The FLASH Code

Klaus Weide
Flash Center
University of Chicago

ANL
February 2011

Flash Center for Computational Science
University of Chicago
Table of Contents

- Introduction to the FLASH Code
- Features Of FLASH 3.3
- Recently developed features
FLASH Capabilities Span a Broad Range…

The FLASH code
1. Parallel, adaptive-mesh refinement (AMR) code
2. Block structured AMR, patch-based AMR (developing)
3. Originally designed for compressible reactive flows
4. Can solve a broad range of problems
5. Portable: runs on many massively-parallel systems
6. Scales and performs well
7. Fully modular and extensible: components can be combined to create many different applications
An application code, composed of units/modules. Particular modules are set up together to run different physics problems.

- Fortran, C, Python, ...
  - More than 600,000* lines of code, roughly 75% code, 25% comments
- Very portable, scales to tens of thousand processors

Capabilities (in Release 3.3)

- Infrastructure
  - Configuration (setup)
  - Mesh Management
  - Parallel I/O
  - Monitoring
    - Performance and progress
  - Verification
    - FlashTest
      - Unit and regression testing
- Physics
  - Hydrodynamics, MHD, RHD
  - Equation of State
  - Nuclear Physics and other Source Terms
  - Gravity
  - Particles, active and passive
  - Material Properties
  - Cosmology
Code Development at Flash Center

- SVN for Version Control
- Test Suite
- Online Coding Violation Tracking and Bugzilla
  - Unfinished tasks, bugs, bad code, developer queries
- Documentation
  - Online documentation for APIs - Robodoc
  - User’s guide in HTML and PDF, “Howtos”
  - Flash-Users mailing list
- Code Releases
  - Latest: FLASH 3.3 in Oct 2010
FLASH Architecture

- **Components / Units**
  - Physics solvers
  - Infrastructure
  - Driver
  - Simulation / setup
  - Monitoring

- **Customization**
  - Simulation unit
    - Users provide at least a few customized files:
      - Initial conditions, runtime params, boundary conditions
  - inheritance
FLASH Architecture

- FLASH is not an application, but a framework for building simulations
- Setup script configures an application for make
  - controlled by Config files
- Object oriented structure through source directory layout
  - Inheritance
  - Encapsulation
FLASH Architecture: Units

- FLASH “Unit”
  - Component of the FLASH code providing a particular functionality – (elsewhere called “module”)
  - Publishes a public interface for other units’ use
  - A combination of units is selected at configure time for a particular problem setup
  - Can have multiple implementations (different algorithms, …)
  - Can have more than one subunit
  - Individual routines can be customized for a given simulation

- Inheritance through configuration tool and directory structure

- Interaction between units governed by the Driver unit

- Not all units are included in all applications
Important Units - Grid

- **Grid Unit**
  - Implements mesh structures
  - Builds the mesh, may reconfigures it
  - Implements parallelism
  - Responsible for guard cell exchange, interpolation
  - Details are hidden from physics solvers (“users”)
  - Users typically only see individual blocks of data

- **Current Implementations**
  - A simple Uniform Grid
  - Block-based AMR with PARAMESH (multiple versions)
  - Patch-based AMR with Chombo (in development)
Example of a Unit - Grid

Grid

GridParticles
- UG
- paramesh
  - MoveSieve
  - MoveTree

GridMain
- UG
  - paramesh
  - Paramesh2

GridSolvers
- Multipole
- multigrid
- Pfft

- Paramesh3
  - PM3_package

Important Units - Hydro

- **Hydro Unit**
  - Implements hydrodynamics (with or without magnetic fields)
  - Advances solution by one hydro time step
  - Several finite volume, Godunov based methods

- **Current Implementations**
  - Directionally Split Hydro
  - Directionally Unsplit Hydro
  - Relativistic Hydro
  - Directionally Split (8-wave) MHD
  - Directionally Unsplit Staggered Mesh MHD
Important Units - Eos

- Eos Unit
  - Implements Equations Of State
  - Called from Hydro, from Grid, and elsewhere

- Current Implementations
  - Ideal gas gamma
  - Multi-gamma
  - Helmholtz (degenerate ionized matter)
  - Tabulated Multi-material – in development
Important Units - Particles

- **Particles Unit**
  - Implements various types of “particles”
  - Subunits for initialization, mapping, time advance

- **Current Types**
  - Passive tracer particles
  - Massive particles (represent dark matter)
  - Charged particles (hybrid PIC code)
  - others…

- Particles infrastructure is being re-used for ray tracing to model Lasers.
Some areas of recent and current code development

- Integration of Chombo as Grid implementation
- 2T-ization of Hydro
- Table-based EOS
- Table-based opacities
- Multigroup diffusion
- Implicit diffusion solvers
- More implicit solvers…