



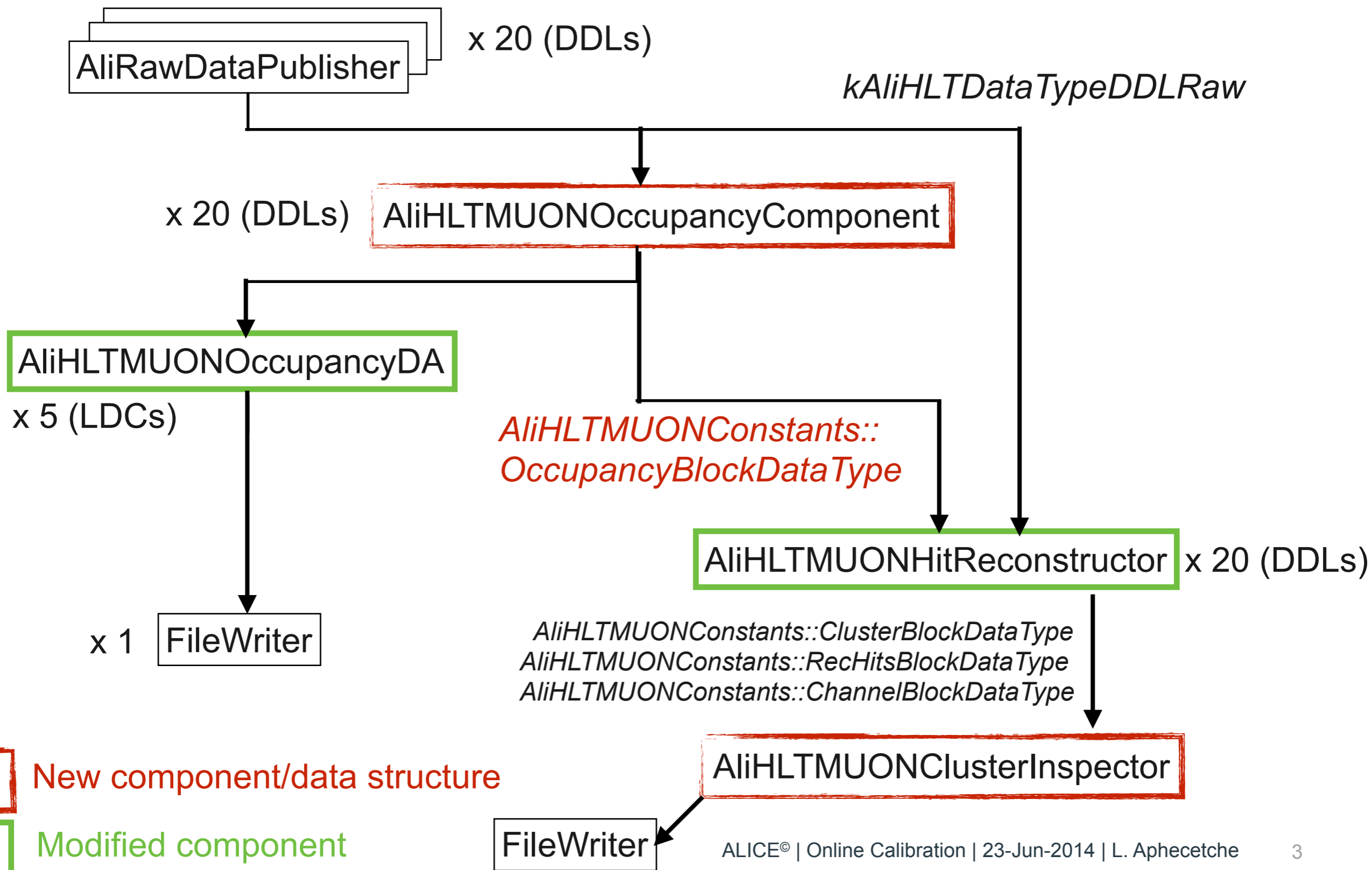
MUON online calibration

Short reminder about Muon calibration

- Muon does not have « calibration » per se
- Yet, some inputs to the reconstruction have to be computed from raw data
 - case in point : MCH needs to compute FEE occupancy to discard some of it prior to clustering stage
- The MCH occupancy computation was done (*) in Run1 by a DAQ DA (MUONTRKOCcda), which writes data into FXS. Shuttle preprocessor then reads FXS and upload results to OCDB.
- This presentation is about :
 - implementing the occupancy DA within existing HLT (FXS part)
 - use the computed occupancy directly within other HLT Muon component(s)
 - as part of a bigger goal of updating the full online Muon reconstruction

(*) and will continue to be done in Run2 until we validate the HLT option

Current test chain(s)



Dealing with occupancy variations

- The OccupancyComponent computes the occupancy for a given (tunable) number of events (or all events by default) within one « cycle »
- At the end of each cycle (or at EOR if cycle is infinite) the occupancy blocks are put into the data stream and the internals reset to start a new cycle
- Not thoroughly tested yet...
 - for the moment the FXS only gets the last cycle
 - will require anyway a change in the file format used by the Shuttle and the corresponding preprocessor

Status

1. Live testing of what it takes (for a newcomer coming from offline) to port code to HLT

- connection of components / source / sinks still a bit touchy to debug
- like all software frameworks, it takes time to get RE-used to...
- trying to find out if the (dHLT) design decisions made long ago are still valid



2. Port MUON Occupancy DA to HLT

- more or less done except for :
 - still have to « polish » the code a bit...
 - still have to see if code can be optimized
- split the occupancy creation and the occupancy output to FXS into two components (only 1 previously which did both)



3. Use output of that DA also within the HLT reconstruction

- proof-of-principle done
- basic data structure to exchange occupancy blocks between components



A bit greener, but not there yet...

Still have to profile

Still have to test on a system a bit more realistic than my laptop

Next (short to medium-term) steps

- Profile and polish (and x-check) the occupancy parts.
- Commit : what are the prerequisites for a HLT commit ?
- Several occupancy maps per run : how exactly ?
- MCH : Inserts a new HitReconstrutor (based on work by Philippe Pillot on pre-clustering) to see how it compares (speed-wise) with the existing
- MTR : follow-up on what's being done on that side

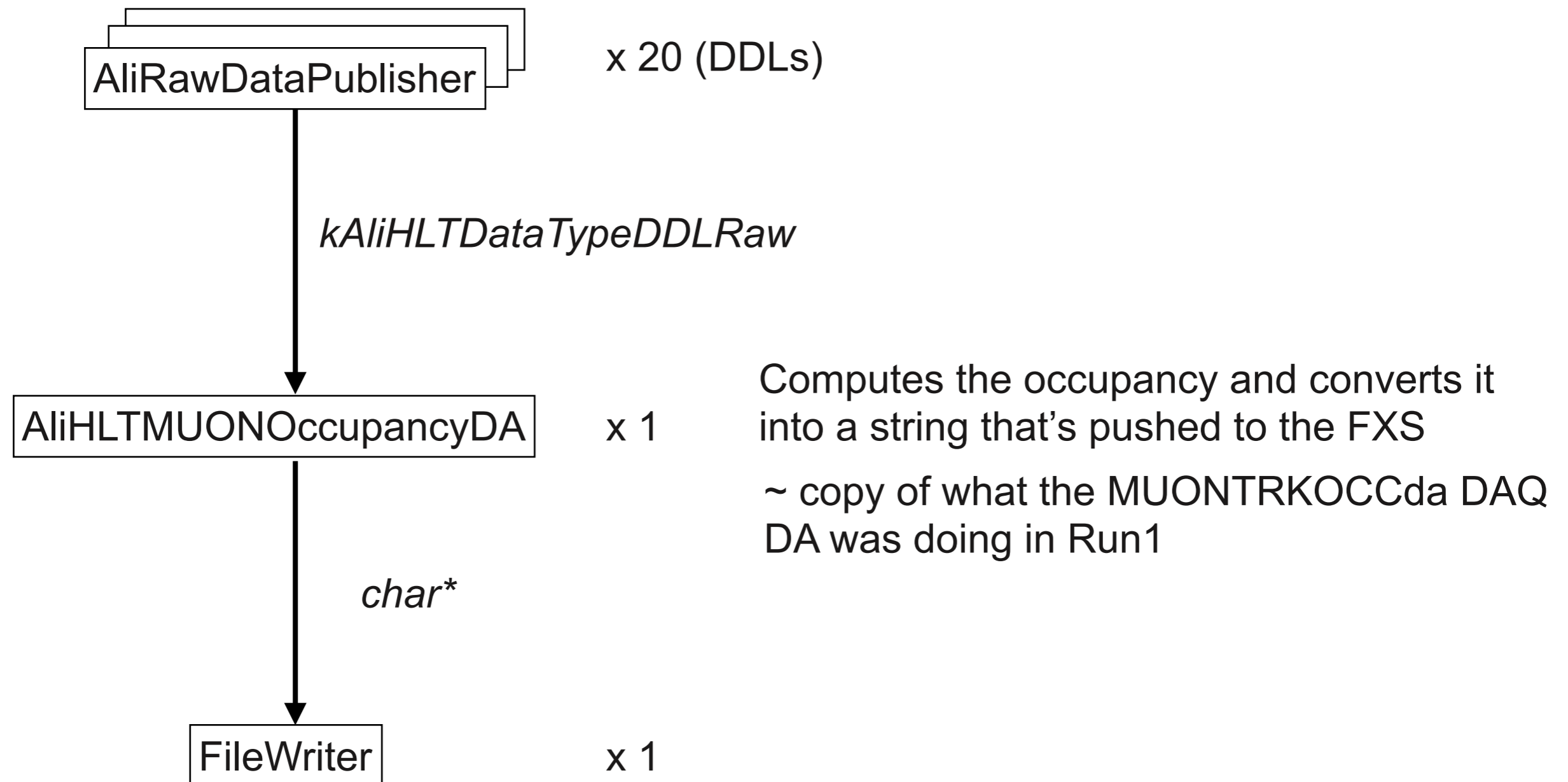
A Large Ion Collider Experiment






ALICE

BACKUP

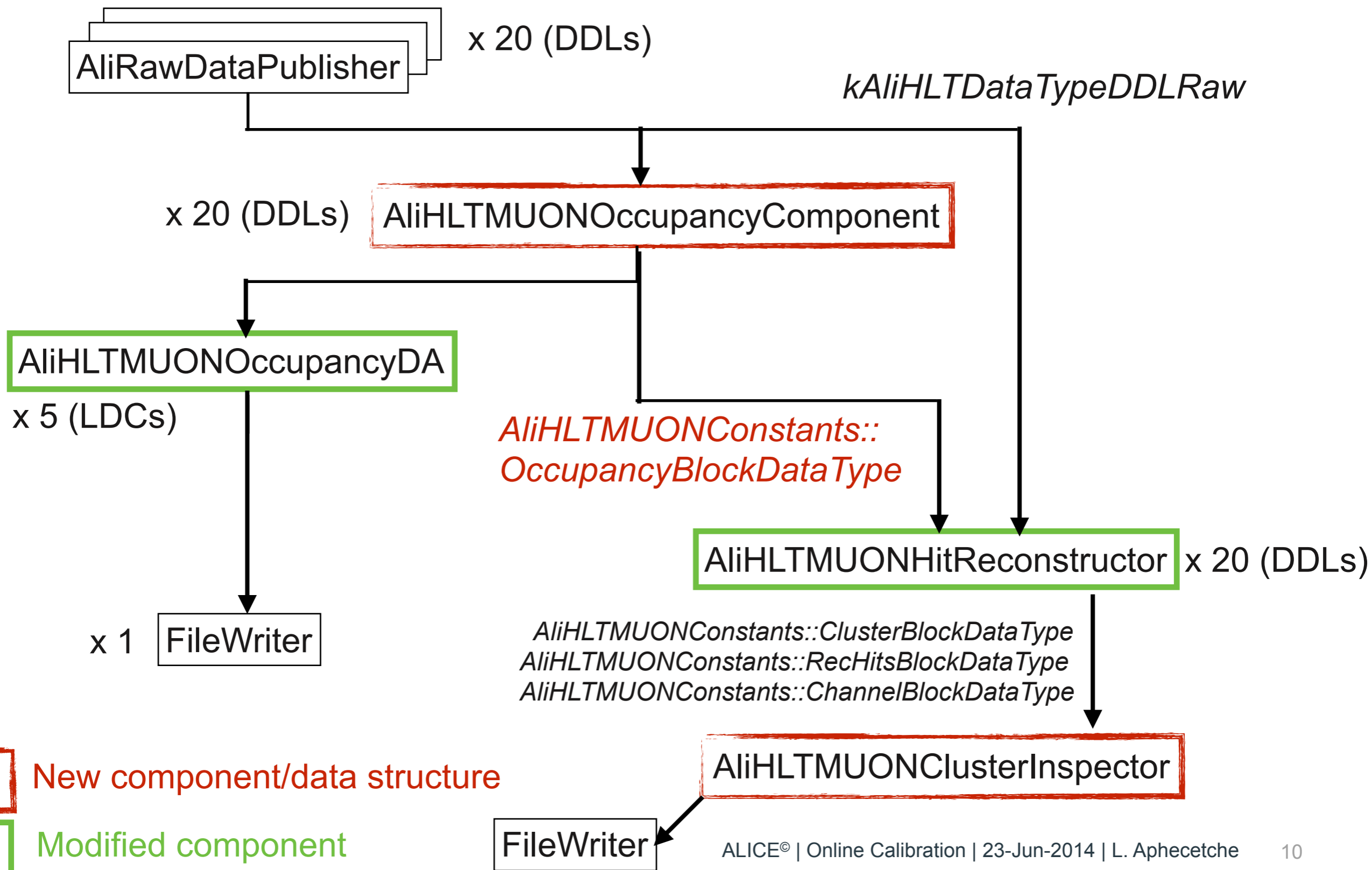
Previous test chain (dec 2013)



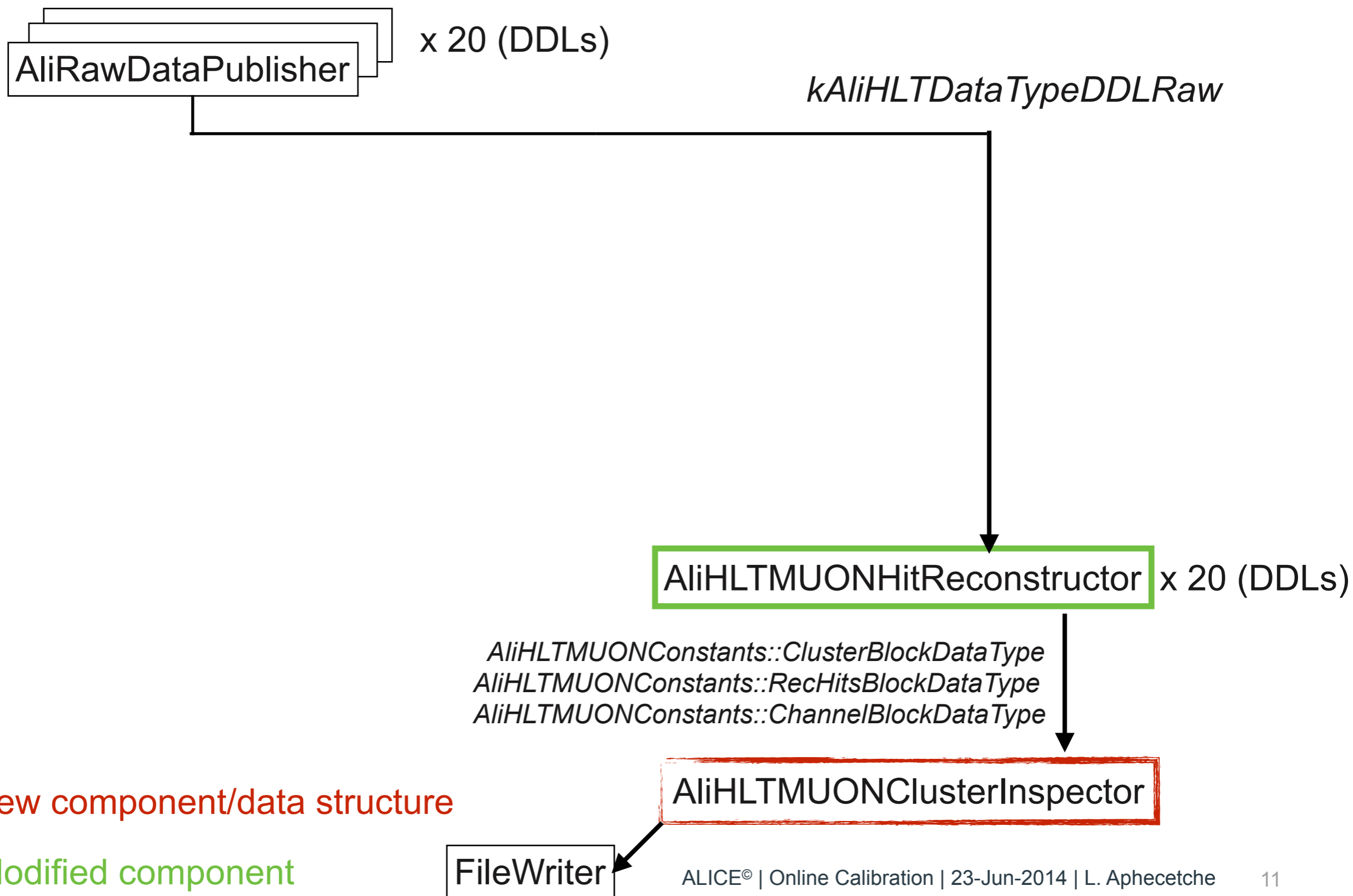
Status (dec 2013)

1. Live testing of what it takes (for a newcomer coming from offline) to port code to HLT 
 - looks like all software frameworks, e.g. bugs are found and fixed ;-)
 - like all software frameworks, it takes time to get used to...
2. Port MUON Occupancy DA to HLT 
 - more or less done for the pure DA-like part, *i.e.* the one which is simply outputting a (text) file
 - well, at least output is the same (see next slides)
 - will still have to « polish » the code a bit...
 - will still have to see if code can be optimized (might be related to 3))
3. Use output of that DA also within the HLT reconstruction 
 - no coding started yet
 - have to think about data structure

Current test chain(s)



Current test chain(s) : « reference » chain



What changed : one new data structure...

```
extern "C" {  
  
    struct AliHLMUONOccupancyStruct  
    {  
        AliHLTInt32_t fBusPatchId;  
        AliHLTInt32_t fManuId;  
        AliHLTInt32_t fSumOfN;  
        AliHLTInt32_t fNofEvents; } occupancy = fSumOfN / fNofEvents  
    };  
  
    struct AliHLMUONOccupancyBlockStruct  
    {  
        AliHLMUONDataBlockHeader fHeader;  
    };  
}
```

... trying to fit into the existing schema of dHLT classes

What changed : components

- One component (**OccupancyComponent**) is computing the occupancy from raw data and adding occupancy data blocks to the data stream every N events (tunable)
- One component (**OccupancyDA**) is converting the occupancy blocks to a string (compatible with what is currently read by the Shuttle) and ships it to FXS
- One component (the current **HitReconstructor**) is updating its occupancy map each time it finds occupancy blocks in the data stream, and discard manus above a given occupancy threshold, prior to perform clustering