PLIFE
A Parallel Adaptive Mesh Framework

Timothy H. Kaiser
tkaiser@arc.unm.edu

University of New Mexico
Albuquerque Resource Center
High Performance Computing Education Resource Center
What is PLIFE?

• Framework for a parallel 2d adaptive grid program
  • Adaptive grid refinement
  • Based on 5 point stencil
  • Inspired by 2d RAGE used at LANL
  • Additional structure derived from SAIC’s NEWPLOT
• Framework handles the following:
  • Splitting / joining cells
  • Moving cells
  • Communications setup
  • Communications
PLIFE framework continued

- To create a parallel adaptive grid program you add or change:
  - Your “numerics”
  - Load balancing algorithms
  - Your algorithm to flag cells to split and join
Overview of PLIFE

Communications management
Find neighbors after we:
1) Split cells
2) Join cells
3) Move cells to other processors

Load balancing

Grid description

List of cells to move between processors

List of cell neighbor values needed by various processors

Values for neighbors

List of cells to split and join

Cell values after a calculation cycle

Routines to flag cells to split & cells to join

Numerics

Application derived from the PLIFE Framework

PLIFE Framework
PLIFE grid is a forest of quad trees

A simple grid and its data structure

- Small number of base cells
- Base cells can be recursively divided into four cell groups
- Dividing / joining on the fly == adaptive mesh refinement
Purpose: PLIFE is a research tool

• The user can:
  • Study load balancing for adaptive grids
  • Study numerical techniques
  • Use it as the basis for a hydrodynamics simulation
• Primary purpose:
  • Used in my dissertation work
  • To have a tool to study adaptive load balancing techniques
PLIFE has many highlights

- Portable
  - Fortran 90
  - MPI
- Easy to modify
  - Add your own computational data type
  - Add your own “physics”
  - Add your own load balancing scheme
  - Add routines to flag cells to split & join
- Good computation speed
- Good communication speed
Fortran 90 allows the particular fine-grained structure

- Fortran 90
- Modules
- Derived data types
- Pointers
- Trees
- Allocatable arrays
- Optional arguments
- Interfaces
- Recursive functions and subroutines
- Operator overloading
- Could be written in C++
Unique features of MPI are also used

- MPI_Alltoallv (ALL to All Variable) is used many times
  - Reduces communication calls
  - By far the most common MPI call in PLIFE
- Derived data types
  - Greatly reduces effort in modification
  - Reduces communication calls
  - Simplifies program
- Use of MPI does not restrict the usefulness on shared memory processors
Purpose dictates course-grained structure

- Primary purpose dictates:
  - Hooks for load balancing subroutine calls
  - All processors can determine location of all cells
- Supporting routines
  - Subroutine call to user defined numerics
  - Large section for communications setup
  - Small communications section
Communication is an important part of PLIFE

- Calculation grid is distributed amongst processors
- Communication is needed to update neighbors on different processors
- One or more communications / numerics iteration
- Can be a significant part of run time of the program
  - Goal of my dissertation studies: reduce communication, maintain load balance
How is communication accomplished

• Each processor holds list of cells needed by others
• Copy derived data type “value” into a contiguous array
• Do a MPI All-to-All-Variable (MPI_Alltoally)
• On receive end
  • Data goes directly to correct location in the grid
  • Copy is not required
Data structures are important in communication setup

- The grid is a forest of quad trees
- Cell has:
  - Parent pointer
  - Child pointers
  - Neighbor pointers
  - Flags indicating path and processor id for a cell
  - Value, a derived data type which holds physics variables
- Each processor has complete view of grid “tree”
- Only processor calculating values hold values
When are communications set up for a particular cell?

- Its children are removed (joined)
- Neighbors are split or joined
- It is a new cell (split)
- Cells are moved between processors
What happens first when grid changes?

- Broadcast list of moved split or joined cells
- Each processor asks the following:
  - Do I own cell?
  - Do I own any cell which may have its neighbor changed
    - Touches cell
    - Touches children of cell
    - Touches parent of cell
  - If true add to list of cells for which to find neighbor
What happens when we find a neighbor on a different processor?

• Collect list of cells needed from other processors and number of times needed
  • If cell was previously needed but not anymore $# = -1$
  • Cell can be a neighbor to multiple cells
• Send list to other processors (MPI_Alltoallv)
• Processor sending cells updates its list of cells to send
  • Eliminates cells no longer needed
  • Adds new cells
• Processor receiving cells:
  • Allocates array of data type value
  • Adjusts pointers for values of cells to point to location in the array
  • Enables a “no copy” on receipt of data
There is a simple interface to the numerics

- We use a five point stencil
- All data for a cell is stored in a derived data type “value”
  - Example:
    - cell%value%Energy
    - cell%value%X_vel
    - cell%value%Y_vel
    - cell%value%Rho
- After communication all data is in place and referenced by pointers
  - cell%topp%value%Energy
  - cell%botm%value%Energy
  - cell%rite%value%Energy
  - cell%left%value%Energy
- Users numerics must also supply list of cells to join or split
The interface to the load balancing routine is simple

- Load balancing routine must generate a list of cells to move to various processors
- Call routine to move cells
- After cells are moved communications are reset automatically
The framework was used to create a parallel adaptive grid program

- Solves Euler's equations
- Has dynamic regrid capability
- Has dynamic load balancing capability
  - Used a hybrid GA to dynamically redistribute the load
  - Also tested bisection techniques for load balancing
- Performed simulations verifying my thesis statement
- Framework is about 12,000 lines out of a 14,000 line program
Improvements can be made to the framework

- Irregular or polygonal cells
- Multiple cells to a side
- Edge based scheme
- 3d
- Could be made more scalable
PLIFE framework summary

- Can form the basis of parallel adaptive grid programs
- Can form the basis of program with dynamic load balancing
- Handles:
  - Communication
  - Communication set up
  - Moving cells between processors
- Used to study dynamic load balancing
- Framework is about 12,000 lines out of a 14,000 line program
- Easy to develop programs using it
  - Easy interface to numerics
  - Easy interface to load balancing routines