raw -> RData reprocessing, v2

Vladislav Balagura
LLR – Ecole polytechnique / IN2P3 / CNRS
10 Dec 2015

- All runs have been reprocessed with the code similar to online_monitor.R (its *load.raw(file)* function)

- Results are stored in *.RData files and are available in CASTOR in the same place as RAW and ROOT data:
  /castor/cern.ch/user/b/balagura/siw_ecal/sps_tb_2015/,
  tarballs:
  runs_001_099_Rdata_v2.tgz, …, runs_400_499_Rdata_v2.tgz,

  see instructions how to access in
  https://twiki.cern.ch/twiki/bin/view/Main/SiWECALAnalysisDataAccess

- Not triggered data (**pedestals** data.table) are stored without prescaling in separate files *_pedestals_v2.Rdata together with **pedestal.means** (mean pedestal positions per SCA).
- Triggered data are in *_v2.Rdata files (**hits, ev, ev.chip** data.tables)

- No attempt is made to correct for the negative pedestal baseline shift in “busy” events (old measurements: pedestal moves by -0.4% * total chip charge). Plan: study pedestal stability in new board and then make a correction (may be in reprocessing v3).

- To avoid memory problems, files >100 MB are split (at spill boundaries) in 50..100 MB slices; they are called *_part1,2,3,4_*.Rdata.
  C++ code for splitting: **raw/split_raw.C** *(<file> <slice_size>),*
  available in https://github.com/balagura/online_monitor,
  returns pairs of (spill, byte offset) of proposed split points.

- R code is in the next slides, on Indico and Twiki pages:
  https://twiki.cern.ch/twiki/bin/view/Main/SiWECALAnalysisDataAccess

R code of reprocessing v2

```r
library(data.table)

TB.dir <- '~/data/tb-cern/2015'

files <- data.table(dir=list.dirs(TB.dir))[, list(file=list.files(dir,pattern='.*_by_dif[0-9]\\.raw$')), keyby=dir]
files[,dif:=as.integer(sub('^run_[0-9]+_by_dif([0-9]+)\\.raw$','\\1',file))]
files[,run.txt:=sub('^run_([0-9]+)_by_dif[0-9]+\\.raw$','\\1',file)]
files[,run:=as.integer(run.txt)]

files[,n.difs:=.N,by=dir]
files[,process:=ifelse(n.difs==4 & dif==0, FALSE, TRUE)] # in the beginning, there were 4 DIFs, and DIF 0 was noisy.
## It was then removed for runs > 76. This noisy DIF 0 is not processed.
files[,dif.out:=ifelse(n.difs==4, dif-1, dif)] # To keep the same DIF numbering, the working DIFs are called 0,1,2 always,
## even before noisy DIF 0 was removed (and these DIFs were, in fact 1,2,3).

online.monitor.dir <- '~/c/ecal/online_monitor'
map <- fread(paste0(online.monitor.dir,'/fev10_chip_channel_x_y_mapping.txt')) # data.table with chip:channel <-> X-Y mapping
map[,i:=channel]

pedestal.suppression <- 1

files[process==TRUE, {
    inp <- paste(dir,file,sep='/')
    out <- paste0(dir,'/run',run.txt,'_dif',dif.out)
    if (file.exists(paste0(out,'_part1_v2.RData')) == FALSE) { # file.info(out)$size == 0
        file.size <- file.info(inp)$size # in bytes
        cat('Processing file',inp,' (size =',file.size,'B)\n')
        n.slices <- ceiling(file.size/(100*1024^2))
        slice.size <- ceiling(file.size / n.slices) # split >100 MB files into (approximately) equal slices (eg. 100.0001 MB is
        ## divided into two 50 MB), the slices may have 50...100 MB
        ## in case of rounding: do not make more slices, ie. choose larger slice size
        slices <- data.table(read.table(pipe(paste0(online.monitor.dir,'/raw/split_raw ',inp,' ',slice.size)),
                                        col.names=c('acqStart','offset'), colClasses='integer', comment.char=''))
        slices[,acqEnd:=c(acqStart[2:nrow(slices)],0)] # sets acqEnd to first acq from the next slice or to 0 for the last slice
        slices[,n:=1:nrow(slices)]
        df.names <- c('acq','bx','sca','chip','i','adc','trig')
        df.classes <- c(rep('integer',6), 'logical')
        slices[, {
            cat(' part',n,'\n')
            hits <- data.table(read.table(pipe(paste0(online.monitor.dir,'/raw/raw ',inp,' high_gain_triggers ',
                                                      pedestal.suppression,' ',offset,' ',acqEnd)),
                                          col.names=df.names, colClasses=df.classes, comment.char=''),
                               key='acq,chip,sca,bx')
            if (nrow(hits) > 0) {
                out.hits      <- paste0(out,'_part',n,          '_v2.RData')
                out.pedestals <- paste0(out,'_part',n,'_pedestals_v2.RData')

                hits[,bx.cor:=cumsum( {
                    dbx = c(0,diff(bx))
                    dbx < 0 | ( dbx == 0 & c(0,diff(sca))>0 )
                })*4096+bx, by=list(acq,chip)] # For each spill,chip: sort in SCA: bx.cor = bx + N*4096,
                ## where N - number of detected oveflows,
                ## ie. number of times when BX goes down, eg. maximally: from 4095 to 0.
## Note, if bx jumps by >4096 crossings, this is not correct (but this is the best one can
## do with only 12 bits for BX).
## dbx < 0 | ( dbx == 0 & c(0,diff(sca))>0 ): treat a jump by 4096 BXs as a special case:
## in this case BX stays the same (dbx==0) but SCA changes

## bx.cor correction is done per chip; BX in different chips may still be compared only modulo(4096):
## eg. in the same event the jump may be detected in one chip, but not in the other. Ie. across chips
## one uses bx (== bx.cor modulo(4096)), within the chip - bx.cor.

ev <- hits[,list(n.trig=sum(trig)), keyby=list(acq,bx)]  # Find retriggers across all chips in dif
ev[,bx.group:=cumsum( c(0,diff(bx)) > 4 ), by=list(acq)]  # c(0,diff(bx)) = distances to previous BX;
## nice trick to group bxid's differing by at most 3 ("successive")
ev[,`:=`(nbx=.N, ibx=1:.N), by=list(acq,bx.group)]   # finally, number of "successive" BXs in each group,
## ibx>1 means retriggerings

## In FEV10, sometimes there are events with negative signals (eg. ADC==4). Pedestals in such events
## are shifted. To avoid errors, pedestals are calculated only if there is no any channel with "negative"
## signal in the same event.
## Algorithm: first, find approximate pedestal values (biased by "negative" signals) from untriggered and not
## retriggered data per chip, channel, SCA.
## Then, take a median pedestal inside every chip and, finally, find the minimum across the chips.
## In the following "unbiased" pedestal calculation, only entries above (0.75 * this value) will be considered.
## Note, for the chip with the minimal pedestals, this assumes that all its pedestals > 0.75 * (chip average).
negative.adc.threshold <- 0.75 * min(
                                     merge(hits,
                                           ev[,list(acq,bx,ibx)], # add ibx to hits to require ibx==1
                                           by=c('acq','bx'))
                                     [trig==FALSE & ibx==1
                                      ][, list(ped=as.double(median(adc))), by=list(chip,i,sca)
                                        ][,list(ped=median(ped)),by=list(chip)]$ped)
## cat('Negitive threshold:',negative.adc.threshold, '\n')

## same per chip (name with .chip), events for one chip are sorted first in SCA
ev.chip <- hits[,list(n.trig.chip=sum(trig),
                      n.neg.trig.chip=sum(adc<negative.adc.threshold)), keyby=list(acq,chip,sca,bx.cor,bx)]
ev.chip[,bx.group.chip:=cumsum( c(0,diff(bx.cor)) > 4 ), by=list(acq,chip)]
ev.chip[,`:=`(nbx.chip=.N, ibx.chip=1:.N), by=list(acq,chip,bx.group.chip)]

## do the same, but taking into account only events with at least one trigger
## (in SKIROC, if trigger arrives at the clock edge, both BX and BX+1 are written, but the latter without
## any triggers, if there is no retriggering)
any.event.wo.trig <- any(ev.chip$n.trig.chip == 0)
if (any.event.wo.trig) {
    ev.trig <- ev.chip[n.trig.chip>0, list(acq,chip,sca,bx.cor,bx)] # remove events wo triggers, name with suffix .trig;
    ## note: this automatically means per chip
    ev.trig[,bx.group.trig:=cumsum( c(0,diff(bx.cor)) > 4 ), by=list(acq,chip)]  # find retriggers per chip AND
    ## only for events with triggers
    ev.trig[,`:=`(nbx.trig=.N, ibx.trig=1:.N), by=list(acq,chip,bx.group.trig)]
    ev.chip <- ev.trig[ev.chip]
} else ev.chip[,`:=`(bx.group.trig=bx.group.chip, nbx.trig=nbx.chip, ibx.trig=ibx.chip)]

ev.chip <- merge(ev, ev.chip, by=c('acq','bx'), all=TRUE)

## find pedestals from trig==FALSE & ibx==1 & (no channel in the same chip with
## ADC < negative.adc.threshold), otherwise hits$a will be set to NA
pedestals <- merge(hits[trig == FALSE], ev.chip[,list(acq,chip,sca,ibx,n.neg.trig.chip)],
                   by=c('acq','chip','sca'))
pedestals <- pedestals[,trig:=NULL]
pedestals <- pedestals[ibx == 1 & n.neg.trig.chip == 0][,ibx:=NULL][,n.neg.trig.chip:=NULL]
pedestal.means <- pedestals[,list(ped=as.double(median(adc))), keyby=list(chip,i,sca)]
```

```
                hits <- merge(hits[trig == TRUE], ev.chip, by=c('acq','chip','sca','bx.cor','bx'))

                ##        hits <- hits[ibx==1] # remove retriggers

                hits <- merge(hits, map[,list(chip,i,x,y)], by=c('chip','i'), all.x=TRUE)
                hits <- merge(hits, pedestal.means, by=c('chip','i','sca'),all.x=TRUE)
                hits[,a := as.double(adc) - ped]

                setkey(pedestals, chip, i, sca, acq, bx.cor)
                setkey(hits, acq, chip, bx.cor, i)
                setkey(ev.chip, acq, chip, bx.cor)
                setkey(ev, acq, bx)

                save(hits, ev, ev.chip,         file=out.hits)
                save(pedestals, pedestal.means, file=out.pedestals)

                rm(hits,pedestals,ev,ev.chip)
                gc(reset=TRUE)
            }
            NULL
        }, by=n]
    }
    NULL
}, by=list(file,dir)]
```