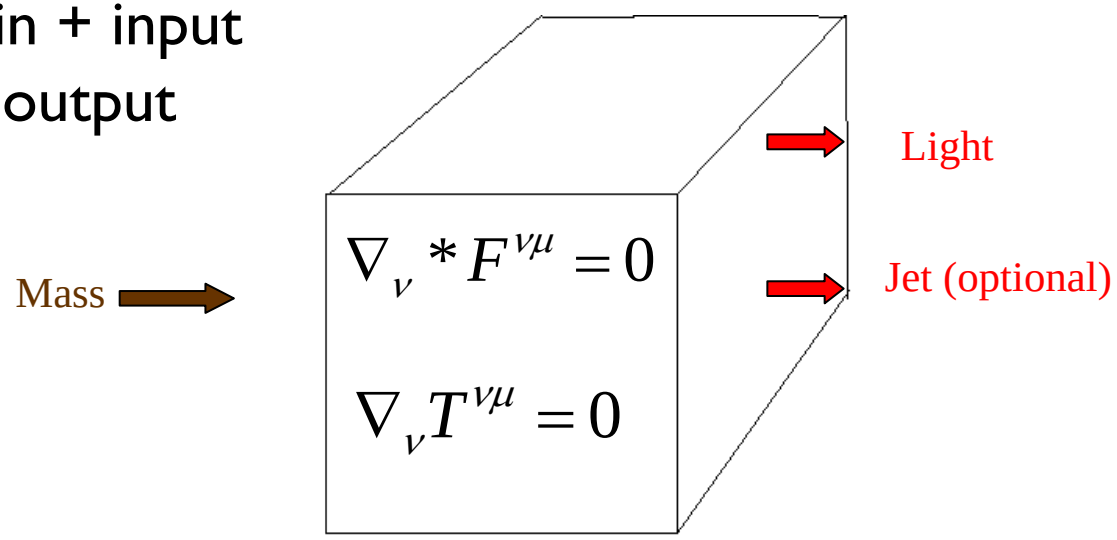
Computational Astrophysics

John Hawley

University of Virginia

The Goal of Accretion Simulations

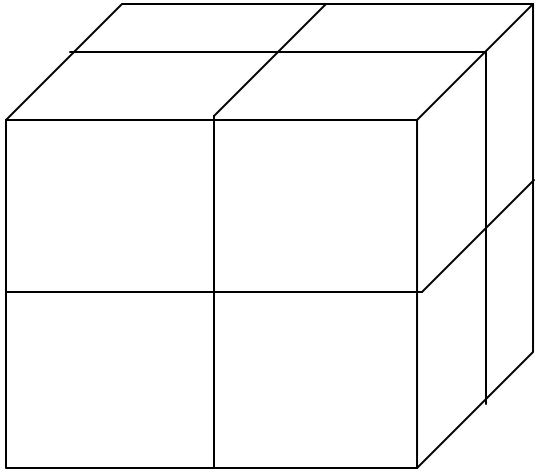
- Let the equations determine the properties of accreting systems
- Black hole mass, spin + input fuel and field yields output



Aspects of Computational Astrophysics

- Problem definition
- Selection of physics and physical model
- Analytic equations
- Specific form of equations
- Numerical realization – algorithm
- Software implementation
- Problem specification – initial conditions
- Problem execution – specific instance
- Data analysis
- Conclusions

Grid-based methods



1. Discretize space into zones

$$\mathbf{x} \rightarrow (x_i, y_j, z_k)$$

2. Discretize the continuous variables

$$\rho(\mathbf{x}, t) \rightarrow \rho^n_{i,j,k} \quad \text{where} \quad \rho^n_{i,j,k} = \int \rho(\mathbf{x}, t^n) dV / \int dV$$

3. Difference the conservation laws $\dfrac{\partial \mathbf{U}}{\partial t} = -\nabla \cdot \mathbf{F}(\mathbf{U})$

$$\text{as} \quad \mathbf{U}_i^{n+1} - \mathbf{U}_i^n = -\frac{\Delta t}{\Delta x}\left(\mathbf{F}^{n+1/2}_{i+1/2} - \mathbf{F}^{n+1/2}_{i-1/2}\right)$$

Difficulty is computing accurate and stable fluxes: $\mathbf{F}^{n+1/2}_{i\pm 1/2}$

Two non-relativistic MHD codes:

- ZEUS (Stone & Norman 1992a; b)
  - Higher-order upwind methods used to compute fluxes
  - MOC-CT method developed to evolve magnetic fields
  - Wide range of physics options
  - Parallelized with OpenMP and MPI
  - Freely distributed to astrophysics community

- ATHENA (Stone et al. 2008)
  - Conservative formulation
  - Uses linearized Riemann solver to compute fluxes
  - Works with nested and adaptive meshes
  - Parallelized with MPI
  - Freely available to community

Algorithm types: ZEUS

- Finite difference (ZEUS type)
  - Explicit differencing of terms; quantities located on grid appropriately
  - Operator splitting – “transport” plus “source”
  - Directional splitting – grid directions handled separately in turn
  - Flexible – additional terms easily added
  - Relatively low order
  - Not explicitly conservative. Total or internal energy equations may be evolved

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\mathbf{v} = 0,$$

$$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p - \rho\nabla\Phi + \frac{1}{4\pi}(\nabla\times\mathbf{B})\times\mathbf{B}$$

$$\rho\frac{D}{Dt}\left(\frac{e}{\rho}\right) = -p\nabla\cdot\mathbf{v}$$

$$\frac{\partial\mathbf{B}}{\partial t} = \nabla\times(\mathbf{v}\times\mathbf{B}),$$

Algorithm types: Athena

- Godunov (Athena type)
  - Built around flux conservative form of conservation laws
  - Fluxes constructed from approximate solutions to hydro/MHD characteristic equations
  - Directional splitting (1D Riemann problem)
  - Can be harder to add additional physics; non-Cartesian coordinates trickier

$$\mathbf{U}_i^{n+1} - \mathbf{U}_i^n = -\frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+1/2}^{n+1/2} - \mathbf{F}_{i-1/2}^{n+1/2}\right) + S_i$$

U is the conservative state vector = (ρ, $\rho\mathbf{v}$, E, $\mathbf{B}$)

F is the flux vector F(U)

Source terms can be added to the right hand side (e.g. external forces)

The two challenges of numerical MHD

1. There are 3 wave families in MHD, which are sometimes degenerate
   → Greatly complicates computation of fluxes of conserved variables

2. Evolved field must satisfy the divergence-free constraint
   → requires a conservative scheme for the *magnetic flux*

(Evans & Hawley 1988, Stone & Norman 1992)

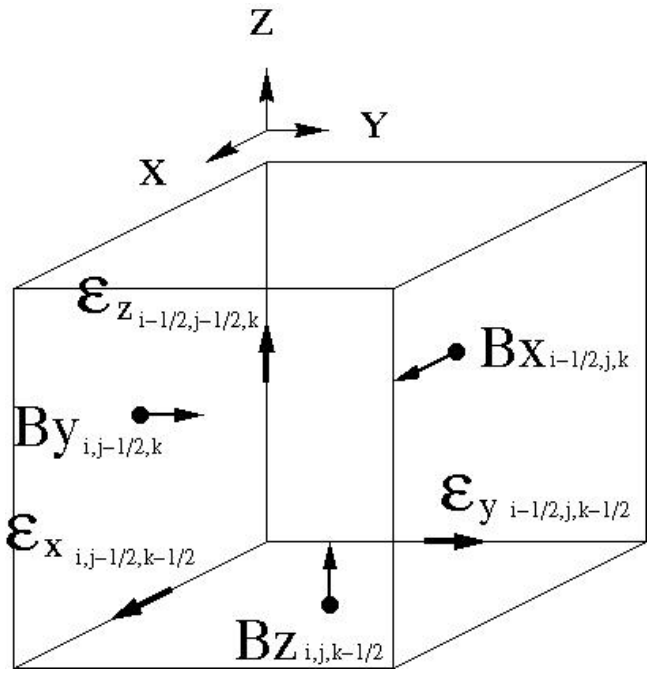
Integrate the induction equation

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}),$$

using Stoke's Law to give

$$\frac{\partial \Phi_B}{\partial t} = \oint_{\partial S} (\mathbf{v} \times \mathbf{B}) \cdot dl \quad \text{where} \quad \Phi_B = \int \mathbf{B} \cdot dA$$

Difference using a staggered B and EMFs located at cell edges.



To gauge their accuracy in different regimes, numerical methods must be tested.

1. Compare to analytic solutions where possible
2. Compare solutions generated by different methods
3. Compare to results of laboratory experiments
4. Convergence Testing

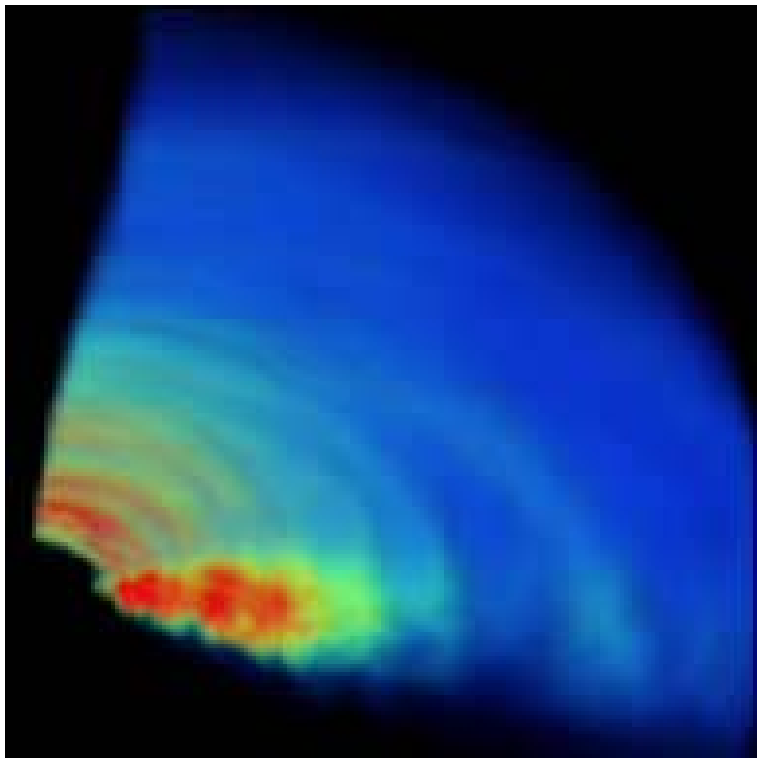
Athena code test pages:

http://www.astro.princeton.edu/~jstone/tests/index.html

http://www.astro.virginia.edu/VITA/athena.php

Black Hole Accretion Simulations

- General Relativistic MHD codes can be constructed along these same lines
- Newtonian codes can also be adapted through the use of an effective potential $\Phi = GM/(r - r_g)$ where r_g = horizon size



Some General Relativistic Magnetohydrodynamics Codes

- Wilson (1975)
- Koide et al. (2000)
- Gammie, McKinney & Toth (2003)
- Komissarov (2004)
- De Villiers & Hawley (2003)
- Duez et al. (2005)
- Fragile & Anninos (2005)
- Anton et al. (2005)
- Noble (2008), McKinney (2008)

Equations of GR MHD

Continuity Equation

$$\nabla_\mu (\rho U^\mu) = 0,$$

Conservation of stress energy

$$\nabla_\mu T^{\mu\nu} = 0,$$

$$T^{\mu\nu} \equiv T^{\mu\nu}_{(fluid)} + T^{\mu\nu}_{(EM)}$$

$$T^{\mu\nu}_{(fluid)} = \rho h U^\mu U^\nu + P g^{\mu\nu}$$

$$T^{\mu\nu}_{(EM)} = \frac{1}{4\pi}\left(F^\mu{}_\alpha F^{\nu\alpha} - \frac{1}{4} F_{\alpha\beta} F^{\alpha\beta} g^{\mu\nu}\right)$$

Maxwell's equations

$$\nabla_\mu F^{\mu\nu} = 4\pi J^\nu,$$

$$\nabla_\mu {}^*F^{\mu\nu} = 0,$$

The Schwarzschild Metric

$$\Delta s^2 = \left(1 - \frac{2GM}{c^2 R}\right) c^2 \Delta t^2 - \frac{\Delta R^2}{1 - \frac{2GM}{c^2 R}} - \text{angle terms}$$

- Boyer-Lindquist coordinates
- Coordinate singularity at the Schwarzschild radius
- Actual Singularity at *R=0*

Kerr Metric

$$ds^2 = -\left(1 - \frac{2Mr}{\Sigma}\right) dt^2 - \frac{4aMr \sin^2\theta}{\Sigma} dt d\phi$$

$$+\frac{\Sigma}{\Delta} dr^2 + \Sigma d\theta^2$$

$$+\left(r^2 + a^2 + \frac{2Mra^2 \sin^2\theta}{\Sigma}\right) \sin^2\theta d\phi^2$$

$$\Sigma = r^2 + a^2 \cos^2\theta$$

$$\Delta = r^2 - 2Mr + a^2$$

Formulation of GRMHD Equations

Advection:
$$\partial_t D + \frac{1}{\sqrt{\gamma}}\partial_j\left(D\sqrt{\gamma}V^j\right) = 0$$

Momentum:
$$\partial_t\left(S_j - \alpha b_j b^t\right) + \frac{1}{\sqrt{\gamma}}\partial_i\sqrt{\gamma}\left(S_j V^i - \alpha b_j b^i\right) + \frac{1}{2}\left(\frac{S_\epsilon S_\mu}{S^t} - \alpha b_\mu b_\epsilon\right)\partial_j g^{\mu\epsilon} + \alpha\partial_j\left(P + \frac{\|b\|^2}{2}\right) = 0$$

Internal Energy:
$$\partial_t(E) + \frac{1}{\sqrt{\gamma}}\partial_i\left(\sqrt{\gamma}EV^i\right) + P\partial_t(W) + \frac{P}{\sqrt{\gamma}}\partial_i\left(\sqrt{\gamma}WV^i\right) = 0$$

Induction:
$$\partial_\delta F_{\alpha\beta} + \partial_\alpha F_{\beta\delta} + \partial_\beta F_{\delta\alpha} = 0$$

+ scheme that solves
$V = F(D,E,||b||^2,W,S_j)$; $b^\mu = G(V,F_{\alpha\beta})$
(see Noble et al. 2006)

DeVilliers & Hawley 2003

γ is the determinant of the three metric

Accretion into Black Holes: GRMHD implementation

- Fixed Kerr Metric in spherical Boyer Lindquist coordinates
- Boyer Lindquist t and ϕ coordinates consistent with conserved angular momentum and energy
- Graded radial mesh - inner boundary just outside horizon; θ zones concentrated at equator
- Induction equation of form

$$F_{\alpha\beta,\chi} + F_{\beta\chi,\alpha} + F_{\chi\alpha,\beta} = 0$$

- Baryon Conservation, stress-energy conservation, entropy conservation (internal energy); no cooling
- First order, time-explicit, operator split finite differencing
- Similar to ZEUS code

Conservative GR MHD Scheme (HARM)

Conservative state vector

$$\mathbf{U} = \sqrt{-g}\left(\rho U^t, T_t^{\ t}, T_t^{\ i}, B^i\right)$$

Primitive state vector

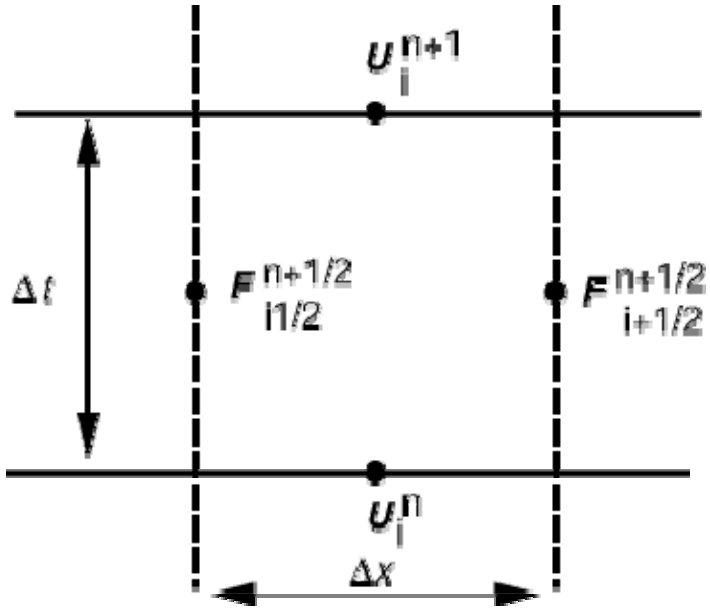
$$\mathbf{P} = \left(\rho, \epsilon, v^i, B^i\right)$$

Conservative update scheme

$$U_i^{\ n+1} = U_i^{\ n} - \frac{\Delta t}{\Delta x}\left(F_{i+1/2}^{\ \ n+1/2} - F_{i-1/2}^{\ \ n+1/2}\right)$$

Overall timestep update

$$\mathbf{U}^n \rightarrow \mathbf{P} \rightarrow \mathbf{F}(\mathbf{P}) \rightarrow \mathbf{U}^{n+1}$$



Use of Kerr-Schild coordinates avoid coordinate singularity at horizon

Introduction to High Performance Computing

Why Compute in Parallel?

Most large scientific applications exceed the capabilities of a single processor

Get a greater amount of computing done in less real time

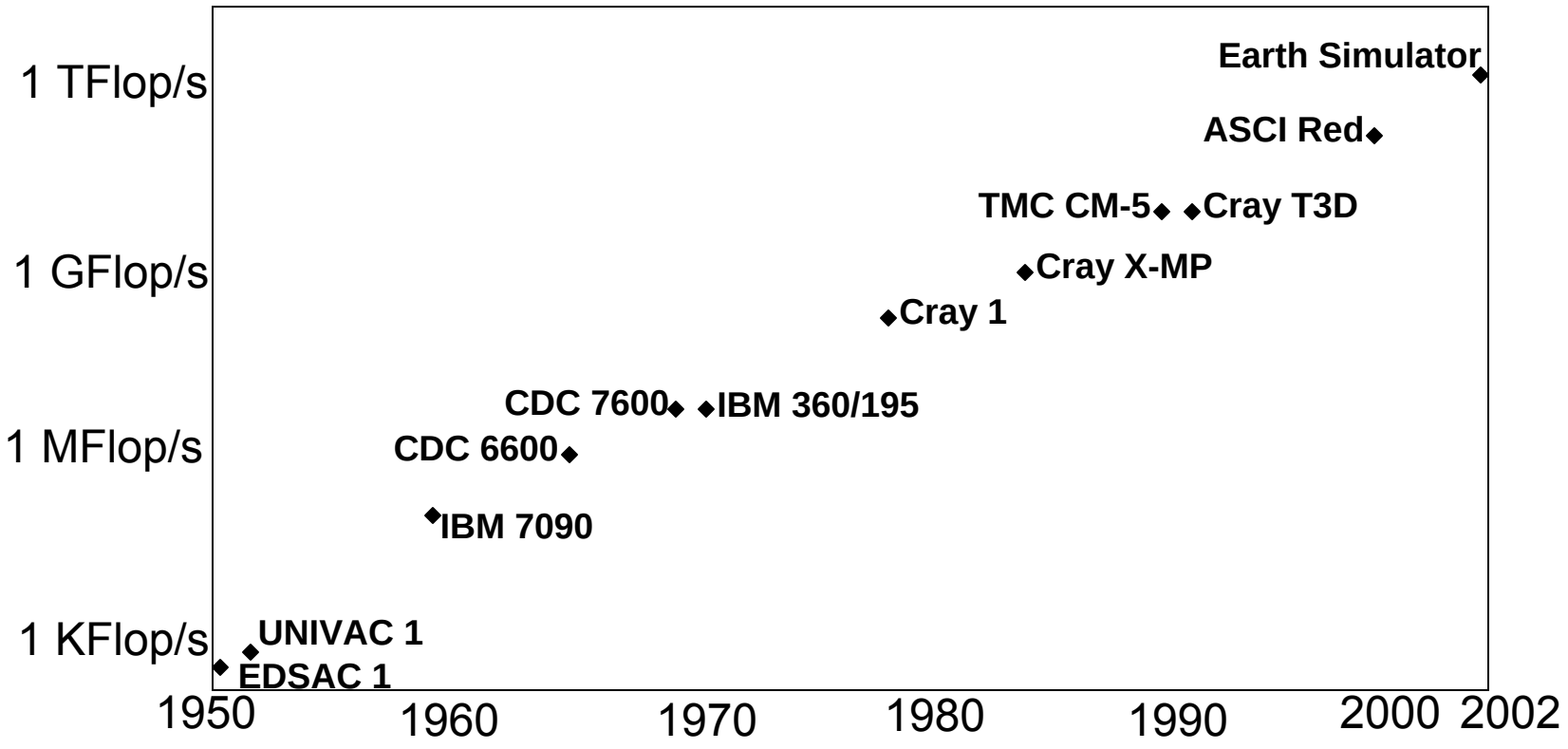
Moore's Law (doubling of cpu clockspeed every 18-24 months) has effectively reached its end. Manufacturers are trying to compensate by producing multi-core chips.

Furthermore:

The rate of decrease in memory latency (the time required to access the first bit of information) is very slow.

Other factors: disk capacities have increased enormously, but they are mechanical devices and their latencies decrease fairly slowly.

Network speeds (internal and external) have also increased fairly slowly, and this speed is ultimately limited by light travel times across the network.



Peak performance of supercomputers versus time

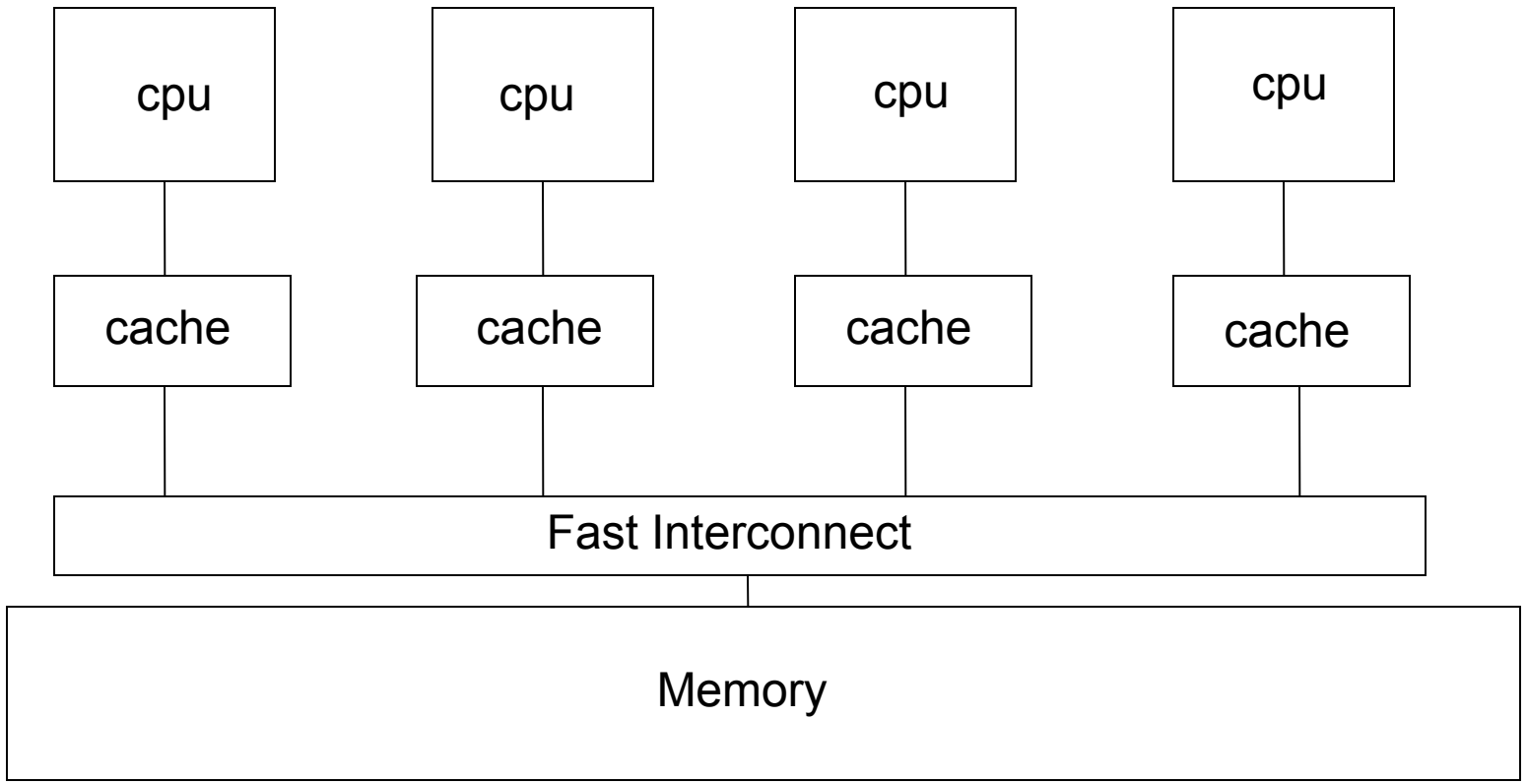
Flop/s = floating-point operations/second

Peak performance for supercomputers has followed Moore's Law quite well, but only about half the improvement is due to increases in single-processor speed, while the rest is due to an increasing number of processors.

In many cases, parallelism has been the only means by which large computational runs can be performed.

Parallel Architectures

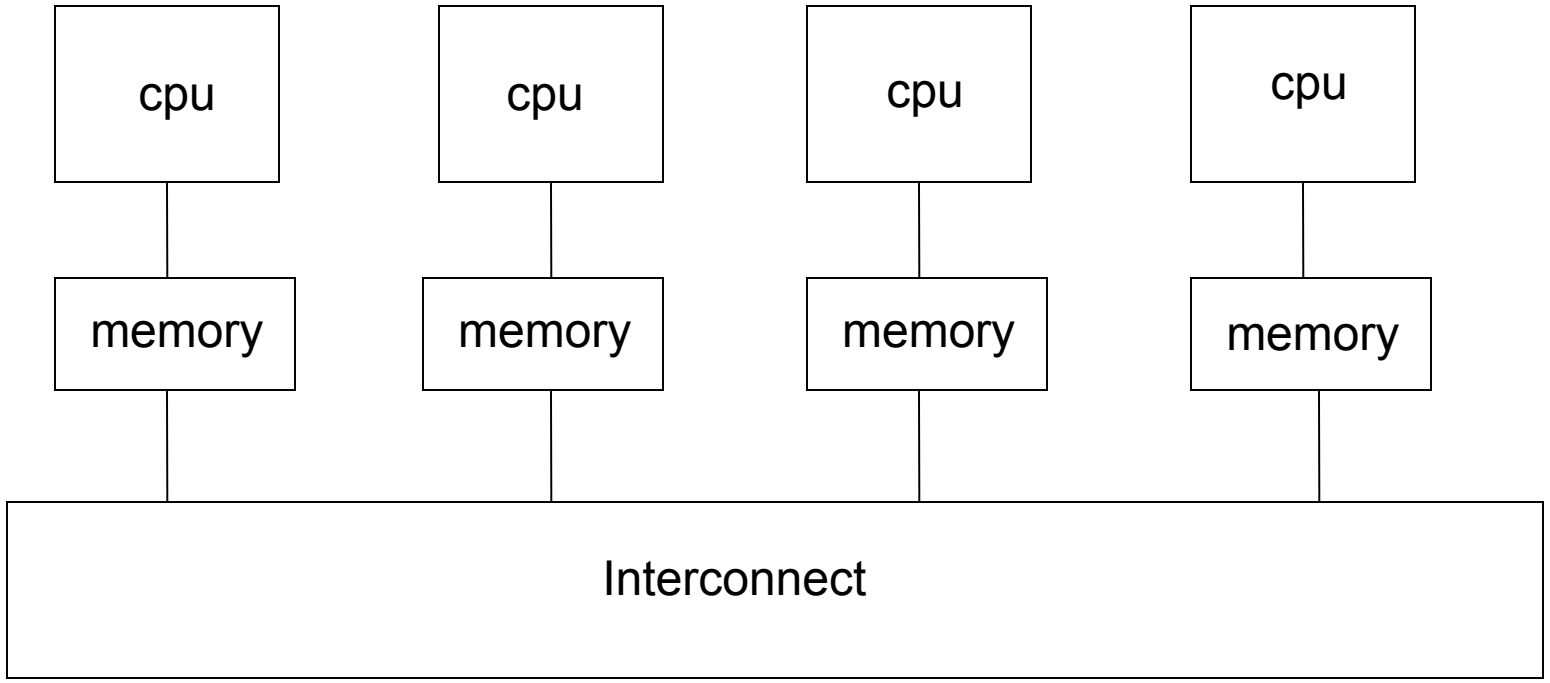
SMP – Symmetric Multiprocessing
Uniform Memory Access (UMA)



Parallel Architectures

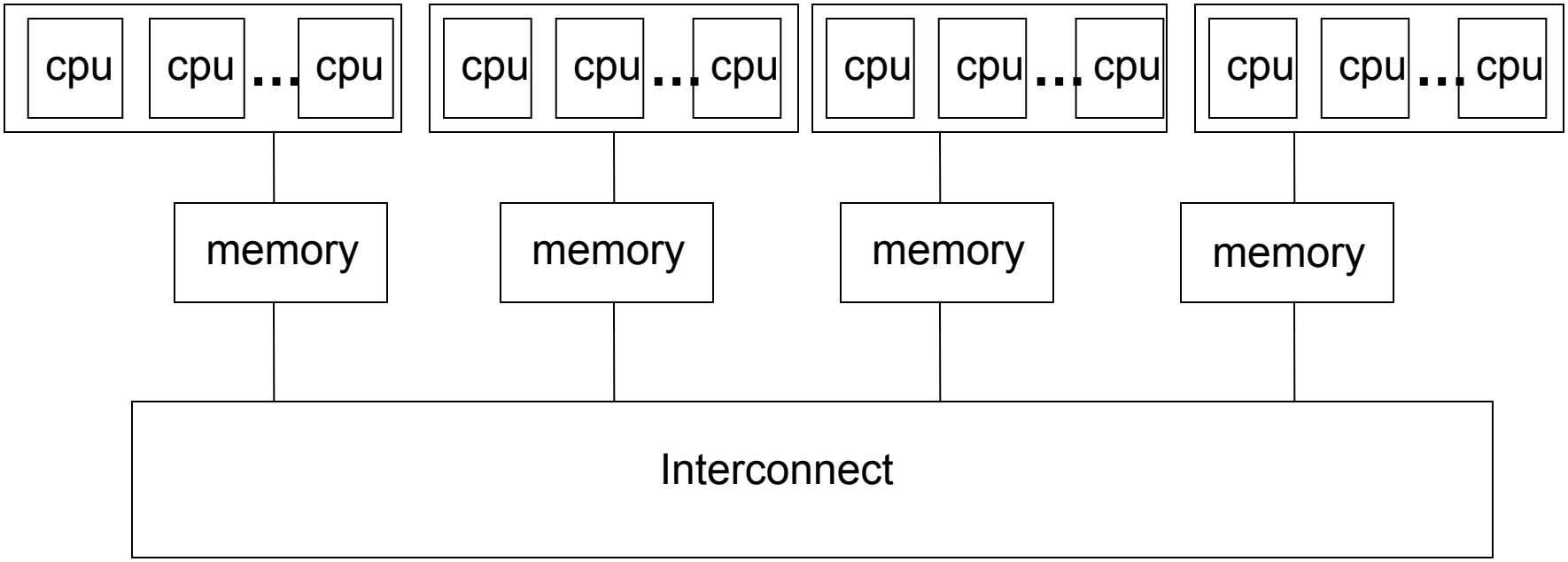
MPP – Massively Parallel Processor

NonUniform Memory Access (NUMA): each processor can see other processors' memory
Distributed Memory: memory inaccessible to other processors



Parallel Architectures

"Constellation"



Some Problems with Parallel Architectures

- SMP
  - Interconnect latency and bandwidth
  - Interconnect contention
  - Memory contention
  - Cache coherence
- NUMA
  - Communication lags
  - Cache coherence
- Distributed Memory
  - Communication latency, bandwidth, overhead

Cache coherence

This is a problem for systems that rely on access to common memory. Suppose that an array A is in memory. Before using it, the processors will load it into their local caches. Now suppose that Processor 100 makes some change to A and writes it to main memory. Unless all the other processors are explicitly told to refresh their caches, they will not get the updated value.

Cache coherence is difficult to ensure and causes interrupts. It is a major reason that SMP and NUMA systems cannot scale beyond a few thousand cpus.

Computing Models

There are two dominant types of problem decomposition

- Task parallelism (MIMD)
  - divide tasks among the processors.
  - sometimes can be “embarrassingly parallel” with very little interprocess communication required.
- Data parallelism (SIMD)
  - divide data among the processors.
  - Each process executes same instructions on different data

Programming Models

- Threads/OpenMP
  - For SMP (SIMD)
  - Easy to use but can be hard to get optimal speedup
- Message Passing
  - For MPP or Distributed Memory Clusters
  - Corresponds to MIMD computing model
  - Harder to use but generally gives best results
  - Can be used with SMP/NUMA systems with right libraries
- Hybrid
  - For “constellations” – more complex programming
  - May or may not result in any benefit over message passing

Communication Overhead

Broadly speaking, parallel codes can be divided into two categories:

Coarse grained:

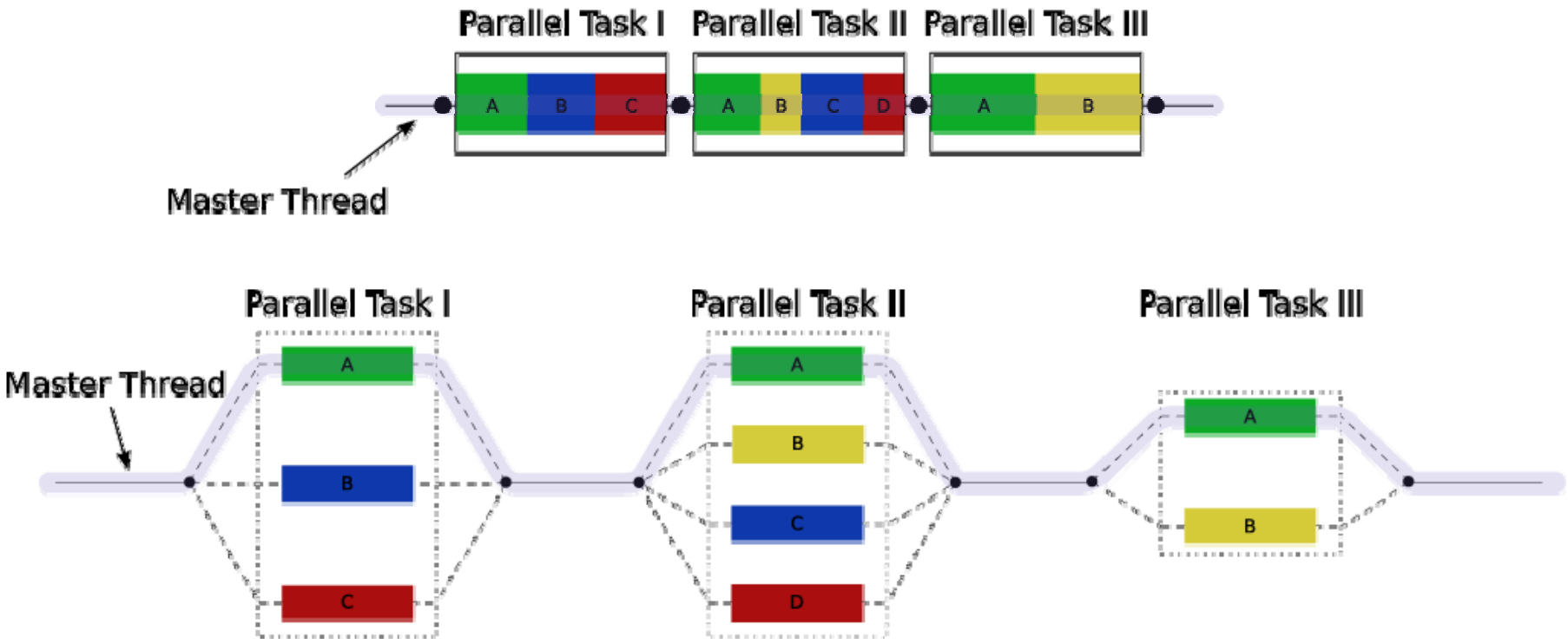
Few communications: little time relative to computations. Computations are nearly independent of one another and have little or even no need to communicate: the "embarrassingly parallel" type of problem.
Examples: many Monte-Carlo type codes, protein sequence comparisons, sorting, etc. Not very sensitive to overhead in communications.

Fine grained:

Many communications, communication time significant in comparison to computation time. Examples: some finite-difference and finite-element codes, molecular dynamics codes, etc. Can be very sensitive to communication costs, especially interconnect latency.

Open_MP

- Multi-processing in shared memory threads



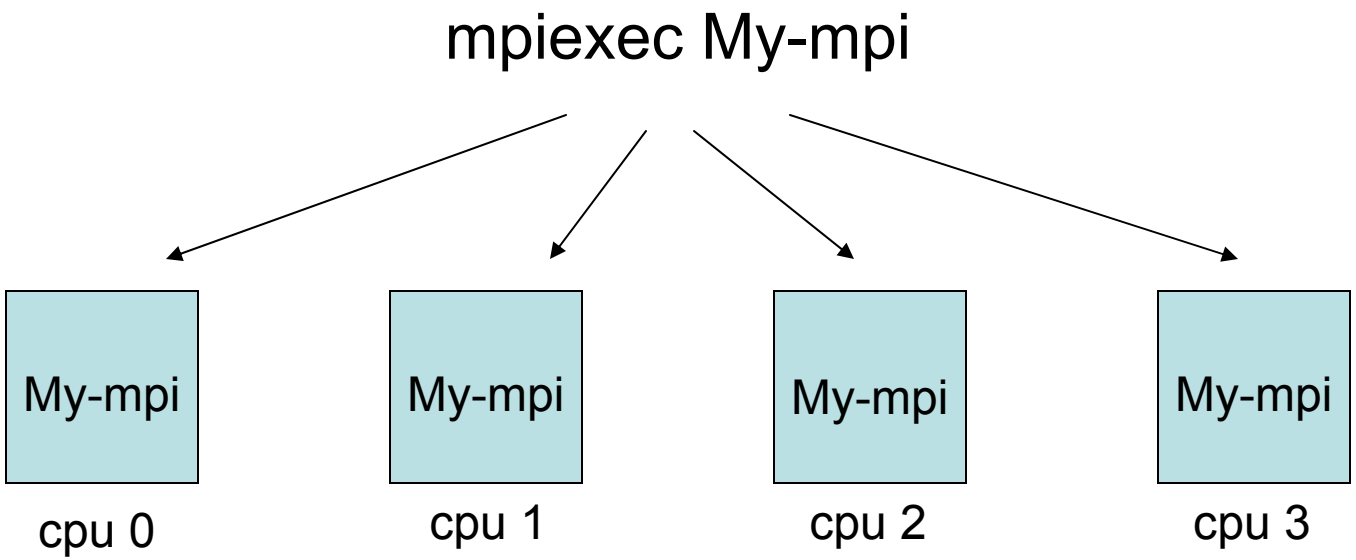
Message Passing

Message passing requires that the programmer manage the interchange of data among the processes.

Identical program runs on multiple processes and each process will work only on a subset of the data.

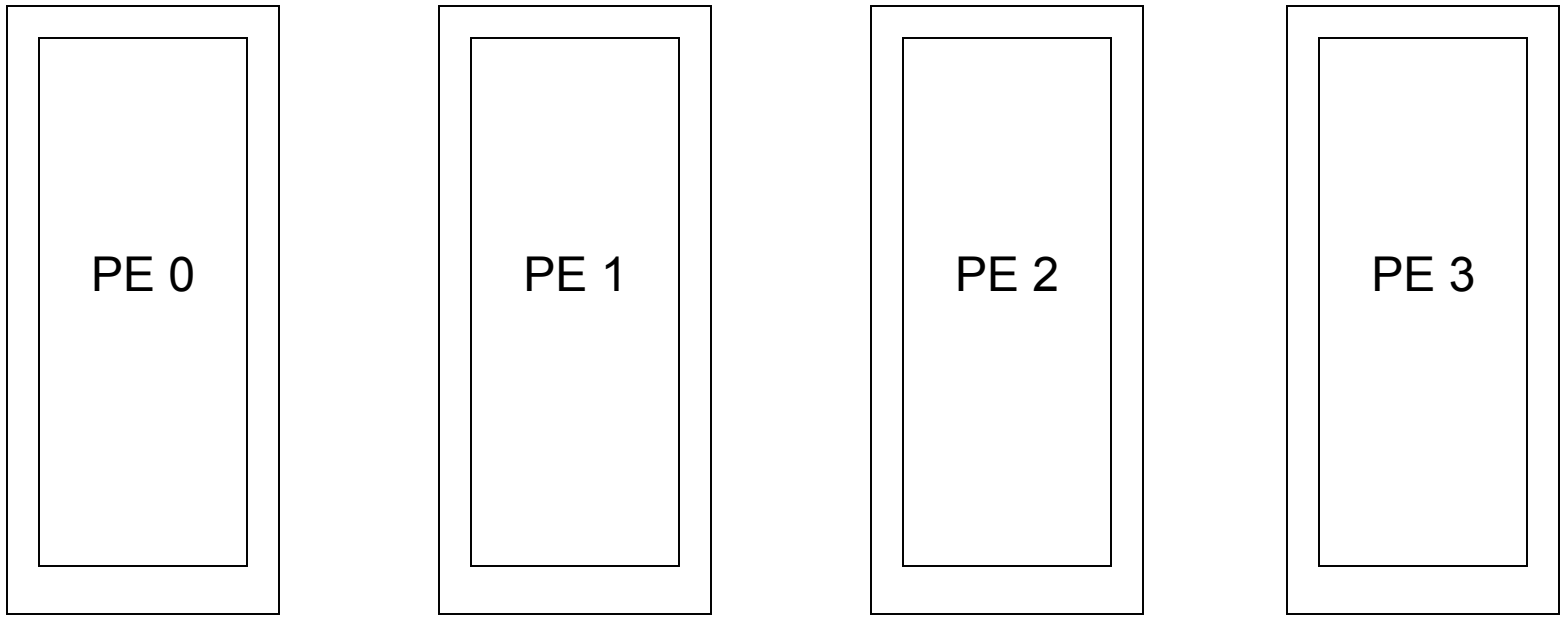
The programmer is responsible for distributing data to each process, keeping the processes up to date with whatever data they need, and collecting the results as required.

An MPI Program



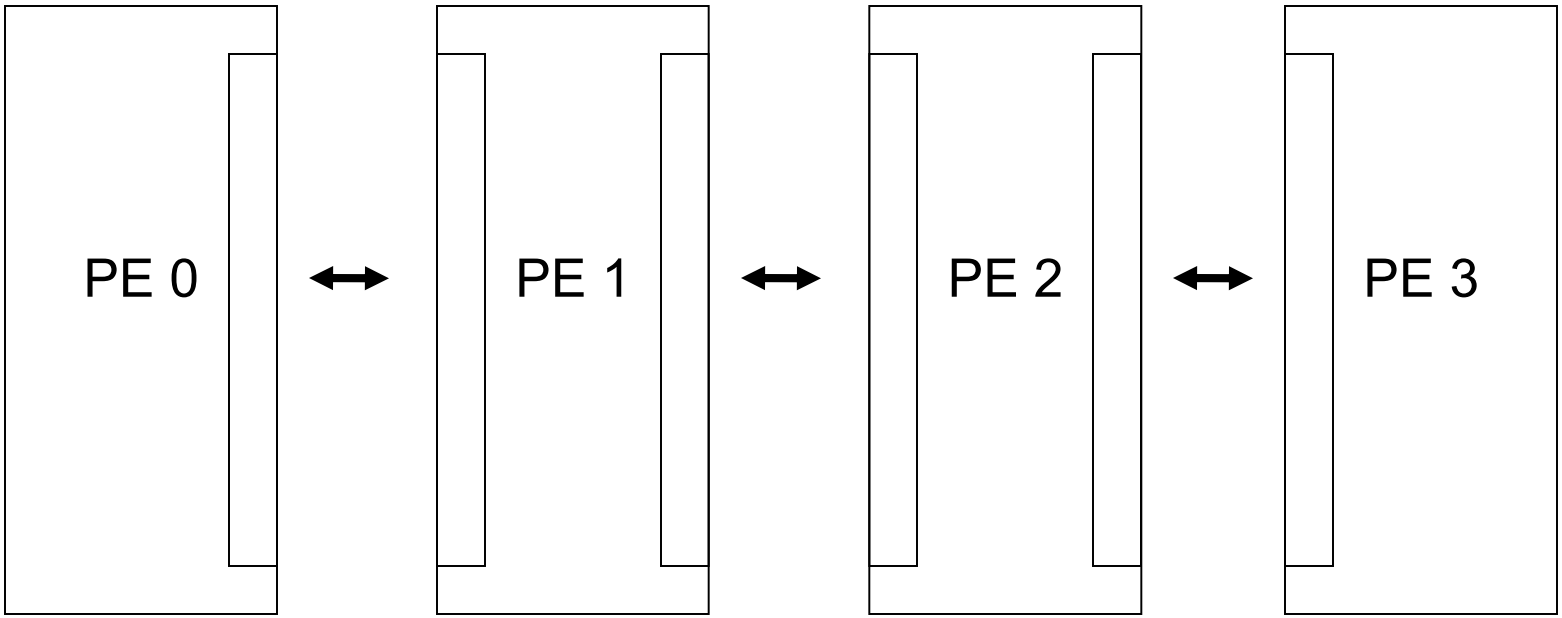
Parallelization Strategy – Domain Decomposition

Break the global grid into subgrids



The outer rectangles accommodate boundary and “ghost” zones.

Exchange Edge Data at Each Iteration



Summary

- Simultaneous use of multiple compute resources.
- Parallelism can be coarse- or fine-grained.
- Saves wall-clock time, solves bigger problems
- Make sure serial program optimized before parallelizing it.
- Effective parallelization generally requires significant effort on part of the programmer.