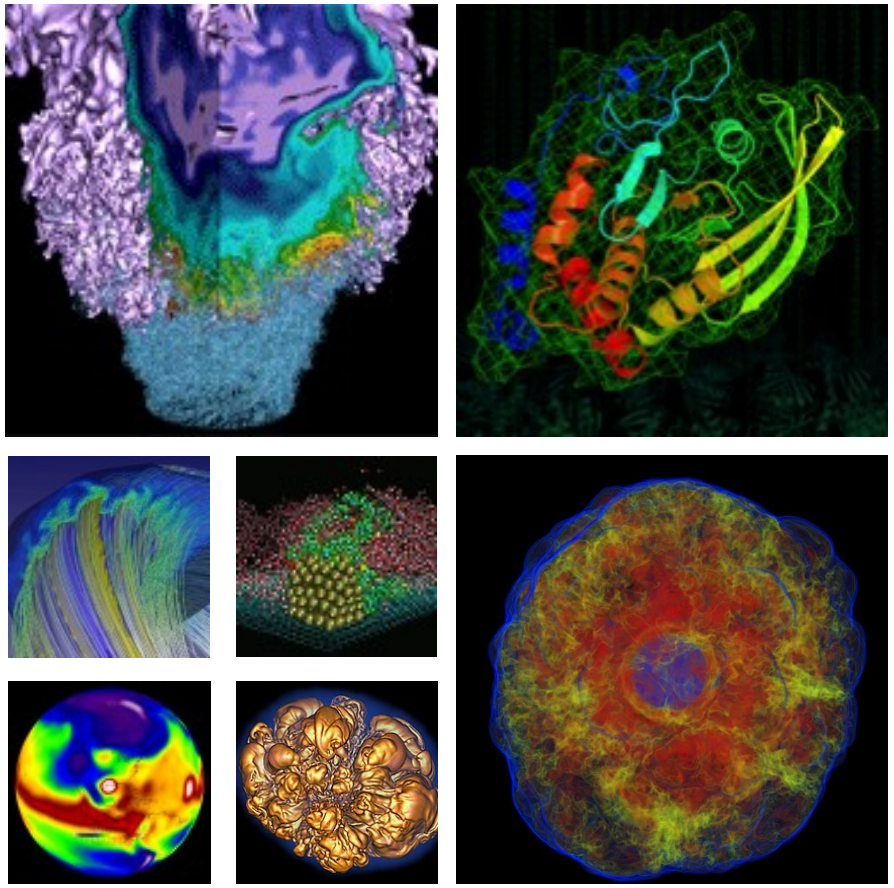
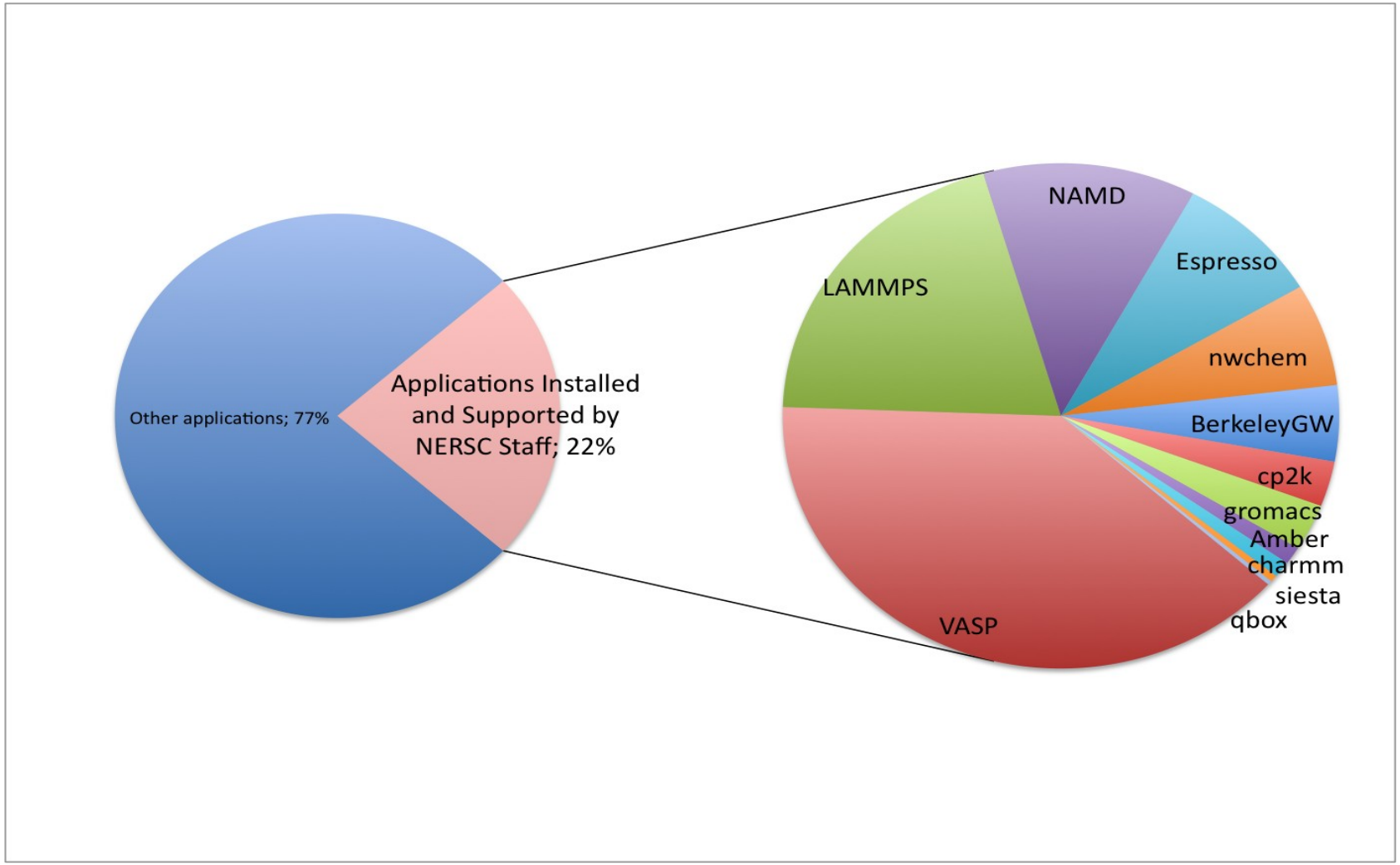
Compiler and Library Performance in Material Science Applications on Edison



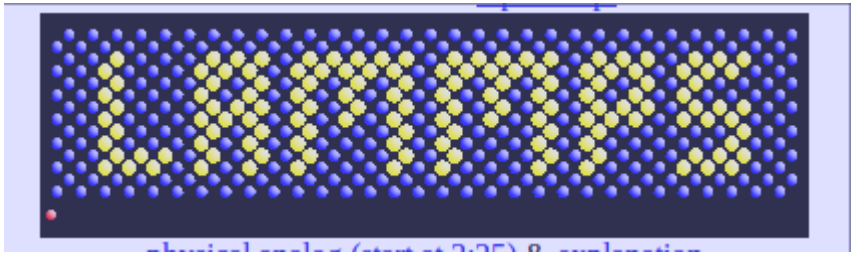
Jack Deslippe and Zhengji Zhao

Materials Science Application Support at NERSC



The Top 6 Material Science + Chemistry Codes at NERSC

b-initio
VASP
ackage
ienna
imulation



NWCHEM
HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE

QUANTUM

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 math 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.

-Run each test twice. Keep fastest value.

-Threaded applications use:

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

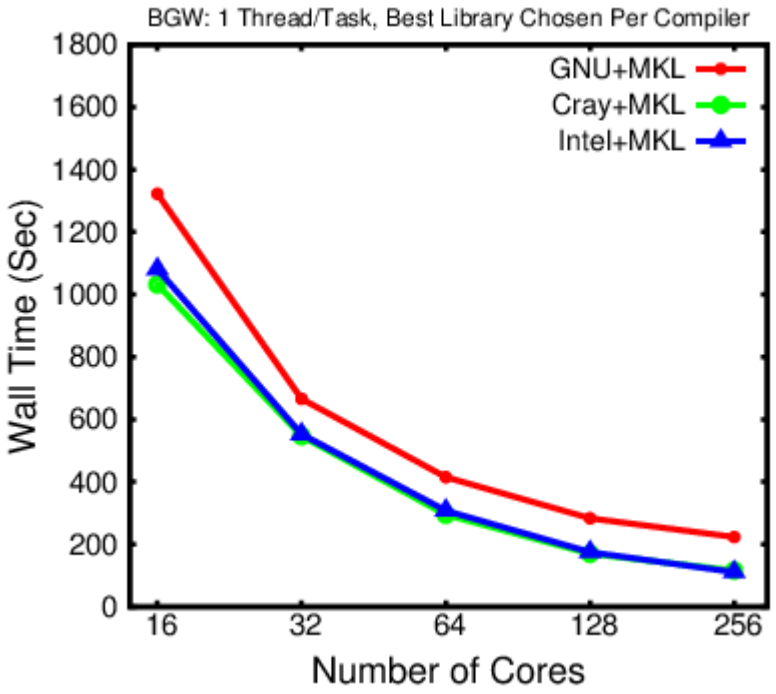
Methodology Continued

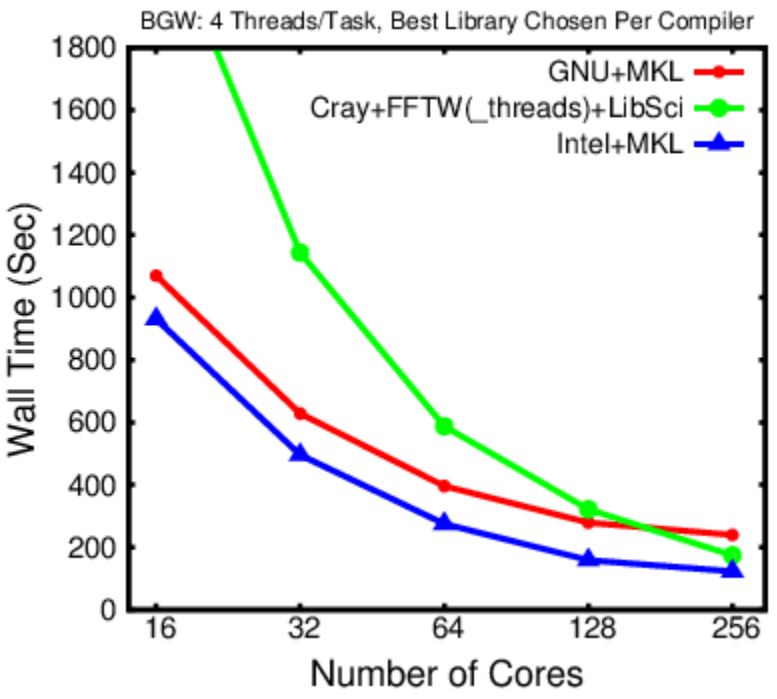
-Compiler Options*:

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

http://www.nersc.gov/users/computational-systems/edison/performance-and-optimization/

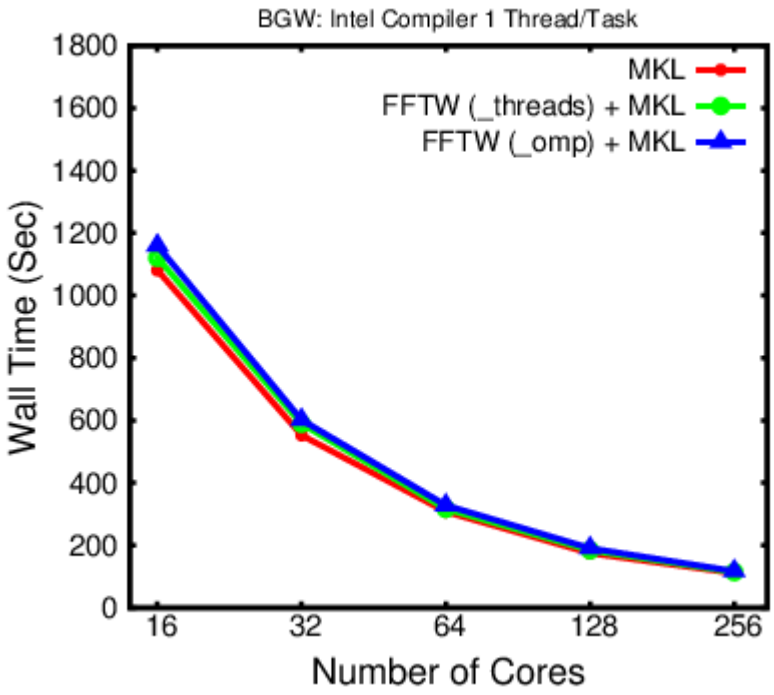
BGW 1.1 (Beta) – (8,0) Carbon Nanotube Example

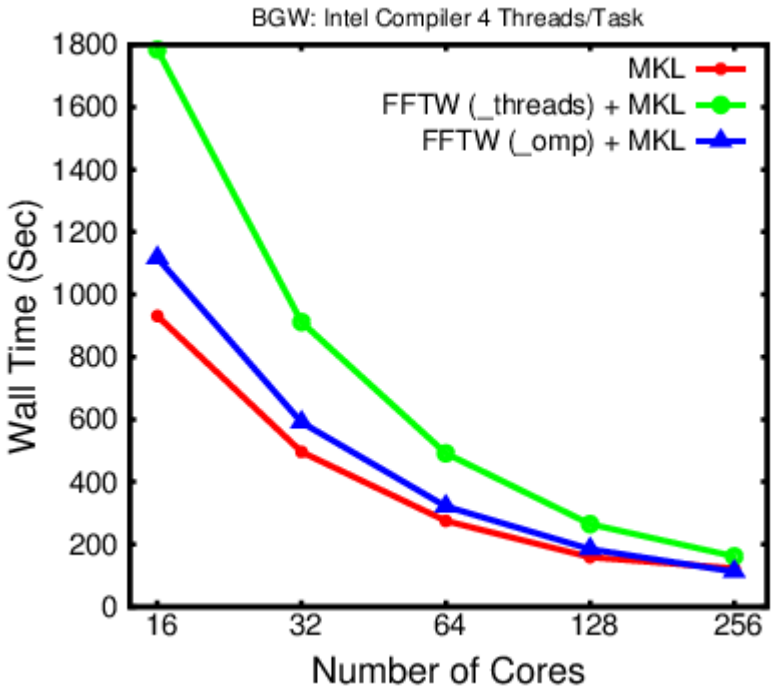




Intel + MKL is Clear Overall Winner. Cray + MKL is best with 1 Thread. Cray + MKL threaded performance suffers.

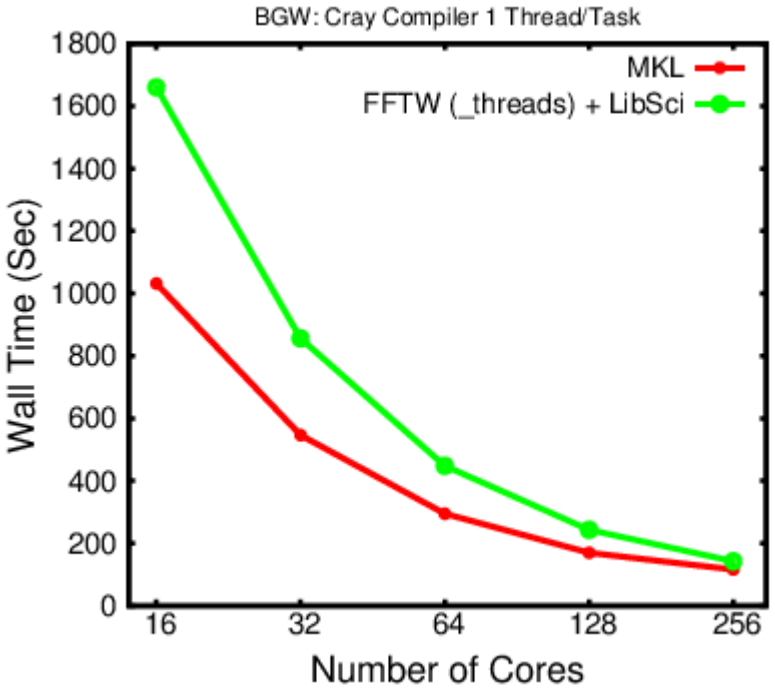
Comparing FFT Libraries with Intel Compiler

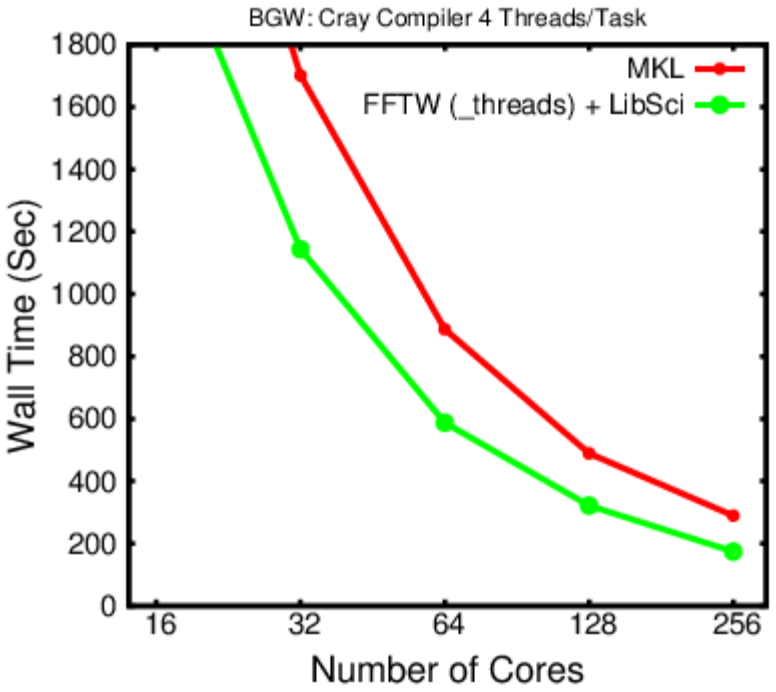




MKL FFTs perform better than FFTW in BerkeleyGW.

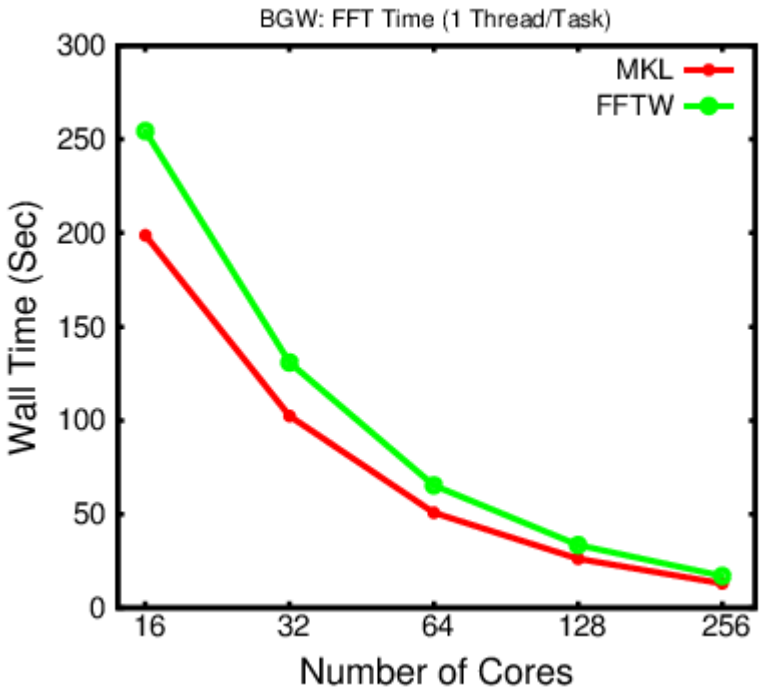
BerkeleyGW Cray Summary

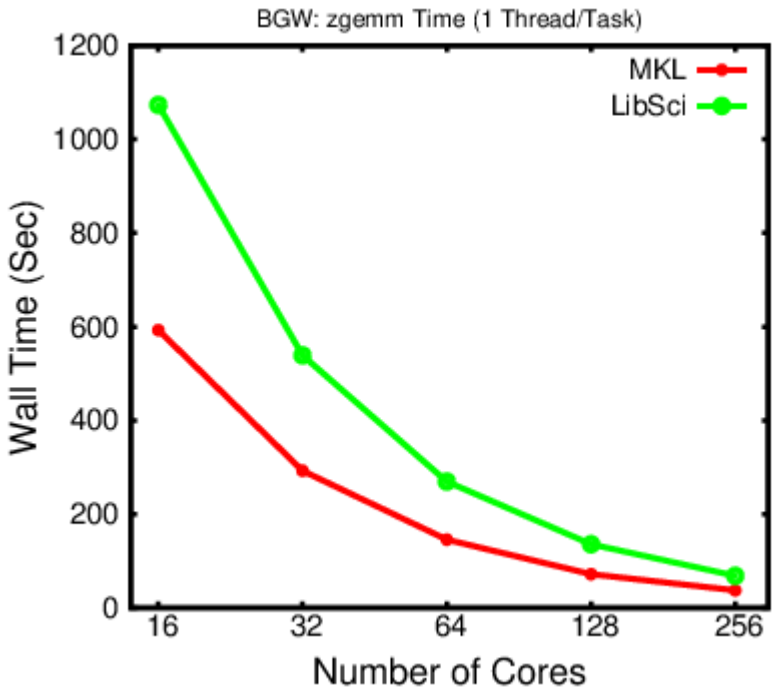




Cray + MKL (linked against GNU version) performs well with 1 thread. Poor multi-threaded performance.

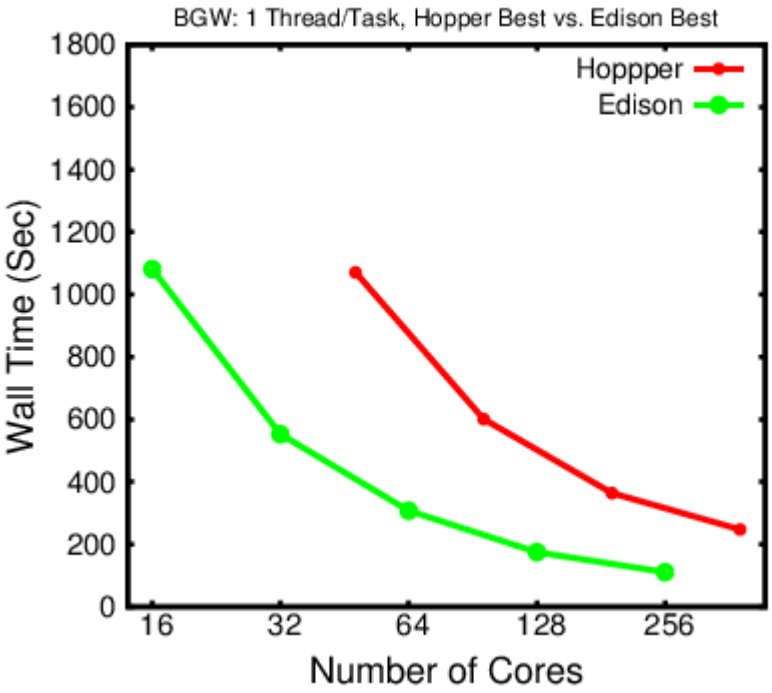
Library Performance With GNU Compiler





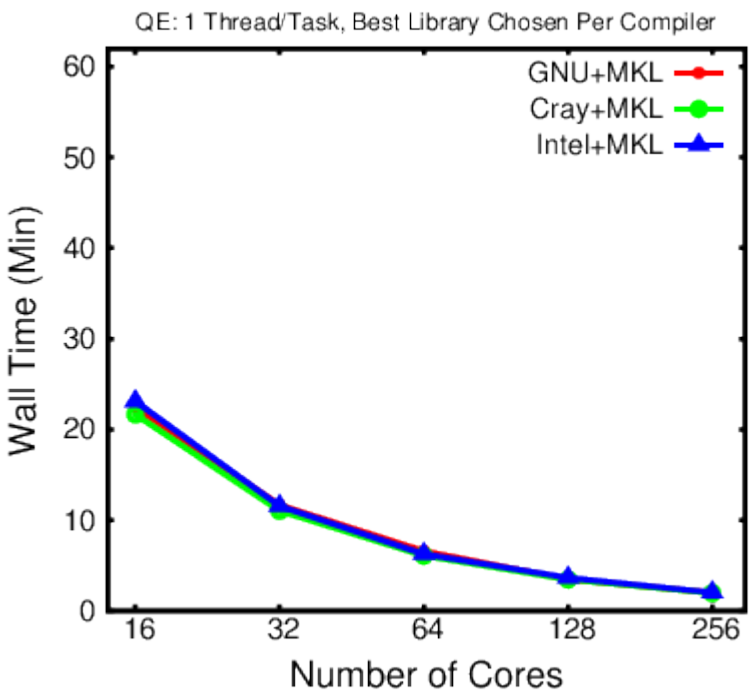
MKL beats FFTW. And MKL beats LibSci.
ZGEMM's in LibSci ~ 50% slower than MKL.
(DGEMM Gap is Smaller)

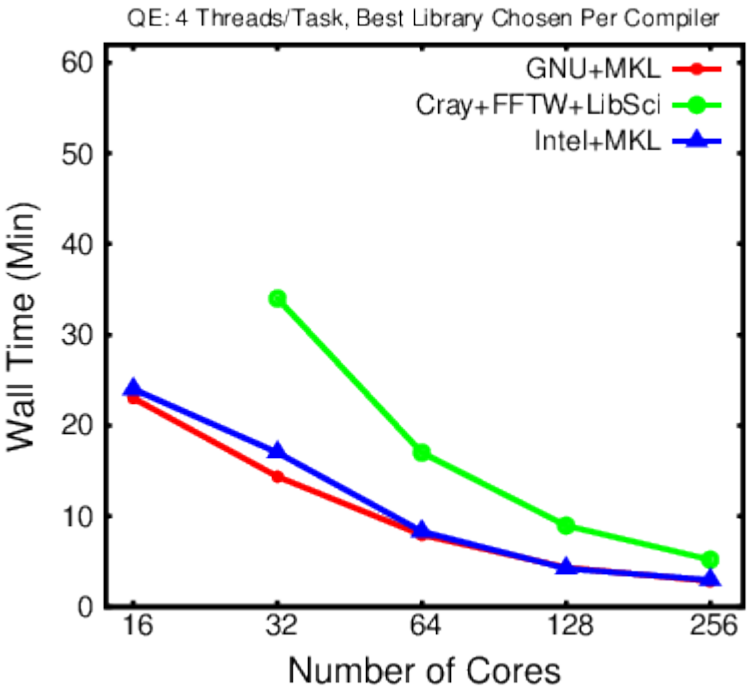
BerkeleyGW Hopper Vs. Edison



~ 3x Improvement on core per core comparison.

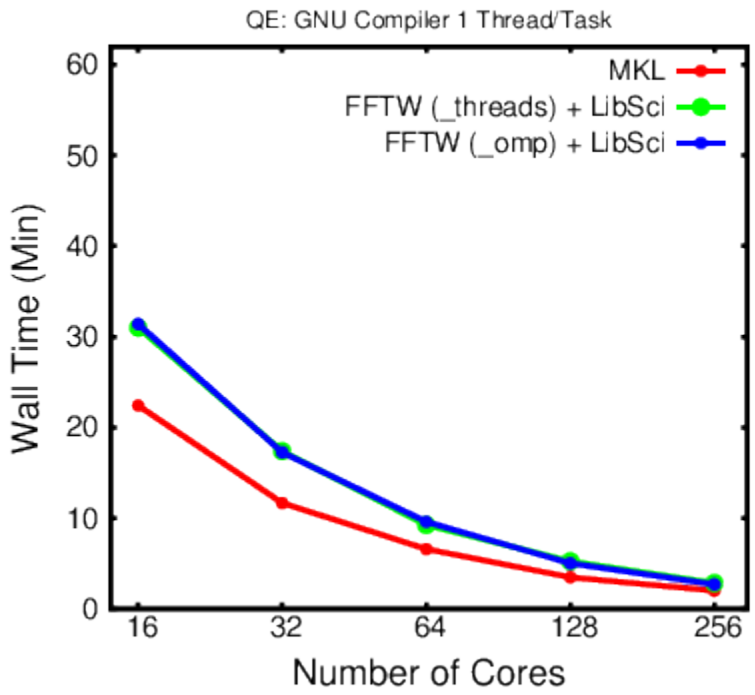
QE 5.0.2. (8,0) Single Walled Carbon Nanotube Example

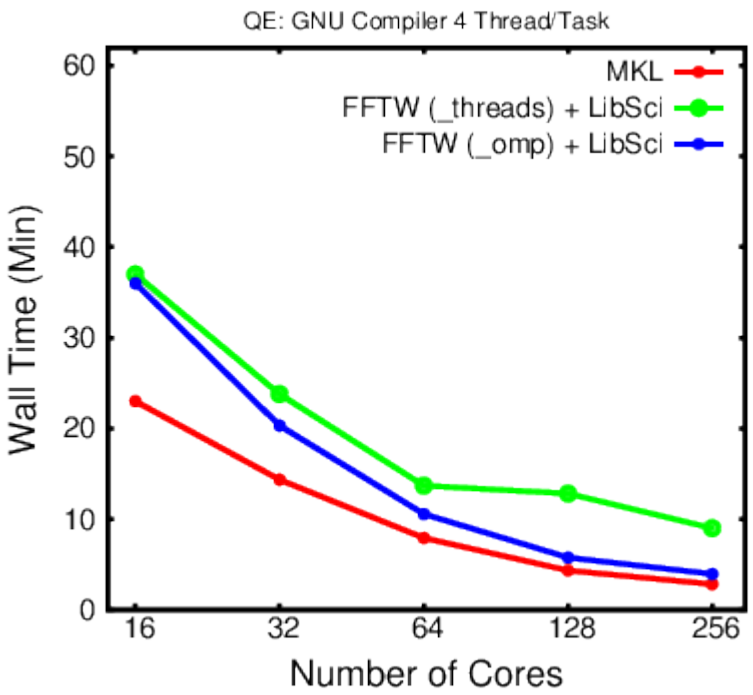




Cray + MKL fastest combination for 1 thread. GNU + MKL & Intel + MKL are the best overall combinations.

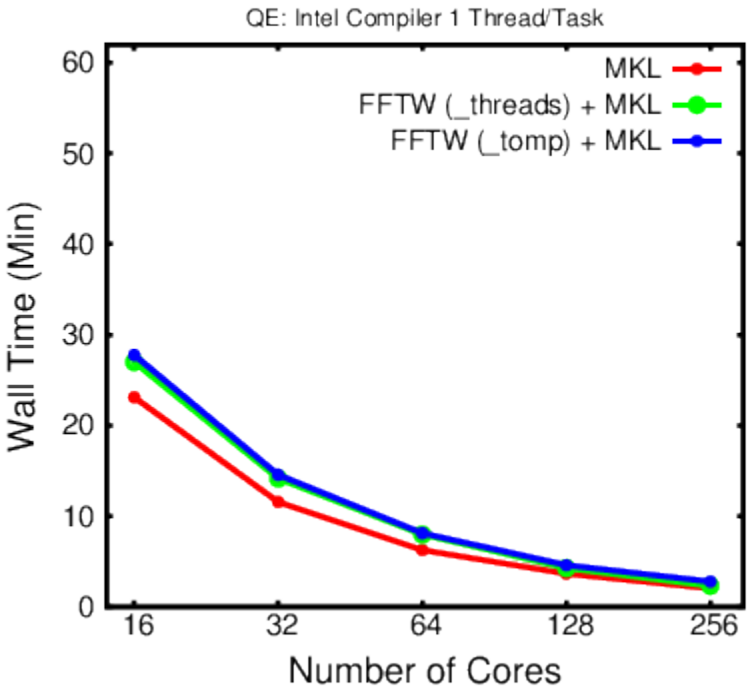
QE GNU Summary

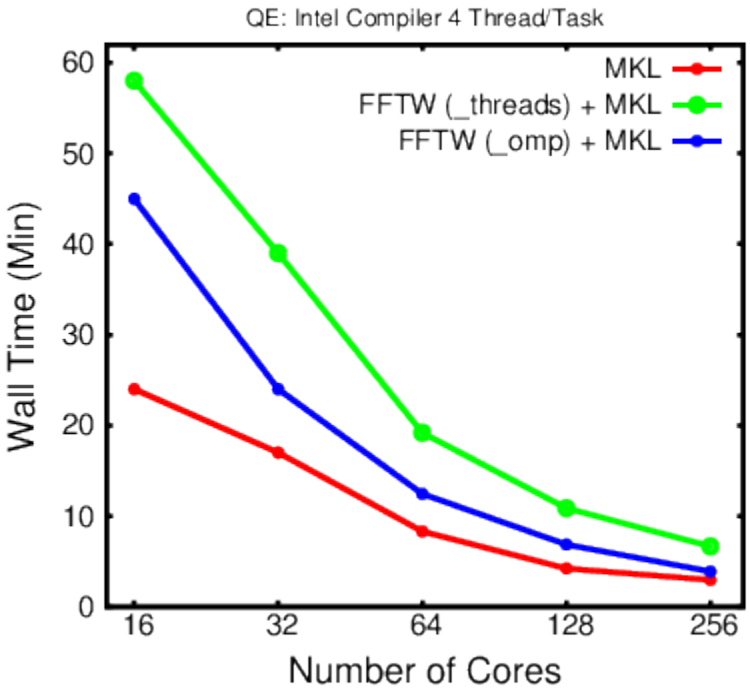




Again, MKL is Faster that FFTW+LibSci

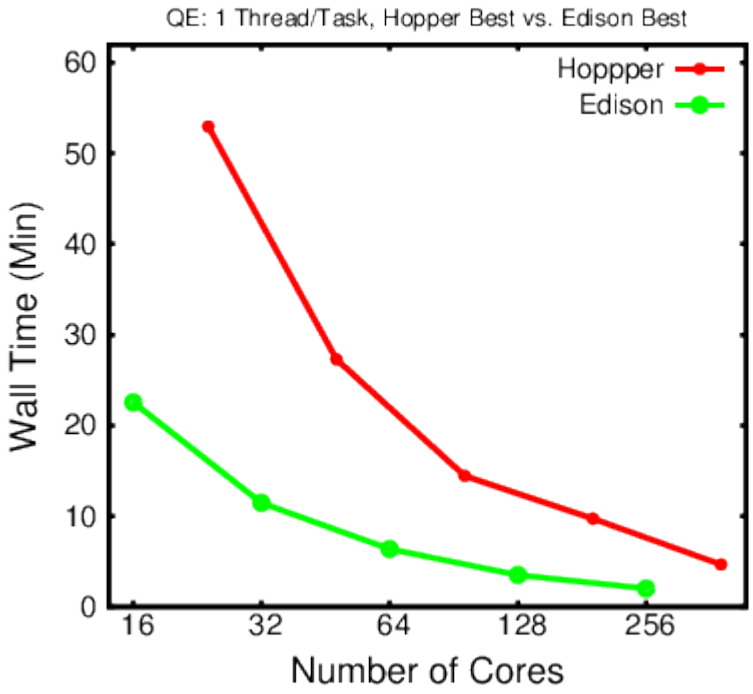
QE Intel Summary





MKL FFTs one again are superior.

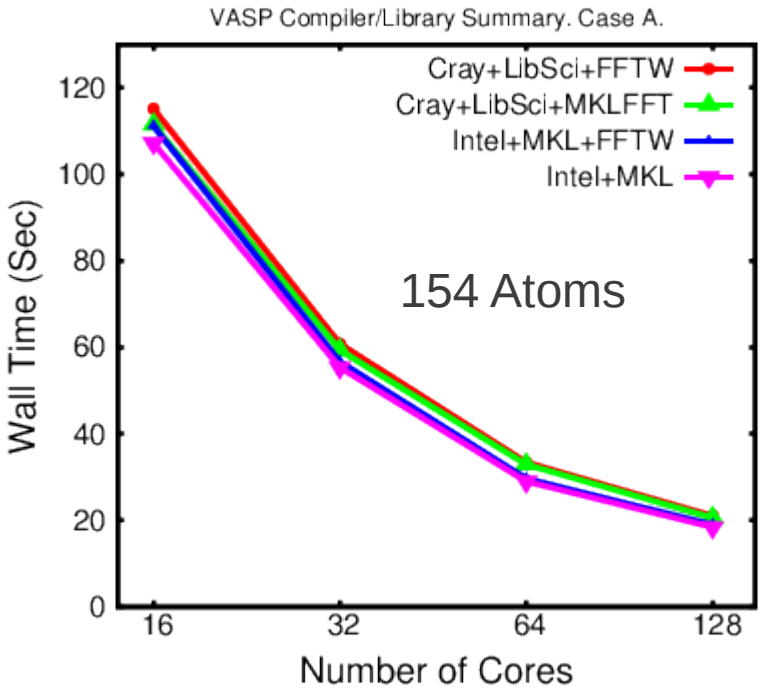
QE Hopper Vs. Edison

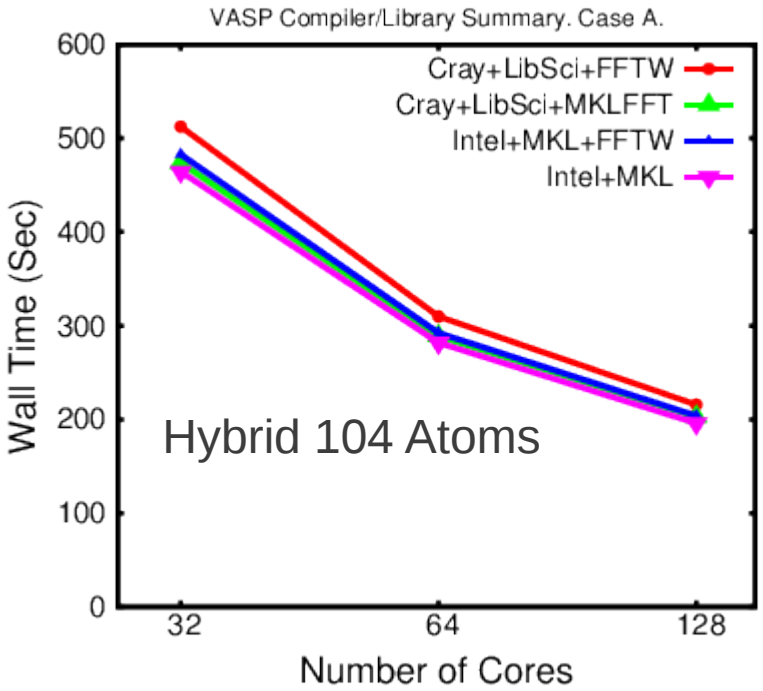


~ 3X Speedup on core-per-core comparison

VASP Summary

Version 5.3.3



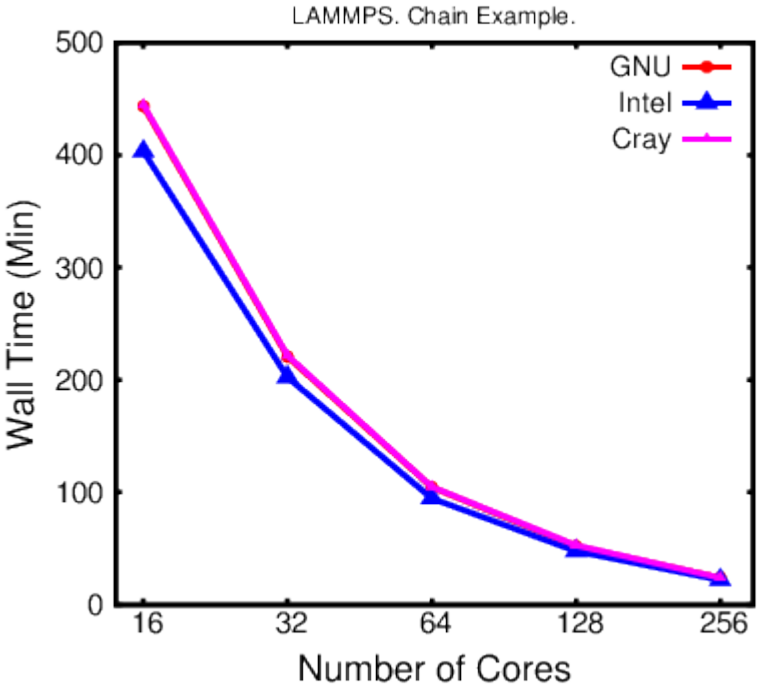


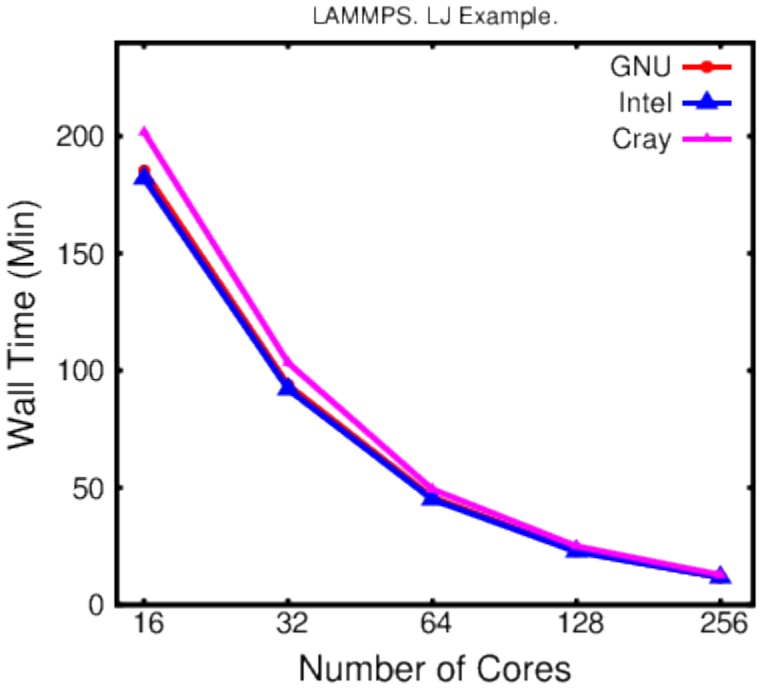
Intel + MKL again the best compiler. Cray + MKL for linear algebra yields runtime problems.

Cray + MKL Lin. Alg. Runtime Failures
GNU Builds Yield Runtime Failures

LAMMPS Summary

Version 22Mar13

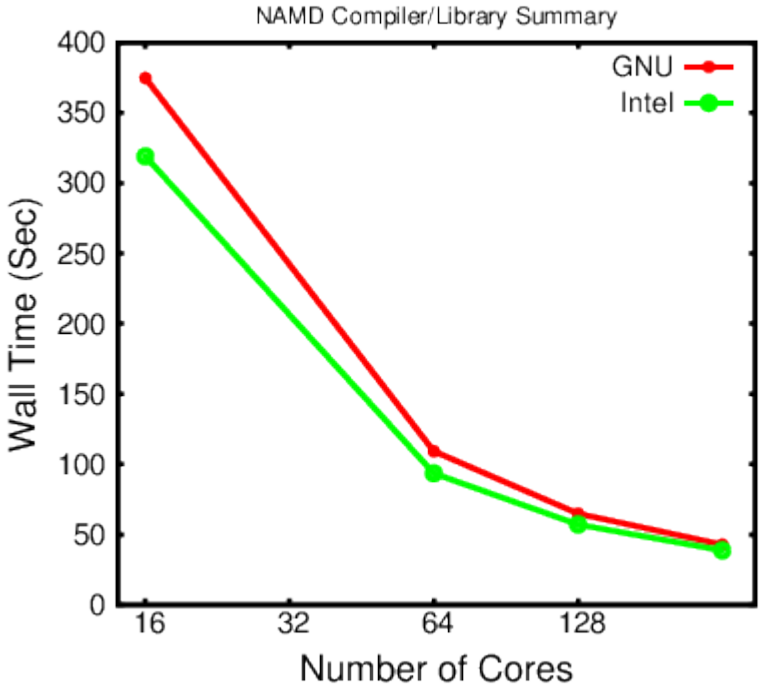




Intel and GNU compilers have the highest performance for LAMMPs.

NAMD Summary

STMV 1,066,628-atom system



Intel once again is the highest performing compiler.
Failed to produce successful build with Cray compiler.

Summary

1. In tested codes, MKL outperforms LibSci and FFTW on Edison.

2. Degraded performance likely in libraries when mixing multiple thread implementations. (Mixing pthreads w/ openmp & openmp from different compilers)

3. Intel had best overall performance on all codes. In large part due to library support and compilation success rate.

National Energy Research Scientific Computing Center