KNL Performance Comparison: *CP2K*

March 2017

1. Compilation, Setup and Input

Compilation

CP2K and its supporting libraries libint, libxsmm, libgrid, and libxc were compiled following the instructions found online at:

https://github.com/ARCHER-CSE/build-instructions/blob/master/CP2K/README.md

- Revision 17772 of CP2K (February 22nd, 2017) was used

This code version can be obtained by checking out the CP2K repository as described at https://www.cp2k.org/download

Compiler and library versions used were:

Compiler/Library	ARCHER Xeon	ARCHER KNL
Intel Compiler	16.0.2.181	17.0.0.098
MKL	2016.2.181	2017.0.098
FFTW	3.3.4.5	3.3.4.10
libint	1.1.4	1.1.4
libxsmm	1.6.3	1.6.3
libgrid	in CP2K distribution	in CP2K distribution
libxc	2.2.2	2.2.2

Setup

- The CP2K PSMP (hybrid MPI+OpenMP) executable was used on both systems
- ARCHER KNL nodes were used in "quad_100" configuration (all the MCDRAM configured as an additional cache level)
- Runs on ARCHER Xeon nodes were executed with the following aprun options:
  ```
  –d $OMP_NUM_THREADS  –cc numa_node
  ```
- For runs on ARCHER Xeon nodes the Intel OpenMP runtime environment variable KMP_AFFINITY was set as follows:
  ```
  export KMP_AFFINITY=none
  ```
- Runs on ARCHER KNL nodes were executed with the following aprun options:
  ```
  –d $OMP_NUM_THREADS  –cc depth
  ```
- On both Xeon and KNL nodes the number of MPI ranks and number of OpenMP threads per rank were always chosen so that the multiple of these two numbers was equal to the number of physical cores (24 for Xeon, 64 for KNL)
- Following initial experimentation, hyperthreads were left disabled (equivalent to the aprun option –j 1) as is the default on both systems, as no significant performance benefit was observed using hyperthreading.

Input

Three benchmark calculations were run:

1. H2O-64
2. H2O-512
3. LiH-HFX

H2O-XXX Benchmarks

For both the H2O-XXX benchmarks, the following files should be copied from the CP2K distribution to a working directory on the parallel filesystem accessible to compute nodes:

```
/tests/QS/benchmark/H2O-XXX.inp
/data/POTENTIAL
/data/GTH_BASIS_SETS
```

To run an H2O-XXX benchmark on 2 ARCHER Xeon nodes with 2 OpenMP threads per MPI rank:

```
export OMP_NUM_THREADS=2
aprun -n 24 -d $OMP_NUM_THREADS -cc numa_node cp2k.psmp -i H2O-XXX.inp -o H2O-XXX.log
```

To run the same benchmark on 2 ARCHER KNL nodes also with 2 OpenMP threads per MPI rank:

```
export OMP_NUM_THREADS=2
aprun -n 64 -d $OMP_NUM_THREADS -cc depth cp2k.psmp -i H2O-XXX.inp -o H2O-XXX.log
```

The timing information can be extracted by looking at the final column output from:

```
grep "CP2K   " H2O-XXX.log
```

LiH-HFX Benchmark

For the LiH-HFX benchmark, the following files should be copied to a working directory on the parallel filesystem accessible to compute nodes:

```
/tests/QS/benchmark_HFX/LiH/*
```

The following line in input_bulk_HFX_3.inp:

```
 MAX_SCF 20
```

Should be changed to

```
 MAX_SCF 1
```

This reduces the maximum number of SCF cycles to a more convenient runtime.

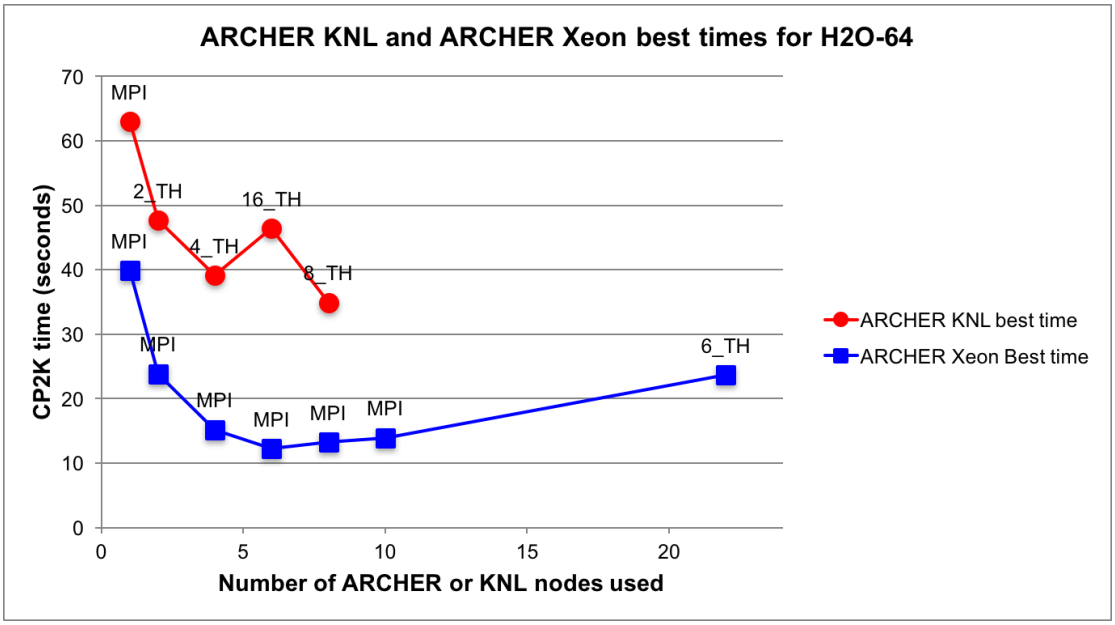
The README file describes the steps needed to run the benchmark. Aprun statements should be formed in the same way as described above for the H2O-XXX benchmarks, taking into account the appropriate –cc option depending on the architecture. Timings can be extracted as above but referring to the final log file that is output.

2. Performance Data

Figures and their constitutive data are provided in this section. Times shown are for those threading configurations that give the best performance in each case. The CP2K shorthand for naming threading configurations is:

- MPI: pure MPI
- X_TH: X OpenMP threads per MPI rank

H2O-64



Nodes	Cores used	Best time (s)	Threading Configuration
1	64	62.993	MPI
2	128	47.735	2_TH
4	256	39.095	4_TH
6	384	46.458	16_TH
8	512	34.935	8_TH

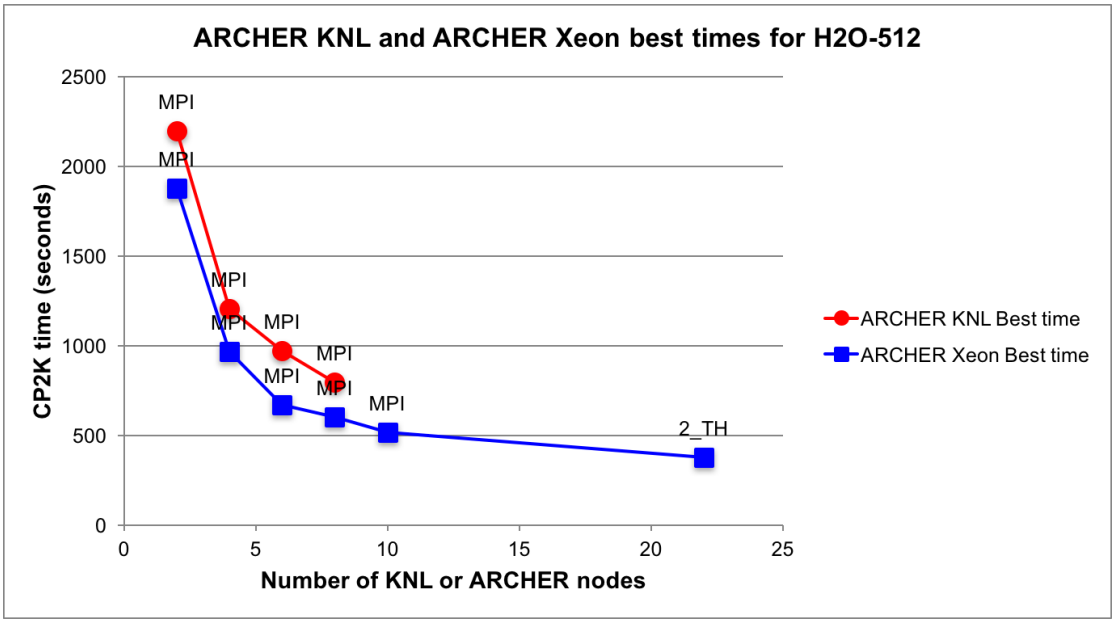
Table 1: best times for H2O-64 on ARCHER KNL

Nodes	Cores used	Best time (s)	Threading Configuration
1	24	39.86	MPI
2	48	23.878	MPI
4	96	15.202	MPI
6	144	12.205	MPI
8	192	13.227	MPI
10	240	13.953	MPI
22	528	23.709	6_TH

Table 2: best times for H2O-64 on ARCHER Xeon

Full performance data is included in tab-delimited file H2O-64.txt

H2O-512



Nodes	Cores used	Best time (s)	Threading Configuration
2	128	2197.655	MPI
4	256	1204.649	MPI
6	384	972.262	MPI
8	512	796.237	MPI

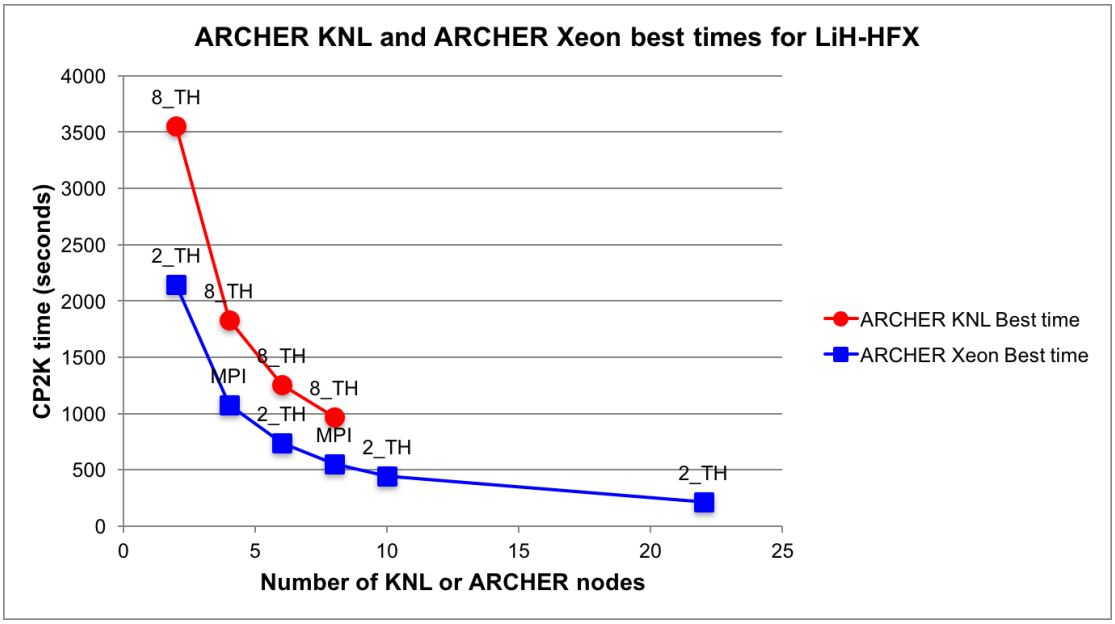
Table 3: best times for H2O-512 on ARCHER KNL

Nodes	Cores used	Best time (s)	Threading Configuration
2	48	1877.049	MPI
4	96	965.1	MPI
6	144	670.713	MPI
8	192	602.771	MPI
10	240	514.817	MPI
22	528	375.655	2_TH

Table 4: best times for H2O-512 on ARCHER Xeon

Full performance data is included in tab-delimited file H2O-512.txt

LiH-HFX



Nodes	Cores used	Best time (s)	Threading Configuration
2	128	3552.465	8_TH
4	256	1830.474	8_TH
6	384	1255.725	8_TH
8	512	967.011	8_TH

Table 5: best times for LiH-HFX on ARCHER KNL

Nodes	Cores used	Best time	Config
2	48	2143.466	2_TH
4	96	1076.069	MPI
6	144	738.721	2_TH
8	192	552.808	MPI
10	240	447.037	2_TH
22	528	212.659	2_TH

Table 6: best times for LiH-HFX on ARCHER KNL

Full performance data is included in tab-delimited file LiH-HFX.txt

3. Summary and Conclusions

- CP2K is slower on ARCHER KNL nodes than on equal numbers of ARCHER Xeon nodes
    - H2O-64 benchmark: KNL nodes on average 2.5x slower
    - H2O-512 benchmark: KNL nodes on average 1.3x slower
    - LiH-HFX benchmark: KNL nodes on average 1.7x slower

- Single threaded pure MPI is often fastest,
    - On KNL multithreading is more likely to be beneficial, especially in problems such as the LiH-HFX benchmark, in which having fewer MPI ranks means more memory is available to each rank, allowing partial integrals to be stored in memory instead of expensively recomputed on the fly.

- Hyperthreading does not significantly aid performance

- On the basis of these benchmark results it does not appear to be worth using KNL instead of Xeon