High Performance Computing Driven Software Development for Next-Generation Modelling of the Worlds Oceans

Xiaohu Guo[a]*, Gerard Gorman[b], Mike Ashworth[a], Stephan Kramer[b],
Matthew Piggott[b], Andrew Sunderland[a]
[a]Science and Technology Facilities Council, Daresbury Laboratory,
Daresbury Science and Innovation Campus, Warrington, Cheshire WA4 4AD, UK.
[b]Department of Earth Science and Engineering, Imperial College London,
London SW7 2AZ, UK.

ABSTRACT: The Imperial College Ocean Model (ICOM) is an open-source next generation ocean model build upon finite element methods and anisotropic unstructured adaptive meshing. Since 2009, a project has been funded by EPSRC to optimise the ICOM for the UK national high-end computing resource, HECToR (Cray XT4). Extensive use of profiling tools such as CrayPAT and Vampir, has been made in order to understand performance issues of the code on the Cray XT4. Of particular interest is the scalability of the sparse linear solvers and the algebraic multigrid preconditioners required to solve the system of equations.

KEYWORDS: ICOM, parallel, unstructured mesh, ocean model

1. Introduction

The Imperial College Ocean Model (ICOM) has the capability to efficiently resolve a wide range of scales simultaneously. This offers the opportunity to simultaneously resolve both basin-scale circulation and small-scale processes such as boundary currents, through-flows, overflows and geostrophic eddies. For Earth system and climate modellers, the underlying numerical software technology offers the opportunity to focus resolution in regions of particular importance, such as boundary currents and overflows, without increasing the computational cost above that of a conventional coarse-resolution model. As the climate change research agenda moves to addressing impacts and considering how to adapt to them, fully integrated models that can address interactions between global and localised phenomena will need to be developed. Therefore, there is an urgent need to improve both their fidelity and their capabilities to provide more confident assessments at small, regional and global scales. Furthermore, the development of such a model will have a wide range of applications in other related areas, such as ocean forecasting, flood defence, pollution/contaminant dispersal, sustainability of water quality and fisheries.

ICOM is build on top of Fluidity, an adaptive unstructured finite element code for computational fluid dynamics. It consists of a three-dimensional non-hydrostatic parallel multiscale ocean model, which implements various finite element and finite volume discretisation methods on unstructured anisotropic adaptive meshes so that a very wide range of coupled solution structures may be accurately and efficiently represented in a single numerical simulation without the need for nested grids [1, 2, 3].

Developing an unstructured mesh ocean model is significantly more complex than traditional finite difference models. Apart from the numerical core of the model, significant effort is also required to develop pre- and post-processing tools as there are no standards yet established within the community. For example, initial mesh generation must confirm to complex bathymetry and coastlines, which exerts a critical influence over the dynamics of the ocean. In practice, there is a trade-off between how close the discretised domain is to reality, and how appropriate it is for numerical simulation with limited computational resource. To achieve this, specialised mesh generators such as Terreno [4] have been developed as the geometries do not lend themselves to standard packages which use CAD models. A typical bathymetry conforming unstructured mesh is presented in Figure 1.

ICOM can also optimise the numerical mesh to control modelling error estimates using anisotropic mesh adaptivity. Large load imbalances are to be expected following the mesh adaptivity, therefore dynamic load-balancing is required. When rebalancing the mesh a combination

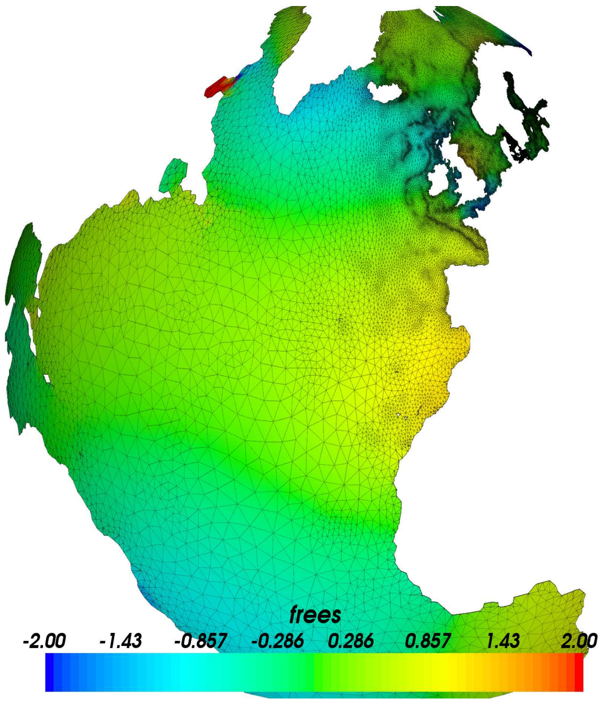

Figure 1: Unstructured meshes are an ideal choice for representing complex problem domains

of diffusion repartitioning, and clean parallel repartitioning with domain remapping, is used to minimise data migration. While this is currently done using a combination of ParMETIS and a bespoke code for data migration (see [5]), work is being done on integrating Zoltan [10] instead.

Solving sparse linear equations is one of the most time consuming parts in ICOM. For systems of equations with low/moderate condition number, ICOM uses standard preconditions and solvers from PETSc [9]. When the condition number is high ICOM use an AMG preconditioner designed specifically for this problem which outperforms all conventional preconditioners and black box AMG preconditions tested (*e.g.* Prometheus and HYPRE/BoomerAMG). However, attention must always be paid to the scaling as coarsening the matrix (reducing the size of the problem) makes scaling more difficult.

ICOM use state-of-the-art and standardised 3rd party software components whenever possible. For example, PETSc [9] is used for solving sparse linear systems while Zoltan [10] is used for many critical parallel data-management services both of which have compatible open source licenses. Python is widely used within ICOM at run time for user-defined functions and for diagnostic tools and problem setup.

The present paper mainly considers scaling ICOM on HECToR for an non-adaptive simulation. The detailed profiling and performance analysis are described in the following.

2. Benchmark Test Cases and Numerical Background

2.1 Wind-driven baroclinic gyre

A gyre in oceanography is a large system of rotating ocean currents, particularly those involved with large wind movements. The flow is dominated by a balance in the horizontal between the Coriolis force and the free surface gradient. In a baroclinic gyre a density stratification, typical for the ocean domain, is taken into account. The equations used by ICOM to model this configuration are the 3D non-hydrostatic Boussinesq equations. In a domain $V \subset \mathbb{R}^3$ these take the form:

$$\frac{D\boldsymbol{u}}{Dt} + 2\boldsymbol{\Omega} \times \boldsymbol{u} - \nabla\cdot(\boldsymbol{\nu}\cdot\nabla u) + \tag{1a}$$

$$+\nabla p + g\nabla\eta = f_{wind} + \rho g\boldsymbol{k}, \tag{1b}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{1c}$$

$$\frac{DT}{Dt} + \nabla\cdot(\boldsymbol{\kappa}_T\nabla T) = 0, \tag{1d}$$

$$\rho \equiv \rho(T), \tag{1e}$$

where $D/Dt = \partial/\partial t + \boldsymbol{u}\cdot\boldsymbol{\nabla}$ is the total derivative, p is the perturbation pressure, g is the acceleration due to gravity, η is the free surface height, ρ is the perturbation density and f_{wind} is the wind forcing term. Rotation of the Earth is taken into account via the vector $\boldsymbol{\Omega} = (0,0,f)^T$, where $f = f_0 + \beta y$, the so called beta-plane approximation. T is the temperature, but additional scalar fields such as salinity and tracers can be dealt with analogously, in which case they may be included in the equation of state (1d). A diagonal viscosity tensor $\boldsymbol{\nu}$ is used to represent (effective) stresses in the model. $\boldsymbol{\kappa}_T$ is the thermal diffusivity tensor, also assumed to be diagonal. No turbulence models have been applied in this benchmark.

A linear stratification has been used as an initial condition for the temperature:

$$T = T_{\text{surface}} + \frac{z\Delta T}{H}, \tag{2}$$

with a surface temperature of $T_{surface} = 20$ and a temperature difference $\Delta T = 10$. The vertical coordinate z is chosen such that $z = 0$ at the surface and $z = -H = -2000\,m$ at the bottom. A simple linearised equation of state is used:

$$\rho(T) = \rho_0\left(1 - \alpha(T - T_0)\right), \tag{3}$$

where $\alpha = 2.0\times10^{-4}\ K^{-1}$ is the thermal expansion coefficient, and $T_0 = 10$.

The wind forcing, f_{wind}, is applied as a stress integrated over the surface, given as the analytical function:

$$\tau_{\text{wind}} = -\tau/\rho_0\cos(2\pi y/L_y). \tag{4}$$

The domain extends from $x = 0$ to $x = L_x = 1000\ km$ and $y = 0$ to $y = L_y = 1000\ km$. This simulates a mid-latitude double gyre, with easterly winds in the north and south, and westerly wind in the middle of the domain. The amplitude of the wind stress is $\tau_0 = 0.1\ Nm^{-2}$.

2.2 Numerical configuration of the test case

The Boussinesq equations (1a) and (1b) are discretised via a finite element integration using a $P1_{\text{DG}} - P2$ velocity, pressure element pair. This element pair has a number of advantageous properties for ocean simulations, *a.o.* its beneficial balance properties [6]. The incompressibility constraint (1b) is enforced through a pressure correction approach. The non-linearity in the advective terms, and the coupling of the heat equation and the buoyancy term are dealt with in a Picard iteration. The free surface is solved in conjunction with the pressure equation [7]. Finally the heat advection diffusion equation is solved with a standard $P1$ SUPG discretisation [11].

The mesh used in the baroclinic gyre benchmark test case has 10 million vertices; resulting in 200 million degrees of freedom for velocity due to the use of DG. The basic configuration is set-up to run for 4 time steps and not to adapt. It hence considers primarily the matrix assembly and linear solver stages of a model run. For these problems the simulation reads in the input files at start and default behaviour is to dump output files at the beginning and at the end. In addition, a statistics file is output every time step, this includes global integrals, maxima and minima of solution fields and other diagnostic output. All other input/output is switched off, although this can be switched on to examine I/O performance.

2.3 Solver comparisons

ICOM uses the PETSc library for the solution of the sparse linear systems arising from the numerical discretisation. A wrapper **petsc_solve**, it takes in the CSR matrix, and constructs a PETSc matrix data type, sets the solver and preconditioner options, and then passes the system to PETSc to solve. This wrapper offers full access to several classic algorithms contained in PETSc and also get the extra benefits with new added algorithms without extra work.

For the gyre test case, the pressure matrix has a very high condition number, making it very difficult to solve using conventional preconditioners and solvers. Here we use an AMG preconditioner which is the best choice for large systems (because of their better scaling properties), and ill-conditioned systems, such as those in large aspect ratio problems, or more generally problems in which there is a large variety in length scales. An algebraic multigrid method targeted specifically at large-scale, large aspect ratio ocean problems is developed for the ICOM model [8]. This multigrid method is applied as a preconditioner within a Conjugate Gradient iterative method.

PETSc provides a $3rd$ interface to other AMG methods such as HYPRE/BoomerAMG [13]. The parallel efficiency for pressure solver using ICOM MG and BoomerAMG preconditioners are in Figure 2. We can see ICOM MG has better scalability than BoomerAMG due to its specialised nature.

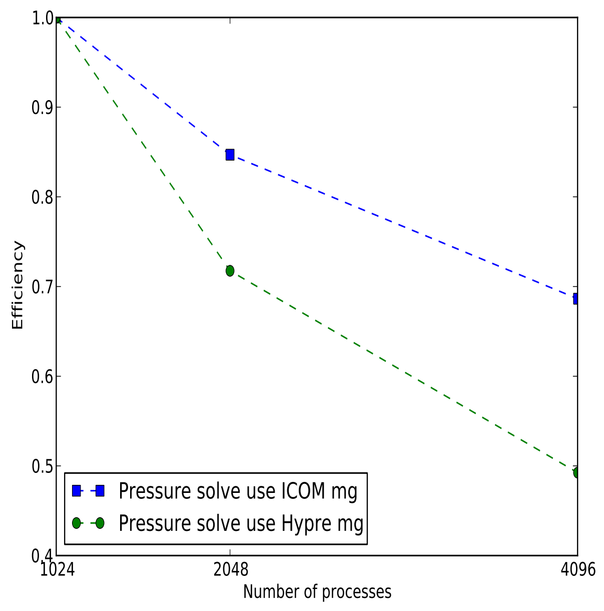

Figure 2: Comparison of pressure solver efficiency using algebraic multigrid method from ICOM MG and HYPRE BoomerAMG

3. Profiling and Performance Analysis

Users should not spend time optimising a code until after having determined where it spends the bulk of its time on realistically sized problems. Profiling is the best way to address both the serial execution of the code(such as cache usage, vectorisation) and parallel aspects, such as parallel efficiency, load balancing and communications overheads. Profiling using CrayPAT and Vampir on HECToR has been performed on the gyre benchmark test case which is of particular relevance to GFD applications.

Ideally, it would be expected that the process for using profiling tools like CrayPAT and Vampir would consist of firstly running a representative benchmark test case with tracing enabled to produce trace files, then viewing the tracing data using specialised tools. However, in the case of large benchmark test cases this may not be possible. Even for a relatively small ICOM dataset of 0.21 million nodes, with CrayPAT suggested automatic profiling options, trace files can be hundreds of GBytes which makes analysis impossible. The size of the trace file data depends on the nature and intensity of the profiling experiment and the duration of the program run. Using runtime variables such as PAT_RT_SUMMARY in CrayPAT could make a big reduction of data size but at the cost of fine-grain details. Specifically, when running tracing experiments, the formal parameter values, function return values, and call stack information are not saved. Determining a way to control profiling data size whilst at the same time gathering in-depth and informative is key for understanding the performance bottlenecks in large realistically sized problems. Under such circumstances, a starting point is to use simple timing hooks in the code to get a coarse grain profile of code performance, then to use these results as a basis for more fine grain profiling with the CrayPAT/Vampir API in the identified areas of interest.

3.1 Basic timings

Following above idea, we focus on the solution of the momentum equation (1a) in combination with the incompressibility constraint given by the continuity equation (1b), as this is by far the main cost of the simulation, and dominates the overall scaling of the simulation. The solution process consists of the assembly of the linear systems representing the discretised momentum equation and the pressure equation, and the solution of those. Thus the scaling analysis of the momentum equation is naturally broken down into 4 parts: assembly of the pressure matrix, linear solve for the pressure equation, assembly of the discretised momentum (velocity) equation, and linear solve of the momentum (velocity) equation.

From Figure 3, we can see that matrix assembly for pressure and velocity can take more than 30% of the total simulation time with 1024 cores, where pressure solver occupy nearly 53.9% of the total simulation time. The matrix assembly phase is expensive for a number of reasons including: significant loop nesting, where the innermost loop increases in size with increasing quadrature; indirect addressing (due to unstructured meshes) and cache re-use.

Comparing with 1024 cores, Figure 4 and Figure 5 show the speedup and efficiency of momentum solver and each of its components. As we can see from the graphs, the scaling is very respectable. Velocity is being solved using a discontinuous Galerkin method (DG). This is showing very good scaling characteristics.

The pressure assembly is showing the less parallel efficiency than the velocity assembly. It is to be noted that this assembly only occurs once in an entire model run, so is expected to take only a small fraction of runtime in a normal run with a lot more time steps. Due to non-linearities the momentum equation does have to be re-assembled every non-linear iteration within a time step.

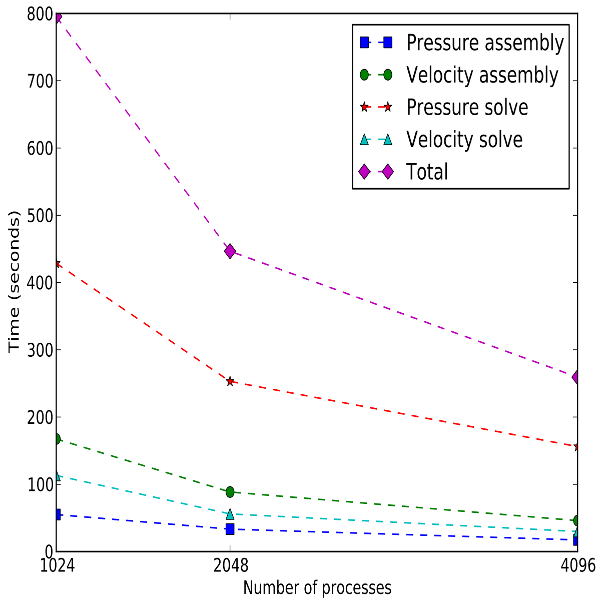

Figure 3: Wall time for the assembly and solve of the momentum and pressure equation

3.2 Communication overhead and load balance analysis

Using CrayPAT, we obtained the statistic of three groups of functions, namely MPI functions, USER functions and MPI_SYNC functions. MPI_SYNC is used in the trace wrapper for each collective subroutine to measure the time spent waiting at the barrier call before entering the subroutine. Therefore, MPI_SYNC statistics can be a good indication of load imbalance. The time percentage of each group is shown in the Figure 6.

With core counts from 1024 to 4096, we can see that the time percentage spent in MPI increases from 28.7% to 33.1% while USER functions drop from 45.5% to 24.9%, and time percentage of MPI_SYNC increase from 25.7% to 42.0%. This lead us to identify the top time consuming functions in each group along with their calling hierarchy.

3.3 Top time consuming functions in each group

Figures 7-9 give the top time consuming functions in each group. In Figure 7, the speed up of the linear solver KSP-

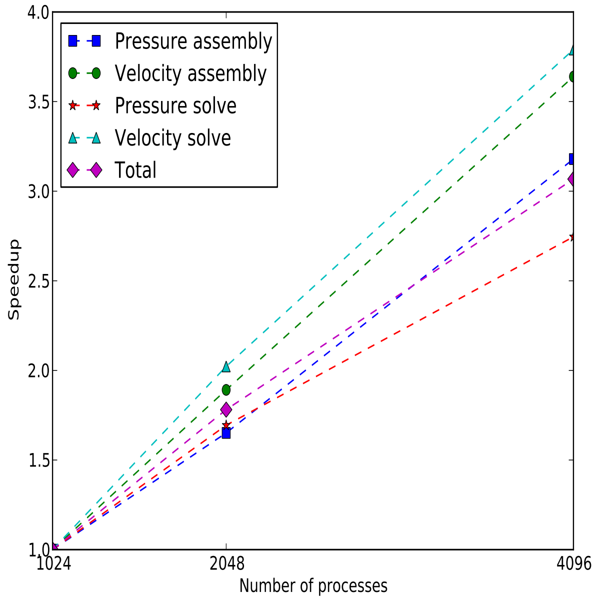

Figure 4: Speedup of the assembly and solve of the momentum and pressure equation

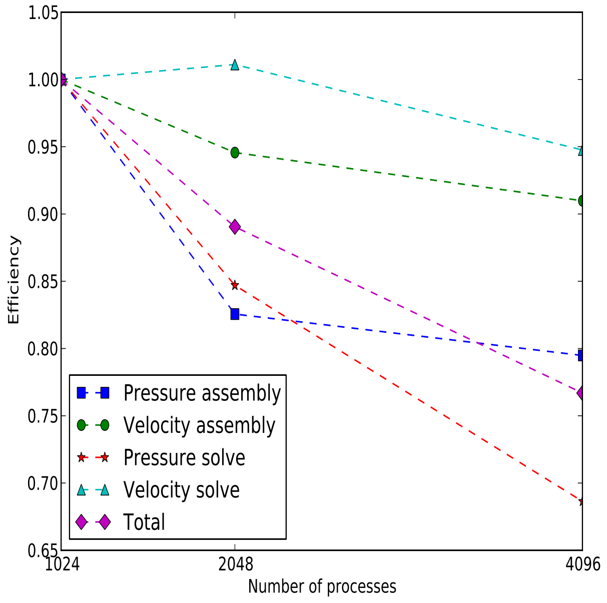

Figure 5: Efficiency for the assembly and solve of the momentum and pressure equation

Solve is about 3.5 with 4096 cores comparing with 1024 cores according to the CrayPAT tracing results. The function main represents the functions that have not been traced in the code. These functions are outside of momentum solver. Future work will focus on these functions of poor scaling behaviour.

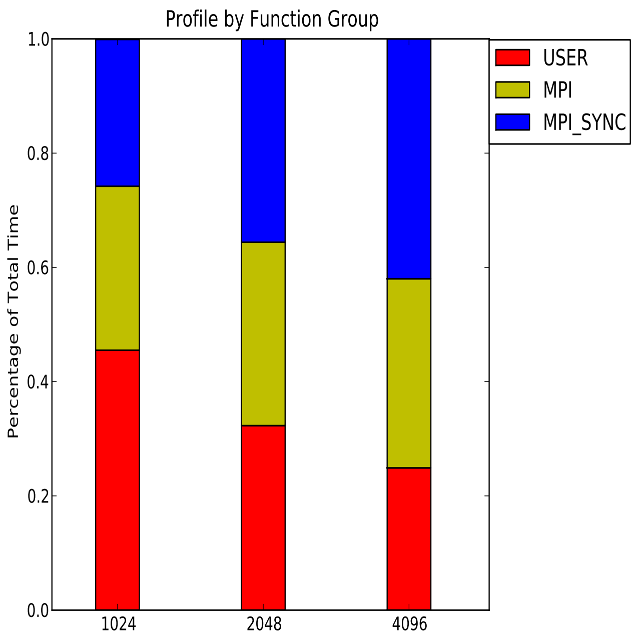

Figure 6: Profile by function group

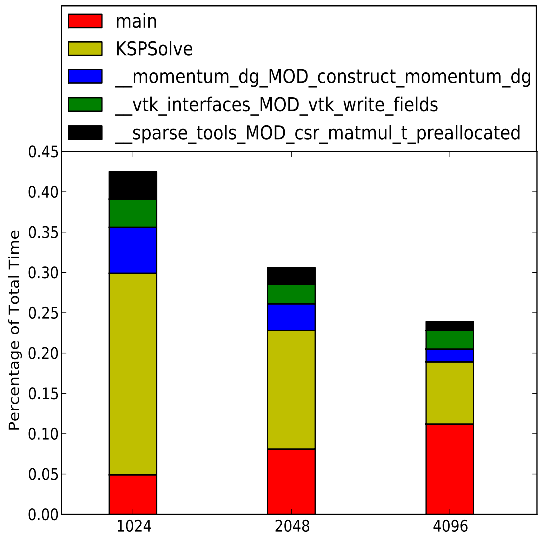

Figure 7: Top time consuming user functions got from CrayPAT

The most time consuming of the MPI groups is MPI_Allreduce. It is expected that this collective operation does not scale well. However on the XT4 the scaling is relatively good from 2048 to 4096 cores in Figure 8. From the call tree generated by CrayPAT, it becomes clear that this function is called from PetscMaxSum within PETSc. MPI_Waitany is indicative of the quality of the load balancing. Given that this amount does not increase significantly between runs on 1024 to 4096 cores in Figure 9, it does not appear that load-balancing is worsening noticeably as the core count increases.

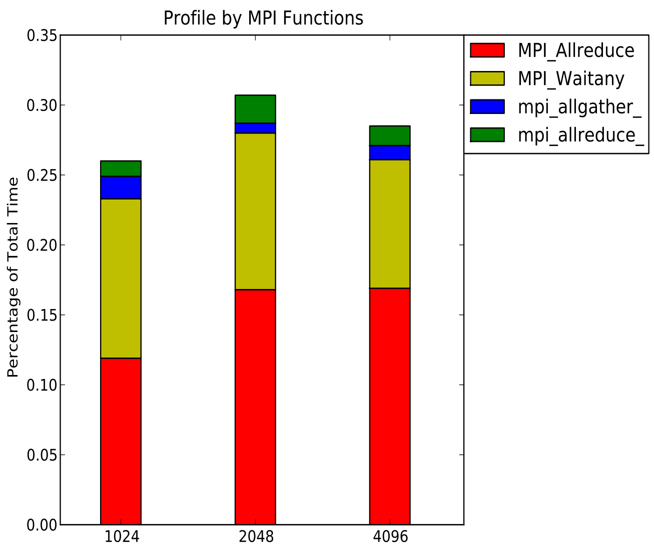

Figure 8: Top time consuming MPI functions

In Figure 9, MPI_Allreduce accounts the most part of waiting time spent in the barrier, it is worth to check if there are possibility to combine several MPI_Allreduces together. MPI_Bcast and MPI_SCAN are becoming more significant on 4096 cores, compared to runs on 1024 and 2048 cores.

3.4 Some guidelines for ICOM supporting third party libraries tracing

As ICOM uses makes extensive use of third party software, it is also important to obtain insight in performance issues in these other software packages.

Profilers like CrayPAT and Vampir normally require direct access to the source file or the object file, which are typically not available for third party package software installed on a given system. This will limit the view on the overall performance of applications widely using such software packages. For instance, ICOM use PETSc as the sparse linear system solver; With nearly 70% time spent in the PETSc, it is necessary to know about the performance profile within PETSc that will determine the whole performance of ICOM. But without direct access to the source

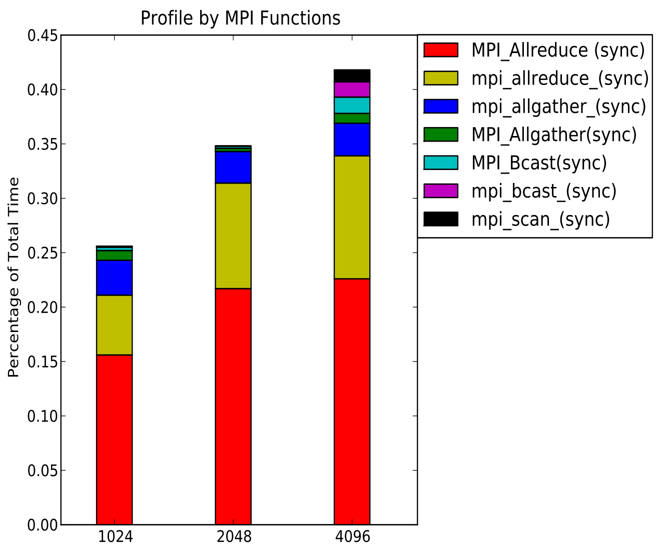

Figure 9: Top time consuming MPI_SYNC Functions

and object file, it is very hard to trace the specific functions in PETSc. One possible solution is to rebuild and install PETSc in the user home directory. In this case CrayPAT can profile PETSc as normal USER functions. A problem will then be how to properly reduce the profiling data from PETSc as there are very many different PETSc functions, called from various places. This will generate a large amount of profiling data. Using the API functions of profiling tools may be a solution but it will require recompiling the PETSc each time a different grouping is chosen. Another choice is to generate a specific function list which is a subset of PETSc functions. Directly instrumenting the subset of PETSc functions can also help to reduce the size of large data and get some useful statistics.

Vampir has some valuable MPI tracing features. These have been explored by recompiling ICOM with the Vampir wrapper functions. A different benchmark, modelling a so called open ocean deep convection process, has been used to check the performance of the adaptivity part of ICOM. Figure 10 shows the MPI message length statistics generated by the Vampir wrapper, used to instrument the adaptivity part of ICOM. The zero length messages were found to come from calls within Parmetis.

By directly tracing a subset of PETSc functions, the MPI statistics and call tree also show that MPI_Allreduce is called from PETSCSolve_CG within PETSc (see Figure 8). With the new release of PETSc, this function is optimised by combining the MPI_Allreduce calls together. The performance test will be carried out in the near future.

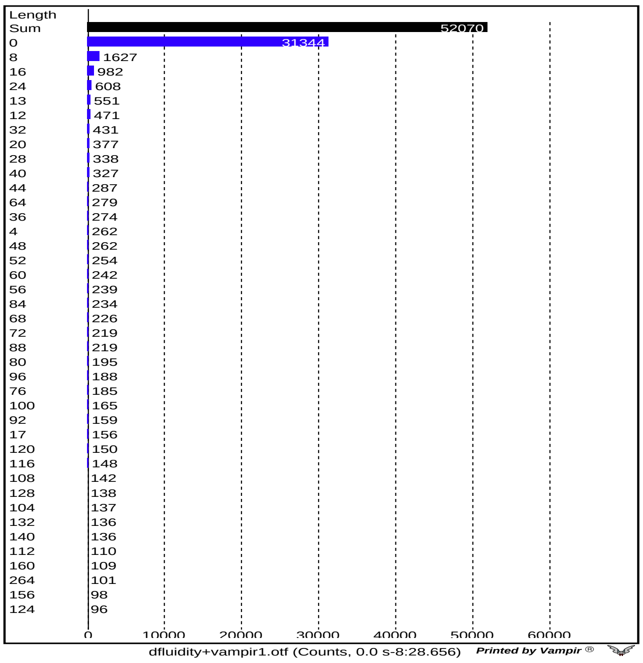

Figure 10: MPI Message Length Statistics for OODC with Adaptivity

4. Summary and Conclusions

The ICOM dCSE project has significantly improved the performance of the code to enable efficient usage of large high performance computing systems such as the Hector Cray XT4. Presently the code is now scaling well up to 4096 cores on HECToR. Runs on even larger core counts could be achieved if suitably partitioned datasets existed, these are currently under construction. The current barrier to running larger problem sizes is the memory footprint and computational cost of preprocessing tools; specifically the initial domain decomposition which is essentially a serial bottleneck.

Porting the code to HECToR has involved several challenges. Firstly, the code requires a range of third party libraries which need to be maintained on the target platform. Secondly, the code uses some of the latest features included in Fortran 95 and this has on occasion tested the HECToR compilation environment, leading eventually to several bug fixes in the PGI Fortran compiler. This requires a lot of effort from different group including the developers, STFC ARC group and HECToR Support.

Profiling the real world applications is a big challenge, it required a considerable understanding of profiling tools and extensive knowledge of the software itself. Determining a suitable way to reduce the profiling data size without losing the fine grain details was critical for successfully profiling. Inevitably this procedure involved much experimentation requiring large numbers of profiling runs.

The introduction of manual instrumentation was required in order to focus on specific sections of the code. To successfully and informatively profile a real world application such as ICOM, an in-depth knowledge of profiling tool usage became essential. Generally, profiling the real world applications will face how to get proper data from profiling tools. Finding a proper way to reduce the profiling data size without losing the fine grain details is critical for a successfully profiling.

As far as profiling ICOM has found, CrayPAT and Vampir are well suited to fine grain profiling on specific sections of the code, once the areas of interest have been identified. Sometimes this has required the introduction of manual timing functions into the code by the user. These tools are also very useful for generating MPI statistics for the whole simulation. In general we can conclude that the profiling analysis has been very useful for developing and optimising the code.

Acknowledgements

The authors would like to acknowledge the support of a HECToR *distributed Computational Science and Engineering* award. The authors would also like to thank the HECToR support team for their help throughout this work. Gorman gratefully acknowledges support from the Leverhulme Trust.

Some experiments of this paper has been carried on the Swiss National Supercomputing Centre's Cray XT5, Rosa, and we would also like to thank their support team.

References

[1] Ford R, Pain CC, Piggott MD, et al. A nonhydrostatic finite-element model for three-dimensional stratified oceanic flows. Part I: model formulation. *Monthly Weather Review*, 132: 2816–2831, 2004

[2] Pain CC, Piggott MD, et al. Three-dimensional unstructured mesh ocean modelling. *Ocean Modelling*, 10: 5–33, 2005

[3] Gorman GJ, Piggott MD, Pain CC, et al. Optimisation based bathymetry approximation through constrained unstructured mesh adaptivity. *Ocean Modelling*, 12: 436–452, 2006

[4] Gorman GJ, Piggott MD, Wells MR, et al, A systematic approach to unstructured mesh generation for ocean modelling using GMT and Terreno, *Computers & Geosciences*, 34:1721–1731, 2008

[5] Gorman GJ, Pain CC, Piggott MD, Umpleby AP, Farrell PE, Maddison JR, Interleaved parallel tetrahedral mesh optimisation and dynamic load-balancing, *Adaptive Modeling and*

Simulation 2009, 101–104, 2009, Bouillard Ph, Diez P (eds.), CIMNE

[6] Cotter CJ, Ham DA, Pain CC A mixed discontinuous/continuous finite element pair for shallow-water ocean modelling *Ocean Modelling*, 26: 86–90, 2009

[7] Kramer SC, Pain CC Modelling the free surface using fully unstructured meshes *In preparation*

[8] Kramer SC, Cotter CJ, Pain CC Solving the Poisson equation on small aspect ratio domains using unstructured meshes *Ocean Modelling*(submitted) and arXiv:0912.2194

[9] http://www.mcs.anl.gov/petsc/petsc-as/.

[10] http://www.cs.sandia.gov/Zoltan/.

[11] Gresho PM, Sani RL. *Incompressible Flow and the Finite Element Method,* Wiley: New York, 1998.

[12] M. D. Piggott, G. J. Gorman, C. C. Pain, P. A. Allison, A. S. Candy, B. T. Martin, M. R. Wells. A new computational framework for multi-scale ocean modelling based on adapting unstructured meshes. *Int. J. Numer. Methods Fluids,* 56:1003–1015, 2008.

[13] https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html

[14] http://www.columbia.edu/ ma2325/prometheus/