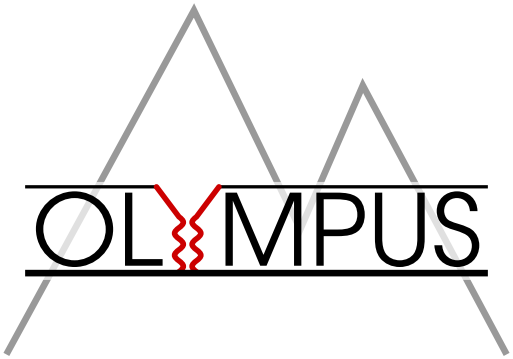
Data analysis

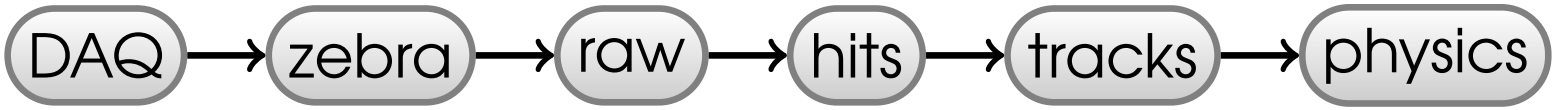
Jan C. Bernauer



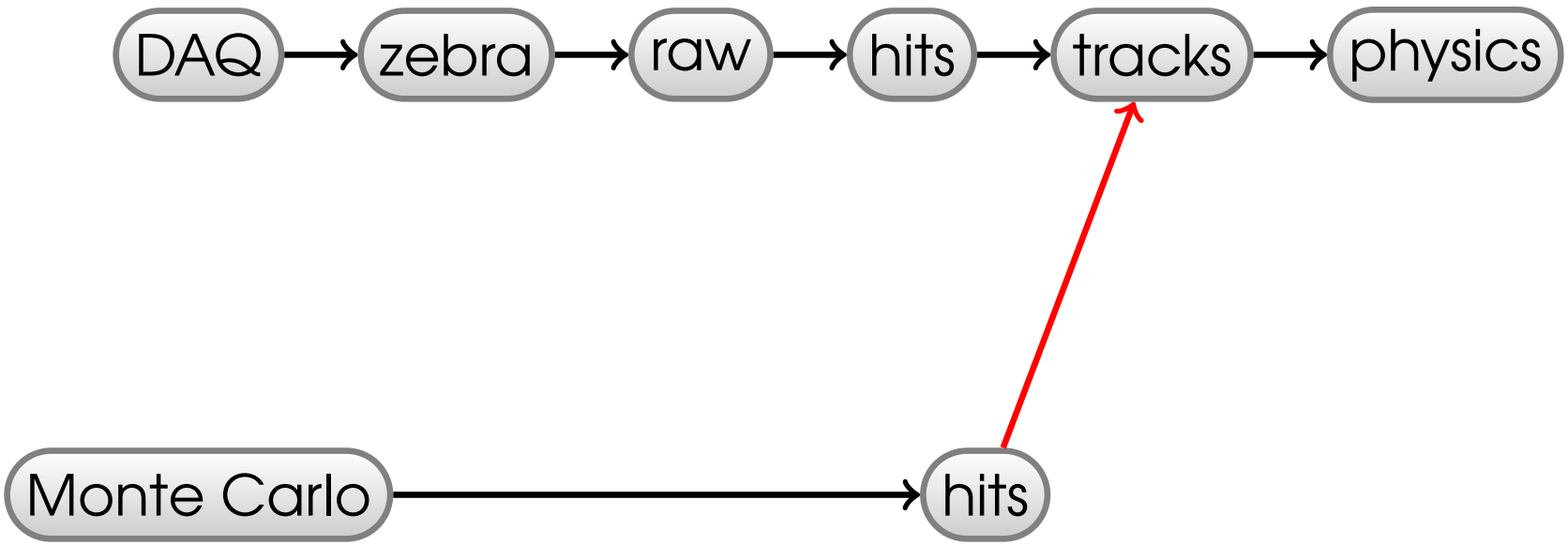
Collaboration Meeting June 2011

Massachusetts Institute of Technology

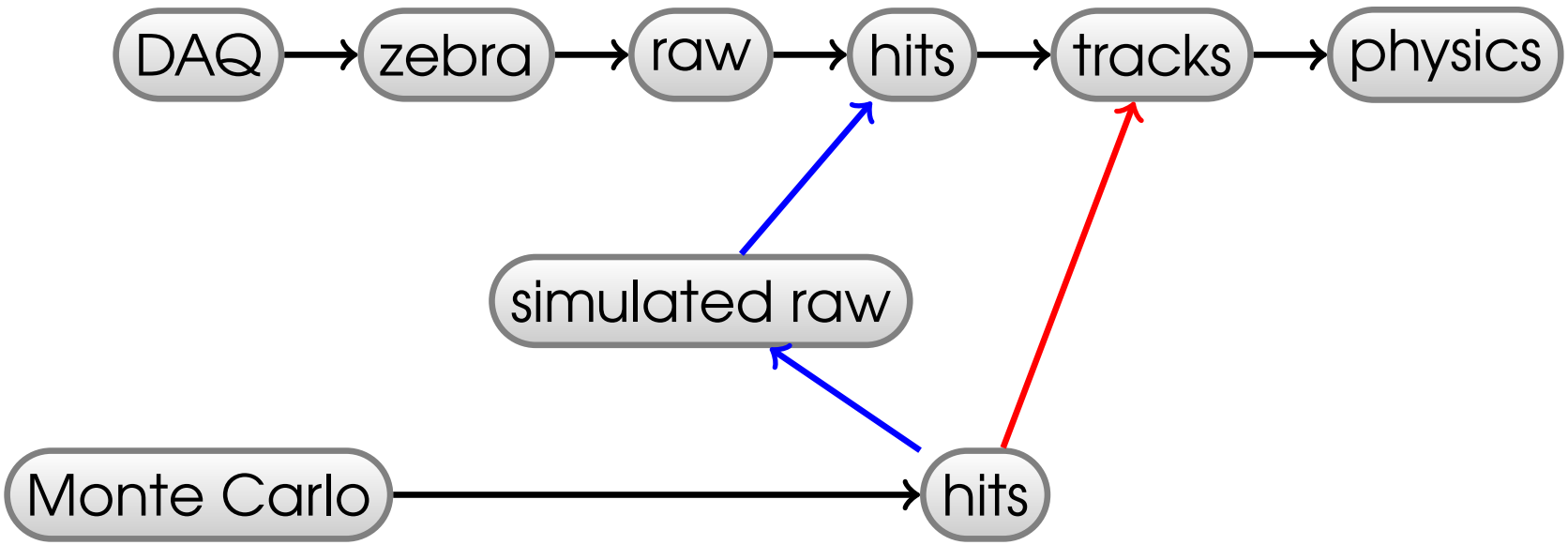
Software overview



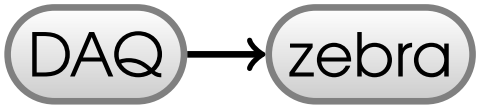
Software overview



Software overview



Raw data root files



- The files generated by the DAQ are zebra files.
- In "hardware order"
- Hard to use, needs "Explora" libraries

Raw data root files

- The files generated by the DAQ are zebra files.
- In "hardware order"
- Hard to use, needs "Explora" libraries

$\longrightarrow$Write Explora plugin to export to standard root file.

- Data is not changed, for most detectors: raw data block.
- But: Mapping to logical channels already performed by Explora framework.

Proposed structure

- Object hierachy, encapsulating logical detectors.
  - Chamber, ToF, Lumi, Moeller
  - Lumi will have members for MWPC and GEMs
- Chambers: Vector of {wire nr, time}
- ToF: Zero suppressed or not? Either array or vector of {bar nr, tube(2) {adc,tdc}}
- Moeller: ???
- GEMs/MWPC: for each detector, a raw data block and its size?

Detector analysis libraries

For each detector, we need software for:

- online monitoring
- data analysis

$\longrightarrow$ Need software libraries which understand raw format.

- Should be provided by the detector groups
- Standardized interface

Proposal: Library Interface

- C++ class
- *int classname::loadData(int len, unsigned char * data)* to load
- *int classname::process(int flags)* to process
- ... some functions for debugging and online monitoring, depends on detector.
- *vector<G4VHit> classname::getHits()* get hit information

Proposal: Library Interface

- C++ class
- *int classname::loadData(int len, unsigned char * data)* to load
- *int classname::process(int flags)* to process
- ... some functions for debugging and online monitoring, depends on detector.
- *vector<G4VHit> classname::getHits()* get hit information
- The actual returned hit type might just inherit from G4VHit. Use the same as the Monte Carlo!

Same code can be used to analyse MC and real data!

Proposal: Initalization files

- Each detector has to have some calibration variables.
- Don't want to reinvent the wheel x times.
- Have a global description file which includes files provided by the groups.
- Should support int, float, strings and arrays of those. More???

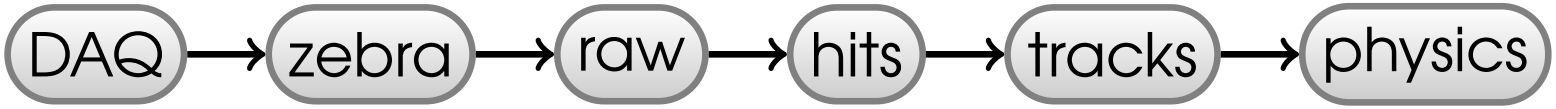
Proposal: Initalization files

- Each detector has to have some calibration variables.
- Don't want to reinvent the wheel x times.
- Have a global description file which includes files provided by the groups.
- Should support int, float, strings and arrays of those. More???
- Detector class has to export known parameters and ways to set it.
- Will be called from main framework for each parameter.

Proposal: Initalization files

- Each detector has to have some calibration variables.
- Don't want to reinvent the wheel x times.
- Have a global description file which includes files provided by the groups.
- Should support int, float, strings and arrays of those. More???
- Detector class has to export known parameters and ways to set it.
- Will be called from main framework for each parameter.
- File format: XML, json, Ini style, name your favorite!

Python, cint or compiled C++



All three have have advantages/disadvantages:

compiled C++
+ fast
- long turnaround times
- lower level language

cint
+ the usual way
+ fast turnaround
- slow execution, but prototype for compiled C++
- Does anyone like cint?

Python
+ higher level, less hassle
+ at least as fast as cint
- harder to switch to compiled C++

Python, cint or compiled C++

All three have have advantages/disadvantages:

compiled C++
+ fast
- long turnaround times
- lower level language

cint
+ the usual way
+ fast turnaround
- slow execution, but prototype for compiled C++
- Does anyone like cint?

Python
+ higher level, less hassle
+ at least as fast as cint
- harder to switch to compiled C++

Proposal: Do anything up to the "physics" stage with compiled C++. Do the physics stage with Python.