HECToR distributed CSE project report: Implementation of OpenMP within MPI-TOMCAT-GLOMAPmode

Dr. Mark Richardson, Numerical Algorithms Group
Dr. Graham Mann, National Centre for Atmospheric Science, School of Earth and Environment, University of Leeds
Prof. Martyn Chipperfield, School of Earth and Environment, University of Leeds
Prof. Ken Carslaw, School of Earth and Environment, University of Leeds
May 2010

Abstract

The 3D offline aerosol-chemistry transport model MPI-TOMCAT-GLOMAPmode has been enhanced by updating and re-designing existing Open MP directives around several do-loops in the dominant CPU costing parts of the code. Utilising OpenMP and MPI in the code allows it to parallelise more effectively on the multi-core HECToR environment with each MPI task acting like an SMP program with several additional worker threads. This new hybrid OpenMP/MPI version substantially reduces model turnaround times and makes longer and higher resolution model configurations possible. The new hybrid version of the model also allows the MPI task to access the whole of the memory on the node without competition.

The enhancements have now given the test case good scalability up to 128 cores whereas previously MPI communication bottlenecks restricted model configurations to running on 64 cores. The speed up, on the Cray XT4, achieved with 2 Open MP threads is 1.4, which is close to ideal as 45% of the code is inherently serial. Using 4 threads provides a speed-up of 2 which is also close to ideal.

The test case has been run on the XT6 and the speed-up due to Open MP has been shown to be different for a case with 32 MPI tasks than for the same case when decomposed for 64 processors. In the former configuration the optimum is 2 threads whereas in the latter it is 3 threads.

List of figures

1 Introduction

The aim of this project is to improve the performance of the MPI-TOMCAT-GLOMAPmode 3D offline aerosol-chemistry transport model by updating and improving existing Open MP directives in the code to better utilise multi-core supercomputing architectures. The work is funded by the EPSRC HECToR distributed CSE programme (dCSE). The technical work is done by NAG Ltd staff, while the Principal Investigator (PI) overseeing the work is a National Centre for Atmospheric Science (NCAS) researcher based at the University of Leeds School of Earth and Environment. The beneficiaries of this work include the large group of academic researchers using TOMCAT-GLOMAPmode in Leeds and other UK Universities as well as overseas users in Finland (Helsinki University) and Australia (CSIRO). .

This work was started while HECToR was configured as a Cray XT4h (four cores per node) and was still active when the system was upgraded to a Cray XT6 (i.e. 24 cores per node but without Gemini interconnect). The test case used for this project is the T42 which iis described in section 4.

1.1 The application software

TOMCAT-GLOMAPmode is a research tool used to simulate the transport and transformation of aerosol particles on the global scale and to understand how changes in aerosol properties over the industrial period are affecting the Earth's climate. . The 3 main components of the model are

i) the transport module containing routines for tracer advection, convection and boundary layer (BL) mixing (see e.g. Chipperfield, 2006),
ii) the atmospheric chemistry module which utilises the ASAD chemical integration software (Carver et al. 1997) and
iii) an aerosol microphysics module GLOMAP-mode (Mann et al., 2010).

The model had already been the subject of one previous HECToR dCSE project to improve its performance on the HECToR supercomputing resources. The 3D global aerosol-chemistry transport model operates on a fixed longitude/latitude grid in the horizontal with vertical levels being terrain-following at the surface via the sigma co-ordinates and shifting to being purely on pressure levels in the upper part of the 3D domain in the stratosphere. The model is "offline" in the sense that the 3D wind, temperature and humidity fields are prescribed according to 6-hourly varying meteorological re-analysis fields such as provide by the European Centre for Medium-Range Weather Forecasting (ECMWF).

The TOMCAT-GLOMAPmode code has already been adapted to follow an MPI parallelisation method with the horizontal domain being decomposed into regular processor-elements of equal size in longitude/latitude space. The full set of altitude levels are retained on all patches. The model source code is built using the nupdate software, linking to three main code libraries for the TOMCAT, ASAD and BL mixing parts of the model. The main TOMCAT library used in this investigation is version UNICATMPP0.91 . This library already contains several OpenMP directives which were utilised on previous national supercomputing resources (CSAR, HPCx), but are not currently utilised when users run the MPI version of the TOMCAT-GLOMAPmode.

A key finding from the previous HECToR dCSE project which profiled and optimised MPI-TOMCAT-GLOMAPmode was that the code performs better with one MPI task per XT4 compute node (i.e. under-utilising the CPU resource, albeit at four times the charge-rate).

This follow-on dCSE project is driven by the knowledge that recent developments in the HECToR hardware have tended to

(i) be focused on the numbers of cores per chip rather than processor speed (i.e. increased capacity rather than faster cores),
(ii) decrease the available memory per core

By utilising the existing OpenMP directives in the MPI version of the code, the model should be able to use the existing node memory more efficiently and also allow existing communications bottleneck at 64 MPI tasks to share memory over several OpenMP threads to parallelise to a higher number of cores.

1.2 A hybrid model for parallel programming

In this report the meaning of "hybrid mode" is that each node can run one or more instances of MPI tasks (chosen and optimised to match the hardware) and those tasks can spawn additional computational threads (in particular Open MP) on any idle cores on the node. This is sometimes referred to as "mixed-mode". A general working practice for cluster computers is that nodes are usually configured per installation (site specific node configuration) and cannot be varied by the user. The job scheduling software will allow for variations in the use of the installed nodes (as they might not be uniform). The hardware configurations are as varied as a single socket dual-core (HECToR phase 1A), single socket quad-core (HECToR phase 2A) or, in the case of a Cray XT6 (HECToR phase 2B), a node is two sockets with 12 cores each.

The investment required to achieve a hybrid code is moderate as the code structure has to be analysed to avoid contention between MPI tasks and threads. However, the modified code can be used as a normal MPI code if the compiler flags are set to ignore the Open MP directives. This feature was used for comparisons of "single-threaded" runs and to give figures for the production performance. Throughout this work there are references to GMM (GloMAP Mode MPI) and GMH (GloMAP Mode Hybrid) to distinguish between the methods of parallelisation.

1.2.1 Schedules in Open MP

The Open MP directive applied to a do loop is the "PARALLEL DO" directive. This can be followed by any number of keywords known as "*clauses*". The keywords used in this project include "shared", "private", "default", and "schedule". Each program variable must have an explicit declaration that it is private or that it is shared. It is possible to define a default treatment of the variables within the Open MP parallel region. In this project it was decided to use a "default none" to force the explicit declaration of whether a variable is private or whether the variable is shared. The "schedule" keyword has several modifier possibilities: static, (static, block size), dynamic, (dynamic, block size).

For PGI the default block size is:

Block size = (number_of_iterations + omp_num_threads() - 1) / omp_num_threads()

The timings in this report have been gained from the "dynamic, 1" schedule where a thread will work on a loop iteration when it is available and until all iterations are complete.

2 Description of the MPI-TOMCAT-GLOMAPmode software

2.1 Overview of code libraries

The nupdate code libraries for TOMCAT and ASAD used in this project are those produced from the previous HECToR dCSE project -- UNICATMPP0.91X2 and UNIASAD0.2X2. UNICAT0.81 was the last version of TOMCAT that followed a shared memory mode of operation (before the MPI parallelisation method was implemented). This UNICAT0.81 library was used initially in the project with a small case to investigate using OpenMP only on the HECToR system. However, this section of work was limited because a single node of the HECToR XT4h system has only four cores per node allow up to four Open MP threads. However, it was used to better understand the existing OpenMP directives in the code and the resource allocation of PBS, the job scheduling system used on HECToR.

2.2 Tracer advection routines

From sampling analysis, with a single MPI task, four subroutines (ADVX2, ADVY2, ADVZ2 and CONSOM) were identified UNICAT0.81 as places where Open MP treatment was applied, are high in the workload profile. These routines deal with the tracer advection in the model. The Cray PAT analysis (table 2) shows them present in the "Top 5" routines shown in order of percentage time over the whole model run. This section discusses an analysis for four of those routines with examples based on four Open MP threads per MPI task.

For ADVX2 the main outer loop is over L=1, NIV where, in the T42 model, NIV is 31 levels. Using a thread count of 4 the loops will divide into blocks of 8 iterations per thread except one thread, which will do 7 iterations. It is indeterminate which thread works on which loop cycle as the (dynamic, 1) schedule is applied.

For ADVY2, the main outer loop is also over L=1, NIV. This is the same as ADVX2 and so the threads also get 8 iterations of work. There is a feature of TOMCAT that the north-south fluxes across the poles are given special treatment. This is because the face areas on the polar side of the "grid-boxes" tend to zero as the lines of longitude approach the poles. Thus, the 0.91 version of the subroutine is different to version 0.81 as several MPI calls are needed to deal with the Polar Regions. The method that had been implemented (*in a completely separate and long finished project*) in the MPI-only version uses a rudimentary global sum. It occurs within the ***L*** iterations and thus inhibits any Open MP implementation. This code was re-written to move these global sums outside the loop and four extra array accumulators were required to achieve that.

For ADVZ2, the main outer loop is over K=1, MYLAT (and therefore dependent on the MPI decomposition of the computational domain. For the case with 32 MPI Tasks, there are 8 latitudes per patch and there are 32 longitudes cells per patch (table 1). For the case configured for 64 MPI tasks there are 4 latitudes per computational domain. For this resolution and time step, the change in number of MPI tasks is along a line of longitude so increasing the number of processes will reduce the number of latitudes per patch.

The outer loop of CONSOM is over K= 1, MYLAT. However, the Open MP implementation in UNICAT0.81 had been inserted within the K loop and applied to a loop over JV = 1, NTRA (that is 36 for this case). The reason is that early versions of the code read convection coefficients from external files per latitude. The position of this Open MP directive was kept the same. The case using 4 Open M P threads would execute 9 trips per thread which remains the same for any variation in the number of MPI tasks

Table 1: Variation of patch size with domain decomposition, T42 is 128x64x31 computational "grid-boxes"			
MPI tasks	**32**	**64**	**128**
No. Longitudinal intervals per patch	32	32	32
No. Latitudinal intervals per patch	8	4	2
No. Altitude levels per patch	31	31	31
NPROCI, number of PE Task in the longitudinal direction	4	4	4
NPROCK, number of PE Tasks in the latitudinal direction	8	16	32

The NPROC factors are the pre-set selected decompositions for the topology generator, MPI_CART_CREATE. NPROCI x NPROCK equals the number of MPI tasks in total.

2.3 Chemistry and aerosol processes

2.3.1 CHIMIE

The CHIMIE routine is the main routine calling the code to integrate the atmospheric chemistry rate equations (ASAD) and between the code to calculate the aerosol microphysical processes (GLOMAP-mode). These two sub-models for the aerosol and chemistry process the data on a per-grid-box, per-latitude manner. Within the loop over latitudes, arrays on the three-dimensional model grid are copied into planes of longitude and altitude for use in ASAD and GLOMAP-mode. These data are stored within one-dimensional arrays of length "NBOX" which are the form of the input arrays to ASAD and GLOMAP-mode. The Open MP treatment is applied to "loop 6" which is the loop over planes of latitude. Each Open MP thread will process a plane as the loop count increases. For example, with four threads, the first four latitudes are processed by different Open MP threads. The schedule is (dynamic, 1) and thus each thread works on a plane for as long as is needed without delaying the processing of other planes. When a thread has finished work it looks to the loop counter to determine which iteration it must process next. This helps balance out any workload variation between latitudes.

2.3.2 GLOMAP-mode

The GLOMAP-mode code (Mann et al., 2010) consists of a set of subroutines, typically prefixed with "UKCA_" that all work on the newly copied one-dimensional arrays of data for grid-boxes (NBOX). The loop over latitudes in "CHIMIE" extracts the plane of data and passes it to the main routine GLOMAP-mode "UKCA_AERO_STEP". The data are all declared as "private" so each thread can process a plane independent of any other. The side effect is that the stack memory requirement is increased. If the limit is too small then the code will fail. So the OMP_STACKSIZE environment variable was set to 1GB.

2.3.3 ASAD

The ASAD code (Carver et al., 1997) originates from a separate research group (University of Cambridge) and is sourced from a separate code library to the main TOMCAT routines (UNIASAD0.2X2). The existing OpenMP directives in that library have not been changed from the version prior to the version in which MPI was added of the code. It is therefore possible to use UNIASAD0.2 with the Open MP version of TOMCAT in UNICAT0.81. The whole of this chemistry sub-system is processed below the CHIMIE loop over latitudes

(loop 6) and operates on individual grid-boxes. Additional care was needed to confirm that data in certain COMMON blocks was retained per thread using the THREADPRIVATE directive. This is due to the separate initialisation of variables for ASAD.

3 Anticipated Overall Effect of Open MP

A representative selection of timings from the tracing experiments is illustrated in Table 2. The UKCA time has been included within the timing for "CHIMIE". The increased MPI and MPI_SYNC for higher numbers of MPI tasks is an indicator that there is still some improvement to be gained in the MPI section of the code.

Table 2: selected Cray PAT results for fully populated nodes GloMAP mode pure MPI version. This is for three decompositions prior to modification for OpenMP directives.

Configuration	**M32 % of whole sim.**	**M32 % of GMM only**	**M64 % of whole sim.**	**M64 % of GMM only**	**M128 % of whole sim.**	**M128 % of GMM only**
ADVX2	4.2	**5.7**	2.8	**5.5**	1.3	**5.7**
ADVY2	11	**14.9**	7.1	**13.9**	2.2	**9.7**
ADVZ2	4.9	**6.6**	3.2	**6.3**	1.4	**6.2**
CONSOM	5.4	**7.3**	3.6	**7.1**	1.1	**4.8**
CHIMIE	40.9	**55.3**	27.4	**53.7**	12.4	**54.6**
MAIN	7.7	**10.4**	7	**13.7**	4.4	**19.4**
TOTAL FOR GMM	74	100	51	100	22.7	100
MPI	13.3	-	28.3	-	47.4	-
MPI_SYNC	12.7	-	20.7	-	29.9	-

The additional columns in table 2 (labelled GMM only) show the percentage of the timing of the simulation (exclusive of MPI work) that is attributed to the five routines. It shows that CHIMIE is consistently more than 50% of the simulation. ADVY2 has a higher workload due to the extra work for the polar "patches".

The potential speed-up is indicated in **Table 3** by factoring the percentage of original serial processing related to the number of available threads. It forms an analysis similar to the Amdahl factor. In the following equation A is the idealised elapsed time for computation excluding communication time.

$$A = S + \frac{P}{N}$$

In this case the simulation time, *A*, is a sum of the serial time, *S*, plus the time taken in the parallel section, *P*, scaled by the number of threads, *N*. However, the practical case is more complicated because some subroutines will stop scaling when the number of threads is greater than the number of loop iterations. This is shown in table 3 as "*Total for GMM*" where the number of Open MP threads exceed the count of the smallest loop (8 in the M32 example). This idealised calculation does not account for the communication overhead but an indication of an idealised outcome is shown in table 3 as "Total for simulation".

Table 3: percentage of run for 32 MPI tasks mixed-mode an attempt to estimate ideal speed-up

	Max loop count	M32 % of whole sim.	Scaled by number of Open MP threads							
			1 thread	**2**	**4**	**6**	**8**	**12**	**24**	1024
ADVX2	31	4.2	4.20	2.10	1.05	0.70	0.53	0.35	0.18	0.14
ADVY2	31	11	11.00	5.50	2.75	1.83	1.38	0.92	0.46	0.35
ADVZ2	8	4.9	4.90	2.45	1.23	0.82	0.61	0.61	0.61	0.61
CONSOM	36	5.4	5.40	2.70	1.35	0.90	0.68	0.45	0.23	0.17
CHIMIE	8	40.9	40.90	20.45	10.23	6.82	5.11	5.11	5.11	5.11
MAIN	1	7.7	7.70	7.70	7.70	7.70	7.70	7.70	7.70	7.70
TOTAL FOR GMM		**74**	**74.10**	**40.90**	**24.30**	**18.77**	**16.00**	**15.14**	**14.28**	**14.08**
MPI		13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3
MPI_SYNC		12.7	12.7	12.7	12.7	12.7	12.7	12.7	12.7	12.7
Total for simulation		**100**	**100.10**	**66.90**	**50.30**	**44.77**	**42.00**	**41.14**	**40.28**	**40.08**
Speed up			**1**	1.495	1.988	2.234	2.381	2.431	2.482	2.49476

As an explanation on how to read table 3 consider the column for 8 threads: the percentage of the original simulation run (i.e. an MPI run with no Open MP active) for GMH is 16% (a decrease from 74%). An assumption is that the workload in the serial parts of a single MPI task does not change. Another assumption is that communication pattern does not change for the MPI section, those original percentages are added to the new figure of scaled parallel sections. This gives a new percentage figure of 42% of the original, which is translated into a speed-up value of 2.38. This is optimistic as there are sections of the five sub-programs that are serial but not directly measured, so the scaling is an idealised value.

Five places in the code now have OMP PARALLEL DO directives. However, they are not all of equal length (in number of iterations). This means that there is an upper limit for the speed-up of the loops with fewer iteration counts i.e. if there are eight loop iterations then adding more than eight threads will not share the work any more than using eight threads. The excess threads will be in a WAIT state. If the loop limit is not exactly divisible by the number of threads then some threads will do more work. An example for 32 MPI tasks is given in table 4.

Table 4: number of trips for each thread for a case with 32 MPI tasks.						
32 MPI tasks	**Loop length**	**2 threads**	**4 threads**	**6 threads**	**8 threads**	**24 threads**
ADVX2	31	16 (15)	8 (7)	6 (5)	4 (3)	2(1)
ADVY2	31	16 (15)	8 (7)	6 (5)	4 (3)	2(1)
ADVZ2	8	4	2	2 (1)	1	1
CONSOM	36	18	9	6	5 (4)	2(1)
CHIMIE	8	4	2	2 (1)	1	1

In table 4 the bracketed numeral is a remainder value where the loop limit is not exactly divisible by the number of threads.

4 Results

4.1 Overview of results section

This section describes the baseline operation of purely MPI with results for 32, 64 and 128 MPI task decompositions. That is followed by the presentation of the results for the mixed-mode operation for 32 and 64 MPI tasks. There is a section for the work on the Cray XT4h and then a section for the work on the Cray XT6.

The result of using the code in its current production form is presented alongside the results of the hybrid configurations (references to GMM in the figures). A systematic labelling was decided upon, to show what the configuration for a particular simulation is. Lower case **n** is the total number of MPI tasks and represents the geometrical decomposition of the computational domain. Upper case **N** is the number of MPI tasks per physical node (as understood by the PBS job scheduler). Lower case d is the number of MPI Open MP threads per MPI task (indicating “depth”). An additional parameter is provided for the XT6; upper case **S** is the number of MPI tasks per hex-core on the physical node.

For example, **n64N4S1d4** is the code for a simulation using 64 MPI tasks with 4 MPI tasks per node and 1 MPI Task per hex-core and using 4 threads (Open MP specifically) per MPI task. Occasionally another label for number of threads is “t”. The “d” is also a flag option for *aprun* that indicates the MPI tasks should be spread out by the indicated number in terms of cores. So “d4” implies that the MPI tasks are on every fourth core.

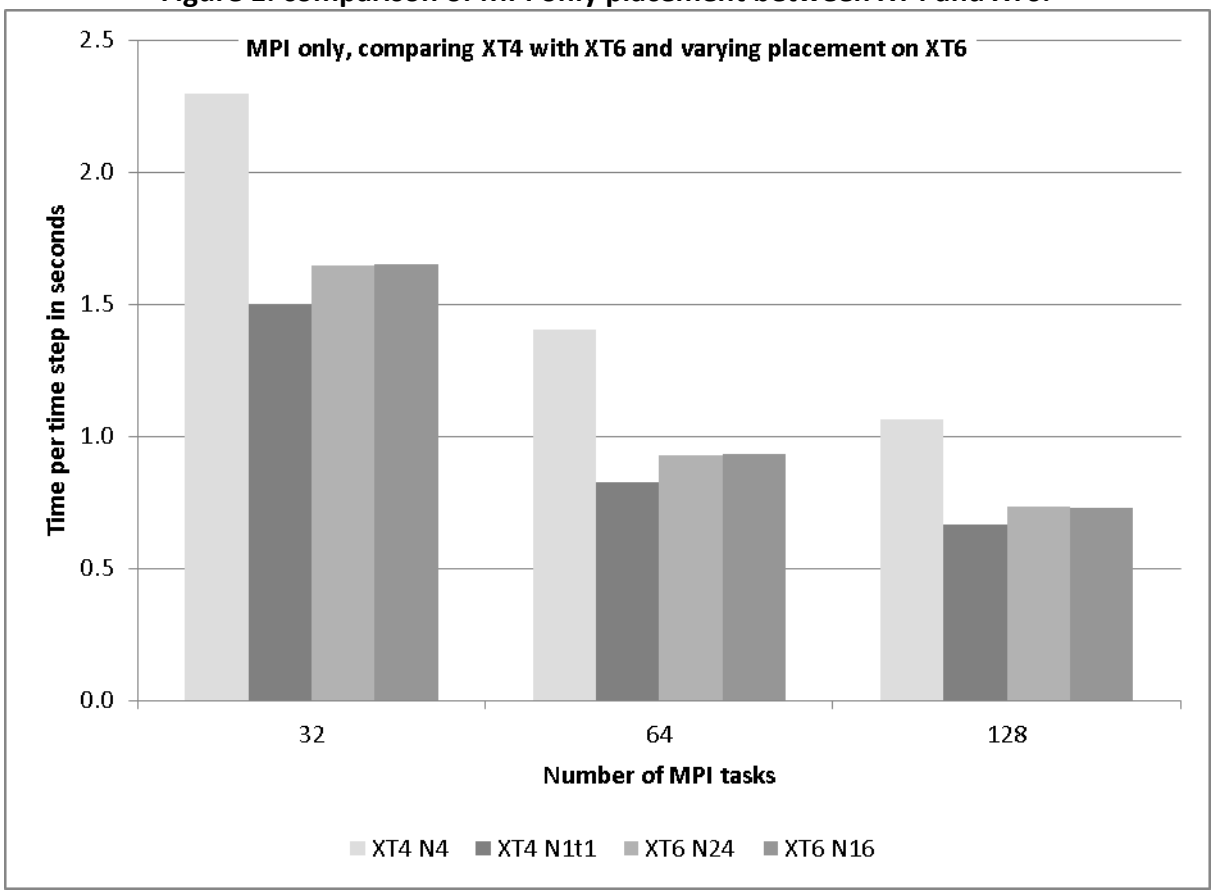
4.1.1 Case description

The time step in this simulation is 1800 seconds and the T42 resolution grid was used. In this work there are frequently two executables of the code in use. One has been compiled for MPI-only i.e. it is the same source code but the compiler flag for activating Open MP directives has not been set. The second executable is the hybrid version where the same source has been compiled with the addition of the Open MP directives option. It is the working practice of the GloMAP research group to compile the code specifically for the number of MPI tasks to be used immediately before a simulation is submitted. The domain decomposition is geometric and the T42 grid (128x64x31 grid-boxes) is partitioned into a rectangular MPI topology (4x16). Each patch (32x4x31 grid-boxes) is half the size of a patch in the configuration for 32 MPI tasks.

4.1.2 MPI only

This mode has been extensively investigated in an earlier DCSE programme. This section shows the comparison of the XT6 and XT4 and whether any XT4 results read across to the XT6 system. Figure 1 shows the effect of placement of MPI-only on the XT4 compared to the XT6. The difference between the two placements for XT4 is that the case of N1d1 gets the whole memory (8GB) and there 3 idle cores on the node. The cases are compared by spreading out the job for the XT6 with four MPI tasks per hex-core die. The deduction is that the MPI intra-node communications for the XT6 are handled very well because the fully packed configuration is as fast as less dense configuration. There will be other factors contributing to the difference between the runs on the different systems such as, the hardware, memory-caches, clock-speed and optimisations available to the compiler.

Figure 1: comparison of MPI only placement between XT4 and XT6.



The difference between the "spread-out" placements on the XT6 is not as great as that seen when spreading out the tasks on the XT4. The single task per node on the XT4 will have access to 8GB memory whereas a task on the XT6 when limited to 16 tasks per node (i.e. four MPI tasks per hex-core-die) will only see 2GB per task.

4.2 The results for work on the Cray XT4

Previous work has shown that reducing the number of MPI tasks per node allows the speed of computation to increase. The subsequent additional improvement through the use of Open MP threads is shown in table 5 and as a column chart in figure 2. The shaded values in table 5 demonstrate the improvement gained by using 4 OMP threads compared to using four times the number of MPI tasks (i.e. 128 cores in use). In the case of 32 MPI tasks and

where 4 Open MP threads (a total of 128 cores) are in use there is approximately 3x speed increase whereas operating in a normal production manner (i.e. four MPI tasks per node) and using 128 cores results in only a 2x improvement. A similar observation can be made for the case where 16 MPI tasks are accelerated by 4 threads although the improvement is not as marked. The configuration for 128 MPI tasks also benefits from the 4 Open MP threads. In all cases not all of the time improvement is due to the Open MP threads as the time per step is much shorter when the MPI job is spread out with one task per node.

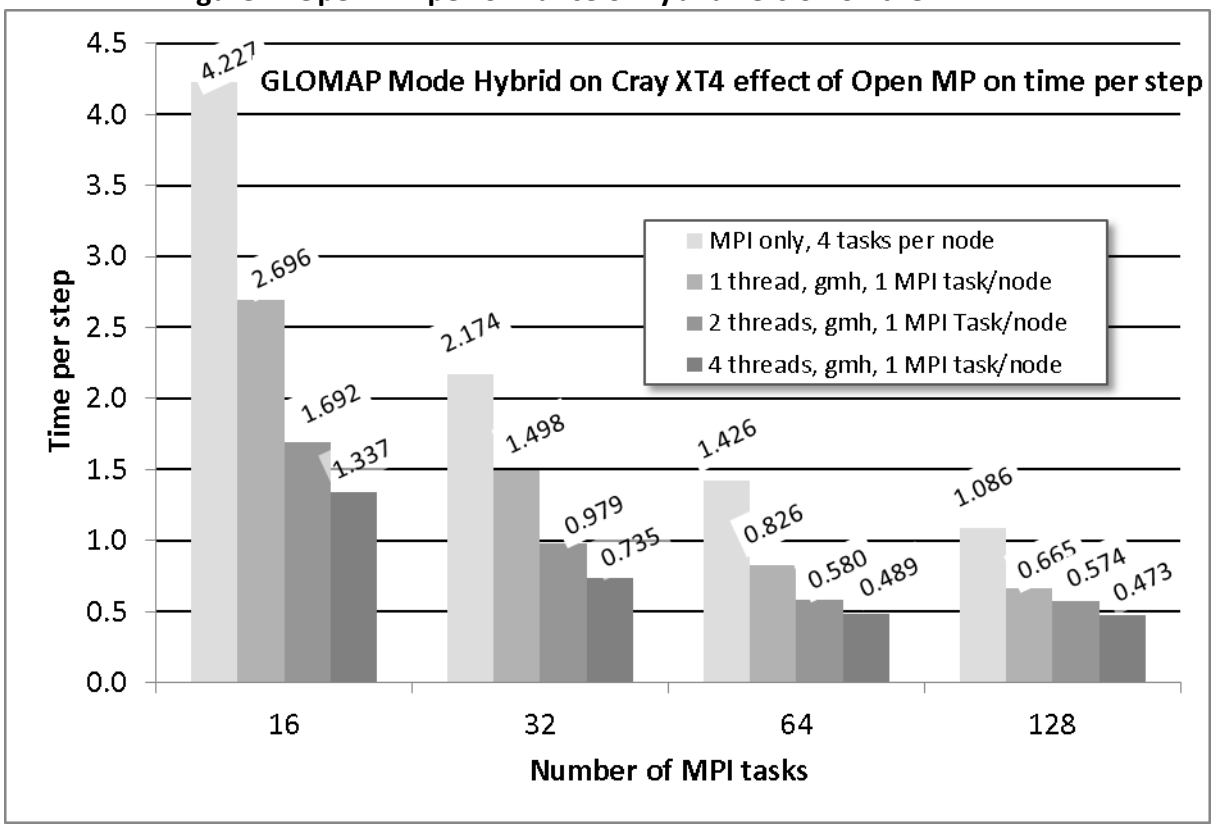
Table 5: time per step for GLOMAP Mode: MPI version and Hybrid version

Number MPI tasks	**16**	**32**	**64**	**128**
XT4h GMM	4.227	***2.174***	1.426	***1.0858***
XT4h GMH N1t1	2.696	1.498	0.826	0.6652
XT4h GMH N1t2	1.692	0.979	0.58	0.5745
XT4h GMH N1t4	1.337	***0.735***	0.489	0.473

A graphical representation of this table is given in figure 2, in which there of four groups of columns based on the MPI configuration. The first column in each group is for the normal mode of operation with 4 MPI tasks per node. The remaining three columns in the groups are for a single MPI task per node along with 1, 2 and 4 threads and indicate the efficiency of Open MP implementation.

The 32 MPI task configuration is the current preferred mode as only a 2x speed up can be achieved when using 128 MPI tasks even though it uses four times the resource i.e. this route is twice as expensive. However, the hybrid mode achieves almost three-fold speed-up for the same charge rate (i.e. using 32 nodes) which makes this route only 30% more expensive. If the 2 thread case is considered then the speed-up are very close to those shown in the idealised calculation of table 2, i.e. 2 threads give 1.4x speed-up and 4 threads give 2x speed up. Considering that a 15 month simulation can take 12 hours of computation, with the current working practice Open MP allows a researcher to do more simulations or even choose to run a simulation for longer (say 36 months).

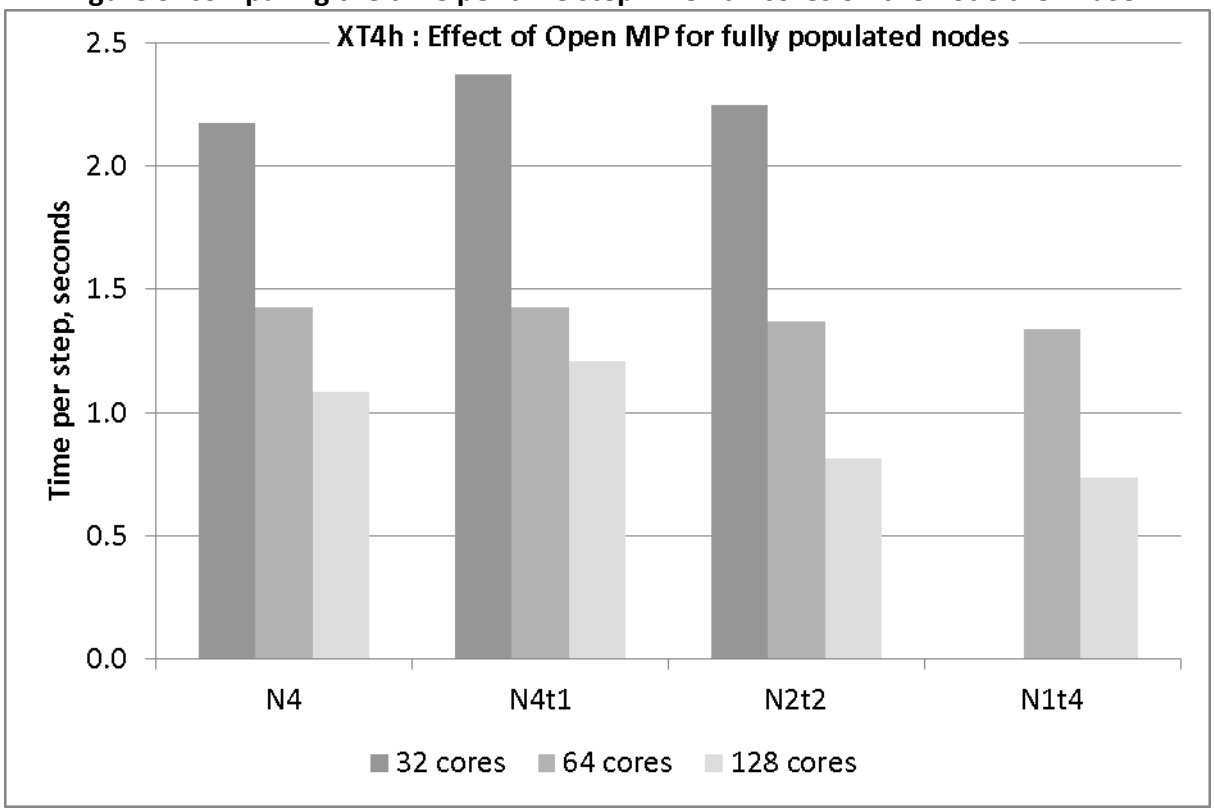
Figure 2: Open MP performance of hybrid version on the XT4h



Another characteristic that has been observed is that for lower number of MPI tasks the effect of Open MP acceleration is not as beneficial as for the simulations with the configurations that use a higher number of MPI tasks. The experiment was carried where all cores per node were in use but some configurations were hybrid. **Figure 3** shows four groups of columns that are grouped by parallel configuration. The first group is MPI only, the second group is the hybrid version running with a single thread, the third group is the hybrid version using two threads (so two MPI tasks per node) and the final group is the hybrid version using four Open MP threads. The Open MP directives will affect the optimisation of certain loops thus making the hybrid code slightly slower. This is seen in the difference between the first and second groups.

The columns within each group each represent a different MPI configuration (i.e. 16, 32, 64, 128 MPI tasks) and therefore indicate the MPI scaling. The greatest benefit from Open MP acceleration is seen for the case of 128 cores-in-use. Reducing the number of MPI tasks per node results in a shorter time per time step. Adding Open MP threads provides a further improvement in speed. There is less of a difference between the configurations that use 64 cores although there is a slight reduction in time step associated with the reducing number of MPI tasks. The charge rate is per node-hour the comparison of the time per step between the bars of one shade will be reflected directly in the cost of a simulation.

Figure 3: comparing the time per time step when all cores on the node are in use.



4.3 The results for work on the Cray XT6

4.3.1 Preparing for mixed mode on Magny-Cours

Several preliminary runs were carried out partly to get familiar with the new operating environment and also to determine the flexibility of controlling task placement on the node. The complexity of the node (that is two Magny-Cours processors) provides some unbalanced paths to memory and it will be important for users to pro-actively place jobs (rather than rely on system defaults) to maximise the return on their resources. For the XT6 experiments with GloMAP, the following variable parameters were available:

i. MPI tasks : n=32 and n=64 (as these are target production configurations)
ii. Number of MPI tasks per node: N = 1,2,4,8,12,24
iii. Tasks per hex-core die: S = 1,2,3,4,5,6
iv. Number of Open MP threads per MPI task: d= 1,2,3,4,6,12,24

As detailed in the earlier section on code analysis, the Open MP parallel loop count is not consistent throughout the program and so the gains are expected to be less significant for higher thread counts. The preliminary runs included the timing for the purely MPI version configured as fully packed.

To enable Open MP the executable is built using the “-mp” flag. This changes the manner in which the built-in optimisers can treat blocks of code. For example, there is a section in “CONSOM” where the Open MP is *within* the outermost do-loop (explained in an earlier section). Therefore, whatever optimisation could have been applied to the outer loop during the compilation is likely to be inhibited when the Open MP directives are activated. The MPI-only run is compared, in figure 4 and figure 6, with a mixed-mode run where

"OMP_NUM_THREADS" is set to unity. Typically the single threaded mixed mode run is slightly slower.

The test case used with this project (T42 grid) has a computational domain is organised by powers of 2 i.e. 128 Longitudes by 64 Latitudes. This has been convenient previously for one-core-per-node HPC systems that often used powers of 2 for arranging the nodes. However, the recent configuration of many-cores systems is not necessarily bound to follow that practice as seen in the already described XT6 with its 24 cores per node.

In the case of 32 MPI tasks this uses two nodes and leaves 16 cores idle and in the case of 64 MPI tasks using 3 nodes leaves only 8 cores idle. The consequence is the nonlinear characteristic of the cost per time step; this is due to the charging method being on a nodal basis.

Users have generally been advised to associate the options of the aprun command with the PBS resource requests using environment variables and the qstat command. This is no longer correct for the mixed-mode operation as the "-d" option for aprun is used to allocate cores for the Open MP threads. It is better to factor the mppwidth option for PBS so that it is a multiple of actual cores per node to ensure enough cores are allocated. For example, mppwidth should be set as 192 for 32 MPI tasks and 6 Open MP threads (8 nodes). The subsequent aprun command will specify the run exactly, including MPI task placement within the set of allocated nodes. For the example of 32 MPI tasks the command will be

aprun –n 32 –N 4 –S 1 –d 6 ./glomap.exe

The job submission script will need the OMP_NUM_THREADS to be set and that environment variable can be used in the aprun command as the value supplied for the "-d" option. So the example now looks like:

aprun –n ${NTASK} –N ${NTASKPN} –S 1 -d ${OMP_NUM_THREADS} ./glomap.exe

where the additional values NTASK and NTASKPN are set independent to the PBS resource.

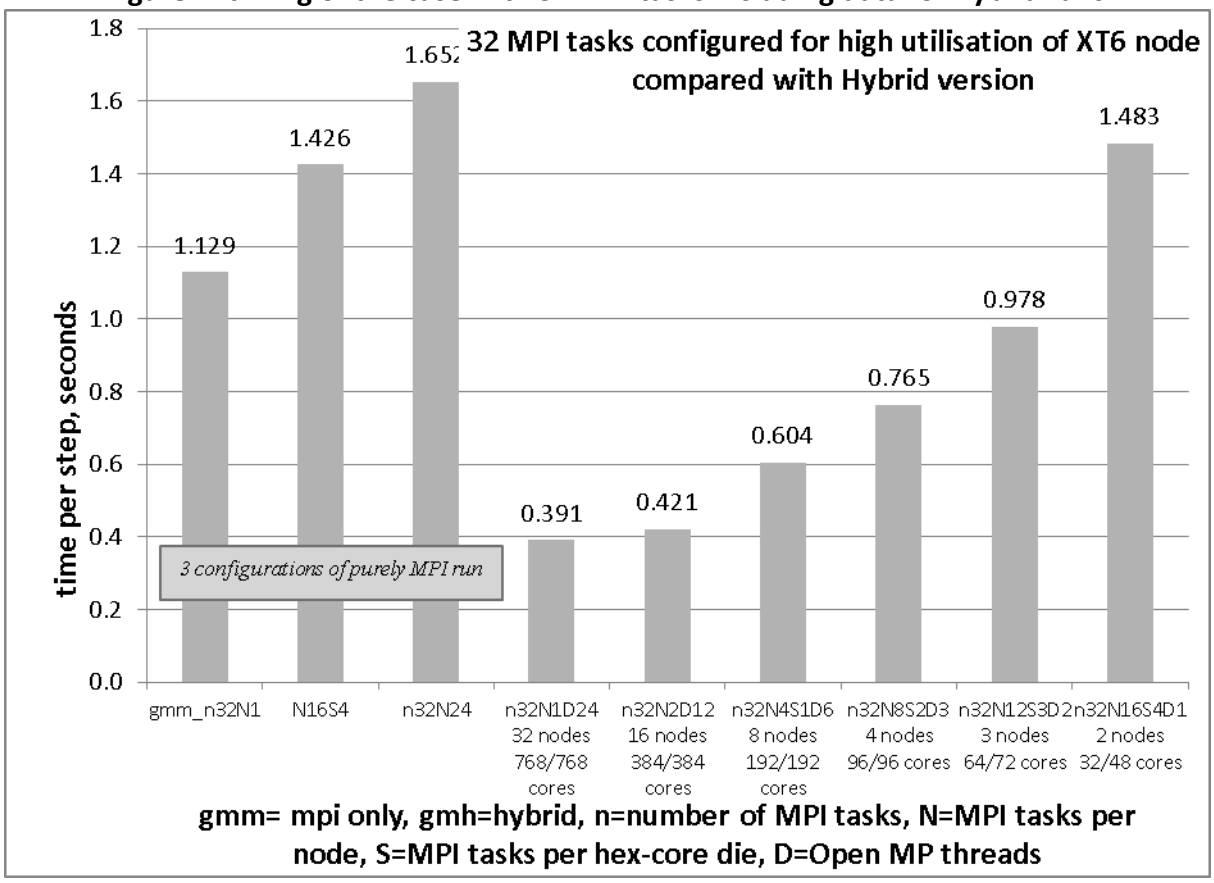
4.3.2 Results for MPI-only mode on Cray XT6

For the MPI-only mode of operation there is an optimal configuration for each of the 32 tasks and 64 tasks cases. This is based on AU per time step because the distribution of tasks among the nodes will leave idle cores (that still incur charges) and the run-time may be affected by the differing use of cache and paths to memory. Each of the hex-core dies has access to the nodal memory of 32GB of RAM, thus, the memory available per core reduces from 2GB to 1.3GB when compared to the XT4h.

In the case of 32 MPI tasks this uses two nodes and leaves 16 cores idle and in the case of 64 MPI tasks using 3 nodes leaves only 8 cores idle. The consequence is the nonlinear characteristic of the cost per time step; this is due to the charging method being on a nodal basis.

It would be better for the total number of MPI tasks to be a multiple of 24 otherwise there will be idle cores on the final node. For example, with the case configured for 32 MPI tasks will, by default, place 24 tasks on the first node and 8 tasks on the second node. The additional cost of 16 idle cores can be mitigated by distributing the tasks evenly among the eight hex-core dies, leaving only 8 cores per node idle. The reduced cost is achieved because the time per step is improved and the overall runtime is reduced. In figure 4 the first three columns demonstrate the effect on run-time for the extreme cases of 2 nodes against 32 nodes (i.e. one task per node). The remaining 6 columns are discussed in the section on hybrid operation.

Figure 4: timing of the case with 32 MPI tasks including data for hybrid runs



For the experiment using 64 MPI tasks, the minimum number of nodes would be 3, i.e. for 72 reserved cores this implies 8 idle cores. There are two series shown in figure 5; AU per time step and time per time step. The order of the entries is sorted by time per step (the second series). The left-most entry is for the highest density of packing MPI tasks; i.e. fully loading the first two nodes and leaving 8 cores idle on the third node.

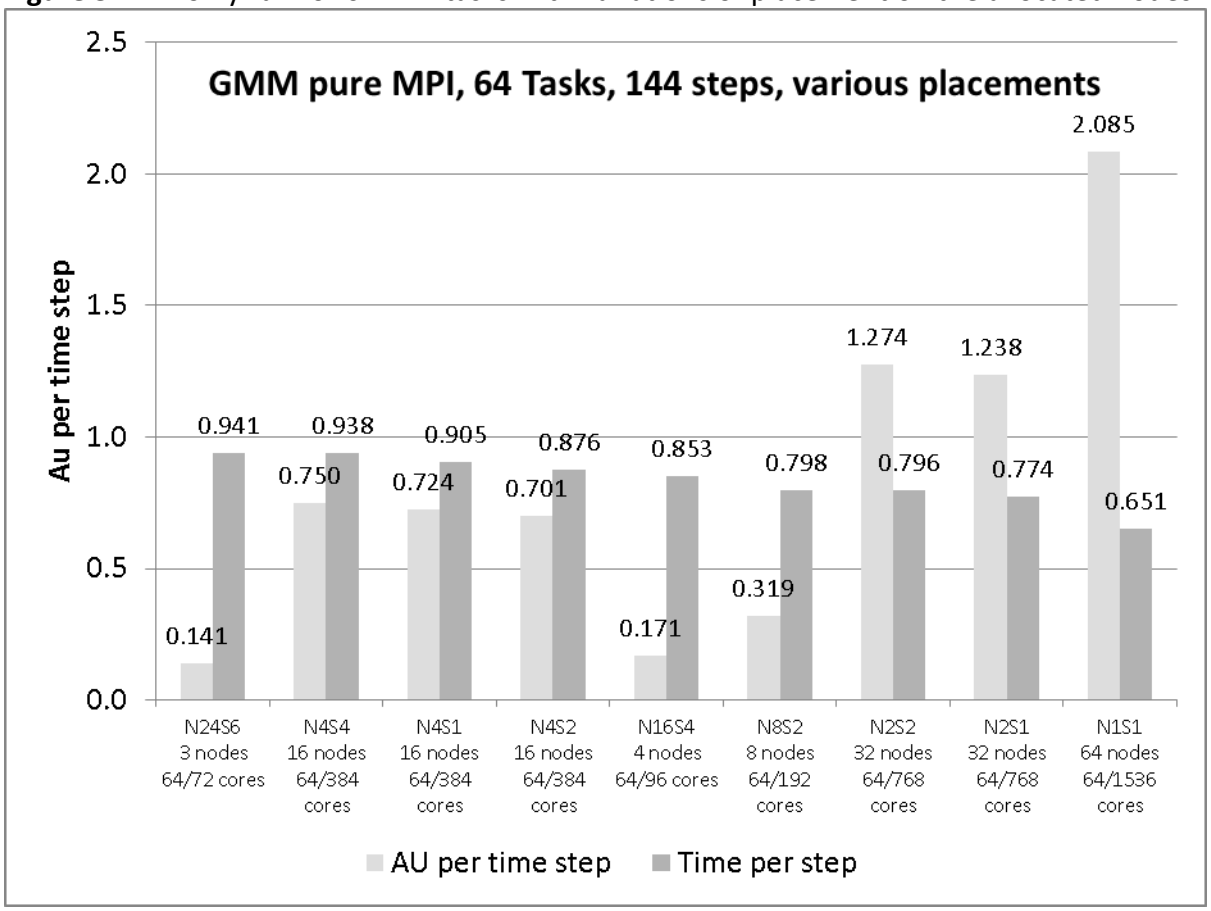
Distributing the 64 tasks over the hex-core dies in an even manner will introduce more idle cores as an extra node is required (i.e. another 24 cores thus a total of 32 surplus cores). This raises the cost per time step unless there is a corresponding improvement in performance from the new distribution. The results show that a naïve placement of 64 MPI tasks on packed nodes (as per normal production runs) provides a more economical use than the uniformly distributed job (n64N16S4).

For the latter the time per step has been reduced by 10% but the charge rate is increased by 30% due to the requirement for a fourth node.

In figure 5 the column pairings 2, 3 and 4 show the case with only four MPI tasks per node (n64N4) but with variations in placement of; 4 MPI tasks per hex-core; 2 MPI tasks per hex-core and 1 MPI task per hex-core. For each configuration there are 16 nodes and thus the same number of inter-node communications and identical cost per minute (AU charge rate). However, the route off the node has differing paths for data communication. The time per step is smallest where all the MPI tasks are on the same socket (a Magny-Cours processor) but spread out with 2 tasks per hex-core die. The case where all four MPI tasks are on the same hex-core die is slowest. This indicates that there may be contention for the MPI transport between nodes; it is more likely to be due to the limited local memory and all tasks needing to share the cache. For the case with one MPI task per hex-core die there is

intra-node communication between sockets that could cause some overhead. This could be attributed to the MPI communication routes between hex-core dies.

Figure 5: MPI only run for 64 MPI tasks with variations of placement on the allocated nodes.



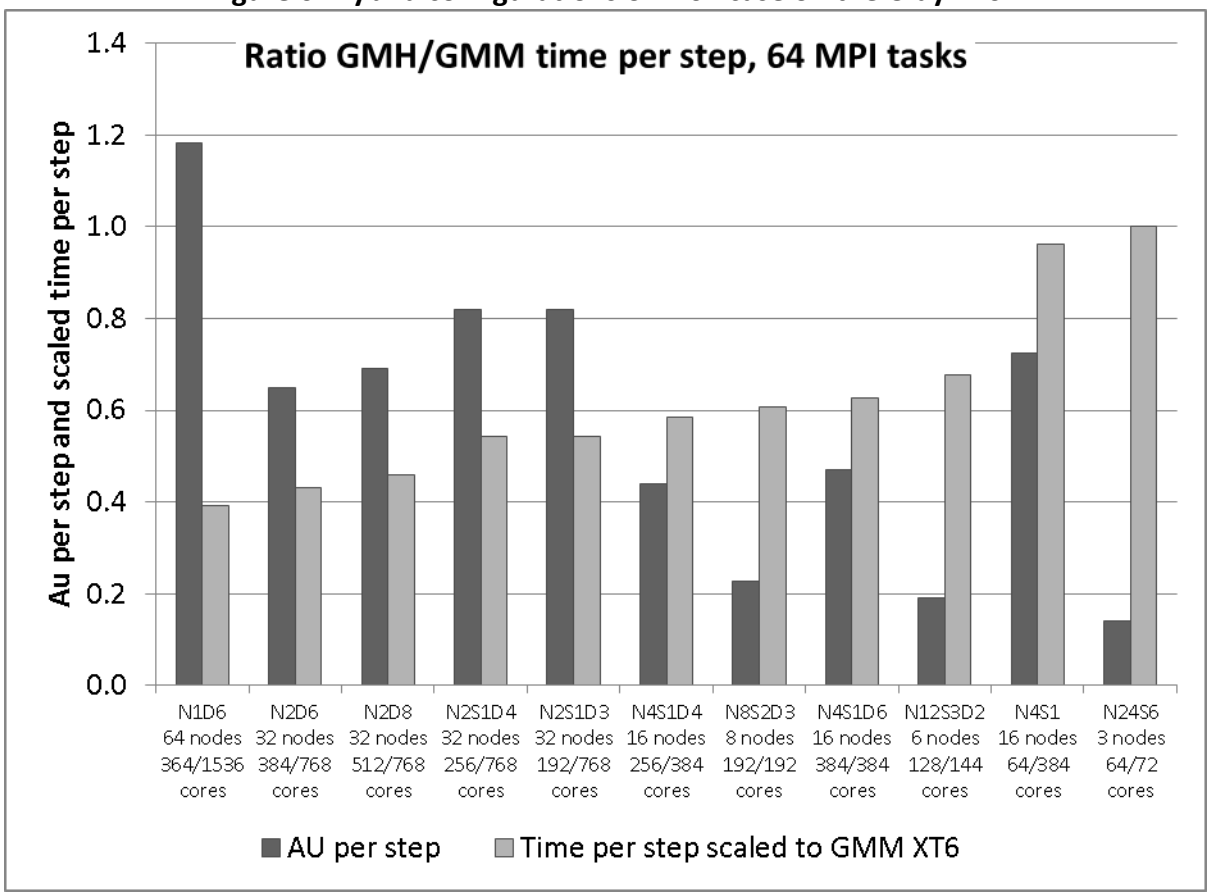
The fraction of 'cores used' against 'cores available' is included in the labelling of each entry. The scale has the same order of magnitude for seconds per time step (the second column per entry) as for the cost per time-step.

4.3.3 Performance of Hybrid version on Cray XT6 with 64 MPI tasks

The case with 64 MPI tasks requires a minimum of three nodes and with the high density packing of MPI tasks, there are eight unused cores that had to be reserved for the run. Spreading out the job so that there are only four MPI tasks per node and only one MPI task per hex-core leaves room for the OMP_NUM_THREADS=6. In this configuration there would be 8GB of local memory per MPI task. However, this is seen to be only 40% faster (in figure 6 the N4S1D6 uses 384/384 cores but takes slightly more than 60 per cent of the reference time.

For this decomposition there are only 4 grid-boxes of latitude per MPI task and so for the major work loop (K=1,NLAT), within the CHIMIE subroutine only uses 4 threads, the two extra threads occupy memory but do not contribute to any calculations.

Figure 6: Hybrid configurations of M64 case on the Cray XT6



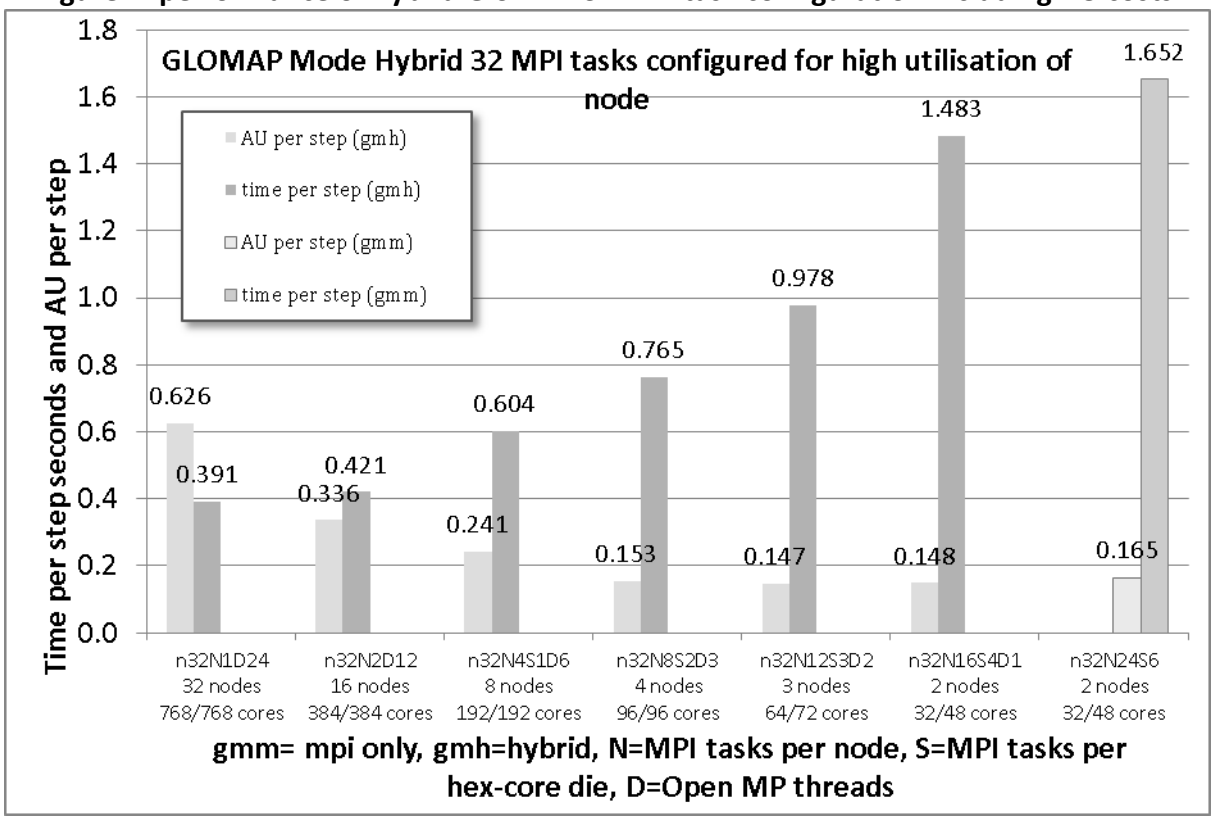
The two columns at the right of the chart in figure 6 are the non-hybrid run. The timing figures for the other conditions are scaled to this time. It shows that all the tested configurations do take less time per time step but with variation in the cost per step.
The four columns associated with the "N2" run illustrate an effect of OpenMP with a strange result that the case with 8 threads is slower the case with 6 threads. This may be attributed in part to the fact that there is not enough work to be distributed evenly among the threads and the additional threads compete for memory.

4.3.4 Performance of Hybrid version on Cray XT6 with 32 MPI tasks

Figure 7 shows how the hybrid code performs for various settings of the number of OMP threads. The increase of number of OMP threads consistently improves the time per time step to the maximum achievable of 24 threads where there is only one MPI task per node. The increasing cost of spreading out the MPI task has a greater rate than the reduction in time gained by adding Open MP threads. This is mainly due to matching the problem size to the number of threads.

The result is that, for this case on the XT6, 1.65 seconds per time step, for the MPI-only case is already 1.3x faster than the 2.17 seconds per time step seen for the same configuration on the XT4h. Further gains can be made by choosing a better placement based on the numbers of nodes reserved for the run. This is demonstrated by running the hybrid code with only one Open MP thread per MPI task. The second group from the right is n32N16S4D1, provides the quickest timing at 1.48 seconds per step.

Figure 7: performance of hybrid GloMAP 32 MPI task configuration including AU costs



The activation of 2 Open MP threads results in a shorter time step that is 1.69x faster than the reference run at a cost per time step that is 90% of the reference MPI-only run. A speed-up of 2.1x can be achieved with 3 Open MP threads. There is an associated reduction in cost of 7%. When compared with the better placed single-threaded configuration these figures change to being a speed up of 1.27x for an increase in cost of only 4% more. This is still a beneficial configuration and should be considered for normal production runs on the XT6 system for the cases using 32 MPI tasks.

4.4 Discussion

The use of Open MP within this software project has been to accelerate only five do-loops. The length (iteration count limit) of the loop will affect the efficiency of the implementation. If the number of trips around a loop is not an integer multiple of the number of threads available for work, then there will be a possible condition where some threads are idle while others are still working. The clause “dynamic” goes some way to alleviate that imbalance but only if the work per loop cycle is unequal.

In the whole program there are two places where Open MP directives have been placed around loops that iterate over the latitudes, one in CHIMIE and the other in ADVZ2. They will not benefit for simulations in which there are more threads than latitudes per patch. The major work of GLOMAP (within CHIMIE) accounts for approx 55% of the runtime so the best that section can be accelerated is equal to the number of latitudes within the patch.

The other three sections of code where Open MP is applied have of the order 32 iterations which are equal to the number of cores per node on the XT6. These sections are a smaller proportion of the “serial” runtime (in a per MPI-task manner) and thus the gains will

not have as great an effect on the timing of the simulation as achieved by the CHIMIE section of work.

The performance of Open MP is such that it will help with the normal operation of this application if the number of threads is chosen to match the number of latitudes in the "patch". The T42 case is limited by the decomposition method to a maximum number of 128 MPI tasks (typical usage is 32 MPI tasks) where there are only 2 latitudes per patch. The hybrid version of GLOMAP Mode has shown that it is now possible for the researchers to use more cores than previously was economically possible i.e. use 128 cores for a simulation with 32 MPI tasks.

The XT6 system will soon become available and that will mean that simulations will run with a higher number of tasks per node than for the XT4h. Using a sparse placement of MPI tasks will allow the code to run more efficiently. The work with the XT6 was brief due to limited access time. However, it was possible to determine an optimal mode of operation for the two MPI decompositions (per node this means 8 tasks with 3 threads AND 12 tasks with 2 threads).

5 Summary

In this project, existing OpenMP directives in the TOMCAT-GLOMAP code have been updated and re-designed for use in the MPI version of the code that has already been ported and optimised on HECToR as part of a previous HECToR dCSE project. This new mixed-mode parallel version of the code has been implemented and tested on both the UK HECToR Cray XT4 system and the second-phase HECToR Cray XT6 system during the early-access testing phase.

The implementation has been successful and has led to substantial performance enhancements as the idle cores on a node can be loaded with work from Open MP threads and reduce the time per iteration of the simulation. The practice of distributing the MPI tasks sparsely has arisen due to the increase in resolution and complexity of simulations. The extra Open MP threads do require extra memory but not as much as replicating the whole code would have required with a dense distribution of MPI tasks.

The analysis showed that there was likely to be a maximum of 2.5 times speed-up and values close to this are demonstrated in the results for this project, particularly on the XT4 architecture. Previously the researchers would rarely try to use more than 64 MPI tasks but this method has shown benefits for configurations of 32 and 64 MPI tasks. The greatest benefit was seen on the XT4 with 32 MPI tasks. For the XT6, there is more complexity on the node with a choice of distribution of the tasks and threads.

Of the original plan only task 1 was addressed for this project. It has 3 sub-tasks which at the time were targeting the dual core and quad core architectures. In the time scale of the project the 24 core XT6 became available and made the analysis more difficult if it was to cover the full extent of the possible combinations of task placement.

5.1 Acknowledgements

This project was funded under the HECToR Distributed Computational Science and Engineering (CSE) Service operated by NAG Ltd. HECToR – A Research Councils UK High End Computing Service - is the UK's national supercomputing service, managed by EPSRC on behalf of the participating Research Councils. Its mission is to support capability science and engineering in UK academia. The HECToR supercomputers are managed by University of

Edinburgh, HPCx Ltd and the CSE Support Service is provided by NAG Ltd.
http://www.hector.ac.uk

References

Carver, G. D., Brown, P. D., and Wild, O.: The ASAD atmospheric chemistry integration package and chemical reaction database, Comp. Phys. Comm., 105, 197–215, 1997.

Chipperfield, M. P.: New version of the TOMCAT/SLIMCAT offline chemistry transport model, Q. J. Roy. Meteor. Soc., 132, 1179–1203, 2006.

Mann, G. W., Carslaw, K. S., Spracklen, D. V., Ridley, D. A., Manktelow, P. T., Chipperfield, M. P., Pickering, S. J., and Johnson, C. E.: Description and evaluation of GLOMAP-mode: a modal global aerosol microphysics model for the UKCA composition-climate model, Geosci. Model Dev., 3, 519–551, doi:10.5194/gmd-3-519-2010, 2010.