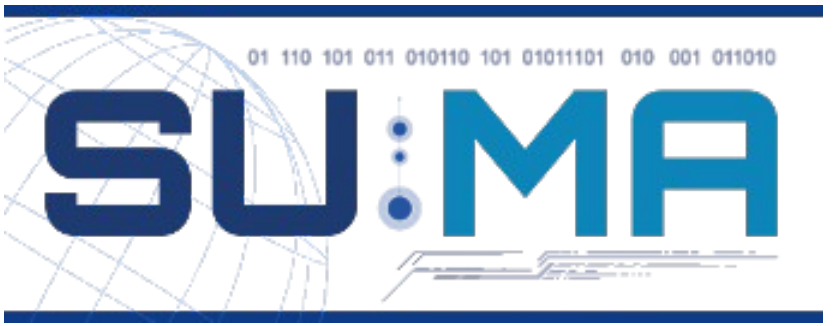
The INFN-SUMA project

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

SM&FT 2015
Bari, December 9th - 11th, 2015



On the menu today

What is SUMA?

A quick survey of the project.

Learning to use "new" computers for physics efficiently and (if possible) comfortably.

Conclusions and take-away lessons.

SUMA: what does it mean?

We have drunk <u>suma (soma??)</u> and become immortal;
We have attained the light, the Gods discovered.
(Rigveda 8,48,3)

One cubic centimetre cures ten gloomy sentiments.
(A. Huxley,
Brave New World, 1932)

A quick survey of the SUMA project

An INFN "special-project" (progetto premiale) running for three years (Jan. 2012 – Dec. 2015); focus is the support of HPC computing for theoretical physics:

NO LONGER developing our own machines (terribly ***unfashionable*** *today), but rather --->*

1.- Making sure INFN physicists have access to HPC resources

2.- Supporting theoretical/computational post-docs

3.- Learning to use "new" HPC computers as efficiently (and comfortably) as possible

the SUMA project: access to HPC machines

An effort going into two different directions:

1) Agreements with large computing centres (CINECA)

Access to 100 Mcore-hours / year on the CINECA Tier-0 machine (BG/Q) - (nicely matches ~100 Mcore-hours / year from PRACE projects with INFN PIs)

SUMA has cofunded the CINECA Tier-1 Cluster (traditional multi-core CPUs + GPUs + Intel/MIC, 1 Pflops peak)

Reserved use of 20% of the machine, i.e. ...
(additional ~30 Mcore-hours BG/Q-equivalent)

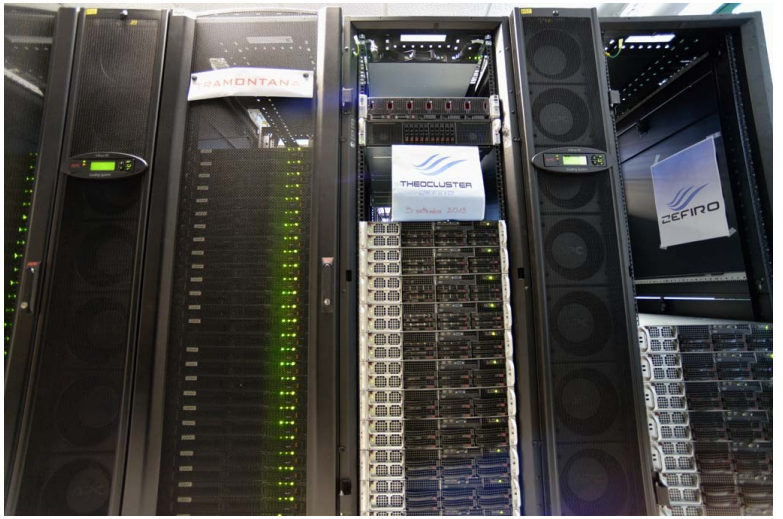
the SUMA project: access to HPC machines

An effort going into two different directions:

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

Some 1000 computing cores
Infiniband QDR
+ Experimental nodes (GPUs)

up and running since Sept. 2013



the SUMA project: post-docs

Several post-docs hired for (on average) 2 years

M. Brambilla	*(Parma, LGT)*
E. Calore	*(Ferrara, LBE)*
L. Scorzato	*(Trento, LGT)*
A. Feo	*(Parma, Grav.)*
G. Engel	*(Milano, LGT)*
M. Schrock	*(Roma 3, LGT)*
F. Negro	*(Pisa, LGT)*
G. Caruso	*(Pisa, Programming)*
F. Stellato	*(Roma 2, Q-Bio)*
P. Vilaseca Mainar	*(Roma, LGT)*
L. Riggio	*(Roma3, LGT)*

the SUMA project: any results??

As of Nov. 30th, 2015 (and counting):

- 35 journal papers (LGT, nucl. Physics, q-bio, complex systems)

- 38 confererence papers (as above...)

Details on some physics results at this conference, see talks by:
C. Bonati
F. Cuteri
L. Giusti
M. Mesiti
S. Morante
A. Papa

SUMA: learning to use "new" computers

Computers are quickly changing todays, in terms of

- *their architecture*
- *(more directly relevant) the way one uses them efficiently*

Problems come from the fact that computers are increasingly parallel in structure and that many decades of computer programming experience have neglected this option…

However, this is going to stay in the foreseeable future, for basic reasons connected to the physics constraints on how computer work …..

Physics hints on how computers should be ....

Basic physics reasoning provides a few hints on how computers should be done....

1) Parallellelism has good reasons to be the way to go ...
... or parallel computing is the physics sponsored way to compute:

The basic object in computers today is the transistor

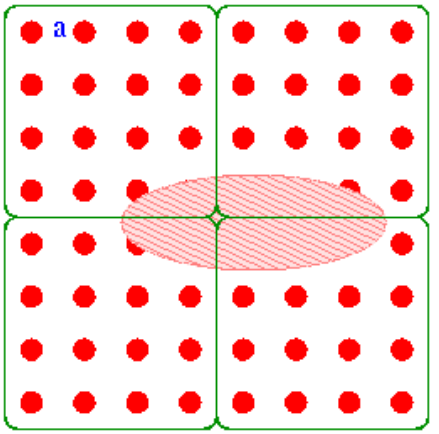
Industry learns to build smaller and smaller transistors. As $\lambda \to 0$
obviously $N \propto 1/\lambda^2$ *but speed scales less favourably* $\tau \propto \lambda$

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

Physics hints on how computers should be....

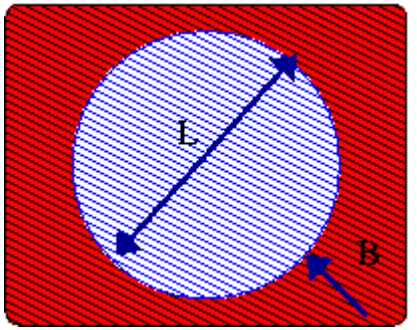
Basic physics reasoning provides a few hints on how computers should be done....

2) Physics moves information locally
This has to go over to the computer structure ->
Keep data close in space to where it is processed

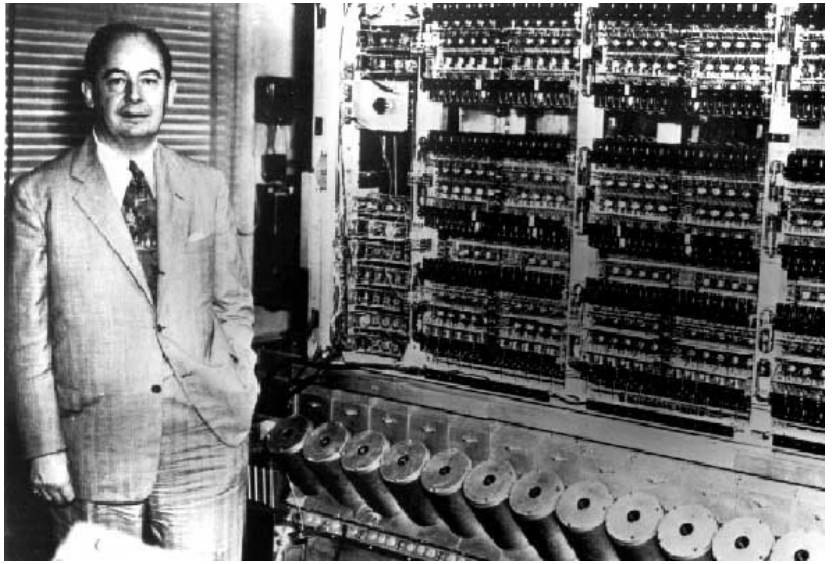


Failure to do so will sooner or later
bring a data bottleneck:

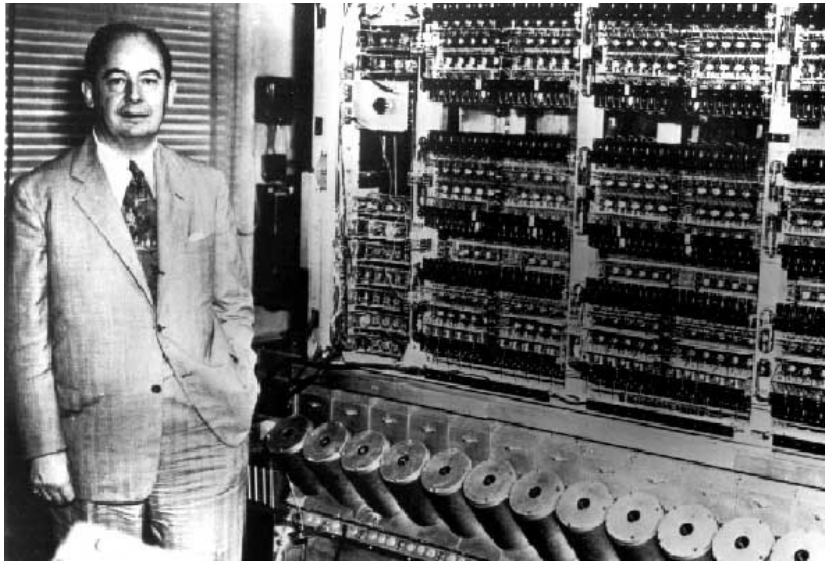
$$B(L) \propto L$$
$$P(L) \propto L^2$$



A historical remark:



A historical remark:



Doing things one after the other (serially)

Keeping data storage and data processing separated (in principle and practice) ….

…are the cornerstones of the famous von Neumann model of computing

Q: So was Von Neumann wrong?
A: No ….

he was interested in the $N_P \to 1$ *regime*

while today we are approaching the $N_P \to \infty$ *regime …..*

SUMA: learning to use "new" computers

Computer companies have apparently learned the lessons of the previous slides and invented two new computer breeds:

- GPUs (e.g., NVIDIA)

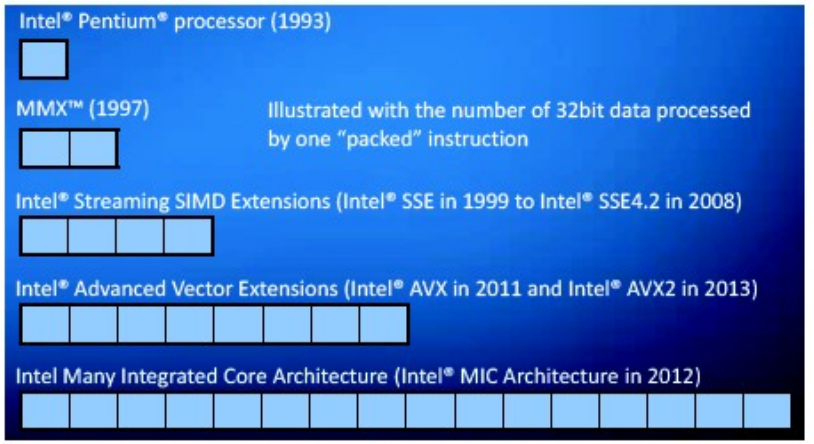
- multi-many core processors, (e.g., Intel MICs)

This basically boils down to:

- Putting many smaller cores within one chip

- Having each instruction operate on many data items

- Keeping a sizeable amount of data within the chip (large caches)



Questions worth being answered…

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?

Trying to provide (experimental) answers…

Direct experience in 2 real-life cases:

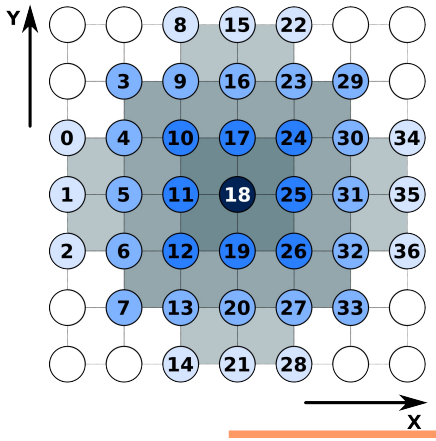
Fluid – dynamics using the Lattice Boltzmann (LB) method

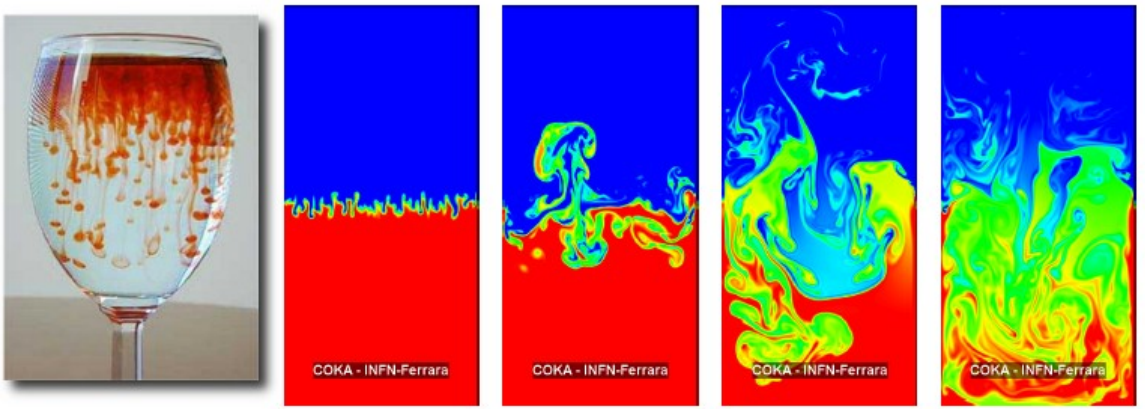
LQCD with staggered fermions.

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.

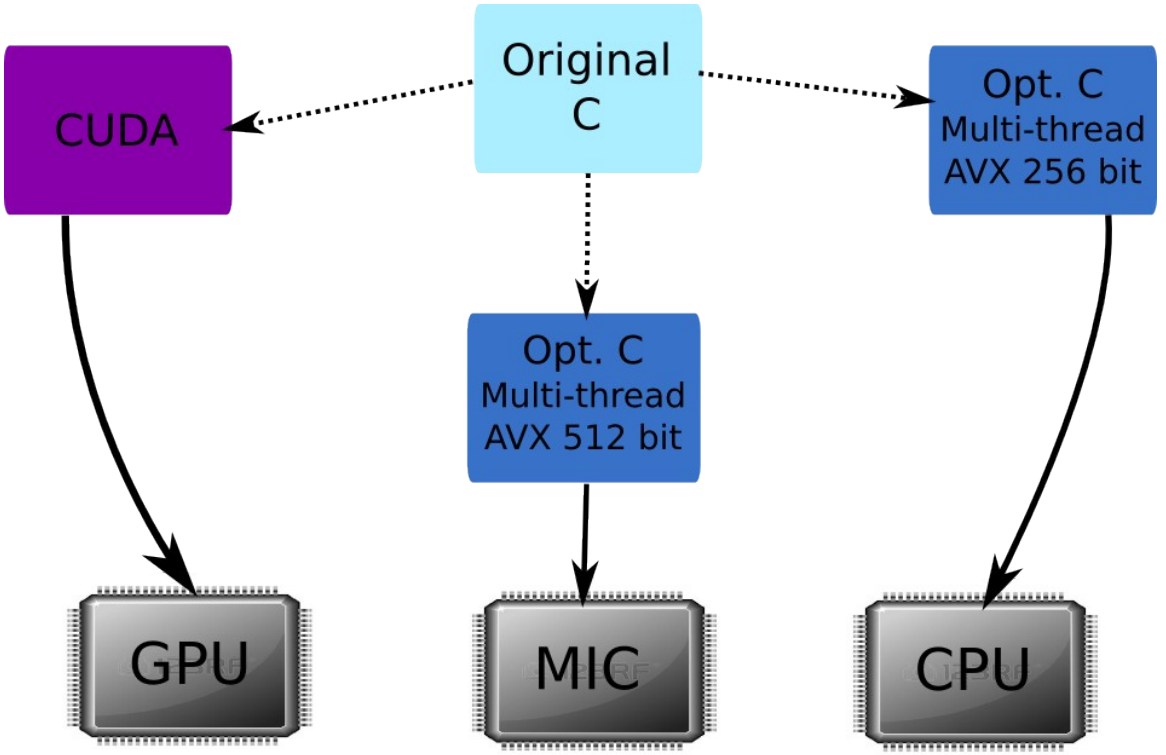
- Massively-parallel algorithm, easily partitioned on many nodes

- Just two computationally critical kernels (collide - propagate)

- No significant role of global sums (at variance with LQCD)

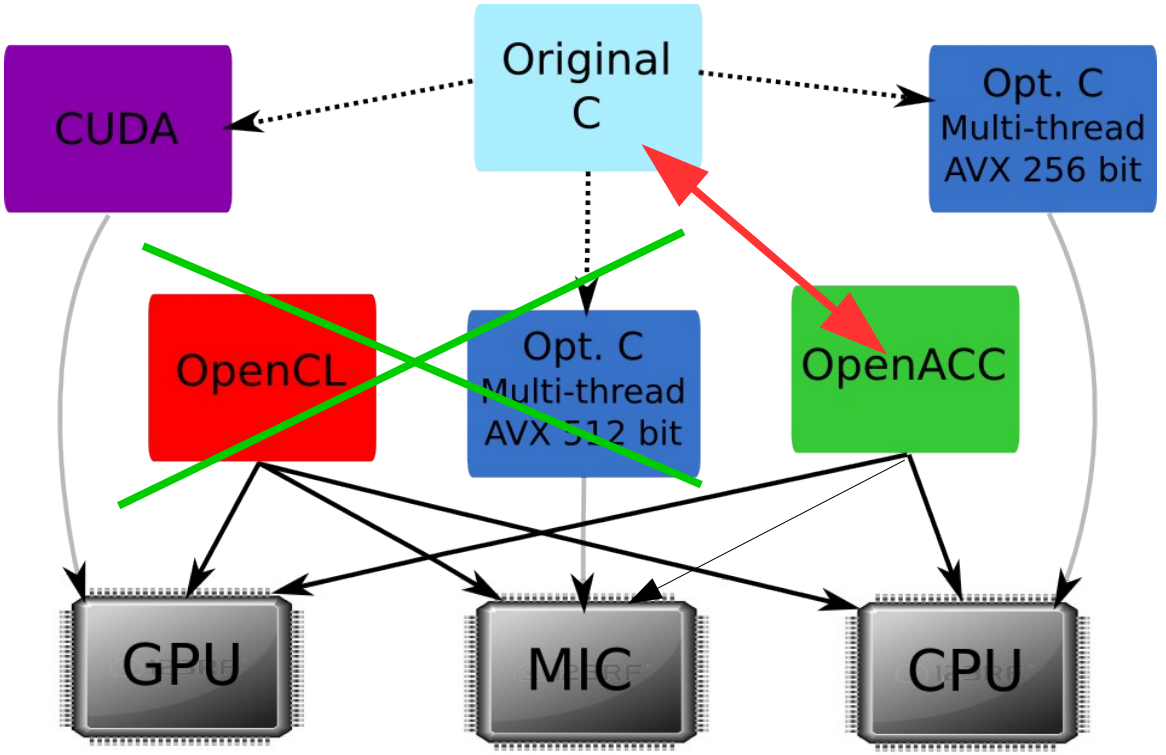
Trying to provide (experimental) answers…

In practical terms: can we make a transition from here ….



Trying to provide (experimental) answers…

In practical terms: can we make a transition from here …. to here



…. and be happy with performance and portability

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??

	Tesla K40			Tesla K80		
Code Version	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_{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
propagate perf. [GB/s]	38	32	32	154	150	155	232	216
$\mathcal{E}_p$	65%	54%	54%	64%	62%	65%	73%	70%
collide perf. [MLUPS]	14	11	12	117	32	55	76	54
collide 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

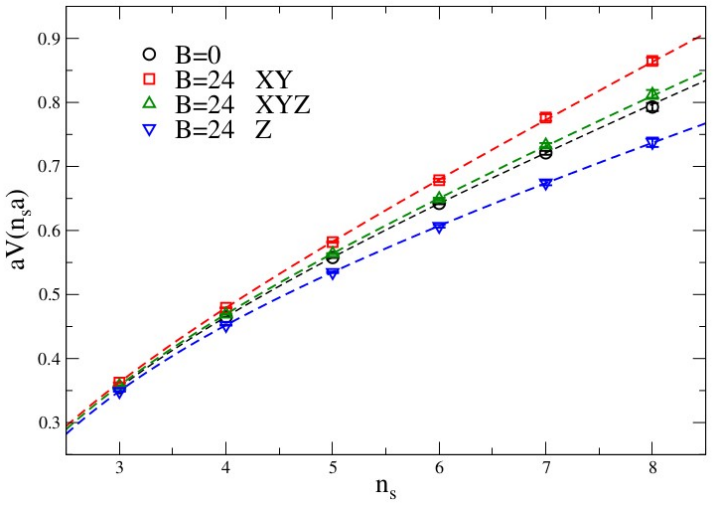
A LQCD code with staggered fermions + improved action + (two level) stout smearing

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) → (48^3 x 96)



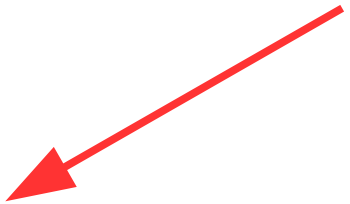
From LB to LQCD

Code fully (re-)written and tested (C + openACC)
Almost ready for physics runs

Preliminary performance tests on K20 – K40 – K80 GPUs

Comparison against the golden benchmark today (BG/Q)

	Measured	*Expected*
1 K20 GPU	*→ 80 BG/Q cores*	*(93)*
1 K40 GPU	*→ 96 BG/Q cores*	*(110)*
1 K80 GPU	*→ 160 BG/Q cores*	*(190)*
1 "heavy" GPU node	*→1300 BG/Q cores*	



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

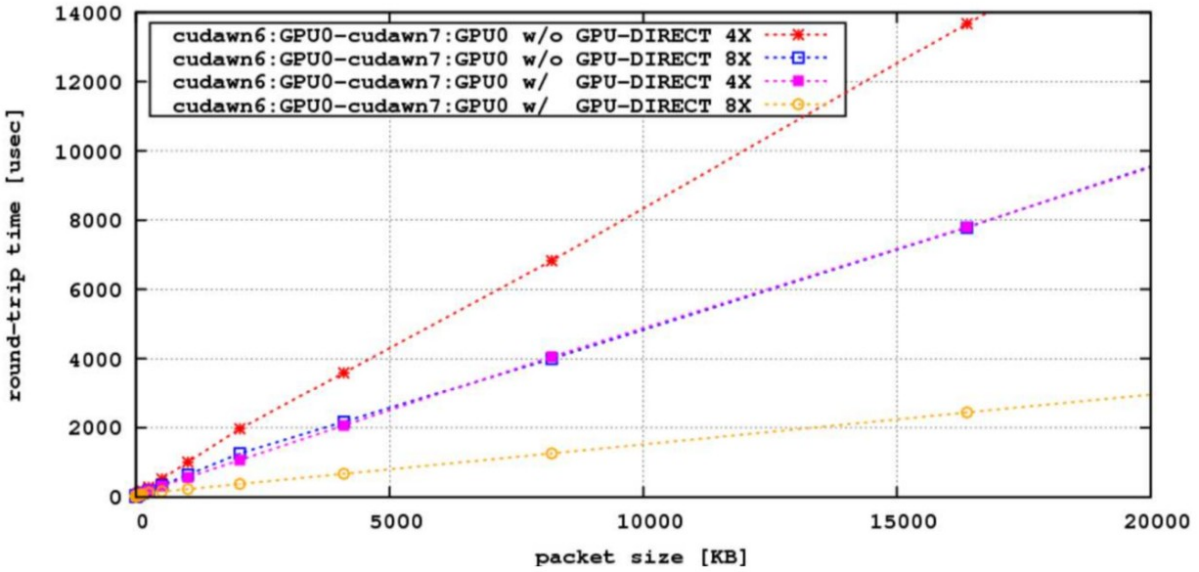
Not all programs automatically OK…

And …

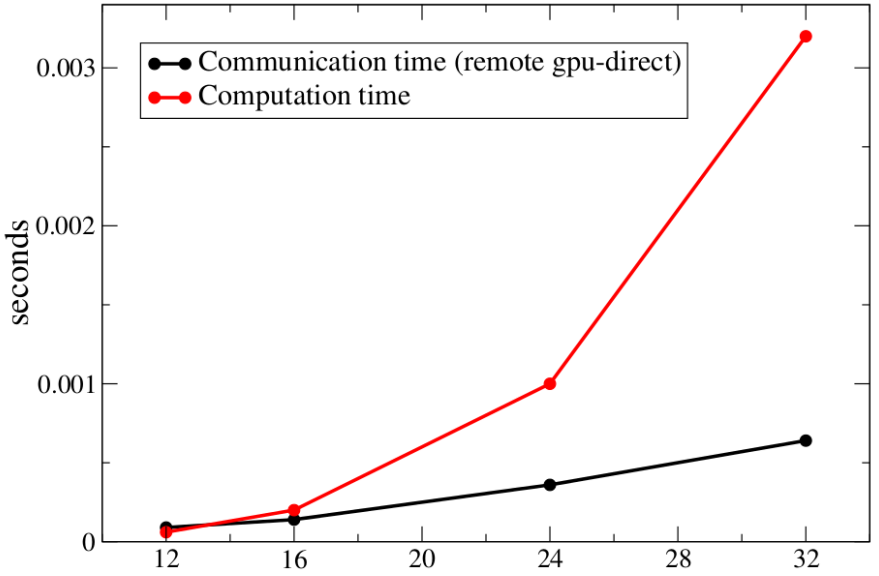
The problem in the next few years will again be in the match between processing power and communication capacity among compute nodes (Nothing as good as BG/X will be available in the near future….)

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

Some specific support is needed (at least in Italy) for the theoretical physics / computational community

SUMA has probably provided a fair amount of support in the last three years

The future however looks gloomy, as it is not clear whether a second round will be funded

Lack of "continuous" support may have dramatic consequences

Concluding remarks (2)

Fighting against computer programming to have fast-running codes has always been in the rules of the game

This has become more serious in recent years, as computer evolution has become faster

Good news is that we start to have programming environments that are very close to providing portable AND efficient codes with reasonable human effort.