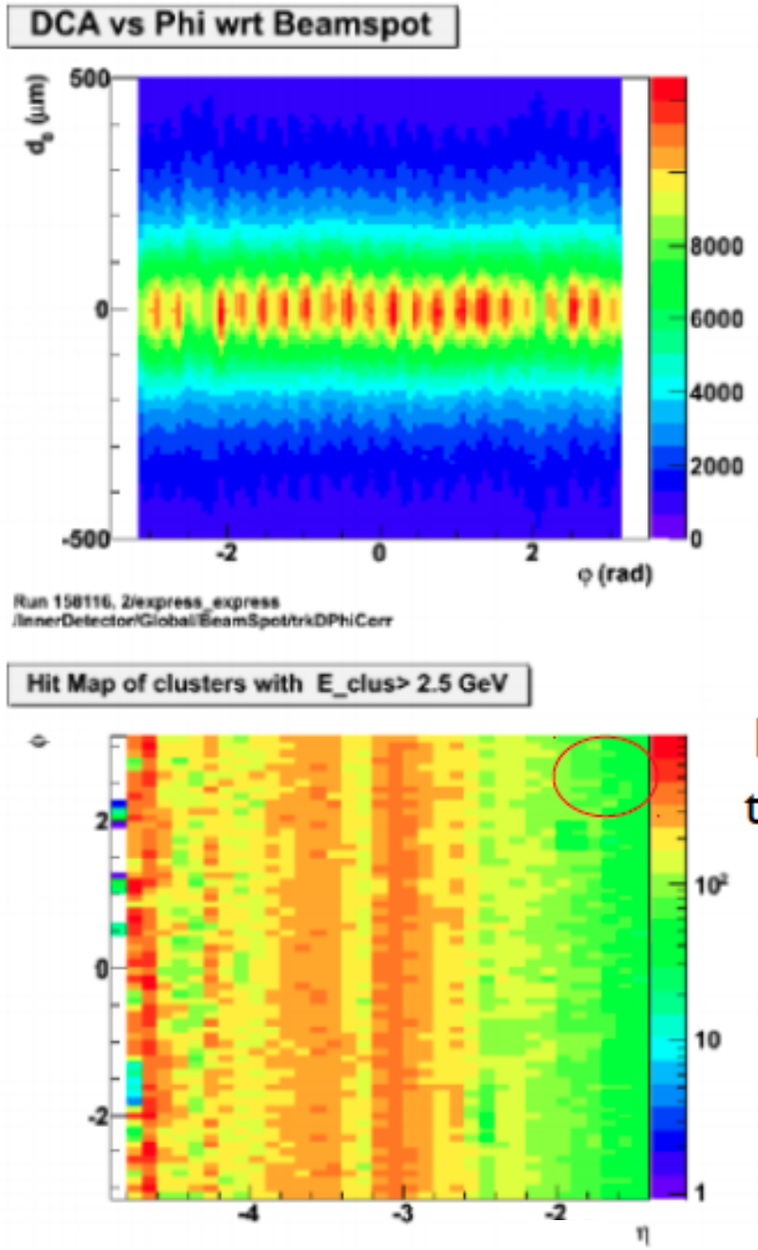
- The data are also used in the prompt calibration loop
  - Mask out noisy channels
  - Determine time-dependent calibrations and alignments
  - Determine the beamspot position for use in the bulk reconstruction
  - …etc…

- Not intended for physics

- Bulk reconstruction only starts once a run is released from the Calibration Loop, usually 48 hours after the run finishes

Express Stream & Prompt Calibrations





Beam spot before/after calibration loop

Hot channel in the calorimeter is masked in the physics stream processing.

- Bulk reconstruction only starts once a run is released from the Calibration Loop

Bulk reconstruction

- Bulk reconstruction uses the majority of the Tier0 resources
- Processes the RAW data using updated calibrations determined in the Prompt Calibration
- Produces many outputs used for a variety of purposes, the most important being the AOD

Data Quality and Luminosity

M

Summary Details

190503 - 191933

Peak Lumi	$3.61 \times 10^{33} cm^{-2} s^{-1}$
Peak Events/BX	32.06
Delivered Ready (pb^{-1})	1174.020
Recorded Ready (pb^{-1})	1121.768

CP and Physics				Inner Detector		Calorimeters		Muon Detectors		Trigger+Lumi	
Jet	95.3	MEt	94.3	Pixel	99.8	LAr EM	95.3	MDT	99.9	L1	98.6
Electron	93.7	Photon	93.7	SCT	99.7	LAr HAD	99.6	RPC	99.4	HLT	100.0
Muon	97.5	Tau	91.8	TRT	99.2	LAr FWD	99.7	CSC	100.0	LUMI	100.0
B-jet	93.7	Tracking	98.4			Tile	100.0	TGC	99.3		
All CP	90.9	Top Phys.	90.9								

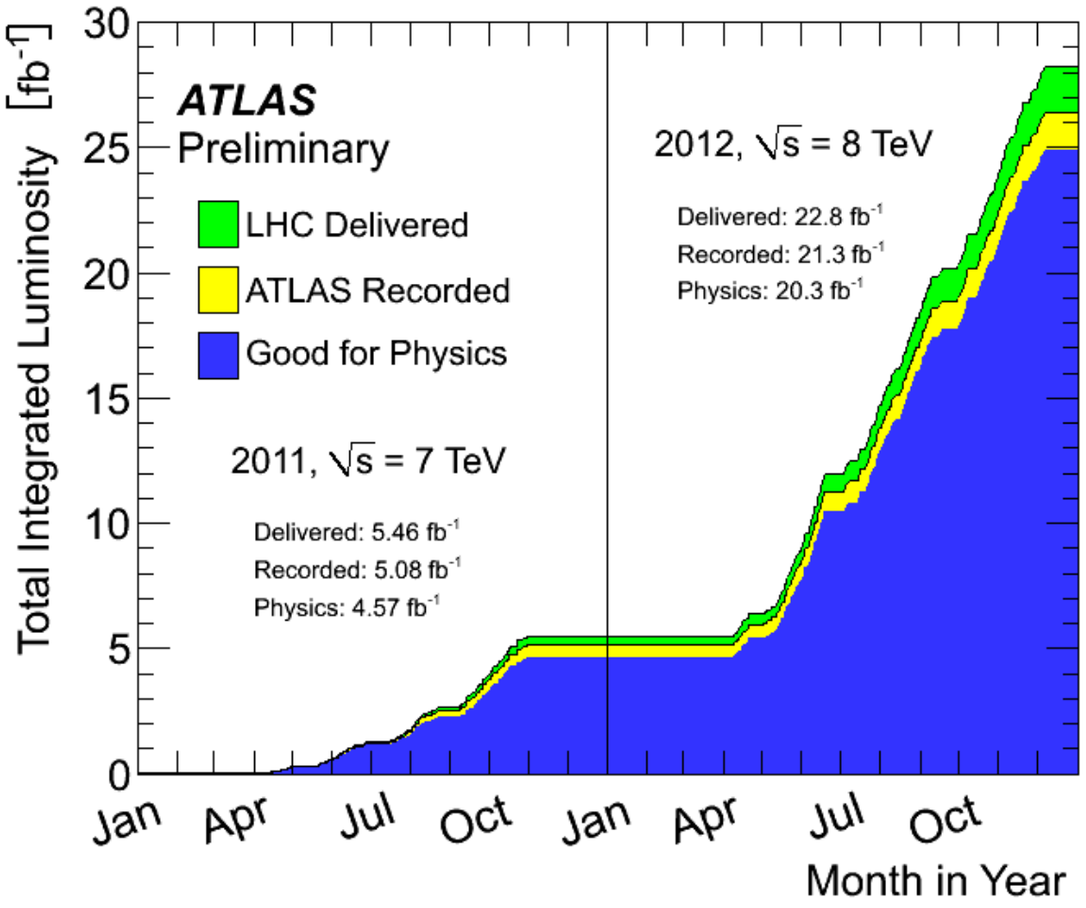
- Reconstruction jobs run monitoring algorithms used to determine
  - if the detectors are functioning properly
  - if the performance of the physics object (e.g. muon) reconstruction is acceptable
  - Distinguish tolerable from intolerable data quality “defects”

- Data Quality is assessed within 1-2 weeks of data-taking
- Data in specific luminosity blocks (~1 minute of data-taking) are flagged
  - should be ignored in analysis

- Used to create GoodRunLists that mask out these bad luminosity blocks in analysis

- The luminosity used in your analysis has to account for this missing data

Data Quality and Luminosity



Totals:

(proton-proton)
2012: 8 TeV, 21.7 fb^{-1} recorded
2011: 7 TeV, 5.2 fb^{-1} recorded
2010: 7 TeV, ~45 pb^{-1} recorded

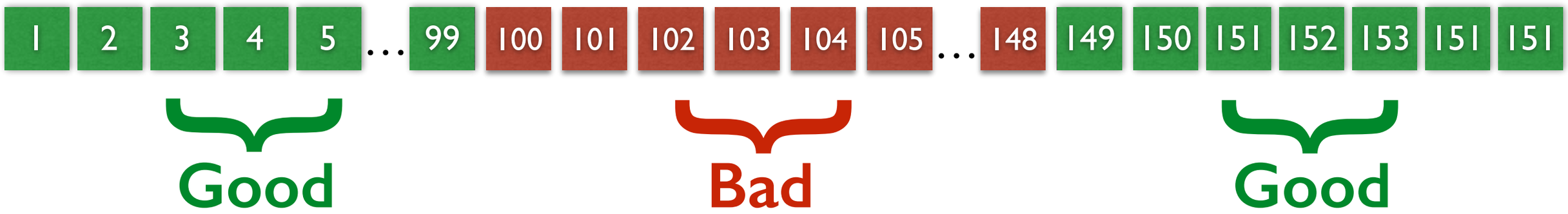
Also lead-lead, lead-proton, and lower energy p-p

ATLAS p-p run: April-December 2012										
Inner Tracker			Calorimeters		Muon Spectrometer				Magnets	
Pixel	SCT	TRT	LAr	Tile	MDT	RPC	CSC	TGC	Solenoid	Toroid
99.9	99.1	99.8	99.1	99.6	99.6	99.8	100.	99.6	99.8	99.5
All good for physics: 95.5%										

Luminosity weighted relative detector uptime and good quality data delivery during 2012 stable beams in pp collisions at Vs=8 TeV between April 4th and December 6th (in %) – corresponding to 21.3 fb^{-1} of recorded data.

Good runs lists

- Remember those luminosity blocks? ~ 1 minute of data taking
  - Several hundred lumi blocks per run; one or several lumi blocks per file
  - Some lumi block ranges will have *bad data quality* and should not be used for analysis



- The data quality flags are set in the data quality monitoring
- They are encoded in a database
- Different versions of the data quality flags get a different name, e.g. DetStatus-v61

Lumi blocks and analysis

- When you do your analysis you need to filter out the bad lumi blocks
- This is done by means of a *good runs list (GRL)*
  - XML file listing each run and the range of good lumi blocks in each: read in by the analysis code
  - GRLs are produced for each database tag
  - In principle there can be many types of GRL restricting the selection to (e.g.) muons or calorimetry, but in practice almost everyone uses the "All Good" lists
  - Location of GRLs: /afs/cern.ch/user/a/atlasdqm/grlgen/
  - GRL documentation: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/GoodRunListsForAnalysisRun2

What a GRL looks like inside


```
<LumiRangeCollection>
<NamedLumiRange>
<Name>PHYS_StandardGRL_All_Good</Name>
<Version>2.1</Version>
<Metadata Name="ARQEquivalentQuery">find run data12_8TeV.periodAllYear and dq PHYS_StandardGRL_All_Good DEFECTS#DetStatus-v61-pro14-02 g</Metadata>
<Metadata Name="Query">Period: data12_8TeV.AllYear; Defect: PHYS_StandardGRL_All_Good; Defect tag: DetStatus-v61-pro14-02; Ignoring None</Metadata>
<Metadata Name="RQTSVNVersion">DQDefects-00-01-00</Metadata>
<Metadata
Name="RunList">200842,200863,200913,200926,200965,200967,200982,200987,201006,201052,201113,201120,201138,201190,201191,201257,201269,201280,201289,201489,201494,201555,201556,202660,202668,202712,202740,202798,202965,202991,203027,203169,203191,203195,203228,203256,203258,203277,203335,203336,203353,203432,203454,203456,203523,203524,203602,203605,203636,203680,203719,203739,203745,203760,203779,203792,203875,203876,203934,204025,204026,204071,204073,204134,204153,204158,204240,204265,204416,204442,204474,204564,204633,204668,204726,204763,204769,204772,204796,204853,204857,204910,204932,204954,204955,204976,205010,205016,205017,205055,205071,205112,205113,206368,206369,206409,206497,206573,206614,206955,206962,206971,207044,207046,207221,207262,207304,207306,207332,207397,207447,207490,207528,207531,207532,207582,207589,207620,207664,207696,207749,207772,207800,207809,207845,207864,207865,207931,207934,207975,207982,208123,208126,208179,208184,208189,208258,208261,208354,208485,208631,208642,208662,208705,208717,208720,208780,208781,208811,208870,208930,208931,208970,208982,209024,209025,209074,209084,209109,209161,209183,209214,209254,209265,209269,209353,209381,209550,209580,209608,209628,209629,209736,209776,209787,209812,209864,209866,209899,209980,209995,210302,210308,211620,211670,211697,211772,211787,211867,211902,211937,212034,212103,212142,212144,212172,212199,212272,212619,212663,212687,212721,212742,212809,212815,212858,212967,212993,213039,213079,213092,213130,213155,213157,213204,213250,213359,213431,213479,213486,213539,213627,213640,213684,213695,213702,213754,213796,213816,213819,213900,213951,213964,213968,214021,214086,214160,214176,214216,214388,214390,214494,214523,214544,214553,214618,214651,214680,214714,214721,214758,214777,215027,215061,215063,215091,215414,215433,215456,215464,215473,215541,215571,215589,215643</Metadata>
<LumiBlockCollection>
<Run>200842</Run>
<LBRange Start="27" End="36"/>
<LBRange Start="43" End="47"/>
<LBRange Start="49" End="85"/>
</LumiBlockCollection>
<LumiBlockCollection>
<Run>200863</Run>
<LBRange Start="57" End="74"/>
<LBRange Start="76" End="83"/>
<LBRange Start="86" End="156"/>
<LBRange Start="158" End="165"/>
<LBRange Start="167" End="191"/>
…….
…….
<LumiBlockCollection>
<Run>215643</Run>
<LBRange Start="400" End="460"/>
<LBRange Start="462" End="479"/>
<LBRange Start="481" End="493"/>
<LBRange Start="505" End="509"/>
<LBRange Start="511" End="517"/>
<LBRange Start="522" End="567"/>
</LumiBlockCollection>
</NamedLumiRange>
</LumiRangeCollection>
```

GRLs and luminosity calculation

- The luminosity "seen" by your analysis depends on three things:
  - the luminosity blocks processed (from the GRL)
  - the trigger prescales applied
  - the level-1 trigger live-time (fraction of delivered luminosity that ATLAS recorded)
- To get the luminosity you pass the GRL to the lumi-calc tool and provide it with the trigger you used
- The tool will calculate the luminosity by adding up the lumi-blocks and applying the prescales to each
  - Uses both the trigger DB and the luminosity DB
- **Important: if you use the number from this tool in your analysis, you MUST ensure that you processed ALL of your data!**
  - You will find out how to do this later in the week…

GRLs and luminosity calculation

https://atlas-lumicalc.cern.ch

Recommendations

Data Sample	Recommended Luminosity Tag	Recommended Livefraction Trigger	Comments
data13_hip	OflLumi-HI-002	L1_LUCID_A_C	Preliminary vdM calibration
data13_2p76TeV	OflLumi-2p76TeV-002	L1_LUCID_A_C	Final 2.76 lumi
data12_8TeV	OflLumi-8TeV-003	L1_EM30	Preliminary 2012 vdM calibration
data11_7TeV	OflLumi-7TeV-004	L1_EM30	Final 2011 results (with beam-beam corrections)
data10_7TeV	OflLumi-7TeV-004	L1_MBTS_2	Final 2010 results (with beam-beam corrections)
data11_2p76	OflLumi-2p76TeV-002	L1_LUCID_A_C	Final 2.76 lumi
data10_hi	OflLumi-HI-000	L1_LUCID_A_C	Online estimate

Luminosity Calculator

See below for a synopsis of these quantities.

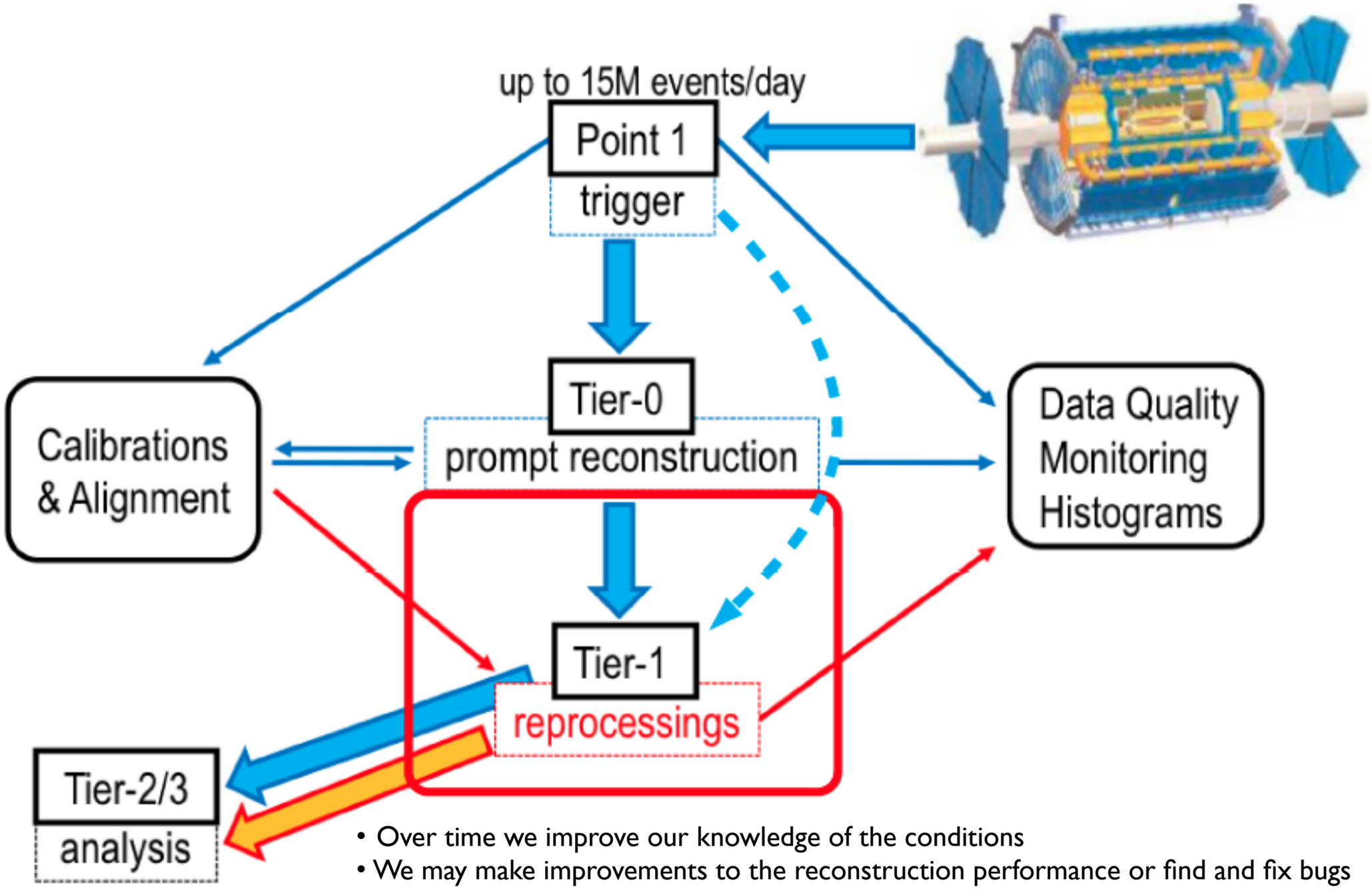
GRL XML File:	Choose File no file selected	GRL XML file which defines the data sample of interest. See below for information how to specify a subset of a GRL.
Luminosity Tag:	OflLumi-8TeV-003 (preferred 2012)	Tag which defines the specific version of the luminosity measurement. Only use --online for diagnostic checks.
Live Fraction Trigger:	L1_EM30	This trigger is only used to compute the L1 live fraction. Any high rate L1 trigger can be used. Please see below for recommendations.
Physics Trigger:	None	This trigger is used to compute the prescale correction. It is possible to use a L1 trigger, but if that trigger is prescaled at the HLT, you will get the wrong result! Specify None to assume a prescale of 1 for all luminosity blocks.
LAr EventVeto Tag:	LARBadChannelsOflEventVeto-UPD4-04 (recommended)	Tag which defines the LAr Event Veto information. Must be used to assess LAr Event Veto inefficiency. Use None to turn this off. See below or here for details.
Other Options:	☐ Calculate BGRP7 luminosity (--lumichannel=17) ☐ Require online beamspot (--beamspot) ☐ Create output plots and ntuples (--plots) ☐ Verbose output (--verbose) Additional options	Add any additional options here. Use --help, or just click the Calculate Luminosity button below with no GRL specified, to find other possibilities.

Remember, this process can take considerable time (minutes) for large data samples. Pushing the button again will only make things worse. Please be patient!

Calculate Luminosity

Reprocessing

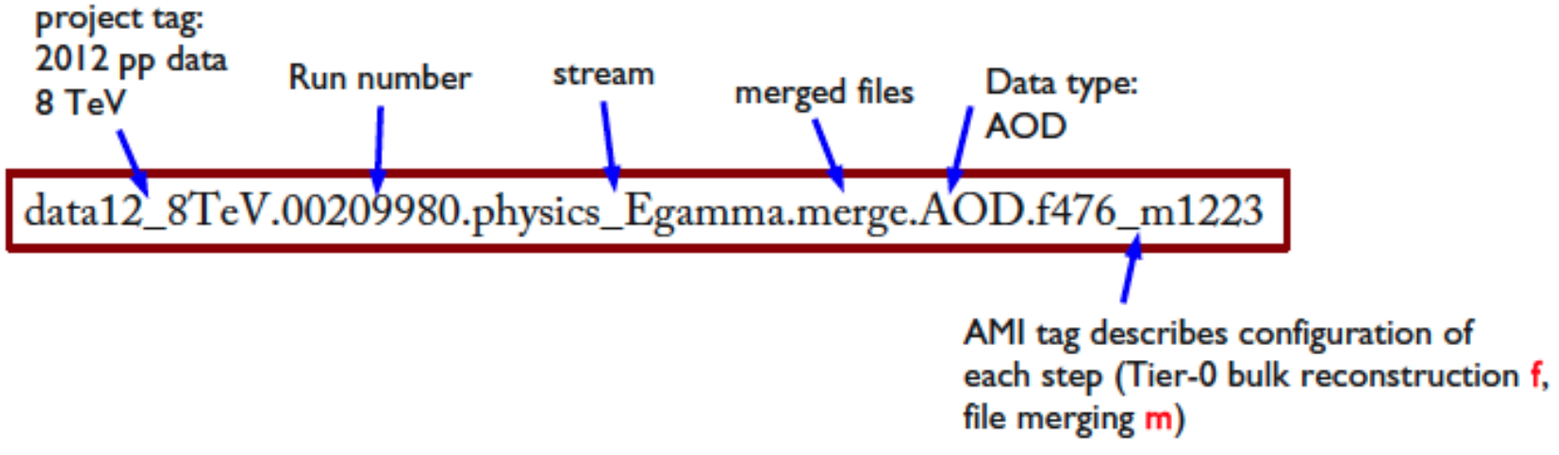
Reprocessing at Tier-1s



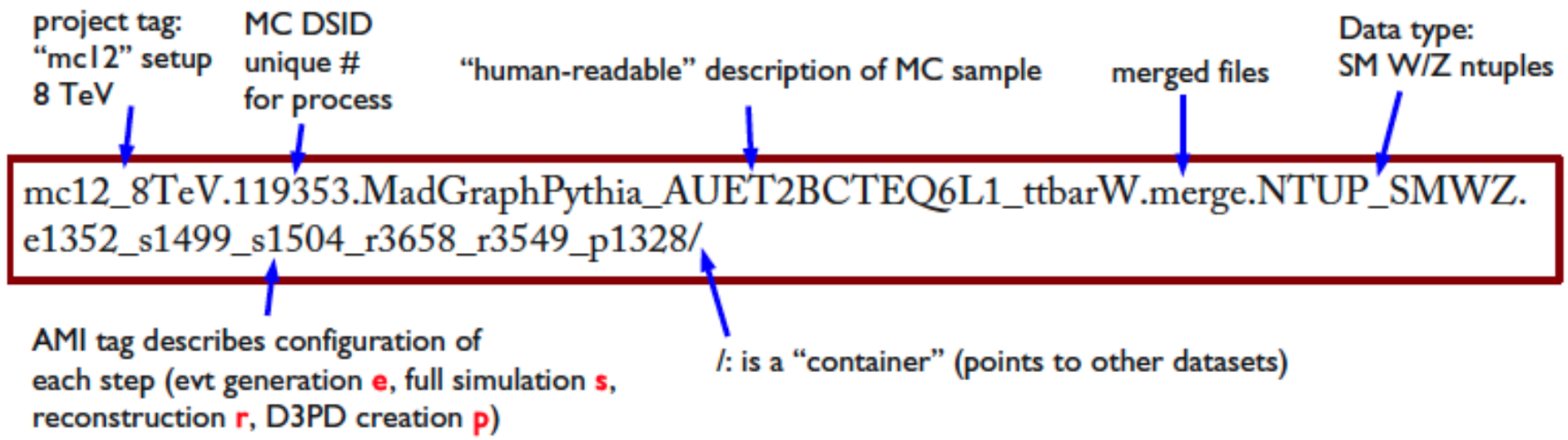
- Over time we improve our knowledge of the conditions
- We may make improvements to the reconstruction performance or find and fix bugs
- After a time, we reprocess the existing raw data with the improved software/conditions
- This has to be done on the Grid since the Tier-0 will be busy with prompt data processing
- Once we start a reprocessing, we also switch the Tier-0 to the new software to ensure a consistent dataset

Dataset nomenclature

Data:



Simulation:



- Datasets are collections of files, form fundamental unit of ATLAS data and MC
- Each processing step changes the data type, and adds the AMI tag used
- Changes with Run 2
  - the merge m-tag has mostly disappeared for reprocessed data and MC
  - the p-tag will in general refer to the Derivation Framework version - DAOD

Dataset Nomenclature & Analysis

- Real data is grouped together into “periods” with ~stable running conditions
  - 2012 data broken down into periods A-L
  - “AllYear” covers the whole year

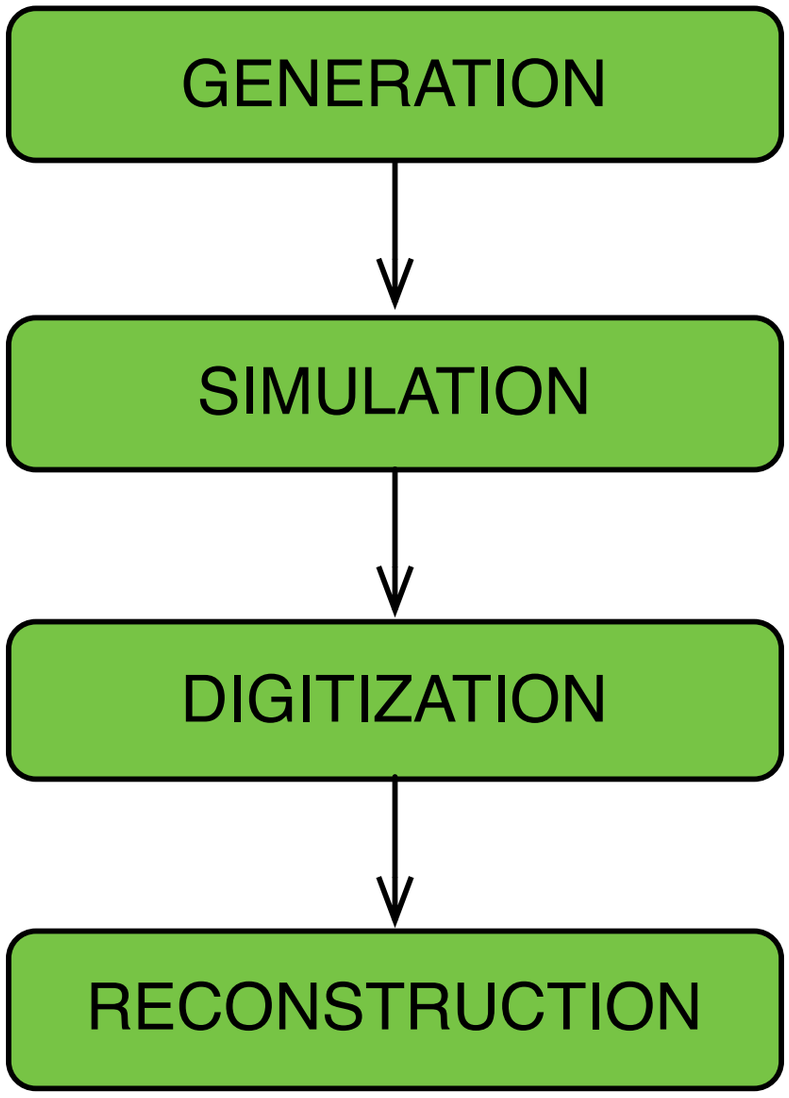
- Possible to have several “processing campaigns” for a given dataset
  - First processing at Tier0 is e.g. t0pro14
  - A reprocessing campaign at Tier1s is e.g. repro14
  - The combination with overlaps removed pro14

- We create “containers” to collect several datasets that belong together, easier for use in your analysis, e.g.:
  - data12_8TeV.periodAllYear.physics_Muons.PhysCont.AOD.pro14_v2/

- Finding data: see the tutorial on AMI and use the portal here: http://ami.in2p3.fr/
- Getting data: see the tutorials on DQ2/Rucio later this week

- And finally, a useful link for analysis in general, and in particular the current ‘best version’:
  - https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/DataMCForAnalysis

Monte Carlo Production

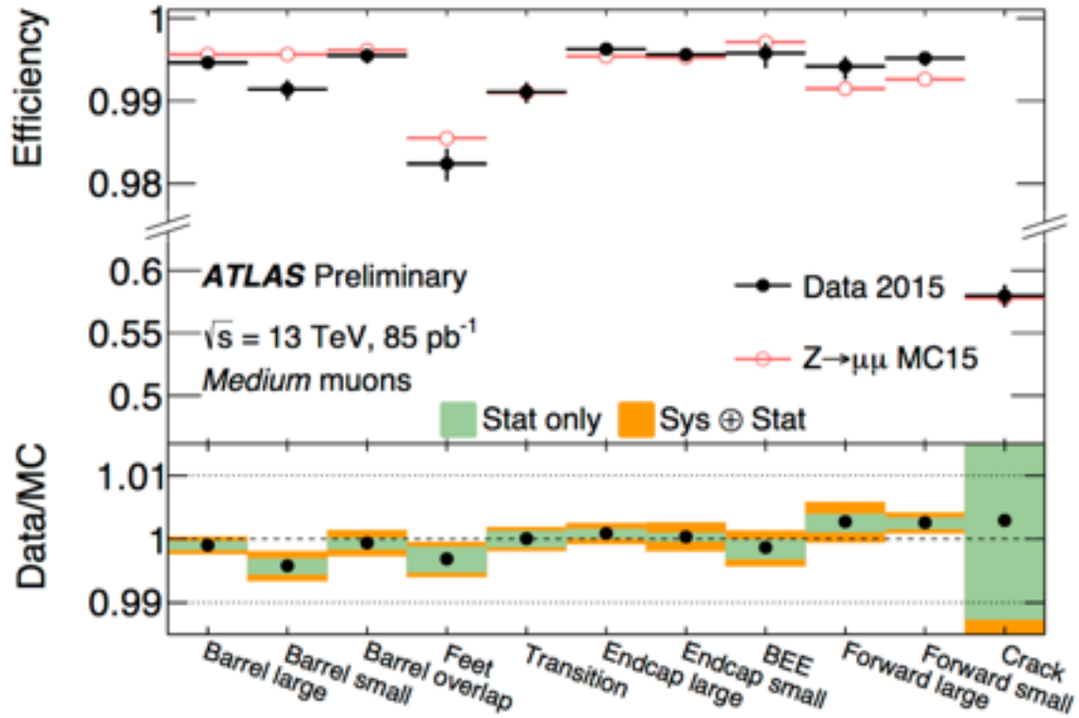


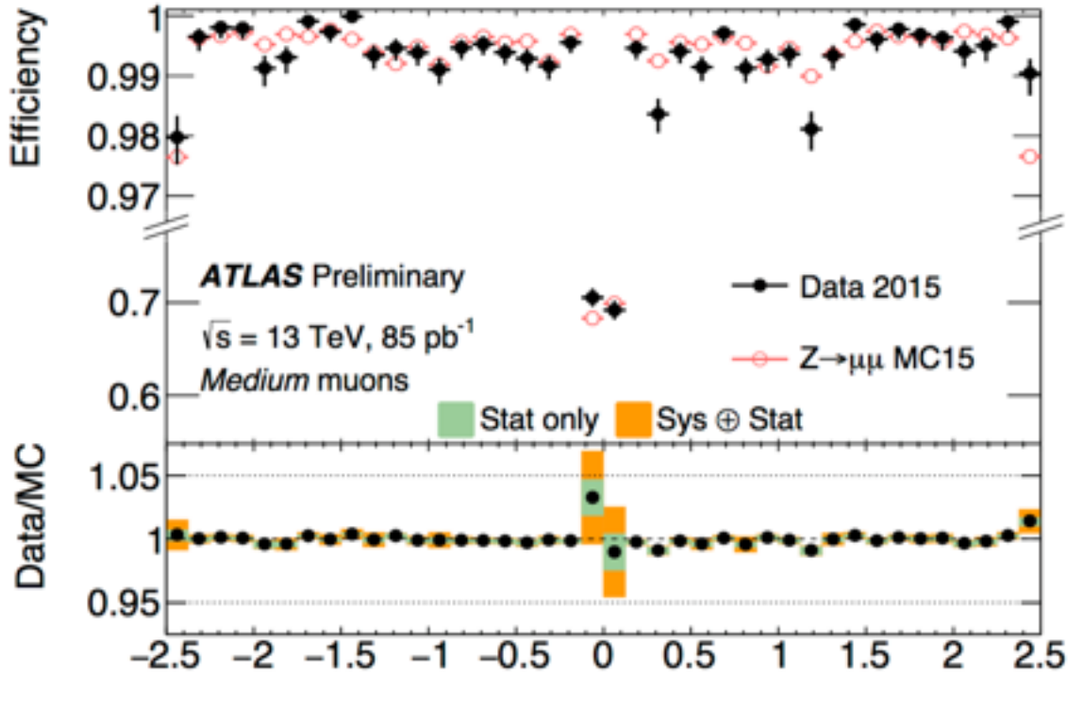
MC tutorial: https://indico.cern.ch/event/440423

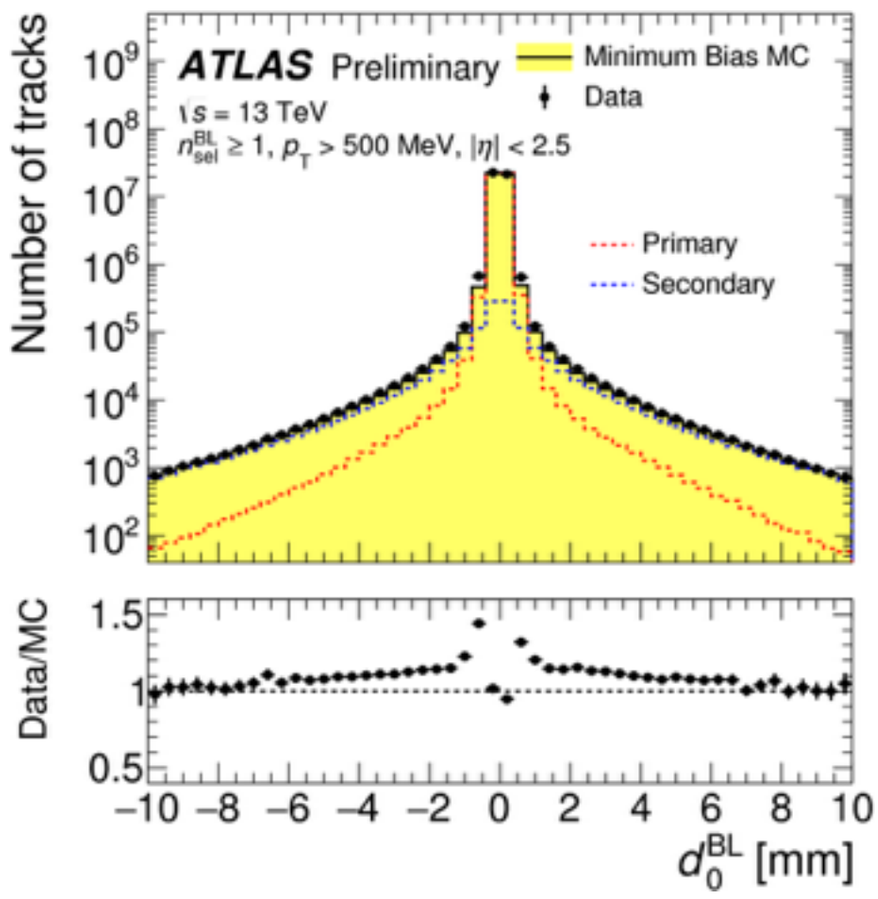
Why do we need Monte Carlo?

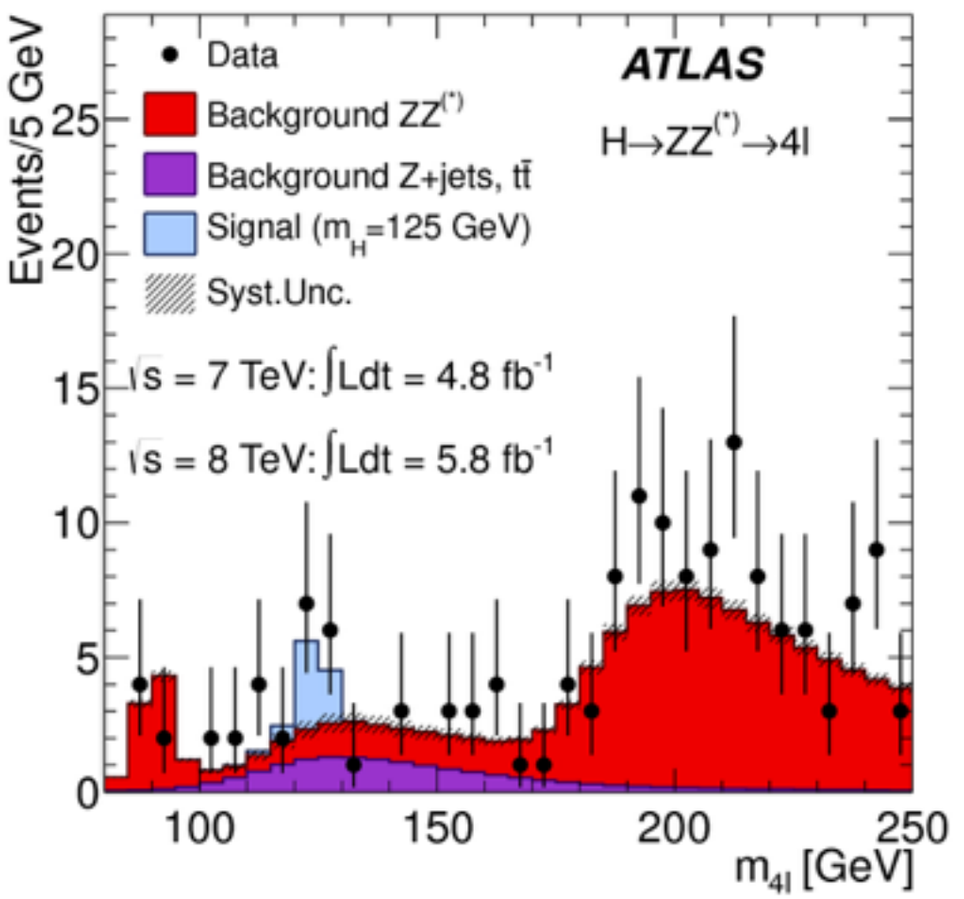
- Testing whether we understand the performance of the detector
- Calculating reconstruction efficiencies (usually for validating data-driven methods)
- Modelling the expected background under a process of interest
- Modelling the process of interest
- Training multivariate classifiers (neural networks, boosted decision trees etc)
- Setting systematic uncertainties
- etc etc etc

Why do we need Monte Carlo?





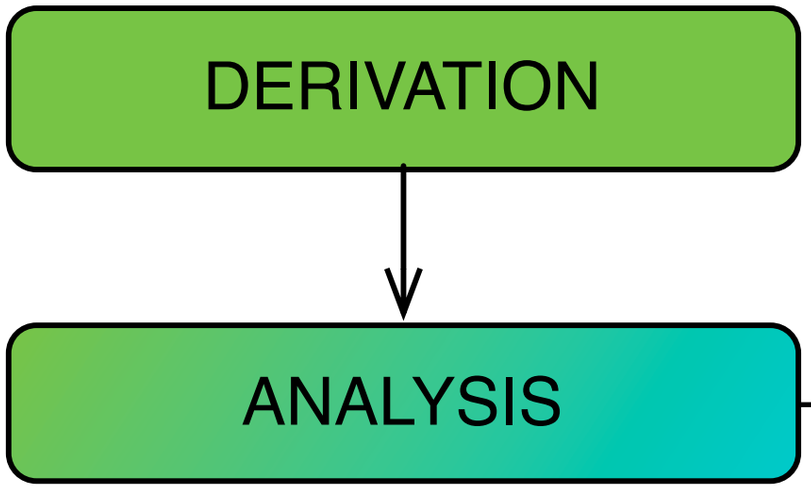


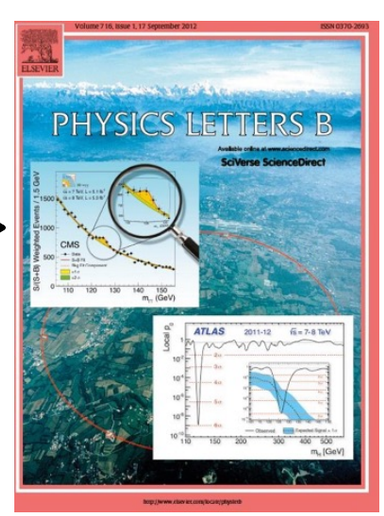


Monte Carlo production

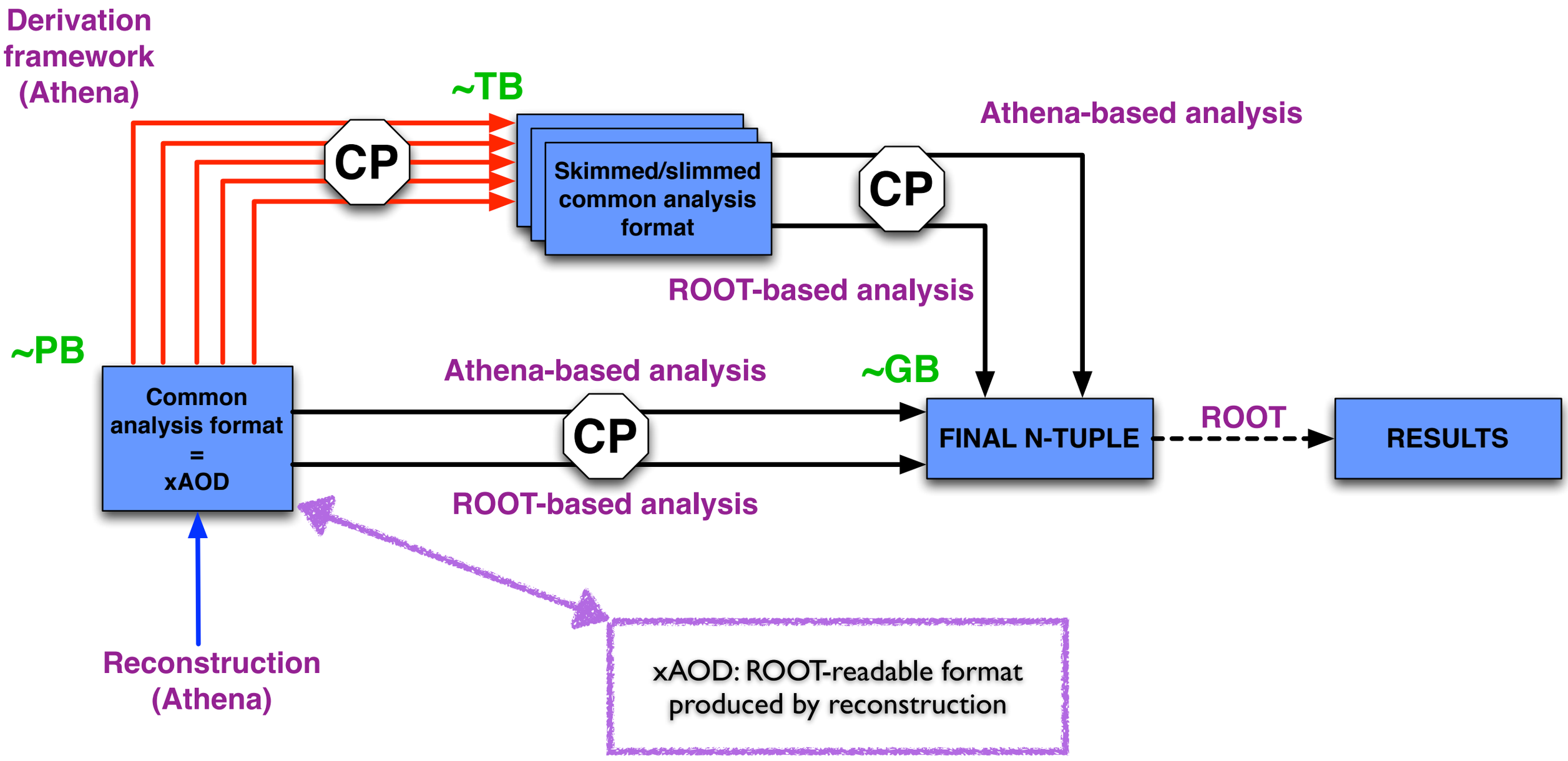
- All official Monte Carlo production is done on the Grid
- Several-step process
  - *Event generation*: simulation of the interaction between the quarks and gluons in the colliding protons, the subsequent parton showering and hadronization and decays into stable particles
  - *Detector simulation:* calculation of how the particles from the generator interact with the detector material; how they shower into secondaries; how much energy they deposit in each sensitive element
  - *Digitisation*: turning the simulated energy deposits into a detector response that "looks" like the raw data from the real detector
  - After this step, the process is the same as for real data
- The analysis data for MC and real data looks the same, *except* that in MC the original generated events (the "truth") are available as well as the reconstructed objects
- Extra low momentum events must be injected into the chain to simulate the presence of multiple proton collisions ("pile-up") in a given LHC bunch crossing
  - Complication: the average number of collisions per bunch crossing is a function of the LHC parameters, and we typically do not know this when we start the Monte Carlo production
  - Monte Carlo events must be weighted in analysis to account for discrepancies between the real and simulated pile-up. This re-weighting is something you will learn to "love" during your PhD

The ATLAS Analysis Model and the xAOD



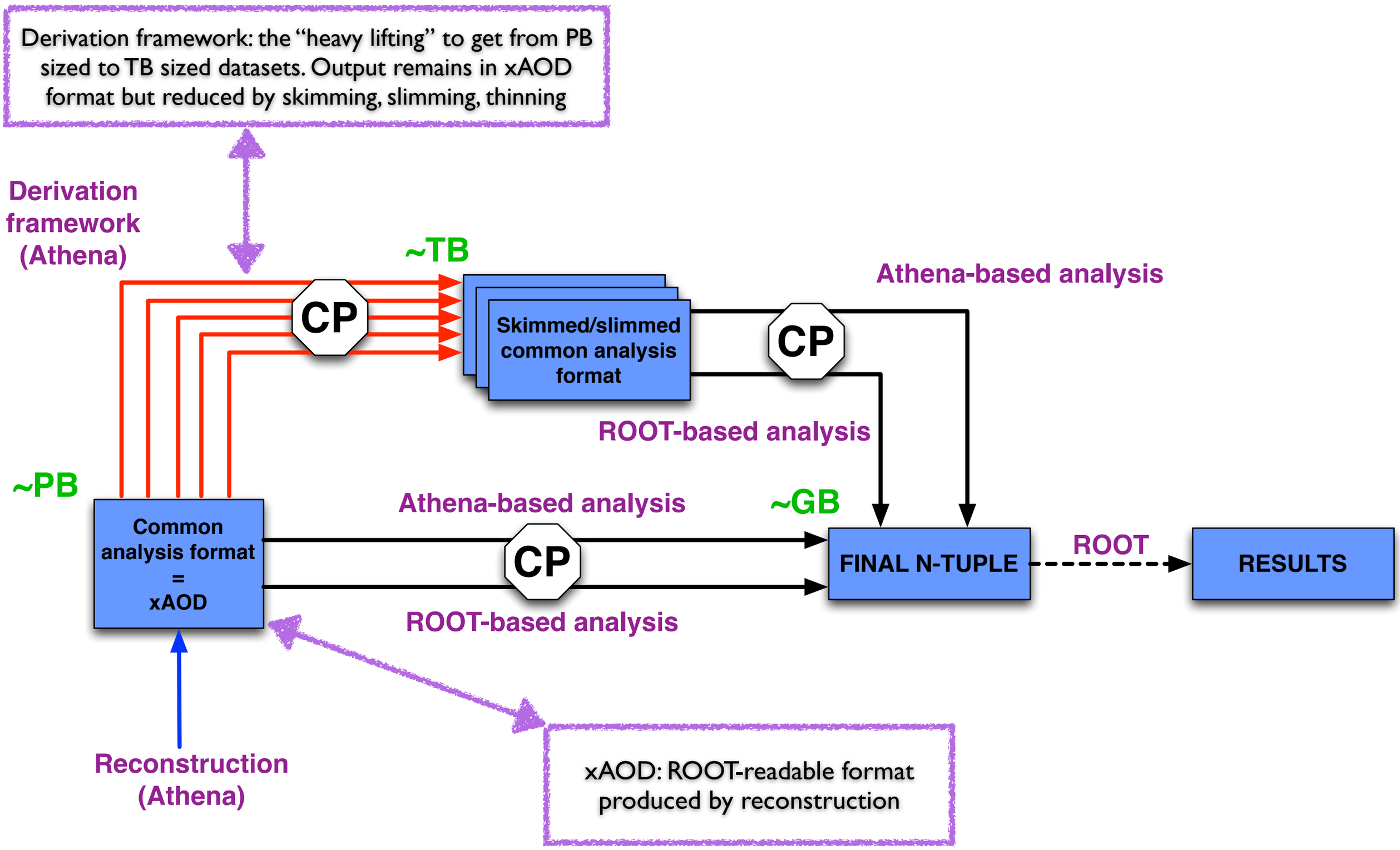


The ATLAS analysis model



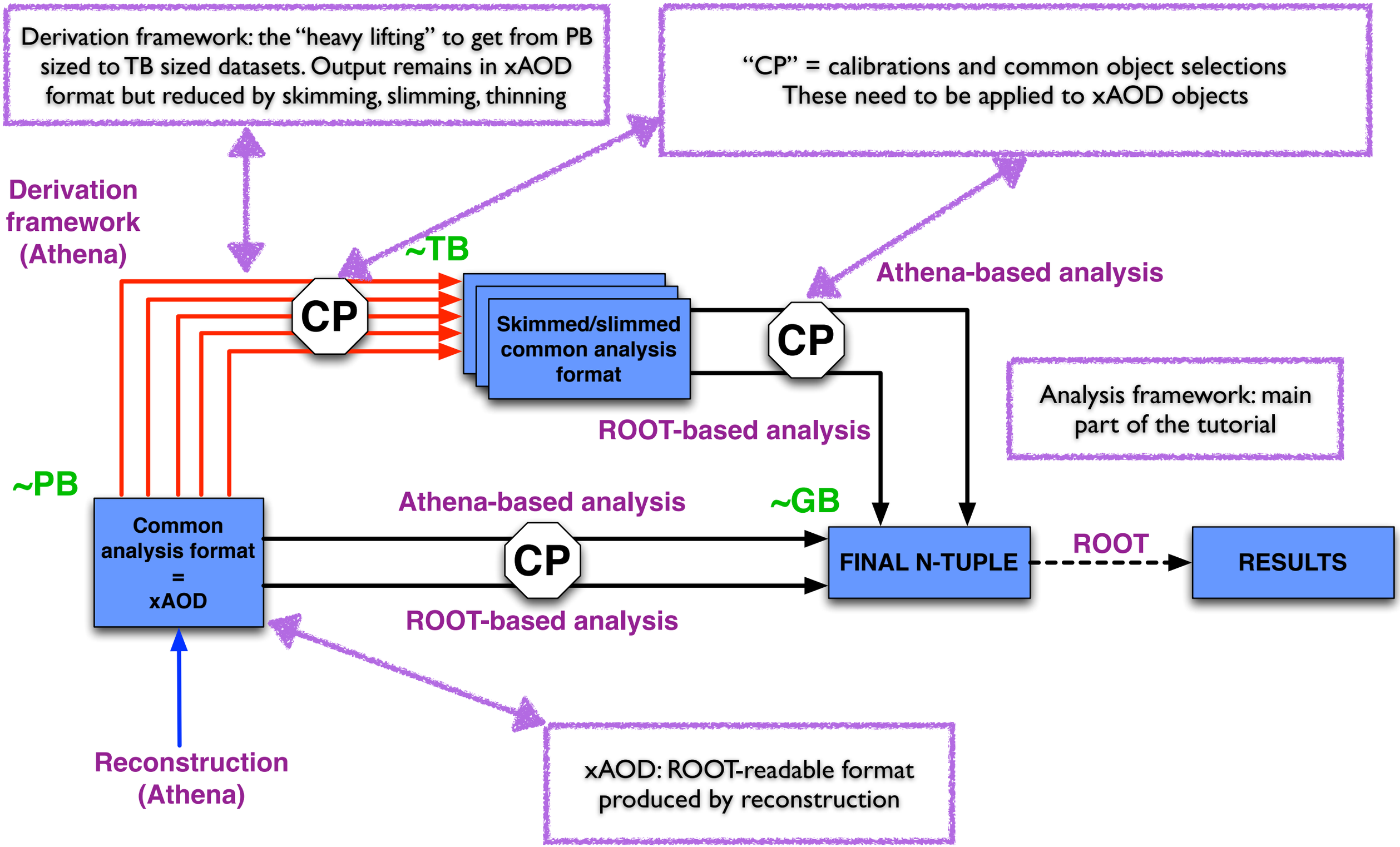
The ATLAS analysis model

Derivation framework: the "heavy lifting" to get from PB sized to TB sized datasets. Output remains in xAOD format but reduced by skimming, slimming, thinning



xAOD: ROOT-readable format produced by reconstruction

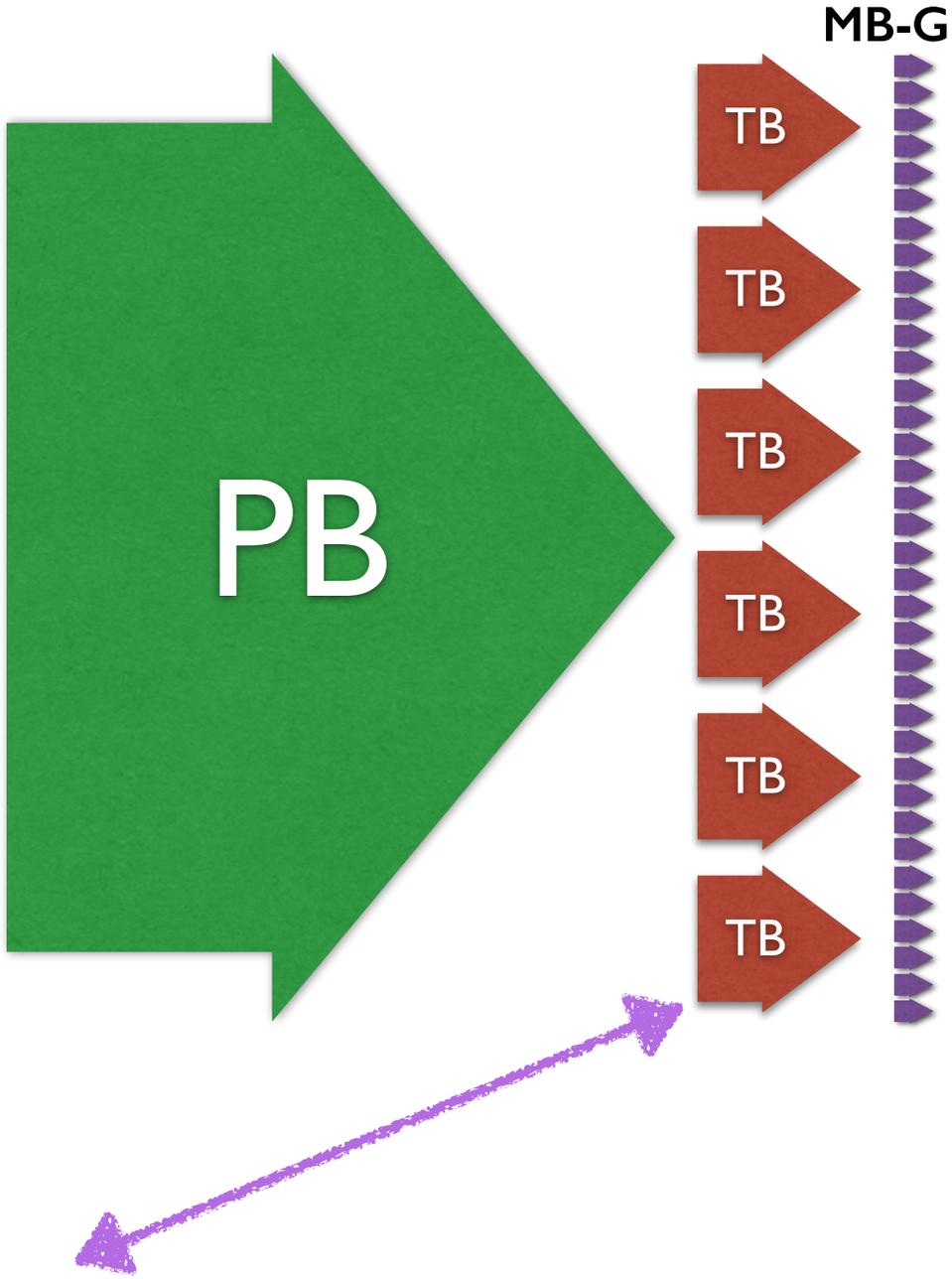
The ATLAS analysis model



Components of the analysis model

- xAOD
  - ROOT-readable data format and event data model
  - Produced by directly by reconstruction
  - Much more on this later
  - Note: we refer to the *files* as AODs, but the *contents* as xAOD. Confusing, I agree.
- Derivation framework
  - AODs are too big to analyse directly so we centrally reduce them according to the needs of the physics groups, producing DAODs which are still made of xAOD objects, but are much smaller
  - Avoids users having to do this themselves on the Grid
  - Currently producing close to 90 DAOD formats for all ATLAS activities
- Analysis framework
  - What you use to do your analysis
  - Capable of reading the xAOD objects and applying all tools from the combined performance groups
  - Currently we have three centrally provided frameworks (in addition some people wrote their own)
    - Athena/AthAnalysisBase, EventLoop
    - All read the xAOD objects and can use the common tools; we will focus on AthAnalysisBase in this tutorial

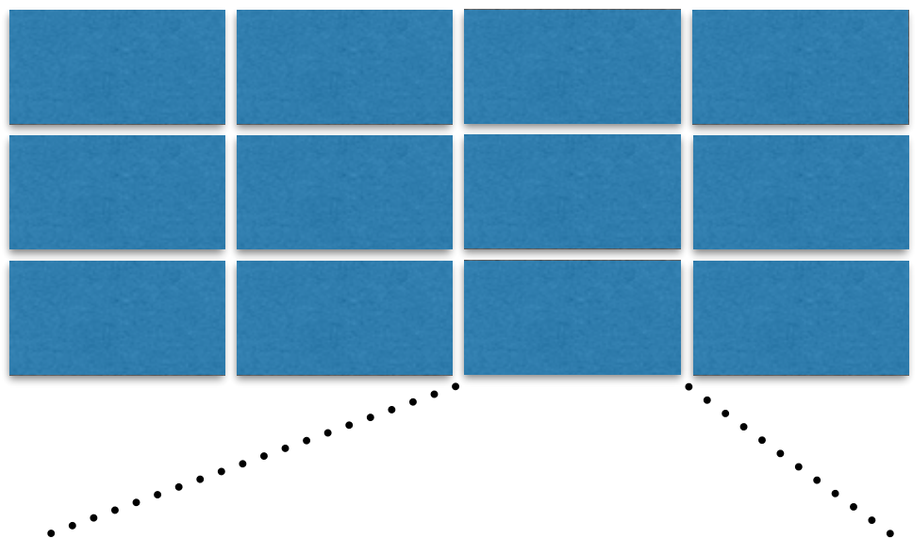
Derivation step



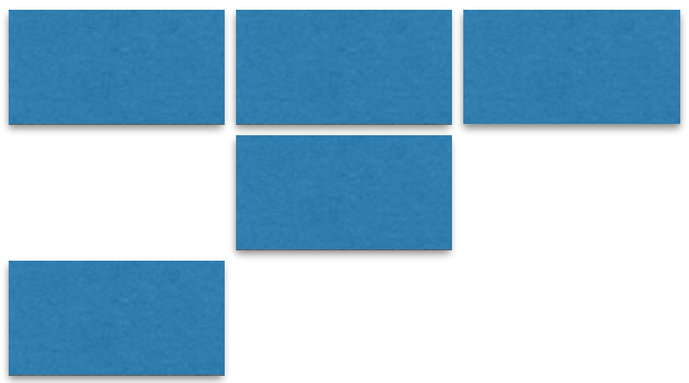
	Full output of reconstruction, ~PB size	One format
	Intermediate analysis format ~TB size	~100 formats
	Final n-tuple ~MB-GB size	~1000 formats

- These formats tend to be specific to a single analysis or group of analyses
- Calibrations and common object selections are often applied as they are made
- They generally need to contain all variables needed for calculating systematics

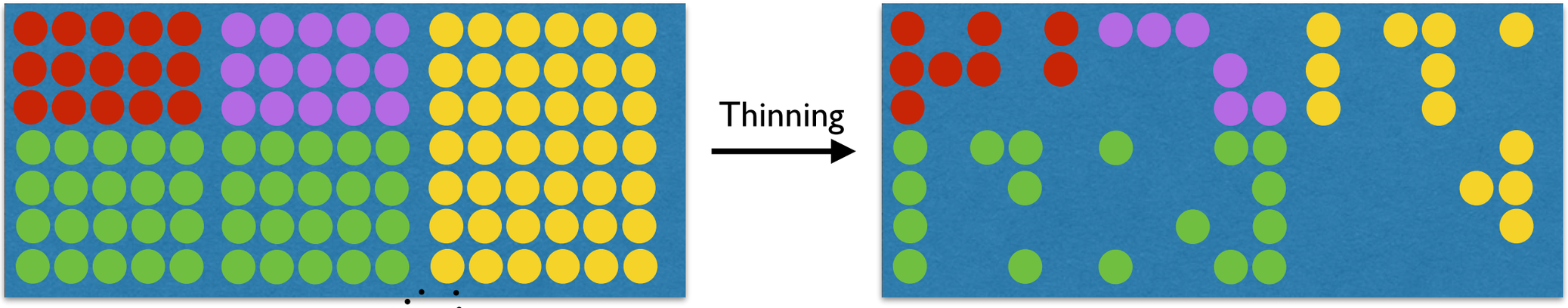
Derivation step: data reduction



Skimming

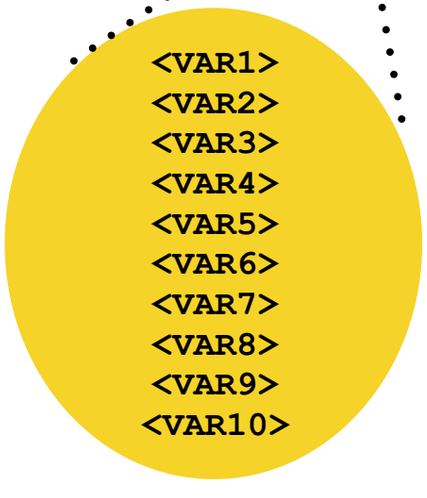


Skimming:
removal of whole events based on pre-set criteria

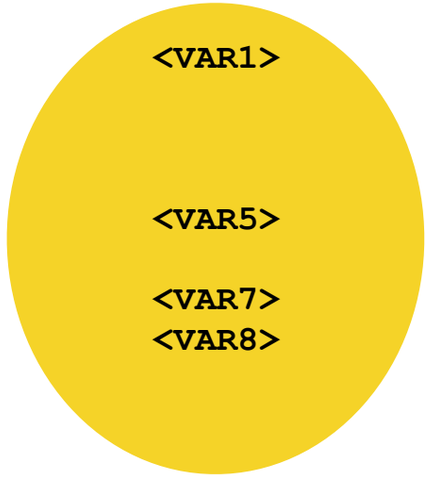


Thinning

Thinning:
removal of whole objects within events based on pre-set criteria



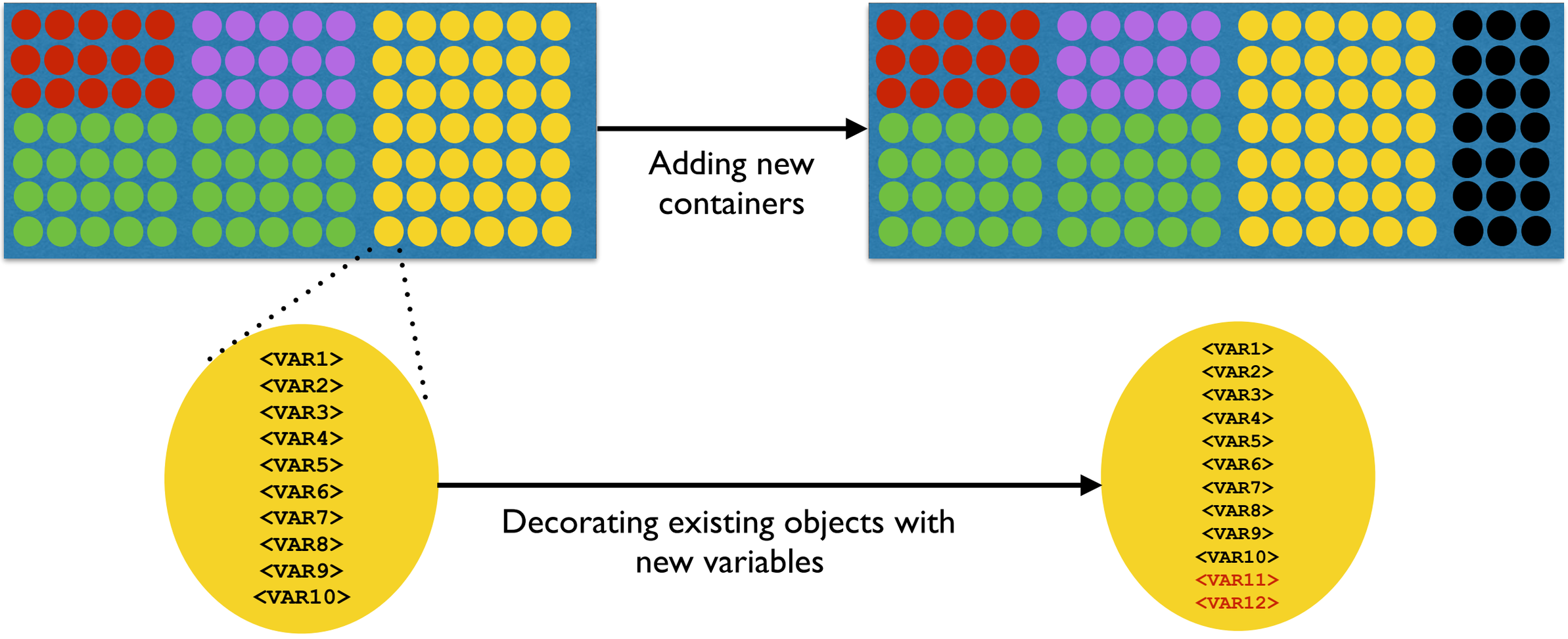
Slimming



Slimming:
removal of variables within objects uniformly across events

Derivation step: augmentation

- New information (augmentation) is typically done in two ways:
  - Adding new reconstructed object containers: typically jets made with a modified algorithm.
  - Decorating existing objects with extra variables: typically the results of object selection by combined performance tools (e.g. "this is a good muon")



The xAOD

- ROOT-readable data format
  - AODs produced by the reconstruction or DAODs from the derivation framework can be immediately read (and browsed) in ROOT
  - Doesn't sound very impressive…? Well it is!
  - In run 1 the AODs could not be read by ROOT so we had to convert them to n-tuples first
    - A bit like having to stream Netflix movies onto VHS tapes before you can watch them…
    - The xAOD system has been a massive improvement (and took two years of work by some of the best software experts at CERN)
- Common event data model (EDM)
  - Single set of objects representing reconstructed physical objects (tracks, muons, electrons, photons etc) used across ATLAS, from reconstruction to final analysis
  - This is a similarly revolutionary idea in ATLAS
  - Means in particular that analysis tools can be applied anywhere in the chain

xAOD basics

- The xAOD consists of
  - Information about the event (EventInfo)
  - Information about reconstructed objects within each event
    - Tracks, muons, electrons, jets etc
    - These all inherit from a common class Particle
- Information in the xAOD is split into
  - the objects themselves
  - the numerical payload for the objects: the *Auxiliary Store*
- The splitting enables on-demand reading of a given variable across many objects/events without having to load the full event
  - Enables browsing in ROOT, fast **`TTree->Draw()`** etc
  - Enables decoration of objects with new variables
- In analysis code you never need to interact with the auxiliary store - just the objects
  - In ROOT browsing it appears as a normal TTree called **`CollectionTree`**
- All xAOD objects are in the **`xAOD::`** namespace

xAOD::EventInfo

- It contains information about the given event, e.g.:
  - What was the pile-up for this event?
  - What is the current run number, event number, luminosity block number?
- There is one and only one object of this type for a given 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.
void	**setBCID** (uint32_t value) Set the bunch crossing ID of the event.

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

xAOD::IParticle

- Pure virtual interface class for:
  - all objects of particle type (electron, muon, jet,…)
  - clustered energy deposits in the calorimeter
  - reconstructed charged particle trajectories

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

xAOD::IParticle

- Provides access to 4-momentum (from Doxygen):

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.

Some more detail

- All particle objects are stored in "containers"
  - such that you can iterate over all jets in a given event
  - (again, the actual payload is in the auxiliary container)
- Objects can have references ("`ElementLink`") to other objects
  - **e.g., an** `xAOD::Electron` **has an** `ElementLink` **to its** `xAOD::TrackParticle` **and an** `ElementLink` **to its** `xAOD::CaloCluster`

xAOD: Easier for tool usage

- Call a method of a tool with D3PDs/ntuples:

```
double getCorrectedEnergy( unsigned int runnumber,
                           PATCore::ParticleDataType::DataType dataType,
                           PATCore::ParticleType::Type ptype,
                           float rawcl_Es0,
                           float rawcl_Es1,
                           float rawcl_Es2,
                           float rawcl_Es3,
                           float cl_eta,
                           float cl_phi,
                           float trk_eta,
                           float cl_E,
                           float cl_etaCalo,
                           float cl_phiCalo,
                           float ptconv = -999,
                           float pt1conv = -999 ,
                           float pt2conv = -999,
                           int convtrk1nPixHits = -999,
                           int convtrk1nSCTHits = -999,
                           int convtrk2nPixHits = -999,
                           int convtrk2nSCTHits = -999,
                           float Rconv = -999,
                           egEnergyCorr::Scale::Variation scaleVar = egEnergyCorr::Scale::None,
                           egEnergyCorr::Resolution::Variation resVar = egEnergyCorr::Resolution::None,
                           egEnergyCorr::Resolution::resolutionType resType = egEnergyCorr::Resolution::SigmaEff90,
                           double varSF = 1.0 ) const;
```

- With xAOD:

```
CorrectionCode applyCorrection( xAOD::Muon& myMuon );
```

xAOD: How does it look like?

ROOT Object Browser

Browser File Edit View Options Tools Help

Files

Draw Option:

ROOT Files
AOD.01572118._003538.pool.root.6
##Shapes;1
##Links;1
##Params;1
CollectionTree;1
AODCellContainer
AntiKt10LCTopoJets
@size
AntiKt10LCTopoJetsAux.
AntiKt10LCTopoJetsAux.xAOD::AuxContainerBa
AntiKt10LCTopoJetsAux.pt
AntiKt10LCTopoJetsAux.eta
AntiKt10LCTopoJetsAux.phi
AntiKt10LCTopoJetsAux.m
AntiKt10LCTopoJetsAux.constituentLinks
AntiKt10LCTopoJetsAux.constituentWeights
AntiKt10LCTopoJetsAuxDyn.ActiveArea
AntiKt10LCTopoJetsAuxDyn.ActiveArea4vec_eta
AntiKt10LCTopoJetsAuxDyn.ActiveArea4vec_m
AntiKt10LCTopoJetsAuxDyn.ActiveArea4vec_phi
AntiKt10LCTopoJetsAuxDyn.ActiveArea4vec_pt
AntiKt10LCTopoJetsAuxDyn.AlgorithmType
AntiKt10LCTopoJetsAuxDyn.Angularity
AntiKt10LCTopoJetsAuxDyn.Aplanarity
AntiKt10LCTopoJetsAuxDyn.AverageLArQF

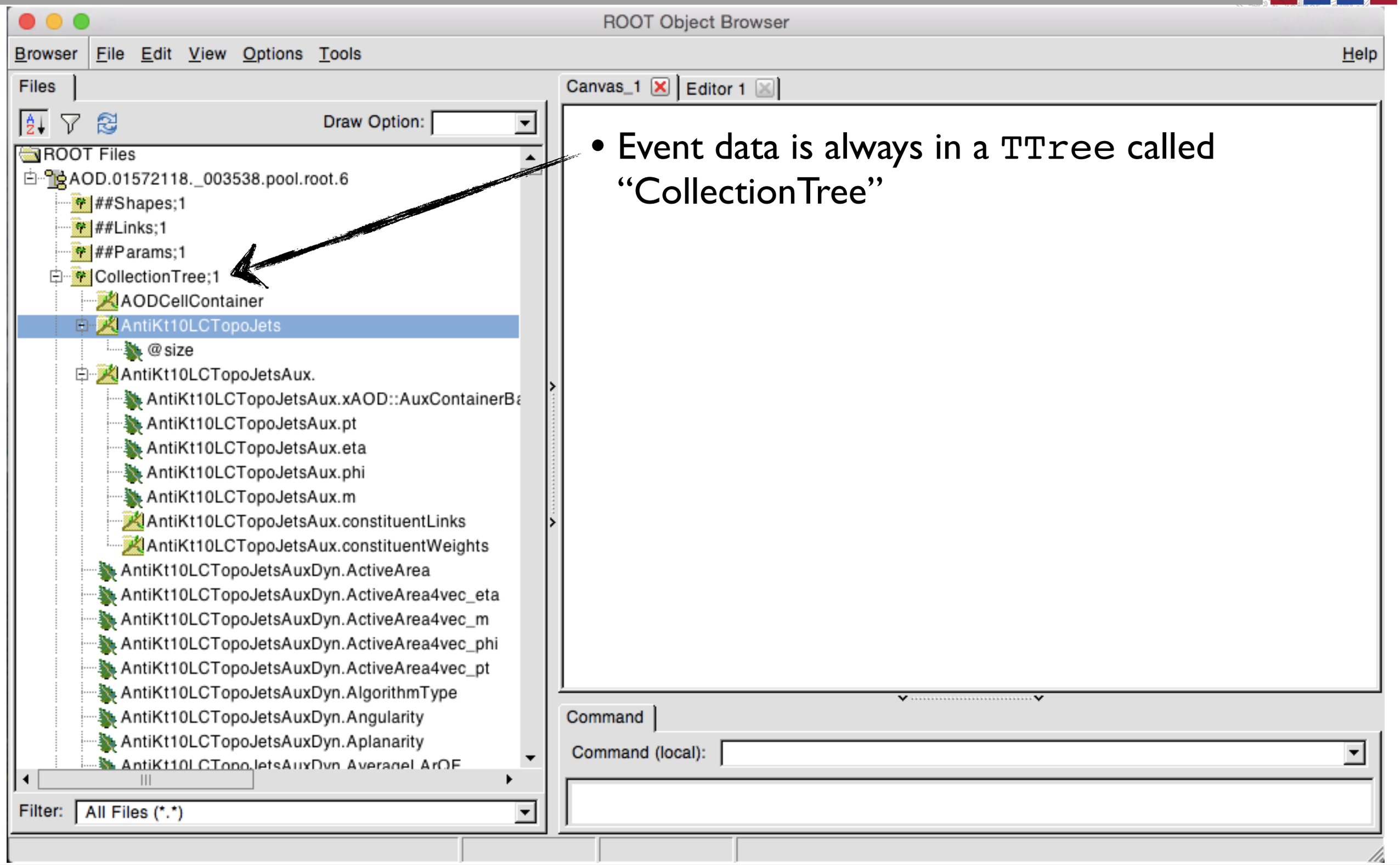
Filter: All Files (*.*)

Canvas_1 Editor 1

Command

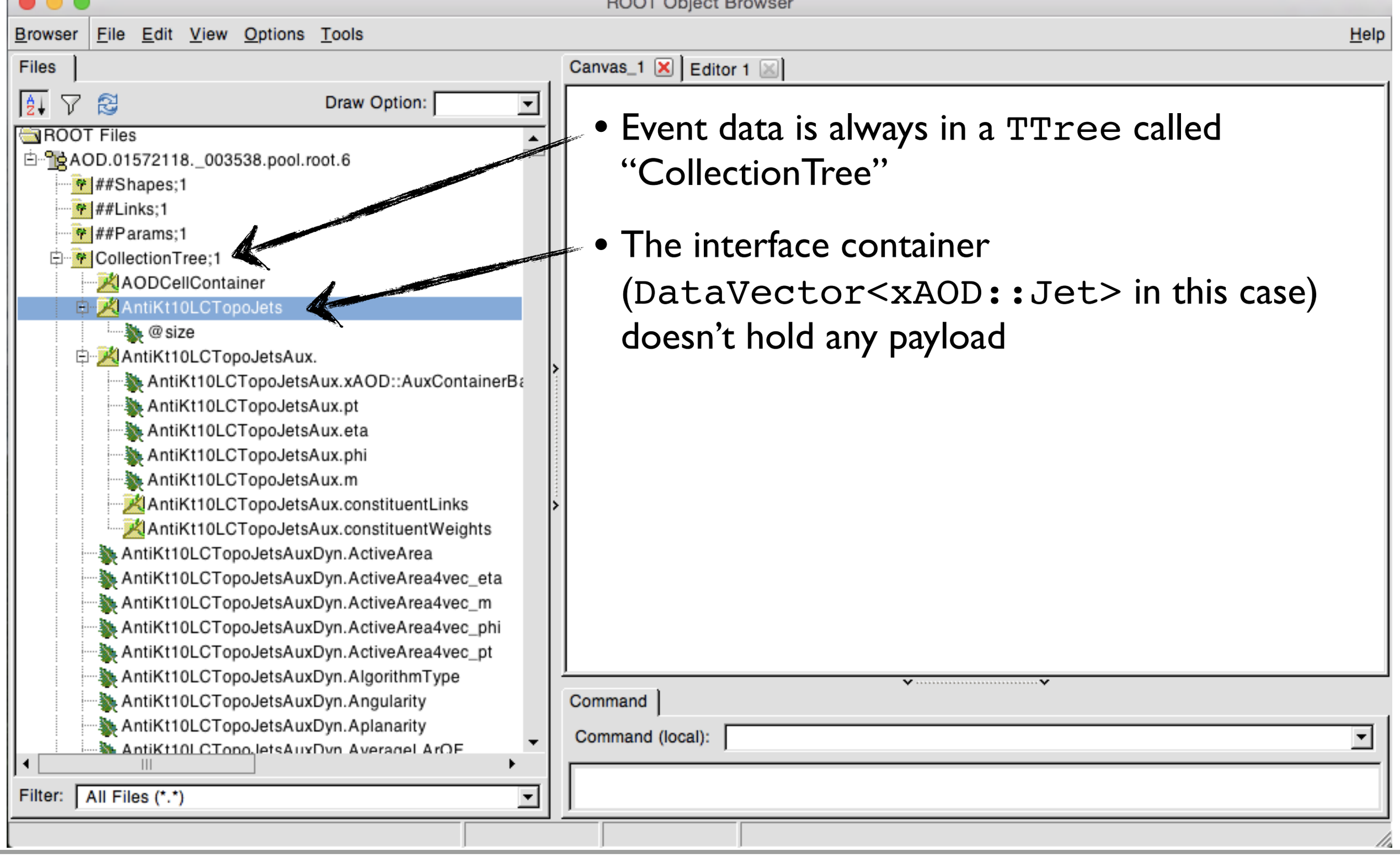
Command (local):

xAOD: How does it look like?



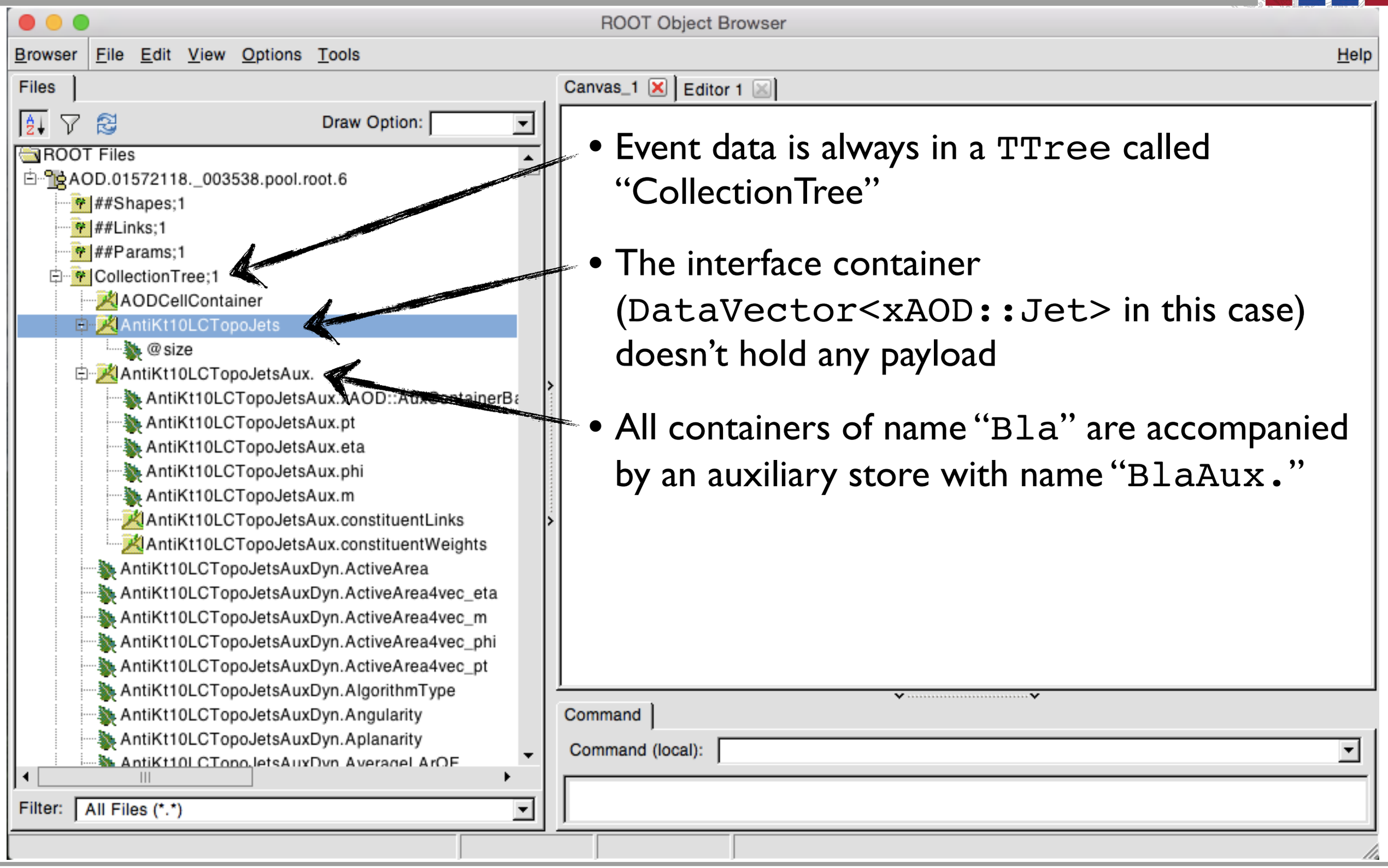
- Event data is always in a `TTree` called "CollectionTree"

xAOD: How does it look like?



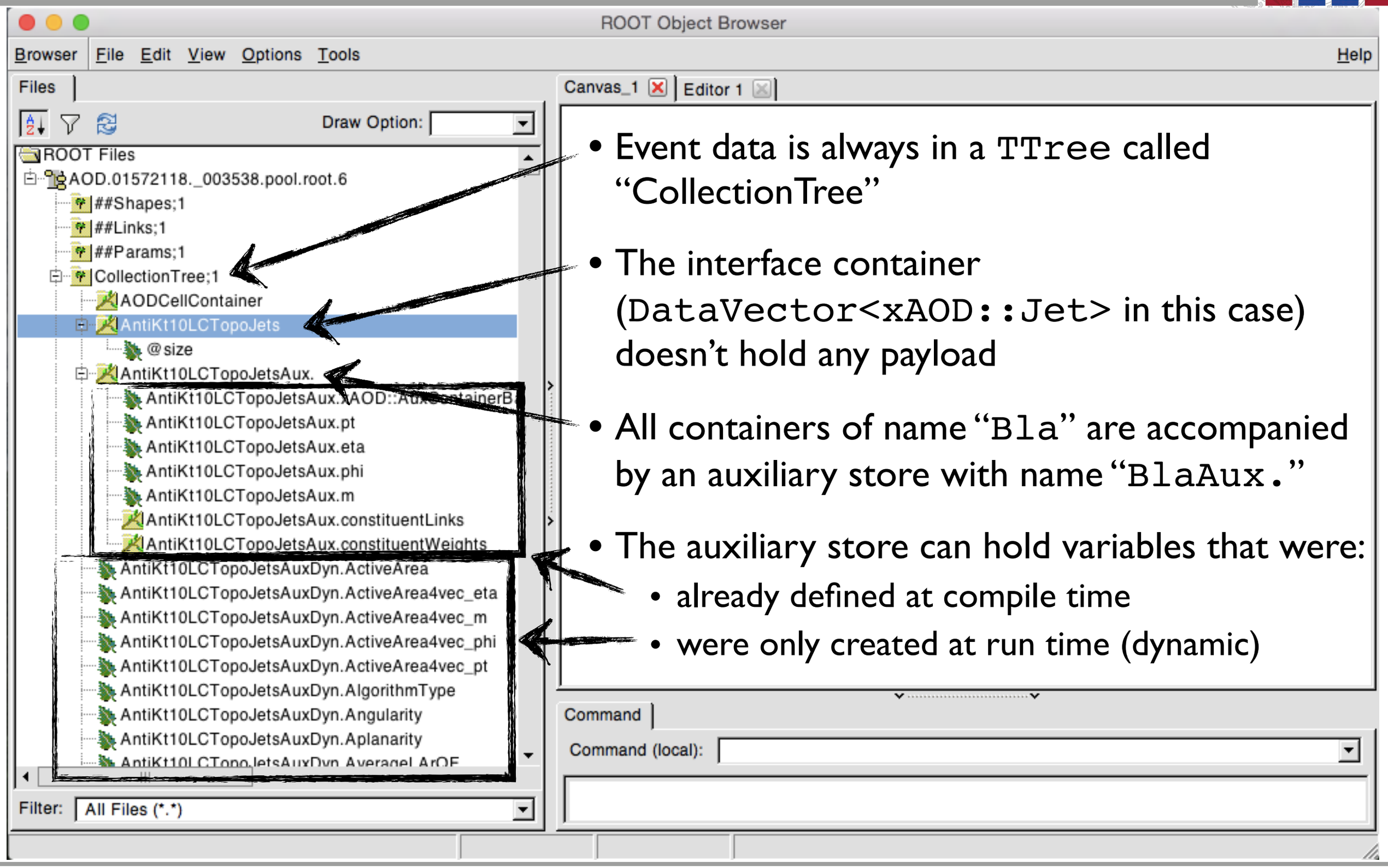
- Event data is always in a `TTree` called "CollectionTree"
- The interface container (`DataVector<xAOD::Jet>` in this case) doesn't hold any payload

xAOD: How does it look like?



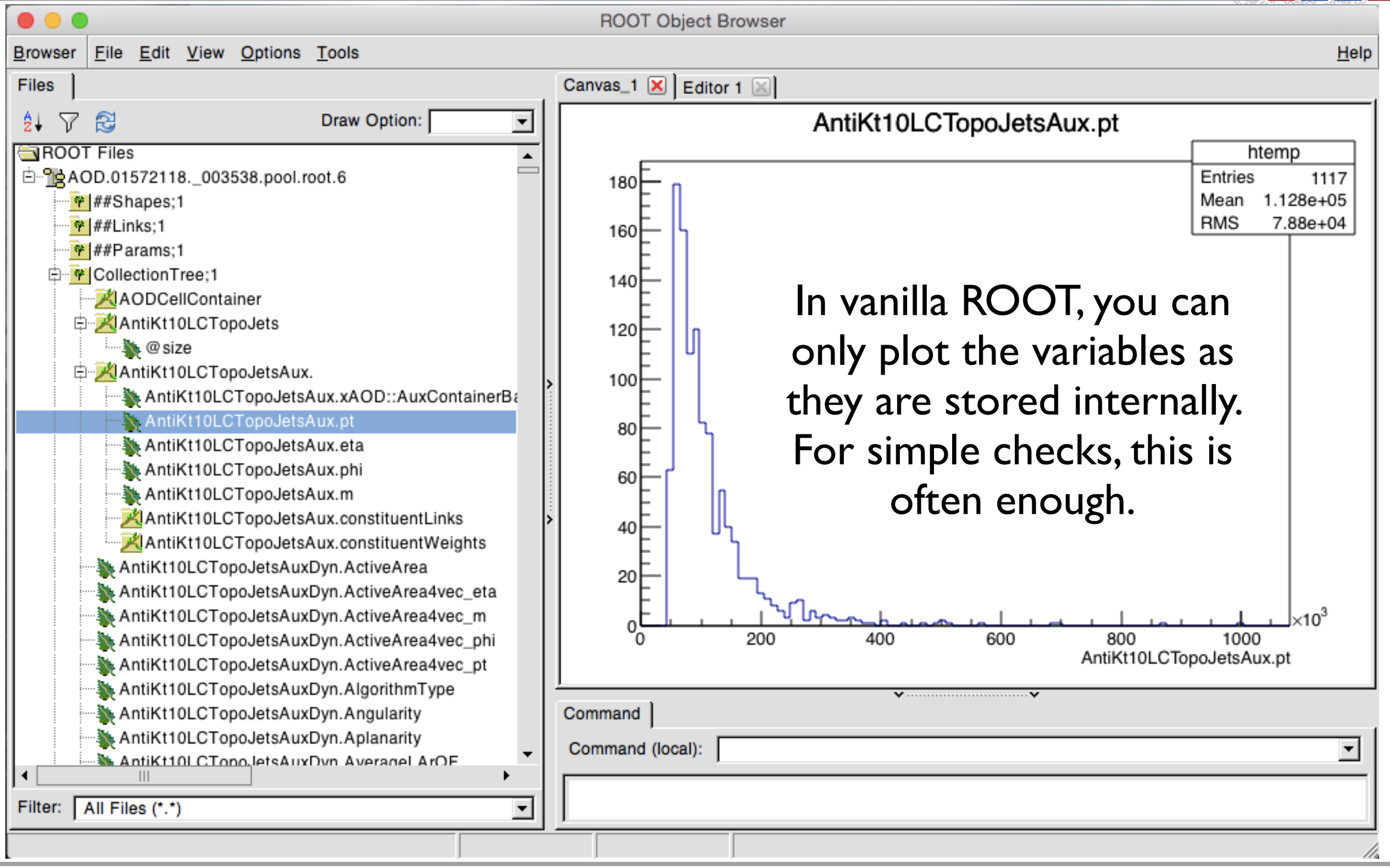
- Event data is always in a `TTree` called "CollectionTree"
- The interface container (`DataVector<xAOD::Jet>` in this case) doesn't hold any payload
- All containers of name "`Bla`" are accompanied by an auxiliary store with name "`BlaAux.`"

xAOD: How does it look like?

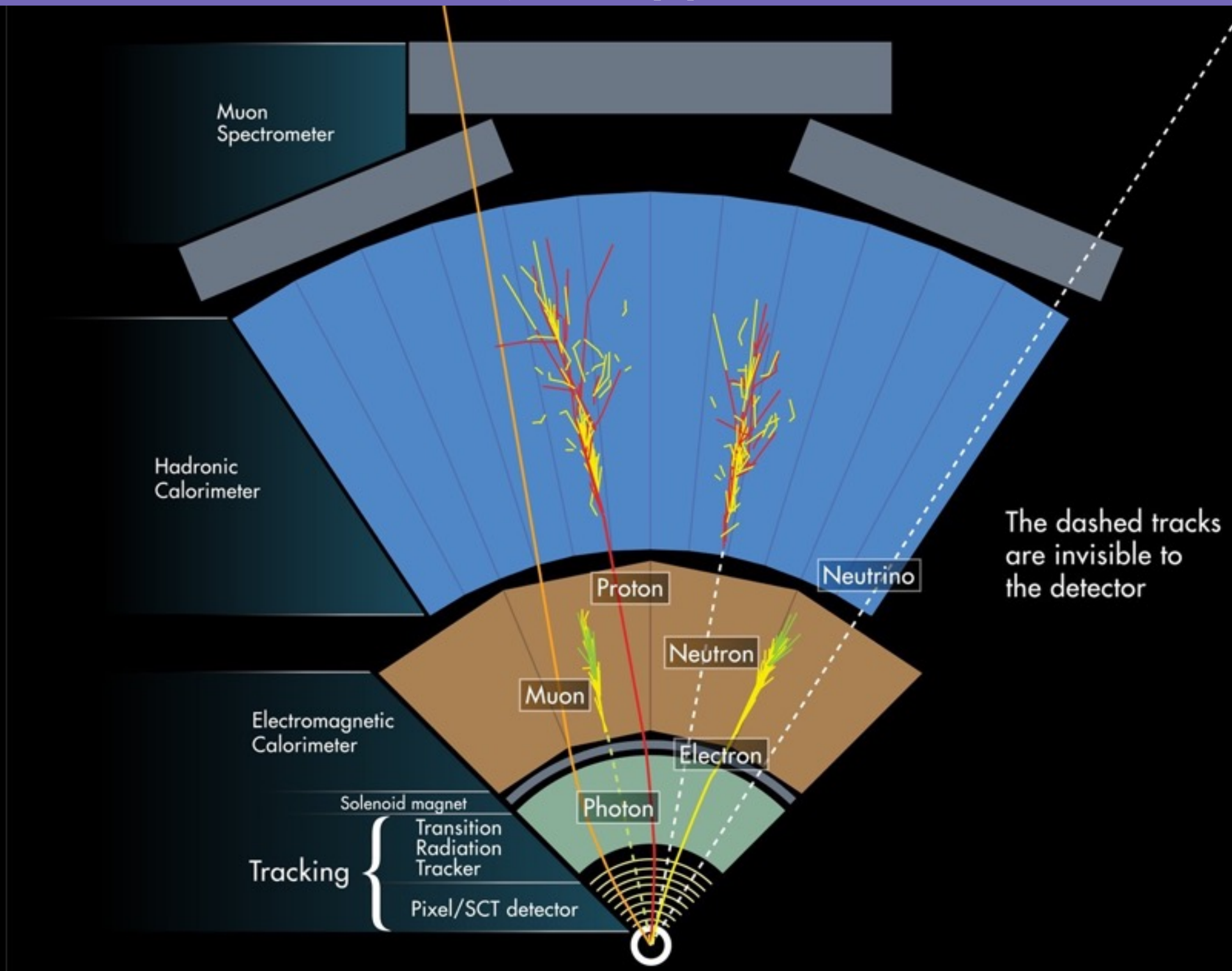


- Event data is always in a `TTree` called "CollectionTree"
- The interface container (`DataVector<xAOD::Jet>` in this case) doesn't hold any payload
- All containers of name "`Bla`" are accompanied by an auxiliary store with name "`BlaAux.`"
- The auxiliary store can hold variables that were:
  - already defined at compile time
  - were only created at run time (dynamic)

xAOD: How does it look like?



What are the main object types in the xAOD?



What are the main object types in the xAOD?

- Inner detector tracks and vertices
  - https://indico.cern.ch/event/403126/session/0/contribution/7/attachments/1155044/1659890/TrackingTutorial.pdf
- Electrons and photons
  - https://indico.cern.ch/event/379324/session/0/contribution/3/attachments/756751/1038105/BL_eGamma_softwareTutorial_2015_05_20.pdf
- Muons
  - https://indico.cern.ch/event/403126/session/0/contribution/10/attachments/1155109/1660042/MuonTutorial_16092015.pdf
- Jets and flavour tagging
  - https://indico.cern.ch/event/403126/session/0/contribution/11/attachments/1155022/1660126/ATLASSoftwareTutorial_Jets.pdf
- Taus
  - https://indico.cern.ch/event/403126/session/0/contribution/13/attachments/1155169/1660131/tau-software-tuturial-2015-09-16.pdf
- Missing energy
  - https://indico.cern.ch/event/403126/session/0/contribution/14/attachments/1155111/1660015/20150916_MET_EDM.pdf
- Monte Carlo Truth
  - https://indico.cern.ch/event/379324/session/0/contribution/7/attachments/756755/1038109/Marshall_Truth_2015.05.20.pdf

What software should I use to do my analysis?

- You should (must) use a framework
- Framework: software that enables your code to focus on physics analysis rather than "infrastructure"
  - ROOT is an analysis framework: you don't have to code up the graphics for drawing histograms before you start plotting
  - ROOT is the basis of our analysis model (since the xAODs are effectively ROOT files)
- You could do your analysis entirely in ROOT, just reading the branches from CollectionTree, without any extra code but it wouldn't be very convenient….

Essential features of an analysis framework

- **EDM**: library of C++ objects representing reconstructed objects
- **Invisible I/O**: you don't want to have to care about how the objects are written to and read from the file
- **Whiteboard**: means of storing data in memory for easy access during the analysis
- **Event loop**: built-in definition of an event, to enable event-by-event processing
- **Algorithm**: built-in definition for the main analysis program
- **Tools**: built-in definition for software that you call during your analysis program
- **Software management**: simple way of retrieving software from the central repository, modifying, compiling etc without worrying about make files
- ATLAS provides all of the above (and some people also wrote their own - but we do not recommend this)

Analysis frameworks in ATLAS

- Athena
  - This is the framework used for all pre-analysis data processing; it can also be used for physics analysis
  - It got a bad name during Run-I because, in order to read the old-style AODs, we had to use the full reconstruction set-up (RecExCommon)
  - With the xAODs we can use a much simpler set-up which uses the basic Athena components, but none of the complicated RecExCommon stuff → faster, easier
    - Some analyses *have* to be done in full RecExCommon mode because they are doing vertex-fitting etc, but we will not discuss this today
- EventLoop
  - Alternative framework written specifically for analysis; not used upstream of analysis
  - The most popular framework currently in use in ATLAS
  - Most of the group-specific frameworks are based on EventLoop

Analysis frameworks in ATLAS

Athena	EventLoop
Run-time configuration possible via Python “job options”	All run-time configuration done via C+
Tools can be scheduled from the job options, so no need to add them directly to your C++	Tools must be explicitly set up in your C++
Used for all pre-analysis data processing (generation, simulation, reconstruction, derivation, HLT)	Only used for analysis
Framework base (Gaudi) shared with LHCb	ATLAS-only
Independent of ROOT	ROOT-based
Managed with CMT	Managed with RootCore

- Which people use is now largely a matter of personal choice - BUT those who use Athena for analysis have a much easier life when they have to work on reconstruction/simulation/derivation etc
- You can use either during the tutorial

Analysis releases

- Software release = precompiled snapshot of the software, relevant to a certain task
  - we have releases for Tier-0 reconstruction, MC production, derivations etc
  - we also have releases for analysis, which contain all of the relevant tools
- The key analysis release is called *AnalysisBase*
  - Contains all of the combined performance tools from the CP groups *plus* the EventLoop components
- In addition we also build *AthAnalysisBase*
  - This contains the same CP tools as AnalysisBase, but instead of the EventLoop components, it has the Athena components instead
- So if you want to do your analysis with EventLoop, you should set up AnalysisBase. If you want to use Athena, set up AthAnalysisBase.
- Some physics group also make their own releases, built on top of AnalysisBase or AthAnalysisBase, with their own special tools included

Where should I do my analysis?

- This is an important question and largely depends on the size of your derivation and final selection
  - Groups with very selective derivations may be able to run the whole analysis on laptops
- The baseline answer:
  - Making plots with ROOT: laptop/desktop
  - Analysis of final selection: laptop/desktop or local cluster
  - Analysing derivations: local cluster or grid
  - Analysing AOD*: grid

- These days, with CVMFS, it is very easy to set up a Linux machine to run the ATLAS software. Bit harder with Mac. Probably not possible with Windows
- LXPLUS can always be used instead of a local cluster but it is sometimes slow
- ROOT runs easily on Linux/Mac, not Windows
  - If you want to use your laptop to do work, get rid of Windows

*You should rarely/never need to run on AOD

Overview of the tutorial

- TODAY
  - Set-up
  - Browsing the xAOD in ROOT
  - Browsing the xAOD code
  - Running analysis code
  - How to find data/MC: metadata tools (RunQuery, COMA, AMI)
- TOMORROW
  - MC truth analysis
  - Metadata in analysis
  - Introduction to the Grid
  - Obtaining data from the Grid
  - Running grid jobs

Main links for the tutorial

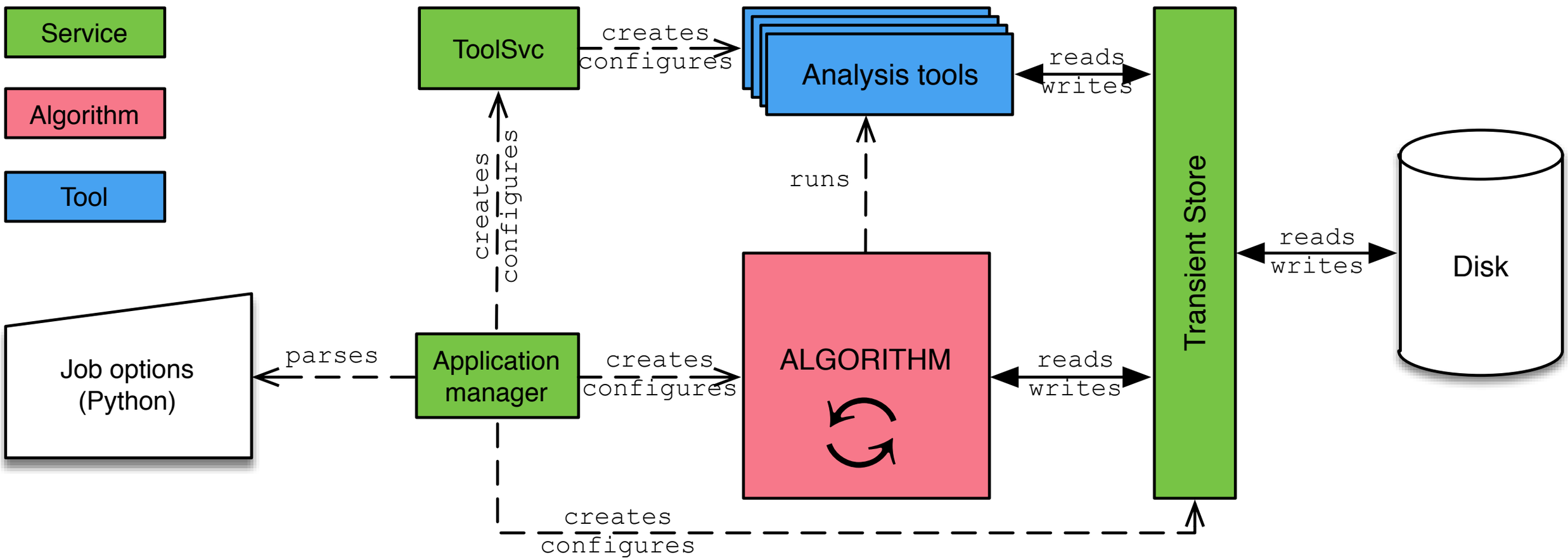
- The agenda: https://indico.cern.ch/event/472469/other-view?view=standard
- All of the exercises are drawn from the main ATLAS tutorial, found here:
  - https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/SoftwareTutorial
  - Agenda from the last CERN-based ATLAS tutorial: https://indico.cern.ch/event/465378/other-view?view=standard
- You will be working through parts of the above, but not all (there isn't enough time)
  - Obviously feel free to look at the parts that we don't cover this week

Backup

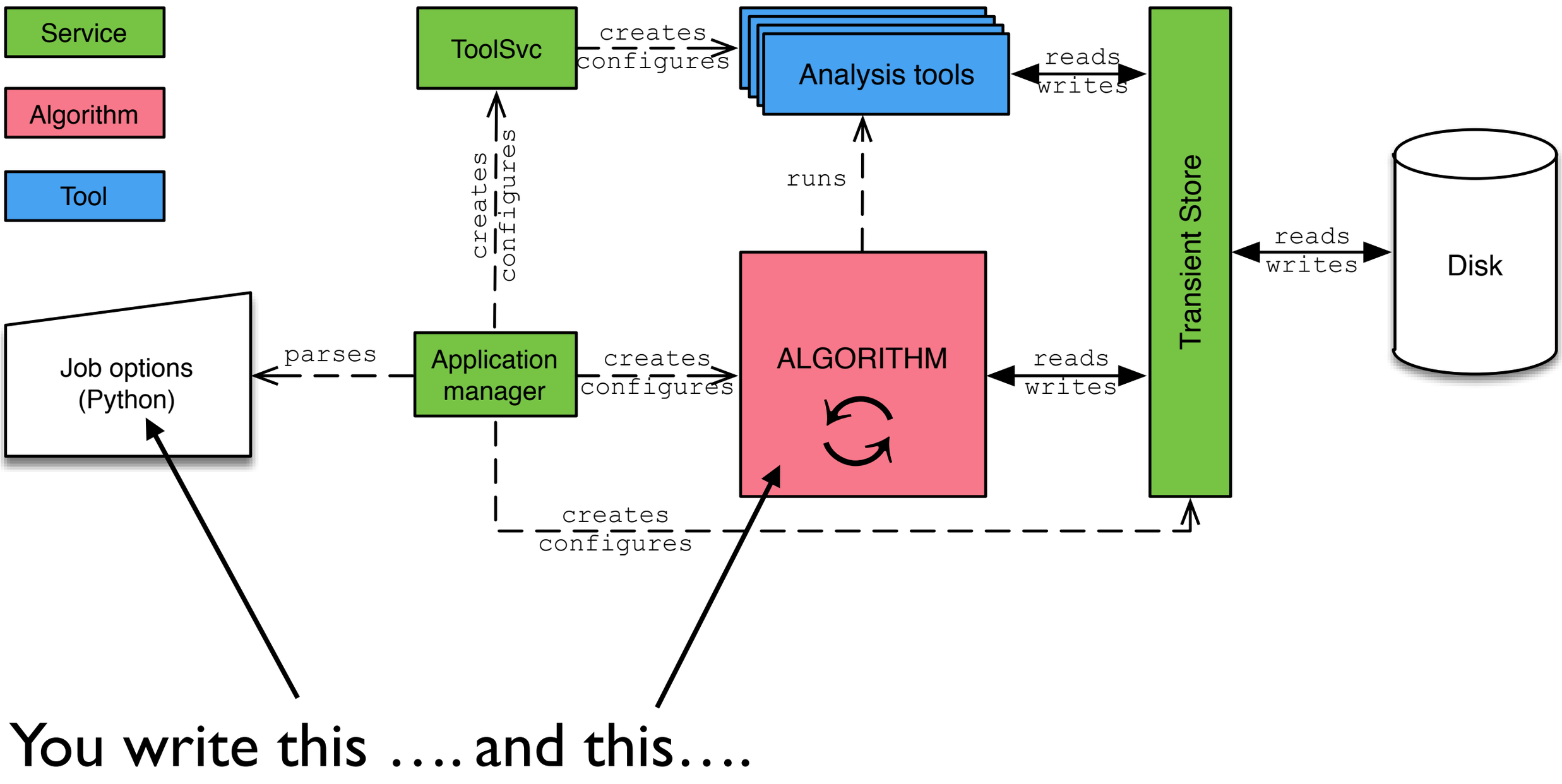
A bit more about Athena

- Built from three main types of component
- **Algorithm:** the main program defining the event processing loop
  - Consists of three methods
    - initialize() : once at start of job
    - execute() : once per event
    - finalise() : once at end of job
  - You can have as many algorithms as you need, running in sequence
- **Tool:** piece of code that can be executed as many times as you need in the execute() method of your algorithms
- **Service:** globally available entity, used across all algorithms and tools
  - Examples: message printing, configuring tools/algorithms, dealing with data storage etc
- Your role is to:
  - write the algorithm(s) needed for your analysis
  - configure the tools and the algorithm(s)
  - run the job!

A bit more about Athena



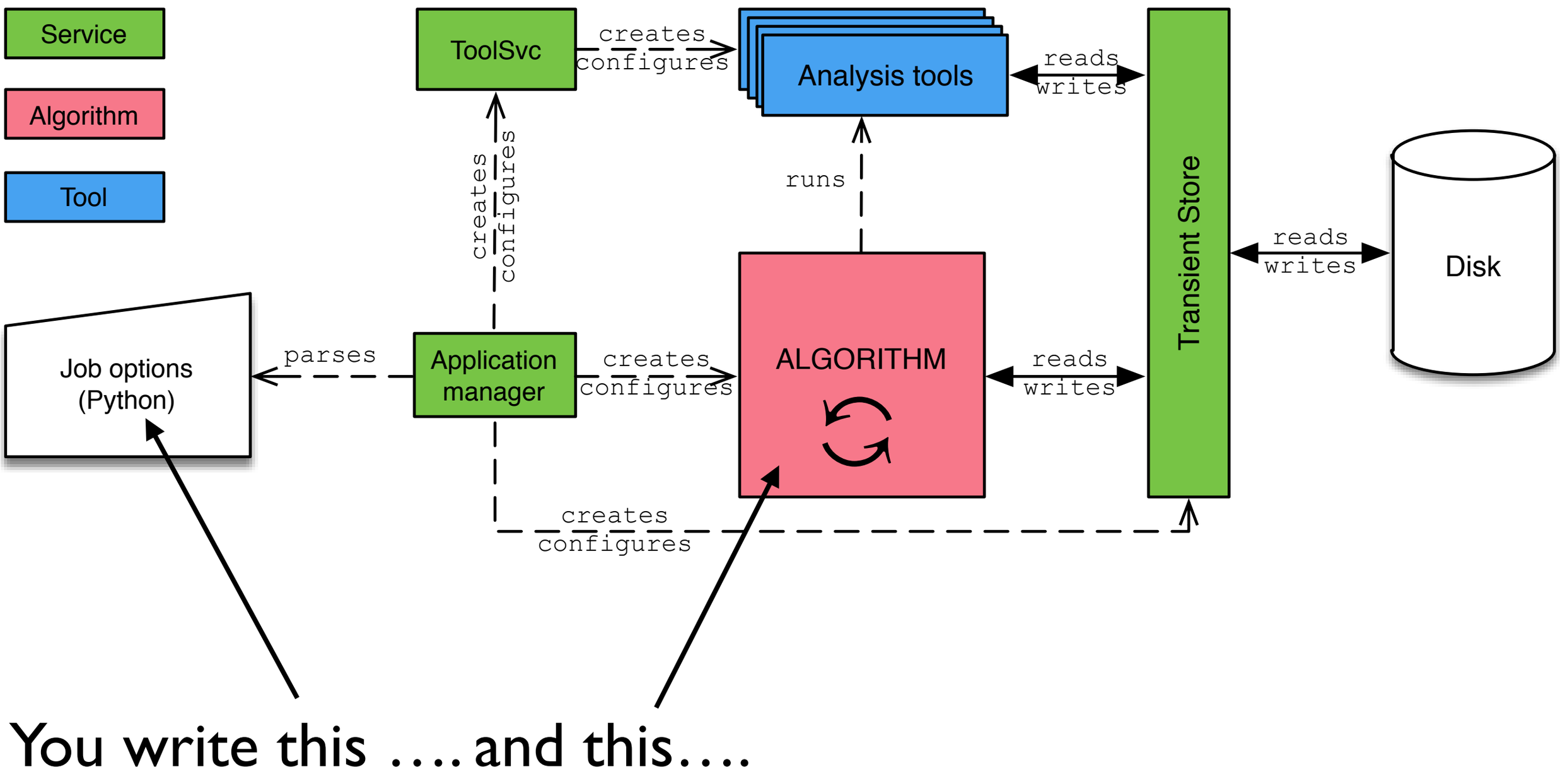
A bit more about Athena



You write this …. and this….

A bit more about Athena

and you need to know what is going on in here….



You write this …. and this….

A bit more about Athena