KNL Performance Comparison: OpenSBLI

March 2017

1. Compilation, Setup and Input

Compilation

The Taylor-Green vortex test case was performed on (a) a single ARCHER CPU node, and (b) a single ARCHER Intel Xeon Phi node. The OpenSBLI code was used to perform the simulation (see http://doi.org/10.1016/j.jocs.2016.11.001). This test case can be found in the *apps* directory of the OpenSBLI source code (v1.0.0 release) on GitHub: https://github.com/opensbli/opensbli

After OpenSBLI and its dependencies (including the OPS library for source-to-source translation / backend targeting) have been installed, the steps needed to run the test case are:

1. module load cray-hdf5-parallel
2. module swap PrgEnv-cray PrgEnv-intel
3. export HDF5_INSTALL_PATH=/opt/cray/pe/hdf5-parallel/1.10.0/INTEL/15.0
4. export LD_LIBRARY_PATH=/opt/cray/pe/hdf5-parallel/1.10.0/INTEL/15.0/lib:${LD_LIBRARY_PATH}
5. export LD_LIBRARY_PATH=/opt/cray/pe/hdf/1.10.0/INTEL/15.0/lib:${LD_LIBRARY_PATH}
6. python taylor_green_vortex.py
7. cd taylor_green_vortex_opsc_code
8. make taylor_green_vortex_mpi

Note that OPS and the generated test case's OPS-C code were compiled using the Intel compiler ("export OPS_COMPILER=intel") for both the CPU and KNL. On the CPU, the following flags were used in the Makefile:

-O3 -ipo -no-prec-div -restrict -fno-alias -fp-model strict -fp-model source -prec-div -prec-sqrt -DMPICH_IGNORE_CXX_SEEK

On the KNL, the following flags were used:

-xMIC-AVX512 -O3 -qopenmp -ipo -no-prec-div -restrict -fno-alias -fp-model source -fp-model precise -DMPICH_IGNORE_CXX_SEEK

Setup

On the CPU, 24 MPI processes were used. On the KNL, 64 MPI processes were used with four hyperthreads and memory option *aoe=quad_100:*

- aprun -n 24 ./taylor_green_vortex_mpi (on CPU)
- aprun -n 64 -d 4 -j 4 -cc depth (on KNL with *aoe=quad_100* in the .pbs file)

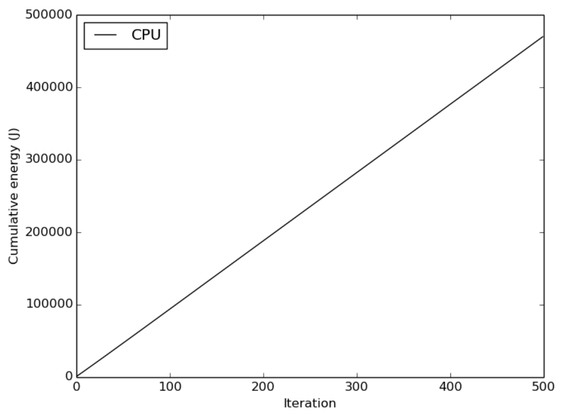
Input

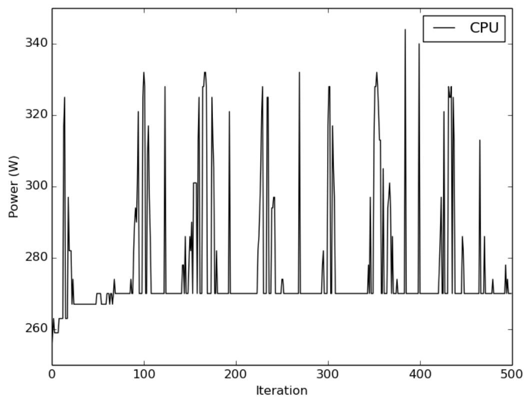
The settings were the same as in the input file here:
https://github.com/opensbli/opensbli/blob/master/apps/taylor_green_vortex/taylor_green_vortex.py
except for the number of grid points (increased from 64^3 to 256^3) and the number of timestep iterations (reduced to 500).

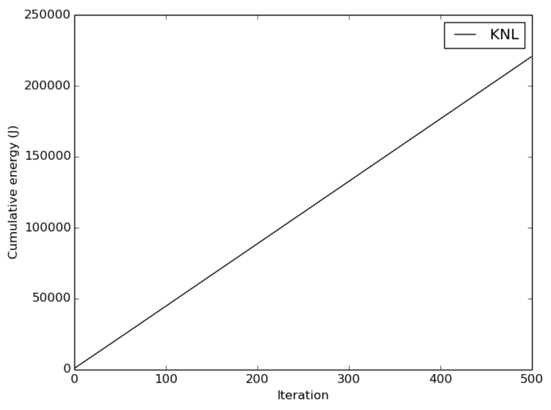
Note that the solution algorithm used here is the "BL" (baseline) algorithm described in detail by Jammy et al. (in press): http://doi.org/10.1016/j.jocs.2016.10.015

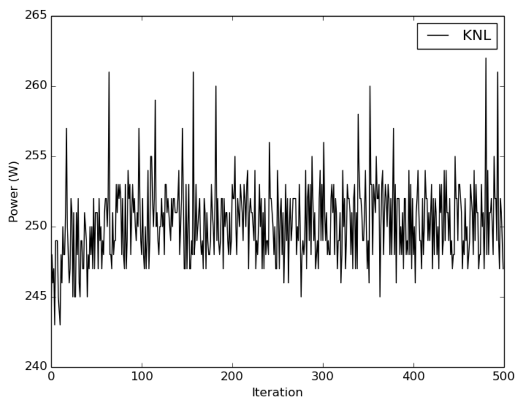
2. Performance Data

The run-time of the simulation on the CPU was 1557.2 s compared to 875.6 s on the KNL. Some initial energy efficiency results were performed using the pat_mpi_lib library: https://github.com/cresta-eu/pat_mpi_lib and show a substantial reduction in power usage and overall energy consumption with the KNL:









3. Summary and Conclusions

- It is certainly worth using a single KNL node over a single CPU node for this particular test case.

- The overall energy consumption and run-time was substantially reduced when using the KNL node. About half the amount of energy was consumed by the KNL compared to the CPU, as a result of the faster simulation run-time and reduced power usage (~250 W compared to ~270 W).

- We found that a significant decrease in run-time could be achieved using the -xAVX-512 flag during compilation on the KNL card.