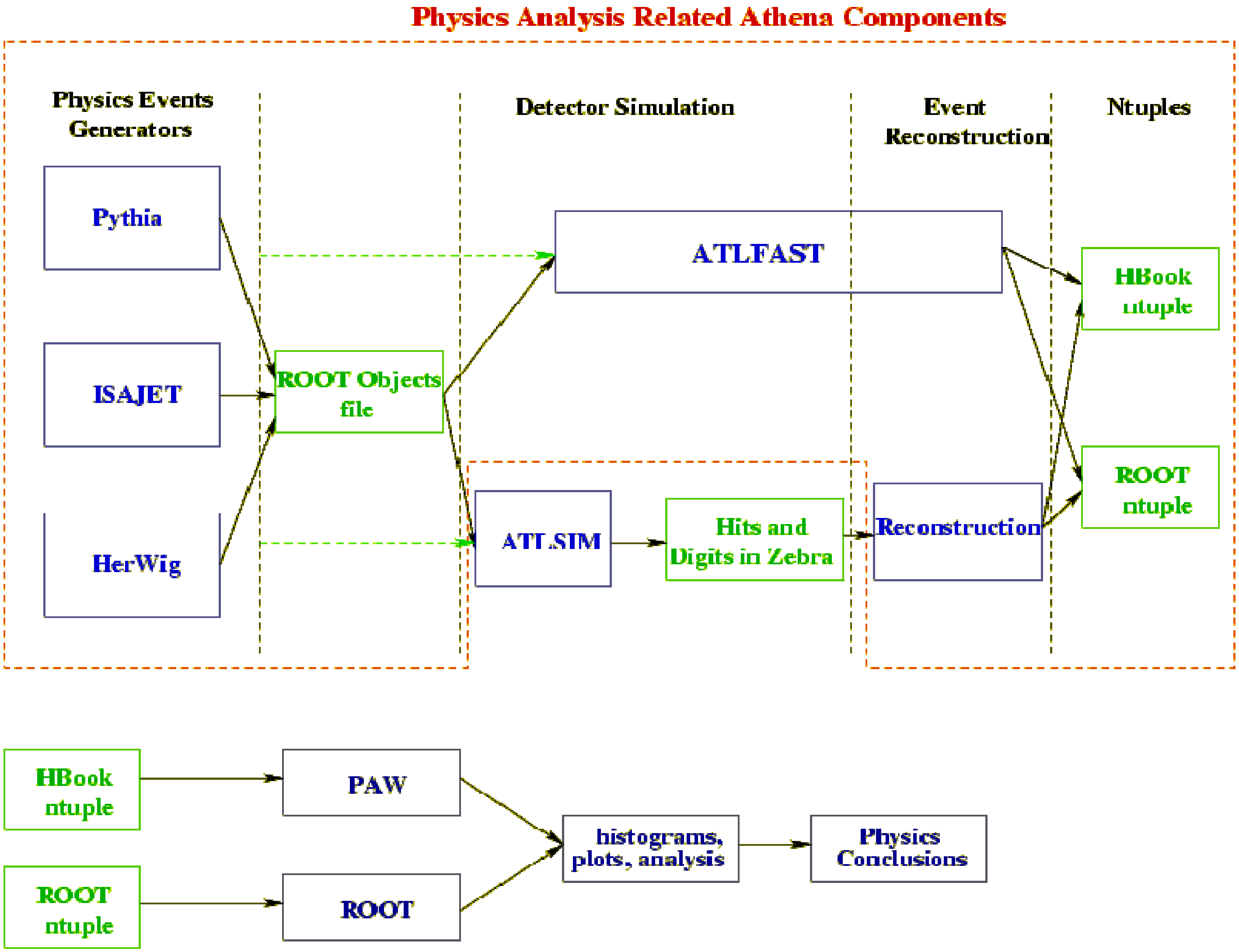
Overview of ATLAS Software

and the Athena Framework

Overview of the ATLAS Software

- General introduction to ATLAS software
- CVS, Packages, build tools and Releases
- Introduction to Athena Software
  - Algorithms, Tools, Services
  - Configuring jobs via jobOptions

Software Flow



A Brief History of Atlas Reconstruction

- Some algorithms development started more than 10 years ago!
- Pursued through the various detector Technical Design Reports till « Physics TDR » in 1999 (still the most relevant reference document)
- This was « atrecon », mostly fortran code in « slug », a zebra based framework
- Then migration to C++ then to Athena
- validation of Athena Reconstruction this year (data challenge 2, Athens Workshop) plus on-going development (Detector Description, Event Data Model, Reconstruction Task Force recommendations, new algorithms…)
- Plenty of things to do! Not least adapt the offline code to test-beam analysis.

CVS, CMT, Releases … ??

- All atlas offline code is stored in **CVS**, which manages the evolution of the source files
- The build of the binaries (compilation option, include files, libraries…) as well as the run-time environment are managed by **CMT**
- A new version of the code is entirely rebuilt approximately every 3 weeks (« developer release » e.g 8.7.0) with a « major release » approximately every 6 months.
  - Release 9.0.0 : October 27, 2004
  - Release 10.0.0 : 16 Feb 2005
  - Bug Fix releases (9.0.1, 9.0,2…) + incremental builds for test-beam
- Every night the release in construction is built (« nightlies ») and content kept for a week (useful only for developers of code in the release)

Packages
a way of grouping related code

- Typically, one package ↔one library which is dynamically loaded at run time. (there is only one very small athena executable for all applications)
- For Example, LArDigitization contains code for:
  - Taking a MC Hit and producing a Digit (5 samples).
  - The package has a structure:
    - LArDigitization/src : contains *.cxx files
    - LArDigitization/LArDigitization : *.h files
    - LArDigitization/share : jobOption files (no code)
    - LArDigitization/cmt : requirement file
- Packages may depend on other packages:
    - ✓ LArDigitization depends on LArIdentifier, LArRawEvent, etc.
    - ✓ Dependencies specified in requirement file
    - ✓ “depends” means in most cases “uses object defined in other packages” ⇔ “uses header file in other packages”
    - ✓ “depends” **does not mean**: need other packages to be run beforehand
- Dependency is uni-directional.
  - Packages at the bottom of the chain must be very robust.

Brief tour of the web/ documentation

- Mailing lists (sw-help, sw-reconstruction, sw-developers, atlas-larg-sw)
- Reconstruction web page

  http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/domains/Reconstruction/

- Following useful tools from software development web page:
  - Howto's ! A must-read
  - viewcvs/lxr ! Browsing the code in cvs
  - Doxygen ! Simple code documentation
  - Savannah ! Bug reporting system
  - release status and plans

Available algorithms (9.0.0)

(Some of these correspond to several Athena Algorithms)

- Truth interface
- xKalman++ (tracking)
- iPatrec (tracking)
- LArg Reconstruction
- Tile Reconstruction
- Muonbox, Moore (Muon reconstruction)
- Jet Reconstruction
- E/gamma identification
- Tau identification
- missing E_T
- Vertexing (primary vertex, interface of secondary vertices)
- Conversion
- Energy flow
- …

CBNT

- ❖ Each Reconstruction algorithm typically fills a block of the combined ntuple (CBNT). They provide an algorithm (e.g. CBNT_CaloCell) that is executed within Athena.
- ❖ Description of variables produced by the CBNT algorithms are at:
  - http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/domains/Reconstruction/packages/CBNT_Athena/CBNT_variables.htm
  - This allow easy checking of basic quantities
- ❖ However, the recommended model is not to write long kumacs to analyse the ntuples, but rather to do the analysis in the Athena (i.e look for Z candidates and fill ntuple with Z variables)
  - We have the ability to write/read all objects into POOL (persistency)
    - ✓ ESD and AOD streams contain reconstruction output
    - ✓ → equivalent of data in CBNT
    - ✓ Write analysis algorithms that produce your speacialized ntuples
  - Additional tools: Interactive Athena, PYROOT for analysis coming online

Documentation

- As with anything, it is the slowest to progress
- Where is the list of reco algorithms available ?
  - RecExCommon/share/RecExCommon_jobOptions is most likely to be up to date
  - Web page: Reconstruction→Reconstruction in Athena (being updated)
  - Something similar for RecExTB to be done
- Knowing the algorithm name how can i have information on the algorithm, and on its output
  - Same web page
  - Find the code: viewCVS, LXR (very useful)
  - Doxygen generated class diagram in <package>/doc/Doxygen/html/ :
  - CBNT_XXX algorithms to fill combined ntuple are usually good examples to see how to use object XXX

The Athena Framework

Athena as a Framework

- ❖ Framework Definition [1,2]
  - ✓ Architectural pattern that codifies a particular domain. It provides the suitable knobs, slots and tabs that permit clients to use and adapt to specific applications within a given range of behavior.

- ❖ In practice
  - ✓ A skeleton of an application into which developers plug in their code and provides most of the common functionality and communications among different components.

[1] G. Booch, “Object Solutions”, Addison-Wesley 1996

[2] E. Gamma, et al., “Design Patterns”, Addison-Wesley 1995

Or simply put, a Framework is:

- The software that makes sure your code
  - Runs at the right time
  - With the right input data
  - And takes care of any output

Athena Terminology [1]

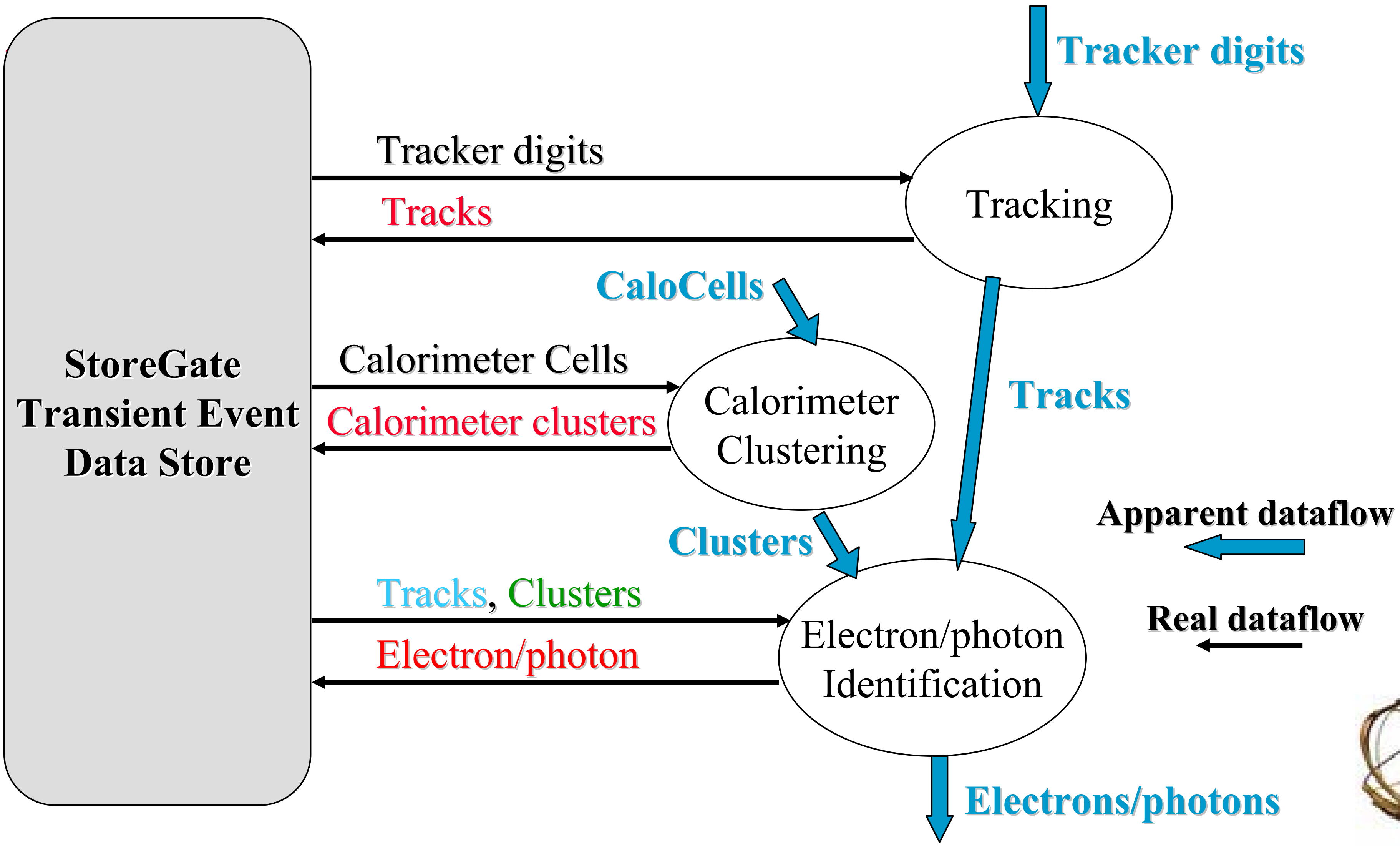
❖ Algorithm:

- User application building block, visible & controlled by framework.
- May delegate processing to AlgTools
- inherits from Algorithm class
- Implements three methods for invocation by framework :
  - ✓initialize(), execute(), finalize()

❖ Data Object:

- The result of your algorithm that is posted publicly and can serve as an input to a subsequent algorithm.
  - ✓e.g., Collection containing Cluster Objects
- Data Objects managed by a Transient Store : aka StoreGate
- Many different type of stores:
  - ✓Event Store, Detector Store

Algorithm & Data Flow



Tracker digits

StoreGate Transient Event Data Store

Tracker digits

Tracks

Tracking

CaloCells

Calorimeter Cells

Calorimeter clusters

Calorimeter Clustering

Tracks

Apparent dataflow

Clusters

Tracks, Clusters

Electron/photon

Electron/photon Identification

Real dataflow

Electrons/photons

ESD, AOD, … streams

- At the end of each event, the reconstruction output is written into several streams:
  - ESD = Event Summary Data
  - Contains intermediate reconstruction output such as
    - ✓ Tracks, Calorimeter cells, Calorimeter clusters
    - ✓ egamma, jets, muons, Missing ET, …
  - AOD = Analysis Object Data
  - Container particle level information
    - ✓ electron, photons, b-jets, muons, MissingET, …
    - ✓ Which allows you to do basic analysis with the ability to navigate back to parent objects if needed
  - For Test-Beam, expect only ESD stream to be written out.
  - Ntuples?
    - ✓ Will be written out to monitor data, but clients should expect to do their analysis off ESD and make their own specialized ntuples.

Athena Terminology [2]

❖ Services

- Globally available software components providing specific framework capabilities, e.g., Message service, Histogram service, etc

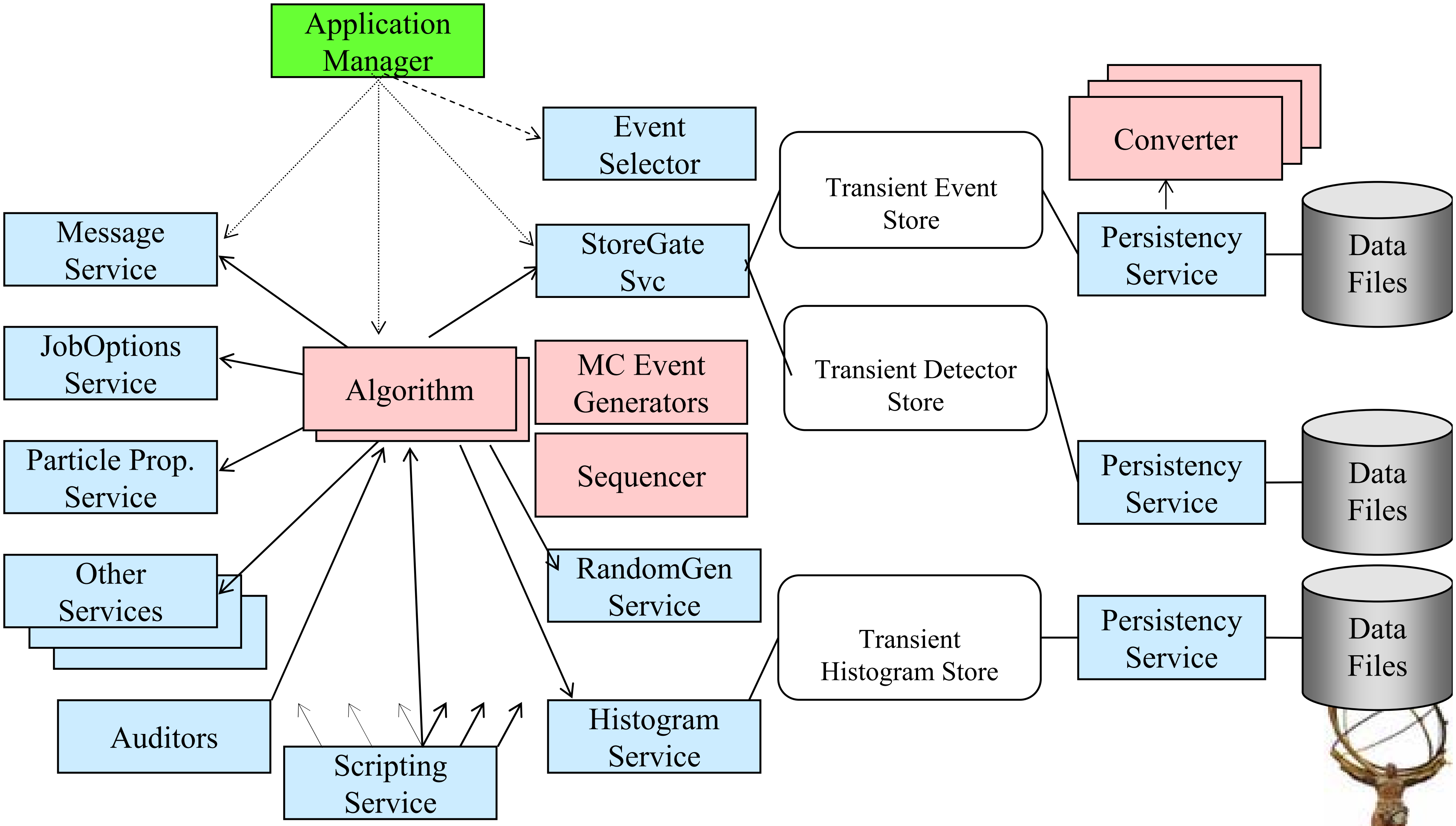
❖ Data Converters

- Provides explicit/implicit conversion from/to persistent data format to/from transient data
- Decouple Algorithm code from underlying persistency mechanism(s)

❖ Properties

- Control and data parameters for Algorithms and Services. Allow for run-time configuration. Specified via startup text files (jobOptions), Python scripts or from the scripting language shell

Athena-Gaudi Object Diagram



Athena Terminology [3]

- Job Options files
  - Conventional python scripts (default jobOptions.py) used to control an Athena application configuration at run-time
- Auditors
  - Monitor various aspects of other framework components
    - ✓NameAuditor, ChronoAuditor, MemoryAuditor, MemStatAuditor, etc
- Sequences
  - Lists of *members* Algorithms managed by a Sequencer.
  - Sequences can be nested. Default behavior: the Sequencer terminates a sequence when an event fails a filter. The *StopOverride* property overrides the default behavior.
- Filters
  - Event selection criteria.

Accessing Services

- Within the Algorithm, services are readily accessible.

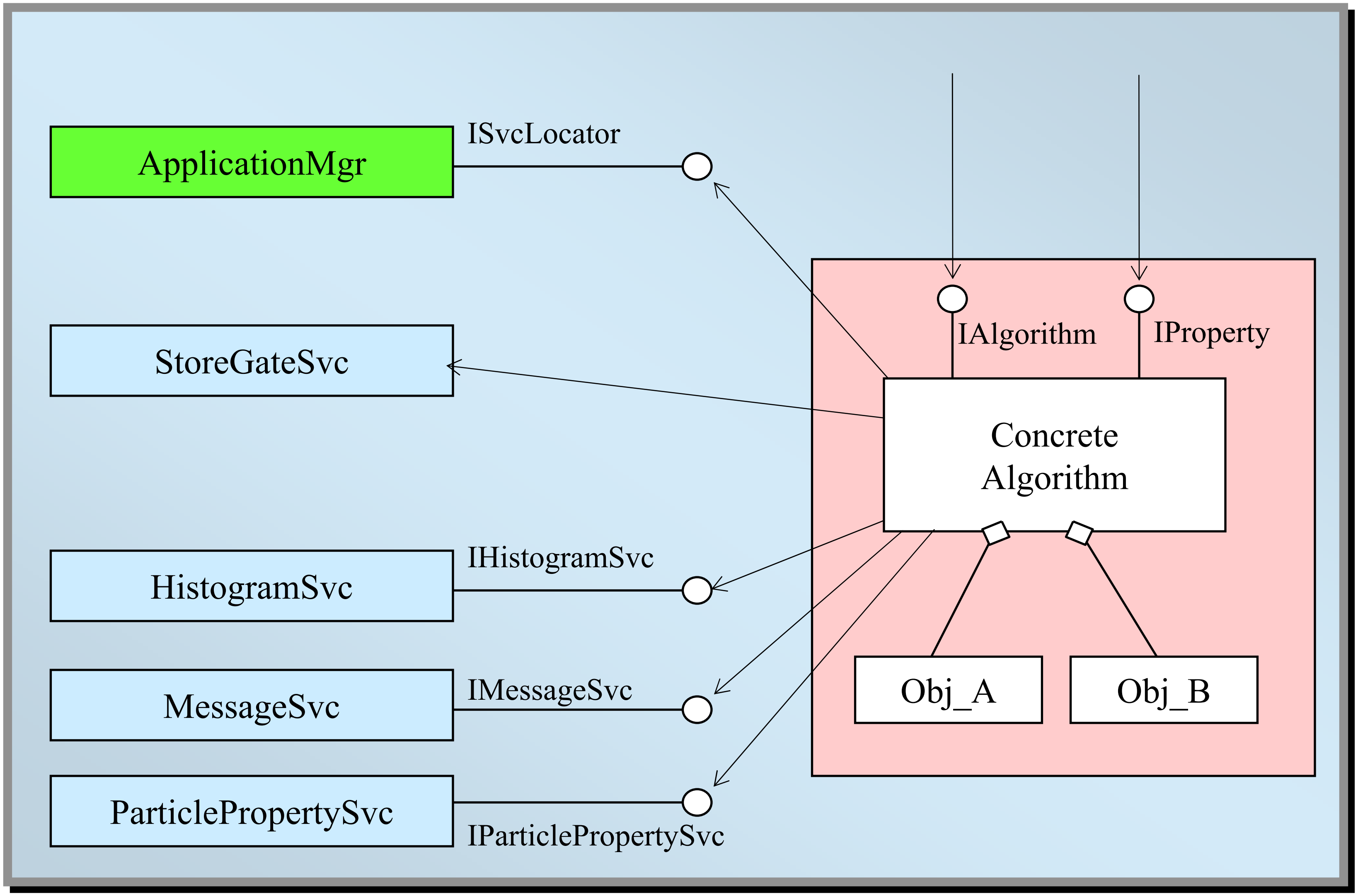
- Access to some of the "basic" services are pre-defined
  - msgSvc( )
  - histoSvc( )
  - ntupleSvc( )

- Access to most other services must be located:

  StatusCode sc = service("StoreGateSvc", m_eventStore);

  - Retrieves the pointer to StoreGate Service
  - Then you use m_eventStore in your algorithm to access data objects
  - Typically done in initialize() of your algorithm and cached

Accessing Services



JobOptions

- The jobOptions file specifies the run-time configuration of your algorithm
  - Specifies all the services needed and their configuration
  - Determines what algorithms need to be run
  - In what order
  - Specifies the properties of the algorithm
  - Control over :
    - ✓ Message Output Level
    - ✓ Number of Events to process
    - ✓ Input file names.
    - ✓ Output file names, objects to save etc.
    - ✓ And so on.

Some Common Entries in jobOptions.py

- Including other python scripts:

```
include ( "RecExCommon/RecExCommon_flags.py" )
include( "CaloRec/CaloCluster_jobOptions.py" )
```

- Component libraries to be loaded

```
theApp.DLLs += [<comma separated array of string>]
e.g.  theApp.DLLs += ["CaloRec", "LArClusterRec","TileRecAlgs" ]
```

- Top level algorithms: “Type/ObjectName”

```
theApp.TopAlg += [<comma separated Array of string>]
e.g.  theApp.TopAlg += ["CaloClusterMaker/CaloSWClusterMaker" ]
```

- Maximum number of events to execute

```
theApp.EvtMax        =        100
RunNumber            =        200100
```

→ internally, theApp.RunNumber = RunNumber (global flag)

- Comments

Preceded by #

Configuring an Algorithm

- If you have an Athena algorithm MyAlg, you need to specify the following in the jobOptions:
- theApp.DLLs += [“MyAlgLib”]
  - Name of the library where MyAlg is located
  - This is typically the package name
- theApp.TopAlg += [“MyAlg/MyAlgName”]
  - MyAlg is your Algorithm class name
  - MyAlgName is any name you give it
  - Algorithms are run in the sequence they appear in the jobOptions
- Specify property of the algorithm

  MyAlgName = Algorithm(“MyAlgName”)

  MyAlgName.someProperty = property_value

Global Flags

- There are many flags that are set to a default value in the jobOptions and can be over-written to suit your needs:
  - These flags are used to control the execution process
- Major Flags:
  - theApp.EvtMax = 100 # maximum number of events
  - RunNumber = 201001 # run number
  - doWriteESD = False # do not write ESD output
- Flags to control which sub-system reconstruction to run:
  - doInDet = False
  - doLar = True
  - doTile = True
  - …
- + Sub-System specific flags

Running athena

- athena
  - By default, it looks for jobOptions.py
- athena MyJobOptions.py
  - Athena will use MyJobOptions.py for input configuration
- athena MyJobOptions.py >& athena.log
  - Directs all screen printout to athena.log
- athena –s MyJobOptions.py >& athena.log
  - -s prints out all jobOptions scripts that are included within
- athena –c “EvtMax=10; doWriteESD=True” >& athena.log
  - Uses the options to overwrite default ones in jobOptions.py
  - Only works today with RecExCommon
- athena – i MyJobOptions.py
  - Interactive mode of running athena

Standard Top Level JobOptions

- RecExCommon_topOptions.py
  - Use to run full ATLAS Reconstruction on G3 or G4 simulation
  - Outputs ESD and AOD data and minimal ntuples
- RecExTB_Combined_2004_jopOptions.py
  - Use to run reconstruction software on TB data and TB simulation
  - Outputs ESD data
- TBAnalysis_topOptions.py
  - Used to analyze the output ESD data from RecExTB
  - Clients can plug in their own analysis algorithms
  - Outputs user ntuples
- Other top level jobOptions available for :
  - ATLAS and TB : Simulation & Digitization
  - That output POOL files with relevant data

Today

❖ You will learn how to :

- Setup the software environment and compile your code
- Familiarize yourself how to configure your code with jobOptions
- use RecExTB to reconstruct test-beam data:
  - ✓Produce ESD
  - ✓And look at ntuples
- Write an analysis algorithm in Athena with ESD data as input
  - ✓And produce your own analysis ntuples
- Write simple python based analysis scripts with ESD data as input
  - ✓Using Athena and ROOT features interactively