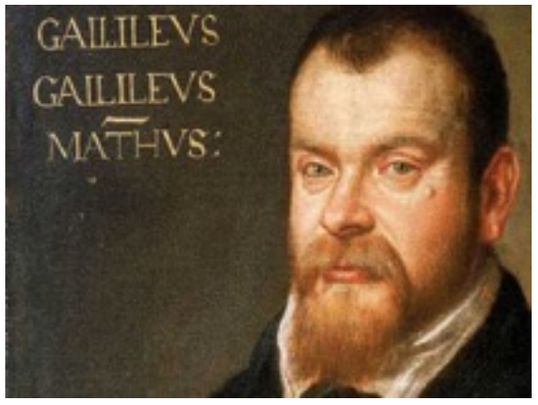
QUDA and QPhiX interfaces for tmLQCD

Mario Schröck

Bonn, May 28, 2015

Motivation

Galileo @ Cineca

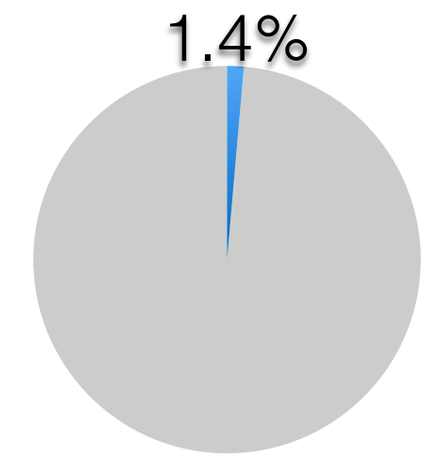


- 516 compute nodes:
- two octa-core Intel Xeon Haswell CPUs (E5-2630 v3 @ 2.40GHz) per node
- 128 GB RAM per node
  - two 16GB Intel Xeon-Phi 7120P (MIC) on 384 nodes
  - two 24GB NVIDIA K80 GPUs on 40 nodes
  - ≳ one Petaflop

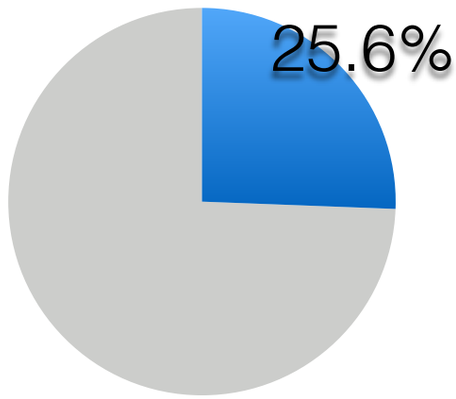
➢ we want to use this machine for propagator calculations!

Galileo performance for (tm)LQCD

- tmLQCD mixed prec. CG-eo on CPU
- best performance: one MPI proc./core: 27 Gflops
- 516 CPUs x 27 Gflops (naiv scaling)



Galileo: 1Pflops

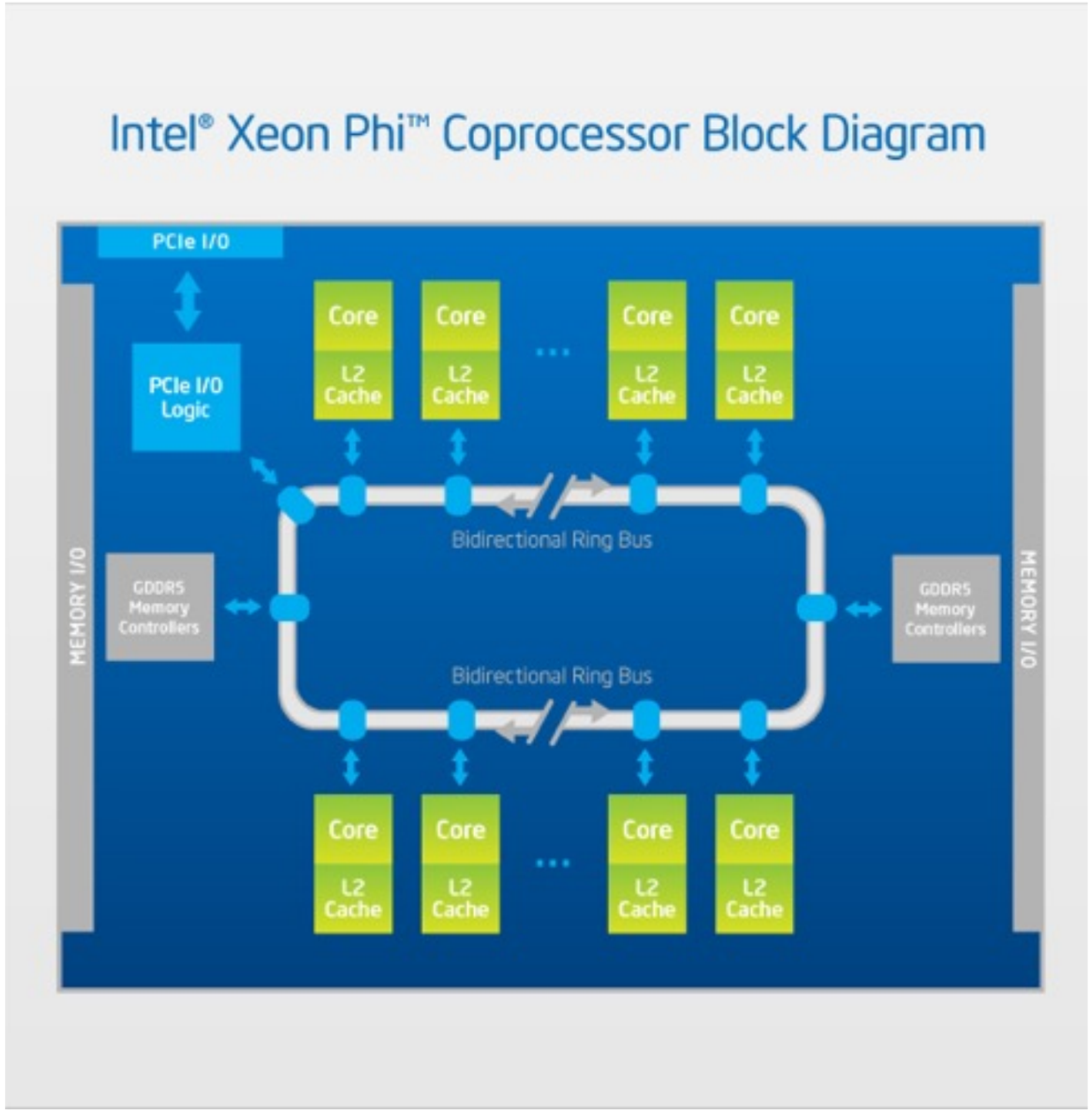


Fermi: 2 Pflops

- compare to Fermi BlueGene/Q
- 10.240 nodes x ~50 Gflops (ideal 12^4 local lattice)

Intel Xeon Phi 7120P

- 61 cores
- 4 threads/core
- 512-bit SIMD vector unit/core
- 512KB L2 cache/core



Can we run tmLQCD natively on MIC?

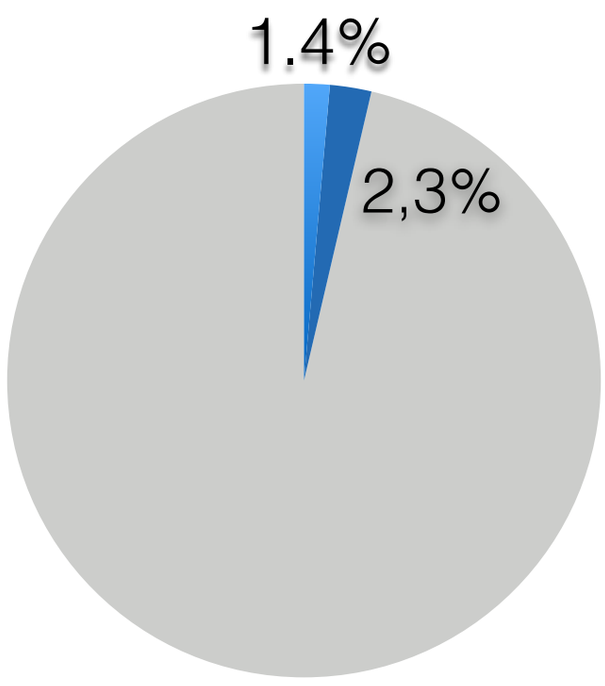
Yes, simply cross-compile with flag -mmic:

- 1 MPI proc., 244 OMP threads: 20.8 Gflops
- 2 MPI proc., 122 OMP threads: 24.1 Gflops
- 4 MPI proc., 61 OMP threads: 26.0 Gflops
- 16 MPI proc., 15 OMP threads: 29.4 Gflops

Performance comparable to CPU *but* only 1/16 of the accounting costs (Cineca policy)!

Galileo performance for tmLQCD

- including native runs on MIC (768 devices)



Galileo: 1Pflops

Interfaces for tmLQCD

Interfaces for tmLQCD

- own code for various types of hardware is difficult to maintain
- a lot of manpower has been put into libraries like QUDA, hard to compete with those
- easy to test correctness of inverter results from external libraries

QPhiX: QCD for Intel Xeon (Phi) processors

(first release 09/2014) offers Dirac operators and solvers for:

- Wilson
- Clover-improved Wilson
- Twisted mass (with Alexei Strelchenko)

Original Authors:

- Balint Joo *(Jefferson Lab)*
- D. Kalamkar, K. Vaidyanathan, M. Smelyanskiy *(Intel Parallel Computing Labs)*

QUDA: A library for QCD on GPUs

QUDA (first release 2009) is leveraging NVIDIA's CUDA platform, is 4D MPI parallel and it includes optimized Dirac operators and solvers for:

- Wilson
- Clover-improved Wilson
- **Twisted mass (including non-deg. doublets)**
- **Twisted mass with clover term**
- Staggered fermions and Asqtad/HISQ
- Domain wall (4-d or 5-d preconditioned)
- Mobius fermions

QUDA Authors

Ronald Babich *(NVIDIA)*
Kipton Barros *(Los Alamos National Laboratory)*
Richard Brower *(Boston University)*
Mike Clark *(NVIDIA)*
Justin Foley *(University of Utah)*
Joel Giedt *(Rensselaer Polytechnic Institute)*
Steven Gottlieb *(Indiana University)*
Balint Joo *(Jefferson Laboratory)*
Claudio Rebbi *(Boston University)*
Guochun Shi *(NCSA)*
Alexei Strelchenko *(Fermi National Laboratory)*
Alejandro Vaquero *(INFN Sezione Milano Bicocca)*
Mathias Wagner *(Indiana University)*

Interface design goals

1. **Safety.** Final residual always checked by tmLQCD
2. **Ease of use.** Set flag `UseQudaInverter` in input file
3. **Minimality.** Single separate file + few `#ifdef QUDA`
4. **Performance.** Will not be 100% by default due to (1.)

For implementation details see the
tmLQCD documentation!
(currently only in my fork)

Installation & Usage

Installation

- QUDA can be installed without any dependencies
- configure tmLQCD with three additional settings:

```
./configure CC=mpicc \
--prefix=$TMLQCDDIR \
--with-limedir=$LIMEDIR \
--with-lapack=<linker-flags> \
--enable-mpi \
--with-mpidimension=4 \
CXX=mpiCC \
--with-qudadir=$QUDADIR \
--with-cudadir=${CUDADIR}/lib
```

Usage example

- run the `invert` executable and adjust the operator(s) in the input file:

```
BeginOperator TMWILSON
  2kappaMu = 0.05
  kappa = 0.177
  UseEvenOdd = yes
  Solver = CG
  SolverPrecision = 1e-14
  MaxSolverIterations = 1000
  UseQudaInverter = yes
EndOperator
```

- supported operators:

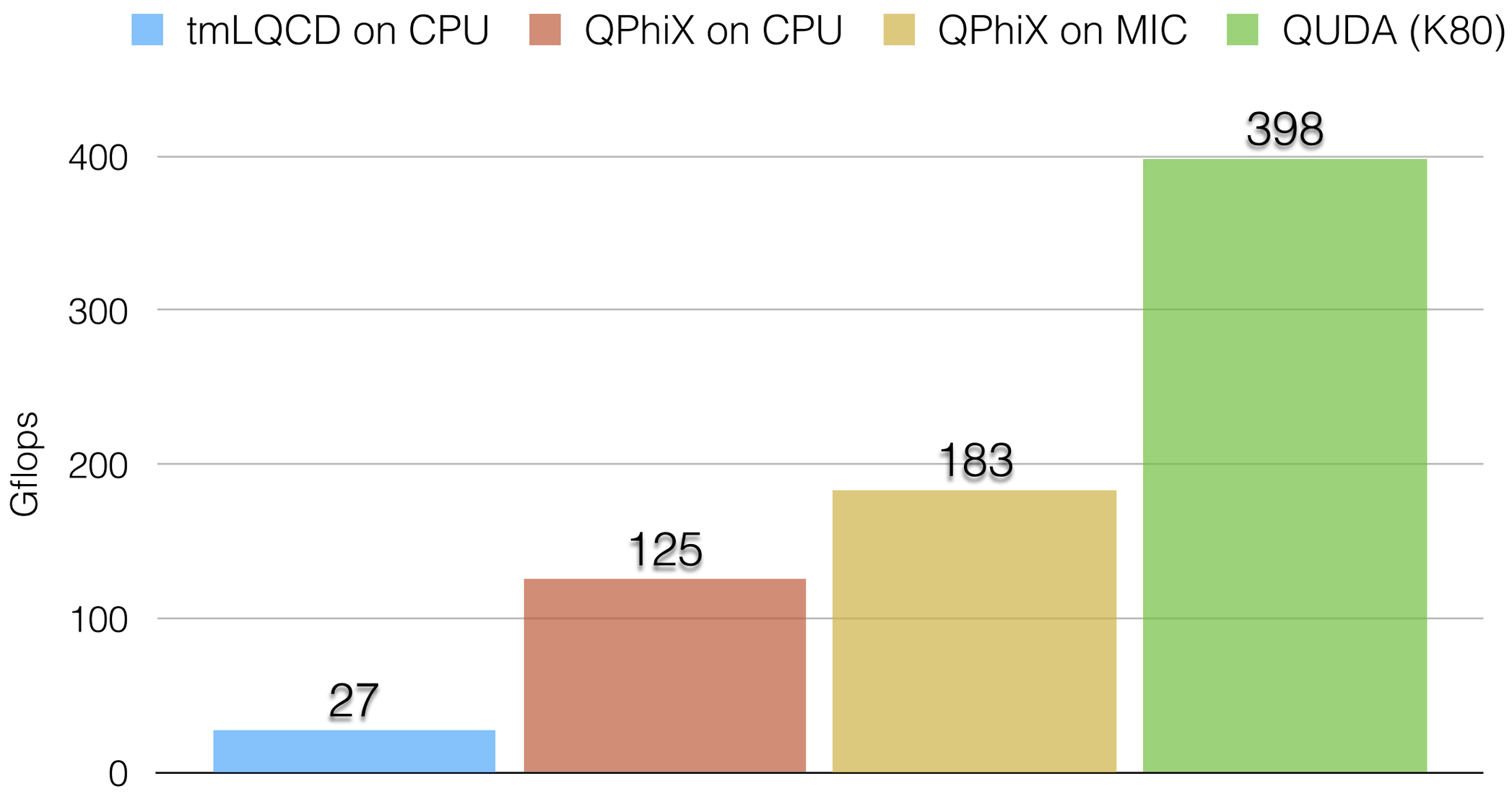
  `TMWILSON, WILSON, DBTMWILSON, CLOVER`

- supported inverters:

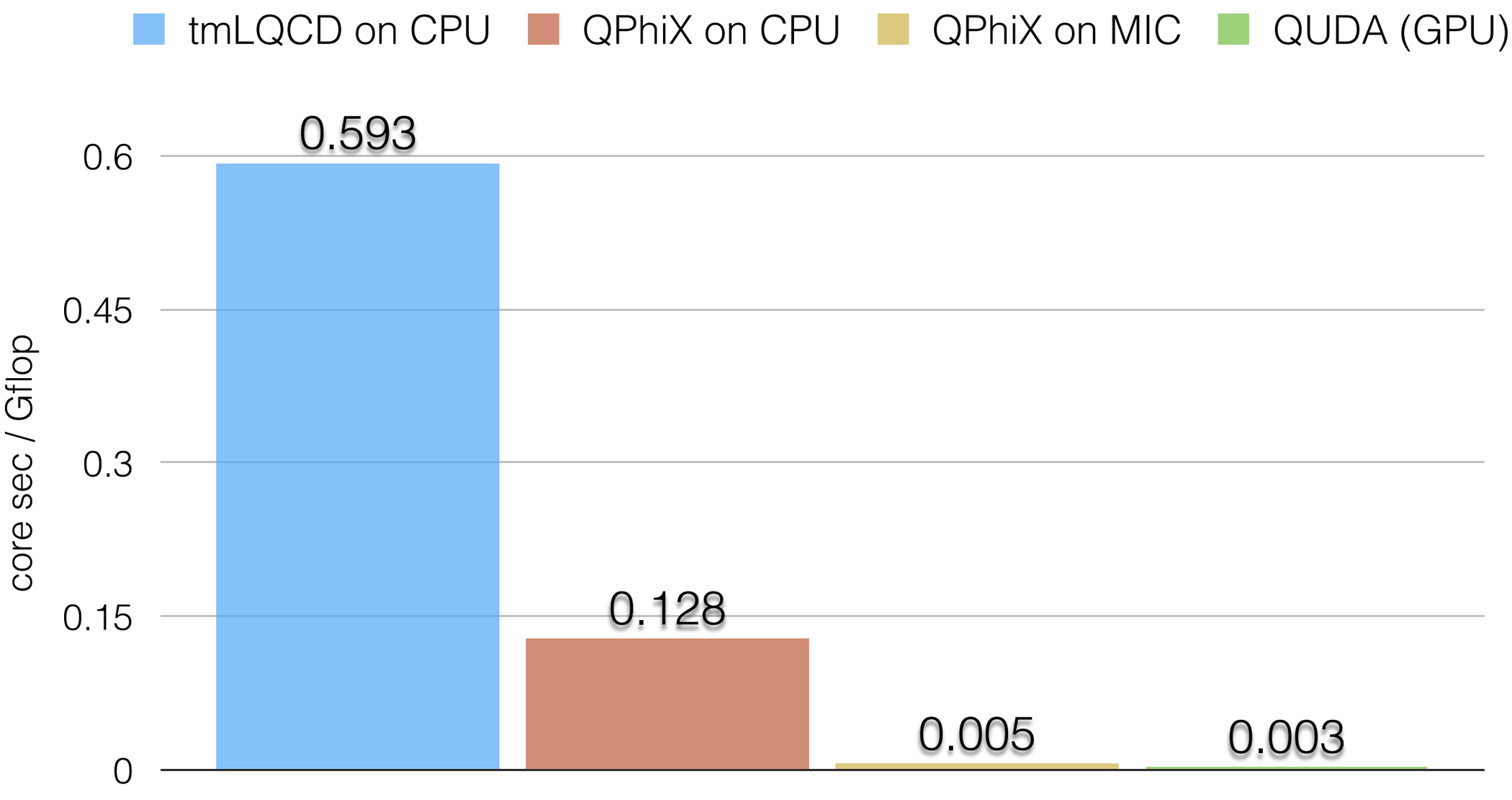
  `mixed prec. CG-eo, BICGSTAB, (eigCG, DD-precond. GCR, multigrid(?))`

Performance

Performance per CPU, GPU, MIC

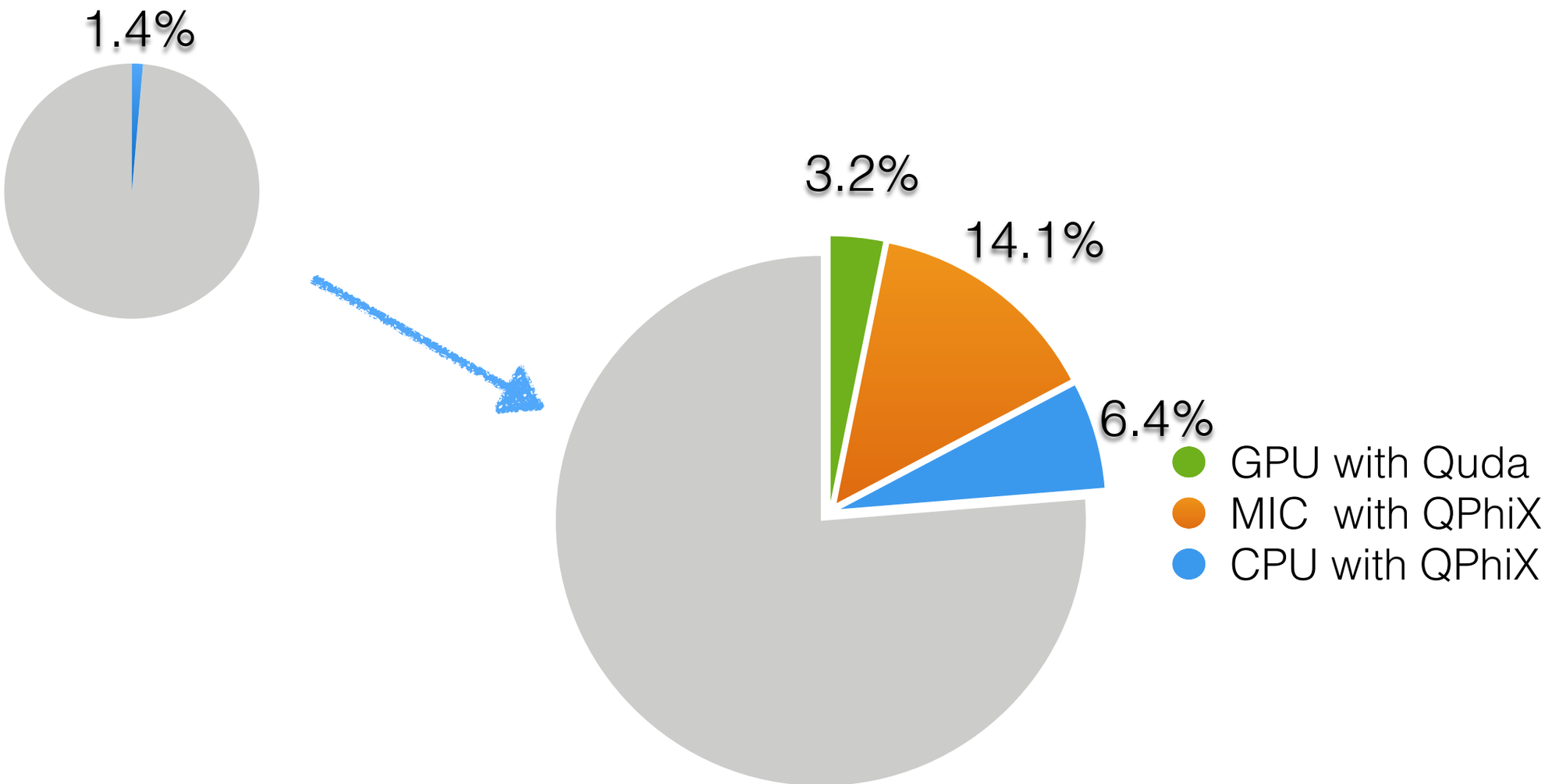


Accounting costs / flop



QPhiX on MIC (Quda on GPU) reduces accounting costs to 9‰ (4‰)!

Galileo performance with Quda/QPhiX interfaces

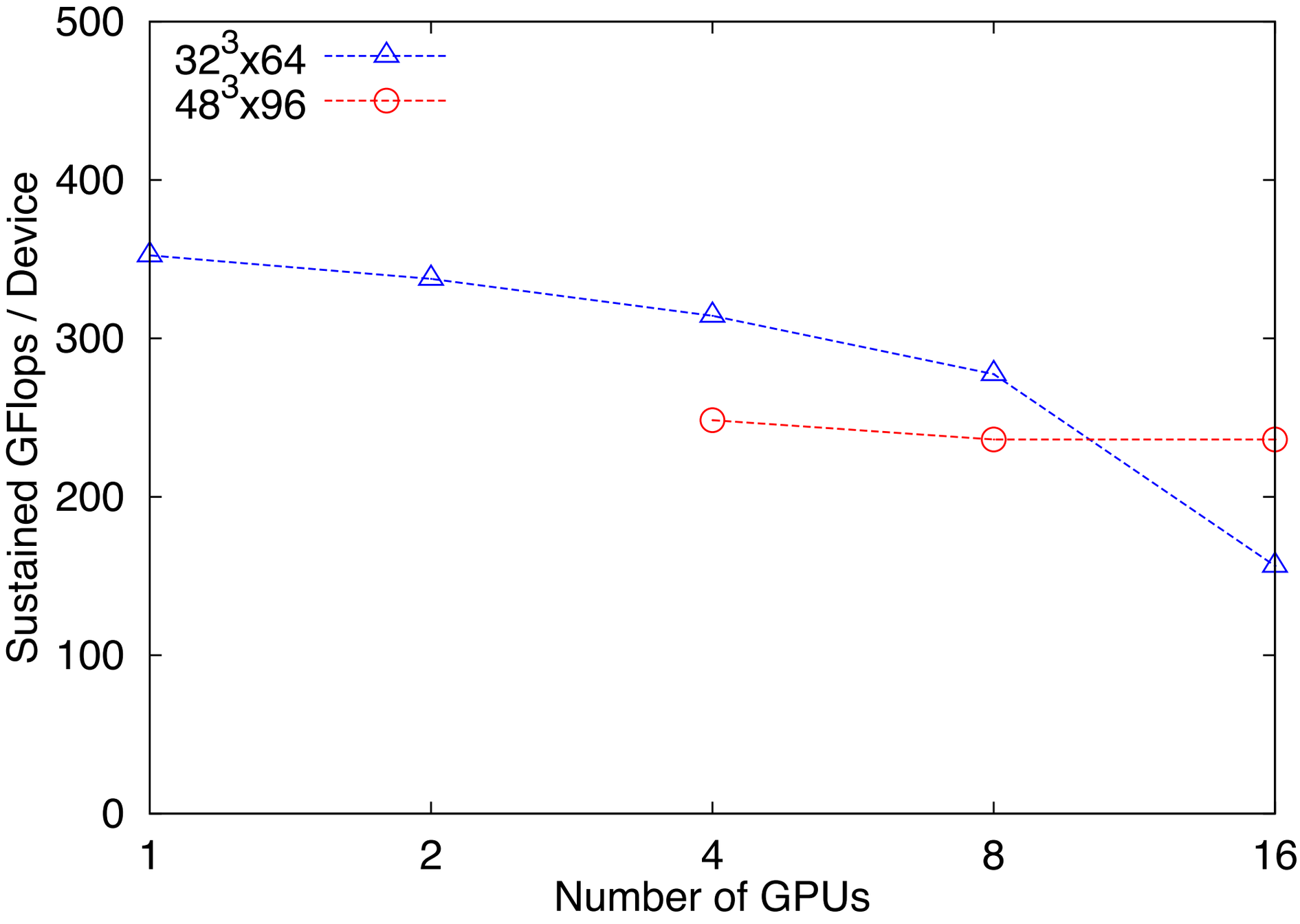


Galileo: 1Pflops

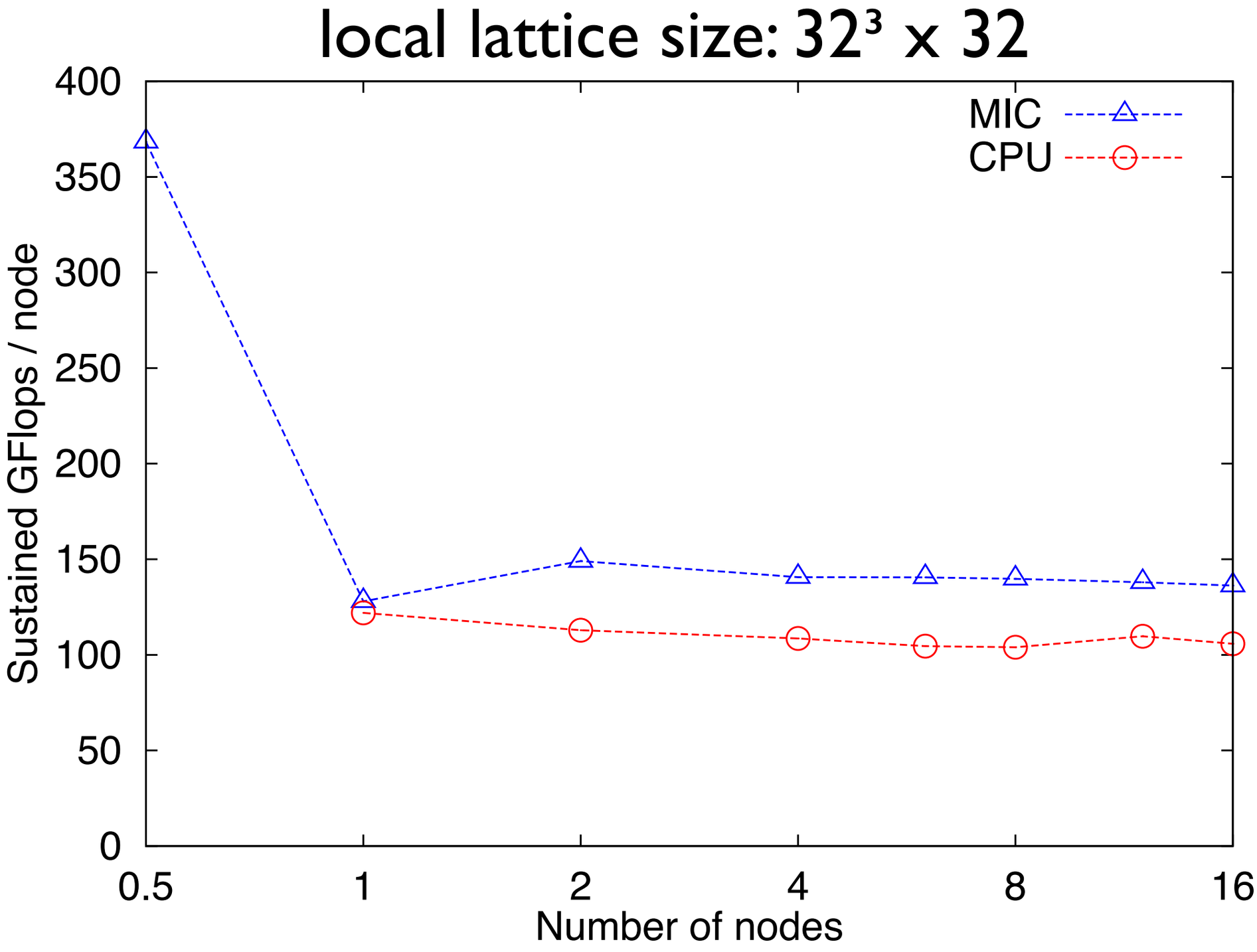
Increasing the performance by 17x while leaving the accounting costs the same!

Scaling

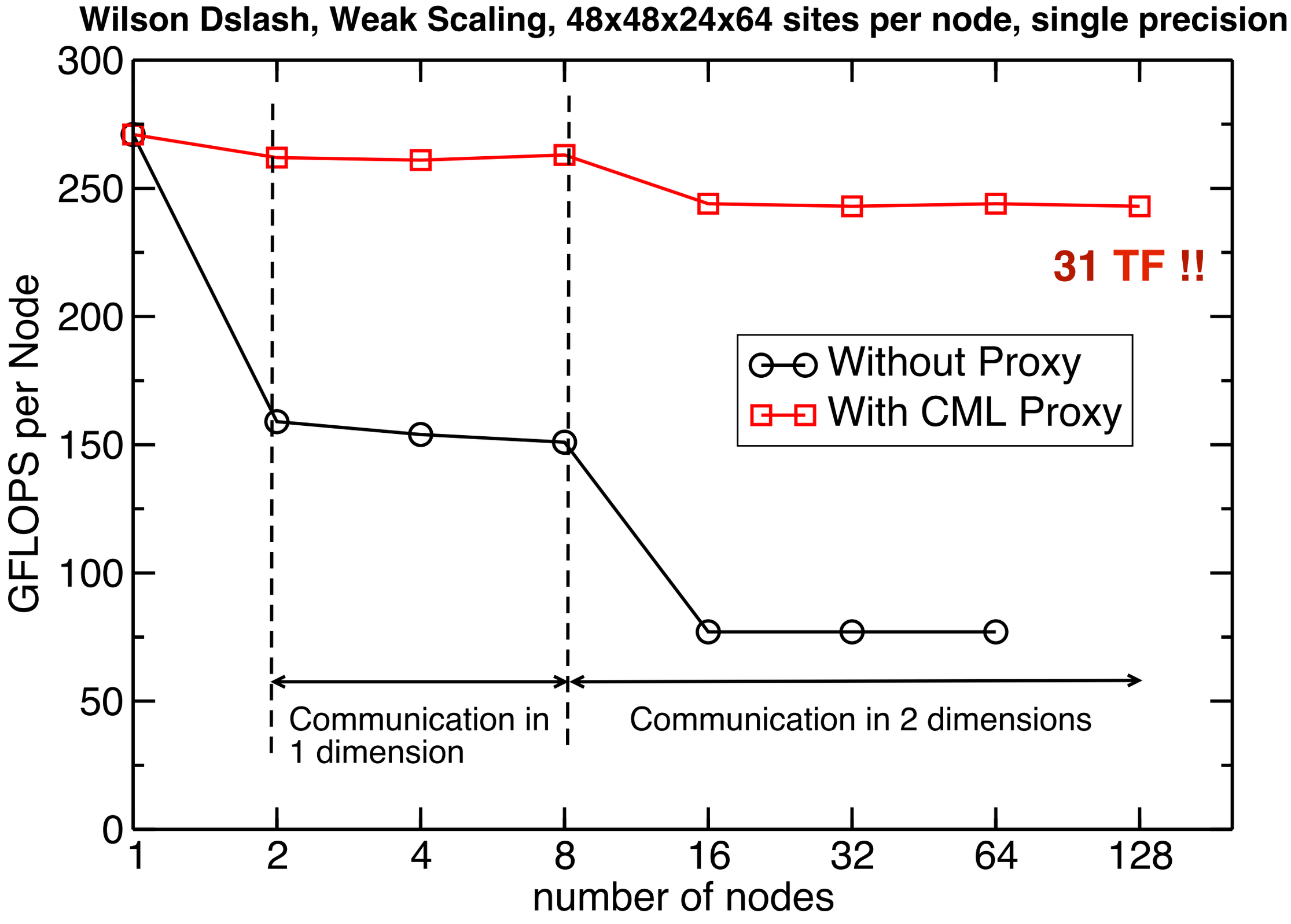
Quda strong scaling (K20)



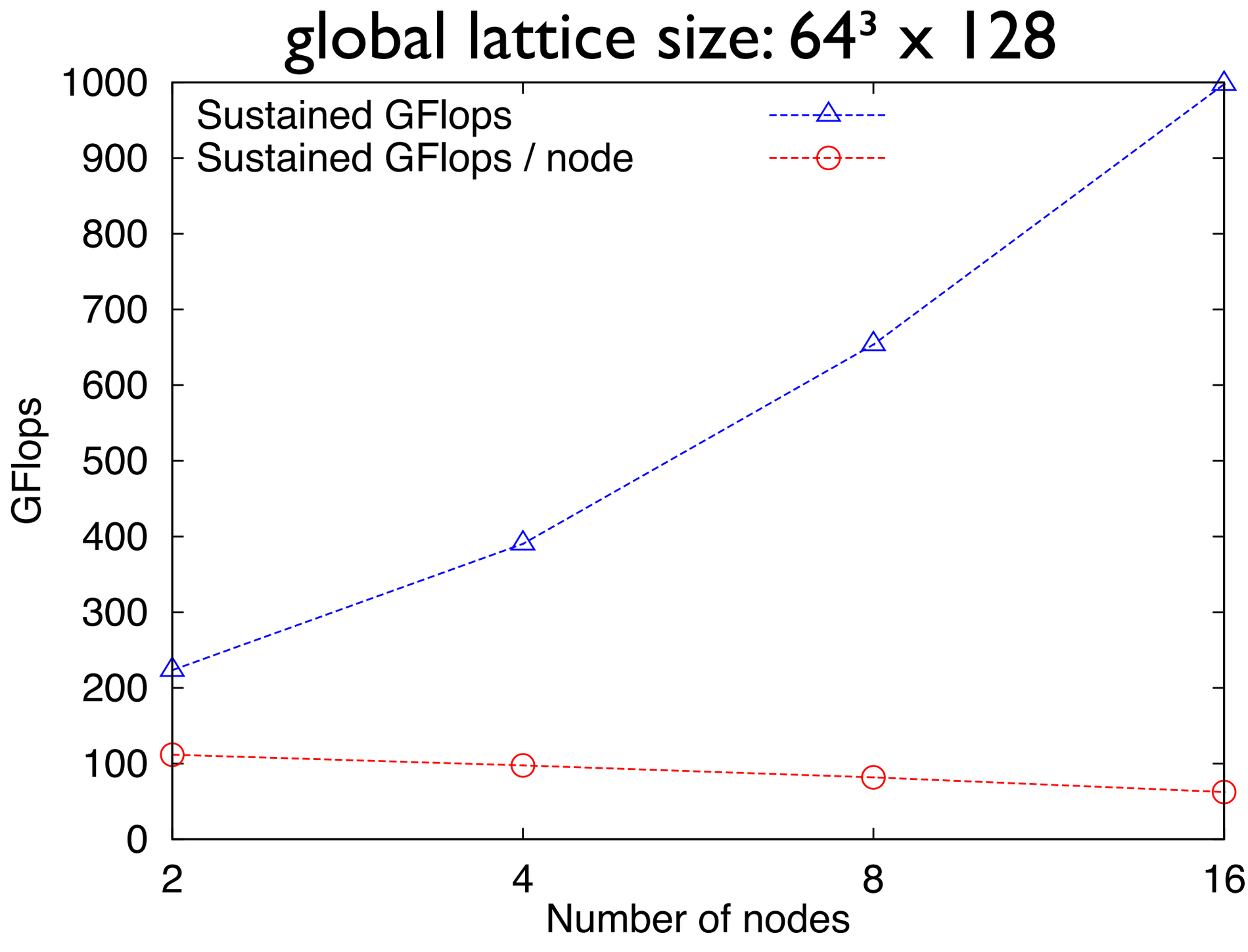
QPhiX weak scaling



© Balint Joo:


Wilson Dslash, Weak Scaling, 48x48x24x64 sites per node, single precision
GFLOPS per Node
300
250
200
150
100
50
0
1
2
4
8
16
32
64
128
number of nodes
31 TF !!
Without Proxy
With CML Proxy
Communication in
1 dimension
Communication in 2 dimensions

QPhiX strong scaling on CPU



Summary

- hybrid machines require optimized code
- difficult to maintain own tmLQCD optimized code for various architectures
- outsource development of optimized code where appropriate (inverters) and use interfaces

- with Quda and QPhiX interfaces we reach ~25% of the peak performance on Galileo
- scaling not perfect, use only as many nodes as necessary to fit the problem

Current status

- **QUDA**

☑ invert executable for Wilson, Clover, TM, TM-Clover, TM-nondeg. doublets with (mixed precision) CG-eo, BiCGStab

☐ TM-nondeg. doublet with Clover term (missing on Quda side)

☑ interface finished, pull-request opened.

- **QPhiX**

☑ invert executable for Wilson, Clover, TM with (mixed precision) CG-eo, BiCGStab

☐ TM-Clover, TM-doublets

☐ interface to be finished within the next weeks.

Thanks for your attention!

And special thanks go to

Mike Clark, Balint Joo, Alexei Strelchenko and Alejandro Vaquero

for help with Quda and QPhiX.