Gyrokinetic Toroidal Code

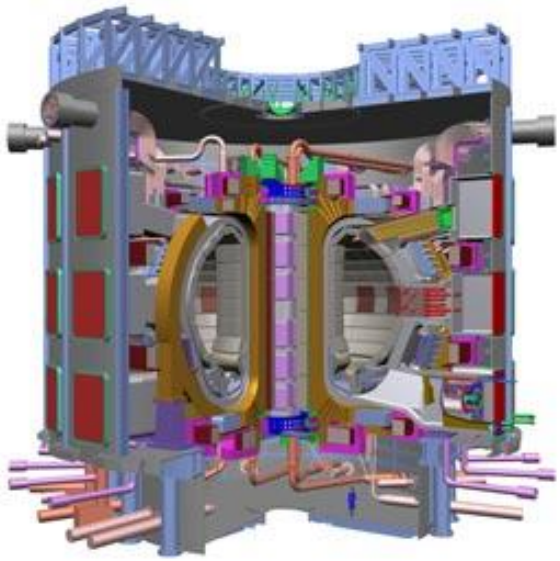
Computational Overview

18 July 2011 – CScADS

Daniel Fulton
University of California, Irvine

Gyrokinetic Toroidal Code (GTC): Basics

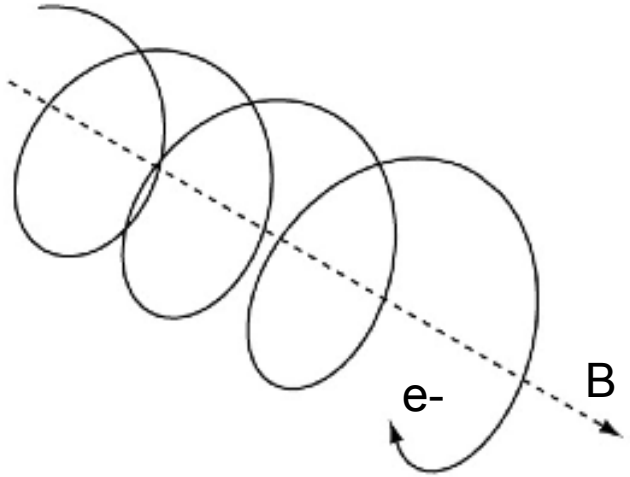
- Goals:
  - Simulate turbulence, transport, instabilities, in fusion plasmas.
  - Support burning plasma experiments (e.g. ITER)

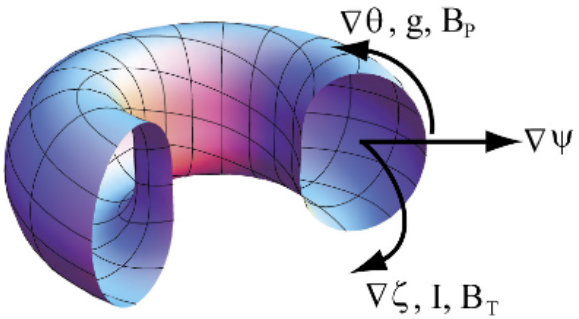


- Group: group leader Zhihong Lin + 2 researchers, 1 post doc, 5 graduate students
- Originally developed by Zhihong Lin at PPPL, now maintained at UC Irvine.

GTC: Some Details

- "Gyrokinetic"
  - Charged particles travel along magnetic field lines in helix shape.
  - Assume "gyro" motion is much faster and can be averaged out.
  - Still accounts for finite radius.



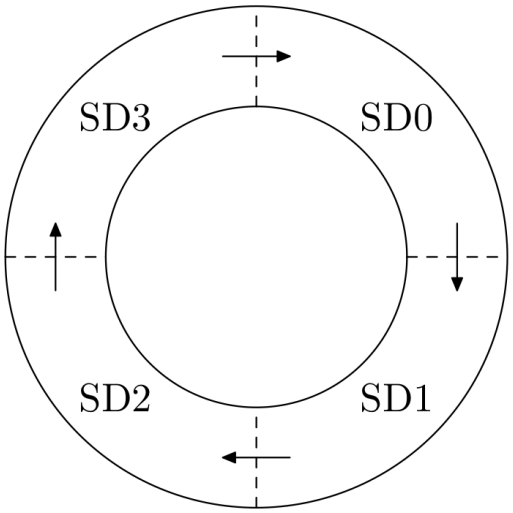


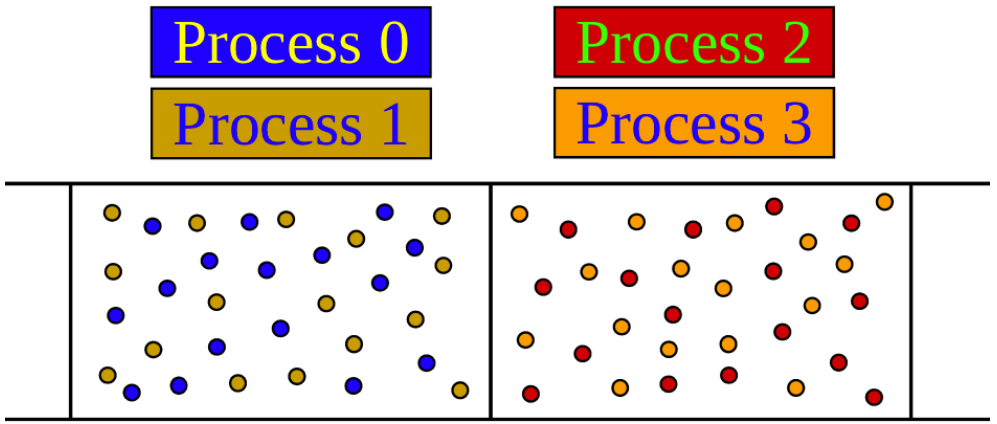
- "Toroidal"
  - Usually, donut shaped spatial domain (most fusion experiments)
  - Has been modified for cylindrical geometry, as well.

- PIC code
- MHD + kinetic corrections…

GTC: Programming Model (i)

- Hybrid OpenMP + MPI code.
  - MPI breaks up spatial domains and groups of particles within these domains.
  - OMP for parallelizing loops/individual particles w/i MPI task.





GTC: Programming Model (ii)

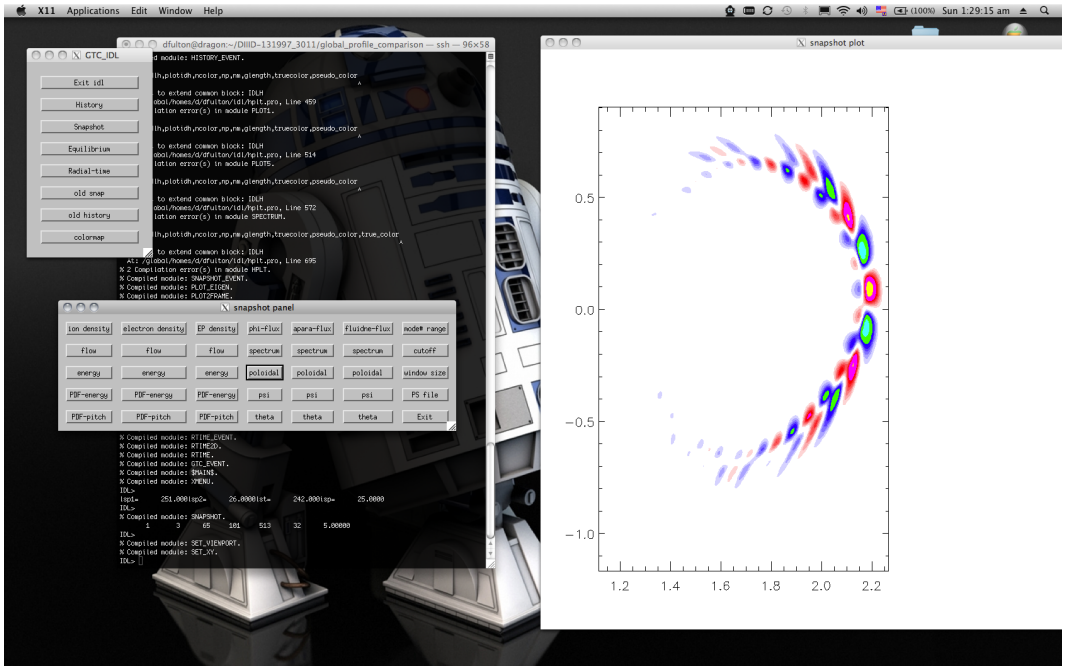
- Fortran90/95 free format
- Runtime libraries:
  - PETSC for poisson solver (one implementation)
  - NetCDF for output, but not required (see next).
- Typically compile with pgf90.
- Runs on jaguarpf, hopper, tianhe-1A…
- In the next year + GPU.
  - Fortran CUDA/OpenCL?

GTC: I/O

- I
  - Small plain text file with input parameters.
  - Requires magnetic equilbrium
    - Analytical – specified by the code
    - Numerical – Experimental or generated by another eq solver. Plain text typically ~ 50MB.
- O
  - 3D requires NetCDF. File size???
  - Produces separate plain text files for various 1D/2D data. Order 10-100+MB total. Depends on grid.
- Snapshot/restarts at user specified interval.
- Currently, no major plans to change I/O.

GTC: Analysis

- IDL GUI for common plots.
- Individuals use other tools for specifics (IDL, Matlab, gnuplot).

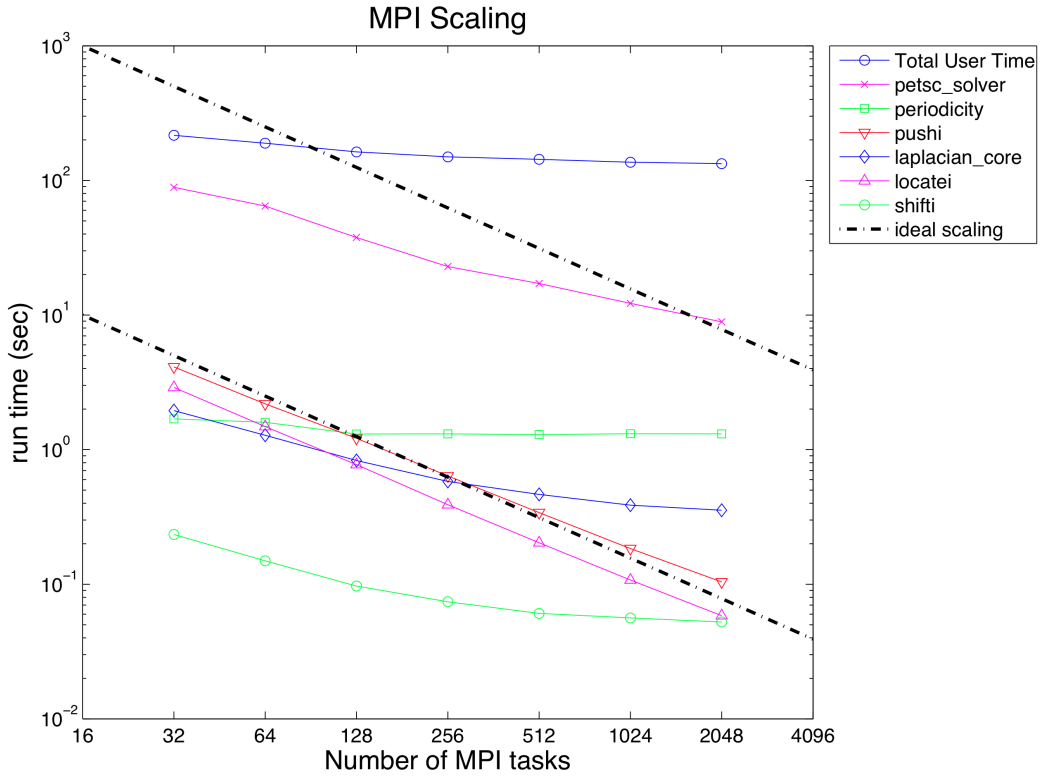


GTC: Performance

- Current performance analysis:
  - By hand scaling studies.
  - Just tried CrayPAT to make this presentation!
- Downside: hard to identify places to improve performance that are not related to scaling.
- Currently: Know fluid algorithms are implemented only in simple way. Would like to refine, but still need to identify best candidates for optimization!
- Approach so far is to run performance analysis on ideal ballooning mode (IBM) case, which involves only fluid parts.

GTC: Scalability (i)

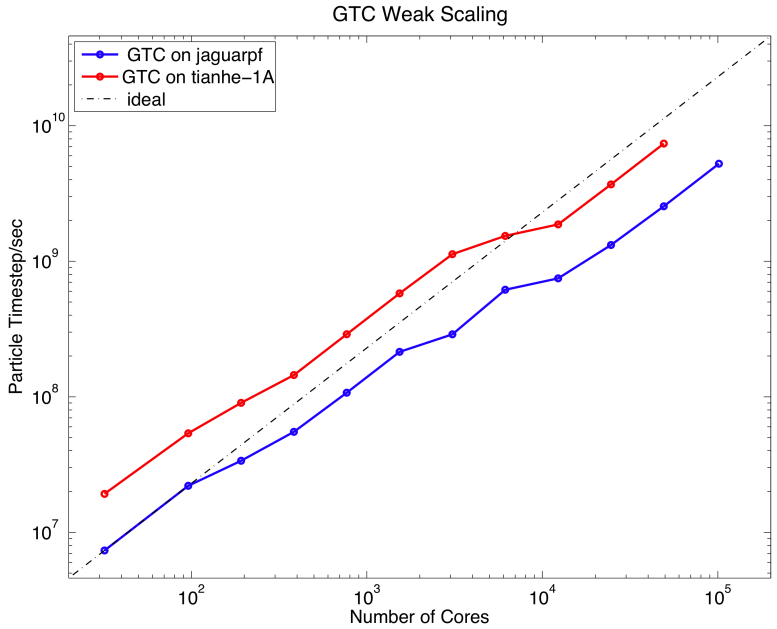
- MPI scaling limited by grid size/# of particles.



- Difficulty: GTC runs require minimum of 32 MPI tasks, so OMP imbalance can be ‘lost’ in MPI imbalance.

GTC: Scalability (ii)

- Overall, electrostatic scaling is good.
- Difficulty:
  For electromagnetic, PETSC not yet compatible with OMP.



- Current scalability achieved by going from MPI → hybrid MPI + OMP.
- Plan to improve scaling with addition of GPU.

GTC: Debugging Tools

- Debugging methods:
  - Version ctrl: compare to old (working) code
  - Cleverness + write statements ☺
  - Brute force + write statements ☹
- Have tinkered with DDT, but has some bugs with PGI fortran.
- No plans to change current methods, but would be nice.

GTC: Roadmap

- Explore kinetic/nonlinear instabilities, where linear analytical techniques do not apply. *New* physics, not old physics.
- Refine GTC fluid algorithm.
  - Numerical stability/accuracy
  - Performance
- Add GPU code for higher order kinetics/ electron subcycles, and to be able to use new machines to full potential!