
Use of Performance Tools for AMR Calculations with the SAMRAI Framework

Andrew Wissink

with

Richard Hornung, Steve Smith, Noah Elliott,
Brian Gunney, David Hysom

***Center for Applied Scientific Computing
Lawrence Livermore National Laboratory***

October 16, 2002

LACSI Performance Tools Workshop



Outline

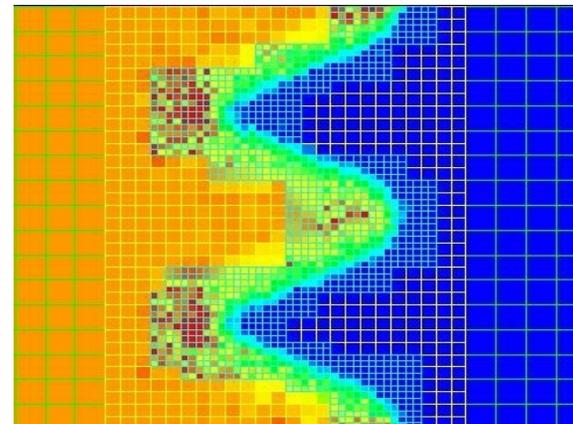
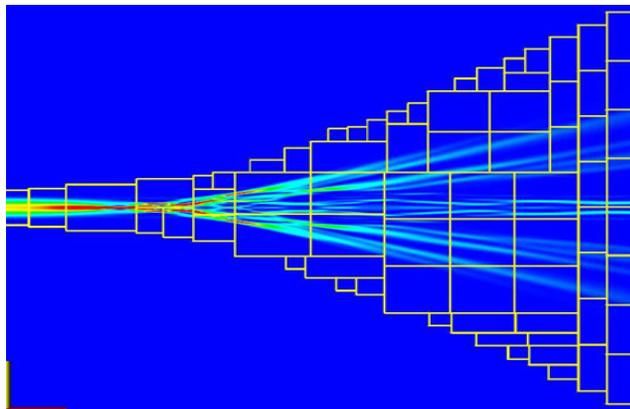
- **SAMRAI introduction**
- **Parallel implementation of SAMR**
- **Parallel performance measurements**
- **Use of performance tools in SAMRAI**
- **Performance information “wish list”**

SAMRAI

Structured Adaptive Mesh Refinement Application Infrastructure

- Object-oriented (C++) software framework for parallel (MPI) adaptive multi-physics applications
- Supports applications investigating multi-scale phenomena.
- High-level reusable code and algorithms shared across a variety of applications.

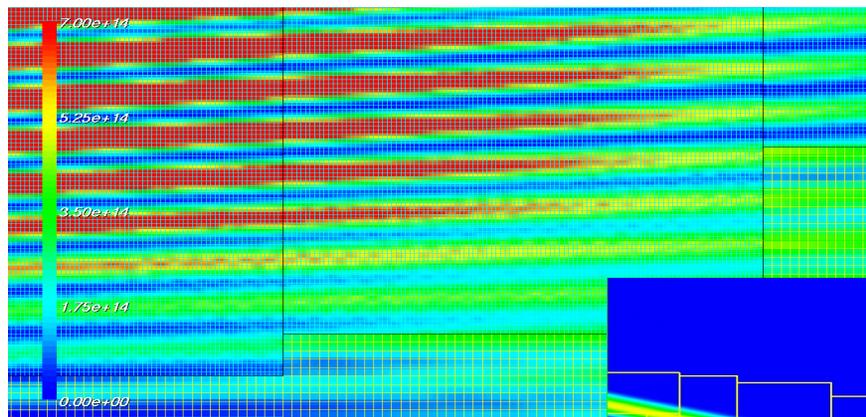
www.llnl.gov/CASC/SAMRAI



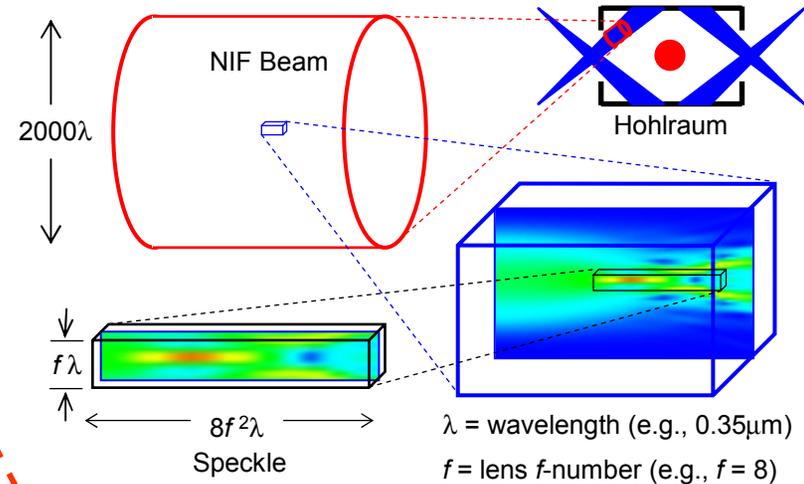
ALPS uses SAMRAI for adaptive laser plasma instability simulation

Understanding instabilities in laser-plasma interactions is critical in the design of plasma physics experiments

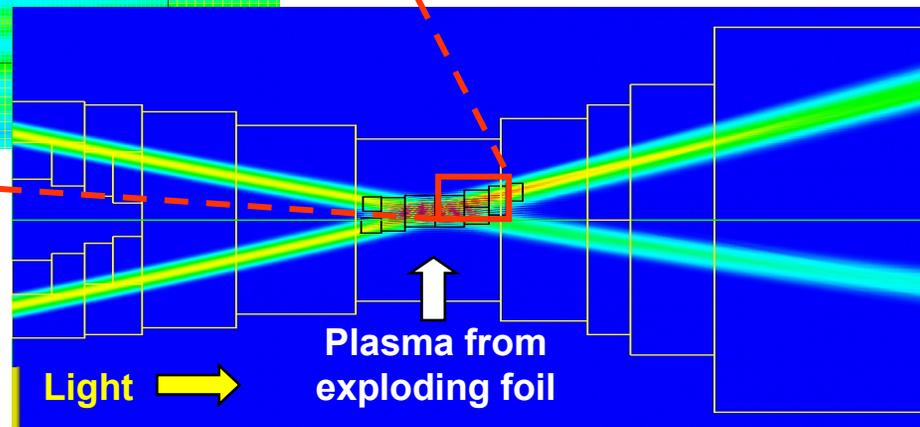
Dorr, Garaizar, Hittinger (CASC-LLNL)



Locally refined grids resolve wave interaction where high accuracy is needed



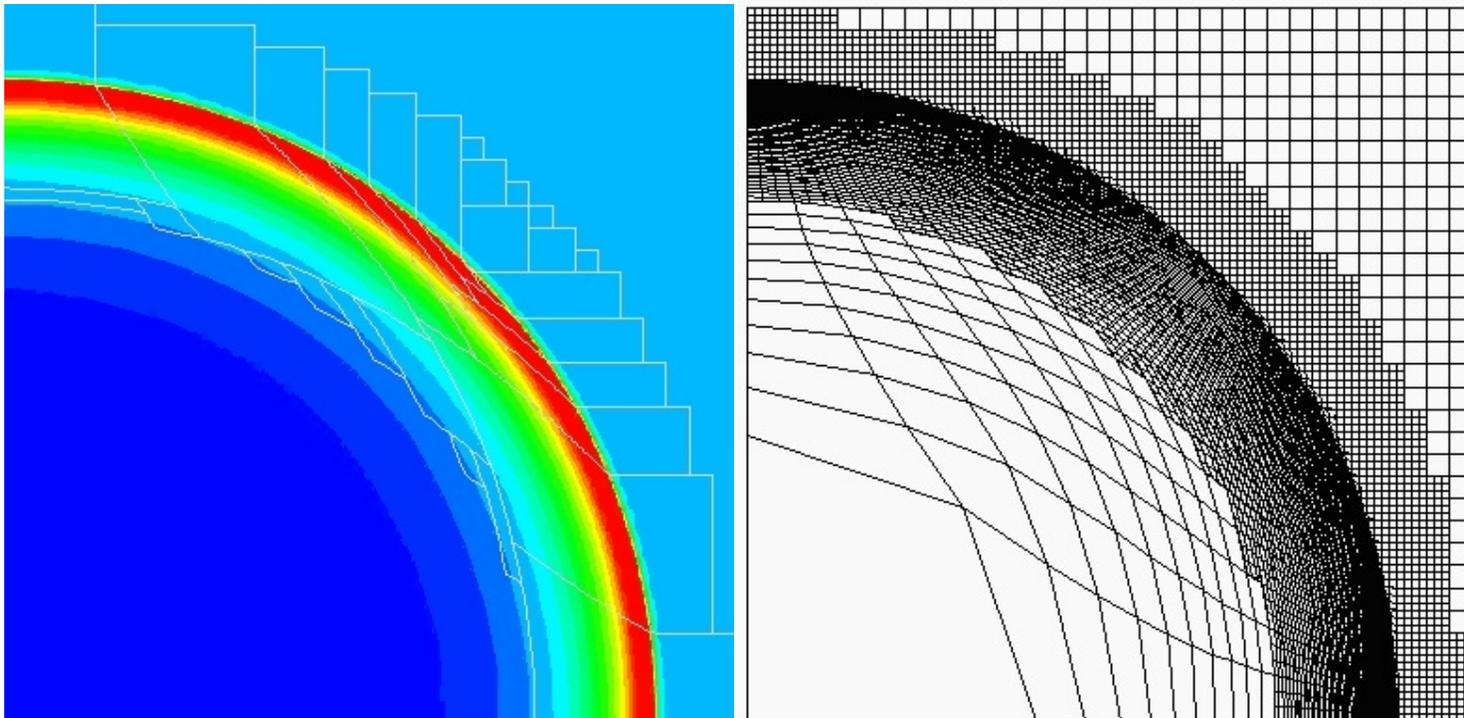
Numerical simulations need to accommodate multiple scales



ALE-AMR couples ALE models with AMR to model shock hydrodynamics

Improve accuracy of ALE simulations by increasing concentration of mesh points around regions of interest

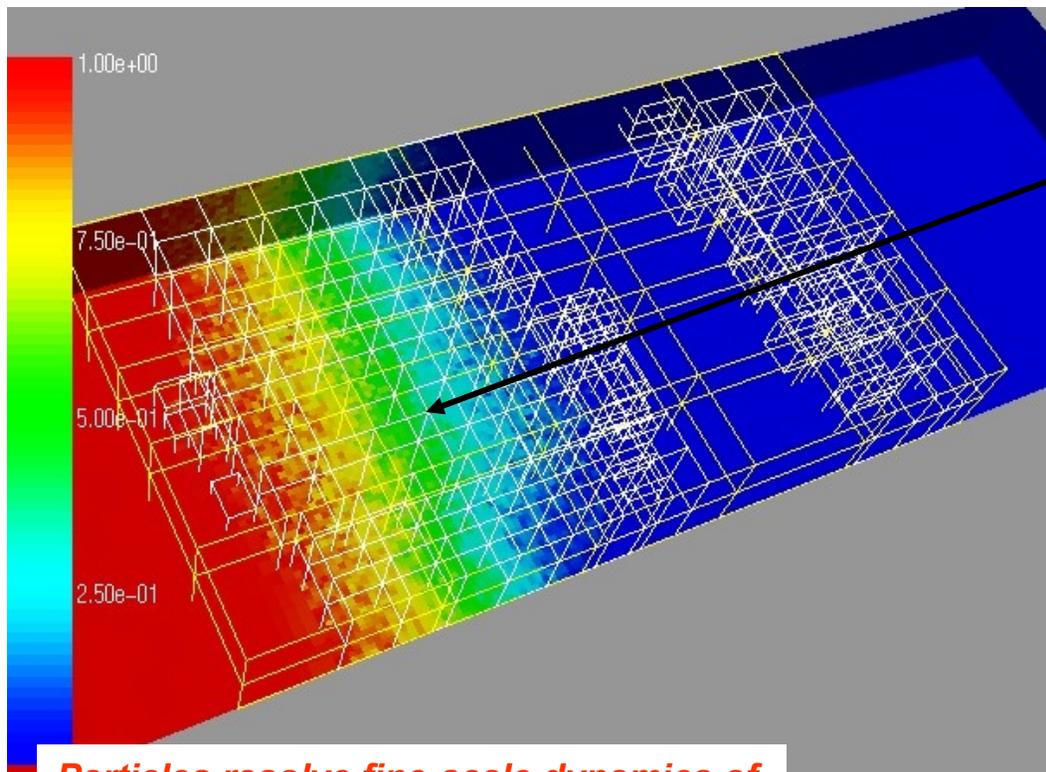
Anderson, Pember, Elliott (CASC-LLNL)



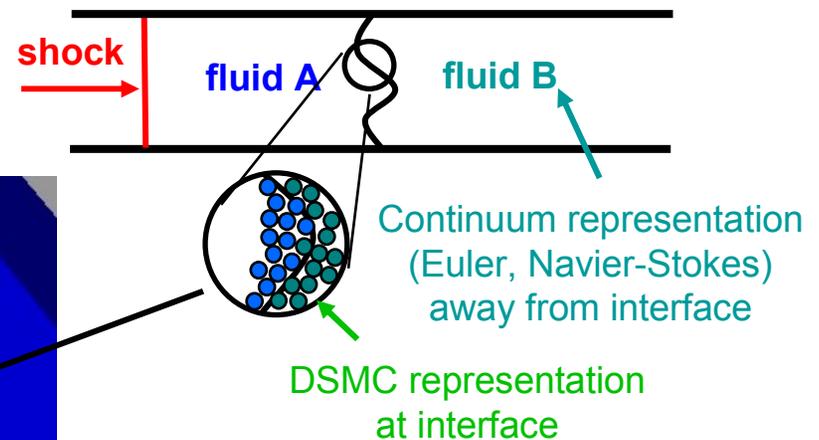
Sedov blast wave density and Lagrangian mesh

Hybrid continuum-DSMC model used to efficiently resolve interface dynamics

Interface instability problems (e.g., Richtmyer-Meshkov) involve coarse-scale hydrodynamic transport and fine-scale molecular diffusion



Particles resolve fine-scale dynamics of mixing region in an adaptive calculation



- Interface region grows and moves as instability evolves
- Standard CFD simulation of turbulent mixing is limited by finest mesh scale
- Particle resolve molecular behavior but are too expensive for large domains

SAMRAI provides infrastructure support for SAMR applications research

- Parallel processing support (MPI)
- Shared algorithms
- Interfaces for SAMR data to solvers (PETSc, PVODE, *hypre*)
- Checkpointing & restart support (HDF)
- Parallel tools (VAMPIR, TAU)

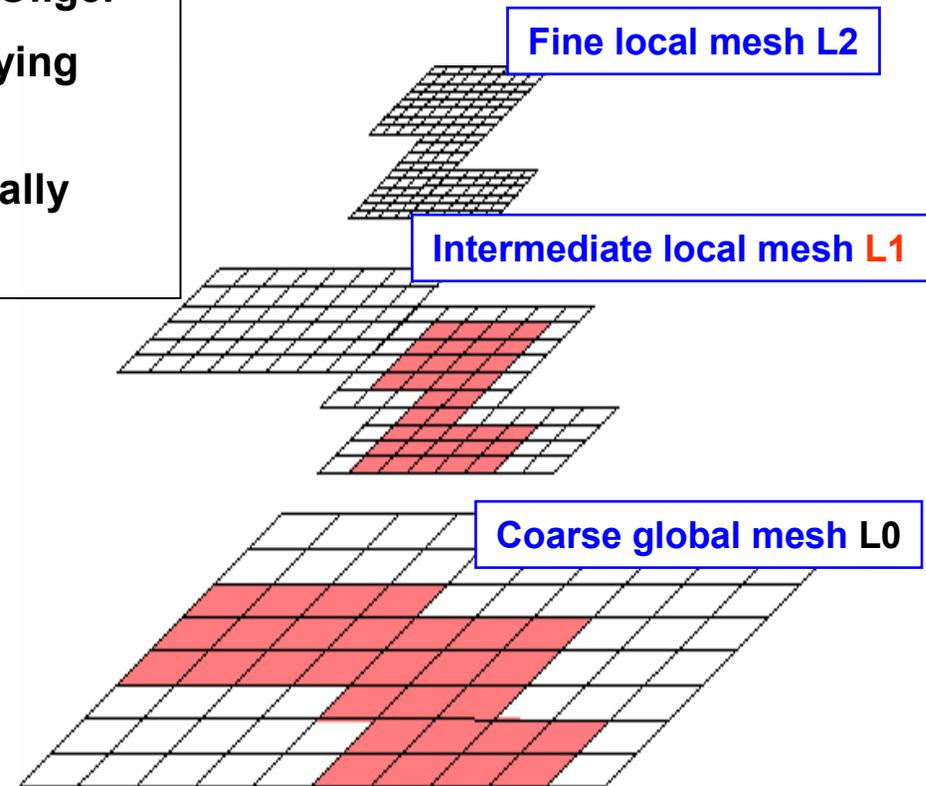
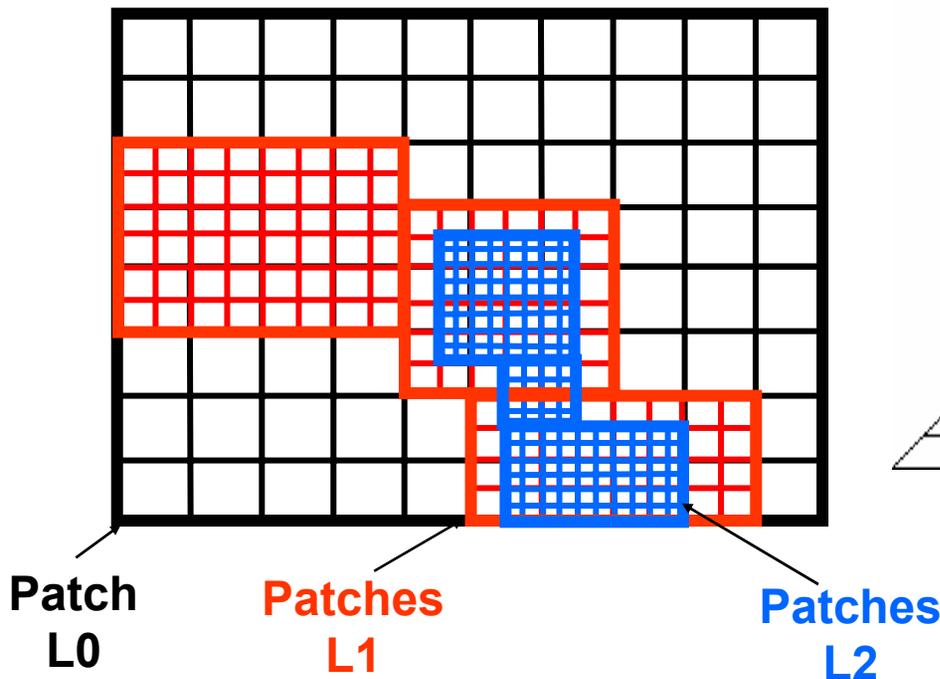
Integrating performance tools with SAMRAI allows them to be used for a variety of applications

Outline

- SAMRAI introduction
- **Parallel implementation of SAMR**
- Parallel performance measurements
- Use of performance tools in SAMRAI
- Performance information “wish list”

Structured AMR (SAMR) employs a dynamically adaptive “patch” hierarchy

- Based on methods of Berger, Colella, Olinger
- Hierarchy defines nested levels of varying mesh resolution
- Data stored on patches covering logically rectangular index space



Dynamic mesh adapts to features as solution evolves

Adaptive solution of Euler equations

Initial conditions:

inside sphere

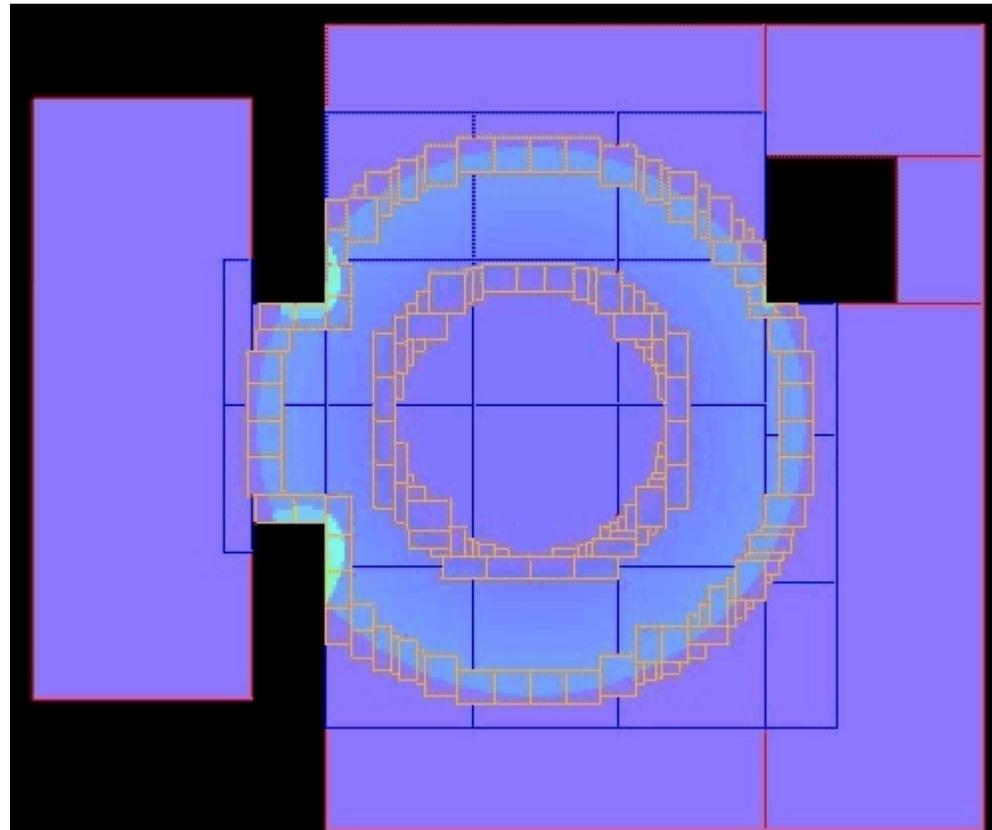
density = 8.0

pressure = 40.0

outside sphere

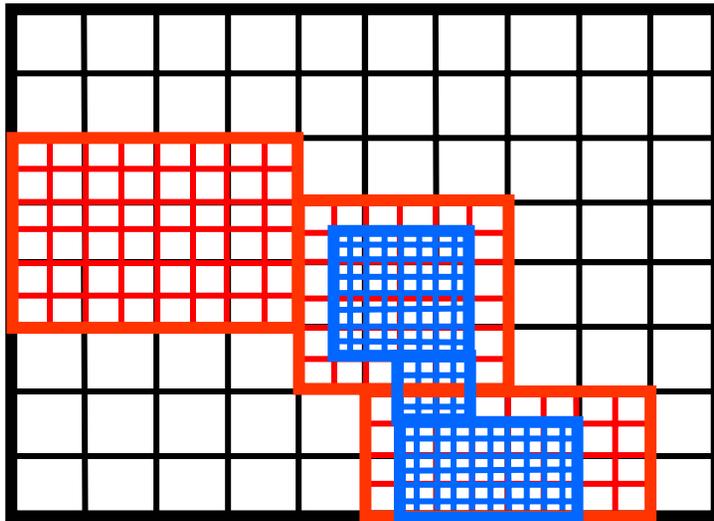
density = 1.0

pressure = 1.0

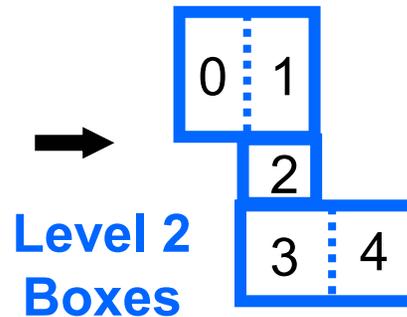


Patches distributed to processors to balance computational workload

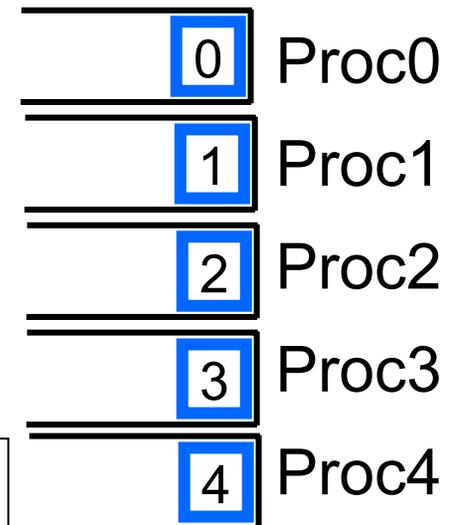
1) Box regions constructed



2) Boxes split to construct patches



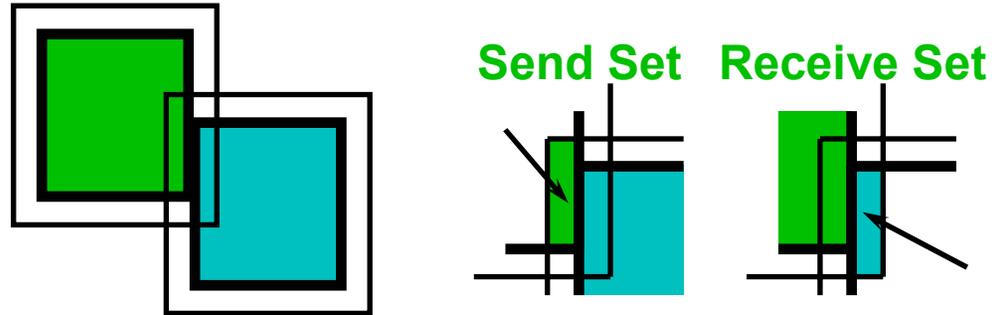
3) Patches bin-packed to processors



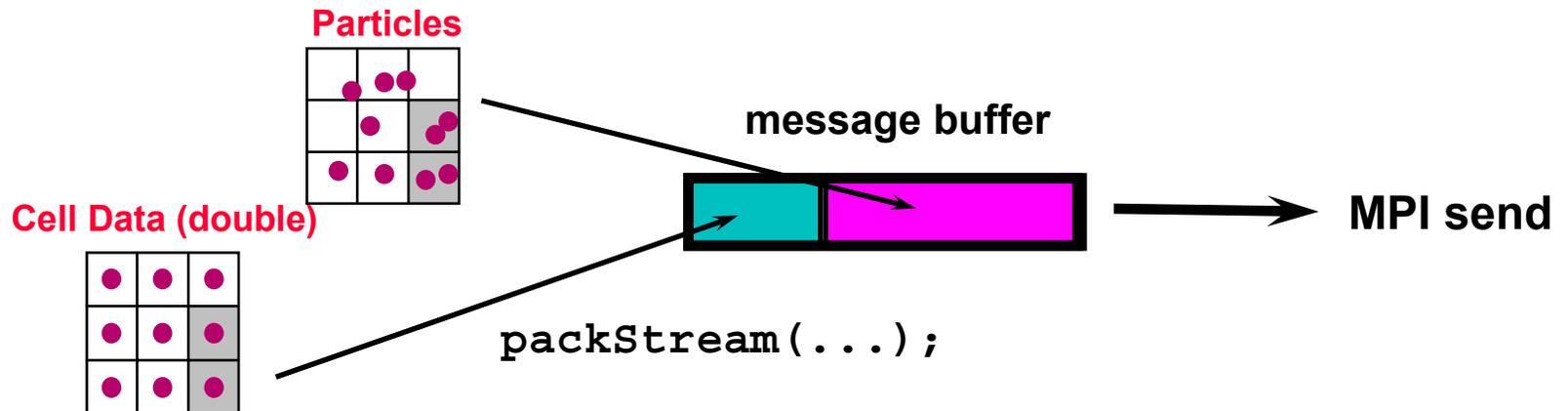
- Generally have multiple patches per processor
- Each level load balanced separately
- Spatial bin packing may be used to maintain locality of patches on processors

Communication schedules create and store data dependencies

- Amortize cost of creating send/receive sets over multiple communication cycles



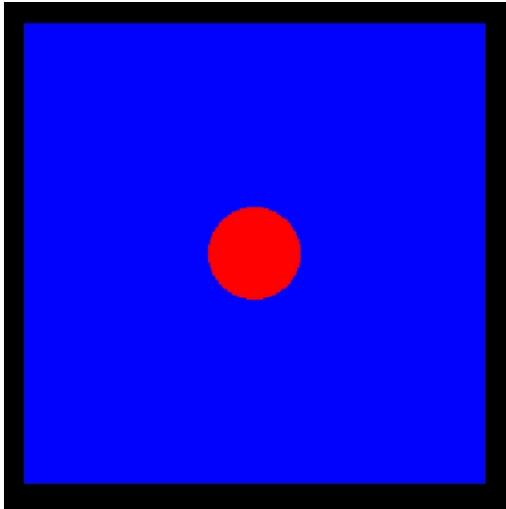
- Data from various sources packed into single message stream
 - supports complicated variable-length data
 - one send per processor pair (low latency)



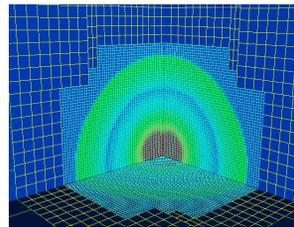
Outline

- SAMRAI introduction
- Parallel implementation of SAMR
- **Parallel performance measurements**
- Use of performance tools in SAMRAI
- Performance information “wish list”

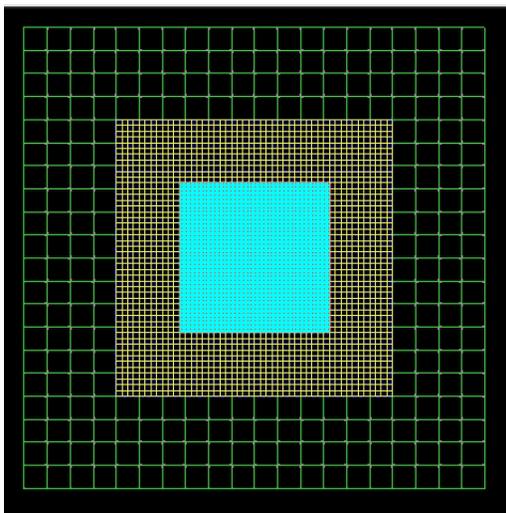
Non-scaled Euler benchmark – same problem size run on all processors



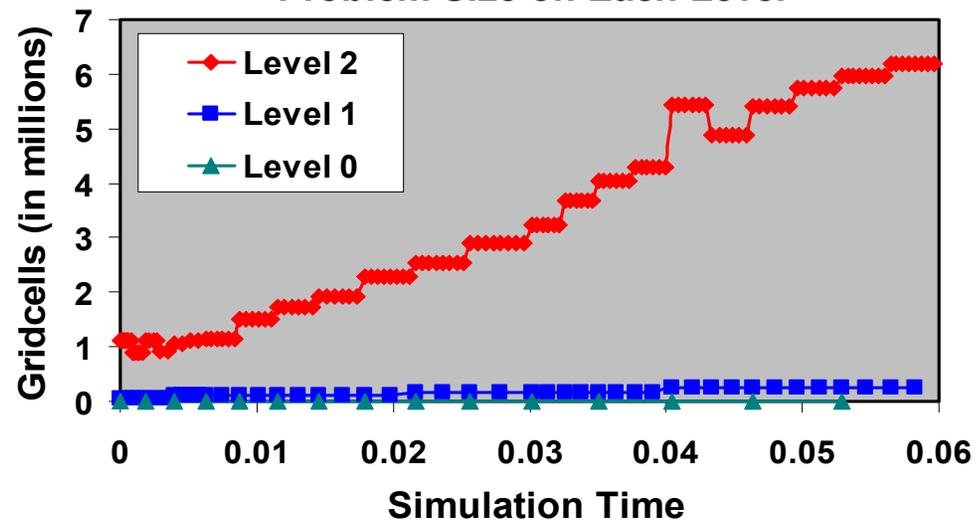
3D spherical shock - Euler hydrodynamics



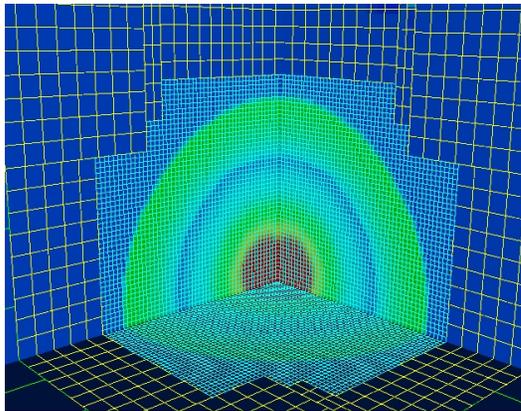
- Workload changes over simulation
- Per-processor workload decreases as number of processors increased



Problem Size on Each Level

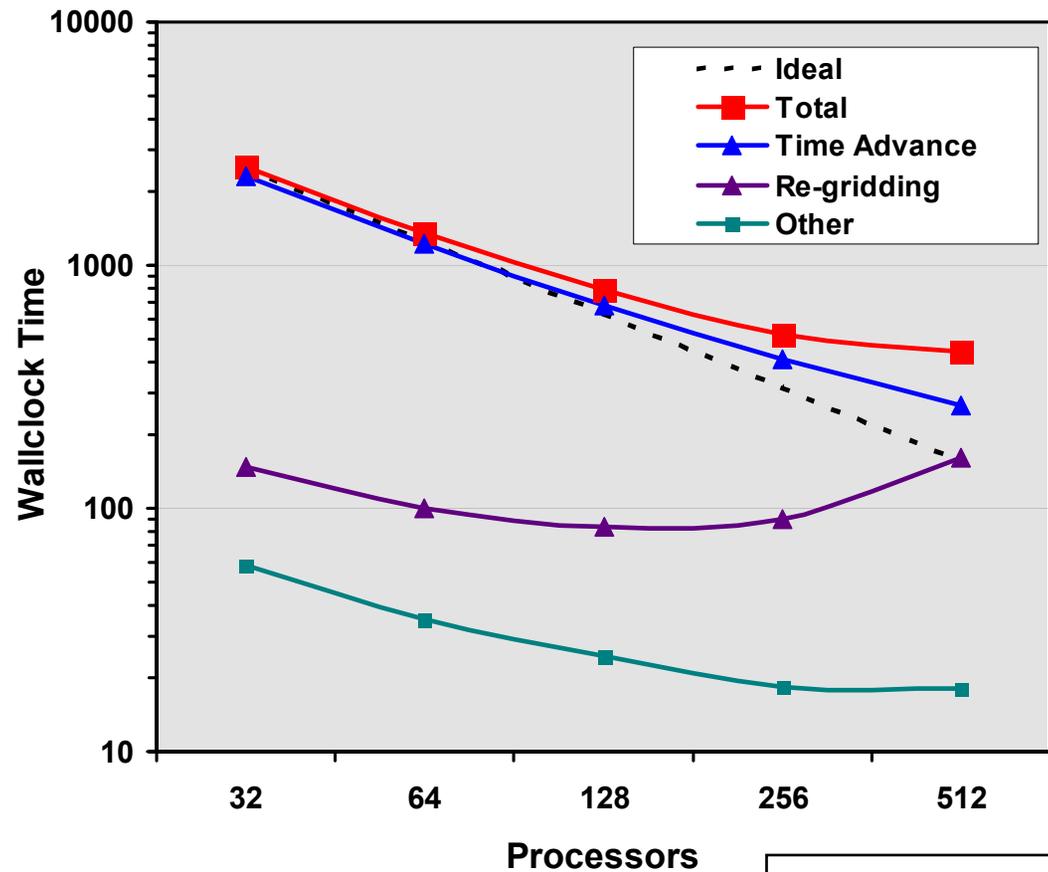


Parallel Performance of *non-scaled* adaptive Euler benchmark



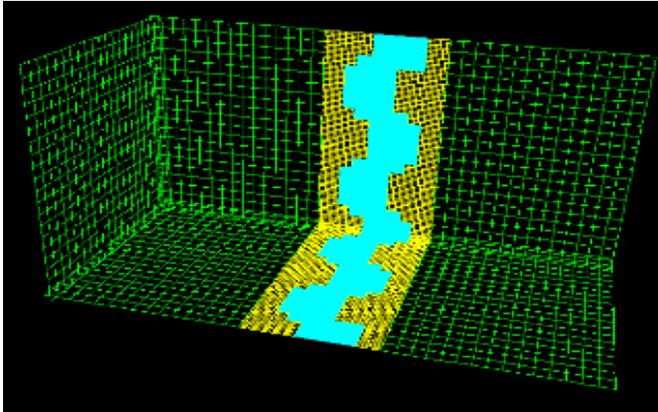
Non-scaled
Euler calculation
ASCI IBM Blue Pacific

Measured Solution Time on Various Processors (3 Level Euler Sphere Problem)



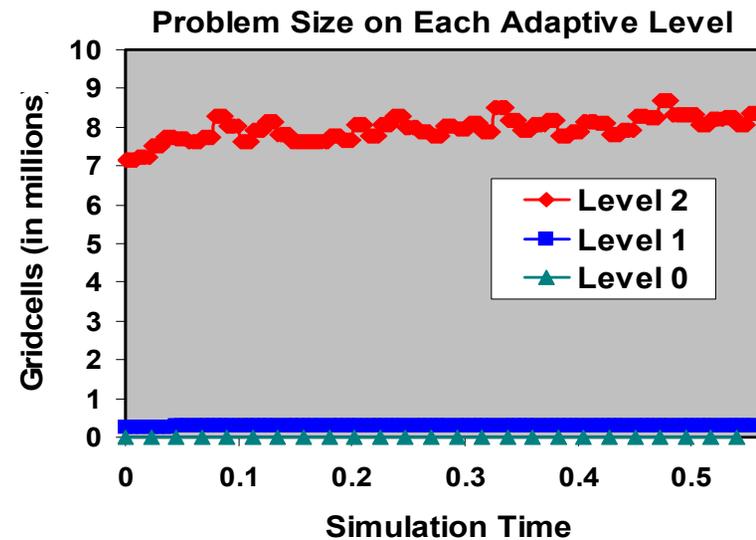
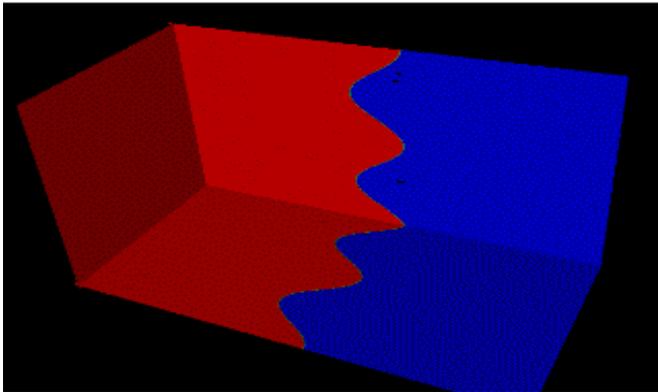
November 2001

Scaled linear advection benchmark – problem size increased with processors

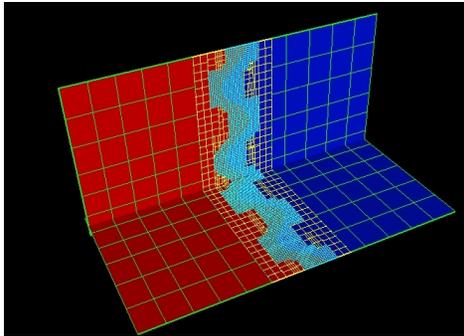


3D advecting sinusoidal front - linear advection

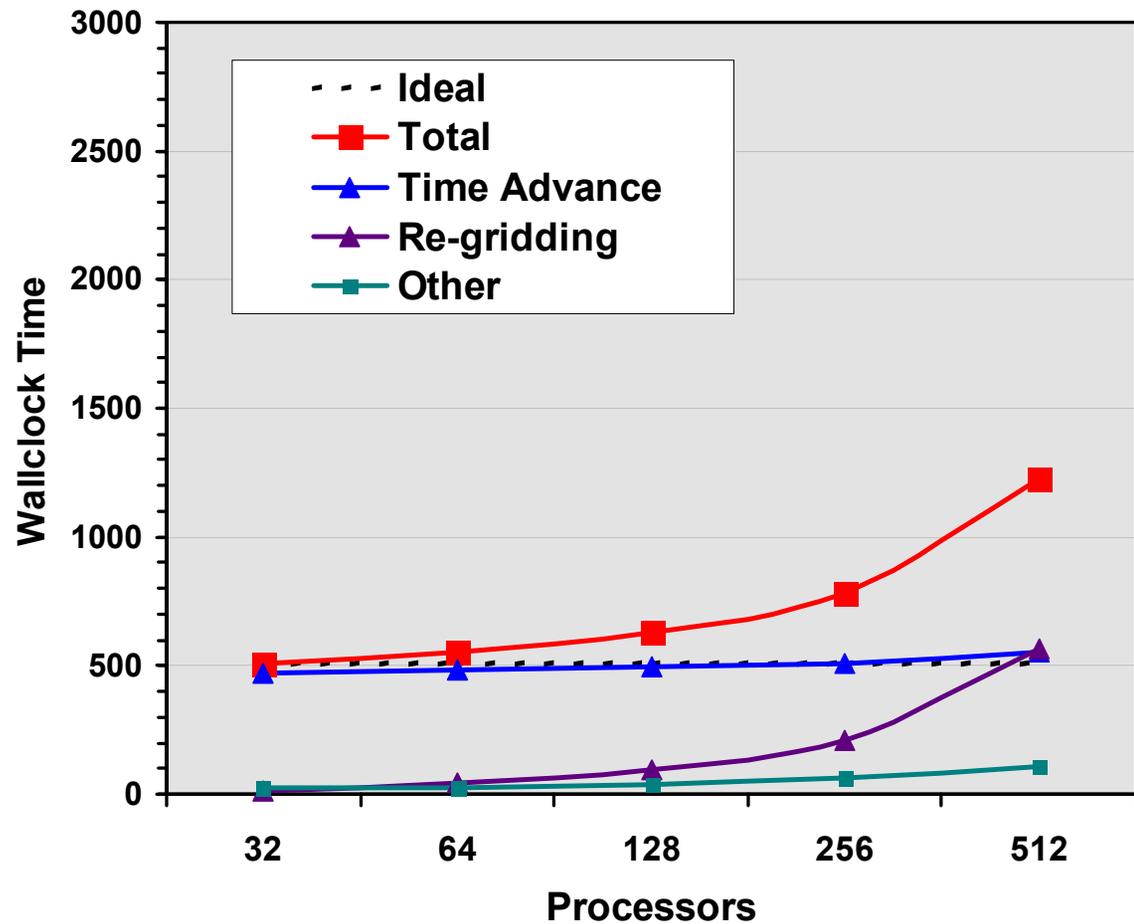
- Workload uniform over simulation
- Per-processor workload remains constant as number of processors is increased



Scaled results with new graph-based schedule construction algorithm



Scaled
Linear advection
calculation
IBM ASCI Blue Pacific



March 2002

Outline

- SAMRAI introduction
- Parallel implementation of SAMR
- Parallel performance measurements
- **Use of performance tools in SAMRAI**
- Performance information “wish list”

“Homegrown” timers in SAMRAI

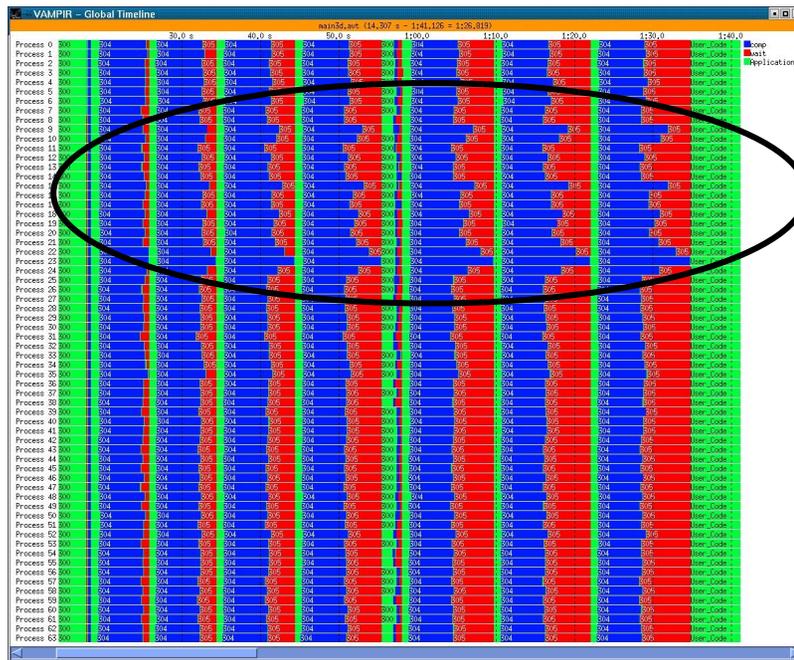
- SAMRAI Timers:

```
static Timer timer = TimerManager->getTimer("pkg::class::method");  
timer->start();  
...  
timer->stop();
```

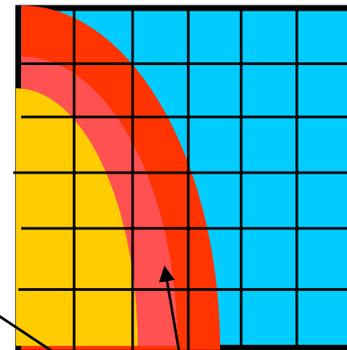
- Library instrumented to maintain a set of timers around routines of interest
- Users can manually enter timer calls into their application
- May be turned on/off via input file (e.g. `timer_list = "pkg::*::*"`)
- Accesses posix `times()` function and/or `MPI_Wtime()`
- Provides per-processor, inclusive and exclusive times as well as reduced time over processors (e.g. max, average time)

SAMRAI timers may invoke VAMPIR

- VAMPIR links added with help from J. Vetter, 10/00
- Useful for tracking down load imbalance problems
- Message stats minimally helpful – trace too dynamic to make sense of.



VAMPIR trace of spherical shock calculation 64 processors

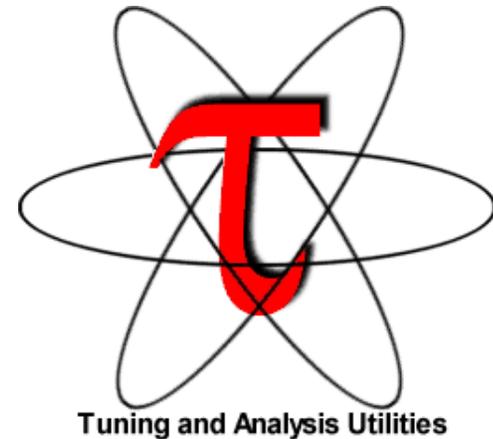


Computation **more expensive** in cells around rarefactions

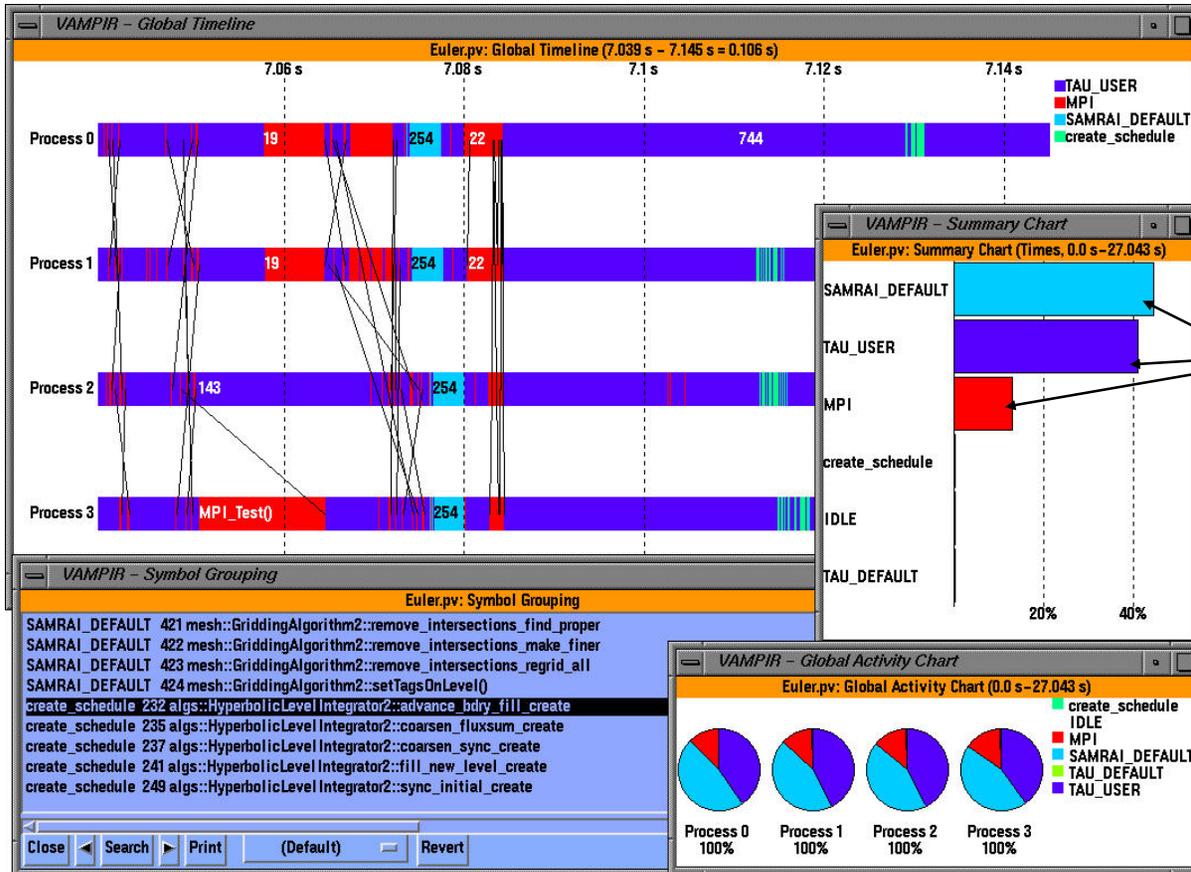
- Colella-Woodward Godunov scheme for strong shocks
- Nonlinear its in Riemann solver

SAMRAI - TAU

- **We began collaboration with Tau in Spring 2002 to apply their tools in SAMRAI.**
- **Advantages of Tau:**
 - Freely available
 - Automatic instrumentation capability
 - Readily configurable
- **SAMRAI v1.3 (released 9/02) contains links to Tau (thanks to S. Shende!)**

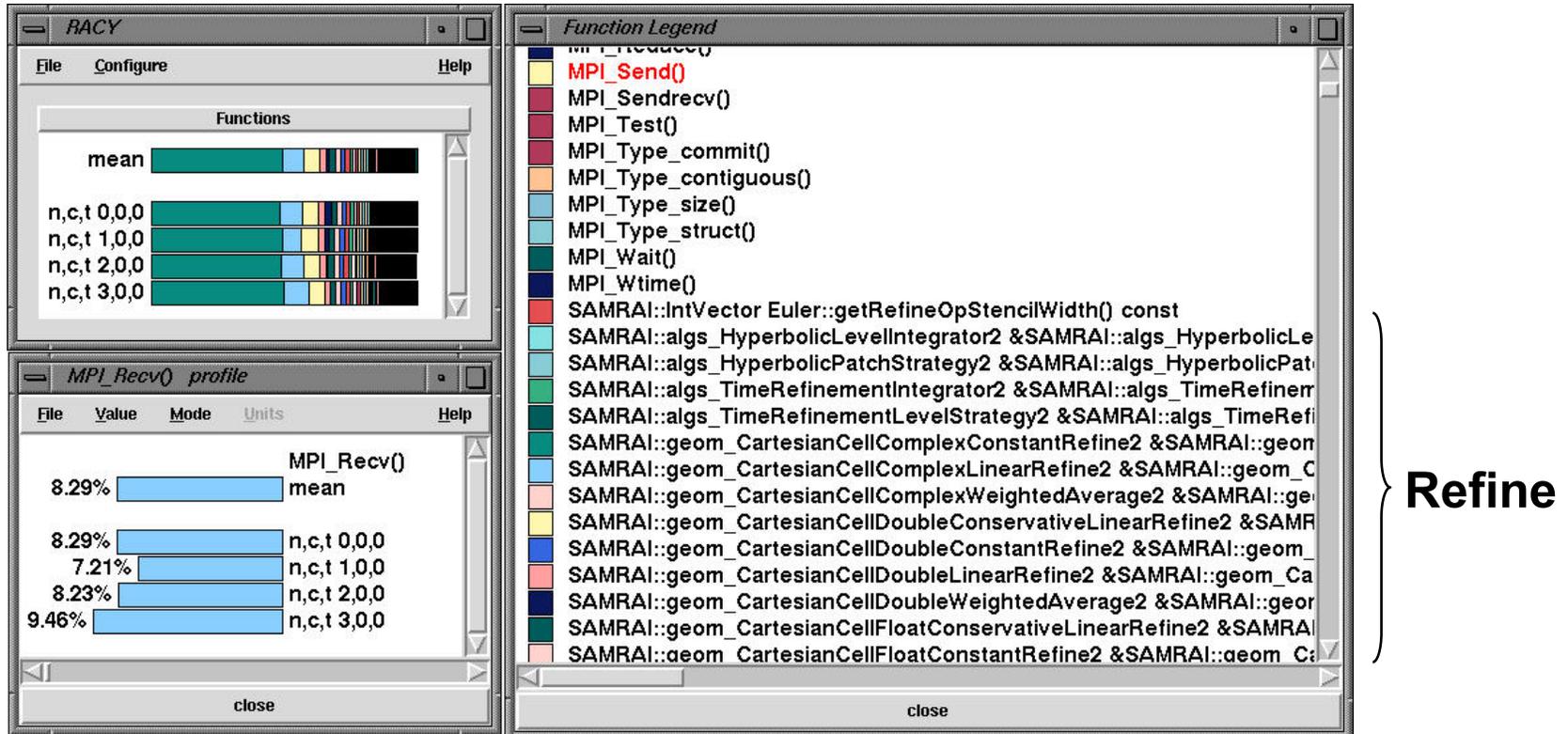


TAU Groups



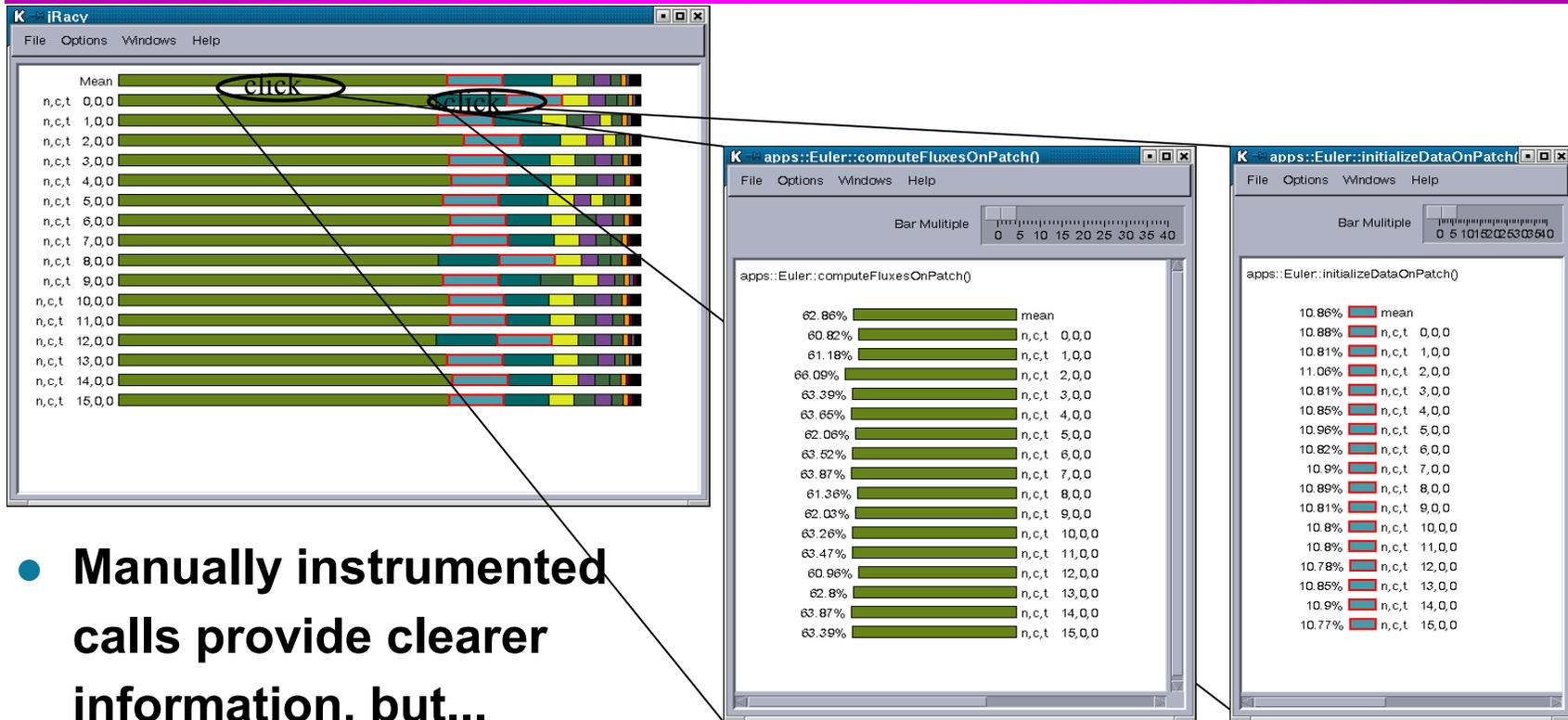
- Organizes timers into groups
- Useful for application codes built on top of SAMRAI to distinguish their costs.

Automatic Instrumentation with TAU Programming Database Toolkit (PDT)



- Automatic instrumentation desirable to users, but...
- High-level classes in SAMRAI invoke many lower level routines, leading to information overload.

TAU calls invoked by SAMRAI Timers



- Manually instrumented calls provide clearer information, but...
- Requires more intervention by users.

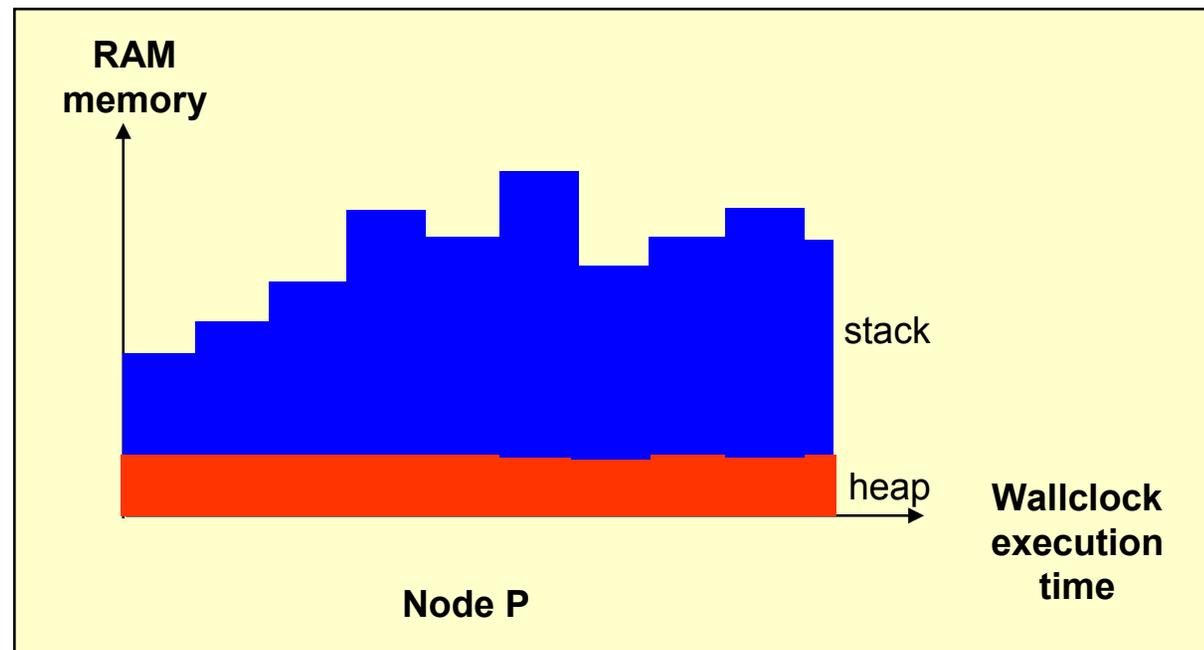
Tradeoff: time to manually instrument vs. time to parse automatically instrumented information.

Outline

- SAMRAI introduction
- Parallel implementation of SAMR
- Parallel performance measurements
- Use of performance tools in SAMRAI
- **Performance information “wish list”**

Memory Analysis

- Dynamic applications like AMR often lead to **memory imbalances**.
- Many scientific applications are **memory bound**, rather than CPU bound.
- A tool that provides information about dynamic memory usage would be useful.



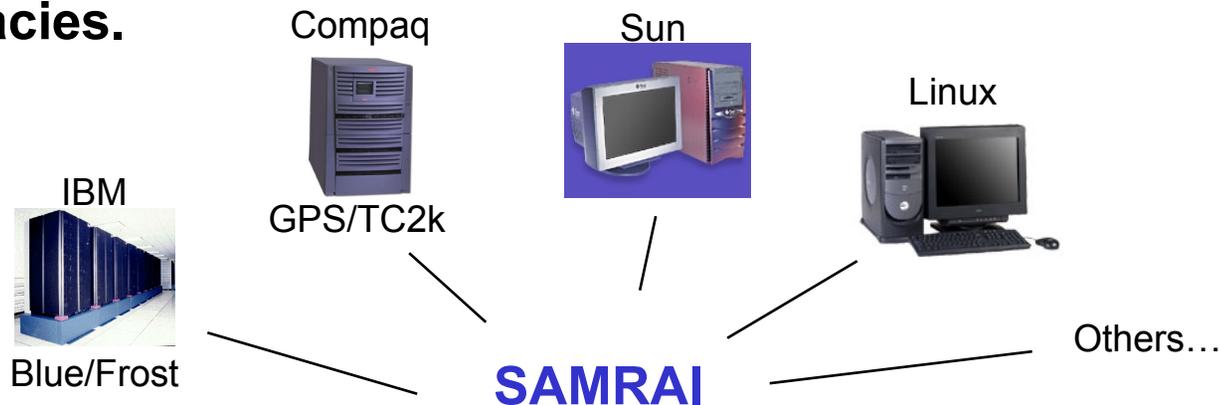
Execution rate analysis

- **SAMRAI makes efforts to maximize execution performance:**
 - C++ classes in SAMRAI manage “overhead” operations (e.g. gridding, communication dependencies, etc).
 - Numerical operations by F77 kernels
- **Applications are interested in **execution rates** of various parts of their algorithm.**
- **VAMPIR/Tau have links to PAPI provide hardware counter statistics.**

It would be useful to combine hardware counters and timing information to provide statistics on execution rates.

Performance information in code testing

- Like many larger-scale software development efforts, SAMRAI runs nightly autotests on various platforms to detect bugs, inaccuracies.



- It is also desirable to identify significant changes in performance
 - Tools that distinguish performance variations and alerts when significant differences arise.

Concluding Remarks

- **Performance tools are essential for parallel scientific applications, particularly for dynamic calculations like AMR.**
- **Experiences with existing tools:**
 - Simple text-based timers are heavily used.
 - VAMPIR/Tau useful in providing more detailed information, such as load imbalances.
 - Communication pattern information only marginally useful – AMR calculations too dynamic to establish trends.
- **Features that we would find useful in future tools:**
 - RAM memory use analysis.
 - Reports of execution *rates*.
 - Incorporation of performance analysis into testing.

Auspices Statement

- **This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.**

- **Document UCRL-PRES-150627**