# Trigger CPU performances

**T. Chiarusi[1], L. A. Fusco[1,2] F. Giacomini[3] and M. Manzali[3,4]**

**[1] INFN - Sezione di Bologna , [2] Dipartimento di Fisica e Astronomia dell'Università di Bologna [3] INFN - CNAF, [4] Università degli Studi di Ferrara**

## Introduction

In this paper we report the studies done for determining the computational performances required by the Trigger CPU in the framework of the KM3NeT-Italian 8-Towers detector Trigger and Data Acquisition System (TriDAS).

The work consisted of:

1. Simulation of atmospheric muons events with optical poissonian background in the KM3NeT-Italian 8-Tower detector.
2. Implementation of the event readout codes and the basic trigger algorithm for the reduction of the events.
3. Tests with benchmarked server at CNAF

## 1 Simulating the hits

The procedure was started once the Physics WorkingGroup provided us with a 8-Towers detector file [1] .

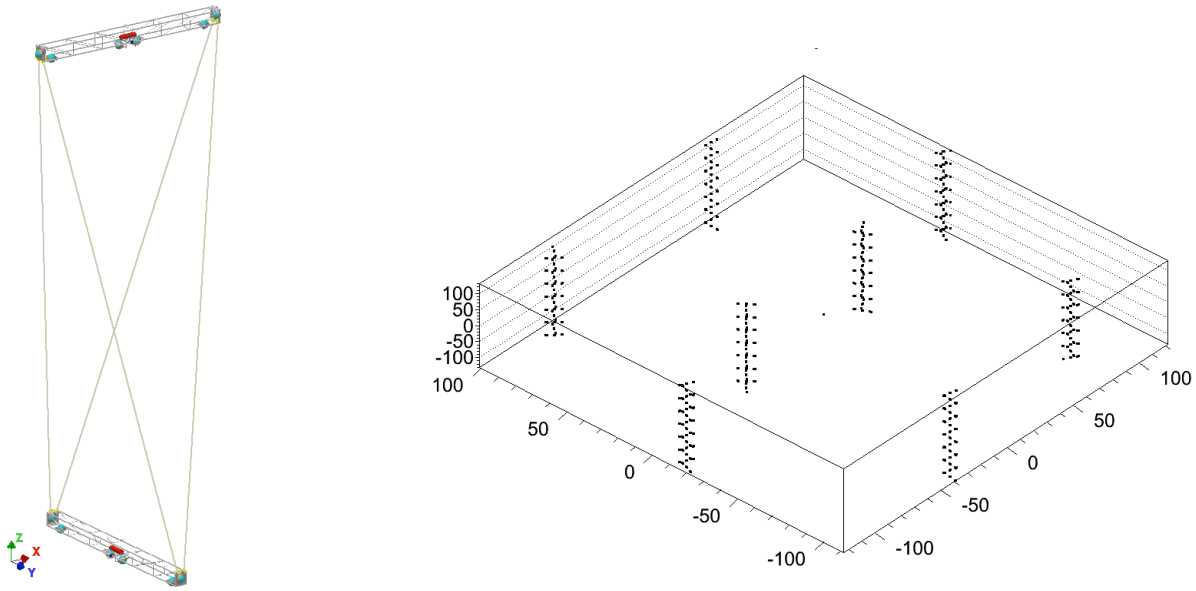



Figure 1. left: an excerpt of the 14 floor tower; right: the 3D arrangement of the towers.

It is consistent with the following detector specs: 8 towers ; 14 floors per tower; 6 PMTs per floor (Fig. 1 left).

The standard NEMO simulation chain was used [2] such as sketched in Figure 2. The aim was to simulate events of atmospheric muons + background.

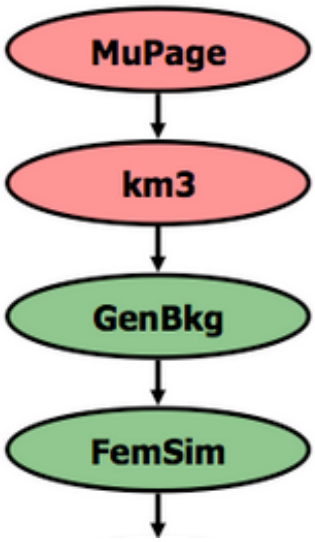



Figure 2. the progressive MonteCarlo processing phases.

1. MUPAGE, the atmospheric muon generator [3] , [4], which generates a number of muons on the surface of a given "can" surrounding the detector. The following parameterization was used:

```
physics:      HEMAS-DPM          07.01
propag:       MUSIC seawater     02.03
simul:        MUPAGE             03.05-100416    130920 1203
seed:         MUPAGE-MT19937     3   1001   0   0
can:          -224.212    330.204    431.260
seabottom:    3430.
cut_in:       1.    500000.    -1.000000    -0.087156
norma:        0      1000000
livetime:     5.66e+03     7
```

Note that the generated events ($10^6$ events on the given "can") correspond to a live-time of 5660 seconds (about 1 hour and 35 minutes).

2. KM3, the program which generates the hits on the detector PMTs, according to the detector file, and (must of all) according to the parameterization of the optical properties of the water and of the Optical Modules (i.e. the angular acceptance of the PMTs, the OM's transparency etc.).

A very important note: not all the muons generated by MUPAGE give hits on the PMTs. Actually this depends on the size of the "can". The bigger the "can" is, the less percentage of muons give hits in the detector.

The output file produced by KM3 will contain 2 kinds of events:

- good events with direct Cherenkov hits, i.e. coming from the muon;

- fake events, practically empty. For sure without the direct Cherenkov hits; indeed with a possible set of hits from scattered photons or from secondary particles.

The production of fake events can be toggled on/off in KM3. In this case we decided to keep them, since they were used to produce with the application of GENBKG program the "background" events.

3. GENBKG, the program which adds the optical background to each event (good or fake) determined by KM3. It applies a poissonian background related to a constant single rate $\nu_{bkg}$ properly tuned to simulate the convolution of the optical contribution of $^{40}K$ decays and bioluminescence.

As it will be reported further in the text we considered two kind of $\nu_{bkg}$ : 50 kHz and 110 kHZ.

4. FEMSIM, the program which simulates the electronic sampling.

We left the default settings, in particular a s.p.e is rated 8 pC. It produces an output file with the hits written according to a format similar to what used in the real online DAQ.

## 2 Five approaches to prepare the TimeSlices

The hits contained at the FEMSIM level are organized into a discrete number of events. There is not any continuous flow of hits, as it is in the real case of the online DAQ. This difference is crucial . In order to estimate the necessary resources to manipulate the hits and to apply the trigger algorithms we need to prepare the available data in a way that is as much similar as the real case. Moreover, in the online DAQ we don't know when a muon event occurs. It is the very trigger algorithm to cut out events from the continuous data-stream. In the following we explain how we manipulated the FEMSIM events in order to prepare a more realistic set of data.

The online TriDAS operates thanks to the concept of TimeSlice. The TimeSlice abstractly refers to a particular set of hits, all from the whole detector and all occurred during a certain interval of time of a given duration. According to this, before that TCPUs could apply the trigger algorithms, the continuity of time-flow is sliced into a series of subsequent TimeSlices. This is done inside the HitManager servers. Refer to [5] for a detailed description of TriDAS. It is enough to know here that all the TriggerCPU processes act individually on different TimeSlices. In the present context of the off-line analysis reported in this paper, as a preamble for the application of the trigger algorithm, we needed to reassemble the FEMSIM file. All the events were opportunely combined into groups, each one with the following characteristics:

a) good (muons) a fake (background) events were mixed together according to a prefixed balance (see paragraph 2.1);

b) the summation of the event durations was approximatively equal to the TimeSlices duration (200 ms). From the FEMSIM file, considering both good and fake events, the average event duration turned out to be of about 6.2 $\mu$s. Being the total number of events produced by MUPAGE equal to $10^6$ , we reassembled the good and fake events within 31 TimeSlices, each of the duration of 200 ms.

Such reassembling of the events lead to writing different files, called "TimeSlice files".

## 2.1 BALANCING MUONS AND BACKGROUND

We produced 5 kind of TimeSlice files from the simulated data, each one according to the following approach:

1. the "Condensed" approach (an extremely high 25 kHz muon rate and a realistic $\nu_{bkg}$ = 50 kHz background rate per PMT);
2. the "Realistic" approach ( a realistic $\nu_{bkg}$ = 50 kHz background rate per PMT and a 27 Hz muon rate).
3. the "Hardly Realistic" approach (a realistic $\nu_{bkg}$ = 50 kHz background rate per PMT and a hard 270 Hz muon rate);
4. the "Heavy Background" approach (a heavy $\nu_{bkg}$ = 110 kHz single rate background per PMT and realistic 27 Hz muon rate);
5. the "Heavy Background + Hardly Realistic" approach (a $\nu_{bkg}$ = 110 kHz background per PMT and a 270 Hz muon rate)

## 2.2 CONSIDERATIONS ABOUT REASSEMBLING THE FEMSIM FILE

Let's follow the scheme sketched in Figure 3. In the topmost box you find some muons/muon bundle events (black dots) displaced along time (the right-arrow). Note that MUPAGE does not estimate the correct progressive occurring time for any of the muons/muon bundles it simulates. All the muons/muon bundle determine a stand alone event. The development of each event is independent with respect to the others. In any case, MUPAGE gives back a live-time estimation for the requested number of events. So we could retrieve an averaged muon event rate by dividing the number of generated events by the corresponding amount of live-time. In the present case it is about $\nu_g$ = 177 Hz. Please, note once more that $\nu_g$ increases with the "can" size.

As mentioned above, not all the muons give signals in the detector, when processed by KM3. Table 1 shows that good events release detected signals with a rate of $\nu_d$ = 27 Hz. Note again that $\nu_d$ does not increase with the size of the "can". Indeed, with the size of the can, the KM3 processing time to find out the detected events is enhanced.

In Figure 3 - middle box, blue dots represent those muons that produced good events after KM3, while with red dots the muons that produced "fake" events. The green and yellow bars represent the event duration, for good and fake events, respectively. There is a minimum requested duration of 6 $\mu$s for all events. The duration of good events could result larger due to the fact that GENBKG adds the background hits along a time window built in the following way: starting from the effective duration of the muon signals inside the detector (~ 1 $\mu$s) as it is simulated by KM3 , the total event window is retro-extended by 3 $\mu$s in the past (starting from the first muon hit) and forward-extended by 3 $\mu$s in the future (starting form the event last muon hit). For fake events, being empty of muon hits, the event duration is just 6 $\mu$s instead. It is apparent from Figure 3 that the summation of all the time intervals represented by the yellow and green bars doesn't cover the full extension of time. Indeed there are time-gaps which depend to chosen global event duration. Finally note, again from the middle box, that the good events (blue dots) occurred in a TimeSlice interval is 2.

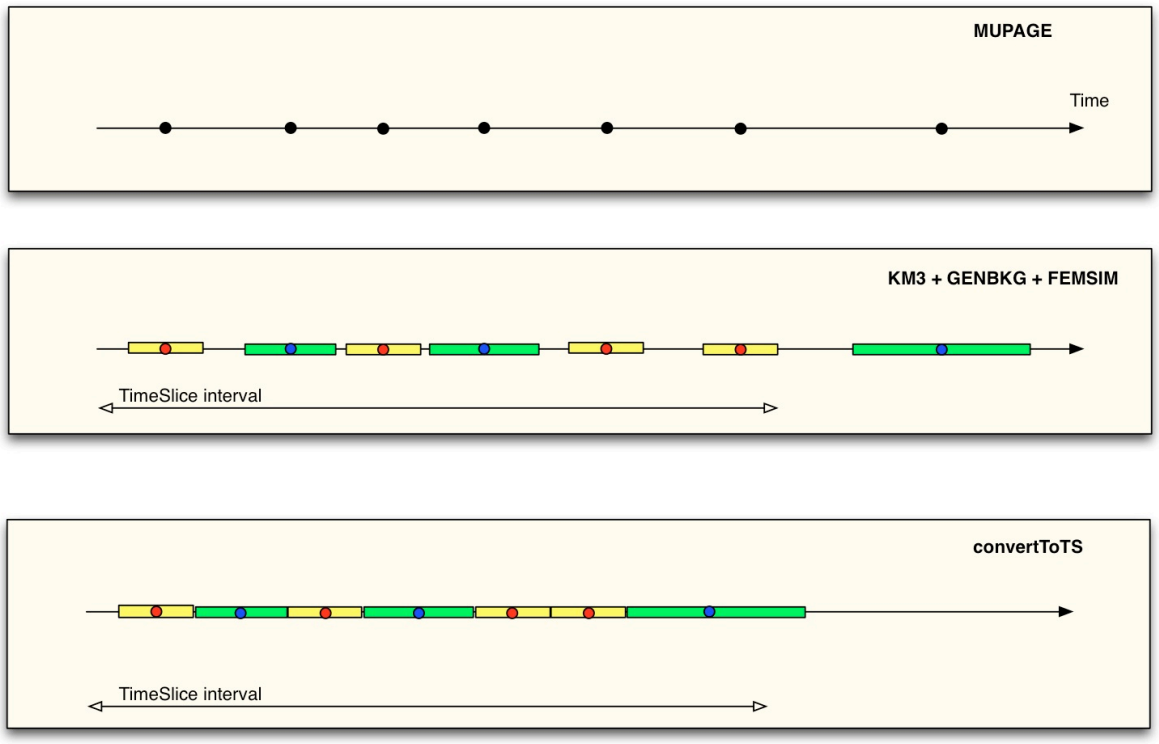



Figure 3. From the muon generation to the condensed case: the detected muon rate is enhanced due the shrinking of the events to cover a TimeSlice time interval (see text for details).

### 2.2.1 CONDENSED APPROACH

The bottommost box shows how we setup a "condensed" TimeSlice. We simply concatenated all the present events, good and fake, as they were listed in the FEMSIM output file. The progressive time of all the hits is computed as following: the time of the first hit of the next event is obtained by adding the relative hit time in the event to the time of the last hit of the previous event.

This turned out into a shrinking of the data flow, enhancing the rate of good events inside each TimeSlice. In the bottommost box of Figure 3 this effect is clearly visible with the good events occurring in a TimeSlice being now equal to 3.

In the end, the condensed approach consists of producing TimeSlices with a muon rate much larger than in the realistic case. As it will be shown further in this note, such data sample stressed the trigger algorithm more than a normal realistic data sample. Table 1 reports a summary of the events statistics and the relative rates computed with respect to a certain live-time.

| | N. ev | live-time (s) | rate (Hz) |
|---|---|---|---|
| MUPAGE | 1000000 | 5660 | 177 |
| KM3 - Good | 150894 | 5660 | 27 |
| KM3 - Fake | 849106 | 5660 | 150.0 |
| condensed TS signal | 5066 | 0.2 | 25330 |

Table 1.

By simply considering the atmospheric muon rate in the shrunk TS of Table 1, the "conservative" approach could appear fairly unrealistic: we have 25 kHz in the shrunk TS against 27 Hz expected in the real life.

In any case it is worthy to note that good and background events contain similar averaged number of hits. For the presented simulation, from all 5066 good events we get 1071856 hits, and so the averaged muon hit number per event is 212. Assuming 50 kHz of single rate per PMT, the averaged background hit number per event is 204. This imply the same processing load necessary to handle muon and background hits in the TCPU. The crucial point is that searching for causality correlations will stress more the TCPU performances when a larger fraction of muon events is present. This fact will become more clear in the section concerning the report from the trigger application. In any case we can anticipate that the larger stress is caused by the larger number of trigger seed that the trigger program must take into account.

### 2.2.2 REALISTIC AND HARDLY REALISTIC APPROACHES

Apart from the "condensed" approach, the other approaches needed a more complex data manipulation in order to mix the generated optical background with the wanted muon rate. For building-up the correct data-stream compliant with a requested number of muon events in each TimeSlice we followed the following steps:

- for each TimeSlice of duration $\Delta T$ , the number of present muons $N_\mu$ is extracted by the Poisson probability distribution $P(N_\mu, \nu_d \Delta T) = (1/N_\mu!) \exp(-\nu_d \Delta T) (\nu_d \Delta T)^{N_\mu}$ .

Note that the duration of the TimeSlice is much more bigger than the duration of a muon event (i.e. about few $\mu$s). This allow us to neglect the occurrence of muon events which are split by the end of a TimeSlice and the beginning of the subsequent one.

- $(N_\mu - 1)$ time intervals $\Delta t$ are randomly extracted by mean of the exponential law $\exp(-\nu_g \Delta t)$.

In this way we can compose the TimeSlice content by opportunely adding background events as to fill the $\Delta t$ gap between two muon events. It is that even doing so, the number of TimeSlices won't cover the original live time corresponding to the generated muons at the MUPAGE level. In the pure "realistic" case the muon rate at the detector $\nu_d$ = 27 Hz is preserved in the TimeSlice data-stream; in the "hardly" realistic approach $\nu_d$ was enhanced by a factor 10 ($\nu_d' = 270$ Hz).

Figures 4 and 5 show some statistical information about the reassembled events according to the "Realistic" and "Hardly Realistic" approaches. Refer to the figure caption for details. Note that the tails of the left plots of Figure 4, present in both distribution for muon and background only events, are due to the presence of hits from secondary particles and scattered photons which extend the event duration. Finally note in Figure 5, right plot, that the average number of hits per TimeSlice for both the two approaches is the same.

### 2.2.3 HEAVY BACKGROUND APPROACH

GENBKG was re-run getting an alternative simulation of background: the random optical noise single rate was enhanced up to 110 kHz per PMT. The intention of this simulation set was to have a TimeSlice richer of hits, so to stress the memory allocation feature of the trigger programs. Also in this case, similarly to the "realistic" line-up, we prepared two set of TimeSlices: one with a realistic muon rate of $\nu_d$ = 27 Hz ("Heavy Background"), the second with 10 x $\nu_d$ = 270 Hz (Heavy Background and Hardly Realistic) . Figures 6 and 7 shows the above discussed statistical observables. We note that the number of hits per event, and so per TimeSlice, is about a factor 2.2 larger than for the "Realistic" and "Hardly Realistic" approaches, as expected.

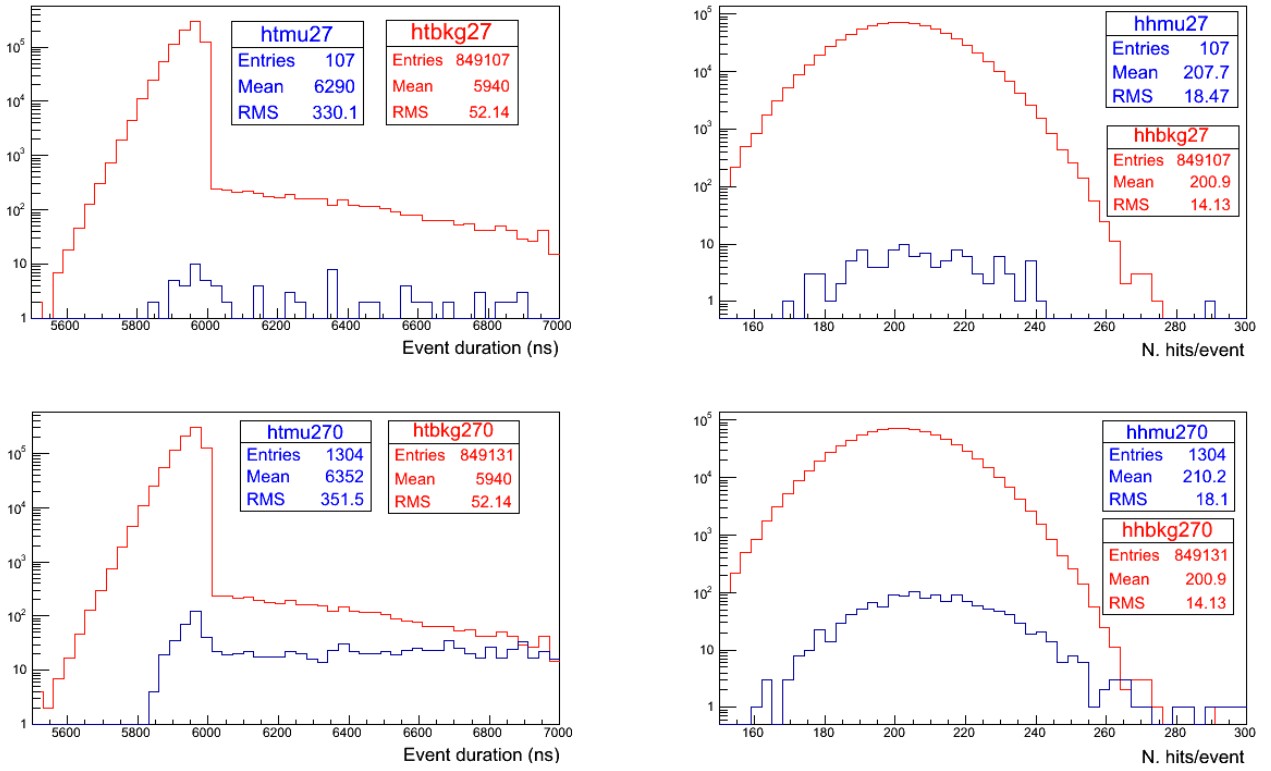



Figure 4. "Realistic" and "Hardly Realistic" ( $\nu_{bkg}$ : 50 kHz ). Muon events in blue, background only events in red; top plots according to muon rate $\nu_d$ = 27 Hz, bottom plots according to muon rate $\nu_d$ = 270 Hz.. Left column: distribution of event duration. Right column: distribution of the number of hits per event.

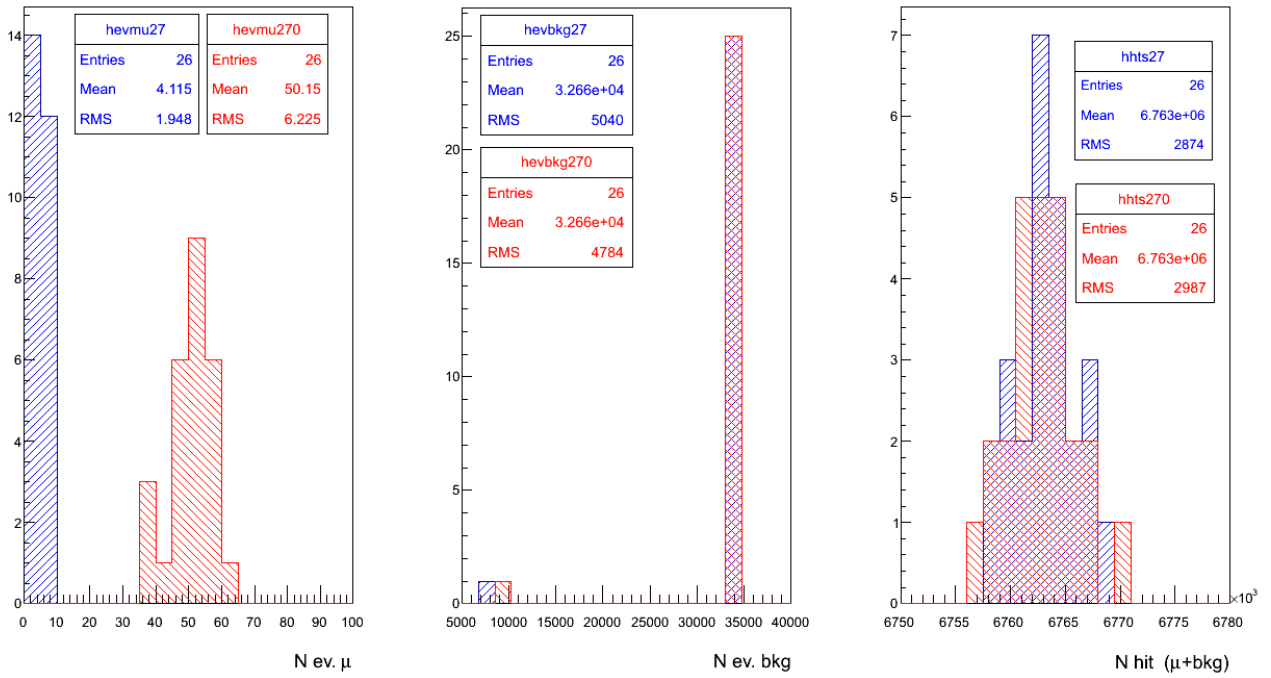



Figure 5. "Realistic" and "Hardly Realistic" ( $\nu_{bkg}$ : 50 kHz ). Muon rate $\nu_d$ = 27 Hz in blue, muon rate $\nu_d$ = 270 Hz in red. Left: distribution of number of muon events per TimeSlice. Center: distribution of number of background only events per TimeSlice. Right: distribution of hits (from muon and background events) per TimeSlices

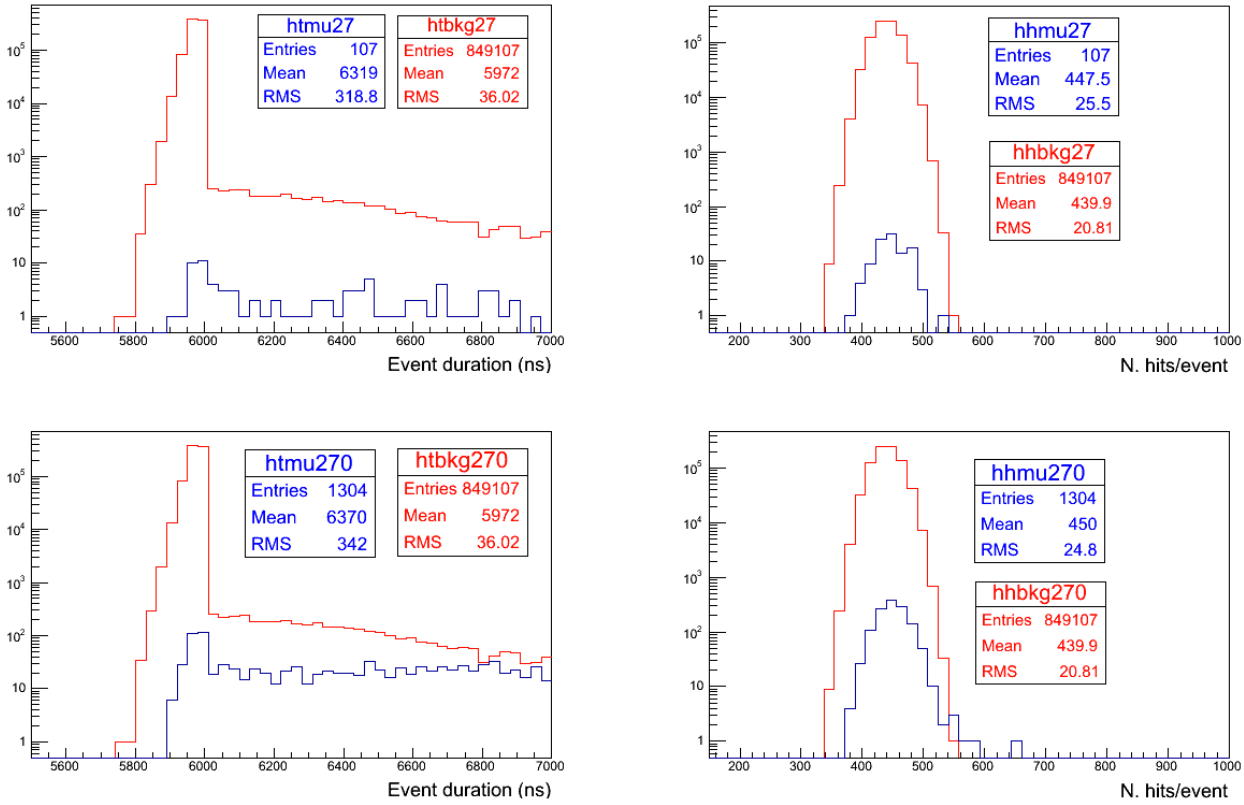



Figure 6. "Heavy Bkg" and "Heavy Bkg + Hardly Real." ( $\nu_{bkg}$ : 110 kHz ). Muon events in blue, background only events in red; top plots according to muon rate $\nu_d$ = 27 Hz, bottom plots according to muon rate $\nu_d$ = 270 Hz. Left column: distribution of event duration. Right column: distribution of the number of hits per event.

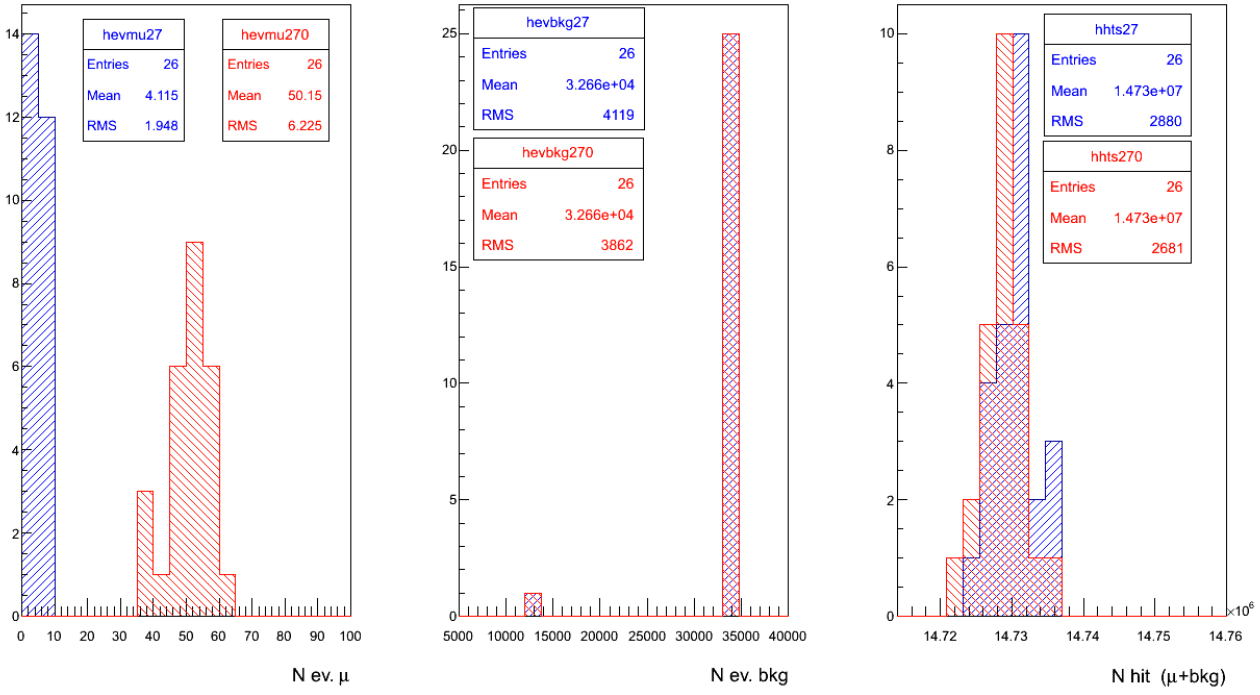



Figure 7. "Heavy Bkg" and "Heavy Bkg + Hardly Real." ( $\nu_{bkg}$ : 110 kHz ). Muon rate $\nu_d$ = 27 Hz in blue; muon rate $\nu_d$ = 270 Hz in red. Left: distribution of number of muon events per TimeSlice. Center: distribution of number of background only events per TimeSlice. Right: distribution of hits (from muon and background events) per TimeSlices

# 3 Data manipulation strategy

Three layers of data handling were set as shown in Figure 8. Starting from the FEMSIM output file the events were separated into 2 files: the good event (muons) file and the fake event (background only) file.

## 3.1 WRITING THE TIMESLICE FILES

The TimeSlices were formed by properly mixing the background events with the wanted amount of muon. For each one of the 5 approaches described in the previous section one final output file was dumped with the collection of all the 33 TimeSlices.

The production of such 5 TimeSlice files was completed on a commodity PC, running SL6, which was a different resource from the one used for the benchmark test ( see further in Section 5). On the contrary the next two steps described in the following text represented the core of the benchmarked code.

## 3.2 PARSING DATA AND MEMORY ALLOCATION

The second step consisted of parsing the TimeSlice file, reading the TimeSlice groups one by one. The parsed data were allocated into the RAM in a way useful to handle all the hits from all the PMTs belonging to a given TimeSlice.

For this purpose, the events were decomposed: all the hits related to one PMT were pushed back into a STL vector, the so call PMT vector. All the PMT vectors were then pushed back into a second STL vector, which was called the TimeSlice vector.

## 3.3 TIME SORTING, HIT CALIBRATING AND EVENT TRIGGERING

The third step was itself subdivided into three parts:

- all the hits of the parsed TimeSlice were "calibrated". Actually, such a calibration was necessarily a fake one, since neither time offsets between PMTs neither charge pedestals were included in the detector simulation. A fake constant correction was then added the hit timestamp and charge with the aim to introduce time-consuming operations. It is worthy to note that the hit timestamp was represented according to the online data format (a fine timing in units of 5 ns thanks to a simulated parity flag and a coarse timing in units of 500 $\mu$s), while the charge was measured in pC.

- first of all we created a time-sorted array containing all the hits of the TimeSlice. For quickly completing this task, a supplementary STL vector of vectors was composed, similar to the TimeSlice vector, but containing only those information of a hit used by the sorting and the trigger algorithms (i.e. the hit time and a pointer to the effective area of the memory where all the rest of information reside ).

- the last action consisted of applying the 2 trigger levels, which are described in the next section.

The actions described in paragraphs 3.2 and 3.3 were implemented in the single threading program called "TCPUSimu" which underwent the benchmarking tests, as reported in Section 5.

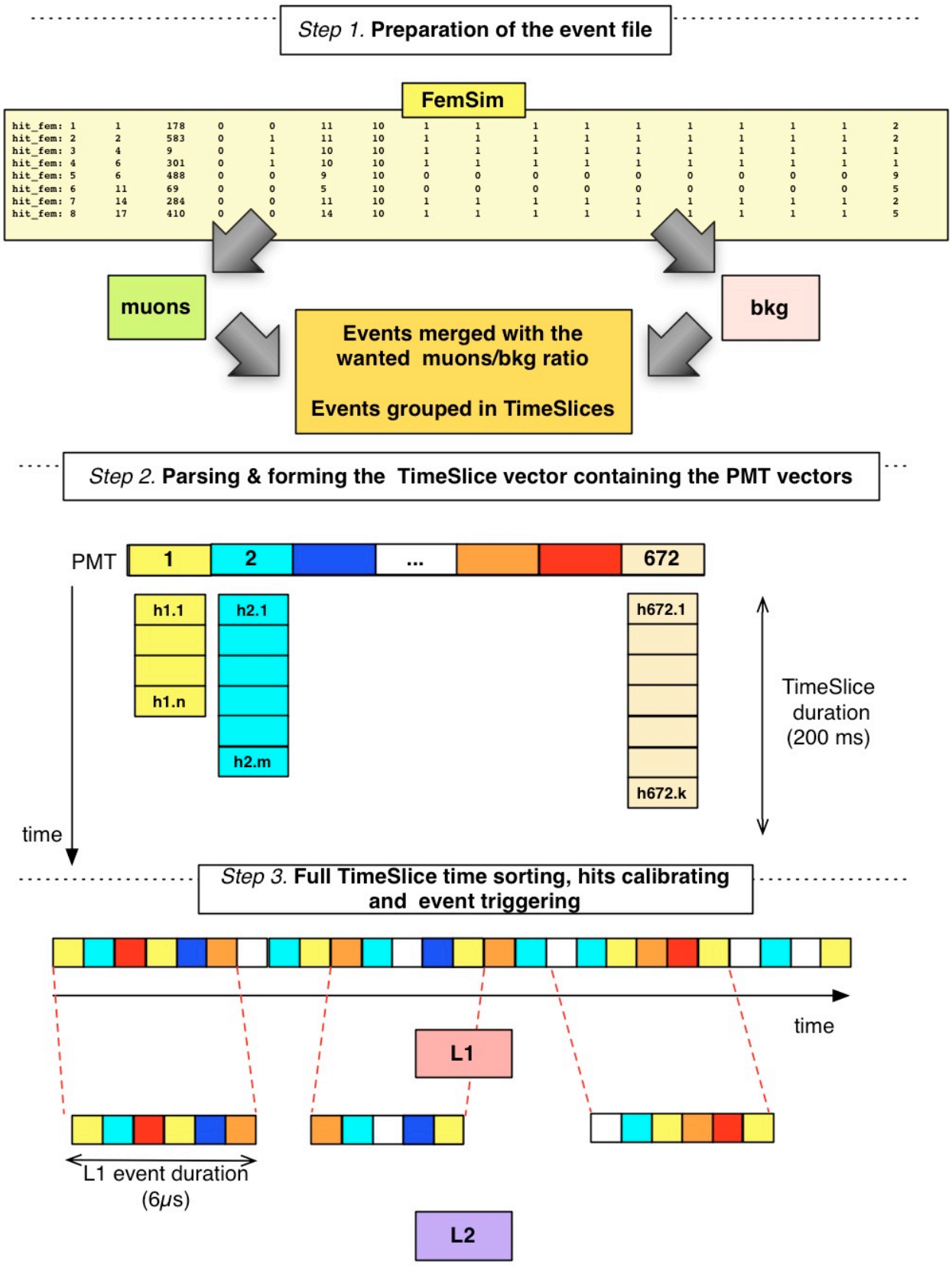



Figure 8. The data manipulation: from the FEMSIM output file to the trigger levels.

## 4 Trigger Levels

The Level 1 trigger consisted of a logic OR of the following conditions:

- Simple Coincidences: i.e. topological time-like trigger conditions involving hits occurring on near PMTs within a time-window of 20 ns. The circles in Figure 9 indicate the pairs of PMTs of each floor that can trigger Simple Coincidences L1 events.

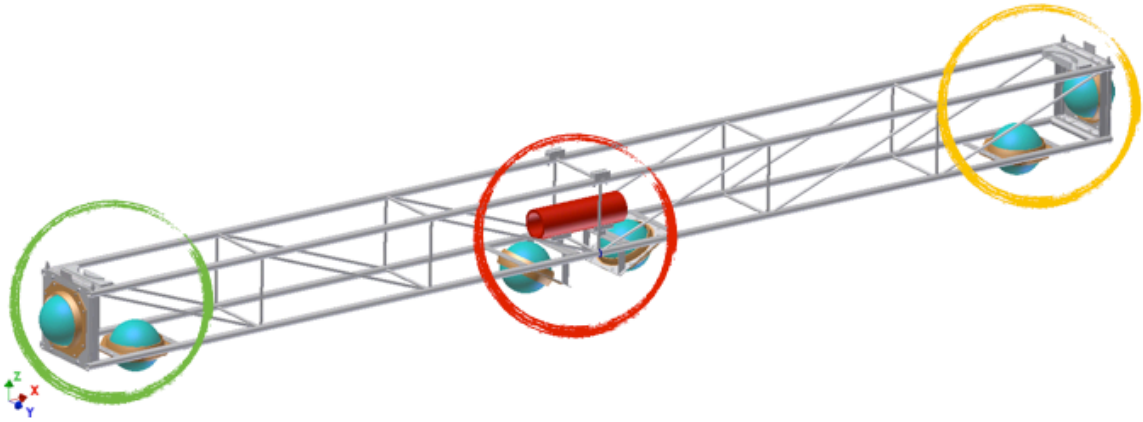


Figure 9. The circles groups PMTs in pairs: each pair could trigger a Simple Coincidence.

- Charge over threshold: when a hit charge is above a certain threshold (here set to 20 pC - note that 1 single photon electron hit charge is about 8 pC).

When a trigger seed was found in the stream of one TimeSlice, a L1 event was cut, saving all the hits from all the PMTs within $\pm$ 3 $\mu$s from the trigger seed occurrence. If a new seed had occurred before the end of the event, the event window was extended by counting additional 3 $\mu$s from the last found seed.

The Level 2 trigger consisted into a causality filter [6]. Only L1 events containing a number of L1 seeds $\geq 5$ were considered. The strategy consisted of testing the causality among the hits of the L1 seeds assuming being originated by a muon coming from a guess direction out of a collection of 210 ones which cover the full solid angle $4\pi$ (see Figure 10).

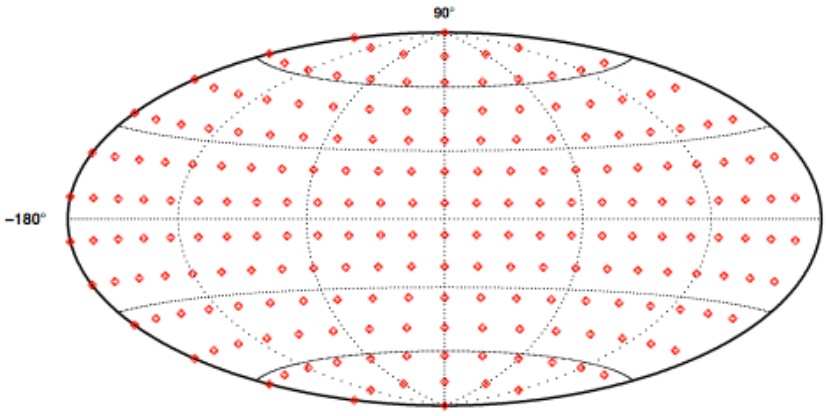



Figure 10. Hammer-Aitoff projection of the default grid of 210 directions used in the standard trigger algorithm

The L2 algorithm proceeded as the following: given the direction, the detector frame is rotated till the chosen direction becomes the vertical one. Then the causality relation was tested with equation (1) for all the pairs of L1 hits (refer also to Figure 11):

$$|(t_i - t_j)c - (z_i - z_j)| \leq \tan\theta_c \sqrt{[(x_i - x_j)^2 + (y_i - y_j)^2]} = \tan\theta_c |R_{ij}| \qquad (1)$$

where $i$ and $j$ represent two hits of the L1 seeds, $c$ is the speed of light in vacuum, $\theta_c$ is the Cherenkov angle in water (about $43^\circ$) and $(x,y,z,t)$ are the spatial-time coordinate of the PMT that recorded the hit.

When the number of hits satisfying equation (1) was equal or larger than 5, the trigger was set.

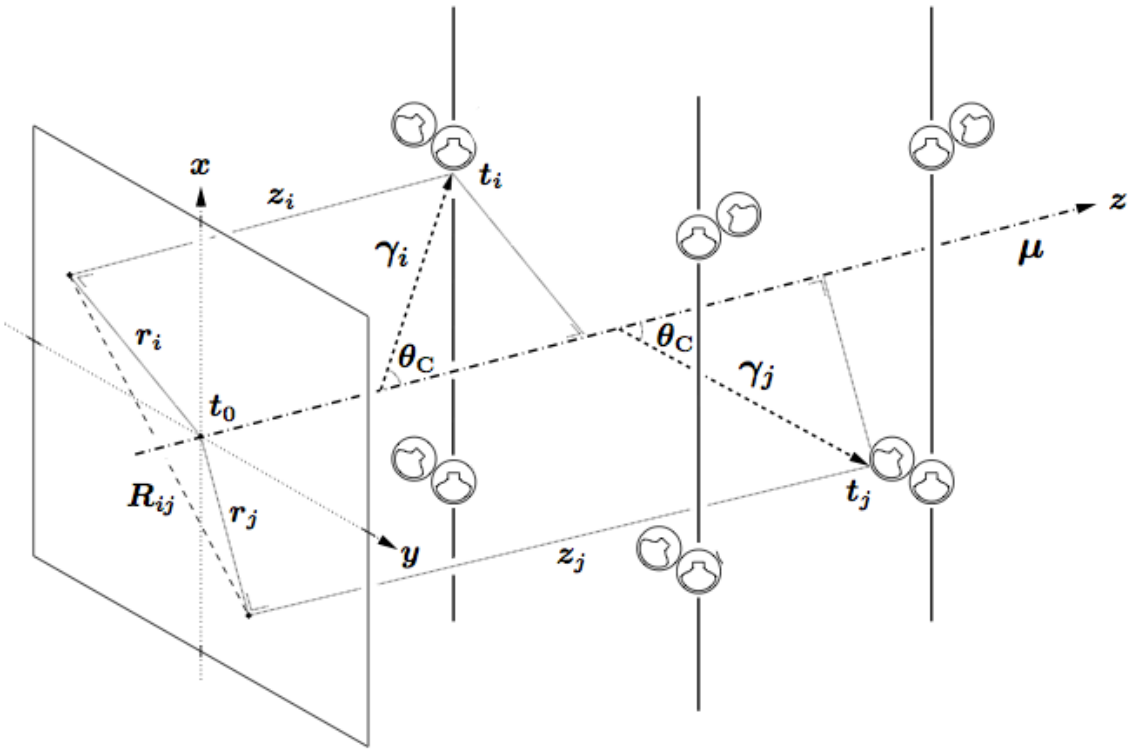



Figure 11. Schematic view of a muon transversing a part of the instrumented volume of the detector.

The application of the $L_1 + L_2$ trigger levels determined a trigger efficiency ranging 3-5%. It is a reasonable efficiency considering that the triggered good events are, for this simulation, downward going atmospheric muons, and being the NEMO-like towers optimized for the detection of upward going particles.

Moreover, approximatively the same trigger efficiency is found for the ANTARES detector [7], which has almost the same instrumented volume.

We conclude that we didn't implemented any kind of optimization of the trigger algorithms, since for the scope of this study, the dimensioning of the TCPU farm, we wanted to keep a conservative profile.

# 5 Benchmark set-up and results

We analyzed the TimeSlice files, produced according to the 5 approaches described in Section 2.2 , by running the TCPUSimu program on a HEPSPEC-06 benchmarked CNAF worker node, having the following characteristics:

```
Hostname: wn-204-03-25-02-b.cr.cnaf.infn.it
OS: Scientific Linux release 6.4 (Carbon)
Kernel: 2.6.32-358.18.1.el6.x86_64
CPU type: Intel(R) Xeon(R) CPU E5520  @ 2.27GHz
Cores: 8 x 2
CPU MHz: 2268.000
Cache size: 8192 KB
Memory: 24 GB
HEPSPEC version: hs06 sl6-32bit (hyperthreading on)
HEPSPEC score: 128.12
```

Note that the effective physical cores are 8, which are enhanced to 16 by the "hyperthreading", which is turned ON. We recall that the hyperthreading enhance of about 40% maximum the performances of the machine.

## 5.1 TIMING THE HIT MANIPULATION (8 PROCESSES / 8 CORES)

The main goal of this work is to determine the CPU power required to handle the collected hits and to apply the trigger algorithm. The result of this study is to output a CPU power quantity in HEPSPEC-06 units, so that it is spendable for planning the TCPU farm. For this reason, in order to better distribute the CPU load to the benchmarking worker node, we decided to run as many TCPUSimu program as the effective number of effective physical cores.

So for the principal purpose of this note, we made the main benchmark by running simultaneously 8 versions of the program TCPUSimu, compiled with

```
c++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3)
```

with option `-O2`, analyzing the same TimeSlice file. We took profit of the multiple running for gathering statistics in order to present more reliable results.

Tables 2 summarizes, for all the 5 approaches, the averaged measures of time needed to complete the hit manipulation sequence implemented in the TCPUSimu code. The various timings are referring to the following actions:

- parsing data from the TimeSlice file and allocating the necessary memory ($T_p + T_{ma}$) to host all the hits in a TimeSlice.
- calibrating the hits ($T_{cal}$)
- time-sorting all the the hits of each TimeSlice ($T_{sort}$)
- applying the $L_1$ and $L_2$ trigger level ($T_{L1}$ and $T_{L2}$, respectively)

| approach | $T_p + T_{ma}$ (ms) | $T_{ca}$ (ms) | $T_{sort}$ (ms) | $T_{L1}$ (ms) | $T_{L2}$ (ms) |
|---|---|---|---|---|---|
| realistic | 11280 ± 123 | 61 ± 1 | 1176 ± 12 | 596 ± 7 | 171 ± 2 |
| hardly realistic | 11251 ± 118 | 57 ± 1 | 1167 ± 12 | 588 ± 7 | 174 ± 2 |
| condensed | 11318 ± 111 | 57 ± 1 | 1146 ±11 | 579 ± 6 | 400 ± 4 |
| heavy background | 24792 ± 227 | 134 ± 2 | 2626 ± 24 | 1693 ± 18 | 5526 ± 49 |
| heavy background hardly realistic | 24904 ± 212 | 139 ± 2 | 2624 ± 21 | 1685 ± 17 | 5529 ± 46 |

Table 2.

Let's present some consideration on the results obtained so far.

First of all, let's recall from Figure 5 right that the number of hits per TimeSlice is more or less the same for both the "Realistic" and the "Hardly Realistic" approaches. This is true also for the "Condensed" approach. The explanation is because the contribution of direct Cherenkov hits in muon events is estimated to be of the order of 7 or 12 on top the additional background hits. Given the assumption of $\nu_{bkg}$ = 50 kHz, the pure background events have an averaged number of hits equal to 201 ($= \nu_{bkg} \times N_{PMTs} \times \Delta T_{TS}$), and this is confirmed by Figure 4, right. So the enhancement of hits in the "Hardly Realistic case"

Now, from Table 2 we note that the time needed for applying the L2 trigger ($T_{L2}$) increases with the number of muon events (i.e. passing from "Realistic" to "Hardly Realistic" and finally to "Condensed"): this is due to the fact that more seeds are found which need to be handled.

When considering the "Heavy Background" and "Heavy Background + Hardly Realistic" approaches, the numbers tell us that we are strongly dominated by the background, and the variations due to different muon rates $\nu_d$ = 27 Hz or 270 Hz are suppressed.

## 5.2 MEASURING THE COMPUTING POWER IN HEPSPEC-06 UNITS

In order to evaluate the effective computing resources needed by the TriDAS, we need to use the data reported in Table 2. However we need to consider also that real TCPU processes won't parse any file, since the TimeSlices will be received from the HitManagers through 10 GbE connections. What really matters in the ($T_p + T_{ma}$) measures is only the memory allocation time (i.e. $T_{ma}$). With independent tests, we measured that $T_{ma}$ = 1700 ms for the approaches with $\nu_{bkg}$ = 50 kHz, and about $T_{ma}$ = 3700 ms for the approaches with $\nu_{bkg}$ = 110 kHz.

In addition to this, to be more conservative, we considered an additional time contribution from the time-overhead necessary to receive a full TimeSlice from the HitManager through a 10 GbE point-to-point connections, which was estimated as $T_{HM}$ = 175 ms with $\nu_{bkg}$ = 50 kHz (size of one TimeSlice: 180 MB) and $T_{HM}$ = 386 ms with $\nu_{bkg}$ = 110 kHz (size of one TimeSlice: 395 MB).

We computed the estimated total amount of time required by a TCPU to process a TimeSlice with the following equation:

$$T_{tot} = T_{HM} + T_{ma} + T_{ca} + T_{sort} + T_{L1} + T_{L2} \qquad (2)$$

Finally, the required HEPSPEC-06 power is obtained with the following relation:

$$\text{HEPSPEC-06} = \text{HEPSPEC-06}_{\text{SCORE}} \times T_{tot} \times (\Delta T_{\text{TS}})^{-1} \quad (3)$$

where HEPSPEC-06$_{\text{SCORE}}$ is the score of the used server, 128.12 HEPSPEC-06.

The final results are reported in Table 3, where last column shows the computed number of estimated TCPU processes running.

| approach | $T_{tot}$ (ms) | HEPSPEC-06 | N. TCPU processes |
|---|---|---|---|
| realistic | 3880 ± 23 | 2483 ± 15 | 20 |
| hardly realistic | 3862 ± 22 | 2472 ± 14 | 20 |
| condensed | 4058 ± 21 | 2597 ± 13 | 21 |
| heavy background | 12064 ± 60 | 7721 ± 38 | 61 |
| heavy background + hardly realistic | 12062 ± 56 | 7720 ± 36 | 61 |

Table 3.

Note that the in Table 3, the number of requested processes is obtained by dividing $T_{tot}$ by the TimeSlice duration (200 ms). Obviously the shown number must always be linked to the worker-node CPU power used for this benchmark. That's why is preferable use the HEPSPEC value.

## 5.3 SCALABILITY-TEST: SINGLE RUNNING PROCESS

The same measurements of the previous subsection were repeated with only 1 process runnning on the worker-node. The results are reported in the following Tables 4 and 5.

| approach | $T_p + T_{ma}$ (ms) | $T_{ca}$ (ms) | $T_{sort}$ (ms) | $T_{L1}$ (ms) | $T_{L2}$ (ms) |
|---|---|---|---|---|---|
| realistic | 10574 ± 328 | 43 ± 2 | 1016 ± 29 | 506 ± 17 | 161 ± 5 |
| hardly realistic | 10844 ± 324 | 60 ± 3 | 1058 ± 28 | 611 ± 24 | 163 ± 5 |
| condensed | 10516 ± 287 | 43 ± 1 | 1015 ±25 | 501 ± 14 | 375 ± 10 |
| heavy background | 23235 ± 594 | 95 ± 2 | 2314 ± 54 | 1520 ± 43 | 5205 ± 130 |
| heavy background hardly realistic | 23706 ± 566 | 146 ± 5 | 2350 ± 50 | 1738 ± 50 | 5209 ± 121 |

Table 4.

| approach | $T_{tot}$ (ms) | HEPSPEC-06 | N. TCPU processes |
|---|---|---|---|
| realistic | 3603 ± 62 | 2306 ± 40 | 19 |
| hardly realistic | 3768 ± 63 | 2412 ± 40 | 20 |
| condensed | 3811 ± 55 | 2439 ± 35 | 20 |
| heavy background | 11009 ± 153 | 7046 ± 98 | 56 |
| heavy background + hardly realistic | 11318 ± 146 | 7243 ± 93 | 58 |

Table 5.

The results are compatible with the case when 8 processes were simultaneously running, apart from the errors which are larger now because of the lower statistics.

## 5.4 STRESS-TEST: 16 PROCESSES / 8 ×2 CORES - HYPERTHREADING

We repeated the same benchmark, this time with 16 processes running simultaneously. Tables 6 and 7 show the result of this test.

| approach | $T_p + T_{ma}$ (ms) | $T_{ca}$ (ms) | $T_{sort}$ (ms) | $T_{L1}$ (ms) | $T_{L2}$ (ms) |
|---|---|---|---|---|---|
| realistic | 20459 ± 172 | 97 ± 1 | 1641 ± 16 | 936 ± 9 | 275 ± 3 |
| hardly realistic | 20421 ± 164 | 94 ± 2 | 1601 ± 15 | 915 ± 9 | 278 ± 2 |
| condensed | 20213 ± 149 | 88 ± 1 | 1548 ±14 | 884 ± 7 | 649 ± 5 |
| heavy background | 43002 ± 351 | 186 ± 2 | 3514 ± 44 | 2511 ± 23 | 8549 ± 80 |
| heavy background hardly realistic | 44292 ± 304 | 188 ± 2 | 3485 ± 27 | 2541 ± 23 | 8748 ± 81 |

Table 6.

| approach | $T_{tot}$ (ms) | HEPSPEC-06 | N. TCPU processes |
|---|---|---|---|
| realistic | 4825 ± 24 | 3088 ± 15 | 25 |
| hardly realistic | 4764 ± 22 | 3049 ± 14 | 25 |
| condensed | 5044 ± 21 | 3228 ± 13 | 26 |
| heavy background | 16845 ± 95 | 10781 ± 61 | 85 |
| heavy background + hardly realistic | 17048 ± 90 | 10910 ± 58 | 86 |

Table 7.

In this case we are running 2 processes per physical core, but instead of measuring double-sized timings, the hyperthreadding comes in our help. In fact, hyperthreading increases the performances of about the expected 40%. However this help is not sufficient to avoid a worsening of the

averaged performances: by comparing the $T_{tot}$ column of Tables 3 and 5 we see that when running 16 processes on the same worker-node, each benchmark gives timings about 24% larger (which obviously yield to the larger CPU power request). This lead to the following first conclusion: for optimizing the resources to purchase, we need to limit the number of independently running processes on one server to the number of physical cores.

## 5.5 TRICKING TIMING ANOMALIES

Given a certain $\nu_{bkg}$, one should generally expect $T_{tot}$ to increase with the muon rate, since the background remains constant with the additional hits from direct Cherenkov photons. We have seen that when passing from a muon rate of $\nu_d$ = 27 Hz to $\nu_d$ = 270 Hz, the number of hits per TimeSlice is not sensibly enhanced. This almost true for the two cases of $\nu_{bkg}$ = 50 kHz and $\nu_{bkg}$ = 110 kHz.

Nonetheless, looking at Tables 2, 4 and 6 we see that when $\nu_{bkg}$ = 50 kHz, so for the "Realistic", "Hardly Realistic" and "Condensed", there is an unexpected trend of the timings, which indeed decrease with the muon rate. Consider for example the $T_{sort}$ column of Table 2, related to the run of 8 simultaneous processes. The "Realistic" and "Hardly Realistic" timings are compatible within the errors; on the other side the "Condensed" timing could be unlikely compatible with the "Realistic" one, having a more significant deviation of 1.8 sigmas, enough to state that the probability of a statistical fluctuation is less then 6.4%. We find the same in Table 6, related to the run of 16 simultaneous processes, supported by the hyperthreading. Indeed here, the "Condensed" $T_{sort}$ is far less than the one for the "Realistic" with a significance of 4 sigmas.

Still in the framework of the 16 running processes, we repeated the measurement again for both the "Realistic" and the "Hardly realistic" approaches, with the following similar results:

| approach | $T_p + T_{ma}$ (ms) | $T_{ca}$ (ms) | $T_{sort}$ (ms) | $T_{L1}$ (ms) | $T_{L2}$ (ms) | $T_{tot}$ | HEPSPEC -06 | N. TCPU processes |
|---|---|---|---|---|---|---|---|---|
| realistic 2 | 20382 ± 169 | 89 ± 1 | 1612 ± 20 | 901 ± 8 | 276 ± 3 | 4753 ± 26 | 3042 | 25 |
| h. realistic 2 | 20297 ± 165 | 88 ± 1 | 1551 ± 14 | 898 ± 9 | 278 ± 3 | 4690 ± 22 | 3001 | 24 |

Table 8.

Finally note that the re-measured global $T_{tot}$ are a little smaller with respect to the first set of measures. This difference is due to the repetition of the benchmarking tests, so biased by unpredictable fluctuation of the worker-node performances.

The possible explanation is that when you enhance the $\nu_d$ (stepping from 27 Hz to 270 Hz to the extreme 25 kHz of the "Condensed" approach), the number of Cherenkov hits per muon event is enhanced correspondently. For example have a look at Figure 4 - right. Being the Cherenkov hits causally correlated, their sorting is easier.

## 5.6 SUPPLEMENTARY TEST: NOT OPTIMIZED SINGLE PROCESS RUNNING

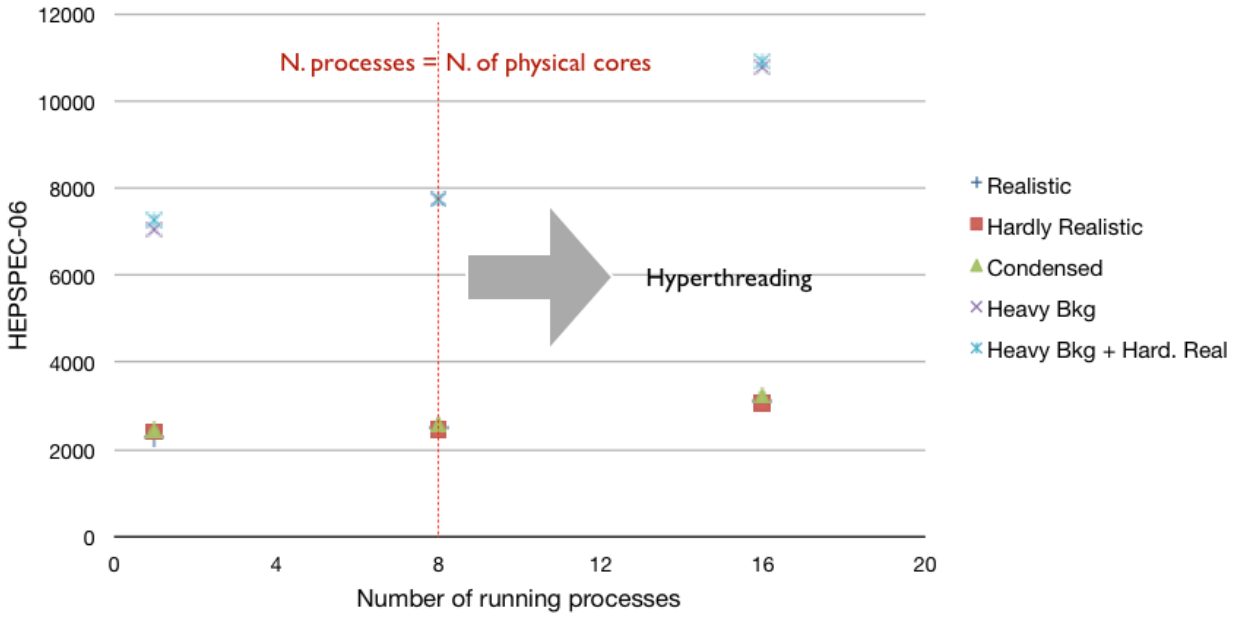
In order to get the feeling about the impact of the compile-level optimization on the process running, we repeated the same benchmark such as reported in subsection 5.3, with a not optimized code. As it is shown in Table 9, the various timings are dramatically larger.

| approach | $T_p + T_{ma}$ (ms) | $T_{ca}$ (ms) | $T_{sort}$ (ms) | $T_{L1}$ (ms) | $T_{L2}$ (ms) |
|---|---|---|---|---|---|
| realistic | 12397 ± 339 | 242 ± 7 | 3983 ± 100 | 1311 ± 37 | 986 ± 27 |
| hardly realistic | 12151 ± 379 | 243 ± 7 | 3985 ± 53 | 1252 ± 41 | 414 ± 13 |
| condensed | 12324 ± 363 | 242 ± 7 | 3997 ±108 | 1309 ± 38 | 423 ± 12 |
| heavy background | 27015 ± 693 | 533 ± 14 | 9036 ± 215 | 3895 ± 105 | 14133 ± 351 |
| heavy background hardly realistic | 26785 ± 636 | 534 ± 13 | 9102 ± 202 | 3832 ± 100 | 14172 ± 329 |

Table 9.

## 6 Conclusions

Figure 12 shows the HEPSPEC-06 scoring for the different approaches and for different number of running processes. It was compiled with data taken from HEPSPEC-06 column of Tables 3,5 and 7. It is apparent the scalability of the performances in the region when the number of running processes is less or equal than the number of physical cores. When the number of running processes is larger, the performances are worsened, even though the hyperthreading is providing some speed-up. We then recall the conclusion of section 5.4, where we stated that it is worthy to limit the maximum running processes to the number of the physical cores (8 in this case).




We refer to Table 3 , where results are a bit more conservative and statistically reliable, to address the conclusions of this work.

Let's answer to the question: "What is the required number of TCPU servers and the number of running TCPU processes per server, for the 8 Tower Detector DAQ?

In order to answer this question, we preliminary note that, from the experience of the TriDAS for NEMO Phase 2, the TCPU process must handle a TimeSlice FIFO, used for buffering the

incoming data from the HitManagers which are sending frames of the TimeSlices in an asynchronous way.

Such a FIFO could be dimensioned up to 20 elements, allowing a circular buffering of even 4 seconds of data. Beside technical motivations, one would like to buffer a macroscopic (~O(1 s))also for Physics reasons, like buffering data for GRB triggers and possibly other kind of follow-up triggers.

All this has strong implication for the RAM of the TCPU server, since in the Heavy Background case about 8 GB per process could be needed. This puts a further constraint forcing us to limit more the number of running processes even with respect to the number of physical core.

We should then assume to run not more than 2 TCPU process per server. We note that this conservative assumption accomplishes with the case of multi-threading processes.

Assuming the conservative approach, i.e. considering the "Heavy Background" result of Table 3, this imply to purchase not less than 30 servers. The safe-proof suggested number is 40 servers.

This choice is fairly conservative and it allows to easily reconfiguring the TCPU computing farm by running, on demand, one or two TCPU processes on each server, and so being able to accomplish even with the most dramatic scenario.

## Bibliography


1. R. Coniglione, D. Lattuada, private communications
2. http://wiki.infn.it/cn/csn2/km3/monte_carlo
3. G. Carminati, M. Bazzotti, A. Margiotta, M. Spurio, Atmospheric MUons from PArametric formulas: a fast GEnerator for neutrino telescopes (MUPAGE), Computer Physics Communications, Volume 179, Issue 12, 15 December 2008, Pages 915–923
4. G. Carminati, M. Bazzotti, S. Biagi, S. Cecchini, T. Chiarusi, A. Margiotta, M. Sioli, M. Spurio. MUPAGE: a fast atmospheric MUon GEnerator for neutrino telescopes based on PArametric formulas , Jul 2009. 4 pp. , e-Print: arXiv:0907.5563 [astro-ph.IM]
5. T. Chiarusi, The NEMO Trigger and Data Acquisition System, Volume 725, 11 October 2013, Pages 129–132
6. B. Bakker - Trigger studies for the Antares and KM3NeT neutrino telescopes - Bachelor Thesis, 29/07/2011
7. M. Spurio, A. Margiotta, private communications