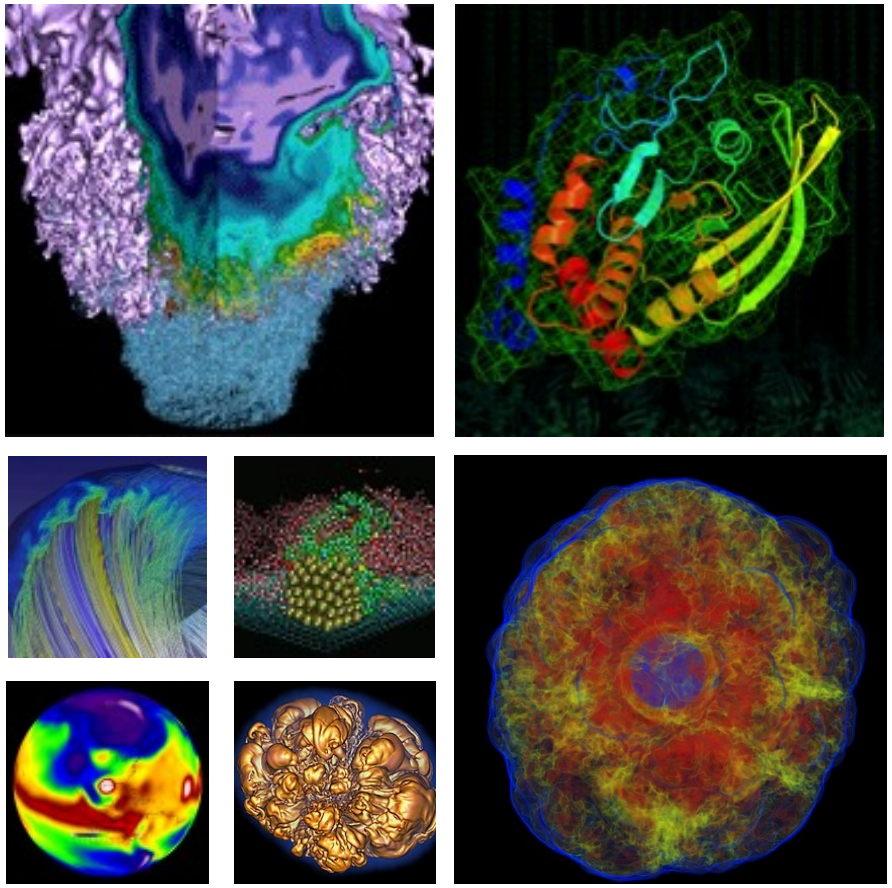
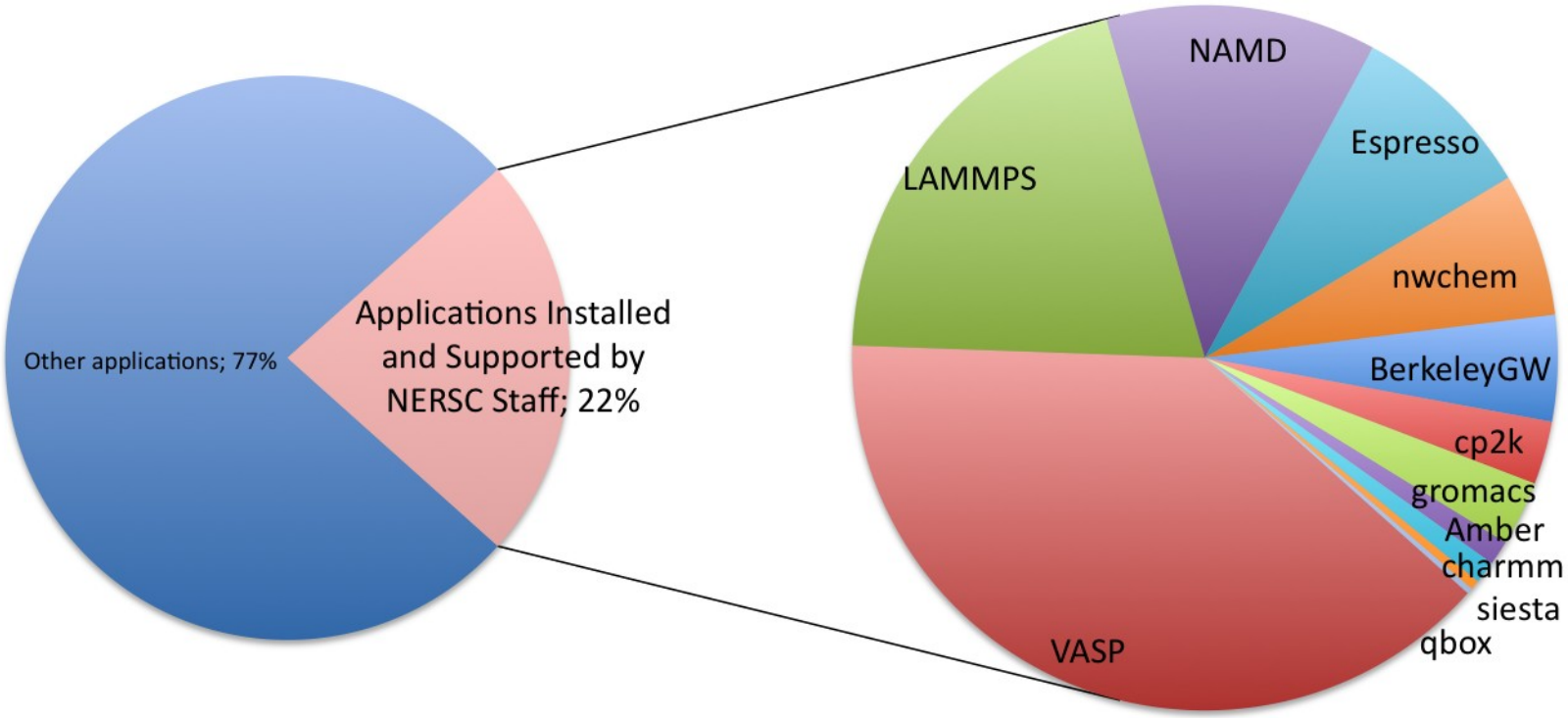
Compiler and Library Performance in Material Science Applications on Edison

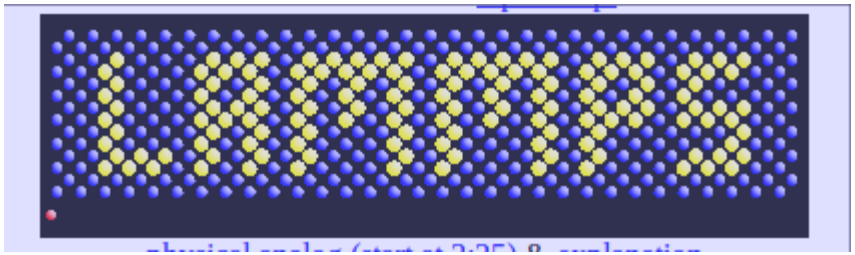


Jack Deslippe and Zhengji Zhao

Materials Science Application Support at NERSC



The Top 6 Material Science + Chemistry Codes at NERSC



BerkeleyGW

NAMD
Scalable Molecular Dynamics

Question:

How do compilers and libraries affect performance in these apps??

Methodology

Test: Intel, GNU and Cray Compilers.

Test: FFTW2&3, LibSci, MKL and internal libraries.

-Test each application across a range of MPI tasks and OpenMP threads (if applicable)

-Run out of Lustre scratch. Minimize IO at runtime when possible.

-Ru each test twice. Keep fastest value.

-Threaded applications use:

% aprun -S <even number per numa> -cc numa_node -ss ...

Methodology Continued

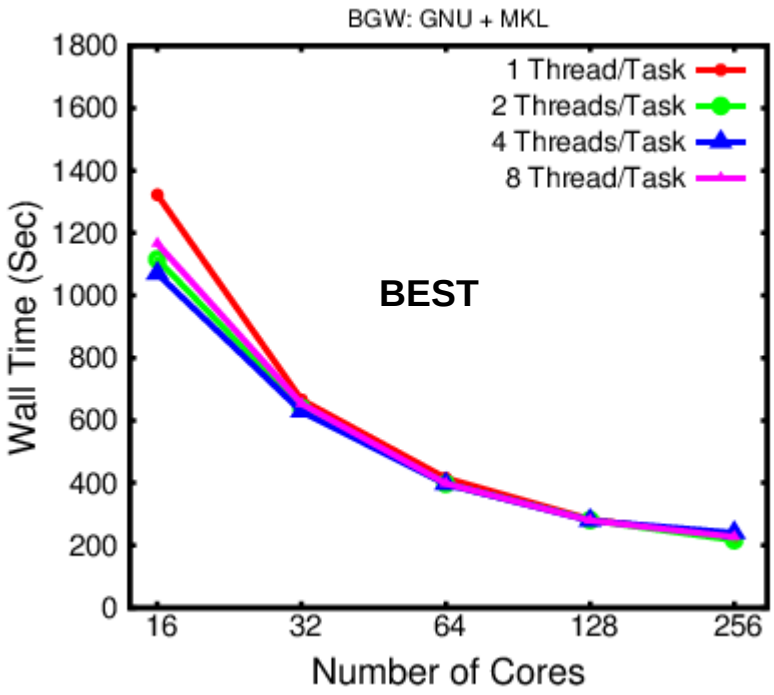
-Compiler Options:

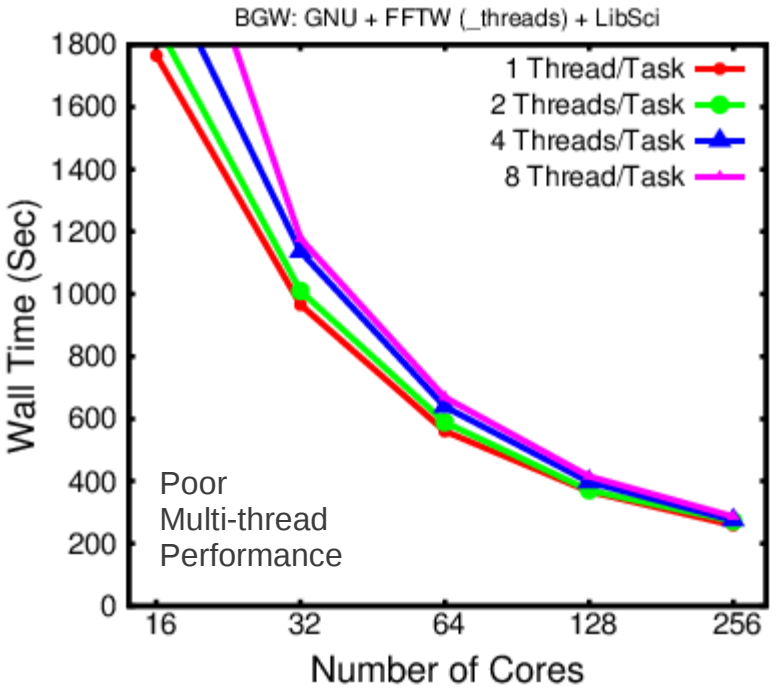
GNU: -O3 -fast-math
Cray: (default)
Intel: -fast -no-ipo

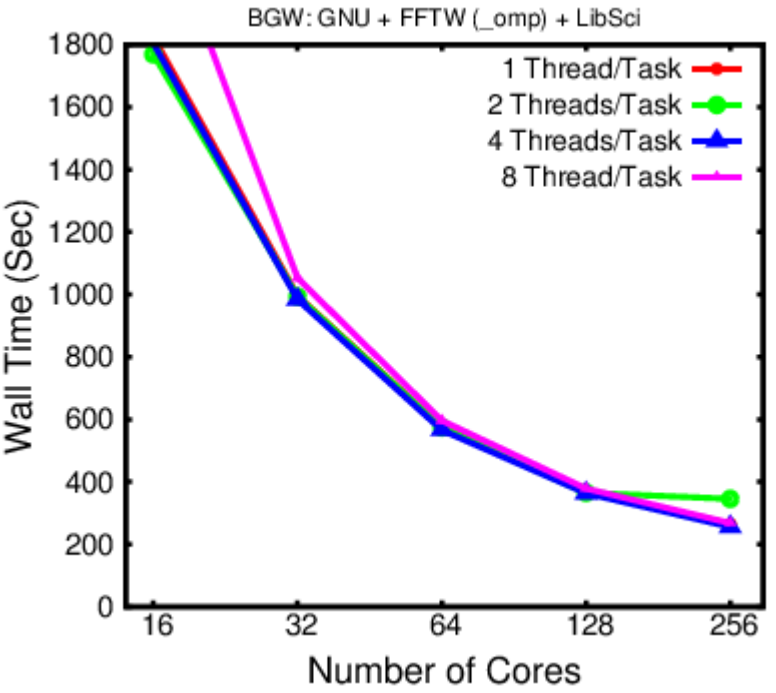
-Since there is no Cray specific MKL library. For Cray compiler we link against the MKL GNU libs.

BerkeleyGW GNU Compiler Summary

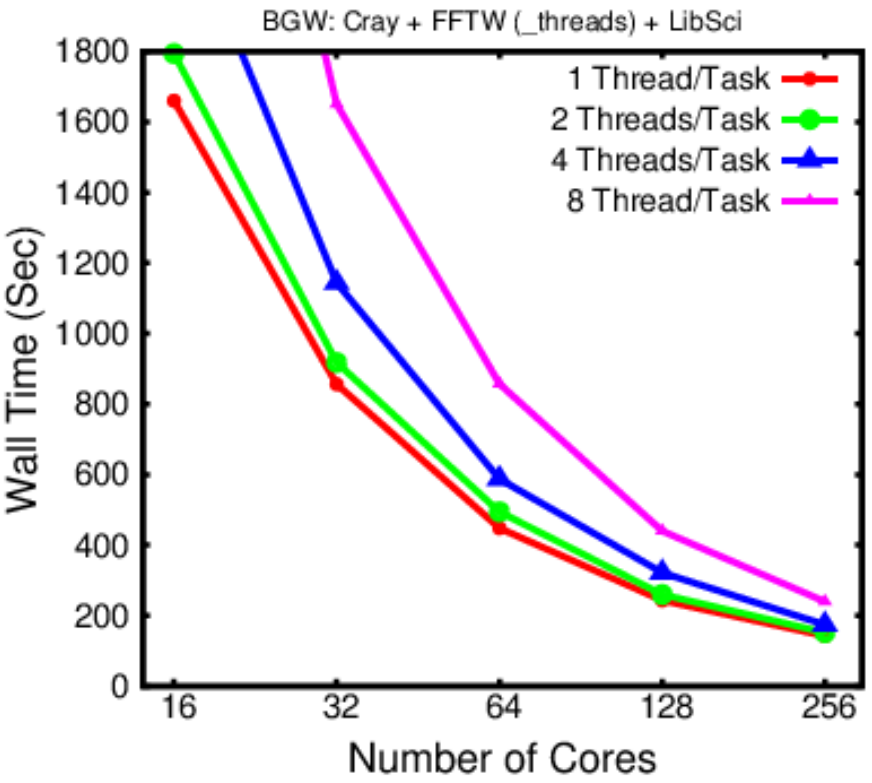
BGW 1.1 (Beta) – (8,0) Carbon Nanotube Example

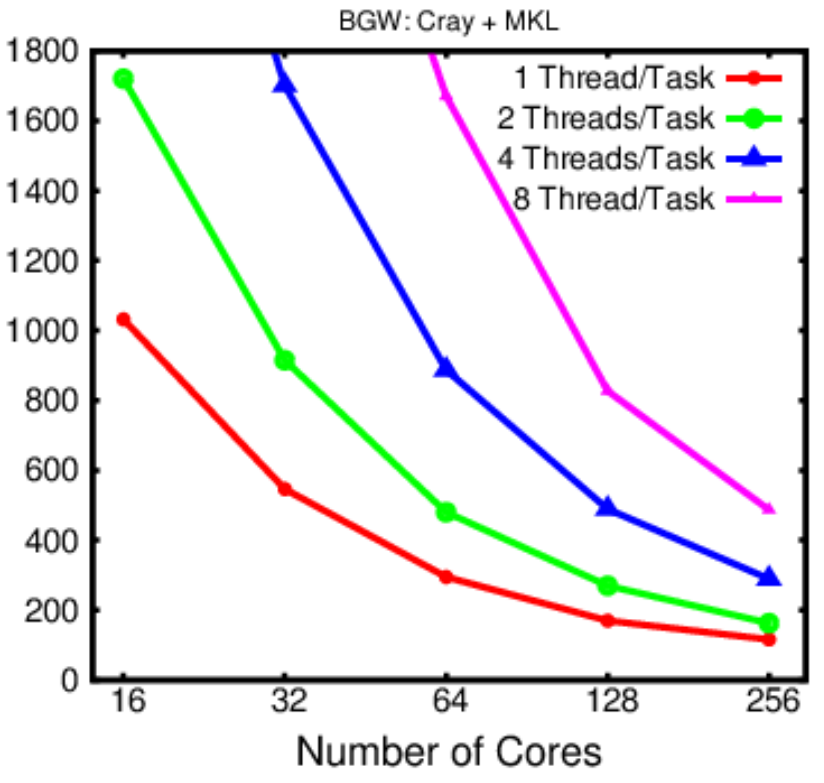






BerkeleyGW Cray Summary

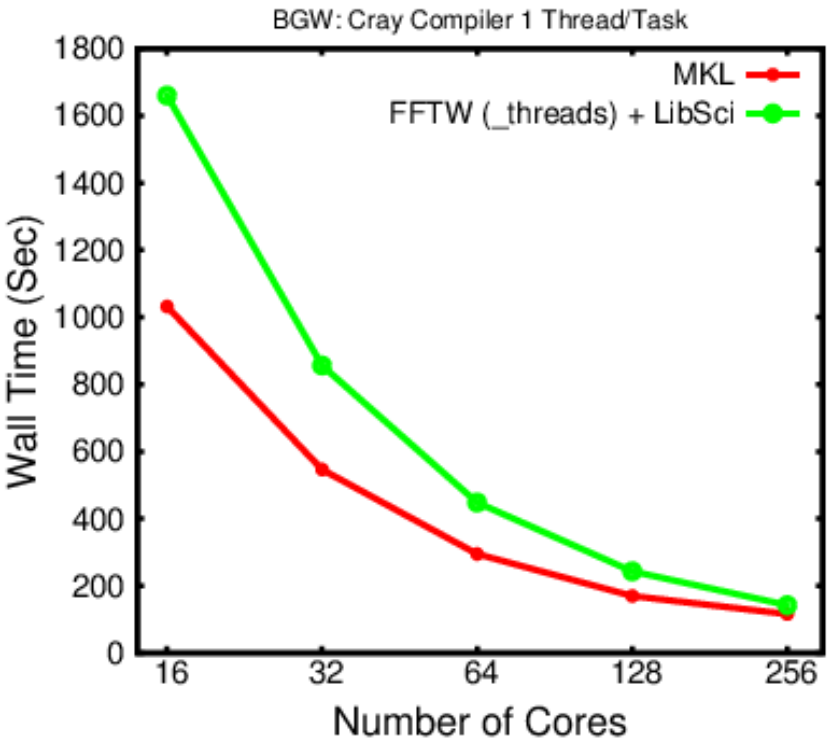


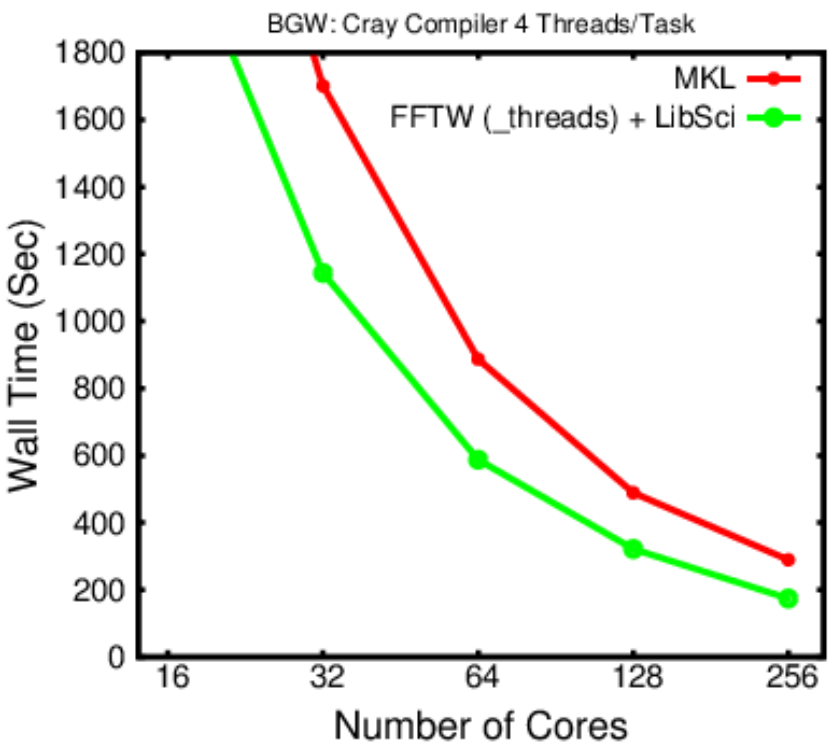


BAD
Default fftw3_threads doesn't play nice with other OpenMP in code. Single thread performance worse than MKL.

BOTH BETTER AND WORSE
Cray OMP + MKL (linked against GNU version) causes very poor performance with more than 1 thread.

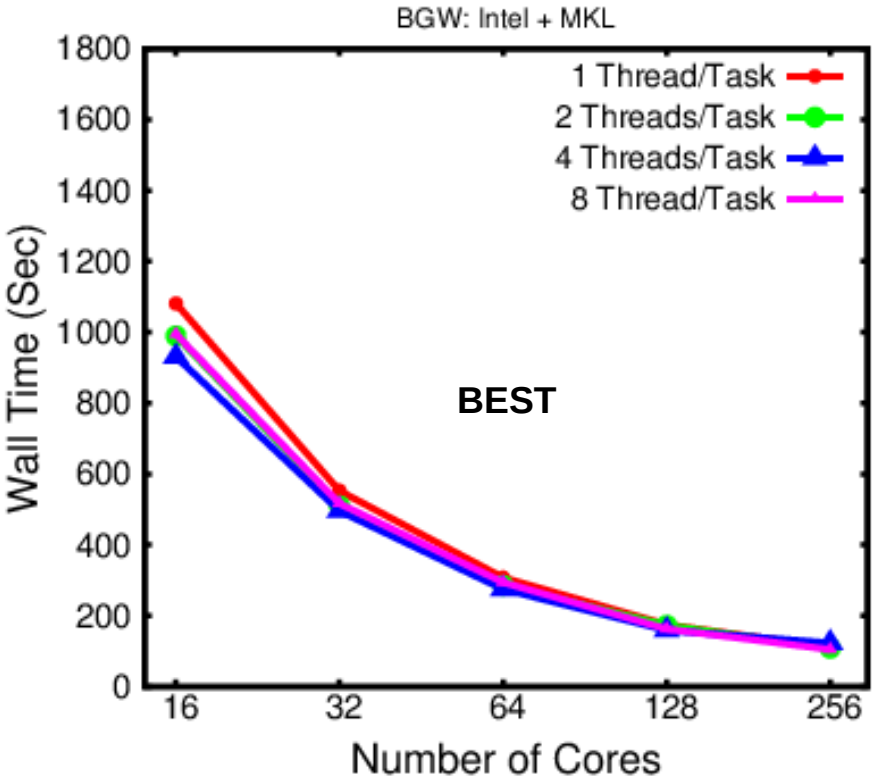
BerkeleyGW Cray Summary

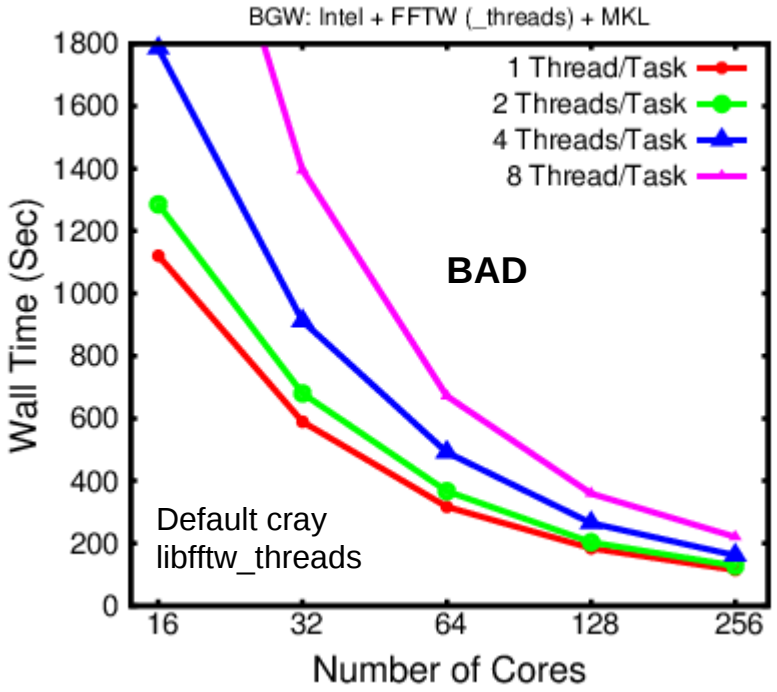


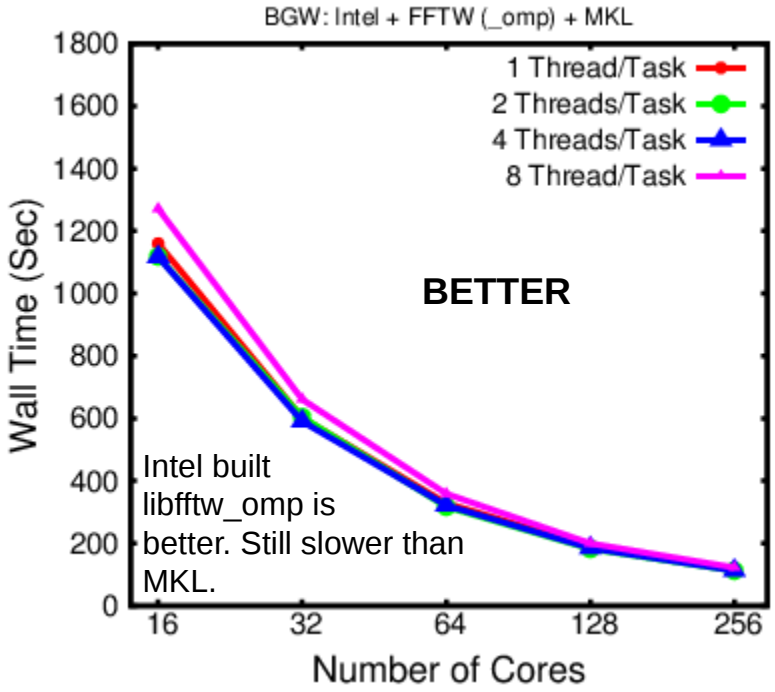


Cray + MKL (linked against GNU version) performs well with 1 thread. Poor multi-threaded performance.

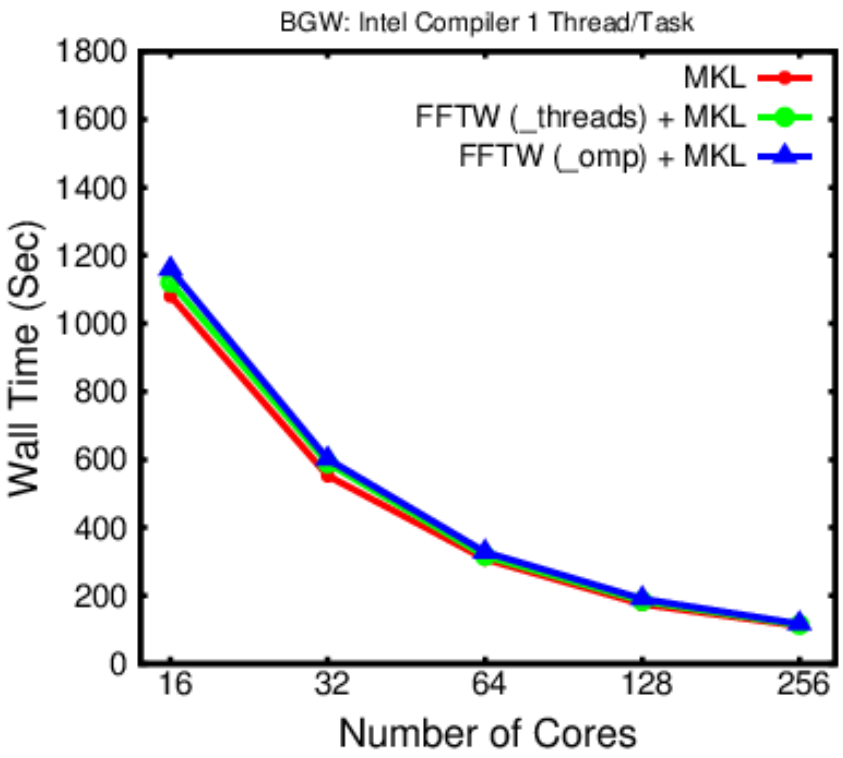
BerkeleyGW Intel Summary

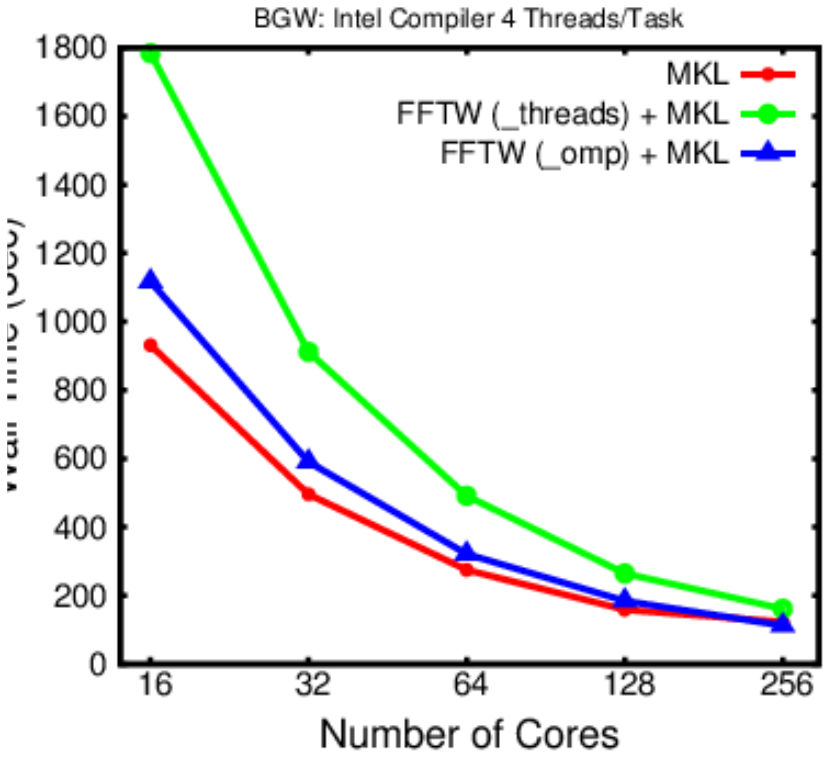






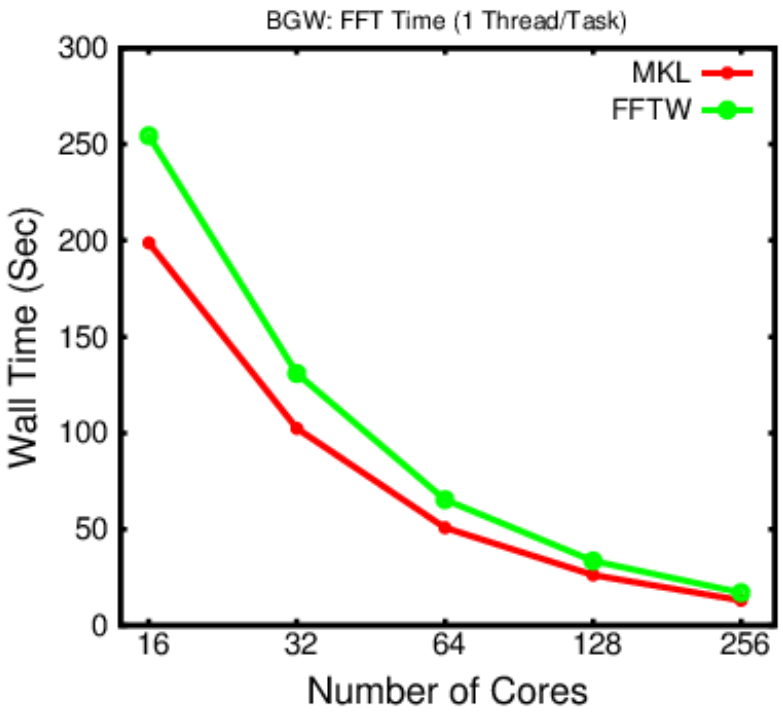
BerkeleyGW Intel Summary

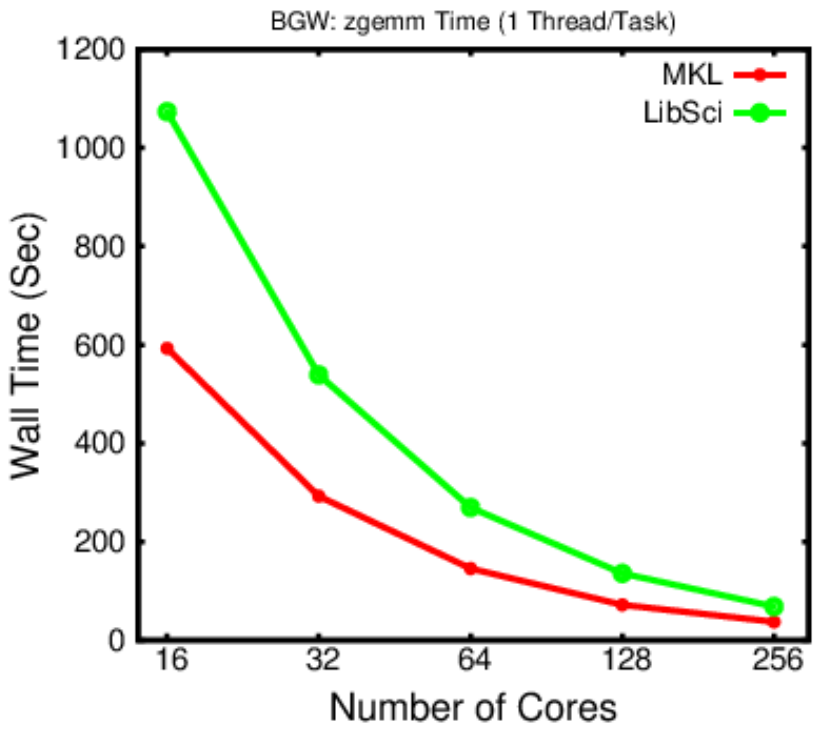




MKL FFTs perform better than FFTW in BerkeleyGW.

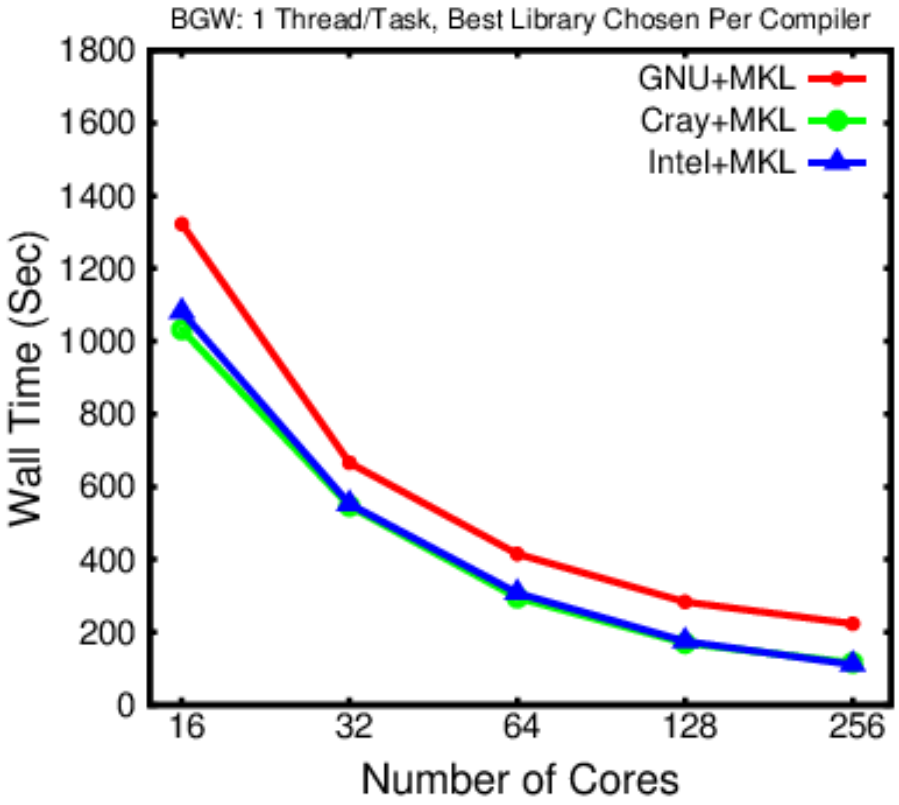
BerkeleyGW Library Summary

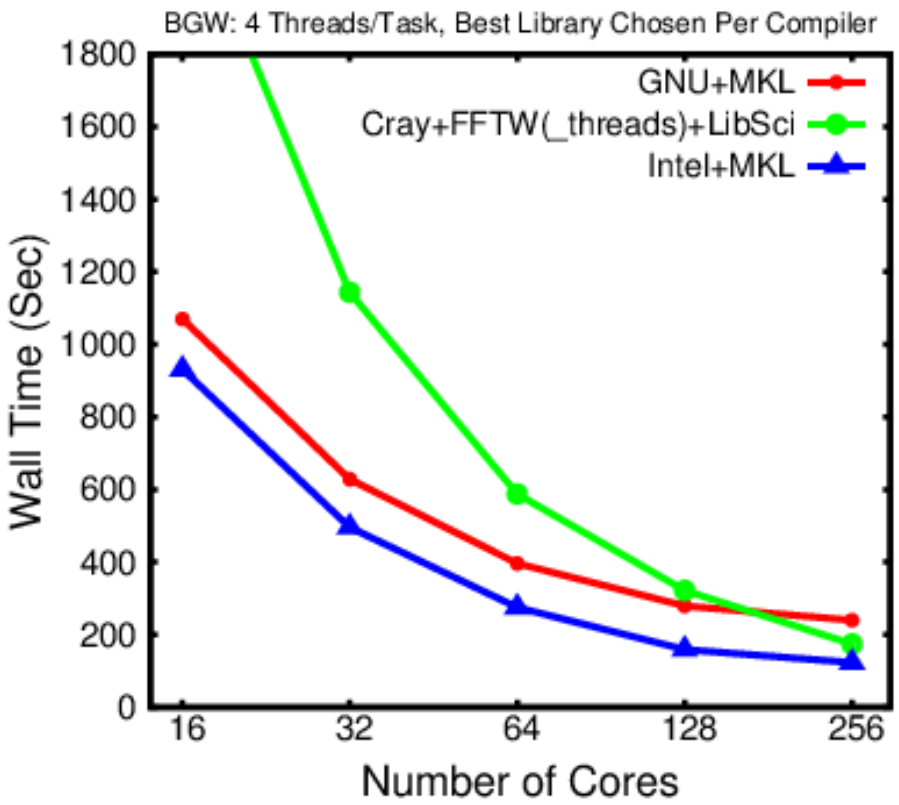




MKL beats FFTW. And MKL beats LibSci. ZGEMM's in LibSci ~ 50% slower than MKL. DGEMMs are within a couple percent. Cray will likely close this Gap.

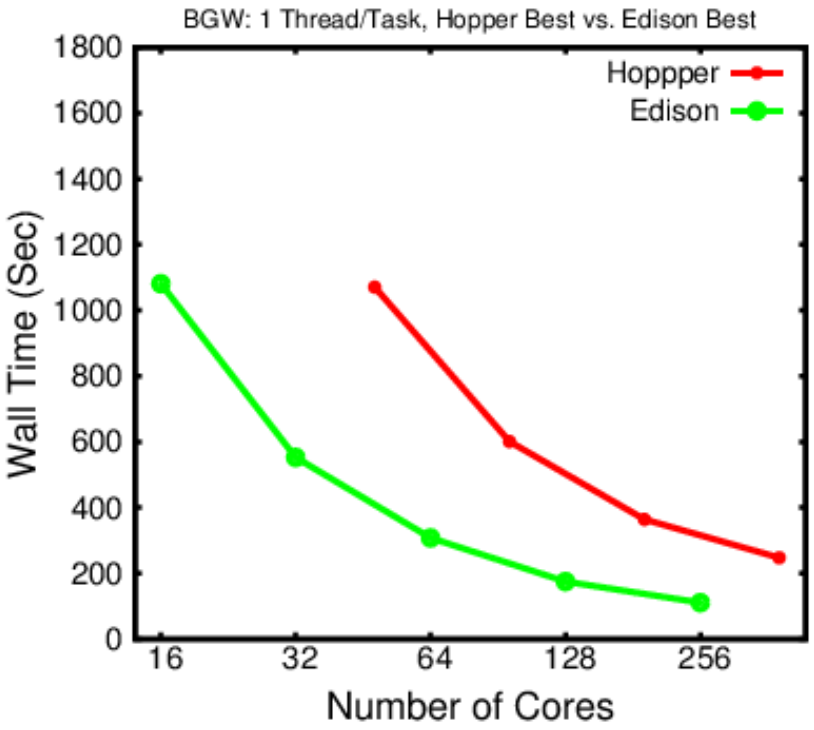
BerkeleyGW Compiler Summary





Intel + MKL is Clear Winner! Cray + MKL is best with 1 Thread.

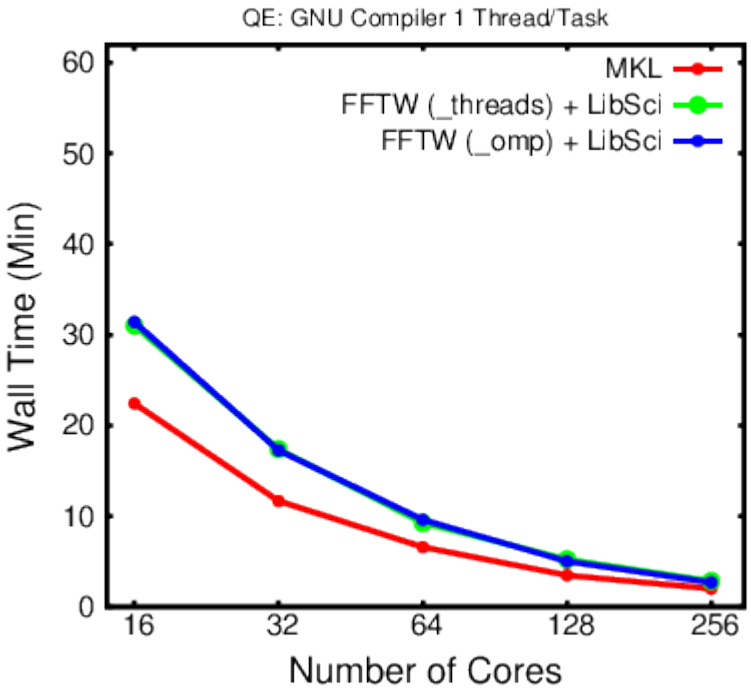
BerkeleyGW Hopper Vs. Edison

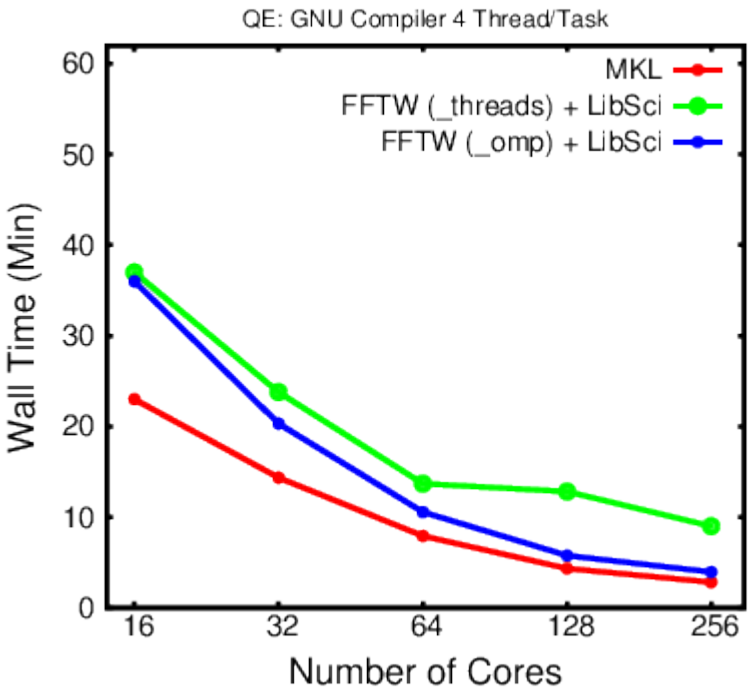


~ 3x Improvement on core per core comparison.

QE GNU Summary

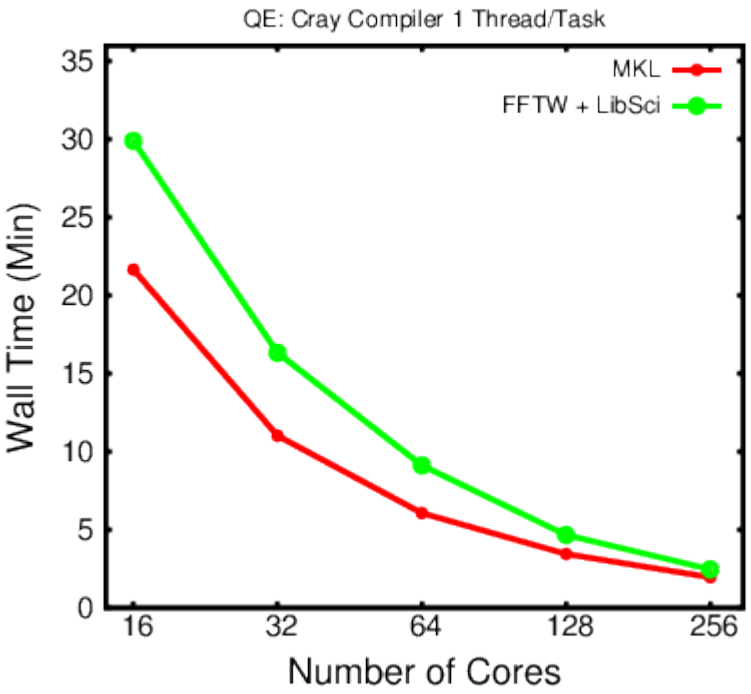
QE 5.0.2. (8,0) Single Walled Carbon Nanotube Example

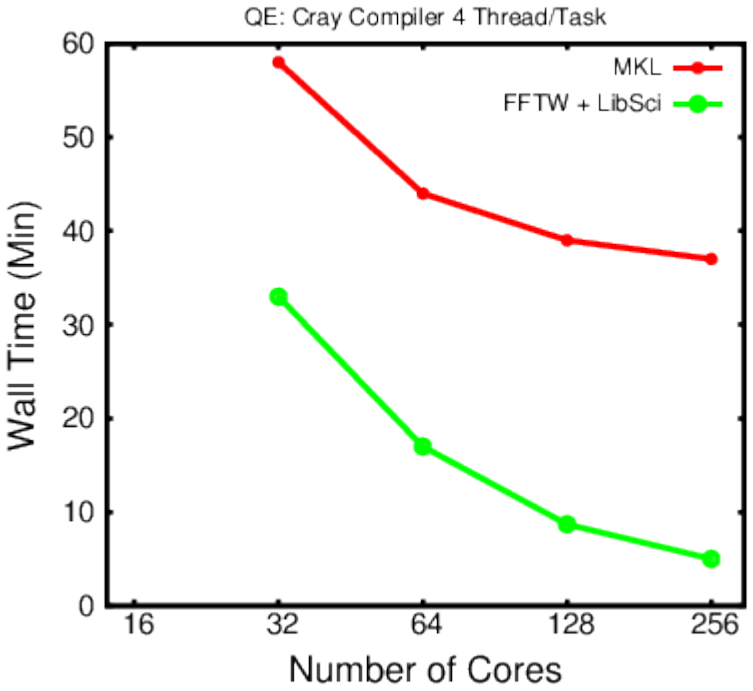




Again, MKL is Faster that FFTW+LibSci

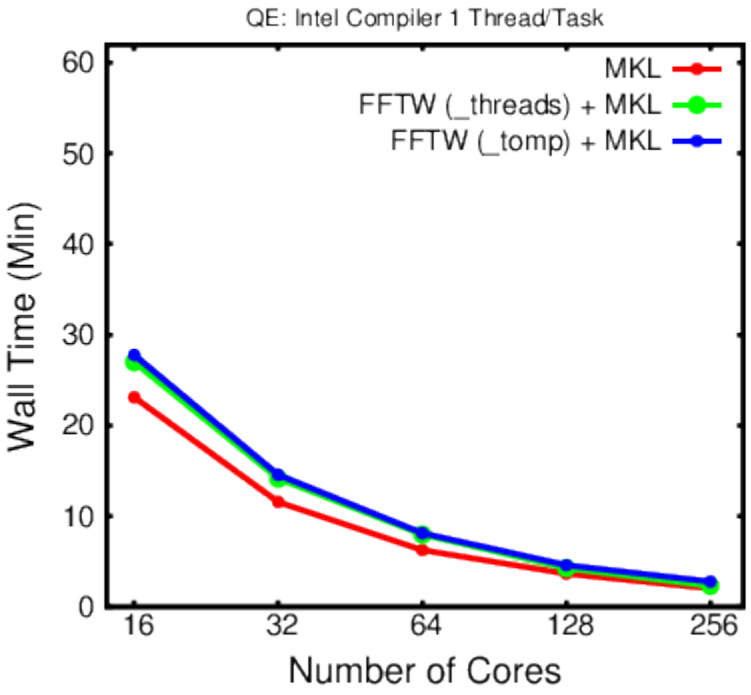
QE Cray Summary

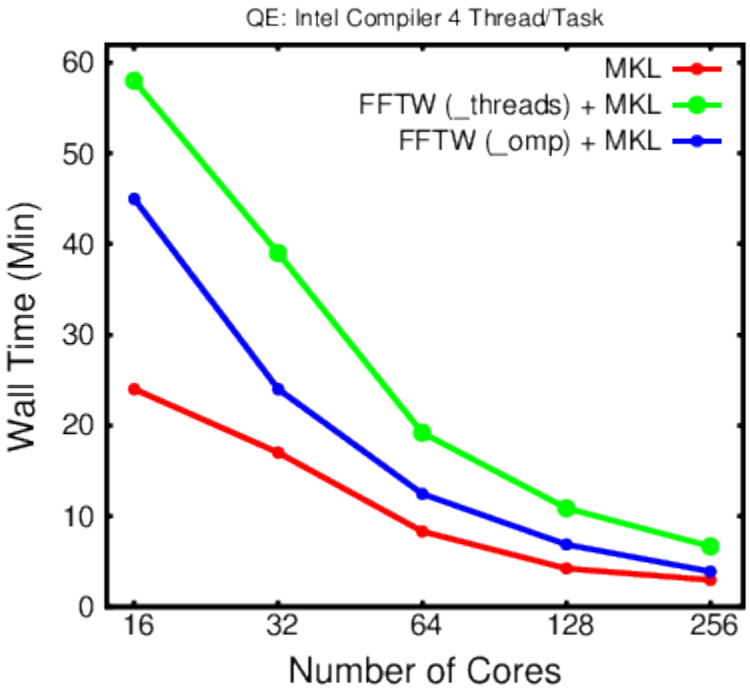




Cray+MKL (linked with GNU MKL) Performs well for 1 Thread. Poorly with multiple threads.

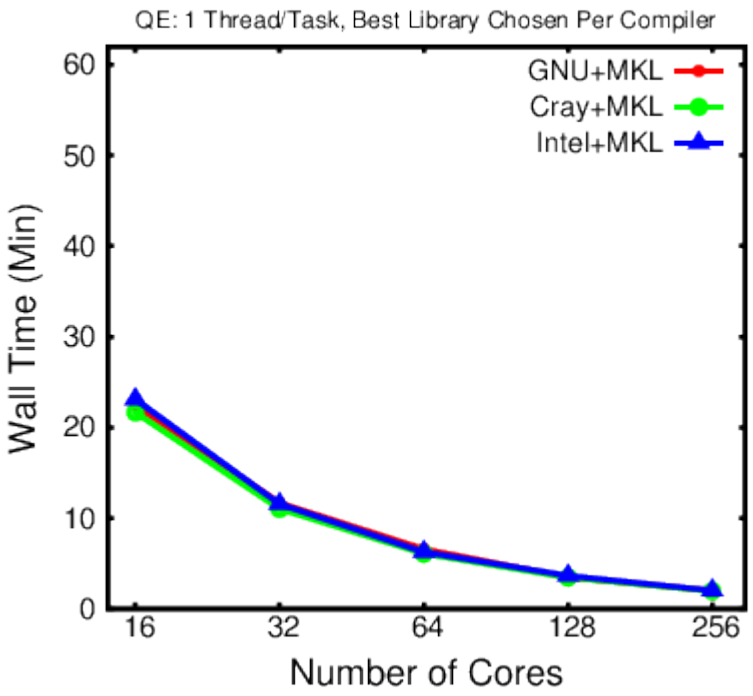
QE Intel Summary

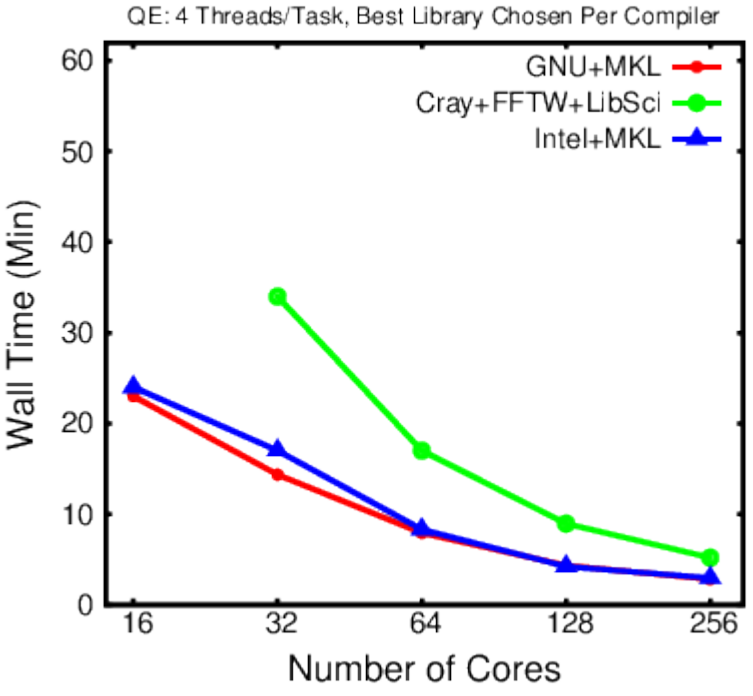




MKL FFTs one again are superior.

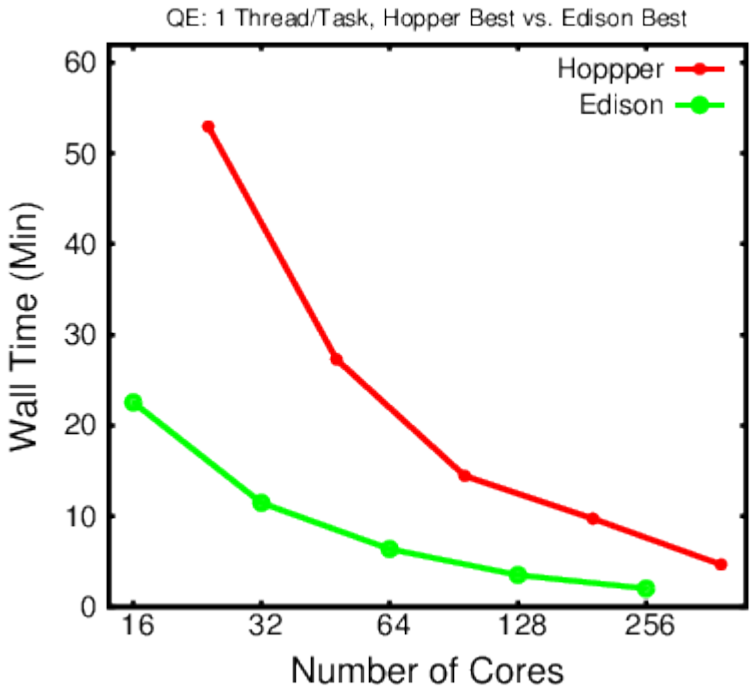
QE Compiler Summary





Cray + MKL fastest combination for 1 thread. GNU + MKL & Intel + MKL are the best overall combinations.

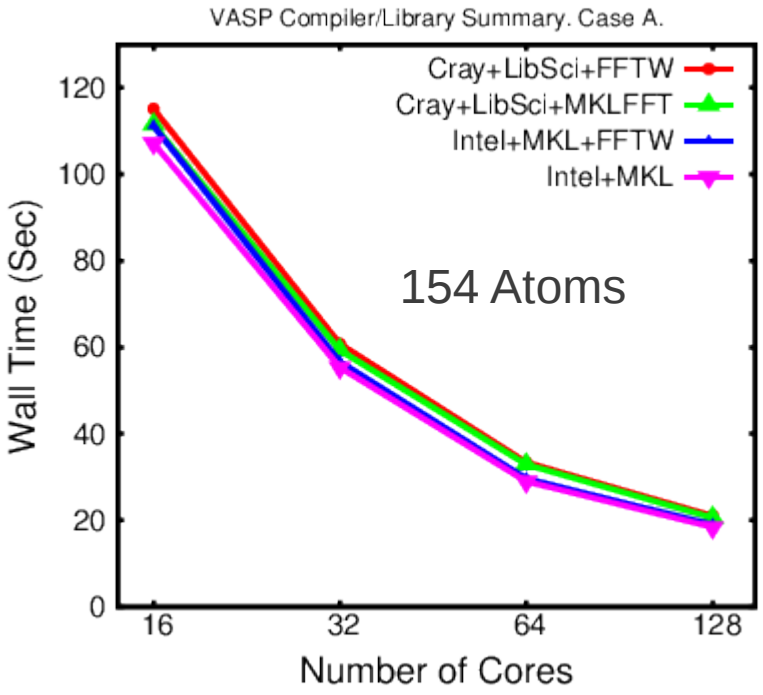
QE Hopper Vs. Edison

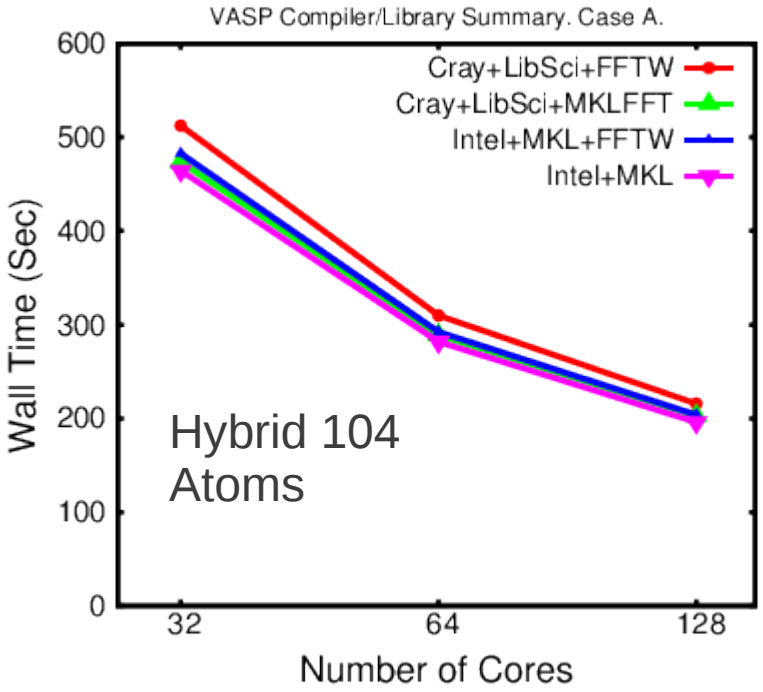


~ 3X Speedup on core-per-core comparison

VASP Summary

Version 5.3.3

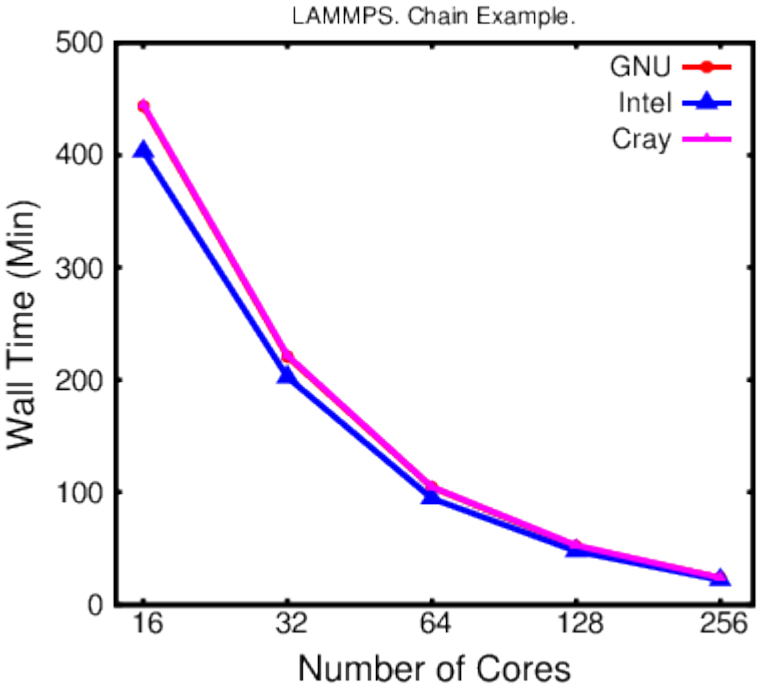


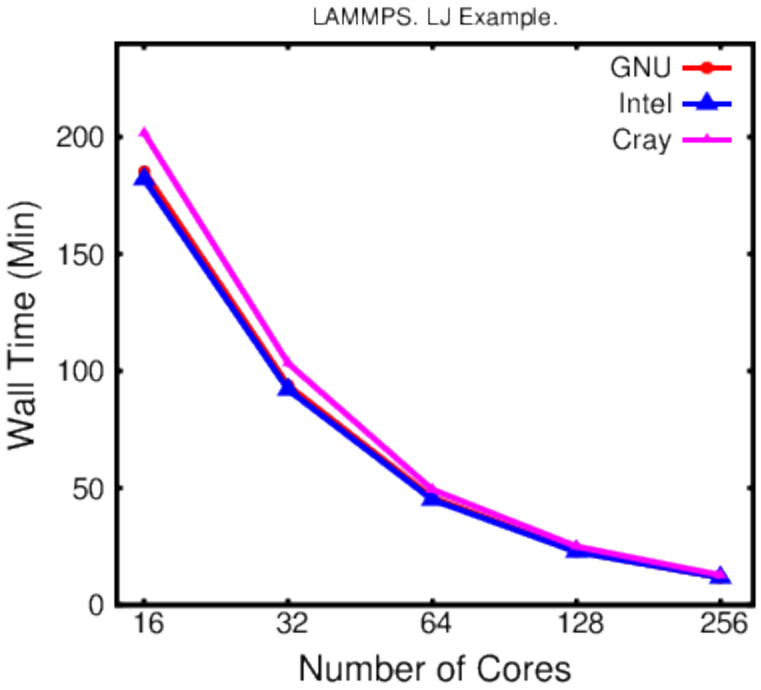


Intel + MKL again the best compiler. Cray + MKL for linear algebra yields runtime problems.

LAMMPS Summary

Version 22Mar13

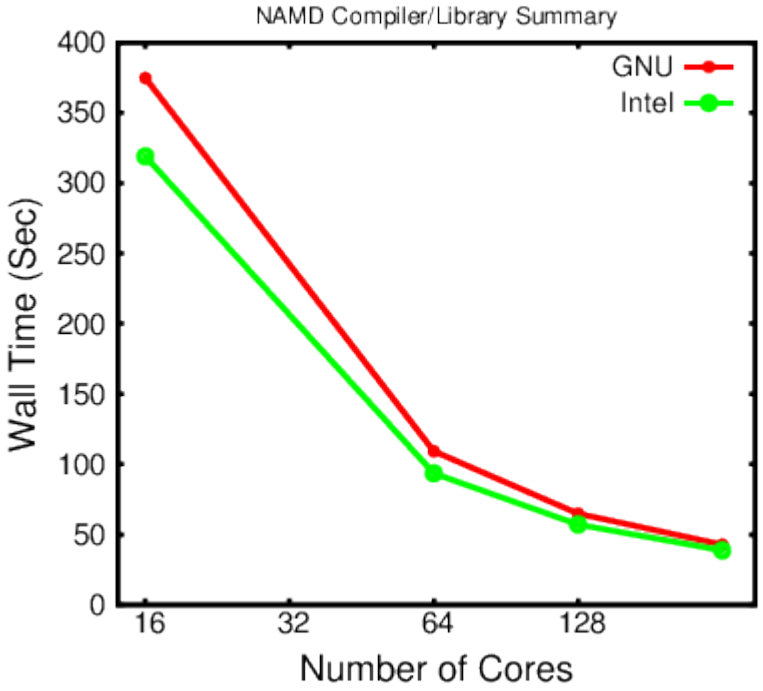




Intel and GNU compilers have the highest performance for LAMMPs. See paper for benchmark descriptions.

NAMD Summary

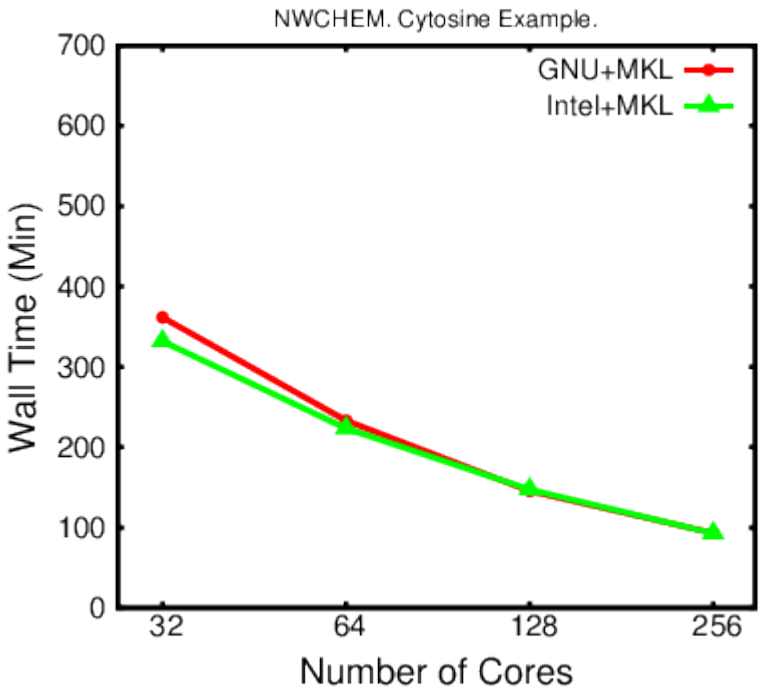
STMV 1,066,628-atom system



Intel once again is the highest performing compiler. See paper for benchmark description.

NWCHEM Summary

Version 6.1.1



Used armci-mpi with GA 5.0. Intel again is highest performing compiler.

Summary

1. MKL outperforms LibSci and FFTW on Edison.

2. Additional performance problems observed in libfftw3_threads and MKL when using multiple thread implementations.

3. Intel was the best overall compiler on all codes. In large part due to library support and compilation success rate.

National Energy Research Scientific Computing Center