



# High Level Trigger

---

Components

AliRoot simulation

AliRoot reconstruction

HLTOUT handling



- 
- What is an HLT component?
  - How to implement an HLT component?
  - How to implement a new detector library?
  - How to run HLT components in AliRoot?
  - How to integrate offline code into HLT?
  - Treatment of HLTOUT?



# What is an HLT component?

---

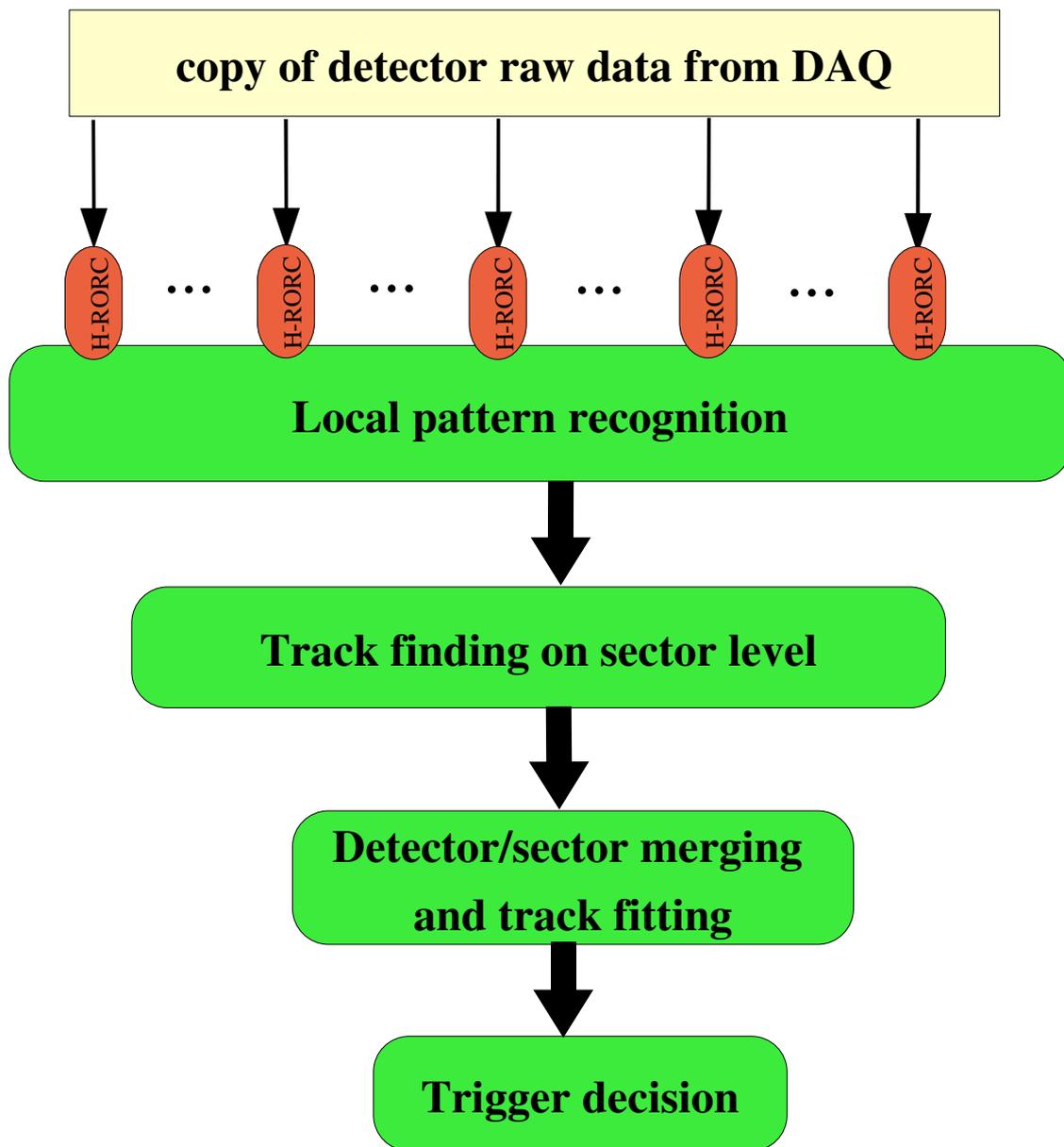
- HLT component implements a self-contained step in the analysis
- HLT components follow common data processing interface
- HLT components (can) run in separate processes
- Distributed over cluster nodes
- Organized in shared libraries (part of AliRoot)



Only the input and calibration data  
is available for processing



# HLT analysis chains



- Analysis carried out by individual software components
- Running in parallel on the nodes
- pipe-lined data flow push architecture
- Local data processing directly on the Front-end processors (FEPs)
- Global data processing with reduced data and event merging done on the compute nodes
- Data exchange via shared memory



# How to implement an HLT component

---

- Generalized interface provided by base class
- Component interface functions need to be implemented:
  - Getters for general Component Properties
  - Component initialization
  - Data Processing
- Component registration via library agent or global object



# Available Base Classes

---

- ➔ all classes inherit component interface from `AliHLTComponent`
- `AliHLTProcessor`
  - general base class for data processors with input and output
- `AliHLTDataSource`
  - external data sources (no HLT stream input)
- `AliHLTDatSink`
  - external data sink (no HLT stream output)
- `AliHLTOfflineDataSource`
  - as `AliHLTDataSource`, provides access to `AliRunLoader/AliRawReader`



# General Component Properties

---

Virtual methods which must be implemented by the component

- `GetComponentID`
  - unique id string
- `GetInputDataTypes` (processors and sinks)
  - list of input data types
- `GetOutputDataType` (processors and sources)
  - output data type
- `GetOutputDataSize` (processors and sources)
  - output data scaling, size estimation
- `Spawn`
  - component creation



# Component Implementation

- Components are the HLT interface to the the actual algorithm
- algorithms implemented in dedicated classes, e.g.  
AliHLTPCClusterFinderComponent uses AliHLTClusterFinder
- `int DoInit( int argc, const char** argv )`
  - internal initialization, allocations, component argument scan
- `int DoDeinit( )`
  - Deallocations, clean-up
- AliHLTProcessor
  - `int DoEvent( const AliHLTComponentEventData& evtData, const AliHLTComponentBlockData* blocks, AliHLTComponentTriggerData& trigData, AliHLTUInt8_t* outputPtr, AliHLTUInt32_t& size, vector<AliHLTComponentBlockData>& outputBlocks )`
    - loop over all input blocks
    - write output data to output buffer (outputPtr) and create block descriptors



# Data exchange between components

---

## C-structures

- Light-weight data exchange
- Analysis can work directly on output buffer

## Transport of ROOT objects

- Based on TMessage (which is restricted to be used by T(P)Socket)
- Allows to transport ROOT objects between components/nodes
  - More convenient, but
  - Overhead due to keys and class description
- Foundation for Calibration and Monitoring
- Foundation of running offline code



# High Level Component Interface

---

- Simplified DoEvent function

```
DoEvent( const AliHLTComponentEventData& evtData,  
         AliHLTComponentTriggerData& trigData
```

- GetFirstInputObject / GetNextInputObject
  - Iteration through input blocks
- PushBack( TObject\* , ... )
  - Insert TObject into component output
- Suited for Monitoring and Calibration components, but not high-performance analysis

<http://web.ift.uib.no/~kjeks/doc/alice-hlt-curent/classAliHLTComponent.html>



---

# Detector libraries



# Detector libraries

---

- Detector dependent source code is organized in shared libraries
- Analysis code and components
- HLTOUT Handlers
- Modular system: Plugin mechanism through module agents
  - provide properties of the module
  - Transparent instantiation and registration of Components, Analysis Chains, HLTOUT handlers
  - A new library just needs to be loaded, no adjustments in the framework
- Recipe:
  - [http://web.ift.uib.no/~kjeks/doc/alice-hlt-current/sample\\_readme.html](http://web.ift.uib.no/~kjeks/doc/alice-hlt-current/sample_readme.html)
  - In AliRoot: HLT/SampleLib/README



# Library agents

---

- Module agent: `AliHLTModuleAgent`
  - `RegisterComponents`: define and register components
  - `CreateConfigurations`: define tasks
    - Component / Component arguments / Data sources (parent publishers)
  - `GetReconstructionChains`: chains to run
    - AliRoot simulation or event reconstruction
    - Chains can be defined depending on the availability of the `AliRunLoader` or `AliRawReader`
  - `GetHandlerDescription`: HLTOUT processing
    - Implementation and description of HLTOUT handlers

<http://web.ift.uib.no/~kjeks/doc/alice-hlt-curent/classAliHLTModuleAgent.html>



---

# HLT in AliRoot



# HLT Reconstruction interface

---

- AliHLTSystem

- Loads the HLT component libraries
- Module agents define chains to run
- Identical for AliRoot Simulation and reconstruction

- Options

<code>config=conf-macro.C</code>	configurations defined by macro instead of agents
<code>chains=chain1,chain2</code>	custom chains
<code>loglevel=0x&lt;level&gt;</code>	HLT loglevel (0x3f all, 0x3c info and higher)
<code>libAliHLT&lt;...&gt;.so</code>	load component library instead default libs

- [http://web.ift.uib.no/~kjek/s/doc/alice-hlt-current/group\\_\\_alihlt\\_\\_tutorial.html](http://web.ift.uib.no/~kjek/s/doc/alice-hlt-current/group__alihlt__tutorial.html)



# HLT @ AliSimulation

- AliHLTSimulation
  - Last step of AliRoot simulation
  - Loads the HLT component libraries
  - Module agents define configurations and chains to un
  - Custom configurations and chains possible
- Example (processing of already simulated data)

```
AliReconstruction sim;           // the simulation instance
sim.SetRunGeneration(kFALSE);    // do not generate particles
sim.SetMakeDigits("");          // disable
sim.SetMakeSDigits("");         // disable
sim.SetMakeDigitsFromHits("");  // disable
sim.SetMakeTrigger("");
sim.SetRunHLT("libAliHLTPPC.so  config=conf-tpc-esd.C  chains=sink1");
sim.Run();
```

- HLT Simulation = HLT Reconstruction of simulated data



# Custom HLT system configuration

```
{// conf-tpc.C
  int iMinSlice=0;
  int iMaxSlice=35;
  int iMinPart=0;
  int iMaxPart=5;
  TString writerInput;
  for (int slice=iMinSlice; slice<=iMaxSlice; slice++) {
    TString trackerIn;
    for (int part=iMinPart; part<=iMaxPart; part++) {
      TString arg, pub, cf;

      // digit publisher components
      arg.Form("-slice %d -partition %d", slice, part);
      pub.Form("DP_%02d_%d", slice, part);
      AliHLTConfiguration pubconf(pub.Data(), "TPCDigitPublisher", NULL , arg.Data());

      // cluster finder components
      cf.Form("CF_%02d_%d", slice, part);
      AliHLTConfiguration cfconf(cf.Data(), "TPCClusterFinderUnpacked", pub.Data(), "pp-run timebins 446");
      if (trackerIn.Length(>0) trackerInput+=" ";
      trackerIn+=cf;
    }
    TString tracker;
    // tracker finder components
    tracker.Form("TR_%02d", slice);
    AliHLTConfiguration trconf(tracker.Data(), "TPCSliceTracker", trackerIn.Data(), "-pp-run -bfield 0.5");

    if (writerInput.Length(>0) writerInput+=" ";
    writerInput+=tracker;
  }

  // the esd writer configuration
  AliHLTConfiguration esdwconf("esd-writer", "TPCESdWriter", writerInput.Data(), "-datafile AliHLTESDs.root");
}
```



# HLT Simulation: components

---

## INPUT

- Detector specific offline source component(s) convert(s) digits to native input format of the first component in the chain
  - e.g. TPCDigitPublisher
- Common component of the framework can publish digit tree:
  - AliHLTLoaderPublisherComponent

## OUTPUT

- Common component of the framework takes all incoming data blocks, adds HOMER descriptor and stores it in the appropriate place of the RunLoader
  - Identical to HLTOUT nodes
- Note: Detector specific offline sink component(s) have been removed -> data is forwarded to HLTOUT and corresponding handler



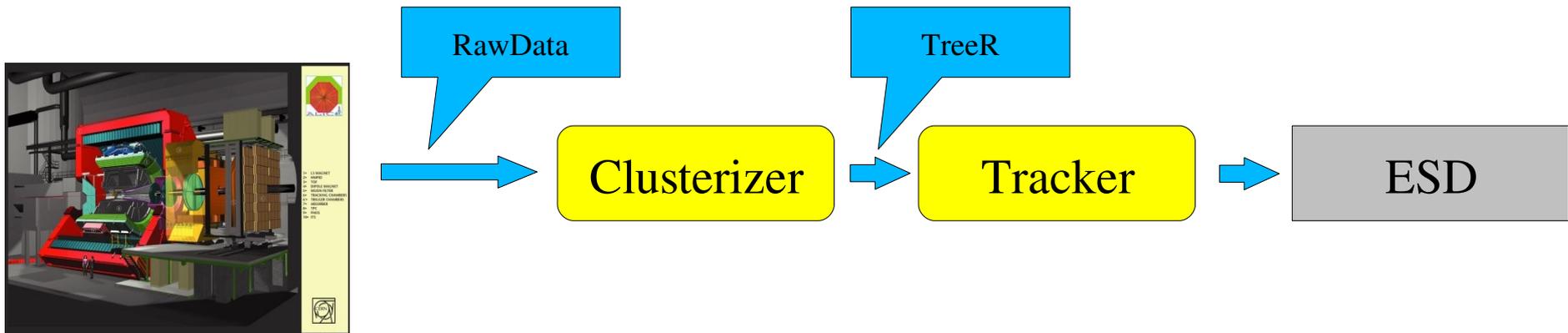
# HLT @ AliReconstruction

---

- AliHLTReconstructor
  - Loads the HLT component libraries
  - Module agents define chains to run
  - Chains run on data provided by RawReader, produced data is added to already existing HLTOUT
  - HLTOUT loop executed during FilleSD
- Example

```
AliReconstruction rec;           // the reconstruction instance
rec.SetRunTracking("");         // switch off tracking
rec.SetFilleSD("HLT");          // run rec only for HLT
rec.SetOption("HLT", "libAliHLTSample.so");
rec.Run();
```

# Integration of offline code



- Simple wrapper components allow running the offline code on-line
- Clusterizer:
  - Input: RawReader, for HLT AliRawReaderMemory
  - Output: same object as written to TreeR is forwarded to the tracker
- Tracker:
  - Input: TObject structure(s) as provided by clusterer
  - Output: AliESDEvent pushed into the component output

See [Talk by Mateusz Ploskon \(Alice week Mar 07\)](#)



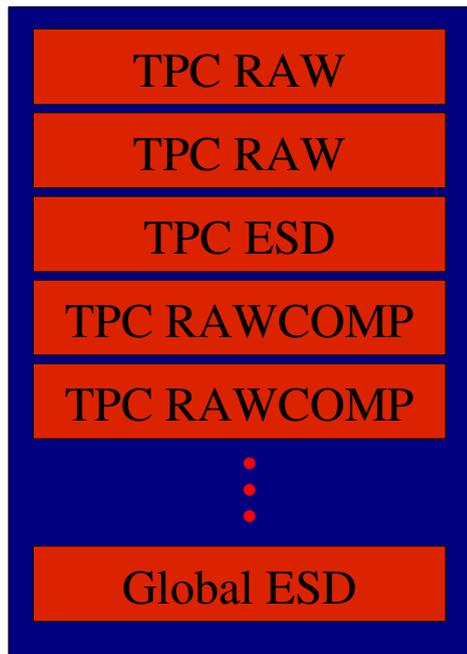
---

# HLTOUT



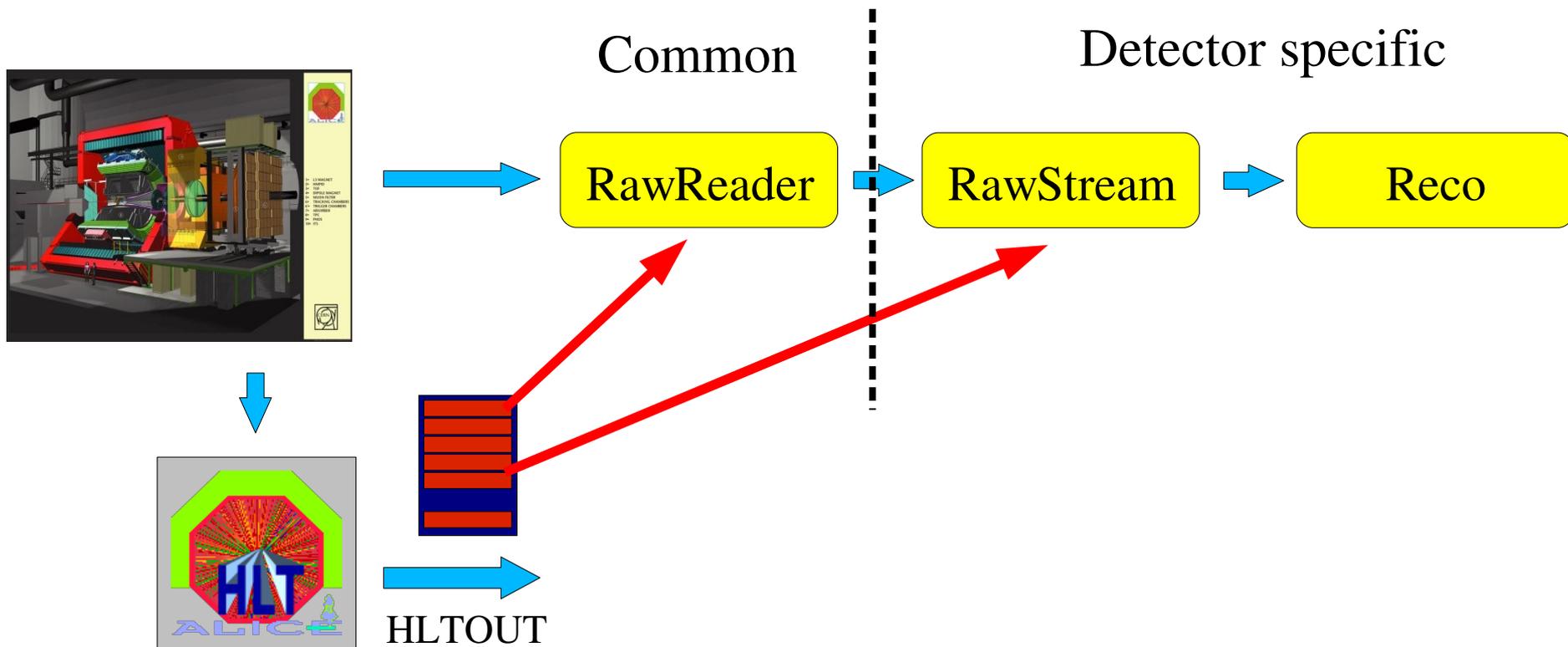
# HLT output

- Concatenated data blocks in the HOMER format
- Content of the HLTOUT blocks depends on the active configuration
- AliRoot: common sink component generates HLT output equivalent to HLTOUT nodes



- HLTOUT is a collection of different and possible independent data blocks
- Detector plugins required
- ESDs handled by framework with possibility for detector plugin
- Handler Types:
  - ESD, RawReader, RawStream, Chain, Proprietary

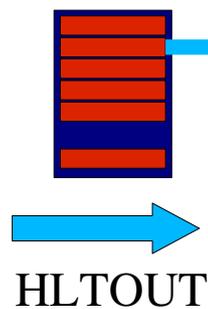
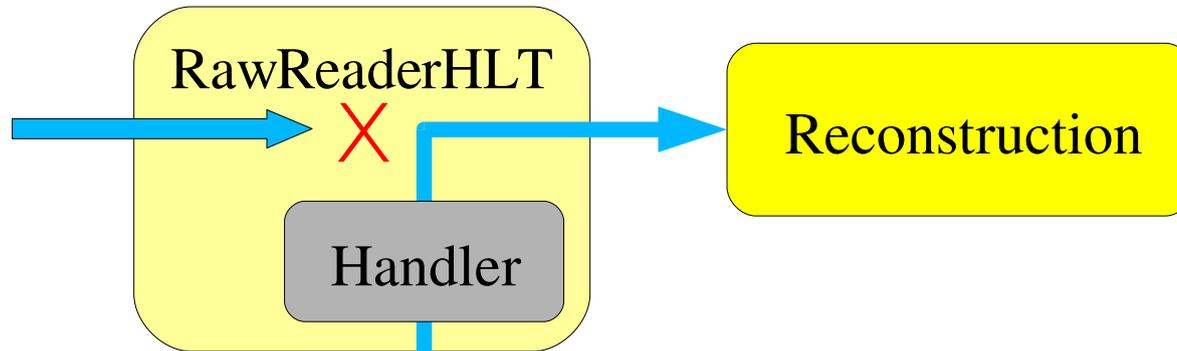
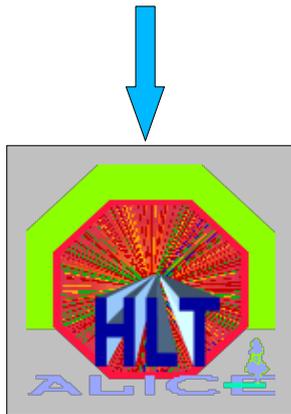
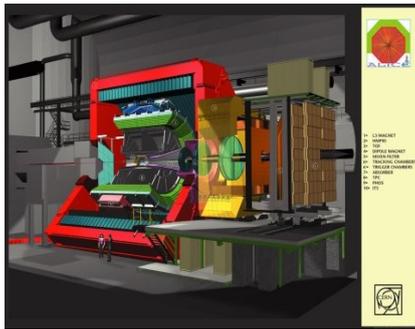
# Data Replacement



- AliRawReaderHLT: transparent replacement of reconstruction input
- Replacement on RawStream level is detector dependent, needs more discussion because of different RawStream models



# AliRawReaderHLT



```
AliReconstruction rec;  
rec.SetUseHLTData("TPC");  
rec.SetRunReconstruction("TPC TRD");  
rec.SetInput("./");  
rec.SetRunVertexFinder(kFALSE);  
rec.Run();
```

- ➔ Handler must be provided in order to extract equipment id from data type and specification



# TPC RawDataHandler

```
int AliHLTPCAgent::AliHLTPCRawDataHandler::ProcessData(AliHLTOUT* pData)
{
    // TPC HLTOUT handler for TPC raw data blocks
    // returns equipment id
    if (!pData) return -EINVAL;
    AliHLTComponentDataType dt=kAliHLTVoidDataType;
    AliHLTUInt32_t spec=kAliHLTVoidDataSpec;
    int iResult=pData->GetDataBlockDescription(dt, spec);
    if (iResult>=0) {
        int slice=AliHLTPCDefinitions::GetMinSliceNr(spec);
        int part=AliHLTPCDefinitions::GetMinPatchNr(spec);
        if (slice==AliHLTPCDefinitions::GetMaxSliceNr(spec) &&
            part==AliHLTPCDefinitions::GetMaxPatchNr(spec)) {
            iResult=768;
            if (part>1) iResult+=72+4*slice+(part-2);
            else iResult+=2*slice+part;
        } else {
            HLTErrror("handler can not process merged data from multiple ddls:"
                " min slice %d, max slice %d, min part %d, max part %d",
                slice, AliHLTPCDefinitions::GetMaxSliceNr(spec),
                part, AliHLTPCDefinitions::GetMaxPatchNr(spec));
            iResult=-EBADMSG;
        }
    }
    return iResult;
}
```



# Status of AliRoot integration

---

- Simulation: Framework for Detector plugins ready
  - TPC conformal mapping tracker embedded
  - TPC ALTRO Huffman compression embedded
- HLTOUT framework
  - Framework for detector plugins
  - Standard handling of ESD data blocks
- Detector ESDs of detectors written to separate files
- Merging of ESDs under discussion (can be done already during the HLT online analysis)



# All ways lead to ... ESD

