Development of Real time Air Quality Prediction System

Jai-Ho Oh[1], Tae Kook Kim[1], Hung Mok Park[2], and Youngtae Kim[3]

[1]Department of Environmental Atmospheric Sciences, Pukyong National Univ., Busan, 608-737, Korea
[2]Department of Chemical Engineering, Sogang University, Seoul, 121-747, Korea
[3]Department of Computer Science and Engineering, Kangnung National Univ., Kangnung, 210-702, Korea

In this research, we implement Realtime Air Diffusion Prediction System which is a parallel Fortran model running on distributed-memory parallel computers. The system is designed for air diffusion simulations with four-dimensional data assimilation. For regional air quality forecasting a series of dynamic downscaling technique is adopted using the NCAR/Penn. State MM5 model which is an atmospheric model. The realtime initial data have been provided daily from the KMA (Korean Meteorological Administration) global spectral model output. It takes huge resources of computation to get 24 hour air quality forecast with this four step dynamic downscaling (27km, 9km, 3km, and 1km). Parallel implementation of the realtime system is imperative to achieve increased throughput since the realtime system have to be performed which correct timing behavior and the sequential code requires a large amount of CPU time for typical simulations.

The parallel system uses MPI (Message Passing Interface), a standard library to support high-level routines for message passing. We validate the parallel model by comparing it with the sequential model. For realtime running, we implement a cluster computer which is a distributed-memory parallel computer that links high-performance PCs with high-speed interconnection networks. We use 32 2-CPU nodes and a Myrinet network for the cluster. Since cluster computers more cost effective than conventional distributed parallel computers, we can build a dedicated realtime computer. The system also includes web based GUI (Graphic User Interface) for convenient system management and performance monitoring so that end-users can restart the system easily when the system faults. Performance of the parallel model is analyzed by comparing its execution time with the sequential model, and by calculating communication overhead and load imbalance, which are common problems in parallel processing. Performance analysis is carried out on our cluster which has 32 2-CPU nodes.

Key words : Realtime, Meso-scale, Downscaling, Air Quality, Dispersion, parallelization, MPI, Cluster

1. Introduction

Air quality prediction system is a numerical model which is a Fortran program designed for air quality simulations with four-dimensional data assimilation. This paper discusses the parallel implementation of air quality prediction system for distributed-memory parallel computers. Distributed-memory parallel computers are more cost effective and scalable than conventional shared – memory / vector parallel computers[6,7].

Corresponding Author ; Jai-Ho Oh, Department of Environmental Atmospheric Sciences, Pukyong National University, Busan 608-737, Korea
Phone : +82-51-620-6287
E-mail : jhoh@pknu.ac.kr

The parallel model we implement can also execute on the shared-memory parallel computers. To parallelize the model, we used MPI (Message Passing Interface), a library to support high-level routines for message passing[8]. We validate the parallel model by comparing it with the sequential model. Performance of the parallel model is also analyzed by comparing its execution time with the sequential model. Performance analysis was carried out on a PC cluster, a distributed-memory parallel computer that links high-performance PCs with high-speed interconnection networks.

2. Numerical Solution of Stiff Equations of Chemical Kinetics in the Atmospheric Dispersion Model

Human activities are a major force affecting the chemical composition of the earth's atmosphere. Carbon dioxide (CO_2), methane (CH_4), nitrons oxide (N_2O), and various chlorocarbons (ClC_S) and (FC_S) and VOS's are some important gases in the atmosphere that affect also the composition of ozone. The compositions of these gases are modeled by the following chemical - transport model.

$$\frac{\partial}{\partial t}(\rho C_i)+\nabla\cdot(\rho v C_i)=\nabla\cdot\Gamma\nabla C_i+S_i+S_i^{\gamma} \quad (1)$$

$$S_i=\sum_{m=1}^{MS}S_m^i(t)\delta(\xi-\xi_m)\delta(\eta-\eta_m)\delta(\zeta-\zeta_m) \quad (2)$$

Here, C_i is the composition of the i-th chemical species, S^i_m is the emission strength of the i-th chemical species emitted at the m-th location (point or area source), S^r_i is the chemical transformation rate of the i-th species, Γ is the turbulent eddy diffusivity tensor, δ is the Dirac delta function and (ξ_m, η_m, ζ_m) is the location of the m-th emission source. In the present investigation, the velocity field υ is determined by solving the MM5, and Eq.(1) is solved in a body-fitted coordinates to take care of the complex terrain.

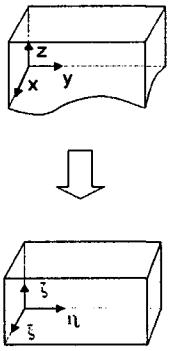
In the computational domain, Eq.(1) may be rewritten as

$$\begin{aligned}&\sqrt{g}\frac{\partial}{\partial t}(\rho C_i)+\frac{\partial}{\partial\xi}(\rho UC_i)+\frac{\partial}{\partial\eta}(\rho VC_i)+\frac{\partial}{\partial\zeta}(pWC_i)\\&=\frac{\partial}{\partial\xi}\{\Gamma\sqrt{g}(g^{11}\frac{\partial C_i}{\partial\xi}+g^{12}\frac{\partial C_i}{\partial\eta}+g^{13}+\frac{\partial C_i}{\partial\zeta})\}\\&+\frac{\partial}{\partial\eta}\{\Gamma\sqrt{g}(g^{21}\frac{\partial C_i}{\partial\xi}+g^{22}\frac{\partial C_i}{\partial\eta}+g^{23}+\frac{\partial C_i}{\partial\zeta})\}\\&+\frac{\partial}{\partial\zeta}\{\Gamma\sqrt{g}(g^{31}\frac{\partial C_i}{\partial\xi}+g^{32}\frac{\partial C_i}{\partial\eta}+g^{33}+\frac{\partial C_i}{\partial\zeta})\}\\&+\sqrt{g}S_i+\sqrt{g}S_i^{\gamma}\end{aligned} \quad (3)$$

where g^{ij} are geometric factors determined by the coordinate transformation relations.

Eq.(4) represents the convection - diffusion process and Eq.(5) takes care of the chemical transformation. Eq.(4) is solved by a finite volume method in a general curvilinear coordinates with the adoption of MUSCL to reduce artificial diffusivity. Perhaps, the most difficult part of the numerical solution of the atmospheric dispersion model lies in Eq.(5).

The equations of atmospheric chemical kinetics are very stiff. The problem with stiff equations is the severe limitation placed on the step size of an explicit scheme due to stability, which forces the use of an implicit scheme. The problem with the implicit scheme, however, is that a Jacobian matrix must be inverted at each time step. In the present investigation, we consider an algorithm that yields unconditionally stable, explicitly computable method for a class of chemical kinetics equations without requiring the use of a Jacobian matrix.



Physical Domain

Computational Domain

Fig.1. Schematic diagram of conversion physical domain to computational domain

The above atmospheric dispersion model is solved by employing the following splitting scheme

$$\frac{(\rho C_i)^*-(\rho C_i)^n}{\Delta t}=\nabla\cdot(\rho v C_i^*)=\nabla\cdot\Gamma\nabla C_i^*+S_i \quad (4)$$

$$\frac{(\rho C_i)^{n+1}-(\rho C_i)^*}{\Delta t}=S_i^{\gamma} \quad (5)$$

The chemical kinetics for the i-th species may be written as

$$\frac{dy_i}{dt}=R_i(y,t)=P_i(y,t)-L(y,t)y_i-\bar{L}(y,t)y_i^2 \quad (6)$$

where $Pi(y,t)$ is the matrix representing the generation of the i-th species and, L and $\bar{L}$ are diagonal matrices representing the destruction of the i-th species. When the nonlinear terms $\bar{L}(y,t)y_i^2$ are absent, the algorithm is given by

$$y_i^{n+1}=(y_i^{n+1})^l=y_i^n+\Delta t(P_i^n-L_i^n y_i^{n+1}) \quad (7)$$

where P_i^n and L_i^n are evaluated at the previous time step n Solving Eq.(7) for y_i^{n+1}, we find

$$y_i^{n+1} = \frac{y_i^n + \Delta t P_i^n}{1 + \Delta t L_i^n} \quad (8)$$

which has a first order accuracy in Δt. We may repeat the above calculation as follows.

$$(y_i^{n+1})^s = \frac{y_i^n + \Delta t (P_i^{n+1})^{s-1}}{1 + \Delta t (L_i^{n+1})^{s-1}} \quad (9)$$

where s is the iteration number. For the case where the second-order term $\overline{L_i}$ is included, the algorithm is generalized to:

$$(y_i^{n+1})^s = \frac{A + B}{2\Delta t (\overline{L_i}^{n+1})^{s-1}} \quad (10)$$

where

$$A = -(1 + \Delta t (L_i^{n+1})^{s-1}) \quad (11)$$

$$B = \sqrt{(1 + \Delta t (L_i^{n+1})^{s-1})^2 + 4\Delta t (\overline{L_i}^{n+1})^{s-1} (y_i^n + \Delta t (P_i^{n+1})^{s-1}} \quad (12)$$

To test the accuracy of the above algorithm, we consider a chemical kinetics involving the following 9 species.

$$O, O_3, OH, HO_2, H_2O_2, N_2, O_2, H_2O, O(^1D) \quad (13)$$

The results of the present algorithm are compared with those from an implicit scheme for stiff equations and reveals that the present algorithm yields reasonably good results at a decent amount of computer time. Figure 2 shows the diurnal variation of HO_2 in this model problem and the error of the present algorithm with respect to the exact implicit stiff algorithm such as the Newton-Raphson method. The present algorithm has been successful implemented in the three-dimensional atmospheric dispersion model of Eq.(3), where the model for the chemical kinetics is based on RADM2 treating 63 chemical species with 155 chemical reactions.

3. Parallelization of Air-Chemistry Model

The parallel implementation of Air-Chemistry Model uses MPI (Message Passing Interface) which is a standard library for message passing[7,8]. In this section, we describe parallelization of Air-Chemistry Model with MPI.

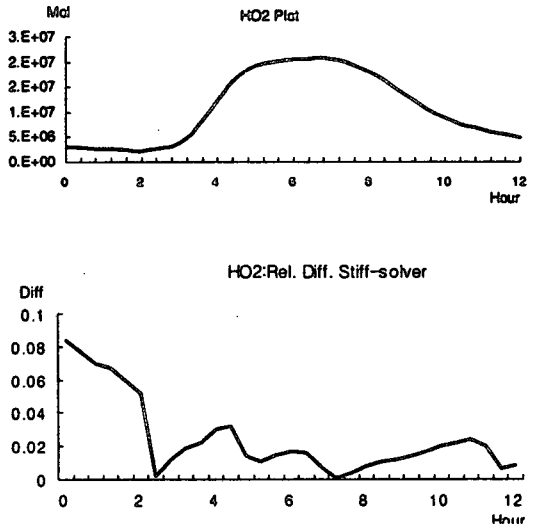


Fig.2. The diurnal variation HO2 (Mol) and differences with respect to stiff-solver.

3.1 Data mapping

The parallel Air-Chemistry Model does not decompose the three-dimensional domain onto processors, but each processor computes the allocated data area out of the whole domain. Figure 3 shows the data mapping of an example of using four processors.

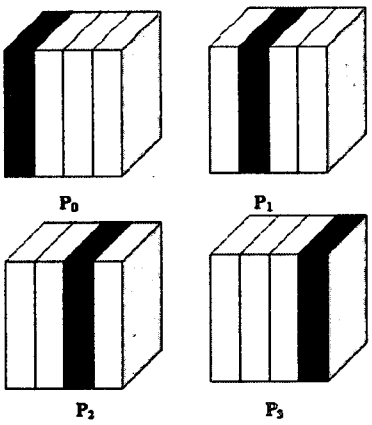


Fig.3. The data mapping

In the figure, each processor computes only the dark area which is allocated to itself. This mapping has same effect as the one-dimensional domain decomposition. Initially all processors have same data, and as computations proceed all processors have different data since each processor computes its own data area. Since the

parallel Air-Chemistry Model does not require frequent data exchanges between processors, this mapping can significantly reduce communication overheads[2].

3.2 Parallelization

We separate the Air-Chemistry Model code into two different parts for parallelizing. In the first part computations on each grid point are done independently upon the other grid points so that parallelism exists based on the domain decomposition. The second part, on the other hand, has data dependency so that it requires data exchanges between processors. Figure 4 shows the structure of the code.

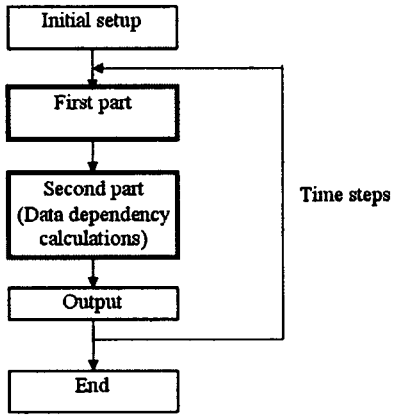


Fig.4. Program structure

The first part includes intensive computations to simulate chemical activities. The execution profile of the sequential model shows that the first part takes 97% of total execution time. Since we use one-dimensional domain decomposition, we modified the inner-most loop index (underlined) in the code as follows.

In the parallel code on the right hand side, **istart** and **iend** are local indices which are different on all processors. For example, if we use four processors and the **imax** value is 34, local index values on each processor is shown Table 1.

The second part computations have data dependency with the following code as an example.

con(i,3,2)=con(i+1,3,2)-con(i-1,3,2)

To calculate the variable **con**, each processor except boundary processors has to get data from processors in the left and the right respectively. Since computation of the second part takes only 7% of total execution time, we use **gather** and **broadcast** instead of data exchange between processors. After **gather**, all processors can calculate the above calculation since they already have required data in the whole domain. On the same domain, all processors do same calculations. In fact, there is no parallelism in the second part, however that does not effect on performance since the part is very small portion of all computations. Figure 5 shows the sub-domain calculations of the parallel program on each processor.

```
do k=1,kmax                          do k=1,kmax
   do j=1,jmax                          do j=1,jmax
      do i=1,imax                          do i=istart,iend
        call enviro                          call enviro
        call knair                           call knair
        call chemis                          call chemis
      end do                               end do
   end do                               end do
end do                               end do
```

Table 1. Local indices on 4 processors

Processor	**istart**	**iend**
P_0	1	9
P_1	10	18
P_2	19	27
P_3	28	34

4. Performance Analysis

The parallel code shows that its output is matched within numerical error the output of the original sequential code. In this section, we present performance evaluation based on various analysis.

4.1 Performance evaluation

The PC cluster which we tested the codes consists of 4 single Pentium III PCs and 1 dual CPU Pentium III PC. Table 2 shows the hardware and software of the cluster.

Figures 6 shows execution time on a 34×29×15 grid on different number of processors. We run the sequential model on a 450MHz CPU PC. In the figure, using 6 (and 8) processors means 4 processes on 4 single 450 MHz CPUs and 2 (and

4) processes on 1 dual 1GHz CPU. The comparison is fairly reasonable since the dual CPU is about twice faster than the single CPU.

The parallel execution on the dual CPU PC utilizes a characteristic of shared-memory architecture so that there is no explicit data communication but data exchange through the system bus. Figure 7 compares the execution time on the dual CPU PC. Execution time excludes the file I/O and message displaying time.

The results from the parallel model show good speedups over the sequential model. In Figure 6, for instance, the parallel model on 8-processor Cluster reduces CPU time from 64 minutes using the sequential model into less than 15 minutes with 14,790 grid points (34×29×15).

Table 2. Hardware and software of the cluster

Hardware			Software	
4 × 450MHz single CPU	Main memory	512 MB	OS	LINUX (Kernel
	Cache memory	512 KB		
1 × 1GHz dual CPUs	Main memory	1 GB	MPI lib	MPICH (version
	Cache memory	512 KB		
Switch hub	Intel T510 100 Mbps			

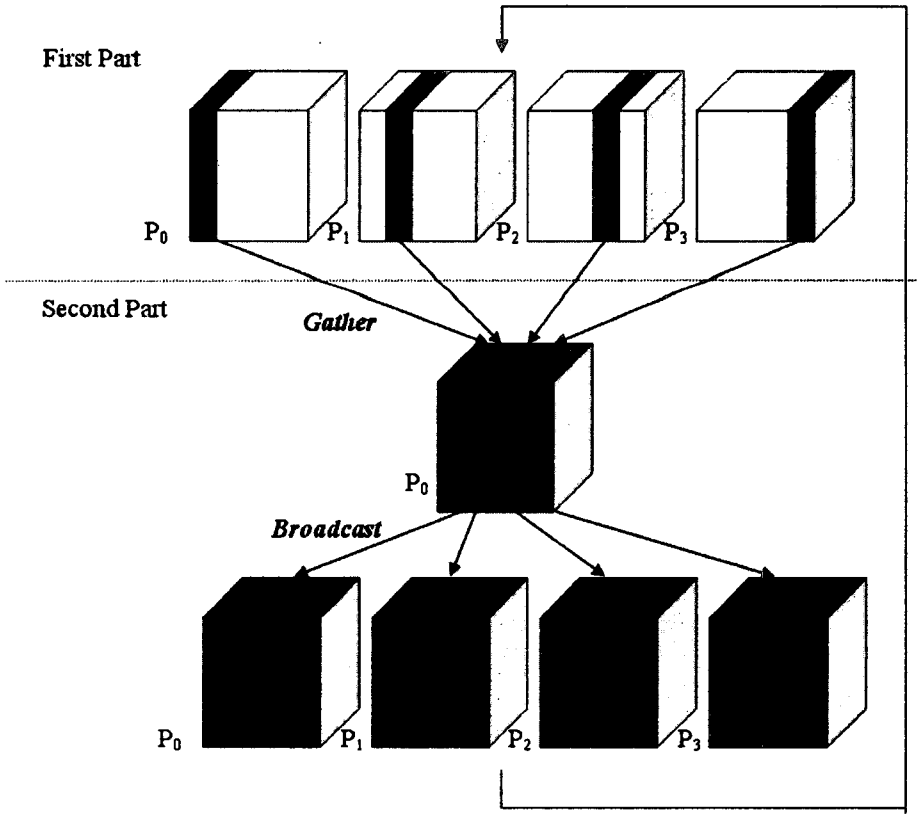


Fig.5. Sub-domain on each processor

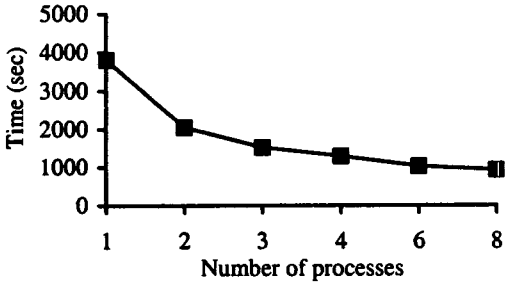


Fig.6. Comparison of execution time

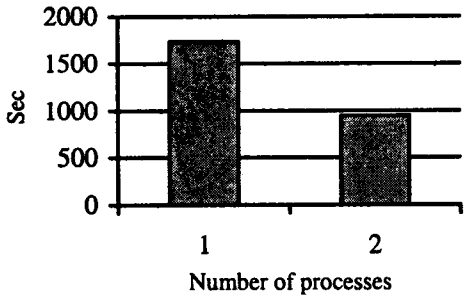


Fig. 7. Comparison of execution time on a dual CPU PC

5. Conclusion

We described el implementation of Air-Chemistry Model for distributed memory parallel computers. For the best efficiency, the parallelization of the model uses whole domain calculation instead domain decomposition. MPI supports parallelization of the model in a machine independent manner. We have validated the parallel model by comparing it to the sequential code using field data. To show the efficiency of parallel model, we have presented performance analysis. The performance analysis explains why our parallel Air-Chemistry model runs efficiently, and the parallel model shows good speedups compared to the ideal speedups that gives the maximum throughput.

References

1) G. Agrawal, A. Sussman and J. Saltz, *An Integrated Runtime and Compile-Time Approach for Parallelizing Structured and Block Structured Applications,* IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 7, pp. 747-754, July 1995.

2) J. Ambrosiano, J. Bolstad, A. Bourgeois, J. Brown, B. Chan, W. Dannevik, P. Eltgroth, B. Grant, C. Matarazzo, A. Mirin, D. Shumaker, and M. Wehner,, *High-Performance Climate System Modeling using a Domain and Task Decomposition Message-Passing Approach,* Proc. 1994 Scalable High-Performance Computing Conf., IEEE, pp. 397-405, 1994.

3) V. Bala, J. Bruck, R. Cypher, P. Elustonodo, A. Ho, C. Ho, S. Kipis, and M. Snir, *CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computers,* IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 2, pp. 154-163, Feb. 1995.

4) G. Fox, *Solving Problems on Concurrent Processors V1,* Englewood Cliffs, New Jersey: Prentice-Hall, 1988.

5) T. Huntsberger, *Distributed Algorithm for Two-Dimensional Global Climate Modeling,* Proc. 1994 Scalable High-Performance Computing Conf., IEEE, pp. 392-396, 1994.

6) K. Johnson, J. Bauer, G. Riccardi, K. Droegemeier, and M. Xue, *Distributed Processing of a Regional Prediction Model,* Mon. Wea. Rev., 122, 2558-2572, 1994.

7) Y. Kim, Z. Pan, E. Takle, and S. Kothari, *Parallel Implementation of Hydrostatic MM5 (Mesoscale Model),* The 8th SIAM Conference on Parallel Processing for Scientific Computing, 1997.

8) P. Pacheco, *Parallel Programming with MPI,* San Francisco, CA: Morgan Kaufmann, 1997.