

INSA 2023

# Des architectures parallèles Aux programmes parallélisés

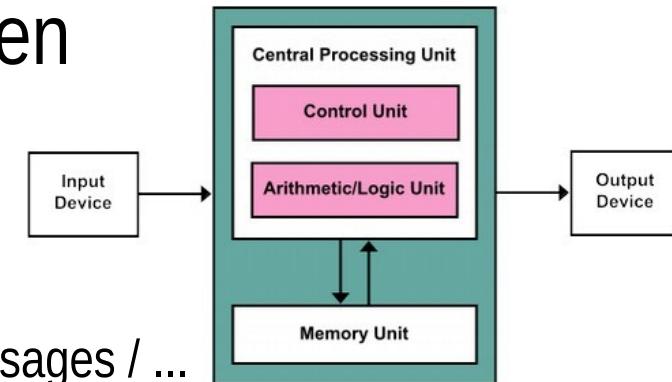
De la métrologie  
À la mesure de la performance

Loi de Brooks :  
*“Ne demandez pas à 9 femmes de faire en 1 mois  
ce qu'une peut faire en 9...”*

Emmanuel Quémener

# Quelle vue du contexte ? Point de vue d'un physicien...

- Approche « système » du physicien
  - Héritage des calculateurs analogiques
  - Le « système » informatique :
    - Réseau / matériel / OS / librairies / codes / usages / ...
- Approche « Saint Thomas »
  - Apprentissage inductif par ma seule mesure
- Approche « pilote d'essai »
  - Caractérisation, recherche d'une exploitation optimale...



# Précautions pour cette présentation

## Ce que cela ne sera pas !

- Une introduction générale au parallélisme
  - Les cours organisés par Lyon Calcul
  - [https://computing.llnl.gov/tutorials/parallel\\_comp/](https://computing.llnl.gov/tutorials/parallel_comp/)
- Une introduction aux langages de parallélisation
  - MPI : <https://computing.llnl.gov/tutorials/mpi/>
  - Posix Threads : <https://computing.llnl.gov/tutorials/pthreads/>
  - OpenMP : <https://computing.llnl.gov/tutorials/openMP/>
  - C'est là que j'ai pas mal appris !
- Je veux partager ce qui n'est JAMAIS dit (ou peu...)

# Centre Blaise Pascal ~ Dryden FR

## Un petit exemple illustratif

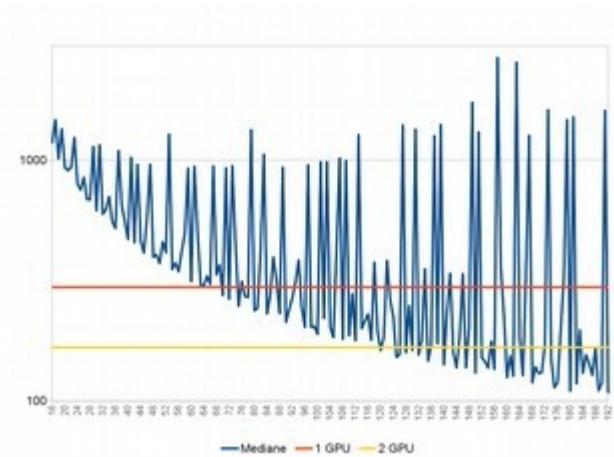
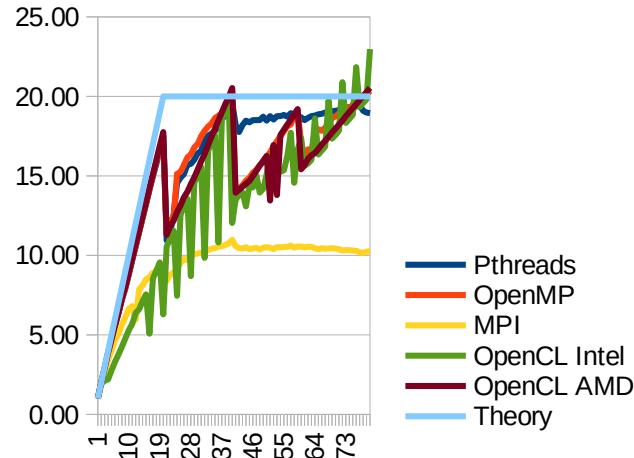
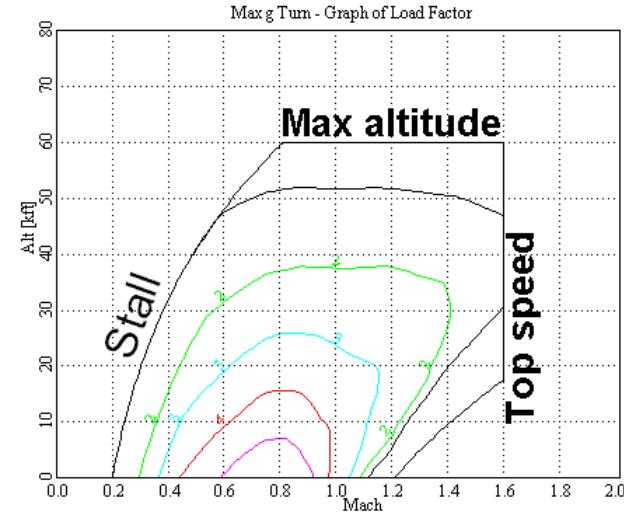
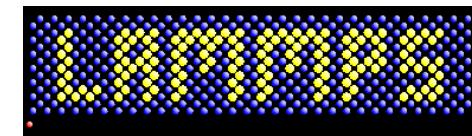


Dryden Flight Research Center EC87 0182-14 Photographed 1987  
X-29

- Nasa X-29
- Cellule de F-5
- Moteur de F-18
- Train de F-16
- Etudes
  - Plans « canard »
  - Incidence  $>50^\circ$
  - « Fly-By-Wire »

Recycler, réutiliser, explorer de nouveaux domaines...

# Le but : de l'enveloppe de vol... ... aux enveloppes de parallélisme



# De la démarche scientifique... ... À l'expérience numérique

Even a baby  
knows the  
**scientific method!**



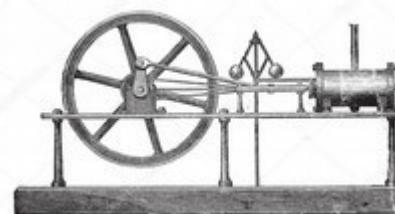
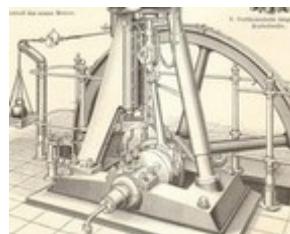
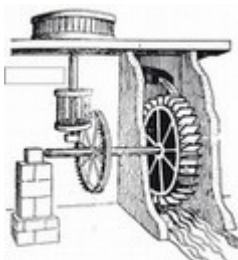
# Quelques définitions & sigles...

- ALU : Arithmetic & Logic Unit, Unité Arithmétique et Logique
- CPU : Central Processing Unit, Unité de traitement Centrale,
- Flops : Floating Point Operations Per Second, Opérations flottantes par seconde
- (GP)GPU : (General Purpose) Graphical Processing Unit, Circuit graphique
- MPI : Message Passing Interface, Interface de communication par messages
- RAM : Random Access Memory, Mémoire à accès aléatoire
- SMP : Shared Memory Processors, Processeurs à mémoire partagée
- TDP : Thermal Design Power, Enveloppe thermique
- Et quelques autres :
  - PR : Parallel Rate, taux de parallélisme (NP en MPI, Threads en OpenMP, Blocks, WorkItems en GPU)
  - Itops : Iterative Operations Per Second
  - QPU : Quantum Processing Unit (program exécuté avec PR=1)
  - EPU : Equivalent Processing Unit (PR déduit de l'optimal d'exécution du programme parallèle)

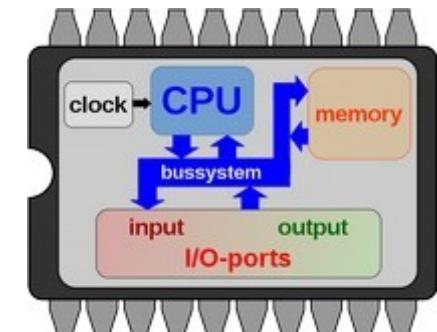
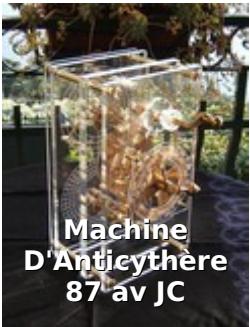
# Travaux physique ou intellectuel

## Moteurs & Ordinateurs :

### Des auxiliaires de « puissance »

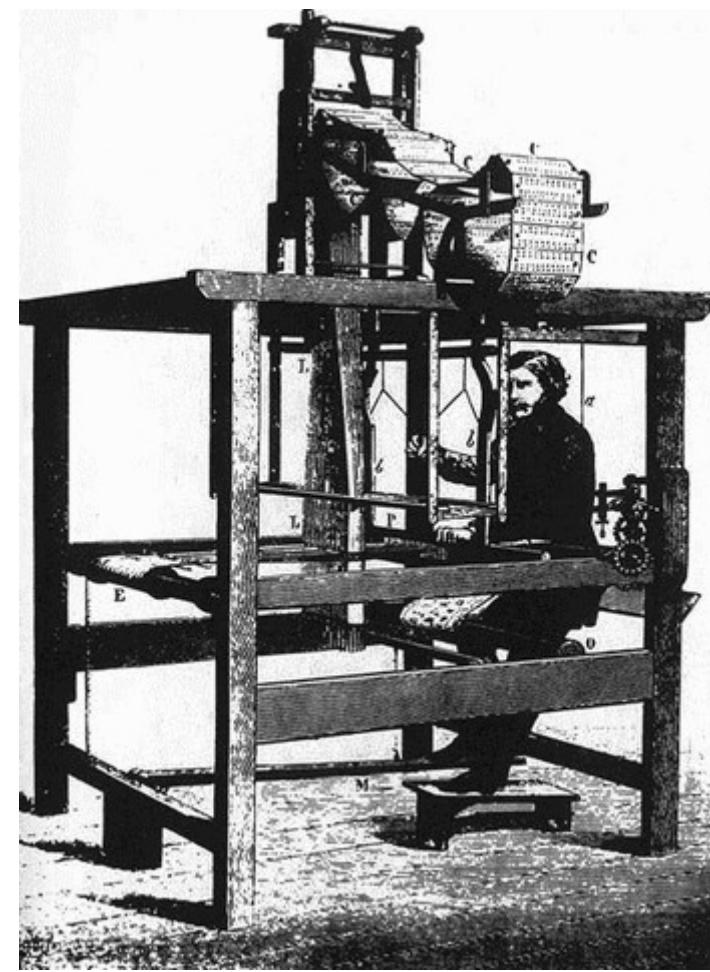
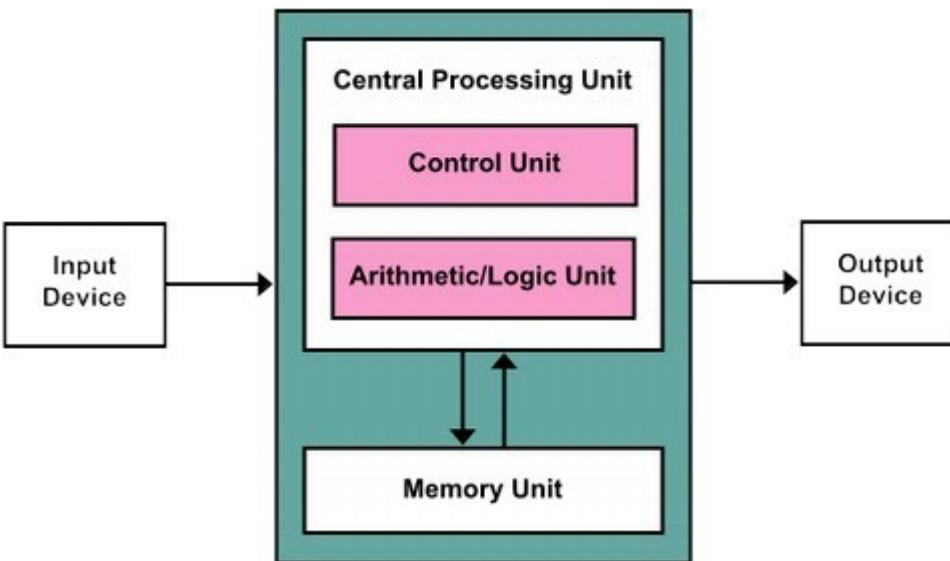


Des anciens temps aux temps modernes...



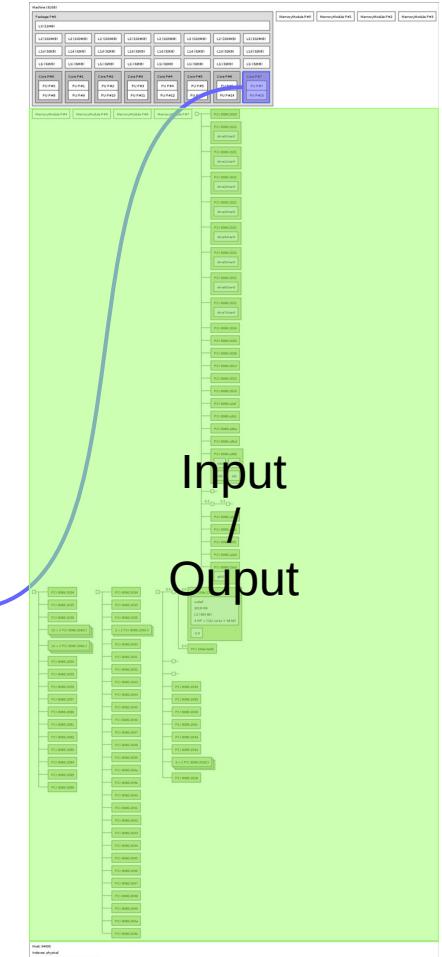
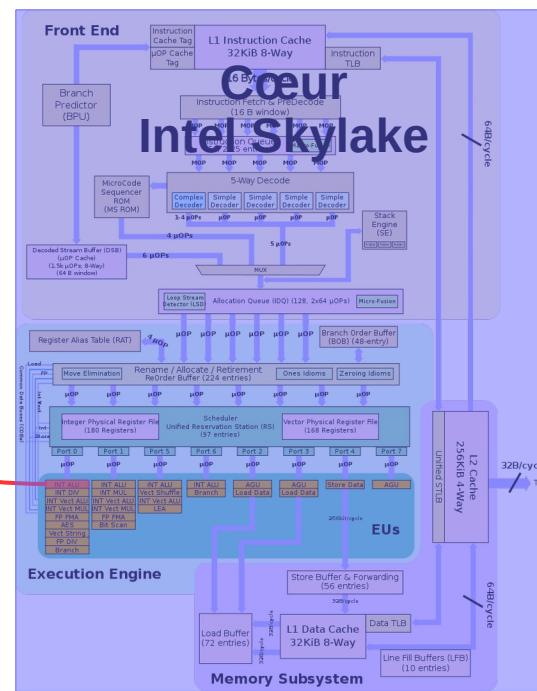
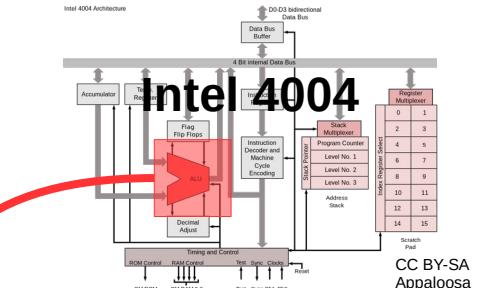
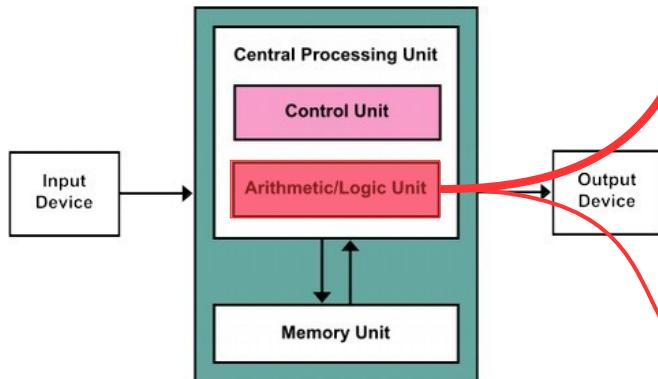
# Un ordinateur : un métier à tisser ?

## Architecture de Von Neumann

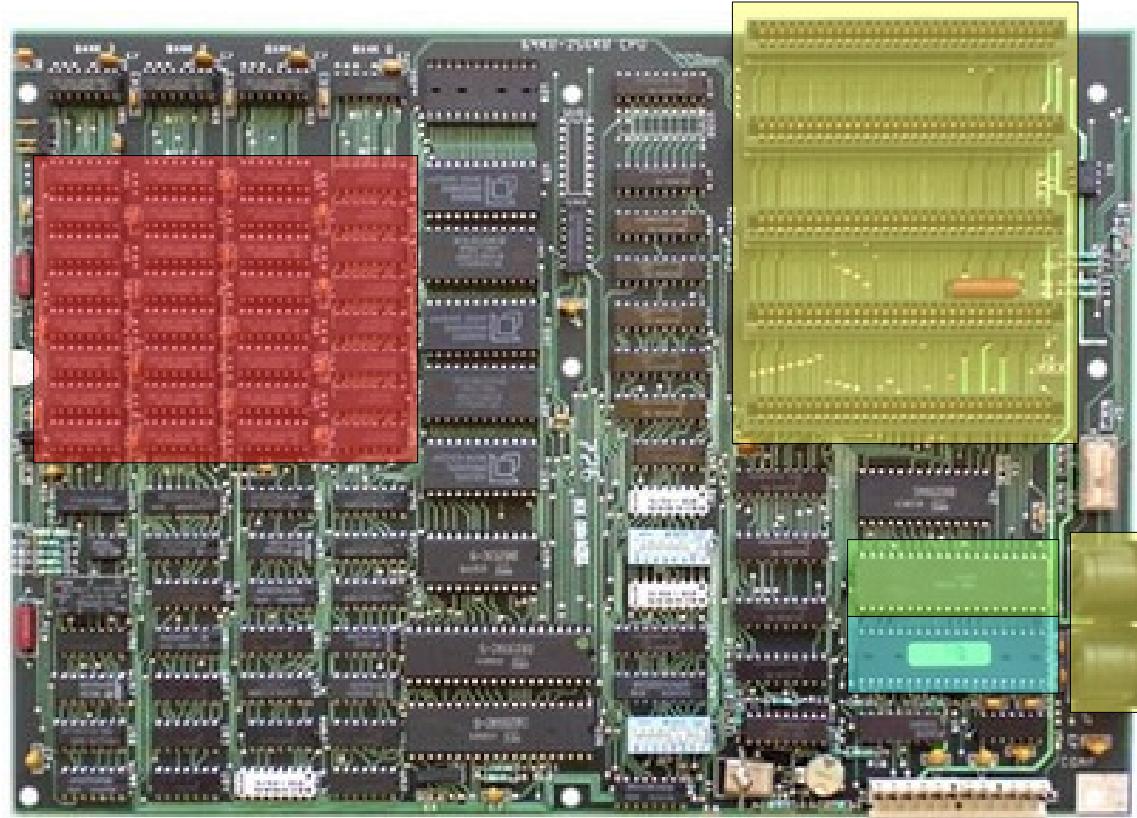
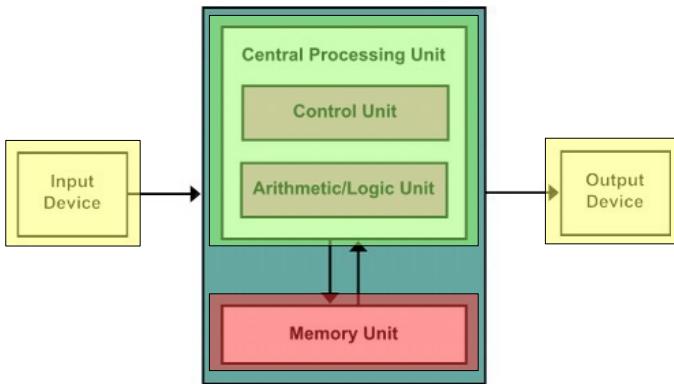


# Architecture de Von Newman

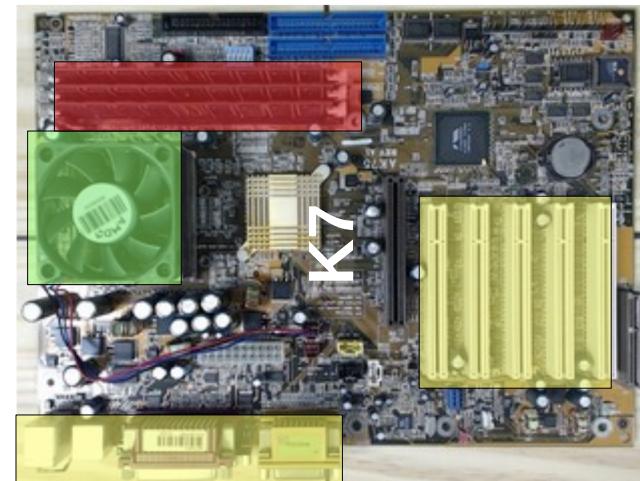
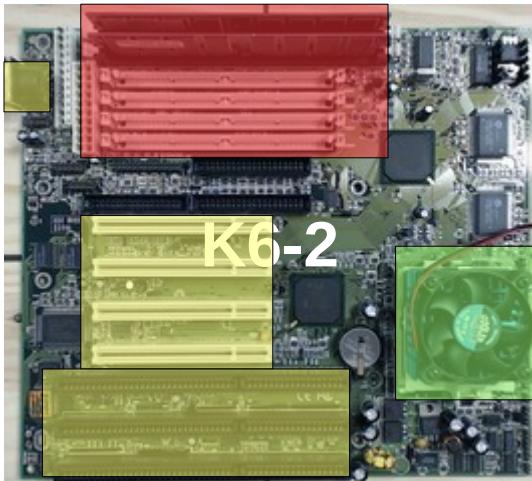
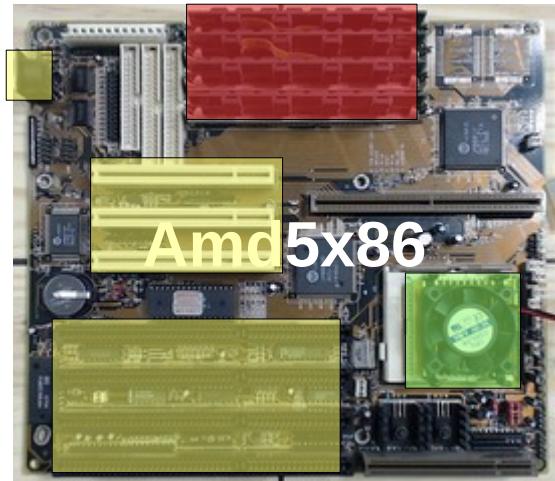
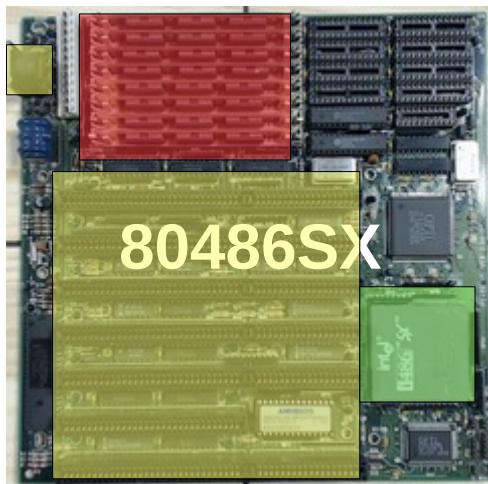
## 50 ans d'évolution chez Intel



# 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



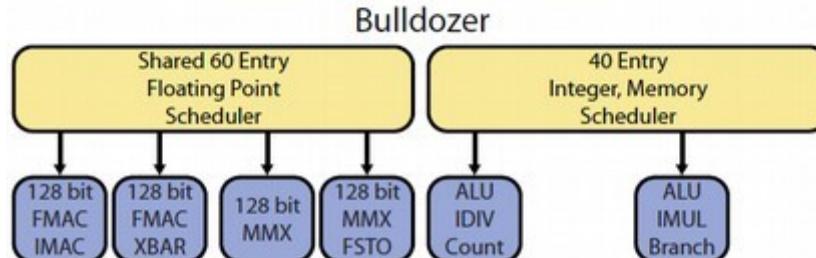
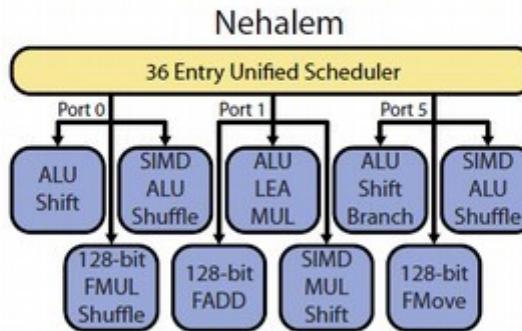
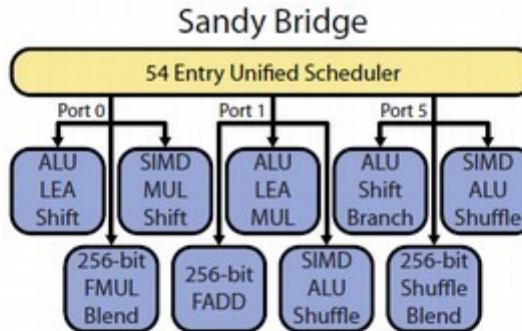
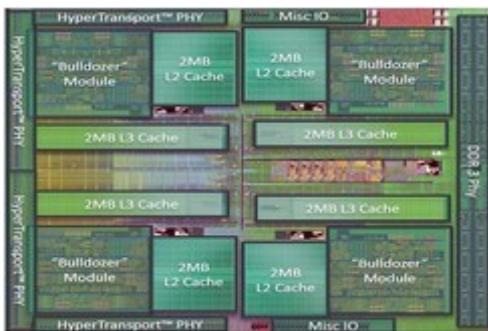
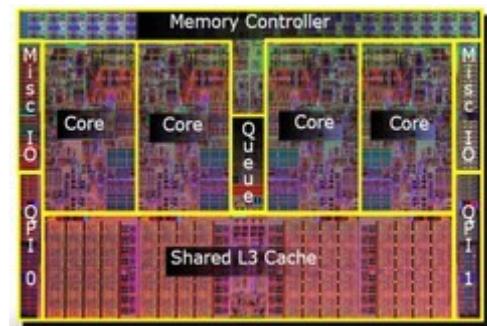
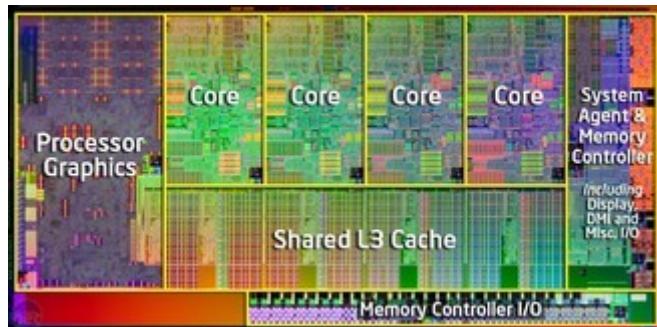
Athlon64 X2



Threadripper 1950X

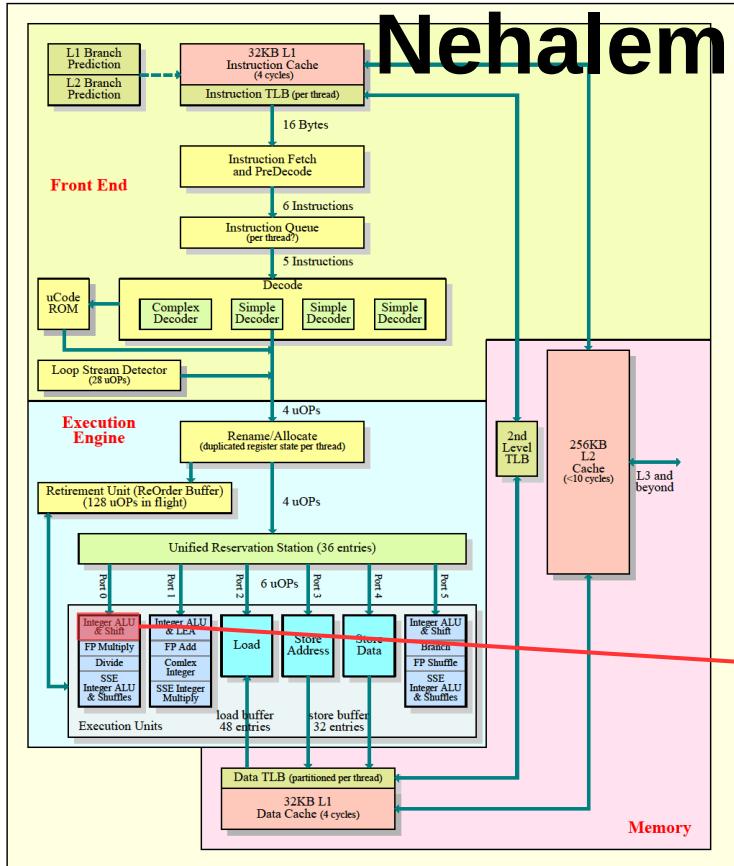
# A l'intérieur d'un socket

## Exemples de 3 quadri-cœurs

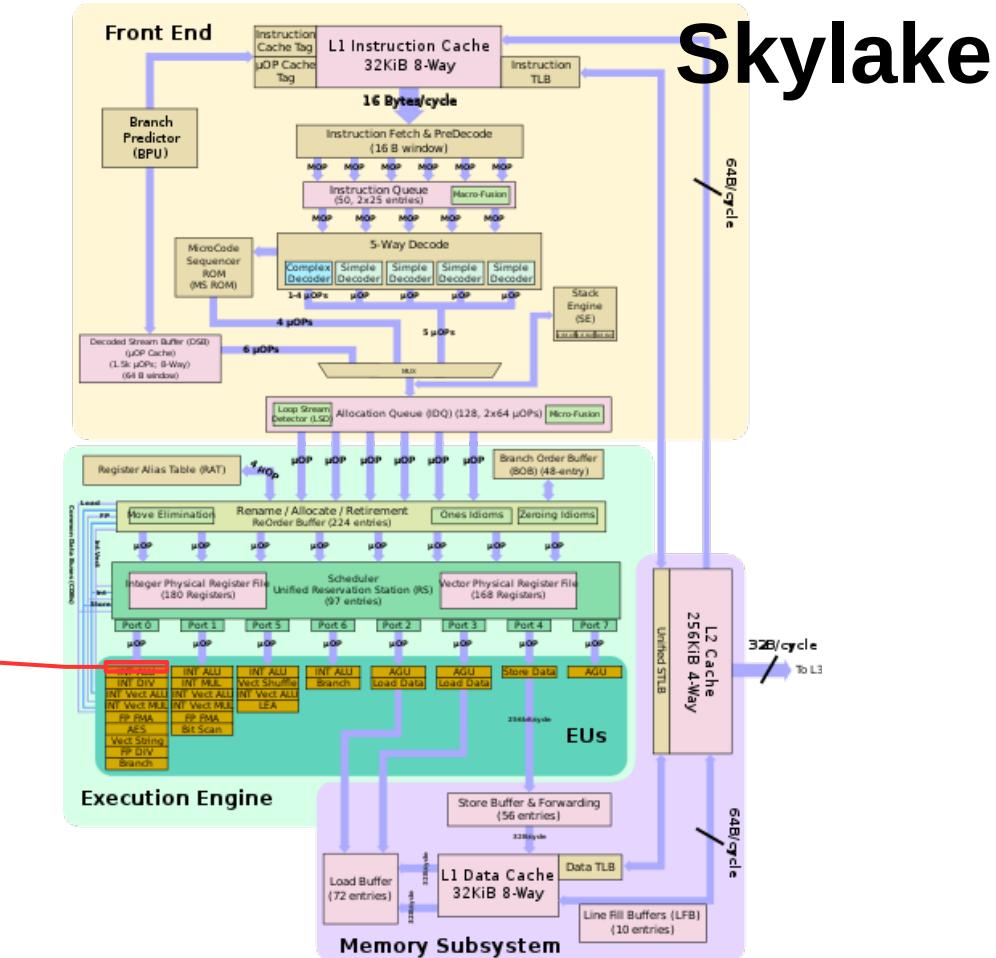


# Mais maintenant, chaque cœur est un cauchemar...

Nehalem Block Diagram

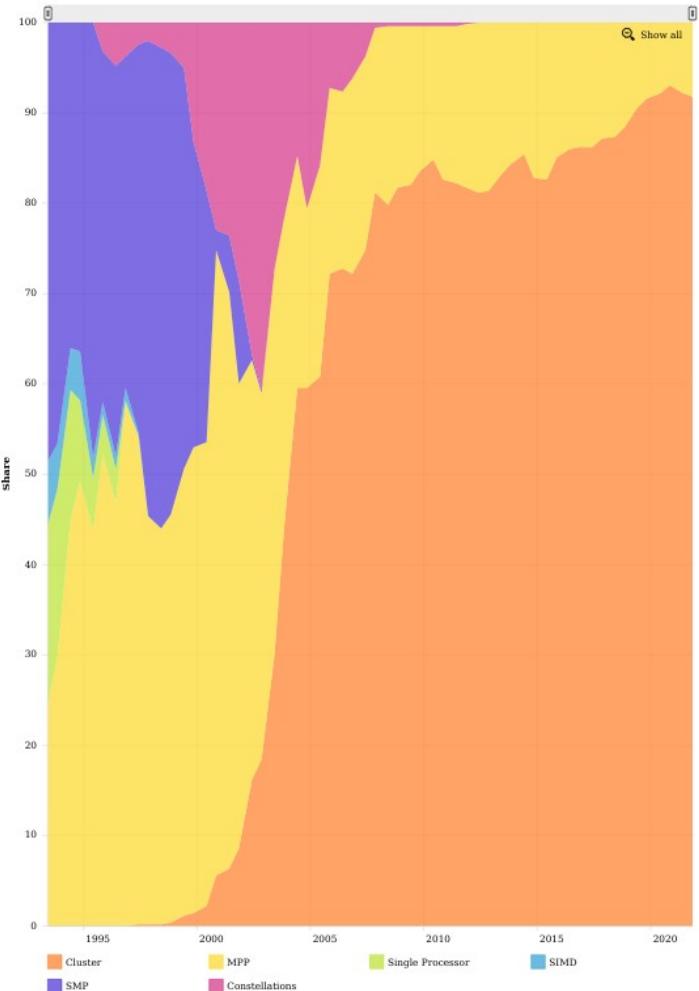


Copyright (c) 2008 Hirohige Goto All rights reserved.



# Mais tout ça, c'était du PC... De 1993 à nos jours, le Top 500

Architecture - Systems Share

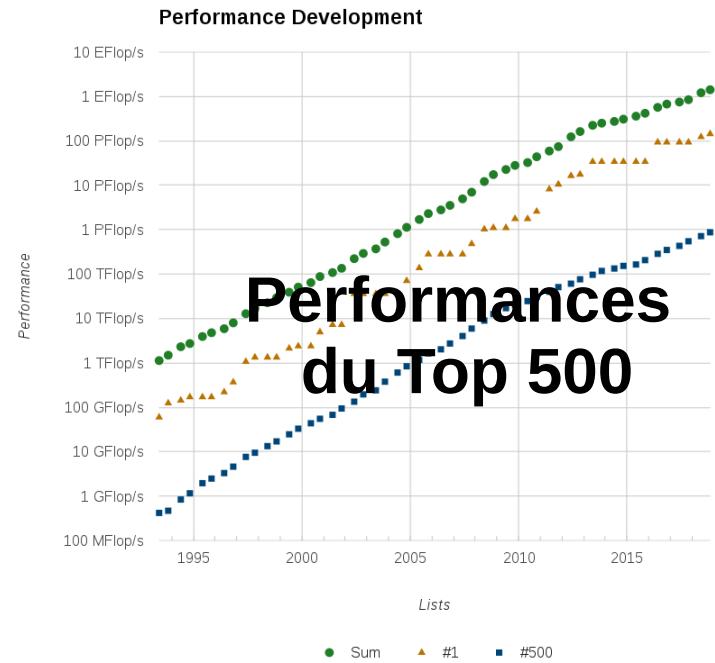
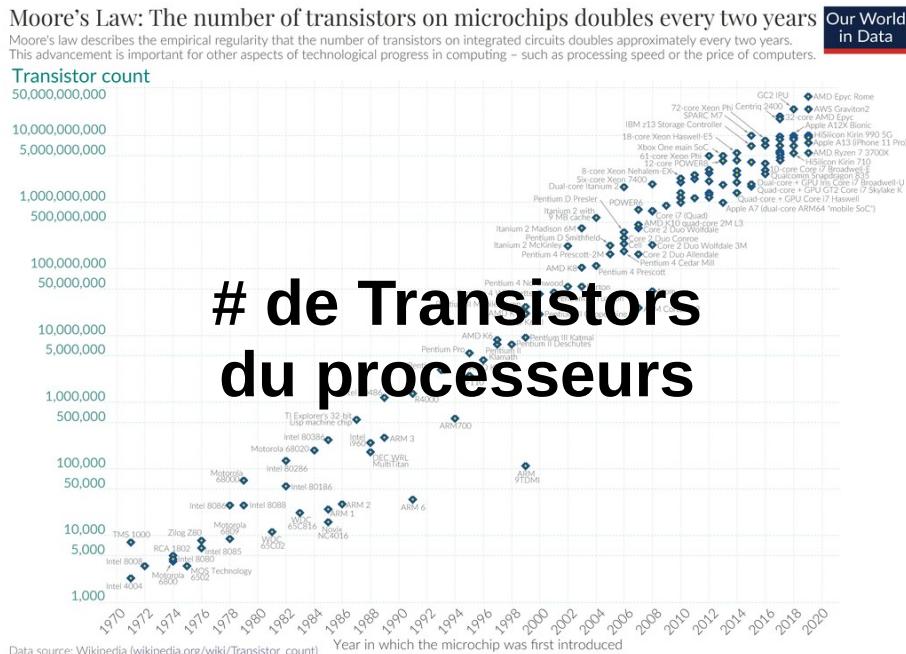


- Les catégories :
  - *Single Processor*
  - SMP : « Symmetric Multi Processor »
  - SIMD : « Simple Instruction Multiple Data »
  - MPP : « Massive Parallel Processor »
  - Constellation : Cœurs >> Nœuds
  - Cluster : machines « génériques »
- 4 ont « disparu »...

# « Lois » en informatique

## La loi de Moore (et son évolution)

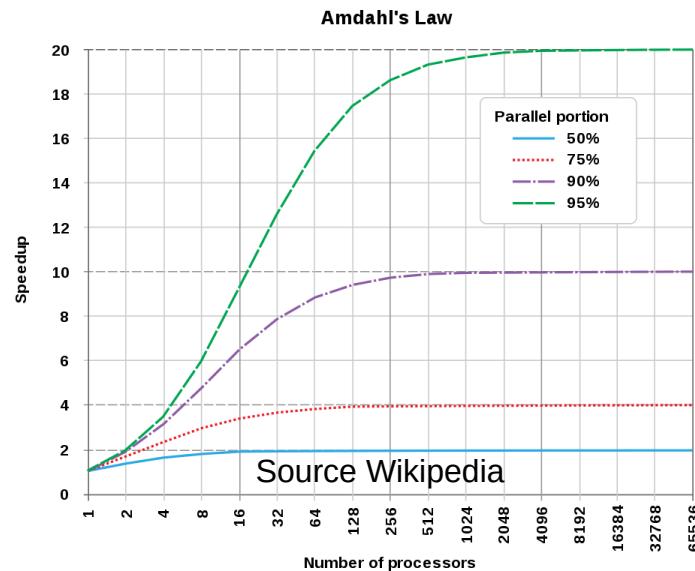
- 1965 : « complexité » des transistors tous les ans
  - 1975 : x2 des transistors sur processeur tous les 2 ans
  - Actuelle : x2 de « performance » tous les 18 mois



# « Lois » en informatique

## La loi d'Amdahl (et ses limites)

- Loi d'Amdahl :  $T = T_1(s + p/n)$ 
  - $T$  &  $T_1$  : durées d'exécution & pour  $n=1$
  - $s$  &  $p$  : % séquentiel & parallèle du code
  - $n$  : unités de traitement
- Accélération =  $T_1/T = 1/(1-p+p/n)$



Parallel Rate	N=500		N=1000		
	Part parallèle	Accélération	Efficacité	Accélération	Efficacité
90%		9.8	2%	9.9 (+0.1%)	1%
99%		83	17%	91 (+9%)	9%
99.9%		334	66%	500 (+50%)	50%
99.99%		476	95%	909 (+91%)	91%

# Codes & Performance : Quelles définitions choisir ?

- Etymologie (Etymonline)
  - **Code** : du latin codex « livre, livre de lois »
    - « compilation systématique de lois » (1236)
    - « système de communication télégraphique » (1866)
  - **Performance** :
    - « accomplissement » (de quelque chose)
    - Signification « une chose réalisée » autour des années 1590
    - « ensemble de caractéristiques optimales d'un système » (1929)
  - **Benchmark** :
    - De « bench-mark », « point de référence pour un contrôle » (1838), sens figuratif depuis 1884 :
    - « un problème ou un test standardisé servant de base à l'évaluation ou la comparaison »
- Et nous choisissons
  - **Code** : les deux :-), **Performance** : les trois :-), **Benchmark** : le figuratif

# Qu'est-ce donc que le Code ? Un protocole d'expérimentation !

- Dans la cuisine :
  - Nous avons les ingrédients, mais nous voulons un plat !
- Dans le domaine scientifique, 3 formes :
  - **Simulation** : « Au service (discret ?) de la théorie »
  - **Traitement** : pour expérimentateurs « exigeants »
  - **Visualisation** : voir (les choses) pour percevoir (leurs interactions) (et aussi partager !)
- Chaque exécution est une expérience (et une unique!)
  - **Recettes** : « codes » devenant des « processus »
  - **Ustensiles** : librairies, OS, matériels, réseaux, ...
  - **Ingrédients** : modélisation, données
  - **Exécution** : et une expérience NE peut se réduire à ses résultats !

# Si calculer, c'est cuisiner... Le code, ce n'est que la recette...

Code ~ Recette

Ordinateur ~ Cuisine

Données d'entrée ~ Ingrédients

Données de sortie ~ Repas

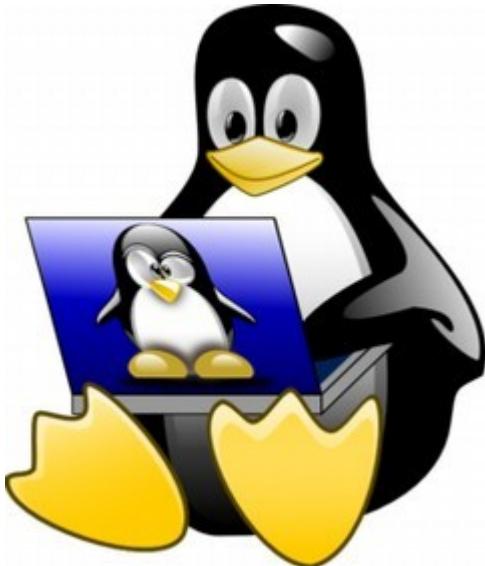
Processus ~ Cuisine

Unité de contrôle ~ Cuisinier

ALU ~ Ustensile

Moi ~ Client

Requête de batch ~ Commande



# Les familles de programmes

- Comment distinguer les différents codes que j'utilise ?
  - « Mon code à moi dont je suis fier ! »
  - Le code de mon chef
    - ou plutôt une stratification produite par des générations successives d'étudiants
  - Un code « métier »
    - Modèle Ikea : distribué avec des instructions de compilation
    - Modèle Crozatier : prêt à l'emploi
- Comme dans chaque famille, les problèmes avec l'héritage !
  - Dépendances avec :
    - Des librairies génériques : BLAS, Lapack, FFTw
    - Des librairies propriétaires : Mathworks, Intel, Nvidia, AMD, ...
    - Le matériel !

# Performance : comment ? Une question d'objectifs !

- Mettre tous les bagages et la famille dans la voiture
- Attirer l'attention des filles en sortant de boîte de nuit
- Aller d'un point A à un point B dans une ville embouteillée
- Gravir Pikes Peak aux USA



# Performance : comment ? Une question d'observables



## La performance en sport

- Pour faire un 100 mètres ?
- Pour boucler un marathon ?
- Pour lancer un poids ?
- Pour accomplir un heptathlon ?



# Performance : Conditionnée par les objectifs

- **Vitesse** : plutôt une fréquence,  $1/(temps \text{ écoulé})$  (ou Wall Clock)
- **Travail** : immobilisation de ressources sur une durée
- **Efficacité** : exploitation optimale des ressources
- **Scalabilité** : capacité à passer à une échelle supérieure
- **Portabilité** : intégration à d'autres infrastructures informatiques
- **Maintenabilité** : temps humain passé à maintenir le système
- Approche générale :
  - Définir un critère
  - Rechercher les valeurs extrêmales sur un ensemble de tests pertinents

# La vitesse comme critère

## « Speed, I'm Speed... »

- Toutes les durées, et pas seulement le temps d'exécution
- Dans l'utilisation d'un code, les 3 coûts (temporels) :
  - **Coût d'entrée** : apprendre à l'utiliser, l'intégrer à l'infrastructure, ...
  - **Coût d'exploitation** : le maintenir, l'utiliser
  - **Coût de sortie** : le remplacer par un autre code équivalent, ou une technologie équivalente
- Optimisation (et son biais) :  $DD/DE > 1$  est-il pertinent ?
  - DE : Durée totale de toutes mes exécutions (DuréeExécution)
  - DD : temps passé à tenter de minimiser la durée d'exécution (DuréeDéveloppement)
- Pour estimer ces valeurs :
  - Outils système, outils de métrologie dans les langages, les codes, les matériels, ...
- « Et après moi ? Le déluge ? » : quel avenir pour le code ?



# Le travail comme critère de perf'

- Travail : « Time is money »
  - Ressources : CPU, RAM, GPU, stockage, réseau, ...
  - En fait, une Matriochka :
    - CPU : plusieurs coeurs, CU, ALU, piles, ...
    - RAM/SRAM : 4 niveaux
    - Stockages : local, lent & partagé (NFS), rapide & partagé (GlusterFS, Lustre, ...)
    - Réseaux : lent (Gigabit), rapide & basse latence (InfiniBand, Omnipath)
- Job : réservation (& immobilisation) de ressources
  - Classiquement, dans un système de batchs : Slots \* Wall Clock
- Pour un code, quelle est l'empreinte système ?
  - Outils de profilage, outils système

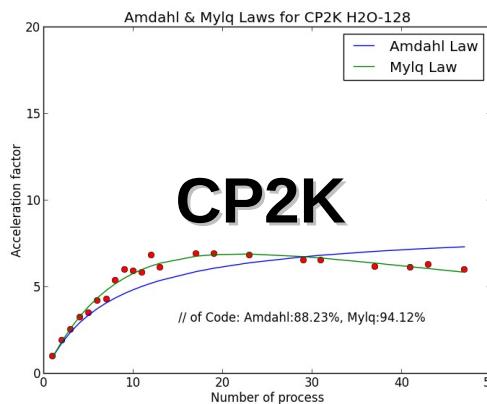
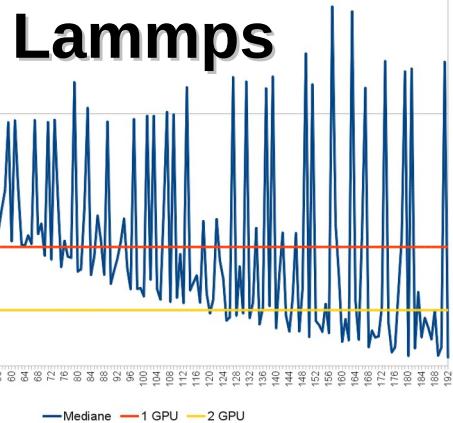
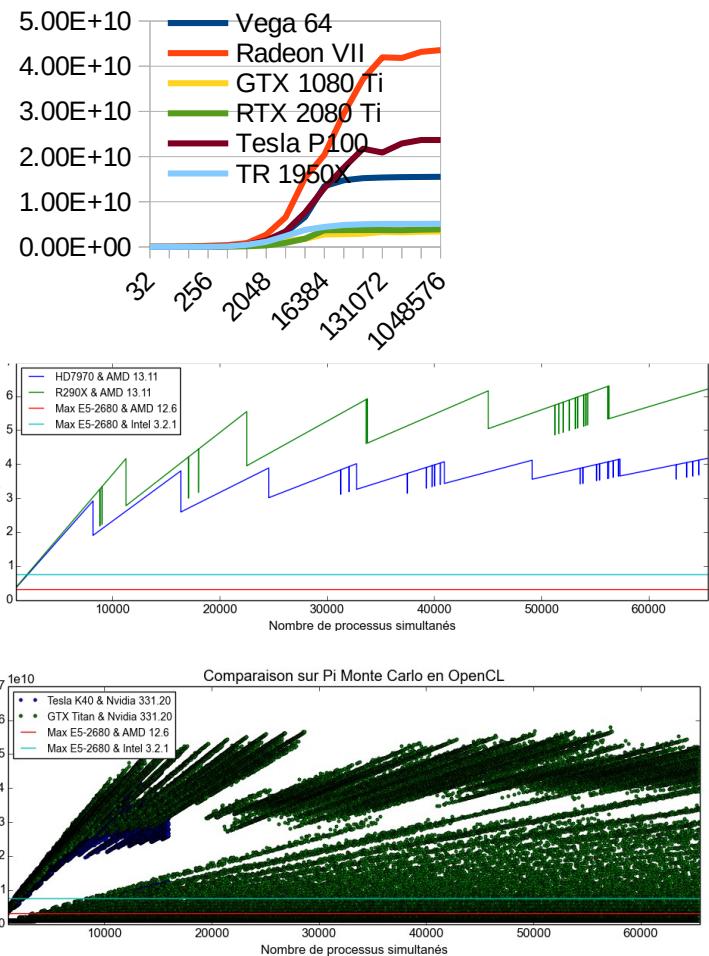
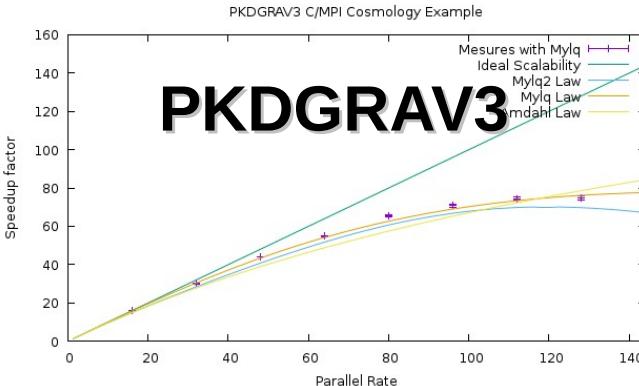
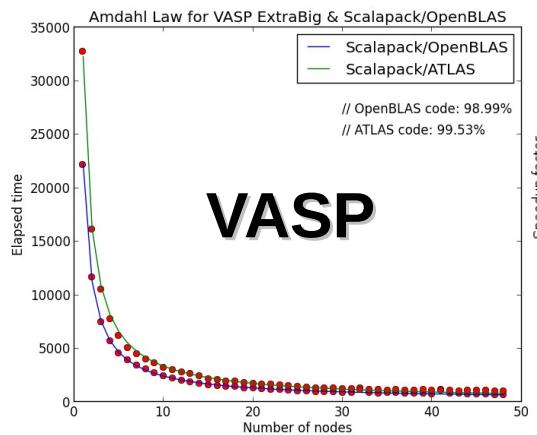
# La scalabilité comme critère

- La scalabilité :
  - Dans une tâche à accomplir : Temps écoulé ?  $f(\text{Temps écoulé})$
  - Dans les ressources à mobiliser :  $g(\text{Ressources Système})$
- Pièges à éviter :
  - Les effets d'échelle (en fait, les effets de seuil sont pire)
  - Besoin d'un chef d'orchestre ? Du quatuor à l'orchestre symphonique...
  - Quel que soit le programme, les ressources machines sont limitées...
    - Vous pensez vraiment que ça me fait rire :-/ ?

## La parallélisation incontournable, mais pourquoi ?

# Codes « métiers » & hardware

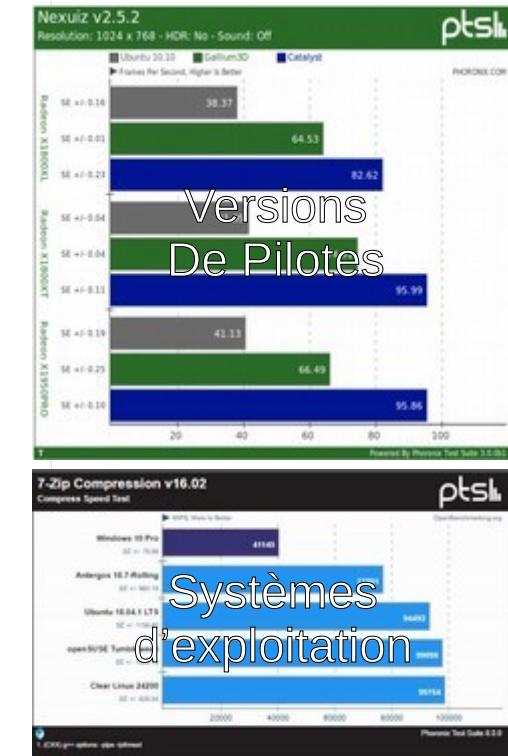
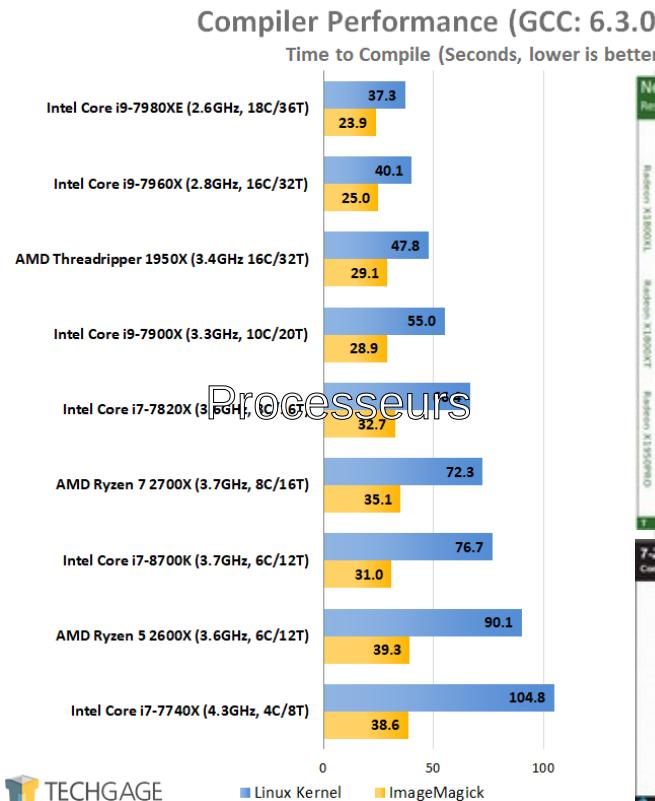
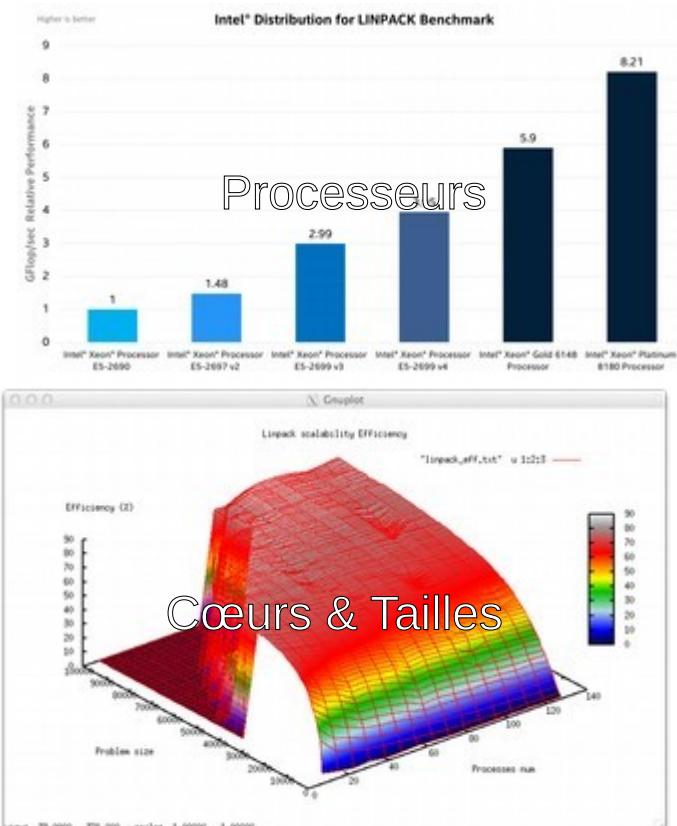
## Exemples de scalabilité



# Performances des ordinateurs Entre Linpack & Phoronix...

## Linpack

## Phoronix



# Performances des ordinateurs Linpack & le Top 500

- **Quoi** : les 500 super-ordinateurs les plus performants de la planète
- **Quand** : 2 fois par an, en juin et en novembre
- **Où** : tout autour du monde
- **Qui** (a écrit le code) : ICL de University of Tennessee
- **Comment** : High Performance LinPack en FP64, décomposition LU pour résoudre des systèmes
- **Combien** : Open Source, avec dépendances BLAS : <https://www.netlib.org/benchmark/hpl/>
- **Pourquoi** : « pour montrer ses muscles » entre pays (enjeu stratégique)...

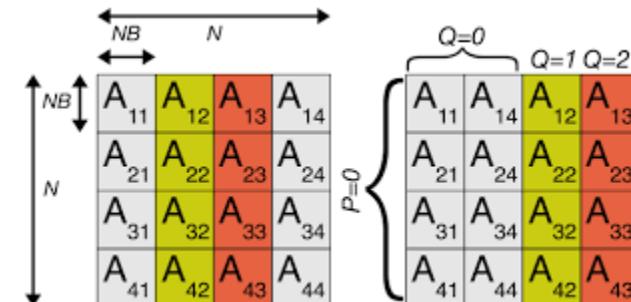
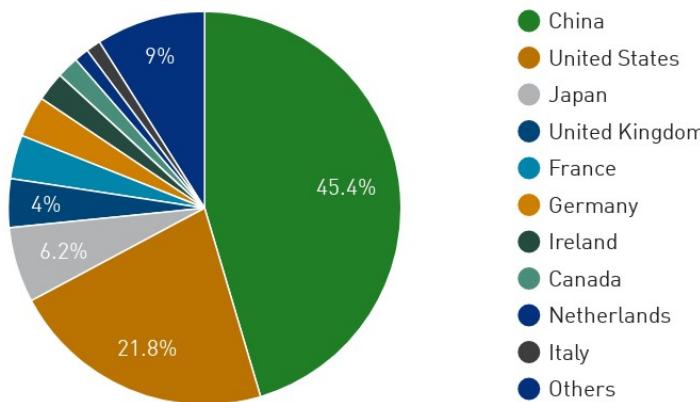


Innovative Computing Laboratory

UNIVERSITY OF TENNESSEE

COMPUTER SCIENCE DEPARTMENT

Countries System Share



# Linpack & le Top 500

# Loi de Moore respectée. Comment ?

Performance Development

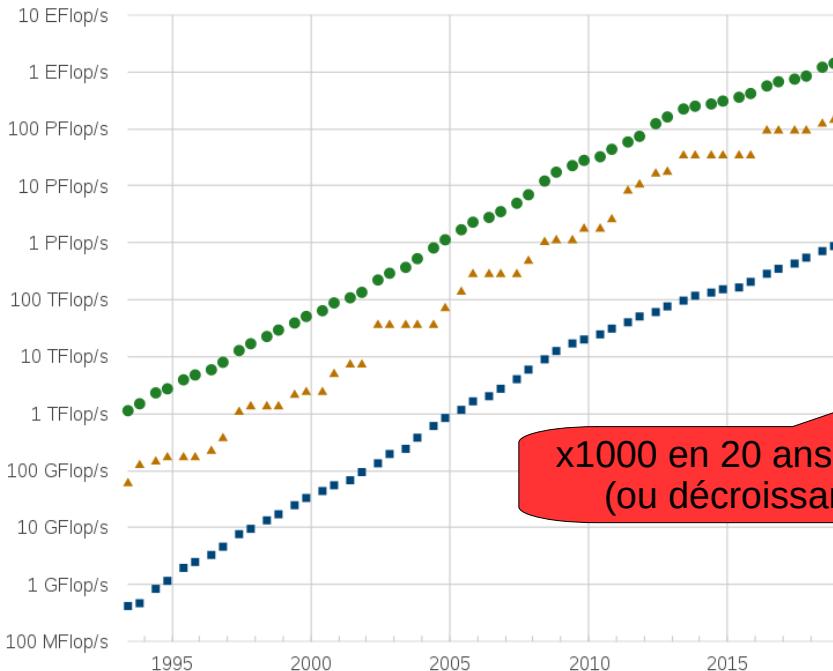
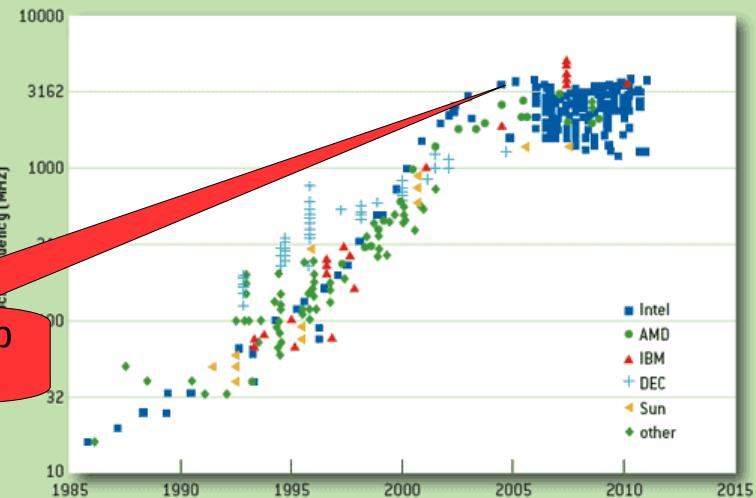


FIGURE 7

Processor Frequency Scaling Over Time

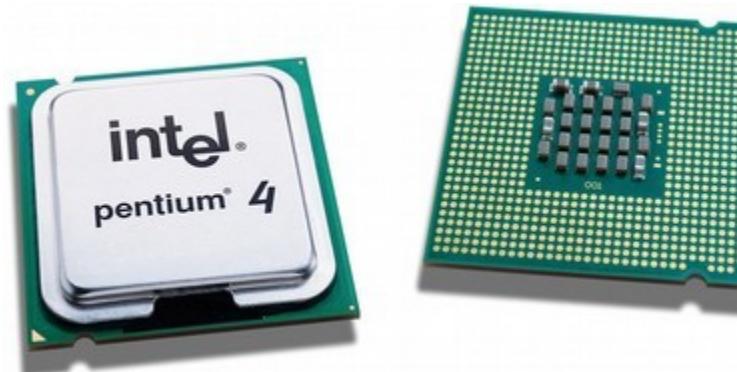


Lists  
● Sum    ▲ #1    ■ #500

## 2 voies pour casser le « mur de chaleur »

- De multi-cœurs à many-cœurs
- Accélérateurs avec des Myri-ALUs

# Energie dans les ordinateurs... La lorgnette du physicien...



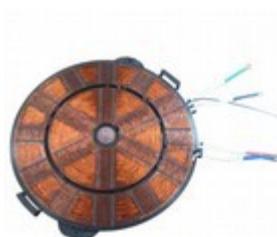
## Électronique & Thermique



# Pourquoi le parallélisme (est inévitable) ? Et sa contrainte est la TDP

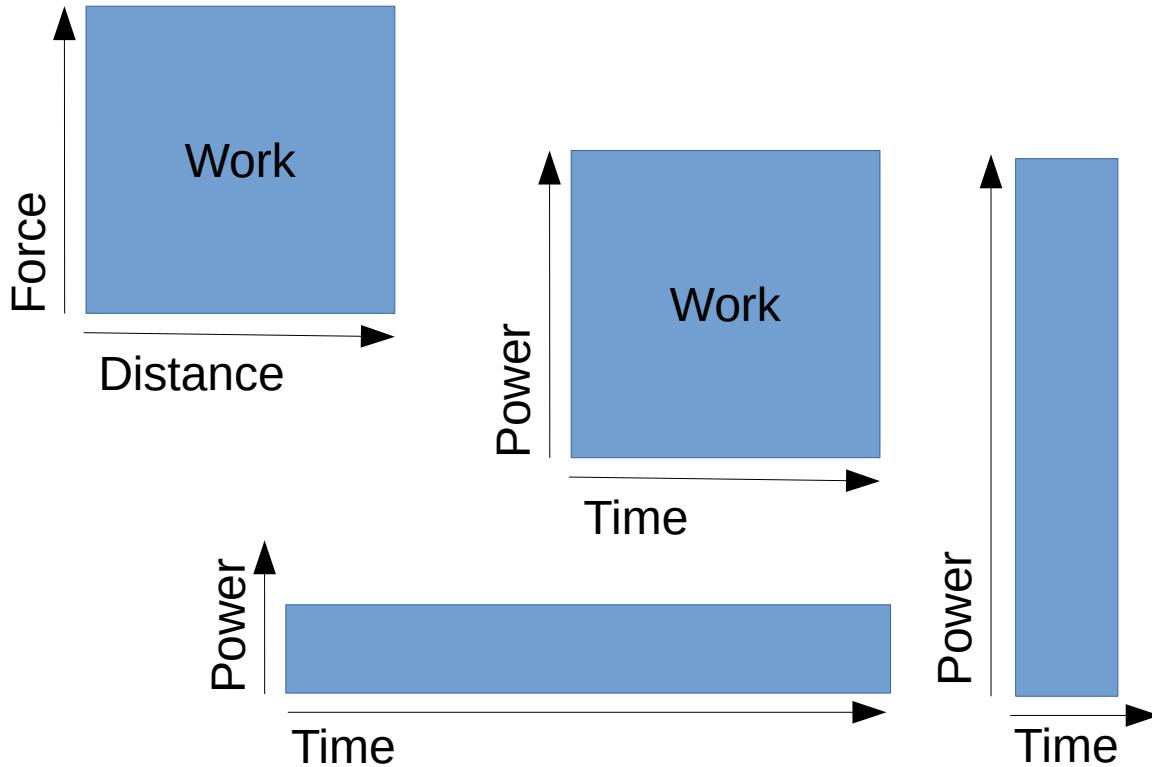
- Grandeur et décadence de la fréquence
  - Entre 1981 et 1999 : de 4 MHz à 400 MHz x100 en ~20 ans
  - Entre 1999 et 2004 : de 400 MHz à 3 GHz x~10 in 5 years
  - Entre 2004 et 2009 : de 3 GHz à 2 GHz
- Thermal Design Power : enveloppe thermique de dissipation maximale
- $TDP = \frac{1}{2} C V^2 f$ 
  - C = Capacitance, f = fréquence, V = tension d'alimentation (fonction de f !)
- TDP pour un processeur : 150 W (sur 4 cm<sup>2</sup>)
  - Densité de chaleur d'une plaque à induction !
- TDP devient le facteur limitant de puissance

Capacitance = Finesse<sup>2</sup> . Nb Transistors . Constante de Mylq (~ 0.015)



**Solution viable : augmenter les unités de traitement (PU)**

# Energie en calcul scientifique... Le point du vue du physicien



## Mécanique

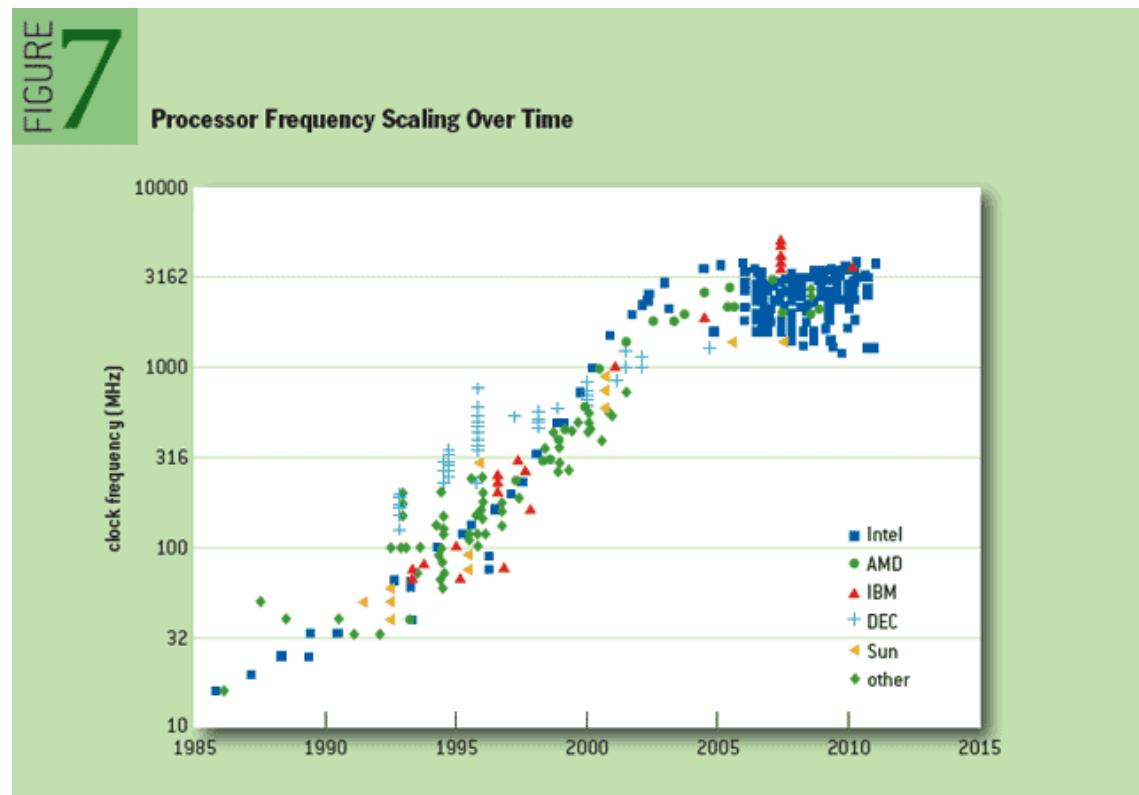
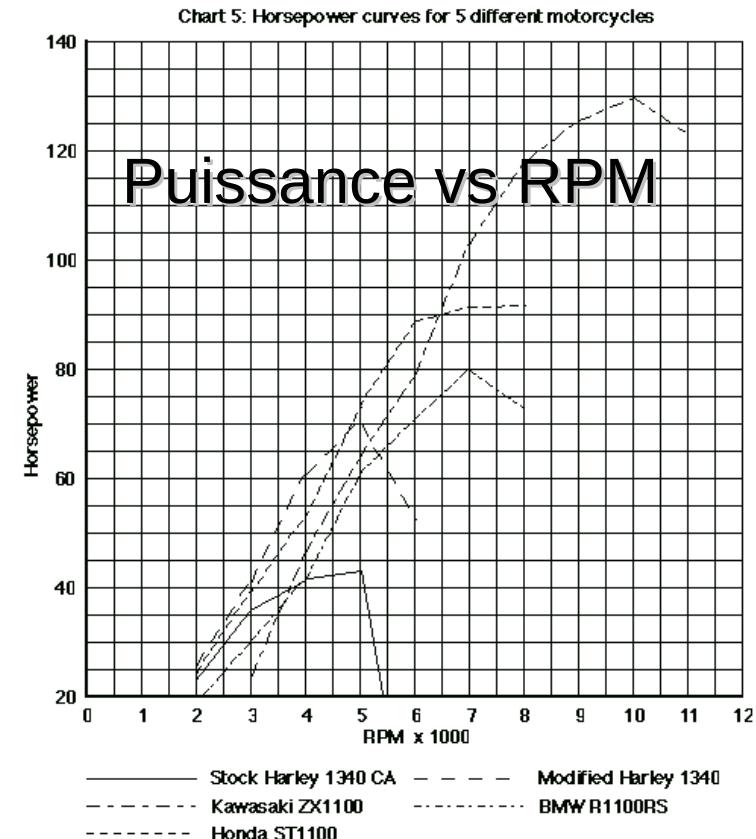
La puissance est définie  
Comme le produit de :

- Fréquence
- Nombre d'unités
- Puissance unitaire

$$W = \int_{x_1}^{x_2} F dx = \int_{t_1}^{t_2} P dt$$

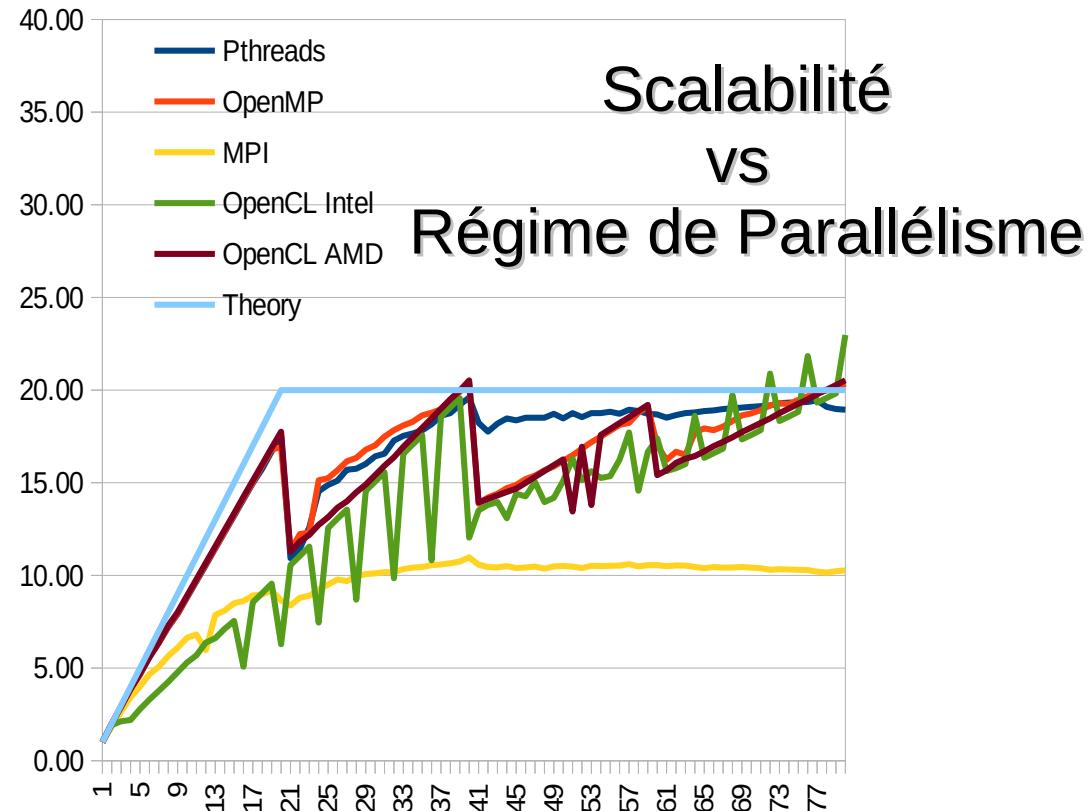
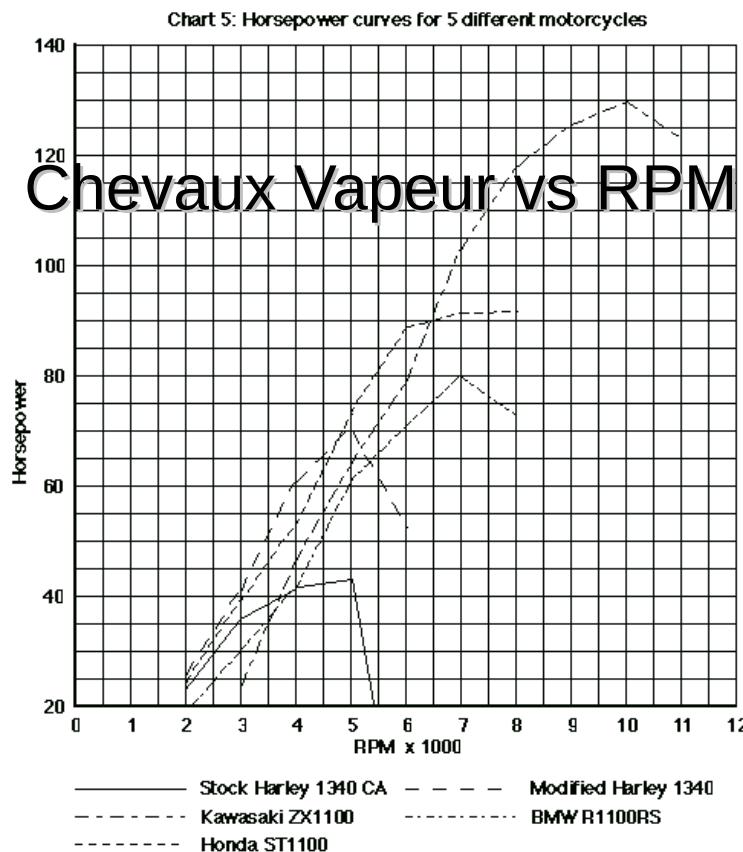
Caractérisation des « moteurs » de traitement...

# Il y a longtemps, entre 1985... et 2005, la variable est la fréquence !



# Travail & Ressources Informatiques

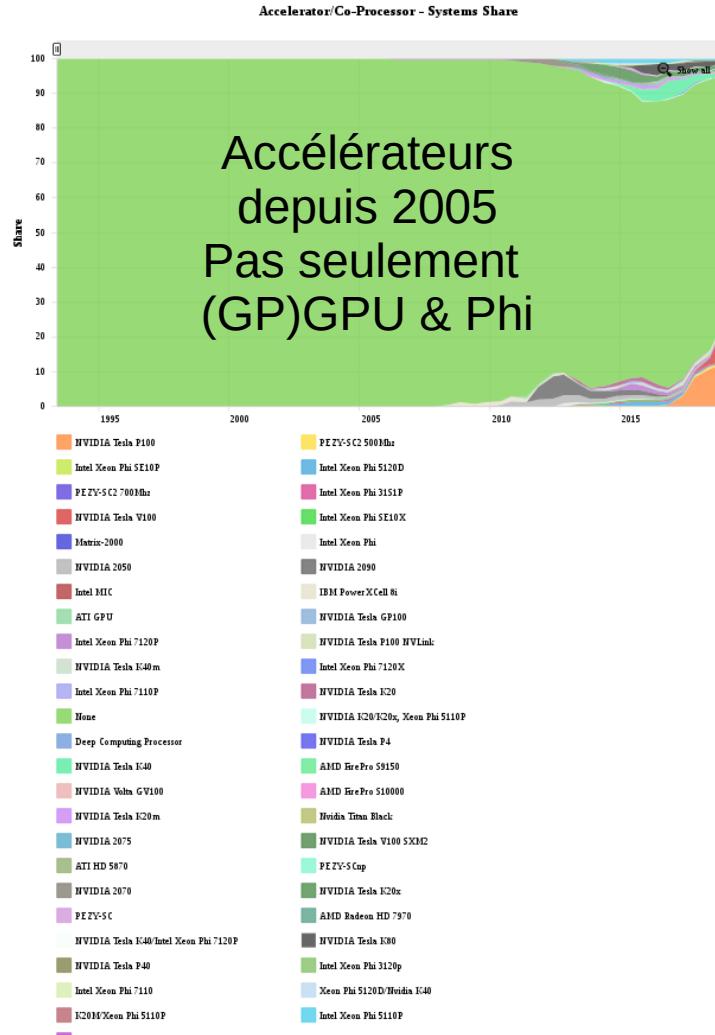
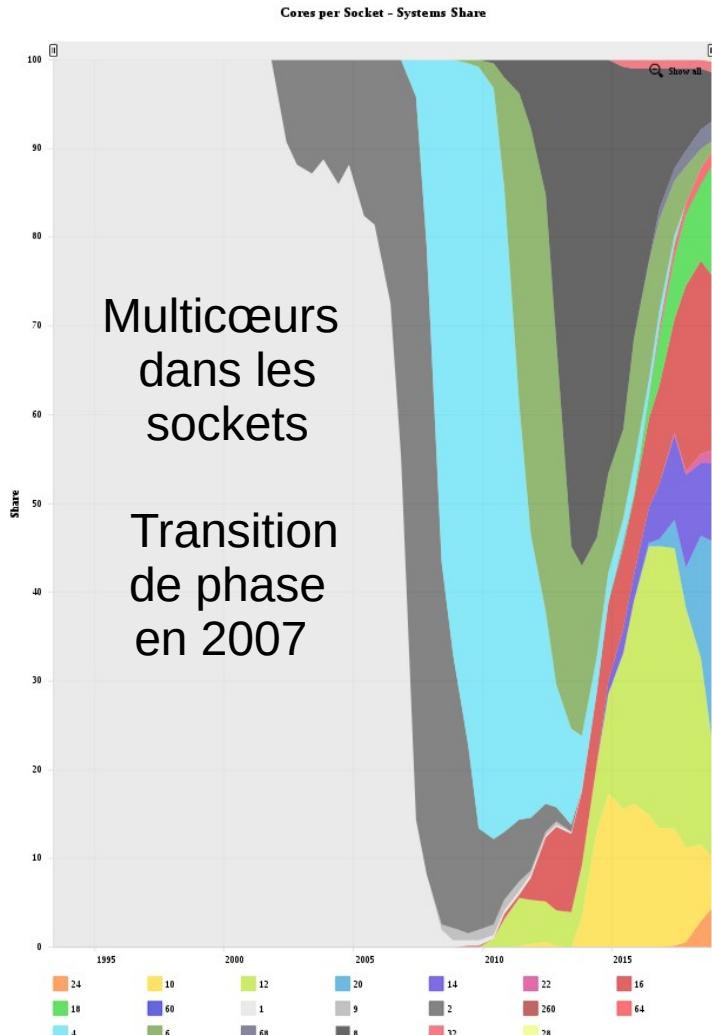
## Un moteur comme source de puissance



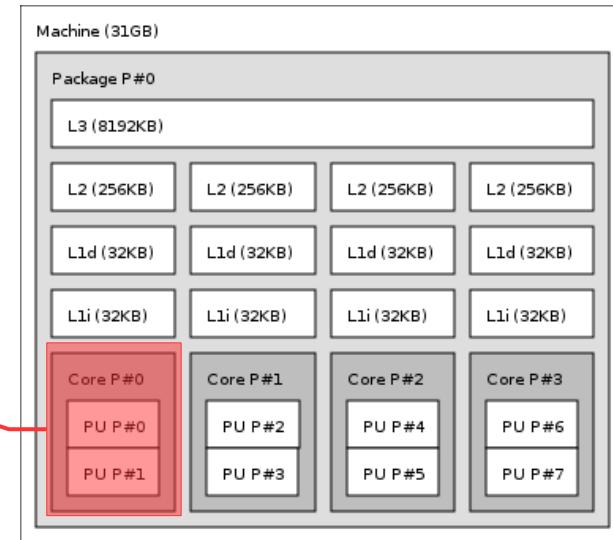
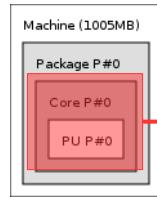
- Quel comportement de ces « moteurs » à la charge ?
- Le « moteur », un « système » englobant matériel, OS et logiciels

# Linpack & le Top 500

# Loi de Moore respectée. Comment ?

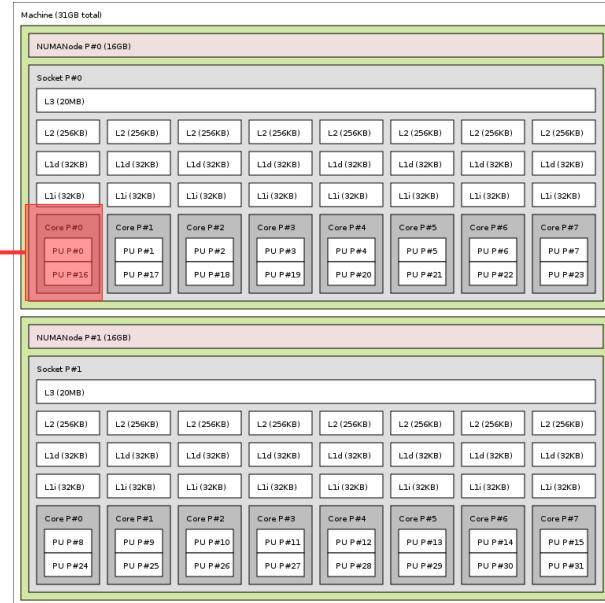
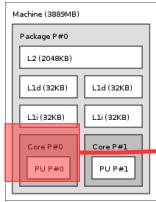


# Evolution des machines... De 2004 à 2013, sur les laptops !

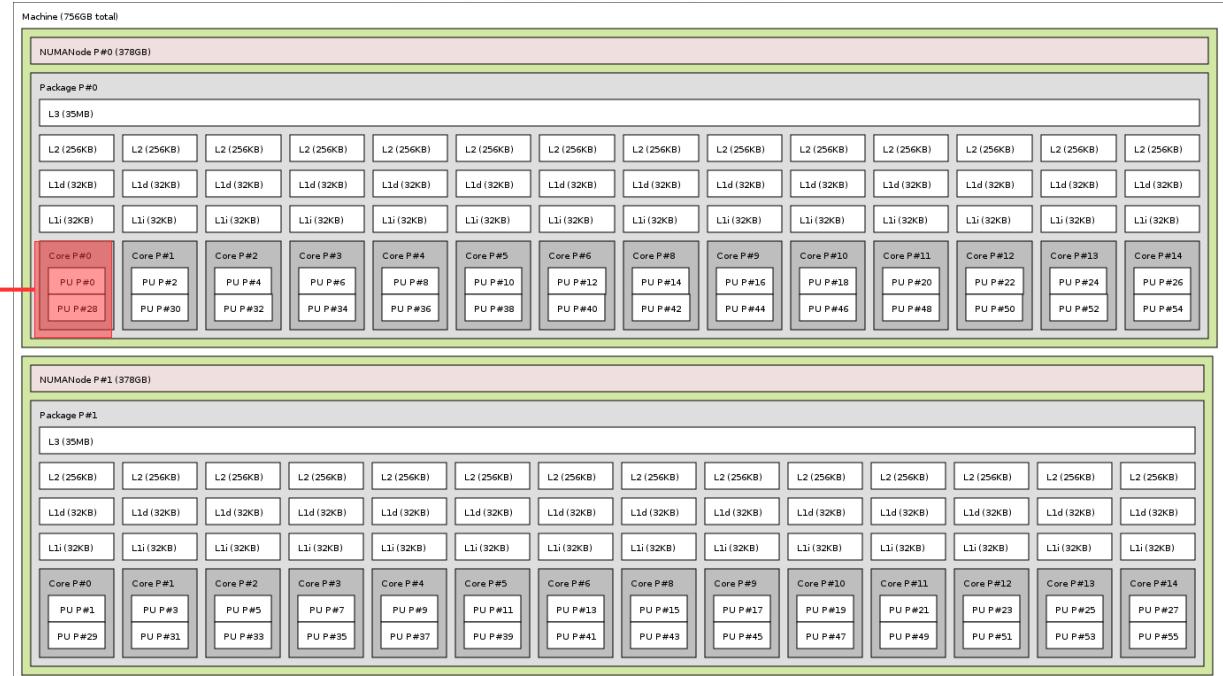
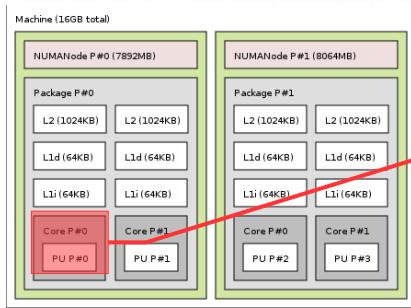


# Evolution de 2007 à 2016

## Sur les stations de travail



# Evolution de 2007 à 2016 sur des serveurs...



Et même pire... Dans 2U (unités)  
4 serveurs très modernes



Matrice (Désordre) (G)									
Matrice (Désordre) (P)									
Matrice (Désordre) (C)									
L1 (G)	L2 (G)	L3 (G)	L4 (G)	L5 (G)	L6 (G)	L7 (G)	L8 (G)	L9 (G)	L10 (G)
L1 (G)	L2 (G)	L3 (G)	L4 (G)	L5 (G)	L6 (G)	L7 (G)	L8 (G)	L9 (G)	L10 (G)
L1 (G)	L2 (G)	L3 (G)	L4 (G)	L5 (G)	L6 (G)	L7 (G)	L8 (G)	L9 (G)	L10 (G)
L1 (G)	L2 (G)	L3 (G)	L4 (G)	L5 (G)	L6 (G)	L7 (G)	L8 (G)	L9 (G)	L10 (G)
C1 (P)	C2 (P)	C3 (P)	C4 (P)	C5 (P)	C6 (P)	C7 (P)	C8 (P)	C9 (P)	C10 (P)
C1 (P)	C2 (P)	C3 (P)	C4 (P)	C5 (P)	C6 (P)	C7 (P)	C8 (P)	C9 (P)	C10 (P)
C1 (P)	C2 (P)	C3 (P)	C4 (P)	C5 (P)	C6 (P)	C7 (P)	C8 (P)	C9 (P)	C10 (P)
C1 (P)	C2 (P)	C3 (P)	C4 (P)	C5 (P)	C6 (P)	C7 (P)	C8 (P)	C9 (P)	C10 (P)
C1 (C)	C2 (C)	C3 (C)	C4 (C)	C5 (C)	C6 (C)	C7 (C)	C8 (C)	C9 (C)	C10 (C)
C1 (C)	C2 (C)	C3 (C)	C4 (C)	C5 (C)	C6 (C)	C7 (C)	C8 (C)	C9 (C)	C10 (C)
C1 (C)	C2 (C)	C3 (C)	C4 (C)	C5 (C)	C6 (C)	C7 (C)	C8 (C)	C9 (C)	C10 (C)
C1 (C)	C2 (C)	C3 (C)	C4 (C)	C5 (C)	C6 (C)	C7 (C)	C8 (C)	C9 (C)	C10 (C)
P1 (G)	P2 (G)	P3 (G)	P4 (G)	P5 (G)	P6 (G)	P7 (G)	P8 (G)	P9 (G)	P10 (G)
P1 (G)	P2 (G)	P3 (G)	P4 (G)	P5 (G)	P6 (G)	P7 (G)	P8 (G)	P9 (G)	P10 (G)
P1 (G)	P2 (G)	P3 (G)	P4 (G)	P5 (G)	P6 (G)	P7 (G)	P8 (G)	P9 (G)	P10 (G)
P1 (G)	P2 (G)	P3 (G)	P4 (G)	P5 (G)	P6 (G)	P7 (G)	P8 (G)	P9 (G)	P10 (G)
P1 (P)	P2 (P)	P3 (P)	P4 (P)	P5 (P)	P6 (P)	P7 (P)	P8 (P)	P9 (P)	P10 (P)
P1 (P)	P2 (P)	P3 (P)	P4 (P)	P5 (P)	P6 (P)	P7 (P)	P8 (P)	P9 (P)	P10 (P)
P1 (P)	P2 (P)	P3 (P)	P4 (P)	P5 (P)	P6 (P)	P7 (P)	P8 (P)	P9 (P)	P10 (P)
P1 (P)	P2 (P)	P3 (P)	P4 (P)	P5 (P)	P6 (P)	P7 (P)	P8 (P)	P9 (P)	P10 (P)
P1 (C)	P2 (C)	P3 (C)	P4 (C)	P5 (C)	P6 (C)	P7 (C)	P8 (C)	P9 (C)	P10 (C)
P1 (C)	P2 (C)	P3 (C)	P4 (C)	P5 (C)	P6 (C)	P7 (C)	P8 (C)	P9 (C)	P10 (C)
P1 (C)	P2 (C)	P3 (C)	P4 (C)	P5 (C)	P6 (C)	P7 (C)	P8 (C)	P9 (C)	P10 (C)
P1 (C)	P2 (C)	P3 (C)	P4 (C)	P5 (C)	P6 (C)	P7 (C)	P8 (C)	P9 (C)	P10 (C)

# Et au Centre Blaise Pascal Pour les deux extrêmes...



# Linpack & le Top 500

## Comment retrouver Rpeak :-/ ?

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	Sierra - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371

- Cores : unités de calcul
- Rmax : résultat du HPL
- Rpeak : perf' théorique
- Ratio max/peak ~ 73 %

### • La chinoise, troisième...

- Cores = 40960 SW26010
  - 260 cores/SW26010
- Rpeak ~ Cores\*1.45e9\*8
- Donc : 8 instructions / cycle

### • L'Américaine, première...

- Cores = 9216 Power9 + 27648 V100
  - 22 cores/Power9, 80 StreamMultiprocessors/V100
- Rpeak ~ Rpeak<sub>CPU</sub>+Rpeak<sub>GPU</sub>
  - Rpeak<sub>CPU</sub> = 9216\*22\*3.07e9\*64 ~ 20 %
  - Rpeak<sub>GPU</sub> = 27648\*80\*1.13e9\*64 ~ 80 %
- Donc 64 instructions / cycle on CPU&GPU

# Autant d'« Instructions par Cycle » !

## Comment c'est possible ?

- Vieille & profonde évolution dans le CPU :
  - RISC remplace le CISC :
    - RISC : 1 instruction par cycle
  - Le « Pipelining » devient la «règle» :
    - Les 5 opérations « semblent » exécutées en 1 cycle
  - L'unité en virgule flottante (Floating Point Unit) est intégrée dans le CPU
    - C'est une ALU spécifique
  - Multiplication d'ALU spécialisées dans le CPU
    - De 5 dans AMD K6 à 10 par cœur dans une architecture Zen
  - Chaque ALU intègre la vectorisation :
    - A l'origine des opérations 3D (MMX, 3DNow, SSE, SSE2, ..., AVX512)
  - Le Socket intègre plusieurs coeurs (UC+ALUs+Cache mémoire)

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX

Clock Cycle	1	2	3	4	5	6	7

# Linpack & le Top 500

# Puissance des machines hybrides

Rank	System	Cores	Rmax [TFlop/s]	Rpeak [TFlop/s]	Power [kW]
1	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	Sierra - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	Tianhe-2A - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384
6	Trinity - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States	979,072	20,158.7	41,461.2	7,578
7	AI Bridging Cloud Infrastructure (ABCi) - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 8XSM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649
8	SuperMUC-NG - ThinkSystem SD530, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path , Lenovo Leibniz Rechenzentrum Germany	305,856	19,476.6	26,873.9	
9	Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , Cray Inc. DOE/SC/Oak Ridge National Laboratory United States	560,640	17,590.0	27,112.5	8,209
10	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom , IBM DOE/NNSA/LLNL United States	1,572,864	17,173.2	20,132.7	7,890

Avec (GP)GPU

Avec accélérateur Xeon Phi

- Novembre 2018 :
  - 7/10 sont Hybrides dans le Top 10
  - 25 % dans le Top 500
- Avant 2012 :
  - Hybride : MPI+OpenMP
- Après 2012 :
  - OpenMP+Cuda
  - OpenMP+MPI
  - OpenMP+Cuda+MPI
  - OpenCL+MPI

# Pourquoi je n'utilise pas LinPack ? Testez vous-même... Escroquerie...

- Trop de paramètres « libres » (jamais publiés)
- Trop « d'escroqueries » des constructeurs
  - HPL & HPCC : ~ 15 % d'efficacité
  - Linpack Intel MKL : de 57 % à 92 % d'efficacité
  - CUDA de Nvidia : ~ 20 % d'efficacité (et un cauchemar à compiler)



Robert\_Crovella

MODERATOR

Yes, the best HPL performance will come from HPL code specifically provided on a case-by-case basis by NVIDIA. It is not publicly available, and the hpl-2.0\_FERMI\_v15 will not achieve highest performance on GPUs newer than FERMI.

Posted 02/02/2017 04:18 AM

#4

- Comment ils atteignent 75 % d'efficacité sur des millions de cœurs ?
- Trop de « bits » : seulement 64 bits, trop pour des applications
- Donc chercher ailleurs, plus simple, plus universel

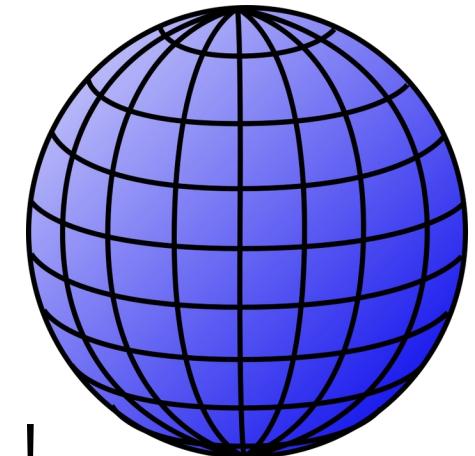
# Pourquoi le parallélisme : chemin... Où sommes-nous ? Où allons-nous ?

- Où sommes-nous ? Nous utilisons des codes quotidiennement
- Où allons-nous ? Nous voulons plus de « performance »
- Comment y aller ? Le parallélisme est une voie mais pourquoi ?
- Avant de « tomber dans le trou du lapin blanc » du parallélisme :
  - Quelles sont les pratiques d'utilisations ?
  - Comment définir la performance d'un code ?
  - Quel critère de performance choisir ?
  - Comment atteindre la performance souhaitée ?
  - Comment vérifier que la performance est bien atteinte ?

# Qu'est-ce que le parallélisme ?

## Retournons à la source

- Etymologie (etymonline.com) : à côté d'un autre
  - De para- « à côté »
  - De allelois « chacun », de allos « autre »
- Parallélisme : tâches à accomplir, ressources limitées
  - Exécution de différentes tâches en parallèles
  - Exécution d'une tâche sur plusieurs ressources
    - Communications éparses : gros grain
    - Communicatons fréquentes : grain fin
- Paradoxe du parallélisme, des méridiens !



# Combien pour le Parallélisme ? Temps, Silicium, Complexité...

- The 3 coûts temporels :
  - Coût d'entrée, coût d'exploitation, coût de sortie
  - Un temps d'exécution à comparer avec le temps d'adaptation
- Silicium : les technologies ont des coûts très différents
  - SMP (Shared Memory Processors) : chères et limitées
  - MPP (Massively Parallel Processing) : exigeantes en réseaux très spécifiques
  - Les clusters sont facilement extensibles
- Complexité : corollaire d'un large nombre de portes logiques
  - Un « cœur » GPU (QPU) est plus simple qu'un cœur CPU
  - Un « cœur » GPU (QPU) est jusqu'à 50 fois plus lent qu'un cœur CPU

# Comment penser « parallélisme » « Le grain, le problème, c'est le grain... »

- 1 Entrée / 1 Processus ? Optimisez le processus !
- 1 Entrée / Y Processus ? Optimisez chaque processus !
- X Entrées / 1 Processus ? Optimisez la distribution !
- X Entrées / Y Processus ? Optimisez les deux !

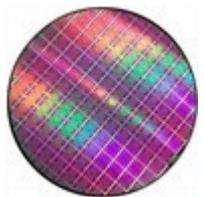
**Le Grain est défini par le taux de communication !**

- Grain fin : communications fréquentes ( $\sim >> 1/\text{seconde}$ )
- Gros grain : communications éparses ( $\sim < 1/\text{seconde}$ )
- « Embarrassing parallelism » : tâches indépendantes

# Comment penser le ParallelisMe

## La Taxonomie de Flynn

- SISD : Simple Instruction Simple Data
- SIMD : Simple Instruction Multiple Data
  - Vectorisation 
- MISD : Multiple Instructions Simple Data
  - Pipelining
- MIMD : Multiple Instructions Multiple Data



Jetons un petit regard

« Derrière la porte de la cuisine » (In Silicon) ?



# Comment exploiter le parallélisme ?

## Stratégies d'exécution parallèle...

- Pipelining à grain fin, sur le silicium :

- 5 instructions simples à la fois
    - Récupération de l'instruction
    - Décodage de l'instruction
    - Exécution
    - (Accès si nécessaire à la mémoire)
    - Ecriture du résultat

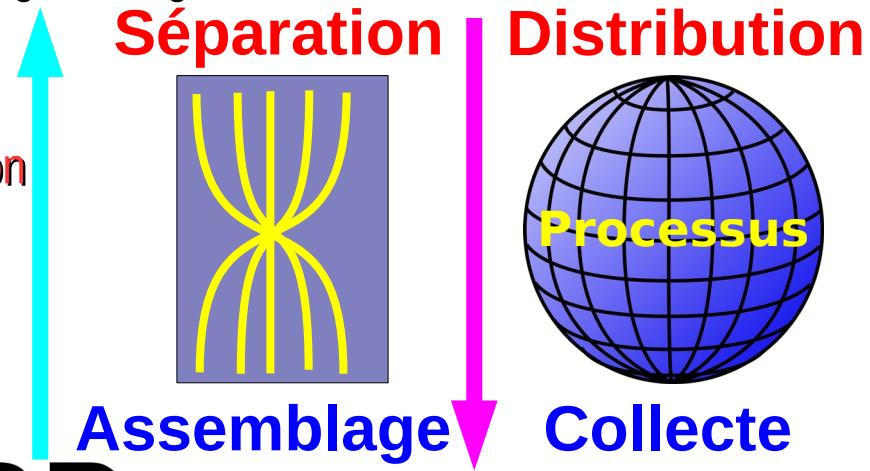
- 2 spécifications du RISC : 1 instruction/cycle, usage de registres

- 2 approches différentes :

- **Vectorisation** : Assemblage/Processus/**Séparation**
  - **Distribution** : **Distribution**/Processus/**Collecte**

- En fait, médianisation ;-)

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7



# Au delà de la cuisine

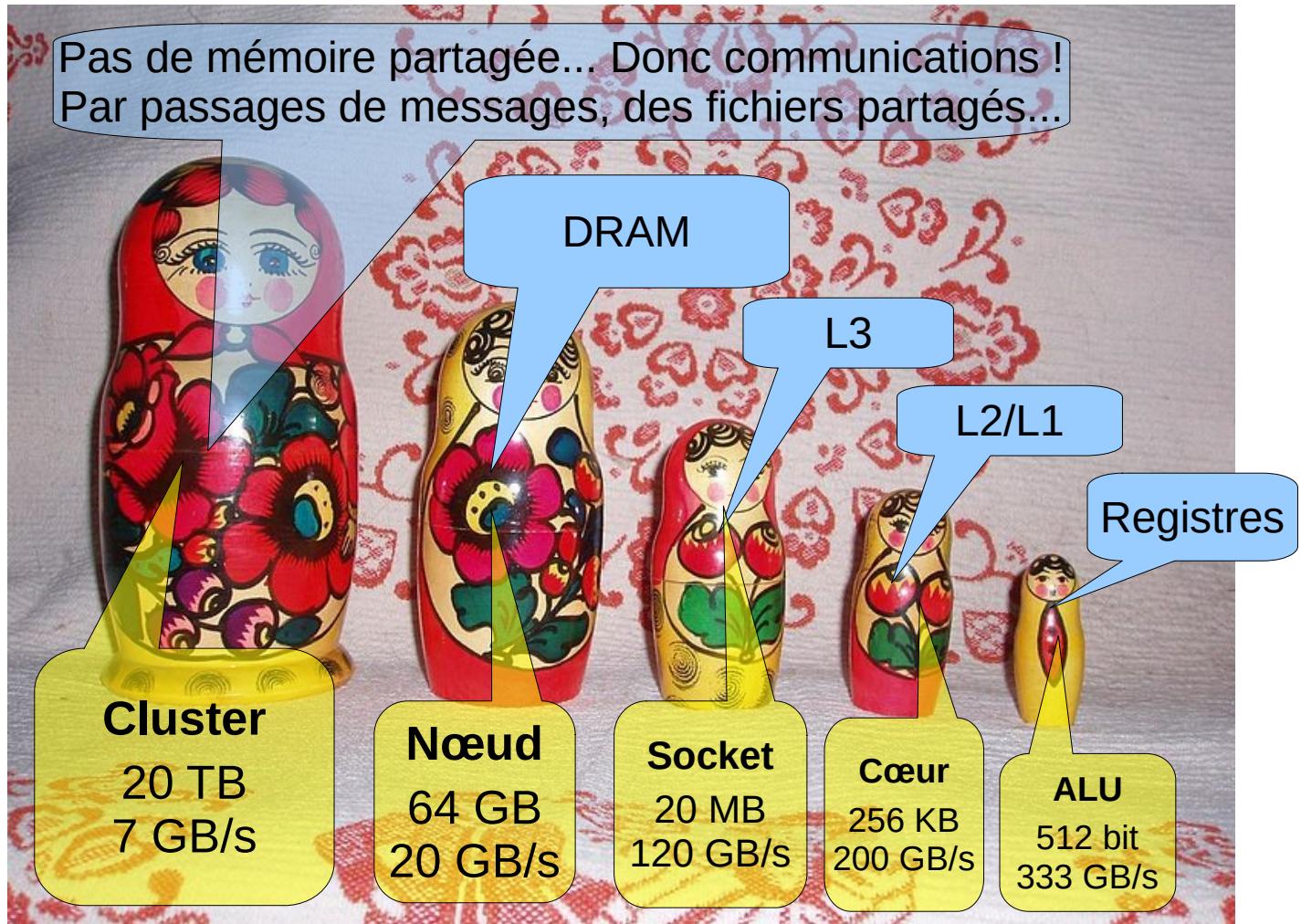
## Le déménagement.

- Objectif : déménager des 420 cartons de A à B
- Moyens :
  - 6 personnes pour chacune porter 1 carton
  - 2 chariots pouvant porter 6 cartons
  - Un ascenseur de chaque côté pouvant accueillir 12 cartons
  - Une camionnette pouvant contenir 100 cartons
- Comment organiser le déménagement ?
  - Quelles sont les unités parallèles, quelle est leur nature ?
  - Où retrouve-t-on les éléments de Flynn ?

# Le point de vue des unités de traitement



# Mémoires hiérarchiques !



# Si calculer était cuisiner... Et pour la mémoire...

Code ~ Recette

Ordinateur ~ Cuisine

Données d'entrée ~ Ingrédients

Données de sortie ~ Plat cuisiné

Processus ~ Préparation

Unité de contrôle ~ Cuisinier

ALU ~ Ustensile

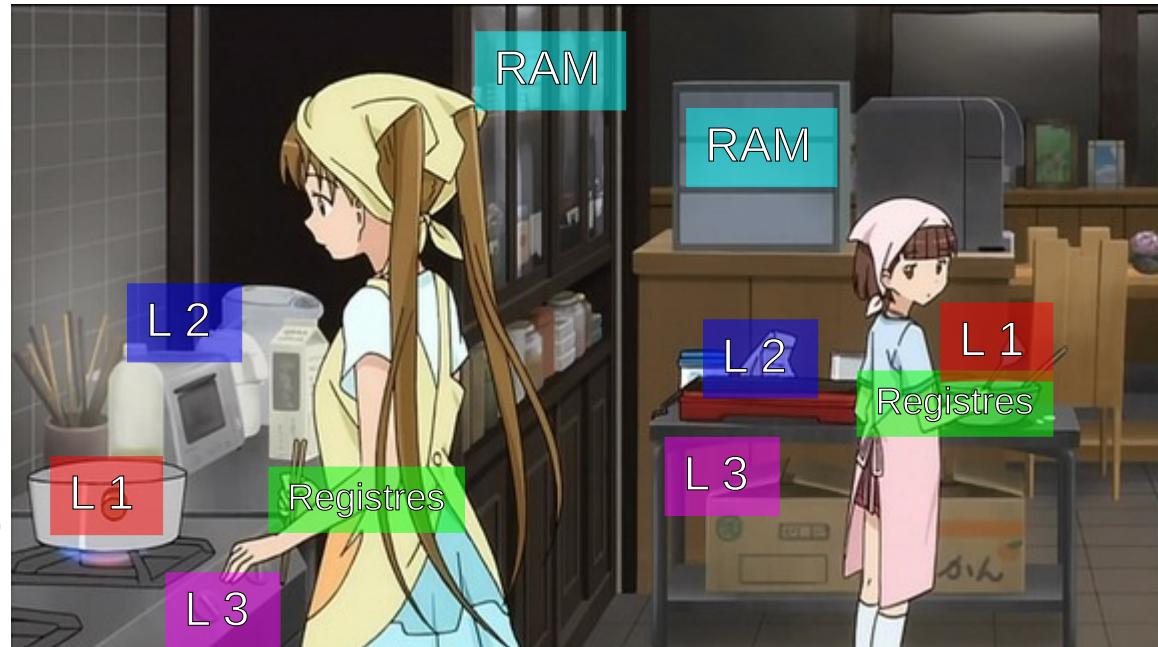
Dynamic RAM ~ Placards, tables, ...

L3 Cache ~ Tout plan de travail

L2 Cache ~ Plan de travail à un pas

L1 Cache ~ Plan de travail, récipient

Registres ~ Mains de la cuisinière



# Comment programmer en // ?

## Modèles de programmation

	Cluster	Node CPU	Node GPU	Node Nvidia	Accélérateur
<b>MPI</b>	Oui	Oui	Non	Non	Oui*
PVM	Oui	Oui	Non	Non	Oui*
<b>OpenMP</b>	Non	Oui	Non	Non	Oui*
<b>Pthreads</b>	Non	Oui	Non	Non	Oui*
<b>OpenCL</b>	Non	Oui	Oui	Oui	Oui
<b>CUDA</b>	Non	Non	Non	Oui	Non
TBB	Non	Oui	Non	Non	Oui*
OpenACC	Non	Oui	Non	Oui	Oui
Kokkos	Non	Oui	Non	Oui	Oui
SyCL	Non	Oui	Oui	Oui	Oui

## Librairies haut niveau de programmation parallèle

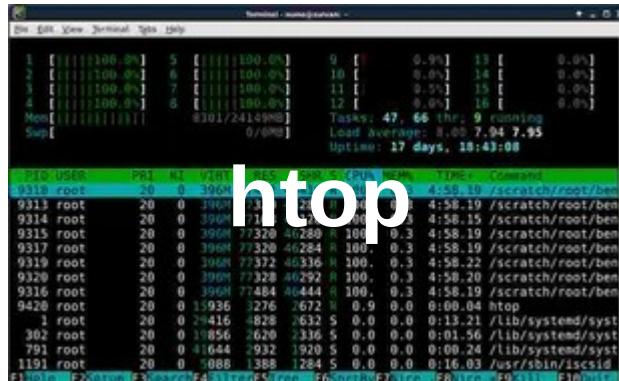
	Cluster	Nœud CPU	Nœud GPU	Nœud Nvidia	Accélérateur
BLAS	BLACS MKL	OpenBLAS MKL	cBLAS	CuBLAS	OpenBLAS MKL
LAPACK	Scalapack MKL	Atlas MKL	cIMAGMA	MAGMA	MagmaMIC
FFT	FFTw3	FFTw3	clFFT	CuFFT	FFTw3

# Votre « permis de conduire » en calcul scientifique ;-) ?

- Dans un livre français de mathématiques appliquées
    - « Les physiciens ont un usage des mathématiques que les mathématiciens assimilent facilement à de l'inconscience ! »
  - Comme un BOFH de ressources informatiques
    - « Les scientifiques ont un usage typique des ressources informatiques que j'assimile aisément à de l'inconsistance ! »
  - Est-ce que vous « conduisez » vos usages ?

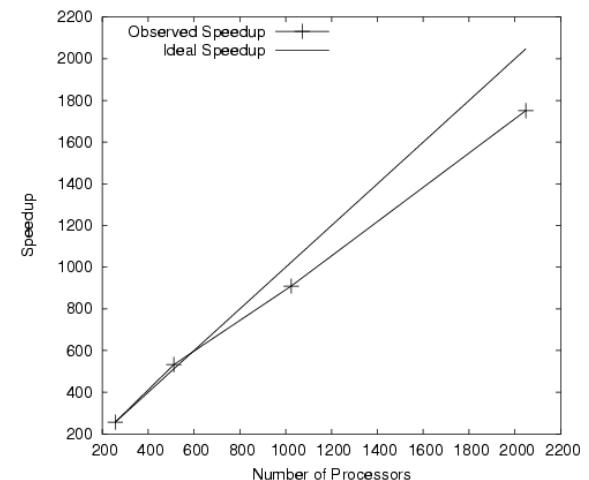
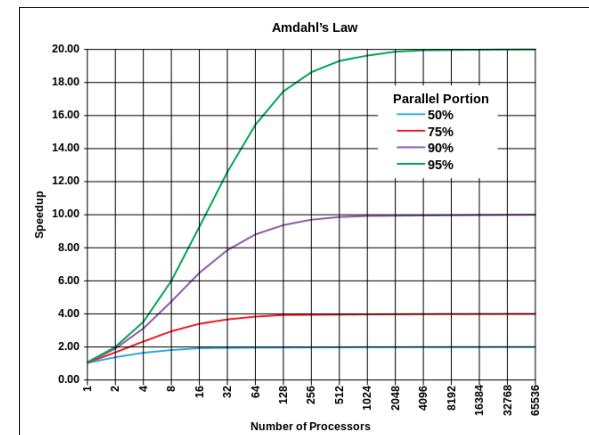
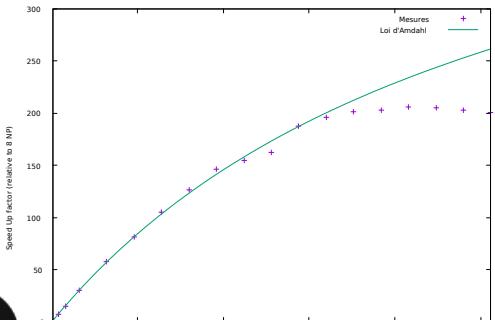
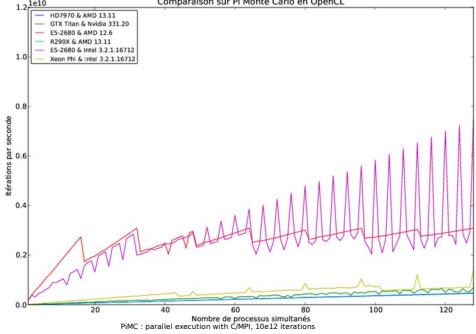
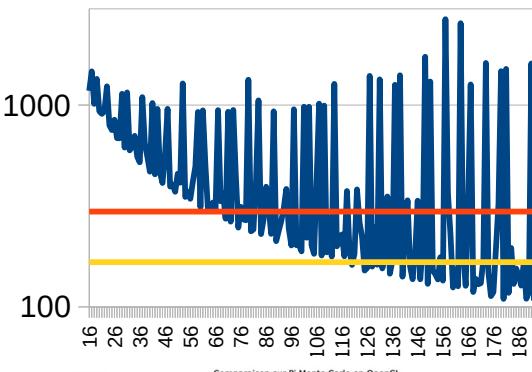


# htop



# Loi d'Amdahl ? Vérité ou mensonge

## Prêt à prendre la « pilule rouge » ?



# L'observable en informatique...

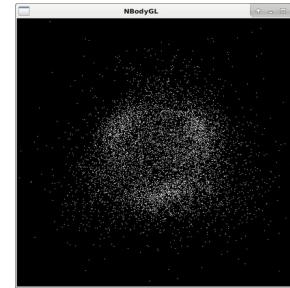
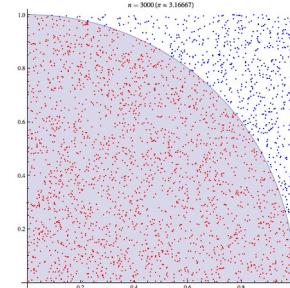
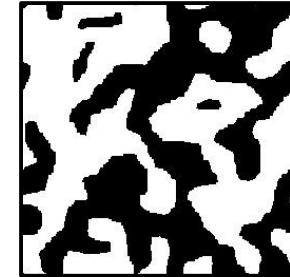
- Observable = fonction(code,données,backoffice)
  - Qu'est-ce que je maîtrise ?
- Le code ?
  - Tout le code ?
- Les données ?
  - L'accès aux données
- Le back-office
  - Réseau ? Systèmes ?
- Une constante : observer perturbe...
  - Out-of-code : time et al, mais aussi le reste
  - In-Code : commande système avec timer

# A l'origine, une notion essentielle en sciences : la « référence »

- Histoire personnelle : retour au HPC fin 2009
- Déferlante des (GP)GPU par Nvidia : GTX295 ou T1060
- Code de référence : LinPack du Top 500
  - Exercice personnel : portage du hpl de BLAS à CuBLAS
  - Finalement, plutôt inefficace face aux xBLAS : mais pourquoi ?
- Besoin de revenir aux « fondamentaux »
  - Écrire ses propres programmes exploitant les BLAS
    - Nombreuses implémentations : FBLAS, CBLAS, CuBLAS, cIBLAS, gslBLAS...
  - Aborder les problèmes avec deux approches : intégrateur & développeur

# Le souci de l'universalité... L'émergence de « codes matricés »

- Pyphi 2011 : modèle d'Ising, transition de phase
  - comparaison C, MPI, OpenCL, CUDA...
- Fin 2012 : code Pi Monte Carlo
  - Gros grain, simple, parallélisable de manière quasi-continue...
  - Charge sur le RNG, nature des variables (INT, FP, 32&64)
  - Implémentations sur toutes les approches parallélisées
- Fin 2014 : code Nbody
  - Grain fin, physique, parallélisable de manière aussi continue
  - Charge sur les calculs, nature des variables (FP32&FP64)

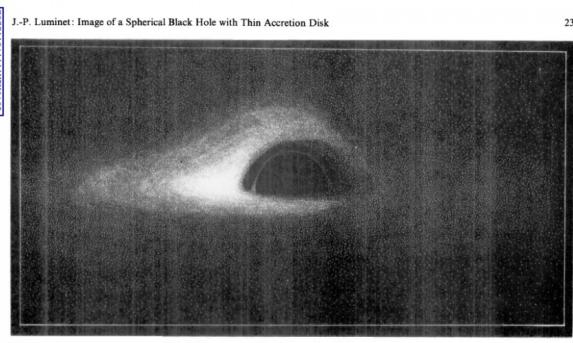


**Mais aucune exploitation massive de la mémoire...**

# 40 ans entre simulation et mesure

## De 2019 à 1979...

1979 : JP Luminet A&A



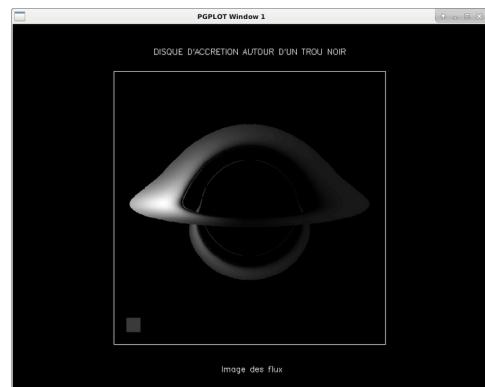
2014 : Film Interstellar



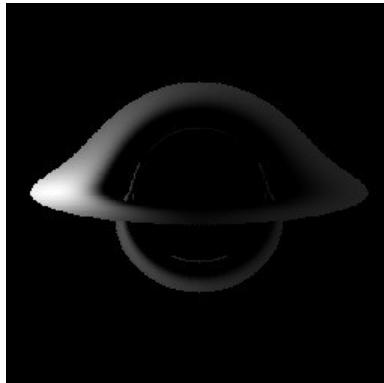
2019 : EHT, Messier 87



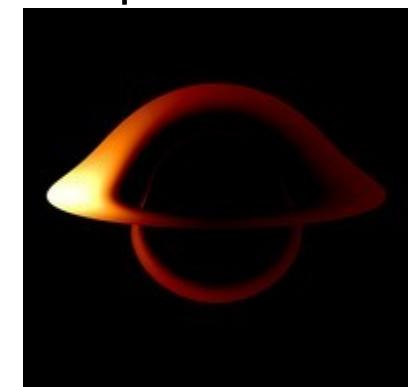
1994 : Code Fortran



1997 : Code C



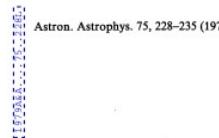
2019 : OpenCL/CUDA



# 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 secondary 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

(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 computationally interesting to calculate the 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  $r$ ,  $\theta$ , and  $\phi$  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:

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

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  $\phi_0$  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

© European Southern Observatory • Provided by the NASA Astrophysics Data System

# De l'article au rapport

- Métrique de Schwarzschild :

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

- Équation 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 :  $u=1/r$

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

- Déivation de l'équation polaire :

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

- Système d'équations à résoudre :  $v = \frac{du}{d\phi}$

$$\text{et } \frac{dv}{d\phi} = 3 \frac{m}{b} u^2 - u$$

DISQUE D'ACCRETION  
AUTOOUR D'UN TROU NOIR

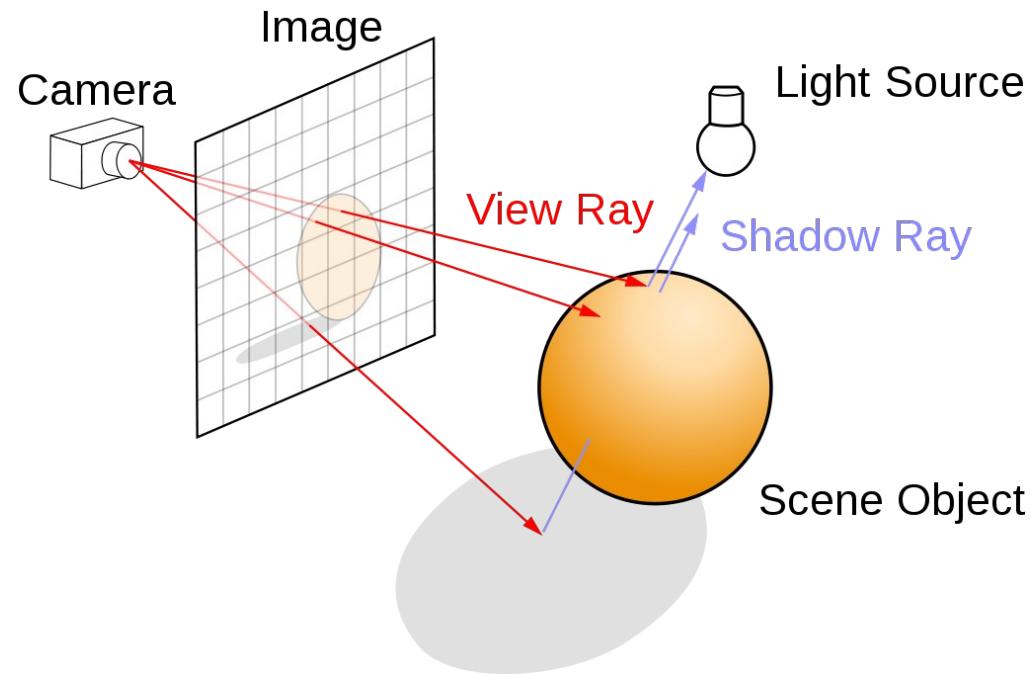
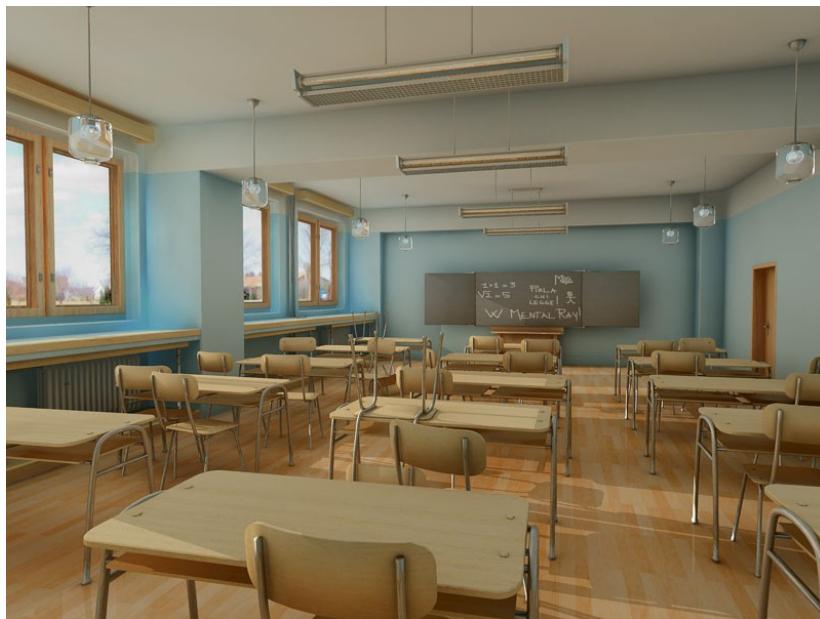
IMAGE & SPECTRE

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...

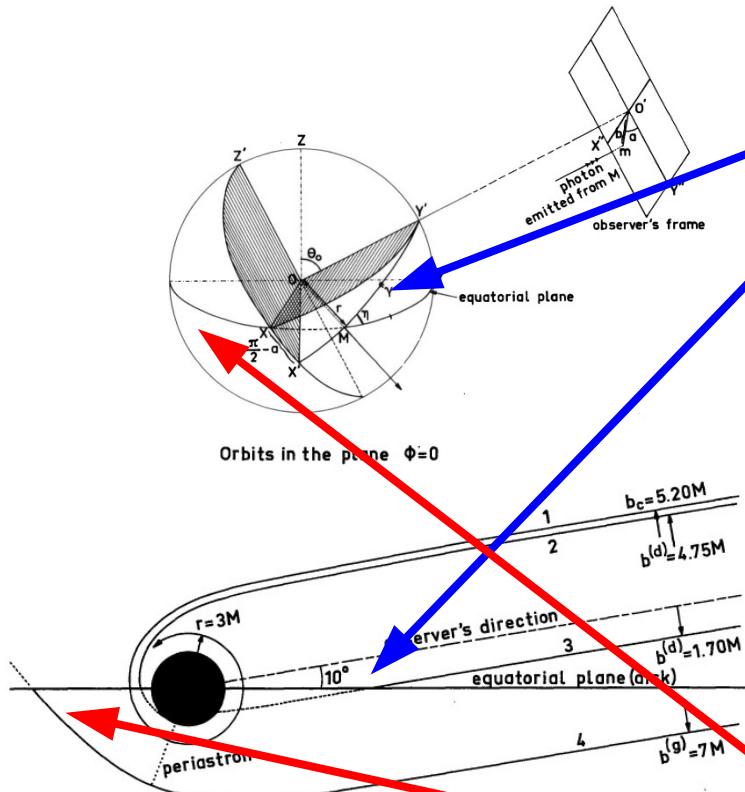
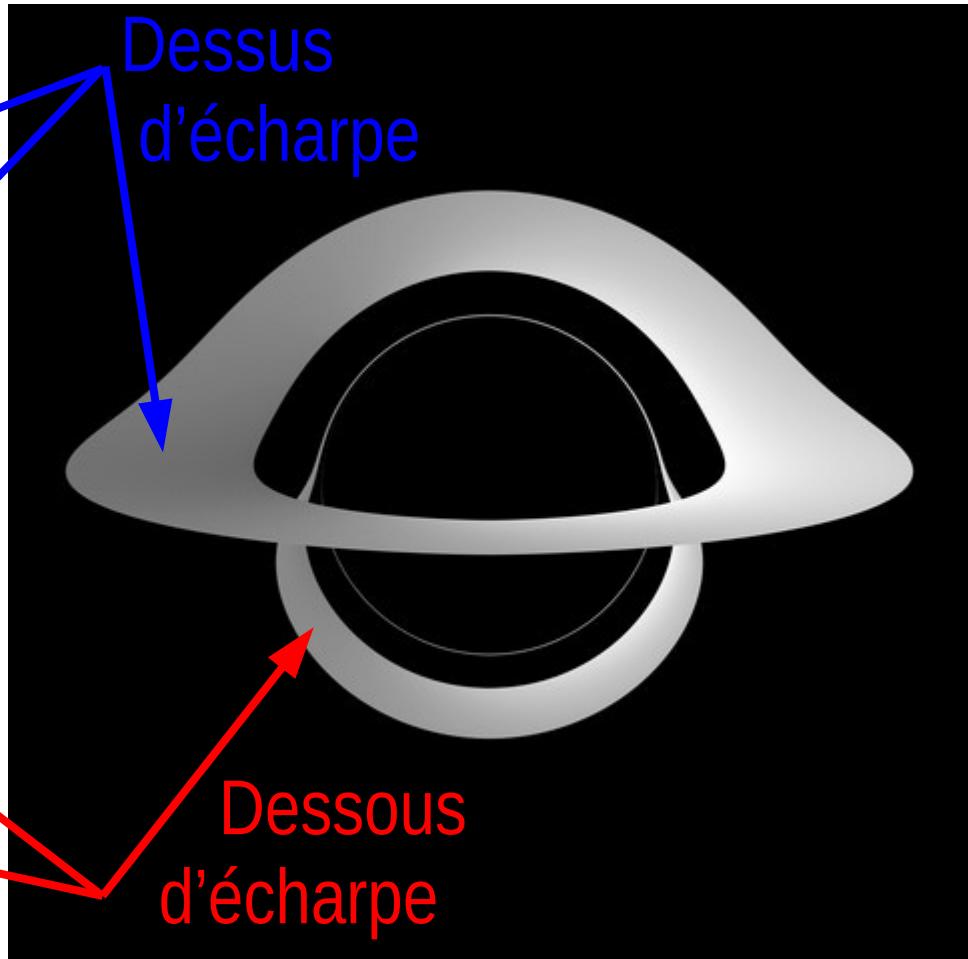


Fig. 4. Illustrative orbits in the plane  $\{\phi=0\}$ . Trajectory 1 has the critical impact parameter and circles infinitely around the black hole; trajectories 2 and 3 give direct images, trajectory 4 gives a secondary image

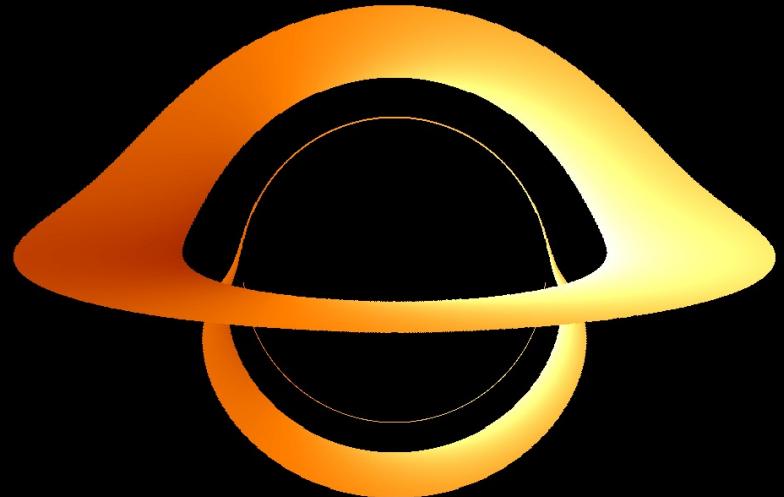


# Calculer les trajectoires inverses

## Entre Newton et Einstein

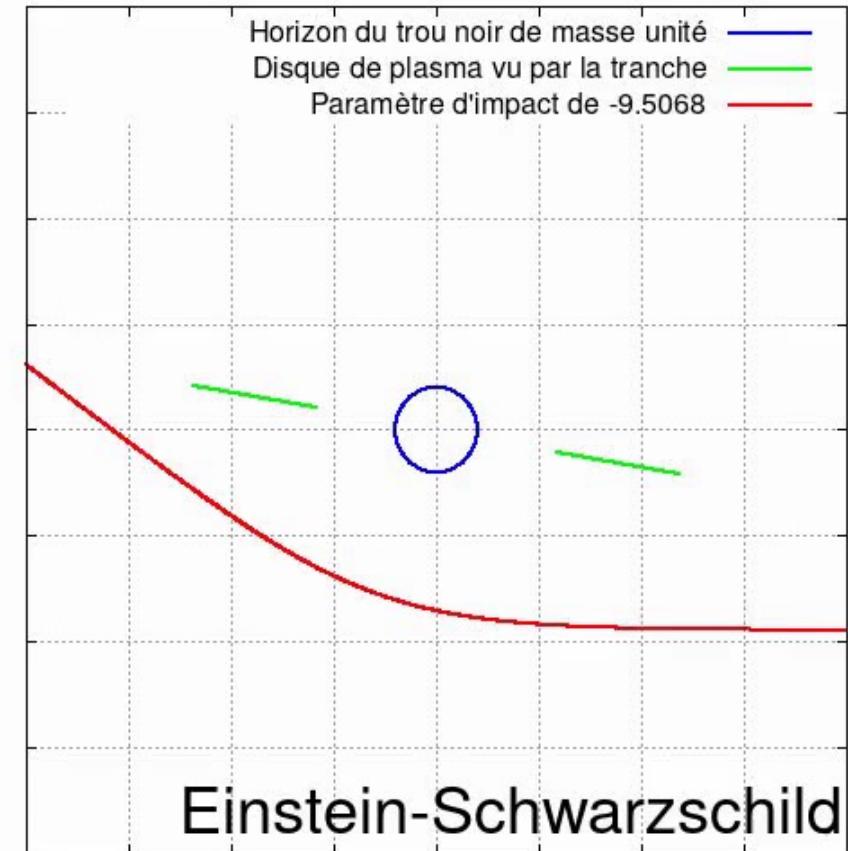
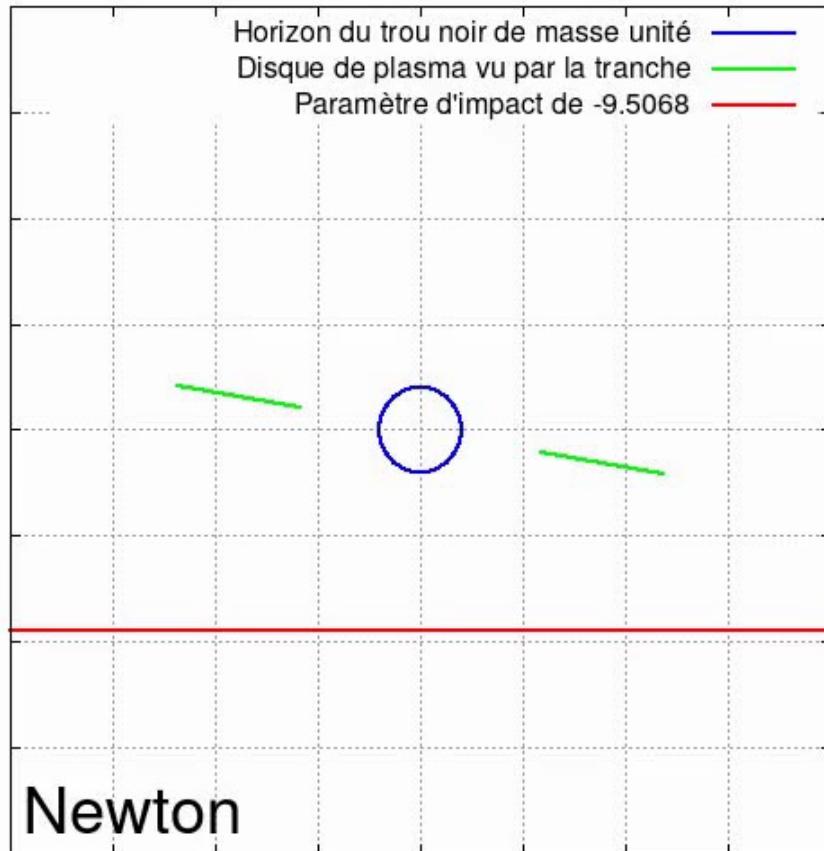


Newton



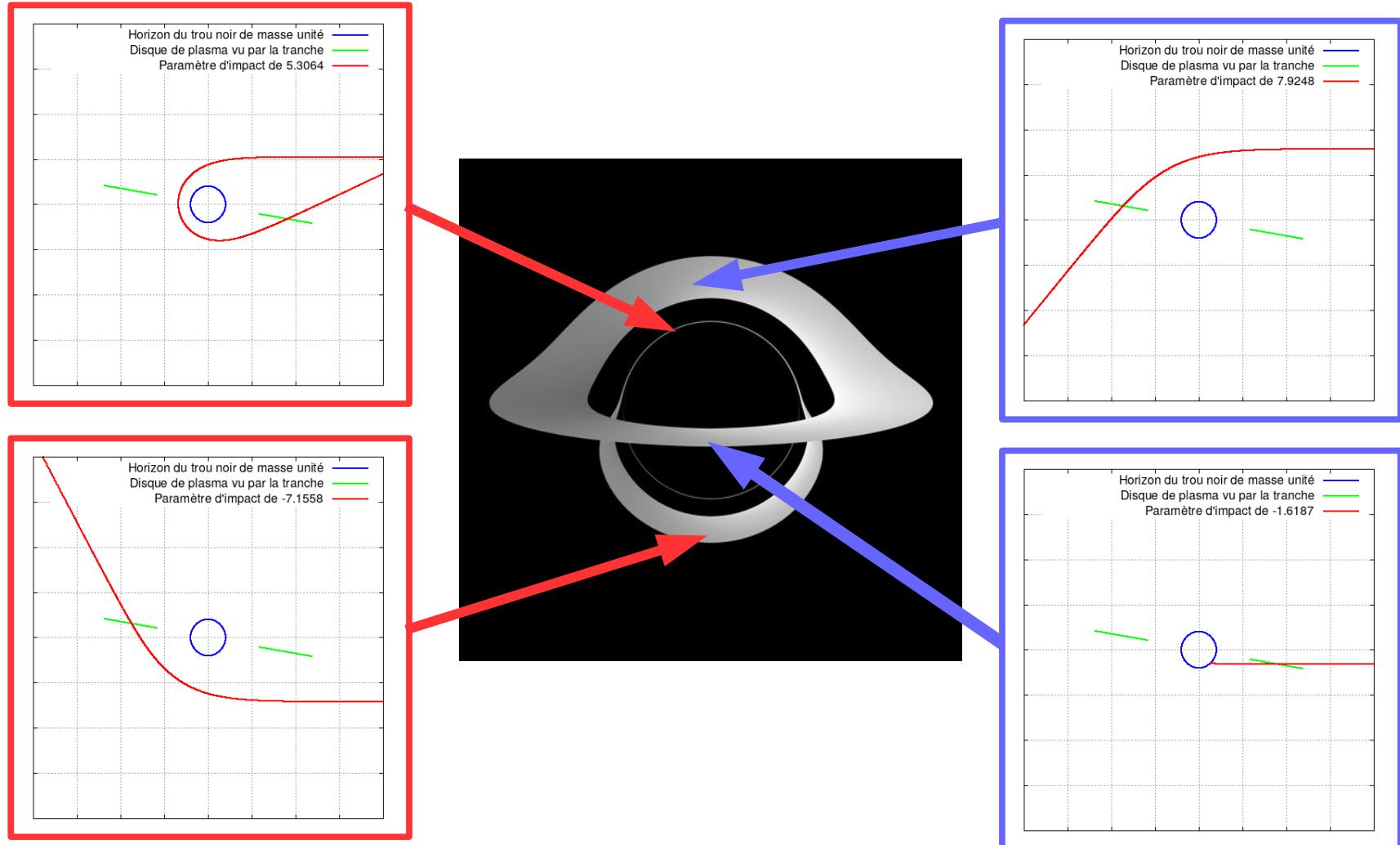
Einstein

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

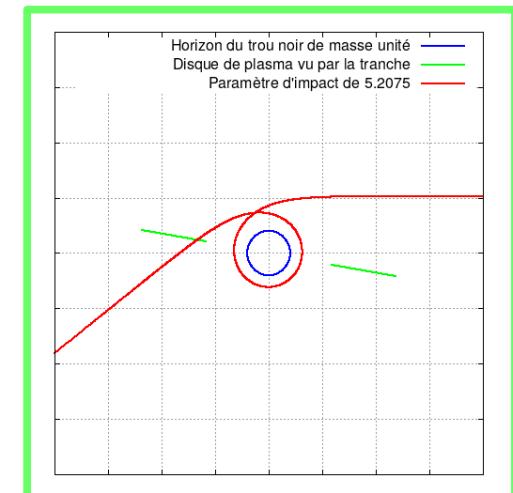
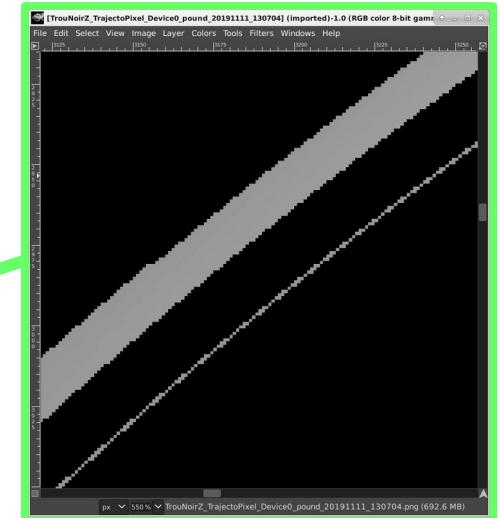
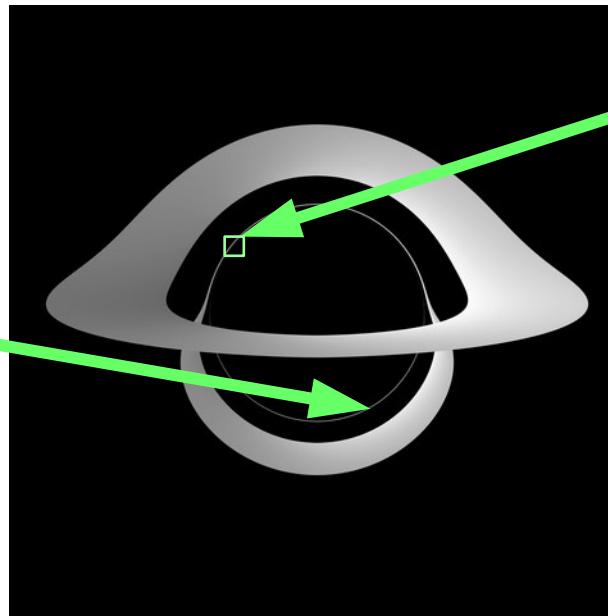
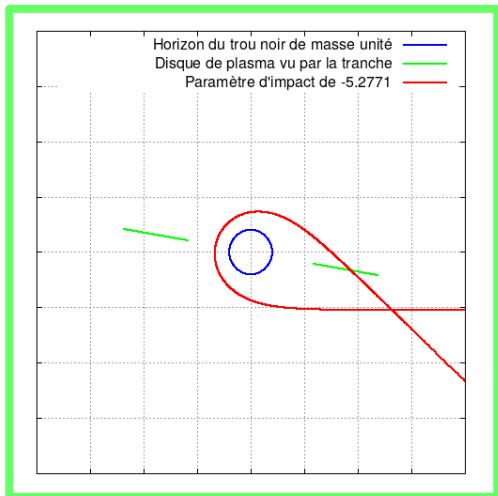


Copyleft James MYLQ 2020

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



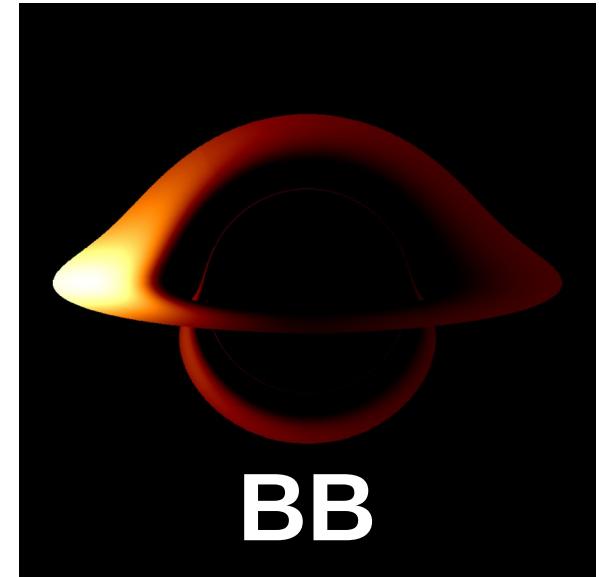
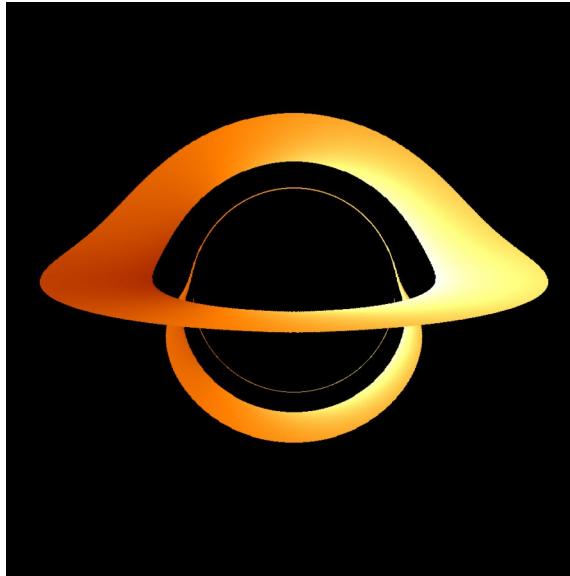
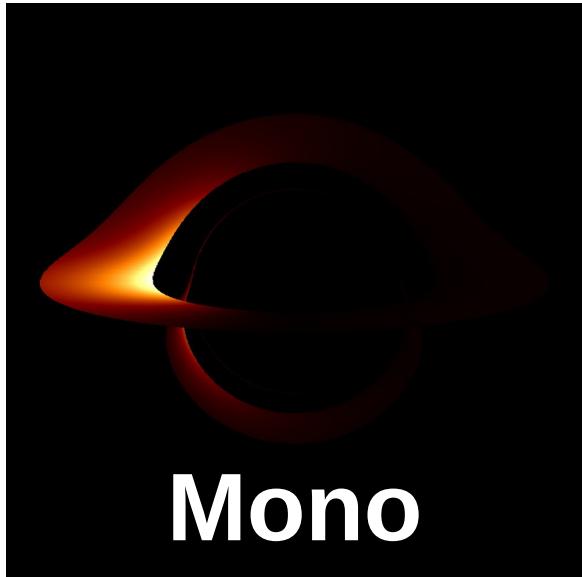
# Quelques cas particuliers : le redessus, dessus et dessous



# A chaque impact, deux physiques Monochrome & Corps noir

« Puissance 4 » sur z

Spectre de planck



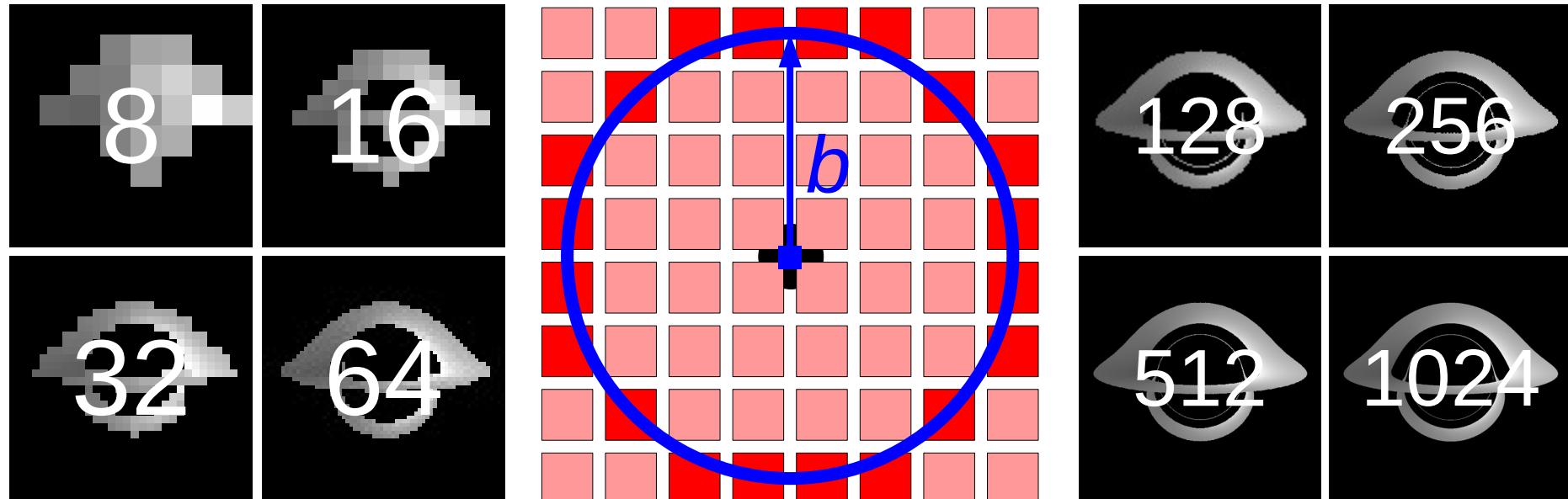
# La méthode : « lancer des rayons » de l'oeil au disque de plasma

- Pour chaque pixel de l'image
  - Calculer la trajectoire (résoudre le système d'équations)
  - Regarder si le photon intercepte le disque
    - Si la distance du photon inférieure au rayon de Schwarzschild
      - Dommage... (en même temps, c'est le principe du « trou noir »)
    - Si le photon traverse le plan du disque entre ses rayons intérieur & extérieur
      - 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
- Méthode systématique mais très coûteuse :
  - Aucun exploitation de la symétrie du problème physique

# 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  $\sim \# \text{pixels}$ 
  - 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));
}
```



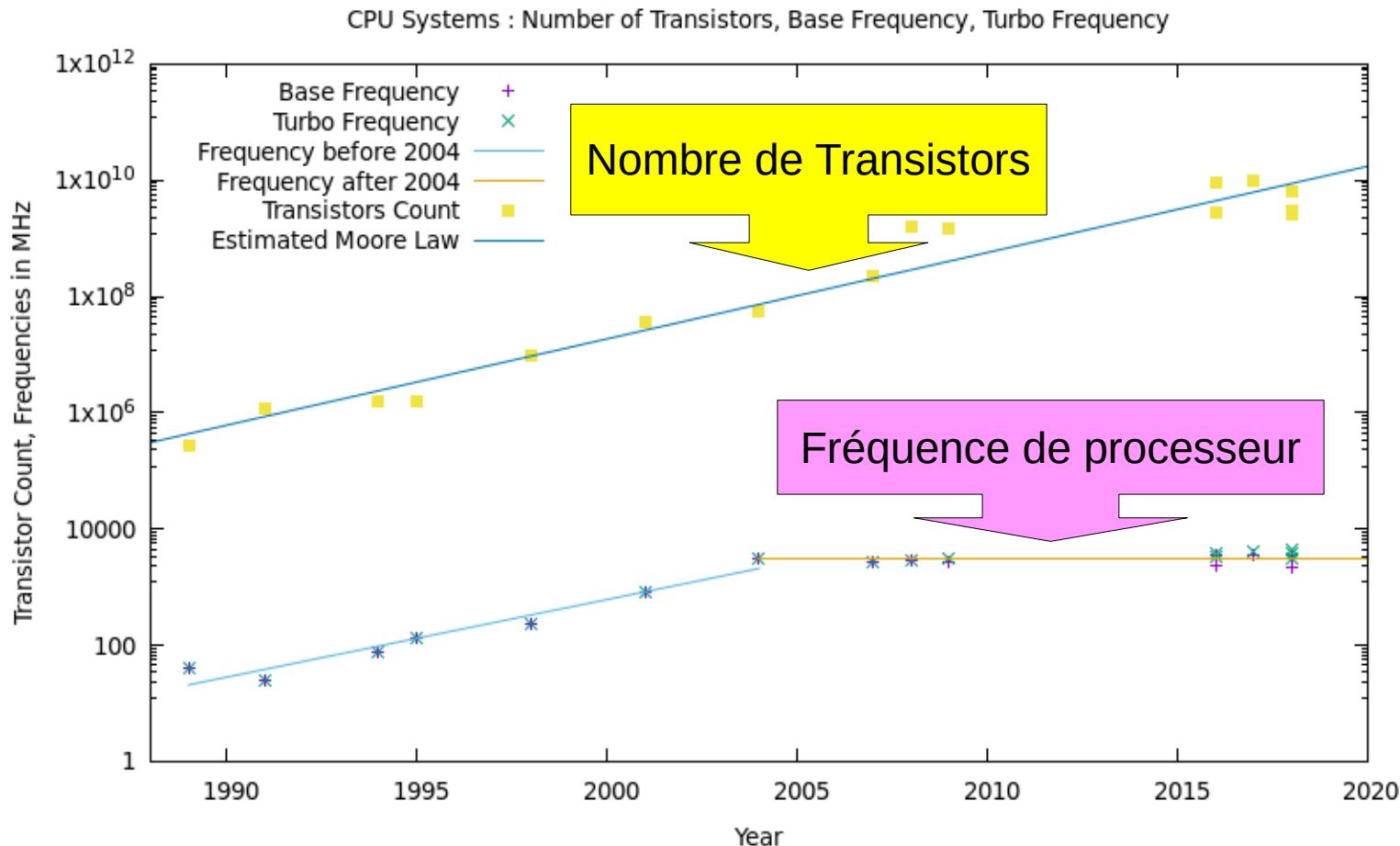
# Le banc d'essai : échantillon 30 ans d'évolutions technologiques

- La fréquence : de 40 MHz à 3.6 GHz
- Ces 30 dernières années : 5 (r)évolutions
  - Intégration systématique du **FPU** dans les processeurs : #1
    - Avant : 80386SX à 40 MHz (1989) et 80486SX à 25 MHz (1991)
    - Après : Overdrive DX4 à 75 MHz (1994) et Amd5x86 (1995)
  - Exploitation interne de **RISC86** et intégration **unité vectorielle** : #2 et #3
    - AMD K6-2 à 233 MHz avec unité 3DNow
  - **Multiplication** des coeurs (d'abord virtuel) : #4
    - Premiers : Pentium 4 Northwood à 3 GHz avec HyperThreading et AthlonX2 à 2.6 GHz
    - Systèmes avec HarperTown, Nehalem, Broadwell, Skylake, Threadripper : 8 à 28 coeurs
  - Détournement de l'usage des **GPU** : #5
    - De la Tesla C0160 à la Tesla V100

# Banc de test sur les 16 CPUs

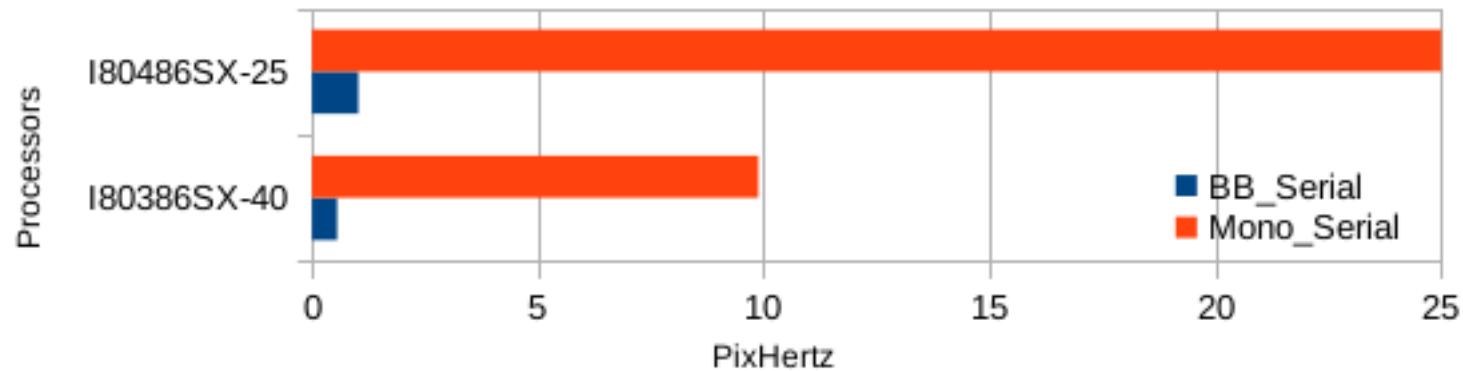
- Les processeurs et leurs distributions :
  - 80386SX, 80486SX, Overdrive DX4, Amd5x86 : Debian Buzz & Hamm
  - K7, Northwood, AthlonX2 : Debian Stretch
  - E5440x2, X5550x2, E5-2637v4x2, E5-2680v4, Gold5122, Silver4144, W-2145 : Debian Stretch
  - Threadripper 1950X : Ubuntu 18.10
- Images de 64x64 à 16384x16384 pixels :  $2^6$  à 2
  - 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 »

# Distribution de CPU pertinente ? Transistors & Fréquences...



Doublement des transistors tous les 2 ans...

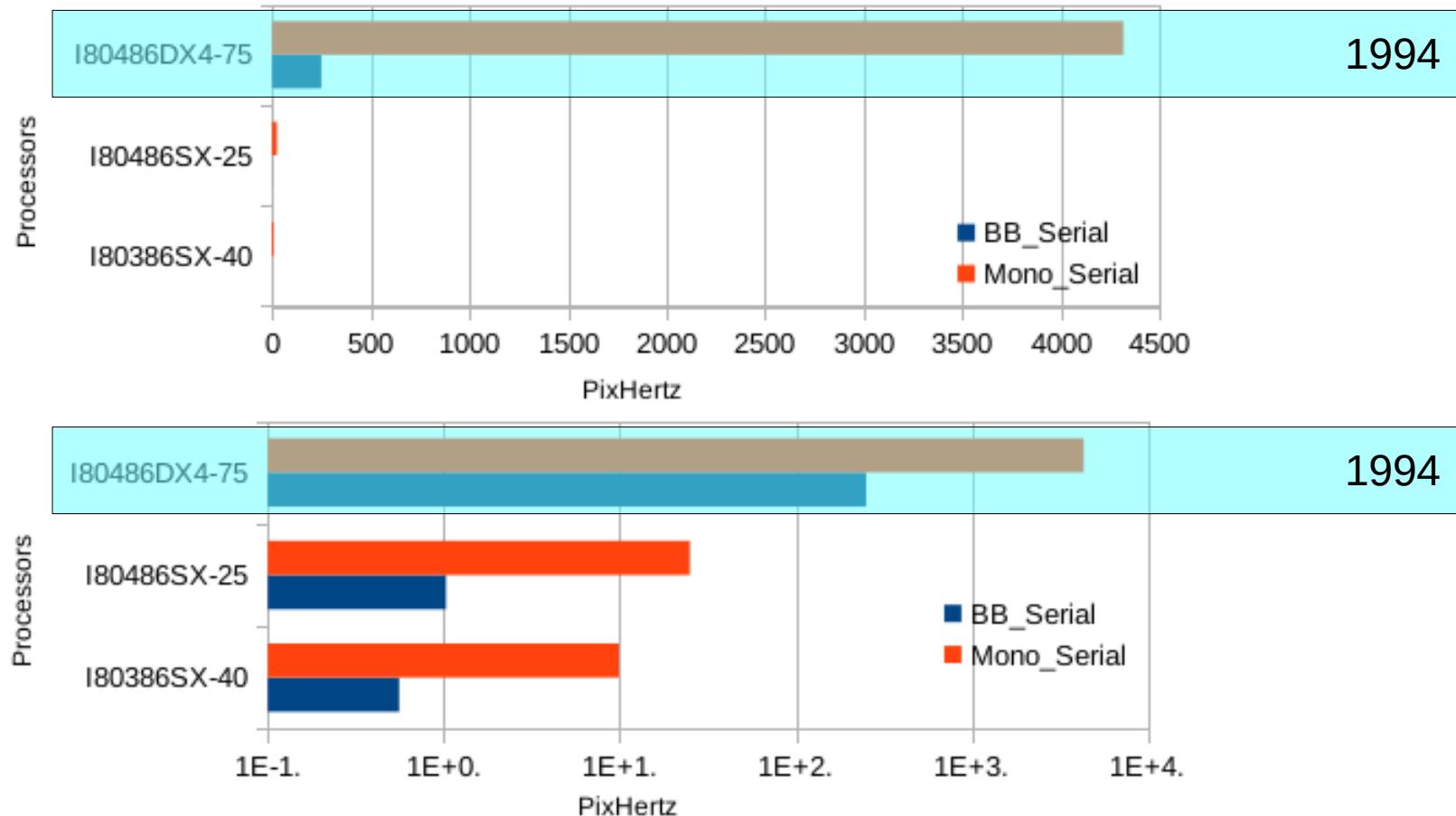
# L'ère préFPU... Le 80386SX et le 80486SX



La fréquence ne fait pas tout... Sous le PixHertz en BB

# Le FPU là, les fréquences grimpent

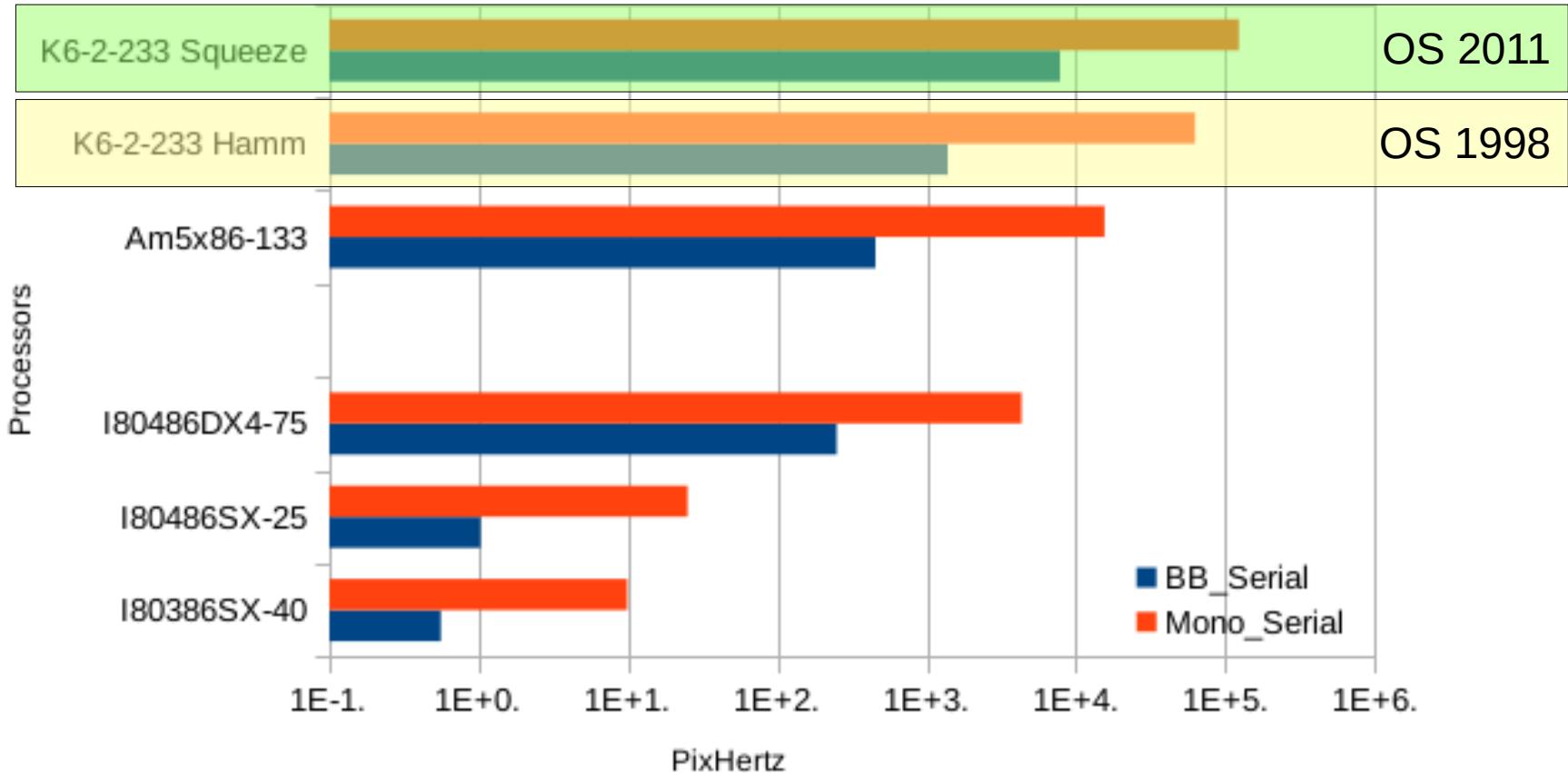
## #1 : Le 80486DX4 à 75 MHz



x240 en performance : « des minutes en secondes »

# Le RISC & la vectorisation

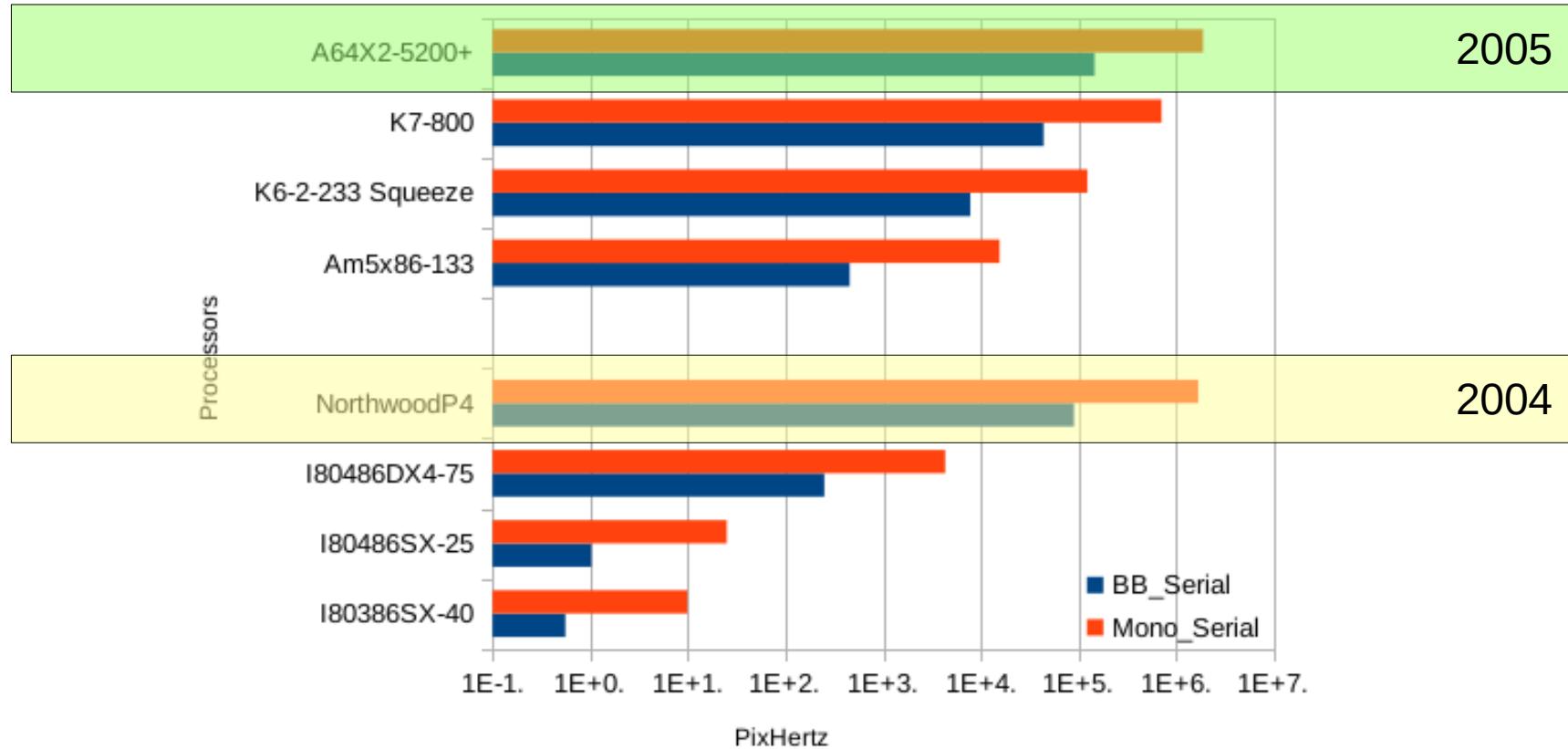
## #2 & #3 : le AMD K6-2 à 233 MHz



L'optimisation vient surtout 15 ans plus tard...

# Le multicoeurs : logique & physique

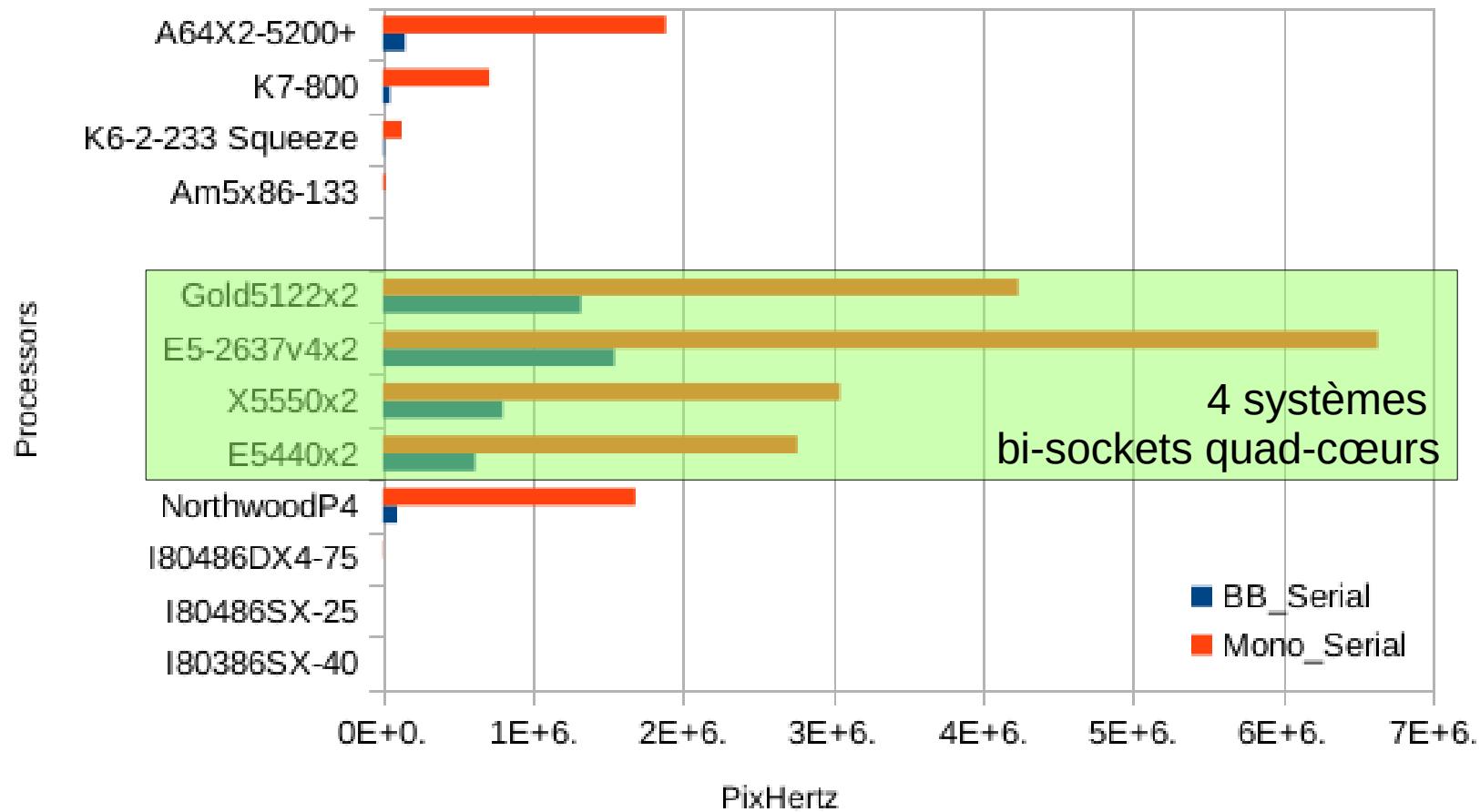
## #4 : le Northwood Intel, l'Athlon64x2



Un facteur 10 à 15 entre en 8 ans

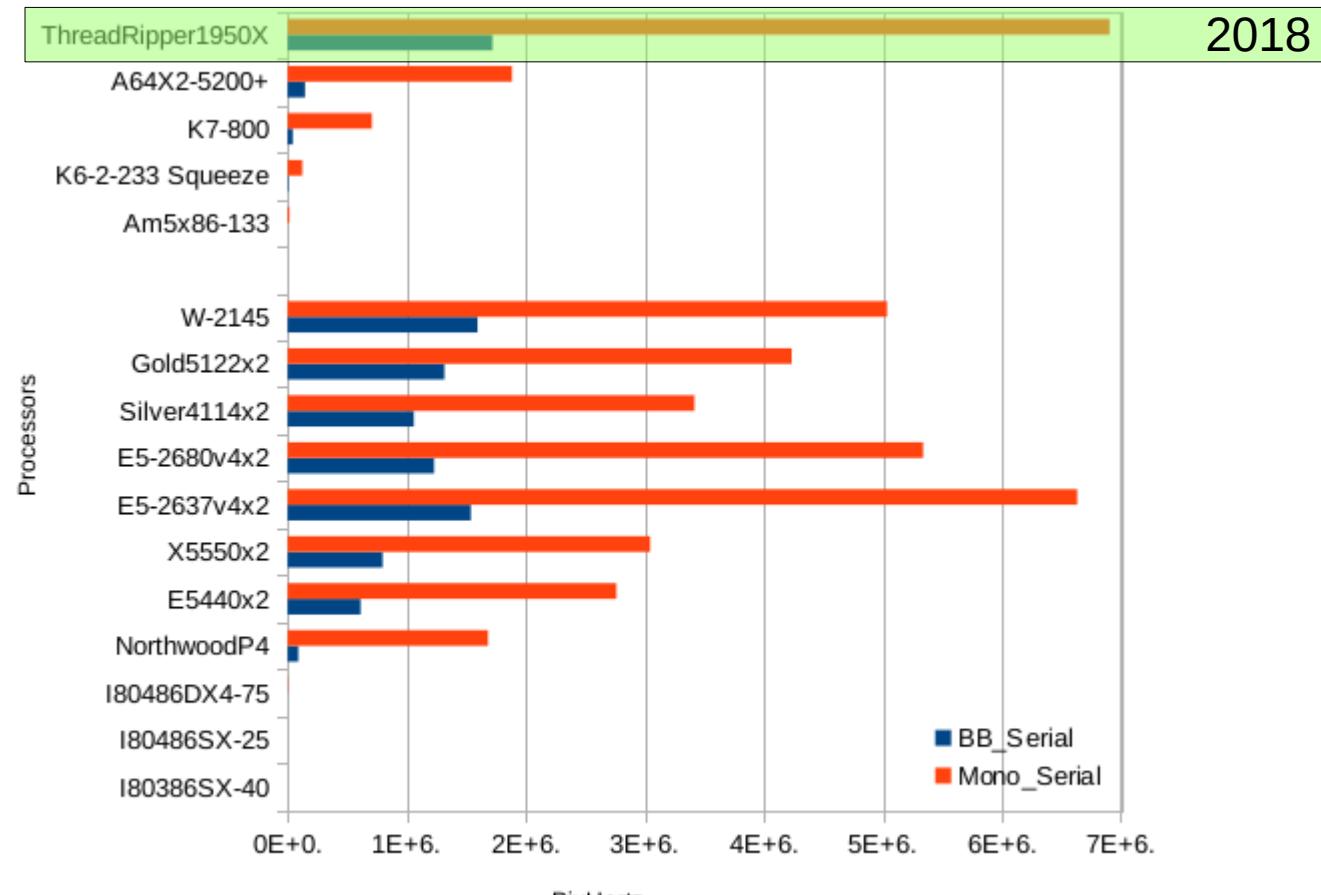
Programme séquentiel, pas d'exploitation des coeurs...

# 4 générations de bi-quad-cores de 2009 à 2019



Un facteur 3 en 15 ans : peut mieux faire !

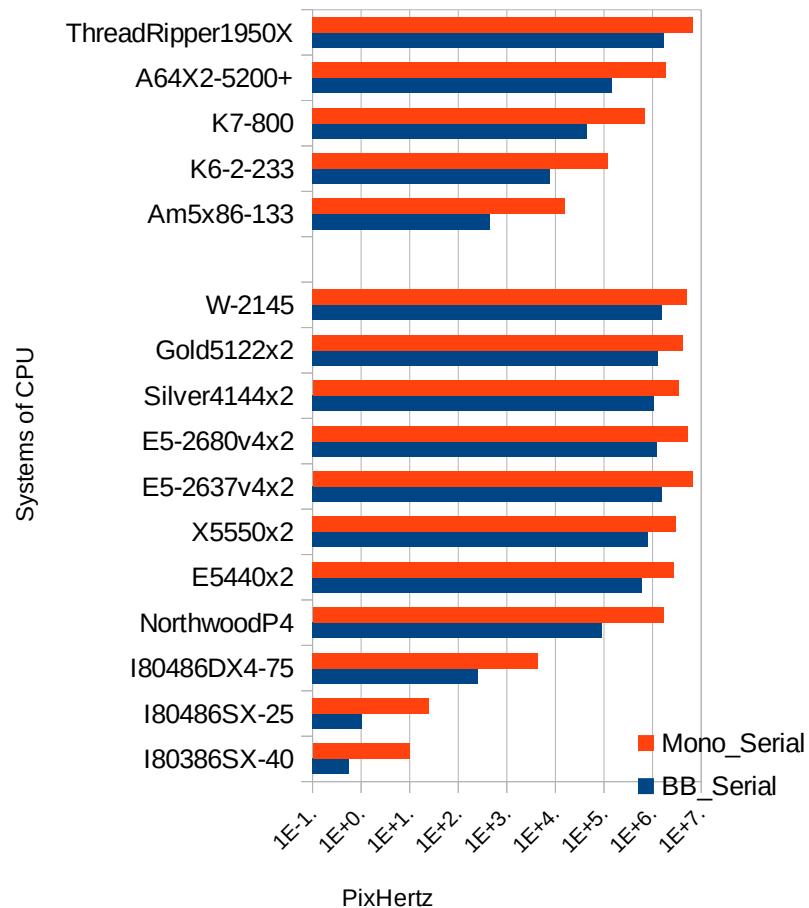
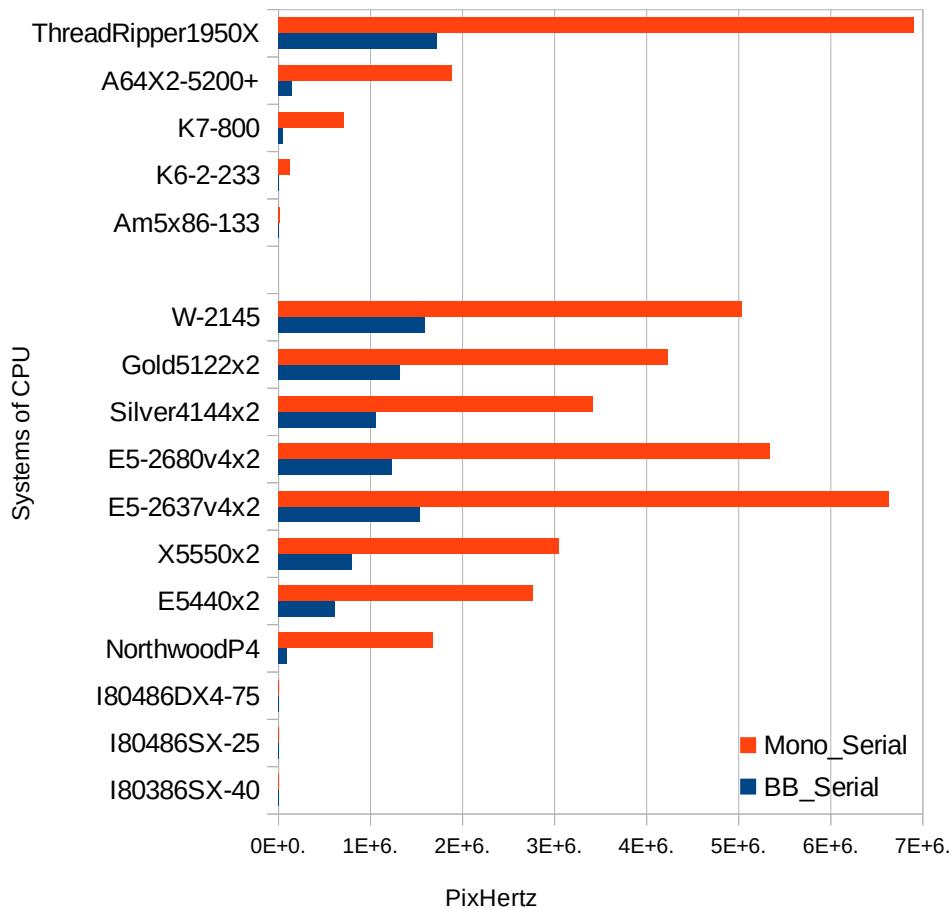
# 16 processeurs de 1989 à 2019 : « and the winner is »...



Le AMD Threadripper 1950X à 3.6 GHz

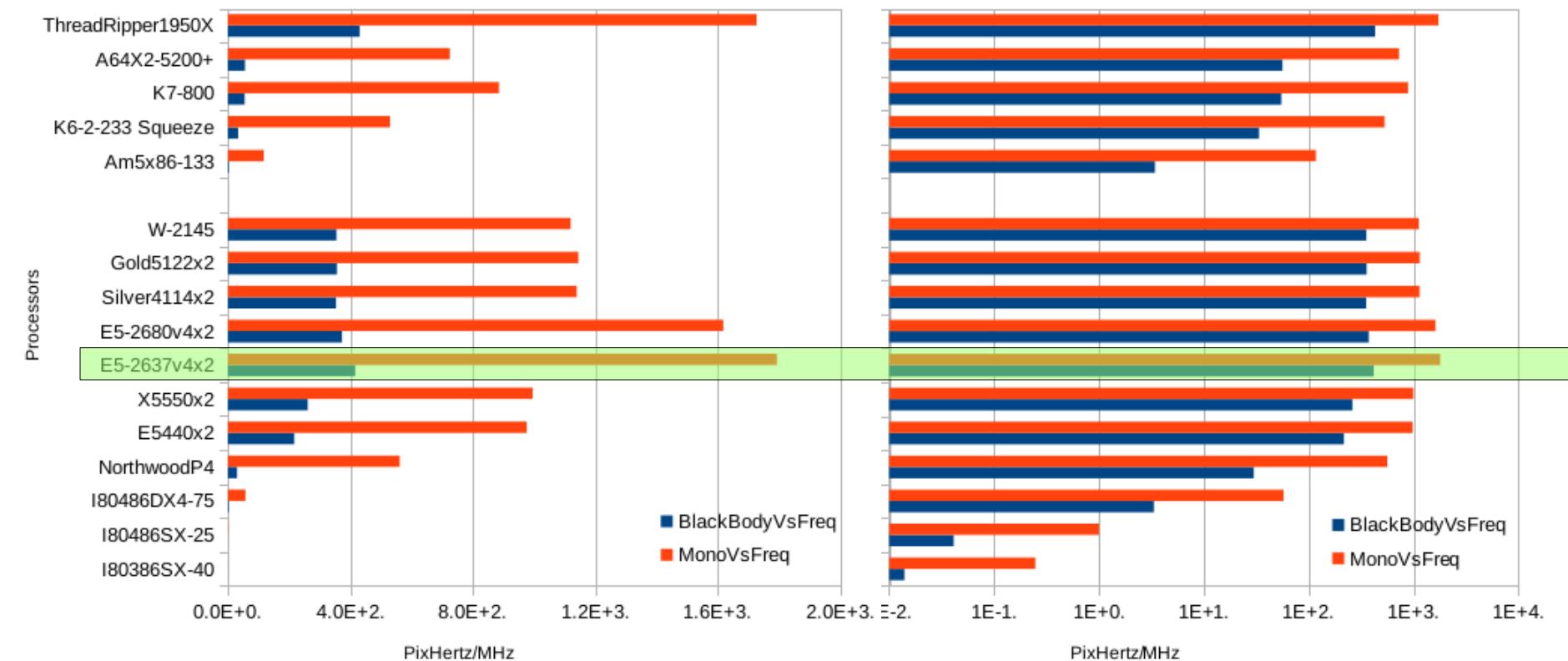
# Exécution du code séquentiel

## 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



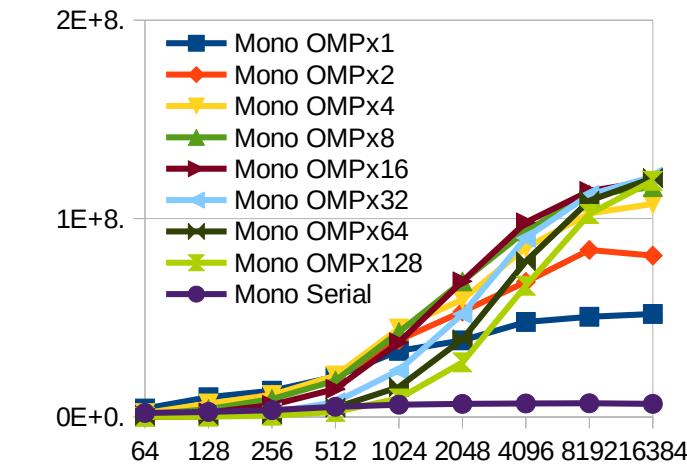
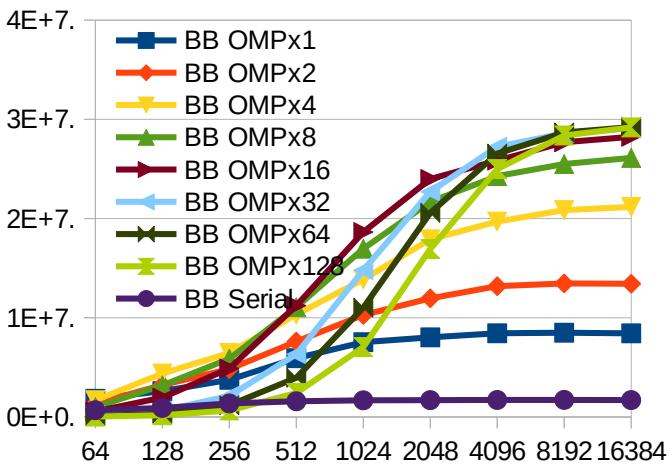
Sur 15 ans, entre un facteur 3 et un facteur 15...

Il va falloir trouver autre chose pour accélérer !

# Parallélisation du code

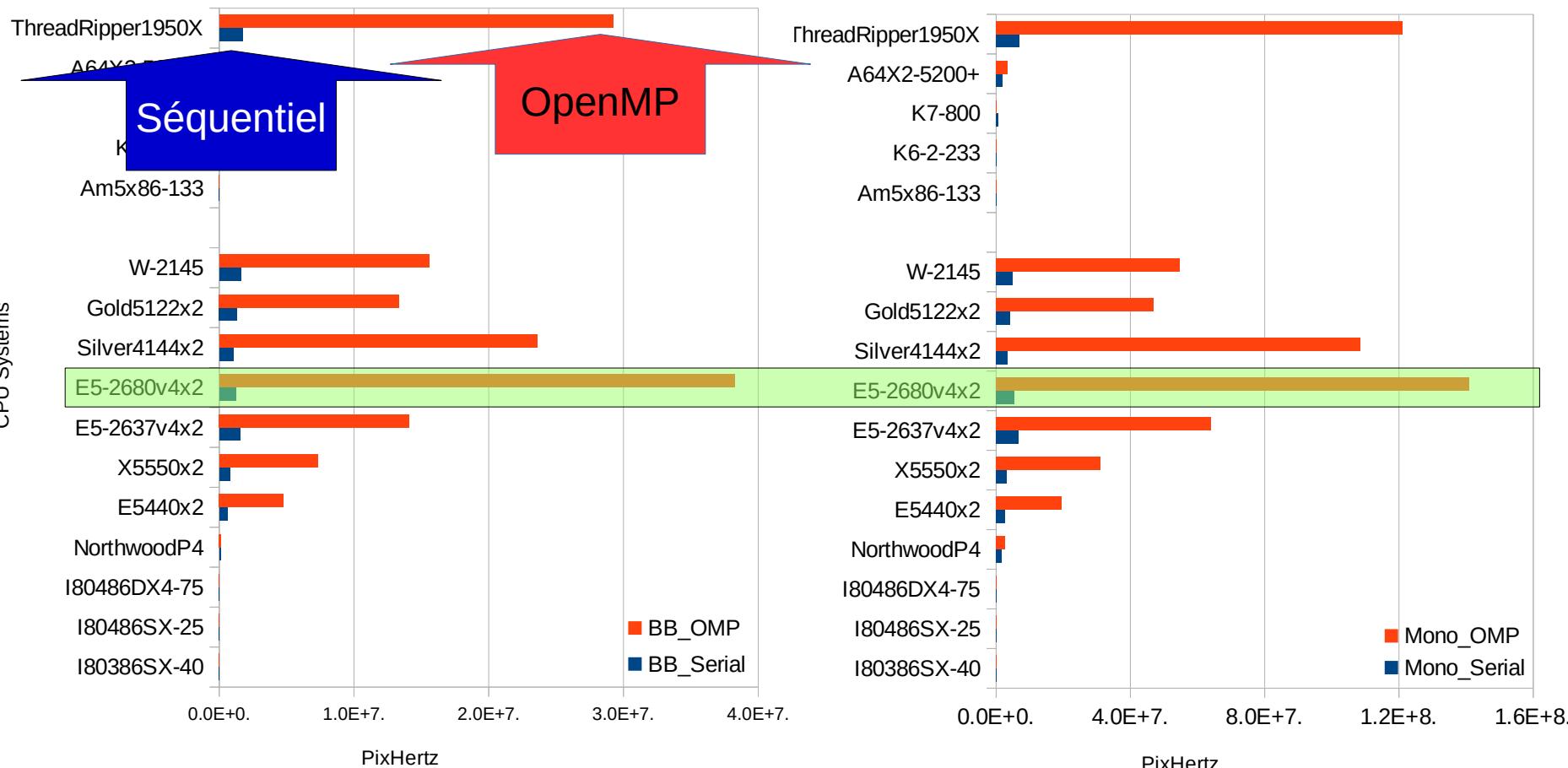
## Passage en OpenMP & étrangetés

- 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 » !



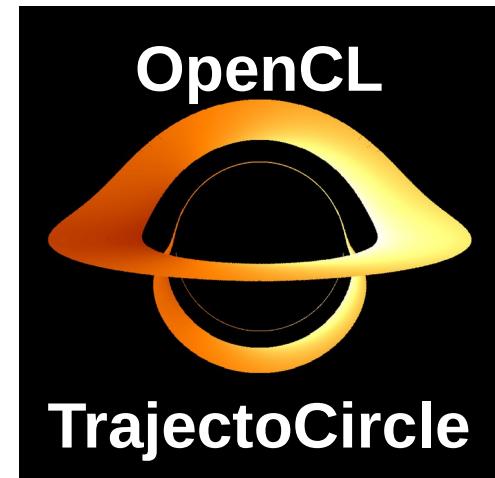
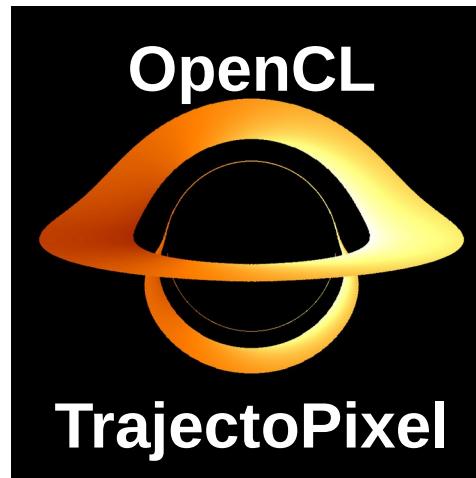
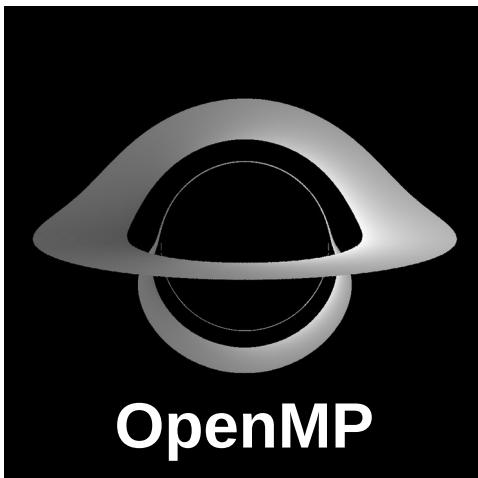
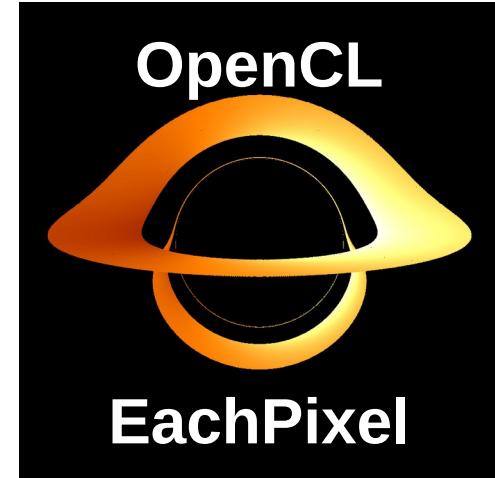
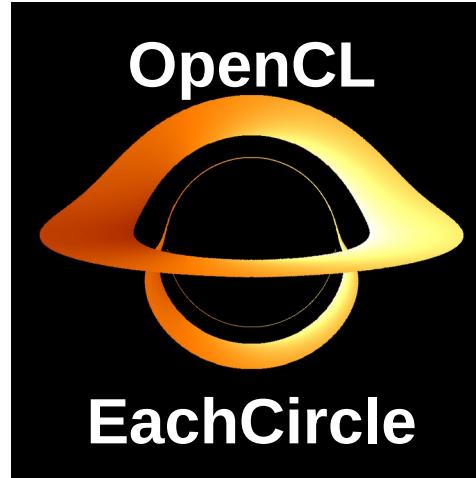
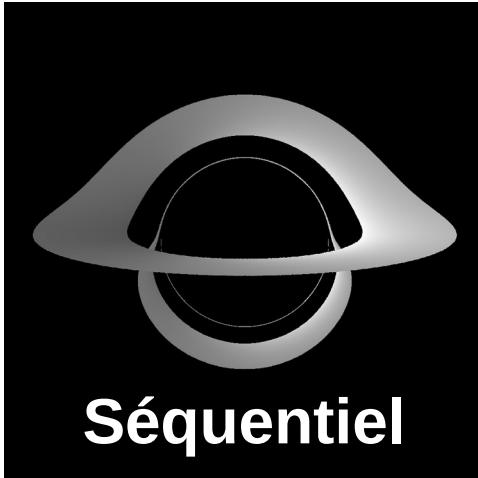
OpenCL

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

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

# Obtient-on les mêmes résultats ?

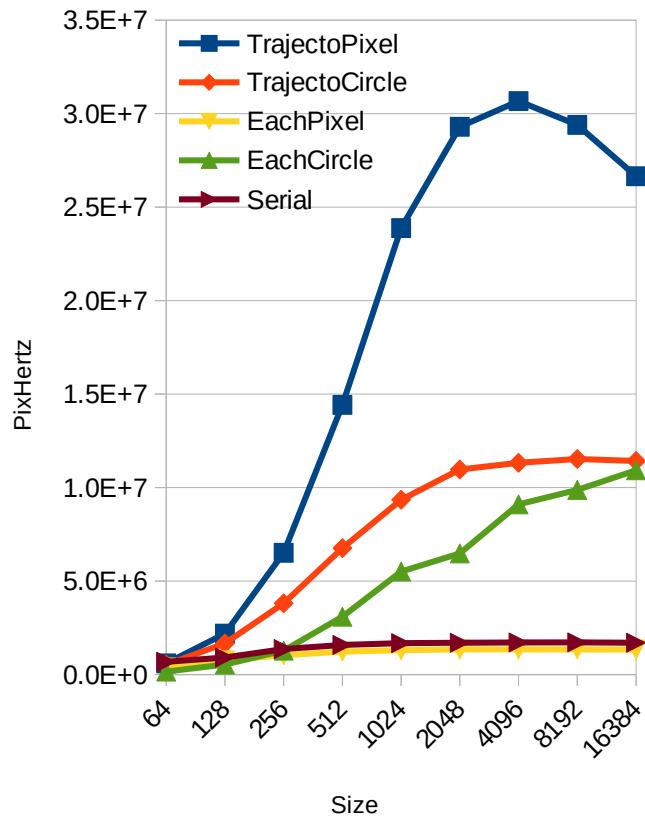
## Laissons l'œil juger...



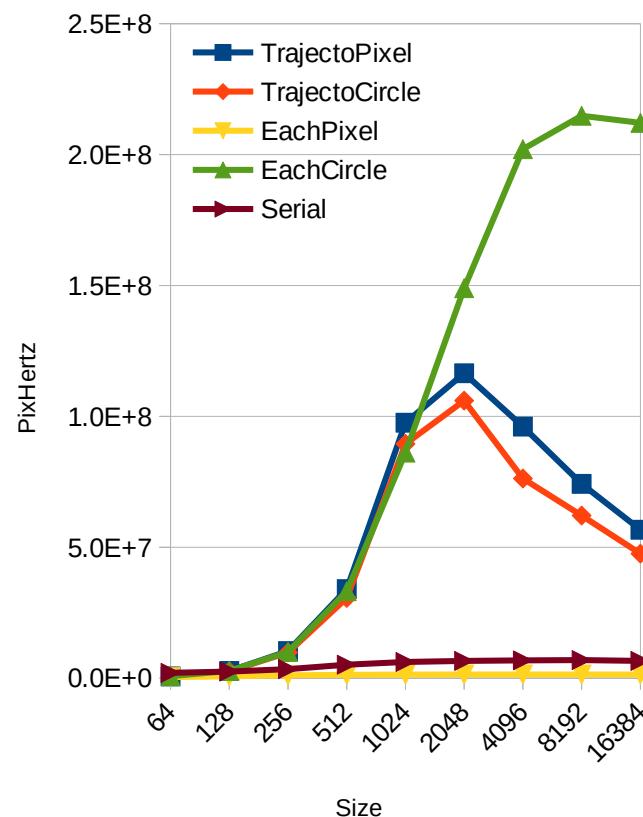
# OpenCL sur Threadripper 1950x

## La méthode de // importante...

BB on Threadripper 1950X with OpenCL



Mono on Threadripper 1950X

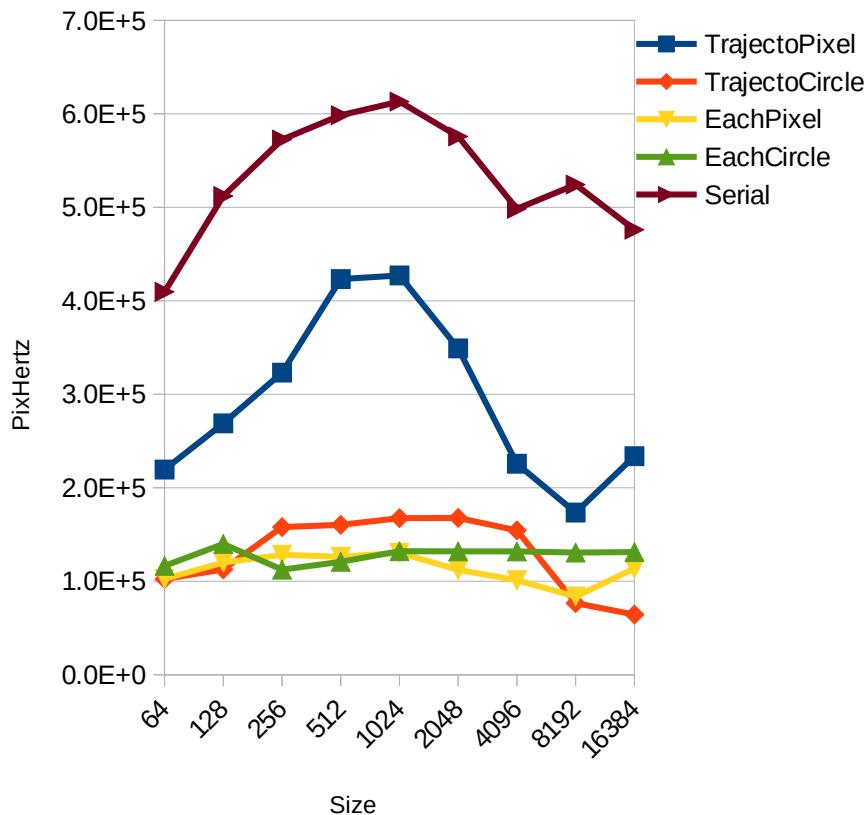


TrajetoPixel pour BB, EachCircle pour Mono...

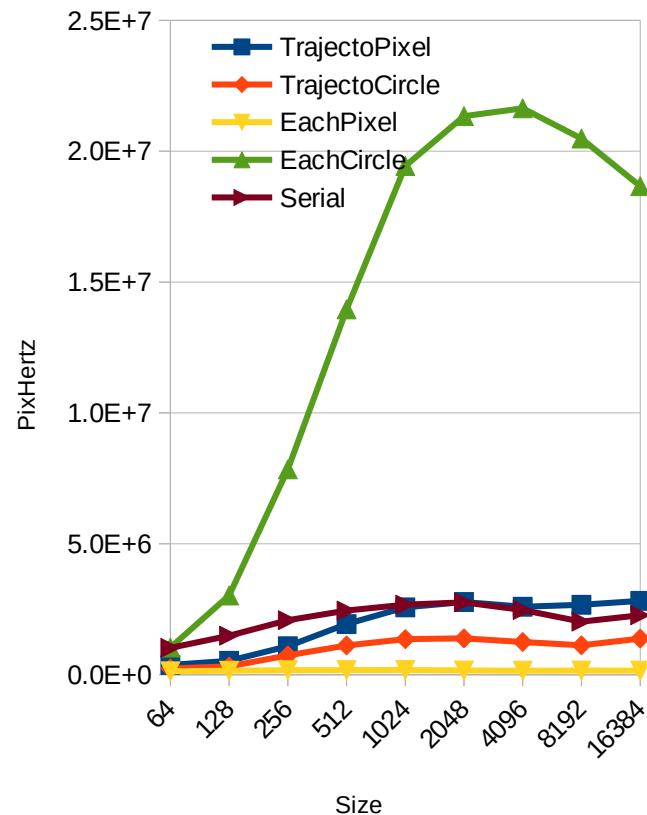
# OpenCL sur Harpertown E5440x2

## OpenCL (AMD) pas terrible

BB on Harpertown E5440 with OpenCL



Mono on Harpertown E5440 with OpenCL

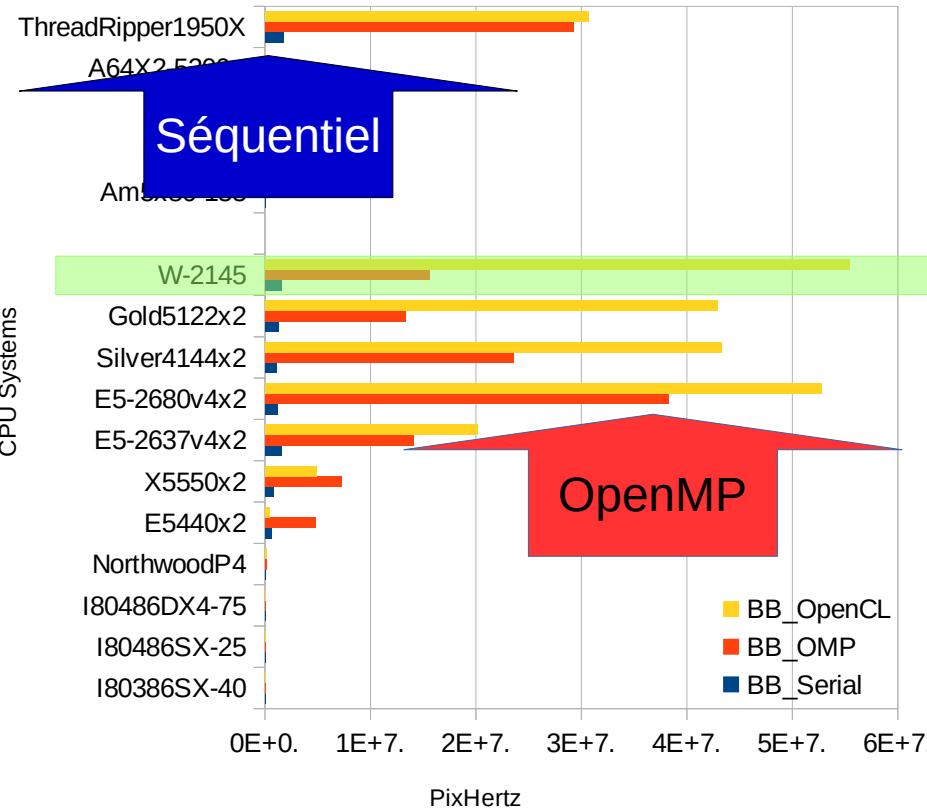


A chaque processeur, sa courbe de scalabilité...

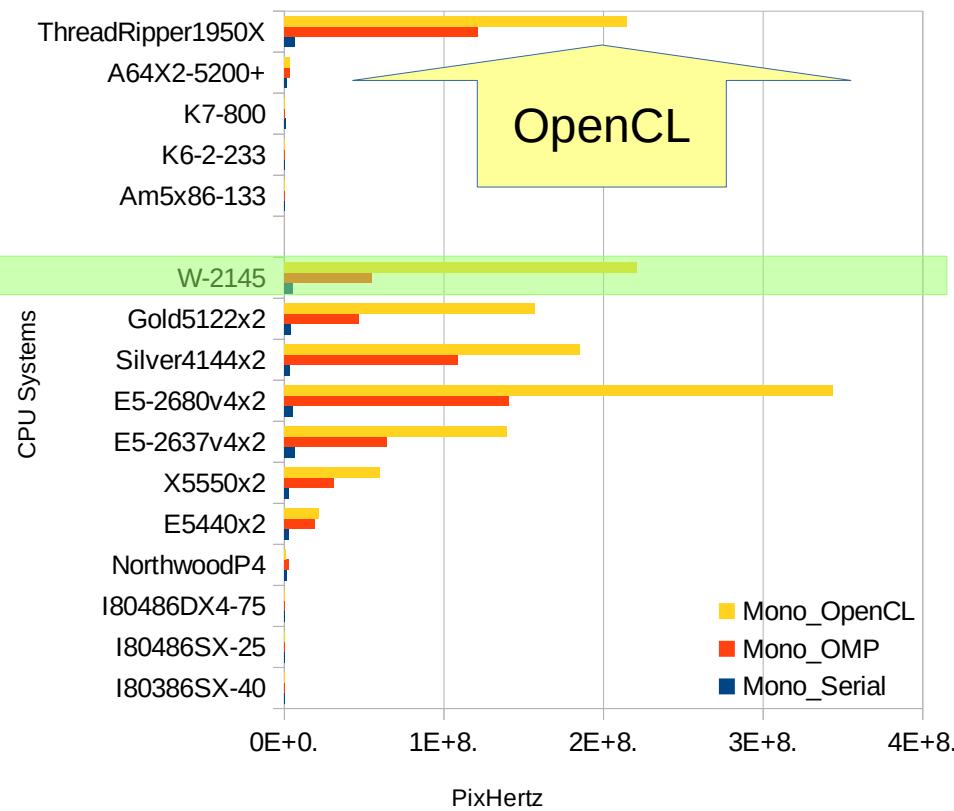
# Parallélisation OpenCL

## Pour tous les processeurs...

BB with Serial, OpenMP, OpenCL

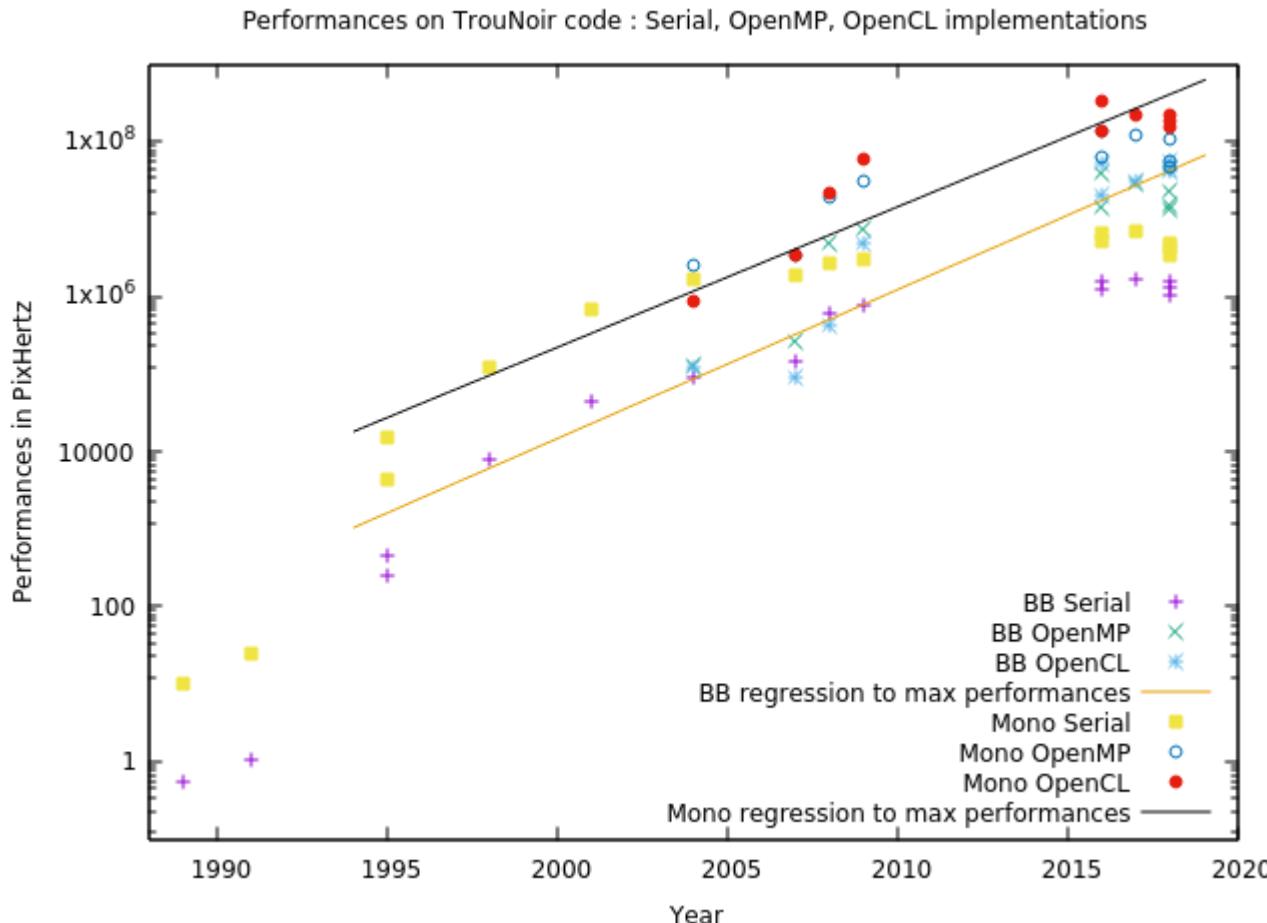


Mono with Serial, OpenMP, OpenCL



OpenCL Intel très efficace avec CPU Intel !

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



Performance : x2 tous les 18 mois !