MILC Code Performance on High End CPU and GPU Supercomputer Clusters

Ruizi Li, Carleton DeTar,
Steven Gottlieb, Doug Toussaint

Acknowledgements

- Intel
  - K. Rama, A. Jha, D. Kalamkar, M. Tolubaeva, T. Phung
- NERSC/NESAP
  - D. Doerfler
- JLab
  - B. Joó
- This work is supported by Intel® PCC at Indiana University

Outline

- Development of the MILC code
- Libraries: QOPQDP, Staggered QPhiX, QUDA
- Benchmarks of major LQCD routines
  - Staggered multi-mass CG
  - Symanizik 1-loop gauge force
  - HISQ fermion force
- Conclusion

Development of the MILC code

- Originally single level of parallelism with message passing, e.g., MPI
  - MPP machines, Symmetric Multiprocessing (SMP) machines
  - No threading
- OpenMP as a "second" level of parallelism
  - SMP machines
  - Threads may help
- QUDA
  - GPUs
- SIMD instructions add another layer
  - SSE, AVX2, AVX512, …
  - MIC architecture, vector processing units

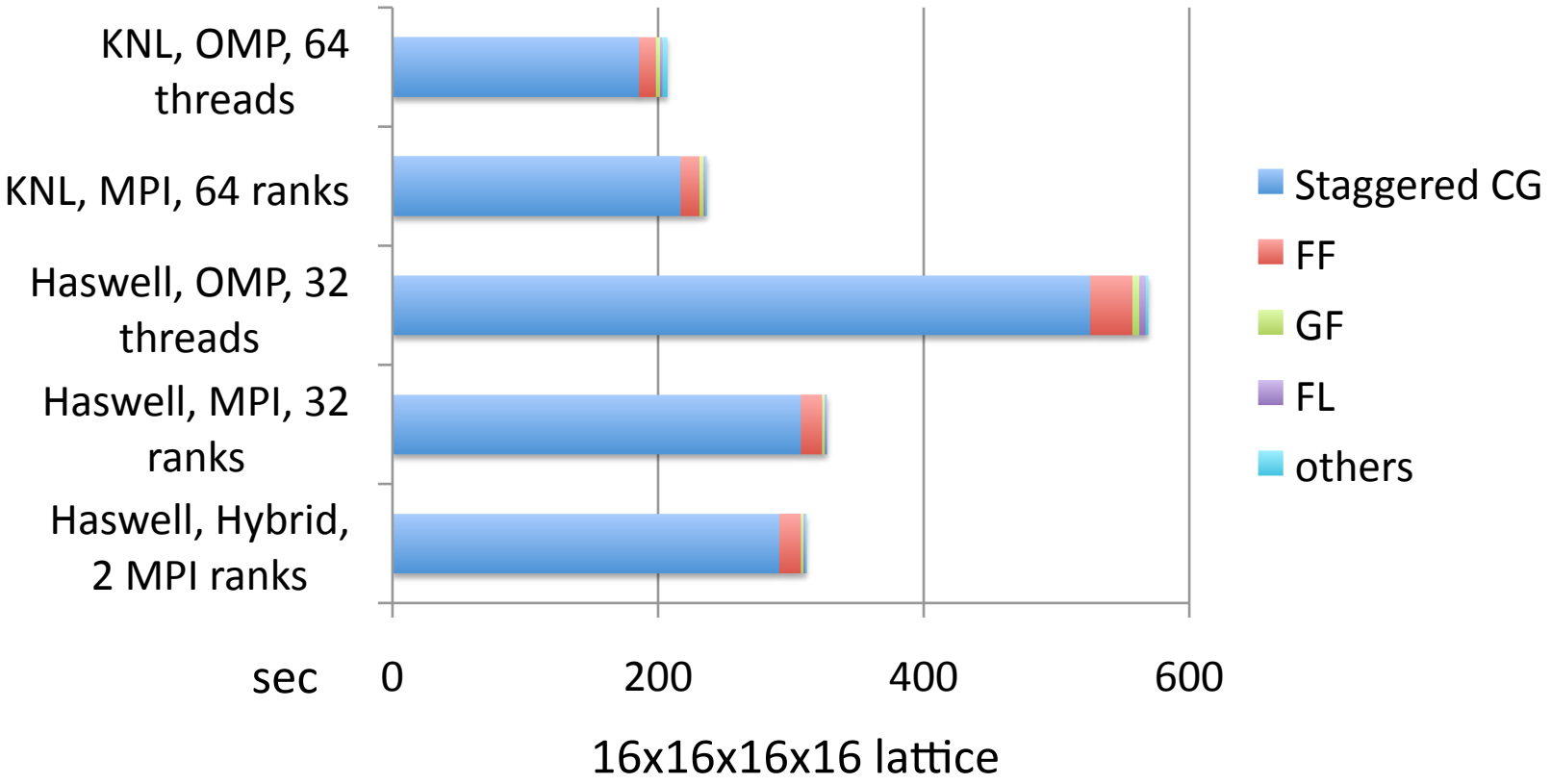
Target architectures

- CPU
  - Intel Xeon: Haswell
    - NERSC Cori cluster
  - Intel Xeon Phi: Knights Landing
    - NERSC Cori2, KNL 7250, Cray Dragonfly
    - TACC Stampede, KNL 7250, Intel Omnipath
    - ALCF Theta, KNL 7210, Cray Dragonfly
- GPU
  - Nvidia Tesla
    - Indiana University, Big Red 2, K20
    - OLCF Summitdev, P100

Libraries

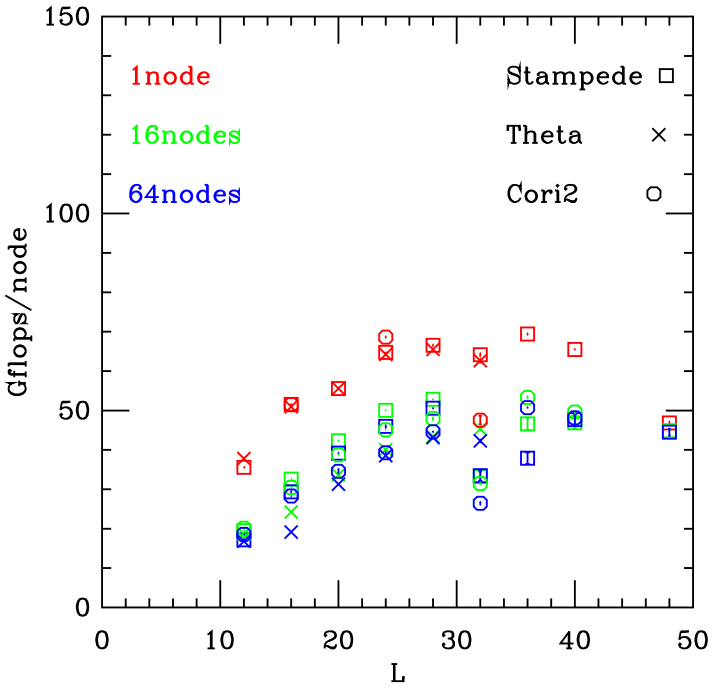
- QOPQDP
  - The SciDAC (Scientific Discovery through Advanced Computing) software package, various CPU architectures
  - Build on KNL, SSE2 intrinsic (currently no direct AVX support), -xMIC-AVX512
- Staggered QPhiX
  - Main target Intel Xeon Phi products: KNC, KNL
  - Make full use of VPUs: explicit implementation of AVX512 intrinsic instructions
  - Improve cache reuse: array-of-structure-of-array (AOSOA) replaces AOS data structure
  - Optimize OpenMP parallelization: pencil blocks of data for work threads
- QUDA
  - Version 0.8.0

- Benchmarks 2+1+1 *su3_rhmd_hisq,* <u>all in double precision</u>
- MILC code run time breakdown, one node

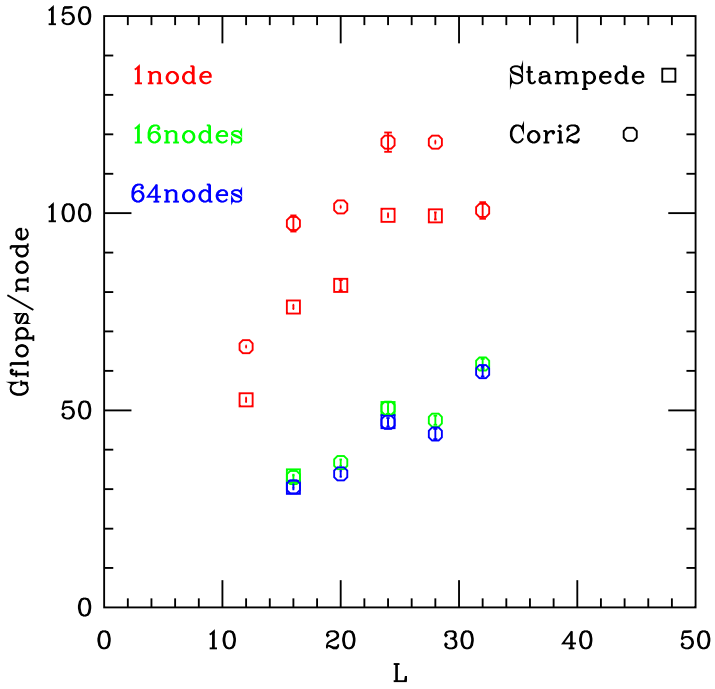


Benchmarks: Staggered Multi-mass CG

KNL clusters



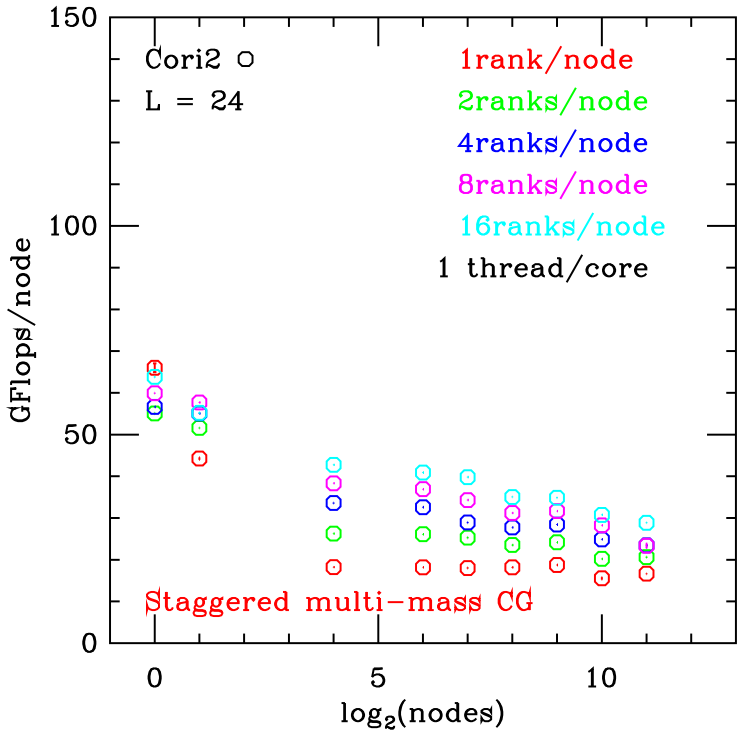
Left: MILC baseline,
64 MPI ranks per node



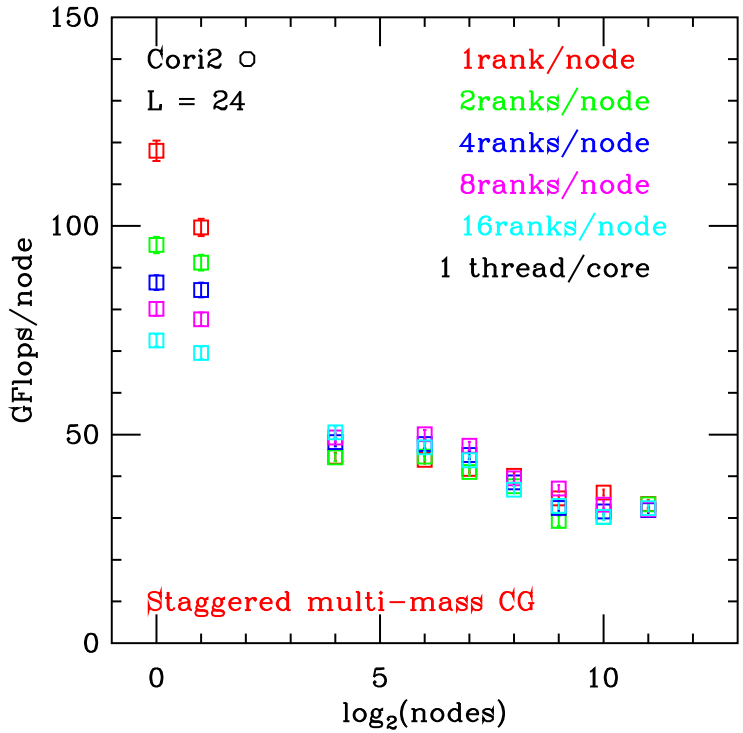
Right: QPhiX, 1 MPI rank on 1 node,
up to 16 ranks on multiple nodes

CG performance in GFLOPS/node, various lattice sizes per node L

More shown, weak scaling on up to 2048 nodes, Cori2



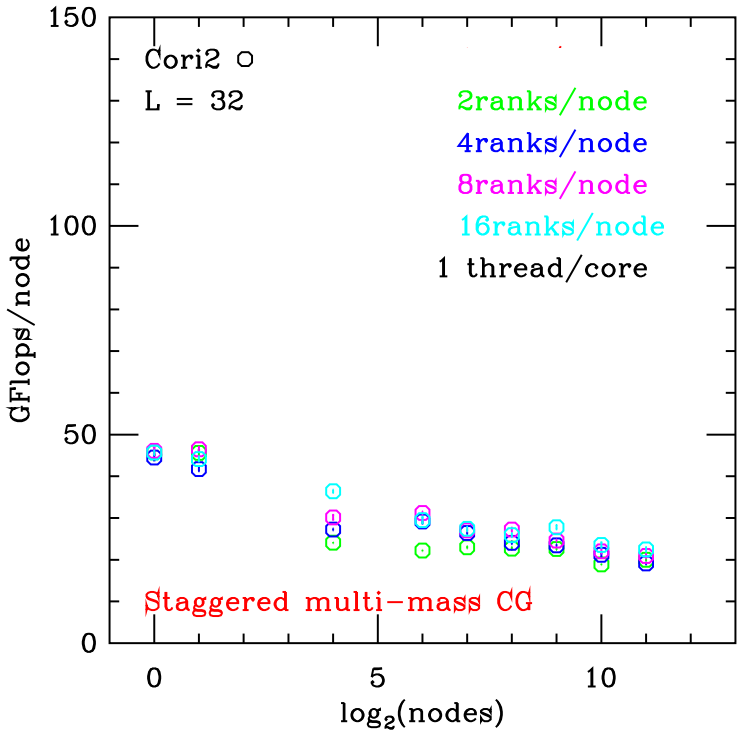
Left: MILC baseline



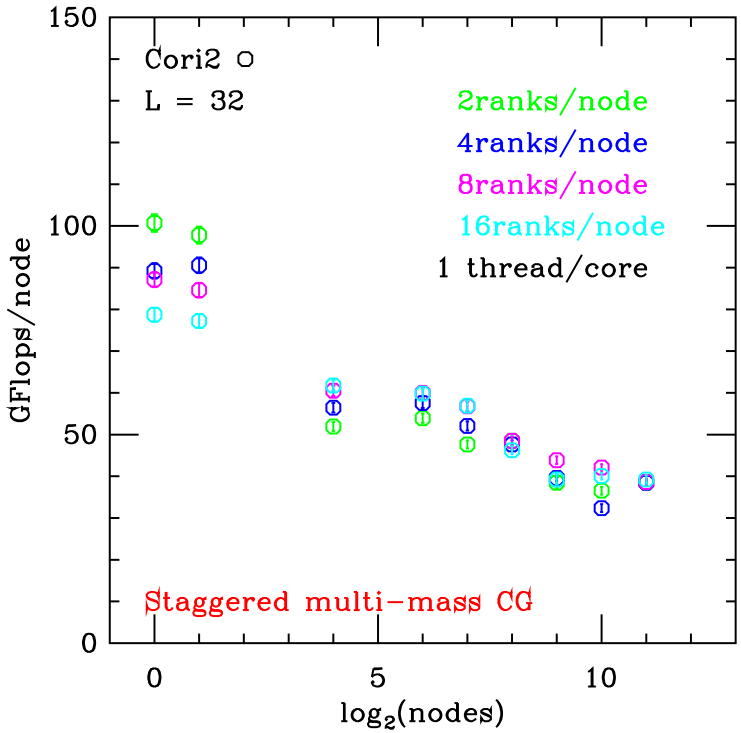
Right: QPhiX

Various MPI rank / OMP thread combinations, L = 24

… continue with L = 32



Left: MILC baseline



Right: QPhiX

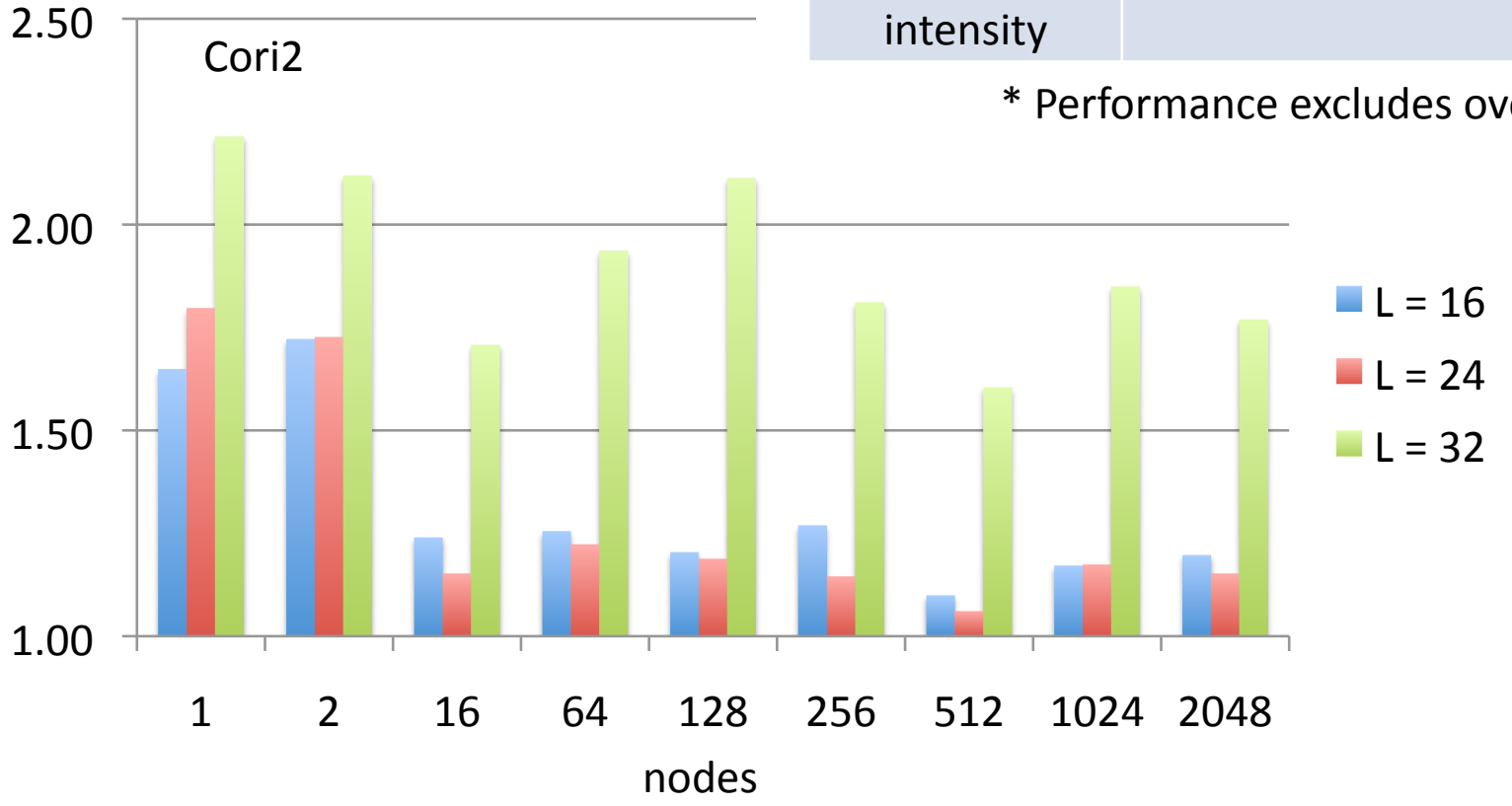
Similar performance with hyper-threading (2 threads/core) on many nodes

Right: Performance chart on Cori2, comparing baseline MILC (2nd column) and QPhiX (3rd column) CG

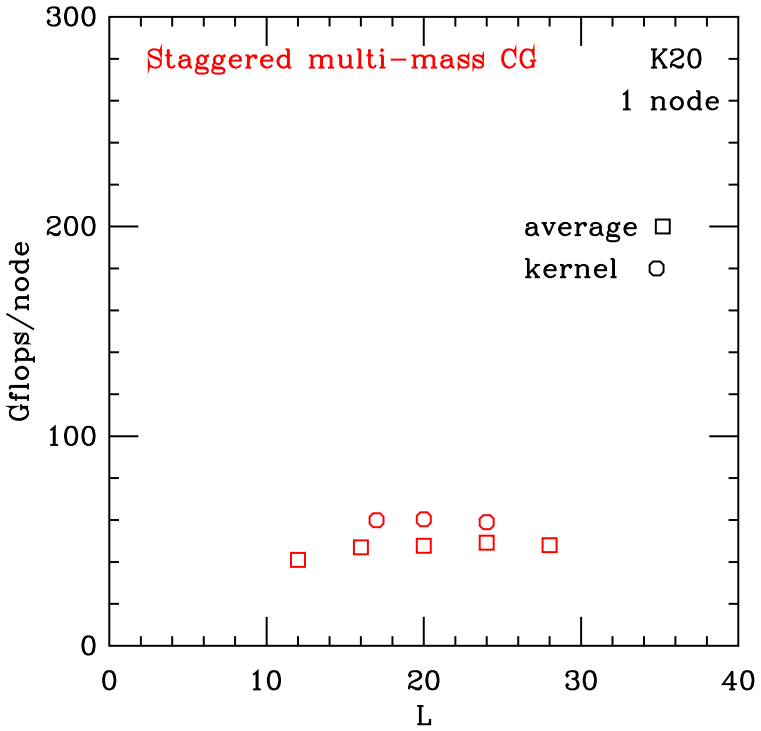
Down: Speedup with QPhiX at 1000 CG iterations, including data reconstruction

\	MILC b.	w. QPhiX
Gflops, 1node	70	120 *
Scaling factor (16 ≤ L ≤ 32)	0.3 ~ 0.55	0.25 ~ 0.45
Arithmetic intensity	0.26, multi-mass	

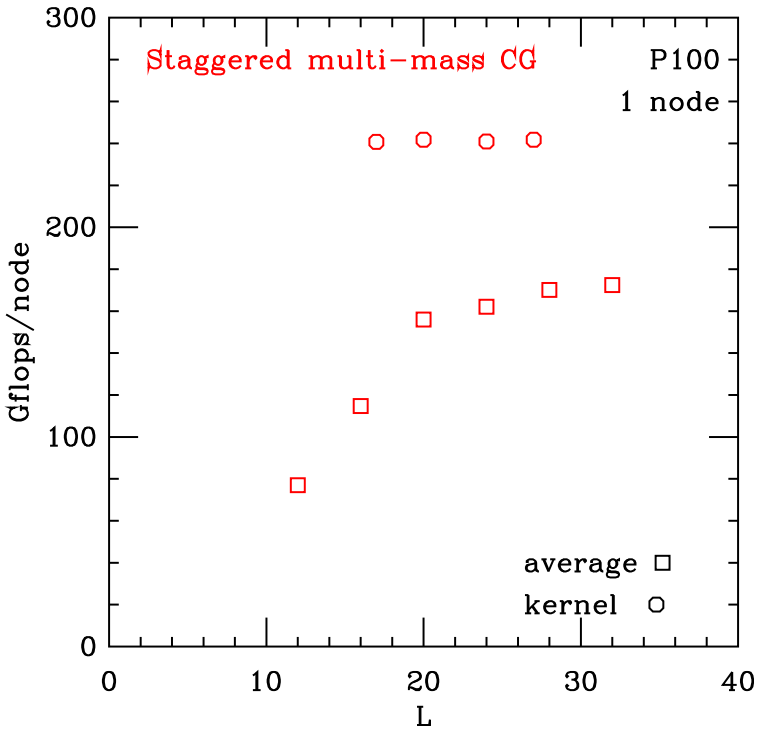
* Performance excludes overhead



GPU one node, QUDA



Left: K20, L up to 28



Right: P100, L up to 32

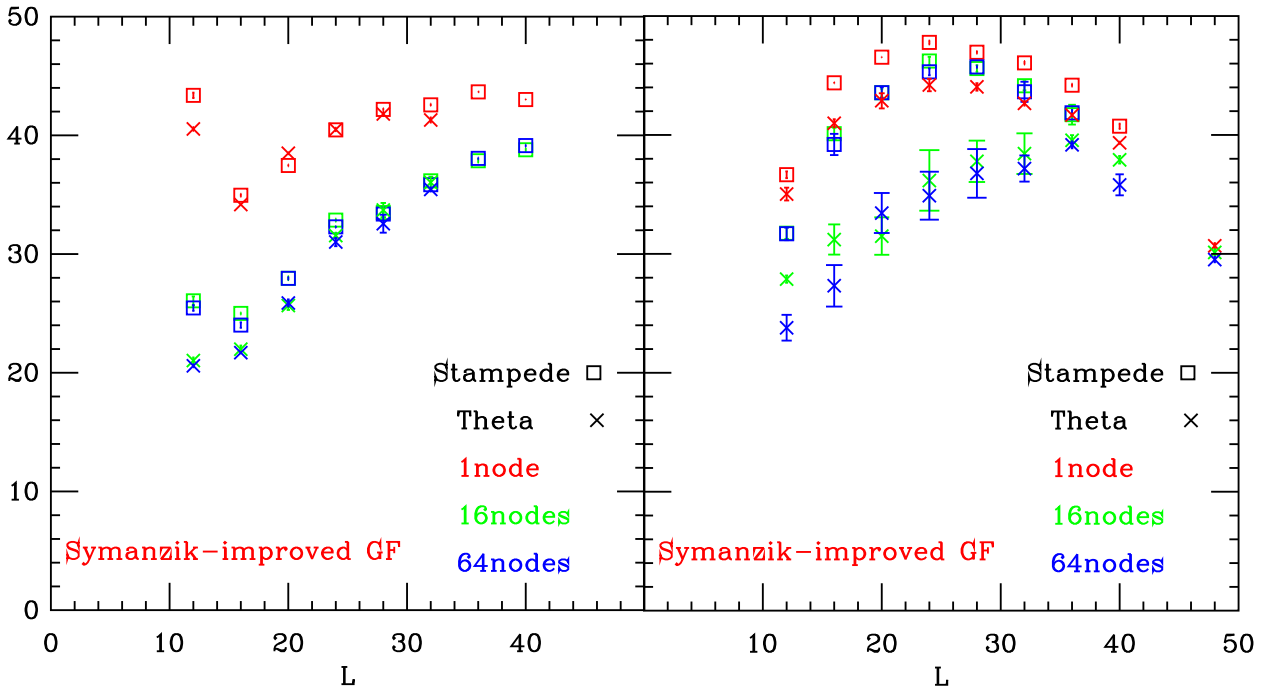
Kernel: staggered dslash

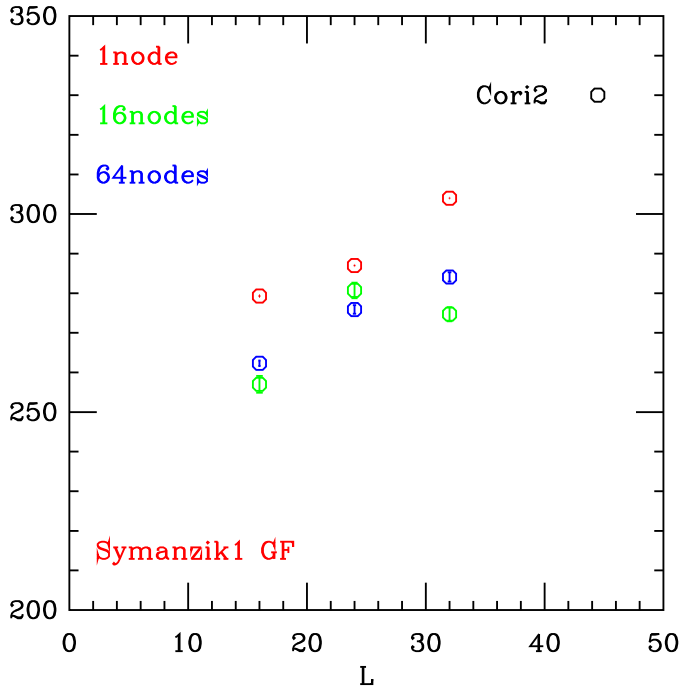
Performance excludes overhead, preliminary

Benchmarks: Symanzik Gauge Force

KNL clusters

1 thread/core





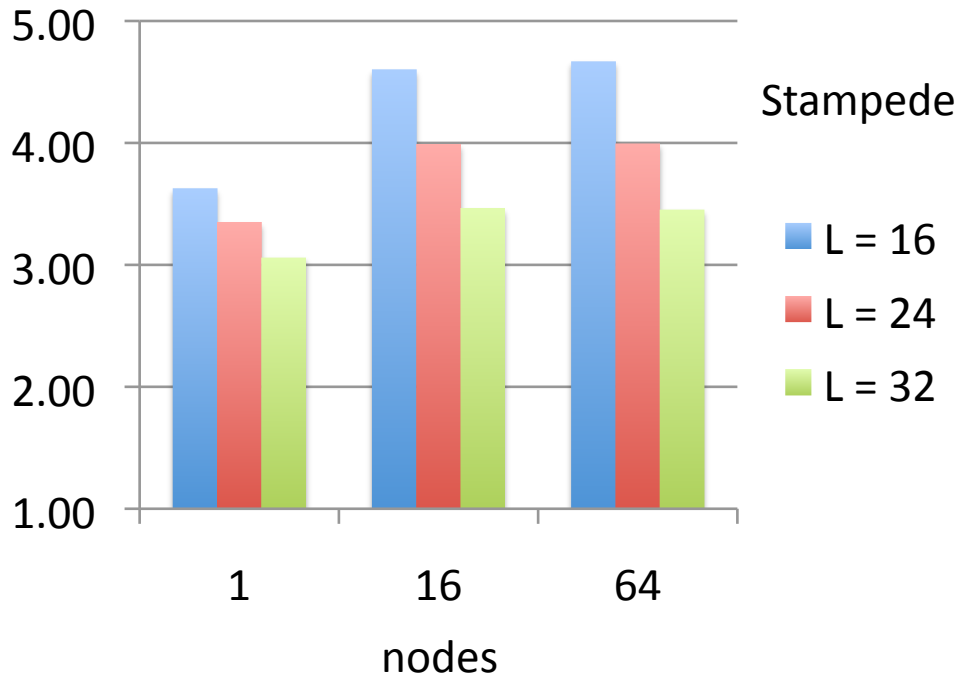
Left: MILC baseline,
64 MPI ranks per node

Middle: QOPQDP,
64 MPI ranks per node

Right: QPhiX,
1 MPI rank with 1 node,
16 ranks/node with more nodes

GF performance in GFLOPS/node, various lattice sizes per node L

Speedup with QOPQDP

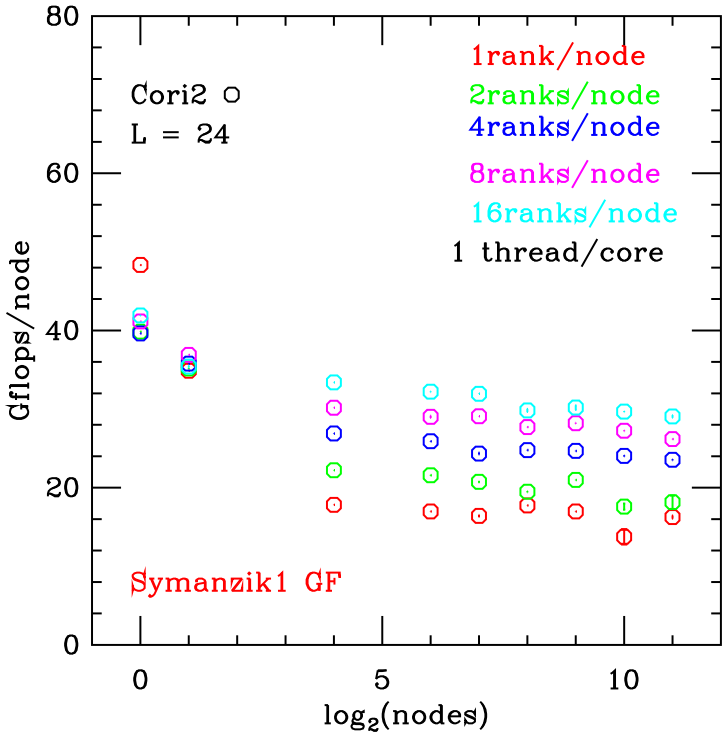


Modified algorithm vs. MILC GF

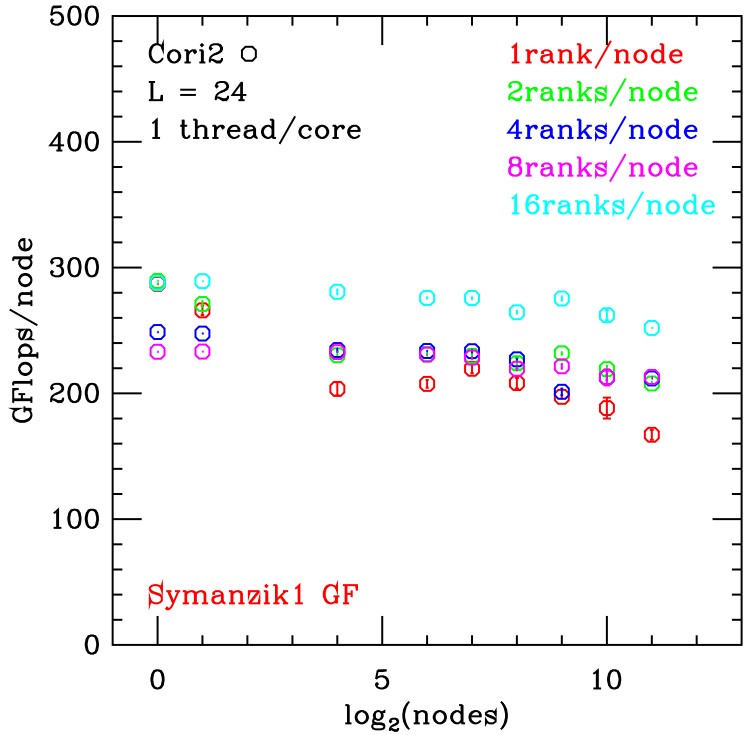
1. Reduced amount of calculations (flops) to 34%

2. Extra temporary data storage for 3-link staples

More shown, weak scaling on up to 2048 nodes, Cori2



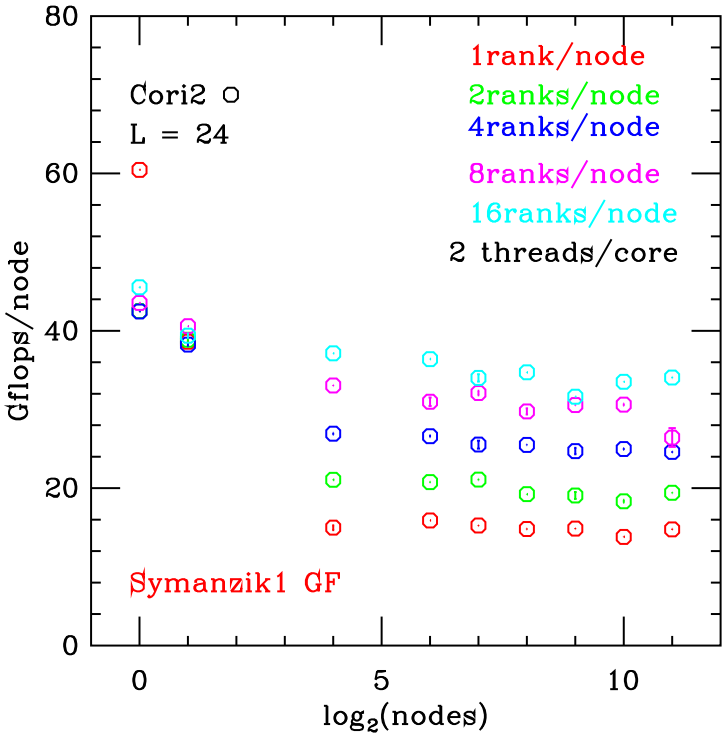
Left: MILC baseline



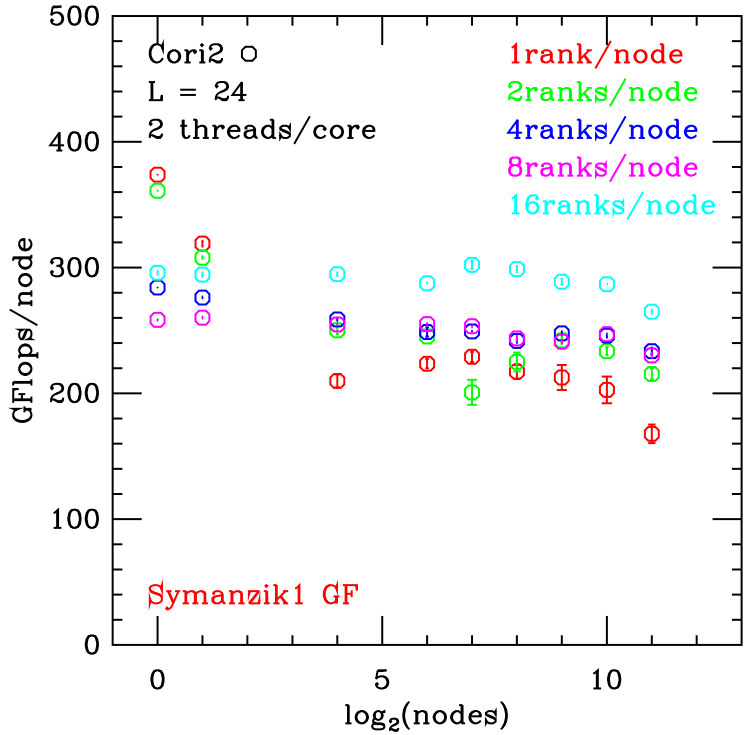
Right: QPhiX

Various MPI rank / OpenMP thread combinations, L = 24

… continue with 2 threads per core



Left: MILC baseline

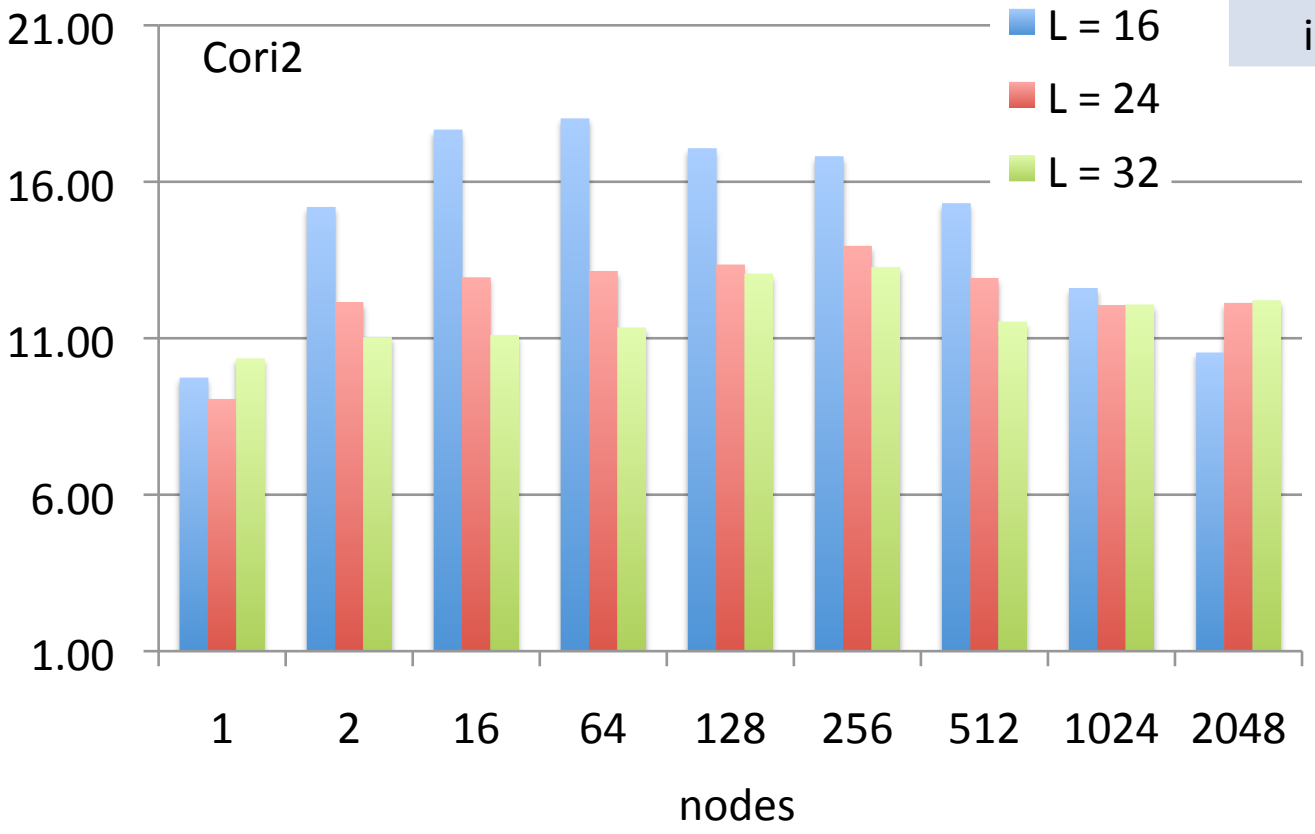


Right: QPhiX

Hyper-threading helps

Right: Performance chart on Cori2,
comparing baseline MILC (2nd column)
and QPhiX (3rd column) gauge force

Down: Speedup with QPhiX, including data reconstruction,
preliminary

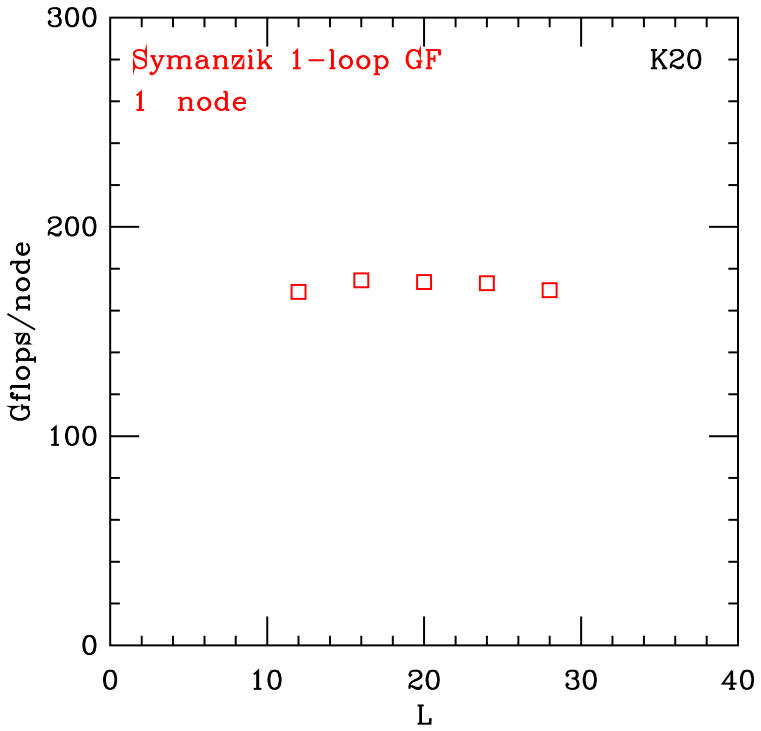


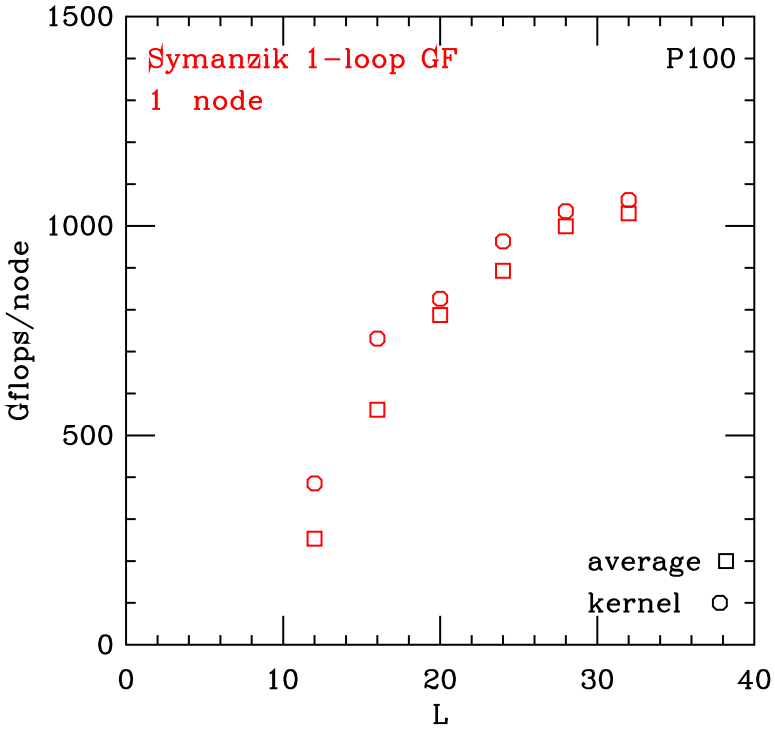
\	MILC b.	w. QPhiX
Gflops, 1node	60	380 (kernel)
Scaling factor (16 ≤ L ≤ 32)	0.4 ~ 0.85	0.65 ~ 0.9
Arithmetic intensity	1.16	2.83

Modified algorithm vs. MILC GF

1. Reduced amount of calculations to 40%, extra temporary momentum storage
2. Modified memory access pattern, similar as in dslash (gather from eight nearest neighbors), increased cache reuse
3. Reduced amount of communications, better scalability

GPU one node, QUDA





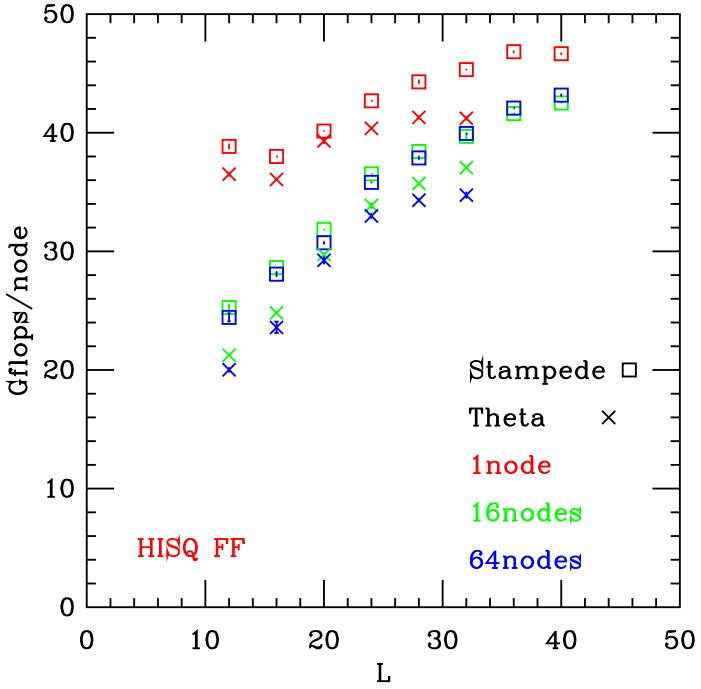
Left: K20, L up to 28

Right: P100, L up to 32

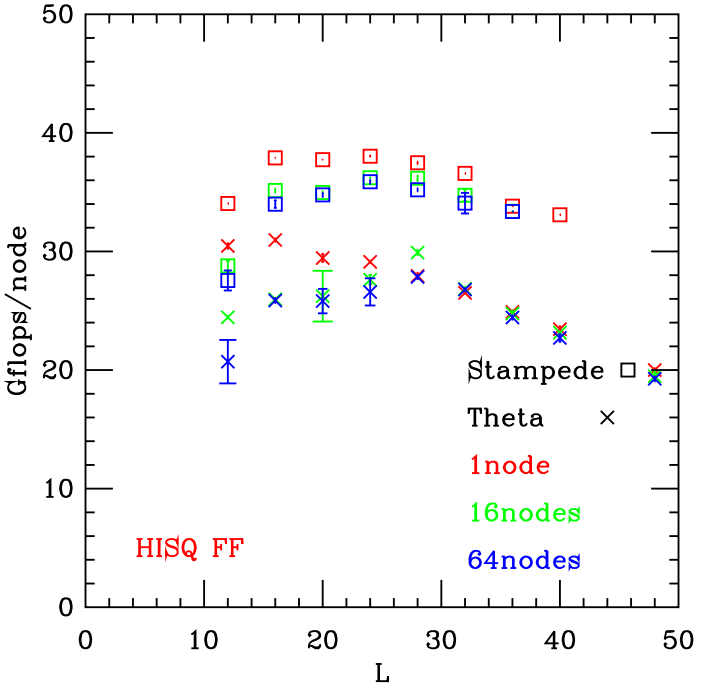
Performance excludes overhead, preliminary

Benchmarks: HISQ Fermion Force

KNL clusters

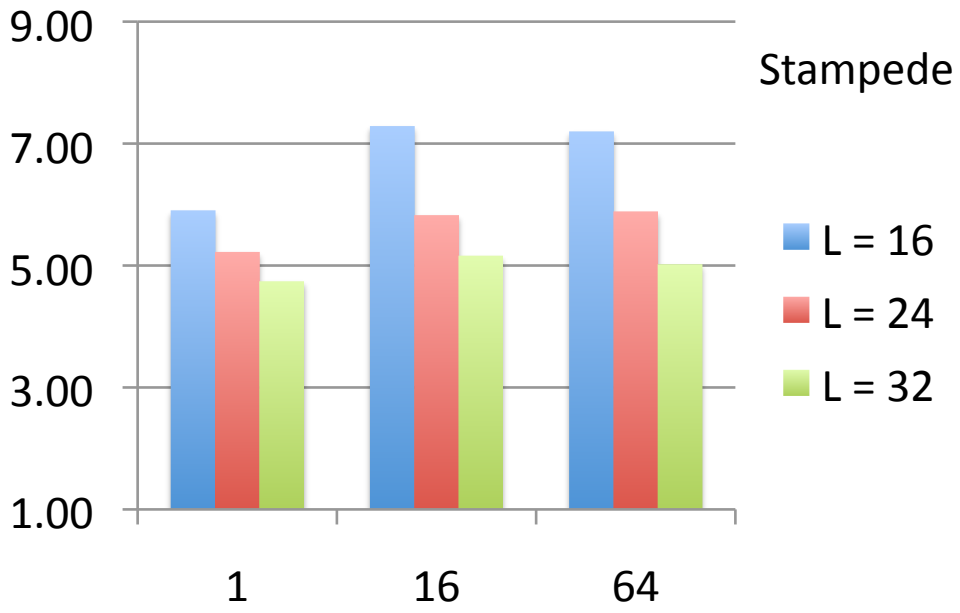


Left: MILC baseline,
64 MPI ranks per node



Right: QOPQDP,
64 MPI ranks per node

Speedup with QOPQDP



Modified algorithm vs. MILC FF

1. Reduced amount of calculations (flops) to 17%
2. Extra temporary data storage for 3-link and 5-link staples

Conclusion

- MILC staggered multi-mass CG on both CPU (Intel KNL) and GPU (Nvidia P100) performance
  - one node saturates over 80% performance roof, bounded by memory BW
  - gains over three times from K20 to P100, kernel routine
  - bounded by network on CPU clusters

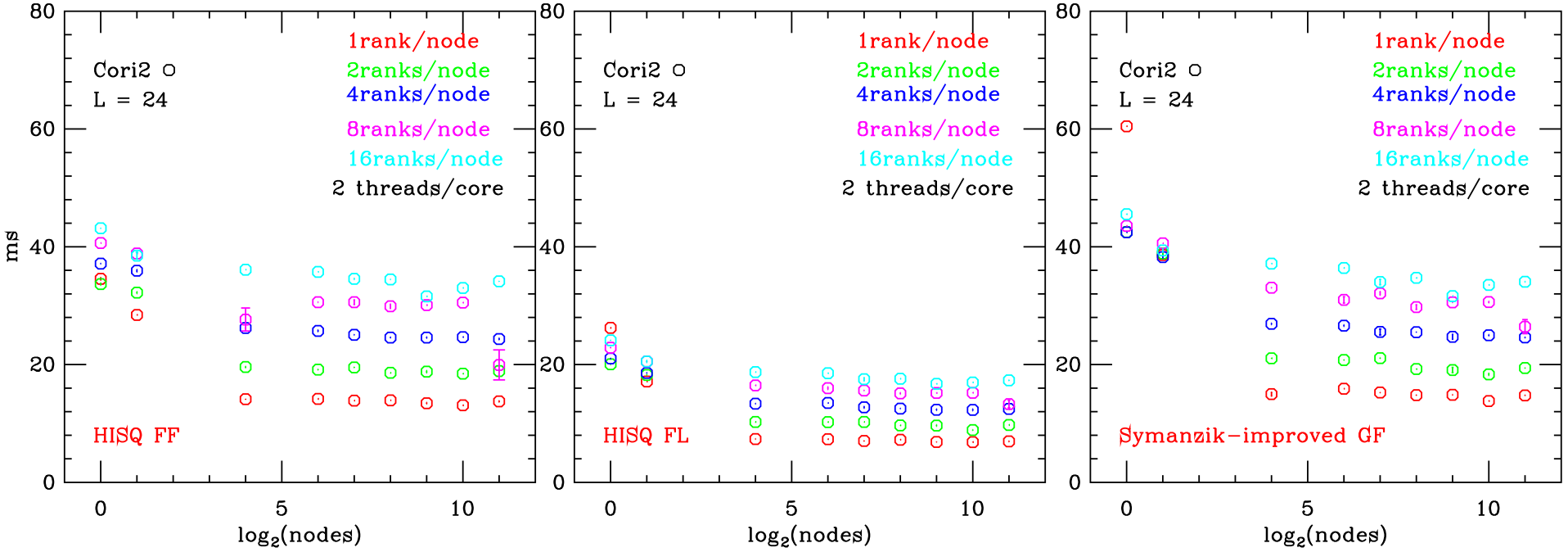
- Other major routines with preliminary performance gain
  - Symanzik GF >10x on Cori2 KNL cluster from baseline MILC code
  - Symanzik GF kernel 5x on P100 compared to K20, single node
  - HISQ FF 7x on Stampede up to 64 nodes from baseline MILC code

- Outlook to other algorithms, e.g., blocked (multi-RHS) staggered CG, staggered multigrid CG

Thank you!

Backup slides

MILC code breakdown: FF, GF, FL



… more shown, 2 threads per core

Hyper-threading helps

Staggered QPhiX Library Structure

- Structure of arrays for base elements, e.g.,
  - *Real KS[3][2][VECLEN];*
  - *Real Gauge[8][3][3][2][VECLEN];*
- Sub-block checkerboard lattice of dimension 2 along 3 (double precision) or 4 (single precision) directions
  - Special treatment on the virtual boundary
- Pencil blocks (along x, y, z direction) of size Nx/2 × By × Bz on each thread at a time, By = 4, Bz = 1
  - Large number of blocks against load imbalance for L ≥ 16
  - Reduced L2 cache misses, e.g., L=16, blocked staggered color vector size = 12KB
- Vector intrinsic instructions (use code generator) replace scalar arithmetic instructions