



INSA 2021

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

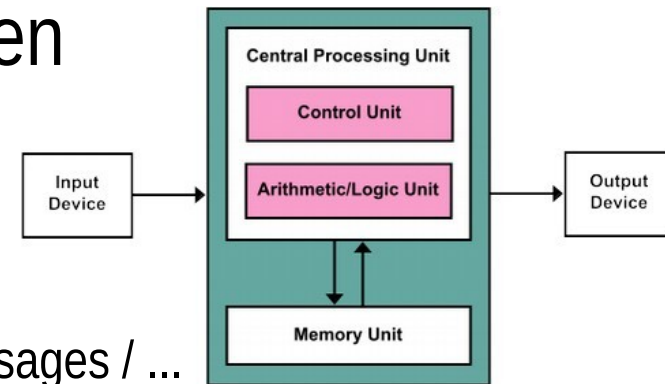
Quelle vue du contexte ?

Point de vue d'un physicien...

- Approche « système » du physicien

- Héritage des calculateurs analogiques
- Le « système » informatique :

- Réseau / matériel / OS / bibliothèques / 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/
- 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

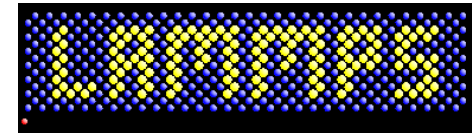
Un petit exemple illustratif



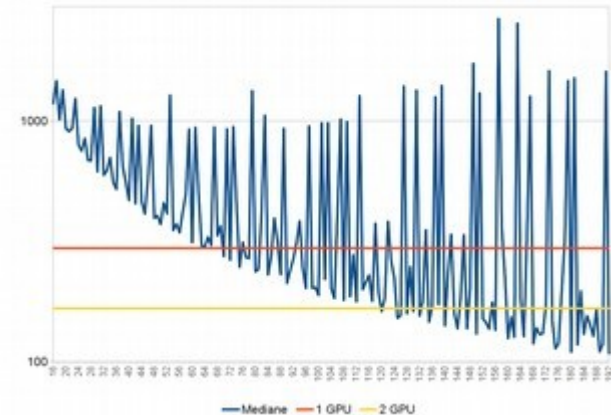
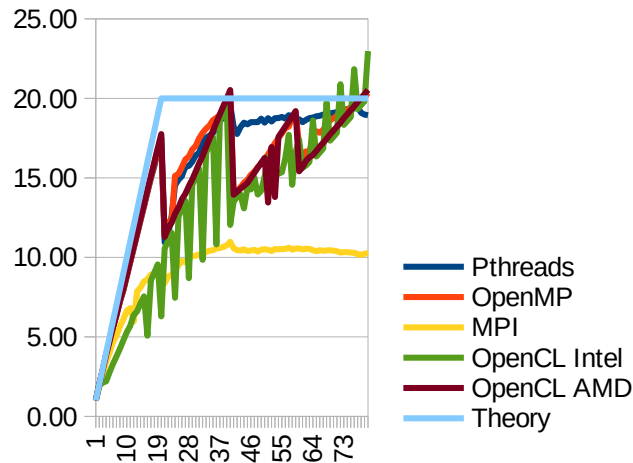
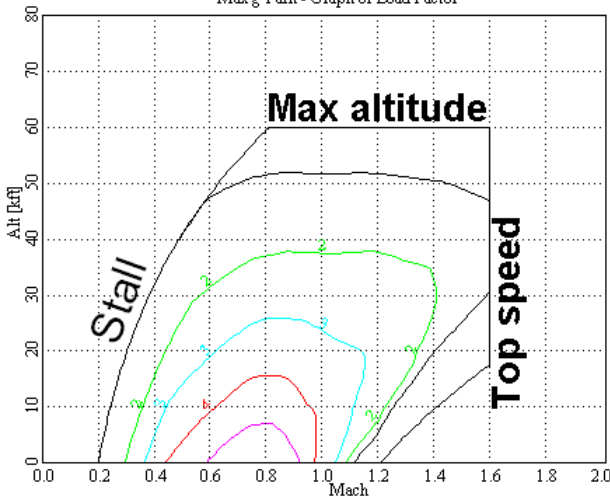
- Nasa X-29
- Cellule de F-5
- Moteur de F-18
- Train de F-16
- Etudes
 - Plans « canard »
 - Incidence $>50^\circ$
 - « Fly-By-Wire »

Recycler, réutiliser, explorer de nouveaux domaines...

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



Max g Tum - Graph of Load Factor



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



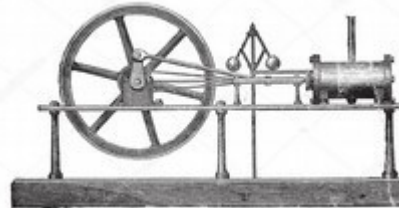
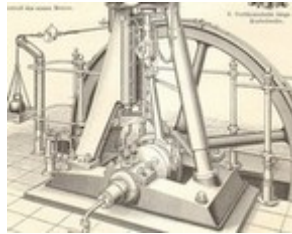
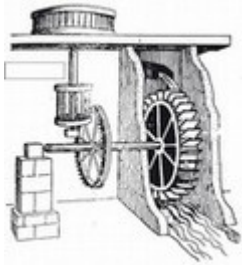
Quelques définitions & sigles...

- 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)

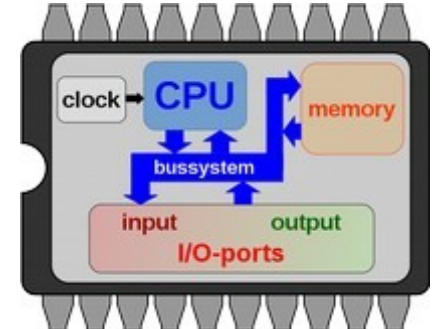
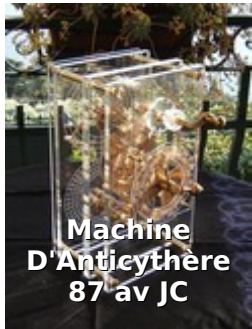
Travaux physique ou intellectuel

Moteurs & Ordinateurs :

Des auxiliaires de « puissance »

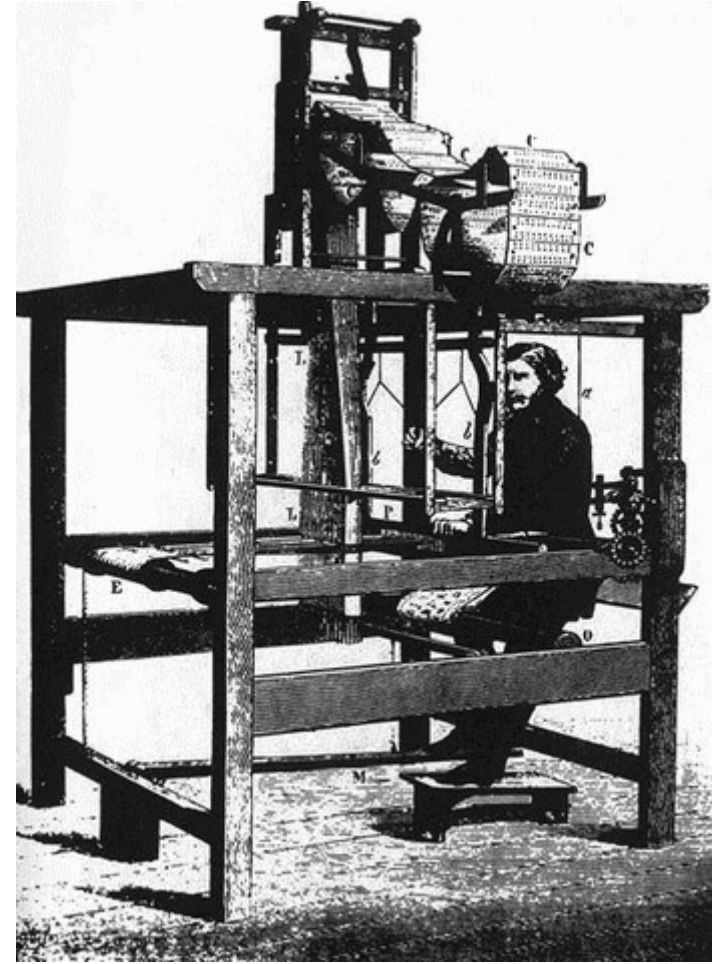
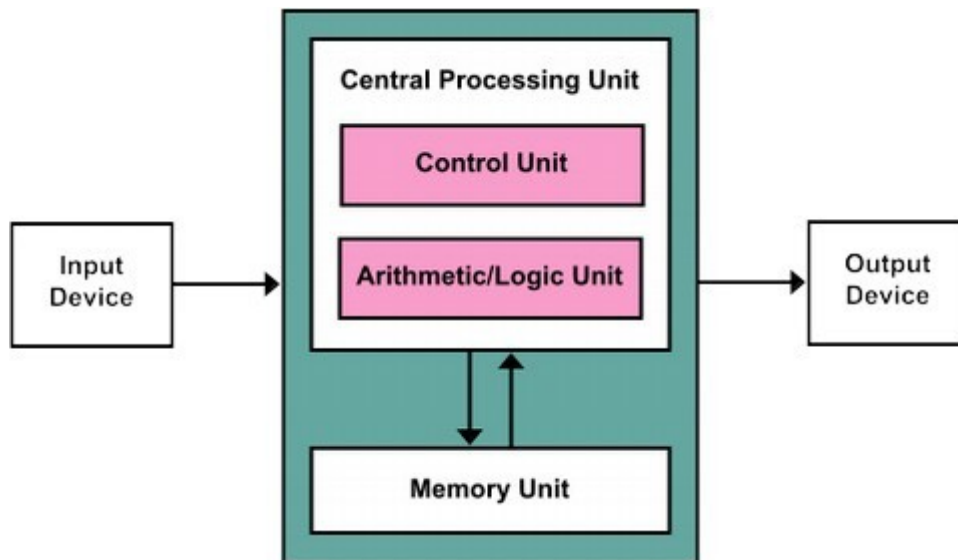


Des anciens temps aux temps modernes...

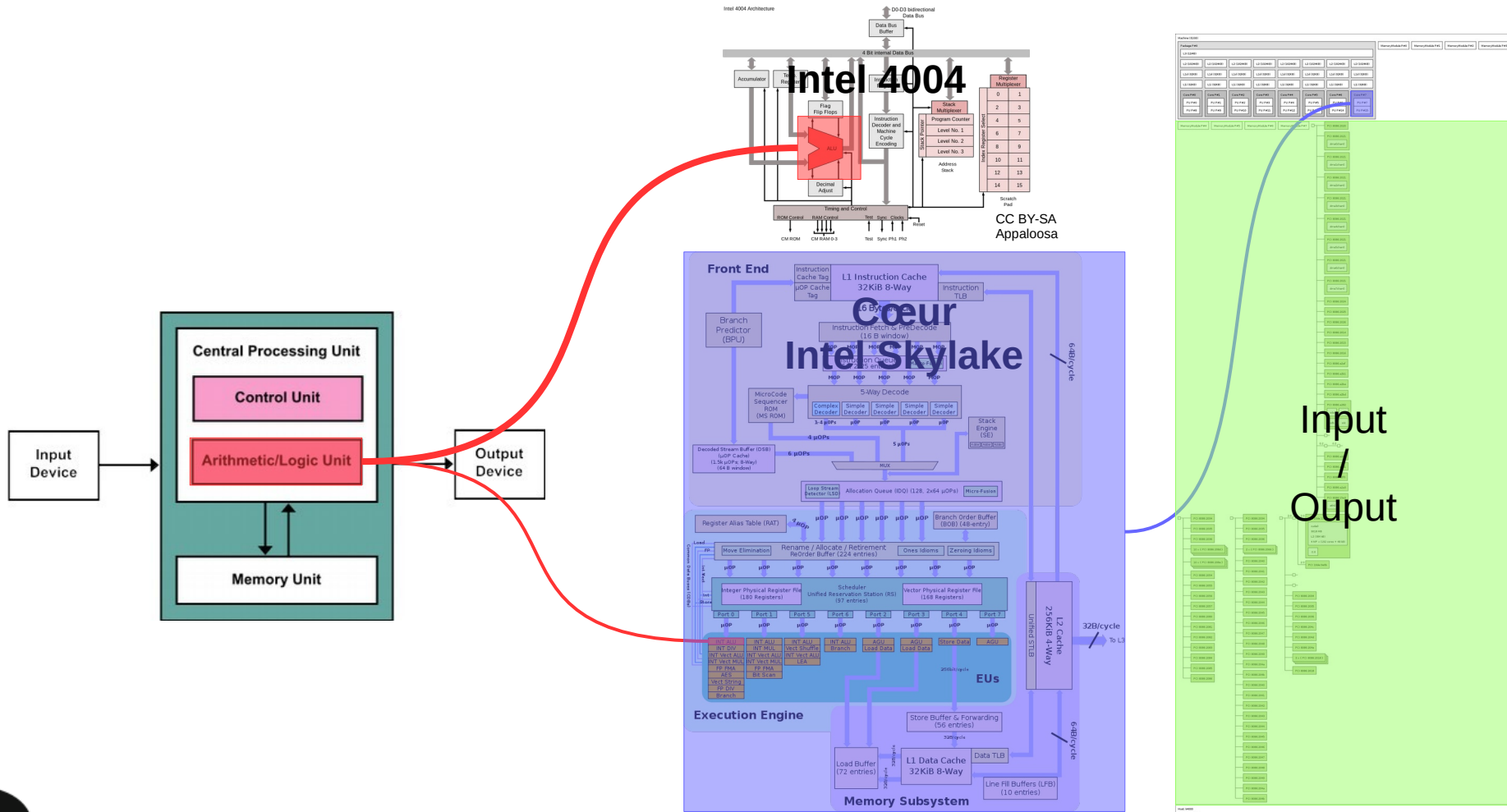


Un ordinateur : un métier à tisser ?

Architecture de Von Neumann

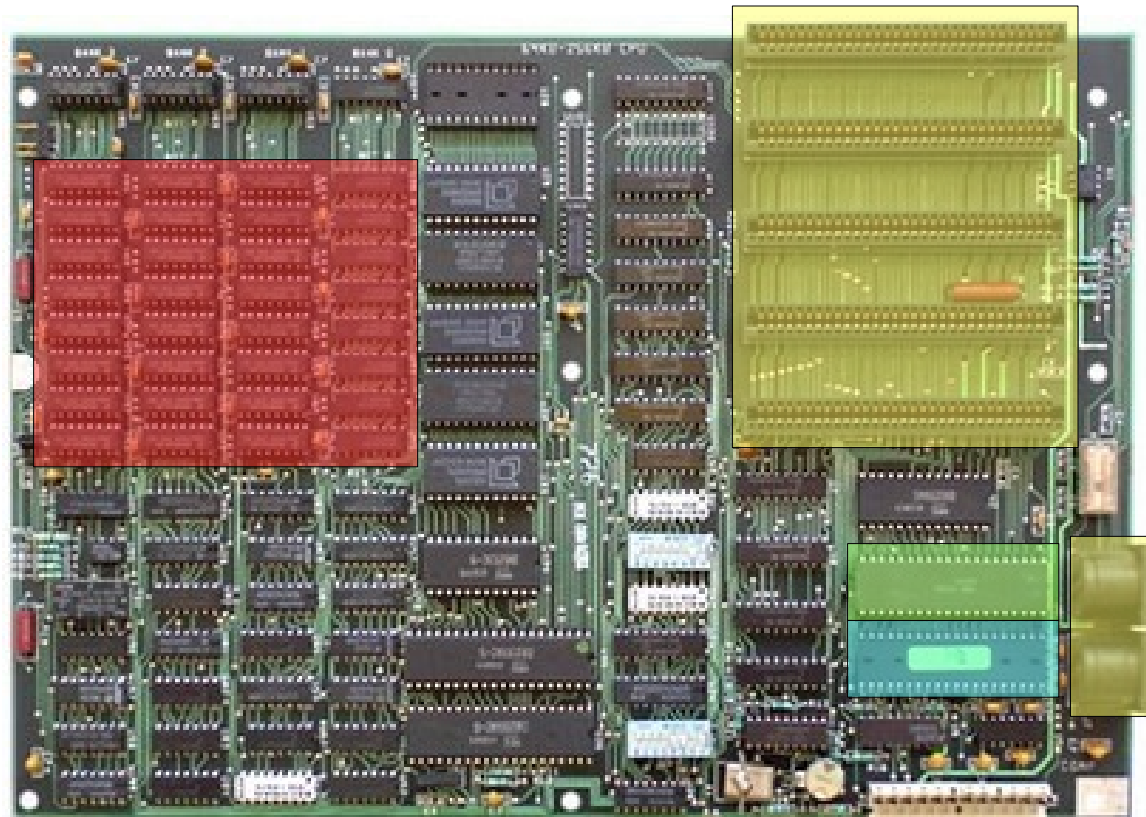
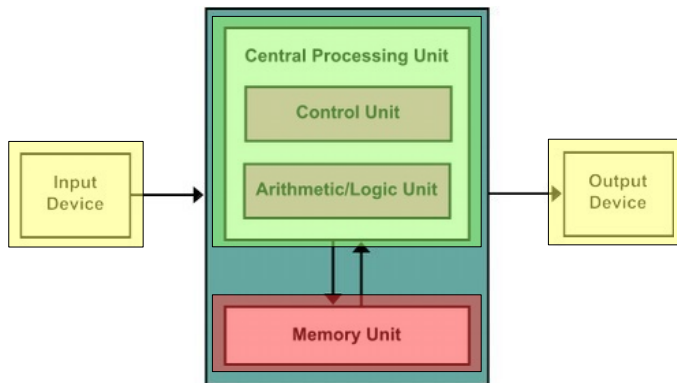


Architecture de Von Newman 50 ans d'évolution chez Intel

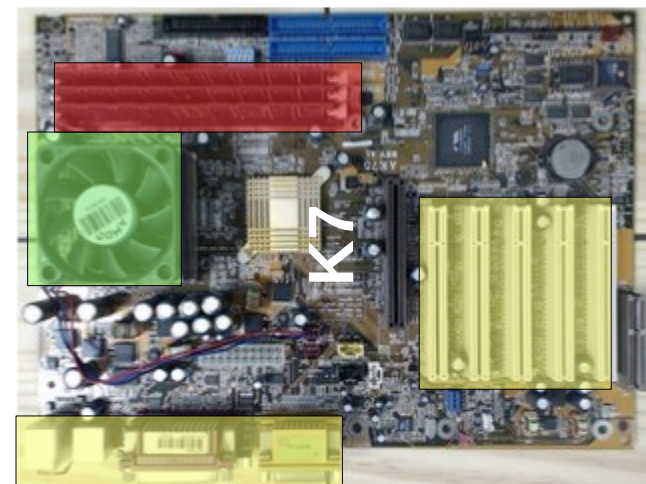
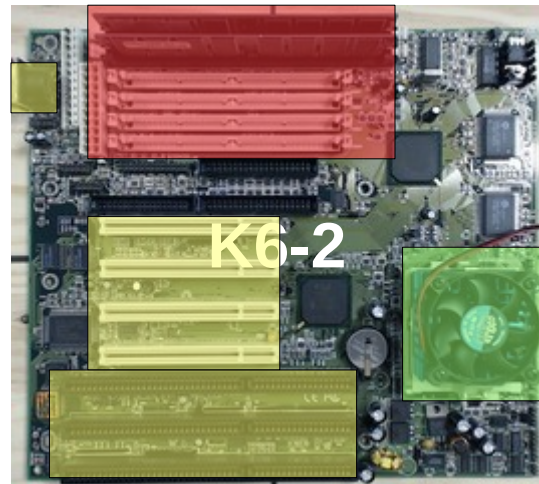
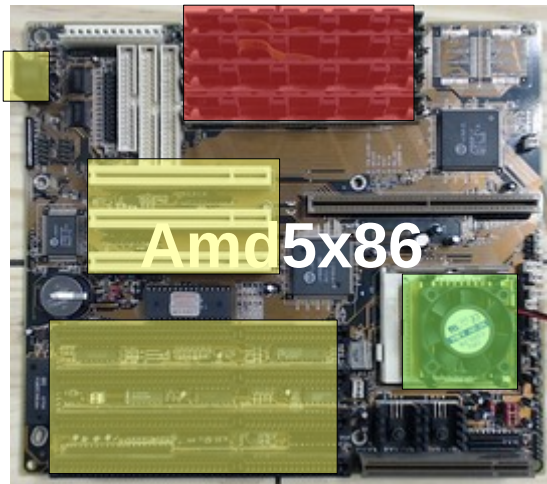
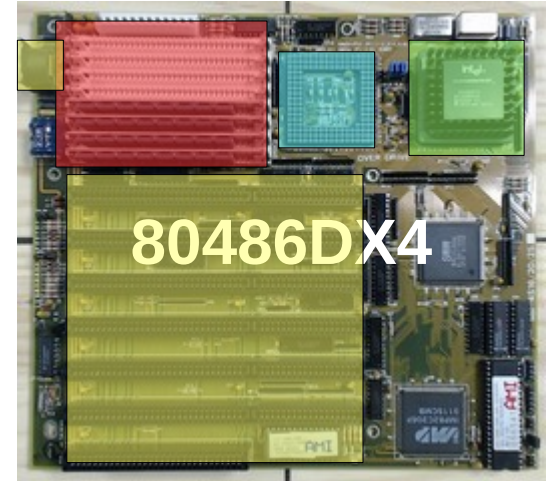


Une révolution informatique

IBM Personal Computer, le « PC »



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

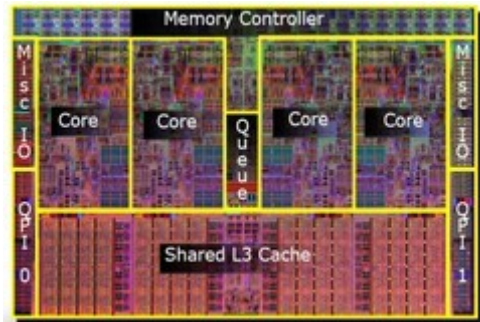
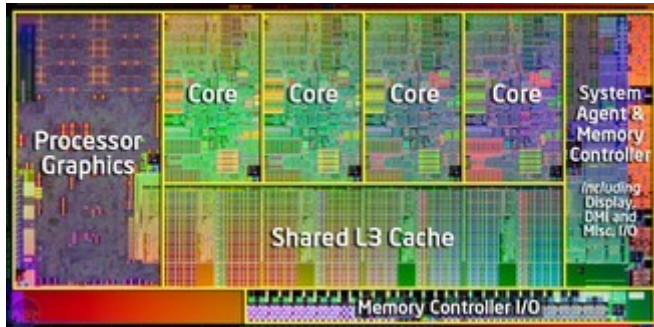


Evolution des cartes mères en un clin d'œil, de 2005 à 2018

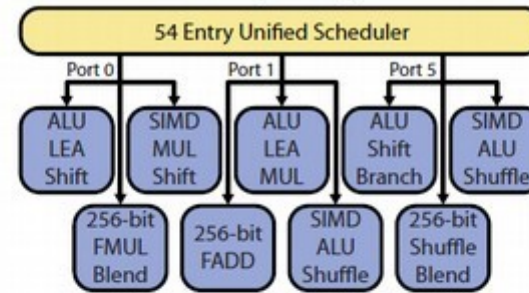


A l'intérieur d'un socket

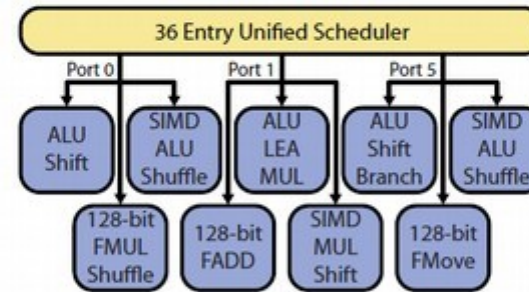
Exemples de 3 quadri-cœurs



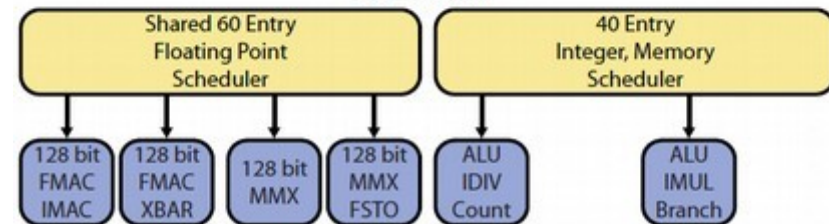
Sandy Bridge



Nehalem

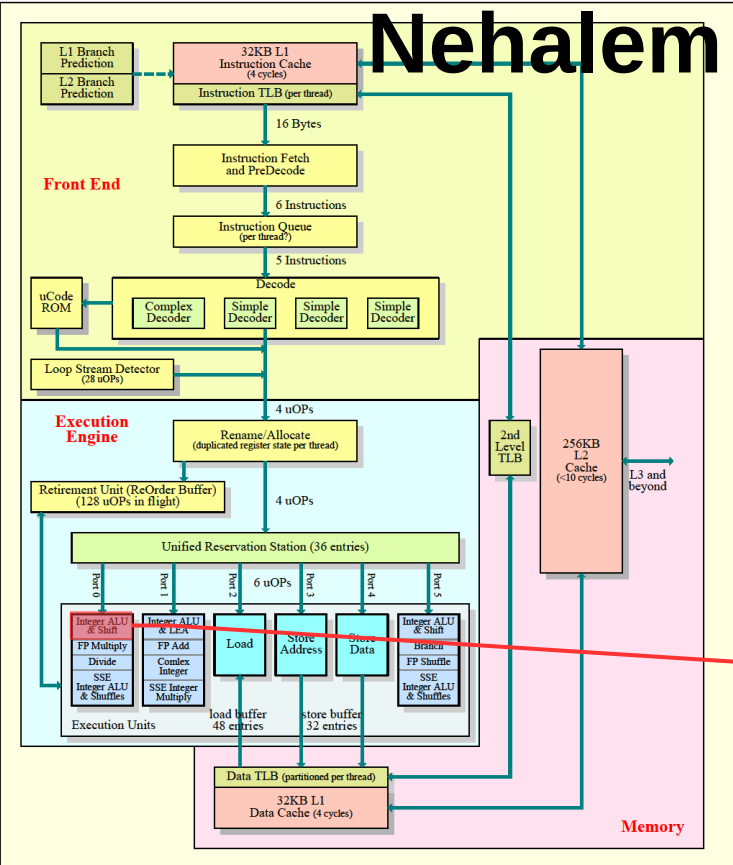


Bulldozer

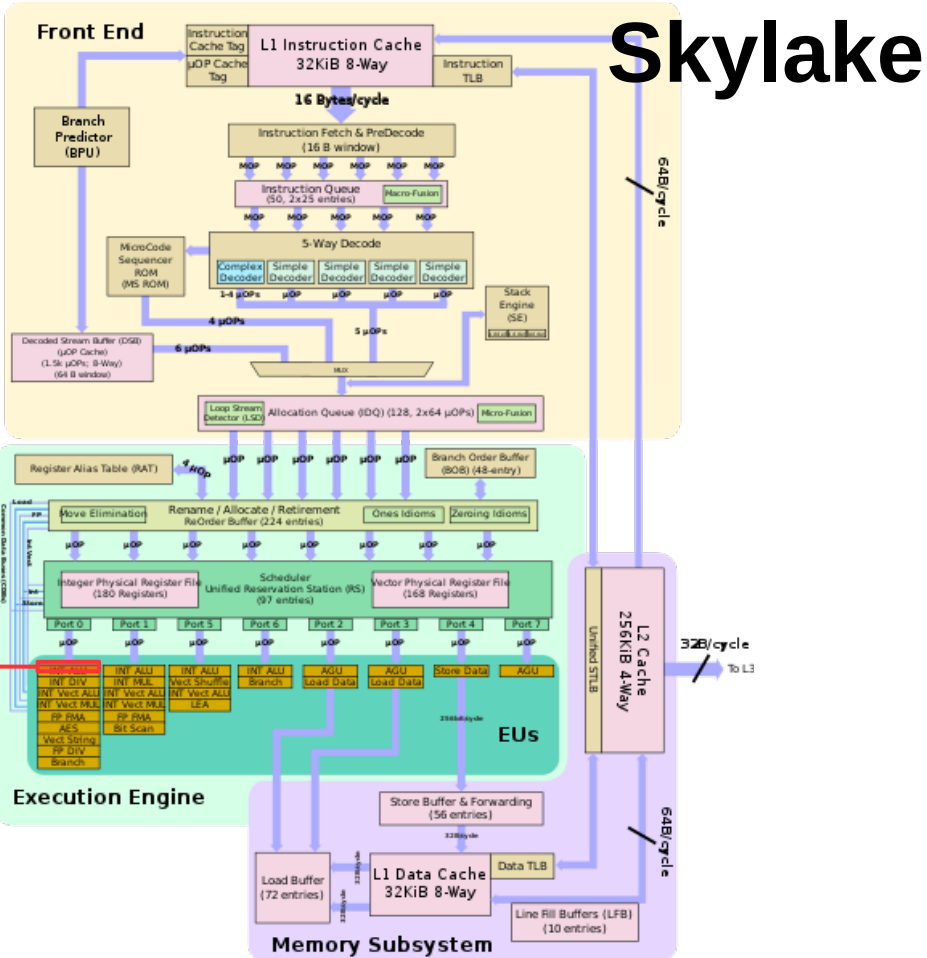


Mais maintenant, chaque cœur est un cauchemar...

Nehalem Block Diagram



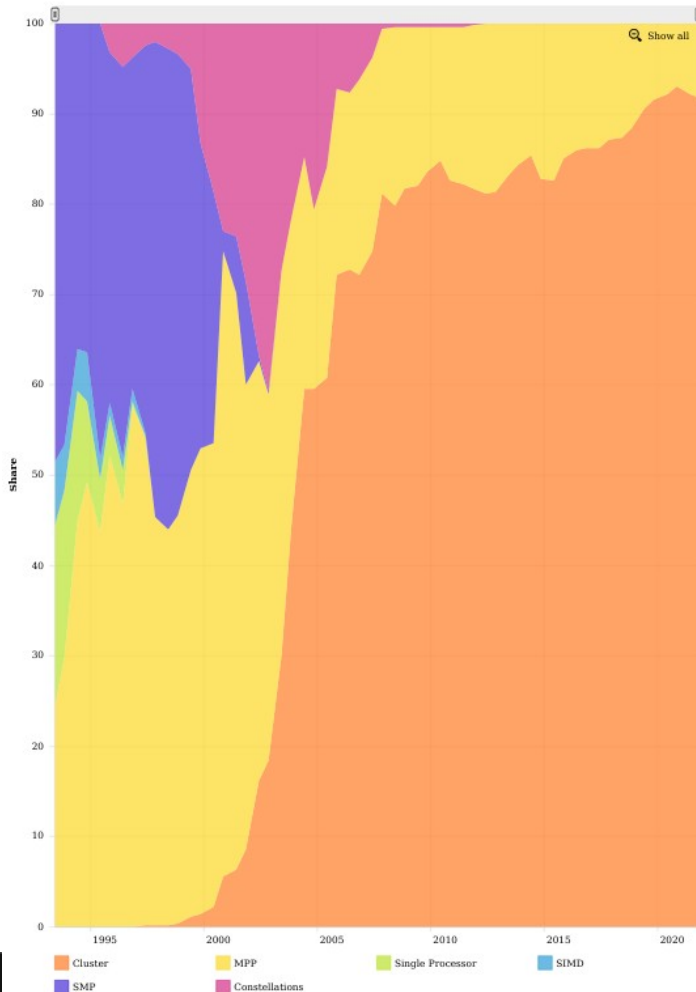
Copyright (c) 2008 Hiroshige Goto All rights reserved.



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 »

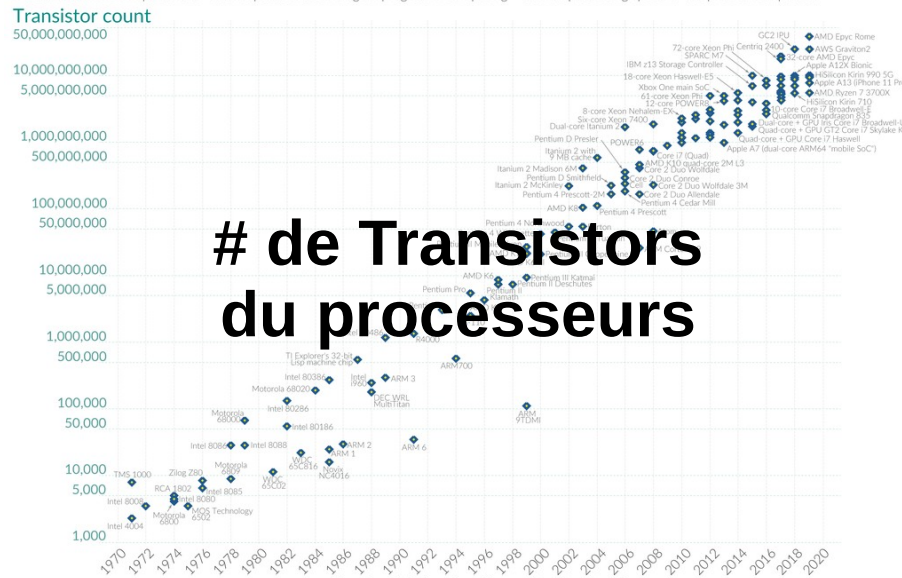
- 4 ont « disparu »...

« Lois » en informatique

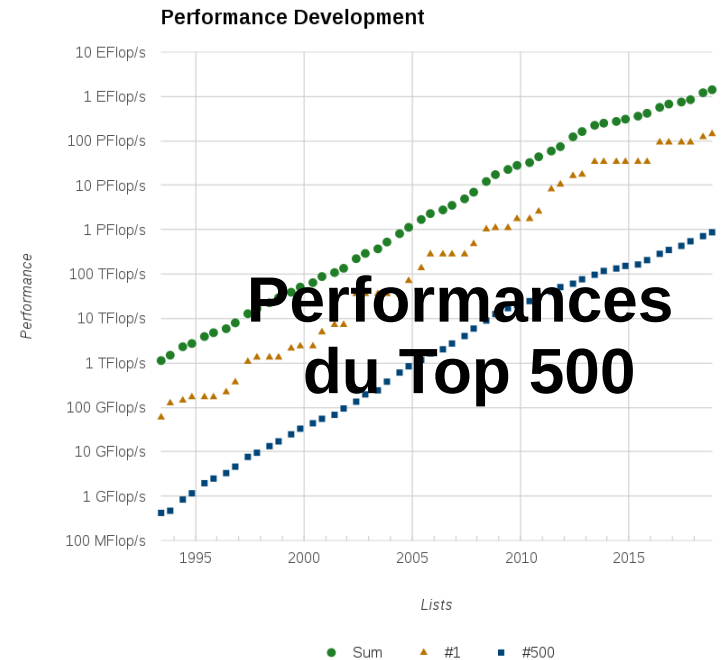
La loi de Moore (et son évolution)

- 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

Moore's Law: The number of transistors on microchips doubles every two years 
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



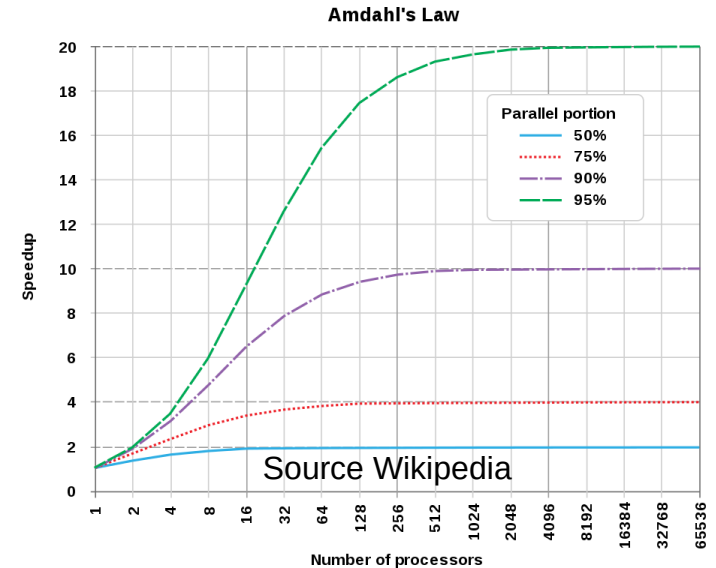
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems. Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.



« 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 bibliothèques génériques : BLAS, Lapack, FFTw
 - Des bibliothèques 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



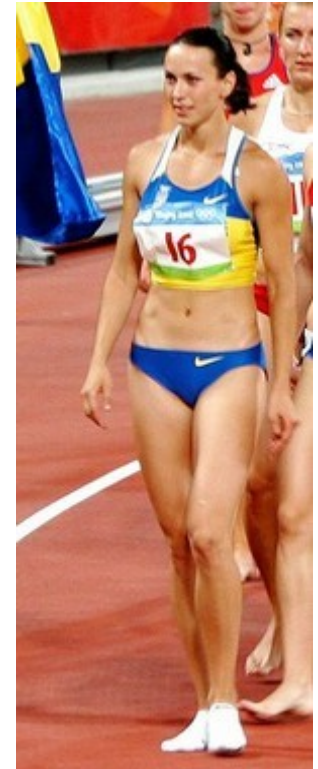
Performance : comment ?

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, $1/(\text{temps écoulé})$ (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 extrémales 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 cœurs, 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

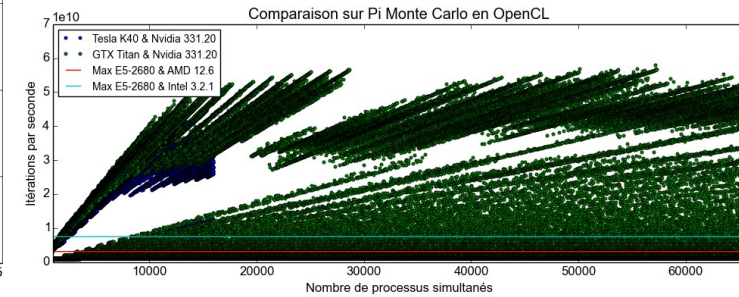
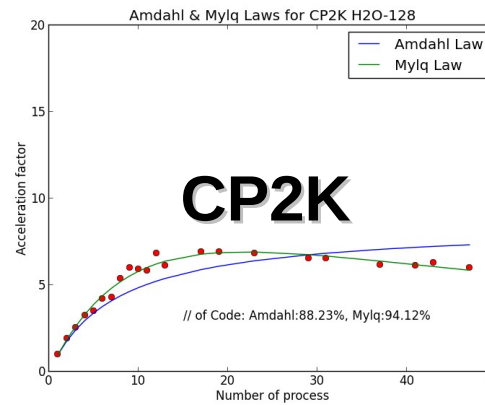
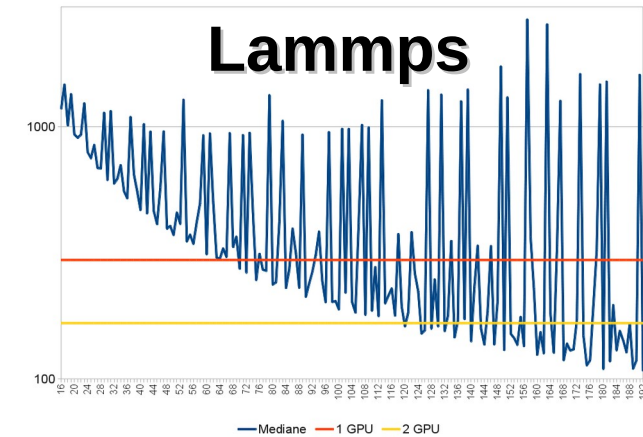
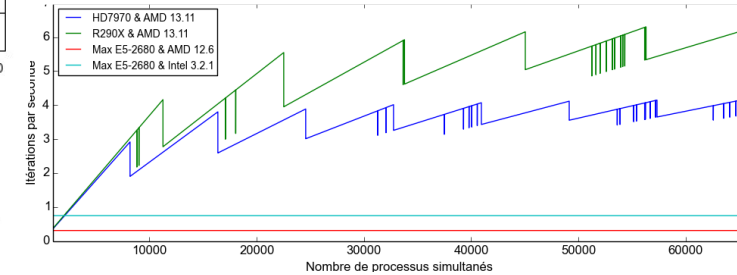
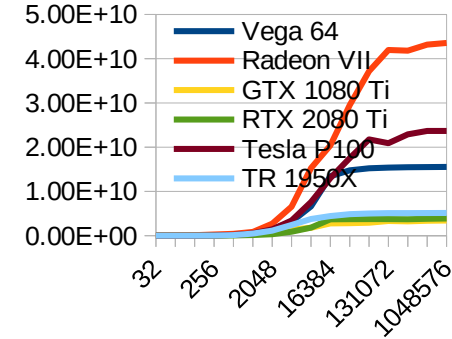
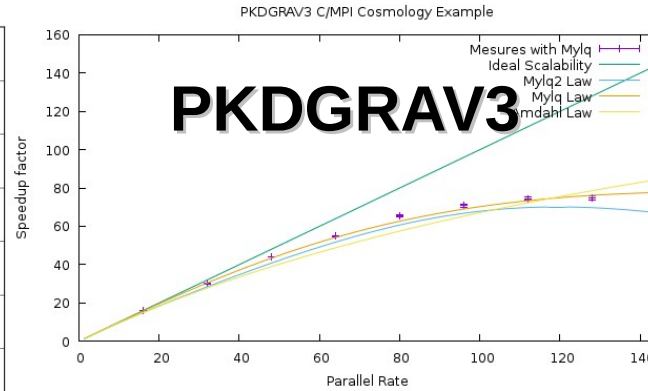
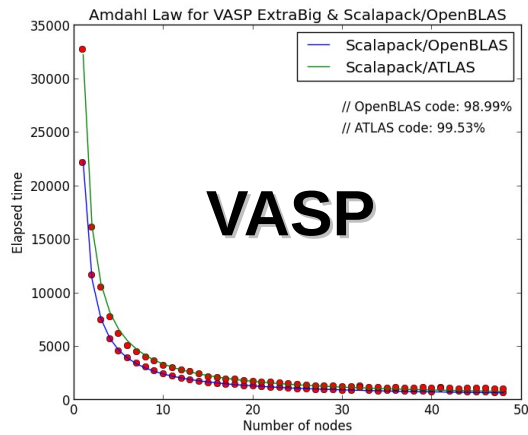
La scalabilité comme critère

- 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 ?

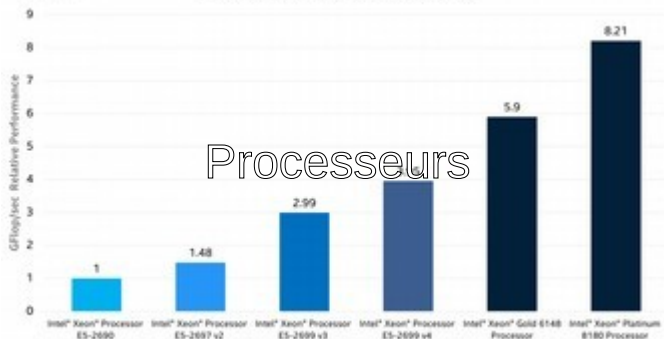
Codes « métiers » & hardware

Exemples de scalabilité

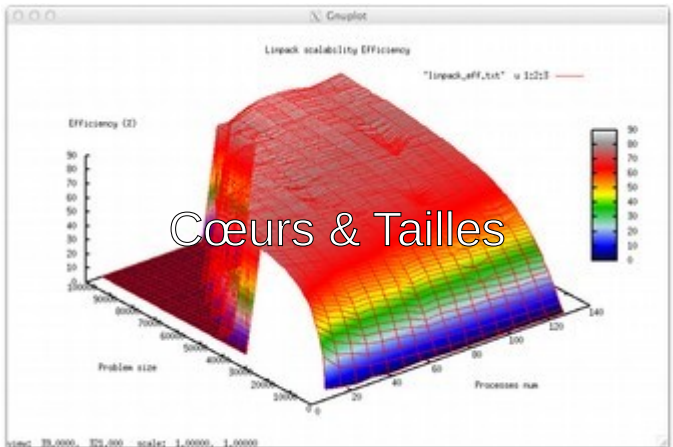


Performances des ordinateurs Entre Linpack & Phoronix... Linpack Phoronix

Intel® Distribution for LINPACK Benchmark

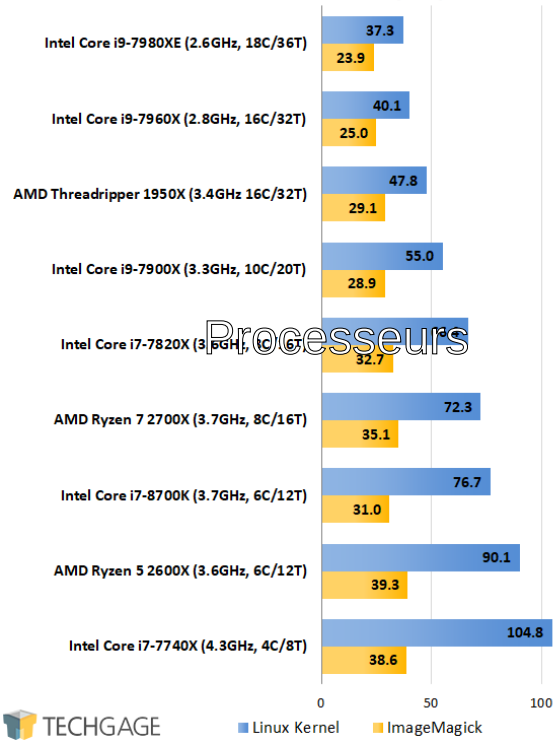


Processeurs

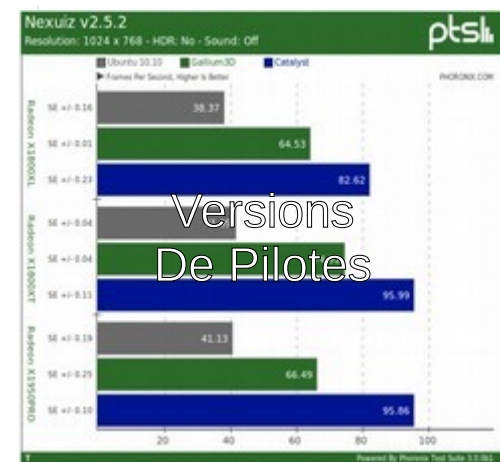


Cœurs & Tailles

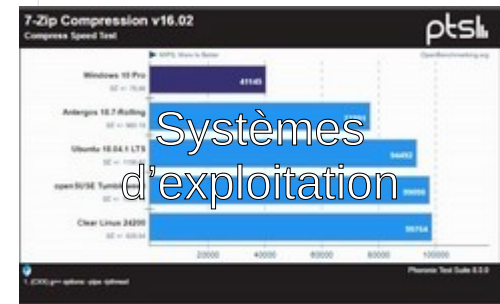
Compiler Performance (GCC: 6.3.0)
Time to Compile (Seconds, lower is better)



Processeurs



Versions De Pilotes



Systèmes d'exploitation



Linux Kernel ImageMagick

Performances des ordinateurs

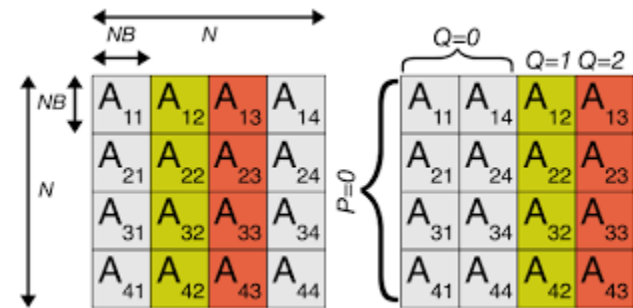
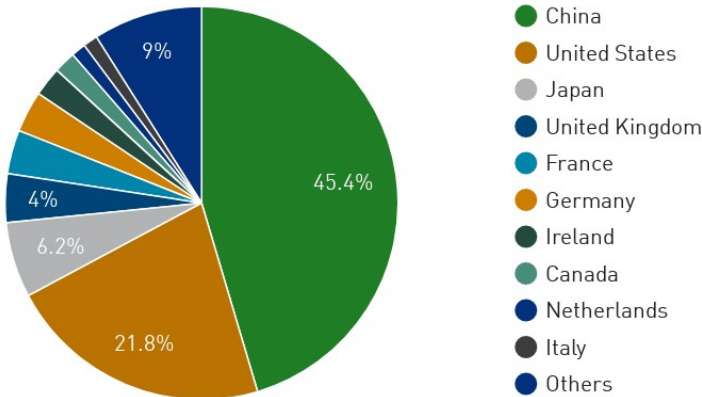
Linpack & le Top 500



- **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)...



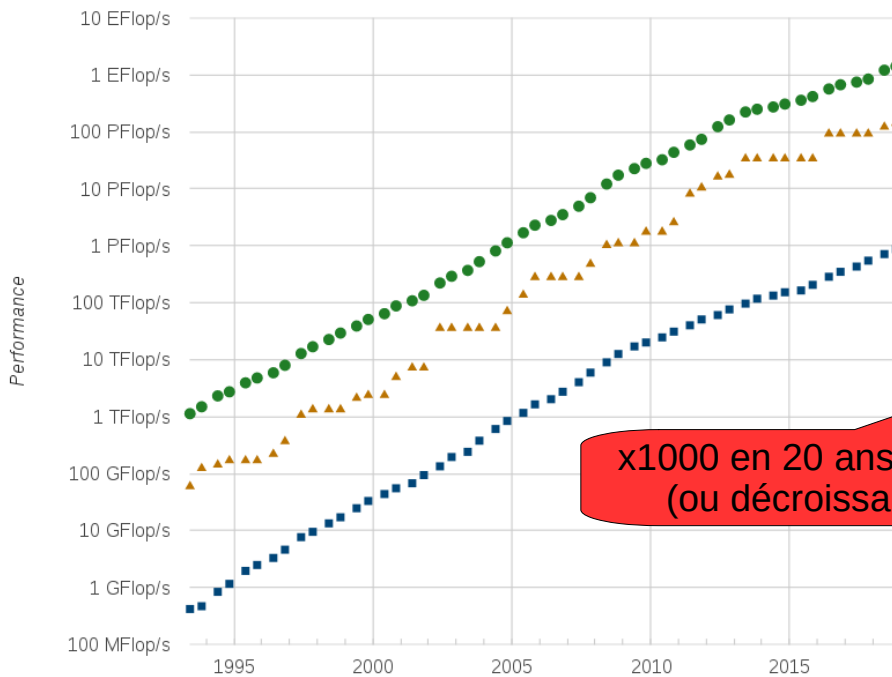
Countries System Share



Linpac & le Top 500

Loi de Moore respectée. Comment ?

Performance Development



x1000 en 20 ans & stop
(ou décroissance)



Lists

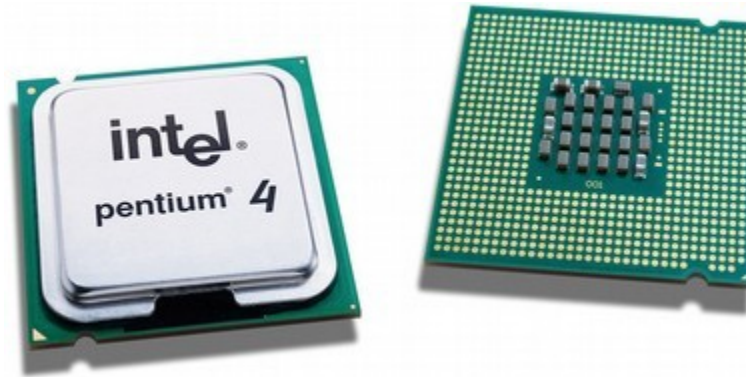
● Sum ▲ #1 ■ #500

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 cm²)
 - Densité de chaleur d'une plaque à induction !
- TDP devient le facteur limitant de puissance

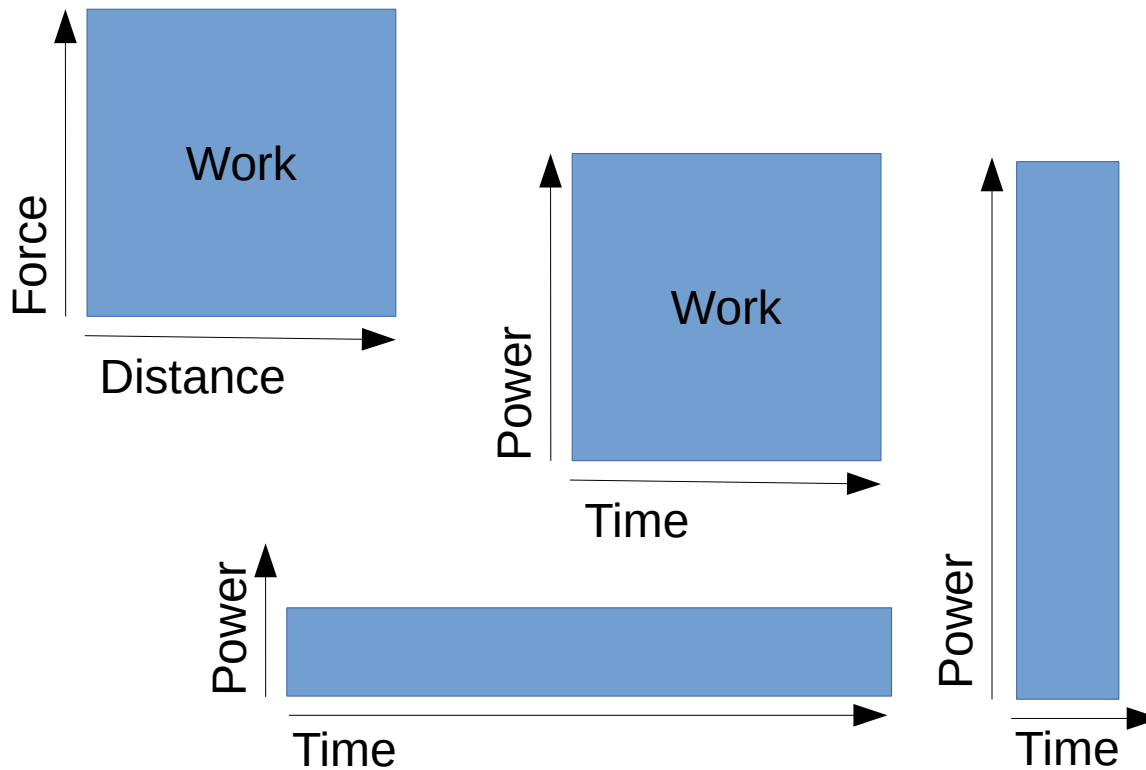


Capacitance = Finesse² . Nb Transistors . Constante de Mylq (~ 0.015)

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 !

Chart 5: Horsepower curves for 5 different motorcycles

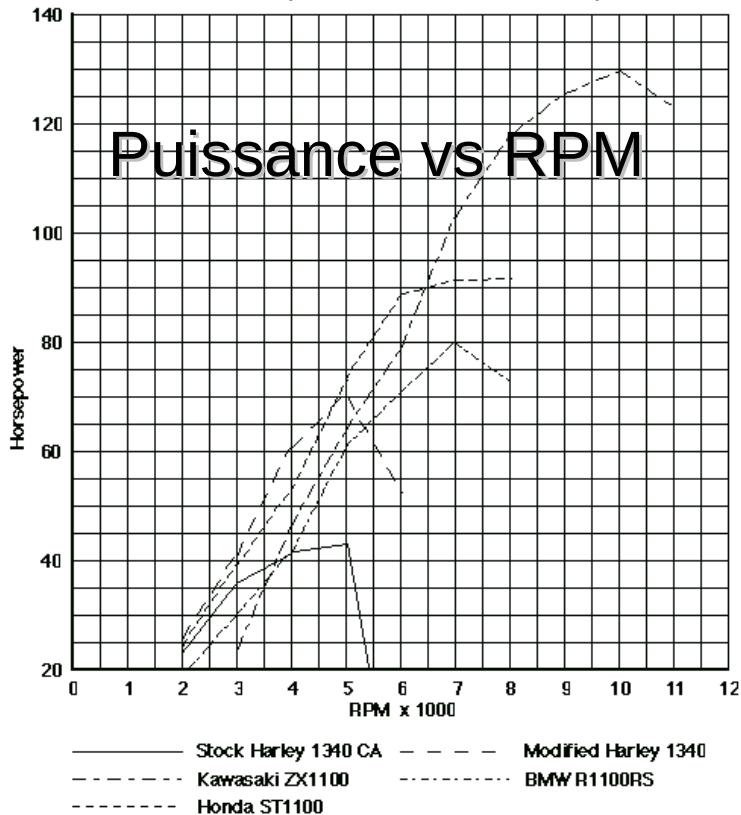
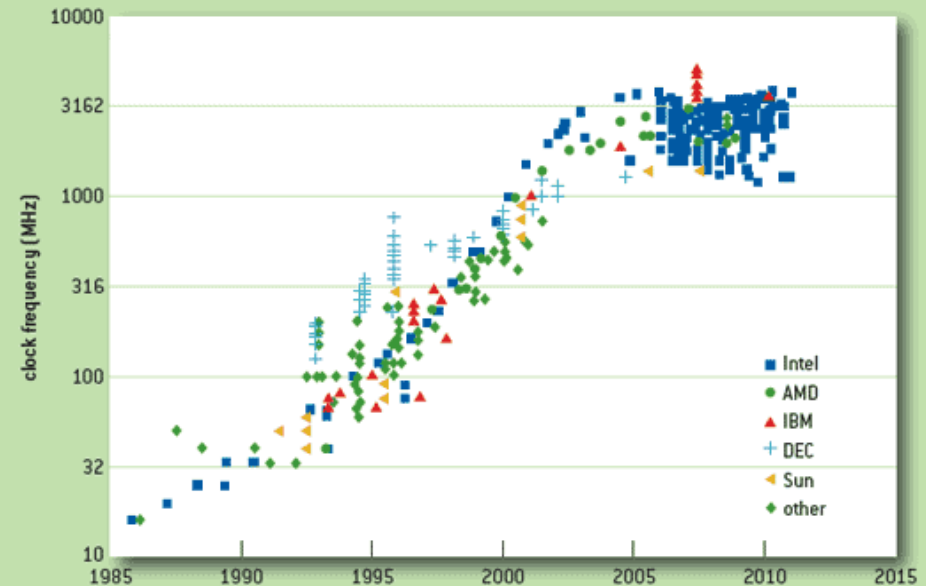


FIGURE 7

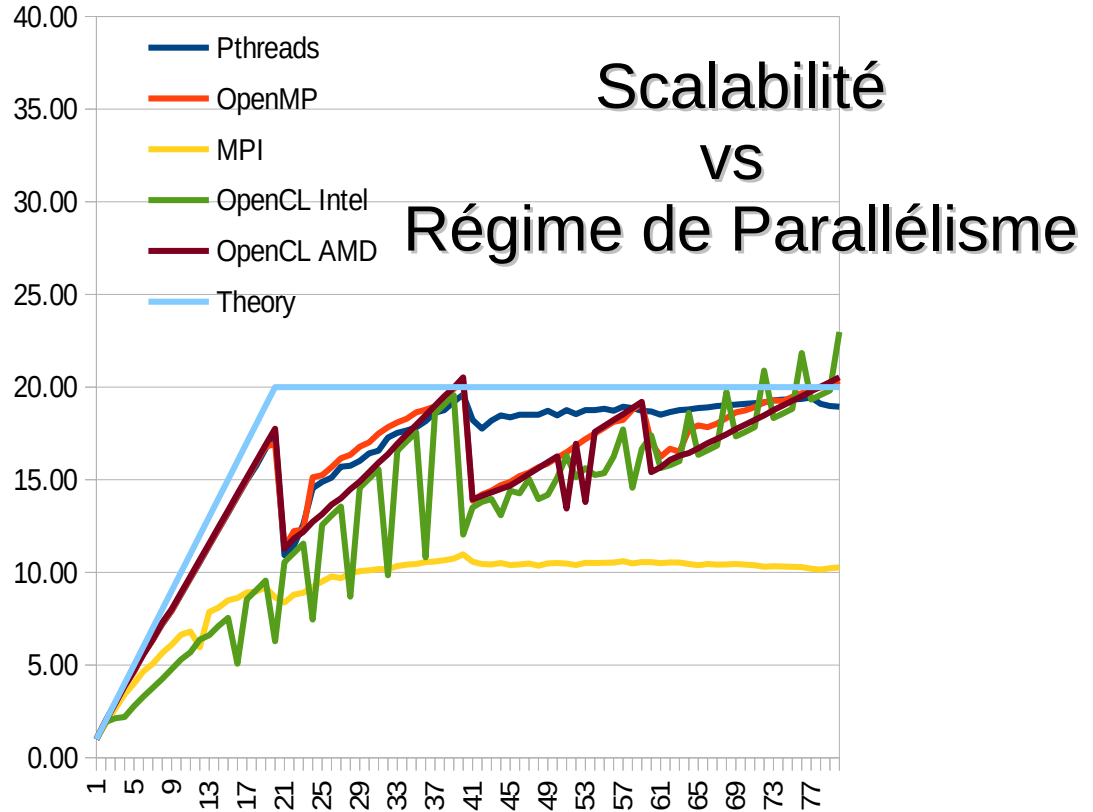
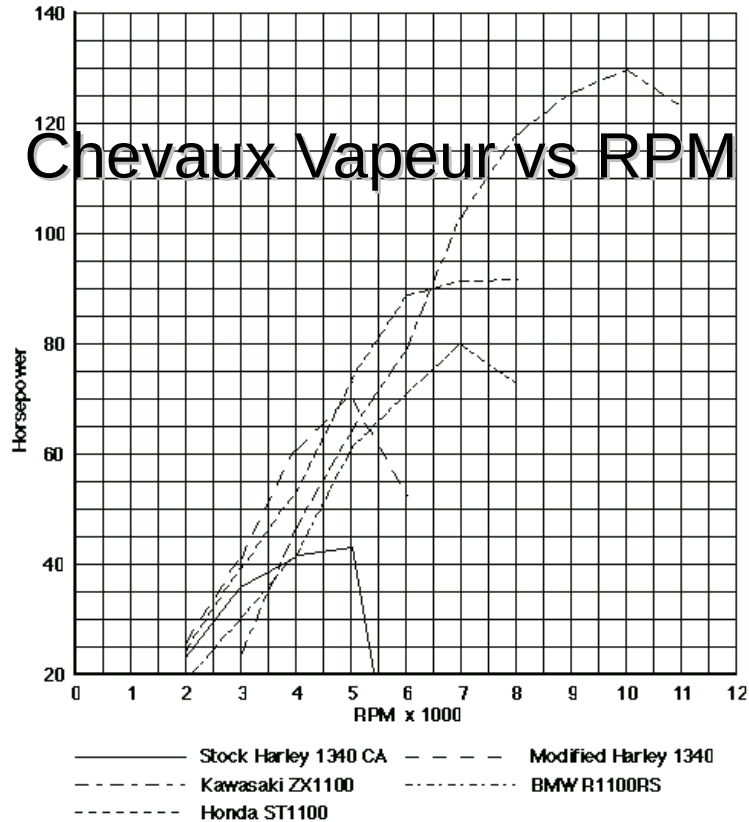
Processor Frequency Scaling Over Time



Travail & Ressources Informatiques

Un moteur comme source de puissance

Chart 5: Horsepower curves for 5 different motorcycles

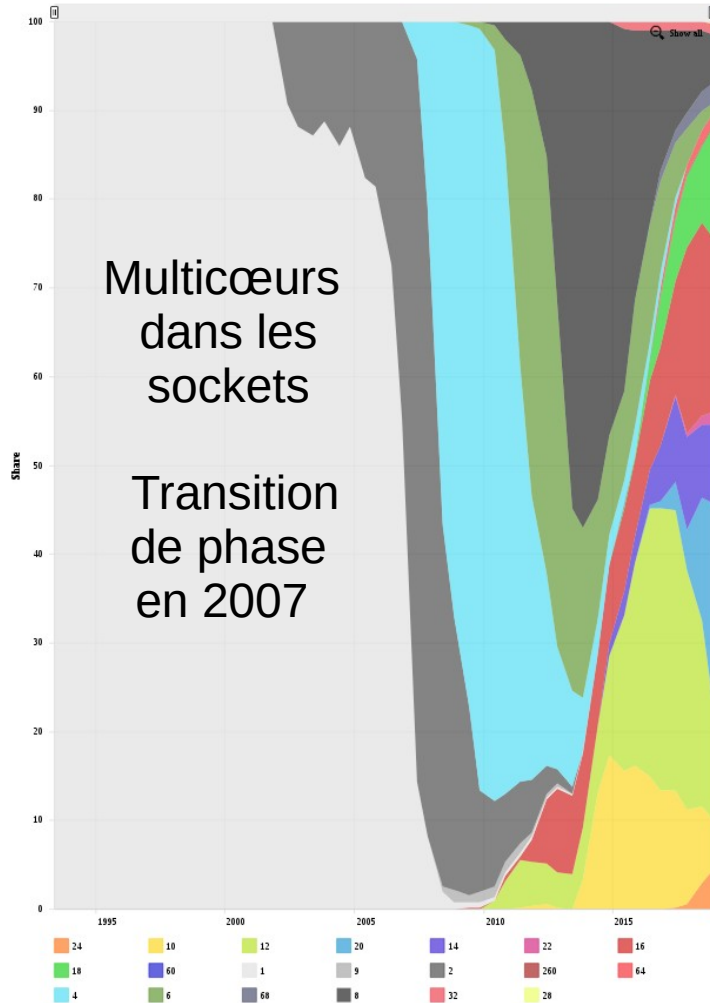


- Quel comportement de ces « moteurs » à la charge ?
- Le « moteur », un « système » englobant matériel, OS et logiciels

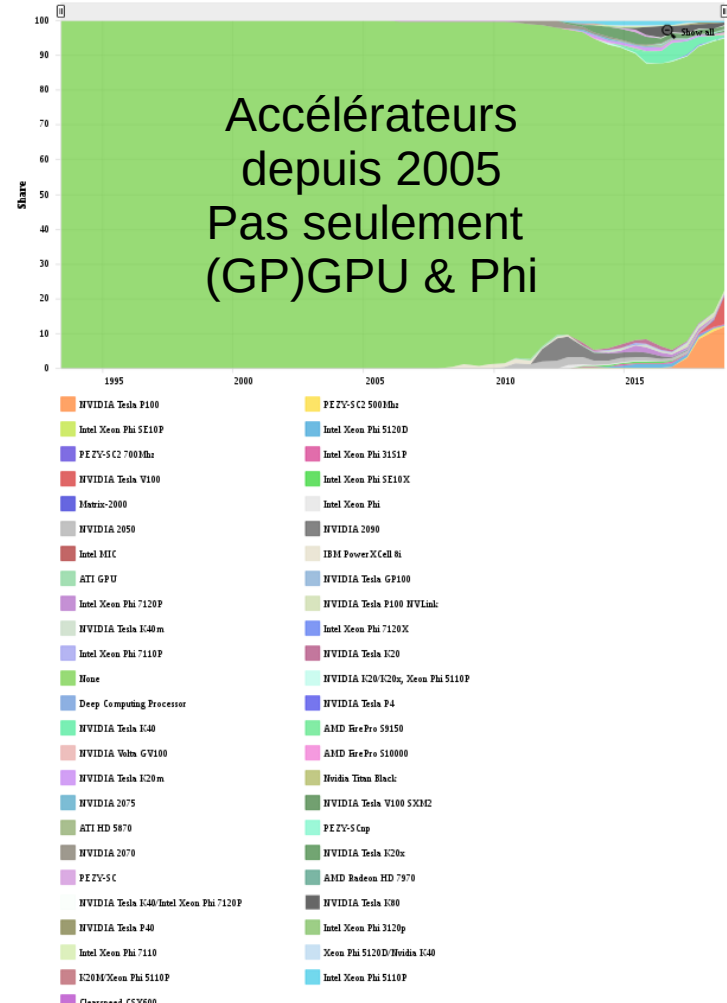
Linpack & le Top 500

Loi de Moore respectée. Comment ?

Cores per Socket - Systems Share

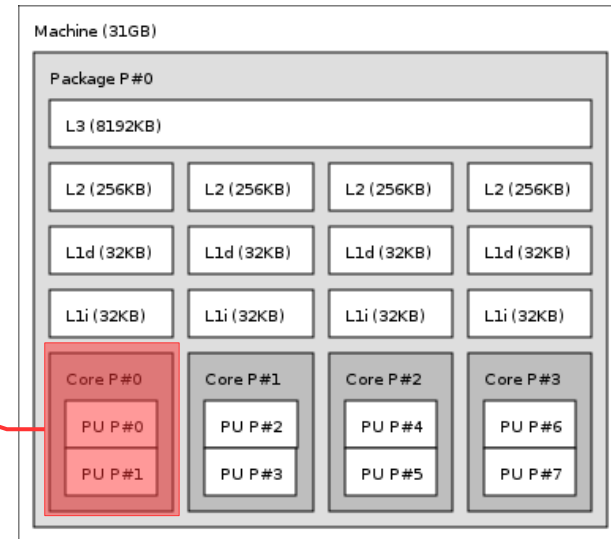
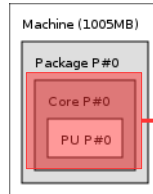


Accelerator/Co-Processor - Systems Share

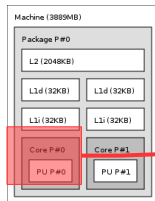


Evolution des machines...

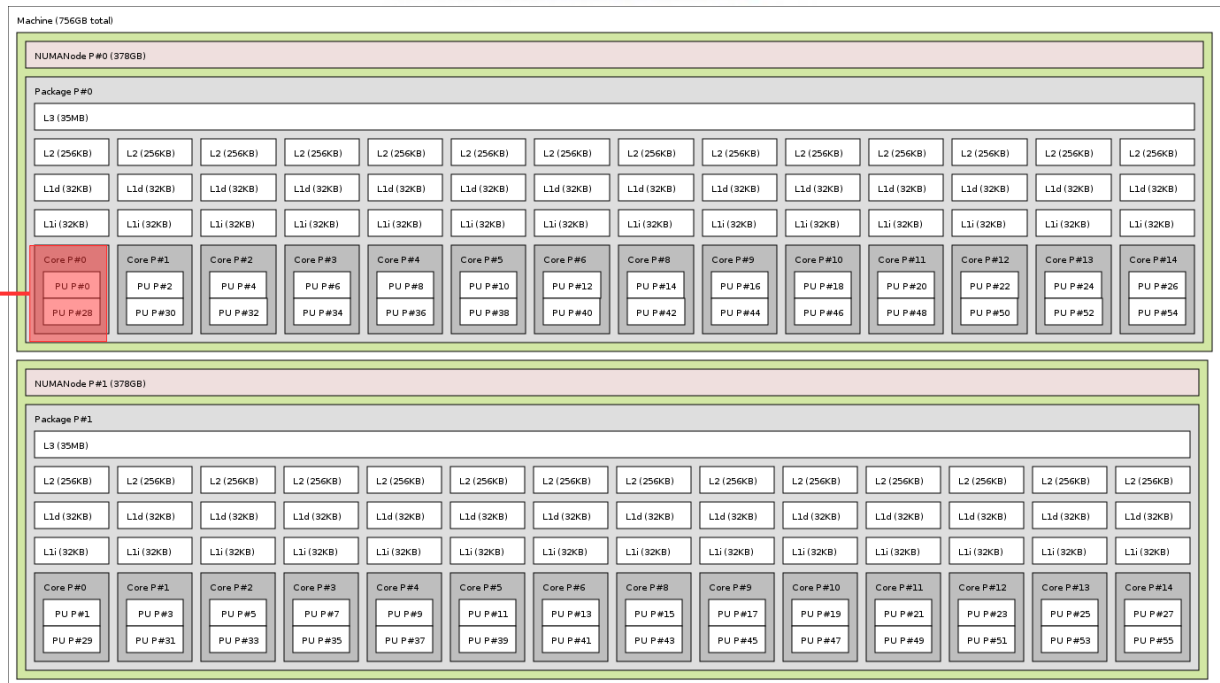
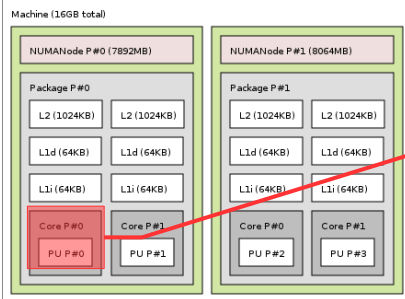
De 2004 à 2013, sur les laptops !



Evolution de 2007 à 2016 Sur les stations de travail



Evolution de 2007 à 2016 sur des serveurs...

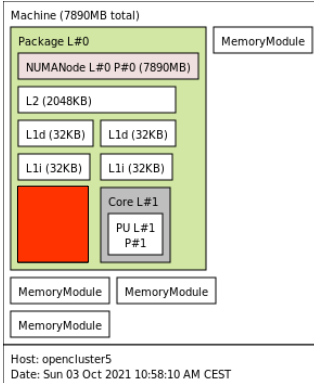


Et même pire... Dans 2U (unités) 4 serveurs très modernes

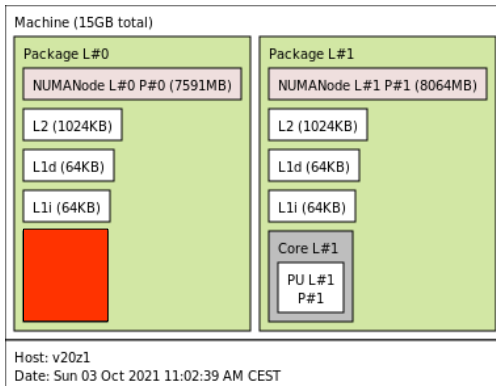


The image displays 16 server configuration sheets arranged in a 4x4 grid. Each sheet represents a server in a 2U rack, showing detailed specifications for various components like processors, memory, and storage. The sheets are organized into four groups of four, labeled 'Machine 0208-0104', 'Machine 0208-0105', 'Machine 0208-0106', and 'Machine 0208-0107'. Each sheet contains a table of hardware specifications, including CPU models, memory capacities, and storage configurations. The sheets are organized into four groups of four, labeled 'Machine 0208-0104', 'Machine 0208-0105', 'Machine 0208-0106', and 'Machine 0208-0107'. Each sheet contains a table of hardware specifications, including CPU models, memory capacities, and storage configurations.

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



Machine (7890MB total)		Package L#0	Package L#1
NUMANode L#0 P#0 (7890MB)	NUMANode L#1 P#0 (7890MB)	NUMANode L#0 P#0 (7890MB)	NUMANode L#1 P#0 (7890MB)
L2 (2048KB)	L2 (2048KB)	L2 (2048KB)	L2 (2048KB)
L1d (32KB)	L1d (32KB)	L1d (32KB)	L1d (32KB)
L1i (32KB)	L1i (32KB)	L1i (32KB)	L1i (32KB)
Core L#1	Core L#1	Core L#1	Core L#1
PU L#1 P#1	PU L#1 P#1	PU L#1 P#1	PU L#1 P#1
MemoryModule	MemoryModule	MemoryModule	MemoryModule
MemoryModule	MemoryModule	MemoryModule	MemoryModule
MemoryModule	MemoryModule	MemoryModule	MemoryModule



Machine (15GB total)		Package L#0	Package L#1
NUMANode L#0 P#0 (7591MB)	NUMANode L#1 P#0 (7591MB)	NUMANode L#0 P#0 (7591MB)	NUMANode L#1 P#0 (7591MB)
L2 (1024KB)	L2 (1024KB)	L2 (1024KB)	L2 (1024KB)
L1d (64KB)	L1d (64KB)	L1d (64KB)	L1d (64KB)
L1i (64KB)	L1i (64KB)	L1i (64KB)	L1i (64KB)
Core L#1	Core L#1	Core L#1	Core L#1
PU L#1 P#1	PU L#1 P#1	PU L#1 P#1	PU L#1 P#1
MemoryModule	MemoryModule	MemoryModule	MemoryModule
MemoryModule	MemoryModule	MemoryModule	MemoryModule
MemoryModule	MemoryModule	MemoryModule	MemoryModule

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_{CPU} + Rpeak_{GPU}
 - Rpeak_{CPU} = 9216 * 22 * 3.07e9 * 64 ~ 20 %
 - Rpeak_{GPU} = 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 cœurs (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 SXM2, 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...

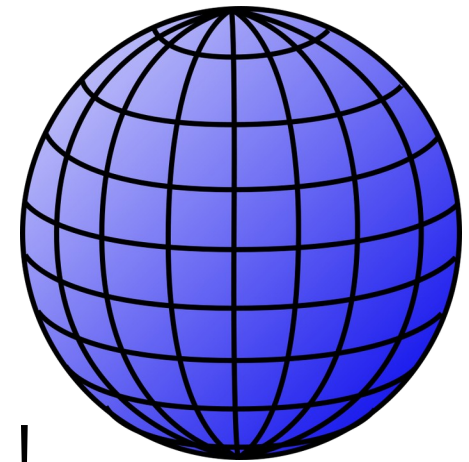
Où sommes-nous ? Où allons-nous ?

- 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
 - Communications 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 !

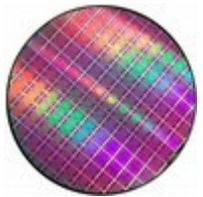
Le Grain est défini par le taux de communication !

- Grain fin : communications fréquentes ($\sim \gg 1/\text{seconde}$)
- Gros grain : communications éparées ($\sim < 1/\text{seconde}$)
- « 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

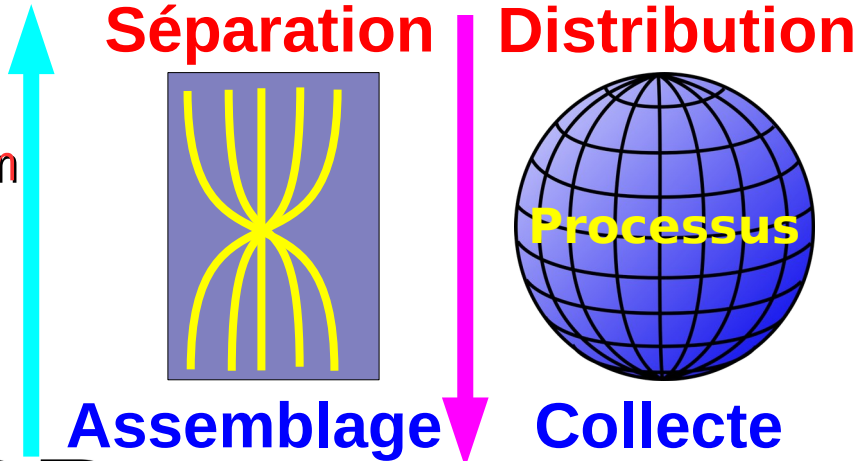
Instr. No.	Pipeline Stage						
	IF	ID	EX	MEM	WB		
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

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

- 2 approches différentes :

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

- En fait, médianisation ;-)



Au delà de la cuisine

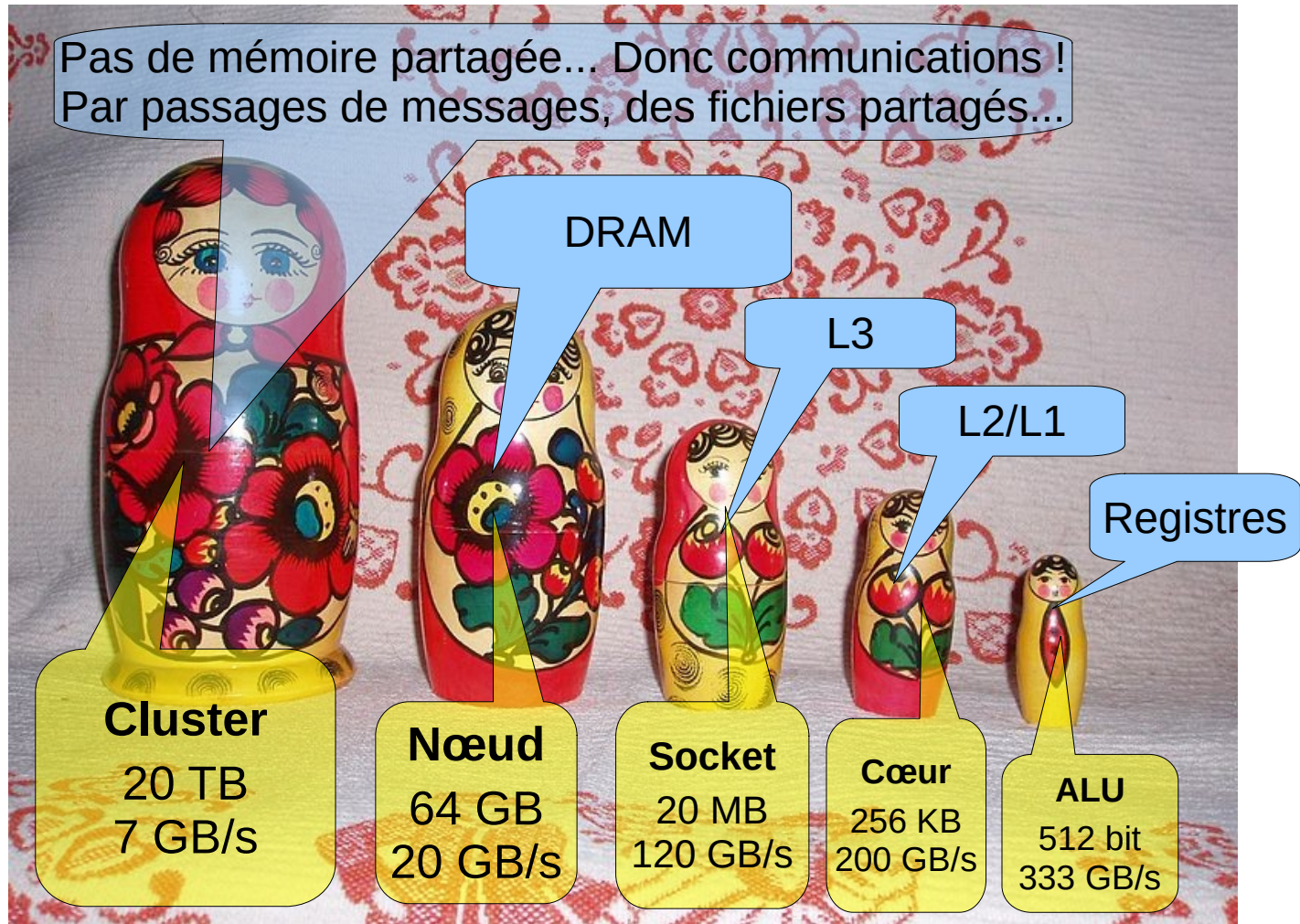
Le déménagement.

- 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

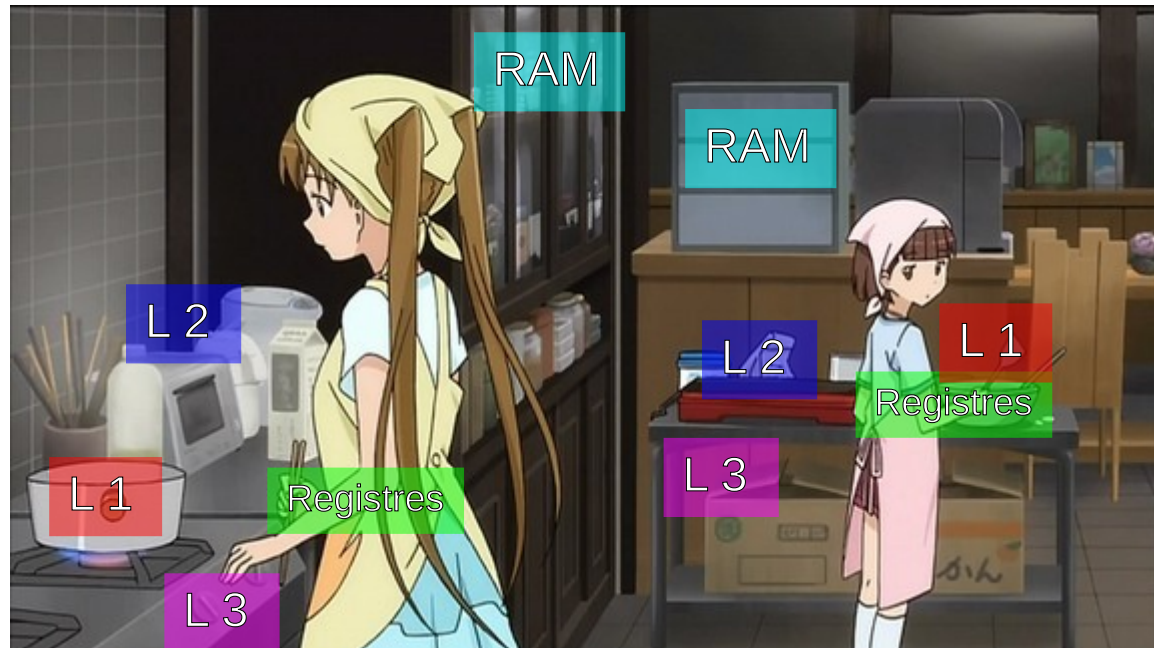
Dynamic RAM ~ Placards, tables, ...

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
MPI	Oui	Oui	Non	Non	Oui*
PVM	Oui	Oui	Non	Non	Oui*
OpenMP	Non	Oui	Non	Non	Oui*
Pthreads	Non	Oui	Non	Non	Oui*
OpenCL	Non	Oui	Oui	Oui	Oui
CUDA	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	cMAGMA	MAGMA	MagmaMIC
FFT	FFTw3	FFTw3	cFFTW	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
```

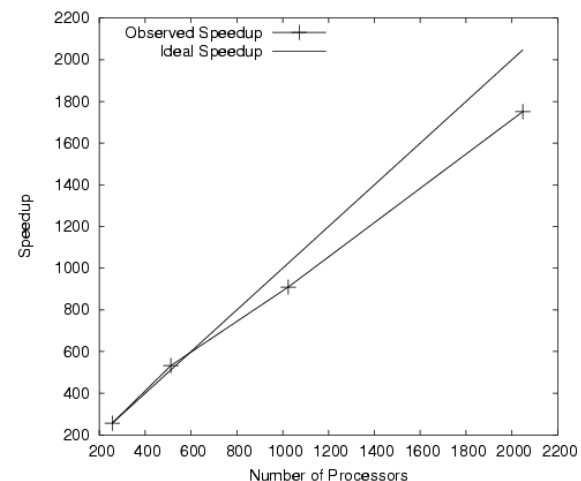
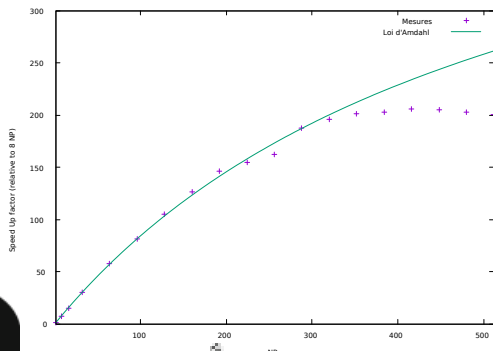
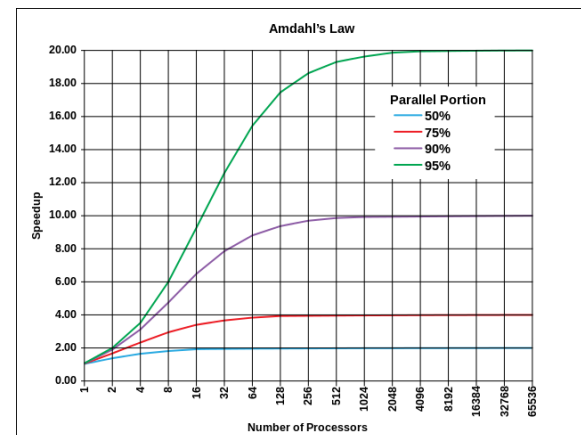
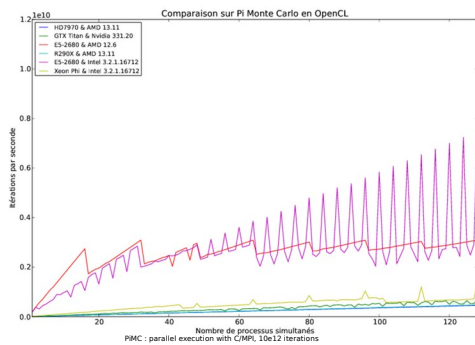
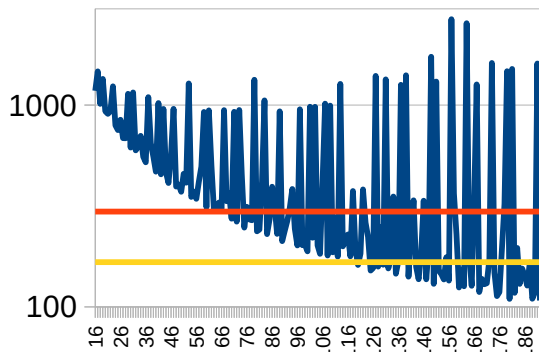
PID	USER	PRI	NI	VM	VSZ	PMEM	TIME	COMMAND
1	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd
9	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd
13	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd
18	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd
11	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd
15	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd
12	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd
16	root	20	0	3968	1024	0	0:00.00	/usr/sbin/sshd

```
dstat
```

usr	sys	idl	wai	hiq	xiq	run	cmw	cmx
24	0	76	0	0	0	0	111	9594
50	0	50	0	0	0	0	15	6400
50	0	50	0	0	0	0	28500	6400
50	0	50	0	0	0	0	89560	25760
50	0	50	0	0	0	0	49340	6400
50	0	50	0	0	0	0	16	6400
50	0	50	0	0	0	0	22	6400
50	0	50	0	0	0	0	11	6400
50	0	50	0	0	0	0	94640	7460
50	0	50	0	0	0	0	24	19740
50	0	50	0	0	0	0	19	6400

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

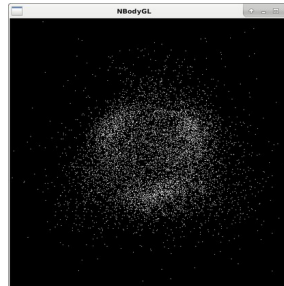
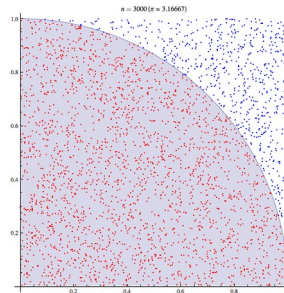
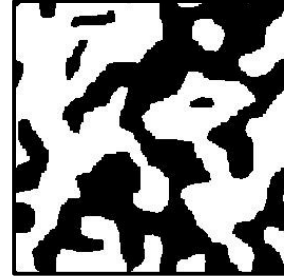
A l'origine, une notion essentielle en sciences : la « référence »

- 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, clBLAS, gslBLAS...
 - Aborder les problèmes avec deux approches : intégrateur & développeur

Le souci de l'universalité...

L'émergence de « codes matrices »

- 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)

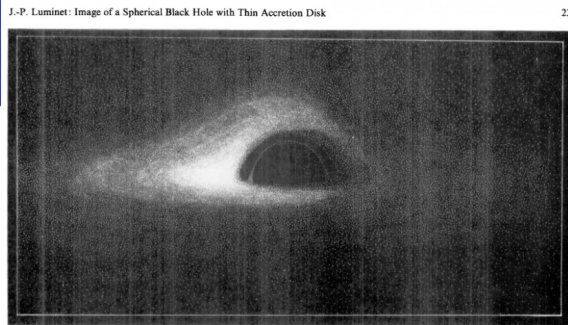


Mais aucune exploitation massive de la mémoire...

40 ans entre simulation et mesure

De 2019 à 1979...

1979 : JP Luminet A&A



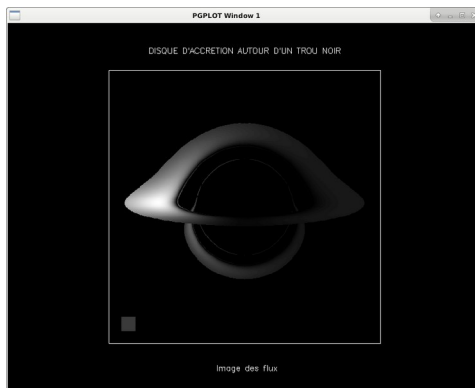
2014 : Film Interstellar



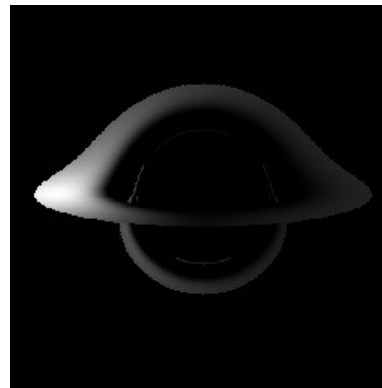
2019 : EHT, Messier 87



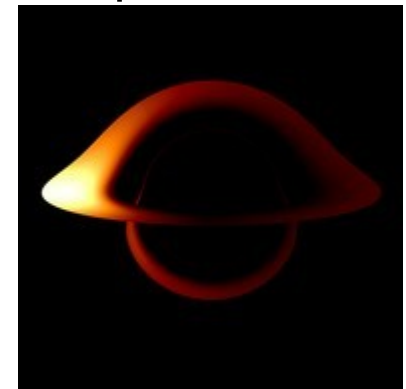
1994 : Code Fortran



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

1. Introduction

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

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

(1973) of the problem of energy release by a thin accretion disk in a general astrophysical context, focusing attention more particularly on the analytic solution for the surface distribution of energy release that was derived by Page and Thorne (1974) in the limiting case of a sufficiently low accretion rate. In terms of this idealized (but in appropriate circumstances, realistic) model, we calculate the distribution of bolometric flux as seen by distant observers at various angles above the plane of the disk (Figs. 9–11).

2. Image of a Bare Black Hole

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

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

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

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

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

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

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

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

De l'article au rapport

- Métrique de Schwarzschild :

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

- Equation polaire :

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

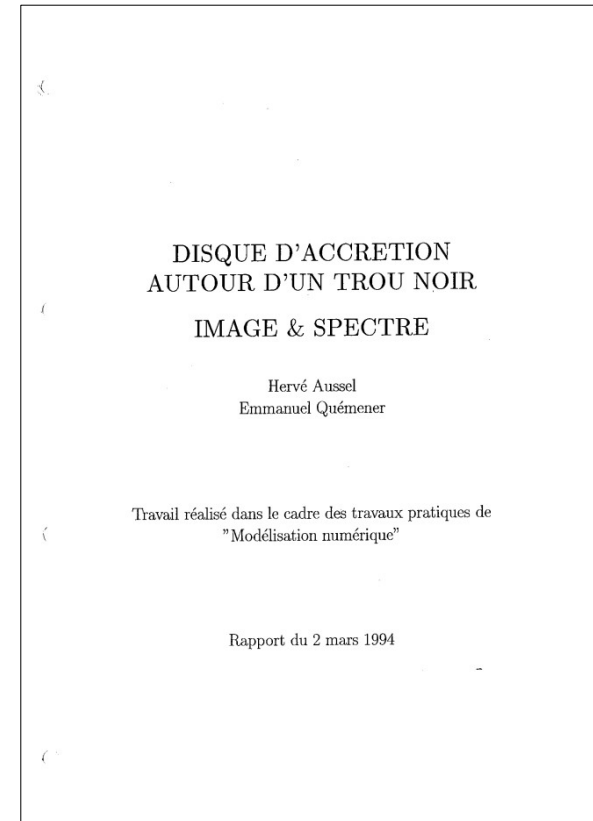
- Changement de coordonnées : $u=1/r$

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

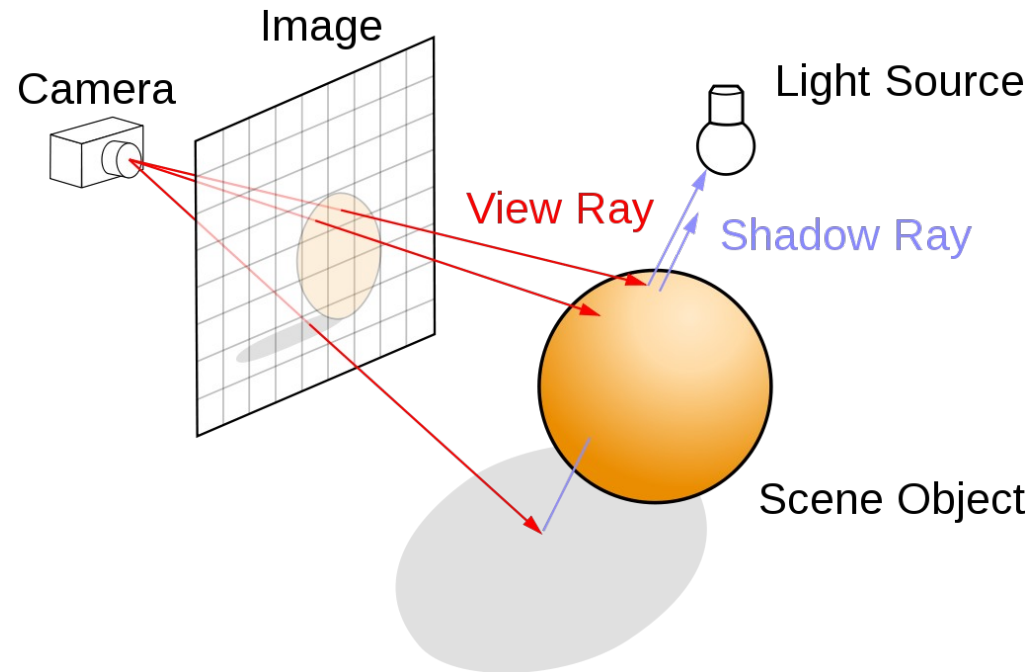
- Dérivation de l'équation polaire :

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

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



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...

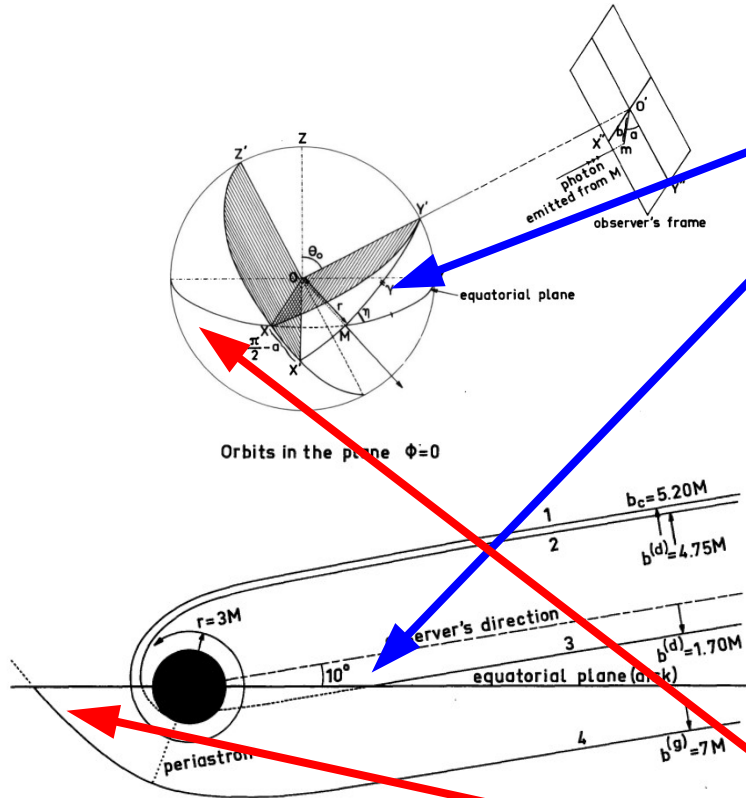
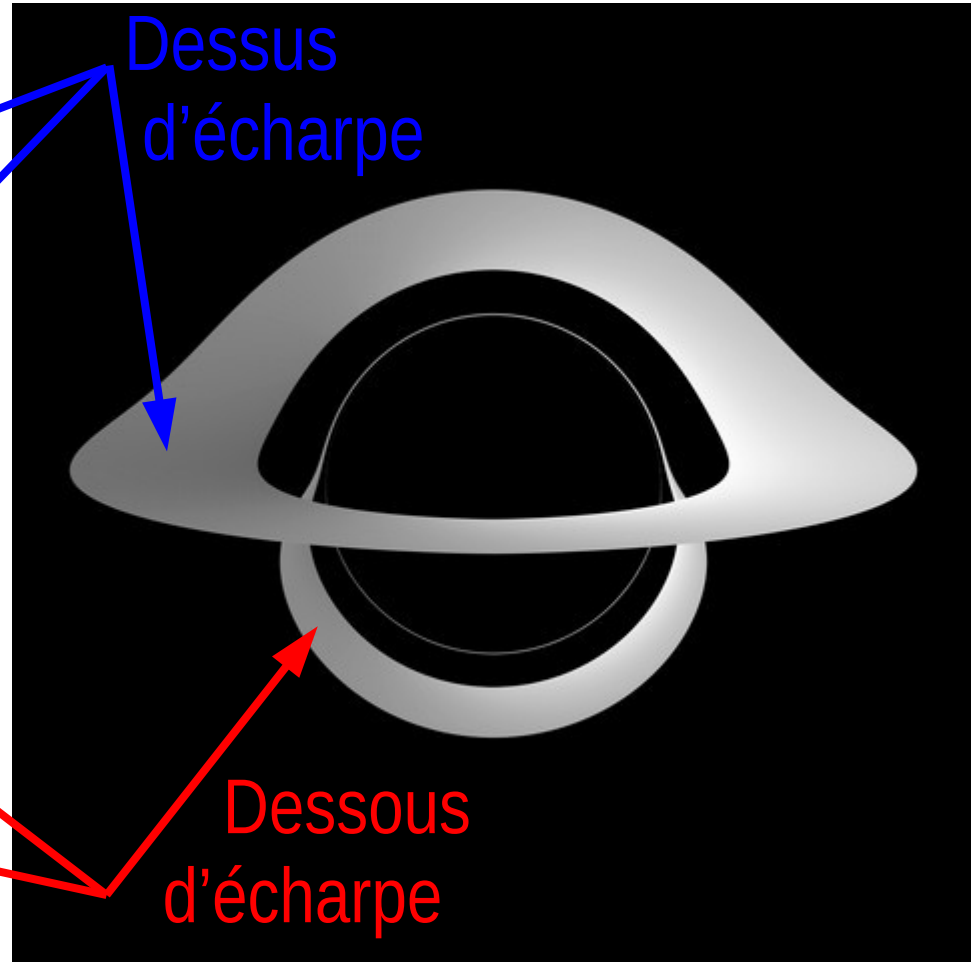
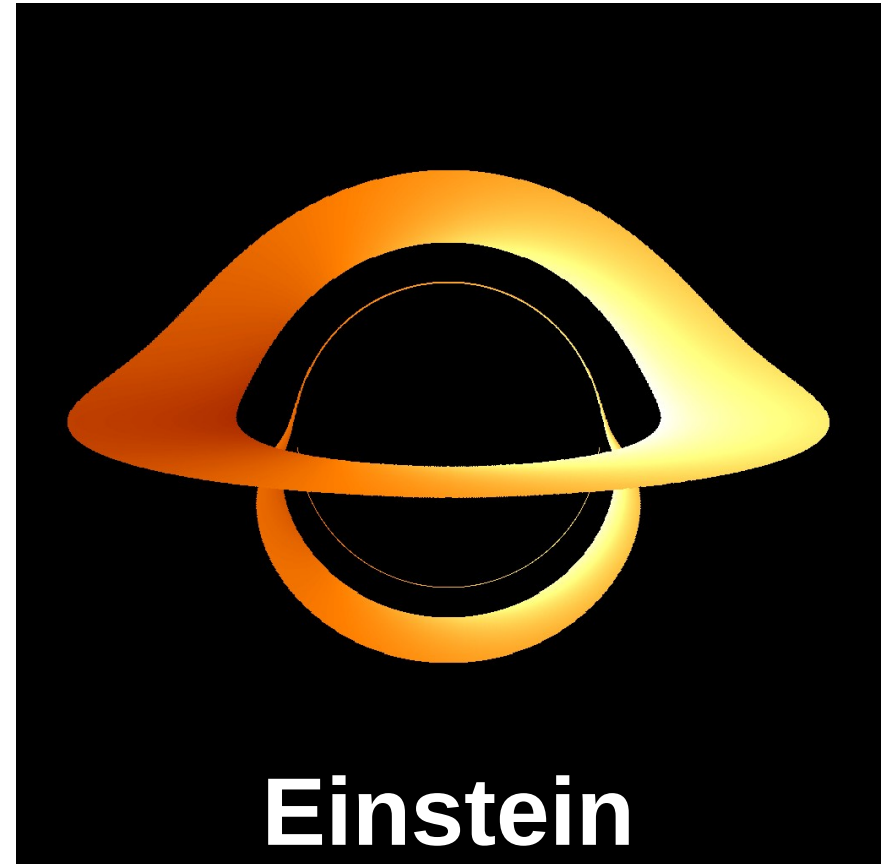
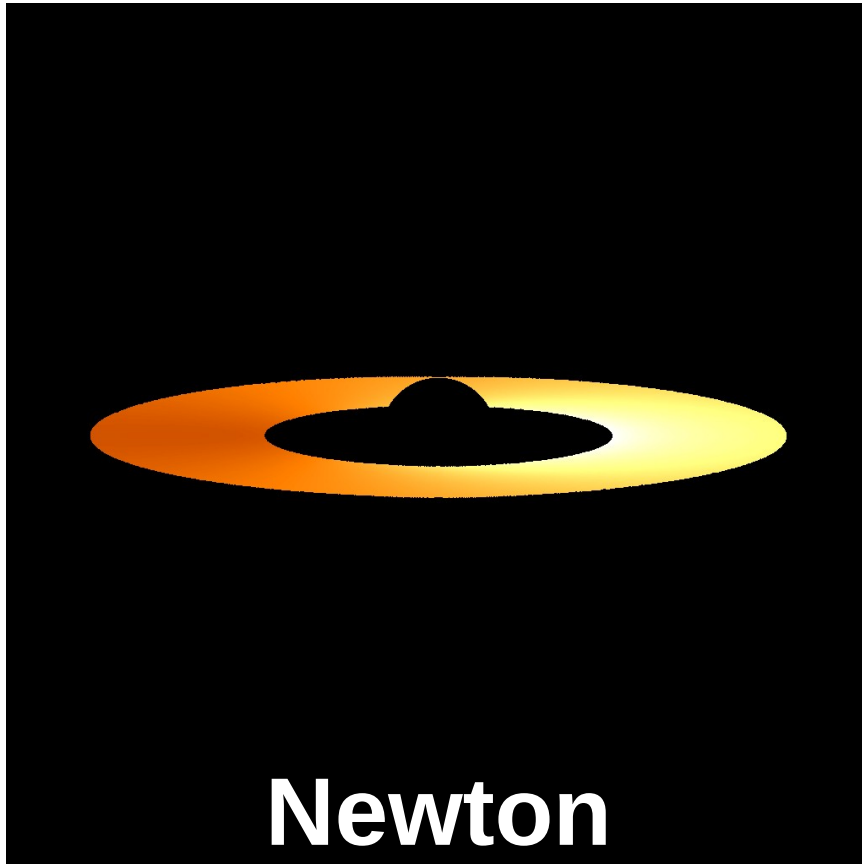


Fig. 4. Illustrative orbits in the plane $\{\phi=0\}$. Trajectory 1 has the critical impact parameter and circles infinitely around the black hole; trajectories 2 and 3 give direct images, trajectory 4 gives a secondary image

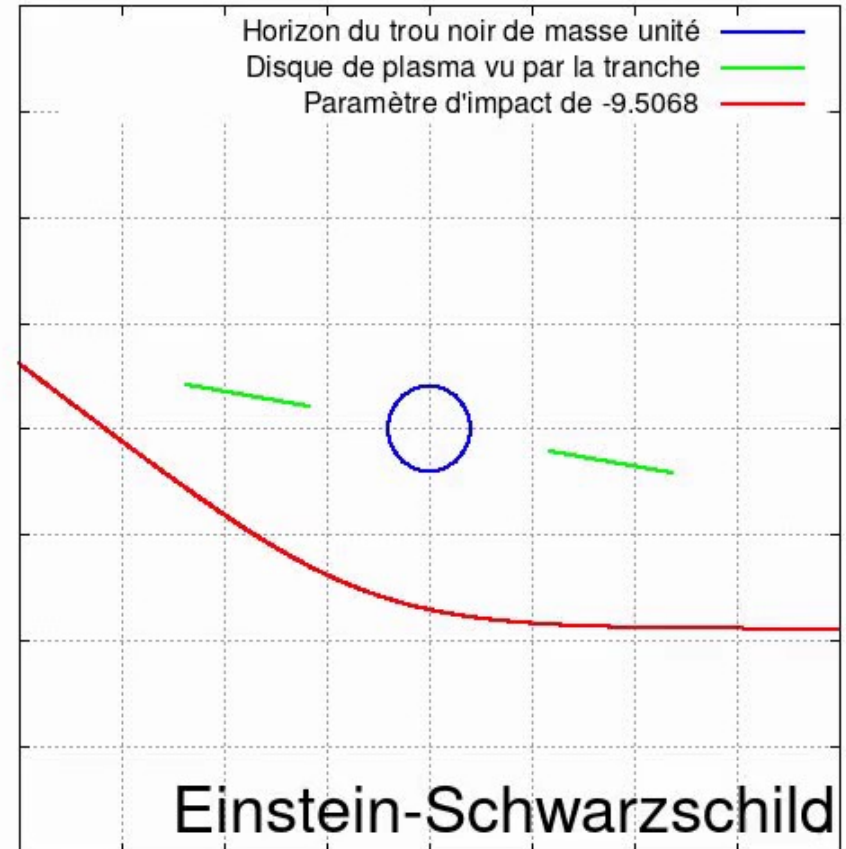
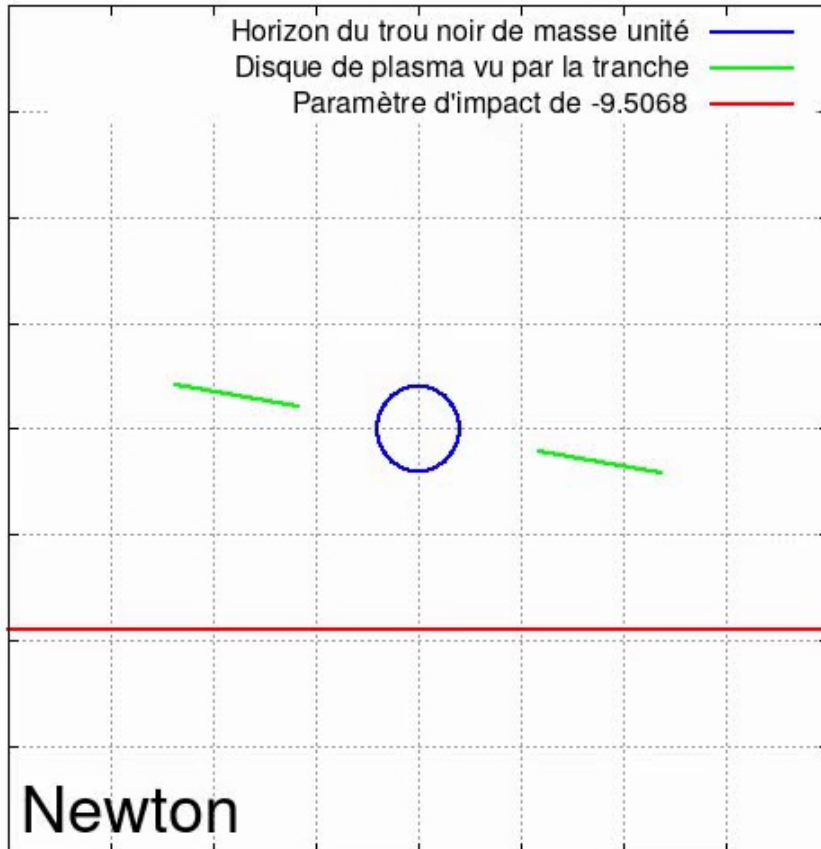


Calculer les trajectoires inverses

Entre Newton et Einstein

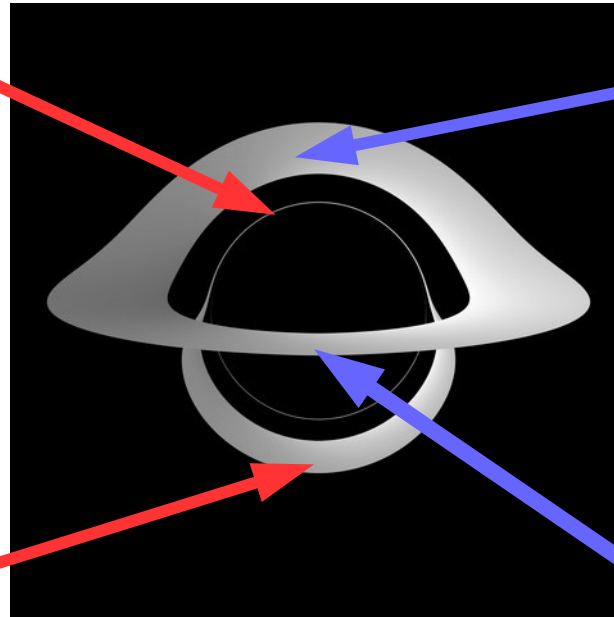
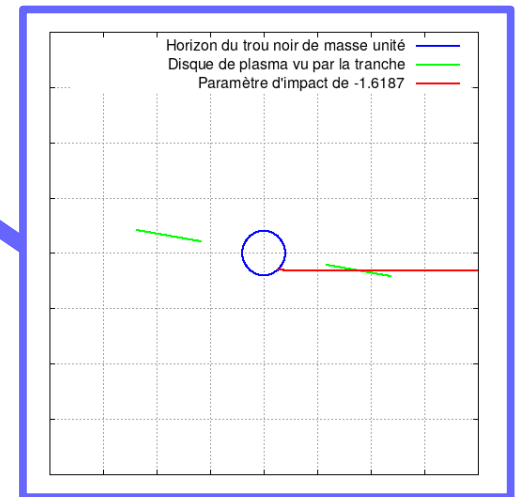
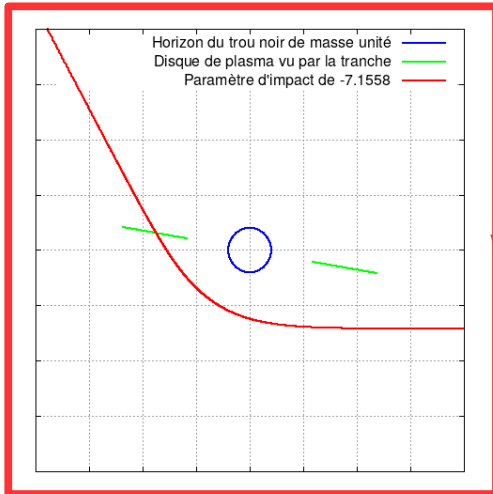
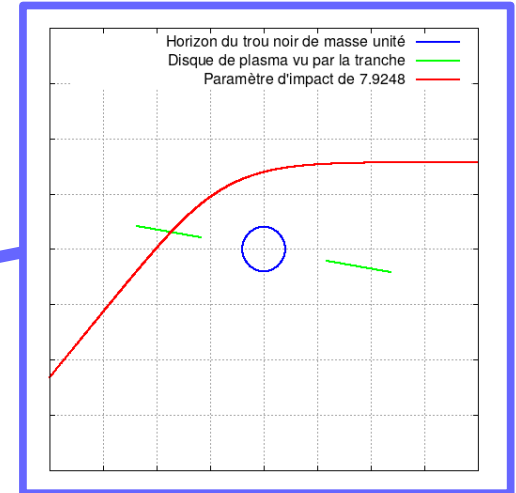
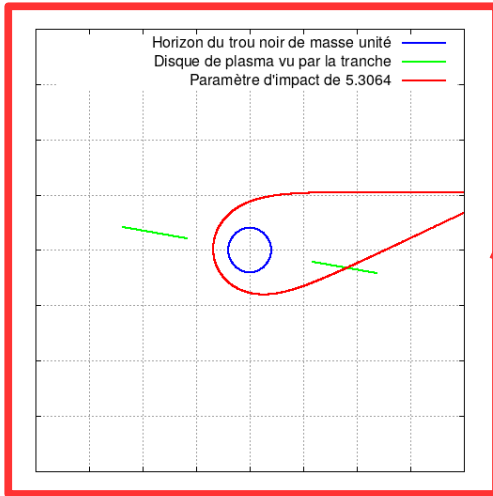


Et ça donne quoi pour tous les paramètres d'impacts ?

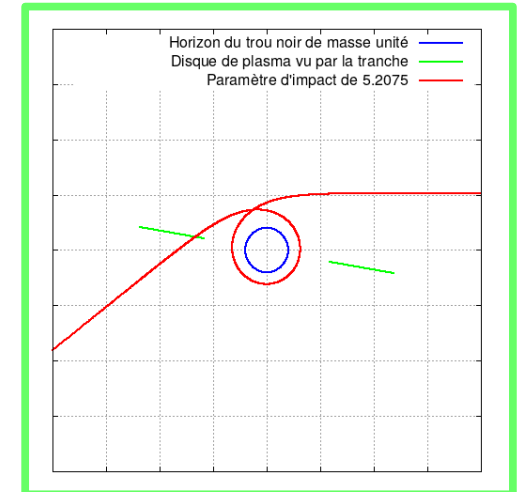
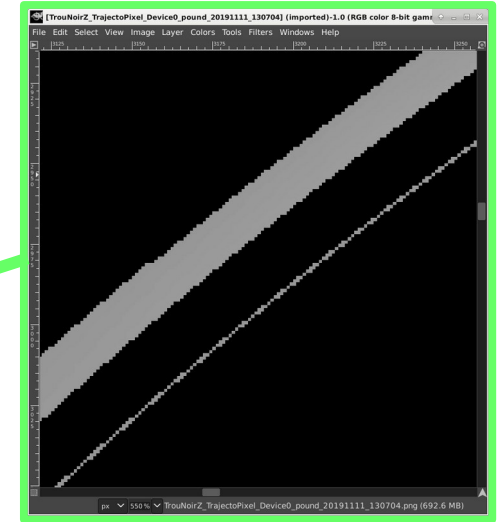
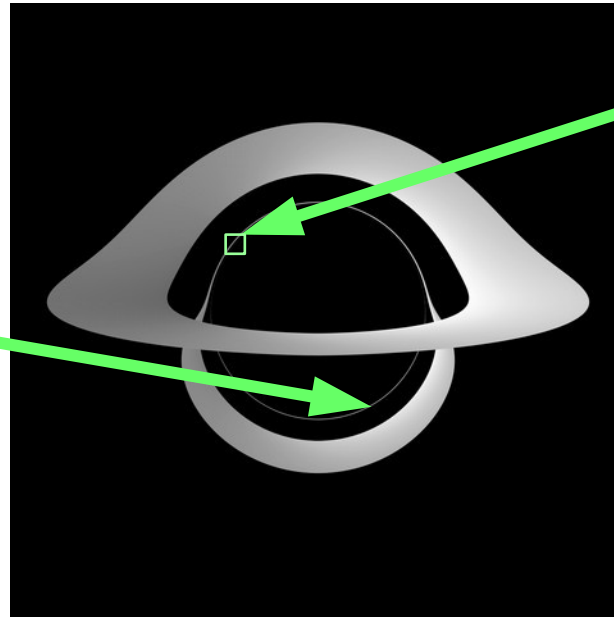
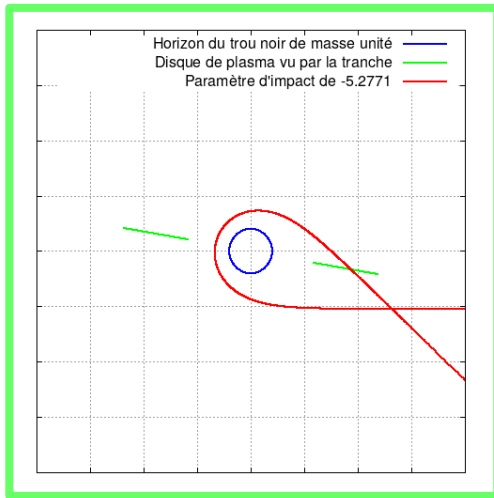


Copyright James MYLQ 2020

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



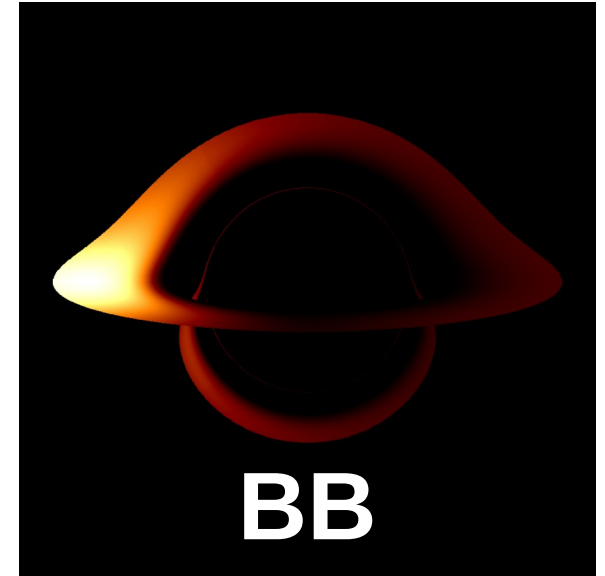
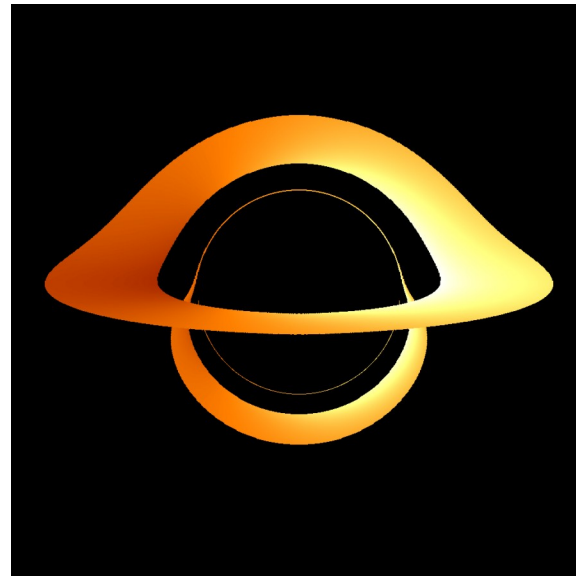
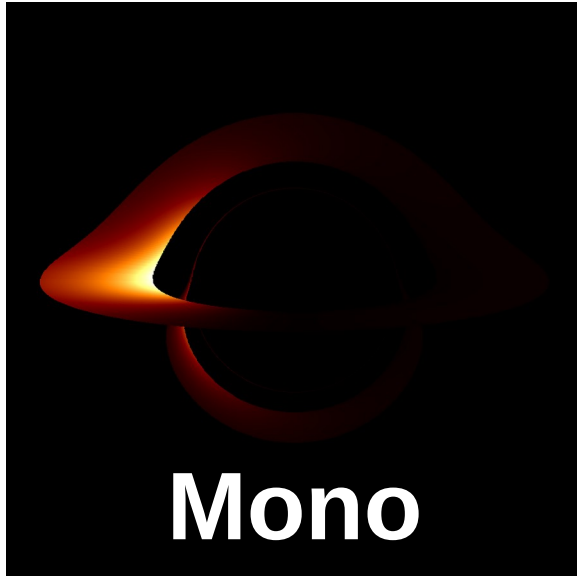
Quelques cas particuliers : le redessus, dessus et dessous



A chaque impact, deux physiques Monochrome & Corps noir

« Puissance 4 » sur z

Spectre de planck



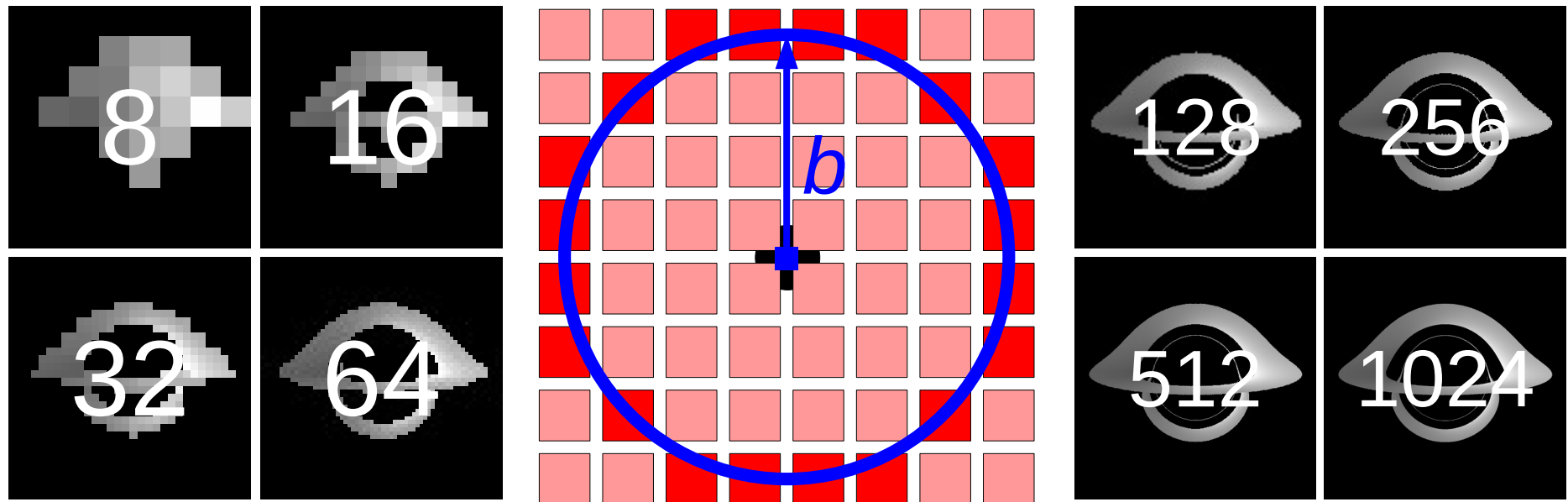
La méthode : « lancer des rayons » de l'oeil au disque de plasma

- 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 :
 - Aucune 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 #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

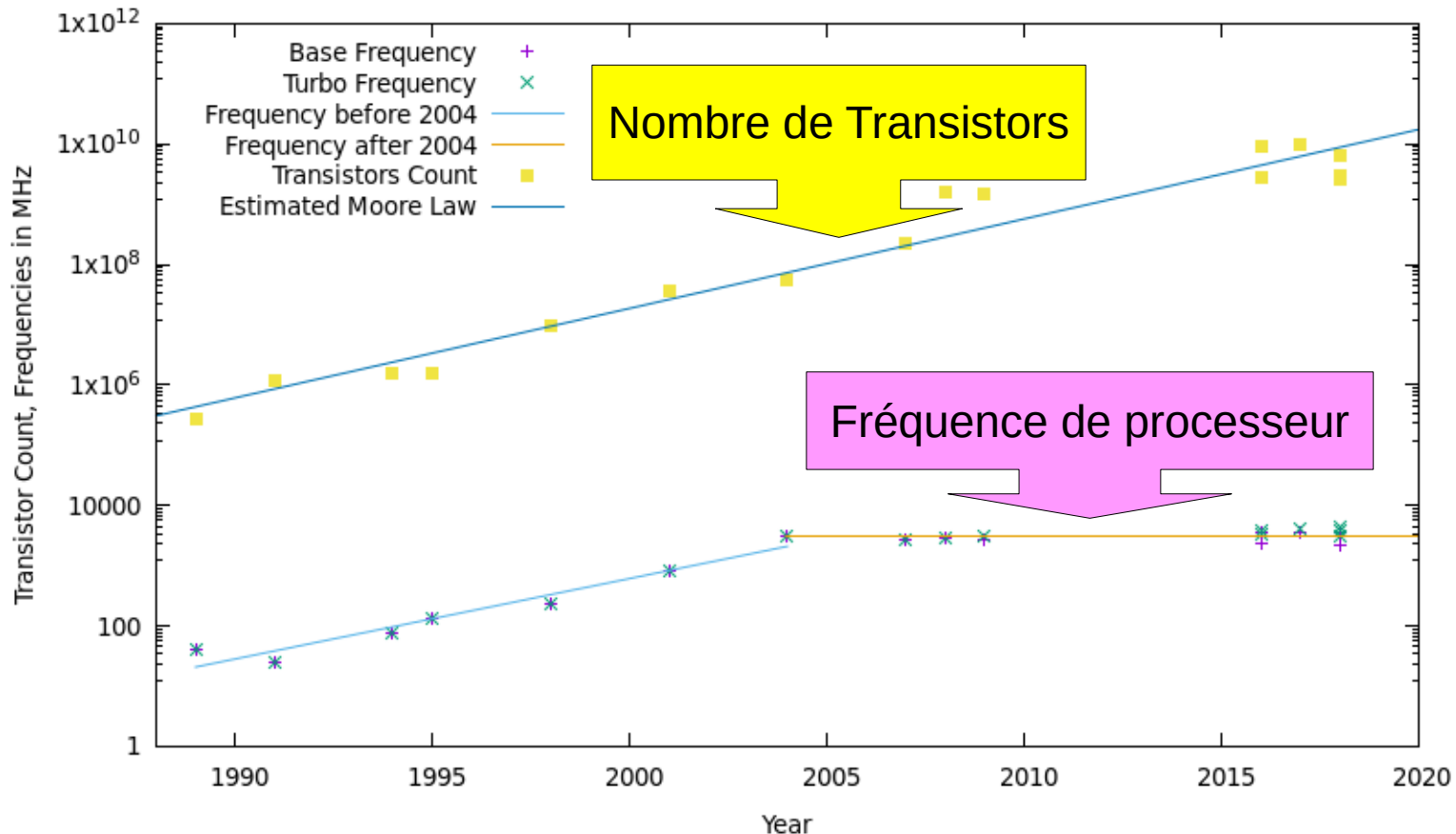
Banc de test sur les 16 CPUs

- 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^{14}
 - Sauf pour les très très vieux CPU : limitation à 256x256
- Méthodes : 2 à explorer avec « charges » différentes
 - Charge calculatoire faible : « Monochromatique » (ak Mono)
 - Charge calculatoire élevée : « Corps Noir » (aka BB)
- Statistiques : 10 lancements successifs
 - Exploitation de la médiane pour le « Elapsed Time »

Distribution de CPU pertinente ?

Transistors & Fréquences...

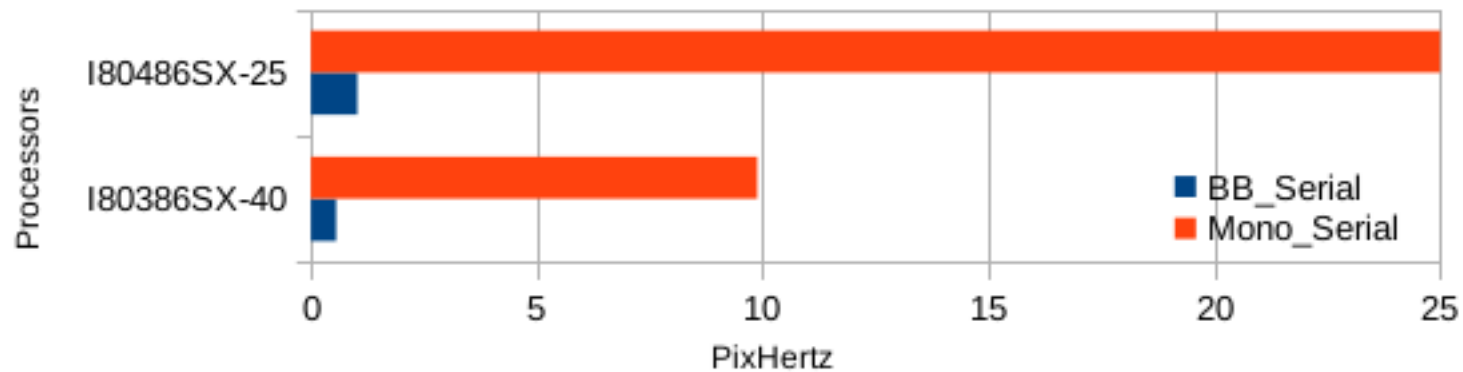
CPU Systems : Number of Transistors, Base Frequency, Turbo Frequency



Doublement des transistors tous les 2 ans...

L'ère préFPU...

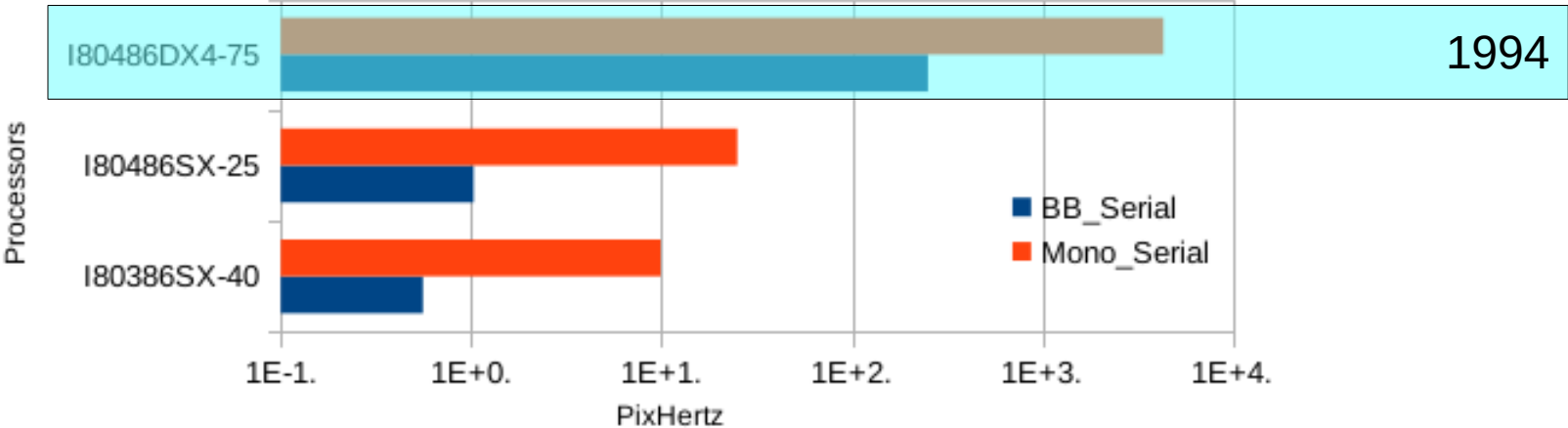
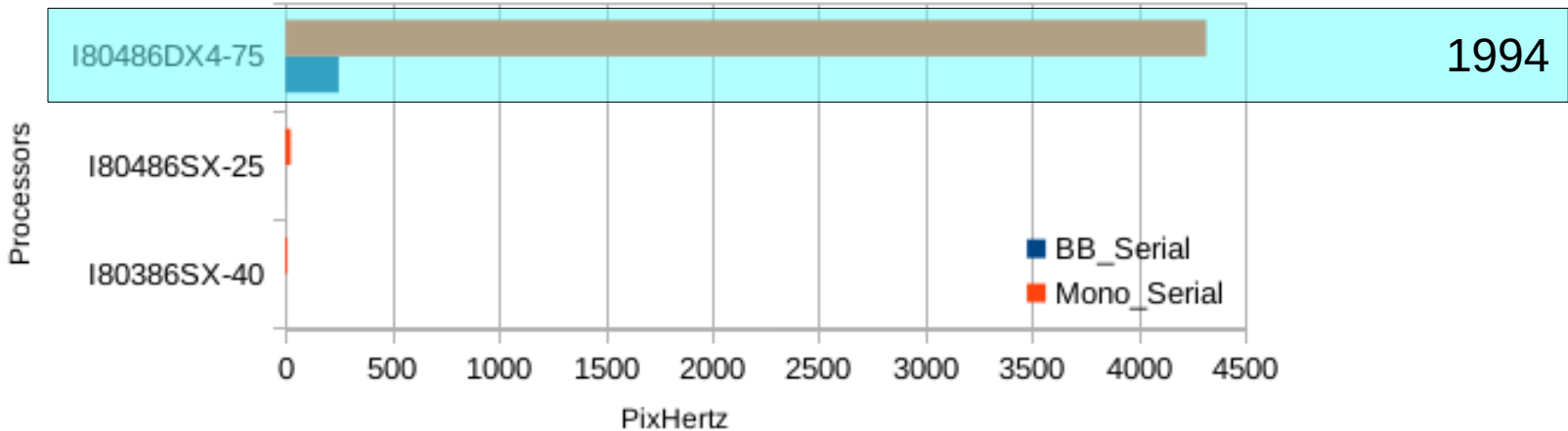
Le 80386SX et le 80486SX



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

Le FPU là, les fréquences grimpent

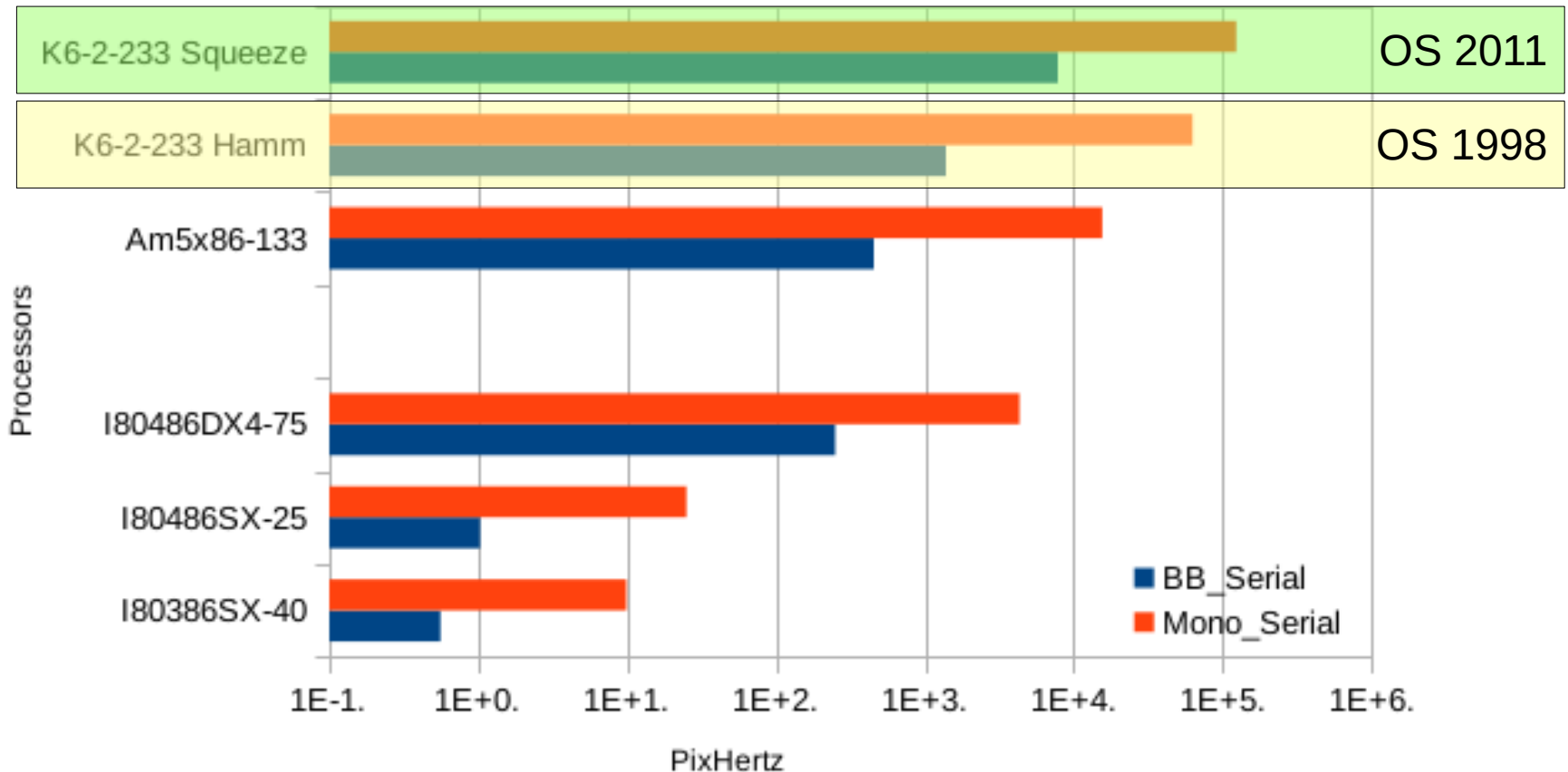
#1 : Le 80486DX4 à 75 MHz



x240 en performance : « des minutes en secondes »

Le RISC & la vectorisation

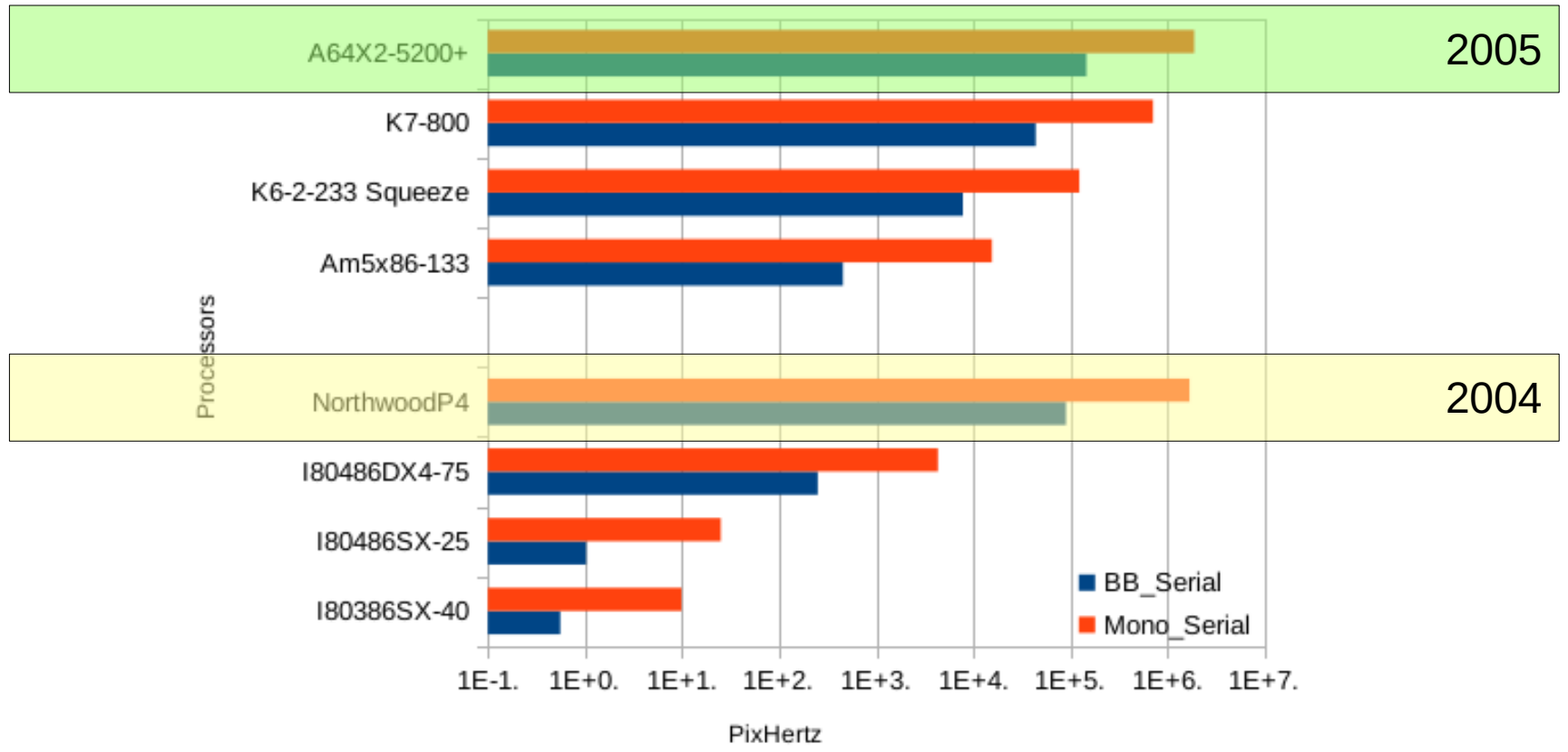
#2 & #3 : le AMD K6-2 à 233 MHz



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

Le multicœurs : logique & physique

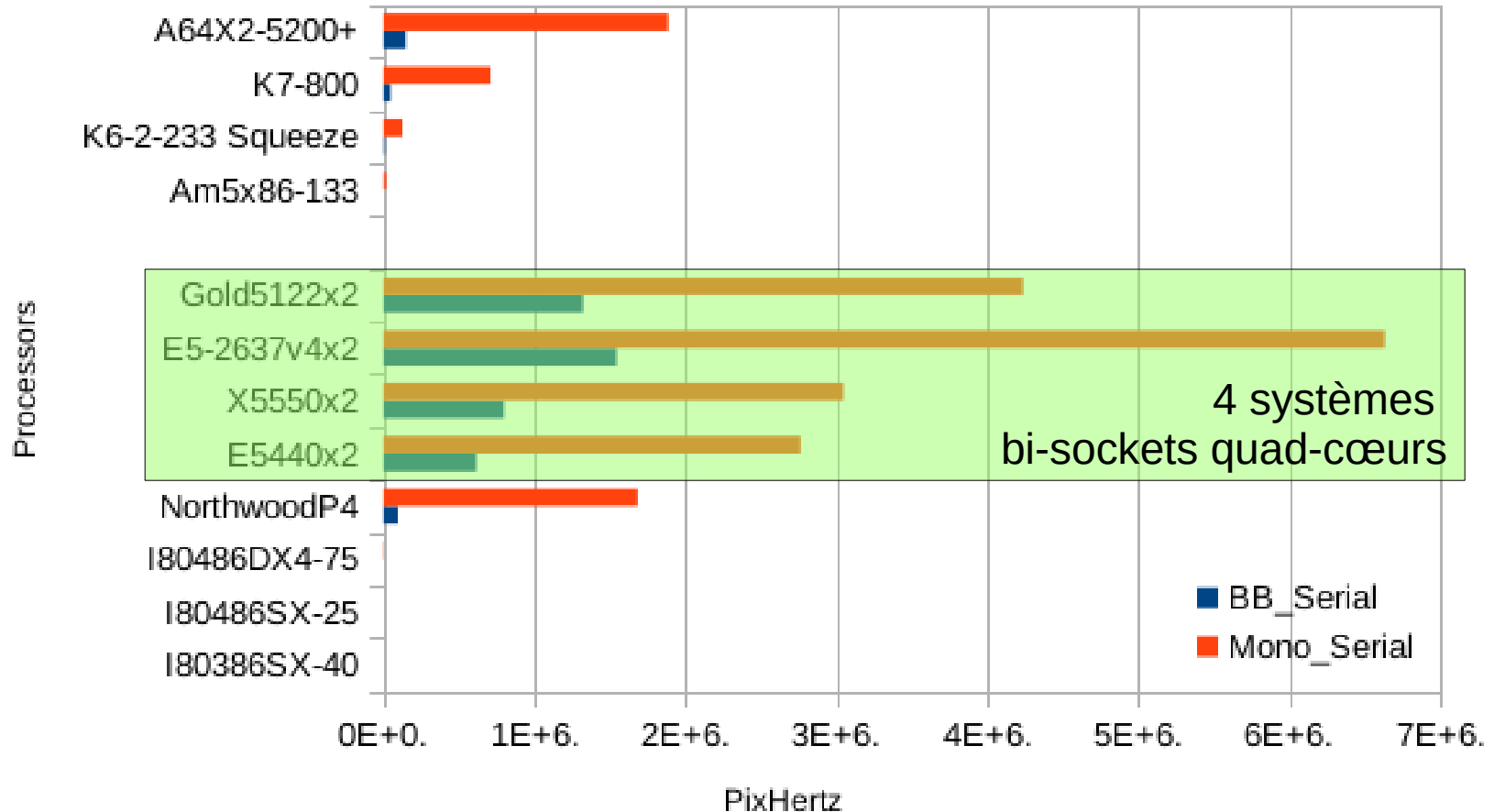
#4 : le Northwood Intel, l'Athlon64x2



Un facteur 10 à 15 entre en 8 ans

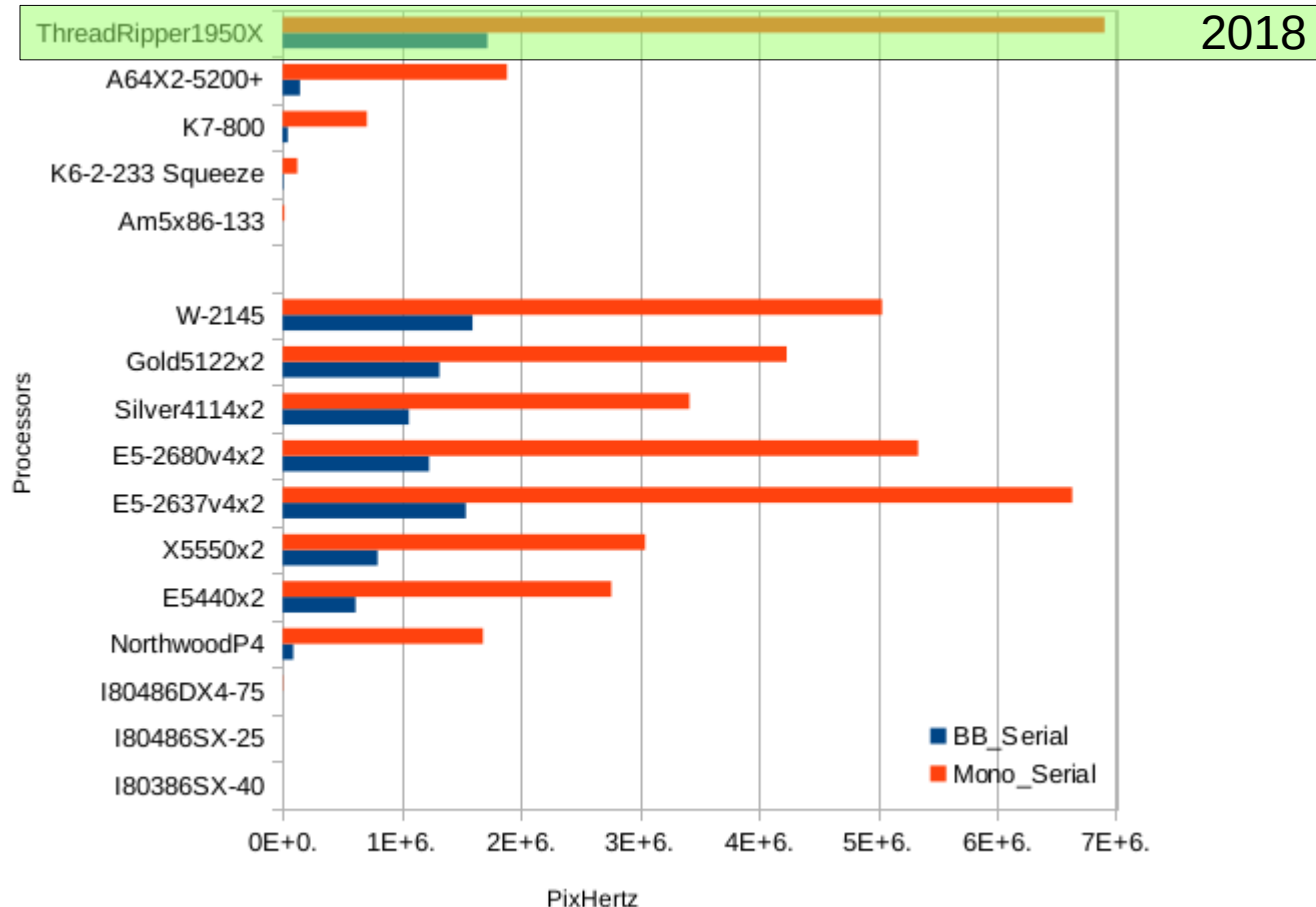
Programme séquentiel, pas d'exploitation des cœurs...

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



Un facteur 3 en 15 ans : peut mieux faire !

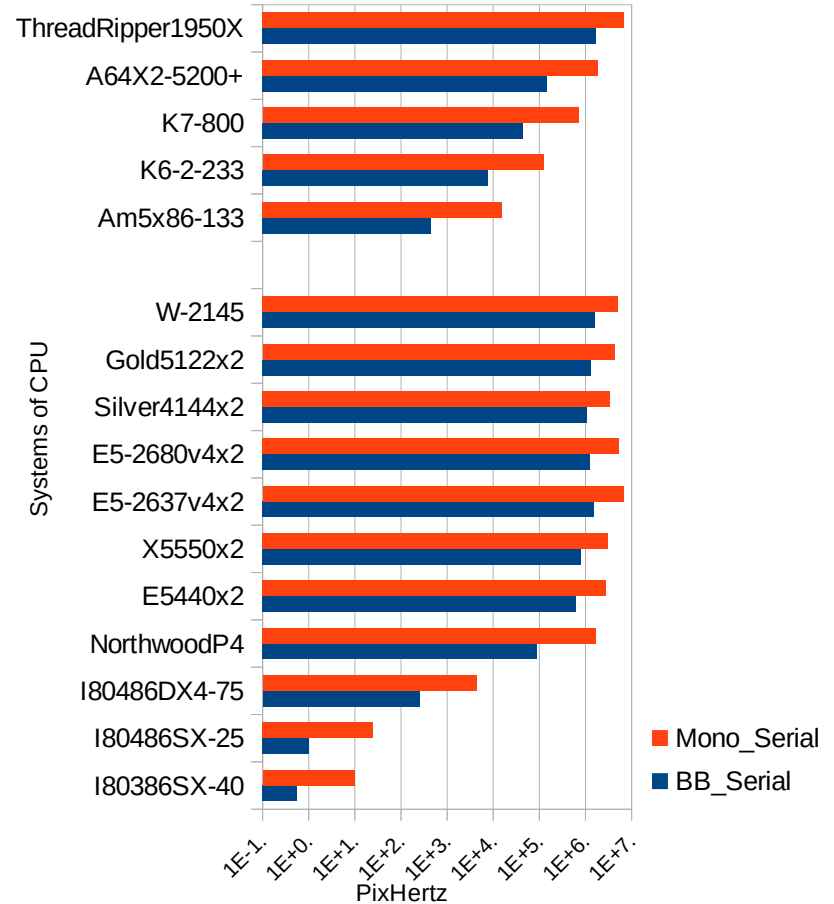
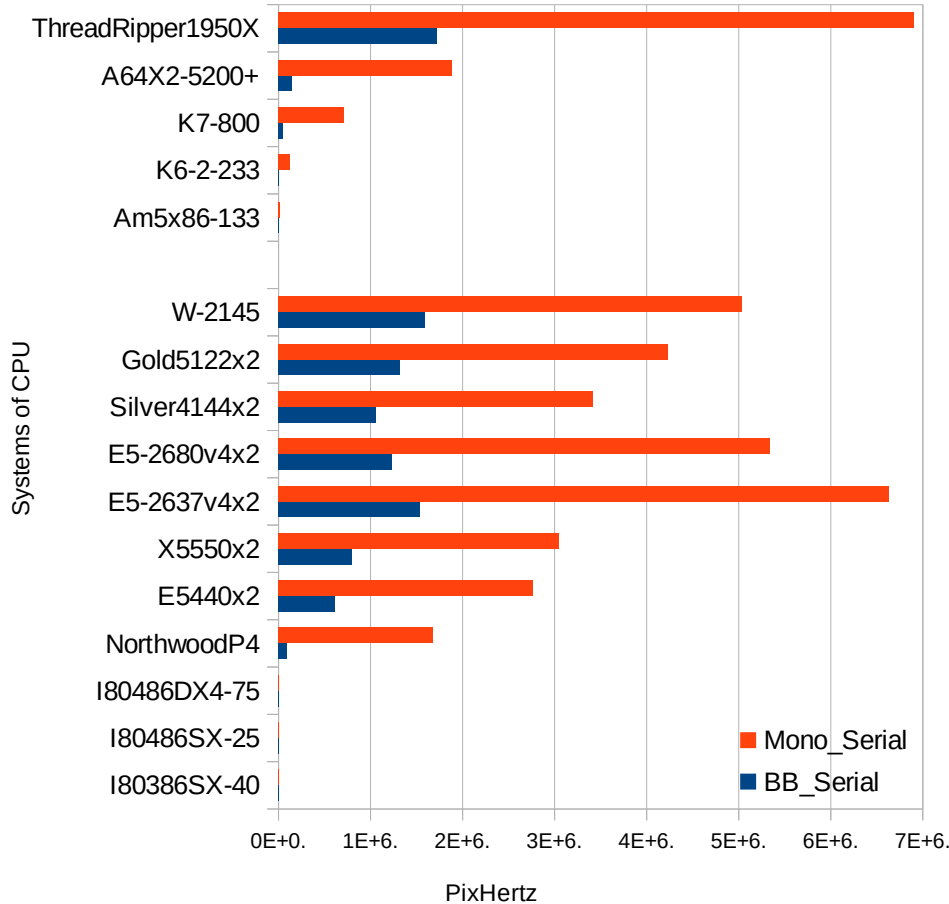
16 processeurs de 1989 à 2019 : « *and the winner is* »...



Le AMD Threadripper 1950X à 3.6 GHz

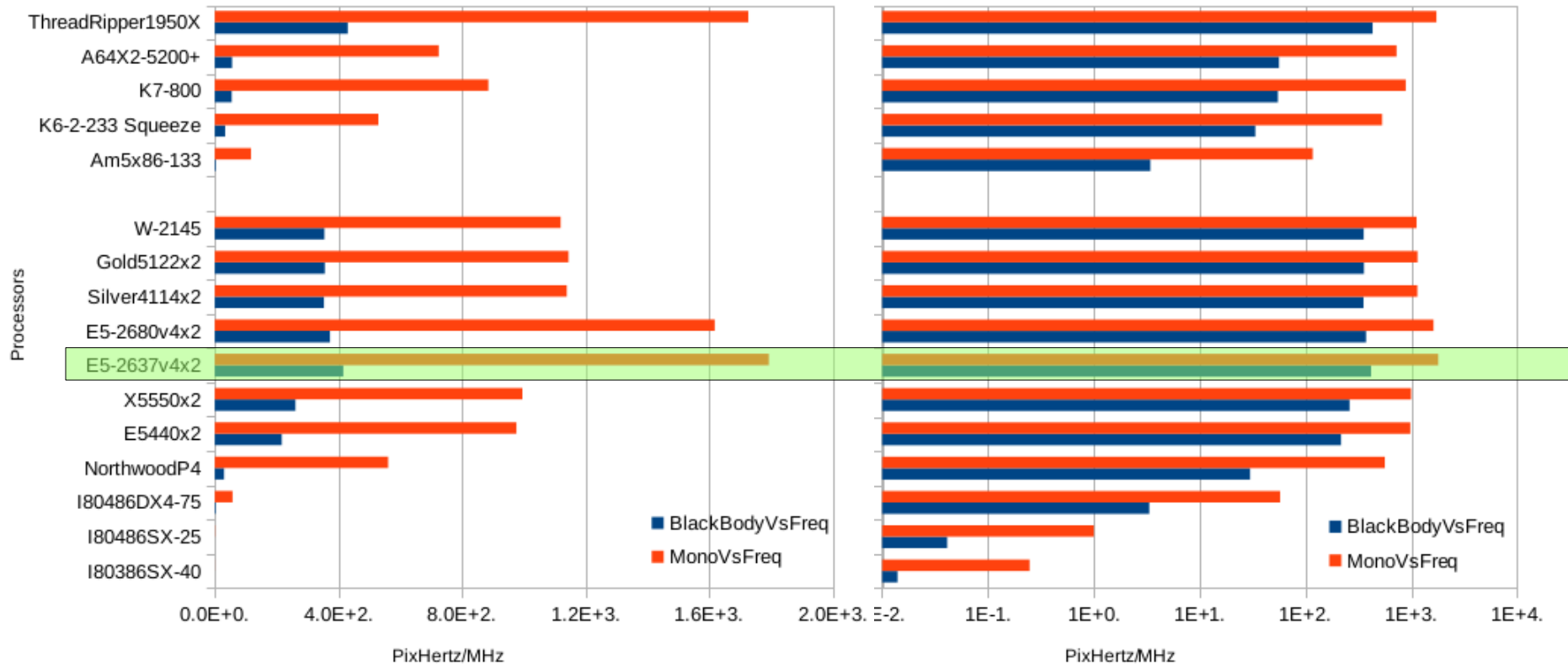
Exécution du code séquentiel

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



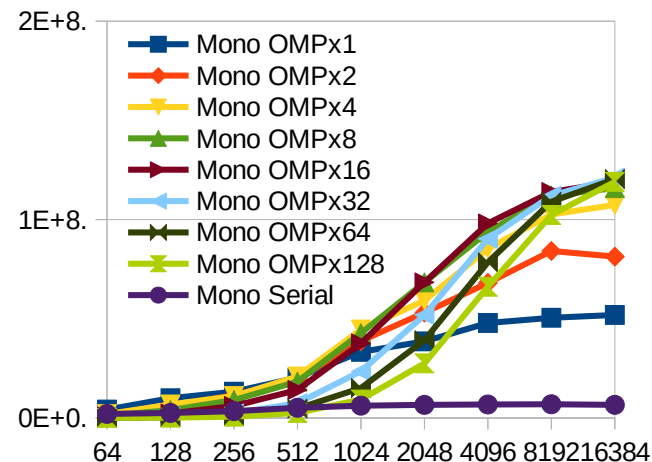
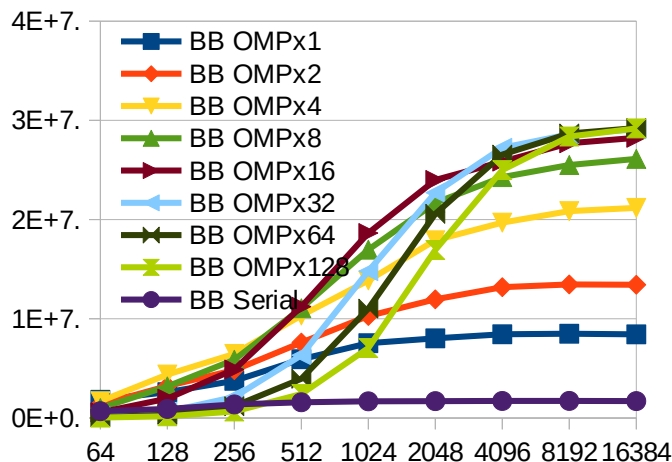
Sur 15 ans, entre un facteur 3 et un facteur 15...

Il va falloir trouver autre chose pour accélérer !

Parallélisation du code

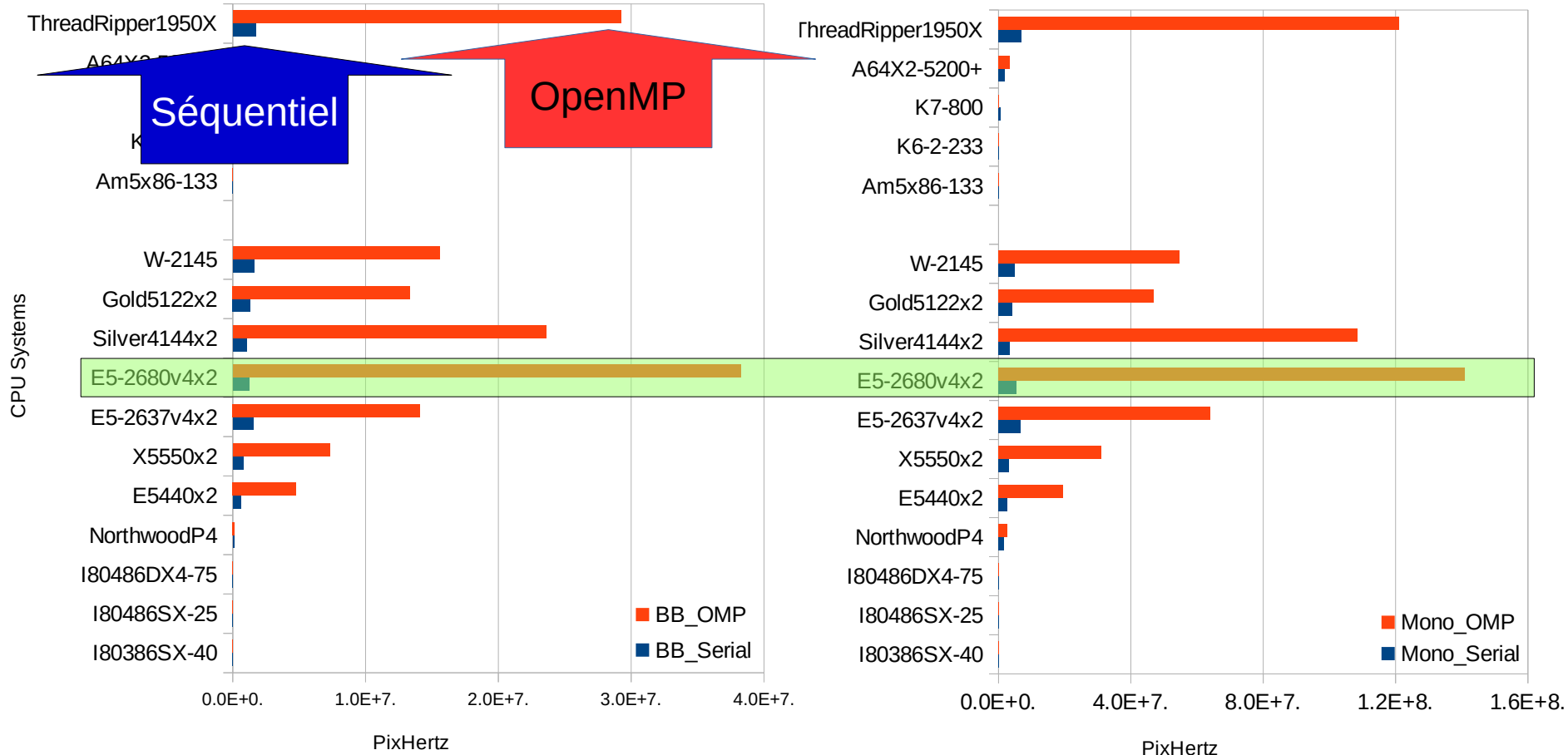
Passage en OpenMP & étrangement

- 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 » !

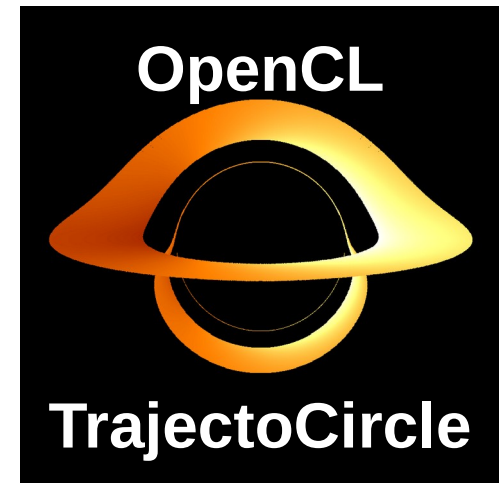
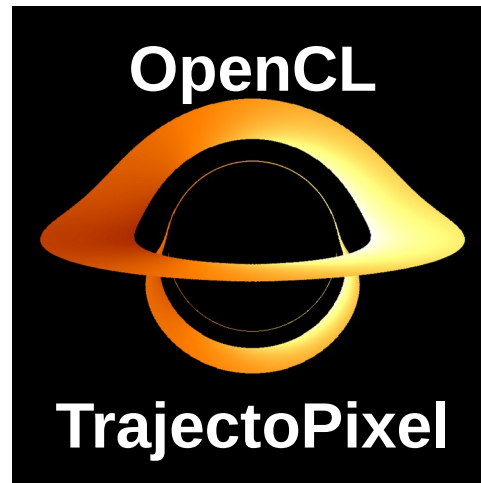
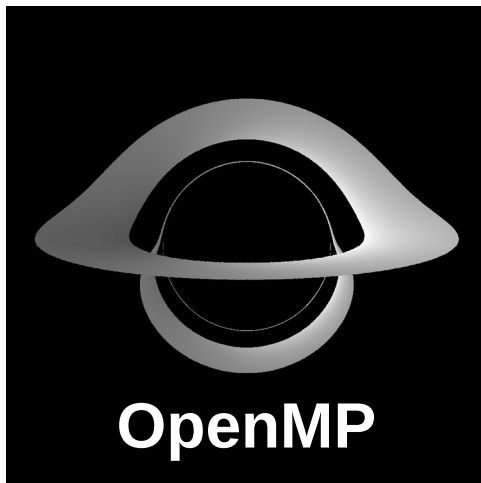
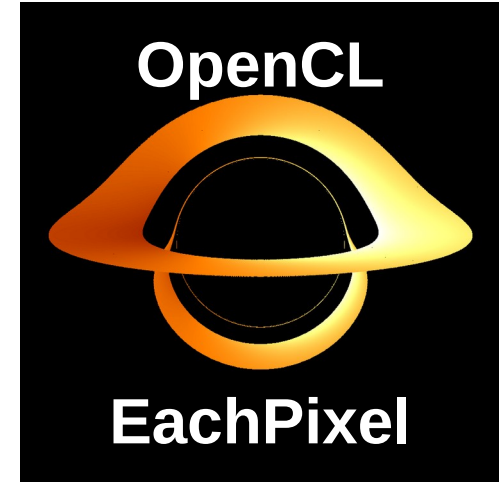
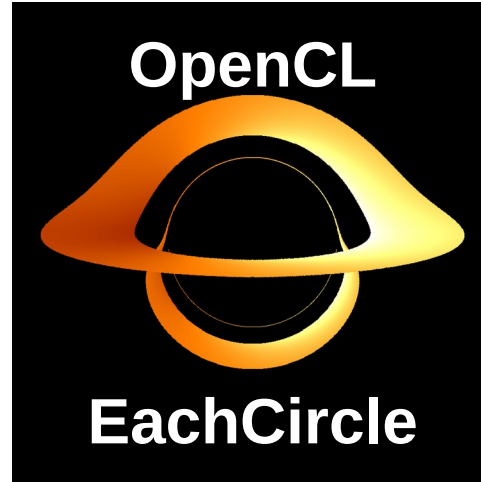
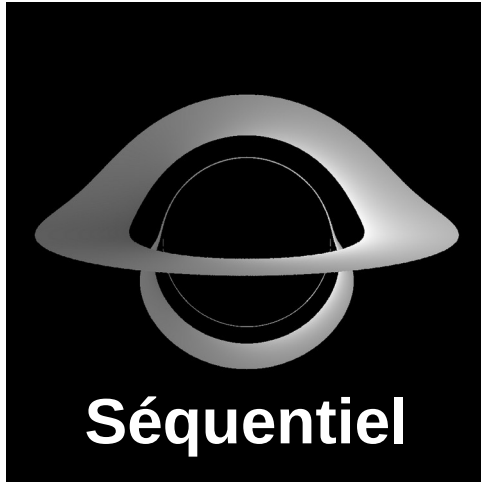


OpenCL

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 !

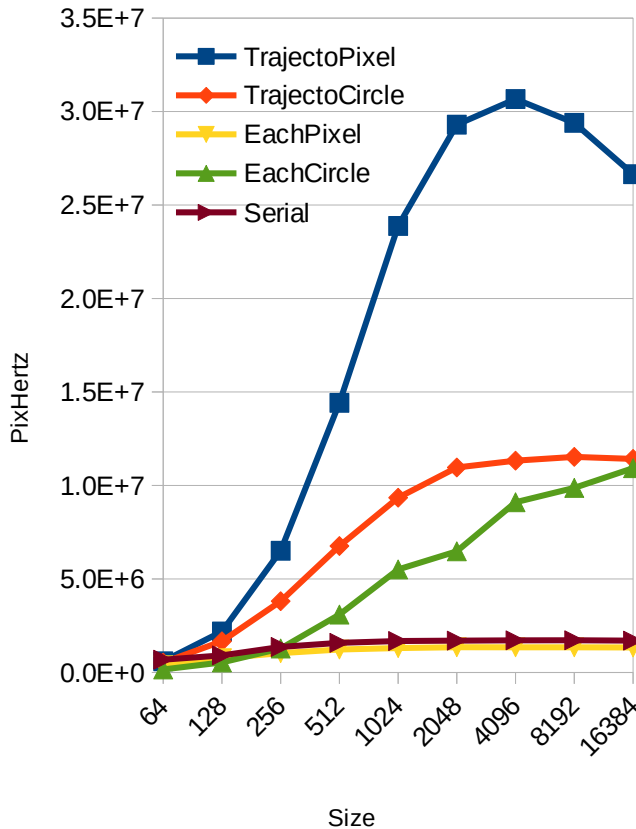
Obtient-on les mêmes résultats ? Laissons l'œil juger...



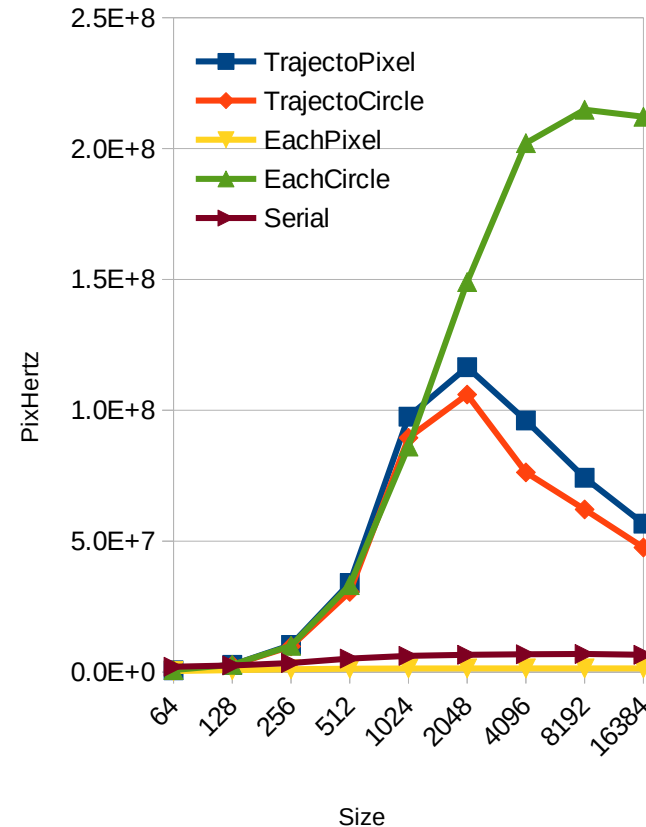
OpenCL sur Threadripper 1950x

La méthode de // importante...

BB on Threadripper 1950X with OpenCL



Mono on Threadripper 1950X

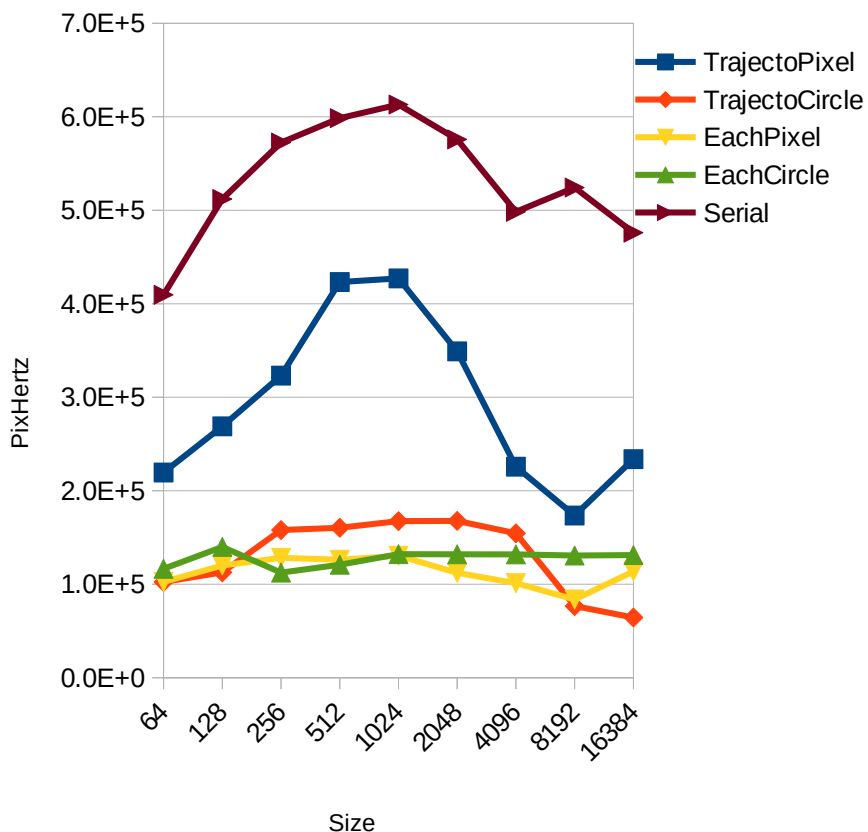


TrajectoPixel pour BB, EachCircle pour Mono...

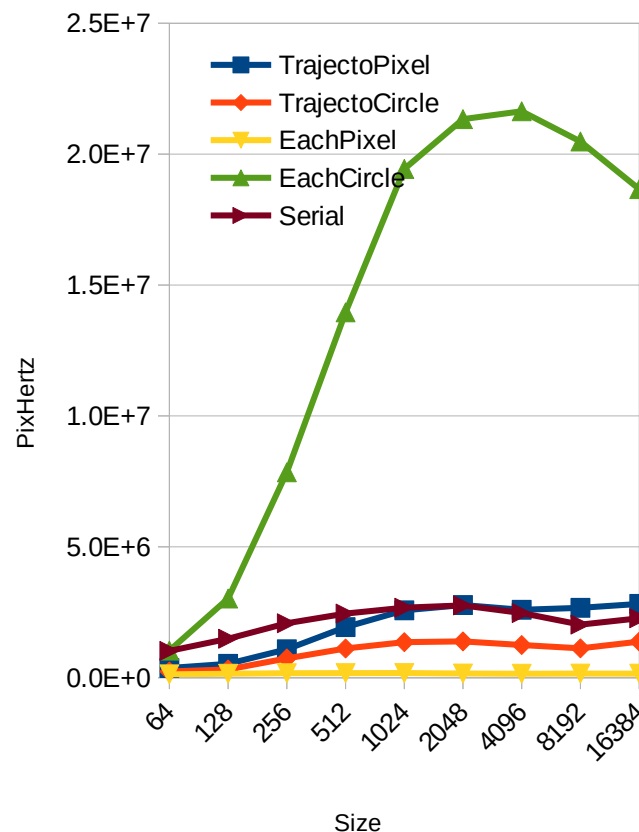
OpenCL sur Harpertown E5440x2

OpenCL (AMD) pas terrible

BB on Harpertown E5440 with OpenCL



Mono on Harpertown E5440 with OpenCL



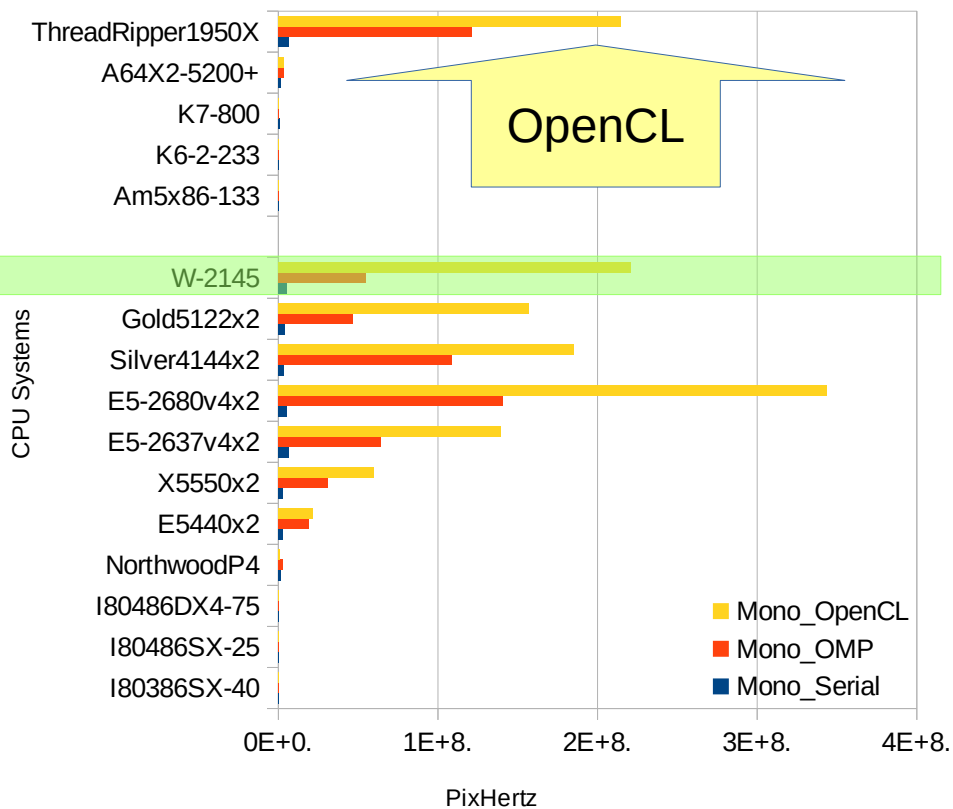
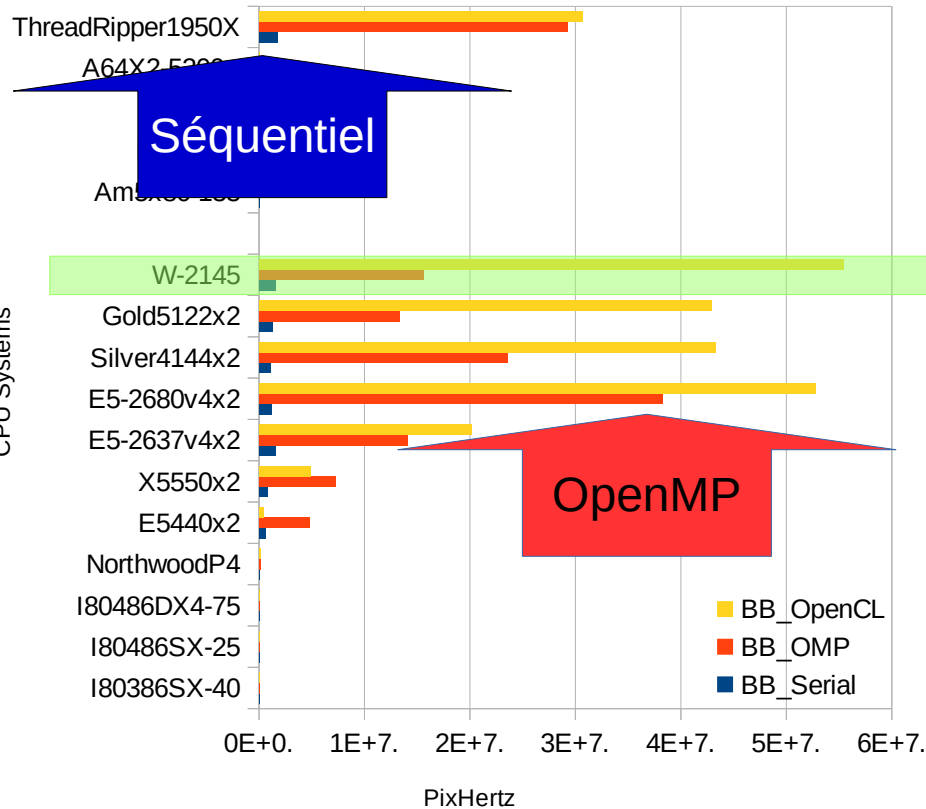
A chaque processeur, sa courbe de scalabilité...

Parallélisation OpenCL

Pour tous les processeurs...

BB with Serial, OpenMP, OpenCL

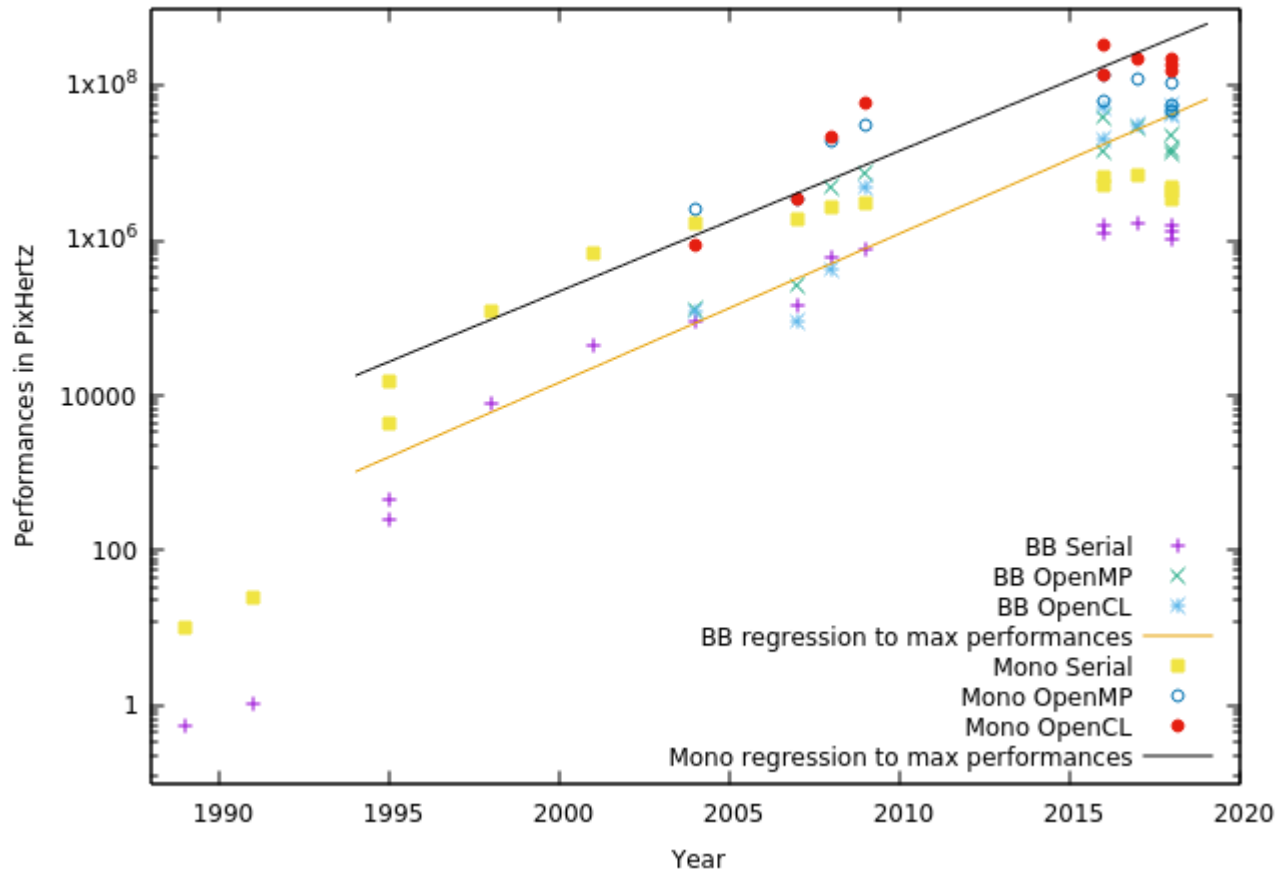
Mono with Serial, OpenMP, OpenCL



OpenCL Intel très efficace avec CPU Intel !

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

Performances on TrouNoir code : Serial, OpenMP, OpenCL implementations



Performance : x2 tous les 18 mois !

Une méthode : les deux temps...

time & /usr/bin/time

- 2 versions de « Time »
 - Dans le terminal : build in time
 - time sleep 10
 - real 0m10.003s
 - user 0m0.000s
 - sys 0m0.000s
 - Commande : /usr/bin/time
 - /usr/bin/time sleep 10
 - 0.00user 0.00system 0:10.00elapsed 0%CPU (0avgtext+0avgdata 640maxresident)k
 - 0inputs+0outputs (0major+207minor)pagefaults 0swaps
- Exploiter /usr/bin/time, MAIS mieux !!!

/usr/bin/time : le minimum...

La définition de la variable TIME

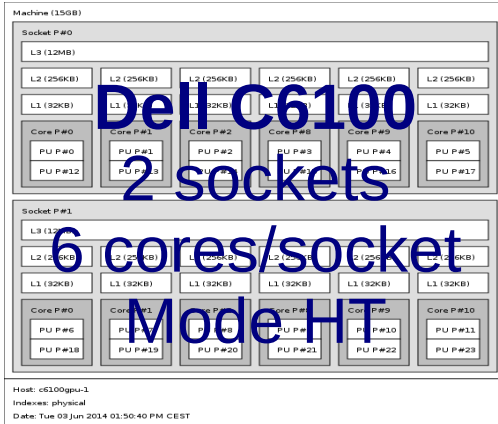
```
export TIME=""
TIME Command being timed: "%C"
TIME User time (seconds): %U
TIME System time (seconds): %S
TIME Elapsed (wall clock) time : %e
TIME Percent of CPU this job got: %P
TIME Average shared text size (kbytes): %X
TIME Average unshared data size (kbytes): %D
TIME Average stack size (kbytes): %p
TIME Average total size (kbytes): %K
TIME Maximum resident set size (kbytes): %M
TIME Average resident set size (kbytes): %t
TIME Major (requiring I/O) page faults: %F
TIME Minor (reclaiming a frame) page faults: %R
TIME Voluntary context switches: %w
TIME Involuntary context switches: %c
TIME Swaps: %W
TIME File system inputs: %I
TIME File system outputs: %O
TIME Socket messages sent: %s
TIME Socket messages received: %r
TIME Signals delivered: %k
TIME Page size (bytes): %Z
TIME Exit status: %x"
```

Petit exemple d'exécution

```
TIME Command being timed: mpirun -np 128 ./Pi_MPI_FP32_MWC 1000000000
TIME User time (seconds): 44.16
TIME System time (seconds): 8.84
TIME Elapsed (wall clock) time : 8.14
TIME Percent of CPU this job got: 651%
TIME Average shared text size (kbytes): 0
TIME Average unshared data size (kbytes): 0
TIME Average stack size (kbytes): 0
TIME Average total size (kbytes): 0
TIME Maximum resident set size (kbytes): 85116
TIME Average resident set size (kbytes): 0
TIME Major (requiring I/O) page faults: 307
TIME Minor (reclaiming a frame) page faults: 243144
TIME Voluntary context switches: 1184545
TIME Involuntary context switches: 903070
TIME Swaps: 0
TIME File system inputs: 0
TIME File system outputs: 304296
TIME Socket messages sent: 0
TIME Socket messages received: 0
TIME Signals delivered: 0
TIME Page size (bytes): 4096
TIME Exit status: 0
```

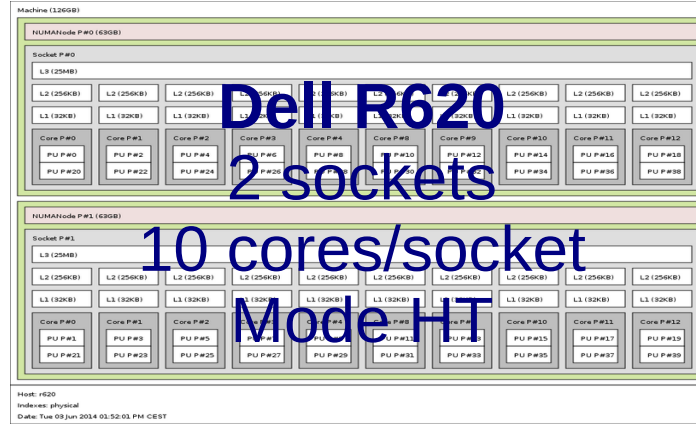
Bienvenue, Amdahl dans le monde réel !

10 machines de 2 à 40 cœurs...



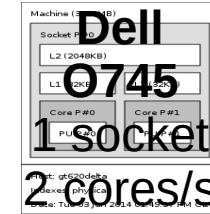
Dell C6100
2 sockets

6 cores/socket
Mode HT

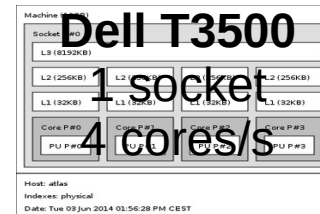


Dell R620
2 sockets

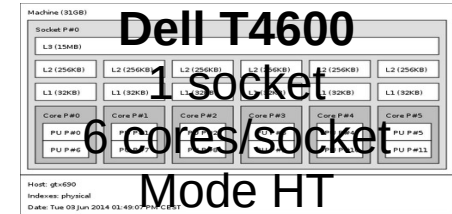
10 cores/socket
Mode HT



Dell O745
1 socket
2 cores/s

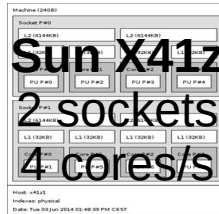


Dell T3500
1 socket
4 cores/s



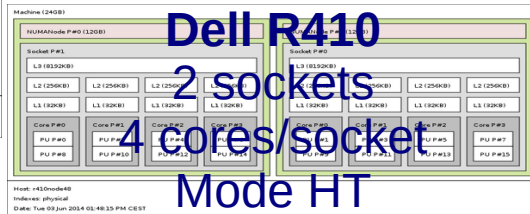
Dell T4600
1 socket
6 cores/socket
Mode HT

hwloc-ls comme
commande

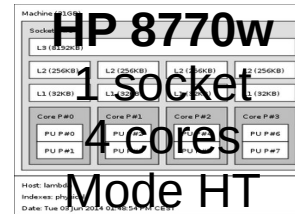


Sun X41z
2 sockets
4 cores/s

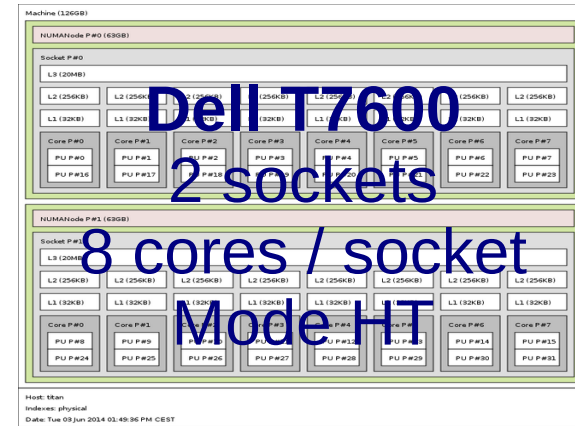
Sun X2200
2 sockets
4 cores/socket



Dell R410
2 sockets
4 cores/socket
Mode HT



HP 8770w
1 socket
4 cores
Mode HT

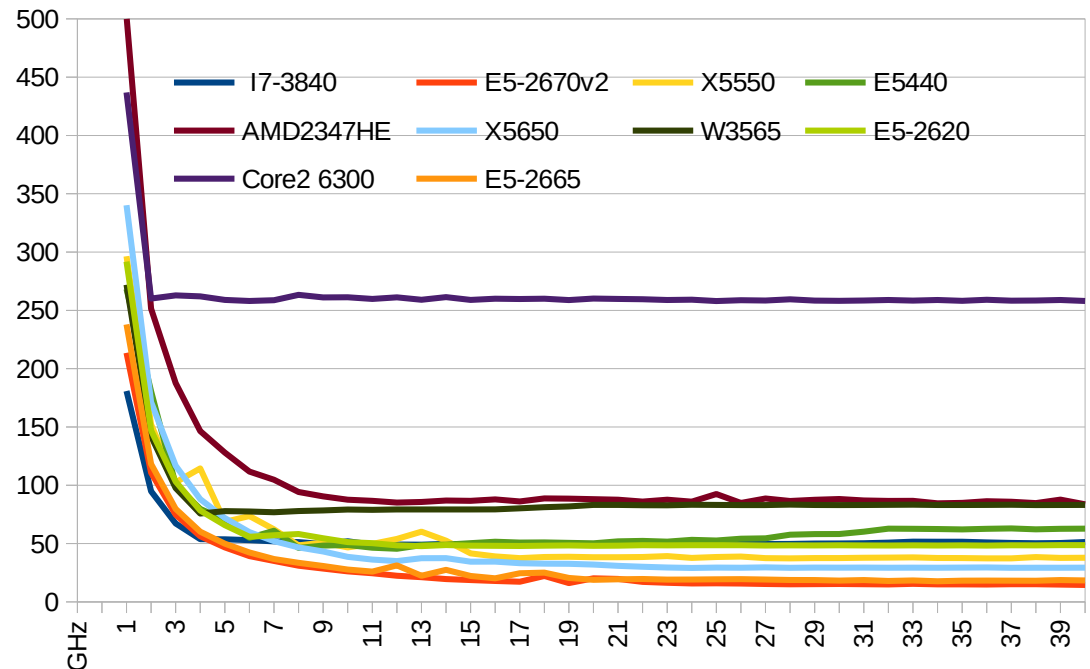


Dell T7600
2 sockets
8 cores / socket
Mode HT

Loi d'Amdahl dans le monde réel

Un banc de test : 10 CPUs, 1 code

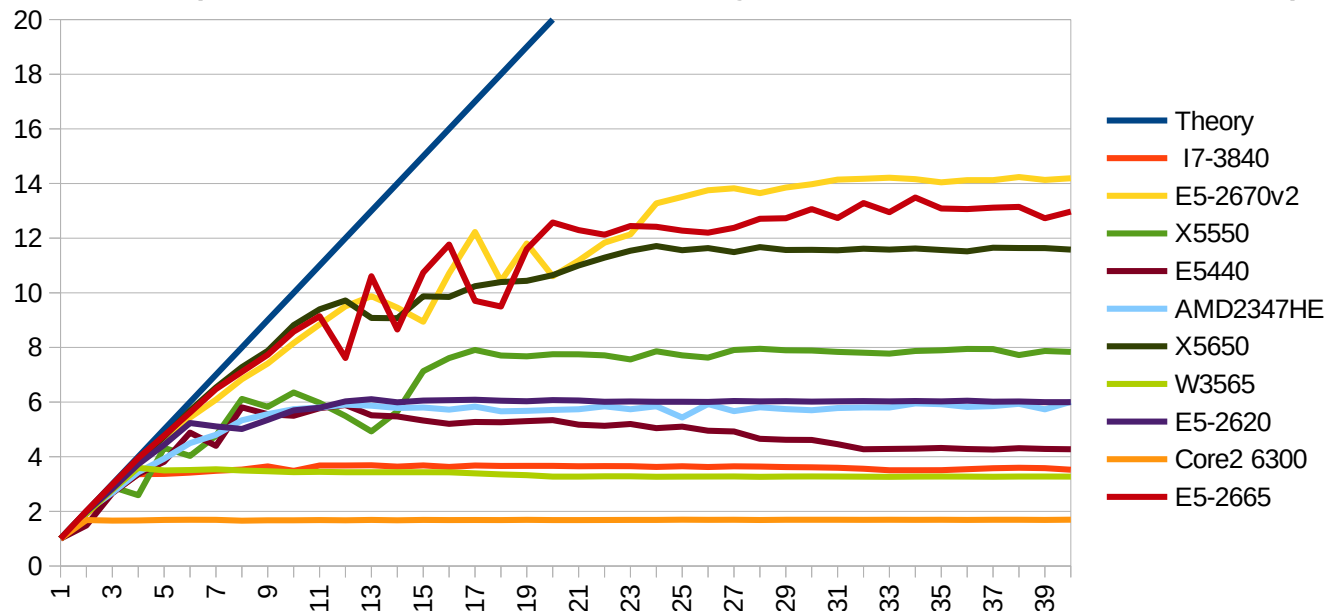
- Dans un nœud, 2 processeurs :
- 10 CPUs différents, de 2 à 20 cœurs (40 en HT)
- 1 application : pbzip2 (parallel bzip2)
- 1 donnée d'entrée : film encodé (1.4 GB) (pire scénario)
- De 1 processus to 80 processus
- Outil de métrologie : temps
- Observable : temps écoulé



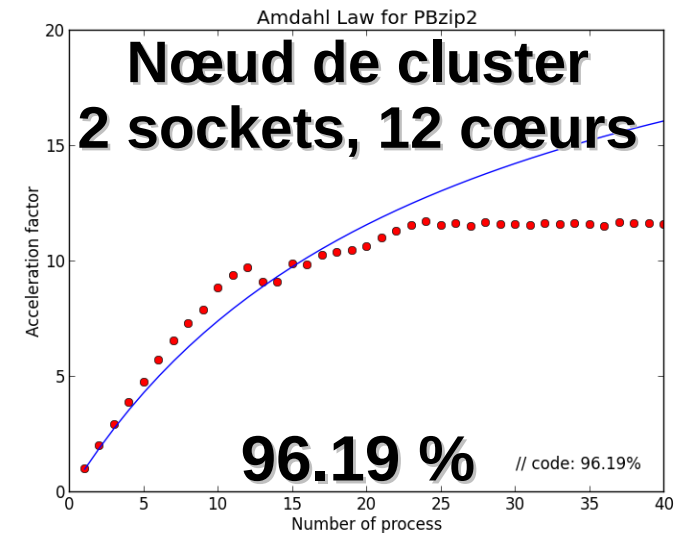
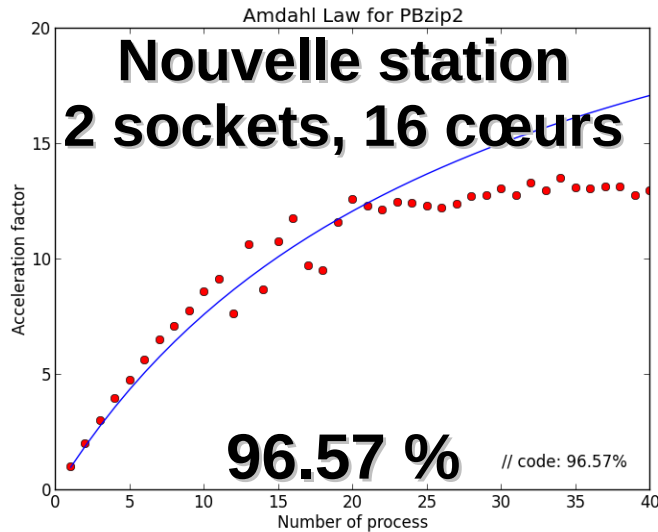
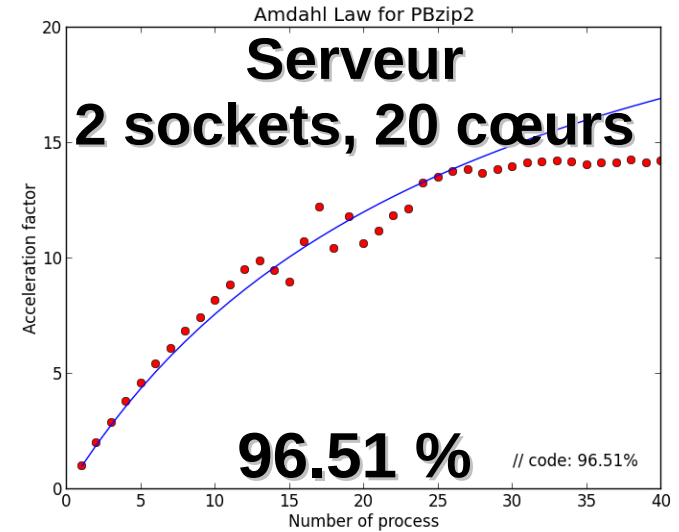
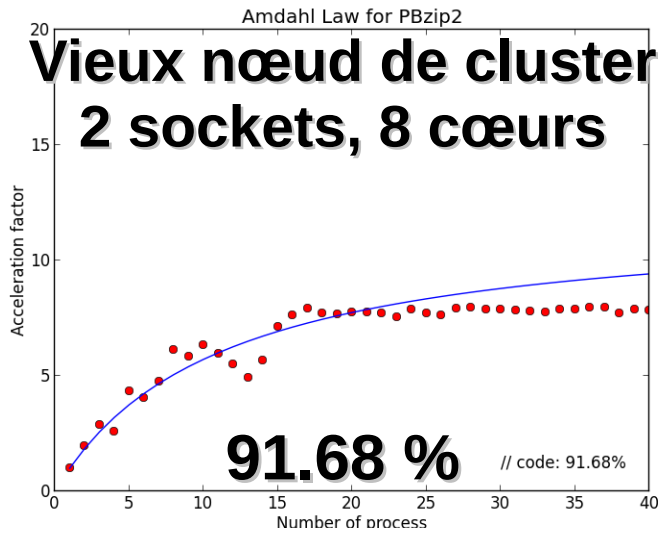
Loi d'Amdahl dans le monde réel

Accélération & Variations

- Symptômes :
 - Sur un large nombre de cœurs, une efficacité de 70 % to 80 %
 - Grandes variations sur les machines double sockets
 - Décroissance de performances sur les charges élevées avec les vieux processeurs

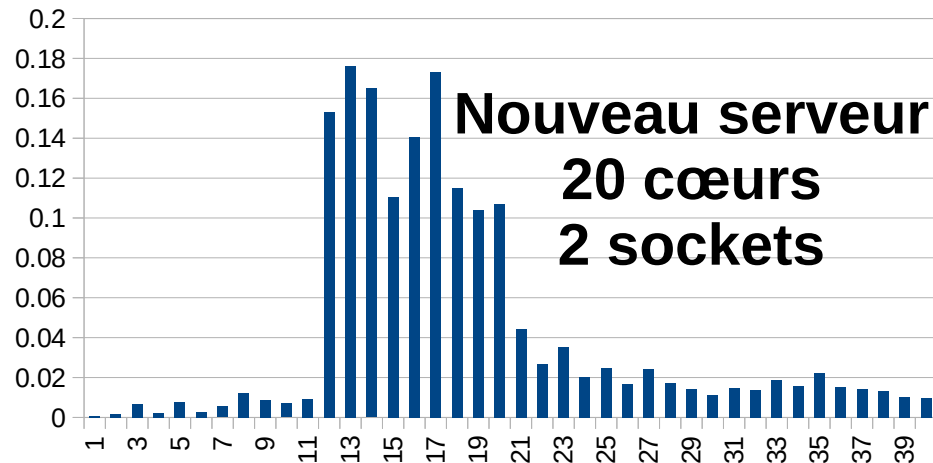
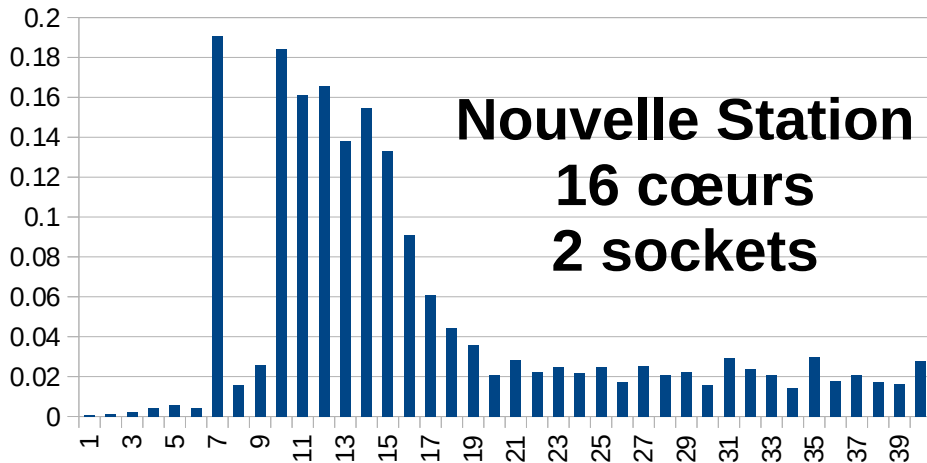
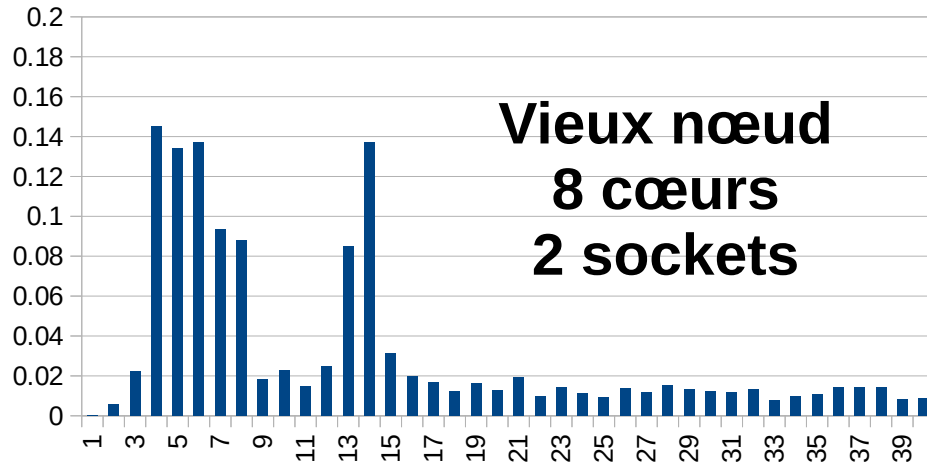


Loi d'Amdahl : « Fitting images » !



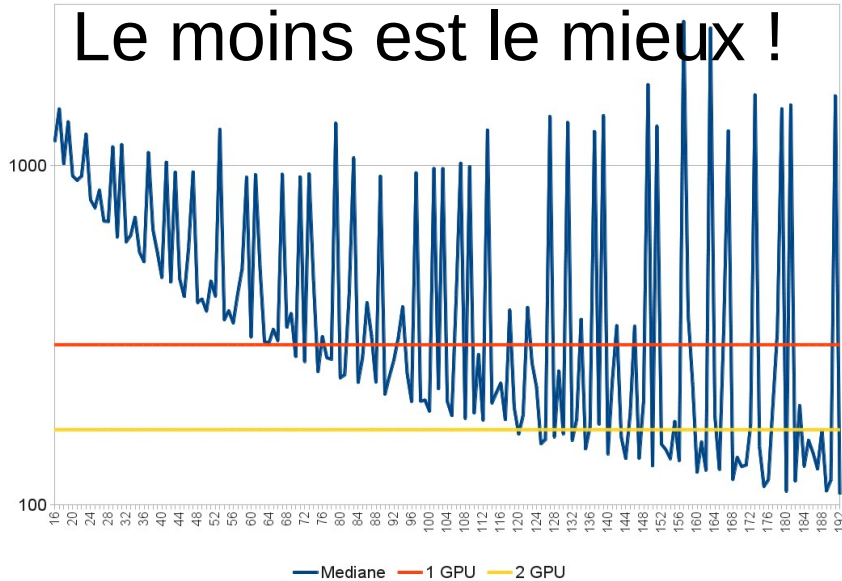
Et pire, la variabilité

Variabilité = Stddev/Median



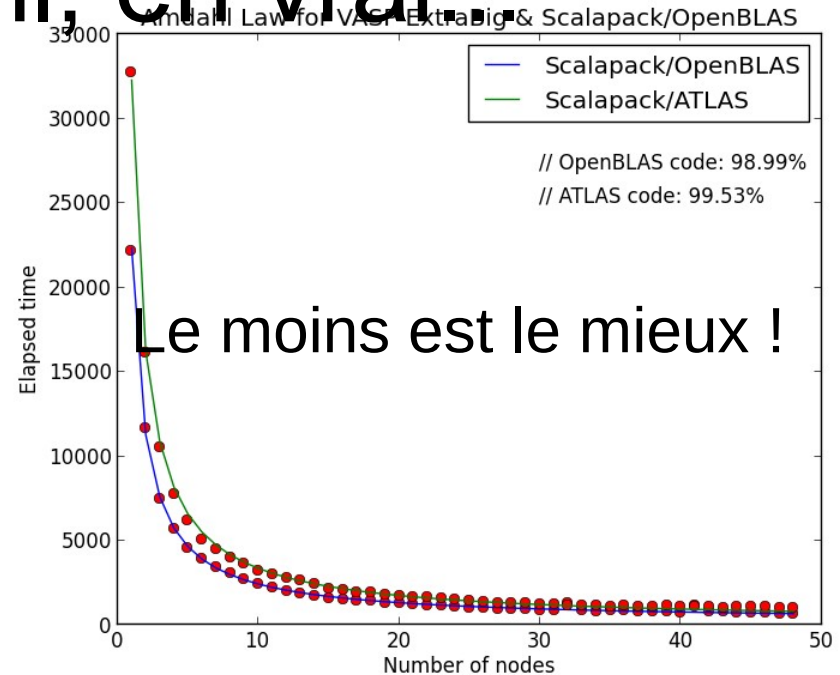
Applications MPI, retour à la science

Loi d'Amdahl, en vrai.



Lammps, application de dynamique moléculaire

- De 16 à 192 NP
- Comparaison à des GPGPU
- 99.96 % //ized (record !)
- À l'époque, 2GPGPU équivaut 120 NP



Application DFT VASP

- De 1 à 48 NP
- 99 % à 99.5 % //ized
- OpenBLAS vs ATLAS...

Et sur un code astrophysique ?

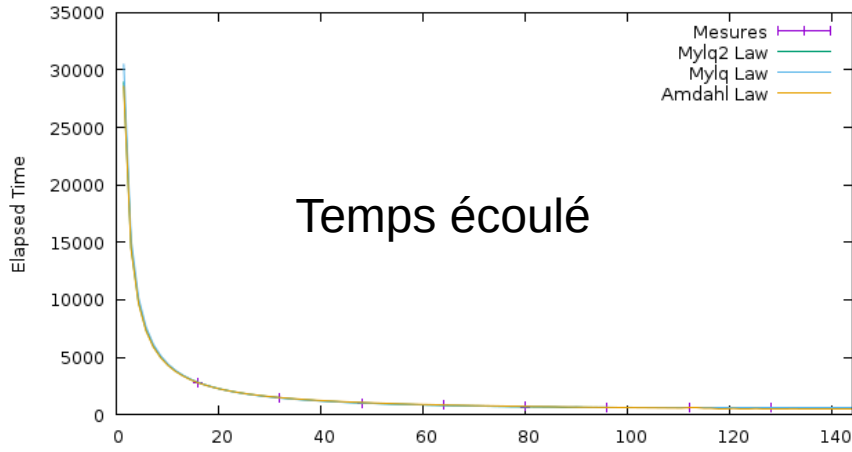
PKDGRAV3

- Cluster : 8 nœuds C6320, GE & Infiniband FDR
- Nœud : 2 sockets avec 8 cœurs chacun, 128 GB RAM
- OS : Debian Stretch sur SIDUS, kernel 4.
- Dépendances logicielles :
 - OpenMPI, Pthreads, FFTW
- Entrée : test exemple cosmology.par, Grid de 128^3 à 256^3
- Exploration de 1 à 8 nœuds :
 - MPI : NP de 16 à 128, OMP_NUM_THREADS=1
 - MPI/Pthreads : NP de 1 à 8, OMP_NUM_THREADS=16
 - Sans et avec hwloc-bind -p pu:00-15

PKDGRAV3

MPI pur, de 16 à 128 de NP

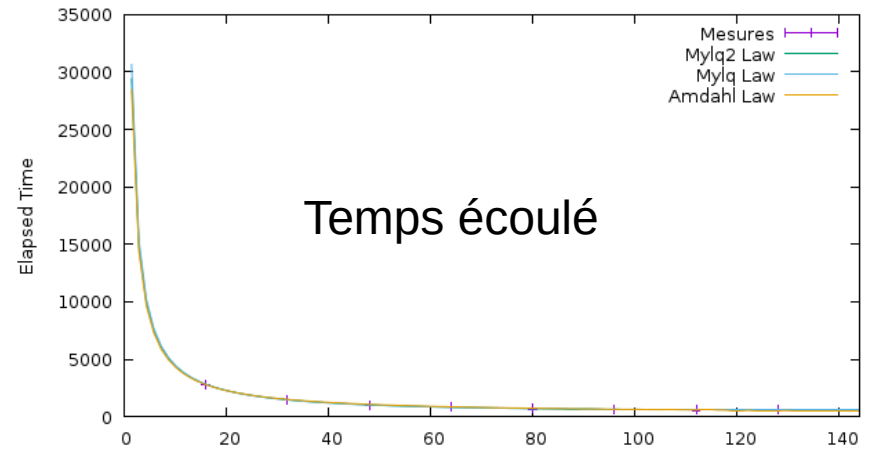
PKDGRAV3 C/MPI Cosmology Example



Temps écoulé

Pas de localisation

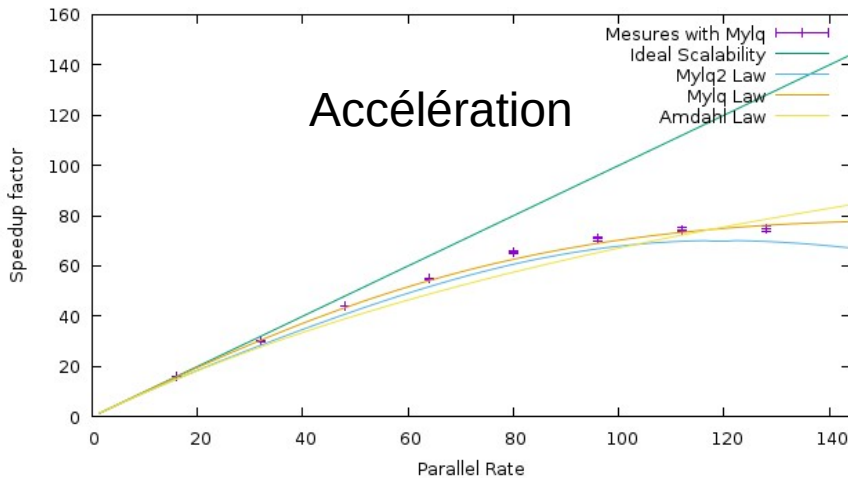
PKDGRAV3 C/MPI Cosmology Example



Temps écoulé

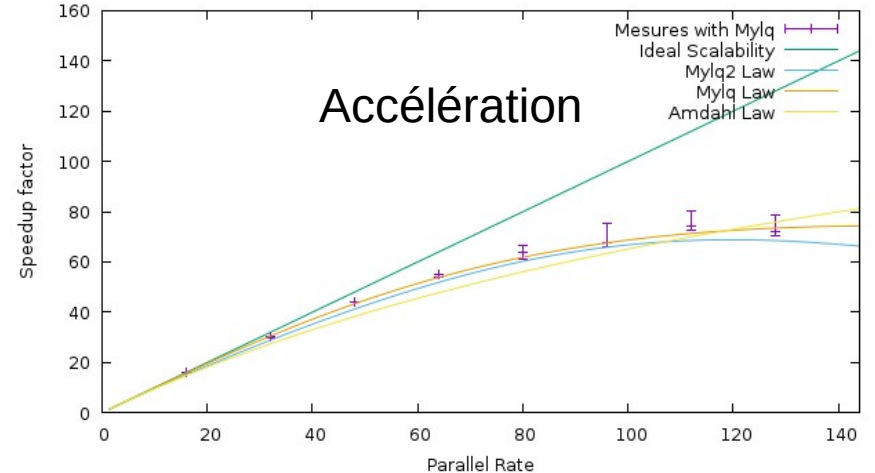
Localisation 16 premiers

PKDGRAV3 C/MPI Cosmology Example



Accélération

PKDGRAV3 C/MPI Cosmology Example



Accélération

PKDGRAV3

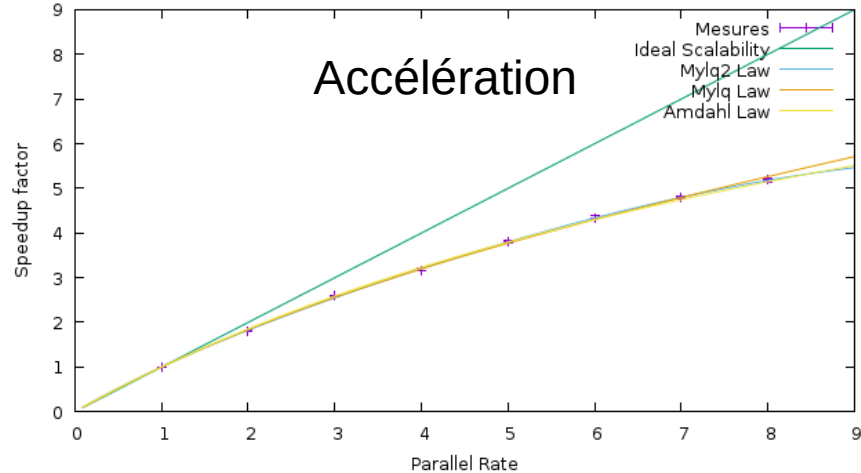
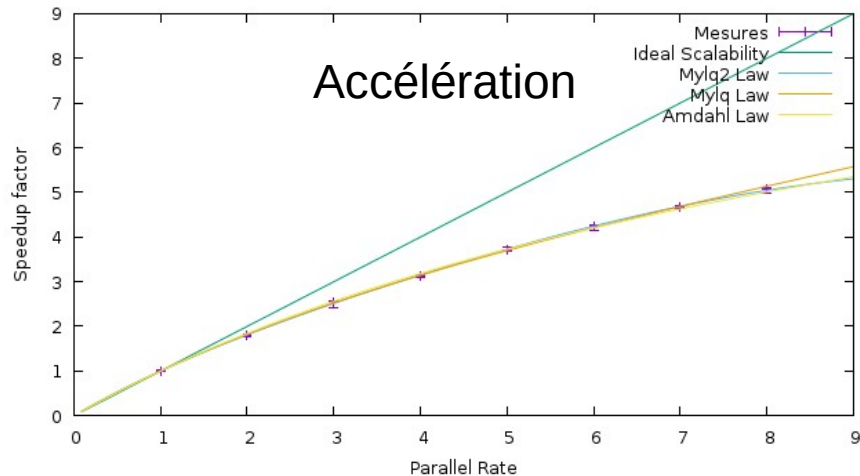
Hybrid MPI, de 1 à 8 de NP

PKDGRAV3 C/MPI Cosmology Example

PKDGRAV3 C/MPI Cosmology Example

Accélération

Accélération

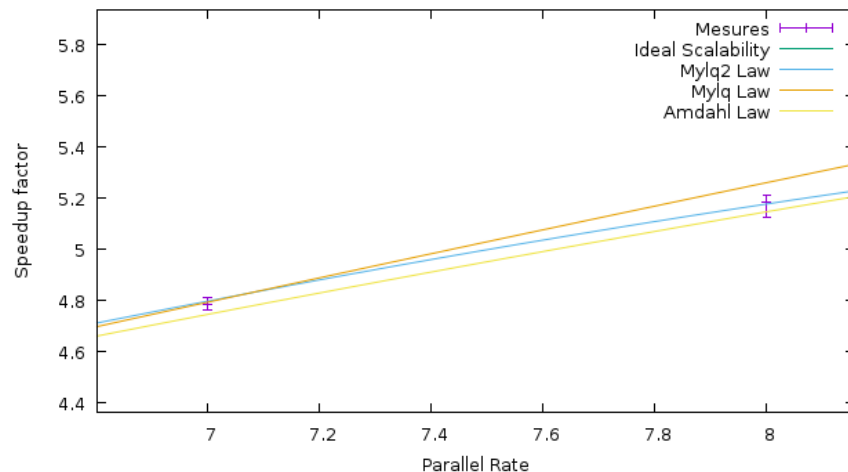
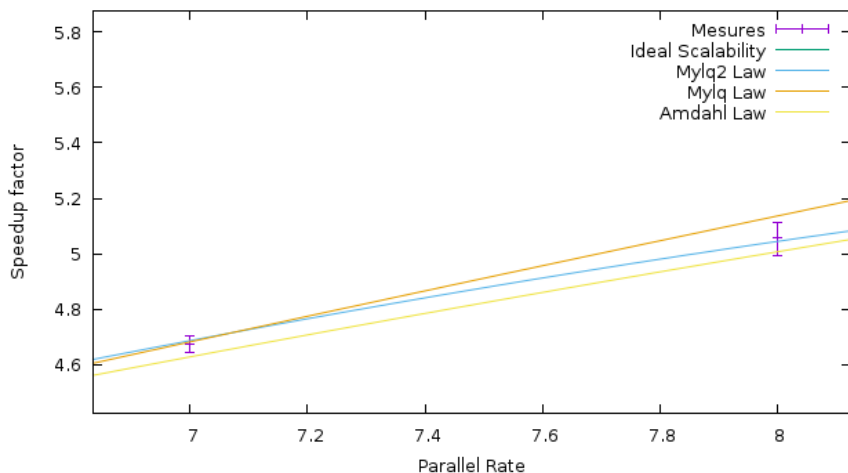


Pas de localisation

Localisation 16 premiers

PKDGRAV3 C/MPI Cosmology Example

PKDGRAV3 C/MPI Cosmology Example



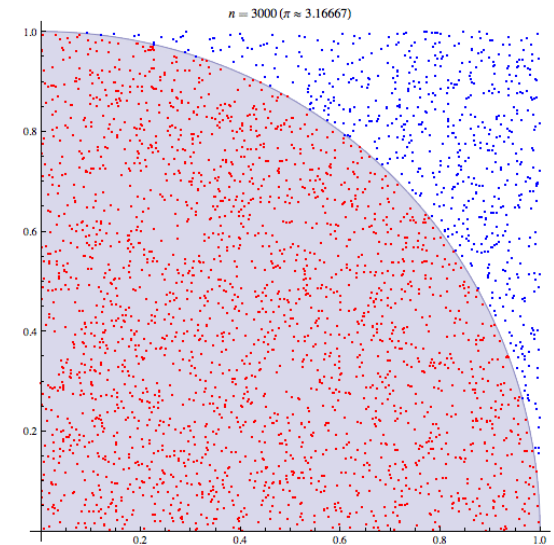
PKDGRAV3 : conclusions

- En MPI pur, moins de variabilité sur le délocalisé
- En hybride, moins de variabilité sur le localisé
- La loi de Mylq est bien pour du MPI pur
- La loi de Mylq2 est meilleur sur l'Hybride
- Moralité :
 - La loi de Mylq du premier ordre est efficace pour des codes MPI
 - La loi de Mylq du second ordre est meilleur pour les codes hybrides
 - L'influence de la « localisation » est à étudier soigneusement...

Et pour des implémentations simples ?

PiMC : Pi by Dart Board Method

- Exemple classique du calcul de Monte Carlo
- Implémentation parallèle : distribution
 - De 2 à 4 paramètres
 - Nombre total d'itérations
 - Régime de Parallélisme (PR)
 - (Type de variable : INT32, INT64, FP32, FP64)
 - (RNG : MWC, CONG, SHR3, KISS)
 - 2 observables simples :
 - Estimation de Pi estimation (just indicative, Pi n'est pas rationnel :-)
 - Temps écoulé




Pas très pertinent comme exemple ? Tutoriel sur le //isme du LLNL;-)

Introduction to GPU Parallel Programming
Data Heroes Summer HPC Workshop
June 27, 2016

Donald Frederick,
Livermore Computing

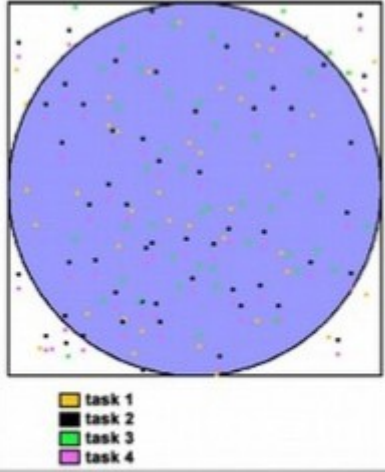
Lawrence Livermore
National Laboratory



LLNL-PRES-XXXXXX
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC02-07NA27344. Lawrence Livermore National Security, LLC

Approximation of Pi by Monte Carlo – Parallel Version

- Another problem that's easy to parallelize: All point calculations are independent; no data dependencies
- Work can be evenly divided; no load balance concerns
- No need for communication or synchronization between tasks
- Parallel strategy: Divide the loop into equal portions that can be executed by the pool of tasks
- Each task independently performs its work
- A SPMD model is used
- One task acts as the master to collect results and compute the value of Pi



task 1
task 2
task 3
task 4

Lawrence Livermore National Laboratory

PiMC juste pour la 6^e école des Houches

- Banc d'essai :

- Matériel : 64 R410 avec 8 cœurs en mode HT,
 - Interconnexion Infiniband QDR (40 Gb/s)
- OS : Debian Jessie SIDUS
- Logiciel : OpenMPI/C

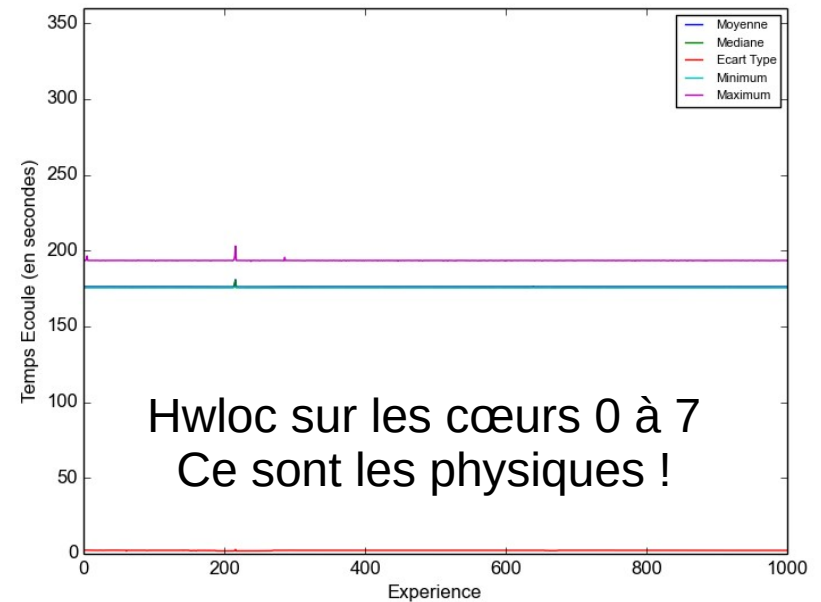
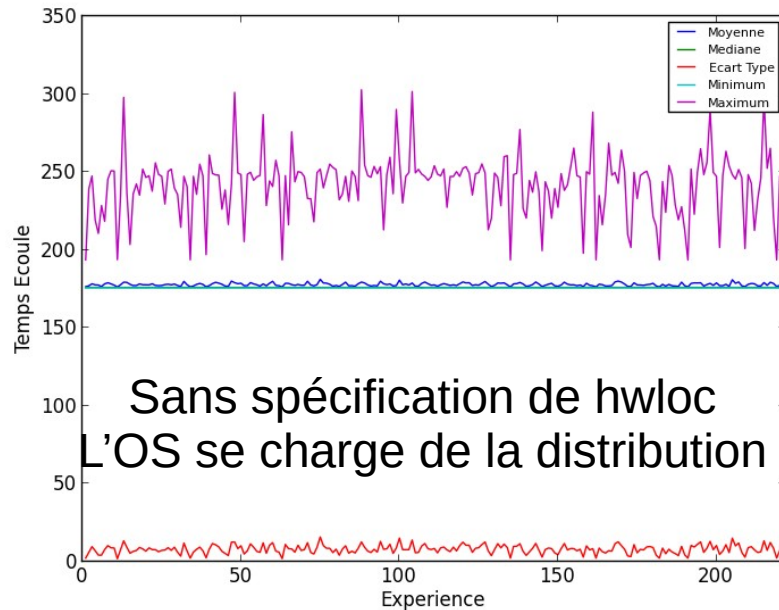
- Expériences :

- Communications réduites au minimum
- 1 Piterations : 10^{12} également distribuées
- Régime de parallélisme de 1 à 512 (distribution par saut)
- 40 lancements pour chaque régime
- La métrologie est assurée par le programme « time »
- `/usr/bin/time mpirun.openmpi -np $PR -mca btl self,openib,sm -hostfile $MyHostFile -loadbalance hwloc-bind -p pu:$AFF`
`/scratch/root/bench4gpu/Pi/C/MPI/Pi_MPI_FP32_MWC $ITERATIONS`



PiMC : pourquoi une « affinité » ?

- Durant la qualification d'un cluster de 48 nœuds
- Des centaines de lancements pour éprouver l'infra...
- Moralité : « localiser » les processus est pas mal...



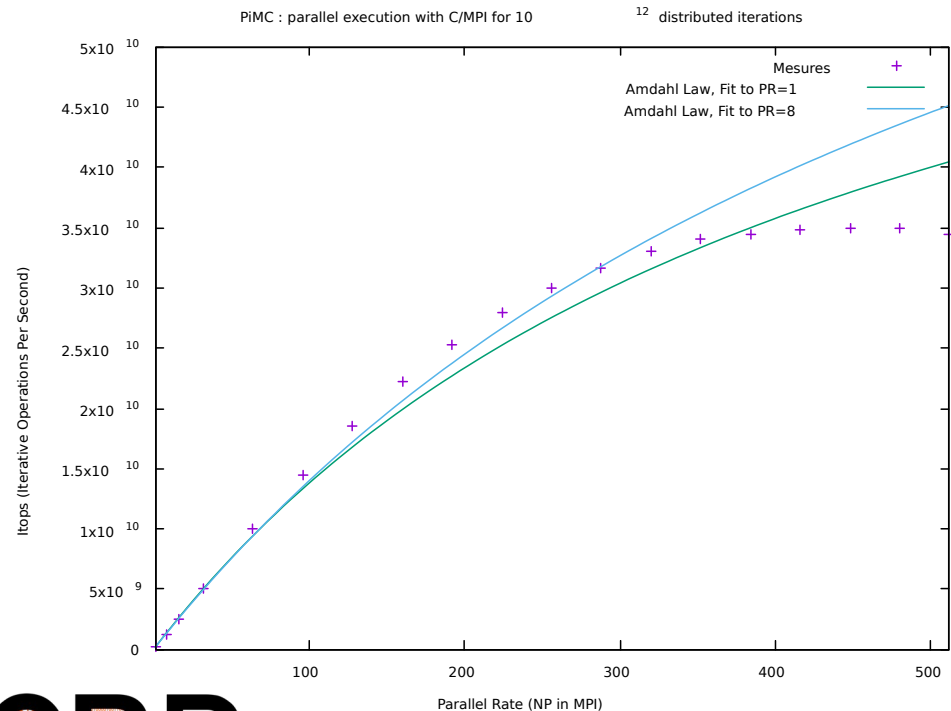
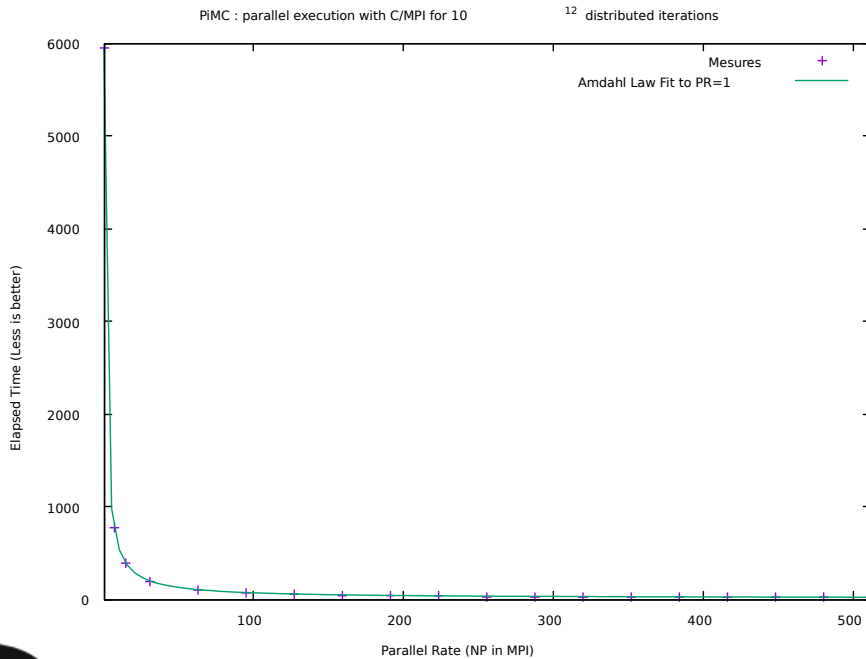
PiMC : et les résultats sont...

NP	Itops	Speedup 1	Speedup 8	Total Time	Variability %	Average	Median	Stdev	Minimum	Maximum
1	1.68E+08	1.00	1.05	5952	0.04	5953.22	5952.11	2.46	5950.32	5958.86
8	1.28E+09	7.62	8.00	6245	0.05	780.54	780.65	0.35	780.02	781.05
16	2.55E+09	15.21	15.96	6263	0.04	391.41	391.41	0.16	391.04	391.71
32	5.08E+09	30.22	31.71	6302	0.07	196.95	196.93	0.13	196.82	197.53
64	9.96E+09	59.30	62.22	6424	0.08	100.39	100.38	0.08	100.25	100.63
96	1.45E+10	86.31	90.56	6621	0.25	68.94	68.97	0.18	68.32	69.53
128	1.86E+10	110.86	116.32	6872	0.55	53.75	53.69	0.29	53.14	54.74
160	2.22E+10	132.12	138.63	7208	0.75	45.09	45.05	0.34	44.47	46.28
192	2.53E+10	150.53	157.95	7592	0.59	39.52	39.54	0.23	38.85	40.20
224	2.80E+10	166.38	174.57	8014	0.81	35.80	35.78	0.29	35.21	37.10
256	3.00E+10	178.80	187.60	8522	0.74	33.32	33.29	0.25	32.78	34.28
288	3.17E+10	188.54	197.82	9092	0.82	31.58	31.57	0.26	31.04	32.30
320	3.30E+10	196.28	205.94	9704	1.04	30.37	30.33	0.31	29.87	31.26
352	3.40E+10	202.14	212.10	10365	1.43	29.52	29.45	0.42	28.83	30.58
384	3.44E+10	204.86	214.94	11157	1.29	29.08	29.06	0.38	28.34	30.19
416	3.48E+10	207.14	217.34	11954	1.03	28.70	28.74	0.30	28.19	29.67
448	3.50E+10	208.08	218.32	12815	1.16	28.67	28.61	0.33	28.04	29.69
480	3.49E+10	207.97	218.21	13738	1.34	28.65	28.62	0.38	27.99	29.77
512	3.45E+10	205.10	215.20	14858	1.28	29.10	29.02	0.37	28.58	30.18

PiMC : graphiquement

Amdahl est-elle une bonne loi ?

- Temps écoulé en secondes
 - Ca me semble correct...
- Performance en Itops
 - Fit to 1 : $p=99.78\%$
 - Fit to 8 : $p=99.83\%$



Evolution de la loi d'Amdahl en Mylq

Intégration d'une influence linéaire

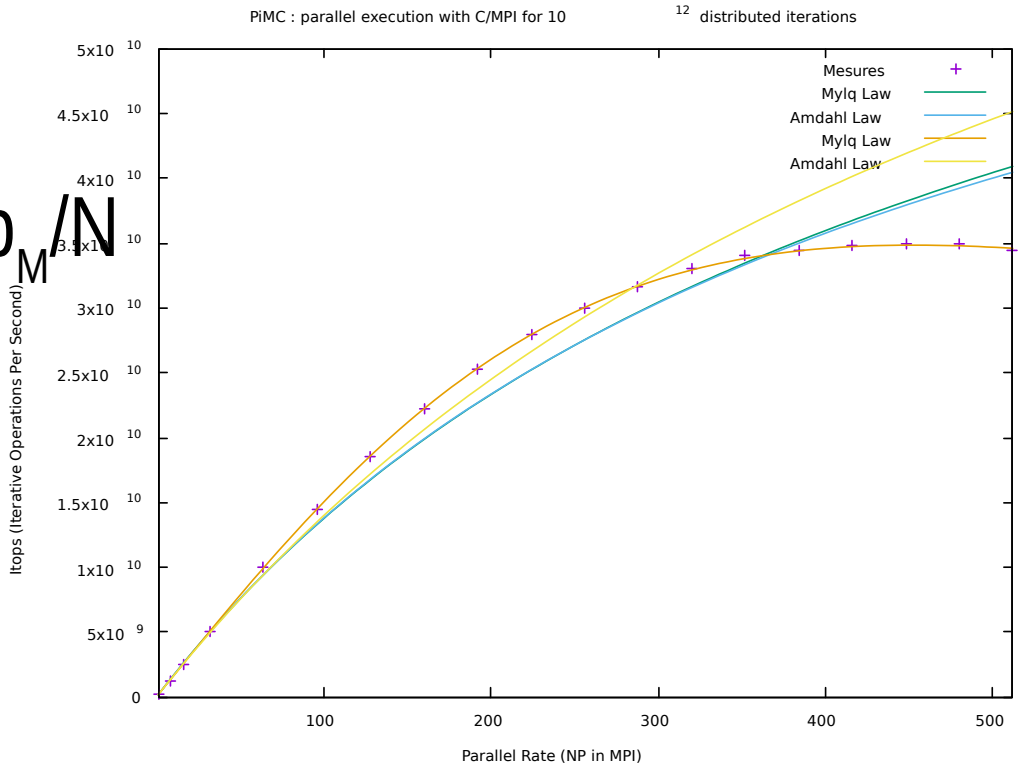
- Loi d'Amdahl : $T = s_A + p_A / N$

- Loi de Mylq : $T = s_M + c_M N + p_M / N$

- Signification de c_M :

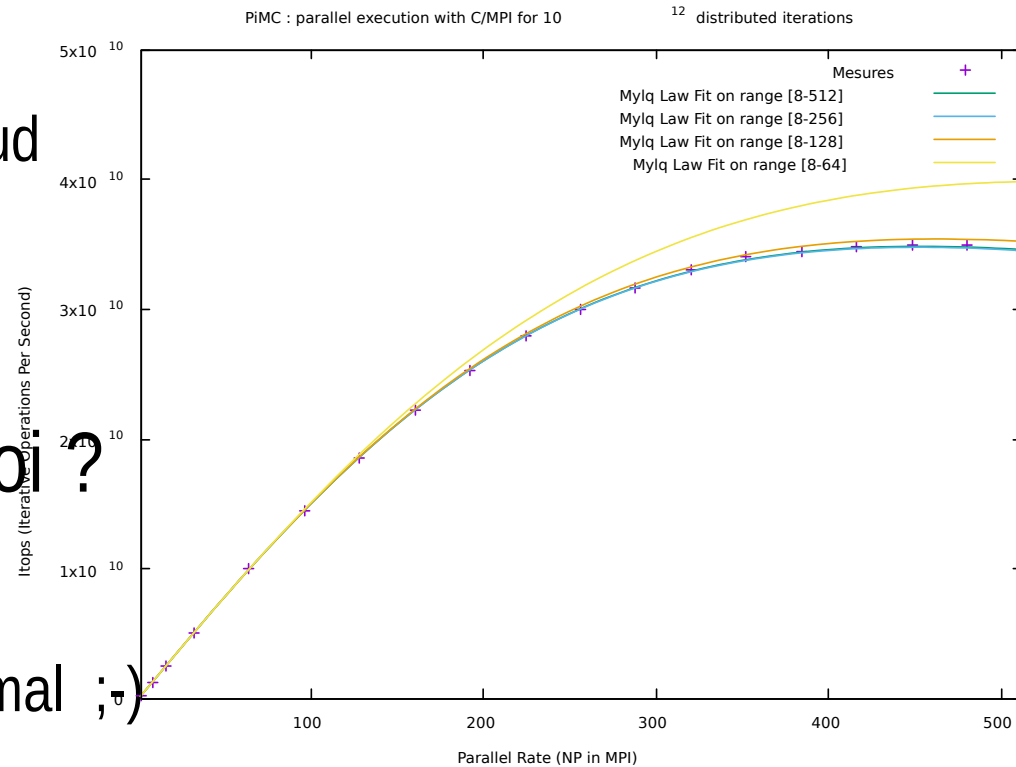
- Communications
- Processus d'initialisation
- and $c_M \sim 0.03$,

- Et p_M normalisé ~ 0.9998 si on exclut la valeur pour PR=1



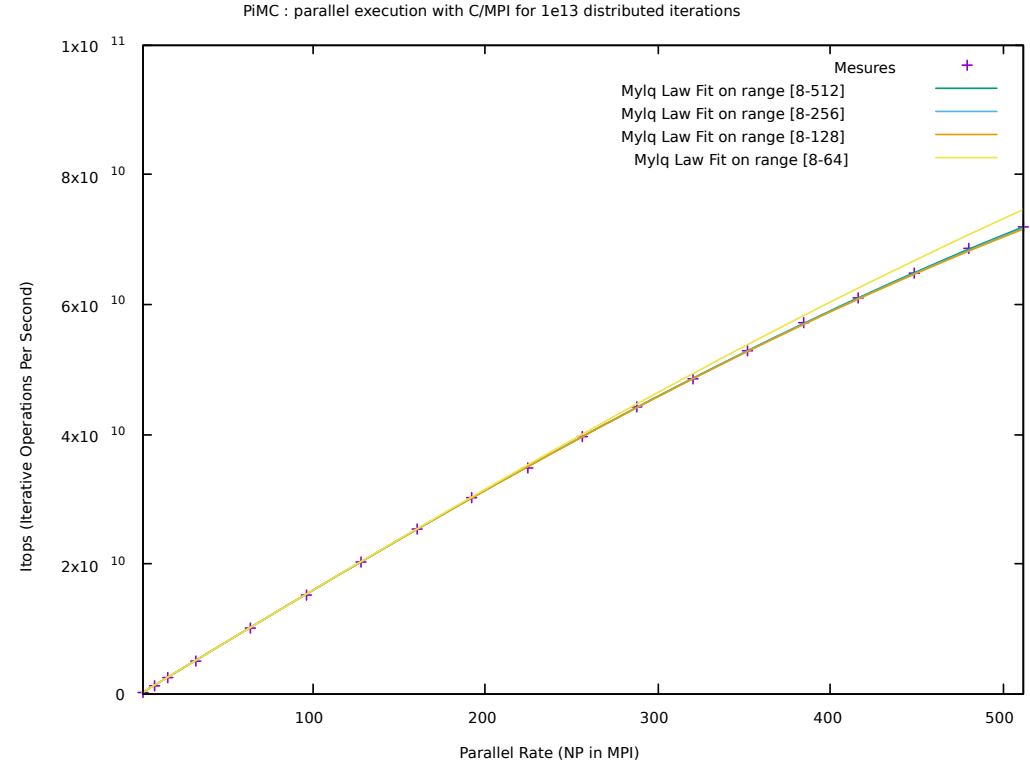
Loi de Mylq : Pourquoi exclure $PR=1$? Est-ce une loi plus prédictible ?

- Pourquoi exclure $PR=1$
 - Mécanismes internes au nœud
 - Effets de l'OS
 - Effets du Processeur : Turbo
- Plus prédictible comme loi ?
 - Essai de « fit » : 1/2, 1/4, 1/8
 - Pour 1/4, ça ne marche pas mal ;)
- Mais d'autres effets à intégrer...



Influence sur le temps écoulé Et si j'augmente le nb d'itérations ?

- De 10^{12} à 10^{13} itérations
- Accélération de 208 à 448
- Efficacité de 40 % à 87 %
- Itops de 34 à 72 Gitops
- Paramètres de Mylq :
 - p_M déduit de 0.99998
 - $c_M=0.032$ (le précédent 0.03)
- Moralité : ne soyez pas trop « léger » sur vos sets d'entrée...



Pourquoi ce qui est précédent n'est pas très honnête...

- Naguères, Mylq n'était pas si bon, pourquoi ?

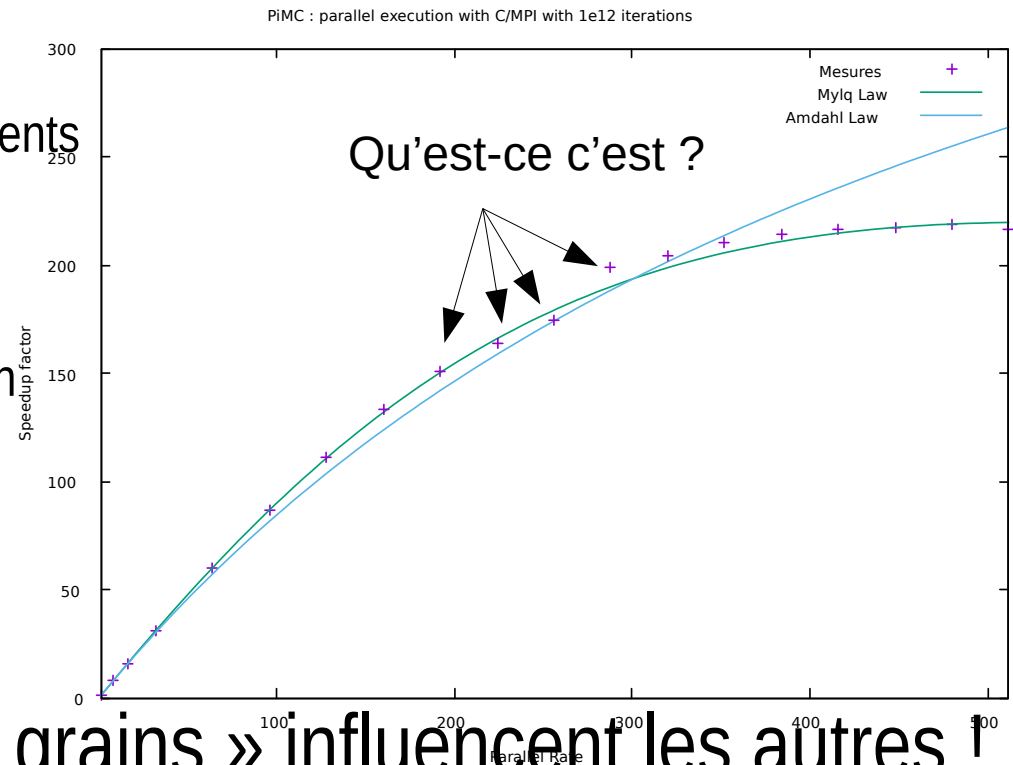
- Défaut de statistiques ?

- Pour chaque PR, 10 lancements
- Distribution sur 64 nœuds
 - Jobs concurrents
 - Exclusif dans l'exécution
- Variabilité autour 1 %

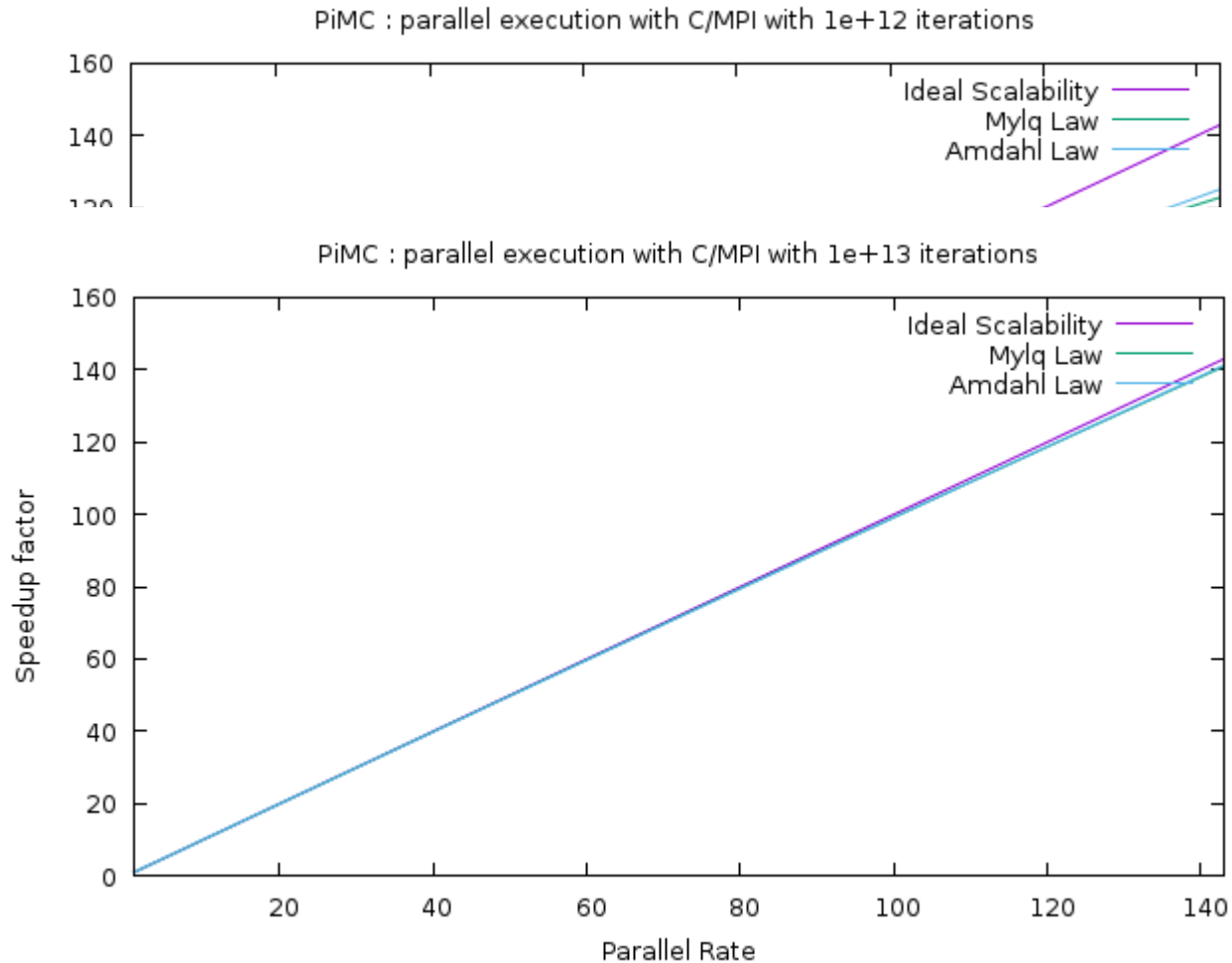
- Solution :

- Lancements exclusifs

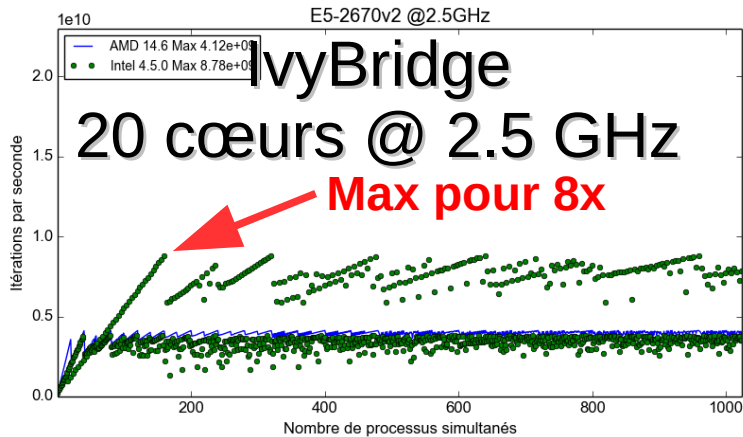
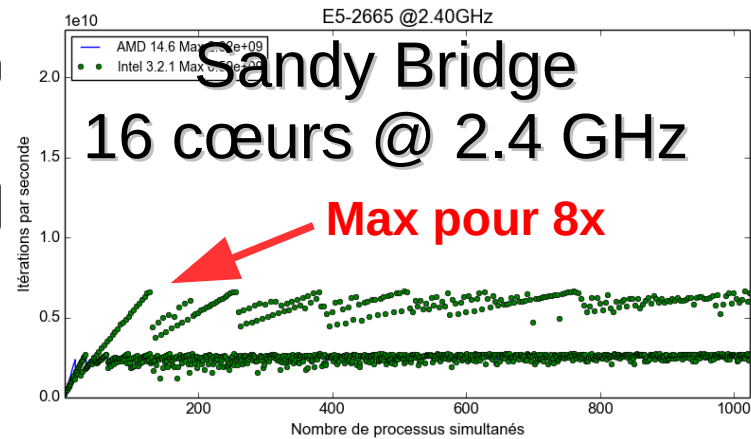
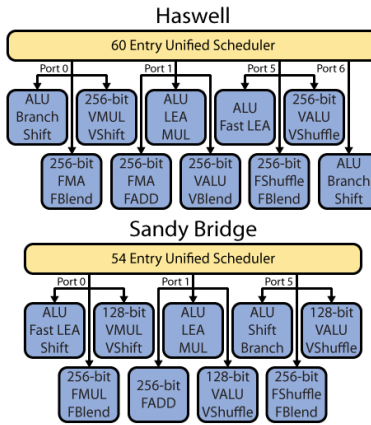
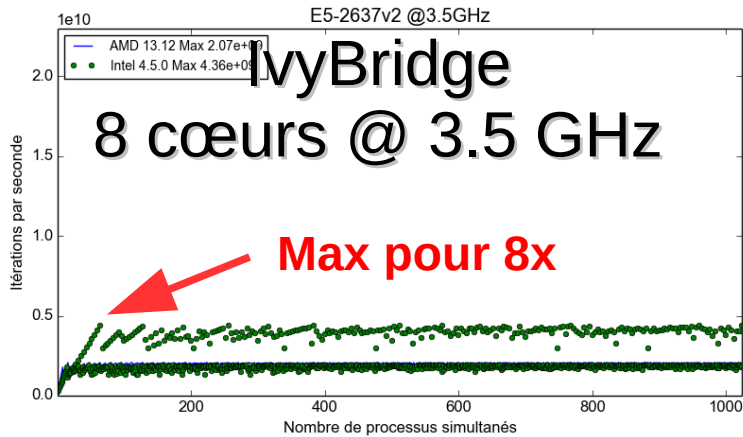
- Même les codes « gros grains » influencent les autres !



Et sur un cluster récent ? Les mêmes comportements ?

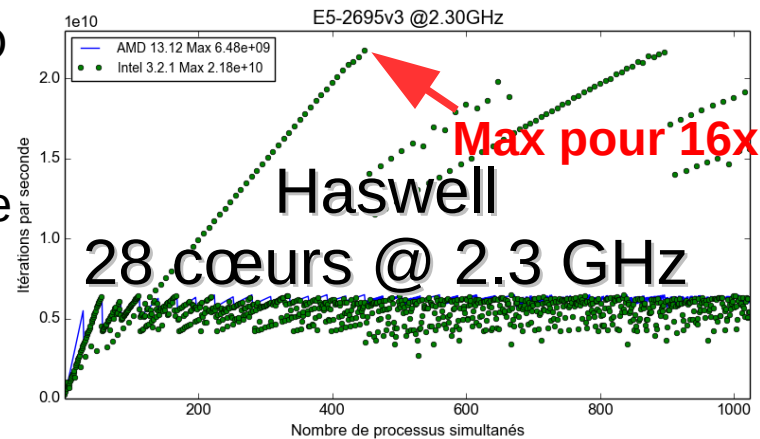


Et pour les PR >> Nombre de cœurs EPU depend de l'architecture !



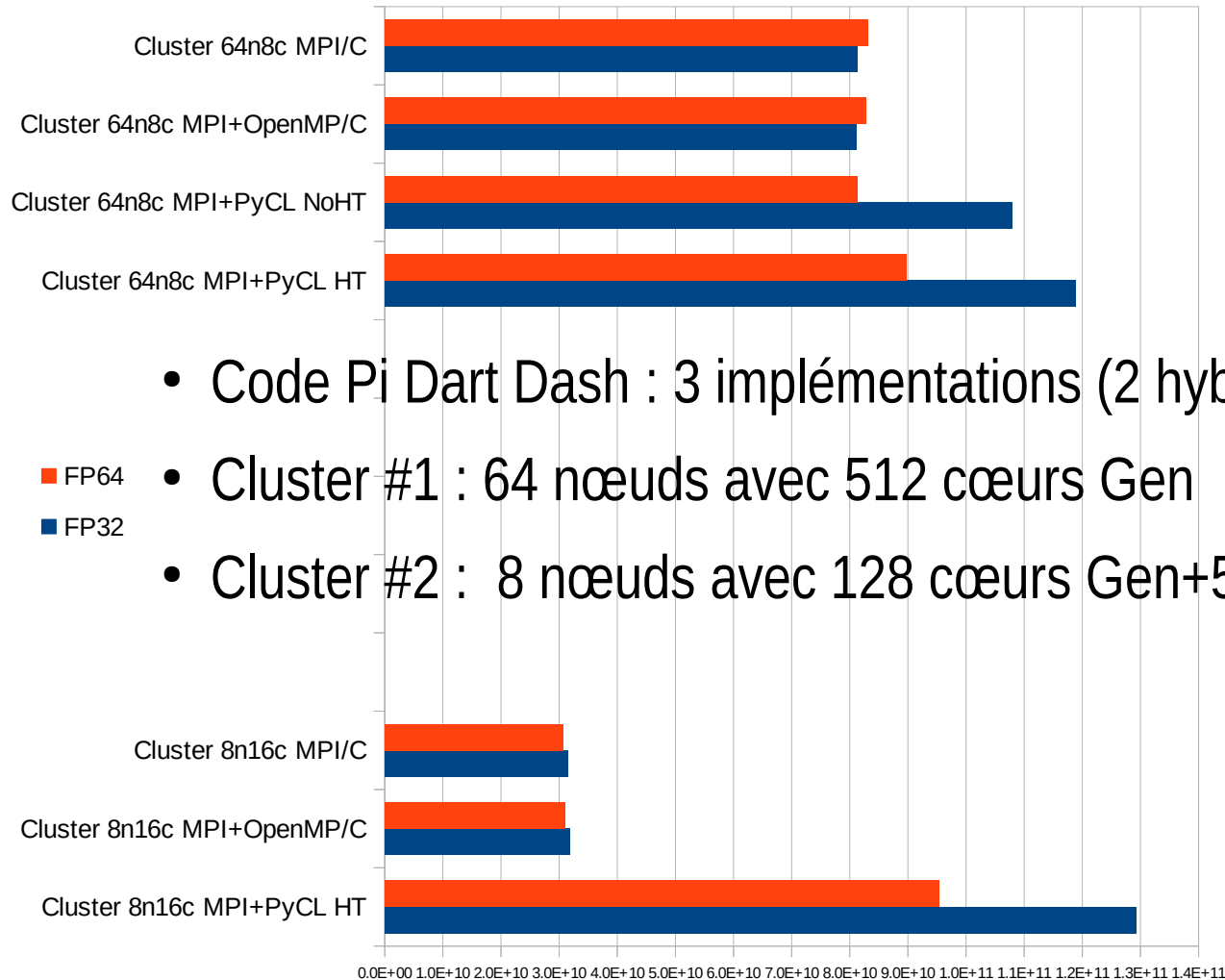
Intel x2,3 vs AMD
Period of 4
Max Perf :

- x8 Sandybridge
- x8 IvyBridge
- x16 Haswell



Sur les clusters ? Python efficace ?

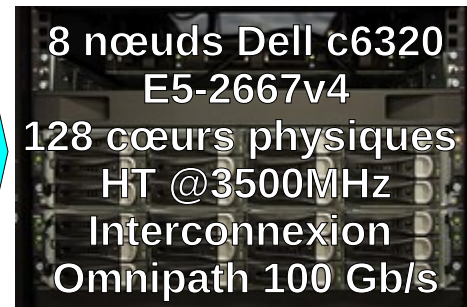
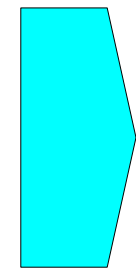
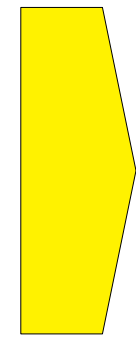
Petite comparaison avec GNU



- Code Pi Dart Dash : 3 implémentations (2 hybrides)

■ FP64
■ FP32

- Cluster #1 : 64 nœuds avec 512 cœurs Gen
- Cluster #2 : 8 nœuds avec 128 cœurs Gen+5



Introduction à la conclusion :

IT : le nouveau monde de la complexité

- Compliqué : de « cum plicare », « plier ensemble »
 - Descartes : « Le tout est la somme des parties. »
- Complexe : de « cum plexus », « tisser ensemble »
 - Immense nombre d'interaction, non linéarité, émergence, ...
- Les ressources de calcul sont des systèmes
 - Un Operating System porte au moins 200 processus en tâche de fond
 - Les cœurs CPU changent leur fréquence & voltage tout le temps, start/stop, ...
 - La DRAM change sa fréquence tout le temps
 - Les éléments réseau exploitent tous des mécanismes avec accès aléatoire

Modèle OSI & Loi d'Amdahl

Évolutions & perspectives

- Modèle OSI : La couche du dessous comme un service
 - ignorance de l'infrastructure socle : suicide l'étude de la scalabilité...
- Loi d'Amdahl : Ne dépend que de T_1 et p
 - Ce n'est jamais représentatif de ce qui s'exécute...
- Loi de Mylq : ajout d'un terme du premier order (et second)
 - Finalement assez représentatif des comportements observés, avec vigilance
 - Utilisable pour évaluer la scalabilité et prédire une performance
- Dans un nœud, aucune loi ne fonctionne...
 - Et c'est bien pire pour les GPU et les accélérateurs !