Parallelizing the denoising stage of an acoustic manatee count algorithm using OpenACC directives

Advance Computing Laboratory

M.Sc. Jorge Castro C.

jcastro@cenat.ac.cr

February, 2018

Collaborators

Problem description

- Count methods:

Problem description

- Count methods:
  - Aerial census [1]

Problem description

- Count methods:
  - Aerial census [1]
  - Direct sightings [2]

Problem description

- Count methods:
  - Aerial census [1]
  - Direct sightings [2]
- Detection methods:

Problem description

- Count methods:
  - Aerial census [1]
  - Direct sightings [2]
- Detection methods:
  - Infrared cameras [3]

Problem description

- Count methods:
  - Aerial census [1]
  - Direct sightings [2]
- Detection methods:
  - Infrared cameras [3]
  - Side scan sonar [4]

Problem description

- Count methods:
  - Aerial census [1]
  - Direct sightings [2]
- Detection methods:
  - Infrared cameras [3]
  - Side scan sonar [4]
  - Vocalizations [5]

Problem description

- Count methods:
  - Aerial census [1]
  - Direct sightings [2]
- Detection methods:
  - Infrared cameras [3]
  - Side scan sonar [4]
  - Vocalizations [5]

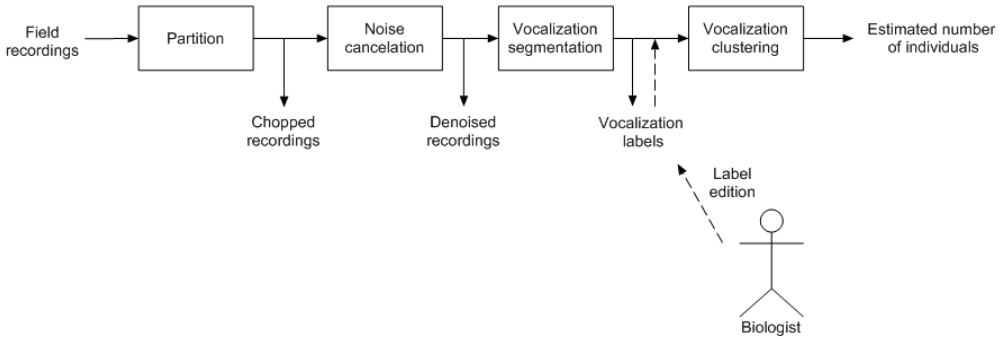
¿How to automatically count manatees using their vocalizations?

Problem description

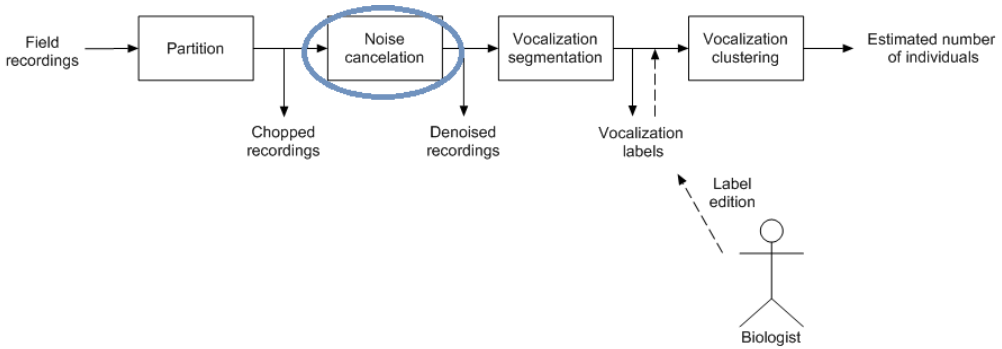
- Count methods:
  - Aerial census [1]
  - Direct sightings [2]
- Detection methods:
  - Infrared cameras [3]
  - Side scan sonar [4]
  - Vocalizations [5]

¿How to automatically count manatees using their vocalizations?

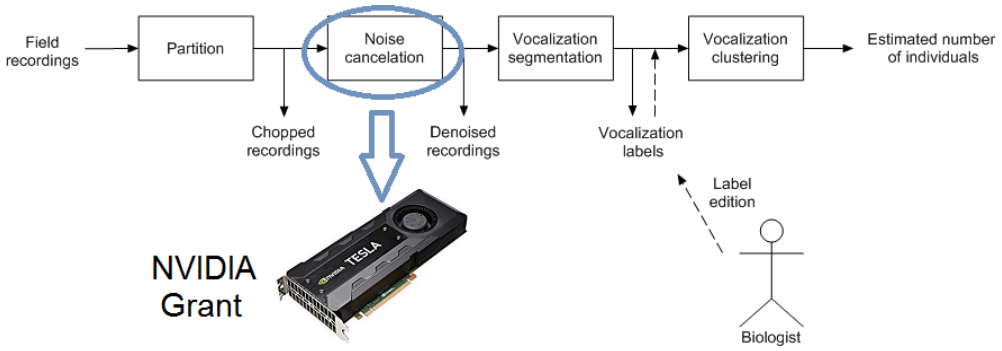
Automatic Manatee Count Method



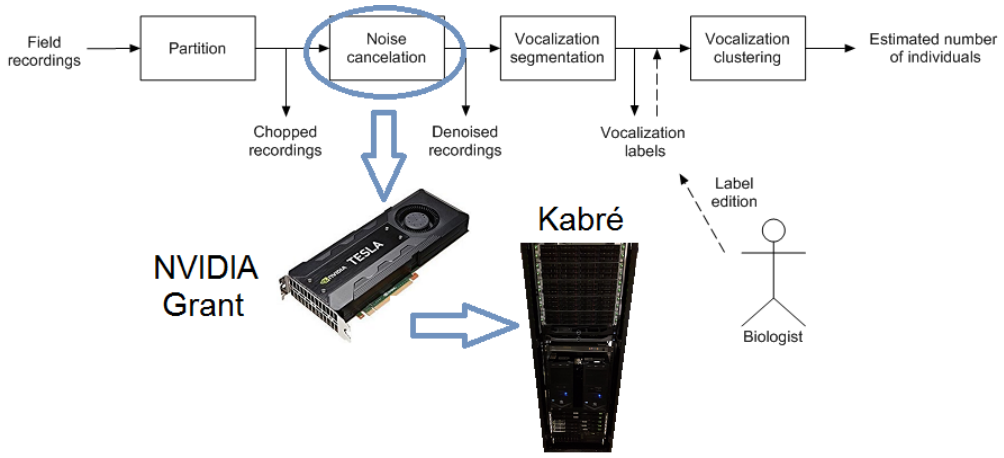
Automatic Manatee Count Method



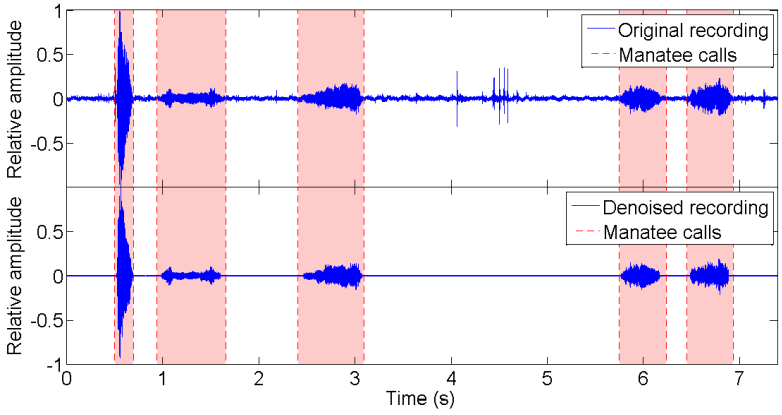
Automatic Manatee Count Method



Automatic Manatee Count Method

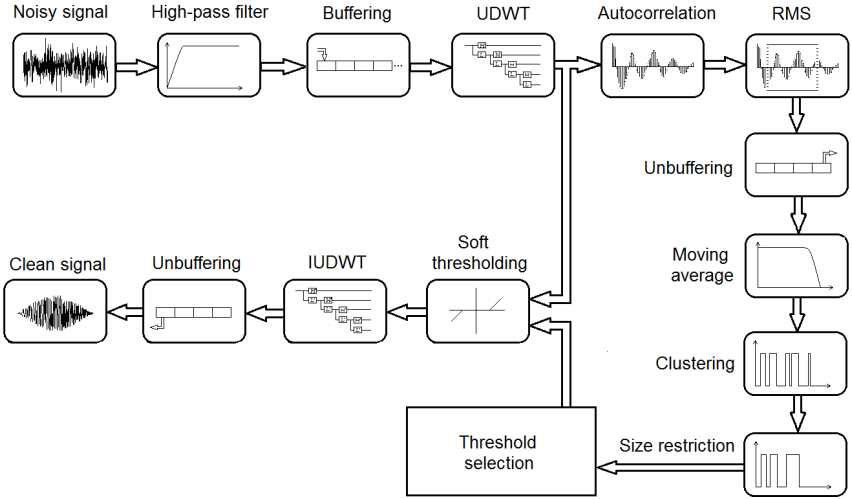


Denoising method

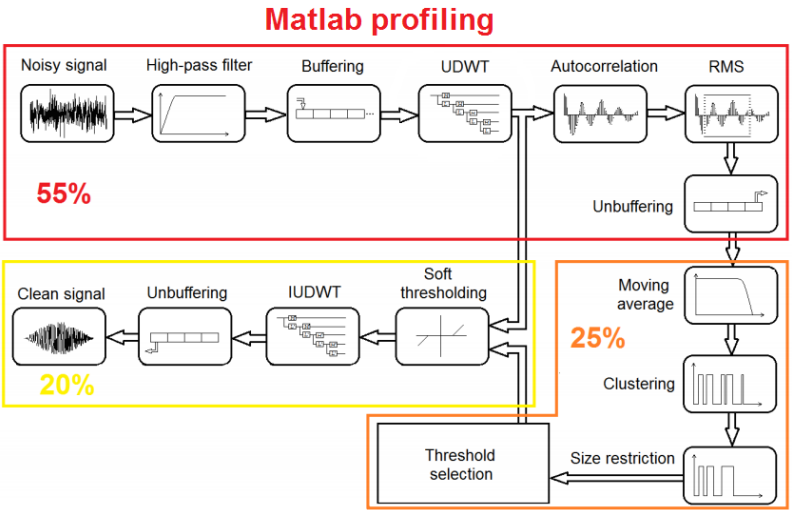


- Original
- Denoised

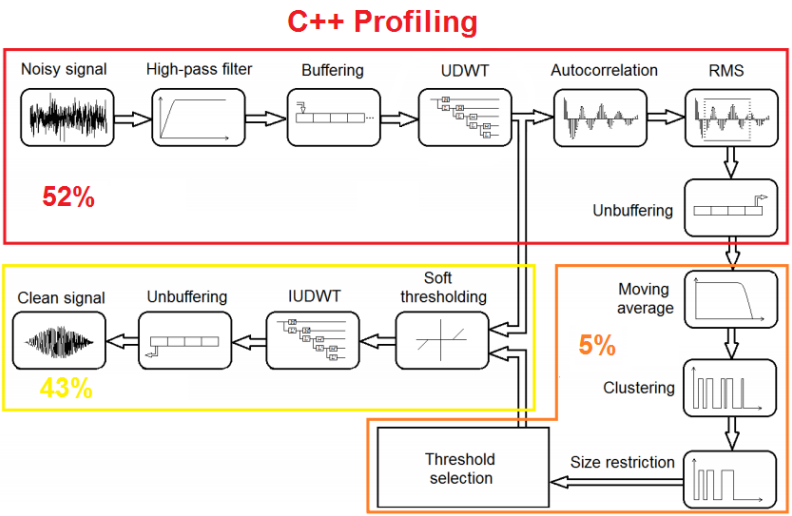
Denoising method



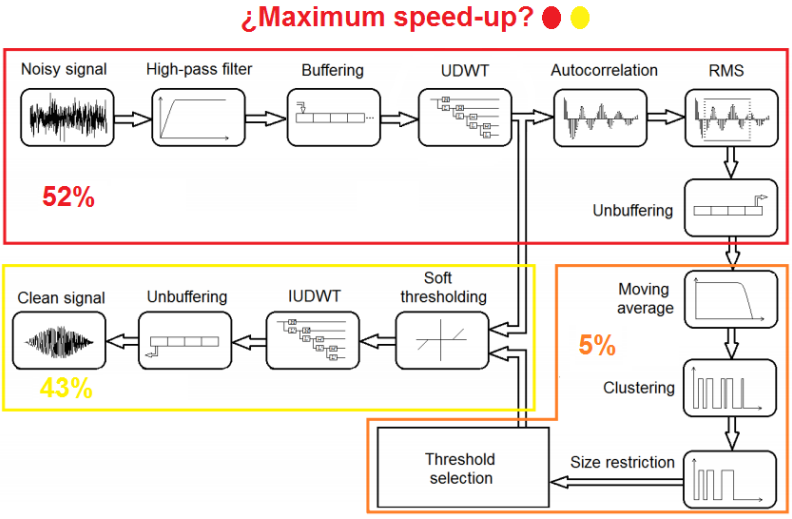
Denoising method



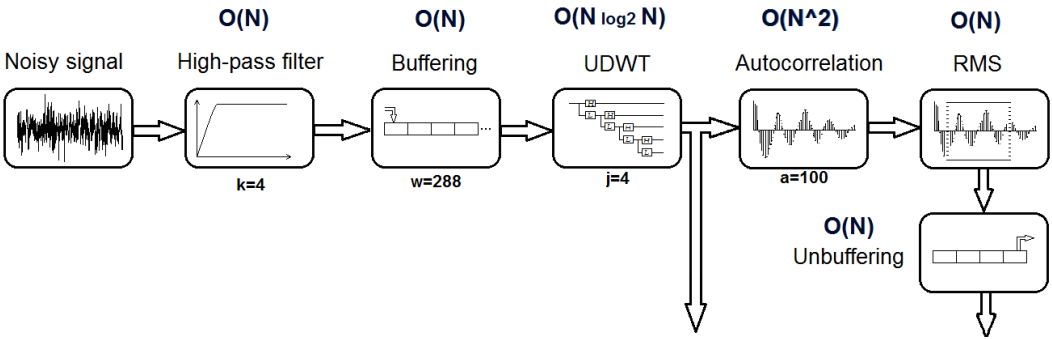
Denoising method



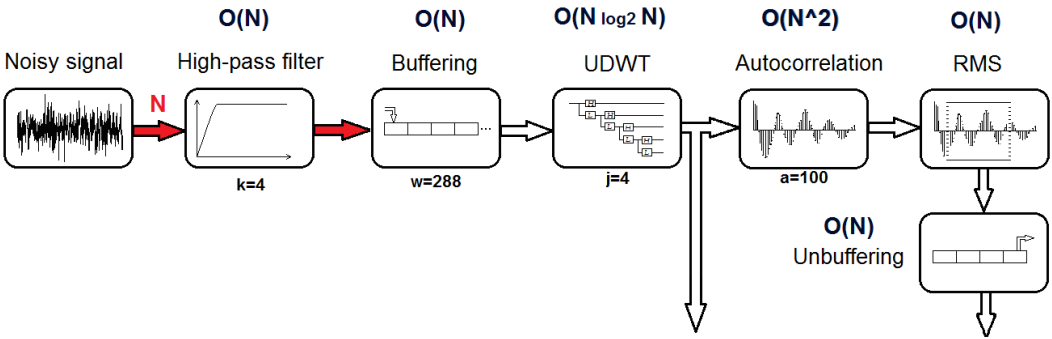
Denoising method



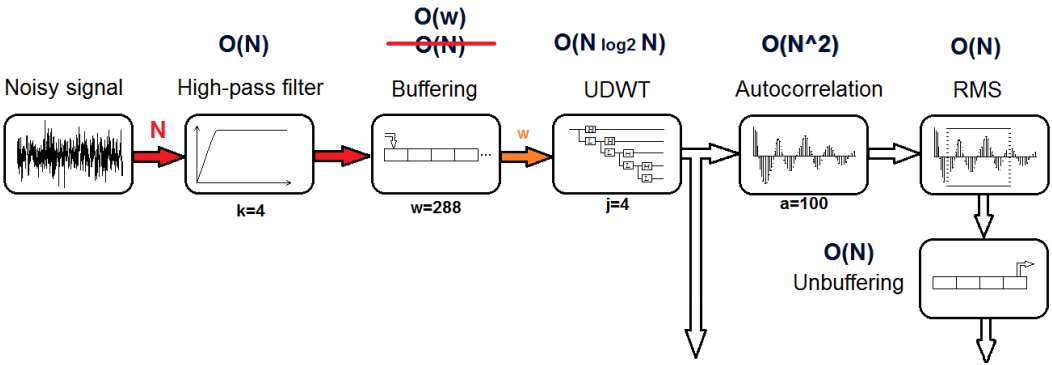
Time complexity



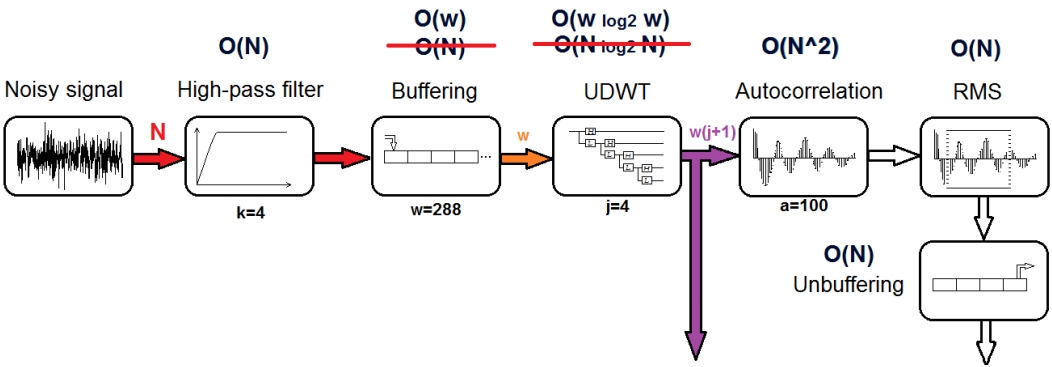
Data volume


O(N)
O(N)
O(N log2 N)
O(N^2)
O(N)
Noisy signal
High-pass filter
Buffering
UDWT
Autocorrelation
RMS
N
k=4
w=288
j=4
a=100
O(N)
Unbuffering

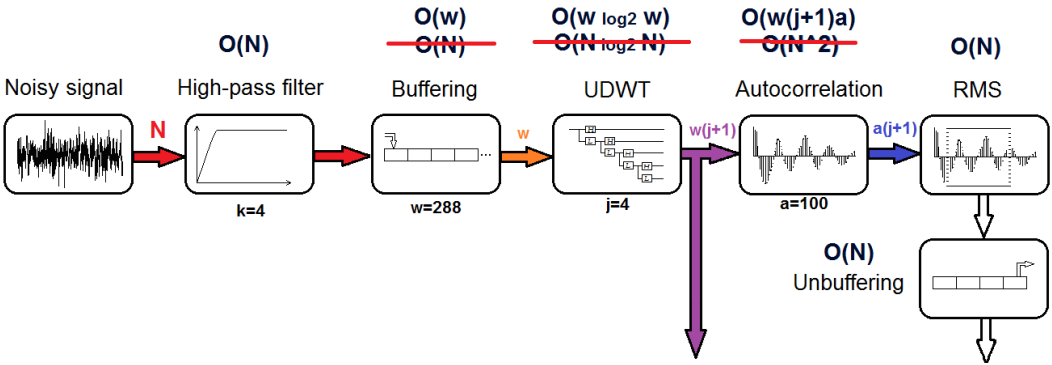
Data volume


O(N)
O(w)
O(N)
O(N log2 N)
O(N^2)
O(N)
Noisy signal
High-pass filter
Buffering
UDWT
Autocorrelation
RMS
N
w
k=4
w=288
j=4
a=100
O(N)
Unbuffering

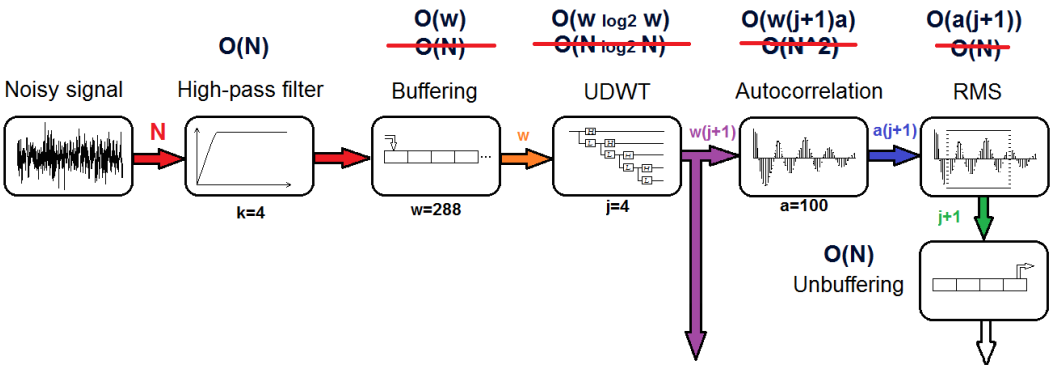
Data volume


Noisy signal
N
O(N)
High-pass filter
k=4
O(w)
O(N)
Buffering
w=288
w
O(w log2 w)
O(N log2 N)
UDWT
j=4
w(j+1)
O(N^2)
Autocorrelation
a=100
O(N)
RMS
O(N)
Unbuffering

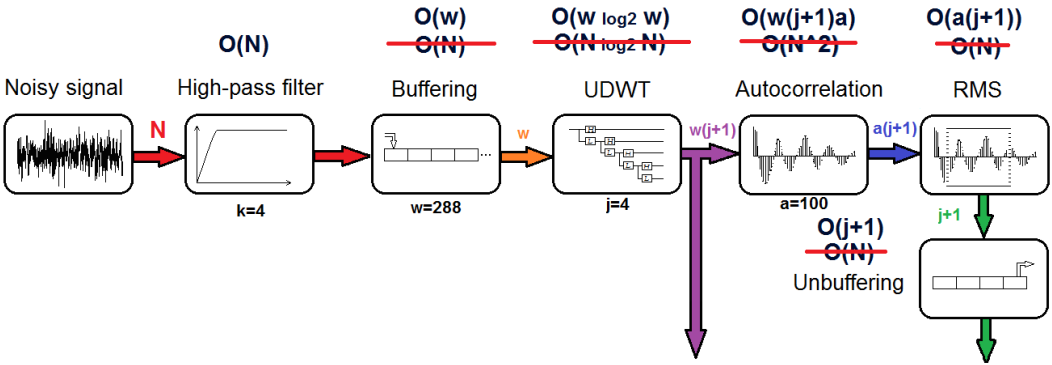
Data volume


O(N)
O(w)
O(N)
O(w log2 w)
O(N log2 N)
O(w(j+1)a)
O(N^2)
O(N)
Noisy signal
High-pass filter
Buffering
UDWT
Autocorrelation
RMS
N
k=4
w=288
w
j=4
w(j+1)
a=100
a(j+1)
O(N)
Unbuffering

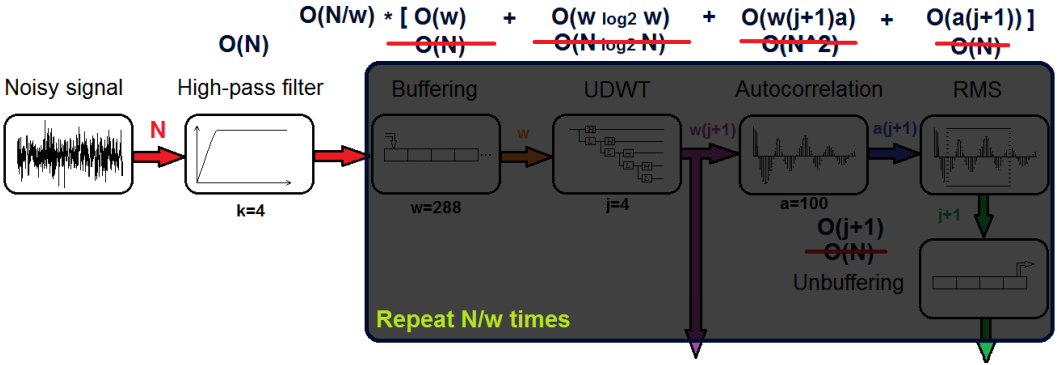
Data volume


Noisy signal
N
O(N)
High-pass filter
k=4
O(w)
O(N)
Buffering
w=288
w
O(w log2 w)
O(N log2 N)
UDWT
j=4
w(j+1)
O(w(j+1)a)
O(N^2)
Autocorrelation
a=100
a(j+1)
O(a(j+1))
O(N)
RMS
j+1
O(N)
Unbuffering

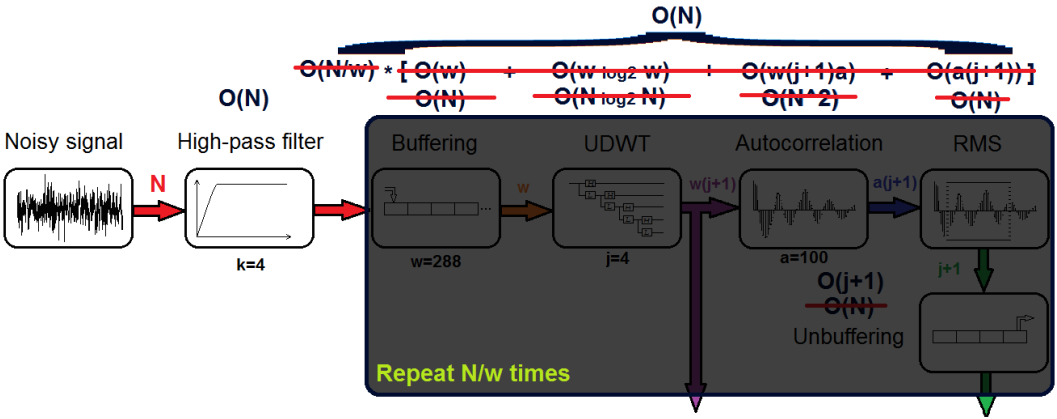
Data volume


Noisy signal
N
High-pass filter
O(N)
k=4
Buffering
O(w)
O(N)
w=288
w
UDWT
O(w log2 w)
O(N log2 N)
j=4
w(j+1)
Autocorrelation
O(w(j+1)a)
O(N^2)
a=100
a(j+1)
RMS
O(a(j+1))
O(N)
j+1
O(j+1)
O(N)
Unbuffering

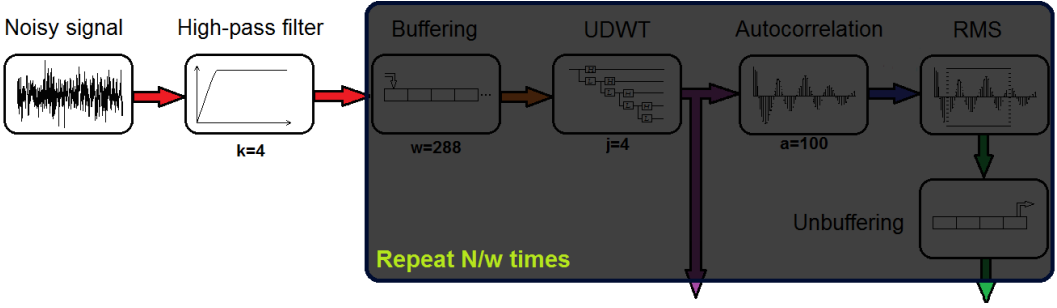
Data volume


O(N/w) * [O(w) + O(w log2 w) + O(w(j+1)a) + O(a(j+1))]
O(N) O(N) O(N log2 N) O(N^2) O(N)
Noisy signal
High-pass filter
Buffering
UDWT
Autocorrelation
RMS
N
k=4
w
w=288
j=4
w(j+1)
a(j+1)
a=100
j+1
O(j+1)
O(N)
Unbuffering
Repeat N/w times

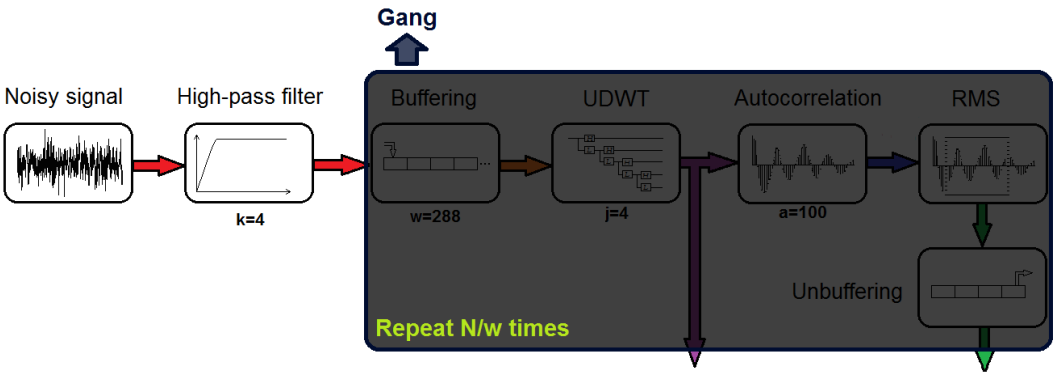
Data volume


O(N)
O(N/w) * [O(w) + O(w log2 w) + O(w(j+1)a) + O(a(j+1))]
O(N) O(N log2 N) O(N^2) O(N)
O(N)
Noisy signal
High-pass filter
N
k=4
Buffering
w=288
w
UDWT
j=4
w(j+1)
Autocorrelation
a=100
a(j+1)
RMS
j+1
O(j+1)
O(N)
Unbuffering
Repeat N/w times

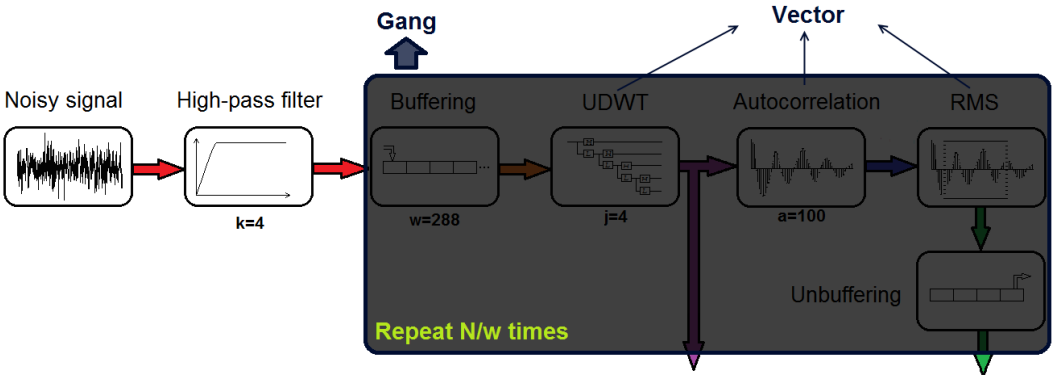
Parallelization using OpenACC



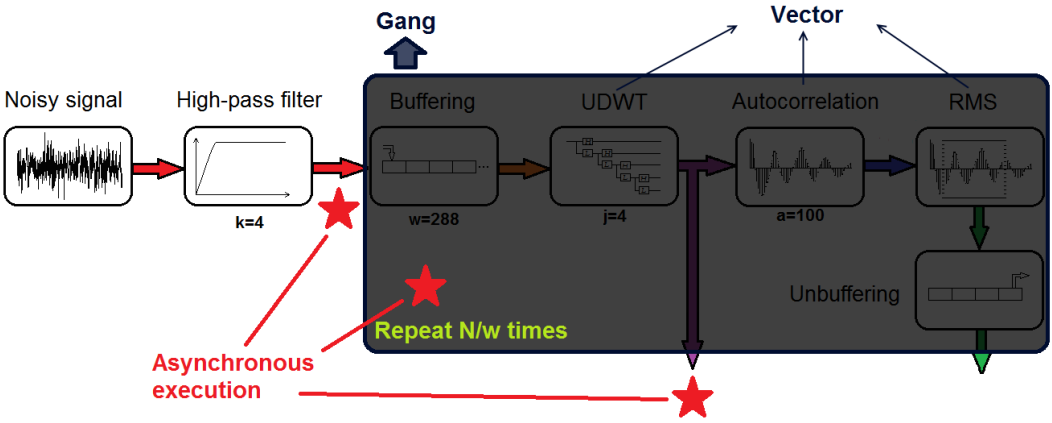
Parallelization using OpenACC



Parallelization using OpenACC



Parallelization using OpenACC



OpenACC/C++ code

```
1  //Compute the sample autocorrelation of vector "in"
2  #pragma acc routine vector
3  void autocorrvACC(double* in, int szIn, int fl, int ll, double * out,
       int szOut){
4
5      if ((ll-fl+1) <= szOut) {
6
7          double mean = 0;
8          #pragma acc loop reduction(+:mean)
9          for(int i = 0;i < szIn; i++) {
10                 mean = mean + in[i];
11         }
12
13         mean = mean / szIn;
14         double var = 0;          //Sample variance
```

OpenACC/C++ code

```
1           //Compute sample variance
2           #pragma acc loop reduction(+:var)
3           for (int i = 0; i < szIn; i++) {
4                   var += (in[i]-mean)*(in[i]-mean);
5           }
6           var = var / szIn;
7
8           //Compute sample-ACF
9           #pragma acc loop vector
10          for(int i = fl;i <= ll; i++) {// Lag loop
11                  out[i-fl] = 0;
12
13                  for(int j = i;j < szIn; j++) { //Dot product
14                    out[i-fl] += (in[j] - mean) * (in[j-i] - mean);
15            }
16                  out[i-fl] = out[i-fl] / (var * szIn);
17          }
18      }
```

OpenACC/C++ code

```
1  #pragma acc data copyin(hpd[:lpd_len],lpd[:lpd_len]), copyout(
       rms_vecACC[:ln+1][:nWs]), create(w_vecACC[:wvSz],inter_vecACC[:xSz
       ])
2  {
3      for (int q = 0; q < nBlocks; q++) { //For each block of windows
4          int begI = q * bSz * wszINT; //block beggining
5          int endI = min(begI + (bSz * wszINT),((nBlocks-1) * bSz *
               wszINT) + (resWsz * wszINT)); //block end
6
7  #pragma acc update device(inter_vecACC[begI:endI-begI]) async (q%4) //
       **ASYNC COPYIN**
8  #pragma acc parallel loop gang private(sub_vecACC[:svSz],paramsOUT[:
       wsvSz],cA[:len_X],cD[:len_X],filt[:2*lpd_len]) async (q%4) //**
       ASYNC EXECUTION**
9          for (int i = begI ;i < endI; i = i + wszINT) { //For each
               window compute the ACF-rms value in the wavelet domain
10
11             //Get subvector      (window)
```

Speed-up

Table: Ten runs using a nine-minute manatee vocalization recording.

Code	Duration (s)	Speed-up$_M$	Speed-up$_C$
Matlab	434.31 ± 0.90	X	$0.08X$
C++ Seq	35.86 ± 0.03	$12.1X$	X
Multicore	10.54 ± 0.00	$41.2X$	$3.4X$
OpenACC	5.77 ± 0.00	$75.3X$	$6.2X$
OA Async	$\mathbf{3.32 \pm 0.01}$	$\mathbf{130.8X}$	$\mathbf{10.8X}$

- CPU: Intel Xeon CPU E3-1225 v5 @ 3.30GHz (quadcore)
- GPU: Tesla k40c

References

B. Morales-Vela, L. D. Olivera-Gómez, J. E. Reynolds, and G. B. Rathbun, "Distribution and habitat use by manatees (trichechus manatus manatus) in belize and chetumal bay, méxico," *Biological Conservation*, vol. 95, pp. 67–75, 2000.

D. N. Castelblanco-Martínez, A. L. Bermúdez-Romero, I. V. Gómez-Camelo, F. C. Weber-Rosas, F. Trujillo, and E. Zerda-Ordoñez, "Seasonality of habitat use, mortality and reproduction of the vulnerable antillean manatee trichechus manatus manatus in the orinoco river, colombia: implications for conservation," *Oryx*, vol. 43, pp. 235–242, 4 2009.

E. O. Keith. (2002) Boater manatee awareness system. [Online]. Available: http://floridamarine.org/features/viewlarticle.asp?id514362

D. Gonzalez-Socoloske, L. D. Olivera-Gomez, and R. E. Ford, "Detection of free-ranging west indian manatees trichechus manatus using side-scan sonar," *Endangered Species Research*, vol. 8, pp. 249–257, Oct. 2009.

C. Niezrecki, R. Phillips, and M. Meyer, "Acoustic detection of manatee vocalizations," *Journal of the Acoustical Society of America*, vol. 114, pp. 86–91, 2003.

¡Thank you!

Maximum speed-up:

- $130.8X$ with respect to Matlab seq
- $10.8X$ with respect to C++ seq

¿Questions?

http://www.cenat.ac.cr/cnca
cnca@cenat.ac.cr