



Houches 2016 : 6th School

Computational Physics

Travels, tries, traces, traps, tricks, trends and trolls...

Event log of a native physicist inside the world of parallelism

how the heat frontier led disruptive technologies...

Emmanuel Quémener

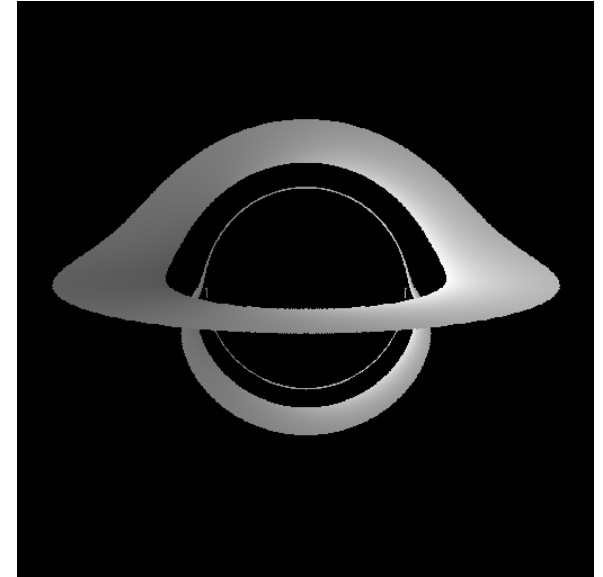
Warnings about your lecturer...

- I'm french
 - And all TV series are translated in France (so, no improving english via TV :-())
- I'm a « production » of french university 25 years ago
 - And english learning & speaking was not clearly a priority
- I'm not graduate in computers
 - But I use computers since 1984 and Debian Linux for 20 years
- I'm a physicist
 - And I worked on gravitational lenses and their application to Lambda in 1994
- I'm research engineer
 - But I improve my knowledge on all IT domains since 20 years...
- The most important thing I learn this 25 years :
 - « If you can not prove that the work is done , it is not worth undertaking it ! »

My (chaotic) adventure in 1 slide

From astrophysics to computing

- 1993 : Master 1 in astrophysics (Toulouse)
 - Code simulating of Gravitational Lenses
- 1994 : Master 2 in astrophysics (Paris-Meudon)
 - « Reloaded » of JPLuminet Black Hole image
 - Use of Gravitational Lenses to constrain Cosmological Constant
- 1995-1999 : PhD in optical processing (ENST-Bretagne)
 - Lots of Modelisation, Simulations of optical benches
 - System Administration of my laboratory, Debian user since june 1996
- 1999-2005 : System & Network Engineer (ENS-Cachan)
- 2006 : Project engineer on JWST Nirspec (CRAL, Lyon)
- 2007 : Research engineer in LIP (Computing & Parallelism Laboratory, ENS-Lyon)
 - Gridification of applications as RAMSES in LEGO project
- 2007-2009 : IT supervisor of ENS-Lyon
- 2009- : Research Engineer & IT Test pilot (Blaise Pascal Center, ENS-Lyon)



Centre Blaise Pascal :

Experimental platform with 10 technical facilities

- Multi-nodes : 5 clusters from 4 to 64 nodes, Nodes/Cores : 4/24, 8/64, 8/64, 64/512
- Multi-cores : 10 from 2 to 20 cores,
 - Nodes/Servers : from 8 to 20 cores, Workstations : from 2 to 16 cores
- GPU & Accelerator : 36 different models of GPU (AMD & Nvidia), 1 Intel MIC
 - GPGPU : 8 ; GPU Nvidia : 18 ; GPU AMD/ATI : 10 types ; Xeon Phi 7120P
- Integration : 14 virtual machines : Debian from Lenny to Sid in 32 & 64 bits, ...
- Exotic hardware : 3 machines ARMv7 under Debian Jessie or Ubuntu
- 3D facilities : 2 workstations, 2 video projectors, 20 monitors, 4 glasses
- Remote desktop facilities : more than 25 hosts with x2go/VirtualGL
- COMOD with SIDUS : « Compute On My Own Device » for laboratories users
 - SIDUS is « Single Instance Distributing Universal System »
- Galaxy project demonstrator for data intensive biomedical research

Blaise Pascal center as Dryden FR

A small but illustrative example



Nasa X-29

- Cell of F-5
- Engine of F-18
- Gear of F-16
- Studies
 - Fwd swept wing
 - Incidence $>50^\circ$
 - « Fly-By-Wire »

Recycle, Re-use and explore new domains

Warnings about this course

What it will **not** be ...

- A introduction to general parallel computing
 - https://computing.llnl.gov/tutorials/parallel_comp/
- An introduction to parallel languages
 - MPI : <https://computing.llnl.gov/tutorials/mpi/>
 - Posix Threads : <https://computing.llnl.gov/tutorials/pthreads/>
 - OpenMP : <https://computing.llnl.gov/tutorials/openMP/>
 - **That's where I learn alone how to...**
- And I'd like to provide you how I would like to learn it.

Parallelism in 7 questions

5 Ws & 2 Hs

- Analytical Method : answer 7 questions
- Why ? What ? When ? Where ? Who ?
- How much ? How ?
- In french, CQQCOQP !
- Interesting approach not to forget :
- Problem : intrication between questions
- Advantage : really separate ambiguities
- Try to answer without to many overlappings !

Why parallelism : //ism is a way...

Where we go ? Where we are ?

- Where we are : we all oftently use codes
- Where we go : we want more « performance »
- How to go : parallelism is one way, but why ?
- Before « falling through the rabbit hole » of parallelism :
 - What are the practices on codes ?
 - How to define a performance of a code ?
 - What selected criterium of performance to choose ?
 - How to reach the selected performance ?

Codes & Performance :

Which definitions to choose...

- Etymology (Etymonline)
 - Code : from latin codex « book, book of laws »
 - « systematic compilation of laws » (1236)
 - « system of telegraphic communication » (1866)
 - Performance :
 - « accomplishment » (of something)
 - meaning « a thing performed » is from 1590s
 - « set of optimal capabilities for a system » (1929)
- And we will choose
 - Code : both :-)
 - Performance : three :-)

If computing was cooking...

Code : only the recipe...

Code ~ Recipie

Computer ~ Kitchen

Input Data ~ Ingredients

Output Data ~ Meal Dish

Process ~ Cooking process

Control Unit ~ Cooker

ALU ~ Utensil

Me ~ Client

Batch Request ~ Order



Some definitions and letters...

- ALU : Arithmetic & Logic Unit
- CPU : Central Processing Unit
- Flops : Floating Point Operations Per Second
- (GP)GPU : (General Purpose) Graphical Processing Unit
- MPI : Message Passing Interface (communication between nodes)
- RAM : Random Access Memory
- SMP : Shared Memory Processors
- TDP : Thermal Design Power
- And several new ones :
 - PR : Parallel Rate (NP in MPI, Threads in OpenMP, Blocks, WorkItems in GPU)
 - Itops : Iterative Operations Per Second
 - EPU : Equivalent Processing Unit (optimal parallel rate deduced)

What's this ?

Code, protocol of experimentation

- In cuisine :
 - We have all the ingredients, we want to make a dish !
- In scientific ways :
 - Simulation : « On Its Theory (Discrete ?) Service »
 - Processing : for « demanding » experimentaters
 - Visualisation : to see to perceive things (and share)
- Each launch is an experience (and unique one)...
 - Recipies : « codes » becaming « processus »
 - Utensils : librairies, OS, hardware, networks, ...
 - Ingredients : modelisation, data, ...
 - Execution : and the experience cannot be restricted to Results

Families of Codes

- What distinguish the different codes I use ?
 - « My code I did of mine and I'm proud of »
 - My supervisor code
 - In fact, the stratification of codes produced by previous generations of students
 - Code «business»
 - « Ikea » model : delivered with assembly instructions (without toolbox)
 - « Crozatier » model : (almost) ready to use
- Like in every family, problems occur for inheritance
- Dependencies to :
 - Generic librairies : BLAS, Lapack, FFTw
 - Proprietary librairies : Mathworks, Intel, Nvidia, AMD, ...
 - Hardware !

Performance : how ?

A question of observables !



Sport performance

- To run a 100-metre ?
- To run a marathon ?
- To make shot put ?
- To complete an heptathlon ?



Performance : how ?

A question of objectives !

- To put all luggages & family inside the car
- To draw the attention of females outside the night clubs
- To get from point A to point B in a town with traffic jam
- To climb to Pikes Peak



Performance :

Conditioned by objectives

- Speed : elapsed time (only?)
- Work : immobilization of resources
- Efficiency : best use of available resources
- Scalability : incremental progress when more resources are dedicated
- Portability : diffusion to other IT infrastructure
- Maintainability : time spent to maintain the system operational
- General approach :
 - Define un criterium
 - Research extreme values (maximum or minimum) for a pertinent test suite

Speed as Performance Criterion

« Speed, I'm Speed... »

- All time, but not only « Elapsed time »
- To use code : the 3 costs
 - Entry cost : to learn software, to integrate in infrastructure, ...
 - Operational cost : to maintain, to operate
 - Exit cost : substitution by an equivalent code, an equivalent technology (Cell...)
- Optimization (and its problem) : $DD/DE > 1$ is pertinent ?
 - DE : Total elapsed time for my code
 - DD : time spent to minimize this total elapsed time
- To estimate the value :
 - System tools, metrology tools in langages, codes, ...
 - « Et après moi ? Le déluge ? » : what future for the code ?



Work as Performance Criterion

- Work : « Time is money »
 - Ressources : CPU, RAM, GPU, storage, network, ...
 - In fact, a Matriochka :
 - CPU : several cores, CU, ALU, piles, ...
 - RAM/SRAM : 4 levels
 - Storages : local, slow & shared (NFS), fast & shared (GlusterFS, Lustre, ...)
 - Networks : slow (Gigabit), fast & low latency (InfiniBand)
- Job : reservation (& immobilization) of resources
 - Classical : Nodes * Elapsed time
- For a code, « system fingerprint »
 - Profiling tools, System tools

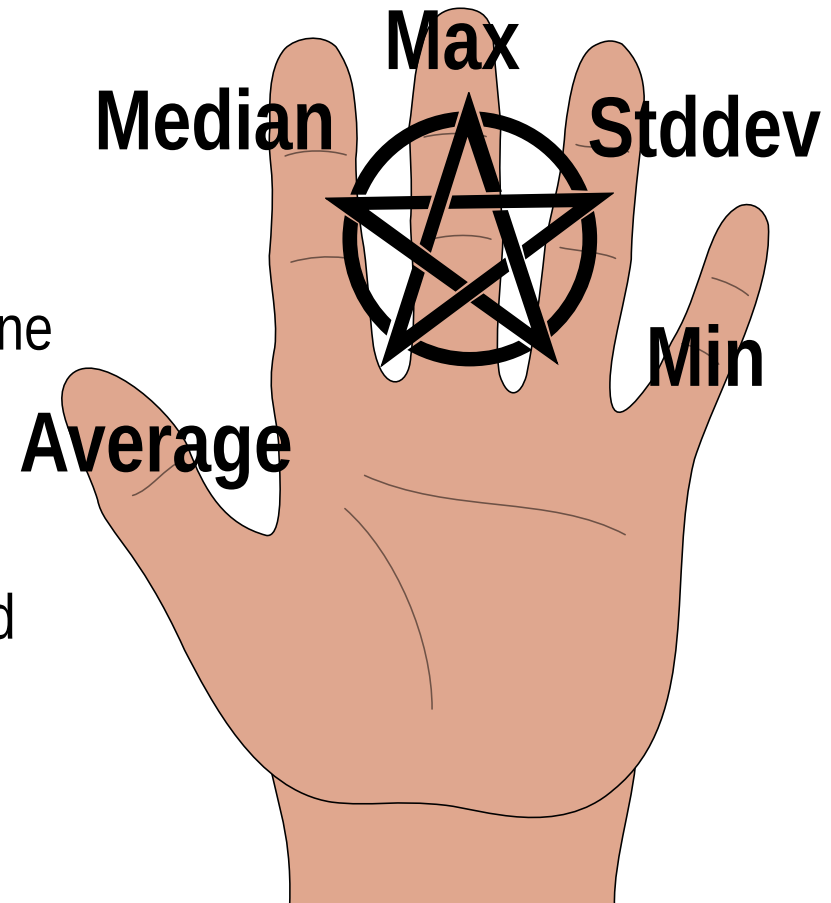
Scalability as Criterion

- Scaling :
 - In the tasks to be done : Elapsed time ? $f(\text{Elapsed Time})$
 - In required resources : $g(\text{System Resources})$
- Reefs to avoid :
 - Scaling effects (in fact, threshold effects are even worse)
 - Needing conductor ? From a Quatuor symphony orchestra...
 - Although you execute, the available resources are limited...
 - You think I'm joking :-/ ?
 - **Parallelization becomes unescapable, but why ?**

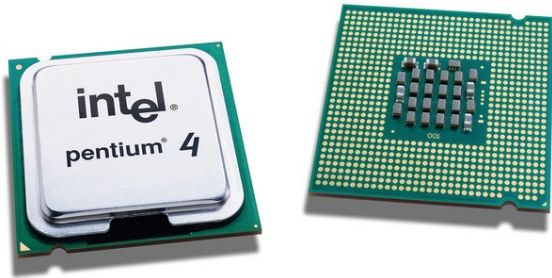
Measure of Performance & Scalability

Pen(s)tacle of statistics:

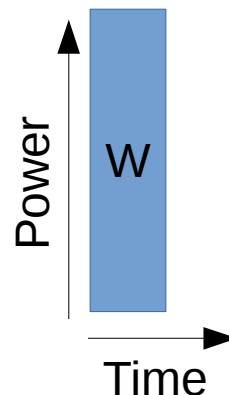
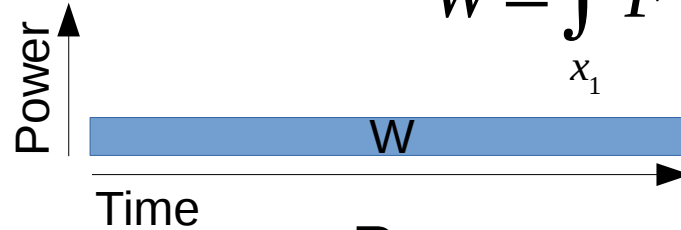
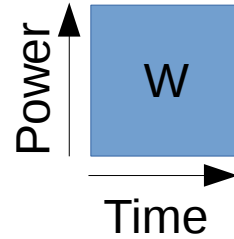
- Why improving statistics ?
 - Because you practice sciences !
- The pentacle of statistics
 - Average : the first we think, but bad one
 - Initialisation process, random tasks
 - Median : the one to prefer
 - Max : The slowest is the most awaited
 - Stddev : indicator of variability
 - Min : the best case is to know



Work on Computing Resources from a Physicist Point of View



Thermodynamics



Mechanics

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

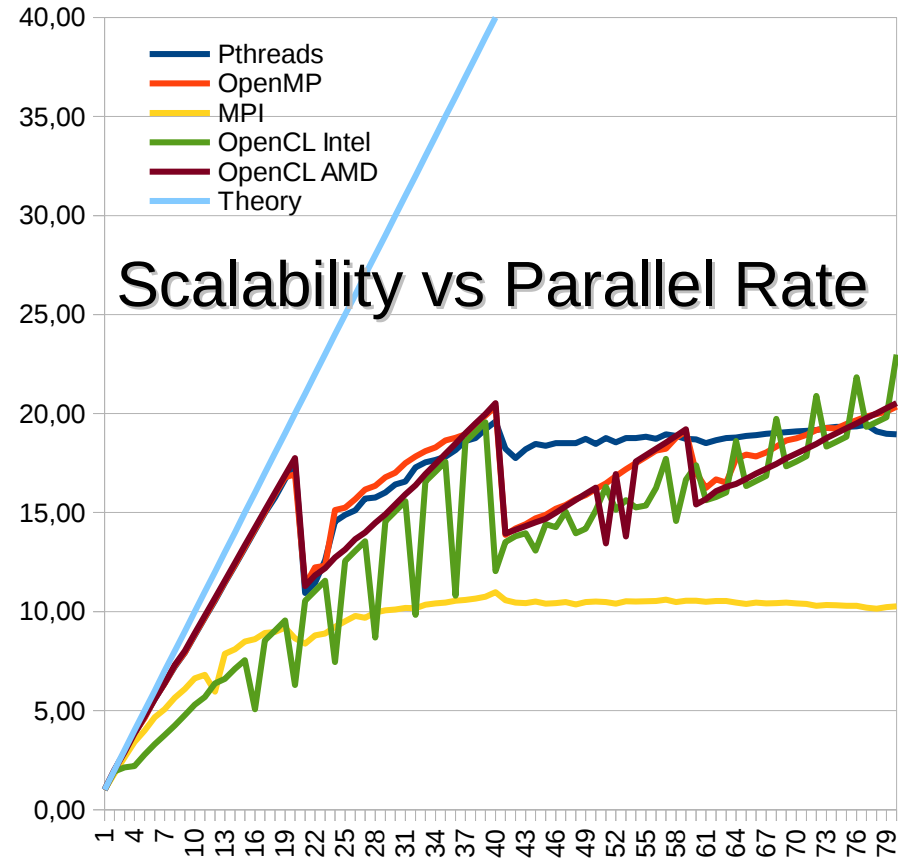
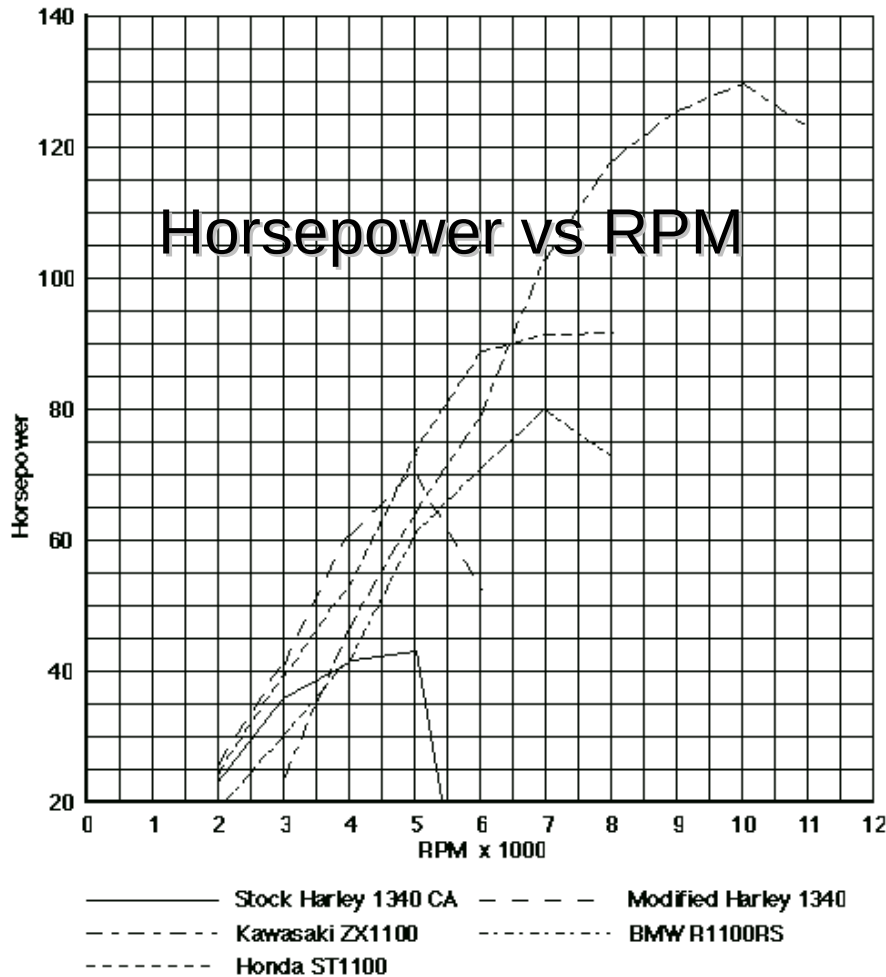
Power equals product :

- Frequency
- Number of Workers
- Power of 1 Worker

Work on Computing Resources

An Engine as the source of Power

Chart 5: Horsepower curves for 5 different motorcycles



Why Parallelism (is inevitable) ?

And its constraint is TDP

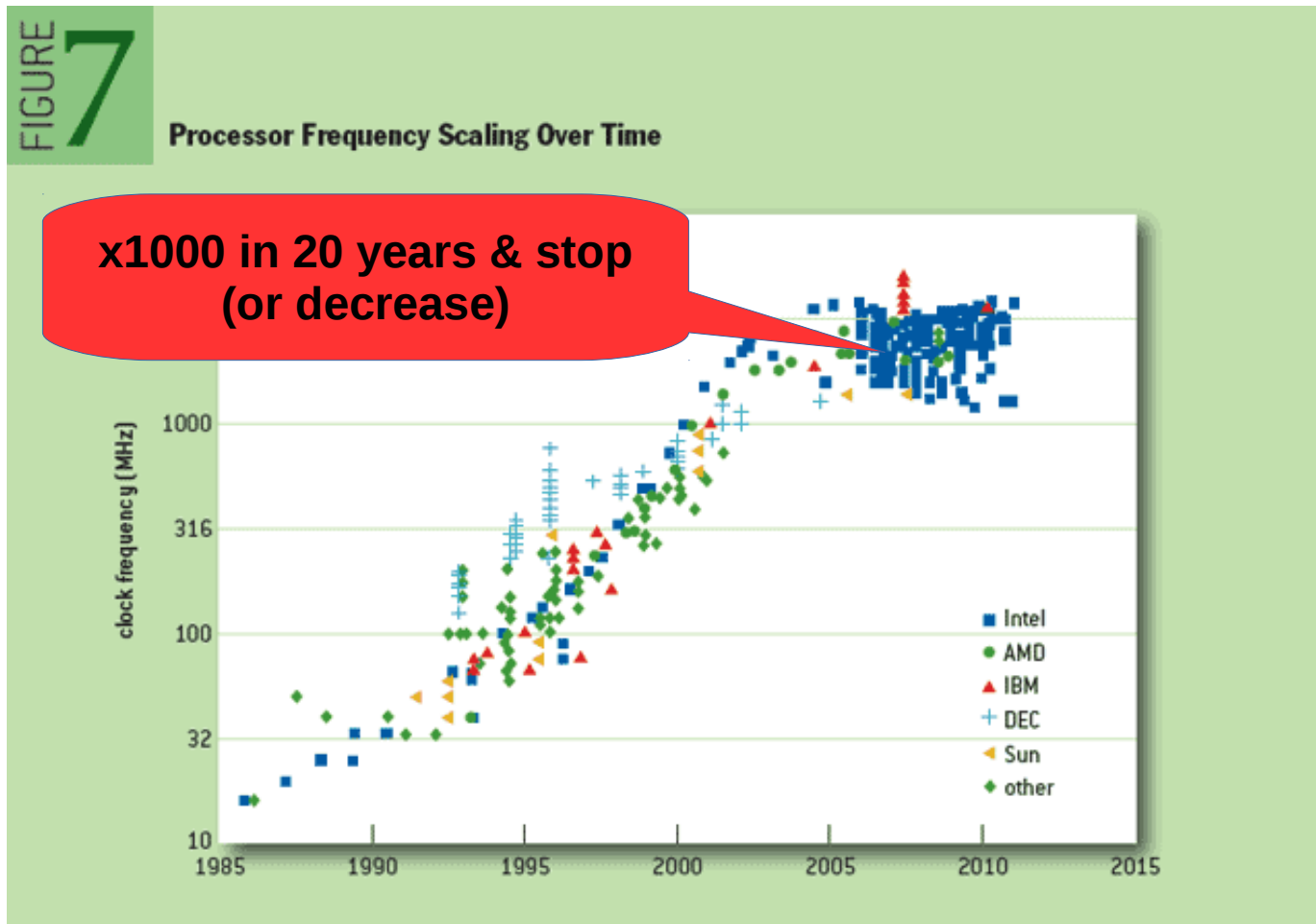
- Rise and Fall of Frequency
 - Between 1989 and 1999 : from 4 MHz to 400 MHz x100 in 10 years
 - Between 1999 and 2004 : from 400 MHz to 3 GHz x~10 in 5 years
 - Between 2004 and 2009 : from 3 GHz to 2 GHz
- Thermal Design Power : limited power of socket no to overshoot...
- $TDP = \frac{1}{2} C V^2 f$
 - C = Capacitance, f = frequency, V = voltage
- TDP for a processor : 150 W (on 4 cm²)
 - Density of heat of an Induction Hob
- TDP becomes the blocking factor of a processor

Capacitance = $Schrink^2 \cdot Nb \text{ Transistors} \cdot Mylq \text{ Constant} (\sim 0.015)$



Why Parallelism ?

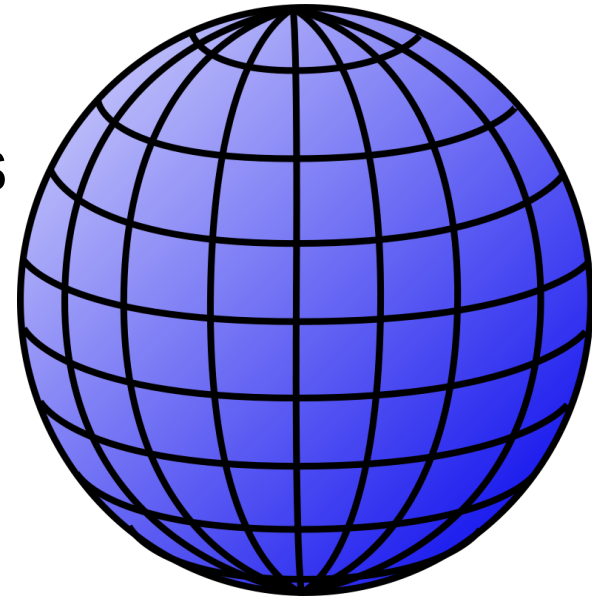
When Clock Speed ~ Velocity...



What is Parallelism ?

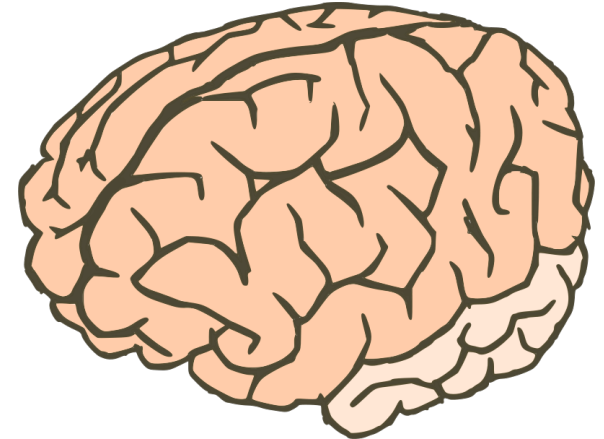
Let's « Return to the Source »

- Etymology (etymonline.com) : beside one another
 - From para- « beside »
 - From allelois « each other », from allos « other »
- Parallelism : tasks to achieve, limited ressources...
 - Execute independant tasks in parallel
- Execute one task in parallel on all resources
 - Sparse communications : Coarse grain
 - Heavy communications : Fine grain
- Paradox of parallelism, meridian !



Where is Parallelism ?

- Where is the best computer ?
 - Between your ears !
 - 20 to 200 billions neurons
 - 125 to 220 trillions synapses
 - Computational capacity (IBM) : 36 Pflops, 3.2 Pbytes
- The best GPGPU processing card :
 - 2880 ALUs
 - 12 GB and 288 GB/s of Bandwidth
 - Process capability : 4 Tflops
 - In fact, getting 1 Tflops (FP64), it's amazing !



Who uses Parallelism : Are-you like *Mr Jourdain* ?

- Mister Jourdain :
 - « Le Bourgeois Gentilhomme » by Molière
 - Speaking prose all his life without knowing it !
- You :
 - Have you got any smartphone, tablet, or laptop ?
 - Do you know how much cores in your component ?



How much is Parallelism ?

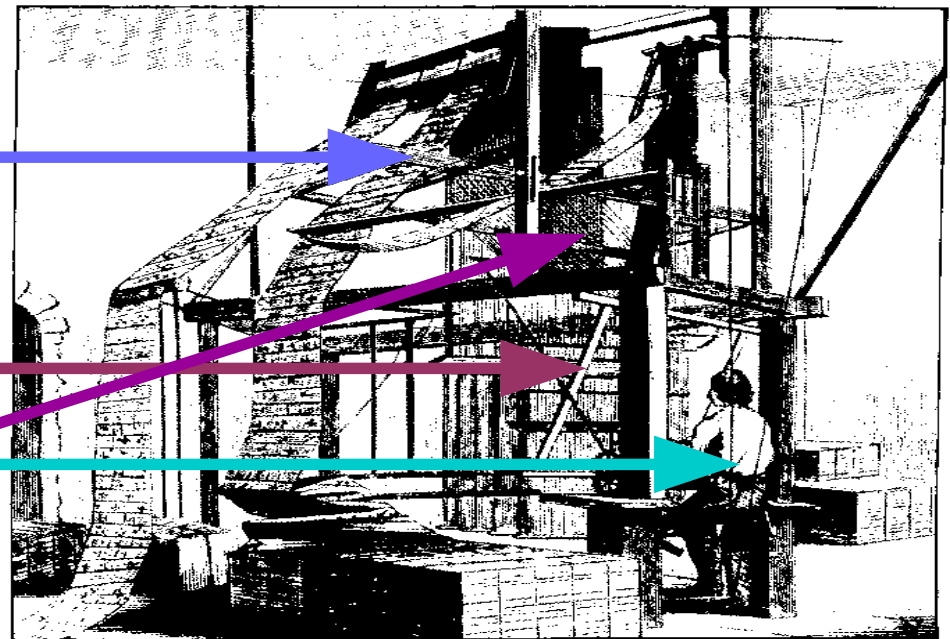
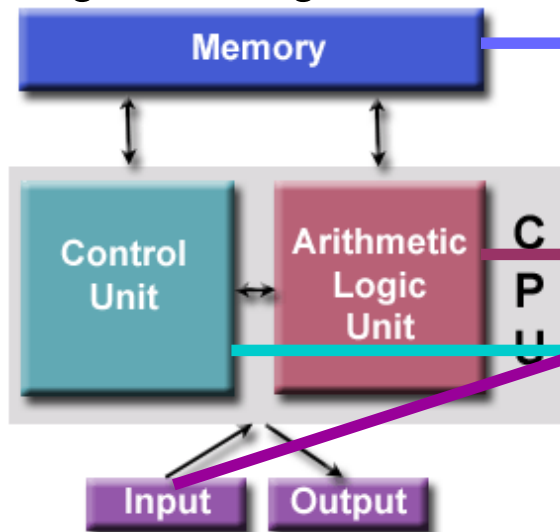
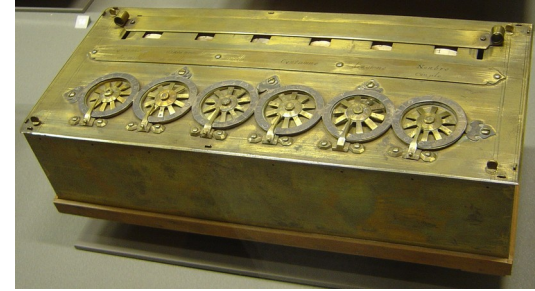
Time, Silicone, Complexity...

- The 3-Time costs :
 - Entry cost, Operating cost , Exit cost
 - Execution time compared to adaptation time
- Silicone : technologies have different prices
 - SMP (Shared Memory Processors) are expensive & limited
 - MPP (Massively Parallel Processing) need very specific networks
 - Clusters are easily extensible
- Complexity : corollary of large amount of gates
 - A GPU « core » (QPU) is simpler than a CPU core
 - A GPU « core » (QPU) is about 50 times slower than CPU core

When appeared Parallelism ? With « computing » machines !

Which is the first ?

- Analogical One ?
- Numerical One ?
- Programming One ?



How to « think » : Parallelism : « Grain... The problem is Grain. »

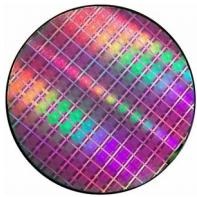
- 1 Input / 1 Process ? Optimize process !
- 1 Input / Y Process ? Optimize each process !
- X Inputs / 1 Process ? Optimize distribution !
- X Inputs / Y Process ? Optimize both !

Grain is defined by communication rate !

- Fine grain : heavy communications ($\gg 1/\text{second}$)
- Coarse grain : sparse communications ($< 1/\text{second}$)
- Embarrassing parallelism : independant tasks

How to « think » Parallelism : Flynn Taxonomy

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



Let's have a look

« Behind the Kitchen Door » (In Silicon) ?



How to Parallel Programming ?

Split/Merge between process(es)

- Pipelining fine grain, a job for silicon :

- 5 simple instructions @ a time

- Instruction Fetch
- Instruction Decode
- Execute
- (MEM)
- Write Back

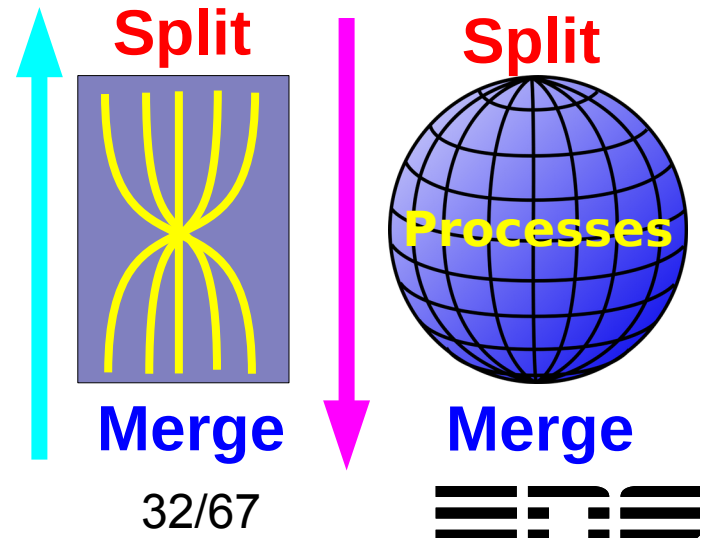
- 2 specs of RISC : 1 instruction/cycle, using registers

- 2 approaches :

- Vectorization : Merge/Process/Split
- Distribution : Split/Process/Merge

- In fact, not parallelize but meridianize

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

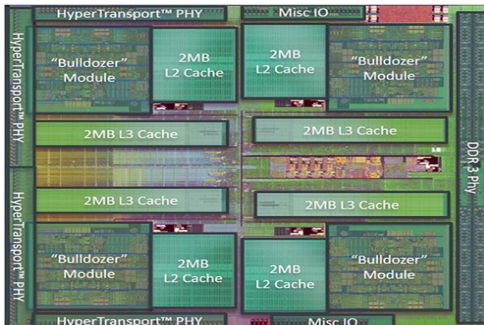
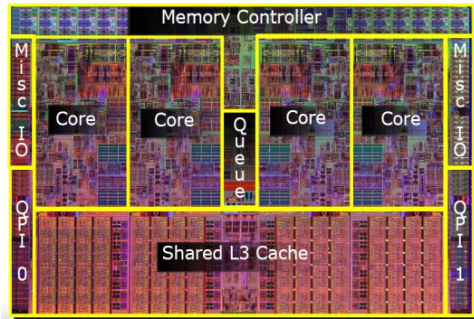
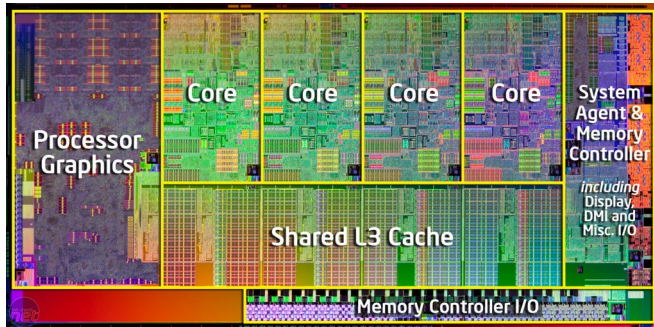


To be (a matriochka) or not to be ?

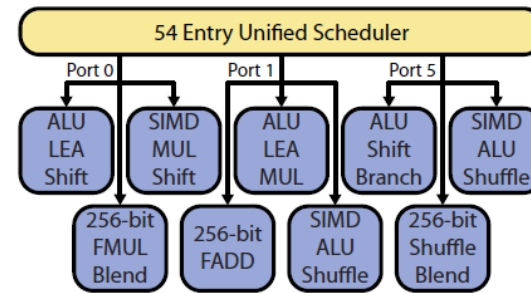
From a processing point of view



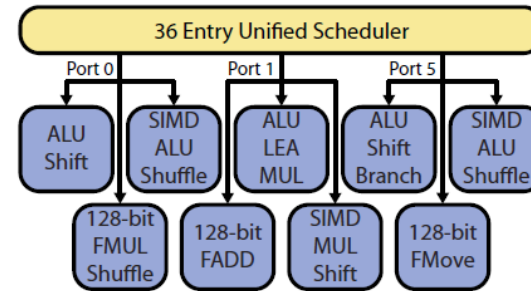
Inside a processor (on a Socket) 4-cores Processors Example



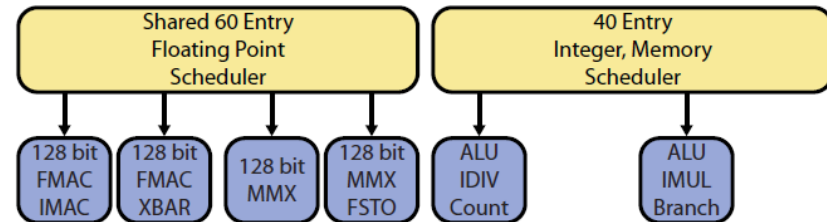
Sandy Bridge



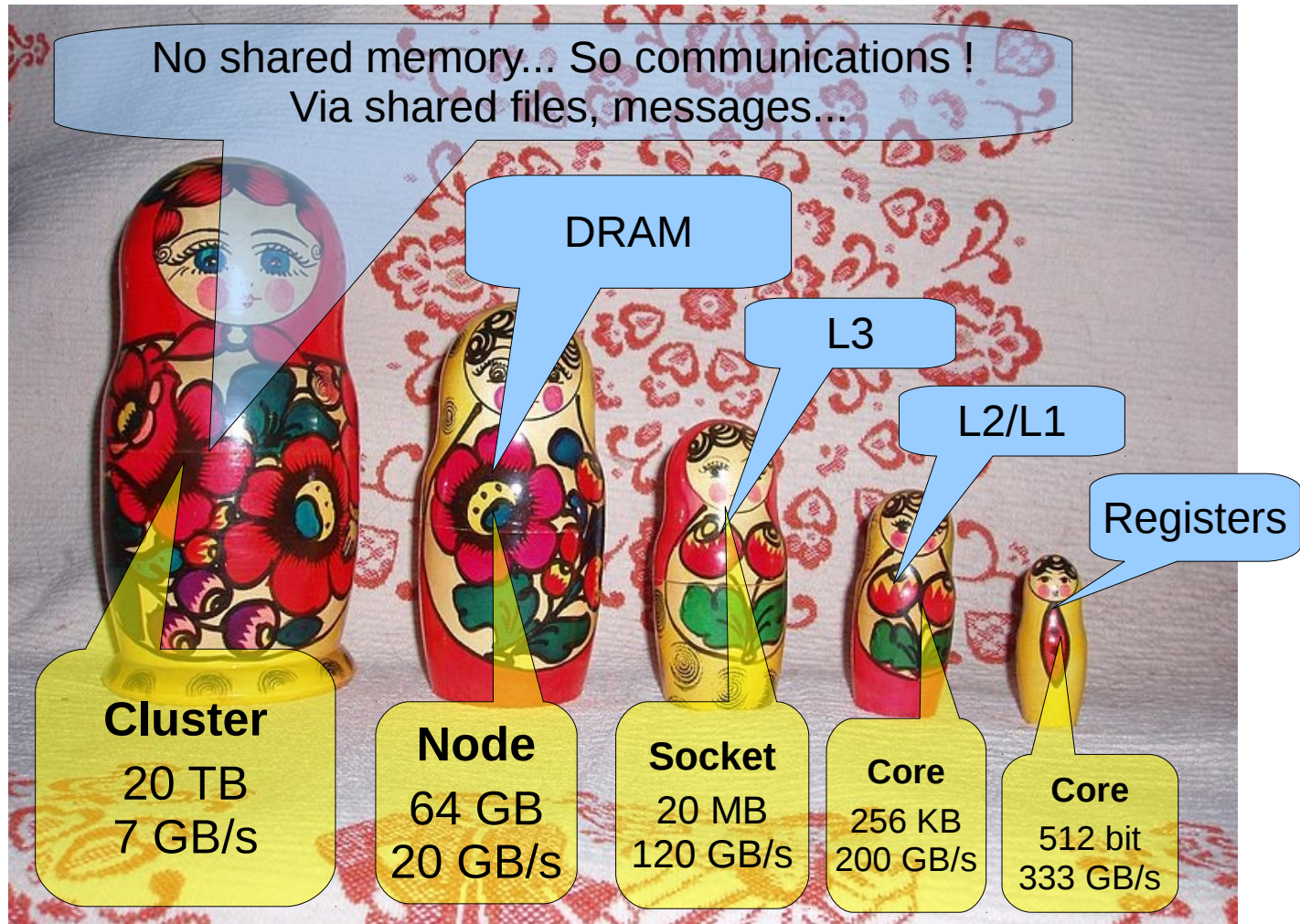
Nehalem



Bulldozer



To be (a matriochka) or not to be ? Hierarchical Memories !



If computing is cooking...

For Memory...

Code ~ Recipie

Computer ~ Kitchen

Input Data ~ Ingredients

Output Data ~ Meal Dish

Process ~ Cooking process

Control Unit ~ Cooker

ALU ~ Utensil

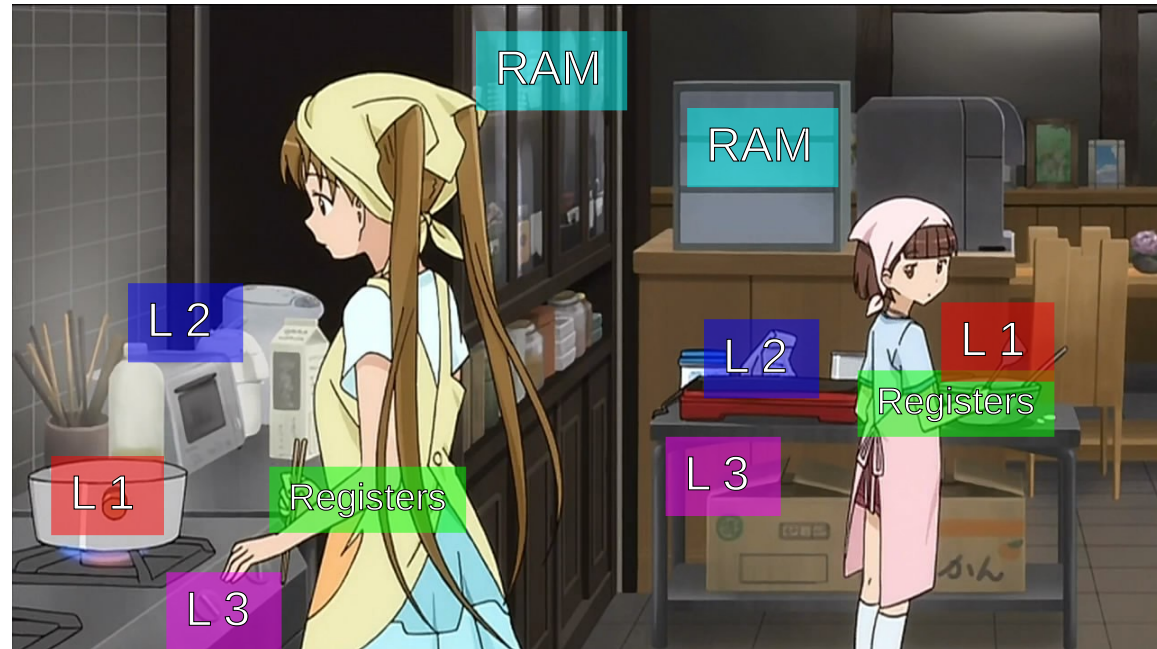
Dynamic RAM ~ Cupboards, tables, ...

L3 Cache ~ All Working planes

L2 Cache ~ Near Working plane

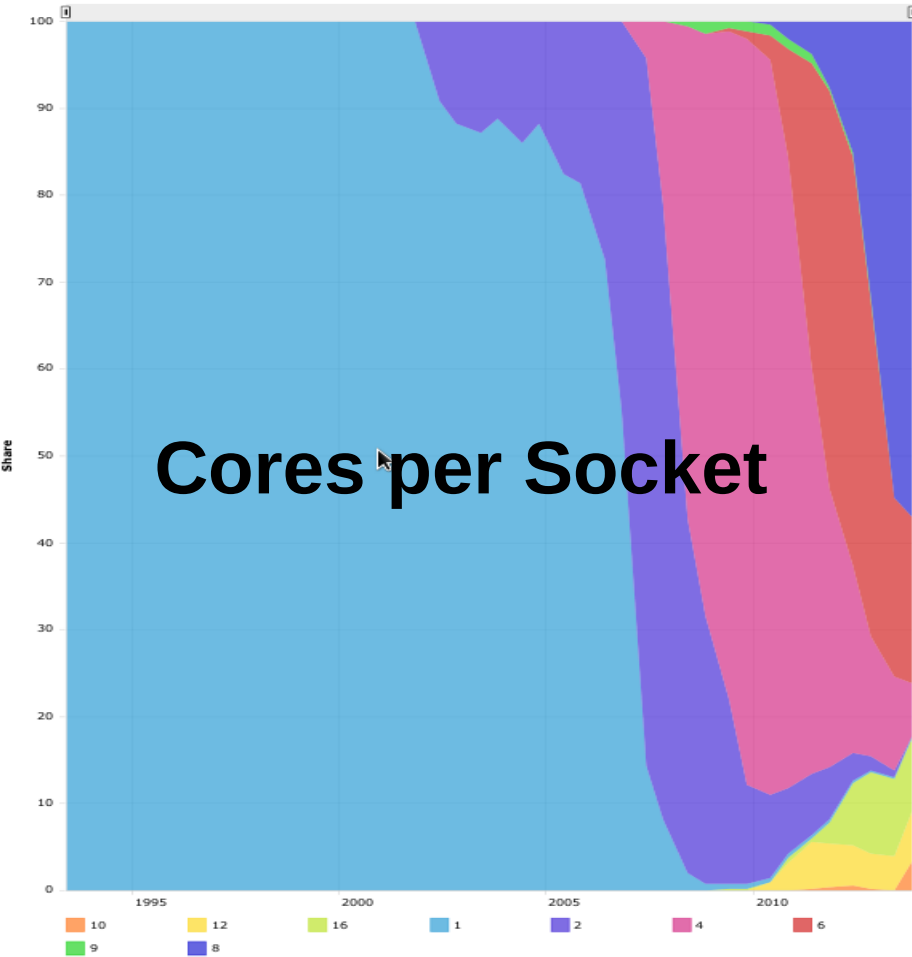
L1 Cache ~ Cutting board, container

Registers ~ Hands of Cooker

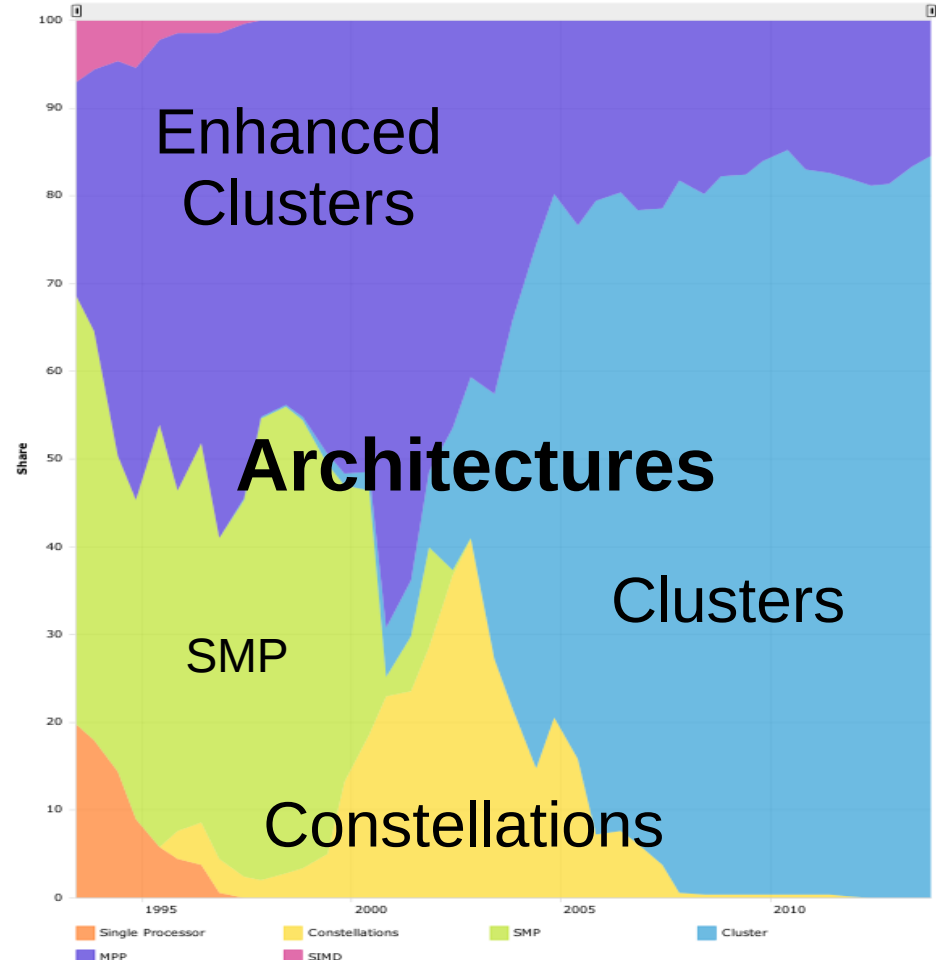


Where you are ? Where you (will) go ? Cores per Socket & Architecture...

Cores per Socket - Systems Share

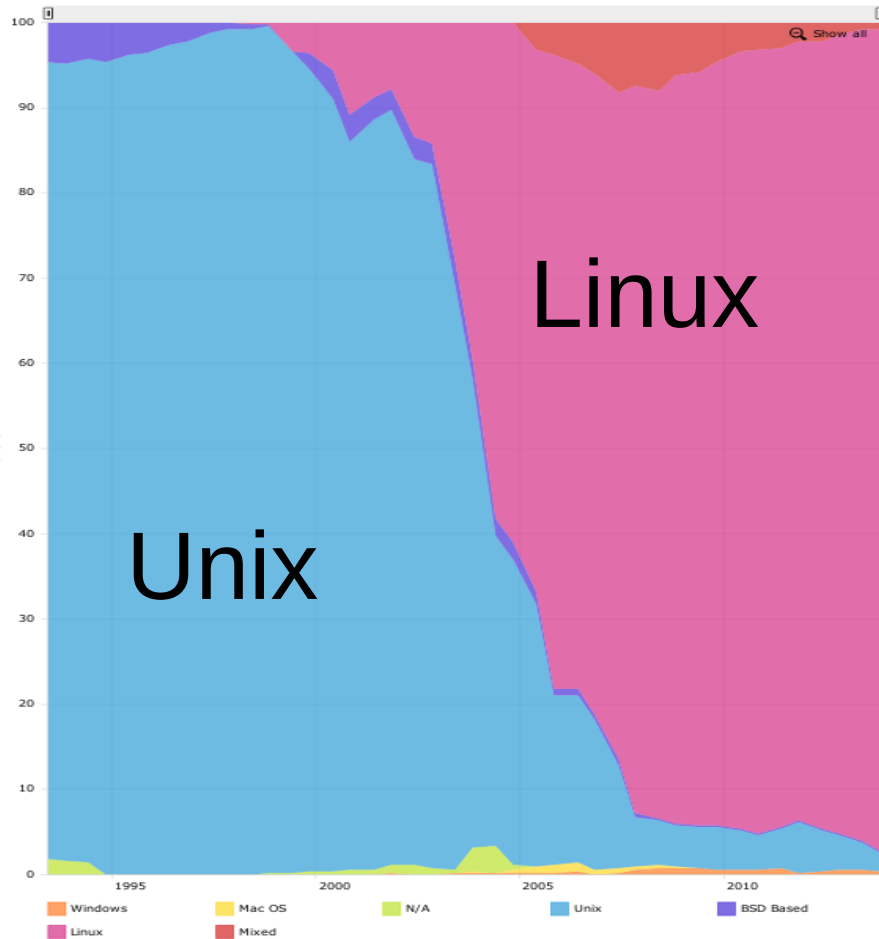


Architecture - Systems Share

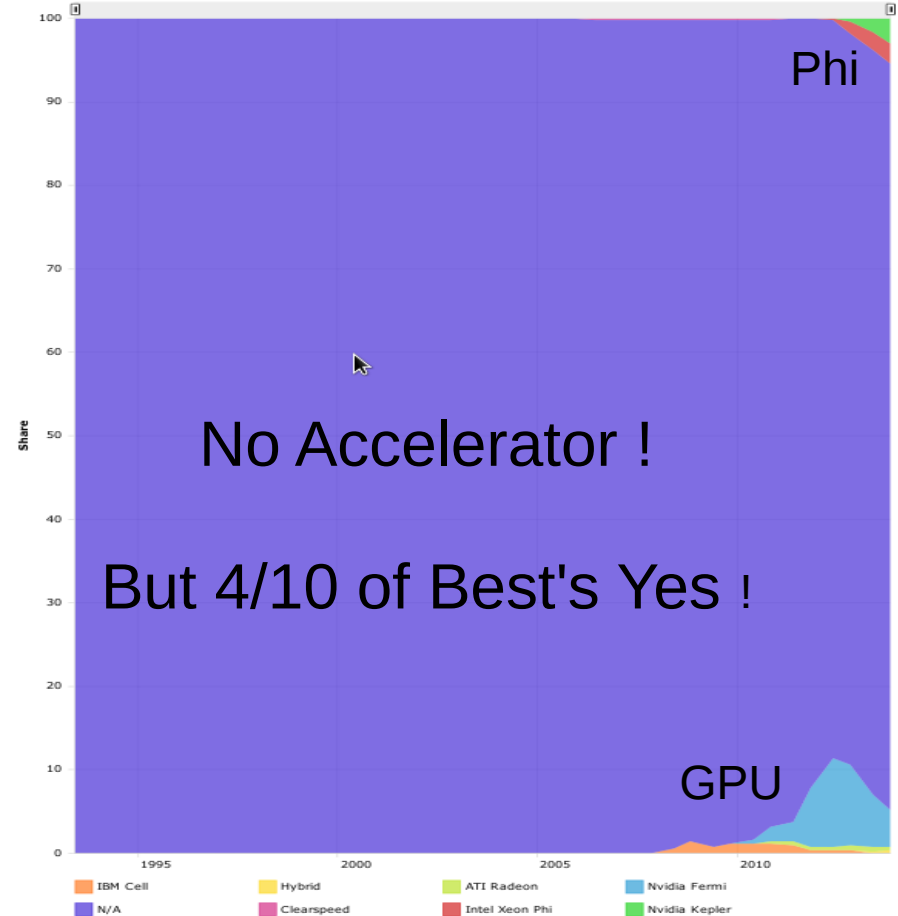


Have a quick look on OS And accelerator stuff !

Operating system Family - Systems Share



Accelerator/CP Family - Systems Share



Phi

No Accelerator !

But 4/10 of Best's Yes !

GPU

How much Parallelism ?

From multi-core to myri-ALU

- CPU, 4 in laptop, 16 in workstation, 48 in node
- From GPU to GPGPU :
 - A tiny GPU card : 128 ALU, 512 MB of RAM
 - A huge GPU card : 4096 ALU, 6 GB of RAM
- A huge GPGPU card : 2*2496 ALU, 12 GB of RAM
- Accelerator Xeon Phi : 61 CPU (Pentium like units)

How to program parallelism ?

Different approaches

Parallel Programming Models

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

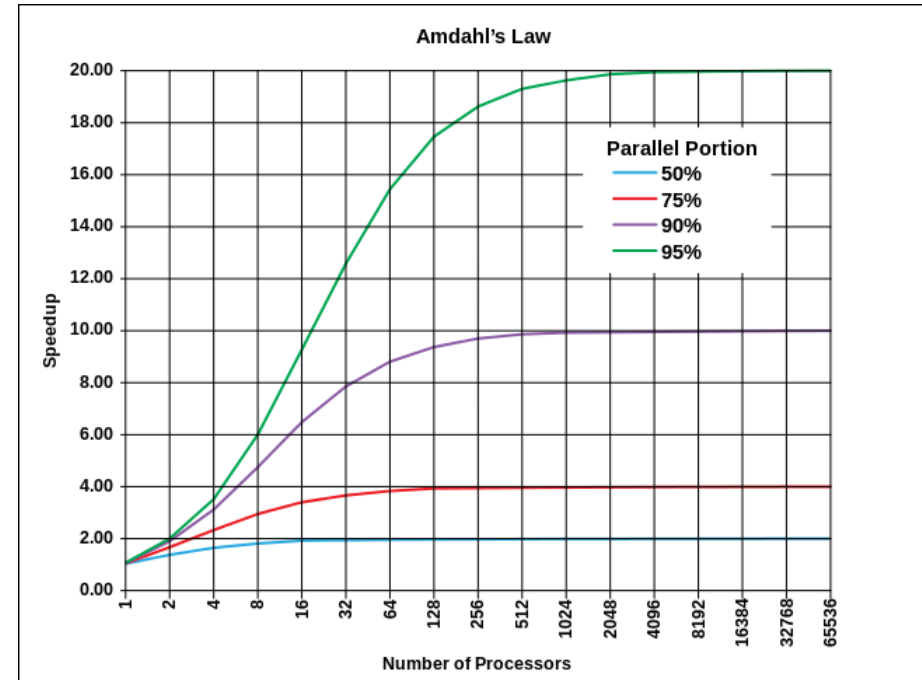
Parallel Programming Libairies

	Cluster	Node CPU	Node GPU	Node Nvidia	Accelerator
BLAS	BLACS MKL	OpenBLAS MKL	cIBLAS	CuBLAS	OpenBLAS MKL
LAPACK	Scalapack MKL	Atlas MKL	cIMAGMA	MAGMA	MagmaMIC
FFT	FFTw3	FFTw3	cIFFT	CuFFT	FFTw3

How to estimate // Efficiency ?

Amdahl Law, order (and decay)

- In the process, 2 parts
 - Sequential part, in fraction s
 - Parallel part, in fraction p
 - Elapsed Time : $T_N = T_1(s + p/N)$
 - Speedup : $1/(1 - p + p/N)$
 - Efficiency : $1/N(1 - p + p/N)$



- Speed up (& efficiency) :
 - 2 systems : $N=500$ & $N=1000$
 - 4 cases : 90 %, 99 %, 99.9 %, 99.99 %

How to estimate Parallel Efficiency ?

Amdahl Law, order (and decay)

- Speed up (& efficiency) : N=500 & N=1000

Parallel Rate	N=500		N=1000	
Parallel Part	Speedup	Efficiency	Speedup	Efficiency
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%

- Questions :
 - What's about scalability of my code ?
 - Is Amdahl law representative of « real » applications ?

Have you got your driving licencing... In computer sciences ;-)

- In an applied mathematics french book :
 - « Physicists are casual in the use of mathematics that mathematicians often equate with carelessness... »
- As a BOFH of IT resources :
 - « Scientists are casual in the use of computing resources that I often equate with my english speaking ! »
- Do you « drive » computing resources ?



```
Terminal - numa@survati -
1 [|||||100.0%] 5 [|||||100.0%] 9 [ 0.0% 0.0%] 13 [ 0.0% 0.0%]
2 [|||||100.0%] 6 [|||||100.0%] 10 [ 0.0% 0.0%] 14 [ 0.0% 0.0%]
3 [|||||100.0%] 7 [|||||100.0%] 11 [ 0.0% 0.0%] 15 [ 0.0% 0.0%]
4 [|||||100.0%] 8 [|||||100.0%] 12 [ 0.0% 0.0%] 16 [ 0.0% 0.0%]
Mem[|||||] 8301/24149MB Tasks: 47, 66 thr: 9 running
Swp[|||||] 0/9MB Load average: 8.00 7.94 7.95
Uptime: 17 days, 18:43:08

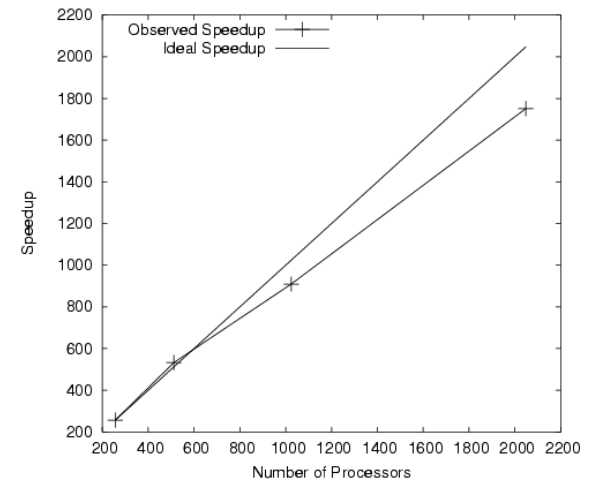
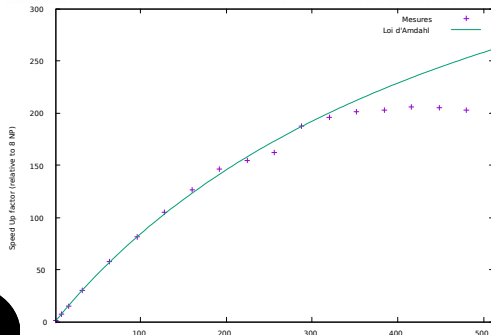
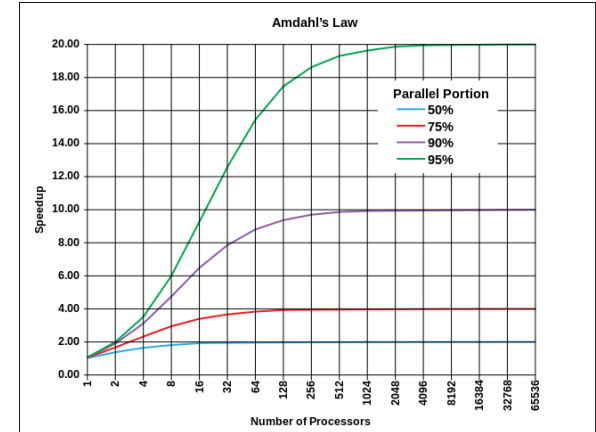
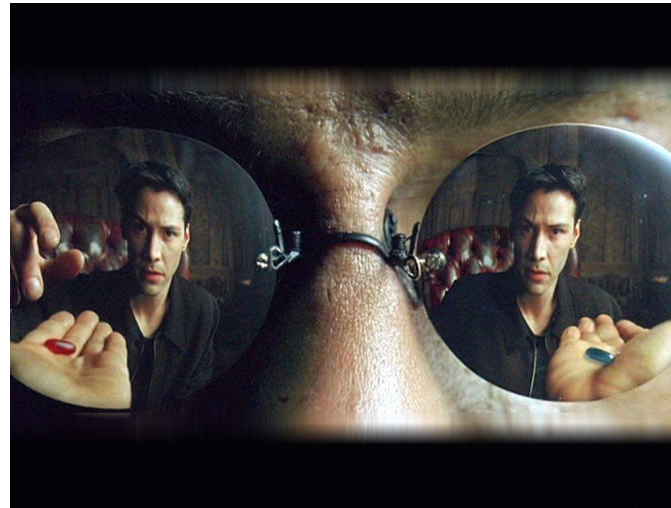
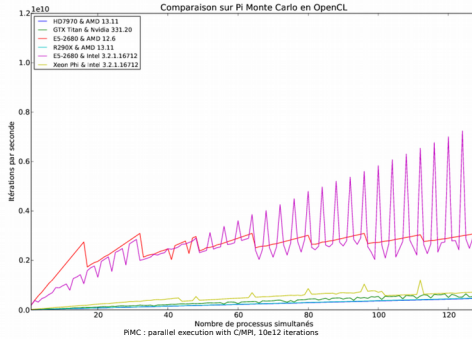
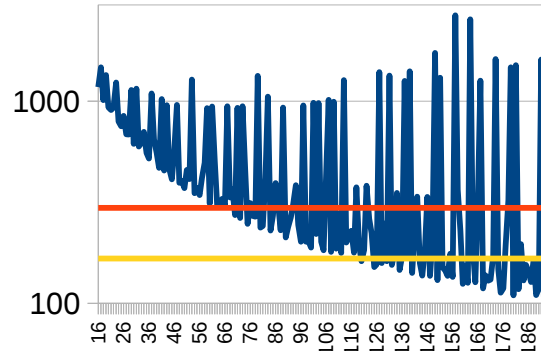
PID USER PRI NI VIRT RES SHR CPU MEM% TIME+ Command
9318 root 20 0 396M 73 28 0 0 0 4:58.19 /scratch/root/ben
9314 root 20 0 396M 71 28 0 0 0 4:58.15 /scratch/root/ben
9315 root 20 0 396M 7320 46288 100 0.3 4:58.19 /scratch/root/ben
9317 root 20 0 396M 7320 46284 100 0.3 4:58.19 /scratch/root/ben
9319 root 20 0 396M 7372 46336 100 0.3 4:58.22 /scratch/root/ben
9320 root 20 0 396M 7328 46292 100 0.3 4:58.20 /scratch/root/ben
9316 root 20 0 396M 7484 46444 100 0.3 4:58.19 /scratch/root/ben
9420 root 20 0 15936 3276 2672 0 0.9 0.0 0:00.04 htop
1 root 20 0 29416 4828 2632 5 0.0 0.0 0:13.21 /lib/systemd/syst
302 root 20 0 19856 2620 2336 5 0.0 0.0 0:01.56 /lib/systemd/syst
791 root 20 0 11644 2932 1920 5 0.0 0.0 0:00.24 /lib/systemd/syst
1191 root 20 0 5888 1388 1284 5 0.0 0.0 0:16.03 /usr/sbin/iscsid
```

```
Terminal - numa@survati -
Individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu May 19 16:14:27 2016 from lethc.cluster.zone
root@r410node64:~# htop
root@r410node64:~# dstat
You did not select any stats, using -cdngy by default.
Module dstat_disk2old failed to load. (No suitable block devices found to moni
or)

usr sys idl wai hix sig rmi cu swp
24 0 76 0 0 0 111k 9504k 0 0 2325 159
50 0 50 0 0 0 15k 640k 0 0 2340 145
50 0 50 0 0 0 2850k 640k 0 0 2380 107
50 0 50 0 0 0 8956k 2576k 0 0 2397 157
50 0 50 0 0 0 4934k 640k 0 0 2439 103
50 0 50 0 0 0 16k 640k 0 0 2355 156
50 0 50 0 0 0 22k 640k 0 0 2419 146
50 0 50 0 0 0 11k 640k 0 0 2478 133
50 0 50 0 0 0 8464k 746k 0 0 2405 126
50 0 50 0 0 24k 1974k 0 0 2356 183
50 0 50 0 0 19k 640k 0 0 2453 97
```

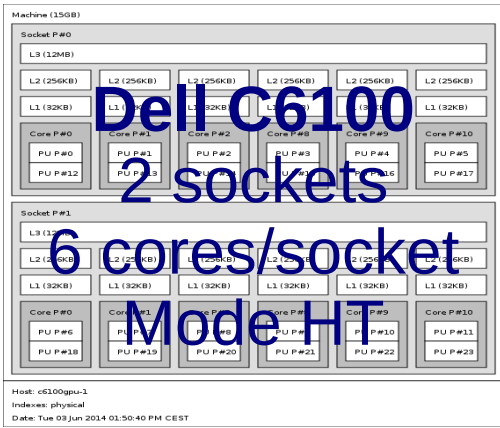
Amdahl law or lie ?

Are you ready to take the Red Pill ?



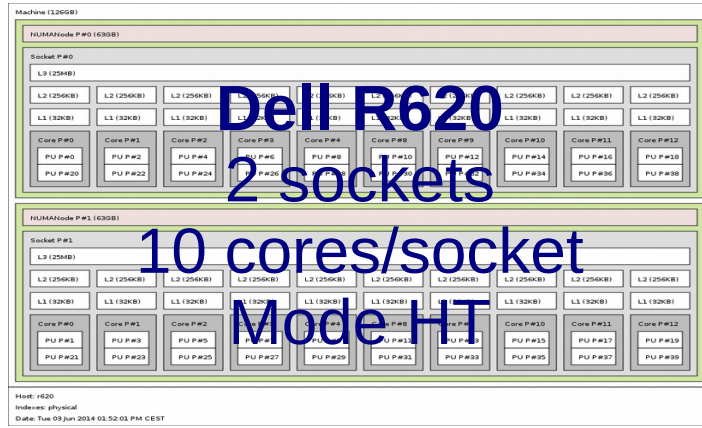
Welcome, Amdahl, to the Real World !

10 machines from 2 to 40 cores...



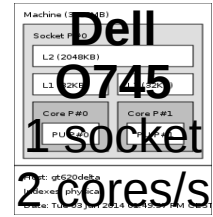
Dell C6100
2 sockets

6 cores/socket
Mode HT

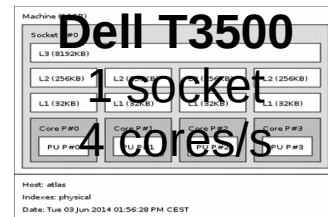


Dell R620
2 sockets

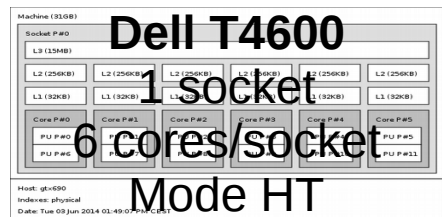
10 cores/socket
Mode HT



Dell O745
1 socket
2 cores/s



Dell T3500
1 socket
4 cores/s

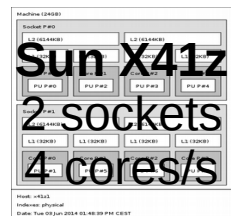


Dell T4600
1 socket
6 cores/socket
Mode HT

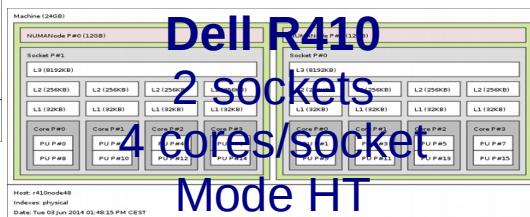
hwloc-ls as command



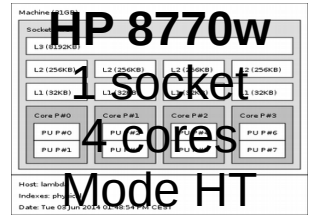
Sun X2200
2 sockets
4 cores/socket



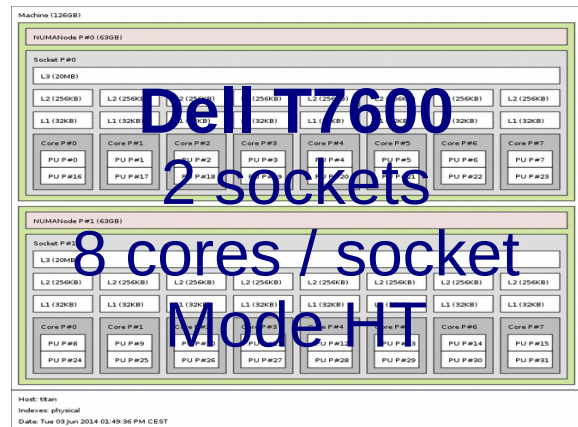
Sun X41z
2 sockets
4 cores/s



Dell R410
2 sockets
4 cores/socket
Mode HT



HP 8770w
1 socket
4 cores
Mode HT

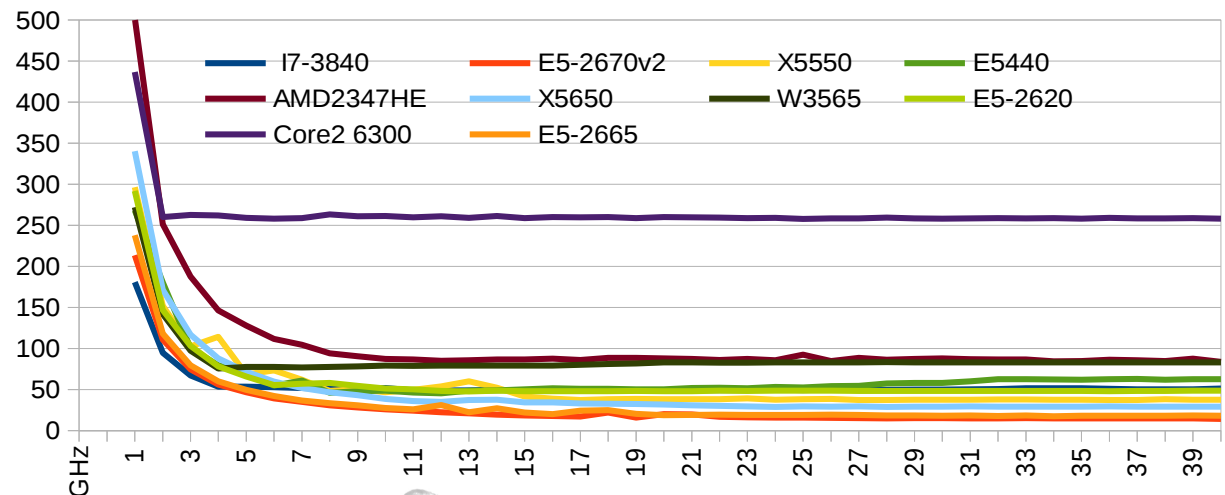


Dell T7600
2 sockets
8 cores / socket
Mode HT

Amdahl Law in the real world

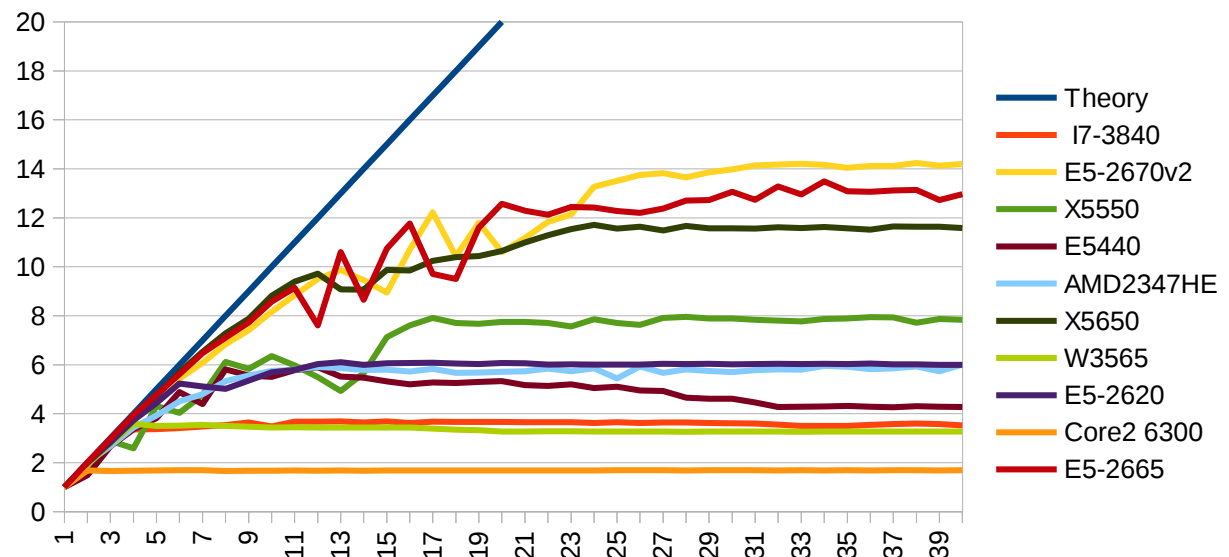
A test bench : 10 CPUs, 1 code

- Inside a node, 2 processors :
- 10 different CPUs, from 2 to 20 cores (40 in HT)
- 1 application : pbzip2 (parallel bzip2)
- 1 data set : encoded film (1.4 GB) (worst scenario)
- From 1 process to 80 process
- Metrology Tool : time
- Observable : elapsed

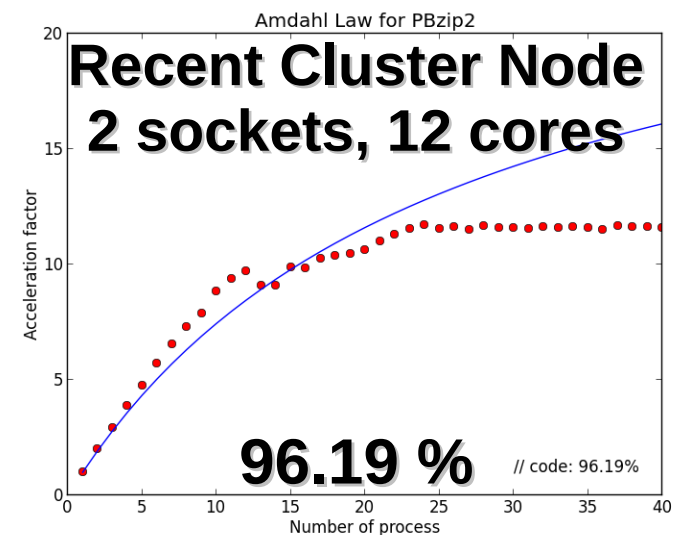
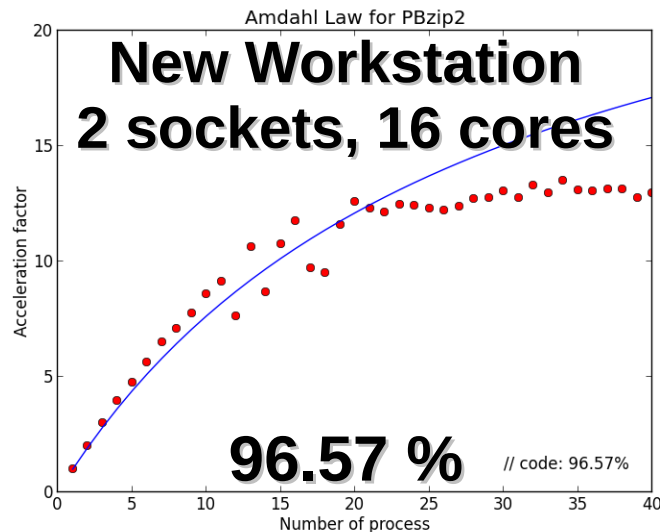
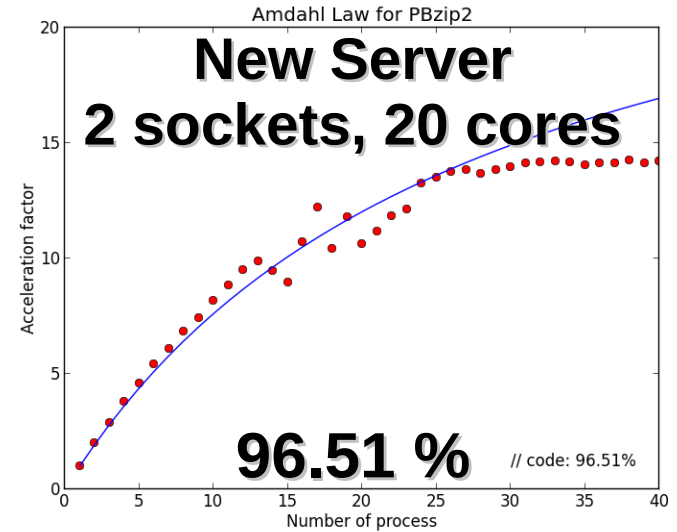
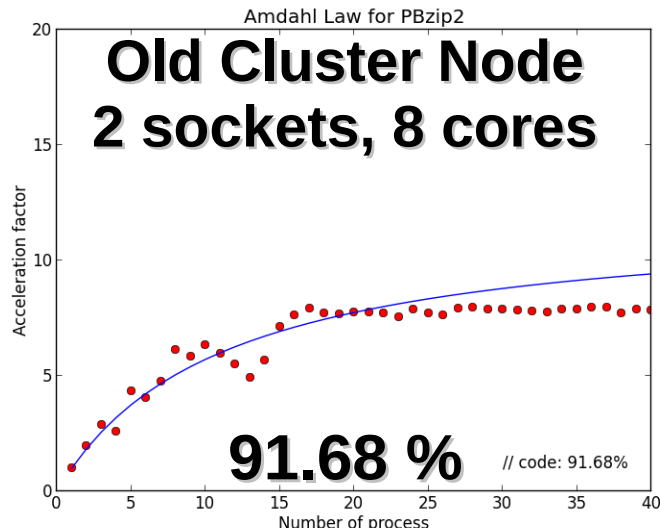


Amdahl Law in real world Acceleration & Variations

- Symptoms :
 - On large number of cores, 70 % to 80 % efficiency
 - Great variations on recent twice-sockets machines
 - Decrease for heavy charges on old processors

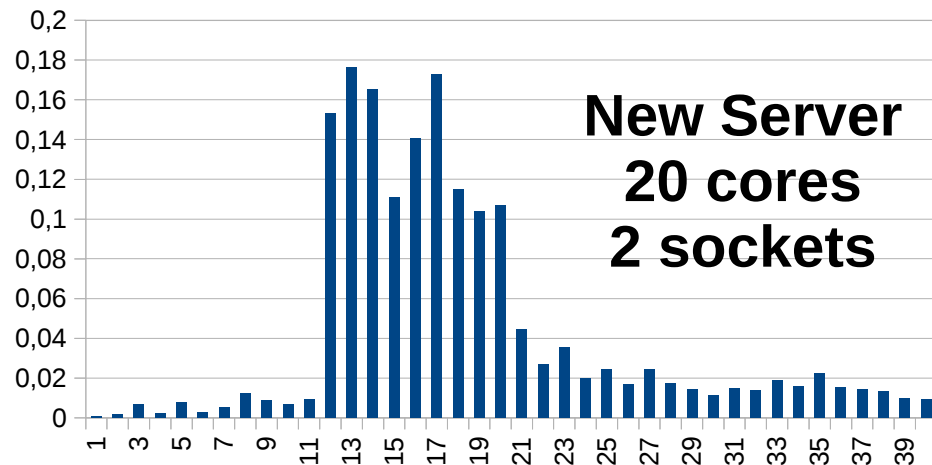
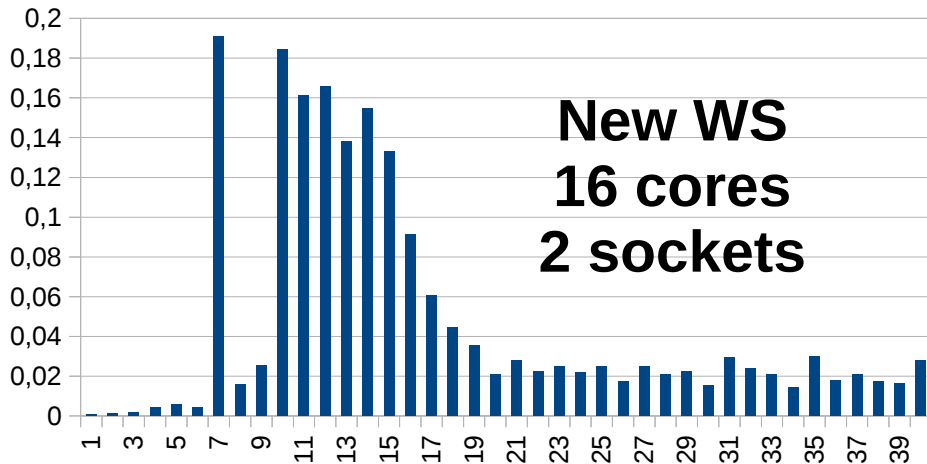
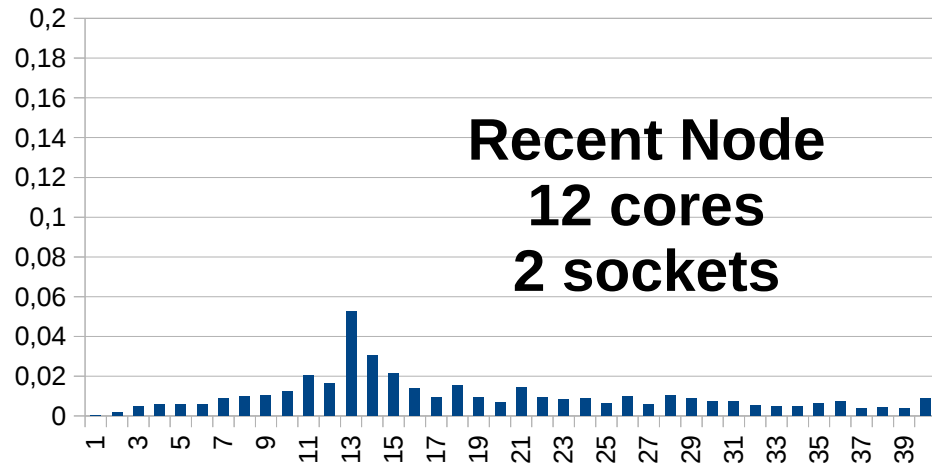
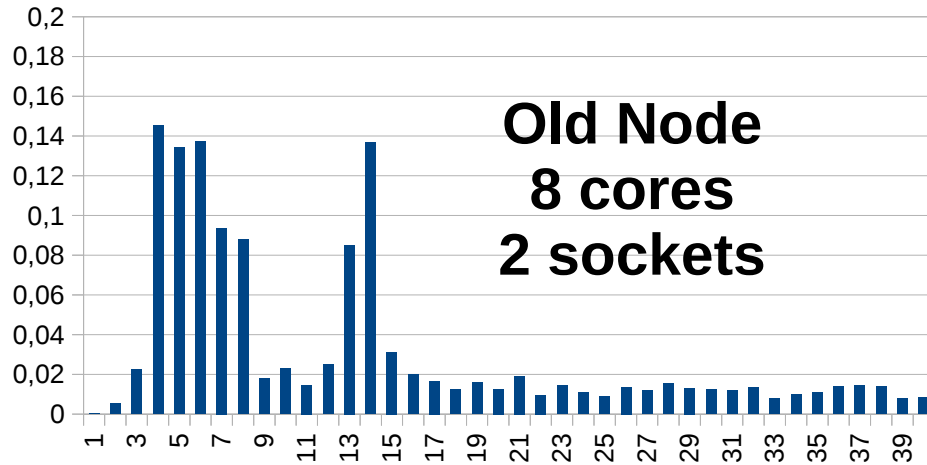


Amdahl Law : Fitting images !



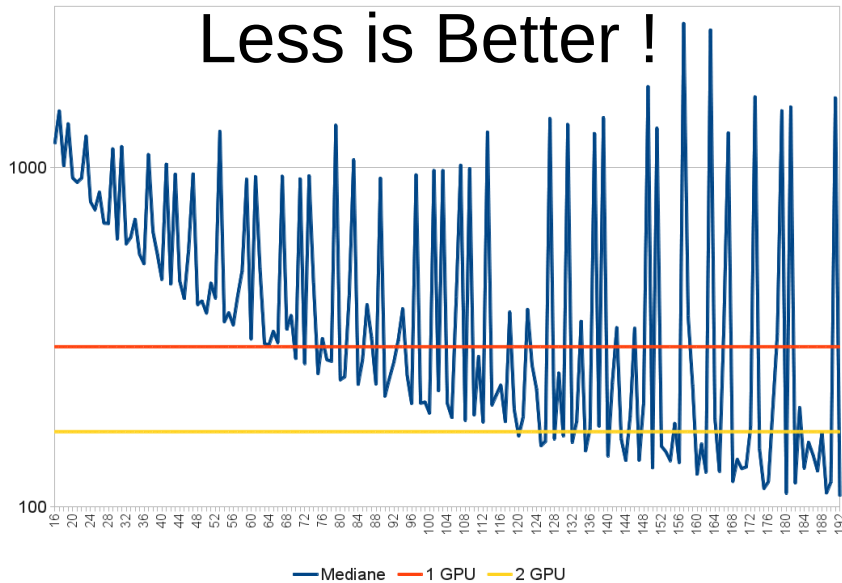
And worse : variability

Variability = Stddev/Median



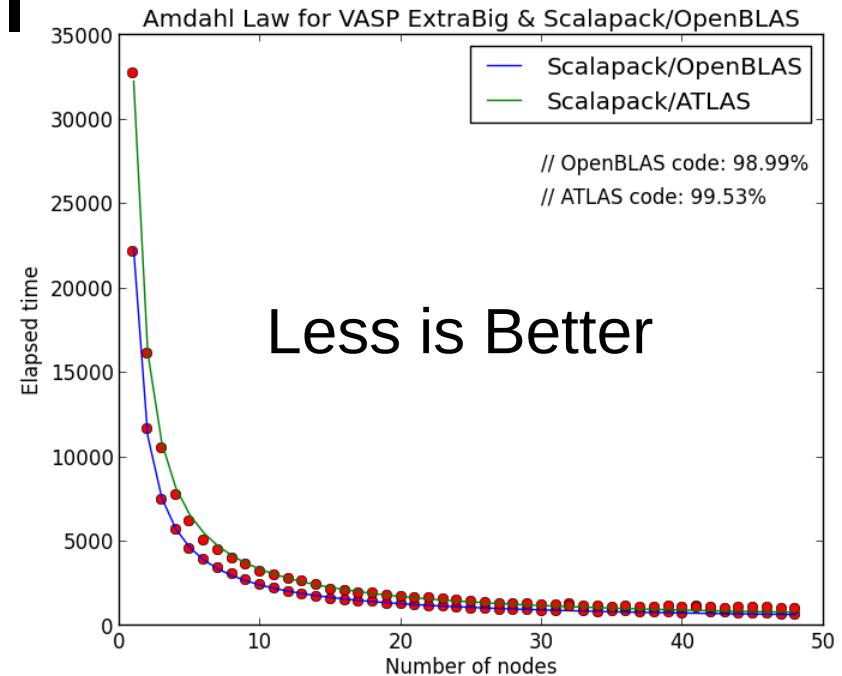
MPI applications, back to science

Amdahl law, in real



Lammps Molecular Dynamics application

- From 16 to 192 NP
- Compared to GPGPU
- 99.96 % //ized (personal record!)
- 2GPGPU equal 120 NP



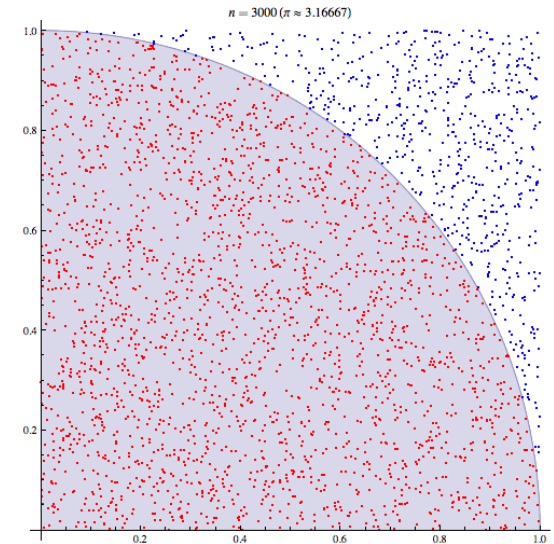
VASP DFT application

- From 1 to 48 NP
- 99 % to 99.5 % //ized
- OpenBLAS vs ATLAS...

What for simplistic implementations ?

PiMC : Pi by Dart Board Method

- Historical exemple for Monte Carlo Method : distribution
- Parallel implementation : distribution
 - From 2 to 4 parameters
 - Number of iterations
 - Parallel rate
 - (Type of variable : INT32, INT64, FP32, FP64)
 - (RNG : MWC, CONG, SHR3, KISS)
 - 2 simple observables :
 - Pi estimation (just indicative, Pi not rational :-)
 - Elapsed time



PiMC just for Houches 6th school

- Bench :

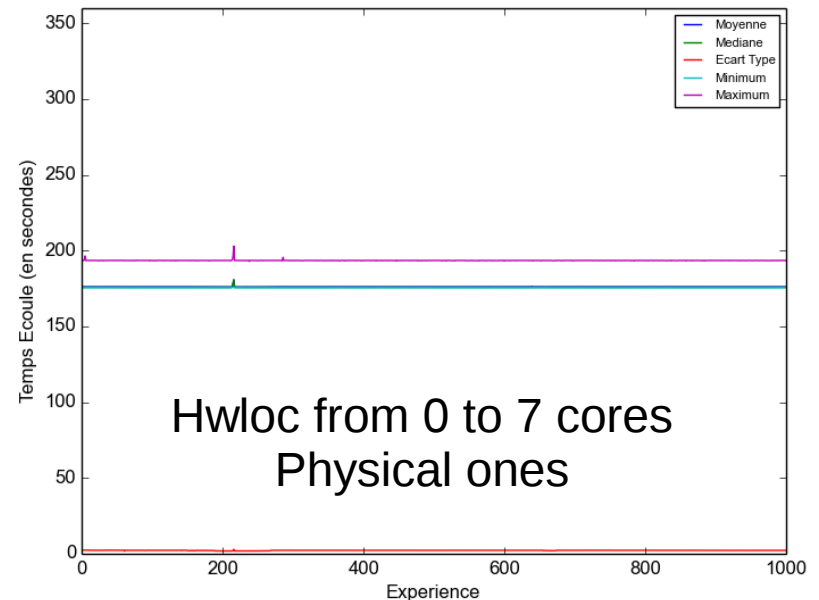
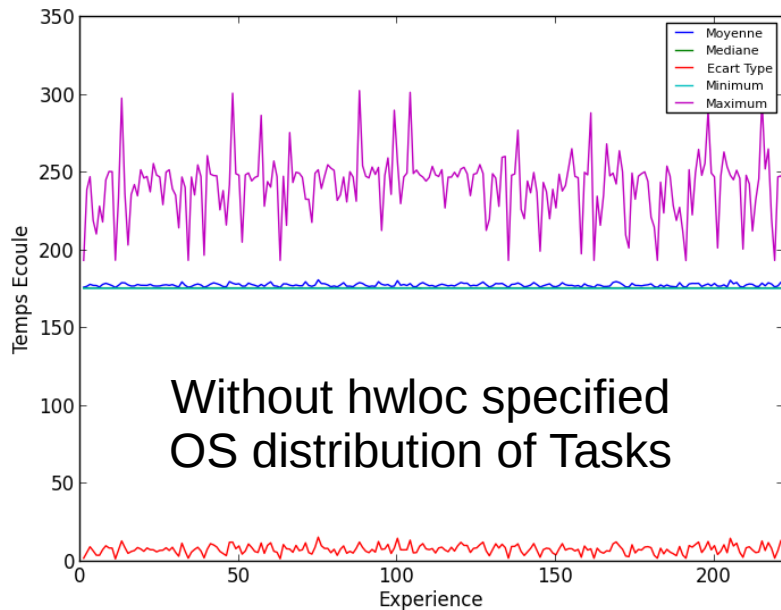
- Hardware : 64 R410 with 8 cores in HT mode, Infiniband Interconnect
 - Infiniband interconnect
- OS : Debian Jessie SIDUS
- Software : OpenMPI/C

- Experiences :

- Communications reduced to minimum
- 1 Piterations : 10^{12} equally distributed
- Parallel rate from 1 to 512 (sparse distribution)
- 40 launches for each Parallel Rate selected
- Metrology done by « time » program
- `/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 just for Why affinity selected ?

- During qualification of 48 nodes cluster
- Hundred of launches to evaluate reproducibility
- Morality : localize your process can be useful !



PiMC : and the results are...

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 : graphically

Does really Amdahl a good law ?

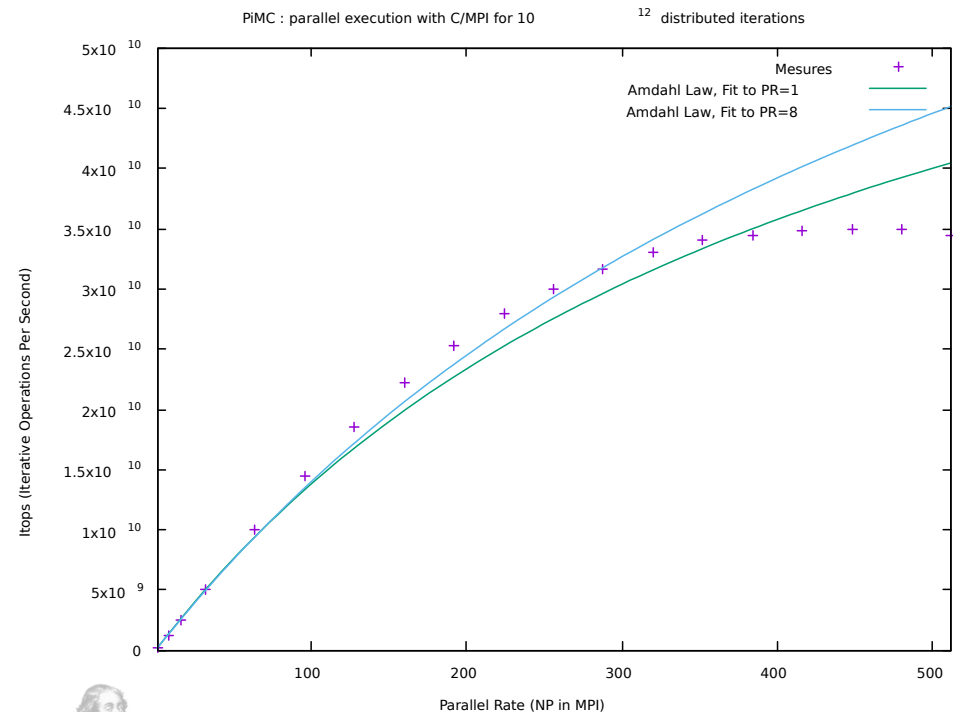
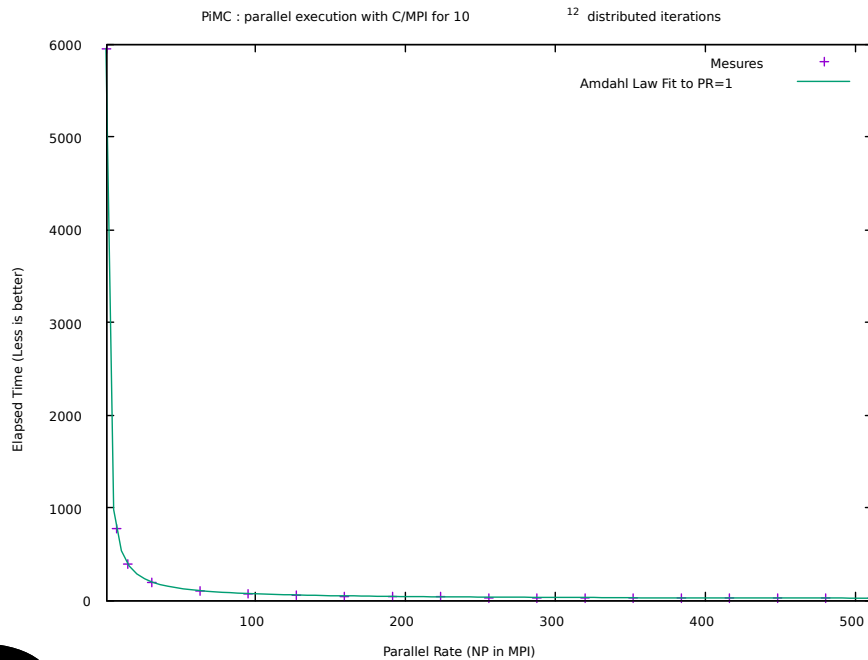
- Elapsed Time in seconds

- Seems to be nice, but...

- Performance in Itops

- Fit to 1 : $p=99.78\%$

- Fit to 8 : $p=99.83\%$



Evolution of Amdahl law :

Integer a linear influence : Mylq

PIMC : parallel execution with C/MPI for 10

12 distributed iterations

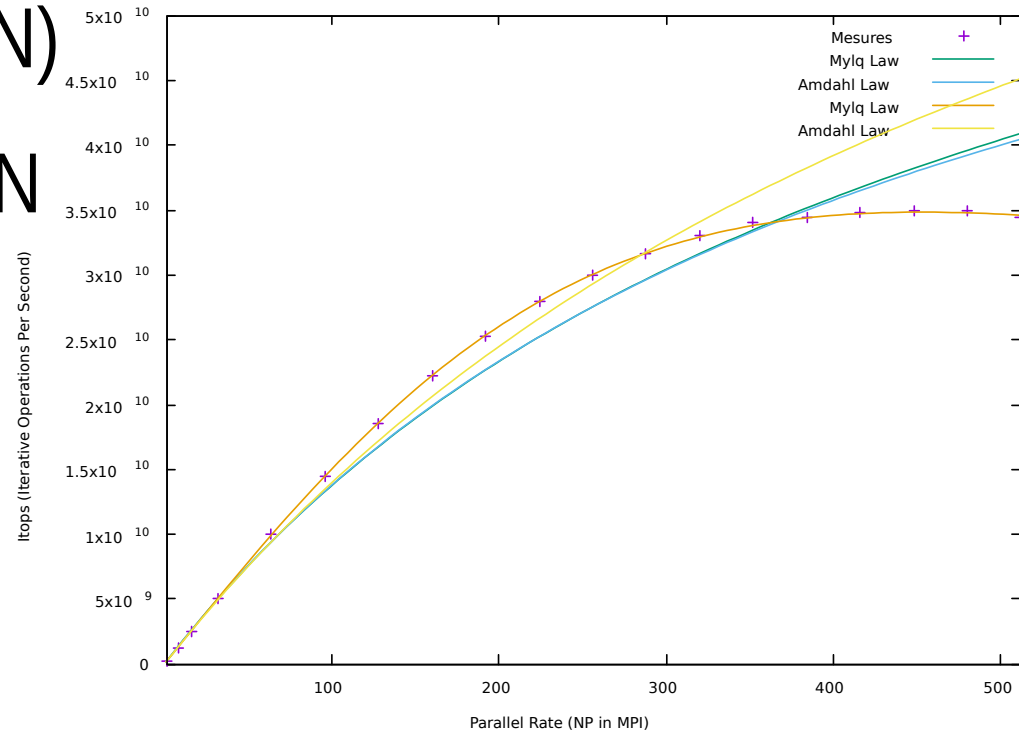
- Amdahl law : $T = T_1(1 - p/p/N)$

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

- Signification c_M :

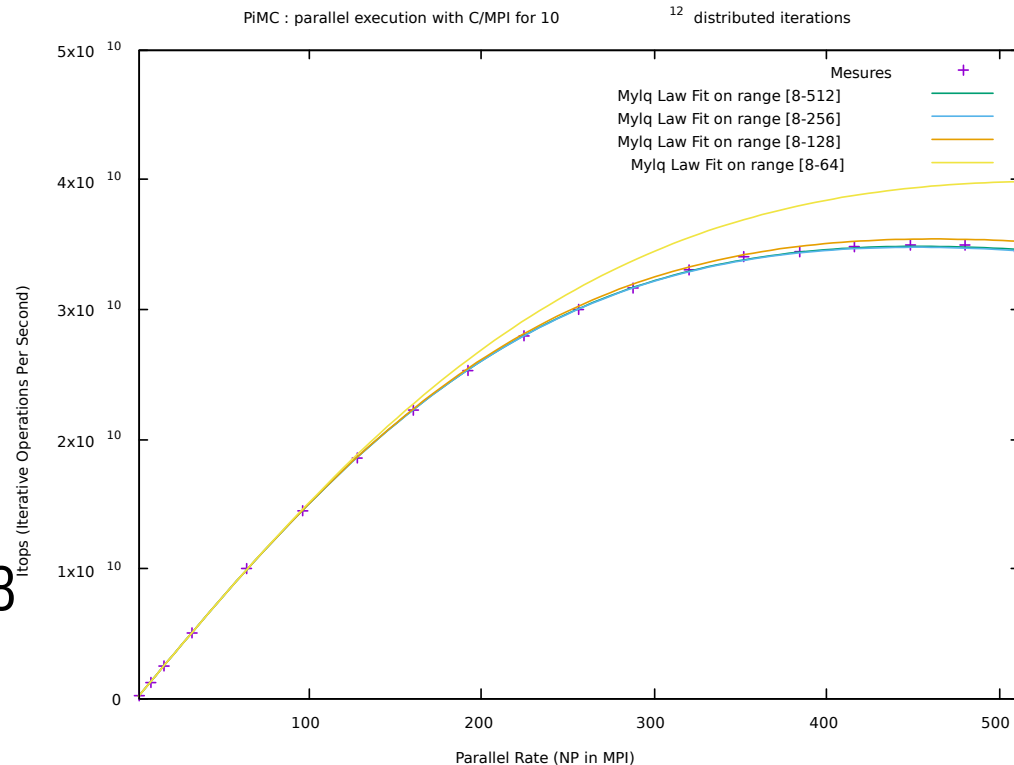
- Communications
- Initialization processes
- and $c_M \sim 0.03$,

- And $p_M \sim 0.9998$ with a fit which excludes PR=1 value



Mylq Law : Why exclude PR=1 Is a better predictable law ?

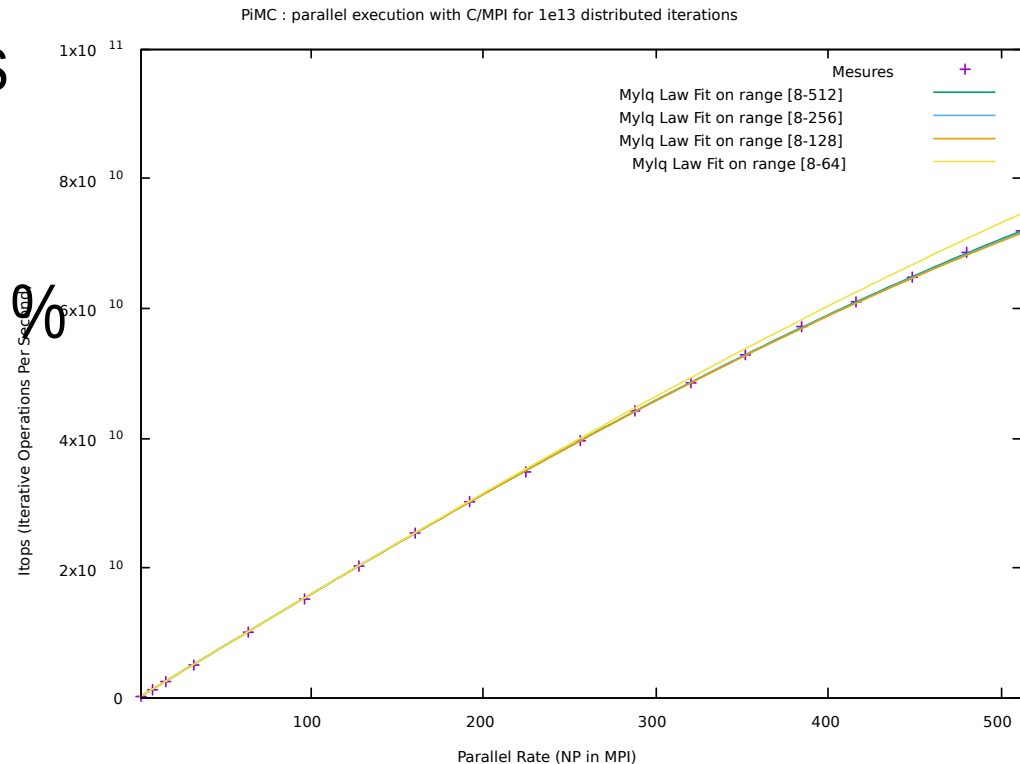
- Why exclude PR=1
 - Internal node mechanisms
 - OS effects
 - Processor effects : Turbo
- More predictable law ?
 - Try to fit with only : 1/2, 1/4, 1/8
 - On 1/4, it works fine ;-)
- But there are other effects to include...



Influence of Elapsed time

And if I increase iterations ?

- From 10^{12} to 10^{13} iterations
- Speedup from 208 to 448
- Efficiency from 40 % to 87 %
- Itops from 34 to 72 Gitops
- Mylq Parameters :
 - p_M reduced of 0.99998
 - $c_M=0.032$ (previous 0.03)



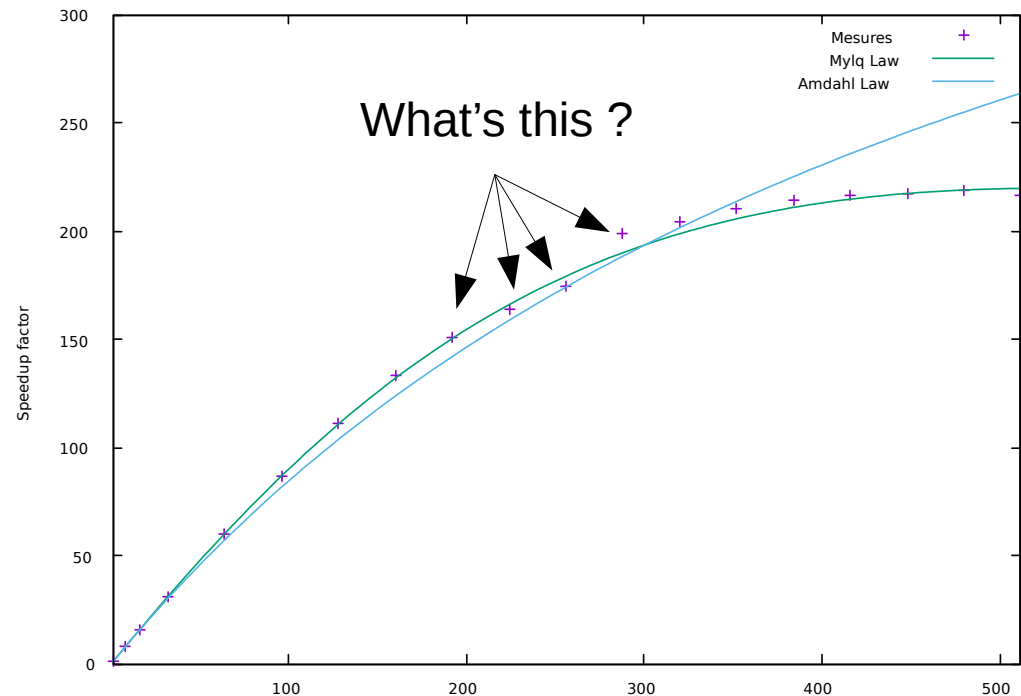
- Morality : don't be too stingy on your test sets ;-)

Why previous conclusions are not really honest...

- On last friday, Mylq fit was not so good... Why ?

- Lack of statistics ?
 - For each PR, 10 runs
 - Distribute on 64-nodes
 - Concurrent jobs
 - Exclusive in parts
 - Variability around 1 %
- Solution :
 - Exclusive runs in time

PIMC : parallel execution with C/MPI with 1e12 iterations

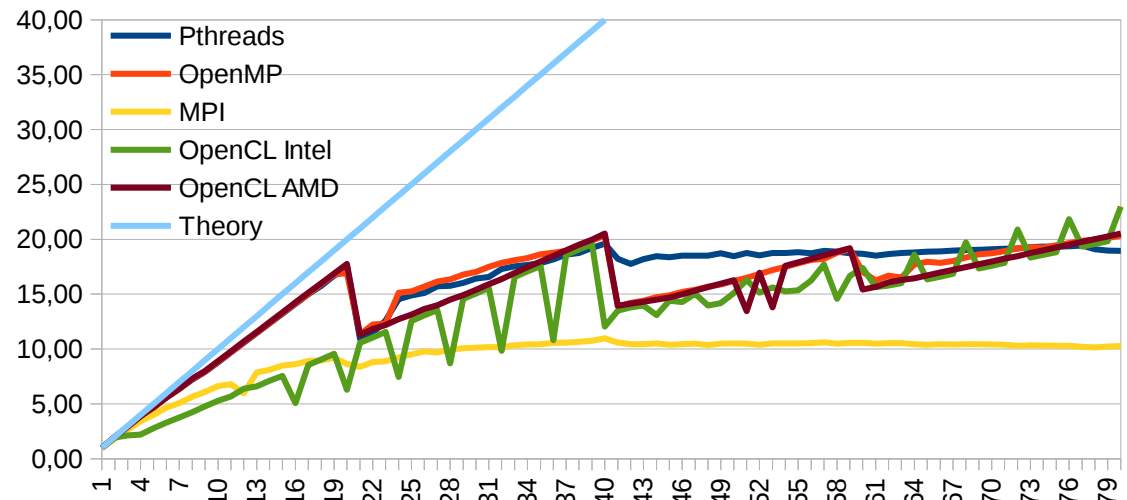
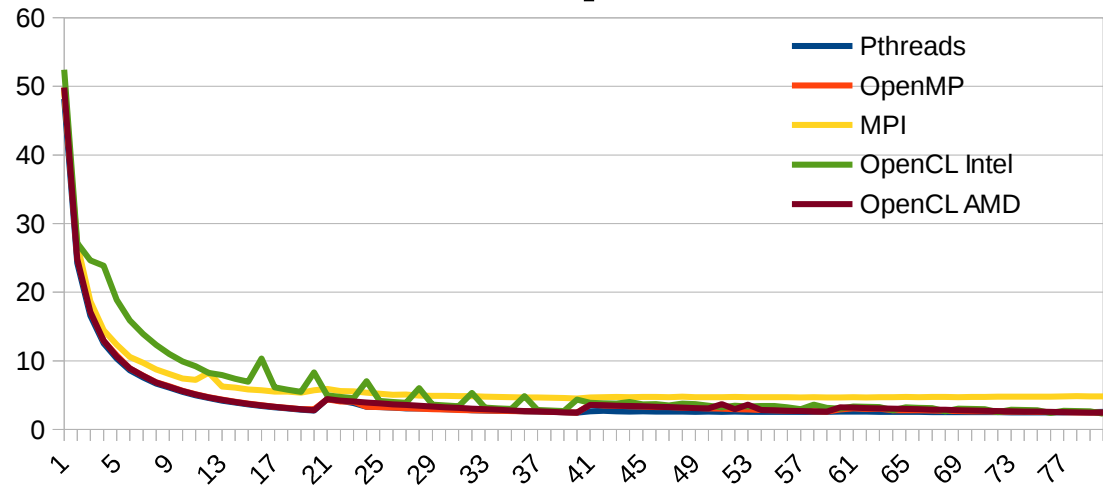


- So, coarse grain codes influence each others...

What about other parallelisms ?

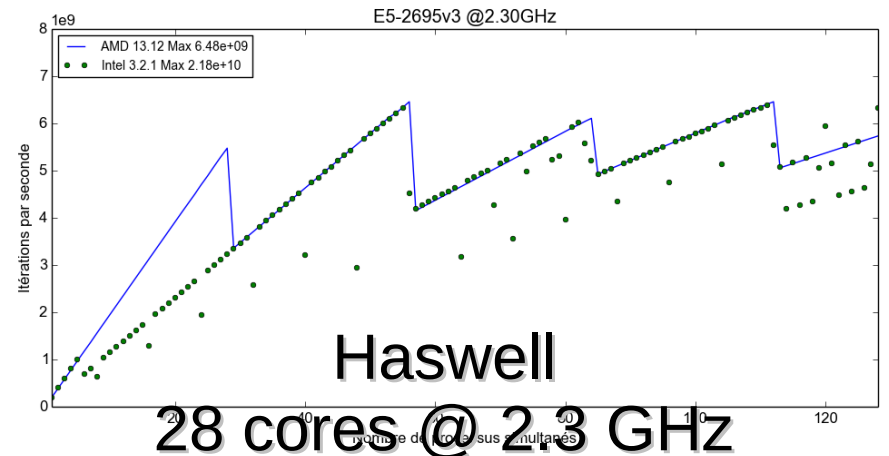
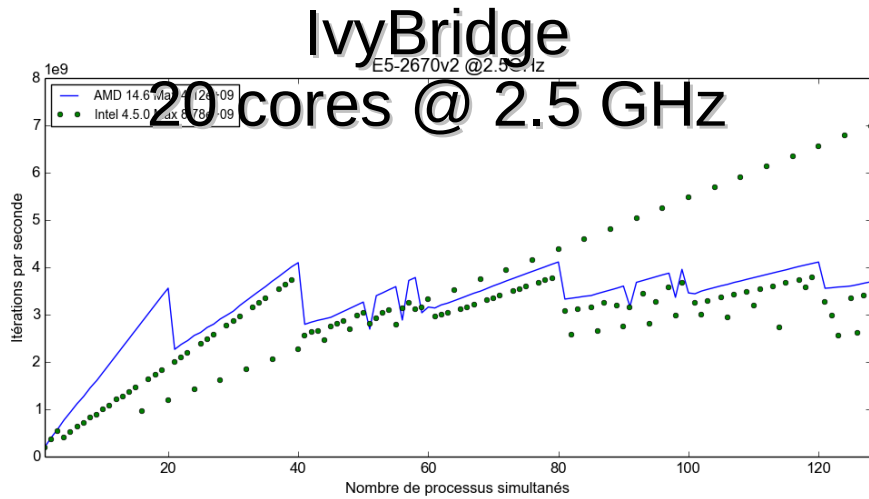
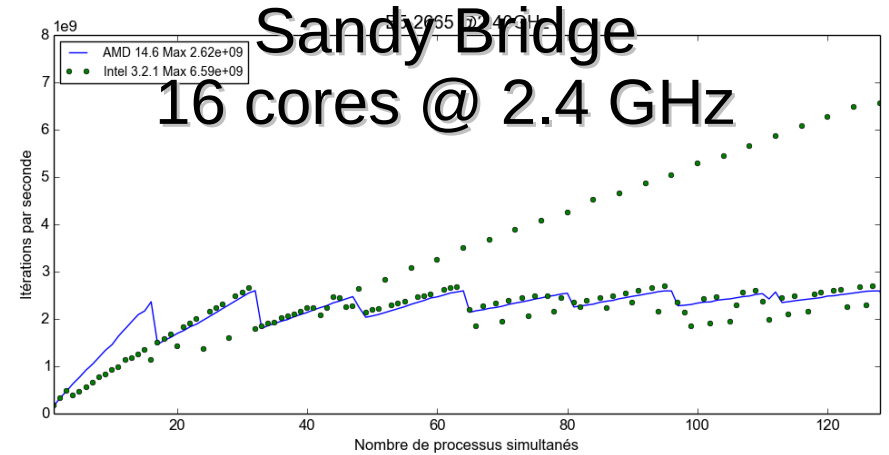
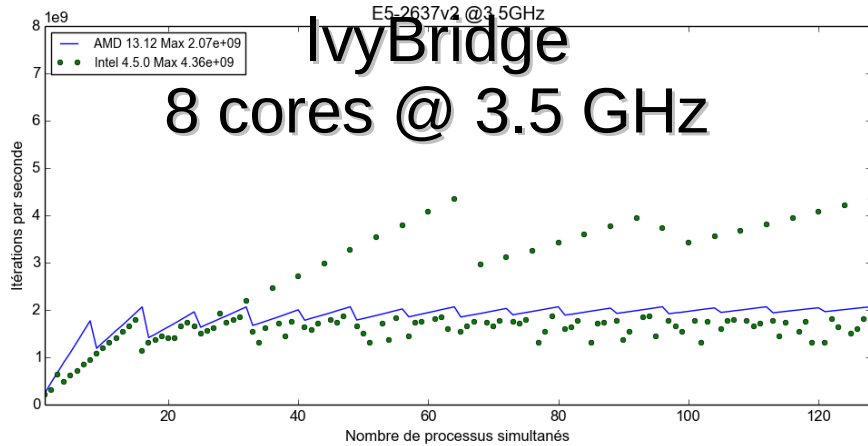
It's worse... Small example

- Let's return inside the node...
- A Dell server PowerEdge R620
 - Bi-socket, 10-cores : 20 cores
 - Hyperthreading mode activated : 40
- Parallel implementations
 - MPI in C
 - OpenMP in C
 - Pthreads in C
 - OpenCL in Python
 - OpenCL by AMD
 - OpenCL by Intel
- Finalement, pas mal OpenCL !

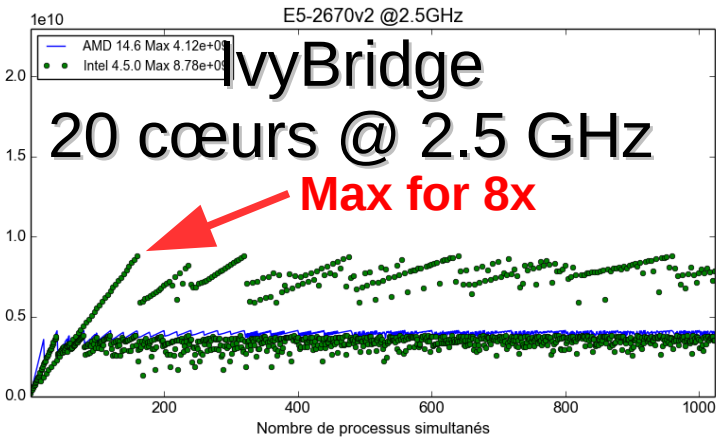
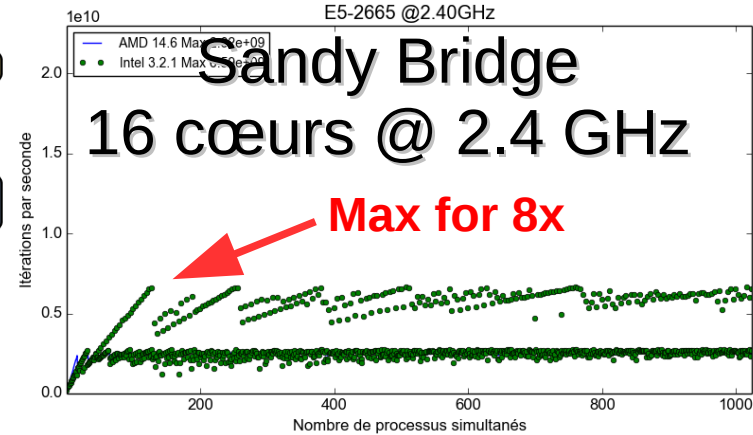
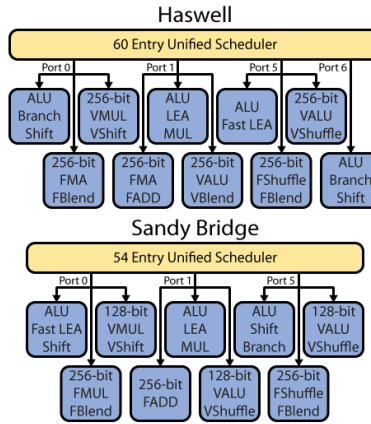
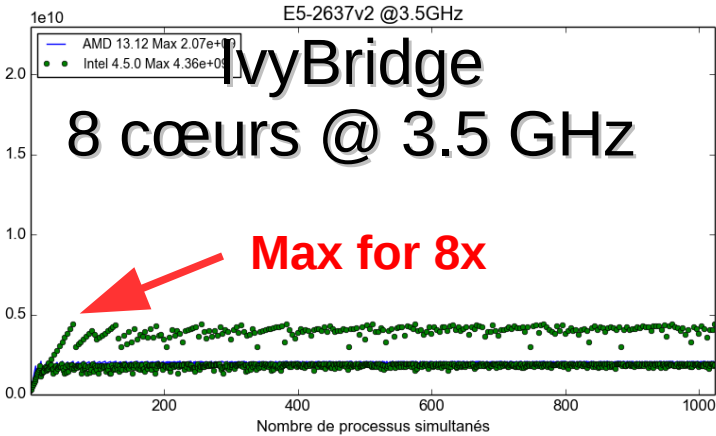


Ok, but for other processors ?

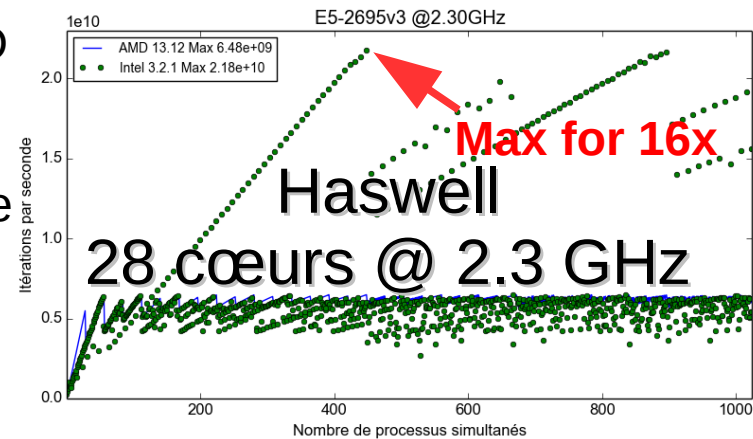
Mmmm... Interesting...



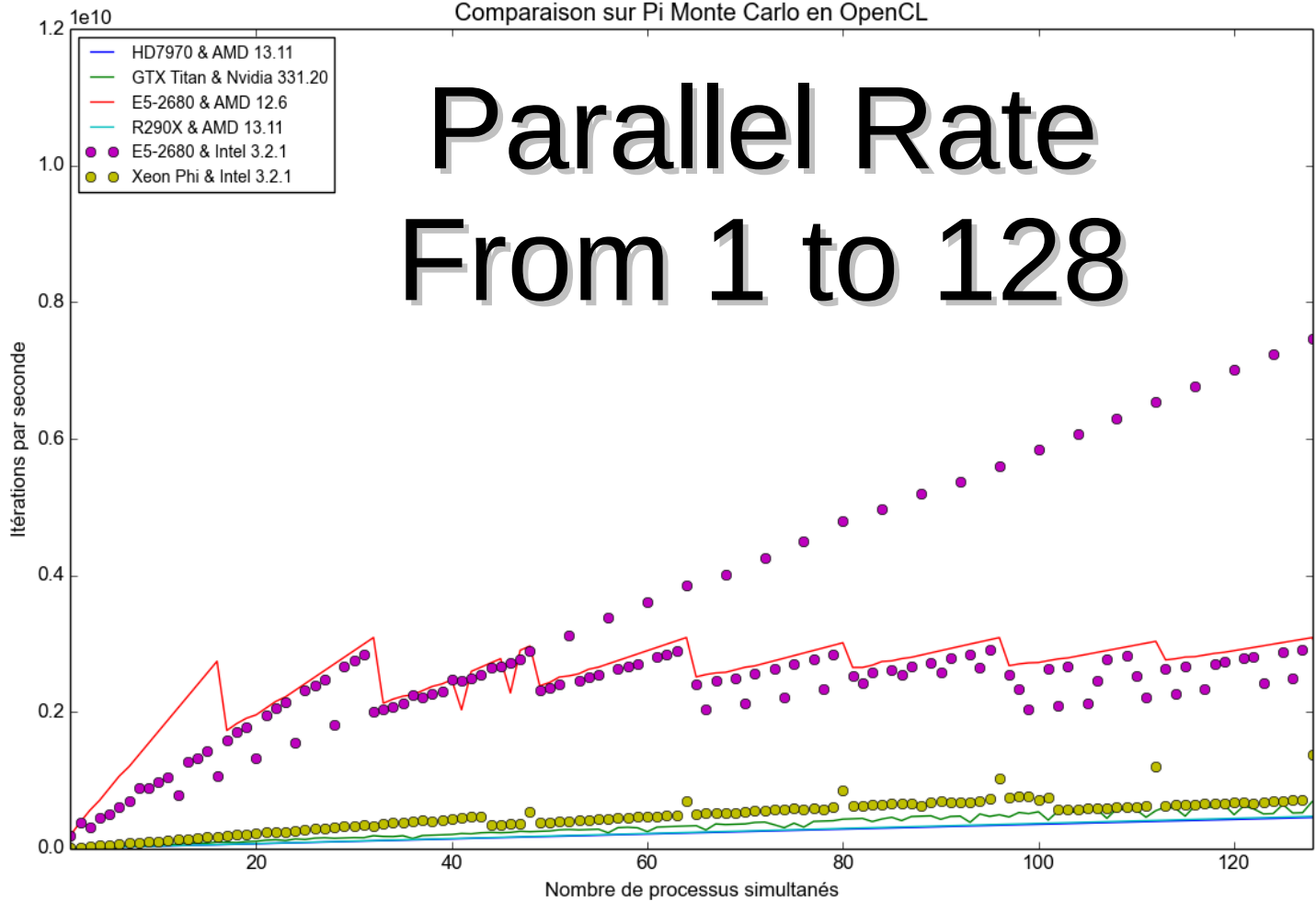
And for PR >> Number of cores EPU depends of architecture !



- Intel x2,3 vs AMD
Period of 4
Max Perf :
- x8 Sandybridge
 - x8 IvyBridge
 - x16 Haswell

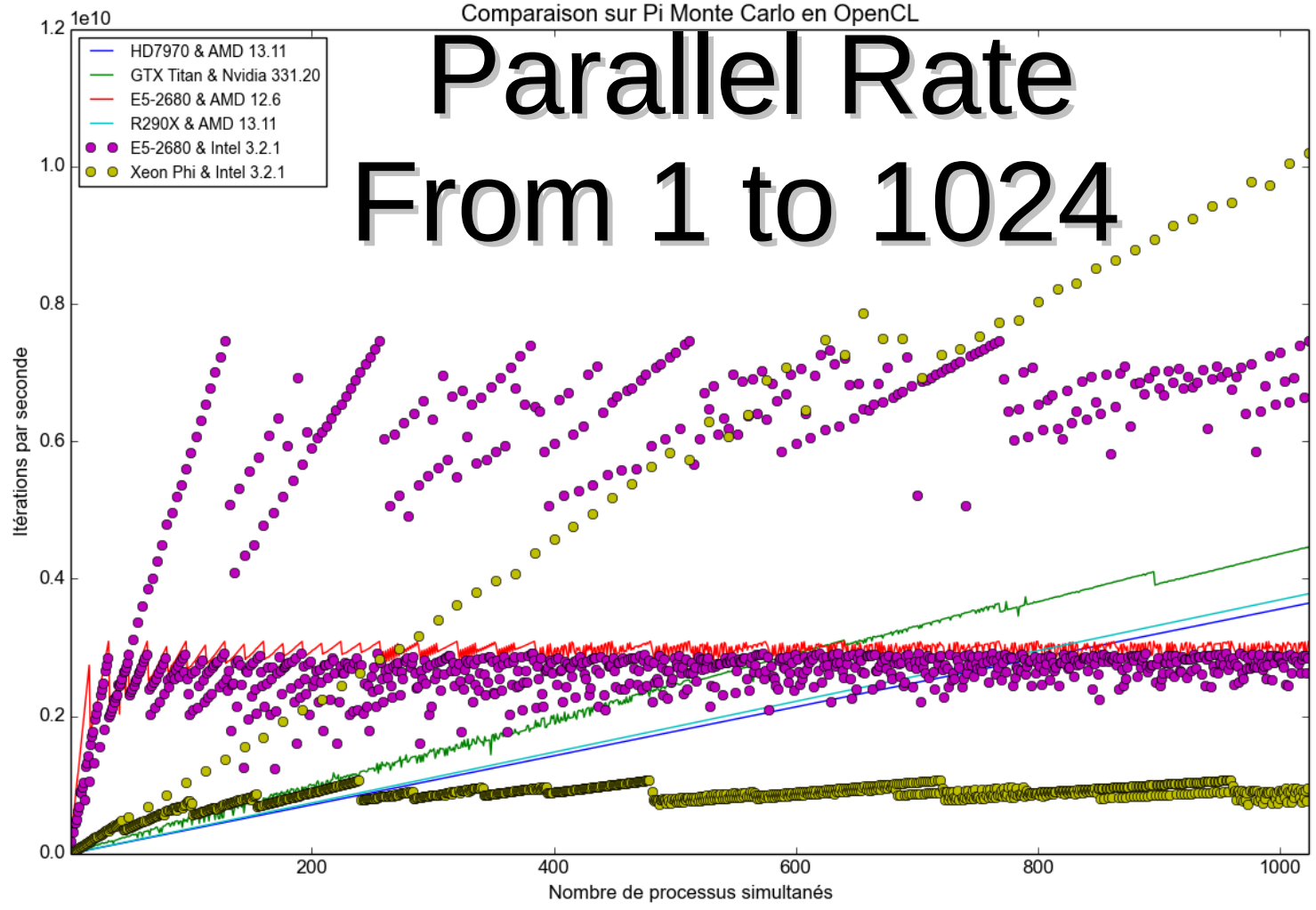


And for others architectures Mouth-watering before tomorrow

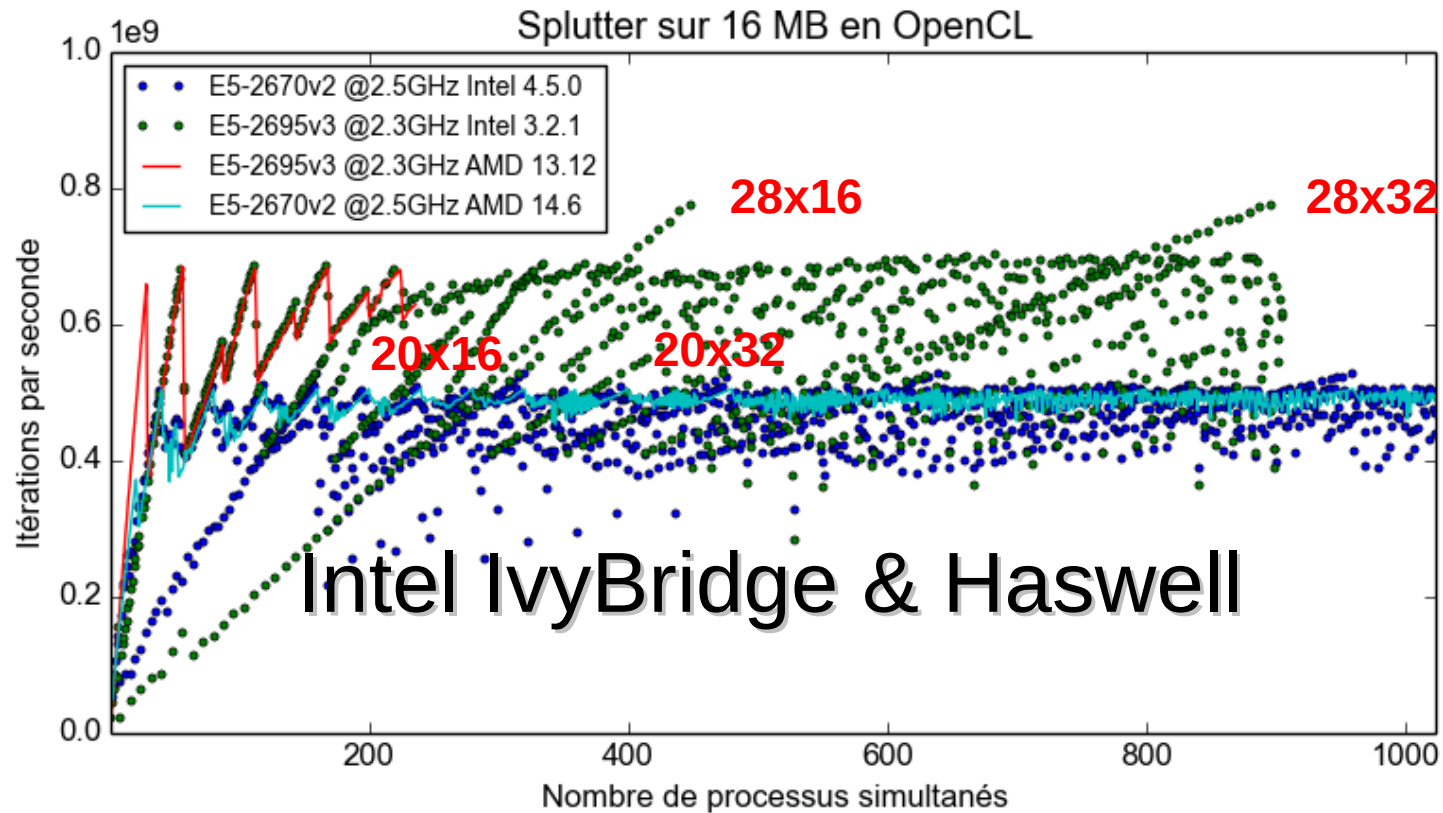


Parallel Rate From 1 to 128

And for others architectures Increasing PR to explore...



After, *PU bound, Memory Bound The « splutter » to stress memory



Introduction to conclusion :

IT : The new world of complexity

- Complicated : from « cum plicare », « fold together »
 - Descartes : « All is the sum of parts. »
- Complex : from « cum plexus », « weave together »
 - Huge amount of interactions, non linearity, emergence, ...
- Computing resources are complex systems
 - A running Operating System has at least 200 process running background
 - CPU cores change frequency & voltage all the time, start/stop, ...
 - DRAM change frequency all the time
 - Communication devices (network) are all random access components

OSI Model & Amdahl Law Evolutions & perspectives

- OSI Model : Layer below seen as a service
 - Ignoring all the infrastructure is clearly a suicide for scalability
- Amdahl Law : Only depends of T_1 and p
 - It cannot be used...
- Mylg Law : add a simple proportional factor
 - Can help you to evaluate scalability and predictive performance
- Inside a node, nothing works
 - And in a GPU or accelerator