Some Scalability Issues Of PETSc and VORPAL Applications

S. Ovtchinnikov and P. Messmer

Tech-X Corporation

CScADS Workshop, July 30^{th}, 2007

PETSc

- Portable Extensible Toolkit for Scientific computation available from Argonne National Laboratory
- Started using it sometime around Y2K
- Implemented applications such as radiation diffusion problem, concrete absorption, fast magnetic reconnection
- C and Fortran
- Platforms: Linux clusters, IBM SP, IBM BG/L, Windows
- Used $np \in [1, 2025]$

VORPAL

- Relativistic, arbitrary dimensional, hybrid plasma and beam simulation code from Tech-X Corporation
- Started using it four months ago
- Applications: EM simulations, plasma simulations, accelerator cavities simulations, structure optimizations
- VORPAL is used by researcher from JLAB, FNAL,LBNL, Brookhaven NL, etc.
- Plain MPI, C++, Python, Trilinos, HDF5, newer autotools. Code development by Tech-X and University of Colorado at Boulder
- Platforms: Linux clusters, IBP SP, BG/L, SGI, Windows, etc.

Problem Description (PETSc)

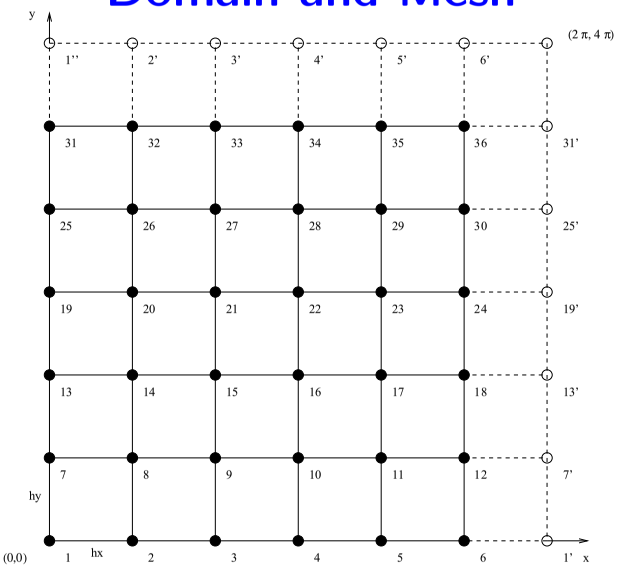
- Work with plasma equations in magnetohydrodynamics (MHD) formalism
- Simulate magnetic reconnection problem in reduced, two-dimensional, Hall-MHD formulation (Fitzpatrick, 2004; Grasso *et al*, 1999) – time-dependent, nonlinear problem with a non-smooth solution
- Apply fully coupled, nonlinearly implicit, parallel algorithms. Contrast with: explicit, semi-implicit, implicit (physics-based preconditioning, matrix-free)
- Study the parallel performance of algorithms and assess their applicability to solutions of problems with millions of unknowns using thousands of processors

Model MHD System

$$
\begin{cases}
\nabla^2\phi &= U \\
\nabla^2\psi &= \dfrac{1}{d_e^2}(\psi - F) \\
\dfrac{\partial U}{\partial t} + [\phi, U] &= \dfrac{1}{d_e^2}[F, \psi] + \nu\nabla^2 U \\
\dfrac{\partial F}{\partial t} + [\phi, F] &= \rho_s^2[U, \psi] + \eta\nabla^2(\psi - \psi^0),
\end{cases}
$$

where U is the vorticity, F is the canonical momentum, ϕ ($\mathbf{V_i} = \hat{\mathbf{z}} \times \nabla\phi + V_z\hat{\mathbf{z}}$) and ψ ($\mathbf{B} = \nabla\psi \times \hat{\mathbf{z}} + B_z\hat{\mathbf{z}}$) are the stream functions for the vorticity and current density, respectively, ν is the plasma viscosity, η is the normalized resistivity, $d_e = c/\omega_{pe}$ is the inertial skin depth, and $\rho_s = \sqrt{T_e/T_i}\rho_i$ is the ion sound Larmor radius. The current density is obtained by $C_z = (F - \psi)/d_e^2$. This is equivalent to $\beta = 0$, but keeping ρ_s finite (Fitzpatrick, 2004).

Domain and Mesh



Rectangular domain $\Omega = [0, 2\pi) \times [0, 4\pi)$ with doubly periodic boundary conditions and a sample 6×6 mesh. The solid circles indicate actual mesh points and empty circles denote virtual points that correspond to the boundary mesh points on the opposite side of the domain.

Discretizations

- Spatial discretizations are the standard second-order central differences (five-point-stencil) used at every mesh point in the domain (including the boundary)
- Temporal discretizations for the implicit time stepping are based on multi step formulas (BDF); the discretization accuracy is up to the fourth-order (start with lower order (backward Euler) and gradually increase the order as more solution history becomes available)
- Temporal discretizations for the explicit method are based on the second-order Adams ($y^{k+1} = y^k + \Delta t(\frac{3}{2}f(x^k) - \frac{1}{2}f(x^{k-1}))$) with the CFL-based timestep size reduction
- The main focus is on using third-order accurate temporal and second-order accurate spatial discretizations in the implicit version of our code
- In the explicit version, we use second-order accurate temporal and second-order accurate spatial discretizations

Discrete System

$$\begin{cases} R_\phi^{k+1}(i,j) = 0 \\ R_\psi^{k+1}(i,j) = 0 \\ \frac{h_x h_y}{6\Delta t}\left(11U_{i,j}^{k+1} - 18U_{i,j}^{k} + 9U_{i,j}^{k-1} - 2U_{i,j}^{k-2}\right) - R_U^{k+1}(i,j) = 0 \\ \frac{h_x h_y}{6\Delta t}\left(11F_{i,j}^{k+1} - 18F_{i,j}^{k} + 9F_{i,j}^{k-1} - 2F_{i,j}^{k-2}\right) - R_F^{k+1}(i,j) = 0, \end{cases}$$

where $R_\phi^{k+1}(i,j)$, $R_\psi^{k+1}(i,j)$, $R_U^{k+1}(i,j)$, and $R_F^{k+1}(i,j)$ are the second-order accurate spatial discretizations of the time-independent components.

- R_U and R_F are highly nonlinear
- R_ϕ and R_ψ are linear and time independent

A Fully Coupled Inexact Newton

- At each time step, $\{R_\phi = 0, R_\psi = 0, \tilde{R}_U = 0, \tilde{R}_F = 0\} \rightarrow G(E) = 0$, where

$$E = (\phi_{11}, \psi_{11}, U_{11}, F_{11}, \phi_{21}, \psi_{21}, U_{21}, F_{21}, \cdots)^T,$$

and

$$G = (G_\phi(1,1), G_\psi(1,1), G_U(1,1), G_F(1,1), \\ G_\phi(2,1), G_\psi(2,1), G_U(2,1), G_F(2,1), \cdots)^T$$

- $E_{k+1} = E_k - \lambda_k J(E_k)^{-1} G(E_k)$, $k = 0, 1, ...$, where E_0 is a solution obtained at the previous time step

- $J(E_k) = G'(E_k)$ is the Jacobian at E_k ($dim(Null(J)) = 1$), and λ_k is the step-length determined by a line search procedure

- The accuracy of the Jacobian solve is determined by some $\eta_k, \gamma_k \in [0, 1)$ and the conditions:

$$\|G(E_k) + J(E_k) s_k\|_2 \leq max\{\eta_k \|G(E_k)\|_2, \gamma_k\}$$

Jacobian Matrix

- For every mesh point $E_i = \{\phi_i, \psi_i, U_i, F_i\}$ and a solution vector

$$u = \{u_1, u_2, u_3, ..., u_n\} = \{\phi_1, \psi_1, U_1, F_1, \phi_2, \psi_2, U_2, F_2, ..., \phi_n, \psi_n, U_n, F_n\},$$

where n is the number of mesh points

- Jacobian

$$J = \begin{pmatrix} \partial G_1/\partial u_1 & \partial G_1/\partial u_2 & ... \\ \partial G_2/\partial u_1 & \partial G_2/\partial u_2 & ... \\ ... & ... & ... \end{pmatrix}$$

- Analytic expression for J_{ij}

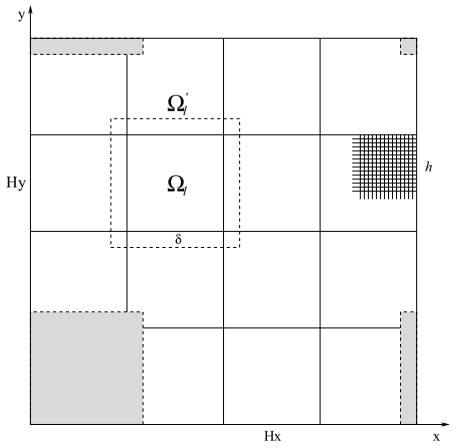
- Compute J_{ij} with finite differences and the multi-colored algorithm (Coleman *et al*, 1983):

$$J_{ij} = \frac{1}{\xi}(G_i(E_j + \xi) - G_i(E_j)),$$
$$J_{ij} = \frac{1}{(2\xi)}(G_i(E_j + \xi) - G_i(E_j - \xi))$$

Newton–Krylov–Schwarz

1. Inexactly solve the linear system $J(E_k)s_k = -G(E_k)$ for s_k using Schwarz-preconditioned GMRES(30)
   - One-level additive Schwarz right preconditioner (M^{-1}): $J(E_k)M^{-1}Ms_k = -G(E_k)$
   - Two-level additive Schwarz right preconditioner (M_2^{-1}): $J(E_k)M_2^{-1}M_2s_k = -G(E_k)$
   - Inner-outer preconditioning technique, where a Krylov subspace method is used as the preconditioner
2. Perform a full Newton step with $\lambda_0 = 1$ in the direction s_k
3. If the full Newton step is unacceptable, backtrack λ_0 using a backtracking procedure until a new λ is obtained that makes $E_+ = E_k + \lambda s_k$ an acceptable step
4. Set $E_{k+1} = E_+$ and return to step 1 unless a stopping condition has been met

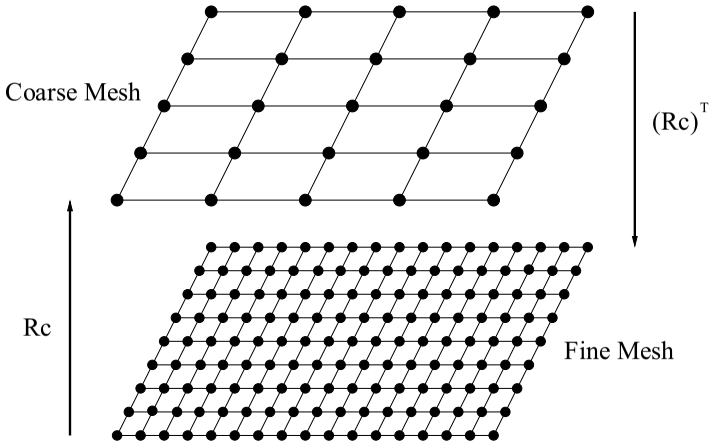
One-level Additive Schwarz Preconditioner



Restricted additive Schwarz (RAS):

$$M^{-1} = \sum_{j=1}^{N} (R_j^0)^T B_j^{-1} R_j^\delta$$

Two-level Additive Schwarz Preconditioner



$$M_2^{-1} = R_c^T A_c^{-1} R_c + \sum_{j=1}^{N} (R_j^0)^T B_j^{-1} R_j^\delta$$

Inner-outer Preconditioner

(1) $$AM^{-1}y = b,$$

(2) $$Mx = y$$

- FGMRES – outer method, applied to (1)
- GMRES – inner method, applied to systems $Mz = v_k$
- At the k^{th} outer cycle, the inner GMRES performs a certain number of iterations (m_k) for solving

  $$Mz = v_k,$$

  where m_k is limited by stopping criteria and v_k is the k- Arnoldi vector of FGMRES
- M^{-1} is the one-level restricted additive Schwarz preconditioner

Parallel Implementation

- Implemented in the Portable Extensible Toolkit for Scientific computation (PETSc) framework (2.3.2), Argonne National Laboratory; PETSc library is compiled with Fortran kernels
- Extensive use of distributed memory multigrid object
- The code's multiple configurations can be accessed via command line arguments (physical parameters, stopping conditions, restart sequence, solver configurations, etc.)
- Compilers: gcc and vendor-optimized proprietary systems
- Target hardware: Linux cluster and IBM BG/L

Parameters

- Physical parameters:

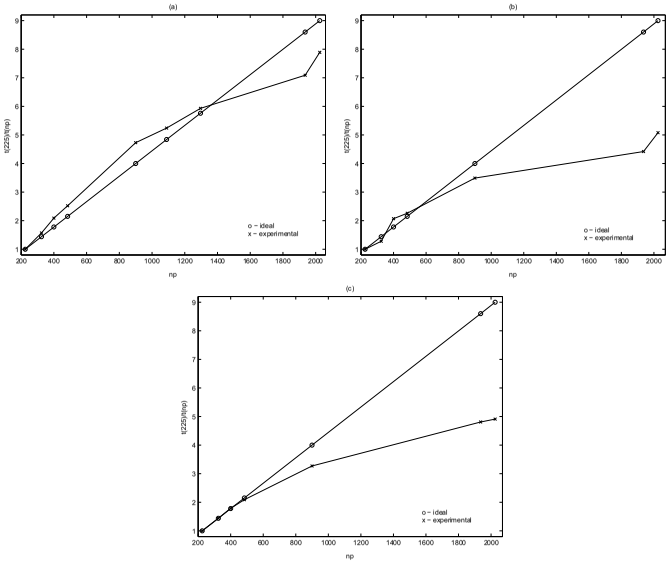
$$\begin{aligned} d_e &= 0.08, \\ \rho_s/d_e &\in [1.0, 10.0], \\ \nu &\in [10^{-5}, 10^{-2}], \\ \eta &\in [10^{-5}, 10^{-2}], \\ \epsilon &= l_x/l_y = 1/2 \end{aligned}$$

- Relative reduction in nonlinear function norm $\|G(E_k)\|_2 \leq 10^{-7}\|G(E_0)\|_2$
- Absolute tolerance in nonlinear function norm $\|G(E_k)\|_2 \leq 10^{-7}$
- Relative reduction in linear residual norm $\|r_k\|_2 \leq 10^{-10}\|r_0\|_2$
- Absolute tolerance in linear residual norm $\|r_k\|_2 \leq 10^{-7}$
- Subdomain solve: LU

Scalability: 1980×1980 Mesh

One-level Preconditioner, $t = 100\tau_A$			
np	Computing Time [sec]	Total Nonlinear Iterations	Linear/Nonlinear
225	716.8	10	42.2
324	457.8	10	46.1
400	342.3	10	56.0
484	284.7	10	56.4
900	151.7	10	63.0
1936	101.0	10	96.5
2025	90.8	10	98.4
One-level Preconditioner, $t = 200\tau_A$			
225	965.5	11	65.6
324	754.3	12	53.6
400	465.7	10	68.7
484	427.8	10	74.0
900	276.8	12	67.7
1936	218.6	14	162.4
2025	189.9	13	149.5
One-level Preconditioner, $t = 280\tau_A$			
225	2473.1	24	113.5
324	1691.9	24	127.7
400	1359.6	24	135.1
484	1185.0	25	141.3
900	742.8	25	181.0
1936	514.8	27	226.6
2025	504.8	26	244.3

Parallel Speedup: 1980×1980 Mesh



Speedup curves $t(np = 225)/t(np)$ as compared to the ideal speedup for 1980×1980 mesh, LU factorization for all subproblems, $\Delta t = 1.0\tau_A$, 10 time steps at times $t = 100\tau_A$ (a), $t = 200\tau_A$ (b) and $t = 280\tau_A$ (c).

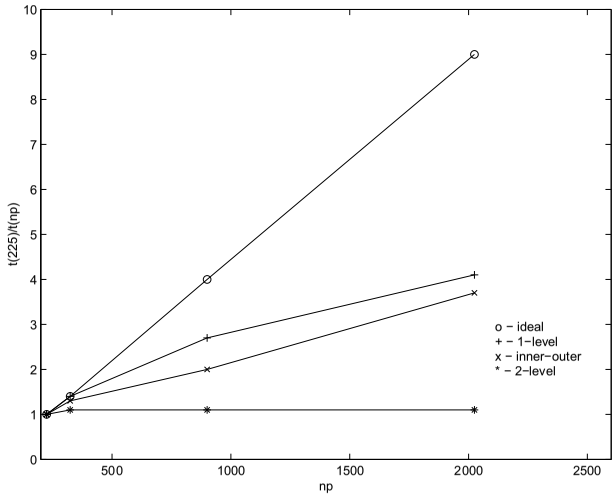
Execution Times of GMRES Orthogonalization

$t = 100\tau_A$						
np	Times Called	Exec. Time [sec]	Percentage	Per Call	Speedup	Ideal
400	510	4.65	1.5%	0.00912	1.00	1.00
900	716	3.04	2.3%	0.00425	2.15	2.25
2025	1074	2.84	3.1%	0.00264	3.45	5.06

$t = 280\tau_A$						
np	Times Called	Exec. Time [sec]	Percentage	Per Call	Speedup	Ideal
400	4603	44.2	4.0%	0.00960	1.00	1.00
900	6318	26.6	4.5%	0.00421	2.28	2.25
2025	8026	26.0	6.3%	0.00324	2.96	5.06

Execution times of the GMRES orthogonalization, one-level algorithm, 1980×1980 mesh, LU factorization for all subproblems, $\Delta t = 1.0\tau_A$, 10 time steps taken at simulation time $t = 280\tau_A$. The number of times the function is called, the function's cumulative execution time, the percentage of the total execution time, time per call, and experimental and ideal speedups are shown.

Comparison of Preconditioners: Speedup



Parallel speedup, 1080×1080 fine mesh, 90×90 coarse mesh, redundant solves on the coarse mesh, LU factorization for all subproblems, $\Delta t = 1.0\tau_A$, 10 time steps taken at simulation time $t = 200\tau_A$. The overlap $\delta = 8$. The experiments are conducted with the number of processors $np = 225, 324, 900, 2025$ in the "vn" mode, and each subdomain is assigned to one processor.

Comparison of Preconditioners: Computing Time

$t = 100\tau_A$			
np	Inner-outer Time [sec]	One-level Time [sec]	Two-level Time [sec]
225	50.02	134.3	670.9
324	35.10	97.1	608.1
900	35.59	39.8	524.9
2025	23.10	27.3	585.3
$t = 200\tau_A$			
225	95.3	253.5	682.2
324	73.5	177.7	617.3
900	48.6	92.9	604.2
2025	26.0	61.2	605.8
$t = 280\tau_A$			
225	2472.9	528.3	2449.8
324	1751.9	389.9	2178.4
900	1003.8	220.6	2052.9
2025	556.8	147.6	2573.5

Computing times of inner-outer, one-level and two-level preconditioning algorithms, 1080×1080 fine mesh, 90×90 coarse mesh, redundant solves on the coarse mesh, LU factorization for all subproblems, $\Delta t = 1.0\tau_A$, 10 time steps taken at simulation times $t = 100\tau_A$, $t = 200\tau_A$ and $t = 280\tau_A$. The overlap $\delta = 8$.

GFLOPS

1080×1080 mesh			
np	$t = 100\tau_A$	$t = 200\tau_A$	$t = 280\tau_A$
225	22.1	25.8	30.0
324	31.8	33.8	43.0
900	88.0	94.6	115.8
2025	166.1	175.9	252.6

1980×1980 mesh			
np	$t = 100\tau_A$	$t = 200\tau_A$	$t = 280\tau_A$
400	42.9	46.4	52.0
484	53.9	57.7	63.8
900	81.6	95.5	116.1
2025	188.0	198.4	249.3

GFLOP as a function of the number of subdomains, inner-outer preconditioning algorithm, LU factorization for all subproblems, $\Delta t = 1.0\tau_A$, 10 time steps taken at simulation times $t = 100\tau_A$, $t = 200\tau_A$ and $t = 280\tau_A$. The overlap $\delta = 8$.

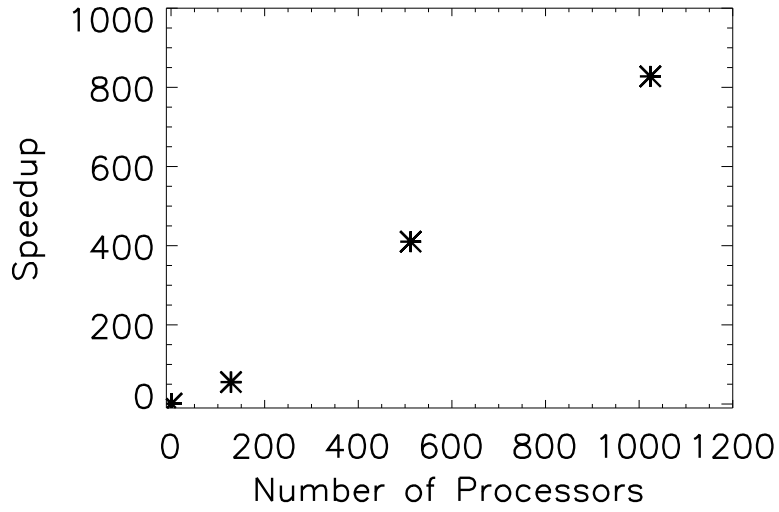
Some Observations and Future Work

- The one-level additive Schwarz preconditioner works quite well on fine meshes (1980×1980) with up to several hundred subdomains and for simulation times up to $t = 150\tau_A$
- For larger numbers of subdomains, in the range of one to two thousands, the efficiency of the one-level preconditioner degrades, and the parallel speedup is no longer close to ideal
- The two-level additive Schwarz preconditioning technique demonstrates an inferior performance on the time-dependent problem with periodic boundary conditions both in terms of the iteration count and the execution time
- In some cases, the application of the inner-outer preconditioner results in much shorter execution times and allows for a good parallel performance
- Multilevel versions of the algorithm require improvements – solve the coarse problem in parallel using MUMPS (Amestoy *et al*, 2000) or SuperLU (Demmel *et al*, 2003)
- Extend our fully implicit parallel approach to adaptive mesh refinement

VORPAL Project(s)

- Currently work on accelerator cavities simulations (Crab cavity) - $3D$ problems
- There's a need to refine spatial resolution - going from small clusters ($np < 100$) to IBM BG/L systems
- Issues on a BG/L system? Compilers (IBM), HDF5, Python - cross-compiling is somewhat problematic...
- Fortunately, it's possible to overcome these issues for some classes of simulations
- Yee algorithm, FDTD (finite difference time domain), explicit, leap-frog type of time stepping with finite difference discretization of spatial components, PIC

VORPAL Speedup



Simple EM wave propagation, problem sizes (192×128, $1536 \times 512 \times 512$, $1536 \times 1024 \times 1024$, and $3072 \times 1024 \times 1024$ cells) weak scaling with speedup $t_{np=1}/(t_{np}/np)$, I/O isn't included.

Some Observations and Future Work

- PIC: particle-dominated approach vs. field-dominated approach. Particle-dominated approach may scale better due to larger local computation load ("pushing" particles). Field-dominated approach implies weaker scalability?
- Use PETSc (Poisson solves, matrix-vector products)?
- Parallel efficiency needs improvement
- Fine-tune the communication pattern
- Improve I/O - $> 100GB$ files can be produced by VORPAL simulations
- Plan to make better use of TAU
- Would like to get in the neighborhood of 300 TFLOPS sometime soon
- Immediate: do something about Python on BG/L compute nodes

References and Acknowledgements

- S. Ovtchinnikov, F. Dobrian, X.-C. Cai, and D. Keys, Additive Schwarz-based fully coupled implicit methods for resistive Hall magnetohydrodynamic problems, J. Comput. Phys., (2007), to appear

- S. Ovtchinnikov, F. Dobrian, X.-C. Cai, and D. Keyes, Domain-decomposed fully coupled implicit methods for a magnetohydrodynamics problem, Lecture Notes in Computational Science and Engineering, Springer, (2006)

- S. Ovtchinnikov and X.-C. Cai, One-level Newton-Krylov-Schwarz algorithm for unsteady nonlinear radiation diffusion problem, Numer. Lin. Alg. Applics., 11(2004), pp.867–881

Computer time was provided in part by NSF MRI Grant CNS-0421498, NSF MRI Grant CNS-0420873, NSF MRI Grant CNS-0420985, NSF sponsorship of the National Center for Atmospheric Research, the University of Colorado, and a grant from the IBM Shared University Research program. The research was supported in part by the Department of Energy under cooperative agreements DE-FC02-01ER25479 and DE-FC02-04ER25595, and in part by the National Science Foundation under grants CCR-0219190, ACI-0305666, and CCF-03-52334.