ELSEVIER

Parallel Computing 22 (1996) 111–130

PARALLEL COMPUTING

Practical aspects and experiences

Parallel computation in atmospheric chemical modeling

Donald Dabdub, John H. Seinfeld *

Department of Chemical Engineering, California Institute of Technology, Pasadena, CA 91125, USA

Received 20 January 1995; revised 13 July 1995, 7 August 1995

Abstract

The porting of atmospheric chemical dynamic models to massively parallel computers presents interesting computational challenges. Strategies for the parallelization of the transport and chemistry operators of atmospheric models are outlined. The use of parallel buffers to perform input/output operations is described. Results are given for implementation of the CIT urban air pollution model on distributed memory multiple instruction/multiple data (MIMD) machines ranging from a cluster of workstations to a 512 node Intel Paragon. A speed-up factor of 94.9 is achieved when the I/O, transport, and chemistry portions of the model are performed in parallel using 256 nodes of the Intel Paragon.

Keywords: Atmospheric models; Air quality modeling; Parallel I/O; Distributed memory multiprocessors; Workstation cluster

1. Introduction

Atmospheric chemical dynamic models are mathematical descriptions of atmospheric transport and transformation that predict how species concentrations change in response to changes in emissions over scales ranging from urban ($\sim 10^4$ km^2) to regional ($\sim 10^6$ km^2) to global. Such models are, for example, the computational tools used to determine emission control strategies needed to comply with urban air quality standards [21]. At the urban and regional scales, these models are often referred to as air quality models (AQMs).

The basic approach to air quality modeling is Eulerian, in which atmospheric dynamics are simulated on a three-dimensional Cartesian grid. Typical horizontal grid cell dimensions are about 5 km $\times$ 5 km for urban applications and 100

* Corresponding author. Email: john_seinfeld@starbase1.caltech.edu

0167-8191/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0167-8191(95)00063-1

Table 1
Aspects of current urban and regional air quality models

Model	Scale	Vertical resolution	Horizontal resolution	Aqueous phase	Aerosol phase
UAM Morris and Myers [20]	Urban	4 layers up to 1.5 km	4–10 km	No	No
CIT McRae and Seinfeld [18] Harley et al. [13]	Urban	5 layers up to 1.5 km	5 km	No	No
ROM Schere and Wayland [27]	Regional	3 layers up to 2 km	18 km	No	No
RADM-II Chang et al. [7]	Regional	15 layers up to 10 km	80 km	Yes	No
STEM-II Carmichael et al. [5]	Regional	10 to 14 layers up to 6 km	10–56 km	Yes	No
ADOM Venkatram et al. [35]	Regional	12 layers up to 10 km	60–120 km	Yes	No
Next Generation AQMs	Urban and Regional	25–30 layers up to 10 km	4–80 km	Yes	Yes

km × 100 km for regional applications. Current urban-scale atmospheric models subdivide the vertical dimension into 4–7 layers (up to an altitude of ~ 1500 m), while regional models use 6–15 layers (up to ~ 10000 m). These resolutions are consistent with those of source emission inventories and the ability to resolve meteorological processes that drive the flow fields. Increasing the grid resolution increases the computational cost as well as the data acquisition requirements. Table 1 presents some characteristics of existing and anticipated AQMs.

Historically, the degree of physics and chemistry incorporated in atmospheric models has been limited to some extent by the computational power available [30]. The basic atmospheric chemical dynamic model describes gas-phase species. Pilinis and Seinfeld [25] incorporated the particle (aerosol) phase into a three-dimensional gas-phase AQM, leading to an increase in computing time by a factor of 3. Other models [5,7,35] incorporate aqueous-phase processes in an effort to simulate acid deposition phenomena. As shown in Table 1, the next-generation of AQMs is anticipated to incorporate both aqueous and aerosol phases. These next-generation models will require state-of-the-art massively parallel supercomputers to achieve reasonable [1] wall-time turnaround. Clusters of workstations will provide the entry-level computational power demanded by models of such complexity.

The goal of this paper is to address the techniques for parallel computation with atmospheric models by means of implementation of a state-of-the-art urban-scale air quality model on a massively parallel architecture. Table 2 summarizes some of the current advances towards the efficient parallel implementation of AQMs. Pai and Tsang [23,24] studied splitting finite element techniques used in the parallel

[1] Less than 1-hour of computing time per 24-hours of simulation time.

Table 2
Parallel implementation of urban and regional atmospheric models

Reference	Work description	Maximum speed-up	Architecture
Pai and Tsang [23]	Investigate the effectiveness of parallelization of time-splitting finite element schemes using MIMD machines. Assess the portability of schemes to shared-memory parallel computers.	13.1	Sequent Symmetry S81
Pai and Tsang [24]	Develop a parallel time-splitting finite element scheme for three-dimensional first- and second-order dispersion models to simulate plume behavior of passive contaminants in a convective boundary layer.	5.2	IBM 3090-600J
Shin and Carmichael [31]	Parallel implementation of STEM-II on a shared-memory MIMD machine. Chemistry implemented in parallel. Concluded that vectorization produces inferior performance than concurrency.	2.5	Alliant FX/8
Saylor and Fernandes [26]	Parallel implementation of STEM-II on a shared-memory MIMD machine. Chemistry and transport implemented in parallel.	4.6	IBM 3090-600J
Abramson et al [1]	Parallel implementation of UAM on a shared-memory MIMD machine. Experiments with automatic parallelizing compilers.	6.8	Silicon Graphics SGI 4D/380 S
Dabdub and Seinfeld [8]	Parallel implementation of CIT on a distributed memory MIMD machine. Chemistry and transport implemented in parallel.	28.5	Intel Touchstone Delta
Dabdub and Seinfeld [9]	Parallel implementation of CIT on a distributed memory MIMD machine. Develop new strategies to implement transport operator in parallel.	87.7	Intel Touchstone Delta
This work	Implementation of chemistry, transport, and I/O in parallel. Develop buffering techniques to reduce communication. Develop machine independent code.	94.9	Intel Paragon

solution of operators in air pollution modeling. Shin and Carmichael [31] parallelized the chemistry operator of an AQM using shared MIMD architectures. Saylor and Fernandes [26] and Dabdub and Seinfeld [8] parallelized the chemistry and transport operators of AQMs. In the work reported here, we: consider current parallel implementation techniques of chemistry and transport computations; present techniques to parallelize input/output routines; and develop an AQM code independent of the number of nodes available, modular, and portable across distributed MIMD architectures using different message passing libraries.

After the model is described in Section 2, we discuss in Section 3 the techniques used in the parallelization of the chemistry operator of AQMs. Emphasis is placed on the paradigms used. Section 4 studies and compares the performance of different parallel implementations of the atmospheric transport operator. We describe in Section 5 the implementation of buffering techniques used to minimize data read/write operations and the implementation of the input/output opera-

tions in parallel. Finally, Section 6 comments on portability issues to achieve machine independence within distributed memory MIMD architectures.

2. Model description

The governing equation of three-dimensional Eulerian gas-phase AQMs is the atmospheric diffusion equation [29]:

$$\frac{\partial c_i}{\partial t} + \boldsymbol{u} \cdot \nabla c_i = \nabla \cdot (\boldsymbol{K} \cdot \nabla c_i) + R_i(\boldsymbol{c}, T) + S_i(\boldsymbol{x}, t) \tag{1}$$

where $c_i(\boldsymbol{x}, t)$ are the elements of the gas-phase species concentration vector $\boldsymbol{c}$, t is time, $\boldsymbol{x} = (x, y, z)$, $\boldsymbol{u} = (u, v, w)$ is the advective flow field, $\boldsymbol{K}$ is the eddy diffusivity tensor, R_i is the chemical production of species i, T is the temperature, and S_i is the volumetric source of i from emissions (Ground-level emissions are included in the surface-level boundary conditions to Eq. (1).) The flow field $\mathbf{u} = (\mathbf{x}, t)$ is provided as input to Eq. (1), either from an interpolation and extrapolation of data or from a solution of the primitive flow equations. We do not address here the problem of meteorological modeling to generate $\mathbf{u} = (\mathbf{x}, t)$.

Atmospheric aerosols are ubiquitous and often observable by eye as dust, smoke, and haze. Particles comprising the atmospheric aerosol range in sizes from nanometers (nm) to tens of micrometers (μm) — that is, from large clusters of molecules to visible flecks of dust. Most of the smallest particles (less than ~ 1 μm) are produced by condensation, either from reactive gases in the atmosphere (e.g. sulfur dioxide) or in high temperature processes (e.g. fire). Particles larger than ~ 1 μm are usually from mechanical production (wind blown soil, sea spray, etc.). As a result of myriad production processes, atmospheric aerosol chemical composition is highly variable; both with size and spatially/temporally. The aerosol is said to be internally mixed when all particles of a given size have the same composition. When the composition is not a unique function of size the aerosol is called externally mixed. The general dynamic equation that describes the evolution of the composition and size of an internally mixed aerosol is [34]

$$\begin{aligned}
&\frac{\partial q_i(m, \boldsymbol{x}, t)}{\partial t} + (\boldsymbol{u} - V_s(m)\boldsymbol{k}) \cdot \nabla q_i \\
&\quad = H_i(m, \boldsymbol{x}, t) q(m, \boldsymbol{x}, t) - \frac{(m q_i H)}{\partial m} \\
&\qquad + \int_0^m \Gamma(m', m - m', \boldsymbol{x}, t) q_i(m', \boldsymbol{x}, t) \frac{q(m - m', \boldsymbol{x}, t)}{m - m'} \, \mathrm{d}m' \\
&\qquad - q_i(m, \boldsymbol{x}, t) \int_0^\infty \Gamma(m', m, \boldsymbol{x}, t) \frac{q(m', \boldsymbol{x}, t)}{m'} \, \mathrm{d}m' \\
&\qquad + \nabla \cdot (\boldsymbol{K}(\boldsymbol{x}, t) \nabla q_i(m, \boldsymbol{x}, t)) \\
&\qquad + S_i(m, \boldsymbol{x}, t) + R_i(m, \boldsymbol{x}, t) + N_i(m, \boldsymbol{x}, t)
\end{aligned} \tag{2}$$

where $q(m, \boldsymbol{x}, t)$ is the total mass distribution such that $q_i(m, \boldsymbol{x}, t)dm$ is the mass concentration of species i in the mass range $[m, m + dm]$ and $\sum_i q_i = q$, m_i is the mass of species i in a particle of total mass m, H_i is the inverse of the characteristic time for particle growth from condensation or evaporation of species i, $\Gamma(m', m) = \Gamma(m, m')$ is the binary coagulation coefficient, $\boldsymbol{k}$ is the unit vector in the vertical direction, V_s is the particle gravitational settling velocity, and N_i the nucleation rate of species i in the mass range $[m, m + dm]$. In the present work, while we will confine our attention to gas-phase models, based on Eq. (1), the general considerations will apply to models that couple gas and particle phases. (Coupling of Eq. (2) to Eq. (1) occurs through H_i, R_i, and N_i.)

The direct numerical solution of Eq. (1) using fully implicit methods leads to systems of linear equations that are prohibitively expensive to solve on most sequential machines. Therefore, AQMs use splitting methods, alternating direction implicit (ADI) techniques, and locally one-dimensional (LOD) methods to solve Eq. (1). Splitting, ADI, and LOD methods reduce significantly the computational requirements of the models, however they introduce an inherent sequential dependence in the solution of Eq. (1). When using any of techniques mentioned above, the numerical solution of a particular operator (or dimension) must be solved in sequence. An AQM tailored for parallel computation entails a restructure of the overall numerical techniques, a redesign of the algorithms, and a reorganization of the database containing input data.

This work focuses on the parallel implementation of one particular urban-scale AQM, the CIT model [13,18]. Other typical urban- and regional-scale AQMs have a structure similar to that of the CIT model; thus there is no loss of generality in considering this model as a test case. Performance data reported here are for the same conditions as those reported in [13] for simulating the air pollution episode on August 27, 1987 in the South Coast Air Basin of California.

The model uses operator splitting techniques to solve Eq. (1):

$$\boldsymbol{c}^{t+2\Delta t} = \mathscr{L}_x^{\Delta t} \mathscr{L}_y^{\Delta t} \mathscr{L}_{cz}^{2\Delta t} \mathscr{L}_y^{\Delta t} \mathscr{L}_x^{\Delta t} \boldsymbol{c}^t \tag{3}$$

where $\mathscr{L}_x$ and $\mathscr{L}_y$ are the advection-diffusion operators in the x and y coordinates, and $\mathscr{L}_{cz}$ is the chemical kinetics and vertical transport operator. Chemical kinetics and vertical transport are coupled into a single operator since these phenomena occur on comparable time scales. Horizontal advection-diffusion is solved using a Galerkin finite-element method [17,19]. The chemistry operator is solved using a hybrid predictor-multi-corrector algorithm [36]. Tables 3 and 4 summarize numerical techniques used in current AQMs to solve the chemistry and advection operators. Fig. 1 shows the data flow of the sequential model.

Fig. 2 shows the computational region used to simulate the South Coast Air Basin of California. The grid contains 994 columns; each column consists of 5 grid cells in the vertical direction. Each grid cell contains the concentrations of 35 chemical species together with meteorological data (temperature, relative humidity, solar radiation, and advective flow field). The chemical reaction mechanism includes 106 reactions among the 35 species. These numbers are quite representative of other urban-scale AQMs.

Table 3
ODE integration methods for chemical kinetics of air quality models

Method	Reference	Characteristics
LSODE	Hindmarsh [15]	General stiff and nonstiff solver. Uses Adams methods (predictor-corrector) in the nonstiff case, and backward differentiation methods for the stiff case. Treats Jacobian matrix as a full or banded matrix. The linear systems that arise are solved by direct methods (LU factor/solve).
VODPK	Brown and Hindmarsh [3]	General stiff and nonstiff solver. Uses Adams methods in the nonstiff case, and backward differentiation methods for the stiff case. Uses Krylov subspace iterative methods with right, left or no preconditioners and scaling.
QSSA	Hesstvedt, et al. [14]	Stiff solver specifically designed for chemical kinetics problems. Uses analytical characteristics of rate equations for integration. Fixed time step. Some stability problems in long time simulations.
Hybrid	Young and Boris [36]	Stiff solver specifically designed for chemical kinetics problems. Uses asymptotic approximations to rate equations for integration. Variable time step. Predictor-multi-corrector algorithm.
Extrapolation	Dabdub and Seinfeld [10]	A general solver that uses Richardson extrapolation to improve accuracy over single time step solution. To solve chemical kinetics problems the algorithm proposed in this work uses a modified QSSA scheme.

Table 4
Numerical advection methods used in air quality methods

Method	Characteristics	Reference
SMOL Smolarkiewicz method	Iterative scheme based on upstream differences. Positive definite.	Smolarkiewicz [32]
NMOL Numerical method of lines	Fourth-order directional difference in space. SDRIV2 integration in time. Produces negative conc.	Schiesser [28] Carver and Hinds [6] Kahaner et al. [16]
BOTT Bott method	Nonlinear renormalization of advective fluxes. Positive definite	Bott [2]
EMDE Emde method	Continuous curvature cubic-spline. Positive definite.	Emde [11]
GLRK Galerkin method	Chapeau function Galerkin finite element. Produces negative conc.	McRae et al. [19]
ASD Accurate Space Derivative method	Fourier techniques to accurately compute space derivatives. Third-order Taylor expansion to integrate in time. Produces negative conc.	Gazdag [12]

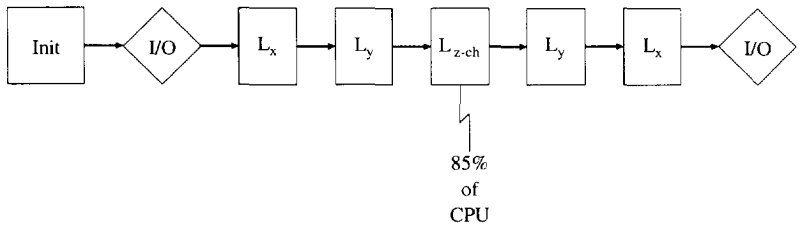


Fig. 1. Schematic data flow of one time step of sequential implementation of the CIT model.

We will proceed to study the parallelization of the model based on numerical techniques used in the sequential code. The task of designing an *ab initio* tailored numerical scheme to solve the atmospheric diffusion equation on a parallel architecture is left to further research. The initial challenge is the parallelization of the chemistry and transport operators, the core of AQMs. Domain decomposition strategy and coarseness of the parallelization are identified in this step. The second phase of the design of the parallel code is to optimize parallel I/O operations. This aspect has been neglected in previous development of parallel AQMs.

3. Parallelization of the chemistry operator

Approximately 85% of the CPU time in a typical urban or regional air pollution simulation is spent in the solution of the chemistry operator. Since the data required to solve the chemistry operator are local for each grid column, this suggests a starting point towards the parallelization of the model using a single producer/many consumers paradigm. The producer's (host's) task is to read and

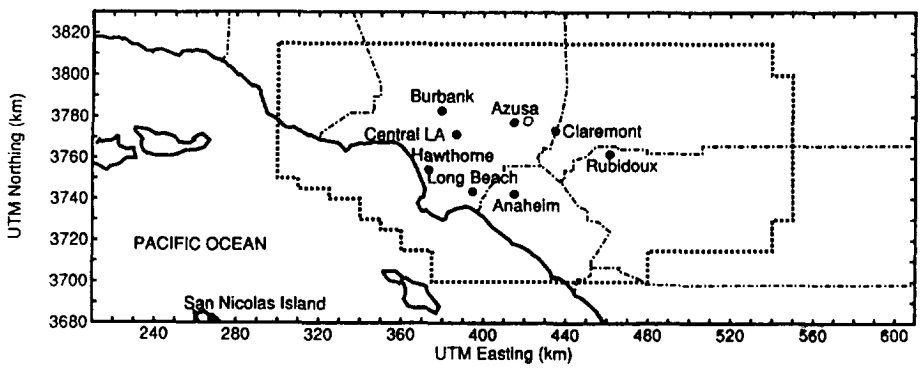


Fig. 2. Modeling region used in simulation of the South Coast Air Basin. The rectangle represents the x-y projection of the master data storage grid of the model. Area enclosed by the dotted line represents the actual computational domain.

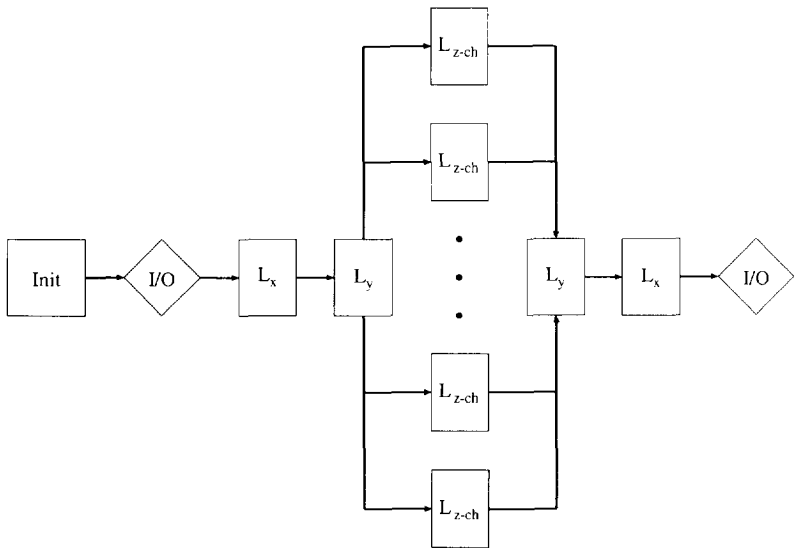


Fig. 3. Schematic data flow of one time step of the CIT model implementing a producer/consumer paradigm to solve the chemistry operator in parallel.

pack the data required to solve the chemistry operator, and to perform the horizontal transport computations. The producer also receives the processed data and produces formatted output. Each consumer's (slave's) task is to receive that data, perform the time step integration, and send them back to the producer. Fig. 3 shows a data flow of this type of parallelism. Note that this approach does not require any communication among consumers.

Parallelization of the chemistry presents the following question: What is the optimal distribution of columns among the nodes? If we assume that the time required to process each column is constant for each time step, the answer is simple; each node takes approximately the same number of columns to process. In each time step a particular consumer node takes the same section of a predetermined domain decomposition. Under this assumption, the performance of the parallel code is independent of the geometry of decomposition implemented.

The assumption of constant time per column must be discussed further. The processing of each column consists of the integration of a system of coupled, stiff, nonlinear ordinary differential equations representing the chemistry. Processing time may differ from column to column depending on the degree of stiffness of the equations in different areas of the spatial domain. The columns in which a higher degree of stiffness exists require smaller time steps to advance the solution and thus greater overall computation time. If the distribution is one column per consumer, then the load will be slightly off-balanced. In such a situation, the time to solve the chemistry operator each time step is determined by the processing time of the stiffest column. Therefore, one expects the speed-up curves to flatten at the high number of processors regime. When using a low to medium number of

processors (compared with the number of columns), the load balance is significantly improved, because of reduced differences in processing time among the consumer nodes.

Two types of variables arise naturally when implementing the vertical transport and chemistry in parallel: constant (Those variables that do not change over the course of the simulation, e.g. surface roughness) and variable data (e.g. temperature and relative humidity). Constant data are broadcast to all nodes at the end of the initialization stage. If a predetermined domain decomposition is used, some of the constant data are decomposed before being broadcast. Variable data are stored in a buffer so that, to minimize communication overhead, only one message is passed from producer to consumer.

A predetermined domain decomposition was used by Dabdub and Seinfeld [8] on the CIT model and a speed-up of 13.9 was reported when performing only the chemistry in parallel. The use of a predetermined decomposition has two main advantages. First, it simplifies the debugging process; the programmer knows which columns are being sent to which nodes. Second, it reduces the communication of sending initial data that can be decomposed. For example, if a determined decomposition is used the entire topographical data need not be sent to all the nodes. Instead, only the section of the predetermined domain is sent from the host.

Fig. 9, to be shown later, presents the measured times for a standard 24-hour simulation of the California South Coast Air Basin using the CIT model after parallelizing the chemistry operator only. The simulation was performed on the Intel Touchstone Delta using NX as the message passing library. The speed-up curve is monotonically decreasing. However, the rate of change of the curve diminishes (almost flattens) in the high number of processors regime.

Parallel implementation of the chemistry, as described above, is a straight-forward approach, requiring only introductory parallel programming techniques. Furthermore, because of its simple data flow structure, a predetermined decomposition avoids the most common pitfalls of MIMD programming: algorithmic and cyclic deadlocks as well as process starvation. One of the major problems of the single producer/many consumers paradigm is that the producer might not be distributing the data sufficiently fast for all the consumers. This effect explains the flattening of the chemistry curve.

4. Parallelization of the horizontal advection operator

The advection equation, one of the component operators in the splitting scheme, accounts for the horizontal transport of pollutants under a given wind field. In the sequential mode the advection solution requires only about 5% of the CPU time when a Galerkin method is used. Whereas this is a small percentage of the overall sequential time, it represents a significant percentage of the CPU time once the chemistry has been parallelized. Therefore, parallelizing the transport operator promises higher speed-up than that obtained by parallelizing chemistry

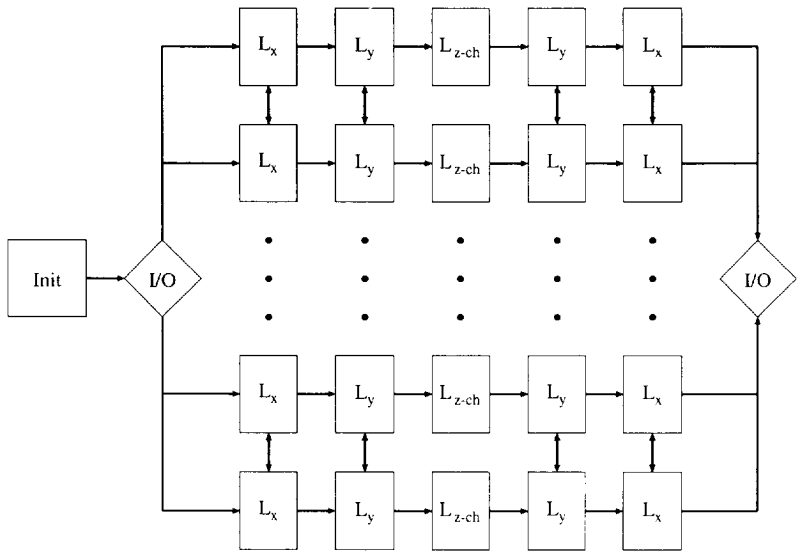


Fig. 4. Schematic data flow of one time step of the CIT model implementing a foreman/worker paradigm to solve the transport operator in parallel.

alone. The basic idea is to perform the computations involved in the solution of the advection equation of all rows (or columns) simultaneously. Since the data required to advance the solution are distributed among different nodes, there is an inherent need for extensive interprocessor communication. Thus, parallelizing the transport operator is more challenging than the chemistry operator. Furthermore, different advection solvers require different amounts of data. To maintain the modular design of the code we send all possible data that might be required to the node performing the transport computations. The extra communication costs provide freedom to use any advection solver and an infrastructure to solve the transport operator that is independent of the domain decomposition used by the solution of the chemistry operator.

The general approach for implementing the transport is to use the foreman-worker paradigm (see Fig. 4). The foreman (host node) divides the computational work among the workers (slave nodes). The workers are arranged in such an order that they can communicate with the foreman and with any other worker. Given this configuration, some questions need to be answered: How will the work be distributed? After each transport process has modified its data set, it must return the new concentrations to its node of origin. Then: Which node is the originating one? Is the originating node ready to receive the modified data? From the originating node perspective, after the data have been modified in different places, how does it know which data are coming from which node? Could there be a situation where two workers are attempting to send data to each other? Does the communication strategy guarantee a one-to-one correspondence between the sends

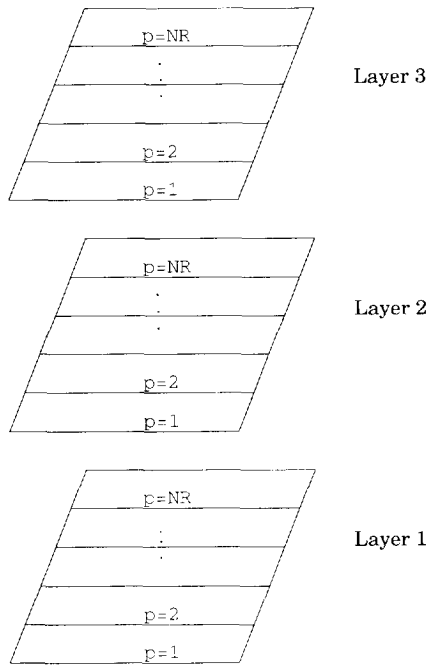


Fig. 5. Example of row distribution to solve the advection-diffusion operator in the x-direction using a DTN approach. Each row contains the processor number where the advection computations for such row are to be performed. The maximum number of processors used in the solution of the operator is number of rows (NR) of the domain.

and receives? Finally, will the communication strategy still be valid if the numerical solution to the transport operator is changed?

Different answers to these questions represent different strategies. Dabdub and Seinfeld [9] selected one of the extreme nodes of each row (or column) as the worker that will process the transport of all species of the column for all layers. A buffer is created for each species and layer containing the required data to solve the transport operator. The buffer is appended information describing the origin and destination worker of each species. To avoid deadlock situations, algorithm synchronization is implemented as the first step of the transport operator. This approach is known as Designated Transport Node (DTN) [9]. Fig. 5 shows the data distribution used by DTN when solving the advection-diffusion operator in the x-direction. Using DTN a speed-up of about 30 with chemistry in parallel is achieved as will be presented subsequently in Fig. 9. Results show that in the low number of processors regime, the overhead in message communication used in the transport implementation overshadows the reduced amount of work performed by the host. The DTN approach has the advantage that the underlying message passing structure is still valid if the algorithm to solve the advection equation is modified. On the other hand, one problem with DTN is that when the number of

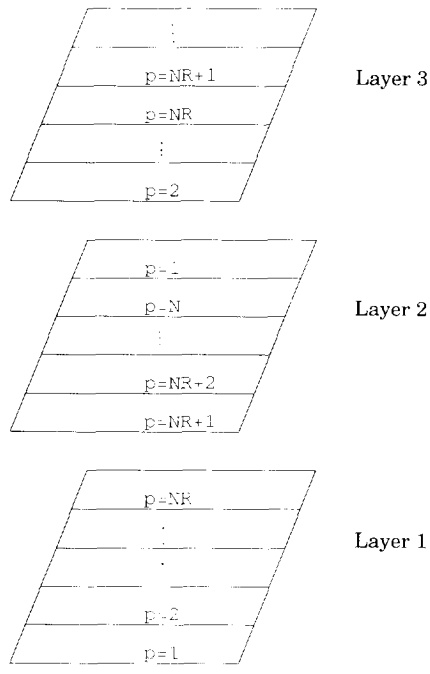


Fig. 6. Example of row distribution to solve the advection-diffusion operator in the x-direction using a DYB approach. Each row contains the processor number where the advection computations for such row are to be performed. All N processors are involved in the advection-diffusion computations by decoupling the vertical layers to achieve higher parallelism. The number of rows (NR) of the domain is not a limiting factor on the number of processors used.

workers is greater than the number of rows (or columns) the overall efficiency is decreased. In such a situation the extra workers will be idle while the transport computations are being carried out.

To overcome the idle time of the extra workers when the number of workers is greater than the number of rows, a dynamical decomposition approach is implemented. The Dynamical Balanced (DYB) [9] approach consists of distributing the first N rows of the lower layer to the first N workers. N is the number of available workers. That is, instead of processing all the species for all the layers in a single worker, the different layers are distributed among all other nodes. The DYB approach uses the same foreman-worker paradigm illustrated in Fig. 4, but with a higher number of workers solving the $\mathscr{L}_x$ and $\mathscr{L}_y$ operators. Fig. 6 shows the data distribution used by DYB when solving the advection-diffusion operator in the x-direction. The DYB approach requires a more elaborate use of indexing and pointers in the code. In addition, since all the nodes perform transport computations, the required transport data need to be sent to all the nodes. As long as the time used by the extra communication is smaller than the time saved by the extra workers, the DYB approach is worth implementing. As it will be shown subse-

quently in Fig. 9, there is no advantage of using the DYB approach when a fast transport solver such as the Galerkin method is used. On the other hand, when using the more computationally intensive Accurate Space Derivative method (ASD), the advantages of using the DYB strategy are clear, as will be shown in Fig. 10. The ASD technique, a highly accurate algorithm to solve the advection operator [12], imposes such a high computational demand that it is almost prohibitively expensive to use on sequential computers. The fact that ASD has a high computational requirement makes its computation to communication ratio greater than other techniques. As a result, it is more attractive for use in a parallel environment. Parallel computers not only permit the use of a highly demanding algorithm, but also encourage its use.

5. Parallelization of the I/O

Up to this point we have discussed parallelization of the chemistry and the advection operators of AQMs. Can anything more be done at this point to further increase the speed-up of the model? It is observed from Fig. 4 that all nodes must wait idly while the host is reading and writing data to the file system. The I/O is not a major fraction of the total time of the code when executed sequentially (about 5%). However, at this point when the two main operators of the model have been parallelized, the I/O becomes a significant bottleneck that affects all the worker nodes. There are several approaches to overcome the I/O bottleneck that are discussed in this section.

The most alluring strategy seems to eliminate the I/O from the host node. In that case, the bottleneck would completely disappear. Furthermore, there would not be any messages that need to be sent. In this strategy all workers perform their own I/O. Despite its promising potential, this idea turns out to be disastrous. When there is a high number of computational nodes trying to access the file system, the performance of the I/O operations is hindered by the operating system overhead. Worse yet, when the number of nodes is sufficiently large, the entire file system collapses. Another strategy must be considered.

When all the workers are performing chemistry and transport computations the host is waiting idly for the results to arrive. Once the results arrive they are organized, formatted, and written to the output files. The use of this idle time is the basis of the next step to increase the performance of the code. To accomplish that, a first-in-first-out (FIFO) queue is created. When the host reads the data initially it sends them to the worker nodes. Instead of waiting for the workers to send the processed data back, the host proceeds immediately to read the input data for the next time step. The host decomposes and places the newly read data in the input queue while the workers are still performing the chemistry and transport computations. The use of the queue consumes the same amount of message passing time but reduces the reading time. The overhead of keeping track of the queue is small compared to that saved during the reading process. The data flow is illustrated in Fig. 7.

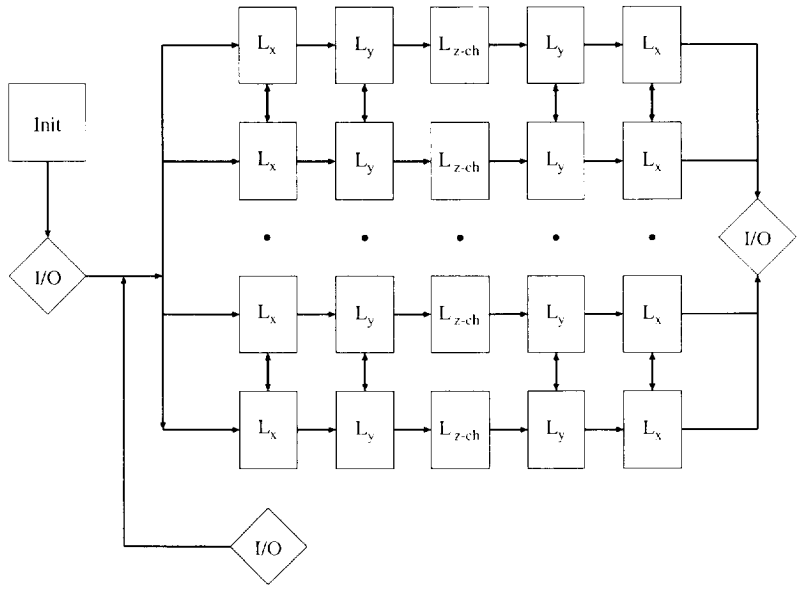


Fig. 7. Schematic data flow of one time step of the CIT model implementing a buffering queue for input/output operations.

Performance measurements show that in the high number of processors regime the host is still a bottleneck, even after eliminating its idle time. The workers are still processing the data faster than the foreman is able to read and distribute them. The next step taken is to eliminate one of the workers and to create a second host. That is, the paradigm now used is that of two foremen and many workers, in which each of the foremen performs approximately half the task of the original foreman. The new data flow is illustrated in Fig. 8. The two foremen read different files simultaneously. Two processes reading data do not present any problems in the file system performance. Another benefit of using two foremen is that the message passing time is reduced by almost half. Therefore, the parallelization of the I/O described here focuses on distributing file access operations between two hosts. In addition, the Intel Paragon I/O sub-system is automatically distributing or striping the data across the disks.

Fig. 9 shows that the performance of the model significantly increases after the I/O has been implemented using the queue and two foremen technique. Fig. 10 also shows a dramatic improvement when using the more computationally intensive ASD algorithm to solve the transport operator. The fastest run measured using the Galerkin solver is on the Intel Paragon with 128 nodes. A 24-hour simulation consumes less than 7 minutes which corresponds to a speed-up of about 45. The fastest run measured using the ASD solver is also on the Intel Paragon with 128 nodes. A 24-hour simulation requires 11.5 minutes, which corresponds to a speed-up factor of 94.

When the model is highly parallelized there is a performance decrease when

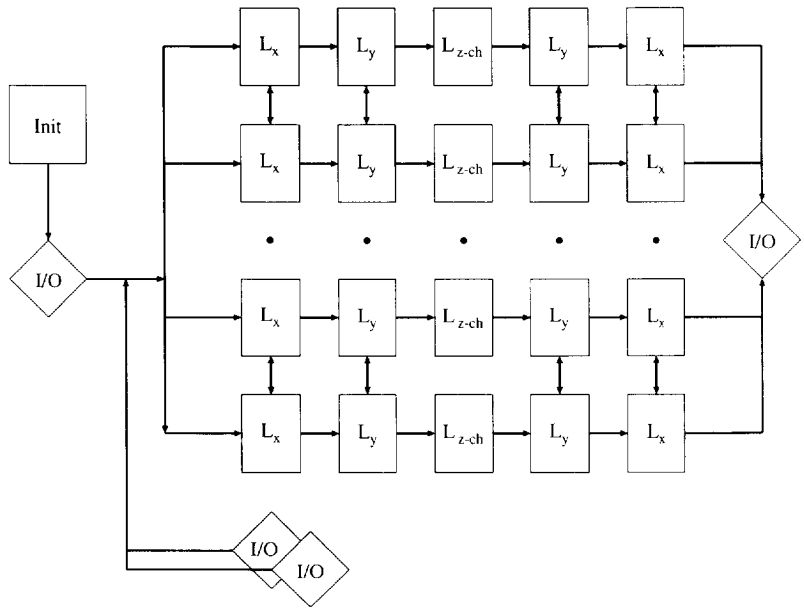


Fig. 8. Schematic data flow of one time step of the CIT model implementing parallel buffering queue for input/output operations.

adding extra processors in the high number of processors regime. This effect occurs because the computation to communication ratio inherent in the model is not sufficiently high. Stated in a different manner, the task of simulating this Los

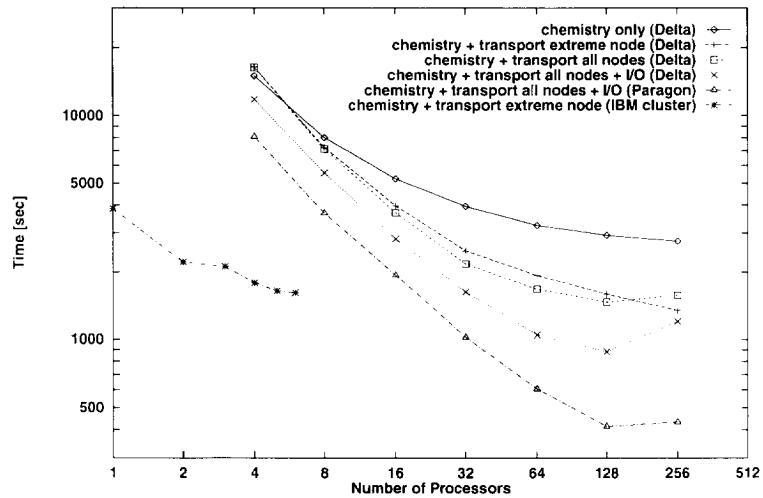


Fig. 9. CPU time for a 24-hour simulation of the South Coast Air Basin versus number of nodes for the Galerkin transport implementation of the CIT using different parallelization strategies.

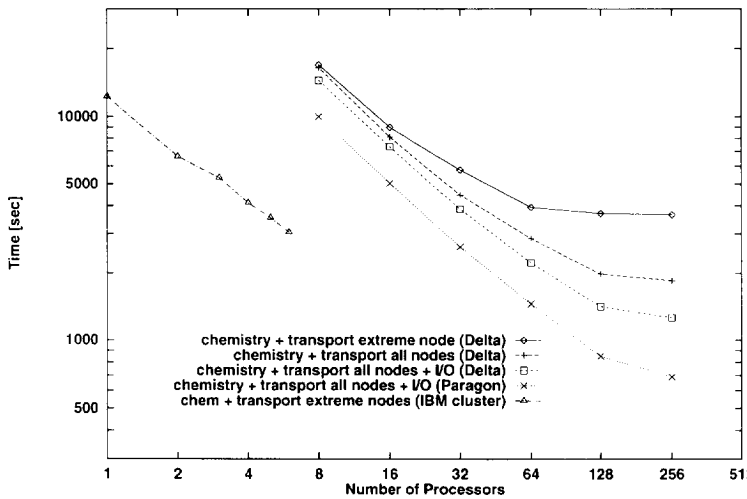


Fig. 10. CPU time for a 24-hour simulation of the South Coast Air Basin versus number of nodes for the ASD transport implementation of the CIT model using different parallelization strategies.

Angeles Basin gas-phase air pollution episode is sufficiently large to run optimally on 128 nodes but not on 256 or more processors. Moreover, it can be seen that the speed-up of the model using the ASD method is higher than that obtained using the Galerkin method since the former has a higher computation to communication ratio.

6. Key issues on portability

The use of the NX message passing library provides various parallel computers commercially available as platforms to run the model: the Touchstone Delta, the Intel Paragon, the Gamma machine, the IPSC and the IPSC2. However, a few workstations connected in a network are more likely to be accessible than such machines. Moreover, there is a wide variety of distributed MIMD type architecture that do not support the NX library. It is crucial for the wide assessibiltity of parallel air quality models that they be portable across different computing platforms.

With that motivation, we first implemented a parallel version of the model using P4 as the message passing library [4]. The cluster of workstations is considered a MIMD distributed memory concurrent computer. Under that perspective the porting of the NX model to the P4 environment consists of modifying the calls and syntax of the communication library. A script that automates such conversion was designed. Processor efficiencies of 50–80% are obtained when running the model with the Galerkin advection solver on a cluster of workstations. When the ASD solver is used processor efficiencies reach 80–90% because of the higher computation to communication ratio inherent to ASD. The cluster is made up of IBM

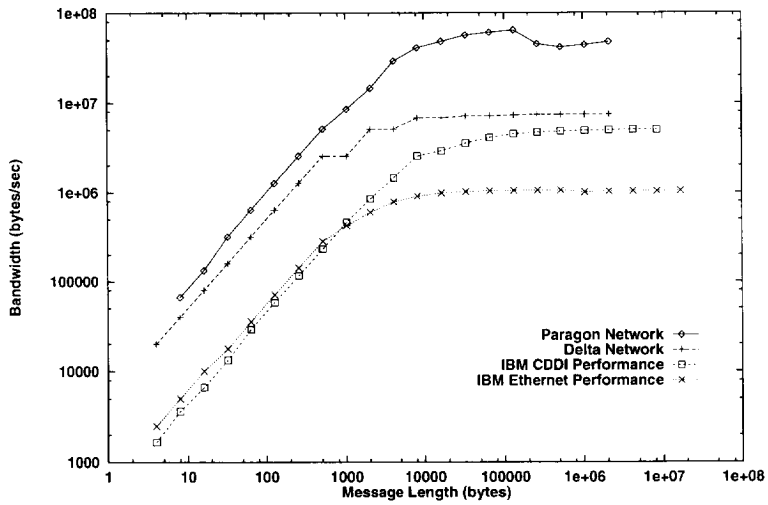


Fig. 11. Message passing performance of various networks used. Bandwidth measurements correspond to rendez-vous sends.

RISC 390 nodes and a IBM RISC 580 host using a CDDI network. The scalability of the model with the Galerkin transport solver on the network of workstations is hindered by several factors. First, the Intel Delta and Paragon machines have significantly higher network bandwidth as shown in Fig. 11. In addition, the performance of CDDI and Ethernet networks degrades linearly with the number of processors. Third, the performance of the compute nodes of the network of workstations (RS/6000-390 processors) is significantly greater than those of the Delta and Paragon machine (Intel i860 processors). Fig. 10 shows that increasing the computation per node, by using ASD, improves the scalability of the network of workstations.

Regardless of the availability of the script, it is still cumbersome to manage software development when different codes for different environments are available. This need has been the driving force of MPI, a proposed standard message passing interface [33]. MPI would permit the point-to-point and global routines under different communication contexts and network topologies. At the time we started developing the parallel code a full implementation of MPI was not available. To overcome the portability of the code we designed our own message passing interface and incorporated it into the parallel model. The routines are designed to run on top of P4 and NX. Other libraries such as PVM or EXPRESS can be easily incorporated. The use of such routines produces a "universal" code that runs on different architectures under different basis communication libraries.

7. Conclusions

This paper describes the strategies used to implement a typical urban- or regional-scale air quality model in a parallel environment. Parallel implementation

of the chemistry operator, transport operator, and I/O routines are required to obtain the highest speed-ups. Using a Galerkin horizontal transport solver a typical model run requires less than 7 minutes on the Intel Paragon with 128 nodes. This corresponds to a speed-up of 45. Using the more computationally intensive ASD transport solver, the same run requires less than 11.5 minutes, which corresponds to a speed-up of 95. The speed-up numbers reported here represent the parallel implementation of a model that was designed to run on a sequential machine. Parallel computing motivates research to find new solution strategies to the atmospheric diffusion equation (with nonlinear chemistry) not envisioned when only uniprocessors are used.

Parallel processing gives the computational power necessary to increase the level of physical and chemical detail in the models. For instance, the incorporation of the aerosol and aqueous phases are now receiving attention by the research community. Aerosol particles play a significant role in human health and urban visibility. The incorporation of the aqueous phase permits the study of acid deposition and acidic fogs. In either case, the addition of new physics will substantially increase the computational task required by the new models. From the data flow perspective, both of these phenomena would require a producer/consumers paradigm similar to that used here to parallelize the chemistry. Therefore, some of the computational infrastructure to incorporate new physico-chemical phenomena in parallel has been laid out here. Adding new phenomena to the models increases their computation to communication ratio. Consequently, higher speed-ups are to be expected. A higher degree of complexity in atmospheric chemical dynamic models than currently exists is well suited for the 500 + number of processors regime.

Acknowledgments

This work was supported in part by a grant from the IBM Environmental Research Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the IBM Corporation. The information in this document has been funded wholly or in part by the United States Environmental Protection Agency High Performance Computing and Communication Program under cooperative agreement CR 822051-01-0. This research was performed in part using the Intel Delta System and the Intel Paragon System operated by Caltech on behalf of the Concurrent Supercomputing Consortium.

References

[1] D. Abramson, M. Cope and R. McKenzie, Modelling photochemical pollution using parallel distributed computing platforms, in: C. Halatsis, D. Maritsas, G. Philokyprou and S. Theodoridis eds., *Lecture Notes in Computer Science 817* (Springer, New York, 1994)

[2] A. Bott, A positive definite advection scheme obtained by nonlinear renormalization of the advective fluxes, *Mon. Wea. Rev.* 117 (1989) 1006–1015.

[3] P.N. Brown and A.C. Hindmarsh, Reduced storage matrix methods in stiff ODE systems, *J. Appl. Math. Comput.* 31 (1989) 40–91.
[4] R.M. Butler and E.L. Lusk, Monitors, messages, and clusters — the p4 parallel programming system, *Parallel Computing* 20 (1994) 547–564.
[5] G.R. Carmichael, L.K. Peters and T. Kitada, A second generation model for regional-scale transport/chemistry/deposition, *Atmospheric Environment* 20 (1986) 173–188.
[6] M.B. Carver and H.W. Hinds, The method of lines and the advection equation, *Simulation* 31 (1978) 59–69.
[7] J.S. Chang, R.A. Brost, I.S.A. Isaksen, S. Madronich, P. Middleton, W.R. Stockwell and C.J. Walcek, A three-dimensional eulerian acid deposition model: physical concepts and formulation, *J. Geophys. Res.* 92 (1987) 14681–14700.
[8] D. Dabdub and J.H. Seinfeld, Air quality modeling on Massively Parallel Computers, *Atmospheric Environment*, 28 (1994) 1679–1687.
[9] D. Dabdub and J.H. Seinfeld, Numerical advective schemes used in air quality models-sequential and parallel implementation, *Atmospheric Environment* 28 (1994) 3369–3385.
[10] D. Dabdub and J.H. Seinfeld, Extrapolation techniques used in the solution of stiff ODEs associated with chemical kinetics of air quality models, *Atmospheric Environment* 29 (1995) 403–410.
[11] D.V.D. Emde, Solving conservation laws with parabolic and cubic splines, *Mon. Wea. Rev.* 120 (1992) 82–92.
[12] J. Gazdag, Numerical convective schemes based on accurate computation of space derivatives, *J. Comp. Phys.* 13 (1973) 100–113.
[13] R.A. Harley, A.G. Russell, G.J. McRae, G.R. Cass and J.H. Seinfeld, Photochemical modeling of the Southern California air quality study, *Environ. Sci. Technol.* 27 (1993) 378–388.
[14] E. Hesstvedt, O. Hov, and I.S.A. Isaksen, Quasi-steady-state approximations in air pollution modeling: comparison of two numerical schemes for oxidant prediction, *Int. J. Chem. Kinetics* 10 (1978) 971–994.
[15] A.C. Hindmarsh, LSODE and LSODI, two new initial value ordinary differential equation solvers, *ACM-SIGNUM* 15–4 (1980) 10–11.
[16] D. Kahaner, C. Moler and S. Nash, *Numerical Methods and Software* (Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1989)
[17] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method* (Cambridge University Press, England, 1987)
[18] G.J. McRae, W.R. Goodin and J.H. Seinfeld, Development of a second generation mathematical model for urban air pollution-I. Model formulation, *Atmospheric Environment* 16 (1982) 679–696.
[19] G.J. McRae, W.R. Goodin and J.H. Seinfeld, Numerical solution of the atmospheric diffusion equation for chemically reactive flows, *J. Comp. Phys.* 45 (1982) 1–42.
[20] R.E. Morris and T.C. Myers, User's guide for the Urban Airshed Model — Volume I: User's manual for UAM (CB-IV), U.S. Environmental Protection Agency (1990) (EPA-450/4-90-007a).
[21] National Research Council, *Rethinking the Ozone Problem in Urban and Regional Air Pollution* (National Academy Press, Washington, D.C., 1991).
[22] M.T. Odman, N. Kumar and A.G. Russell, A comparison of fast chemical kinetic solvers for air-quality modeling, *Atmospheric Environment* 26 (1992) 1783–1789.
[23] P. Pai and T.T.H. Tsang, Parallel computations of turbulent-diffusion in convective boundary-layers on shared-memory machines *Atmospheric Environment* 26A (1992) 2425–2435.
[24] P. Pai and T.T.H. Tsang, On parallelization of time-dependent 3-dimensional transport-equations in *air*-pollution modeling, *Atmospheric Environment* 27 (1993) 2009–2015.
[25] C. Pilinis and J.H. Seinfeld, Development and evaluation of an eulerian photochemical gas-aerosol model, *Atmospheric Environment* 22 (1988) 1985–2001.
[26] R.D. Saylor and R.I. Fernandes, On the parallelization of a comprehensive regional-scale air quality model, *Atmospheric Environment* 27A (1993) 625–631.
[27] K.L. Schere and P.K. Wayland, EPA regional oxidant model (ROM 2.0): Evaluation on 1980 NEROS databases (U.S. Environmental Protection Agency, Research Triangle Park, N.C., 1991).
[28] W.E. Schiesser, *The Numerical Method of Lines* (Academic Press, San Diego, CA, 1991).

[29] J.H. Seinfeld, *Atmospheric Chemistry and Physics of Air Pollution*. (Wiley, New York, 1986).
[30] J.H. Seinfeld, Urban air pollution: state of the science, *Science* 243 (1989) 745–752.
[31] W.C. Shin and G.R. Carmichael, Comprehensive air pollution modeling on a multiprocessor system, *Computers Chem. Engng.* 16 (1992) 805–815.
[32] P.K. Smolarkiewicz, A simple positive definite advection scheme with small implicit diffusion, *Mon. Wea. Rev.* 111 (1984) 479–486.
[33] D.W. Walker, The design of a standard message-passing interface for distributed-memory concurrent computers, *Parallel Computing* 20 (1994) 657–673.
[34] A.S. Wexler, F.W. Lurmann and J.H. Seinfeld, Modelling urban and regional aerosol — I. Model development, *Atmospheric Environment* 28 (1994) 531–546.
[35] A. Venkatram, P.K. Karamchandani and P.K. Misra, Testing a comprehensive acid deposition model, *Atmospheric Environment* 22 (1988) 737–747.
[36] T.R. Young and J.P. Boris, A numerical technique for solving stiff ordinary differential equations associated with the chemical kinetics of reactive-flow problems, *J. Phys. Chem.* 81 (1977) 2424–2427.