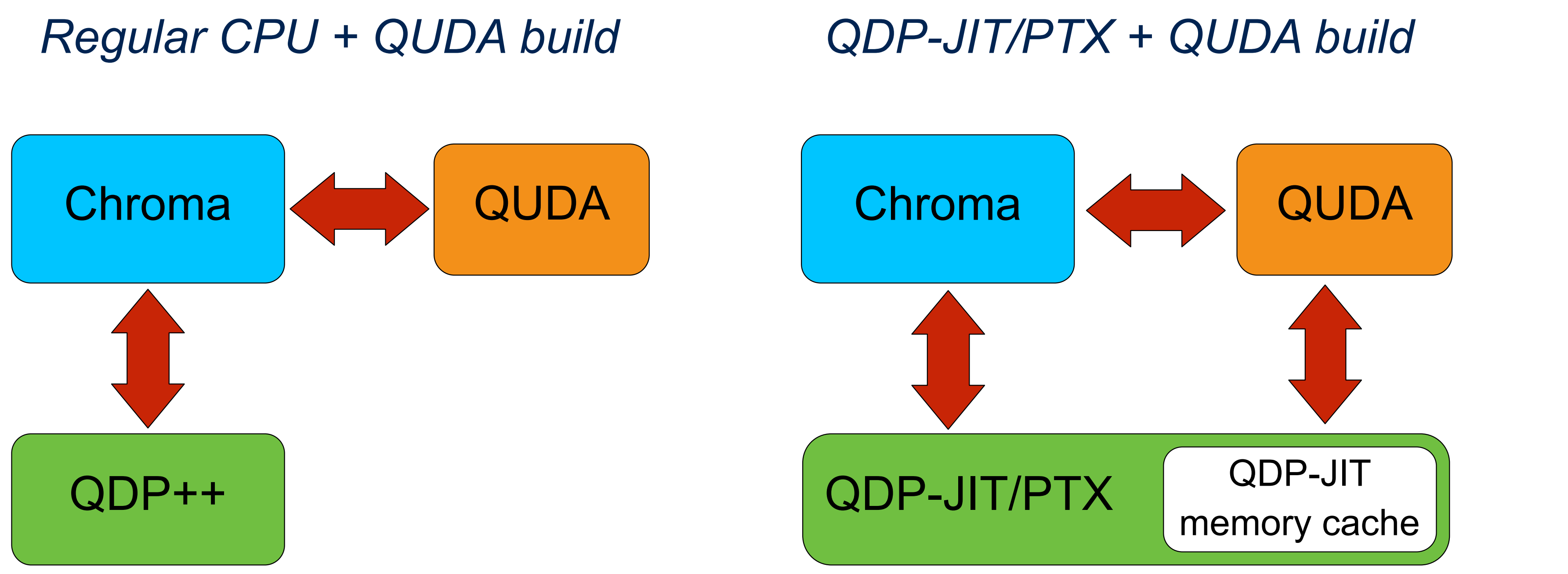
Advanced compilation techniques, architectures and algorithms for Nuclear Physics calculations

Frank T. Winter, Bálint Joó, Robert G. Edwards, **David G. Richards (presenter)**, *Jefferson Lab*
fwinter@jlab.org, bjoo@jlab.org, edwards@jlab.org, dgr@jlab.org

Introduction: QDP-JIT/PTX

The current diversity in computing architectures makes Domain Specific Language (DSL) approaches to constructing codes highly attractive. Users can write code using DS constructs, and have the DSL implementation ensure the best mapping onto the available system. The success of this approach is demonstrated by the QDP-JIT/PTX implementation of the QDP++ domain specific framework for Lattice QCD. The Chroma application, built over the QDP-JIT/PTX implementation is successfully accelerated on NVIDIA GPU based systems. This, in combination with the QUDA optimized solver library enabled Chroma on large scale GPU based systems such as Titan at OLCF and Blue Waters at NCSA.


Regular CPU + QUDA build

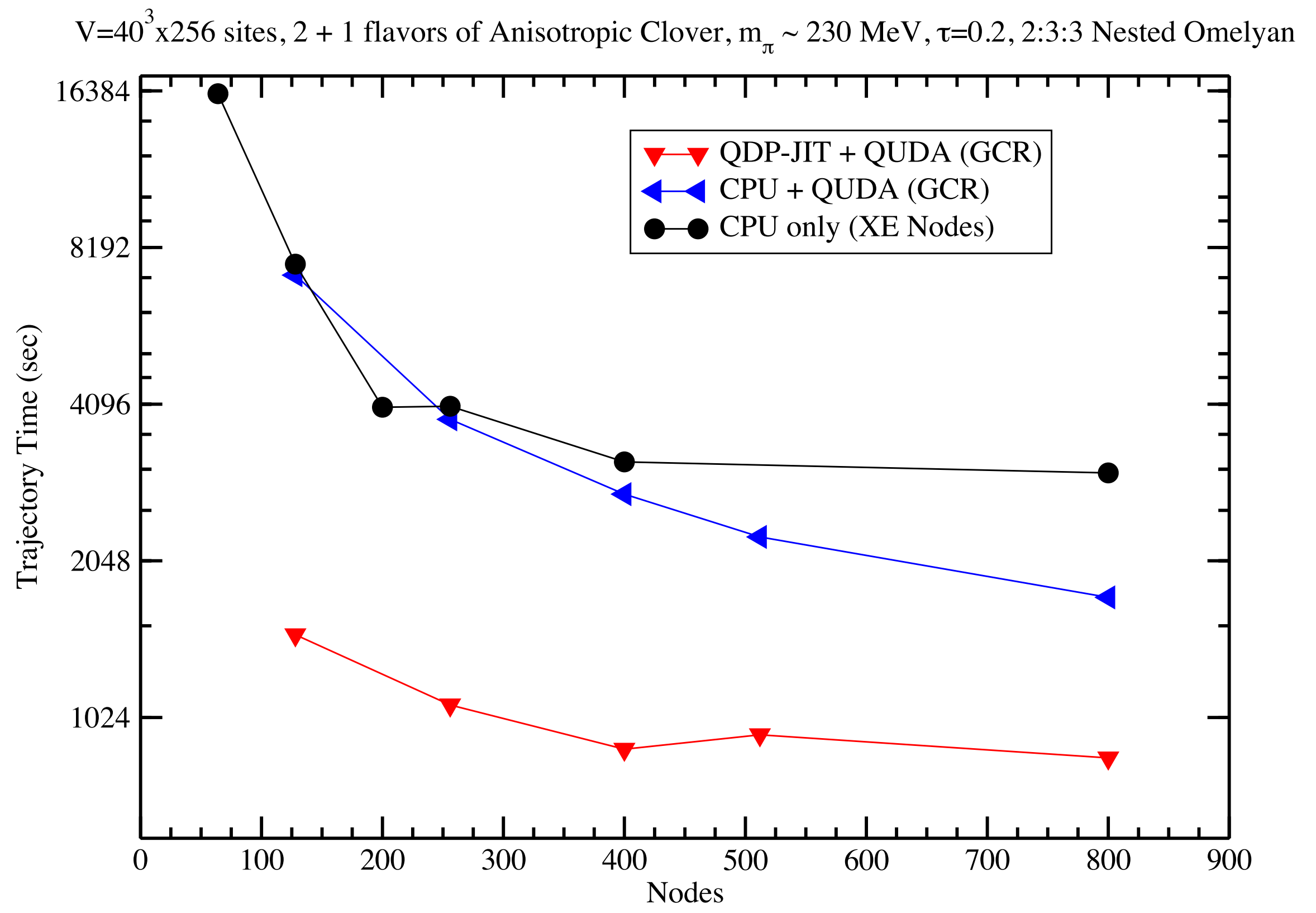
Chroma
QUDA
QDP++

QDP-JIT/PTX + QUDA build

Chroma
QUDA
QDP-JIT/PTX
QDP-JIT memory cache

In the regular CPU + QUDA build QDP++ generates code for user expressions using Expression Templates (ETs). The C++ compiler cannot immediately generate GPU code from ETs and hence non-QUDA parts of the code run on the CPU

In the QDP-JIT/PTX + QUDA build, the QDP++ expression templates generate code-generators, which generate the GPU kernels at run-time. The QDP-JIT/PTX implementation also contains a cache to manage movement of data between CPU & GPU. Changes to Chroma were minimal and only to parts where it broke out of QDP++

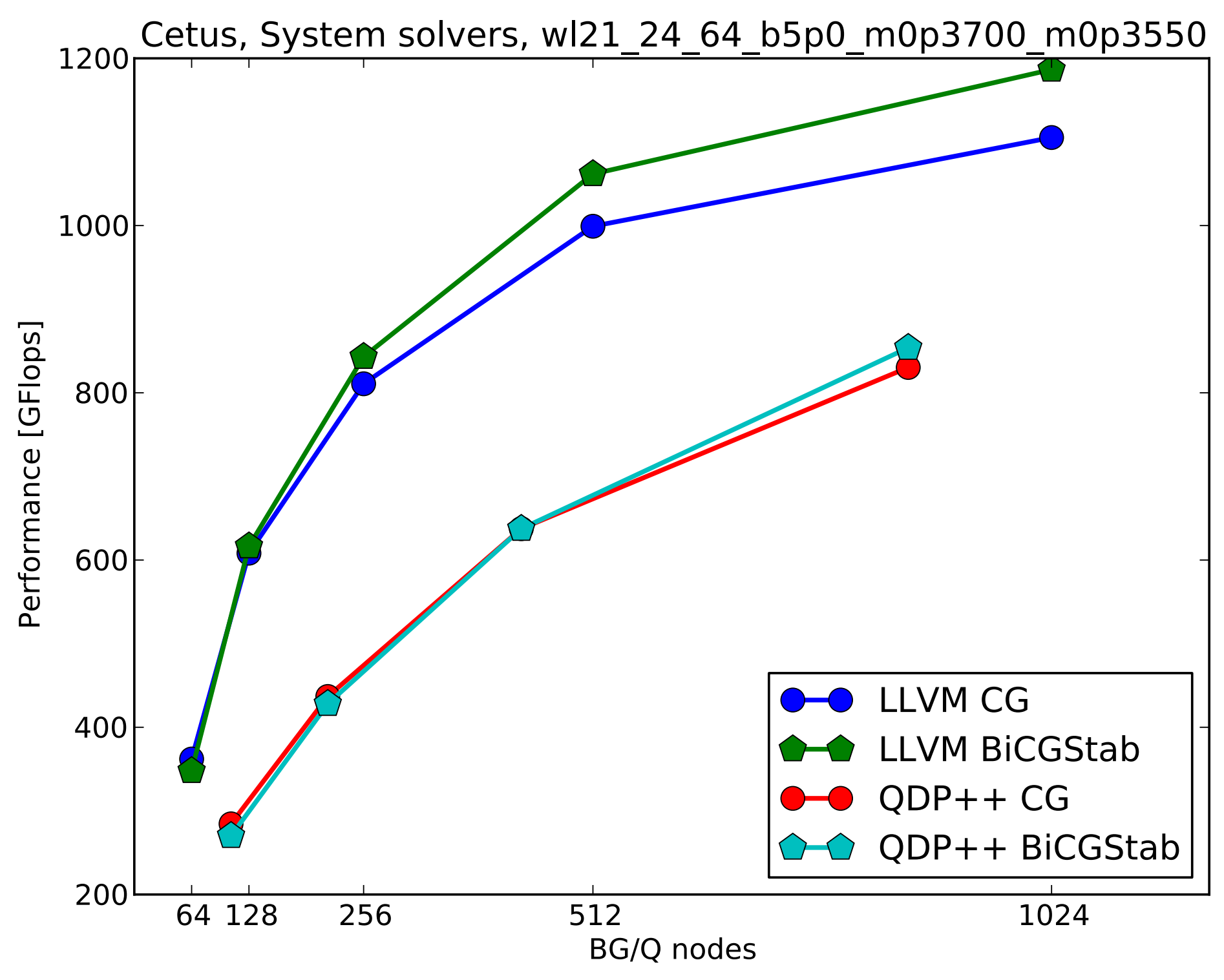


The JIT approach provides opportunities for several performance improvement, such as auto-tuning of the JIT-ed kernels and memory layout transformations when moving data between the host and GPU, e.g., to allow coalesced access on GPUs or vectorized access on the host.

Speedup of an HMC trajectory on NCSA Blue Water, using data from F. Winter et. al. IPDPS'14. Here we compare performance on GPU enabled XK nodes (1 NVIDIA K20X GPU + 1 AMD Interlagos CPU), with Dual Socket XE nodes (2 AMD Interlagos Sockets). Chroma + QDP-JIT/PTX is the clear winner

The Future: QDP-JIT/LLVM

Rearchitecting QDP-JIT, to generate Internal Representation (IR) for the LLVM compiler framework allows the JIT approach to be harnessed also for non-GPU targets. Generating LLVM IR has several advantages: efficient code generation for several targets (x86, PowerPC, GPU), as well as advanced code transformations and optimizations (fusion, fission, tiling, etc.)

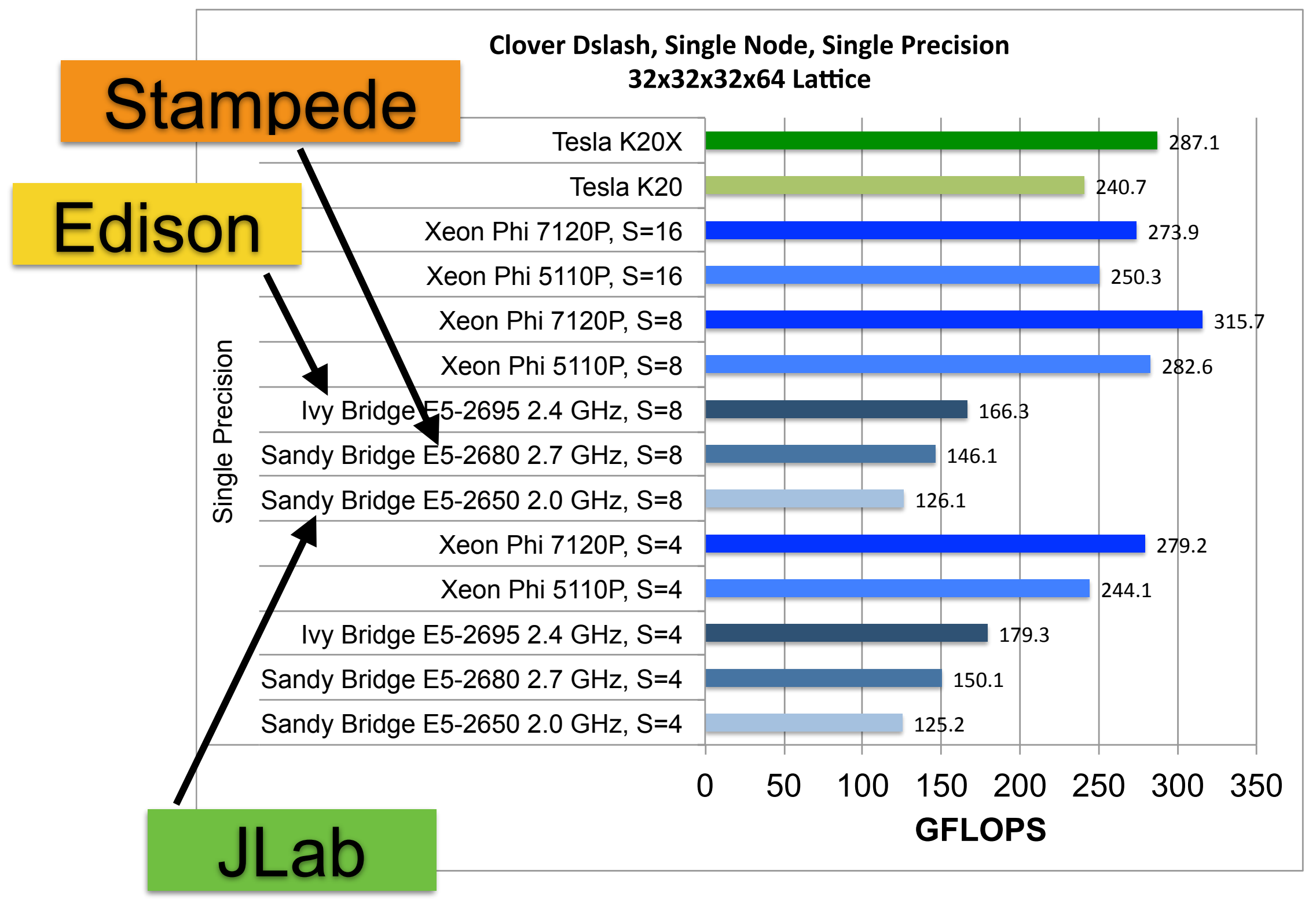


Performance of 'Native Chroma' solvers on BlueGene/Q based on Chroma built over QDP++ and QDP-JIT/LLVM respectively. This result is preliminary, as the LLVM approach does not yet provide for vectorization. Nonetheless even in this preliminary result, the approach outperforms the regular expression template based approach.

QDP-JIT/LLVM shows promise on BlueGene/Q and we plan to use it as the primary approach for efficient implementation of QDP++ on the recently announced Xeon Phi Knights Landing architecture, e.g. on the NERSC-8 Cori system.

Xeon Phi and x86 Optimization

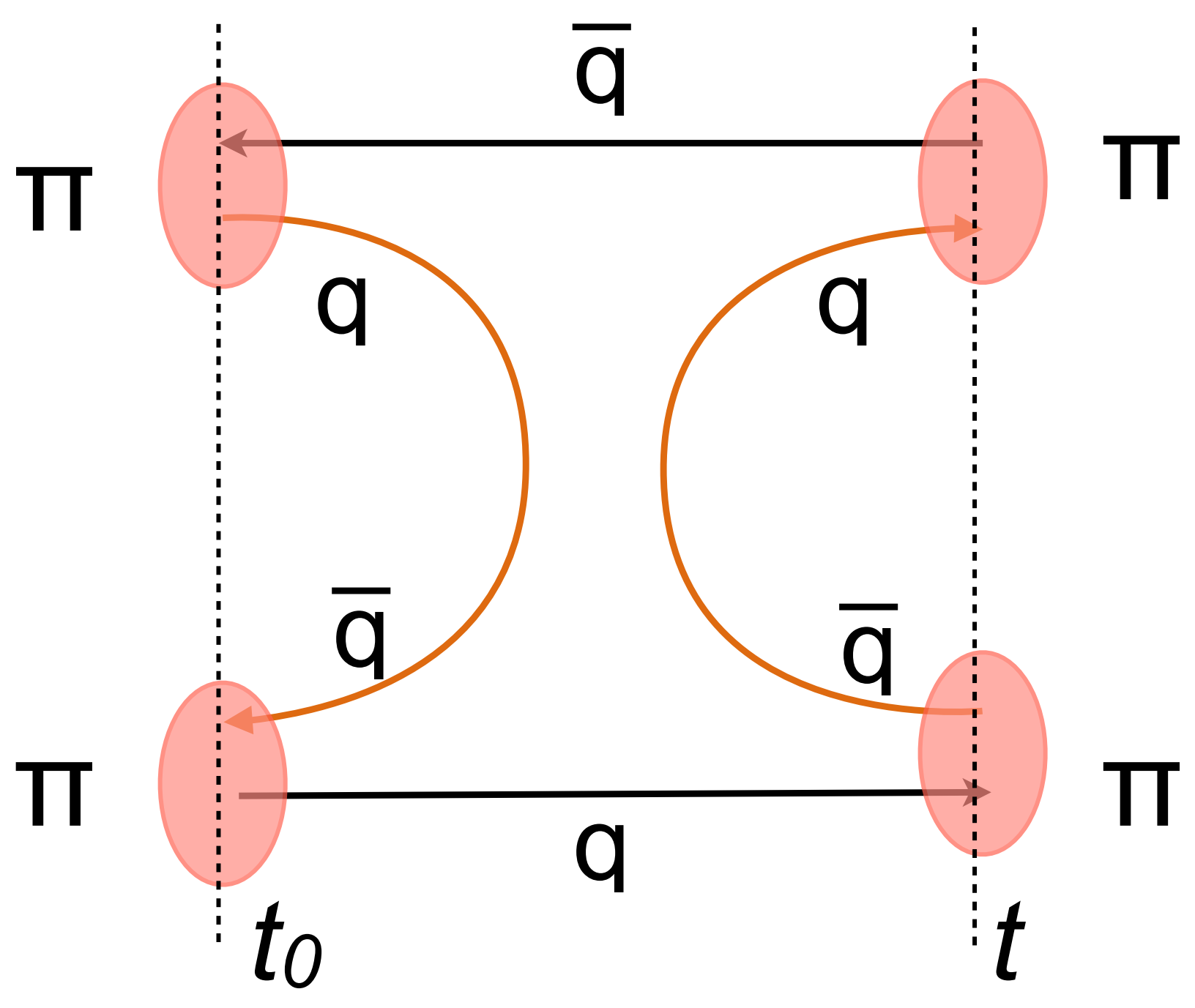
While the JIT approach is successful, there is always room for highly optimized, hand tuned solvers. We have continued our work on optimized solvers for Xeon Phi and Xeon (Joo et. al. ISC'13). Our latest code features double precision and also 16 bit up-downconversion on Xeon Phi. Communication can now be performed in all 4-dimensions.



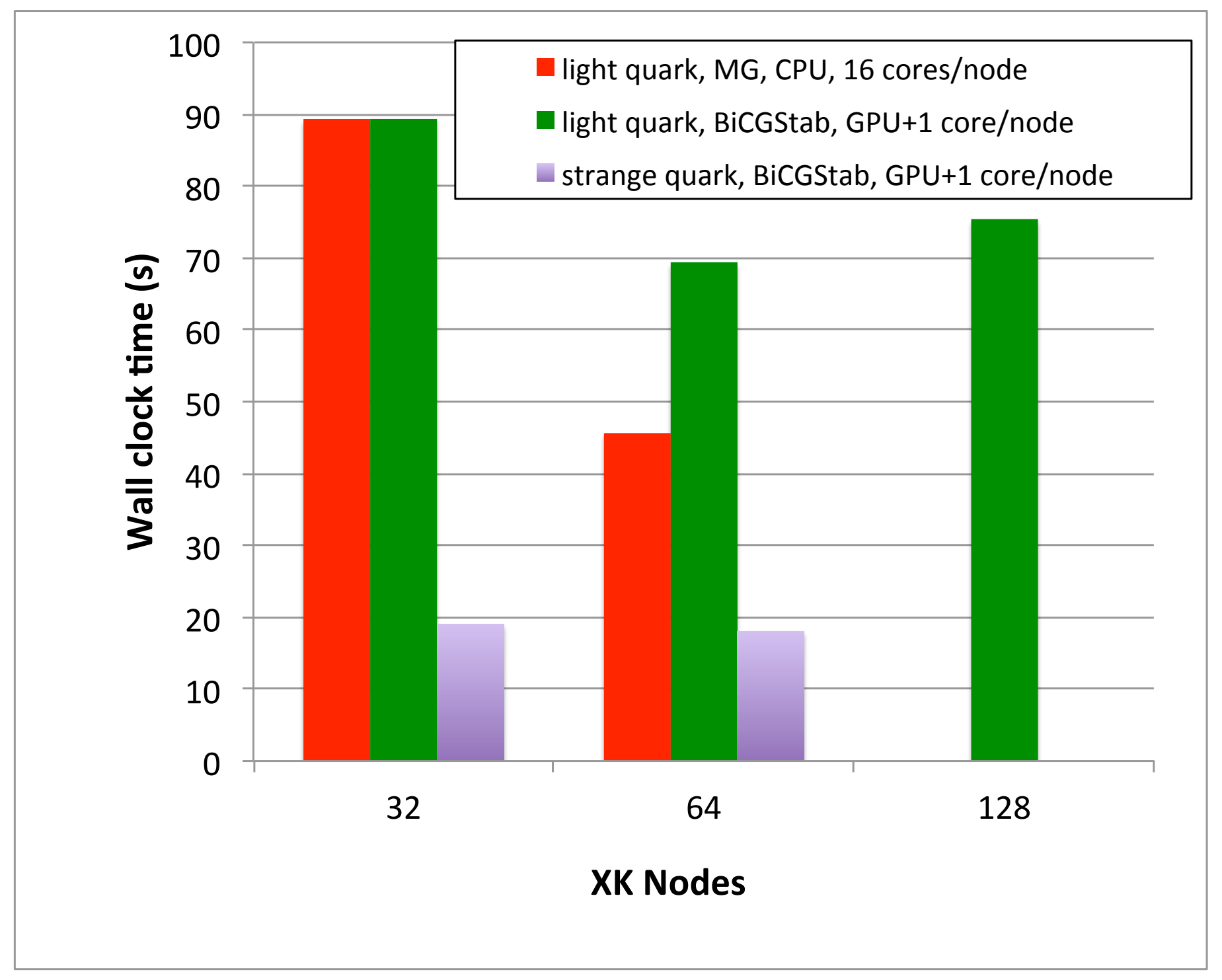
Performance of our Clover-Dslash operator on a Xeon Phi Knight's Corner and other Xeon CPUs as well as NVIDIA Tesla GPUs in single precision using 2-row compression. Xeon Phi is competitive with GPUs. The performance gap between a dual socket Intel Xeon E5-2695 (Ivy Bridge) and the NVIDIA Tesla K20X is only a factor of 1.6x.

Multigrid Solvers

Correlation functions for Hadron Spectroscopy calculations are computed from quark line diagrams. Each quark line is a quark propagator and is the result of solving the QCD Dirac Equation. Each configuration requires O(1M) solutions of the Dirac Equation. This motivates the use of optimized solver algorithms, such as recently developed Algebraic Multi-Grid Solvers



A quark line diagram used in computing 2π → 2π transition, potentially via a resonance. The orange propagators start and end on the same time-slice and hence need to be computed for all (e.g. 256) time-slices. With 368 sources, 4 spins and 2 quark masses (light & strange) one needs 753K solves per configuration for the orange propagators alone.



*The Dirac equation is solved on the GPUs using the QUDA solver library integrated with the **Chroma** code. The Algebraic Multi-Grid (AMG) solver for CPUs from the QDPQOP library (Osborn) has also been integrated (S. D. Cohen & B. Joo) with Chroma. As can be seen AMG matches and outscales the performance of BiCGStab from QUDA for light quarks on the NCSA Blue Waters system. A GPU version of the AMG solver is in development and we will soon embark on an implementation for Xeon Phi*

Conclusion

We have shown progress both on applying advanced compilation techniques to our code and exploiting advanced algorithms and architectures. These efforts enable our calculations on current systems and are paving the way for the next generation of extreme scale platforms