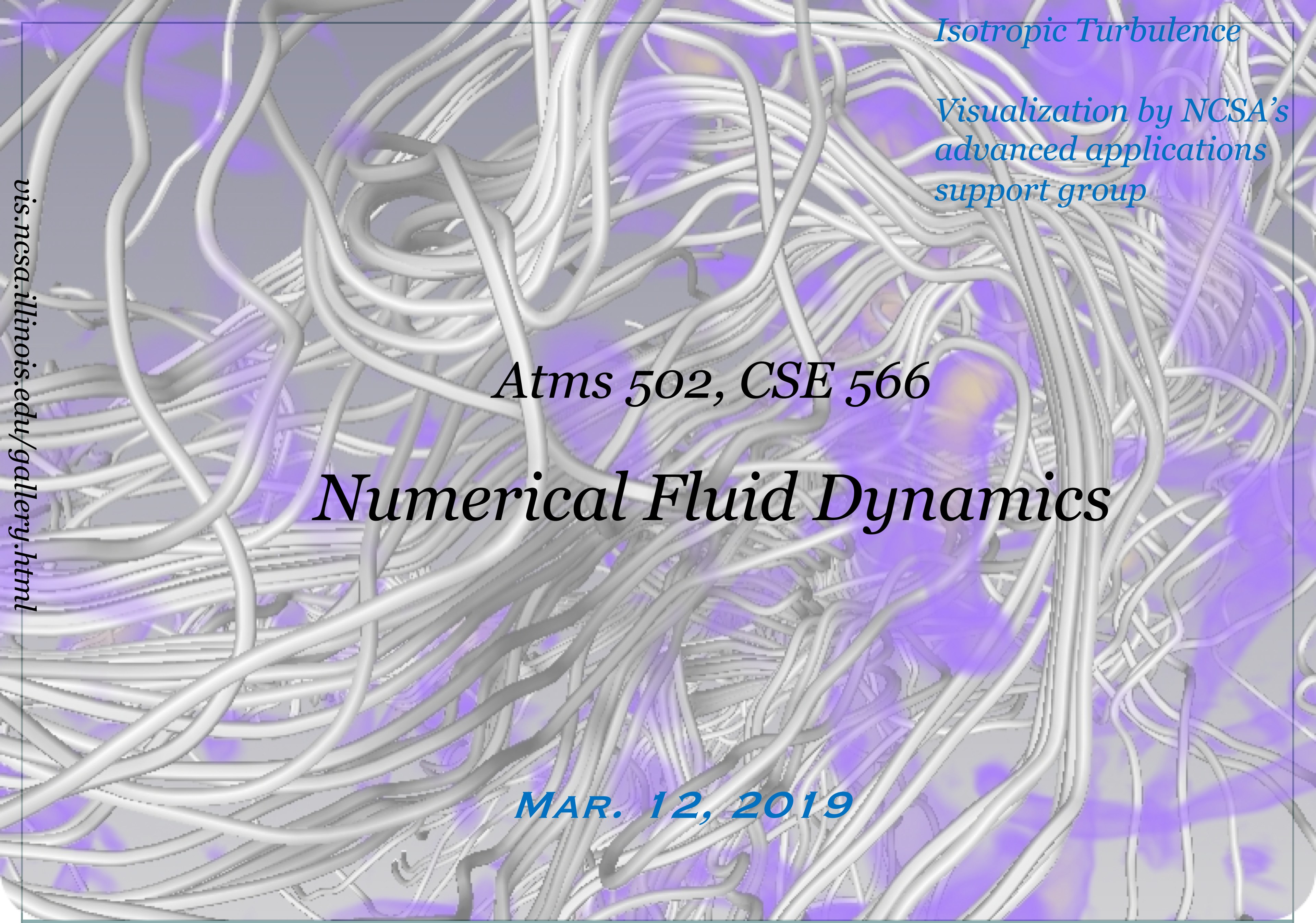


Isotropic Turbulence

Visualization by NCSA's advanced applications support group

vis.ncsa.illinois.edu/gallery.html

Atms 502, CSE 566

Numerical Fluid Dynamics

MAR. 12, 2019

ATMS 502
CSE 566

Tuesday,
12 March 2019

Class #17

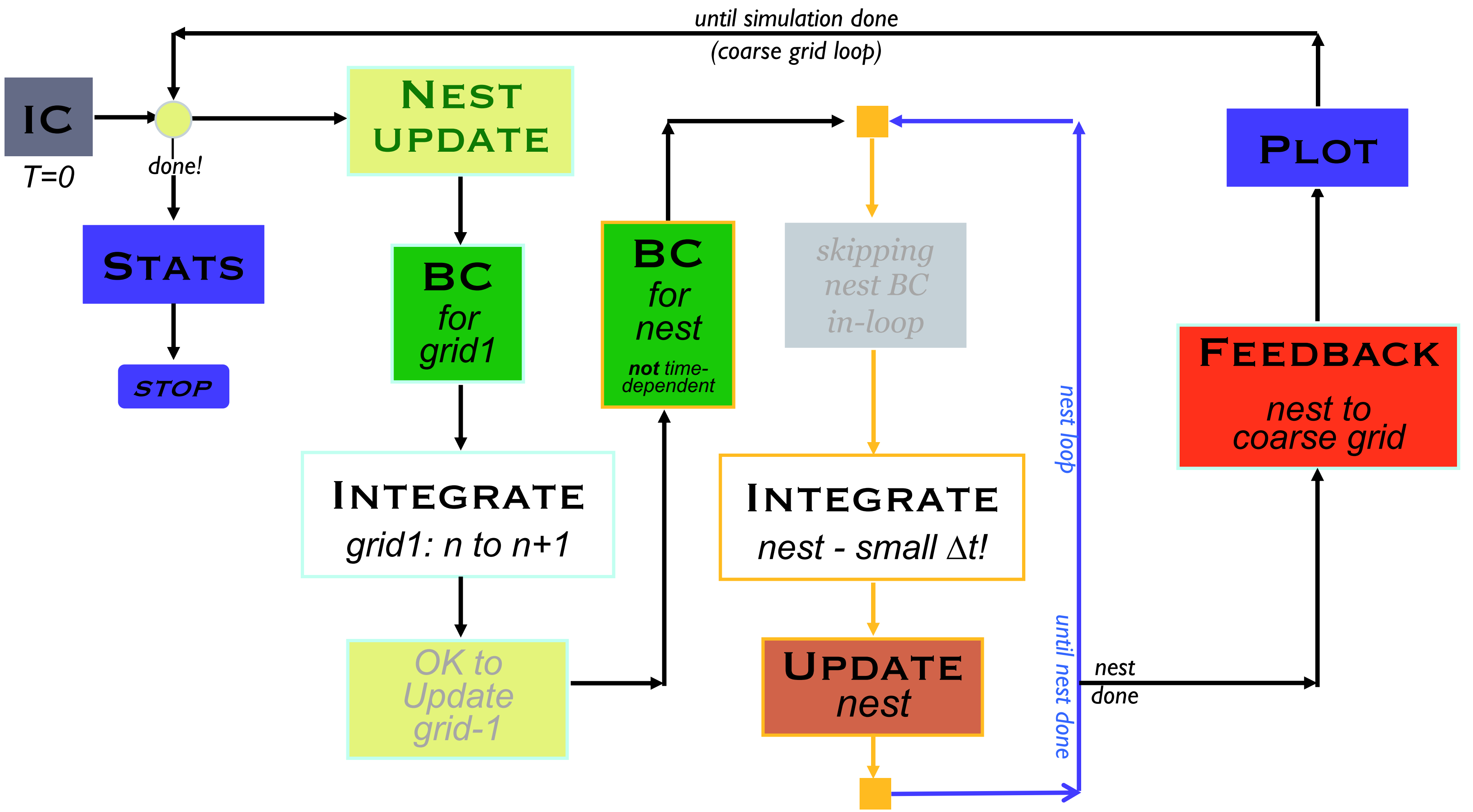
Plan for Today

- 1) Program 4, continued
  - Sequence & testing
- 2) Skamarock & Klemp
  - Introductory material missing last time
- 3) Parallel performance
  - Fundamentals - introduction

Program 4: suggested development

- Restore the rotating flow initial conditions (pgm2)
- Add nested-grid *s1, s2, u, v* arrays
- Nest-testing:
  - call your "truncation-error & nest-locating" routine after *IC*
  - call dointerp() to initialize the nest (fill nested s1 array)
  - call nestwind() to fill the nested u/v arrays
  - plot: coarse & nested grid arrays! Does they make sense?
    - Plot nest position on coarse grid plots. Plot all fields – all OK?
- Fake-nest
  - Main time step loop: every 5 steps, call the above routines to "re-initialize" the nest; show nest box on coarse grid plots.

Program #4: main routine



Skamarock & Klemp

THE MISSING INTRODUCTION
FROM LAST TIME.

Local vs. Global refinement

- Global: *move points*
  - Advantages:
    - Key: smoother transition between high, low resolution (less wave-reflection)
  - Disadvantages:
    - complex solver due to irregularity of grid
    - grid regenerated every time step
    - increased resolution in one area reduces it elsewhere.
    - time step set by that in highest resolution region
- Local: *add points*
  - Advantages:
    - Simple(r) solver
    - less development time
    - easier parallelization
    - Different time steps for different grid resolutions - more efficient.
  - Disadvantages:
    - Abrupt resolution changes cause noise e.g. wave reflection

Nesting strategy

- Identification & clustering

Richardson Extrapolation

$$\tau \approx \frac{Q_h^2(u(x,t)) - Q_{2h}(u(x,t))}{2^{q+1} - 2}$$

- In English! The procedure is:
  - Take two time steps as usual
  - Take one giant step with 2Δx,2Δt
  - Difference approximates the truncation error
  - "... if the solution is smooth"

"Adaptive Grid Refinement for Numerical Weather Prediction" - *William Skamarock, Stanford, 1987*

Richardson extrapolation (more)

- Richardson extrapolation – notes
  - From Skamarock – Stanford Dissertation, pp. 11-12 *(link)*
  - In Richardson extrapolation method, Q_h "is an operator representing the finite difference scheme
  - "Q_h^2 is the operator Q_h twice applied to u(x,t)
  - Advantages of this method to estimate truncation error:
    - Exact form of truncation error need not be known
    - For systems of equations with several variables, calculating truncation error accurately "can be very difficult"
    - The same solver is used to integrate equations & estimate error
    - Estimator is independent of finite difference method and PDE

Nesting strategy

- Identification
  - grid points exceeding some threshold, e.g. truncation error
- Clustering
  - fit to enclose points
  - general AMR allows overlapping grids, arbitrary orientation
- Nest: Initial conditions
  - Interpolated from coarse grid, or existing nests
- Nest: Bound. conditions
  - Time dependent
    - from coarse grid, using current and 'next' step.
  - Spatial dependence
    - interpolated from coarse grid. in our case, nest BC overlap with coarse points

- Feedback: nest to coarse grid
  - Average nest interior to coarse points

Finite volume method; van Leer

Overview:

- van Leer published five papers between 1973-79
- J. Comp. Phys., vol. 23, 276-299 (1977):

  "Towards the ultimate conservative difference scheme: IV. A new approach to numerical convection"

Handout:

- Hourdin and Armengaud, 1999: Use of finite volume methods for atmospheric advection of trace species
  - flux forms
  - monotonicity
  - "approximating the sub-grid-scale distribution by a polynomial" *(HA99 p. 823)*

Introduction: Parallel performance

Some figures from
High Performance Computing
by David Kuck (Oxford Press, NY)

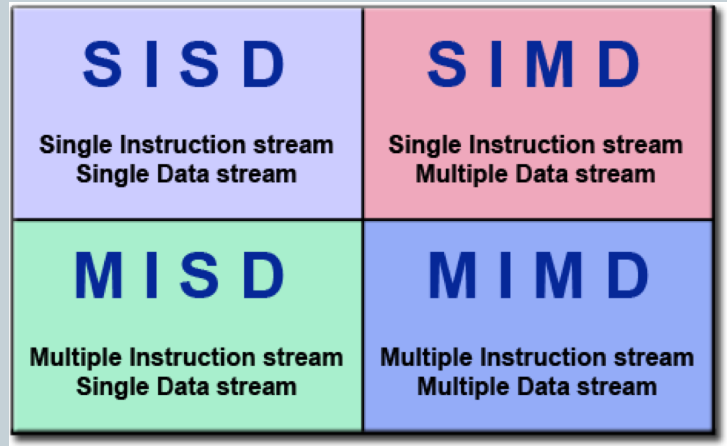
Others: LLNL pages on parallel computing:
https://computing.llnl.gov/tutorials/parallel_comp/

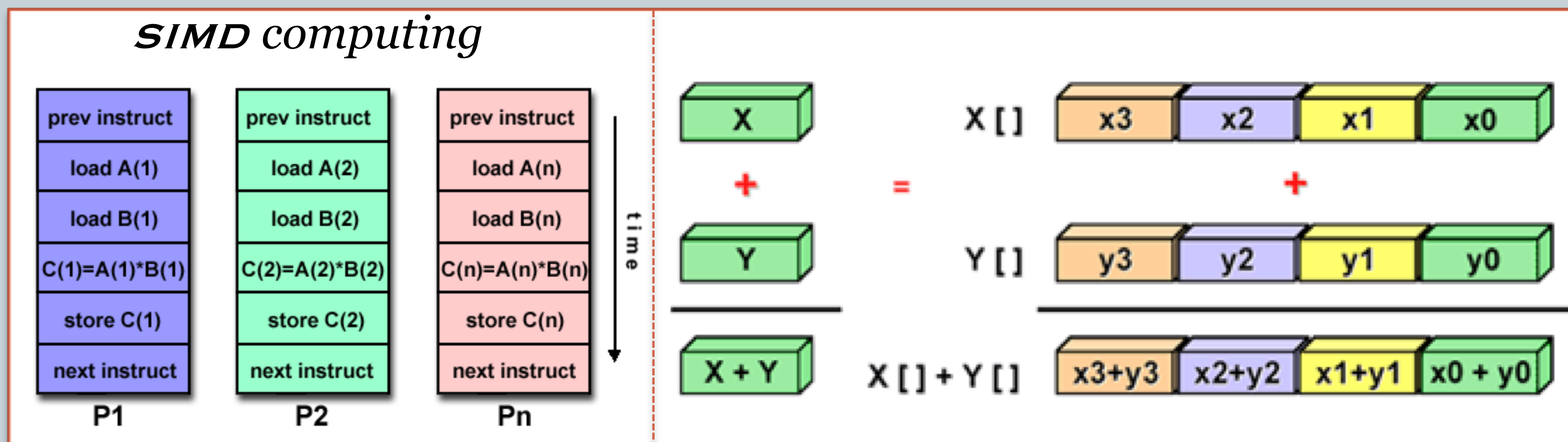
Really?

- Yes.
  - As computational scientists, some knowledge of how computers *work* gives you advantages over those who don't
  - We'll discuss *only* the basics. More information..
    - *U.I. Computer Science,* cs.illinois.edu
    - *Computational Science & Engineering,* www.cse.illinois.edu
      - UI Ph.D computer science ranked 5th in U.S.; Engineering, 4th (world)
- This information may help you answer questions like:
  - Why did my code not run *(even)* faster with more processors?
  - Why did the size of my grid make such a difference?
  - What delays / difficulties am I likely to encounter?
  - How do I make best use of a computer cluster to do my work?

Introductory concepts

- Types of computing: Flynn's taxonomy
- Doing the same task on all cores is *SIMD*

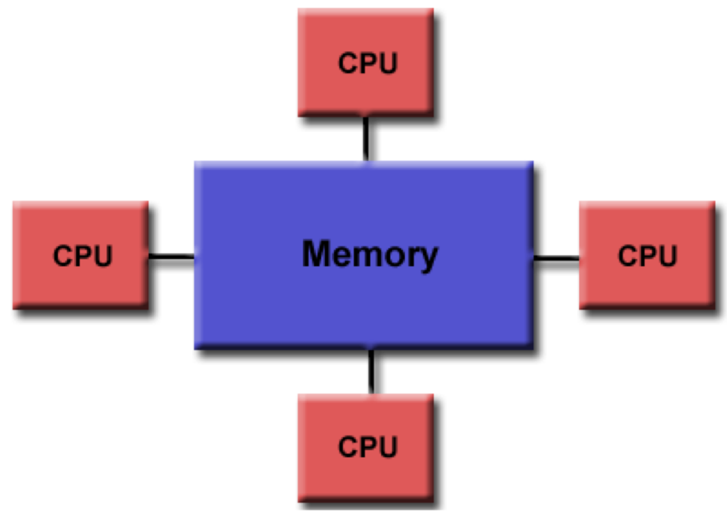




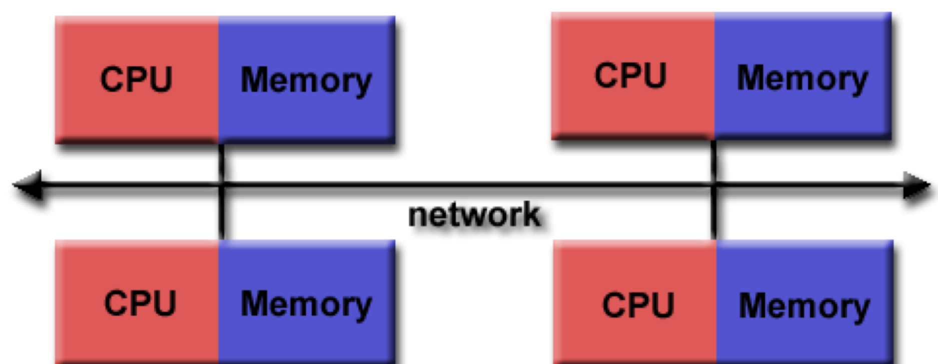
 all figures: https://computing.llnl.gov/tutorials/parallel_comp/

Shared vs. Distributed Memory Architecture

- **Shared** memory
  - multiple processors access all memory as *global address space*.



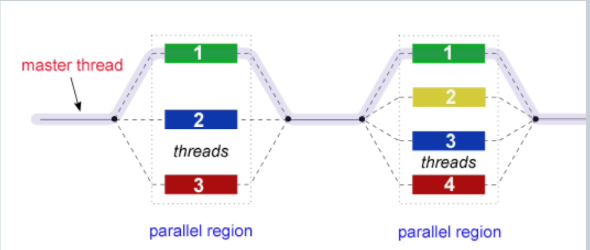
- **Distributed** memory
  - processors have local memory
  - programmer handles data movement explicitly.

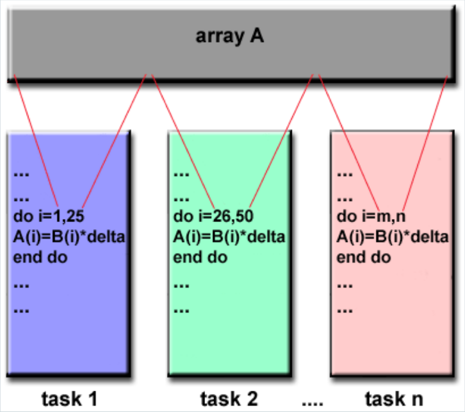


all figures: https://computing.llnl.gov/tutorials/parallel_comp/

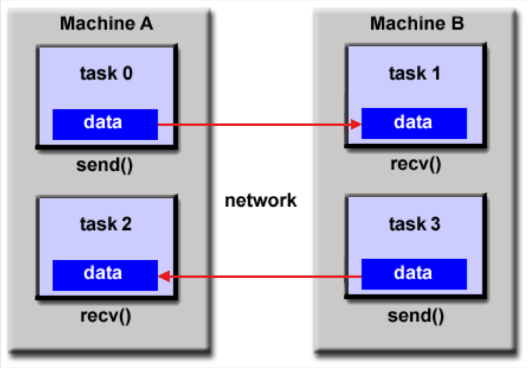
Shared vs. Distributed Memory Parallelism

- **Shared** memory parallelism: *OpenMP*
  - main program creates tasks *(threads)* that can be run concurrently (simultaneously).
    - in *loop-level* parallelism, threads handle loop tasks.





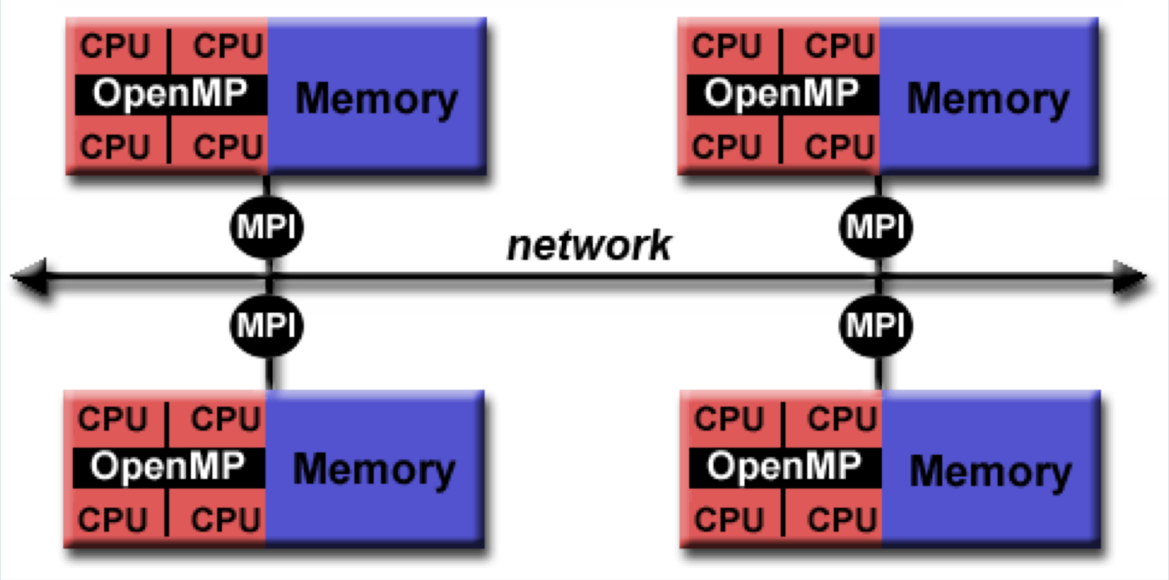
- **Distributed** memory / Message passing interface: *MPI*
  - Each task uses local memory
    - Data exchanged by sending and receiving messages
    - Latest version is MPI 3.1



 all figures: https://computing.llnl.gov/tutorials/parallel_comp/

Hybrid architectures & parallelism

- **Hybrid parallelism** combines OpenMP threads model with the message passing (MPI) model
  - threads handle local (on-node) data
  - communication between nodes is done via MPI



 all from: https://computing.llnl.gov/tutorials/parallel_comp/