Gyrokinetic PIC Simulation on Massively Parallel Processors

Timothy J. Williams[†]
Andris Dimits[‡]

[†]NATIONAL ENERGY RESEARCH SUPERCOMPUTER CENTER
[‡]LLNL MFE THEORY AND COMPUTATIONS

Overview

- Particle-in-cell method
- Gyrokinetic PIC
- Early parallel programming
- Message-passing 3D slab-geometry code
- New effort on toroidal flux-tube geometry
- Parallel I/O

Particle In Cell (PIC) Simulation of Plasmas

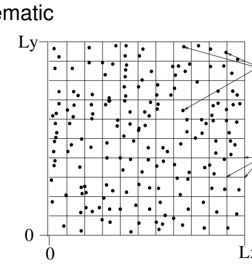
- Model plasma with simulation particles carrying charge, mass, and velocity
  - ❍ each represents many ions/electrons
  - ❍ equivalently, each represents piece of distribution $f(\mathbf{x}, \mathbf{v})$; PIC is method to solve kinetic equation for f
  - ❍ Evolved via Newton's equations

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \quad , \quad m\frac{d\mathbf{v}}{dt} = \mathbf{F}(\mathbf{x}, t) + q\mathbf{E}(\mathbf{x}, t) + q(\mathbf{v} \times \mathbf{B}(\mathbf{x}, t))$$

- Model electromagnetic fields by discretizing spatial domain
  - ❍ Evolved via Maxwell's equations (electrostatic shown:)

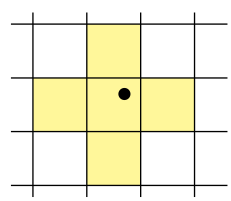
$$\mathbf{E}(\mathbf{x}, t) = -\nabla\phi(\mathbf{x}, t) \quad , \quad \nabla^2\phi(\mathbf{x}, t) = -4\pi\rho_{ch}(\mathbf{x}, t)$$

- 2D PIC Schematic



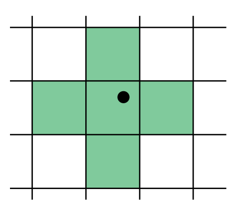
Gathers and Scatters in PIC

- Get force at particle's position field grid values
  - ❍ computationally: GATHER



Interpolate EM field values to particle position weighted values at set of neighbor cells.

- Get charge density by accumulating charge from particles' positions to grid
  - ❍ computationally: SCATTER–ADD



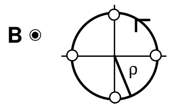
Particle contributes weighted fraction of charge to ρ_{ch} in set of neighbor cells.

Gyrokinetic PIC

- Gyrokinetic ordering

$$\left\{\frac{\omega}{\Omega}, \frac{\rho}{L}, \frac{k_\parallel}{k_\perp}, \frac{e\phi}{T}\right\} = O(\varepsilon) \quad , \quad k_\perp = O(1)$$

- Formal transformation of coordinates to eliminate gyro-orbit motion to any desired order in ε

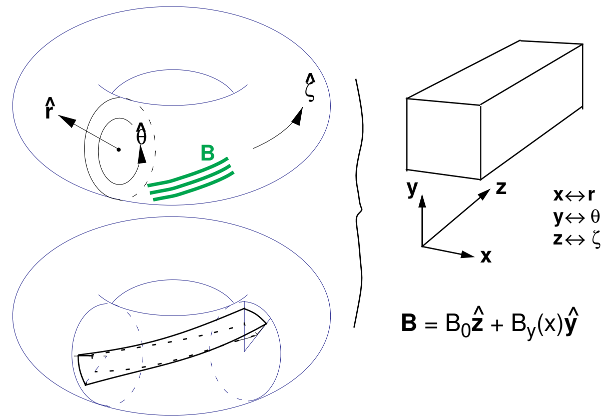


  - ❍ Modified, gyrokinetic particle equations of motion
    - ■ Describe motion of gyrocenters
  - ❍ Modified, gyrokinetic EM field equation
- Usefulness: low-frequency tokamak microinstabilities (Numerical Tokamak Project (NTP))

The δf Method

- Model only perturbation of $f(\mathbf{x}, \mathbf{v})$ from fixed background $(f \equiv f_0 + \delta f)$
  - ❍ Greatly reduces noise

Slab Geometry for Tokamak Core



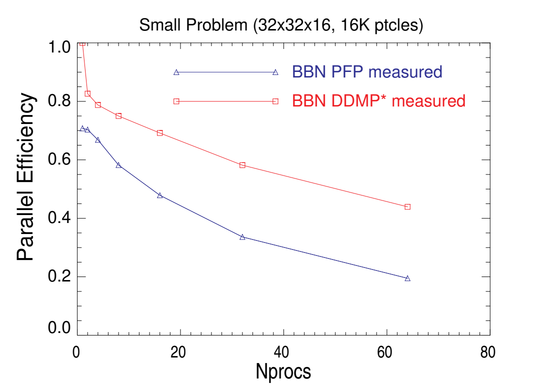
- Electrostatic $\rightarrow$ gyrokinetic Poisson on grid
- Starting point: 3D electrostatic C90 code (fully vectorized)
  - ❍ Adiabatic electron response
  - ❍ 4-point gyroaveraging
  - ❍ CIC interpolation

The Need For Speed

- A goal of NTP is understanding turbulent transport (and subsequent loss) of energy and particles in the core of tokamaks, believed caused by plasma microinstabilities
- Small production runs
  - ❍ Study effects of $\mathbf{E} \times \mathbf{B}$ velocity shear on a microinstability
  - ❍ 32x32x16 grid
    606797 particles
    5000 timesteps (nonlinear saturation)
  - ❍ O(10) hours on C90 (1 CPU)
    - ■ 114 MFLOPS, 110 are vector ops
    - ■ average vector length 91.24
- Desired production runs, est. from NTP proposal parameters
  - ❍ moderately large tokamak
  - ❍ run to transport time (100-1000x nonlinear saturation time)
  - ❍ $O(10^4)$ C90 hours
- SPP allocation on C90 is not enough
- Not enough C90 memory for some near-term runs

Early Parallel Programming Efforts

- Control parallel (PFP code on BBN TC2000)
  - ❍ Shared interleaved storage of grid arrays
  - ❍ Lock-protected parallel charge accumulation

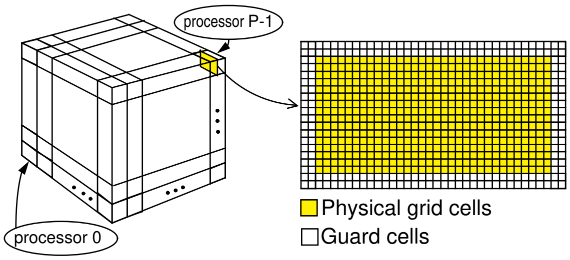


*DDMP = Domain Decomposition and Message Passing

- Data parallel (CM Fortran code on CM200)
  - ❍ GATHER *communication* is 30-40% of execution time
  - ❍ SCATTER-ADD *communication* is 20-30% of execution time

Domain Decomposition and Message Passing

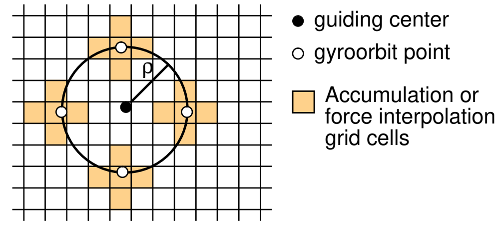
- Justification for choosing this paradigm
  - Generic
    - Message-passing calls small part of code
    - All message-passing libraries have same basic functions
  - Maximal data locality $\Rightarrow$ minimal communication
- 3D primary domain decomposition
  - 3D $\Rightarrow$ *scalable* to maximal #procs. (fixed problem size)
  - Decompose field grid arrays
  - Processors work on particles in their subdomains



  - Send particles leaving your subdomain to neighbor PEs

Guard Layers

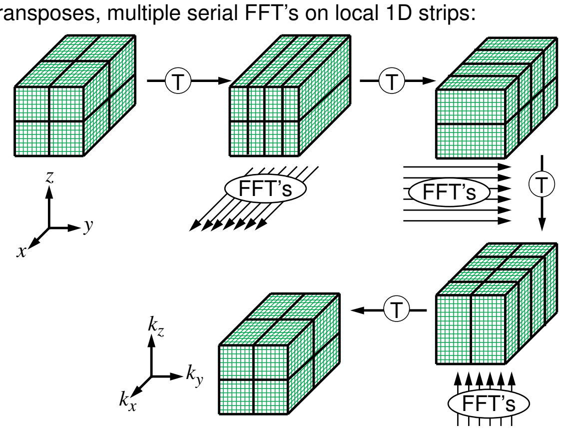
- *Gyrokinetic* PIC makes special demands on guard layers in the dimensions perpendicular to $\mathbf{B}$
  - Gyroaveraging
    - Charge accumulation around gyroorbit
    - Force interpolation to gyrocenter
    - Typically sample 4 points on orbit:



    - ρ varies from particle to particle
    - max ρ may be 4 or 5 grid cells in typical simulations
    - number of guard cells in these 2 directions known only after initializing particles (depends on temperature)
      - must span ρ_{max} + number of cells in interpolation

FFT for Poisson Solve

- Transposes, multiple serial FFT's on local 1D strips:



Outline of a Timestep

1. Charge accumulation [+ border data comm.]
2. Field solve
   - FFT $x \rightarrow k$
   - Poisson solve in k space
   - FFT $k \rightarrow x$
   - [Field border data comm.]
3. Predictor push
4. Charge accumulation
5. Field solve
6. Corrector push [+ particle-passing comm.]

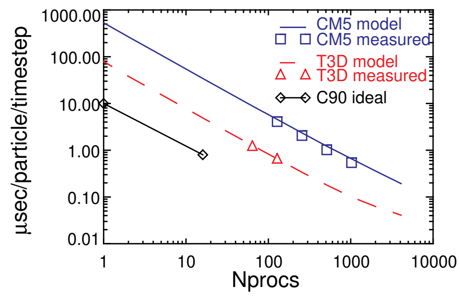
Performance Model

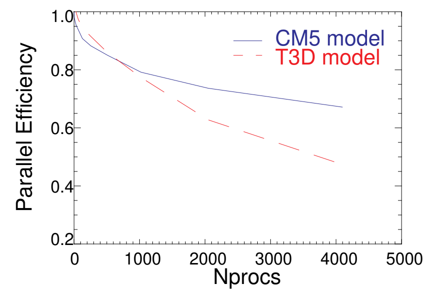
Adjustable parameters in model:

- ❍ $t_f \equiv$ time for one 64-bit binary FLOP
- ❍ $t_{ss} \equiv$ startup time to send message (latency)
- ❍ $t_{sr} \equiv$ startup time to receive message
- ❍ $t_1 \equiv$ time to send 1 64-bit word

- ● Determine "t" parameters empirically for each computer
- ● Count/estimate/measure FLOP's in code
- ● The model:
  Analytical expression in these parameters, processor counts, and problem size parameters

Performance Predictions and Measurements



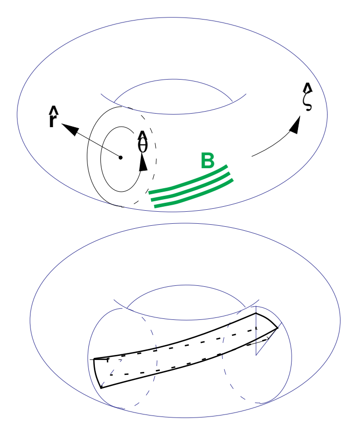


- ● 256x256x32 grid, 8M particles (8 per cell)
- ● Parallel efficiency $\varepsilon = \frac{1}{P}\frac{T_1}{T_P}$; T_1 time for optimal serial algorithm used on 1 processor
- ● C90: combine *best ever* (for *any* problem) time & overlap

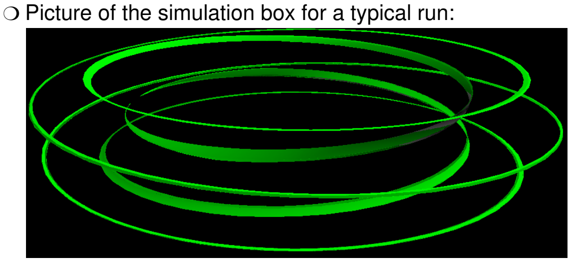
Nice Work! Too Bad It's Irrelevant Now.

- ● Parallel Processing:
  - ❍ 3D slab code $\rightarrow$ Williams $\rightarrow$ parallel 3D slab code
  - ❍ 3D slab code $\rightarrow$ Dimits $\rightarrow$ 3D toroidal flux tube code

Flux Tube Geometry

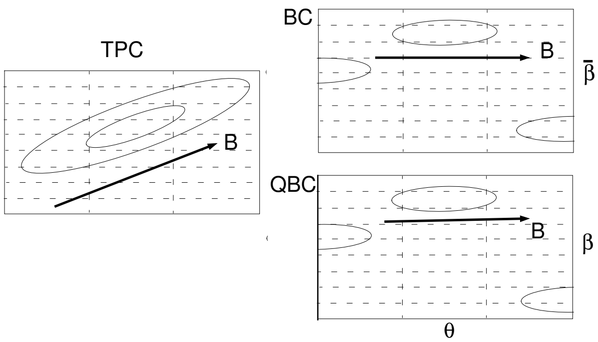


- ● Large-aspect-ratio, circular-cross-section, toroidal flux-tube geometry
  - ❍ Picture of the simulation box for a typical run:



Quasiballooning Coordinates

- Toroidal-poloidal coordinates (r, θ, ζ) .
- Ballooning coordinates $(r, \theta, \bar{\beta})$
  - ❍ $\bar{\beta} = \zeta - q(r)\theta$
- Quasiballooning coordinates (r, θ, β)
  - ❍ $\beta = \zeta - \hat{q}(r)\theta$
    - ■ At each grid surface r_i, choose integer m_i s.t. $\hat{q}(r_i) \equiv \dfrac{m_i}{N_\beta}$
    - ■ Resolution advantage of quasiballooning coordinates:

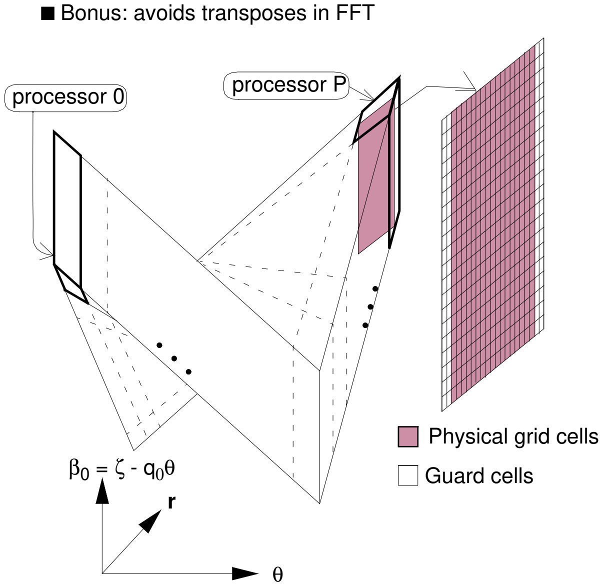


Numerical Methods for QBC

- Differences from slab-geometry methods
- Field equation solved in (r, k_θ, θ)
  - ❍ Complex periodic-tridiagonal solve in r for each (k_β, θ)
  - ❍ Smoothing
    - ■ FIR $\rightarrow$ complex periodic-tridiagonals in r and θ for each k_β
    - ■ k_β form factors
  - ❍ All matrices set by magnetic geometry: constant
- Periodic boundary conditions:
  - ❍ Periodic in β
  - ❍ Standard toroidal periodicity in θ:
    $\psi(r, \theta + 2\pi, \beta - 2\pi q) = \psi(r, \theta, \beta)$
  - ❍ Radial periodicity with a twist so that $\mathbf{B}$ appears continuous across radial boundary:
    $\psi(r + L_r, \theta, \beta) = \psi(r, \theta, \beta)$
    - ■ Toroidal and radial periodicity consistent: Choose L_r so that $[q(r + L_r) - q(r)] = \dfrac{M}{N_\beta}$

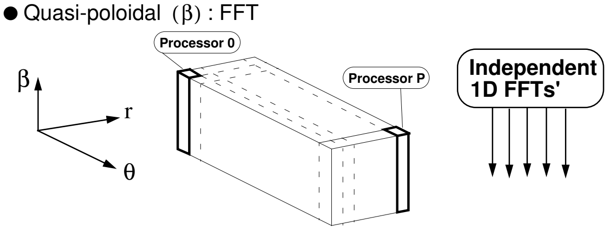
Domain Decomposition for Flux Tube

- 2D primary domain decomposition (was 3D in slab)
  - ❍ Single processor in β direction $\Rightarrow$
    - ■ Avoids complicated particle passing at box ends
    - ■ Bonus: avoids transposes in FFT



Parallel Field Solution

- Gyrokinetic Poisson equation
- Quasi-poloidal (β) : FFT



  - ❍ In place
  - ❍ No communication
- Radial (r) : finite difference $\rightarrow$ periodic tridiagonal
  - ❍ Parallel solver of Mattor, Williams, Hewett, Dimits
  - ❍ Branch to serial algorithm when P_r is one
  - ❍ Parallel algorithm $\approx O\left(17\dfrac{N_r}{P_r}\right)$ operations
    Serial algorithm $\approx O(15N_r)$ operations
    - ■ Asymptotic parallel efficiency 88%

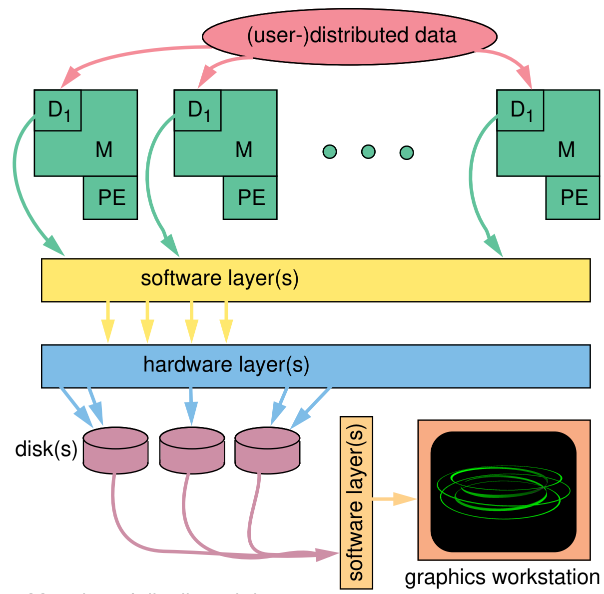
How Much Pain to Face the Present?

3D slab $\rightarrow$ 3D toroidal flux tube

- Re-used from slab code:
  - ❍ Message-passing wrapper library MESS
  - ❍ Calibrated timer procedures
  - ❍ Performance model
    - ■ Mostly stripped down from 3D slab
    - ■ Same machine parameters $\{t_f, t_{ss}, t_{sr}, t_1\}$
  - ❍ 2D DD special case of slab's 3D DD
  - ❍ 1D FFT easy strip-down from slab's 3D FFT code
- Re-used from parallel DADI work: tridiagonal solver

- Pain:
  - ❍ QBC code computes globally and β-θ averaged fields $\Rightarrow$ communication
    - ■ $\log_2(P)$ -stage data exchange from parallel tridiagonal solver re-used

Parallel I/O



- Meaning of distributed data:
  - ❍ Programmer/compiler has *global view* of distributed data
    - ■ 64x64x128 float array distributed block-block-block
    - ■ 3D scalar field on regular mesh: electrostatic potential

Parallel I/O (cont'd)

- I/O Specification
  - ❍ Scientist-programmer sees: High-level scientific I/O library
    - ■ "Function calls" express global view of data, local ownership of pieces, type of data
    - ■ Highest level: ND field in MD space
    - ■ Self-describing, machine--independent "files"
    - ■ Examples: HDF, NetCDF (none general enough!)
  - ❍ Systems programmer sees: Parallel I/O library
    - ■ Takes multiple streams of bytes
      - ❑ Maybe gathered/strided from memory
    - ■ Outputs (multiple) streams of bytes in global-view to disk
      - ❑ Maybe *not* programming-language storage order
      - ❑ Maybe *not* single traditional file
      - ❑ Examples: MPI-IO, Vesta

...None of this really exists

- Kim Yates working on MPI-IO for the T3D

NetCDF Library for I/O

- Public domain (Unidata (related to UCAR/NCAR))
- Data files
  - ❍ Machine independent format (XDR)
  - ❍ Binary
  - ❍ Random-access
  - ❍ Self-describing
- Based on multidimensional arrays of floats/ints/bytes/....
  - ❍ *Unlimited* dimension allows time histories
  - ❍ Functions to read/write *hyperslabs* of arrays (subrange of each index)
  - ❍ Multiple variables of multiple types per file
  - ❍ Provision for tagging units, *etc.*
- Understood by a number of postprocessing packages
  - ❍ AVS
  - ❍ IDL
  - ❍ DDI (C.Anderson, NERSC)
- ASCII dump utility
- "Agreed upon" for testing by NTP for data exchange

NetCDF for Parallel I/O

- Each PE writes its subdomain as hyperslab
  - NetCDF output array specified with global array dims.
  - At least one disk seek per PE
- CM5 performance (Rick Light's optimized library)
  - Dump 3D 64-bit array 32x32x16 (potential from PIC code)
    - 3D domain decomposition
    - Measured 1-2 MB/s using 1-128 PEs
      - Adequately small impact on run time
- T3D performance (Steve Luzmoor's optimizations)
  - In progress; early indications are *O*(10) MB/s