Possible Stack for Parallel Programming Models for Scientific Computing

Zeyao Mo

Institute of Applied Physics and Computational Mathematics

Sepetember 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.

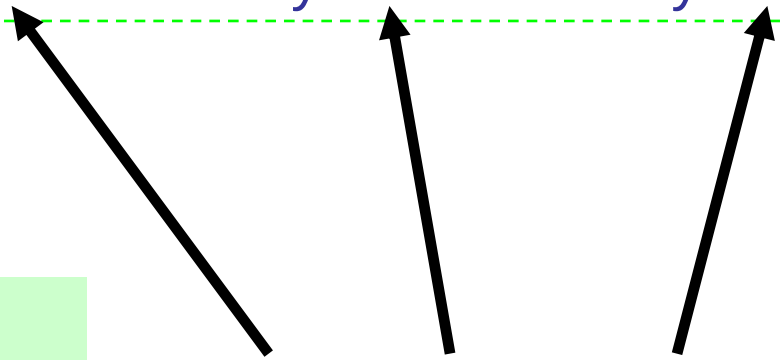
Terascale to Petascale Machines

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

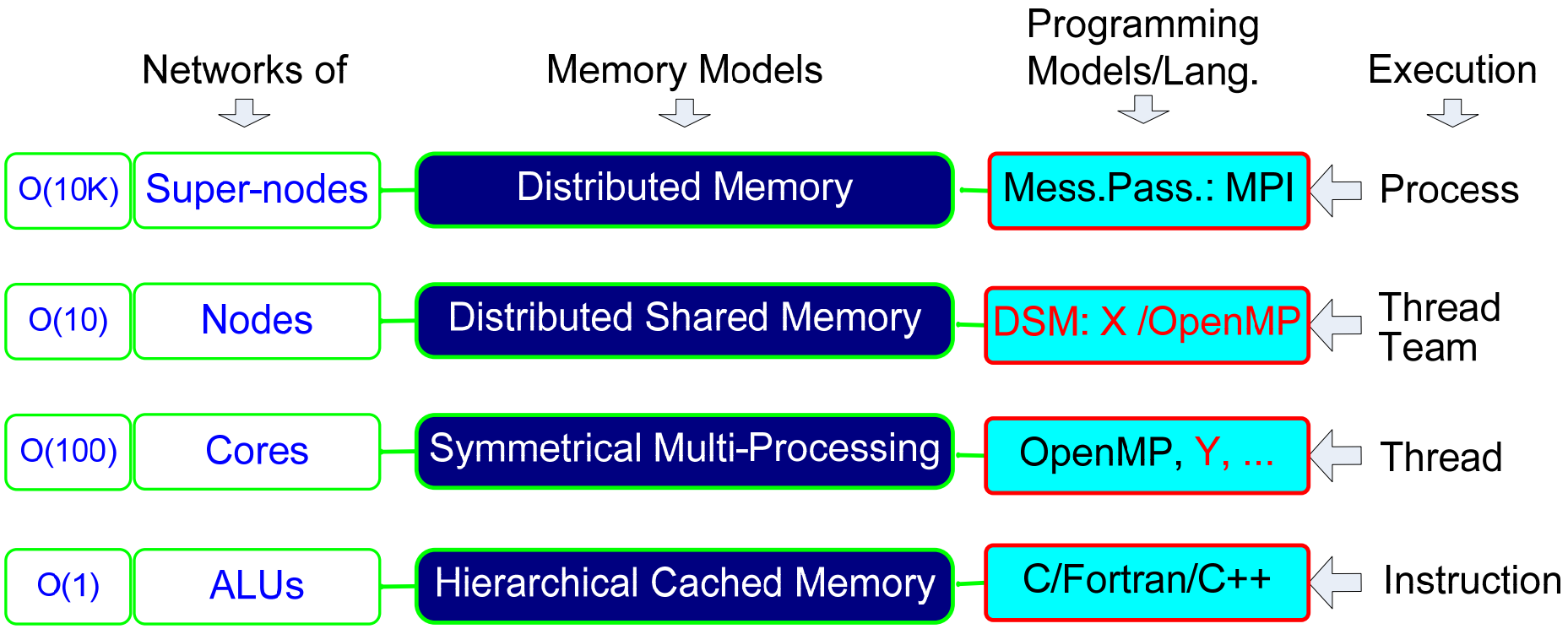


2018 Goal : Make

Petascale = Terascale + more;

Exascale = Petascale + more.

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(10K), DSM-X O(10), OpenMP O(100), ILP O(1);

Accelerator: OpenCL/CUDA; Resilience, Energy, …

Petascale to Exascale machines

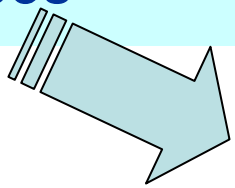
In the next ten years: Parallel Programming

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

Frameworks enable:

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

Encapsulates and seperates 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

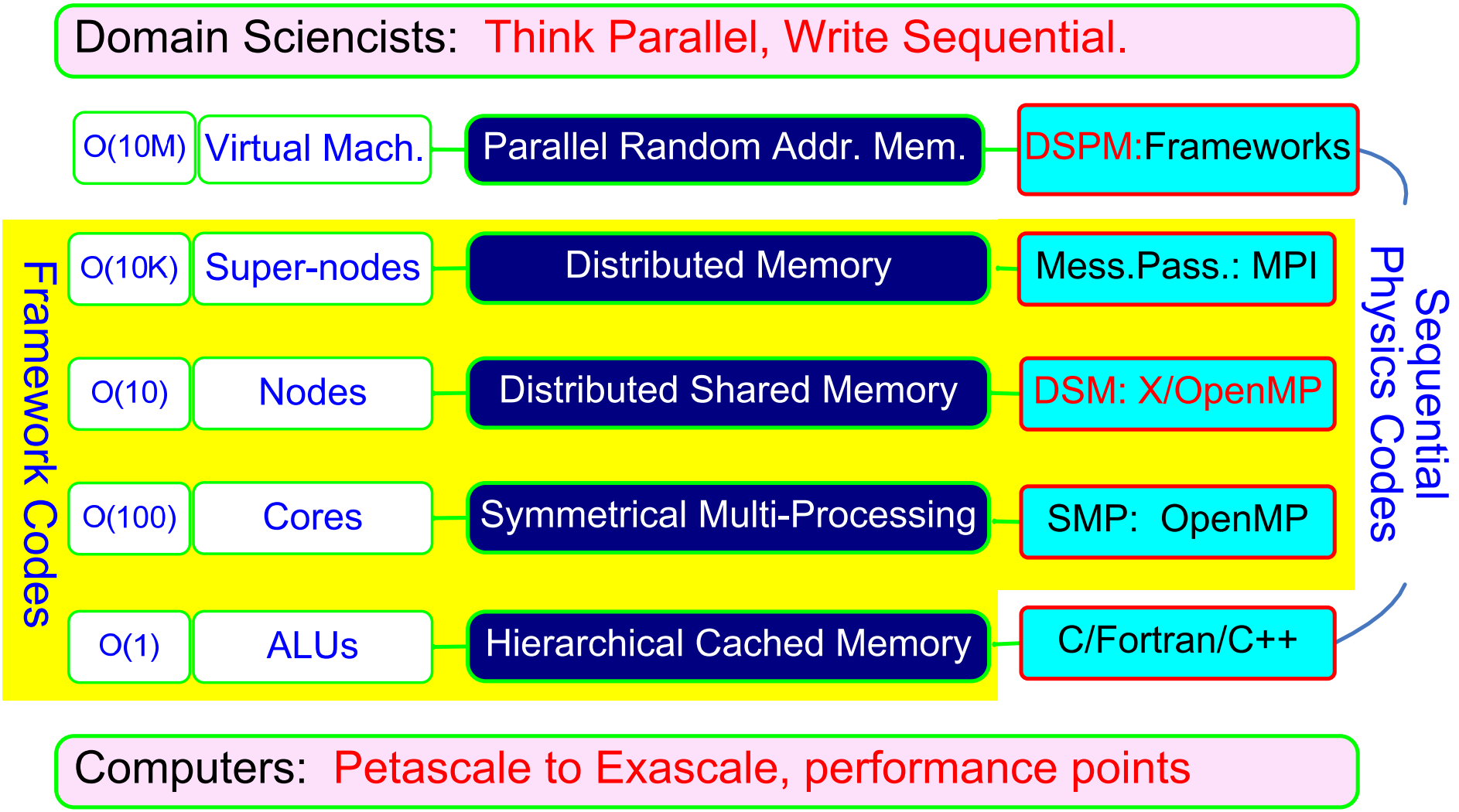
1. Significantly simplify or reuse the parallelism patterns using the emerging programming models;
2. The return on their rewrite efforts can be leveraged for multiple years even the machine is rapidly changing (e.g., 20 years ? ----- 20 or more years for MPI) ;
3. Once infrastructure in place, ratio 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



Framework-based DSPM is possible for scientific computing ?

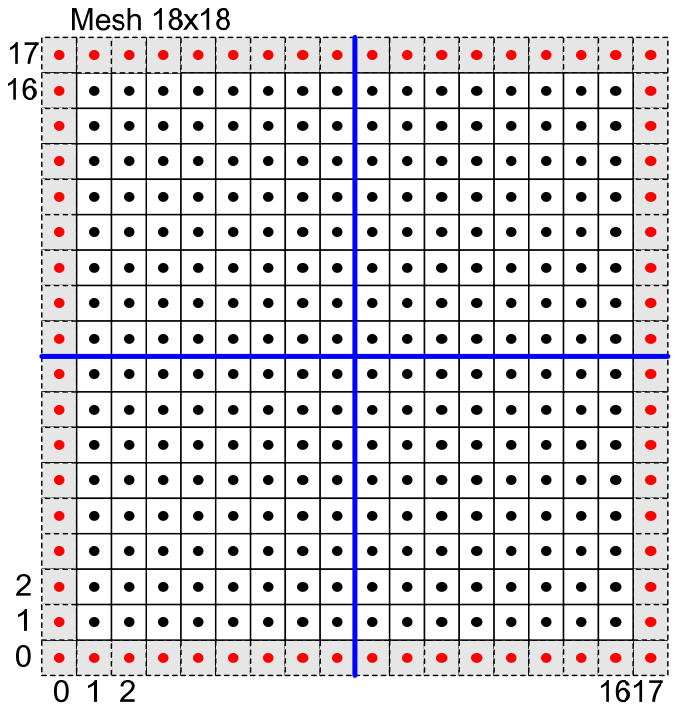
DDM Graph Based Patterns: Halo exchanges, Collectives

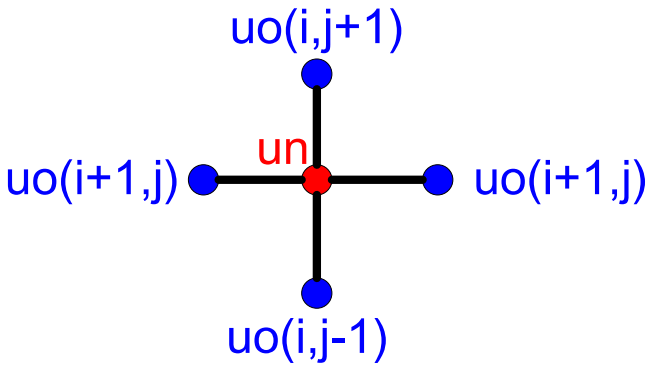
separate parallel programming from serial codes:

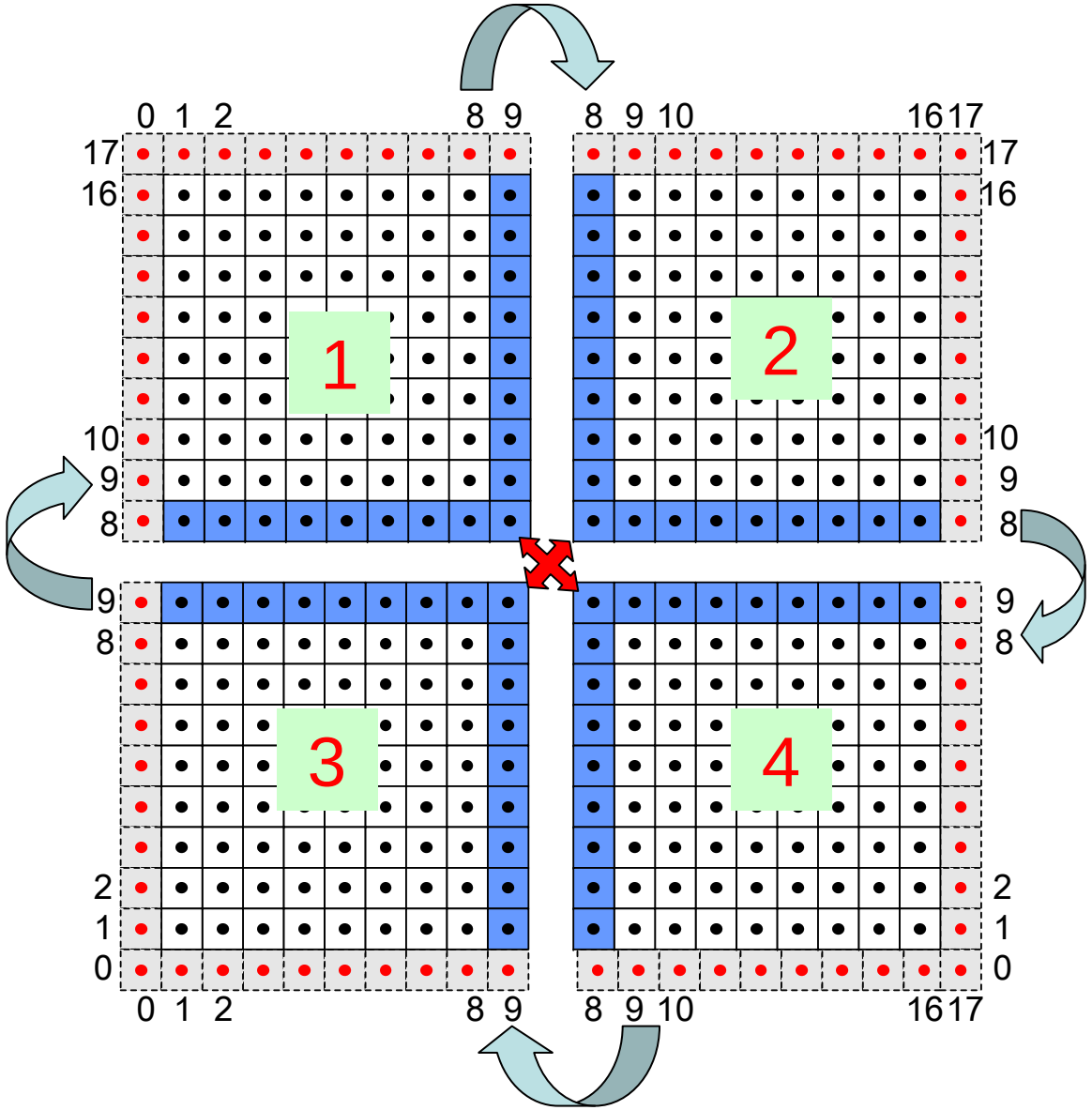
Computational Patterns

Digraph Based Patterns: data driven, dynamic tasks

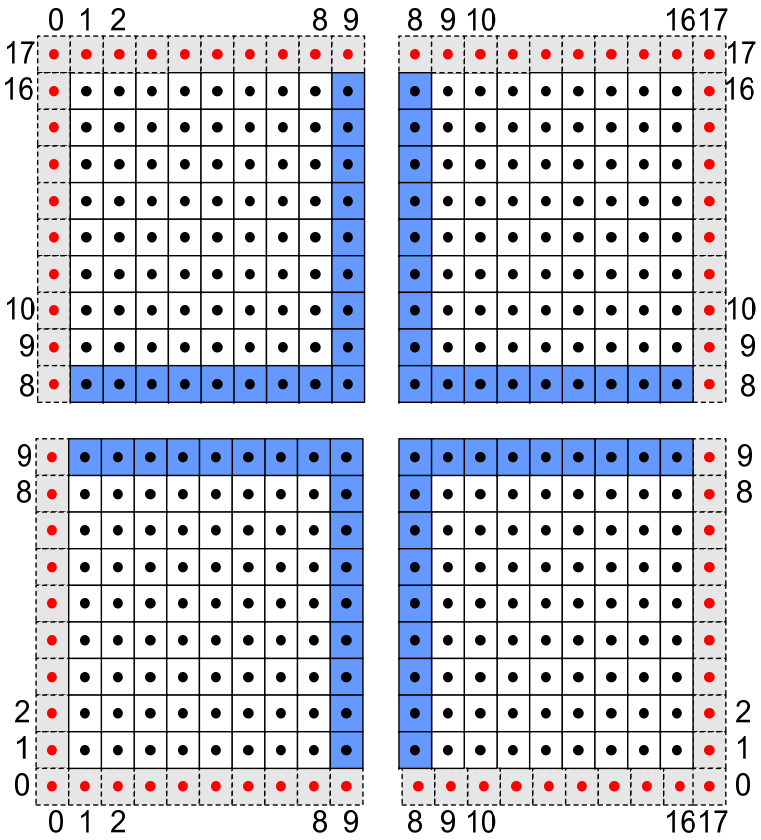
Halo Exchange Pattern: Separate Parallel Programming





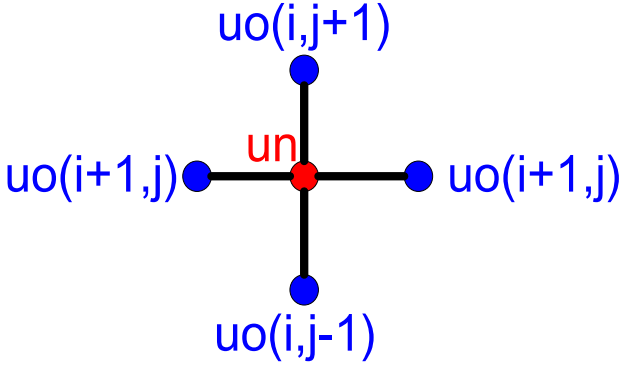


Halo Exchange Pattern: Separate Parallel Programming



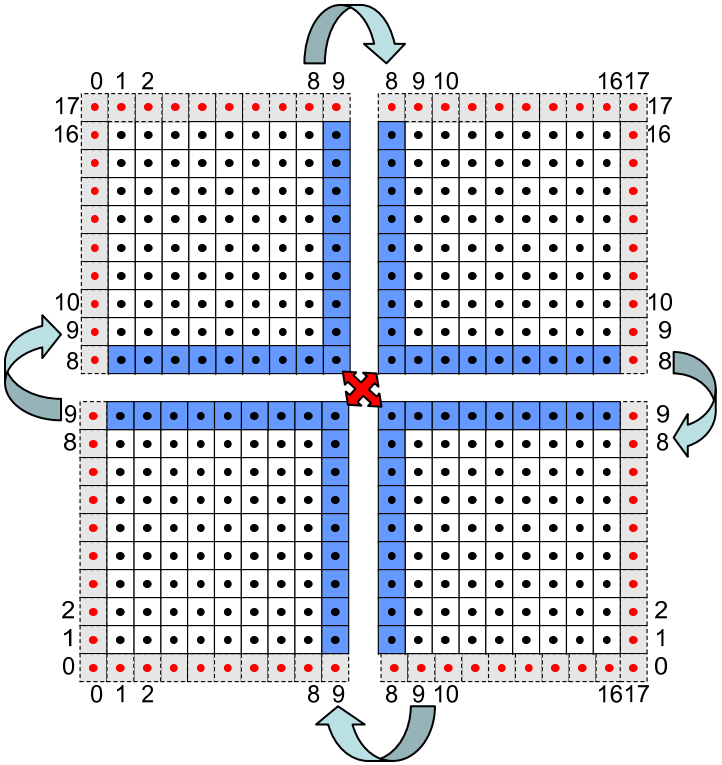

```
ist(1)=ist(3)=1, iend(1)=iend(3)=8  ;
jst(1)=jst(2)=9, jend(1)=jend(2)=16;
ist(2)=ist(4)=9, iend(2)=iend(4)=8  ;
jst(3)=jst(4)=1, jend(3)=jend(4)=16.
```

```
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



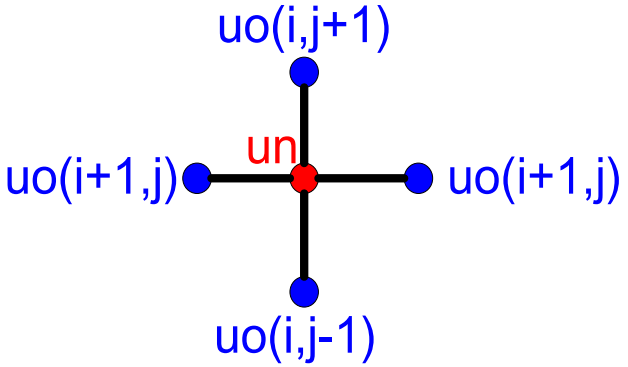
Do MPI message passing

fill ghost cells for uo ;

DO b= 1, 4 in parallel

DO i = ist(b), iend(b)

DO j = jst(b), jend(b)



ENDDO for i,j,b

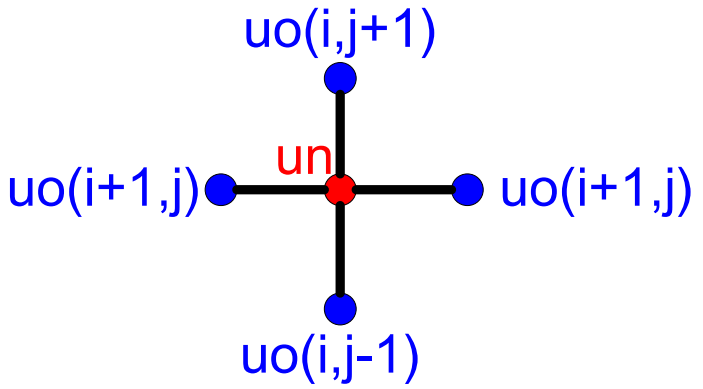
Halo Exchange Pattern: Separate Parallel Programming

DO MPI message passing
fill ghost cells for 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)

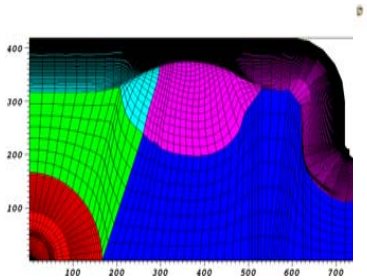


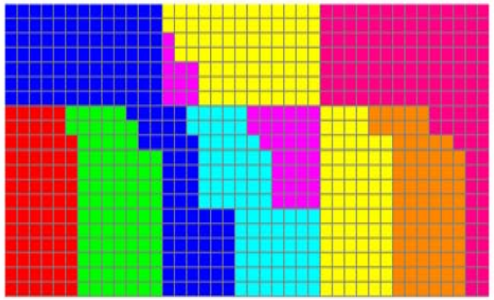
ENDDO for i,j

User Part

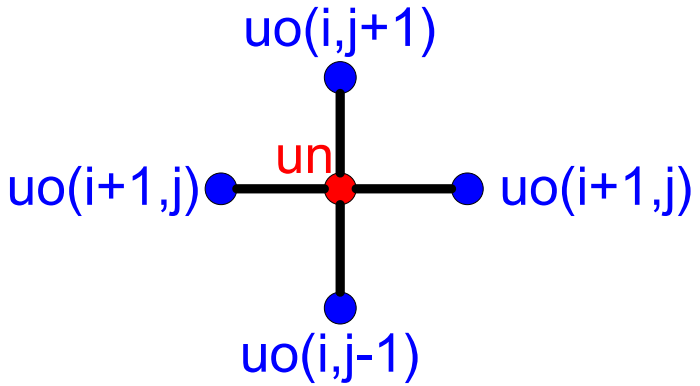
Halo Exchange Pattern: Separate Parallel Programming

```
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
```



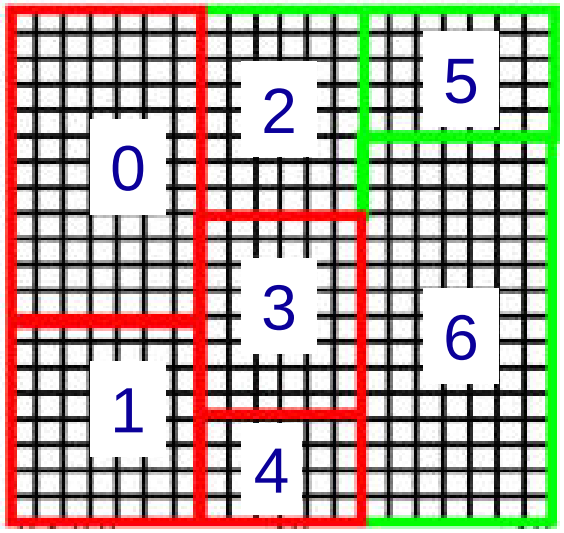


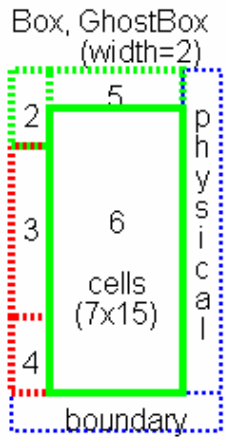
```
DO i = ist(b), iend(b)
DO j = jst(b), jend(b)
```




```
ENDDO for i,j
```

User Part

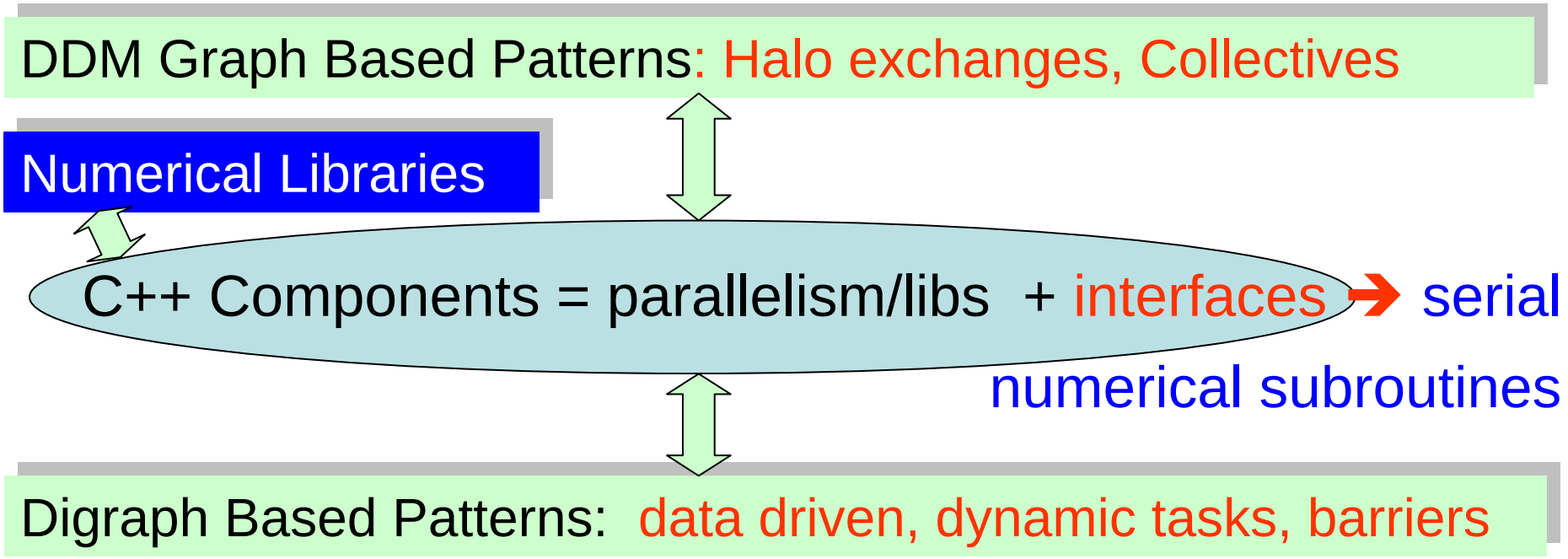




3. An instance: JASMIN framework

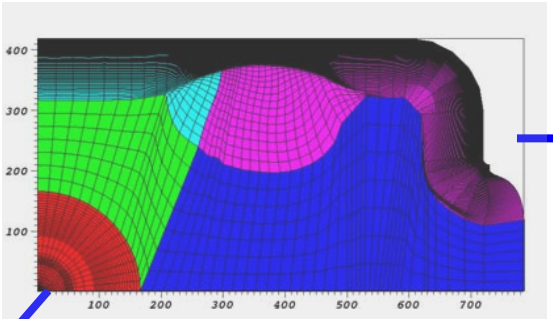
------ Think Parallel, Write Sequential

JASMIN : Parallel Patterns + Libarary ➔ DSPM



3.1 JASMIN

Structured Grid



J parallel
A daptive
S tructured M esh
IN frastructure

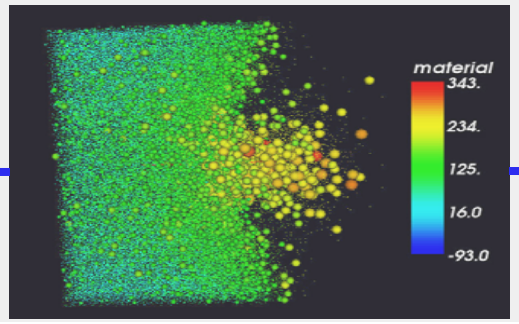
JASMIN

Inertial Confinement Fusion

Global Climate Modeling

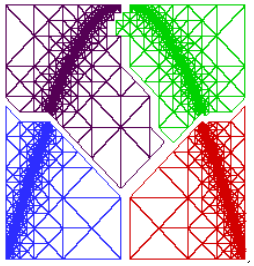
……

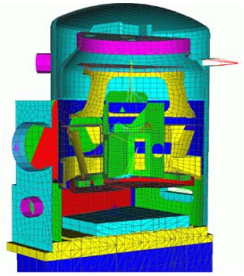
Particle Simulation



http:://www.iapcm.ac.cn/jasmin，

Unstructured Grid





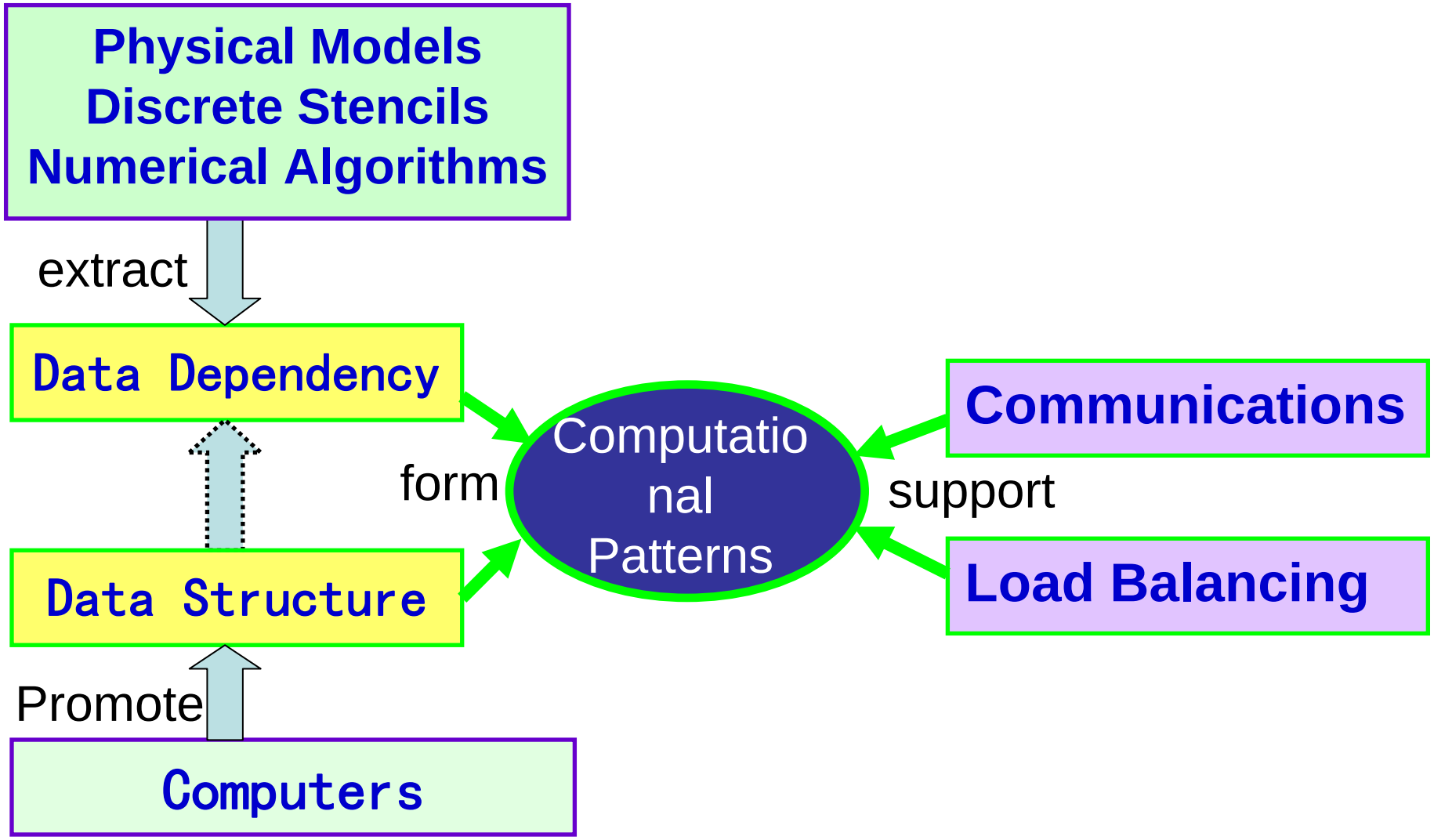
3.1 JASMIN

Motivations:

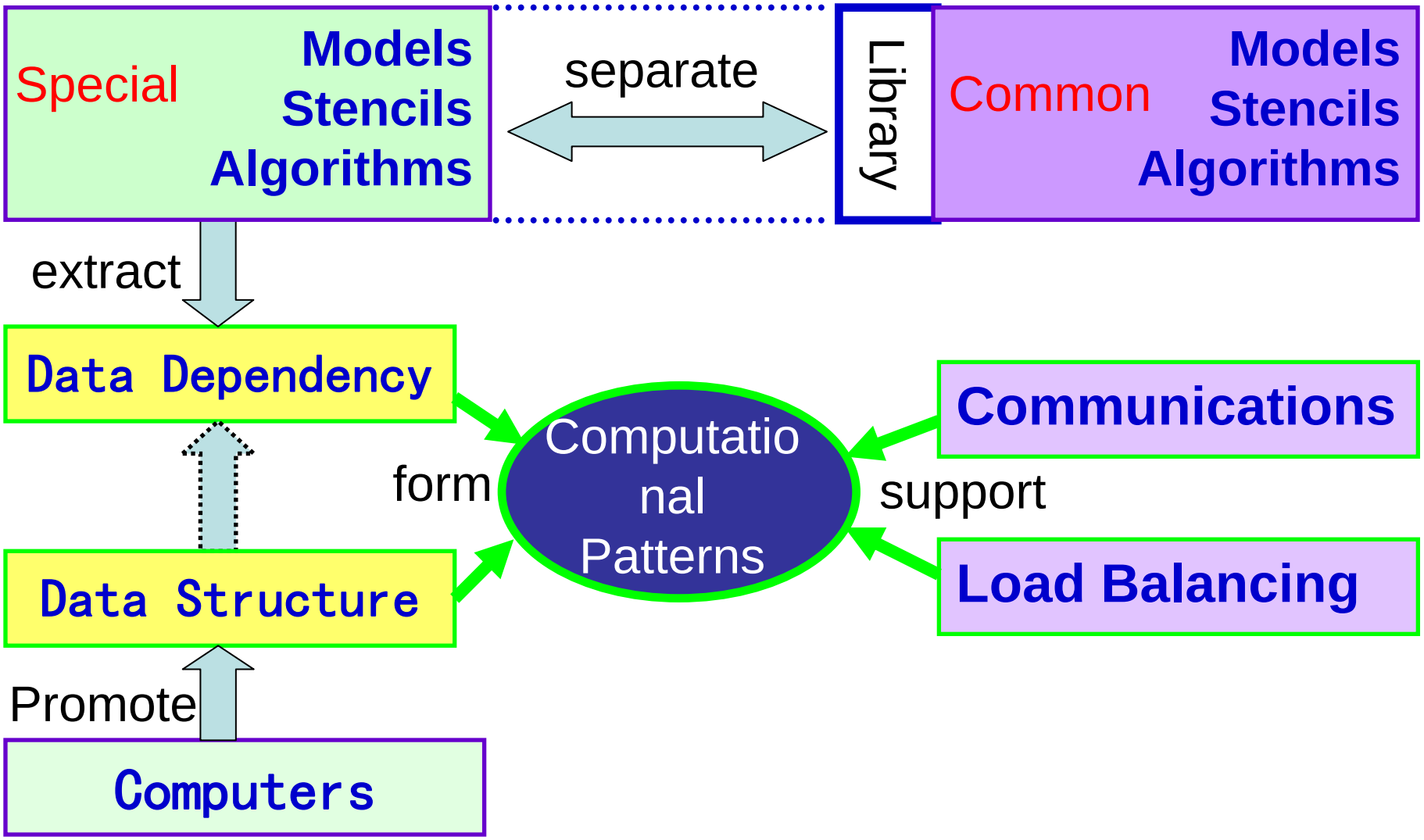
Supports the developments of parallel codes for large scale scientific computing on personal computers.

- Hides parallel programming using millons 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.

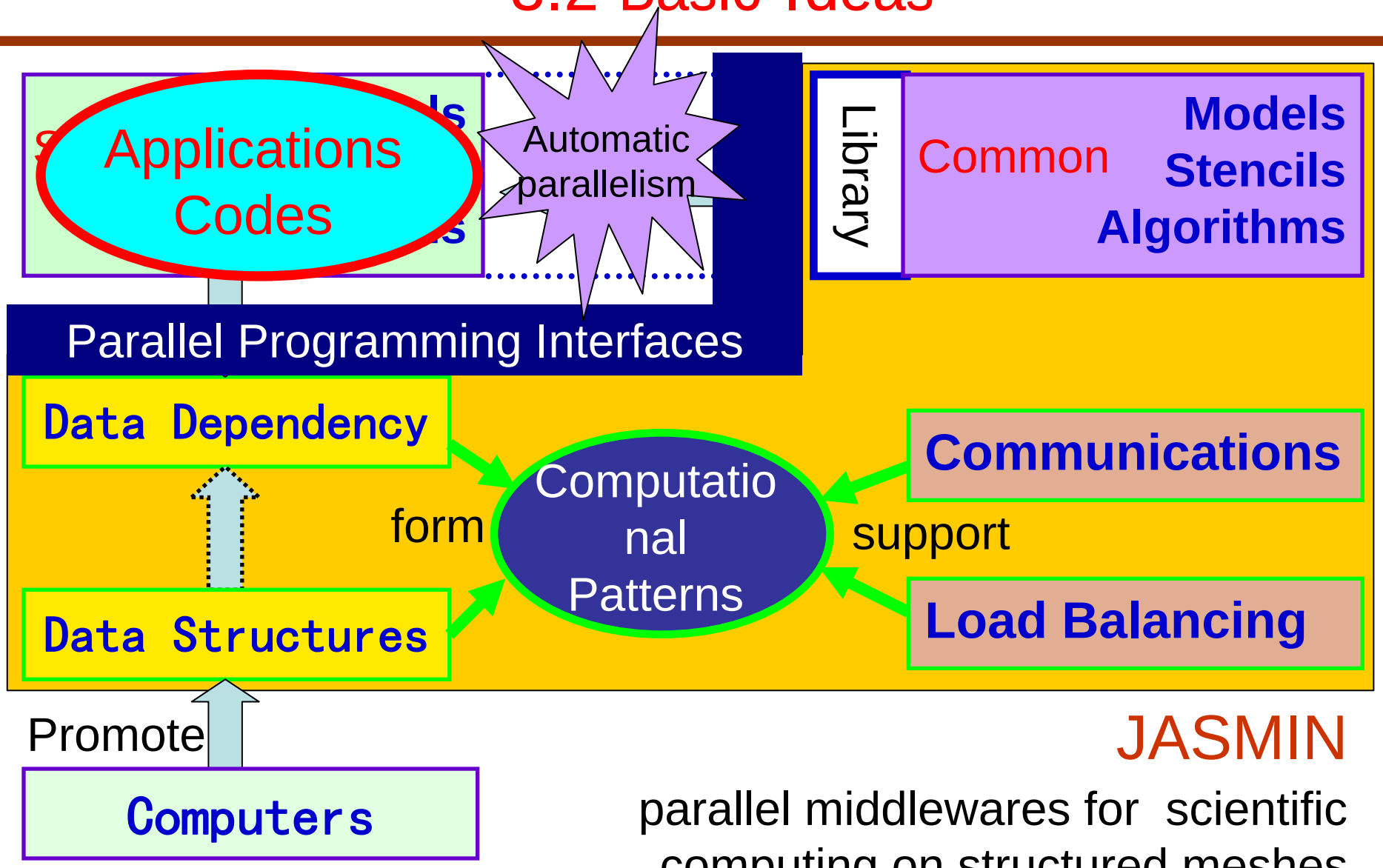
3.2 Basic Ideas



3.2 Basic Ideas



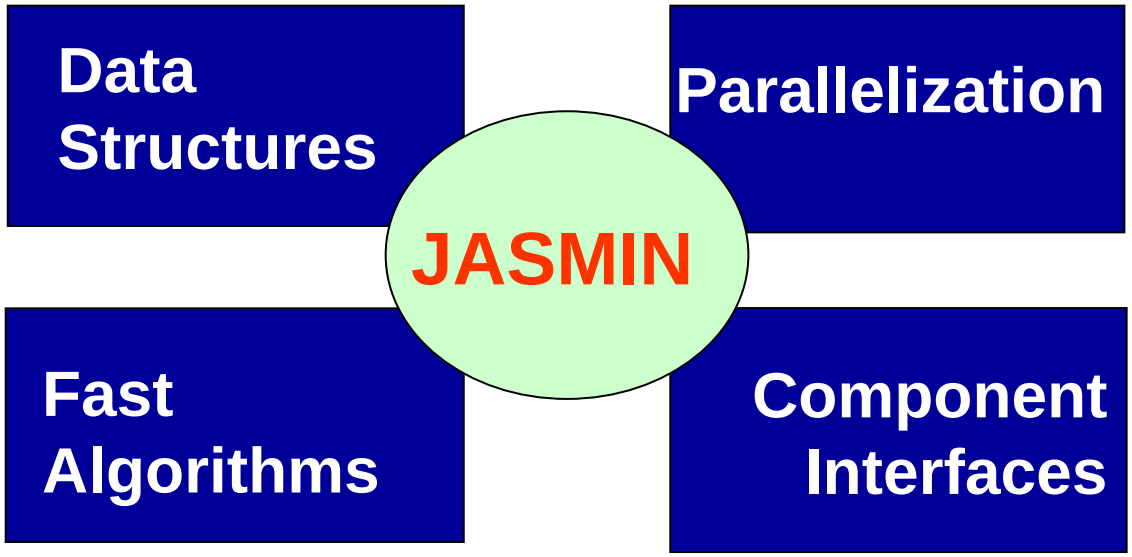
3.2 Basic Ideas



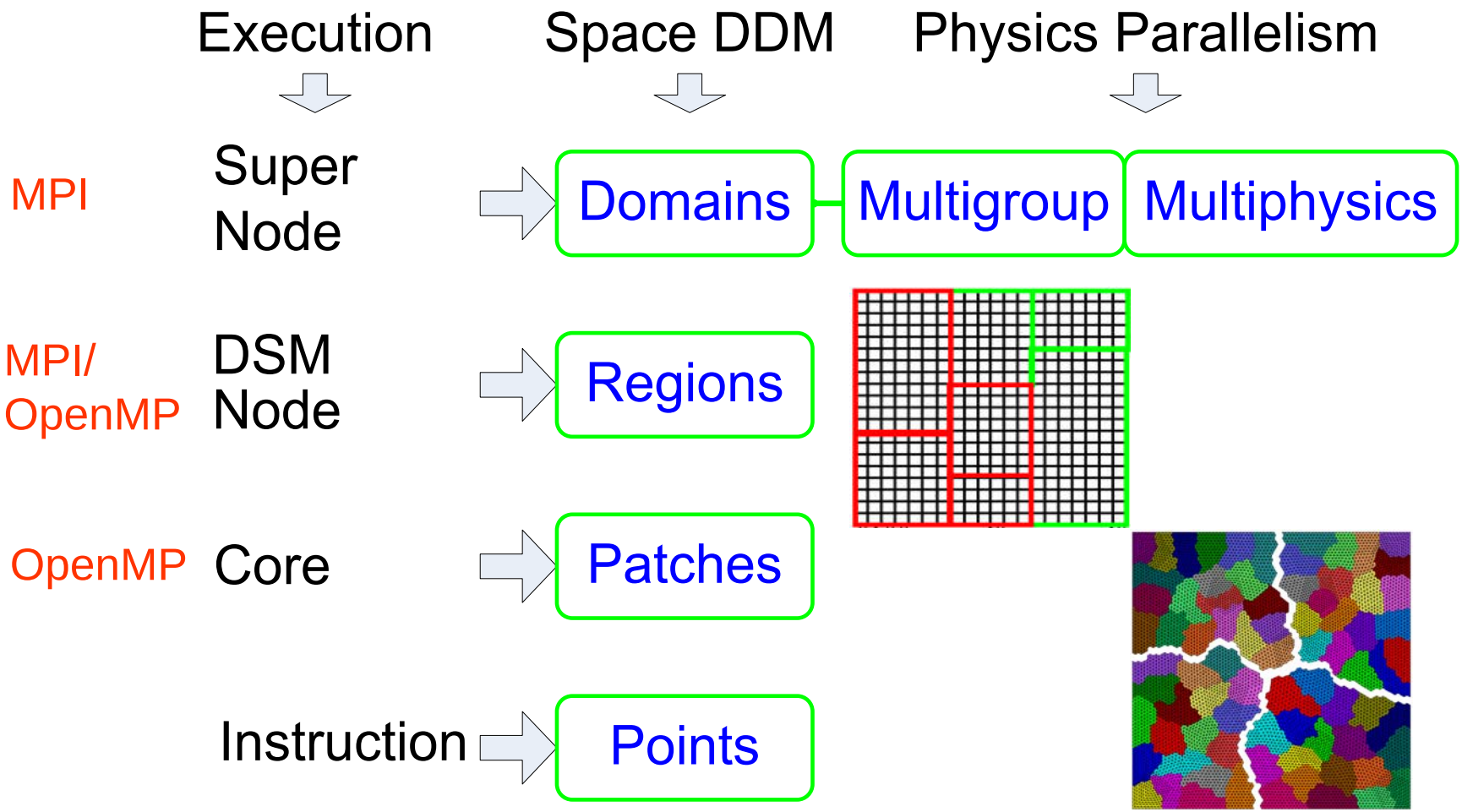
JASMIN

parallel middlewares for scientific computing on structured meshes

3.3 key factors



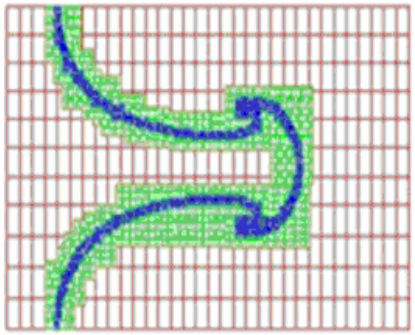
3.3.1 Data Structures



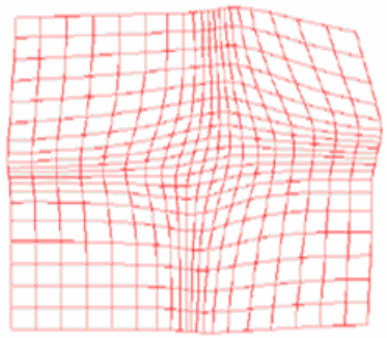
Neighboring Graph Based Patterns: Halo exchanges, Collectives

3.3.1 Data Structures

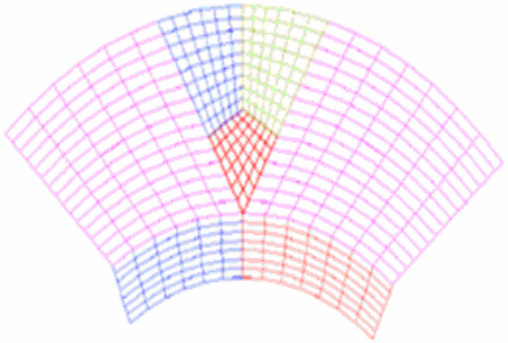
Mesh supported



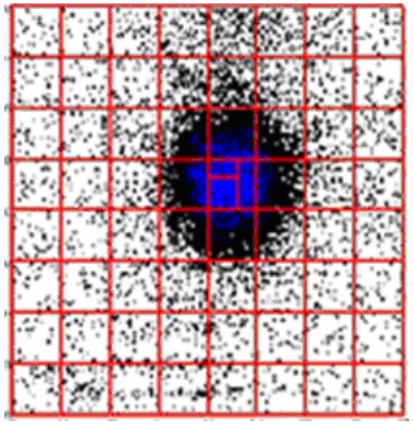
patch-based SAMR



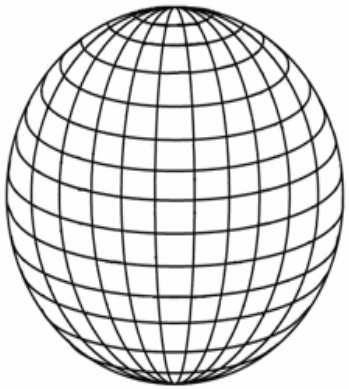
single block deforming mesh



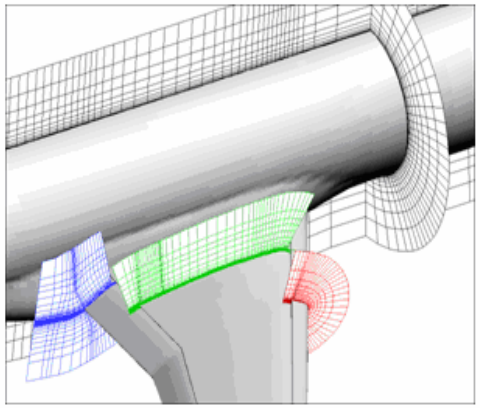
Multi-block deforming mesh



particles

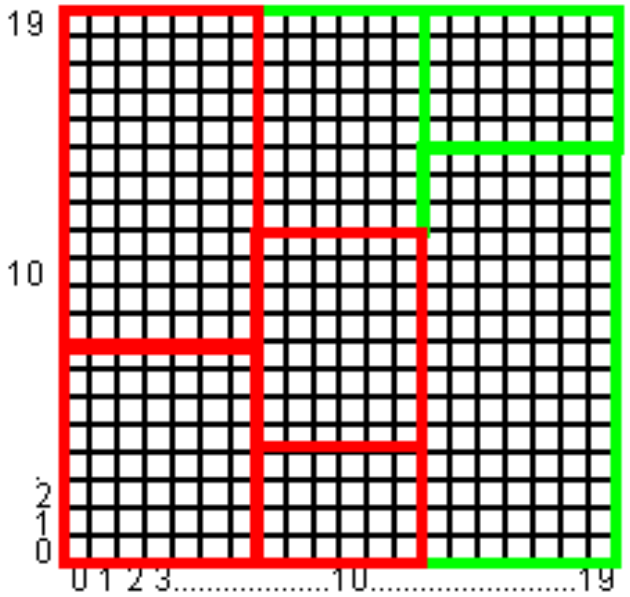


longitude-latitude mesh

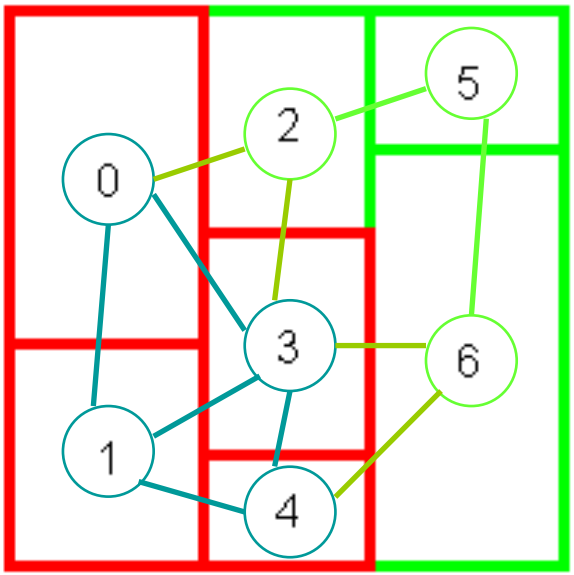


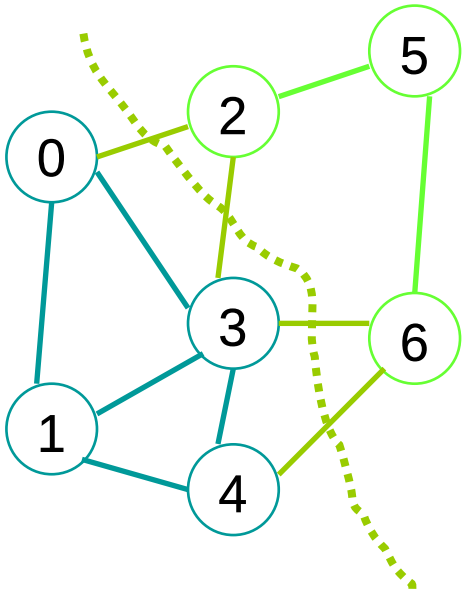
CFD multi-block mesh

3.3.2 Communications



Mesh cells : 20x20
Cell Indices : (i , j)

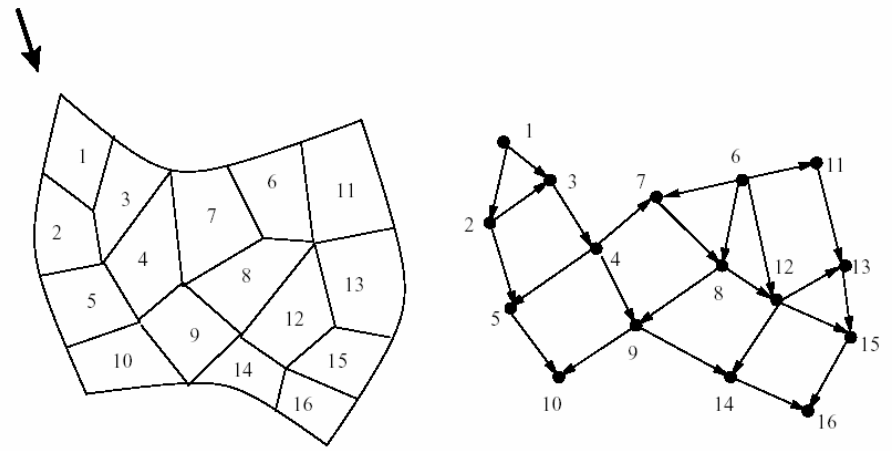




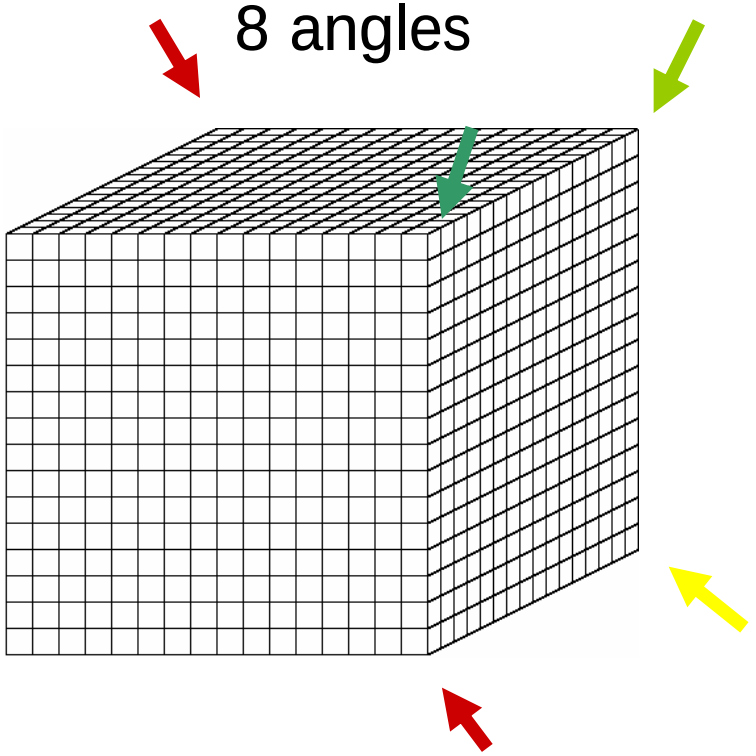
Distributed Undirected Graph
(2 processors)

3.3.2 Communications

distributed directed graph (digraph)



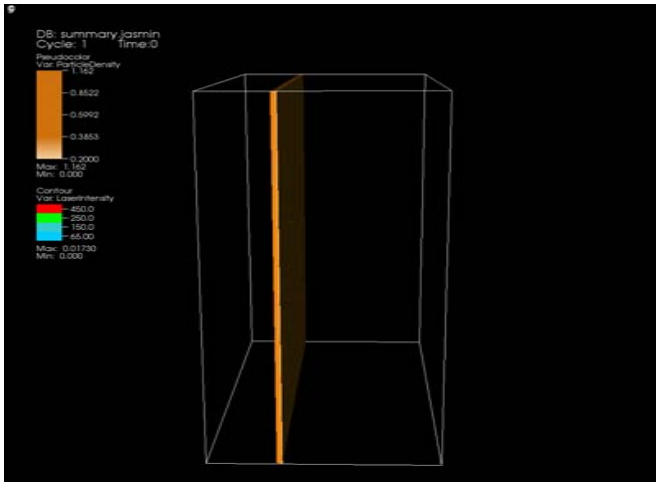
8 angles



Parallel Sweeping for Sn transport

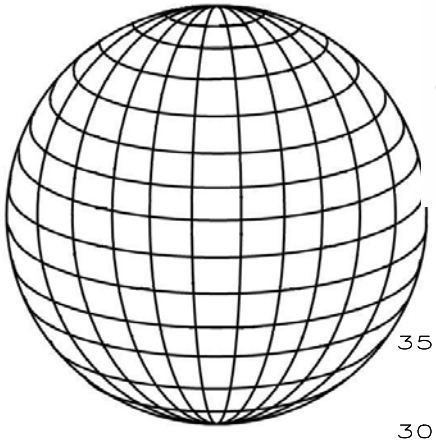
Digraph Based Patterns: data driven, dynamic tasks

3.3.3 dynamic load balancing

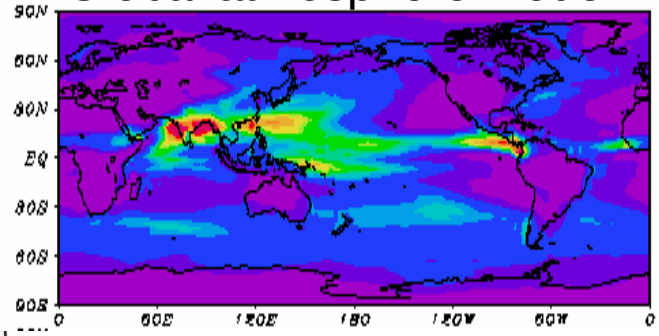


extreme load imbalance

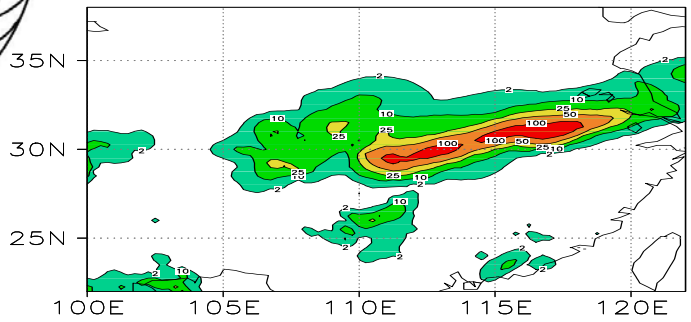
day, night

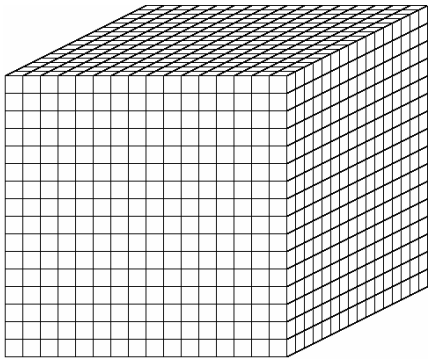


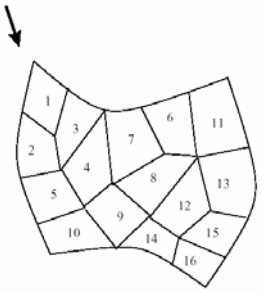
Global atmosphere model

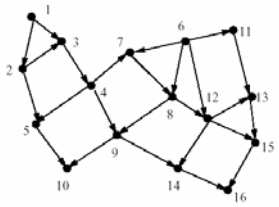


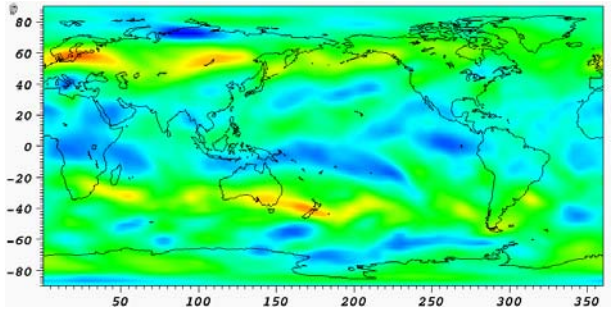
Regional rainstorm model





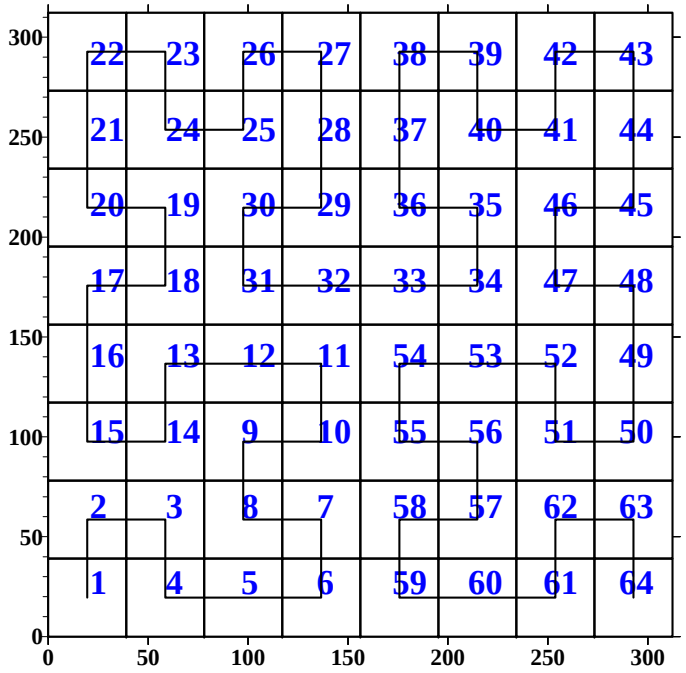






Radiation and neutron transport

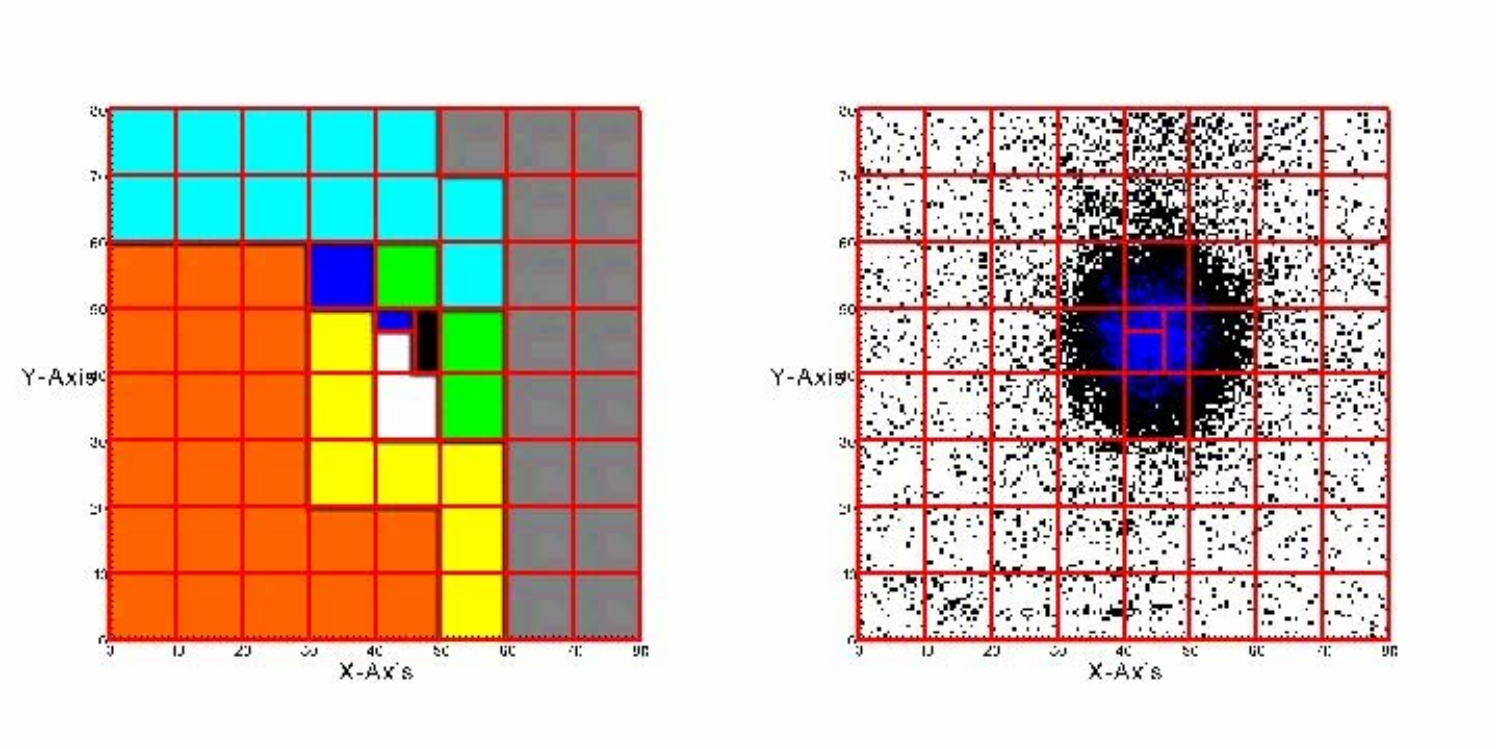
3.3.3 dynamic load balancing



space filling curves : 2D,3D->1D

MAW/Cycle methods : 1D balancing

3.3.3 dynamic load balancing



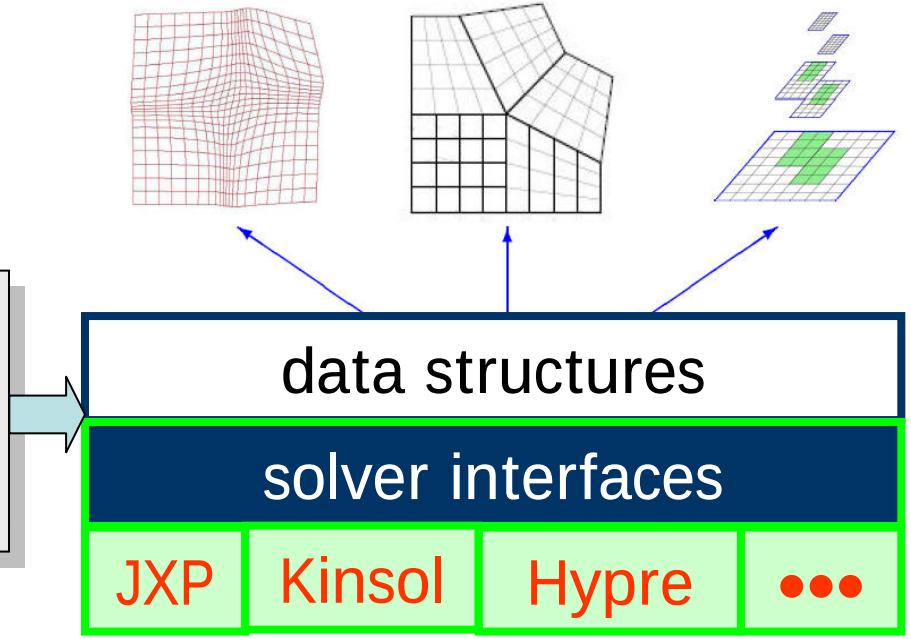
Redistributes particles among 8 Processes

3.3.4 Solvers for linear systems

Radiation hydrodynamics:

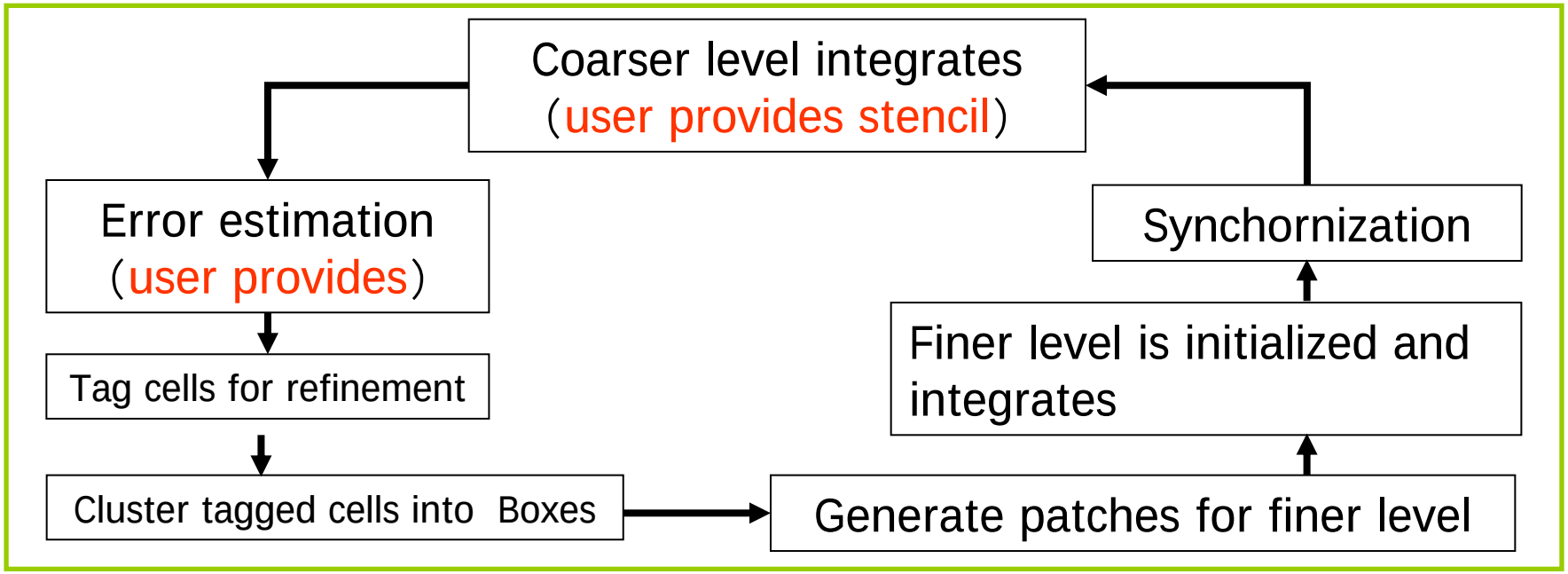
Usual algorithms : $O(N^{1.5})$

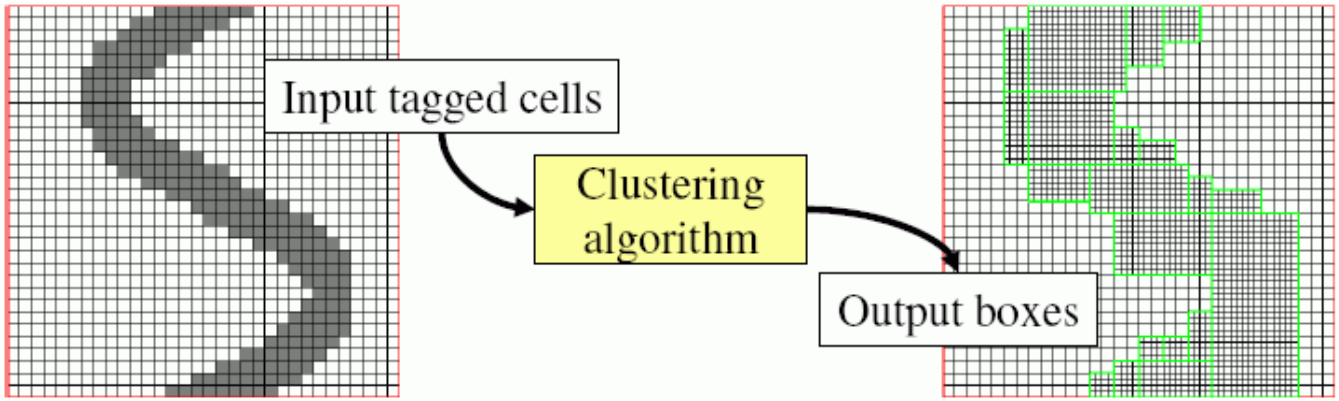
PAMG+DDM : $O(N\log N)$



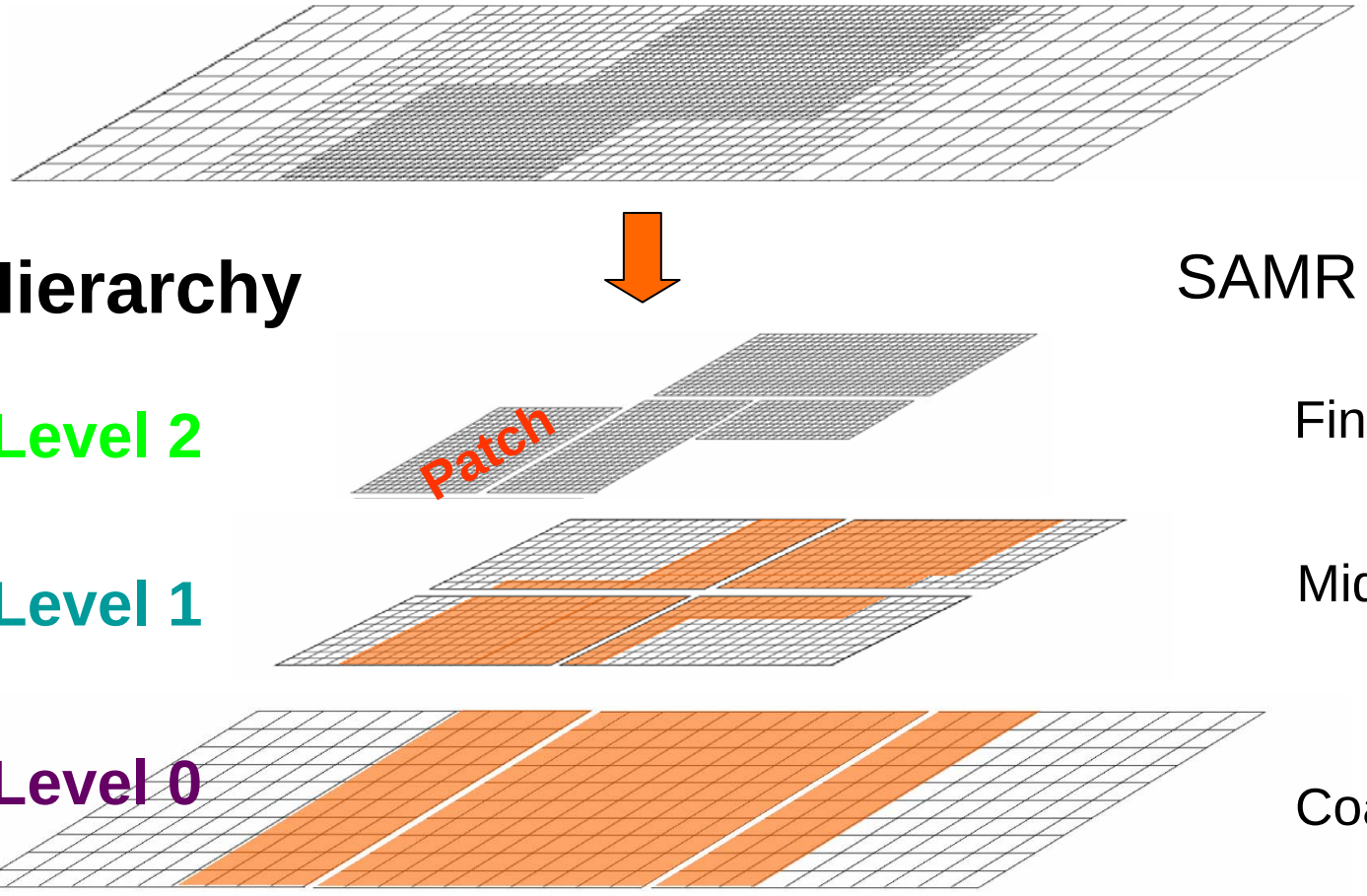
3.3.5 SAMR

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

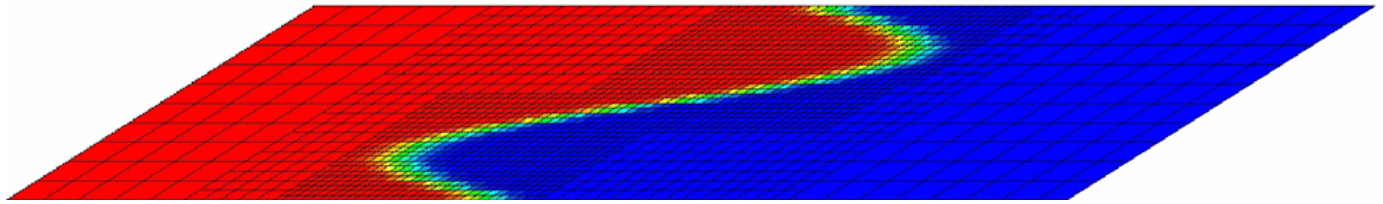




3.3.5 SAMR：data structure



3.3.5 SAMR：data structure



PatchHierarchy

SAMR

PatchLevel 2

Finest level

PatchLevel 1

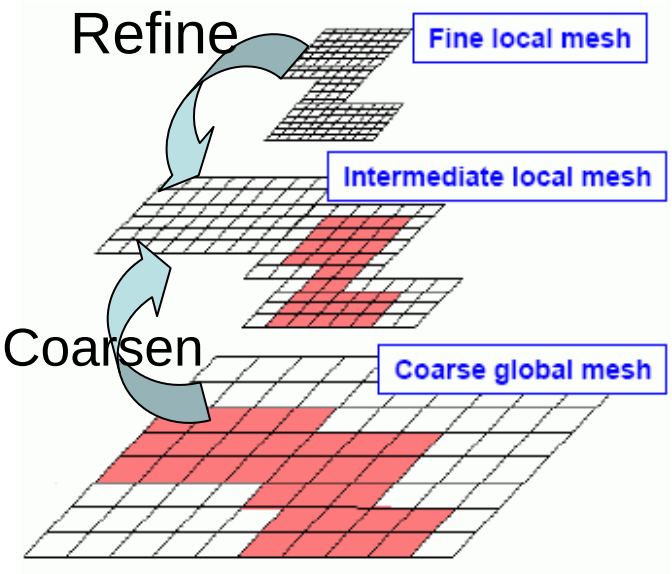
Middle level

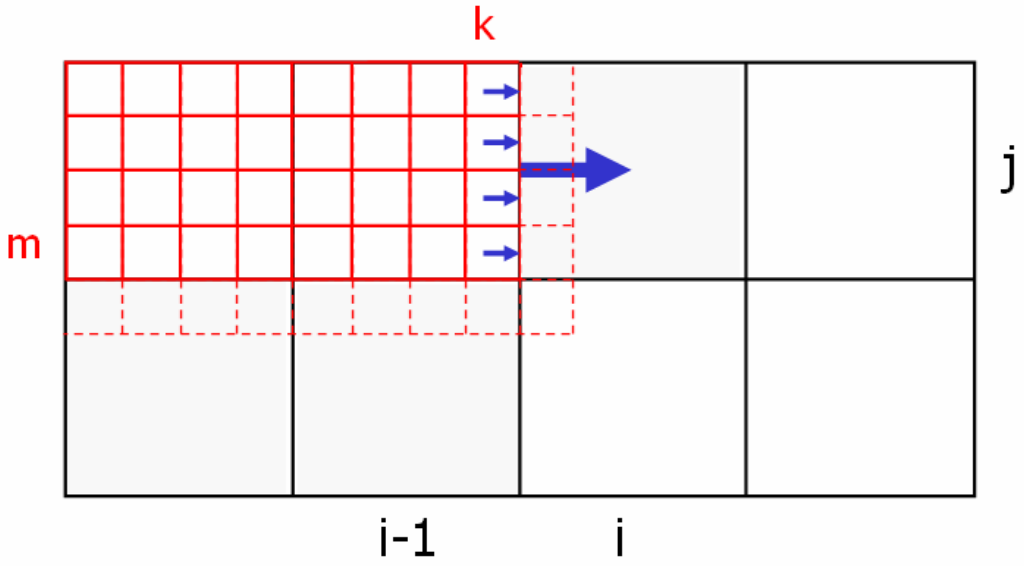
PatchLevel 0

Coarsest level

PatchHierarchy – PatchLevel – Patch (PatchData)

3.3.5 SAMR：Communication



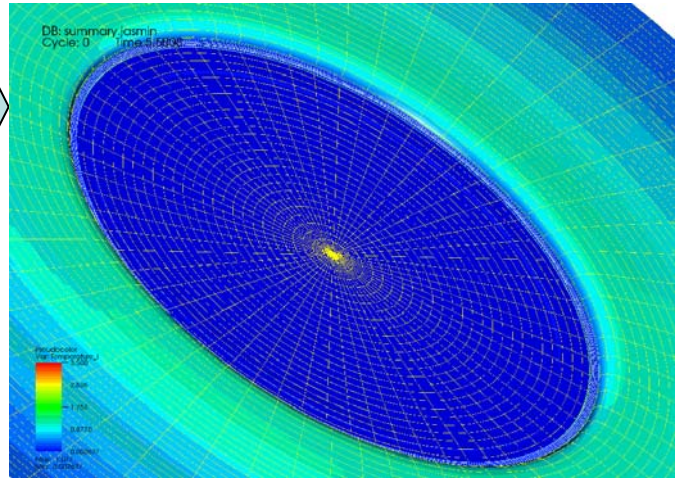


SAMR: using flux conservation interpolation schemes of high order precision

$$F^{coarse}_{i-\frac{1}{2},j} = \frac{1}{r^2}\sum_{q=0}^{r-1}\sum_{p=0}^{r-1} F^{fine}_{k+\frac{1}{2},m+p}\left(t+q\Delta t^{fine}\right)$$

3.3.5 SAMR

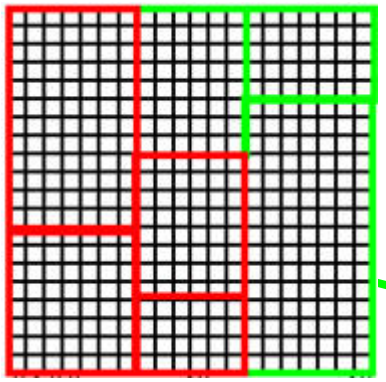
ICF 2-D radiation hydrodynamics simulation using LARED-S using three levels of SAMR meshes.



Speedup = 160.

Resolution of coarsest level	# levels	# cells	# steps	# CPU cores	Time (hours)
Resolution of finest level: 10240x2048. Physical time =1.324ns					
10240x2048	1	2097	1024 cores require 20 days		
640x128	3	38 ~ 124	28700	64	**24.6**

3.3.6 User Interfaces



Main program

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

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

User time integrator for patch level

User implements

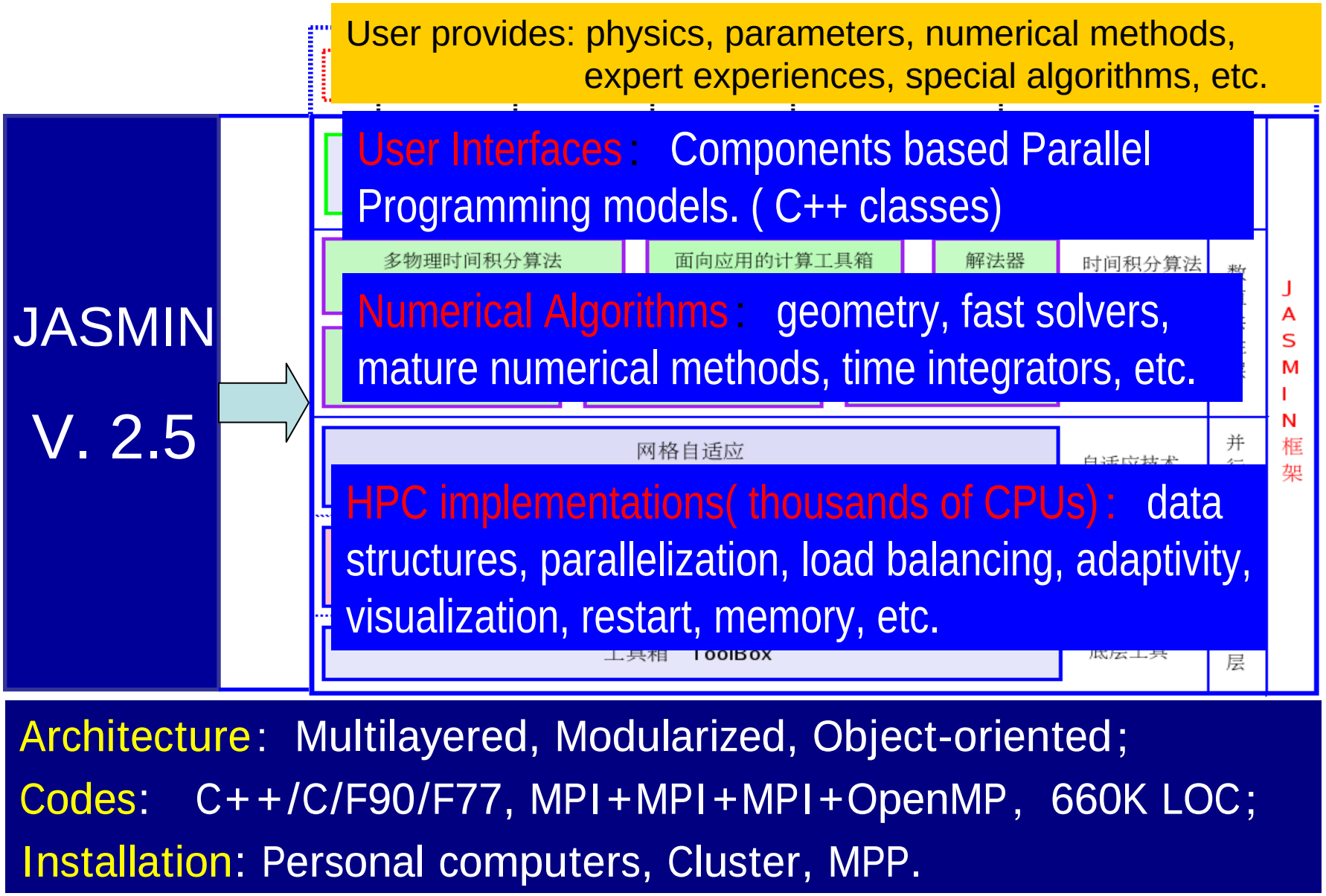
User numerical
subroutines for patches

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

JASMIN
supports

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

3.4 Current Version



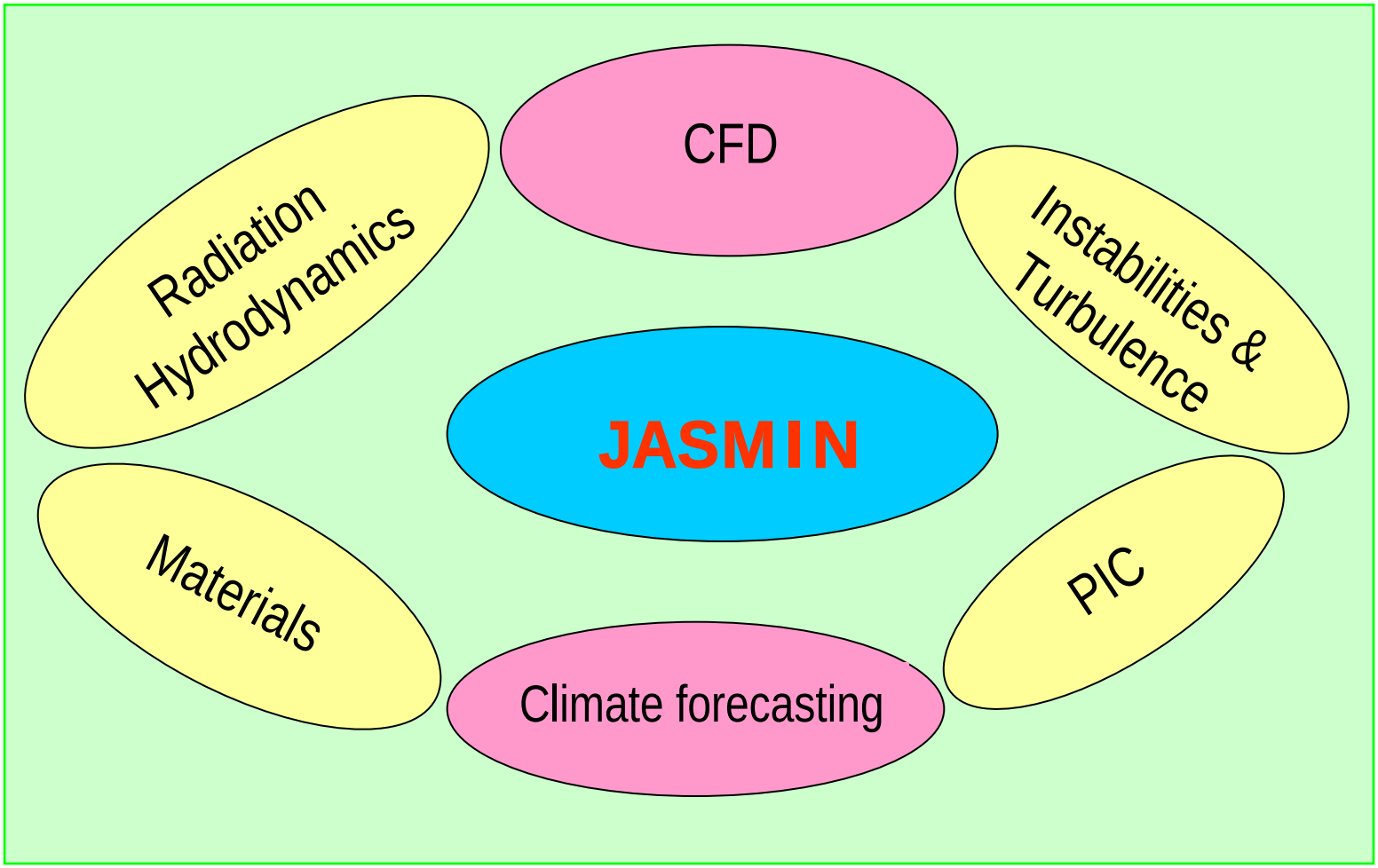
Architecture: Multilayered, Modularized, Object-oriented;

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

Installation: Personal computers, Cluster, MPP.

3.4 Current Version

Applications currently



25 codes

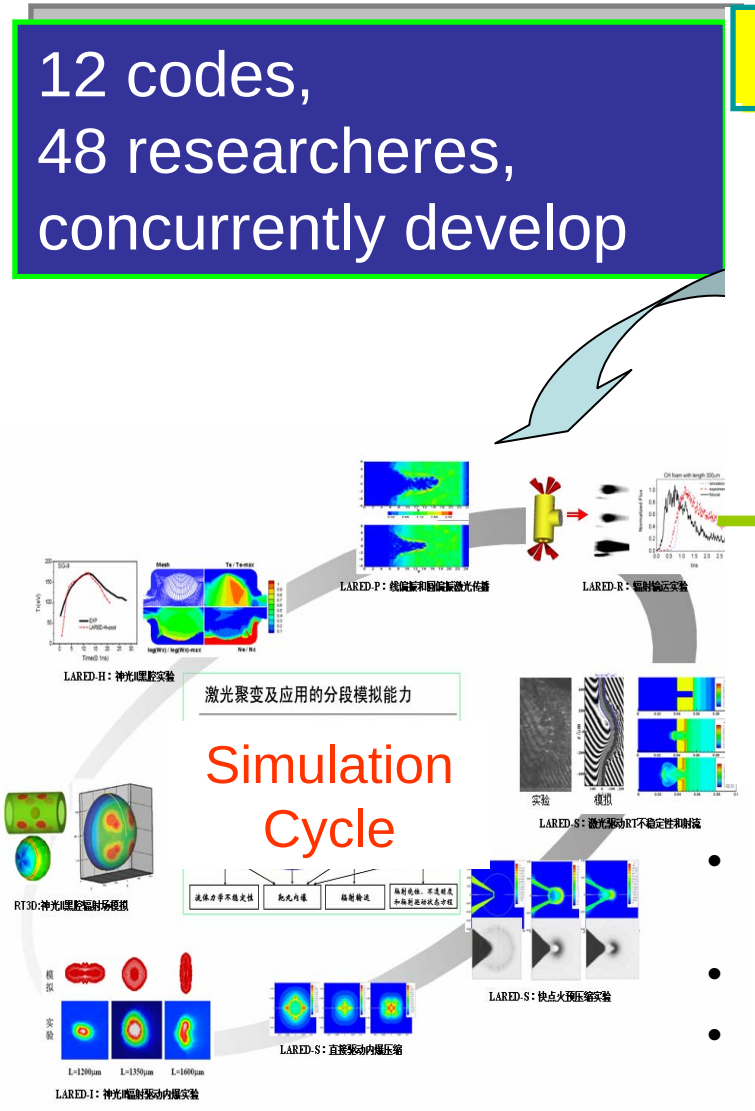
3.5 Some Applications on PetaFlops System TianHe-1A

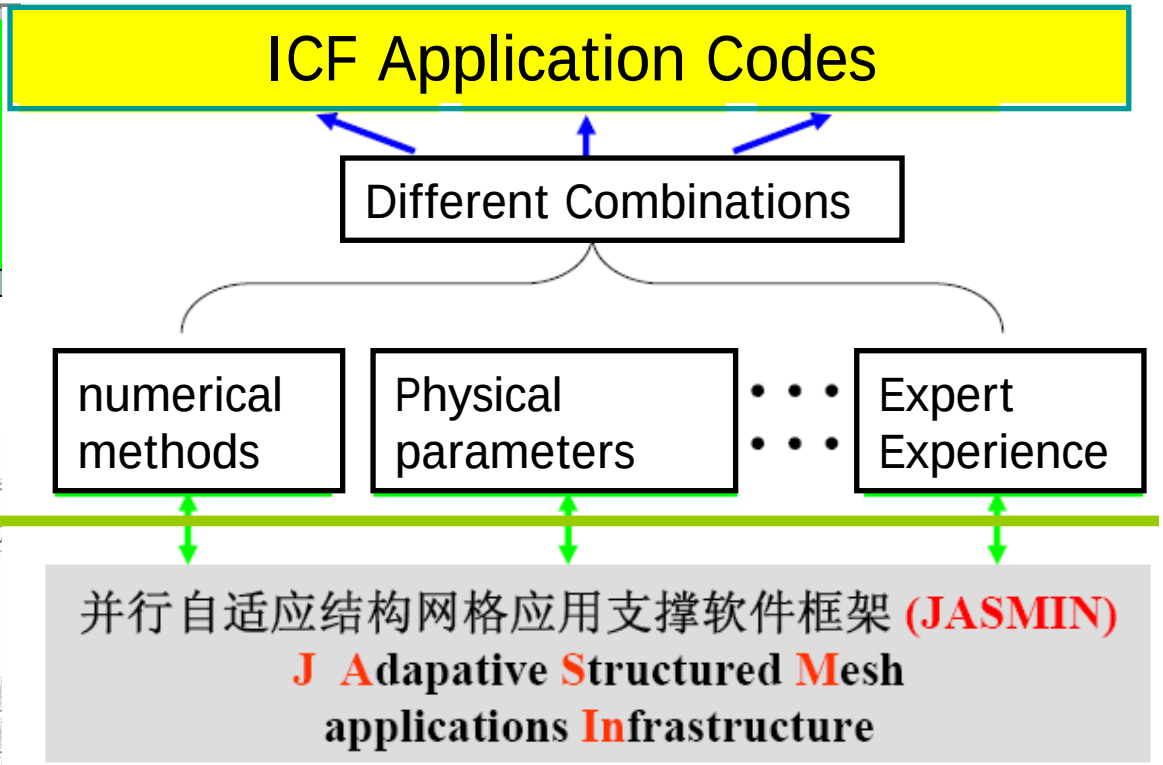
Codes	# CPU cores	Codes	# CPU cores
LARED-S	32,768	RH2D	1,024
LARED-P	72,000	HIME3D	3,600
LAP3D	16,384	PDD3D	4,096
MEPH3D	38,400	LARED-R	512
MD3D	80,000	LARED Integration	128
		RT3D	1,000
JMES-FDTD	60,000	NEPTUNE	1,024

Simulation times : several hours to tens of hours.

Applications-1：Inertial Confinement Fusion

12 codes,
48 researcheres,
concurrently develop





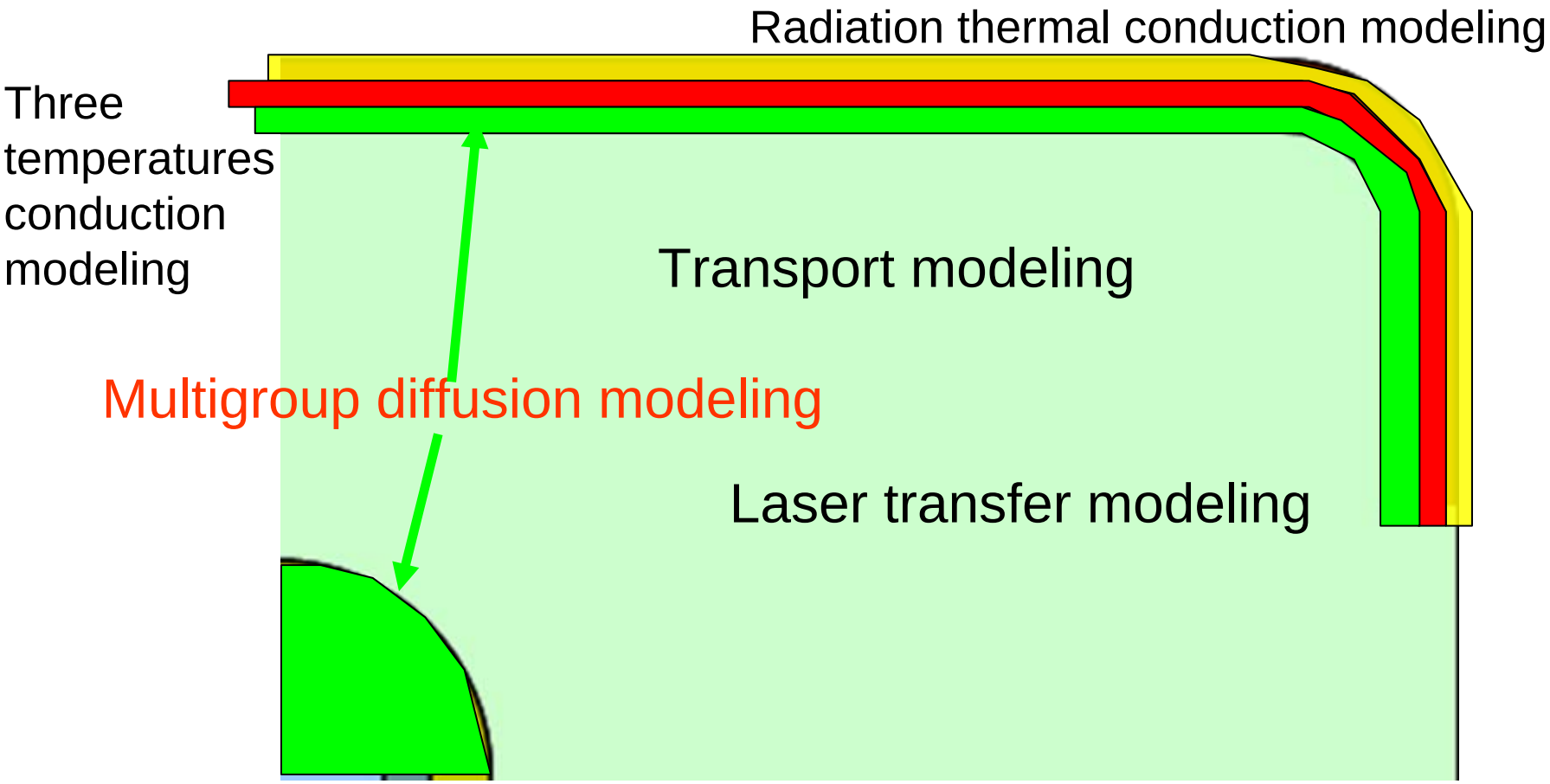
- Hides parallel computing and adaptive implementations using tens of thousands of CPU cores；
- Provides efficient data structures, algorithms and solvers;
- Support software engineering for code extensibility.

Applications-1：Inertial Confinement Fusion

Codes	Year 2004	Year 2010
LARED-H 2-D radiation hydrodynamics Lagrange code	serial	Parallel
	Single bolck	Multiblock
	Without capsule	NIF ignition target
LARED-R 2-D radiation tra	Serial	Parallel (2048 cores)
LARED-S 3-D radiation hydrodynamics Euler code	Single level	SAMR
	2-D: single group	Multi-group diffusion
	3-D: no radiation	3-D: radiaiton multigroup diffusion
LARED-P 3-D laser plasma interaction code	serial	Parallel (36000 cores) Terascale of particles

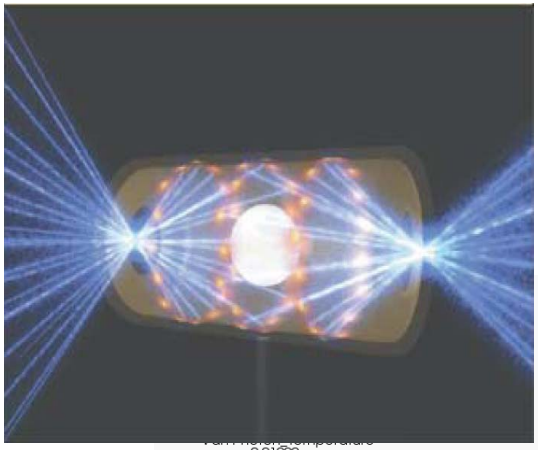
Scale up a factor of 1000

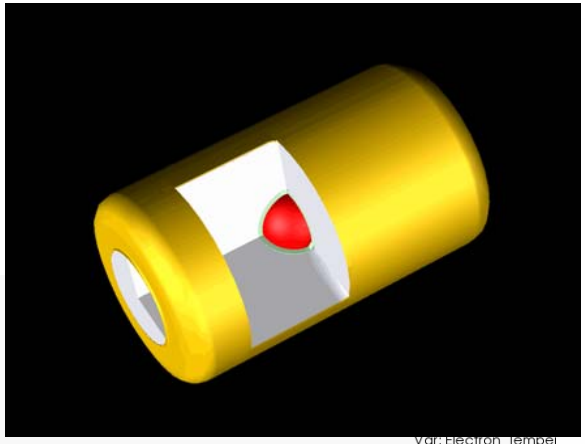
ICF-1: Integration simulations for ignition targets

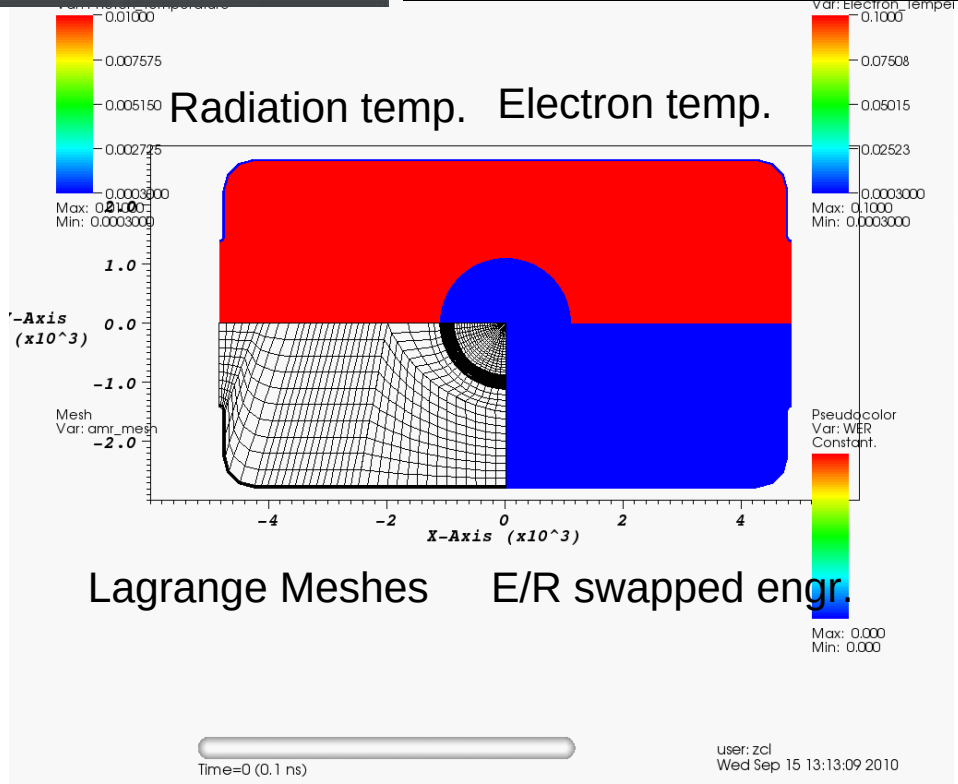


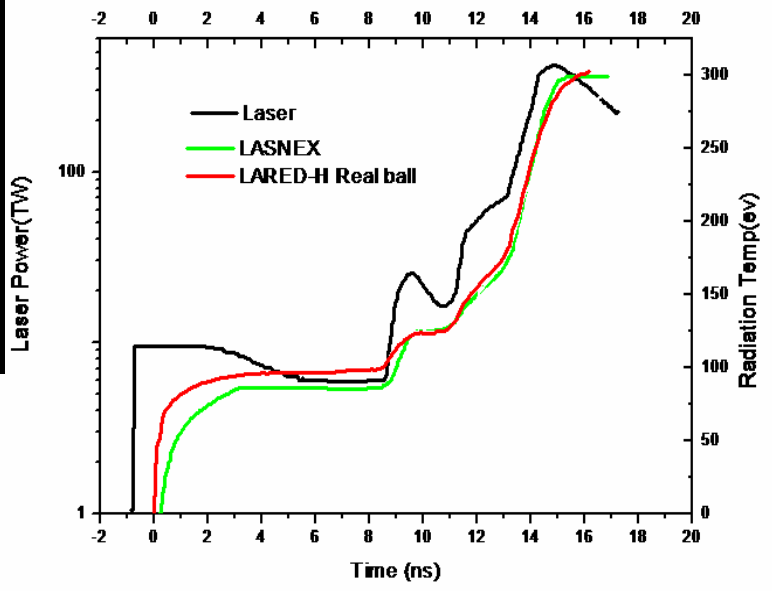
Multi-physical modeling for radiation hydrodynamcs simulations

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

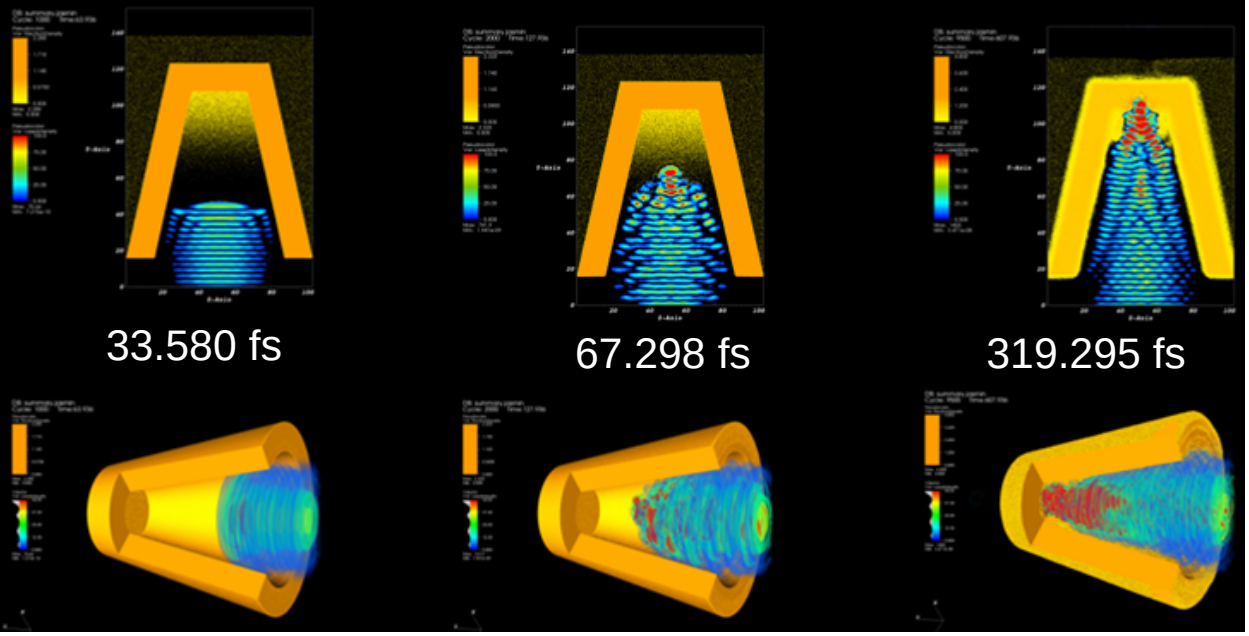
LARED-P: super-strong lasers transfer and focus over the plasma cone and generate high energy electrons.

#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



ICF-3: 3-D simulations for laser plasma hydrodynamics

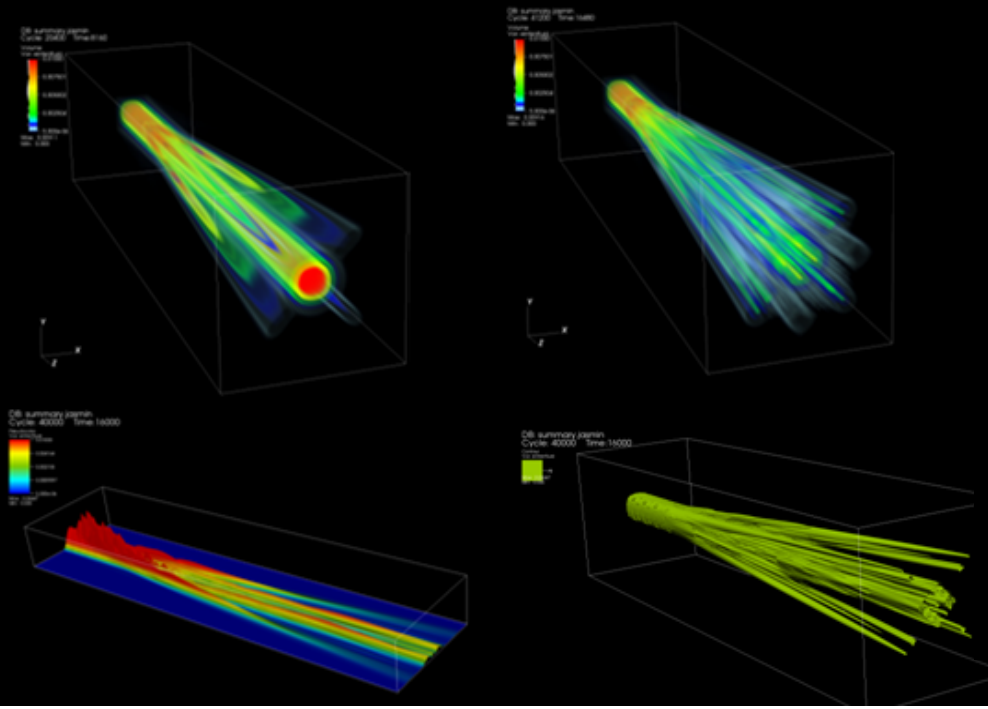
LAP3D: lasers filament and self-focus while transfer over a long distance in the lower density plasma environments.

#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

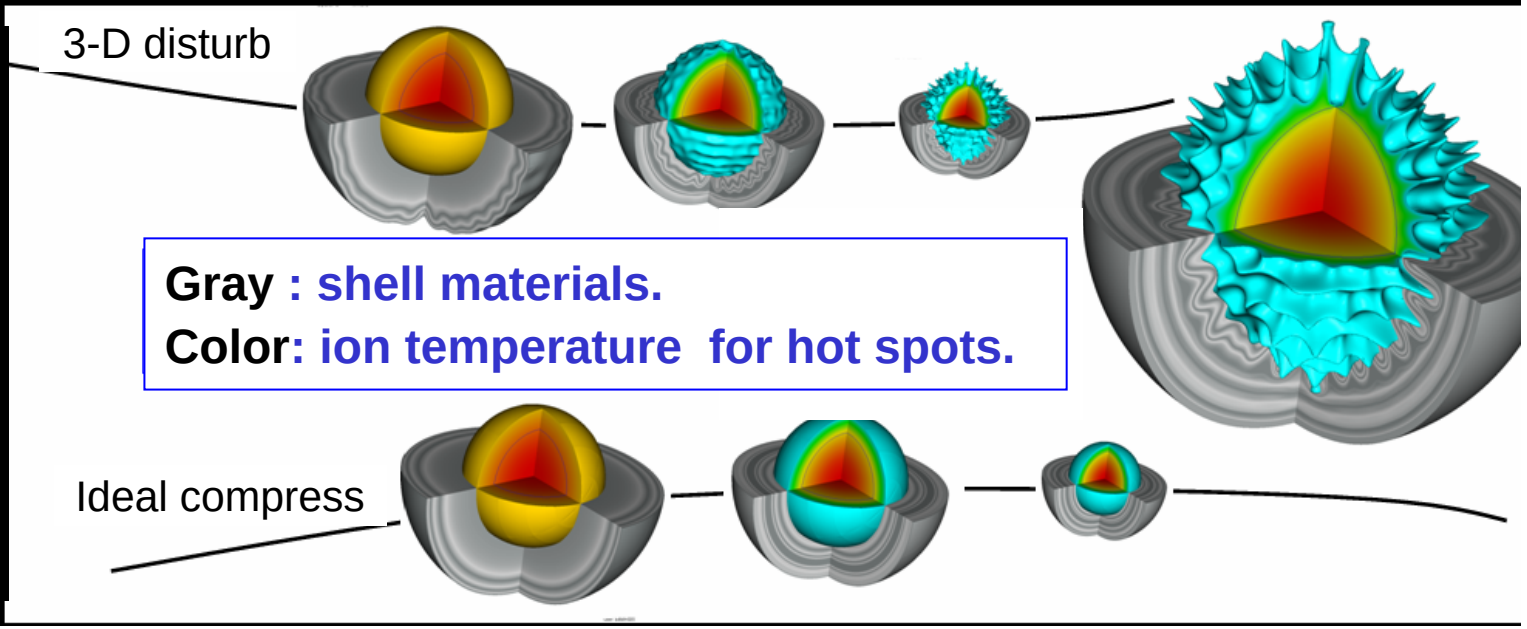
LARED-S: radiation hydrodynamics instabilities occur over the capsule interfaces while the capsule shell rapidly slows down.

#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



Applications-2： Material simulations

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

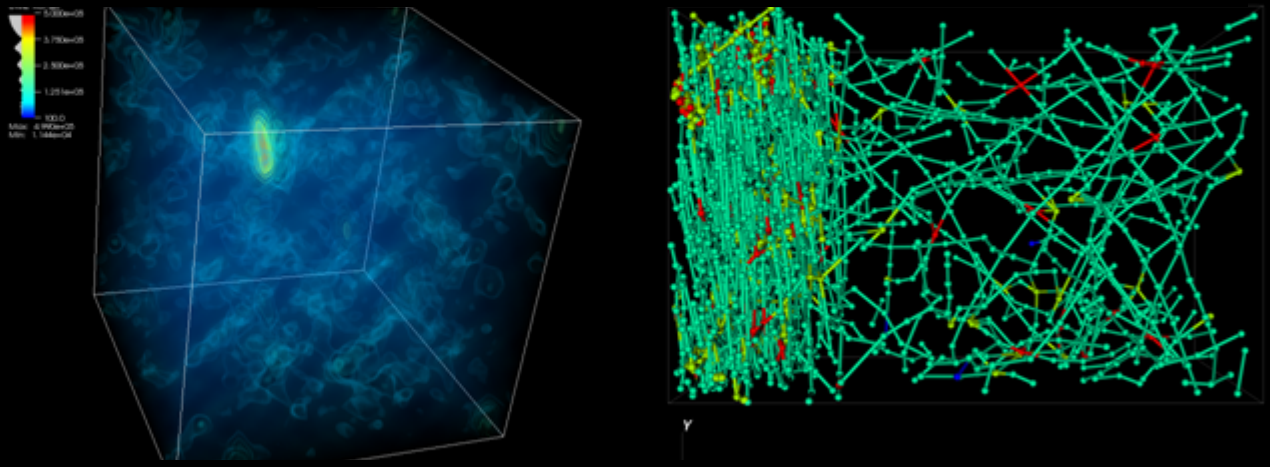
#dislocation: 3 M

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 mm^3, $r=10^7/s$).
Left: dislocation density; Right: local structures.

Applications-2： Material simulations

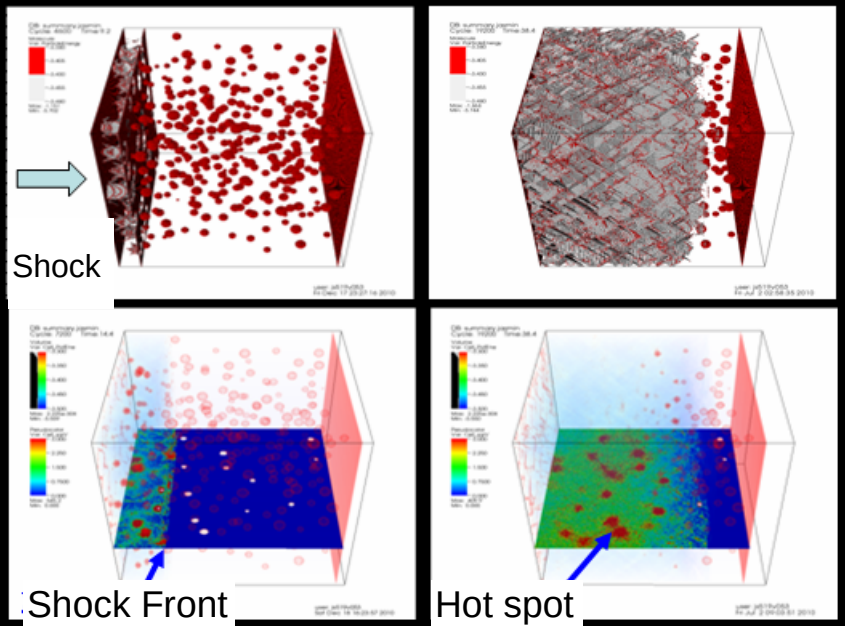
MD3D: 3-D molecule dynamics simulations for the structures and dynamics shock responses of metal materials with nano-scale defects.

#Molecules: 50G

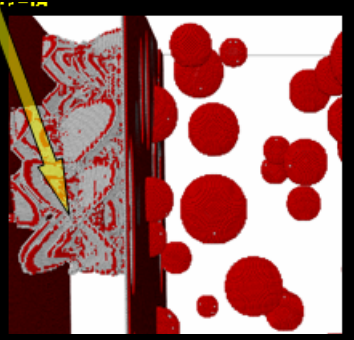
#Cores: 80,000
Para.Effi.: 62%

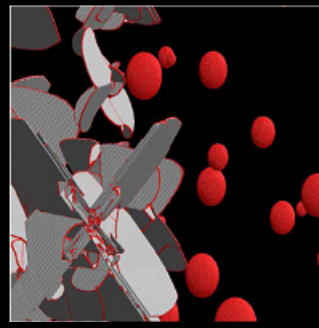
Ph.time: 5.5 ps
Sim. time: 3.0 hours
#steps: 30,000

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

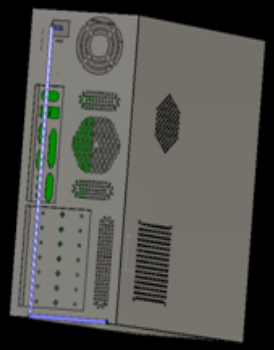
JMES3D: 3-D FDTD simulations for destroy mechanism of electromagnetic waves with different frequencies and directions.

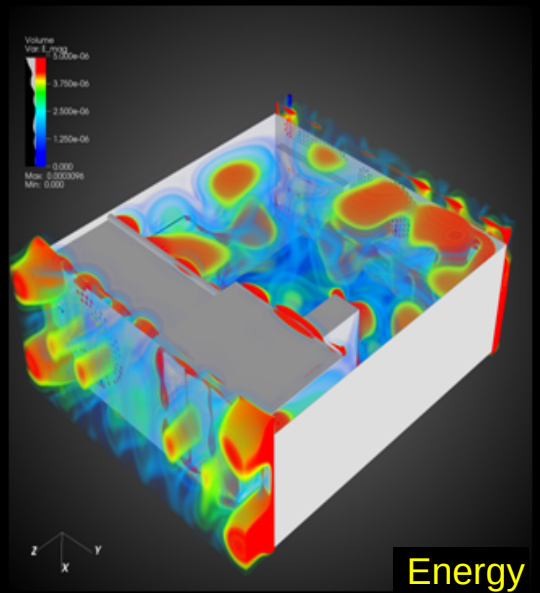
mesh: 1.2 G

#Cores: 60,000
Para.Effi.: 70%

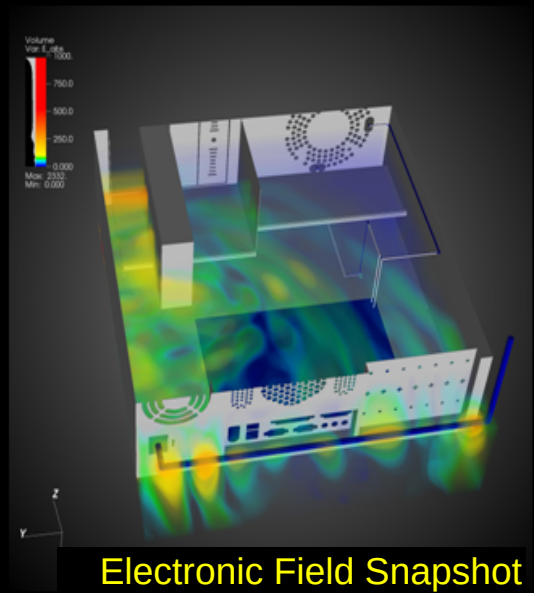
Phy.ime : 250 ns
S.time: 8.1 hours
#steps : 254,000

Output data
736 GB/ 172





Energy

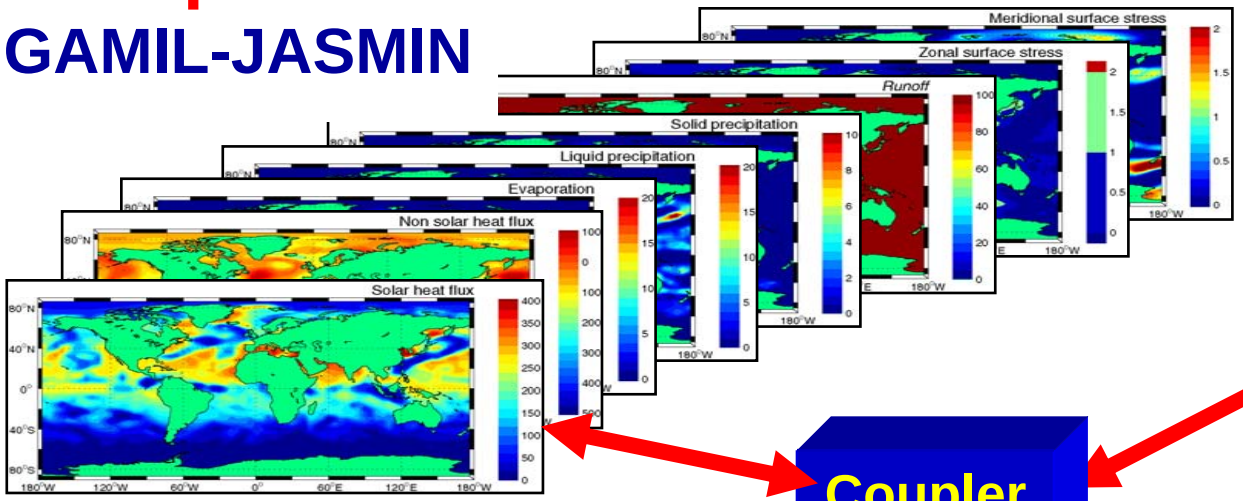


Electronic Field Snapshot

Applications-4：Climate Forecasting

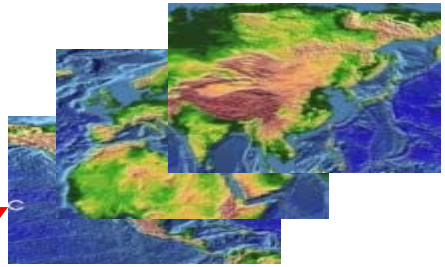
Atmosphere:

GAMIL-JASMIN

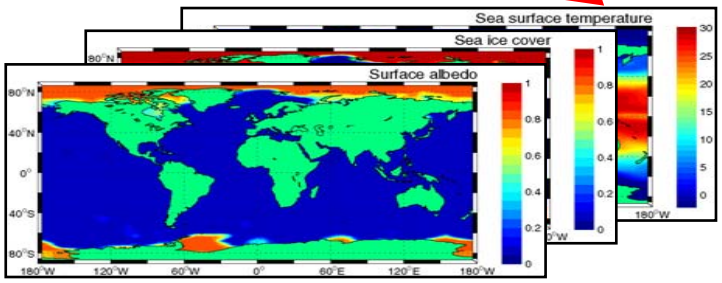


Land:

CLM-JASMIN

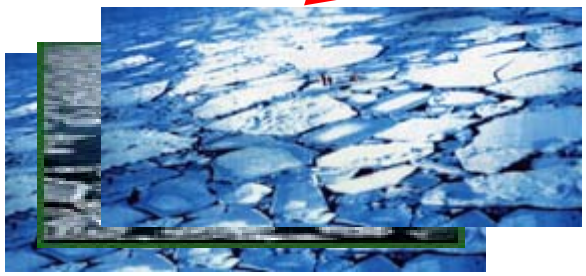


Coupler JASMIN



Ocean:

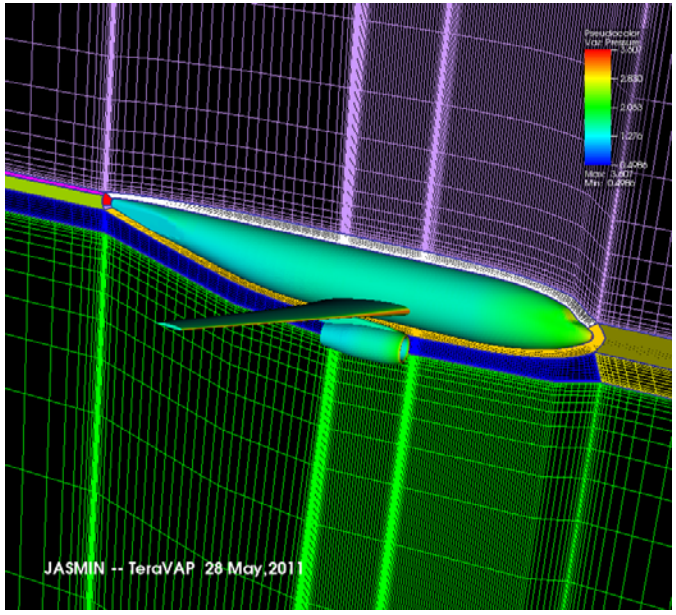
LICOM-JASMIN



Ocean ice:

CSIM4-JASMIN

Applications-5：CFD



mesh blocks distribution

cells:

9,574,784

blocks:

194

cores:

2048

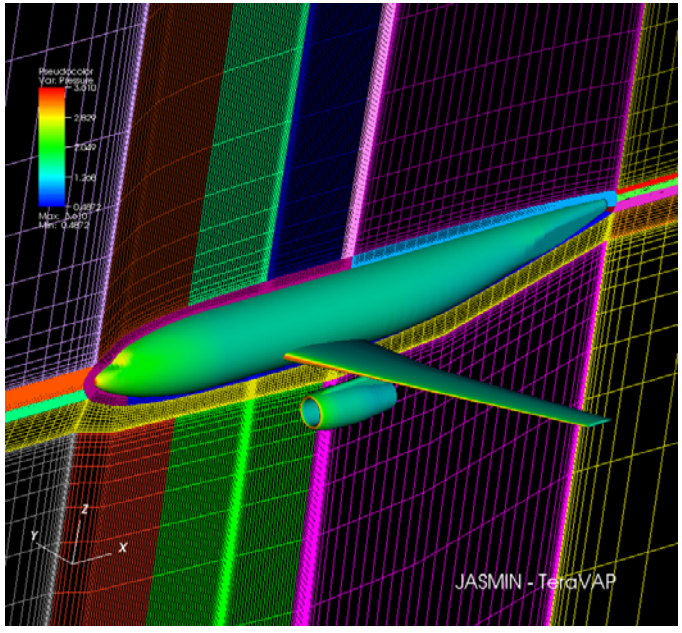
s/10K steps

800

speedup

580

DDM 64 cores



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.