PERGAMON

Atmospheric Environment 33 (1999) 1853–1860

ATMOSPHERIC ENVIRONMENT

Atmospheric pollution transport: the parallelization of a transport & chemistry code

Marc Martin[a], Olivier Oberson[a], Bastien Chopard[a,*], Frank Mueller[b], Alain Clappier[b]

[a]*Computer Science Department, CUI, University of Geneva, 24, rue du General Dufour, CH-1211 Geneva 4, Switzerland*
[b]*DGR EPFL, CH-1015 Lausanne, Switzerland*

Received 25 November 1997; received in revised form 30 September 1998; accepted 1 October 1998

Abstract

We discuss a model for computing the chemical reactions occurring among air pollutants and predict their dispersion in the atmosphere. We consider a parallel implementation on a CRAY T3D, using up to 128 processors. A regular subdomain decomposition is considered to optimize inter-processor communications, while load balancing – the main technical difficulty to obtain a good efficiency – is achieved by re-assigning temporarily pieces of work to other processors, according to a global strategy. The performance of our implementation is described by an analytical relation whose validity is checked on realistic data sets. A speedup of 100 is obtained with 128 processors. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Air quality models; Air pollution; Numerical simulations; Parallelization; Load balancing

1. Introduction

A better control and understanding of atmospheric pollution phenomena relies more and more on numerical models and large scale computer simulations. Due to the presence of many chemical and physical processes, an air quality simulation requires a considerable computing power and large memory space. The run time of a realistic model on a high-end workstation can be prohibitive (up to one day for a 24 h prediction).

In this paper we propose the parallelization of an air pollution code and show the benefit of having many processors cooperate to the solution. We present a load balancing strategy and give an analysis of the performance we achieve on p processors.

The current version of our chemical solver is working with more than 100 equations involving around 54 species. The quantities taken into account are: (i) The meteorological fields affecting pollution transport and chemistry (wind acts on the transport phenomenon and atmospheric stability; temperature, pressure, humidity and solar radiations modifies the chemical reaction speeds); (ii) air quality, used to define initial and boundary conditions on the simulation domain. Two categories of pollutant emission are considered: dispersive sources like road traffic and point sources like factories; (iii) ground occupation (forest, lake, town) which is used to calculate the surface roughness and the elimination by deposition of the active chemical species from the

*Corresponding author. Tel.: 0041 22 705 7623; fax: 0041 22 705 7780; e-mail: Bastien.Chopard@cui.unige.ch.

1352-2310/99/$ – see front matter
PII: S1352-2310(98)00348-3

atmosphere. All these data are obtained from topographical maps and measurements by weather stations.

2. Model description

2.1. Gas-phase mechanism

The chemistry used in our model is the condensed LCC mechanism (Lurmann et al., 1987). The original gas-phase mechanism was developed to represent the photo-oxidation of non-methane organic compounds and nitrogen oxides in urban-scale air quality simulation models. The gas-phase chemical mechanism has been expanded (Harley et al., 1993) in order to treat explicitly the biogenic emissions and fuel additives. Two aerosol species are introduced to take into account the gas phase production of ammonium nitrate (Russell et al., 1983) and sulfate. The modified LCC gas-phase chemical mechanism was successfully used throughout a number of photosmog episode simulations (Giovannoni et al., 1995; Grossi et al., 1996; Kuebler et al., 1996) and, therefore, is used as a sound basis for the simulations in the parallelized model system. Summarizing, the chemical gas phase system considers 35 gas-phase species, 2 aerosol species, and 9 steady-state species. The complete list of species defined in the gas phase chemical system is given in Table 1. These species are coupled via 106 gas-phase reactions (Lurmann et al., 1987).

Table 1
Complete list of species considered in the gas-phase chemical mechanism

List of species			
Species code	Species name	Species code	Species name
	Gas-phase prognostic species		Gas-phase prognostic species
NO	Nitric oxide	ALKA	C4 + alkanes
NO_2	Nitrogen dioxide	ETHE	Ethene
O_3	Ozone	ALKE	C3 + alkenes
HONO	Nitrous acid	TOLU	Toluene
HNO_3	Nitric acid	AROM	Higher aromatics
HNO_4	Pernitric acid	DIAL	Unknown dicarbonyls
N_2O_5	Dinitrogen pentoxide	CRES	Cresols
NO_3	Nitrogen trioxide	NPHE	Nitrophenols
HO_2	Hydroperoxy radical	H_2O_2	Hydrogen peroxide
CO	Carbon monoxide	MEOH	Methanol
HCHO	Formaldehyde	ETOH	Ethanol
ALD_2	Higher aldehydes	MTBE	Metyl tert-butyl ether
MEK	Methyl ethyl ketone	ISOP	Isoprene
MGLY	Methyl glyoxal	NH_3	Ammonia
PAN	Peroxyacetyl nitrate	ALKN	Alkyl nitrate
RO_2	Total RO_2 radicals	SO_2	Sulfur dioxide
MCO_3	CH_3CO_3 radical	SO_3	Sulfur trioxide
CH_4	Methane	NIT	Particulate nitrate
SULF	Particulate sulfate		
	Aqueous-phase prognostic species		Aqueous-phase prognostic species
O_3(a)	Ozone	H_2O_2(a), HO_2^-	Hydrogen peroxide
HNO_3(a), NO_3^-	Nitric acid	HCl(a), Cl^-	Hydrochloric acid
NH_4OH, NH_4^+	Ammonia	CO_2(a), HCO_3^-, CO_3^{2-}	Carbon dioxide
SO_2(a), HSO_3^-, SO_3^{2-}	Sulfur dioxide	H_2SO_4(a), HSO_4^-, SO_4^{2-}	Sulfuric acid
OH^-, H^+	Water, pH		
	Steady-state species		Steady-state species
OSD	Oxygen–singlet D(^{1}D)	RO_2N	Alkyl nitrate RO_2
O	Atomic oxygen (^{3}P)	RO_2P	Phenol RO_2
OH	Hydroxyl radical	BZN_2	Benzaldehyde N-RO_2
RO_2R	General RO_2 #1	BZO	Phenoxy radical
R_2O_2	General RO_2 #2		

2.2. Condensed aqueous-phase mechanism

In order to test the capabilities of massively parallel computations the gas-phase chemical mechanism was extended by a simple aqueous-phase reaction system. The liquid phase chemistry is activated for grid cells with liquid water (i.e. clouds are present in the respective grid element). Since the additional consideration of aqueous-phase chemistry in local areas of the model domain requires a considerable extra amount of CPU time (1996), the case of multi-phase chemistry is well suited for the involvement of parallelization.

The condensed mechanism follows essentially the work of Walcek and Taylor (1986) and Lin and Chameides (1991). Beside the mass transfer of the species SO_2, O_3, H_2O_2, CO_2, HCl, HNO_3, and NH_3 (see also Table 1) between the gas and the aqueous-phase, it comprises the most essential oxidation pathways of S(IV) to S(VI) by O_3 and H_2O_2 in the liquid-phase (Lagrange et al., 1991). The involved gas-phase species have primary importance for the acidity development in the liquid-phase (Pandis and Seinfeld, 1989). The formulation used for the sulfur oxidation reactions follows Moeller and Mauersberger (1995). To be able to use the full advantage of the numerical solver (Young and Boris, 1977), the hydrogen ion concentration is treated as a predictive variable.

2.3. Numerical approach

2.3.1. Resolution principle

In order to use the operator splitting method (Marchuk, 1975) the atmospheric diffusion Eq. (1), which includes advection (first term on the rhs. of Eq. (1)), diffusion (second term), and chemical sources/sinks (third term), is decomposed into one-dimensional transport and chemical problems.

$$\frac{\partial c_j(\mathbf{r}, t)}{\partial t} = -\nabla(\mathbf{v}c_j) + \nabla(K\nabla c_j) + S_j(c_j, t), \tag{1}$$

where c_j is the concentration of chemical species j, $\mathbf{v}$ is the mean wind velocity, K is turbulent diffusion coefficient, S_j is the chemical source. For the x-direction, this results in

$$\frac{\partial c_j}{\partial t} = L_a c_j = -\frac{\partial u c_j}{\partial x}, \quad \frac{\partial c_j}{\partial t} = L_d c_j$$

$$= -\frac{\partial}{\partial x} K \frac{\partial c_j}{\partial x}, \quad \frac{\partial c_j}{\partial t} = L_c c_j = S_j, \tag{2}$$

where L_a, L_d, and L_c are the operators for advection, diffusion, and chemistry, respectively. The complete solution is obtained from the sequence

$$c_j^{n+1} = A_x(\Delta t)A_y(\Delta t)A_z(\Delta t) \times A_c(2\Delta t)A_z(\Delta t)A_y(\Delta t)A_x(\Delta t)c_j^{n-1}, \tag{3}$$

where Δt represents the numerical time step according to the Courant criterion and n is the time level. The A_i are the numerical approximations of the transport and chemical operators given by

$$A_x = L_a(x)L_d(x), \quad A_y = L_a(y)L_d(y),$$
$$A_z = L_a(x), \quad A_c = L_c L_d(z). \tag{4}$$

The arguments x, y, z are indicating the spatial direction of the operator. Vertical diffusion is treated in a special way, as a source/sink term which is implemented in the chemical solution mechanism. The advantage of this technique is the application of the highly accurate, and computationally very efficient, chemical solver on both the stiff chemical equations and the diffusion process, which requires small time steps.

The dry deposition and emissions are treated as boundary conditions at the ground

$$-K\frac{\partial C_i}{\partial z} = E_i - v_g^i C_i,$$

where K is the vertical eddy diffusivity, E_i the emission flux, and v_G^i the dry deposition velocity for species i.

2.4 Chemical solver

The equations describing the interaction between the pollutants are coupled, non-linear, ordinary differential equations. The vector function $S_j(t, c_j)$ is the source function in the ensemble averaged atmospheric diffusion equation for the chemical species concentration c_j (Eq. (5)), which is a common basis for most air quality models:

$$\left(\frac{\partial c_j(t)}{\partial t}\right)_{\mathrm{CHEM}} = P_j(t) - L_j(t, c_i)c_j = S_j(t, c_j). \tag{5}$$

Since the chemical module is set up as a column over all vertical levels, $S_j(t, c_i)$ represents the chemical transformations, emissions, dry deposition, and the contributions of vertically turbulent transport of the jth chemical compound. In Eq. (5) P_j and $L_j c_j$ are the production and loss terms, respectively, while $L_j(t, c_i)$ is a matrix representing the interaction between the jth chemical component with the corresponding chemical species c_i.

In order to solve the chemical reaction system, we employ the highly efficient "hybrid scheme" (Young and Boris, 1977). This is a predictor-multi-corrector algorithm classifying the equations as stiff or non-stiff, based on the ratio of the step-size to characteristic reaction-time. Very rapid reacting species are pre-selected as steady-state species and eliminated from the system of equations. Because the hybrid method is not strictly mass conservative, a built-in feature is used to reduce the step size if convergence cannot be achieved. No explicit mass conservation technique is applied. In order to optimize

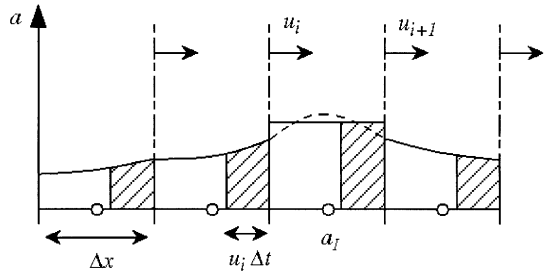

Fig. 1. Principle of the PPM algorithm. Piecewise parabolic interpolation points are fitted to conserve mass and give a continuous function in one dimension. If the fit leads to an internal maximum or minimum, it is replaced in that interval by a constant value. The solid line is the function as used in the method. The dashed line is the polynomial fit used before correcting for the local maxima found in that interval.

the integration with respect to mass conservation and computer time, we set the minimum time step for the integration to $t_{\min} = 5 \times 10^{-6}$ s and the relative error to $\varepsilon = 0.1\%$. For an integration interval of 1 h, we obtain for all simulated species, concentrations which differ much less than 1% from a reference solution computed with the LSODE method (Hindmarsh, 1980).

2.5. Advection scheme

The advection operator, L_a, is calculated with the Piecewise Parabolic Method (PPM) developed by (Colella and Woodward, 1984). The concentration c is defined as an average over each computational grid cell, whereas the wind velocity is defined at the cells faces (see Fig. 1).

The concentration at time $n+1$ in the cell I, is calculated as a function the value at time n

$$c_i^{n+1} = c_I^n - \frac{\Delta t}{\Delta x}(v_{i+1}\bar{c}_{i+1}^x - v_i\bar{c}_i^x), \tag{6}$$

where $(\Delta t/\Delta x)v_i\bar{c}_i^x$ is the flux at face i, and $\bar{c}_i^x$ represents the average value which crosses face i during the time step Δt:

$$\bar{c}_i^x = \frac{1}{v_i\Delta t}\int_{-v_i\Delta t}^{0} c(x)\,dx, \tag{7}$$

where $c(x)$ is a polynomial interpolation of a over the grid. Note that the flux depends only on $c(x)$ in the upwind cell. The interpolation of a high-order polynomial can produce local extrema which can lead to over- and under-shooting. In PPM, Colella and Woodward (1984) interpolate the value for tracer with continuous parabolic pieces. The parabola is replaced by a constant value, or a linear interpolation, if it is found that a local maximum or minimum results from the parabolic fit (see Fig. 1). This technique is explained in detail in (Carpenter et al., 1990).

3. Parallelization

The above differential equation is solved on a three-dimensional domain which is spatially discretized in cells. The most important part of the computation is the chemistry phase because of the numerical methods that are used (stiff equations, multi-step technique). It turns out that this reaction phase involves local information only and can be computed independently for each cell. Thus there is a large degree of parallelism to be exploited because many processors could be used simultaneously to calculate the reaction at various places. Transport and diffusion phases, on the other hand, require communications across the processors. However, these are regular and local communications which do not prevent the scalability of the application.

Here we consider an implementation on the CRAY T3D installed at EPFL, using the message passing programming model. The CRAY T3D is a distributed memory architecture which contains 256 DEC-alpha processors, each with 150 Mflops and 64 Mbyte of memory. The total peak performance is 38 Gflops. Communications among the processors are done through an interconnection network with the topology of a 3D torus.

3.1. Data decomposition

Because of communication due to transport computation, we must keep subdomains as regular as possible at least during transportation-phase (Skalin et al., 1997). So we assume a domain with $L \times l \times h$ cells, where L and l represent the length and width of the domain and h its vertical thickness. The work is distributed across a logical array of $P \times p$ PEs (Processing Elements) so that each processor receives a rectangular block of adjacent columns of size $(L/P) \times (l/p) \times h$.

Due to the way the differential operators are discretized, each lattice cell must know the state of its three nearest neighbors in order to compute the transport step. For this reason, we impose that (L/P) and (l/p) be at least 3 so that the communication pattern remains simple.

Columns are the basic unit of work in the model. However, the amount of computer time necessary to solve the chemical equations may change from one column to the other. This is due to several physico-chemical reasons, such as the effect of radiations on the reactions, the presence of emission sources on the ground or the existence of water in the atmosphere. Here we focus on a situation where both aqueous and gaseous chemistry can be present but the other sources of work imbalance could be treated in the same way.

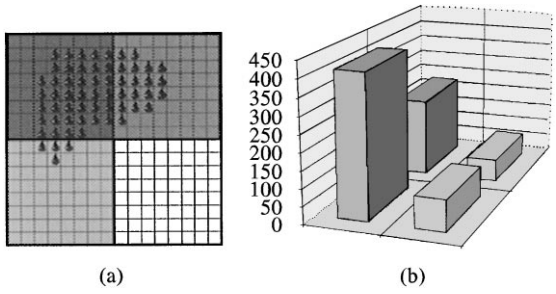

Fig. 2. (a) Each square represents a vertical column of h cells. Different shaded regions are assigned to different processors (4 PEs here). Droplets show the columns containing water and illustrate a situation of unbalanced load. (b) Time per processor to compute each domain without a re-distribution of the work. Here, a time 1 and 15 are assigned, respectively, to gaseous and aqueous columns.

Because it involves more complicated chemical reactions, a cell which contains water requires 10–15 more computing time than a cell without water. Columns with water are usually not uniformly distributed across the domain. Their location depend on the position of clouds, which move.

Since the transport and diffusion phases require full synchronization of the PEs, the processors with less work have to wait for those having more computations, thus leading to an important loss of efficiency.

A typical example of an uneven water distribution is shown in Fig. 2a: PE_1 has 39 fully aqueous columns while PE_2 and PE_3 have only 18 and 3, respectively. The last PE has none of them. Fig. 2b shows the difference of run time resulting from this situation.

3.2. Load balancing

An even load distribution across the PEs requires an efficient load balancing algorithm. Such algorithms have been proposed by various authors (Kuchen and Wagener, 1990; Dabdub and Seinfeld, 1994; Elbern, 1997; Michalakes, 1997; Schättler and Krenzien, 1997).

Dabdub and Seinfeld (1994) uses a static decomposition; a master PE distributes, row-wise, the columns to the PEs as fairly as possible. This leads to non-rectangular subdomains. Such a decomposition makes it easy the conversion from a sequential to a parallel program. Reported speedup is 30 with 256 nodes.

Michalakes (1997) uses arbitrary subdomains (not necessarily rectangular). Dynamic load balancing is obtained by modifying subdomains size (exchanging columns locally). This technique allows a very fine grain load balancing, but requires complicate inter-processors communications during the transport steps. Calculating a new data distribution may cost a lot of time. Performance on a IBM SP2 ranges from 176 Mflop/s on 4 PE up to 1592 Mflop/s on 64 PE.

Schättler and Krenzien, 1997) suggests to use the previous step timing to compute a new columns redistribution. Such a solution is model independent. He also brings to the fore the problem of a non-rectangulars decomposition as requiring more complex communications.

As opposed to the above approaches, we propose here to concile efficient communications with dynamic load balancing. We consider two alternating domain decompositions: columns with water are distributed on the processors which are less loaded, so that aqueous chemistry is shared among the PEs in a fair way. The transport phase, on the other hand, is still performed on the initial regular cartesian decomposition, for the sake of efficiency.

The work associated to each columns can be estimated in several ways: the percentage of aqueous cells in the column, or the amount of CPU time devoted to this columns during the last computation step. In the present implementation we consider only significant sources of load imbalance, such as those produced by emissions or aqueous cells in a column. We disregard small increases of CPU that may be induced in gaseous chemistry by variations of the time step in the chemical solver. This decision is justified by the observation that, in the typical episodes we want to simulate, the load imbalance is mainly due to a rather small fraction of columns that are much longer to compute than the others. With a small overhead, these columns can be fairly distributed among other PEs. A finer load distribution is also possible, with the same method, but with a more important data reorganization. The price for such an overhead increase may not be justified in regard to the benefit of a fine grain load distribution.

Our algorithm works as follows: each PE sends a list of its overloaded columns to a master PE, arbitrary chosen. This master deduces the total quantity of work due to these columns and computes a new, balanced distribution. To communicate this new work distribution, the master performs a one-to-all personalized communication so that:

- Overloaded processors receive the columns they no longer should compute and the addresses of the PEs these columns must be sent to.
- The other processors receive the list of extra columns they will have to compute and wait until all of them are received.

Once all columns have been computed, the extra columns are sent back to their owner so that the transport step can be performed. The same redistribution pattern can be used for several consecutive time steps if the source of load imbalance varies slowly in time. However, in our tests this distribution is recomputed after every

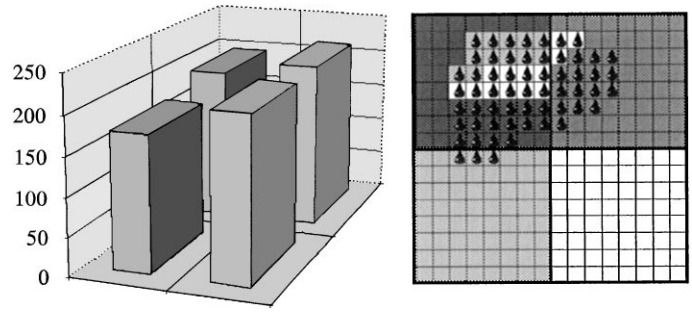

Fig. 3. Situation of Fig. 2 after load balancing: on the left load distribution and on the right work distribution.

transport step because, due to the wind, the aqueous columns (clouds) move fast.

In the case described in Fig. 2 there is total of fully 60 aqueous columns and, ideally, each PE should get 15. As shown in Fig. 3, the master decides to give 12 PE_1's columns to PE_3, and 12 PE_1's columns plus 3 PE_2's columns to PE_4.

4. Performance: the benefit of parallelism

This section presents a performance model describing the run time of our parallel implementation. In order to check the validity of our analysis, our transport/chemistry code is tested on a realistic situation, whose precise description (location and size of the domain, emission inventory, meteorological fields...) is given in (Kuebler et al., 1996). The primary purpose of this simulation was to study the photochemical pollution over the swiss plateau. To this end, the chosen period was 28–30 July, 1993, characterized by high-pressure conditions and cloud free skies. Without any cloud the original data were not adapted for an evaluation of the load balancing procedure. To remedy this problem an artificial, but realistic, cloud is introduced in the simulation domain, and transported as another chemical species during the calculation.

Let a be the average local CPU time for each cell (taking into account computations devoted to transport, gaseous and aqueous chemistry). The sequential computing time is then $T_{\text{seq}} = aLlh$.

In parallel, a time T_{lb} is devoted to column shifts and the computation of a balanced distribution. Once load is fairly distributed, the CPU time spent by each processor can be estimated as $a(Llh/Pp)$. The communications take place on regular subdomains and the time necessary to receive data from adjacent processors is proportional to the volume of data exchanged, i.e. the subdomain perimeter. Due to the rectangular geometry, the data in the interprocessor overlap can be packed in one message for each boundary. If the domain is large enough, the message latency can be neglected in compariason with the message size. Therefore, the parallel run time can be estimated as

$$T_{\text{par}} = T_{\text{lb}} + a\frac{Llh}{Pp} + 2bh\left(\frac{L}{P} + \frac{l}{p}\right), \tag{8}$$

where b is proportional to the interconnect bandwidth (note that the width 3 of the overlap is included in b). T_{lb} and b are related to the parallel overhead.

For the problem size under consideration and the number of processors at hand, a good load balancing is achieved with an overhead less than 1% of the total computation. Thus we assume that $T_{\text{lb}} \approx 0$.

In this case, Eq. (8) gives for the speedup on $N = pP$ processors

$$S_N = \frac{T_{\text{seq}}}{T_{\text{par}}} = \frac{N}{1 + \dfrac{2b}{a}\left(\dfrac{P}{L} + \dfrac{p}{l}\right)}.$$

Speedups close to N can be obtained provided that $l \gg p$ and $L \gg P$, or as soon as $b \ll a$.

From the above relations, the most efficient data distribution on the PEs consists of using square subdomain ($L/P = l/p$). This minimizes the boundary area between adjacent subdomains and minimizes parallelism overhead. In this case Eq. (8) can be cast into

$$\frac{pT_{\text{par}}}{l} = ah\frac{l}{p} + 4bh \tag{9}$$

and we conclude that $(pT_{\text{par}})/l$ must grow linearly with (l/p). This is verified in Fig. 4 for most measurements, except for an anomaly that appears for $l = L = 24$ where the computing time is unusually high. Although perfectly reproducible, this seems an accidental effect of the computer which does not affect algorithm performances for other domain sizes.

We can estimate a and b from the slope and intercept of the least-squares plot shown in Fig. 4. We find $a = 1.09$ s and $b = 0.36$ s. These figures are obtained with 39 iterations of the model and correspond to 1 h of simulation.

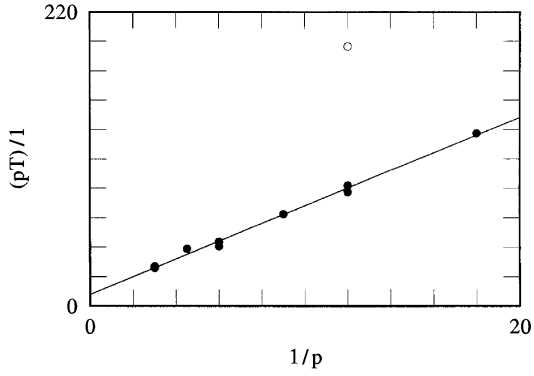

Fig. 4. Measured performance for square subdomains: the dots show $(pT_{par})/l$ versus (l/p); as predicted by Eq. (9) a linear dependence is found. From the slope and intercept of the least square fit, we may compute estimates for a and b. We find $a = 1.09$ s and $b = 0.36$ s. The white dot shows the anomalous timing for $l = L = 24$.

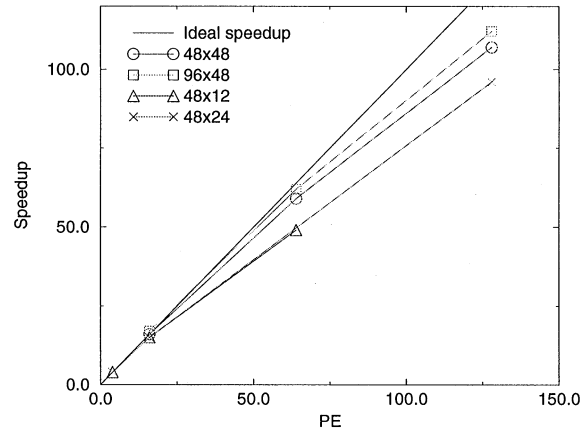

Fig. 5. Indicative speedup, for various PE configuration (4 up to 128) and ground area (576 cells to 4608). The reference value (time for 1 PE) is calculated as four times the fastest 4-PE timing because of the unavailibilty of 1-PE timing on the T3D.

For many values of l, L and N, the optimal square subdomains decomposition is not possible. Using the general formula (8) with $T_{lb} = 0$, we can estimate the computing time for various cases. The corresponding timings obtained for our transport/chemistry scenario on the T3D lie within 5% of the theoretical predictions given by the above values of a and b.

On the CRAY T3D with 128 processors, we have been able to obtain a speedup of 100 compared with a 1 DEC-alpha processor. Fig. 5 illustrates this behavior for actual computations. The T3D parallel version also runs 20 faster than a sequential SUN Sparc 10 workstation. Thus, one night of SUN computation can be converted in an half hour on the T3D. This definitely proves the usefulness of parallelizing such an application.

I/O times have not been considered in the above performance measurements. However, they are a noticeable point and some design decisions have been taken here. In order to avoid out of control simultaneous accesses to a shared file system, I/O are centralized on a master PE. The other PEs read and write their data through the master using communication primitives. With 64 PEs, a 24 hr simulation takes 41 min, with intermediate disk accesses every hour, and 39 min without. This 5% difference increases to 10% when using 128 PEs. A parallel file system, with disk striping, would of course alleviate the problem. Another solution is to use binary data files that are 10 times faster to handle than the text files (44 Mbyte) that are currently used.

5. Conclusions

This paper presents an efficient parallel implementation of a transport and chemistry code for atmospheric pollution. A dynamic load balancing scheme has been proposed which is based on two alternating domain decompositions: (i) regular subdomains, used for the transport phase to optimize inter-processor communications; (ii) irregular domains, obtained by an exchange of columns among the processors. This dynamic re-distribution is based on a global decision, made by a single processor, according to the local work excess. The back and forth motion of columns between the two partitions is efficient when the local computation (chemistry) is intensive and when the columns responsible for a significant load imbalance are easily identified and not too numerous.

The speedup of 100 we report here shows that our approach is well suited to the present problem and yields a better efficiency than the previous approaches published in the literature. The simple performance model we propose is valid even with unbalanced scenarios because the load balancing overhead is low. The coefficients a and b give a machine-specific performance metrics independent of the problem size and the number of processor used in the computation.

Currently, we also consider local load balancing strategies and the parallelization of our air quality model on clusters of workstations or PCs.

Acknowledgements

We thank Michel Roche for useful advises about the Cray T3D. This work is supported by the Swiss National Science Foundation.

References

Carpenter, R.L., Droegemeier, K.K., Woodward, P.R., Hane, C.E., 1990. Application of the piecewise parabolic method (PPM) to meteorological modeling. Monthly Weather Review 118, 586–612.

Colella, P., Woodward, P.R., 1984. The piecewise parabolic method (PPM) for gas-dynamical simulations. Journal of Computational Physics 54, 174–201.

Dabdub, D., Seinfeld, S., 1994. Air quality modeling on massively parallel computers. Atmospheric Environment 28, 1679–1688.

Elbern, H., 1997. Atmospheric Environment 31.

Giovannoni, J.-M., Clappier, A., Russell, A.G., 1995. Ozone control strategy modeling and evaluation for Athens, Greece: ROG vs. NO_x effectiveness and the impact of using different wind field preparation techniques. Meteorology and Atmospheric Physics 57, 3–20.

Grossi, P., Giovannoni, J.-M, Russell, A.G., 1996. Intercomparison of meteorological models applied to the Athens Area and the effect on Photochemical pollutant predictions. Journal of Applied Meteorology 35, 993–1008.

Harley, R.A., Russell, A.G., McRae, G.J., Cass, G.R., Seinfeld, J.H., 1993. Photo-chemical modeling of the southern California air quality study. Environmental Science and Technology 27, 378–388.

Hindmarsh, A.C., 1980. LSODE and LSODI, Two new initial value ordinary differential equation solvers. ACM-SIGNUM Newsletters 15(4), 10–11.

Kuchen, H., Wagener, A., 1990. Comparison of Dynamic Load Balancing Strategies. http://fiachra.ucd.ie/~david/loadbalancingpapers.html, 1990.

Kuebler, J., Giovannoni, J.-M., Russell, A.G., 1996. Eulerian modeling of photochemical pollutants over the Swiss Plateau and control strategy analysis. Atmospheric Environment 30, 951–966.

Lagrange, J., Paalares, C., Wenger, G., Lagrange, P.H., 1991. Mechanism of the aqueous oxidation of sulfur (IV) to sulfur (VI) by strong oxidants. EUROTAC Annual Report 6, 28–33.

Lin, X., Chameides, W.L., 1991. Model studies of the impact of the chemical inhomogeneity on SO_2 oxidation. Journal of Atmospheric Chemistry 13, 109–129.

Lurmann, F.W., Carter, W.P., Coyner, L.A., 1987. A surrogate species chemical reaction mechanism for urban-scale air quality simulation models. Technical Report, EPA Report No. 600/3-87/014A.

Marchuk, G.I., 1975. Methods of Numerical Mathematics. Springer, New York.

Michalakes, J., 1997. MM90: A scalable parallel implementation of the Penn State/NCAR Mesoscale Model (MM5). Parallel Computing 23, 2173–2186.

Moeller, D., Mauersberger, G., 1995. Aqueous chemical reaction system used in cloud chemistry modelling, In: Flossmann, A., Cvitas, T. (Eds.), Clouds: model and mechanisms. EUROTAC ISS, Garmisch-Partenkirchen.

Mueller, F., Clappier, A., Kuebler, J., Reiss, H., Martilli, A., Russell, A.G., Krueger, B.C., van den Bergh, H., 1996. Further development of a comprehensive air quality model, to include aqueous phase and aerosol chemistry for calculating fluxes of ozone, as well as nitrogen and sulfur containing pollutants in Europe. Technical Report, BBW-EUREKA (93) 2 EPFL Lausanne, Switzerland.

Pandis, S.N., Seinfeld, J.H., 1989. Sensistivity analysis of a chemical machanism for aqueous phase atmospheric chemistry. Journal of Geophysical Research 1994, 1105–1126.

Russell, A.G., McRae, G.J., Cass, G.R., 1983. Mathematical modeling of the formation and transport of ammonium nitrate aerosol. Atmospheric Environment 17, 949–964.

Schättler, U., Krenzien, E., 1997. The parallel Deutschland Modell – a message passing version for distributed memory computers. Parallel Computing 23, 2215–2226.

Skalin, R., Bjorge, D., 1997. Implementation and performance of a parallel version of the HIRLAM limited area atmospheric model. Parallel Computing 23, 2161–2172.

Walcek, C.J., Taylor, G.R., 1986. A theorical method for computing vertical distributions of acidity and sulfate production within cumulus clouds. Journal of Atmospheric Science 43, 339–335.

Young, T.R., Boris, J.P., 1977. A numerical technique for solving stiff ordinary differential equations associated with the chemical kinetics of reactive flow problems. Journal of Physical Chemistry 81, 2424–2427.