
Proceedings Paper:
Richardson, M, O'Connor, F, Mann, GW et al. (2016) Computational Efficiency Of The Aerosol Scheme In The Met Office Unified Model. In: CUG 2016 Proceedings. ECMWF Cray User Group (CUG) conference, 08-12 May 2016, London.

Computational Efficiency of the Aerosol Scheme in the Met Office Unified Model

Mark Richardson, Graham Mann,
NCAS, School of Earth and Environment
University of Leeds
Leeds, United Kingdom
earmgr@leeds.ac.uk

Fiona O'Connor, Paul Selwood
UK Met. Office
Exeter, United Kingdom

fiona.oconnor@metoffice.gov.uk

Abstract - A new data structuring has been implemented in the Met Office Unified Model (MetUM) which improves the performance of the aerosol subsystem. Smaller amounts of atmospheric data, in the arrangement of segments of atmospheric columns, are passed to the aerosol sub-processes. The number of columns that are in a segment can be changed at runtime and thus can be tuned to the hardware and science in operation. This revision alone has halved the time spent in some of the aerosol sections for the case under investigation. The new arrangement allows simpler implementation of OpenMP around the whole of the aerosol subsystem and is shown to give close to ideal speed up. Applying a dynamic schedule or retaining a simpler static schedule for the OpenMP parallel loop are shown to differ related to the number of threads. The percentage of the run spent in the UKCA sections has been reduced from 30% to 24% with a corresponding reduction in runtime by 11% for a single threaded run. When the reference version is using 4 threads the percentage of time spent in UKCA is higher at 40% but with the OpenMP and segmenting modifications this is now reduced to 20% with a corresponding reduction in run time of 17%. For 4 threads the parallel speed-up for the reference code was 1.78 and after the modifications it is 1.91. Both these values indicate that there is still a significant amount of the run that is serial (within an MPI task) which is continually being addressed by the software development teams involved in MetUM.

Keywords- OpenMP; cache-blocking; climate modelling, aerosol

I. INTRODUCTION

A. *Climate simulations*

The United Kingdom Met. Office (UKMO) develop, produce, use and distribute versions of a numerical weather simulation software known as the "Unified Model" (MetUM). It is used by several agencies worldwide for weather forecasting, climate modelling and several other atmospheric situations.

For the climate modelling work there is a requirement for higher detailed calculation of the aerosol and chemistry processes than for production forecasting. Previously this has been done using "CLASSIC" but this work is focused on the replacement tool, UKCA, which incorporates GLOMAP and ASAD. Specifically we will be reporting work with a configuration of GLOMAP that uses offline oxidants. In this configuration there is limited chemical processing so the work can demonstrate performance improvements purely associated with GLOMAP. This approach will help to get the developments into production more quickly than trying to deal with the complexity of all UKCA at once.

The configuration being used for an international assessment of climate modelling is as follows: N96L85 (i.e. 192 cells East to West by 144 cells North to South and 85 levels.) this will be run in two modes one with atmosphere only and the second by coupling it to an ocean model (1 degree resolution by 75 levels). On a high priority queue, the model is achieving 15 months per day using 1 month per chained job. The MPI decomposition is 16x14 2D topology with 2 OpenMP threads. Also this configuration calls the aerosol and chemistry every 3rd dynamical time step which is 20 minutes.

B. *Further detail about GLOMAP*

The GLObal Model of Aerosol Processes (GLOMAP) is a size-resolved microphysical aerosol model which has been developed at Leeds [1,2] and is an extension to the TOMCAT chemical transport model. The aerosol scheme developed for UKCA is a simplified version of GLOMAP-bin, using a 2-moment modal approach rather than a 2-moment sectional approach to reduce CPU cost, and has been named GLOMAP-mode when used in the TOMCAT environment.

There are four soluble modes — nucleation, Aitken, accumulation and coarse — and three insoluble modes — Aitken, accumulation and coarse. The nucleation mode consists purely of sulfate aerosol and the insoluble accumulation and coarse modes consist purely of (fresh) dust aerosol. The insoluble Aitken mode consists of an external mixture of black carbon and organic carbon aerosol. All other modes consist of internally mixed particles each with identical composition given by the partial component masses (which does not vary with size).

It is clear that this level of sophistication brings with it additional computational load. The notable aspect is that all this processing occurs within a single computational cell (grid-box), which can be processed in any order.

II. WORK PROGRAMME APPROACH

The work is being done in two stages: segmentation of data for cache-blocking and implementation of high level OpenMP to encompass the aerosol calculation. The segmentation method itself gives rise to performance

improvement and subsequent application of OpenMP around those cache blocks allows MetUM to utilise cores that would be usually idle while the aerosol calculations happen.

A. Test case

Test case represents the global atmosphere with192x144x85 grid-boxes. For this work only one configuration was used and that is representative of general use: 128 MPI tasks and 2 OpenMP threads. However, we will show that this development version is less wasteful of resource and provides an overall speed improvement of the simulation. The development work has set the test case to run for only 3 days that is a restart from a "spun-up" atmosphere i.e. the standard UM was used with standard GLOMAP to generate an atmosphere after 14 months of simulated time. With an atmospheric time step of 20 minutes there are 216 steps for the subsequent 3-day development simulation.

B. Performance timing

There are already two timing mechanisms within MetUM that were extensively used during this work, one is a basic timer that summarises the time spent in code sections at the end of the simulation. In recent versions the DrHook system [2] has been implemented and gives a finer grained time information. After each run a set of files (one per MPI task) are processed to provide tables organised by the average time spent in traced functions.

There are some limitations found for the analysis with Dr. Hook. When post processing the log files one selects the number of routines to include but adding extra OMP threads adds as many more timing entries for the functions. Some of the functions that appear for a run with, e.g. two threads, do not necessarily appear in the results for a run with eight threads. One reason for function times disappearing from the list is that the function time may have reduced and moved the function out of the range of functions selected for post processing.

III. ANALYSIS AND RESULTS

A. Timing analysis

Figure 1 shows two examples of the DrHook output for runs with two OpenMP threads active. In figure 1(a) for the reference code, it is clear that UKCA functions make a significant contribution to the runtime. In figure 1(b) for the code where UKCA has been modified for cache-blocking and OpenMP, ukca_cond_coff_v() has simply halved but the ukca_coagwithnucl function as quartered. The latter is due to the effect of reducing the size of the arrays being processed.

B. Cache blocking

1) Implementation of the segmenting method

Early plans for this work considered examining each high-workload subroutine and restructuring the do-loops for cache blocking. Some experiments were done and to get improvements into production quickly one particular solution has been adopted separate to this work. However, in general the amount of effort to make changes to all the possible loops would be very time consuming. The approach detailed in this paper is seen as a better general solution without impacting the GLOMAP source code other than at the interface with whichever general circulation model it is incorporated.

There is a section of MetUM where the code enters an interface that sets up the data for the aerosol calculations. This is named "ukca_aero_ctl()" and it is here where some data restructuring already occurs to map from three-dimensional arrays of the whole atmosphere into the one-dimensional vectors used by GLOMAP. The adoption of a segmenting approach replaces those transformations with "segment extraction" so that only a small amount of atmosphere is passed to GLOMAP making those one-dimensional vectors smaller. This requires a further two nested loops around the aerosol calculations: the outer loop for the rows in the sub-domain and a second immediately nested inside the first for iterating over the number of segments on a row.

This implementation limits the segments to be a factor of the number of columns in a row of computational space. In the

Number of PEs: 128

Header fields

	Min	Mean	Max	(Max-Min)
Instrument overhead (%)	0.8 (PE 122)	0.99	1.23 (PE 61)	0.43
Heap (MB)	567 (PE 9)	577.20	607 (PE 18)	40
RSS (MB)	473 (PE 69)	574.47	615 (PE 18)	142
Stack (MB)	0 (PE 0)	0.00	0 (PE 0)	0
Paging	0 (PE 0)	0.00	0 (PE 0)	0
Wall Time (s)	399.33 (PE 124)	401.47	407.91 (PE 95)	8.58
Thread#1 (s)	399.3 (PE 9)	399.31	399.31 (PE 0)	0
Thread#2 (s)	22.14 (PE 115)	36.42	52.64 (PE 40)	30.50
Thread#1 (%)	97.89 (PE 95)	99.46	99.99 (PE 107)	2.10
Thread#2 (%)	5.54 (PE 115)	9.07	13.03 (PE 40)	7.49

Ordering routines by self: mean

	Min	Mean	Max	(Max-Min)
UKCA_*	(PE)	140.70	(PE)	0
TIMER@1	22.511 (PE 61)	44.59	68.422 (PE 125)	45.91
UKCA_COAGWITHNUCL@1	28.523 (PE 5)	30.90	32.733 (PE 24)	4.21
ATMOS_PHYSICS1@1	23.358 (PE 82)	25.97	28.39 (PE 30)	5.03
UKCA_ABDULRAZZAK_GHAN@1	10.16 (PE 108)	22.68	29.909 (PE 70)	19.75
UKCA_COND_COFF_V@1	15.662 (PE 118)	17.50	19.009 (PE 23)	3.35
HALO_EXCHANGE:SWAP_BOUNDS_NS_DP@1	10.071 (PE 125)	15.36	20.534 (PE 72)	10.46
UKCA_RADAER_BAND_AVERAGE@2	7.44 (PE 116)	11.69	14.808 (PE 63)	7.37
UKCA_RADAER_BAND_AVERAGE@1	7.485 (PE 124)	11.57	14.414 (PE 61)	6.93
eg_CUBIC_LAGRANGE@1	9.897 (PE 71)	10.07	11.525 (PE 119)	1.63
U_MODEL_4A@1	1.175 (PE 70)	9.90	25.209 (PE 108)	24.03
EG_CORRECT_TRACERS_UKCA@1	7.221 (PE 121)	7.62	7.879 (PE 48)	0.66
GLUE_CONV_6A@1	3.456 (PE 101)	6.68	12.816 (PE 59)	9.36
GLUE_CONV_6A@2	3.355 (PE 108)	6.63	12.65 (PE 77)	9.29
UM_WRITDUMP@1	0.482 (PE 0)	6.45	6.751 (PE 6)	6.27
EG_INTERPOLATION_ETA@1	4.232 (PE 71)	6.30	9.533 (PE 122)	5.30
HALO_EXCHANGE:SWAP_BOUNDS_EW_DP@1	5.801 (PE 121)	6.22	6.568 (PE 90)	0.77
SCATTER_FIELD_MPL@1	1.644 (PE 0)	6.20	6.64 (PE 108)	5
UKCA_CONDEN@1	4.916 (PE 115)	6.02	6.555 (PE 61)	1.64

Number of PEs: 128

Header fields

	Min	Mean	Max	(Max-Min)
Instrument overhead (%)	0.97 (PE 121)	1.18	1.44 (PE 61)	0.47
Heap (MB)	1391 (PE 112)	1403.03	1430 (PE 40)	39
RSS (MB)	501 (PE 112)	525.65	563 (PE 40)	62
Stack (MB)	0 (PE 0)	0.00	0 (PE 0)	0
Paging	0 (PE 0)	0.00	0 (PE 0)	0
Wall Time (s)	343.48 (PE 107)	345.88	352.42 (PE 95)	8.94
Thread#1 (s)	343.44 (PE 0)	343.44	343.44 (PE 0)	0
Thread#2 (s)	49.7 (PE 123)	65.70	78.93 (PE 46)	29.23
Thread#1 (%)	97.45 (PE 95)	99.30	99.99 (PE 107)	2.54
Thread#2 (%)	14.46 (PE 123)	18.99	22.61 (PE 46)	8.15

Ordering routines by self: mean

	Min	Mean	Max	(Max-Min)
UKCA_*	(PE)	118.98	(PE)	0
TIMER@1	24.25 (PE 49)	44.68	66.561 (PE 125)	42.31
ATMOS_PHYSICS1@1	23.491 (PE 59)	25.96	28.447 (PE 49)	4.96
UKCA_ABDULRAZZAK_GHAN@1	10.483 (PE 108)	22.61	29.431 (PE 71)	18.95
HALO_EXCHANGE:SWAP_BOUNDS_NS_DP@1	9.884 (PE 13)	15.45	20.501 (PE 111)	10.62
UKCA_RADAER_BAND_AVERAGE@2	7.485 (PE 116)	11.76	14.873 (PE 63)	7.39
UKCA_RADAER_BAND_AVERAGE@1	7.488 (PE 124)	11.58	14.488 (PE 64)	7
eg_CUBIC_LAGRANGE@1	9.884 (PE 19)	10.06	11.45 (PE 119)	1.57
UKCA_COND_COFF_V@1	7.415 (PE 119)	8.41	9.099 (PE 43)	1.68
UKCA_COND_COFF_V@2	7.584 (PE 97)	8.41	9.147 (PE 58)	1.56
UKCA_COAGWITHNUCL@2	7.872 (PE 116)	8.13	8.468 (PE 85)	0.60
UKCA_COAGWITHNUCL@1	7.82 (PE 2)	8.10	8.403 (PE 81)	0.58
U_MODEL_4A@1	0.485 (PE 71)	7.70	20.268 (PE 123)	19.78
EG_CORRECT_TRACERS_UKCA@1	7.237 (PE 120)	7.62	7.839 (PE 48)	0.60
GLUE_CONV_6A@1	3.389 (PE 101)	6.68	12.87 (PE 59)	9.48
GLUE_CONV_6A@2	3.35 (PE 65)	6.62	12.605 (PE 77)	9.26
UM_WRITDUMP@1	0.473 (PE 0)	6.32	6.632 (PE 2)	6.16
EG_INTERPOLATION_ETA@1	4.433 (PE 55)	6.29	10.008 (PE 117)	5.57
HALO_EXCHANGE:SWAP_BOUNDS_EW_DP@1	5.804 (PE 13)	6.20	6.645 (PE 124)	0.84

Figure 1 DrHook timing information for (a) reference code (b) the development code

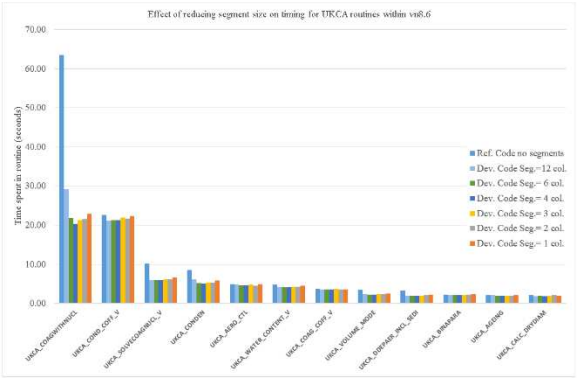

Figure 3: Reducing segment size for top 12 UKCA functions

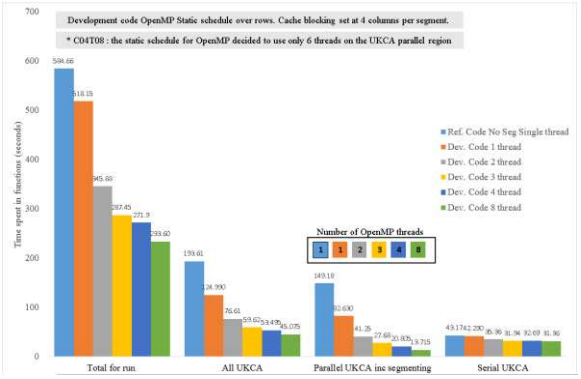

Figure 2 Effect of OpenMP on UKCA highlighting contributions from parallel and serial regions

case under investigation the row length is 12 and thus a value of 4 for the number of columns per segment will result in 3 segments per row and is the same for all rows. It is expected that the optimum segment size will be less than the row length as the resolution of simulations increases. This implementation makes it straightforward to activate an OpenMP region as discussed later in this report.

2) Segment size

The whole simulation HPC job is configured with the UMUI (or Rose depending on the version of MetUM) where the number of columns per segment are set. This is read from a name list during the run and it is used to size the segments during the transformations within ukca_aero_ctl(). The inclusion of this new parameter required modification of those configuration tools.

For this experiment the case was repeatedly run for variations in the choice of number of columns per segment i.e. 12, 6, 4, 3, 2 and 1. The most notable effect is that the time spent in UKCA_COAGWITHNUCL, the highest workload that was identified in the reference runs, is halved as shown in figure 2. Some of the functions demonstrate very little variation of the time for execution due to the dominant processes not being directly related to the number of grid-boxes being processed. Only 12 functions are shown and these occur below the aerosol interface and contribute a significant amount to the runtime. Many of the other (approximately 70) subroutines account for less than a second each, some are too short to register a time and assigned "zero". There are two other subroutines outside the segmenting region that are significant, of the order of 25seconds each for the reference run. They are not modified by this work but appear in the overall results.

C. OpenMP Investigation

The investigation of the OpenMP implementation is focused on the optimal 4 columns per segment i.e. only 3 segments per row, which equates to 54 segments. The fact that there are two do loops to count those segments is slightly inhibiting as the parallel loop is applied to the outer loop only (over rows) so there are only 18 levels of potential parallelism. The work presented here is for applying OpenMP to only the outer loop over rows.

1) Number of threads

There is already a certain amount of MetUM within OpenMP parallel regions and it is seen to run faster with more threads. Some of the speed-up will also come from under populating the nodes as was seen in previous work with GLOMAP in TOMCAT [5, 6]. The under population refers to the spreading out of the MPI tasks to allow CPUs for the OpenMP threads to be "close" to the MPI task that spawns them.

The four groupings in Figure 3 have been collated to demonstrate the benefits from the development. Within each group is the effect of varying the number of threads with the first column of each group being for the reference run with a single thread. The first group is all of the MetUM including the UKCA work, i.e. the total wall time for the run and is likely of most interest to the scientist using the software. The

TABLE I. EFFECT OF OPENMP ON UKCA DATA FOR FIGURE 3

	Ref 1 thread	Dev 1 thread	Dev 2 thread	Dev 3 thread	Dev 4 thread	Dev 8 thread
Wall time	584.66	518.15	345.88	287.45	271.90	233.60
All UKCA	193.61	124.99	76.61	59.62	53.50	45.08
Parallel UKCA	149.18	82.63	41.25	27.68	20.81	13.72
Serial UKCA	43.17	42.29	35.36	31.94	32.69	31.36
Percentage in UKCA	**33.11**	**24.12**	**22.15**	**20.74**	**19.67**	**19.29**

second group is for all of the UKCA functions within a simulation – this group is active for a climate run and constitutes an "overhead" for the aerosol calculations. The third group is the components of UKCA that are within OpenMP parallel regions. The fourth group is the parts of UKCA that are not in any OpenMP region. The original sentiment from users that there is a large overhead for using UKCA is demonstrated as even with the choice of hourly aerosol processing it accounts for 33% of the run (Table 1). Now that segmenting and OpenMP have been implemented it can be seen that even for single threaded operation the data restructuring has reduced this to be only 24% of the run.

It is clear from Figure 3 that there is still some development needed to improve the performance of UKCA by locating the remaining serial parts of UKCA and implementing an OpenMP region around them. A few of the UKCA routines are separate from the OpenMP parallel region that was introduced by this work but they lay within a parallel region elsewhere in MetUM. Their times have been accumulated with the UKCA functions in the newly implemented parallel region.

2) Choice of scheduling

When implementing an OpenMP parallel region the first approach is to determine which do-loops will be parallel enabled. Within the OpenMP standard there is a specification of the options that can be applied to a do-loop. Apart from which data are shared and which data are private, the manner in which the iterations are distributed can be assigned to be static or dynamic [3]. For a static schedule the iterations are divided equally among the threads in "chunks". For example, a 3 thread run for this case would assign the first 6 rows to thread 0, the next 6 rows to thread 1 and then the final 6 rows to thread 2. This is indiscriminate and does not account for any difference in the time required for each iteration (row).

A dynamic schedule will assign the first (number of threads x OpenMP chunk size) iterations as one segment to each thread and then when a thread becomes available the next chunk of iterations are handed to it. If any chunk of work takes more time then it will not hold up the other threads (except possibly one of the final "number of threads" chunks is the difficult one.)

Again an optimal segment size of 4 columns per segment was chosen the chunk size was left as the default of 1. The case was run several times varying the number of threads i.e. 2, 4 and 8. One set of runs was performed using the SCHEDULE (STATIC) clause and the second series ran with SCHEDULE (DYNAMIC) clause.

In figure 4 the DYNAMIC schedule is seen to have a detrimental effect on the low thread (2) count compared to the STATIC clause. It appears that the higher thread count (8) benefits from the dynamic scheduling. This is likely due to the fact that the 18 rows are not a multiple of 8. In practice, the static scheduler decided to use only 6 threads and divide the work evenly. This was confirmed during examination of the log files where each thread reports the iteration with which it is working. It is also likely that some of the rows require different amounts of work so that the dynamic method has the opportunity to assign a new tranche of work to the next available thread. The impact of this indirect load balancing is

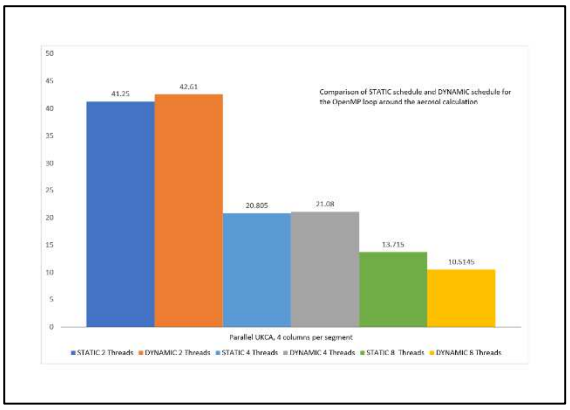

Figure 4 Comparison of STATIC and DYNAMIC OpenMP schedules for 2, 4 and 8 threads.

beneficial for the case where there are more threads than tasks, it was confirmed that all threads did some work when DYNAMIC scheduling was active for the 8-thread run.

As normal working practice is to use a small number of threads it is recommended that in this parallel region STATIC scheduling is applied until more of MetUM is OpenMP parallel enabled.

3) COLLAPSE clause to merge nested do loops

The methodology as explained above was the most straightforward to implement of several possible methods. The inherent limitation to segments being a factor of the row length could be alleviated by merging the inner loop over segments with the outer loop over rows such that the OpenMP scheduler sees the total number of segments. A simple way of achieving this is to add the COLLAPSE clause to the OpenMP loop directive. Unfortunately the version of compiler (8.3.4) cannot merge the loops where the limits are set during the run. This work will be done as a new compiler becomes available.

An alternative is to implement the more complex coding solution of choosing a segment size and extracting the 3D data into these arbitrary segments within a single loop structure that iterates over all segments rather than restricting it to segments per rows. This is a possible development direction but brings with it further maintenance and clarity issues.

IV. SUMMARY

The well-known cache-blocking concept has been applied to the MetUM around the calculations for aerosol. Benefits were seen in the form of reduced time for calculation of the aerosols. In turn the "block" loop was used as the level of parallelism for OpenMP. This is not a traditional tiling of nested do loops as the loop is applied at such a high level as to encompass many subroutines (almost all) that do the aerosol calculations.

The scaling for the aerosol section is close to ideal. The overall scaling effect of OpenMP on the reference code and the development code can be seen in figure 5. The improvement of the development code over the reference code for 1, 2 and 4 threads is 11%, 14% and 17% and is a combination of segmented approach and introducing OpenMP around the aerosol calculations.

TABLE 2: DATA FOR FIGURE 5: OVERALL EFFECT OF CACHE-BLOCKING AND OPENMP

	1 thread	2 threads	4 threads
Ref Code wall time	584.89	401.47	328.33
Dev Code wall time	518.15	345.88	271.90

An additional benefit is that it has also made better use of reserved resource when using OpenMP. Current working practice is to select a small number of OpenMP threads as those additional cores are idle during the aerosol calculation. Activating OpenMP around the aerosol at the high level interface results in further reduction in the cost of the aerosol calculations. This complements other work elsewhere in the MetUM where OpenMP is being extended to apply to more of the atmosphere calculations.

V. PLANNING FURTHER WORK

Currently work is in progress to take the successful branch development and merge it with the trunk core MetUM. This is a significant software engineering exercise because the quality assurance criteria have to be maintained. After a development has been validated, a test suite is run and any errors investigated and corrected.

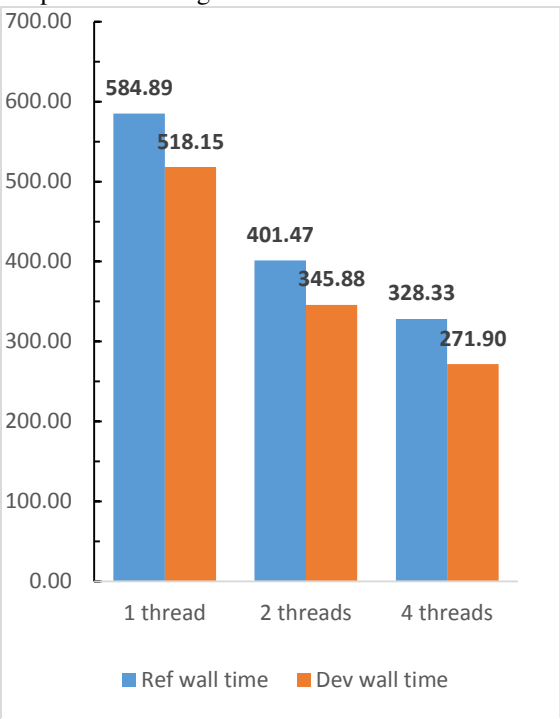

Figure 5 Overall effect of this work comparing reference and development runs

At the outset of this project the whole of UKCA has been identified as needing performance enhancement. The aerosol processes were chosen as a starting point with the chemistry to follow over the next 12 months. There exist structures in the chemistry calculations where the atmosphere is treated in a layer by layer fashion. These are arbitrary as the solver will work on individual computational boxes in a similar fashion to the aerosol processes. The longer view is to deal with chemistry in a column fashion and that work will examine the feasibility of introducing columns at the outset or to focus on applying OpenMP to the existing layering.

ACKNOWLEDGMENTS

Joint Weather and Climate Research Programme (http://www.jwcrp.org.uk/) for funding this work.

National Centre for Atmospheric Science (https://www.ncas.ac.uk/) for hosting the research at the University of Leeds.

The UK Met Office for guiding the research and providing the collaborative resource (MONSooN) which is a Cray XC40 that complements their internal systems.

REFERENCES

[1] Spracklen, D. V., Pringle, K. J., Carslaw, K. S., et al.: A global off-line model of size-resolved aerosol microphysics: I. Model development and prediction of aerosol properties, Atmos. Chem. Phys., 5, 2227–2252, doi:10.5194/acp-5-2227-2005, 2005

[2] G. W. Mann, K. S. Carslaw, D. V. Spracklen, D. A. Ridley, P. T. Manktelow, M. P. Chipperfield, S. J. Pickering, and C. E. Johnson , "Description and evaluation of GLOMAP-mode: a modal global aerosol microphysics model for the UKCA composition-climate model" Geosci. Model Dev. , 2010, 3, pp519–551

[3] Chandra R, Dagum L, Kohr D, Mayden D, McDonald J, Menon R, Parallel Programming in OpenMP, Academic Press, 2001, ISBN-13:978-1-55860-671-5

[4] DrHook, time measurement system, ECMWF (www.ecmwf.int)

[5] M.Richardson, M.Chipperfield, Cray User Group 2013 (Nappa, California, USA) "Improvement of TOMCAT-GLOMAP File Access with User Defined MPI Datatypes"

[6] M.Richardson, G.W.Mann, Cray User Group 2010 (Edinburgh, Scotland, UK) "Combining OpenMP and MPI within GLOMAP Mode: An Example of Legacy Software Keeping Pace with Hardware Developments".