From multicores of CPUs to myrialus of GPUs:

Disruptive technologies Viewed by a physicist

Emmanuel Quémener

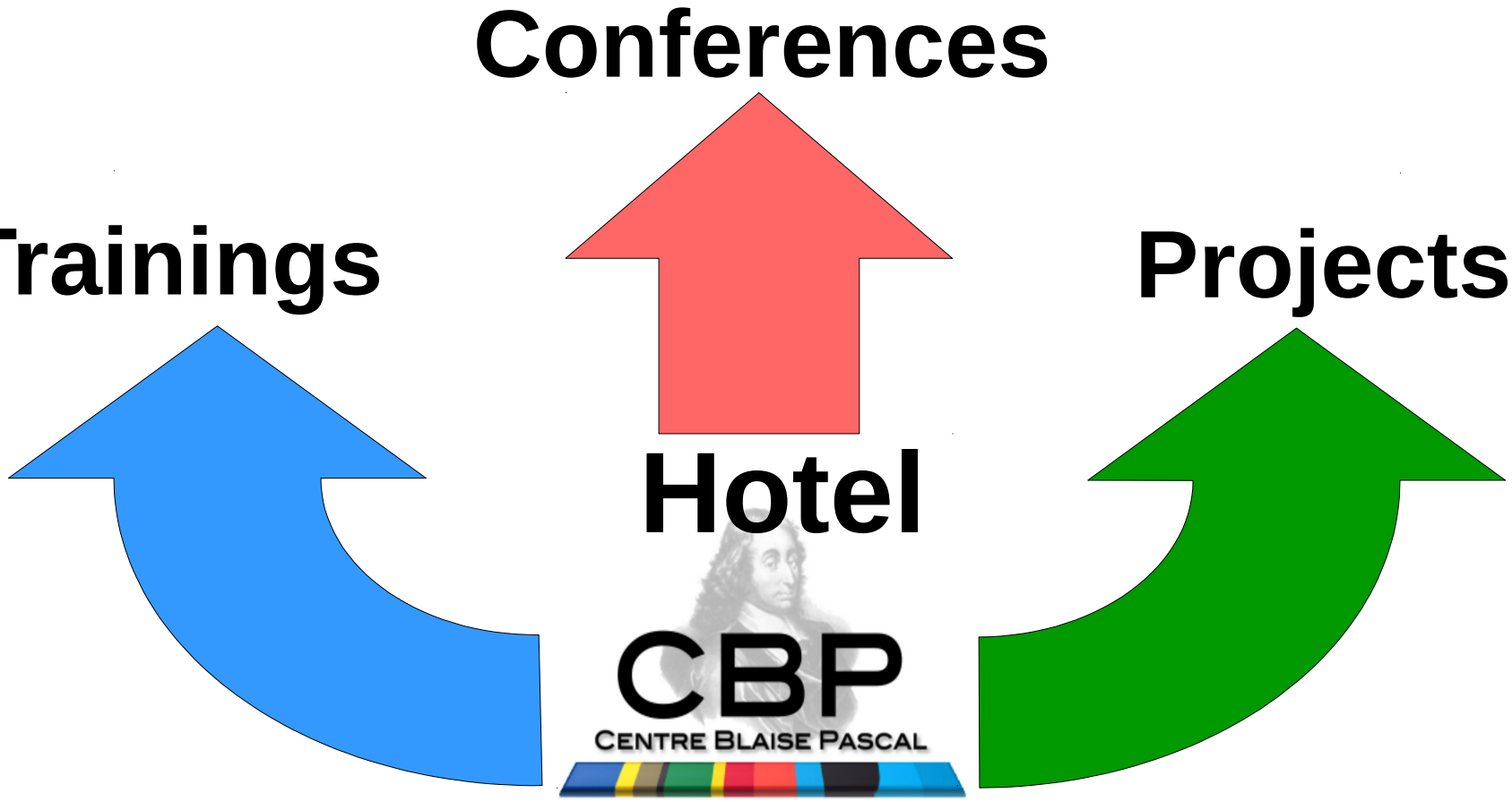
Disruptif : “qui sert à rompre”, de disrumpere (rompre)

Myriade : “nombre de dix mille”

Starting with a tiny joke ? (or not :-/ ?)

- How do you call people speak 3 languages ?
  - Trilingual people !
- How do you call people speak 2 languages ?
  - Bilingual people !
- How do you call people speak 1 language ?
  - French people !
- I'm french :
  - if I twist your eardrums, I apologize...

Blaise Pascal Center
« House of Modelisation » & ...



CBP also a **Test Center**

Experimental platform with 10 technical benchs

- **Multi-nodes**: 5 clusters from 4 to 64 nodes, Nodes/Cores : 4/24, 8/64, 8/64, 64/512
-
  -
-
  -
- **Integration** : 24 virtual machines : Debian from Lenny to Sid both 32 & 64 bits, ...
- **Exotic hardware** : 3 machines ARMv7 under Debian Jessie or Ubuntu
- **3D Vizualisation**: 2 stations , 2 videoprojectors, 20 monitors, 4 glasses
- **3D Virtual Desktop** : up to 30 hosts with x2go/VirtualGL
- **COMOD** with SIDUS : « Compute On My Own Device » for laboratories users
  - SIDUS is « Single Instance Distributing Universal System »
- **Galaxy prototype** for intensive processing of biological data

Centre Blaise Pascal ~ Dryden FR
Small illustrative example

- Nasa X-29
- Cell from F-5
- Engine from F-18
- Gears from F-16
- Studies
  - Flap wings
  - Incidence >50°
  - « Fly-By-Wire »

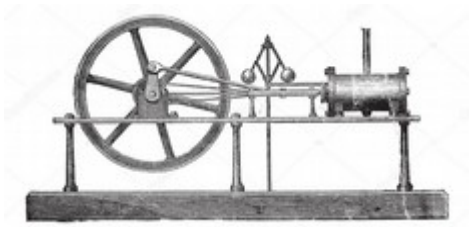
To recycle, to reuse and explore from new domains

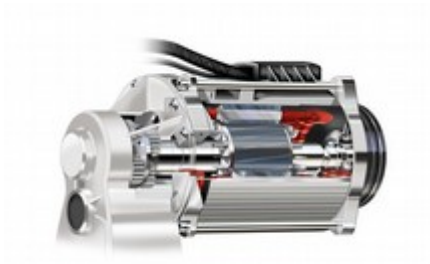
What view of the context ? From a physicist point of view...

- Physicist's « system » approach
  - Inheritance of analog computers
  - The computer system:
    - network / hardware / OS / software / codes / uses / ...
- Approach « Saint Thomas »
  - Apprehension by my sole measure
- Test pilot approach
  - Characterization, search for optimal exploitation ...

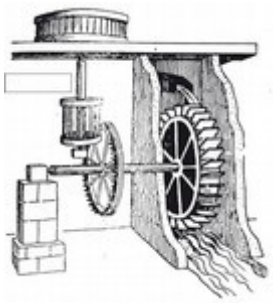
Energy : Engines & Humanity
APU from beasts to silicone…

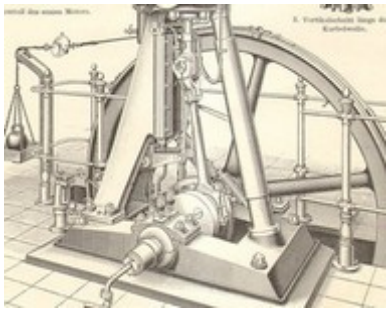
APU : Auxiliary Power Unit

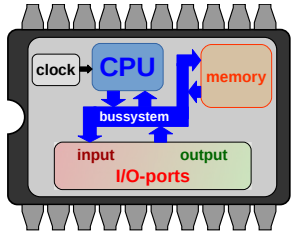




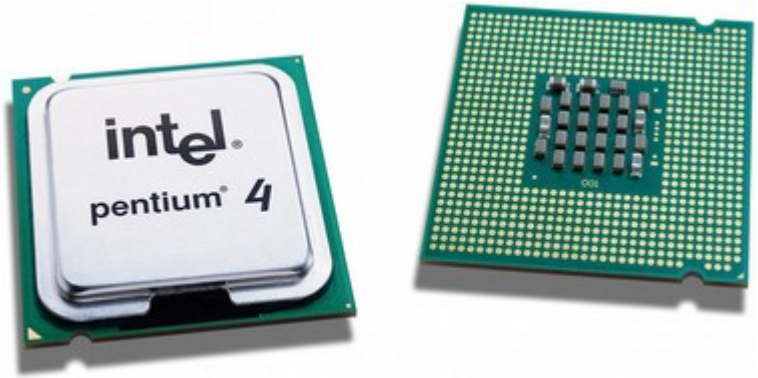
From ancient time to the present day...





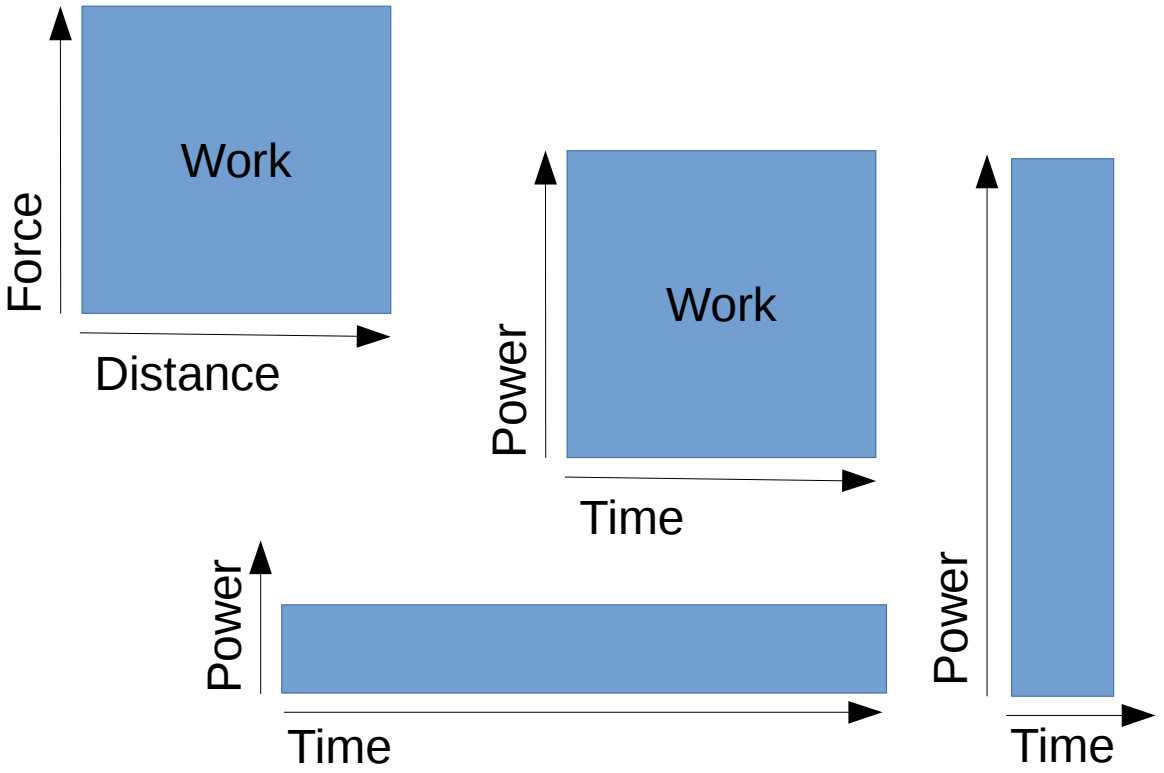


Energy in computers...
Reduced view of a physicist...



Electronic & Thermal

Energy in scientific computing... Extended view of a physicist



Mechanics

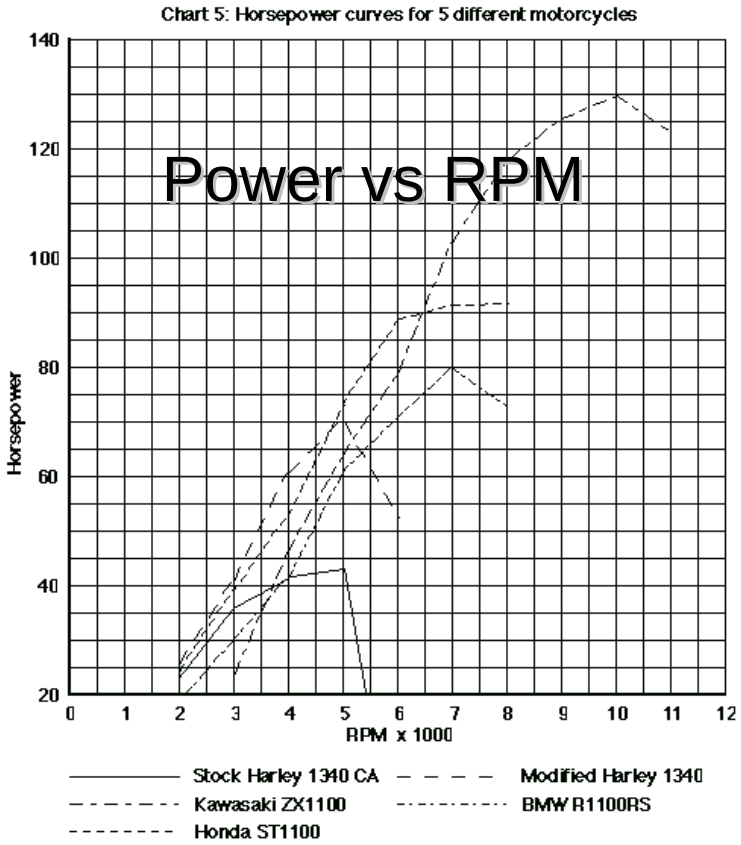
The power is defined as the product of

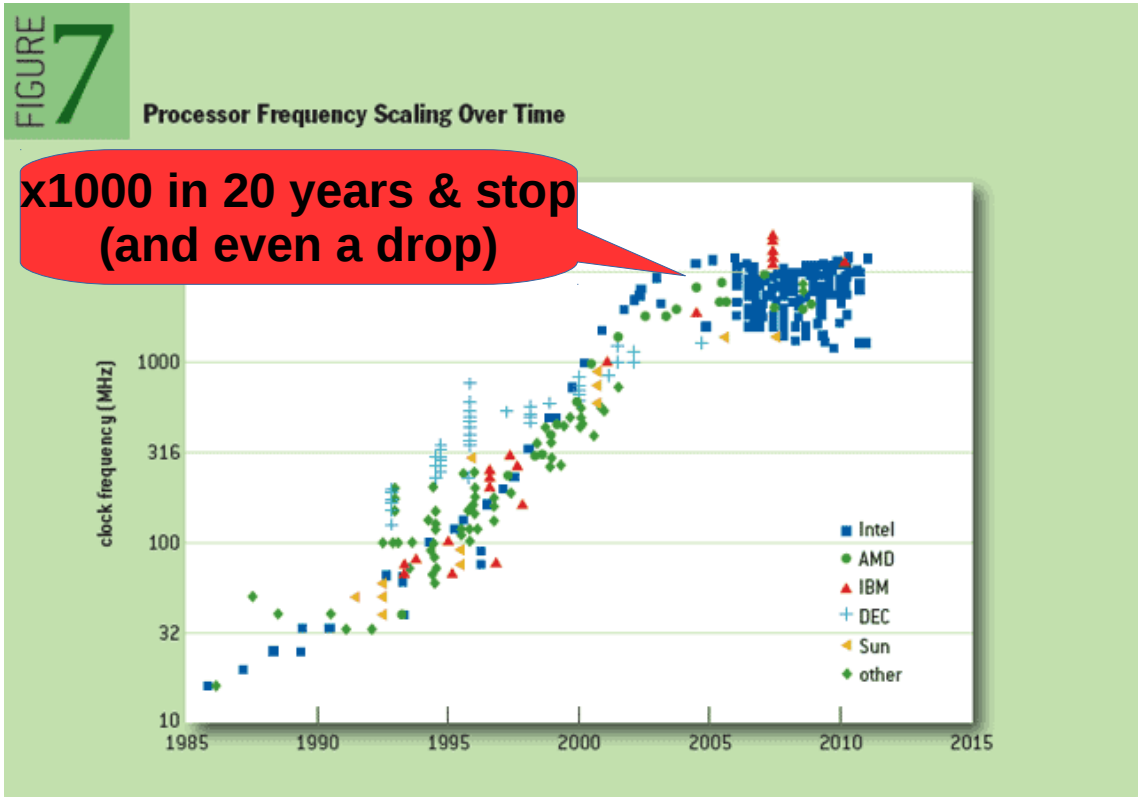
- Frequency
- Number of units
- Unit power

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

Characterize the power of computer engines

A long long time ago, between 1985... and 2005, variable is frequency !

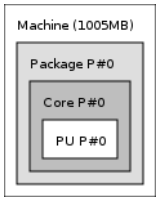


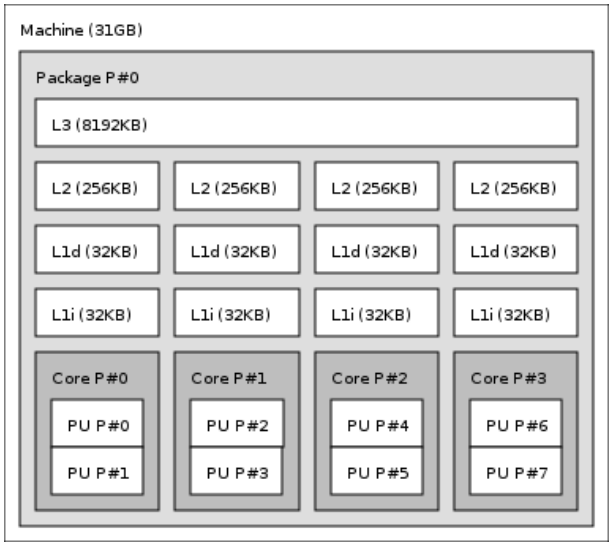


Why the multiplication of PU ? Bypass the « wall » of TDP…

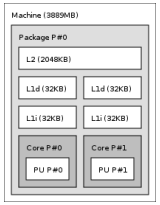
- **Rise & and Fall of frequency**
  - **Between 1989 and 1999: from 4 MHz to 400 MHz x100 in 10 years**
  - **Between 1999 and 2004: from 400 MHz to 4 GHz x ~ 10 in 5 years**
  - **Between 2004 and 2009: from 4 GHz to 2 GHz**
- **TDP: Thermal Design Power: socket limit power**
  - **TDP = ½ C V^2 f**
  - **C = Capacitance, f = frequency, V = voltage, but V = function (frequency)**
  - **TDP (or thermal envelope) for a socket: 150 W (on 4 cm^2)**
  - **Capacitance = Finesse ^{2}. Nb Transistors. Mylq Constant (~ 0.015)**
- **Viable solution: increase number of processing units (PU)**

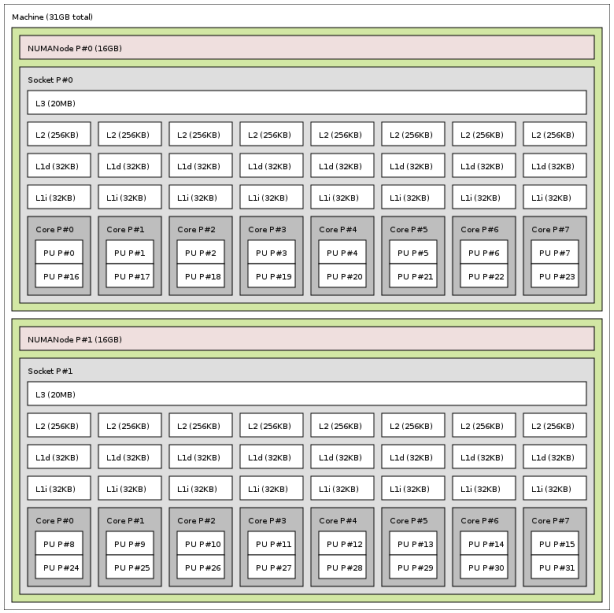
Evolution of machines…
From 2004 to 2013, on laptops !



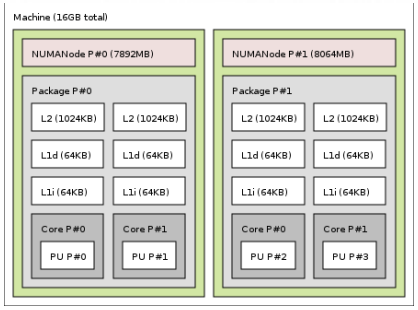


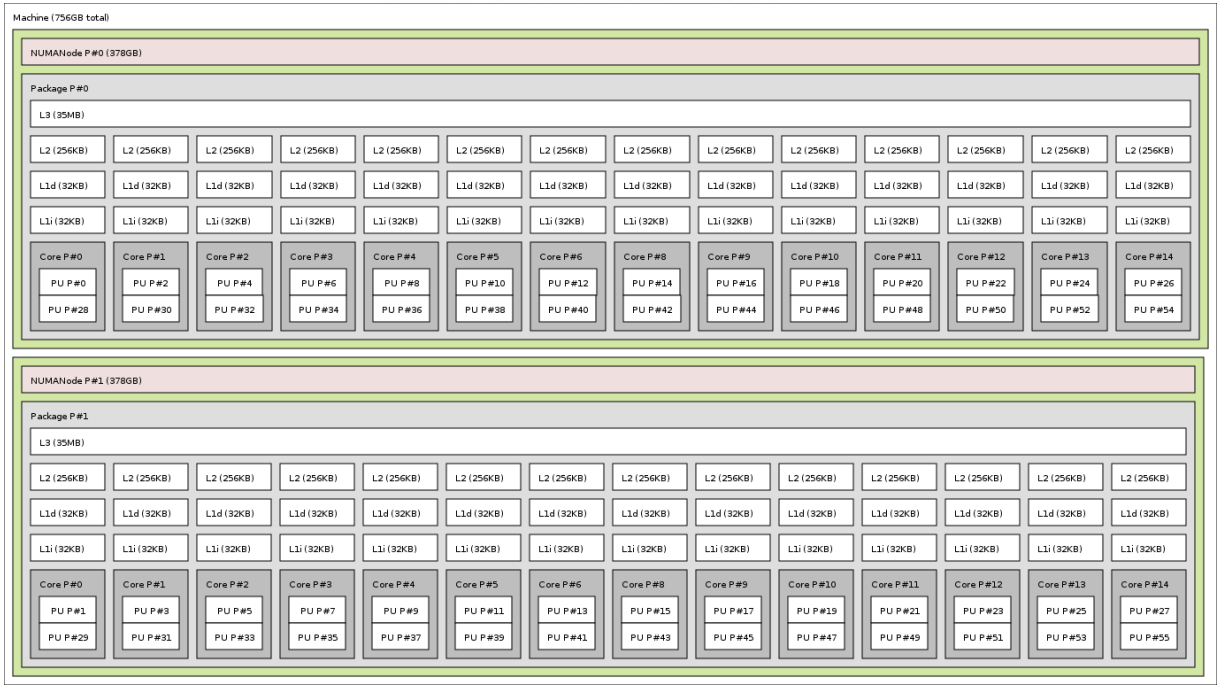
Evolution of machines…
From 2007 to 2016, on workstations



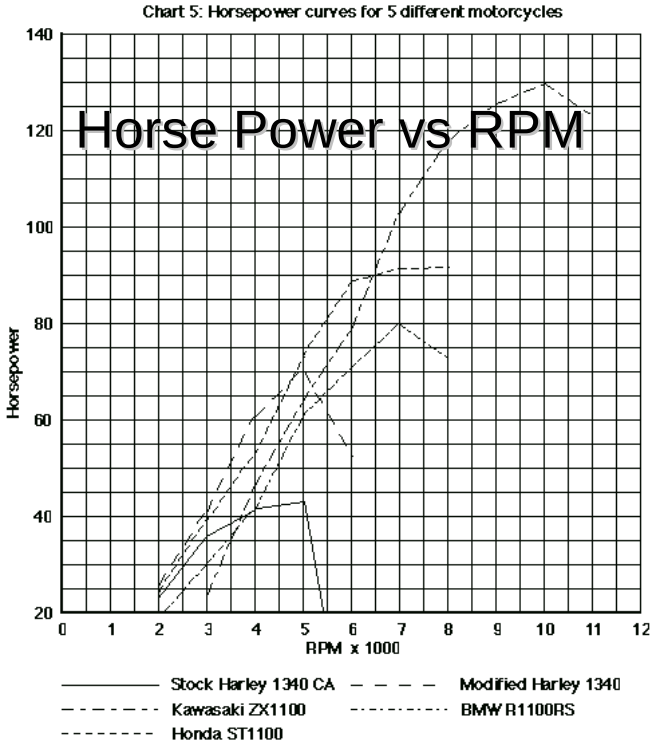


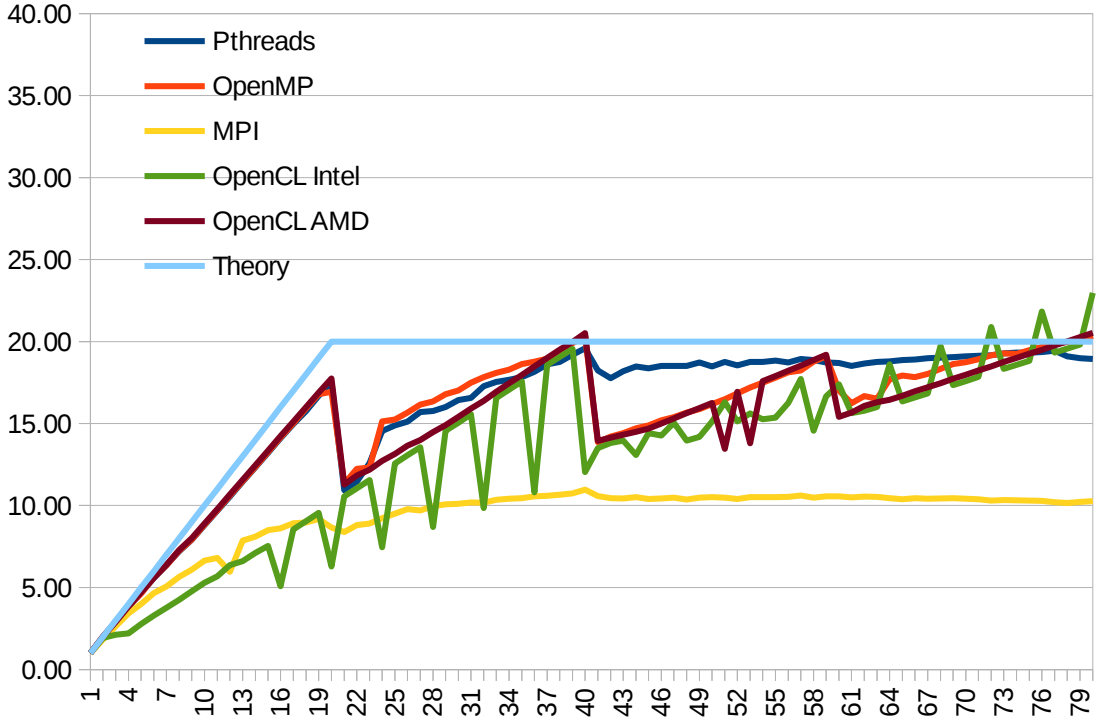
Evolution of machines…
From 2007 to 2016, on servers...





The "engine", source of "power"
Forget frequency, change RPM to PR

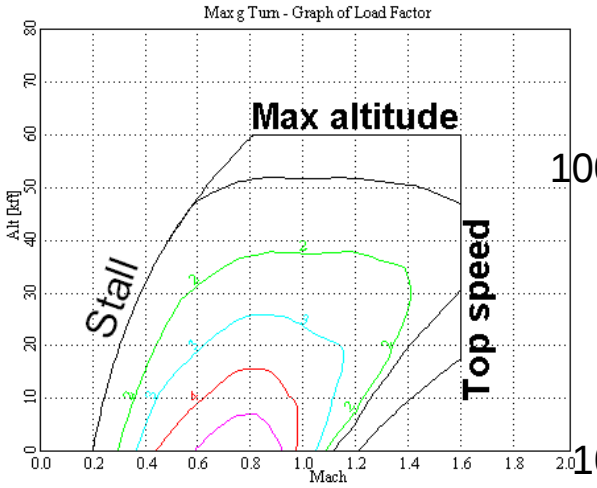




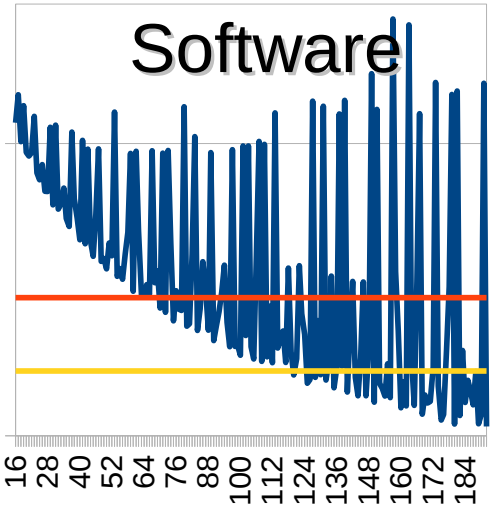
- What "behaviors" for these engines at "load-bearing"?
- The "engine", a system entangling hardware, OS and software

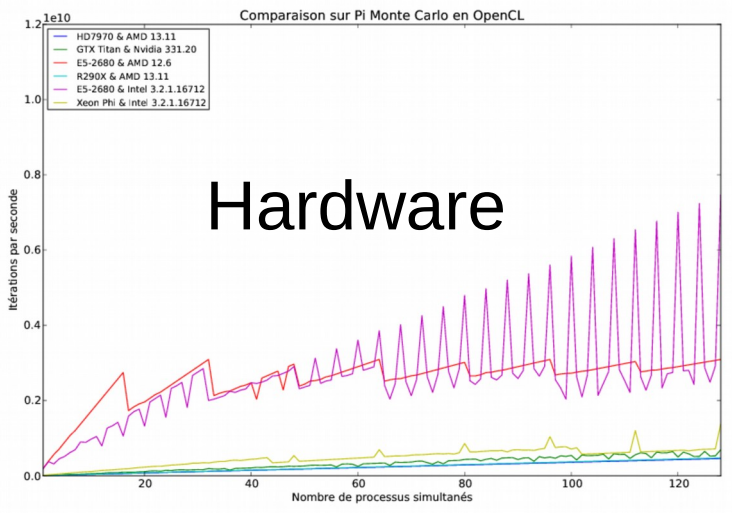
From engines to parallel envelope
A necessary exploration?

Flight Envelope



Parallel Envelope



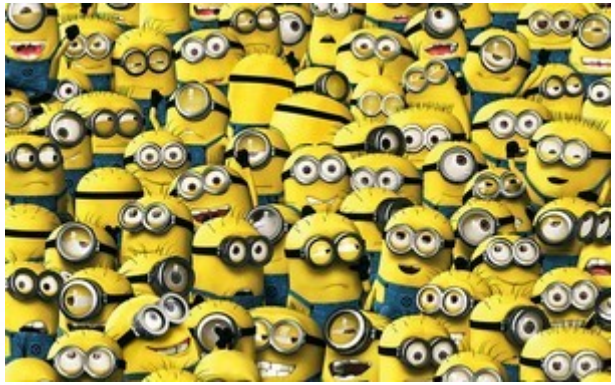


- Speed/altitude
- Load factor

- Parallelization/Memory/CPU/GPU
- Input/Output/Contexts/...

From (multi|many)-cores to myri-alus ?
A parallel laboratory on a chip !

- You got from 1 to dozen of thousands of equivalent PU !
- You got the choice between : **CPU MIC GPU**

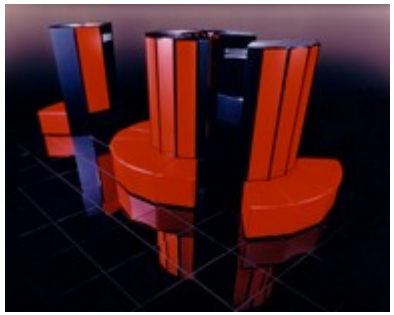


How to choose and distinguish them for its use?

A (human) generation...
A B-series movie film in 1984

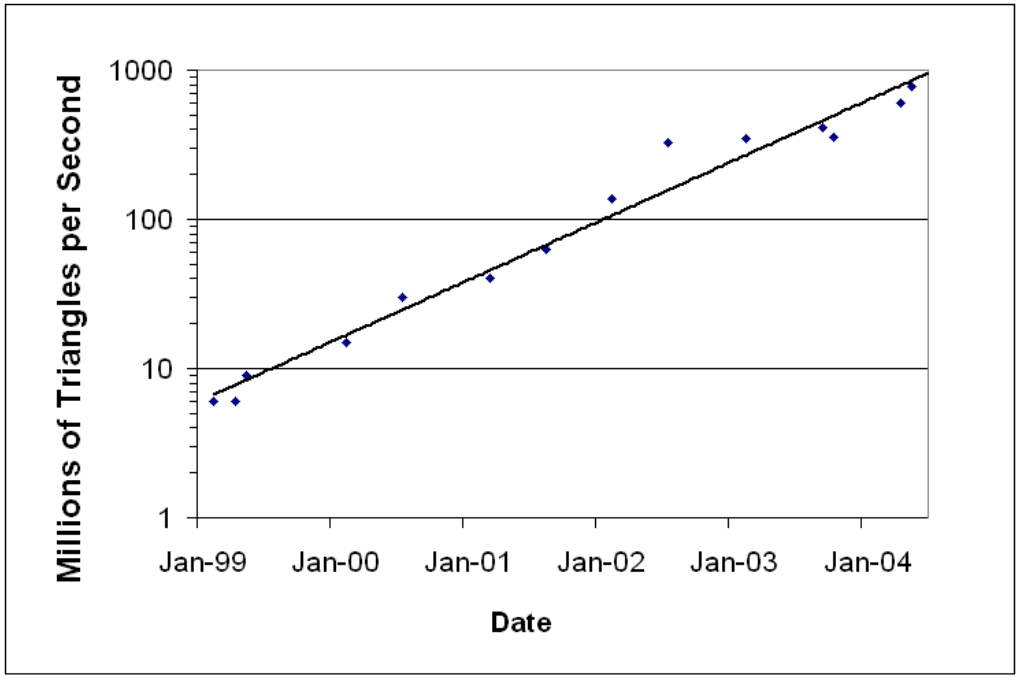
- 1984 : The Last Starfighter
  - 27 minutes of CGI
  - 22.5 10^9 operations per image
  - One Cray X-MP (130 kW)
  - 66 days (in fact, 1 year needed)

- 2017: sur la GTX 1080Ti (250 W)
  - 3.4 seconds
  - Comparaison GTX1080Ti / Cray
    - Performance : x1600000 !
    - Consumption /900000000 !



Why the GPU is « disruptif » ? The « big diversion » in HPC

- In a tiny conference at ENS-Cachan 2006Q1, this...
  - x100 in 5 years

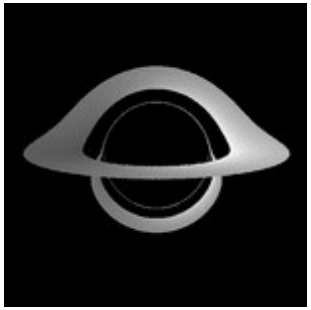


- Between 2000 and 2015
  - GeForce 2 Go/GTX 980Ti : from 286 to 2816000 MOperations/s : x10000
  - For a classical CPU : x100

In the beginning on all things

- Nvidia launchs the « personal supercomputer »
- When : 2008-11-19 Who : Nvidia
- Where : on Tom's Hardware
- What : a PCIe card Tesla C1060 PCIe with 240 cores
- How much : 933 Gflops Float32 (but 78 Gflops Float64)

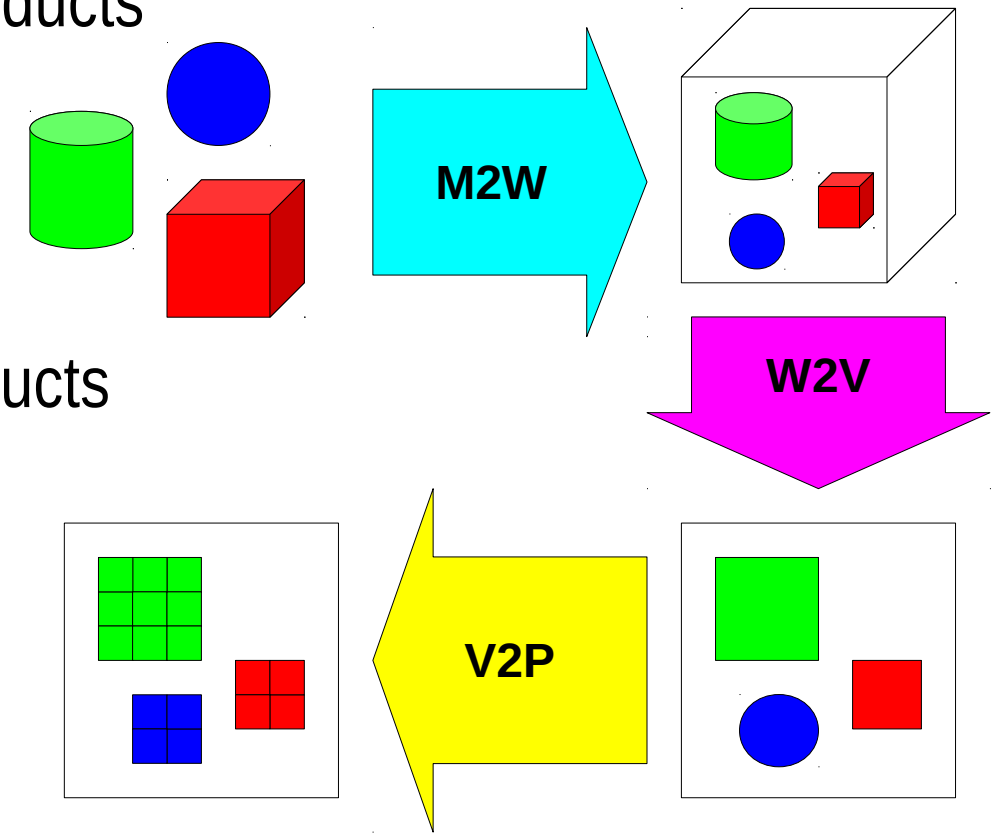
Why is the GPU so powerful? To build a 3D image!



- 2 approaches:
  - *Raytracing* : PovRay (« dimensions » from UMPA)
  - *Shadering* : 3 operations
- « Raytracing » :
  - From the eye to the contacts of each object
- « Shadering »
  - Model2World: Vector objects placed in the scene
  - World2View: Projects objects in a vector view plane
  - View2Projection: pixelation of the view plane

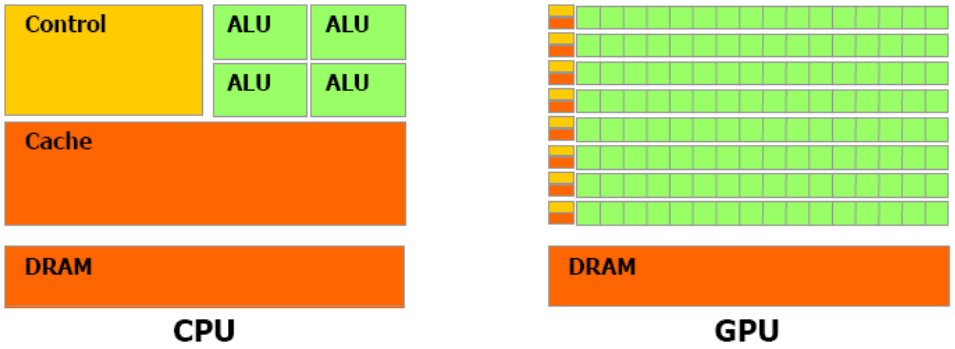
Why is the GPU so powerful? Shadering & Matrix Calculation

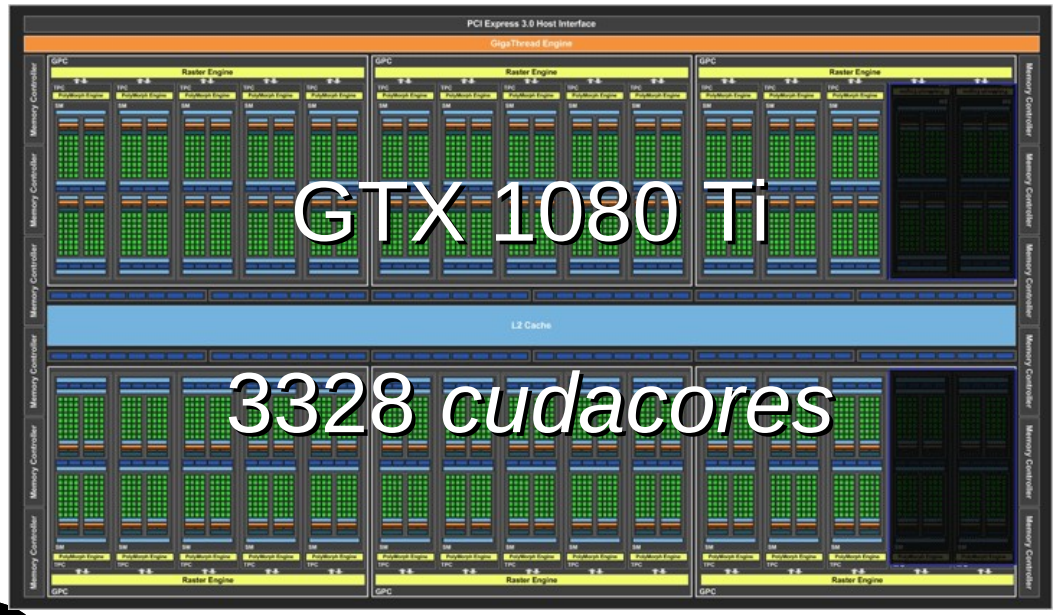
- Model 2 World: 3 matrix products
  - Rotation
  - Translation
  - Scalinge
- World 2 View: 2 matrix products
  - Positioning the camera
  - Location Direction
- View 2 Projection
  - Pixelisation



So a GPU: it's a "big" Matrix Multiplier

How many components inside? What difference between GPU/CPU



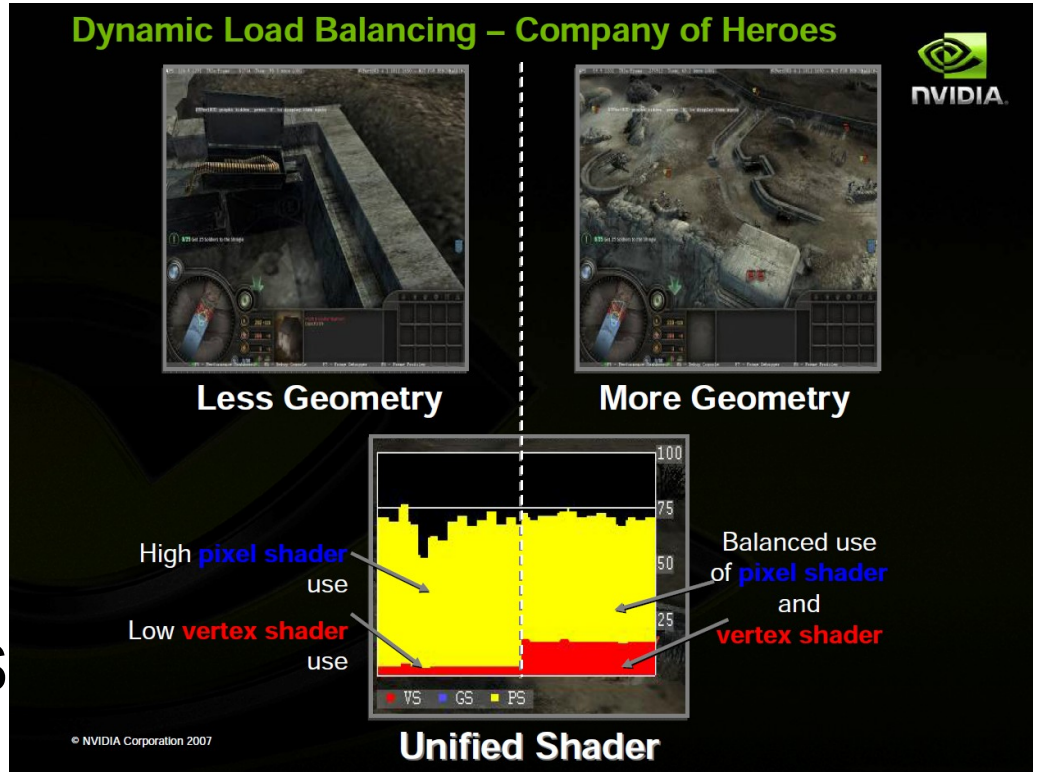


- Operations
  - Matrix multiplication
  - vectorization
  - "Pipelining"
  - Shader (multi) processor
- Schedule: 1993
  - OpenGL, Glide, Direct3D,
- Genericity: 2002
  - CgToolkit, CUDA, OpenCL

Why is the GPU so powerful? The return to a "generic" basis

In the GPU

- Specialized units
- Efficiency of pipelining
- Loss of adaptivity
- Moving Scenes
- Nature of different details



The solution

Return to more generic processing units!

Little Overview about parallelism

Parallel Programming Model

	Cluster	Node CPU	Node GPU	Node Nvidia	Accelerator	FPGA
MPI	Yes	Yes	Non	No	Yes*	No
PVM	Yes	Yes	No	No	Yes*	No
OpenMP	No	Yes	No	No	Yes*	No
Pthreads	No	Yes	No	No	Yes*	No
OpenCL	No	Yes	Yes	Yes	Yes	Yes
CUDA	No	No	No	Yes	No	No
TBB	No	Yes	No	No	Yes*	No

- OpenCL « seems » to be the more versatile :-)
- CUDA can ONLY be used with Nvidia board
- Accelerators seem to more agnostic to program...

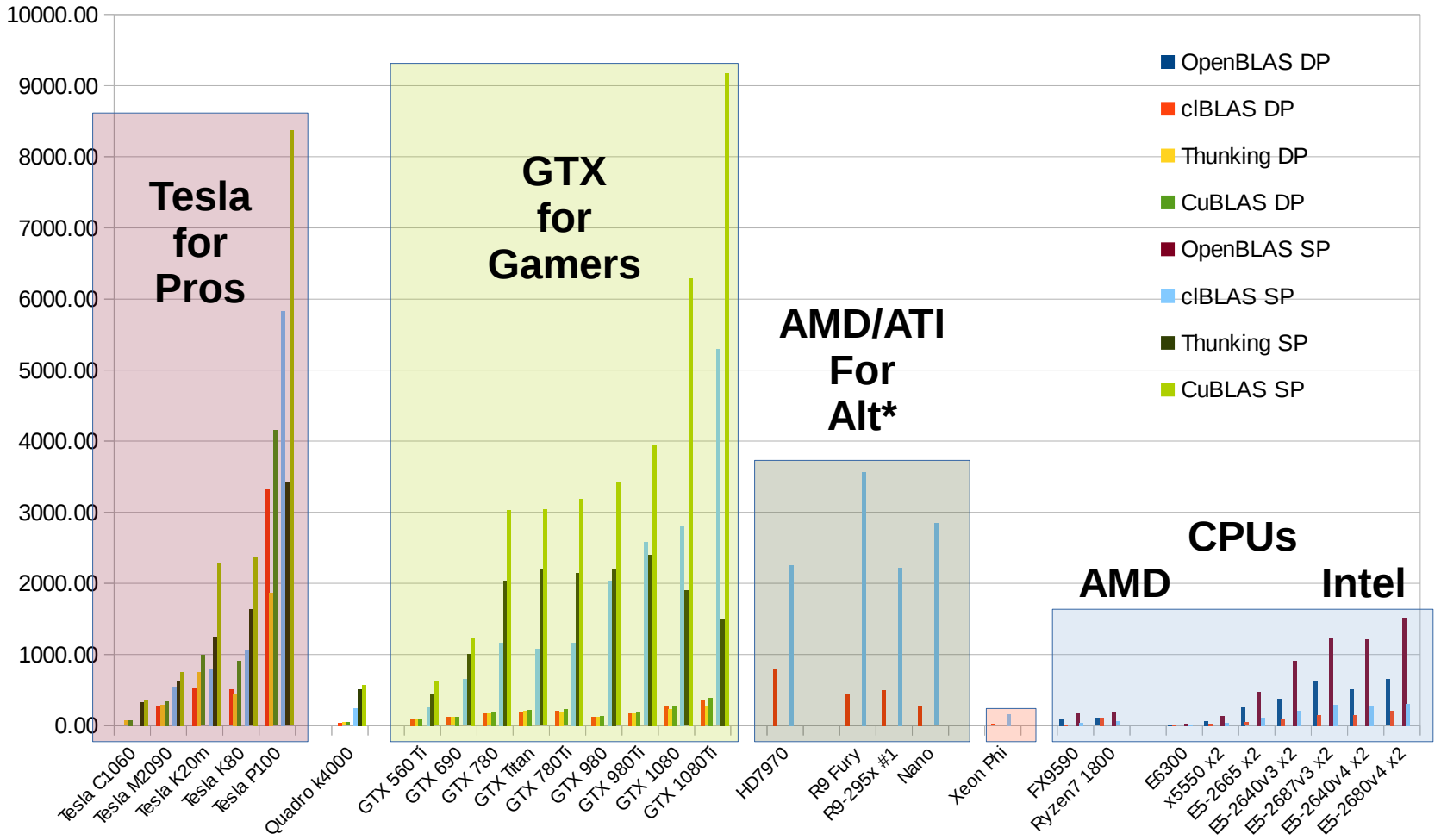
Act as « integrator » : No to reinvent the wheel...

Librairies of parallel programming

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

- Classical librairies are usable with GPUs
  - Nvidia provides lots of implementations (BLAS, FFT, ...)
  - OpenCL provides some, but not complete at all !

BLAS comparaison : CPU vs GPU xGEMM when x=D(P) or S(P)

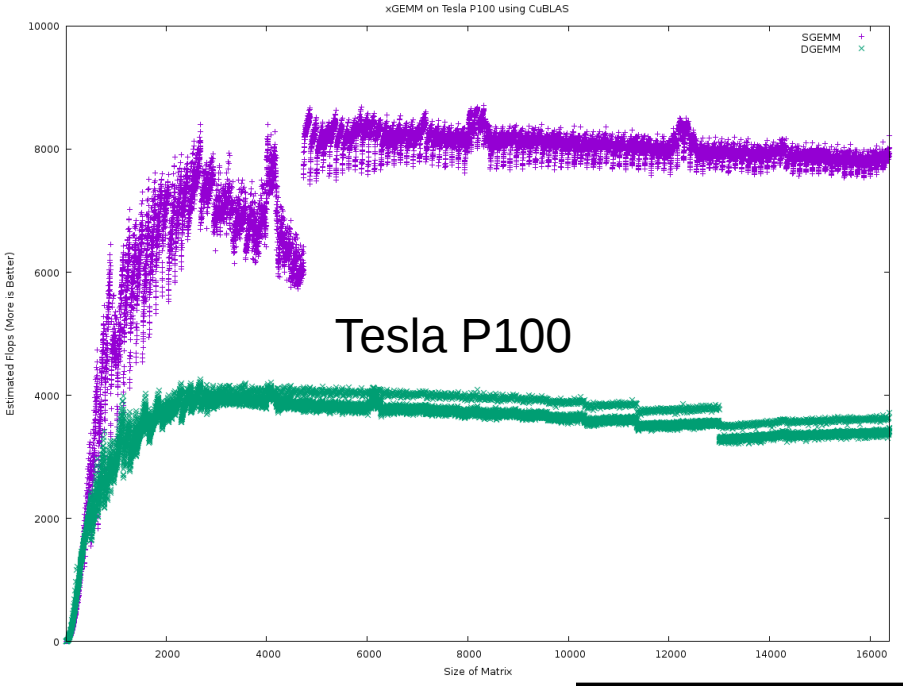


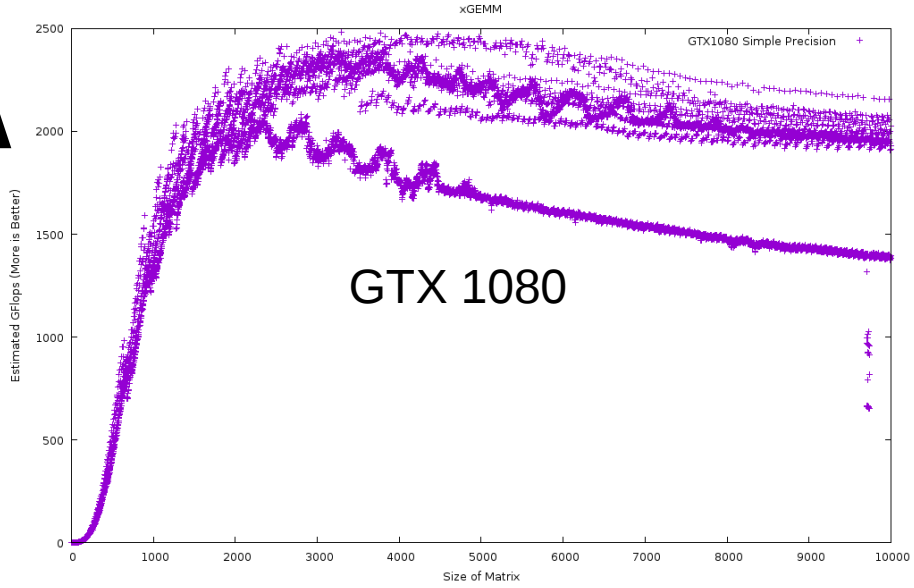
Already in 2010, strange ... The Tesla C1060 ...

- Only matrix product: xGEMM
  - Property: Transposed (A * B) = Transposed (A) * Transposed (B)
- Results (in Gflops): Yess!
  - SP: FBLAS / CBLAS: 12, CuBLAS: 350/327: x27 !!!
  - DP: FBLAS / CBLAS: 6, CuBLAS: 73/70: x11!
- Surprise: Mmmm! CuBLAS prefers the x16!
  - SP: 16000^2, 350, but 15999^2 or 16001^2, 97: x3.6!
  - DP: 10000^2, 73, but 9999^2 or 10001^2, 31: x2.35

What Nvidia never broadcasts... xGEMM on a large domain ...

Performance





Size

Yes, performance is there, but in specific conditions !

Why OpenCL ?
A non « politically correct history »

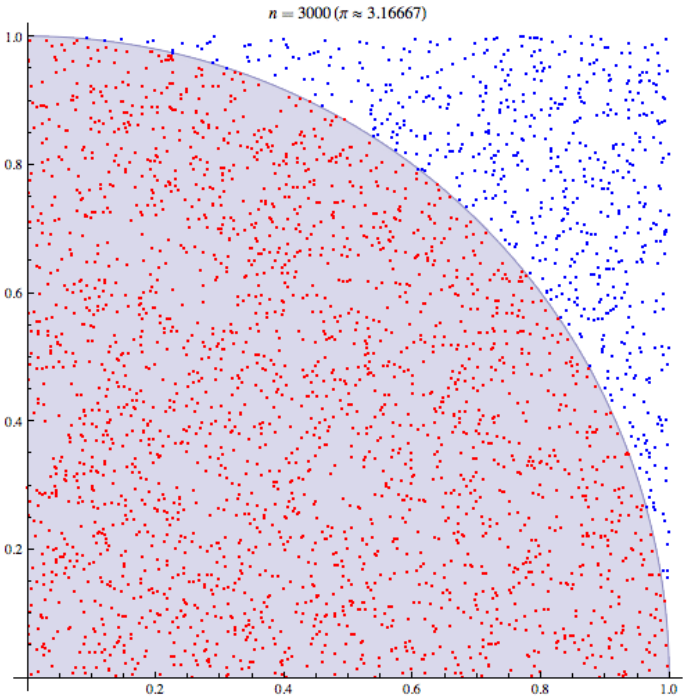
- In the middle of 2000, MacOSX needs som power
  - Processing can be offload to Nvidia GPU
  - CUDA already exists as the successor of CgToolkit
- But Apple does not want to be jailed...
  - (as their users ;-))
- Apple impulses the creation of Khronos consortium
  - AMD, IBM et ARM came quickly
  - … and Intel & Nvidia came backwards...

What OpenCL offers
10 implementations on x86

- 3 implementations for CPU:
  - The AMD: the first ...
  - Intel's: very effective (but not always)
  - The OpenSource PortableCL (POCL)
- 4 implementations for GPU:
  - The owner of AMD: for AMD / ATI graphics
  - Nvidia: for the eponymous circuits
  - "Beignet": Open Source for very recent Intel graphics
  - Mesa: Open Source for graphic circuits with good cores
- 1 implementation for Accelerator:
  - Intel's: for the MIC Xeon Phi Knight Corner
- For other platforms, possible (but not experienced)

Which simple code to implement in //? PiMC: Pi by the method of the target

- Historical example of the Monte Carlo method
- Parallel implementation:
  - Equivalent distribution of iterations
- From 2 to 4 parameters
  - Number of iterations
  - Parallel Rate (PR)
  - (Type of variable: INT32, INT64, FP32, FP64)
  - (RNG: MWC, CONG, SHR3, KISS)
- 2 simple observables:
  - Estimation of Pi (just indicative, Pi not being rational :-))
  - Elapsed Time (Individual or Global)

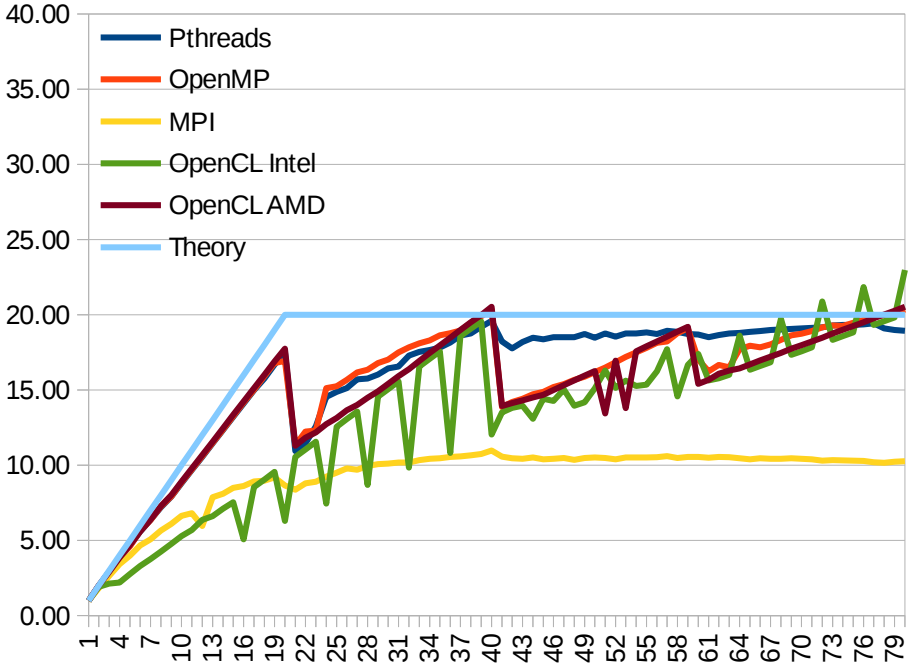


Differences between CUDA/OpenCL
The core of GPU calculations

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

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

Strange behavior ...
On the same processor,



- 20-core processor
  - 2x E5-2670v2
- 5 implementations
  - Of which 2 OpenCL ...
- Similarities ...

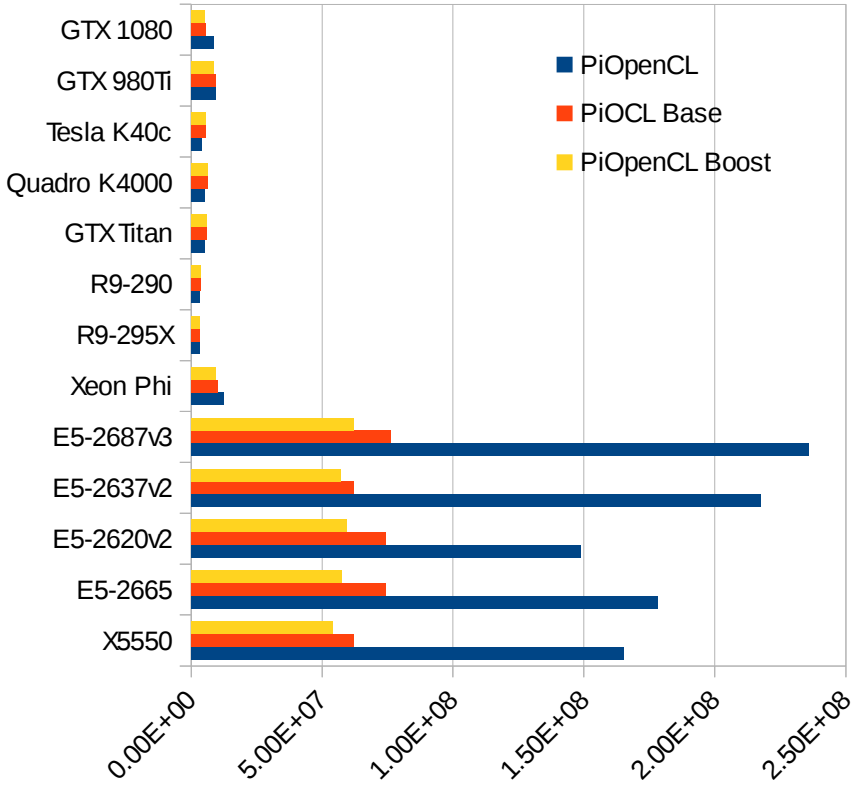
- But mostly different behaviors!

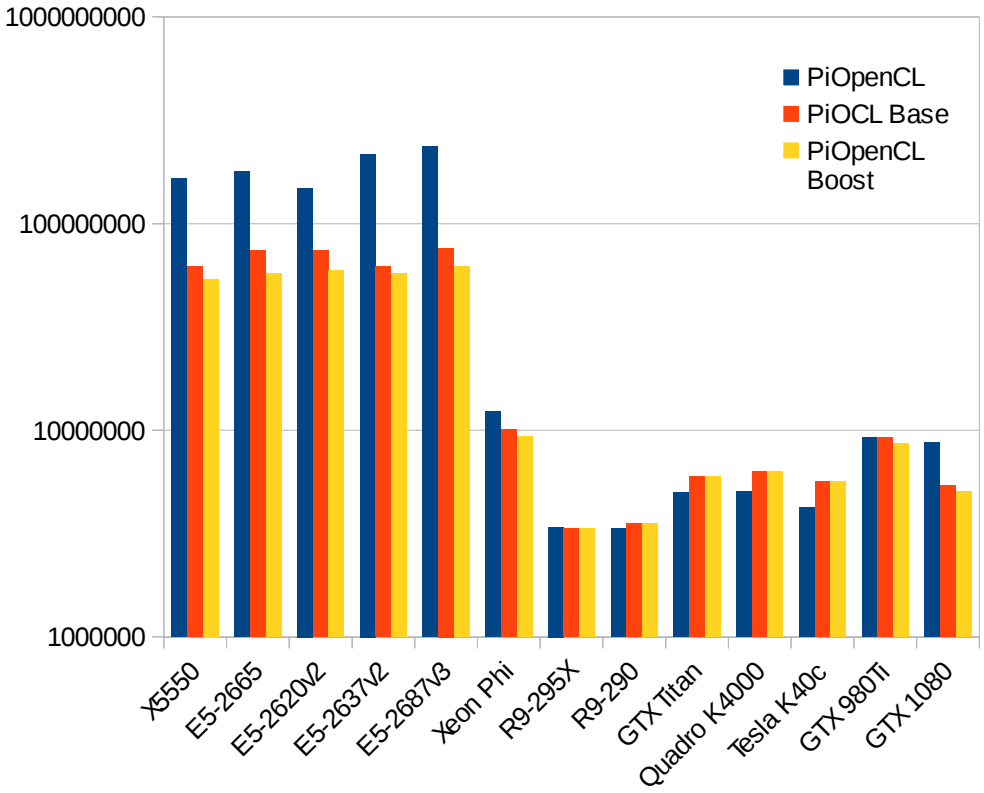
After the CPU and the GPU, Definition of QPU and EPU

- PR: "Parallel Rate" or "Parallelism Plan"
- CPU: Central Processing Unit
- GPU: Graphical Processing Unit
- QPU: Quantum Processing Unit
  - For a "use", launch with PR = 1
  - Example: launch for Threads = 1, Blocks = 1 or Work Items = 1
- EPU: Equivalent Processing Unit
  - For "use", optimum power for a given RA
  - Example: for an R9-290x, optimum for 8x the number of "shaders"

With 1 QPU, low "regime" ... The CPU explodes the GPU!

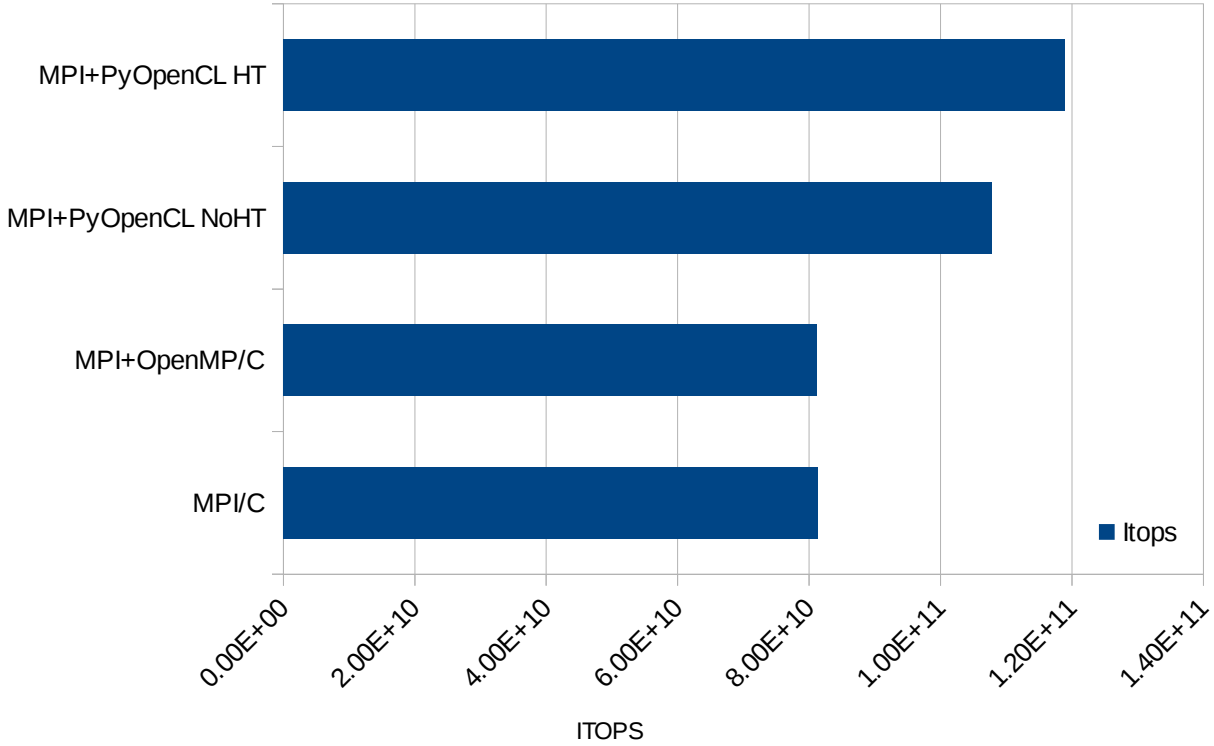
- From 20 to 50x slower!





- 1.5 order of magnitude ...

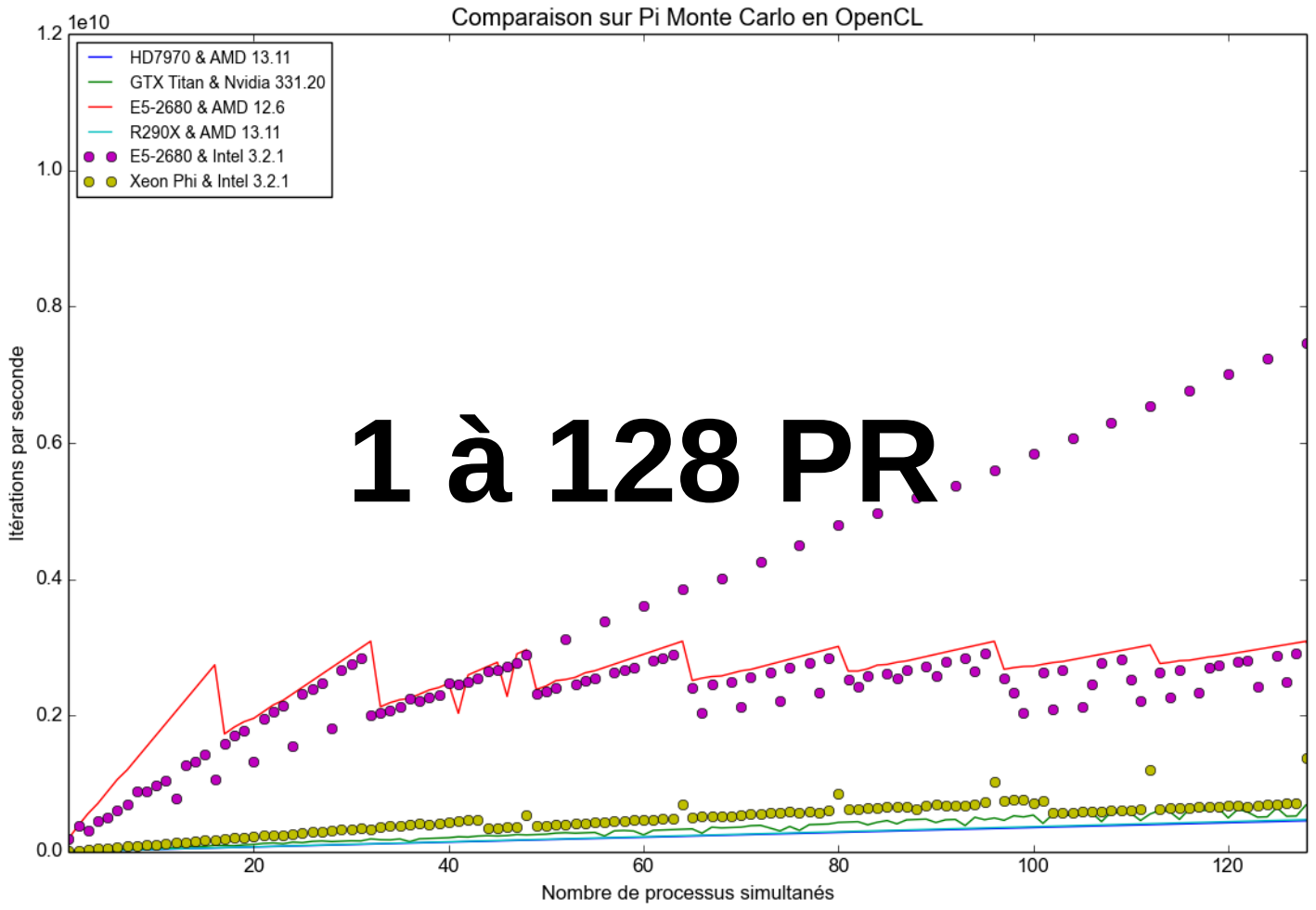
Is Python effective?
A quick comparison ...



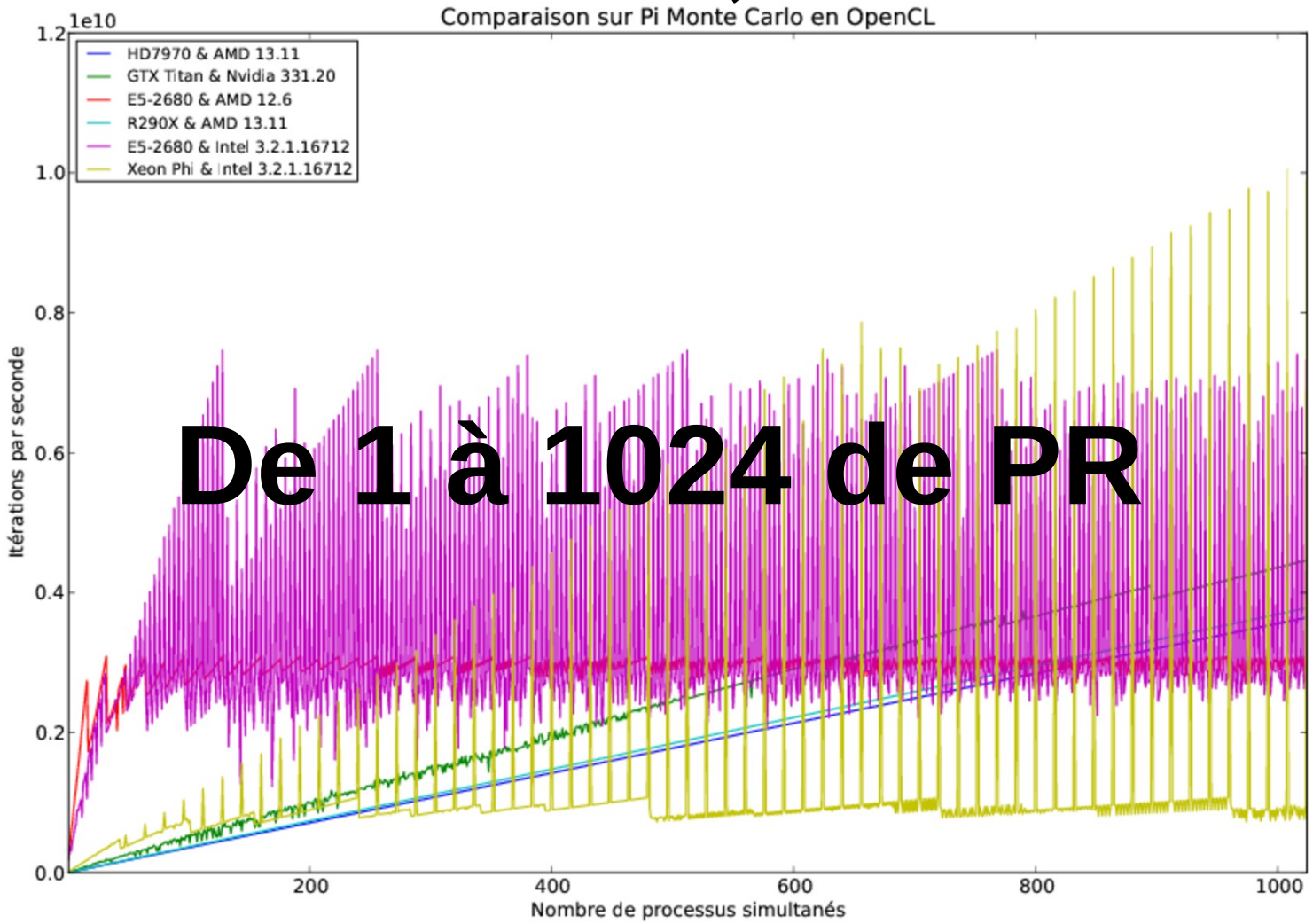
- 64 nodes with 512 cores
- 3 implementations (2 hybrid)

Pi Monte Carlo: the match ... Python vs C & CPU vs GPU vs Phi

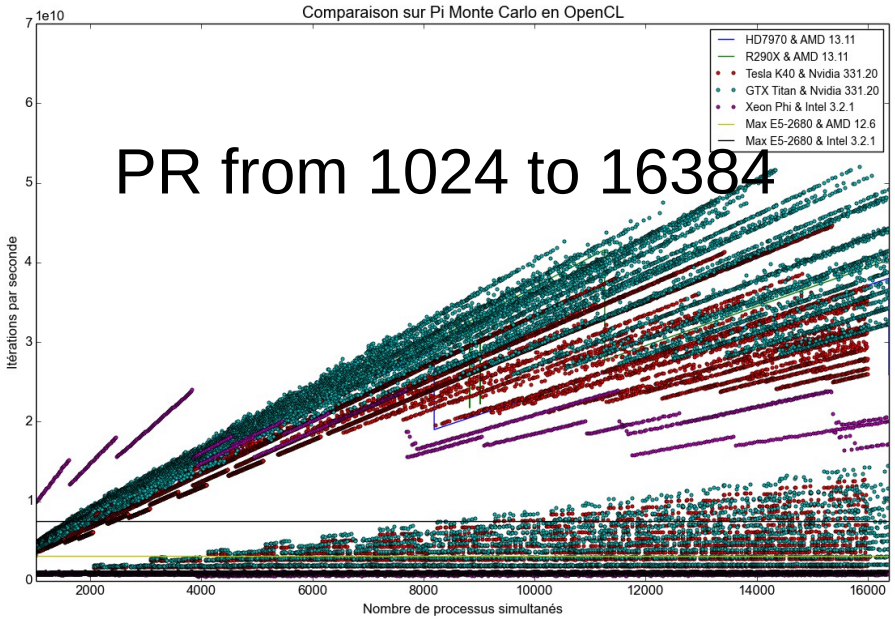
Small comparison between * PU
Low pararegime: the reign of the CPU

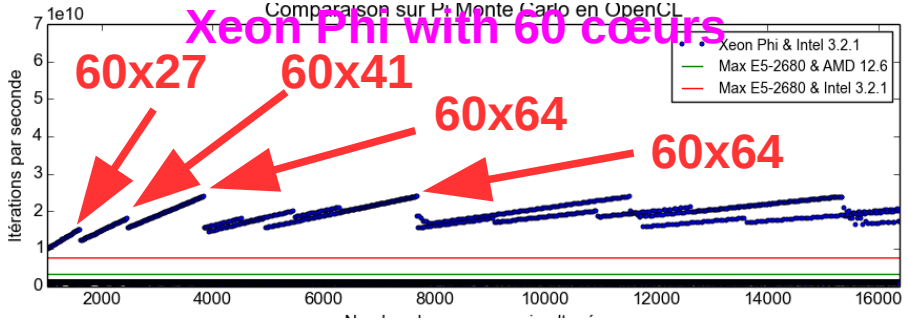


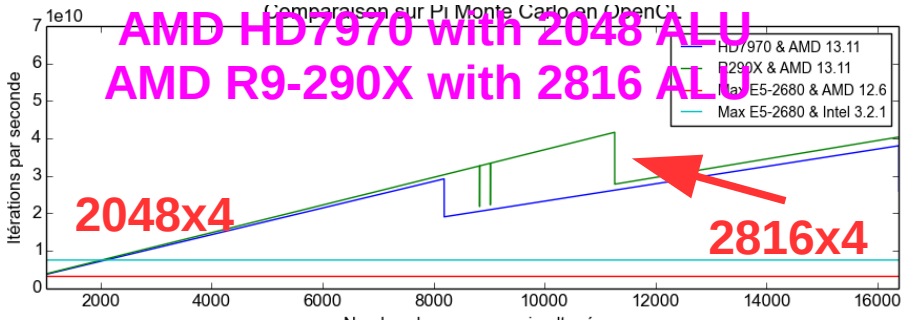
Small comparison between *PU
MIC Phi Out of Wood, GPU

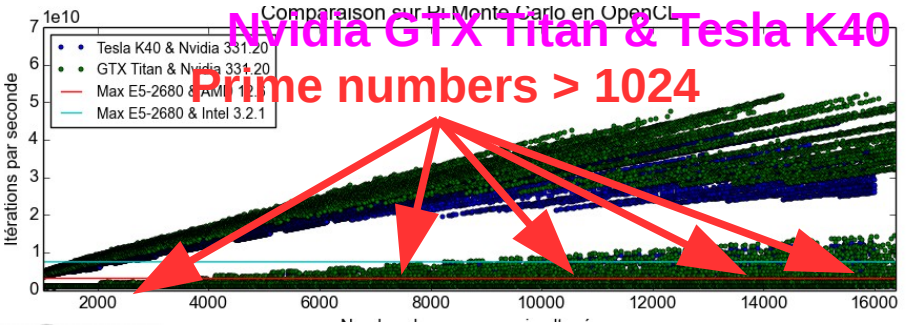


Deep Exploration of PR ParaRégime from 1024 to 16384

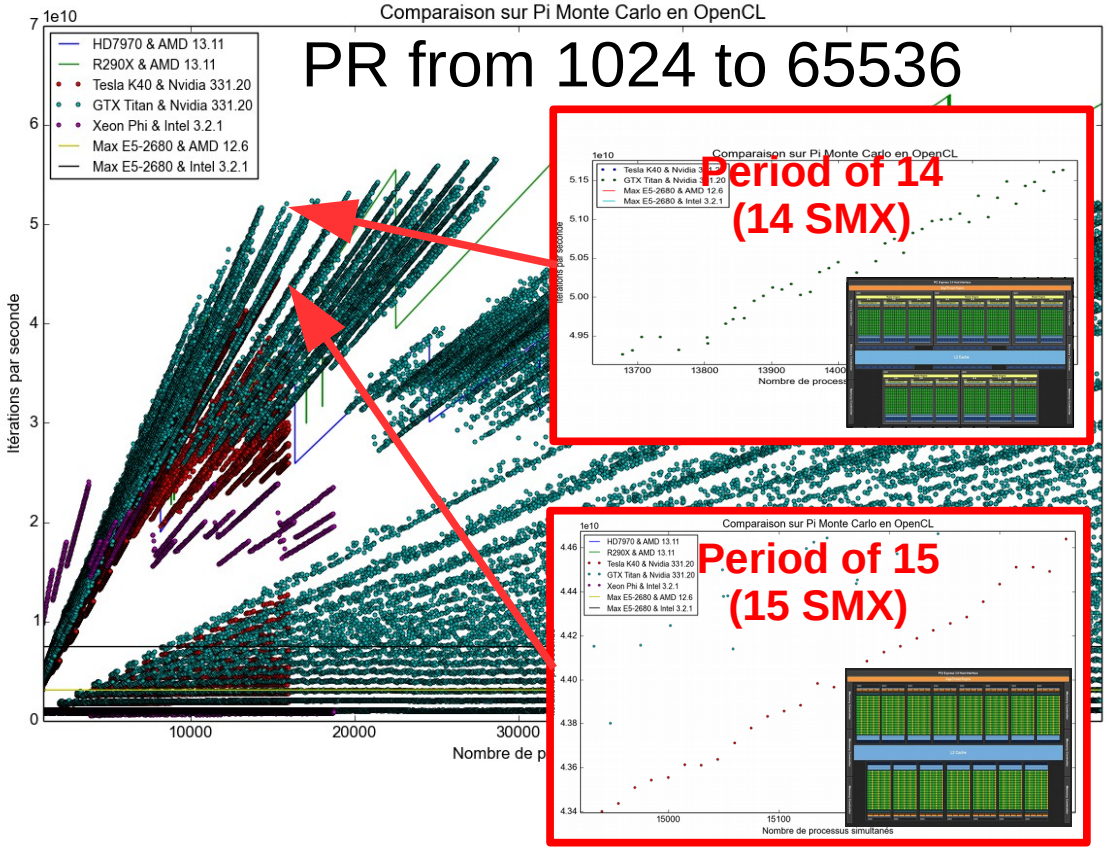




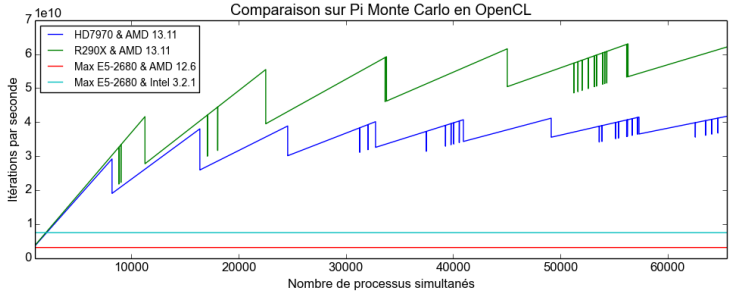




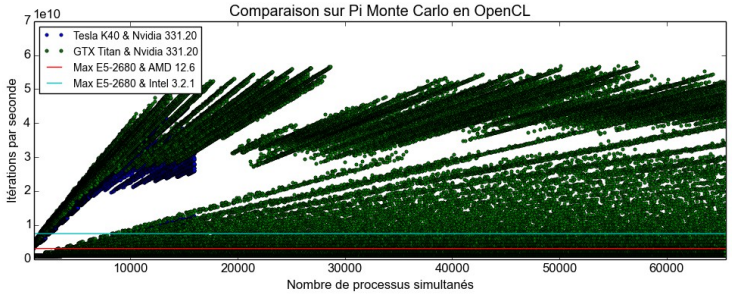
Very deep exploration of PRs
Regime of //ism from 1024 to 65536



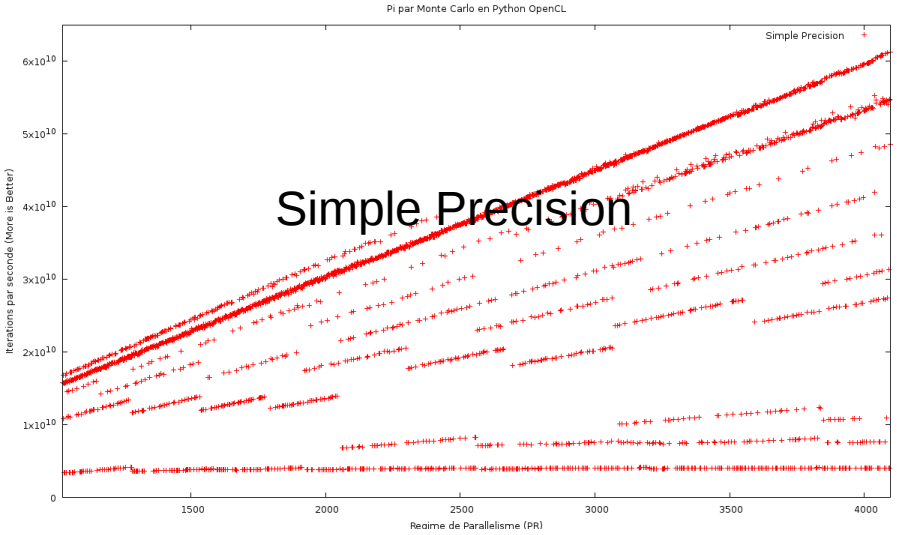
AMD HD7970 & R9-290
Large Period : 4x number of ALU



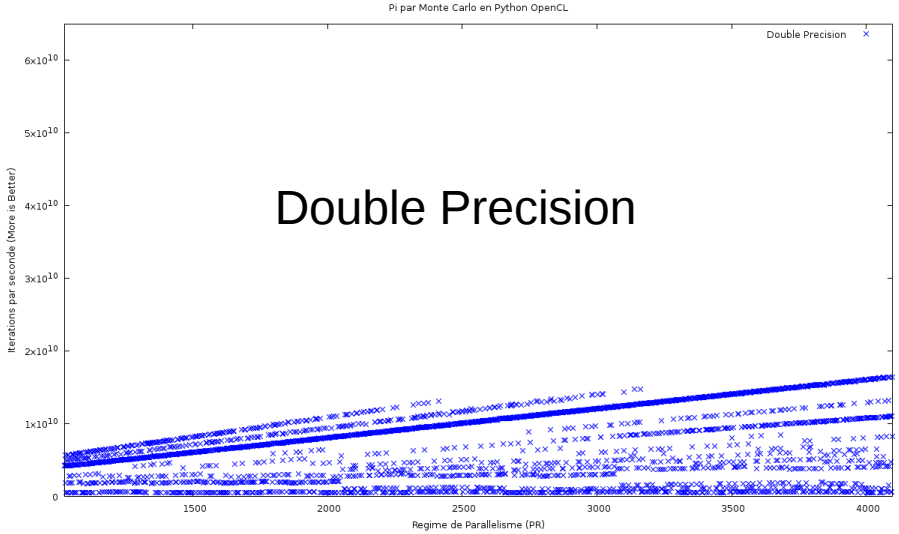
Nvidia GTX Titan & Tesla K40
Small Period : number of SMX units



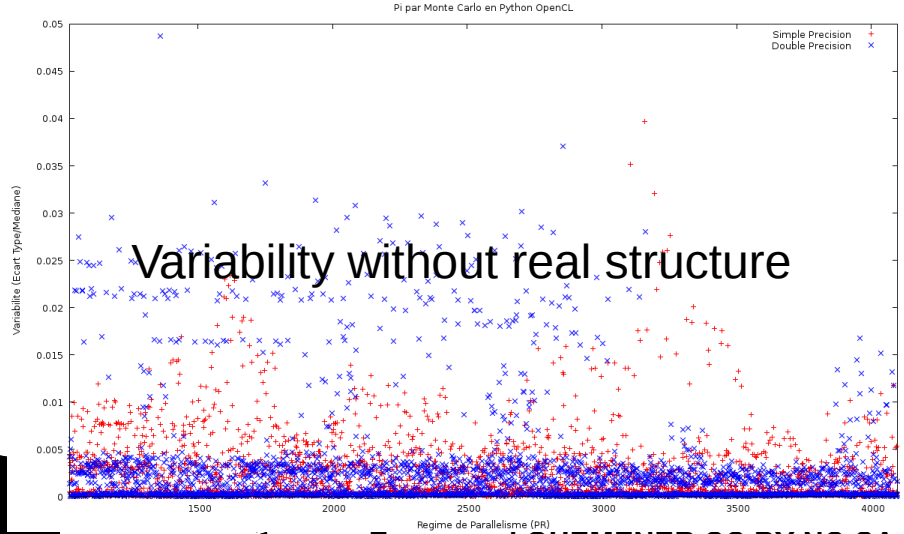
And the latest Nvidia GTX1080, Always affected with oddities?



Simple Precision



Double Precision

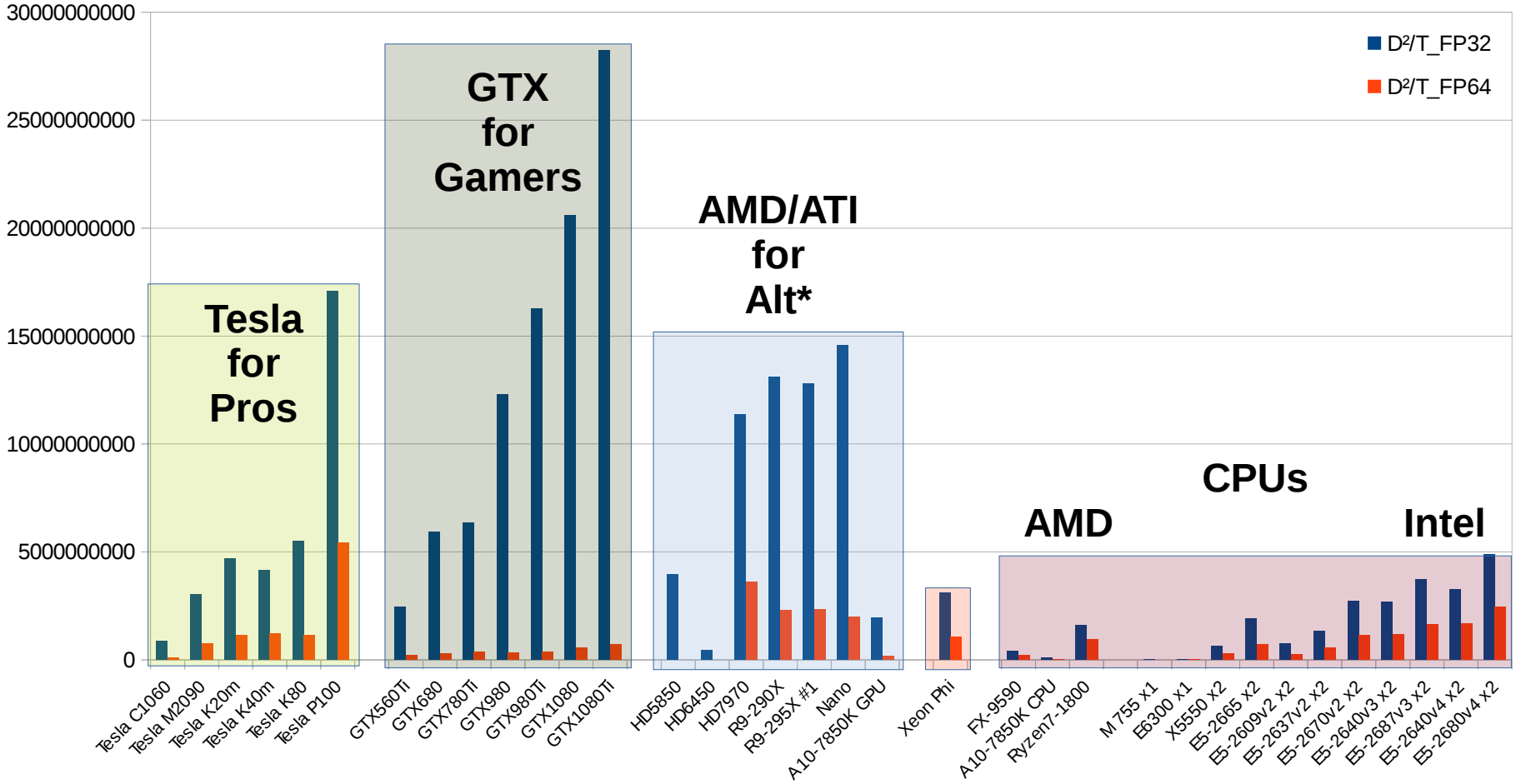


Variability without real structure

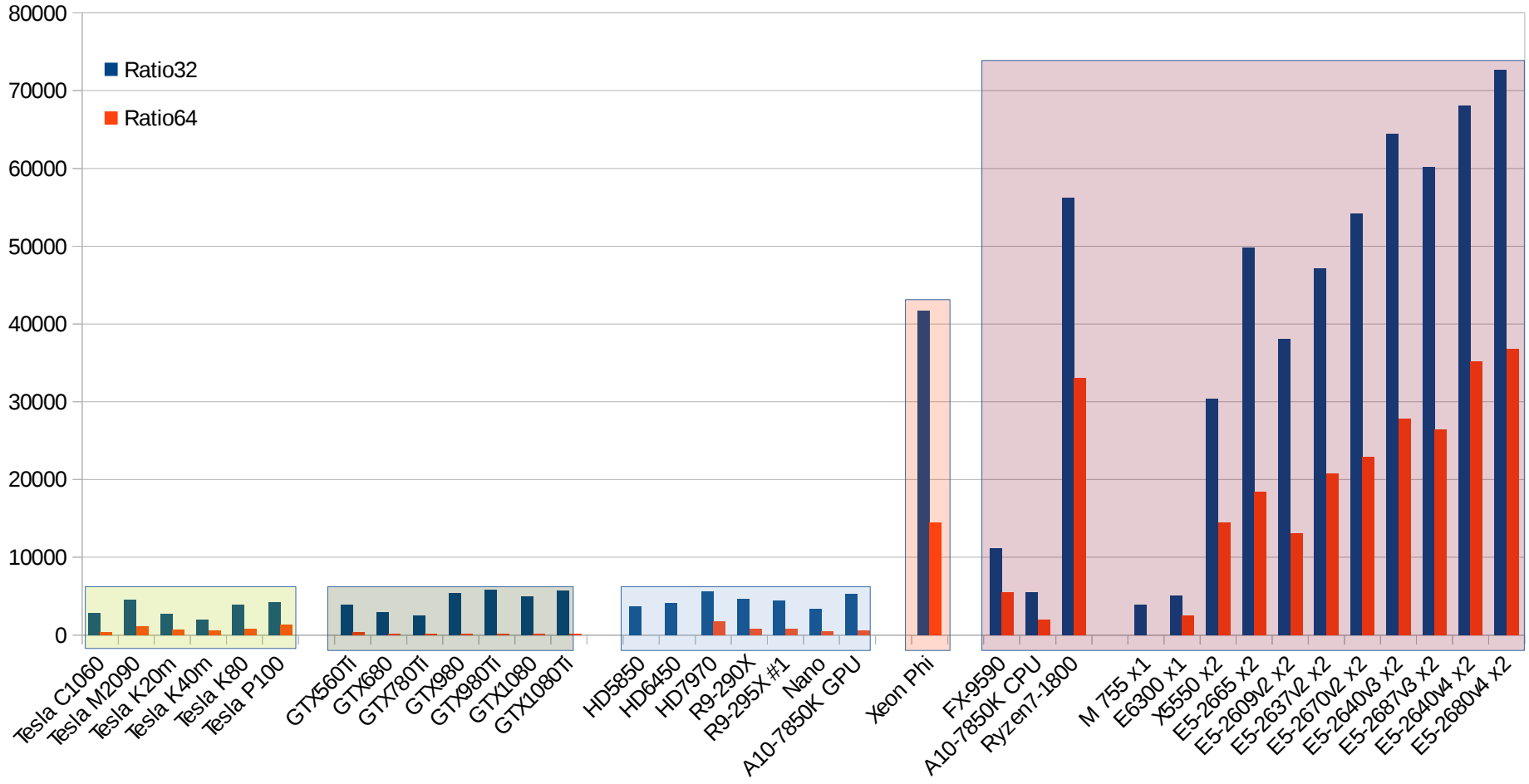
"Back to the Physics" A Newtonian N-body code ...

- Pass on a code "fine grain"
  - Code N-body very simply integrated
  - Newton's second law, autogravitating system
- Differential integration methods:
  - Euler Implicite, Euler Explicit, Heun, Runge Kutta
- Settings :
  - Physical: number of particles
  - Numeric: no integration, number of steps
  - Then: influence FP32, FP64

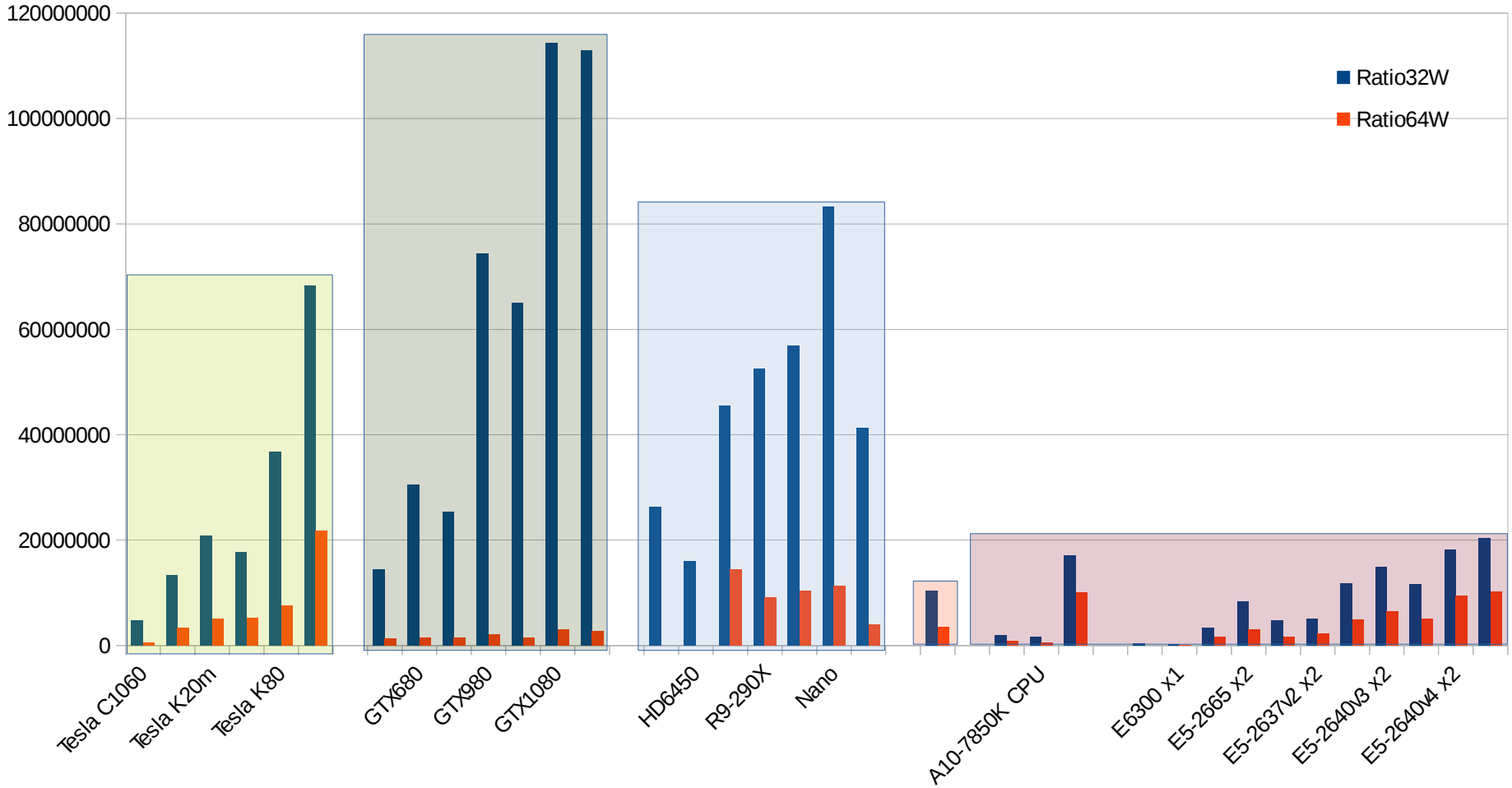
Very different performances Nvidia, AMD, Intel, Single/Double



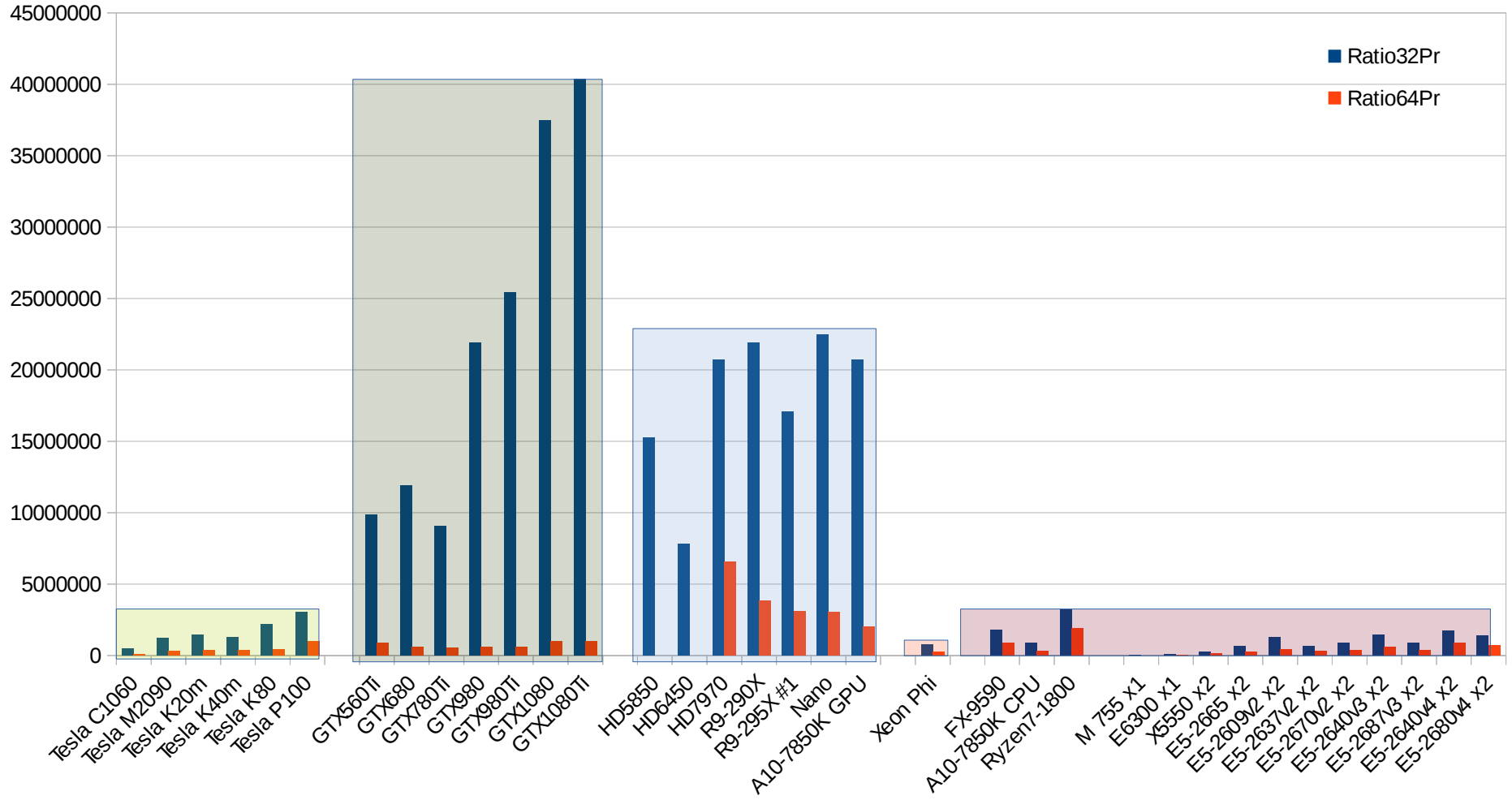
Performances normalized to cores Nvidia, AMD, Intel, Single / Double



Performances normalized to TDP
Nvidia, AMD, Intel, Single / Double



Performances normalized to Price Nvidia, AMD, Intel, Single/Double



But what's the use of all this? Characterize & avoid regrets ...

- What you must remember :
  - In a standard machine, in 2016, the "raw" power is provided by the GPU
  - For a low parallelism, the CPU is much more efficient than the GPU
  - A GPU is greater than the CPU only for PR> 1000
  - The GPU achieves its optimum ONLY for some PRs
  - Without a deep characterization, disappointments to foresee ...
  - OpenCL is a good compromise like language * PU
  - Python remains the fastest approach of OpenCL
- What to do: experiment!