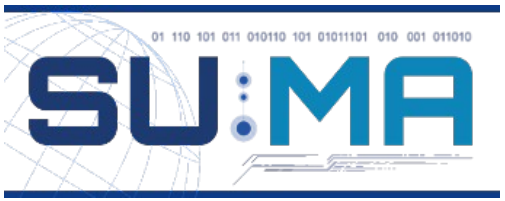
Computing for theory
OR
HPC computing
OR
(tightly coupled) parallel processing

R. (lele) Tripiccione
Dip. di Fisica e INFN
Ferrara (Italy)
tripiccione@fe.infn.it

Workshop CCR - Genova
29 maggio 2013



Menu del giorno...

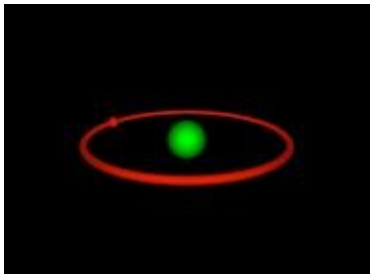
- *Cosa ci serve*

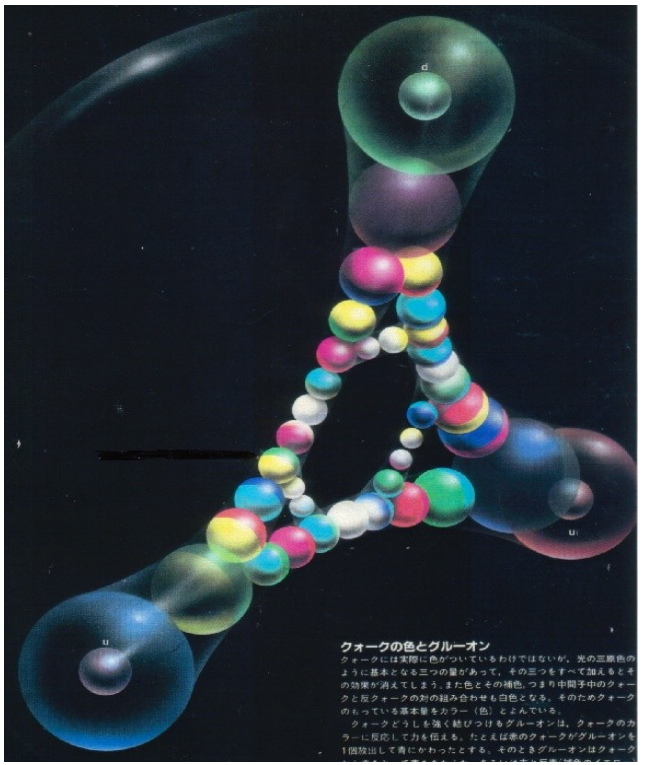
- *Come cerchiamo di avere quello che ci serve*
    - *Ora*
    - *In prospettiva*

- *Quali competenze possiamo offrire agli altri*

Lattice Quantum Chromo-Dynamics (LQCD)

The computer-friendly approach to QFT !





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

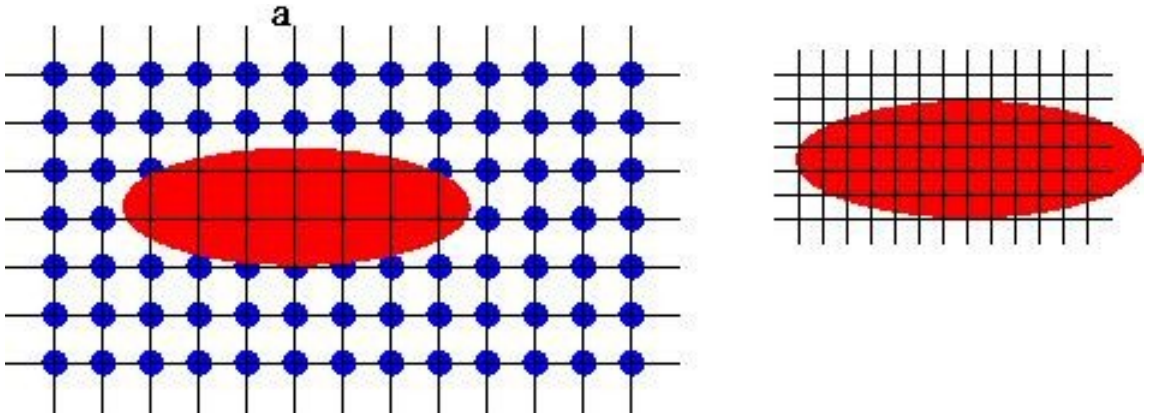
LQCD algorithms are regular and a huge amount of paralellelism is easily identified and exploited

Lattices are (usually) 4D and ...

… the computational cost is huge,

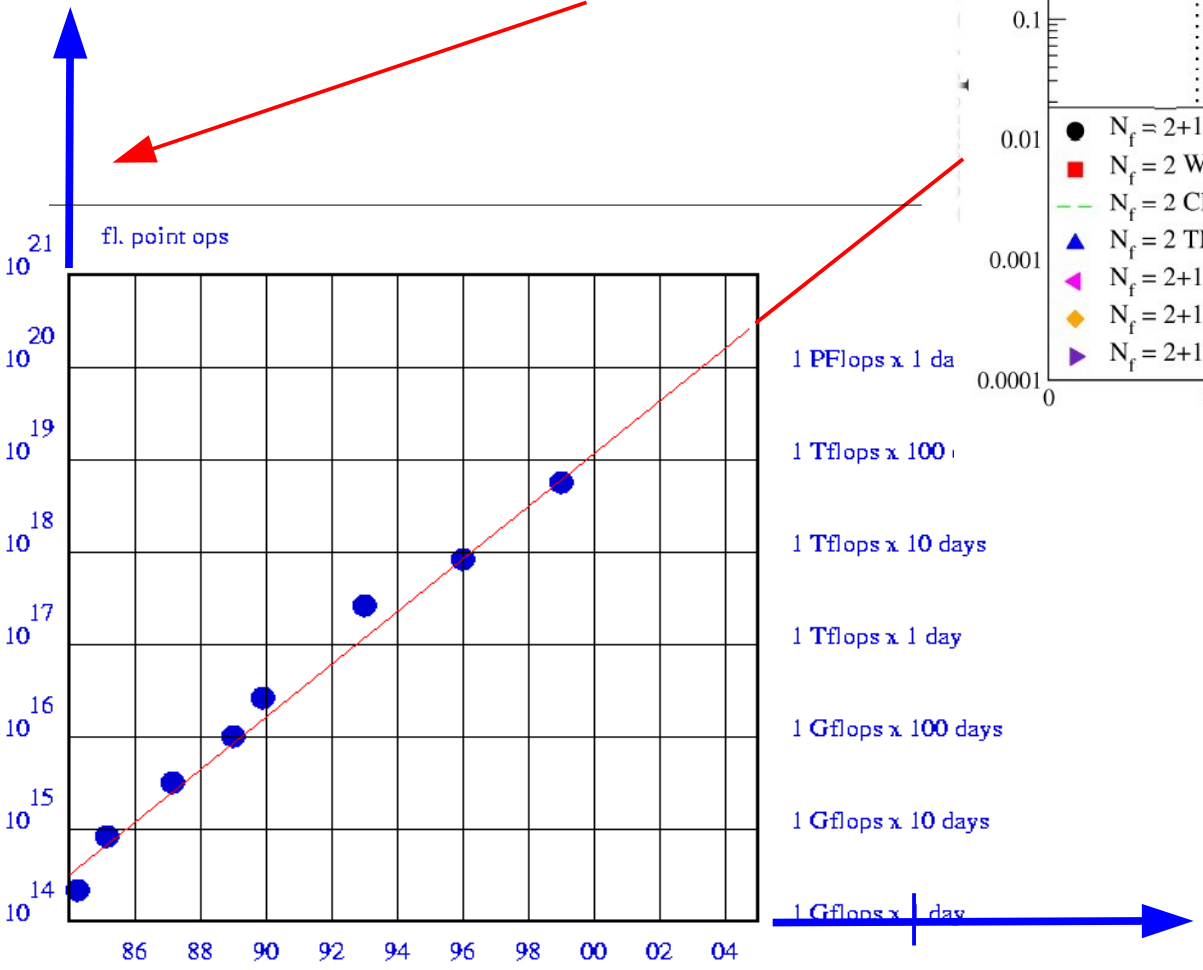
$$N_{flop} \sim L^{5\ldots6} \times (1/a)^{6\ldots7} \times (1/m_q)^{1\ldots2}$$

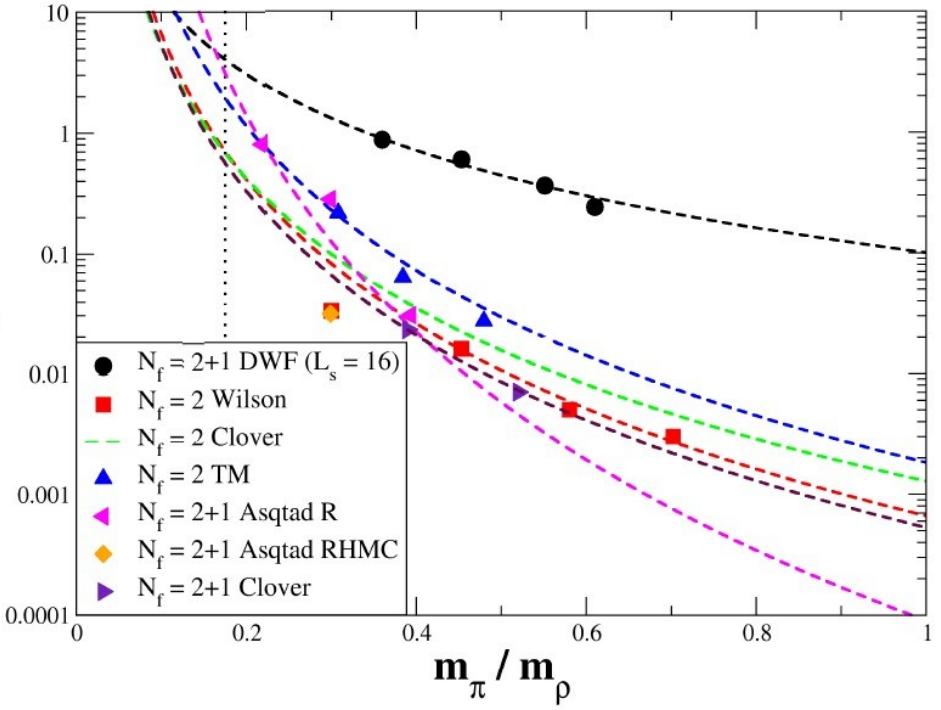
… as one tries to take into account all relevant scales of the problem.



Two different views of the same problem

0.5 Pflops x 100 d ~ 100 Tflops x 500 d





L = 2 fm
a = 0.08 fm
1000 configs

Qualche numero ….

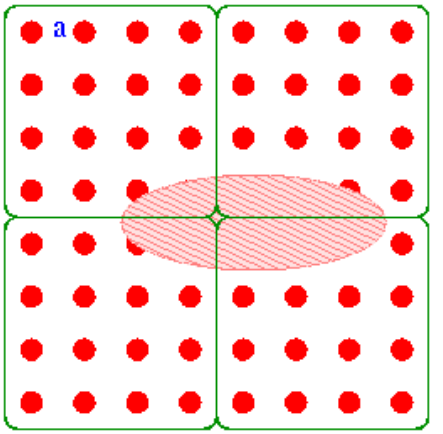
L' account-unit in questo campo e' la core-hour
1 core hour ~ 10 Gflop/s x 3600 sec = 3.6 10^{13} ops

In queste unita', l' estrapolazione della slide precedente diventa

→ ~ 140 Mcore-hour

Che e' leggermente superiore alla scala di calcolo tipica di un grosso progetto di LGT nel 2012-2013 (ETMC → 100 Mcore-hour/anno)

Tightly - coupled ???



Un esempio stato dell' arte:
Reticolo 128^4 simulato su 512 nodi ~ 8000 core
Sub-lattice 16 x 16 x 16 x 128 su ogni nodo

Su ogni sito del reticolo ~1000 operazioni

Per ogni punto sulla sup. del sub-reticolo devo trasferire 12 numeri complessi mentre eseguo i calcoli su tutto il sub-reticolo

Potenza effettiva di un nodo ~ 100 Gflops → ***B ~ 7 Gbyte/sec***

Ma granularita' naturale: 12 x 2 x 8 bytes → ***30 ns di latenza***

Qualche numero ….

	# nodi	#core	Int su un anno	Investimento	FTE
TheoPhys	100 nodi	2000 core	16 Mcore-hour	300 Keuro	1 … 2
Sissa	1000 nodi	10000 core	80 Mcore-hour	3 Meuro	3 … 6
Cineca BG/Q	10000 nodi	160000 core	1.3 Gcore-hour	~20 Meuro	10 … 20
Juqeen	8192 nodi	132000 core	1.1 Gcore-hour	?	
SuperMUC	18432 nodi	147000 core	1.2 Gcore-hour (1.5x)	?	

I centri di calcolo

*A livello **europeo** i grossi centri di calcolo sono strutturati nella organizzazione PRACE, che mette a disposizione ingenti risorse di calcolo sulla base del peer – review*

PRACE-Tier0:
tipicamente due call all' anno ~ 1 Gcore-hour per call
Procedura lunga e complessa ….
… referaggio scientifico e tecnico …
… 6 mesi tra la scadenza della call e la effettiva disponibilita' di tempo macchina

Media del 2011-2012 : 30% del tempo per LQCD
Fair enough…

Solo PRACE??

L' utilizzo dei centri di calcolo e' inevitabile per il prossimo futuro....

*Rimane il problema di un accesso **rapido e flessibile** senza troppi ostacoli e necessita' di pianificazione a lungo termine ....*

Due soluzioni:
Per il calcolo massiccio → Accordo diretto INFN-Cineca
Per il calcolo medio → Il cluster INFN (+accordi Sissa)

SUMA

Accordo INFN-Cineca

Accordo discusso tra varie difficolta' in primavera 2012, in vista dell' istallazione del Blue-Gene/Q

230 Mcore-hours su BG/Q a disposizione dell' INFN tra "giugno"[settembre] 2012 e dicembre 2013

"hopefully" rinnovabile

Coordinamento informale (S. Simula, LT + S. Bassini[Cineca])

Guardiamo al prossimo futuro ….

Ricordate??

1 core di calcolo: *10 → 15 Gflops*
1 nodo di calcolo: *40 → 200 Gflops*

*Un nodo di calcolo di “**prossima generazione” ~ 2000** Gflops*

GPU (Nvidia) – MIC (Intel) - ?

Grazie ad un sostanziale aumento del parallelismo del processore

Intra-node (vs. inter-node)

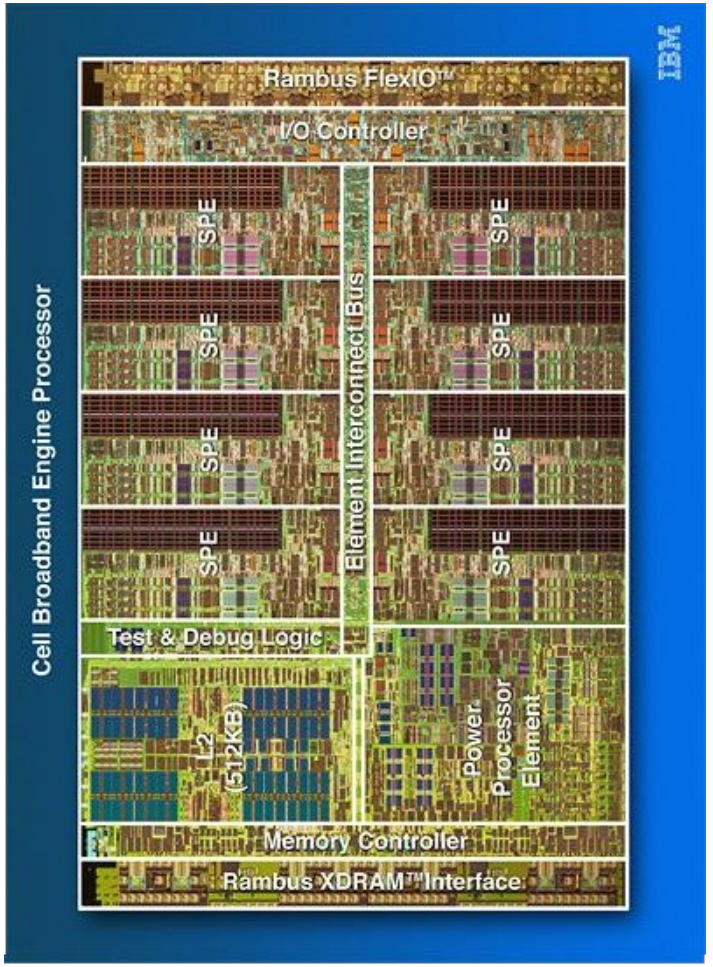
One processor today is many independent processors (cores)

There is a multi-scale environment of data bandwidths and latencies

Cores have to be kept busy at all times: a difficult balancing problem

Consequences: distributing a job on 1000 nodes → easy

On 8-64 cores → difficult



Punto di contatto tra HPC for theory e esperimenti??

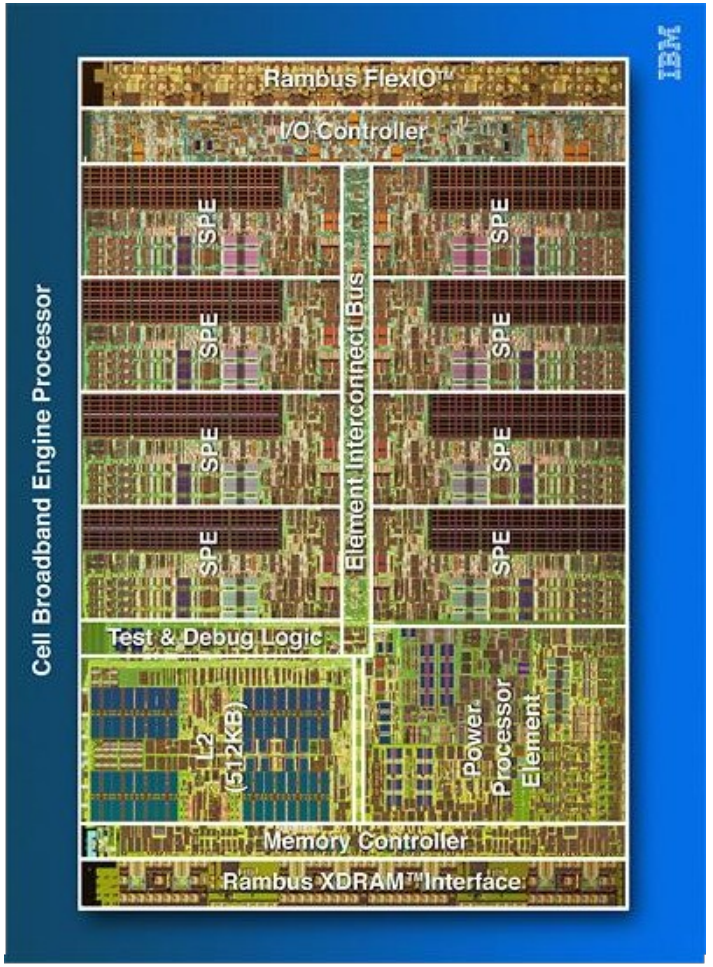
Utilizzare in modo efficiente un intero processore sta diventando sempre piu' difficile

Vari livelli di parallelismo devono integrarsi e bilanciarsi tra di loro

La performance dipende fortemente dall' utilizzo di memoria on-chip

La programmazione (almeno per ora) non puo' ignorare piu' di tanto l' architettura del nodo

.....



Punto di contatto tra HPC for theory e esperimenti??

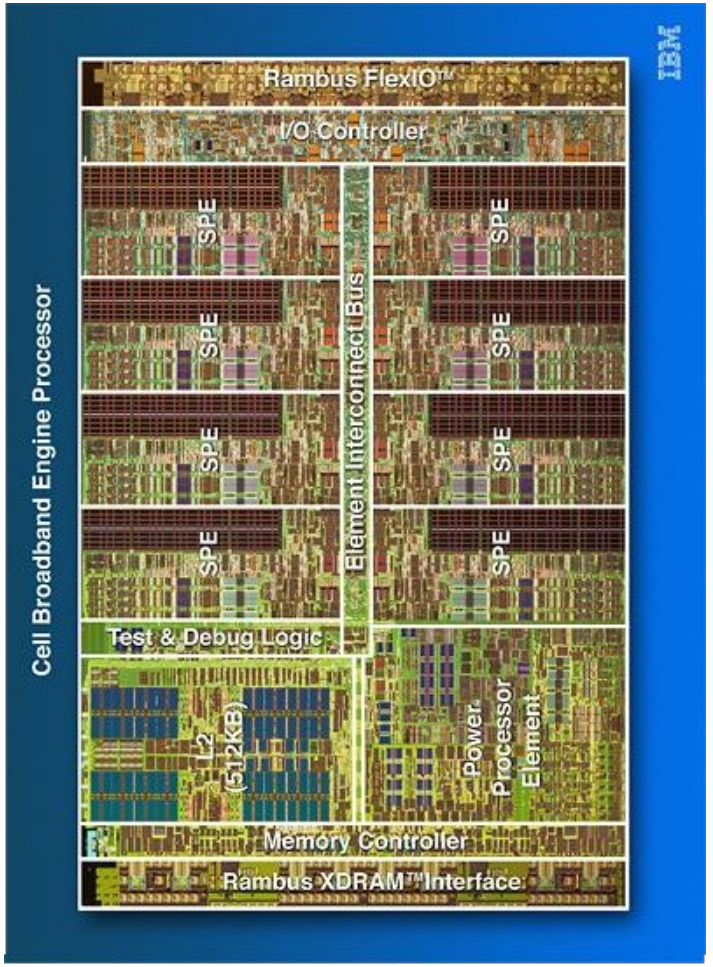
Le architetture HPC proposte per il prossimo futuro sono svariate combinazioni di

- *Mic*
- *GPU*
- *Arm (?????)*

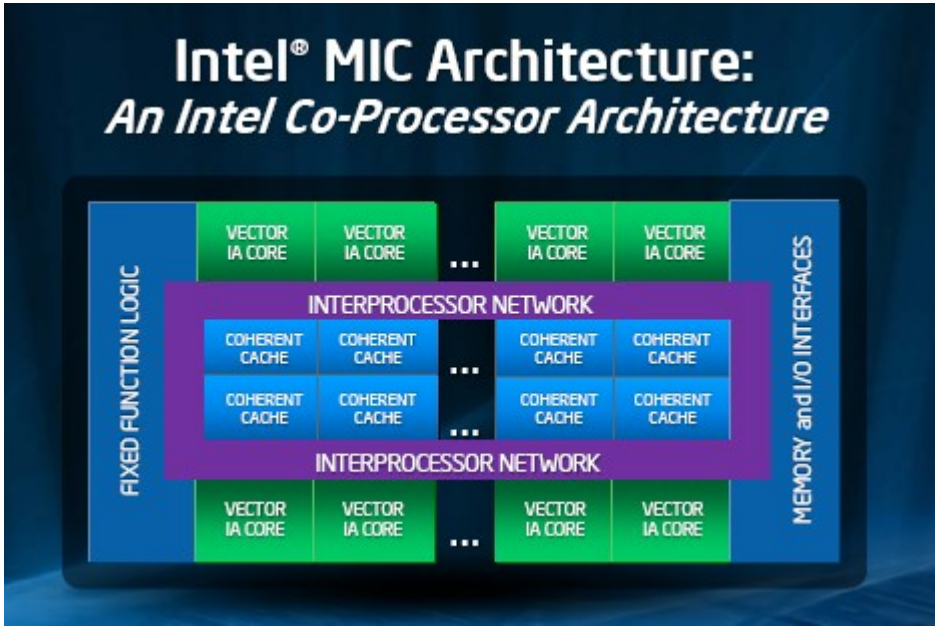
La comunita' HPC sta cercando di capire come usare queste bestie in maniera efficiente da un paio di anni

--------->

Gia' da ora spazio per un travaso reciproco di competenze



Intel MIC processors ….



Un numero alto (ma non altissimo) di core massicci

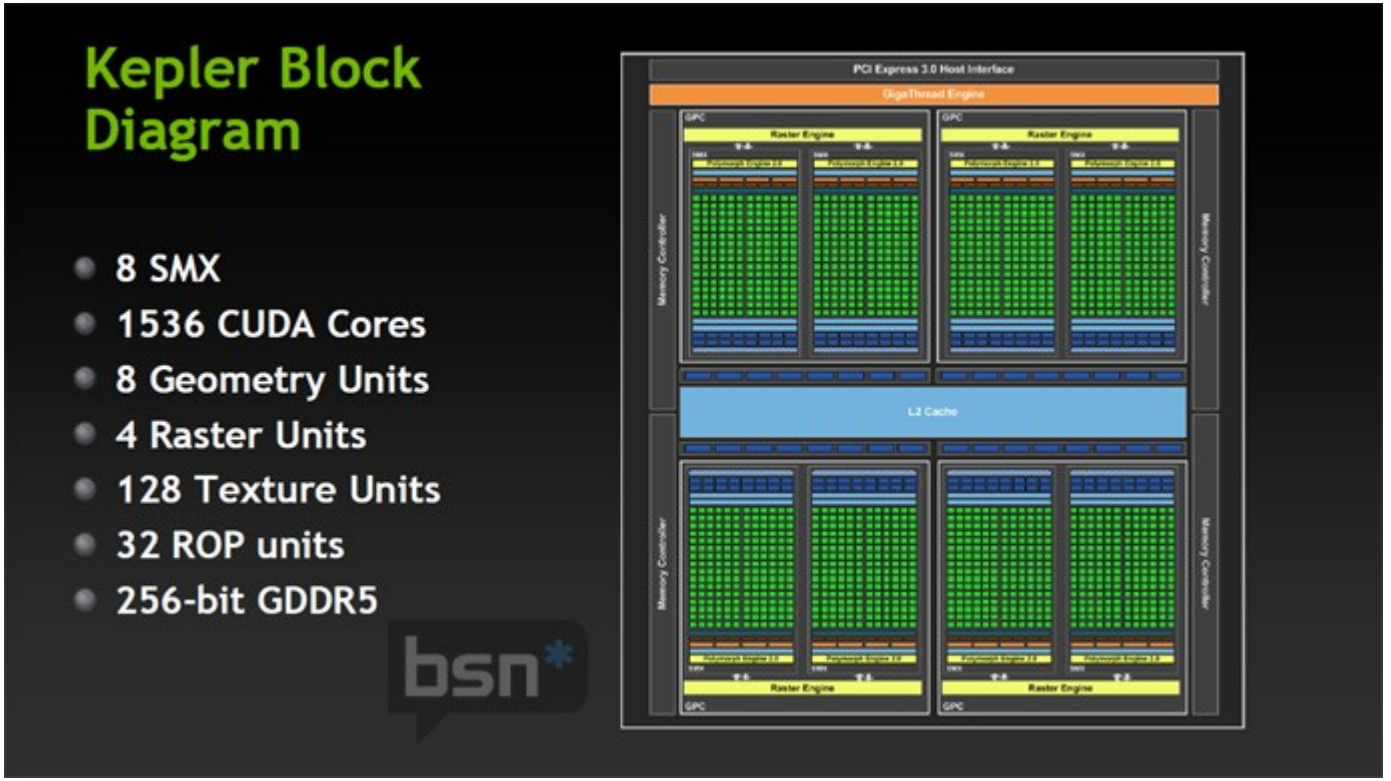
e.g: 60 + 1 cores x 32 Gflops

Tecniche di programmazione "relativamente simili" a quelle attuali ……

… ma avere buone prestazioni non e' facile →

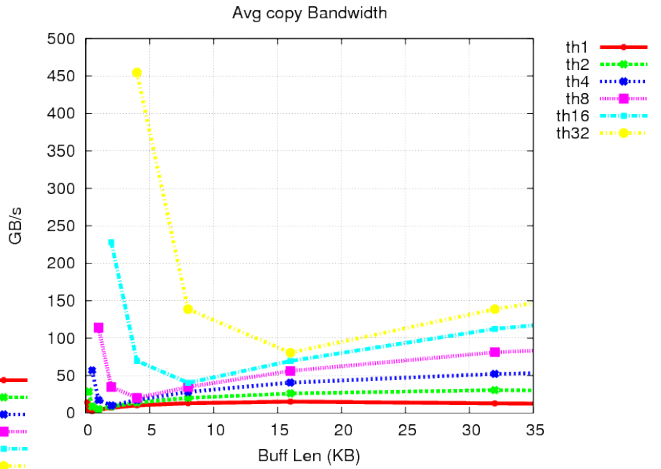
NVIDIA GPU processors ….

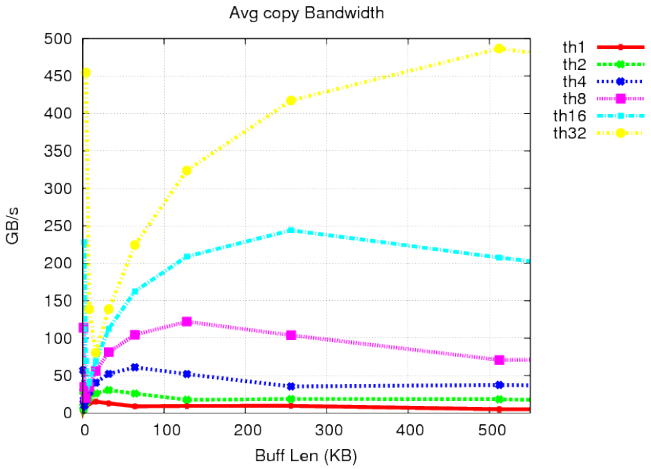
Un numero molto alto di core di calcolo molto semplici,
Programmati con linguaggi ad hoc (CUDA) friendly ed efficaci
ma con un comportamento spesso caotico.

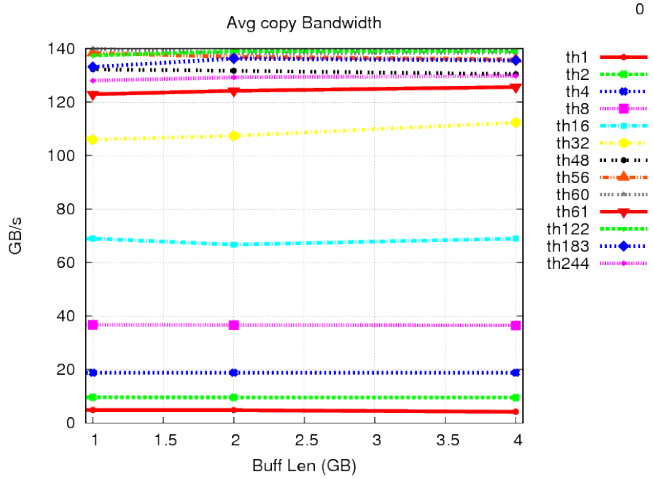


Intel MIC processor: Basic benchmarks

Un analisi accurata della banda processore - "memoria"







Uno studio accurato su un algoritmo Lattice Boltzman

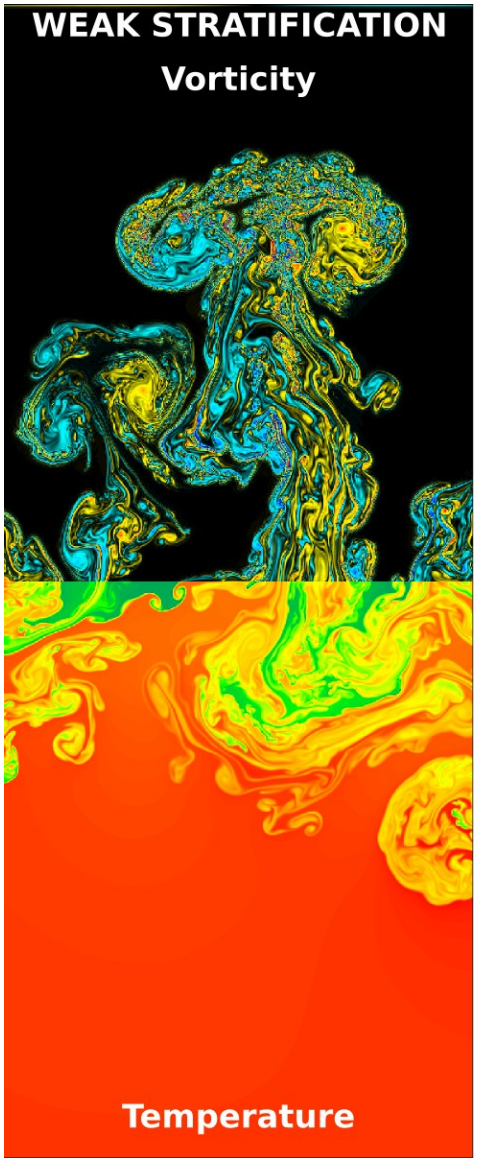
Dinamica dei fluidi in regime turbolento

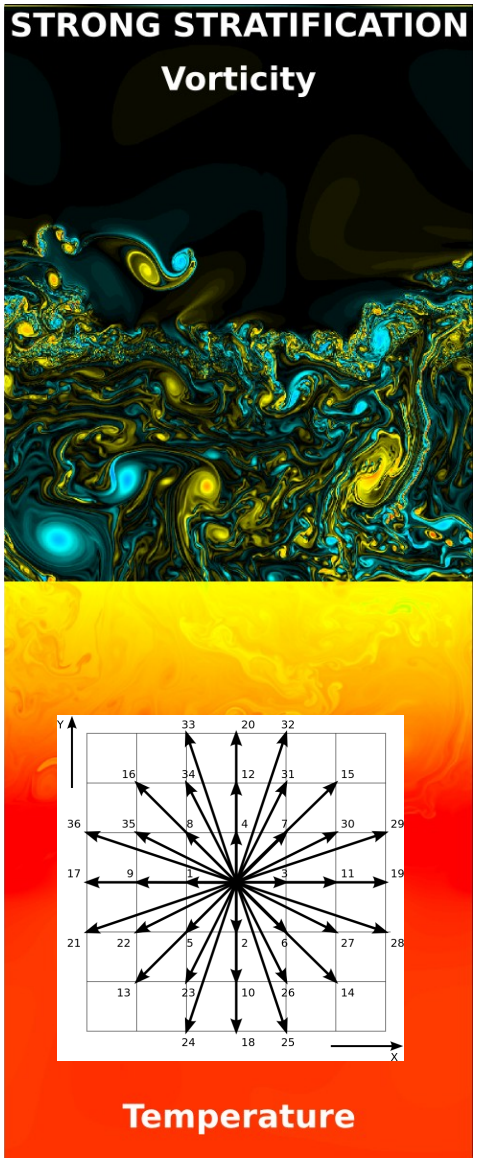
Reticolo discreto (in 2 o 3 dim)

Algoritmica simile alla LGT
Ma struttura piu' semplice

Concettualmente basato su "particell virtuali" che si muovono in un reticolo discreto

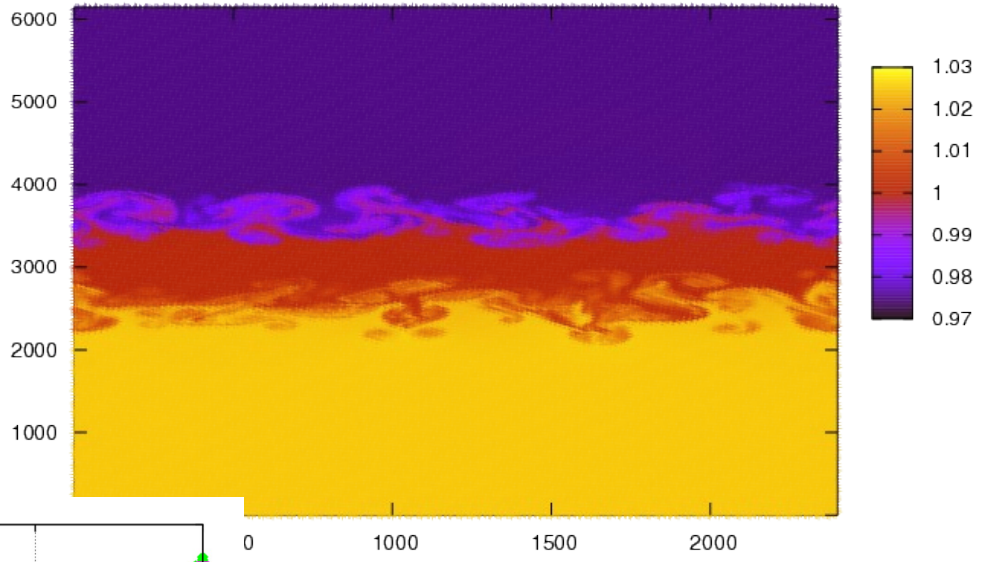
Le quantita' fisiche sono medie
Pesate su queste popolazioni

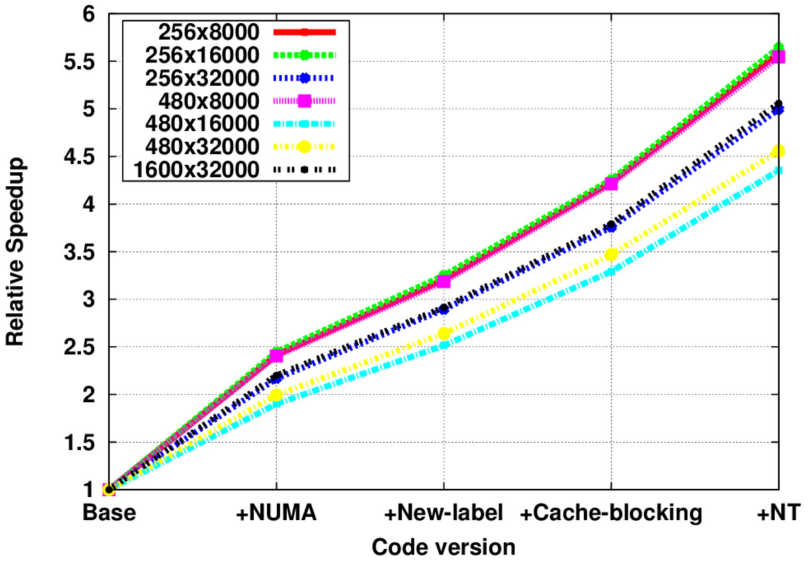




Intel MIC processors e LB

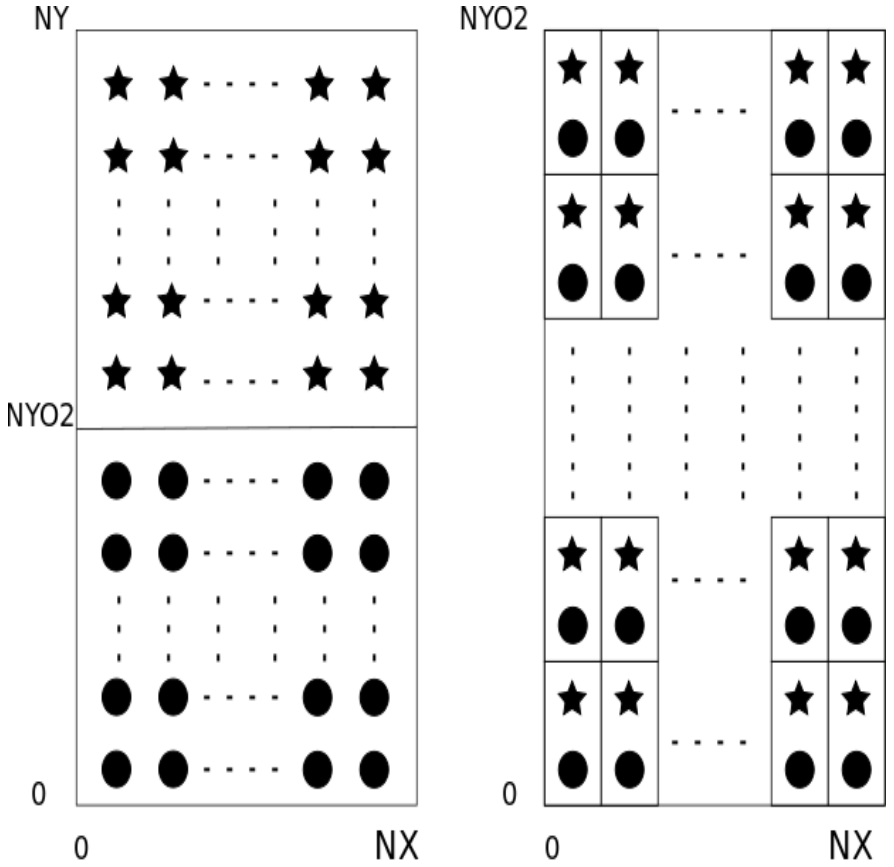
Uno schema di calcolo tra
I piu' friendly per il
Calcolo massicciamente
Parallelo …
(e una palestra per la
LQCD)





Parallelize over intra-core SIMD (vector) operators

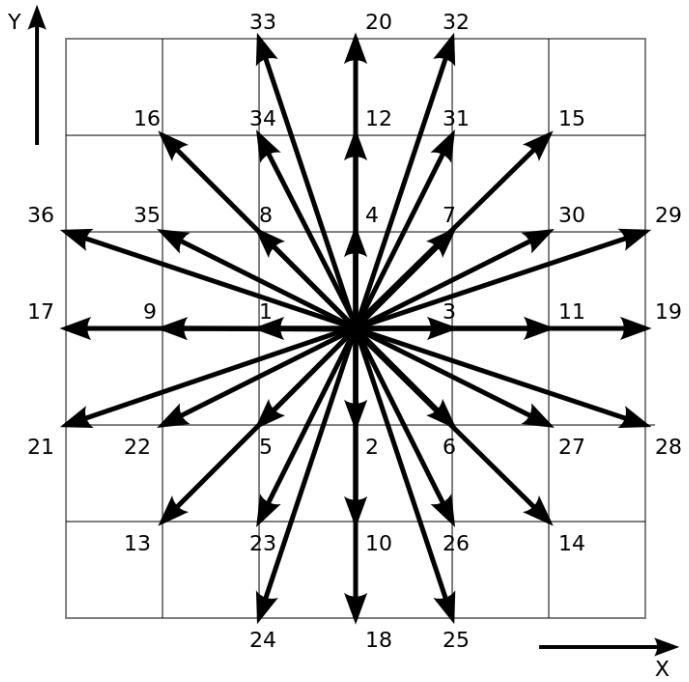
Make sure each floating point operation effectively counts as 2, 4, 8 ….

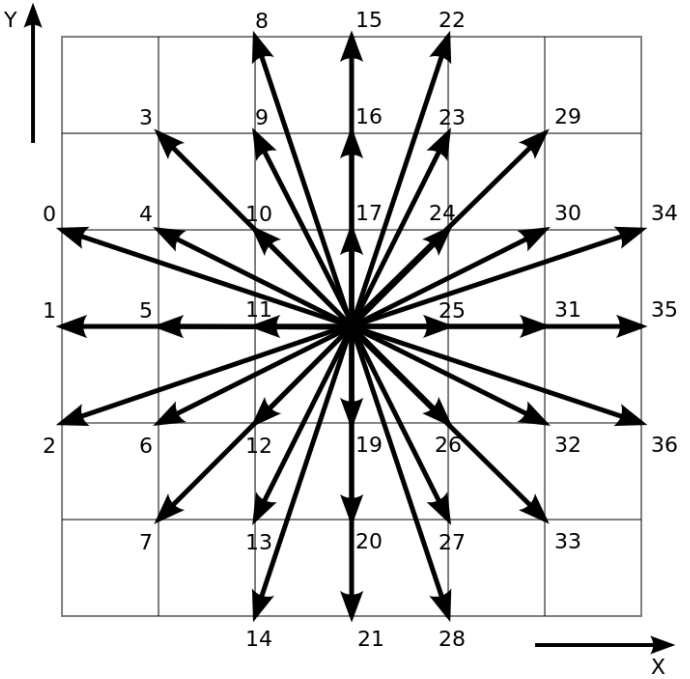


Make memory allocation cache-friendly …

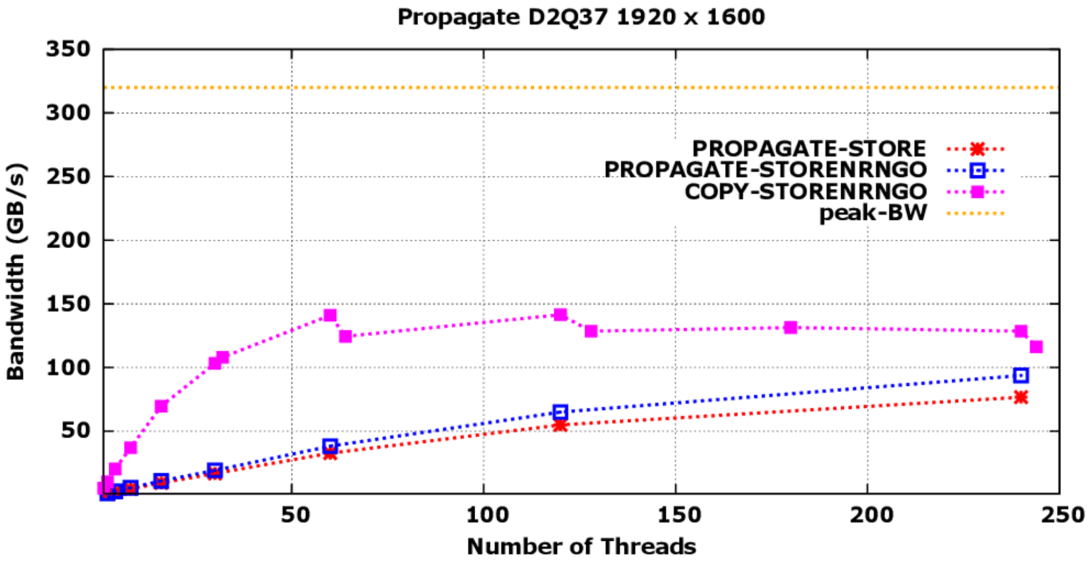
Make sure data retrieved from memory (and parked in the cache) can be reused as many times as possible …

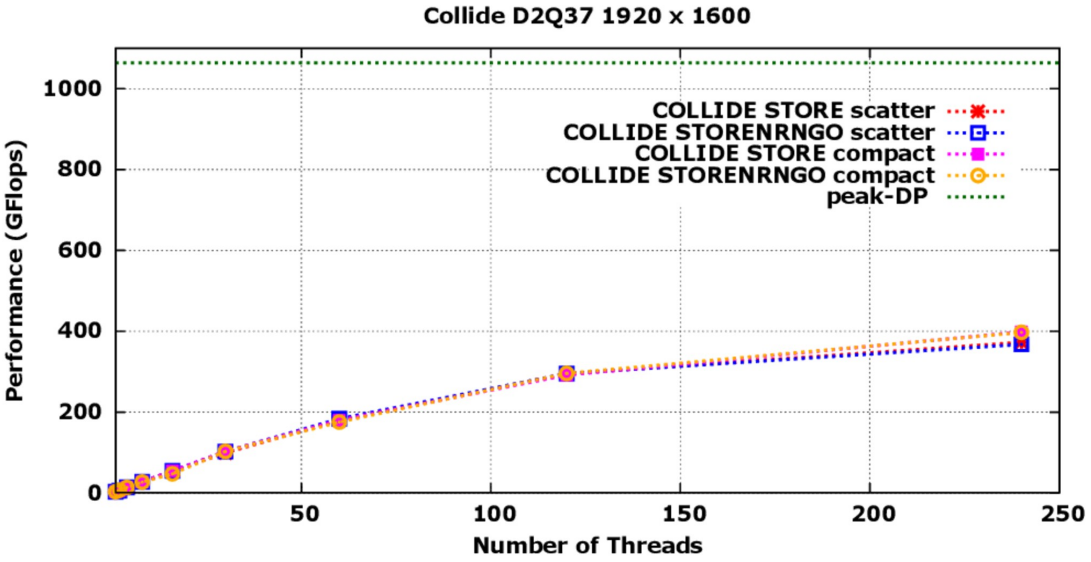
… and also that data belongs to the memory bank close to the processor that most often uses them.



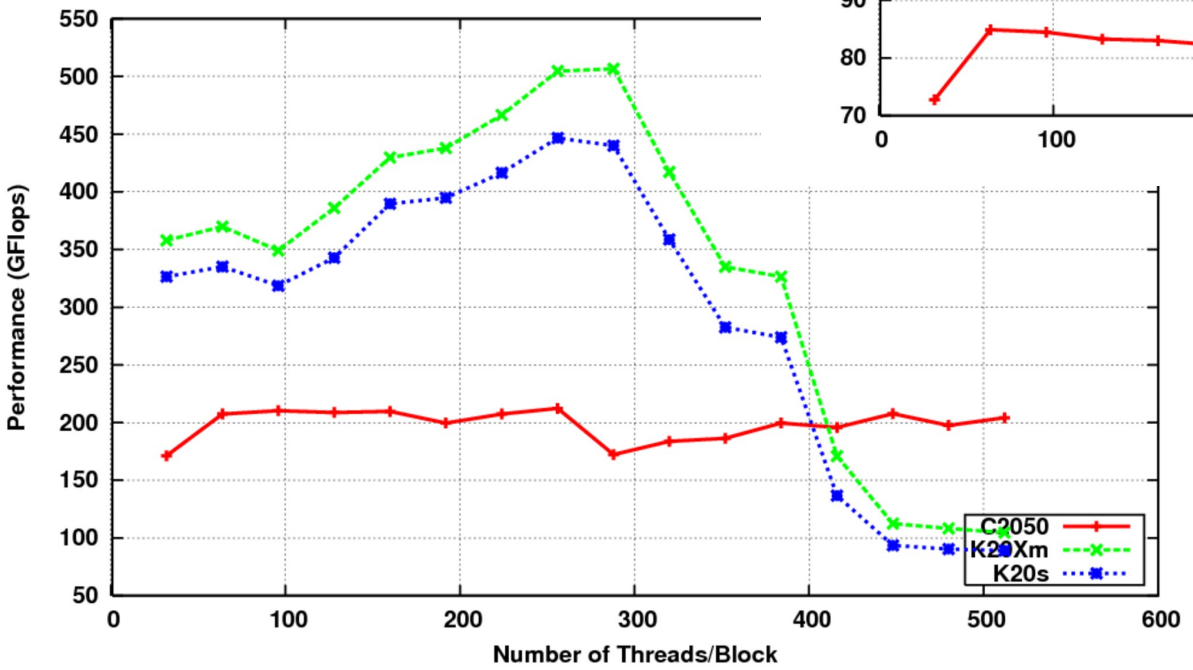


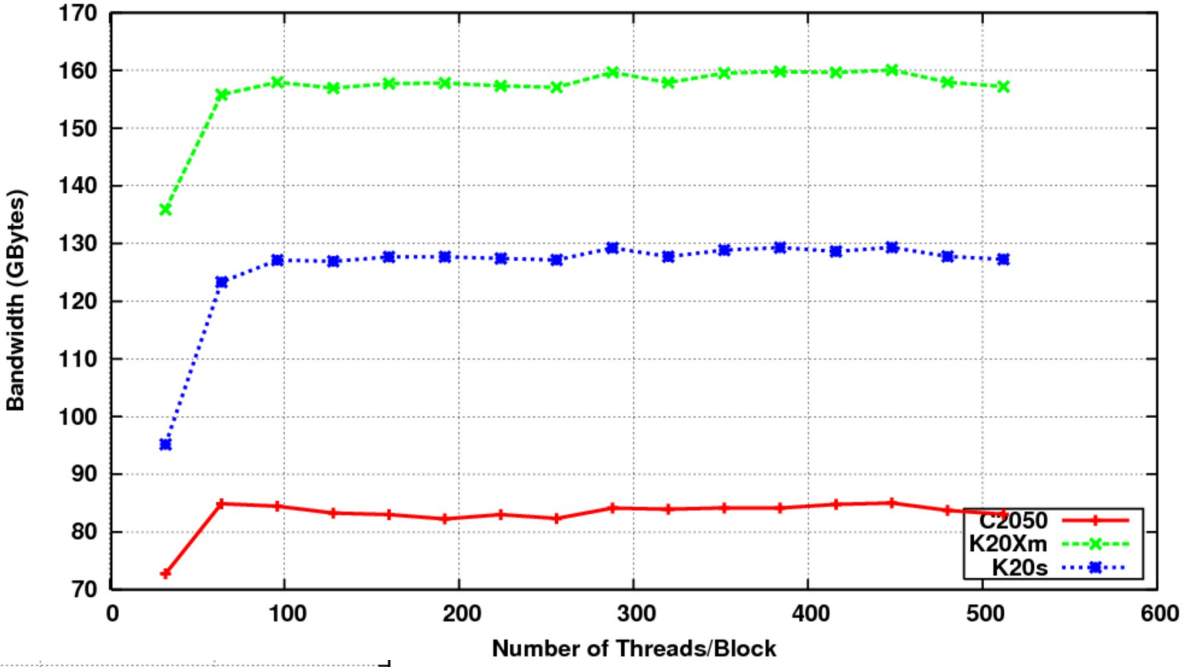
Misure di performance per Intel MIC





Misure di performance per Kepler K20





Conclusioni I (the small picture)

Nel prossimo futuro l' unica via aperta per un sostanziale aumento di prestazioni sono i co-processori multi-core → many-core

Notevole esperienza acquisita in ambito HPC e fisica computazionale

Significativo aumento di prestazioni al costo di un complesso lavoro di adattamento degli algoritmi e di riscrittura dei programmi

Le GPU sono ormai mature ma con ancora un notevole potenziale di sviluppo

I processori MIC stanno ancora muovendo i primi passi e raggiungono Prestazioni modeste anche con un sostaziale sforzo di programmazione

Il problema chiave e' la mancanza di un ambiente di programmazione sufficientemente comodo ma anche decorosamente efficiente

Conclusioni II (the big picture)

In un ottica di uno scambio di know-how all' interno dell' INFN:

La comunita' teorica-HPC e' in grado di fornire:

Informazioni dettagliate sulle "nuove" architetture"
Expertise sul match tra architettura e algoritmo
Modellazione delle performance aspettate
Test-case da usare come riferimento

La comunita' teorica-HPC vorrebbe imparare:

A programmare in maniera un po' piu' "comoda".....

Predicting performance

Well, at least an upper bound on what one may hope to get...

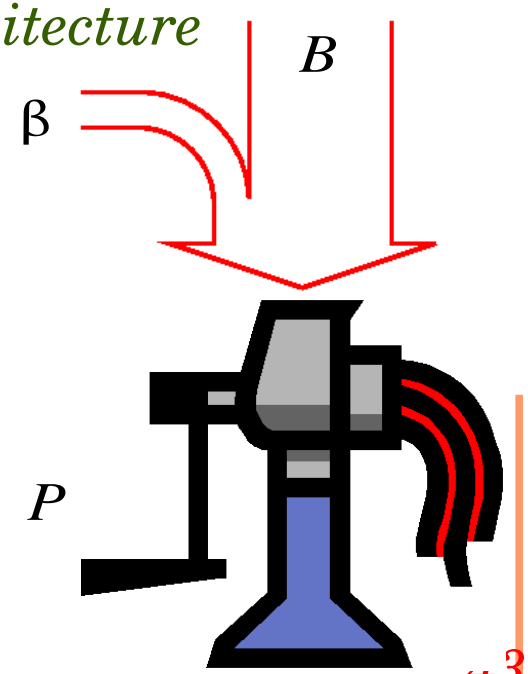
$$T \geqslant max\left\{\frac{W}{nP}, \frac{W}{nRB}, \frac{W}{nR\,\rho\,\beta}\right\} = \frac{W}{nP}\, max\left\{1, \frac{P}{RB}, \frac{P}{R\,\rho\,\beta}\right\}$$

R, ρ *are set by the algorithm (on a given machine / problem size).*

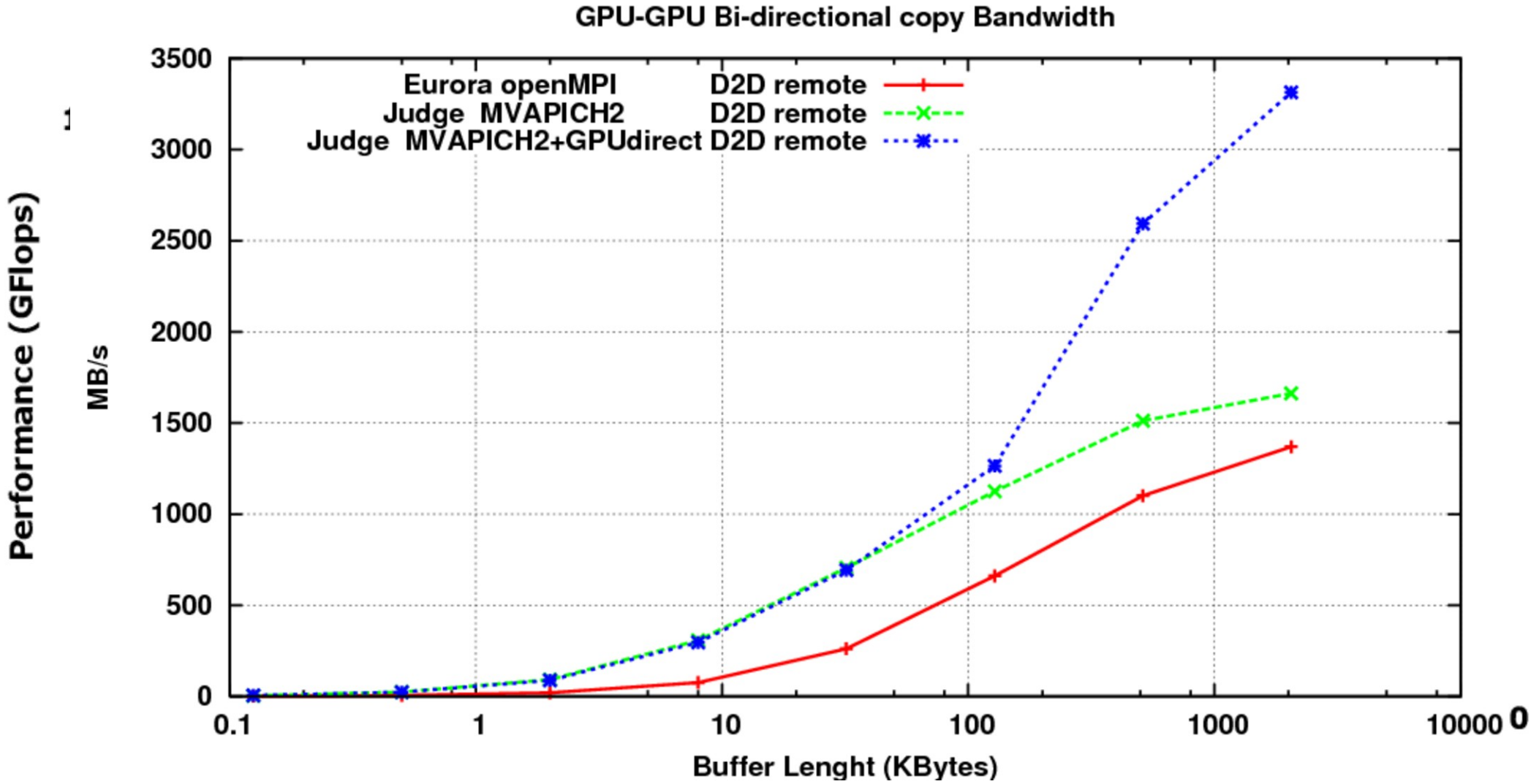
P, B, β *are associated to the target computer architecture*

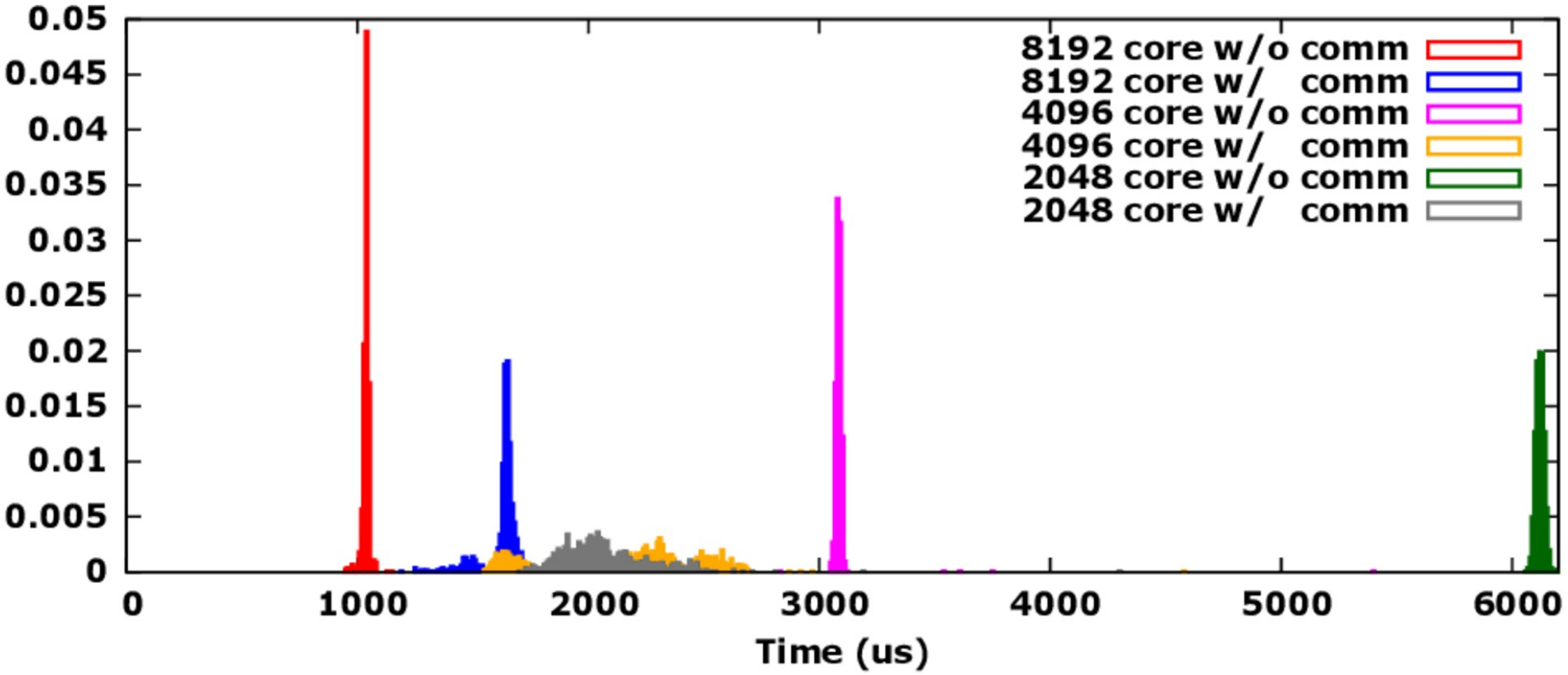
Inserting the appropriate figures →

$$T \geqslant \frac{W}{nP}\, max\left\{1, \approx 0.2, \approx 0.1\right\}$$



Can hope to reach very high efficiency ....


GPU-GPU Bi-directional copy Bandwidth
Eurora openMPI D2D remote
Judge MVAPICH2 D2D remote
Judge MVAPICH2+GPUdirect D2D remote
Performance (GFlops)
MB/s
3500
3000
2500
2000
1500
1000
500
0
0.1
1
10
100
1000
10000
0
Buffer Lenght (KBytes)


0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
1000
2000
3000
4000
5000
6000
Time (us)
8192 core w/o comm
8192 core w/ comm
4096 core w/o comm
4096 core w/ comm
2048 core w/o comm
2048 core w/ comm

… measuring performance

16 nodes (192 cores) Intel Wesmere processors @ 3 Ghz

4032 x 16000 grid points

The starting point ~ 9 – 10 % sustained performance

	Ver. 1.1	Ver. 1.2	Ver. 1.3
T_{pbc}	0.34 s	0.25 s	0.12 s
T_{stream}	0.36 s	0.26 s	0.14 s
T_{bc}	0.9 ms	0.5 ms	0.2 ms
$T_{collide}$	0.39 s	0.39 s	0.39 s
$T_{time/site}$	12.5 ns	11.2 ns	8.7 ns
MLUps	78	89	115
R_{max}	23.8 %	27.0 %	35.2 %

	Ver. 2.1	Ver. 2.3	Ver. 2.3
STEP 1	0.06 s	0.06 s	0.06 s
STEP 2	1.36 ms	1.32 ms	0.64 ms
STEP 3	0.53 s	0.47 s	0.43 s
$T_{time/site}$	9.3 ns	8.7 ns	7.5 ns
MLUps	103	113	130
R_{max}	31.5 %	34.4 %	39.6 %

Preliminary performance on GPUs, ~ 1.5x better ….

Scaling …

One also wants to check that performance stays good on a large window of machine-sizes / physical problem sizes …

A simple back-of-envelope estimate…

$$T_{tot} = c_1 L_y + c_2 \frac{L_x L_y}{N_p}$$

Or, dividing by L_y

$$\frac{T_{tot}}{L_y} = c_1 + c_2 \frac{L_x}{N_p}$$

That is, we have a simple linear relation that should hold for a wide range of processor numbers and grid-sizes

Scaling ...

One also wants to check that performance stays good on a large window of machine-sizes / physical problem sizes ...

48 ... 192 cores / 2000 x 3600 .... 7600 x 32000