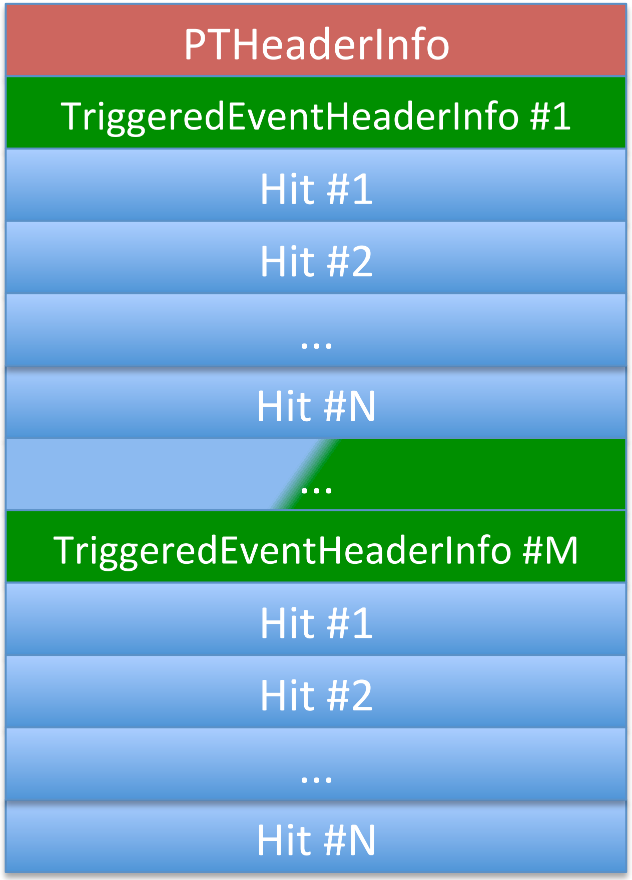
Post trigger file dataformat

Proposal from C. Pellegrino

for the KM3NeT-Italy TriDAS

The actual PT file format

Every PostTrigger (PT) file has the following structure:



PTHeaderInfo

TriggeredEventHeaderInfo #1

Hit #1

Hit #2

…

Hit #N

…

TriggeredEventHeaderInfo #M

Hit #1

Hit #2

…

Hit #N

s_PTGeneralHeaderInfo + array of triggered events

The header struct must contain all the relevant informations for the off-line analysis and for the DAQ group.

Actual fields of PTHeaderInfo (to be discussed)

Field type	Size (bit)	Name	Description
uint	32	VersionPTFile	Data format release
uint	32	RunNumber	The run number
uint	64	StartTime5ns	Start time of the run in 5ns units
uint	32	MaxFileSize	the maximum size for a pt-file
uint	32	EffectiveFileSize	the effective size of this very pt-file
uint	32	FileNumber	File counter for the run
uint	32	TotEventsInFile	
uint	32	NPMTperTower	the total number of PMTs in the tower
uint	32 x NPMTTOWER_MAX	TimeOffset	[ns] Time offsets corrections
uint	32 x NPMTTOWER_MAX	PedestalA	*1000 and converted from float to int - pedestal to be subtracted to odd samples
uint	32 x NPMTTOWER_MAX	PedestalB	/ *1000 and converted from float to int - pedestal to be subtracted to even samples
uint	32 x NPMTTOWER_MAX	QThreshold	*1000 and converted from float to int - minimum requested (and calibrated) charge to be accepted for a trigger

Use a struct similar PTHeaderInfo

Pros: well defined format, fast and optimized for memory.

Cons: dedicated programming is needed to perform read and write operations, difficult to extend mantaining backward compatibility.

The experience of phase 2 says that request of new fields can arrive from the analysis group.

Proposed solution

Substitute the PTHeaderInfo with a ASCII-based header (i.e. JSON)

- Pros: extensibility, human readable (using shell programs like head, less, strings etc...) and debugged.
- Cons: (negligible) overhead in the file size and parsing time.

Local log for the Datacards

Proposal from C. Pellegrino

for the KM3NeT-Italy TriDAS

- Problem: we want a log of the datacard of every run that is independent from the DB.
- Proposal: use a git repo hosted in the TSC home. The TSC should handle the commit and push operations

This concept explained in shell commands

At the beginning of each run:

```
$ cp datacard.ini ${REPO_ROOT}/repo/
$ cd ${REPO_ROOT}/repo/
$ git add datacard.ini
$ git commit –m “run number ${run_number} $(date)
$ git push
$ cd -
```

Advantages

- All the needed tools are already written, easy to use and install
- There are free solution to publish/backup the repo (bitbucket, github)
- The repo is already compressed
- We can use all the features of git: log, diff, tag, etc...