Nested Parallelism in OpenMP with Application to Adaptive Mesh Refinement

Ragnhild Blikberg
Parallab/Department of Informatics
University of Bergen
Norway

February 2003

Thesis submitted in partial fulfillment of the requirements for the degree
Doctor Scientiarum

Acknowledgments

First of all I am grateful to my supervisor, Professor Tor Sørevik, for his guidance, support and patience, and for always making time for me when I came knocking on his door. I especially appreciate his good sense of humour, and his ways of giving help and motivation.

Thanks to the Research Council of Norway for providing three years of funding through the Strategic University Program, and to Parallab for one year of funding and for giving me unlimited access to the supercomputer facilities.

I would also like to thank my colleagues at Parallab and the Department of Informatics for being a source of knowledge in so many fields, and for providing such a friendly atmosphere.

In the beginning of September 2001, I went to the Courant Institute at New York University for a three months stay. Due to the tragic events of September 11th, I decided to interrupt the stay. Nevertheless, I would like to thank Professor Marsha J. Berger and her colleagues for giving me a warm welcome at their institute, and the opportunity to experience their stimulating environment.

I am very thankful to my parents, Gunhild and Olav, who have been a great inspiration, and have always motivated me in seeking education. My brother, Alf Ivar, has chosen a totally different career than I, global health issues for the World Economic Forum. I am very proud of the work he is doing, and thank him for being someone I can ask for advise.

Rune, thank you for showing me in tough periods that computational science is not ”the whole world”. Being together with you everyday is the most important for me, and I am really looking forward to our wedding in August this year.

Made weak by time and fate, but strong in will
To strive, to seek, to find, and not to yield...
Ulysses, Alfred, Lord Tennyson

Contents

List of papers:

[A] Ragnhild Blikberg and Tor Sørevik, ”Nested parallelism: Allocation of threads to tasks and OpenMP implementation”, *Journal of Scientific Programming*[1], 9(2,3):185-194, 2001.

[B] Ragnhild Blikberg and Tor Sørevik, ”Costs and Savings of Adaptive Mesh Refinement”, Technical report no. 225 at Department of Informatics, University of Bergen, November 2001.

[C] Ragnhild Blikberg, ”Nested parallelism applied to AMRCLAW”, in *Proceedings of EWOMP’02, Fourth European Workshop on OpenMP*, Rome, September 2002.

[D] Ragnhild Blikberg and Tor Sørevik, ”Computing the Shallow Water Equations on a Sphere by Finite Volume Methods”, unpublished manuscript, February 2003.

[1] Due to copyrights, the publisher objects to include their `pdf`-file in this thesis. Therefore this article has a different layout in this thesis. The text, however, is identical.

1 Introduction

Solving partial differential equations numerically for real life problems has always been considered computationally demanding. As the capacity of large scale computers has increased, scientists have been able to solve increasingly larger problems. Nevertheless, the need for more efficient solution techniques and faster computers seems unlimited. Therefore, developing numerical techniques and achieving additional knowledge about utilizing super computers efficiently, will for many years to come be important in order to accommodate the increasing need for computational efficiency.

Performance improvement can be achieved in several ways. It can be achieved by minimizing the number of grid points without sacrificing accuracy, by some sort of local (adaptive) mesh refinement. Improvement of performance can also be achieved on algorithmic level by efficient solvers, for instance by iterative methods. On computer level, performance can be improved by using optimized libraries, tuning for cache effects, or by parallelization.

In this thesis, the effort has mainly been put in two kinds of performance improvement: nested parallelism and adaptive mesh refinement. In paper [A], a nesting technique for multilevel parallelization has been developed, while costs and savings of an adaptive mesh refinement application have been studied in paper [B]. The techniques of adaptive mesh refinement and nested parallelism are in themselves complex to apply, and when they are used in combination, the challenges become even harder to overcome. This is dealt with in paper [C].

Paper [D] can be seen as quite independent from the papers [A], [B] and [C]. This paper reports on our experiences with applying a finite volume method to solve the spherical shallow water equations, represented in spherical coordinates using a regular longitude-latitude discretization of the grid.

1.1 The structure of the thesis

The thesis consists of an extended introduction, followed by four papers.

The introduction is organized as follows: Section 2 contains a presentation and discussion of the development of OpenMP, with special emphasis on nested parallelism. Section 3 provides an introduction to adaptive mesh refinement (AMR), and the outcome of our study of costs and savings of AMR. Then in Section 4, the results from applying our nesting techniques to an AMR-application are given. Section 5 contains an overview of the diversity of numerical methods for computing shallow water equations on a sphere, with special attention to our finite volume approach. Plans and possible directions for future work are given in Section 6.

The main part of this thesis, the papers, are presented in the following order:

[A] ”Nested parallelism: Allocation of threads to tasks and OpenMP implementation”
[B] ”Costs and savings of adaptive mesh refinement”
[C] ”Nested parallelism applied to AMRCLAW”
[D] ”Computing the shallow water equations on a sphere by finite volume methods”

2 OpenMP and nested parallelism

There are currently two industrial standards for parallelization, the Message Passing Interface (MPI) [23] for message passing programming (MPP), and OpenMP [4] for shared memory programming (SMP). In MPP-parallelization, memory is local to each processor and explicit subroutines are used for communication (sending and receiving data) between processors. When using SMP-parallelization, program loops and sections are executed in parallel using processors that share the same memory.

MPI has, since it was introduced [23] in 1994, been widely used in the world of parallelization. MPI is known for its portability and very good scalability, but writing message passing code often requires a large effort from the user.

In recent years, OpenMP has become the new standard for SMP-parallelization. The selling arguments for OpenMP are *”Simple, Portable and Scalable SMP-Programming”* [4]. We find these arguments attractive, and discuss in the following subsections their validity, with special attention to nested parallelism.

2.1 OpenMP

In this thesis, Fortran has been used as the programming language. The following presentation and discussion of the development of OpenMP is therefore related to the Fortran version of OpenMP.

2.1.1 OpenMP version 1.0

When the OpenMP Architecture Review Board (ARB) in October 1997 released the first OpenMP specification, Fortran version 1.0, the ”simplicity” criterion was illustrated by a 40-page manual, in which the `!$OMP PARALLEL DO` and

!$OMP PARALLEL SECTIONS directives eased the programmers effort of parallelization compared to MPI-programming.

For large parallel do-loops, scalability was achieved up to a certain level. For instance, in [A] we did an experiment parallelizing matrix multiplication of large matrices on loop level, and achieved scalability up to 50 threads. However, the scalability was limited for do-loops with few loop-iterations or too little work inside the parallelized loop. For parallel sections, the scalability was limited to the number of sections, provided the work load in each section was equal. For unequal workload, the scalability was worse.

In theory, improved scalability for both parallel do-loops and sections could have been achieved by the use of nested parallelism, which was a part of the specification through the OMP_SET_NESTED subroutine. However, as also serializing nested parallelism was compliant with the specification, all vendors chose to do so in their first release.

In [A], an explicit thread programming technique used for implementing nested parallelism in OpenMP was presented. We showed theoretically and experimentally that utilizing 2-level nested parallelism when available in the problem, and for a static number of tasks, turned out to be imperative for good performance. To handle the problem of load balance, a distribution algorithm was designed, that optimally distributes P threads to N tasks for $P \geq N$. For the matrix multiplication experiment already mentioned, which in fact showed relatively good scalability for 1-level parallelism, even higher scalability was provided with our nesting techniques: up to 110 threads!

Paper [A] further discusses the shortcomings of the OpenMP 1.0 Fortran compiler directives if nesting was implemented, in particular the impossibility to set the number of threads in nested parallel regions.

2.1.2 OpenMP version 2.0

In OpenMP Fortran version 2.0, released in November 2000, there was a new clause NUM_THREADS which seemed to solve the problem we pinpointed in [A] concerning setting the number of threads in nested parallel regions.

Another shortcoming of OpenMP 1.0, that we mentioned in [16], was the restriction of only using scalars in the REDUCTION clause. It was a relief that this omission was corrected in OpenMP 2.0.

The number of manual pages of OpenMP 2.0 had now increased to 62, but the simplicity from version 1.0 was kept. The extension was mainly due to new functionalities, for instance the !$OMP PARALLEL WORKSHARE directive, which allows parallelization of array expressions in Fortran 90, and the timing functions `OMP_GET_WTIME()` and `OMP_GET_WTICK()`. We find it important for the OpenMP ARB when considering new extensions to the specification, to continue the original idea of keeping it simple for programmers who only need some basic directives to make the code run in parallel.

Unfortunately, also in OpenMP 2.0 nested parallelism is still implementation dependent. This has implied that most vendors still have chosen to serialize nested parallelism. The consequence of this is that programs relying on nesting for scalability will not be ”performance portable” when coded in OpenMP, and since performance is at the very heart of parallel programming, lack of performance portability can be held as an argument against OpenMP.

The reason for OpenMP still giving the possibility to serialize nesting, is probably that it is not straightforward to implement nesting in the compiler. It is therefore important that other compiler environments do research within this area, and that researchers investigate different ways of nesting, like the techniques presented in [A], or the hybrid methods (for instance MPI + OpenMP). This is to document the performance improvement one can achieve from nested parallelism.

Most programmers agree that writing MPI code often is much more complicated and time consuming than inserting OpenMP directives into the code. On the other hand, it is a common opinion that OpenMP does not scale as well as MPI. This is partly due to the fact that the parallel execution is normally done at a fairly low level. This is another reason for utilizing the performance possibilities that nesting gives, to compete with MPI's good ability of scaling.

2.2 Hybrid methods: MPI + OpenMP

There are many papers reporting experiences in combining MPI and OpenMP to exploit multiple levels of parallelism. Examples of this are [26, 27, 37, 38]. This way of achieving nested parallelism has especially grown as the clustered SMP-machines have entered the market. MPI is used on the outer level, communicating between the SMP-nodes, while OpenMP is used on the inner level to do the parallelization within one node.

When new types of machines and architectures are entering the market, there is no doubt that using techniques adjusted to the situation will give the best results given the special environmental conditions. However, when new systems take over, techniques specially adjusted to the previous system, might not be the optimal techniques anymore. Therefore, it is important that, in addition to making methods adapted to the current machines, one improves methods not depending on any particular machine system. We think this is an important task, and consider that the work in [A] contributes to keep the "idealistic" view on parallelism alive.

2.3 Compilers using OpenMP directives, supporting nesting

While commercial compilers were lacking the possibilities to exploit multiple levels of parallelism in the first existing years of OpenMP, experimental compiler environments extended OpenMP with their own functionalities in order to make nested parallelism available on their compiler. Two prominent examples of this are mentioned in the following.

The NanosCompiler [2, 7] is a source-to-source parallelizing compiler for OpenMP applications written in Fortran 77. The compiler is able to exploit multiple levels of parallelism and generate work from multiple simultaneously executing threads. The researchers behind Nanos claim on their web page: *"One of the objectives of the project is to prove that multi-level parallelization can be instrumental in providing scalability in a programming and execution model; nested parallelism provides further opportunities for work distribution, both for parallel loops and code sections."* This statement is in complete agreement with our experiences with nested parallelism, and supports our motivation for continuing working within this field.

When working on [A], we collaborated with Prof. Eduard Ayguade at the European Center for Parallelism of Barcelona (CEPBA). He tested the nested directives on their NanosCompiler on the same applications as we tested our nesting techniques in [A]. His results were similar to our results [6].

Another compiler supporting nesting in OpenMP is the Omni Compiler [3], which is a collection of programs and libraries, written in C and Java, that translates C and Fortran 77 OpenMP programs into a C-program with runtime library calls. Omni/ST is an experimental nested parallelism support for Omni. It is implemented as an external library called "StackThreads/MP" and a runtime library that calls StackThreads/MP. With Omni/ST enabled, the user can compile programs both with and without this nested parallelism support.

2.4 The validity of the selling arguments

As mentioned in the beginning of Section 2, the selling arguments for OpenMP are *"Simple, Portable and Scalable SMP-Programming"* [4], and the validity of the arguments have been discussed in the previous subsections.

The explicit thread programming technique presented in [A] scales quite well. However, parallelizing a code with this nesting technique can not be considered as simple. If the nesting directives were available, it would have simplified the programmer's effort considerable. This is also what we would prefer to do. The technique in [A] is to be considered as the second best thing to do if nested parallelism is not available using directives, and a remedy to show what benefits one can achieve from exploiting nested parallelism when available in a problem.

If nesting directives were implemented in all commercial compilers, programming with OpenMP is in our opinion *simple*, as the need for explicit techniques will not be necessary. OpenMP parallelized codes will also be *portable* as compilers will behave similarly on different machines. Finally, being able to exploit nested parallelism, an OpenMP application will have larger possibilities to be *scalable*.

2.5 Further development

In [A], we as future work mentioned the numerical simulation of partial differential equations using adaptive mesh refinement as our ”ultimate target application for nested parallelism”. Before advancing to nested parallelism of adaptive mesh refinement applications, it was desirable to study the performance of solving partial differential equations numerically using adaptive mesh refinement; What are the costs of adaptivity and what are the savings of mesh refinement?

3 Adaptive mesh refinement

When solving partial differential equations (PDE) numerically in real life applications, the need for dense grid points may differ from region to region and sometimes also in time. This can happen in ocean modeling where the need for fine grid resolution may be larger close to the coast, a stream or a moving obstacle. Another example is weather forecast, where the need for close grid points increases close to special weather phenomena, mountains, etc. In mathematical terms these features will typically be areas around shocks or where the solution has steep gradients or complicated structure. Using the finest resolution needed in a uniform grid will in such cases naturally cause a computational cost that is larger than necessary. This is particularly the case if there is only a small part of the area that needs the fine grid resolution.

With the local mesh refinement method, it is possible to focus on areas where shocks or steep gradients occur, without using computationally expensive fine grid points in flow regions with only minor changes. A refined grid will, with this method, cover the ”interesting” area, and large gains in computer time and memory can be achieved.

If it is not known in advance where the ”interesting” features take place, or if these features are moving, adaptive mesh refinement (AMR) [14] will dynamically create and remove patches of refined mesh as the need for fine resolution changes. This algorithm was introduced by Berger and Oliger in 1984 [15]. Every Kth time step all grids on a particular level are regenerated in order to adapt to moving features. The regions that need a refined grid are identified by an error estimation procedure and/or a spatial gradient check. Special care has to be taken at the interface between coarse and fine grids to preserve conservation.

Even though most computational scientists agree that methods refining the mesh adaptively are highly needed in many compute intensive problems, the use of these methods is still limited. The main counter-arguments are that AMR is hard to implement, and does not necessarily pay off considering the cost of adaptivity and possible loss of accuracy. To test the validity of this last argument, we did a thorough case study of these costs.

3.1 Costs and savings of AMR

Given the arguments against AMR just mentioned, it is important to examine the benefits and drawbacks of AMR. This was our motivation for undertaking the work reported in paper [B].

Paper [B] is a study of costs and savings of an application built on AMRCLAW [13], a freely available software package based on Marsha Berger’s AMR algorithms [12, 15] and AMR software [11] and Randall LeVeque’s software Conservation LAW PACKage, CLAWPACK [31]. *Cost* was measured as loss of accuracy and extra computational work due to the adaptive mesh refinement. *Saving* was a measurement of ”speedup” compared to solving the same problems on a uniform fine mesh (should not be confused by ”parallel speedup”). The main findings from our experiments and performance modeling are:

- AMRCLAW succeeds in preserving the computational desirable features of CLAWPACK.
- Irregularities in the computational grid, introduced by AMR, do not lead to poorer memory management and corresponding lower flop-rate for our

tests.

- The computational costs for incorporating data from multiple grid levels are cheap, while costs associated with error estimation and regridding are much more expensive.
- Considering the computational work, applying AMR every time step and for a refinement factor of 2, pays off when less than 70-75% of the cells are being refined. Decreasing the frequency of refinement will increase this percentage.

Since AMR pays off when up to 70-75% of the area is refined, quite a lot can be saved in computational effort for problems having only minor areas which need a finer resolution. Knowing this, the time spent in implementing AMR into an application, can be looked upon as an investment in computational efficiency.

4 Nested parallelism applied to an AMR application

In [A], nested parallelism is shown to be imperative for good performance, when applied to problems with a fixed and relative small number of tasks. After having studied the possibilities of AMRCLAW in [B], a natural way further was to apply the nesting techniques developed in [A] to an AMRCLAW application. This application has a built-in multilevel nature and differs from the applications tested on in [A], since refined patches are created and dismissed in an unpredictable way as the computation proceeds, creating a *dynamically* changing number and sizes of tasks. The motivation was to investigate the efficiency of applying our techniques to a code with dynamically changing tasks, and check if the overhead lying in the distribution of threads to teams would dominate the benefits one could achieve from nested parallelism.

Paper [C] combines the knowledge achieved in [B] with the nested techniques developed in [A], by applying nested parallelism to an AMRCLAW application. As mentioned in Section 1, both nested parallelism and AMR are complicated means to improve the performance of a program. Combining the techniques therefore increases the complexities considerably. This was the reason why it was important for us to study the characteristics of the techniques separately and gain knowledge, before taking the step of mixing the two techniques.

4.1 Removing the dependency of coupled patches in AMRCLAW

When parallelizing AMRCLAW by 2-level nested parallelization, a natural choice was to execute the coarse-grained parallelization on patch-level, and the fine-grained on loop-level. The threads should be divided in teams, each team assigned to a specific grid patch, which on a given level constituted a task. Next, each task should be processed in parallel on loop-level, and the threads within each team should share the work associated with the respective task.

The work associated with each patch was mainly to be considered as independent from the work associated with the other patches on the same grid level. However, before starting the work of parallelization of AMRCLAW, one dependency had to be removed at the finest level of grids, to avoid synchronization between each fine grid time step. In [C], we showed how this dependency could be eliminated, at the expense of some minor extra computation.

4.2 Nested parallelism applied to AMRCLAW

In [C], a nested parallel AMR algorithm is presented that has the sequential AMR algorithm in [B] as a starting point.

As mentioned, the difference between AMRCLAW [13] and the applications parallelized in [A] is that AMRCLAW, in addition to having a multilevel structure, dynamically changes the number and sizes of tasks. The speedup obtained by parallelizing an AMRCLAW application in 2 levels was not as great as obtained in [A], due to the dynamic nature of the problem. But comparing with two 1-level parallel approaches, a fine-grained and a coarse-grained, nested parallelism was superior and seems to be the only choice for significant speedup on these kinds of problems.

The size of a team of threads should reflect the workload of the task. In order to distribute the work of a varying number of N tasks to P threads, an extension of the distribution algorithm proposed in [A] was necessary to cover the cases where $N > P$. This extended distribution algorithm is proposed in [C]. In this algorithm, the patches are divided in 2 sets, $\mathcal{L}$ (”large” tasks) and $\mathcal{S}$ (”small” tasks), where one or more threads are working on the same patch in $\mathcal{L}$, and where each thread has one or more tasks to work on in $\mathcal{S}$. Profiling showed that the time spent on the distribution of tasks to threads was negligible.

The work of parallelizing AMRCLAW in 2 levels of parallelization in OpenMP was time consuming, mainly because directives appropriate for the team concept were missing. For instance, a team-aware directive like `!$OMP TEAMPRI-`

VATE, corresponding to `!$OMP THREADPRIVATE`, would have saved us for extra changes in the code. This directive is not a part of OpenMP 2.0, even if nesting was implemented in the compiler.

There exist a handful of AMR software packages parallelized with MPI, like PARAMESH [5], DAGH [1], and a software package at Berkeley National Laboratory [34]. These are all using one level of parallelization. There also exists a paper [26] reporting on a hybrid (MPI + OpenMP) nested parallel version of an AMR algorithm, where instead of balancing the load on distributed processes, the processes ask for help by recruiting other processors.

5 Shallow water equations on a sphere

The shallow water equations (SWE), describing a thin layer of fluid flow (liquid or gas) in two dimensions, is the simplest possible model that captures the essential characteristics of fluid flow on the sphere.

There are many different numerical methods computing SWE on a sphere. Therefore, a standard test suite of seven problems for evaluating numerical methods for the SWE in spherical geometry [41] was proposed by Williamson et al. [41] in 1992, and accepted by the modeling community in order to compare newly proposed methods.

5.1 The different discretization techniques

There exists a relatively large group of numerical methods for solving the SWE on a sphere, each of them having advantages and disadvantages concerning accuracy, efficiency, reliability, conservation, flexibility, scalability, etc. The numerous methods can be classified by the different spatial discretization.

The most popular class of methods for solving flow problems on the sphere is the spectral transform method (STM) [39, 22], first introduced in 1970 [33, 21]. STM expresses the variables as series expansions in terms of the spherical harmonic basis functions, and is known for its high level of accuracy and for being extremely stable. The drawback of STM is the computational complexity and communication overhead with respect to parallelization. Besides, it is not suited for dealing with mesh refinement, as it is designed for a fixed global grid.

In order to find a method that competes with the STM in accuracy and stability, but which is more efficient, some scientists [28] have taken the double Fourier method into use. This method uses the fast Fourier transforms (FFT) in the latitude direction rather than the more expensive associated Legendre transforms, to reduce the computational complexity.

The finite element method (FEM) [9, 25, 19, 17, 18, 20] is another commonly used method, introduced into hydrodynamics in 1973 [24]. An advantage of the FEM is its good parallel scalability and its geometric flexibility. It is easy to construct a (near) uniform mesh, and to locally refine this if necessary. The disadvantage appears to be in conservation of mass and momentum.

Spectral element methods (SEM) [40] can be viewed as a special case of the FEM, where the elements use the high order basis functions from STM.

In climate modeling, conservation of mass and momentum are of great importance, and techniques which suffer from small conservation losses in a short term simulation may give unacceptable losses in long running simulation. By its very design criterion, the finite volume methods (FVM) guarantee conservation. This served as a motivation for us to look at these techniques in paper [D]. Implementing the problem was done using CLAWPACK [32] based on spherical coordinates and a regular latitude-longitude grid.

5.2 Computing SWE using finite volume methods

Since the spherical SWE contained a source term, CLAWPACK applied an operator splitting technique, also known as the fractional step method [30]. The PDE are in each time step solved by alternating between

i) the homogeneous differential equation, and
ii) an ordinary differential equation.

The homogeneous differential equations are solved by a second order corrected Godunov's method, a method based on cell averages of the state variables, and approximation of fluxes through the cell interfaces. By the introduction of cell averages, a Riemann problem appears on the cell interfaces, and the fluxes are decomposed into waves determined by the eigenvectors of the Jacobian matrix.

The standard wave-propagation method uses Roe averages [35] to approximate the state variables at the cell interfaces (see [30] for details). It was not possible to compute these averages for our model, since the Roe criteria were not possible to satisfy due to variable coefficients.

To test our method using FVM on the spherical SWE, we used Test case 2 in [41], *Global Steady State Nonlinear Zonal Geostrophic Flow*, which has a steady state solution to the non-linear SWE. The standard wave-propagation method gave a solution far from steady state, probably because our Roe averages did not satisfy the Roe criteria.

A better approach to our problem was to apply *f-waves* [8], a flux-based wave decomposition method for solving the Riemann problem. Applying f-waves with second order correction, the following results were achieved in [D]:

- The mass was 100% conserved.
- A stable steady state solution close to the correct steady state was obtained.
- A pole-problem was observed.

The L_1 and L_2 relative error-norms, defined as $L_p(h) = ||h_{exact} - h_{comp}||_p$ / $||h_{exact}||_p$, showed that the overall errors are acceptable. For the depth h and a 100×100 grid, the error-norms were $L_1(h) = 0.0005$ and $L_2(h) = 0.0013$ after five days. However, the max-norm of the error $L_\infty(h) = ||h_{exact} - h_{comp}||_\infty$ / $||h_{exact}||_\infty = 0.0392$, reflected nonphysical behavior at the poles.

5.3 Operator splitting and steady state

As already mentioned, an operator splitting technique was used to handle the source term (resulting from geometry transformation and the Coriolis force). Our solution did preserve a steady state close to the initial state, but nevertheless a slightly difference between this state and the initial state was observed.

As LeVeque points out in [29], fractional step methods can easily fail since solving i) may lead to large changes in the solution which should then be exactly undone by solving ii). It is unlikely that this will happen exactly, since very different sorts of numerical methods are used in the two steps. The reason why we find a steady state which is slightly different from the initial state is probably that we are not actually solving the original system of spherical SWE but a perturbed system, since the problem is split in two.

In [29], LeVeque has developed an approach, *The Quasi-steady wave-propagation algorithm*, which allows the source term to be incorporated into the wave-propagation algorithm, avoiding fractional steps. However, in [8] the authors claim that an f-waves method is a better approach, since this algorithm can "easily" be extended to incorporate the source term in the f-waves. This is understandable as the authors operate with a well defined spatial direction. Our source term, however, does not lend itself to this kind of decomposition as it has terms which mixes spatial dimensions.

5.4 The pole-problem

Explicit singularities at the pole-interfaces were avoided by using FVM, as the state variables were only represented at the cell centers. However, near singularities at the pole-cells caused computational inaccuracy. The small perturbation introduced by the splitting caused a raise of depth close to the equator, and this perturbation propagated toward the poles. Here a few physically small cells had to pay for the minor displacements of mass, producing relatively large errors in these cells.

5.4.1 Mesh refinement

Having the large performance improvement achieved by AMRCLAW in [B] in mind, and observing that the errors are located to a small area, it was in [D] tempting to try to reduce the errors near the poles by mesh refinement. However, applying mesh refinement to our problem did not reduce the errors at the poles. The reason is that the large gradients in h near the poles are due to numerical error, and not caused by physical phenomena. Consequently, resolving with a finer mesh did not decrease the maximum error.

Another problem also occurred with AMRCLAW, when applying it to a model with spherical coordinates. The way AMRCLAW is built, cells which are refined with a factor R, are propagated in time with R fine grid time steps to catch up with the coarse grid time step. However, for a latitude-longitude grid, the fine grid time steps have to be a factor R^2 less than the coarse-grained time step to fulfill the CFL-criterion at pole-cells.

6 Future work

For important classes of problems, having a built-in multilevel structure, [A] and [C] proved that nested parallelism offers better parallel scalability. Some vendors claim they have implemented nesting in their OpenMP compiler, and people having the possibilities to test early versions of these compilers have shown interest in nested parallelization of AMRCLAW. It would have been interesting to see how these compilers behaved on the AMRCLAW application, to compare the performance achieved by our explicit thread programming technique with the performance achieved by using only nesting directives. However, using directives only, one needs a team-aware directive `!$OMP TEAMPRIVATE` in the parallelization of AMRCLAW, The work reported in [D] confirmed that the FVM is a candidate for solving flow problems on the sphere. In particular, the technique should be seriously considered when conservation is important. However, choosing the right decomposition strategy turned out to be crucial. We are convinced that the pole problems we observed is a result of the spherical coordinates and the computational grid we are using, causing very small physically cells close to the poles. In the near future, we plan to transform the sphere into a ”spherical cube” [36], based on six equal-sized squares and rectangular cells. This kind of computational grid will not have any artificial boundaries, and we believe the pole problem observed will be reduced or removed.

In [B], we showed that using AMR can improve the performance for problems having a varying need for fine resolution. Due to these good experiences, we also plan to try AMR on the spherical SWE. We believe this could be a great advantage when turbulence occurs locally, as it will in many cases when topography is introduced. A prime example is Test case 5 of the test suite [41].

A research group at Munich University of Technology (TUM) is also working with the spherical SWE, and collaboration between our research group at the University of Bergen and this group has already been started. Both groups work in the field of SWE and AMR, but with different approaches. Our FVM approach and experiences with AMR will be compared to the TUM group's results of implementing the same problem on top of their FEM-based AMATOS software [10]. This will hopefully lead to a better understanding of what can be achieved with AMR techniques and the benefits of the different frameworks it is implemented in.

References

[1] DAGH: Data-Management for Parallel Adaptive Mesh-Refinement Techniques. http://www.caip.rutgers.edu/~parashar/DAGH/.

[2] NanosCompiler. http://research.ac.upc.es/nanos/.

[3] OMNI OpenMP compiler. http://phase.etl.go.jp/Omni/.

[4] OpenMP. http://www.openmp.org/.

[5] PARAMESH V2.0, Parallel Adaptive Mesh Refinement. http://webserv.gsfc.nasa.gov/rib/repositories/inhouse_gsfc/Users_manual/amr.html.

[6] E. Ayguade. *Private communication*. European Center for Parallelism of Barcelona (CEPBA), 2000.

[7] E. Ayguade, M. Gonzales, J. Labarta, X. Martorell, N. Navarro, and J. Oliver. NanosCompiler: A Research Platform for OpenMP Extensions. In *Proceedings of EWOMP'99, First European Workshop on OpenMP*, 1999.

[8] D. S. Bale, R. J. LeVeque, S. Mitran, and J. A. Rossmanith. A Wave Propagation Method for Conservation Laws and Balance Laws with Spatially-Varying Flux Functions. *SIAM Journal on Scientific Computing*, 24:955–978, 2003.

[9] J. Behrens. Atmospheric and ocean modeling with an adaptive finite element solver for the shallow-water equations. *Applied Numerical Mathematics*, 26:217–226, 1998.

[10] J. Behrens. AMATOS: Adaptive Mesh Generator for Atmosphere and Ocean Simulation, API Documentation Version 1.2. http://www-m8.mathematik.tu-muenchen.de/m3/software/amatos/amatos.pdf, March 2002.

[11] M. J. Berger. Adaptive Mesh Refinement for Hyperbolic Conservation Laws. http://cs.nyu.edu/cs/faculty/berger/amrsoftware.html, 1995.

[12] M. J. Berger and P. Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics*, 53(3):561–568, 1984.

[13] M. J. Berger and R. J. LeVeque. AMRCLAW, Adaptive Mesh Refinement + CLAWPACK. `http://www.amath.washington.edu/~rjl/amrclaw`, 1997.

[14] M. J. Berger and R. J. LeVeque. Adaptive Mesh Refinement using Wave-Propagation Algorithms for Hyperbolic Systems. *SIAM Journal on Numerical Analysis*, 35(6):2298–2316, 1998.

[15] M. J. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53:484–512, 1984.

[16] R. Blikberg and T. Sørevik. Early experiences with OpenMP on the Cray Origin 2000. In *Proceedings of Fourth European CRAY-SGI MPP Workshop*, 1998.

[17] S. Chippada, C. N. Dawson, M. L. Martinez-Canales, and M. F. Wheeler. Finite Element Approximations to the System of Shallow Water Equations I: Continuous-Time A Priori Error Estimates. *SIAM Journal on Numerical Analysis*, 35(2):692–711, 1998.

[18] S. Chippada, C. N. Dawson, M. L. Martinez-Canales, and M. F. Wheeler. Finite Element Approximations to the System of Shallow Water Equations, Part II: Discrete-Time A Priori Error Estimates. *SIAM Journal on Numerical Analysis*, 36(1):226–250, 1998.

[19] J. Coté and A. Stanisforth. An accurate and efficient finite-element global model for the shallow-water primitive equations. *Monthly Weather Review*, 118:2707–2717, 1990.

[20] C. N. Dawson and M. L. Martinez-Canales. Finite Element Approximations to the System of Shallow Water Equations, Part III: On the Treatment of Boundary Conditions. *SIAM Journal on Numerical Analysis*, 38(1):149–159, 2000.

[21] E. B. Eliassen, B. Machenhauer, and E. Rasmussen. On a numerical method for integration of the hydrodynamical equation with a spectral representation of the horizontal fields. Technical report, Department of Meteorology, University of Copenhagen, 1970.

[22] W. L. Gates. AMIP: The atmospheric model intercomparison project. *Bulletin of the American Meteorological Society*, 73(12), 1992.

[23] W. Gropp and R. Lusk. *Using MPI, portable parallel programming with the Message Passing Interface*. The MIT Press, 1994.

[24] G. Grotkop. Finite element analysis of long period water waves. *Computer Methods in Applied Mechanics and Engineering*, 2(148), 1973.

[25] T. Heinze. The shallow water equations on the sphere and their Lagrange-Galerkin-solution. *Meteorology and Atmospheric Physics*, 81(1):129–135, 2002.

[26] W. Huang and D. K. Tafti. A Parallel Computing Framework for Dynamic Power Balancing in Adaptive Mesh Refinement Applications. In *Proceedings of Parallel Computational Fluid Dynamics 1999*, 1999.

[27] T. H. Kaiser and S. B. Baden. Overlapping communication and computation with OpenMP and MPI. *Scientific Programming*, 9(2,3):73–82, 2001.

[28] A. T. Layton and W. F. Spotz. A Semi-Lagrangian Double Fourier Method for the Shallow Water Equations on the Sphere. Submitted to *Journal of Computational Physics*, 2002.

[29] R. J. LeVeque. Balancing Source Terms and Flux Gradients in High-Resolution Godunov Methods: The Quasi-Steady Wave-propagation Algorithm. *Journal of Compututational Physics*, 146(1):346–365, 1998.

[30] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[31] R. J. LeVeque and J. O. Langseth. CLAWPACK 3.0, a Software Package for Conservation Laws and Hyperbolic Systems. `http://www.amath.washington.edu/~rjl/clawpack`, 1997.

[32] R. J. LeVeque, J. O. Langseth, M. J. Berger, and S. Mitran. CLAWPACK 4.0, a Software Package for Conservation Laws and Hyperbolic Systems. `http://www.amath.washington.edu/claw`, 2000.

[33] S. A. Orzag. Transform method for calculation of vector coupled sums: Application to the spectral form of the vorticity equation. *Journal of the Atmospheric Sciences*, 27:890–895, 1970.

[34] C. A. Rendleman, V. E. Beckner, M. Lijewski, W. Crutchfield, and J. B. Bell. Parallelization of Structured, Hierarchical Adaptive Mesh Refinement Algorithms. *Computing and Visualization in Science*, 3:147–157, 2000.

[35] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[36] J. A. Rossmanith. *A Wave Propagation Method with Constrained Transport for Ideal and Shallow Water Magnetohydrodynamics*. PhD thesis, Applied Mathematics, University of Washington, 2002.

[37] L. Smith. Development and performance of a mixed OpenMP/MPI quantum Monte Carlo code. *Concurrency - Practice and Experience*, 12(12):1121–1129, 2000.

[38] L. Smith and M. Bull. Development of Mixed Mode MPI / OpenMP Applications. *Journal of Scientific Programming*, 9(2,3):83–98, 2001.

[39] P. N. Swarztrauber. Spectral transform methods for solving the shallow water equations on the sphere. *Monthly Weather Review*, 124:730–744, 2000.

[40] M. Taylor, J. Tribbia, and M. Iskandarani. The spectral element method for the shallow water equations on the sphere. *Journal of Computational Physics*, 130(1):92–108, 1997.

[41] D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber. A Standard Test Set for Numerical Approximations to the Shallow Water Equations in Spherical Geometry. *Journal of Computational Physics*, 102(1):211–224, 1991.

Errata

Paper A:

Ref [1]: The classification ”Technical report” shall be removed.

Ref [2]: The classification ”Technical report” shall be removed.

Ref [3]: The classification ”Technical report” shall be removed.

Ref [4]: The text ”openmp” shall be replaced by ”OpenMP”.

Ref [9]: The correct text shall be ”...compression of two-dimensional...”.

Paper B:

Page 19, last section in chapter 5: ”AMRCLAW” shall be replaced by ”an AMR application”.

Ref [RBL+99]: ”Technical report, April 1999” shall be replaced by ”*Computing and Visualization in Science*, 3:147-157, 2000”.

Paper C:

All figures (colored in greyscale) in the paper presented at EWOMP 2002 have in this thesis been replaced by color figures.

Ref [RBL+99]: ”Technical report, April 1999” shall be replaced by ”*Computing and Visualization in Science*, 3:147-157, 2000”.