TITLE: Parallel Algorithm Development

AUTHOR(S): Thomas F. Adams

SUBMITTED TO: For presentation at the 4th Russian-American Mathematical Conference, May 20-24, 1996, Snezhinsk (Chelyabinsk-70), Russia

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
Rapid changes in parallel computing technology are causing significant changes in the strategies being used for parallel algorithm development. One approach is simply to write computer code in a standard language like FORTRAN 77 or with the expectation that the compiler will produce executable code that will run in parallel. The alternatives are: 1) to build explicit message passing directly into the source code, or 2) to write source code without explicit reference to message passing or parallelism, but use a general communications library to provide efficient parallel execution. Application of these strategies is illustrated with examples of codes currently under development.

The first example is RAGE, an Eulerian radiation hydrocode with adaptive mesh refinement. This code was written in FORTRAN 77. It currently executes on the Cray parallel vector supercomputers (PVP), such as the YMP and the J90, and on the Cray massively parallel T3D. Its data structure, designed to accommodate adaptive mesh refinement, is ideally suited to the global shared memories found in PVPs. RAGE executes reasonably well on the T3D simply allowing the compiler to provide parallelization. However, in order to achieve better parallel performance and scalability, an exploratory effort is being undertaken to convert RAGE to FORTRAN 90 and use an underlying communications library. Examples of RAGE calculations of Richtmyer-Meshkov and Kelvin-Helmholtz instabilities will be shown.

A different approach is being taken to parallelization of the TECOLOTE code, an Eulerian hydrocode with high order advection and mixed cell interface reconstruction. This code is written in C with explicit message passing. However, the message passing is all done in an isolated set of subroutines, effectively producing a message passing library within the code. By isolating the message passing, the actual communication calls can be adapted to the computing platform. It has been very easy to substitute message passing using the Cray SHMEM library with standard PVM or MPI calls. TECOLOTE will run well on a wide range of computing platforms from workstations to the T3D, and is expected to run well on more traditional massively parallel processors (MPP), such as the Intel Paragon. In the future, another alternative will be explored, in which the TECOLOTE communications library is replaced with the general POOMA library.

The next approach is to write the code in a standard language, such as FORTRAN 90, with gather/scatter constructs, but use an underlying communications library, the Parallel Gather/Scatter Library, PGSLib for the communications. The strategy is a layered one in which the source code is first written to be optimized by the compiler. Under that layer, there may be communications calls customized by the user that access a communication library at the next lower level. The communications library is built on a communication layer interface to a hardware-specific system communication layer. The communications library and interface insure that the code is portable, which the system communication layer is optimized for communications efficiency on each platform of interest.

An example of the use of PGSLib is given with the CHAD code. CHAD is a 3D implicit hydrocode using an unstructured hybrid grid with all types of elements ranging from tetrahedra to hexahedra. CHAD also has implicit heat conduction, multiple species, chemistry, and spray dynamics. CHAD is primarily used for automotive design including in-cylinder combustion, external aerodynamics, under-hood cooling, and air conditioning. Examples to be shown including the mesh partitioning of an automotive piston, and simulations of swirling flow in an engine and in the stationary vortex Gresho test problem, compression in the standard spherical Noh problem, and heat conduction through a highly disordered mesh.

Another example of codes written in FORTRAN 90 and using PGSLib are the NIKE/ATHENA code for 3D radiation transport, and the TELLURIDE code for simulating
fluid flow and solidification. TELLURIDE calculations of aluminum alloy casting will be shown.

The strategies described here already give good parallel performance on existing computers. In order to scale to even greater computing power, we expect that a new generation of parallel computers will be developed by clustering high-end shared memory multiprocessors (SMPs). These computers will have the advantage of the shared memory within an SMP for efficient execution, coupled with the scaling possible by linking the SMPs with high speed communications. There will be a natural hierarchy of latencies in such a cluster of SMPs. An important research topic will be to see how well current parallelization strategies work with the clustered SMPs, and to develop new strategies to take advantage of their hierarchical nature.
The RAGE Code: A 3D Adaptive Mesh Refinement Eulerian rad-hydro code

developed by M.L. Gittings
Science Applications International Corporation

- multi-dimensional Eulerian hydrodynamic code
  * 1D, 2D and 3D -- with radiation
- adaptive mesh refinement (AMR) ***
- second-order accurate Godunov
- all cell centered variables
- multi-material EOS with strength
- square zones in 2D, cubes in 3D
- implicit, gray, non-equilibrium radiation diffusion
- JWL explosive EOS
- Gittings is under contract with LANL to collaborate on 2D/3D AMR rad-hydro code development
Holmes, Grove & Sharp - Single Interface

- Richtmyer-Meshkov Instability
- $M_s = 1.2$
- $\text{SF}_6$-air $[\rho(\text{SF}_6)/\rho(\text{air}) = (4.85\times10^{-3})/(9.5\times10^{-4}) = 5.1]$  
  - initial amplitude = 0.24 cm  
  - wavelength = 3.75 cm  
  - 8 levels: 1.25 cm - 0.0097 cm

Velocity of Amplitude Growth

- Linear
- Impulsive model
- Experiment
- HG&S paper
- RAGE**

**coarse RAGE: time resolution

Los Alamos National Laboratory
Thermonuclear Applications XTA
2-D KELVIN-HELMHOLTZ INSTABILITY

RAGE VOLUME FRACTION PLOTS

- Dimensions: 8.0 cm x 10.0 cm
- Periodic Boundary Conditions in x
- Mach Number: 0.254
- Equal & Opposite Velocities
- Constant Density
- $\delta x_{\text{min}} = \delta y_{\text{min}} = 0.03125\text{ cm}$

<table>
<thead>
<tr>
<th>(I)</th>
<th>(II)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1 = 0.25, \ A_1 = 0.10$</td>
<td>$\lambda_1 = 0.25, \ A_1 = 0.10$</td>
</tr>
<tr>
<td>$\lambda_2 = 0.40, \ A_2 = 0.0625$</td>
<td>$\lambda_2 = 0.40, \ A_2 = 0.0625$</td>
</tr>
<tr>
<td>$\lambda_3 = 10.0, \ A_3 = 0.015625$</td>
<td></td>
</tr>
</tbody>
</table>

Los Alamos National Laboratory
3D Eulerian code

- ASCI applications / platforms
- Portable
- Efficient Memory Usage
- Extends MESA / PAGOSA
TECOLOTE PHYSICS

- Explicit, finite-difference approximation
  - Lagrangian phase
  - Advection remap
- Mixed-cell multi-material treatment with
  - A variety of equations of state
  - Elastic-plastic material strength
  - High-explosive reactive burn
  - Damage
TECOLOTE is written in C with message passing using object-oriented techniques.

Current Advantages:

- Sharing strength, EOS, and damage models with SPH
- Extremely easy to add new physics models
- Runs on T-3D, but has been ported to other platforms
  - HP cluster
  - CRAY T-90 and J-90
  - SGI and SUN workstations
- Will be straightforward to transform into C++ (true object-oriented code)
CHAD

- 3-D implicit hydrodynamics using control-volume finite element formulation
- Scalable, designed to run on uniprocessor workstations, symmetric multiprocessors, clusters of workstations, and massively parallel platforms
- Unstructured hybrid grid (all types of elements ranging from tetrahedra to hexahedra)
- Arbitrary mesh motion
- Second-order accurate
CHAD (Continued)

- Variable implicit convection
- Implicit heat conduction
- All variables colocated at nodes
- Laminar or turbulent flow
- Multiple species and chemistry
- Spray dynamics
- Primary application: automotive design including in-cylinder combustion, external aerodynamics, underhood cooling, and air-conditioning
**NO-UTOPIA Algorithm**

- **NOde centered**
- **Unstructured TOpology** (mixture of tetrahedra, hexahedra, prism, pyramids, etc.)
- **Parallel** (uniprocessors, SMPs, clusters of workstations, and massively parallel platforms)
- **Variable Implicit Advection**
- **Second-order accurate in space and time**
** DESIGN STRATEGY: SOFTWARE LAYERS **

- Computation optimized by compiler
- Customized by user
- Insures library portability
- Currently MPI, but designed to evolve with hardware. Can be optimized for a particular platform.

**User Application**

**User Communication Calls**

**Communication Library**

**Communication Layer Interface**

**System Communication Layer**
Parallel Gather/Scatter Library (PGSLib)

- **Communication Library**
  - Separate computation and communication
  - Specialized library for high performance gather/scatter
  - MPI based, can change with HW

- **FORTRAN 90**
  - Modularity isolates hardware dependent portions, such as communication
  - Object-based maintainable code

- **Use Compiler As Much As Possible**
  - Portability and performance on current and future systems

Platform Independent Source
F90 + Communication Library (+ HPF Directives)

- FORTRAN 90 (or HPF)
- HPF or Library
- PGSLib

Native compiler generates high performance, on processor, computation
Input/Output
Computation Hardware
Communication Hardware and Software (MPI)
Nike/Athena

- 3-D radiation transport
- Fully implicit
- Linear continuous finite element
- Sn or SPn angular differencing
- 2-T multigroup (does not assume Plankian distribution of radiation)
- Electron and ion thermal conduction
- Multifrequency gray (MFG) and diffusion synthetic acceleration (DSA)
TELLURIDE: Physical Model

- Hydrodynamics
- Interface kinematics: volume tracking
- Interface dynamics: surface tension, phase change
- Momentum diffusion
- Thermal transport
- Solidification models
- Species (solute) diffusion
- Strength and residual stress models
- Geometry: generalized unstructured hexahedra
- Portable to all modern computing platforms
- MP paradigm: explicit message passing
# The TELLURIDE Team

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doug Kothe</td>
<td>LANL T-3</td>
<td>Lead Developer</td>
</tr>
<tr>
<td>S. Jay Mosso</td>
<td>LANL XHM</td>
<td>CFD/Interface Tracking</td>
</tr>
<tr>
<td>Jerry Brock</td>
<td>LANL T-3 (PostDoc)</td>
<td>Particle-Based Methods</td>
</tr>
<tr>
<td>Anand Reddy</td>
<td>LANL T-3 (PostDoc)</td>
<td>Solidification/Micro-Macro Segregation</td>
</tr>
<tr>
<td>John Turner</td>
<td>LANL XTM</td>
<td>Linear Equation Solver Library</td>
</tr>
<tr>
<td>Bryan Lally</td>
<td>LANL ESA-EPE</td>
<td>Heat Transfer/Solidification</td>
</tr>
<tr>
<td>Robert Ferrell</td>
<td>CPCA, Ltd</td>
<td>Parallel Gather-Scatter Library</td>
</tr>
<tr>
<td>C. Beckermann</td>
<td>Univ. of Iowa</td>
<td>Solidification Theory and Models</td>
</tr>
</tbody>
</table>

Los Alamos
The TELLURIDE Project: A Team Effort

- The current level of effort affords a very diverse team of staff members, PostDocs, and contractors

- Development is application-driven: modeling the gravity-pour casting processes currently in use at LANL

- Entire team face-to-face meetings are rare! Daily communication takes place electronically (mailing lists, etc.)

- Coordinated team software development a must:
  - Would not be possible without CVS (concurrent versions system)
  - CVS allows parallel development: all developers work on “same” source!
  - CVS allows remote development: source code is easily “checked out” on any remote platform

- Strict adherence to F90 standard enables development on multiple platforms