Data Locality Optimization Strategies for AMR Applications on GPU-accelerated Supercomputers

Motivation
Adaptive Mesh Refinement (AMR), which is a model for adapting the resolution of a stencil mesh locally. AMR is one of the paths to multi-scale exascale applications. However, producing efficient AMR code is hard, especially for GPUs. As a result, typical AMR frameworks require the user to write his or her own optimized code for the target architecture. In addition, the generated AMR code is not optimized for reducing data transfer in GPU-accelerated supercomputers.

Our approach to achieve target
- A compiler-based framework for producing efficient AMR code (for GPUs)
- Architecture-independent interface provided to the user
- A performance model for quantifying the efficiency

Key results
Our framework generates code comparable in speedup and scalability to hand-written optimized GPU AMR implementations using up to ~1000 GPUs.

AMR Framework
The mesh is organized into a hierarchy of refinement levels. The mesh is usually decomposed into relatively small fixed-sized octants of mesh cells.

Octree-based AMR

Problem Statement

Data-centric AMR

Framework implementation
The implementation is based on LLVM compiler infrastructure.

Applications
Hydrodynamics Solver: We model a hydrodynamics application using Euler equations extending the GAMER implementation [2].
Shallow-water Solver: We model shallow water simulations by depth-averaging the Navier–Stokes equations.
Phase-field Simulation: We evaluate an AMR version of a phase-field simulation for modeling 3D dendritic growth [3].

Results

Table 1: Phase-field runtime breakdown (%) for Daino AMR

Table 2: Hydrodynamics runtime breakdown (%) for Daino AMR

Figure 1: An example of a quadtree for a 2D AMR mesh (work divided on three processors)

Figure 2: An illustration of the architecture-neutral interface used in the framework

Architecture-independent interface
The framework is composed of a compiler and runtime. The input to the framework is serial code applying stencils operations on a uniform grid. The user adds directives to identify the stencil functions and relevant data arrays.

Figure 3: Minimal example directives

Figure 5: Apply translations and optimizations as compiler passes

Figure 6: Weak scaling of uniform mesh, hand-written and automated AMR (GAMER-generated AMR included in hydrodynamic)