# Real-time local weather forecasting using CReSS on GPU clusters

Naoya Nomura, Masato Gocho, Takuya Uesugi, Kei Akama, Tetsutaro Yamada, and Hiroshi Sakamaki

*Mitsubishi Electric Corporation, Kamakura, Japan*

## 1. Introduction

In this study, a real-time cloud-resolving storm simulator (CReSS)[1] is developed for local weather forecasting, such as local heavy rain and wind maps, which can be used in safety systems for industrial and transport infrastructure, logistics, and unmanned traffic management systems (UTMs). For these systems, an accurate sensor, such as the weather radar and water vapor LIDAR, and forecasting simulators are required to predict the rainy and windy conditions. However, the weather forecasting simulator, CReSS, requires a significant amount of computation time. Therefore, to overcome this problem, we developed the CReSS simulator on graphics processing unit (GPU) clusters. In this study, we demonstrate a prototype version of our developed CReSS for a single GPU system.

## 2. Flow of weather prediction (CReSS)

Figure 1 shows the computational flow of weather forecasting in the CReSS. In CReSS, there are three types of differential computations for each time integration loop. For example, the time differential calculation for the mixing ratio of water vapor was performed using the following formula:

$$\frac{\delta \rho q}{\delta t} = -\rho \left( u\frac{\delta q}{\delta x} + u\frac{\delta q}{\delta y} + u\frac{\delta q}{\delta z} \right) + f(q). \qquad (1)$$

Here, $q$ is the mixing ratio of water vapor, $\rho$ is the density considering the water vapor and water quality, *(x,y,z)* is the direction at each axis, and $f(q)$ is the coefficient relating to turbulence and the source terms of clouds and precipitation by $q$. When processing using a computer, it is necessary to discretize equation (1) using the central difference method. Therefore, the calculation by referring to adjacent areas (called stencil computation) is required during processing.

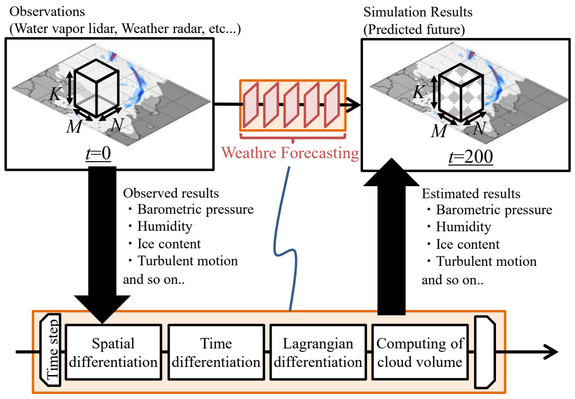

**Figure 1. Outline of weather forecasting in CReSS**

## 3. Implementation

In this study, we implemented parallel CReSS using a single GPU as a prototype version of the GPU cluster computing system. OpenACC [2] was used to implement the GPU. In this prototype system, the weather data are first received and managed in the CPU and transferred to the GPU. Second, the process of time integration (shown in Figure 1) is performed in the GPU. We set the GPU cores to 128 cores per 1 thread block and used multiple thread blocks to calculate the entire mesh. Finally, the CPU receives the processed data from the GPU and outputs the prediction results.

Table 1 lists the computational environment in the numerical experiment, and Table 2 lists the results of the numerical experiments. From Table 2, we can see that when the mesh size is set to 123 × 123 × 53, the execution time of the GPU is 8.8 seconds, and the speed-up ratio is 71 times compared to the execution time of the CPU (624.4 s).

**Table 1. Computational environment**

| | |
|---|---|
| CPU | Intel Xeon Gold 6142 (2.6[GHz])×16 Cores |
| GPU | NVIDIA Tesla V100-SXM2 (1.37[GHz])×5,120 Cores |

**Table 2. Experimental results**

| | Mesh size 43×43×53 | | Mesh size 123×123×53 | |
|---|---|---|---|---|
| | CPU | GPU | CPU | GPU |
| Spatial differentiation | 1.0 | 0.1 | 8.2 | 0.3 |
| Time differentiation | 23.6 | 1.4 | 166.4 | 3.7 |
| Lagrangian differentiation | 58.5 | 0.6 | 357.5 | 1.6 |
| Computing cloud volume | 11.7 | 2.8 | 92.3 | 3.2 |

## 4. Conclusion

In this study, we demonstrate GPU-accelerated CReSS towards real-time local weather forecasting on GPU clusters. To evaluate the efficiency of the calculation speed, we measured the execution time of the developed CReSS for a single GPU system. Future work will implement the GPU clusters using Xcalable-ACC[3] and evaluate the efficiency of the computational time.

## REFERENCES

[1] K. Tsuboki and A. Sakakibara, “Large-scale parallel computing of Cloud Resolving Storm Simulator”, High Performance Computing, Springer, pp.243–259, 2002.

[2] S. Wienke, P. Springer, and C. Terboven. “OpenACC—first experiences with real-world applications.” European Conference on Parallel Processing. Springer, Berlin, Heidelberg, pp.859-870, 2012.

[3] RIKEN AICS and University of Tsukuba, “XcalableACC Language Specification Version 1.0 (2017).”, http://xcalablemp.org/download/XACC/xacc-spec-1.0.pdf, Retrieved December 20, 2022.