Concurrent Divide-and-Conquer Library
with Petascale Electromagnetics Applications

Johan Carlsson, Tech-X Corporation

CScADS Workshop on Libraries and Algorithms for Petascale Applications, 07/30/2007, Snowbird, Utah

Background

- Particle In Cell (PIC) in a sentence: solve Faraday and Ampere laws for electromagnetic fields and get current closure from set of particles accelerated by Lorentz force
- For the purpose of this talk: think FDTD EM + explicit current-source calculation
- Two main uses for implicit field updates
  - Allow time steps beyond CFL limit, $\Delta t > \Delta x/c$
  - Numerical dissipation can suppress particle and other numerical noise (grid heating)
- VORPAL PIC code used by INCITE project (PI’d by Geddes) and OFES SciDAC project (PI’d by Bonoli), etc.
- Explicit (Yee) field update has excellent scalability
- Original implicit field update was added to VORPAL in February 2004
Two implicit field updates were on shortlist
- Bowers adds damping terms to Faraday and Ampere laws
  \[ \nabla \times \mathbf{F} \rightarrow \nabla \times [(1 + \tau F \partial_t)\mathbf{F}], \text{ for } \mathbf{F} = \mathbf{E}, \mathbf{B} \nabla \times \mathbf{F} \rightarrow \nabla \times [(1 + \tau F \partial_t)\mathbf{F}], \text{ for } \mathbf{F} = \mathbf{E}, \mathbf{B} \]
- Use Crank-Nicholson (CN) time discretization
- Damping times \( \tau_E, \tau_B \) correspond to implicitness parameters
- \( \tau_E = \tau_B = \Delta t/2 \Rightarrow \text{fully implicit} \)
- \( \tau_E = \tau_B = -\Delta t/2 \Rightarrow \text{fully explicit} \)
- Numerical dispersion well analyzed by Bowers

ZCZ ADI (Alternating Direction Implicit)
- Not charge conserving!

Bowers came out ahead

Implementation of Bowers implicit field update in VORPAL

- An electrostatic (ES) field update had just been added to VORPAL
  - Trilinos/Aztec used for linear solve
- Bowers implicit field update implemented making maximal reuse of existing ES solver
- Implementation was done in a couple of weeks as evidenced by SVN log of main class file:

  Thu 05 Feb 12:59:02 Adding new class for the implicit field solver.
  Thu 05 Feb 13:41:20 First iteration of changes.
  Fri 06 Feb 19:35:00 Flipped the indices around.
  Mon 09 Feb 19:42:19 Coded up the coefficient matrix.
  Tue 10 Feb 11:46:06 Bug fix in the coefficient matrix.
  continued...
Implementation of Bowers implicit field update in VORPAL

Tue 10 Feb 12:25:05 Added the proper expressions for the coefficient-matrix elements (including Bowers’ damping coefficient, etc.).
Tue 10 Feb 19:56:03 This afternoon’s changes.
Wed 11 Feb 15:46:34 Coded up the right-hand side.
Wed 11 Feb 18:53:23 The core code should now be pretty much complete (setting up coefficient matrix and RHS, updating the field).
Mon 16 Feb 18:26:35 Bugfix. Now it builds with HAVE_AZTEC defined.
Fri 20 Feb 11:18:07 Misc bugfixes.
Tue 24 Feb 21:49:24 Minor changes, still having convergence problems with the iterative solver.
Wed 25 Feb 19:44:23 Several critical bugfixes! Implicit solver now seems to work.

Performance of Bowers implicit field update in VORPAL

![Graph showing Courant number vs. CPU time for different methods.]
Performance of Bowers implicit field update in VORPAL

- \( \sim 3 \times \) slower than explicit at CFL limit
- \( 2 - 3 \times \) more memory used by field (can’t update in place)
- Good numerical stability (convergence for CFL number above \( 10^3 \))
- However, \( \sim 10 \) iterations in 1D, \( \sim 40 \) in 2D, \( \sim 400 \) in 3D!
- Became orphaned when OFES SBIR Phase I project didn’t go to Phase II
- Has only been clearly superior in particle-dominated (hundreds per cell) 2D simulations
- Numerical dispersion not suitable for suppressing grid heating
- Concern about scalability due to global solve
- Multigrid preconditioning recently tried, but found ineffective

Recent developments on Maxwell solvers using Alternating Direction Implicit (ADI)

- Seminal paper by ZCZ mentioned above with first unconditionally stable ADI field update
- Work at Tech-X over last year by Smithe, Cary and Carlsson
  - Improved version of ZCZ that is Space-Charge Conserving (SCC) ADI
  - SCC ADI with “perfect dispersion” (avoids numerical Cherenkov radiation)
- Should lead to high-fidelity solutions
- But does it scale?!?
ADI seems to have largely fallen out of favor

- Was supposedly popular for elliptic problems before multigrid became dominant
- Divide timestep into substeps and Alternate which single Direction is Implicit in each substep
- Reduces single global solve into series of small tridiagonal solves
- Could make implicit field update more like explicit update (Yee) in terms of communication needs (keep field data local) and scalability
- Bondelli introduced parallel algorithm for solving tridiagonal system: Divide & Conquer

ADI non-trivial to parallelize efficiently

- Use Divide & Conquer to parallelize solve of tridiagonal system
  - Begin solve in parallel
  - Sequential bottleneck
  - Finish solve in parallel
Concurrent Divide-and-Conquer

- Keep multiple tridiagonal solves in flight simultaneously to overlap communication with computation

**Multiple Simultaneous Divide & Conquers**

- Let $C_{1D}$ be number of cells in one direction, so that $C_{3D} = C_{1D}^3$
- Let $N_{1D}$ be number of processors in one direction, so that $N_{3D} = N_{1D}^3$
- Let $\tau_{cell}$ = time it takes to do backsolve of a single cell
- Let $\tau_{latency}$ = time it takes to receive a message
- Then the maximum number of cell-rows that can be done simultaneously by a processor is $N_{cellrows} = \frac{C_{1D}^2}{N_{1D}^2}$
- A processors share of the common-matrix-inversions is therefore, $N_{immediate} = N_{cellrows} / N_{1D}$
- The number of cells in a single-processor’s backsolve is $N_{cellsSolved} = C_{1D} / N_{1D}$

So immediately following a processor’s common-matrix-inversion-and-send on the $N_{immediate}$ matrices, it can proceed with $N_{immediate}$ backsolves, which will take a time:

$$N_{immediate} N_{cellsSolved} \tau_{cell}$$

to perform. This must exceed $\tau_{latency}$ in order to ensure that there is no idle processor time.
Might be possible to achieve good scaling on biggest available machines

\[
N_{\text{immediate}} N_{\text{cellsSolved}} \tau_{\text{cell}} \geq \tau_{\text{latency}} \\
(N_{\text{cellrows}}/N_{1D})(C_{1D}/N_{1D}) \tau_{\text{cell}} \geq \tau_{\text{latency}} \\
((C_{1D}^2/N_{1D}^2)/N_{1D})(C_{1D}/N_{1D}) \tau_{\text{cell}} \geq \tau_{\text{latency}} \\
(C_{1D}^3/N_{1D}^4) \tau_{\text{cell}} \geq \tau_{\text{latency}} \\
(C_{3D}/N_{3D}^{4/3}) \tau_{\text{cell}} \geq \tau_{\text{latency}} \\
N_{3D} \leq (C_{3D} \tau_{\text{cell}} / \tau_{\text{latency}})^{3/4}
\]

With $\tau_{\text{latency}} = 100 \tau_{\text{cell}}$, and a billion cells ($10^3 \times 10^3 \times 10^3$), it would then be possible to scale beyond $10^5$ processors.

ADI can simplify handling of complex boundaries

ADI in ILC SRF cavity
How do we get an efficient and portable implementation?

We have received an ASCR SBIR Phase I grant to implement a prototype ADI library and prove the scalability of our new SCC ADI algorithm

- Must balance efficient implementation vs. portability
- MPI1 or MPI2? Or UPC?
  - MPI2 becoming ubiquitous
  - Any new killer features?
- Multilevel parallelism with MPI/OpenMP?
  - gcc4.2 supports OpenMP
  - BlueGene/P nodes are SMP
- More library dependencies can be added to VORPAL only if there are very compelling reasons
- Other considerations for efficient implementation?