

HPC for theoretical physics

R. (lele) Tripiccione
tripiccione@fe.infn.it

Workshop CCR
La Biodola, May 19th, 2016

On the menu today

Computational theoretical physics and HPC.

HPC vs. HTC

HPC workhorses today

Trade secrets

The INFN way ...

WhatNext & WhatNextNext (Marconi / IBM Coral)

Programming: Present & Future

Conclusions & take-away lessons

The starting point:

*Computational theoretical physics, when **heavy**, is intrinsically **parallel**. Indeed*

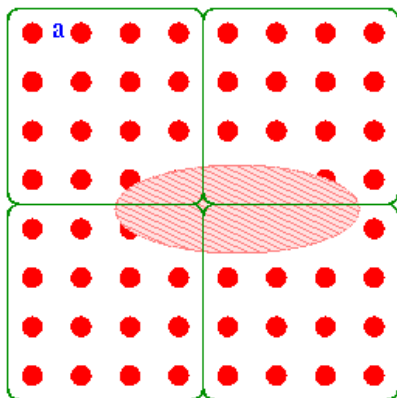
*The world is “run” by partial differential equations -->
Implying short-distance communication in space and time*

So you can divide the physical system in chunks, and compute on each of them independently

**** **IF** ****

You can exchange the needed informations among those chunk

An example from Lattice QCD →



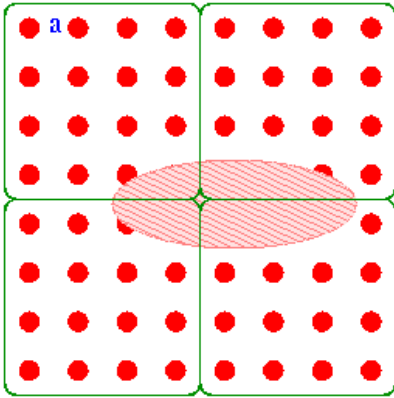
An example from Lattice QCD

When computing the Dirac operator (the key computing kernel of LatticeQCD)

For each 4-D slice of your physical system of size L^4

$$C \sim 584 L^4 \text{ ops} \quad I \sim 8 \times (3 \times 8) L^3 \text{ bytes}$$

$$\frac{C}{I} = \frac{P}{B} \equiv \rho \approx \frac{584 L^4}{8 \times 3 \times 8 L^3} = 3 L$$



An example from Lattice QCD

If you put a tile of size L^4 on each processing element, you need

$$\frac{C}{I} = \frac{P}{B} \equiv \rho \approx \frac{584 L^4}{8 \times 3 \times 8 L^3} = 3 L$$

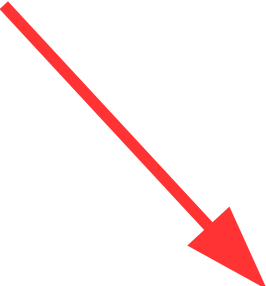
*If your computer **fulfills** this requirement, can do it **parallel***

*Just an example: if $P/B \sim 10$ ops/byte, can study a (not too large) physical system of 32^4 points on **10000 processing elements***

HPC vs HTC

The typical HPC workhorse of 2013 – 2016 -->

Fermi @ Cineca	Blue Gene/Q
Cores/node	16
Perf. / core	12.5 Gflops
Mem/cores	1 Gbyte
# of cores	160000
Peak Flops	2.1 Pflops
Energy eff.	3.3 Gflops/W
Node bandwidth	20 GB/sec
Perf / Bwidth	10 Flops / byte



HPC vs HTC

*The typical HTC workhorse of 2013 – 2016 →
(thanks to G. Lo Presti)*

MEYRIN DATA CENTRE		WIGNER DATA CENTRE	
	last_value		last_value
● Number of Cores in Meyrin	133,507	● Number of Cores in Wigner	43,328
● Number of Drives in Meyrin	80,926	● Number of Drives in Wigner	23,100
● Number of 10G NIC in Meyrin	7,107	● Number of 10G NIC in Wigner	1,399
● Number of 1G NIC in Meyrin	22,369	● Number of 1G NIC in Wigner	5,067
● Number of Processors in Meyrin	23,007	● Number of Processors in Wigner	5,418
● Number of Servers in Meyrin	12,273	● Number of Servers in Wigner	2,712
● Total Disk Space in Meyrin (TB)	170,963	● Total Disk Space in Wigner (TB)	71,738
● Total Memory Capacity in Meyrin (TB)	544	● Total Memory Capacity in Wigner (TB)	172

HPC vs HTC

Any differences????

	CINECA	CERN
Cores/node	16	
Perf. / core	12.5 Fflops	12.5 Gflops ??
Mem/cores	1 Gbyte	4 Gbyte
# of cores	160000	~ 177000
Peak Flops	2.1 Pflops	2.3 Pflops ??
Energy eff.	3.3 Gflops/W	???
Node bandwidth	20 GB/sec	
Core bandwidth	1.25 GB/sec	~ 0.06 GB/sec
Perf / Bwidth	10 Flops / byte	~ 208 Flops / byte

HPC vs HTC

Any differences????

	CINECA	CERN
Cores/node	16	
Perf. / core	12.5 Fflops	12.5 Gflops ??
Mem/cores	1 Gbyte	4 Gbyte
# of cores	160000	~ 177000
Peak Flops	2.1 Pflops	2.3 Pflops ??
Energy eff.	3.3 Gflops/W	???
Node bandwidth	20 GB/sec	
Core bandwidth	1.25 GB/sec	~ 0.06 GB/sec
Perf / Bwidth	10 Flops / byte	~ 208 Flops / byte

Any reason for these differences?????

HPC vs HTC

Any reason for these differences???



At CERN ~ 1.5 cores per job

At FERMI, on average 20 jobs on 160K cores → 8000 cores / job

HPC workhorses today

Looking at the recent past ...



Current HPC machines

Two examples two-three years between the two

	Fermi @ Cineca	SuperMUC
Cores/node	16	28
Perf. / core	12.5 Fflops	41.6 Gflops
Mem/cores	1 Gbyte	
# of cores	160000	86000
Peak Flops	2.1 Pflops	3.58 Pflops
Energy eff.	3.3 Gflops/W	3.5 Gflops/W
Node bandwidth	20 GB/sec	18 Gbyte/sec
Core bandwidth	1.25 GB/sec	~ 0.65 GB/sec
Perf / Bwidth	10 Flops / byte	~ 63 Flops / byte

Current HPC machines

Two examples

	Fermi @ Cineca	SuperMUC
Cores/node	16	28
Perf. / core	12.5 Fflops	41.6 Gflops
Mem/cores	1 Gbyte	
# of cores	160000	86000
Peak Flops	2.1 Pflops	3.58 Pflops
Energy eff.	3.3 Gflops/W	3.5 Gflops/W
Node bandwidth	20 GB/sec	18 Gbyte/sec
Core bandwidth	1.25 GB/sec	~ 0.65 GB/sec
Perf / Bwidth	10 Flops / byte	~ 63 Flops / byte

.... showing a worrying trend!

Trade secrets 1: the official view

The Top500 ranking list in 1978 !!!!

$\frac{2}{3} N^2$ $2N \sim \log$ \downarrow TIME

UNIT = 10^{**6} TIME / ($1/3$ 100^{**3} + 100^{**2})

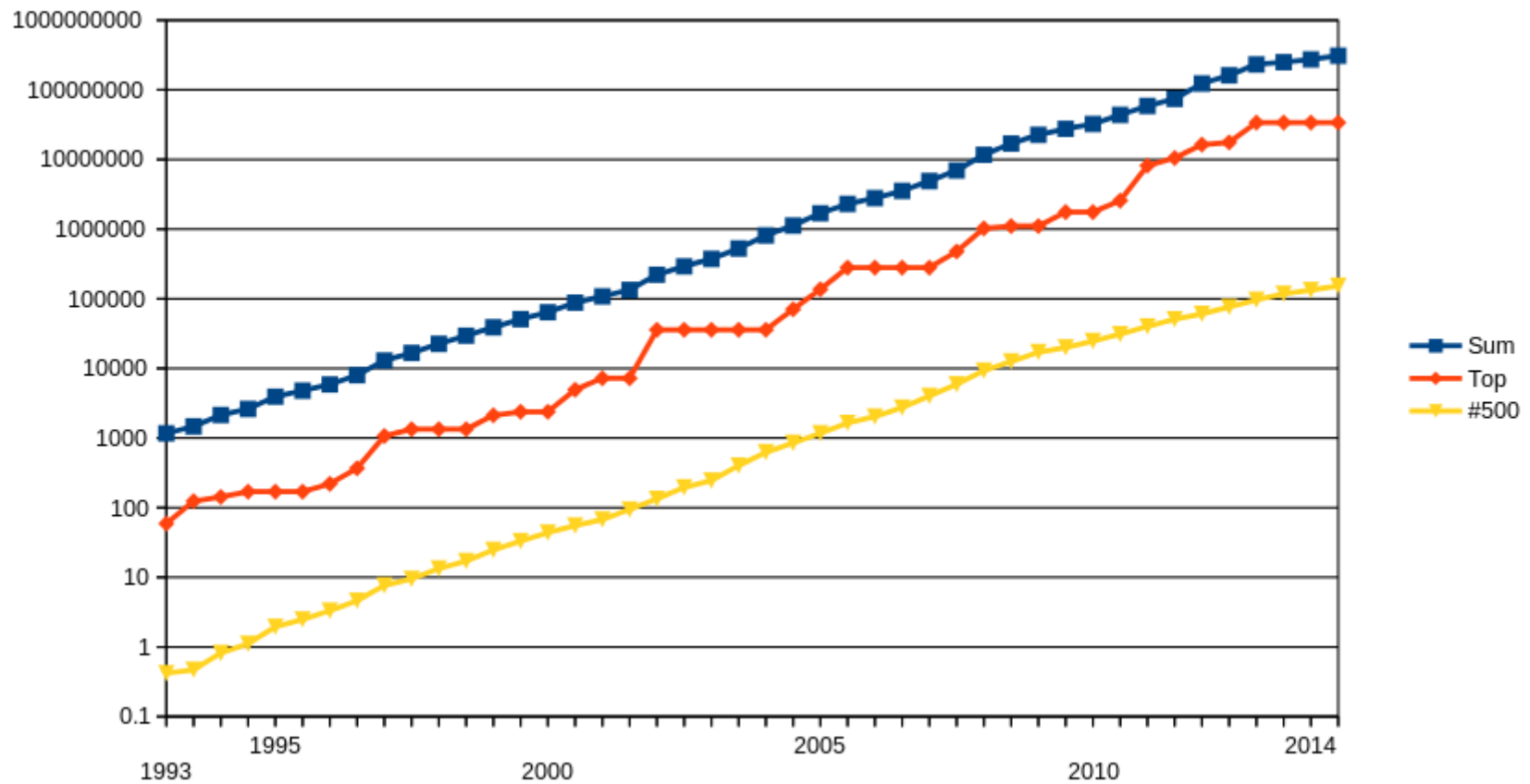
Facility	TIME N=100 secs.	UNIT micro- secs.	Computer	Type	Compiler
NCAR	14.0 .049	0.14	CRAY-1	S	CFT, Assembly BLAS
LASL	4.44 .148	0.43	CDC 7600	S	FTN, Assembly BLAS
NCAR	3.54 .192	0.56	CRAY-1	S	CFT
LASL	3.27 .210	0.61	CDC 7600	S	FTN
Argonne	2.31 .297	0.86	IBM 370/195	D	H
NCAR	1.91 .359	1.05	CDC 7600	S	Local
Argonne	1.77 .388	1.33	IBM 3033	D	H
NASA Langley	1.40 .489	1.42	CDC Cyber 175	S	FTN
U. Ill. Urbana	1.44 .506	1.47	CDC Cyber 175	S	Ext. 4.6
ILL	1.44 .554	1.61	CDC 7600	S	CHAT, No optimize
SLAC	1.19 .579	1.69	IBM 370/168	D	H Ext., Fast mult.
Michigan	1.09 .631	1.84	Amdahl 470/V6	D	H
Toronto	.774 .690	2.59	IBM 370/165	D	H Ext., Fast mult.
Northwestern	.477 1.44	4.70	CDC 6600	S	FTN
Texas	.354 1.93*	5.63	CDC 6600	S	FOR
China Lake	.352 1.95*	5.69	Univac 1110	S	V
Yale	.345 2.59	7.53	DEC KL-20	S	F20
Bell Labs	.477 3.46	10.1	Honeywell 6080	S	V
Wisconsin	.477 3.49	10.1	Univac 1110	S	V
Iowa State	.454 3.54	10.2	Intel AS/5 mod3	D	H
U. Ill. Chicago	.454 4.10	11.9	IBM 370/158	D	G1
Purdue	.454 5.69	16.6	CDC 6500	S	FOR
U. C. San Diego	.454 13.1	38.2	Burroughs 6700	S	H
Yale	.454 17.1*	49.9	DEC KA-10	S	F40

* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGEFL(75)

Thanks to J. Dongarra for this (and a few more) hard to find pictures

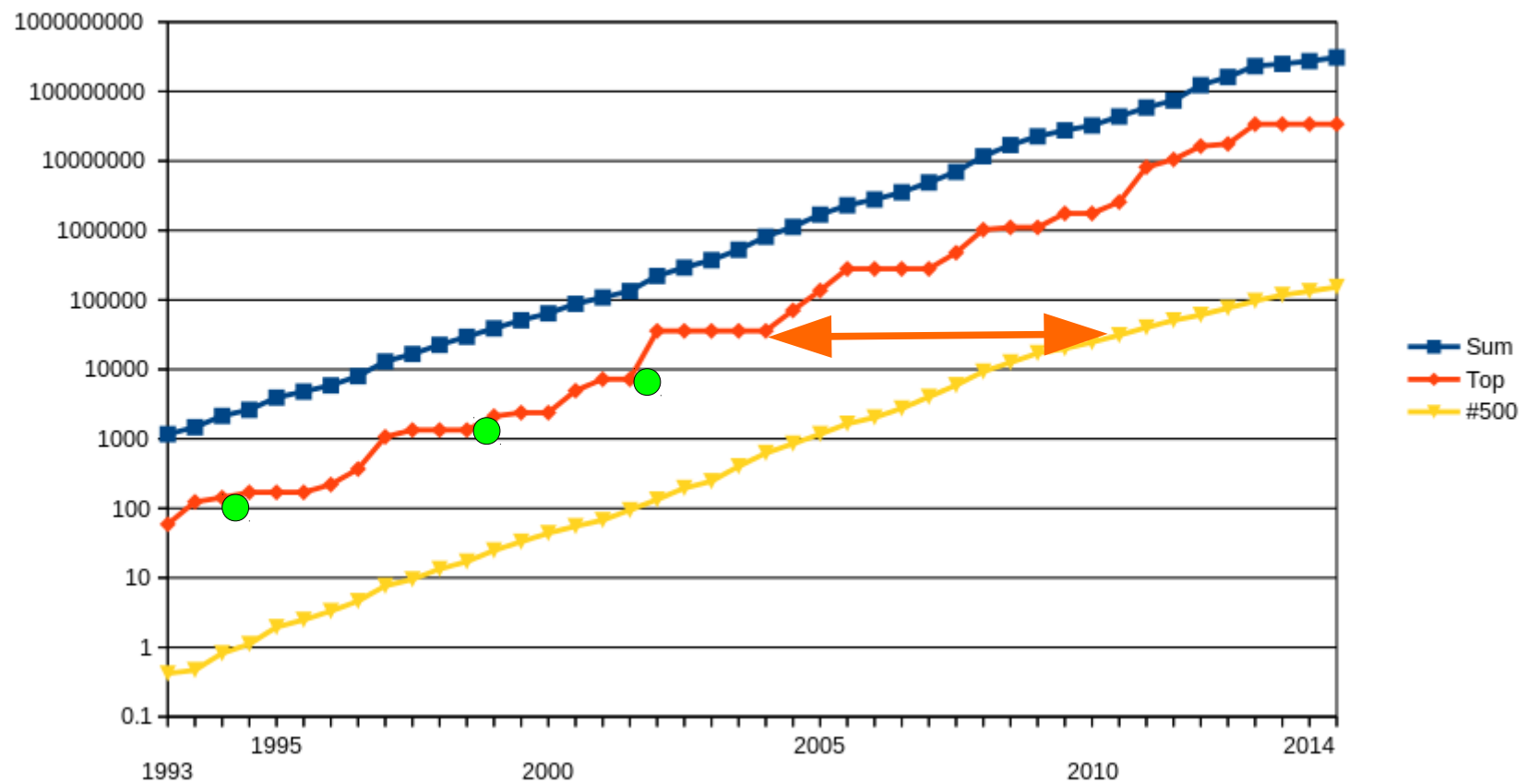
Trade secrets 1: the official view

The Top500 ranking list in the last 30 years !!!!



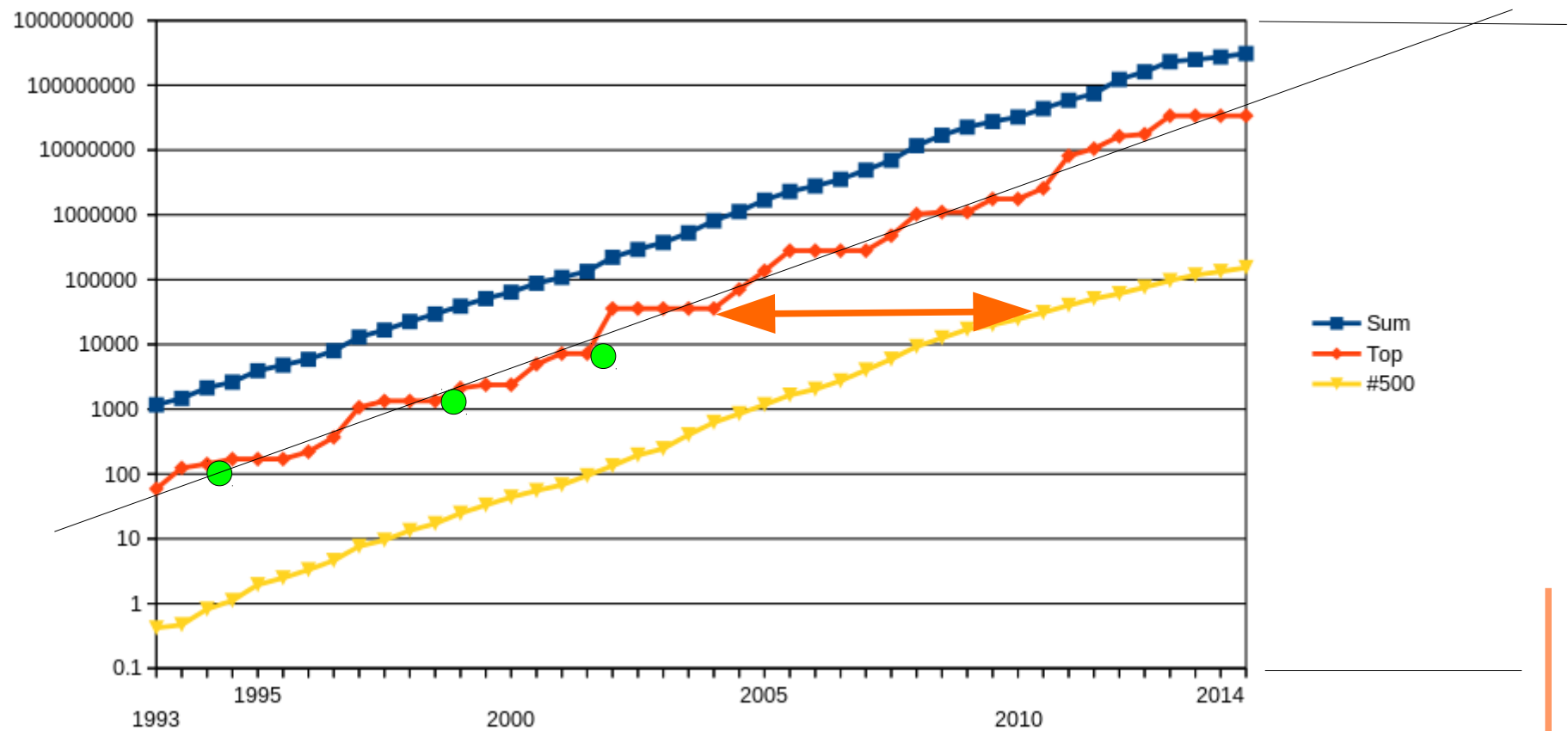
Trade secrets 1: the official view

The Top500 ranking list in the last 30 years !!!!



Trade secrets 1: the official view

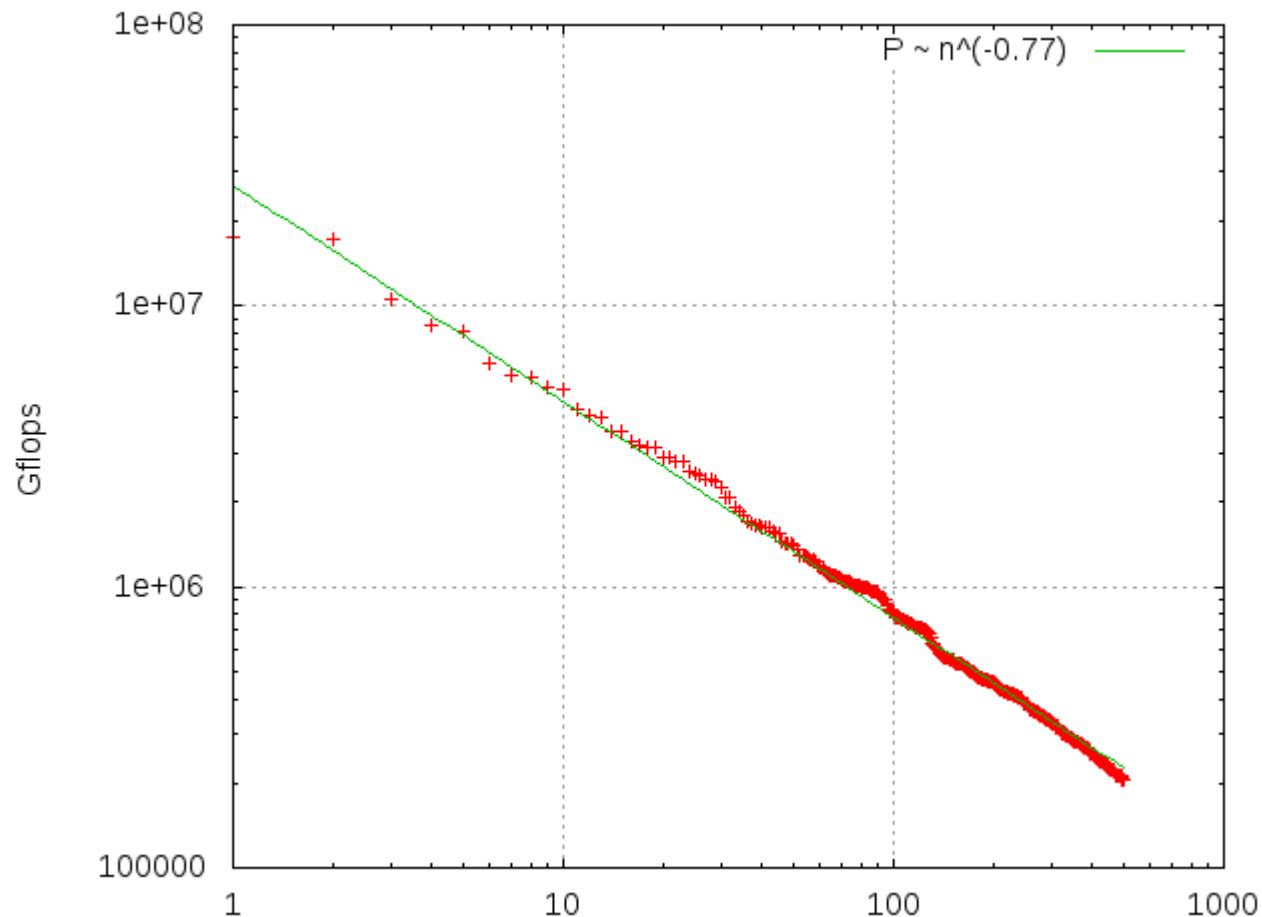
The Top500 ranking list in the last 30 years !!!!



Trade secrets 2: Less official views

Why 500 ????? Or ...

... Is being a club member a true sign of distinction??



Trade secrets 2: Less official views

Entry # 10 is the true boundary between the” haves” and the “haves not”....

#18 ENI

(3.188 P / 4.605P)

#36 CINECA (Fermi)

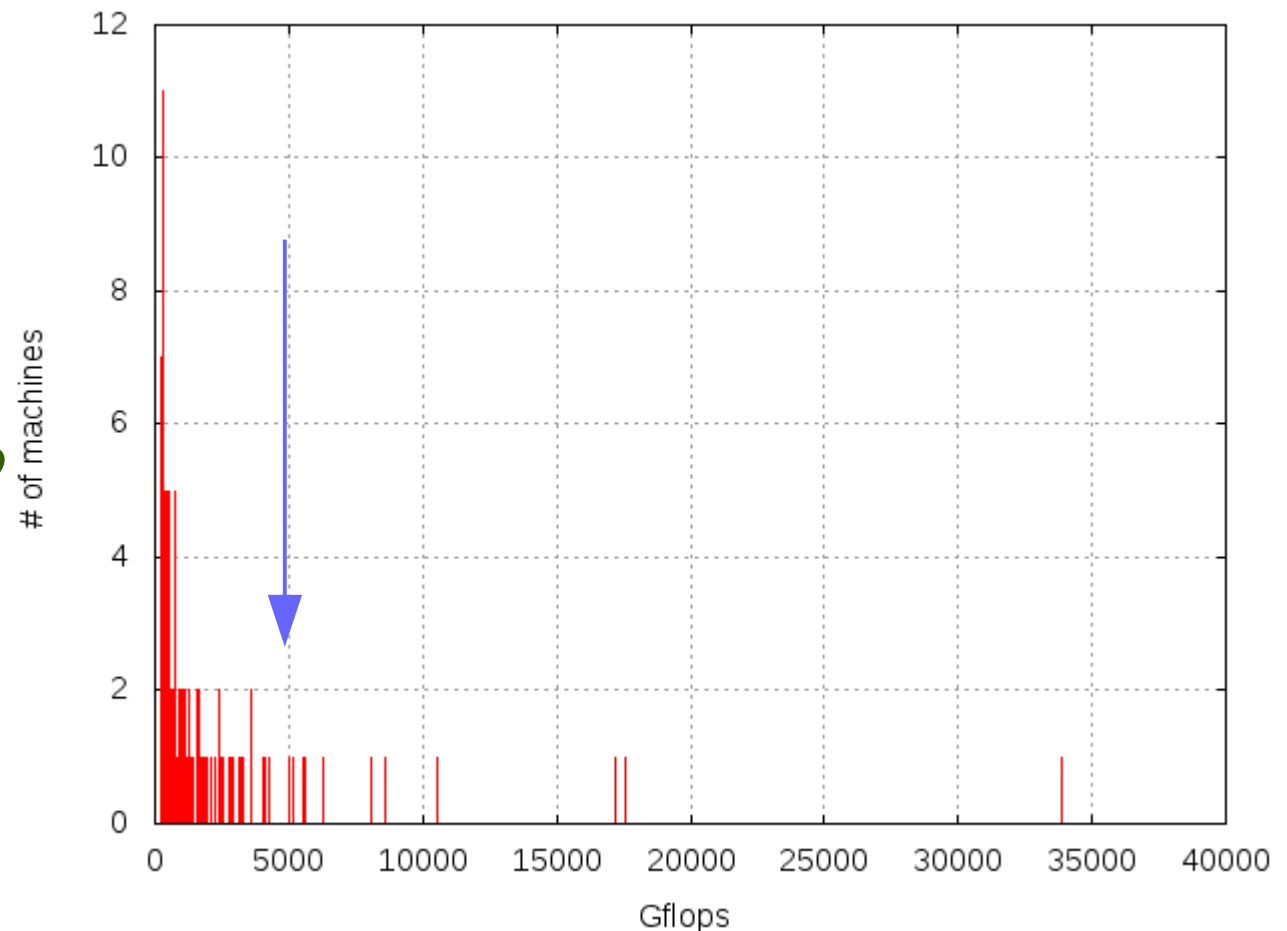
(1.789P / 2.097P)

#129 CINECA Galileo

(0.684P / 1.103P)

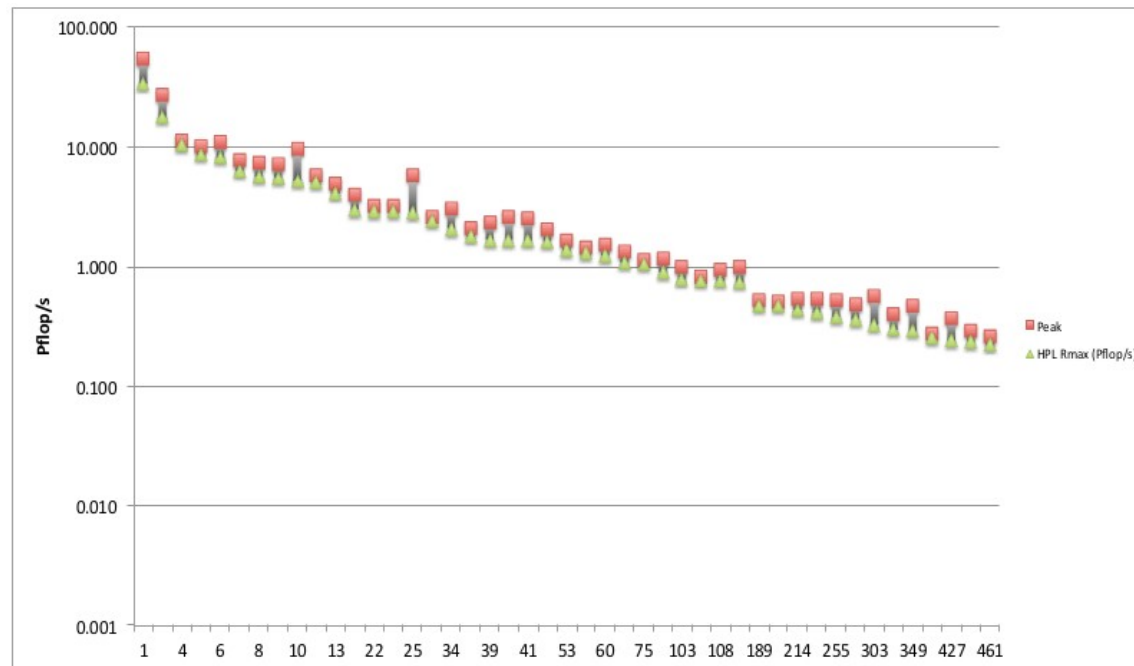
#205 ENI

(0.454P / 0.496P)



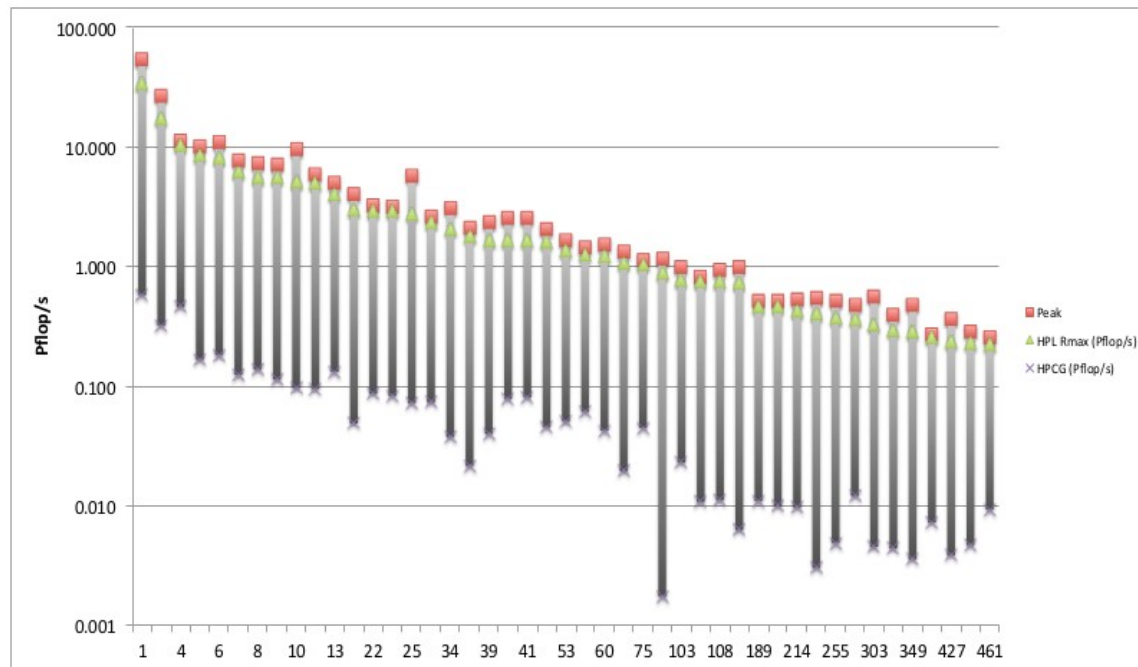
Trade secrets 3: Less well known goodies

*By the way, is the Linpack TOP500 test a sensible benchmark ??
Make your choice !!!!!*



Trade secrets 3: Less well known goodies

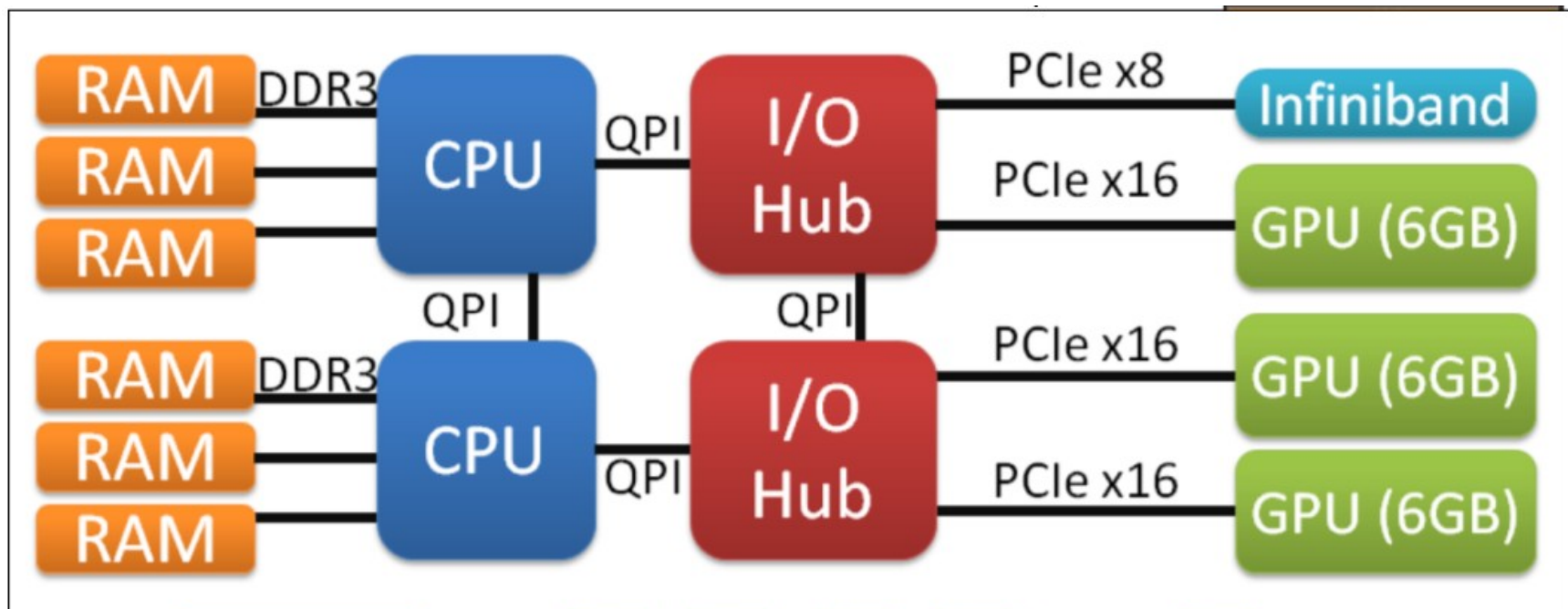
*By the way, is the Linpack TOP500 test a sensible benchmark ??
Make your choice !!!!!*



*Applying the “new” proposed Linpack has a sobering effect, but
the scaling law does not change at all (except for just a few
entries)*

Trade secrets 3: Less well known goodies

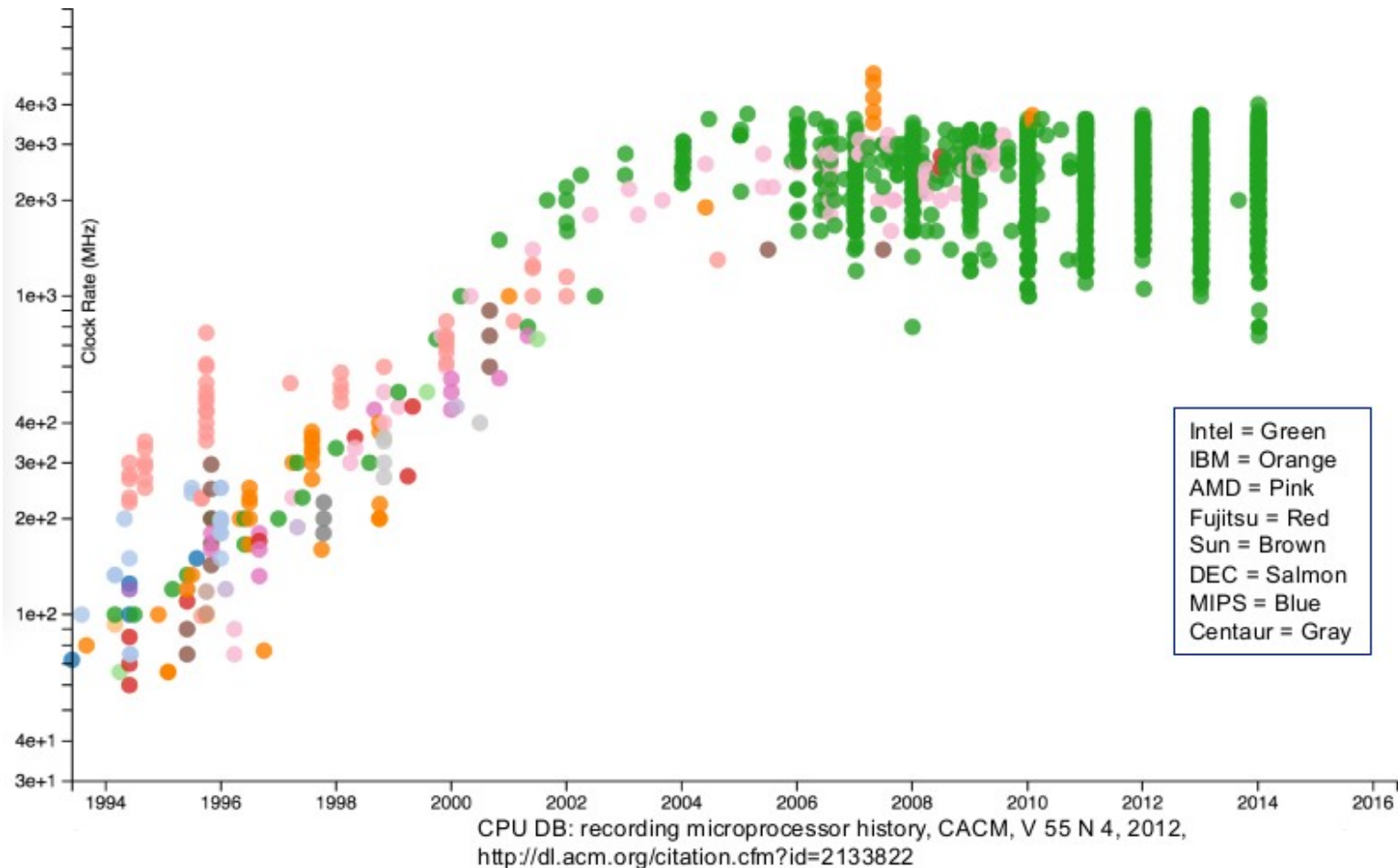
More and more high-end machines look like this



.... heterogeneous systems exploiting (in one way or another) massively parallel accelerators (GPUs or similar beasts)

Trade secrets 3: Less well known goodies

This is so because boosting the clock no longer works.....



Trade secrets 3: Less well known goodies

By the way, physics explains why it is so....

parallel computing is the physics sponsored way to compute:

The basic object in computers today is the transistor

*Industry learns to build smaller and smaller transistors. As $\lambda \rightarrow 0$
obviously $N \propto 1/\lambda^2$ but speed scales less favourably $f \propto 1/\lambda$*

*Trade rules: perform more and more things in parallel
rather than a fixed number of things faster and faster*

The INFN way ... to HPC

1) Agreements with a “large computing centre” (CINECA)

Access to 100 (out of 1600) Mcore-hours / year on the FERMI system (BG/Q); started in late 2012 ending in June 2016

Access to an additional 15 (x2) Mcore-hours / year on GALILEO 2015 through 2017

*Access to ~ 6% of the new MARCONI system (see later...)
Starting June 2016, ending November 2018*

After Nov. 2018, ??????

The INFN way ... to HPC

2) A smaller INFN-maintained cluster (Zefiro) for algorithm / code development, tests, fine-tuning of programs ...

*Some 1600 computing cores (~1.4 Mcore-hours)
Infiniband QDR + Experimental nodes (GPUs)*

up and running in Pisa since Sept. 2013



The INFN way ... to HPC

3) Probably early 2017: The first significant investment in HPC since the previous millennium ...

*In the framework of the CIPE project on “Integration of HPC and HTC computing”, approved in late December 2015;
“Decreto” published a few days ago →*

*an HPC island of roughly 1.5 Pflops peak power --->
1000 Mcore-hours / year (BG / Q equivalent)*

plus (probably) an upgraded “R&D small” machine

The INFN way ... to HPC

4) Are smaller installations sensible choices??

*Just an example of a smaller HPC machine
at Università di Ferrara*

*5 Nodes with 2 Intel CPUs and 16 GPUs
on each node (assembled by E4)*

*→ 110+ Tflops peak performance
(5.5 Gflops / W)*

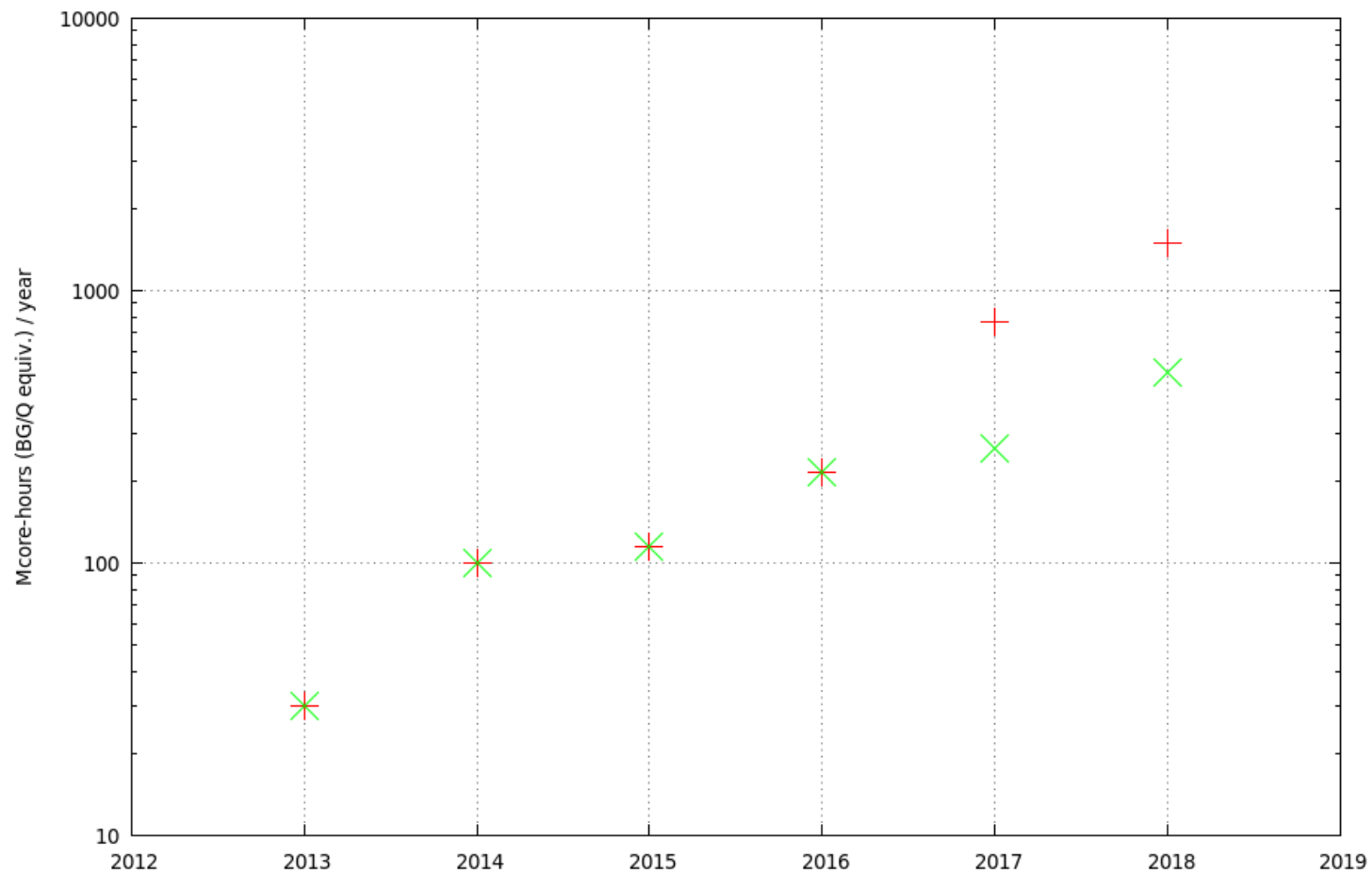
*→ 96 (BG / Q) Mcore hours GOOD!!!!
(4 years later...)*

*→ But only if you can use GPUs efficiently
... LESS GOOD!!*



The INFN way ... to HPC

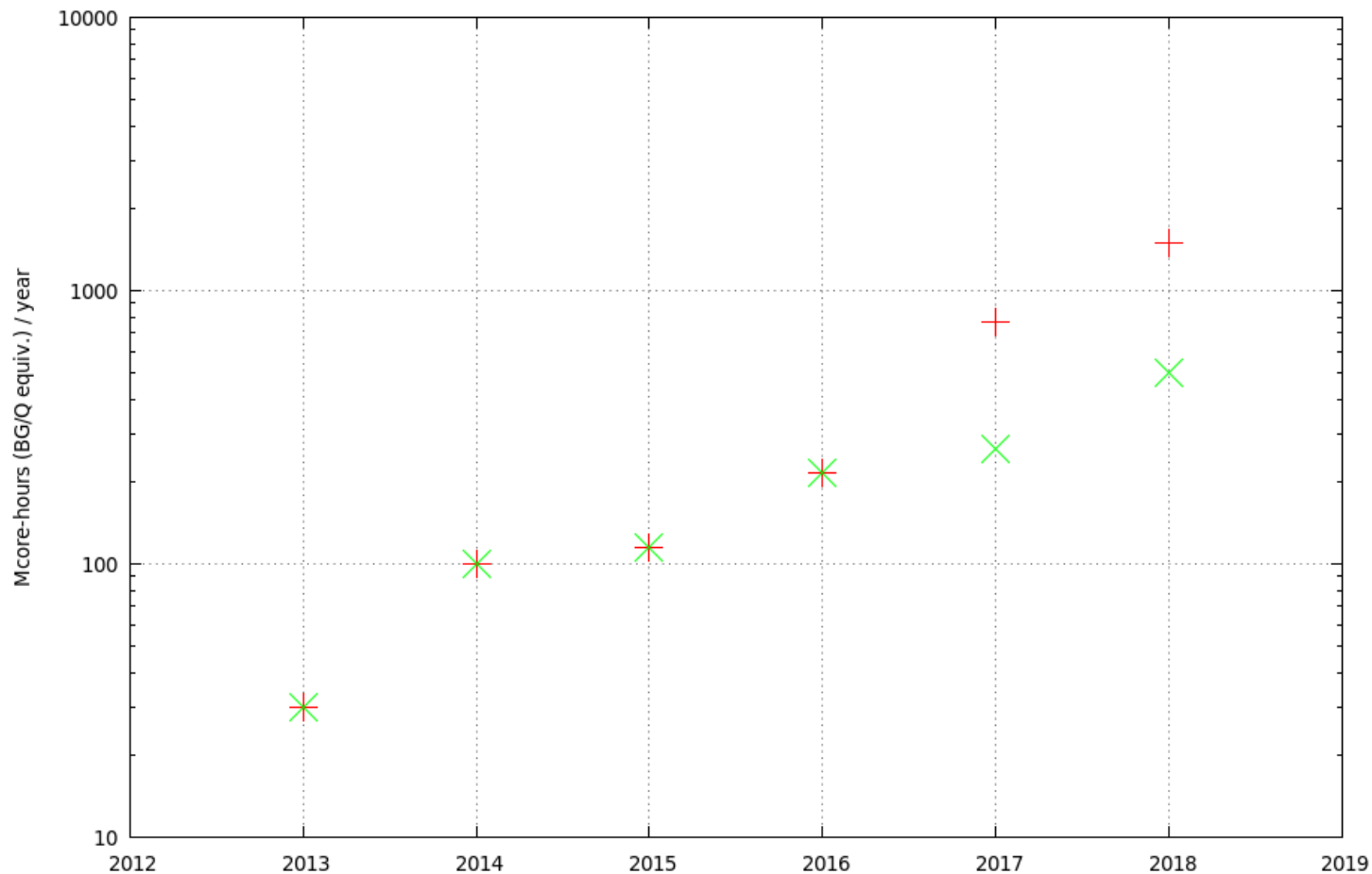
A new Moore's law ????



The INFN way ... to HPC

*A new Moore's law (**Zoccoli's law**) ????*

Still some worries ...



A fashionable question: WhatNext ??



WhatNext: Fermi → Marconi

A large upgrade scheduled in 2016 – 2017 for the Tier-0 machine at CINECA

A preview of a trend likely to be seen elsewhere

A long installation process

Three successive phases, going from 2 Pflops → ~ 18 Pflops

Approximately 26 ME for the full upgrade (~1.4 MEuro / Pflops)

WhatNext: Fermi → Marconi

A summary of the main figures

	Fermi	Marconi 1	Marconi 2	Marconi 3
When	Oct. 2013	Jun. 2016	Dec. 2016	Jun. 2017
Processor	B/G	Broadwell	KNL B1	Skylake
Cores/node	16	36	68	40
Perf. / core	12.5 Gflops	36.7 Gflops	44.9 Gflops	74.4 Gflops
Mem/cores	1 Gbyte	3.55 Gbyte	1.4 Gbyte	4.8 Gbyte
# of cores	160000	54432	244800	60480
Peak Flops	2.1 Pflops	2.0 Pflops	11.0 Pflops	4.5 Pflops
Energy eff.	3.3 Gflops/W	Valore medio	--->	12 Gflops/W
Node bandwidth	20 GB/sec	25 GB/sec	25 GB/sec	25 GB/sec
Perf / Bwidth	10 Flops / byte	51 Flops / byte	122 FI/byte	119 FI/byte

WhatNextNext:

The next big step forward might be the Coral project:

<i>2 GPU based machines (IBM)</i>	<i>Argonne + LLL</i>
<i>1 MIC based machine (Cray)</i>	<i>OakRidge</i>

Promised for (late) 2017

Each machine in the 100 – 200 Pflops range

The two IBM machines cost 325 M\$ (1.3 M\$ / Pflops)

A closer look at the Argonne machine (codename: Summit)

WhatNextNext:

The Summit / Sierra systems are the first attempt at designing from scratch an HPC GPU-based system ...

... as opposed to just assembling whatever processor with whatsoever GPU ...

... trying to eliminate (or at least reduce) the two critical bottlenecks of these “accrocchi”:

Data bottleneck among memory and compute engine

Data bottleneck among GPU and CPU

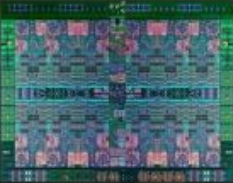
WhatNextNext:

	Fermi @ Cineca	Coral / Sierra
Cores/node	16	???
Perf. / node	0.2 Tflops	5 x 8 Tflops
Mem/node	16 Gbyte	256 Gbyte ????
# of nodes	10000	3400
Peak Flops	2.1 Pflops	~ 140 Pflops
Energy eff.	3.3 Gflops/W	~ 15 Gflops/W
Node bandwidth	20 Gbyte/sec	23 – 46 Gbyte/sec
Core bandwidth	1.25 GB/sec	
Perf / Bwidth	10 Flops / byte	860 Flops/byte


WhatNextNext:

A couple of critical architectural improvements here:

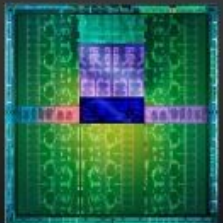
**Accelerated Computing
5x Higher Energy Efficiency**




IBM POWER CPU
Most Powerful Serial Processor



NVIDIA NVLink
Fastest CPU-GPU Interconnect

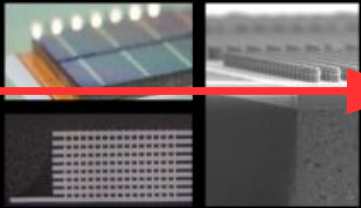


**VOLTA GPU Featuring
NVLINK and Stacked Memory**



NVLINK

- GPU high speed interconnect
- 80-200 GB/s



3D Stacked Memory

- 4x Higher Bandwidth (~1 TB/s)
- 3x Larger Capacity
- 4x More Energy Efficient per bit

160GByte / sec →

720 GByte / sec →

WhatNextNext:

A couple of critical architectural improvements here ...

But a serious network bottleneck coming back ???

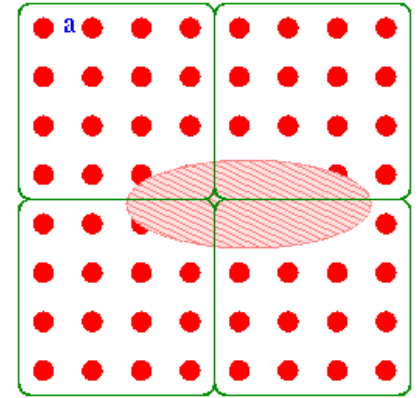
WhatNextNext:

A couple of critical architectural improvements here ...

But a serious network bottleneck coming back ???

YES and NO!!!

WhatNextNext:



Assume that:

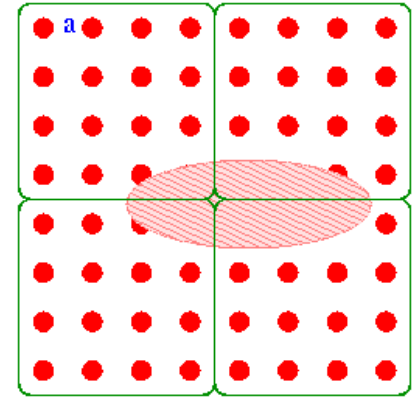
- *your pattern of node-to-node-communication is local, so ...*
- *the information exchange is proportional to the “surface” of your computing volume* $I \propto V^\epsilon$ $\epsilon < 1$
- *you use the increasing power of each processing node to stuff larger and larger parts of your problem into each processing nodes*
- *more formally, you are happy with “weak scaling”*

Define $\rho = P/B$

Then you can derive a scaling law telling that you’re happy as long as:

$$\rho \leq P^{1-\epsilon} \quad \epsilon \propto 0.5$$

WhatNextNext:



Example:

When computing the Dirac operator (the key computing kernel of LatticeQCD)

For each 4-D slice of your physical system of size L^4

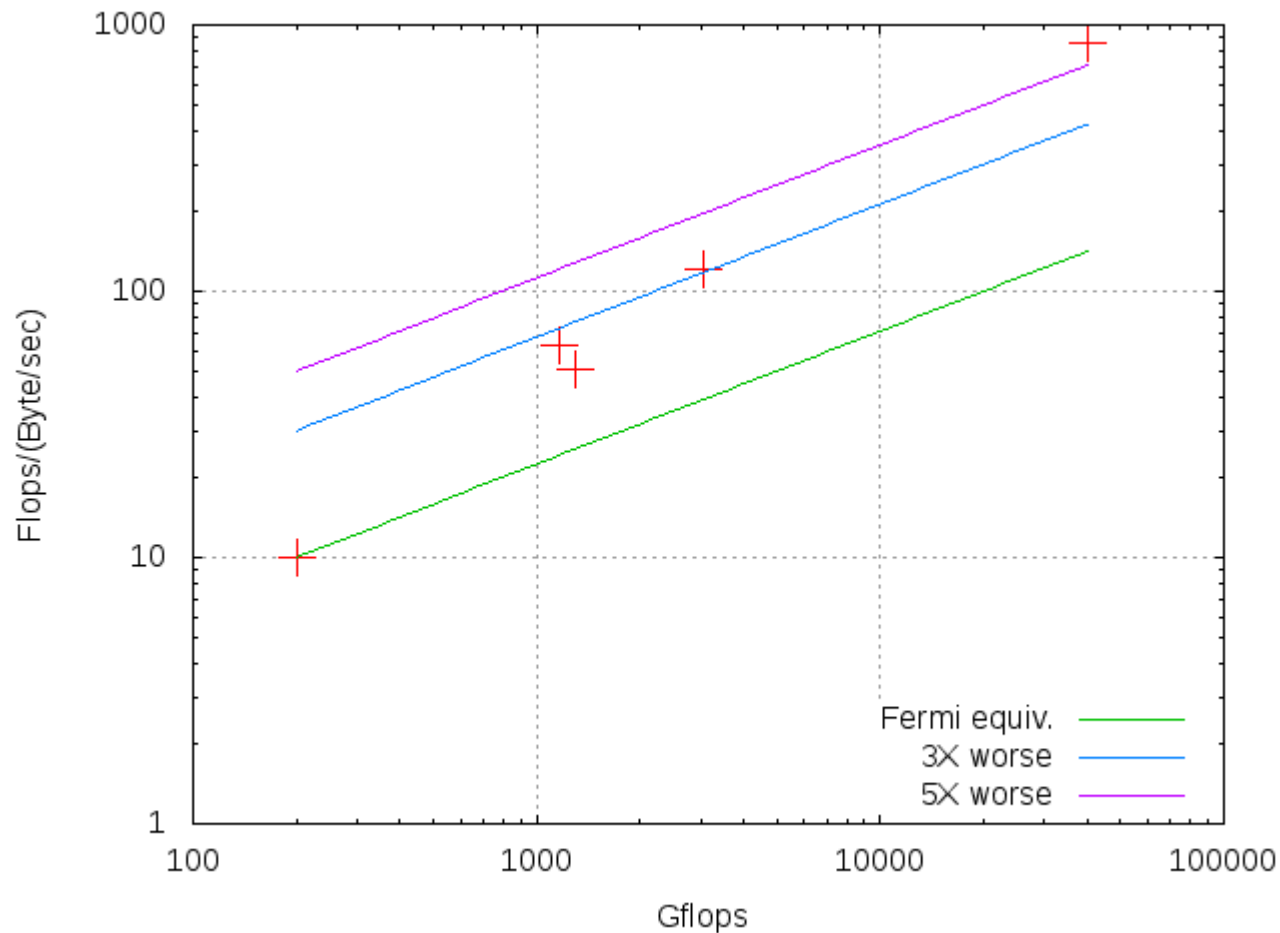
$C \sim 584 L^4$ ops $I \sim 8 \times 3 \times 8 L^3$ bytes

$$\rho \approx \frac{584 L^4}{8 \times 3 \times 8 L^3} = 3 L$$

(not too large) lattices of 24^4 (32^4) sites scale in performance on BG/Q up to 4096 (10000) cores

WhatNextNext:

It is not as bad as it looks like at the beginning



... but not very good either !!!!

WhatNextNextNext:

“La Cina e’ vicina.....”

China Accelerator

天河

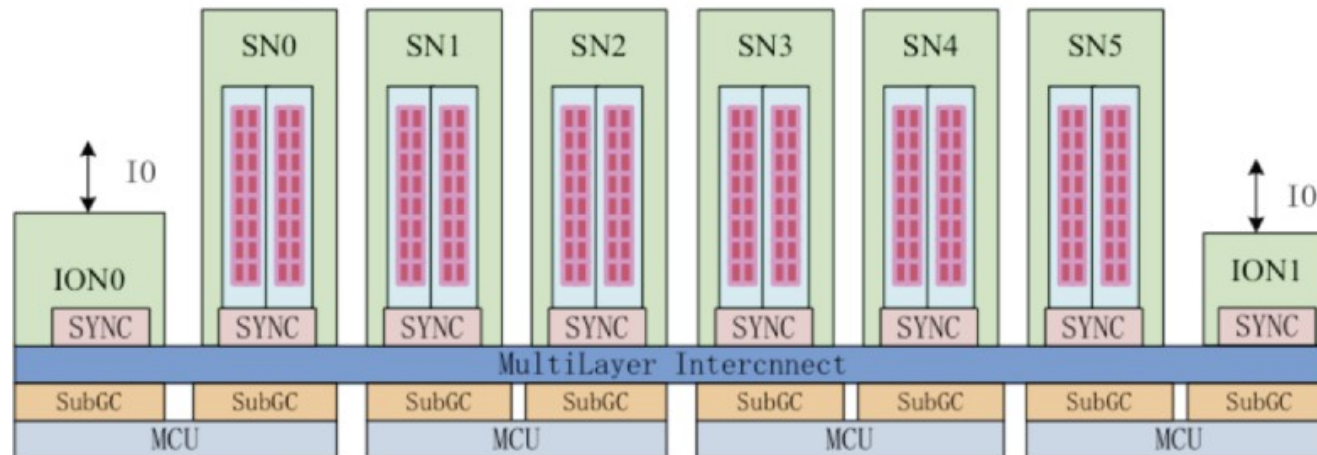
Matrix2000 GPDSP

□ High Performance

- 64bit Supported
- ~2.4/4.8TFlops(DP/SP)
- 1GHz, ~200W

□ High Throughput

- High-bandwidth Memory
- 32~64GB
- PCIE 3.0, 16x



国防科学技术大学
National University of Defense Technology



Programming trends for HPC

THE KEY PROBLEM:

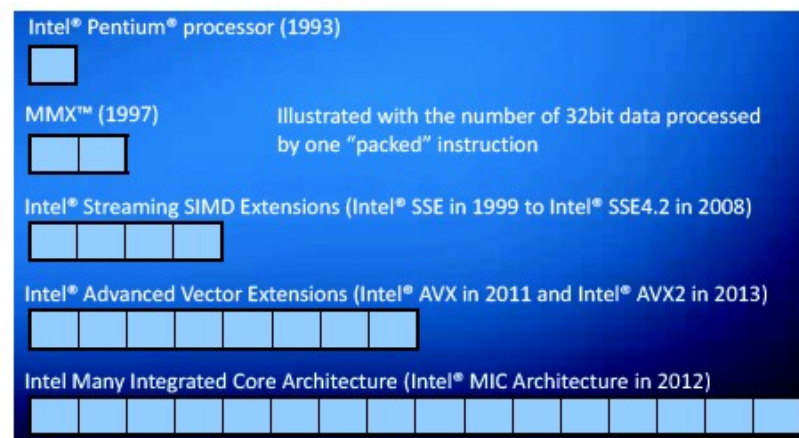
No way to increase performance if you do not go parallel

And you have to do so AT THE SAME time at many level

- 1. Coarse (node) parallelism*
- 2. Fine (core) parallelism*
- 3. Finer(intra-core) parallelism (aka, vectorization, SIMD)*

The “old” way...

- 1. MPI*
- 2. openMP*
- 3. ??????*



Programming trends for HPC

The “old” way...

- 1. MPI*
- 2. openMP*
- 3. ?????*

Is there a “new” way to:

replace ?????

possibly combine (at least) 2) and 3)

A possible approach to answer this question

Programming trends for HPC

*Are current massively-parallel processors efficient compute engines for our typical (massively-parallel) algorithms ...
... if you are ready to make an “unreasonable” effort to adapt your algorithm / code to the machine?*

Is there a way to squeeze a large fraction of the potentially available computing power with a “reasonable” effort?

Is this “reasonable effort” re-usable, as new processors / computers become available in the near future.

Can we have “(mildly-)quantitative” answers to these questions?

Programming trends for HPC

Direct experience in 2 real-life cases:

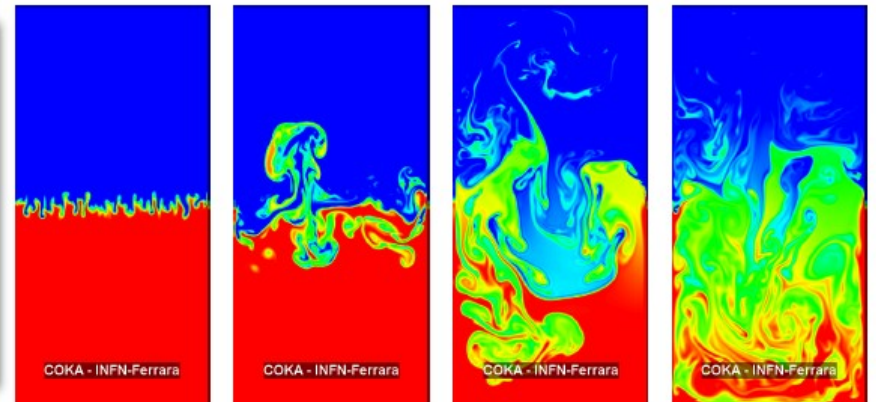
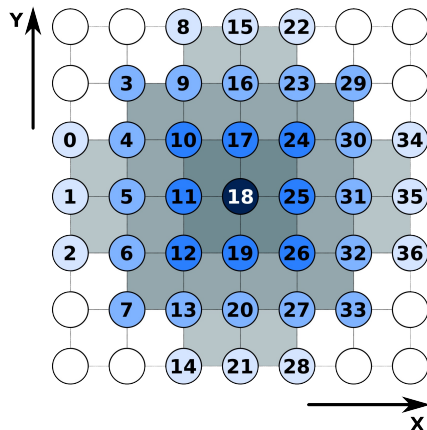
Fluid – dynamics using the Lattice Boltzmann (LB) method

LQCD with staggered fermions.

The LB method

LB is discrete in position and momentum space.

LB solves numerically the Boltzmann equation on a lattice with sufficient accuracy (i.e. up to a given power of momenta) to reproduce the features of the fluid-flow described by the Navier-Stokes equations ($\delta x \gg l$)



The LB method

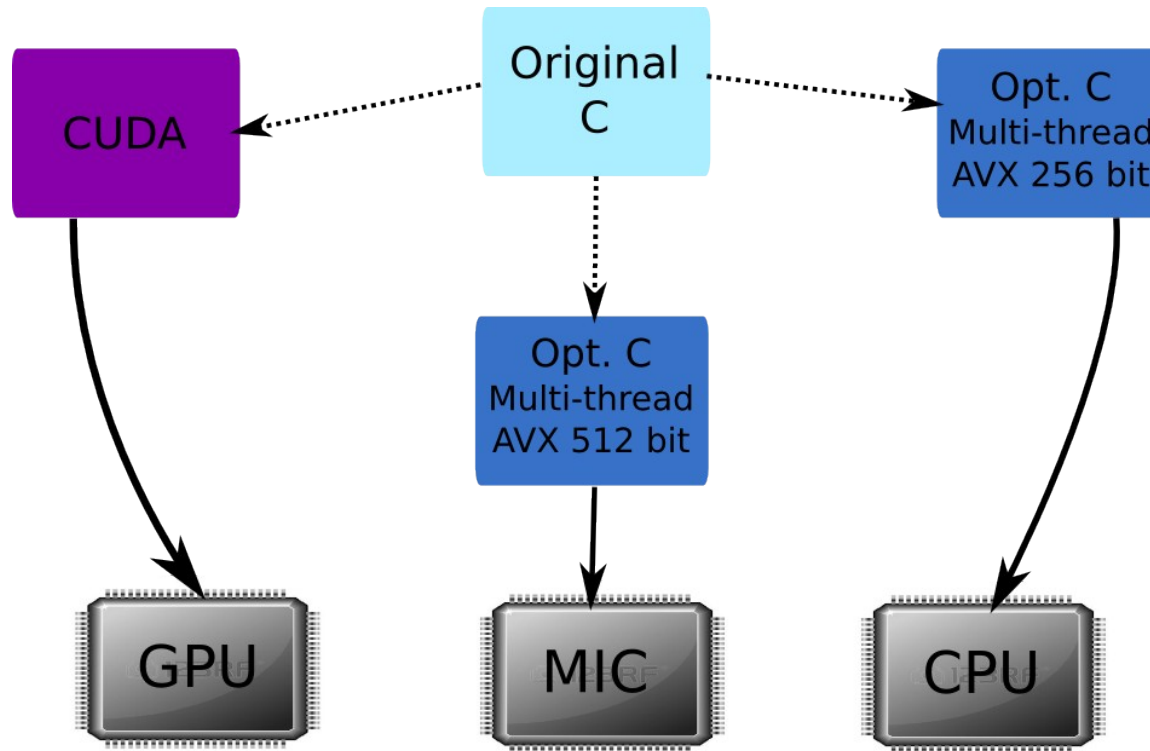
An extended set of tests on the D2Q37 LB model

Surprisingly similar to LGT simulations in many of its computational details.

- Massively-parallel algorithm, easily partitioned on many nodes*
- Just two computationally critical kernels (collide - propagate)*
- No significant role of global sums (at variance with LQCD)*

Programming trends for HPC

In practical terms: can we make a transition from here



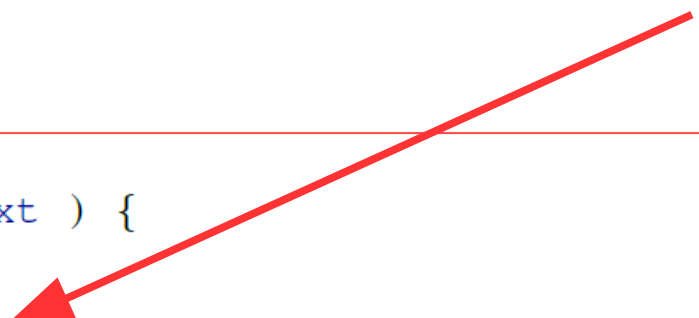
Our Graal: openACC

Is there a programming language that makes this perspective possible and efficient?

It looks like it does: it is called openACC (or maybe openMP4)

Let's have a look ---->

```
inline void propagate (  
    const data_t* restrict prv, data_t* restrict nxt ) {  
    int ix, iy, site_i;  
    #pragma acc kernels present(prv) present(nxt)  
    #pragma acc loop gang independent  
    for ( ix=HX; ix < (HX+SIZEX); ix++) {  
        #pragma acc loop vector independent  
        for ( iy=HY; iy<(HY+SIZEY); iy++) {  
            site_i = (ix*NY) + iy;  
            nxt[      site_i] = prv[      site_i-3*NY+1];  
            nxt[NX*NY+site_i] = prv[NX*NY+site_i-3*NY  ];  
            ...  
        } } }  
}
```



Our Graal: openACC

How much do we lose in performance, if we are only ready to make a reasonable effort?? (look only at the line with )

Code Version	Tesla K40			Tesla K80		
	CUDA	OCL	OACC	CUDA	OCL	OACC
T_{Prop} [msec]	13.78	15.80	13.91	7.60	7.79	7.51
GB/s	169	147	167	306	299	310
\mathcal{E}_p	59%	51%	58%	64%	62%	65%
T_{Bc} [msec]	4.42	6.41	2.76	1.11	1.58	0.71
T_{Collide} [msec]	39.86	136.93	78.65	16.80	61.73	36.39
MLUPS	99	29	50	234	64	108
\mathcal{E}_c	45%	13%	23%	52%	14%	24%
$T_{\text{WC/iter}}$ [msec]	58.07	159.14	96.57	26.84	71.12	44.61
MLUPS	68	25	41	147	55	88



Our Graal: openACC

Once we have our nice openACC code, can we use it (without changes) on a different (yet a new) computer????

	E5-2630 v3			GK210			Hawaii XT	
compiler model	ICC 15 Intrinsics	ICC 15 OMP	PGI 15.9 OACC	NVCC 7.5 CUDA	NVCC 7.5 OCL	PGI 15.9 OACC	GCC OCL	PGI 15.9 OACC
<i>propagate</i> perf. [GB/s]	38	32	32	154	150	155	232	216
\mathcal{E}_p	65%	54%	54%	64%	62%	65%	73%	70%
<i>collide</i> perf. [MLUPS]	14	11	12	117	32	55	76	54
<i>collide</i> perf. [GFLOPs]	92	71	78	760	208	356	494	351
\mathcal{E}_c	30%	23%	25%	52%	14%	24%	19%	14%
Tot perf. [MLUPS]	11.5	9.2	9.8	80.7	28.1	45.6	63.7	47.0



From LB to LQCD

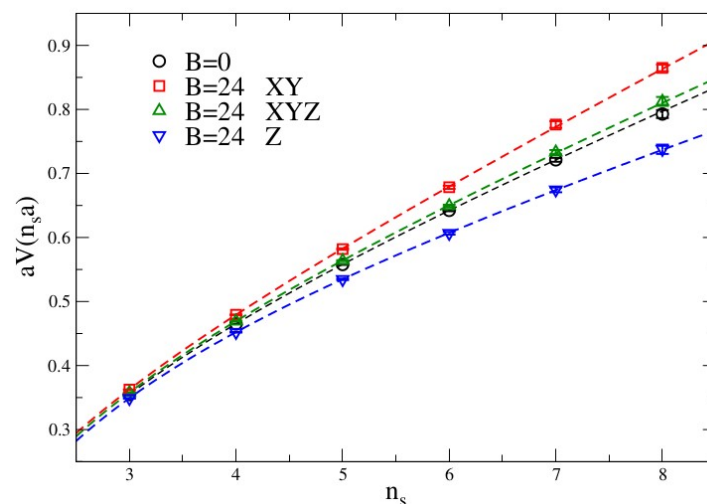
A LQCD code with staggered fermions + improved action + (two level) stout smearing

*earlier moved to GPUs by M. Delia and collaborators
... and now going back to good old C + openACC*

The physics problem: what happens to the quark potential in an external magnetic field

(C. Bonati et al., arXiv:1403:6094)

$(40^4) \rightarrow (48^3 \times 96)$



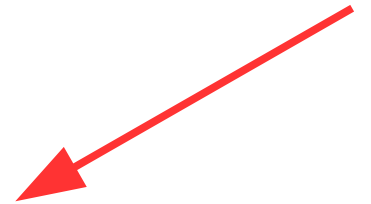
From LB to LQCD

*Code fully (re-)written and tested (C + openACC)
/Running physics on the GPU cluster in Ferrara)*

Preliminary performance tests on K20 – K40 – K80 GPUs

Comparison against the golden benchmark today (BG / Q)

	<i>Measured</i>	<i>Expected</i>
<i>1 K20 GPU</i>	<i>→ 80 BG / Q cores</i>	<i>(93)</i>
<i>1 K40 GPU</i>	<i>→ 96 BG / Q cores</i>	<i>(110)</i>
<i>1 K80 GPU</i>	<i>→ 160 BG / Q cores</i>	<i>(190)</i>
<i>1 “heavy” GPU node</i>	<i>→ 1300 BG / Q cores</i>	



Is it all gold that glitters??

Obviously not

Some new processors not yet supported by openACC / openMP4

Non negligible (but acceptable) performance gap still present

Not all programs automatically OK...

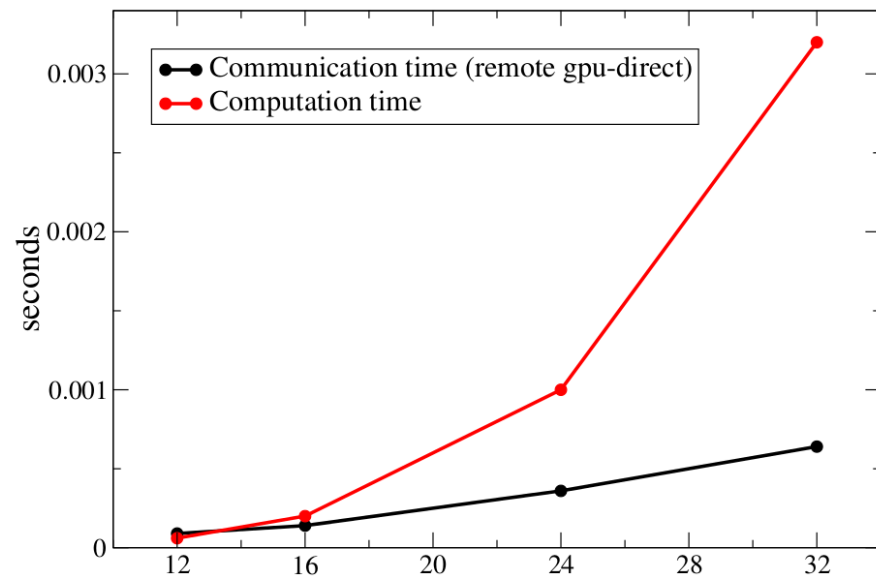
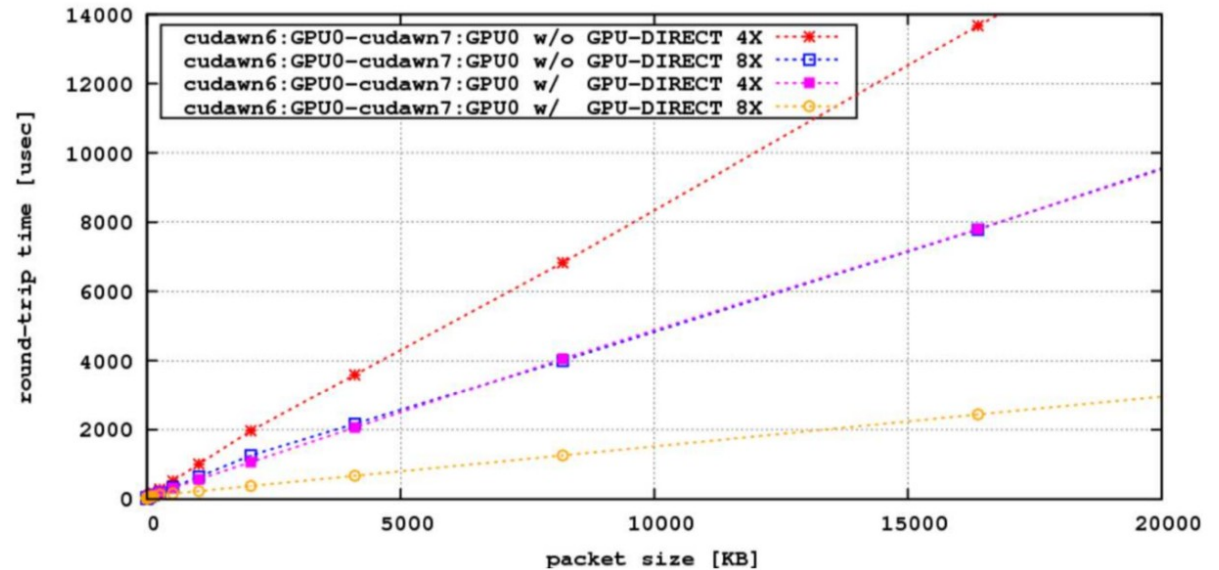
And ...

The problem in the next few years will again be in the match between processing power and communication capacity among compute nodes (Nothing as good as BG/X will be available in the near future....)

Is it all gold that glitters (2) ??

Still, after some (non trivial) tuning is done (once for all) ...

Reasonable windows for efficiency can be found



Concluding remarks (1)

HPC differs from HTC because the inter-node(core) communication requirements are hugely different

HPC machines happily increase their peak performance levels ...

... but interconnection technology is barely adequate to keep them working on useful problems

In the future HPC and HTC machine may look increasingly similar

Competitive systems are not within reach of INFN alone: agreements with other partners necessary

Concluding remarks (2)

Processing elements are increasingly powerful, but also complex and heterogeneous, and (luckily??) no clear winner has emerged.

Programming (for performance) in a machine-independent way is difficult.

But recent progress in this direction is promising.

And is probably the key place in which collaboration among the theoretical and experimental communities within INFN can be very fruitful.