THE UNIVERSITY OF TOKYO

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

JCAHPC

AMR Framework for Realizing Effective High-Resolution Simulations on GPU

Application Development based on Framework-based Approach

Introduction

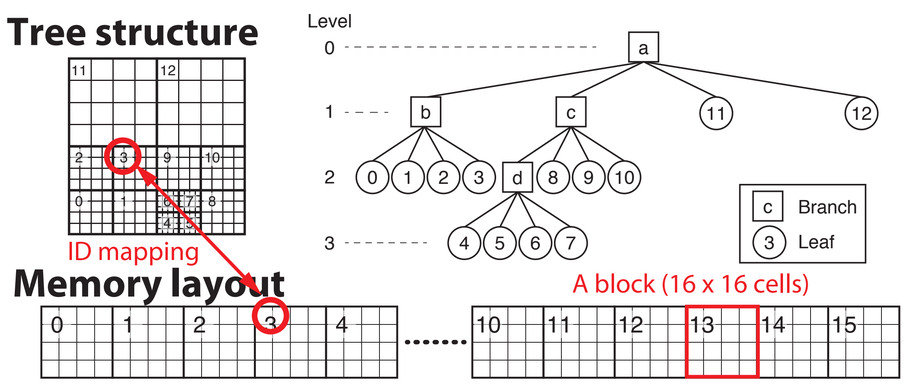
Recently grid-based physical simulations with GPU require effective methods to adapt grid resolution to certain sensitive regions of simulations. An adaptive mesh refinement (AMR) method is one of the effective methods to compute certain local regions that demand higher accuracy with higher resolution. The AMR method on the GPU is, however, complicated and requires the high development cost.

AMR Framework for GPU computing

We are currently developing an AMR framework for realizing effective high-resolution simulations on GPU. Our proposed AMR framework is based on the high-productivity framework for stencil applications [1, 2]. The proposed framework is implemented in the C++ language and automatically translates user-written functions that update a grid point and generates both GPU and CPU code.

AMR data structure

The AMR framework uses Octree for 3D and Quadtree for 2D to represent spatial distribution of decomposed grids. In order to achieve high performance on GPU, physical values on all leaves of the trees are actually stored in a single array allocated as contiguous memory area. The single array is decomposed as multiple "blocks" , each of which typically contains 16 x 16 cells in 2D with halo regions. Each leaf of the trees just holds an ID that indicates the assigned block on the array.



Writing and Executing Stencil Computation

To use this framework, programmer just provides "stencil functions" that update a grid point, which are defined as C++ functors. Since the framework is based on block-based AMR, stencil functions for Cartesian grid can be also used for AMR. Thanks to the data structure described above, the framework can easily execute given stencil functions over multiple blocks by just using a single CUDA kernel even when each block represents different resolution. This contributes to achieving higher performance on GPU.

- User-written function (C++ functor) that updates a grid point
- ArrayIndex represents the coordinate of the point where this function is applied.

```
struct Diffusion3d {
__host__ __device__
float operator() (const float *f, const ArrayIndex &idx,
float ce, float cw, float cn, float cs, float ct, float cb, float cc) {
const float fn = cc*f[idx.ix()]
  +ce*f[idx.ix(1,0,0)] +cw*f[idx.ix(-1,0,0)]
  +cn*f[idx.ix(0,1,0)] +cs*f[idx.ix(0,-1,0)]
  +ct*f[idx.ix(0,0,1)] +cb*f[idx.ix(0,0,-1)];
return fn;
}};
```

- The functor is executed over all grid points on the trees by the following expression

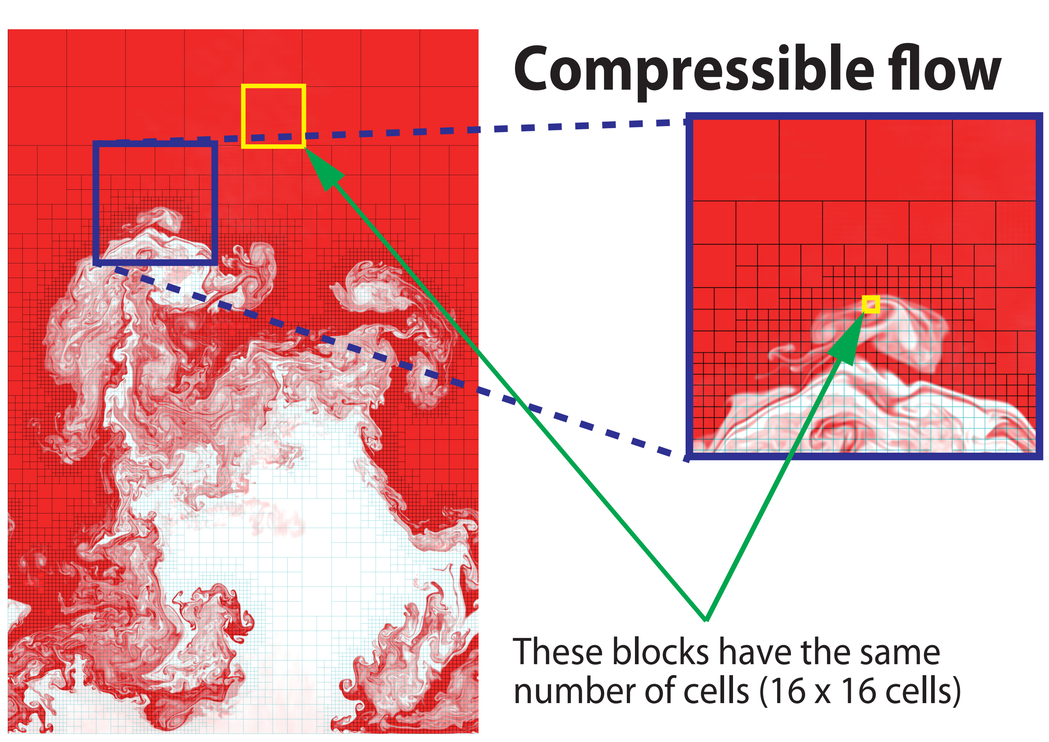
```
Range3D inside; // 3D rectangular range where stencil functions are applied.
viewma(fn, inside, level >= 0)
    = funcf<float>(Diffusion3d(), ptr(f), idx(f),ce,cw,cn,cs,ct,cb,cc);
```

Halo exchange

In halo exchange to update halo regions of blocks, each leaf accesses its neighbor leaves. Since the tree structures are not suitable for GPU computation, the framework introduces the table structure that holds block IDs to represent adjacency relationship of blocks. This table has fixed length column to improve performance on GPU. We use the same strategy for the mesh refinement processes.

Rayleigh-Taylor instability simulation using AMR

Figure shows a snapshot of the Rayleigh-Taylor instability simulation using the AMR framework on NVIDIA Tesla P100 GPU. This simulation uses 5 levels for AMR; width of the biggest cell is equal to 16 x width of the smallest cell. This simulation can shorten the computation time by almost half in the early stage of this simulation compared to those on Cartesian grid with equivalent resolution. The framework also contributes to drastically reducing the memory consumption of this simulation.



References: [1] T. Shimokawabe et al., "High-productivity framework on GPU-rich supercomputers for operational weather prediction code ASUCA," ACM/IEEE SC14. (2014)
[2] T. Shimokawabe, et al., "A Stencil Framework to Realize Large-scale Computations Beyond Device Memory Capacity on GPU Supercomputers," IEEE Cluster (2017).

Contact: Takashi Shimokawabe <shimokawabe@cc.u-tokyo.ac.jp>
http://www.cspp.cc.u-tokyo.ac.jp/shimokawabe/