# Enabling Adaptive Mesh Refinement for Single Components in ECHAM6

Yumeng Chen*, Konrad Simon, and Jörn Behrens

Center for Earth System Research and Sustainability, Department of Mathematics, Universität Hamburg, Hamburg 20144, Germany,
yumeng.chen@uni-hamburg.de

**Abstract.** Adaptive mesh refinement (AMR) can be used to improve climate simulations since these exhibit features on multiple scales which would be too expensive to resolve using non-adaptive meshes. In particular, long-term climate simulations only allow for low resolution simulations using current computational resources. We apply AMR to single components of the existing earth system model (ESM) instead of constructing a complex ESM based on AMR. In order to compatibly incorporate AMR into an existing model, we explore the applicability of a tree-based data structure. Using a numerical scheme for tracer transport in ECHAM6, we test the performance of AMR with our data structure utilizing an idealized test case. The numerical results show that the augmented data structure is compatible with the data structure of the original model and also demonstrate improvements of the efficiency compared to non-adaptive meshes.

**Keywords:** AMR · data strucuture · climate modeling

## 1 Introduction

Atmospheric components of earth system models used for paleo-climate simulations currently utilize mesh resolutions of the order of hundreds of kilometers. Since hundreds of components need to be computed on each mesh node, computational resources are limited even with such low resolution. However, relevant processes, such as desert dust or volcano ash clouds, cannot be resolved with sufficient fidelity to capture the relevant chemical concentrations and local extent. Improving resolution even in one single component should improve the general simulation result due to more accurate interactions among different components [1].

AMR dynamically refines a given mesh locally based on user-defined criteria. This approach is advantageous, when local features need higher resolution or accuracy than the overall simulation, since the computational effort scales with the number of mesh nodes or cells. Compared to uniform refinement fewer cells are added for the same quality of results. Berger and Oliger[2] introduced this approach for hyperbolic problems using a finite difference method on structured meshes. Since then the method has gained popularity due to its applicability in

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

a variety of multi-scale problems in computational physics. However, implementation of numerical algorithms on adaptive meshes is more complicated than on uniform meshes. In order to ameliorate the difficulty, various established AMR software implementations are available [3–8]. These packages can generate meshes on complex geometries and provide tools to manage AMR. For example, Jablonowski et al.[9] proposed a general circulation model on the sphere using the AMR library by Oehmke and Stout[5]. McCorquodale et al.[10] built a shallow water model on a cubed-sphere using the Chombo library[8]. However, it is difficult to incorporate these so-called dynamical cores into current climate models for imminent use.

We enable adaptive mesh refinement (AMR) for selected constituents of an atmospheric model, ECHAM6 [11], with a tree-based data structure. Unlike many other AMR implementations that use specially designed mesh data structures and implement numerical schemes in their context our approach aims at a seamless integration into an existing code. Thus, the data structures presented in this paper remain transparent to the hosting program ECHAM6, while enabling locally high resolution. The most natural data structures for efficient AMR implementation are tree-based, more precisely forest of trees data structures [7]. The forest of trees data structure is a collection of trees, which allows the flexibility of adding or deleting cells on the mesh. On the other hand, as an atmospheric general circulation model that solves the equations of atmospheric dynamics and physics on non-adaptive meshes, ECHAM6 uses arrays as its predominant data structure. In order to seamlessly incorporate AMR into individual components of the hosting software ECHAM6, we use the forest of trees data structure combined with a doubly linked list such that it can take arrays as input, while retaining flexibility of the tree structure. We also combine the forest of trees data structure with an index system similar to [12] to uniquely identify individual cells on adaptive meshes and facilitate search operations.

We describe our implementation of AMR in Section 2, which includes the description of our indexing system, data structure and the AMR procedure. In Section 3, we present the transport equation as an example to demonstrate the performance of our data structure for AMR on an idealized test case. We conclude and plan our future work in Section 4.

## 2 Method

We explore the use of the forest of trees data structure to incorporate an AMR approach into ECHAM6. Our implementation is similar to the forest of trees by Burstedde et al.[7], but it is less complicated because our application is limited to 2-D structured rectangular meshes. In order to facilitate the implementation, we use the index system by [12].

### 2.1 Index system

ECHAM6 uses arrays for rectangular mesh management. 2-D arrays are indexed by pairs and each entry of the arrays represents a cell on the mesh. The use of

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

an index system greatly helps the construction of numerical schemes for solving partial differential equations and the search of adjacent cells on the mesh.

If we construct the mesh by recursively refining the cells on the domain starting from one cell that covers the whole domain, the index of each cell can be computed correspondingly. After one refinement of the cell $(i, j)$, the resulting four cells have indexes $(i, j = 0, 1, 2, \ldots)$:

$$\begin{array}{cc} (2i, 2j+1) & (2i+1, 2j+1) \\ (2i, 2j) & (2i+1, 2j) \end{array} \tag{1}$$

If the mesh is coarsened, every four fine cells coalesce and the index of the resulting coarse cell is:

$$(\lfloor \frac{i}{2} \rfloor, \lfloor \frac{j}{2} \rfloor) \tag{2}$$

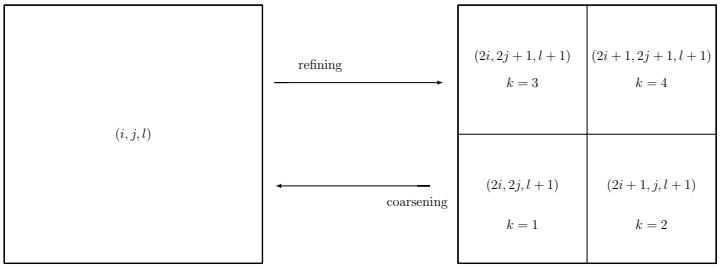


**Fig. 1.** Illustration of the refinement and coarsening process of a single cell and the corresponding index. $k$ represents the index of the children in the tree

This works perfectly on uniformly refined meshes as all cell indices increase proportionally with each refinement. Thus, each pair can uniquely define a cell. However, conflicts can occur on adaptive meshes, where cells with different levels of refinement appear at the same time. Such conflicts can cause ambiguous cell identification, which in turn may result in the use of wrong values for numerical schemes leading to erroneous numerical results. We adopt the concept of an additional index for the refinement level, $l$, from [12]. The idea can be illustrated in 1-D cases. If the mesh is generated by recursively refining all cells on the domain from one cell covering the whole domain, we can get the number of cells $nx = 2^l$, where $l$ is the number of refinements. We define the number of refinements as refinement level:

$$l = \log_2 nx \tag{3}$$

The refinement level is defined for each cell. Once a cell is refined, the refinement level of this cell increases by one. Hence, on uniformly refined meshes, all cells have the same refinement level. Our goal is to enable adaptivity on existing meshes. Since the number of cells on the existing mesh is not necessarily an even

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

number, we take $\lceil \log_2 nx \rceil$ as the refinement level, $l$, such that $nx \leq 2^l$. This concept can be extended to 2-D cases:

$$l = \lceil \log_2 \max(nx, ny) \rceil \tag{4}$$

where $nx$ and $ny$ are the number of cells of the input mesh in each dimension, respectively. Since cells on adaptive meshes have various refinement levels, the triple $(i, j, l)$ forms the index of a cell such that no conflicts can occur. After refining the cell $(i, j, l)$, the index becomes:

$$(2i + a, 2j + b, l + 1) \tag{5}$$

where $a = 0, 1$ and $b = 0, 1$. If four cells are coarsened into one, the four cells coalesce and the index of the resulting cell is:

$$(\lfloor \frac{i}{2} \rfloor, \lfloor \frac{j}{2} \rfloor, l - 1) \tag{6}$$

Such index system guarantees that each cell owns a unique index on the mesh. The system is shown in Figure 1.

### 2.2 Data structure

Without adaptivity, a cell is treated as an entry of a 2-D array on 2-D meshes. However, arrays lack the flexibility to organize cells on adaptive meshes. In order to enable adaptivity with existing meshes, it is natural to adopt the idea of a forest of trees to manage AMR [7]. A schematic illustration is shown in Figure 2. A forest is a set of trees. In our application, a tree node represents a cell.

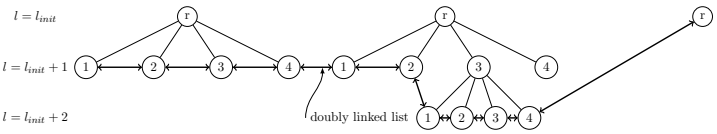


**Fig. 2.** Illustration of the data structure. The numbers in the tree node represent the indices of children. $l$ is the refinement level, $l_{init}$ is the initial refinement level and $r$ represents the root of each tree. The two way connectors are a representation of a doubly linked list. Each tree node represents a cell and the leaves of the trees are active cells on the computational mesh. A mesh corresponding to this tree is shown in Figure 3.

Each entry of the input array is a root of a tree. Hence, the number of trees in the data structure depends on the number of cells on the input mesh. The input array can also be viewed as a forest, where each tree just has one root. The roots of the trees are presented as a 1-D array in our current implementation. This reduces the data structure to arrays as in ECHAM6 for non-adaptive meshes. If

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

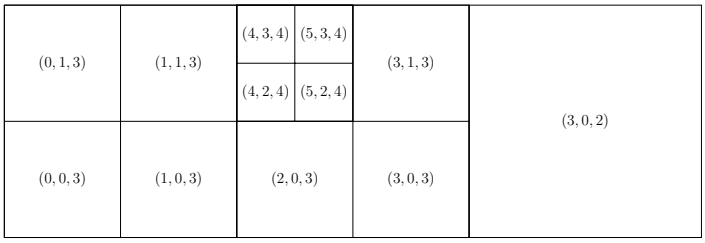


**Fig. 3.** The mesh organized by the forest of trees shown in Figure 2. The index of each cell on the adaptiveh mesh avoids the conflicts at different refinement levels. The initial refinement level, $l_init$, is 2

the input mesh has $nx \times ny$ number of cells, where $nx$ and $ny$ is the number of cells in each dimension, the index of each cell in the forest is $nx \times j + i$, where $(i, j)$, with $i = 0, \ldots, (nx - 1)$, and $j = 0, \ldots, (ny - 1)$, is the index of the cell in the input mesh. This is the same as the row-wise ordering that transforms values on 2-D meshes into 1-D vectors for numerical computation. We maintain the index of each cell from the (original) input mesh and compute the refinement level of cells in the input mesh by equation 4. The refinement level of cells in the roots of the trees is defined as initial refinement level, $l_{init}$. The refinement process divides a cell into four cells, which is equivalent to adding four children to the current tree node of the tree. The children become leaves of the tree and appear on the mesh as a cell and we refer these leaves as active tree nodes, while the parent is non-active tree node as it is not treated as a cell on the mesh. The four children of each tree node in the tree are indexed by $k$. It is necessary to relate, $k$, with the index system of cells, $(i, j, l)$. Using $a, b$ in equation 5, $k = a + 2b + 1$. An example of index $k$ in cells after refinement is shown in Figure 1 and the index of children in the tree is shown in Figure 2. The index $a$ and $b$ can be recovered from $(i, j, l)$:

$$\begin{aligned} a &= i - 2\lfloor \frac{i}{2} \rfloor \\ b &= j - 2\lfloor \frac{j}{2} \rfloor \end{aligned} \tag{7}$$

Correspondingly, as a reverse operation of mesh refinement, the coarsening is equivalent to deleting four leaves that share the same parent. Here, the parent node is again marked as active tree node, which appears as a cell on the mesh. The data structure is intuitive for adaptive meshes and enables a simple search algorithm on rectangular meshes with the help of our index system. Searching a cell with the index $(i, j, l)$ requires $l - l_{init}$ operations, which is the same as the depth of the tree node in the tree. This is particularly useful as the numerical schemes for solving PDEs usually need values at adjacent cells. While a forest of trees is a suitable data structure for adaptive refinement and coarsening, the numerical computation of PDEs usually requires (many) traversals of all active

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

cells of the mesh. It is inefficient to traverse each of the trees just to access the leaves. Therefore, a doubly linked list is used to connect all the leaves as shown in Figure 2. A linked list can meet the requirement for repeated traversals of the mesh. Similar to arrays, only $n$ operations are required for the traversal of the whole mesh, where $n$ is the number of cells on the mesh. Also, the tree nodes on the doubly linked list can be added or removed flexibly and therefore it is well suited for AMR.

### 2.3 Adaptive algorithm and refinement strategy

The effectiveness of the AMR also depends on the refinement procedure. Our refinement strategy is inspired by the adaptive semi-Lagrangian algorithm in [13] and is similar to most AMR procedures [14–16]. Assuming a one level time stepping method is used, the implementation involves two meshes. One mesh, $M^n$, keeps information of the $n^{\text{th}}$ time step, and another, $M^{n+1}$, keeps the information of the $(n+1)^{\text{st}}$ time step. The computation of $nt$ time steps are summarized in algorithm 1 and algorithm 2. ECHAM6 has an independent module for tracer transport. If the AMR method is integrated into ECHAM6, ECHAM6 would parse information on the coarse meshes in the form of arrays to the AMR module. The information on coarse resolutions are supposed to be interpolated.

**Data:** $M^n$
Initialize the input mesh $M^n$;
Perform **mesh refinement procedure** on mesh $M^n$ based on the initial condition of the PDE;
Recompute the initial condition on refined mesh $M^n$;
Generate mesh $M^{n+1}$ for new time step, which is a copy of mesh $M^n$;
**for** $n = 1$ **to** $nt$ **do**
    Perform **mesh refinement procedure** on mesh $M^{n+1}$;
    Solve the PDE and store results on mesh $M^{n+1}$;
    Regenerate mesh $M^n$ as a copy of mesh $M^{n+1}$ for next time step;
**end**

**Algorithm 1:** The process of solving the PDEs with AMR. $nt$ is the total number of time steps, and the input data is from an array. The mesh refinement procedure mentioned above is iterative in itself. The details of the step **mesh refinement procedure** at each time step can be found in Algorithm 2.

We limit the differences of refinement levels between adjacent cells to guarantee a relatively smooth resolution variation since abrupt resolution changes can result in artificial wave reflections [17]. This also facilitates the search for adjacent cells since the number of adjacent cells for each cell is less or equal to two.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

**Data:** $M$
$numofiter = 0$;
$numofcoarsened = numofrefined = 1$;
**if** $M == M^{n+1}$ **then**
  Solve PDE by a first-order scheme (predictor step);
**end**
**while** $numofcoarsened/ = 0$ **do**
  Mark cells that will be coarsened according to a coarsening criterion;
  Remove coarsening marker for those cells with neighbors differing by more than one level;
  Update mesh and obtain number of coarsened cells $numofcoarsened$;
**end**
**while** $numofiter < N$ ***or*** $numofrefined/ = 0$ **do**
  **if** $M == M^{n+1}$ **then**
    Solve PDE by a first-order scheme (predictor step);
  **end**
  Mark cells that will be refined according to a refinement criterion;
  Mark those cells with neighbors differing by more than one level for refinement;
  Update mesh and refinement levels of cells and obtain number of refined cells $numofrefined$;
  $numofiter = numofiter + 1$;
**end**

**Algorithm 2:** The **mesh refinement procedure** in each time step. $N$ is the maximum number of iterations, $numofcoarsened$ is the number of cells coarsened in the current iteration, $numberofrefined$ is the number of cells refined in the iteration, $numofiter$ records the total number of iterations.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

## 3 Results

We test our data structure for adaptive mesh management with an idealized moving vortices test case [18]. The test case is designed to test transport schemes on the sphere. We generate the initial condition of tracer concentration and velocity as arrays and parse these into our data structure such that we can use our own implementation instead of adding the test case into ECHAM6. We use the Flux-Form Semi-Lagrangian (FFSL) [19] transport scheme in ECHAM6, which is a finite volume scheme that conserves mass and permits long time steps. The scheme uses an operator splitting technique, which computes 2-D problems by applying a 1-D solver four times. Here, we choose the cell-integrated semi-Lagrangian scheme [20] as the 1-D solver, where a piecewise parabolic function is used as reconstruction function.

### 3.1 Moving vortices test case

In this test case, two vortices are developing at opposite sides of the sphere while rotating around the globe. The test case simulates 12 days of model time and has the benefit that an analytical solution is available. The velocity field is given by:

$$\begin{aligned} u =& a\omega_r\{\sin\theta_c(t)\cos\theta - \cos\theta_c(t)\cos[\lambda-\lambda_c(t)]\sin\theta\} \\ &+ u_0(\cos\theta\cos\alpha + \sin\theta\cos\lambda\sin\alpha), \\ v =& a\omega_r\cos\theta_c(t)\sin[\lambda-\lambda_c(t)] - u_0\sin\lambda\sin\alpha, \end{aligned} \tag{8}$$

where $u_0$ is the velocity of the background flow that rotates the vortices around the globe, $(\lambda, \theta)$ is the longitude and latitude, $(\lambda_c(t), \theta_c(t))$ is the center of the current vortex. In our experiment, we set $u_0 = \frac{2\pi a}{12\text{days}}$, where $a$ is the radius of the earth and $(\lambda_c(0), \theta_c(0)) = (\frac{3\pi}{4}, 0)$. The computation of the position of the vortex center can be found in [18]. $\omega_r$ is the angular velocity of the vortices:

$$\omega_r = \begin{cases} \frac{3\sqrt{3}u_0\,\text{sech}^2(r)\tanh(r)}{2ar} & r \neq 0 \\ 0 & r = 0 \end{cases} \tag{9}$$

where $r = r_0\cos\theta'$. $\theta'$ is the position of the rotated sphere where the vortex center is at the north and south poles and $r$ is the radial distance of the vortex. We set $r_0 = 3$.

The moving votices test case is particularly useful but hard test for AMR schemes because the tracer does not only appear in a limited area, which is common in climate simulations. It covers a large area of the globe and the concentration of the tracer is:

$$\rho = 1 - \tanh[\frac{r'}{\gamma}\sin(\lambda' - \omega_r t)] \tag{10}$$

where $r' = r_0\cos\theta_d$, and $\theta_d$ is the departure position of background rotation and $\lambda'$ is the departure position on the rotated sphere where the vortices' centers are at the poles at $t = 0$.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

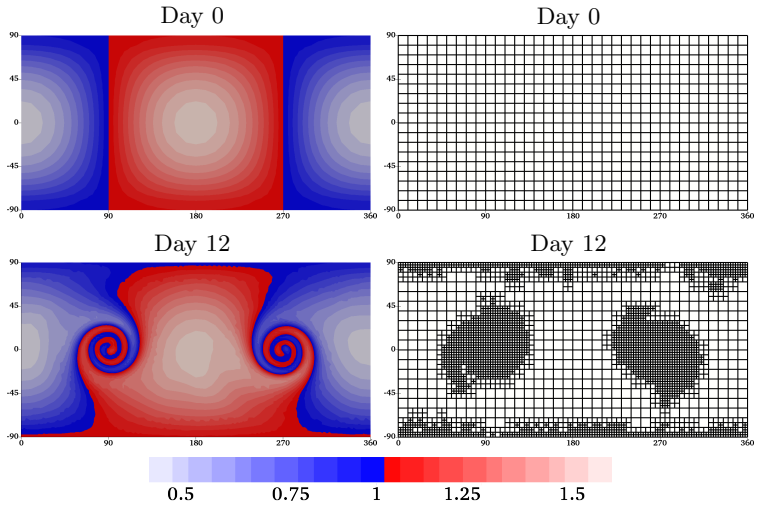


**Fig. 4.** Numerical solution of the moving vortices test case with base resolution of 10 degrees and 2 levels of refinement which leads to fine grid resolution 2.5 degrees. The left column shows the numerical solution and the right column shows the corresponding mesh evolution.

We choose to set the flow orientation to $\alpha = \frac{\pi}{4}$ considering that this could be the most challenging test set-up for operator splitting schemes [14]. Since the vortices are moving around the globe and the mesh has different sizes around the sphere, the maximum Courant number changes with time. The maximum Courant number appears when the vortices move close to the poles. We use a maximum Courant number of 0.96 ~~and 5.3 respectively~~. A snapshots of the numerical solution on adaptively refined meshes is shown in Figure 4.

Similar to [14], we use a gradient based criterion. Since we use a cell-based AMR, each cell is assigned an indicator value, $\theta$. This value is computed as the maximum of gradients in cell mean values with respect to the four adjacent cells:

$$\theta = \max\left(\frac{\partial \rho}{a \cos\theta \partial \lambda}, \frac{\partial \rho}{a \partial \theta}\right) \tag{11}$$

If $\theta > \theta_r$, the algorithm refines the cell; if $\theta < \theta_c$, the algorithm coarsens the cell. The threshold of $\theta_r = 1$ and $\theta_c = 0.95$ is chosen for this test case. This criterion is justified by the fact that flux-form semi-Lagrangian schemes show little numerical diffusion when strong variations in the tracer are highly resolved. Still, only limited areas are covered by fine resolution cells. The refinement criterion successfully captures areas where vortices are located because strong distortion of the tracer distribution leads to large gradient in tracer concentrations $\rho$. Due to the higher resolution around the poles and the highly distorted velocity field, the mesh is refined around the poles even if the vortices do not directly cross

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

the poles. This leads to extra high resolution cells on adaptive meshes. A better representation of the velocity field on refined meshes still helps to get more accurate results.

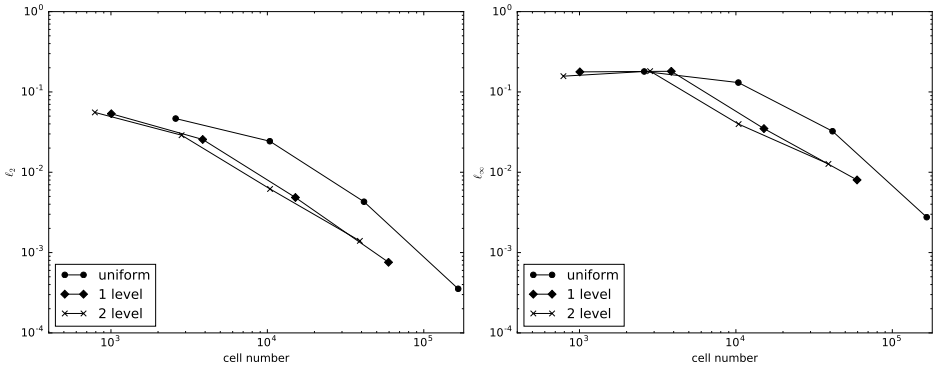


**Fig. 5.** Convergence rate of the numerical solution with respect to the cell number on the domain. The left one shows the $\ell_2$ and the right shows the $\ell_\infty$-norm

The convergence rate in Figure 5 shows that, although the results on the non-adaptive mesh can have the best accuracy, similar accuracy can be achieved with fewer cells using adaptive meshes. It is expected that the numerical result on the adaptive mesh is less accurate because the initial condition is defined on a coarser resolution. Furthermore, the $\ell_2$ and $\ell_\infty$ norms are a measure of the global accuracy and the results on the coarse resolution have an impact on the error. Nevertheless, AMR shows improvement in the accuracy compared with the non-adaptive mesh on coarse resolutions. The results are consistent with the results from [14].

Figure 6 show that the wall clock time for tests on adaptive meshes is less than on uniform meshes with the same finest resolution. The test is run in serial. The wall clock time is measured on Debian 3.2 operating system and the machine has 4 Intel Xeon X5650 CPUs, each of which has 6 cores with a clock speed of 2.67 GHz and 12 MB L3 cache. The machine also has a RAM of 24 GB. It is worth noting that the wall clock time is affected by various factors and is not an accurate measure of the effectiveness of AMR. In particular, the implementation is not fully optimized. A more objective measure is that AMR runs use fewer cells compared to uniform meshes with the same resolution. The cell number shown in Figure 6 represents the average number of cells over all time steps. For this test case the ratio of cell number on adaptive meshes to cell number on uniform meshes remains approximately constant even with different finest resolutions. A possible explanation is that the vortices develop only after some simulation time. Therefore, the (uniform) coarse mesh cell number dominates

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

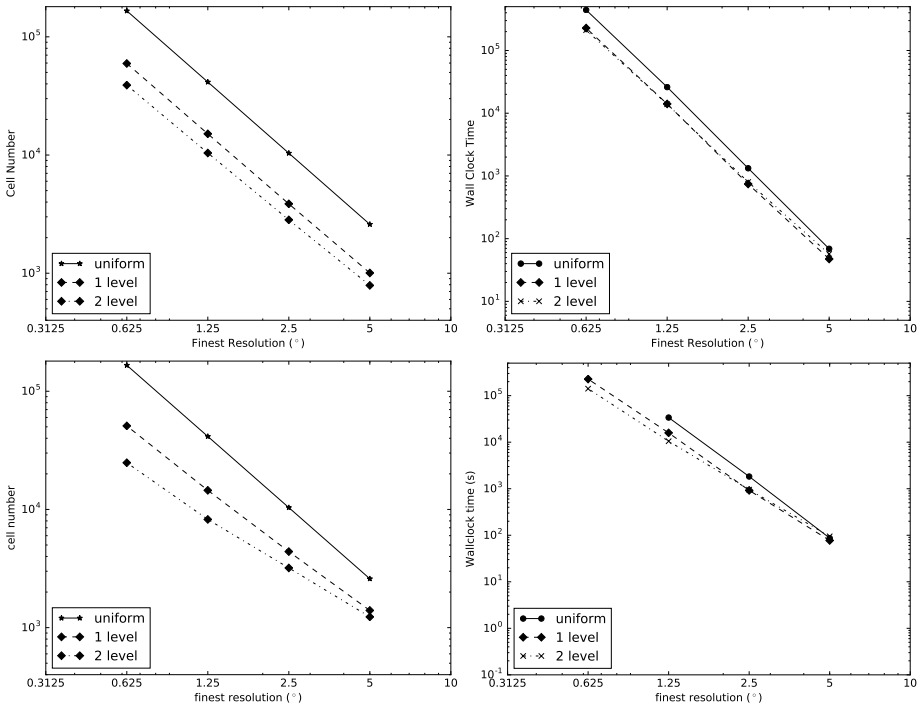


**Fig. 6.** Used time and cell number of the numerical scheme in the moving vortices test case using a *loglog* plot. The upper left graph shows the cell number on the mesh with the same finest resolution and the upper right graph shows the time used on different refinement levels with the same finest resolution in serial in moving vortices test cases. The lower left and lower right is the cell number and the time consumption for solid body rotation test case.

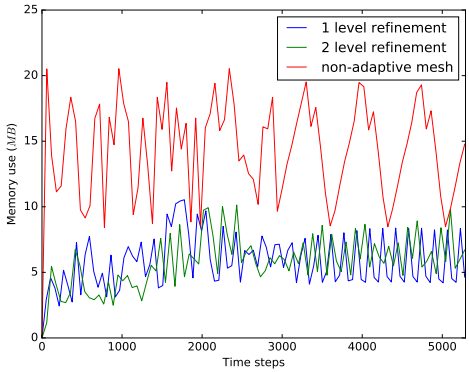


**Fig. 7.** Time evolution of the total heap memory usage for different refinement levels using moving vortices test case with a maximum resolution of 2.5° on the mesh

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

the average over time. The cell number and the time consumption is also quite problem dependent. In the cross-pole solid body rotation test case by [21], the cell number shows a different variation in terms of resolutions.

It could be argued that the cell number is not the only a measure of the usefulness of AMR. Compared with the non-adaptive meshes, the data structure and extra steps that allows us to enable AMR can lead to overhead, as stated in the Algorithm 2. However, with careful choice of the refinement criterion, fewer memory and less time is required relative to the implementations on non-adaptive meshes. This is because numerical schemes use less time with fewer cells and the overhead can be compensated as shown in Table 1. Additionally, it is expected that an optimized implementation has similar behavior while the specific values may differ. In [7] successful optimization and parallelization of forest of trees data structures could be demonstrated.

**Table 1.** The time used for different components of the adaptive mesh refinement. Update represents the time used for FFSL, velocity is the time used for updating the velocity for next time step and update mesh from $M^n$ to $M^{n+1}$, refine is the extra time used for refinement, including the predicting time and mesh refinement.

| Finest | Zero level Refinement | | One level Refinement | | | Two level Refinement | | |
|---|---|---|---|---|---|---|---|---|
| Resolution | update | velocity | update | refine | velocity | update | refine | velocity |
| 5° | 8.162 | 60.80 | 3.33 | 30.37 | 17.81 | 2.65 | 36.84 | 21.64 |
| 2.5° | 1193.81 | 132.56 | 466.45 | 291.14 | 52.89 | 459.91 | 338.74 | 38.55 |
| 1.25° | 2216.10 | 23883.67 | 937.61 | 8977.73 | 5843.90 | 622.01 | 7150.97 | 6111.59 |

Compared with wall clock time, the cell number is more closely related to the memory usage. As shown in Fig 7, the adaptive mesh runs use significantly less memory compared with non-adaptive mesh runs. Similar memory usage appears on all maximum resolutions.

The test case shows that forest of trees data structure is able to handle AMR with various initial refinement levels. Although the implementation is not fully optimized, benefits of AMR can still be observed. With the current refinement criterion, AMR achieves better accuracy with less memorie and time usage. AMR runs require less wall clock time and fewer cells than uniformly refined simulations at the same finest resolution. The results also show that the forest of trees data structure can successfully handle the information from arrays.

## 4 Summary and Future Work

We explore the use of a forest of trees data structure to enable AMR in single components of an existing atmospheric model. Our data structure is tested on a

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

tracer transport scheme used in the atmospheric model ECHAM6 for an idealized test case.

We show that our data structure is compatible with the arrays used in ECHAM6. Compatibility between the array data structure used in ECHAM6 and the forest of trees is guaranteed as the forest of trees can simply be reduced to an array on non-adaptive meshes. We combine a forest of trees data structure with an indexing system for mesh management. The data structure is equivalent to arrays on the uniform meshes since no leaves are present on the trees. With the help of a doubly linked list the traversal of potentially adaptively refined meshes is the same as a traversal of an array and the operation for finding arbitrary cells by index is limited by the level of refinements for adaptivity. Therefore, the asymptotical computational complexity of the numerical scheme on adaptive meshes does not increase over the scheme on non-adaptive meshes.

We use a simple gradient based refinement criterion for our numerical test. Although the scheme is not fully optimized and parallelized less computation time is used for AMR while similar accuracy can be achieved using fewer cells - provided the refinement criterion is chosen with care. The results of the AMR runs show less memory and time use compared to non-adaptive meshes.

## Acknowledgment

This work was supported by German Federal Ministry of Education and Research (BMBF) as Research for Sustainability initiative (FONA); www.fona.de through Palmod project (FKZ: 01LP1513A).

## References

1. Aghedo, A.M., Rast, S., Schultz, M.G.: Sensitivity of tracer transport to model resolution, prescribed meteorology and tracer lifetime in the general circulation model echam5. Atmospheric Chemistry and Physics **10**(7) (2010) 3385–3396
2. Berger, M.J., Oliger, J.: Adaptive mesh refinement for hyperbolic partial differential equations. Journal of Computational Physics **53**(3) (1984) 484–512
3. Berger, M.J., LeVeque, R.J.: Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. SIAM J. Numer. Anal. **35** (1998) 2298–2316
4. MacNeice, P., Olson, K.M., Mobarry, C., De Fainchtein, R., Packer, C.: Paramesh: A parallel adaptive mesh refinement community toolkit. Computer physics communications **126**(3) (2000) 330–354
5. Oehmke, R.H., Stout, Q.F.: Parallel adaptive blocks on a sphere. In: PPSC. (2001)
6. Behrens, J., Rakowsky, N., Hiller, W., Handorf, D., Läuter, M., Päpke, J., Dethloff, K.: amatos: Parallel adaptive mesh generator for atmospheric and oceanic simulation. Ocean Modelling **10**(1–2) (2005) 171–183
7. Burstedde, C., Wilcox, L.C., Ghattas, O.: p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. SIAM Journal on Scientific Computing **33**(3) (2011) 1103–1133
8. Adams, M., Schwartz, P.O., Johansen, H., Colella, P., Ligocki, T.J., Martin, D., Keen, N., Graves, D., Modiano, D., Van Straalen, B., et al.: Chombo software package for amr applications-design document. Technical report (2015)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5

9. Jablonowski, C., Oehmke, R.C., Stout, Q.F.: Block-structured adaptive meshes and reduced grids for atmospheric general circulation models. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **367**(1907) (2009) 4497–4522
10. McCorquodale, P., Ullrich, P., Johansen, H., Colella, P.: An adaptive multiblock high-order finite-volume method for solving the shallow-water equations on the sphere. Communications in Applied Mathematics and Computational Science **10**(2) (2015) 121–162
11. Stevens, B., Giorgetta, M., Esch, M., Mauritsen, T., Crueger, T., Rast, S., Salzmann, M., Schmidt, H., Bader, J., Block, K., et al.: Atmospheric component of the mpi-m earth system model: Echam6. Journal of Advances in Modeling Earth Systems **5**(2) (2013) 146–172
12. Ji, H., Lien, F.S., Yee, E.: A new adaptive mesh refinement data structure with an application to detonation. Journal of Computational Physics **229**(23) (2010) 8981–8993
13. Behrens, J.: An adaptive semi-Lagrangian advection scheme and its parallelization. Mon. Wea. Rev. **124**(10) (1996) 2386–2395
14. Jablonowski, C., Herzog, M., Penner, J.E., Oehmke, R.C., Stout, Q.F., Van Leer, B., Powell, K.G.: Block-structured adaptive grids on the sphere: Advection experiments. Monthly weather review **134**(12) (2006) 3691–3713
15. Blayo, E., Debreu, L.: Adaptive mesh refinement for finite-difference ocean models: first experiments. Journal of Physical Oceanography **29**(6) (1999) 1239–1250
16. Behrens, J.: Atmospheric and ocean modeling with an adaptive finite element solver for the shallow-water equations. Applied Numerical Mathematics **26**(1-2) (1998) 217–226
17. Ullrich, P.A., Jablonowski, C.: An analysis of 1d finite-volume methods for geophysical problems on refined grids. Journal of Computational Physics **230**(3) (2011) 706–725
18. Nair, R.D., Jablonowski, C.: Moving vortices on the sphere: A test case for horizontal advection problems. Monthly Weather Review **136**(2) (2008) 699–711
19. Lin, S.J., Rood, R.B.: Multidimensional Flux-Form Semi-Lagrangian Transport Schemes. Mon. Weather Rev. **124** (1996) 2046–2070
20. Nair, R.D., Machenhauer, B.: The mass-conservative cell-integrated semi-lagrangian advection scheme on the sphere. Monthly Weather Review **130**(3) (2002) 649–667
21. Williamson, D.L., Drake, J.B., Hack, J.J., Jakob, R., Swarztrauber, P.N.: A standard test set for numerical approximations to the shallow water equations in spherical geometry. Journal of Computational Physics **102**(1) (1992) 211–224

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_5