
Introduction to the Athena Software

Hong Ma

BNL

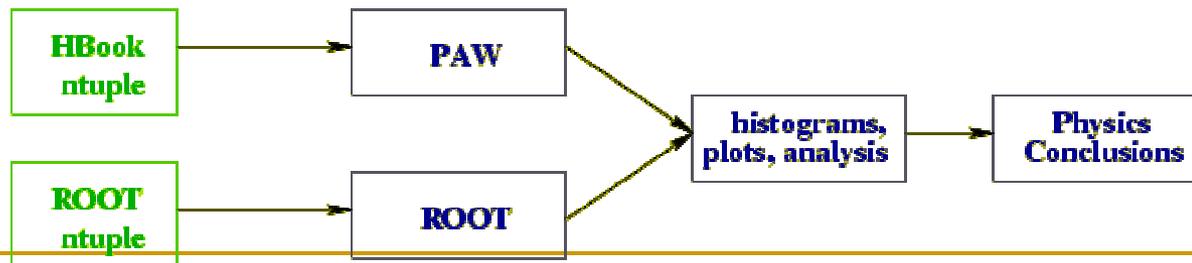
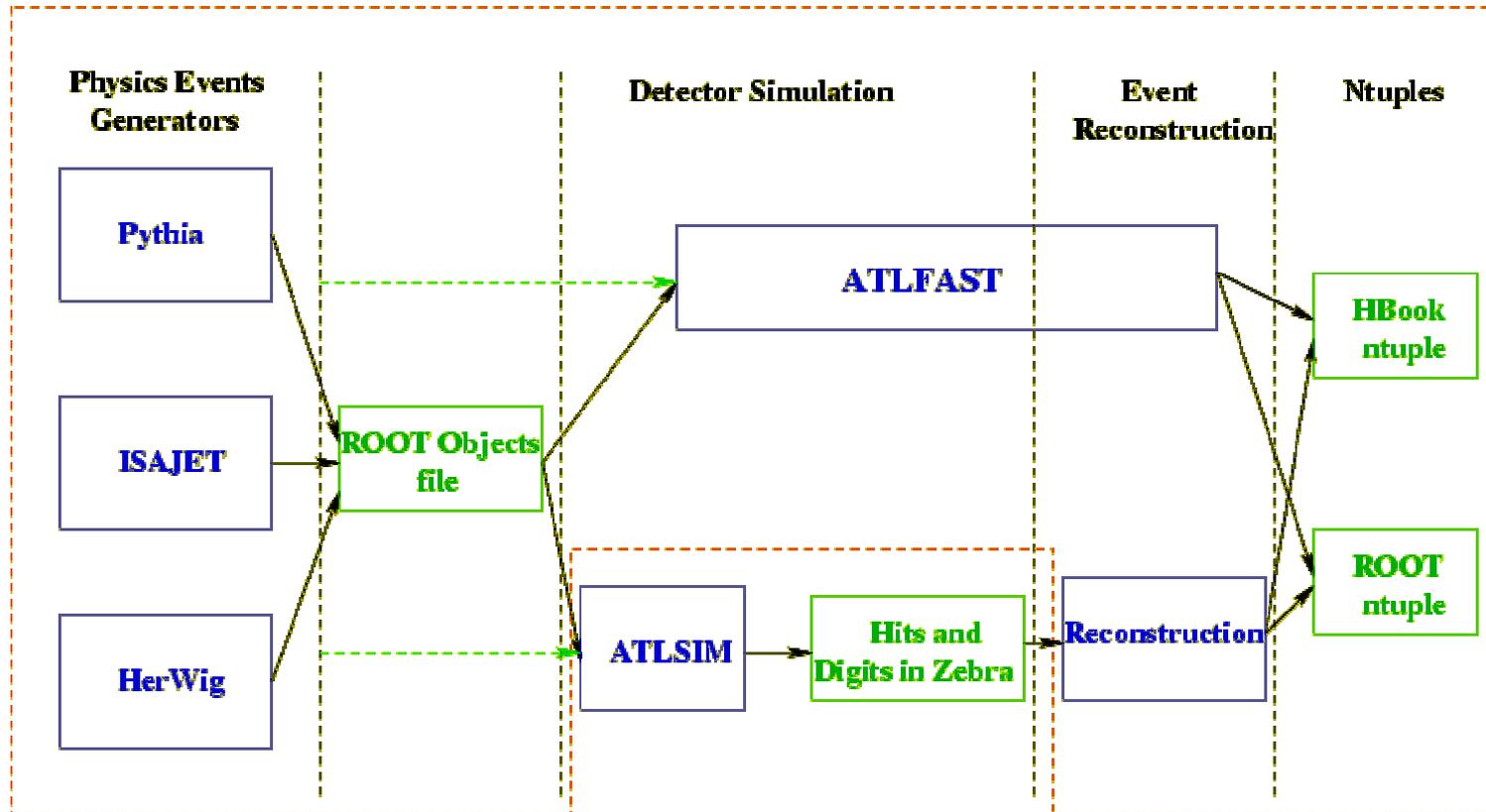
Athena Tutorial

USATLAS Software Workshop

Outline

- Introduction to components relevant to end-users
 - Athena concepts
 - Typical dataflow
 - Web/documentation tour
 - Reconstruction
- End-user interface:
 - Customize job configuration
 - Running Athena job (AthASK, next talk)
 - Using ntuple output
 - Analysis Examples at USATLAS
- Goal:
 - Understand the basic Athena concepts, availabilities of various simulation/reconstruction components, be ready to analyze the Athena ntuple output.

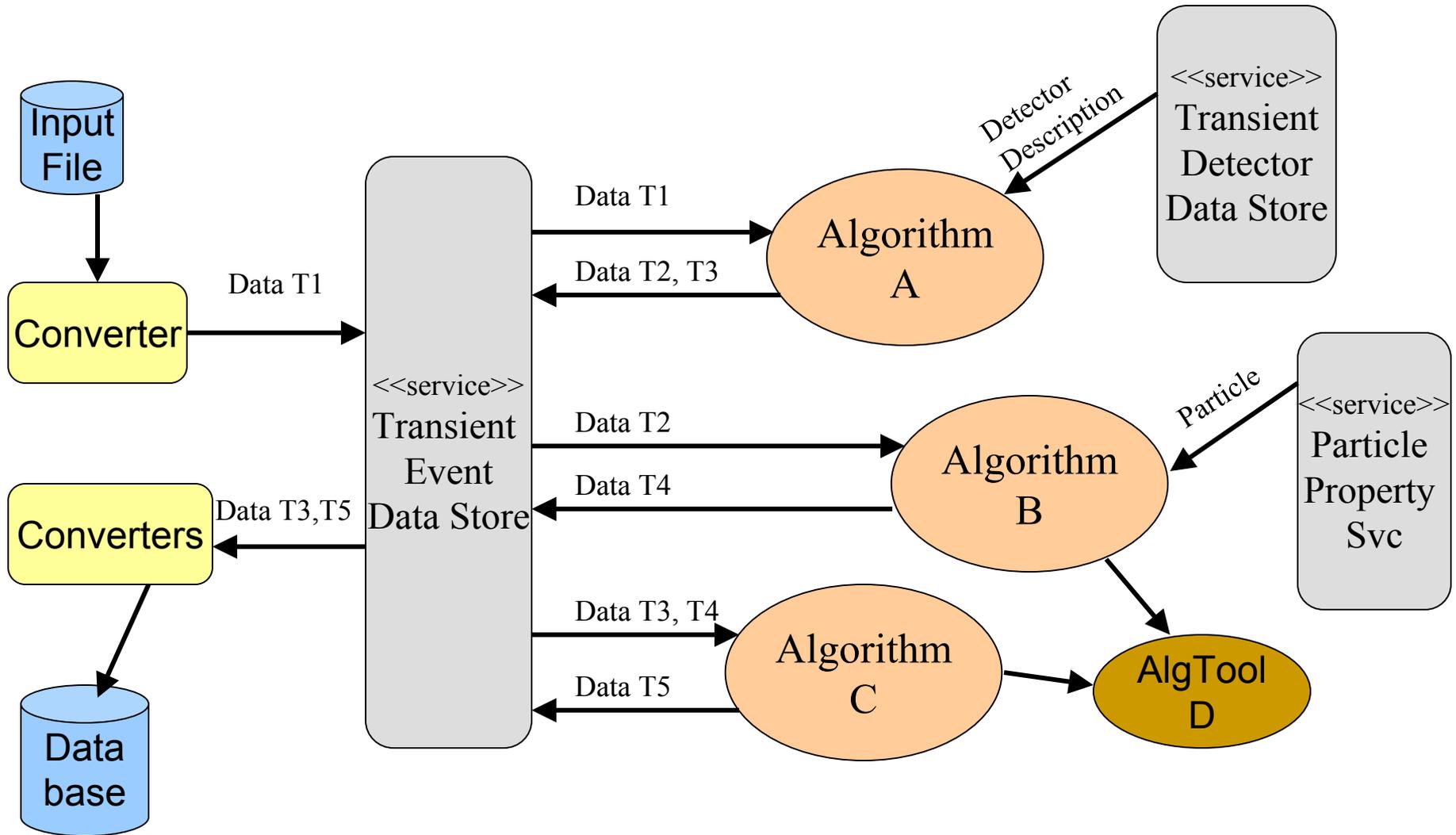
Physics Analysis Related Athena Components



Athena as a Framework

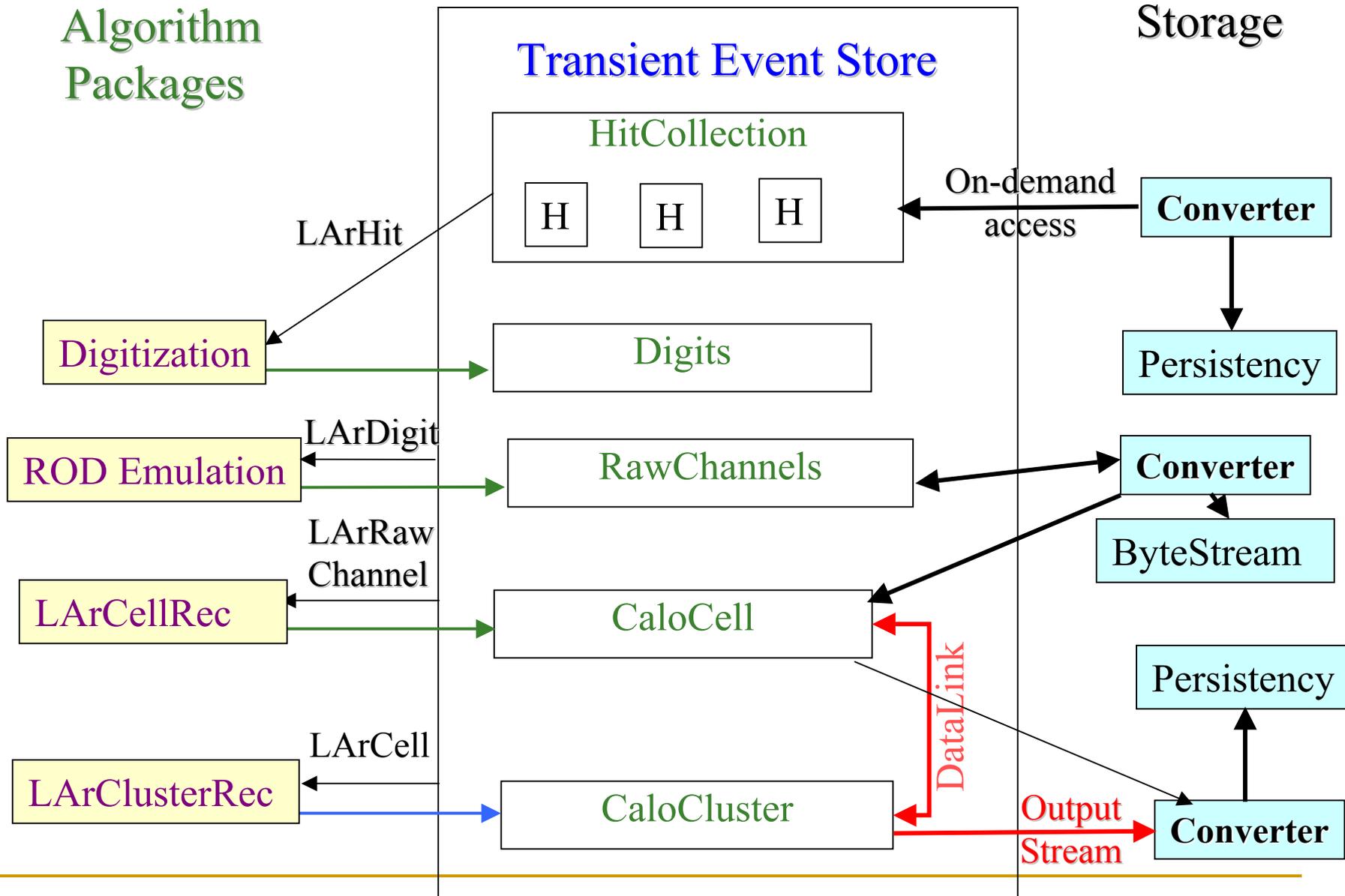
- A skeleton of an application into which developers plug in their code and provides most of the common functionality and communications among different components.
 - Gaudi is a common framework used by ATLAS and LHCb. Athena is built on top of Gaudi.
- Responsible for
 - Define interface for each component;
 - Loading shared libraries;
 - Instantiate components;
 - Event Loop;
 - IO mechanism;
 - Job configuration ...

Illustration of Components



LAr Event Data Model

Reconstruction / Data Flow



Packages and CVS

- Code is stored in packages, which are organized in hierarchical structures in CVS:
 - It maintains the history of each file.
- Access to CVS
 - Browse content using cvsweb server ([USATLAS](#))
 - Check out package from CVS server (>cvs co ...)
- Package structure:
 - [e.g. LArCellRec](#)
 - LArCellRec/ Header files (class definition)
 - src/ Source code (implementation of classes)
 - cmt/ cmt requirements file
 - share/ jobOptions, data files, etc.
 - ChangLog text file containing change history
- A specific version of the package is associated with a cvs tag.
 - e.g., LArCellRec-02-08-11 is a tag for LArCellRec in release 6.5.0
 - User can check out the package with a tag.

CMT

- **Configuration Management Tool**
 - Manages the relationship between the packages
 - Dependency between packages
 - Compile and build the packages
 - creates Makefile for you.
 - Setting up runtime environment
 - Env variables: PATH, LD_LIBRARY_PATH, etc.
 - ...
- **Hidden from user if AthASK is used**

Releases

- Release: A complete set of code compiled, built and frozen for use
 - Defined by a set of tags for each package,
 - Tag collector
 - Nightly builds
 - Used by developers to work towards a release.
 - Not guaranteed to work
 - Developer Release: approximately every 3 weeks (currently 6.6.0)
 - Mostly working
 - Major release: approximately every 6 months (currently
 - Major milestones, such as Data Challenges
- Release plan and Release Status can be found on the “Software Development” page
- USATLAS builds same releases as CERN
 - `/afs/usatlas.bnl.gov/software/dist/`

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 and on-going development (Detector Description, Event Data Model, new algorithms...)
- Plenty of things to do!

Reconstruction in Athena

- Each subsystem develops event data model and algorithms to produce the
 - e.g.: Inner Detector creates tracks with fitting algs
- High level reconstructed objects are based on objects from detector reconstruction.
 - e.g.: e- γ reconstruction combines InDet tracks with Calorimeter clusters
- Then physics objects
- Plan for the future development of reconstruction by the Reconstruction Task Force

Reconstruction Algorithms

(Some of these correspond to several Athena Algorithms and packages)

- MC Truth (Generators)
- InDet Reco (cluster, spactpoints) (InnerDetectors)
- xKalman++ (tracking) (Reconstruction)
- iPatrec (tracking) (Reconstruction)
- Calorimeter Reco (Cell, Clusters) (LArCalorimeter, TileCalorimeter)
- Muonbox (Muon reconstruction) (MuonSpectrometer)
- Moore (Muon reconstruction) (MuonSpectrometer)
- Jet (Reconstruction)
- e/gamma identification (Reconstruction)
- tau identification (Reconstruction)
- missing E_T (Reconstruction)
- Vertexing (Reconstruction)
- Conversion (Reconstruction)
- Energy flow (Reconstruction)
- MuonIdentification (Reconstruction)
- Atlfast (Simulation)

RecExCommon

- All the available reconstruction algorithms are run from package RecExCommon, (no code, jobOptions only).
- RecExCommon is systematically run to check a release is (not) working, and used in reconstruction data challenge too.
- Each reconstruction task fills a block of the combined ntuple (CBNT) with all the variables needed at analysis level
 - This allows easy access to basic quantities
 - Trade-off between writing algorithms to make final physics objects, or doing it with analysis tool (PAW or ROOT)
e.g., $Z \rightarrow ee$ Athena Algorithm or $Z \rightarrow ee$ ROOT macros
 - We do not have object persistency (besides ntuple) so full reconstruction chain needs to be rerun anytime the Algorithm is changed

Combined Ntuple (CBNT)

- Description of variables can be found in on reco [page](#)
 - Example: EM cluster in e-gamma block:

eg_nc: number of em clusters
eg_et: ET of cluster
eg_eta: eta of cluster
eg_phi: phi of cluster

eg_etap: eta calculated from pointing
eg_zvertex: z-vertex position
eg_errz: error on z-vertex
....

- The combined ntuple is built from many CBNT algorithms, each converts certain data object into CBNT entries
 - E.g; CBNT_egamma class is an algorithm in Reconstruction/egammaRec package

Running the job with jobOptions

- Job is essentially steered by a conventional text file (to be replaced by Python scripts after Release 7.0.0)

> athena jobOptions.txt

- Common entries in jobOptions.txt

- Example of Standard Configuration

```
#include "StandardAtlfastOptions.txt"
```

- Prefix with env variable if not in local area

```
#include " $ATHENACOMMONROOT/share/Atlas_Gen.UnixStandardJob.txt"
```

- Maximum number of events to execute

```
ApplicationMgr.EvtMax      = 100;
```

- Component shared libraries to be loaded

```
ApplicationMgr.DLLs += {<comma separated array of string>}
```

- Top level algorithms: "Type/Name"

```
ApplicationMgr.TopAlg += {<comma separated Array of string>}
```

- Comments

```
Preceded by //
```

More on jobOptions

- Algorithm and Service classes can declare “properties” that can be modified at runtime by jobOptions, e.g.

```
ApplicationMgr.DLLs += {"MyReco"};
```

```
ApplicationMgr.TopAlg += {"MyAlgorithm/MyAlg1"};
```

```
MyAlg1.Threshold=50.;
```

Data files

- ATLAS Data Challenges produce large data samples requested by physics groups.
- Locate input files using DC1 websites
 - Check DC1 [site](#) for data sample
 - Link to e/gamma, then to $Z \rightarrow ee$, dataset 002046
- Find the file with Magda
 - Use magda [site](#) for finding the logical filenames
 - Search for LFN substring with “002046.simul”
 - Find files like: `dc1.002046.simul.00001.hlt.Z_ee.zebra`
- Retrieve data with magda command
 - `> magda_getfile dc1.002046.simul.00001.hlt.Z_ee.zebra`
 - Symlink will be created in your directory

Guide for Physics Analysis at U.S.ATLAS

- A set of web pages that help end-users to start using ATLAS software for physics analysis
 - New User Guide for starting up at USATLAS
 - ATLAS in general, and USATLAS specific
 - Physics Analysis Guide for all the components
 - Supplement the existing ATLAS documentation
 - An example ($H \rightarrow ZZ^* \rightarrow e^+e^-e^+e^-$) through the analysis chain:
Generator \rightarrow FastSimulation/FullSimulation \rightarrow Reconstruction \rightarrow Analysis
 - Production and Analysis Tools
 - Feedbacks are very welcome.

More on Analysis Guide

- How to generate Pythia Events using Generator software
- Fast simulation with ATLFAST. Analyzing ATLFAST ntuple.
- Production tools
 - For running Full Detector Simulation (ATLSIM)
 - For running Athena jobs in batch, (Reconstruction)
 - How to process large amount of data (~100k events)
- Analysis examples for Atlfast and reconstruction output
 - e.g.: [Reconstruction/analysis of the DC1 SUSY sample](#)