Referee #1

Page 7396 line 9 : The paper mentions about "Shortest elapsed times" and then again about finding the "most efficient run configurations" at line 2 , page 7397. Which is more important?

Addressing this comment and other comments provided for the Introduction section, several parts of this section have been modified. The new version of the full Introduction section is the following:

1 Introduction

The Unified Model (UM) numerical modelling system (Brown et al., 2012) is used for short and medium range weather forecasting, for both high resolution weather modelling and for relatively coarser climate modelling. Such modelling software requires relatively powerful High Performance Computing (HPC) systems to support operational forecast production. Typically the computing systems have a peak performance comparable to the computer systems included in the TOP500 list released every 6 months. Since September 2009 the UM has been used in the Numerical Weather Prediction (NWP) component of the Australian Community Climate and Earth System Simulator (ACCESS; Puri et al., 2010) at the Australian Bureau of Meteorology (BoM).

Current operational systems at BoM are based on the UM version 7.5 (vn7.5) and the next operational systems upgrade will be based on UM vn8.2. The latter version therefore was used for the work described here.

UM versions include both science and performance upgrades, and extensive evaluation of both types of changes in development mode is required prior to operational implementation. In addition, changes to these systems have major consequences for many downstream applications. For these reasons changing UM versions for operational systems is only done every one to two years at the BoM.

Leading HPC systems have from tens of thousands to several million very powerful cores. Since 1980 the trend in HPC development has been for the available processor performance to increase at a greater rate than the available memory bandwidth (Graham et al., 2005, pp.106–108). The authors concluded that a growing gap between processor and memory performance could become a serious constraint in performance scaling for memory bound applications. The gap between processor performance and memory bandwidth has been growing especially quickly for multicore processors in the past 10 years (Wellein et al., 2012). This gap forces the cores on a node to compete for the same memory causing resource contention, which can become a major problem for memory intensive applications such as the UM.

Increasing the resolution of numerical models is one of the key approaches to improving forecast accuracy. However, in an operational setting these models are constrained to run within a fixed elapsed time on available computing resources.

> Increasing resolution requires increased computation and therefore the performance efficiency (from herein just referred to as efficiency) as measured by the run time on a given number of cores.
>
> Finding the most efficient usage of the system for a particular application and the shortest elapsed times varies depending on whether the application is run on all node cores (fully committed case) or on a subset of the cores available on each node (partially committed case). The placement of the threads and/or Message Passing Interface (MPI) processes across partially committed nodes (sockets) also needs to be done carefully taking into consideration all shared resources available to these cores.
>
> Another practical aspect of the performance analysis discussed in the paper is to estimate whether the coming upgrade for the BoM's operational models will be feasible given the operational time windows and available HPC resources.
>
> The performance analysis described here shows that on some modern HPC systems the shortest run times can be achieved with the usage of partially committed nodes. The concept of using partial nodes for UM applications allows reduced resource contention and improves application performance (Bermous et al., 2013).

having

> . . . the performance efficiency (from herein just referred to as efficiency) as measured by the run time on a given number of cores.
> Finding the most efficient usage of the system for a particular application and the shortest elapsed times varies depending on whether the application is run on all node cores (fully committed case) or on a subset of the cores available on each node (partially committed case).

to address the above mentioned referee comment.

Page 7396 line 15: "memory bandwidth intensive applications" – Suggestion is that Memory intensive or memory bandwidth limited applications will be better suited.

The original text

> memory bandwidth intensive applications

was replaced with

> memory intensive applications

in the modified version of the Introduction section.

Similar references in the paper (page 7397, line 22; page 7405, line 4; page 7409, line 2) have been corrected in the same way.

Page 7396 line 23: “relatively powerful HPC systems” - Comparison to other HPC systems will be helpful.

The comment was addressed in the updated version of the Introduction section with the following text

> Such modelling software requires relatively powerful High Performance Computing (HPC) systems to support operational forecast production. Typically the computing systems have a peak performance comparable to the computer systems included in the TOP500 list released every 6 months.

Page 7396 line 24: numerical weather prediction – The first letters should be capitalized.

The recommended change was made in the updated version of the Introduction section

> Since September 2009 the UM has been used in the Numerical Weather Prediction (NWP) ...

Page 7397 line 9: “asynchronous computation and I/O “should be “computation and asynchronous I/O” as only the I/O is asynchronous

The latest version of the Introduction section does not include this text.

Page 7397 line 12: “these are not expected to significantly impact on the major results of this paper.” is an assumption which can be avoided.

The latest version of the Introduction section does not include this text.

Page 7399 line 11: It is not clear if it is 3 day simulation or 3 model day simulation.

For clarity the pointed out text has been changed from

> 3 day simulation

to

> 3 model day simulation

Page 7400 line 1: First use of 3D-VAR and has not been described before.

Reference to the 3D-VAR has been removed:

> The experimental system was created in 2012 and it is currently running 24 times per day.

Page 7400 line 11: It is not clear if 18 GB of data produced per run or per model day?

For clarity the recommended change was made:

> The I/O in the job producing only 18 GB per run of the output data is relatively small in comparison to the size of I/O in the global model job.

Page 7401 line 5: "reduced by over 2 times" should be reworded.

The original version of

> It should be noted that this ratio has been reduced by over 2 times on the latest Ngamai and Raijin systems in comparison with Solar.

was changed to

> For the newer Ngamai and Raijin systems this ratio is less than half of that for Solar.

Page 7401 line 10: The more advanced optimizations are not stated.

The first sentence in section 3.2 **(page 7401, lines 7-10)**

> The HPC systems described in Table 1 support both Intel and GNU compilers; however, extensive testing has shown that the Intel compiler provides more advanced optimisation.

was removed and two paragraphs following the removed sentence were merged:

> With the UM vn8.2 sources separate executables were required for each model type: global and limited-area. The Intel compiler version 12.1.8.273 was used to produce UKV executables. In order for the global N512L70 system to use the UM async I/O features the Intel 14.0.1.106 compiler release was needed to avoid crashes due to interaction between the older compiler and the new I/O code.

Previous studies by the NCI staff showed that Intel compiler provides faster executables. Determining why one compiler produces faster executables than another is beyond the scope of this paper.

Page 7403 line 25: Figure 1 and 2 shows only the elapsed times and not the efficiency.

Addressing this comment two sentences (on page 7403, lines 23-26)

> Comparison of the best elapsed times produced by running the UKV model on Raijin and Ngamai is given in Figs. 1 and 2 correspondingly. On each figure the best elapsed times are provided for comparison between the efficiency using pure MPI vs. MPI/OpenMP hybrid parallelism.

were replaced with

> Comparison of the best elapsed times produced by running the UKV model with the usage of pure MPI and MPI/OpenMP hybrid parallelism on Raijin and Ngamai is given in Figs 1 and 2 correspondingly.

The term "efficiency" in the paper is used in a way to find on whether pure MPI or hybrid parallelism, fully committed or partially committed nodes provide the shortest run times for a specified number of cores. The usage of the term "efficiency" is different from the "parallel efficiency" metric used in the performance analysis. In this regard the meaning of term "efficiency" used in the paper has been included in the modified version of the Introduction section.

Page 7404 line 1: The author describes using partially committed nodes by using 12 cores from 16 cores of Raijin (with OPENMP). Why is using only 8 cores from 16 cores not explored? And similarly for Ngamai why is 6 cores from 12 not explored.

Addressing the comment the following text was included in the paper:

> On Raijin the usage of 8 out of 16 cores with hybrid parallelism and 2 OpenMP threads showed between 16.6% (for high core counts) to 31.0% (for low core counts) slower run times in the 768-3072 reserved core range in comparison with the corresponding results obtained using pure MPI. On Ngamai the usage of half committed (6 cores from 12 cores) nodes with the hybrid parallelism gives two possible configurations for running an application. Firstly there is the symmetrical case with 1 MPI process and 3 OpenMP threads running on each socket. Another option is the non-symmetrical case with 2 MPI processes running on one socket and 1 MPI process running on another socket with 2 threads per each MPI process. Taking into account the limited OpenMP coverage in the UM vn8.2 sources, tests using hybrid parallelism with a symmetrical case of 3 threads or an asymmetrical case with 2 threads on half committed nodes were not performed on Ngamai.

after line 6 on page 7405.

With the above mentioned addition to the text, the last paragraph in Section 4.2 (**page 7408, lines 16-26**) has been changed to:

> Performance results for a 6 cores-per-node case are not presented in Fig.10. As discussed at the end of Sect. 4.1.2 there are two possible run configurations for this case. With symmetrical usage of 3 threads per MPI process run on each socket, the model performance was even worse in comparison with the fully committed node case. At the same time the non-symmetrical usage with 3 MPI processes and 2 threads gave similar performance results as in the 8 cores-per-node case.

to avoid repetition.

Page 7405 line 19: The author should describe the options orte_num_sockets and orte_num_cores used.

The original version of

> For example, running the model with pure MPI on Raijin and using 12 cores per node the following options
>
> ```
> mpirun -npersocket 6 -mca orte_num_sockets 2
>                      -mca orte_num_cores 8 ...
> ```
>
> were used in the mpirun command with OpenMPI 1.6.5. The last two options were required to specify to avoid the related bugs found in the OpenMPI software.

was changed to

> In the example of running the model with pure MPI on Raijin and using 12 cores per node the following options
>
> ```
> mpirun --npersocket 6 –mca orte_num_sockets 2 \
>
>                       –mca orte_num_cores 8 . . .
> ```
>
> were used in the mpirun command with OpenMPI 1.6.5. The last two options specify the number of sockets on a node and the number of cores on each socket. These options were required to avoid bugs found in the OpenMPI 1.6.5 software.

with the description of the options referred in the comment.

Page 7406 line 1: The two efficiency measures are not described in the paper and are not shown in Fig 3 or Fig 4.

The corresponding sentence (page 7405, line 29 – page 7406, line 2)

> Note that the ratio of these two efficiency measures continues to increase with an increasing number of reserved cores.

has been removed.

Page 7406 line 23: The author does not describe if and why the number of sub domains in the East –West direction is limited to only 36.

The original statement in the paper (page 7406 lines 23-25) of

> So a 36×48 decomposition of 1728 cores represents near the maximum number of cores which can be used to achieve the best performance for the application.

was replaced with

> The shortest run times using fully committed nodes on Ngamai and Raijin with the usage of up to 1728 cores were achieved on a decomposition of 36x48 with pure MPI. The largest allowable decomposition under the constraints provided in Sect. 2.2.1 is 42x48 on 2016 cores. Increasing the number of cores from 1728 to 2016 provides further improvements in the elapsed times of 1.7% on Ngamai and 2.2% on Raijin. This indicates that the corresponding performance scaling have not begun to level off at the largest allowed decomposition of 42x48.

Also as per a comment provided by referee #2, a new section "2.2.1 UKV decomposition constraints" has been added to the paper. This section includes information on the maximum allowed decomposition sizes for the UKV model.

Page 7408 line 15: The turbo boost or its usage is not described by the author.

The following text was added at the end of section 3.1 (page 7401, line 6)

> It should be noted that turbo boost was enabled in Basic Input/Output System (BIOS) on Raijin only. As per Intel turbo boost 2.0 technology processor can run at above its base operating frequency which is provided in Table 1 ("Node processor cores" line). Having idle cores on processor, power that would have consumed by these idle cores can be redirected to the utilised cores allowing them to run at higher frequencies. Based on the turbo boost additional multipliers a Base Clock Rate (BCLK) can be calculated. For example, we have
>
> BCLK=(3.3GHz–2.6GHz)/7=100MHz
>
> and utilised cores operate at 2.6GHz + 5*BCLK = 3.1GHz if an application is run on 6 cores from 8 cores available on each Raijin processor.

As a result of this change the following text

> (derived from information in Table 1)

was removed on page 7405, line 6.

Also an additional change was made in Table 1, where the following line

Usage of turbo boost	No	No	Yes

was replaced with

Turbo boost	OFF	OFF	ON

Page 7409 line 26: "Couple of hundred cores" is vague and exact number should be mentioned.

Addressing a comment provided by referee #2, the last paragraph in the Conclusion section (page 7409, line 24 – page 7410, line 4)

> The usage of partially committed nodes has been successfully employed to improve run times for other two tasks run on Raijin, including a coupled climate model running at a relatively low resolution of N96. Adding a couple of hundred cores gave a significant reduction in elapsed time of over 20%. This level of improvement is very important for climate modelling experiments that usually require many months of elapsed time. A second application was a UK Met Office four-dimensional variational analysis system at a resolution of N320L70. The usage of partially committed nodes gave up to a ten-fold improvement in performance scaling for used core counts in the range between 500-1500.

containing the referred text has been removed.

Page 7415 table 3: Table 3 does not have information about the number of cores used per node.

Table 3 does not have information about the number of cores used per node as per the corresponding statement in the text (for details please see page 7408, lines 3-4) "…the most efficient usage is achieved using fully committed nodes" and according to the information provided in Table 1, a Ngamai node has 12 cores. Also there are references in the text preceding this section stating a number of cores on Ngamai node, for example on page 7404, line 8. As a result an additional column having "12" in all rows was not included in Table 3.

Addressing the comment the title of Table 3 has been changed from

> The best elapsed times for N512L70 on Ngamai using 2 threads.

to

> The best elapsed times for N512L70 on Ngamai using 2 threads on fully committed nodes.

There are some technical details which are not described and some assumptions which are not clearly stated.

It was not clear what are the missing technical details or unstated assumptions.

Referee #2

For me the key finding is that–on Raijin, but not on Ngaimai and Solar– both the regional and global models run faster when only partially committed nodes are used.

In relation to this comment the following text

> A relatively poor UKV performance on partial nodes especially on Ngamai system in comparison with Raijin was due to unavailability of turbo boost on that system. Turbo boost would allow active cores to run at up to 16% higher clock speeds with the usage of 8 cores-per-node on Ngamai.

has been added on **page 7406, line 16**.

I believe that the paper would be clearer and easier to read is the record of the method and the empirical findings were more clearly separated. For example, the nature of the domain decomposition constraint for the UKV model presented in section 4.1 wasn't immediately clear to me.

Explanation addressing decomposition constraints was included in a new "2.2.1 UKV decomposition constraints" section with the following content

> ### 2.1.1 UKV decomposition constraints
>
> The MPI decomposition in the Unified Model is based on horizontal domain decomposition where each subdomain (MPI process) includes a full set of vertical levels. Within the computational core of the UM, OpenMP is generally used to parallelise loops over vertical dimension.
>
> Due to semi-Lagrangian dynamics implementation, the halo size for each sub-domain limits the maximum number of sub-domains in each direction. With the halo size of 10 grid points used in this study and the horizontal grid size of 648x720, the corresponding limits for the MPI decomposition sizes were 42 in the West-East direction and 48 in the South-North direction. Another constraint in the UM model implementation is that the decomposition size in the West-East direction must be an even number.

At the same time the following paragraph

> The MPI decomposition in the Unified Model is based on horizontal domain decomposition where each subdomain (MPI process) includes a full set of vertical

> levels. Within the computational core of the UM, OpenMP is generally used to parallelise loops over vertical dimension.

moved into the new section has been removed from Section 4 (**page 7403, lines 8-11**)

I believe that the graphs using 'Number of used cores' on the x-axis are not helpful and detract from the core message. A stated aim for the work was to find the most efficient model configuration with regard to computational resources. If cores are left idle, they should still be accounted for in a measure of efficiency and so the graphs using 'Number of reserved cores' on the x-axis are, for me, the right ones to use.

Addressing the comment the following text

> The performance relative to the number of reserved cores is the most important metric, however performance relative to the "Number of used cores" provides additional information on the value of reducing the number of active cores per node. This extra information is particularly relevant to circumstances where the elapsed time is more important than using nodes as efficiently as possible. Examples include climate runs and cases where other restrictions mean that the number of nodes available is not a significant constraint on an application's run time. The related performance information cannot be easily seen on the graph using the "Number of reserved cores" metric.

was included on page 7405, line 11.

If it were possible, some empirical measures of memory bandwidth (perhaps offered by PAPI calls?) would be very interesting and would bolster the key findings.

We agree, but neither PAPI library nor Intel Vtune software is available on all systems.

I believe that the comments regarding 4D-VAR and other N96 resolution model should be removed from the conclusion. The reason for this is that the conclusion summaries points previously examined in the paper, and these codes were not discussed anywhere else.

The last paragraph in the Conclusion section has been removed. At the same time some minor changes and additions have been made in this section. So the latest version of the full section is the following:

5 Conclusions

With a trend in the HPC industry of decreasing the Byte/Flop ratio especially on multicore processors and increasing the number of cores per CPU, the most efficient system usage by memory intensive applications can be achieved with the usage of partially committed nodes. In other words, the following factors such as increasing

memory bandwidth per active core, reduction in the communication time using less MPI processes and active cores running at higher clock speeds with turbo boost can more than compensate for the reduced number of cores in action. This approach can improve an application performance and most importantly the application performance scaling. A conclusion on whether a specific application should be running on fully or partially committed nodes depends on the application itself as well as on the base operating frequency of the processor and memory bandwidth available per core. Other factors such as availability of turbo boost, hyper-threading and type of node interconnect on the system can also influence the best choice. This study showed that both the regional and global models can run faster if partially committed nodes are used on Raijin. At the same time taking into an account the similarities between Raijin and Ngamai systems, there is a reasonable expectation that a similar affect would have been achieved on Ngamai if turbo boost would be available on this system.

The usage of partially committed nodes can further reduce elapsed times for an application when the corresponding performance scaling curve has flattened.

Another example when the use of partially committed nodes can reduce run times is when the performance scaling has not flattened but the number of used cores cannot be increased due to other constraints in the application. This case was illustrated by the UKV model example in Sect. 4.1.2. This approach can be used for climate models based on the UM sources and run at a relatively low horizontal resolution. As per the results of this Sect. 4.1.2 the usage of partial nodes can reduce elapsed times significantly. This has a very important practical value for climate research experiments that require many months to complete.

The approach of using partially committed nodes for memory bandwidth-bound applications can have a significant practical value for efficient HPC system usage. In addition, this can also ensure the lowest elapsed times for production runs of time critical systems. This approach is a very quick method for providing major performance improvements. In contrast, achieving similar improvements through code related optimisation can be very time consuming and may not even be as productive.

Specific comments ==========

p7397 l21: Would 'resource contention' be better than 'memory contention'? Since it is the bandwidth to memory rather than the use of particular memory addresses that is in competition.

The recommended change was made in the new version of the Introduction section.

p7402 l4: Please explain why removing -xHOST ensured reproducibility of results across clusters.

Addressing the comment Section 3.3 was rewritten with more details provided on the compilation options used to build executables:

3.3 Intel compiler options

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

```
-g -traceback -xavx -O3 -fp-model precise      (1)
```

Option `-O3` specifies the highest optimisation level with the Intel compiler. Combination of "`-g -traceback`" options was required in order to get information on a failed subroutine call sequence in case of a run time crash problem. The usage of these two options had no impact on the application performance. Bit reproducibility of the numerical results on a rerun is a critical requirement for the BoM operational systems. For this purpose compilation flag "`-fp-model precise`" (Corden et al., 2012) was used in spite of causing between 5% to 10% penalty in the UM model performance. An additional pair of compilation options "`-i8 -r8`" was used to compile Fortran sources of the UM. These options make integer, logical, real and complex variables 8 bytes long. Option `-openmp` was specified to compile all model sources and to link the corresponding object files to produce executables used for MPI/OpenMP hybrid parallelism.

Due to a very limited capacity for non-operational jobs on the BoM operational system, Ngamai, the testing and evaluation of the forecast systems prior their operational implementation is predominantly performed on the relatively larger Raijin system. The BoM share on Raijin is 18.9%. 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. Compatibility of binaries across systems was achieved by having the same Intel compiler revisions and OpenMPI library versions on the both Ngamai and Raijin systems as well as system libraries dynamically linked to the executables at run time. Note that the usage of Intel compilers and MPI libraries on all systems is via the environment modules package (http://modules.sourceforge.net). It was found empirically that using Intel compiler options (1) as described above provided both compatibility of the executables between the systems and reproducibility of the numerical results between Ngamai and Raijin.

General COMmunications (GCOM) library which provides the interface to MPI communication libraries is supplied with the UM sources. GCOM version 4.2 used with UM vn8.2 was compiled with a lower `-O2` optimisation level.

p7402 l24: Please explain why the given environment setting improved the stability of the measured run times.

The following sentence (page 7402, lines 21-23)

> Using the environment setting
> `OMPI_MCA_hwloc_base_mem_alloc_policy=local_only`
> greatly improved stability of the run times

was replaced with

> Support staff at NCI (D. Roberts) investigated this problem. It was found that if cached memory is not being freed when a Non-Uniform Memory Access (NUMA) node has run out of memory, any new memory used by a program is allocated on the incorrect NUMA node and as a result slowing down access. The UM is particularly sensitive to this issue. An environment setting of
>
> ```
> OMPI_MCA_hwloc_base_mem_alloc_policy=local_only  (2)
> ```
>
> was recommended to include in the batch jobs running UM applications. Setting (2) forces all MPI processes to allocate memory on the correct NUMA node, but if the NUMA node is filled, the page file will be used. As a result the usage of (2) greatly improved stability of the run times on Raijin. Addressing these findings on Ngamai, it appeared that (2) was a default setting on that system.

p7403 l14: Please explain why the given Lustre configuration optimised the I/O performance.

Addressing this comment lines **12-14 on page 7403** were replaced with

> Starting from the old Solar system it was found that for I/O performance improvement especially for applications with relatively heavy I/O in order of at least tens of gigabytes Lustre striping had to be used. Based on the experimentation done on all three systems, Lustre striping with a stripe count of 8 and a stripe size of 4M in a form of
>
> ```
> lfs setstripe -s 4M -c 8 <run_directory>
> ```
>
> was used to optimize I/O performance. Here `<run_directory>` is a directory where all output files are produced during a model run. One of the criteria on whether striping parameters were set to near optimal values was based on the consistency of the produced elapsed times for an application running on a relatively busy system.

p7407 l13: Please explain why the given Lustre configuration optimised the I/O performance.

The original text of

> Note that on a very busy such as Raijin system some improvement in the runtimes for a few cases were achieved using different Lustre striping parameters, namely
> `lfs setstripe -s 8M -c <number_of_IO_servers> <run_directory>`

on **page 7407, lines 9-12** was replaced with

> Note that on a very busy such as Raijin system some improvement in the run times for a few cases were achieved using different from (2) Lustre striping parameters, namely
>
> `lfs setstripe -s 8M -c <number_of_IO_servers> <run_directory>`
>
> As in the previous case (2) the values for the Lustre stripe count and the stripe size were found experimentally.

Other changes made in the paper

1. Added a new name in the list of the authors

> **change:** I. Bermous => I. Bermous, P. Steinle

with a couple of related changes:

page 7410, line 14: change “Author” => “Authors”

page 7410, line 15: reference “Peter Steinle (BoM),” was removed

2. Included meaning of the abbreviations used in the text (abstract & text)

Page 7396, line 6
change: “HPC” => “High Performance Computing (HPC)”

Page 7396, line 8
change: “MPI” => “Message Passing Interface (MPI)”

Page 7401, line 16
The original sentence

> On the BoM HPC systems OpenMPI was the only library available.

was replaced with

> On the BoM HPC systems an open source implementation of MPI (OpenMPI) was the only library available.

3. With the changes implemented in the Introduction section the following sentence

> It is very important that the async I/O capability is used.

was replaced with

> With relatively large amount of I/O, performance of the N512 global model and especially its performance scalability is significantly affected by the I/O cost at high core counts. Improvements in the scalability can be achieved with the usage of the asynchronous I/O feature (Selwood, 2012) introduced into the model sources from UM release vn7.8. The I/O servers functionality has been continually improving since then.

on **page 7399, line 14**.

4. The numbers quoted "7.3-10.4%" (**page 7406, line 6**) and "5.5%" (**page 7406, line 11**) were slightly incorrect and have been corrected to "8.2-11.0%" and "5.3%" correspondingly.

5. Addressing a number of referee comments additional UKV model runs have been made on Raijin. A setting in one of the scripts had been inadvertently omitted. The corrected timings and figures for Raijin are now included. There were no substantial changes to the main results. The latest Fig.1, Fig.3 and Fig.4 are attached below. In the titles for the Figures 3 and 4 the original value of 9598 s was changed to 9523 s. The original section "4.1 UKV performance results" has been modified addressing referee comments (see above) and the latest results obtained for the UKV model on Raijin. Also the content of this section has been split up into two sections "4.1.1 Pure MPI vs MPI/OpenMP hybrid" and "4.1.2 Fully committed nodes vs partially committed nodes". Here is the replacement text for the modified section 4.1:

4.1 UKV performance results

> Due to an earlier development of the Unified Model system at the end of 1980's - beginning 1990's on a massively parallel (MPP) system on which each CPU had its own memory, initially the model had only a single level of parallelism using MPI. With the appearance of symmetric multi-processor (SMP) systems in the 1990's and Open Multiprocessing (OpenMP) software the hybrid MPI/OpenMP parallel programming concept which combines MPI across the system nodes and multithreading with OpenMP within a single node was introduced. This concept uses the shared address space within a single node. From the mid 2000's starting from release 7.0 the hybrid parallel programming paradigm was introduced in the UM code and since then the OpenMP implementation has been consistently improving in the

model. Most recent studies (Sivalingam, 2014) showed that even with UM vn8.6 the OpenMP code coverage is limited. Furthermore the efficiency of pure MPI versus the hybrid parallelism depends on the implementation, the nature of a given problem, the hardware components of the cluster, the network and the available software (compilers, libraries) and the number of cores used. As a result, there is no guarantee hybrid parallelism will improve performance for every model configuration.

4.1.1 Pure MPI vs MPI/OpenMP hybrid

Comparison of the best elapsed times produced by running the UKV model with the usage of pure MPI and MPI/OpenMP hybrid parallelism on Raijin and Ngamai is given in Figs 1 and 2 correspondingly. For simplicity the elapsed times are provided for 4 different decompositions starting from the usage of 384 cores with a stride of 384. The run decompositions were 16x24, 24x32, 32x36 and 32x48 with pure MPI usage. In case of the hybrid parallelism, two OpenMP threads were used and the related run configurations using the same number of cores as in the pure MPI case were 2x6x32, 2x12x32, 2x16x36 and 2x16x48, where the first value is the number of threads used. Fig.1 and Fig.2 include results for two cases: fully and partially committed nodes. With partially committed nodes the application was running with the same decomposition as in the fully committed node case, but only a part of each node was used: 8 cores from 12 on Ngamai and 12 cores from 16 cores on Raijin.

With the use of partially committed nodes, the placement/binding of cores to the nodes/sockets should be done in a symmetrical way to give the same number of free cores on each socket. This allows for a better usage of the shared L3 cache on each socket.

Based on the performance results plotted in Fig.2, the usage of pure MPI gives shorter elapsed times than with the usage of the hybrid parallelism for all decompositions on Ngamai. Fig.1 shows that a similar conclusion can be made for the elapsed times obtained on Raijin, excluding the last point with the usage of 1536 cores. At the same time the shortest elapsed times on Raijin are achieved using pure MPI on partially committed nodes (12 cores-per-node) for the whole range of the used cores. Due to the limited proportion of the UM code that can exploit OpenMP, the use of more than 2 threads (3 or 4) showed no improvement in the performance efficiency.

Comparing the actual set of the obtained elapsed times between the two systems with the usage of pure MPI on fully committed nodes of (2604; 1480; 1103; 942) s on Ngamai and (2587; 1484; 1131; 1010) s on Raijin shows that the model performance on Raijin is slightly worse than on Ngamai. At the same time comparing the corresponding elapsed times of (2282; 1288; 947; 844) s on Ngamai and (2125; 1167; 857; 720) s on Raijin with the usage of partially committed nodes for pure MPI, performance and especially performance scaling is better on Raijin. For the same decomposition of 32x48 the elapsed time of 720 s on 2048 reserved cores on Raijin is

14.7% better than the corresponding elapsed time of 844 s obtained on Ngamai on 2304 reserved cores. This improvement reduces with the number of cores used, with only a 6.9% faster time for a decomposition of 16x24. Contributing factors to this include the Raijin cores being slightly faster than Ngamai cores and turbo boost was not enabled in BIOS on Ngamai. With the use of partially committed nodes, memory contention between the processes/threads running on the same node is reduced, improving performance for memory intensive applications. At the same time enabling turbo boost can increase processor performance substantially, reaching peak speeds of up to 3.1 GHz on Raijin using 12 cores-per-node.

On Raijin the usage of 8 out of 16 cores with hybrid parallelism and 2 OpenMP threads showed between 16.6% (for high core counts) to 31.0% (for low core counts) slower run times in the 768-3072 reserved core range in comparison with the corresponding results obtained using pure MPI. On Ngamai the usage of half committed (6 cores from 12 cores) nodes with the hybrid parallelism gives two possible configurations for running an application. Firstly there is the symmetrical case with 1 MPI process and 3 OpenMP threads running on each socket. Another option is the non-symmetrical case with 2 MPI processes running on one socket and 1 MPI process running on another socket with 2 threads per each MPI process. Taking into account the limited OpenMP coverage in the UM vn8.2 sources, tests using hybrid parallelism with a symmetrical case of 3 threads or an asymmetrical case with 2 threads on half committed nodes were not performed on Ngamai.

4.1.2 Fully committed nodes vs partially committed nodes

Elapsed times for the UKV model with pure MPI usage on partially committed nodes on all 3 systems are provided in Fig.3- Fig.8. Each pair of figures (Fig.3-4 for Raijin; Fig.5-6 for Solar and Fig.7-8 for Ngamai) shows speedup as a function of the number of cores actually used as well as a function of the reserved cores (i.e. total number of cores allocated to the run, both used and unused). The performance relative to the number of reserved cores is the most important metric, however performance relative to the “Number of used cores“ provides additional information on the value of reducing the number of active cores per node. This extra information is particularly relevant to circumstances where the elapsed time is more important than using nodes as efficiently as possible. Examples include climate runs and cases where other restrictions mean that the number of nodes available is not a significant constraint on an application's run time. The related performance information cannot be easily seen on the graph using the "Number of reserved cores" metric.

For example, a 12 cores-per-node case on Raijin and a 6 cores-per-node case on Solar each reserved full nodes (16 and 8 cores respectively), but left a quarter of cores unused. This indicates a requirement of specifying by 1/3 of more cores using –npersocket or –npernode option of the mpirun command in comparison with the fully committed case using the same run configuration. In the example of running the model with pure MPI on Raijin and using 12 cores per node the following options

```
mpirun --npersocket 6 –mca orte_num_sockets 2 \
                      –mca orte_num_cores 8 . . .
```

were used in the mpirun command with OpenMPI 1.6.5. The last two options specify the number of sockets on a node and the number of cores on each socket. These options were required to avoid bugs found in the OpenMPI 1.6.5 software.

Fig.3 shows the value of partially committed nodes on Raijin where using 12 cores-per-node significantly improves the model scaling. This improvement generally increases as the number of active cores increases. The improvement reaches 28.5% the performance of using 1728 fully committed nodes. The usage of 8 cores-per-node on Raijin gives an additional performance improvement in comparison with the 12 cores-per-node case varying from 16.8% at 96 cores to 9.6% at 1728 cores. Examining the same performance results on the reserved cores basis as in Fig.4 shows that it is more efficient to use 12 cores-per-node than fully committed nodes just with over 768 cores. On 768 reserved cores the 12 cores-per-node case has value of 1495 s for a 24x24 decomposition and the fully committed node case has value of 1484 s for a 24x32 decomposition.

Fig. 5 shows that the usage of partially committed nodes on Solar improves the runtimes with 6 cores-per-node by 6.9-16.5% and a further reduction of 8.2-11.0% is achieved with the usage of 4 cores-per-node.

The speedup curves as a function of used cores on Ngamai shown in Fig.7 indicate that the model runs 10.4-14.6% faster with 8 cores-per-node. Unlike the other two systems (Raijin and Solar), the use of half utilised nodes with 6 cores-per-node on Ngamai gives only a very modest reduction of no more than 5.3%. These latter results indicate that a reduction in memory contention with the 6 cores-per-node case has almost no impact over using 8 cores-per-node.

Speedup curves as functions of the reserved cores for Solar (Fig. 6) and Ngamai (Fig. 8) show that unlike Raijin, the efficiency gains on partial nodes were not achieved on up to 1152 reserved cores on Solar and 1728 on Ngamai. A relatively poor UKV performance on partial nodes especially on Ngamai system in comparison with Raijin was due to unavailability of turbo boost on that system. Turbo boost would allow active cores to run at up to 16% higher clock speeds with the usage of 8 cores-per-node on Ngamai.

The shortest run times using fully committed nodes on Ngamai and Raijin with the usage of up to 1728 cores were achieved on a decomposition of 36x48 with pure MPI. The largest allowable decomposition under the constraints provided in Sect. 2.2.1 is 42x48 on 2016 cores. Increasing the number of cores from 1728 to 2016 provides further improvements in the elapsed times of 1.7% on Ngamai and 2.2% on Raijin. This indicates that the corresponding performance scaling have not begun to level off at the largest allowed decomposition of 42x48. At the same time, with the use of

partially committed nodes on Raijin, an elapsed time of 950 s obtained on fully committed nodes for a 36×48 decomposition can be improved by: 28.5% (679 s) on 2304 reserved cores with 12 cores-per-node or 35.4% (614 s) on 3456 reserved cores with 8 cores-per-node.

The above mentioned constraint of 2016 cores on fully committed nodes is applied for pure MPI only. With the usage of hybrid parallelism and 2 OpenMP threads on fully committed nodes performance of the model is still improving when the number of cores is increasing from 1536 for decomposition of 16x48 to 3072 for a decomposition of 32x48 but the corresponding run times are still greater than the run times obtained with pure MPI on partial nodes using the same number of reserved cores. For example, the use of multithreading with 2 OpenMP threads and decomposition of 30x48 on 3072 cores gives an elapsed time of 669sec. This run time is improved by 9.1% (608 s) with the use of pure MPI for a decomposition of 40x48 run on 10 cores-per-node with the same 3072 cores.

6. Page 7396, line 5: removed the indefinite article "a" from the expression

for a better application performance

to

for better application performance.

7. Page 7398, line 18: changed the reference from original

Fraser, 2012

to

NMOC, 2012

8. Page 7399, lines 4-5: two minor changes in the original sentence from

The resolution of the currently used Global N320 (40km) model with 70 vertical levels in APS1 will be upgraded to N512 (25km) 70 levels in APS2.

to

The resolution of the currently operational Global N320 (40km) model with 70 vertical levels in APS1 will be upgraded to N512 (25km), 70 levels in APS2.

9. Page 7399, lines 10-11: changes in the sentence from

The operational systems run 4 times daily include two different Global model runs for 3 and 10 days.

to

The operational systems run 4 times daily with two runs for 3 and two runs for 10 days.

10. Page 7407, line 4: added the definite article "the" with the following change from

Using UM multithreaded I/O servers feature

to

Using the UM multithreaded I/O servers feature

11. Added name of Martyn Corden (Intel) in the Acknowledgements section.

12. Page 7411, line 14: Removed "Fraser, J.:", as a result the corresponding 2 lines (14 and 15) of text were moved after line 17 on the same page.

13. Page 7411, line 26: added after this line a new reference

Sivalingam, K.: Porting and optimisation of MetUM on ARCHER**,** The 16th ECMWF Workshop on High Performance Computing in Meteorology, available at: http://www.ecmwf.int/sites/default/files/HPC-WS-Sivalingam.pdf (last access: 28 January 2015), 2014

14. For the consistent usage of the horizontal grid notation as per its first reference of "West–East × South–North" on page 7399, line 7 the following change has been made:

page 7400, line 9: "E-W x N-S" => "W-E x S-N"

15. Page 7410, line 20: added text

We thank two anonymous reviewers for the recommendations to improve the original version of the manuscript.

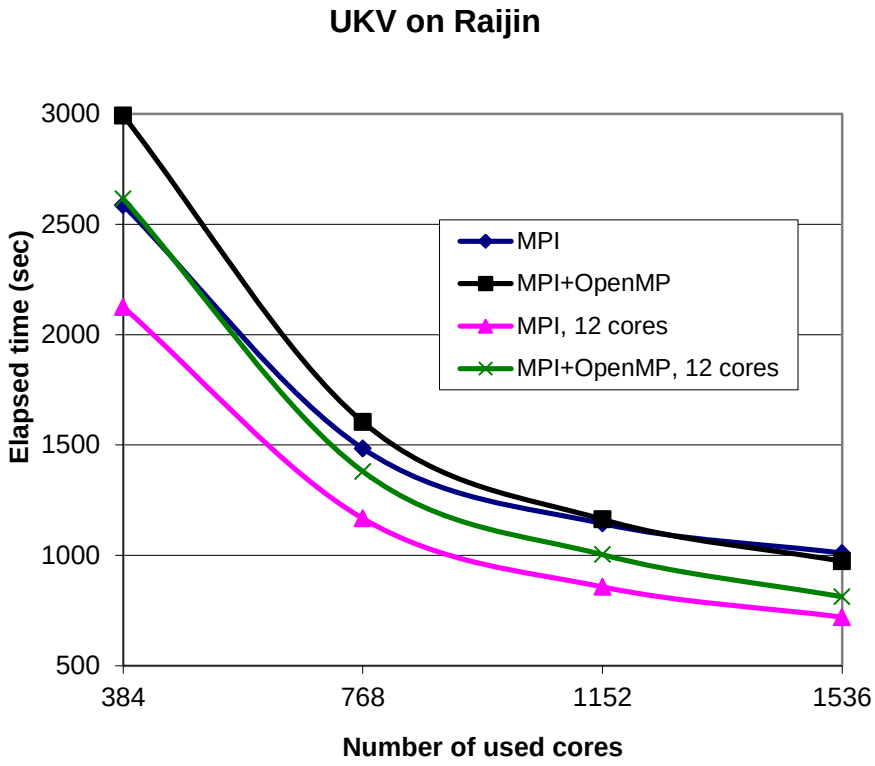

Figure 1. Elapsed times for the UKV model runs on Raijin versus the number cores actually used in each run.

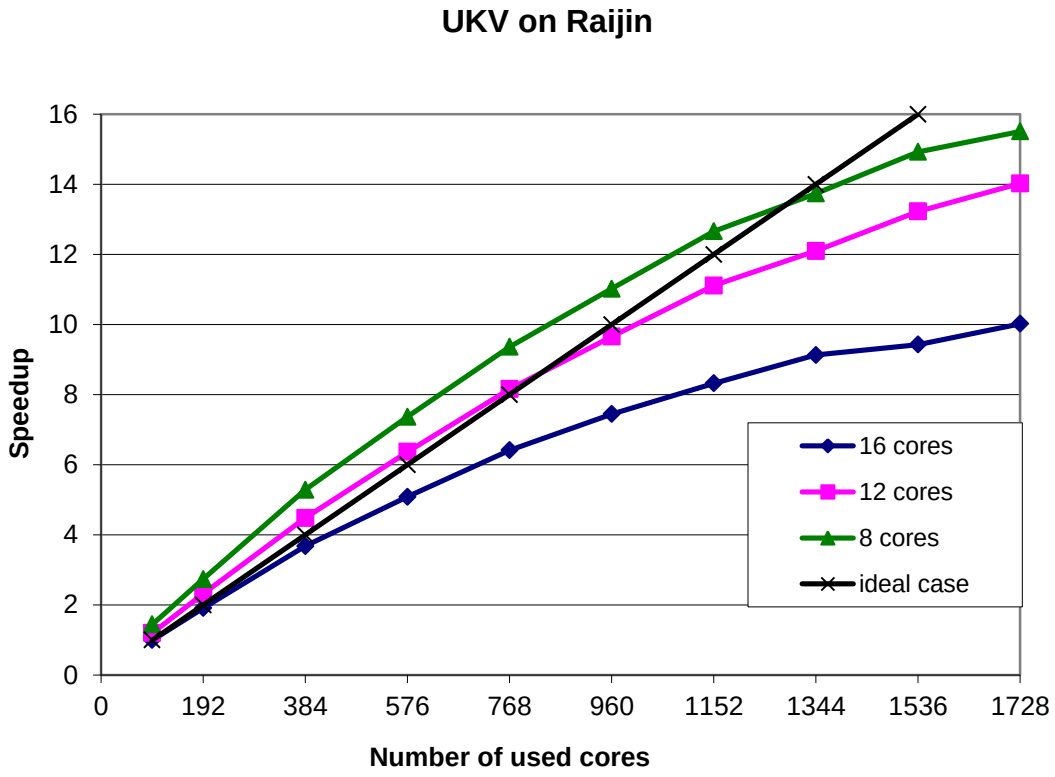

Figure 3. Speedup as a function of number of used cores on Raijin. Speedup was calculated in relation to the elapsed time of 9523 s obtained for a 96 core run on fully committed nodes.

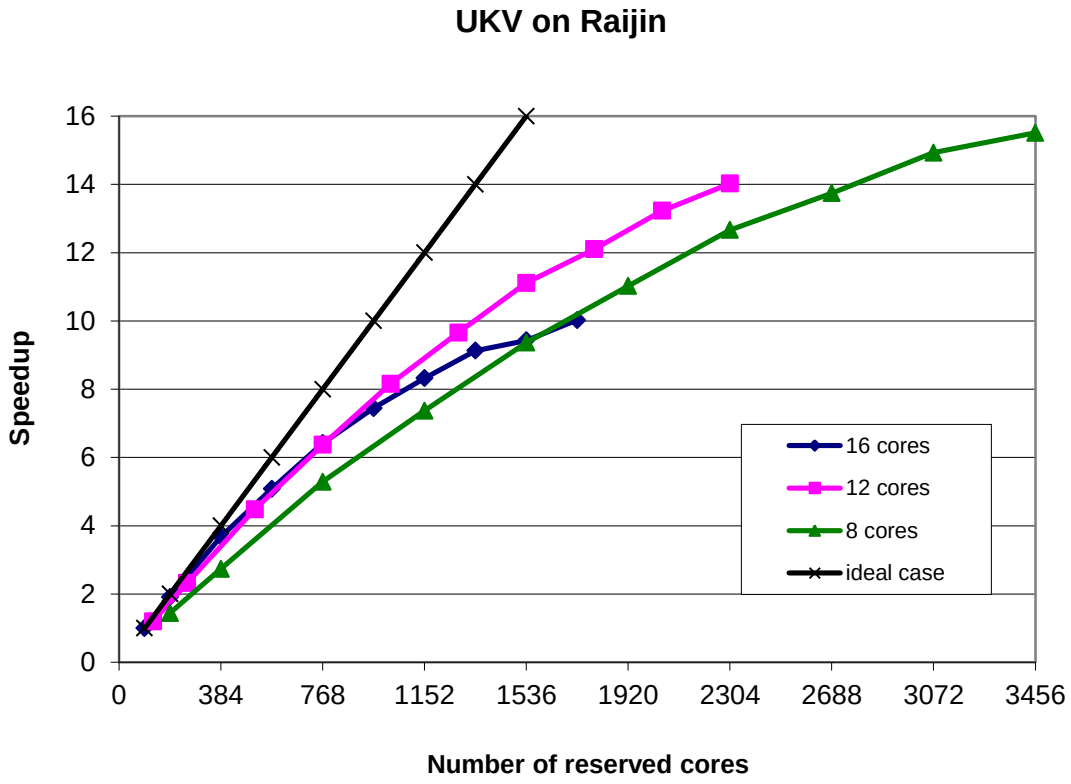

Figure 4. Speedup as a function of number of reserved cores on Raijin. Speedup was calculated in relation to the elapsed time of 9523 s obtained for a 96 core run on fully committed nodes.