Possible Stack for Parallel Programming Models for Scientific Computing

Zeyao Mo

Institute of Applied Physics and Computational Mathematics
September 3-5, Juelish, Germany
Contents

1. Challenges for parallel programming
2. Possible domain-specific programming model for scientific computing
3. An instance: JASMIN framework
4. Conclusion
1. Challenges for parallel programming
In the past decade: Parallel Programming

18 years of legacy codes for various applications such as fusion energy, high energy physics, climate forecasting, facility and experimental design, materials, chemistry, etc.

MPI for O(10K) parallelism, OpenMP for O(1) parallelism.

Terascale to Petascale Machines
In the next decade: Three parallel computing Points

Terascale Laptop: Uni-Supernode — few-node — many-core
Petascale Desktop: Multi-Supernode — multi-node — many-core
Exascale Center: Many-Supernode — many-node — many-core

2018 Goal: Make
Petascale = Terascale + more;
Exascale = Petascale + more.

Common elements
Machine Parallelism: Nested/Hybrid Programming Models

Emerging models: Accelerators, Resilience, Energy.
In the next ten years: Parallel Programming

Petascale to Exascale Applications

Reconstruction of 15 years of MPI legacy codes,
New generation codes for multi-physics complex systems

Big and increasing gaps for realization

Nested/Hybrid Parallel Programming Models
MPI $O(10^4)$, DSM-X $O(10)$, OpenMP $O(10^2)$, ILP $O(1)$;
Accelerator: OpenCL/CUDA; Resilience, Energy, ...

Petascale to Exascale machines
Big and increasing gaps for realization

1. Evolving and nested/hybrid parallel programming
2. Complex management for data structure, hierarchical memory, irregular communication, etc.;
3. Multilevel/Hybrid load imbalance arising from physics, numerical stencils, communication, run time status of machine, etc.;
4. Implementation and integration for fast numerical algorithms;
5. Code extensibility for more and more complex applications: O(10K~1M) lines;
6. Data management and visualization.
Great Challenges:

Fussy Parallel Programming
Load imbalance
Fast algorithms implementation
Code complexity
Visualization Interfaces

Good solutions

Scientific Computing and Engineering Applications
Frameworks enable:

Applications, algorithms, parallel experts, computer experts can cooperate tightly in the development of complex codes.

- Encapsulates and separates fussy works of parallel computing from applications (e.g. data structure, parallel programming, numerical libraries);
- Encapsulates code complexity and applies software engineering for code extensibility and maintenance;
- Accelerates the developments of codes towards petascale/exascale.
Meet the expectation of application/physics experts

Think Parallel, Write Sequential

1. Significantly simplify or reuse the parallelism patterns using the emerging programming models;

2. The return on their rewrite efforts can be leveraged for multiple years even the machine is rapidly changing (e.g., 20 years ? 20 or more years for MPI);

3. Once infrastructure in place, ratio of science experts vs. parallel experts: 10:1, physics added as serial code, now and in the future.

--M.Heroux, LANL, July. 2011, DOE Workshop on Exascale Programming Challenges
2. Possible Domain Specific Programming Model for Scientific Computing

------ Think Parallel, Write Sequential
Possible Stack of Programming Models: Frameworks

Domain Scientists: Think Parallel, Write Sequential.

Computers: Petascale to Exascale, performance points
Framework-based DSPM is possible for scientific computing?

DDM Graph Based Patterns: Halo exchanges, Collectives

separate parallel programming from serial codes:
Computational Patterns

Digraph Based Patterns: data driven, dynamic tasks
Halo Exchange Pattern: Separate Parallel Programming
Halo Exchange Pattern: Separate Parallel Programming

ist(1)=ist(3)=1, iend(1)=iend(3)=8 ;
jst(1)=jst(2)=9, jend(1)=jend(2)=16;
ist(2)=ist(4)=9, iend(2)=iend(4)=8 ;
jst(3)=jst(4)=1, jend(3)=jend(4)=16.

DO b= 1, 4
DO i = ist(b), iend(b)
DO j = jst(b), jend(b)

ENDDO for i,j,b
Halo Exchange Pattern: Separate Parallel Programming

Do MPI message passing fill ghost cells for \texttt{uo} ;

DO \texttt{b} = 1, 4 \texttt{ in parallel}
DO \texttt{i} = \texttt{ist(b), iend(b)}
DO \texttt{j} = \texttt{jst(b), jend(b)}

\texttt{uo(i,j+1)} \quad \texttt{uo(i+1,j)} \quad \texttt{uo(i+1,j)}

\texttt{uo(i,j-1)}

\texttt{ENDDO} for \texttt{i,j,b}
Halo Exchange Pattern: Separate Parallel Programming

DO MPI message passing
    fill ghost cells for uo ;
DO b = 1, 4 in parallel

    serial computation for block b ;

ENDDO for b

DO i = ist(b), iend(b)
DO j = jst(b), jend(b)

ENDDO for i,j

User Part
Halo Exchange Pattern: Separate Parallel Programming

DO b=1, NB
  fill ghost cells for uo ;
DO p=1, NP in parallel
DO b=bst(p), bend(p)
  serial computation for block b ;
ENDDO for b
ENDDO for p in parallel
3. An instance: JASMIN framework

------ Think Parallel, Write Sequential
JASMIN: Parallel Patterns + Library ➔ DSPM

- DDM Graph Based Patterns: Halo exchanges, Collectives
- Digraph Based Patterns: data driven, dynamic tasks, barriers
- Numerical Libraries
- C++ Components = parallelism/libs + interfaces ➔ serial numerical subroutines
3.1 JASMIN

Structured Grid

Inertial Confinement Fusion
Global Climate Modeling

Particle Simulation

Unstructured Grid

http://www.iapcm.ac.cn/jasmin,

J parallel Adaptive Structured Mesh Infrastructure
3.1 JASMIN

Motivations:

Supports the developments of parallel codes for large scale scientific computing on personal computers.

- Hides parallel programming using millions of cores and the hierarchy of parallel computers;
- Integrates the efficient implementations of parallel fast algorithms;
- Provides efficient data structures and solver libraries;
- Supports software engineering for code extensibility.
3.2 Basic Ideas

Physical Models
Discrete Stencils
Numerical Algorithms

Data Dependency

Data Structure

Computational Patterns

Communications
Load Balancing

Computers

extract
form
Promote

support
3.2 Basic Ideas

- Special Models
  - Stencils
  - Algorithms

- Extract
- Data Dependency

- Form
- Data Structure
- Computers
- Promote

- Separate
- Computational Patterns

- Support
- Communications
- Load Balancing

- Library
  - Common Models
  - Stencils
  - Algorithms
3.2 Basic Ideas

- Applications Codes
- Parallel Programming Interfaces
- Data Dependency
- Data Structures
- Computational Patterns
- Communications
- Load Balancing
- Automatic parallelism
- Support
- Parallel middlewares for scientific computing on structured meshes
3.3 key factors

- Data Structures
- Fast Algorithms
- Parallelization
- Component Interfaces
3.3.1 Data Structures

Neighboring Graph Based Patterns: Halo exchanges, Collectives
3.3.1 Data Structures

Mesh supported

- patch-based SAMR
- single block deforming mesh
- Multi-block deforming mesh
- particles
- longitude-latitude mesh
- CFD multi-block mesh
3.3.2 Communications

Distributed Undirected Graph
(2 processors)
3.3.2 Communications

Parallel Sweeping for Sn transport

Digraph Based Patterns: data driven, dynamic tasks
3.3.3 dynamic load balancing

Global atmosphere model

day, night

extreme load imbalance

Radiation and neutron transport

Regional rainstorm model
3.3.3 dynamic load balancing

space filling curves : 2D,3D->1D
MAW/Cycle methods : 1D balancing
3.3.3 dynamic load balancing

Redistributes particles among 8 Processes
3.3.4 Solvers for linear systems

Radiation hydrodynamics:
- Usual algorithms: $O(N^{1.5})$
- PAMG+DDM: $O(N \log N)$

Data structures
Solver interfaces
- JXP
- Kinsol
- Hypre
- ...
3.3.5 SAMR

h-adaptivity: Advance-Estimate-Tag-Refine-Synchronize

- Coarser level integrates
  - (user provides stencil)

- Error estimation
  - (user provides)

- Tag cells for refinement

- Cluster tagged cells into Boxes

- Finer level is initialized and integrates

- Synchronization

- Generate patches for finer level

Input tagged cells

Clustering algorithm

Output boxes
3.3.5 SAMR: data structure

PatchHierarchy

PatchLevel 2

PatchLevel 1

PatchLevel 0

SAMR

Finest level

Middle level

Coarsest level
3.3.5 SAMR: data structure

PatchHiearchy → SAMR

PatchLevel 2 → Finest level

PatchLevel 1 → Middle level

PatchLevel 0 → Coarsest level

PatchHiearchy → PatchLevel → Patch (PatchData)
3.3.5 SAMR: Communication

**SAMR:** using flux conservation interpolation schemes of high order precision

\[
F^{\text{coarse}}_{i-\frac{1}{2}, j} = \frac{1}{r^2} \sum_{q=0}^{r-1} \sum_{p=0}^{r-1} F^{\text{fine}}_{k+\frac{1}{2}, m+p} \left( t + q\Delta t^{\text{fine}} \right)
\]
ICF 2-D radiation hydrodynamics simulation using LARED-S using three levels of SAMR meshes.

Speedup = 160.

<table>
<thead>
<tr>
<th>Resolution of coarsest level</th>
<th># levels</th>
<th># cells</th>
<th># steps</th>
<th># CPU cores</th>
<th>Time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution of finest level: 10240x2048. Physical time =1.324ns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10240x2048</td>
<td>1</td>
<td>2097</td>
<td></td>
<td>1024</td>
<td>20 days</td>
</tr>
<tr>
<td>640x128</td>
<td>3</td>
<td>38 ~ 124</td>
<td>28700</td>
<td>64</td>
<td>24.6</td>
</tr>
</tbody>
</table>
3.3.6 User Interfaces

Time integration class on hierarchy:
algs::HierarchyTimeIntegrator<DIM>

Strategy for time integration on levels:
algs::TimeIntegratorLevelStrategy<DIM>

JASMIN supports

Main program

User time integrator for patch level

Strategy for time integration on patches:
algs::StandardComponentPatchStrategy<DIM>

User numerical subroutines for patches

C++ Components for automatic parallelization:
initialize, step, numerical, memory, copy, synch., sweeping, particles, etc.

User implements
3.4 Current Version

User provides: physics, parameters, numerical methods, expert experiences, special algorithms, etc.

**Architecture:** Multilayered, Modularized, Object-oriented;

**Codes:** C++/C/F90/F77, MPI +MPI +MPI +OpenMP, 660K LOC;

**Installation:** Personal computers, Cluster, MPP.
3.4 Current Version

Applications currently

25 codes

- CFD
- Climate forecasting
- Instabilities & Turbulence
- Materials
- PIC
- Radiation Hydrodynamics

JASMIN

3.4 Current Version

Applications currently

25 codes

- CFD
- Climate forecasting
- Instabilities & Turbulence
- Materials
- PIC
- Radiation Hydrodynamics

JASMIN
### 3.5 Some Applications on PetaFlops System TianHe-1A

<table>
<thead>
<tr>
<th>Codes</th>
<th># CPU cores</th>
<th>Codes</th>
<th># CPU cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>LARED-S</td>
<td>32,768</td>
<td>RH2D</td>
<td>1,024</td>
</tr>
<tr>
<td>LARED-P</td>
<td>72,000</td>
<td>HIME3D</td>
<td>3,600</td>
</tr>
<tr>
<td>LAP3D</td>
<td>16,384</td>
<td>PDD3D</td>
<td>4,096</td>
</tr>
<tr>
<td>MEPH3D</td>
<td>38,400</td>
<td>LARED-R</td>
<td>512</td>
</tr>
<tr>
<td>MD3D</td>
<td>80,000</td>
<td>LARED Integration</td>
<td>128</td>
</tr>
<tr>
<td>JMES-FDTD</td>
<td>60,000</td>
<td>NEPTUNE</td>
<td>1,024</td>
</tr>
</tbody>
</table>

Simulation times: several hours to tens of hours.
12 codes, 48 researchers concurrently develop Inertial Confinement Fusion Application Codes.

- Different Combinations
  - Numerical methods
  - Physical parameters
  - Expert experience

- Hides parallel computing and adaptive implementations using tens of thousands of CPU cores;
- Provides efficient data structures, algorithms and solvers;
- Support software engineering for code extensibility.
## Applications-1: Inertial Confinement Fusion

<table>
<thead>
<tr>
<th>Codes</th>
<th>Year 2004</th>
<th>Year 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LARED-H</strong></td>
<td>serial</td>
<td>Parallel</td>
</tr>
<tr>
<td>2-D radiation</td>
<td>Single block</td>
<td>Multiblock</td>
</tr>
<tr>
<td>hydrodynamics</td>
<td></td>
<td>NIF ignition target</td>
</tr>
<tr>
<td>Lagrange code</td>
<td>Without capsule</td>
<td></td>
</tr>
<tr>
<td><strong>LARED-R</strong></td>
<td>Serial</td>
<td>Parallel (2048 cores)</td>
</tr>
<tr>
<td>2-D radiation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transport code</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LARED-S</strong></td>
<td>Single level</td>
<td>SAMR</td>
</tr>
<tr>
<td>3-D radiation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hydrodynamics</td>
<td>2-D: single group</td>
<td>Multi-group diffusion</td>
</tr>
<tr>
<td>Euler code</td>
<td></td>
<td>3-D: radiation multigroup</td>
</tr>
<tr>
<td></td>
<td></td>
<td>diffusion</td>
</tr>
<tr>
<td></td>
<td>3-D: no radiation</td>
<td>3-D: radiation multigroup</td>
</tr>
<tr>
<td></td>
<td></td>
<td>diffusion</td>
</tr>
<tr>
<td><strong>LARED-P</strong></td>
<td>serial</td>
<td>Parallel (36000 cores)</td>
</tr>
<tr>
<td>3-D laser</td>
<td></td>
<td>Terascale of particles</td>
</tr>
<tr>
<td>plasma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>code</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ICF-1: Integration simulations for ignition targets

Multi-physical modeling for radiation hydrodynamics simulations

- Radiation thermal conduction modeling
- Three temperatures conduction modeling
- Transport modeling
- Multigroup diffusion modeling
- Laser transfer modeling
ICF-1: Integration simulations for ignition targets

Radiation temperatures in the hohlraum.

LLNL NIF base target
ICF-2: 3-D simulations for laser plasma interactions

LARED-P: super-strong lasers transfer and focus over the plasma cone and generate high energy electrons.

#mesh: 768M
#partiles: 40G
CPU Cores: 72,000
Para.Effi.: 44%

Phys. time : 320 fs
Sim. time: 4.5 hours
#steps : 9500

Output data
640GB/ 20

33.580 fs
67.298 fs
319.295 fs
ICF-3: 3-D simulations for laser plasma hydrodynamics

LAP3D: lasers filament and self-focus while transfer over a long distance in the lower density plasma environments.

#mesh: 2.1G
CPU Cores: 16,384
Para.Effi.: 50%
Phys. time: 56.5 ps
Sim. time: 12.8 hours
#steps: 41,200

Output data
1.74TB/104
Parallel rendering using 72 cores

Solve the hydrodynamics equations coupled with laser paraxial transfer equations.

Simulation results are coincides with the ALPS codes.
ICF-4: 3-D simulations for radiation hydrodynamics instabilities

LARED-S: radiation hydrodynamics instabilities occur over the capsule interfaces while the capsule shell rapidly slows down.

- # mesh: 160M
- 256x256x256
- # Cores: 32,768
- Para.Effi.: 52%
- P. time: 0.1 ps
- S. time: 39 hours
- # steps: 778,580
- Output data: 4.4TB/670

3-D disturb

Gray: shell materials.
Color: ion temperature for hot spots.

Ideal compress
PDD3D: 3-D discrete dislocation simulations for the locally plastic deformations of metal materials while the shock is enforced.

- #dislocation: 3 M
- CPU Cores: 4096
  - Para.Effi.: 47%
- Phys. time : 5.5 ns
- Sim. time: 12 hours
  - #steps : 1,800
- Output data
  - 196 GB/ 196

The details of dislocation structures are discovered.

The stretch simulations of Cu crystal (0.12 mm³, r=10⁷/s).
- Left: dislocation density;  Right: local structures.
MD3D: 3-D molecule dynamics simulations for the structures and dynamics shock responses of metal materials with nano-scale defects.

- #Molecules: 50G
- #Cores: 80,000
- Para.Effi.: 62%
- Ph.time: 5.5 ps
- Sim. time: 3.0 hours
- #steps: 30,000

Output data: 162 GB / 150

Dislocation holes release and collapse.

Discation holes collapse and interact with each other, hot spots are formed to generate various dynamics responses. **Left:** local spots; **Right:** LLNL’s results in 2005.
Applications-3: Electromagnetic Simulations

**JMES3D**: 3-D FDTD simulations for destroy mechanism of electromagnetic waves with different frequencies and directions.

- # mesh: 1.2 G
- #Cores: 60,000
- Para.Effi.: 70%
- Phy.ime : 250 ns
- S.time: 8.1 hours
- #steps : 254,000

Output data
736 GB/ 172

Energy
Electronic Field Snapshot
Applications-4: Climate Forecasting

Atmosphere: GAMIL-JASMIN

Land: CLM-JASMIN

Ocean: LICOM-JASMIN

Ocean ice: CSIM4-JASMIN
Applications-5: CFD

- mesh blocks distribution
- DDM 64 cores

# cells: 9,574,784
# blocks: 194
# cores: 2048
# s/10K steps 800
# speedup 580
4. Conclusion

- Domain specific programming models are possible to enable domain scientists “think parallel, write sequential”.

- A possible stack of programming model for scientific computing is “Framework-based DSPM — MPI — DSM-X — OpenMP — ILP”.

- Numerical fast algorithms are essential components for the implementation of DSPM.

- JASMIN shows the possibility of DSMP on structured meshes.