Interactive In-situ Visualization of GPU-accelerated simulations using Particle-based Volume Rendering

Takuma Kawamura  
kawamura.takuma@jaea.go.jp  
Japan Atomic Energy Agency

Yasuhiro Idomura  
yasuhiro.idomura@jaea.go.jp  
Japan Atomic Energy Agency

Naoyuki Onodera  
onodera.naoyuki@jaea.go.jp  
Japan Atomic Energy Agency

1 INTRODUCTION
In-situ PBVR [2, 3] enables an interactive visualization of batch processing simulations on CPU supercomputer. In this research, we extend in-situ PBVR for an Adaptive Mesh Refinement (AMR) based CFD code on GPU platforms. In-situ PBVR consists of three main components, “Sampler” on computing nodes, “Daemon” on an interactive node, and “Viewer” on a client PC. Since the maximum particle data size is several hundreds MB, Viewer works with small memory and enables interactive frame rate on PCs. Another important function of Viewer is to provide GUI for designing a transfer function (TF) for multivariate data [1], which enables algebraic synthesis of volume data and multi-dimensional TFs. Sampler is implemented using Monte-Carlo sampling with hybrid MPI and OpenMP parallelization, where MPI parallelization employs the same domain decomposition as the simulation, OpenMP parallelization is applied to cell-by-cell parallelization in each subdomain. Daemon mediates between Sampler and Viewer via socket communication through ssh tunnel, while it interacts with Sampler via files on the storage.

2 PARTICLE GENERATION FOR AMR
The CFD code is calculated using a Block Structured AMR (BSAMR) method optimized for GPU memory layout. In the BSAMR method, the smallest grid unit called as “leaf” is defined by \( N^3 \) uniform structured grids, and leaves with different resolutions are connected in each AMR layer. Therefore, the BSAMR grid system has \( N \times N \times N \times L \) (\( L \) is the number of leaves) four-dimensional data structure.

In the particle generation process, Sampler is computed for each leaf, where volume data is extended including neighboring leaves to avoid artifact at the interface of leaves. In the proposed method, the number of generated particles is calculated by multiplying the particle density and the cell volume in each AMR layer. In the light calculation, normal vectors are used to calculate light reflection and shading. Here, the normal vector is defined by multiplying the gradient of scalar field and Jacobian of the cell.

3 IN-SITU FRAMEWORK ON GPGPU
GPU supercomputers have a heterogeneous configuration with CPUs and GPUs on each node. In this research, we compute Sampler on CPU and construct a framework that executes visualization processing and simulation in task parallel. The original CFD code transfers simulation data to CPU memory for data I/O. This interface is connected to Sampler on CPU, which generates particle data in an asynchronous manner during the simulation, and outputs the particle data. The particle data is visualized on a client PC as in the conventional in-situ PBVR.

4 RESULTS AND DISCUSSION
In order to test the effectiveness of the proposed method, we applied in-situ PBVR framework to a heat convection simulation on a GPU cluster, and compared the temperature field against ParaView in Fig. 1. From this figure, it is confirmed that the boundaries of BSAMR grids do not appear in the proposed method, and in-situ PBVR can obtain the same visualization result as ParaView. Table 1 shows the cost distribution of in-situ PBVR and ParaView. From this result, it is seen that in-situ PBVR is about 30 times faster than ParaView. This indicates that in-situ PBVR enables efficient in-situ visualization for accelerated exascale simulations.

Figure 1: The visualization result of the temperature of the thermal air flow simulation. Left: Slice of AMR grid. Middle: ParaView. Right In-situ PBVR

Table 1: The comparison of the total performance of In-situ PBVR and ParaView

<table>
<thead>
<tr>
<th></th>
<th>In-situ PBVR</th>
<th>ParaView</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data size [MB]</td>
<td>581</td>
<td>5144</td>
</tr>
<tr>
<td>Particle gen. [sec/step]</td>
<td>38</td>
<td>-</td>
</tr>
<tr>
<td>Data trans. [sec/step]</td>
<td>52</td>
<td>456</td>
</tr>
<tr>
<td>Image product. [sec/step]</td>
<td>7.7</td>
<td>900</td>
</tr>
</tbody>
</table>

REFERENCES