Pergamon

Atmospheric Environment Vol. 28, No. 9, pp. 1679–1687, 1994

1352-2310/94 $7.00+0.00

AIR QUALITY MODELING ON MASSIVELY PARALLEL COMPUTERS

DONALD DABDUB and JOHN H. SEINFELD
Department of Chemical Engineering, California Institute of Technology, Pasadena, CA 91125, U.S.A.

(*First received* 26 *March* 1993 *and in final form* 13 *October* 1993)

Abstract—The use of massively parallel computers provides an avenue to overcome the computational requirements of air quality modeling. General considerations on parallel implementation of air quality models are outlined including domain decomposition. The implementation of the CIT urban photochemical model on the Intel Touchstone Delta, a distributed memory multiple instruction/multiple data (MIMD) machine is described. When both the transport and chemistry portions of the model are parallelized, a speed-up of about 30 is achieved using 256 processors.

Key word index: Air quality modeling, parallel computers.

1. INTRODUCTION

Air quality models are essential tools in the understanding of pollutant dynamics in the atmosphere. In recent years, our understanding of the scientific foundations of the chemical and physical phenomena occurring in the atmosphere has continued to expand. We are able to construct comprehensive models that describe the dynamics of air pollution. The inherent complexity and nonlinearity of the governing equations has made air quality modeling a computational "Grand Challenge" (Levin, 1989). Currently, air quality modeling is most often performed on sequential computers.

Computational constraints have always been a limiting factor in the amount of physics and chemistry one can include in air quality models (AQMs). For example, particulate formation processes are not currently incorporated in most models due to the significant time demands of the aerosol phase computations (see, for example, Pilinis and Seinfeld, 1988). Phenomena occurring in the sub-grid scale are also ignored or crudely represented by current AQMs. The use of parallel computers provides an avenue to overcome the computational requirements of air quality modeling.

The work reported here has as a major goal to lay the foundation to implement air quality models on parallel computers. To accomplish this goal it is necessary to study and compare different approaches to distribute the computational work among the available nodes. It is necessary to test, compare, and evaluate current numerical schemes employed in the solution of AQMs. Implications of parallel computation on restructuring of the input and output data must also be examined.

2. COMPUTATIONAL BREAKDOWN OF CURRENT AQMS

The governing equation of three-dimensional Eulerian AQMs is the atmospheric diffusion equation

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

where c_i are the elements of the concentration vector $\mathbf{c}$, t is time, $\mathbf{x}=(x,y,z)$, $\mathbf{u}=(u,v,w)$ is the advective flow field, $\mathbf{K}$ is the eddy diffusivity tensor, R_i is the chemical production of species i, T is the temperature, and S_i is the source rate of i.

The different chemical and physical processes that contain inherently wide variations in their time scales pose the major challenge in constructing numerical methods to solve equation (1). Operator splitting methods have been developed and refined for the solution of AQMs (McRae *et al.*, 1982a). Splitting methods provide a numerical approach that is both accurate and economical.

The basic idea of the splitting process is the sequential use of operators, $\mathscr{L}$, that govern the different phenomena. Horizontal transport is described by

$$\mathscr{L}_H c_i=\frac{\partial c_i}{\partial t}=-\nabla_H\cdot\mathbf{u}c_i+\nabla_H\cdot\mathbf{K}\nabla_H c_i \quad (2)$$

where H represents the (x,y) plane. Vertical transport is described by

$$\mathscr{L}_z c_i=\frac{\partial c_i}{\partial t}=-\frac{\partial}{\partial z}(wc_i)+\frac{\partial}{\partial z}\left(K_{zz}\frac{\partial c_i}{\partial z}\right). \quad (3)$$

Finally, chemistry and emissions are described by

$$\mathscr{L}_c c_i=\frac{\partial c_i}{\partial t}=R_i(\mathbf{c},T)+S_i(\mathbf{x},t). \quad (4)$$

In some AQMs $\mathscr{L}_H$, the horizontal transport operator, is decomposed into two separate operators, $\mathscr{L}_x$ and $\mathscr{L}_y$, describing the transport in the x and y directions, respectively. In addition, current AQMs often combine $\mathscr{L}_c$ and $\mathscr{L}_z$ into a single operator $\mathscr{L}_{cz}$ that performs the chemistry and vertical transport computations simultaneously.

Table 1 summarizes several existing AQMs from the point of view of their computational characteristics. UAM and CIT are urban scale air quality models. Urban models typically have a vertical domain extending up to about 2 km, as opposed to regional models that extend up to about 10 km in order to treat vertical redistribution of species above the planetary boundary layer. RADM, ADOM and STEM-II are regional acid deposition and oxidant models that treat gas- and aqueous-phase chemistry. They have been applied primarily to simulations of acid deposition in eastern North America and central Japan. ROM is a regional oxidant model designed to simulate ozone formation and transport over the eastern United States.

The solution of the atmospheric diffusion equation in the operator splitting framework is obtained from the following sequence:

$$\mathbf{c}^{t+2\Delta t} = \mathscr{L}_x^{\Delta t}\mathscr{L}_y^{\Delta t}\mathscr{L}_{cz}^{2\Delta t}\mathscr{L}_y^{\Delta t}\mathscr{L}_x^{\Delta t}\mathbf{c}^t. \quad (5)$$

The amounts of time spent computing the solutions of the different operators of the CIT model, for example, are: chemistry 85.2%, horizontal transport 5.4%, "other" 9.4%. The chemistry loop of the code contains the vertical transport computations since they are coupled in the $\mathscr{L}_{cz}$ operator. In addition, the chemistry loop also contains the deposition computations for all the species within all air columns, as well as the vertical transport routines. All these computations make up approximately 90% of the code. "Other" represents the reading and writing of data, the initialization of the model, and other minor computations. It is expected that other three-dimensional AQMs have approximately the same computational breakdown.

Most of the computer time involved in solving the core equations of urban and regional scale photochemical models is consumed by the $\mathscr{L}_{cz}$ operator. The "chemistry" part of the operator consists of solving a coupled system of stiff, nonlinear ordinary differential equations as described by equation (4). Classical methods for solving systems of stiff ordinary differential equations, such as Gear's method, are not of practical use in AQMs. The reason is due not to their accuracy, but to the small time steps required, the inversion of large matrices, and/or repeated solution of large sets of nonlinear equations. A comparison of different numerical schemes to perform the integration of the chemistry in AQMs can be found in Hov *et al.* (1989) and Odman *et al.* (1992).

The solution of the advection–diffusion equation (2) requires 5.4% of the computational time in the CIT model. When solving this equation numerically, depending on the scheme used, both the amplitudes and the phases of different Fourier components of the solution will be altered. This produces so-called numerical diffusion and dispersion. This is a classic problem in computational fluid dynamics, for which a large number of specialized numerical techniques have been developed (Rood, 1987). Many have been tested and compared to determine their applicability to AQMs (Chock, 1985, 1991; Chock and Dunker, 1983). The numerical method currently implemented in the CIT model, for example, employs a fourth-order (space) Chapeau-function based finite element scheme as part of an implicit procedure to solve the advection part of $\mathscr{L}_H$. Following the advection step, a nonlinear noise filter is used to remove most of the computational noise generated by the scheme

Table 1. Aspects of current photochemical and acid deposition models

Model	Vertical resolution	Horizontal resolution	Gas-phase chemistry	Aqueous-phase chemistry	Transport
UAM	4 layers up to 1.5 km	4 km × 4 km to 10 km × 10 km	71 Reactions 30 Species	Not treated	3D wind field
CIT McRae *et al.* (1982b) Harley *et al.* (1993)	5 layers up to 1.5 km	5 km × 5 km	71 Reactions 30 Species	Not treated	3D wind field
ROM Lamb (1982) Schere and Wayland (1991)	3 layers up to 2 km	18 km × 18 km	71 Reactions 30 Species	Not treated	3D wind field with vertical transport through cumulus clouds
RADM-II Chang *et al.* (1987)	15 layers up to 10 km	80 km × 80 km	104 Reactions 58 Species	42 equilibria, 5 rxn for SO_2 oxidation	3D wind field with vertical transport through cumulus clouds
STEM-II Carmichael *et al.* (1986)	10–14 layers up to 6 km	10 km × 10 km to 56 km × 56 km	112 Reactions 53 Species	26 equilibria, 30 rxn for SO_2 and NO_x oxidation	3D wind field with vertical transport through clouds
ADOM Venkatram *et al.* (1988)	12 layers up to 10 km	60 km × 60 km to 120 km × 120 km	~100 Reactions ~50 Species	25 Reactions 13 Species	3D wind field with vertical transport through clouds

(Forester, 1977). Finally, the diffusion step is computed with an explicit second-order finite difference scheme (McRae *et al.*, 1982a). In brief, the challenges of solving equation (2) relate largely to computational accuracy.

The analysis of where computational effort is expended shows that a parallel implementation of an AQM should start by focusing on performing the chemistry integrations simultaneously on different processors.

3. PARALLEL ARCHITECTURES

Traditionally, parallel architectures can be classified into two broad categories: SIMD and MIMD. Each has different programming approaches.

SIMD architectures, denoting single instruction/multiple data, execute the exact same instruction on different sets of data simultaneously. SIMD machines are well suited for problems that primarily require the manipulating of large matrices.

MIMD architectures, for multiple instruction/multiple data, can execute different instructions on different sets of data simultaneously. The way that the different processors communicate determines the different flavors of MIMD. In a shared-memory MIMD computer, all the processors have access to a common memory. Shared-memory MIMD machines are relatively easy to program, but they are limited by scalability and they might present cache-coherency problems. On the other hand, in a distributed-memory MIMD machine every processor contains its own local memory. Distributed-memory MIMD machines are also known as multicomputers. Intel's Touchstone Delta is one of the newest and fastest multicomputers. A network of workstations optically interconnected qualifies also as a parallel multicomputer.

It is not currently known with full certainty which architecture provides the best environment for air quality models. The issue of exploiting the different advantages of a given parallel architecture still remains to be studied. Carmichael *et al.* (1989) have studied the solution of transport/chemistry calculations on SIMD machines. They have observed that simple (4 species) chemical mechanisms are well suited for SIMD machines. However, the implementation of transport algorithms efficiently on SIMD machines leaves much to be desired.

Pai and Tsang (1992) have used different shared-memory MIMD machines to study common time-splitting finite difference or finite element schemes used in air pollution modeling. Specifically, they simulated turbulent diffusion in convective boundary layers. The highest speed-up reported is 13.11 using a Sequent Symmetry S81 machine.

Shin and Carmichael (1992) have parallelized the STEM-II air pollution model. They used a shared-memory ALLIANT FX/8. Their work focused on parallelizing the chemistry portions of the model. The speed-up reported reached 2.5. Furthermore, they mention that the efficient use of vectorization would require alternative algorithms.

We report here on the use of distributed-memory MIMD machine architectures for air quality modeling. We have implemented the CIT model on the Intel Touchstone Delta (Intel Corporation, 1991). The Delta contains 570 nodes: 528 numeric nodes, 34 mass storage nodes, 2 gateway nodes, and 6 service nodes. The nodes are interconnected as a two-dimensional mesh. A maximum configuration would provide 10 Gigabytes of distributed main memory and 200 Gigabytes of online storage.* The 528 numeric nodes are based on the 64-bit Intel i860 microprocessor. The i860 has a peak speed of 60 double-precision MFLOPS and 80 single-precision MFLOPS. Each numeric node includes 16M bytes of parity-check DRAM, a 4K byte instruction cache, and an 8K byte data cache.

The programming was done in FORTRAN 77 using NX version 1.4 as the extended communication library used for message passing. The issue of software portability requires special attention to different aspects of the programming. First, only the simple synchronous send and synchronous receive routines should be used. By avoiding the use of more sophisticated message passing routines, a degree of freedom is gained on code portability. That is, porting of the parallel code to other message passing routines such as EXPRESS, LINDA, or PVM can be achieved by replacing the low level synchronous sends and receives with the appropriate library call. Furthermore, the use of NX as the underlying message passing library makes the code executable on other parallel computers commercially available with no changes. Specifically, the code has been run successfully on the Intel GAMMA, the IPSC and IPSC2. Second, one should write general purpose send/receive routines. By doing so, these general send/receive routines can call any appropriate communication protocol dependent send/receives. Third, the code should be independent of the number of nodes available. The number of nodes available should be determined at execution time or given as a parameter so that no further code modifications are needed. Finally, the use of parallel file systems should be avoided from the point of view of portability (not performance) since they are highly architecture-dependent.

The use of MIMD machines provides a promising approach to AQM calculations. MIMD provides a friendly environment for SPMD (single process multiple data) and for MPMD (multiple process multiple data) programming methodologies. The chemistry

* Currently, three-dimensional air quality models considering only gas-phase phenomena do not impose extreme memory requirements, nor extreme storage space requirements. However, with the implementation of aqueous phase and/or aerosol phase, the memory and storage requirements would increase drastically.

computations in AQMs follow a SPMD approach, while the transport computations, as described below, inherently follow a MPMD approach. By definition, SIMD architectures are not suitable to MPMD programming. Furthermore, the usage of distributed-memory MIMD opens the possibility of porting the code to a network of workstations, which might be more readily available than a parallel supercomputer. In short, MIMD seems to be the most promising architecture to perform AQM computations.

4. KEY ISSUES IN THE PARALLEL IMPLEMENTATION OF AQMS

The first step towards parallelizing a code such as an AQM is to perform a detailed profile of the code. A code profile, as discussed earlier, shows the computational breakdown of the code, that is, the computational time spent in each routine for an entire run. In principle, the computationally intensive parts are the desired subroutines to be performed in parallel. For the case of comprehensive AQMs, as noted above, the most computationally intensive routine is the chemistry integration. The amount of time spent here ranges from 75 to 95% of the total computing time.

The next issue of concern is that of data dependency. The AQM chemistry portion does not present any difficulty on data dependency. All the data needed to compute the vector of species concentrations after the chemical changes of the corresponding time step are local. The results of the chemistry integration at a particular grid point do not depend on the concentrations at other grid points. On the other hand, transport computations, regardless of the numerical scheme used, are, in principle, dependent on data located at neighboring grid points. Communication among the nodes is therefore imperative. The implementation of node communication is dependent on the transport numerical scheme used, as well as on the decomposition of the domain implemented.

Speed-up is a common measure of performance gain from a parallel environment. It is the factor denoting how many times faster the parallel version of the code runs in comparison to the sequential version. Indeed, it is defined as the ratio of the time required to complete a job using one processor to the time required to complete the same job with N processors. The maximum speed-up is limited by the fraction of that job that is performed in parallel, p. An ideal speed-up, S, achieved with parallelization is described by Amdahl's law

$$S=\frac{1}{(1-p)+p/N}. \tag{6}$$

The speed-up is ideal in the sense that it does not account for the time taken to send/receive messages. Increasing p by a small amount at a point where a significant amount of the code is already parallelized, say for $p>0.96$, produces substantial additional speed-up.

4.1. *Domain decomposition*

The objective of domain decomposition is to distribute the computational load as evenly as possible among the available nodes. The different ways to decompose the three-dimensional spatial domain present several questions to be addressed for an effective parallel AQM implementation. There are two main approaches to this matter.

The first approach to decompose the domain is called dynamical domain decomposition. One starts with a number of tasks to be performed. A so-called "master" node is in charge of sending one of those tasks to a node that is idle at the moment. The idle node receives the message and some appropriate data needed to perform the task. The appropriate data might come from the master node and/or from other nodes that should be notified of such requests. After the task is completed, the node sends the results to the master node, or to other nodes, and/or keeps it stored in its own memory. The node also signals the master that it has finished the task. The process is continued in this way until all the desired tasks are performed.

The other approach to distribute the computational load is to set a predetermined decomposition. In this case, the master sends a specific number of tasks to each node. The tasks are distributed at once among the available nodes. The nodes, when having completed their collection of tasks, signal the master that they have done so. The advantage of this technique arises when the programmer is able to set an approximately evenly distributed set of tasks. The technique also facilitates debugging of the code, since the programmer knows what task is in what node. Furthermore, there is a decrease in overhead work since the communication between the nodes and the master node is minimized.

Another issue to be considered while determining the domain decomposition to be used is the dimensionality of the decomposition. One has to decide on performing either a one-dimensional or a two-dimensional decomposition. Fox *et al.* (1988) discuss the implications of the dimensionality of the decomposition. In the case of AQMs the chemistry computations are independent of the dimensionality of the decomposition. However, the dimensionality of the decomposition will likely affect the transport computations.

Figure 1 shows the different domain decompositions used as a function of the number of slave nodes available for the simulation of the South Coast Air Basin using the CIT model. The rectangle represents the x–y projection of the master data storage grid of the model. The grid is currently subdivided into 80 cells in the x-direction, 30 cells in the y-direction, and 5 layers. An individual cell in the figure represents a 5×5 km^2 area. The shaded section in the 1-node case of Fig. 1 represents the computational region

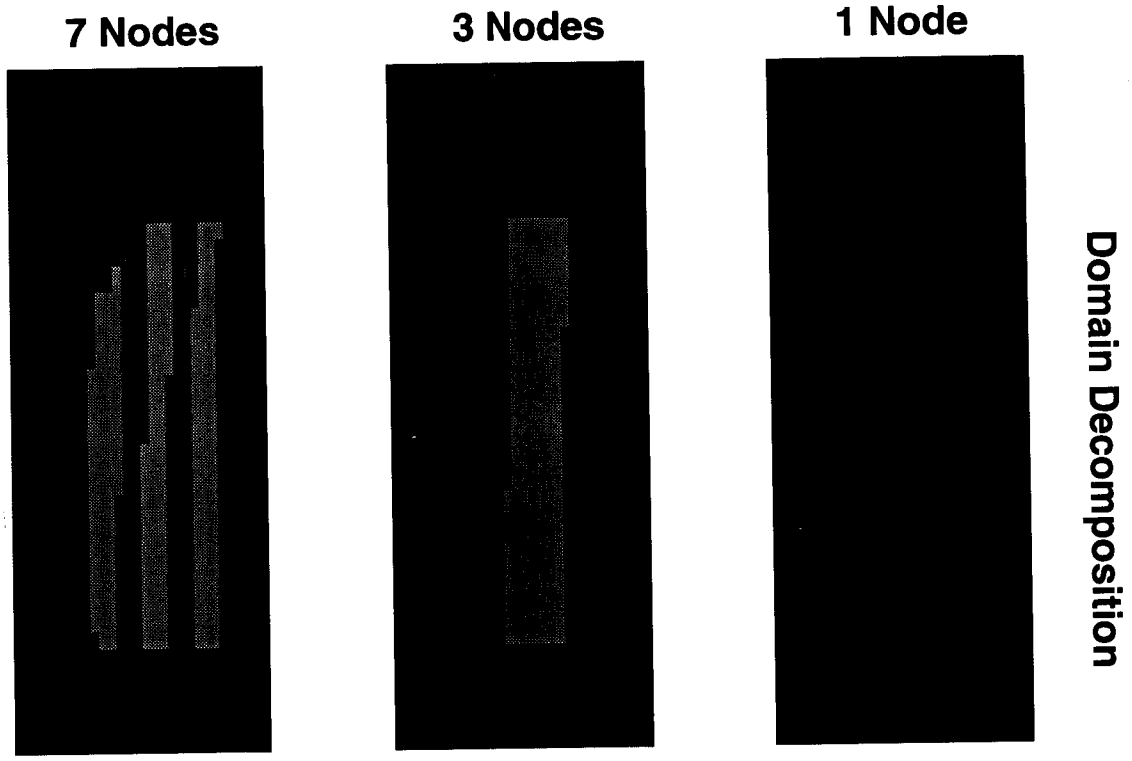

Fig. 1. Domain decomposition of the South Coast Air Basin region for different numbers of slave nodes. The dark rectangle represents the overall domain. The gray shaded area represents the computational domain. Different shades of gray represent sub-sections of the computational domain to be distributed to a specific node.

within the overall region. In the 3-node and 7-node decompositions of Fig. 1 the alternations of gray shades indicate the boundaries of the data as they are distributed among the available slave nodes. As can be seen, we have implemented a domain decomposition as close to a purely one-dimensional decomposition as possible. The reason to stay close to the one-dimensional case is discussed subsequently in the Horizontal Transport sub-section.

The rule followed to determine the domain decomposition is to have an approximately equal number of vertical columns sent to each node. In this manner, the load is well balanced. For chemistry implementation purposes only, the efficiency of the implementation is not dependent on the dimensionality of the decomposition.

4.2. *Chemistry*

As mentioned before, chemistry computations are the major computational load of typical AQMs. Their parallel implementation is rather simple since it does not involve communication among the "slave" nodes. The chemistry integrator used in the CIT model is an implicit, hybrid, asymptotic, exponential scheme developed by Young and Boris (1977). Parallel implementations of the chemistry integrator are a function neither of the numerical scheme used for the integration nor of the particular photochemical mechanism. Thus, parallel implementation of the chemistry does not interfere with the modularity of the code to any degree.

An effective approach is to send a collection of vertical columns to a node, using a predetermined domain decomposition. The programmer can set a predetermined decomposition rather easily in an AQM. Assuming that the time it takes to integrate any vertical column of cells is approximately the same, then one tries to send the same number of vertical columns to each slave node available, so that all the nodes finish their tasks at approximately the same time. The number of columns to be sent to each node should be computed dynamically. It is approximately equal, of course, to the total number of columns divided by the number of slave nodes available.

Figure 2 shows the time taken to perform a 24-h standard simulation of the South Coast Air Basin of California using the CIT model. The points plotted correspond to 2, 4, 8, 16, 32, 64, 128 and 256 nodes of the Intel Delta. The distance between the ideal and measured curves in Fig. 2 represents the time spent in the communication between the master node and the slave nodes, as well as idle time among the slave nodes. Figure 2 shows that the time continues to decrease as more and more nodes are used. This is the expected behavior in the small number of processors regime. Sometimes, however, an increase in time actually occurs when the number of processors is increased beyond a certain point. This phenomenon, if

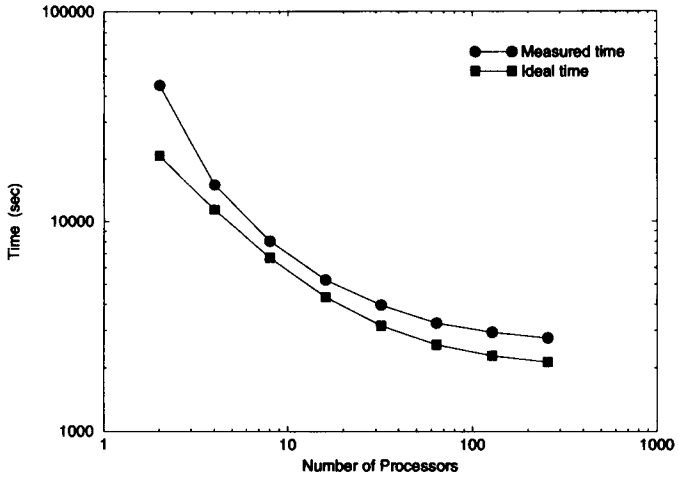

Fig. 2. Execution time as a function of number of nodes when parallelizing the chemistry loop of the model. Ideal time presented is calculated from Amdahl's law. Time reported corresponds to a 24-h standard simulation of the South Coast Air Basin.

present, occurs at the high number of processors regime (i.e. the massively parallel regime). The reason is that the master node has so many slaves to manage that the overall productivity of the group as a whole decreases. Using 256 nodes, the speed-up obtained by parallelizing the $\mathscr{L}_{cz}$ operator is 13.9. That is, a 24-h simulation takes 46 min to complete.

4.3. *Vertical transport*

In the CIT model vertical transport is part of the $\mathscr{L}_{cz}$ operator. In the chemistry implementations in parallel the entire air column was sent to the nodes to be integrated. Therefore, by sending the rest of the data needed, such as the z component of the advective flow, one can perform the vertical transport computations with no further complications. In the same way, deposition velocity calculations and other computations included inside the "chemistry" loop of the code should be performed in parallel.

If the chemistry operator is not coupled with the vertical transport, parallel performance is not greatly affected. Usually the percentage of the time spent in computing vertical transport is quite small. Particularly, in the CIT model it is less than 3% and is included in the "other" category. According to Amdahl's law, that 3% of code would, however, be crucial if most of the rest of the code could be implemented in parallel.

4.4. *Horizontal transport*

When the chemistry integration has been implemented successfully, as well as other computations such as the vertical transport and deposition, significant speed-ups are obtained. At this point, any implementation of other sections of the code will have a great impact on increasing speed-ups in accordance with Amdahl's law. The effect of implementing in parallel a constant small percentage of the code is proportional to the percentage of the code already parallelized. In particular, the CIT model spends 5.4% of its time performing the horizontal transport computations. This 5.4% would increase the speed-up of the code if implemented in parallel, and, the value of the increased speed-up would be proportional to the percentage of the code already parallelized as reflected in Amdahl's law.

The implementation of transport computations requires communication among the slave nodes. The communication needed for optimal performance is dependent on the numerical scheme used. The scheme determines the number of neighboring grids used to compute the values of the species concentration to advance the next transport step. An optimal performance implementation of horizontal transport can use the standard technique known as extended arrays. Extended array techniques consist of having the nodes send the values of species concentrations contained in the boundaries of their domains. The nodes would also receive the values of the concentration boundaries of their neighboring nodes.

The horizontal transport operator in the CIT model is decomposed into two separate operators $\mathscr{L}_x$ and $\mathscr{L}_y$. Our implementation of the transport computations did not make use of extended arrays. Thus, optimal performance was sacrificed for the sake of maintaining modularity.

We can illustrate the implementation of the transport computations with the approach we have used. For the $\mathscr{L}_x$ operator, the node containing the leftmost row of the computational grid is the one dedicated to perform the transport computations for that particular row. The assigned node requests data from the other nodes that contain data of that particular row. Note that data might be needed from more than

one node. The other nodes send the data to the assigned node. All the data required to perform transport computations are now in the assigned node. At this point, any algorithm can be used to compute the concentrations at the next transport step. All the communications within rows, as well as the transport computations, are carried out simultaneously at all the assigned nodes. Finally, all the assigned nodes must send the computed concentrations to the appropriate neighboring nodes.

The choice of the node as the place where computations are to be performed is arbitrary. For this idea to be implemented successfully, a unique message identifier must be set for each message going back and forth to the assigned node. This method, however, can be taken a step further. For instance, instead of having only one assigned node to compute the next transport step of all the species in all 5 vertical layers, one can assign 5 nodes to compute simultaneously all the species on each layer.

It is now clear that a one-dimensional domain decomposition helps to reduce the message passing when performing transport computations in the x-direction and using a small number of nodes. The y-direction transport is not affected. In comparison, the use of a two-dimensional decomposition results in a greater number of messages passed in the x- and y-directions. This is the case even when computing with a small number of slave nodes.

The transport in the y-direction was implemented using the same basic idea. All the y-column transport computations are to be performed at an arbitrarily assigned node. After receiving data from other columns, and performing the y-transport computations, the assigned node sends back the vector of species concentrations to the corresponding nodes. Even though this approach is not the optimal implementation to perform the horizontal transport computations, it is independent of the numerical scheme used.

The implementation of the horizontal transport in parallel increases the percentage of parallelized code by 5.4%. Since a significant percentage of the code has been already implemented in parallel, the speed-up increase is quite noticeable. Figure 3 shows that computing time continues to decrease as the number of nodes increases up to 256. Furthermore, it shows that the distance between the measured time and the ideal time is quite dependent on the number of nodes. Figure 3 shows a greater discrepancy between the measured times and ideal times when the chemistry and transport are both performed in parallel than Fig. 2 when only chemistry is parallelized. The reason is that the current implementation of horizontal transport sacrifices optimal transport implementation to gain modularity in the code. However, Fig. 4 shows that the parallel implementation of the transport makes the code run faster as long as 8 or more nodes are used. The time saved is proportional to the number of nodes used, as expected. When using only 4 nodes, the parallel transport implementation actually runs slower than when only the chemistry is implemented in parallel, a direct consequence of the overhead of intensive message passing that occurs to perform the transport calculations. As more nodes become available, the increase in performance overshadows such overhead.

When implementing the transport in parallel a greater speed-up is obtained than when only the chemistry part of the code is parallelized. The best speed-up measured is 28.47 using 256 nodes. That is, a 24-h run takes less than 23 min to compute. Figure 5 shows the computational time of the CIT model on

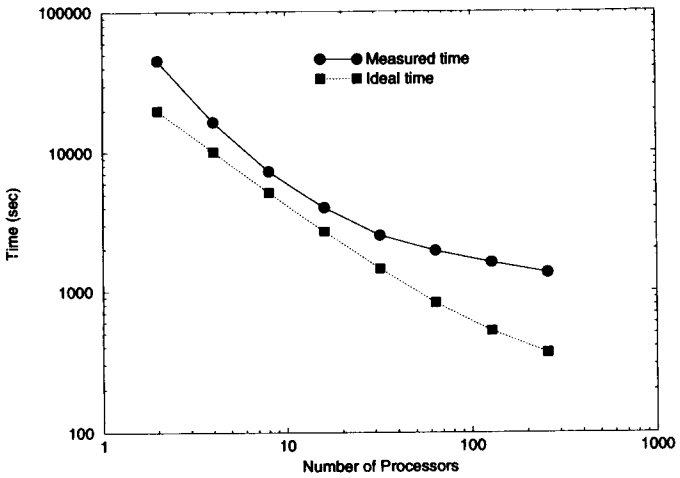

Fig. 3. Execution time as a function of number of nodes when parallelizing the chemistry loop and the transport equation solver of the model. Ideal time presented is calculated from Amdahl's law. Time reported corresponds to a 24-h standard simulation of the South Coast Air Basin.

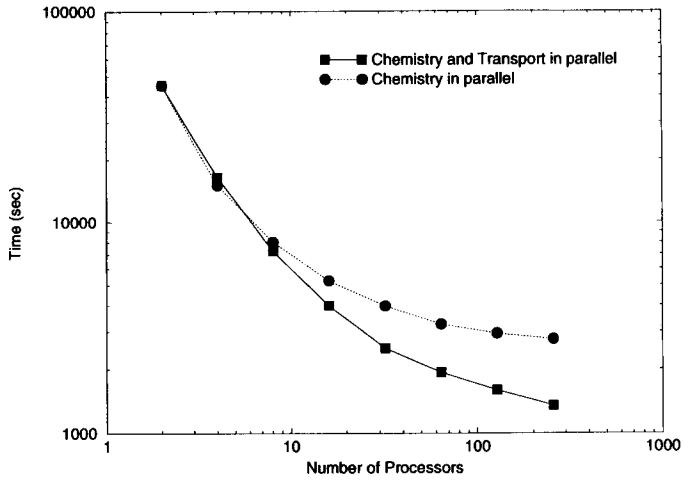

Fig. 4. Execution time as a function of number of nodes. One set of data corresponds to the parallelization of the chemistry loop only. The other data set corresponds to the model having the chemistry loop and the transport equation solver implemented in parallel. Time reported corresponds to a 24-h standard simulation of the South Coast Air Basin.

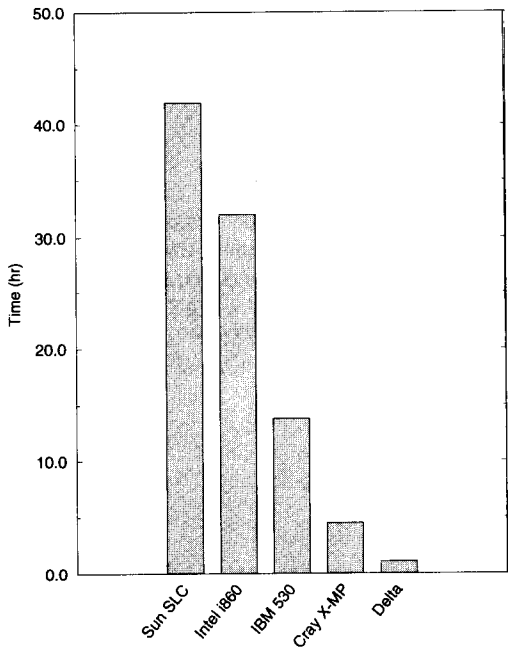

Fig. 5. Execution time for different computing platforms. Time reported corresponds to a 3-day simulation of the South Coast Air Basin. Delta result corresponds to the model having the chemistry loop and the transport equation solver implemented in parallel on the Intel Touchstone Delta using 256 processors.

different systems. The times reported correspond to simulating a 3-day air pollution episode in the South Coast Air Basin.

5. CONCLUSIONS

This work presents the implementation of the CIT photochemical air quality model on the Intel Touchstone Delta. It has been found that the use of MIMD computer architectures provides an excellent environment for air quality models.

The implementation of the chemistry section of the code involves a simple host to node communication pattern. The parallel implementation of the chemistry is independent of the numerical scheme and the photochemical mechanism used. When the chemistry portion of the CIT model is parallelized, a speed-up factor of 13.9 is achieved using 256 nodes.

Implementation of the horizontal transport section of the code requires more careful programming than for the chemistry since it inherently contains communication among nodes. The parallel implementation of the horizontal transport is dependent on the order of the numerical scheme, if extended array techniques are used. To obtain independence and keep the model modular with respect to the transport solver, a small performance price must be paid. When the horizontal transport portion of the CIT model is parallelized in addition to the chemistry, a speed-up factor of 28.47 is achieved using 256 nodes.

Parallel computing is a powerful tool for air pollution modeling. By significantly reducing the computing time, it allows more detailed treatment of the dynamics of the atmosphere and it provides the numerical power necessary to meet the needs of future generation AQMs.

Acknowledgements—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. This research was performed in part using the Intel Delta System operated by Caltech on behalf of the Concurrent Supercomputing Consortium.

REFERENCES

Carmichael G. R., Peters L. K. and Kitada T. (1986) A second generation model for regional-scale transport/chemistry/deposition. *Atmospheric Environment* **20,** 173–188.

Carmichael G. R., Cohen D. M., Cho S. Y. and Oguztuzun M. H. (1989) Coupled transport–chemistry calculations on the massively parallel processor computer. *Comput. Ch. E.* **13,** 1065–1073.

Chang J. S., Brost R. A., Isaksen I. S. A., Madronich S., Middleton P., Stockwell W. R. and Walcek C. J. (1987) A three-dimensional eulerian acid deposition model: physical concepts and formulation. *J. geophys. Res.* **92,** 14,681–14,700.

Chock D. P. (1985) A comparison of numerical methods for solving the advection equation—II. *Atmospheric Environment* **19,** 571–586.

Chock D. P. (1991) A comparison of numerical methods for solving the advection equation—III. *Atmospheric Environment* **25A,** 853–871.

Chock D. P. and Dunker A. M. (1983) A comparison of numerical methods for solving the advection equation. *Atmospheric Environment* **17,** 11–24.

Forester C. K. (1977) Higher order monotonic convective difference schemes. *J. comp. Phys.* **23,** 1–22.

Fox G., Johnson M., Lyzenga G., Otto S., Salmon J. and Walker D. (1988) *Solving Problems on Concurrent Processors Vol. I.* Prentice Hall, New Jersey.

Harley R. A., Russell A. G., McRae G. J., Cass G. R. and Seinfeld J. H. (1993) Photochemical modeling of the Southern California air quality study. *Envir. Sci. Technol.* **27,** 378–388.

Hov Ø., Zlatev Z., Berkowicz R., Eliassen A. and Prahm L. P. (1989) Comparison of numerical techniques for use in air pollution models with nonlinear chemical reactions. *Atmospheric Environment* **23,** 967–983.

Intel Corporation (1991) A Touchstone DELTA system description, Intel Supercomputer Systems Division, 26 Feb., 1991.

Lamb R. G. (1982) regional scale model of photochemical air pollution, part 1: theoretical formulation. U.S. Environmental Protection Agency, Research Triangle Park, N.C.

Levin E. (1989) Grand Challenges to computational science. *Comm. ACM.* **32,** 1456–1457.

Lurmann F. W., Carter W. P. L. and Coyner L. A. (1987) A surrogate species chemical reaction mechanism for urban-scale air quality simulation models. EPA No. 68-02-4104.

McRae G. J., Goodin W. R. and Seinfeld J. H. (1982a) Numerical solution of the atmospheric diffusion equation for chemically reactive flows. *J. comp. Phys.* **45,** 1–42.

McRae G. J., Goodin W. R. and Seinfeld J. H. (1982b) Development of a second-generation mathematical model for urban air pollution. I. Model formulation. *Atmospheric Environment* **16,** 679–696.

Odman M. T., Kumar N. and Russell A. G. (1992) A comparison of fast chemical kinetic solvers for air-quality modeling. *Atmospheric Environment* **26,** 1783–1789.

Pai P. and Tsang T. T. H. (1992) Parallel computations of turbulent-diffusion in convective boundary-layers on shared-memory machines. *Atmospheric Environment* **26A,** 2425–2435.

Pilinis C. and Seinfeld J. H. (1988) Development and evaluation of an eulerian photochemical gas–aerosol model. *Atmospheric Environment* **22,** 1985–2001.

Rood R. B. (1987) Numerical advection algorithms and their role in atmospheric transport and chemistry models. *Rev. Geophys.* **25,** 71–100.

Schere K. L. and Wayland P. K. (1991) EPA regional oxidant model (ROM 2.0): evaluation on 1980 NEROS databases. U.S. Environmental Protection Agency, Research Triangle Park, N.C.

Shin W. C. and Carmichael G. R. (1992) Comprehensive air pollution modeling on a multiprocessor system. *Comput. Ch. E.* **16,** 805–815.

Venkatram A., Karamchandani P. K. and Misra P. K. (1988) Testing a comprehensive acid deposition model. *Atmospheric Environment* **22,** 737–747.

Young T. R. and Boris J. P. (1977) A numerical technique for solving stiff ordinary differential equations associated with the chemical kinetics of reactive-flow problems. *J. phys. Chem.* **81,** 2424–2427.