Monitoring for HLT and offline

T. Bold AGH UST Krakow
for HLT s/w upgrade group

Monitoring in re-entrant HLT code

- NO to storing monitored quantities in class attributes
  - that is what we have now
- Functionally similar to existing HLT code
- Available also for offline alg/tools
  - because HLT would use them w/o wrapping

Work done with Piotr Sarna

Idea

- Separate **histograms management**
  - this is boiler plate code:
    - create (decide on names, binning, actual implementation i.e. TH1?),
    - handle (registration)
- from **histogram filling**
  - always specific to the alg/tool
  - i.e. when is Fill operation happening

Monitored variables

- In order to make monitoring ubiquitous but not invasive information should be silently extracted from any variable (i.e. local temporary)
  - Extraction may happen at:
    - every assignment
    - end of lifetime (this is what we have in HLT now)
    - on explicit fill
- This would need to be policy decided for every variable independently

Monitored Variables in code

```
::execute() {
…
Monitored<double> pt = m_monitoringTool->create_monitored<double>(“pt”, 0, FillAtAssign);
pt = 7.12; // here histogram filling occurs
ptSum = ptSum + pt; // ptSum is a plain double here increased by value in “pt”
…
```

- Temporary creation associates variable with observer (histogram filler/proxy)
  - name (string) default value and policy are decided then
  - from then it is like every other “double” (we can make is as alike as we want via. operators overload)

Histograms management

- MonitoringTool fucntions
  - construction of monitored var.

```
Monitored<T> create_monitored(const std::string& id,
const T& default_value = {},
FillPolicy fillPolicy = FillDisabled)
```

  - creation with histogram proxies pre-acquired

```
Monitored<T> create_monitored(std::shared_ptr<StorageProxy> monitored_proxy,
const T& default_value = {},
FillPolicy fillPolicy = FillDisabled)
```

Configuring MonitoringTool

```
Hists configuration:
from L1Decoder.L1DecoderConf import MonitoringTool
tool = MonitoredTool('monitoredTool')
tool.Histograms = [
    'eta, NoStorage',
    'pt, /monitored, exampleTitle, 100, 0, 100']
```

- For each variable a histogram can be defined
  - Optional NoStorage would mean: not interesting, no histogram no overheads
- There could be only one implementation
  - Simplify maintenance i.e. move to LWHist of OH::Hist would be straightforward
  - Thread safety in one place, muttexing Fill or accumulation … easy to introduce/change, LB awareness

Configuration options

- When no configuration for histogram:
  - provide NoStorage proxy
    - this would help in migration
      - no side effects after replacing variable XYZ with the Monitored<> XYZ
    - It would allow to switch/on off monitoring entirely: Histograms=[]
- Other option is to complain if:
  - some variables are not monitored (alert at create_monitored)
  - histograms never used (tricky as it is data dependent) i.e. decoding issues bookkeeping histogram

Status

- Functional code in SVN (fine tuning pending)
- Test jobs indicate viability of this approach (timing measurements will follow)
- Discussed with DQ (Peter Onyisi) - follow up path established
- In summer we will have ready to go implementation in AthenaMonitoring
  - On us will be to advocate usage of this tool everywhere
    - HLT will be easy to convince because we have now similar thing, new approach offers certain simplifications (FillAtAssign), can monitor tools & services
    - Offline should follow too: reduce boiler plate in monitoring, when offline code in HLT no wrapping needed to have online monitoring