Scalability Improvement of the NASA Multiscale Modeling Framework for Tropical Cyclone Climate Study

Bo-Wen Shen[1,2] , Bron Nelson[3], Samson Cheung[3], and Wei-Kuo Tao[2]

Dr. Bo-Wen Shen,

Earth System Science Interdisciplinary Center (ESSIC)

University of Maryland at College Park (UMCP), and

Mesoscale Atmospheric Processes Laboratory, 612

NASA Goddard Space Flight Center, Greenbelt, MD 20771

email: Bo-Wen.Shen-1@nasa.gov

http://tiny.cc/bws6u

[1] UMCP/Earth System Science Interdisciplinary Center (ESSIC)

[2] NASA/Goddard Space Flight Center (GSFC)

[3]NASA Ames Research Center (ARC)

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

Abstract

One of the current challenges in tropical cyclone (TC)[1] research is how to improve our understanding of TC interannual variability and the impact of climate change on TCs. Recent advances in global modeling, visualization, and supercomputing technologies at NASA show potential for such studies. In this study, we discuss recent scalability improvement to the Multiscale Modeling Framework (MMF) that makes it feasible to perform long-term TC-resolving simulations. The MMF consists of the finite-volume General Circulation model (fvGCM), supplemented by a copy of the Goddard Cumulus Ensemble model (GCE) at each of the fvGCM grid points, giving 13,104 copies of GCEs. The original fvGCM implementation has a 1D data decomposition. The revised MMF implementation retains the 1D decomposition for most of the code, but uses a 2D decomposition for the massive copies of GCEs which uses a second level of parallelism to execute it. Since the vast majority of computation time in the MMF is spent computing the GCEs, this approach can achieve excellent speedup without incurring the cost of modifying the entire code. Intelligent process mapping allows differing numbers of processes to be assigned to each domain for load balancing. The revised parallel implementation shows very promising scalability, obtaining a nearly 80-fold speedup by increasing the number of cores from 30 to 3,335. Future work will be discussed in the concluding remarks.

[1] Depending on their location, TCs are referred to by other names, such as hurricane (in the Atlantic region), typhoon (in the West Pacific region), tropical storm, cyclonic storm, and tropical depression.

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

1. Introduction

Studies in TC interannual variability and the impact of climate change (e.g., global warming) on TCs have received increasing attention, particularly due to the fact that 2004 and 2005 were the most active hurricane seasons in the Atlantic, while 2006 was not as active as predicted. In addition, during the past ten years, statistics have shown that hurricanes are the deadliest weather events in the US (http://www.nws.noaa.gov/os/hazstats.shtml). Therefore, an urgent need exists to improve short-term and long-term hurricane forecasts.

It is well known that hurricane dynamics involve multiscale processes. To accurately simulate the evolution of a TC at the mesoscale (medium-scale), an ideal solution is to improve the model to realistically capture (1) the downscaling processes associated with the modulation by the large-scale flows, such as Madden-Julian Oscillations (MJO) or tropical waves[2], and (2) the upscaling process associated with the feedback by small-scale processes such as convection or precipitation. However, it has been challenging to accurately simulate TC activities with traditional numerical models because of artificial scale separations. For example, Earth (atmospheric) modeling activities have been conventionally divided into three major categories: global-scale (large-scale), meso-scale, and cloud (micro)-scale. As shown in Figure 1, typical resolutions for the global, regional, and cloud models are on the order of 100km, 10km, and 1km, respectively.

Historically, partly due to limited access to computing resources, TC climate studies have been conducted mainly with coarse-resolution general circulation models (GCMs) (Bengtsson et al., 2007), and partially with fine-resolution regional mesoscale models (MMs). The former have

[2] By examining the role of different tropical waves in tropical cyclogenesis with observations, Frank and Roundy (2006) found a strong relationship between TC formation and enhanced activity in equatorial Rossby waves, mixed Rossby gravity waves, TD-type disturbances (or easterly waves), or the MJOs. With a 45- to 60-day time scale, eastward-propagating MJOs, which are typically characterized by deep convection originating over the Indian Ocean, have one of the most prominent large-scale features of the tropical general circulation. These large-scale tropical systems may provide determinism in regards to the timing and location of TC genesis, which can be viewed as a pre-cursor. Thus, it becomes possible to extend the lead time for TC genesis prediction and thus increase our confidence in the model's ability to simulate TC climate as long as the model can improve the simulations of the precursor and its modulation on TC activities. More information on modeling the association between the TC genesis and these large-scale tropical systems can be found in Shen et al. (2011;2012a,b).

the advantage of simulating the impact of global large-scale flows on TC activities, but may not be able to accurately simulate mesoscale and small-scale flows. In contrast, the latter make it possible to simulate realistic TC intensity and structure with fine grid spacing, but this approach may still have difficulties in capturing the accurate impact of large spatial- and temporal- scale flow. Furthermore, for TC climate studies, the resolutions that were previously used in GCMs and MMs were still too coarse to resolve small-scale convective motion, and therefore "cumulus parameterizations" (CPs) were required to emulate the effects of unresolved subgrid-scale cloud motion. Because the development of CPs has been slow, their performance is a major limiting factor in TC simulations. To overcome the issues with CPs, a recent trend is to take full advantage of modern supercomputing power to explicitly resolve the impact of clouds in a global environment. Two approaches are to (1) deploy high-resolution global models (e.g., the global mesoscale model in Shen et al. 2006, 2010, 2011, 2012a,b) and (2) couple a coarse-resolution global model with massive copies of cloud models. The latter approach is known as super-parameterization or Multiscale Modeling Framework (MMF, Randall et al., 2003; Tao et al., 2008, 2009). Recently, we have successfully integrated these models with visualizations into the Coupled Advanced multiscale Modeling and Visualization system (CAMVis) on NASA's Pleiades supercomputer (Shen et al., 2011, 2012a,b), showing promise in short-term and extended-range TC simulations.

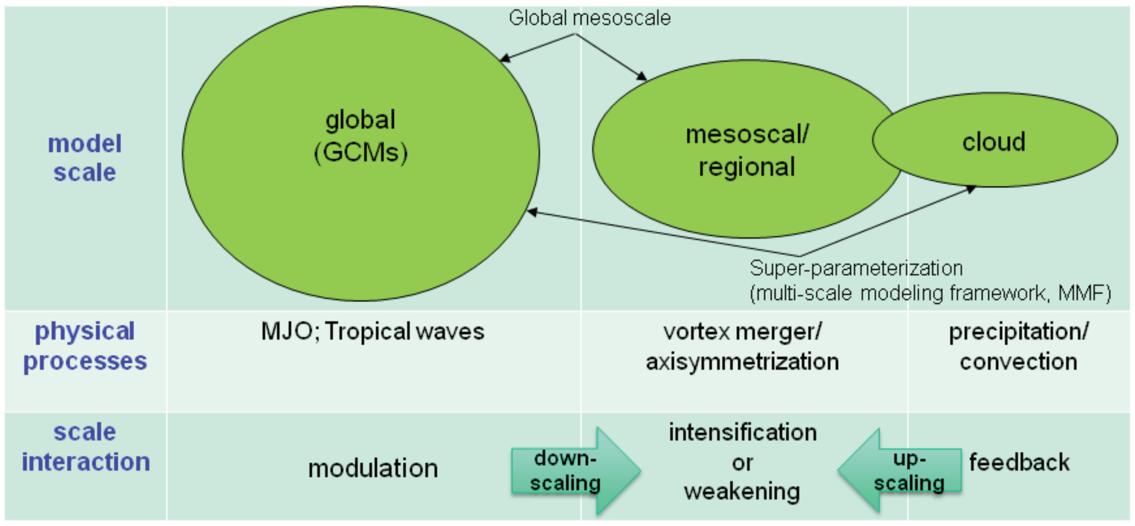


Figure 1: *The three major categories of Earth atmospheric modeling (from left to right): global-scale (large-scale), meso-scale (medium-scale), and cloud (micro)-scale. Typical resolutions for these models are about 100km, 10km, and 1km, respectively.*

In the MMF, the conventional CP at each grid point of a GCM is replaced by a copy of cloud-resolving model (CRM), which is used to accurately represent non-hydrostatic, cloud-scale convection and its interaction with environmental flows. Therefore, the MMF has the advantages of both the large- and global-scale processes of a GCM and the sophisticated microphysical processes of a CRM, and can be viewed as an alternative to a global CRM. However, this approach poses great challenges with computing resources, data storage, and data analyses for multi-decadal simulations of global TC activities. These challenges include but are not limited to (1) running more than 10,000 instantiations of the CRM with great parallel performance; (2) increasing the GCM's resolution for capturing realistic TC structure; (3) efficiently archiving massive volumes of data; (4) effectively conducting analyses in order to discover the predictive relationship between the meteorological (short-term) and climatological (long-term) events. To address the first two issues, we propose a revised MMF coupling approach to improve the model's scalability, enabling higher resolutions in both the GCM and CRM, reducing time to finish TC climate simulations.

In section 2, we introduce the advanced global- and cloud-scale models at NASA, including some details on the parallelism in these models. We discuss the implementation of the revised parallelism to improve the performance of the MMF in sections 3.1 and 3.2, and present the benchmark with the improved MMF on the NASA Pleiades supercomputer in section 3.3. We conclude with a summary and discussion of future plans in section 4. To facilitate discussions, the following convention is adopted: the name of each routine ends with parentheses (), and the the name of each variable is in italics. The term "task" and "process" are used interchangeably. The former emphasizes the work being done, while the latter emphasizes the worker doing it. For example, task A is performed using the process A.

2. The NASA Multiscale Modeling Framework (MMF)

The Goddard MMF (Tao et al., 2008, 2009) is based on the NASA Goddard finite-volume GCM (fvGCM) and the Goddard Cumulus Ensemble model (GCE). While the high-resolution fvGCM has shown remarkable capabilities in simulating large-scale flows, and thus hurricane tracks (Atlas et al., 2005; Shen et al. 2006, 2010, 2011, 2012a,b), the GCE is well known for its superior performance in representing small, cloud-scale motions and has been used to produce more than 100 referred journal papers (Tao et al., 1993, 2003). In the MMF, the fvGCM is running at a coarse (2^ox2.5^o) resolution, and one copy of the GCE is running within each of the fvGCM grids (Figure 1 of Tao et al. 2009). The 2^ox2.5^o resolution is still widely used in climate simulations with conventional climate models and recent MMFs because it is computationally affordable. This resolution has grid points of (91, 144) in the (y, x) directions, giving a total of 91x144 (13,104) horizontal cells. Thus, 13,104 GCEs are "embedded" in the fvGCM to allow explicit simulation of cloud processes in a global environment. Currently, only averaged thermodynamic fields such as temperature and water vapor in the GCEs are fed back to the fvGCM, which is called "thermodynamic feedback." The time step for the individual 2D GCE is ten seconds, and the fvGCM-GCE coupling interval is one hour at this resolution. Under this configuration, 99% or more of the total wall-time for running the MMF is spent on the GCEs. Thus, wall-time could be significantly reduced by efficiently distributing the large number of GCEs over a massive number of processors on a supercomputer. We next introduce computational perspective of the GCE and fvGCM, and then discuss a revised strategy for coupling the latter with massive copies of the GCE to improve scalability.

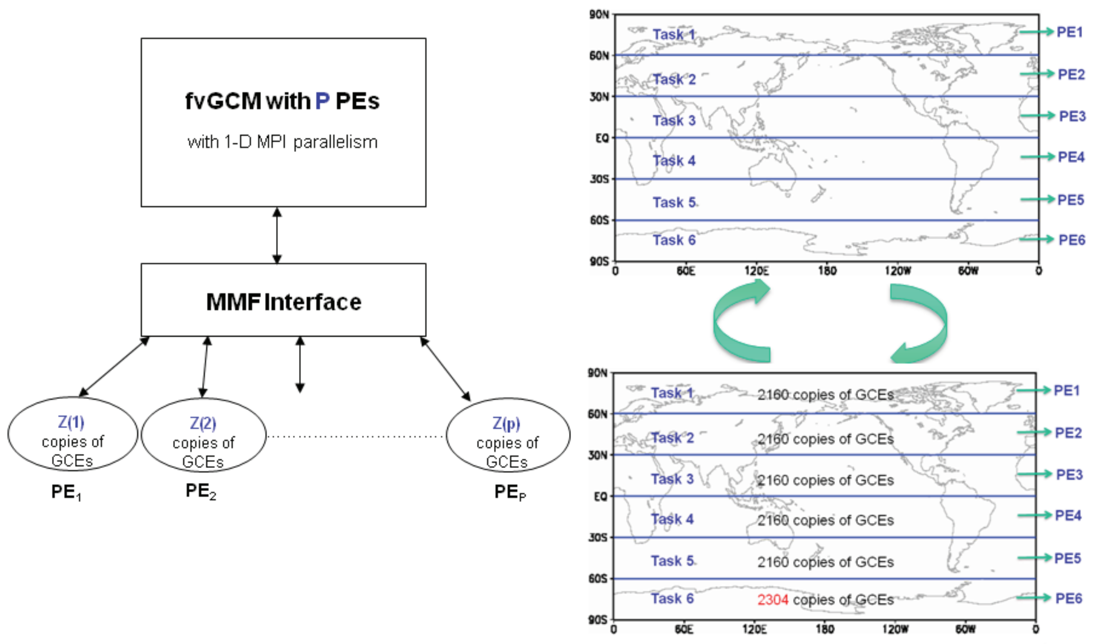
2.1 The Goddard Cumulus Ensemble (GCE) Model

Over the last two decades, the GCE has been developed, extensively tested, and continuously improved. Its main features were described in detail in Tao et. al., (1993, 2003). Typical model runtime configurations for MMF runs are: (a) 64 grid points in the x direction with a grid spacing of 4 km; (b) 32 vertical stretched levels; (c) cyclic lateral boundary conditions; and (d) a time step of 10 seconds. The GCE itself has been implemented with a 2D domain decomposition using MPI-1 (Message Passing Interface version 1) with good parallel efficiency (Juang et al. 2007). Thus, an ideal solution for the course-grain parallelism implemented in the MMF is to run more copies of GCEs with higher CPU counts in parallel, while still keeping the option of taking advantage of the fine-grain parallelism inside the GCE.

2.2. The finite-volume General Circulation Model (fvGCM)

Resulting from a development effort of more than 15 years, the fvGCM is a unified numerical weather prediction (NWP) and climate model that can run on daily, monthly, decadal, or century time-scales (Lin et al., 2003; Atlas et al. 2005). The 1990s model was originally designed for climate studies at a coarse resolution of about 2x2.5 degrees, and its resolution was increased to 1 degree in 2000 and 1/2 degree in 2002 for NWP. Since 2005, the ultra-high-resolution (i.e., 1/8 and 1/12 degree) fvGCM has been deployed on NASA's Columbia supercomputer, showing remarkable TC forecasts (Shen et al. 2006).

The parallelization of the fvGCM was carefully designed to achieve efficiency, parallel scalability, flexibility, and portability. Its implementation had distributed- and shared-memory, two-level parallelism, including a coarse grain parallelism with MPI[3] (MPI-1, MPI-2, MLP, or SHMEM) and fine-grain parallelism with OpenMP (Putman et al. 2005). However, the latter is not applicable for the current MMF runs. Because MPI parallelism is applied to a 1D decomposition over latitude, and each MPI task needs at least three grid points in latitude for parallel efficiency reasons, the MPI parallelism is very limited for small computational grids (Figure 2a). Assume NY is the number of grid points in the y direction. In general, NY is equal to (180^o/DY + 1), where DY is an increment in latitude that can be 0.08^o, 0.125^o, 0.25^o, $0.5^{\ o}$, 1^o, or 2^o. For the 2x2.5 degree grids where NY=91 (with DY=2), the maximum of MPI tasks is only 30 (~91/3). In addition, since NY is generally not divisible by the number of selected processes (P), the domain decomposition is non-uniform, which leads to load unbalancing because some MPI processes are given more latitudes than the other. This requires a special treatment to handle load balancing.



[3] To simplify discussion in this article, the term "MPI" used along with the fvGCM will be referred to as any one of MPI-1,MPI-2,MLP, or SHMEM communication paradigms.

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

Figure 2: *Parallelism in the fvGCM and the MMF v1.0. The fvGCM has the MPI parallel implementation with a 1D domain decomposition along the y direction. The approach of embedding one copy of GCE into each of the fvGCM's grids inherits the parallelism and thus limited scalability of the fvGCM. PE stands for processing element. The right panels show the distribution of tasks over six PEs. Task J (here, J=1~5) performs 2,160 copies of GCEs, while Task 6 has 2304 copies of GCEs.* $\sum_{J=1}^{P} Z(J) = 13{,}104$ *with P=6 in this case.*

3. Revised parallelism in the MMF

3.1 The meta-global GCE (mgGCE) and 2D Domain Decomposition

In the MMF version 1, each of 13,104 GCEs is embedded into a grid cell of the fvGCM. Although the implementation of this version is straightforward, by adopting the parallel framework of the fvGCM, it inherits both the advantages and disadvantages of the fvGCM framework. One of the disadvantages is that this approach limits the parallel scalability of the MMF for small domain grids, which requires many copies of GCEs to run within one PE. For example, 2,160 copies of GCE are running in series in one PE when only 6 PEs are used, as shown in Figure 2, above. Note that load unbalancing due to more copies of GCEs in some of tasks (e.g., Task 6 in Figure 2) will later be addressed in section 3.2. Assume that the maximum number PEs (30) is used, each PE performs one copy of GCE at a time and needs to run 432 copies of GCEs sequentially. Therefore, making more copies of GCEs to run in parallel with more PEs is the key to reducing the wall-clock time for running. To achieve these goals, a different strategic approach is proposed to couple the fvGCM and GCEs.

From a computational perspective, the concept of ``embedded GCEs into the fvGCM'' restricts the MMF's view of the parallelism; namely it can only inherit the parallelism from the fvGCM. Because a GCE uses a periodic, lateral boundary condition, execution of each embedded GCE is independent of the other GCEs during a timestep of the fvGCM model, and thus the GCE can be run in a separate MPI task. Accordingly, we propose a new coupling approach to improve the MMF's parallel scalability. Conceptually, we refer to the 13,104 (144x91) copies of GCEs as a super-component called meta-global GCE (mgGCE) in a meta

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

grid-point system. To facilitate discussion, this grid system is assumed to be the same as the latitude-longitude grid structure in the fvGCM, although it is not necessarily tied to any specific grid system.

With this concept in mind, each of the two individual components (the fvGCM and mgGCE) in the MMF could have its own domain decompositions. Note that as cyclic, lateral boundary conditions are used in each GCE, there is no data communication between any two GCEs. In other words, the mgGCE has no ghost region. Thus, a 2D domain decomposition in the mgGCE can significantly reduce the run-time for massive copies of the GCE. This can greatly help improve MMF's scalability, as most of wall-time is spent on these GCEs.

With the aim of achieving the aforementioned functionalities, the implementation of the data parallelism to couple the fvGCM and mgGCE are briefly described as follows: (i) creation of two groups of processes with the MPI intercommunication, one group with P fvGCM processes and the other with Q mgGCE processes; (ii) sophisticated static mapping between the P fvGCM and Q mgGCE processes; and (iii) dynamic distribution of input values to mgGCE processes from fvGCM processes that can finish execution of 13,104 copies of GCEs and handle load balancing. *This implementation indeed leads to an effective 2D domain decomposition in the mgGCM, while the original 1D domain decomposition remains in the fvGCM. Simply speaking, the first-level parallelism is implemented to decompose latitudes in fvGCM, the second-level parallelism is implemented to decompose longitudes in mgGCE. The schematic diagram of these domain decompositions is shown in Figure 3, and the technical details are discussed in the next sections.*

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

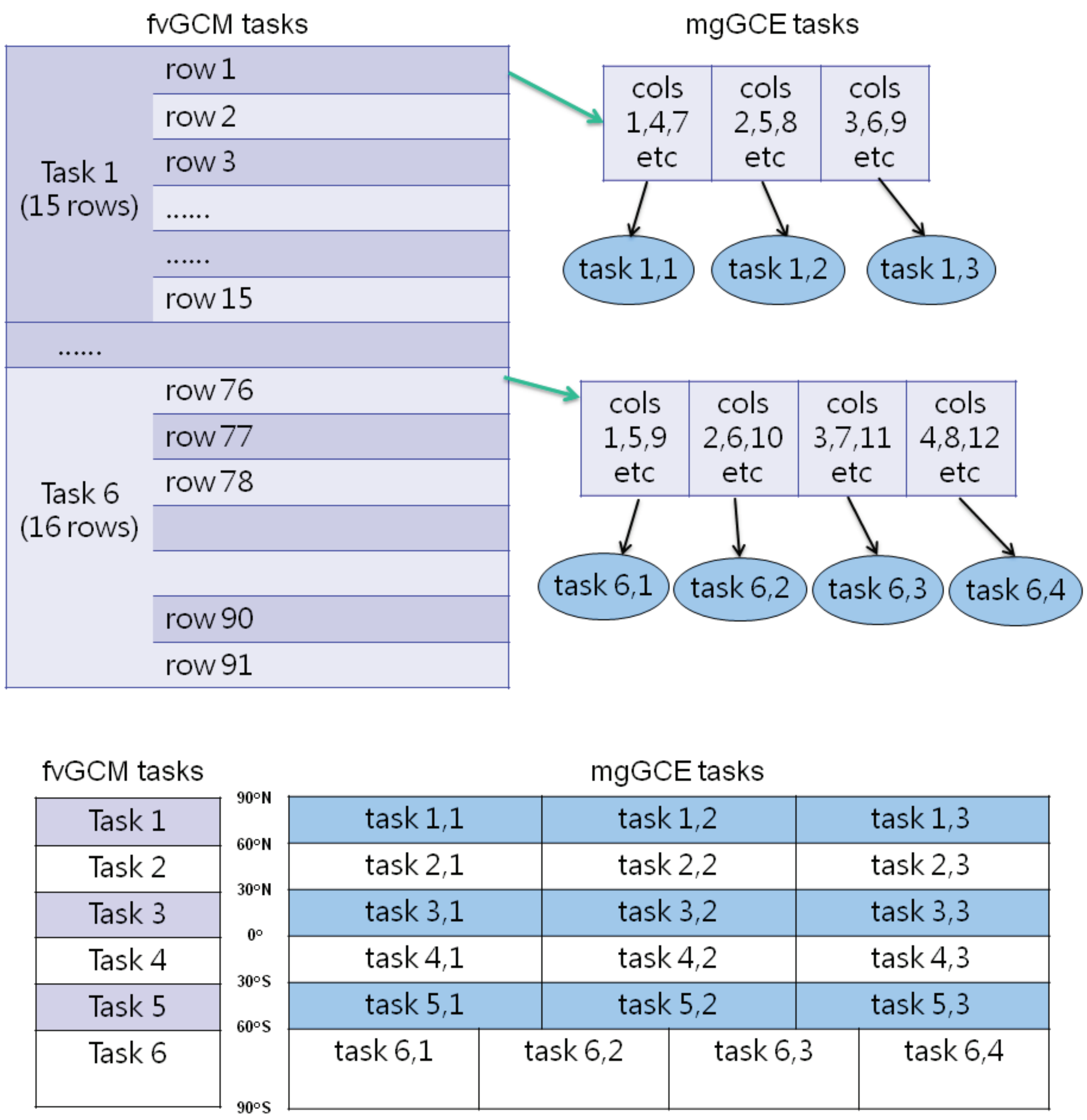


Figure 3: A conceptual diagram of the revised parallelism implementation using 6 fvGCM tasks and 19 mgGCE tasks. In the original 1-D decomposition, each fvGCM process (top left panel) gets three or more latitudes (or rows), and computes the mgGCEs for each longitude (or column) sequentially within a row. In the new version, a second level of parallelism is implemented, decomposing a single row into individual columns (top right panel). "cols 1,4,7 etc" refers to the 1st, 4th 7th columns. Each fvGCM process has an associated set of mgGCE processes that compute the GCEs for a row in parallel. More mgGCE processes are assigned to the fvGCM process with more rows (i.e., more copies of GCEs). For example, fvGCM Task 6 has four

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

mgGCE processes while the other fvGCM tasks have only three mgGCE processes. Note that this second level is only for the mgGCE computation. The 13,104 copies of GCEs are distributed over these mgGCE processes (and the fvGCM processes as needed), achieving an effective 2D domain decomposition that is shown in the bottom panel where the horizontal axis is no longer the longitude. The technical details of this implementation are discussed in section 3.2. This revised parallelism can improve the scalability of MMF by allowing more copies of GCEs running in parallel, and thus reduces wall-time significantly because executing the GCEs costs 99% or more of wall-time for an MMF run.

3.2 Parallel Implementation

As discussed in the previous section, the parallelism of fvGCM was implemented with MPI communication that allows point-to-point and/or collective communication between processes in the same group. This kind of "conventional" communication, which also appears in many existing MPI simple-program multiple-data codes, is defined as an intra-communication. In contrast, an inter-communication is a communication between processes in different groups, consisting of a local group and a remote group. A local group is defined as the group within which a process initiates an inter-communication operation. Specifically, the local group is the sender (receiver) in a send (receive) call. A remote group is defined as the group that contains the target process, which is the receiver (sender) in a send (receive) call. Note that these two groups do not overlap. When the target process needs to be addressed, a (inter-communicator, rank) pair with the rank relative to the remote group is used.

Because of its unique features, the inter-communication is used in our revised parallelism implementation to couple the fvGCM and mgGCE. The calling tree of the main program, fvGCM.F, contains the following steps to finish a MMF run: (1) a master process calls the mmf_gce2d_task_init() to create two groups of processes and inter-communicators. One group has P fvGCM processes and the other has Q mgGCE processes. Inside this subroutine, *the mgGCE processes start waiting for inputs to perform GCE calculations and interact with the fvGCM processes to receive their inputs in step (5);* all fvGCM processes return back to the main

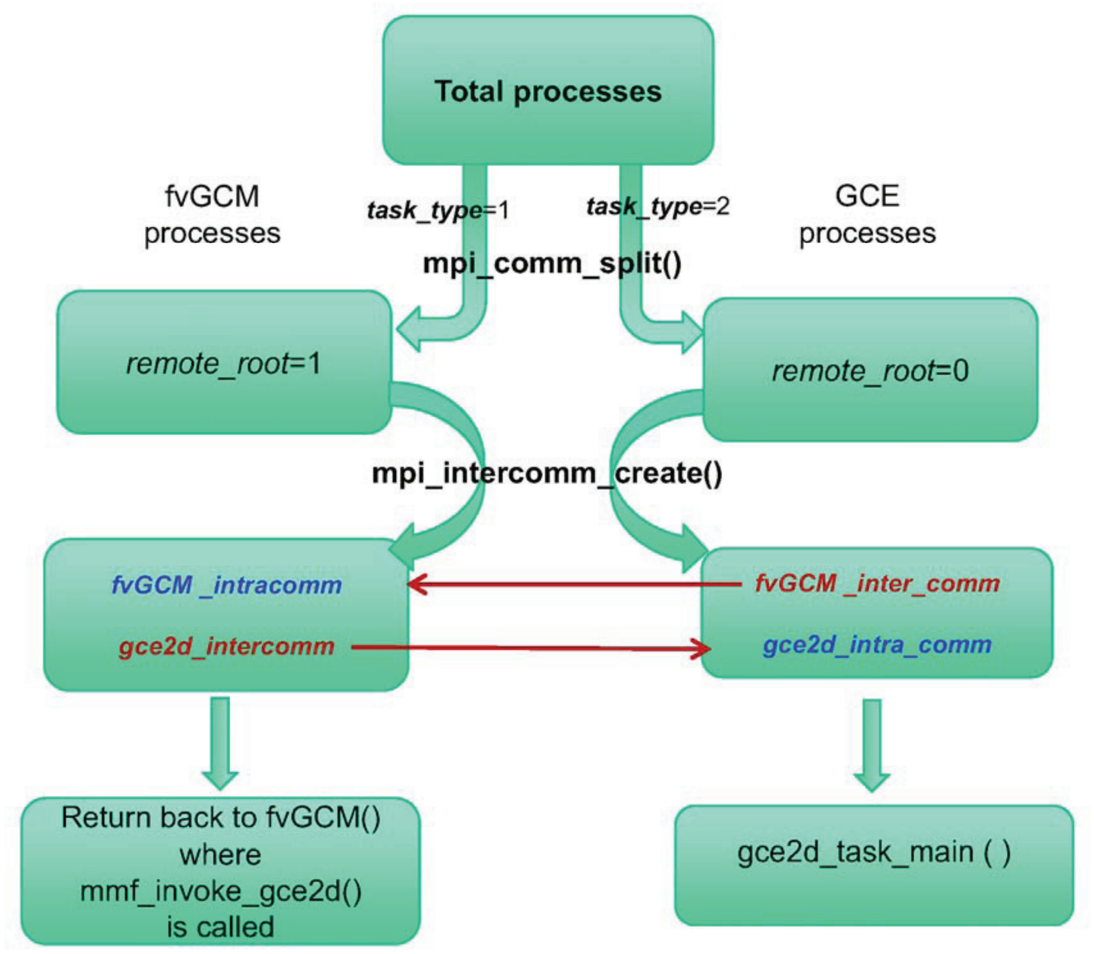
Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

program to continue the execution; (2) mp_init() and y_decomp() are called by all fvGCM processes to perform a 1-D domain decomposition; (3) each of the fvGCM processes calls the mmf_gce2d_task_assignment() to *associate itself with a* subset of ("T(J)") mgGCE processes. Here, "J" is the rank of the fvGCM process and T(J) is proportional to the latitudes assigned to the fvGCM process, aimed at achieving load balancing; (4) mmf_init() is called by all fvGCM processes to initialize the MMF run**;** (5) all P fvGCM processes invoke the mmf_run(), where the mmf_invoke_gce2d() is called, to interact with Q mgGCE processes which are running the gce2d_task_main() to finish 144x91 gec2d runs; (6) mmf_finalize() and mmf_gce2d_task_finalize() are sequentially called by the fvGCM processes to finalize the fvGCM tasks and mgGCE tasks, respectively. Next, we provide more detailed discussions about steps (1), (3) and (5) in sections 3.2.1, 3.2.2, and 3.2.3, respectively.

3.2.1 Creation of Two Process Groups and MPI Inter-Communicators

Figure 4, below, displays major functions in the subroutine mmf_gce2d_task_init() that (i) create two groups of processes by calling MPI subroutine mpi_comm_split()**;** (ii) create inter-communicators among these two groups with the mpi_interomm_creat(), which include *gce2d_intercomm* with the fvGCM processes (the left-hand side of Figure 4) as local processes, and *fvGCM_inter_comm* with the mgGCE processes (the right-hand side of Figure 4) as local processes. While mgGCE processes enter a loop of gce2d_task_main() and start waiting for inputs to perform GCE calculations, all fvGCM processes return back to the main program where mmf_invoke_gce2d() is called to send data to the mgGCE processes. Major functionalities of these two subroutines are discussed using List 1 and List 2, respectively, in section 3.2.3. These mgGCE processes are homogenous and undifferentiated. In fact, they are designed to be "stateless"—they don't need to know anything about the fvGCM side of things. They do not know what timestep it is, who might send them input, or anything about load balancing or task assignments, etc. They simply receive a block of input from fvGCM processes, use that input to call the gce2d() routine, and then return a block of output back to whoever sent them the input. Everything they need to know is either constant (e.g., configuration switches that are read in from the namelist file) or provided within the block of input.

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE



<u>Figure 4:</u> Creation of two groups of processes and their intra-communicators and inter-communicators in the subroutine mmf_gce2d_task_init(). The fvGCM group appears on the left-hand side, while the mgGCE group on the right-hand side. gce2d_intercomm and fvgcm_inter_comm (in red) are inter-communicators, while gce2d_intra_comm and fvGCM_intracomm (in blue) are intra-communicators. A pseudocode of gce2d_task_main() and mmf_invoke_gce2d() can be found in List 1 and List 2, respectively.

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

3.2.2 Process Mapping and Load Balancing

For a given *total_number_of_processes* (P+Q) and *total_number_of_mgGCE_processes (Q)* at runtime, the mapping between P fvGCM and Q mgGCE processes happens in the routine mmf_gce2d_task_assignment() illustrated in Figure 5. The fvGCM processes agree among themselves to carve up the group of mgGCE processes into disjoint subsets, with each fvGCM process getting one of these subsets. Although the assignment is static, it is not necessarily uniform.

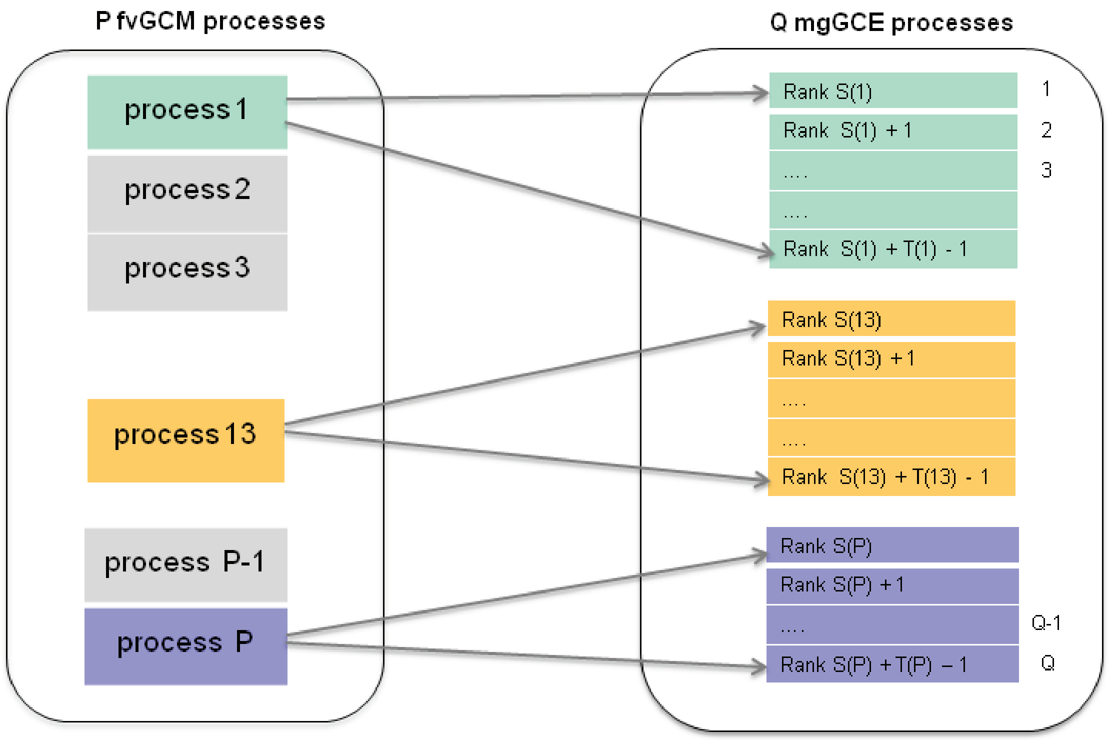


Figure 5: Mapping of between P fvGCM process (left) and Q mgGCE processes (right). The fvGCM process with the rank of "J" receives a subset of Q mgGCE processes, say T(J), depending on the amount of work assigned to the fvGCM task (i.e., the number of latitudes). $\sum_{J=1}^{P} T(J) = Q$. *Let S(J) be the rank of the first mgGCE process (i.e., starting index) in the* J^{th} *subset of mgGCE processes. S(1)=1 and S(J+1) = S(J) + T(J), J=1, 2,… P-1.*

For simplicity, these disjoint subsets are chosen to be a contiguous block of processes, where "contiguous" is relative to their rank numbering within the inter-communicator *(with the mgGCE as a remote group)*. The array *firstExtraGce2dTask*, associated with the fvGCM processes, gives the comm rank of the first mgGCE process of the subset assigned to that fvGCM process, and *numGce2dTasksAssigned* is the number of mgGCE processes assigned to that fvGCM process; *firstExtraGce2dTask and numGce2dTasksAssigned* are referred to as S(J) and T(J), respectively, in Figure 5, here "J" is the rank of the fvGCM process. For example, fvGCM process with the rank of 13 has "*numGce2dTasksAssigned(13)*" or *"T(13)"* processes, starting with intercomm rank "*firstExtraGce2dTask(13)*" or *"S(13)"*, and going up contiguously. *T(J) is roughly equal to (Q/P) but is larger for the fvGCM process that has more latitudes than the other processes. The summation of all T(J) should be equal to Q, namely* $\sum_{J=1}^{P} T(J) = Q$. S(1)=1 and S(J+1) = S(J) + T(J), J=1~P-1. These intercomm task index values are what the fvGCM process uses for the "destination" field of the mpi_send() call, which requires a (inter-communicator, "rank") pair with the "rank" relative to the remote group.

Therefore, the load balancing can be achieved by giving more mgGCE processes (i.e., larger T(J)) to the fvGCM processes which have more latitudes. As shown in the top panel of Figure 3, if one fvGCM process has to do three latitudes, and a different fvGCM process has to do four latitudes, we give the second fvGCM process more mgGCE processes (i.e., larger "*numGce2dTasksAssigned* " or "T") than we give to the first. Thus, the second fvGCM process may be able to do the work within a single latitude in only three-quarters as much time, and so the two fvGCM processes will both finish their work in roughly equal amounts of time. Simply speaking, although the load imbalance was introduced in association with the 1D non-uniform decomposition in the y direction in the original MMF, it can be mitigated or resolved by another 1D non-uniform decomposition in the x direction in the new MMF. Note that each of the fvGCM processes process does not share any of its work with other mgGCE processes outside of its assigned subset.

List 1: A pseudocode of the gce2d_task_main() routine. Each of the mgGCE processes does a simple loop: receive inputValues from any fvGCM process, call gce2d() model, and return outputValues to the same fvGCM process that sent the inputValues.

List1: gce2d_task_main()

```
mainLoop: do

  call mpi_recv(inputValues, , , MPI_ANY_SOURCE, MPI_ANY_TAG,  fvGCM_inter_comm, status,
         ierror)
  tag = status(MPI_TAG)
  source = status(MPI_SOURCE)

  if (tag .gt. 0) then

    call mmf_call_gce2d(inputValues,outputValues)  !! Compute gce2d()

    !! Pass the outputs back to the sender, using the supplied tag
    call mpi_send(outputValues, , MPI_BYTE, source, tag, fvGCM_inter_comm, ierror)
  else if (tag .eq. 0) then
    !! Exit the do-loop and terminate
    exit mainLoop
  else
    !! Unexpected tag value
  end if

enddo mainLoop
```

*List 2: A pseudocode of the mmf_invoke_gce2d() routine, which addresses distribution of global initial values to Q mgGCE processes from P fvGCM processes. The fvGCM processes send inputs to mgGCE processes (which are running in parallel) and then receive outputs from them. The fvGCM processes continue to do so until the GCE calculation over the entire globe (i.e., the calculation of the 13,104 copies of GCE) is completed. Q is equal or larger than the multiple of P, namely Q>=C*P, C is an integer.*

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

List2: mmf_invoke_gce2d ()

```
sendLoop: do d=1, depth
  do i=1, numGce2dTasks
     tag = 2 * nextColumn
     call mpi_isend(inputValues(nextColumn), ..., tag,  gce2d_intercomm, request(nextColumn), ierror)
     nextColumn = nextColumn + 1
  enddo
enddo sendLoop

sendRecvLoop: do while (nextColumn .le. NumLongitudes)

  call mpi_iprobe (MPI_ANY_SOURCE, MPI_ANY_TAG, gcen2d_intercomm, pending, status, ierror)

  If (pending) then

    whichTask = status (MPI_SOURCE)
    whichColumn = status(MPI_TAG) / 2
    call mpi_recv (outputValues(whichColumn),...., whichTask, ..., gce2d_intercomm   ....)

    call mpi_request_free (request(whichColumn), ierror)

    tag = 2 * nextColumn
    call mpi_isend (inputValues(nextColumn), ... tag, gce2d_intercomm, request(nextColumn), ierror)

  else

    call mmf_call_gce2d (tmpInput ......)  ! Running with fvGCM processes

  endif

  numCompleted = numCompleted + 1
  nextColumn = nextColumn + 1

enddo sendRecvLoop

recvLoop: do while (numCompleted .lt. NumLongitudes)

  call mpi_probe (MPI_ANY_SOURCE, MPI_ANY_TAG, gcen2d_intercomm, pending, status, ierror)
  whichColumn = status(MPI_TAG) / 2

  call mpi_recv (outputValues(whichColumn), ... gce2d_intercom ......)
  call mpi_request_free (request(whichColumn), ierror)

  numCompleted = numCompleted + 1

enddo recvLoop
```

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

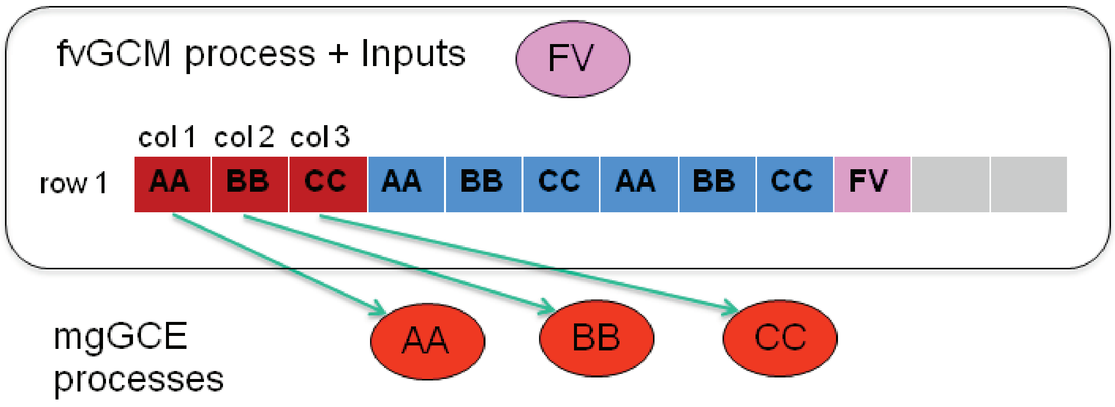
3.2.3 Dynamical Distribution of mgGCE Inputs over mgGCE Processes

Lists 1 and 2 provide pseudo codes to show how mgGCE and fvGCM processes interact to perform the calculations of 13,104 copies of GCE. List 1 displays a main loop in gce2d_task_main() where all mgGCE processes are waiting to receive *inputValues* from any fvGCM process (line 2), call the gce2d() model (line 6), and return *outputValues* (line 8) to the same fvGCM process that sent the *inputValues*. In short, mgGCE process does a simple loop: "recv input - compute gce - send output." As mentioned, these mgGCE processess are essentially stateless (e.g., in section 3.2.1). However, in the current implementation, each mgGCE process is bound (assigned statically) to a single, particular fvGCM process (e.g., in section 3.2.2), but that is just a convenience for the fvGCM side.

All the controlling business happens on the fvGCM side, which is shown in List 2. Each fvGCM process uses the fixed and static process mapping to interact with the corresponding subsets of mgGCE processes. In addition to the controlling business, the fvGCM processes can also help perform GCE calculations to take advantage of their computational potential. Therefore, a dynamic distribution of inputs and computations for the entire globe (i.e., 13,104 copies of gce2d()) to the mgGCE processesis implemented by the following-three phases, which correspond to the three loops in the routine in List 2:

i. in the "send" loop (lines 1-7), the fvGCM process just sends out a lot of inputs to all of its mgGCE processes, queuing up *"depth"* inputs to each mgGCE process. Here the variable "*depth*" indicates the number of inputs which are already sent asynchronously to the input queue of each mgGCE process;
ii. in the "sendRecv" loop (lines 8-22), the fvGCM process checks if any of its mgGCE processes has produced an output, and if so, sends another input piece of work to the mgGCE's queue. However, if none of the mgGCE processes has produced an output, rather than waiting for the outputs from the mgGCE process(es), one of the fvGCM process pitches in and does the next piece of work (i.e., running a copy of gce2d()). Then it goes back to checking for output. The fvGCM processes repeat doing the above until all the input has been sent (e.g., using *nextColumn* to keep track);
iii. in the "recvLoop" (lines 23-29), the fvGCM processes keep receiving outputs until all the output has been received (using *numCompleted* to keep track).

During the phases (i)-(iii), the matchup between a particular mpi_[i]send() and a particular mpi_[i]recv() among fvGCM and mgGCE processes happens dynamically and asynchronously.



<u>Figure 6:</u> Dynamic distribution of mgGCE input values. Here, we assume that one specific fvGCM process (at top, in pink) has 3 mgGCE processes (at bottom, in red). The fvGCM process sends inputs to each of the 3 mgGCM processes, which begin processing them (red). The fvGCM process also queues additional inputs (blue) for the mgGCE processes. While waiting for results from mgGCE process, the fvGCM process can also help perform a GCE calculation (middle, in pink). When computations with inputs in red are done, the mgGCE process returns results and begins processing the next input in its queue. On receiving a result, the fvGCM process queues another input (grey) to whichever mgGCE process returned the result.

A simple illustration of calling mmf_invoke_gce2d() by an fvGCM process in List 2 is now discussed with Figure 6. Assume that this fvGCM process has (*numGce2dTasks, NumLongitudes*) to be (3, 144) in List 2. Thus, it interacts with three mgGCE processes, call them AA, BB, and CC in Figure 6, to dynamically distribute 144 inputs and finish 144 copies of GCE runs. With the *depth=3* in the sendLoop (line 1 in List 2), the fvGCM process initiates 9 *(depth*numGce2dTasks)* mpi_isend() calls (line 4) to send *inputValue* records as follows: the 1st to AA, the 2nd to BB, the 3rd to CC, the 4th to AA, 5th to BB, 6th to CC, 7th to AA, 8th to BB, and the 9th to CC. This ends the sendLoop. Each of the three mgGCE processess now has 3 (*depth*=3) input records queued up. Each of these mgGCE processes starts running the

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

subroutine gce2d() as soon as they received some input. The fvGCM process now enters the sendRecv loop (between lines 8-22) and calls mpi_iprobe() (line 9) to check for outputs sent from any of the mgGCE processes. Let us suppose that none has done so. Rather than wait, fvGCM process takes the next piece of input (the 10th) and calls gce2d() (line 18). When the fvGCM process finishes and returns, it again checks for outputs from any mgGCE processes (line 9). Suppose BB has finished one gce2d() calculation, and so BB has returned the output for the 2nd piece. BB then immediately begins work on its next piece, namely the 5th piece. As soon as the fvGCM process sees the existence of a pending message (i.e., the outputs from the mgGCE), it calls mpi_recv() (line 13) to receive the outputs. The fvGCM process can tell it is the 2nd piece by looking at the tag value, and can tell that it came from process BB. So the fvGCM process sends the next piece of input (the 11th) to BB (line 16) and then returns back to the beginning the loop in line 9 to check for outputs from any mgGCE processes. This loop continues until all 144 pieces of input have been handed out. The distribution of inputs and gce runs is handed out dynamically as each previous piece of work is finished. The fvGCM process now enters the recvLoop (line 23). There is no more input to be handed out, so the fvGCM process waits until all the remaining pieces of work have been finished and returned.

3.3 Computational Results

In this section, we present computational results with the improved MMF on the NASA Pleiades supercomputer, which is currently one of the most powerful general-purpose supercomputers in the world.

3.3.1 NASA Supercomputer

The Pleiades supercomputer at NASA Ames Research Center, Moffett Field, CA is an SGI Altix ICE system with a peak performance 1.34 Pflop/s. With 191TB of total memory, and 112,896 cores (plus 64 GPU nodes, each with 512 CUDA cores.), it achieves LINPACK performance of 1.09 Pflop/s (June 2011, using 11,648 nodes). The system contains three different types of Intel® Xeon® processors: X5670 (Westmere), X5570 (Nehalem), and E5472

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

(Harpertown) to reach different needs and capacities from different NASA projects. Hehalem processors were used in our benchmark.

3.3.2 A Performance Benchmark

Figure 7 shows a benchmark with encouraging parallel scalability up to 3,335 CPUs on Pleiades. Here, the speedup is determined by T_{30}/T, where T is the wall-time to perform a five-day forecast with the MMF and T_{30} is the time spent using 30 CPUs. The run with 30 CPUs was chosen as a baseline because this configuration was previously used for production runs (Tao et al., 2008, 2009). A speedup of 3.42, 10.81, 19.46, 38.46, 50.16, and 79.77 is obtained by increasing the number of CPUs from 30 to 91, 273, 546, 1115, 1680, and 3335 cores, respectively. As the baseline has load imbalances and excessive memory usage in the master process, it is not surprising to obtain a super-linear speedup with lower CPU counts up to 1115. Further analysis of the MMF's throughput indicates that it takes about 41 minutes to finish a five-day forecast using 3335 cores, which meets the requirement for performing real-time numerical weather prediction—that is, completion of a run within one hour. A three-year simulation would only take one day to run with 3335 cores, as opposed to about 80 days with 30 cores. This speedup in wall-time makes it far more feasible for increasing the resolution in the fvGCM and studying TC climate. The enhanced coupled model is being used to perform multi-year runs for studying large-scale tropical weather systems (e.g., MJOs).

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

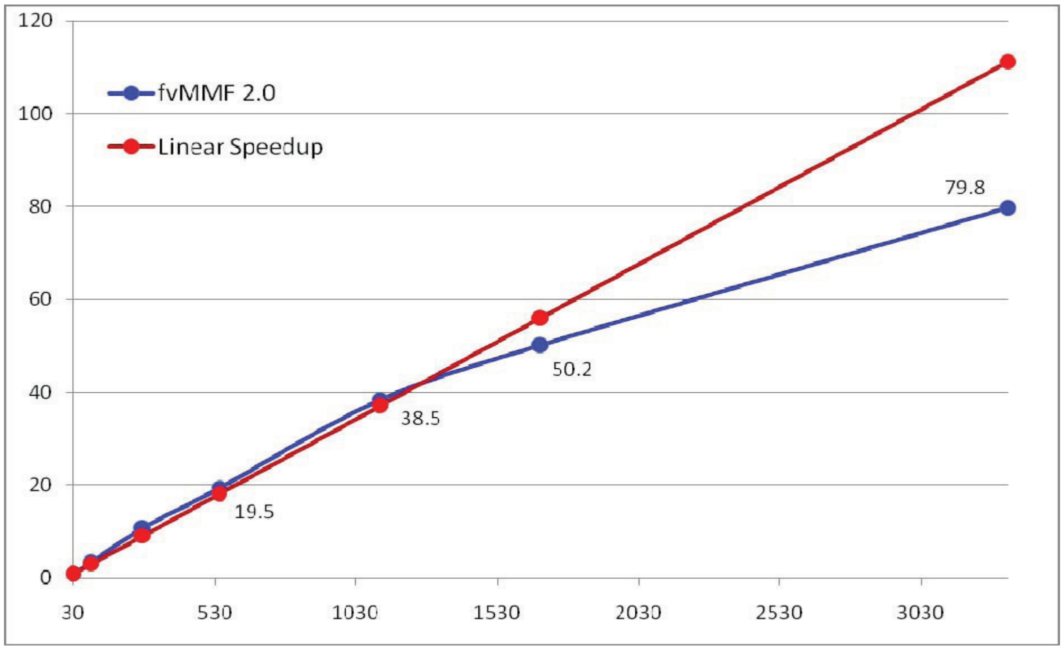


Figure 7: *Parallel scalability of the MMF v2.0 with a revised parallel implementation on the NASA Pleiades supercomputer. This figure shows that a speedup of nearly 80x is obtained as the number of cores increases from 30 to 3,335. Note that the original MMF could use only 30 cores.*

4. Concluding Remarks

Improving our understanding of TC inter-annual variability and the impact of climate change (e.g., doubling CO2 and/or global warming) on TC activities brings both scientific and computational challenges to researchers. As TC dynamics involves multiscale interactions among large-scale (synoptic-scale) flows, mesoscale vortices, and small-scale cloud motions, an ideal numerical model suitable for TC studies should demonstrate its capabilities in simulating these interactions. Thanks to recent advancements in global numerical models and supercomputer technology, these topics can be addressed more accurately and quickly than ever before. This article focuses on the recent improvements to the MMF, which consists of a coarse (2^ox2.5^o) resolution fvGCM and more than 10,000 copies of fine-resolution GCE (cloud model).

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

The major task was to distribute massive copies of GCE over a large number of CPUs. To facilitate discussion, these GCE processes, which cover the entire globe, are referred to as a super-component meta-global GCE (mgGCE). In the revised parallelism, MPI intercommunication is used to provide data exchanges between two groups of processes, one with P fvGCM processes and the other with Q mgGCE processes. The implementation of a sophisticated algorithm for P-to-Q mapping allows the mgGCE to be running with an effective 2D domain decomposition, while the fvGCM still remains to be running with its original 1D domain decomposition. In addition, load balancing is being taken cares by dynamical allocating available processes to perform calculation. As 99% of the computing time for the MMF is spent on the mgGCE, the revised parallelism leads to a substantial performance. For example, a speedup of nearly 80 is obtained by increasing the number of CPUs from 30 to 3,335 where 30 is the upper limit of CPU counts in the first version of MMF. A three-year run with 3,335 cores needs just 24 hours of wall-time. This improved scalability makes it more feasible to perform long-term climate simulations.

Ideally, the parallelism that leads to an effective 2D domain decomposition in the mgGCE can be applied to other types of column-based physics parameterizations (e.g., calling a radiation routine instead of the gce2d routine). This approach also lays the groundwork for a more sophisticated modeling approach to solve extraordinarily complex problems with advanced computing power. *For example, the improved scalability makes it possible to deploy a more advanced MMF with the fvGCM at a higher resolution, say $0.25^o x 0.36^o$, which has much larger grid points (721x1000) and thus requires much more copies of GCEs.* In addition, by taking load balance into consideration, the current implementation makes it feasible to choose a variety of GCEs (e.g., 3D GCE or more advanced microphysical processes) in the mgGCE. Further improvements include the implementation of an efficient I/O module and/or a parallel I/O module with a CPU layout that can be different from that in the either fvGCM or the mgGCE.

The current parallel implementation in the MMF is a coarse-grained parallelism as compared to the parallelism inside a GCE. Since an individual GCE was previously implemented with its native 2D domain decomposition, another level of parallelism (fine-grain parallelism) inside each copy of GCE can greatly expand the number of CPUs for MMF runs. Potentially, the

Digital Object Indentifier 10.1109/MCSE.2012.90 1070-9924/$26.00 2012 IEEE

coupled MMF, along with the mgGCE, could be scaled at a multiple of 13,104 CPUs. This is subject to a future study.

Acknowledgements:

We are grateful to the following organizations for their support: the NASA Earth Science Technology Office, the Advanced Information Systems Technology Program, and the NASA Modeling, Analysis Prediction Program. We thank the NASA High-End Computing Program and the NASA Advanced Supercomputing facility at ARC for the support of computing resources and services. We thank Ms. Jill Dunbar of NASA ARC for proofreading this manuscript.

References

Atlas, R. et. al., 2005: Hurricane forecasting with the high-resolution NASA finite volume general circulation model. *Geophys. Res. Lett*., **32**, L03801, doi:10.1029/2004GL021513.

Bengtsson, L., K., I. Hodges, and M. Esch, 2007: Tropical cyclones in a T159 resolution global climate model: comparison with observations and re-analyses. Tellus A 59 (4), 396.416 doi:10.1111/j.1600-0870.2007.00236.x

Frank, W. M. and P. E. Roundy, 2006: The Role of Tropical Waves in Tropical Cyclogenesis. *Mon. Wea. Rev.* **134**, 2397-2417.

Juang, J.-M. et al., 2007: Parallelization of NASA Goddard Cloud Ensemble Model for Massively Parallel Computing. *Terrestrial, Atmospheric and Oceanic Sciences.*

Lin, S.-J., B.-W. Shen, W. P. Putman, 2003: Application of the high-resolution finite-volume NASA/NCAR Climate Model for Medium-Range Weather Prediction Experiments. EGS - AGU - EUG Joint Assembly, Nice, France, 6 - 11 April 2003.

Putman, W., S.-J. Lin, and B.-W. Shen, 2005: Cross-Platform Performance of a Portable Communication Module and the NASA Finite Volume General Circulation Model. *International Journal of High Performance Computing Applications*. **19**: 213-223.

Randall, D. et. al., 2003b: Breaking the Cloud Parameterization Deadlock. *Bull. Amer. Meteor. Soc.*, 1547-1564.

Shen, B.-W. et. al., 2006: Hurricane Forecasts with a Global Mesoscale-Resolving Model: Preliminary Results with Hurricane Katrina (2005). *Geophys. Res. Lett.,* **33**, L13813, doi:10.1029/2006GL026143.

Shen, B.-W. et al., 2010: Predicting Tropical Cyclogenesis with a Global Mesoscale Model: Hierarchical Multiscale Interactions During the Formation of Tropical Cyclone Nargis (2008). J. Geophys. Res.,115, D14102, doi:10.1029/2009JD013140.

Shen, B.-W., W.-K. Tao, and B. Green, 2011: Coupling Advanced Modeling and Visualization to Improve High-Impact Tropical Weather Prediction (CAMVis), IEEE Computing in Science and Engineering (CiSE), vol. 13, no. 5, pp. 56-67, Sep./Oct. 2011, doi:10.1109/MCSE.2010.141

Shen, B.-W. et al., 2012a: Genesis of twin tropical cyclones as revealed by a global mesoscale model: The role of mixed rossby gravity waves, *J. Geophys. Res.*, doi:10.1029/2012JD017450, in press.

Shen, B.-W. et al., 2012b: Advanced Visualizations of Scale Interactions of Tropical Cyclone Formation and Tropical Waves. Computing in Science & Engineering, 10 May 2012. IEEE computer Society Digital Library. (in press)

Tao, W.-K. and J. Simpson, 1993: The Goddard Cumulus Ensemble Model. Part I: Model description. *Terrestrial, Atmospheric and Oceanic Sciences*, **4**, 19-54.

Tao, W.-K. et. al., 2003: Convective Systems over South China Sea: Cloud-Resolving Model Simulations. *J. Atmos. Sci.*, **60**, 2929-2956.

Tao, W.-K. et. al., 2008: A Goddard Multi-Scale Modeling System with Unified Physics. WCRP/GEWEX Newsletter, Vol 18, No 1, 6-8.

Tao, W.-K., et al. 2009: Multi-scale modeling system: Development, applications and critical issues, *Bull. Amer. Meteor. Soc.* **90**, 515–534.