Experiences and Challenges Scaling PFLOTRAN, a PETSc-based Code For Implicit Solution of Subsurface Reactive Flow Problems, Towards the Petascale on Cray XT systems

Richard Tran Mills

Computational Earth Sciences Group

Computer Science and Mathematics Division

Oak Ridge National Laboratory

+ Lots of other people

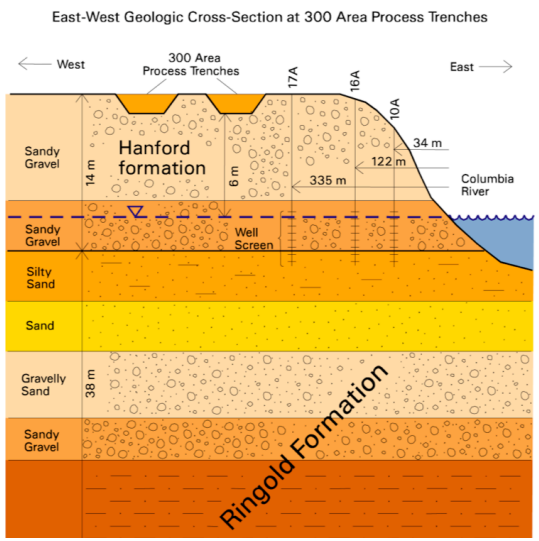
Managed by UT-Battelle
for the Department of Energy

Oak Ridge National Laboratory

Introduction

- Funded by SciDAC-II project, "Modeling Multiscale-Multiphase-Multicomponent Subsurface Reactive Flows using Advanced Computing", involving subsurface scientists, applied mathematicians, and computer scientists at several institutions:
  - LANL: Peter Lichtner (PI), Chuan Lu, Bobby Philip, David Moulton
  - ORNL: Richard Mills
  - ANL: Barry Smith
  - PNNL: Glenn Hammond
  - U. Illinois: Al Valocchi
- Also collaborating with SciDAC PERI:
  - NC State: G. (Kumar) Mahinthakumar, Vamsi Sripathi
  - ORNL: Pat Worley
- Project goals:
  - Develop a next-generation code (PFLOTRAN) for simulation of multiscale, multiphase, multicomponent flow and reactive transport in porous media.
  - Apply it to field-scale studies of (among others)
    - Geologic CO2 sequestration,
    - Radionuclide migration at Hanford site, Nevada Test Site

Motivating example -- Hanford 300 Area



- At the 300 area, U(VI) plumes continue to exceed drinking standards.
- Calculations predicted cleanup by natural attenuation years ago!
- Due to long in-ground residence times, U(VI) is present in complex, microscopic inter-grain fractures, secondary grain coatings, and micro-porous aggregates. (Zachara et al., 2005).
- Models assuming constant K_d (ratio of sorbed mass to mass in solution) do not account for slow release of U(VI) from sediment grain interiors through mineral dissolution and diffusion along tortuous pathways.
- In fact, the K_d approach implies behavior entirely contrary to observations!
- We must accurately incorporate millimeter scale effects over a domain measuring approximately 2000 x 1200 x 50 meters!

Fundamental challenge:

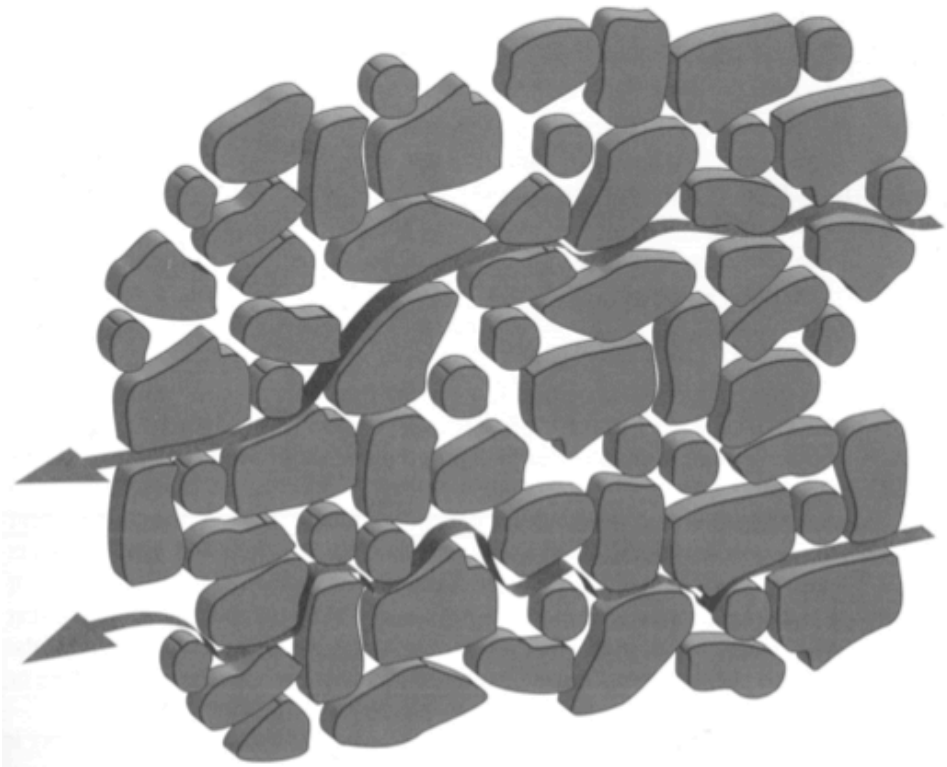
- Need to capture millimeter-scale (or smaller) processes within kilometer scale domains! (Similar variations in time scales.)
- Discretizing 2km x 1 km x 500 m domain onto cubic millimeter grid means 10^18 computational nodes!
- Address the problem via
  - Massively parallel computing
    - Continuing development of PFLOTRAN code
  - Multi-continuum ("sub-grid") models
    - Multiplies total degrees of freedom in primary continuum by number of nodes in sub-continuum
  - Adaptive mesh refinement
    - Allows front tracking
    - Introduce multi-continuum models only where needed

Outline

- Subsurface flow and reactive transport
- Numerical discretization of governing eqns.
- Parallel implementation (solvers, code arch.)
- Computation phase performance
- I/O performance
- Future directions

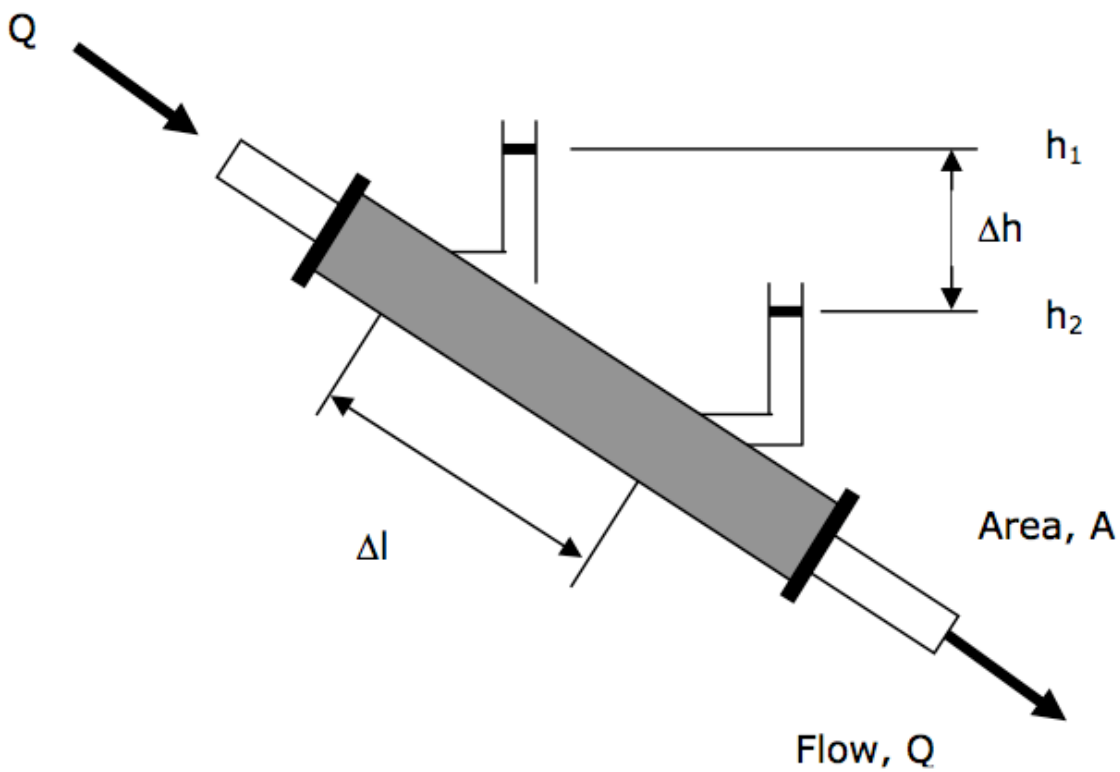
Porous media flow

- **Continuity equation (mass conservation)**
- **Darcy's law in place of momentum eqn.**



Porous media flow

- **Continuity equation (mass conservation)**
- **Darcy's law in place of momentum eqn.**



$$q = \frac{Q}{A} = -K\nabla h$$

PFLOTRAN governing equations

Mass Conservation: Flow Equations

$$\frac{\partial}{\partial t}(\phi s_\alpha \rho_\alpha X_i^\alpha) + \nabla \cdot \left[q_\alpha \rho_\alpha X_i^\alpha - \phi s_\alpha D_i^\alpha \rho_\alpha \nabla X_i^\alpha \right] = Q_i^\alpha$$

$$q_\alpha = -\frac{k k_\alpha}{\mu_\pi} \nabla (p_\alpha - W_\alpha \rho_\alpha g z) \qquad p_\alpha = p_\beta - p_{c,\alpha\beta}$$

Energy Conservation Equation

$$\frac{\partial}{\partial t}\left[\phi \sum_\alpha s_\alpha \rho_\alpha U_\alpha + (1-\phi)\rho_r c_r T \right] + \nabla \cdot \left[\sum_\alpha q_\alpha \rho_\alpha H_\alpha - \kappa \nabla T \right] = Q_e$$

Multicomponent Reactive Transport Equations

$$\frac{\partial}{\partial t}\left[\phi \sum_\alpha s_\alpha \Psi_j^\alpha \right] + \nabla \cdot \left[\sum_\alpha \Omega_\alpha \right] = -\sum_m v_{jm} I_m + Q_j$$

Total Concentration

$$\Psi_j^\alpha = \delta_{\alpha l} C_j^\alpha + \sum_i v_{ji} C_i^\alpha$$

Total Solute Flux

$$\Omega_j^\alpha = (-\tau \phi s_\alpha D_\alpha \nabla + q_\alpha)\Psi_j^\alpha$$

Mineral Mass Transfer Equation

$$\frac{\partial \phi_m}{\partial t} = V_m I_m \qquad \phi + \sum_m \phi_m = 1$$

PFLOTRAN governing equations

Mass Conservation: Flow Equations

$$\frac{\partial}{\partial t}(\phi s_\alpha \rho_\alpha X_i^\alpha) + \nabla \cdot \left[q_\alpha \rho_\alpha X_i^\alpha - \phi s_\alpha D_i^\alpha \rho_\alpha \nabla X_i^\alpha \right] = Q_i^\alpha$$

$$q_\alpha = -\frac{k k_\alpha}{\mu_\pi} \nabla (p_\alpha - W_\alpha \rho_\alpha g z)$$

$$p_\alpha = p_\beta - p_{c,\alpha\beta}$$

Darcy's law
(homogenized momentum eq.)

Energy Conservation Equation

$$\frac{\partial}{\partial t}\left[\phi \sum_\alpha s_\alpha \rho_\alpha U_\alpha + (1-\phi)\rho_r c_r T \right] + \nabla \cdot \left[\sum_\alpha q_\alpha \rho_\alpha H_\alpha - \kappa \nabla T \right] = Q_e$$

Multicomponent Reactive Transport Equations

$$\frac{\partial}{\partial t}\left[\phi \sum_\alpha s_\alpha \Psi_j^\alpha \right] + \nabla \cdot \left[\sum_\alpha \Omega_\alpha \right] = -\sum_m v_{jm} I_m + Q_j$$

Total Concentration

$$\Psi_j^\alpha = \delta_{\alpha l} C_j^\alpha + \sum_i v_{ji} C_i^\alpha$$

Total Solute Flux

$$\Omega_j^\alpha = (-\tau \phi s_\alpha D_\alpha \nabla + q_\alpha)\Psi_j^\alpha$$

Mineral Mass Transfer Equation

$$\frac{\partial \phi_m}{\partial t} = V_m I_m$$

$$\phi + \sum_m \phi_m = 1$$

PFLOTRAN governing equations

Mass Conservation: Flow Equations

$$\frac{\partial}{\partial t}(\phi s_\alpha \rho_\alpha X_i^\alpha) + \nabla \cdot \left[q_\alpha \rho_\alpha X_i^\alpha - \phi s_\alpha D_i^\alpha \rho_\alpha \nabla X_i^\alpha \right] = Q_i^\alpha$$

$$q_\alpha = -\frac{k k_\alpha}{\mu_\pi} \nabla (p_\alpha - W_\alpha \rho_\alpha g z) \qquad p_\alpha = p_\beta - p_{c,\alpha\beta}$$

Relative permeability depends on saturation -- introduces nonlinearity

Energy Conservation Equation

$$\frac{\partial}{\partial t}\left[\phi \sum_\alpha s_\alpha \rho_\alpha U_\alpha + (1-\phi)\rho_r c_r T \right] + \nabla \cdot \left[\sum_\alpha q_\alpha \rho_\alpha H_\alpha - \kappa \nabla T \right] = Q_e$$

Multicomponent Reactive Transport Equations

$$\frac{\partial}{\partial t}\left[\phi \sum_\alpha s_\alpha \Psi_j^\alpha \right] + \nabla \cdot \left[\sum_\alpha \Omega_\alpha \right] = -\sum_m \nu_{jm} I_m + Q_j$$

Total Concentration

$$\Psi_j^\alpha = \delta_{\alpha l} C_j^\alpha + \sum_i \nu_{ji} C_i^\alpha$$

Total Solute Flux

$$\Omega_j^\alpha = (-\tau \phi s_\alpha D_\alpha \nabla + q_\alpha)\Psi_j^\alpha$$

Nonlinear function of the concentrations of primary chemical components

Mineral Mass Transfer Equation

$$\frac{\partial \phi_m}{\partial t} = V_m I_m$$

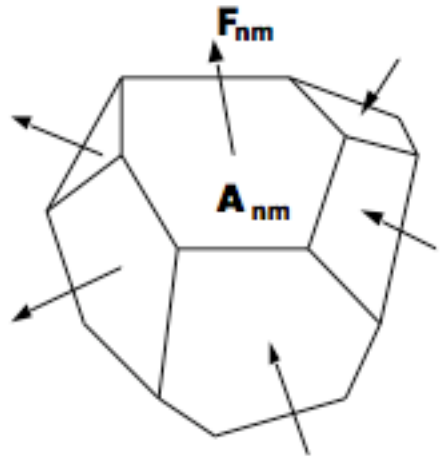
$$\phi + \sum_m \phi_m = 1$$

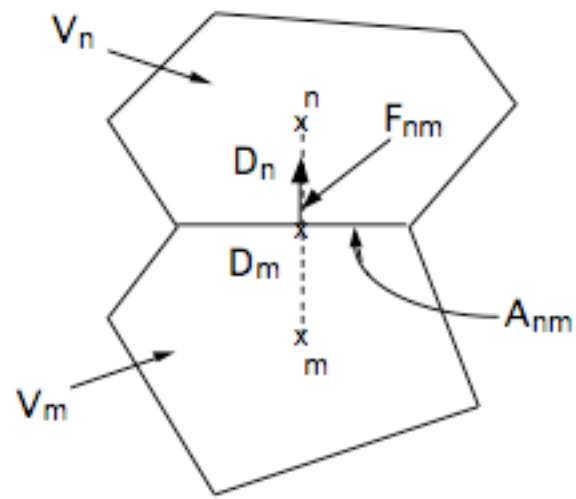
Integrated finite-volume discretization

Form of governing equation:

$$\frac{\partial A}{\partial t} + \nabla \cdot F = S \qquad F = qpX - \phi Dp\nabla X$$

Integrated finite-volume discretization





Discretized residual equation:

$$R_n = \left(A_n^{k+1} - A_n^k\right)\frac{V_n}{\Delta t} + \sum_{n'} F_{nn'} A_{nn'} - S_n V_n$$

$$F_{nn'} = (q\rho)_{nn'} X_{nn'} - (\phi D\rho)_{nn'} \frac{X_n - X_{n'}}{d_n + d_{n'}}$$

(Inexact) Newton iteration: $\sum_{n'} J^i_{nn'} \delta x^{i+1}_{n'} = -R^i_n \qquad J^i_{nn'} = \frac{\partial R^i_n}{\partial x^i_{n'}}$

IFV solver time-step

- **At each time step:**
  - **Calculate residual** $$R_n = \left(A_n^{k+1} - A_n^k\right)\frac{V_n}{\Delta t} + \sum_{n'} F_{nn'} A_{nn'} - S_n V_n$$
    - **For each node, calculate accumulation term**
    - **For each connection, calculate flux term**

$$F_{nn'} = (q\rho)_{nn'} X_{nn'} - (\varphi D\rho)_{nn'} \frac{X_n - X_{n'}}{d_n + d_{n'}}$$

    - **Calculate source-sink term for appropriate connections**
  - **Calculate Jacobian** $$J_{nn'}^i = \frac{\partial R_n^i}{\partial x_{n'}^i}$$
    - **Via finite differences**
    - **…or analytically (analogous to residual calculation)**
  - **Solve linear system**

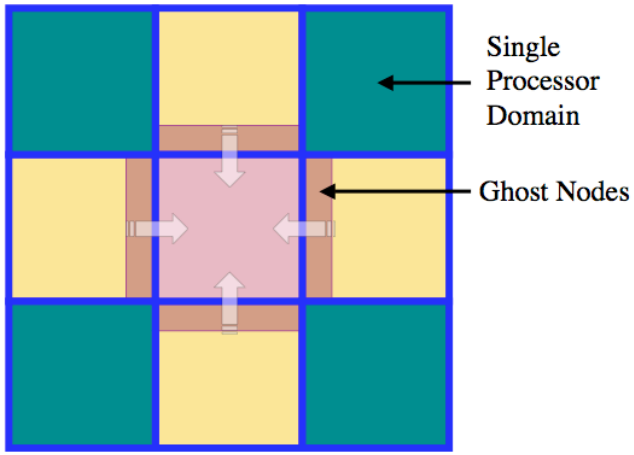
$$\sum_{n'} J_{nn'}^i \delta x_{n'}^{i+1} = -R_n^i$$

Outline

- Subsurface flow and reactive transport
- Numerical discretization of governing eqns.
- Parallel implementation (solvers, code arch.)
- Computation phase performance
- I/O performance
- Future directions

Domain-decomposition parallelism

- **PFLOTRAN parallelism comes from domain decomposition.**
- **Each processor responsible for nodes in one subdomain. (Plus associated vector, matrix entries)**



- **Accumulation terms are easy: Each processor calculates terms for the nodes it owns.**
- **Flux terms are not!**
  - **Must calculate fluxes across subdomain boundaries.**
  - **Scatter/gather of *ghost nodes* required (*halo exchanges*)**

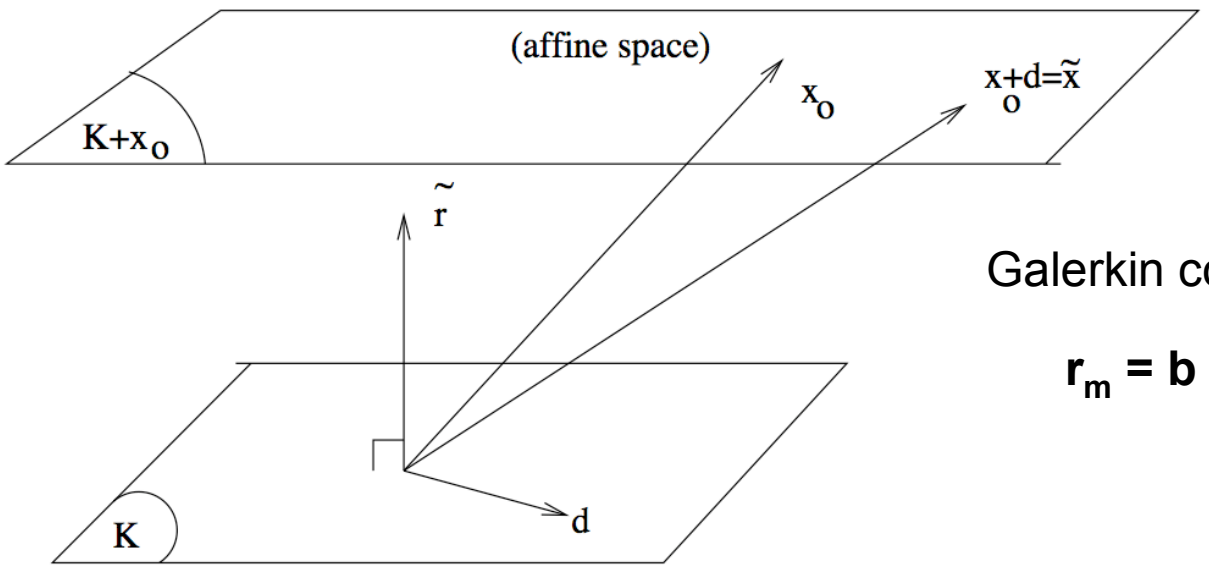
Solving the linear systems

- **Linear system solve often accounts for > 90% of time.**
- **Linear systems are large (N=total degrees of freedom) and sparse (very few nonzeros)**
- **Do not want to do Gaussian elimination (LU factorization):**
  - **Fill-in can eliminate sparsity (unmanageable memory cost)**
  - **Extremely difficult to parallelize!**
    - **In general, cannot do row k+1 w/o first doing row k**

$$\begin{bmatrix} x & x & & & x & & & x \\ x & x & x & & & & & \\ & x & x & x & & & & \\ & & x & x & & & & \\ x & & & & x & x & & x \\ & & & & x & x & x & \\ & & & & & x & x & \\ x & & & & x & & & x \end{bmatrix} \xrightarrow{\text{LU factorization}} \begin{bmatrix} x & x & & & x & & & x \\ x & x & x & & \bullet & & & \bullet \\ & x & x & x & \bullet & & & \bullet \\ & & x & x & \bullet & & & \bullet \\ x & \bullet & \bullet & \bullet & x & x & & x \\ & & & & x & x & x & \bullet \\ & & & & & x & x & \bullet \\ x & \bullet & \bullet & \bullet & x & \bullet & \bullet & x \end{bmatrix}$$

Krylov subspace methods

- **Iteratively solve Ax=b starting from initial guess x_0.**
- **Build affine subspace $x_0 + K_m$**
  - **$K_m(A,r_0) = \text{span}(r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0)$**
  - **Where $r_0 = b - Ax_0$ (initial residual)**
- **Extract approximations x_m from $x_0 + K_m$ subject to orthogonality constraint $r_m = b - Ax_m \perp L_m$**



Galerkin condition $L_m = K_m$

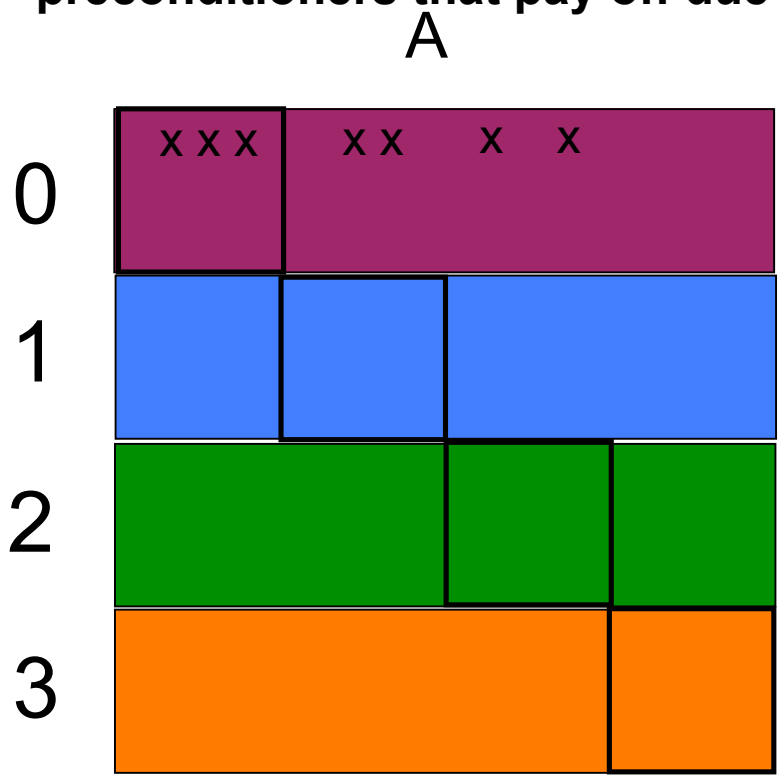
$r_m = b - Ax_m \perp K_m$

Krylov method implementation

- Krylov methods require few computational kernels:
  - Vector update: $y \leftarrow y + \alpha \cdot x$
  - Dot product: $\alpha \leftarrow x \cdot y$
  - Matrix-vector multiply: $y \leftarrow Av$
- Low storage requirements
  (In fact, storage for A is optional!)
- Can require fewer FLOPs than direct solve.
- Easily parallelized
  - Vector updates trivial
  - Dot products require MPI_Allreduce()
  - Mat-vecs require gather/scatter

Domain decomposition preconditioners

- Need preconditioning to improve Krylov solver performance.
- Many powerful preconditioners (e.g., ILU) are difficult to compute in parallel.
- Domain decomposition preconditioners are example of weaker preconditioners that pay off due to better parallelism:



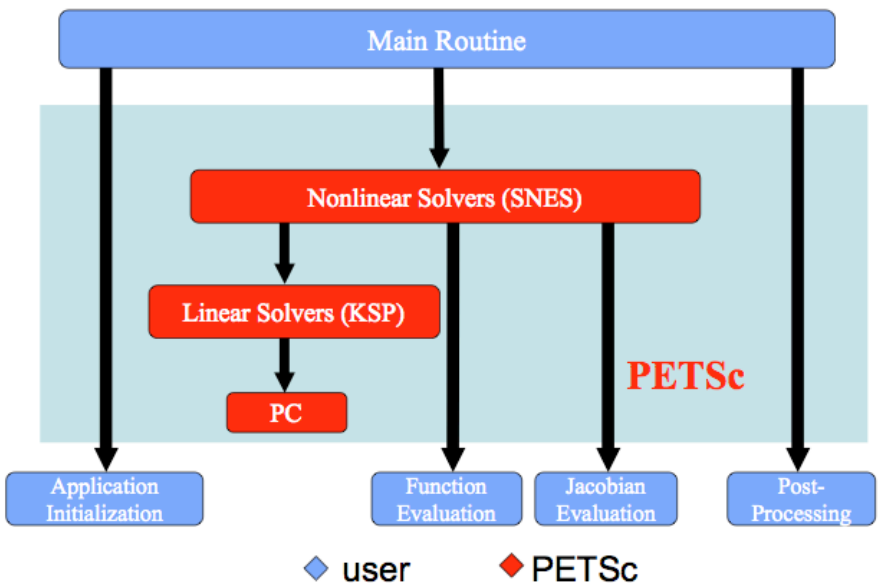
Block-Jacobi:

- Solve equations ignoring off-domain (off-processor) connections.
- No communication required!
- Apply whatever method you like on each each subdomain;
  For example, ILU.
- Much less powerful than ILU on entire domain…
- …but may be considerably faster in wall-clock time.

PFLOTRAN architecture

- Built on top of PETSc, which provides
  - Object-oriented management of parallel data structures,
    - Create parallel objects (e.g., matrices and vectors) over a set of processors.
    - Methods (e.g., `MatMult`())called on those objects handle parallel coordination.
  - Parallel solvers and preconditioners,
  - Efficient parallel construction of Jacobians and residuals
- We provide
  - Initialization, time-stepping, equations of state, post-processing
  - Functions to form residuals (and, optionally, Jacobians) on a local patch (PETSc routines handle patch formation for us)

Flow of Control for PDE Solution



Building PFLOTRAN with PETSc

- PETSc has allowed us to develop a complex, scalable code in very little time:
  - Initial PTRAN by Glenn Hammond for DOE CSGF practicum
  - Initial PFLOW by Richard Mills for DOE CSGF practicum
  - Subsequent development of multiphase modules by Peter Lichtner and Chuan Lu during Lu's postdoc
  - Rapid improvements during first year of SciDAC
- PETSc is more than just a "solvers package"
- Provides a comprehensive framework:
  - Parallel mesh and associated linear algebra object management
  - Nonlinear solvers
  - Linear (iterative) solvers
  - Performance logging and debugging
  - Interfaces to many other packages

PETSc nonliner solver framework

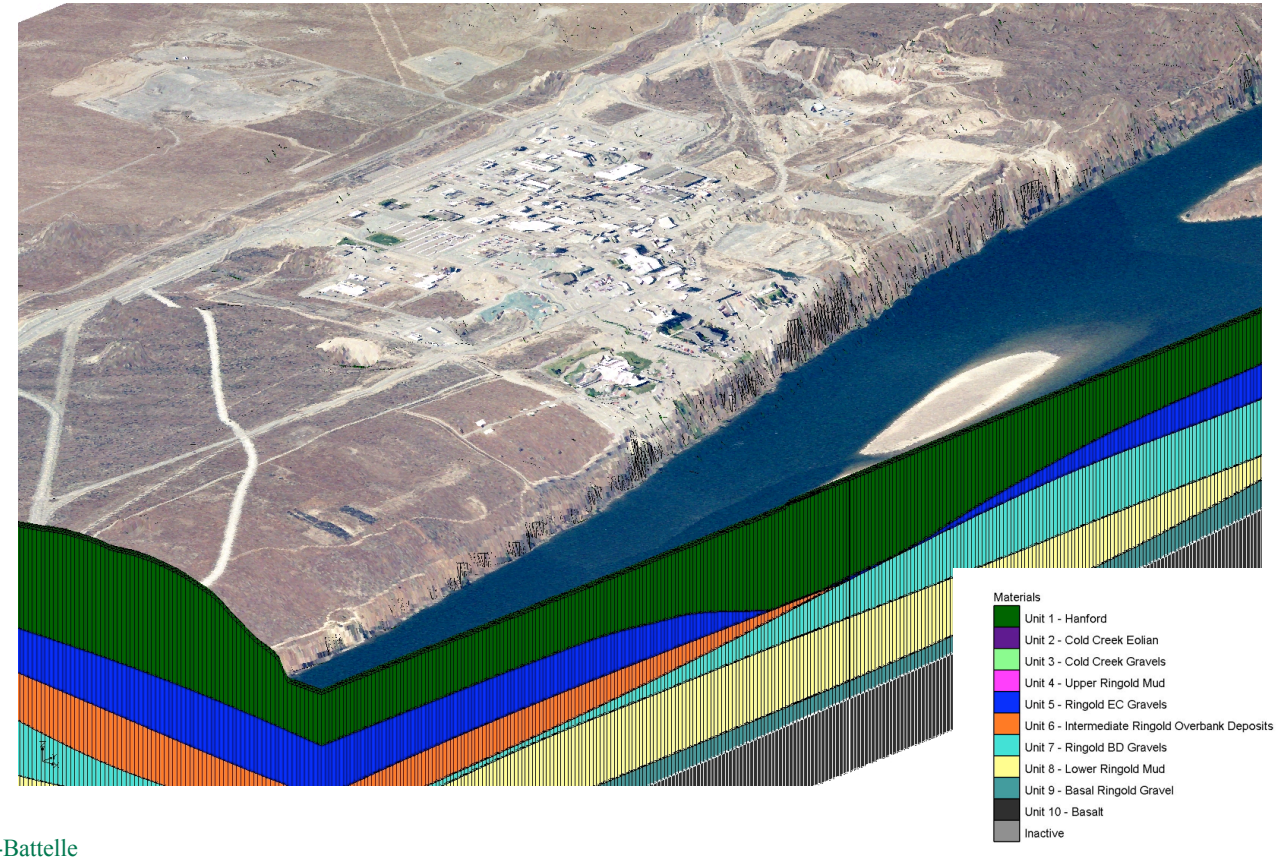
- **Inexact Newton with various globalization strategies (line search, trust region)**
- **User provides SNES with**
  - **Residual:** `PetscErrorCode (*func) (SNES snes, Vec x, Vec r, void *ctx)`
  - **Jacobian:** `PetscErrorCode (*func) (SNES snes, Vec x, Mat *J, Mat *M, MatStructure *flag, void *ctx)`
- **Our functions expect a patch (local, contiguous region at single refinement level).**
- **Assembly of patch handled by PETSc (or SAMRAI in case of AMR)**

Outline

- Subsurface flow and reactive transport
- Numerical discretization of governing eqns.
- Parallel implementation (solvers, code arch.)
- Computation phase performance
- I/O performance
- Future directions

Hanford 300 Area strong scaling benchmark

- Migration of hypothetical uranium plume
- 1350 x 2500 x 20 m domain
- Complex stratigraphy



Hanford Modeling Challenges

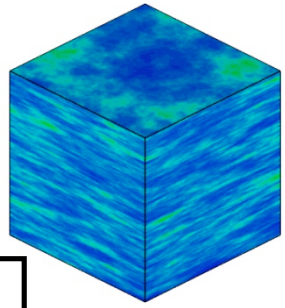
- 3D Domain: length and time scales
  - field scale domain (~km)
  - hourly river fluctuations, ~ 1000 year predictions
  - fast flow rates (5 km/y)

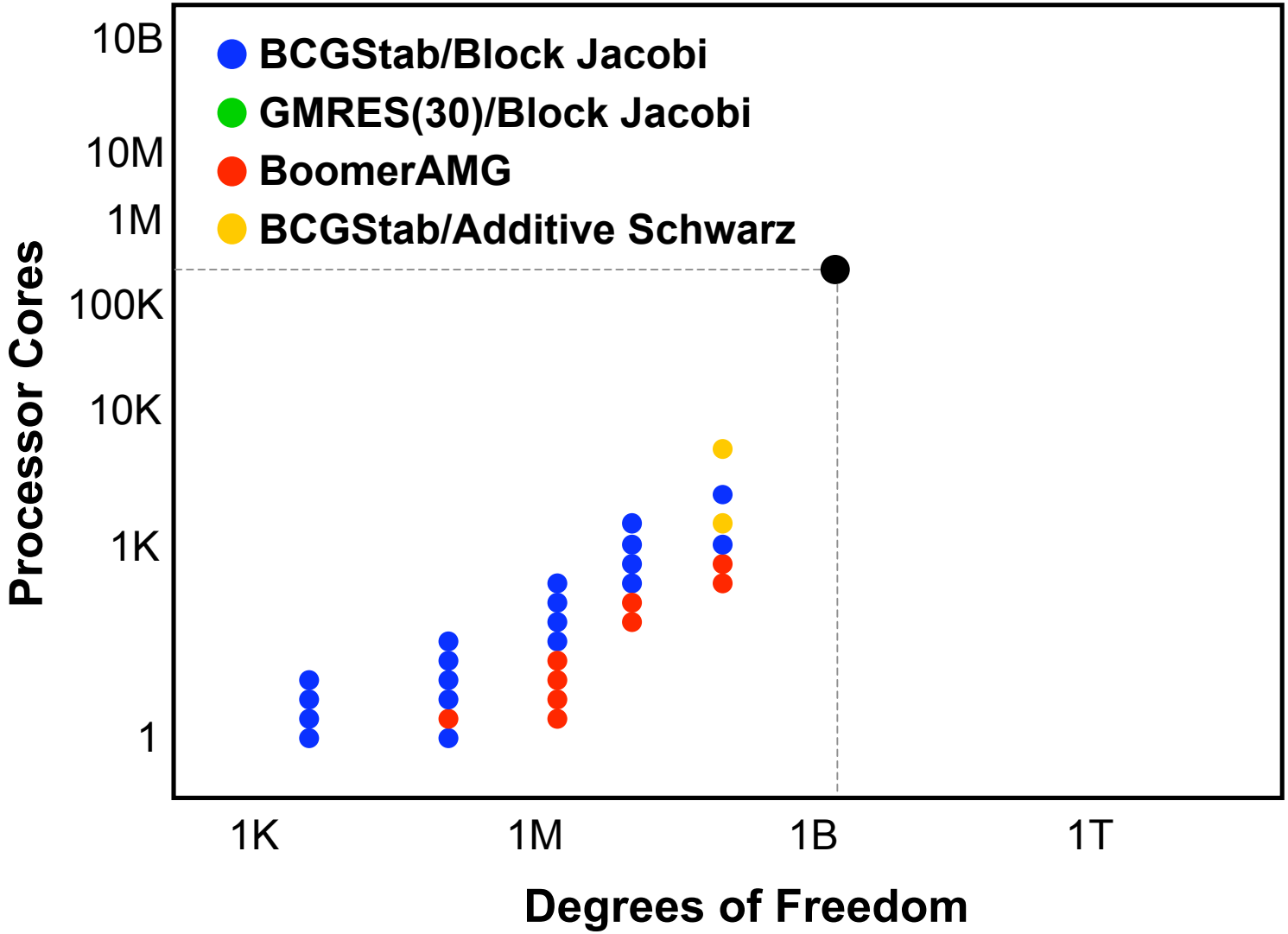
- Complex chemistry: Na-K-Ca-Fe-Mg-Br-N-CO_2-P-S-Cl-Si-U-Cu-H_2O (~15 primary species)
- Multiscale processes (μm-m)
- Highly heterogeneous sediments
  - fine sand, silt; coarse gravels; cobbles
- Variably saturated environment
- Frequent and large river-stage fluctuations

Solvers for Hanford 300 benchmark

- Inexact Newton method w/ fixed tolerances
- BiCGstab linear solver
- Preconditioned with block-Jacobi
- ILU(0) applied on each block
- Nothing fancy, but we have been surprised by how well this has worked!
- We have also tried simple geometric multigrid; algebraic multigrid with Hypre, but have not yet gotten better performance out of them.

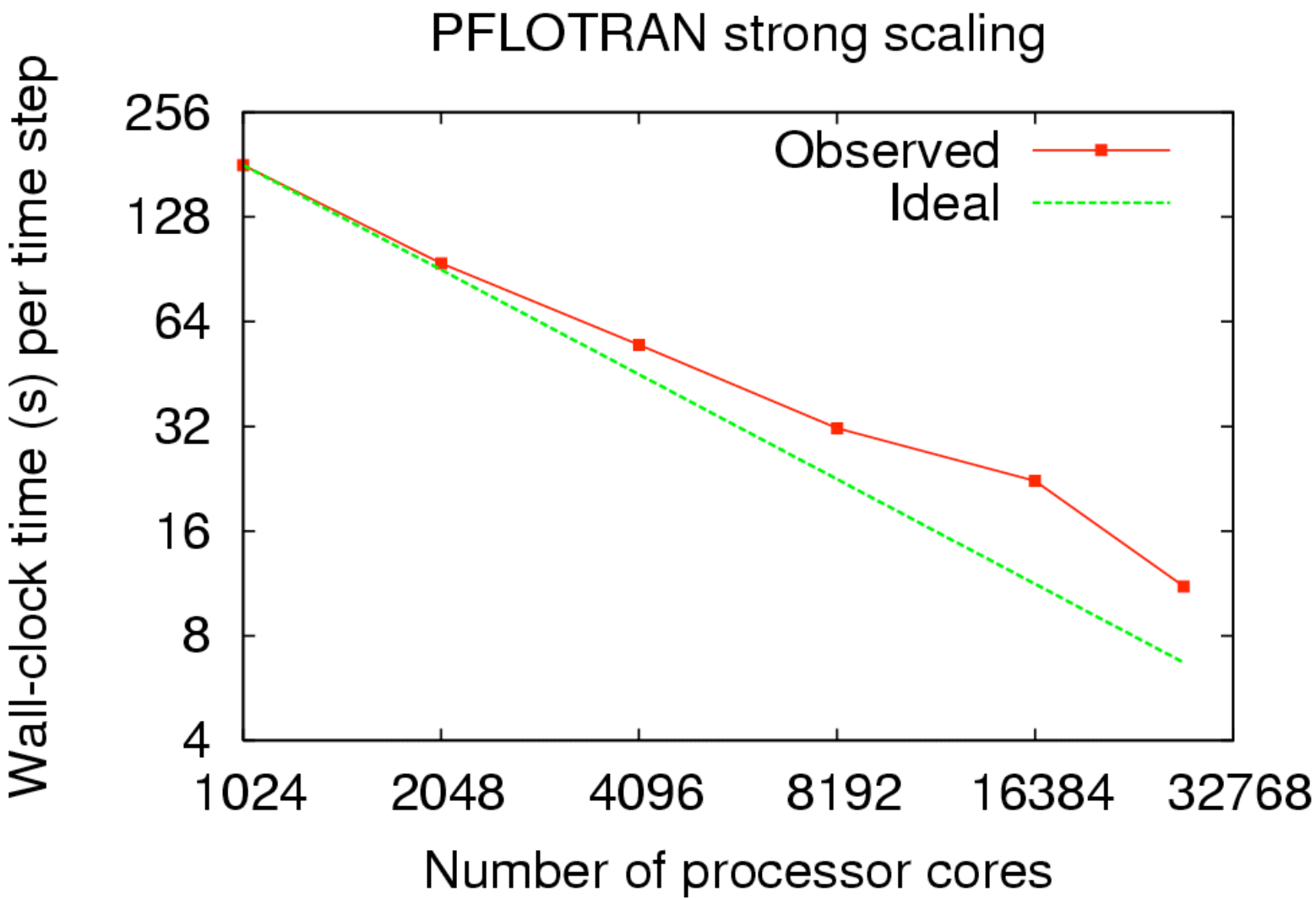
Jaguar: Heterogeneous geology





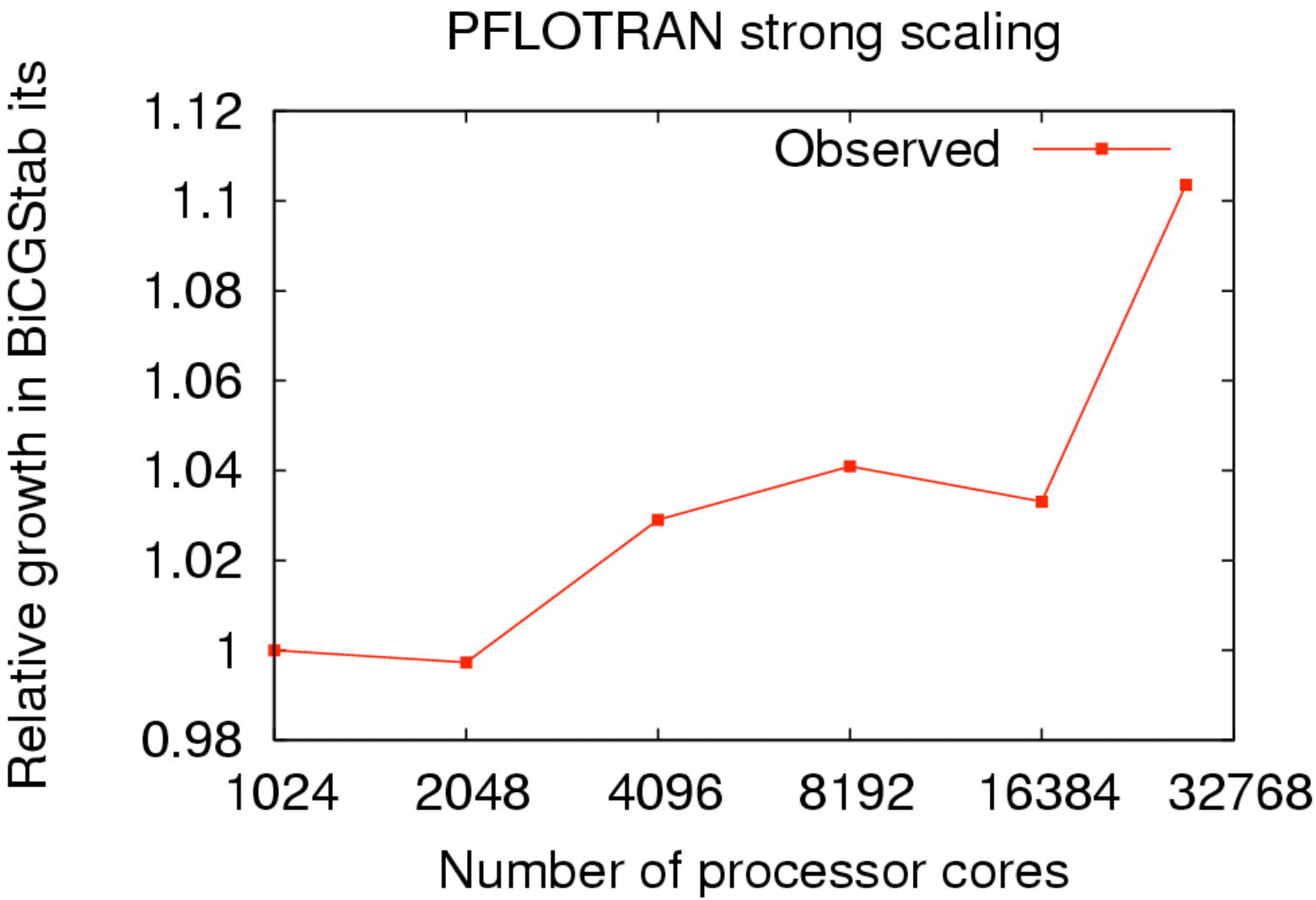
Hanford 300 area: Strong scaling

- **270 million DoF (1350 x 2500 x 80 grid)**



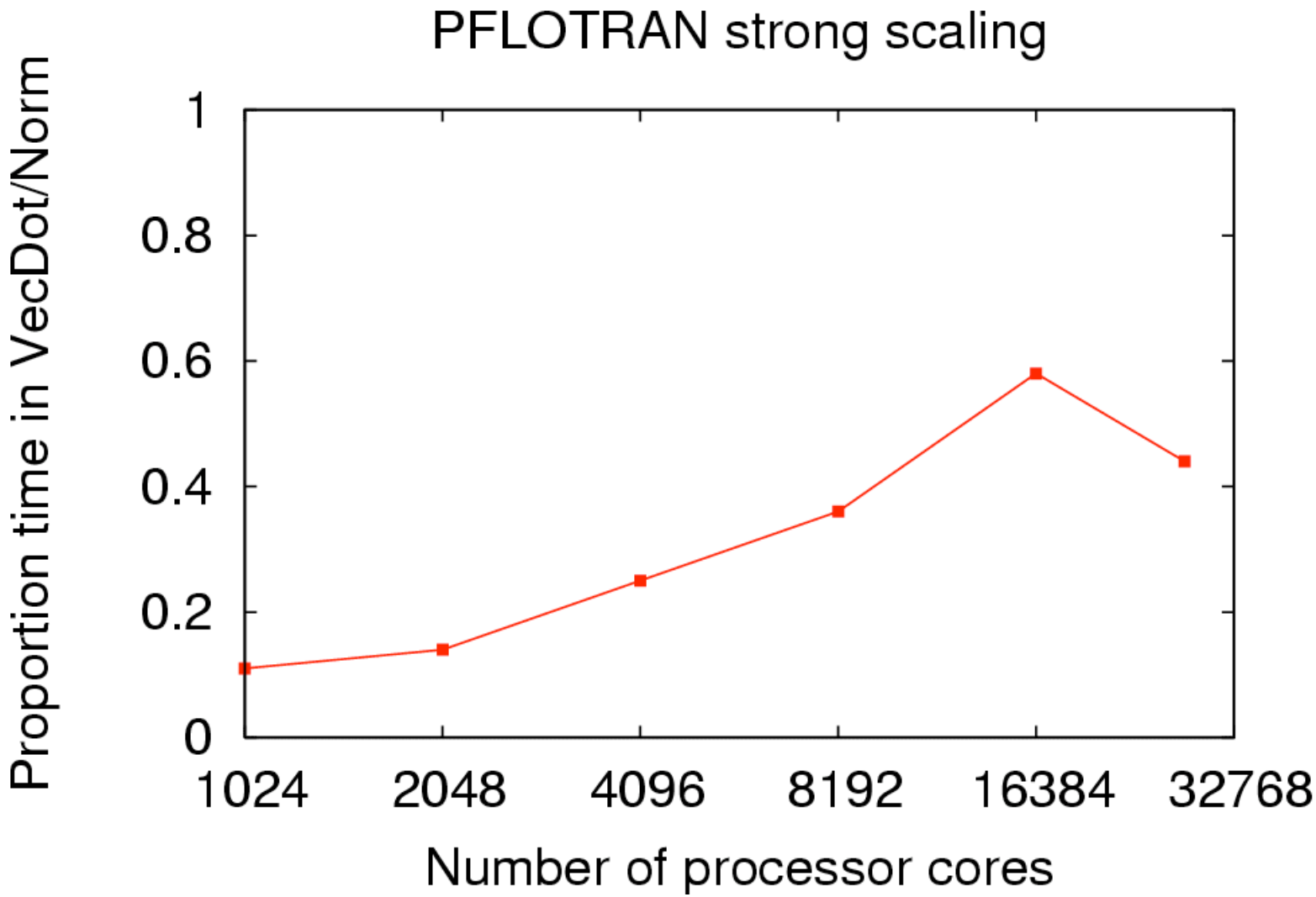
270 M DoF: Flow: BiCGstab its

- **270 million DoF (1350 x 2500 x 80 grid)**



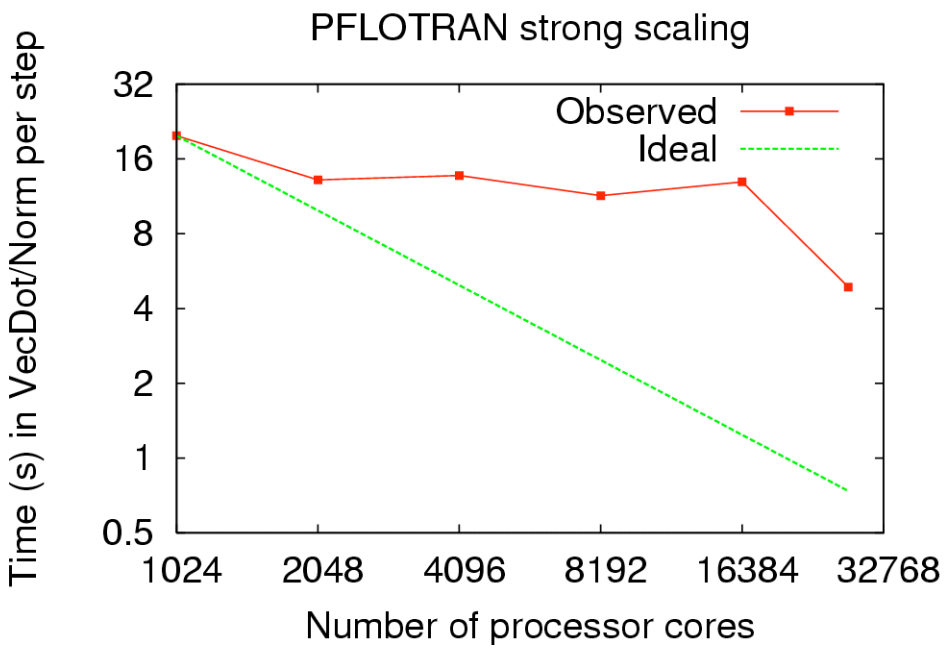
Hanford 300 area: Strong scaling

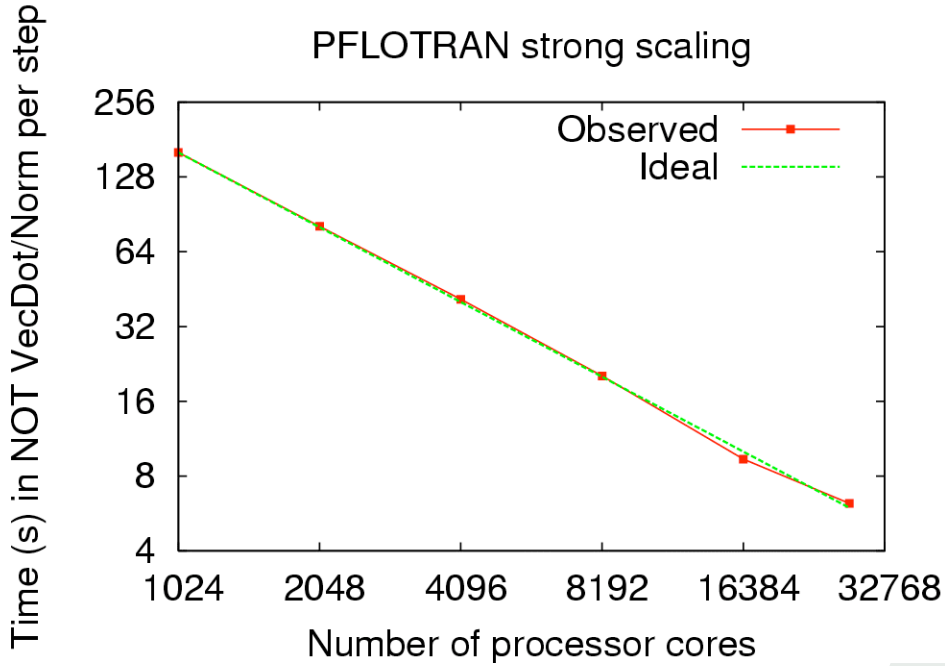
- **270 million DoF (1350 x 2500 x 80 grid)**



Hanford 300 area: Strong scaling

- **270 million DoF (1350 x 2500 x 80 grid)**





BiCGStab improvements

- **Cost of MPI_Allreduce() calls inside Krylov solver are big scalability barrier**
- **Original PETSc BiCGstab had 4 allreduces/iteration (including convergence check)**
- **Reworked PETSc BiCGstab has 3**
- **Also added "Improved" BiCGStab (IBCGS)**
  - **Considerably more complicated; requires transpose matrix vector product, extra vector operations**
  - **Only 2 MPI_Allreduce()'s per iteration required**
  - **By lagging residual norm calc., can wrap everything into one MPI_Allreduce(), at cost of doing one additional IBCGS iteration**

16384 core,
30 time step run:

Group	BCGS	IBCGS
MPI_SYNC	196	120
MPI	150	79
User	177	200

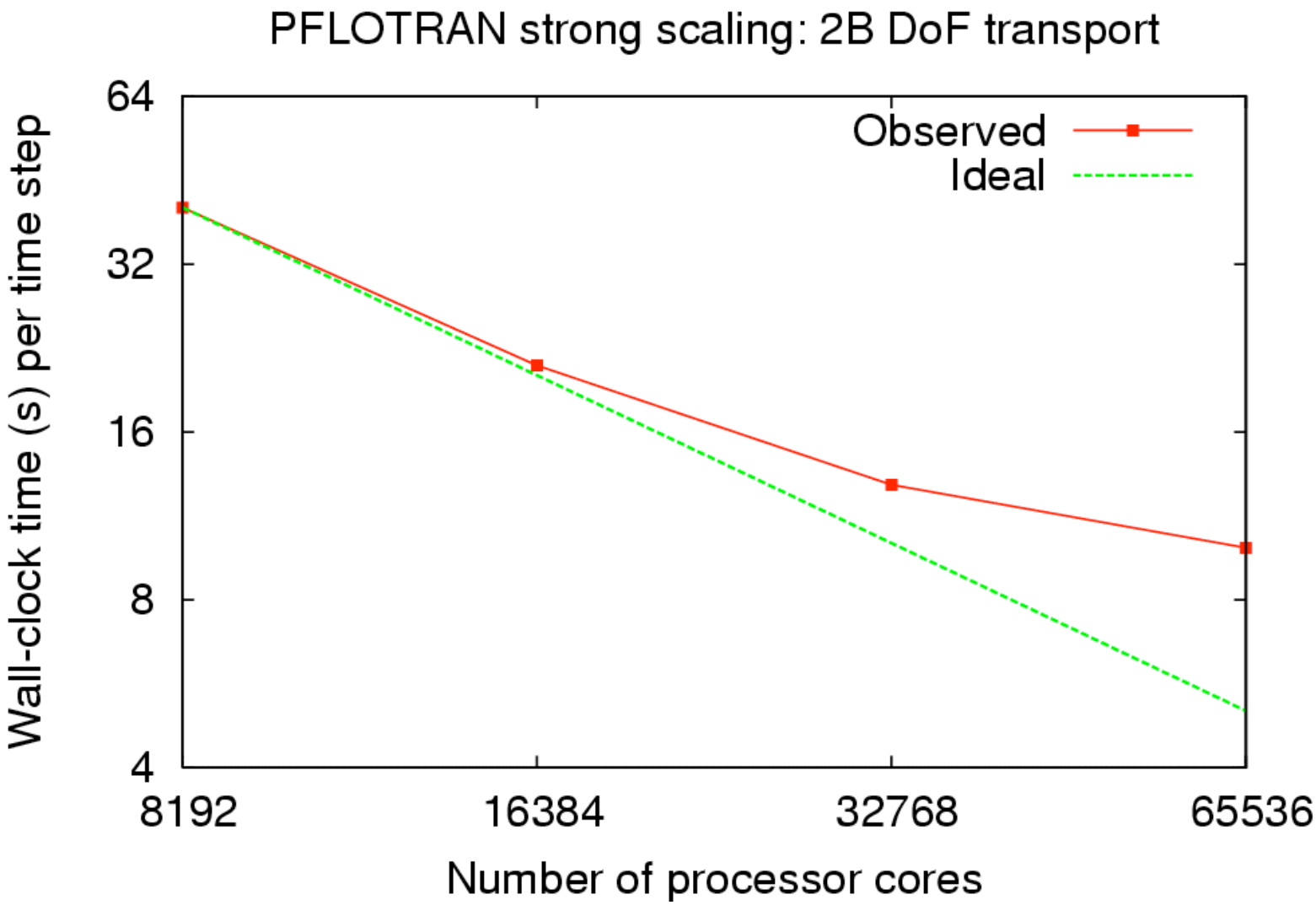
Possible near-term improvements

	Savings	Required	Time (focused)
Reorganize MatSolve	3-4%	Coding	20-30 hours
BiCGStab w/ 1 reduction DONE	20%	Coding	10 hours
Eisenstat-Walker	20-30%	Thinking and experimentation	24 hours
Geometric multigrid	0-50%	Coding, experimentation, thinking	30 (?) hours
GOOD preconditioner	0-70%	Brilliance	???

Maybe improve exact same run by 35% in next six months.

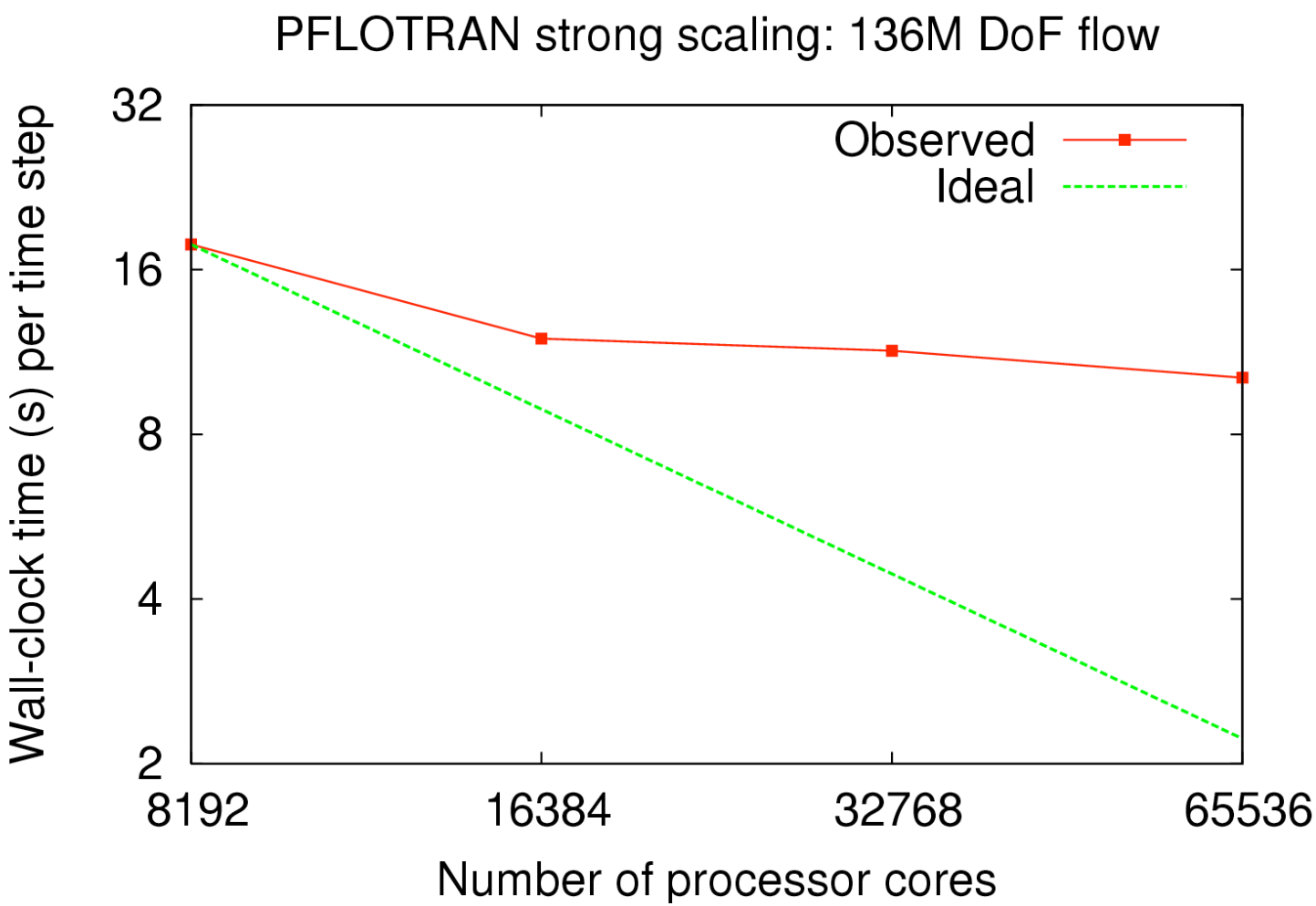
2B DoF: Flow + Transport: Transport

- **2 billion DoF (850 x 1000 x 160 grid, 15 species)**
  (Limit of 32-bit indices!)



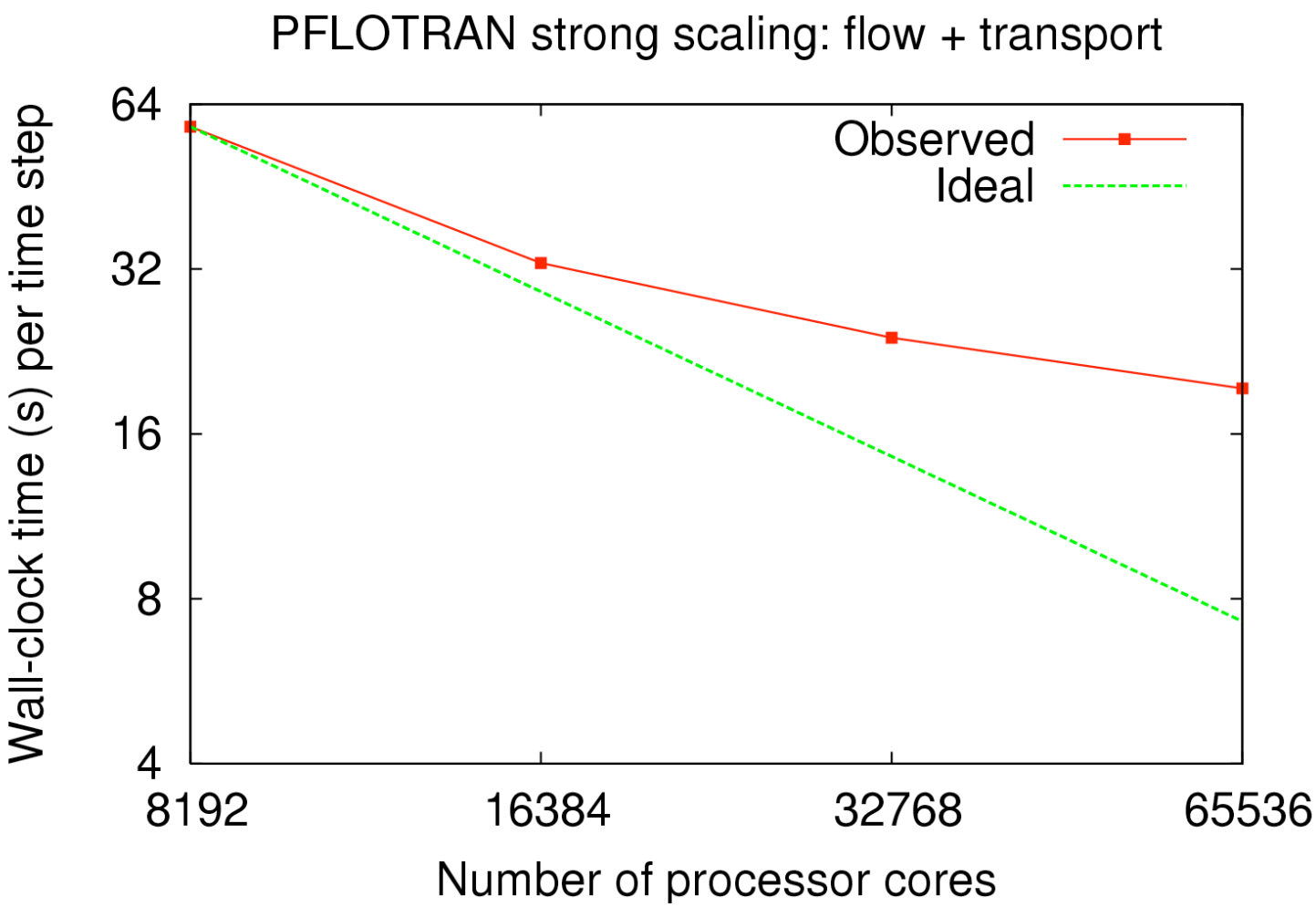
2B DoF: Flow + Transport: Flow

- **136M DoF (850 x 1000 x 160 grid, pressure)**



2B DoF: Flow + Transport: Combined

- **2 billion DoF (850 x 1000 x 160 grid, 15 species)**
  (Limit of 32-bit indices!)



Flow solver as bottleneck

- Flow problem is only 1/15 size of transport
- Disparity becomes larger with more components, kinetic sorption
- At some point, may want to redundantly solve flow problem on subsets of procs (like coarse grid solve in parallel multigrid)
- May not be necessary:
  - Cost of transport solve dominates
  - In many cases, need finer time resolution in transport

Outline

- Subsurface flow and reactive transport
- Numerical discretization of governing eqns.
- Parallel implementation (solvers, code arch.)
- Computation phase performance
- I/O performance
- Future directions

Improvements to I/O

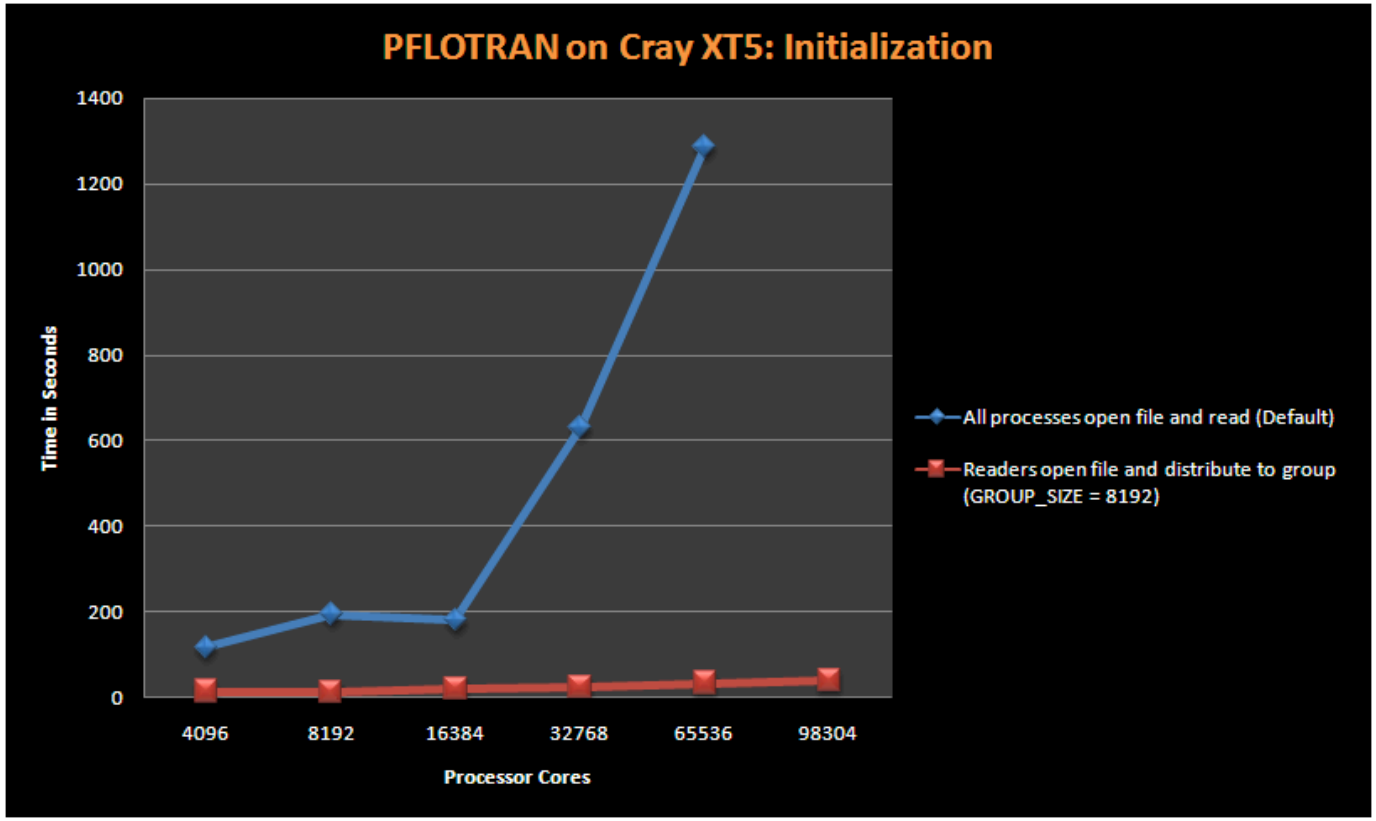
- Above 8000K cores, serial I/O becomes impractical
- Added parallel IO:
  - MPI-IO backend to PETSc Viewers (for checkpoints)
  - Parallel HDF5 input/output (for input files, simulation output files)
- This fixed lots of problems, but initialization was still a big problem above 16384 cores
  - Lots of reads from single, large HDF5 file
    - Every process parses entire file
    - Each process read chunk of data set starting from its offset
  - Problem: Only 1 meta-data server! All the file/open closes hammer it!
  - Solution: only subset of procs read, then broadcast to others

Improved initialization IO

```
group_size = HDF5_GROUP_SIZE (e.g., 1024)
call MPI_Init(ierr)
call MPI_Comm_rank(MPI_COMM_WORLD,global_rank,ierr)
MPI_comm iogroup, readers
global_comm = MPI_COMM_WORLD

color = global_rank / group_size
key = global_rank
call MPI_Comm_split(global_comm,color,key,iogroup,ierr)

   if (mod(global_rank,group_size) == 0) then
       reader_color = 1 ; reader_key = global_rank
   else
      reader_color = 0 ; reader_key = global_rank
   endif

call
MPI_Comm_split(global_comm,reader_color,reader_key,readers,ierr)
if (reader_color==1) then h5dreadf(…)
call MPI_Bcast(…,readers)
```

Improved initialization I/O

- **Read aggregation yields significant performance improvements:**



- **Next up: improve write performance**

Where now?

PFLOTRAN is now

- **Highly modular, object-oriented, extensible**
- **Can scale all the way from laptop computers to the largest scale computers in the world**
  - **Not nearly as efficient as we might like…**
  - **…but capable of solving leadership-class simulation problems NOW**

Current focus is now on

- **Unstructured grid implementation.**
  Critical for several problems we want to run on Jaguar.
- **Multiple-continuum support**
- **Making structured adaptive mesh refinement really work**
- **Developing better solvers/preconditioners**
  - **Multilevel (geometric and algebraic approaches)**
  - **Physics-based**