Abstract—We develop a communication reduced multi-time-step (CRMT) algorithm for a Lattice Boltzmann method (LBM) based on a block-structured adaptive mesh refinement (AMR). This algorithm is based on the temporal blocking method, and can improve computational efficiency by replacing a communication bottleneck with additional computation. The proposed method is implemented on an extreme scale airflow simulation code CityLBM, and its impact on the scalability is tested on GPU based supercomputers, TSUBAME and Reedbush. Thanks to the CRMT algorithm, the communication cost is reduced by ~64%, and weak and strong scalings are improved up to ~200 GPUs. The obtained performance indicates that real time airflow simulations for about 2km square area with the wind speed of ~5m/s is feasible using 1m resolution.

Index Terms—Locally Mesh Refinement, Communication Reduced Multi-time-step Algorithm, GPU, Wind Simulation

I. INTRODUCTION

It is very important for nuclear security to predict environmental dynamics of radioactive substances based on detailed simulations of airflows and the resulting plume dispersion. This kind of airflow simulations in urban areas are important tools also in designing smart cities. So far, large eddy simulations (LESs) based on the Navier-Stokes (NS) equation have been widely used in airflow analysis and climate analysis [1], [2]. However, different types of CFD methods are emerging due to the rapid development of GPU-based supercomputers.

The lattice Boltzmann Method (LBM) is one of CFD methods that solves a discrete-velocity Boltzmann equation [3]. The flow field is expressed by pseudo particles, which are convected in a limited number of directions, and the macroscopic values of the density and the velocity are not directly calculated unlike NS solvers. The discretized velocity vectors of the LBM are defined on each Cartesian grid, so that they satisfy conservation laws and Galilean invariance of the NS equation. The LBM was originally developed for direct numerical simulations at low Reynolds number. Thanks to sub-grid scale models and state-of-the-art collision operators [4], [5], it is possible to realize turbulent flow simulation at high Reynolds number. Recently, such turbulent analysis is successfully applied to airflow analysis using a realistic urban geometry [6]. The advantage of the LBM is that it is suitable for high performance computing because of simple algorithm with continuous memory accesses [7], [8]. In Refs. [9] and [10], petascale CFD simulations based on LBM were reported.

Recently, GPU architecture has received great attentions for high performance computing because of simple algorithm for real-time wind prediction is of critical importance for nuclear security, the GPU acceleration of CFD codes is an attractive option. However, it is difficult to perform multiscale CFD simulations with a uniform grid from the viewpoint of computational resources and calculation time. To resolve this issue, we developed an adaptive mesh refinement (AMR) method for the LBM [8].

The AMR method is one of key techniques to accelerate multi-scale simulations [12]–[14]. Since fine grids are arranged only in a region, where smaller scale flows develop, the number of grid point can be reduced dramatically. In addition to the AMR-LBM with a fixed time step width [15], [16], a multi-time-step (MT) algorithm is also available. Although the MT algorithm can reduce the number of time steps in coarser grid regions, it requires additional halo data communications at interfaces of regions with different grid resolutions. This leads to a severe communication bottleneck on GPU platforms.

As a solution for such overhead, a temporal blocking (TB) method [17]–[19], which optimizes data structures to reduce the number of halo data communication, were proposed. This paper proposes an acceleration algorithm for the AMR-LBM, named a communication reduced multi-time-step (CRMT) algorithm. The CRMT algorithm can reduce the number of halo data communication as in the TB method. This feature is of critical importance for extreme scale real-time simulation on future exascale supercomputers where a gap between the interconnect bandwidth and FLOPS may further increase.

In this work, we apply the CRMT algorithm to the CityLBM code. We optimize MPI communication on the GPU-based supercomputers, and compare the performances of the intra-node communication through the NVLink and the inter-node communication through the Infiniband or the Omni-path. We also perform weak and strong scalability tests to evaluate the
possibility of real time simulation.

II. CITYLBM CODE

A. Lattice Boltzmann Method

The LBE is obtained by reducing the Boltzmann equation into a finite number of discrete velocities. The fluid flow is expressed by pseudo particles on a uniform Cartesian grid, which are convected with a limited number of velocities. Since the speed of the particles is normalized by the sound speed, the fluid is assumed to be weakly compressible. The LBE consists of the streaming and collision processes with an explicit time integration. Since pseudo particles move onto the neighbor lattice points after one time step in the streaming process, this process is completed without any error. The time evolution of the discretized velocity function is

\[ f_{ijk}(x_{ijk} + c_{ijk} \Delta t, t + \Delta t) = f_{ijk}(x_{ijk}, t) + \Omega_{ijk}. \]

Here, \( \Delta t \) is the time interval, \( c_{ijk} \) is the lattice vector of the pseudo particle, \( f_{ijk} \) is the velocity distribution function corresponding to the lattice vector \( c_{ijk} \), \( x_{ijk} \) is the configuration space, \( \Omega_{ijk} \) is the collision operator.

The lattice BGK model is widely used because of its simplicity \cite{3}. The collision operator of a single relaxation time (SRT) model is defined as

\[ \Omega_{ijk} = \frac{1}{\tau} \left( f_{ijk}(x_{ijk}, t) - f_{eq}^{ijk}(x_{ijk}, t) \right), \]

where \( \tau \) is relaxation time, and \( f_{eq}^{ijk} \) is a local equilibrium distribution function. We assume isothermal and low Mach number. The equilibrium distribution function is given as

\[ f_{eq}^{ijk} = \omega_{ijk} \rho \left( 1 + \frac{3c_{ijk} \cdot u}{c^2} + \frac{9(c_{ijk} \cdot u)^2}{2c^4} - \frac{3u^2}{2c^2} \right). \]

Here, \( \rho \) is the density, \( c \) is the sound speed, \( u \) is macroscopic velocity, and \( \omega_{ijk} \) is weighting factor.

It is important to choose a proper lattice velocity model by taking account of the trade off between efficiency and accuracy. The D3Q27 model is suitable for high Reynolds number flow because of its advantages in terms of accuracy and spatial symmetry in turbulent flow analysis \cite{20}.

Various collision models have been proposed to improve accuracy and stability. The single relaxation time (SRT) model is the most popular model, because of its simple formulation and low computational cost. One of problems of the SRT model is a numerical instability in turbulent flow analysis. The SRT model requires hyper viscosity or subgrid-scale models with excessive viscosity \cite{21, 22} to suppress unphysical oscillations, but the excessive eddy viscosity often makes the results diffusive.

The cumulant relaxation time (CRT) model \cite{5} is a promising approach to solve the above problems. In the CRT model, the collisional relaxation process is formulated for cumulant variables, and multiple relaxation times are introduced for each cumulant variable. Each decay process can be modeled by an individual rate equation as follows,

\[ C_{\alpha\beta\gamma}^{*} = \omega_{\alpha\beta\gamma}^{CRT} C_{\alpha\beta\gamma}^{eq} + (1 - \omega_{\alpha\beta\gamma}^{CRT}) C_{\alpha\beta\gamma}. \]

Here, \( \omega_{\alpha\beta\gamma}^{CRT} \) is the relaxation frequency, and \( C_{\alpha\beta\gamma}^{eq} \) is the value of each cumulant.

B. Large-eddy Simulation

A large-eddy simulation (LES) resolves the flow dynamics of large-scale structures on a grid scale (OG), and it takes into account the effect of subgrid-scale (SGS) turbulent structures by using eddy viscosity as

\[ \nu_{SGS} = C \tilde{\Delta}^2 \left| \tilde{S} \right|. \]

Here, \( C \) is the model coefficient, \( \tilde{\Delta} \) is the filter width, and \( \tilde{S}_{ij} \) is velocity strain tensor.

In a conventional Smagorinsky model (SM), the model coefficient is a constant in the entire computational domain, and the SGS viscosity does not describe the correct asymptotic behavior near a wall. The dynamic Smagorinsky model (DSM) \cite{21} overcomes this defect by dynamically calculating the model parameter by using two types of grid filter. Although DSM is the most notable breakthrough in LES, the model parameter requires the average over the global domain for the sake of numerical stability. This introduces a large overhead of large-scale simulations.

The coherent structure model (CSM) \cite{22} is a promising approach suitable for large-scale simulations. The model coefficient \( C_{CSM} \) is locally calculated by the second invariant of the velocity gradient tensor \( Q \) and the magnitude of the velocity gradient tensor \( E \) as

\[ C_{CSM} = C'' \left( \frac{Q}{E} \right)^2, \quad (-1 \leq \frac{Q}{E} \leq 1). \]

Since the CRT model has both numerical accuracy and stability, excessive numerical viscosity is not required. The coefficient \( C'' = 1/25 \) is a fixed model parameter, and it is optimized for turbulent wind simulations. In this work, we adopt the CSM to resolve subgrid-scale phenomena.

III. ADAPTIVE MESH REFINEMENT (AMR) METHOD

A. AMR Method for LBM

The LBM is suitable for modeling boundary conditions with complex shapes. The boundary condition is represented by a level set function, and a pseudo particle is reflected on a wall by using the interpolated bounce-back (IBB) method \cite{23}. Although the IBB method can calculate the non-slip boundary condition well, it is not possible to directly model the turbulent boundary layer. Since the scale of vortices near the wall is very small at high Reynolds number, a lot of grid points are required in the entire calculation domain.

The AMR method is one of key techniques to accelerate multi-scale simulations \cite{12, 14, 24}. High-resolution grids are arranged only in a region with fine scale flows, and the number of grid points can be reduced dramatically. The LBM is a dimensionless method in time and space. The time step width \( \Delta t \) is determined to satisfy \( c \Delta t / \Delta x = 1 \). The kinematic viscosity and the relaxation time depend on the time step.
width. To keep a constant viscosity on coarse and fine grids, the relaxation time $\tau$ is defined by the following expression,

$$\left(\frac{\tau_f - \frac{1}{2}}{\frac{1}{2}}\right) = m \left(\frac{\tau_c - \frac{1}{2}}{\frac{1}{2}}\right).$$

(7)

Here the super- and sub-scripts $c$ and $f$ denote the value of the coarse and fine grids, respectively. The coefficient $m$ is the refinement factor, which is usually set to 2 for stability and simplicity reasons. To ensure the continuity of the stress tensor between two resolutions, the distribution function is defined as

$$f_{ijk}^f = \left(1 - \frac{\tau_f - \frac{1}{2}}{m \tau_c - \frac{1}{2}}\right) f_{ijk}^{eq} + \frac{1}{m \tau_c - \frac{1}{2}} f_{ijk}^f,$n

$$f_{ijk}^c = \left(1 - \frac{\tau_c - \frac{1}{2}}{\tau_f - \frac{1}{2}}\right) f_{ijk}^{eq} + \frac{\tau_c - \frac{1}{2}}{\tau_f - \frac{1}{2}} f_{ijk}^f.$$

(8)

Here, the equilibrium distribution function $f_{ijk}^{eq}$ should be the same both on the fine- and coarse-grid points.

B. Data Structure and Grid Arrangement for GPU Computing

Data structure of the AMR is very important from the viewpoint of computational efficiency [25]. Especially it is important to choose a proper AMR model by taking account of the trade off between adaptivity to physical phenomena and continuity of memory accesses. A cell-based AMR method subdivides a whole domain per cell unit, and it is the most efficient method for memory consumption. Meanwhile, memory accesses become discontinuous, so that its computational performance becomes worse. A block-structured AMR has efficient data structure, and this method subdivides whole domain by block units. Since several cells are contained in a block unit, it enables continuous memory accesses [26], [27] and is suitable for a GPU architecture.

We developed the block-structured AMR method optimized for a GPU-based supercomputer. The connectivity of each domain is managed based on a forest-of-octrees approach. One domain called leaf contains $N_{\text{leaf}}^3$ cells, and the memory access in the leaf is the same as the Cartesian based LBM. The computational domain is composed of many leaves, and the leaf itself has connections to neighbors, a parent, and children in an unstructured manner. As each leaf uses different $\Delta t$ depending on the resolution, the boundary leaves at the interface of different resolution are computed by interpolating the leaves at the other level following Ref. [27]. In parallel computing, MPI communications are needed for this boundary treatment.

It is important where to define values because it is greatly related to the calculation procedure at the interface of different resolution. We define values on the cell center, and this method does not share the value at the intersection of grids. The value on the coarse- and fine-grid in Eq. (5) is defined for different regions, and it is necessary to construct an interpolation function. For interpolating a coarse grid value from fine grid values, values are simply averaged over the coarse grid region. On the other hand, for interpolating a fine grid value from coarse grid values, values are spatially interpolated by a linear interpolation function which satisfies mass conservation.

IV. COMMUNICATION REDUCED MULTI-TIME-STEP ALGORITHM

A. Multi-time-step Algorithm

The LBM is normalized by sound speed, and the ratio of the time step width and the spatial grid width is fixed [15], [16]. While it enables efficient time integration using different time step widths for each resolution, the calculation algorithm becomes complicated. Figure 1 illustrates the flowchart of the computational procedure on coarse and fine grids. Since the physical quantities depend on values at different resolution, it is necessary to perform temporal evolution in an appropriate order. At first, streaming and collision terms are calculated independently at $Lv.1$ and $Lv.0$. Before $Lv.1$ calculation starts at time $t + \Delta t/2$, boundary values are interpolated from $Lv.0$ grids at time $t$ and $t + \Delta t$, and the streaming and collision are calculated at $Lv.1$. Finally, the boundary values are interpolated and exchanged between $Lv.0$ and $Lv.1$. The spatio-temporal interpolations in boundary regions are executed in Figs. (b) and (d). MPI communications are executed after the computational procedures in Figs. (a) and (c). In our block-structured based AMR-LBM, halo data communications are processed in leaf units, and thus, the data size of each communication becomes larger than that in an ordinary LBM.

Figure 2 shows a pseudo code of multi-time step algorithm shown in Fig. 1. Each function name corresponds to the item in Fig. 1. This function supports multi-resolution calculation of AMR with two levels or more by applying it recursively.

B. Communication Reduced Multi-time-step Algorithm

The multi-time-step algorithm is an efficient method that can reduce the computational memory and the number of time steps. However, this method leads to an increase in traffic of MPI communications in parallel computation. Recently, the computing power of FLOPS has been greatly improved due
Fig. 2. Pseudo code of multi-time-step algorithm. The function named substep is called recursively.

to a dramatic increase in the number of cores. The interconnect performance is also improved, but the improvement rate is lower than the performance of FLOPS, and relative performance degradation of MPI communication is becoming a performance bottleneck [8].

The TB method [17]–[19] is an acceleration algorithm that can improve computational efficiency by replacing the communication bottleneck with additional computation. For example, the number of halo data communications can be reduced by transferring halo data for multiple time steps in advance, and performing additional computations for the extra halo data.

This paper proposes the CRMT algorithm based on the TB method. Figure 3 illustrates the schematic figure of the CRMT algorithm. The computational domain is surrounded by boundary leaves, and the MT algorithm updates the boundary values by MPI communication every step. The CRMT algorithm directly calculates multiple boundary values using the extra halo data in the boundary leaves, and MPI communication is executed only when the extra halo data runs out. Since the LBM refers one adjacent stencil in one time step, the coarse-to-fine process requires a first-order interpolation function, it is sufficient to perform MPI communication once every $N_{leaf} - 1$ steps with the leaf size of $N_{leaf}$.

V. IMPLEMENTATION AND OPTIMIZATION

A. Implementation of CityLBM Code

The CityLBM code is written in C++ and CUDA. Computational array is allocated by using the CUDA runtime API "cudaMallocManaged", which treats CPU and GPU memory space in the same pointer. The connection of the block-structured AMR method is managed by the forest-of-octree approach, and these memory structures are stored by offset indices from the start address of the pointer. Each array is based on the Array of Structures (AoS) memory layout suitable for GPU computing.

The code is parallelized by using a MPI library, and communication functions are well tuned to achieve high performance on the TSUBAME and the Reedbush supercomputers shown in Tables I and II. OpenMPI 1.10.7 and MVAPICH2-GDR are CUDA-aware MPI libraries that enable communications between GPUs directly. The computational array is copied to a communication buffer by using a GPU kernel function, and the buffer is transferred by non-blocking point-to-point communication, “MPI_Isend” and “MPI_Irecv”. Each buffer can be selected from "host", "device", and "unified" memory. Here, host memory and device memory are explicitly allocated respectively on DDR4 and on GPU, while unified memory is automatically allocated between the two memory devices by CUDA. These differences may have some impact on RDMA functions used in MPI We study this point in Sec. VI.

B. Optimization for GPU Computation

Reduction of the number of memory accesses and usage of thousands of CUDA cores are very important to speed up a GPU code. In our implementation, since the streaming and collision processes are fused, the velocity distribution functions are read and written only once in the time-integration process in Eq. (1). As a problem of kernel function, it requires much register memories, and the number of threads is limited by the usage of them per streaming multiprocessor.

As a simple solution to reduce the amount of registers, it is effective to create a kernel function for each conditional branch related to boundary conditions in advance. We create a kernel that does not require boundary conditions separately from the kernel that needs it. Each kernel can assign its own registers to calculations that the kernel actually needs, resulting in decreasing the usage of the register memory.

As an additional solution, it is effective to use shared memory. In our block-structured AMR method, CUDA threads are assigned to cells in the leaf (block) unit, and thread blocks
are assigned to leaves of the whole computational domain. One leaf refers to offsets of 26 surrounding leaves, and these indices can be shared by threads in the leaf. We replace that register memory with shared memory, and this optimization enables more CUDA threads to be launched.

VI. PERFORMANCE EVALUATION ON GPU-BASED SUPERCOMPUTERS

A. Intra-node Communication

We scan the performance of the CityLBM code with different types of communication buffers using device, host, and unified memory. Table III shows the environmental variables of TSUBAME and Reedbush, which enable GPU direct RDMA and related functions.

Table IV shows discretization parameters on a single process. One computational domain size is $320 \times 320 \times 896$ $m^3$ corresponding to two horizontal directions and a vertical direction. We compute a simulation with three refinement levels, fine resolution leaves located near the ground ($z = 0 \sim 128$ $m$), middle-resolution leaves above fine resolution leaves ($z = 128 \sim 384$ $m$), and coarse-resolution leaves at the end of the vertical domains ($z = 384 \sim 896$ $m$). The leaf size is chosen as $N_{leaf} = 4$ in all cases, and dim3(4,4,4) threads are set in a CUDA block. The whole computational domain has $2 \times 2$ sub-domains, and four processes are assigned on TSUBAME and Reedbush. We generate the computational boundary condition based on real building data of Tokyo. Inflow and outflow boundary conditions are applied at the end of horizontal domains and at the top of computational domain.

Table V shows the benchmark results for intra-node communication obtained using four GPU on a single node, consists of one step on coarse-resolution leaves, two steps on middle-resolution leaves, and four steps on fine-resolution leaves. The functions of "LBM", "AMR", and "other" are the stream-collusion function in Eq. (1), the coarse-fine interpolation of the AMR method, and the boundary conditions etc., respectively. The cost of MPI communication is measured in detail, and are divided into P2P communication ("comm"), packing to buffer memory ("pack"), unpacking from buffer memory ("unpack").

Communications through device memory got the best performance both on TSUBAME and Reedbush. The bidirectional interconnect bandwidth exceeded more than 20 GB/s, which is expected from the performance of GPU-Direct RDMA communications through the NVLink. Communications through host and unified memory showed significant performance degradation, and the cost of P2P communication increased by an order of magnitude both on TSUBAME and Reedbush. This performance degradation may be attributed to misallocation of buffer memory between host memory and device memory. The total performance was worst when host memory was used. Since host memory requires device-to-host communication in pack and unpack processes, it leads to large overheads in GPU computing.

We also compared the performance of TSUBAME and Reedbush in a single node. According to P2P communication, TSUBAME got better performance in the device memory case, while Reedbush gave better performance in the host and unified memory cases. In the unified memory case, Reedbush showed performance degradation also in pack and unpack. Since packing and unpacking functions are common between device memory and unified memory, the difference may came from the MPI library or environmental variables.

B. Inter-node Communication

Our target is large-scale CFD simulations using several computational nodes. In order to optimize inter-node parallelization, it is important to estimate the performance gap between intra-node and inter-node communications.

Hardware and software configurations are completely different between TSUBAME and Reedbush. TSUBAME has four Intel Omni-Path interconnections in a node. Reedbush has two Infiniband interconnections in a node. Theoretically, the inter-node communication performance of TSUBAME can be expected to be twice of Reedbush.
Table [VI] shows the benchmark results for inter-node communication obtained using four GPUs on four nodes (one GPU per node). Unlike the intra-node communication, each GPU is not directly connected with the NVLink. The device memory case gave the best performance both on TSUBAME and Reedbush, and is much faster than intra-node communication through host and unified memory. The performance degradation of from intra-node communication is ×0.29 on TSUBAME and ×0.50 on Reedbush, respectively. Although TSUBAME has higher inter-node peak bandwidth than Reedbush, TSUBAME showed lower performance. Under the current operation of TSUBAME, we can use only two paths for GPU-direct RDMA communication, and thus, the actual peak performance of inter-node communication is comparable between TSUBAME and Reedbush.

The host memory case showed better performance than intra-node communication both on TSUBAME and Reedbush. On the other hand, in the unified memory case, TSUBAME showed almost the same performance as intra-node communication, while Reedbush gave lower performance.

From the above results, we confirmed that device memory gives the highest intra-node and inter-node communication performances, and that the cost ratio between intra-node and inter-node communications is about 0.3 ~ 0.5. In the following numerical experiments, we use device memory.

### C. Communication Reduced Multi-time-step Algorithm

We evaluate the performance of the CRMT algorithm through device memory on four processes in a single node of TSUBAME. The computational condition is the same as the previous subsection VI-A.

Table [VII] shows the elapsed time applied with the CRMT algorithm. The function of "MPI" is sum of the elapsed time of "comm", "pack", and "unpack". The original algorithm performs halo data communications at every step, and the communication cost is about 22% of the total calculation time. The CRMT algorithm can reduce the number of communications to every $N_{leaf} - 1$ step ideally. In the current numerical experiment, the leaf size is chosen as $N_{leaf} = 4$, and thus, the communication reduction is expected to be 1/3. The communications cost of the CRMT algorithm is about 9%, and it is ×2.78 faster than the original algorithm. In the following sections, we show impacts of the CRMT algorithm on the weak and strong scalability at > 100 GPUs.

### D. Weak scalability on TSUBAME

In the present weak scalability test, each GPU computes the same domain size as shown in Table [IV] The whole computational domain is decomposed two-dimensionally in horizontal directions.

Figure 4 presents the weak scaling results. The horizontal axis indicates the number of processes, and the vertical axis indicates the MLUPS (Mega-Lattice Update Per Second), respectively. Red circles show the performance with the CRMT algorithm, blue squares indicate the performance without the CRMT algorithm. A black and gray dashed lines are an ideal scalability based on the results with the CRMT algorithm on four GPUs with intra-node communication and inter-node communication, respectively.

The total number of cells on 196 GPUs is $3.37 \times 10^9$, which corresponds to 19% compared to the finest uniform grid in the whole domain of $4,480 \times 4,480 \times 896 \, m^3$. The parallel efficiencies from 4 GPUs to 196 GPUs, based on the four GPUs performance with inter-node communication, is 93% and 80% with the CRMT algorithm and without the CRMT algorithm, respectively. We achieved maximum performance of 83,538 MLUPS by using 196 GPUs with the CRMT algorithm, and is ×1.6 faster than the original algorithm. Even in the AMR method which has a large overhead about MPI communications, we obtained good scalability thanks to the CRMT algorithm.

### E. Strong Scalability for Real-time Simulation on TSUBAME

Although we achieved high performance in the weak scalability results, it is still slower than real time. In this subsection, we show strong scaling results for a real time simulation.
Table VIII and IX present the strong scalability of CityLBM with the CRMT and without the CRMT algorithm, respectively. The whole computational domain is $1,920 \times 1,920 \times 896 \, \text{m}^3$ with 1m resolution, and that is composed of $6 \times 6$ sub-domains in Table IX. The simulation was accelerated by increasing the number of GPUs. The CRMT algorithm achieved better scalability, and the total elapsed time is approximately 2 times faster than the original algorithm on 225 GPUs.

We here estimate the possibility of a real time simulation based on the strong scalability results with the CRMT algorithm. When a wind speed and the Courant-Friedrichs-Lewy (CFL) condition are given, the real time step width can be determined as

$$\Delta t_{\text{real}} = \frac{CFL_{\text{real}} \Delta x}{U_{\text{real}}}. \quad (9)$$

If the ratio of $\Delta t_{\text{sim}}/\Delta t_{\text{real}}$ is less than 1, it is possible to achieve a real time simulation.

Figure 5 shows $\Delta t_{\text{sim}}/\Delta t_{\text{real}}$ for various wind speed, and the results are estimated by the strong scaling results in Table VIII. Wind speed is set to 2.5, 5.0, 7.5, and 10.0 m/s under the condition of $CFL = 0.25$. Real time simulation has been achieved up to 2.5 m/s on 64 GPUs, 5.0 m/s on 144 GPUs, 7.5 m/s on 225 GPUs. Since average wind speed is 4m/s in Tokyo, we can cover that condition on 144 GPUs.

VII. Summary and Conclusions

This paper presented the CRMT algorithm and the optimum implementations to realize real time simulations of airflows and environmental dynamics of radioactive substances. The CityLBM code is designed to achieve high performance on GPU-based supercomputers, and block-structured memory layout enables coalesced memory access in the AMR method. The bottleneck part of MPI communication is replaced by additional computation in the CRMT algorithm.

We showed the performance of the CityLBM code with different types of communication buffers using device, host, and unified memory. The intra-node communication through device memory achieved more than 20GB/s, which is expected for the GPU-Direct RDMA communications through the NVLink. The inter-node communication through device memory also gives higher performance, and is faster than intra-node communication through host and unified memory. From these results, we confirmed optimal choice of the buffer memory type.

We evaluate the performance of the CRMT algorithm on four processes in a single node of TSUBAME. The communications cost of the CRMT algorithm is about 9% and is approximately 2.78 times faster than the original algorithm. This is of critical importance for real time simulations, which require good strong scaling up to hundreds GPUs, where communication overheads become significant.

We discussed the weak scalability results on TSUBAME. The parallel efficiencies from 4 GPUs to 196 GPUs was 93%, and the maximum performance of 83,538 MLUPS was achieved by using 196 GPUs. The present scaling studies show the effectiveness of the CRMT algorithm which resolves performance bottleneck due to MPI communication.
Finally, we estimate the possibility of a real time simulation based on the strong scaling results on TSUBAME. Real time simulations of about 2km squares with 1m resolution are feasible up to the wind speed of 5.0m/s on 144 GPUs and 7.5m/s on 225 GPUs, respectively. We conclude that the CRMT algorithm is indispensable for the AMR-LBM to realize a real time simulation on GPU-based supercomputers.

ACKNOWLEDGMENT
This research was supported in part by the Japan Society for the Promotion of Science (KAKENHI), a Grant-in-Aid for Scientific Research (C) JP17K06570, JP17K00165, and (B) JP17H03493 from the Ministry of Education, and "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" in Japan (Project ID: jh180041-NAH, jh180061-NAH). Computations were performed on the TSUBAME at the Tokyo Institute of Technology, the Reebush at the University of Tokyo, and the ICEX at the Japan Atomic Energy Agency.

REFERENCES