Berkvens, P.J.F. and Botchev, M.A. (2002) Parallel Processing and Non-Uniform Grids in Global Air Quality Modeling. In: Proceedings of the Second Conference on Air Pollution Modelling and Simulation, APMS'01, April 9-12, 2001, Champs-sur-Marne, France. pp. 215-224. Springer Verlag. ISBN 3-540-42515-2

Parallel Processing and Non-Uniform Grids in Global Air Quality Modelling[1]

P.J.F. Berkvens[2] and M.A. Botchev[3]

Abstract

A large-scale global air quality model, running efficiently on a single vector processor, is enhanced to make more realistic and more long-term simulations feasible. Two strategies are combined: non-uniform grids and parallel processing. The communication through the hierarchy of non-uniform grids interferes with the inter-processor communication. We discuss load balance in the decomposition of the domain, I/O, and inter-processor communication. A model shows that the communication overhead for both techniques is very low, whence non-uniform grids allow for large speed-ups and high speed-up can be expected from parallelization. The implementation is in progress, and results of experiments will be reported elsewhere.

1 Introduction

We consider computational aspects of global air quality models (AQMs). We describe two methods for enhancing the efficiency of AQMs, namely two-way nesting (TWN) and domain decomposition with parallelization (DDP). We model the computer time and the wall clock time used by the computations and the communications. We estimate the speed-ups that can be achieved by using TWN and DDP.

Numerical methods for AQMs are typically based on a grid-like discretization of the atmosphere, but Lagrange-type methods exist. The species concentrations evolution on the grid is computed by a time-stepping method. In off-line models, which we consider here, the data for the meteorological and other mechanisms that influence the concentrations are obtained during execution of the model program from stored data sets or from a concurrently running dynamical meteorological model. Results like time series of concentrations, averages, and budgets (contribution to evolution per mechanism) are stored while the program is running.

The costs of simulations with a comprehensive AQM on a fine grid are high compared to the hardware available today. The number of grid cells is then at least $\mathcal{O}(10^7)$ based on cells of $1^\circ \times 1^\circ$ (longitude $\times$ latitude) and 60 layers. The number of timesteps is $\mathcal{O}(10^5)$ based on a 10-year run with 1-hour timesteps. The number of species is $\mathcal{O}(10^2)$. The I/O is $\mathcal{O}(10)$ byte per real time second.

These high costs call for very efficient computations. Apart from optimization of the numerical methods and the code, there are two additional techniques to enhance

[1]This work has been done within the NWO project no. 613-302-040

[2]CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, e-mail: berkvens@cwi.nl

[3]FOM-Institute for Plasma Physics, P.O. Box 1207, 3430 BE Nieuwegein, The Netherlands, e-mail: botchev@rijnh.nl

the efficiency of a method: two-way nesting (TWN) is meant to assign the available *computer time* more efficiently, domain decomposition with parallelization (DDP) is meant to assign the available *wall clock time* more efficiently.

A TWN technique has been implemented and tested [2], and combined with the upgrade TM5 of a particular AQM called TM3 [5, 9]. It has been applied to a case study [8]. The authors are currently implementing and testing DDP with TM3 and TM5. We present a model for the computer and wall-clock times for the calculations and communications when running an AQM code equipped with TWN and DDP. Performance measurements are not available yet.

The contents are as follows. Sect. 2 describes computational aspects of the numerical methods in an AQM, biased towards TM5. Sect. 3 discusses computational aspects of our TWN method. Sect. 4 does the same for DDP. In Sect. 5 the computer and wall clock time speed-ups of a method are defined, and their enhancement by TWN and DDP is demonstrated theoretically. In Sect. 6 we give our conclusions.

2 Numerical Methods for AQM

Three-dimensional AQMs covering the whole atmosphere typically contain the following mechanisms (with unit amounts of work):

- advection (u_{adv})
- turbulent mixing, cumulus convection ($u_{ver1}, u_{ver2}, u_{ver3}$)
- dry deposition, wet deposition (u_{dep})
- volume and surface emission (u_{emi})
- kinetic chemistry, photochemistry (u_{chm})

For the unit amounts of work we refer to Table 1. We restrict ourselves to gas phase chemistry, but the computational aspects of aqeous and solid phase chemistry can be modelled in a similar vein.

Parameters quantifying these mechanisms have to be provided. They include (proportionality with numbers of grid cells) [number of inputs per month real time]:

- wind velocity ($2N_V$), surface pressure (N_A), large scale precipitation rates (N_A), humidity (N_V), temperature (N_V) [n_{met}];
- turbulent diffusivity ($N_A N_C$), up/downdraft en/detrainment rates ($4N_A N_C$), bottom/top of convection layer ($2N_A$), convective precipitation rates (N_V), cloud cover, cloud bottom/top ($3N_V$) [n_{ver}];
- dry deposition rates (N_A), emission rates (N_A), land use (N_A) [n_{emi}].

Results that are stored may include full running model status (N_V) (n_{stp} times per month) for restarts, zonal ($N_y N_z$) and time (N_V) averages of concentrations,

and budgets (N_V), all n_{out} times per month real time. The above properties of AQMs are partly taken from [7] and the current TM3 and TM5 code [9].

Operator splitting makes the computations tractable by calculating the changes due to the various mechanisms consecutively rather than simultaneously. Reduction of the degree of operator splitting is studied [1, 4] and gives better accuracy at the same computational costs. Explicit time-stepping is used for advection, deposition, and emission, implicit time-stepping for the remaining mechanisms.

We consider the case of a constant number of cells per grid column and of a constant number of split steps (for the mechanisms) per timestep. Then the computational work load per grid column and per time-step is nearly constant. The main differences are caused by the photochemistry which is active only in the sunlit part of the atmosphere, and by differences in the vertical extent of the cumulus convection. We treat this amount $\hat{W}_{mec}$ of work related to the time-stepping of the mechanisms per column as a constant. In the nesting algorithm, data needs to be copied from parent to child and vice versa. The unit communication volume for a column to be updated by copying tracer data from or to a parent region is $\hat{V}_{cop}$. The communication volume $\hat{V}_{I/O}$ (i.e. amount of data in bytes) of the I/O per *column of the basic region* is not constant per timestep but rather per *real time* second. Because we need their values later on, we will give rough estimates:

$$\begin{aligned}
\hat{W}_{mec} &\simeq 2(3N_V S_t u_{adv} + N_A S_t(u_{ver1}N_C + u_{ver2}N_C^2 + u_{ver3}N_C^3) + \\
&\quad N_A(S_d + N_C S_w)u_{dep} + N_A(S_a + N_C S_v)u_{emi} + \\
&\quad N_V K S_c u_{chm} + 2N_V S u_{bud})/N_A \simeq 2.2 \cdot 10^7 \text{ flop}, \qquad (1)\\
\hat{V}_{cop} &\simeq 2N_z S u_{byt} \simeq 96000 \text{ flop}, \qquad (2)\\
\hat{V}_{I/O} &\simeq ((4N_V + 2N_A)n_{met} + (4N_V + 5N_A N_C + 2N_A)n_{ver} + \\
&\quad (N_A(S_d + U + S_a) + N_V S_v)n_{emi} + N_V S n_{stp} + \\
&\quad (N_V + N_y N_z)(1 + M)S n_{out})u_{byt}/(N_A n_{sec}) \simeq 7.2 \text{ byte}. \qquad (3)
\end{aligned}$$

We have used the data from Table 1, where the symbols are explained and rough estimates for their values are given. The (insignificant) costs for coarsening input data from fine regions to coarse regions (to have the data on overlapping regions correspond) are not modelled explicitly. The cell numbers are for a regular grid without nesting. The machine characteristics are estimated from specifications provided by SARA, owner of the TERAS platform TM5 is implemented on. To simplify the calculations model below, R_{cal}, Q_{com}, and $Q_{I/O}$ are assigned the same values. This does not affect the conclusions. For later use we define:

$$\varepsilon_c \equiv \hat{V}_{cop}/\hat{W}_{mec} \ , \ \varepsilon_i \equiv \hat{V}_{I/O}/\hat{W}_{mec} \ . \qquad (4)$$

3 Two-Way Nesting

Starting from a regular three-dimensional grid with a basic resolution, TWN uses predefined fixed block-shaped nesting regions within the basic grid. They may be

Table 1: Rough estimates for model quantities

quantity	value	meaning
N_x	360	cells in west-east direction
N_y	180	cells in south-north direction
N_z	60	cells in vertical direction
N_C	30	cells in troposphere part of column
N_A	64800	cells in the horizontal directions
N_V	3888000	total number of cells
S_t	50	transported species
S_d	20	dry-deposited species
S_w	20	wet-deposited species
S_v	10	volumewise emitted species (e.g. aircraft)
S_a	10	areawise emitted species (e.g. car traffic)
S_c	100	chemically active species
S	100	total number of species
K	15	average number of reactions+coreactants per species
M	8	number of mechanisms
U	5	land use characteristics
u_{adv}	25 flop	work for advection step
u_{ver1}	15 flop	work for vertical mixing proportional to N_C
u_{ver2}	5 flop	work for vertical mixing proportional to N_C^2
u_{ver3}	1 flop	work for vertical mixing proportional to N_C^3
u_{dep}	5 flop	work for dry or dry deposition
u_{emi}	5 flop	work for emission
u_{chm}	100 flop	work for (photo)chemistry
u_{bud}	1 flop	work for budgets and averages
u_{byt}	8 byte	number of bytes per real
n_{met}	120/month	large-scale meteorological inputs per month
n_{ver}	240/month	subgrid scale meteorological inputs per month
n_{emi}	1/month	surface and emission inputs per month
n_{out}	1/month	outputs per month
n_{stp}	360/month	time-steps in basic region per month
n_{sec}	2592000 s	seconds per 30-day month
R_{cal}	250Mflop/s	average calculation speed
Q_{com}	250Mbyte/s	communication speed among processors
$Q_{I/O}$	250Mbyte/s	I/O speed
τ_{lat}	15μs	latency time in interprocessor communication

recursively nested. Each nesting child region is spatially and/or temporally refined with respect to its parent region, which contains it. Because local refinement in the vertical direction is less important, we only consider horizontal spatial refinement. A parent and a child are coupled by advection through their interface. The concentrations in the child's interior evolve autonomously. On the interfaces the advection flux is determined from data on *both* sides, so there is a two-way coupling.

We give an example for a time refinement factor of 2 to illustrate how the nesting algorithm works, but it works for any refinement factor. Let $\hat{X}$, $\hat{Y}$, $\hat{Z}$, $\hat{R}$ denote x, y, z-advection steps and a step of the remaining mechanisms in the parent region, and similarly $\hat{x}$, $\hat{y}$, $\hat{z}$, $\hat{r}$ in the child region. Then a symmetric algorithm reads:

$$\hat{R}\hat{X}\hat{Y}\hat{Z} - \hat{r}\hat{x}\hat{y}\hat{z}\hat{z}\hat{y}\hat{x}\hat{r} - \hat{Z}\hat{Y}\hat{X}\hat{R} - \hat{z}\hat{y}\hat{x}\hat{r}\hat{r}\hat{x}\hat{y}\hat{z} \ , \tag{5}$$

corresponding to one time-step in the parent region and two in the child region. Before each $\hat{X}$ and $\hat{Y}$ step data is copied from the parent to the interface of its child. After a time-step in the child has been finished, its data is copied back to its parent. Our nesting algorithm conserves mass, positivity, and homogeneity (in terms of the mixing ratio) of the solution if the underlying advection scheme does so [3, 2], while allowing symmetric operator splitting in each region (except for the interface cells). When $\hat{R}$ and $\hat{r}$ consist of substeps for various mechanisms, their order should be reversed in their second occurrence in a timestep to maintain symmetric splitting.

We now model the computational work W_n for the TWN method. Let $n \in \{0, \ldots, N\}$ be the number of nesting regions (0 indicating the basic region). Let $L_{x,n}$, $L_{y,n}$, L_z, L_t be the spatial dimensions and the time interval, per nesting region, and define $L_n \equiv L_{x,n}L_{y,n}L_t$. ($L_z$ and L_t are the same for each region.) Let $r_{x,n}$, $r_{y,n}$, r_z, $r_{t,n}$ be the spatial and temporal resolutions per nest, and define $r_n \equiv r_{x,n}r_{y,n}r_{t,n}$. ($r_z$ is the same in each region.) We model W_n as:

$$W_n = \sum_{n=0}^{N} \hat{W}_{mec} L_n r_n + \sum_{n=1}^{N} \left(\hat{V}_{cop} \frac{R_{cal}}{Q_{I/O}} \right) L_n r_n \ . \tag{6}$$

For the I/O volume V_i we model:

$$V_i = \hat{V}_{I/O} L_0 \ . \tag{7}$$

Assuming region N has the largest refinement, we model the computational work W_o with the original method without TWN, that is needed to get at least the same resolution everywhere as with TWN, as:

$$W_o = \hat{W}_{mec} L_0 r_N \ . \tag{8}$$

4 Domain Decomposition with Parallelization

In DDP all (nesting) regions are divided into as many subdomains as available parallel processors. The computational work is distributed over the processors as evenly

as possible. For the parallel processing we use an SPMD approach with MPI. Extra overhead is caused by the communication among the processors, and by the fact that in MPI only one processor is guaranteed to be available for I/O. For long enough computations, we expect that I/O is the largest intrinsically serial part of the computations. Other serial parts include initializations ('bookkeeping' of data structures) and 'finalizations'.

If $P = P_x \times P_y$ processors are assigned, the basic region and each of the zoom regions are decomposed as evenly as possible into P_x parts in the west-east direction and into P_y parts in the south-north direction, such that a Cartesian grid of rectangular subdomains results in each region. Assuming a corresponding virtual Cartesian topology of $P_x \times P_y$ processors, each processor is assigned the corresponding subdomain of each region. Since the amount of work per grid column and per time step is fixed (Sect. 2) and because no self-adjusting time-steps are used, we have the advantage that only this static load balancing is needed.

Parallel computations need extra computer time due to inter-processor communication. It consists of three parts, namely intra-region communication, inter-region communication, and scatter and gather of I/O. Intra-region communication is associated with the advection calculations in a *single* region, where each processor needs data from neighbouring subdomains, residing on other processors, to determine the fluxes on and near its boundaries. Inter-region communication is needed for advection calculations in *multiple* regions, where for a particular region a processor may need data from its parent or child to compute fluxes at or near the interfaces. I/O-related communication consists of the scattering of input data from processor 0 to the other processors and of the gathering of output data back to processor 0.

We approximate the communication volumes as follows. The intra-region communication volume V_x for e*x*changes between neighbouring subdomains, the inter-region communication volume V_d for up*d*ates, and the communication volume V_s for the *s*catter and gather of the I/O are modelled as:

$$\begin{aligned} V_x &= \sum_{n=0}^{N} 2\hat{V}_{cop}(P_x L_{y,n} r_{y,n} + P_y L_{x,n} r_{x,n}) L_t r_{t,n} \\ &= \sum_{n=0}^{N} 2\hat{V}_{cop}(\frac{P_x}{L_{x,n} r_{x,n}} + \frac{P_y}{L_{y,n} r_{y,n}}) L_n r_n \ , \end{aligned} \tag{9}$$

$$V_d = \sum_{n=1}^{N} \hat{V}_{cop} L_n r_n \ , \tag{10}$$

$$V_s = \sum_{n=0}^{N} \hat{V}_{I/O} L_n \ , \tag{11}$$

where we have neglected the fact that on the boundaries of a (nesting) region, intra-region communication may be absent. Such communication procedures have to be set up M_x, M_d, and M_s times respectively. Let the factor N_p account for the fact that for any region its overlap in the parent may be distributed over more than 1

processor. Each child has only one parent, which is larger than or at most equal in size to the child, so each particular child subdomain has $1, 2,$ or 4 'parent' processors to communicate with. We then have:

$$M_x = \sum_{n=0}^{N} 4P_xP_yL_tr_{t,n} \ , \tag{12}$$

$$M_d = \sum_{n=0}^{N} 2P_xP_yN_pL_tr_{t,n} \ , \tag{13}$$

$$M_s = \sum_{n=0}^{N} P_xP_y\frac{n_{met}+n_{ver}+n_{emi}}{n_{sec}}L_t \ . \tag{14}$$

The total parallelization communication volume V_p and the total number M_p of communication set-ups are:

$$V_p = V_x + V_d + V_s \ , \ \ M_p = M_x + M_d + M_s \ . \tag{15}$$

5 Speed-Up

We determine the speed up due to nesting (S_n) and due to parallelization (S_p).

5.1 Nesting

We define the speed-up S_n of the nesting method as the ratio of computer time consumed for a particular choice of nesting regions and the amount of computer time needed when no nesting is used. Writing R_{cal} for the calculation speed (flop rate) and $Q_{I/O}$ for I/O speed (byte rate) of the actual computer, we express:

$$S_n \equiv \frac{(1+\omega_n)(\frac{W_o}{R_{cal}}+\frac{V_i}{Q_{I/O}})}{(1+\omega_n)(\frac{W_n}{R_{cal}}+\frac{V_i}{Q_{I/O}})} = \frac{\frac{W_o}{R_{cal}}+\frac{V_i}{Q_{I/O}}}{\frac{W_n}{R_{cal}}+\frac{V_i}{Q_{I/O}}} \ . \tag{16}$$

In actual computations there is overhead, which we have modelled with an overhead factor $\omega_n \geq 0$ depending on the coding, the compiler, and the computer.

Define $\lambda_n \equiv L_n/L_0 \leq 1$ and $\rho_n \equiv r_n/r_0 \geq 1$. Substituting equations (6) to (8) and $R_{cal}/\text{flop} = Q_{I/O}/\text{byte} = Q_{com}/\text{byte}$ in equation (16) yields:

$$S_n \simeq \frac{\rho_N}{\sum_{n=0}^{N}\lambda_n\rho_n}\left(\frac{1+\frac{\varepsilon_i}{r_N}}{1+\frac{\varepsilon_cZ+\varepsilon_i/r_0}{1+Z}}\right) \tag{17}$$

where $Z \equiv \sum_{n=1}^{N}\lambda_n\rho_n$. We use the estimates of $\hat{W}_{mec}$, $\hat{V}_{cop}$, and $\hat{V}_{I/O}$ to conclude $\varepsilon_i \simeq 3.3\cdot 10^{-7} \ll 1$ and $\varepsilon_c \simeq 5.5\cdot 10^{-4} \ll 1$. If $r_0 > 3.3\cdot 10^{-5}(^\circ)^{-1}$ then:

$$S_n \simeq \frac{\rho_N}{\sum_{n=0}^{N}\lambda_n\rho_n} \ . \tag{18}$$

5.2 Parallelization

We define the speed-up S_p of the parallelized method as the ratio of wall clock times spent when multiple processors (and subdomains) are used and when a single processor is used. Let $s \in [0,1]$ be the scalar fraction of the calculation work W_n with TWN on 1 processor, see equation (16). Introducing Q_{com} for the communication speed (byte rate) of the actual computer, we define S_p as:

$$S_p \equiv \frac{(1+\omega_n)(\frac{W_n}{R_{cal}} + \frac{V_i}{Q_{I/O}})}{(1+\omega_p)(1+\omega_n)\left\{\left[\frac{(1-s)\frac{W_n}{R_{cal}} + \frac{V_p}{Q_{com}}}{P_x P_y} + s\frac{W_n}{R_{cal}} + \frac{V_i}{Q_{I/O}}\right] + M_p \tau_{lat}\right\}} \quad . \tag{19}$$

Here τ_{lat} is the latency time for a communication set-up, and $\omega_p \geq 0$ is an organizational overhead factor related to the division of work into smaller amounts.

Substituting from the equations in Sects. 3 and 4, and dividing the numerator and the denominator by $\hat{W}_{mec}(1+Z)/(P_x P_y)$, we find for S_p:

$$\begin{aligned} S_p &= \left[\frac{P_x P_y (1-s+sP_xP_y)^{-1}}{1+\omega_p}\right] \cdot \\ &\qquad \frac{1 + \frac{\varepsilon_c Z + \varepsilon_i/r_0}{1+Z}}{1 + \frac{\varepsilon_c Z(2-s+sP_xP_y) + \varepsilon_c Q + (\varepsilon_i/r_0)Y + (\varepsilon_i/r_0 + \varepsilon_m T)P_xP_y}{(1+Z)(1-s+sP_xP_y)}} \quad , \end{aligned} \tag{20}$$

where use was made of $R_{cal}/\text{flop} = Q_{I/O}/\text{byte} = Q_{com}/\text{byte}$ and where

$$Q = \sum_{n=0}^{N} q_n \lambda_n \rho_n \leq 1 + Z \quad , \tag{21}$$

$$q_n = 2\left(\frac{P_x}{L_{x,n} r_{x,n}} + \frac{P_y}{L_{y,n} r_{y,n}}\right) \leq 1 \quad , \tag{22}$$

$$Y = \sum_{n=0}^{N} \lambda_n \leq 1 + Z \quad , \tag{23}$$

$$\varepsilon_m = \frac{\tau_{lat} Q_{com}}{(1+\omega_n)\hat{W}_{mec}} \lesssim 1.7 \cdot 10^{-4} \quad , \tag{24}$$

$$T = \frac{M_p}{L_0 r_0} = \sum_{n=0}^{N} \frac{((4+2N_p) r_{t,n} + N_m)}{(L_0 r_0 / L_t)} \simeq \frac{4+2N_p}{L_0 r_0} \sum_{n=0}^{N} L_t r_{t,n} \quad , \tag{25}$$

$$N_m = \frac{n_{met} + n_{ver} + n_{emi}}{n_{sec}} \simeq 1.39 \cdot 10^{-4} \quad . \tag{26}$$

Since $P_x \ll L_{x,n} r_{x,n}$ and $P_y \ll L_{y,n} r_{y,n}$ (a region is not divided into nearly as many subdomains as cells), we have $Q \ll 1 + Z$. The terms in equation (20) which 'threaten' the scalability most, are $f \equiv \varepsilon_m T P_x P_y / ((1+Z)(1-s+sP_xP_y))$ and

$(\varepsilon_i/r_0)P_xP_y/((1+Z)(1-s+sP_xP_y))$, apart from the factor $(1+\omega_p)^{-1}$. $N_{x,0}$, $N_{y,0}$ being the numbers of cells in the horizontal directions in region 0, we have:

$$f \lesssim \varepsilon_m(4+2N_p)\frac{\sum_{n=0}^{N} r_{t,n}/r_{t,0}}{\sum_{n=0}^{n}\lambda_n\rho_n}\frac{P_xP_y}{N_{x,0}N_{y,0}} \quad , \tag{27}$$

The first quotient in the above equation is no larger than $(N+1)r_{t,N}/r_{t,0}$, so (e.g.) $f < 0.01$ is certainly guaranteed if:

$$\frac{P_xP_y}{N_{x,0}N_{y,0}} < \frac{0.01}{\varepsilon_m(4+2N_p)(N+1)(r_{t,N}/r_{t,0})} \quad . \tag{28}$$

For realistic values (e.g. $N=3$, $N_p=1.5$, and $r_{t,N}/r_{t,0}=8$) this inequality holds for realistic numbers of processors in region 0. The second 'threatening' term can only become near or larger than 1 (and therefore dangerous) if *not* $(\varepsilon_i/r_0)P_xP_y \ll 1$, which only happens for unrealistically coarse resolutions in the basic region.

Using the smallness of many terms we approximate equation (20) as follows:

$$S_p \simeq \frac{P_xP_y(1-s+sP_xP_y)^{-1}}{(1+\omega_p)} \quad . \tag{29}$$

Defining σ as the scalar part of the computer *time* spent on actual computations by P_xP_y processors [6], then $s=\sigma/(\sigma+P_xP_y(1-\sigma))$, and

$$S_p \simeq \frac{(1-\sigma)P_xP_y+\sigma}{(1+\omega_p)} \quad . \tag{30}$$

We expect ω_p to increase with the number of processors P_xP_y, but we assume $|\omega_p| \ll 1$ for large enough problems. Then the speed-up is almost proportional to P_xP_y, depending on the smallness of σ.

5.3 Example

We give a realistic example of computations with an AQM and the expected speed-ups from TWN and DDP. First consider TWN with three regions: region 0 contains region 1, which in turn fully contains region 2. For the dimensions of the regions we specify $L_0 = 360\cdot 180L_t$, $L_1 = 90\cdot 60L_t$, and $L_2 = 15\cdot 10L_t$, whereas $r_0 = (1/9)(1/6)(1/7200)$, $r_1 = (1/3)(1/2)(1/3600)$, and $r_2 = (1/1)(1/1)(1/1800)$. With these numbers, we find $S_n \simeq 72$, which is a considerable speed-up factor.

Next we consider DDP with $P_x=4$, $P_y=8$, and $s=0.05$. Since the above resolutions and nesting are nowhere near the danger zone for (almost) perfect speed-up, we find $S_p \simeq 30.45/(1+\omega_p)$ which clearly favours DDP.

6 Conclusion

We propose two techniques which make atmospheric chemistry simulations far cheaper. The *computer time* can be strongly reduced by Two-Way Nesting. The

wall clock time is expected to be reduced according to almost ideal speed-up for large problems with Domain Decomposition with Parallelization. This brings more realistic simulations within reach. Developments towards this goal are underway.

References

[1] P. J. F. Berkvens, M. A. Botchev, M. C. Krol, W. Peters, and J. G. Verwer. Solving vertical transport and chemistry in air pollution models. Technical Report MAS-R0023, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, Aug 2000.

[2] P. J. F. Berkvens, M. A. Botchev, W. M. Lioen, and J. G. Verwer. A zooming technique for wind transport of air pollution. In R. Vilsmeier, D. Hänel, and F. Benkhaldoun, editors, *Finite Volumes for Complex Applications*. Hermes Science Publications, 1999. (Abridged from [3]).

[3] P. J. F. Berkvens, M. A. Botchev, W. M. Lioen, and J. G. Verwer. A zooming technique for wind transport of air pollution. Technical Report MAS-R9921, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, Aug 1999.

[4] P. J. F. Berkvens, M. A. Botchev, and J. G. Verwer. On the efficient treatment of vertical mixing and chemistry in air pollution modelling. In M. Deville and R. Owens, editors, *CD-ROM Proceedings of the 16th IMACS World Congress On Scientific Computation, Applied Mathematics and Simulation*, 2000. Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, Aug 21-25, 2000.

[5] F. J. Dentener, J. Feichter, and A. Jeuken. Simulation of transport of ^{222}Rn using on-line and off-line global models at different horizontal resolutions: A detailed comparison with measurements. *Tel. Ser. B*, 51:573–602, 1999.

[6] D. R. Emerson. Introduction to parallel computers: architecture and algorithms. In P. Wesseling, editor, *High Performance Computing in Fluid Dynamics*, volume 3 of *ERCOFTAC Series*, chapter 1, pages 1–42. Kluwer Academic Publishers, Dordrecht, Boston, London, 1996.

[7] M. Heimann. The global atmospheric tracer model TM2. Technical report, DKRZ (Deutsches Klimarechenzentrum), Hamburg, Germany, 1995.

[8] M. C. Krol, W. Peters, M. A. Botchev, and P. J. F. Berkvens. A new algorithm for two-way nesting in nlobal models: principles and applications. In B. Sportisse, editor, *Proceedings of the Second Conference on Air Pollution Modelling and Simulation 2001, Champs-sur-Marne, France*. Springer Verlag, to appear in 2001. submitted.

[9] Url http://www.phys.uu.nl/∼peters/TM3/TM3S.html.