*Proc. of the 2002 International Conference on Computational Science (ICCS02), pp 419-*419

A Case Study for Automatic Code Generation on a Coupled Ocean–Atmosphere Model

P. van der Mark[1,⋆], R. van Engelen[2,⋆⋆], K. Gallivan[2,⋆⋆], and W. Dewar[3]

[1] Leiden Institute of Advanced Computer Science,Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands
pmark@liacs.nl

[2] Department of Computer Science, Florida State University, Tallahassee, FL 32306-4530, USA
{engelen,gallivan}@cs.fsu.edu

[3] Department of Oceanography, Florida State University, Tallahassee, FL 32306-4320, USA
dewar@ocean.fsu.edu

Abstract. Traditional design and implementation of large atmospheric models is a difficult, tedious and erroneous task. With the Ctadel project we propose a new method of code generation, where the designer describes the model in an abstract high-level specification language which is translated into highly optimized Fortran code. In this paper we show the abilities of this method on a coupled ocean–atmosphere model, in which we have to deal with multi-resolution domains and different time-steps. We, briefly, describe a new concept in compiler design, the use of templates for code generation, to elevate the burden of choosing architecture optimized numerical routines.

1 Introduction

Simulation of atmospheric models is a computationally intensive task, where developers of those models have to make a trade-off between numerical accuracy and computational resources. First, approximations are made in the formulation of the physical processes in a mathematical model by discarding certain details that are assumed to have little impact on the overall outcome. Second, approximations are made by the discretization of the model and its numerical solving methods and the resolution of the discrete grid.

Increasing processing power and the development of new high-performance architectures such as large networks of workstations (NOW) (for example the Beowulf project [9]) have led to opportunities for developers of numerical models to change the focus on more numerical detail, finer resolution or larger computational domains. This requires a significant programming effort to implement the changes in the simulation application since a simple plug-and-play development

⋆ Supported in part by the FSU Cornerstone Program for Centers of Excellence.
⋆⋆ Supported in part by NSF grant CCR-0105422.

paradigm with software components for scientific applications does not yet exist. This can lead to huge amounts of undocumented code, for which new versions have to be developed with every new computer architecture emerging.

In most cases, however, a model can be described on an abstract level, for example mathematical equations. One of the aims of the CTADEL project is to enable developers to formulate the problem on a high level, refraining from architecture specific dependencies, while producing highly efficient codes for a variety of target platforms [13]. In this paper we show the possibilities of CTADEL on a coupled ocean–atmosphere model, a quasi-geostrophic climate dynamics model [2]. Coupled models impose certain difficulties on their implementation; often these models use multiple resolutions on the computational grids and separate time steps. This also assesses certain constraints on the interaction between the different parts of the model and the parallelization of the model.

For specifications in CTADEL the ATmospheric MOdeling Language (ATMOL) [14] was developed in close collaboration with meteorologists at the Royal Netherlands Meteorological Institute (KNMI) to ensure ease of use, concise notation, and the adaptation of common notational conventions. The high-level constructs in ATMOL are *declarative* and *side-effect free* which is required for the application of transformations to translate and optimize the intermediate stages of the model and its code. ATMOL is *strict* and requires the typing of objects before they are used. This helps to pinpoint problems with the specification at an early stage before code synthesis takes place. ATMOL supports both high-level as well as low-level language constructs such as Fortran-like programming statements which are used to implement and optimize the target numerical code.

This paper is organized as follows, in section 2 we will explain some of the basic theory behind the original quasi-geostrophic model, while we discuss our specification of the model in section 4 and the CTADEL system in section 3. Results from experiments conducted with the produced codes by CTADEL and the original hand-written code are shown in section 5 and we draw some conclusions in section 7.

2 Quasi-Geostrophic Dynamics

In this section we will explain the quasi-geostrophic dynamics model some more. For the sake of understandability and readability we only briefly touch the theory and interested readers are referred to [2]. The model describes a mid-latitude coupled climate model, which is used in an attempt to understand how the ocean climatology is modified by atmospheric coupling. At present, the best comprehensive coupled climate models run at resolutions far coarser than those needed to model the inertial recirculation. The model presented here is an attempt to explicitly include eddies in general, and inertial recirculation in particular, within the framework of an idealized climate setting. The basic model consists of a quasi-geostrophic channel atmosphere coupled to a simple, rectangular quasi-geostrophic ocean. Heat and momentum exchanges between the ocean and the atmosphere are mediated via mixed layer models and the system is driven by steady, latitudinally dependent incident solar radiation.

Solar radiation is the basic force behind the global climate. About 30% is reflected back to space while the remaining part is absorbed mostly at bottom interface, whether it be land or water. Heat transfers to the atmosphere occur either through sensible or latent fluxes, or long wave radiative surface emissions to which the atmosphere is almost totally opaque.

The model is based on a classical β-plane mid-latitude representation and a two layered version of the quasi-geostrophic (QG) equations. For the most part, classical QG models are adiabatic, i.e. thermodynamics are neglected [4]. If the standard QG scaling is used and the usual Rossby number expansion is employed, the momentum equations yield the non-dimensional vorticity equation

$$(\frac{\partial}{\partial t} + J(p_i, .))(\nabla^2 p_i + \beta y) = w_{i1z} + HF, \tag{1}$$

where J denotes the usual Jacobian operator, p_i is the layer pressure, y is the meridional coordinate, HF denotes the horizontal frictional effects and all variables are non-dimensional.

The final dimensional forms of the QG equations for both ocean layers can now be extracted, for example the first layers reads,

$$\begin{aligned} \tfrac{d}{dt} q_1 &= A_{1h} \nabla^6 p_1 + \tfrac{f_0}{H_1}(w_{ek} - E(-H_1 - h_{1+})) \\ q_1 &= \tfrac{\nabla^2 p_1}{f_0} + \beta y - \tfrac{f_0}{g\prime H_1}(p1 - p2), \end{aligned} \tag{2}$$

where F_1 denotes the effects of the upper and lower layer horizontal frictional processes. Here $\nabla^6 p_1$ is used as closure for for the latter and super slip boundary conditions (i.e. $\nabla^2 p_i = \nabla^4 p_i = 0$).

3 Ctadel

Many attempts have been made to solve scientific or physical models numerically using computers. Today, a large collection of libraries, tools, and Problem Solving Environments (PSEs) have been developed for these problems. The computational kernels of most PSEs consist of a large library containing routines for several numerical solution methods. These routines form the templates for the resulting code. The power of this library determines the numerical knowledge of such a system.

A different approach to those so-called library-based PSEs consists of a collection of tools that generate code based on a problem specification without the use of a library. The numerical knowledge is determined by the expressiveness of the problem specification language and the underlying translation techniques. Ctadel belongs to this class of PSEs. Besides, a hardware description of the target platform is included in Ctadel's problem specification. This makes it possible to produce efficient codes for different types of architectures.

The Ctadel system provides an automated means of generating specific high performance scientific codes. These codes are optimized for a number of different architectures, like serial, vector, or shared virtual memory and distributed memory parallel computer architectures. One of the key elements of this system is the usage of algebraic transformation techniques and powerful methods for global common subexpression elimination. These techniques ensure the generation of efficient high performance codes.

For a more detailed description of CTADEL see [11].

4 Specification and Implementation

In this section we describe the specification of the quasi-geostrophic model using the ATMOL specification language for CTADEL and one of the main problems with this specification, the separate domain resolutions.

4.1 Spatial resolutions

As seen in section 2 both the ocean and atmosphere model have their own spatial domain resolution, which leads to several problems when combining both models into a mixed model. For example, when exchanging data between the two models, a spatial interpolation has to take place. Another problem arises with differentiation of a function, CTADEL can't assume a default grid size but has to determine this from the context. For example, one would normally write a differentiation like,

```
coordinates := [dx,dy].
Q = diff(P,y).
```

which would be translated into

```
   DO 10 j = 1,m-1
      DO 20 i = 1,n
 Q(i,j) = (P(i,j-1) - P(i,j)) / dy
20    CONTINUE
10 CONTINUE
```

Since we work with two different grid sizes, this is not conceivable. One possibility is to hard-code the distance between two grid-points into the specification, which is not desirable. Therefor we couple the specification of a domain to it's spatial grid sizes. For example, the definition of a domain could look like,

```
domain_a := i=1..X_a by j=1..Y_a by k=1..Z_a.
domain_a‘coordinates := [dxa,dya].
```

If we define a variable with this domain CTADEL knows which distances to choose from. A problem arising with this implementation is when two variables with different domains are combined in one equation. Since this has been avoided in the original model, we did not investigated this problem.

4.2 Implementation

For the sake of separation of concern we divided the original model in three clearly distinguishable sub-parts,

1. Initialization, in this initial data for the model is calculated for the ocean and atmosphere models. We discuss this step some more in subsection 4.3.
2. Ocean step, this sub-part performs one time step for the ocean model, which we will explain in subsection 4.4.
3. Atmosphere step, analogue with the ocean part, this sub-part performs one time for the atmosphere model. This part highly resembles the ocean step and we therefor don't discuss this part.

A possible extension to the specification could be a high-grain parallelization of steps two and three. In figure 4.2 we see the current sequential execution of the model, while figure 4.2 a possible parallel implementation is shown. Unfortunately data dependencies did not allow for a trivial parallelization; for example it has to be taken into account when data has to be exchanged between the models. This would require a redesign of the original model, which is beyond the focus of this work.

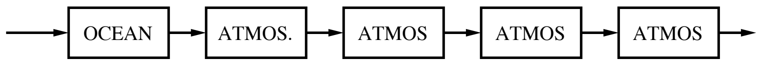

Fig. 1. Sequential execution of the ocean and atmosphere steps

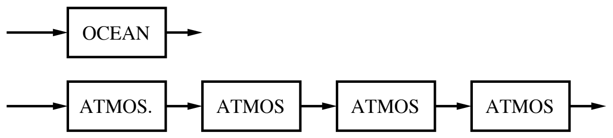

Fig. 2. Parallel execution of the ocean and atmosphere steps

In the implementation of the original model, several numerical library calls where made. For example, for finding the solution to a tridiagonal matrix a tridiag solver was called. In [10] we examined the use of calling external library calls for implicit equations, while for this specification we explore the use of templates even more [14] The use of templates makes it possible to offer the designer of the model a number of standard numerical solution methods, but leaves the actual implementation to CTADEL. The user can therefor specify the usage of a method while CTADEL can pick an appropriate and optimal implementation for the target architecture. For example when a Fourier analysis is needed the user can specify this like

```
V = Fourier(Q),
```

without bothering about the actual implementation; Ctadel decides if a library call should be made to an existing optimized library or fill in a the code itself. In the current implementation Ctadel makes use of default numerical functions, like `spline2` from [7].

4.3 Initialization

In the first phase of the model, a number of general input parameters and initial data are read in, after which model specific data is calculated from these parameters. The original implementation of the model also could perform special tasks in this initialization phase, like some sort of crash recovery from an aborted previous run. We did not implement this in our specification.

An example equation belonging to this first step is equation (2) which reads in the Ctadel specification like,

```
qcomp(P,DX,DY,H,GP,NL) := (C + sign *(f_zero/(GP * H))*(DPO)
          where C=(DX*(lapl P)/f_zero + beta * Y)
          where sign=(1 if k>1 \\ -1 otherwise)
          where (DPO=(P-(P@(k=k+1))) if k<NL \\
               ((P@(k=k-1))-P) otherwise)
          where Y=(j-1)*DY
      ).
CalcO(P,DX,DY, H,GP) :=
       ( zq(P,DY, H,GP,nlo) if borderj_O \\
         mqo(P,DY, H,GP) if borderi_O \\
         qcomp(P, DX, DY, H, GP,nlo) otherwise
       ).
```

This specification shows two generalized functions (including some boundary conditions) which can be used like,

```
qo = CalcO(po,dxom2, dyo, H_o, g_prime).
```

In the quasi-geostrophic model for the calculation of the barotropic mode, a `chsolv` routine function is used, which uses a Fourier and a tridiag solver. In our specification we make use of templates to deal with these solvers; in the current specification Ctadel calls an external library function [7] to find a solution. A sample specification for this reads like,

```
tp_ch2 = (ch_solv(boundary_south, aa, ba) where aa=1/dya^2).
tp_ch3 = (ch_solv(boundary_north, aa, ba) where aa=1/dya^2).
```

4.4 Ocean step

In this part of the specification, one ocean time step is calculated. A typical 'dynamics' equation which can be found is the calculation of the auxiliary currents, e.g. wind tendencies. The specification for equation (refeqn:aux) for the ocean dynamics read like,

```
u_ag = (-C*(P1-P2)*dxam2/f_zero
        where P1=((pa@(j=j+1)) if j<nya \\
                  (pa@(j=nya)) otherwise)@(k=1)
        where P2=((pa@(j=j-1)) if j>1 \\
                  (pa@(j=1)) otherwise)@(k=1)
        where C=(0.5 if noborder(nya) \\ 0 otherwise)
       ).
v_ag = (-1/2*((P@(i=i+1))-(P@(i=i-1)))*dxam2/f_zero
          where P=pa@(k=1)) if noborder(nya) \\
                0 otherwise.
```

5 Preliminary results

In this section the code generated by CTADEL for the quasi-geostrophic model is compared with the original hand-written code. Since the original code was not written with efficiency in mind, we can expect a performance difference with the generated code by CTADEL which performs some aggressive optimizations like common subexpression elimination. Standard compiler optimizations like loop optimization [5] where not applied. We ran the code on a scalar type of architecture; a commodity personal computer, running the linux/GNU operating system, with linux kernel version 2.4.10. The used machine was equipped with a 700 Mhz AMD Athlon processor and 128 MByte.

All test-runs used an input grid with variable horizontal and vertical points and two layers per model. For some experiments we used a fixed number of thousand time-steps while we also conducted experiments with diverging time-steps. In order to reduce external influences on these times, we run each experiment a number of times and calculated the average execution time. Because the deviations from these average times are in the order of tenths of percents, we do not include them in the figures.

For the compilation of the programs we used the GNU Fortran compiler, version 2.95.2, with standard optimization turned on (`g77 -O2`). With each run of the programs we compared the output of the generated code with the reference code. Since numerical solution methods are used over several time-steps, small differences appear in the output of both programs. However, these differences turned out to be relatively small and therefore acceptable.

In figure 3 we run both generated code and the hand-written code with a varying input grid size and a constant number of 1000 time-steps. Since both the atmosphere and the ocean model use different input grid sizes we have scaled them equitable. As we see from both the graphs, the code generated by CTADEL

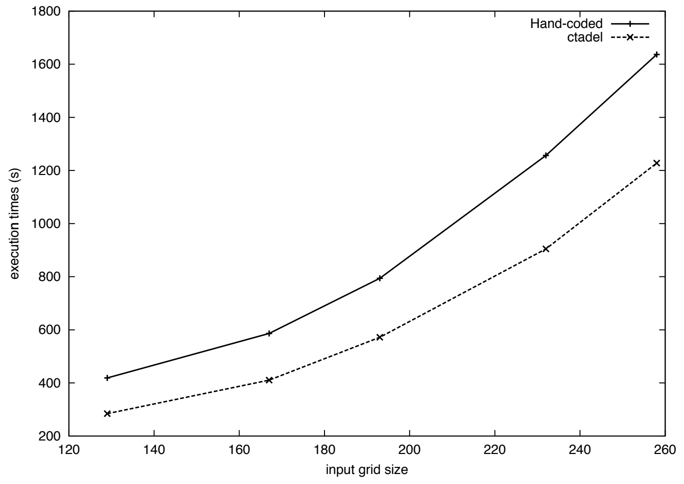

Fig. 3. Execution of both models with a varying input grid size, execution times in seconds

has a performance gain of 30% over the hand-written code. We also see that neither of the codes has a linear execution time with increasing input sizes.

In figure 4 we have examined the execution times of both the codes as a function of the number of time-steps. As one might expect, this is a linear relation. As for figure 3 the automaticly generated code from CTADEL has a performance gain of around 30% over the hand-written code.

These experiments show that automatic generated code can vanquish the hand-written codes for the quasi-geostrophic model. We should, however, point out that the original code was not optimized with respect to performance. Previous studies [10, 12] have shown that generated code can compete with hand-optimized codes.

6 Related work

Not much work has been done in the field of automatic generation of program codes based on a mathematical specification. Therefore programmers tend to ‘convert’ the problem by hand into a program or use libraries and/or problem solving environments (PSE). Some PSEs exist which produce program codes. For example the MathWorks package is a compiler for Mathlab specifications which generates `C` and `C++` codes. There also exists special purpose libraries for PSEs, which can handle a specific problem. Examples are the REDTEN [3] for Maple and RICCI [6] for Mathematica.

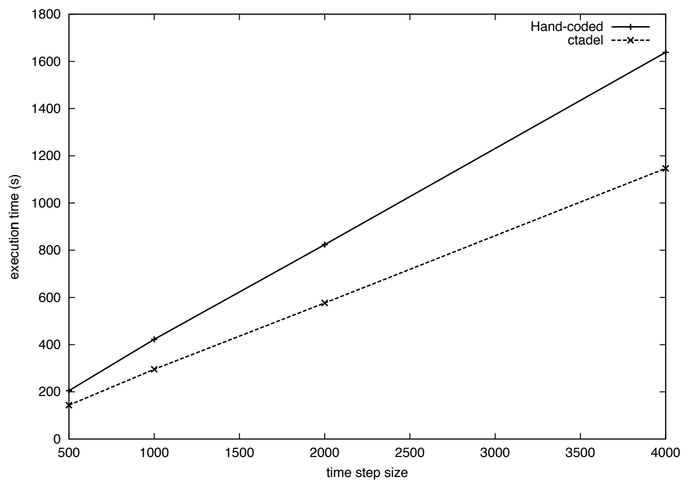

Fig. 4. Execution of both models with varying number of time steps, execution times in seconds

The Falcon project [1] takes a different approach, comparable with Ctadel. It uses an existing high-level array language, MATLAB, as source language and performs static, dynamic, and interactive analysis to generate Fortran 90 programs with directives for parallelism. It includes capabilities for interactive and automatic transformations.

7 Conclusion

In this paper we show a new method of code generation for large scale numerical models. Instead of translating a mathematical model into source code by hand and maintaining hand-optimized versions for several target computer architectures, the designer can specify the model in an abstract specification language which is automaticly compiled into highly-optimized target-dependent Fortran codes. We explored this on a coupled ocean–atmosphere model, in which we have to deal with multi-resolution domains and different time-steps. We also, briefly, addressed the concept of templates in our Ctadel compiler, which can elevate the burden from the model-designer of picking efficient numerical solution methods.

As we can see from the preliminary results in section 5 the generated code by Ctadel is clearly more efficient considering execution time. Since the original hand-written code was not optimized for speed, this is not a big surprise. More important is the ability to generate code for multiple architectures by using

CTADEL. For example, it's fairly trivial to automaticly generate codes for a distributed memory architecture with use of the MPI message parsing while this is a tedious task to do with the original hand-written code. Another advantage from specifying the model arises when more than the current two layers are needed.

References

1. L. DeRose, K. Gallivan, E. Gallopoulos, B. Marsolf, and D. Padua. Falcon: A matlab interactive restructuring compiler. In *8th International Workshop, LCPC'95*, Languages and Compilers for Parallel Computing, pages 269–288, Columbus OH, USA, Aug. 1995. Springer Verlag.
2. William K. Dewar. Quasigeostrophic climate dynamics. *Journal of Marine Research*, Submitted.
3. John F. Harper and Charles C. Dyer. http://www.scar.utoronto.ca/ harper/redten/.
4. W. Holland. The role of mesoscale eddies in the general circulation of the ocean: numerical experiments using a wind-driven quasi-geostrophic model. *Journal of Physical Oceanography*, 8:363–392, 1978.
5. E. Houstis, T. Papatheodorou, and C. Polychronopoulos. Advanced loop optimizations for parallel computers. In *Proceedings of the First International Conference on Supercomputing*, pages 255–277, New York, USA, 1987. Springer-Verlag.
6. University of Washington. Department. of Mathematics. http://www.math.washington.edu/~lee/Ricci/.
7. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in Fortran 77.* Cambridge University Press, 1992.
8. K. Shafer Smith and Geoffrey K. Vallis. The scales and equilibration of midocean eddies: Freely evolving flow. *Journal of Physical Oceanography*, 31:554–571, February 2001.
9. T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*, pages I:11–14, Oconomowoc, WI, 1995.
10. Paul van der Mark, Gerard Cats, and Lex Wolters. Automatic code generation for a turbulence scheme. In *Proceedings of the 15th International Conference of Supercomputing*, pages 252–259, Sorrento, Italy, June 2001. ACM.
11. R.A. van Engelen. *Ctadel: A Generator of Efficient Numerical Codes.* PhD thesis, Universiteit Leiden, 1998.
12. Robert van Engelen, Lex Wolters, and Gerard Cats. Ctadel: A generator of multiplatform high performance codes for pde-based scientific applications. In *Proceedings of the 10th International Conference on Supercomputing*, pages 86–93, Philadelphia, USA, May 1996. ACM.
13. Robert van Engelen, Lex Wolters, and Gerard Cats. Tomorrow's weather forecast: Productive program generation in atmospheric modeling. *IEEE Computational Science and Engineering*, 4(3):22–31, 1997.
14. Robert A. van Engelen. Atmol: A domain-specific language for atmospheric modeling. *Journal of Computing and Information Technology*, 2001.