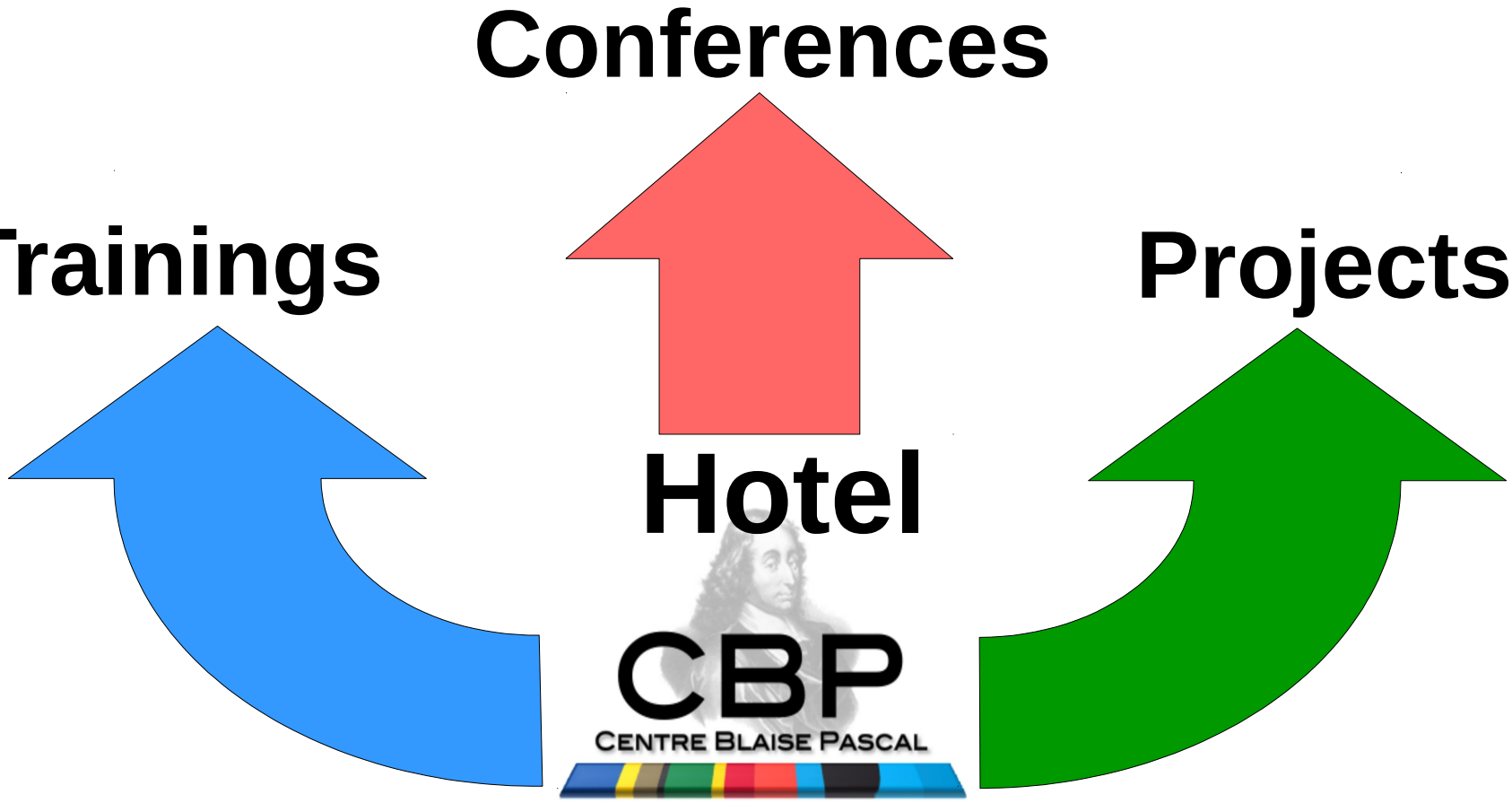
Presentation of IT resources provided by Blaise Pascal Center

From “modelisation house”
To “test center”

Emmanuel Quémener

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



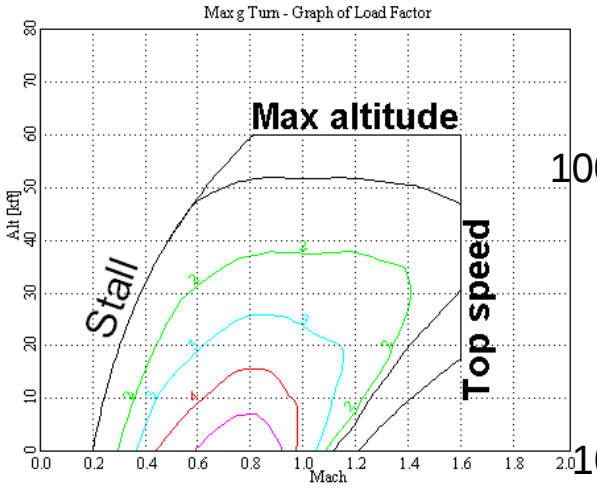
Centre Blaise Pascal ~ Dryden FR
Small illustrative example

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

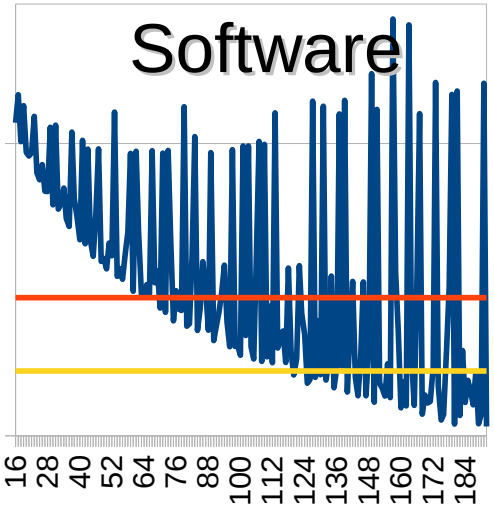
To recycle, to reuse and explore from new domains

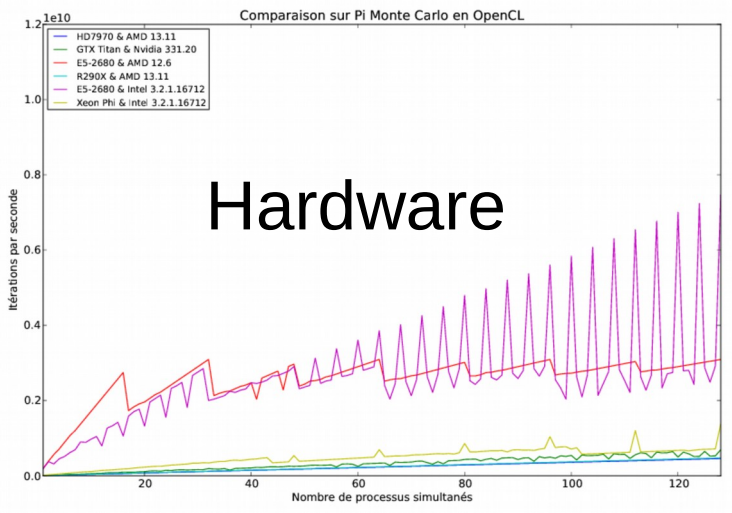
From engines to parallel envelope
A necessary exploration?

Flight Envelope



Parallel Envelope





- Speed/altitude
- Load factor

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

CBP also a **Test Center**

Experimental platform with 10 technical benchs

- **Multi-nodes**: 5 clusters from 4 to , Nodes/Cores : 4/24, 8/64, 8/64, 64/512
- **Multi-cores** : more than 100 machines from 2 to ,
  - Nodes/Servers: from 8 to 28 cores, Workstations: from 2 to 16 cores, 12 types of CPU différents
- **GPU & Accelerator** : of GPU (AMD & Nvidia),
  - GPGPU : 9 ; GPU Nvidia : 25 ; GPU AMD/ATI : 12 types ; Xeon Phi 7120P
- **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

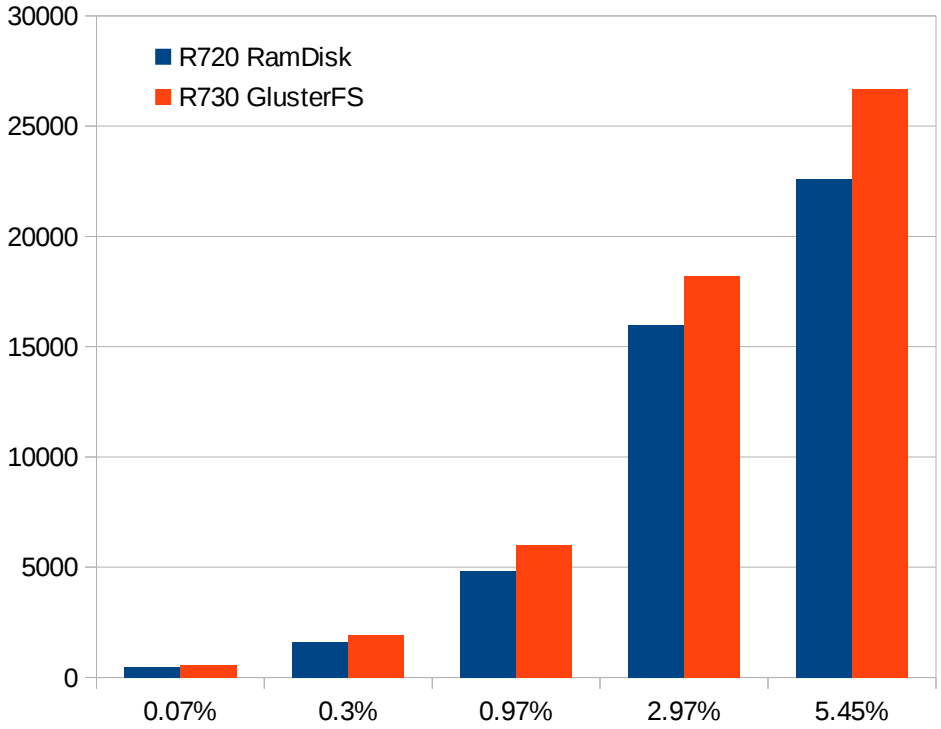
Some « Digital Benchs » permanent or ephemeral !

- Schools in Les Houches about « computational physics » :
  - « standalone » complete virtual machine
  - SIDUS virtual machine
- Digital Humanities :
  - 20 virtual machines and 10 already operational ones
- Geography :
  - Vizualisation workstations for graphical processing
- Biology :
  - Galaxy gateway (with cluster nodes processing)
  - Processing and display stations for MorphoGraphX
  - Server provisioning to study & process under Repeat* with GlusterFS, TMPFS & BRD

To what does it serve?
Example : Repeat* GlusterFS/TMPFS

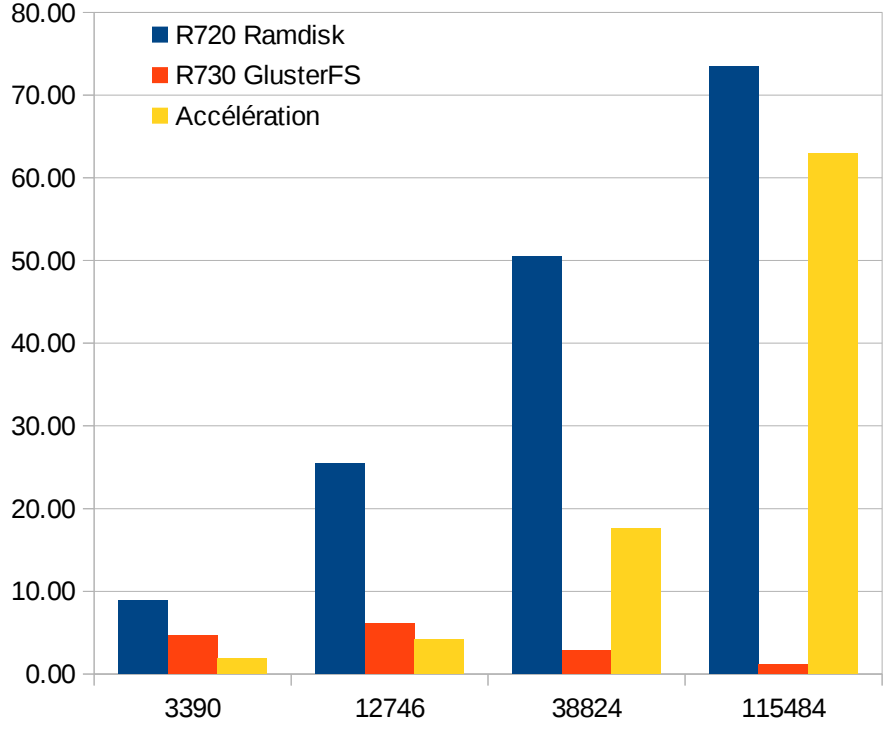
Progression « Input Database Coverage »

Less is Better !



Progression « Family Refinement »

More is Better !



An example of interactions Between Laboratory/CBP/PSMN

Evaluation
Test center

Buy adaptive Equipment GPU

Know-How SIDUS Systems & Networks

AMD

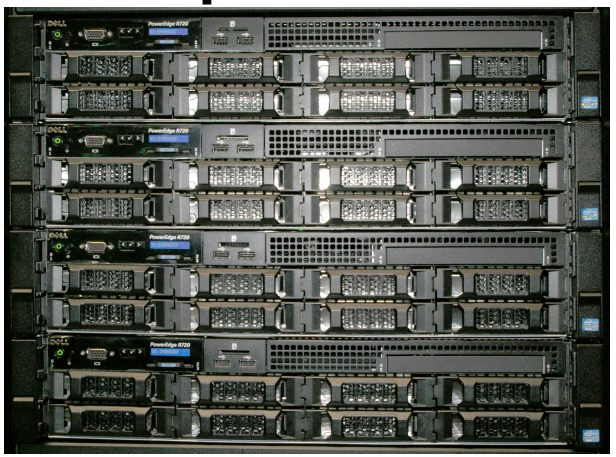
Nvidia

Preproduction Laboratory

Hardware **CBP** / Sidus OS **CBP**

Influence the choice of hardware background

Production Compute Center

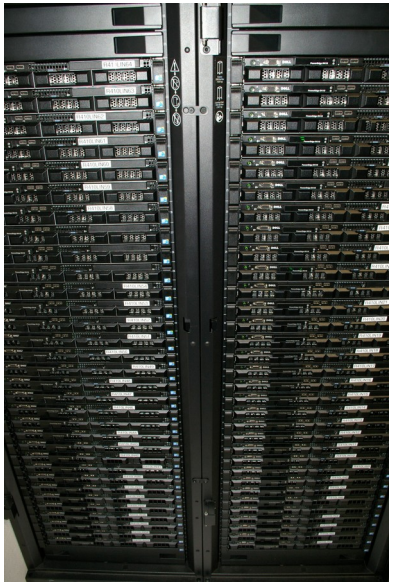


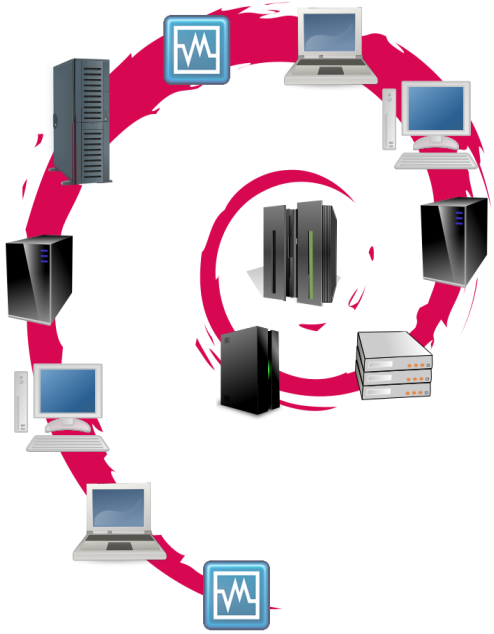
Hardware **LBMC** / Sidus OS **CBP**

Hardware **PSMN** / Sidus OS **PSMN**

How to manage 120+ machines ? SIDUS !

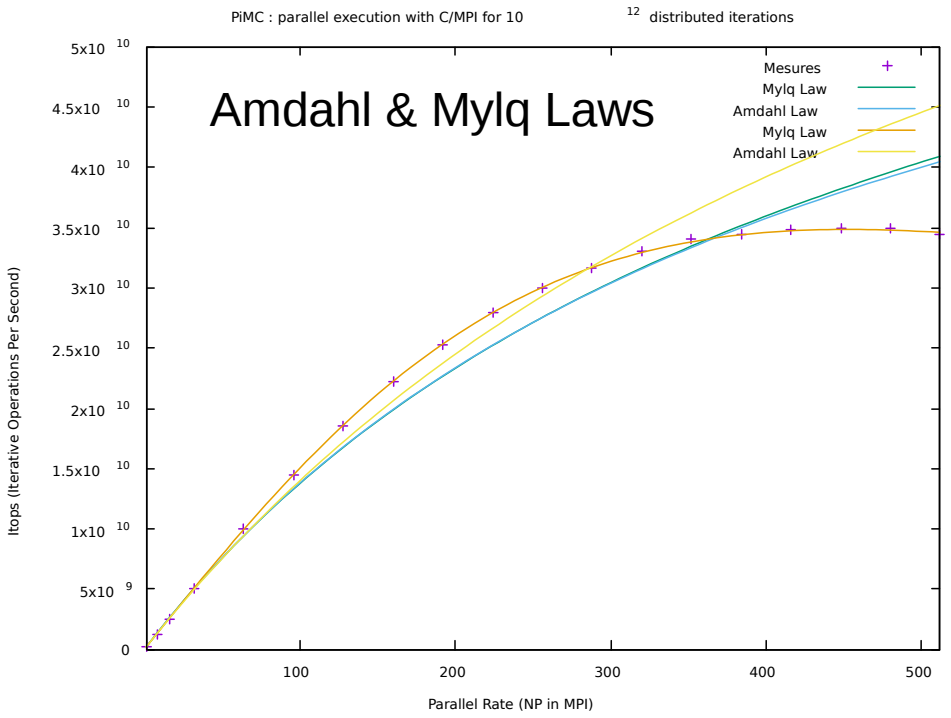
Single Instance Distributing Universal System

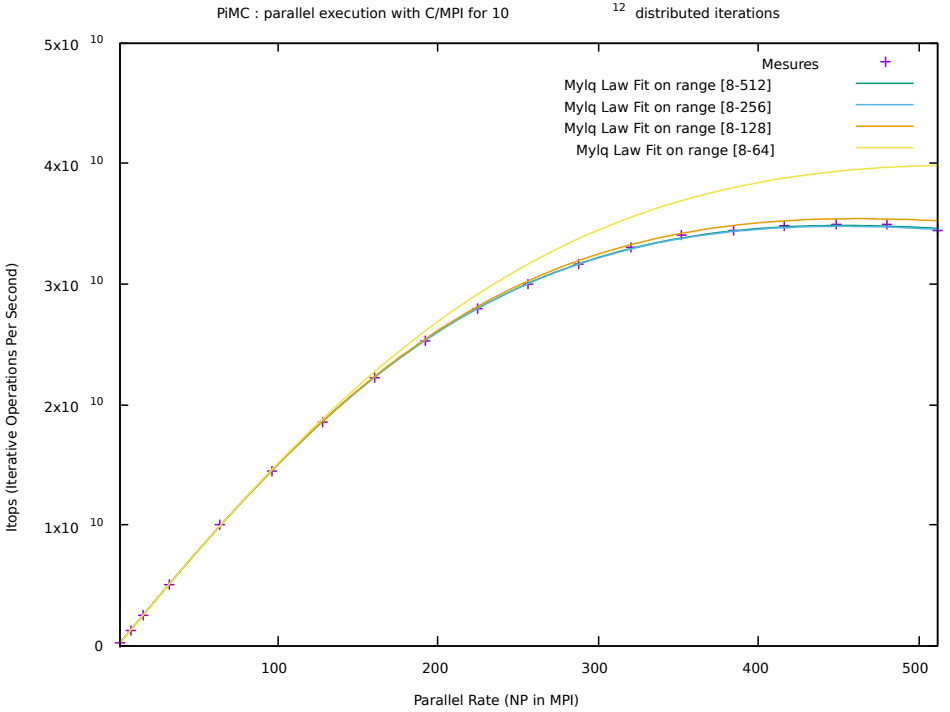




Exemples of studies : predictions... Overshoot Amdahl Law with Mylq One

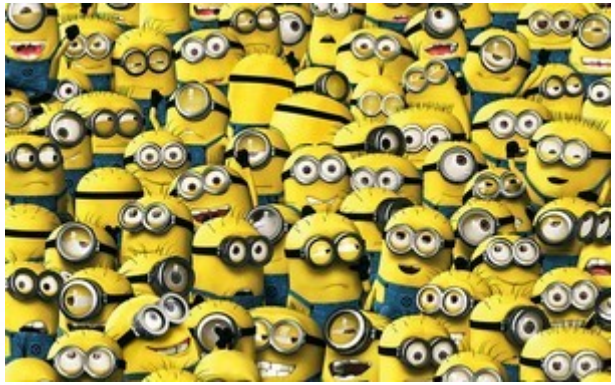
- From T=T1+p/N to T=T1+c*N+p/N





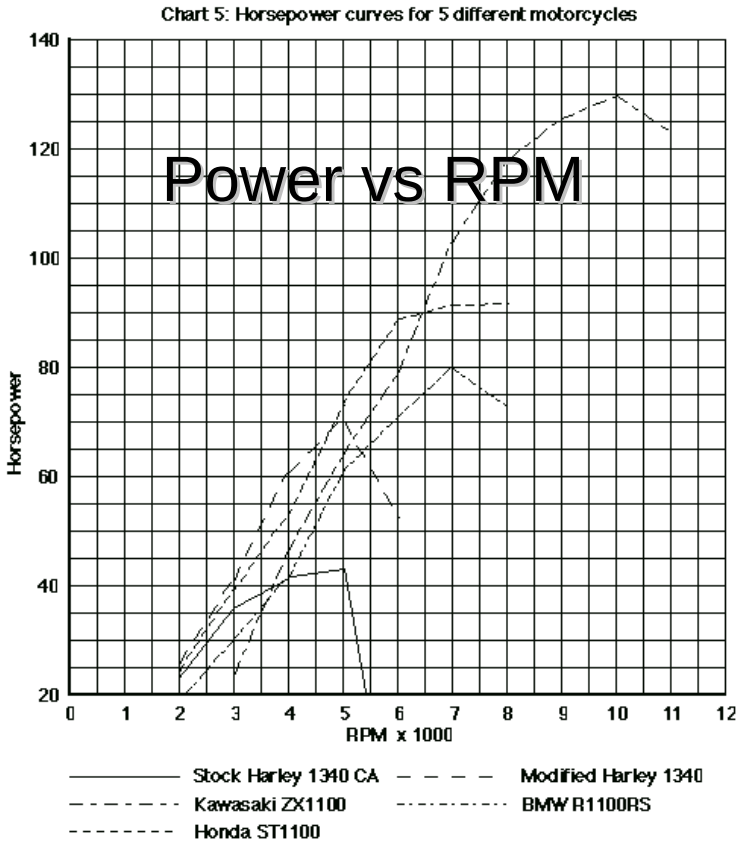
From (multi|many)-cores to myri-alus ?
A parallel laboratory on a chip !

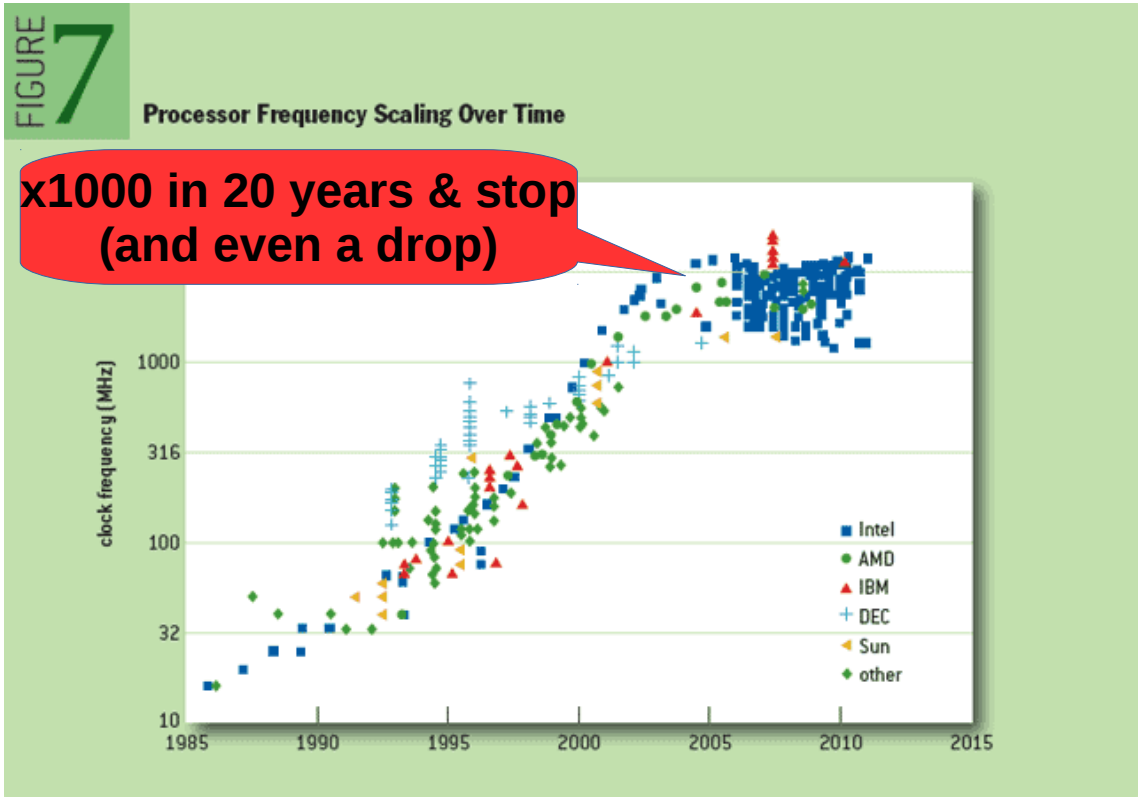
- You got from 1 to dozen of thousands of equivalent PU !
- You got the choice between : **CPU MIC GPU**



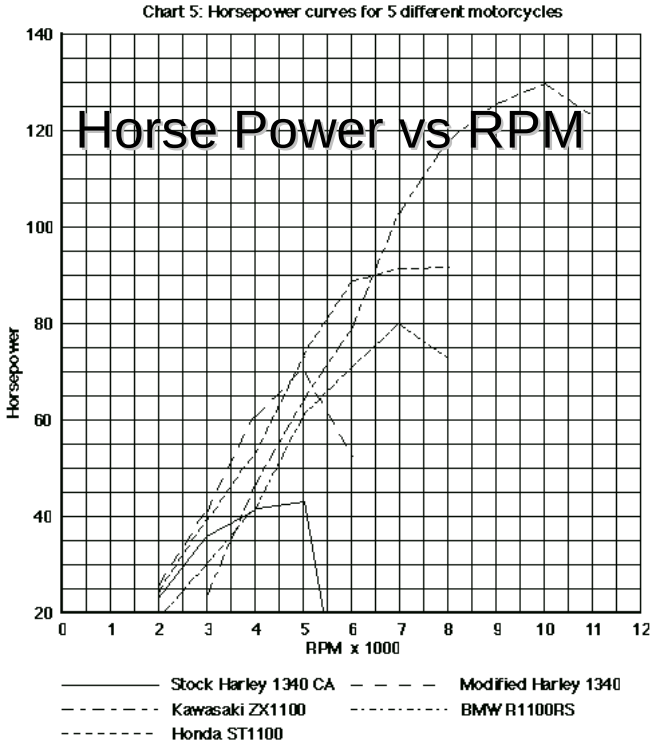
How to choose and distinguish them for its use?

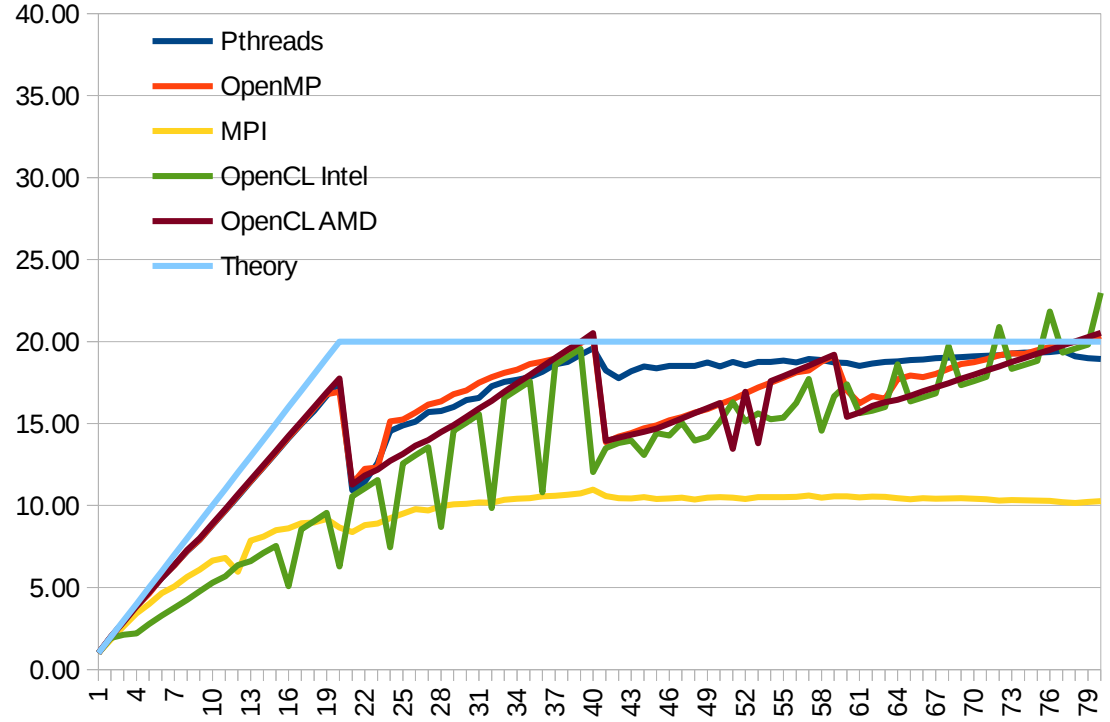
A long long time ago, between 1985... and 2005, variable is frequency !





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

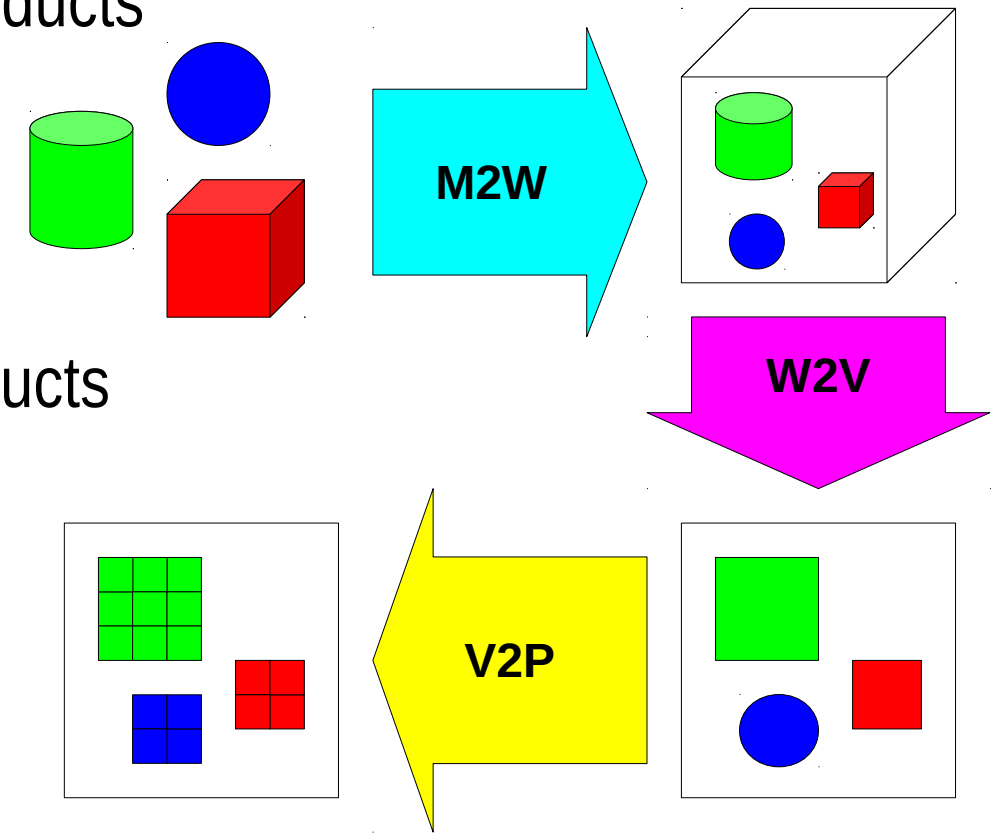




- What "behaviors" for these engines at "load-bearing"?
- The "engine", a system entangling hardware, OS and software

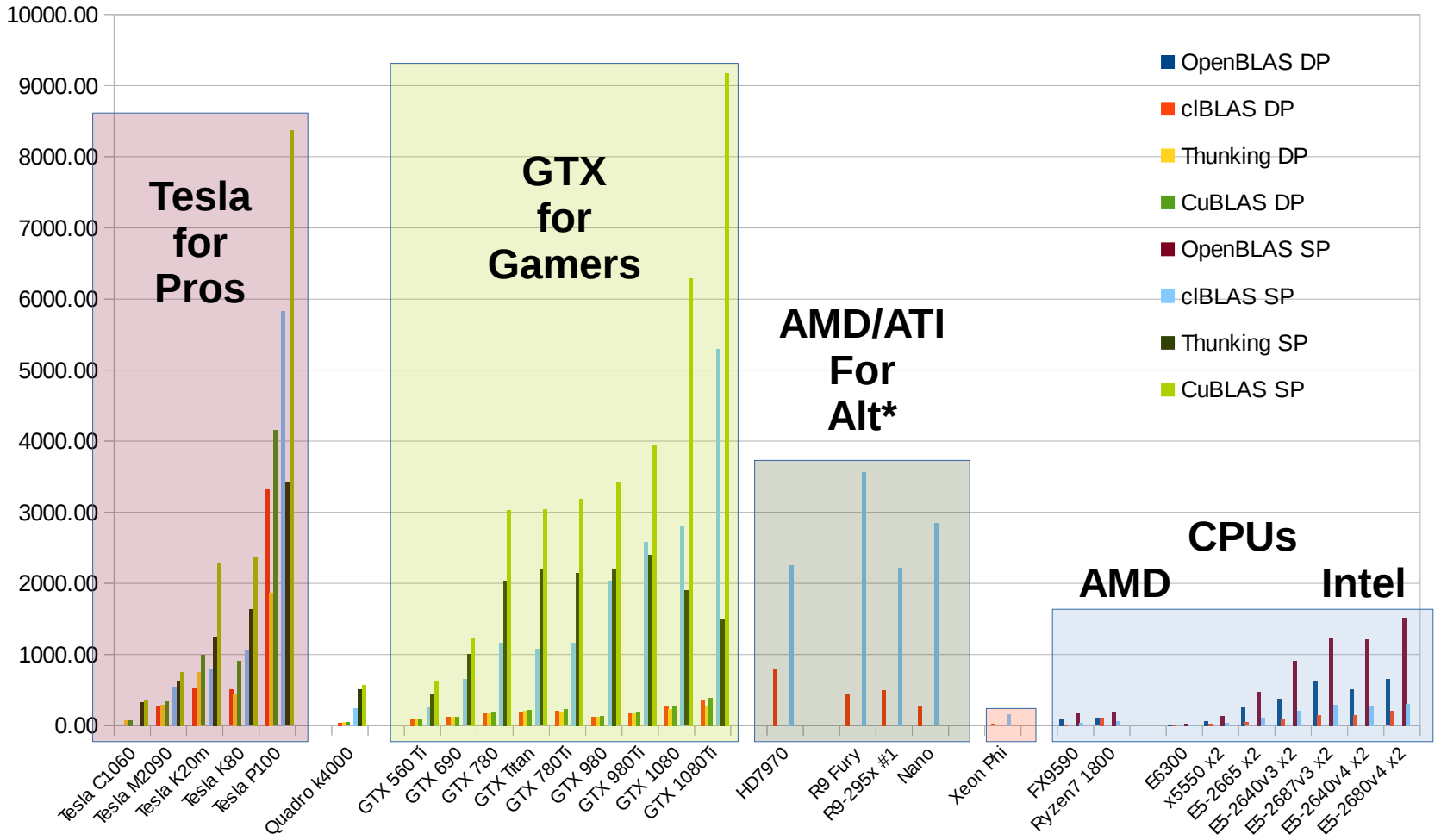
Why is the GPU so powerful? Shadering & Matrix Calculation

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



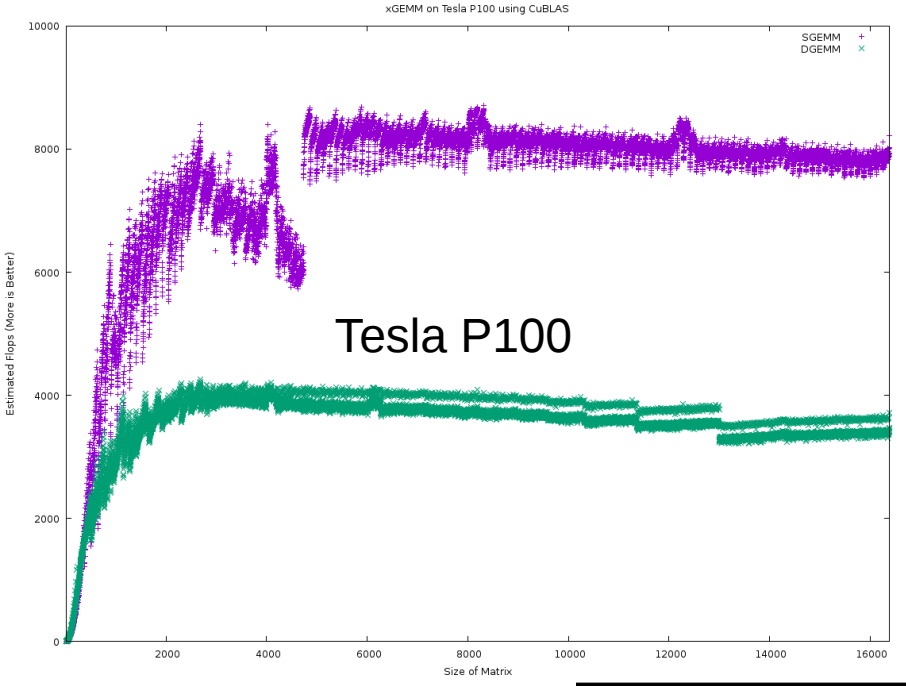
So a GPU: it's a "big" Matrix Multiplier

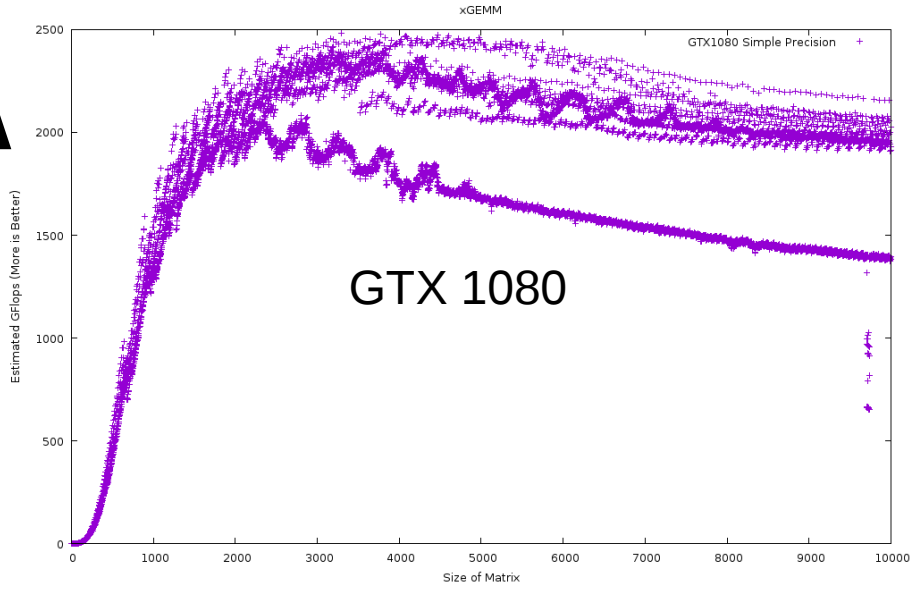
BLAS comparaison : CPU vs GPU xGEMM when x=D(P) or S(P)



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

Performance



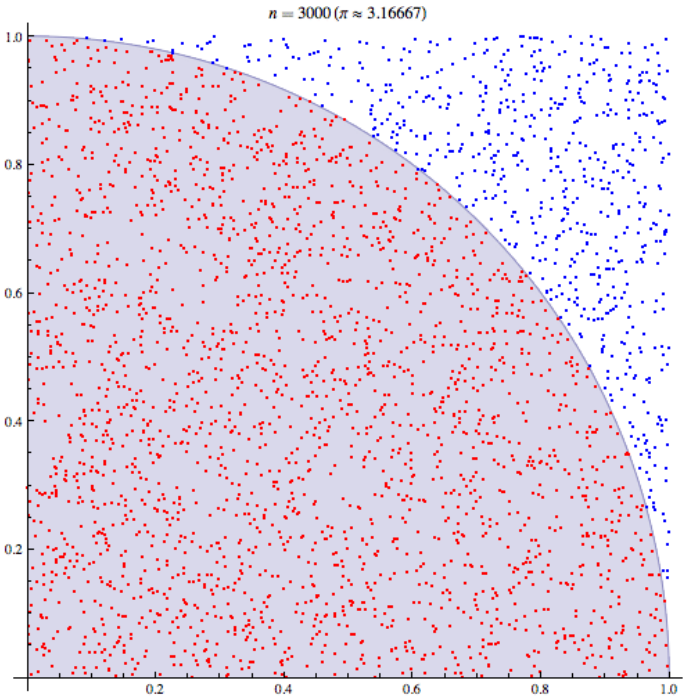


Size

Yes, performance is there, but in specific conditions !

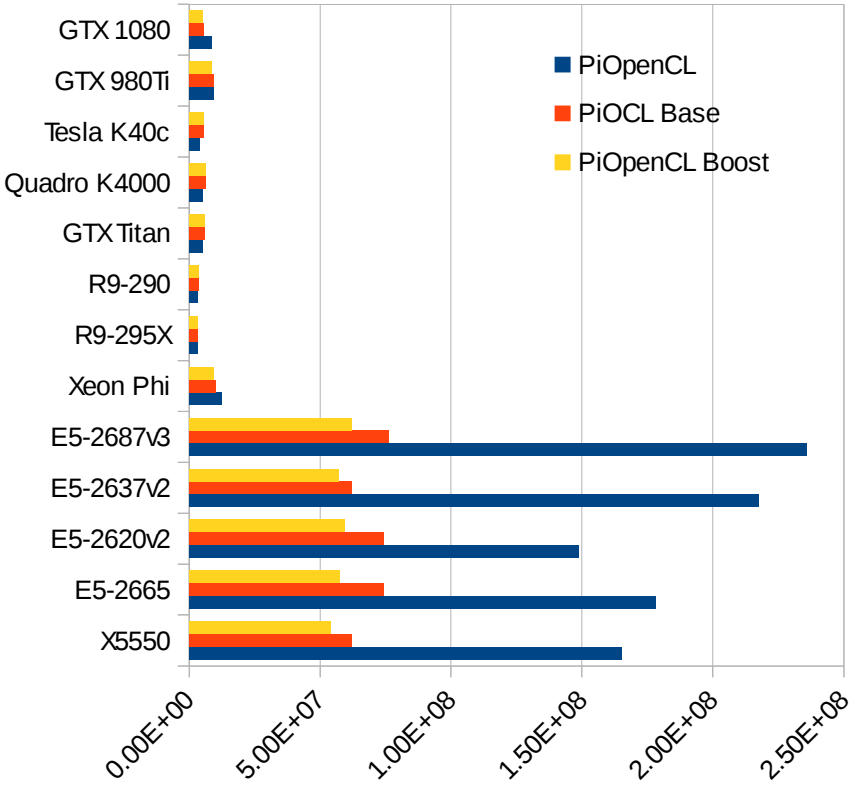
Which simple code to implement in //? PiMC: Pi by the method of the target

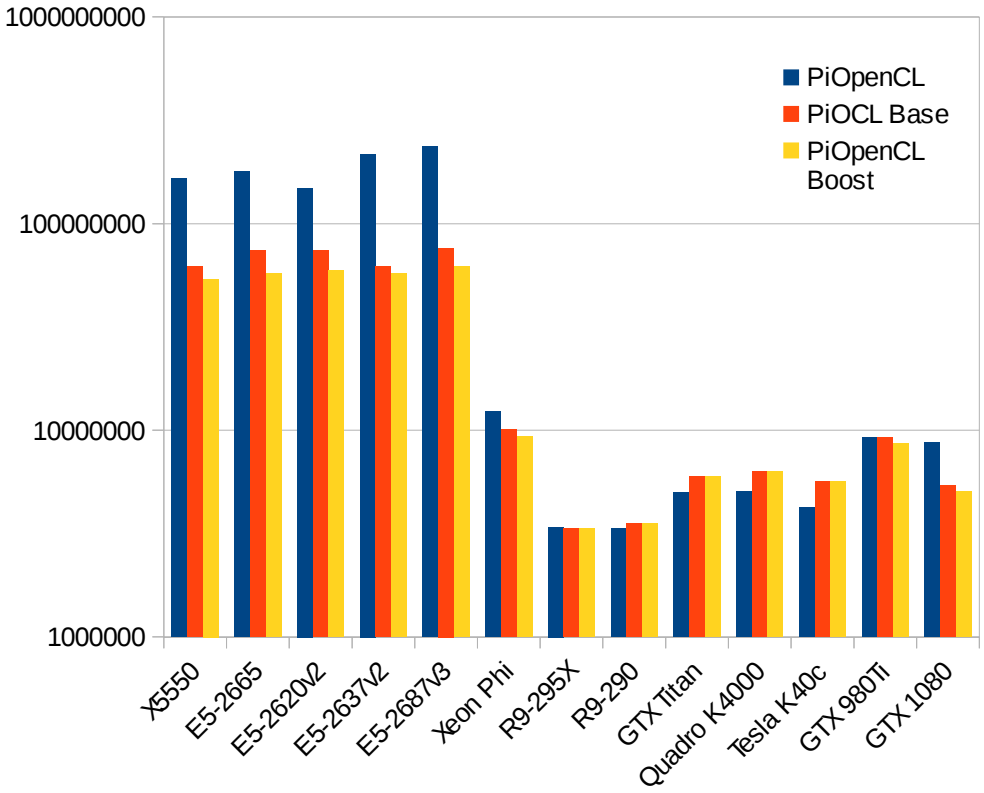
- Historical example of the Monte Carlo method
- Parallel implementation:
  - Equivalent distribution of iterations
- From 2 to 4 parameters
  - Number of iterations
  - Parallel Rate (PR)
  - (Type of variable: INT32, INT64, FP32, FP64)
  - (RNG: MWC, CONG, SHR3, KISS)
- 2 simple observables:
  - Estimation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



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

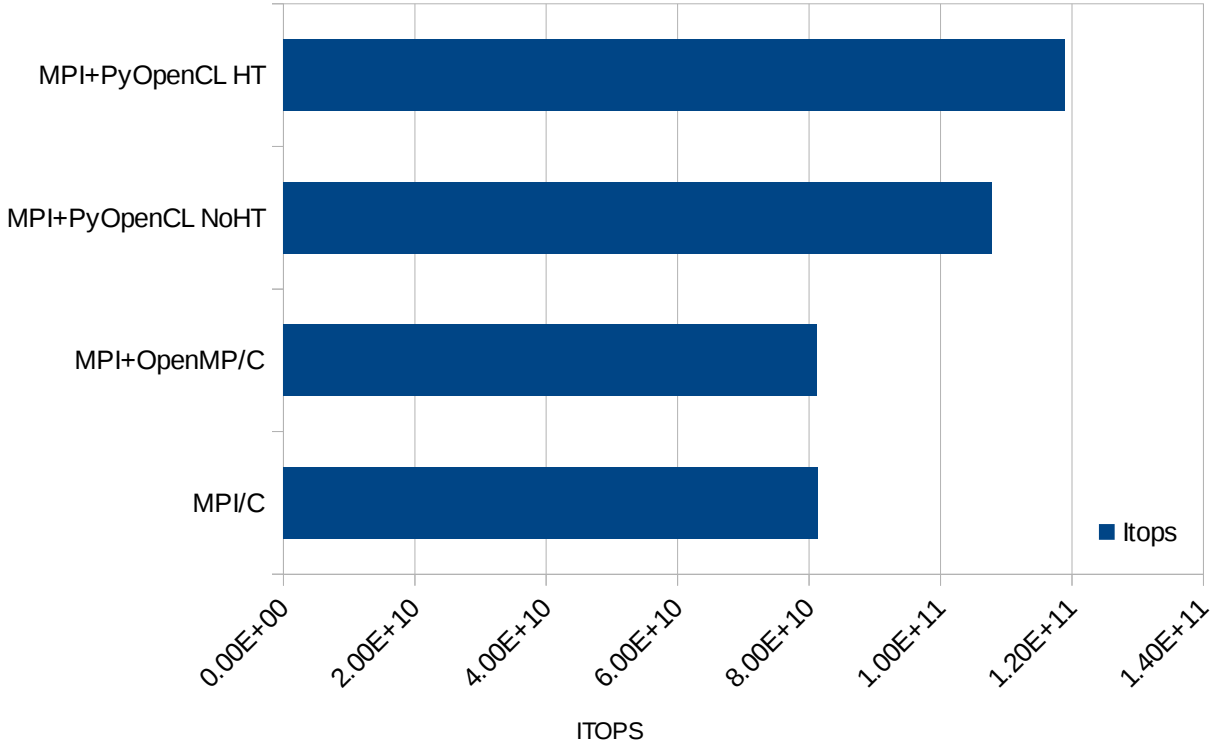
- From 20 to 50x slower!





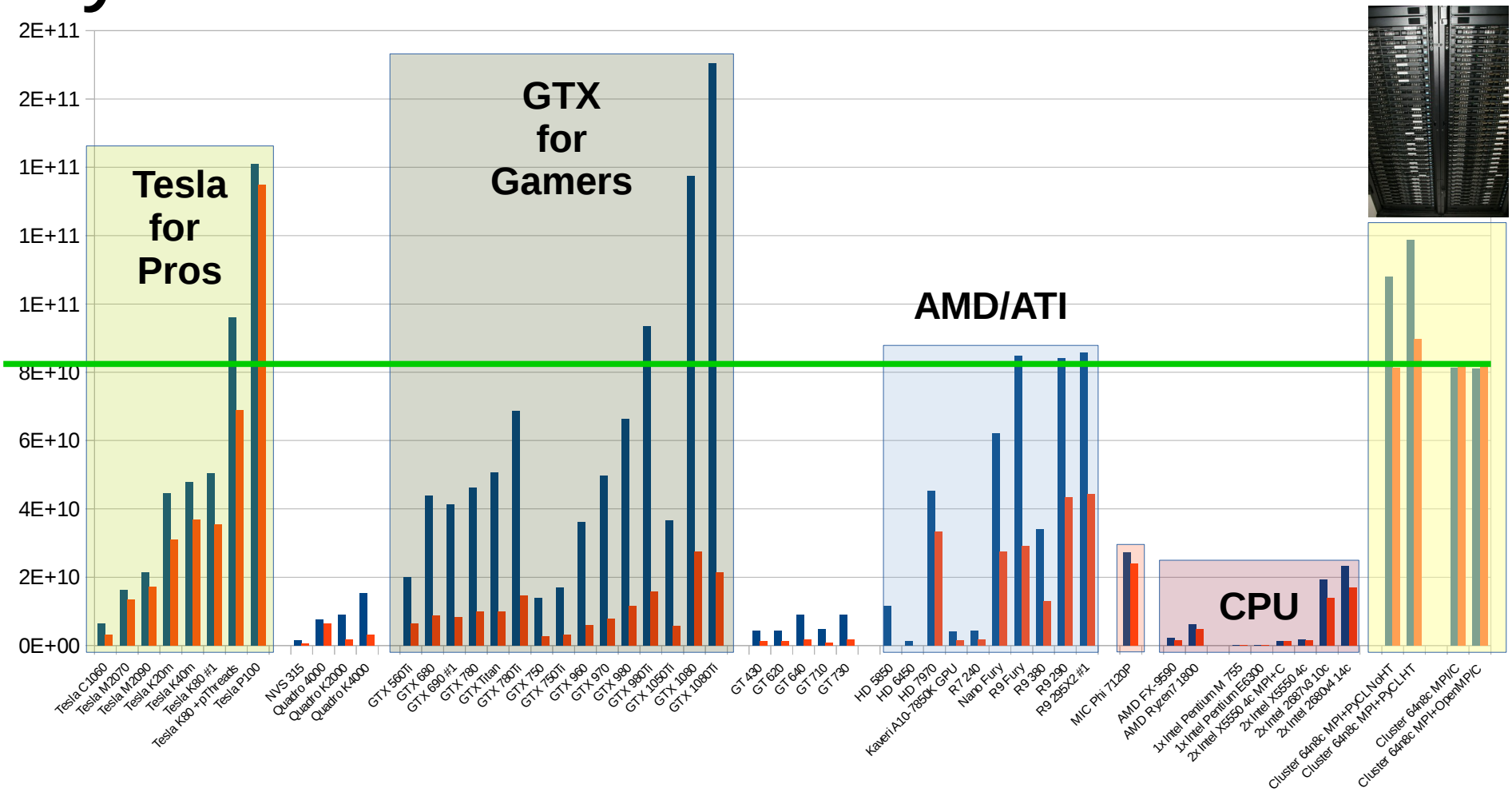
- 1.5 order of magnitude ...

Is Python effective?
A quick comparison ...

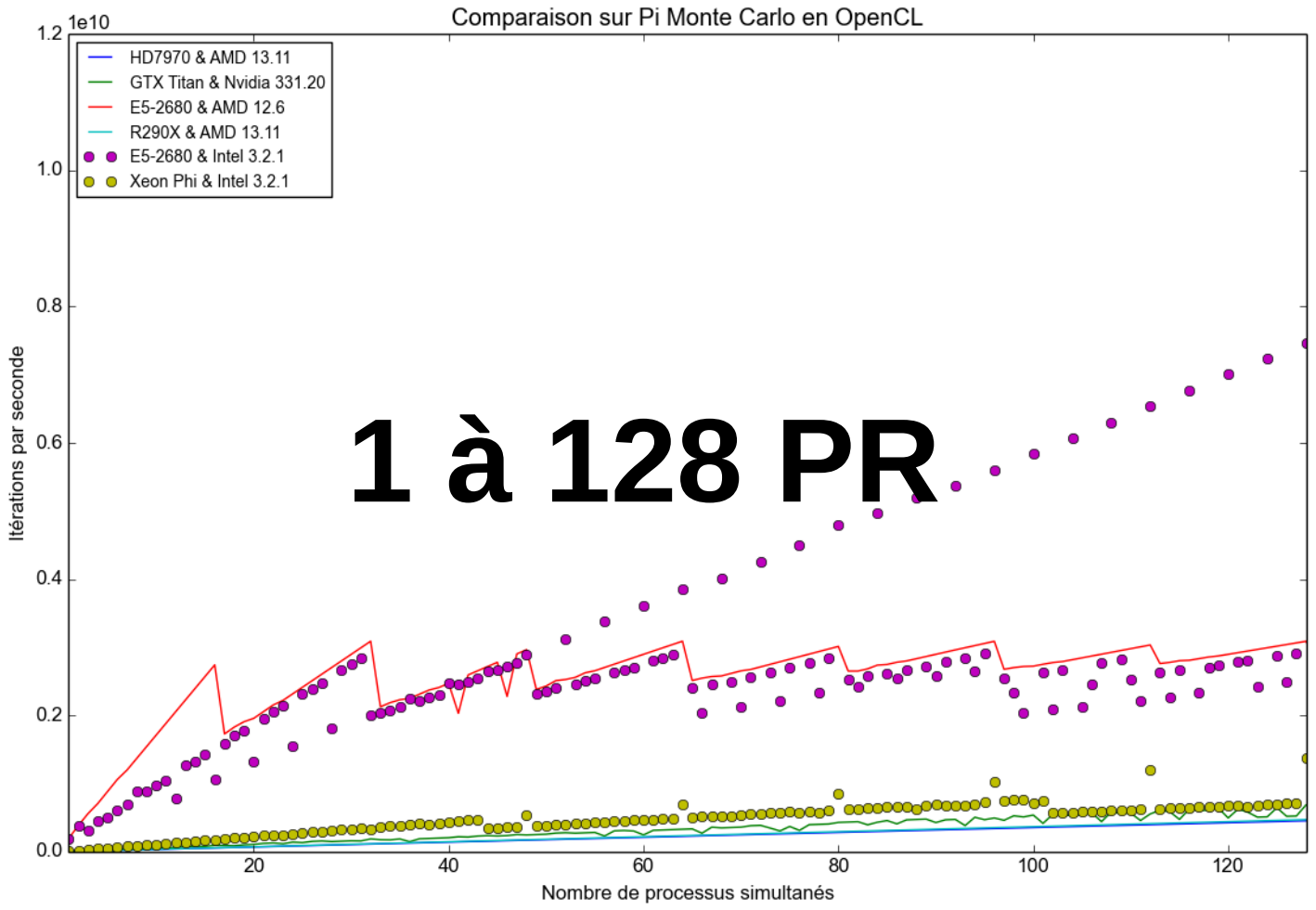


- 64 nodes with 512 cores
- 3 implementations (2 hybrid)

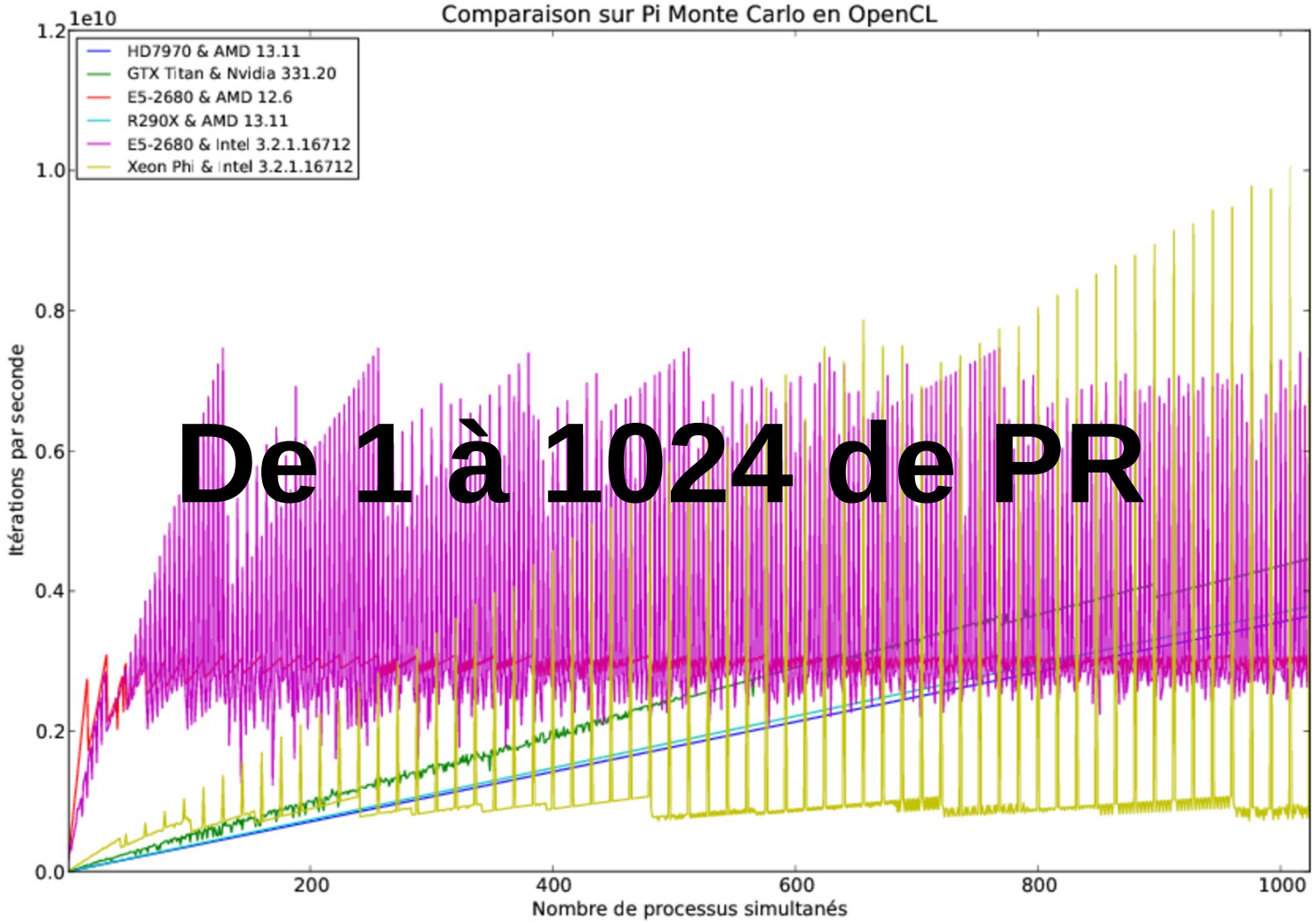
Pi Monte Carlo: the match ...
Python vs C & CPU vs GPU vs Phi



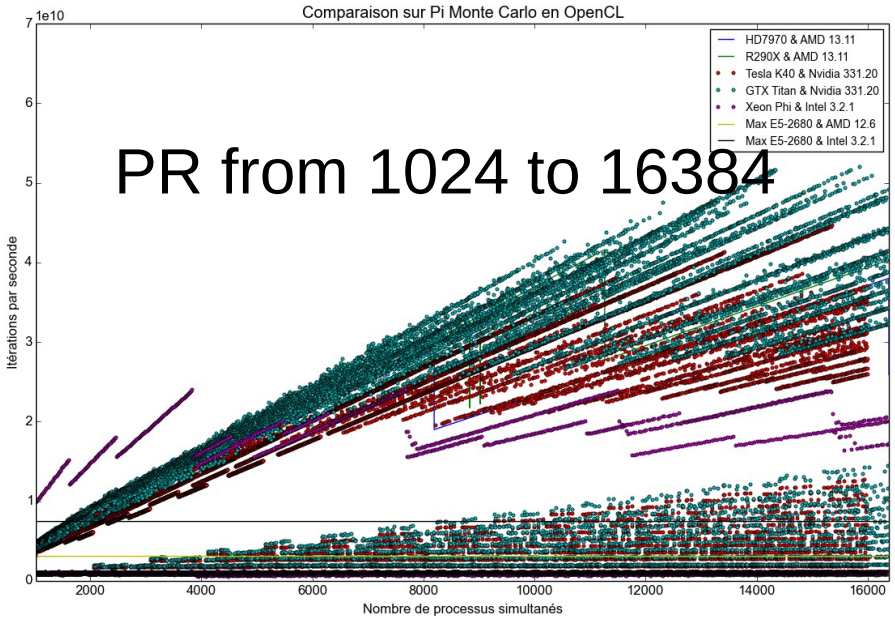
Small comparison between * PU
Low pararegime: the reign of the CPU

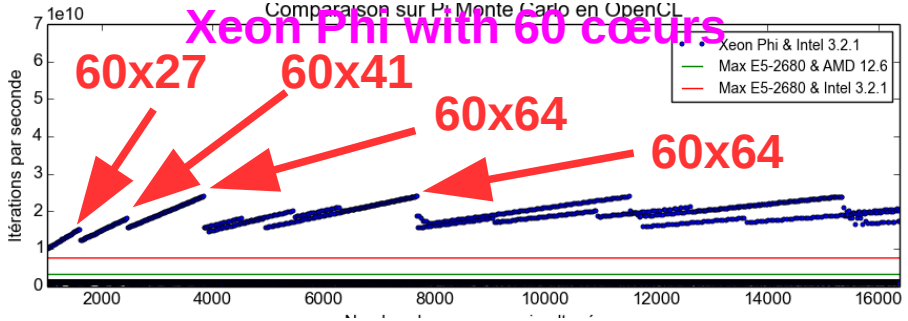


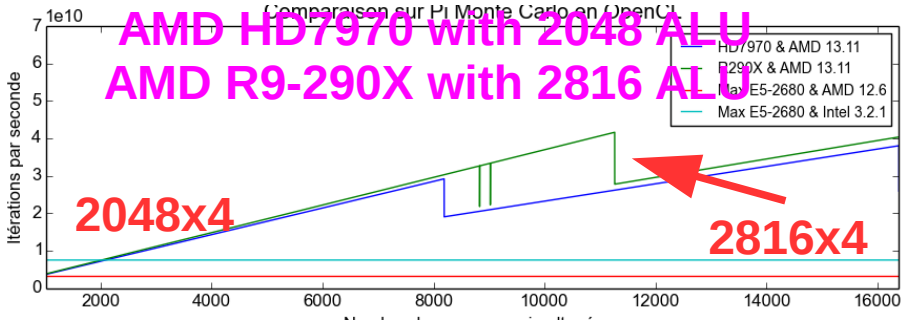
Small comparison between *PU
MIC Phi Out of Wood, GPU

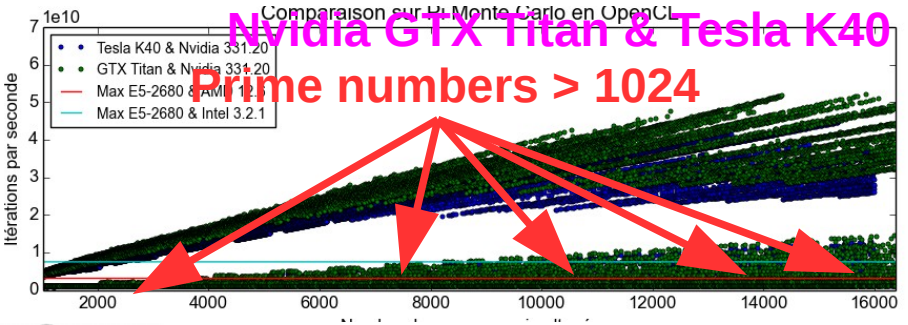


Deep Exploration of PR ParaRégime from 1024 to 16384

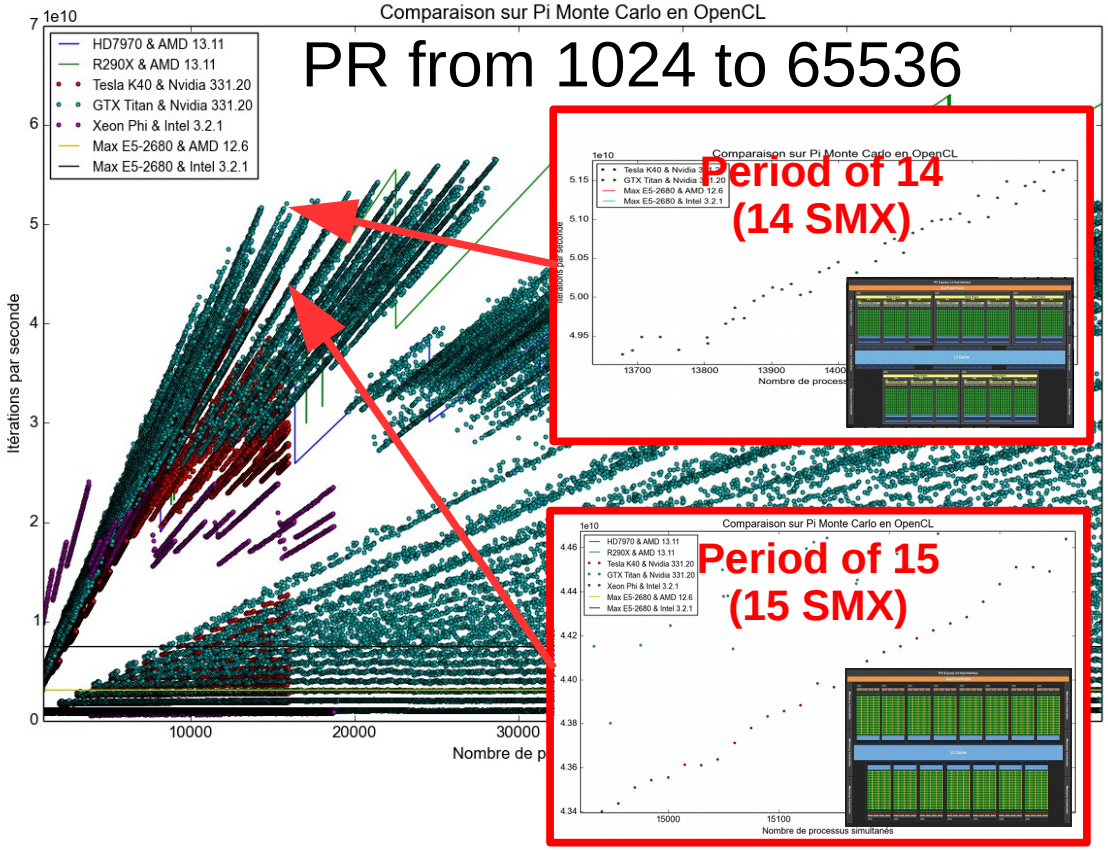




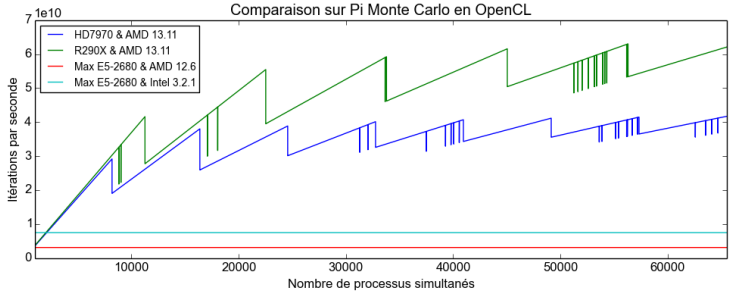




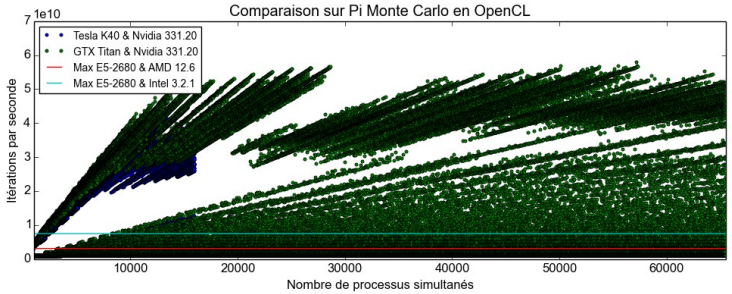
Very deep exploration of PRs
Regime of //ism from 1024 to 65536



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



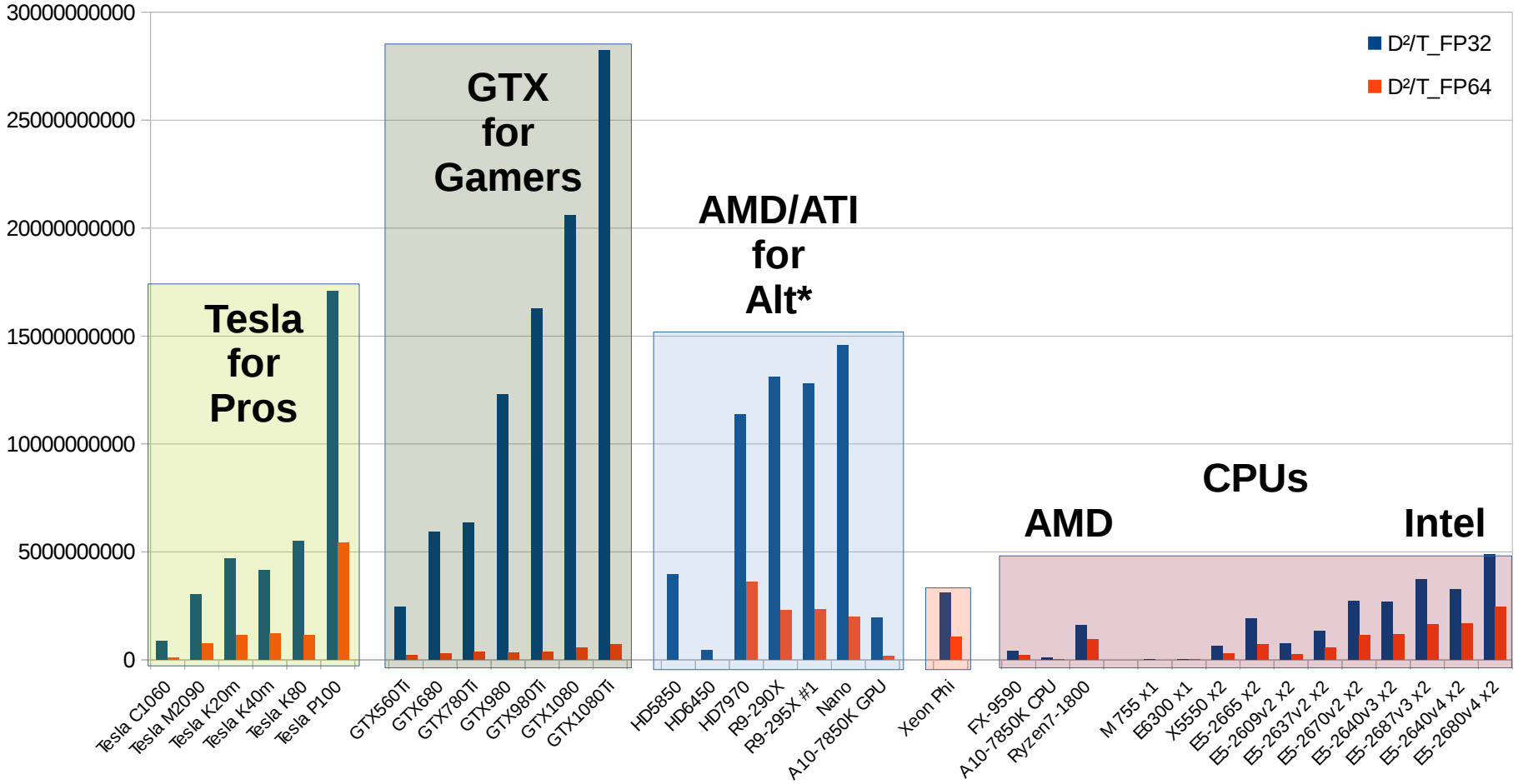
Nvidia GTX Titan & Tesla K40
Small Period : number of SMX units



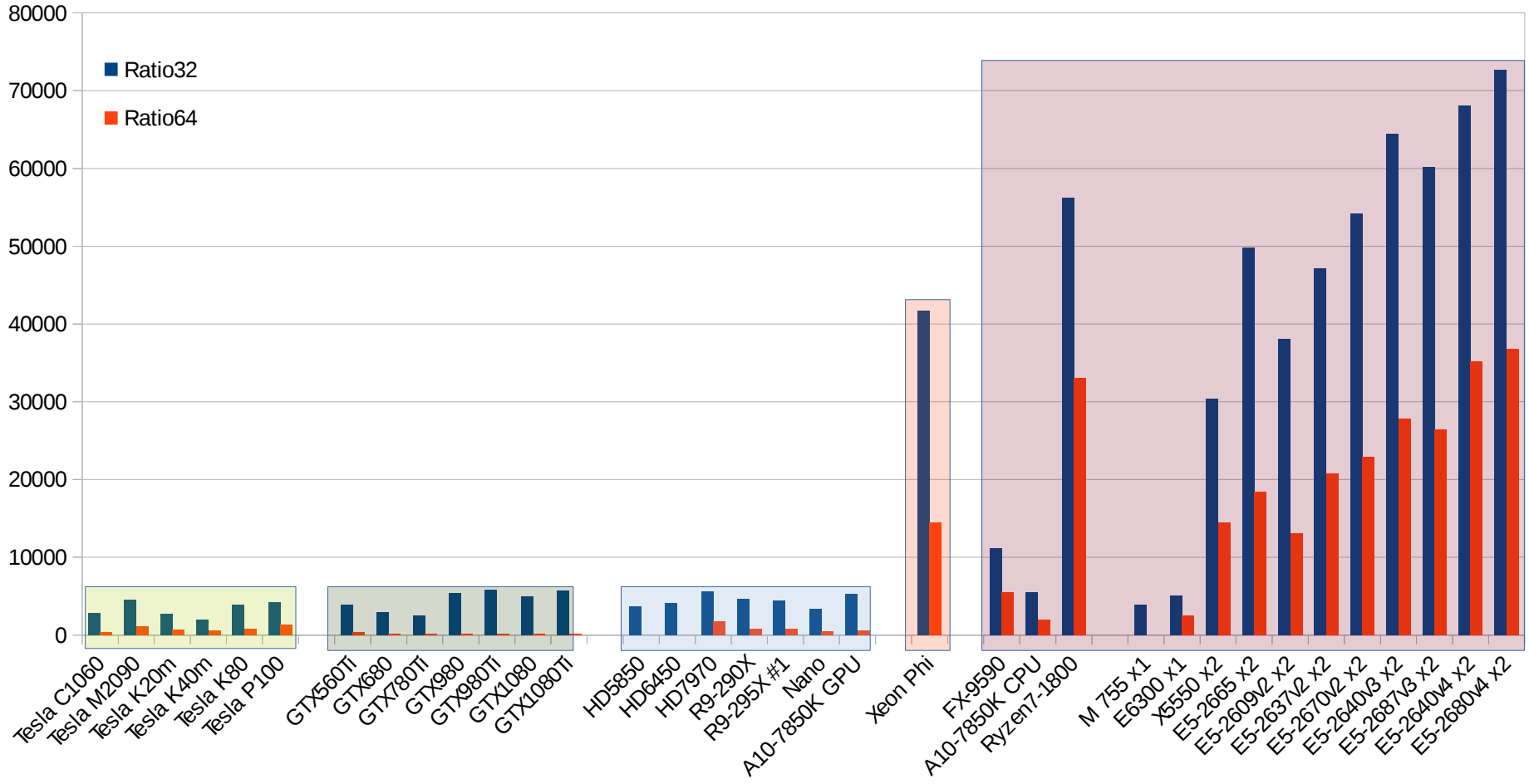
"Back to the Physics"
A Newtonian N-body code ...

- 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

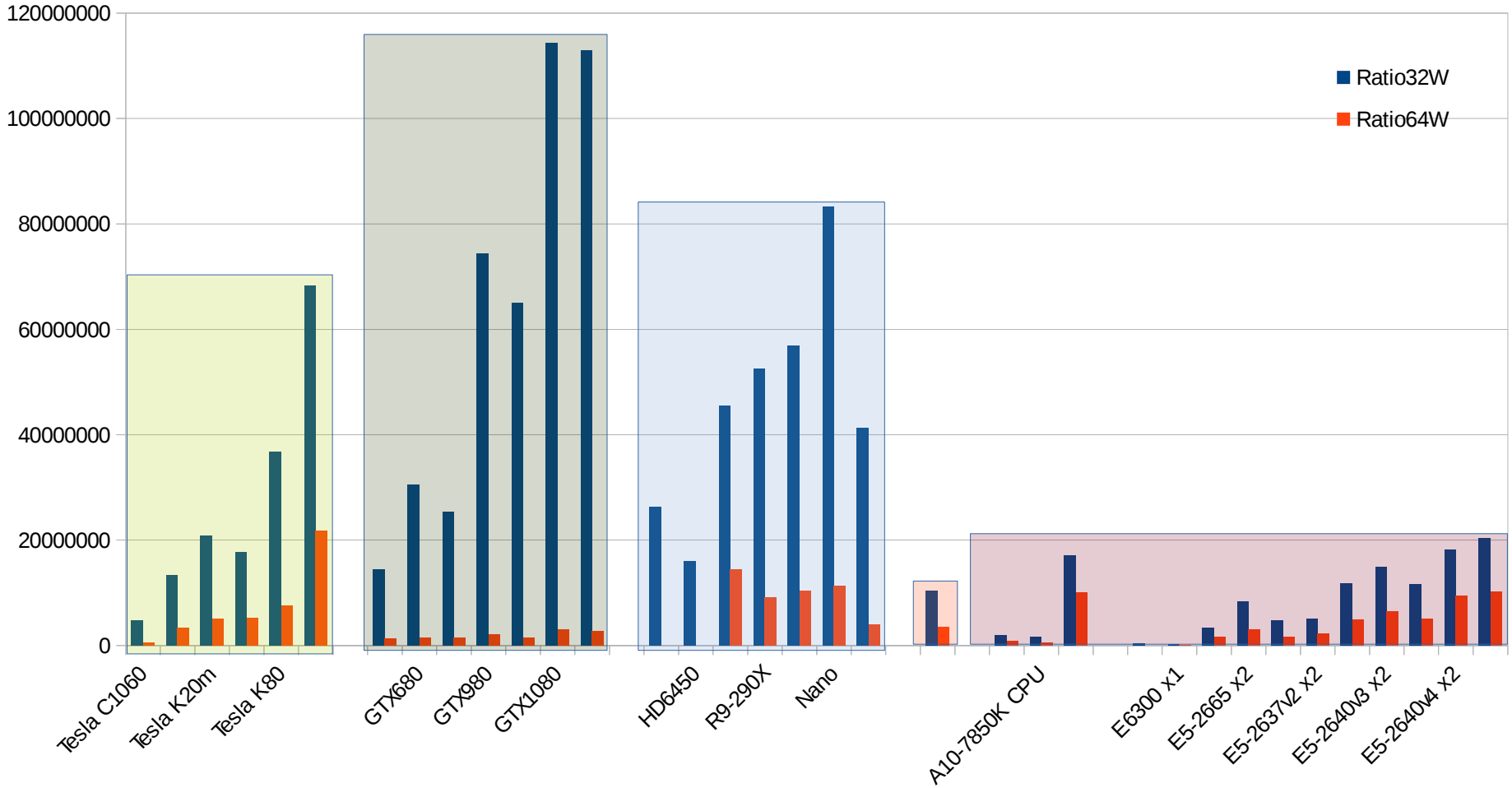
Very different performances Nvidia, AMD, Intel, Single/Double



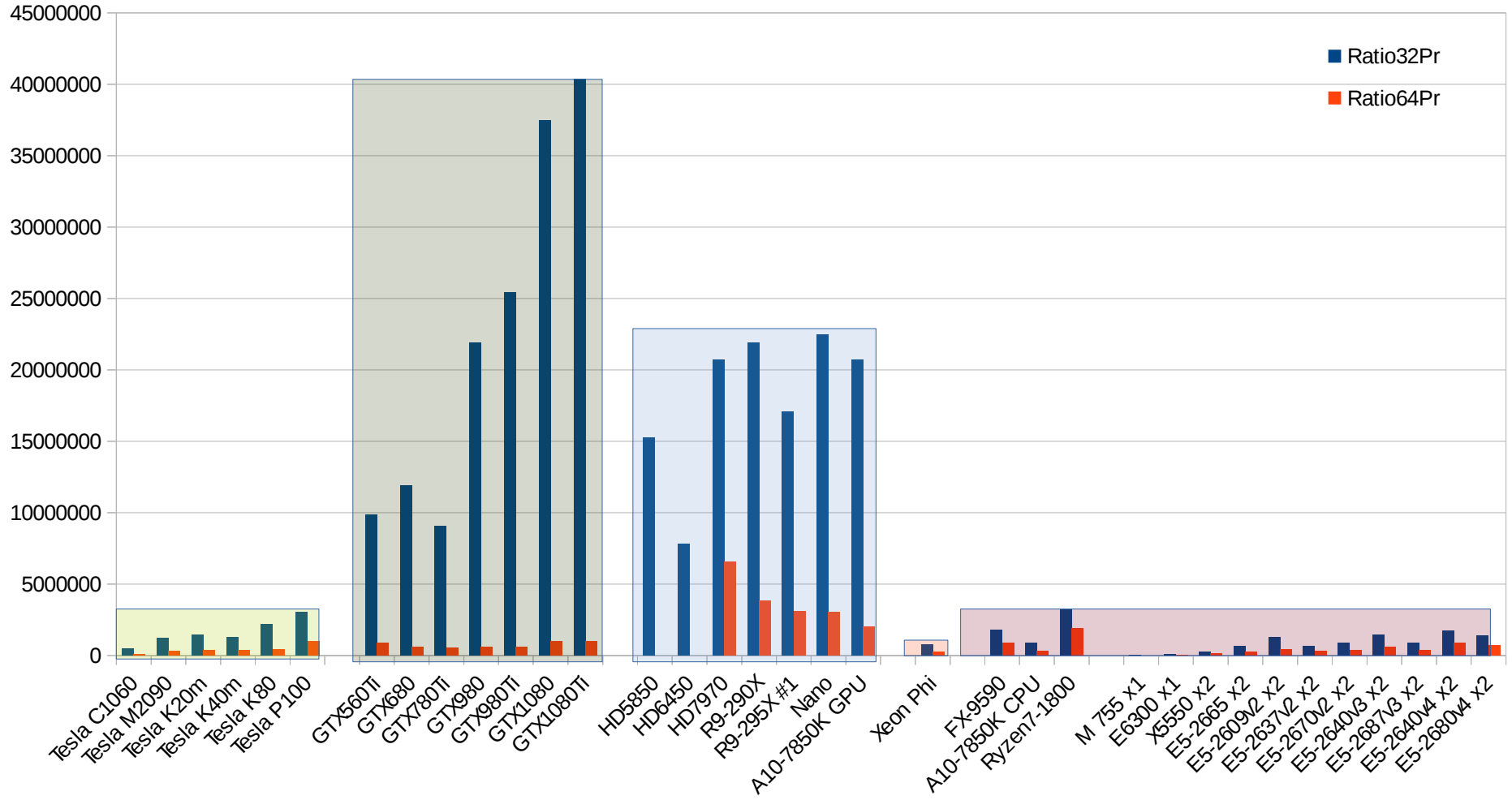
Performances normalized to cores Nvidia, AMD, Intel, Single / Double



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!