AVANTIX

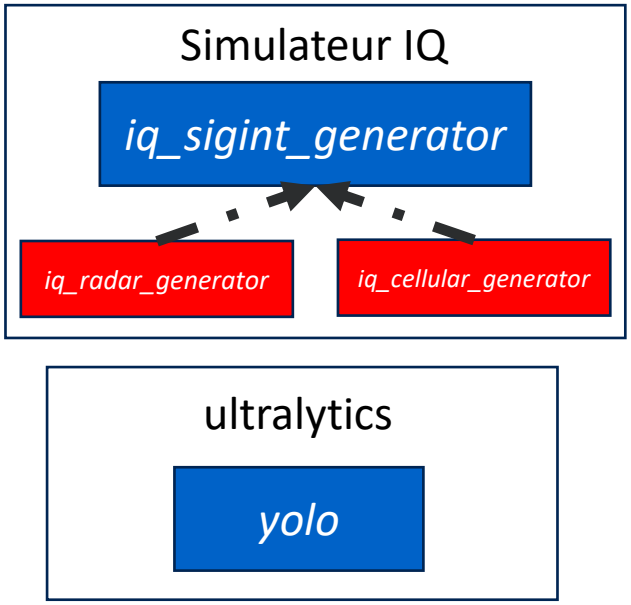
Avancement Simulateur IQ

AVANTIX

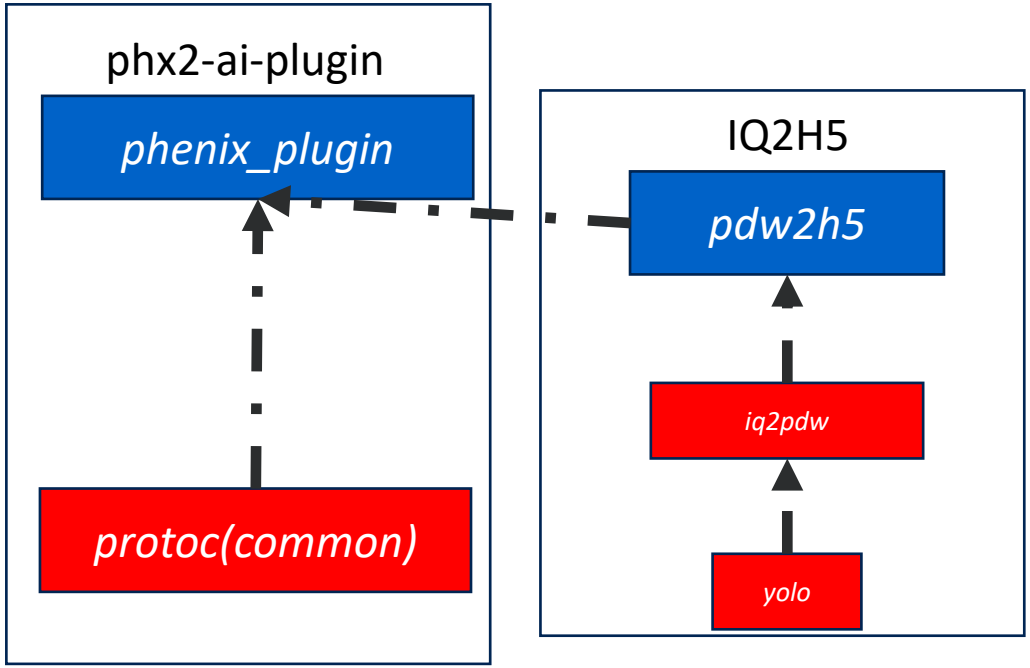
Temps différé

Architecture finale

Apprentissage



Inférence



Travaux

Dev Plugin Phénix 2

V1.0.0 plugin:

- CPU only
- Logging à faire

Meilleur modèle à obtenir → générer nouvelle BT

Démo jupyter notebook

AVANTIX

Temps contraint : SXX

10/01/2024

Méthodologie

Paramètres fixes :

- fs = 4GSps pour 3GHz de bande utile
- NPERSEGS : 128, 256, 512, 1024, 2048, 8192, 32768, 131072
- Quelle latence maximale du traitement complet ?

Travaux :

- Evaluation de l'impact de IMG_WIDTH = Latence = Nombre d'échantillons d'iq
- Evaluation de l'impact de IMG_HEIGHT
- Profiling de la fonction

Rappel fonction spectrogramme

Fonction implémentée en PyTorch

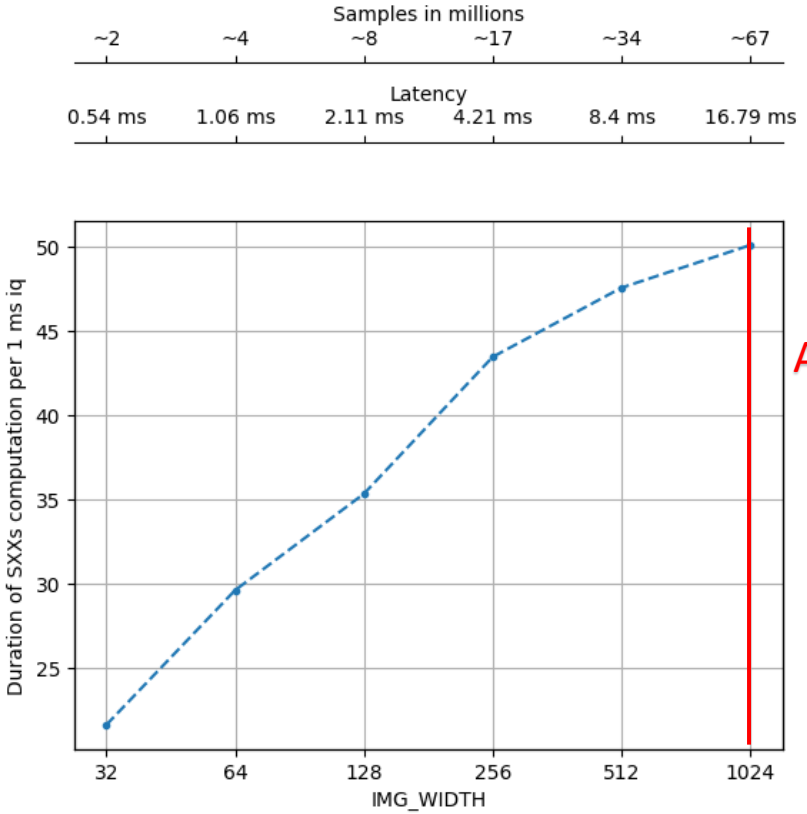
```
def sxx(iq, nperseg):
    # Pad iq pour que len(iq) % nperseg – noverlap
    # iq 1d-array -> iq 2d-array
    # iq * tukey window
    # FFT
    # ZXX = FFTSHIFT(FFT)
    # SXX = real(conj(ZXX)* ZXX)
```

Si on choisit le bon nombre d'échantillons, on évite cette étape

Impact IMG_WIDTH/Latency/Samples

<u>Paramètres expériences :</u>

- Fs=4GSps
- Npersegs = 128 → 131072
- IMG_HEIGHT = 1024

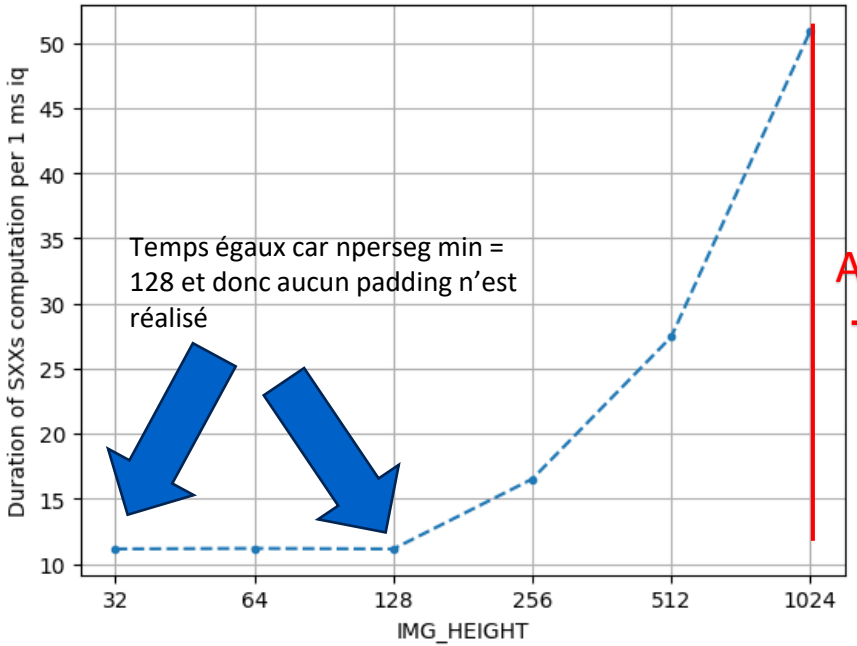


Au-delà de 1024
→Out of Video Memory

Impact IMG_HEIGHT/NFFT min

Paramètres expériences :

- Fs=4GSps
- Npersegs = 128 → 131072
- IMG_WIDTH = 1024



Au-delà de 1024
→Out of Video Memory

Profiling SXX à img_width = 256 et img_height = 1024

Temps de calcul SXX :
44,7ms / 1ms d'iq
Ou

```
test_fft.py:63: UserWarning: ComplexHalf support is experimental and many operators don't support it yet. (Triggered internally at ../aten/src/ATen/EmptyTensor.cpp:31.)
  iq = (torch.randn(n_iq, dtype=dtype) + 1j*torch.randn(n_iq, dtype=dtype)) * scale_noise
Wrote profile results to test_fft.py.lprof
Timer unit: 1e-06 s

Total time: 0.203583 s
File: test_fft.py
Function: _sxx at line 12

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    12                                           @profile
    13                                           def _sxx(sig, nperseg, noverlap, nfft, window=None):
    14                                               """
    15                                               Same func as from iq2pdw import spectrogram
    16                                               """
    17         8          4.2      0.5      0.0      nstep = nperseg - noverlap
    18         8         44.4      5.6      0.0      num_to_pad = (-(sig.shape[-1]-nperseg) % nstep) % nperseg
    19         8          2.6      0.3      0.0      if num_to_pad > 0:
    20                                                   print("num_pad", nperseg)
    21                                                   sig = torch.cat([sig, torch.zeros(num_to_pad, device=sig.device, dtype=sig.real.dtype)])
    22
    23                                               # Reshape for vectorized computation
    24         8        281.9     35.2      0.1      sig = sig.unfold(0, nperseg, nstep).T
    25
    26                                               # Windowing
    27         8          1.8      0.2      0.0      if window is None:  # Default (tukey, alpha=0.25)
    28         8      71751.9   8969.0     35.2          window = torch.tensor(tukey(nperseg, 0.25), device=sig.device, dtype=sig.real.dtype)
    29         8       3074.0    384.2      1.5      sig = sig * window[:, None]
    30
    31                                               # zero padding: TODO a discuter en réunion
    32         8          4.0      0.5      0.0      if nfft > nperseg:
    33         3      12257.7   4085.9      6.0          sig = torch.cat([sig, torch.zeros(nfft-nperseg, sig.size(1), device=sig.device, dtype=sig.real.dtype)], dim=0)
    34
    35                                               # FFT
    36         8      21607.5   2700.9     10.6      zxx = torch.fft.fft(sig, dim=0)
    37         8      88538.1  11067.3     43.5      zxx = torch.fft.fftshift(zxx, axis=0)
    38         8       6012.2    751.5      3.0      sxx = (torch.conj(zxx) * zxx).real
    39
    40         8          2.6      0.3      0.0      return sxx
```

GPU

H100

Graphics Processor	
GPU Name:	GH100
Architecture:	Hopper
Foundry:	TSMC
Process Size:	5 nm
Transistors:	80,000 million
Density:	98.3M / mm²
Die Size:	814 mm²

V100

Graphics Processor	
GPU Name:	GV100
Architecture:	Volta
Foundry:	TSMC
Process Size:	12 nm
Transistors:	21,100 million
Density:	25.9M / mm²
Die Size:	815 mm²

GPU

H100

Graphics Card	
Release Date:	Mar 21st, 2023
Generation:	Tesla Hopper (Hxx)
Predecessor:	Tesla Ada
Successor:	Tesla Blackwell
Production:	Active
Bus Interface:	PCIe 5.0 x16

V100

Graphics Card	
Release Date:	Mar 27th, 2018
Generation:	Tesla Volta (Vxx)
Predecessor:	Tesla Pascal
Successor:	Tesla Turing
Production:	End-of-life
Bus Interface:	PCIe 3.0 x16

GPU

H100

Memory	
Memory Size:	80 GB
Memory Type:	HBM2e
Memory Bus:	5120 bit
Bandwidth:	2.04 TB/s

Clock Speeds	
Base Clock:	1095 MHz
Boost Clock:	1755 MHz
Memory Clock:	1593 MHz 3.2 Gbps effective

V100

Memory	
Memory Size:	32 GB
Memory Type:	HBM2
Memory Bus:	4096 bit
Bandwidth:	897.0 GB/s

Clock Speeds	
Base Clock:	1230 MHz
Boost Clock:	1380 MHz
Memory Clock:	876 MHz 1752 Mbps effective

GPU

H100

Theoretical Performance	
Pixel Rate:	42.12 GPixel/s
Texture Rate:	800.3 GTexel/s
FP16 (half):	204.9 TFLOPS (4:1)
FP32 (float):	51.22 TFLOPS
FP64 (double):	25.61 TFLOPS (1:2)

V100

Theoretical Performance	
Pixel Rate:	176.6 GPixel/s
Texture Rate:	441.6 GTexel/s
FP16 (half):	28.26 TFLOPS (2:1)
FP32 (float):	14.13 TFLOPS
FP64 (double):	7.066 TFLOPS (1:2)

GPU

H100

Render Config	
Shading Units:	14592
TMUs:	456
ROPs:	24
SM Count:	114
Tensor Cores:	456
L1 Cache:	256 KB (per SM)
L2 Cache:	50 MB

V100

Render Config	
Shading Units:	5120
TMUs:	320
ROPs:	128
SM Count:	80
Tensor Cores:	640
L1 Cache:	128 KB (per SM)
L2 Cache:	6 MB

AVANTIX

MERCI