Chasing efficiency for Lattice QCD: some issues on current architectures

F. Di Renzo

University of Parma and INFN
AuroraScience Collaboration
STRONGnet

New Frontiers in Lattice Gauge Theory
GGI, Firenze September 26th 2012

Prolegomena

The AuroraScience collaboration has been running an AURORA 15 TFlops prototype for a couple of years. The machine is a highly dense, liquid cooled parallel system, based on Intel multi-core Xeon CPUs and endowed with both an IB and a custom TORUS network. AuroraScience is an FBK/INFN joint initiative, in collaboration with Eurotech.

A strong trend in current (super)computer architectures is an increasing number of cores per node: multicore, manycore (e.g. MIC), GPUs.
In this talk I will focus on a system based on multicore, i.e. Aurora.
A fingerprint of the machine is the double network: IB and custom 3D-Torus.
Aurora can substain fairly good performances in Lattice QCD.

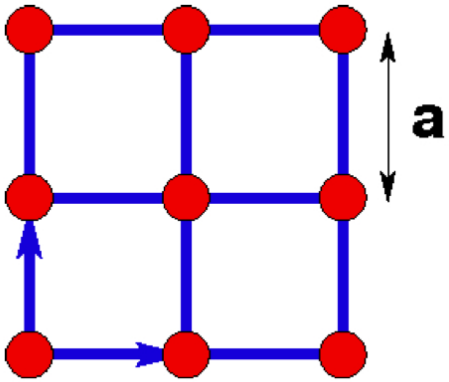
OUTLINE

- Single node performances
- Multi-node MPI over IB performances
- Multi-node TORUS performances

Our basic game: LQCD on a parallel machine

The basic building block of LQCD computations is the (Wilson)Dirac operator, i.e. a (sparse) matrix acting on (multi-indeces) vectors.

$$\psi'_{\alpha i}(x) = \sum_{\mu=1}^{4} (U_\mu^{ij}(x)(1+\gamma_\mu^{\alpha\beta})\psi_{\beta j}(x+\hat{\mu}) + U_\mu^{\dagger ij}(x-\hat{\mu})(1-\gamma_\mu^{\alpha\beta})\psi_{\beta j}(x-\hat{\mu}))$$



- $U_\mu^{ij}(x)$ 3x3 complex matrices, residing on links
- $\gamma_\mu^{\alpha\beta}$ 4x4 complex matrices, sparse (4 complex)
- $\psi_{\alpha i}(x)$ 4x3 complex spin-color, residing on sites

All toghether, 1320 FP on (9x12+8x9 complex) 360 FP words per site.
All toghether, L^3T sites in a lattice: HOW MANY on a COMPUTING NODE?
This choice defines communication vs computation requirements.
Needless to say, you do not make your choice once and for all.

A game we have been playing for a while ...

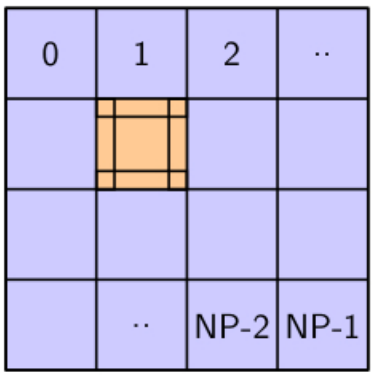
LQCD dates back to 1974; in the mid of the 80's, the LQCD community started thinking of going parallel. What to look for?

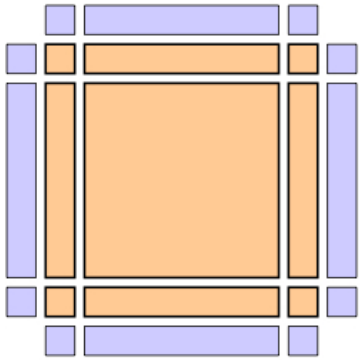
In the end, since then we have been kept on looking at the same formula

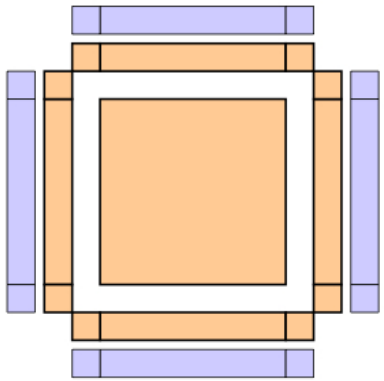
$$T = \max\{\frac{C}{NP}, \frac{I}{NB}, \frac{I_R}{B_R}\} = \frac{C}{NP}\max\{1, \frac{IP}{CB}, \frac{I_R NP}{CB_R}\}$$

being C/N the computational cost per node, P the computational performance, I/N the local exchange of information per node, B the memory bandwith, I_R the *remote* exchange of information and B_R the corresponding bandwith.

A useful cartoon to have in mind with this respect



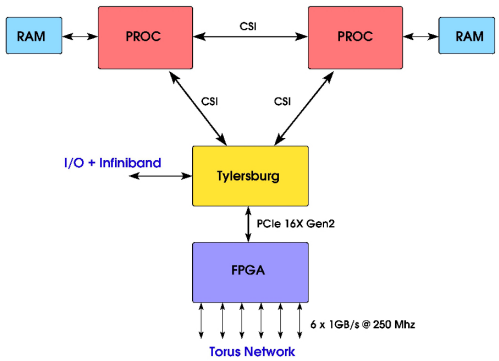


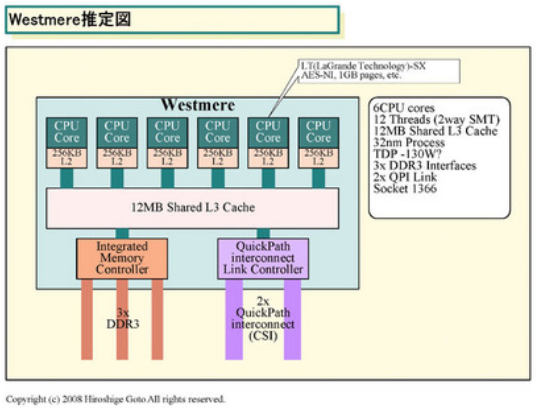


$T = \max\{\frac{C}{NP}, \frac{I}{NB}, \frac{I_R}{B_R}\}$: what is a node?

We said that C/N *is the computational cost per node*.

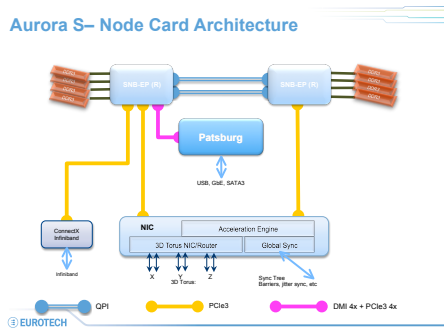
But what is a node on nowadays machines?

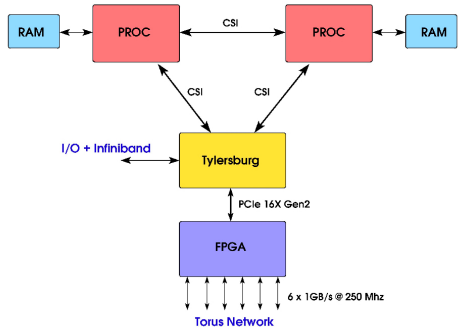


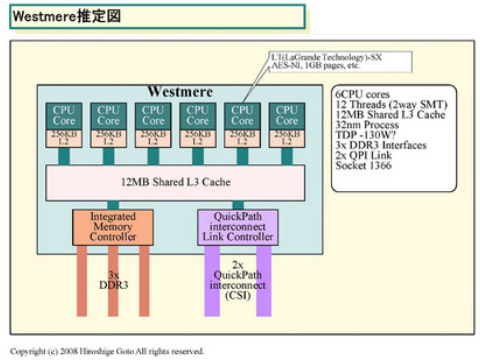


We have to live with intra- and inter- node parallelism. From the bandwith point of view this means that not only B_R (you go remote when you go through the IB or TORUS network), but also B is a non trivial parameter to deal with.

$T = \max\{\frac{C}{NP}, \frac{I}{NB}, \frac{I_R}{B_R}\}$: looking better into local bandwiths







- The Aurora node is a 2-CPU (Westmere or SandyBridge) SMP system.
- Each CPU (we discuss W) has 6 cores (plus Hyperthreading! gain 1.2)
- The 6 cores share an L3 cache memory and access DDR3.
- To fecth from the other CPU one has to go through QPI!
- One can optimize the size of the sub-lattice each core is in charge of ...
- ... only if one takes care of core affinity and memory binding!

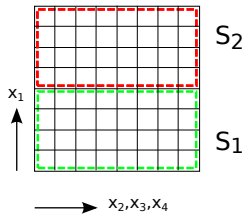
$T = \max\{\frac{C}{NP}, \frac{I}{NB}, \frac{I_R}{B_R}\}$: carefully balancing the bandwiths

Examples of different performances for different node (sub-)lattice data layout and sizes: Wilson Dirac operator application

Single node performances (Parma group)

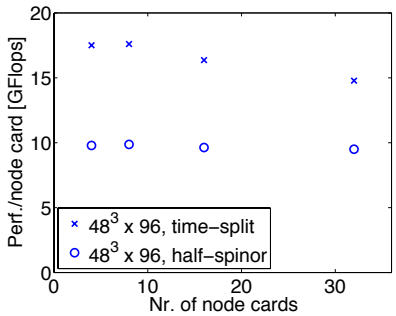
- SSE intrinsics
- MPI/multithread:
  1 rank per node + (HT) *Pthread*
- optimization of L1/L2/L3 usage
- core affinity and memory binding

```
__m128d x2 = _mm_mul_pd(R2,(v+i)->whr[0].m);
v2 = _mm_addsub_pd(_mm_shuffle_pd(x2,x2,1), ...
```



lattice size	Gflops
$8 \times 4 \times 24^2$	52.6
$8 \times 12 \times 48^2$	37.7
$12^2 \times 48^2$	30.4
$12 \times 24 \times 48^2$	26.8
$12 \times 48^2 \times 96$	22.5

Strong scaling of different variants of ETMC code: remote communications via MPI on IB. (L. Scorzato)

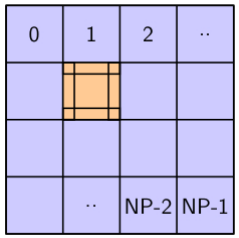


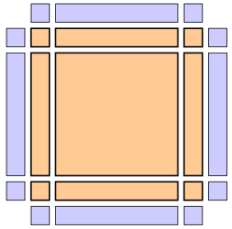
Playing around with data layout, i.e.

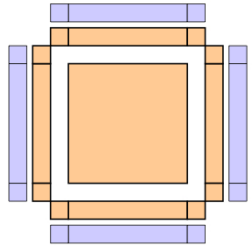
- lattice vs sublattices
- data ordering

one can gain quite a lot.

$T = \max\{\frac{C}{NP}, \frac{I}{NB}, \frac{I_R}{B_R}\}$: remote communication basics



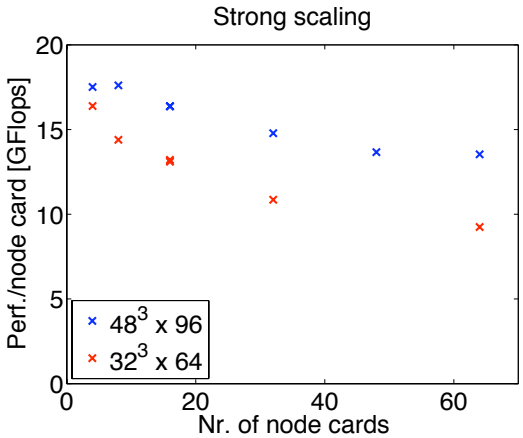




A pragmatic bottom line:

- One wants to hide commications by overlapping them with computations
- Efficient node data layout is not immediately ready for remote data exchange

MPI solutions are well known (L. Scorzato)



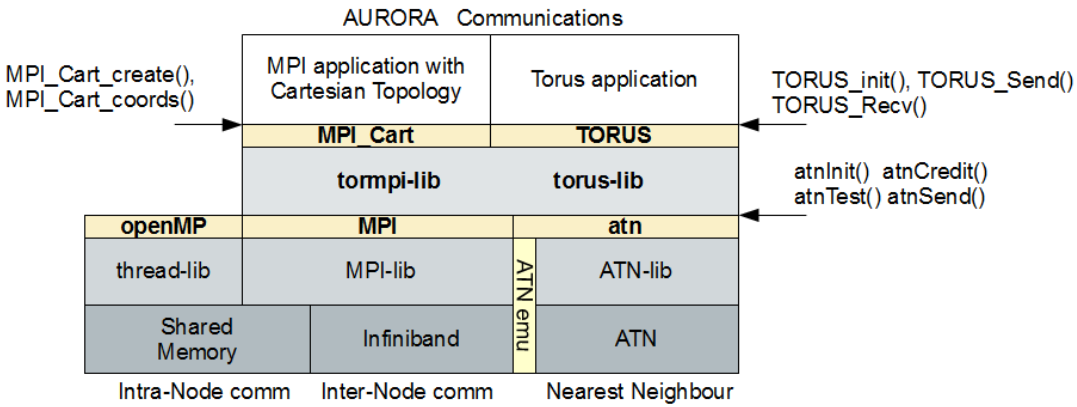
- Use non-blocking communications
- MPI can pack & unpack data for you

A little detour: Aurora communications basic

To exploit the TORUS custom network (FTNW, by M. Pivanti, F.S. Schifano, H. Simma), user programs can call both low level atn communication functions (threads) ...

```
int atnSend   (uint lid, uint cid, void * txbuf, uint txoff, uint len);
int atnCredit (uint lid, uint cid, uint rxoff, uint len, uint nid);
int atnPoll   (uint lid, uint cid, uint rxoff, uint len, void * rxbuf, uint nid);
int atnTest   (uint lid, uint cid, uint rxoff, uint len, void * rxbuf, uint nid);
```

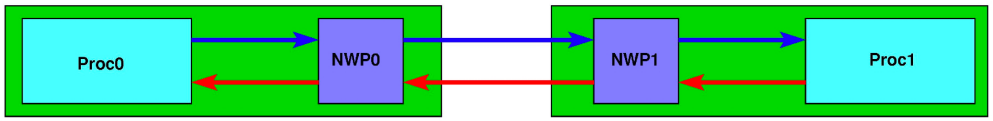
... and high level TORUS or torMPI communication functions (processes). All toghether, we have a composite environment

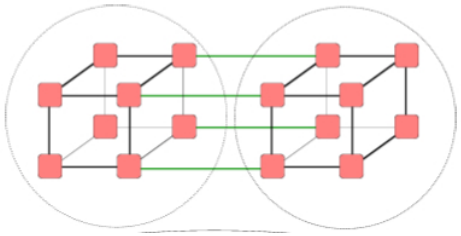


- torMPI mimics MPI
- TORUS focuses nearest neighbor communications (3Dtorus + shared mem)
- they both at the moment rely on a *proxy* process
- TORUS + MPI can be better than MPI ...

$T = \max\{\frac{C}{NP}, \frac{I}{NB}, \frac{I_R}{B_R}\}$: going remote via TORUS

TORUS network basics



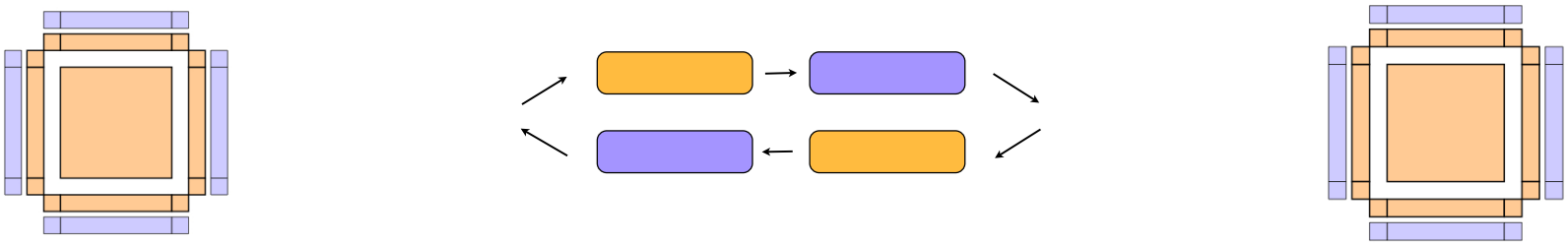


- Credit/Send/Poll mechanism
- Data should be aligned
- Virtual Channels mechanism
- Natural support of multithread

Basic ingredients of our Wilson Dirac Operator basic routine

- We directly manage the (aligned) buffers for borders-exchange ...
- ... which are binded to relevant sockets (like all our data)
- Hyperthreading is in place ...
- ... which is a natural playground for overlapping computations and communications
- Thread load allocation changes during execution
- We go even/odd

The final goal: Wilson Dirac operator kernel

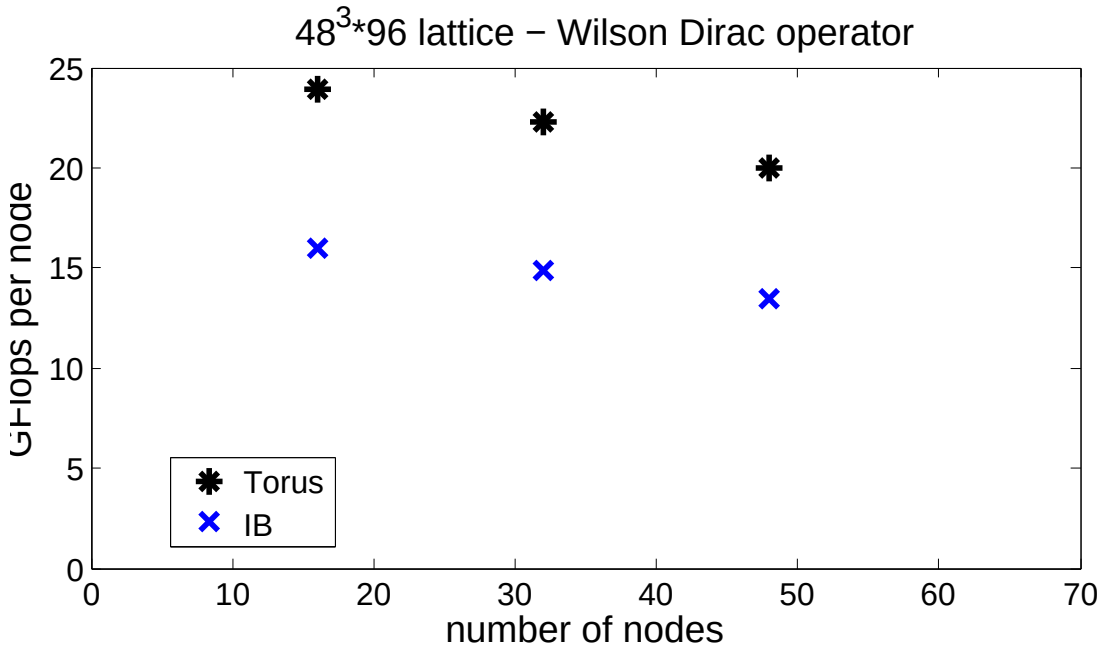


$$\psi'_{\alpha i}(x) = \sum_{\mu=1}^{4} (U_{\mu}^{ij}(x)(1+\gamma_{\mu}^{\alpha\beta})\psi_{\beta j}(x+\hat{\mu}) + U_{\mu}^{\dagger ij}(x-\hat{\mu})(1-\gamma_{\mu}^{\alpha\beta})\psi_{\beta j}(x-\hat{\mu}))$$

- Prepare borders (*) and store them (half spinor!) into dedicated buffers
- Half of the threads update the bulk, the remaining half exchange borders (HT: no competition on FP resources!)
- Notice that bulk is actually a bit non-trivial as a concept (one can re-allocate threads, if needed)
- Reconstruct border contributions (*)

The final goal: it works pretty well

Wilson Dirac Operator: MPI/IB and atn/TORUS



Peak performance percentage comparable with ETMC code on the BG/P (but with bigger node granularity).

NSPT - both IB and TORUS networks in place!

The master formula for inverting (order by order) the Dirac operator ($\psi = M^{-1}\xi$):

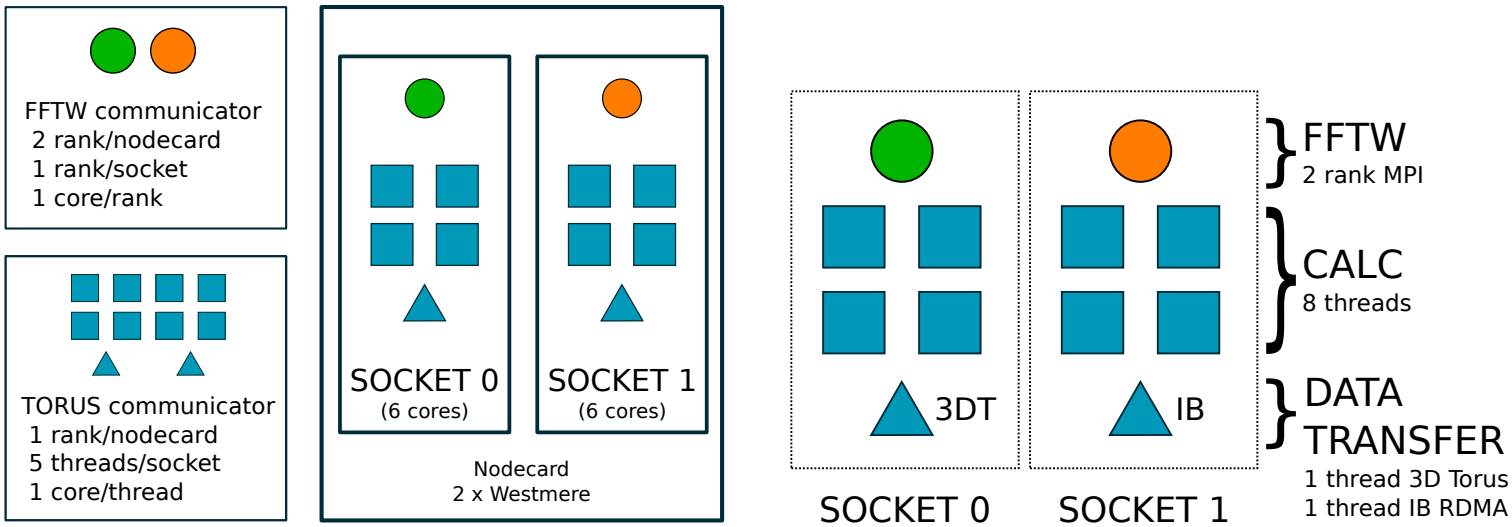
$$
\begin{aligned}
\psi^{(0)} &= M^{(0)\,-1}\xi \\
\psi^{(1)} &= -M^{(0)\,-1} M^{(1)}\psi^{(0)} \\
\psi^{(2)} &= -M^{(0)\,-1}\left[M^{(2)}\psi^{(0)} + M^{(1)}\psi^{(1)}\right] \\
\psi^{(3)} &= -M^{(0)\,-1}\left[M^{(3)}\psi^{(0)} + M^{(2)}\psi^{(1)} + M^{(1)}\psi^{(2)}\right] \\
&\dots \\
\psi^{(n)} &= -M^{(0)\,-1}\sum_{j=0}^{n-1} M^{(n-j)}\psi^{(j)}
\end{aligned}
$$

We notice that

- $M^{(0)-1}$ is diagonal in momentum space, $M^{(i)}$ (almost) diagonal in configuration space: go back and forth from momentum space via FFT! $M^{(i)}$ (brackets) computations in configuration space; FFT before we apply $M^{(0)-1}$ and inverse FFT to make the $\psi^{(i)}$ available to following orders computations;
- once $\psi^{(i)}$ has been computed, advance in the computation of the (configuration space) brackets! computations can overlap;
- torus network for configuration space, IB network can substain FFT.

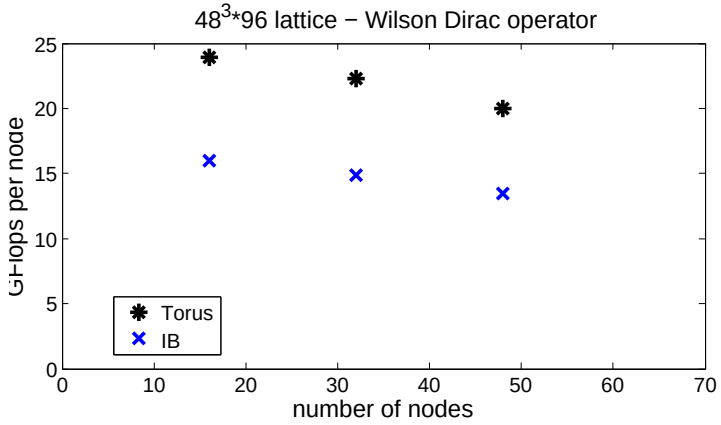
NSPT - an example that profits from both networks

- The overall MPI application has two communicators (FFT and TORUS comm.).
- On each nodecard, we have three MPI processes, or ranks (in MPI jargon).
- Two ranks on each nodecard belong to the MPI communicator; they are single thread processes, residing on two physical cores (one per socket).
- The third rank on each nodecard is a multi-thread process, taking the ten residual cores available on the nodecard. Eight threads in charge of computations; one core in charge of communications on the torus network; one core in charge of the (RDMA) MPI communications which cross-exchange data FFT $<->$ TORUS.



Conclusions

- Intra- and inter- node parallelism is a challenge for nowadays parallel architectures.
- Core affinity and memory binding crucial on modern multicore systems.
- Hiding communications by computations is the main issue: dedicated networks are still a valuable tool.



The Aurora system will have a follow-up: Eurotech is going to install EURORA systems (MIC!)