Terr. Atmos. Ocean. Sci., Vol. 18, No. 3, 593-622, August 2007

Parallelization of the NASA Goddard Cumulus Ensemble Model for Massively Parallel Computing

Hann-Ming Henry Juang [1, *], Wei-Kuo Tao [2], Xiping Zeng [2, 3], Chung-Lin Shie [2, 3], Stephen Lang [4], and Joanne Simpson [2]

(Manuscript received 19 June 2006, in final form 3 January 2007)

ABSTRACT

Massively parallel computing, using a message passing interface (MPI), has been implemented into a three-dimensional version of the Goddard Cumulus Ensemble (GCE) model. The implementation uses the domain-resemble concept to design a code structure for both the whole domain and sub-domains after decomposition. Instead of inserting a group of MPI related statements into the model routine, these statements are packed into a single routine. In other words, only a single call statement to the model code is utilized once in a place, thus there is minimal impact on the original code. Therefore, the model is easily modified and/or managed by the model developers and/or users, who have little knowledge of massively parallel computing.

The model decomposition is highly flexible such that the entire model domain can be sliced into any number of partial domains in one- or two-dimensional decomposition. Data exchange is through a halo-region, which is overlaid with neighboring partial domains. A halo-region is also known as a ghost-cell region. For reproducibility purposes, transposing data among tasks into different decompositions is required, such as Fourier transform and full domain summation.

[1] Environmental Modeling Center, NCEP, NOAA, Washington DC, USA

[2] Laboratory for Atmospheres, NASA/Goddard Space Flight Center, Greenbelt, USA

[3] Goddard Earth Sciences and Technology Center, University of Maryland, Baltimore County, Baltimore, USA

[4] Science Systems and Applications Inc., Lanham, USA

[*] *Corresponding author address:* Dr. Hann-Ming Henry Juang, Environmental Modeling Center, NCEP, NOAA, Washington DC, USA; E-mail: Henry.Juang@noaa.gov

doi: 10.3319/TAO.2007.18.3.593(A)

The well-behaved performance of the implemented codes with anelastic and compressible versions on three different computing platforms indicates a successful implementation. The parallelization of both versions has speed-up of 99% for up to 256 tasks. The anelastic version has better speedup and efficiency because its numerical algorithm is preferred by the parallelization than that of the compressible version.

(Key words: Domain decomposition, Message passing interface, Massively parallel computing, Cloud-resolving model, Numerical weather prediction)

1. INTRODUCTION

Cloud-resolving models (CRMs), which are based on the non-hydrostatic equations of motion, have been extensively applied to cloud-scale and mesoscale processes during the past four decades (see a brief review by Tao 2003). Table 1 lists the major foci and some (not all) of the key contributors to CRM development over the past four decades. Because cloud-scale dynamics are treated explicitly, uncertainties stemming from convection that have to be parameterized in large-scale models are obviated, or at least mitigated, in CRMs. Also, CRMs solve the equations of motion with much higher spatial and temporal resolution and use more sophisticated and physically realistic parameterizations of cloud microphysical processes. CRMs also allow explicit interactions between clouds, radiation and surface processes. For this reason, the Global Energy and Water Cycle Experiment (GEWEX) formed the GEWEX Cloud System Study (GCSS), which chose CRMs as the primary approach to improve the representation of moist processes in large-scale models (GEWEX GCSS 1993 and Randall et al. 2003). Global models will use a non-hydrostatic framework with horizontal resolutions of 5 - 10 km in the next 5 years.

In recent years, exponentially increasing computer power has extended CRM integrations from hours to months (i.e., Wu et al. 1998) and the number of computational grid points from less than a thousand to close to ten million (Grabowski and Moncrieff 2001). Much attention is being devoted to precipitating cloud systems where the crucial 1-km scales are resolved in horizontal domains as large as 10000 km in two-dimensions and 1000×1000 km^2 in three-dimensions. Thus, many CRMs (see the review paper by Tao 2003) have to be re-coded in order to fully utilize the fast advancement of multi-parallel computing technology.

The programming requirement in numerical modeling for massive processor parallel architecture comprises domain decomposition and data communication (Aoyama and Nakano 1999). The techniques in domain decomposition and data communication may depend on particular numerical methods used in the models. Certain methods may do better with certain decompositions. Thus, several different domain decompositions need to be performed. Since atmospheric models are mainly three-dimensional, domain decomposition can be up to three dimensions. However, due to domain dependent computations, such as column physics, which require all model grids in the vertical, decomposition is used only up to two dimensions. Therefore, we can have either one-dimensional (1D) or two-dimensional (2D) decompositions.

Any 1D decomposition is usually simple and trivial to implement, but it limits the maximum number of tasks to be the number in the specific direction. Two-dimensional decomposition can have a larger number of tasks limited by the product of the two given dimensions and is known to have less total data to transfer in terms of exchanging halo data (Johnson et al. 1994). Two-dimensional decomposition was concluded to be more scalable than 1D decomposition in Skalin 1997a and b. Nonetheless, it also depends on the model and the platforms used. In the case of transposition between different decompositions, 1D was found to have the same performance as 2D in an IBM-SP and was better than 2D in the VPP5000. This conclusion can be realized because 1D has longer contiguous array lengths for vector computation on the VPP5000 (Juang and Kanamitsu 2001). However, 2D decomposition can be used to achieve high performance if the 2D array is rearranged to be contiguous in 1D for vector computation. Two-level parallelization with 1D decomposition and OpenMP for multi-tasking has also been done as a hybrid approach (Estrades et al. 2001).

Table 1. Major highlights of cloud modeling development over the past four decades.

	Major Highlights
1960's	Loading, Buoyancy and Entrainment
1970's	Slab- vs Axis-symmetric Model Cloud Seeding Cloud Dynamics & Warm Rain
1980's	Ensemble of Clouds - Cumulus Parameterization Cloud interactions and Mergers Ice Processes Super Cell Dynamics Squall Line Convective and Stratiform Interactions Wind Shear and Cool Pool Gravity Wave and Density Current Cloud Radiation Interaction
1990's	2D vs 3D Cloud-Radiation Quasi-Equilibrium – Climate Variation Implications Cloud Transport and Chemistry Diurnal Variation of Precipitation GEWEX Cloud System Study (GCSS) Coupled with Microwave Radiative Model for Satellite Cloud Retrieval (TRMM)
2000's	Land and Ocean Processes Multi-scale Interactions Energy and Water Cycle Cloud Aerosol-Chemistry Interactions Cumulus Parameterization Improvements Multi Lateral Boundary Conditions (open vs cyclic) Massively Parallel Computing Capability

The message passing interface (MPI) is the most popular massively parallel library binding in the meteorological community as well as parallel programming communities. Nevertheless, there are several ways to use MPI in a numerical model. Some introduce rules and add routines (Juang and Kanamitsu 2001), some use wrapper (Marshall et al. 1997), and some even require re-coding the entire model, such as WRF (Michalakes et al. 2001) and the NCEP GFS (Estrade et al. 2001). Severely modifying a model to obtain peak performance introduces two risks: 1) the code may be hard to read for further scientific and/or numerical improvements; and 2) it may not be reversible to a single cpu (sequential) code and difficult or tedious to implement new packages for future advanced architectures. One solution is to use MPI as an option to be able to run with both a single-processor and multi-processors either for a new model being developed (Juang et al. 2003) or in an existing model (Purnell and Revell 1995). Thus, the model code without MPI can be obtained whenever required. This also retains readability for easy future improvements.

This paper details how massively parallel programming via MPI is implemented into a three-dimensional (3D) version of a CRM, the Goddard Cumulus Ensemble (GCE) model. The implementation is unique, providing both the flexibility to run with any number of processors, even a single cpu, and high performance. Note that no extra dynamics and physics improvement are implemented, except code improvement with MPI in the current version. A brief description of the GCE model and experimental configuration are given in section 2. The concept of MPI implementation, along with the method of domain decomposition and data-communication to avoid aforementioned risks, will be described in section 3. The performance of the model with MPI, regarding the model dynamics (anelastic and compressible), stability, speedup, efficiency, reproducibility, wall-clock time comparisons among different decompositions by using three different computing platforms, will be given in section 4. The summary is given in section 5 with future model developments.

2. GODDARD CUMULUS ENSEMBLE (GCE) MODEL

2.1 GCE Model Description and Applications

The GCE model has been developed and improved at NASA/Goddard Space Flight Center over the past two decades. The development and main features of the GCE model have been described in Tao and Simpson 1993 and Tao et al. 1993. Recent improvements and testing were presented in Baker et al. 2001, Ferrier 1994, Lang et al. 2003, Lynn et al., 1998, Tao et al. 1996, Tao et al. 2003, and Wang et al. 2003. A Kessler-type two-category liquid water (cloud water and rain) microphysical formulation is mainly used with a choice of two three-class ice formulations, namely that by Lin et al. 1983 and the Lin scheme modified to adopt slower graupel fall speeds as reported by Rutledge and Hobbs 1984. An improved four-class, multiple-moment ice scheme has also been developed (Ferrier 1994) and tested for several convective systems in different geographic locations (Ferrier et al. 1995). Recently, two detailed spectral-bin microphysical schemes (Chen and Lamb 1999; Khain et al. 2000) were also implemented into the GCE model. Significant computation is required in applying the explicit spectral-bin

microphysics to study cloud-aerosol interactions and nucleation scavenging of aerosols, as well as the impact of different concentrations of aerosol particles upon cloud formation.

These new microphysics, however, require the use of a multi-dimensional Positive Definite Advection Transport Algorithm (Smolarkiewicz and Grabowski 1990) to avoid "decoupling" between mass and number concentration of cloud species. The positive definite advection scheme also produces more light precipitation, which is in better agreement with observations (Johnson et al. 2002). Solar and infrared radiative transfer processes (Chou et al. 1999; Chou and Suarez 1999) have been included (Tao et al. 1996). A sophisticated seven-layer soil/vegetation land process model has also been implemented into the GCE model (Lynn et al. 1998; Lynn et al. 2001; Lynn and Tao 2001). Subgrid-scale (turbulent) processes in the GCE model are parameterized using a scheme based on Klemp and Wilhelmson 1978, and the effects of both dry and moist processes on the generation of subgrid-scale kinetic energy have been incorporated (Soong and Ogura 1980). Table 2 shows the major characteristics of the GCE model.

Table 2. The main characteristics of the Goddard Cumulus Ensemble (GCE) model.

Parameters/Processes	GCE Model
Dynamics	Anelastic or compressible 2-D (slab- and axis-symmetric) and 3-D
Vertical Coordinate	Z (p)
Explicit Convective Processes	2-class water & 2-moment 4-class ice, 2- or 3-class ice Spectral-Bin Microphysics *
Implicit Convective Processes	Betts & Miller or Kain & Fritsch
Numerical Methods	Positive Definite Advection for Scalar Variables; 4-th Order for Dynamic Variables
Initialization	Initial Conditions with Forcing from Observations/Large-Scale Models
FDDA	Nudging
Radiation	k-distribution and four-stream discrete-ordinate scattering (8 bands) Explicit Cloud-radiation Interaction
Sub-Grid Diffusion	TKE (1.5 order)
Topography	Sigma-z(p)**
Two-Way Interactive Nesting	Radiative-Type*
Surface Energy Budget	7-Layer Soil Model (PLACE) CLM - LIS TOGA COARE Flux Module
Parallelization	1D and 2D decompositions with MPI

* indicates for the processes implemented into 2-D version only;
** for testing

Furthermore, several national and international universities and research institutions (i.e., University of Maryland, University of Virginia, Columbia University, University of New York at Albany, Seoul National University) are using the GCE model and its results in their research. In addition, the GCE microphysical processes, land processes and cloud-radiation interactive processes were implemented into three community mesoscale models (Penn State U/NCAR MM5, WRF and Advanced Regional Prediction System, ARPS). More than 90 referred journal papers were published by using the GCE model.

2.2 Anelastic and Compressible Versions

The GCE model flow can be either anelastic (Ogura and Phillips 1962), filtering out sound waves, or compressible (Klemp and Wilhelmson 1978), which allows the presence of sound waves. The sound waves are not important in thermal convections, but their processes can place severe restrictions on the time step in numerical integrations. For this reason, most cloud modelers use an anelastic system of equations in which sound waves have been removed by neglecting the local variation of air density with time in the mass continuity equation. A 3D diagnostic (elliptic) pressure equation can be solved using direct (e.g., Fast Fourier Transform, FFT) or iterative methods. For current implementation, we used FFT, which will be described more in section 3.4.

In the compressible system, the pressure-equation is derived both by taking the derivative of thermodynamic equation and by using the compressible continuity equation. Due to the presence of sound waves, a very small time step (2 sec for a 1000 m spatial resolution) is needed for time integration. However, Klemp and Wilhelmson 1978 developed a semi-implicit time-splitting scheme, in which the equations are split into sound-wave and gravity-wave components, to achieve computational efficiency. One advantage of the compressible system is its computational simplicity and flexibility. The numerical code then remains a set of explicit prognostic equations and alterations such as stretched or nested grids, surface terrain and boundary conditions (e.g., radiative upper boundary) can be incorporated into the numerical model without complicating the solution procedure.

Anderson et al. 1985 has tested an anelastic system using a 4-sec time step against the results from a fully compressible system without using the time-splitting technique (which needs a 0.3-sec time step). They found that the anelastic system produced essentially identical results to those of the compressible system for 2-D cool pool experiments lasting 500 sec. Ikawa 1988 also found that both simulated systems are similar if sound waves are damped enough in the compressible system for a 2D case involving orography. A pair of sensitivity tests, anelastic versus compressible, were also performed with a mid-latitude squall line using the GCE model (Tao and Simpson 1993). All model physics were activated in these two runs for a 12-h time simulation in order to maximize the possibility of differences. It showed that the GCE model does not produce identical results with the two different systems. The differences between the anelastic and compressible systems are much smaller, however, than those obtained by changing microphysical processes and advection schemes (Tao and Simpson 1993, see their Table 3 and Fig. 7).

2.3 Experimental Design

The model is very flexible and no limitation in terms of domain size and resolution because it is non-hydrostatic. Typically, a 3D version of the GCE model can have 256 × 256 or 512 × 512 horizontal grid points using 1 - 2 km resolution or better, and 40 - 60 vertical layers with stretched grids (up to 10 - 50 mb top). It can have either open or cyclic lateral boundary condition. In this manuscript, an arbitrary initial condition is selected to test the performance of this implementation. Three-dimensional configuration with 256 × 256 computational grid points in horizontal (260 × 260 if lateral boundary grids are included) and 34 grid-points in the vertical is used for all tests, except two cases of 512 × 512 × 34 and 1024 × 1024 × 34 grid-points used for examining larger dimension without increasing resolution. Both anelastic and compressible options of the model are included in all tests with cyclic boundary condition. Full model physics and statistical outputs are included in the test. The model integration time length is 4 hours, except for the long-period (3-day) stability runs. The time-step is 12 seconds in all cases for two different versions of the model. Table 3 shows the summary of the experimental designs. Three clusters are available for this test. They are HALEM (a Dec alpha cluster), the IBM-SP (a Power4 cluster), and CHAPMAN (a SGI Origin 2000 cluster). Note that, platforms used depend on availability, they are not chosen for comparison but for demonstrating the flexibility of MPI implementation into GCE.

3. MASSIVELY PARALLEL IMPLEMENTATION

In this section, it will illustrate how MPI can be implemented into a model (GCE model) without excessive code modification. The concept of adding MPI or grid-decomposition with-

Table 3. Summary of experimental design.

Configurations	Values
Dimensions (x, y, z)	(256, 256, 34) control run (512, 512, 34) or (1024, 1024, 34) different dimension runs
Resolutions	2 km
Time step	12 seconds
Boundary conditions	Horizontal: cyclic boundary condition, 2 grid points in each side Vertical: rigid boundary condition, 1 grid points extra in model top and bottom
Dynamics	Anelastic or compressible
Physics	Full physics as described in Table 2 with 2 clasee water 3 class ice statistical output
Duration	4 hours for control run 3 days for long term run

out disturbing the existing model structure and computation will be introduced. The decomposition and addition of the halo region will be described. The method used to exchange data among all the halo regions and to transpose data for FFT will also be shown. The basic and common MPI technical method such as decomposition can be found in Aoyama and Nakano 1999 and in Gropp et al. 1999a. However, the concept and more details related to current model implementation are in the following sub-sections.

3.1 Concept and Method of Parallelization and Its Flexibility

As mentioned, a major requirement when parallelizing a numerical model is to preserve readability for further scientific development and be reversible to be able to recover the original code for future computer architectures. This leads to some bottom-line solutions for parallelization. First, all the model structures, array indices and computations will be kept original so the code is more readable for further scientific development. Secondly, parallelization will be an add-on package containing all of the necessary MPI calls in one call at one place and not added throughout the code. A preprocessor is used to provide options as well as the ability to recover the original single-processor code.

This concept is different from most community or operational models. For example, the WRF model is written mainly in a structure for MPI with embedded model dynamics and physics (Michalakes et al. 2001); the NCEP GFS has been recoded several times mainly due to the arrangement of the data structure for MPI machines (Estrade et al. 2001). In the case of future platforms that are non-MPI, both codes have to be rewritten. The current implementation is based on sequential code with an option to add in MPI, which is close to the approach in MIT GCM (Marshall et al. 1997), though the methods are different. Thus, the sequential code can be obtained at anytime for future machines. Also the code can be used for full domain as well as partial domain without losing its readability and ease of scientific implementation. Together these features make the current implementation unique.

3.2 Methods of Decomposition

Following the main concept in the previous sub-section, the decomposition in grid-point space is illustrated in Fig. 1 with an example of 12 tasks in terms of three columns and four rows. The upper cube shown by the solid line represents the entire domain of the GCE model; the lower cube also shown by the solid line represents a single partial model domain for any task. The dashed lines form the inner portion of the entire model and partial model domains for the upper and lower cubes. The remaining outer portions (between the solid lines and dashed lines) are called lateral boundaries for the entire model domain in the upper cube and halo for the partial domain in the lower cube. The decomposition is based on the number of the inner grid in the upper cube. The number of east-west (north-south) grid is divided by the number of columns (rows); the remainders after division are the number of columns (rows) to have one more grid (more detail is given in the following paragraph). After each task obtains its portion of the model domain shown by the dotted lines from the upper cube, it also gets its halo grids to form the lower cube in Fig. 1. The shape between the entire domain (upper cube in Fig.1) and the partial domain (lower cube) may be different, but both have an inner portion and an

outer portion, again, which form a lateral boundary for the entire domain and either a halo or lateral boundary for the partial domain.

In order to illustrate decomposition in quantity, the following formula is introduced to define dimensions for any given directions:

$$N_s = INT\left[\frac{F_s - 2L_s - 1}{M_s}\right] + 1 + 2L_s \quad , \tag{1}$$

where N, F, M, and L are all integers, and *INT* followed by brackets indicates the result in the brackets is round to be an integer as in FORTRAN. F is the number of full grid points in any given direction. L is the number of lateral boundary and/or halo grid points. N is the number of partial grid points, and M is the number of decomposition slices. The subscript s can be used for any direction. In the case of a single cpu, $M = 1$, then $N = F$. For 1-D decomposition, then either M_{col} or M_{row} equals one. For 2-D decomposition, both M_{col} and M_{row} are larger than one. F-$2L$ may not be divisible by M without a remainder. The remainder R (always less than M) is distributed to tasks by adding one to each task for R tasks, as illustrated. Thus N is the maximum dimension for any partial domain. Any given task may have a grid number of N or N-1. For example, if total grid number in an east-west direction is given as 516 (F), the lateral boundary grid number is 2 (L), then the number of the inner grid is 512. If there are 3 (M) columns as in Fig. 1, to have equal grid number for each column, we divide 512 by 3, the outcome is 170 with 2 remaining. The number of the remainder, 2, is given to columns, one grid point by one column, so the final grid number distribution for these columns is 171, 171, and 170. Adding lateral boundary points to form a partial domain, we have 175, 175, and 174. Using formula (1), we have $N = INT[(516 - 2 \times 2 - 1) / 3] + 1 + 2 \times 2 = 175$. The same approach is used for north-south and vertical decompositions.

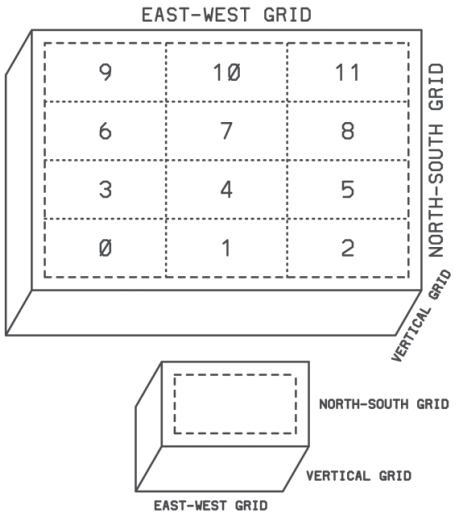

Fig. 1. Schematic diagram showing decomposition in grid-point space for an example using 12 tasks over the entire model domain and a partial domain with a halo for task 4. It shows a similar structure between the entire and partial domains if the dotted lines and numbers are eliminated from the entire domain.

3.3 Data Exchange for Halo Regions and Lateral Boundaries

Since the GCE model is a finite difference model, any given point requires neighboring points in order to compute its derivative. After one complete time step, the lateral boundary (halo portion) of the partial domain has to be updated. Each partial domain is assigned initially knowing its lateral boundary and/or halo regions. If the halo region is a lateral boundary condition for the model domain, the halo update will apply either an open boundary condition or a cyclic boundary condition. If the halo region is not a boundary condition of the model domain, the halo portion has to be updated by the neighboring partial domain. For example, task 5 in Fig. 1 has lateral boundary conditions on the east side but halo boundary conditions on the west, north and south sides. It needs to exchange data with tasks 8, 4, and 2 (side neighbors) and tasks 1 and 7 (corner neighbors) to update its halo. It also needs to exchange data with side neighbor task 3 and corner neighbor tasks 0 and 6 for cyclical boundary conditions.

Except for open lateral boundary conditions, all lateral and halo portions have to be updated by exchanging data among neighboring grids or grids related to the cyclical boundary condition from the related partial domain. Figure 2 illustrates data exchange in two steps with side neighbors. Data exchange illustrated in Fig. 2 is a completed data exchange including corner points. Shaded areas indicate updated data. The base of the arrow indicates where data is ready to be sent out, and the point of the arrow indicates where data are to be received and updated. First the data is exchanged in the x-direction as shown in Fig. 2a. After updating in the x-direction, the data is exchanged in the y-direction as shown in Fig. 2b. After Fig. 2b, all halo-portions should be updated. The lateral boundary has two arrows: the short one represents an open boundary condition that is updated by internal data, and the long one for cyclical boundary conditions, which is updated by the task at the furthest end of the model domain.

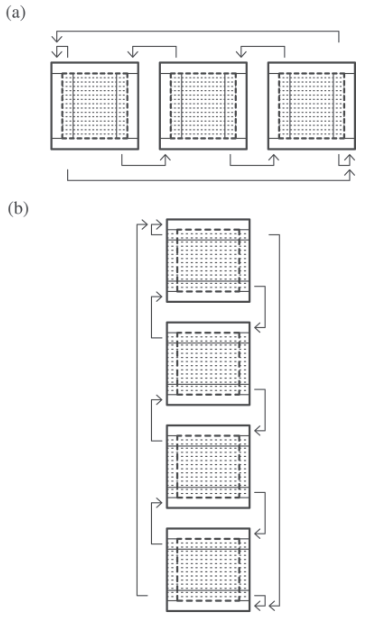

Fig. 2. Schematic diagram showing data exchange in a) the east-west direction and b) the north-south direction for grid-point space. The dashed lines indicate the inner-border of the halo regions, and the arrows indicate the direction of data movement. The thin solid lines show the area of data with other lines indicating where it is to be moved. The lateral boundary conditions determine which data to move and where by using two arrows at the lateral boundaries.

Since all tasks work together, assuming the start-up time (latency) for MPI communication is negligible, the wall clock time can be related to the amount of data exchanged in one complete data exchange (Fig. 2) by the task that has the longest running time. The amount of data exchanged can be measured by using Eq. 1 and Fig. 2. From Fig. 2a, the data exchanged along x are:

$$2L_x\left[\frac{F_y-2L_y-1}{M_{row}}+1\right] \quad , \tag{2}$$

where L_x is the data depth and the quantity in the brackets is the data length. Then from Fig. 2b, the data exchanged along y are:

$$2L_y\left[\frac{F_x-2L_x-1}{M_{col}}+1+2L_x\right] \quad , \tag{3}$$

where, again, L_y is data depth, and the quantity inside the brackets is the data length, which includes lateral boundary grids. For simplicity, assume $L_x = L_y$ so that combining Eqs. 2 and 3 together gives:

$$2L\left[\frac{F_y-2L-1}{M_{row}}+\frac{F_x-2L-1}{M_{col}}\right]+4L^2+4L \quad , \tag{4}$$

where the values inside the bracket determine the variation of data exchange. The two terms in the brackets indicate the number of inner grid points in the y and x directions of the partial domain, respectively. This shows that minimizing the value in the brackets requires minimizing the values of both terms, and it indicates that the larger the M, the smaller the value of each term will be. Thus, 2-D decomposition results in less data exchange than 1-D decomposition. For example, let $L = 2$ and F be 165 for both directions with 16 cpus. For 16 cpus, we can have 1×16, 2×8, and 4×4. In case of 1×16, the value in the brackets is 160 + 160 / 16 = 170. In case of 2×8, the value in the brackets is 160 / 2 + 160 / 8 = 100. In 4×4 case, the value is the brackets is 160 / 4 + 160 / 4 = 80. Thus, 4×4 has the smallest value in the brackets. Some evidence will be shown in section 4.

3.4 Transposing Data for FFT

In an anelastic system, FFT is used to solve the diagnostic pressure equation (Ogura and Phillips 1962). For FFT in the horizontal x and y directions, grid point values of the entire domain are needed. In this case, the model domain should not be sliced in x and y directions, only the z-direction can be sliced. The consequence of this is that only 1D decomposition in the z-direction is possible for FFT. The number of tasks is thus limited to the dimension of the

vertical direction. Fortunately, the FFT in the original design of the GCE model is a 1D FFT. In other words, it does FFT in the x-direction first, called FFTX, then FFT in the y-direction, called FFTY, to obtain grid-point values (spectral coefficients) from spectral coefficients (grid-point values). Thus, an MPI transpose can be used between FFTX and FFTY transformations to provide entire grid-point values or spectral coefficients from one direction to the other. In this case, 2D decomposition can be used because only one direction cannot be sliced due to transformation; another direction and the vertical direction can be sliced. Therefore, a larger number of tasks can be used in 2D decomposition than that in 1D decomposition, because 2D decomposition is limited by the multiplication of two minima grid point numbers instead of one minima grid point number among three directions.

Figure 3 shows the entire process for FFT in an example of 12 tasks using 2D decomposition with 3 columns and 4 rows. In the case of 1D decomposition, the entire process is redone without the three transposes marked by the hollow arrows and with no slicing for the 3 columns, only for the 4 rows. The four thin arrows indicate the transformation between grid-point values and spectral coefficients, and the three thick arrows indicate transposes for either 1D or 2D decomposition. Starting from the bottom left cube in grid-point space, the process follows the arrows to spectral space in the second row from the left of the bottom cube. Then, after spectral computations, it follows the arrows back to the top cube on the left in grid-point space.

Figure 4 shows a schematic diagram illustrating the MPI transpose for a 2D decomposition. The 1D MPI transpose is illustrated in Fig. 2 of section 3 of Juang et al. 2003. The 2D transpose can be described here in the same fashion. The solid lines indicate the existing decomposition for the upper cube after transposing in the bottom cube. The dashed lines indicate the slicing for each task (large-font numbers) before the sliced data is sent out to the tasks (small-font numbers) in the upper cube. The dashed lines in the bottom cube for each task (large numbers) indicate the portion of sliced data received from other tasks (small numbers). This example shows the transpose in x and y directions over the entire grid for FFT in either x or y direction.

There are some parallel FFT packages available, but they are not used in the current implementation. Since regular FFT is already used in the code for a single cpu, and the data is transported so as to have a full grid in one direction for regular FFT, then having a full grid in another direction works efficiently enough without the necessity of implementing a parallel FFT package.

4. PERFORMANCE

This section contains several sub-sections. First, the stability of repeat runs and one long continuous run and the performance of different decompositions have to be examined to establish a benchmark for evaluating the performance of single and short-period runs. Next, the performance of different tasks in terms of wall-clock time, speedup and efficiency, for versions with and without Fourier transforms, different resolutions, and different platforms are shown.

Performance in terms of wall-clock-time in this section is based on experimental design described in section 2.3. Since in-line statistics option in the model is useful and used as a default, all results hereafter are run with in-line statistics for a complete performance test.

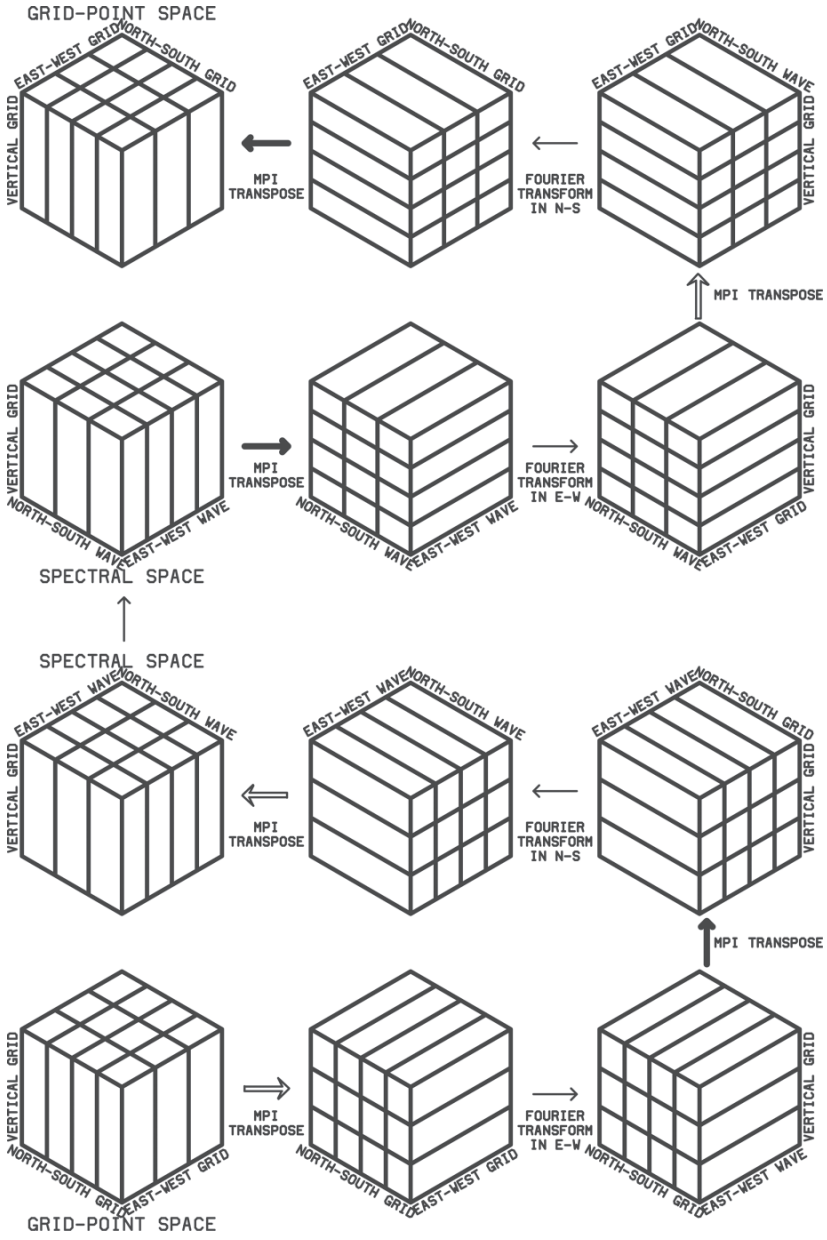

Fig. 3. Transpose of data between different decompositions for different grid-point and spectral spaces for Fourier transform in the anelastic version of the model. Thin arrows indicate Fast Fourier Transforms between grid and spectral spaces, thick arrows MPI transposes needed in any decomposition, and hollow arrows MPI transposes when 2-D decomposition is used.

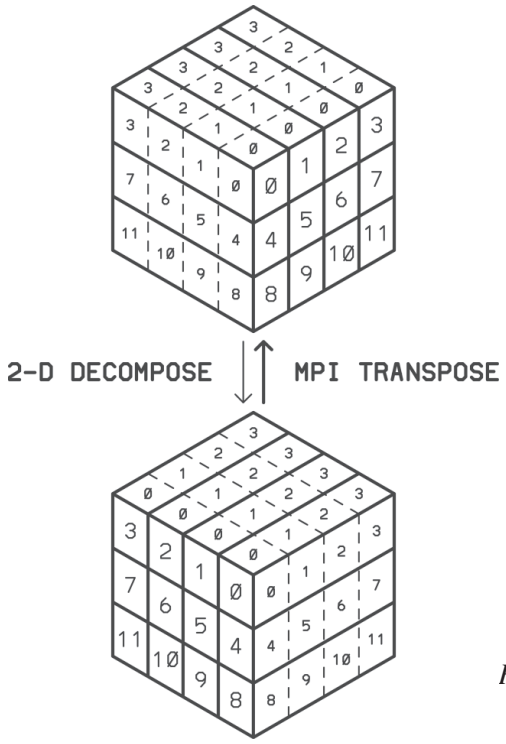

Fig. 4. Schematic diagram showing an MPI transpose using 2-D decomposition.

4.1 Performance Stability

Before showing results for single runs under all conditions, the stability of the performance has to be examined. Table 4 gives the wall-clock times for five arbitrary runs on HALEM and the IBM-SP using the same model configuration and decomposition of 8 × 8 (64 tasks). The same model configuration on CHAPMAN but for 1-hour integration is also listed with four members. The maximal time differences among different runs vary by 25 seconds on HALEM, and about 2.9% of the mean of 866 seconds. For the IBM-SP, the maximal time difference among different runs is 74 seconds, about 6.6% of the mean of 1121 seconds. For CHAPMAN, there is an 8-second possible difference, which is about 2.3% of the mean of 339 seconds. These percentages are needed to measure the possible variation of all the results shown hereafter since there is only one run for each condition. In other words, results shown hereafter can be variated around these percentages.

Table 5 lists the wall-clock times for 3-day and 4-hour integrations using the same configuration (256 × 256 × 34 grid points) and 128 tasks on HALEM. The amount of wall-clock time with respect to the model integration time is about the same between the short-period 4-hour run and the long-period 3-day run. This indicates there is no need to conduct long integrations for performance evaluations.

Table 4. Wall-clock times from repeated runs.

Platform	Dimension	Task	Integration	Wall-clock time
HALEM	256x256x34	64(8x8)	4 hours	866±12 sec
IBM-SP	256x256x34	64(8x8)	4 hours	1121±37 sec
CHAPMAN	256x256x34	64(8x8)	1 hour	339 ± 4sec

Table 5. Wall-clock times for 3-day and 4-hour runs.

Platform	Dimension	Task	Integration	Wall-clock time
HALEM	256x256x34	128(8x16)	3 days 4 hours	7803 sec 460 sec
IBM-SP	256x256x34	128(8x16)	3 days 4 hours	13680 sec 754 sec

4.2 Different Decompositions

As mentioned, the design of the decomposition that was implemented can be integrated with any number of tasks, whether the number is a prime number or not. If it is a prime number, the model will run 1-D decomposition only by itself; otherwise it runs either a 1-D or 2-D decomposition. Since any given non-prime number can have several decompositions, it would be good to know the performances for an optimal decomposition.

Figure 5 gives examples of the performance for several different possible decompositions on the HALEM, the IBM-SP, and the CHAPMAN. There are 7 different decompositions for 64 tasks: 1×64, 2×32, 4×16, 8×8, 16×4, 32×2, and 64×1 given as the number of slices in the x-direction times the number of slices in the y-direction. Note that, 1×64 and 64×1 are 1-D decomposition and others are 2-D decomposition. In the figure, the seven groups of bars represent the seven decompositions. Each group of bar plots has three parts, the left part of each bar is for HALEM, the central part for the IBM-SP, and the right part for CHAPMAN. From these platforms, there are two major conclusions: 1) The closer the numbers of the two slices are, the better the performance; and 2) a decomposition using a smaller number of slices in x than in y is better than the reverse. Hence, 8×8 is the best then 4×16 followed by 2×32, and finally 1×64. And 4×16 is better than 16×4; 2×32 is better than 32×2; and 1×64 is better than 64×1. These results may depend on models and machines, but current results seem to be common to others.

The difference between the best decomposition (8×8) and the worst decomposition (64×1) here is significant. Hence, having the best decomposition for any given number of tasks should

be the default configuration, though the model is designed to be flexible. The reason for the significant differences comes from the combination of array indices and loops for computations, the amount of data communication (see Eq. 4), and the architecture of the scalar machine. For cache type of scalar machine, the array length should not be too large, so the data can be reused in cache with time-saving instead of accessing from memory. 2-D decomposition with about an equal task number in 2-D provides not only less data to communicate with a sub-group but also a short length of array to take advantage of the cache type scalar machine. Nevertheless, this conclusion may not be valid for a vector cluster such as the VPP5000 vector machine as shown in Fig. 3 in Juang and Kanamitsu 2001. In a vector machine, array can be operated at the one time in length, the fewer the contiguous arrays to be computed, the shorter the total wall-clock time for a given domain. So it can be expected that 1×64 will achieve the best performance among vector machines. Thus, the decomposition design is platform dependent.

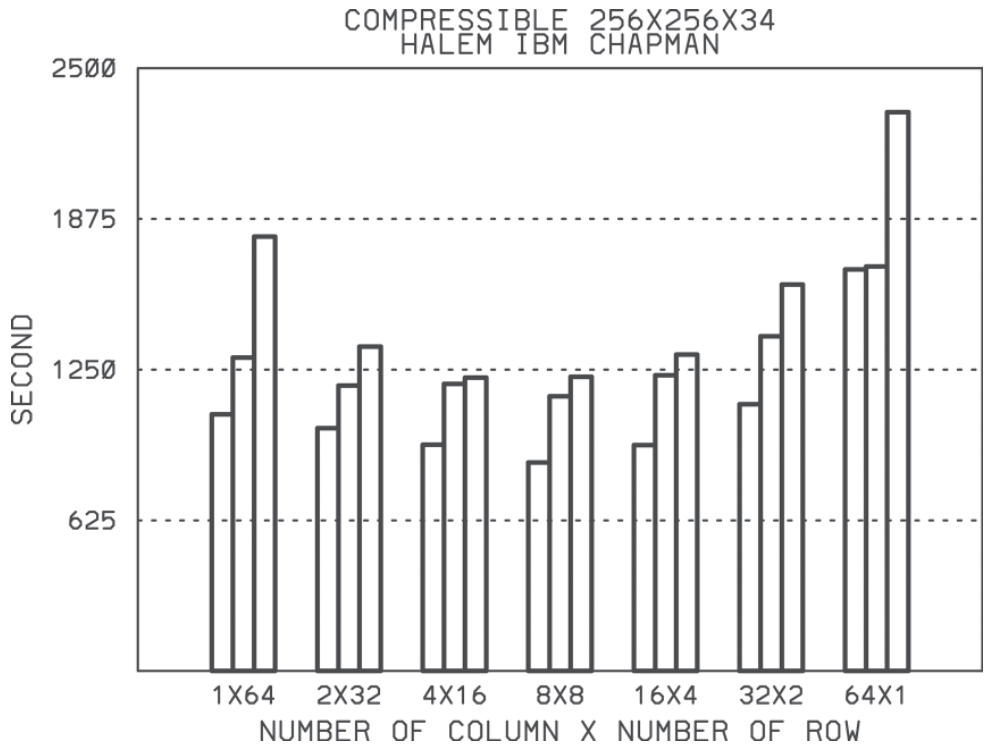

Fig. 5. Bar plot for different decompositions using the compressible version of the model. The left bar is for results on HALEM, the central bar on the IBM-SP, and the right bar on CHAPMAN. The numbers on the x-axis indicate the decomposition in terms of the number of columns times the number of rows.

4.3 Wall-Clock Times for Different Tasks, Platforms, and Versions

Following the previous two sub-sections, a short integration period using the best decomposition can be used to evaluate the performance of a number of different tasks, platforms, and versions of the model. Figures 6, 7, and 8 show the wall-clock time with respect to the number of tasks for model dimensions of 256 × 256 × 34 using (a) compressible, and (b) anelastic versions of the GCE model. Due to geometric increments in the number of tasks, a logarithmic scale is applied to both axes. The number of tasks along with their decomposition used in these figures are 1, 4 (2 × 2), 16 (4 × 4), 32 (4 × 8), 64 (8 × 8), 128 (8 × 16), 256 (16 × 16), and 512 (16 × 32). The symbol "X" in the figure indicates the location of the wall-clock time for each number of tasks, and the number beneath the symbol "X" is the value of the wall-clock time. In order to show the relative performance, one solid line, one dotted curve, and one dashed curve are drawn in each plot along with the individual runs. They represent the theoretical wall-clock time for the number of tasks between 1 and 512. The solid line represents perfect performance, which is a line based on wall-clock time for a single task divided by the given number of tasks (Amdahl's law) between 1 and 512. The dotted and dashed curves indicate 99% and 95% parallelization, respectively. They can be obtained from:

$$T(n) = T(1)(1-p) + \frac{T(1)p}{n} + c(n) \quad , \tag{5}$$

where $c(n)$ is the communication cost (which is neglected to simplify discussion), $T(n)$ is wall-clock time for n number of tasks, $T(1)$ is wall-clock time for a single task (which is the largest value shown on the y axis for each plot), and p is the percentage of all computation, which can be parallelizing. When p equals 100%, $T(n)$ is a solid line; when p equals to 99%, $T(n)$ is a dotted curve; and when p equals 95% , $T(n)$ is dashed curve. The perfect line and curves here are drawn for the case of $c\,(n) = 0$.

Figure 6 shows the wall-clock time results for HALEM for (a) the compressible version, and (b) the anelastic version of the GCE model. The results show a similar trend. The performance is close to 95% for 4 tasks, 99% for 16 tasks, almost perfect parallelization for 128 tasks, then returns to 99%. Figure 7 is the same as Fig. 6 except that it is for the IBM-SP. Wall-clock times are 99% parallelization. Strictly speaking, two versions performed similarly; close to 95% for 4 tasks, 99% for 64 tasks, and less than 99% for up to 512 tasks. Figure 8 is the same as Fig. 6 except for CHAPMAN. The results show both versions of the model perform excellently in parallelization between 16 and 256 tasks.

Comparing different versions of the model on the different platforms, HALEM, IBM-SP, and CHAPMAN, reveals: 1) the compressible version has better speedup for a single cpu or small number of tasks; 2) the anelastic version becomes faster in time spent than or equal to the compressible version for large numbers of tasks; 3) both versions spend less wall time in IBM-SP than those in HALEM or CHAPMAN for a small number of tasks but not for a larger number of tasks. Results 1) and 2) are due to additional computations required in the anelastic version, and 3) indicates that the model computes faster but communicates slower in IBM under our experimental design. However, note that, with certain MPI environment settings;

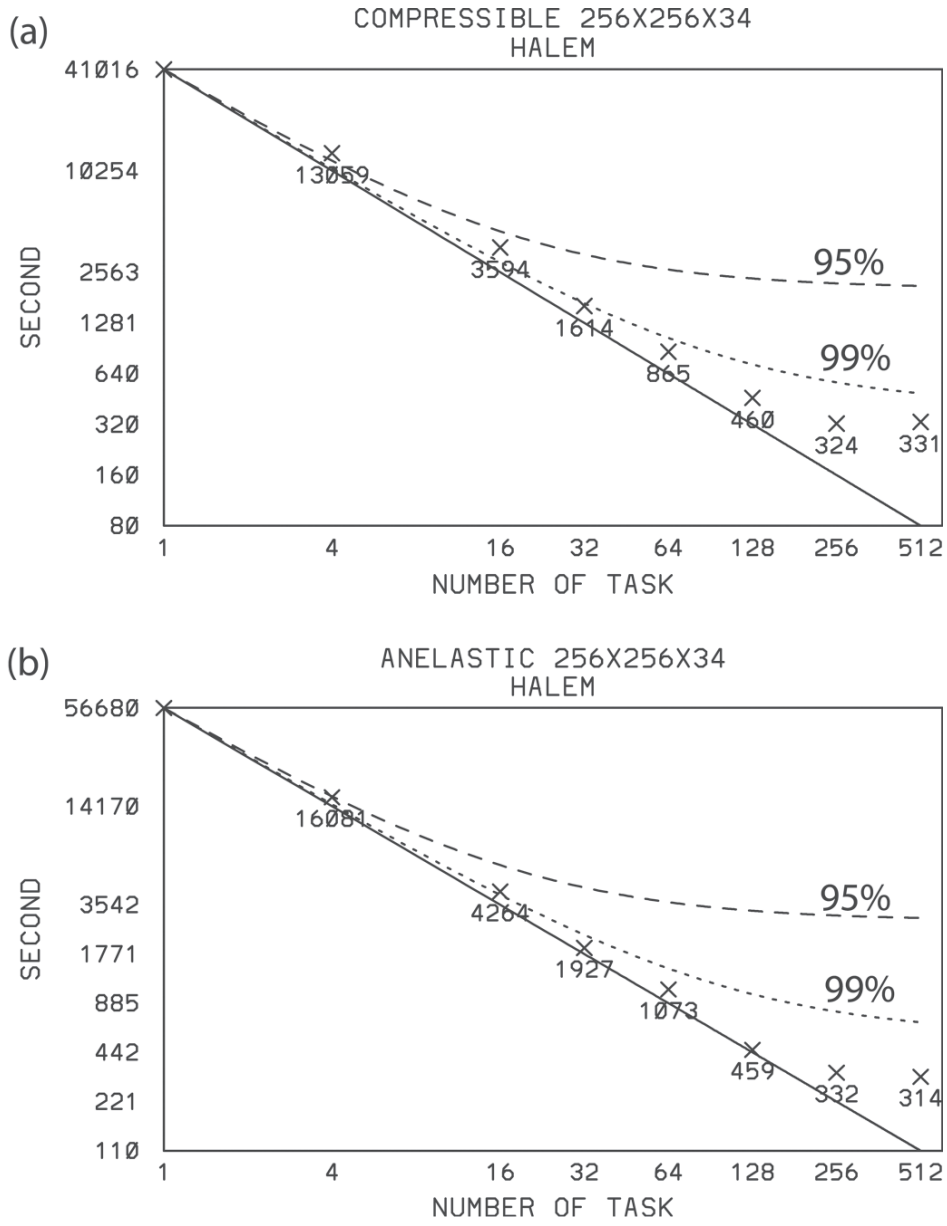

Fig. 6. Wall-clock time in seconds with respect to number of tasks for (a) the compressible and (b) the anelastic versions with 256 grid points in the x- and y-directions and 34 in the z direction on HALEM. The solid line indicates perfect parallelization based on wall-clock time for a single task. Wall-clock time is shown on the y-axis. The dotted curve indicates 99% parallelization, and the dashed curve indicates 95%. The numbers next to the symbol "X" are the actual wall-clock times.

such as memory shared in MPI mode, GCE can have good communication in IBM. The reason why CHAPMAN achieved excellent parallelization may be related to its cache advantage for the proper length of the array used in GCE. Furthermore, the results indicate that it is inefficient to run this model with a dimension of 256 × 256 × 34 using more than 256 tasks.

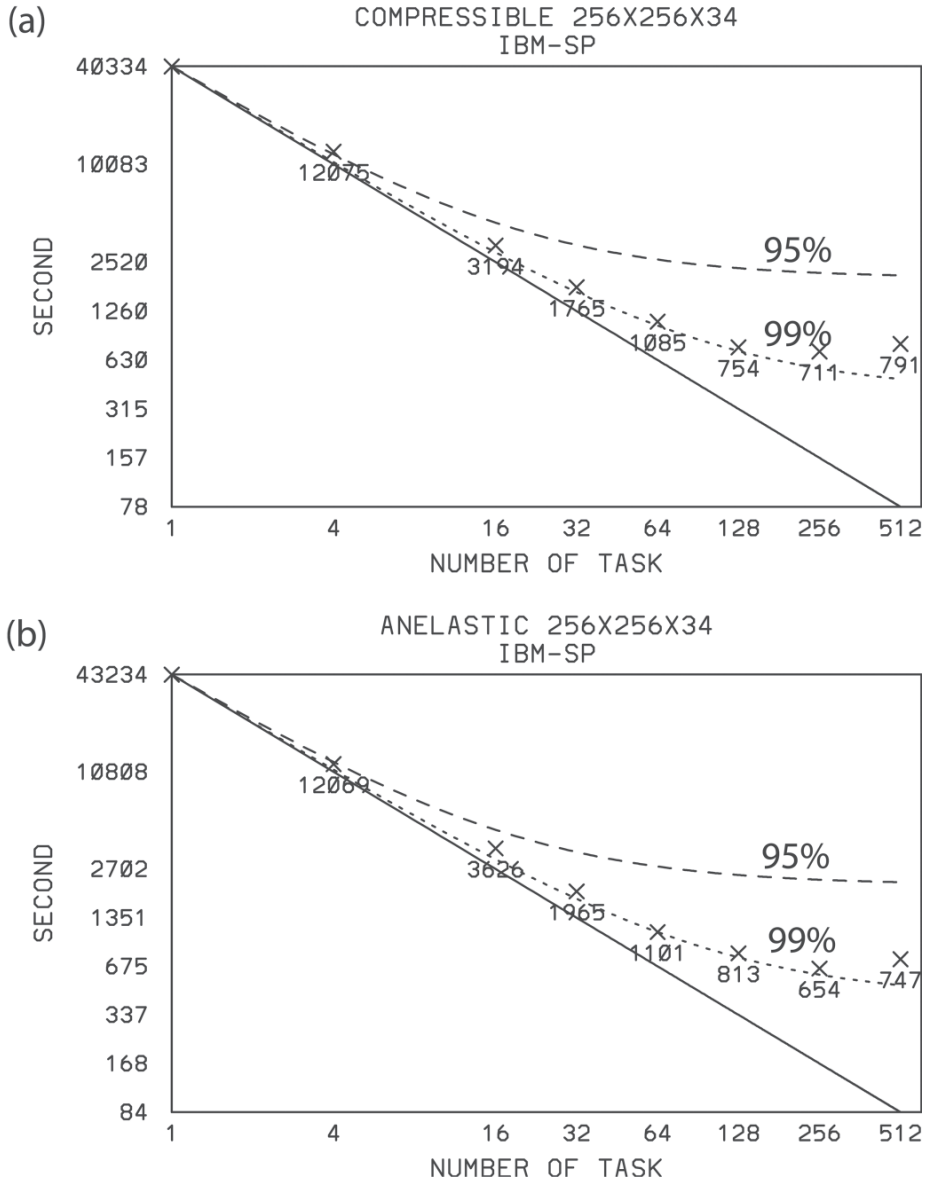

Fig. 7. Same as Fig. 6 except for the IBM-SP environment.

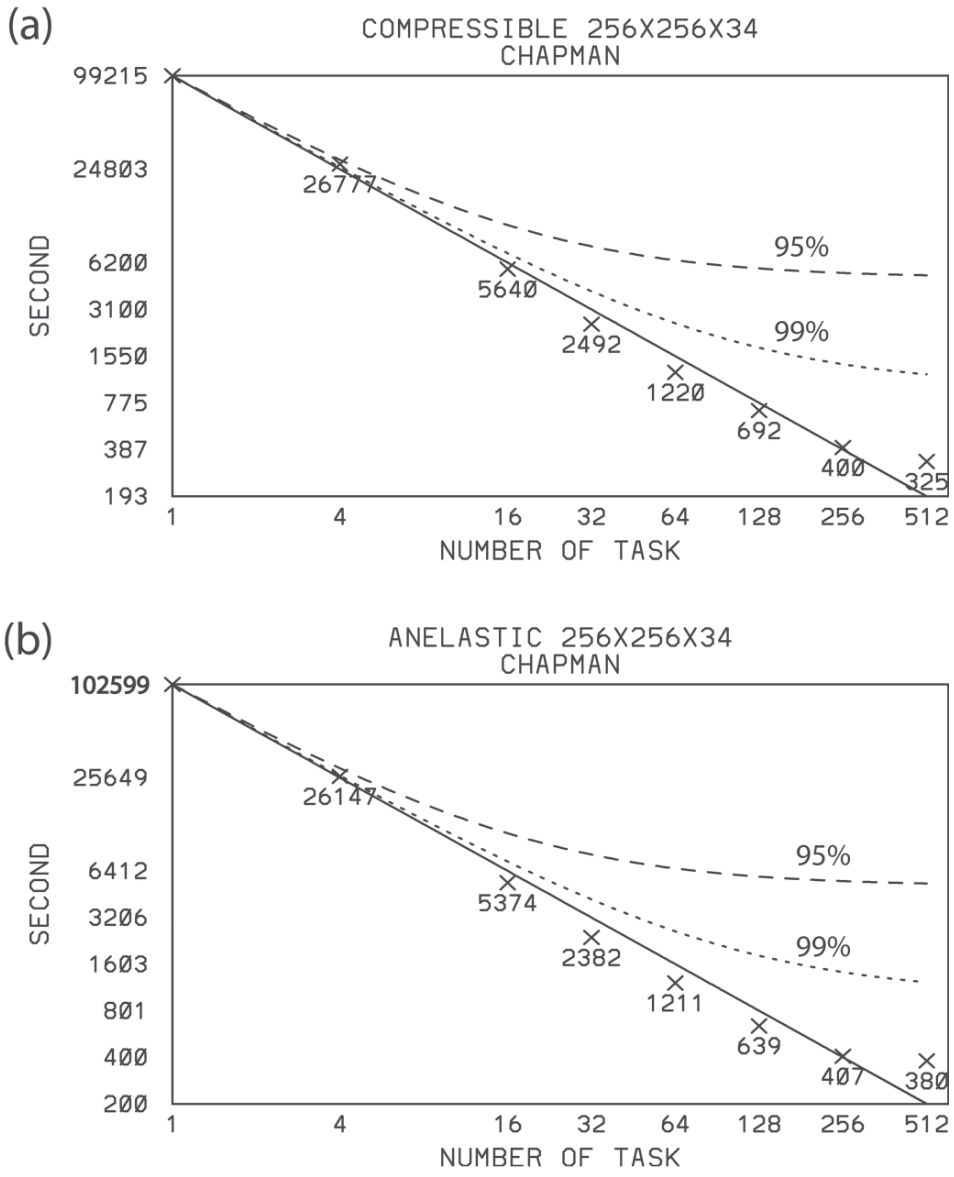

Fig. 8. Same as Fig. 6 except for the CHAPMAN environment.

4.4 Speedup and Efficiency

In addition to wall-clock time, parallel jobs can also be checked in terms of speedup and efficiency. Wall-clock time can show the related speedup for each configuration for a different number of tasks, but different versions running on different machines cannot be compared on

one plot. With speedup and efficiency plots, different versions run on different platforms can be put in one plot, such as in Figs. 9 and 10. The formula for speedup is given as follows:

$$S_p(n) = \frac{T(1)}{T(n)} \quad , \tag{6}$$

and substituting Eq. 5 into Eq. 6 yields the theoretical speedup as:

$$S_p(n) = \frac{T(1)}{T(1)(1-p) + \frac{T(1)p}{n} + c(n)} \quad , \tag{7}$$

where the communication portion $c(n)$ is neglected (as in most published literature) in drawing the theoretical perfect, 99% and 95% speedup in Fig. 9. Following this, Figs. 6, 7, and 8, the solid line indicates perfect speedup, the dotted curve 99%, and the dashed curve 95%.

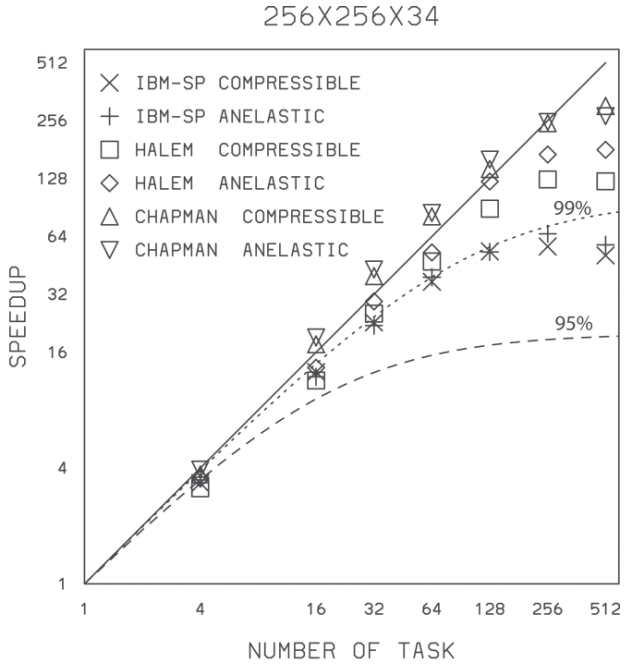

Fig. 9. Speed-up for the compressible and anelastic versions of the model with 256 grid points in the x- and y-directions and 34 in the z-direction on HALEM, the IBM-SP and CHAPMAN. The solid line indicates perfect speedup, the dotted curve 99%, and the dashed curve 95%.

In Fig. 9, compressible and anelastic versions run on the IBM-SP, HALEM, and CHAPMAN are plotted together. Generally speaking, all of the results are along or better than 99% parallelization. It shows clearly that CHAPMAN provides the best speedup for current implementation of MPI in GCM when the number of tasks is larger than 64. The results are less distinctive for a smaller number of tasks, due to the architecture of Origin 2000 with a shared memory system, which significantly reduces communication time. Also, the anelastic version has a better speedup than the compressible on all platforms. Of course, these results are consistent with the results shown in the previous sub-section (Figs. 6, 7, and 8). Though Fig. 9 shows the relative speedup all together, and it is indicated that the anelastic version has a better speedup than the compressible. From the results under current computational environments, if the meteorological performance is nearly equal between them, it is faster in wall time to use the anelastic version on the IBM-SP, the compressible version on HALEM, and any version on CHAPMAN if 256 tasks are available.

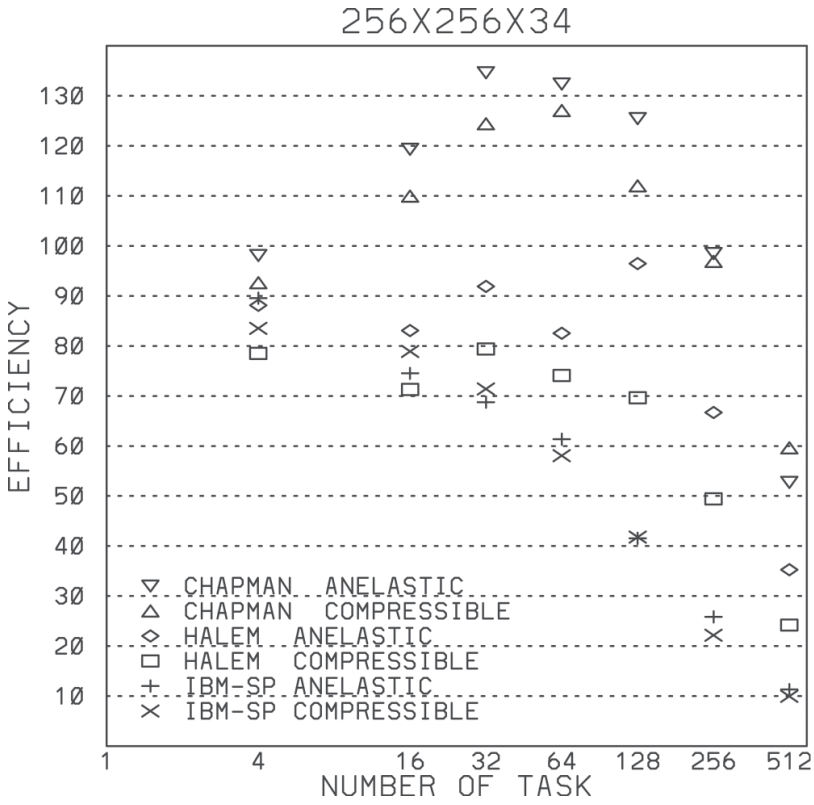

Fig. 10. Efficiency for the compressible and anelastic versions of the model with 256 grid points in the x- and y-directions, and 34 in the z-direction on HALEM, IBM-SP and CHAPMAN.

The efficiency of all versions and platforms can be checked in one plot. The formula for efficiency is:

$$E(n) = \frac{S_p(n)}{n} 100\% \quad , \tag{8}$$

where $E(n)$ is the efficiency with n tasks. Figure 10 show the efficiency of all versions running on all platforms. The results from the IBM-SP show less distinction between the two versions, and efficiency dropping from around 85% with 4 tasks to 10% with 512 tasks. On the contrary, the results from HALEM show more of a difference between the two versions and no consistent drop in efficiency as in the case of the IBM-SP. Though the compressible version on HALEM drops from 80% efficiency with 32 tasks to 24% with 512 tasks, the anelastic version on HALEM has efficiencies of 91% and 97% for 32 and 128 tasks. The special case of the anelastic version on HALEM having close to perfect speedup was shown in Figs. 8b and 9; however, this appears much more clearly in the efficiency plot. Thus it is the most efficient configuration and should be used. Except for 512 tasks, the two versions perform efficiently (above 90%) on CHAPMAN and achieve 120% efficiency with 32 and 64 tasks. Furthermore, in an operational environment, this configuration (128 task anelastic version) should be the choice for routine integrations.

4.5 Different Lengths in Dimensions

So far, only dimensions of 256 × 256 × 34 have been used to test performance. For another case of measurement, wall-clock time will be examined for increasing dimension not increasing resolution. Figure 11 shows wall-clock times by using (a) HALEM, (b) IBM-SP, and (c) CHAPMAN for three different dimensions. The different dimensions are 256 × 256 × 34, 512 × 512 × 34, and 1024 × 1024 × 34; they are used with the compressible version at 128 tasks. The number of grid points in 1024 × 1024 × 34 is four times larger than that of 512 × 512 × 34, which is four times larger than that of 256 × 256 × 34. The wall-clock times are marked with the symbol "X" with the associated times written beneath. The solid curve, again, is the idealized wall-time-spend produced by multiplying the increase in dimensions by the wall-clock time for 256 × 256 × 34. The dotted curve is 80% of the idealized wall-time-spend and the dashed curve is 60%. Thus, the less the percentage is, the less wall time, so the better the performance is.

In Fig. 11a, the wall-clock times from the different dimensions correspond to the idealized cost (solid curve) indicating that the size of the dimensions can be increased up to 16 times without losing or gaining efficiency on HALEM. In Fig. 11b, for the IBM-SP, the results fall below the idealized curve. The result for a 4-time increase in the size of the dimensions is located on the 80% curve, and the result for a 16-time increase is located between the dotted and dashed curves at about 70%. For CHAPMAN, Fig. 11c shows slightly less wall-clock time for 512 × 512 but much more than the idealized wall-clock time for 1024 × 1024. This indicates that the IBM-SP provides the best performance with large dimensions for the model, followed by HALEM, and then CHAPMAN, under current implementation of MPI in GCE.

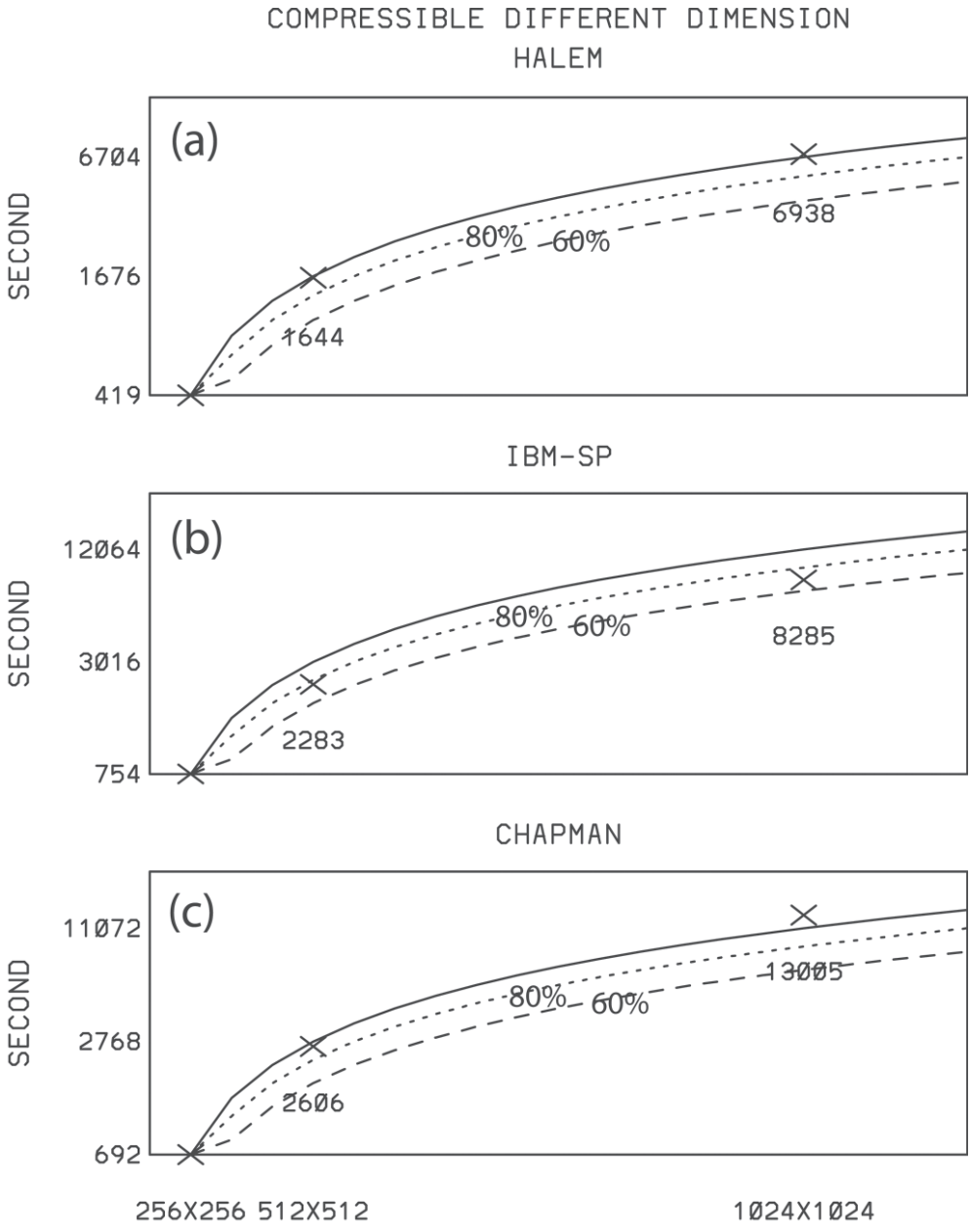

Fig. 11. Wall-clock time in seconds for different dimensions of the model horizontal domain for (a) HALEM, (b) IBM-SP and (c) CHAPMAN for the compressible version of the model. The solid curve indicates wall-clock times linearly extrapolated from the 256 × 256 model domain, the dotted curve indicates 80% of the solid curve and the dashed curve indicates 60% of the solid curve in each environment.

4.6 Reproducibility

If the binary results are the same among different decompositions and/or no decomposition, we call the decomposition reproducible. There are some computations, such as obtaining the mean value from the entire domain that could use a reduced collective MPI call to save wall clock time; however, reduced collective calls may not have the same computational sequence among different decompositions. Thus, a flag was introduced in the code as an option for reproducibility. When the reproducible flag is off in order to save time, the reduced collective MPI call is used, thus binary results are different for different runs as well as different numbers of tasks. However, the difference by using reduced collective MPI calls due to different number of tasks is within truncation error, which is insignificant.

Since rainfall is the most sensitive quantity to be discerned between different model integrations, it will be examined from the results for a 3- to 4-day integration, which is about the averaged integration for GCE. Figure 12 shows total domain rainfall in mm h^{-1} (a) between 0 and 24 h, and (b) between 72 and 76 h of integration using 256 × 256 × 34 grid points. The

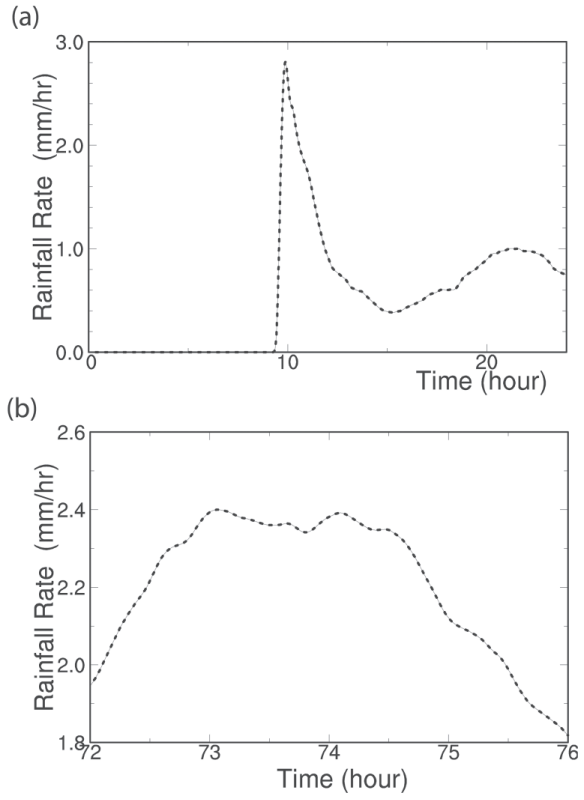

Fig. 12. Rainfall amount over a given domain in mm h^{-1} with respect to time evolution from (a) 00 to 24 and (b) 72 to 76 hr of integration. Solid curve indicates results from a single task and dashed curve from arbitrary multiple tasks.

solid thin curve is from single-cpu integration, and the thick dashed curve is from the non-reproducible option with 64 cpus. Though the binary output between these two experiments is different, the figure shows the difference is un-distinguishable, not only over the initial 24 h of integration but even after 72 h of integration. Figure 13 shows the accumulated rainfall over the domain for a) 1 cpu and b) 64 cpus. The patterns and values are very similar. Furthermore, rainfall is the most significant output for this type of model and thus the most representative; even the vertical velocity and other thermodynamic properties are closely related to precipitation/rainfall.

5. SUMMARY

The concept of parallelization using MPI with little modification to the original model code as well as having flexibility and reproducibility running any number of tasks, including single-cpu, has been adopted for GCE modeling. The GCE model has two different dynamic options: compressible and anelastic. Both versions are decomposed in grid point space in a similar manner and require data exchange for MPI to update the halo regions and lateral boundaries. In order to have all of the grid points in any given direction available for FFT in the anelastic version, a MPI transpose method similar to the data exchange is implemented into the anelastic version to solve the pressure derivative in the horizontal. A preprocessor written in the C language is adopted into the original code to manage the model options, so that the model can be run with a variety of different versions, decompositions and numbers of tasks.

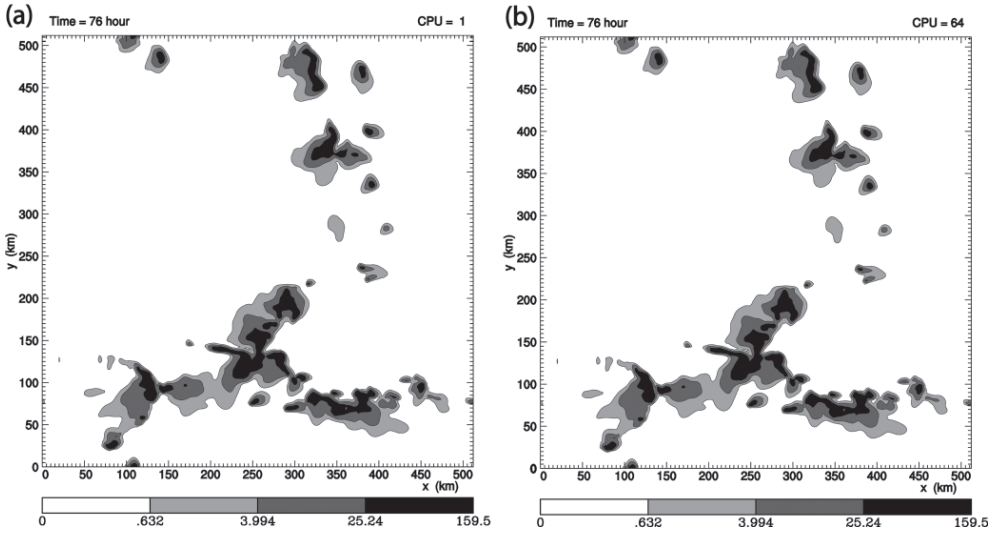

Fig. 13. Accumulated rainfall amount in mm h^{-1} after 76 hr of integration for results from (a) 1 cpu and (b) 64 tasks on HALEM.

Since there are many options available, several sensitivity tests were conducted in order to obtain the optimal values for performance. Three platforms, HALEM (Dec alpha), the IBM SP (power4) and CHAPMAN (SGI Origin 2000), were selected to test the performance of the parallelization using MPI. Repeated runs using the same configuration on different platforms showed about a 5% difference in wall-clock time. Long- and short-term runs have about the same ratio of wall-clock time to forecast time, indicating the performance of the implementation on these platforms is quite consistent.

The performance of the different decompositions reconfirms the theoretical concept that the best performance is with 2D decomposition using equal numbers of columns and rows. The worst performance is with 1D decomposition with one row sliced into columns. These results are mainly due to the smaller amount of data exchange in 2D decomposition and the x-direction grid length in the inner loop. Both versions of the model have wall-clock times and speedups that are around 99% of the theoretical curves up to 256 tasks but not for 512 tasks. This implies that for dimensions of $256 \times 256 \times 34$, the speedup saturates at about 256 tasks or more precisely about 128 tasks. The anelastic version has better speedup and efficiency compared to the compressible version due to the greater number of computations. There is a highly efficient configuration for the anelastic system with 128 tasks on HALEM that produces wall-clock times that are about the same as the compressible version with the same number of tasks. There is no way to compare and conclude the performance of these three machines by using one model and it is not our intention to do so. However, from our limited number of experiments here, GCE has the best speedup and efficiency for dimensions less than $256 \times 256 \times 34$ in CHAPMAN, and GCE's performance using large dimensions with 128 tasks is the best in IBM-SP among these three machines. Again, three machines are arbitrarily selected by their availability to demonstrate the flexibility of the MPI implementation into GCE, they are not used for performance comparison in general.

Even though the performance of the current parallelization with MPI is reasonably good, there are still some portions of the code that can be improved using MPI, for example input and output (IO). MPI-IO was not included in the first version of MPI (Gropp et al. 1999a). Most IO either uses direct access, an IO server with one IO task, or writes out each portion into a file, then all files (all portions) are collected by another program. MPI-2 (Gropp et al. 1999b) has several advanced MPI routines including MPI-IO. However, it was not implemented into the current GCE model. The GCE model will need further parallelization with MPI-2 in the future. Nevertheless, the concept of the current parallelization by using MPI with a C preprocessor provides a viable mechanism for model development, which gives the GCE model the capability to plug/un-plug MPI and/or any future package to easily keep up with rapidly changing computer architectures.

Acknowledgements This work is supported both by the Atmospheric Dynamics and Thermodynamics Program under NASA headquarters and by the NASA Tropical Rainfall Measuring Mission (TRMM). The authors are grateful to Dr. R. Kakar at NASA headquarters for his support of Goddard Cumulus Ensemble (GCE) model development. Acknowledgement is also made to Dr. T. Lee at NASA HQ and Dr. T. Clune at NASA/NCCS. The NASA/Goddard Space Flight Center, NCEP, and AMES are acknowledged for computer time used in this research.

REFERENCES

Anderson, J. R., K. K. Droegemeier, and R. B. Wilhelmson, 1985: Simulation of the thunderstorm subcloud environment. Preprint, 14th Conf. on Severe Local Storms, 147-150.

Aoyama, Y., and J. Nakano, 1999: RS/6000 SP: practical MPI programming. IBM International Technical Support Organization, SG24-5380-00, 221 pp.

Baker, R. D., B. H. Lynn, A. Boone, W. K. Tao, and J. Simpson, 2001: The influence of soil moisture, coastline curvature, and the land-breeze circulation on sea-breeze initiated precipitation. *J. Hydrometeor.*, **2**, 193-211.

Chen, J. P., and D. Lamb, 1999: Simulation of cloud microphysical and chemical processes using a multi-component framework. Part II: Microphysical evolution of a wintertime orographic cloud. *J. Atmos. Sci.*, **56**, 2293-2312.

Chou, M. D., and M. J. Suarez, 1999: A shortwave radiation parameterization for atmospheric studies. 15, NASA/TM-104606, 40 pp.

Chou, M. D., K. T. Lee, S. C. Tsay, and Q. Fu, 1999: Parameterization for cloud longwave scattering for use in atmospheric models. *J. Climate*, **12**, 159-169.

Estrade, J. F., Y. Trémolet, J. Sela, 2001: Experiments with NCEP's spectral model. Developments in Tera Computing, proceedings of the ninth ECMWF workshop on the use of high performance computing in Meteorology, Reading, UK, 13 - 17 November, 2000, 92-99.

Ferrier, B. S., 1994: A double-moment multiple-phase four-class bulk ice scheme. Part I: Description. *J. Atmos. Sci.*, **51**, 249-280.

Ferrier, B. S., W. K. Tao, and J. Simpson, 1995: A double-moment multiple-phase four-class bulk ice scheme. Part II: Simulations of convective storms in different large-scale environments and comparisons with other bulk parameterizations. *J. Atmos. Sci.*, **52**, 1001-1033.

GCSS (GEWEX Cloud System Study), 1993: *Bull. Amer. Meteor. Soc.*, **74**, 387-400.

Grabowski, W. W., and M. W. Moncrieff, 2001: Large-scale organization of tropical convection in two-dimensional explicit numerical simulations. *Quart. J. R. Meteor. Soc.*, **127**, 445-468.

Gropp, W., E. Lusk, and A. Skjellum, 1999a: Using MPI: portable parallel programming with the message-passing interface, 2nd Ed., The MIT Press, Cambridge, Massachusetts, 371 pp.

Gropp, W., E. Lusk, and R. Thakur, 1999b: Using MPI-2: advanced features of the message-passing interface. The MIT Press, Cambridge, Massachusetts, 382 pp.

Ikawa, M., 1988: Comparison of some schemes for non hydrostatic models with orography. *J. Meteor. Soc. Japan*, **66**, 753-776.

Johnson, D., W. K. Tao, J. Simpson, and C. H. Sui, 2002: A study of the response of deep tropical clouds to large-scale processes, Part I: Model set-up strategy and comparison with observation. *J. Atmos. Sci.*, **59**, 3492-3518.

Johnson, K. W., J. Bauer, G. A. Riccardi, K. K. Droegemeier, and M. Xue, 1994: Distributed processing of a regional prediction model. *Mon. Wea. Rev.*, **122**, 2558-2572.

Juang, H.-M. H., and M. Kanamitsu, 2001: The computational performance of the NCEP

seasonal forecast model on FUJITSU VPP5000 at ECMWF. Developments in Tera Computing, proceedings of the ninth ECMWF workshop on the use of high performance computing in Meteorology, Reading, UK, 13 - 17 November, 2000, 338-347.

Juang, H.-M. H., C. H., Shiao, and M. D. Cheng, 2003: The Taiwan Central Weather Bureau Regional Spectral Model for seasonal prediction: Multi-parallel implementation and preliminary results. *Mon. Wea. Rev.*, **131**, 1832-1847.

Khain, A. P., M. Ovtchinnikov, M. Pinsky, A. Pokrovsky, and H. Krugliak, 2000: Notes on the state-of-the-art numerical modeling of cloud microphysics. *Atmosph. Res.*, **55**, 159-224.

Klemp, J. B., and R. Wilhelmson, 1978: The simulation of three-dimensional convective storm dynamics. *J. Atmos. Sci.*, **35**, 1070-1096.

Lang, S., W. K. Tao, J. Simpson, and B. Ferrier, 2003: Modeling of convective-stratiform precipitation processes: Sensitivity to partitioning methods. *J. Appl. Meteor.*, **42**, 505-527.

Lin, Y. L., R. D. Farley, and H. D. Orville, 1983: Bulk parameterization of the snow field in a cloud model. *J. Clim. Appl. Meteor.*, **22**, 1065-1092.

Lynn, B. H., W. K. Tao, and P. J. Wetzel, 1998: A study of landscape generated deep moist convection. *Mon. Wea. Rev.*, **126**, 928-942.

Lynn, B. H., and W. K. Tao, 2001: A parameterization for the triggering of landscape generated moist convection, Part II: Zero order and first order closure. *J. Atmos. Sci.*, **58**, 593-607.

Lynn, B. H., W. K. Tao, and F. Abramopoulos, 2001: A parameterization for the triggering of landscape generated moist convection, Part I: Analyses of high resolution model results. *J. Atmos. Sci.*, **58**, 575-592.

Marshall, J., A. Adcroft, C. Hill, L. Perelman, and C. Heisey, 1997: A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *J. Geophys. Res.*, **102**, 5753-5766.

Michalakes, J., S. Chen, J. Dudhia, L. Hart, J. Klemp, J. Middlecoff, and W. Skamarock, 2001: Development of a next-generation regional weather research and forecast model. Developments in Tera Computing, proceedings of the ninth ECMWF workshop on the use of high performance computing in Meteorology, Reading, UK, 13 - 17 November, 2000, 269-276.

Ogura, Y., and N. A. Phillips, 1962: Scale analysis of deep and shallow convection in the atmosphere. *J. Atmos. Sci.*, **19**, 173-179.

Purnell, D. K., and M. J. Revell, 1995: Field-object design of a numerical weather prediction model for uni- and multiprocessors. *Mon. Wea. Rev.*, **123**, 401-429.

Randall, D. A., J. Curry, P. Duynkerke, S. K. Krueger, M. W. Moncrieff, B. Ryan, D. O. Starr, M. Miller, W. Rossow, G. Tseliudis, and B. A. Wielikci, 2003: The GEWEX cloud system study: A view from 2001. *Bull. Amer. Meteor. Soc.*, **84**, 455-469.

Rutledge, S. A., and P. V. Hobbs, 1984: The mesoscale and microscale structure and organization of clouds and precipitation in mid-latitude clouds. Part XII: A diagnostic modeling study of precipitation development in narrow cold frontal rainbands. *J. Atmos. Sci.*, **41**, 2949-2972.

Skalin, R., 1997a: Scalability of parallel gridpoint limited-area atmospheric models. Part I: Explicit time-integration schemes. *J. Atmos. Ocea. Technol.*, **14**, 427-441.

Skalin, R., 1997b: Scalability of parallel gridpoint limited-area atmospheric models. Part II: Semi-implicit time-integration scheme. *J. Atmos. Ocean. Technol.*, **14**, 442-455.

Smolarkiewicz, P. K., and W. W. Grabowski, 1990: The multidimensional positive advection transport algorithm: Nonoscillatory option. *J. Comput. Phys.*, **86**, 355-375.

Soong, S. T., and Y. Ogura, 1980: Response of tradewind cumuli to large-scale processes. *J. Atmos. Sci.*, **37**, 2035-2050.

Tao, W. K., and J. Simpson, 1993: The Goddard Cumulus Ensemble Model. Part I: Model description. *Terr. Atmos. Ocean. Sci.*, **4**, 19-54.

Tao, W. K., J. Simpson, C. H. Sui, B. Ferrier, S. Lang, J. Scala, M. D. Chou, and K. Pickering, 1993: Heating, moisture and water budgets of tropical and mid-latitude squall lines: Comparisons and sensitivity to longwave radiation. *J. Atmos. Sci.*, **50**, 673-690.

Tao, W. K., S. Lang, J. Simpson, C. H. Sui, B. Ferrier, and M. D. Chou, 1996: Mechanisms of Cloud-radiation interaction in the tropics and midlatitudes. *J. Atmos. Sci.*, **53**, 2624-2651.

Tao, W. K., 2003: Goddard Cumulus Ensemble (GCE) model: Application for understanding precipitation processes, AMS Meteorological Monographs - Cloud Systems, Hurricanes and TRMM, 103-138.

Tao, W. K., C. L. Shie, R. Johnson, S. Braun, J. Simpson, and P. E. Ciesielski, 2003: Convective Systems over South China Sea: Cloud-Resolving Model Simulations. *J. Atmos. Sci.*, **60**, 2929-2956.

Wang, Y., W. K. Tao, J. Simpson, and S. Lang, 2003: The sensitivity of tropical squall lines (GATE and TOGA COARE) to surface fluxes: 3-D Cloud resolving model simulations. *Quart. J. R. Met. Soc.*, **129**, 987-1007.

Wu, X., W. W. Grabowski, and M. W. Moncrieff, 1998: Long-term behavior of cloud systems in TOGA COARE and their interactions with radiative and surface processes. Part I: Two-dimensional modeling study. *J. Atmos. Sci.*, **55**, 2693-2714.

Juang, H.-M. H., W. K. Tao, X. Zeng, C. L. Shie, S. Lang, and J. Simpson, 2007: Parallelization of the NASA Goddard cumulus ensemble model for massively parallel computing. *Terr. Atmos. Ocean. Sci.*, **18**, 593-622, doi: 10.3319/TAO.2007.18.3.593(A).