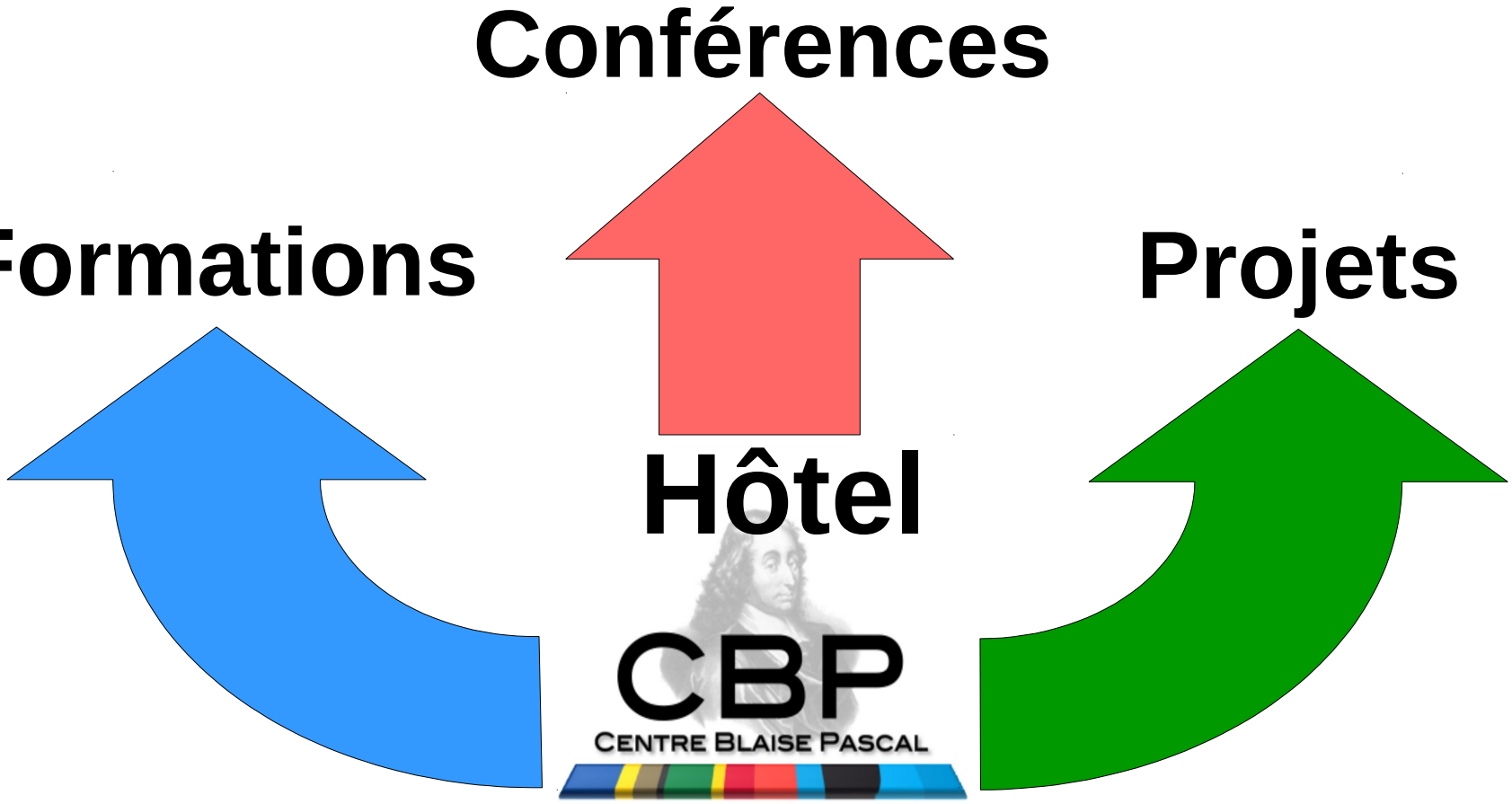
Des ressources du CBP en général à l'exploitation de sa « collection » de (GP)GPUs en particulier

Emmanuel Quemener

Le Centre Blaise Pascal
« Maison de la modélisation » et ...



Centre Blaise Pascal ~ Dryden FR
Un petit exemple illustratif

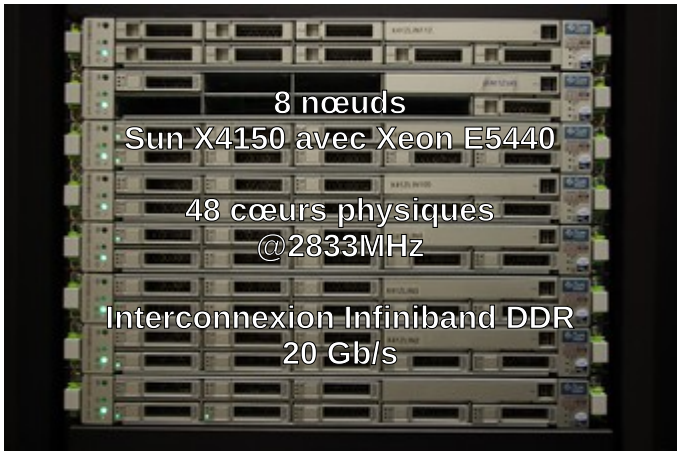
- Nasa X-29
  - Cellule de F-5
  - Moteur de F-18
  - Train de F-16
- Études
  - Empennages canards
  - Incidence >50°
  - « Fly-By-Wire »

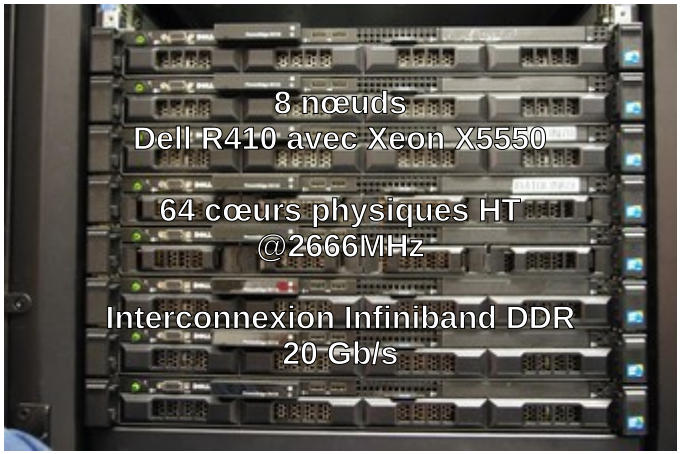
Recycle, Réutilise et explore de nouveaux domaines

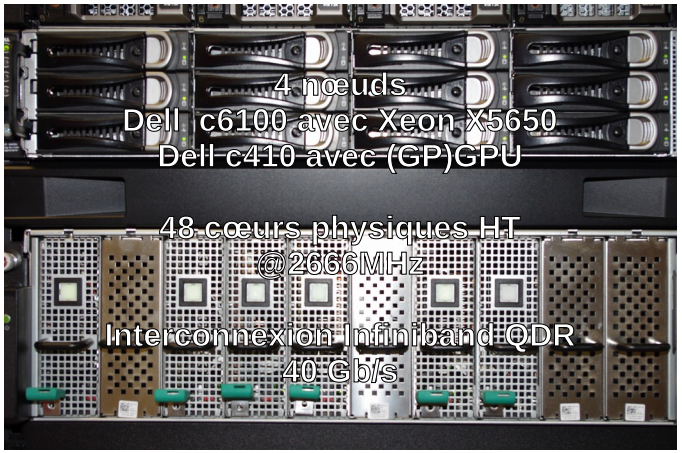
Une Plateforme expérimentale
10 plateaux techniques permanents

- Multi-nœuds : 5 grappes de 4 à 64 nœuds
  - Nœuds/Cœurs : 64/512, 8/64, 8/64, 4/48, ~~8/128~~
- Multi-cœurs : 30 de 2 à 28 cœurs, de 1.8 à 3.5 GHz
- (GP)GPU : 45 modèles différents de GPU (AMD & Nvidia)
- Intégration : 20 machines virtuelles : Debian de Lenny à Sid en 32 & 64 bits, ...
- Matériel exotique : 3 ARMv7 sous Debian Jessie ou Ubuntu
- Plateau technique 3D :
  - 2 stations, 2 vidéoprojecteurs, 20 moniteurs, 4 paires de lunettes
- COMOD : « Compute On My Own Device »
  - Même Single Instance Distributing Universal System (SIDUS)

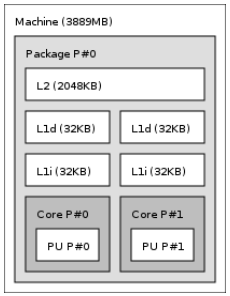
Plateau multi-nœuds : 5 «grappes» 92 nœuds, 3 réseaux spécifiques

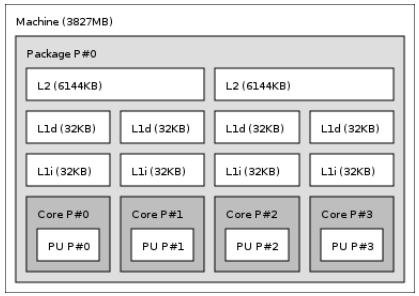


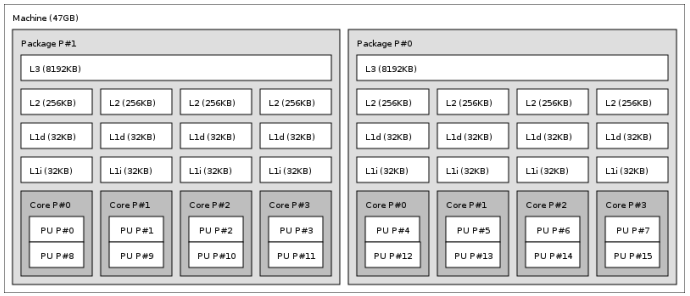


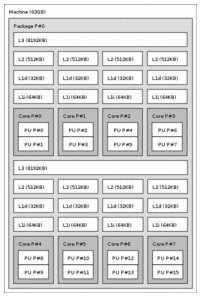


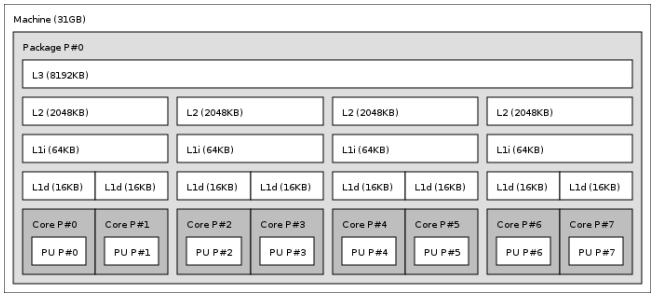
Plateau multi-cœurs
de 2 à 28 cœurs : petit bestiaire...

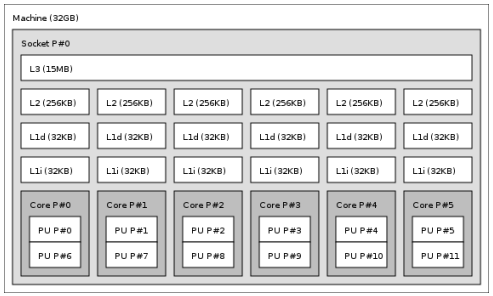


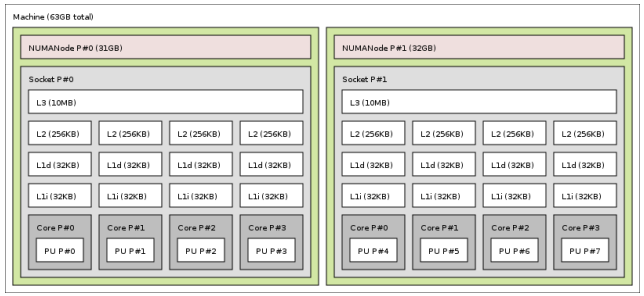


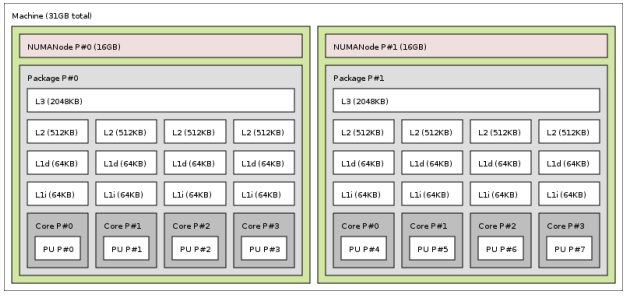


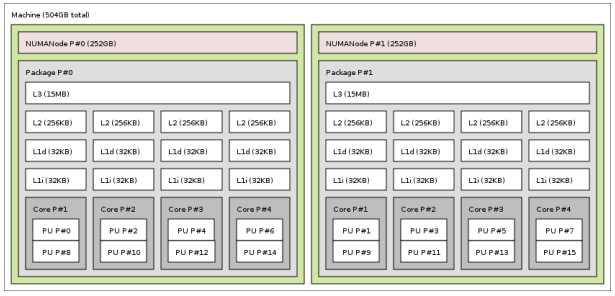












Plateau multi-shaders : (GP)GPU 45 modèles différents...

GPU Gamer : 15

- Nvidia GTX 560 Ti
- Nvidia GTX 680
- Nvidia GTX 690
- Nvidia GTX Titan
- Nvidia GTX 780
- Nvidia GTX 780 Ti
- Nvidia GTX 750
- Nvidia GTX 750 Ti
- Nvidia GTX 960
- Nvidia GTX 970
- Nvidia GTX 980
- Nvidia GTX 980 Ti
- Nvidia GTX 1050 Ti
- Nvidia GTX 1080
- Nvidia GTX 1080 Ti

GPGPU : 9

- Nvidia Tesla C1060
- Nvidia Tesla M2050
- Nvidia Tesla M2070
- Nvidia Tesla M2090
- Nvidia Tesla K20m
- Nvidia Tesla K40c
- Nvidia Tesla K40m
- Nvidia Tesla K80
- Nvidia Tesla P100

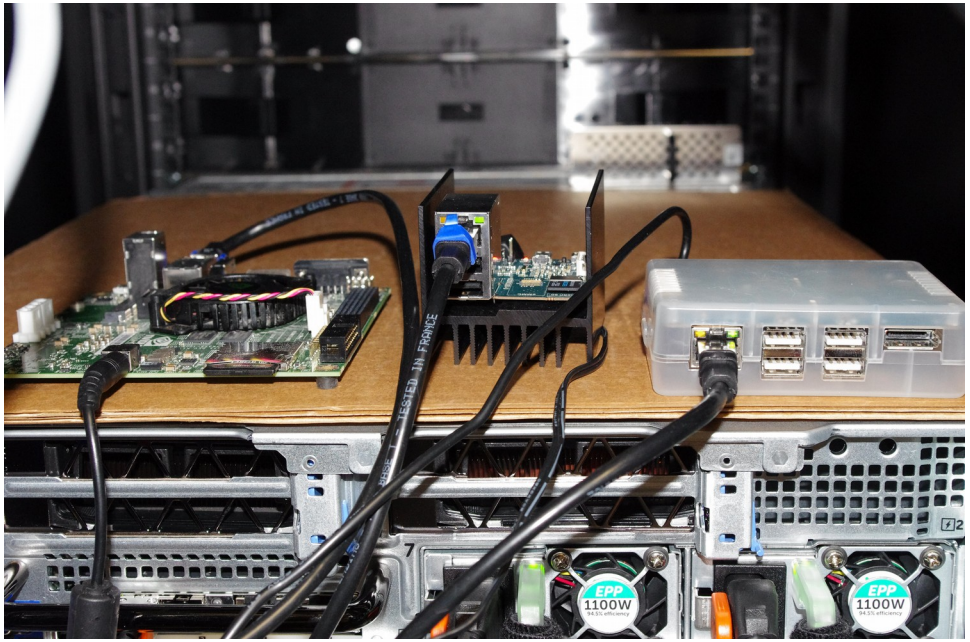
GPU desktop & pro : 11

- Nvidia Quadro 600
- Nvidia Quadro 4000
- Nvidia Quadro K2000
- Nvidia Quadro K4000
- Nvidia NVS 310
- Nvidia NVS 315
- Nvidia GT 430
- Nvidia GT 620
- Nvidia GT 640
- Nvidia GT 710
- Nvidia GT 730

GPU AMD Gamer & al : 11

- AMD Radeon HD 5850
- AMD Radeon HD 7970
- AMD R9 380
- AMD R9 290X
- AMD R9 295x2
- AMD R9 Fury
- AMD Nano Fury
- AMD Radeon HD 6450
- AMD Radeon HD 6670
- AMD R7 240
- AMD Kaveri A10-7850K

Plateau « architectures exotiques »
L'informatique d'avant, d'à côté...



- Processeur ARM
- SoC Exynos

Et de demain !

Plateau d'intégration logicielle
Distributions GNU/Linux : 32 et 64 bits

debian

- Lenny 5
- Squeeze 6
- Wheezy 7
- Jessie 8
- Stretch 9
- Sid

ubuntu

- Yakkety Yak 10.04
- Utopic Ucorn 12.04
- Precise Pangolin 14.04
- Lucid Lynx 16.04

CentOS

- 5.5
- 7

Plateau « Visualisation 3D »

- Pourquoi ?
  - S'immerger dans la scène d'une visualisation physique
- Quoi ?
  - 1 station dans la salle de formation : q4000left + moniteur 24p
  - 1 station dans la salle de réusion : k4000 + moniteur 27p + vidéoprojecteur
  - 4 paires de lunettes 3D
- Comment ?
  - Demander les lunettes (contre une caution)
  - Accès direct obligatoire
  - Applications validées : glxgears, PyMol, VMD, Paraview

Plateau Technique COMOD
Compute On My Own Device

- Vous connaissez le BYOD : Bring Your Own Device
  - Travailler avec son équipement personnel
- COMOD se propose :
  - Disposer en quelques secondes d'un environnement fonctionnel
  - D'exploiter sa machine pour son travail dans des tâches de HPC
  - De se déployer sur une machine complète ou virtuelle (VirtualBox)
- COMOD s'appuie :
  - Sur l'approche SIDUS Single Instance Distributing Universal System
  - L'infrastructure de stockage du Centre Blaise Pascal
  - L'authentification de l'école : le même identifiant, mot de passe

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, enseignants, chercheurs, ingénieurs, ...
- Quand & Où ?
  - Centre Blaise Pascal : depuis 2010, plus de 100 machines
  - PSMN : depuis 2011, plus de ~500 nœuds (sa propre instance)
  - Laboratoires : UMPA, LBMC, IGFL
- Comment ?
  - Utiliser un partage en réseau d'une arborescence
  - Détourner une ruse de LiveCD

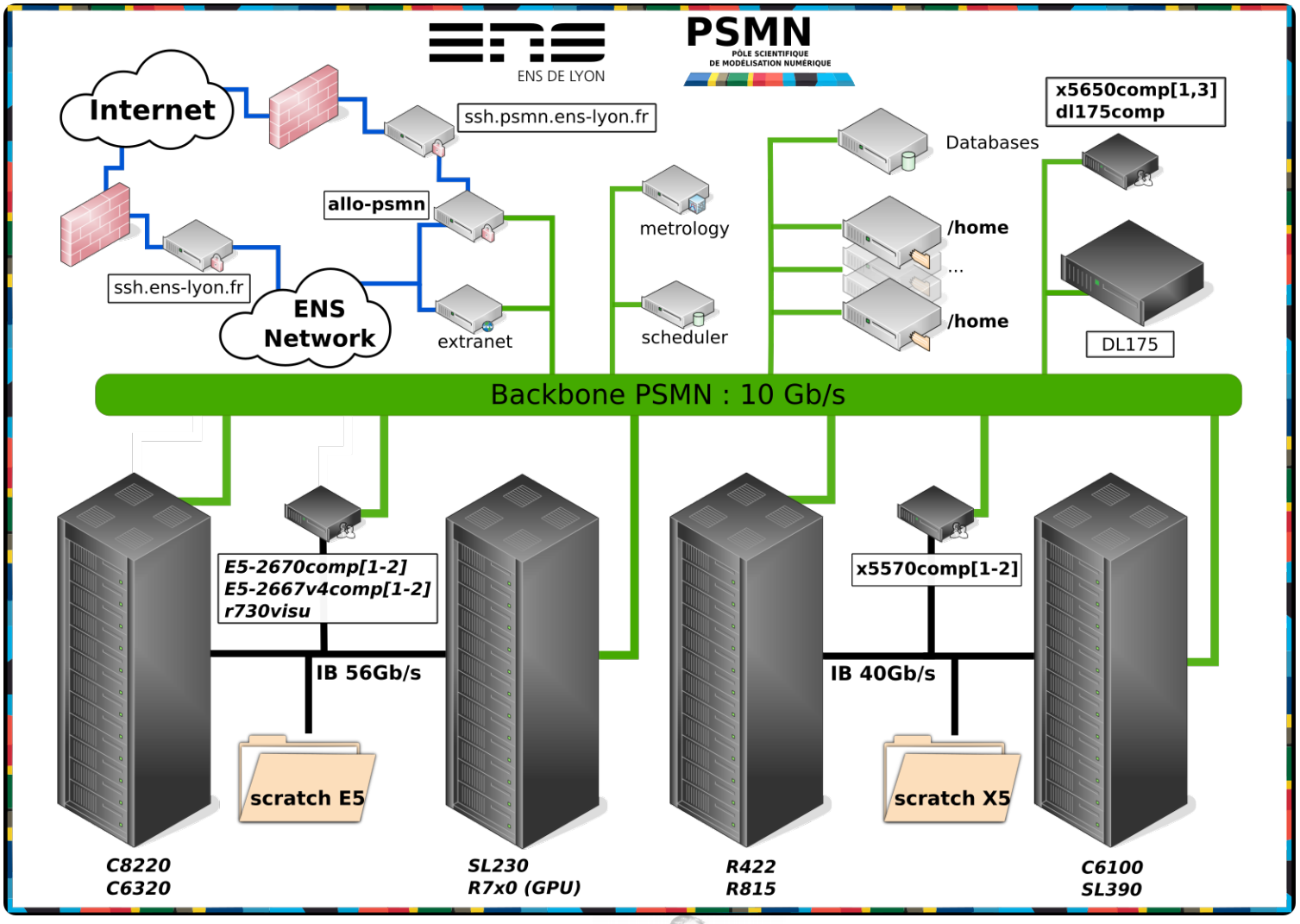
« Deux machines ayant démarré SIDUS ne peuvent pas ne pas avoir le même système ! »

Des « paillasses numériques » permanentes ou éphémères !

- Ecole des Houches « computational physics » : 6 éditions 2011-2016
  - Machine virtuelle « standalone » complète
  - Machine PXE pour SIDUS avec tout l'environnement
- Humanités Numériques :
  - 20 machines dont 10 actives pour les Humanités Numériques
- Géographie :
  - Postes de visualisation & traitement graphique
- Biologie :
  - **Portail Galaxy (avec exploitation de cluster)**
  - Postes de traitement & visualisation graphiques pour MorphoGraphX
  - Serveur pour étude (et traitements) Repeat* sous GlusterFS, TMPFS & BRD

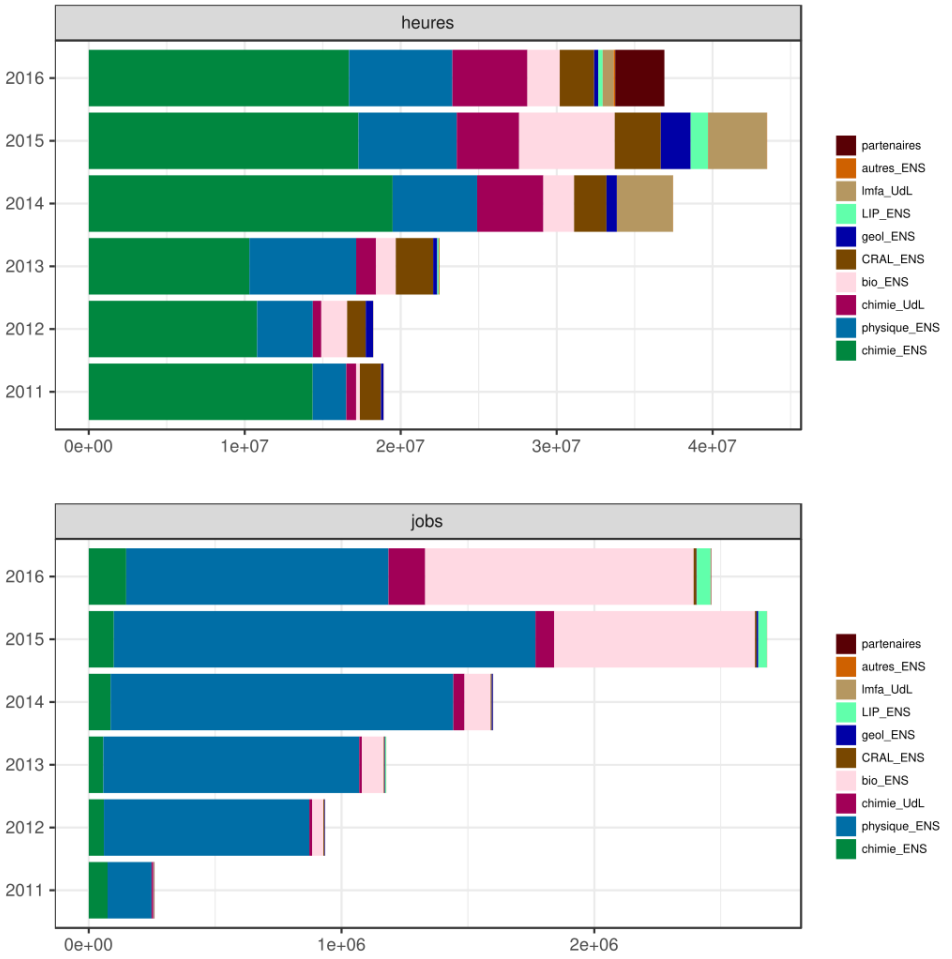
Méso-centre de l'ENS-Lyon : PSMN

Pôle Scientifique de Modélisation Numérique



Méso-centre PSMN
Son utilisation en jobs & durées

- Exploration
  - c6100+c410x
  - OmniPath
- Exploitation d'études
  - SIDUS
  - GlusterFS
  - SIDUS GPU
  - X2go/VirtualGL



Un exemple d'interaction
Entre CBP, laboratoire, Centre de Calcul

Évaluation
Centre d'essais

Achat GPU Adapté

Savoir-Faire SIDUS
Systèmes & Réseaux

AMD

Nvidia

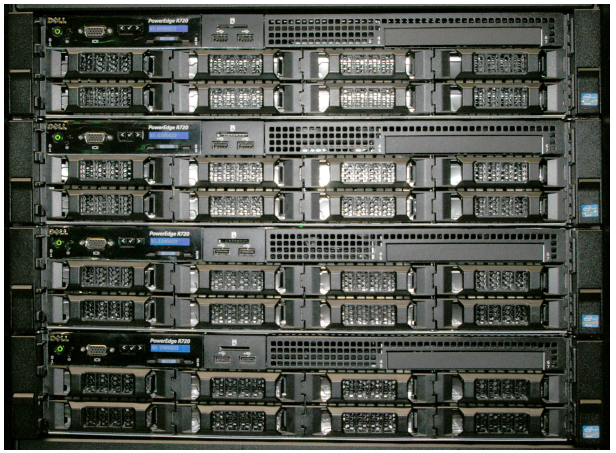
Matériel **CBP** / Sidus OS **CBP**

Pré-production
Laboratoire

Matériel **LBMC** / Sidus OS **CBP**

Influence de l'achat du socle matériel

Production
Centre de Calcul



Matériel **PSMN** / Sidus OS **PSMN**

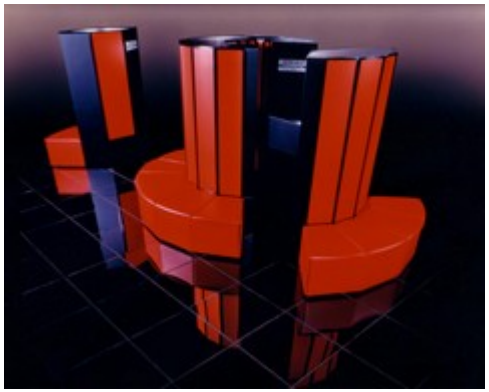
Quel regard sur la discipline ? Celui d'un physicien

- Approche « système » du physicien
  - Héritage des calculateurs analogiques
  - Le « système » informatique :
    - réseau/matériel/OS/logiciel/codes/usages/...
- Approche « Saint Thomas »
  - Appréhension par sa seule mesure
- Approche « pilote d'essais »
  - Caractérisation, recherche d'une exploitation optimale...

Une génération (humaine)... Un film de série B en 1984

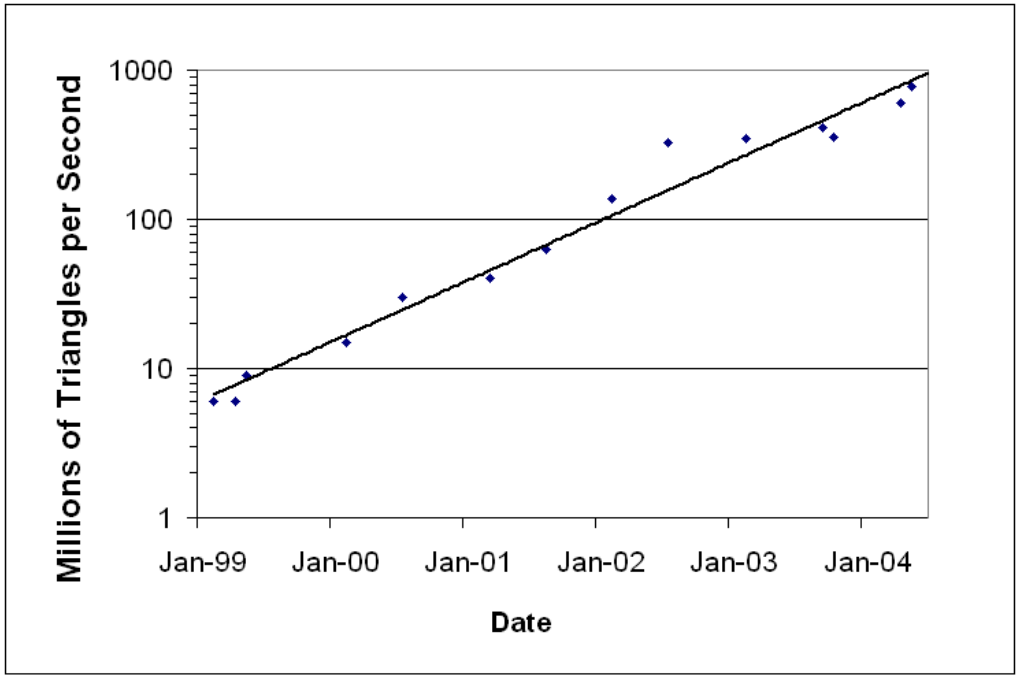
- 1984 : The Last Starfighter
  - 27 minutes d'image de synthèse
  - 22.5 10^9 opérations par image
  - Un Cray X-MP (130 kW)
  - 66 jours (en fait, 1 année nécessaire)

- 2016: sur la GTX 1080Ti (225 W)
  - 3.4 secondes
  - Comparaison GTX1080Ti / Cray
- Performance : x1600000 !
- Consommation /900000000 !



Pourquoi le GPU est « disruptif » ? Le « grand détournement » en HPC

- Dans une conférence à l'ENS-Cachan en février 2006, ceci...
  - x100 en 5 ans



- Entre 2000 et 2015
  - GeForce 2 Go/GTX 980Ti : de 286 à 2816000 MOpérations/s : x10000
- Pour un CPU « classique » : x100

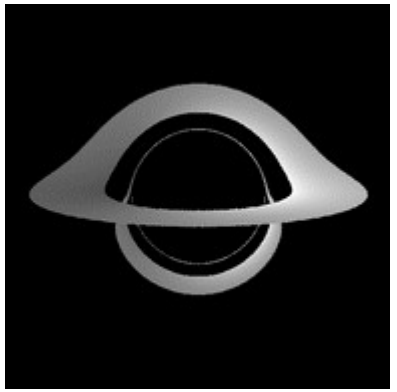
Au commencement de toute chose

- Nvidia lance le « superordinateur personnel »
  - Quand : le 19/11/2008 Qui : Nvidia
  - Où : sur Tom's Hardware
  - Quoi : une carte PCIe C1060 PCIe avec 240 cœurs
  - Combien : 933 Gflops SP (mais 78 Gflops DP)

Position du GPU face aux autres? Les autres « accélérateurs »

- Accélérateur ou la vieille histoire des coprocesseurs...
  - 1980 : 8087 (sur 8086/8088) pour les opérations en virgule flottante
  - 1989: 80387 (sur 80386) et son respect du IEEE 754
  - 1990 : 80486DX et l'intégration du FPU dans le CPU
  - 1997 : K6-3DNow ! & Pentium MMX : SIMD dans le CPU
  - 1999 : fonctions SSE et le début d'une longue série (SSE4 & AVX)
- Quand les circuits restent hors du CPU
  - 1998 : Les DSP de la catégorie des TMS320C67x comme outils
  - 2008: le Cell dans la PS3, IBM dans le Road Runner & un Top1 au Top500
- Travail de compilateurs & forte dépendance au modèle

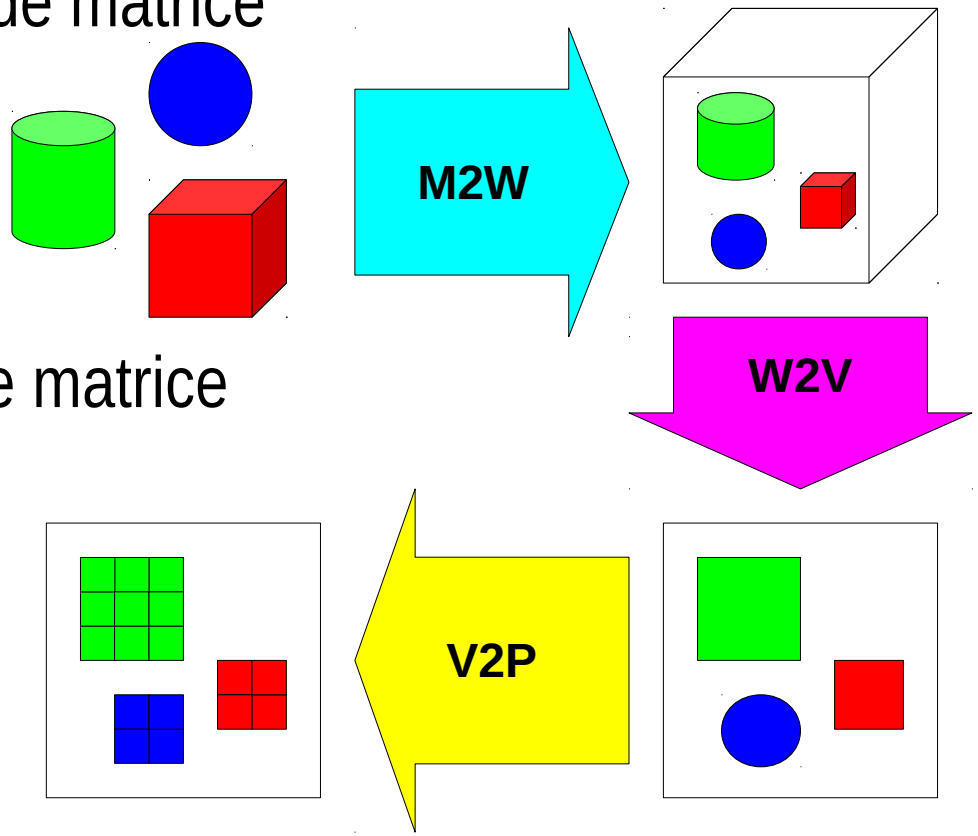
Pourquoi le GPU est-il si puissant ? Pour construire une image 3D !



- 2 approches:
  - Raytracing : PovRay (« dimensions » de l'UMPA)
  - Shadering : 3 opérations
- « Raytracing » :
  - De l'oeil vers les contacts de chaque objet
- « Shadering »
  - Model2World : objets vectoriels placés dans la scène
  - World2View : projection des objets dans un plan de vue vectoriel
  - View2Projection : pixellisation du plan de vue

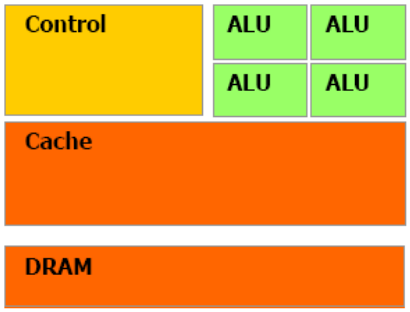
Pourquoi le GPU est-il si puissant ? Shadering & Calcul matriciel

- **Model 2 World** : 3 produits de matrice
  - Rotation
  - Translation
  - Mise à l'échelle
- **World 2 View** : 2 produits de matrice
  - Positionnement de la caméra
  - Direction de l'endroit pointé
- **View 2 Projection**
  - Pixellisation

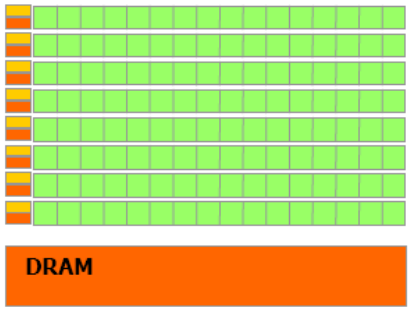


Donc, un GPU : c'est un « gros » Multiplicateur de Matrice

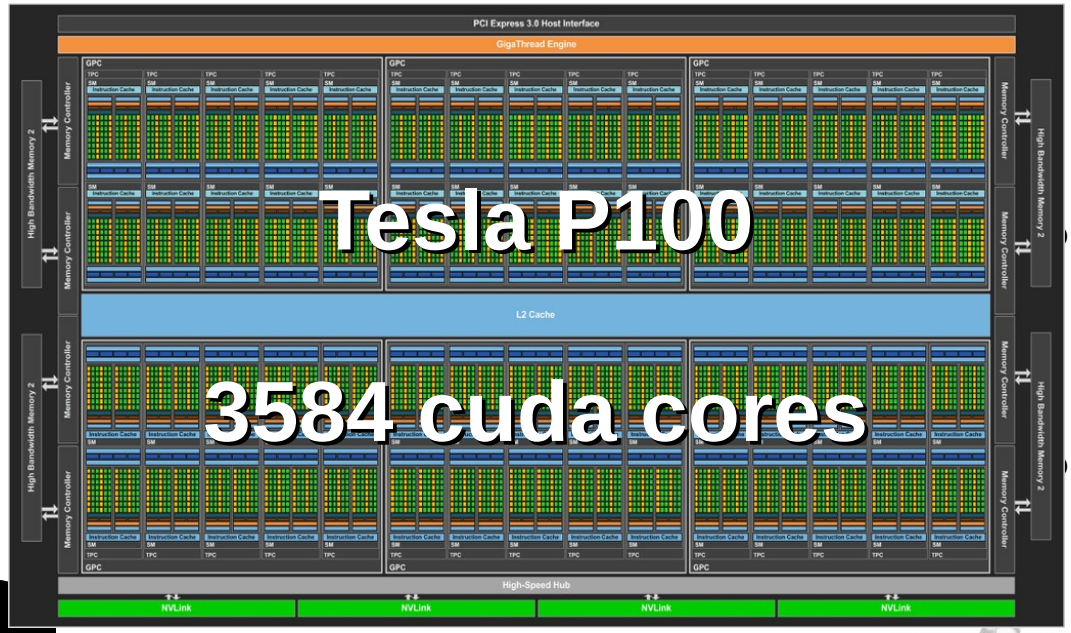
Combien de composants à l'intérieur ? Quelle différence entre GPU & CPU



CPU



GPU



- Opérations
  - Multiplication de Matrice
  - Vectorisation
  - « Pipelining »
  - Shader (multi)processeur

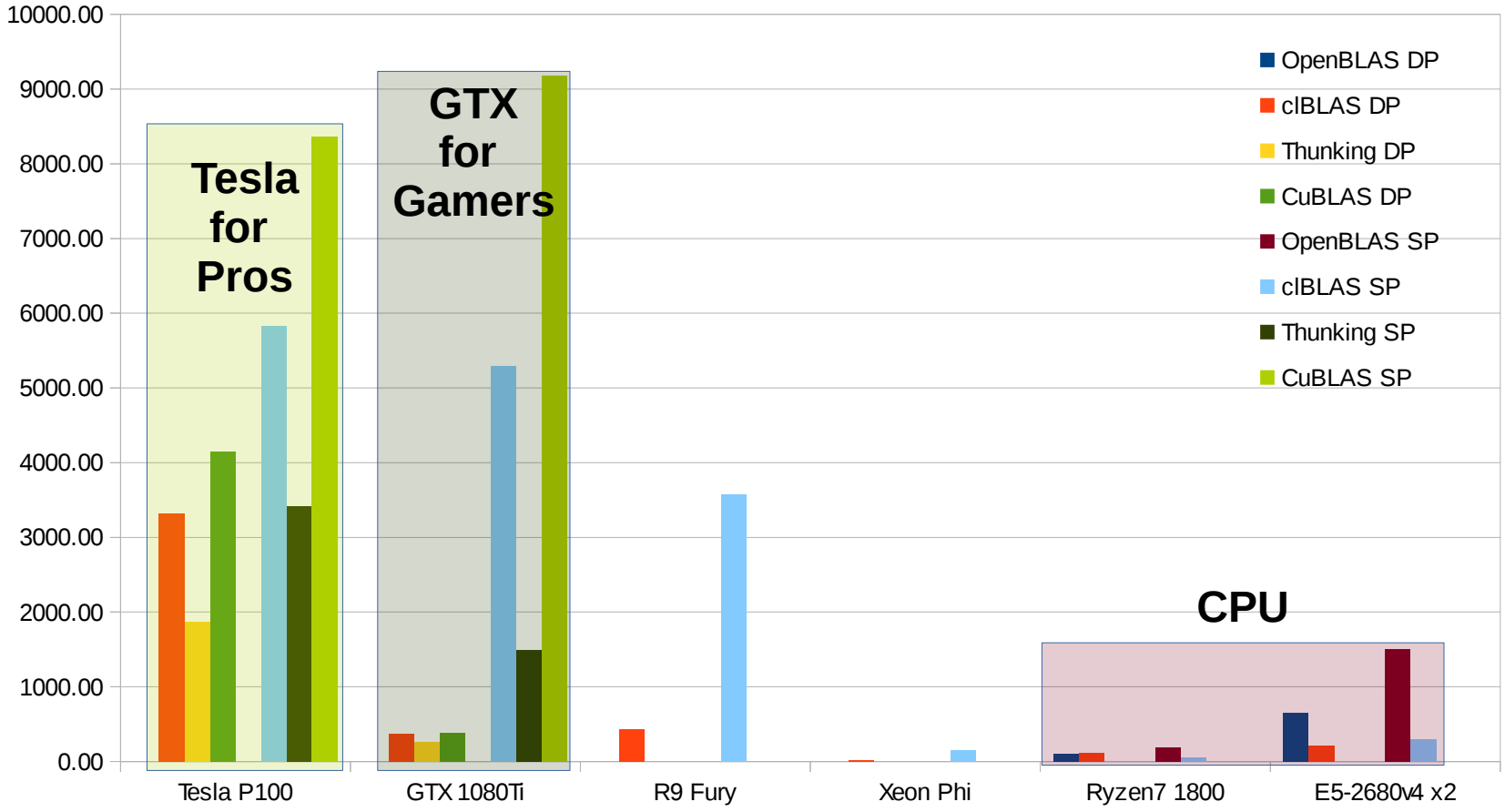
Programmation : 1993

- OpenGL, Glide, Direct3D, ...

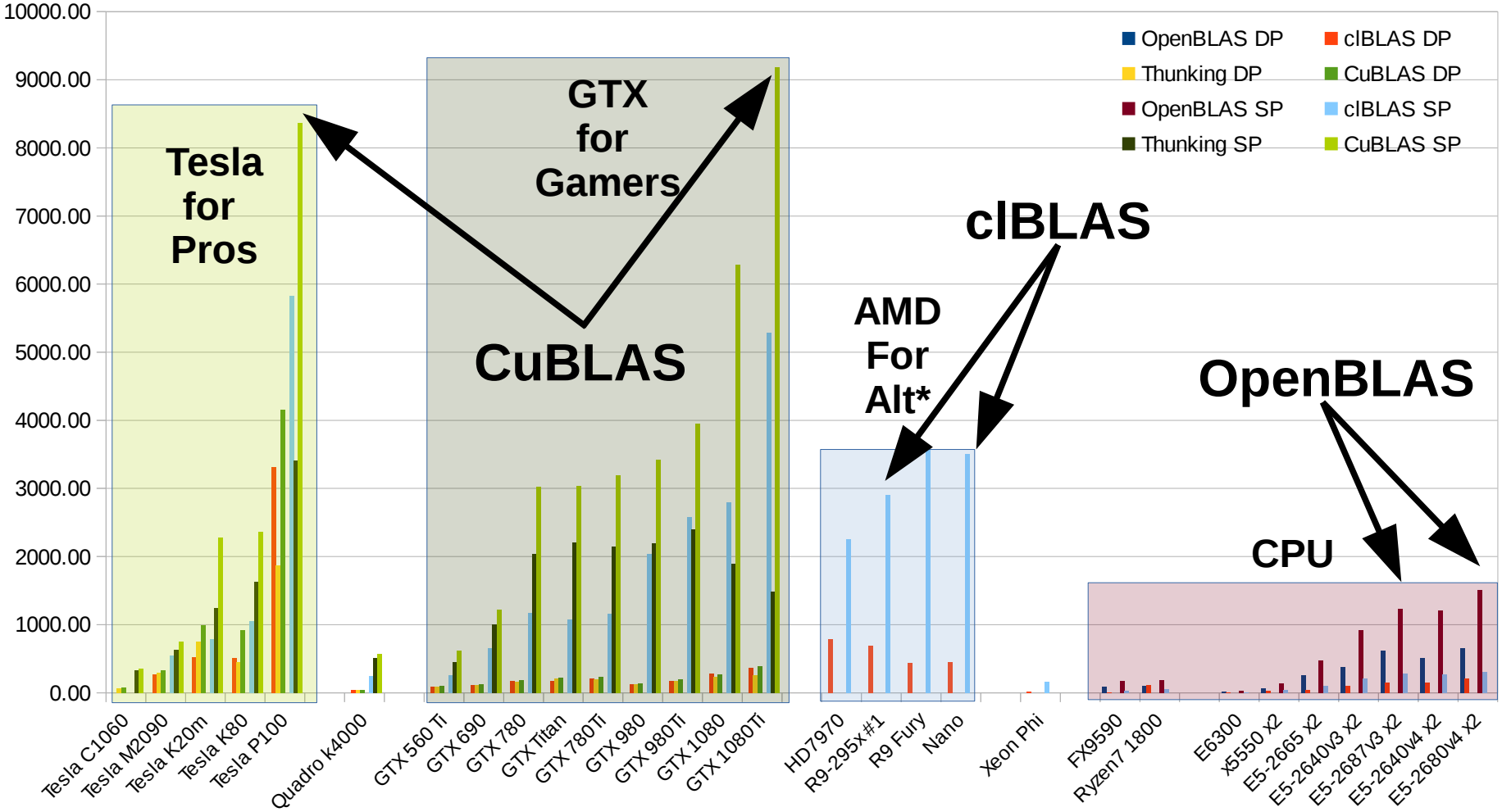
Généricité : 2002

- CgToolkit, CUDA, OpenCL

Est-ce que le GPU est si puissant ? Pour mes meilleures MyriALUs...



BLAS, le match : CPU vs GPU
xGEMM quand x=D(P) or S(P)

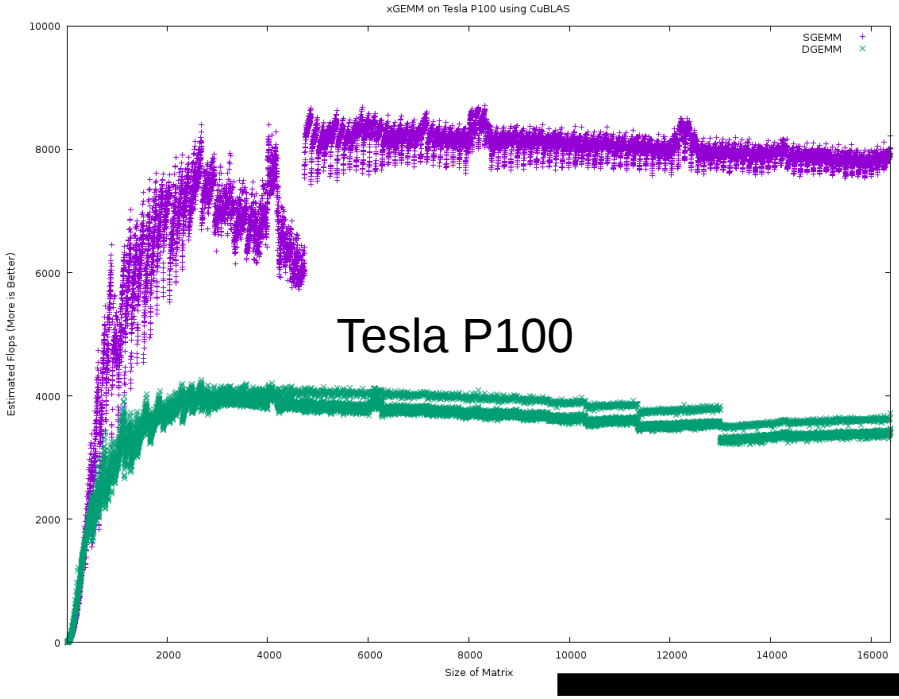


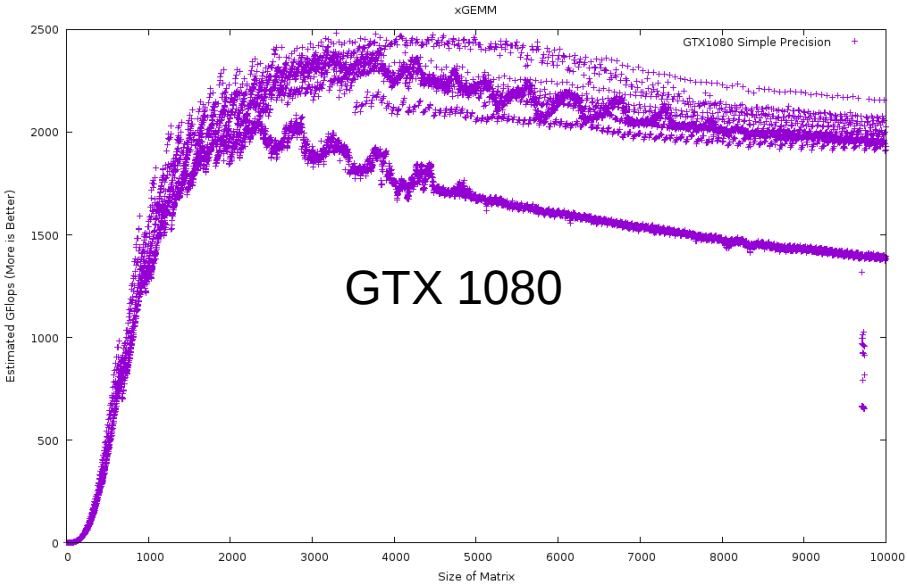
Déjà en 2010, étrange…
Le comportement Tesla C1060...

- Uniquement produit matriciel : xGEMM
- Propriété : Transposé (A * B)=Transposé(A) * Transposé(B)
- Résultats (en Gflops) : Yesss !
  - SP : FBLAS/CBLAS : 12, CuBLAS : 350/327 : x27 !!!
  - DP : FBLAS/CBLAS : 6, CuBLAS : 73/70 : x11 !
- Surprise : Ah !!! CuBLAS préfère les x16 !
  - SP : 16000^2, 350, mais 15999^2 ou 16001^2, 97 : x3,6 !
  - DP : 10000^2, 73, mais 9999^2 ou 10001^2, 31 : x2,35

Ce que Nvidia ne précise jamais…
xGEMM sur des tailles différentes...

Performance





Taille

Oui, il y a la performance, mais dans certaines conditions...

Les modèles de programmation // Petit synoptique...

	Cluster	Nœud CPU	Nœud GPU	Nœud Nvidia	Accélérateur	FPGA
MPI	Oui	Oui	Non	Non	Oui*	Non
PVM	Oui	Oui	Non	Non	Oui*	Non
OpenMP	Non	Oui	Non	Non	Oui*	Non
Pthreads	Non	Oui	Non	Non	Oui*	Non
OpenCL	Non	Oui	Oui	Oui	Oui	Oui
CUDA	Non	Non	Non	Oui	Non	Non
TBB	Non	Oui	Non	Non	Oui*	Non

- OpenCL « semble » dont le plus polyvalent :-)
- CUDA peut SEULEMENT être utilisé avec Nvidia
- Les accélérateurs semblent les plus universels, mais...

Agir en « intégrateur » : Ne pas réinventer la roue...

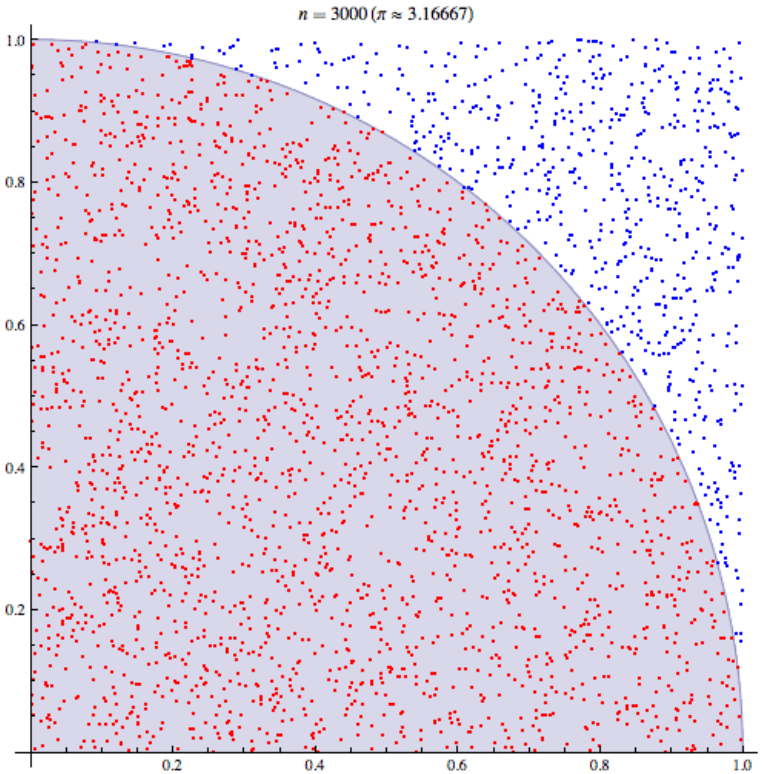
Librairies de programmation parallèle

	Cluster	**Nœud CPU**	Nœud GPU	Nœud Nvidia	Accélerateur
BLAS	BLACS MKL	OpenBLAS MKL	clBLAS	CuBLAS	OpenBLAS MKL
LAPACK	Scalapack MKL	Atlas MKL	clMAGMA	MAGMA	MagmaMIC
FFT	FFTw3	FFTw3	clFFT	CuFFT	FFTw3

- Les librairies classiques utilisables avec les GPU
  - Nvidia fournit plein d'implémentations (notamment BLAS)
  - OpenCL fournit certaines, mais pas complète !

Quel code simple à implémenter en // ?
PiMC : Pi par la méthode de la cible

- Exemple historique de la méthode de Monte Carlo
- Implémentaton parallèle :
  - Distribution équivalente des itérations
- De 2 à 4 paramètres
  - Nombre d'itérations
  - Régime de Parallélisme (Parallel Rate ou PR)
  - (Type de variable : INT32, INT64, FP32, FP64)
  - (RNG : MWC, CONG, SHR3, KISS)
- 2 observables simples :
  - Estimation de Pi (juste indicative, Pi n'étant PAS rationn el :-))
  - Temps écoulé (individuel ou global)

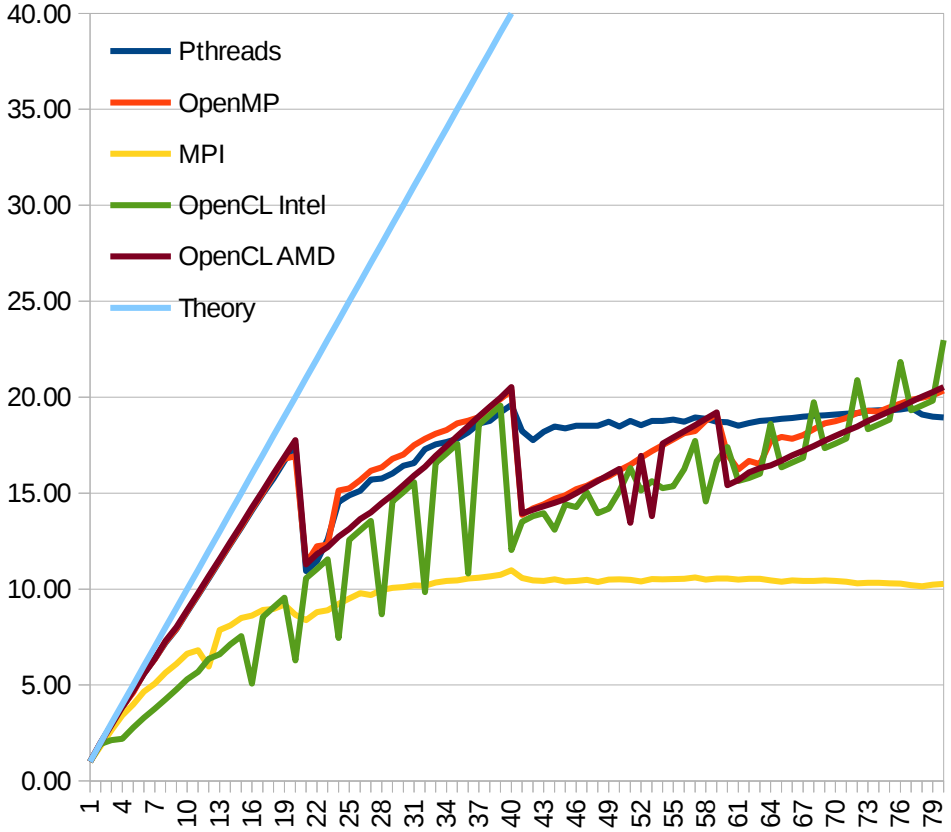


Différences entre CUDA/OpenCL Les noyaux des calculs GPU

```
__device__ ulong MainLoop(ulong iterations,uint seed_w,uint
seed_z,size_t work)
{
  uint jcong=seed_z+work;
  ulong total=0;
  for (ulong i=0;i<iterations;i++) {
      float x=CONGfp ;
      float y=CONGfp ;
    ulong inside=((x*x+y*y) <= THEONE) ? 1:0;
    total+=inside;
  }
__global__ void MainLoopBlocks(ulong *s,ulong iterations,uint seed_w,uint
seed_z)
{
  ulong total=MainLoop(iterations,seed_z,seed_w,blockIdx.x);
  s[blockIdx.x]=total;
  __syncthreads();
```

```
ulong MainLoop(ulong iterations,uint seed_z,uint seed_w,size_t work)
{
  uint jcong=seed_z+work;
  ulong total=0;
  for (ulong i=0;i<iterations;i++) {
      float x=CONGfp ;
      float y=CONGfp ;
    ulong inside=((x*x+y*y) <= THEONE) ? 1:0;
    total+=inside;
  }
  return(total);
}
__kernel void MainLoopGlobal(__global ulong *s,ulong iterations,uint
seed_w,uint seed_z)
{
  ulong total=MainLoop(iterations,seed_z,seed_w,get_global_id(0));
  barrier(CLK_GLOBAL_MEM_FENCE);
  s[get_global_id(0)]=total;
}
```

Des comportements étranges sur un même processeur...



- Processeur 20 cœurs
  - E5-2670v2
- 5 implémentations
  - Dont 2 OpenCL...
- Des similitudes...

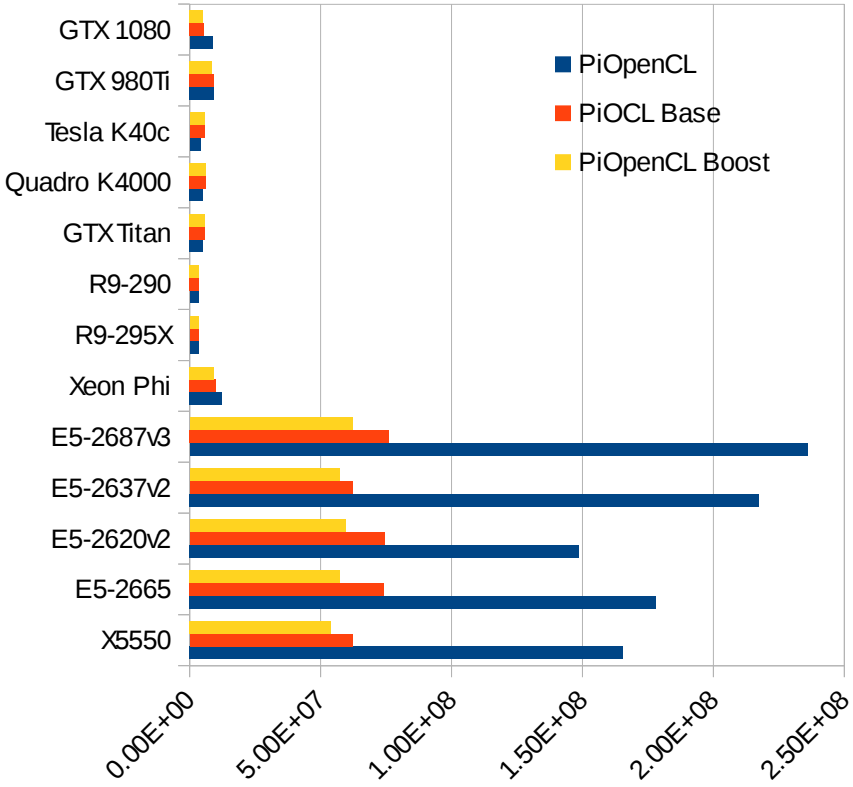
Mais surtout des comportements différents !

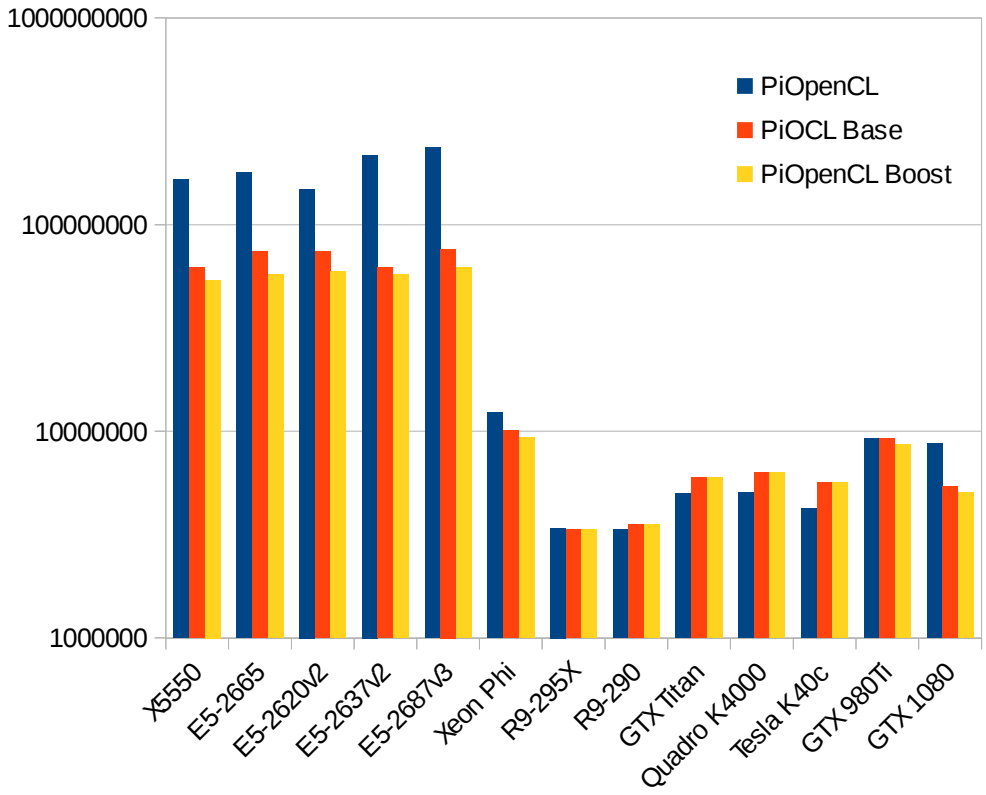
Après le CPU et le GPU, Définition du QPU et EPU

- PR : « Parallel Rate » ou « Régime de Parallélisme »
- CPU : Central Processing Unit
- GPU : Graphical Processing Unit
- QPU : Quantum Processing Unit
  - Pour un « usage », lancement avec PR=1
  - Exemple : lancement pour Threads=1, Blocks=1 ou Work Items=1
- EPU : Equivalent Processing Unit
  - Pour un « usage », optimum de puissance pour un PR donné
  - Exemple : pour un R9-290x, optimum pour 8x le nombre de « shaders »

Avec 1 QPU, à bas « régime »... Le CPU enfonce le GPU !

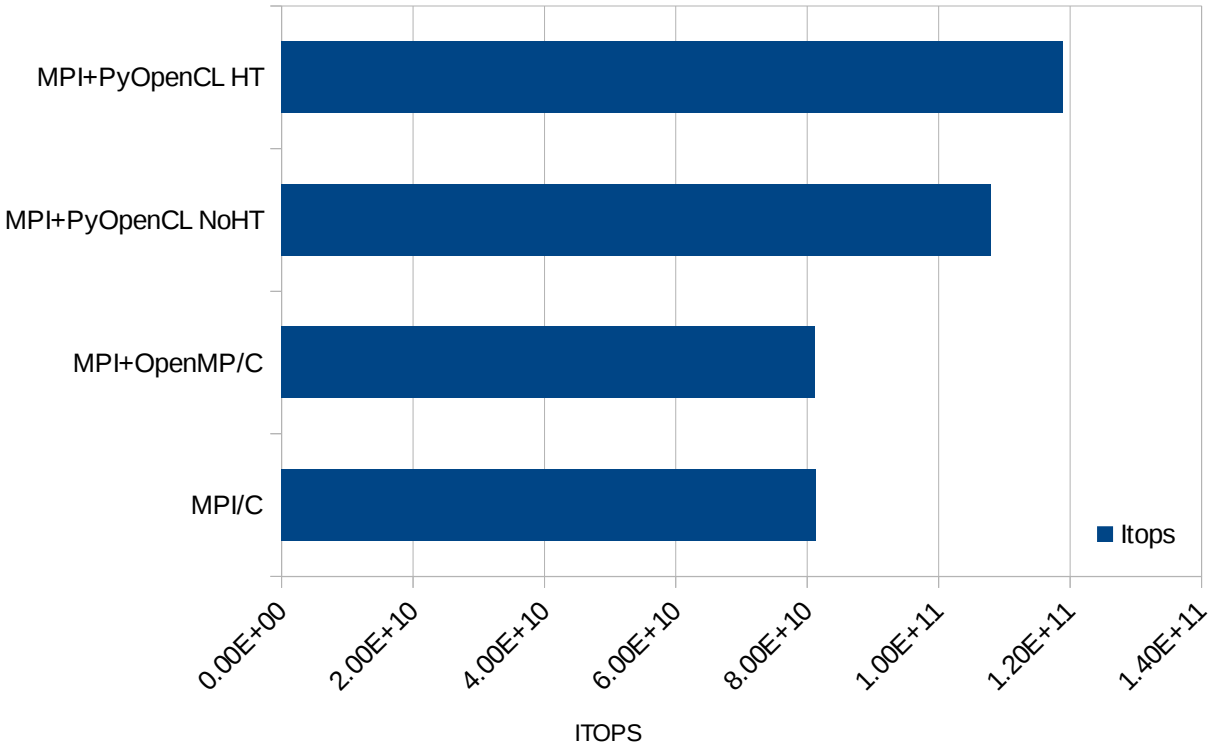
- De 20 à 50x plus lent !





- 1,5 ordre de grandeur...

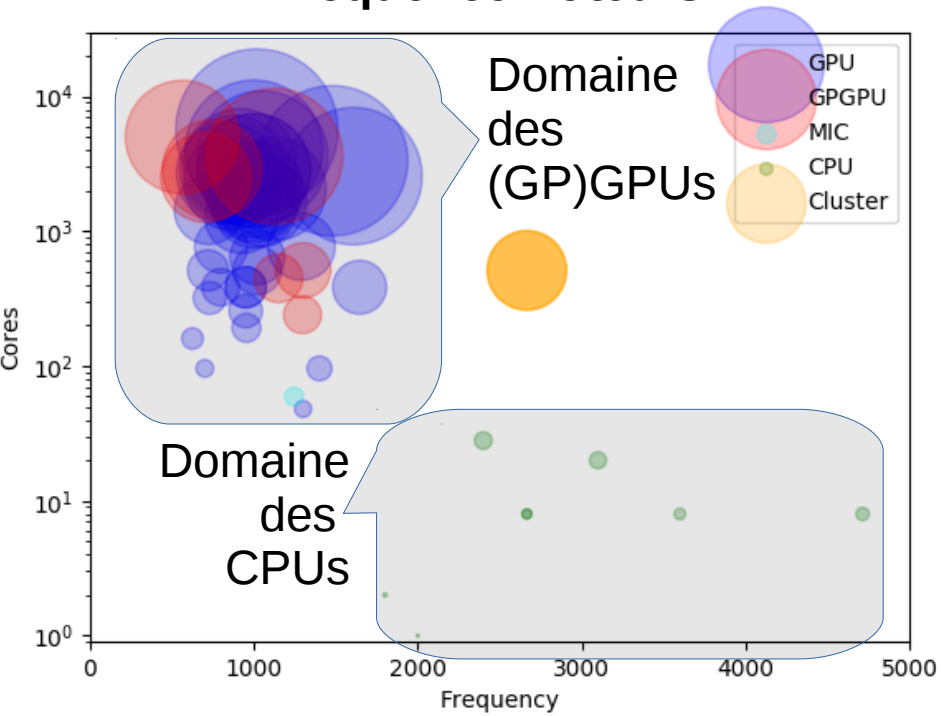
Python est-il efficace ?
Une rapide comparaison avec GNU



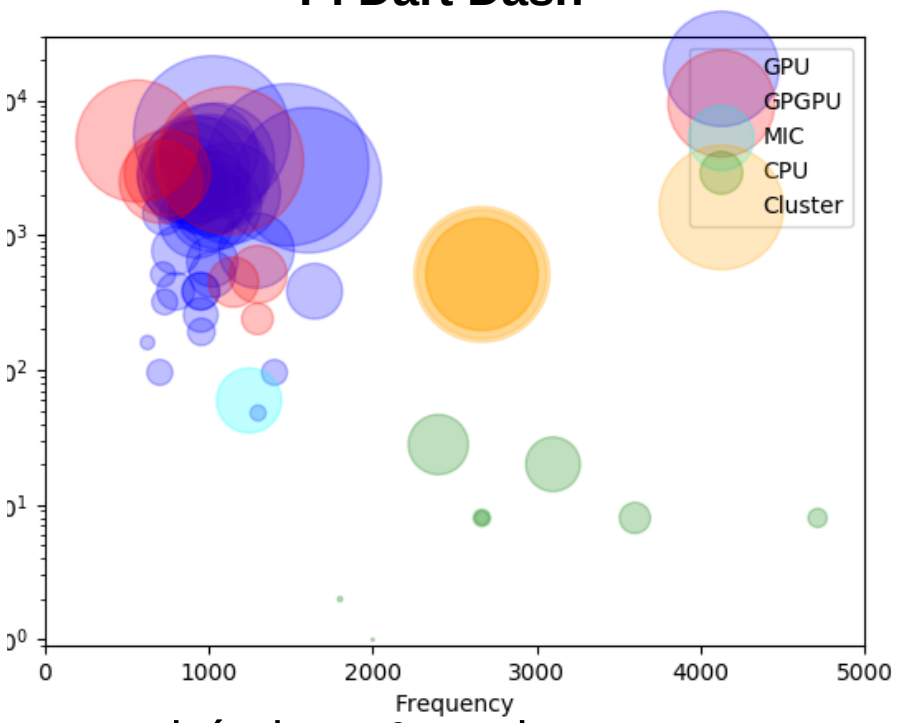
- 3 implémentations (2 hybrides)

Représenter les performances... Question de battement, de cœurs ?

Performance Théorique
Fréquence * cœurs



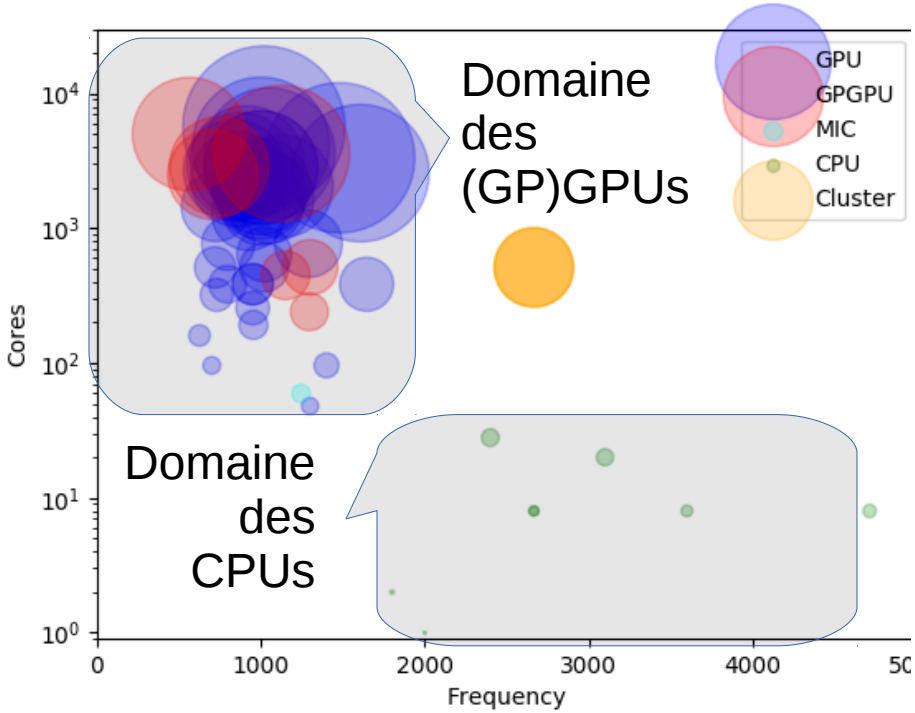
Performance en 32 bits
“Pi Dart Dash”

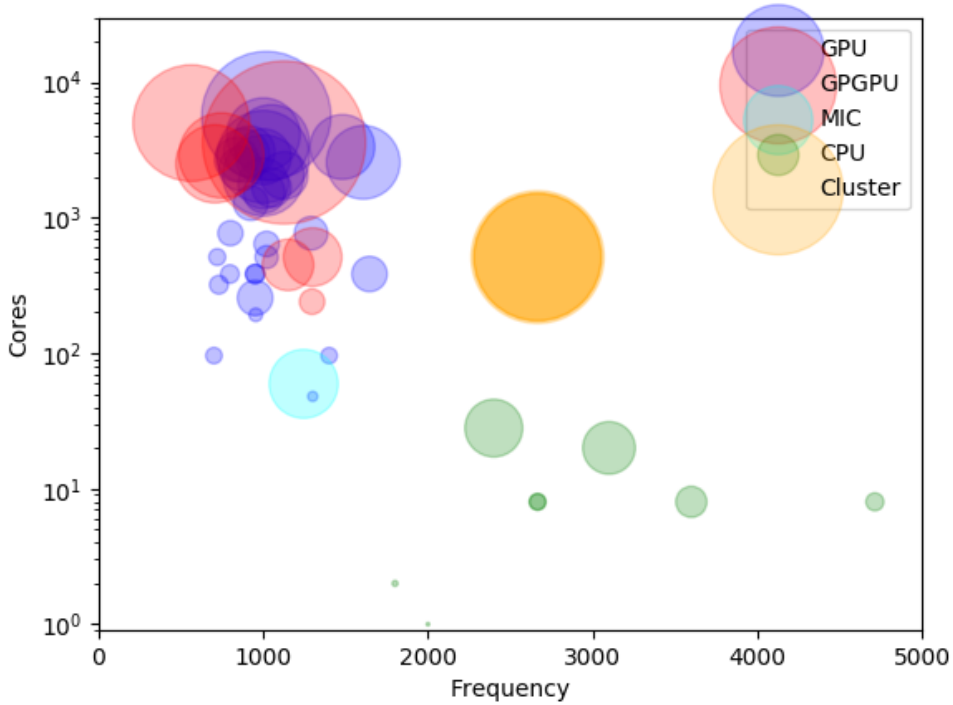


- Sur un GPU, cohérence entre performances théorique & pratique
- Sur un CPU, performance relative meilleure

La performance en informatique

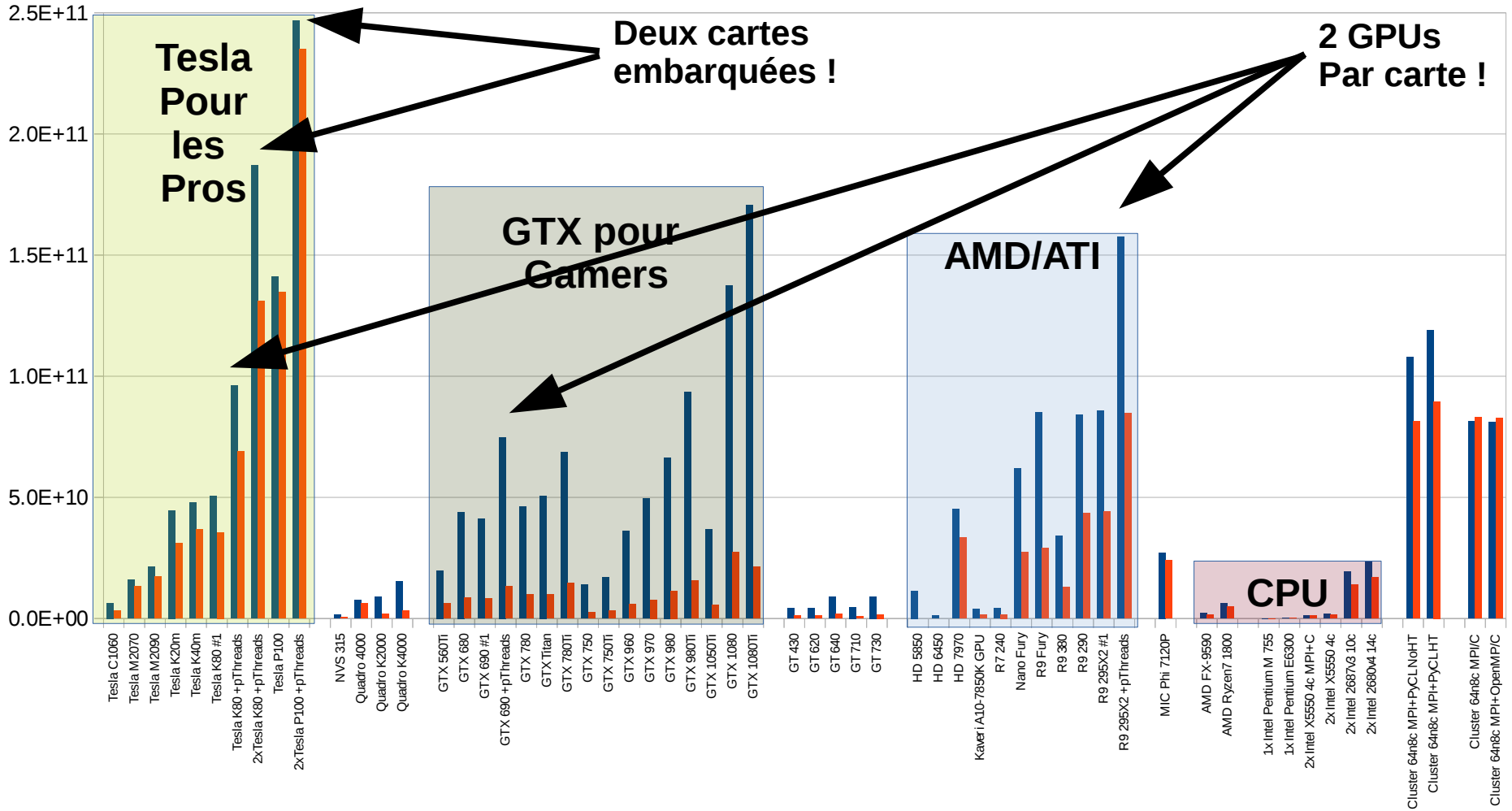
Question de battement, de cœurs, et de bits !



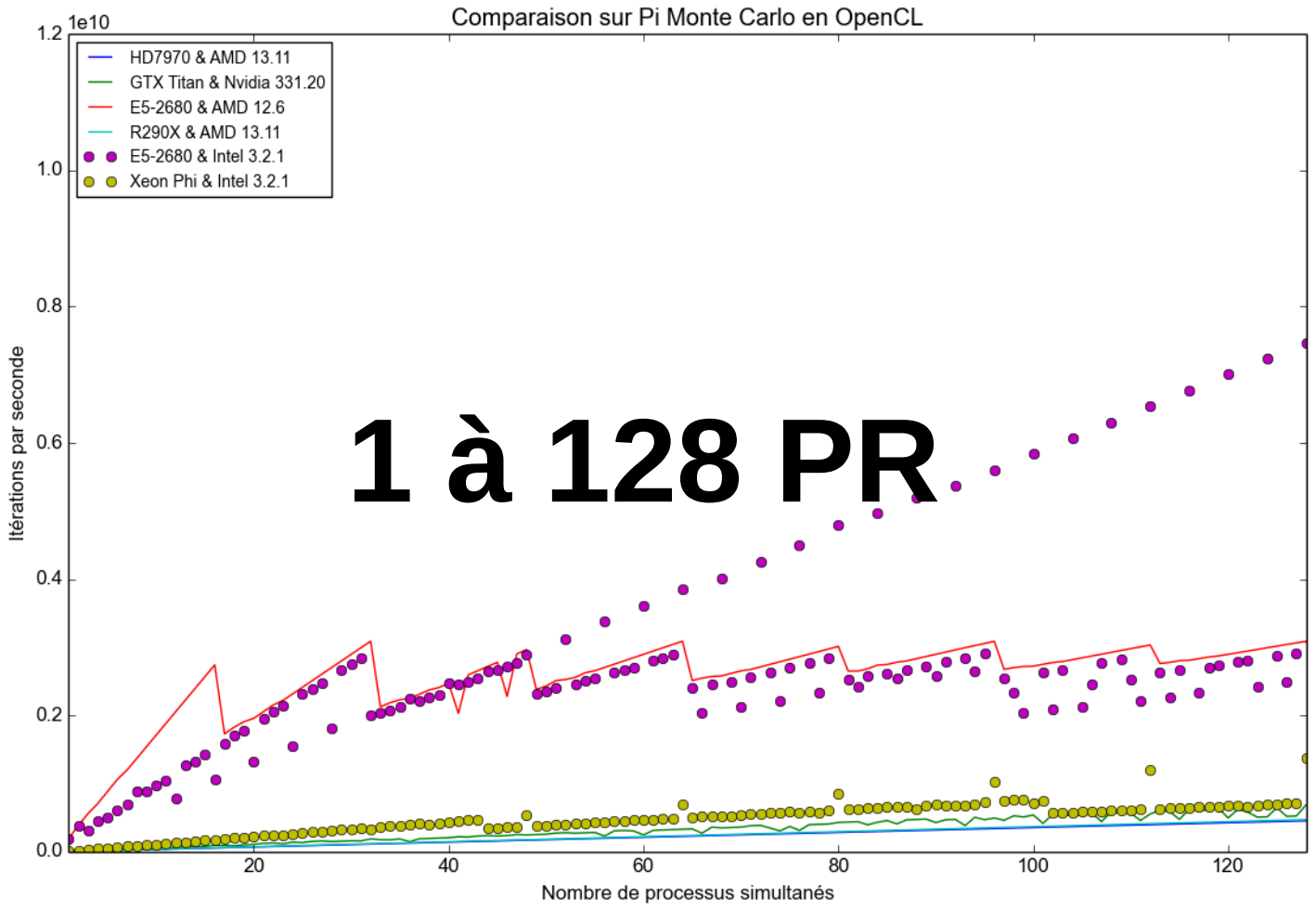


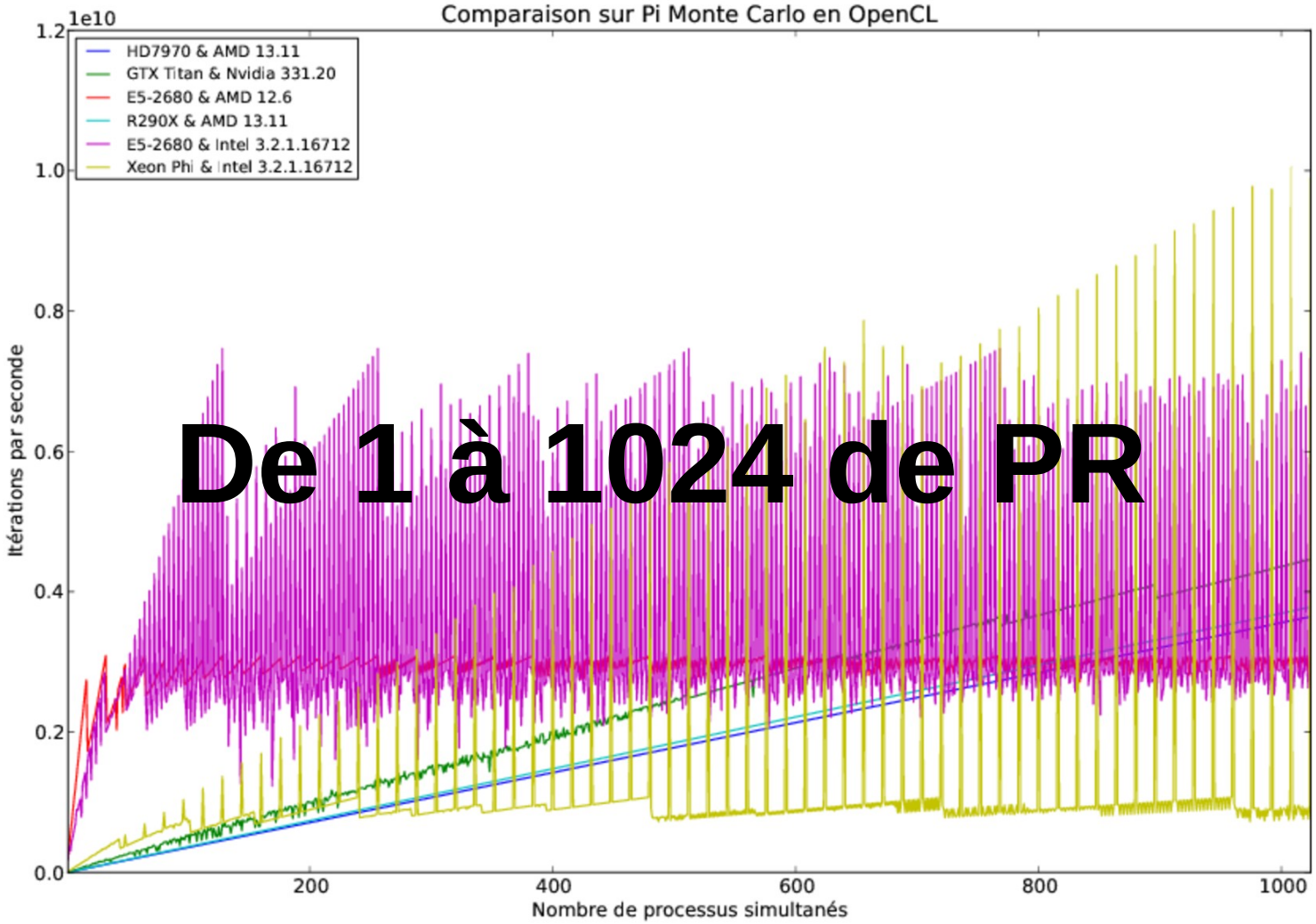
- Sur un GPU, baisse très sensible des performances pour les GPU
- Sur un CPU, performance relative encore meilleure

Le match Pi Monte Carlo Python vs C & CPU vs GPU vs Phi



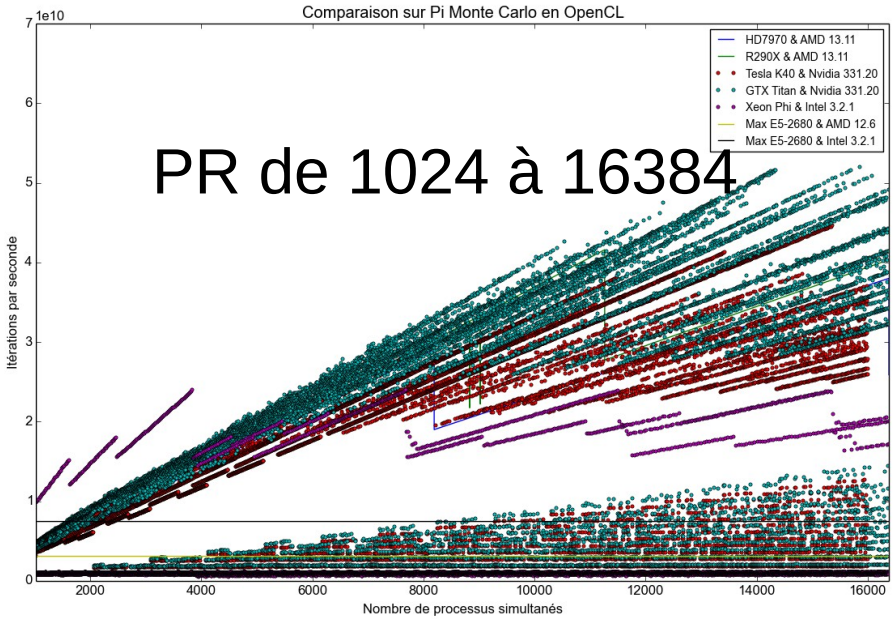
Petite comparaison entre *PU
Bas pararégime : le règne du CPU

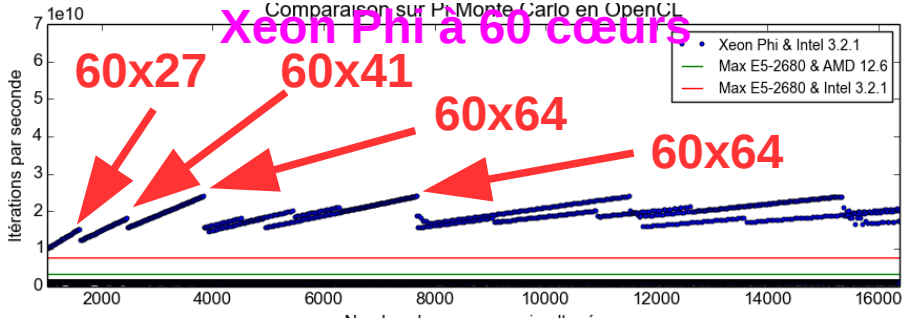


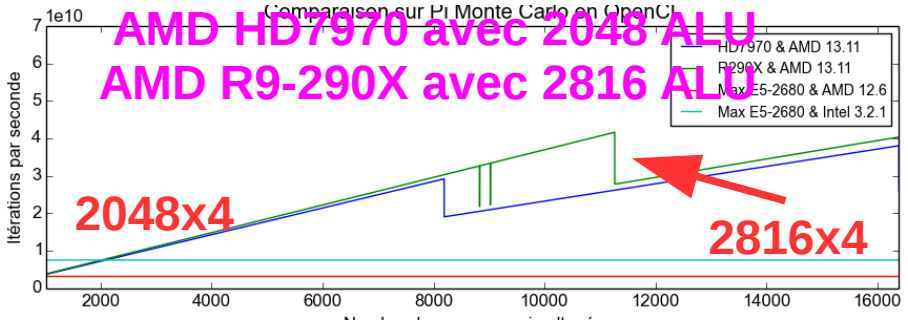


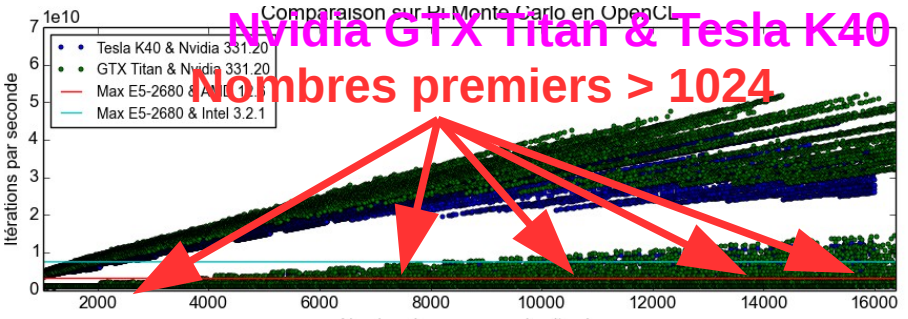
De 1 à 1024 de PR

Exploration profonde de PR ParaRégime de 1024 à 16384

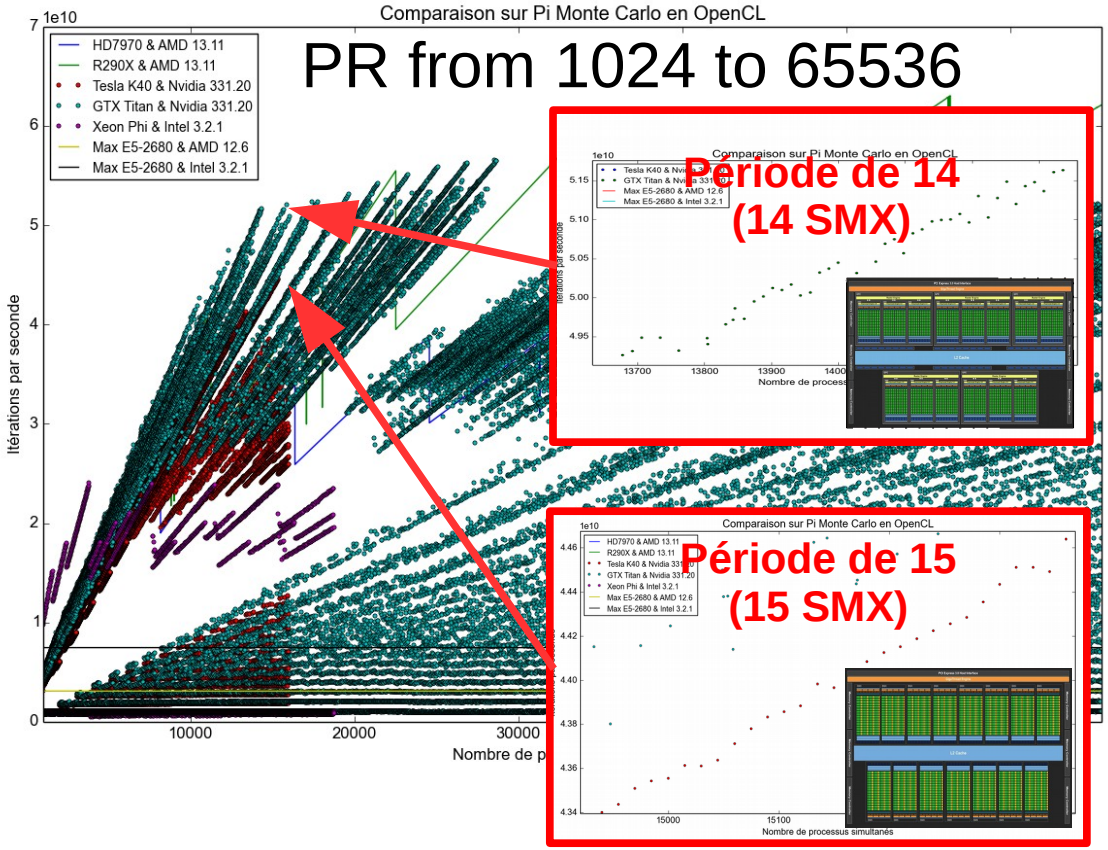




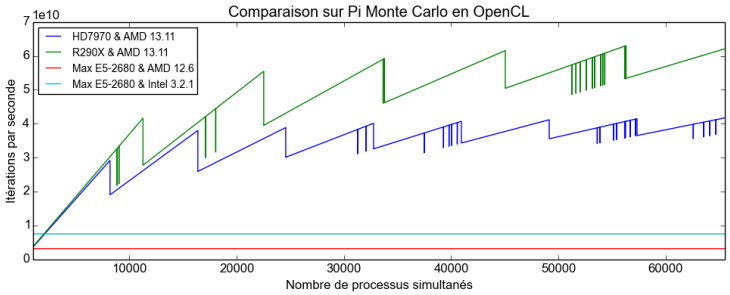




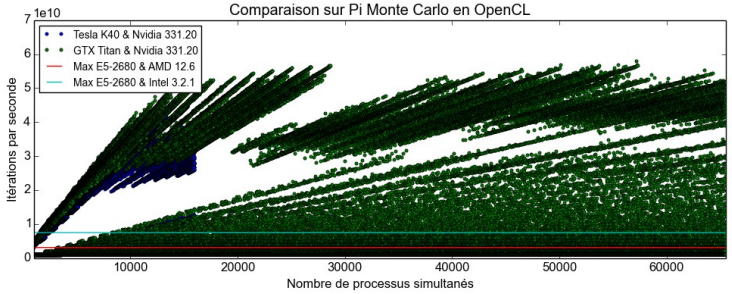
Investiguer la structure interne
Par l'exécution de Pi Dart Dash



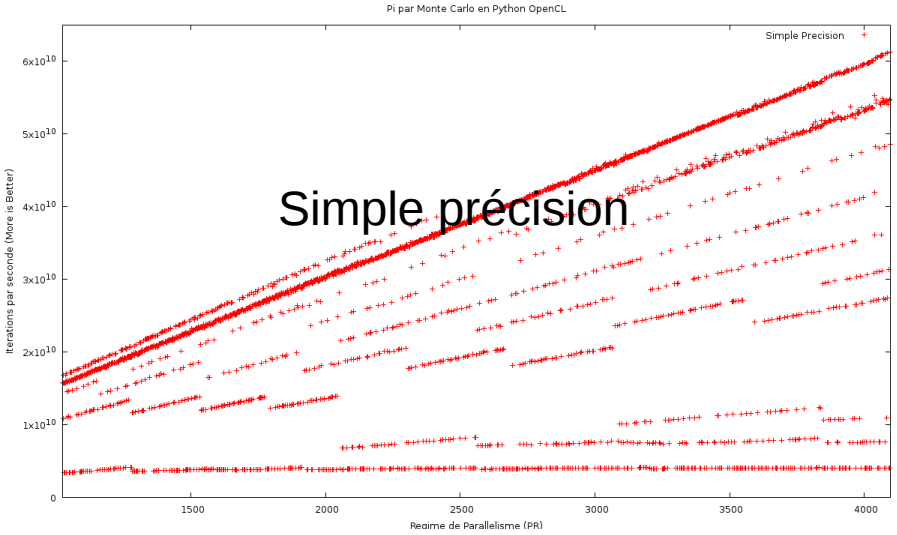
AMD HD7970 & R9-290
Longue Période : 4x nombre d'ALU



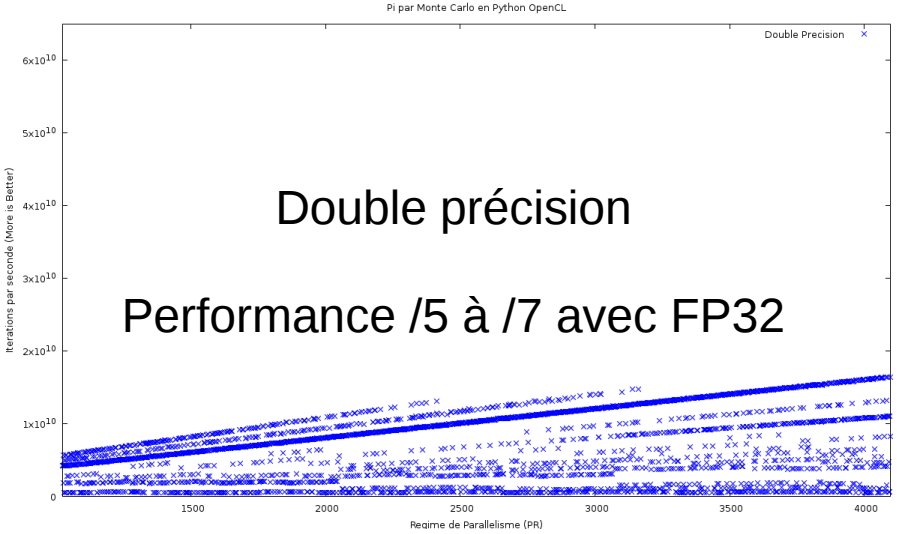
Nvidia GTX Titan & Tesla K40
Courte Période : nombre d'unités SMX



Et les dernières Nvidia GTX1080(Ti) Toujours affectées de bizarreries ?

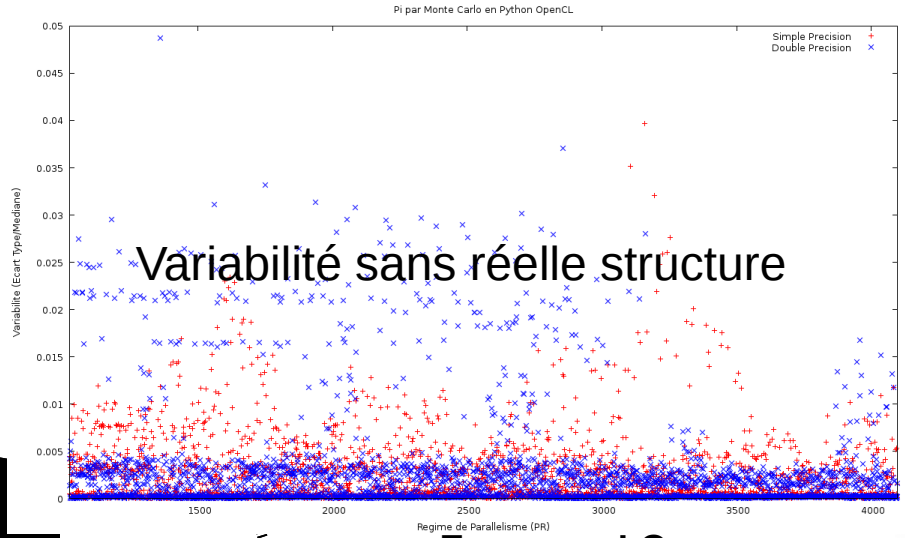


Simple précision

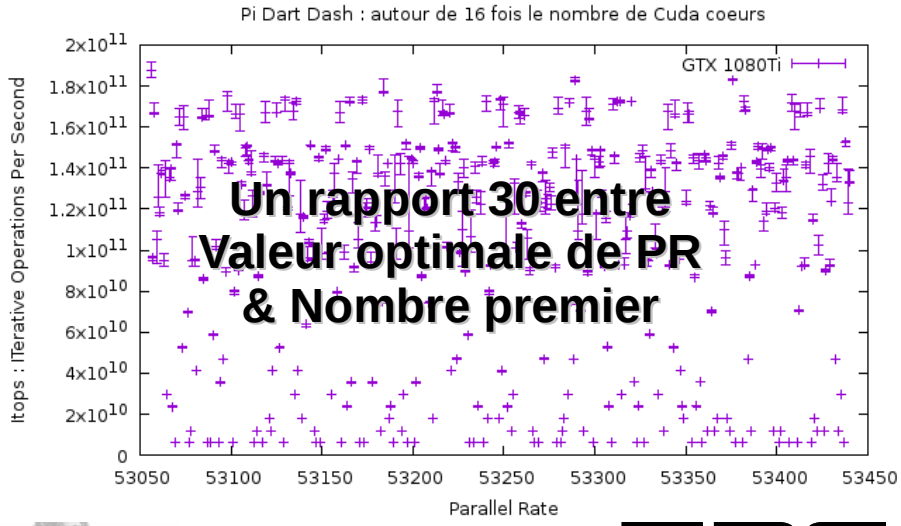


Double précision

Performance /5 à /7 avec FP32

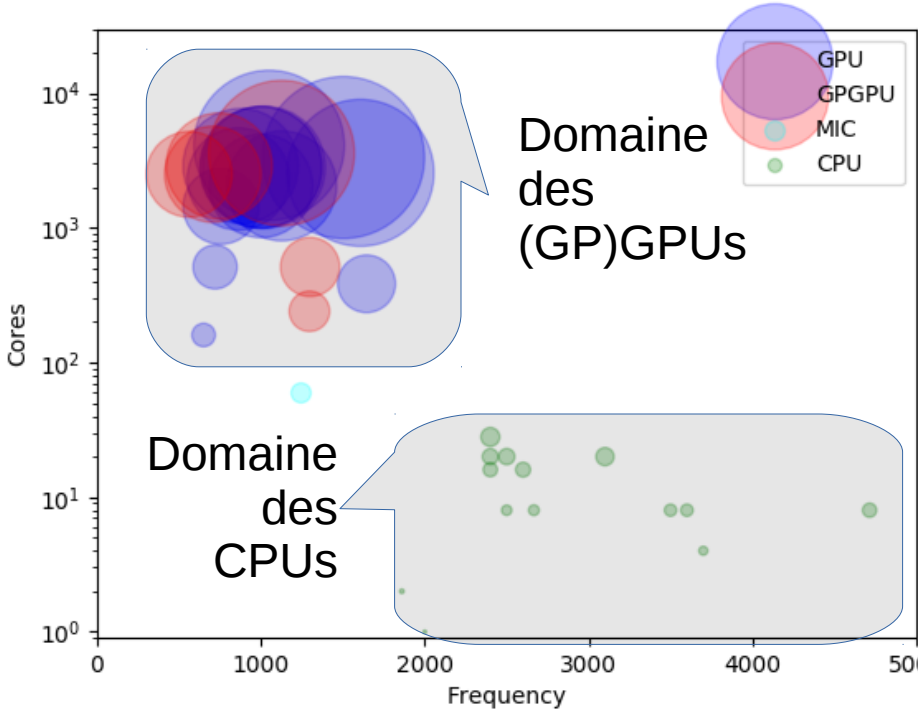


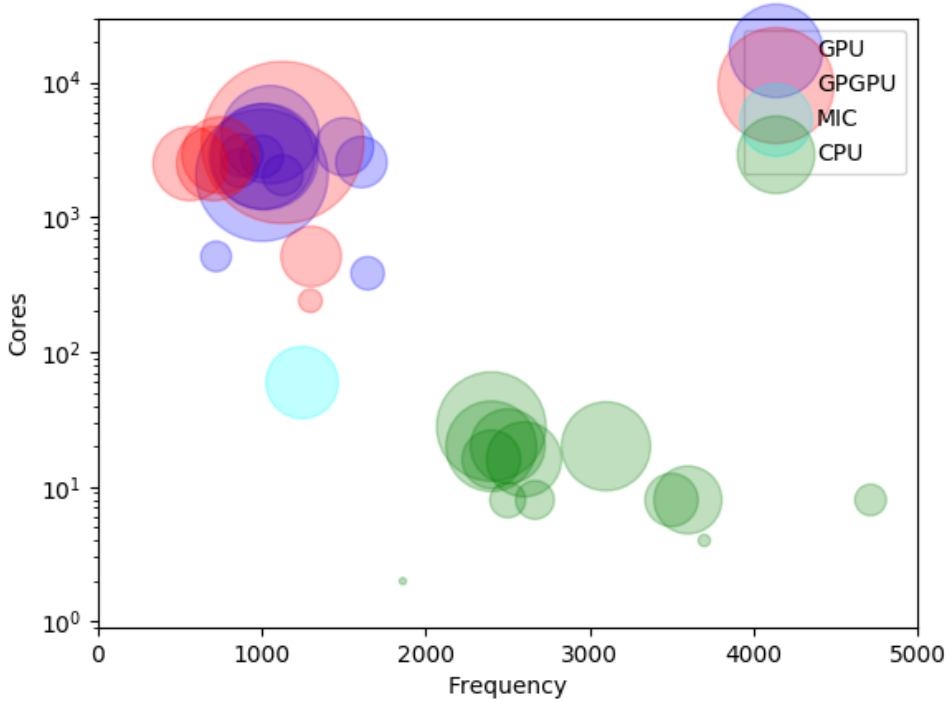
Variabilité sans réelle structure



Un rapport 30 entre Valeur optimale de PR & Nombre premier

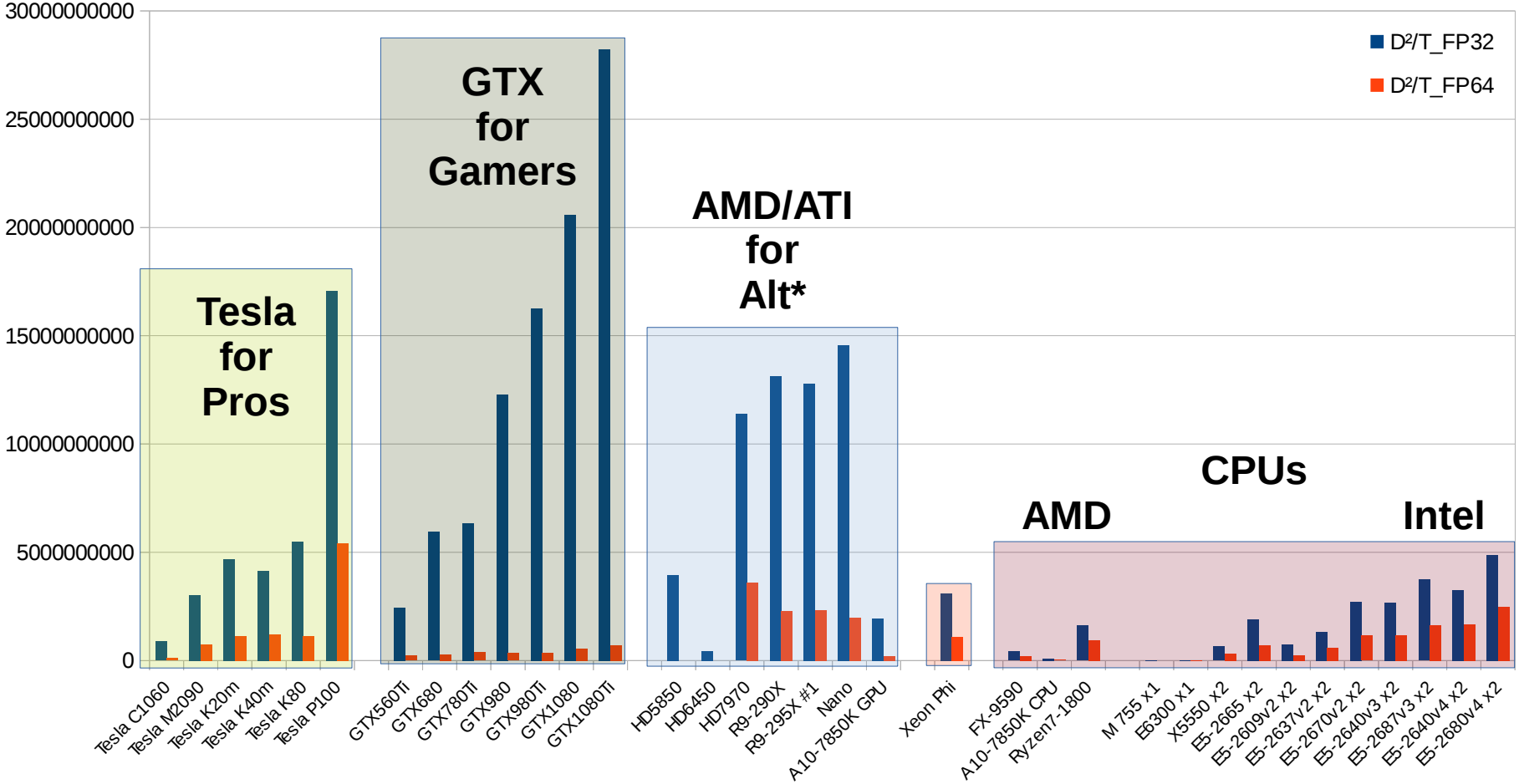
La performance en informatique
Pour un code plus physique...



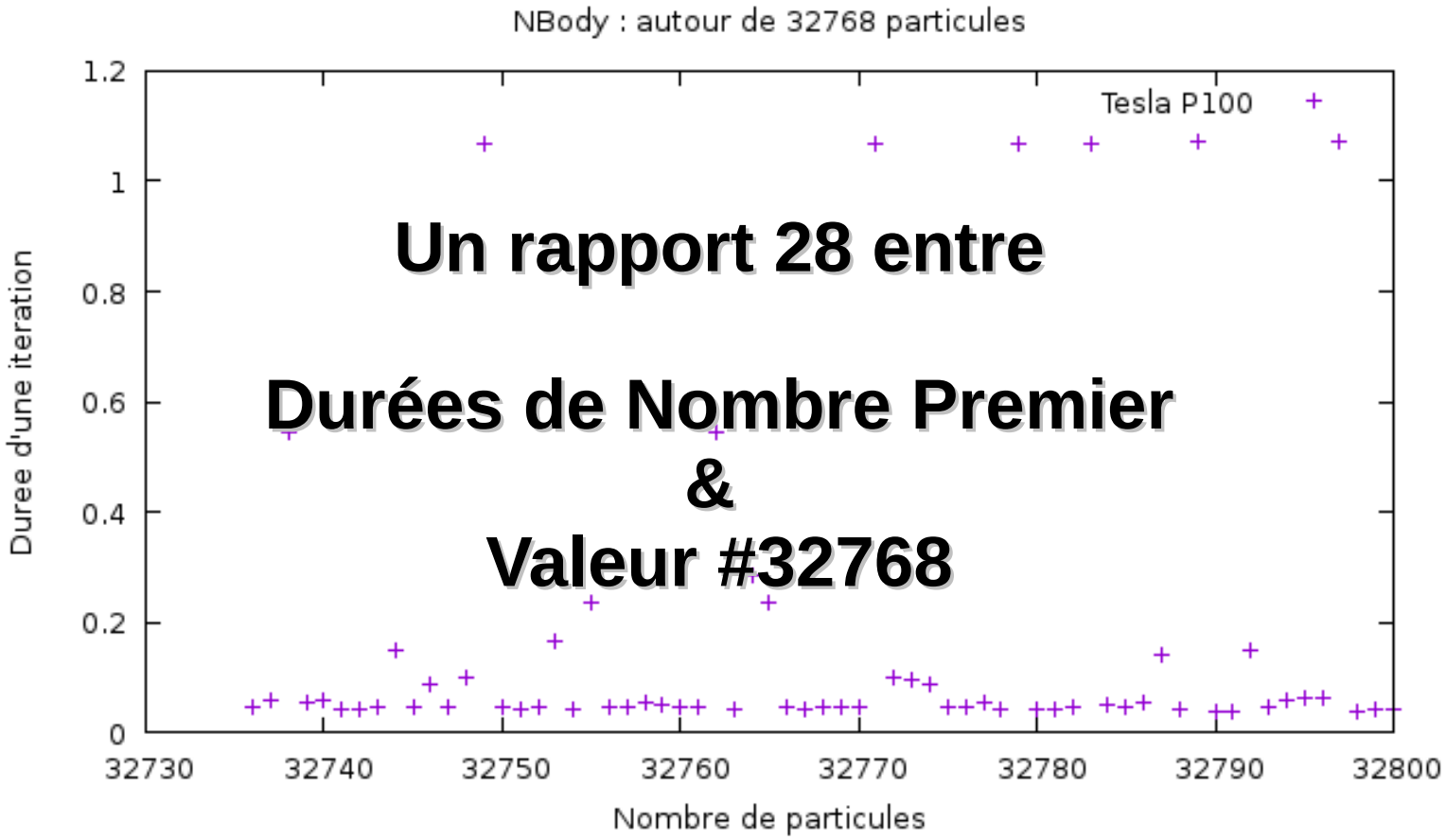


- Sur un GPGPU, performance stable
- Sur un GPU, baisse drastique de la performance (division par 20)
- Sur un CPU, performance relative meilleure

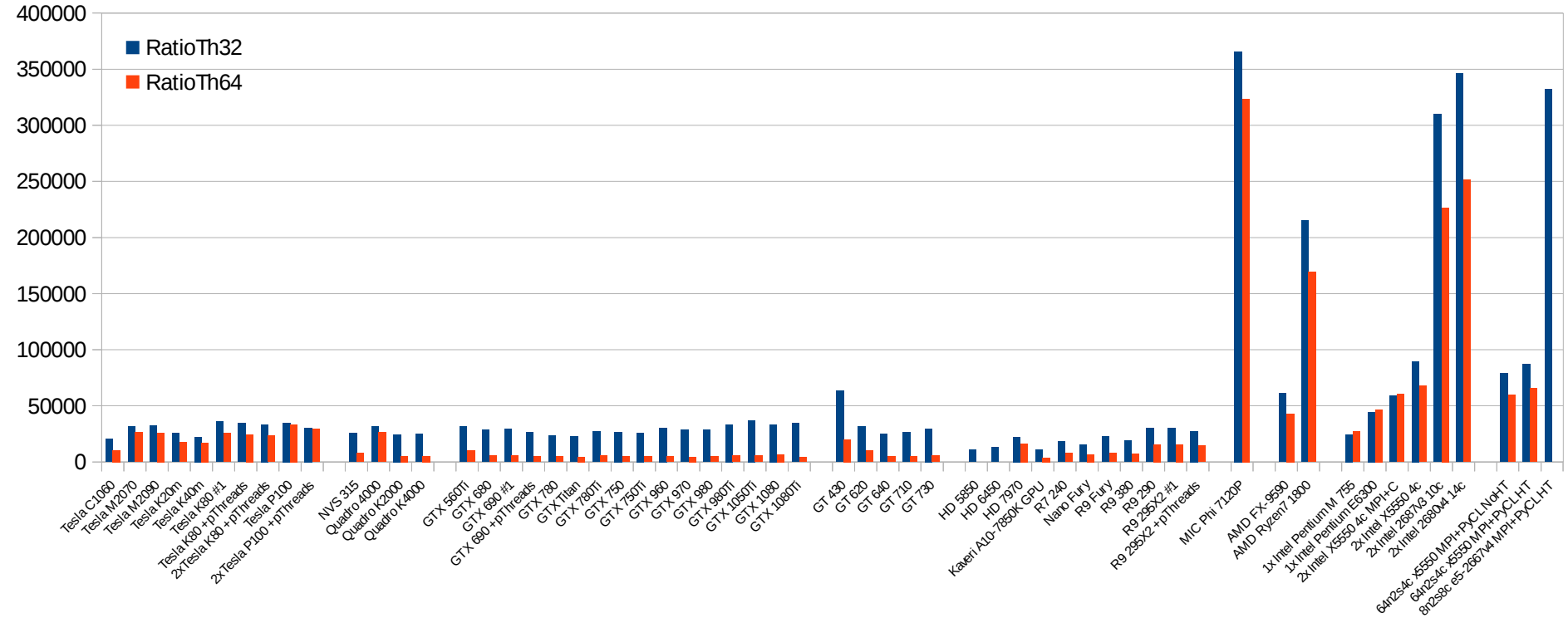
Modèle N-corps « naïf »
Une comparaison classique...



Et les effets « prime numbers » pour les cartes Nvidia ?

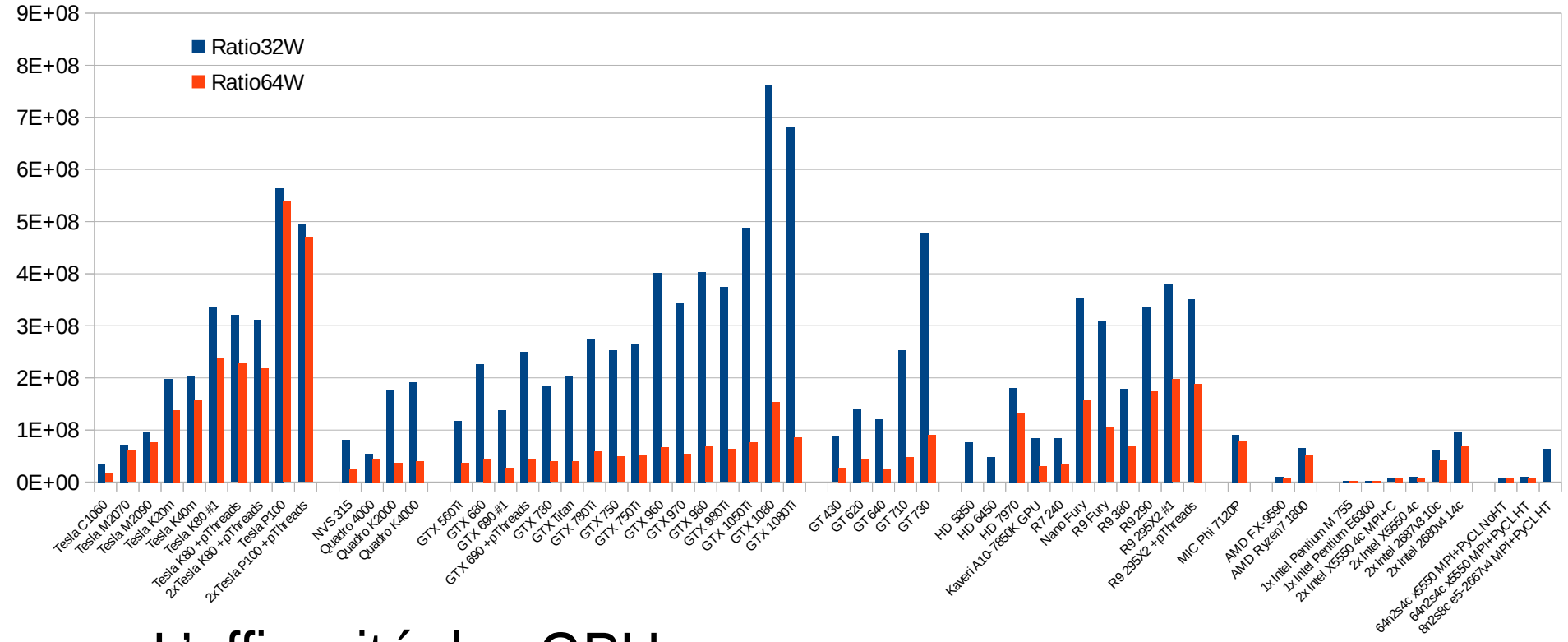


Performances normalisées : quel ratio ?
Pi Dart Dash / Fréquence*Nb cœurs



- Pas d'évolution significative pour les (GP)GPUs
- Ratio largement croissant pour les CPU : Intel & AMD

Performances par Watt
Pi Dart Dash / TDP



- L'efficacité des GPU
- La progression des CPU

Mais à quoi ça sert tout ça ? Caractériser & éviter les regrets...

- Ce qu'il faut retenir :
  - Dans une machine, en 2017, la puissance « brute » est dans le (gros) GPU
  - Pour un régime de parallélisme bas, le CPU enfonce le GPU
  - Un GPU n'est supérieur au CPU que pour PR > 1000
  - Le GPU n'atteint son optimum QUE pour certains PR
  - Sans une caractérisation, des déceptions à prévoir…
  - OpenCL est un bon compromis comme langage *PU
  - Python reste l'approche la plus rapide de OpenCL
- Ce qu'il faut faire : expérimenter !