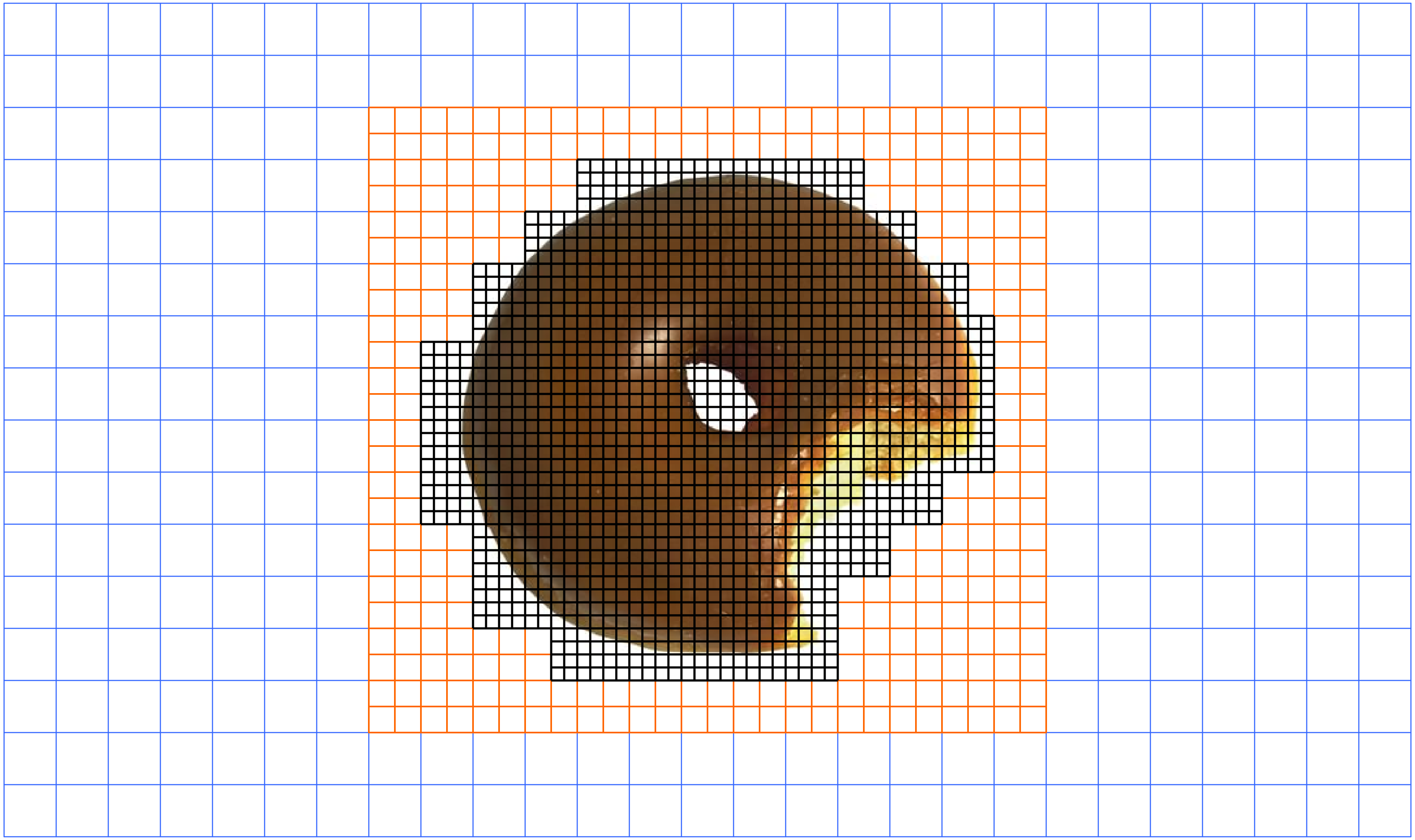
Parallel Performance for Reacting Flow Using Adaptive Mesh Refinement

Andy Nonaka
Center for Computational Sciences and Engineering
Lawrence Berkeley National Laboratory
July 27, 2009

- Personnel
  - John Bell, Director
  - Ann Almgren
  - Andy Aspden
  - Vince Beckner
  - Marc Day
  - Xinfeng Gao
  - Candace Gilet
  - Mike Lijewski
  - George Pau

Our Research

- Our primary focus is in the application of structured-grid finite difference methods on adaptive grid hierarchies for reacting flow.



Scientific Applications

- Incompressible Navier-Stokes
- Compressible Astrophysics
- Low Mach Number Flows
  - Combustion
  - Astrophysics
  - Porous Media

- Machines
  - Franklin @ Lawrence Berkeley National Lab
  - Jaguar @ Oak Ridge National Lab
  - Intrepid @ Argonne National Lab

Example: 3D Incompressible Navier-Stokes

- Entrainment behavior of turbulent jets subject to off-source heating. The movies show a tracer injected with the jets. The jet on the left is unheated. Three different heating rates were applied (increasing to the right).
- 2 levels of refinement; effective 512 x 512 x 768 resolution
- 1000 processors on Franklin, ~10 sec/time step

Movie removed due to large file size.
Visit our website at
ccse.lbl.gov

Example: 3D Low Mach Number Combustion

- Low swirl burner simulation of an H_2 flame. The color represents log(OH^- Concentration). The clouds are contours of constant vorticity.
- 3 levels of refinement; effective 2048^3 resolution.
- 4,000 processors on Franklin, ~2,000 sec/time step

Movie removed due to large file size.
Visit our website at
ccse.lbl.gov

Example: 3D Low Mach Number Astrophysics

- Convection patterns of a carbon-oxygen white dwarf leading up to a type Ia supernovae. Red is outward radial velocity, blue is inward.
- Single level run with 384^3 resolution
- 1,728 processors on Jaguar; ~10 sec/time step

Movie removed due to large file size.
Visit our website at
ccse.lbl.gov

Software Details

- Written in C++, F77, and F90
- Based on the BoxLib framework
- Block structured AMR with subcycling in time
- All parallel communication uses MPI

Algorithm Details

- We solve a system of advection, reaction, and diffusion equations.
  - Advection discretized with Godunov method.
  - Diffusion treated semi-implicitly, uses multigrid.
  - Reactions use an ODE solver within each cell.

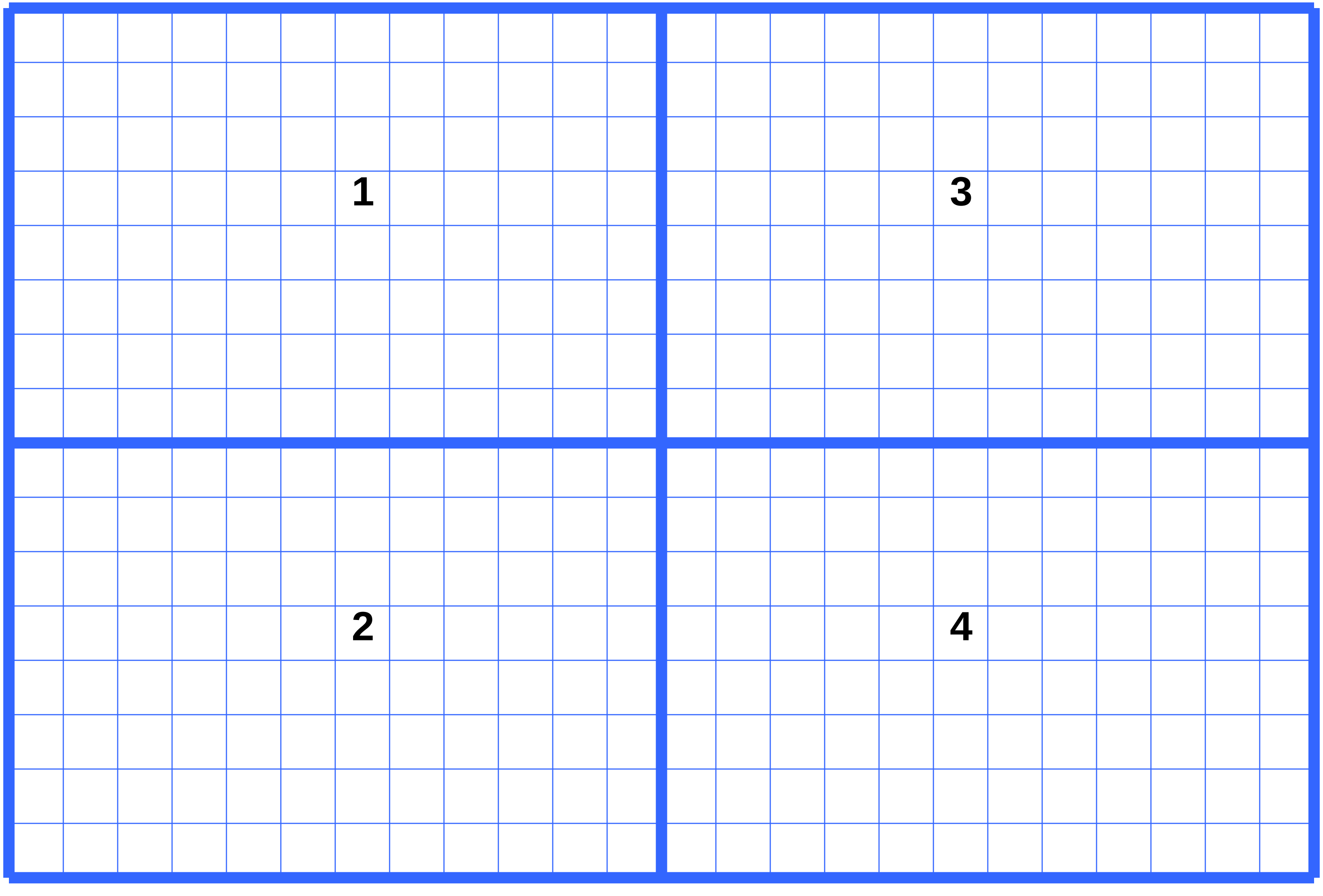
- Elliptic equations are required to constrain the velocity for incompressible and low Mach number equations, or to solve for gravity in our astrophysics codes. Elliptic solves use multigrid.

Implementation Details

- Here is the base grid. Let's say that we have 4 processors available.

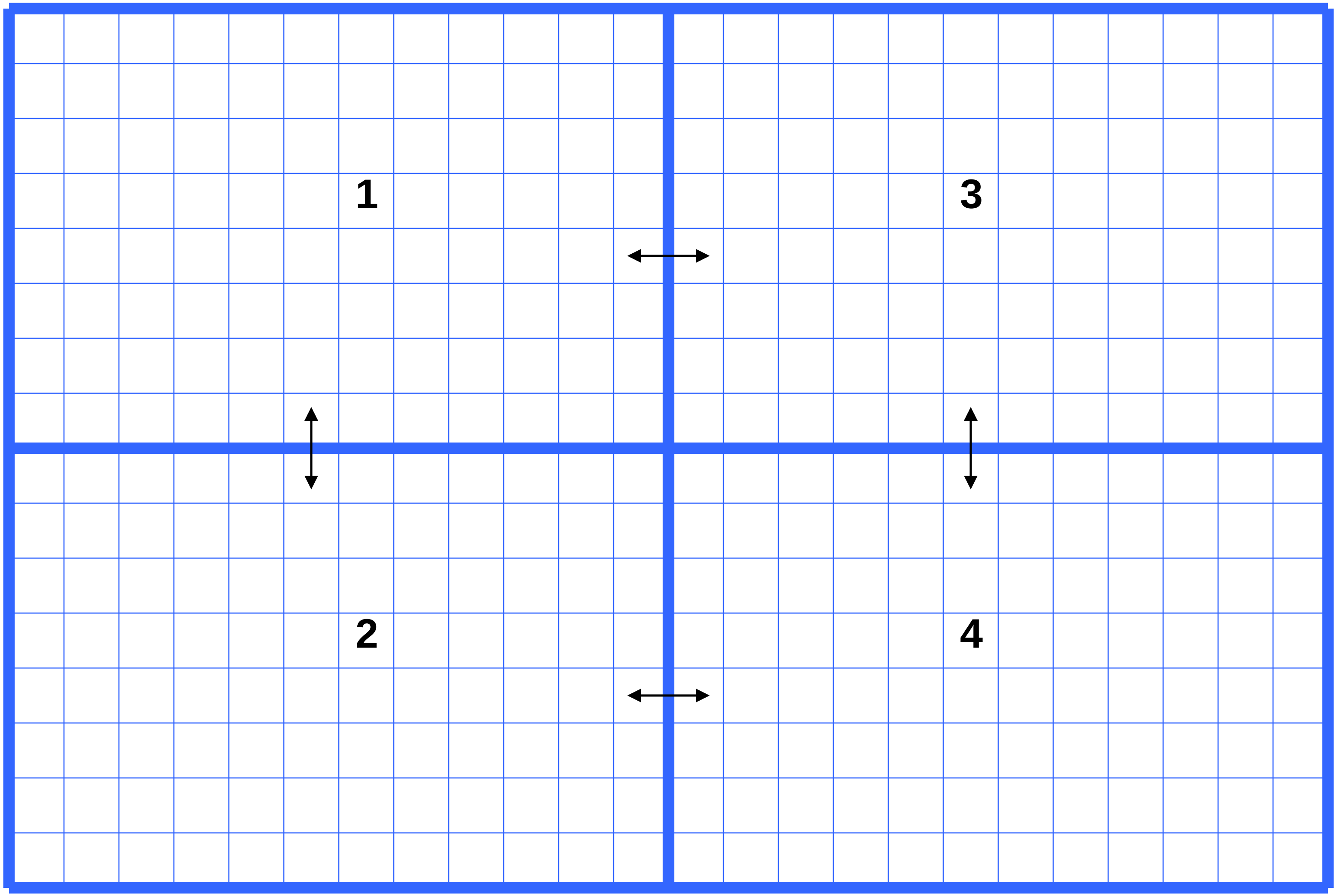
Implementation Details

- We divide up the domain into grids, assigning each grid to a particular processor.



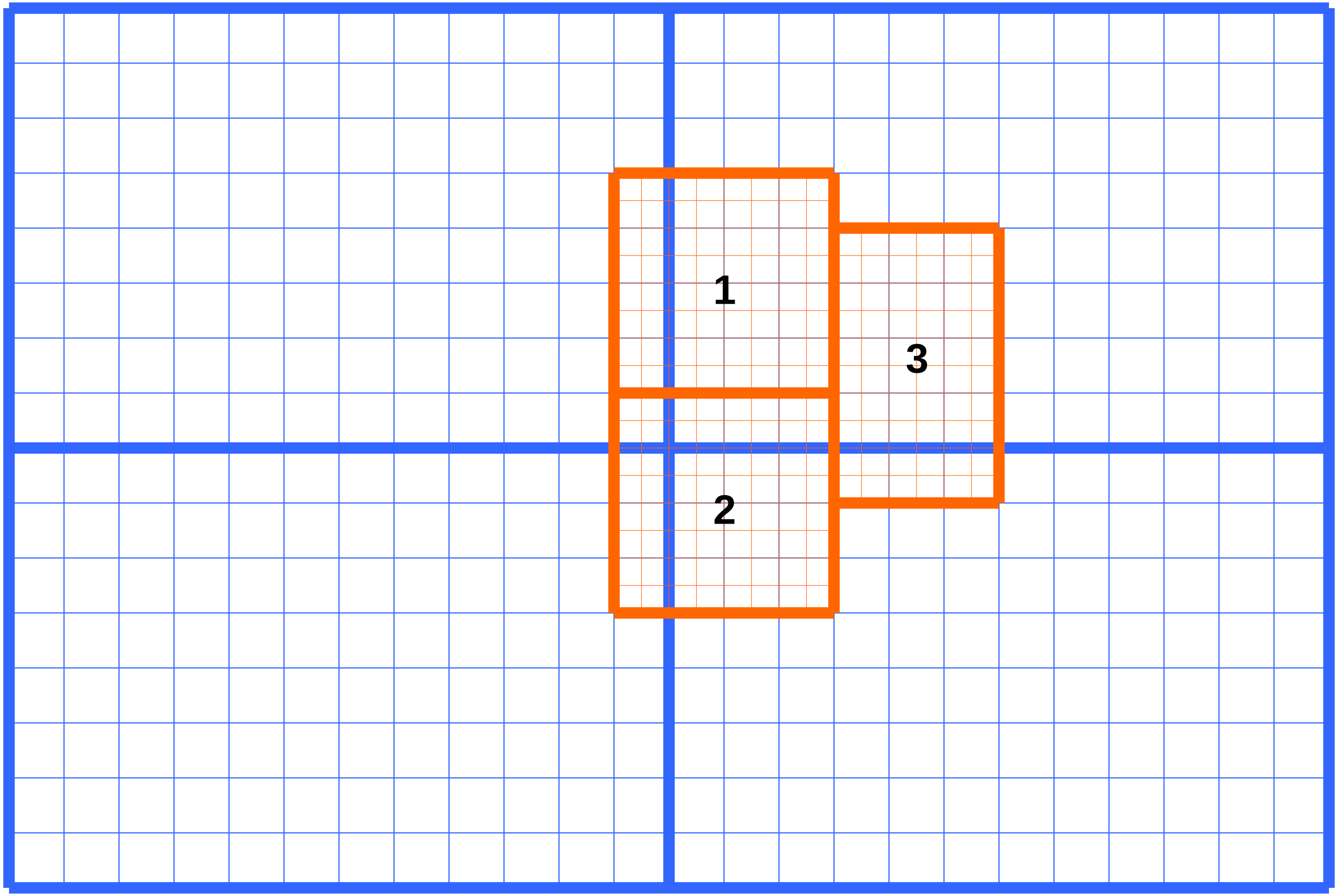
Implementation Details

- In order to advance the solution at this level, we need to communicate between grids.



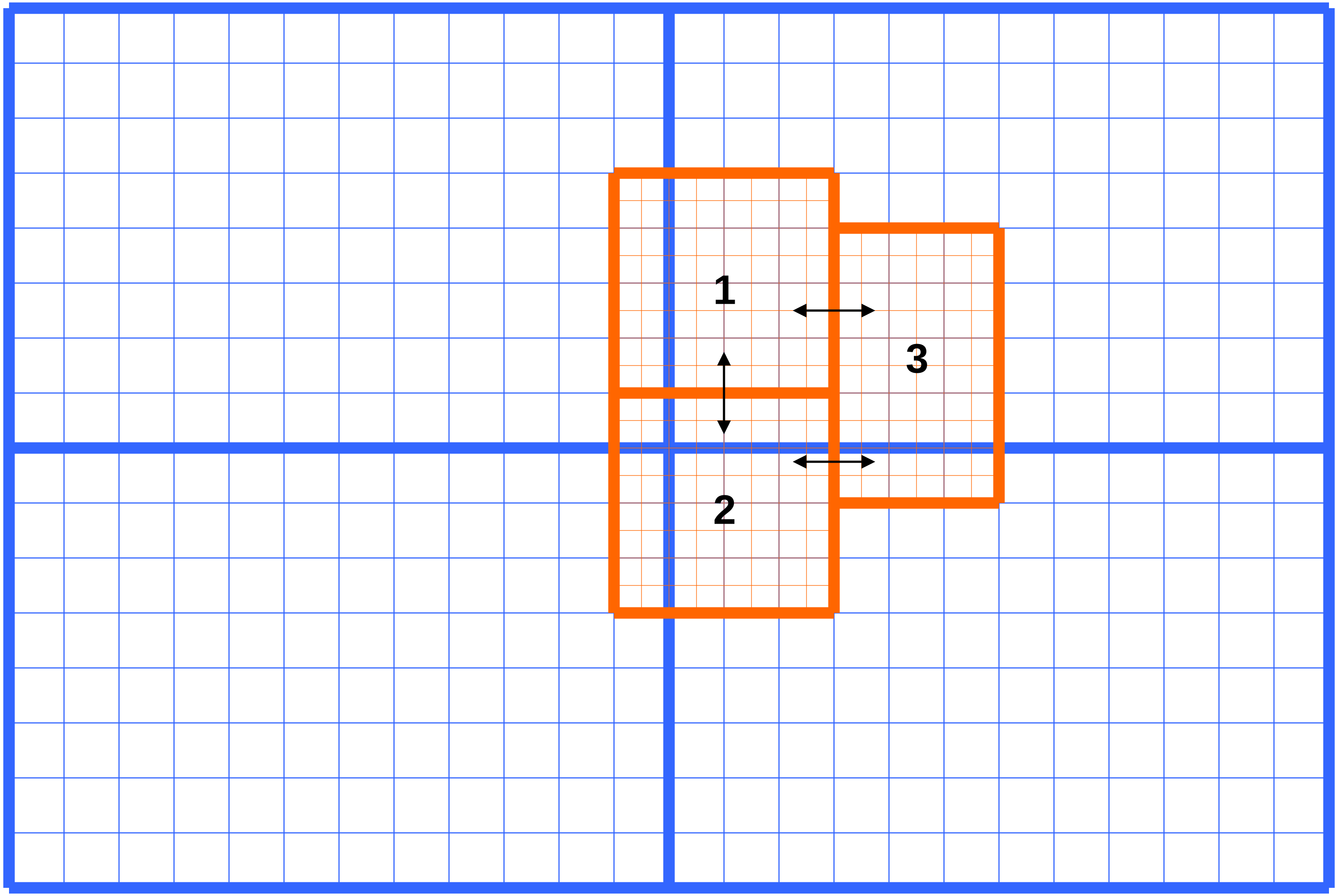
Implementation Details

- We repeat this process for each level of refinement. Note potential load balance problems.



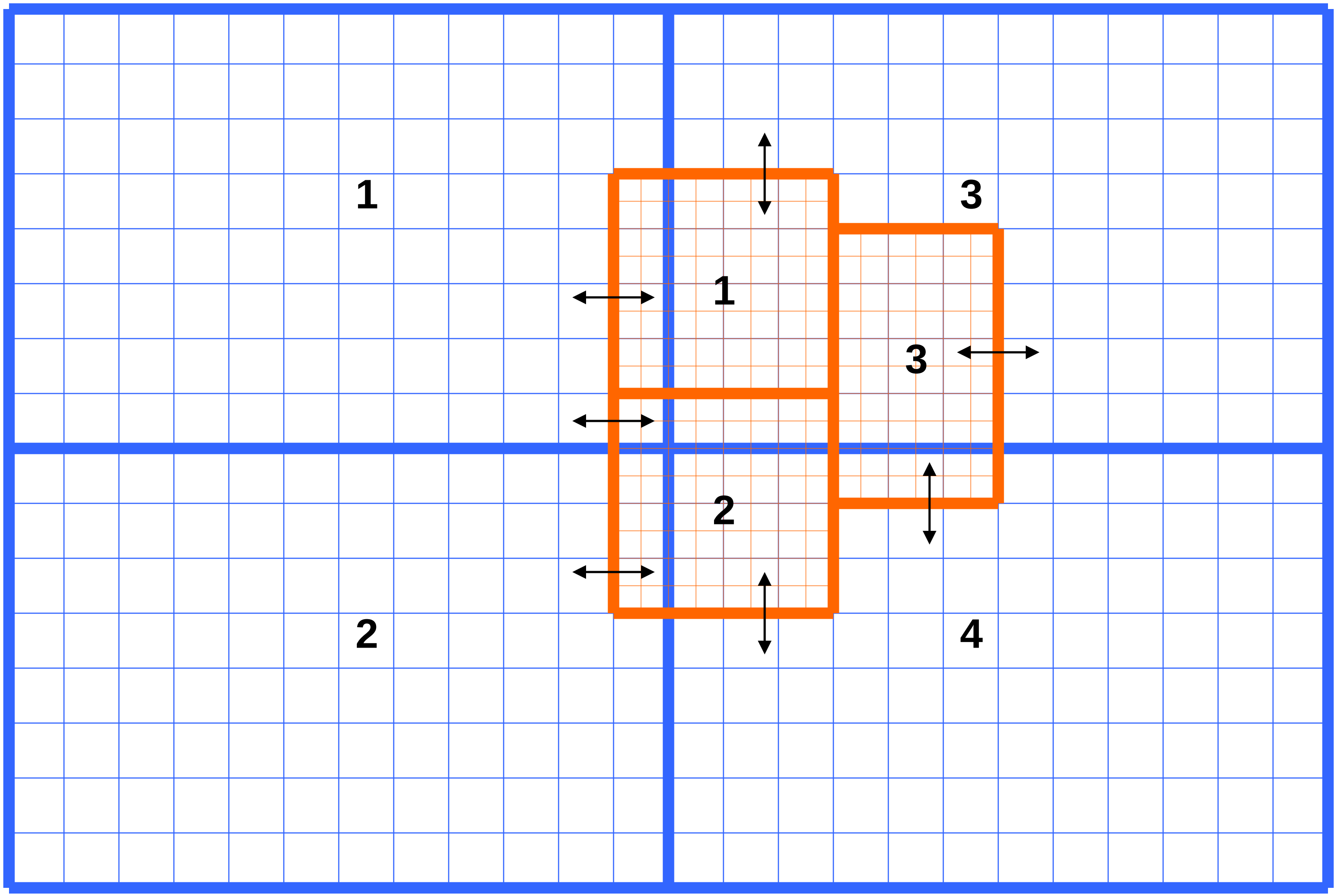
Implementation Details

- In order to advance the solution at this level, we need to communicate between grids.



Implementation Details

- We also require two-way communication between levels.



Scaling Numbers: Low Mach Combustion

- Weak scaling on Franklin
- 2 levels of refinement
- Effective 256 x 256 x 512 resolution for the 64 processor case

# processors	time / time step [s] trial #1	time / time step [s] trial #2
64	3,783	3,627
256	3,418	3,535
1,024	3,417	3,773
4,096	3,990	4,802

note variability in time step for larger runs

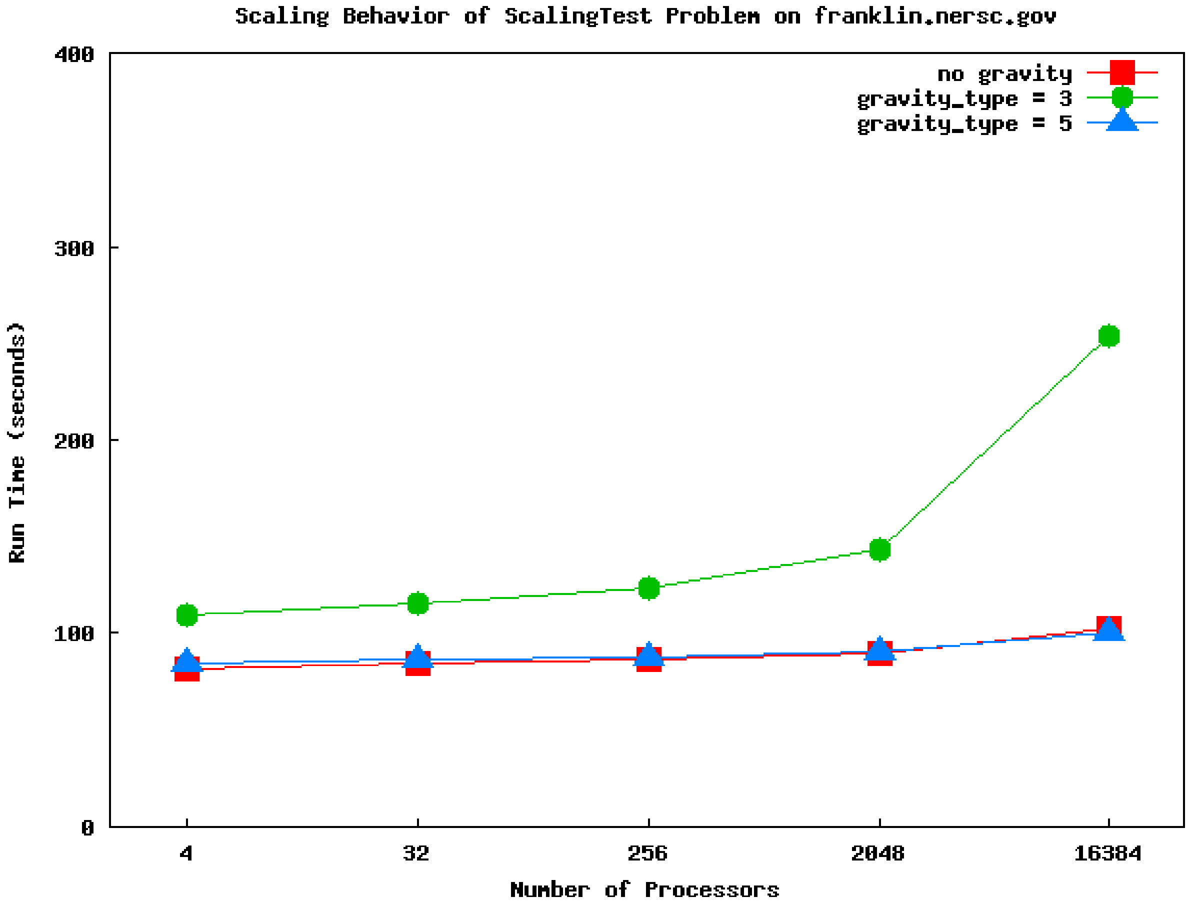
Scaling Numbers: Low Mach Astrophysics

- Strong scaling on Jaguar
- No AMR
- 1024^3 resolution

# processors	time / time step [s] 32,768 – 32^3 grids	time / time step [s] 4,096 – 64^3 grids
2,048	81	71
4,096	48	36
8,192	38	
16,384	29	
32,768	22	

Scaling Numbers: Compressible Astrophysics

- Weak scaling on Franklin
- 1 level of refinement
- Effective 128^3 resolution for 4 processor case



Parallel Performance Issues

- Scaling
  - We scale well up to ~4,000 processors.
  - Timers in our code show that multigrid is the worst offender, but communication in all parts of code is non-trivial as well.
- I/O
  - Data files currently 100GB each. Can easily reach 1TB when we double the effective resolution in 3D.
  - Visualization using VisIt, handles large datasets well.
- Single-node performance
  - What tools are available to analyze this? What can be done to increase performance?
- What about hierarchical programming models (threading?)
  - Automatic tools on Intrepid show minimal speedup. Straight MPI easily outperforms threading.