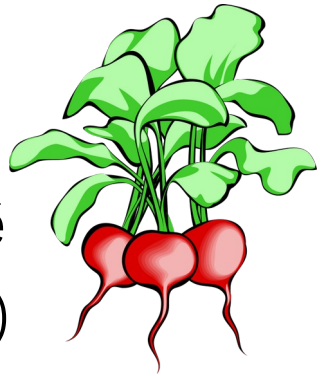
Conférence ASNUM2022

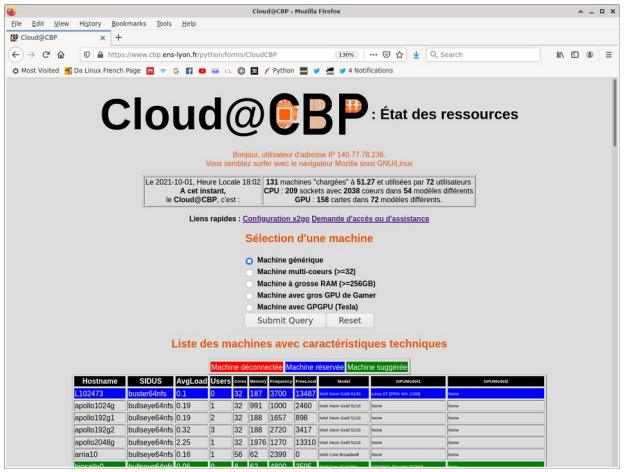
Portage d'un vieux code astrophysique sur CPU ou GPU récentes : retour d'expérience

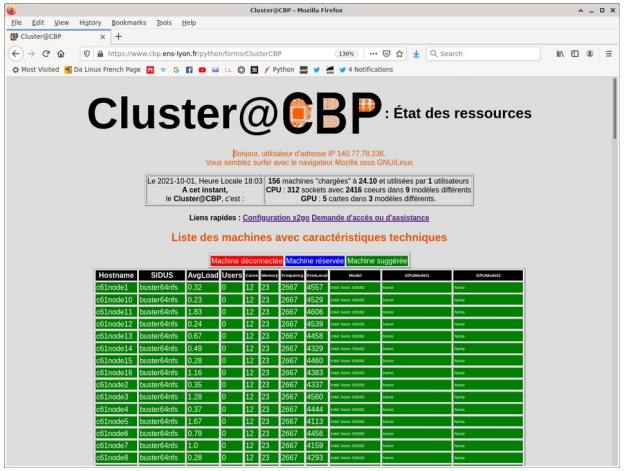
Le Centre Blaise Pascal son évolution depuis 2008

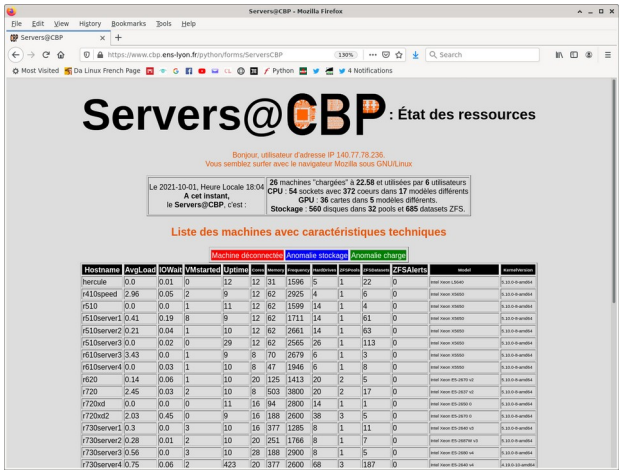
- A l'origine : « Maison de la modélisation »
- Evolution de « hôtel à conférences, personnes ou projets » en :
  - **Plateaux techniques multiples** : CPU, GPU, clusters, stockages, …
  - **Centre d'essais** : pour le centre de calcul local PSMN mais plus encore...
  - **Computhèque** : toute l'histoire « animée » de l'informatique depuis 50 ans
- Ses propriétés affirmées : être un « RADIS »
  - **R**eproductibilité : avec les machines de tous âges et SIDUS
  - **A**daptabilité : avec une internalisation complète offrant de la flexibilité
  - **D**iversité : avec ses centaines de types CPU ou GPU (47, 76 actives)
  - **I**ntéractivité : avec sa prise en main « comme devant l'écran et le clavier »
  - **S**implicité : d'un déploiement universel systématique vers un usage systémique



Le Centre Blaise Pascal : Ressources matérielles & humaines







- Plus de 300 machines « actives » : Cloud, Cluster, Servers
  - **Cloud** : postes de salles de formation, stations de bureau, serveurs rackables, machines « ouvertes », machines virtuelles, …
  - **Clusters** : de 12 à 64 nœuds, de 192 à 1152 cœurs
- Un « couple alpha » pour s'occuper de cette « meute »…

40 ans entre simulation et mesure
De 2019 à 1979...

1979 : JP Luminet A&A

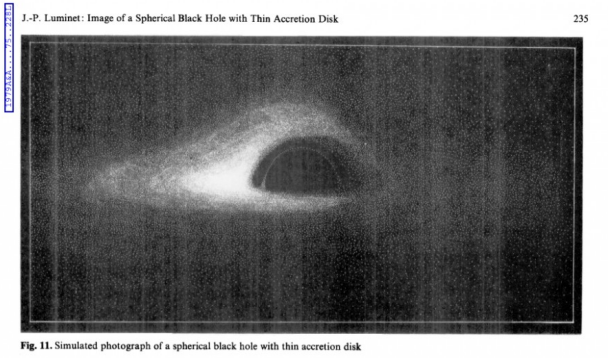
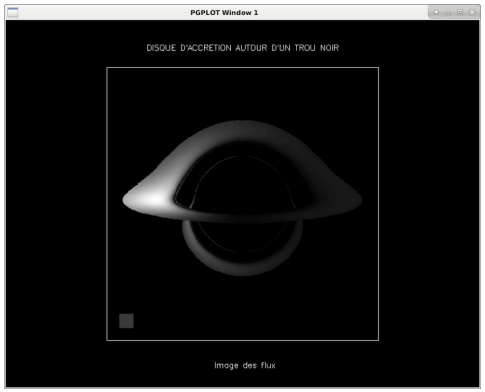

J.-P. Luminet: Image of a Spherical Black Hole with Thin Accretion Disk 235

Fig. 11. Simulated photograph of a spherical black hole with thin accretion disk

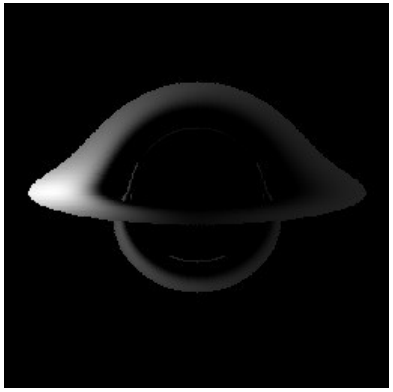
2014 : Film Interstellar

2019 : EHT, Messier 87

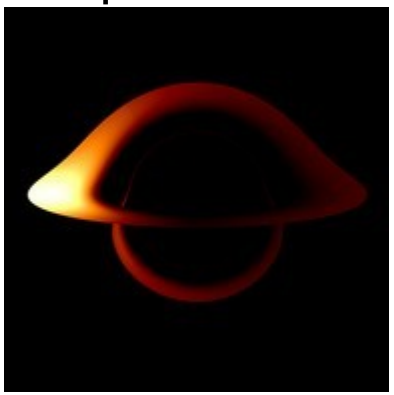
1994 : Code Fortran



1997 : Code C



2019 : OpenCL/CUDA



La physique de base
Tout dans un article de JP Luminet !

- Relativité générale d'Einstein
- Une métrique de Schwarzschild
- Réduction en équation polaire
- Dérivation de l'équation polaire
- Système du second ordre
- Modèle d'émission de disque
  - Raie monochromatique : cas d'école
  - Corps noir : modèle plus réaliste

Astron. Astrophys. 75, 228–235 (1979)

ASTRONOMY AND ASTROPHYSICS

Image of a Spherical Black Hole with Thin Accretion Disk

J.-P. Luminet

Groupe d'Astrophysique Relativiste, Observatoire de Paris, Section d'Astrophysique, F-92190-Meudon, France

Received July 13, 1978

Summary. Black hole accretion disks are currently a topic of widespread interest in astrophysics and are supposed to play an important role in a number of high-energy situations. The present paper contains an investigation of the optical appearance of a spherical black hole surrounded by thin accretion disk. Isoradial curves corresponding to photons emitted at constant radius from the hole as seen by a distant observer in arbitrary direction have been plotted, as well as spectral shifts arising from gravitational and Doppler shifts. By the results of Page and Thorne (1974) the relative intrinsic intensity of radiation emitted by the disk at a given radius is a known function of the radius only, so that it is possible to calculate the exact distribution of observed bolometric flux. Direct and secundary images are plotted and the strong asymmetry in the flux distribution due to the rotation of the disk is exhibited. Finally a simulated photograph is constructed, valid for black holes of any mass accreting matter at any moderate rate.

Key words: black holes – accretion disks – geometrical optics

1. Introduction

The aim of the present paper is to provide a reply to the question that many people ask themselves about the optical appearance of a black hole.

In order to be visible a black hole has of course to be illuminated, like any ordinary body. One of the simplest possibilities would be for the black hole to be illuminated by a distant localized source which in practise might be a companion star in a loosely bound binary system. A more interesting and observationally important possibility is that in which the light source is provided by an emitting accretion disk around the black hole, such as may occur in a tight binary system with overflow from the primary, and perhaps also on a much larger scale in a dense galactic nucleus. The general problem of the optical appearance of black holes is related to the analysis of trajectories in the gravitational field of black holes. For a spherical, static, electrical field-free black hole (whose external space-time geometry is described by the Schwarzschild metric) this problem is already well known (Hagihara, 1931; Darwin, 1959; for a summary, see Misner et al., 1973 [MTW]). In Sect. 2 we give only a brief outline of it with basic equations, trying to point out the major features which will appear later. All our calculations are done in the geometrical optics approximation (for a study of wave-aspects, see Sanchez, 1977). In Sect. 3 we calculate the apparent shape of circular rings orbiting a non-rotating black hole and the results are depicted in Figs. 5–6. In Sect. 4 we recall the standard analysis by Novikov and Thorne (1973) of the problem of energy release by a thin accretion disk in a general astrophysical context, focusing attention more particularly on the analytic solution for the surface distribution of energy release that was derived by Page and Thorne (1974) in the limiting case of a sufficiently low accretion rate. In terms of this idealized (but in appropriate circumstances, realistic) model, we calculate the distribution of bolometric flux as seen by distant observers at various angles above the plane of the disk (Figs. 9–11).

2. Image of a Bare Black Hole

Before analyzing the general problem of a spherical black hole surrounded by an emitting accretion disk, it is instructive to investigate a more simple case in which all the dynamics are already contained, namely the problem of the return of light from a bare black hole illuminated by a light beam projected by a distant source. It is conceptually interesting to calculate the precise apparent pattern of the reflected light, since some of the main characteristic features of the general geometrical optics problem are illustrated thereby.

The Schwarzschild metric for a static pure vacuum black hole may be written as:

$$ds^2 = -\left(1-\frac{2M}{r}\right)dt^2 + \left(1-\frac{2M}{r}\right)^{-1}dr^2 + r^2(d\theta^2+\sin^2\theta\, d\phi^2) \quad (1)$$

where r, θ, and ϕ are spherical coordinates and the unit system is chosen such that $G=c=1$. M is the relativistic mass of the hole (which has the dimensions of length). In this standard coordinate system the horizon forming the surface of the hole is located at the Schwarzschild radius $r_s=2M$.

One can take advantage of the spherical symmetry to choose the "equatorial" plane $\theta=\pi/2$ so as to contain any particular photon trajectory under consideration. The trajectories will then satisfy the differential equation:

$$\left\{\frac{1}{r^2}\left(\frac{dr}{d\phi}\right)\right\}^2 + \frac{1}{r^2}\left(1-\frac{2M}{r}\right) = 1/b^2. \quad (2)$$

The second term in the left member can be interpreted as an effective potential $V(r)$, in analogy with the non-relativistic mechanics. The motion does not depend on the photon energy E and on its angular momentum L separately, but only on the ratio $L/E=b$, which is the impact parameter at infinity.

Let the observer be in a direction fixed by the polar angle ϕ_0 in the Schwarzschild metric, at a radius $r_0 \gg M$. The rays emitted by a distant source of light and deflected by the black hole intersect the observer's detector (for example a photographic plate) at a

De l'article au rapport

- Métrique de Schwarzschild :

$$ds^2 = -\left(1 - \frac{2M}{r}\right) dt^2 + \left(1 - \frac{2M}{r}\right)^{-1} dr^2 + r^2(d\theta^2 + sin^2\theta d\phi^2)$$

- Equation polaire :

$$\left(\frac{1}{r^2}\left(\frac{dr}{d\phi}\right)\right)^2 + \frac{1}{r^2}\left(1 - \frac{2M}{r}\right) = \left(\frac{\pi_t}{\pi_\phi}\right)^2 = \frac{1}{b^2}$$

- Changement de coordonnées : *u=1/r*

$$\left(\frac{du}{d\phi}\right)^2 + u^2\left(1 - \frac{2Mu}{b}\right) = 1$$

- Dérivation de l'équation polaire :

$$\frac{d^2u}{d\phi^2} + u\left(1 - \frac{3M}{b}u\right) = 0$$

- Système d'équations à résoudre : $v = \frac{du}{d\phi}$ et $\frac{dv}{d\phi} = 3\frac{m}{b}u^2 - u$

DISQUE D'ACCRETION
AUTOUR D'UN TROU NOIR

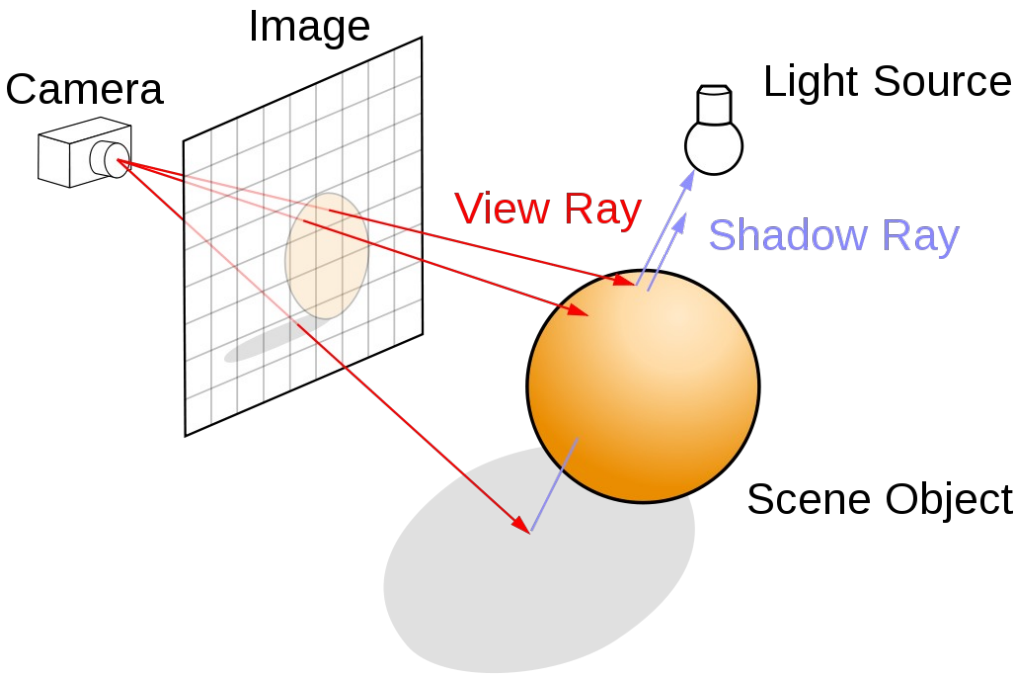
IMAGE & SPECTRE

Hervé Aussel
Emmanuel Quémener

Travail réalisé dans le cadre des travaux pratiques de
"Modélisation numérique"

Rapport du 2 mars 1994

Le « lancer de rayons » : Remonte le temps, forme une image



Source Wikipedia : Mental Ray CC BY-SA 3.0, Henrik CC BY-SA 4.0

Echarpe de plasma autour du Trou Noir
Pas « sans » mais « avec » dessus-dessous...

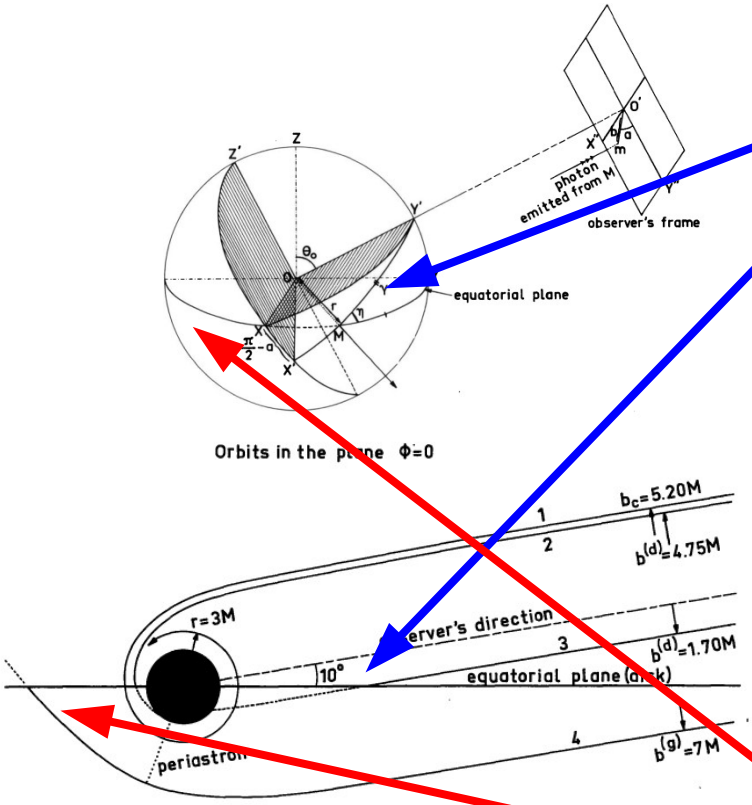
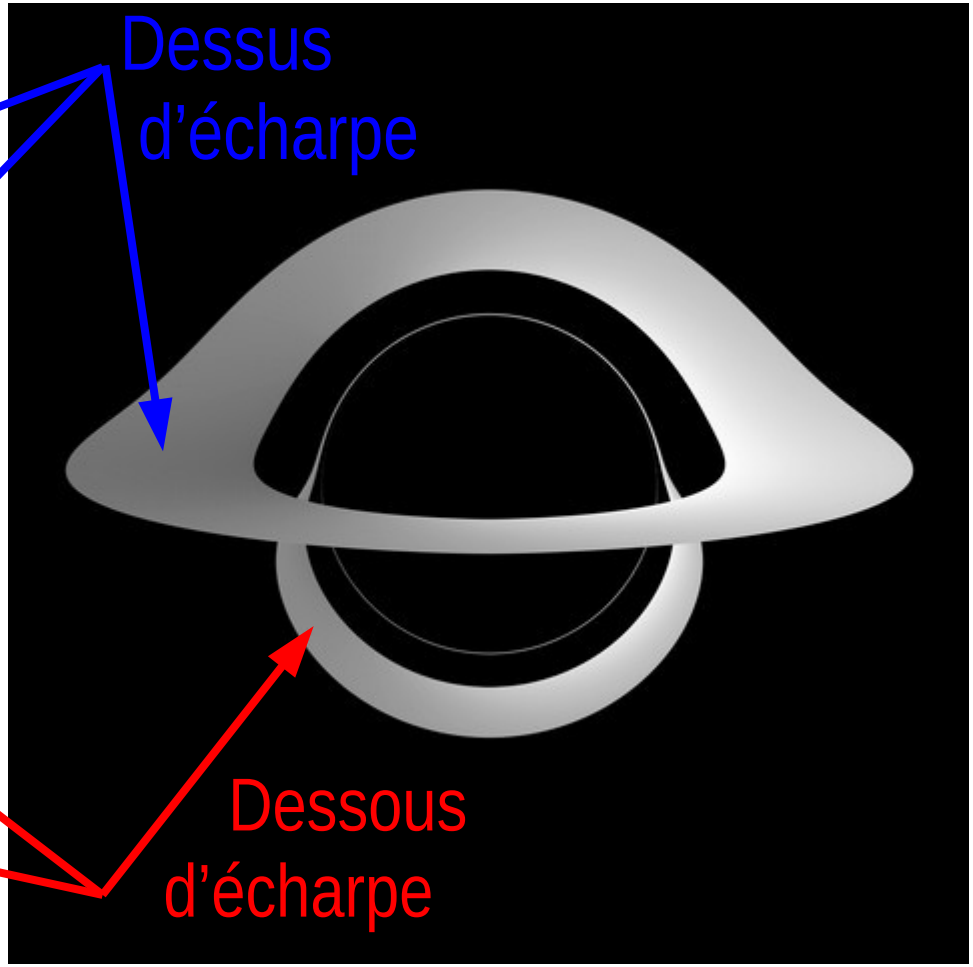
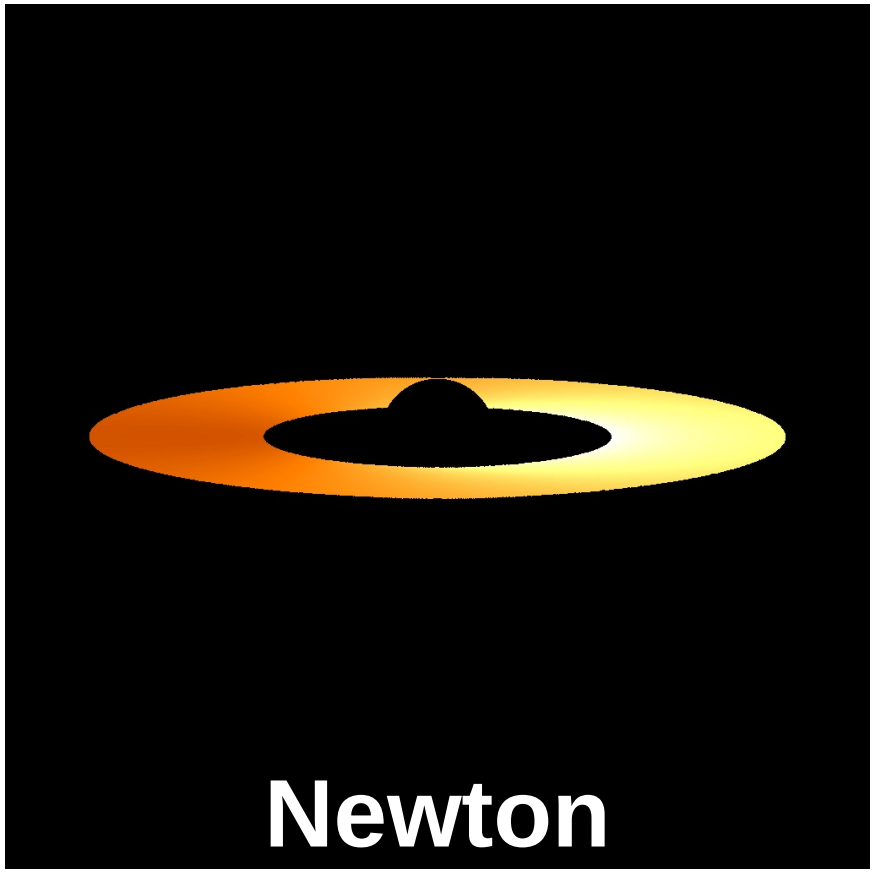
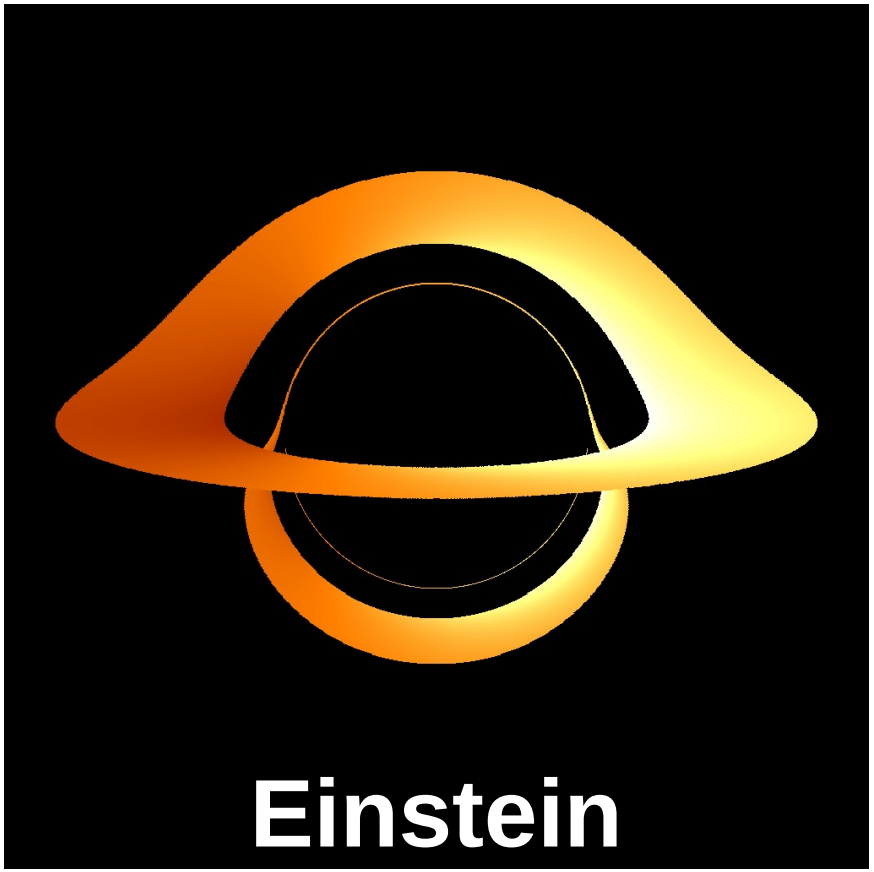


Fig. 4. Illustrative orbits in the plane $\{\phi=0\}$. Trajectory 1 has the critical impact parameter and circles infinitely around the black hole; trajectories 2 and 3 give direct images, trajectory 4 gives a secundary image

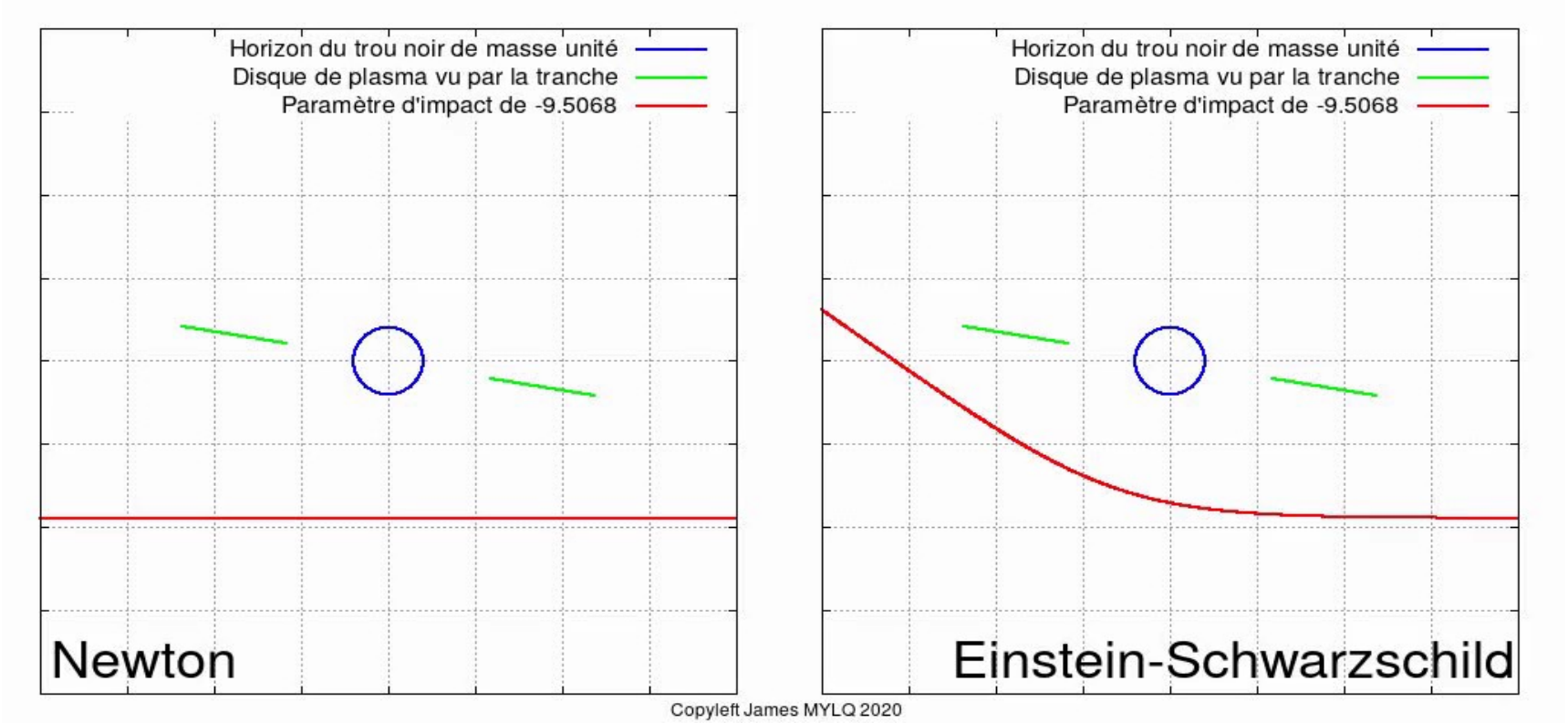


Calculer les trajectoires inverses Entre Newton et Einstein

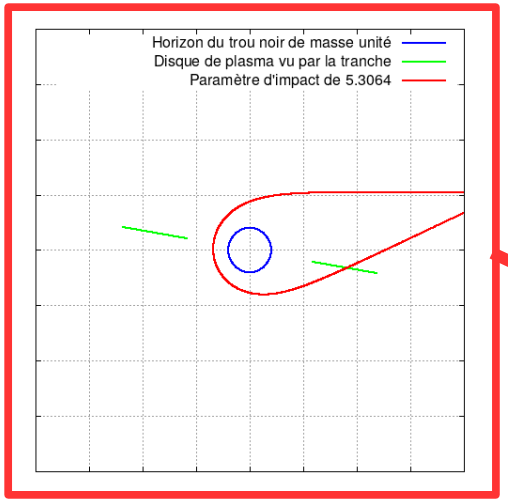


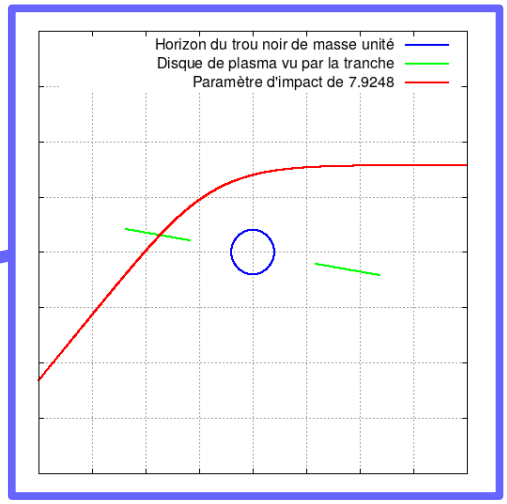


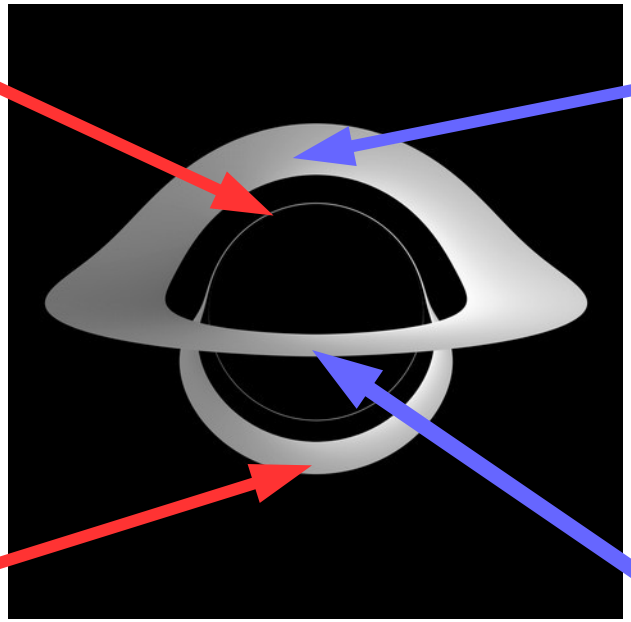
Et ça donne quoi pour tous les paramètres d'impacts ?

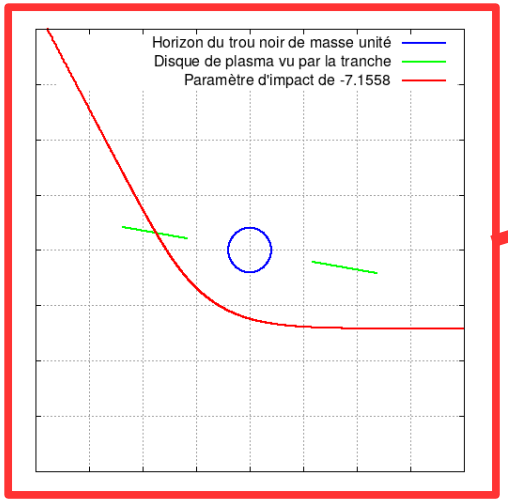


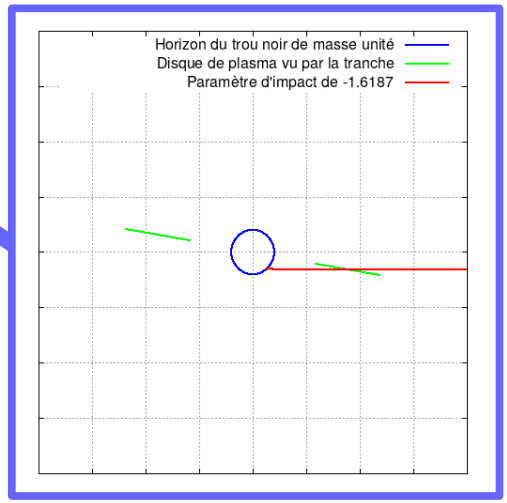
Quelques cas particuliers : le dessus, le dessous...



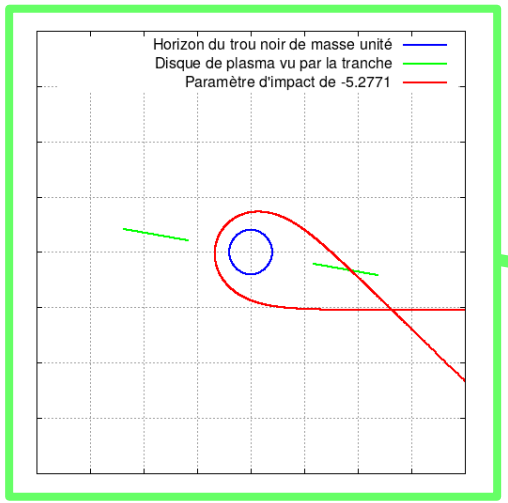


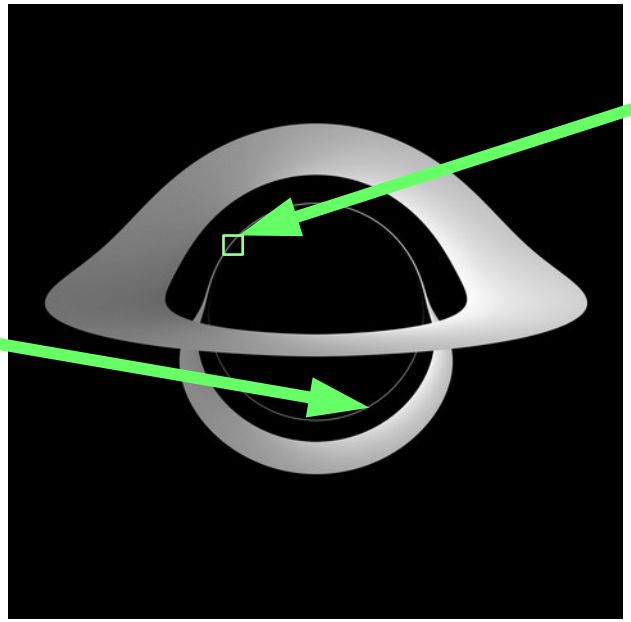


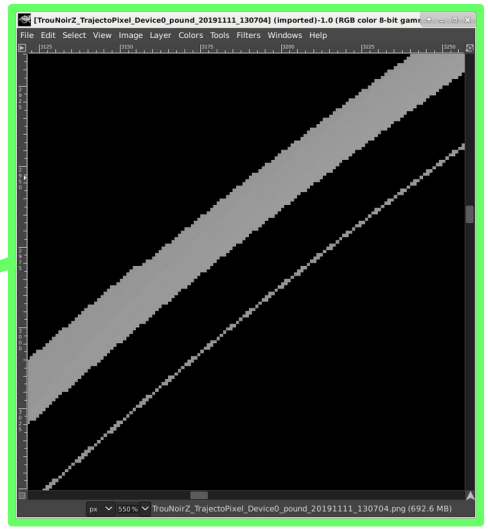


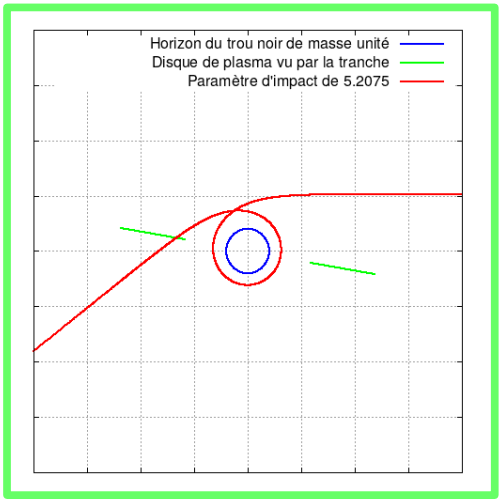


Quelques cas particuliers : le redessus, dessus et dessous





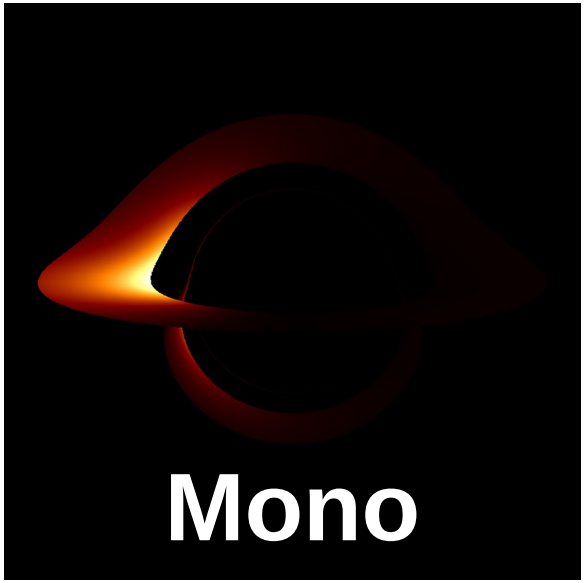


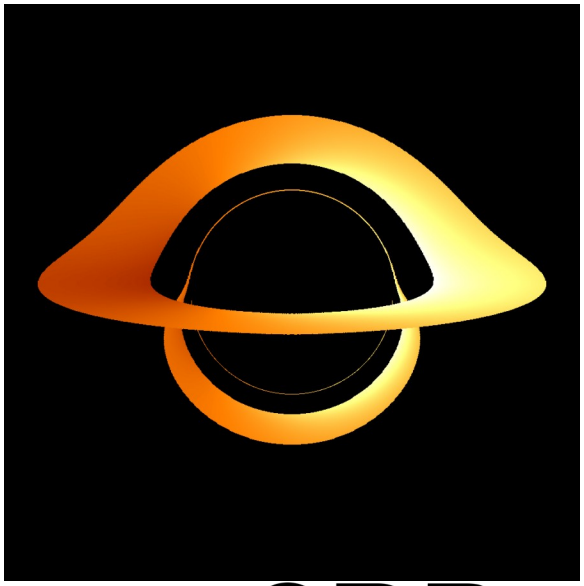


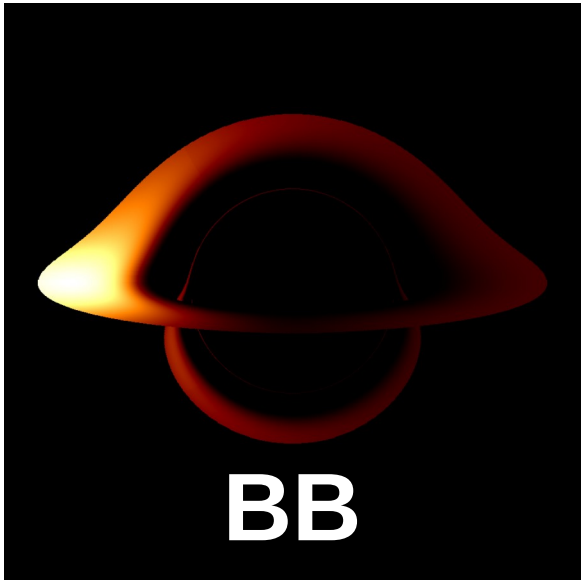
A chaque impact, deux physiques
Monochrome & Corps noir

« Puissance 4 » sur z

Spectre de planck



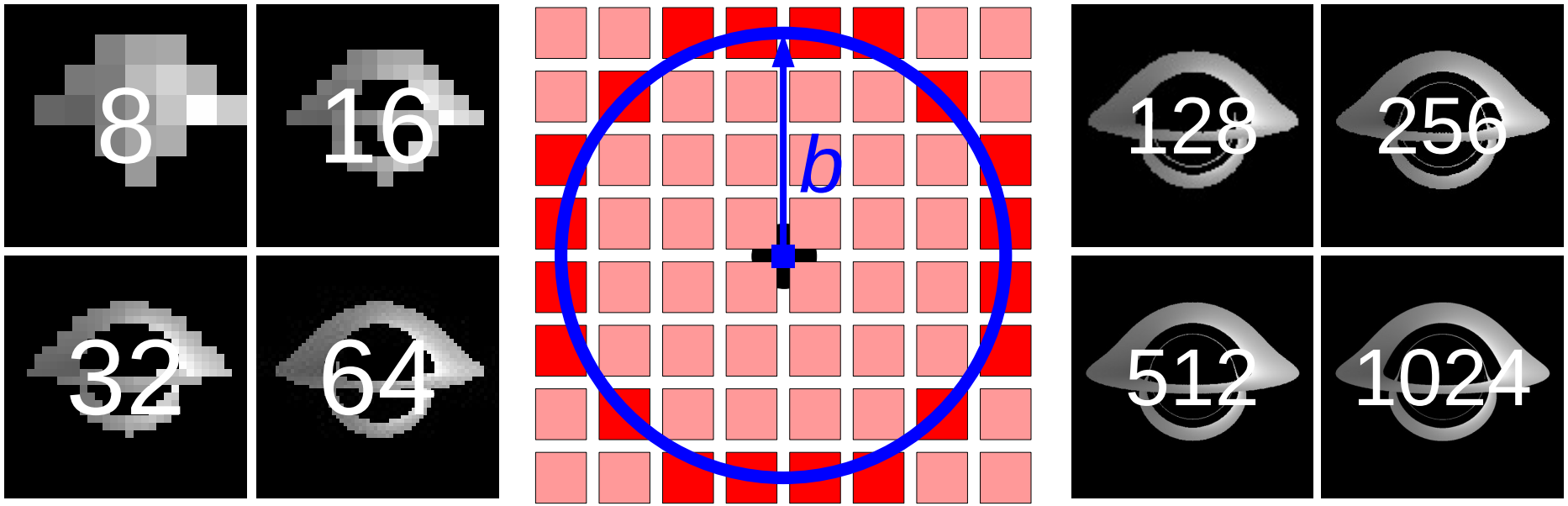




Méthode « économique » : exploitation symétrie cylindrique

- Pour chaque « paramètre d'impact » (distance au centre)
  - Calcul de la trajectoire du photon en fonction de l'angle
  - Pour chacun des pixels de l'image avec ce paramètre d'impact :
    - Estimation de l'indice d'interception correspondant à l'angle du disque
    - Test si la distance au centre pour cet indice est entre les rayons interne et externe
      - Estimation de l'effet Doppler & Einstein
      - Estimation du flux par deux méthodes :
        - Émission monochromatique : simple mais instructive
        - Émission de « corps noir » : plus réaliste mais spectre de Planck
- Beaucoup plus efficace et temps de calcul ~ #pixels
  - Exploitation du PixHertz (nombre de pixels sur temps écoulé)...

Du paramètre d'impact « *b* » Au cercle sur l'image

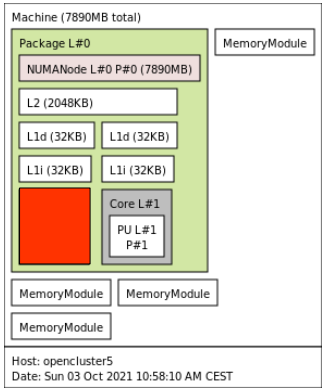


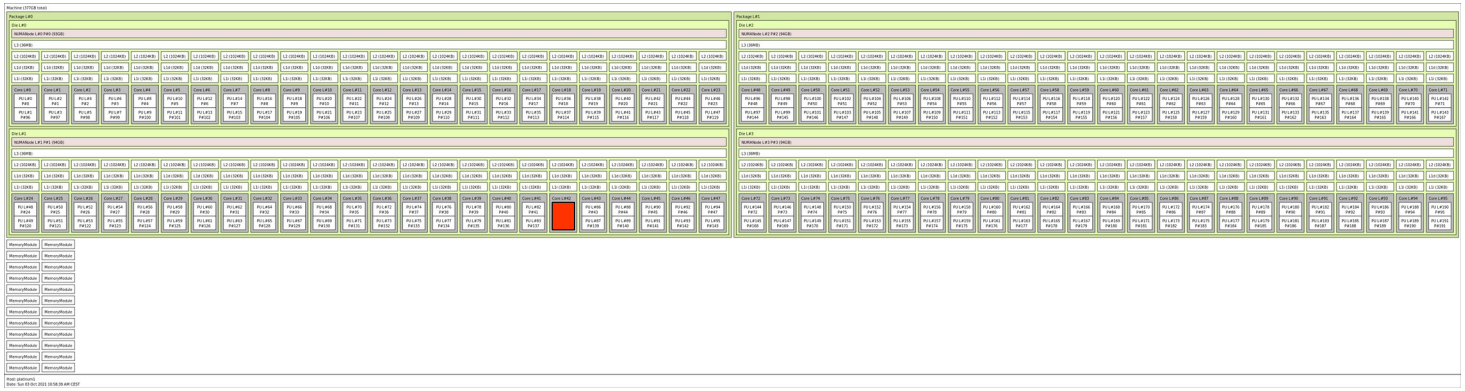
Balayage des pixels : découpage du cercle en *8b* secteurs

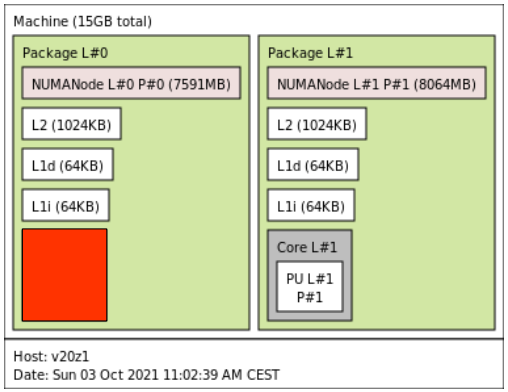
Le code, la boucle principale : paramètres d'impact & angles

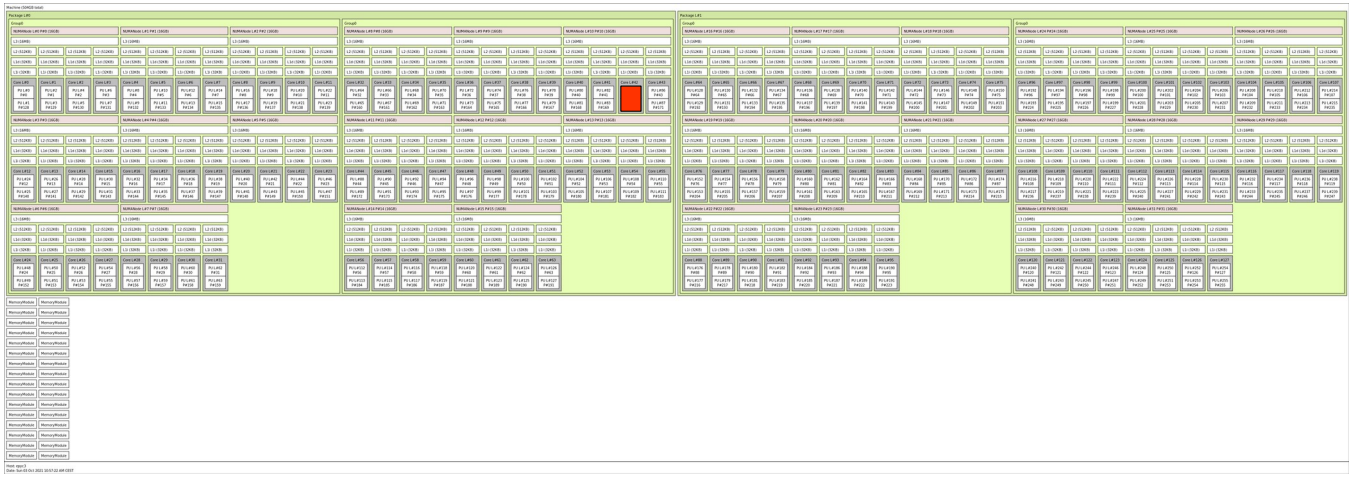
```
for (n=1;n<=nmx;n++)
 {
  h=4.*PI/(MYFLOAT)TRACKPOINTS;
  d=stp*n;
  db=bmx/(MYFLOAT)nmx;
  b=db*(MYFLOAT)n;
  up=0.;
  vp=1.;
  pp=0.;
  nh=1;
  rungekutta(&ps,&us,&vs,pp,up,vp,h,m,b);
  rp[(int)nh]=fabs(b/us);
  do
   {
    nh++;
    pp=ps;
    up=us;
    vp=vs;
    rungekutta(&ps,&us,&vs,pp,up,vp,h,m,b);
    rp[(int)nh]=b/us;
   } while ((rp[(int)nh]>=rs)&&(rp[(int)nh]<=rp[1]));
  for (i=nh+1;i<TRACKPOINTS;i++)
   {
    rp[i]=0.;
   }
  imx=(int)(8*d);
  for (i=0;i<=imx;i++)
   {
    phi=2.*PI/(MYFLOAT)imx*(MYFLOAT)i;
    phd=atanp(cos(phi)*sin(tho),cos(tho));
    phd=fmod(phd,PI);
    ii=0;
    tst=0;
    do
     {
      php=phd+(MYFLOAT)ii*PI;
      nr=php/h;
      ni=(int)nr;
      if ((MYFLOAT)ni<nh)
       {
        r=(rp[ni+1]-rp[ni])*(nr-ni*1.)+rp[ni];
       }
      else
       {
        r=rp[ni];
       }
      if ((r<=re)&&(r>=ri))
       {
        tst=1;
        impact(d,phi,dim,r,b,tho,m,zp,fp,q,db,h,bss,raie);
       }
      ii++;
     } while ((ii<=2)&&(tst==0));
   }
}
```

Plateau multi-cœurs : 280 hôtes de 2 à 128 cœurs : les extrêmes









Plateau multi-shaders : (GP)GPU
108 modèles différents…
dont 72 accessibles **directement** !

GPU Gamer :

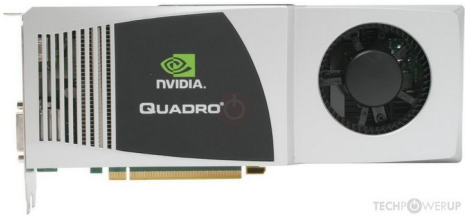
33 modèles

De la GT 640
… à la RTX 4090

GPGPU :

15 modèles

Nvidia Tesla C1060
… à la Nvidia A100



GPU desktop & pro :

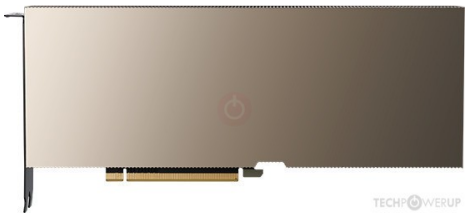
34 modèles

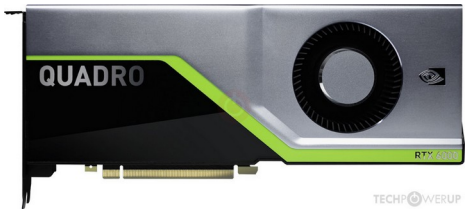
De la Quadro FX 4800
… à la RTX A6000

GPU AMD Gamer :

26 modèles

De la HD 7970
… à la RX 6900 XT

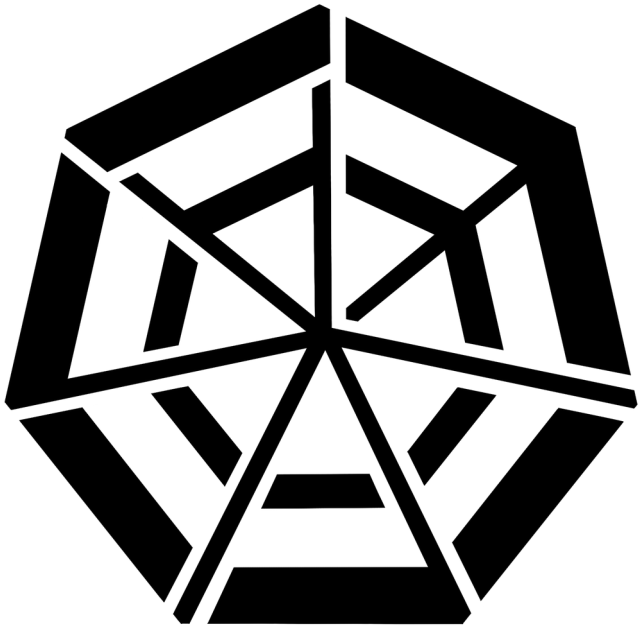




Sur les Machines du CBP : SIDUS

Je n'installe pas, je démarre !

- **Quoi ?**
  - Déployer un système simplement sur un parc de machines
- **Pourquoi ?**
  - Assurer l'unicité des configurations
  - Limiter l'empreinte du système sur les disques
- **Pour qui ?**
  - Étudiants (vous quoi!), enseignants, chercheurs, ingénieurs, ...
- **Quand & Où ?**
  - Centre Blaise Pascal : depuis 2010, plus de 280 machines
  - PSMN : depuis 2011, plus de ~800 nœuds (sa propre instance)
- **Comment ?**
  - Utiliser un partage en réseau d'une arborescence
  - Détourner une ruse de LiveCD

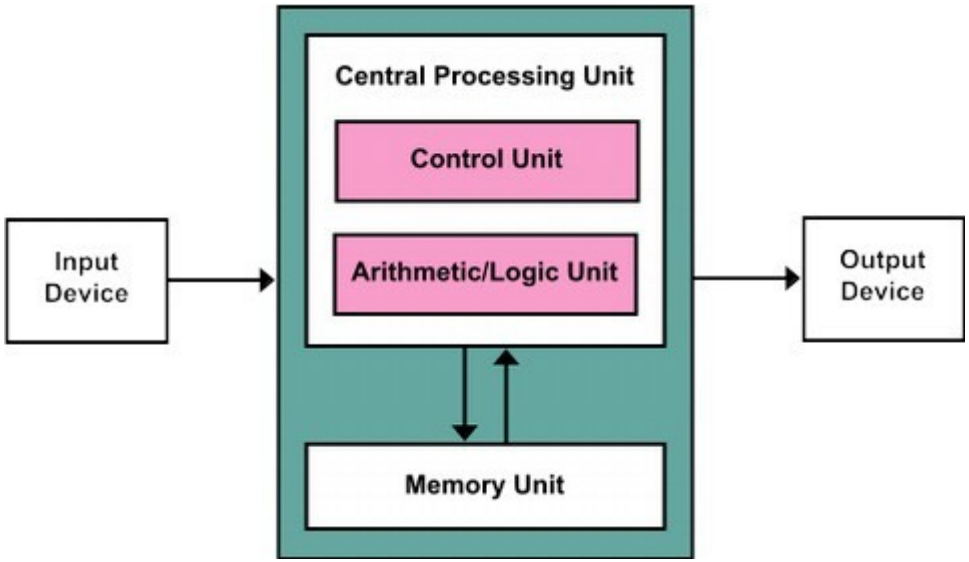


« Deux machines ayant démarré SIDUS ne peuvent pas ne pas avoir le même système ! »

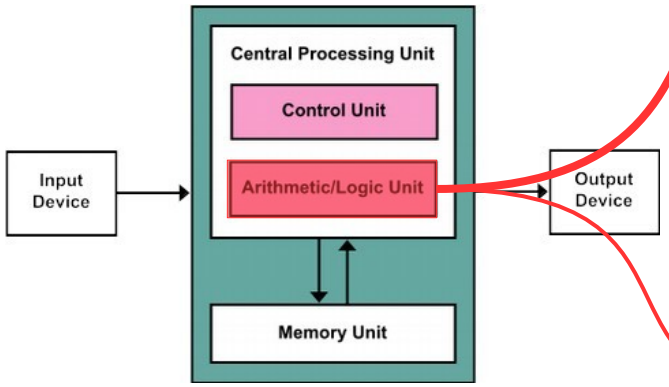
Computhèque
40 ans d'histoire de l'informatique

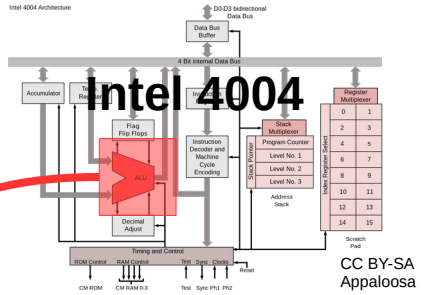
- Ordinateurs :
  - Thomson TO8, Amiga 500, MacSE, …
- Processeurs (et leurs cartes-mère) :
  - 80386SX, 80486SX, Overdrive DX4, Amd5x86, K6-2, K7, …
- Périphériques de stockage :
  - Cartes, câbles, disques, lecteurs de bande, … en SCSI, Firewire, FC
- Périphériques de communication :
  - Ethernet 10Base2, 10Base-T, ATM, Myrinet, Infiniband, …
- OS : natifs ou Debian Buzz, Hamm, Squeeze, ...

Un ordinateur : un métier à tisser ?
Architecture de Von Neumann

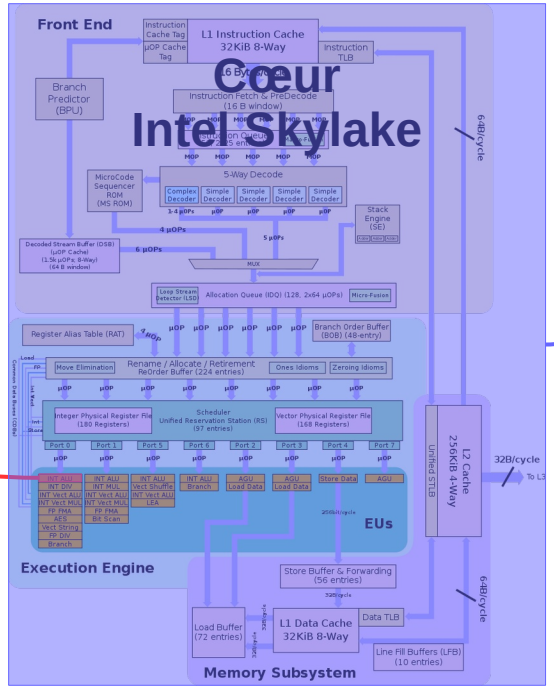


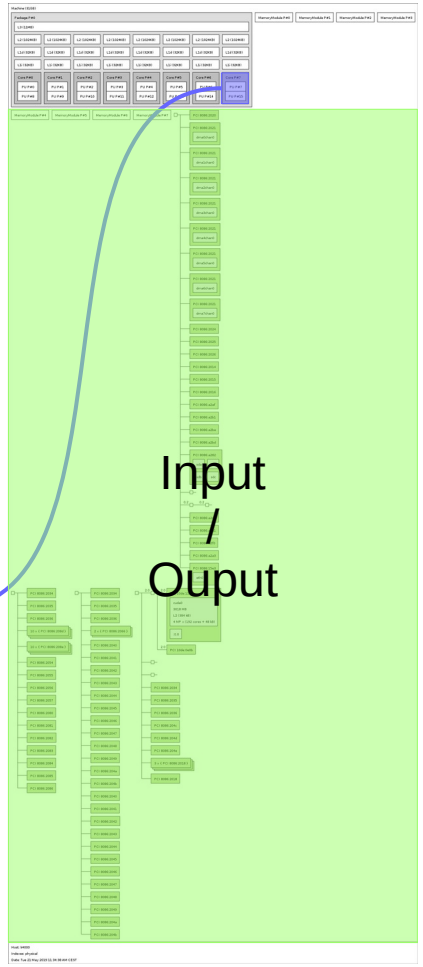
Architecture de Von Newman 50 ans d'évolution chez Intel





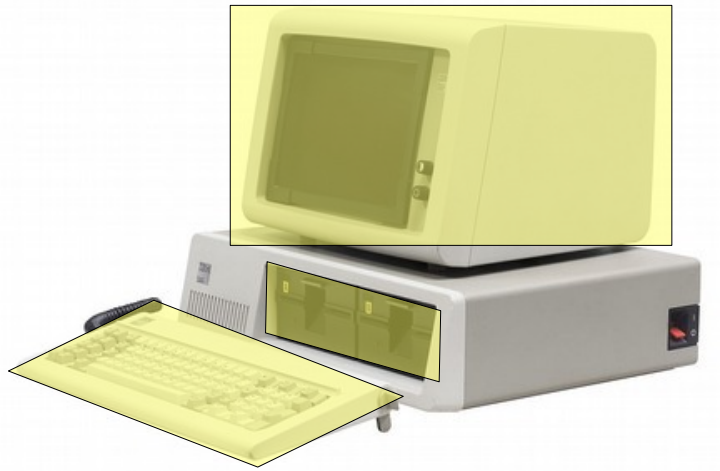
CC BY-SA
Appaloosa

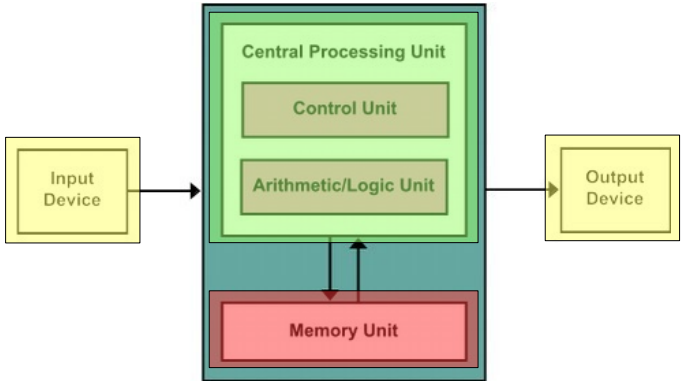


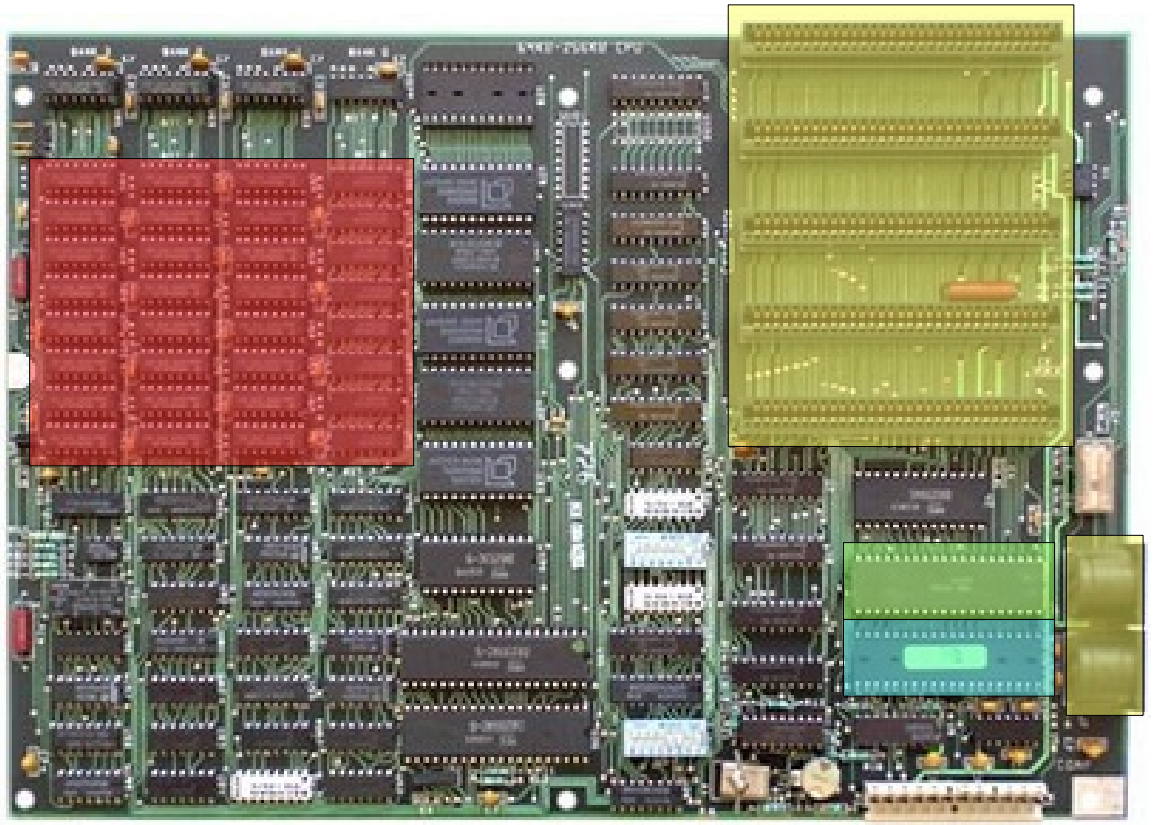


Input
/
Ouput

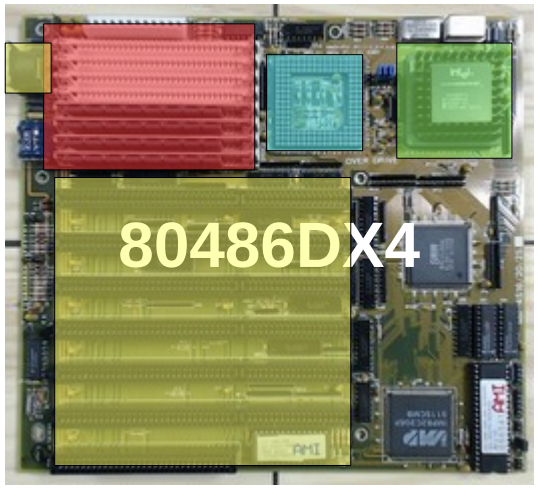
Une révolution informatique
IBM Personal Computer, le « PC »

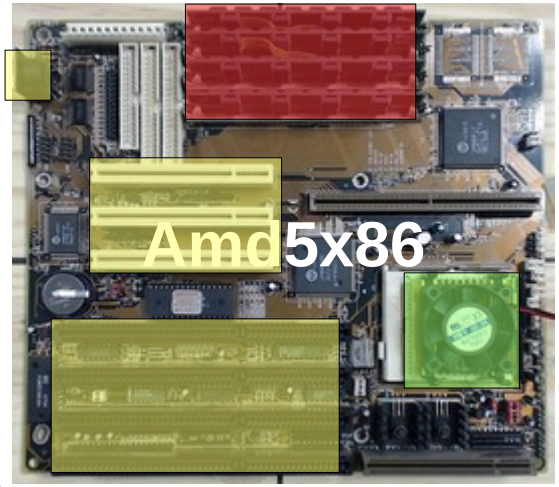


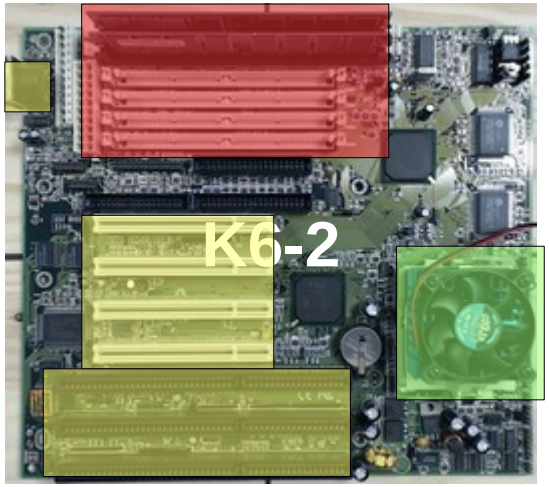


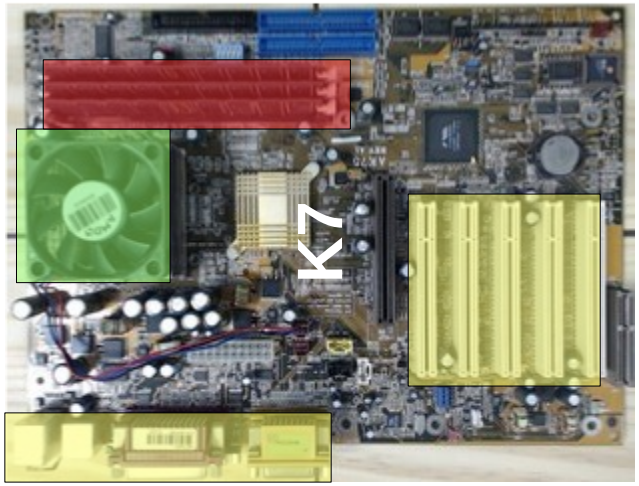


Evolution des cartes mères en un clin d'œil, de 1989 à 2002









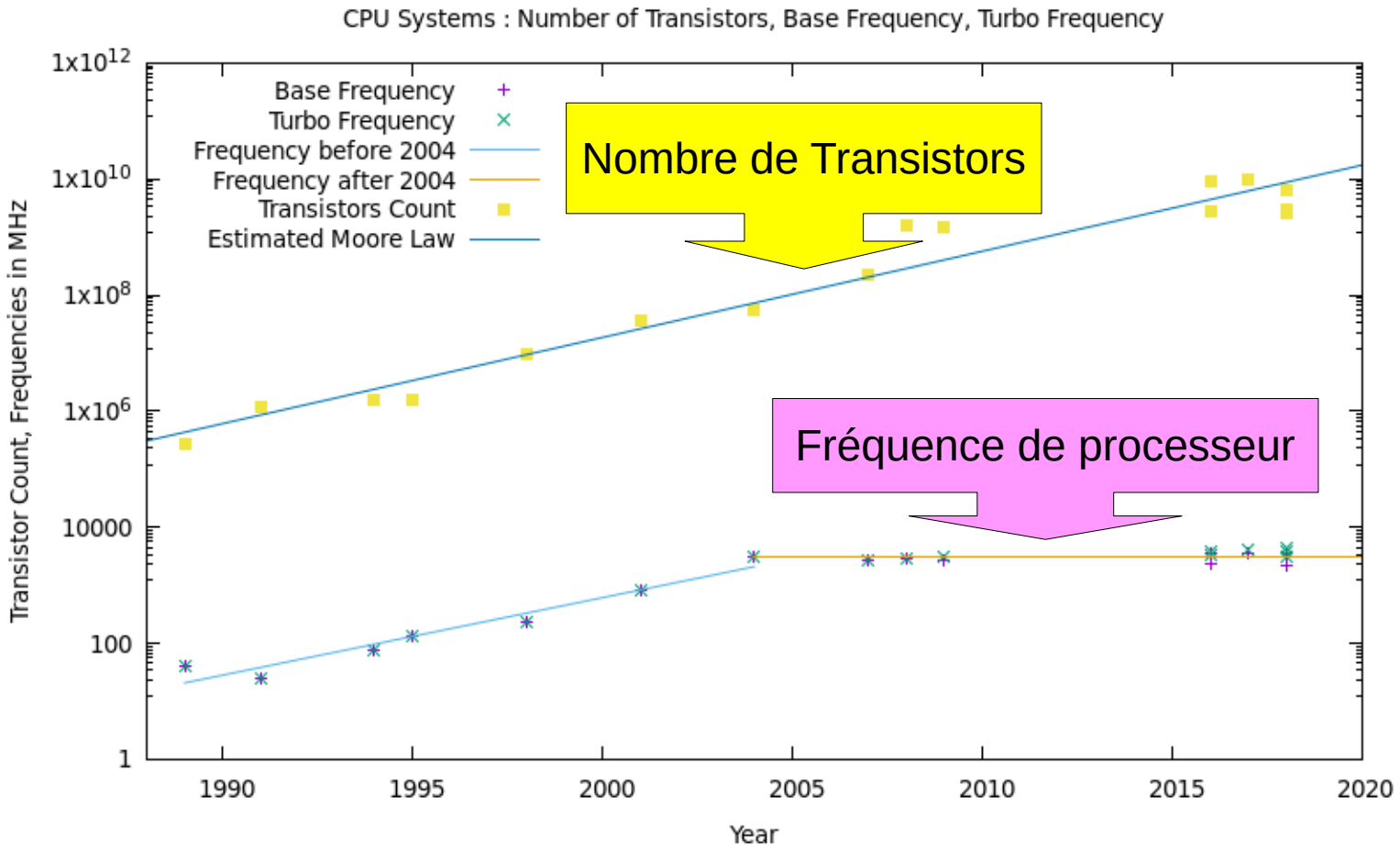
Evolution des cartes mères en un clin d'œil, de 2005 à 2018

Banc de test
Entre le code et le matériel, l'OS

- Une même distribution de 1996 à 2019 : Debian (presque)
  - Debian Buzz : 80386SX
  - Debian Hamm : 80486SX, 80486DX4, Amd5x86, K6-2
  - Debian Stretch 32 bits : K7, Northwood
  - Debian Stretch 64 bits : les 8 autres
  - Ubuntu 18.10 64 bits : TR 1950X

- Composants nécessaires :
  - Programme séquentiel : compilateur C
  - Programme OpenMP : compilateur C et librairie OpenMP
  - Programme OpenCL : interpréteur Python & PyOpenCL, pilotes Nvidia & ROCM
  - Programme CUDA : interpréteur Python & PyCUDA, pilotes Nvidia

Distribution de CPU pertinente ? Transistors & Fréquences...

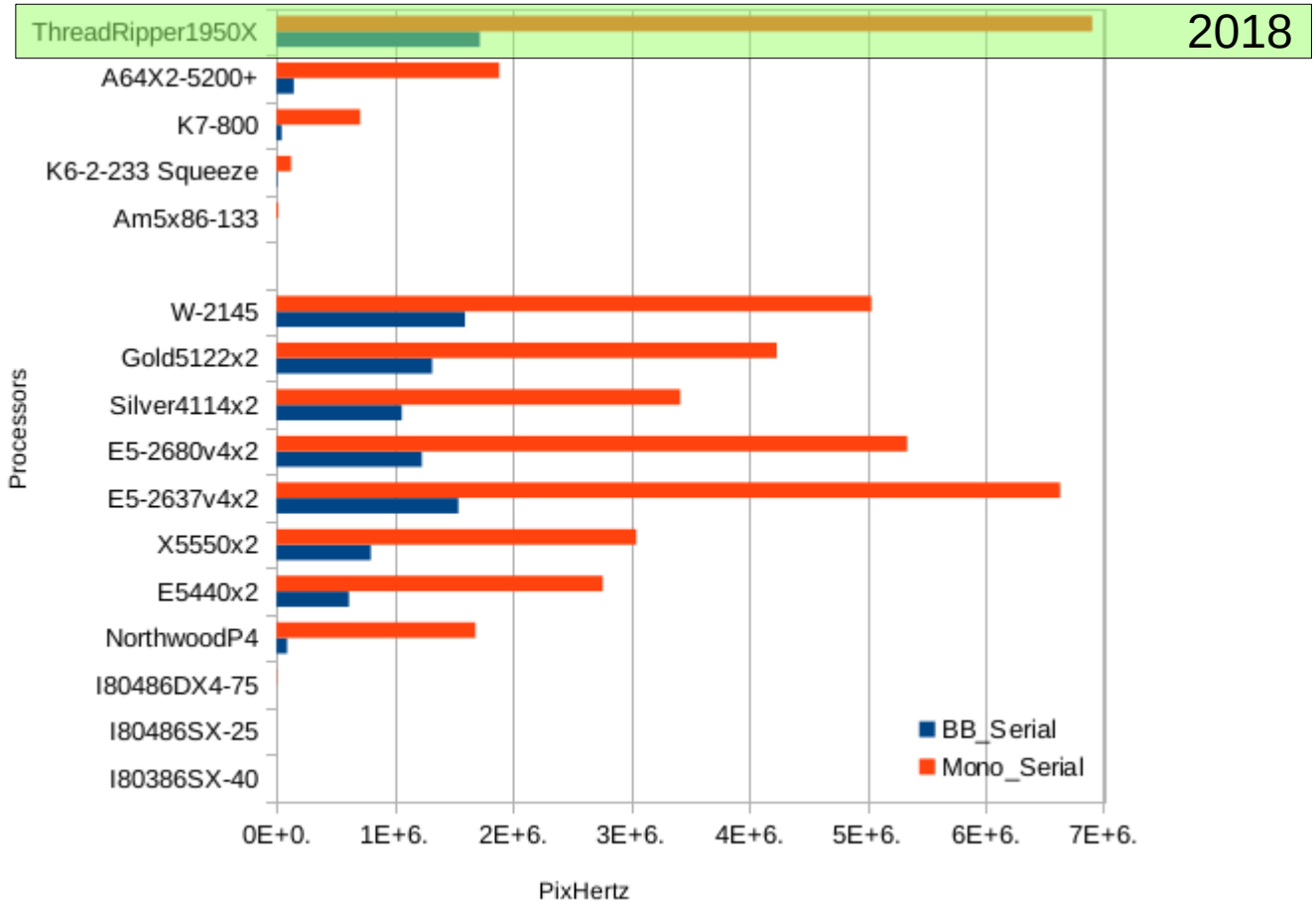


Doublement des transistors tous les 2 ans...

Banc de test sur les 16 CPUs

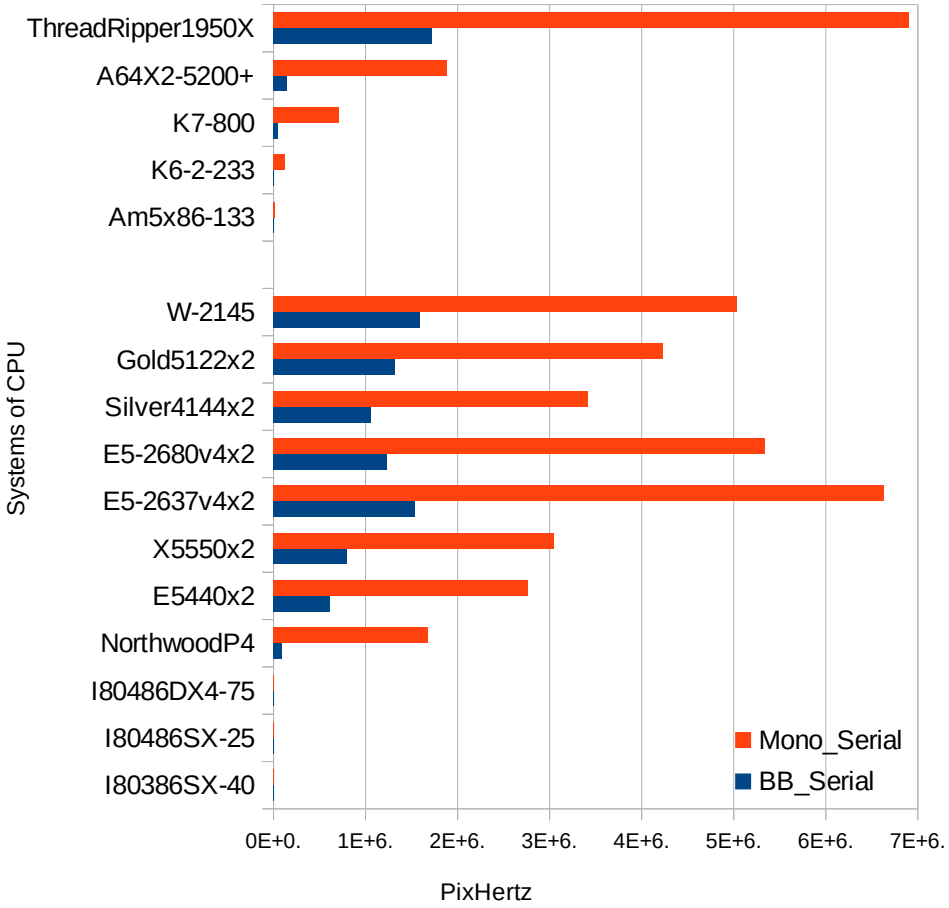
- Les processeurs et leurs distributions :
  - 80386SX, 80486SX, Overdrive DX4, Amd5x86 : Debian Buzz & Hamm
  - K7, Northwood, AthlonX2 : Debian Stretch
  - E5440x2, X5550x2, E5-2637v4x2, E5-2680v4, Gold5122, Silver4144, W-2145 : Debian Stretch
  - Threadripper 1950X : Ubuntu 18.10
- Images de 64x64 à 16384x16384 pixels : 2^6 à 2^{14}
  - Sauf pour les très très vieux CPU : limitation à 256x256
- Méthodes : 2 à explorer avec « charges » différentes
  - Charge calculatoire faible : « Monochromatique » (ak Mono)
  - Charge calculatoire élevée : « Corps Noir » (aka BB)
- Statistiques : 10 lancements successifs
  - Exploitation de la médiane pour le « Elapsed Time »

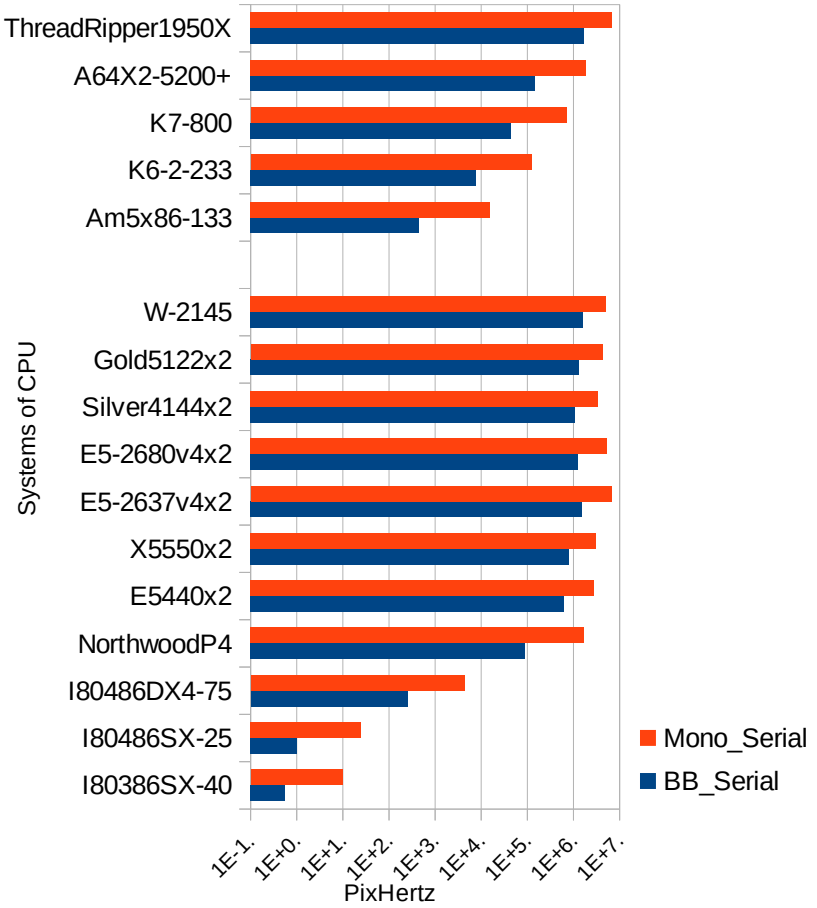
16 processeurs de 1989 à 2019 : *« and the winner is »*...



Le AMD Threadripper 1950X à 3.6 GHz

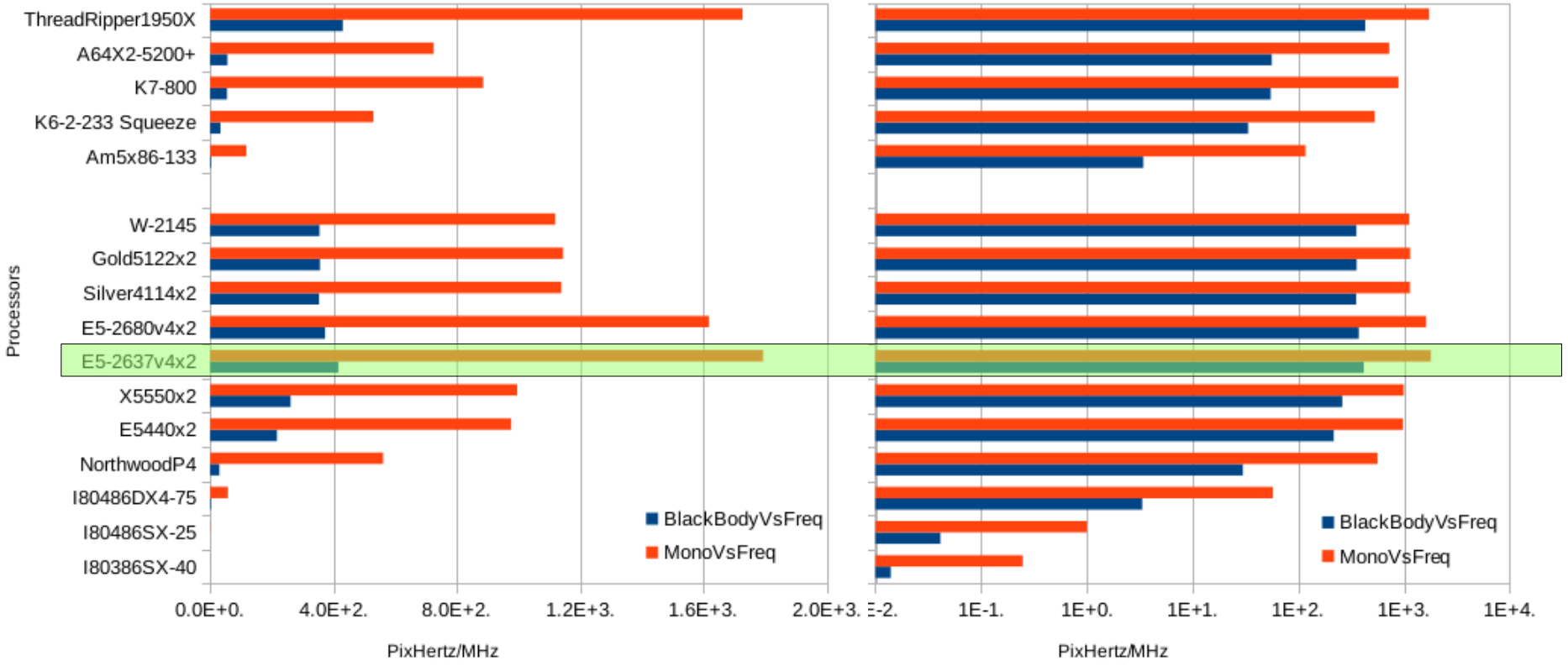
Exécution du code séquentiel
On gagne (quand même) en 30 ans





Gain Best/Worse : 3 millions en BB et 700000 en Mono...

Influence de la fréquence : de 1989 à 2019 : de 40 à 3700 MHz

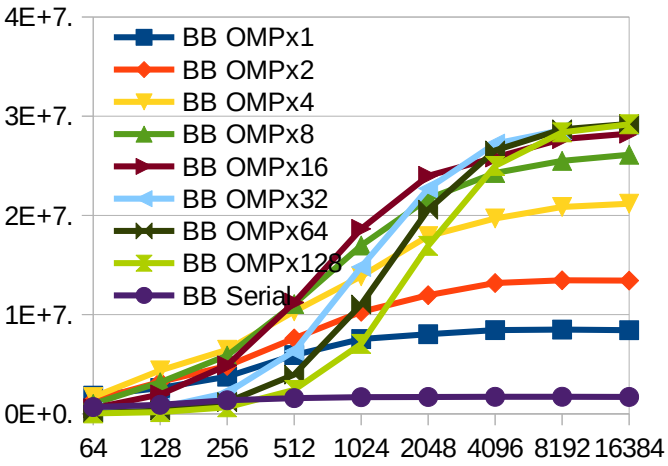


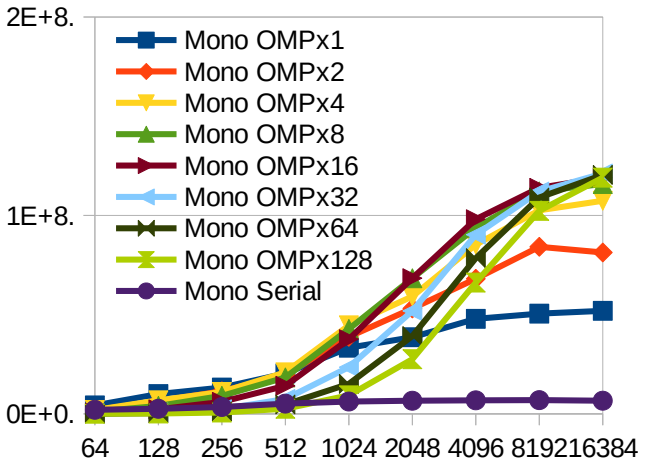
Sur 15 ans, entre un facteur 3 et un facteur 15…

Il va falloir trouver autre chose pour accélerer !

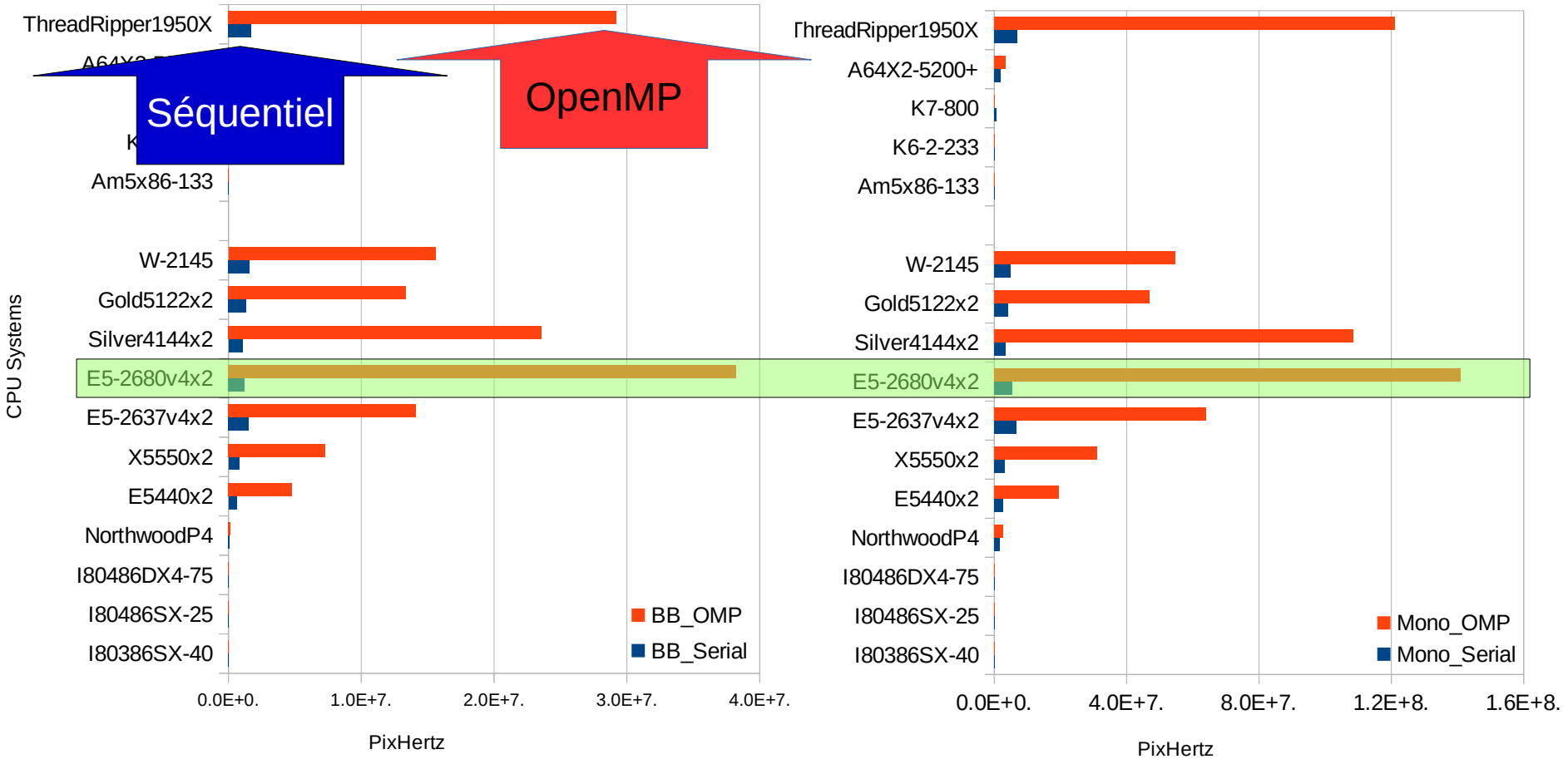
Parallélisation du code
Passage en OpenMP & étrangetés

- Parallélisation « naturelle » : paramètre d'impact
  - Modification mineure du code (déplacement de déclaration de variables)
  - Crainte : charge calculatoire non équivalente pour les différents tâches
  - Exploration pour différents OMP_NUM_THREADS : de 1x à 128x
- Pour le meilleur : le ThreadRipper 1950x, un **x17-x18**





Parallélisation OpenMP
On gagne plus que prévu en BB !



x68 millions en BB et x14 millions en Mono

Peut-on mieux faire ? Testons OpenCL !

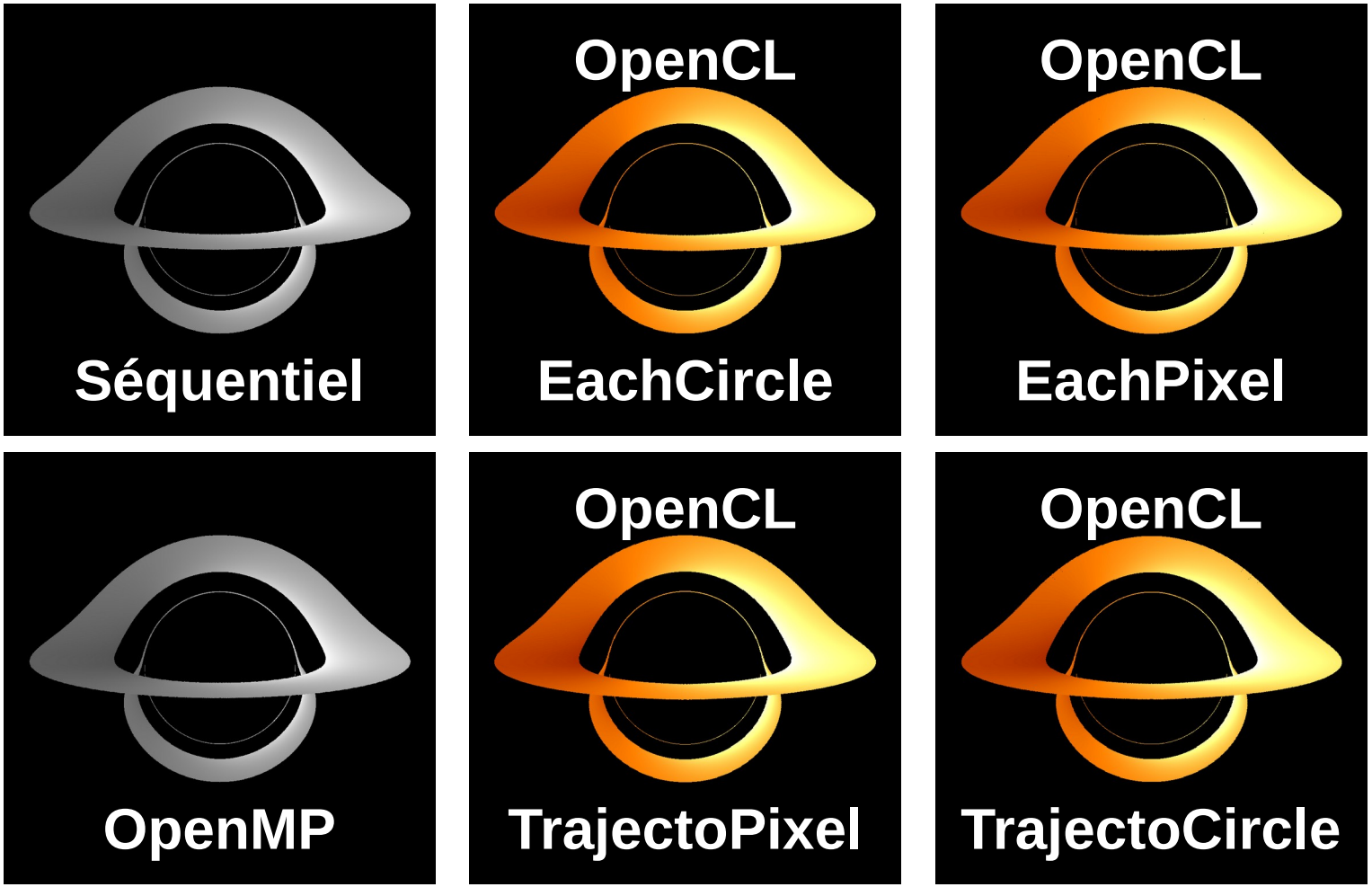
- OpenCL, méconnu mais tellement polyvalent : 13 implémentations
  - GPU : Nvidia, AMD via ROCm, AMD via Mesa, Intel via Beignet et Intel
  - CPU : AMD, PortableCL, **Intel**
  - MIC : Intel pour Xeon Phi
  - FPGA : Altera/Intel, Xilinx
  - (DSP : Texas Instruments)
  - (GPU ARM)

- OpenCL : sa programmation...
  - Principe : des « noyaux » de calcul à distribuer à outrance !
  - Programmation « hardcore » en C, plus facile en C++
  - Programmation via API Python : « la voie » !

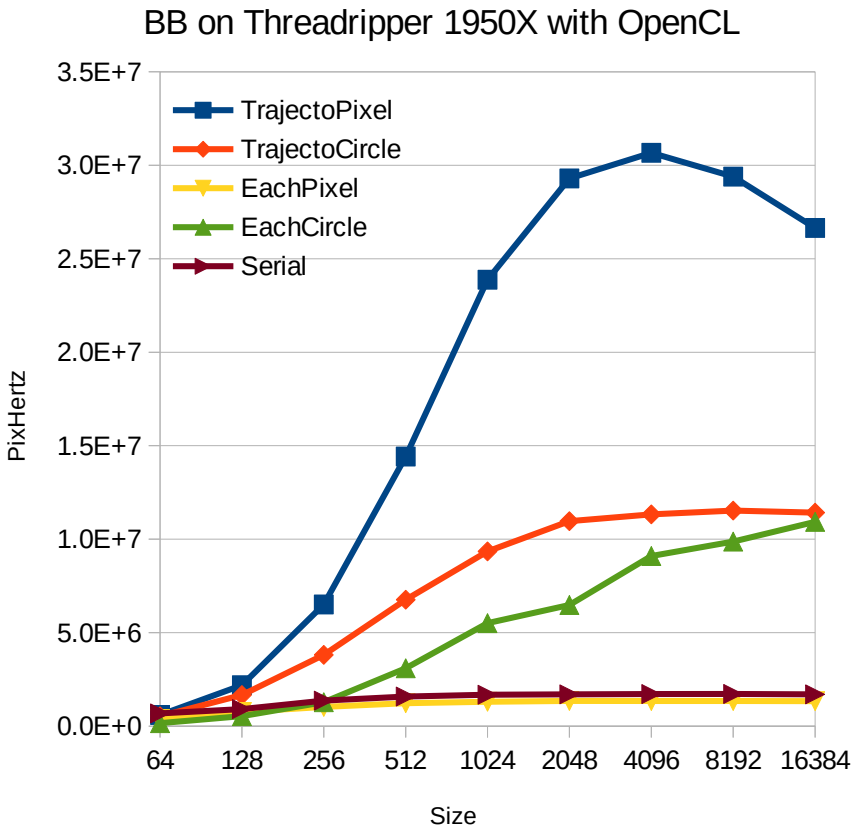
OpenCL : distribuer notre calcul. Quel régime de parallélisme PR ?

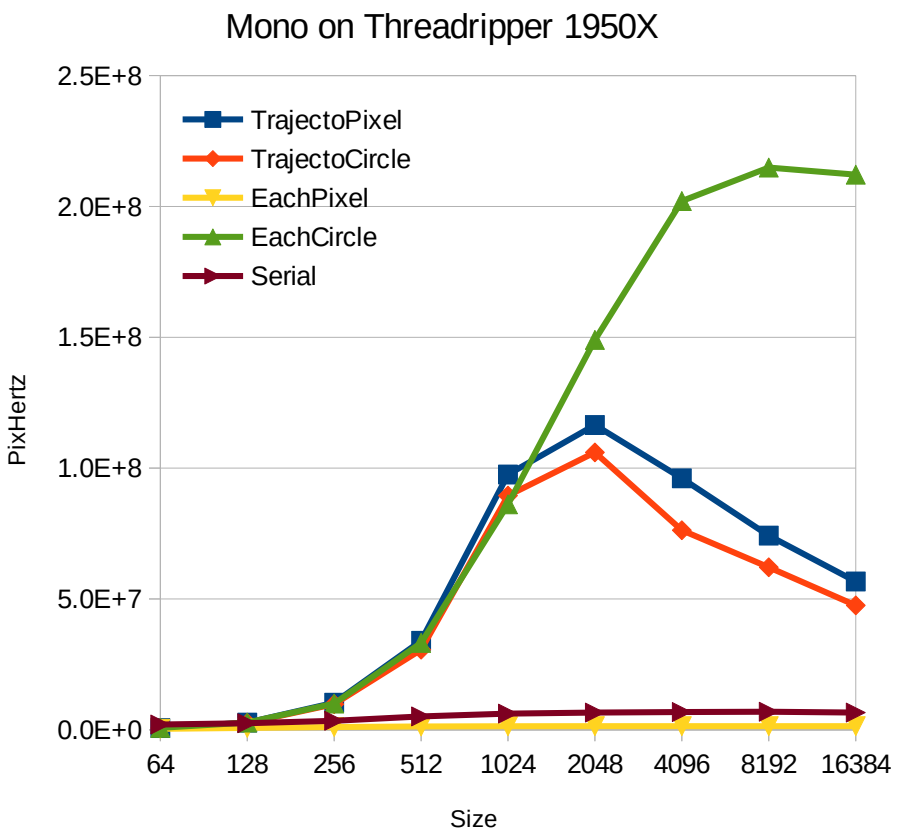
- Approche initiale héritée du code C : **EachCircle**
  - Parallélisé suivant le paramètre d'impact : PR=Taille/2
- Approche brutale : **EachPixel**
  - Parallélisé suivant le nombre de pixels : PR=Taille*Taille
- Approche hybride : **TrajectoPixel**
  - D'abord parallélisé suivant les paramètres d'impact : PR=Taille/2
  - Ensuite parallélisé suivant chaque pixel : PR=Taille*Taille
- Approche hybride sauvage : **TrajectoCircle**
  - D'abord parallélisé suivant les paramètres d'impact : PR=Taille/2
  - Ensuite parallélisé suivant chaque pixel : PR=4*Taille
- Donc 4 méthodes à explorer pour tous nos CPU !

Obtient-on les mêmes résultats ? Laissons l'œil juger...



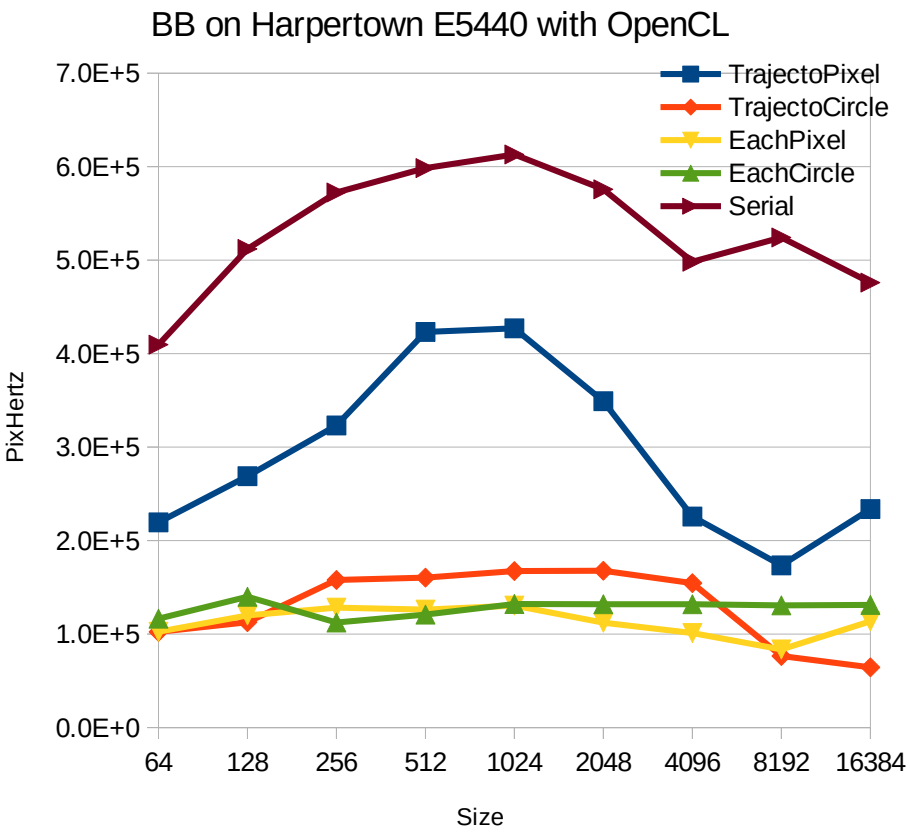
OpenCL sur Threadripper 1950x
La méthode de // importante...

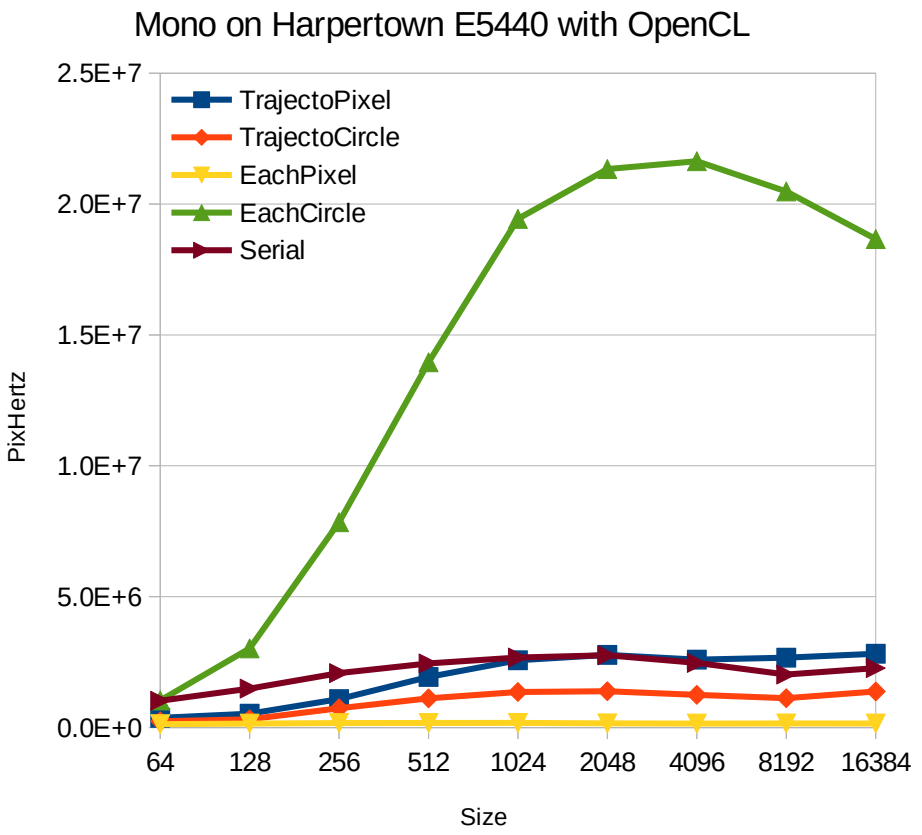




TrajectoPixel pour BB, EachCircle pour Mono…

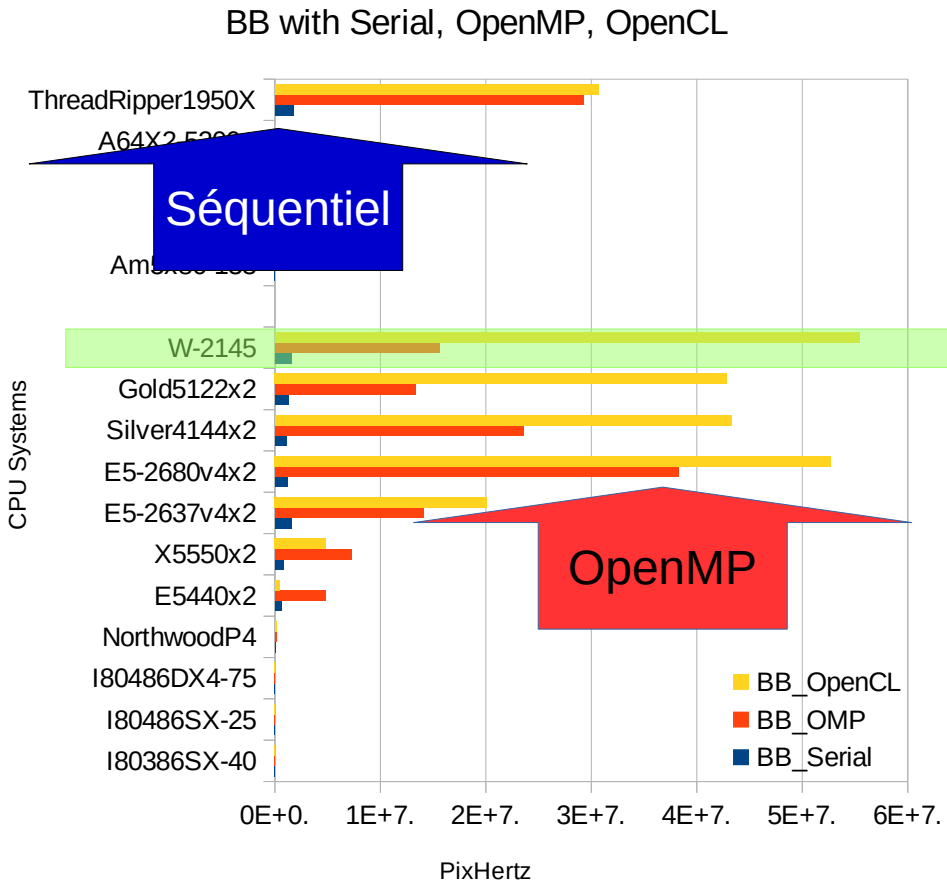
OpenCL sur Harpertown E5440x2
OpenCL (AMD) pas terrible

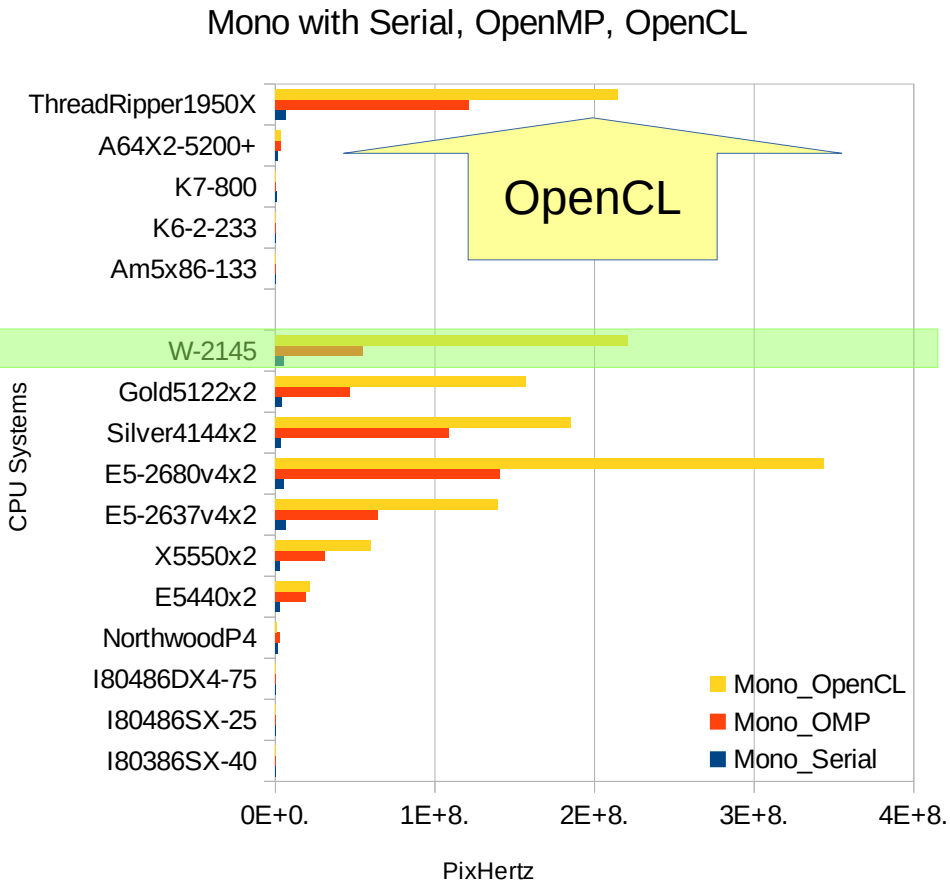




A chaque processeur, sa courbe de scalabilité...

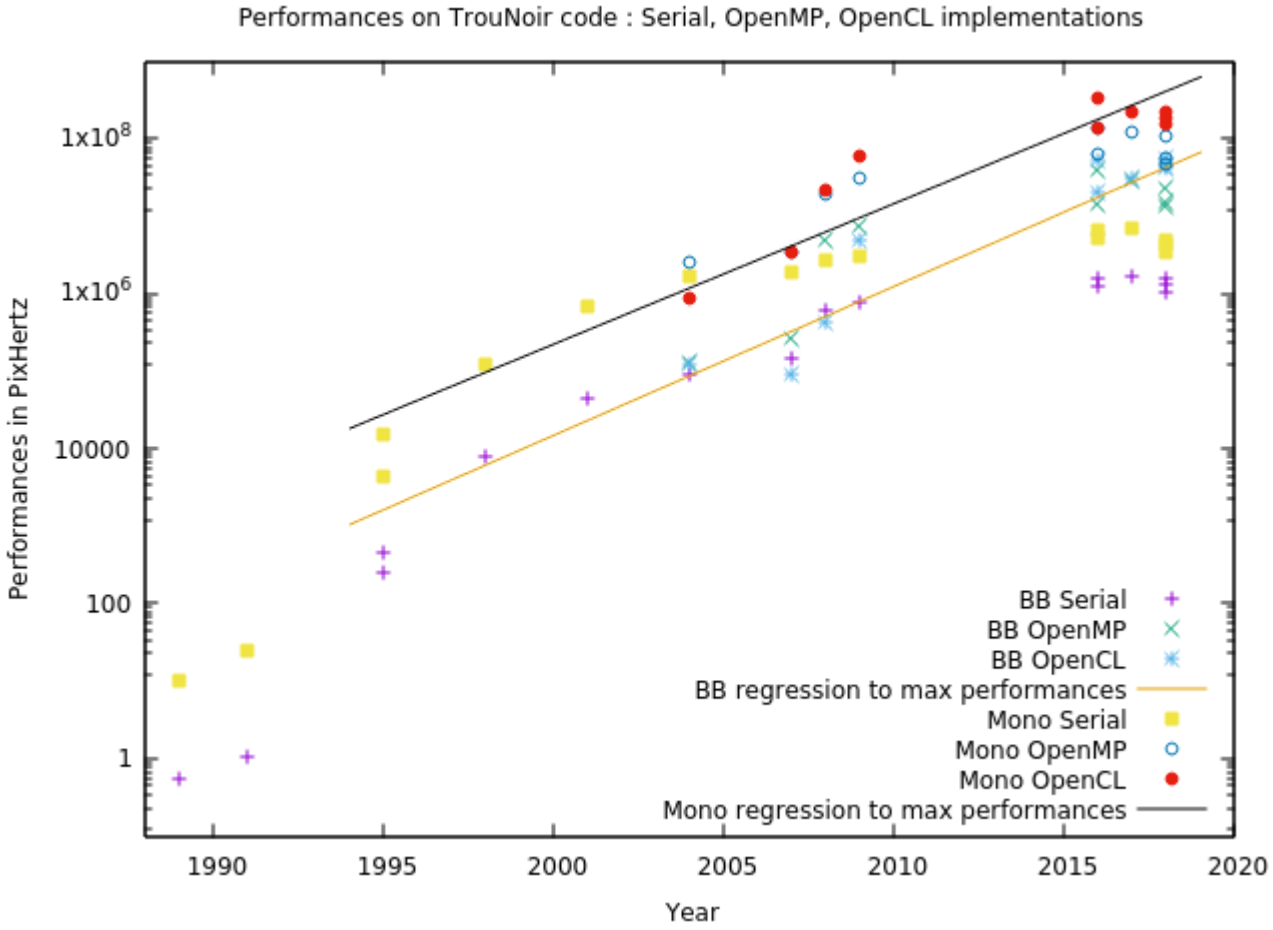
Parallélisation OpenCL
Pour tous les processeurs...





OpenCL Intel très efficace avec CPU Intel !

Pour les processeurs
Loi de Moore* respectée ! Enfin...



Performance : x2 tous les 18 mois !

Ajoutons les GPUs & MIC

Gamer

GPGPU

MIC

AMD GPU

Générations Fermi, Kepler, Maxwell, Pascal, Turing

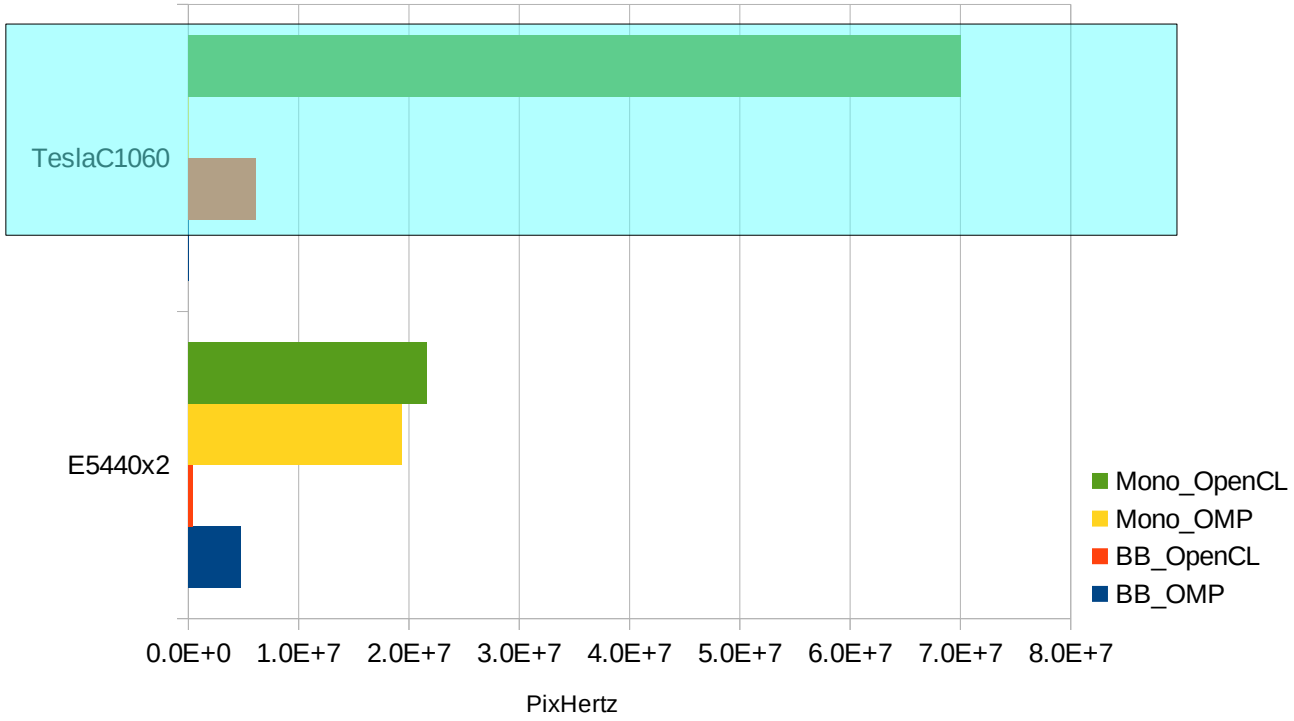
Générations GT200, Fermi, Kepler, Pascal, Volta

Intel : KNC

R9-Nano, Vega 64, Radeon 7

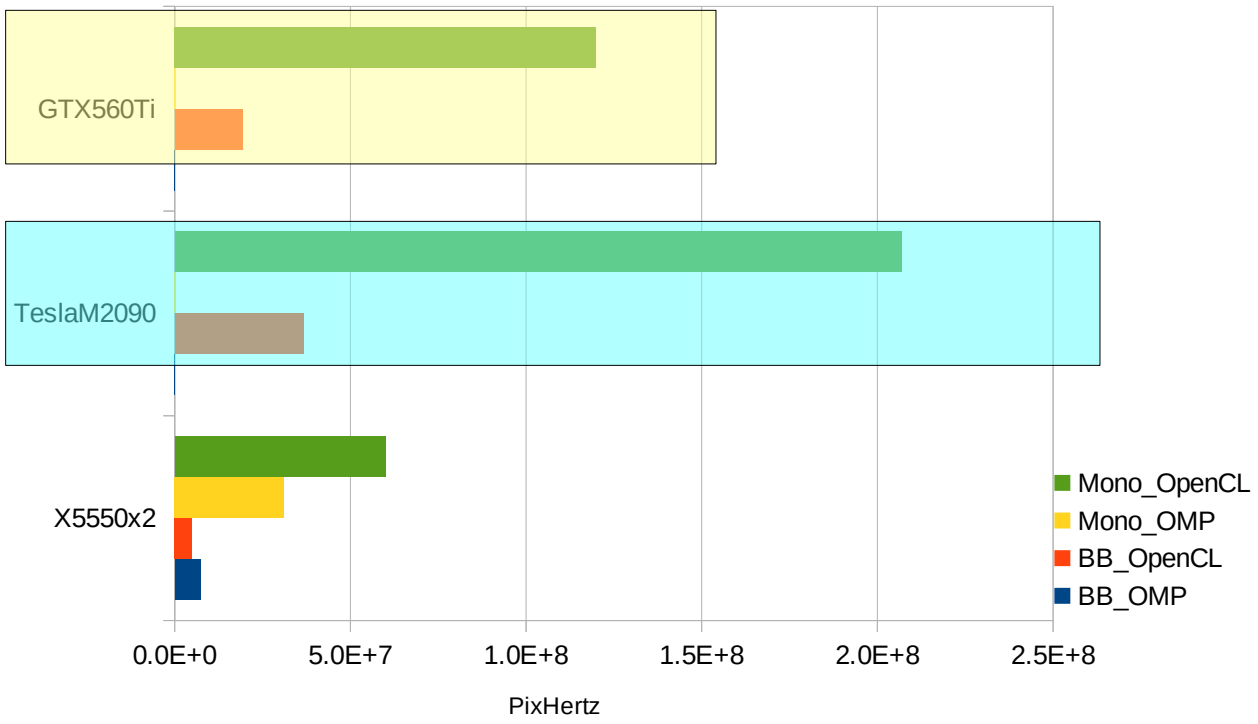
Le meilleur en computhèque de chaque génération...

Comparaison au fil du temps
En 2008, E5440 & C1060



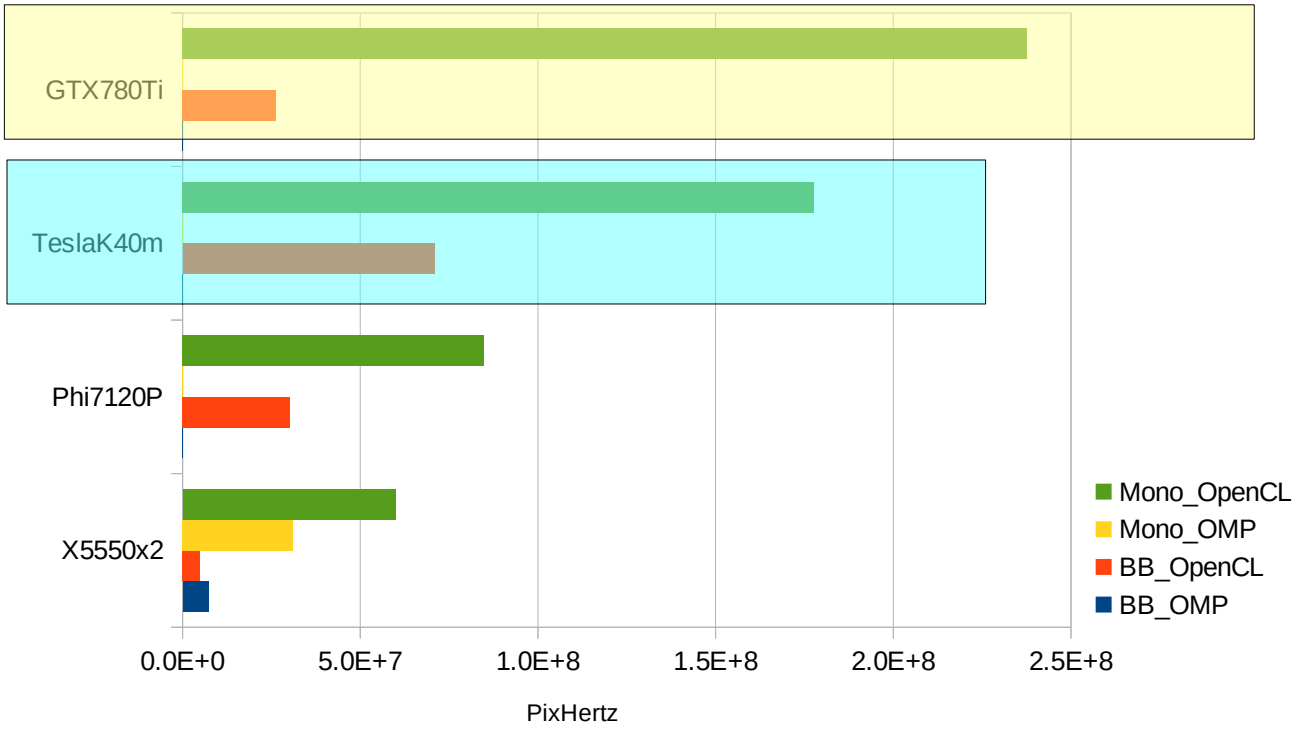
- En Mono, Un x3 en OpenCL pour le GPGPU
- En BB, à peine supérieur en OpenCL face à l'OMP

En 2011, entre Nehalem & Fermi Les différences pas si flagrantes...



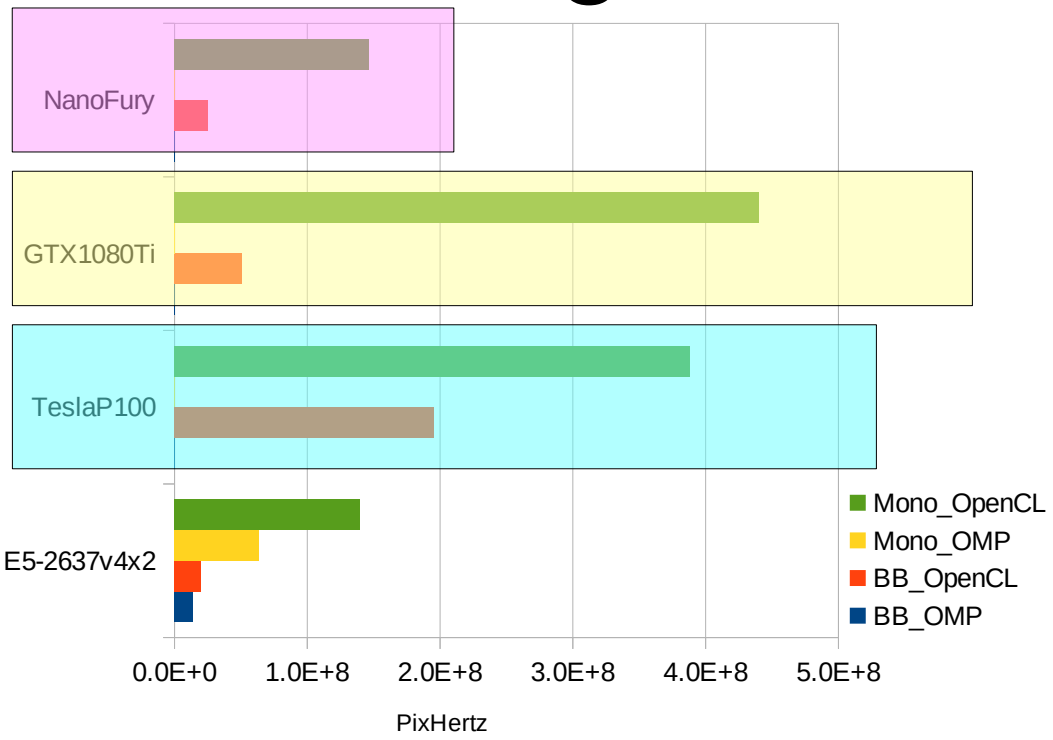
- Le GPGPU M2090 gagne d'un facteur 3 en Mono, mais d'un facteur 5 en BB
- La carte de Gamer l'emporte sur le double CPU d'au moins un facteur 2

En 2013, arrivée de Kepler et Phi Distinction en GPU & GPGPU



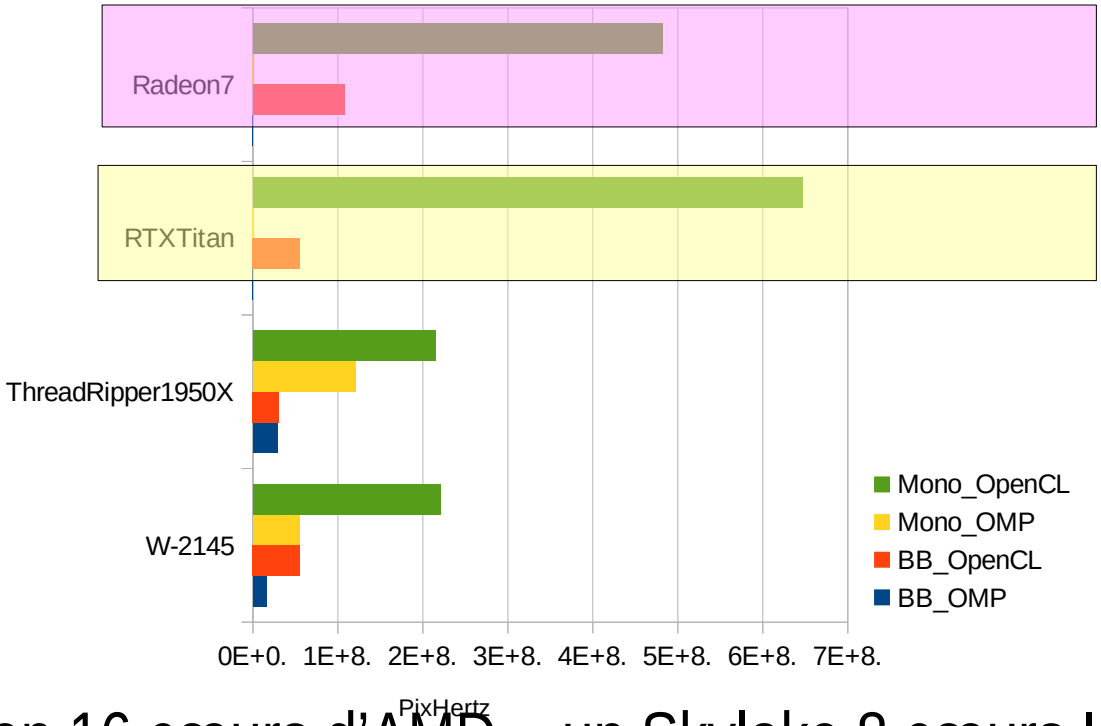
- Un x4 en Mono et en BB pour le GPU GTX780Ti
- Un x10 en BB et seulement x3 en Mono pour le GPGPU K40m
- Un x4 en BB et seulement un x1.4 en Mono pour le MIC Phi 7120P

En 2016, Pascal, Broadwell, Fiji 3 nouvelles générations



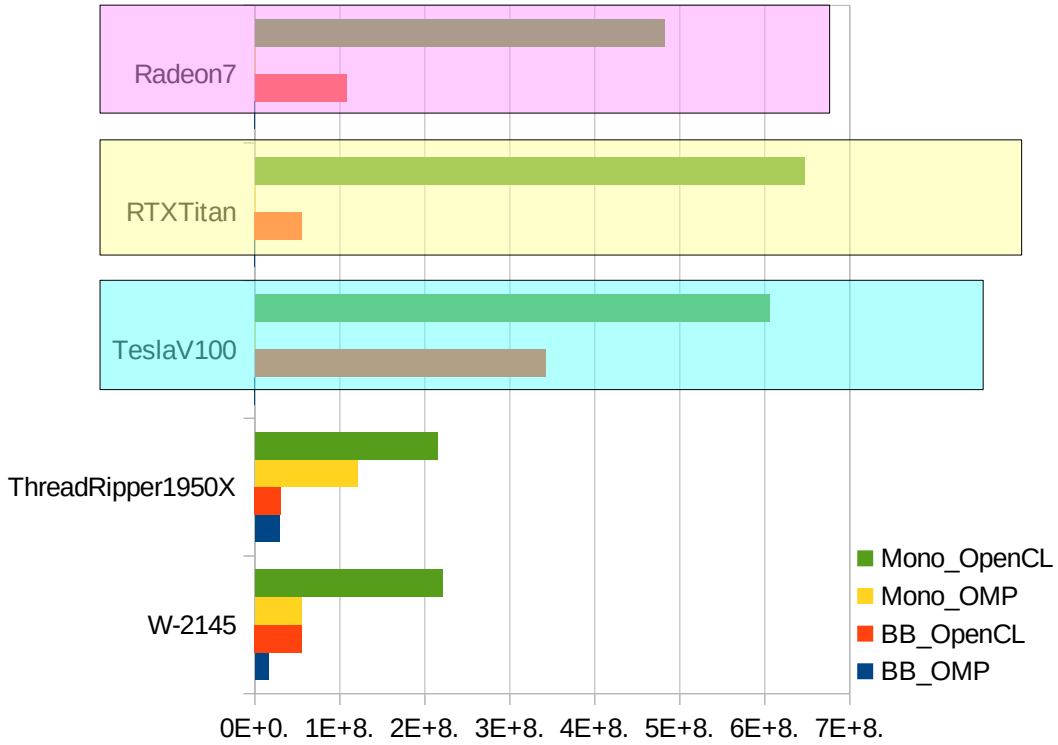
- La Gamer l'emporte sur la GPGPU en Mono (légèrement)
- La Tesla met un x4 au GPU, un x8 à AMD et un x10 au CPU
- Le AMD Fiji est au niveau du CPU Broadwell

En 2019, Turing, Vega2 en GPU Skylake, Zen en CPU, **sans** V100



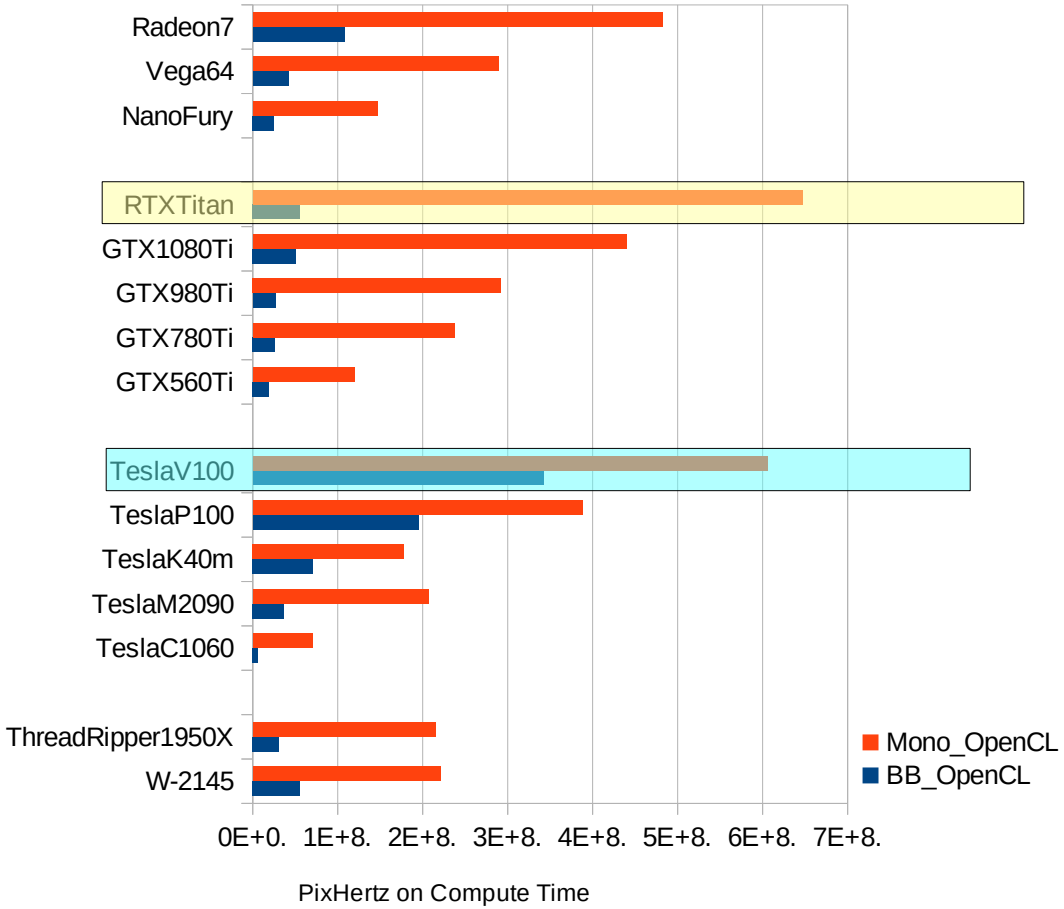
- Un Zen 16 cœurs d'AMD ~ un Skylake 8 cœurs Intel
- Radeon 7 : x2.5 en Mono, x2 en BB (vs Best CPU)
- Turing RTX : x3 en Mono mais comparable en BB (vs Best CPU)

Sortie en 2017, la Tesla V100 Seulement fin 2019 au CBP



- Tesla V100 ~ RTX Titan en Mono, x6 en BB
- Tesla V100 ~ Radeon 7 en Mono, x3 en BB
- CPU à x3 en Mono, mais x10 en BB

Synthèse en OpenCL
GPU vs meilleurs des CPU



Selon la simulation (BB ou Mono), c'est Gamer ou Tesla !

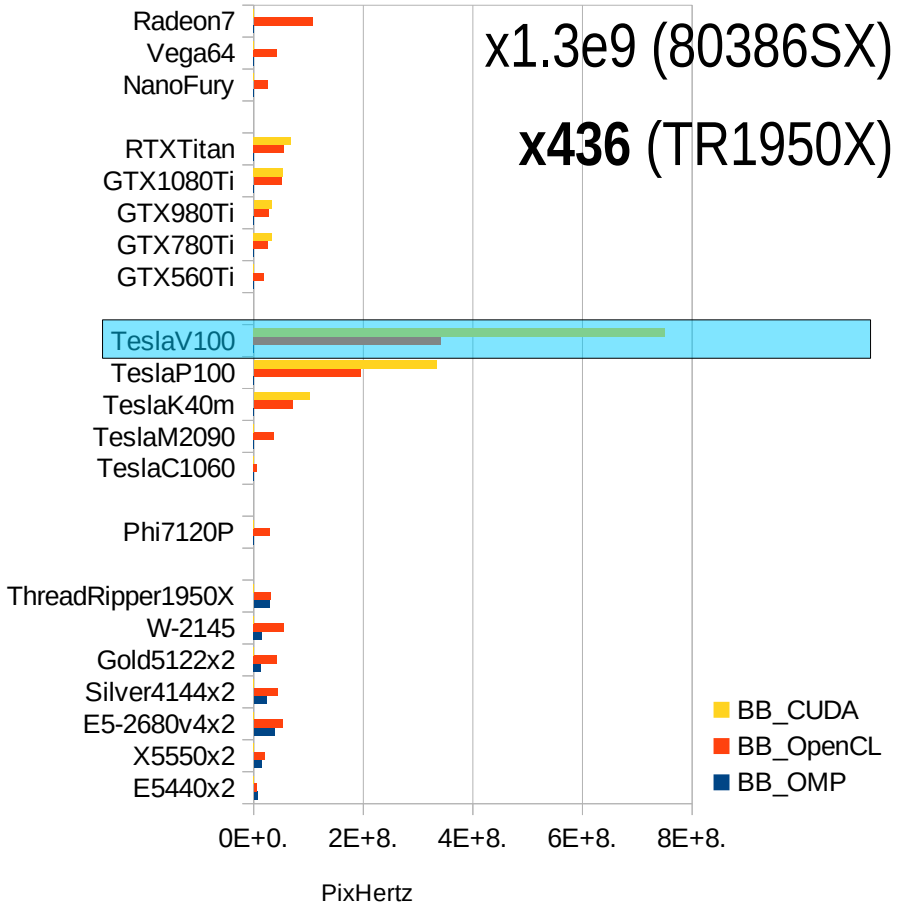
Et CUDA dans tout ça ? d'OpenCL en CUDA avec Python

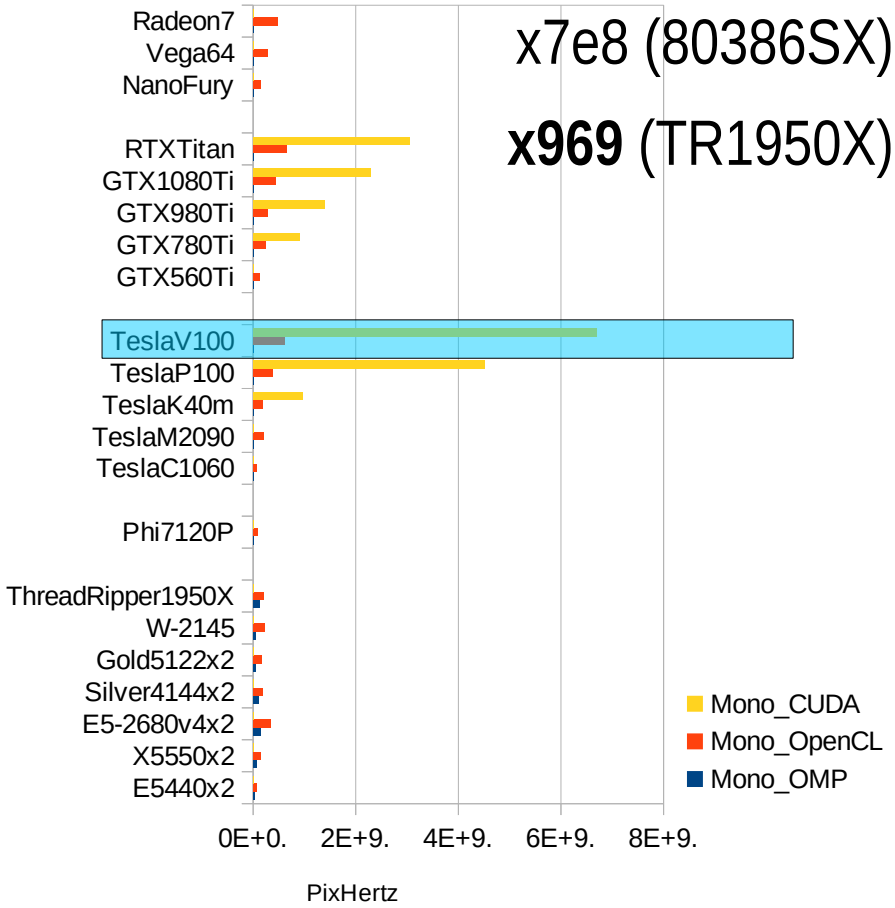
- Approche « naïve » :
  - Copier/Coller de tous les noyaux OpenCL
  - Remplacement des « workitem » par les « BlockIdx »
  - Préfixe des fonctions « internes » des noyaux par « __device__ »
- Petit test en BB, Tesla P100 :
  - En OpenCL : 1.031377 s
  - En CUDA : 24.033100 s **(soit 24 fois plus lent !!!)**
- Petit test en Mono, Tesla P100 :
  - En OpenCL : 0.87633 s
  - En CUDA : 1.003969 s **(un peu moins bon)**
- Bref, ça rappelle un souci déjà rencontré...

En CUDA, l'attaque des threads…

- Exploitation de 2 étages de parallélisme
  - Définition de « grid » et de « threads »
  - Découpage dans les noyaux des tâches mode « pagination »
    - En OpenCL : uint xi=(uint)get_global_id(0);
    - En CUDA : uint xi=(uint)(blockIdx.x*blockDim.x+threadIdx.x);
- Nouveau test en BB, Tesla P100, Mode TrajectoPixel :
  - Threads à 1 : 24.033100 s
  - Threads à 32 et 32^2 : 0.810377 s (soit x30/$CUDA_1$, légèrement meilleur OpenCL)
- Nouveau test en Mono, Tesla P100, Mode TrajectoPixel :
  - Threads à 1 : 1.003969 s
  - Threads à 32 et 32^2 : 0.063761 (soit x15/CUDA , x13/OpenCL)
- En passant en CUDA ; +20 % en BB mais x13 en Mono !

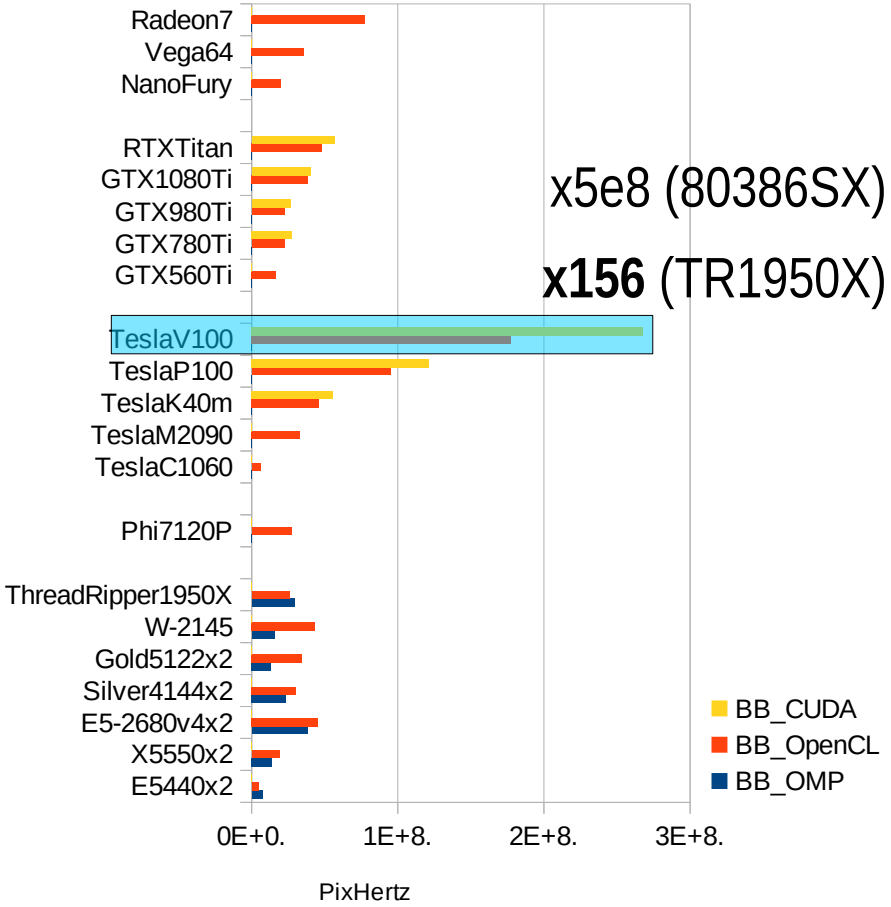
Donc, en rajoutant CUDA, Il y a la Tesla V100 et les autres...





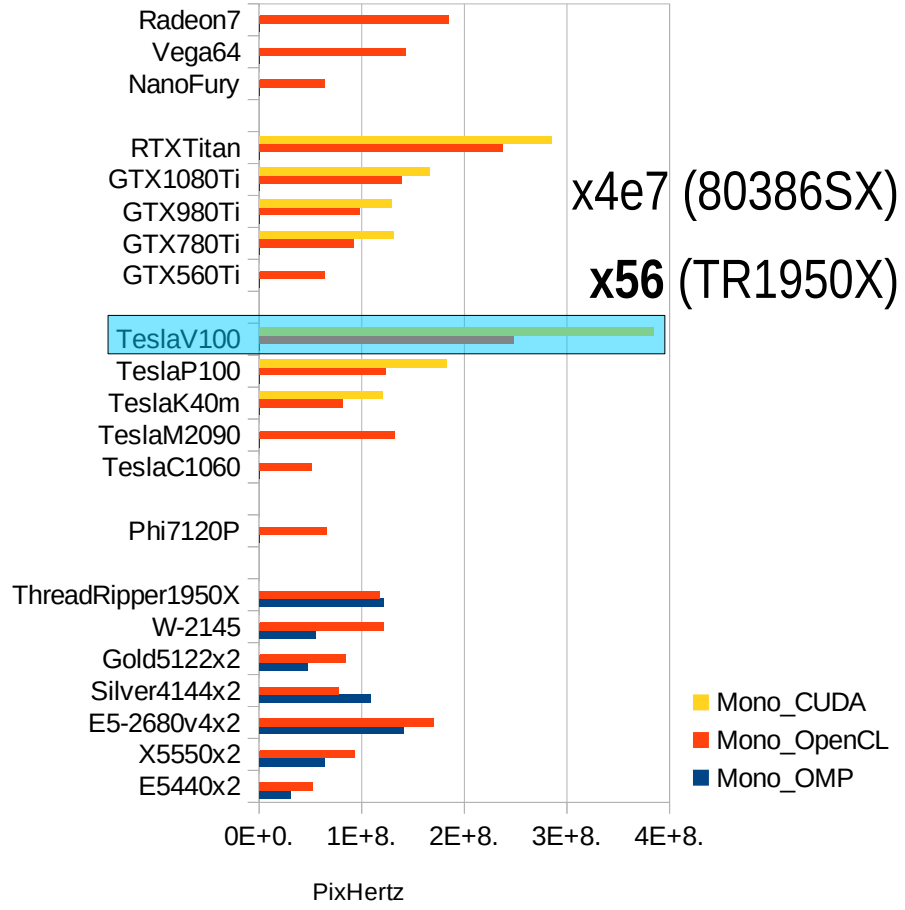
Face au CPU, le milliard en 30 ans, le millier en séquentiel

Mais je serai un gros menteur... Le transfert en CUDA et OpenCL...



x5e8 (80386SX)

x156 (TR1950X)

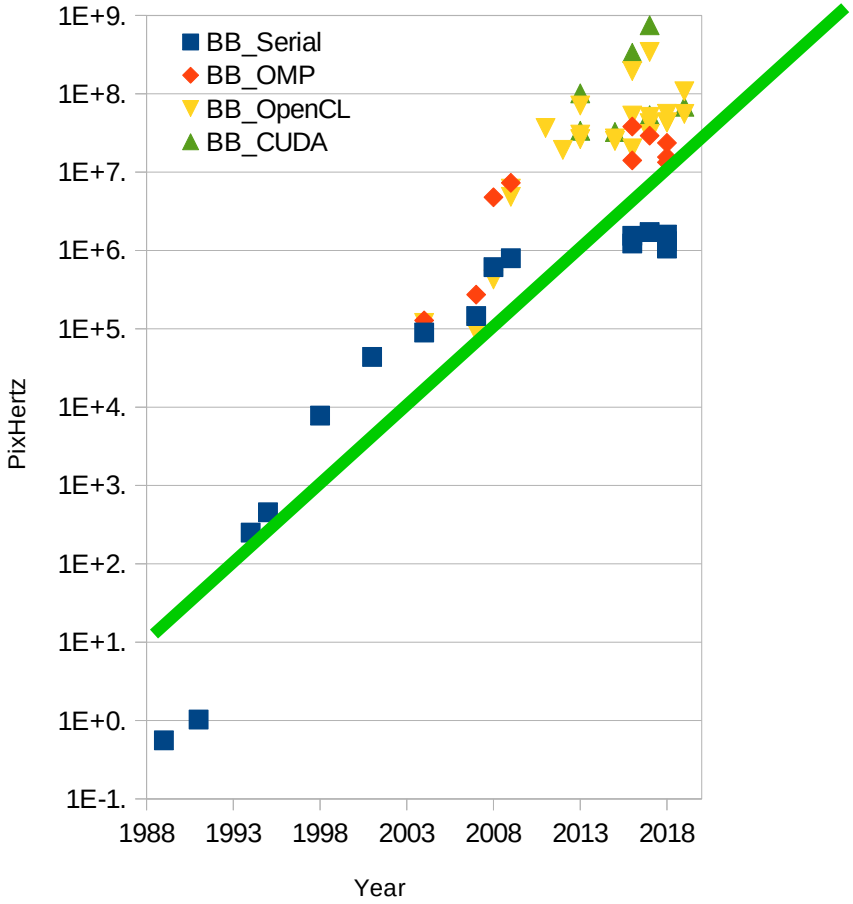


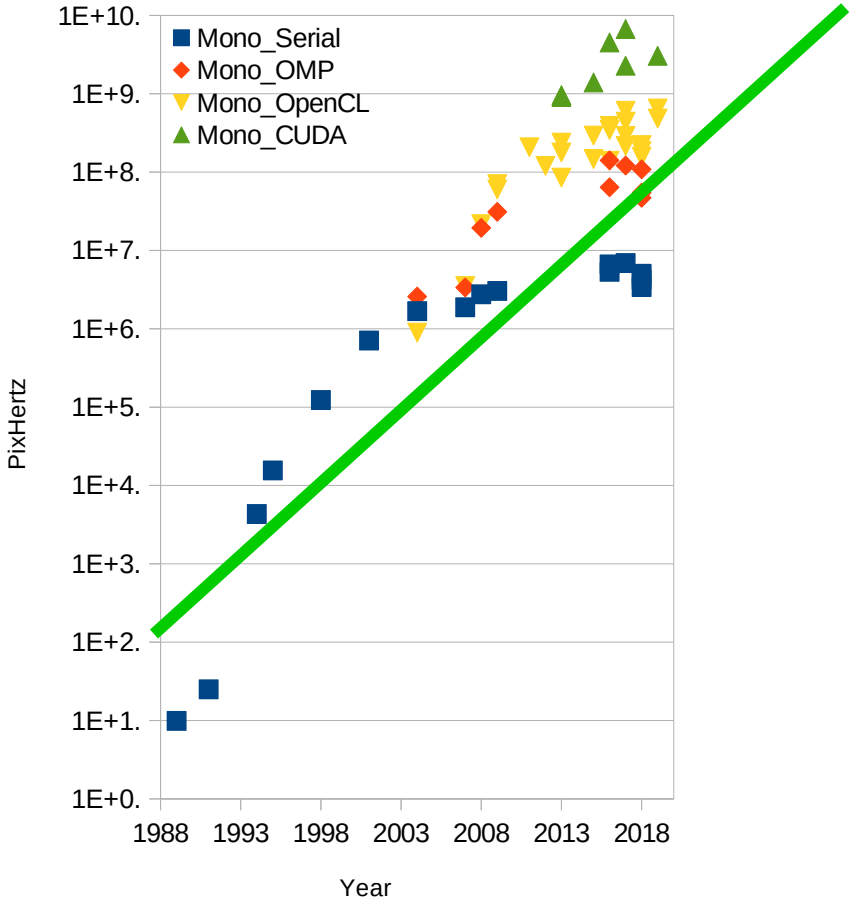
x4e7 (80386SX)

x56 (TR1950X)

Bref, dans ce cas, temps du transfert > temps de calcul

Et la loi de Moore* dans tout ça ?
Grâce à CUDA, on est sur la droite !



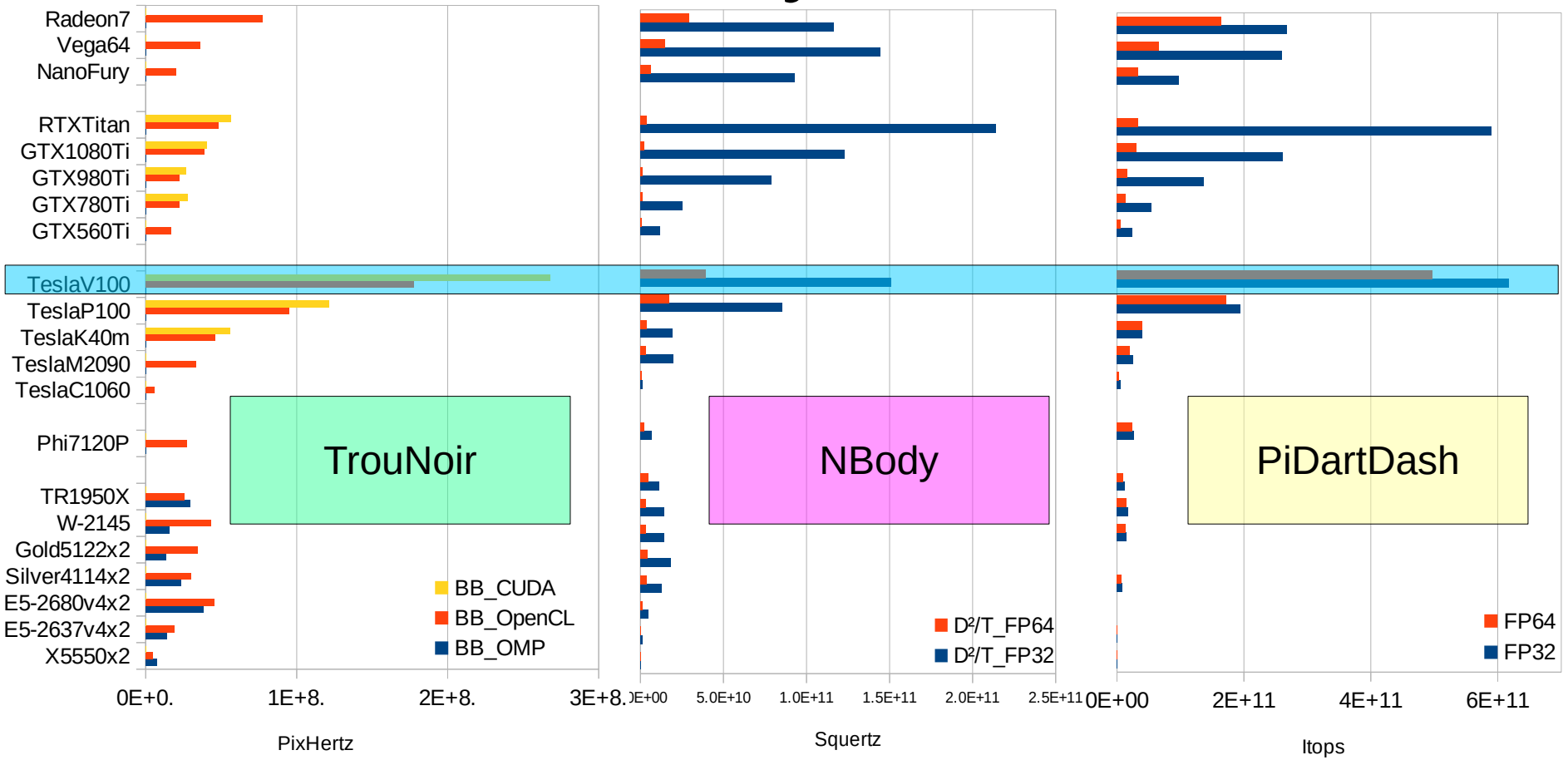


Le (GP)GPU : même caractère « disruptif » que le FPU...

En conclusion de ce portage… … et son exécution sur nos *PU

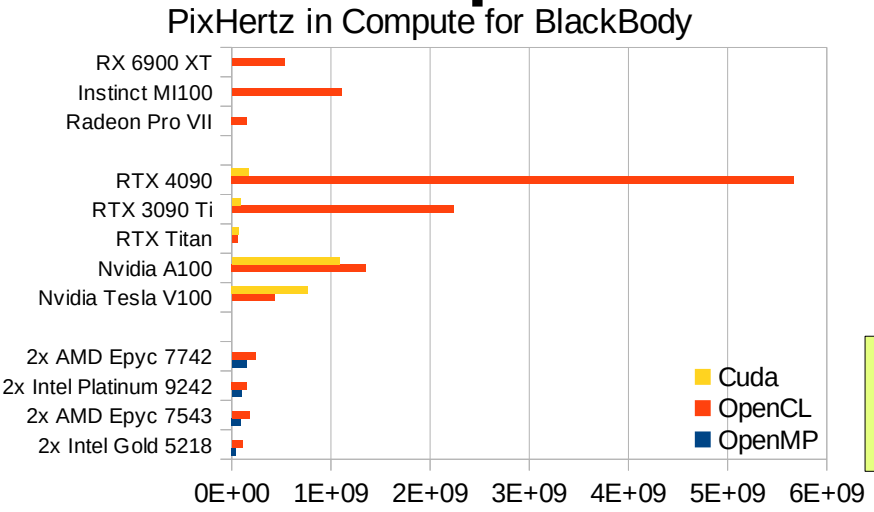
- Le Code, le chemin & les gains (temps écoulés) :
  - De Fortran à C en 1997 : x1
  - De C à C/OMP en 2019 : x22 (pour un 28 cœurs)
  - De C/OMP à Python/OpenCL sur CPU : x25 (pour un 8 cœurs)
  - Du CPU au GPU en Python/OpenCL : de x36 à x106
  - De Python/OpenCL à Python/CUDA : de x56 à x156
- Bref…
  - Il n'y a pas que CUDA ou C++ dans la vie… Il y a Python et OpenCL !
  - Vous pouvez tester :
    - http://forge.cbp.ens-lyon.fr/redmine/projects/bench4gpu/repository/show/TrouNoir

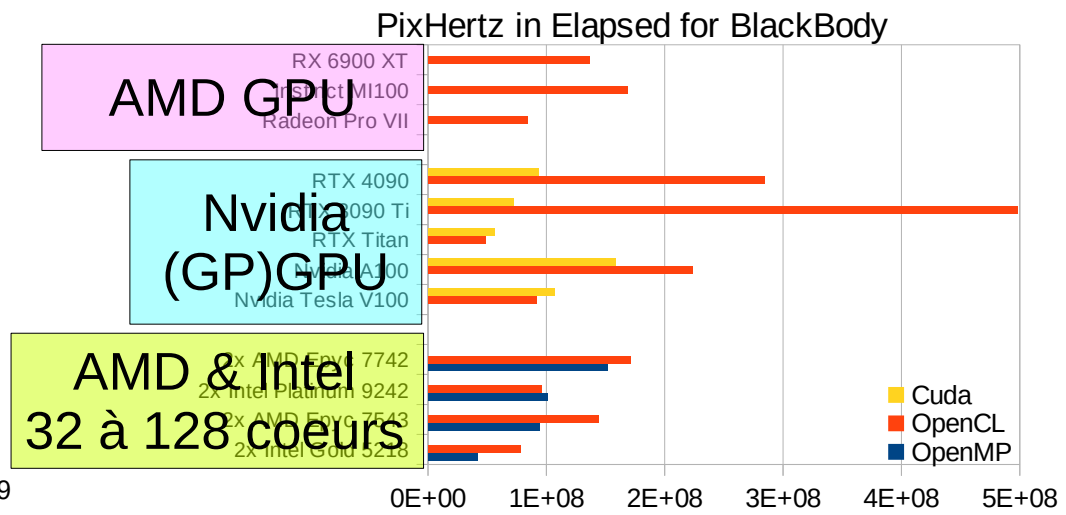
Comparaison aux autres codes… TrouNoir, Nbody et PiDartDash

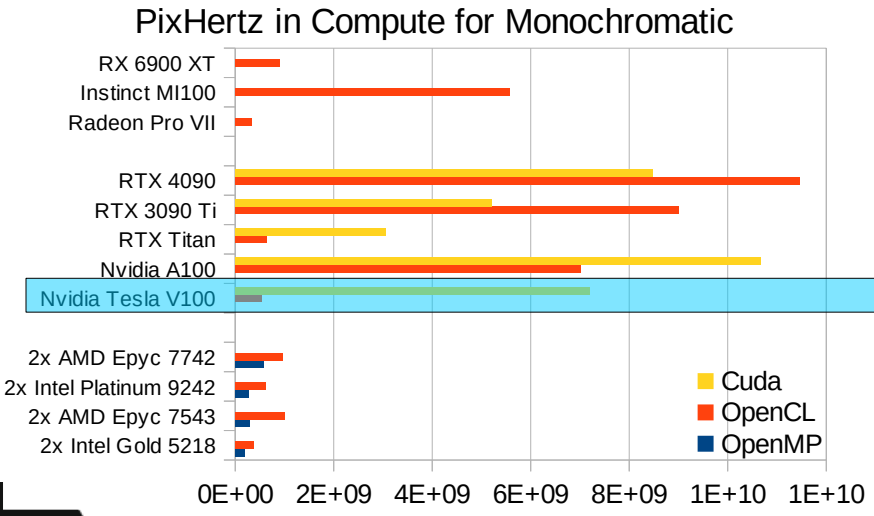


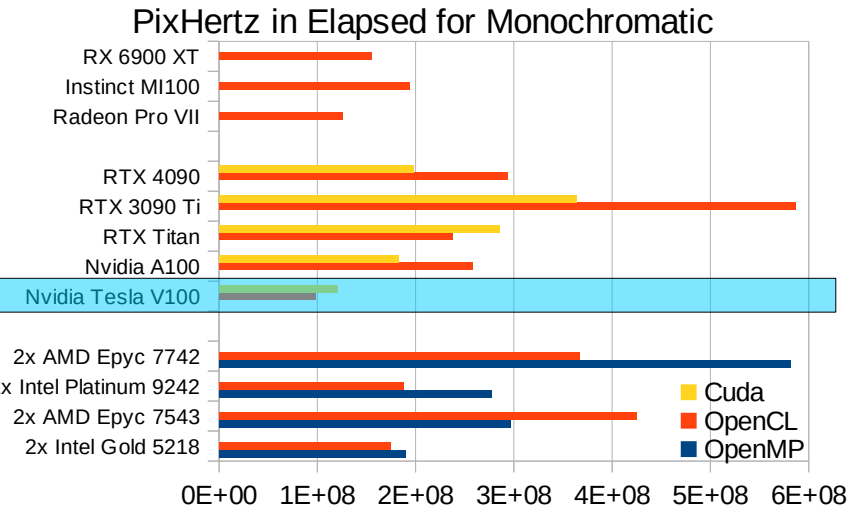
En conclusion, pour les GPU, calculs pas trop compliqués...

3 ans plus tard, larges évolutions ! Ada, Ampere, Navi, Rome, Cascade







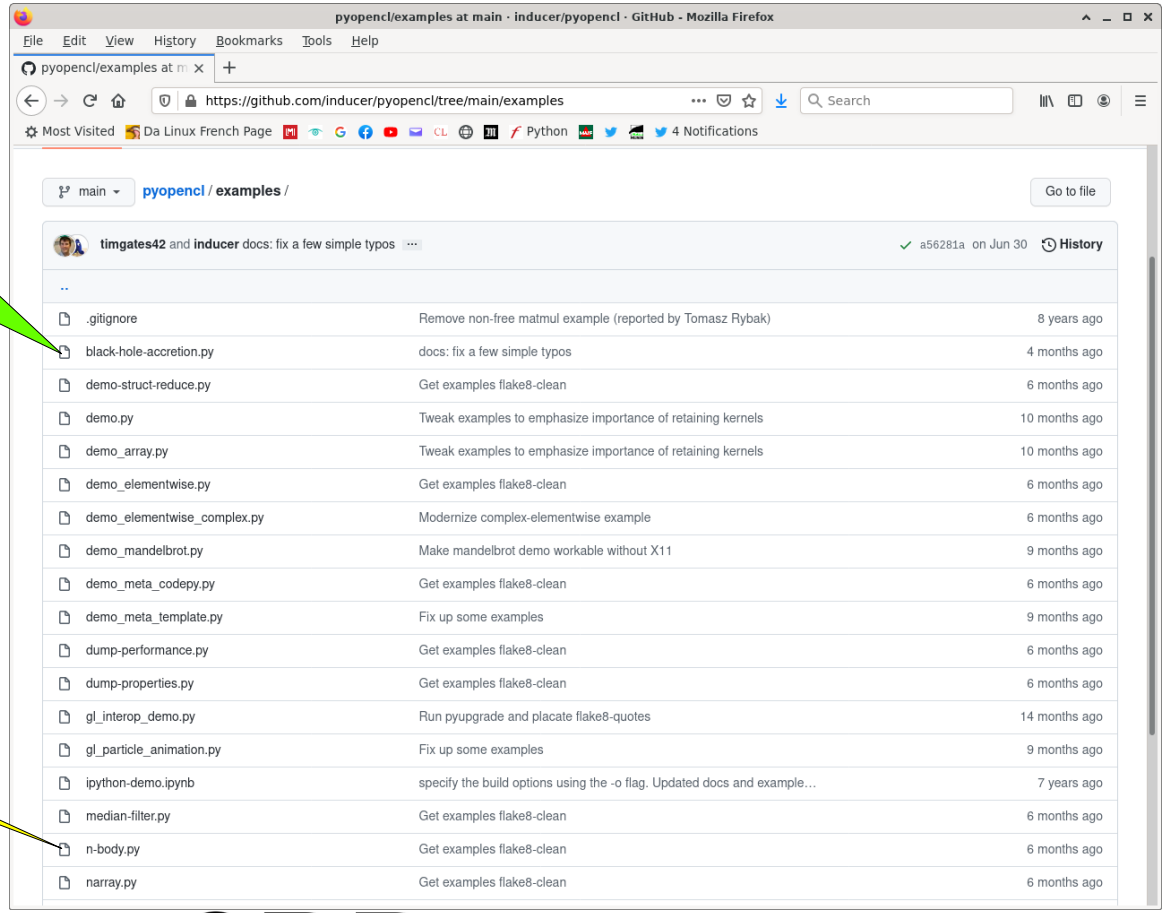


Et « TrouNoir.py » maintenant…
Devenu « black-hole-accretion.py »

Site PyOpenCL : https://github.com/inducer/pyopencl/tree/main/examples

black-hole-accretion.py

n-body.py



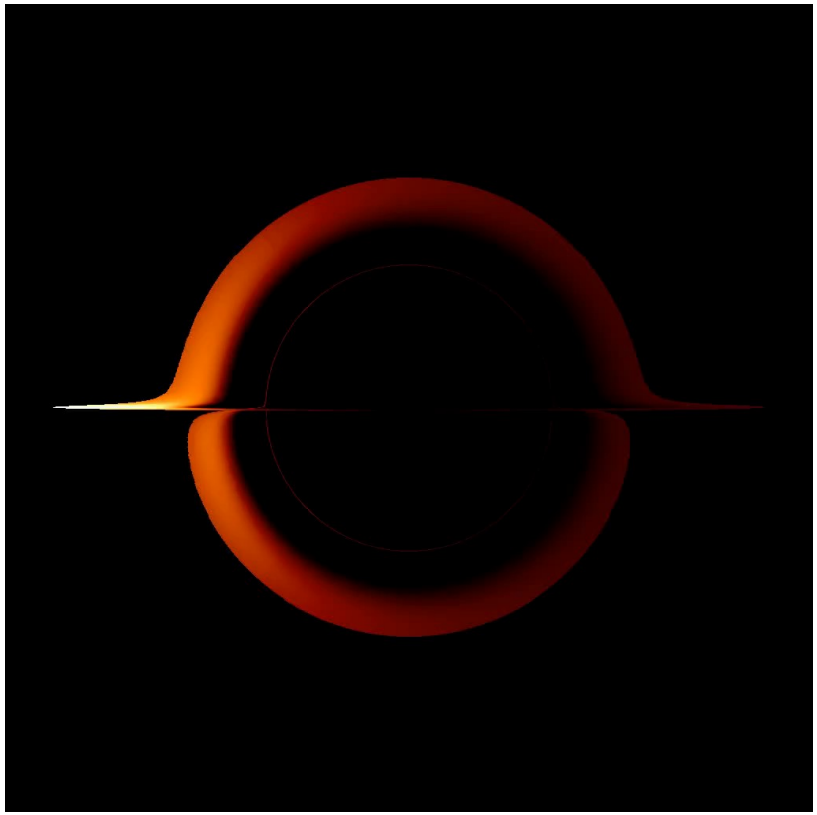
En conclusion pour TrouNoir.py...

- L'informatique a évolué en 30 : des gains colossaux
  - Principaux gains : FPU (début 80), GPU (début 2010)...
- En 2021, puissance de calcul : GPGPU puis HugeCPU
- Paralléliser ou gépufier un code : une question de temps
  - C'est possible, mais il faut qu'il soit bien « conditionné »
  - En OpenMP(ou OpenACC), parfois, distribuer des boucles suffit
  - En OpenCL, copier son code C dans du Python et appeler les noyaux
  - En CUDA, s'inspirer de OpenCL et exploiter les « étages » de //isation
- Constater que tout n'est que « cas d'usage »

Annuaire de « bonnes pratiques » Pour éviter la déception...

- Le constat de 2022 (différent de 2019) :
  - Le (GP)GPU est performant, mais plus le meilleur
- Se poser les « bonnes questions » :
  - Combien de tâches indépendantes puis-je lancer ? >100000 ?
  - Quelles sont les communications entre tâches ?
  - Quel est le « grain » de mon application ?
  - Quelle doit être la pérennité de mon code ?
  - Quel doit être l'oecuménisme de mon code ?

Des questions ?



- Remerciements à :
  - Pierre Léna
  - Jean-Pierre Luminet
  - Didier Pelat
  - Hervé Aussel
  - Ralf Everaers
  - Marine, Yoann, Léna