# CONeP: A cost-effective online nesting procedure for regional atmospheric models

Yoshida, Ryuji ; Nishizawa, Seiya ; Yashiro, Hisashi ; Adachi, Sachiho A. ; Sato, Yousuke ; Tomita, Hirofumi

(Citation)
Parallel Computing,65:21-31

(Issue Date)
2017-07

(Resource Type)
journal article

(Version)
Version of Record



(URL)
https://hdl.handle.net/20.500.14094/90004354

Parallel Computing 65 (2017) 21–31

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

# CONeP: A cost-effective online nesting procedure for regional atmospheric models

Ryuji Yoshida[a,b,*], Seiya Nishizawa[a], Hisashi Yashiro[a], Sachiho A. Adachi[a], Yousuke Sato[a], Tsuyoshi Yamaura[a], Hirofumi Tomita[a]

[a] RIKEN Advanced Institute for Computational Science, Kobe, Japan
[b] Research Center for Urban Safety and Security, Kobe University, Kobe, Japan

ARTICLE INFO

*Article history:*
Received 5 September 2016
Revised 1 February 2017
Accepted 7 April 2017
Available online 11 April 2017

*Keywords:*
Regional atmospheric model
Down scaling
Domain-nesting method
Parallelization technique
Performance
High-performance computing

ABSTRACT

We propose a cost-effective online nesting procedure (CONeP) for regional atmospheric models to improve computational efficiency. The conventional procedure of online nesting is ineffective because computations are executed sequentially for each domain, and it does not enable users freely to determine the number of computational nodes. However, CONeP can completely avoid this limitation through three actions: 1) splitting the processes into multiple subgroups; 2) making each subgroup manage just one domain; and 3) executing the computations for each domain simultaneously. Since users can assign an optimal number of nodes to each domain, the model with CONeP is computationally efficient. We demonstrate the computational advantage of CONeP over the conventional procedure, comparing the elapsed times with both procedures on a supercomputer. The elapsed time with CONeP is markedly shorter than that observed with the conventional procedure using the same number of computational nodes. This advantage becomes more significant as the number of nesting domains increases.



## 1. Introduction

Regional atmospheric models are widely used for numerical weather prediction and regional climate projection. The lateral boundary and initial conditions in such simulations are given by external data that represent large-scale phenomena, e.g., the output of an atmospheric general circulation model or reanalysis data. For example, the fine structures of squall lines accompanied by local heavy rain within the large-scale environment are often investigated using the regional model. To express both the large-scale environment and fine structures simultaneously, it is necessary to configure a wide area for the former, on the order of thousands of kilometers, and a high-resolution area for the latter, on the order of a few kilometers. This type of experiment requires considerable computational resources within a single simulation with a homogeneous spatial resolution.

One possible solution to reduce the computational resources required is the domain nesting method, shown in Fig. 1. This method is adopted in many regional atmospheric models, such as the Advanced Regional Prediction System (ARPS) model

* Corresponding author at: RIKEN Advanced Institute for Computational Science, 7-1-26 Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan.
*E-mail address:* ryoshida@riken.jp (R. Yoshida).

http://dx.doi.org/10.1016/j.parco.2017.04.004
0167-8191/

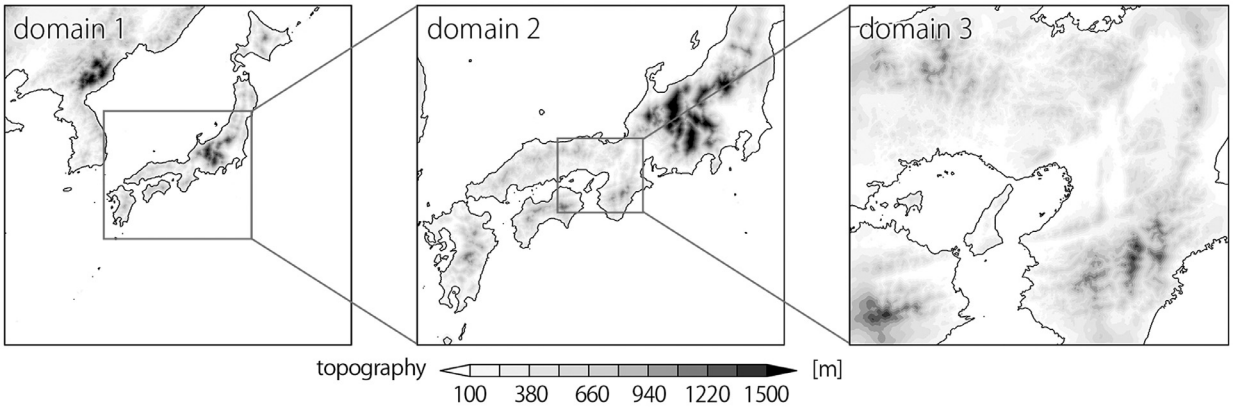


**Fig. 1.** Domain-nesting experiment. The computational domains are triply nested. Gray rectangles indicate the nesting location of the child domain. Shading indicates the topography of the domain.

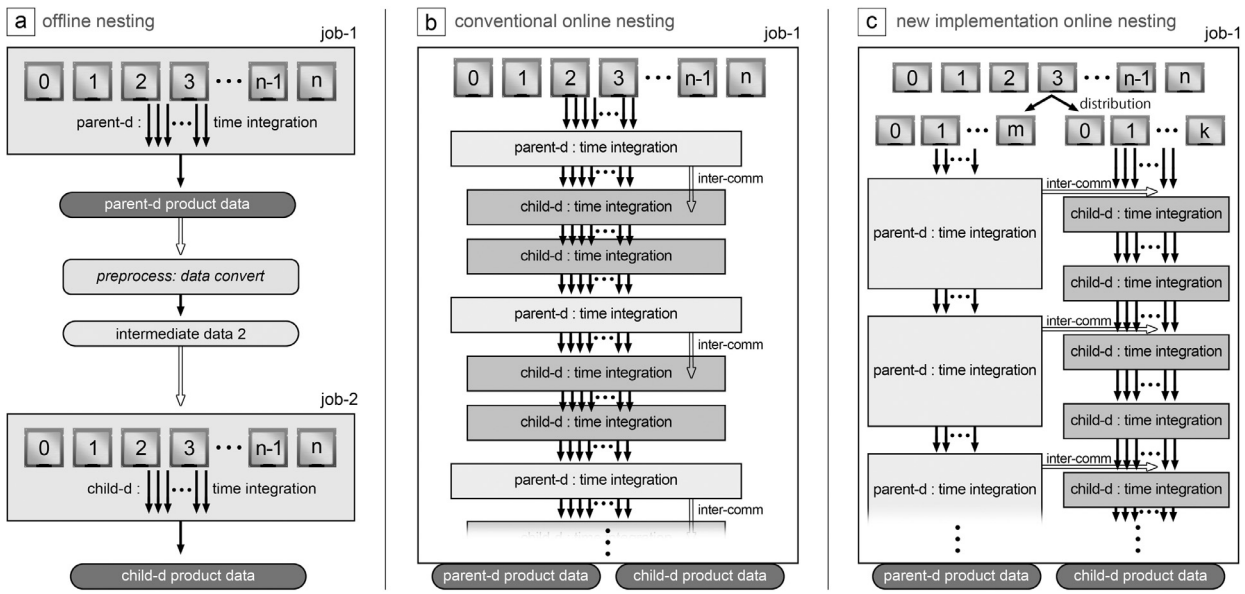


**Fig. 2.** Execution procedures for offline nesting, the conventional procedure of online nesting, and the new implementation of online nesting. "parent-d" and "child-d" denote the parent domain and child domain, respectively. Jobs for each domain are executed separately in offline nesting. A single job is executed for both domains in online nesting.

[1–3], the Regional Atmospheric Modeling System (RAMS) model [4,5], the fifth-generation Pennsylvania State University-National Center for Atmospheric Research Mesoscale Model (MM5: [6,7]), the Weather Research and Forecasting Model (WRF)–Advanced Research WRF (WRF-ARW) model [8], the Japan Meteorological Agency non-hydrostatic mesoscale model [9], and SCALE-RM [10,11]. In this method, the large-scale environment is simulated in an outer domain with coarser resolution, whereas smaller-scale phenomena are simulated in an inner domain with finer resolution. As explained below, a domain-nesting method can be categorized as either offline nesting or online nesting.

Although the procedure for offline nesting is simpler and easier to implement than that for online nesting, intermediate files are necessary for data transfer between the two domains. Fig. 2(a) shows the procedure of offline nesting for a doubly-nested domain. Since the simulations for the outer and inner domains are conducted separately, the history data of the outer domain simulation should be saved once in a storage location. Generally speaking, in order to suppress temporal errors due to the lateral boundary condition from the outer-domain simulation, it is desirable to update its interval in the inner-domain simulation as frequently as possible. The sufficient update interval of the lateral boundary condition has been investigated [12]. Denis [13] concluded that a 6 h interval is sufficient for a 45 km horizontal resolution, but a much shorter interval is required for higher resolution. Mass [14] suggested that a 6 h interval is insufficient for regional numerical weather prediction. Recent high-resolution simulations tend to demand a high updating frequency. As a result, tremendous data storage is required to satisfy the sufficient update interval of the input data to the inner simulation. It is often unrealistic. Practically, the output interval is compromised by the file size of the history data.

Online nesting resolves this issue. It is free from the file-size problem because calculations for the outer and inner domains are conducted simultaneously. The history data of the outer domain is transferred by means of intercommunications instead of files. The boundary conditions for the inner domain can be updated with ideal frequency, e.g., for every time step of the outer domain. Online nesting is available in the current regional atmospheric models. Michalakes [15] implemented online nesting by means of a straightforward extension from offline nesting. Both domains are handled by the same processes in Michalakes's implementation, as shown in Fig. 2(b). This conventional procedure of online nesting (CNV) is commonly used in regional atmospheric models.

Considering modern architectures and future trends in high-performance computing (HPC), the CNV is not suitable for obtaining efficient computational performance because of the difficulty of strong scaling. In modern HPC architectures, the numbers of processors are markedly increasing, outpacing improvements in intercommunication speed. Assuming that the outer domain has fewer grid points and a coarser resolution than the inner domain, the problem size per processor in the outer domain is smaller than that in the inner domain. It is because the same number of computational nodes is used for all domains in the CNV. Thus, the computational efficiency for computing the outer domain is considerably degraded due to amount of data that must be communicated between neighboring nodes.

Does a more effective procedure exist for online nesting with respect to computational performance? This is the motivation of our study. Recently, a new task-assignment scheme has been implemented for parallel computation of multiple domains in the National Centers for Environmental Prediction (NCEP) Nonhydrostatic Multi-Scale Model on the B grid (NMMB) [16,17]. However, the advantage of this procedure over the CNV has not yet been verified formally on any massively parallel system.

In this study, we propose a procedure based on the aforementioned parallelization concept for online nesting that adds several specialized functions and demonstrate its usefulness. Our framework of nesting is called cost-effective online nesting procedure (CONeP). This procedure makes it possible to execute nested domain calculations simultaneously, as in the NMMB. We also add additional convenient functions and flexible settings for vertical grid arrangements and map-projection types, on a domain-by-domain basis.

The other aspect of domain nesting concerns inter-domain communication using either the one-way (constraining) or two-way (interactive) method. We focus on the one-way method, and discuss the merits of one-way method against two-way method in the next section. In addition, we examine the case of a stationary domain and do not address the moved nesting case.

We describe in detail the aspects of CONeP in Section 2, specifically the design concept, management of processes, details of resolving the domain relationship, and computational procedures for domains. We demonstrate the usefulness of CONeP through experiments in Section 3. Finally, we provide the study's conclusion in Section 4.

## 2. Detail framework for CONeP

We assume that processes and inter-process communications are managed by the message passing interface (MPI) [18] and a single process is assigned to a single computational node. Henceforth, process refers to a single specific node.

### *2.1. Design concept*

CONeP is designed so that multiple domain computations can be executed in parallel by splitting computational processes. Fig. 2(c) shows the schematic diagram, in which the calculations of the outer and inner domains overlap; while the inner-domain calculation is performed until the next boundary-update time, the outer-domain calculation is performed in parallel for the next time step. Since the latter result is used immediately for the inner-domain calculation as the boundary condition, the inner-domain calculation can be continued. By assigning all the required computational processes properly into the inner- and outer-domain simulations, the outer-domain calculation can be hidden behind the inner-domain calculation. At this time, the number of processes for each domain can be tuned to achieve optimal computational efficiency. Thus, CONeP is expected to be superior to the CNV in terms of computational efficiency. So far, we have considered only the two-domain case for simplicity. However, the actual implementation of CONeP enables its application to any multi-domain case.

Flexibility in configuring an experiment domain by domain is another issue. In the nesting framework, each domain requires an experimental configuration that is free from restrictions enforced by the settings of the other domains, such as limits on the sizes of the regional and horizontal/vertical resolutions. Although offline nesting easily satisfies such demands, most existing online implementations of models seriously suffer from such restrictions. For example, users of most existing online systems cannot determine the vertical grid arrangements of the inner and outer models individually. The selection of map projections of horizontal grid systems for inner models has a similar problem.

In online nesting, two types of data communication method are available: the one-way and the two-way methods. Originally, the two-way method was proposed to remove numerical noise near the interface between the coarse-grid and fine-grid domains [19–21]. Here, the results in the fine-grid domain are fed back to the coarse-grid domain. On the other hand, a sponge-layer method is then useful for the one-way method to reduce the numerical noise [22]. Based on idealized numerical experiments, Harris [23] concluded that the two-way and sponge-layer methods are effective in reducing numerical noise. Soriano [24] conducted numerical experiments for real atmospheric phenomena and revealed that the capabilities of

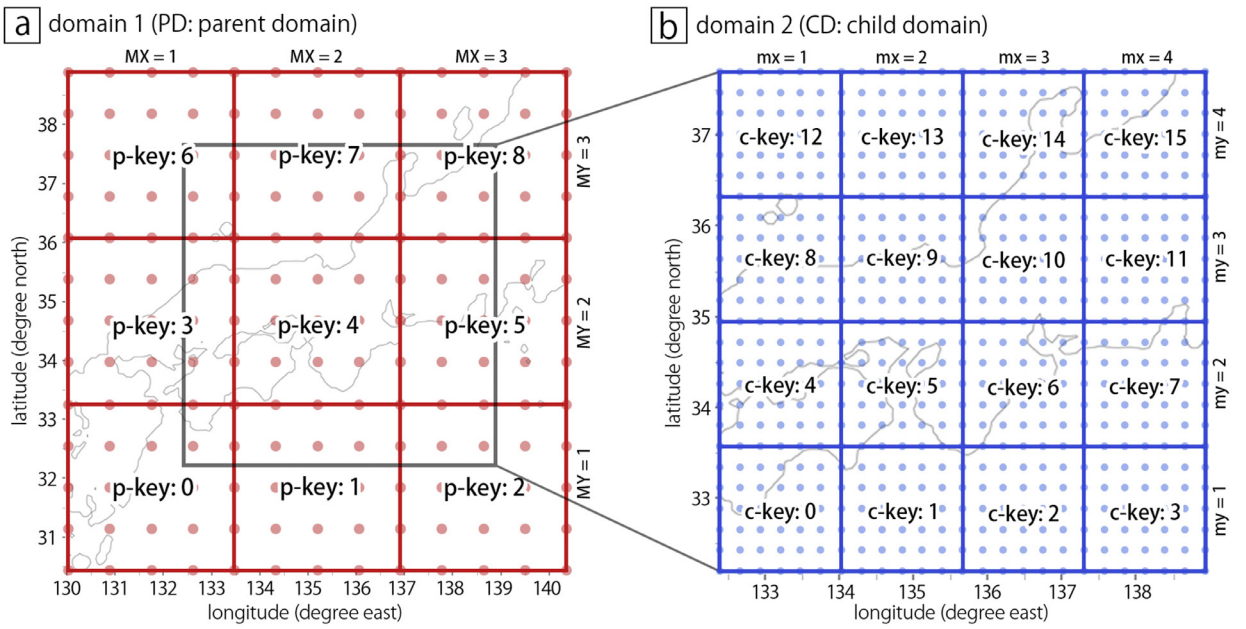


**Fig. 3.** Division maps of computational domains: (a) refers to Domain 1, and (b) refers to Domain 2. Numbers show the corresponding key numbers in each domain cell. In (a), the rectangle shows the location of Domain 2. MX, MY, mx, and my are the node numbers for the x- and y-dimensions for PD and CD, respectively. Dots illustrate grid points in each domain, assuming 13 × 13 grid points in PD and 25 × 25 grid points in CD.

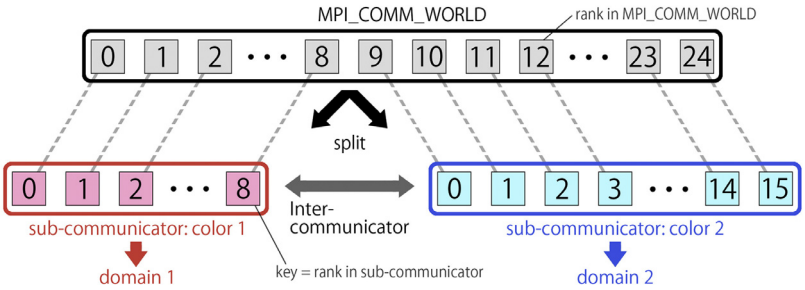


**Fig. 4.** Communicator split. Numbers in each rectangle show rank or key numbers.

the one- and two-way methods vary with different phenomena. In reality, both the one- and two-way methods are used in applications and operational forecasts [23]. Furthermore, a virtue of the one-way method is that we can place an intentional limitation on simulated small-scale phenomena under a certain large-scale state, especially in ideal test cases.

In terms of computational performance, the one-way method has the great merit of performing parallel calculations for nested domains. This is because the time integration of the coarse-grid domain is not dependent on that of the fine-grid domain. The possibility of parallel computation represents a major advantage in reducing the computational time. Therefore, we focus on the one-way method in this study.

The necessary components of CONeP to satisfy the aforementioned requirements are as follows: 1) splitting processes and a communication interface; 2) resolution of domain relationships; and 3) the procedures of computation in each domain. These are explained in the following sections.

### *2.2. Splitting processes and establishing a communication interface*

For simplicity, we assume that 25 processes are used for an experiment with doubly-nested domains, as shown in Fig. 3. The inner domain is located completely within the outer domain. The outer and inner domains are referred to as the parent (PD) and child (CD) domains, respectively. In the MPI, a communicator defines a group of processes, which can communicate with others in the same group. A general communicator, i.e., MPI_COMM_WORLD, is created automatically, including all processes, when the program is launched. To realize parallel computation for multiple domains, MPI_COMM_WORLD is split into several sub-communicators, which are assigned to each domain (Fig. 4). The 25 processes are split into two groups, with 9 and 16 processes for the PD and CD, respectively. To split MPI_COMM_WORLD, an MPI function called “MPI_COMM_SPLIT” is used for ease of implementation.

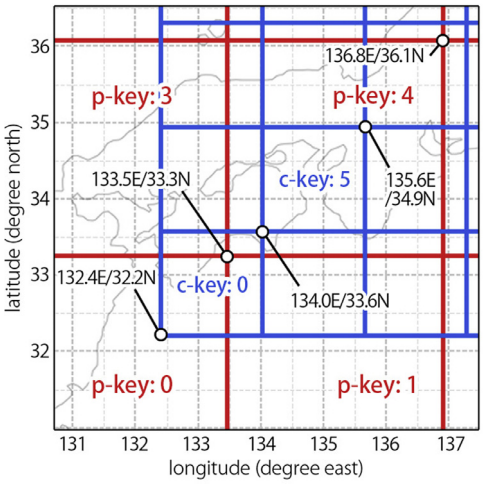


**Fig. 5.** Domain relationship following Fig. 3: zoomed-in picture of Keys 0 and 5 of the child domain over the parent domain.

Any specific process can be identified by its color and its key. Each sub-communicator has its own "color", and each process within any sub-communicator has its own unique "key"; color is an id number of the sub-communicator and key is a rank number or task-id within the sub-communicator. A communication interface is needed to transfer data between different colors. The "inter-communicator" is the interface that connects Color 1 with Color 2 in Fig. 4. Note that even when more than two domains exist, a unique inter-communicator is defined for any two domains that are communicating with one another.

A process assigned to a domain cell is denoted in the manner of "p-key:0" or "c-key:0" in Fig. 3; the first characters "p" and "c" refer to parent and child, respectively, and the number is the key number. As shown in Fig. 3(a), the PD is divided into 9 domain cells with three-by-three mapping, and the 9 processes are assigned to compute over the domain cells. In the same manner, the CD is divided into 16 domain cells with four-by-four mapping, and the 16 processes are assigned to the 16 domain cells, as shown in Fig. 3(b). This is the two-dimensional domain-decomposition scheme of the domain cells, similar to those in Sathye [1] and Michalakes [7]. Intra-domain communication is necessary for halo data communication in stencil calculations. Since processes for each domain belong to one common sub-communicator, halo data communication is executed within this sub-communicator in the same manner as in a single-domain experiment.

When time integration is executed, data are transferred from the PD to the CD using the inter-communicator. Before starting the time integration, we must specify the method in which a key in the sub-communicator of Color 2 is related to the keys in that of Color 1. The detailed method for this specification is explained in the following section.

### *2.3. Resolving domain relationships*

To achieve flexible settings for the domain location, a relationship between the PD and the CD should be resolved within the program automatically. Two tasks must be completed to resolve a domain relationship. The first is to determine the corresponding PD keys for any CD key. The second is to interpolate data from the PD grid points to the CD grid points.

#### *2.3.1. Determining inter-domain key relationships*

To determine communication destinations, we must find the set of keys of the PD domain cells that spatially overlap with a certain CD domain cell. We call the set of keys the corresponding PD keys. For each CD key, we determine the corresponding PD key based on the key distribution over the domain cells and the latitude/longitude information. The identified PD keys are saved in memory as a destination list in the CD keys, and a destination list in the PD keys is generated as a reverse resolution of the list in the CD keys. Thus, two lists for each domain communication consistently form the pairs. Using the list, a PD key specifies all destinations of the corresponding CD keys in the inter-domain communication. When more than two domains exist (i.e., multiple nesting), lists are generated for all parent-child relationships.

Fig. 5 provides an example of the manner in which corresponding PD keys are determined for the same configuration as in Fig. 3. We can usually specify PD keys by the coordinates of the southwestern (SW) and northeastern (NE) corners. For example, consider c-key:5. In this domain cell, the SW corner is located at 134.0°E/33.6°N and the NE corner is located at 135.6°E/34.9°N. In the domain cell of p-key:4, the SW corner is located at 133.5°E/33.3°N and the NE corner is located at 136.8°E/36.1°N. Since both the SW and NE corners of c-key:5 are within the internal region of p-key:4, the corresponding PD key for c-key:5 is determined to be the p-key:4 only. The corresponding PD key for c-key:0 is likewise determined with respect to the NE and SW corners. Thus, the four corresponding PD keys are given as the four keys of 0, 1, 3, and 4. After the PD keys for each CD key are determined, destination lists are generated to describe the relationship, as shown in Fig. 6.

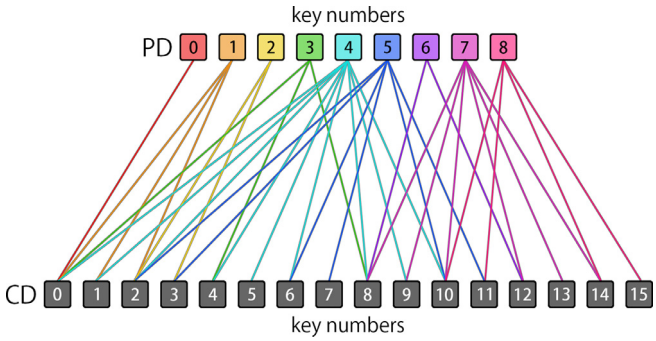


**Fig. 6.** Communication destinations following Fig. 5. The upper numbers are keys for the parent domain (PD) and the bottom numbers are keys for the child domain (CD).

*2.3.2. Data interpolation on the CD grid points*

Horizontal interpolation is conducted based on the distance from the CD grid point to the PD grid point, which is calculated using latitude/longitude data. The linear interpolation is defined as

$$C_{(x,y)} = \sum_{i=1}^{n} F_{x,y,i} P_{(X(i),Y(i))}, \tag{1}$$

$$F_{x,y,i} = \frac{\left(1/Wt_{x,y,i}\right)}{\sum_{i=1}^{n}\left(1/Wt_{x,y,i}\right)}, \tag{2}$$

where $C$ is the interpolated data at the CD grid point, and $P$ represents the original data at the PD grid point. $x$ and $y$ are grid-point indices in the CD, and $X$ and $Y$ are grid-point indices in the PD having the array size of $n$. $F$ is an interpolation factor and $i$ is the index of the source grid points ($i = 1, 2, \ldots n$), where $n$ is the total number of source grid points; in our implementation, the default setting is $n = 4$ for horizontal interpolation. The PD grid points ($P_{(X(i),\, Y(i))}$) are determined as the nearest points to the CD grid point ($C_{(x,\, y)}$). When $n = 1$, the interpolation corresponds to the nearest-neighbor method. $Wt$ is a weighting factor, namely, the value of the distance between the CD and PD grid points. The latitude/longitude data of the PD were previously transferred to the CD by means of inter-domain communication to calculate the weighting factors.

Source grid points are searched and interpolation factors are calculated only for the necessary grid points, i.e., halo regions of stencil calculations and nudging regions at the boundary. Since the domain relationship does not change during time integration, these procedures can be performed before the time integration process starts.

*2.4. Sequence of the computations in each domain*

The sequence of the computations in each domain is chosen to allow for the execution of the time integration in parallel in each domain. The inter-domain communication is implemented with asynchronous communication, using the MPI function MPI_ISEND/MPI_IRECV.

Fig. 7 shows the sequence of a series of inter-domain communications. The program is initialized for the double-nesting case. Here, the time interval of the integration in the CD is assumed to be half that in the PD. In Fig. 7, $t_1$ and $t_2$ ($= t_1 \times 2$) indicate the times at each time step. $t_0$ is the initial time, and the time integration is conducted up to $t_5$. PD data at two time points are required to determine a boundary condition for the CD. The boundary condition at an intermediate time, such as $t_{1/2}$, is calculated with linear interpolation, using the data at these two time points. Thus, the CD process waits at the initialization stage until the PD data at $t_0$ and $t_1$ are transferred to the CD process. The time integration in the PD between $t_1$ and $t_2$ is overlapped with that in the CD between $t_0$ and $t_1$. The inter-domain communication with the PD processes at $t_2$ is also overlapped with the time integration in the CD between $t_0$ and $t_1$. Since the receiving cue is issued before the time integration, the PD data can be received by the CD processes in the background, as indicated by “rv cueing” in Fig. 7. Thus, the two sets of time integration procedures in the two domains are conducted in parallel. The PD processes must wait to receive the final boundary data from the CD processes before finalizing the program.

To run continuously without any pause in the integration of the CD, the sum of elapsed times for the time integration of the PD and the inter-domain communications should be shorter than the elapsed time for the time integration of the CD. The user can satisfy this condition by setting the process numbers assigned to each domain.

*2.5. For flexible experimental settings*

To ensure flexible settings in online nesting, we use a three-dimensional linear interpolation. This allows us to set different vertical grid arrangements for each domain. The vertical interpolation is also defined by (1) and (2); the default setting is $n = 2$ for vertical interpolation.

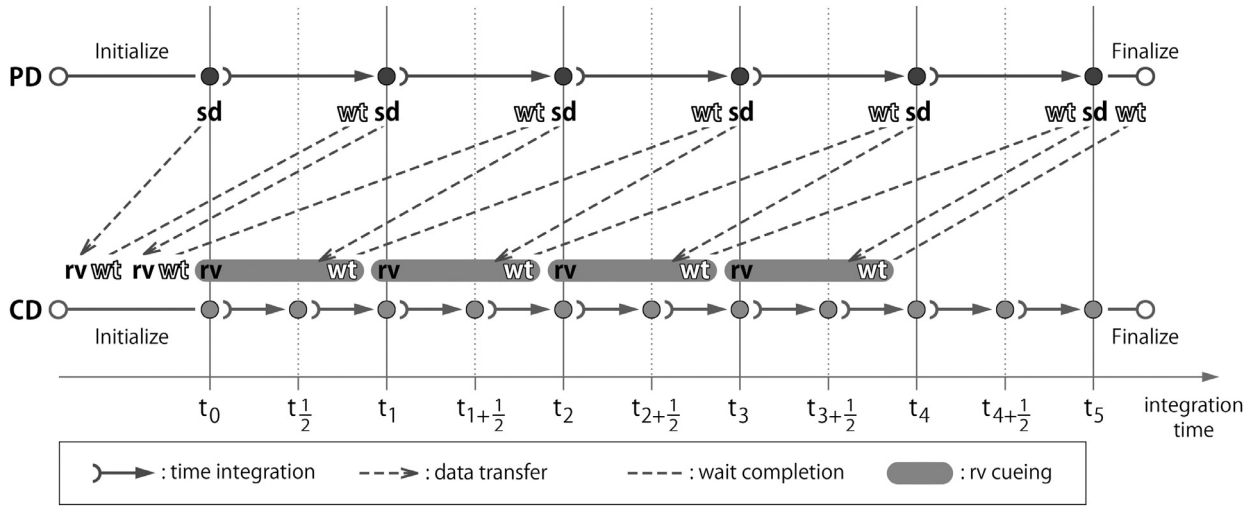


**Fig. 7.** Time diagram of inter-domain communication. The first row is for the parent domain (PD) and the second row is for the child domain (CD). "sd" is the send cue, "rv" is the receive cue, and "wt" is the wait instruction. The ellipses that follow the words 'rv cueing' mean that the receive cue is waiting in the background of the computation.

**Table 1**
Experimental settings for performance assessment with triply-nested domains.

| | Domain 1 | Domain 2 | Domain 3 |
|---|---|---|---|
| Grid space | 27 km | 9 km | 3 km |
| Grid number (nx) | 80 | 80 | 80 |
| Grid number (ny) | 80 | 80 | 80 |
| Vertical levels (nz) | 48 | 64 | 80 |
| Time interval | 27 s | 9 s | 3 s |
| Integration period | 1350 s | 1350 s | 1350 s |
| Time steps | 50 | 150 | 450 |

Since the horizontal interpolation depends only on latitude/longitude data, users can freely set the domain locations and/or the map-projection types for each domain, as long as the CD is located on the inner PD. To interpolate between the different map projections, vector variables such as horizontal wind components ($u(x, y)$, $v(x, y)$) are given special treatment. The vector variables are rotated on a local Cartesian grid system, having been converted from the map-projected grid. Thus, information on the map projection about other domains is not required. The horizontal vector components are converted using relational expressions as follows.

$$\begin{pmatrix} u_{cart} \\ v_{cart} \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} u_{map} \\ v_{map} \end{pmatrix},$$

where $u_{cart}$ and $v_{cart}$ are the west–east component and the south–north component, respectively, on the local Cartesian grid system; $u_{map}$ and $v_{map}$ are the same as $u_{cart}$ and $v_{cart}$ but on the map-projected grid system; and $\alpha$ is a rotation angle between two grids, which is formed by the x-axis on the map-projected grid and that on the Cartesian grid at the same location.

## 3. Test experiment of the new implementation

We applied CONeP to SCALE-RM [10,11] and tested its performance using the model. SCALE-RM is based on the finite volume method and uses a cuboid control volume with the Arakawa-C grid system. The K computer was used for performance measurements and its peak performance reaches to ten petaflops within the full system. The computational performances were measured using the profiler of the Parallelnavi [25,26]. We measured the elapsed time of CNV as the sum of those for all domains, based on Michalakes [15].

### 3.1. Experimental settings

We conducted an experiment to assess performance. Details of the experimental settings are shown in Table 1. The computational domains were triply nested. The nesting relationship was the same as that shown in Fig. 1. The region of

**Table 2**
Setting for the MPI process distribution for parallel computation. "CNV" refers to the conventional procedure and "CONeP" to the new implementation.

| | CNV-d1 | CNV-d2 | CNV-d3 | CONeP-d1 | CONeP-d2 | CONeP-d3 |
|---|---|---|---|---|---|---|
| Node number (mx) | 10 | 10 | 10 | 2 | 4 | 8 |
| Node number (my) | 10 | 10 | 10 | 2 | 4 | 10 |
| Total nodes | 100 | 100 | 100 | 4 | 16 | 80 |
| Grid num in a tile (lx) | 4 | 6 | 10 | 40 | 20 | 10 |
| Grid num in a tile (ly) | 4 | 6 | 10 | 40 | 20 | 8 |

**Table 3**
Elapsed time for 1350-s time integration for a triply-nested case.

| For 1350-s time integration | CNV | CONeP |
|---|---|---|
| Elapsed time for inter-comms | 0.4 s | 0.5 s |
| Elapsed time for computation: d1 | 1.4 s | Behind (8.1 s) |
| Elapsed time for computation: d2 | 4.5 s | Behind (10.1 s) |
| Elapsed time for computation: d3 | 14.5 s | 15.6 s |
| Elapsed time for others | 0.5 s | 0.5 s |
| Total elapsed time | 21.3 s | 16.6 s |

Domain 1 had an area of $2160 \times 2160$ km, and Domain 2 an area of $720 \times 720$ km within Domain 1. In a similar fashion, Domain 3 had an area of $240 \times 240$ km, embedded within Domain 2. The computational space is defined by the three-dimensional Cartesian coordinates in the SCALE-RM model. The total numbers of grid points ($nx \times ny \times nz$, where $nx$, $ny$, and $nz$ are the longitudinal, meridional, and vertical directions, respectively) are 307,200, 409,600, and 512,000 for Domains 1, 2, and 3, respectively. To compare the performance of CONeP to that of CNV, the experimental settings were configured regarding the process distribution, as shown in Table 2. A single process was assigned to a single computational node, and 100 nodes were used in both the CNV and CONeP cases. It is important that the number of computational nodes be the same in the two experimental cases for a fair comparison. If the elapsed time of the simulation with CONeP is shorter when using the same number of computational nodes as with CNV, the leading time against the wall clock (i.e., the available period, as forecasted) becomes longer in the numerical weather prediction model. Since the computational nodes are distributed over the domain by means of a two-dimensional mapping, as shown in Fig. 3, the grid number in a cell in Table 2 is defined by the grid number/node number on each axis. Thus, the grid numbers in the domain cell differ for the two experimental cases. A mapping of computer nodes should be arranged similarly to the mapping of the domain decomposition. This is because neighboring communication in the intra-domain is executed as a data transfer without node hopping in such a node mapping. Since the network topology of the computational nodes on the K computer is a six-dimensional mesh/torus fusion (Tofu) interconnect topology [27], this is achieved with a two-dimensional node mapping. For the CNV case, the node mapping is specified as $(mx, my) = (10,10)$. For the CONeP case, the node mapping should be tuned by considering Domain 3. This is because the elapsed time for Domain 3 is dominant over the other parts of the elapsed time in the new implementation. The node mapping for the 100 nodes is specified as $(mx, my) = (8,13)$ to retain a similarity to the domain decomposition in the nodes of (8,10) in Domain 3, and computational nodes for Domains 1 and 2 are provided from the remaining nodes of (8,3). Although 104 ($= 8 \times 13$) total nodes are assigned to the job, four nodes are not used in the program execution. With these settings, the neighboring communication in the intra-domain is executed as a data transfer without node hopping in all domains for the CNV case, and in Domain 3 for the CONeP case.

In this evaluation, the computational performance was measured only in the dynamical core of the numerical model. The effects of physics schemes, such as cloud microphysics, are discussed in Section 3.3. Note that the initialization part of the program was eliminated from the performance measurement. The experiments were conducted nine times for each case, and the results contained only the minimum elapsed times, i.e., maximum performances, in the experimental runs to avoid contamination by the run-to-run variability. To examine the largest load case for inter-domain communication in this study, the inter-domain communication and corresponding interpolation of CONeP was implemented so that all fields of PD data in the domain cell were transferred to the CD.

### 3.2. Experimental results

The elapsed times for a 1350 s time integration are shown in Table 3. The total elapsed time for the CNV case was 21.3 s, while it was 16.6 s for the CONeP case, approximately 22% faster than CNV with the same number of computational nodes. Thus, CONeP has better computational efficiency than the conventional process. As a reference, we measured the sole domain case for Domain 1 region by Domain 3 grid space, the elapsed time for a 1350 s time integration was 354.0 s by using 100 nodes.

In the CNV case, the elapsed times for computation and inter-domain communication were measured as individual executions in each domain, i.e., as single domain runs. In this experiment, two types of inter-domain communications were

**Table 4**
Measured performance on the K computer.

| Experiment | Performance (MFLOPS/node) | Peak rate (%) |
|---|---|---|
| CNV: domain 1 | 7398 | 1.2 |
| CNV: domain 2 | 9286 | 1.5 |
| CNV: domain 3 | 10,748 | 1.7 |
| CONeP: domain 1 | 23,074 | 4.5 |
| CONeP: domain 2 | 23,565 | 3.7 |
| CONeP: domain 3 | 12,291 | 1.9 |

conducted: from Domain 1 to Domain 2 (d1 d2) and from Domain 2 to Domain 3 (d2 d3). The elapsed times of inter-domain communication for one time step were measured as 0.0020 s in d1 d2, and 0.0021 s in d2 d3, according to the interval measurement. For 1350 s integration, inter-domain communications in d1 d2 and d2 d3 were executed 50 and 150 times, respectively. Therefore, the total elapsed time of inter-domain communication was $(0.0020 \times 50) + (0.0021 \times 150) = 0.42$ s.

In CONeP, the elapsed time of the inter-domain communication should be hidden, ideally by that of the computation. In our experiment, although the elapsed time was sufficiently short, it was not zero; it was measured as 0.5 s by the interval measurement (Table 3). This finding is mainly attributable to the interpolation process and memory copying for updating the lateral boundary condition. The elapsed times for others in Table 3 include the entire process except for time integration calculations and communications, e.g., log file output. We assumed that the elapsed times for others in CNV are exactly the same as those in CONeP because both use the framework of the SCALE-RM model.

### 3.3. Discussion of computational performance

In this study, we discuss computational performance by considering the difference in total elapsed time between CNV and CONeP. The computational performance for each domain was measured on the K computer using the profiler of the Parallelnavi. Table 4 shows the computational performances of CNV and CONeP. This result clearly reveals that CONeP outperformed CNV in all domains. Owing to the appropriate assignment of the node numbers to each domain in CONeP, the computational performance is maintained in the outer domains. On the other hand, in CNV, the computational performances for Domains 1 and 2 are particularly low. This degradation of computational performance derives from the difficulty of strong scaling for the stencil calculation, as described in Section 1.

The performance degradation becomes critical when the number of nesting domains increases. We assumed that the inner domain has more grid points and higher spatial resolution than the outer domain (i.e., the inner domain has a larger problem size than the outer domain). When another innermost domain was added to calculate with higher resolution, the total problem size became dramatically large. To perform the calculation within the shortest elapsed time possible, we would use a greater number of nodes than in the previous configuration. For CNV, the computational performance in the outer domains may degrade as a result of assigning too many nodes for a small problem size. In contrast, for CONeP, the node assignment for pre-existing domains is not changed, and the extra nodes should be assigned to the newly added domain. If sufficiently many nodes are allocated in order to calculate the newly added domain within a similar elapsed time with the other domains, we can add higher resolution domains to the experiment without increasing the elapsed time in CONeP.

For example, if we add a fourth domain to the interior of Domain 3, the performance degradation would become greater in CNV. To demonstrate the quad-nesting case, we conducted another experiment. The details of these experimental settings are shown in Table 5. Considering this quad-nesting case with 576 nodes on the K computer, the total elapsed time for CNV is 62.6 s, and that of CONeP is 43.3 s (Table 6). Thus, the elapsed time for CNV is approximately 31% shorter than that for CONeP. Therefore, CONeP has a greater advantage with an increased number of nested domains.

Considering the recent trends of HPC architecture, the improved interconnect speed is slower than the computational speed. CNV is more significantly affected by the relative degradation of interconnect speed than CONeP because the ratio of intra-domain communications to computations in CNV is greater than that in CONeP. In short, the difference between the elapsed times in CONeP and CNV will be greater in consideration of strong scaling. Thus, CONeP is a more promising method than CNV.

In the previous discussion, we assumed that the inner domain has a larger problem size than the outer domain. However, the advantages of CONeP can be maintained in principle when the outer domain has a larger problem size than the inner domain, or when the intermediate domain has the largest problem size. CONeP is free from such problems. We can assign the proper node numbers for each domain by considering the problem size.

Considering the future of HPC, inter-domain communications should be optimized, depending on the network topology and speed. Potential exists for further improvements in our online nesting procedure. One possibility is node mapping. For example, in the Tofu interconnect topology, the elapsed time for inter-domain communications could be reduced by stacking the nodes assigned to Domain 2 onto those assigned to Domain 1, instead of placing them side-by-side. The distances between computational nodes are shorter than in the side-by-side case, and the collision frequency between intra-domain and inter-domain communications can be decreased by using different axes of the dimensions. In a fat-tree topology, the

**Table 5**
Experimental settings and settings for the MPI process distribution for performance assessment with quad-nested domains.

| | Domain 1 | Domain 2 | Domain 3 | Domain 4 |
|---|---|---|---|---|
| Grid space | 27 km | 9 km | 3 km | 1 km |
| Grid number (nx) | 120 | 120 | 120 | 120 |
| Grid number (ny) | 120 | 120 | 120 | 120 |
| Vertical levels (nz) | 48 | 64 | 80 | 96 |
| Time interval | 27 s | 9 s | 3 s | 1 s |
| Integration period | 1350 s | 1350 s | 1350 s | 1350 s |
| Time steps | 50 | 150 | 450 | 1350 |

| CNV | Domain 1 | Domain 2 | Domain 3 | Domain 4 |
|---|---|---|---|---|
| Node number (mx) | 24 | 24 | 24 | 24 |
| Node number (my) | 24 | 24 | 24 | 24 |
| Grid # in a tile (lx) | 5 | 5 | 5 | 5 |
| Grid # in a tile (ly) | 5 | 5 | 5 | 5 |

| CONeP | Domain 1 | Domain 2 | Domain 3 | Domain 4 |
|---|---|---|---|---|
| Node number (mx) | 2 | 4 | 12 | 24 |
| Node number (my) | 4 | 4 | 6 | 20 |
| Grid # in a tile (lx) | 60 | 30 | 10 | 5 |
| Grid # in a tile (ly) | 30 | 30 | 20 | 6 |

**Table 6**
Elapsed time for 1350-s time integration for a quad-nesting case.

| For 1350 s time integration | CNV | CONeP |
|---|---|---|
| Elapsed time for inter comms | 1.6 s | 1.2 s |
| Elapsed time for computation: d1 | 1.3 s | Behind (9.3 s) |
| Elapsed time for computation: d2 | 4.4 s | Behind (17.7 s) |
| Elapsed time for computation: d3 | 14.1 s | Behind (23.2 s) |
| Elapsed time for computation: d4 | 40.8 s | 41.7 s |
| Elapsed time for others | 0.4 s | 0.4 s |
| Total elapsed time | 62.6 s | 43.3 s |

aggregation layer can be separated between intra-domain and inter-domain communications, and the frequency of communication congestion can be decreased by allocating different branches of the tree to the different domains.

Another possibility for improving its performance involves careful implementation of data transfer in inter-domain communications. In inter-domain communications, all PD data over the domain are transferred to the CD in the SCALE-RM model, as described in Section 3.1, to examine the worst-case scenario. In fact, only PD data corresponding to the boundary and nudging areas in the CD are required for data transfer. In this case, the data size is reduced by transferring only the necessary data, so that the speed of inter-domain communications is increased.

To date, we did not include physics schemes, such as the cloud microphysics scheme, in our performance measurements. Since most physics schemes do not have dependencies in the horizontal direction, higher scalability for computation in areas other than the dynamical core is expected in physics schemes. However, the scalability of physics schemes worsens with larger nodes because of load imbalance (e.g., [28]). Based on architecture trends for the future of HPC, improvements in memory or communication speed will be less significant, compared to improvements in computational performance. The weight of the dynamical core in the total elapsed time will increase in the future. This is the reason why we focused only on an estimation for the dynamical core in this study.

## 4. Conclusions

We proposed a cost-effective online nesting procedure (CONeP), which has great advantages over the conventional procedure of online nesting (CNV). These advantages include an update interval for the data and the lack of a need for an intermediate file. As shown in this paper, the CNV is not suitable for cases with multiple nested domains on a massively parallel computer from the viewpoint of computational efficiency.

In CONeP, the processes are split into multiple sub-communicators, each assigned to a different domain. The time integration in each domain is conducted in parallel by multiple sub-communicators. The computational performance of CONeP is higher than that of CNV because the process parameters can be tuned specifically for each domain in CONeP. The performance improvement was assessed by measuring the elapsed time for ideal experiments. In the quad-nested case using 576 nodes (processes), the total elapsed time of CONeP was 31% shorter than that of CNV.

Considering the future of HPC, internode communication speed will likely increase to a lesser degree than computation speed, and strong scaling will become more difficult. Thus, the framework of CONeP is necessary for adjusting process numbers according to the problem size of each domain. This framework has also advantages for flexible settings with domain location, vertical grid arrangement, and map projection selection. This virtue is inherited from the offline nesting method. In this sense, we can say that CONeP is a promising nesting procedure, directed toward the future of HPC.

## Acknowledgments

We are grateful to Drs. Motohiko Matsuda, Masayuki Hatanaka, and Kazumi Yoshinaga for valuable insights. Thanks also to the members of the Computational Climate Science Research Team, RIKEN AICS, for their input. The experiments were performed using the K computer at the RIKEN Advanced Institute for Computational Science. This work was supported by JST CREST Grant Number JPMJCR1312, Japan, and FOCUS Establishing Supercomputing Center of Excellence.

## References

[1] A. Sathye, M. Xue, G. Bassett, K. Droegemeier, Parallel weather modeling with the advanced regional prediction system, Parallel Comput. 23 (1997) 2243–2256.
[2] M. Xue, K.K. Droegemeier, V. Wong, The Advanced Regional Prediction System (ARPS)-a multiscale nonhydrostatic atmospheric simulation and prediction tool. Part I: model dynamics and verification, Meteorol. Atmos. Phys. 75 (2000) 161–193.
[3] M. Xue, K.K. Droegemeier, V. Wong, A. Shapiro, K. Brewster, F. Carr, D. Weber, Y. Liu, D. Wang, The Advanced Regional Prediction System (ARPS)-a multiscale nonhydrostatic atmospheric simulation and prediction tool. Part II: model physics and applications, Meteorol. Atmos. Phys. 76 (2001) 143–165.
[4] G.J. Tripoli, W.R. Cotton, The Colorado State University three-dimensional cloud/mesoscale model-1982. Part I: general theoretical framework and sensitivity experiments, J. de Rech. Atmos. 16 (1982) 185–220.
[5] W.R. Cotton, R.A. Pielke Sr., R.L. Walko, G.E. Liston, C.J. Trembeck, H. Jiang, R.L. McAnelly, J.Y. Harrington, M.E. Nicholls, G.G. Carrio, J.O. McFadden, RAMS 2001: current status and future directions, Meteorol. Atmos. Phys. 82 (2003) 5–29.
[6] J. Dudhia, A nonhydrostatic version of the Penn State–NCAR mesoscale model: validation tests and simulation of an Atlantic cyclone and cold front, Mon. Wea. Rev. 121 (1993) 1493–1513.
[7] J. Michalakes, MM90: A scalable parallel implementation of the Penn State/NCAR mesoscale model (MM5), Parallel Comput. 23 (1997) 2173–2186.
[8] W.C. Skamarock, J.B. Klemp, J. Dudhia, D.O. Gill, D.M. Barker, M.G. Duda, X.-Y. Huang, W. Wang, J.G. Powers, A Description of the Advanced Research WRF Version 3, 2008, p. 88. NCAR Tech. Note, NCAR/TN-4681STR.
[9] K. Saito, T. Fujita, Y. Yamada, J. Ishida, Y. Kumagai, K. Aranami, S. Ohmori, R. Nagasawa, S. Kumagai, C. Muroi, T. Kato, H. Eito, Y. Yamazaki, The operational JMA nonhydrostatic mesoscale model, Mon. Wea. Rev. 134 (2006) 1266–1298.
[10] Y. Sato, S. Nishizawa, H. Yashiro, Y. Miyamoto, Y. Kajikawa, H. Tomita, Impacts of cloud microphysics on trade wind cumulus: which cloud microphysics processes contribute to the diversity in a large eddy simulation? Prog. Earth Planet. Sci. 2 (2015) 23, doi:10.1186/s40645-015-0053-6.
[11] S. Nishizawa, H. Yashiro, Y. Sato, Y. Miyamoto, H. Tomita, Influence of grid aspect ratio on planetary boundary layer turbulence in large-eddy simulations, Geosci. Model Dev. 8 (2015) 3393–3419, doi:10.5194/gmd-8-3393-2015.
[12] M. Dimitrijevic, R. Laprise, Validation of the nesting technique in a regional climate model and sensitivity tests to the resolution of the lateral boundary conditions during summer, Clim. Dyn. 25 (2005) 555–580.
[13] B. Denis, R. Laprise, D. Caya, Sensitivity of a regional climate model to the resolution of the lateral boundary conditions, Clim. Dyn. 20 (2003) 107–126.
[14] C.F. Mass, Y.H. Kuo, Regional real-time numerical weather prediction: current status and future potential, BAMS 79 (1998) 253–263.
[15] J. Michalakes, RSL: a parallel runtime system library for regional atmospheric models with nesting, in: To Appear in Proceedings of the IMA Workshop Structured Adaptive Mesh Refinement Grid Methods, March 12-13, Minneapolis, 1997, 1997 Also available from ftp://ftp.mcs.anl.gov/chammp/IMA_RSL.ps.gz.
[16] Z.I. Janjic, R. Gall, Scientific Documentation of the NCEP Nonhydrostatic Multiscale Model on the B Grid (NMMB). Part 1: Dynamics, 2012, p. 75. NCAR Tech. Note, NCAR/TN-4891STR.
[17] D. DiMego Geoff, Z. Janjic, T. Black, E. Rogers, M. Pyle, J. Du, H.-Y. Chuang, B. Ferrier, D. Jovic, Update on WRF in NCEP operations, The 12th Annual WRF Users' Workshop, 2011 available from http://www2.mmm.ucar.edu/wrf/users/workshops/WS2011/WorkshopPapers.php; a detail explanation is also available from ftp://ftp.emc.ncep.noaa.gov/wd20vxt/NMMB-tutorial/NMMB_tutorial_nesting.pptx.
[18] MPI-Forum, MPI: A message-passing interface standard version 2.1, 2008. Available from http://www.mpi-forum.org/docs/mpi-2.1/mpi21-report.pdf.
[19] N.A. Phillips, J. Shukla, On the strategy of combining coarse and fine grid meshes in numerical weather prediction, J. Applied Met. 12 (1973) 763–770.
[20] T.L. Clark, R.D. Farley, Severe downslope windstorm calculations in two and three spatial dimensions using anelastic interactive grid nesting: a possible mechanism for gustiness, J. Atmos. Sci. 41 (1984) 329–350.
[21] D.L. Zhang, H.R. Chang, N.L. Seaman, T.T. Warner, J.M. Fritsch, A two-way interactive nesting procedure with variable terrain resolution, Mon. Wea. Rev. 114 (1986) 1330–1339.
[22] HC. Davies, A lateral boundary formulation for multi-level prediction models, Q. J. R. Meteorol. Soc. 102 (1976) 405–418.
[23] L.M. Harris, D.R. Durran, An idealized comparison of one-way and two-way grid nesting, Mon. Wea. Rev. 138 (2010) 2174–2187.
[24] C. Soriano, O. Jorba, J.M. Baldasano, One-way nesting versus two-way nesting: does it really make a difference? in: Air Pollution Model.ing and Its Application XV, 2002, pp. 177–185, doi:10.1007/0-306-47813-7_18.
[25] K. Ida, Y. Ohno, S. Inoue, K. Minami, Performance profiling and debugging on the K computer, FUJITSU Sci. Tech. J. 48 (No. 3) (2012) 331–339.
[26] Fujitsu, 2013: Parallelnavi for MP10 V1.0.
[27] Y. Ajima, S. Sumimoto, T. Shimizu, Tofu: A 6D Mesh/Torus interconnect for exascale computers, IEEE Comput. 42 (11) (2009) 36–40.
[28] M. Terai, H. Yashiro, K. Sakamoto, S. Iga, H. Tomita, M. Satoh, K. Minami, Performance optimization and evaluation of a global climate application using a 440 m horizontal mesh on the K computer, A Proceeding of the SC14 at New Orleans, 2014.