Computational time reconstruction code

Summary

- Study computational time on reconstruction code:
  - Computational time test on CLOUD and LNGS
  - Added to the reco file the time consumption of some pieces of the code:
    - Clustering time per image;
    - Variables calculation time per image;
    - Total time of the run.
  - Close look at the Variables calculation:
    - Grouping the variables in 13 categories;
    - Adding category by category to measure the time of execution;
    - Deeper study on the most time-consuming category.
  - Evaluate reconstruction computational time for Run2 e Run3.
- Conclusions

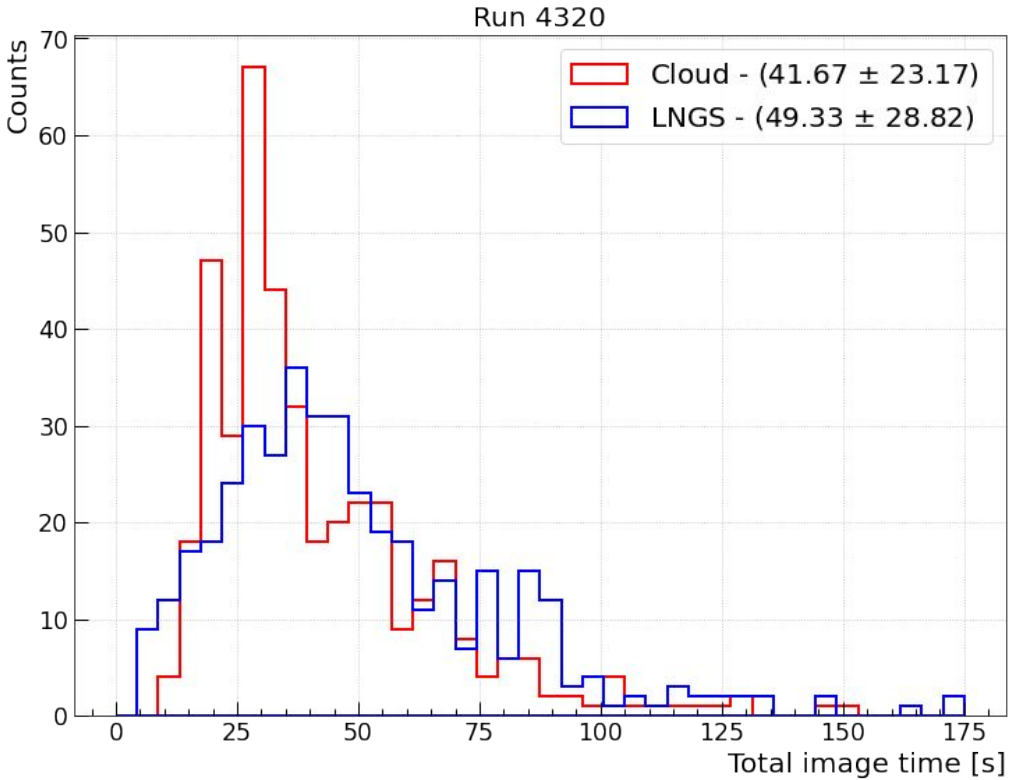
Computational time test on CLOUD and LNGS

We have tested the reconstruction algorithm in terms of computational time using the HTCondor queues on the so-called CLOUD and also on the LNGS batch system (cygno-custom), both of them using SingleCore mode:

- Run1 - 4320 (Fe55) and 400 images;
- Using Autumn22 tag;

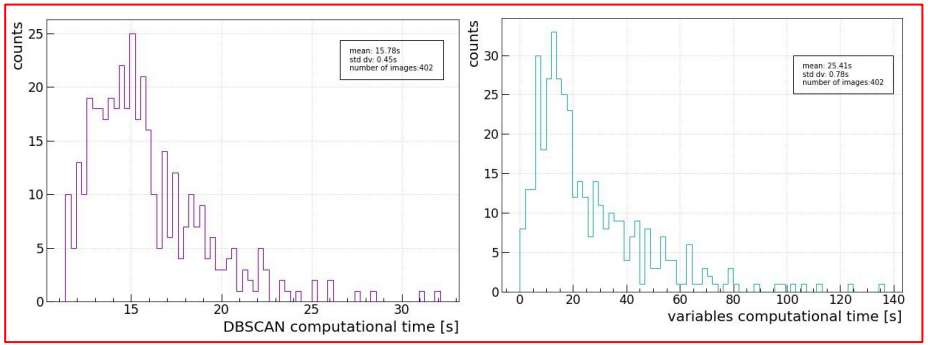
And as you can see the computational time is comparable.

All the analysis in the next slides were done using both environments.



Added to the reco file the time consumption of some pieces of the code:

- ❑ Clustering time per image;
- ❑ Variables calculation time per image;
- ❑ Fill Camera variables time per image;
- ❑ Fill Cluster variables time per image;
- ❑ Total time of the run.



Example:

Using the following configuration:

- Run1 LIME underground number: 4320 (Fe 25cm far from GEMs)
- Reconstruction branch: Autumn22
- Processed at: HTCondor Cloud
- Number of Threads: 1 (Single core)

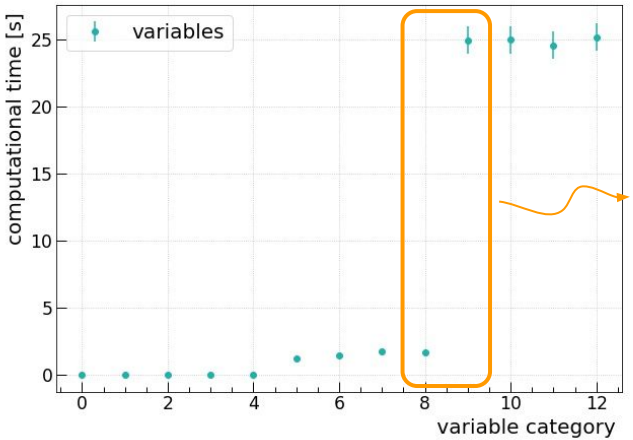
The resulted computational time is:

- Average computational time per image:
  - **DBSCAN: c.a. 16 s**
  - **Variables: c.a 25 s**
  - **Total time: c.a 42 s**
- Average computational time entire run (~400 images):
  - **Total time: 292.86 ± 5.6 minutes**

Close look at the variables calculation

The reconstruction algorithm was tested (using the same run) for all the different kind of categories listed in Table. The categories were being added one by one to study the trend of the computational time.

The result can be seen below:



Between 8 and 9 we have an significant increase on the computational time

variable	description	category
run		0
event	event number	
pedestal_run	run number used for pedestal subtraction	
t_DBSCAN		
t_variables		
cmos_integral	integral counts of the full CMOS sensor	1
cmos_mean	average counts of the full CMOS sensor	
cmos_rms	RMS of the counts of the full CMOS sensor	
nSc	Number of clusters found in the image	2
sc_integral	uncalibrated integral of counts of all the pixels in the cluster	
sc_size	number of pixels of the cluster, without zero-suppression	3
sc_nhits	number of pixels of the cluster above zero-suppression threshold	
sc_rms		4
sc_length	length of the major axis of the cluster	5
sc_width	length of the minor axis of the cluster	6
sc_xmax	x position of the rightmost pixel of the cluster	7
sc_xmin	x position of the leftmost pixel of the cluster	
sc_ymax	y position of the topmost pixel of the cluster	
sc_ymin	y position of the bottommost pixel of the cluster	
sc_xmean	x position of the cluster energy baricenter	8
sc_ymean	position of the cluster energy baricenter	
sc_theta	polar angle inclination of the major-axis of the cluster	9
sc_lfullrms	full RMS of the cluster along the major axis	10
sc_tfullrms	full RMS of the cluster along the minor axis	
sc_longrms	truncated RMS of the cluster along the major axis	11
sc_latrms	truncated RMS of the cluster along the minor axis	
sc_tgaussamp	amplitude of the Gaussian transverse profile	12
sc_tgaussmean	mean position of the Gaussian transverse profile	
sc_tgausssigma	standard deviation of the Gaussian transverse profile	
sc_tchi2	chi-squared of the Gaussian fit to the transverse profile	
sc_tstatus	status of the Gaussian fit to the transverse profile	
sc_lgaussamp	amplitude of the Gaussian longitudinal profile	
sc_lgaussmean	mean position of the Gaussian longitudinal profile	
sc_lgausssigma	standard deviation of the Gaussian longitudinal profile	
sc_lchi2	chi-squared of the Gaussian fit to the longitudinal profile	
sc_lstatus	status of the Gaussian fit to the longitudinal profile	
sc_lp0amplitude	mplitude of the main peak of the longitudinal cluster profile	13
sc_lp0prominence	prominence main peak wrt the baseline along the long. cluster profile	
sc_lp0fwhm	FWHM main peak of the long. cluster profile	
sc_lp0mean	mean position main peak wrt the start of the cluster of long. cluster profile	
sc_tp0fwhm	FWHM main peak of the transverse cluster profile	

Table 5.1: Variables calculated in the full reconstruction code grouped in different categories.

Deeper study on the most time-consuming category

Having a close look at the algorithm we saw that the function “DynamicsProfileBins” was taking most of the time.

In an attempt to improve the computational time we have optimized the function using numpy arrays. The output of the “*DynamicsProfileBins*” function was not changed!

Case with a big track (Run 4320 Image 9):

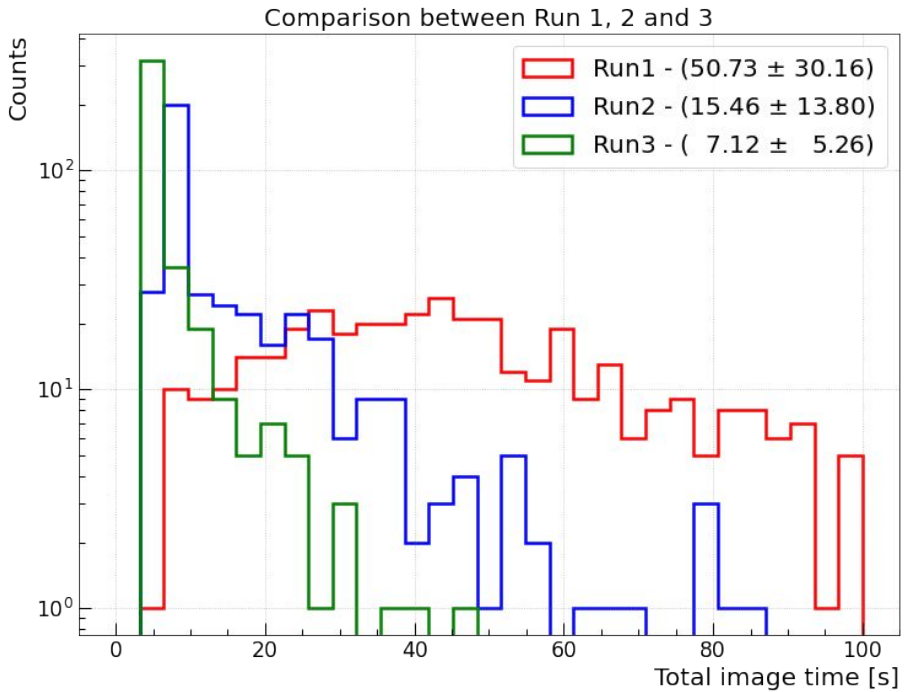
Function	Computational Time [s]
DynamicsProfileBins “Original”	86,1864
DynamicsProfileBins “Arrays”	0,4801
Factor	179,53

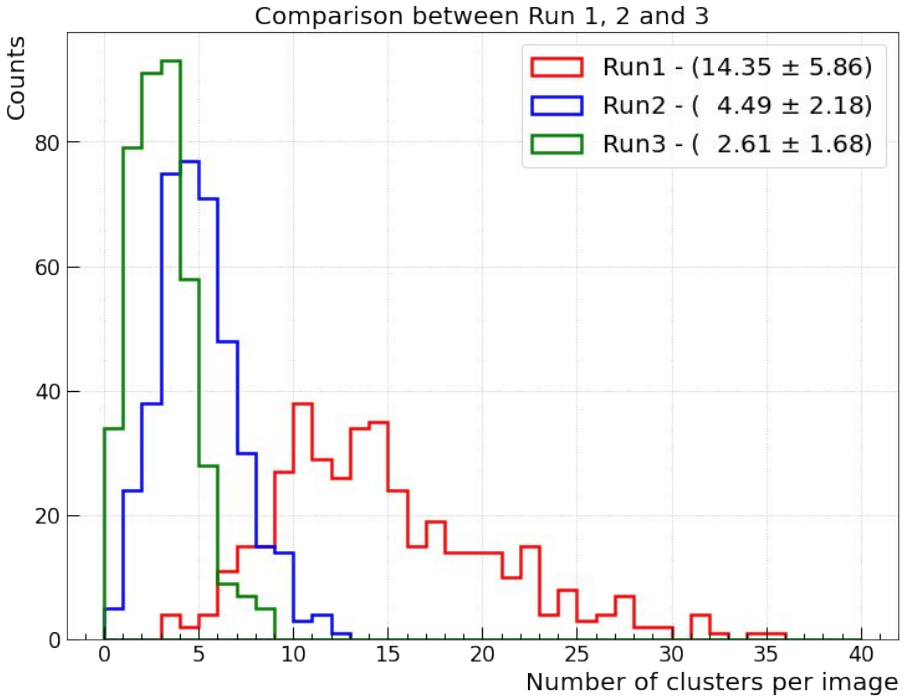
With this new function we could almost *zeroed* the variable calculation contribution.

Evaluate reconstruction computational time for $Run1_{(4320)}$, $Run2_{(9877)}$ e $Run3_{(17408)}$

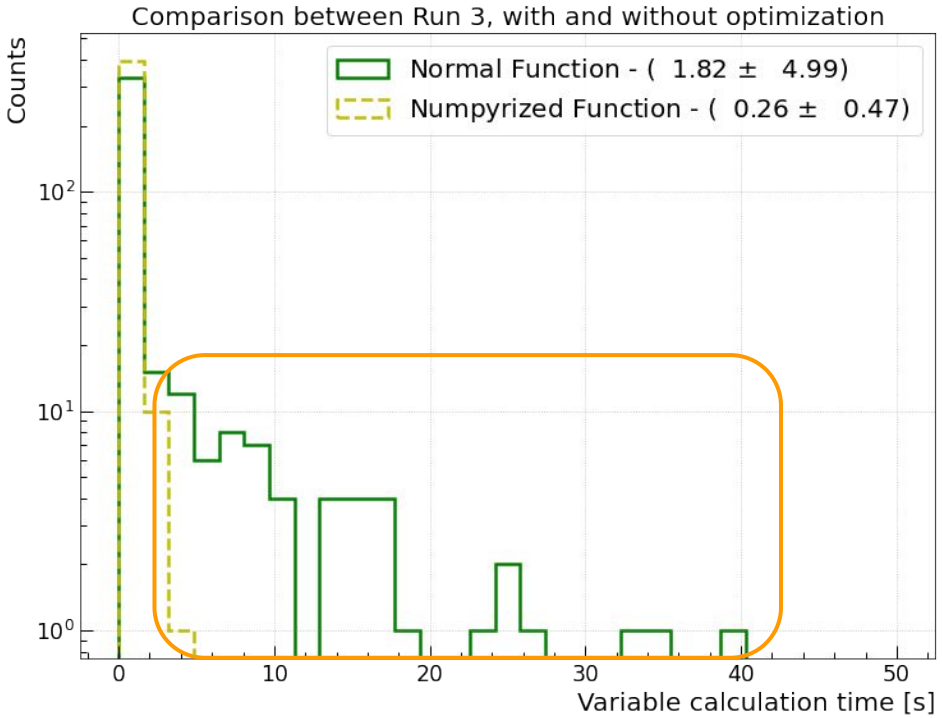
(without the optimization)

Run1 using Autumn22 e
Run2 e 3 using Winter23





Evaluating $Run3_{(17408)}$ with and without the optimization



As we can see in this new implementation is capable of decrease the computational time even for the Run3.

Conclusions

We have two main contributions for the computational time: DDBSCAN + Variables Calculation:

- The first **increase** with the **amount of clusters** found per image;
- And the second with the **size** of the clusters, but with the ‘numpyrized’ function we could almost *zeroed* this contribution.

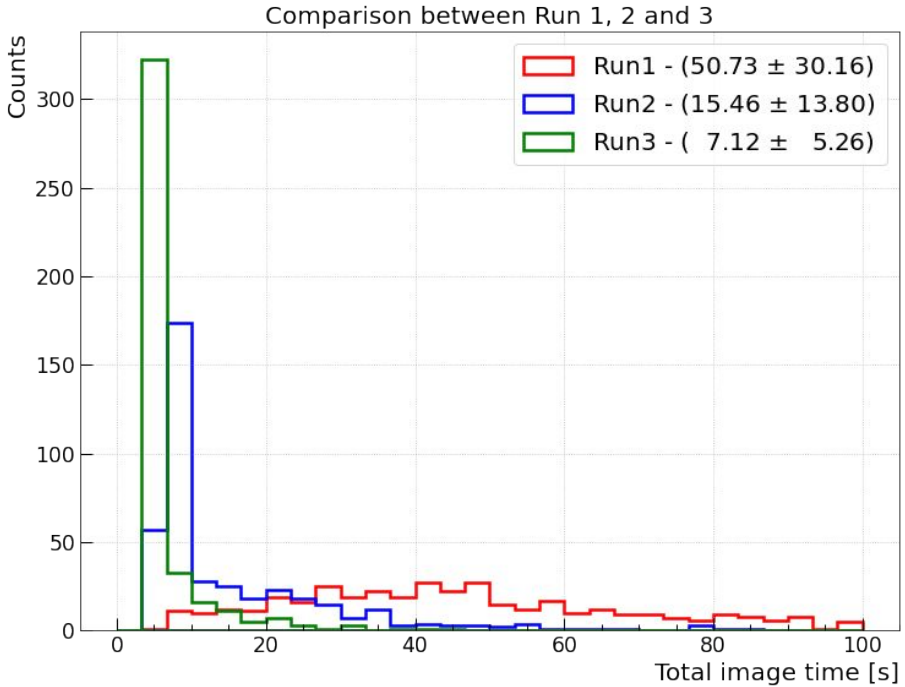
Apart from that we saw that even for Run3 (the last populated one by now) the reconstruction algorithm is in average over 7 seconds per image (but some images could take up to 40 seconds).

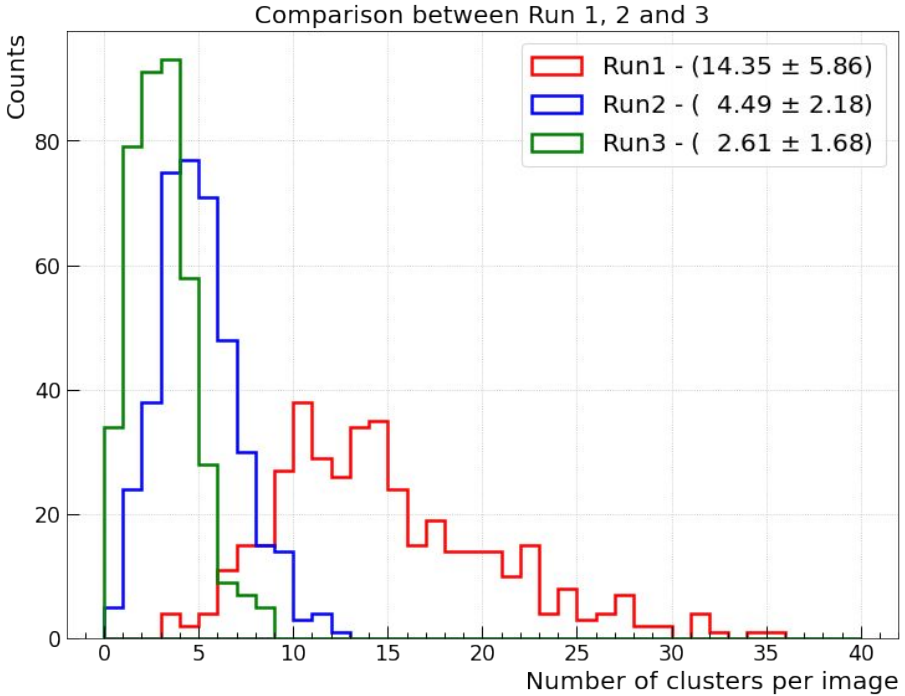
BACKUP

Evaluate reconstruction computational time for $Run1_{(4320)}$, $Run2_{(9877)}$ e $Run3_{(17408)}$

(without the optimization)

Run1 using Autumn22 e
Run2 e 3 using Winter23





Dynamics Profile

```
def dynamicProfileBins(hits,coord='x',relError=0.1):
        minPixels = max(1,1/relError/relError)
        print('minPixels: %f' % minPixels)
        index = 0 if coord=='x' else 1
        xmin=min([h[index] for h in hits])
        print('xmin: %f' % xmin)
        xmax=max([h[index] for h in hits])
        print('xmax: %f' % xmax)
        x=int(xmin)
        print('x: %f' % x)
        xedges=[x]
        integral=0
        while x<int(xmax):
            if integral<minPixels:
                somma = sum([int(h[index])==int(x) for h in hits])
                integral += somma
                print('x:        %f' % int(x))
                print('counts:   %f' % somma)
                print('integral: %f' % integral)
            else:
                print('Saved x: %f' % x)
                xedges.append(x)
                integral=0
            x+=1
        xedges.append(int(xmax))
        return xedges
```

3x for routine

```
def dynamicProfileBins_v2(hits,coord='x',relError=0.1):
    import numpy as np

    minPixels = max(1,1/relError/relError)
    index = 0 if coord=='x' else 1
    h = hits.T[index].astype(int)
    xmin=min(h)
    xmax=max(h)
    x=xmin
    xedges=[x]
    integral=0

    xunique, xcounts = np.unique(h,return_counts=True)
    if xmin>=0:
        xrange = list(range(0, xmax+1))
        c      = np.zeros(len(xrange),dtype = int)
        c[xunique] = xcounts
        c = c[xmin:-1]
    else:
        xrange = list(range(0, (xmax+1)-xmin))
        c      = np.zeros(len(xrange),dtype = int)
        c[xunique-xmin] = xcounts

    for ind, x in enumerate(range(xmin,xmax)):
        if integral<minPixels:
            integral += c[ind]
        else:
            #print('Saved x:  %f' % x)
            xedges.append(x)
            integral=0
    xedges.append(xmax)
    return xedges
```

Tempo Dynamics V1 x Dynamics V2 in one image

```
Processing Run:  4320 - Event  9 ...
DBSCAN creazione classe in 0.0012 seconds
  1.1 DBSCAN in 67.9507 seconds
Elapse time Dynamics V1: 28.884532690048218 seconds
Elapse time Dynamics V2: 0.1162409782409668 seconds
Elapse time Dynamics V1: 0.2740314006805420 seconds
Elapse time Dynamics V2: 0.00939488410949707 seconds
Elapse time Dynamics V1: 0.1410782337887207 seconds
Elapse time Dynamics V2: 0.005676984786987305 seconds
Elapse time Dynamics V1: 14.410872459411621 seconds
Elapse time Dynamics V2: 0.08100485801696777 seconds
Elapse time Dynamics V1: 18.32040810585022 seconds
Elapse time Dynamics V2: 0.0732569694519043 seconds
Elapse time Dynamics V1: 20.196839570999146 seconds
Elapse time Dynamics V2: 0.09746932983398438 seconds
Elapse time Dynamics V1: 1.6447553634643555 seconds
Elapse time Dynamics V2: 0.02428913116455078 seconds
Elapse time Dynamics V1: 0.0052869319915771484 seconds
Elapse time Dynamics V2: 0.0015254020690917969 seconds
Elapse time Dynamics V1: 0.0002174377441406250 seconds
Elapse time Dynamics V2: 0.0003840923309326172 seconds
Elapse time Dynamics V1: 0.03289365768432617 seconds
Elapse time Dynamics V2: 0.0046956539154052734 seconds
Elapse time Dynamics V1: 0.011032819747924805 seconds
Elapse time Dynamics V2: 0.0023643970489501953 seconds
Elapse time Dynamics V1: 0.0005784034729003906 seconds
Elapse time Dynamics V2: 0.0004942417144775391 seconds
Elapse time Dynamics V1: 2.0818562507629395 seconds
Elapse time Dynamics V2: 0.0397791862487793 seconds
Elapse time Dynamics V1: 0.0012869834899902344 seconds
Elapse time Dynamics V2: 0.0008943080902099609 seconds
Elapse time Dynamics V1: 0.12367987632751465 seconds
Elapse time Dynamics V2: 0.011322259902954102 seconds
Elapse time Dynamics V1: 0.023140907287597656 seconds
Elapse time Dynamics V2: 0.0037899017333984375 seconds
Elapse time Dynamics V1: 0.01966238021850586 seconds
Elapse time Dynamics V2: 0.0034668445587158203 seconds
Elapse time Dynamics V1: 0.0031366348266601562 seconds
Elapse time Dynamics V2: 0.0010635852813720703 seconds
Elapse time Dynamics V1: 0.0018310546875 seconds
Elapse time Dynamics V2: 0.0008037090301513672 seconds
Elapse time Dynamics V1: 0.009323835372924805 seconds
Elapse time Dynamics V2: 0.00215792655944824 seconds
```

Dynamics V1 Total	86,1864	seconds
Dynamics V2 Total	0,4801	seconds
Factor	179,53	times

Number of clusters vs computational time