“Easy, maybe even possible!”: Giorgio and computing

R. Tripiccione

Dipartimento di Fisica & INFN, Ferrara

tripiccione@fe.infn.it

Giorgio Parisi's Fest

Rome, September 9th 2008

università di ferrara

DA SEICENTO ANNI GUARDIAMO AVANTI.

Outline

This talks is on CATs (borrowing a term coined by J. Kogut & M. P. Lombardo)

or (even better) super-CATs ---> Computers help understand physics ....
...physics helps shape better computers.

Monte Carlo engines for LGT and for statistical systems
why ?? how ??

History and (conflicting) trends through 2 recent examples
QPACE JANUS

Take-away lessons & concluding remarks

How/when I first met Giorgio ...

In the early eighties we were working hard to get the string tension (measured by Wilson loops) to have a decent scaling behavior ...

$$W(T,R) \sim \exp(T\,V(R)) \qquad \ln(W(T,R)) \sim T\ L$$

$$V(R) = \sigma R + \ldots$$

we struggled to subtract contributions hiding away the string tension

replace $W(T,R) \Rightarrow W(T,R) - W^{(2)}_{pert}(T,R)$

fit $V(R) = \sigma R + \dfrac{\alpha}{R}$

One day, the bad news came: "Giorgio Parisi does not LIKE this ....."

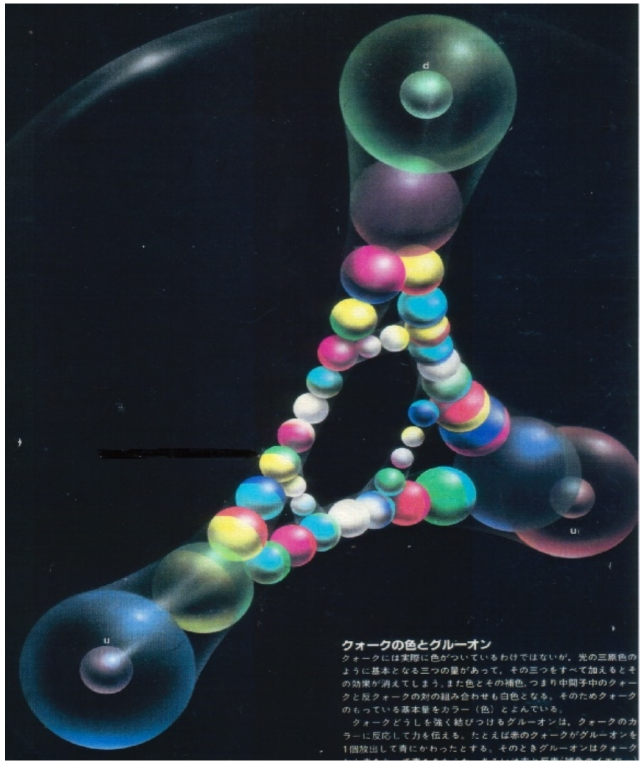
Giorgio??

I started to develop my own private mental picture of what Giorgio had to look like ....

Giorgio

me ...

Lattice Quantum Chromo-Dynamics



クォークの色とグルーオン

クォークには実際に色がついているわけではないが，光の三原色のように基本となる三つの量があって，その三つをすべて加えるとその効果が消えてしまう。また色とその補色，つまり中間子中のクォークと反クォークの対の組み合わせも白色となる。そのためクォークのもっている基本量をカラー（色）とよんでいる。

　クォークどうしを強く結びつけるグルーオンは，クォークのカラーに反応して力を伝える。たとえば赤のクォークがグルーオンを1個放出して青にかわったとする。そのときグルーオンはクォーク

Unfortunately it is not yet known whether the quarks in Quantum Chromodynamics actually form the required bound states. To establish whether these bound states exist one must solve a strong coupling problem and present methods for solving field theories don't work for strong coupling.

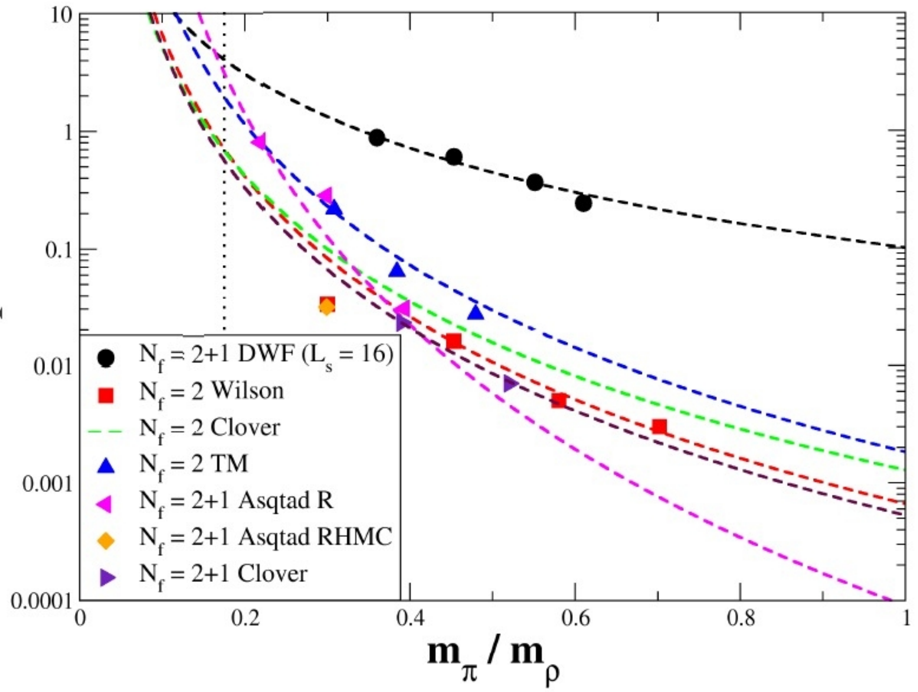
K. Wilson, Cargese Lectures, 1976

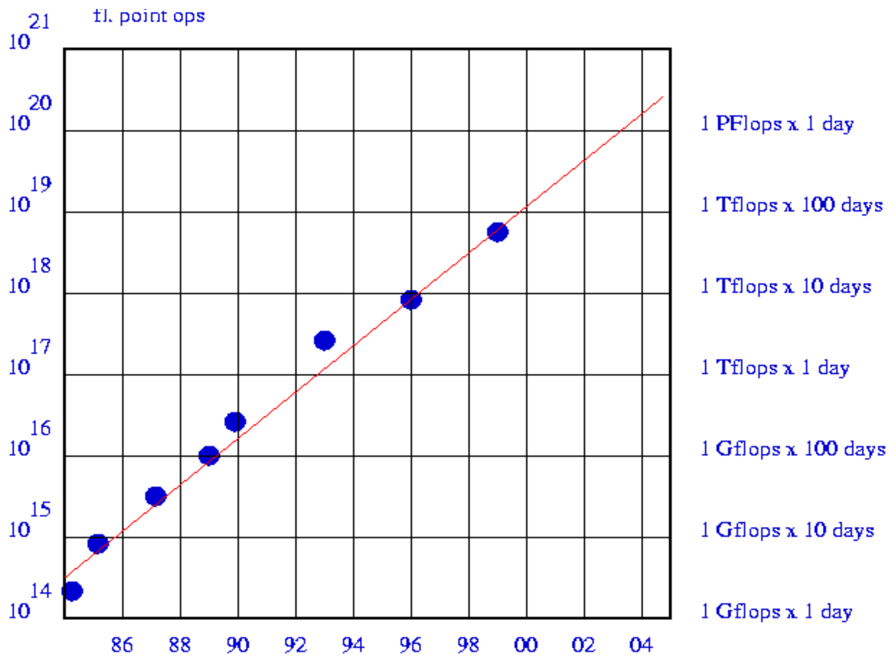
Here is the problem ...

L = 2 fm, a = 0.08 fm

1000 configs

M. Clark, LATTICE2006





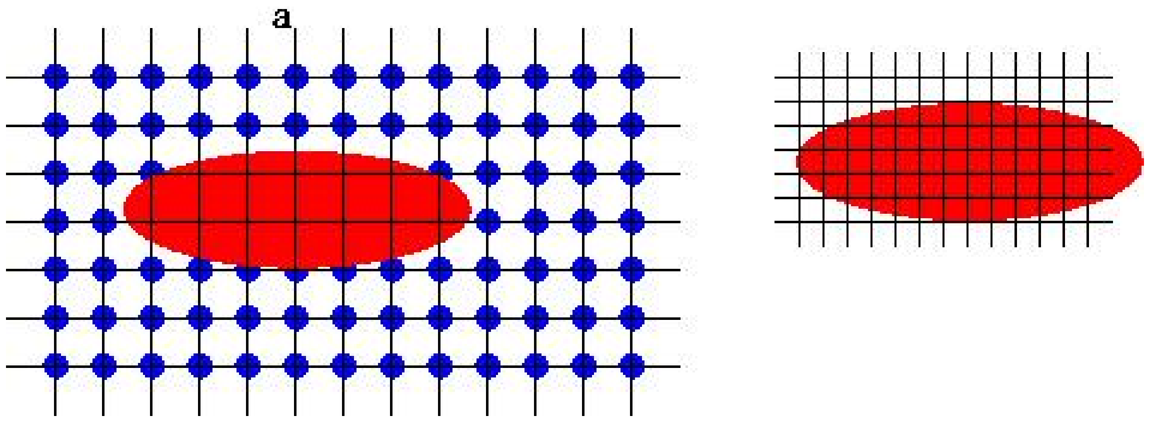
from an early graph

by N. Christ

More quantitatively

$$N_{flop} \sim \; L^{5\ldots6} \times (1/a)^{6\ldots7} \times (1/m_q)^{1\ldots2}$$

… need to take into account all relevant scales of the problem



Spin Glasses

Here one may want to study e.g., the phase structure of the model, or the dynamics of a large system (e.g., 80^3 sites) in the low T phase:
a surprisingly challenging computational problem:

O(10^{12-14}) time steps on ~ 100 sampless of the system -> 10^{20-22} updates

Clever programming on PCs: 1 ns / spin-update --> centuries of wall clock time
HOWEVER:
Embarassingly (as well as non-embarassingly)-easy parallelism can be used to do better by orders of magnitude....

very simple arithmetics ---> <u>simple & easy</u> and quick

One has to be optimistic ...

In most cases, computing accurate predictions of the behaviour of a complex physical system is hopeless, unless numerical techniques are used

One has to be optimistic ...

In most cases, computing accurate predictions of the behaviour of a complex physical system is hopeless, unless numerical techniques are used

However Nature has been friendly to us

One has to be optimistic ...

In most cases, computing accurate predictions of the behaviour of a complex physical system is hopeless, unless numerical techniques are used

However Nature has been friendly to us

so the (simple) physics laws behind the behaviour of computers make it easy to build machines optimized for the simulation of complex physics systems!!!!

Better computers than those you can buy?

Fine: you need a lot of computing power …

… Why on earth do you think you can do better than an established computer company?

Three answers to this question:

1) What we need is not exactly what traditional computers have been good at

2) What we need is very simple to achieve in terms of computer architecture …

… if we proceed in the direction that basic physics laws point to us

Better computers than those you can buy?

Answer 0:

i) we have known (almost) exactly what we want to do for 20 years

ii) it has not changed too much in this time frame

iii) it is just one thing: solve $M(U)x=y$

Krylov-space methods, polynomial approximations ...

Pre-conditioning: SSOR,, Schwartz alternating procedure ....

Routine	**Calls**	**Time**	**Cma**	**Code-lines**
Dirac operator (3 variants)	80844	58.00%	350	O(2000)
Linear algebra (3 routines)	60736	26.00%	100	O(1000)
Gauge force + update	320	8.00%	2000	O(2000)
Global sum (4x8x8 nodes)	83554	0.40%	20	O(200)
Others (~70 routines)		7.00%		O(15000)

alpha-code on apeNEXT (thanks to H. Simma)

Better computers than those you can buy?

1) What we need is not exactly what traditional computers have been good at

Either we need long straight sequences of mostly floating point operations

e.g.: $1/\sqrt{r^2}$ *(stellar dynamics)*

or: $a \times b + c$ *among complex numbers (LQCD)*

or conversely, we need long straight sequences of *<u>extremely simple logic clauses</u>*

$$\sum_{NB(ij)} \sigma_i J_{ij} \sigma_j$$

Better computers than those you can buy?

2) What we need is very simple to achieve in terms of computer architecture
Basic physics help us in two ways:

1) Parallel computing is trivially possible in all cases …
… and parallel computing is the physics sponsored way to compute:

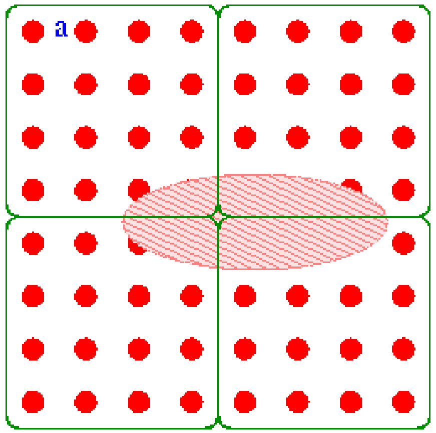
The basic object is the transistor
Industry learns to build smaller and smaller transistors. As $\lambda \to 0$
obviously $N \propto 1/\lambda^2$ *but speed scales less favourably* $\tau \propto \lambda$

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

Better computers than those you can buy?

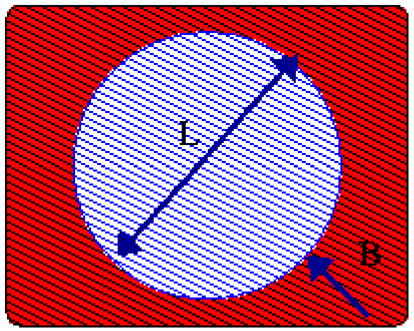
Basic physics helps us in two ways:

2) We are interested in modeling local theories:
This has to go over to the computer structure ->
Keep data close in space to where it is processed



Failure to do so will asymptotically
bring a data bottleneck:

$$B(L) \propto L$$
$$P(L) \propto L^2$$

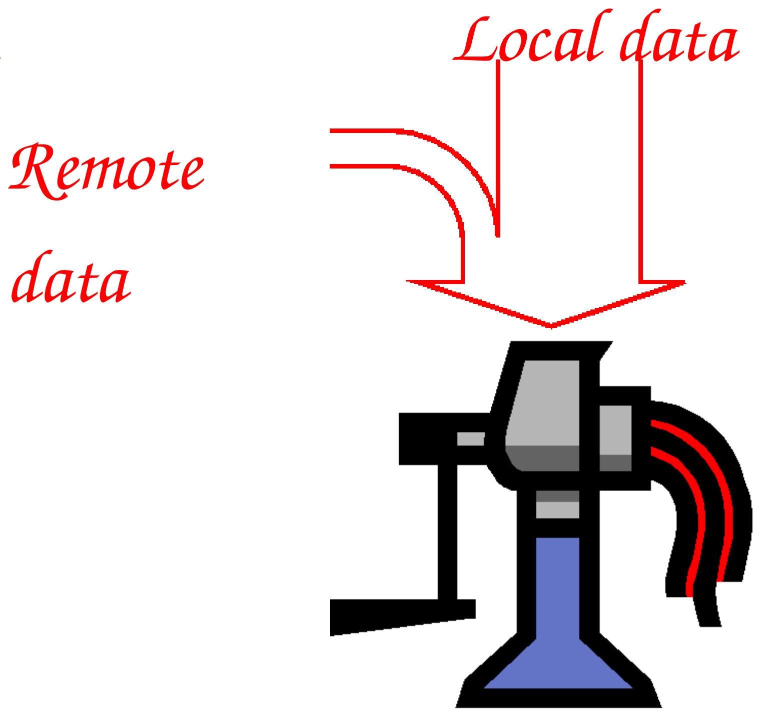


Simulation engines in a nutshell

The quantitative approach:

a simulation engine is a set of sausage-machines: it crunches pork-meat (numbers) coming mainly from its local store as well as data coming from nearby processors.

Computational power must be balanced by a matching flow of data into the processor

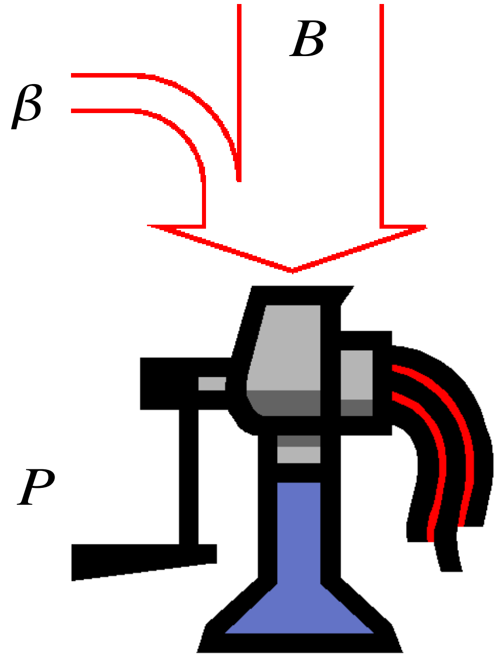


Simulation engines in a nutshell

Computational power must be balanced by sufficient flow of data coming into the processor.

Appropriate parameters:

$n\ P$	*aggregate processor computing power (n processors working together)*
R	*ratio of operations/data words*
B	*memory band width (to local pork-meat stock)*
ρ	*ratio of local/remote words*
β	*memory bandwidth (to remote pork-meat stock)*



Simulation engines in a nutshell

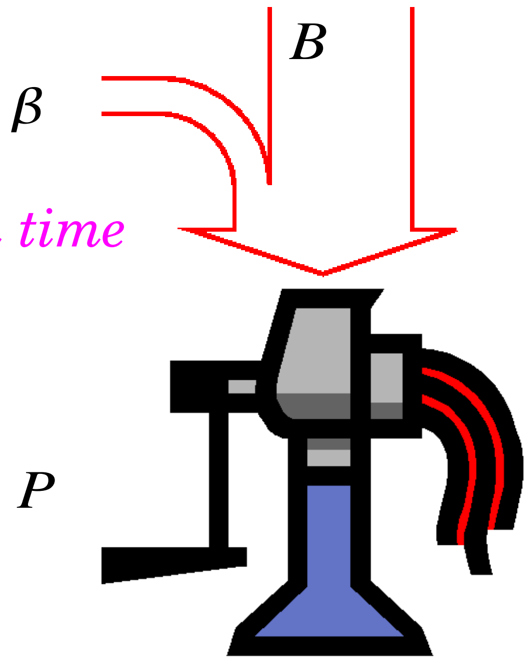
Computational power must be balanced by sufficient flow of data coming into the processor.

Appropriate parameters:

$n\ P$	*aggregate processor computing power*
R	*ratio of operations/data words*
B	*memory band width (to local pork-meat stock)*
ρ	*ratio of local/remote words*
β	*remote memory bandwidth*

Each processor will complete its share of computing in time

$$T = \frac{N}{nP} \qquad T = \frac{N}{nRB} \qquad T = \frac{N}{nR\,\rho\,\beta}$$



Simulation engines in a nutshell

Computing time is given by:

$$T = max\left\{\frac{N}{nP}, \frac{N}{nRB}, \frac{N}{nR\,\rho\,\beta}\right\} = \frac{N}{nP}\, max\left\{1, \frac{P}{RB}, \frac{P}{R\,\rho\,\beta}\right\}$$

Trade rule: accurately balance all terms in this equation.
From this point of view,

Spins is simple and easy (R constant [but small])

LQCD is easy (R constant and large $3.6 < R(m) < 14$)

and complex but NOT too complex (we have been able to manage...)

Important dates in LQCD computing

- ***1979:***

  The early pioneers: the Caltech Ising machine
  (D. Toussant, G. Fox, C. Seitz)

- ***circa 1985:***

  APE (16 nodes, 1 Gflops) 2 MEuro / Gflops
  Columbia (~ 1 Gflops)
  GF11 (IBM/Yorktown)

- ***1990 - 1995:***

  APE100 (500 – 1000 nodes, 50 – 100 Gflops) 40 KEuro / Gflops
  Columbia (also about 100 Gflops)

Trends in LQCD computing

- ***1995 – 2000:***
  APEmille (1.8 Tflops installed) — *3 KEuro/Gflops*
  QCDSP (1 + 1 Tflops at Columbia & Broohhaven)
  CP-PACS (Tsukuba + Hitachi, 600 Gflops)

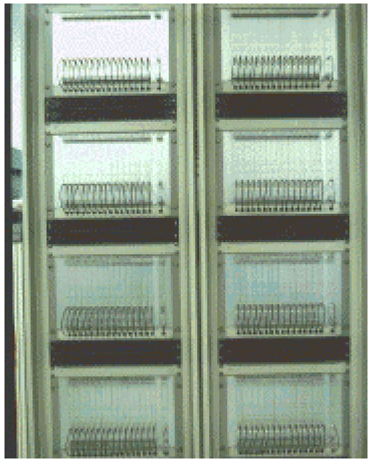
- ***2000 – 2005:***
  apeNEXT — *400 Euro /Gflops*
  QCDOC (Columbia + Brookhaven + IBM/Yorktown)

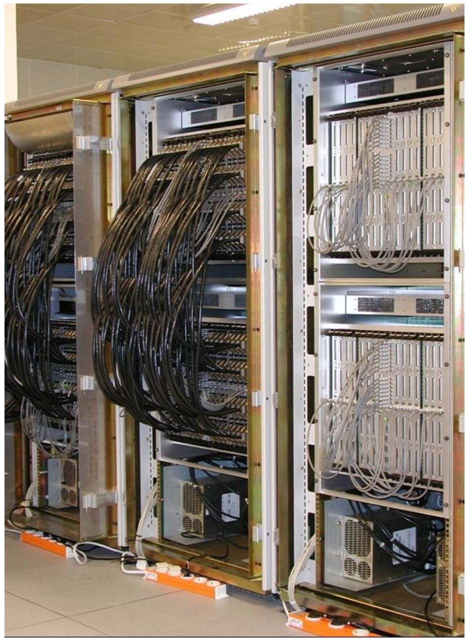
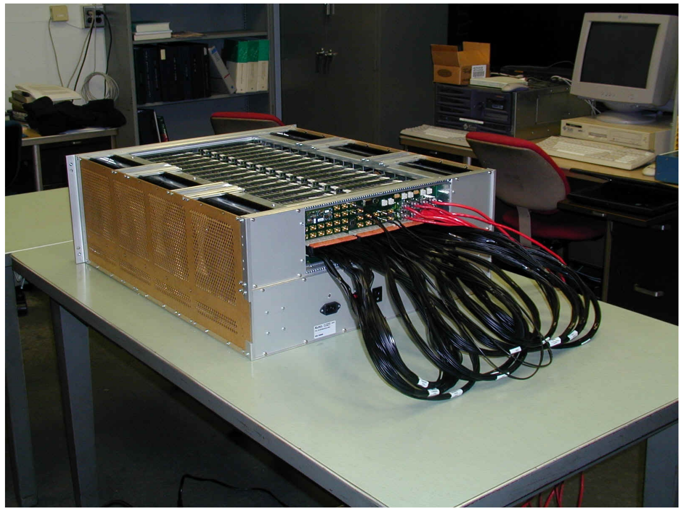
- ***2005-2009***
  The Blue Gene "revolution" — *~ 100 Euro / Gflops*

~2010

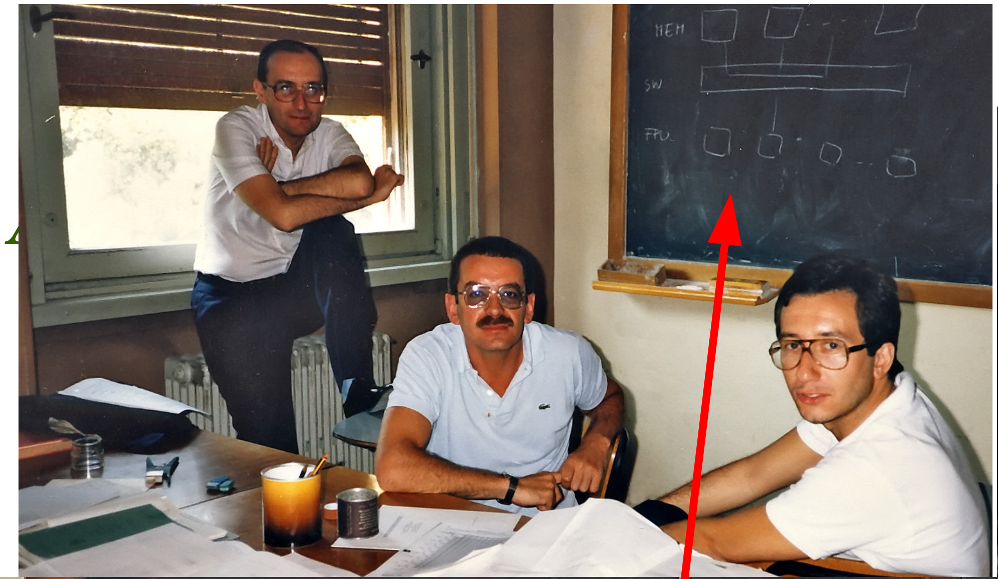
????????

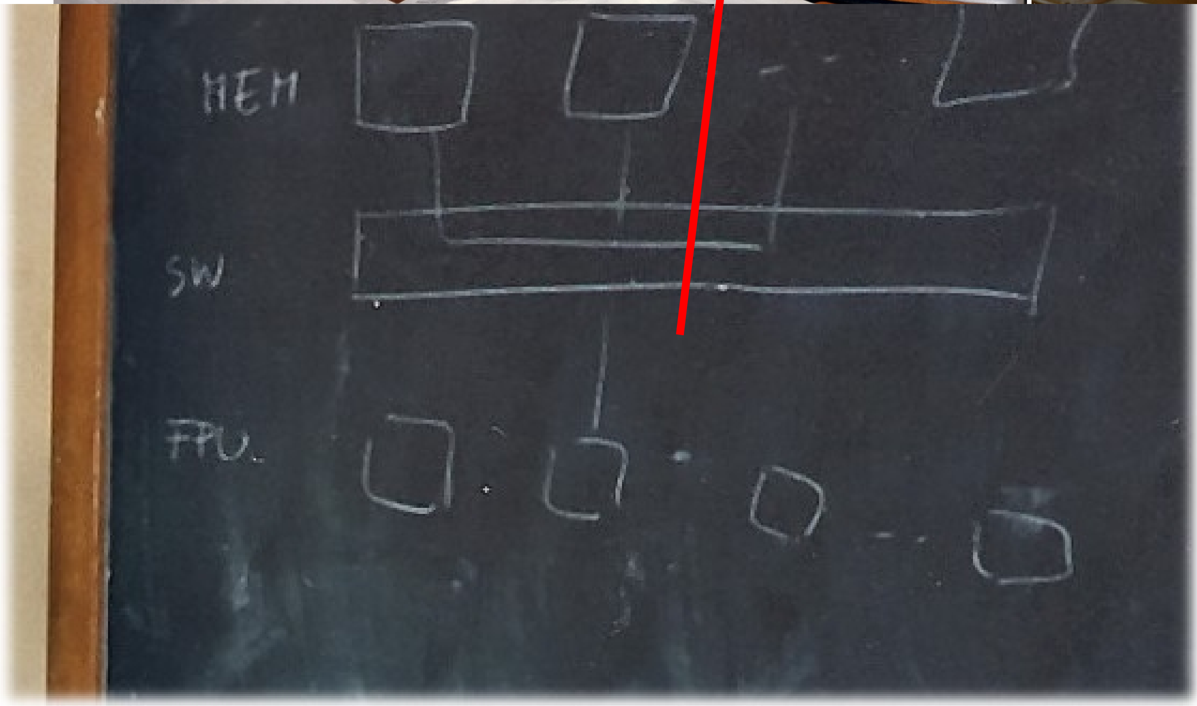
Visiting an LQCD museum

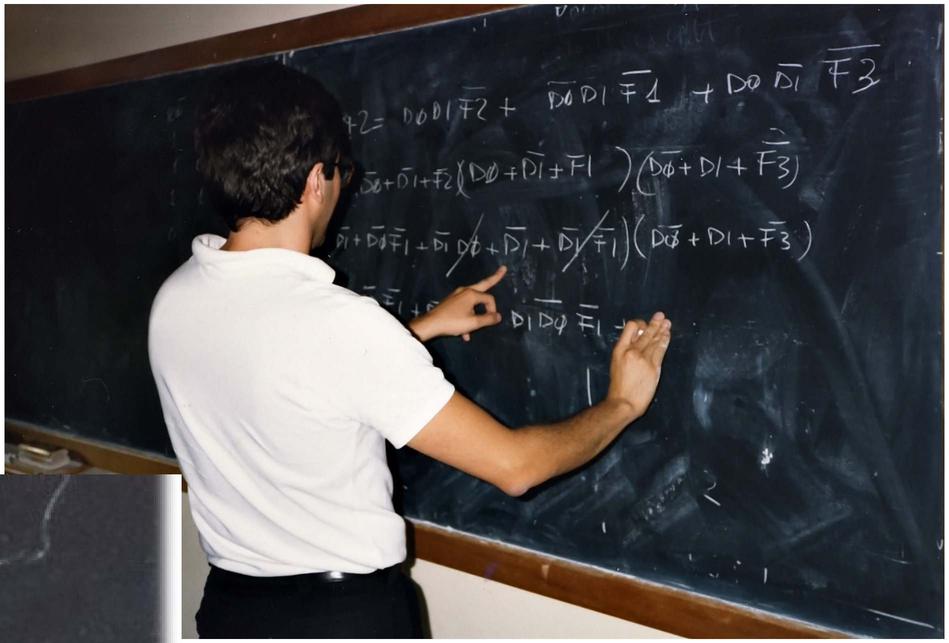


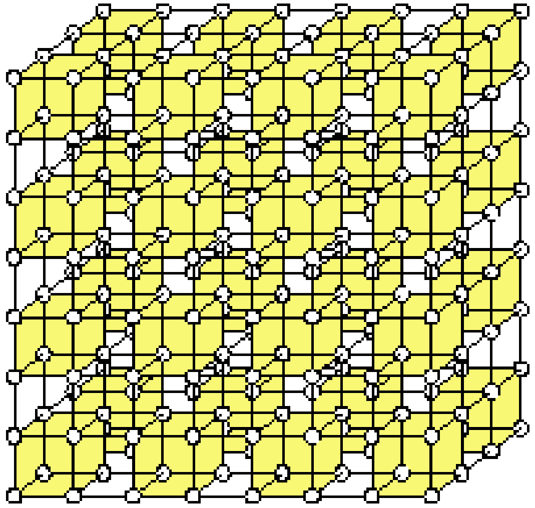



Prehistory: APE

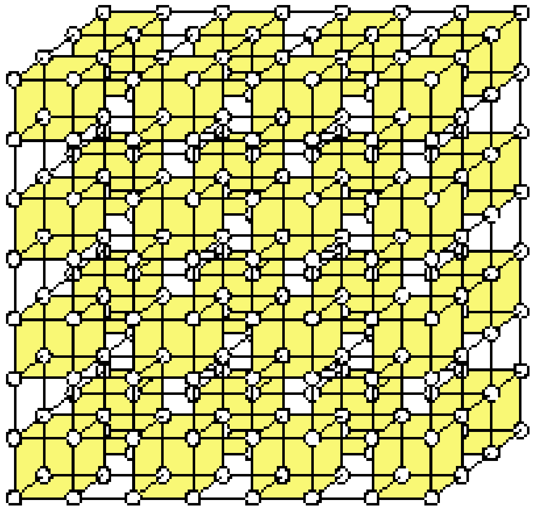








Prehistory: APE

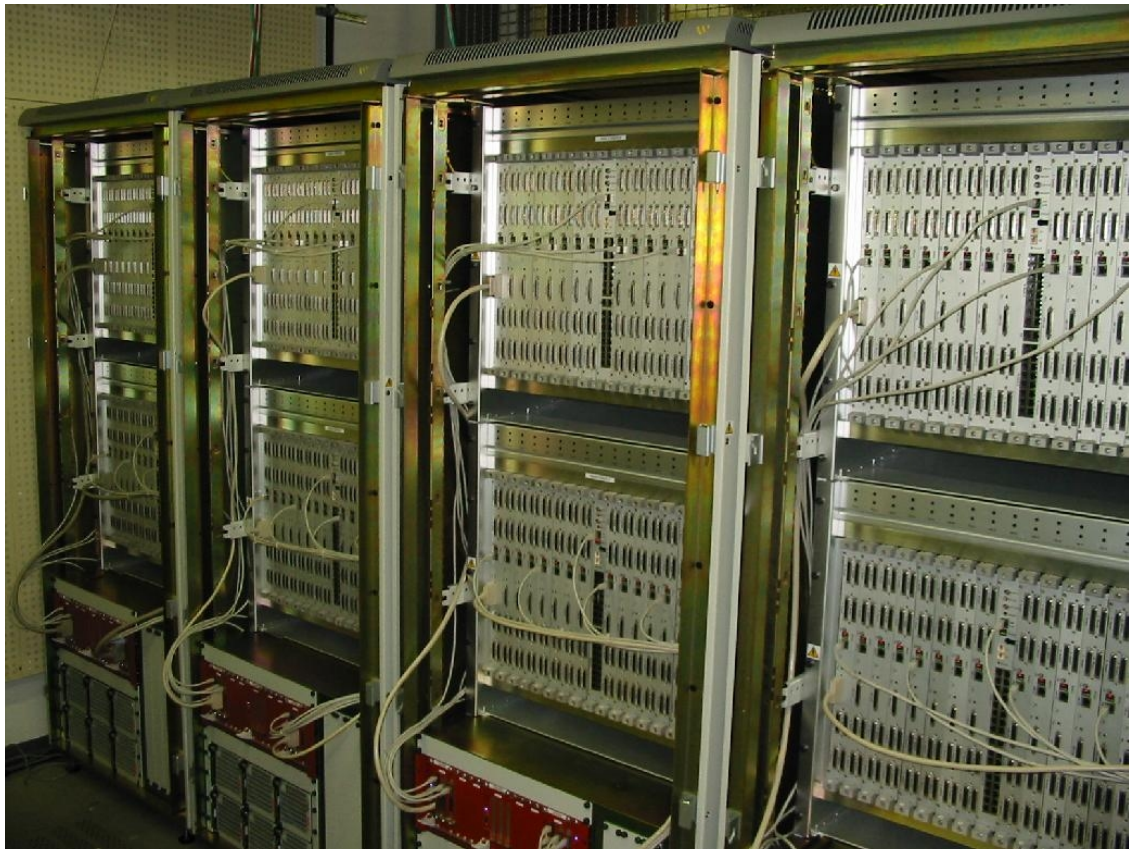


Facile, forse anche possibile !

(Giorgio, ~ 1985)

My daddy said we looked ridiculous, but, boy, we broke some hearts!

(Rod Stewart, "I was only joking")



The Blue Gene revolution …

Around 2003 – 2004 the idea that QCD machines are not just toys is endorsed by a little-known US company called (if I remember correctly) IBM.

The new gospel:
i) large machines can only be made as very large 3D meshes of simple relatively low performance distributed-memory processors --->
(computers are local theories ….)

ii) if you have a HPC application, you better learn to adapt your algos / programs to this specific architecture … or die

The Blue Gene revolution ...

Blue Gene is not too different from QCDOC / apeNEXT

marginally faster
better routing
better software

huge investment in porting
a large set of applications

marginally cheaper than
dedicated machines

carries the BigBlue brand...

Impact on LQCD

On fools's day 2008:

~ 30 - 40 Tflops QCDOC (US & UK)
~ 20 – 25 Tflops apeNEXT
(France, Germany, Italy)

220 Tflops Blue Gene/P
(Germany only,
~ 20-30% for LQCD)

The best money can buy?????

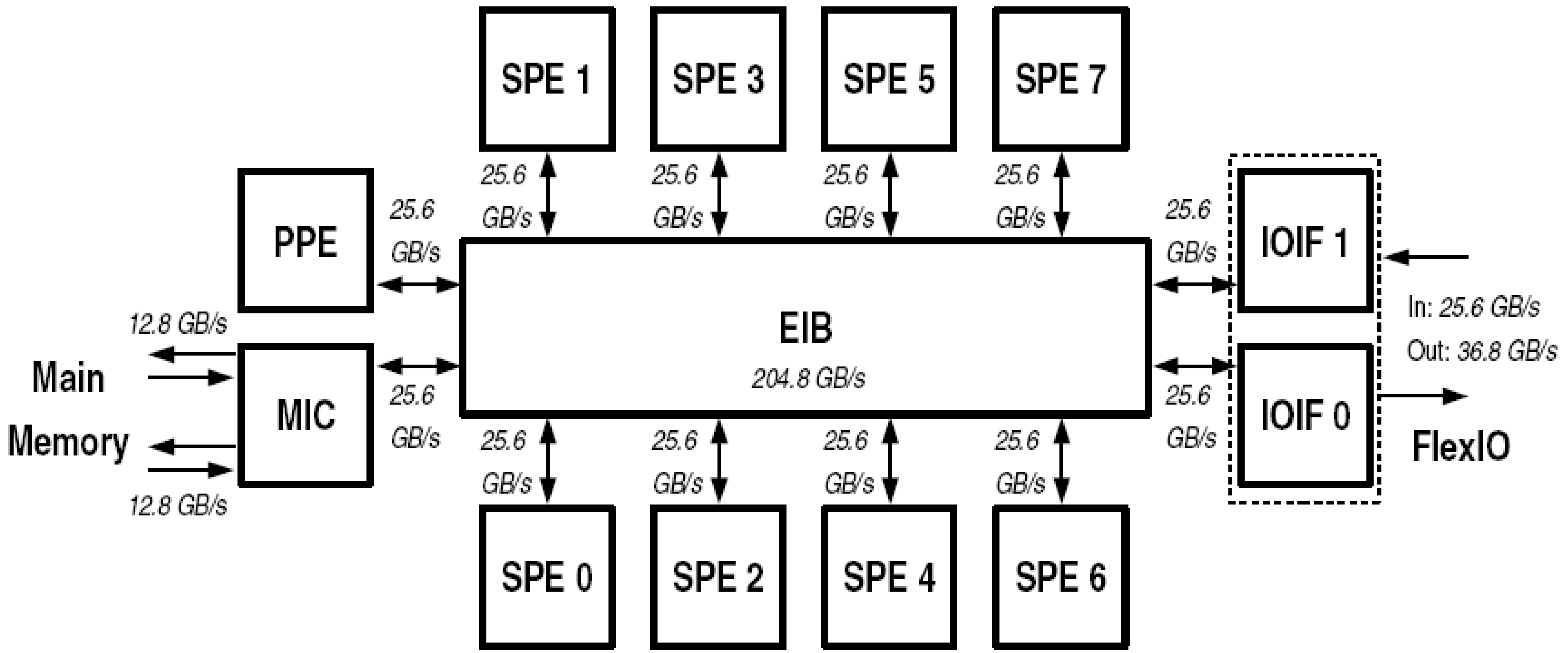
Remember our master equation?
Given avail. B/width and size of on-board memory, can work out the optimal floating point performance for that processor (Fopt)

$\xi = F_{opt} / F$ $\quad$ $\xi > 1$ *underpowered* $\quad$ $\xi < 1$ *low sustained perf.*

	apeNEXT	**BG/L**	**Cell**	**CSX600**	**ITANIUM2**	**QCDOC**
frequency	200Mhz	700Mhz	3.2Ghz	250Mhz	1.6Ghz	500Mhz
λ	180nm	130nm	90nm	130nm	90nm	130nm
L_w	64	32/64	32/64	64	64	64
F	8	4	64/8	192	4	2
m	32kb	32Mb	20Mb	4.5Mb	72Mb	32Mb
b_{lc}	128	62.85	64	102.4	x	41.6
b_{nb}	48	24	192	256	$32 - x$	21.8
ξ_{LM}	n.a.	6.07/2.28	2.27/6.06	0.17	4.04	4.13
ξ_{MM}	0.76	9.53/4.77	0.61/2.42	0.16	2.20	6.30

The cell processor (or, the most chauvinist slide of this talk)

An 8x apeNEXT processing board on just one chip



QPACE

A Cell based 3-d system

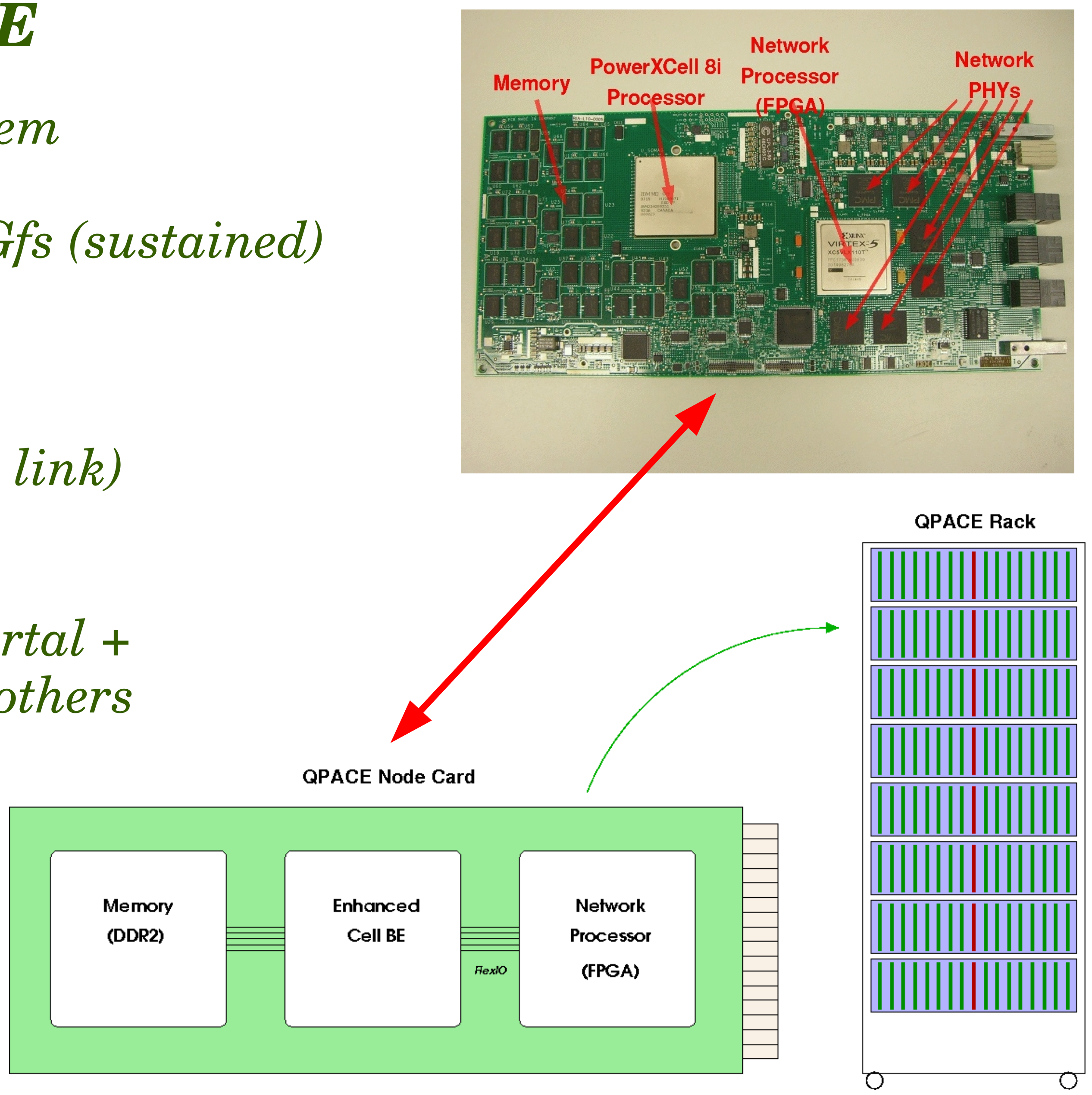
~ 100 Gfs (peak) 30 Gfs (sustained) for each node

Balanced network (1 Gbyte/sec for each link)

Collaboration of Regensburg + Wuppertal + IBM / Boeblingen + others

Proto in early 2009

Big machine (~ 1000 nodes in early 2010)



Partial Conclusions (appropriate for LQCD)

Tailoring a computer to a specific number-crunching application may bring substantial advantages

New computer architectures (both at chip and system level) seem to have incorporated several lessons coming from LQCD computing

It is probably fair to say that LQCD computing has given a non trivial contribution to HPC computing at large

However, now that the lesson has been learnt, collaboration with industry is probably the most effective approach today to ensure that the LGT community has the number crunching tools it needs in the near future

(Conflicting) trends in spin-system computing

- ***1979:***
  The early pioneers: the Caltech Ising machine
  (D. Toussant, G. Fox, C. Seitz)

- ***circa 1985:***
  *Ogielski et al.   (**hardwired** for Ising model)*

- ***circa 2000***
  *SUE (A. Cruz et al.) (**hardwired** for EA spin-glasses)*
  *1 **ns** / spin-flip*

- ***2007 - 2008***
  *JANUS (F. Belletti et al.) (**configurable:** Ising, Potts, KSAT.....)*
  *60 **fs** / spin-flip*

The JANUS project

A collaboration of:

Universities of Rome (La Sapienza) and Ferrara

Universities of
Madrid, Zaragoza,
Badajoz

BIFI (Zaragoza)

Eurotech
Microsoft

We already went a LONG WAY ......

We already went a LONG WAY ……

… but there is still a LONG WAY to go

Use all available parallelism

Spin glass simulations have two levels of available parallelism

1) Embarassingly trivial: need statistics on several samples ---> farm it out to independent processors

2) Trivially identified:
can update in parallel any set of mutually non-interacting spins

make it a black-white checkerboard:
tens of thousands of independent computing threads...

1) & 2) <u>do not</u> commute

Use all available parallelism

1) & 2) do not commute:

Farming out independent computation of many copies of the system, helps till this makes sense from the point of view of physics

Beyond that point one has to expose the parallelism available within the Monte Carlo history of each replica of the system

So the needed ~10^{12} updates of each system can be performed

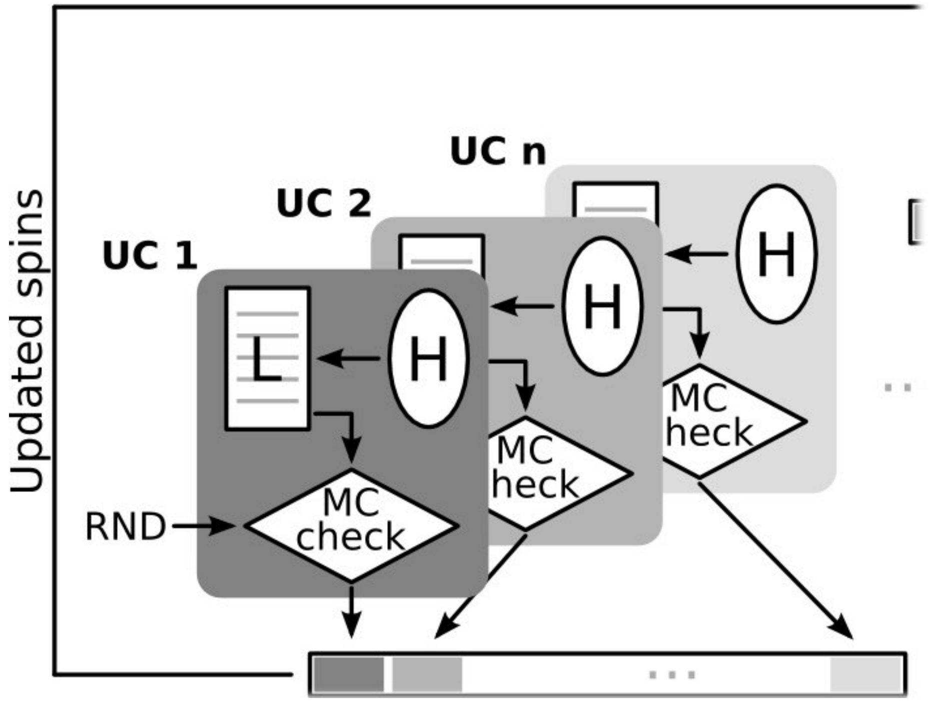
Why does it work ????

One update engine:

computes the local contribution to U
addresses a probability table
compares with a freshly generated random number
sets the new spin value

All this is just a bunch (~1000) of logic gates

$$U = -\sum_{NB(ij)} \sigma_i J_{ij} \sigma_j$$



Why does it work ???

All this is just a bunch (~1000) of gates

And in spite of that a typical CPU, with O(10^7+) logic gates can process perhaps 4 spins (of a given sample) at each clock cycle

If you are able to arrange your stock of gates the way it best suits the algorithm, can expect >1000 update engine

Computer scientists call this a massively-many-core organization

The ideal spin glass machine .....

is an orderly structure (a 2D grid) of a large number of "update engines"

each update engine handles a subset of the physical mesh

its architectural structure is extremely simple

- *each data path processes one bit at a time*
- *memory addresing is regular and predictable*

SIMD processing is OK

however memory bandwidth requirements are huge (need 7 bit to process one bit..)

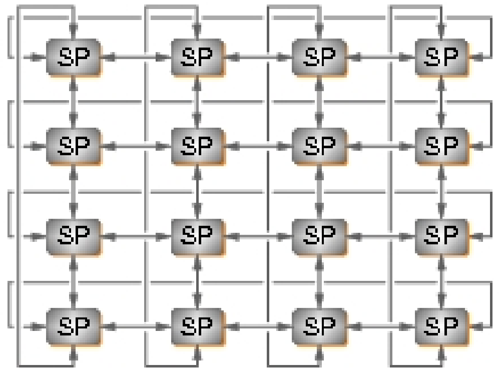
- *however memory is "local to the processor"*

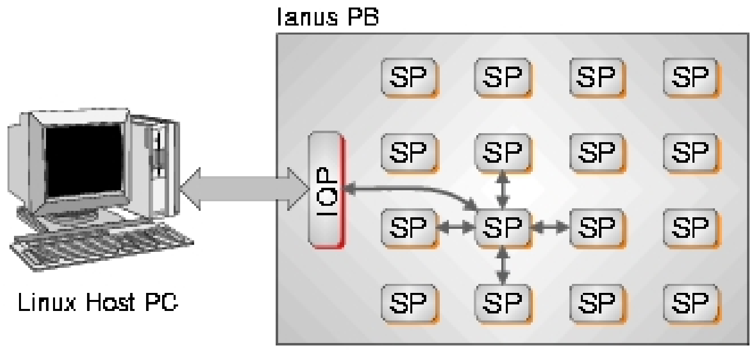
The JANUS system

A parallel system of (themselves) massively parallel processor chips

The basic hardware element:
A 2-D grid of 4 x 4 (FPGA based) processors (SP's)
Data links among nearest neighbours on the grid

One control processors on each board (IOP) with 2 Gbit Ethernet links to host

A.


B.


FPGA, a key building block

FPGAs are chips than can be configured (programmed) to become any given logic system (within a certain set of constraints)

- *large amount of uncommited logic available to build processing cores (embedded parallelism)*

- *reasonably large on-chip memory (several Mbits)*

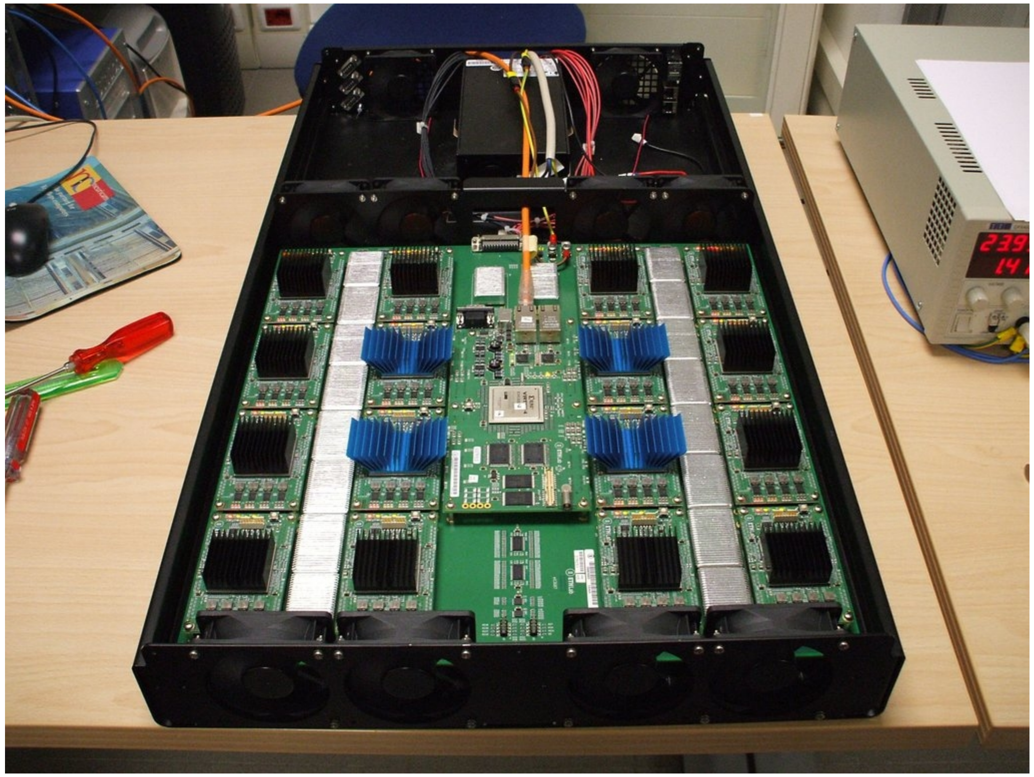
- *huge bandwidth for on-chip "distributed" memory (~ 10000 bits in and out of embedded memory per clock cycle)*

One of the goodies of this project is that

it is a technology NON-challenge ... (plug the LEGO bricks and play)

Ready to go....

(16 x 16) (replicas) x 1024 flips every 16 ns.....



(Measured) Performances

as fast as Nature ...

Only the <u>spin-flip rate R</u> matters....

For each processor: $R = \frac{1}{Nf} = \frac{1}{1024 \times 62.5 \ MHz} \simeq 16\, ps \ / \ flip$

For one element of the IANUS core (16 procs):

$$R = \frac{1}{N_p N f} = \frac{1}{16 \times 1024 \times 62.5 \ MHz} \simeq 1 \ ps \ / \ flip$$

sustaining these performances requires huge bandwidth to/from (fine-grained) memory:

one flip uses 12 more bits from memory ... and generating pseudo-randoms is even worse ...

<u>All in all ~ 1 Tbyte/sec combined memory bandwidth</u>

Early physics runs

Our first large simulation campaign with Janus (spring, 2008):

Isothermal aging (below T_c) of a large EA spin-glass;
$L = 80$ (40, 24)
direct quenches to $T = 1.1, 0.8, 0.7, 0.6$ ($T_c \sim 1.1$)

sample statistics:
$L = 80$: 96@ $T = 0.8, 0.6$; 64@$T = 0.7$; 32@$T = 1.1$
$L = 40, 24$ 32@$T = 0.8$

~ 10^{11} MC steps (HB) ~ 0.1 sec

~ 24 days of continous run (powered by just ~ 15 oil barrels)

Performance figures

Janus helps brings wall clock time down to reasonable times (on a human timescale...)

	JANUS	256 CPUs
Samples	256	256
Wall clock time	24 d	24 y
Acc. CPU time	18 y	6200 y
Energy	22 GJ	18 TJ

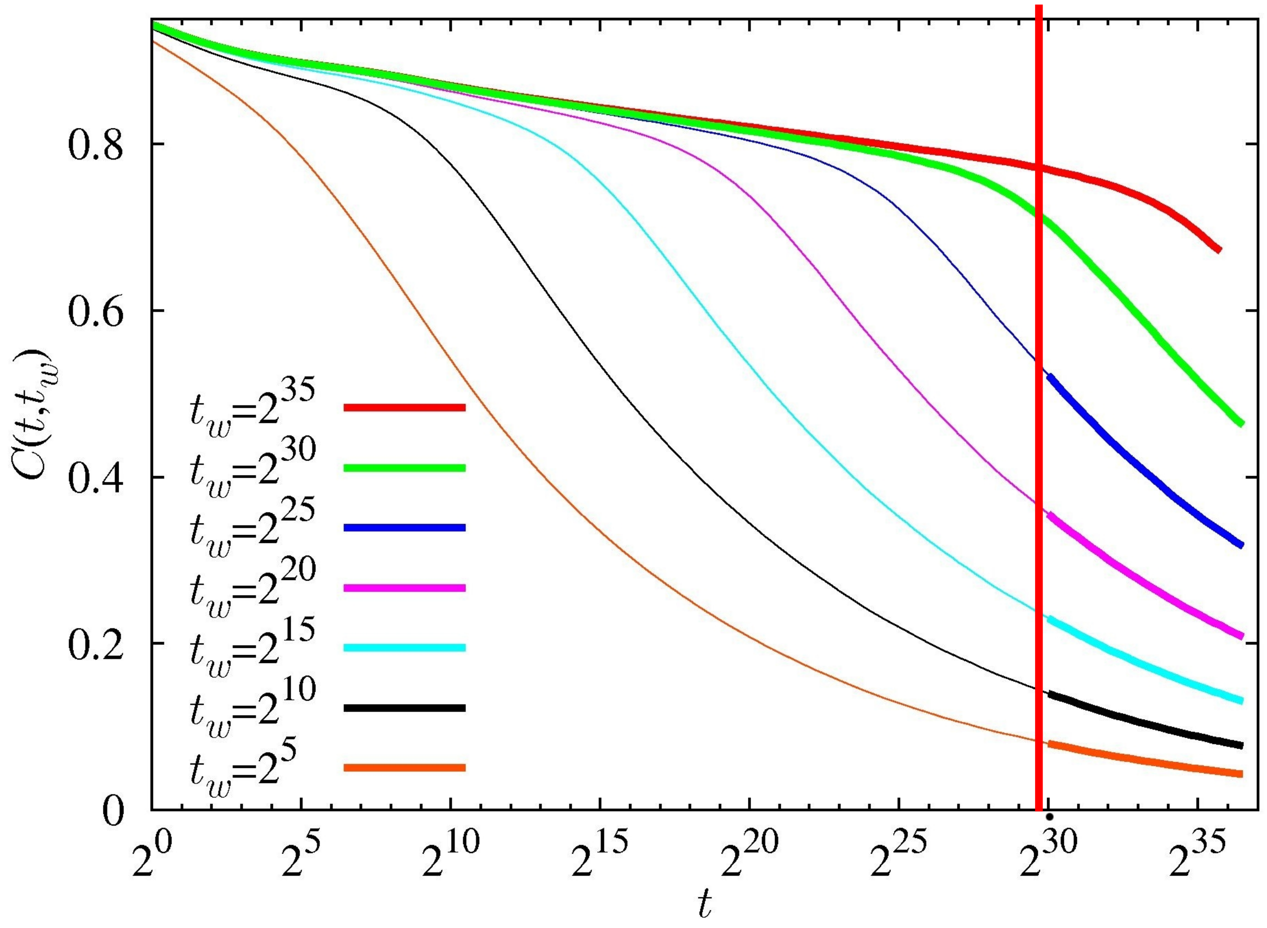
The bottom line:

90 Tera-ops on the whole system

10.27 $ / Giga-ops

8.75 Giga-ops / W (20x better that the top Green500 entry, if ....)

Early physics runs



$$C(t,t_w)=L^{-3}\overline{\sum_x \sigma_x(t+t_w)\sigma_x(t)}$$

Conclusions (i)

Over the years application-driven computing has been an important tool for physics

Ideas from application-specific computing have recently crept into mainstream computing ...

... and mainstream computers (or limited variations thereof) have become the best bet

As a conflicting trend, surprisingly simple toys are still a surprisingly efficient solution for some physics application

Conclusions(ii)

Over the years, Giorgio's ideas , drive and enthusiasm have been an important contribution to this process …

so, thank you, Giorgio, for this … and much more!