KNL Performance Comparison: CRYSTAL

March 2017

1. Compilation, Setup and Input

Compilation

The code used was the CRYSTAL17 TD-DFT development branch from October 2016. The Intel compiler v16.0.2.181 was used on Archer and v17.0.0.098 on the Xeon Phi.

Setup

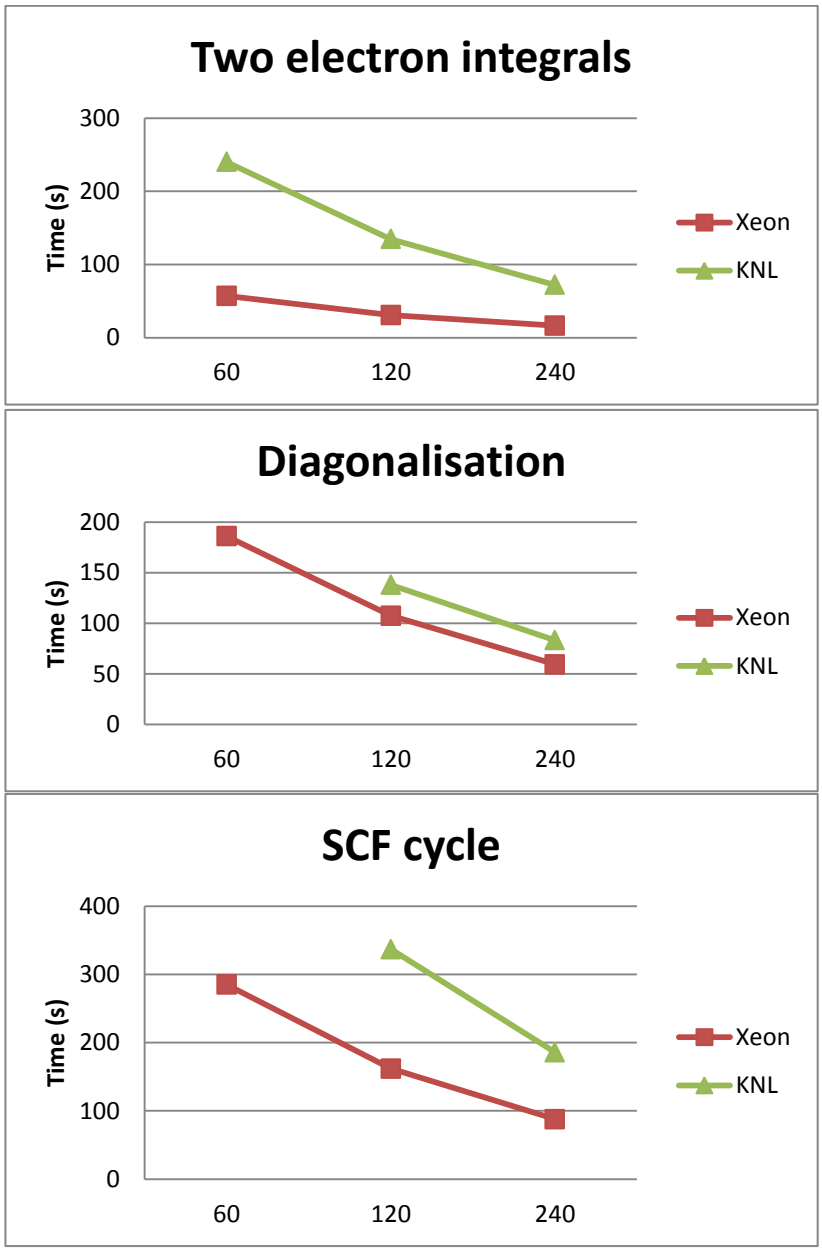
The nodes were used in quad_100 mode with all MCDRAM used as cache.

Input

The test case was a calculation on a 4x4x4 super-cell of rutile (TiO2) with a reciprocal space sampling of 3, a DFT LGRID and 8064 basis functions. The test was run for 2 SCF cycles with timing data taken from cycle 1 in the output.

2. Performance Data

This test was run to gain some initial insight into any issues compiling and running CRYSTAL on the Xeon Phi and to provide information on possible future work. It is not a complete performance analysis.
MPPcrystal is most efficient when the k-point diagonalisation is done in parallel. For the test case chosen this required multiples of 60 cores. This was mapped to 1, 2, and 4 Xeon Phi nodes with only 60 of the 64 cores used, and 3, 5 and 10 nodes of Archer where the 3 node case only used 20 of the 24 cores on each node. The 1 node Xeon Phi case failed to run due to the limited amount of memory on the test system.



The Scalapack calls in the diagonalisation routine perform almost as well on the Xeon Phi as on Archer, but this is not true of the rest of the code.

3. Summary and Conclusions

On the same number of cores the using Xeons will be a better choice than Xeon Phi with the current MPI version of CRYSTAL. It would be necessary for your science problem to scale to at least twice the number of Xeon cores you have available for the Xeon Phi option to become competitive (and probably more than twice for a modern Xeon v4).