QC for FIT
Status / Quickstart

Miłosz Filus

Overview

1. Test data generation procedure
2. Running basic workflow (on one node)
3. Running advanced workflow (on multiple nodes)
4. Local visualization of results
5. Online visualization of results
6. Post processing procedures
7. Dispatcher
8. Using raw data

Test data generation procedure

To be able to test QC workflows one needs generate input data. Generation of FT0 events used in prepared workflows is following (these were my steps - I'm not an expert here):

1. Generate events and transport particles through detectors
   o2-sim -g pythia8 -e TGeant3 -m FV0 FT0 FDD -j 2 -n 100
2. Digitize hits only from selected detector (in my case it was FT0)
   o2-sim-digitizer-workflow --onlyDet FT0 -b

   Now file ft0digits.root should be generated in current workdir, one can use this file or follow next steps to get more experiment like data

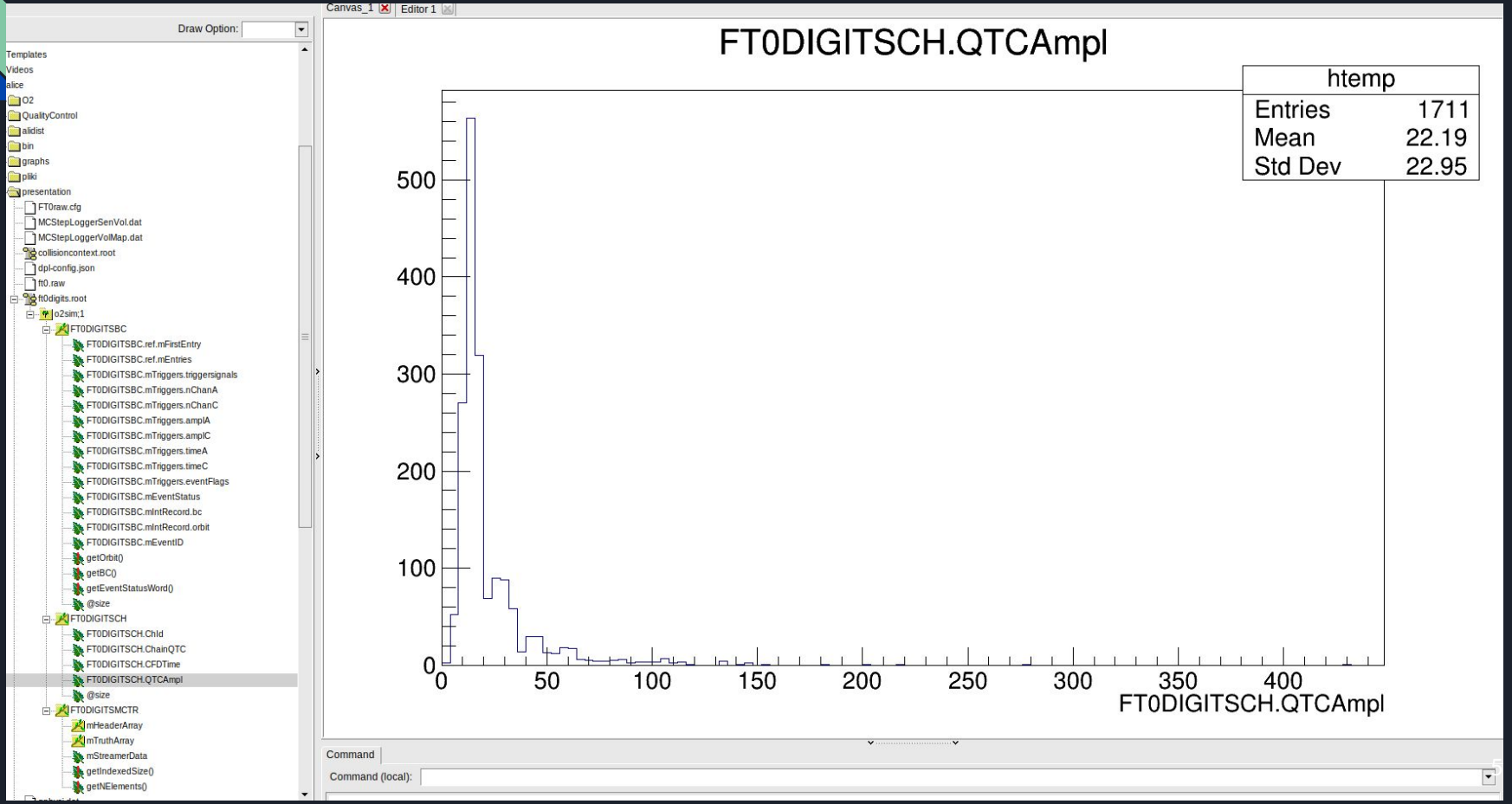
3. Convert MC digits to RAW data format as in experiment
   o2-ft0-digi2raw
4. Convert back RAW data format to digits
   o2-raw-file-reader-workflow --input-conf FT0raw.cfg | o2-ft0-flp-dpl-workflow -b

   Executing steps above, one should get as a result file: o2digits_ft0.root which is ready to use

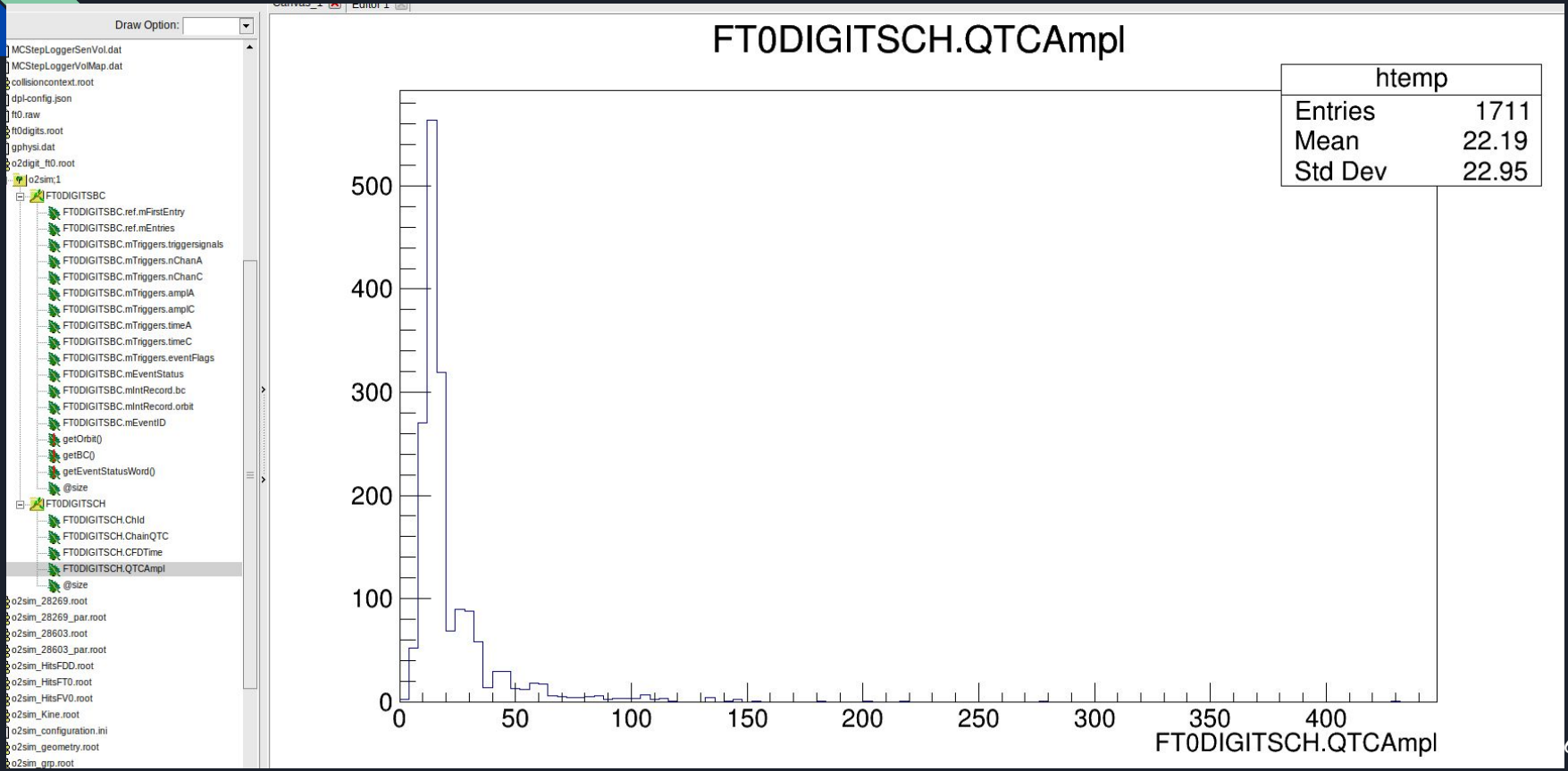
How generated data should look like (more or less)

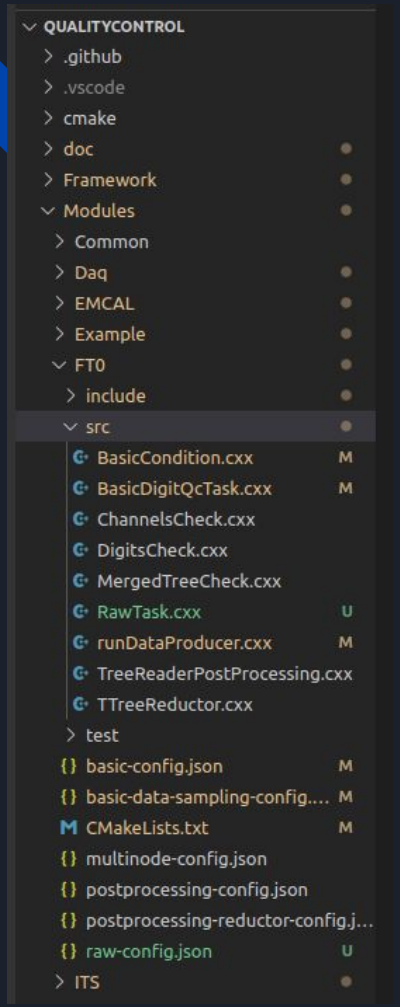
To be able to visualize generated data one can use ROOT and execute following line
TBrowser t;
One should choose generated .root file (ft0digits.root or o2digits_ft0.root)

ft0digits.root tree



o2digits_ft0.root tree

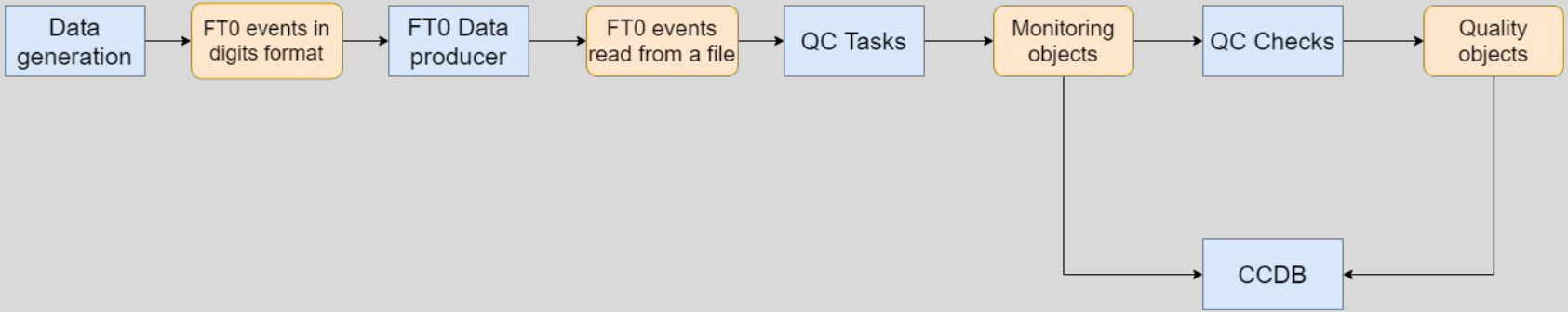




FT0 module in QC framework

Each detector has its own module in QC framework, where one can define tasks, checks, post processing tasks and any additional code which is required for selected detector. In my case it was a data producer. One can module directory for selected detector using ***o2-qc-module-configurator.sh*** script (more detailed information in the qc documentation).

Single node workflow



Single node workflow implementation

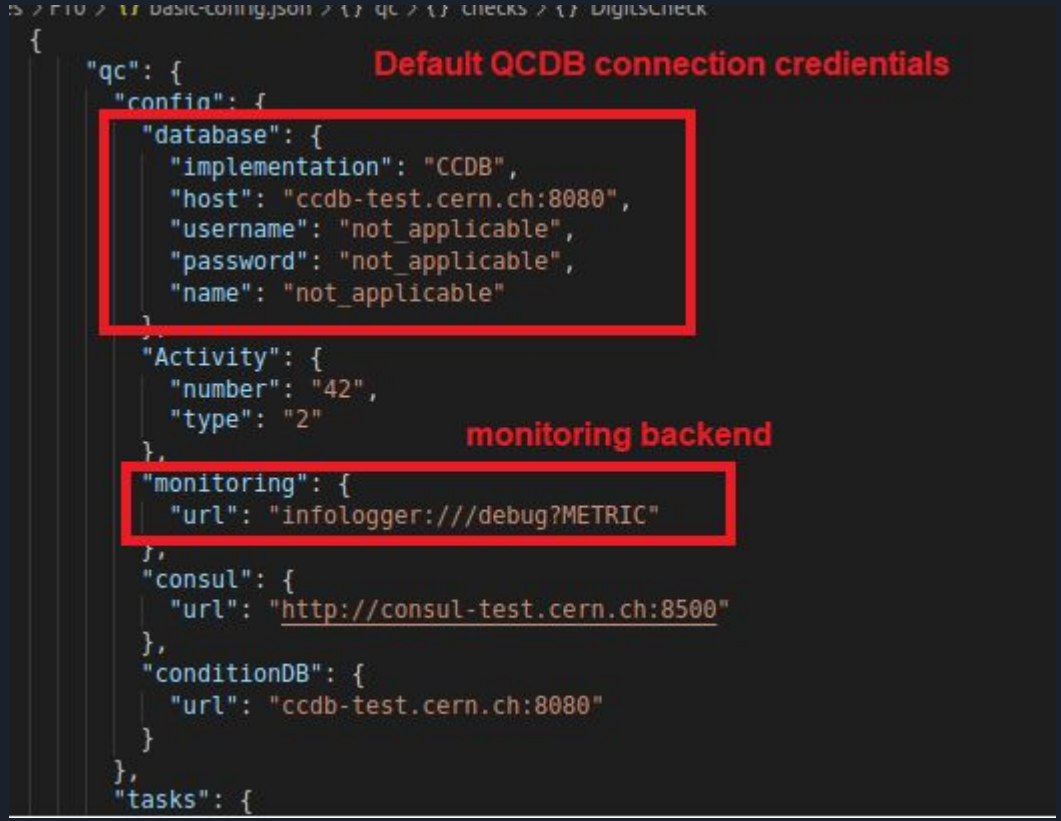
To run this workflow it was needed to define:
- data producer - which reads data from file and sends it to next device in pipeline
src/runDataProducer.cxx

- QC task - which receives data from producer and then creates QC Object like histograms or trees
src/BasicDigitQcTask.cxx
include/FT0/BasicDigitQcTask

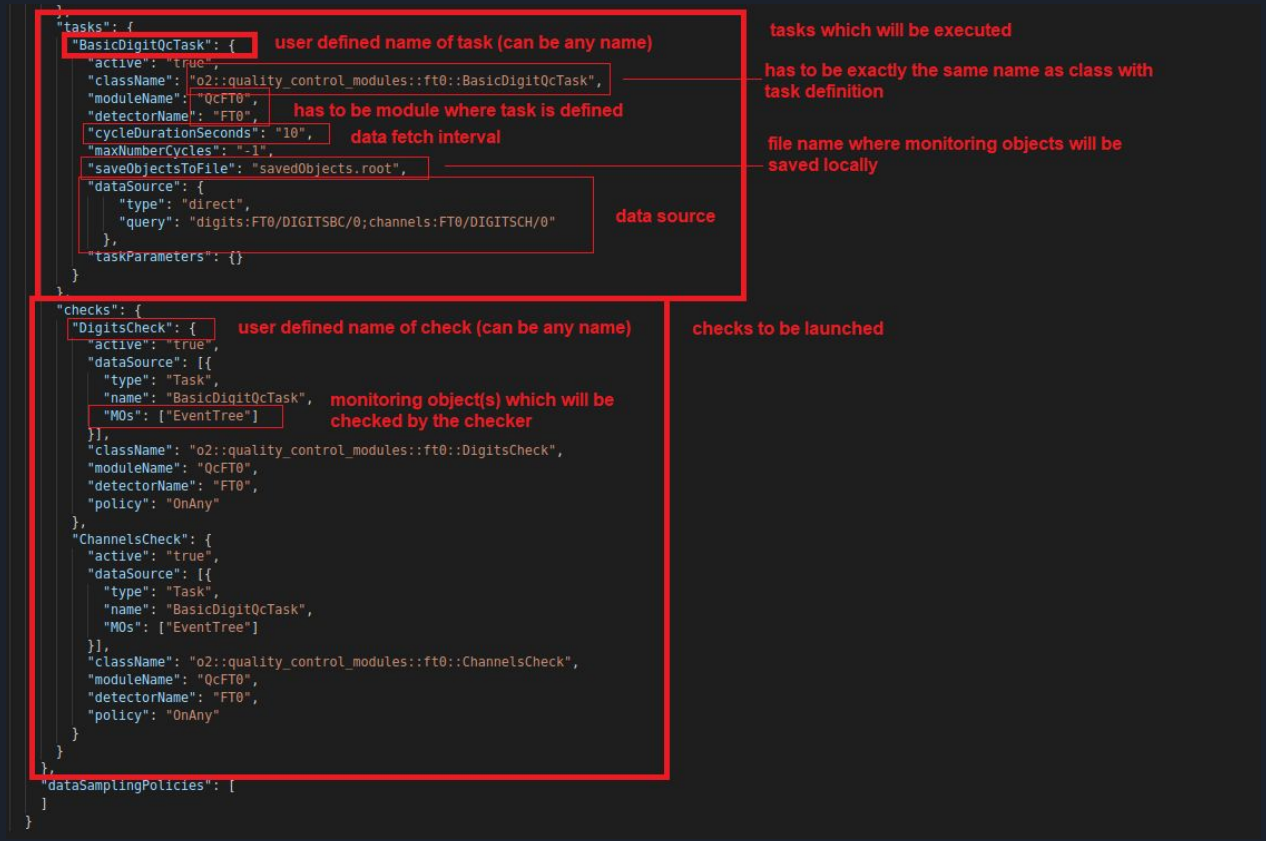
- QC checker - which receives QC Object generated in and performs defined checks
src/ChannelsCheck.cxx
src/DigitsCheck.cxx
include/FT0/ChannelsCheck.h
include/FT0/DigitsCheck.h

Single node workflow config

One needs to define proper config file to execute workflow in a proper way. It is required to define:
- data source
- sampling policy (or not)
- tasks to be executed
- checkers to perform checks
- monitoring objects as an input to checker
- information needed to establish connection with database



Single node workflow config



Single node workflow execution

To execute defined workflow one should execute following instruction:
o2-qc-ft0-data-producer | o2-qc --config json://[PATH_TO_CONFIG] -b

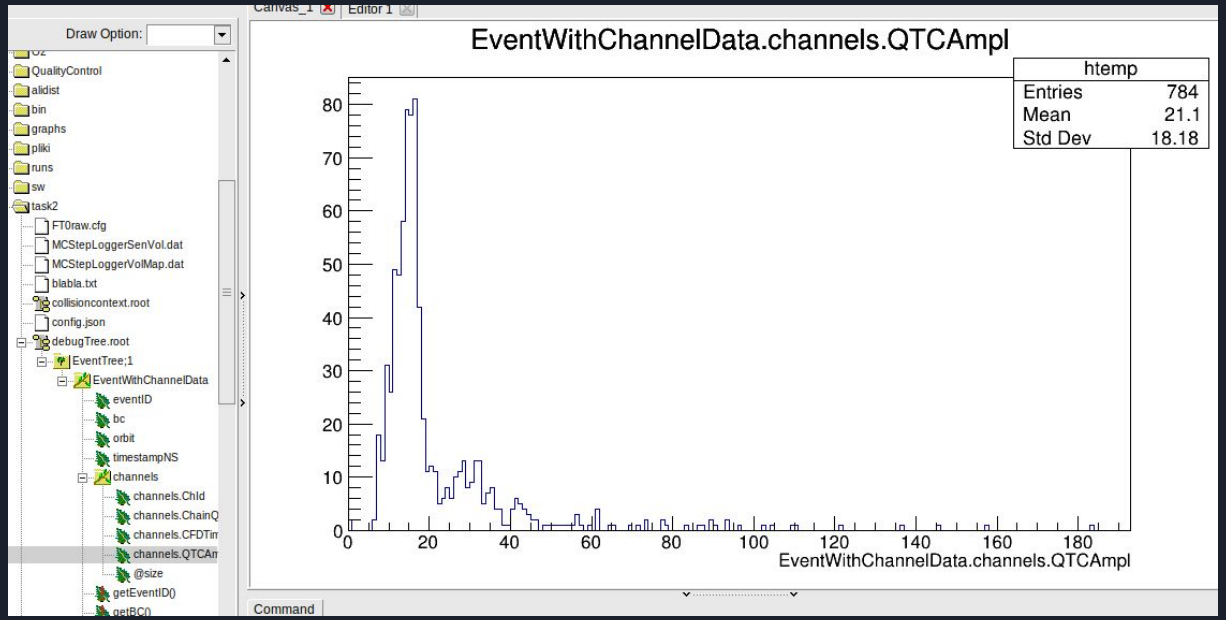
First element is a mentioned producer as an executable program

Second element is a pipe which means that output of first element will be redirected to next element in pipeline as an input

Last element is main qc program which dynamically creates instances of tasks / checks (all workflow topology) based on provided config, input data for tasks should be provided in first executable in this line (o2-qc-ft0-data-producer)

-b option to avoid GUI creation

Single node workflow result



Checker: DigitsCheck

Detector: FT0

Quality Name: Good

Quality Lv.: 1

Inputs: QC/BasicDigitQcT-mo/0

User Metadata: -

Checker: ChannelsCheck

Detector: FT0

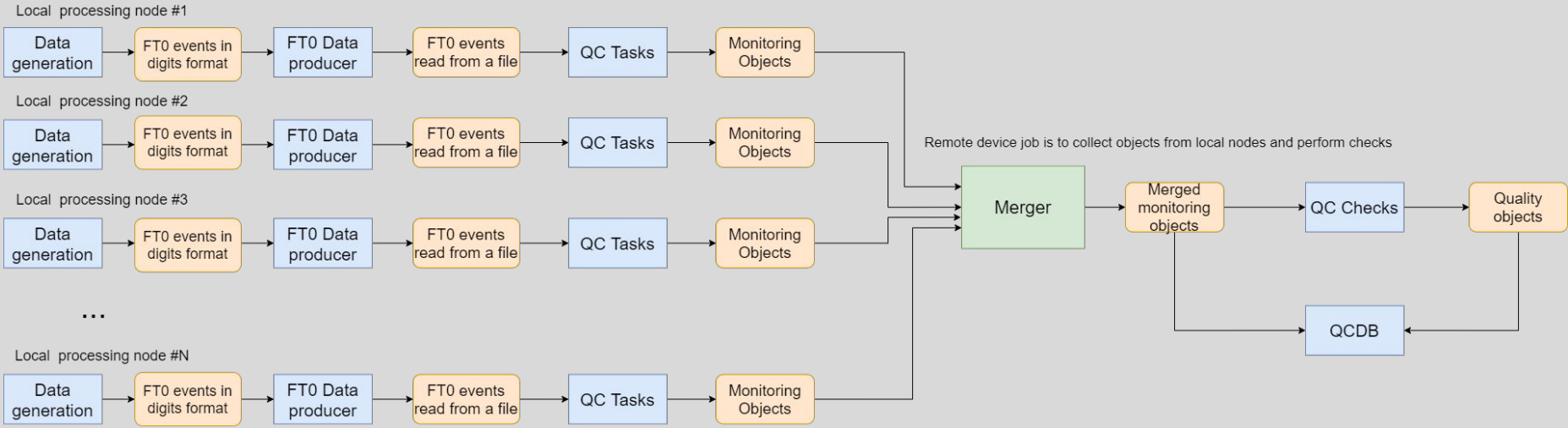
Quality Name: Good

Quality Lv.: 1

Inputs: QC/BasicDigitQcT-mo/0

User Metadata: -

Multiple node workflow



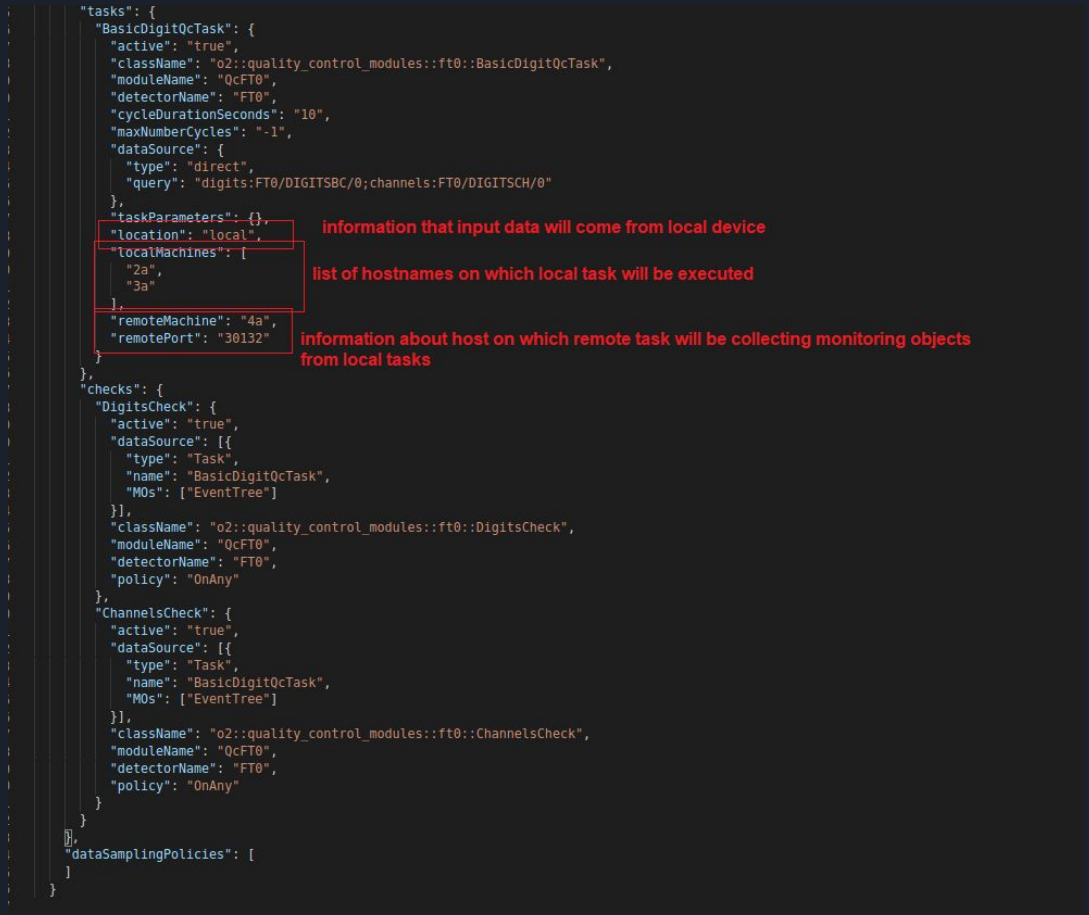
Multiple node workflow implementation

The same code as for single node workflow

Multiple node workflow config

Almost the same as in single node workflow config with some additional information

It is also possible to define remote task - input data will be collected from remote device



Multiple node workflow execution

On hosts defined as localMachines (1):

o2-ft0-data-producer | o2-qc --config json://[PATH_TO_CONFIG] --local --host [HOST_NAME] -b

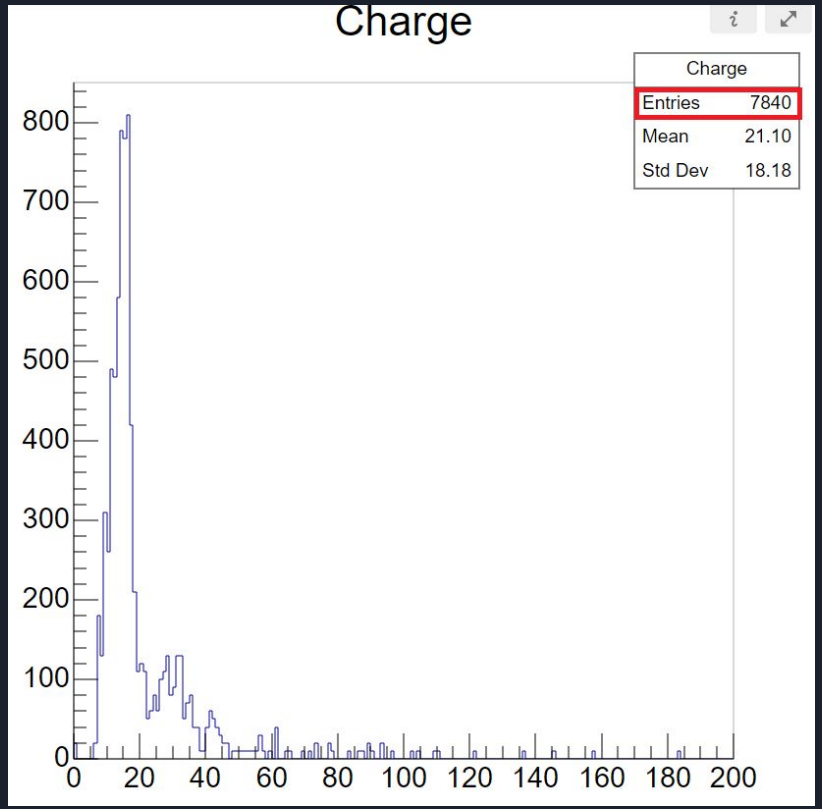
On host defined as remote machine (2):

o2-qc --config json://[PATH_TO_CONFIG] -b --remote

One should run remote workflow first (2), to be able to collect data from local nodes

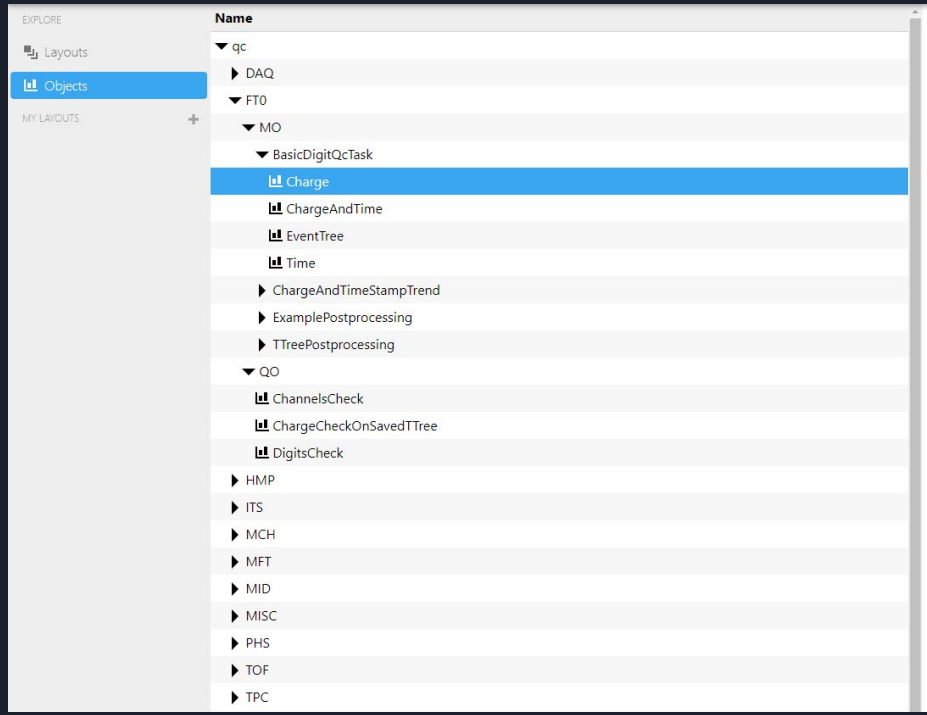
Multiple node workflow result

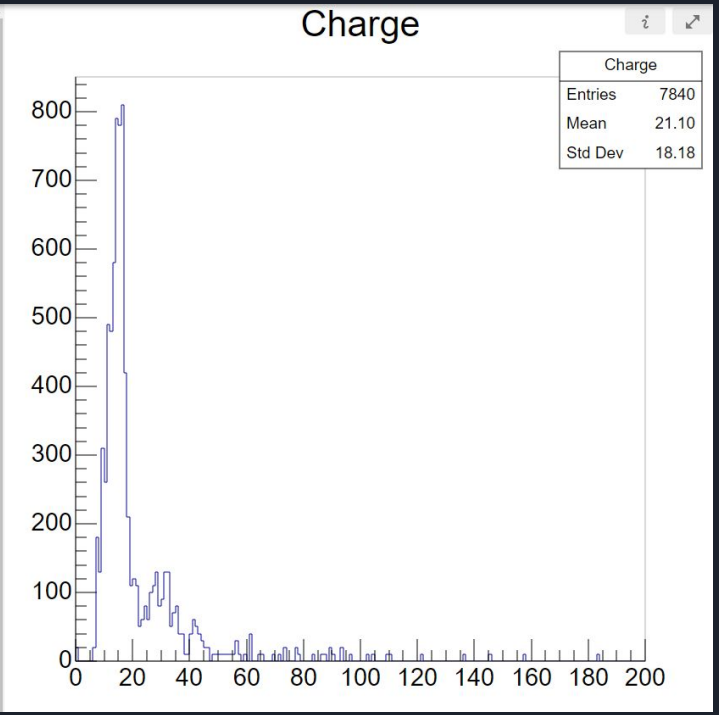
Histograms, trees merged into one bigger instance



Online objects visualization

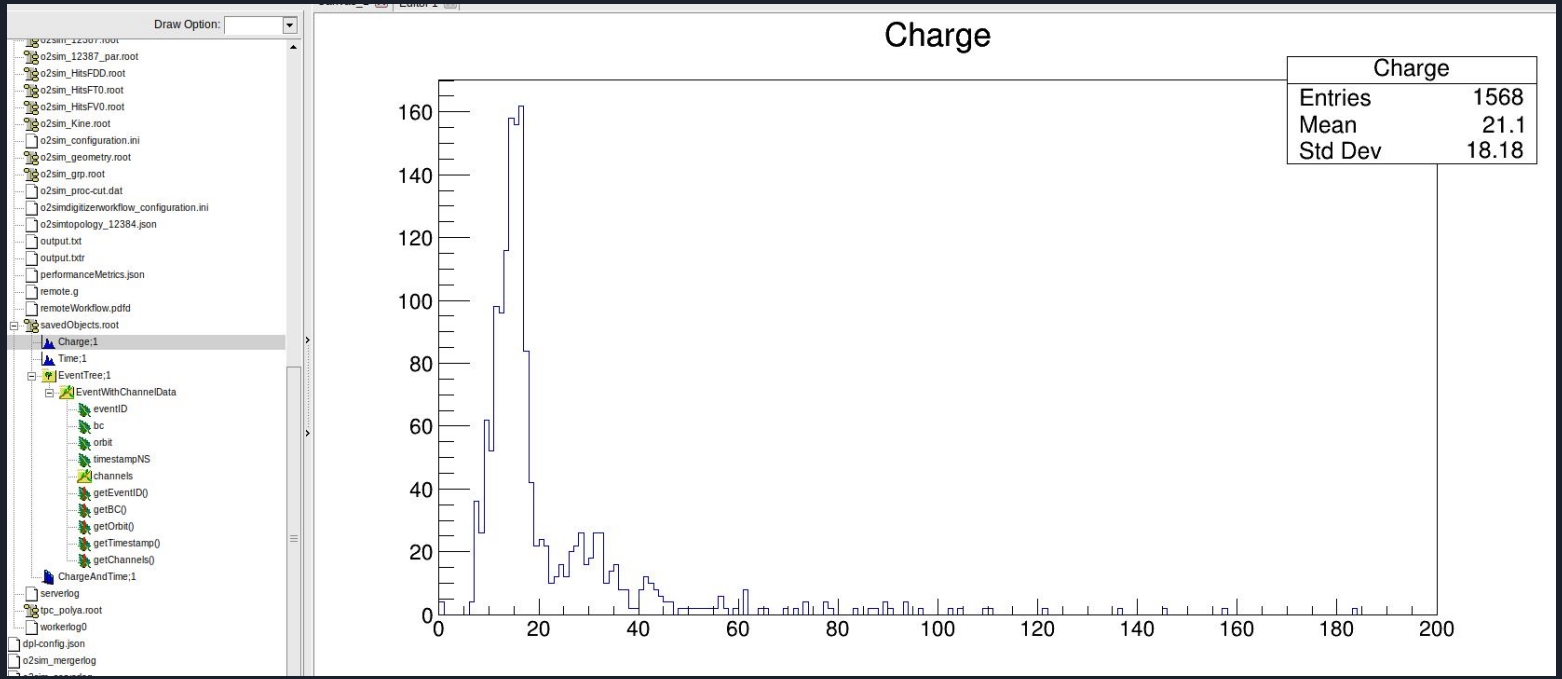
Used database has prepared front-end service, where one can browse quality and monitoring objects stored in database. https://qcg-test.cern.ch/?page=objectTree
Great compatibility with histograms, but lack of TTree visualization feature



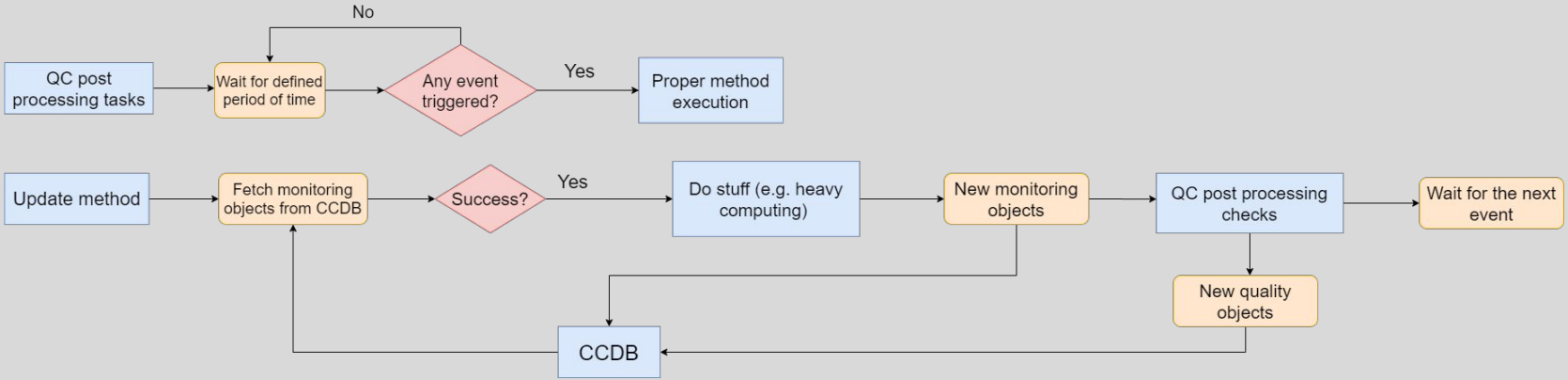


Local objects visualization

Quality objects such as TTrees cannot be visualized via qcdb front-end (yet). One can save all quality objects to one root file and then read in via TBrowser (ROOT). To save those objects it is required to add property saveObjectsToFile: [NAME_OF_OUTPUT_FILE] (check single node workflow config slide)



Post processing workflow



Post processing workflow implementation

Post processing task - waits for proper function trigger
src/TreeReaderPostProcessing.src
include/FT0/TreeReaderPostProcessing.h

Post processing workflow config

```
  "checks": {
    "ChargeCheckOnSavedTTree": {
      "active": "true",
      "className": "o2::quality_control_modules::ft0::MergedTreeCheck",
      "moduleName": "QcFT0",
      "policy": "OnAny",
      "detectorName": "FT0",
      "dataSource": [{
        "type": "PostProcessing",
        "name": "TTreePostprocessing",
        "MOs": ["ChargeHistogram"]
      }]
    }
  },
  "postprocessing": {
    "TTreePostprocessing": {
      "active": "true",
      "className": "o2::quality_control_modules::ft0::TreeReaderPostProcessing",
      "moduleName": "QcFT0",
      "detectorName": "FT0",
      "initTrigger": [
        "once"
      ],
      "updateTrigger": [
        "once"
      ],
      "stopTrigger": [
        "once"
      ]
    }
  }
}
```

type has to be PostProcessing in this case

name of object(s) in the database that one wants to monitor and connect with function triggers

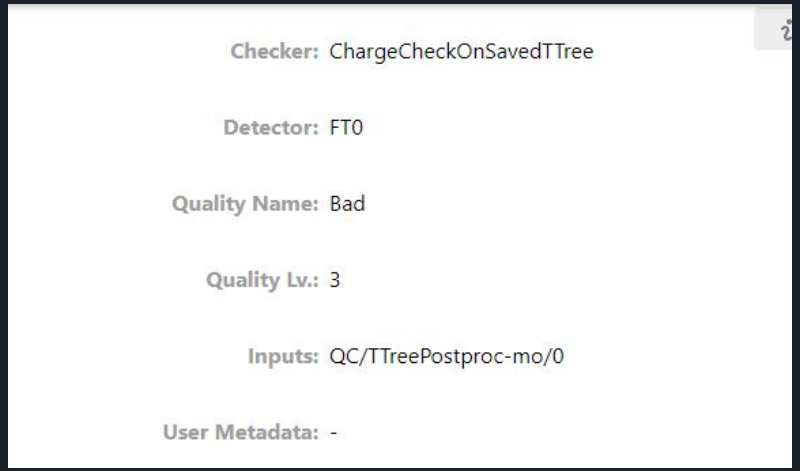
definition of function triggers - each trigger is connected to definition of function

Post processing workflow execution

After executing following line task is waiting for any function trigger
o2-qc -b --config json://[PATH_TO_CONFIG]

To trigger update function, one should run for example single node workflow - ChargeHistogram object update will trigger execution of update function and checks will be performed

Post processing workflow results



Dispatcher

Dispatcher - DPL located between producer and QC task which is responsible for data sampling. Can be defined in workflow config.

```
},
"dataSamplingPolicies": [
  {
    "id": "readout",
    "active": "true",
    "machines": [],
    "query" : "readout:ROUT/RAWDATA",
    "samplingConditions" : [
      {
        "condition": "custom",
        "moduleName": "QcFT0",
        "className": "o2::quality_control_modules::ft0::BasicCondition",
        "fraction": "0.1",
        "seed": "1234"
      }
    ],
    "blocking": "false"
  }
]
}
```

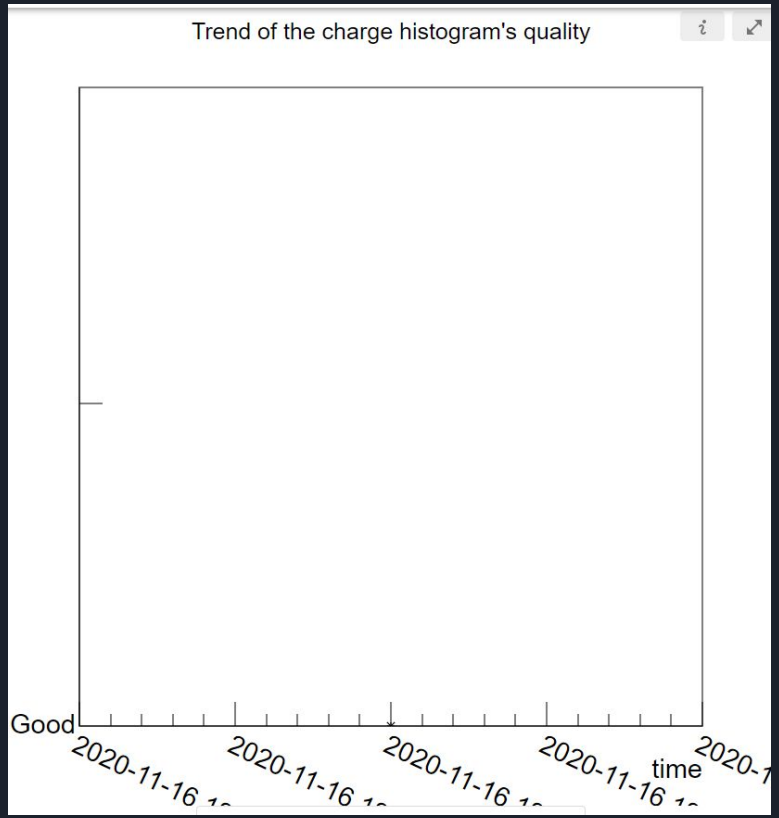
type of data - in this case raw data from readout

user defined condition class, one can also use predefined such as random

arguments for sampler (class defined above)

Post processing reductor usage

One can use reductor mechanism to monitor metrics such as mean value of histogram for each portion of data in a function of time or even quality objects generated by checkers.



Raw data usage

Just to point out that one can use raw data as an input for QC task using https://github.com/AliceO2Group/Readout/