



The ALEGRA FEM Code

Overview and Thoughts on Fault Tolerance

June 7, 2002

Richard Drake, SNL



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

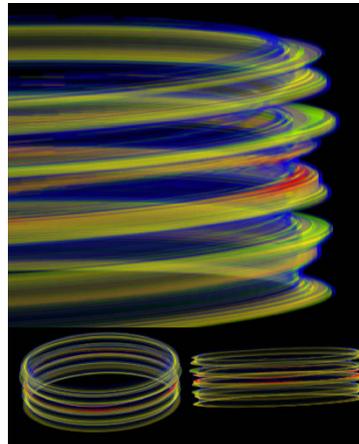
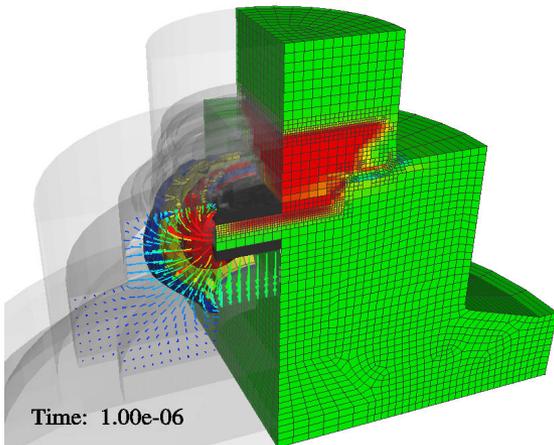


ALEGRA Usage

Relevance & Impact

Developing the physics foundation and solution methods required to simulate:

- NG Power Supply, Fireset, & Contact Fuse Performance
- Weapon Response to Blast & Impact
- Coupled Lagrangian/Eulerian earth penetrator Modeling
- Z-Pinch & ICF Phenomena for AGEX
- Architectural Surety



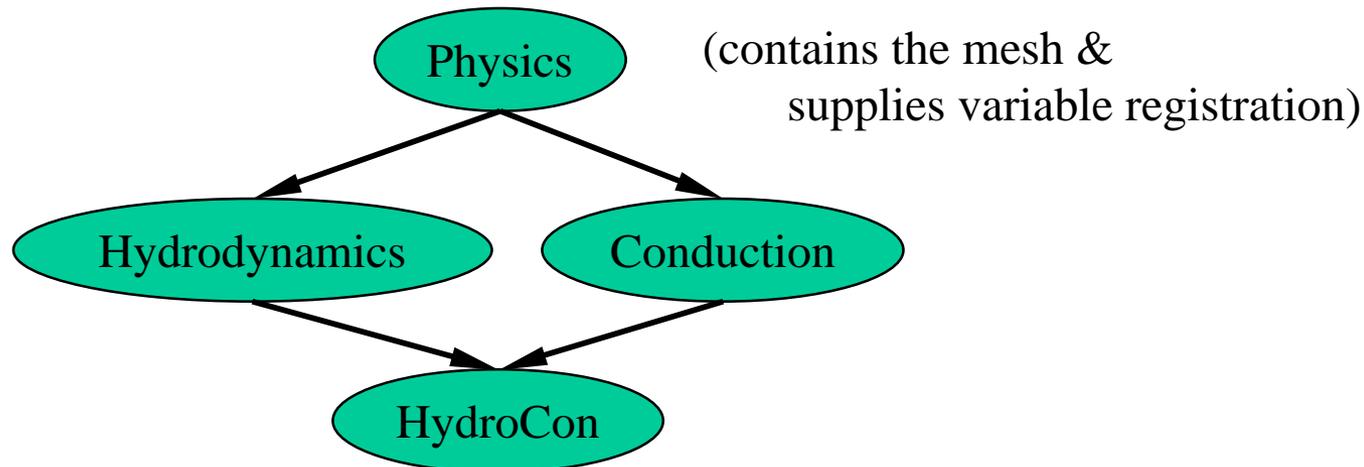
Features

- 2D and 3D
- Unstructured Finite Element Mesh
- Object-Oriented
- Massively Parallel
- Multi-material ALE (Arbitrary-Lagrangian-Eulerian)
- H-Adaptive (mesh element refinement)
- Coupled Physics
 - Solid Dynamics
 - Resistive MHD
 - Radiation Transport
 - Electromechanics
 - Full-wave Electromagnetics

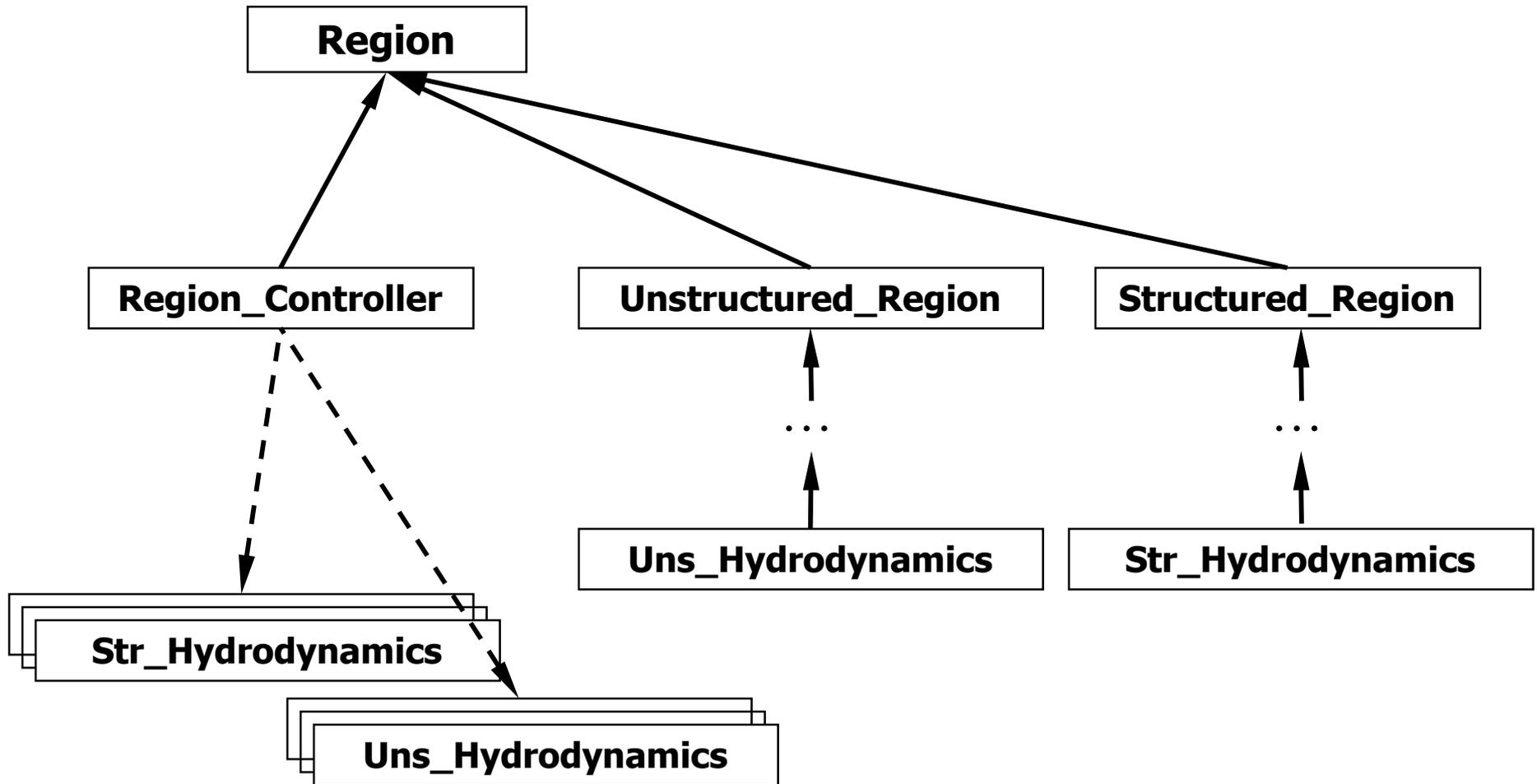
NEVADA Framework

Provides the necessary services and code structure to allow rapid physics algorithm development.

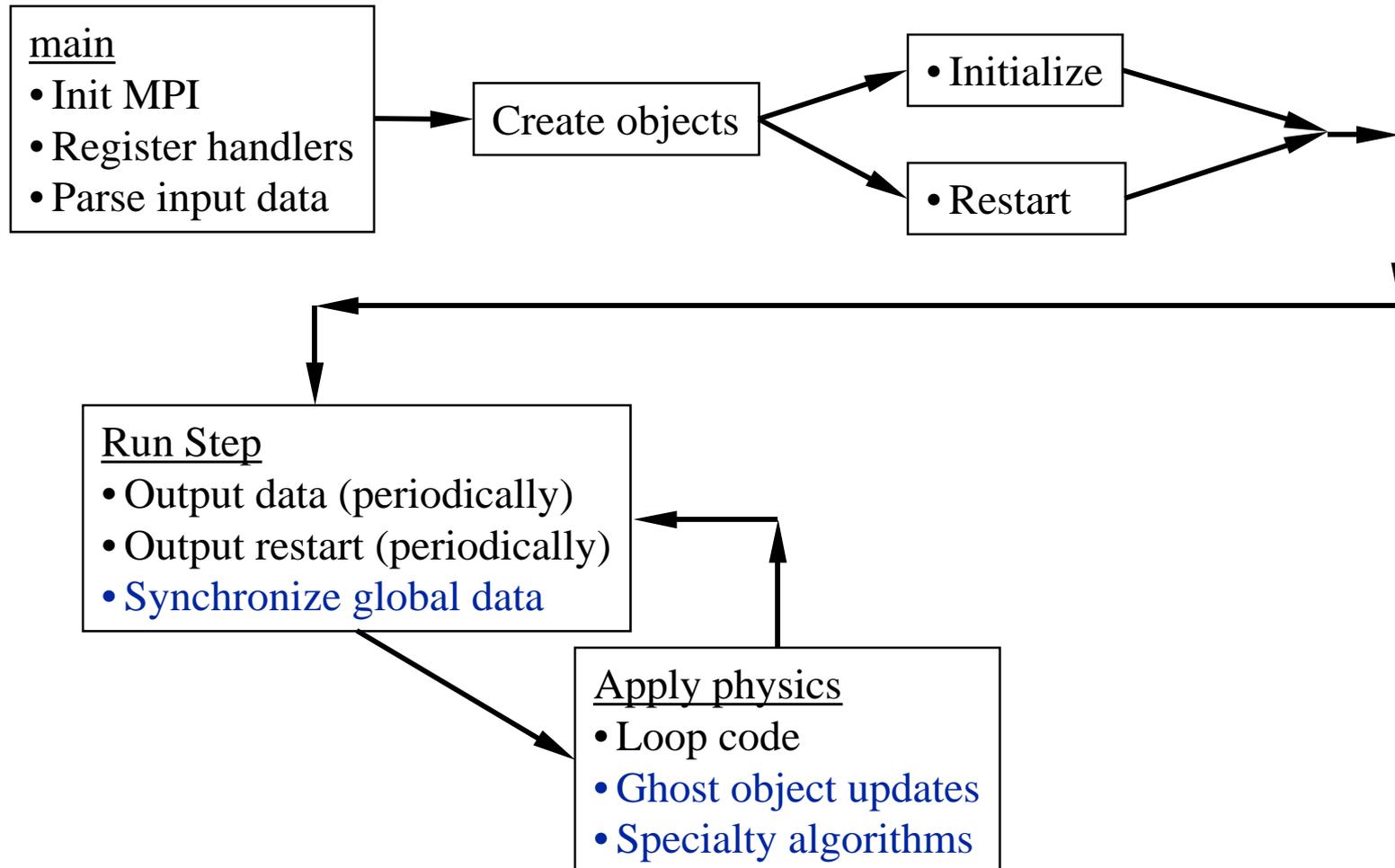
- Language is C++ but C and FORTRAN are also used.
- Services include I/O, structured & unstructured mesh topologies and variable data storage, parallel communication mechanisms, and automated regression testing.
- Physics modules use C++ inheritance to couple algorithms.
- Current work on multi-domain physics algorithms & interactions.



ALEGRA FEM Physics Hierarchy

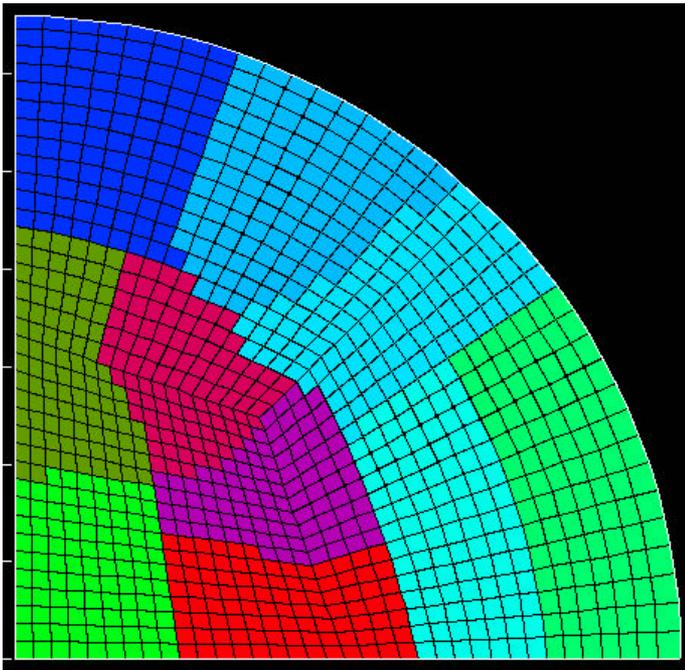


Flow of Execution



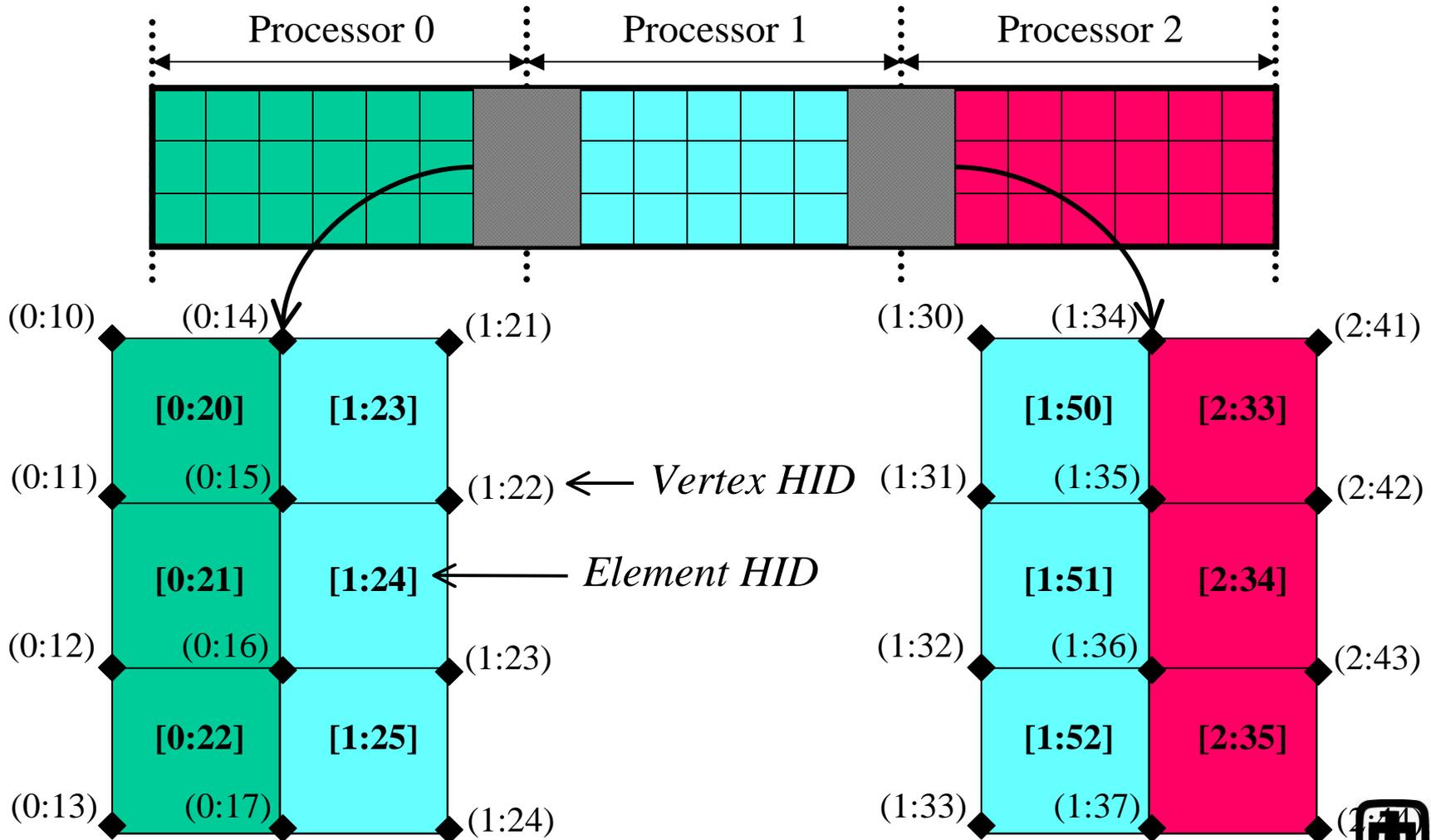
Framework Communication

- All MPI calls under control of the framework are wrapped
- At this time, MPI_COMM_WORLD is the only communicator in use
- Most communications are reductions & point-to-point with a few local processors

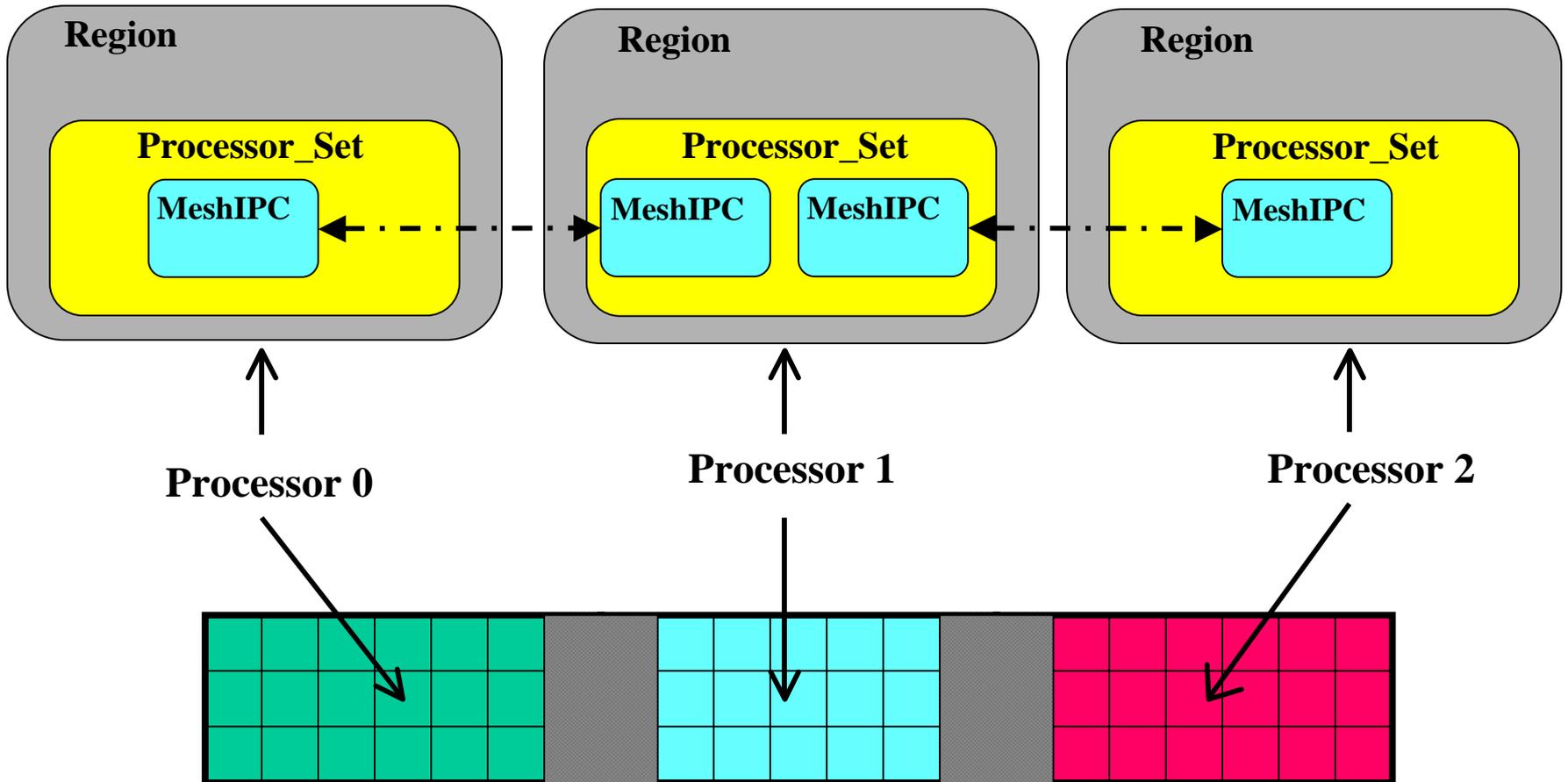


- Colors represent processor rank
- A processor must communicate using point-to-point messaging with any other processor it touches

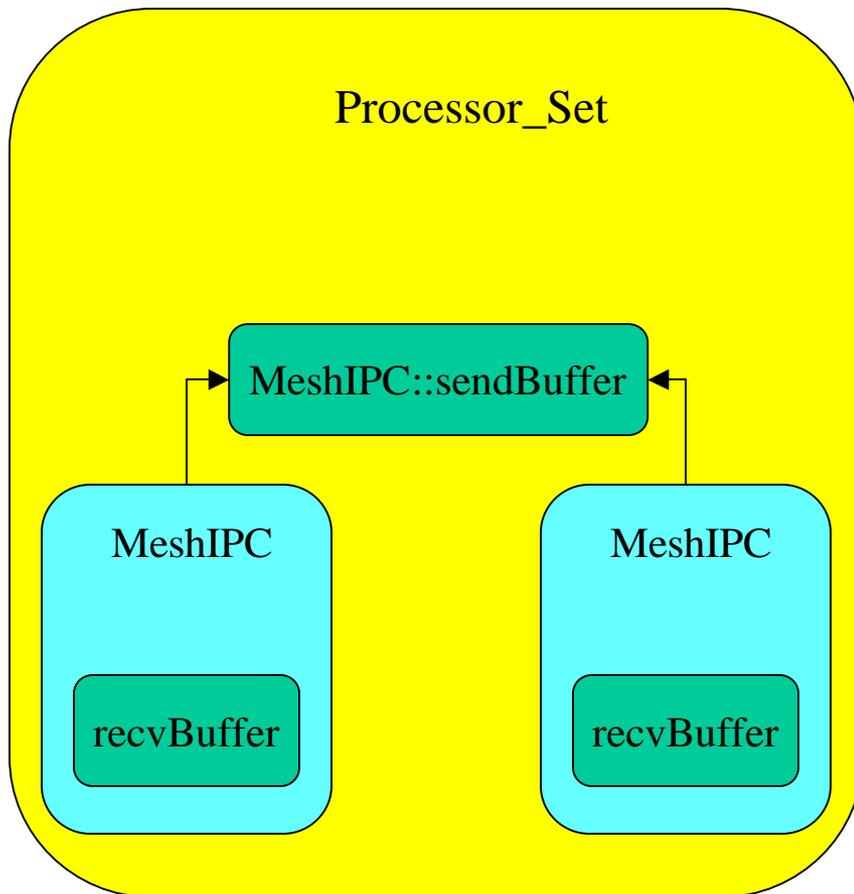
Framework Communication



Framework Communication



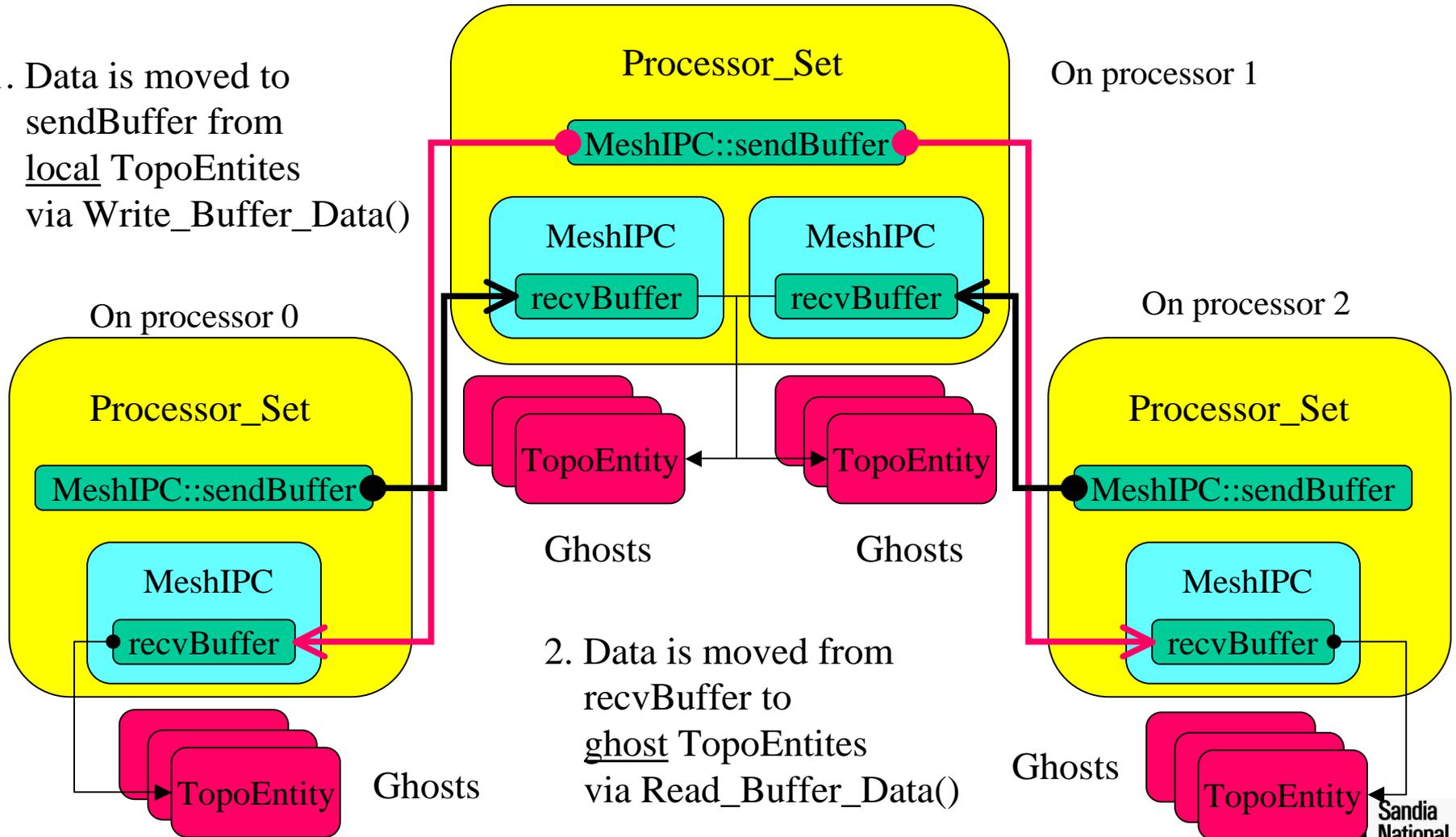
Framework Communication



1. Post non-blocking receives for all MeshIPC's (recvBuffers attached).
2. Collect data from TopoEntity objects into sendBuffer.
3. Complete blocking sends for all MeshIPC's.
4. Wait until all receives are complete.
5. Move data from recvBuffers into TopoEntity storage.

Framework Communication

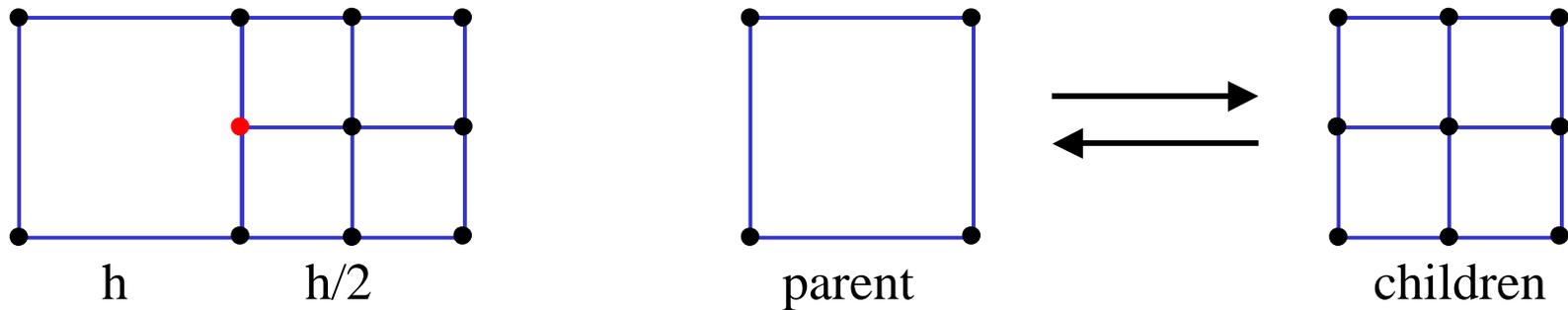
1. Data is moved to sendBuffer from local TopoEntites via Write_Buffer_Data()



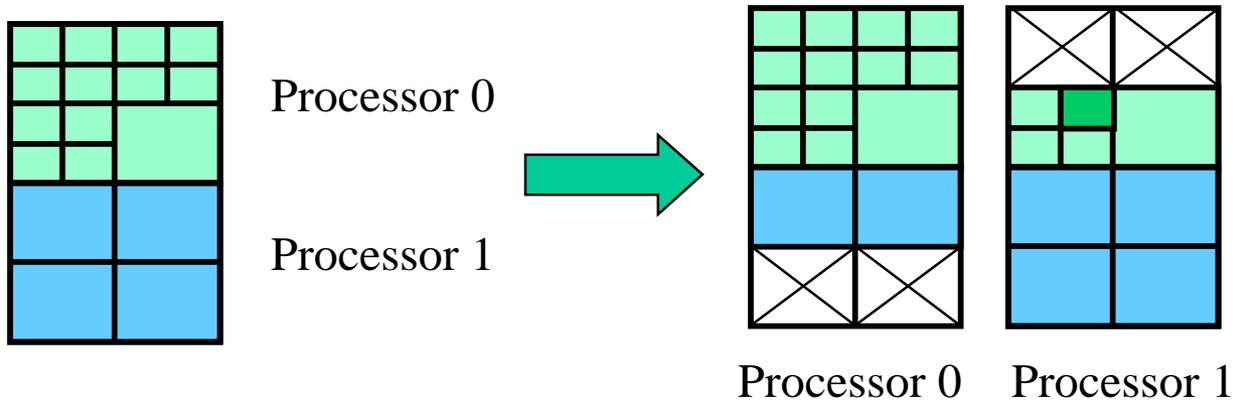
2. Data is moved from recvBuffer to ghost TopoEntites via Read_Buffer_Data()

Adaptivity & Dynamic Load Balancing

Machinery

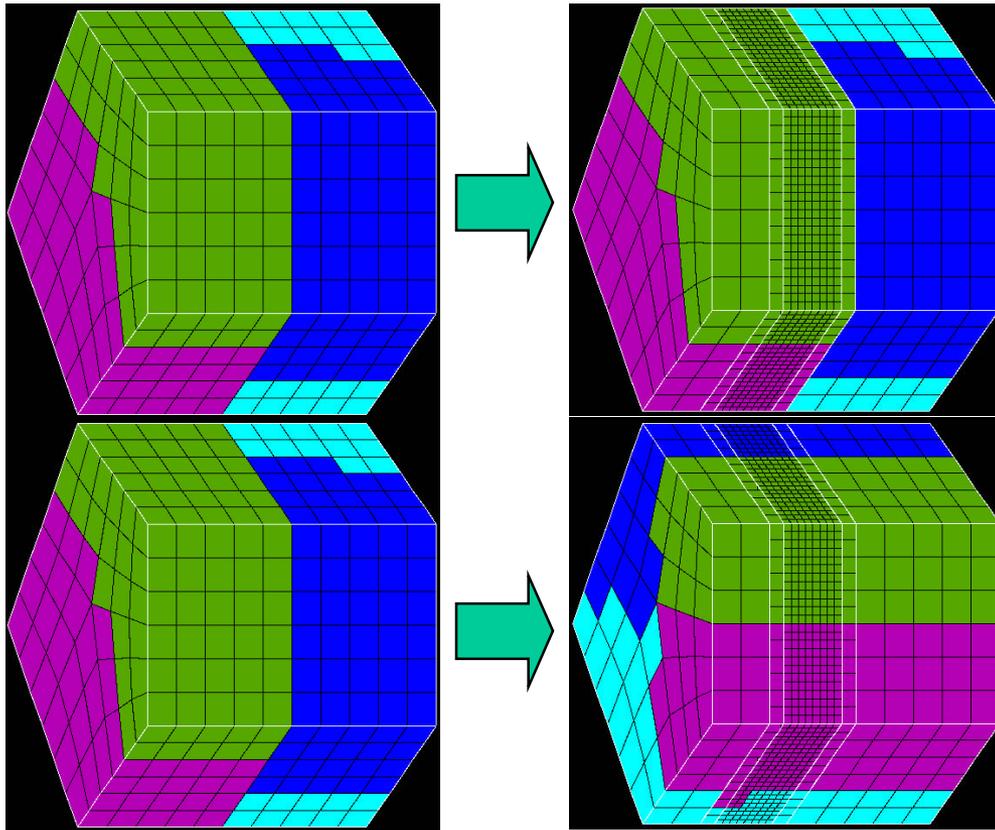


Ghost layer difficult



Adaptivity & Dynamic Load Balancing

- The ZOLTAN library is used as a black box decomposition tool
- Mesh movement & reconstruction is relatively expensive



Without load balancing

- 100 cycles
- Elapsed wall time = 1:54m

With load balancing

- 100 cycles
- Elapsed wall time = 1:40m
- Rebalance cycle interval = 30

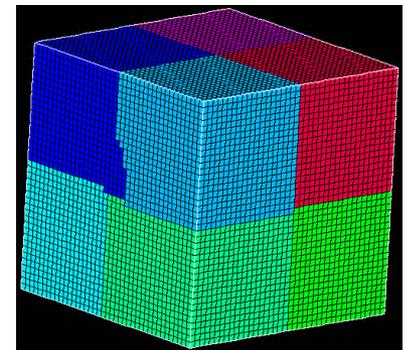
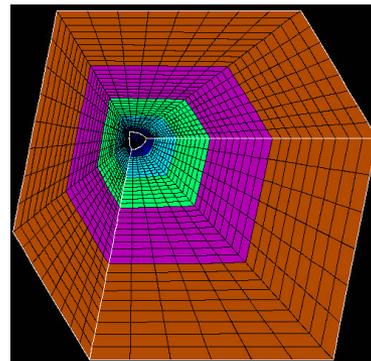
Note:

- 100 cycles
- Rebalance cycle interval = 1
- time = 4:09m

Some ALEGRA Statistics

- Usage:
 - Big: 40 million elements, 2000 processors on ASCI Red (Janus), 1 hour wall clock time, electric field calculation
 - Average big: 1-3 million elements, 100-300 processors, 12 hours to 2 weeks of wall clock time
- Restart sizes: (single node has 256 MB RAM ~ 200 MB available)
 - 10K elmts of MHD = 7 MB
 - 10K elmts of Hydro w/ contact = 3 MB
 - 150 elmts of solid dynamics = 0.2 MB
- Communication:
 - Lagrangian, 3K elmts, 6 procs:
 - = 20 global & 10 P2P per step
 - 860KB max P2P size
 - Eulerian, 8K elmts, 8 procs:
 - = 54 global & 175 P2P per step
 - 613KB max P2P size

$$\frac{\text{restart size}}{\text{core size}} \bullet 5\%$$





Fault Tolerance in ALEGRA/NEVADA

Required features:

- Check-pointing
- Synchronized recovery

Required enhancements:

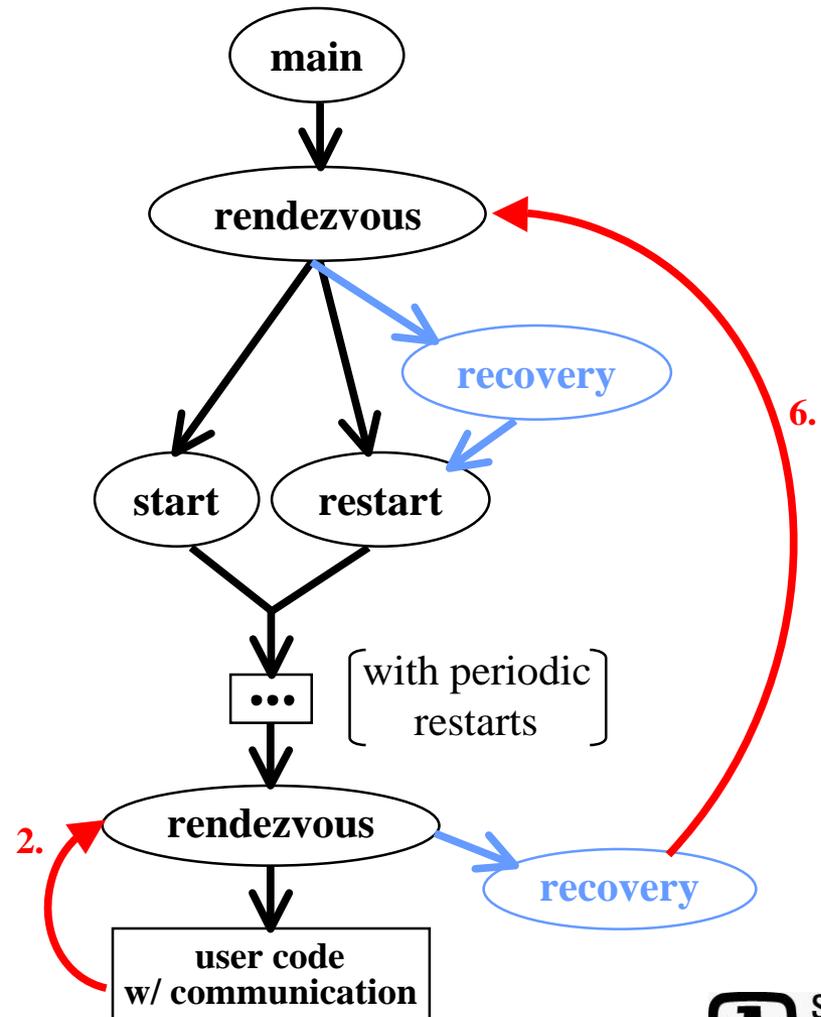
- Improve destruction/construction of high level objects
- Remove use of MPI_COMM_WORLD
- Rewrite communications to exclusively use communicators

Desired enhancements:

- Modify restarts to allow reading & writing to RAM
- Move to 3rd party libraries that use communicators (urge existing 3rd party libraries to use communicators)

Recovery Strategy for Lost Node

1. A compute node crashes or becomes disabled
2. **User code detects an error and jumps to a rendezvous point**
3. Recovery sequence begun
4. Health of process group determined
5. Lost node & associated info determined
6. **Signal sent to all processes to synchronize at a restart location**
7. Replacement node spawned and synchronized
8. Restart files read and application continues





Detection & Synchronization

Detection at the application level:

- Signals (including timeouts)
- Active probing
- MPI return values & MPI error handlers

Synchronization at the application level:

- Signals & setjmp/longjmp's
- “Out-of-band” message passing
- Special MPI communicator

Restart strategies:

- File restarts written & read
- RAM restarts written & read
 - Buddy system
 - Each process stores a fixed number of restarts from other processes
 - Dedicated restart storage processes



Reliability in LAM/MPI

- Requirements Specification draft Oct, 2001
- Jeffrey Squires, Brian Barrett, & Andrew Lumsdaine
- Failure scenarios:
 - Complete failure of a node
 - Node becomes disconnected from network
 - User application failure
- Programming Models:
 - Asynchronous failure notices: registered callback functions triggered when the MPI layer is notified of a remote process
 - MPI function failures: collective calls on wounded communicators & point-to-point communications with failed remote processes
 - Extension of MPI function semantics: MPI_COMM_FREE & MPI_COMM_SPLIT must allow operation on wounded communicators
- Application requirements: invoke “reliable” MPI mode, register error handler, use MPI_COMM_SPLIT to create healthy communicators, check result codes of all MPI operations



Comments

- Lot of work to get ALGRA/NEVADA to be fault tolerant
- The MPI layer is what is visible to the application but is only available in limited fault tolerant form
- The LAM/MPI programming model seems promising for application developers
- Additional complications introduced with batch & queue platforms when MPI_COMM_SPAWN is needed