Ring Buffer implementation

E. Castro

DESY-Zeuthen

Description

The Ring Buffer was developed by software engineer Konstantin Boyanov.

The Ring Buffer data structure was developed in order to decouple the process of data taking from the TDC residing on the VME bus, and the Analysis and DIP Publishing, both being tasks of the data acquisition software, as well as to provide a reliable means to store samples of the hit data for analysis after a beam dump has occurred (the so-called post-mortem analysis).

The RingBuffer Class is a template C++ class that can be configured to store different number of objects of other classes.

Principle of operation

- The monolithic DAQ code is split it into two parts, connected through the RB.
  - **First part:**
- *Process 1*- daq and writing of data to the RB.
  - **Second part:**

NORMAL DAQ OPERATION (NO BEAM DUMP)

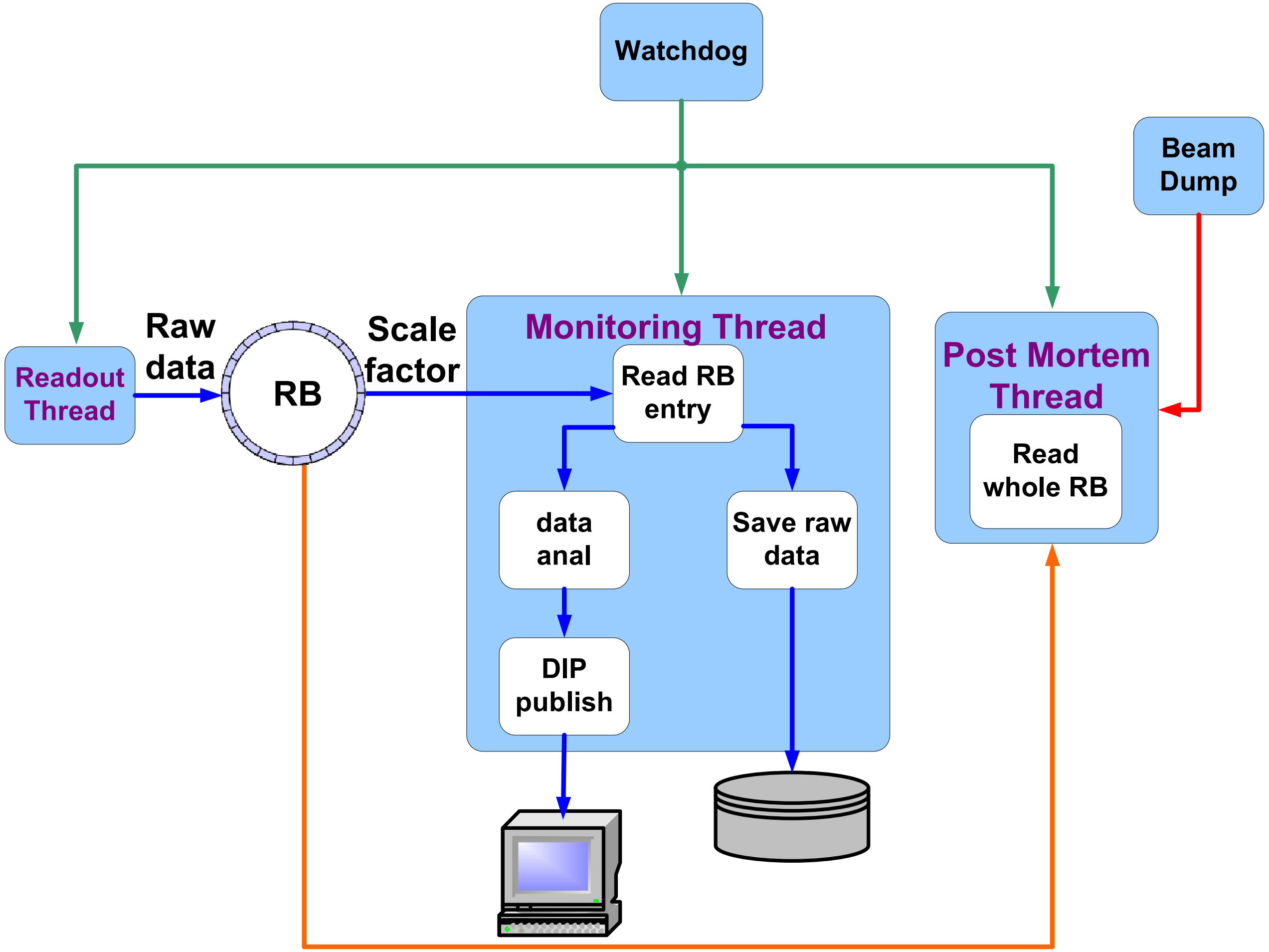
- *Process 2*- a single most recent entry will be read from the RB (scale factor for high Lumi?), saved (ctrl3->CASTOR) and pre-analyzed for publication via DIP.

POST MORTEM (BEAM DUMP)

- *Process 3*- The PM analysis will be a separate process invoked by: beam dump condition, an operator, some other person or some other condition.

- The plan is that, all 3 processes will be started and watched over by a master process (maybe a watchdog or a controller thread)
- In this way, all the relevant tasks will decoupled from each other. The readout, which has to be as fast as possible, is not slowed down by the analysis, and in all the participating processes changes can be made separately from each other (for example, one can make some optimization on the readout while somebody else takes care of integrating new analysis techniques in the analysis part).

RB scheme


Watchdog
Beam Dump
Raw data
Scale factor
Readout Thread
RB
Monitoring Thread
Read RB entry
data anal
Save raw data
DIP publish
Post Mortem Thread
Read whole RB

Code at Zeuthen

3 processes:

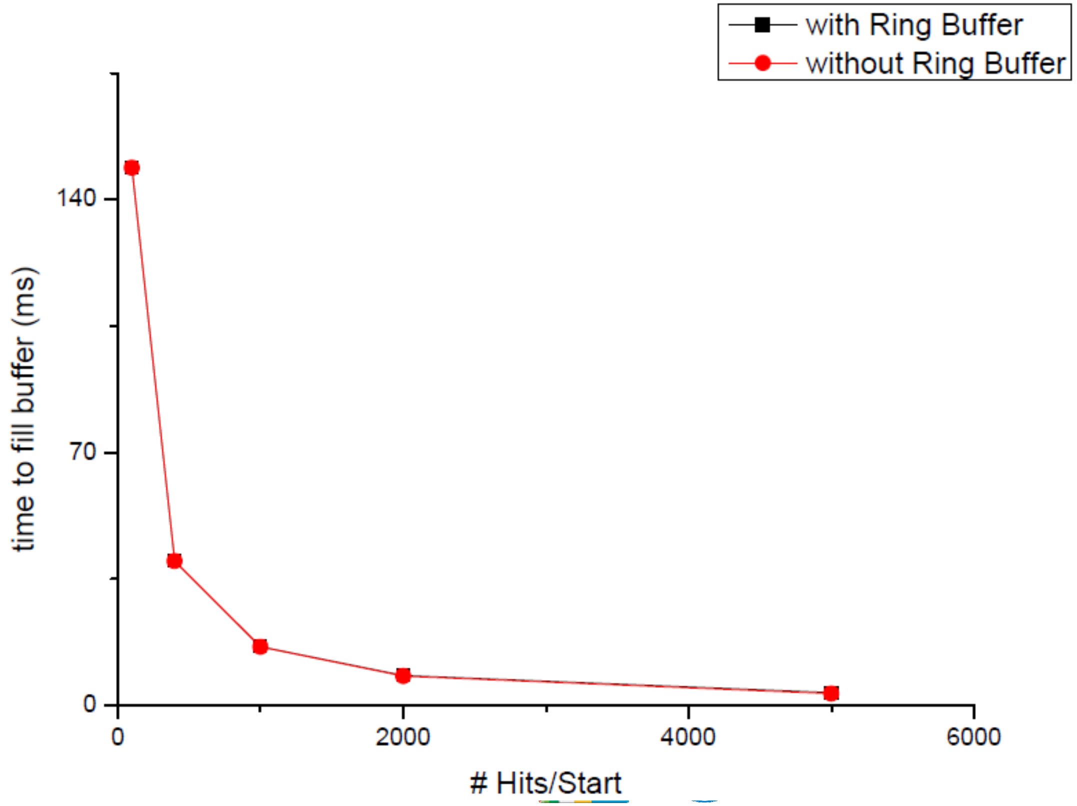
./Readout→Initialize TDC, BLT of raw data, storage in RB

./Analysis→pick up info from RB, store in .root file and make basic data analysis for DIP publishing

./PostMortem→invoqued after Beam Abort

Some results

- The RB was tested in Zeuthen with the TDC module, applying test pulses from a pulse generator.
- We compared the performance of the new code with the usual daq code with BLT. The result shows no difference in normal daq.
- The Beam Abort condition was not tested.



What has still to be done?

- Import the code to CERN
- Adapt the daq code to the current daq scheme in the different modules
- Integrate in the RB based daq code the LUT that detects the hardware Beam Abort signal.
- How to change the RB parameters?
- Contact initiated with expert (K. Boyanov) but need to focus on this subject.