Proxy Design and Optimization in Fusion and Accelerator Physics

Alice Koniges

Lawrence Berkeley National Laboratory/NERSC

with selected results from some members of the

NERSC Petascale Postdoc Program

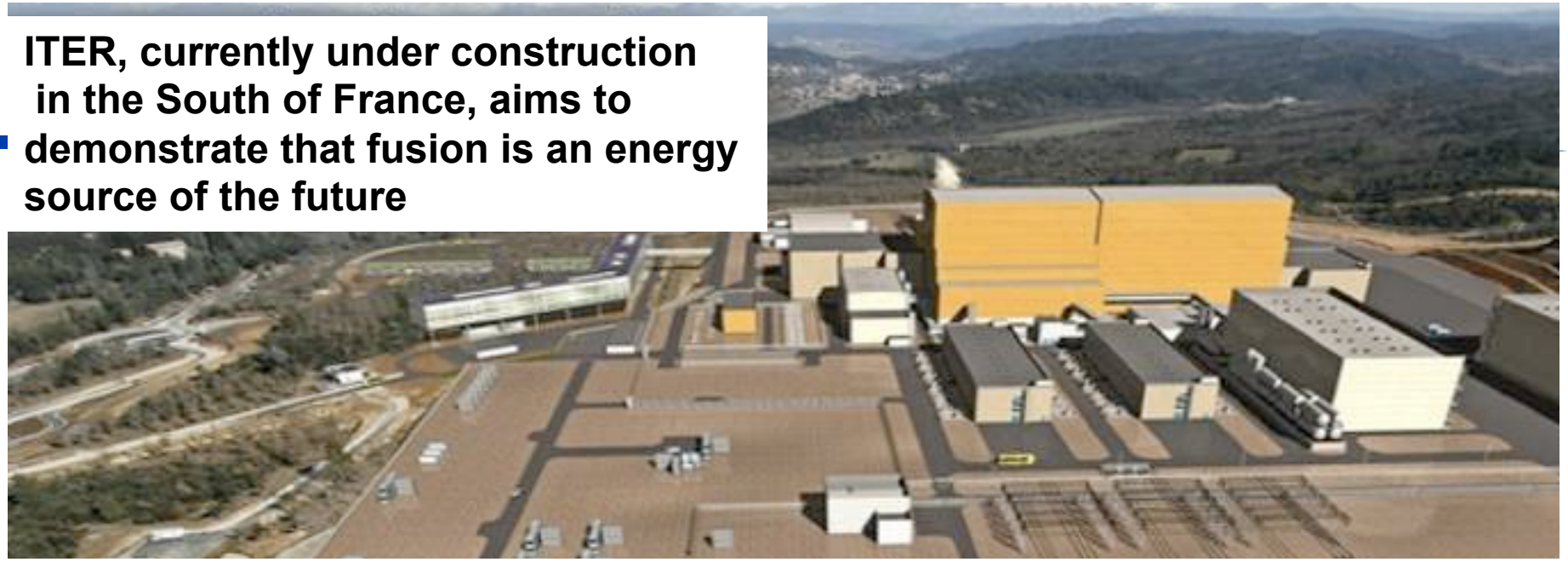
Praveen Narayanan, Robert Preissl, Xuefei Yuan

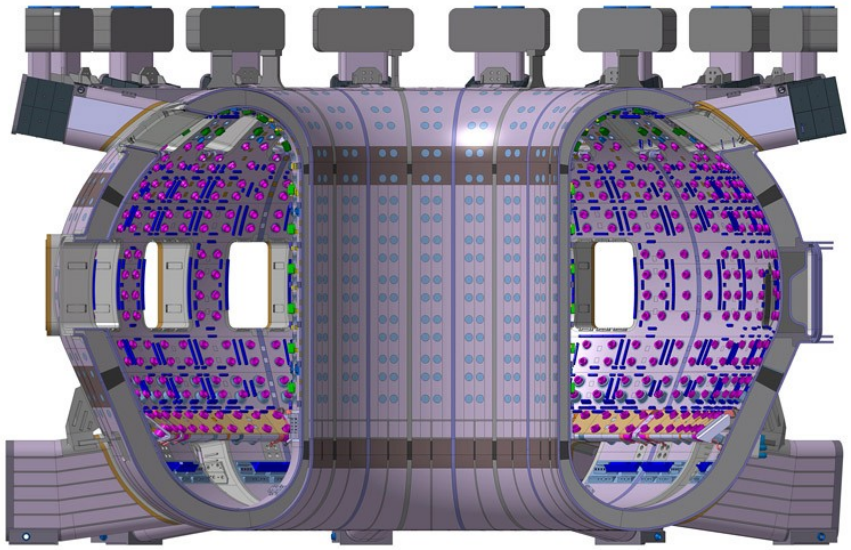
SIAM Conference on Computational Science and Engineering, Boston Feb. 25, 2013

Outline

- **Intro to High Performance Computing Challenge areas for Plasma Physics**
  - **Magnetic Fusion Energy: Tokamak or other magnetically contained plasma for fusion energy (generally toroidally shaped)**
  - **Inertial Fusion Energy: Create fusion reaction using lasers and ion beams**
  - **Accelerators**
    - **As Free-Electron Laser sources for production of very short wavelength light sources—creates 3D images of very small things**
    - **As an Ion Beam Accelerator to study Warm Dense Matter and properties of materials**
- **Primary Proxy App: PIC (Particle-In-Cell) Simulations**
- **Other Areas for Study:**
  - **Solvers for MHD and Solvers within PIC codes**
  - **Behavior of plasma turbulence codes**

ITER, currently under construction in the South of France, aims to demonstrate that fusion is an energy source of the future

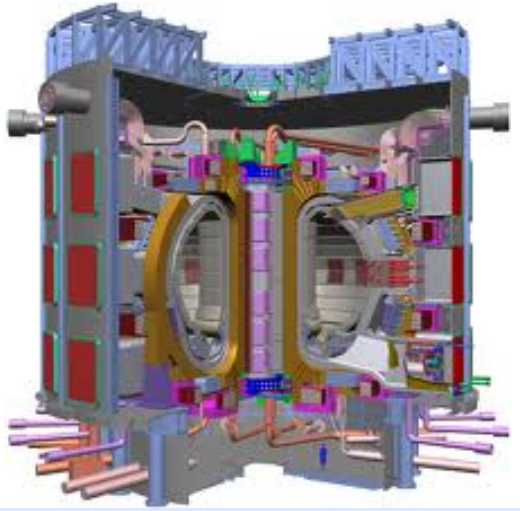




- **Top-to-bottom exascale computer design is essential for efficient design/operation of large-scale experiments**
  - Typical ITER discharge can be estimated at 1M$

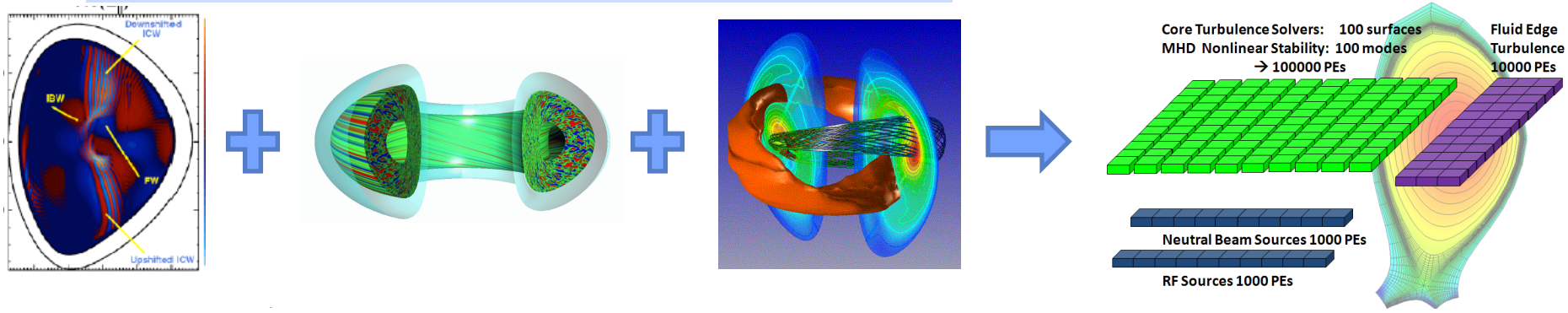
A variety of Fusion apps are required for building and running the ITER

- Fusion program has large suite of petascale applications in use covering many spatial and temporal scales
- The fusion suite of parallel applications brings a wide array of algorithms (implicit, nonlinear fluid, PIC, continuum phase space)



ITER: $10B Reactor

Coupled code set diagram for magnetic fusion



Ω_{ce}^{-1} ω_{LH}^{-1} Ω_{ci}^{-1} τ_A

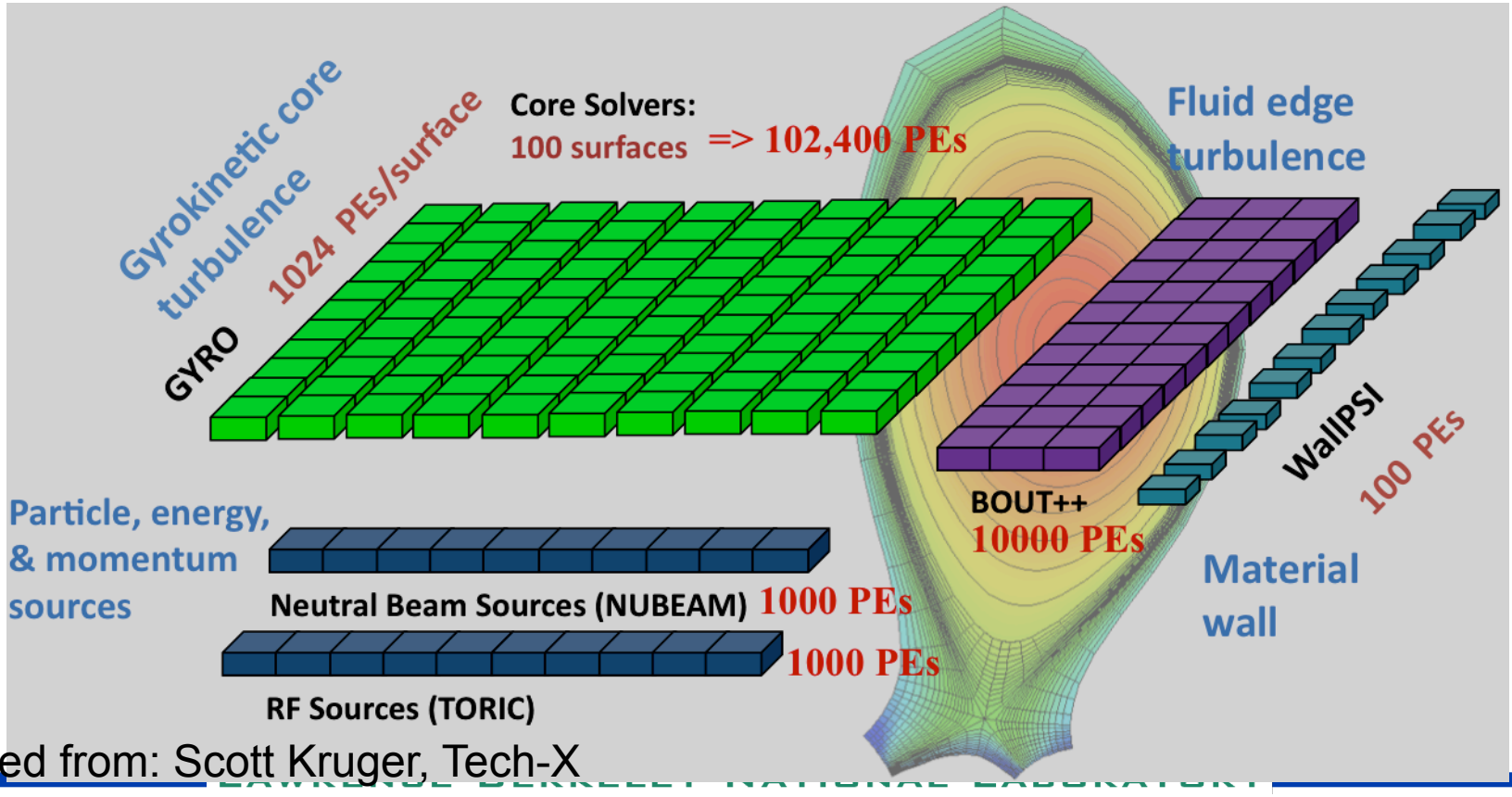
10^{-10} 10^{-8} 10^{-6} 10^{-4} 10^{-2} 10^{0} 10^{2} 10^{4}

Discharge time ~hour

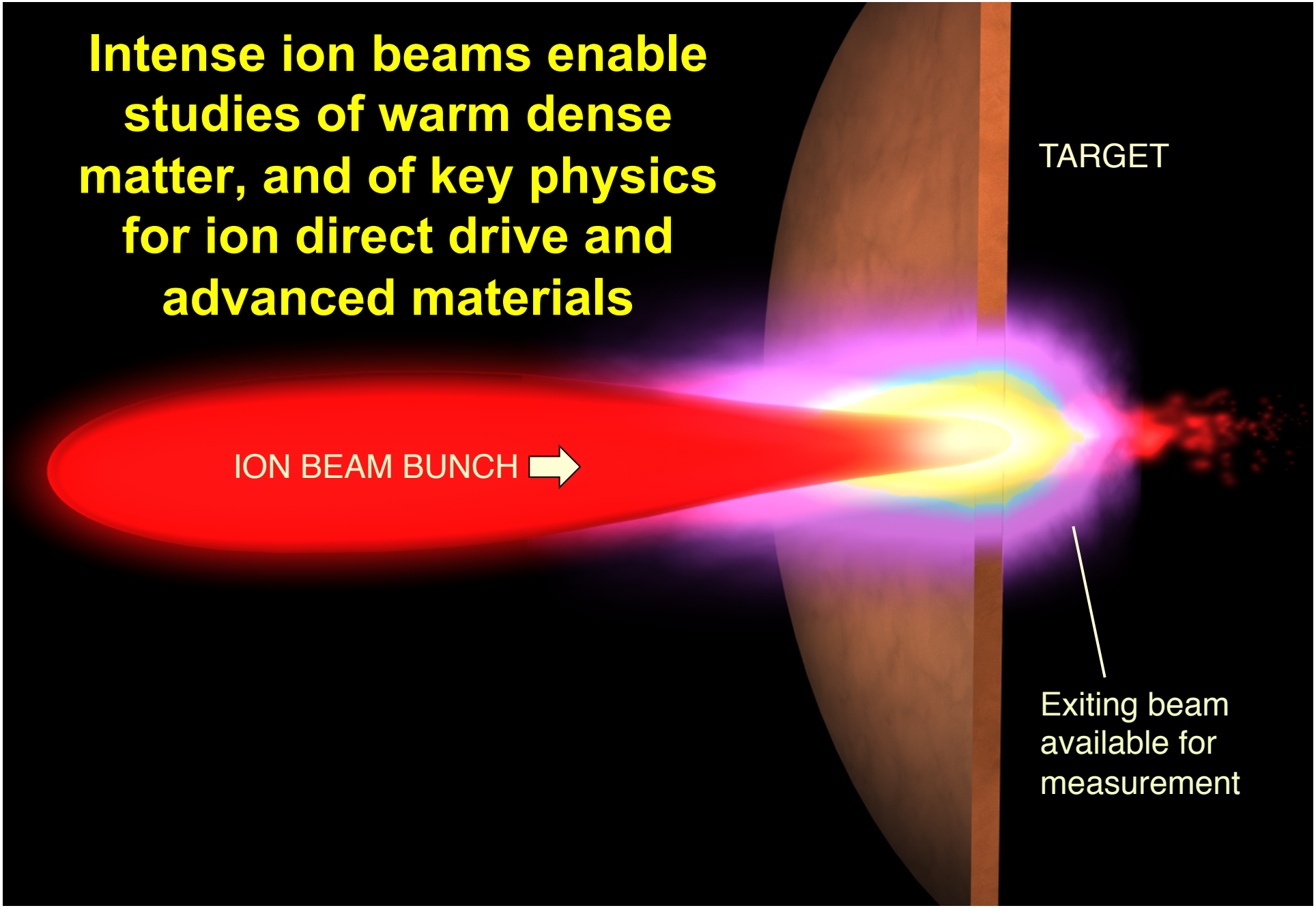
Time in seconds for full-scale magnetic fusion interactions

Next Generation Fusion Modeling hopes to couple the various codes

- Core Transport: GYRO/NEO
- Collisional Edge Plasma: BOUT++
- MHD: M3D-C1, NIMROD
- Explicit PIC Modeling: GTS, VORPAL
- Wave heating, Wall interaction

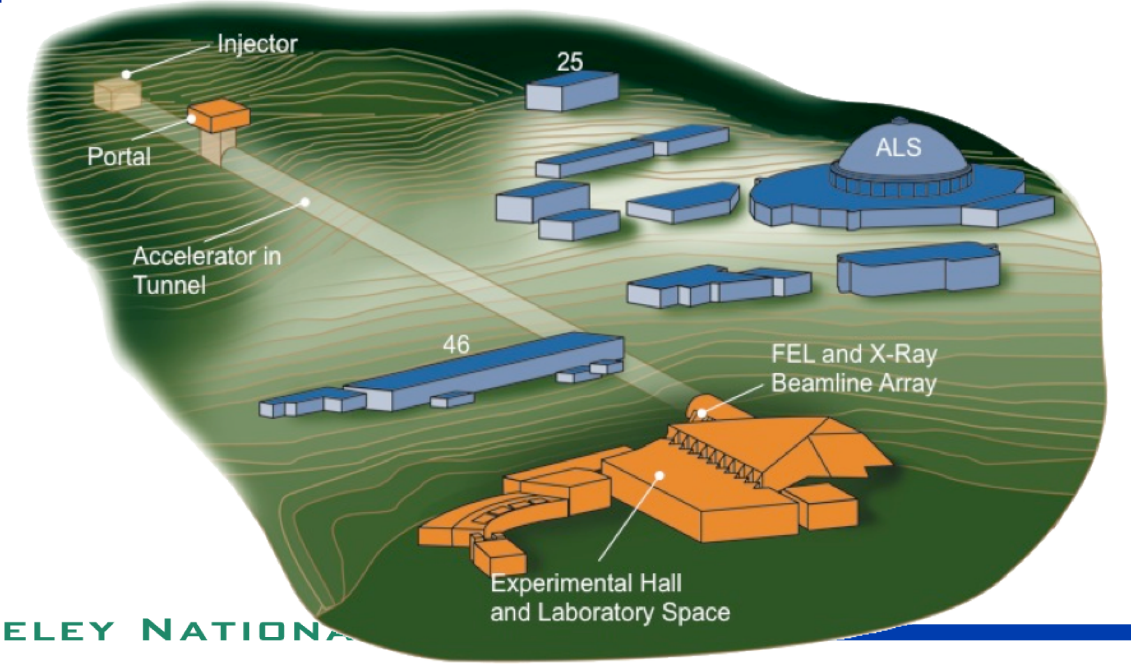


Adapted from: Scott Kruger, Tech-X


Intense ion beams enable studies of warm dense matter, and of key physics for ion direct drive and advanced materials
TARGET
ION BEAM BUNCH
Exiting beam available for measurement

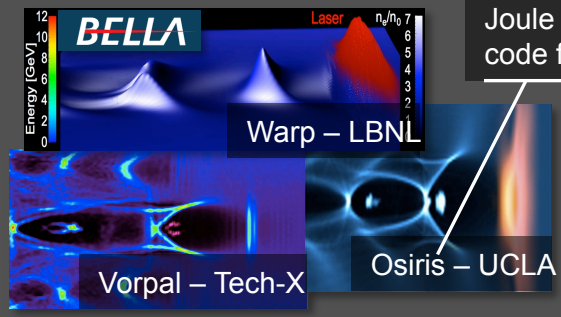
Next Generation Light Sources are being designed with HPC Codes

- X-Ray Free Electron Laser
  - Tunable soft X-ray source (0.1-1.2 keV)
    - Core electron spectroscopy
    - Nanoscale diffraction
  - Short pulse duration (250 as – 500 fs)
    - Ultrafast dynamics
  - Longitudinal and transverse coherence
    - Reduces time to acquire & process diffraction data



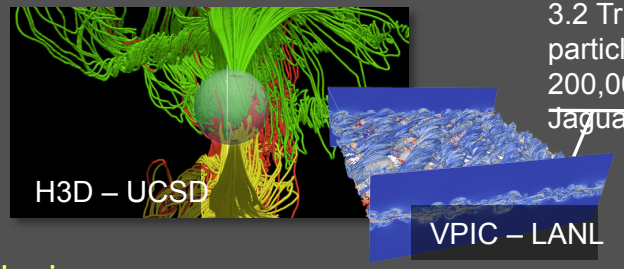
PIC Proxies have applications to electromagnetic kinetic modeling of space & laboratory plasmas

Laser plasma acceleration



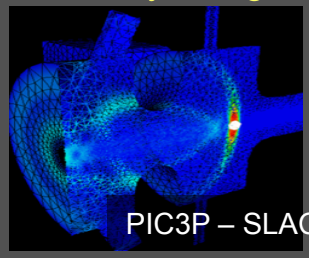
Joule metric code for ASCR

Solar storms/Magnetic reconnection



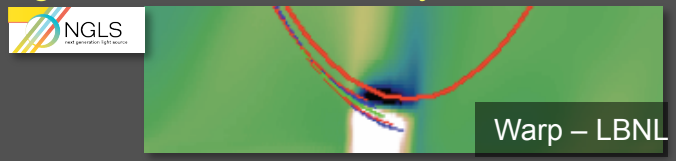
3.2 Trillion particles on 200,000 cores on Jaguar

RF cavity design

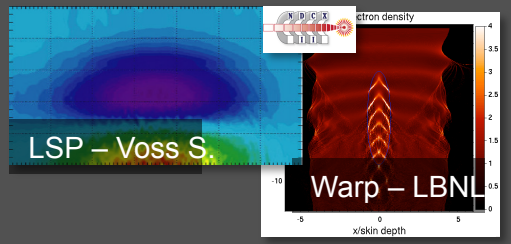


Astrophysical shocks

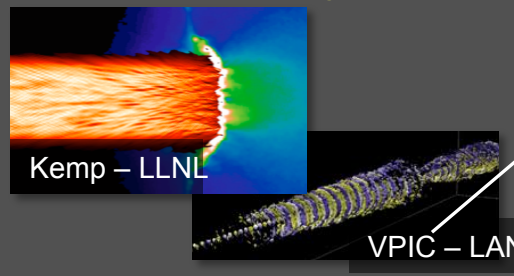
Light sources/Coherent synchrotron radiation



Beam plasma neutralization



Inertial fusion/fast ignition



First at Petascale on Roadrunner

Outline

- Intro to High Performance Computing Challenge areas for Plasma Physics
  - Magnetic Fusion Energy: Tokamak or other magnetically contained plasma for fusion energy (generally toroidally shaped)
  - Inertial Fusion Energy: Create fusion reaction using lasers and ion beams
  - Accelerators
    - As Free-Electron Laser sources for production of very short wavelength light sources—creates 3D images of very small things
    - As an Ion Beam Accelerator to study Warm Dense Matter and properties of materials
- **Primary Proxy App: PIC (Particle-In-Cell) Simulations**
- **Other Areas for Study:**
  - **Solvers for MHD and Solvers within PIC codes**
  - **Behavior of plasma turbulence codes**

Many Proxy studies for fusion apps focus only on Particle-In-Cell methods

- PIC codes are generally best scaling
- Physics content of PIC codes is steadily increasing
- Coupling to MHD in MFE PIC is in progress
- Other time-scale problems are still neglected
- PIC codes also form the basis for much of accelerator physics
- Study of the general design of PIC codes can benefit both accelerator and fusion applications
- A “really nice” proxy app for fusion apps would allow for differing pieces of plug-in physics

Intelligent Design of Proxy Apps for PIC in terms of a Domain Specific Language (DSL)

- A DSL may help to generalize efforts
  - If we can design something it can save a lot of duplicative efforts and help use and maintain advanced components
- Possibilities for DSL
  - Low-level, extensions, change some or all of the MPI calls (replace with GPU?) under hood
  - A high-level construct (abstraction) for meshing or similar operations
  - Orchestrate the DSL to provide abstractions
  - Domain specific functionality should be in a human readable form at best, or a readable language (e.g., our Python example)
- Key to a good proxy app is to think in terms of DSL-like kernels that comprise the app, and how they interact
- This is not much different than traditional pie-type approach to full code profiling

Basics of PIC codes form a rational for an intelligently-designed Proxy

- "particle-in-cell" because plasma macro-quantities (number density, current density, etc.) are assigned to simulation particles
- Particles can live anywhere on the domain, but field and macro-quantities are calculated only on the mesh points
- Steps can lead to a domain specific language/concepts
  - Integration of the equations of motion
  - Interpolation of charge and current source terms to the field mesh
  - Computation of the fields on mesh points (field solve)
  - Interpolation of the fields from the mesh to the particle locations
- PIC codes differ from Molecular Dynamics in use of fields on a grid rather than direct binary interactions, goes from N^2 to N
- PIC codes are radically different from standard PDE solver codes and show real promise for the exascale

Specific Components form the basis for the Proxy App Studies

- Data Structures/Abstractions
  - Lagrangian particles: x, y, z, Vx, Vy, Vz, q, m, etc.
  - Eulerian fields and sources: Jx, Jy, Jz, Ex, Ey, Ez,
  - Bx, By, Bz on grids for electromagnetics;
  - Rho, Phi (or V) for electrostatics
- Goal – don't care where the particles live, e.g., in terms of the parallel decomposition. Want to hide this from application programmer
- Methods/Functional Abstractions
  - Push particles
  - Deposit (scatter) charge or currents from particles onto grid(s)
  - Solve fields (Can we put this into a library call?)
  - Gather forces from grids onto particles.

Questions for hiding complexity and optimizing code lie in the basic design

- Data abstraction -- Need to ask what do you want to do with the data –e.g., to push the particles. User should not care about where the particles live, what processor, etc. and need to conserve movement of data between procs
- Grid abstraction -- fields have to live on a grid. How much of grid in memory, and how is it distributed? What do you need to pull from it? Maybe don't want to replicate the grid on all procs?
- Functional abstraction --can you generate the move for a variety of different problems and let it (DSL combined with compiler) generate the code for this?

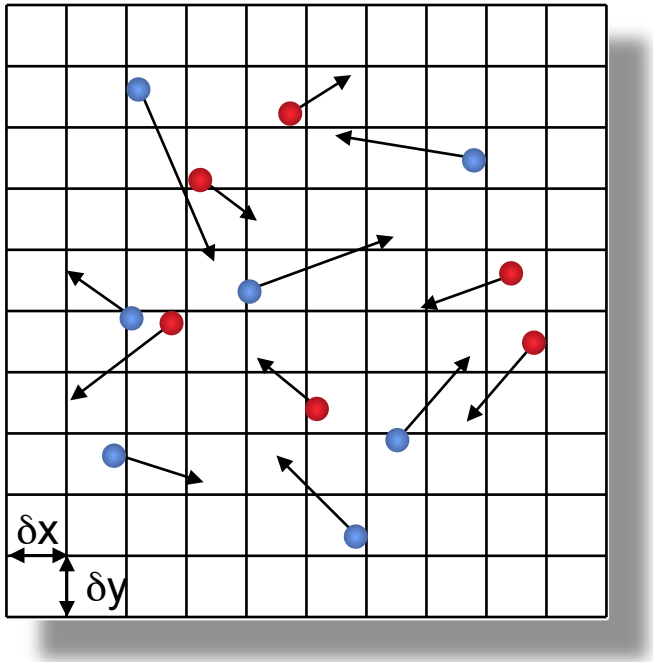
Why can it be difficult to define a PIC DSL

- **Difference in layout of Lagrangian and/or Eulerian quantities in memory**
  - **not a hard barrier per say as layers of translations (copy) between data structure can be added, but usually at the expense of runtime efficiency**
- **Legacy**
- **Competition**

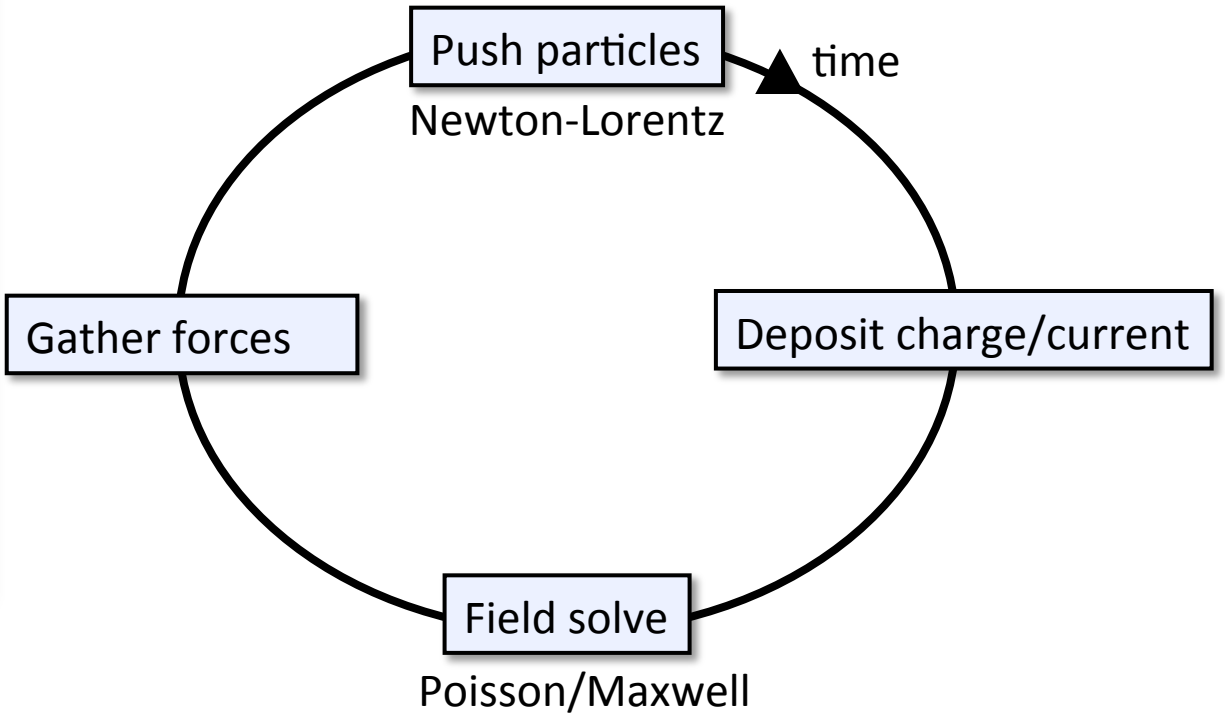
More evolved features like irregular gridding, AMR, complex particle pushers, deposition schemes, or field solvers, call for more sharing as a smaller fraction of developers can effectively maintain such codes.

basic data structures and operations are fairly simple and thus easily reproducible

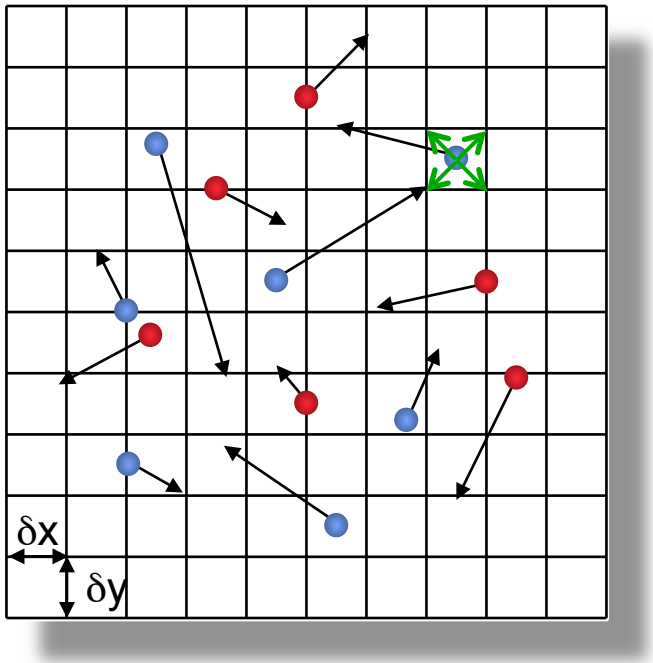
Particle-In-Cell workflow



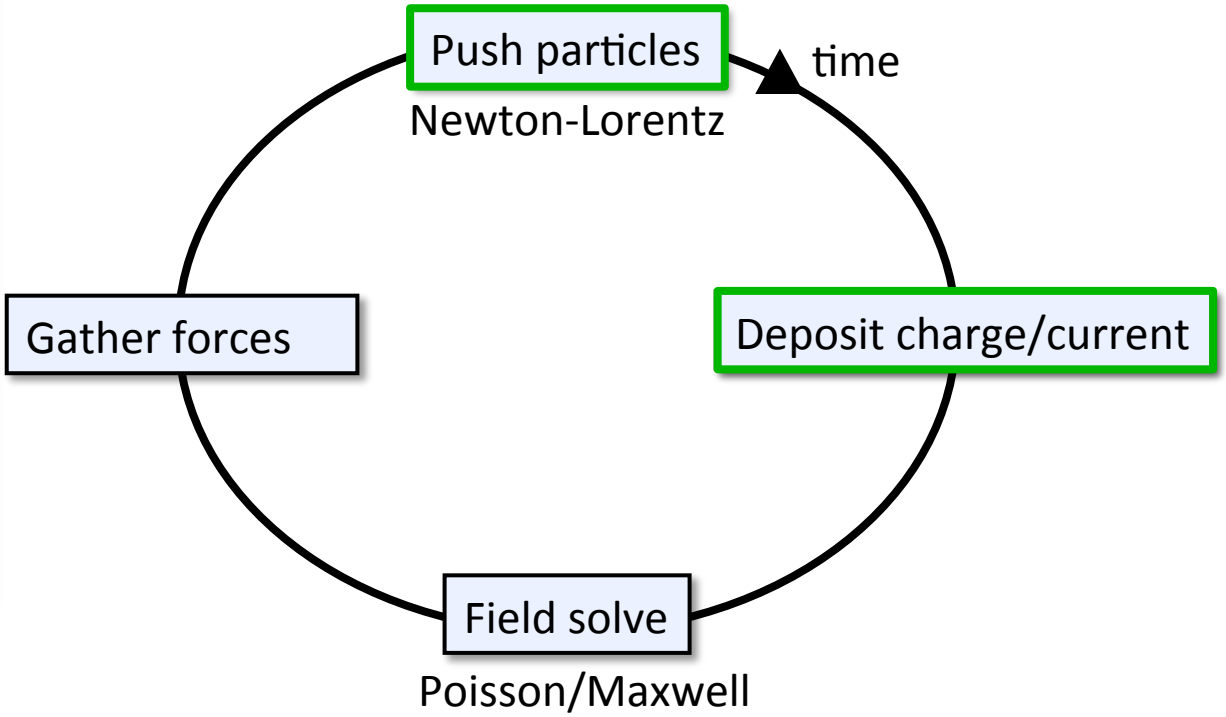
Plasma=collection of interacting charged particles



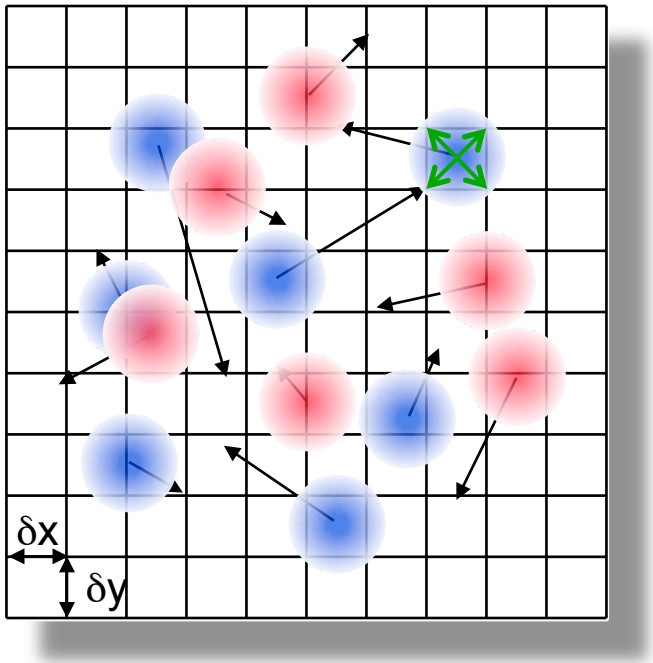
Particle-In-Cell workflow



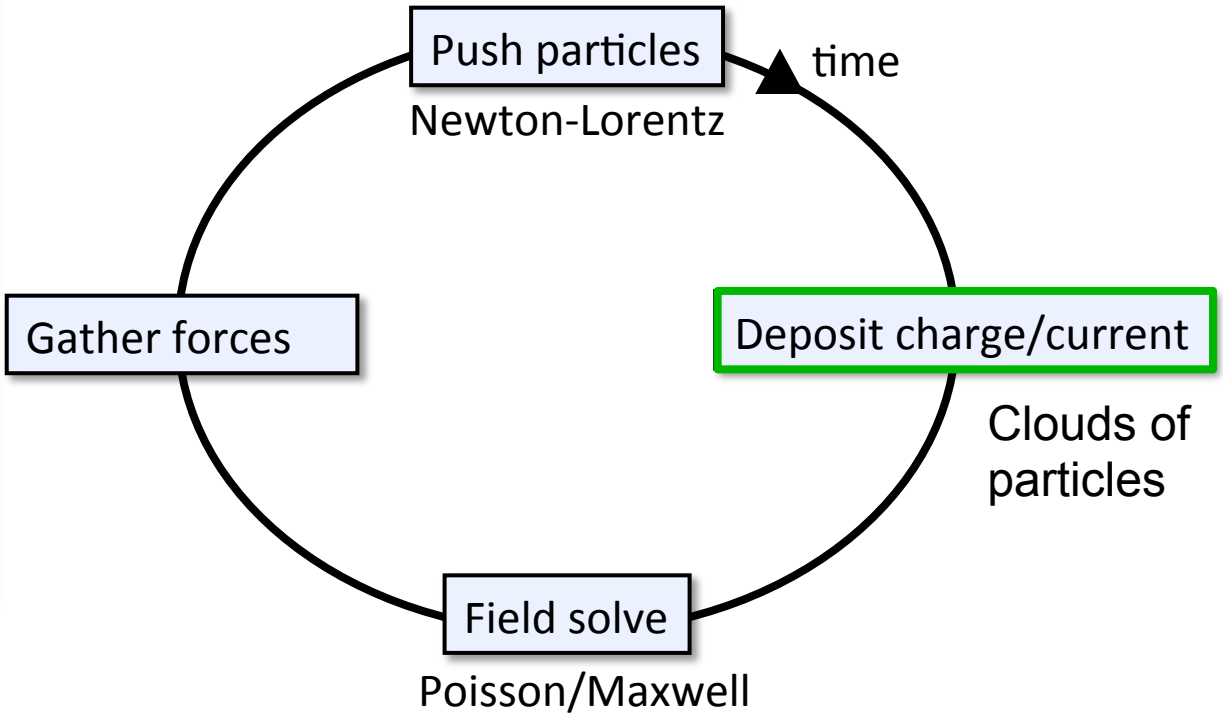
Plasma=collection of interacting charged particles



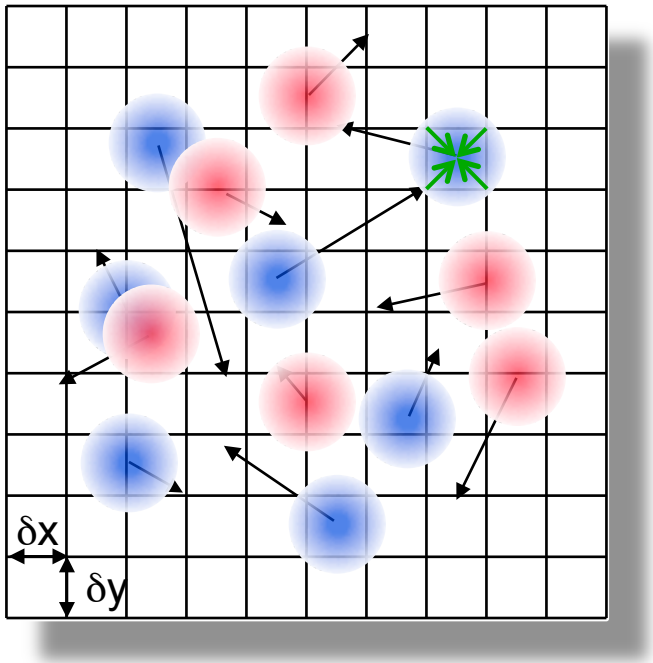
Particle-In-Cell workflow



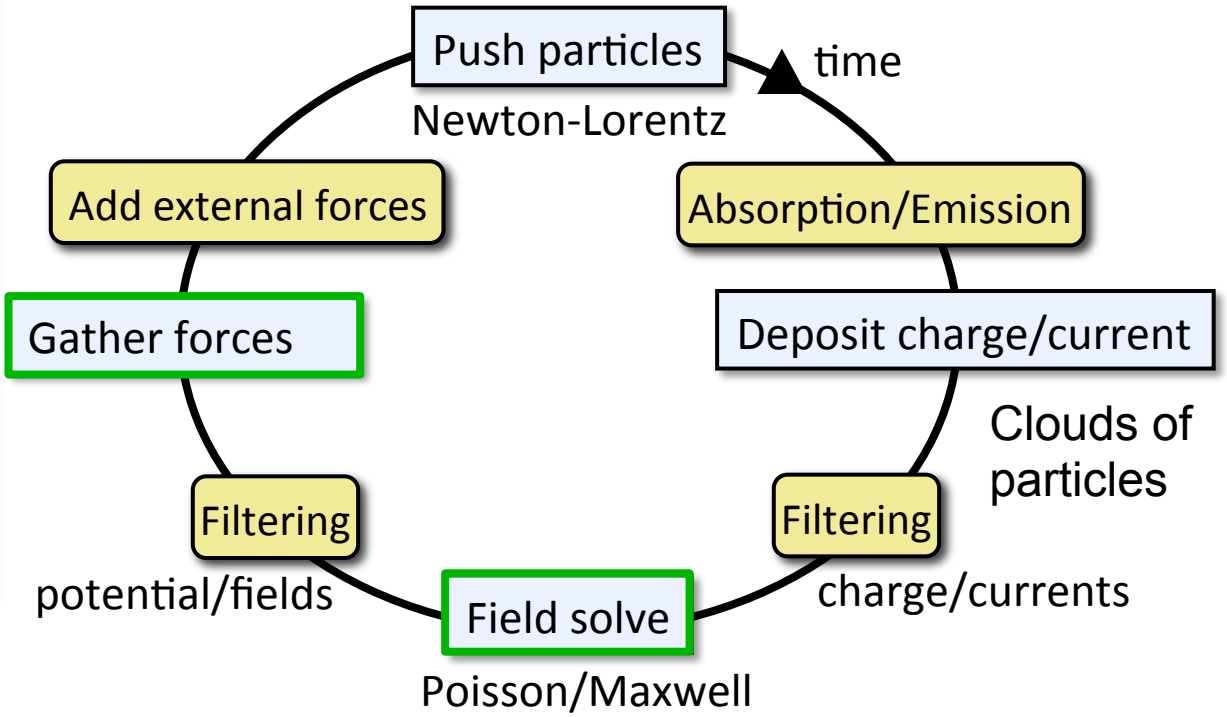
Plasma=collection of interacting charged particles



Particle-In-Cell workflow



Plasma=collection of interacting charged particles



+ filtering (charge,currents and/or potential,fields).

+ absorption/emission (injection, loss at walls, secondary emission, ionization, etc),

+ external forces (accelerator lattice elements),

PIC Proxy should allow for different types of push steps according to physics

Particle Push step uses equations of motion. Here, we see a typical Time-difference of eqns of motion: second order leap-frog scheme

$$\frac{d\mathbf{v}_i}{dt} = \frac{q_i}{m_i}\mathbf{E}_\mathbf{s} \approx \frac{\mathbf{v}_\mathbf{i}(t+\Delta t/2) - \mathbf{v}_\mathbf{i}(t-\Delta t/2)}{\Delta t} = \frac{q_i}{m_i}\mathbf{E}_\mathbf{s}(t)$$

$$\frac{d\mathbf{x}_i}{dt} \approx \frac{\mathbf{x}_\mathbf{i}(t+\Delta t) - \mathbf{x}_\mathbf{i}(t)}{\Delta t} = \mathbf{v}_\mathbf{i}(t+\Delta t)$$

Solution is explicit time advance:

$$\mathbf{v}_\mathbf{i}(t+\Delta t/2) = \mathbf{v}_\mathbf{i}(t-\Delta t/2) + \frac{q_i}{m_i}\mathbf{E}_\mathbf{s}(\mathbf{x}_\mathbf{i}(t))\Delta t$$

$$\mathbf{x}_\mathbf{i}(t+\Delta t/2) = \mathbf{x}_\mathbf{i}(t) + \mathbf{v}_\mathbf{i}(t+\Delta t/2)\Delta t$$

GTS (Gyrokinetic Tokamak Simulation)

- GTS particles are moved along the characteristics in phase space
  - Gyro-averaged Vlasov equation reduced to a simple system of ordinary differential equations for particle push
- Straight-field-line magnetic coordinates in toroidal geometry are employed (natural coordinates for tokamak)
- As before, grid replaces the direct binary interaction between particles by accumulating the charge of those particles on the grid at every time step and solving for the electromagnetic field, which is then gathered back to the particles' positions

The DSL-inspired proxy can allow for more complicated particle movers

- **Equations of motion for the particles along the characteristics, slightly more complicated, same type of calculation:**

$$\frac{d\mathrm{R}}{dt} = \mathrm{v}_{\|}\hat{\mathrm{B}} - \left(\frac{q}{m\Omega}\right)\left(\frac{\partial\Psi}{\partial\mathrm{R}}\times\hat{\mathrm{B}}\right)$$

$$\frac{d\mathrm{v}_{\|}}{dt} = -\left(\frac{q}{m}\right)\frac{\partial\Psi}{\partial\mathrm{R}}\cdot\hat{\mathrm{B}}$$

$$\frac{d\mathrm{w}_j}{dt} = -\left[\left(\frac{q}{m\Omega}\right)\frac{\partial\Psi}{\partial\mathrm{R}}\times\hat{\mathrm{B}}\cdot\hat{\mathrm{x}}_k - \left(\frac{q}{m}\right)\frac{\partial\Psi}{\partial\mathrm{R}}\cdot\hat{\mathrm{B}}\frac{1}{f_0}\frac{\partial f_0}{\partial\mathrm{v}_{\|}}\right]_{\mathrm{R}_j,\mu_j}$$

$$\text{with}\quad \mathrm{w}_j = \delta f / f$$

Differences in Particle-In-Cell Codes

- Particle-in-Cell codes are used for a huge variety of applications as seen in intro
  - Versions at LBNL include Impact and Warp
- The family of codes known as GTC/GTS implements a Particle method to solve the Gyrokinetic Equations in Tokamaks and other toroidal fusion devices
  - First version of GTC was created by Zhihong Lin, UCI
  - Latter US Gyro-PIC include:
    - GTS Stephane Ethier and Weixing Wang, PPPL
    - XGC CS Chang, PPPL
  - Primary benchmark (proxy) is called GTC
  - Unfortunately, pure optimization of GTC can result in a focus on steps that are primarily relevant to the unusual tokamak geometry including gyromotion

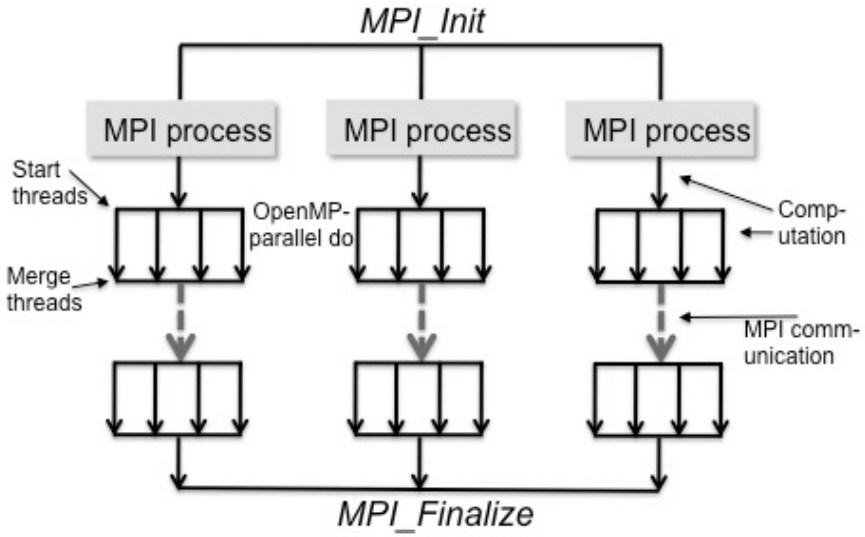
Proxy Apps should be combined with full app study in areas where codes are open

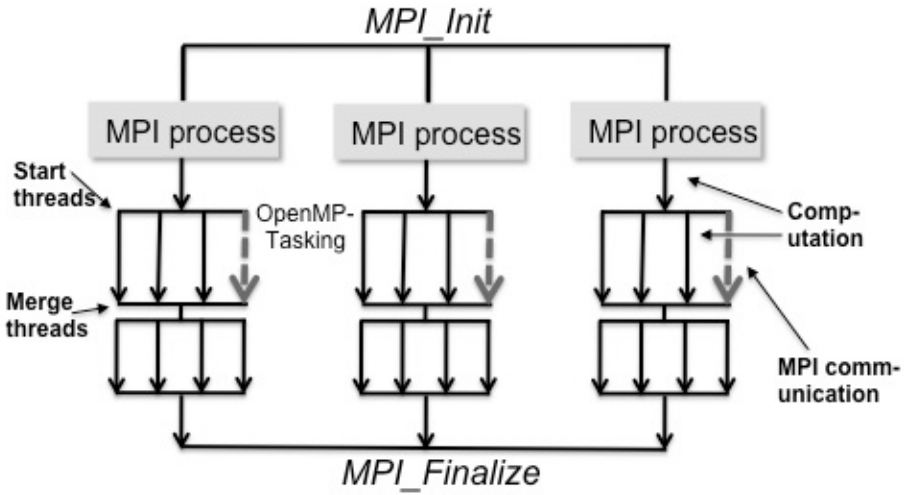
- Proxy Apps are usually the benchmark versions of codes
- Often used for procurements
- For full optimization science, we should also consider the more production versions of codes
  - Typically, these can still be studied as Proxy Apps if the basic steps are compartmentalized and analyzed

Samples of PIC Proxy Optimization

- We have investigated various means of optimizing the steps of one of our gyrokinetic PIC codes
- Investigated use of Advanced OpenMP constructs
- Investigated use of PGAS (Partitioned Global Address Space) languages to replace MPI (allows for re-factoring of algorithm)
- In progress: studies of solver optimization for both Poisson solver (electrostatic) and Maxwell solvers (time-dependent)

Two different hybrid models in GTS: Using traditional OpenMP worksharing and OpenMP tasks



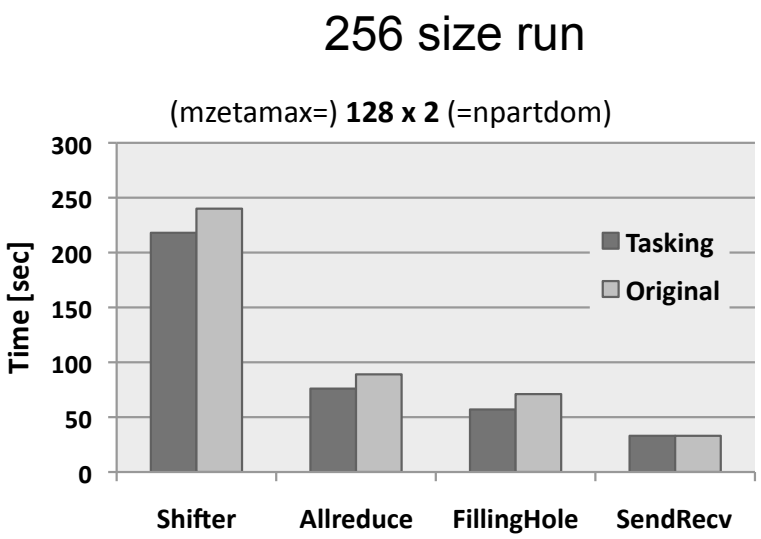


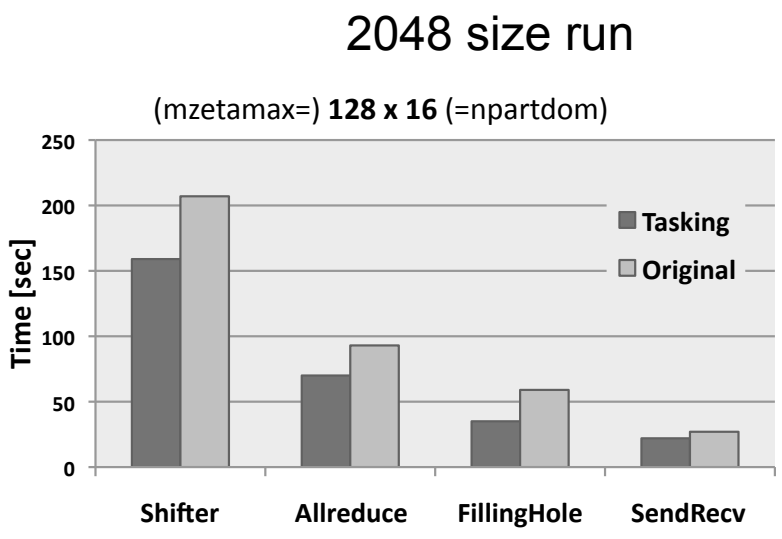
NEW OpenMP Tasking Model gives a new way to achieve more parallelism from hybrid computation.

OpenMP tasks enables us to overlap MPI communication with independent computation and therefore the overall runtime can be reduced by the costs of MPI communication.

Overlapping Communication with Computation using OpenMP tasks on the GTS magnetic fusion code, R. Preissl, A. Koniges, S. Ethier, W. Wang, N. Wichmann, Journal of Scientific Programming, 2011

OpenMP tasking version outperforms original kernel, especially in larger poloidal domains



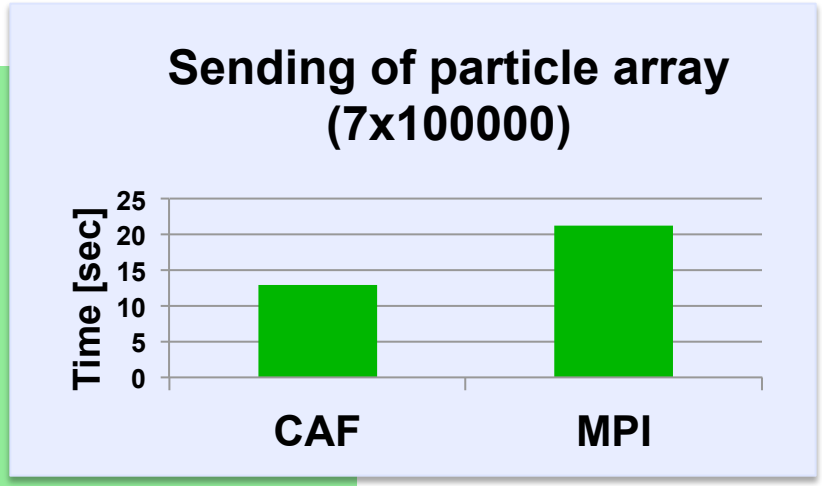


Performance breakdown of GTS shifter routine using 4 OpenMP threads per MPI process with varying domain decomposition and particles per cell.

MPI communication in uses a **toroidal MPI communicator** (constantly 128)
Large performance differences in the 256 MPI run compared to 2048 MPI run!

Speed-Up expected to be higher on larger GTS runs with hundreds of thousands CPUs since MPI communication is more expensive

CAF in GTS (+MPI) reduces lines of codes and speeds up application




```
program main
    include 'mpif.h'
    real array(10)
    integer rank, status(MPI_STATUS_SIZE), error

    ! Initialize MPI, and get process rank
    call MPI_INIT(error)
    call MPI_COMM_RANK(MPI_COMM_WORLD, rank, error)

    ! Initialize array on process 1
    if ( rank == 1 ) array = 0.5

    ! Send/recv data
    if ( rank == 0 ) then
        call MPI_RECV(array, 10, MPI_REAL, 1, 0, MPI_COMM_WORLD, status, error)
    else if ( rank == 1 ) then
        call MPI_SEND(array, 10, MPI_REAL, 0, 0, MPI_COMM_WORLD, error)
    endif

    ! Finish up
    call MPI_FINALIZE(error)
end program
```

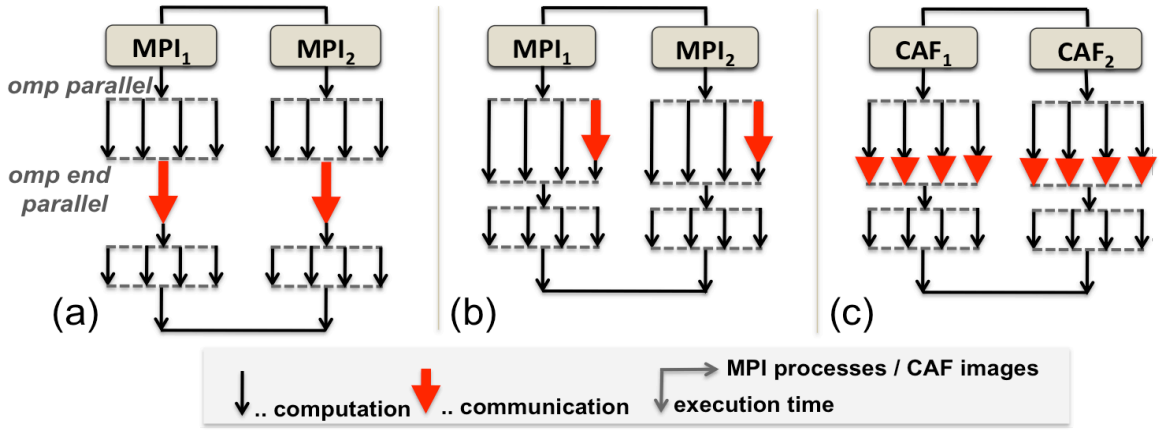
MPI

CAF

CAF = CoArray Fortran

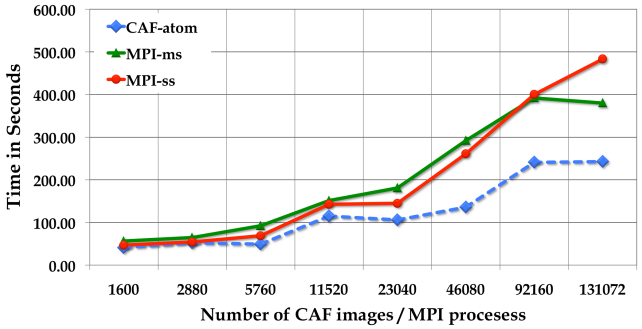
```
program main
    real array(10)[*]
    if ( this_image() == 2 ) array = 0.5
    sync all
    if ( this_image() == 1 ) array(:)[1] = array(:)[2]
end program
```

New kernel using a combination of MPI, OpenMP, and CAF gives significant performance improvement on 130K cores

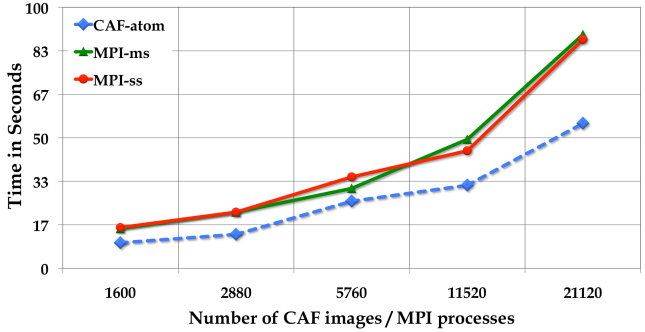


a) Classical hybrid MPI/OpenMP
b) Extension – MPI thread teams for work distribution and collective MPI function calls
c) Hybrid PGAS (CAF) / OpenMP allows ALL OpenMP threads per team to make communication calls to the thread-safe PGAS communication layer

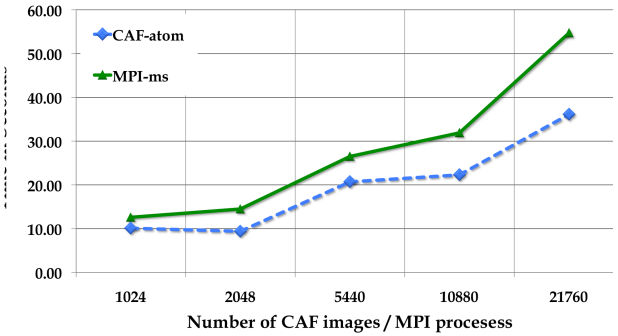
Preissl, Wichmann, Long, Shalf, Ethier, Koniges, SC11 Paper



Single-Threaded (Benchmark)



Multi-Threaded (Benchmark)



Multi Threaded (GTS)

Requires interoperability of MPI, OpenMP, and PGAS

Outline

- Intro to High Performance Computing Challenge areas for Plasma Physics
  - Magnetic Fusion Energy: Tokamak or other magnetically contained plasma for fusion energy (generally toroidally shaped)
  - Inertial Fusion Energy: Create fusion reaction using lasers or ion beams
  - Accelerators
    - As Free-Electron Laser sources for production of very short wavelength light sources—creates 3D images of very small things
    - As an Ion Beam Accelerator to study Warm Dense Matter and properties of materials
- Primary Proxy App: PIC (Particle-In-Cell) Simulations
- **Other Areas for Study:**
  - **Solvers for MHD and Solvers within PIC codes**
  - **Behavior of plasma turbulence codes**

The 2-Fluid MHD Equations are a complicated target for development of a proxy app

$$\frac{\partial n}{\partial t} + \nabla \bullet (n\mathbf{V}) = 0$$ continuity

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \qquad \nabla \bullet \mathbf{B} = 0 \qquad \mu_0 \mathbf{J} = \nabla \times \mathbf{B}$$ Maxwell

$$nM_i\left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \bullet \nabla \mathbf{V}\right) + \nabla p = \mathbf{J} \times \mathbf{B} - \nabla \bullet \Pi_{GV} + \mu \nabla^2 \mathbf{V}$$ momentum

$$\mathbf{E} + \mathbf{V} \times \mathbf{B} = \eta \mathbf{J} + \frac{1}{ne}\left(\mathbf{J} \times \mathbf{B} - \nabla p_e\right)$$ Ohm's law

$$\frac{3}{2}\frac{\partial p_e}{\partial t} + \nabla \bullet \left(\frac{3}{2} p_e \mathbf{V}\right) = -p_e \nabla \bullet \mathbf{V} + \eta J^2 - \nabla \bullet \mathbf{q}_e + Q_\Delta + S_{Fe}$$ electron energy

$$\frac{3}{2}\frac{\partial p_i}{\partial t} + \nabla \bullet \left(\frac{3}{2} p_i \mathbf{V}\right) = -p_i \nabla \bullet \mathbf{V} + \mu \left|\nabla V\right|^2 - \nabla \bullet \mathbf{q}_i - Q_\Delta + S_{Fi}$$ ion energy

Ideal MHD

Resistive MHD

2-fluid MHD

n number density

$\mathbf{B}$ magnetic field

$\mathbf{J}$ current density

$\mathbf{E}$ electric field

$nM_i \equiv \rho$ mass density

$\mathbf{V}$ fluid velocity

p_e electron pressure

p_i ion pressure

$p \equiv p_e + p_i$

e electron charge

μ viscosity

η resistivity

$\mathbf{q_i}, \mathbf{q_e}$ heat fluxes

Q_Δ equipartition

μ_0 permeability

$S_{Fe,i}$ Fusion power

Several MHD Codes are in use for Magnetic Fusion Energy Simulations

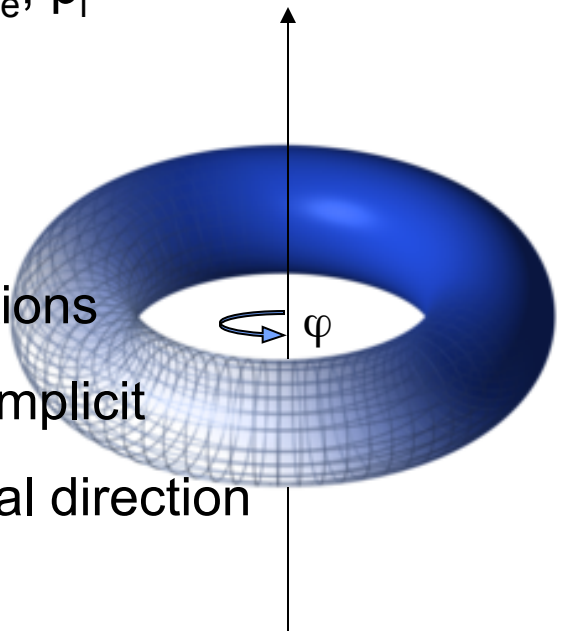
Code Name	Developers/Major Users
NIMROD	C. Sovinec, S. Kruger, D. Schnack, C. Kim, many others
M3D	J. Breslau, L. Sugiyama, H. Strauss, G. Fu, J. Chen, others
M3D-C^1	N. Ferraro, S. Jardin, J. Breslau, J. Chen
PIXIE3D	L. Chacon, others
HiFi	S. Lukin, A. Glasser, others

Slide credit: S. Jardin

Note: M3D-C^1 is an extension of M3D that uses higher-order finite elements and is fully implicit

Should we have a proxy app or focus on solvers for MHD for tokamaks?

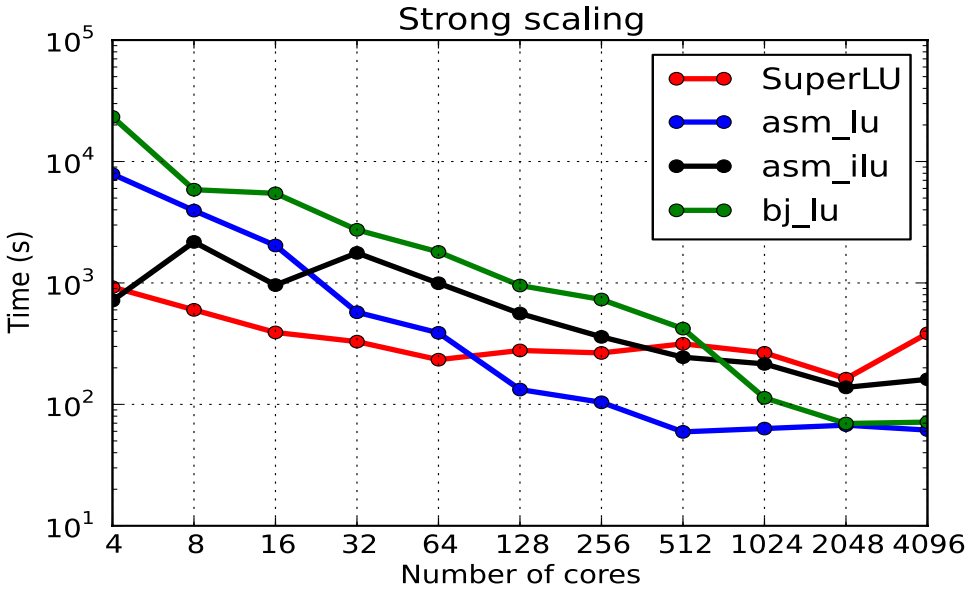
~ 8-9 variables per element (or mesh point) **V**, **B**, ρ, p_e, p_i
~ 10^2 toroidal planes (or Fourier modes)
➔Large sparse matrix equations require low latency

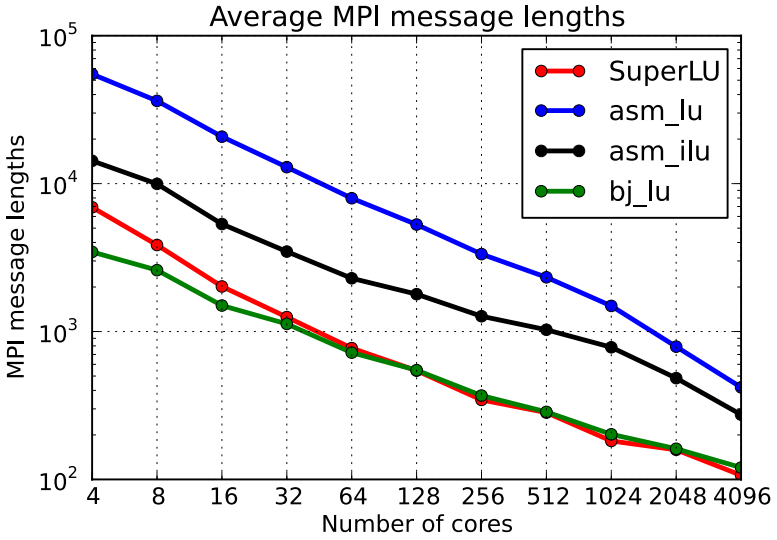


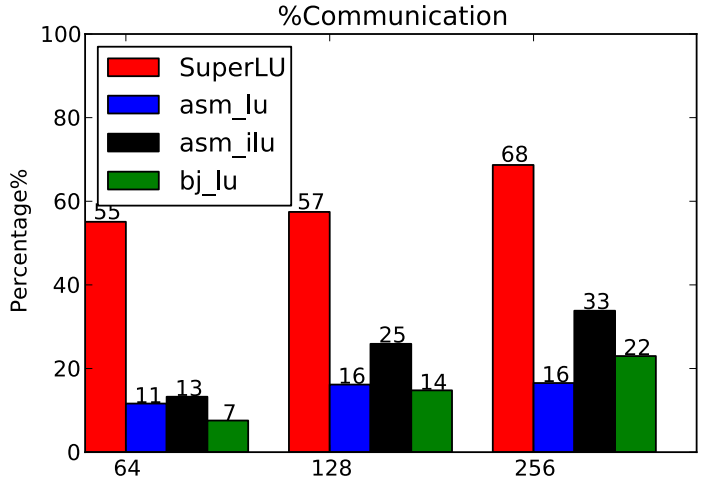
Codes vary in:

- single (big) matrix equation or several smaller equations
- non-linearly implicit (NK), linearly implicit, or partial implicit
- spectral, finite element, or finite differences in toroidal direction

Using a Proxy App inspired Reduced MHD and a 2D solver we gauge performance of full code







- Three iterative solvers (bj_lu, asm_ilu, asm_lu) and the direct solver (SuperLU) for a 256X256 size problem
- SuplerLU and bj_lu has lower MPI message lengths
- the communication percentage of SuperLU is over half of the wall time and increases as the number of cores increases

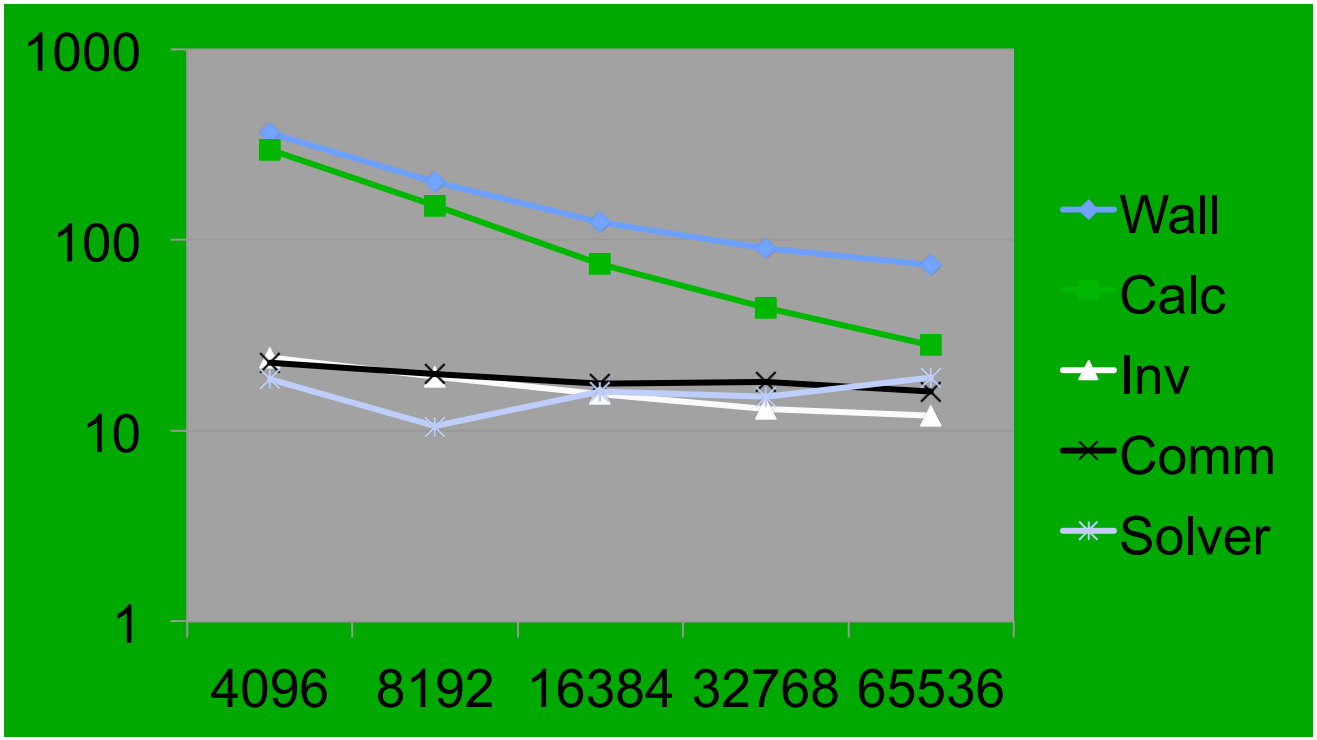
From: Application of PDSLin to the magnetic reconnection problem, Xuefei Yuan, Xiaoye S. Li, Ichitaro Yamazaki, Stephen C. Jardin, Alice E. Koniges and David E. Keyes, Comp. Sci Dis. 6, 2013.

Outline

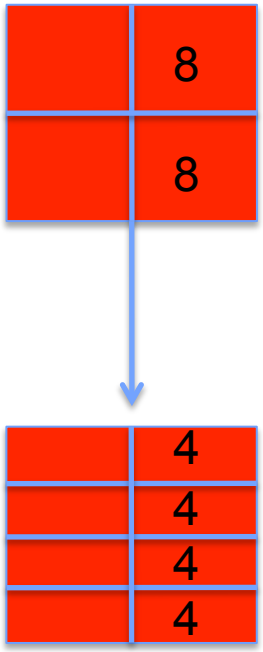
- Intro to High Performance Computing Challenge areas for Plasma Physics
  - Magnetic Fusion Energy: Tokamak or other magnetically contained plasma for fusion energy (generally toroidally shaped)
  - Inertial Fusion Energy: Create fusion reaction using lasers or ion beams
  - Accelerators
    - As Free-Electron Laser sources for production of very short wavelength light sources—creates 3D images of very small things
    - As an Ion Beam Accelerator to study Warm Dense Matter and properties of materials
- Primary Proxy App: PIC (Particle-In-Cell) Simulations
- Other Areas for Study:
  - Solvers for MHD and Solvers within PIC codes
  - **Behavior of plasma turbulence codes**

Bout++: break up times in each kernel to check how they scale

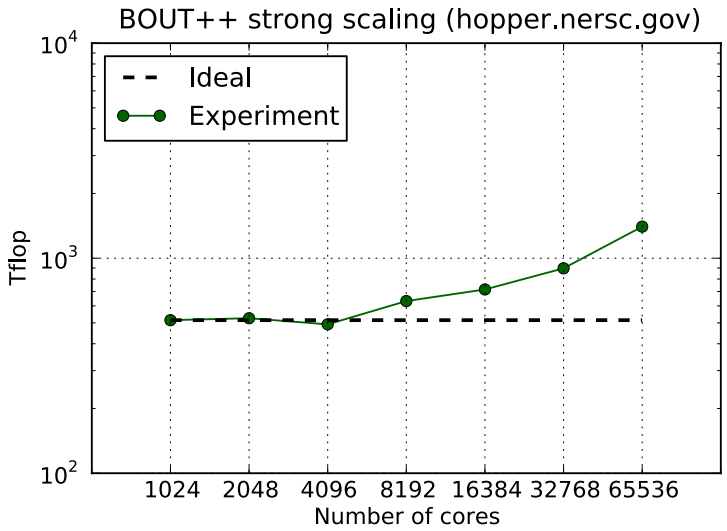
- Breakup by time spent: Calc scales somewhat, but inv, solver do not scale

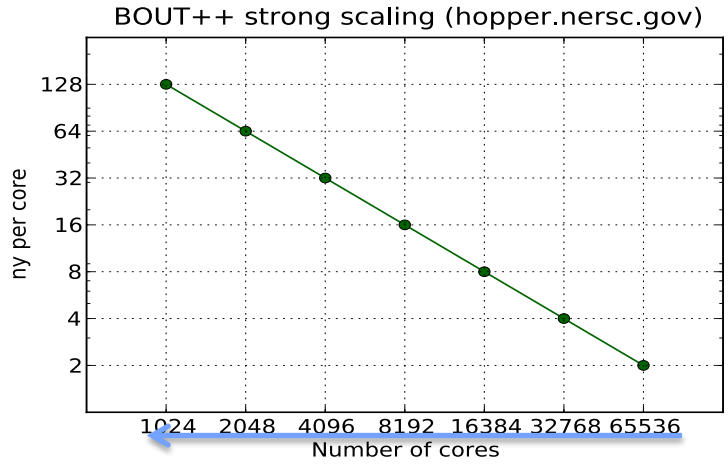


Strong Scaling Studies show that typical problem sizes will not scale well beyond a few thousand



Grid points/proc decreases with concurrency





- Steady increase in flop count
- Extra computations in ghost cells (and more cycles spent in doing these)
- Valid region (excluding ghost region) does same amount of work

More work needed before designing an appropriate proxy

Praveen Narayanan and Alice Koniges, Bout++ Workshop 2011

Conclusions

- PIC codes are a "standard" proxy app for plasma and accelerator applications
  - Grouping these together into a common language (DSL-inspired) can help to spread benefit of proxy
  - Isolating pieces of proxy for study with advanced programming model concepts provides important improvement in application performance
- MHD codes are more difficult. Can focus on solver elements and build the proxy with reduced physics while retaining properties required of the solvers
- Turbulence codes are varied. May be a while to determine best proxy for model representation. As with CFD, stencil dependent.
- Thanks – Applications – Viktor Decyk UCLA, Stephane Ethier PPPL, Steve Jardin PPPL, Sherry Li LBNL, Jean-Luc Vay LBNL, X. Xu LLNL