CABARET on Jet Flap Noise and Quasigeostrophic Ocean Circulation Models

Phil Ridley
Numerical Algorithms Group Ltd,
Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, UK,
email: support@nag.co.uk

October 1, 2012

Abstract

This dCSE project will concern two very different CFD applications, which both share a common method for resolving advection, forcing and dissipation - the Compact Accurately Boundary Adjusting high-REsolution Technique (CABARET). The CABARET method is a low dissipative and low dispersive scheme that constitutes a substantial upgrade of the second-order upwind leapfrog method. The algorithm is very suitable for distributed HPC since it has a very local computational stencil that for scalar advection constitutes only one cell in space and time.

For geophysical fluid dynamics, CABARET is used to model the geostrophic, turbulent midlatitude ocean circulation, in terms of the quasigeostrophic potential-vorticity dynamics. This code is called PEQUOD (Parallel Quasi-Geostrophic Model) and is primarily intended for the study of mesoscale eddy processes on a multi-layer structured rectangular domain. In aeronautics, CABARET is applied to a turbulence model for jet-flap-noise interaction (Cfoam-CABARET). Modelling the unsteady component of the acoustic model for the free-stream flow/jet interaction with the wing-flap, as well as for the study of hot/complex geometry jet physics, is very challenging. The computational model uses a Monotonically Integrated LES (MILES) approach that is explicit Sub-Grid-Scale (SGS) turbulence-model free.

Cfoam-CABARET uses an unstructured domain decomposition which is suitable for complex geometries, but this gives rise to issues related to I-O performance on large HPC systems, whereas PEQUOD does not experience this problem, due to the use of a structured multi-dimensional domain decomposition. However, unlike Cfoam-CABARET, PEQUOD requires an additional global inversion scheme which is used to calculate the potential vorticity (stream function). This project will implement MPI-IO in Cfoam-CABARET to improve the I-O performance, and optimise the parallel Swartztrauber elliptic solver for the potential vorticity in PEQUOD. This work will help both applications: in aeronautics, to investigate the effect of fine-scale-flow structures on far-field noise in the audible range of frequencies; in ocean modelling, a whole set of very interesting and important results are waiting to be revisited and upgraded, with a completely new level of dynamical realism that can only be achieved with a scalable implementation of the existing code.

This project was approved for 6 months effort in December 2010 and was completed August 2012. The work was supervised by Dr Sergey Karabasov of the School of Materials Science and Engineering at Queen Mary University of London and Dr Pavel Berloff of the Department of Mathematics at Imperial College London.

Contents

1 Introduction

1.1 Application Overview

CABARET is a general-purpose advection scheme which is suited for computational aeronautics and geophysics problems [1]. For solving Navier-Stokes equations with Reynolds numbers of 10000, the method gives a very good convergence without any additional preconditioning down to Mach numbers as low as 0.05. In particular for the modelling of a hydrodynamic instability and free jet, CABARET is able to produce results comparable to a conventional second order method with at least 30 times more efficiency [2]. The CABARET method is a low dissipative and low dispersive scheme that constitutes a substantial upgrade of the second-order upwind leapfrog. The algorithm is very suitable for distributed HPC since it has a very local computational stencil that for scalar advection constitutes only one cell in space and time.

For this project, two very important CFD applications which both use CABARET will be optimised for improved performance on HPC architectures: one of which is for modelling aircraft jet engine-flap interaction community noise, and the other for modelling transient mesoscale eddies of the ocean circulation.

The ocean circulation problem is important, because the ocean contributes to the climate variability, and this contribution is largely controlled and even driven by the intrinsic nonlinear dynamics associated with the mesoscale eddies. Some of the corresponding eddy effects are non-diffusive and even anti-diffusive, implying that they are very difficult to parameterize in a simple way. The inability to resolve the eddies dynamically is a major hurdle for predictive understanding of the global climate variability.

Resolving the eddies is a high-priority task, which requires cutting-edge numerical models coupled with supercomputing resource. The Parallel Quasi-Geostrophic Model (PEQUOD) code is a leading application in this area and has been developed to solve the layered quasi-geostrophic potential vorticity equation [3]–[4] subject to forcing and dissipation. The use of an optimised version of PEQUOD on HECToR will enable unprecedented levels of the dynamical realism and structural details of the eddies and their effects on the large-scale ocean circulation. This, in turn, will be an important contribution to climate research, by enabling calculations for the most turbulent and, thus, the most physically relevant regimes of the ocean circulation, in the classical quasigeostrophic set up:

1. Intrinsic large-scale low-frequency variability of the turbulent wind-driven ocean gyres;
2. Physics of the eddy/large-scale ow interactions maintaining large-scale circulation and its variability;
3. Material transport induced by the mesoscale oceanic eddies;
4. Parameterisation of the essential eddy effects for use in oceanic components of comprehensive global-climate models.

Also as important is the aircraft noise problem, because by 2020 the total number of flights is expected to double and, accordingly, each individual aircraft needs to be made at least twice as quiet. In the past, jet noise reduction for civil aircraft could be achieved by increasing the size of the jet engines. This allows a reduction of the jet speed for the same amount of thrust and, since jet noise scales as a high power of the jet exit velocity, this also reduces the noise. However, any further decrease is only possible if detailed noise mechanisms are quantified and jet noise reduction remains a formidable problem.

In addition to the jet noise problem, the airframe noise including the airframe/engine interaction effects is a major contributor to the aircraft noise at approach and its reduction also

becomes a problem [5]–[6]. In particular, when deployed at a large angle of attack at approach conditions, the flaps become a dominant noise source for the airframe noise. Moreover, for engine-under-a-wing configurations, Jet-Flap Interaction (JFI) noise can also become an important noise component at take-off conditions.

For flight conditions such effects are often coupled, which makes high-resolution computational modelling ever more valuable. The complexity of the unsteady flow sets the minimum size of the computational problem that aims to model it has to be extremely large. But by using the high-resolution Cfoam-CABARET code and extending it to large-scale models on HECToR, the important unsteady effects of flap-jet interaction that contribute to noise can then be captured. Cfoam-CABARET uses a Monotonically Integrated LES (MILES) approach to the solution of the Navier-stokes equations. For enforcing the non-oscillatory property of the solution, the CABARET scheme uses a low-dissipative non-linear flux correction that is directly based on the maximum principle for the flux variables. Each time iteration of the method then consists of a conservation phase and a characteristic decomposition phase.

1.2 CABARET and its application to geophysical fluid dynamics

PEQUOD solves the multi-layer quasi-geostrophic equations in a rectangular domain. The model has two primary modes of operation: basin mode and channel mode. Typically, the basin mode is used as a simple model for a wind driven double gyre, with energy input to the system via upper layer wind forcing. The channel mode is used as a simple model for baroclinic zonal jet formation, with the energy input to the system via lateral buoyancy forcing.

The quasi-geostrophic potential vorticity equation (1) is solved subject to forcing and dissipation

$$\partial_t + J(\psi, q) = F, \tag{1}$$

where q is the quasi-geostrophic potential vorticity (hereafter referred to as the "potential vorticity").The LHS in equation (1) is the material derivative, and F is any forcing and dissipation. Equation (1) is then represented in a vertically layered form, which leads to a series of decoupled Helmholtz equations.

In PEQUOD two separate finite-difference approaches may be used to solve the layered quasi-geostrophic potential vorticity equation, however, this work will only be concerned with the CABARET approach. For the advection of relative potential vorticity, the scheme consists of three steps: predictor, extrapolator and corrector. The predictor and extrapolator step are performed using centred differencing in space. The extrapolator step is performed using upwinding of the advection term with a non-Total Variation Diminishing (TVD) flux limiter to bound the approximation. This scheme is second order accurate provided the method used to compute the fluxes is (at least) second order accurate.

Potential vorticity inversion for q to yield the stream function ψ is performed via the resulting linear elliptic problem being inverted using a direct solver. A discrete sine transform [7] is performed in the meridional direction, yielding a series of sparse one-dimensional Helmholtz problems in the zonal direction that can, for example, be inverted via Gaussian elimination. An inverse discrete sine transform of the resulting solutions of the one-dimensional problems completes the inversion.

1.3 Background to the aeronautics application of CABARET

In the framework of the current EPSRC Flap Noise project, the Cfoam-CABARET code is used to perform high-resolution acoustics-sensitive MILES calculations which are aimed at understanding and prediction of jet-wing-flap interaction.

In Cfoam-CABARET, for the Navier-Stokes equations, each time iteration of the CABARET method consists of a conservation phase and a characteristic decomposition phase. This algorithm works by firstly performing a conservative predictor step, followed by calculation of the viscous terms, then an extrapolation step where the local cell-based characteristic splitting is performed. Finally, the conservative corrector step is performed.

1.4 Outline of this project

This software development project will serve two separate purposes: It will enable more efficient use of I-O within the unstructured Cfoam-CABARET code and improve the scalability of the structured PEQUOD code. Here, the terms unstructured/structured refer to the spatial decomposition used within the model.

The main functional difference between Cfoam-CABARET and PEQUOD is that PEQUOD models incompressible flow and therefore requires an additional parallel elliptic pde (Helmholtz) solver for the pressure. The data decomposition in PEQUOD is intrinsically far easier to work with than Cfoam-CABARET. It is also worthwhile noting that relationships between neighbouring cells and partitions are generally more complex to manage in the latter code. The consequence is that data associated with unstructured grid layouts requires a lot of effort so that it can be managed and updated by the methods which determine new values based on other values in their physical proximity.

The original objectives for this project were as follows:

WP1: Enhanced Parallel I-O within Cfoam-CABARET

Update the I-O in Cfoam-CABARET. The original method used single process I-O whereby an unstructured grid was read in by the master process and then broadcast to all other processes. A similar process was also used for output where checkpoint/Tecplot360 binary files would be gathered and then written by the master process, at intervals specified by the user. An MPI-IO model will be implemented such that a 5×10^7 cell grid can be read in less than 5 minutes for a typical simulation with at least 1000 cores.

WP2: Parallelised quasi-geostrophic CABARET code

Develop a scalable multi-layer quasi-geostrophic solver and benchmark on a double gyre quasi-geostrophic model. This model will include 3 layers, forcing and dissipative processes on a 1025×1025 uniform grid, wind forcing parametrisation and the parameters of lateral viscosity and bottom friction. This will be achieved in the following steps:

WP2.1: Implementation of a structured spatial decomposition for the computational grid

The global domain will be partitioned such that each process will each be assigned a sub-section.

WP2.2: Implement parallel structured CABARET (for incompressible flow)

Develop a quasi-geostrophic parallel solver by implementing asynchronous MPI calls at the

conservative predictor step for the interpolation of cell-centred conservative variables to the cell vertices and at the extrapolation step where the local cell-based characteristic splitting for the updated conservative and flux variables is performed. This code will be tested against a simple "classical" parallel implementation of a central leapfrog and the central finite difference scheme for which the results are well known.

WP3: Parallelisation of the Helmholtz solver

Develop a scalable layered quasi-geostrophic solver including performance of the standard elliptic solver that inverts potential vorticity in a closed domain or zonally periodic channel in order to get velocity streamfunction. This method uses the Fourier Analysis and cyclic reduction direction by direction. The code will be benchmarked on the double gyre quasi-geostrophic model.

The overall aim of WP2 and WP3 will be to enable models with a horizontal-grid interval of 1km, which with the CABARET advection scheme implies that the turbulence scales of 6km and longer will be represented quite accurately. Historically, wind-driven ocean circulation models with 1km resolution have never been used for long-time integrations even for poor vertical resolution. The study of important nonlinear interactions of relatively high-order vertical modes and their large-scale consequences will also be enabled - using 10 layers. Finally, runs on 2km, 4km, and 8km grids will become possible. The aim of WP1 is to reduce the amount of time spent performing I-O by 5 times for a typical simulation.

2 Implementation on HECToR Phase 3

2.1 PEQUOD

PEQUOD (the Parallel Quasi-Geostrophic Model) is a finite difference code for solving the multi-layer quasi-geostrophic equations in a rectangular domain, it is primarily intended for the study of mesoscale eddy processes. PEQUOD is parallelised for distributed memory machines using MPI and is written in modern Fortran 90/95, taking advantage of several more advanced features of the language (e.g. derived data types, pointers, dynamic memory allocation, recursive subroutines and function overloading). It is around 10,000 lines of code. The model employs a structured Cartesian grid which may be de-composed in either one or two dimensions - for each layer. This defines a fully structured partitioning of the problem, with each process being assigned a partition of the global domain.

The CABARET solution of the quasi-geostrophic potential vorticity equation is a purely explicit problem (no matrix inversion operations are required) however, differencing at partition boundaries does require information to be communicated from adjacent processes. This is achieved via the use of halo (or ghost) nodes along with non-blocking **MPI_ISEND** and **MPI_IRECV** to communicate data to each neighbouring partition. A cartesian MPI topology is also used to locate neighbouring partitions. As this section of the code drives a purely finite difference, nearest neighbour algorithm, scalability is linear and so will not be discussed.

Although the solution of the quasi-geostrophic potential vorticity equation is a purely explicit problem, the potential vorticity inversion step is not. This is an implicit problem where the Helmholtz problem must be inverted in order to obtain the stream function from the potential vorticity. Furthermore, the elliptic nature of the Helmholtz operator means that this is inherently a non-local procedure (information must, in a direct solver, be communicated).

This step uses a one-dimensional parallel Helmholtz solver, along with a combination of data transposing. The system is repartitioned (via communication) yielding a new partitioning

that is contiguous in the meridional direction (first data transpose). This allows a discrete sine transform in the meridional direction to be performed locally on each process. The one-dimensional Helmholtz problems are then inverted in parallel using the Schumann and Strietzel algorithm [9]. The inverse discrete sine transform step is then performed locally. Finally, the system is repartitioned back to the original partitioning (second data transpose). Due to the requirement to communicate globally, derived data types of varying size at the start of this project, the data transposes were implemented with **MPI_ALLTOALLW**.

2.2 Performance of PEQUOD

Performance on HECToR Phase 3 is given in Figure 1 for the benchmark test case mentioned in WP2, which is for a double gyre quasi-geostrophic model. This has 3 layers on a 1025×1025 uniform grid. All runs were performed with fully populated HECToR Phase 3 nodes and performance with the default compilers was compared. The particular compiler optimisations were as follows: GCC -O3 -ffast-math; PGI -fast -O3, CCE -O3.

It is clear that best performance is given by GCC. Most compute time is spent performing the CABARET extrapolation step, however, this part of the code scales linearly with very little amount of time spent in communication and is therefore is not a significant cost to the overall performance. Whereas, the second most computationally expensive routine is the inverse discrete sine transform step, which is used at each time step for calculating the stream function via the potential vorticity inversion step. But this does not scale well due to the global transpose steps and thus causes a severe problem in terms of code performance. The reason GCC gives better performance than both PGI and CCE is because the inverse discrete sine transform step has been optimised for GCC.

However, even with GCC, PEQUOD had very limited scalability on HECToR for useful problem sizes. This was a significant concern and the optimisation of this calculation was necessary for further use of the code. The development of a parallelised quasi-geostrophic CABARET code for WP2 could be considered a straightforward task, due to the inherent nature of CABARET for structured grids. However WP3 for the efficient parallelisation of the Helmholtz solver would not be considered so straightforward.

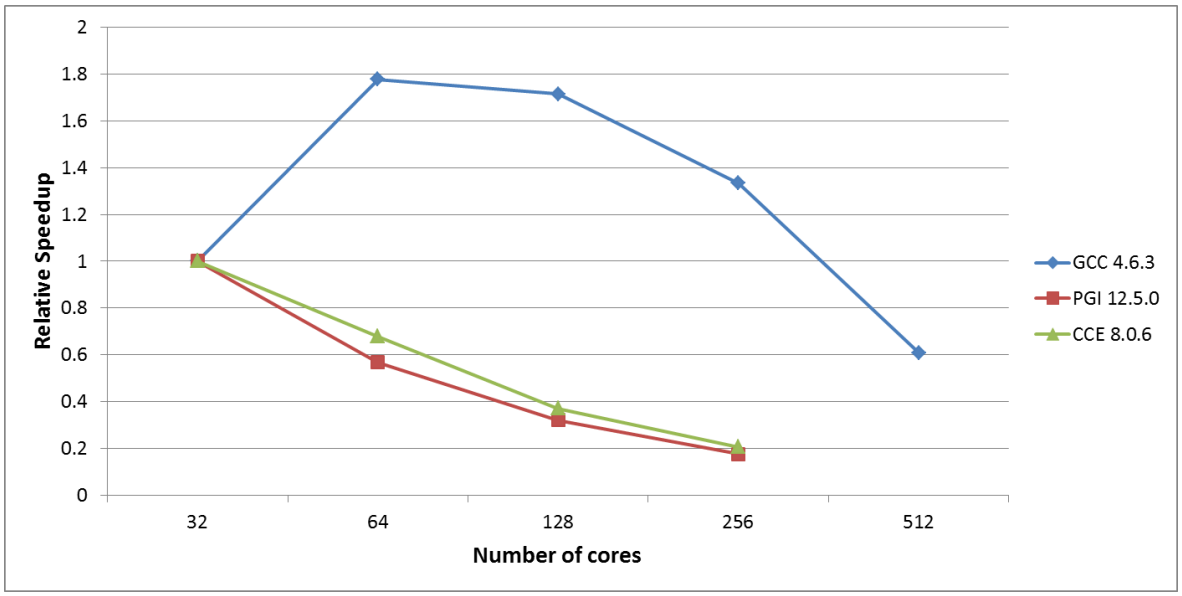

Figure 1: Initial scalability of PEQUOD on HECToR Phase 3.

2.3 Cfoam-CABARET

In the first dCSE project related to the CABARET code, a hybrid OpenMP / MPI parallel version of the unstructured code was developed which was scalable to more than 1000 cores on HECToR for problems of more than 5×10^7 cells. More information about that work is given in [8]. However, this code requires the use of a Gambit (Fluent) generated unstructured grid and initial pre-processing for the decomposition. It also uses a master I-O model which gathers data from each process for check pointing and visualisation and also broadcasts the initial Gambit (Fluent) generated finite element mesh. The code can handle arbitrary data decompositions of the Gambit unstructured mesh, although it is recommended to use structured ones wherever geometrical arrangements allow.

2.4 Performance of Cfoam-CABARET

The production test case used to demonstrate the scalability of the unstructured CABARET code was a 3D backward facing step geometry with boundary conditions for laminar flow, Reynolds number=5000 and Mach number=0.1. The grid was 5.12×10^7 hexahedral cells and scalability was considered for 276 time steps without any I-O involved. On Phase 2b of HECToR (Cray XT6, 24 core magny-cours processors), the code demonstrated more than 80% parallel efficiency for up to 1000 cores (PGI 10.8.0). This was using 4 MPI tasks per node with 6 OpenMP threads and 2 MPI tasks with 12 OpenMP threads.

However, for a typical 5.12×10^7 cell Gambit generated grid, the time taken to read in a typical mesh was around 18mins and to write the Tecplot360 output file more than 25mins. The input data files are stored in ASCII although the checkpoint files are in binary and are therefore much faster to write, typically these take 10mins. But all timings increase linearly for more than 1000 cores. Hence a small part of this dCSE project was to improve the I-O in the existing code.

At the start of this project, the unstructured CABARET code evolved to Cfoam-CABARET. The main difference was that Gambit was no longer used to generate the initial input grid, instead OpenFOAM 1.7.1 had been adopted as both the unstructured grid generation tool and mechanism for parallel decomposition. The timings for reading in the input were improved by taking half as long as the single process method with the Gambit generated mesh, furthermore timings for output were also improved. But this was due to the fact that each process both reads in and writes out to separate data files (for both input and output). This is a consequence of the OpenFOAM I-O structure and a separate post-processing application called outFoamX has been developed to gather the separate output files and generate a single file for Tecplot360.

In more detail, for Cfoam-CABARET the input grid is stored such that the relevant part for each process is held in the directory processor* (where * indicates the number of the MPI process). Each file must be read in at the beginning of a simulation by each process. For restarts and visualisation, the files are written in processor* as results ResCells000000n and ResFaces000000n, where n is related to the time step. For restarts, ResCells000000n and ResFaces000000n will be read in from the relevant time step for each process. For post-processing outFoamX reads in ResCells000000n and ResFaces000000n, along with the OpenFOAM grid. The output from outFoamX is then a series of *.plt files which are then suitable for reading by Tecplot360.

It is worthwhile to note that although this method is faster than using a single process master I-O model, a very large number of post-processing files are required which in turn has potential to cause problems on the HECToR file system. E.g. for a typical simulation 1000's files are required for each MPI process. Therefore the implementation of an MPI-IO model for this data is still appropriate as this will reduce the number of files required by Cfoam-CABARET and for

future simulations it will enable restarts / post-processing with different numbers of processors, if so required, and subject to the OpenFOAM input grid. The work related to this development is described in more detail in Section 4 of this report.

3 Optimising the Potential vorticity inversion routine in PEQUOD

In depth profiling of PEQUOD with CrayPAT highlighted that the section of code concerned with the vorticity inversion routine of the Helmholtz solver was indeed the source of the inefficient communication, which in turn severely constrained any reasonable scalability for useful problem sizes.

3.1 Implementation

The parallel 1D tri-diagonal solver for the Helmholtz equation contained 5 calls to **MPI_ALLTOALLW** in the initial implementation, these are performed for each of the 3 layers for every time iteration. However, the number of calls is not relevant due to their placement within the code, but, the amount of time taken within each call is. The first feature noticed about the collectives was that each entry in the array of data types used in **MPI_ALLTOALLW** was identical. Therefore, since **MPI_ALLTOALLW** and **MPI_ALLTOALLV** only differ in that the former allows varying data type, calls to **MPI_ALLTOALLW** were replaced with calls to **MPI_ALLTOALLV** as shown below:

```
call mpi_alltoallw(psi(1, 2), ge_c%scounts1, ge_c%sindices, ge_c%types, &
  & ge_c%recv(1, 1, 1), ge_c%rcounts1, ge_c%rindices, ge_c%types, &
  & decomp_j%cart_comm, ierr);

call mpi_alltoallv(psi(1, 2), ge_c%scounts1, ge_c%sindices/8, &
  MPI_DOUBLE_PRECISION,  ge_c%recv(1, 1, 1), ge_c%rcounts1, &
  ge_c%rindices/8, MPI_DOUBLE_PRECISION, decomp_j%cart_comm, ierr);
```

It was expected that performance of **MPI_ALLTOALLV** would be better than **MPI_ALLTOALLW**, however it turned out that it was between 5-10% worse.

The next step was to develop a method using **MPI_ALLTOALL** and buffers, which used all the existing information available in the code. The benefit of doing this would be to take advantage of the hardware optimised collective (i.e. for the Gemini interconnect), however, the disadvantage was that **MPI_ALLTOALL** requires fixed length buffer sizes. Extra code would need to be developed to take account of this, to ensure that the buffers were packed and unpacked efficiently.Firstly, the size of the buffer is set to double the maximum size of the packed data for each process, this allows for those processes whose buffers are less than the maximum size, to contain zeros. The purpose of this being that all buffers on each process will be equal and of the same data type (**MPI_DOUBLE_PRECISION**), such that the following call may then be made:

```
call mpi_alltoall(c_full,2*max_ii2s,MPI_DOUBLE_PRECISION, &
  &  coeff_full,2*max_ii2s,MPI_DOUBLE_PRECISION,decomp_j%cart_comm, ierr)
```

Prior to this call the sending buffer (`c_full`), will have to be packed and after the call, the receiving buffer (`coeff_full`) unpacked. This is implemented within loops which contain copies from the array psi to `c_full` and then from `coeff_full` to `ge_c%recv`. The loops do incorporate some memory access which includes striding from `ge_c%scounts1` and `ge_c%rcounts1`, however this is unavoidable and fortunately does not cause any problems to performance.

3.2 Comparison of global collectives performance

Scalability of the new **MPI_ALLTOALL** implementation is shown in Figure 2, here it shown that the strong scalability for the benchmark test case in WP2 is now good up to 256 cores on HECToR Phase 3, whereas it was initially limited to use on only 64 cores. It is also worthwhile noting that although performance with PGI is not as good, there is improved scalability. Although CCE does not show any increase in scalability and the results are similar to those shown in Figure 1 with **MPI_ALLTOALLW**.

An exact comparison of the old and new implementations is shown in Figure 3, here it is easy to see that although the underlying trend is the same for both implementations, the cost of communication has been dramatically reduced with the new global collectives.

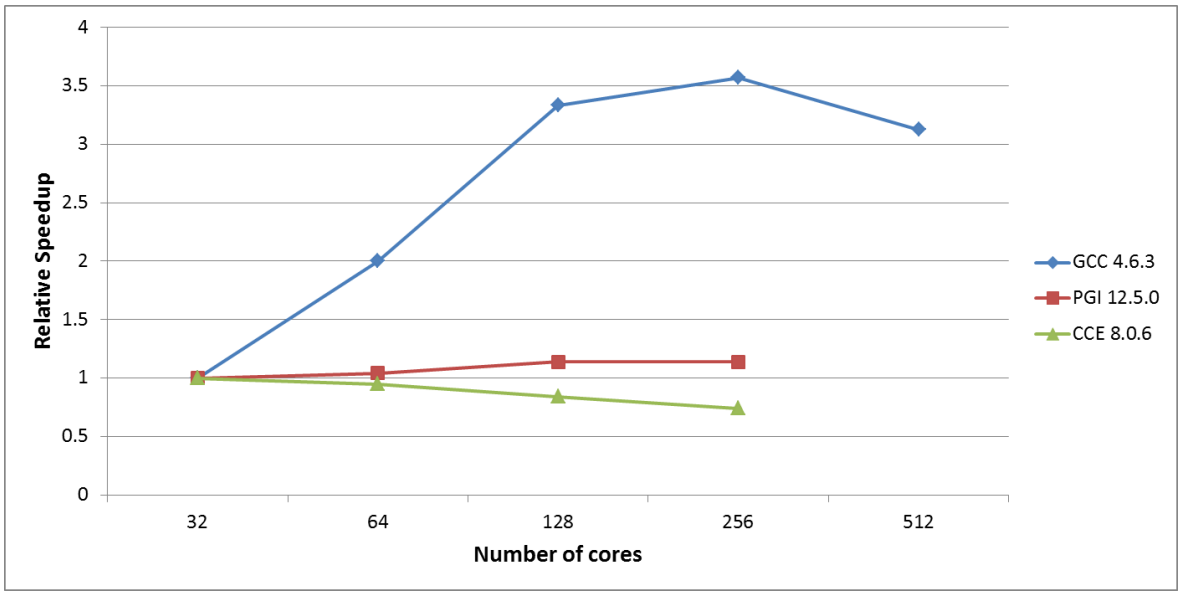

Figure 2: Scalability of PEQUOD with improved global collectives on HECToR Phase 3.

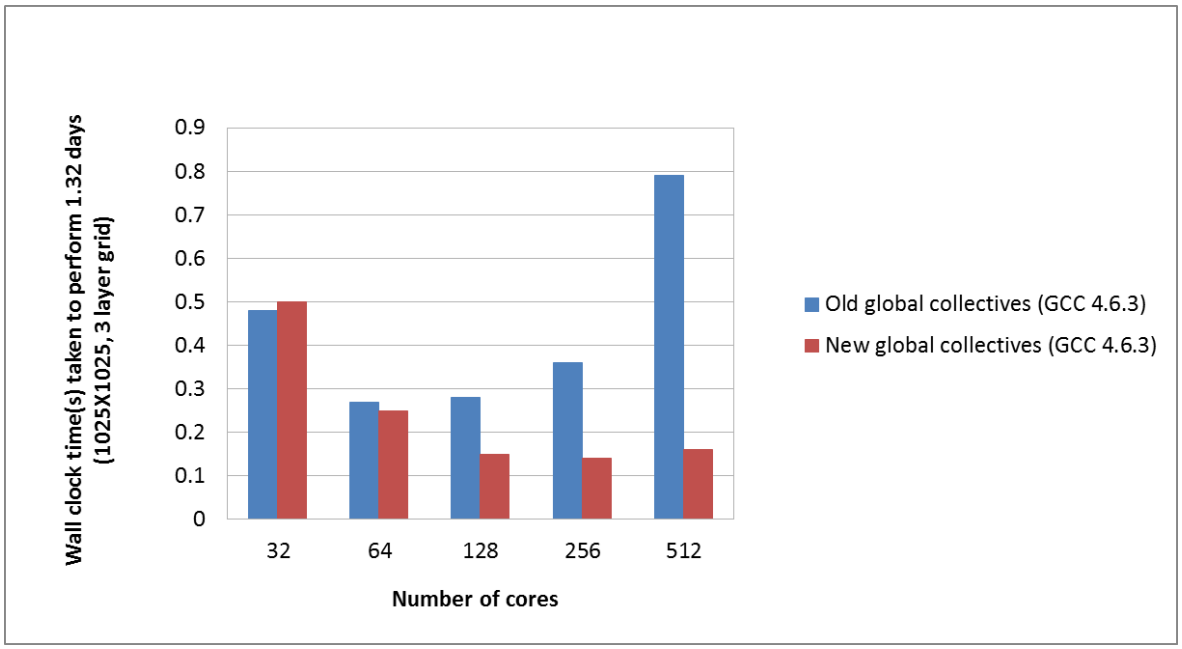

Figure 3: Comparison of the original and improved global collectives within PEQUOD.

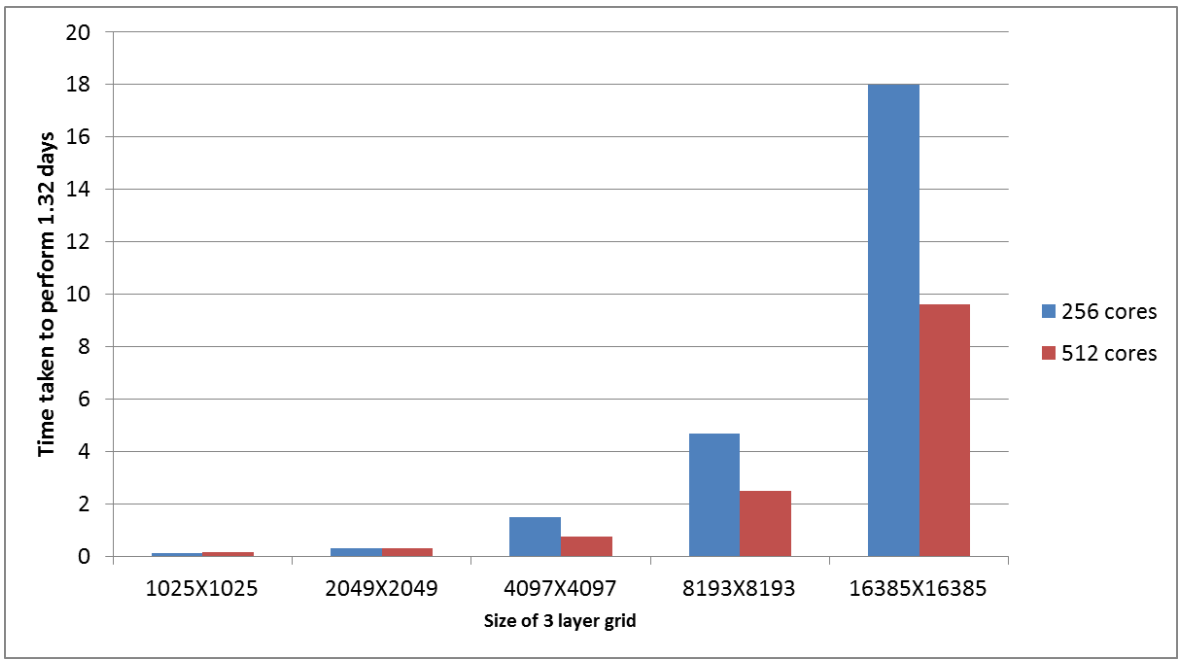

Figure 4: Weak scalability of PEQUOD with improved global collectives.

For strong scalability the performance of the new implementation has been demonstrated to be consistent up to problem sizes of 16385 (this is the maximum tested). These results are shown in Figure 4, with all runs being performed with code compiled using GCC 4.6.3.

4 Improving the I-O in Cfoam-CABARET

This section will summarise the development for the introduction of MPI-IO in the Cfoam-CABARET code. This will be for the parts of the code relevant to producing the restart and visualisation data files. The input grid produced from OpenFOAM will remain the same. Output files are written in the directories processor* as ResCells000000n and ResFaces000000n, where n is related to the time step and * denotes the number of the MPI process. The aim of the work will be to implement MPI-IO such that only a single file is written to ResCells000000n and ResFaces000000n, instead of there being one set of files for each process.

Furthermore, these data files are also used in the post-processing application outFoamX, to produce visualisation data as a series of *.plt files which are then suitable for reading by Tecplot360. Therefore both codes Cfoam-CABARET and outFoamX will need to be updated for MPI-IO. The overall objective will be to reduce the total number of files required.

4.1 Implementation of MPI-IO

Within Cfoam-CABARET for writing restart and visualisation data files ResCells000000n and ResFaces000000n files, the subroutine WriteRes was updated to use MPI-IO such that data for the individual cells and faces may be written to a single file. To enable restarts from the new data file, the subroutine ReadRestartFile was also updated to use the same file structure and MPI-IO. This was fairly straightforward to implement by using **MPI_FILE_READ** and **MPI_FILE_WRITE**, together with existing information from elsewhere in the code for definitions of the data off-sets.

To implement MPI-IO within outFoamX further work was required, mainly due to the post-processing required for Tecplot360. In the initial outFoamX application, the separate files ResCells000000n and ResFaces000000n were firstly read in for each process, then internal post-processing was performed and finally the data was gathered to the master process for writing as single .plt files for input to Tecplot360.

To update outFoamX for MPI-IO the subroutines GatherOutputFromAllProcs, PostProcessor and ReadCubes were all modified. The newly developed code enables data to be read from single input files for ResCells000000n and ResFaces000000n and then output in parallel to single .plt files for Tecplot360.

4.2 Benefit of MPI-IO in Cfoam-CABARET

The addition of MPI-IO to Cfoam-CABARET will be essential for scalable IO on HPC systems when using large numbers of MPI processes. For less than 1000 MPI processes with a 2.5×10^7 cell test case, the MPI-IO proved 20% slower than the original method. However, if many post-processing files are required then MPI-IO is required in order to prevent too many files being accessed at run-time. Furthermore, for more than 1000 processes MPI-IO performance is at least as good as the original approach and more importantly, it eliminates the problems with having many thousands of files present.

5 Conclusion

For this project, two very important CFD applications have been upgraded: one which is used to study mesoscale eddy processes in ocean modelling - PEQUOD (the Parallel Quasi-Geostrophic Model) and the other, Cfoam-CABARET for quasi-DNS simulations for aeroacoustics. Both use the CABARET finite difference scheme to resolve the advection equations. In PEQUOD, the global collectives for the parallel 1D tri-diagonal solver were updated by implementing the hardware optimised collective **MPI_ALLTOALL**. For a representative grid size of 1025, this now enables a 2 times speedup and good weak scaling on a fixed number of grid points per MPI task basis. For Cfoam-CABARET, MPI-IO was introduced for the restart and Tecplot360 visualisation data files. This eliminates the requirement for many thousands of files to be present at run-time and enables more practical use of the application on HECToR and future HPC architectures.

Future results obtained using the PEQUOD code will be beneficial for several communities. Climate modellers will be interested for understanding how internal nonlinear dynamics of the ocean can contribute to the climate variability. Ocean observers will be interested in knowing what are the key features and properties of the large-scale low-frequency variability and the eddy field that needs to be accurately estimated. Ocean modellers who work with comprehensive general circulation models might be able to enhance their non-eddy-resolving or partially eddy-resolving codes; results from PEQUOD will provide some guidance. Theoreticians will also benefit because the issues of the intrinsic large-scale low-frequency variability and the associated eddy effects are very fundamental ones.

Many computational research papers deal with various aspects of the quasigeostrophic turbulence, and there is growing demand for more turbulent and dynamically realistic solutions. There is a set of very interesting and important results that are waiting to be revisited and upgraded, with a completely new level of dynamical realism that can only be achieved with PEQUOD and HECToR, This is particularly relevant to the NERC project (NE/H020837/1) - "A new approach to parameterising ocean eddies: energetics, conservation and flow stability", October 2010-September 2013.

For Cfoam-CABARET and jet-flap-noise, this dCSE project will enhance the machinery required to answer a few important scientific questions. One of them will be "What is the effect of fine-scale-flow structures on far-field noise in the audible range of frequencies?" With the enhanced numerical model based on the unstructured CABARET and HECToR, this will cope with a significant increase in the grid resolution, so that this question may be answered. The EPSRC projects (EP/I017747/1) "Flap Noise", April 2011-September 2013 and (EP/I017771/1) "Aerodynamics and aeroacoustics of complex geometry hot jets", November 2011- September 2015.

On HECToR Phase 2b, 20,000kAUs were used with CABARET and so far more than 17,000kAUs have been used on Phase 3. Under a recent Flap Noise RAP Top Up award a further 19,600 kAUs were received, which together with the remaining 3,000k AUs awarded under the original EPSRC Flap Noise project will be used by September 2013.

6 Acknowledgments

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

Thanks also to the lead developer of PEQUOD, Dr James Maddison, University of Oxford and lead developer of Cfoam-CABARET Dr Vasily Semiletov, University of Cambridge.

References

[1] "Compact Accurately Boundary Adjusting high-REsolution Technique for Fluid Dynamics", S.A. Karabasov and V.M. Goloviznin, J. Comput.Phys., 228(2009), pp7426-7451.

[2] "CABARET in the Ocean Gyres", S.A. Karabasov, P.S. Berloff and V.M. Goloviznin, Journal of Ocean Modelling, 30 (2009), p55168.

[3] "Geophysical fluid dynamics", J. Pedlosky, Springer-Verlag, 2nd edition, 1987.

[4] "Atmospheric and oceanic fluid dynamics: Fundamentals and large-scale circulation", G. K. Vallis, Cambridge University Press, 2006.

[5] "Understanding Jet Noise", 'Visions of the future', S.A. Karabasov, issue of Phil. Trans. of R.Soc. A: Mathematical, Physical and Engineering Sciences, August 13, 2010, 368, pp3593-3608, doi:10.1098.

[6] "Jet Noise - Acoustic Analogy informed by Large Eddy Simulation", S.A. Karabasov, M.Z. Afsar, T.P. Hynes, A.P.Dowling, W.A. McMullan, C.D. Pokora, G.J. Page, and J.J. McGuirk, AIAA Journal, 2010 (July), Volume 48, Number 7.

[7] "Formulation and users' guide for Q-GCM Version 1.3.1", A. McC. Hogg, Blundell J. R., W. K. Dewar, and P. D. Killworth, 2006.

[8] http://www.hector.ac.uk/cse/distributedcse/reports/cabaret/

[9] "Parallel solution of tridiagonal systems for the Poisson equation", U. Schumann and M. Strietzel. Journal of Scientific Computing, 10(2)(1995), pp181-190.