Section 3.3 Intel compiler options.
The compiler options discussed in the text don't match those listed in (1). For example, -i8 and -r8 are discussed but not seen in list given in (1).

For clarity the first sentence in section 3.3

> The UM sources on Sandy Bridge systems Raijin and Ngamai were compiled with the following Intel compiler options

was replaced with

> The Fortran and C sources of the UM were compiled on Sandy Bridge systems Raijin and Ngamai with the following Intel compiler options

Also the test states that -xavx is used instead of -xHost. However, -xHost tells the compiler to "generate instructions for the highest instruction set available on the compilation host processor." So it should select -xavx if possible. My concern is that the treatment of -xHost is factually incorrect.

Addressing this comment an old version of

> During the porting stage of our executables from the old Solar Nehalem chip based system to new Sandy Bridge Ngamai and Raijin systems in order of getting compatibility of executables across these machines the `-xHost` compiler option used on Solar was replaced with the compilation flag `-xavx` for advanced vector extensions supporting up to 256-bit vector data and available for Intel Xeon Processor E5 family. Also it has been found that adding the `-xHost` compiler option to the set of compilation options (1) does not make any impact on either the numerical results or the model performance.

was replaced with

> Code generated by the compiler option `-xHost` targets the processor type on which it is compiled. On the old Solar Nehalem chip based system, this was equivalent to compiling with `-xsse4.2`. During the porting stage of our executables to the new Sandy Bridge Ngamai and Raijin systems, in order to ensure compatibility of executables across these machines, the `-xHost` compiler option used on Solar was replaced with the compilation flag `-xavx` for advanced vector extensions supporting up to 256-bit vector data and available for Intel Xeon Processor E5 family. It was confirmed, as expected, that adding the `-xHost` compiler option to the set of compilation options (1) did not have any impact on either the numerical results or the model performance.

The correctness of the latest text version has been checked with Martyn Coden (Intel).

It would be nice to see a stronger examination of the role of turbo boost. For example, it should be possible to re-scale the performance on raijin to the level which would have been seen without turbo boost.

We do agree that it would be nice to understand the benefits of turbo boost usage on the application performance improvements. Unfortunately our enquiries to the NCI support staff indicate that it is not possible to test our applications without turbo boost.

Lastly, it would be nice to see some form of explanation as to why the benefits of partially committed nodes are only seen for high core counts. This is especially true for the N512L70 model. I would have assumed that the pressure on memory bandwidth would be the least at high core counts due to the finer grained domain decomposition. However, I appreciate that that is a nice-to-have.

We agree it would be nice to have a complete explanation of the benefits of partially committed nodes on large numbers of cores on Raijin. This is being investigated using newly installed software. However it will be some time before we could confidently explain the behaviour on high core counts. As a result we consider this “nice-to-have” recommendation to be beyond the scope of the paper at this time.