GPU pour PAON4/Idrogen

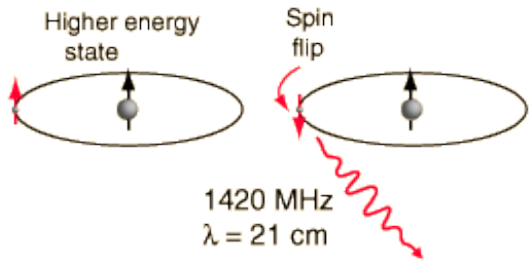
O. Perdereau, R. Ansari

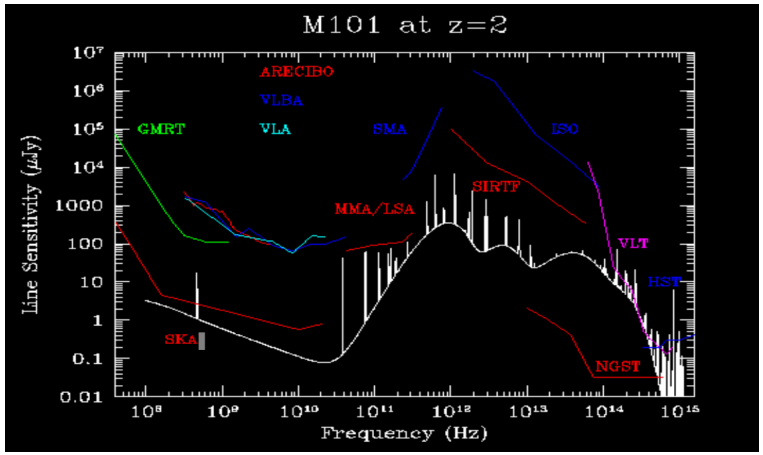
WORK IN PROGRESS

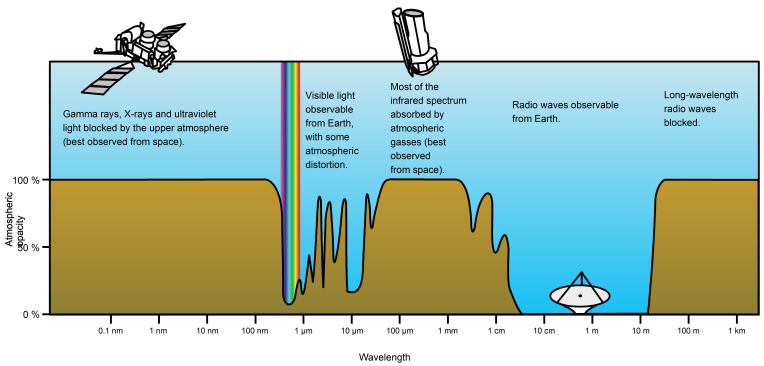
1/12/2020

The 21 cm H line

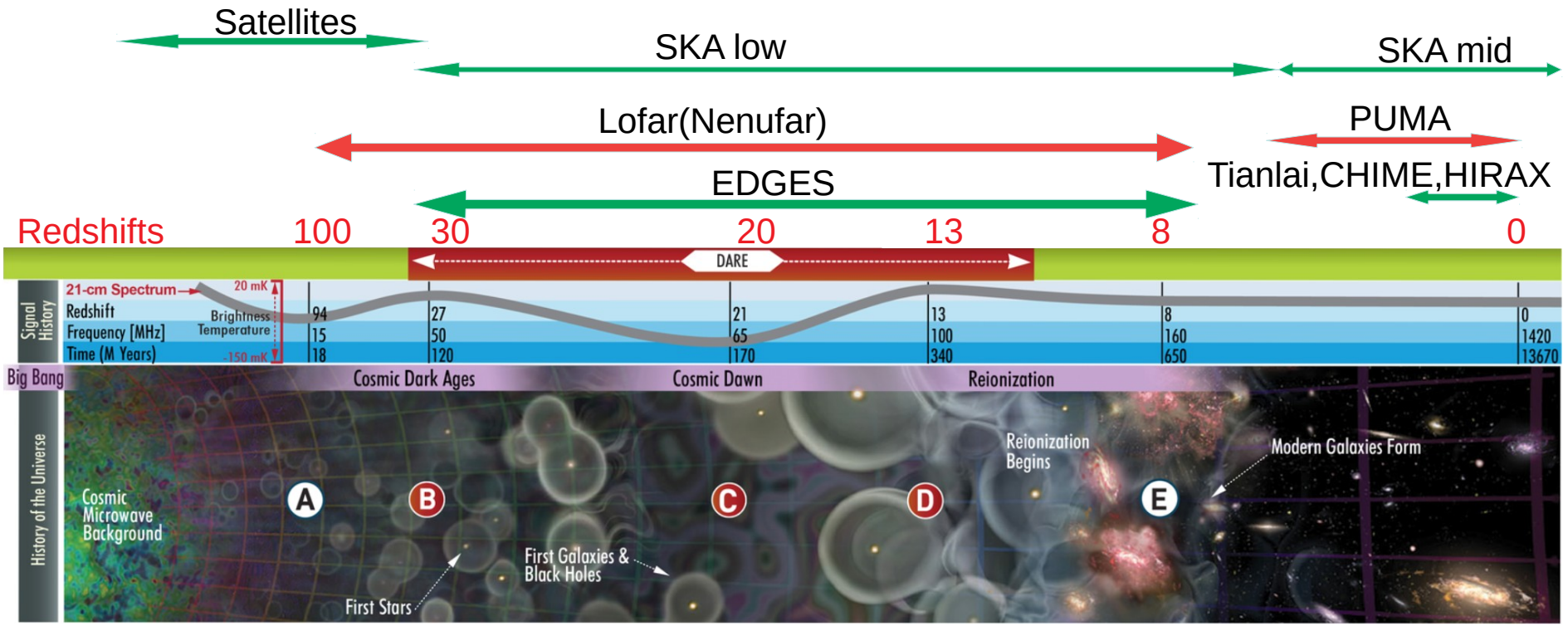
- introduced in astro (radio) in 1944
- ¨isolated¨ line ⇒ enables tomography : $z \Leftrightarrow \nu$
- ground observations possible down to ∼ 30 MHz (ionosphere) i.e. $z \sim 30$
  NB human-made perturbations (3/4/5G, TNT, FM, radars, ..)







21 cm in cosmology



Planck, S4, Litebird

Euclid

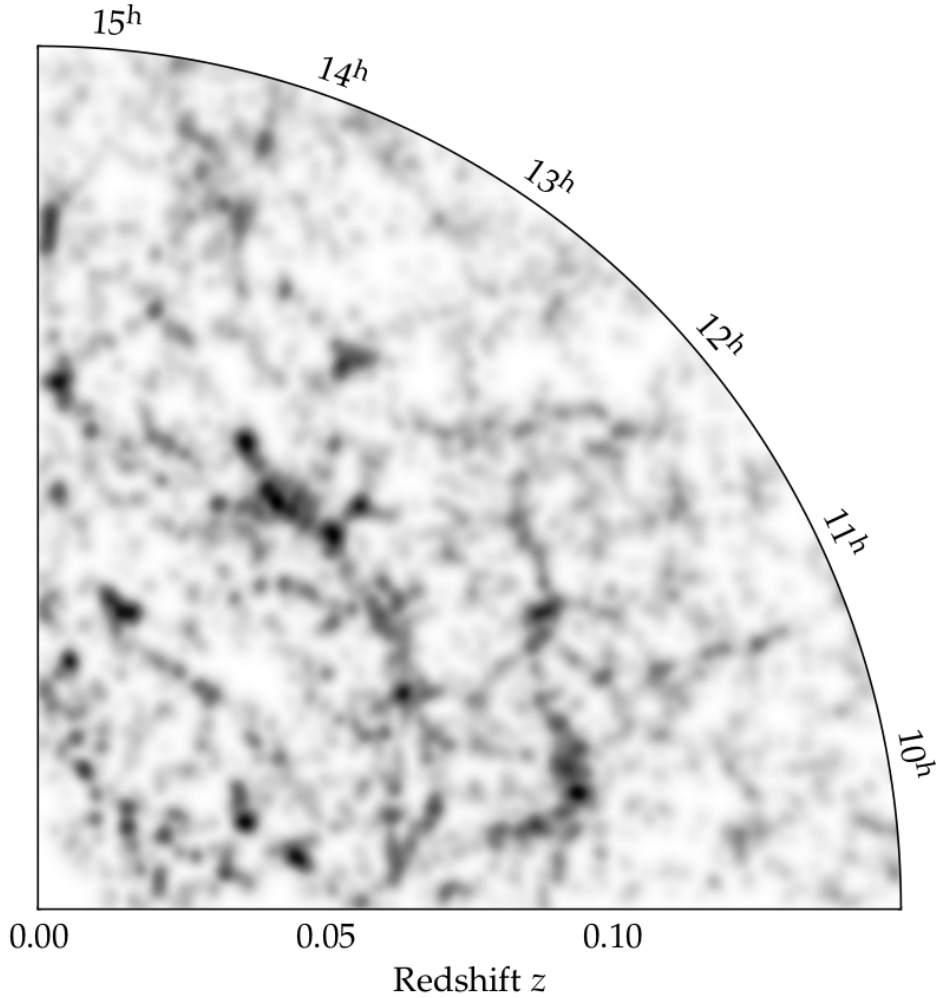
LSST,DESI

MSE

Two aspects : global spectrum shape & spatial anisotropies

LSS from 21 cm line Intensity Maping (IM)

- broad ν band, low angular resolution instrument
- measure intensity at “each” frequency $\Rightarrow z$: “easy” tomography
- prices to pay :
  - low signal (detected up to now only in X-corr with surveys)
  - very high level of foregrounds
  - DAQ & calibration challenging
  - cosmological analysis : HI power spectrum bias wrt matter’s



J. Petterson et al. arXiv:0902.3091

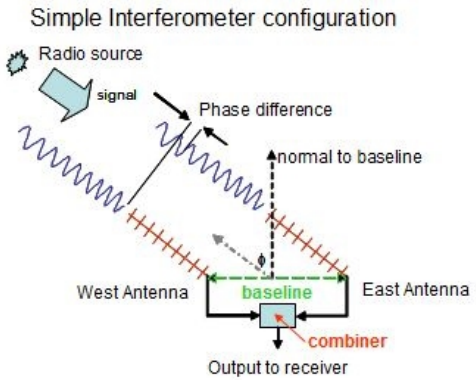
The idealized setup

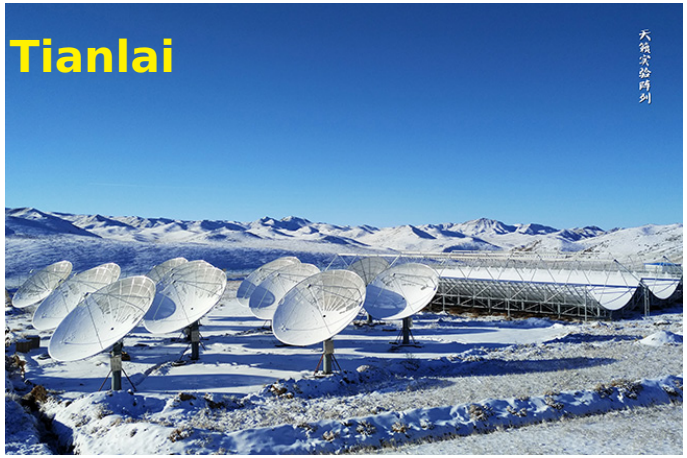
(some) Requirements :

- large collecting surface (S/N ratio)
- large sky area
- moderate (0.1 deg) angular resolution
- broad ν range (large volume)

⇒ tentative solution(s) :

- packed interferometric array (angular res., high S/N at BAO scale)
- (semi) fixed antenna(s) in transit mode (large sky coverage, cost)
- sampling of full signal waveform → FFT, digital correlation, beam forming

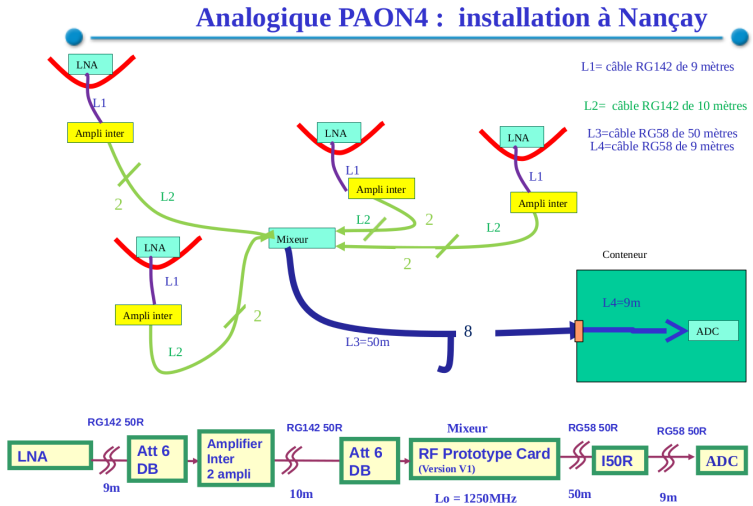




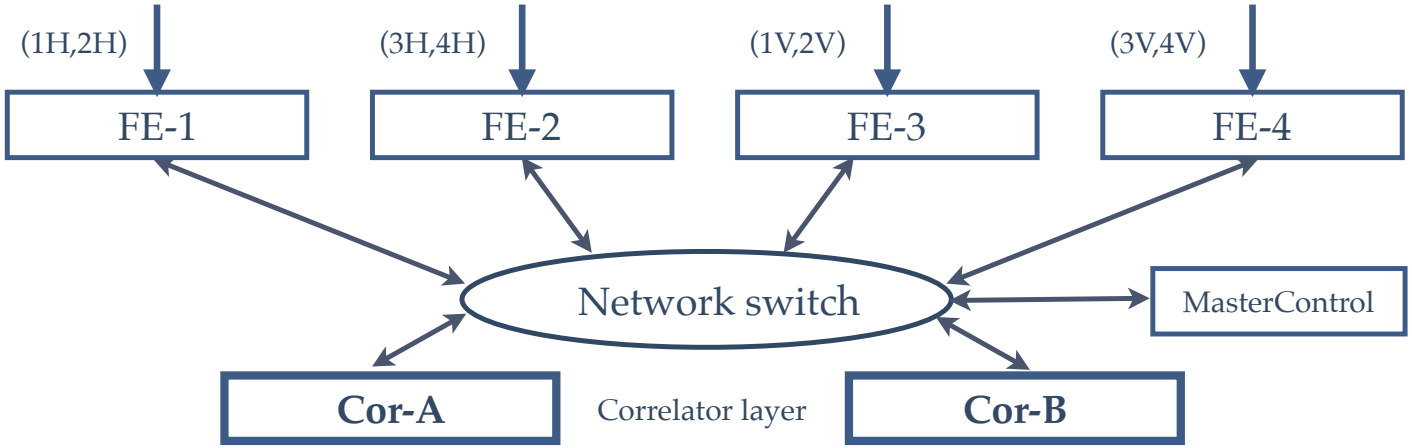
PAON4

Collaboration between LAL, Obs. de Paris (Meudon, Nançay), IRFU
Characteristics :

- 4 dishes (∼ 3 deg beams) in Nancay (∼ 200 km south of Paris)
- 2 polar./antenna
- Frequency band 1250 - 1500 MHz (∼ 1275 - 1480 MHz fiducial)
- ± 20 degrees from zenith
- transit observations ; ∼ 24h scans (fixed elevation) since 2015
- test bench for electronics, DAQ, on-line computing, analysis
- J.E. Campagne et al., MNRAS in press



Le corrélateur de PAON4



- Corrélateur soft de PAON 4 $\Rightarrow$ efficacité sur ciel $\lesssim 10\%$
- Upgrade prévu : 2 nouvelles machines avec GPU (back-end)
- objectif : gagner un facteur 4-6 d'efficacité

Etapes du processing

- les machines du Front-End recoivent des blocs d'échantillons : structure $S(t_n)$ ou $A(\nu_k)$ si FFT en amont
- enventuellement calculs des FFT
- séparation en 2 intervalles de fréquence et organisation
- envoi aux corrélateurs
- dans chaque corrélateur : calcul des cross/auto-corrélations entre feeds pour chaque bloc
- et moyenne de ces blocs dans un intervalle de temps (paramètre du système) :

$$\sum_n a_i^n[\nu_k] a_j^n[\nu_k)]^*$$

Premiers tests

`pc-bao2` : Dell PowerEdge 740 (40 Intel(R) Xeon(R) Silver 4210 @2.20GHz) + GPU Nvidia Quadro P4000 (8 Go)
`serveur-paon4` : idem mais Tesla T4 (16 Go)

- calculs sur CPU
- calculs sur GPU :
  - OpenCL : standard ouvert conçu pour programmer des systèmes parallèles hétérogènes CPU+GPU
  - CUDA : API e.g. C(++) pour calculer sur GPU NVidia
  - thrust : analogue parallèle (CPU et/ou GPU) de la STL - NVIdia
  - CuBLAS : implémentation de BLAS pour GPU - NVidia (CUDA)

Mon petit test (1)

- (cross)-corrélation = produit élément par élement de 2 vecteurs `thrust avec transform + fonction ad-hoc`

(...)

```
struct pxycon_functor
{
  __host__ __device__
  thrust::complex<float> operator()(const thrust::complex<float> & x,
  const thrust::complex<float> & y) const {
                return x*thrust::conj(y);
  }
};
```

(...)

```
thrust::transform( x.begin() , x.end()  ,
                y.begin(), output.begin() , pxycon_functor() );
```

(...)

⇒ remplit le `thrust::vector` avec $x \times y^*$

Mon petit test (2)

- moyenne de n cross-corrélations $\Leftrightarrow$ produit matrice vecteur :

$$\begin{pmatrix} a_i^0[\nu_0]a_j^0[\nu_0]^* & \dots & a_i^n[\nu_0)]_j^n[\nu_0]^* \\ \vdots & \ddots & \vdots \\ a_i^0(\nu_M)a_j^0[\nu_M]^* & \dots & a_i^n[\nu_M]a_j^n[\nu_M]^* \end{pmatrix} \begin{pmatrix} 1/n \\ \vdots \\ 1/n \end{pmatrix} \quad (1)$$

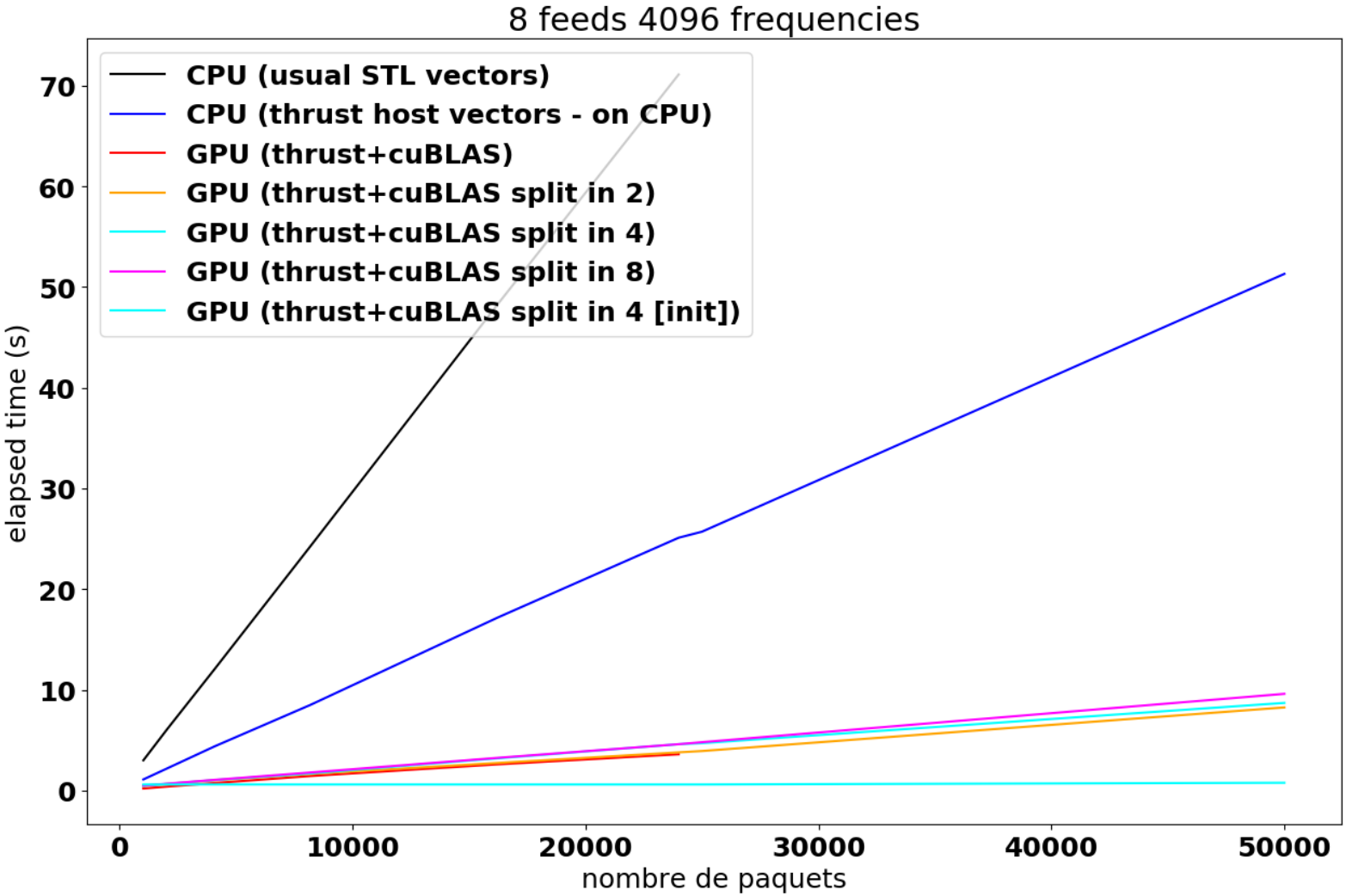
$\Rightarrow$ utilisation d'une fonction de cuBLAS : `cublasCgemv` dérivée de `gemv (BLAS)` : produit matrice vecteur générique

(NB : demande quelques "cast" pour les containers vs thrust : :vector)

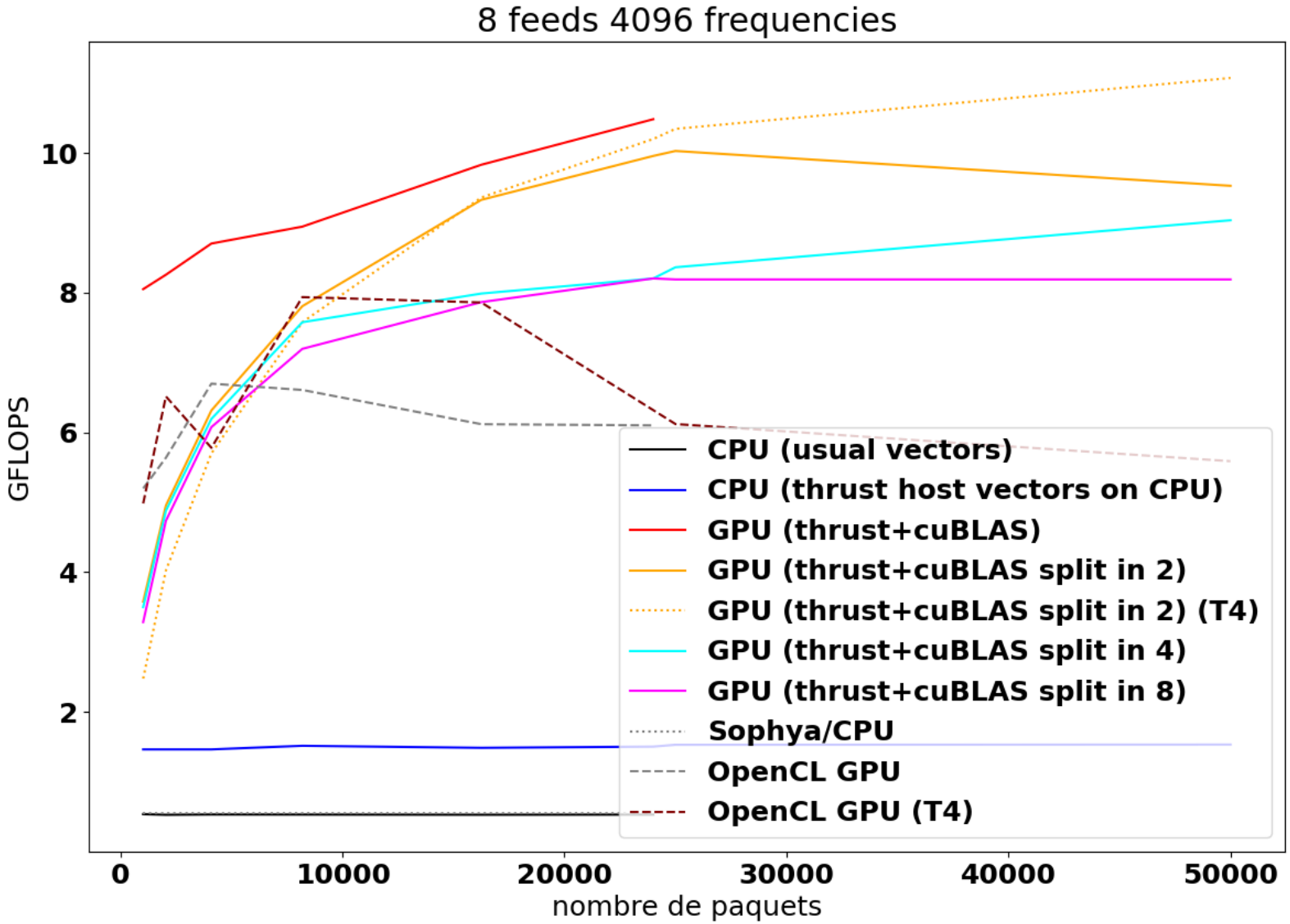
quelques paramètres

- 8 feeds dans PAON4
- data streams simulés comme des gaussiennes de moyenne 2 et sigma 3.5 (arbitraire !)
- chaque paquet de "FTT" comprend 4096 coefs complexes
- `float` précision
- je fait varier le nombre de paquets traités
- éventuellement traités en plusieurs blocs (mémoire du GPU)

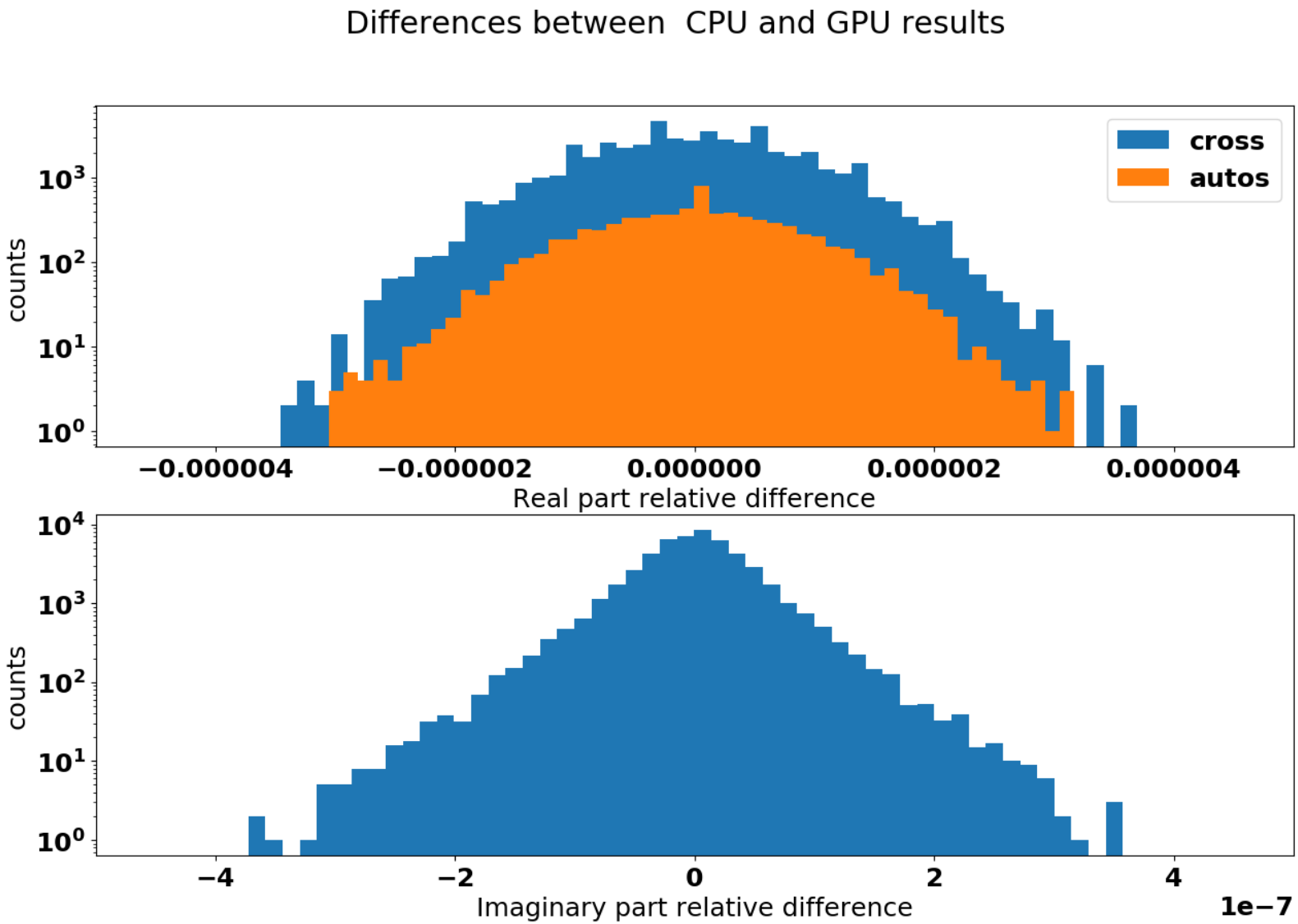
Performances (1)



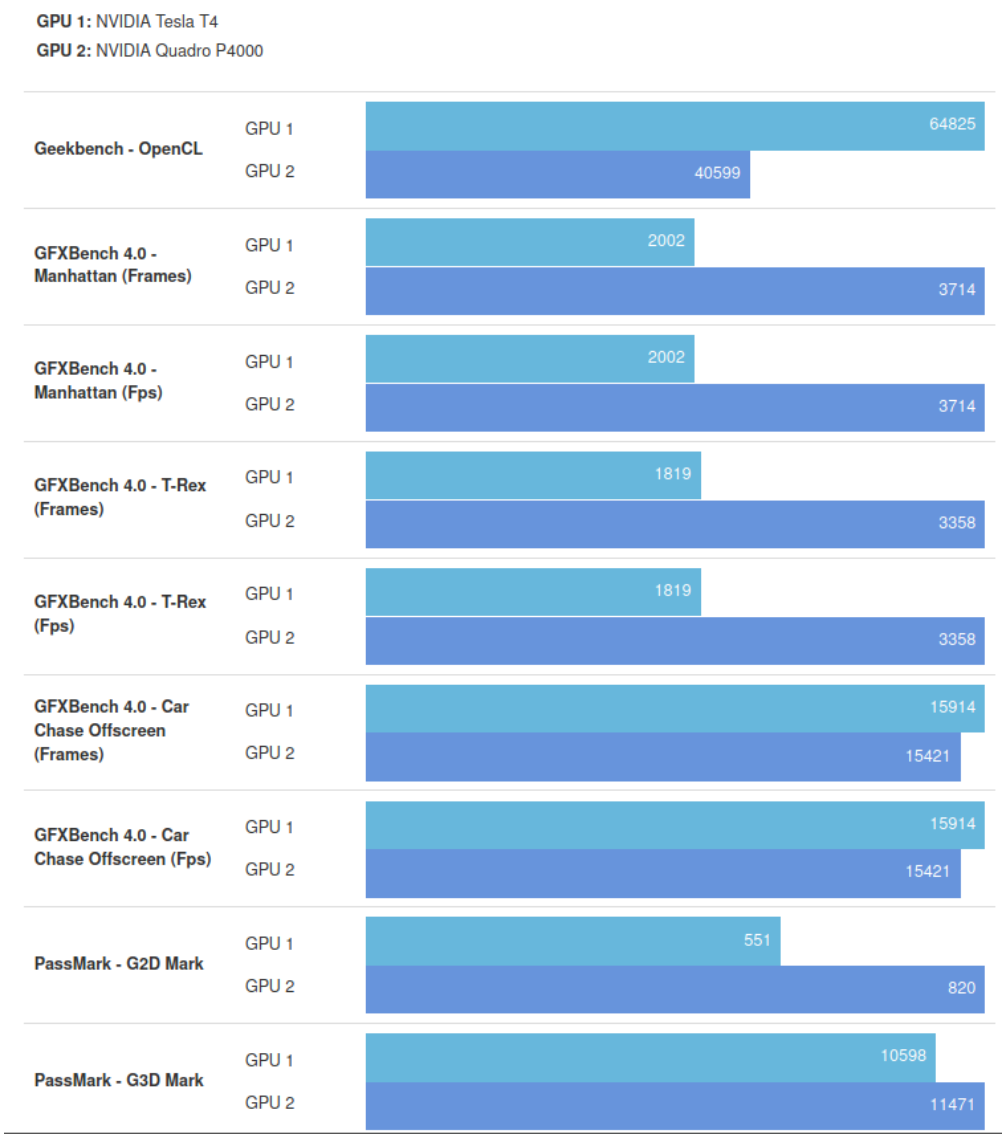
Performances (2)



Des résultats (un peu) différents



Une comparaison entre les GPUs



Petit bilan

- Sur PAON4 :
  - 8 coeurs `Intel(R) Xeon(R) CPU E5620 @2.40GHz`, `15Go` RAM
  - 4 threads
  - 432 MFLOP/s sur chacun
- `pc-bao2 / serveur-paon4` :
  - 40 coeurs `Intel(R) Xeon(R) Silver 4210 CPU @2.20GHz`, 62 Go RAM
  - Avec un GPU NVIDIA Quadro P4000 ou Testla T4
  - On arrive a $\sim$ 8 GFLOP/s
- En ajoutant un second GPU on devrait gagner le facteur voulu (largement ?)
- Diff'erence P4000 vs Tesla T4 pas énorme ?