CHAPTER 6

PFLOTRAN: Reactive Flow & Transport Code for Use on Laptops to Leadership-Class Supercomputers

G. E. Hammond[1], P. C. Lichtner[2*], C. Lu[3] and R. T. Mills[4]

[1]Pacific Northwest National Laboratory, Richland, WA, USA; [2]Los Alamos National Laboratory, Los Alamos, NM, USA; [3]Energy and Geoscience Institute, University of Utah, Salt Lake City, UT, USA and [4]Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract: PFLOTRAN, a next-generation reactive flow and transport code for modeling subsurface processes, has been designed from the ground up to run efficiently on machines ranging from leadership-class supercomputers to laptops. Based on an object-oriented design, the code is easily extensible to incorporate additional processes. It can interface seamlessly with Fortran 9X, C and C^{++} codes. Domain decomposition parallelism is employed, with the PETSc parallel framework used to manage parallel solvers, data structures and communication. Features of the code include a modular input file, implementation of high-performance I/O using parallel HDF5, ability to perform multiple realization simulations with multiple processors per realization in a seamless manner, and multiple modes for multiphase flow and multicomponent geochemical transport. Chemical reactions currently implemented in the code include homogeneous aqueous complexing reactions and heterogeneous mineral precipitation/dissolution, ion exchange, surface complexation and a multirate kinetic sorption model. PFLOTRAN has demonstrated petascale performance using 2^{17} processor cores on problems composed of over 2 billion degrees of freedom. The code is currently being applied to simulate uranium transport at the Hanford 300 Area and CO_2 sequestration in deep geologic formations.

Keywords: High performance computing, reactive transport, carbon sequestration, multiple realizations, multiphase flow and transport, Richards equation, domain decomposition.

1. INTRODUCTION

Over the past several decades, subsurface flow and transport models have become vital tools for the U.S. Department of Energy (DOE) in its environmental stewardship mission. These models have been employed to evaluate the impact of fossil energy resources, such as CO_2 sequestration in deep geologic formations, on the environment, and the efficacy of proposed remediation strategies for legacy waste sites.

For years, traditional models—simulating groundwater flow and solute transport, with basic chemical reactions such as aqueous complexing, mineral precipitation/dissolution, sorption to rock/soil surfaces and radioactive decay—have been employed in 1D or 2D systems [1]. Although these simplified groundwater models are still in wide use, advances in subsurface science have enabled the development of more sophisticated models that employ multiple fluid phases and chemical components coupled through a network of biological and geochemical reactions at multiple scales. With this increased complexity, however, comes the need for increased computing power, typically far beyond that of the average desktop computer. This is especially true when applying these models to large-scale three-dimensional problem domains. This paper gives a brief description of PFLOTRAN—a next-generation, highly-scalable code for simulations of reactive flows in geologic media—and discusses some of the challenges encountered and the progress made in scaling PFLOTRAN simulations to the petascale on Cray XT4 and XT5 architectures.

PFLOTRAN is a subsurface multiphase, multicomponent reactive flow and transport code intended for use on a variety of computer architectures ranging from laptops to leadership-class supercomputers (see Section 4). It is founded upon established frameworks for high-performance computing [*i.e.* HDF5 (Hierarchical

[*]**Address correspondence to P.C. Lichtner:** Los Alamos National Laboratory, Los Alamos, NM, USA; Tel: 1-505-667-3420; E-mail: lichtner@lanl.gov

Data Format 5), MPI (Message Passing Interface), PETSc (Parallel Extensible Toolkit for Scientific computing), SAMRAI (Structured Adaptive Mesh Refinement Application Interface)], and supports seamless integration of Fortran 9X, C and C^{++} programming languages. PFLOTRAN is licensed under an open source GNU Lesser General Public License (LGPL).

2. GOVERNING EQUATIONS

The governing equations employed in PFLOTRAN to model subsurface flow depend on the physical and chemical processes simulated. Thus, the code is divided up into several flow modes including multiphase CO_2-H_2O, air-liquid water, Thermal-Hydrologic-Chemical (THC), and Richards' equation for variably saturated porous media. The flow modes are coupled to a multicomponent geochemical transport mode through temperature, pressure, flow velocity, and phase saturation state. Likewise, the geochemical transport mode may alter the flow field through changes in porosity, permeability and tortuosity caused by chemical reactions.

2.1. Multiphase Flow

Local equilibrium is assumed between phases for modeling multiphase systems with PFLOTRAN. The multiphase partial differenttial equations for mass and energy conservation solved by PFLOTRAN have the general form [2]:

$$\frac{\partial}{\partial t}\left(\varphi\sum_{\alpha} s_\alpha \rho_\alpha X_i^\alpha\right)+\nabla\cdot\sum_{\alpha}\left[q_\alpha\rho_\alpha X_i^\alpha-\varphi s_\alpha D_\alpha\rho_\alpha\nabla X_i^\alpha\right]=Q_i \tag{1a}$$

for the *i* th component, and

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

for energy. In these equations α designates a fluid phase (*e.g.* H_2O, supercritical CO_2) at temperature T and pressure P_α with the sums over all fluid phases present in the system; species are designated by the subscript i (*e.g.* w = H_2O, c = CO_2); φ denotes the porosity of the geologic formation; s_α denotes the phase saturation state; X_i^α denotes the mole fraction of species $i\left(\sum_i X_i^\alpha=1\right)$; ρ_α, H_α, U_α refer to the molar density, enthalpy, and internal energy of each fluid phase, respectively; and q_α denotes the Darcy flow rate defined by

$$\boldsymbol{q}_\alpha = -\frac{kk_\alpha}{\mu_\alpha}\boldsymbol{\nabla}\left(P_\alpha - W_\alpha\rho_\alpha g\boldsymbol{z}\right), \tag{2}$$

where k refers to the intrinsic permeability, k_α denotes the relative permeability, μ_α denotes the fluid viscosity, W_α denotes the formula weight, g denotes the acceleration of gravity, and z designates the vertical of the position vector. The source/sink terms, Q_i and Q_e, describe injection and extraction of mass and heat at wells, respectively. The quantities ρ_r, c_r, and κ refer to the density, heat capacity, and thermal conductivity of the porous rock.

Additional constitutive relations are needed to account for capillary pressure, and changes in phase which are not discussed in detail here (see [3]). In PFLOTRAN a variable switching approach is used to account for phase changes enforcing local equilibrium. According to the Gibbs phase rule there are a total of N_C +1 degrees of freedom where N_C denotes the number of independent components. This can be seen by noting that the intensive degrees of freedom are equal to $N_{\text{int}} = N_C - N_P + 2$, where N_P denotes the number of phases. The extensive degrees of freedom equals $N_{\text{ext}} = N_P - 1$. This gives a total number of degrees of freedom $N_{\text{dof}} = N_{\text{int}} + N_{\text{int}} = N_C + 1$, independent of the number of phases N_P in the system.

2.2. Richards' Equation

The governing mass conservation equation for PFLOTRAN's variably-saturated single phase flow mode is given by

$$\frac{\partial}{\partial t}(\varphi s \rho) + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{q}) = Q_w, \tag{3}$$

and

$$\boldsymbol{q} = -\frac{k k_r}{\mu} \boldsymbol{\nabla} (P - \rho g z). \tag{4}$$

Here, φ denotes porosity, s saturation, ρ water density, q Darcy flux, k intrinsic permeability, k_r relative permeability, μ viscosity, P pressure, g gravity, and z the vertical component of the position vector. Supported relative permeability functions for Richards' equation include van Genuchten, Books-Corey and Thomeer-Corey, while the saturation functions include Burdine and Mualem. Water density and viscosity are computed as a function of temperature and pressure through an equation of state for water.

2.3. Geochemical Transport

In PFLOTRAN, the geochemical transport equations may be coupled to the flow equations or run in standalone mode. In coupled mode, the flow equations provide the pressure, temperature, Darcy flow velocity, and saturation as functions of time and position. In standalone mode, these quantities are given constant values. Chemical reactions currently implemented in the code are listed in Table **1** and consist of homogeneous aqueous complexing reactions, and heterogeneous mineral dissolution/precipitation, ion exchange and surface complexation reactions. Thermodynamic data are read from an extensive database for equilibrium constants over a range of temperatures from 0–300$^{\text{o}}$C and fixed pressure at 1 bar for temperatures below 100℃ and along the saturation curve for pure water for higher temperatures, reaction stoichiometry, mineral molar volumes, species valence, and Debye-Hückel parameters. The user may also use a customized database for higher temperatures, although pressure must be fixed in the current implementation. Surface complexation reactions may be treated either as intrinsically fast reactions in local chemical equilibrium or through a kinetic multirate model defined below. Reactions are transformed to canonical form [4] using a basis set of N_c aqueous primary species that may differ from the species used to construct the database.

The governing mass conservation equation for PFLOTRAN's geochemical transport mode for a multiphase system written in terms of a set of independent aqueous primary or basis species has the form.

$$\frac{\partial}{\partial t}\left(\varphi \sum_{\alpha} s_{\alpha} \Psi_j^{\alpha}\right) + \nabla \cdot \sum_{\alpha} \left(q_{\alpha} - \varphi s_{\alpha} D_{\alpha} \nabla\right) \Psi_j^{\alpha} = Q_j - \sum_m v_{jm} I_m - \frac{\partial S_j}{\partial t} \tag{5}$$

where the sums over α are over all fluid phases in the system, and where Ψ_j^{α} denotes the total concentration in the α th fluid phase for primary species $\mathcal{A}_j^{\text{pri}}$ defined by:

$$\Psi_j^{\alpha} = \delta_{l\alpha} C_j^l + \sum_{i=1}^{N_{\text{sec}}} \nu_{ji} \mathcal{C}_i^{\alpha}. \tag{6}$$

Table 1: Chemical reactions implemented in PFLOTRAN written in terms of primary species $\mathcal{A}_j^{\mathrm{pri}}$, secondary species $\mathcal{A}_j^{\mathrm{sec}}$, minerals $\mathcal{M}_m$, gaseous species $\mathcal{A}_l^{\mathrm{g}}$, sorbed species $X_{z_k}\mathcal{A}_k$, surface complexes $\Xi\mathcal{A}_k^{\sigma}$ and empty surface sites $\Xi\mathcal{X}_{\sigma}$, with corresponding stoichiometric coefficients ν_{jl}, ν_{jm}, ν_{jl}^{g}, and ν_k^{σ}. Reaction rates are based on local equilibrium (LEQ) or kinetic rate laws. Primary and secondary aqueous species may be interchanged provided the resulting reactions are linearly independent.

Type	Reaction	Rate
	Homogeneous	
Aqueous	$\sum_j \nu_{ji}\mathcal{A}_j^{\mathrm{pri}} \rightleftharpoons \mathcal{A}_i^{\mathrm{sec}}$	LEQ
	Heterogeneous	
Mineral	$\sum_j \nu_{jm}\mathcal{A}_j^{\mathrm{pri}} \rightleftharpoons \mathcal{M}_m$	Kinetic
Gaseous	$\sum_j \nu_{jl}^{g}\mathcal{A}_j^{\mathrm{pri}} \rightleftharpoons \mathcal{A}_l^{g}$	LEQ
Ion Exchange	$z_k\mathcal{A}_j^{z_j+} + z_j X_{z_k}\mathcal{A}_k \rightleftharpoons z_j\mathcal{A}_k^{z_k+} + z_k X_{z_j}\mathcal{A}_j$	LEQ
Surface Complexation	$\nu_k^{\sigma}\,\Xi\mathcal{X}_{\sigma} + \sum_j \nu_{jk}^{\sigma}\mathcal{A}_j^{\mathrm{pri}} \rightleftharpoons \Xi\mathcal{A}_k^{\sigma}$	LEQ/Kinetic Multirate

The superscript l denotes the aqueous phase and C_j^l represents the concentration of primary species $\mathcal{A}_j^{\mathrm{pri}}$ assumed to be chosen from the set of aqueous species. The secondary species concentration C_i^{α} is computed in terms of the primary species from the mass action relation

$$C_i^{\alpha} = \left(\gamma_i^{\alpha}\right)^{-1} K_i^{\alpha} \prod_{j=1}^{N_c} \left(\gamma_j^l C_j^l\right)^{\nu_{ji}}, \tag{7}$$

with equilibrium constant K_i^{α}, and activity coefficients γ_j^{α}, γ_i^{α}. The activity coefficients are currently computed using the Debye-Hückel equation. Both kinetic and equilibrium sorption are described through the quantity S_j which denotes the concentration of sorbed species in units of mol per bulk volume (see [5] for more details). The quantity Q_j denotes a source/sink term. The coefficient D_{α} represents hydrodynamic dispersion and molecular diffusion as a diagonal tensor. A full dispersion tensor is not currently supported, as the approach presents the potential for oscillatory behavior and negative concentrations that are physically impossible and cause the geochemical algorithms to fail. Although other codes may truncate minimum concentrations to resolve this problem, cutoffs for concentrations are arbitrary and can introduce error themselves. Diffusion is considered to be species-independent in the current formulation, which greatly simplifies the flux term in the transport equations. Mineral reactions are described by the reaction rate I_m for mineral $\mathcal{M}_m$ with the mineral concentration computed from the equation

$$\frac{\partial \varphi_m}{\partial t} = \overline{V}_m I_m, \tag{8}$$

with molar volume $\overline{V}_m$. The reaction rate I_m is based on transition state theory with the form

$$I_m = -k_m a_m \mathcal{P}_m \left(1 - K_m Q_m\right) \zeta_m, \tag{9}$$

using the sign convention of a positive value for precipitation and negative for dissolution, where k_m and K_m denote the kinetic rate constant and equilibrium constant, respectively, both functions of temperature and pressure, and a_m refers to the mineral specific surface area. The ion activity product Q_m is defined by

$$Q_m = \prod_k \left(\gamma_k^l C_k^l\right)^{\nu_{km}}. \tag{10}$$

The quantity $\mathcal{P}_m$ is a prefactor with accounts for the pH dependence and other attenuation factors on the rate. The factor ζ_m takes on the values zero or one to ensure that if a mineral is not present at some particular point in space, it cannot dissolve:

$$\zeta_m = \begin{cases} 1, & \varphi_m > 0 \text{ or } K_m Q_m > 1, \\ 0, & \varphi_m = 0 \text{ and } K_m Q_m \le 1 \end{cases} \tag{11}$$

Porosity and permeability may be altered through chemical reactions and coupled back to the flow equations. Porosity is obtained from the relation

$$\varphi = 1 - \sum_m \varphi_m, \tag{12}$$

which presumes that the total porosity of the porous medium is connected. From this relation permeability may be computed using a phenomenological relation between permeability and porosity, for example, a power law relation of the form

$$k(\varphi) = k_0 \left(\frac{\varphi}{\varphi_0}\right)^n, \tag{13}$$

for some power *n*, with initial porosity and permeability φ_0, k_0, respectively. Likewise, mineral surface area can vary according to a power law relation of the form

$$a_m = a_m^0 \left(\frac{\varphi_m}{\varphi_m^0}\right)^{n_m}, \tag{14}$$

where typically n_m=2/3, and a_m^0 and φ_m^0 denote the initial mineral surface area and volume fraction. This relation only applies to primary minerals with $\varphi_m^0 \neq 0$.

3. PFLOTRAN FRAMEWORK

PFLOTRAN is developed for use with leadership class, high-performance computing architectures. The code is designed from the ground up to efficiently scale on the latest ultrascale supercomputers. It can also run efficiently on desktops and notebook computers and is developed on the latter by most of the development team. The code is written in a modular manner facilitating the incorporation of new computational algorithms and scientific processes. This section describes several concepts or features that are considered to be important for modern code development and key to attaining scalability on the latest leadership-class supercomputers. These key aspects of PFLOTRAN's development include the integration of computational science within an object-oriented coding paradigm, PFLOTRAN's founding upon the robust and sophisticated PETSc library, and the incorporation of scalable I/O through parallel HDF5.

3.1. Object-Oriented Fortran 9X

In order to facilitate code development and modularity in PFLOTRAN, an object-oriented coding paradigm is employed through the judicious use of Fortran9X features. The nature of Fortran9X dictates that the object-oriented paradigm employed within PFLOTRAN differ somewhat from traditional object-oriented languages such as C^{++} and Java in that no attempt is made to explicitly leverage polymorphism or inheritance within the framework, though some use is coincidental. Bea *et al.* [6] explains that Fortran9X polymorphism can be achieved through the use of appropriate interfaces within the code. Although their statement is technically correct, the user must still provide a unique argument list to enable the interface to select the correct subroutine. Within PFLOTRAN, most science subroutines that could exhibit polymorphism pass identical arguments, and thus the interface cannot differentiate between routines. Besides, it can be argued that the user might as well explicitly declare the name of the subroutine called for improved clarity and readability within the code. That being said, interfaces are used widely throughout the code, but not with the explicit intent of enabling polymorphism.

It is our view that the most important aspects of the object-oriented paradigm are data encapsulation and modularity within the code. Since data abstraction, polymorphism and inheritance are not necessarily naturally implemented within Fortran9X, it can be argued that one should use a truly object-oriented language such as C^{++} or Java for those purposes. As of late, Fortran 2003/2008 has increased the number of object-oriented features within Fortran (*e.g.* type extension and inheritance, polymerphism, dynamic type allocation and type-bound procedures) [7]. However, in PFLOTRAN extensive utilization of Fortran 2003/2008 paradigms has been avoided due to inconsistent implementation across compiler platforms, the use of which would greatly limit the platform independence of the code.

As would be expected with any object-oriented code, the numerical algorithms, data structures, and scientific processes within PFLOTRAN all revolve around *classes* and their instantiations, *objects*. A PFLOTRAN class is defined in Fortran through derived data types composed of standard Fortran data structures (*i.e.* dynamically allocated arrays of characters, integers, reals) and pointers to other lower-level derived types/classes; it is not an actual Fortran 2003/2008 CLASS statement described by [7]. Within the code a hierarchy of classes exist ranging from lower-level ones such as auxiliary data structures that contain raw data such as pressures, temperatures, or concentrations, to mid-level ones such as the realization class, which encompasses all low-level data sets and objects, or the timestepper class that controls the procedural workflow to the high-level simulation class that encompasses all other objects in a PFLOTRAN simulation.

Although a comprehensive description of PFLOTRAN's object-oriented paradigm is beyond the scope of this paper, a brief descryption of several mid-to high-level classes is provided. For instance, PFLOTRAN's highest level object is the simulation object (or multi-realization simulation object if run in stochastic mode). The simulation object contains pointers to two types of objects, the realization and the timesteppers for flow and transport, as shown in Fig. **1**.

```
type :: simulation_type
  type(realization_type), pointer :: realization
  type(timestepper_type), pointer :: flow_stepper
  type(timestepper_type), pointer :: tran_stepper
end type simulation_type
```

Figure 1: Fortran9X data type for PFLOTRAN simulation class.

The realization object contains a hierarchy of data structures and objects that define the problem statement and scientific algorithms being employed to solve the mathematical/scientific equations that govern the problem statement. The timestepper encompasses the nonlinear and linear solvers utilized to solve the systems of PDEs either for steady-state conditions or for an increment in time (*i.e.* a time step). The timestepper essentially utilizes the realization object to first determine which scientific processes are being simulated and then populate the system of equations being solved through residual and Jacobian function evaluations based on realization parameters/properties (*e.g.* rock/soil permeabilities, reaction rate constants, time step size, *etc.*) and processes (Richards' equation, saturation functions, geochemical reaction equations, *etc.*).

Fig. **2** illustrates a lower-level class for defining an aqueous species using linked lists. Within the aqueous species object, primary and secondary aqueous species are assigned an id, name, Debye-Hückel ion size parameter (a0), molecular weight (molar_weight), valence (Z) and flag (print_me) for I/O purposes. For the case where the species is a secondary aqueous complex, an equilibrium reaction object (*i.e.* equilibrium_rxn_type) is allocated, which contains parameters for calculating the complex concentration (*i.e.* primary species ids, stoichiometries, equilibrium constant). As PFLOTRAN reads each species from the prescribed input file, an aqueous species object is created and appended to a linked list of species. Parameters from these species lists are later parsed into compact arrays for efficient lookup during PFLOTRAN execution.

```
type, public :: aq_species_type
  PetscInt :: id
  character(len=MAXWORDLENGTH) :: name
  PetscReal :: a0
  PetscReal :: molar_weight
  PetscReal :: Z
  PetscTruth :: print_me
  type(equilibrium_rxn_type), pointer :: eqrxn
  type(aq_species_type), pointer :: next
end type aq_species_type
```

Figure 2: Fortran9X data type for PFLOTRAN aqueous species class.

Object-oriented Fortran9X syntax can appear quite different in comparison to traditional Fortran programming. Fig. **3** illustrates two *do* loops for setting total component concentrations to a value using object-oriented Fortran9X with PFLOTRAN objects and traditional Fortran77. Notice that in the case of object-oriented Fortran9X, the total component concentration array total, of size reaction%ncomp, is embedded within each element of the rt_aux_vars array, which is an array of reactive transport auxiliary objects of size grid%nlmax. In this case, the grid object owns the size of the local on-processor portion of the computational grid (*i.e.* nlmax) and the reaction object owns the number of primary chemical degrees of freedom (*i.e.* ncomp). All of these objects (grid, reaction, rt_aux_vars) are compartmentalized as descendants of the high-level realization object. The arrays and parameters are accessed by referencing descendants of the realization object instead of using a common statement. This object-oriented approach improves the modularity of the code and better ensures the correct data locality (*i.e.* there is no need for common statements throughout the code).

```
grid => realization%patch%grid                 common/array/a(nlmax,ncomp)
reaction => realization%reaction
rt_aux_vars => &
realization%patch%aux%Global%aux_vars

                                               do n = 1, nlmax
do n = 1, grid%nlmax                              do j = 1, ncomp
   do j = 1, reaction%ncomp                          a(n,j) =..
      rt_aux_vars(n)%total(j) =..                 enddo
   enddo                                       enddo
enddo
                                                     Traditional Fortran77
               OO Fortran9X
```

Figure 3: Comparison of object-oriented Fortran9x and traditional Fortran loops.

Constructor and destructor routines exist for all classes employed within PFLOTRAN. Objects are created as needed and initialized to default values in their respective constructor routines. The first object to be created is the simulation object (*i.e.* SimulationCreate()), after which all underlying objects are created. In certain cases, objects can be replicated in the construction process by passing the original object to the constructor interface. The use of constructors guarantees that the initialization of data types within PFLOTRAN objects is uniform throughout the code. At the end of a simulation, the destructor routine is called for the highest simulation object (*i.e.* SimulationDestroy()). This routine in turn calls the destructors for its realization and timestepper objects, each of which do the same for underlying objects and data structures. In the case of standard Fortran data structures (*e.g.* arrays of integers, reals), deallocate is called instead of the destructor. This process propagates to the lowest-level objects until all objects are destroyed and memory is freed.

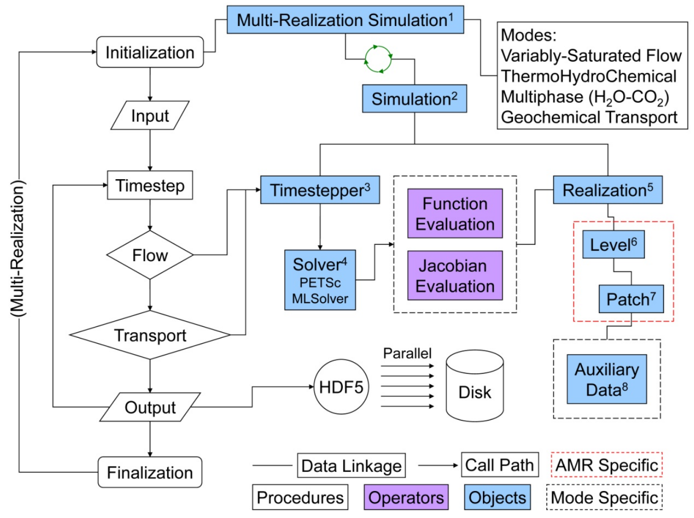


Figure 4: Schematic of PFLOTRAN workflow and data dependence illustrating the use of procedures, operators, and objects within the code (see text for explanation of numbers).

Fig. **4** illustrates PFLOTRAN's workflow and data dependence. Explanation of numbers appearing in the figure are:

1. Multi-Realization Simulation object: Highest level data structure providing all information for running simulations composed of multiple realizations.
2. Simulation object: Data structure providing all information for running a single simulation.
3. Timestepper object: Pointer to Newton-Krylov solver and tolerances associated with time stepping.
4. Solver object: Pointer to nonlinear Newton and linear Krylov solvers (PETSc) along with associated convergence criteria.
5. Realization object: Pointer to all discretization and field variables associated with a single realization of a simulation.
6. Level object: Pointer to discretization and field variables associated with a single level of grid refinement within a realization.
7. Patch object: Pointer to discretization and field variables associated with a subset of grid cells within a level.
8. Auxiliary Data object: Pointer to auxiliary data within a realization/patch.

A comprehensive list of PFLOTRAN objects and their linkage to the processes in the workflow would be difficult to include in a single schematic, and thus, only higher-level objects/processes are shown. To the left is the standard flowchart for flow and transport simulators where the model initializes, reads the input deck, and steps through time computing flow and transport while writing results at select times. In addition to this traditional sequence, a multi-realization loop has been added to signify the simulation of more than one realization. To the right, higher-level data objects are illustrated along with their connectivity within the code's data space. The flowchart illustrates that the flow and transport algorithms call a solver within the timestepper. The solver then calls for a (residual) function evaluation and the calculation of the associated Jacobian matrix. Both solver function calls require auxiliary data provided by the realization object to complete the task (*e.g.* permeability, reaction rates, *etc.*), and thus, the realization object possessing this data is passed to the routines.

For Adaptive Mesh Refinement (AMR), auxiliary data is associated with levels of grid refinement and patches of cells within each level; thus, the linkage through the level and patch objects. The hierarchy of objects is designed to minimize the amount of data explicitly passed to procedures (without resorting to global variables) and maximize compartmentalization of the parameters and processes. In doing so, parameters and processes may be altered and customized with minimal impact to the remainder of the code (*i.e.* procedure argument lists), thus greatly facilitating code development and maintenance. The object-oriented paradigm is also vital in the implementation of the AMR framework within PFLOTRAN as the physical domain must be divided among grid levels and patches, which are treated as lists of objects and are easily accommodated within the PFLOTRAN framework.

Critics of object-oriented programming and the extensive use of pointers within Fortran codes may argue that computational performance degrades due to limitations in the compilers' ability to optimize the code. In the case of PFLOTRAN, the bulk of computation time is spent in isolated "kernels", and it has been demonstrated that the object-oriented design of the code has minimal, if any, negative impact on the code's execution time. Considering that a large fraction of the code's execution time is spent within the PETSc nonlinear solvers for larger parallel runs (*e.g.* 95+% for large-scale flow simulations and 60+% for large-scale reactive transport simulations) and that the residual function evaluations scale exceptionally well, one could argue that the ease of programming and exceptional modularity of the code far outweigh any existing degradation in performance.

3.2. Parallel Implementation

PFLOTRAN is founded upon parallel data structures and solvers provided by PETSc [8]. The code employs domain decomposition through PETSc DA (Distributed Array) or DM objects (The DM object is a generalization of the DA object for managing an abstract grid object.) to partition a physical gridded domain across processor cores, depending on whether structured or unstructured grids are employed (Note: Unstructured grids are currently in the development phase, yet to be completed). With this approach, each processor core possesses locally the data necessary to calculate its portion of the global problem being solved, regardless of the algorithm employed. With the decomposition in place, PETSc provides the necessary index sets or mappings for passing data between processors during the course of the simulation (*e.g.* updating ghost cells, checkpointing vectors, *etc.*). In fact, PETSc actually performs the necessary vector gather/scatters when requested, masking the details of communication.

PFLOTRAN accesses PETSc linear and nonlinear solvers through the SNES (Scalable Nonlinear Equation Solvers) component which provides an interface to various methods (primarily Newton-based) for solving systems of nonlinear equations. Through the SNES and the underlying linear equation solver (KSP) and preconditioner (PC) components, the user may specify which solver algorithms and associated parameters (*e.g.* tolerances, maximum iteration counts, *etc.*) are to be employed to solve the system PDEs for flow and transport. From the "physics" or "science" side, PFLOTRAN provides to SNES two types of function evaluations for solving the Newton-Raphson method: one that computes the residual vector on the right hand side and another that calculates the coefficients for the Jacobian matrix. For each nonlinear solver iteration, SNES calls the function evaluation routines on each processor core to compute the local entries in the residual vector and coefficients for the local rows of the Jacobian. The SNES solver then employs the selected parallel KSP solver to solve the global, linear system of equations and update the nonlinear solution on each processor. This iterative process continues until the solution converges as dictated by the tolerances specified. From the outside, the SNES solve appears no different in parallel than in serial. This is the beauty of parallel domain decomposition within the PETSc framework: the framework masks the intricate details of parallel computing (data decomposition, message passing, *etc.*) at the upper-level user interfaces, exposing them only if the user so desires.

In order to obtain scalable solutions on massively-parallel leadership class machines, the developer must take care to prevent unnecessary bottlenecks from impeding efficient parallel performance. The developer must understand the data requirements and communication patterns of the parallel algorithms employed, including underlying numerical libraries, and resolve parallel inefficiencies. The better the developer understands the parallel implementation of the entire code, the easier it is to resolve parallel bottlenecks. For this reason, the use of poorly understood, *black bo*x algorithms within parallel codes increases the likelihood of inefficient scalability. If the user introduces algorithms that continually require off-processor data, a potential exists for excessive communication, which may limit parallel scalability. Often, such bottle-necks are not so evident on tens to hundreds of processor cores. However, as the number of cores increases, the inefficiencies become more evident.

Take for instance, parallel I/O through processor zero with a round-robin approach where data is passed to processor zero on a processor by processor basis (*i.e.* a global vector is not required). The model may scale well up to a few hundred processor cores even through such an I/O algorithm is a serial bottleneck; the bottleneck has just not yet become obvious. However, as the number of processor cores grows into the thousands, I/O through processor zero is a known bottleneck, especially for ASCII I/O. At that point, the developer must improvise and employ a more novel approach to I/O such as parallel HDF5, as is employed in PFLOTRAN. At tens to hundreds of thousands of processor cores, the algorithms that limit PFLOTRAN's scalability are in the parallel solvers (*i.e.* global reductions or dot products) and I/O.

3.3. Multiple Realization Simulations

Perhaps one of the most unique and innovative features of PFLOTRAN is its ability to launch multiple simulations of different realizations simultaneously, each realization being executed across multiple processor cores. Although the embarrassingly parallel execution of multiple simultaneous realizations is common these days, the discharge of each in parallel through domain decomposition within a processor subcommunicator group is a novel feat within subsurface simulation. This ability will revolutionize Monte-Carlo style analyses used to better quantify uncertainty in the subsurface.

For the user, the implementation of a stochastic multi-realization simulation is quite straightforward. For example, suppose multiple realizations of permeability are to be employed in a Monte-Carlo fashion. Assuming the correlated random fields have been generated beforehand, a simple script is utilized (*e.g.* a Python script using h5py and numpy libraries) to load these datasets into an HDF5 formatted file under a dataset with a name describing the dataset and the realization *id* (*e.g.* Permeability1 – permeability for realization #1). At the prompt or within the job script, the user enters command line arguments that specify that the simulation be run in stochastic mode with a specified number of realizations and processor groups (Note: the number of processor groups must be less than or equal to the number of processor cores). An example of the command line arguments follows:

```
mpirun -np 10000 pflotran
   -stochastic
   -num_realizations 1000
   -num_groups 100
```

Upon execution, the realizations and parallel job's processor cores are divided as evenly as possible among the processor groups. In this case, 1000 realizations will be run on 100 processor groups using 10,000 processor cores.

Each processor group will utilize 100 processor cores (np⁄num groups) and run 10 realizations apiece (num realizations/num groups), one after another. Thus, only 100 realizations may be executed simultaneously as each processor group may only simulate a single realization on 100 processor cores at a time. Each processor group continues to run realizations until its allocation of 10 has completed.

An alternative approach would be a masterslave paradigm where the root processor core assigns realizations to sub-communicator groups on a one by one basis. This approach would prove beneficial should significant load imbalance exist between sub-communicator groups. Either way, the implementation of the algorithm is straightforward and embarrassingly parallel.

Output for the stochastic simulation is written to files labeled by the realization id and/or processor group id. The user then employs scripts or codes to postprocess the results, computing statistical averages, sampling data, *etc.* To date, this approach has been successfully demonstrated with PFLOTRAN on stochastic simulations composed of hundreds of thousands of realizations and utilizing thousands of processor cores.

3.4. Parallel HDF5 I/O

Parallel I/O plays an important role in enabling, or perhaps better stated, not degrading parallel scalability within a code. PFLOTRAN employs scalable I/O through the parallel HDF5 [9] which leverages collective MPI-IO operations across high-performance file systems. The HDF5 data model provides a sophisticated set of data objects and associated metadata for archiving virtually any combination of data within a single binary file. The HDF5 API provides the programmer with flexible interfaces for reading/writing data from/to a file with relative ease. Through this interface, one specifies the attributes, properties, and data types associated with the data set, all of which can be compartmentalized within a hierarchy of groups within the file. In fact, data sets within an HDF5 file are arranged in a manner similar to the file and directory structure of a standard file system, but within a single file. That is not to say that the file can be navigated in the same manner.

HDF5 files are platform-independent, and data can be accessed through high-level API interfaces written in C, C^{++}, Fortran 90, Java, and Python regardless of the native datatypes on a particular machine (*e.g.* 32-bit vs 64-bit, bigendian vs little-endian, C-specific vs Fortran-specific). For instance, one may write an HDF5 file using Python on a 32-bit, Windows-based laptop computer and read it in parallel across 100,000 processor cores using Fortran90 on a 64-bit Linux-based supercomputer. The library also provides linkage to external libraries such as zlib and szip for compressed storage. All data are stored in an optimal binary format for rapid access.

Critical to the scalability of its parallel I/O, HDF5 employs MPI-IO to enable super-computers to write data either independently or collectively to the same file across hundreds of thousands of processor cores. Our profiling has demonstrated that parallel HDF5's collective writes (all processor cores within an MPI communicator writing in unison) are much more scalable than independent writes (processor cores writing independently), as one would expect. For data sets associated with structured grids, PFLOTRAN scalable parallel I/O relies heavily on the HDF5 hyperslab data layout (*i.e.* up to 3D in memory and file space) to efficiently write data in parallel.

All input data files are read as one-dimensional data in file space and mapped accordingly to memory space, using hash tables to sample for local on-processor values. This approach has reduced initialization time by orders of magnitude for highly parameterized and heterogeneous problem domains. The one-dimensional file space enables PFLOTRAN to read a wider variety of grid structures, as opposed to solely structured grids. Another enhancement to the HDF5 reading algorithms within PFLOTRAN is the use of subsets of processors through MPI sub-communicators to read data and distribute it to the remainder of the processor cores (*e.g.* within the global communicator): this results in more optimal communication patterns and reduces contention at file metadata servers.

4. PARALLEL PERFORMANCE

We have put significant focus on achieving parallel scalability on leadership-class high-performance computing architectures as part of a SciDAC-2 groundwater project (http://ees.lanl.gov/pflotran/). Designed from the ground up for parallel efficiency, PFLOTRAN delivers to the end user the ability to simulate real-world problems at scales limited by only the scalability of linear/nonlinear system solvers. The code runs on machines ranging from the laptop to the largest massively-parallel computer architectures. In fact, core development of the code generally takes place on lap-tops and desktops (Mac, Linux, Windows) with testing on smaller 1D, 2D or 3D problems sized to fit within the machine's memory limits. Testing at scale is carried out on a supercomputer. It should be noted that on any specified architecture, a single PFLOTRAN executable can be run utilizing any of the code's flow or geochemical transport modes in both serial and parallel without modifying the executable (assuming that the job fits into the available memory). In other words, changes to the number of processor cores, parallel decomposition, simulated physical and chemical processes, problem size, *etc.* do not require recompiling the code, since PFLOTRAN utilizes dynamic memory and processor allocation. Production jobs using PFLOTRAN can be run in serial or parallel (provided MPI is installed) on desktop or laptop computers for smaller problem sizes (*e.g.* 1D or small 2D or 3D problems), whereas to simulate large 3D field-scale problems generally much larger supercomputing resources are required.

Perhaps the most significant factor in achieving PFLOTRAN's scalability is the consistent use of PETSc data structures and solvers. Because PFLOTRAN uses PETSc's parallel data structures and associated routines to manage parallel domain decomposition, details such as MPI communication patterns are handled by highly efficient algorithms in PETSc. Furthermore, the large number of solver and preconditioner algorithms made available within PETSc and through PETSc interfacing allows the user to choose the most suitable and scalable solver algorithms for a particular application. For instance, for steady-state flow in a saturated, heterogeneous porous medium, one might choose PETSc's stabilized biconjugate gradient algorithm (Bi-CGStab) coupled with a multilevel preconditioner such as Hypre's PFMG solver [10]. Whereas, for multicomponent biogeochemical transport, PETSc's block Jacobi algorithm may adequately precondition Bi-CGStab for most problem scenarios. Thus through use of PETSc, PFLOTRAN has access to a wide range of algorithms for testing and comparison purposes. In addition, PETSc provides shells for research and development of user-defined solvers/preconditioners (*e.g.* [11]).

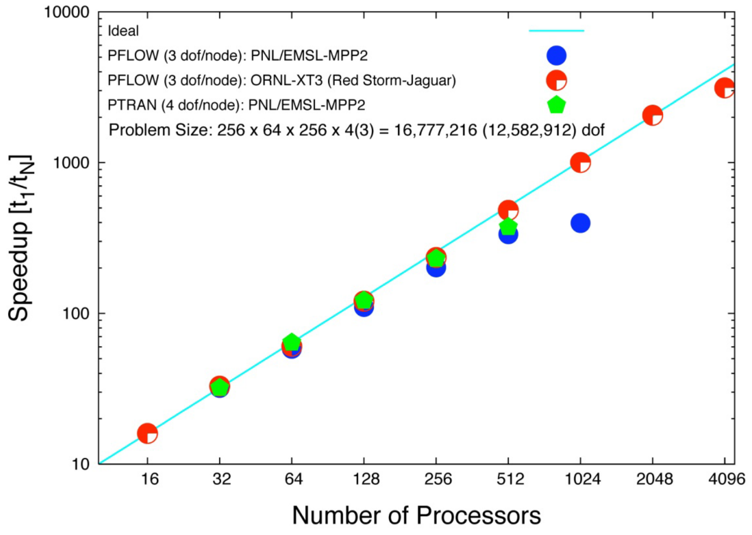


Figure 5: Performance of PFLOTRAN (PFLOW refers to flow and PTRAN to reactive transport) running a single phase thermohydrologic benchmark problem on a 256×64×256 grid with three and four degrees of freedom per node, respectively (approximately 12.6 and 16.8 million degrees of freedom total), on the now-defunct Cray XT3 incarnation of Jaguar at ORNL and the Itanium2-based MPP2 cluster at PNNL.

Prior to being funded by SciDAC, PFLOTRAN demonstrated efficient scalability as shown in Fig. **5** for a moderately-sized, single-phase, thermo-hydrologic benchmark problem composed of 12 to 16 million degrees of freedom (dofs) executed on 16 to 4096 processors on Oak Ridge National Laboratory's (ORNL) Cray XT3 supercomputer. Since SciDAC-funded development, PFLOTRAN has been run on problems composed of up to two-billion degrees of freedom and utilizing up to 131,072 (2^{17}) processor cores on ORNL's Jaguar Cray XT5, the world's fastest open science supercomputer. These large-scale problems are based on real-world variably-saturated flow and geochemical transport modeling of uranium at the Hanford 300 Area in Washington State (see Section 5.1). At this site the migration of a Cold-War era uranium plume is being modeled to better quantify the mass flux of uranium leaching into the neighboring Columbia River, a quantity crucial to environmental decision making and policy at the site.

Fig. **6** demonstrates the strong scaling performance of PFLOTRAN while simulating variably-saturated flow at the Hanford 300 Area using Richards' equation on a problem composed of 270 million degrees of freedom on up to 27,580 processor cores on ORNL's Jaguar XT4 supercomputer. The scenario was run for 50 time steps with I/O turned off. Although good, the scalability is not perfect as the performance deviates from ideal as the processor core count grows. There are several factors contributing to this behavior. First, at 27,580 cores, a 10% increase in the number of linear Bi-CGStab solver iterations was observed relative to the number in the 1024 core run. This increase in linear solver iterations is attributable to an expected and well-understood loss of effectiveness in single-level domain decomposition preconditioners such as block Jacobi. Block-Jacobi preconditioning approximates the inversion of the local on-processor portion of the Jacobian matrix within the linear solver (in this case, through ILU[0] factorization). As the number of cores grows, each processor core owns a smaller portion of the global matrix, resulting in decreased coupling of matrix coefficients (fewer rows and columns of coefficients are coupled), and thus a less-accurate approximate inverse of the matrix.

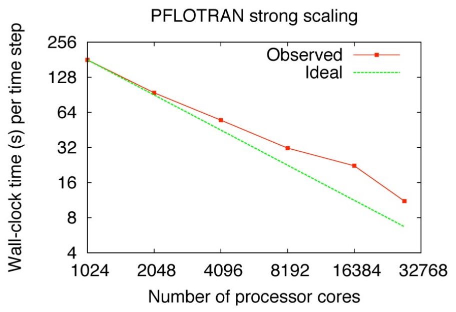


Figure 6: Strong scaling performance on the Cray XT4 partition of ORNL's Jaguar for a variably-saturated flow problem at the Hanford 300 Area with 270 million degrees of freedom (no I/O).

A perhaps more important factor in the departure from linear scaling is the large growth in the relative cost of vector norms and inner products performed within the linear solver as more cores are employed to solve the problem. The cost of these operations is dominated by the cost of global reduction operations, which increases substantially as the number of cores approaches extreme scales. For instance, the time spent in *MPI Allreduce*() calls grew from 10% of the total wall clock time on 1024 cores to up to 56% on 16,384 cores. (We note that we have since worked with the PETSc development team to implement a restructured version of Bi-CGStab that requires only one *MPI Allreduce*() operation per iteration.) Finally, with the current structured-grid formulation of the Hanford 300 Area problem, inactive grid cells exist on the eastern edge of the problem domain where grid cells lie above the river bank and river bottom (see Section 5.1). As the number of processor cores grows, a load imbalance develops, the impact of which is not necessarily easy to quantify when superimposed on the inefficiencies in the preconditioner and increased communication costs.

Aside from the communication issue, which is likely hardware bound, there are several ongoing efforts to further improve PFLOTRAN performance, such as the development of novel solver and preconditioner algorithms to improve both strong (fixed problem size) and weak (fixed problem size per processor core) scalability of the code. Preliminary studies have demonstrated that for steady-state problems, where the Jacobian is less diagonally dominant, multi-level solvers such as multigrid improve weak scalability with much success [12]. With regard to the inactive grid cells, the addition of unstructured gridding is currently underway and will enable a more optimal domain decomposition of the structured grid outside the PETSc Distributed Array (DA) through mesh partitioning software such as ParMETIS, an MPI-based parallel library that implements a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices. The PETSc DA currently does not handle the removal of inactive grid cells, which requires an unstructured grid formulation. However, parallel efficiency is not expected to be nearly as good with unstructured grids as that obtained from structured grids due to such issues as the difficulties in generating optimal mesh partitioning (load balancing), and the lack of geometric multigrid solvers/preconditioners.

Fig. **7** further illustrates the strong scalability of PFLOTRAN for the simulation of uranium transport (see Section 5.1) depicted in Fig. **8** which utilizes up to 65,536 cores on ORNL's Cray XT5 petaflop incarnation of Jaguar. Here, 15 chemical components depicting a subset of geochemical components coupled to uranium mobility through changes in pH, carbonate and calcium concentrations as groundwater and river water mix, are transported within a grid composed of 136 million cells or spatial degrees of freedom. Thus, for flow the nonlinear system is composed of 136 million dofs while geochemical transport consists of 2 billion dofs. Not surprisingly, these performance results reflect relatively poor scalability for flow at larger processor core counts. This is expected given the increased size of the utilized interconnect (increasing cost of global communications) and the small number of flow degrees of freedom per processor core at near 2000—as a general rule of thumb a minimum of 10,000 dof per core is needed to obtain good scaling performance—which results in a very poor ratio of computation to communication. On the other hand for transport, the number of unknowns per core remains above 30,000 using 64,536 cores, and thus it scales rather well considering that only a conventional PETSc solver and preconditioner are being employed (*i.e.* Bi-CGStab with block-Jacobi ILU[0]). For situations such as this where the size of the flow problem is much smaller than the transport problem, it may make sense to add the ability for PFLOTRAN to solve the flow problem redundantly on disjoint groups of MPI processes, much as parallel multigrid solvers do for coarse-grid problems. This may prove unnecessary, however, as in many cases the cost of the transport solve dominates.

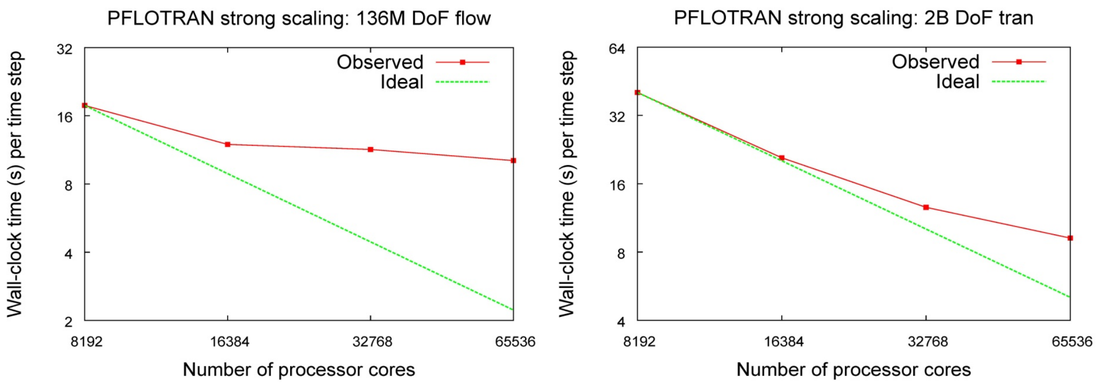


Figure 7: Strong scalability of flow (left) and geochemical transport (right) portions of PFLOTRAN for the Hanford 300 Area 136M dof and 2B dof problems, respectively.

5. APPLICATIONS

In this section two applications of PFLOTRAN are described briefly: (i) uranium migration at the Hanford 300 Area, and (ii) CO_2 sequestration in a deep geologic formation. These examples illustrate, respectively, the use of complex time-dependent boundary conditions and multi-component chemistry coupled to variably saturated flow, and two-phase simulation of H_2O and supercritical CO_2.

5.1. Hanford 300 Area

The Hanford 300 Area is located along the Columbia River in southeastern Washington State. The site was part of plutonium production beginning in 1943. Significant quantities of uranium, copper and other contaminants were disposed at the site in liquid waste streams. Waste was disposed of in two ponds which lie at a distance of approximately 100 m west of the river referred to as the North (NPP) and South (SPP) Processing Ponds. In addition, waste was also placed in nearby trenches. Today a uranium plume persists at the site with concentrations exceeding the EPA 30 μg/L maximum contaminant level [13] despite excavation of contaminated sediments. Particularly perplexing is the persistence of the uranium plume and the factors controlling uranium mobility. Simulations using a K_d model predicted that uranium would be removed by ambient groundwater flow within a decade. Fifteen years later the uranium plume remains where it was with no apparent measurable degradation [14]. Presumably, this is a consequence of slow leaching of uranium from millimeter or smaller scales which causes the uranium source to persist over long time spans.

In a preliminary effort, PFLOTRAN has been applied to model the migration of uranium at the Hanford 300 Area taking into account multicomponent chemistry with a realistic description of uranium sorption through surface complexation [5, 15]. This work differs from other attempts to model the site (*e.g.* [16]) in that the evolution of the uranium plume is divided into three distinct phases. These correspond to: (I) the disposal of uranium and other contaminants and their subsequent migration into the Columbia River, (II) present-day conditions in which non-labile forms of uranium are released at slow rates characterized by diffusion-limited mass transfer from various source regions providing an approximate steady release of uranium into the river, and finally (III) the period in which all non-labile forms of uranium have been removed and multirate sorption governs the behavior of the uranium plume [5]. It is extremely difficult, if not impossible, to model Phase I because of the lack of historical data for the waste stream. Therefore, in Phase II it is assumed that uranium is present in both labile and non-labile forms with sorbed uranium extending from the source regions to the river.

The Hanford 300 Area model consists of a three-dimensional domain measuring 900×1300×20 meters (*x*, *y*, *z*) in size with orientation aligned with the Columbia River at 14° west of north. The base of the model domain lies at 90 meters elevation above sea level. The computational grid is composed of 1,872,000 grid cells with 5-meter horizontal and 0.5-meter vertical grid spacing. Aquifer material properties (*e.g.* permeability, porosity, *etc.*) are assigned to grid cells based on hydrostratigraphic data available at the site [17]. The predominant hydrologic unit at the Hanford 300 Area is the highly-permeable Hanford Unit, which is underlain by several less transmissive Ringold Units. Hydraulic conductivities in the Hanford Unit are on the order of thousands of meters per day, while those of the Ringold Units are hundreds of meters or less *per* day.

Model geochemistry consists of 15 primary aqueous species (H^+, Ca^{2+}, Cu^{2+}, Mg^{2+}, UO_2^{2+} K^+, Na^+, HCO_3^-, Cl^-, F^-, HPO_4^{2-}, NO_3^-, SO_4^{2-} and 2 tracers), 88 secondary aqueous complexes, 2 kinetically-formulated minerals (Calcite and Metatorbernite), 2 surface complexes ($>SOUO_2OH$ and $>SOHOU_2CO_3$) and 1 surface site (>SOH). Important geochemical reactions for equilibrium and multirate surface complexation and mineral dissolution include:

$$>SOH + UO_2^{2+} - 2H^+ + H_2O \rightleftharpoons >SOUO_2OH, \quad \textbf{(15a)}$$

$$>SOH + UO_2^{2+} -2H^+ + H_2O + CO_2 \rightleftharpoons >SOHUO_2CO_3, \quad \textbf{(15b)}$$

$$2\, UO_2^{2+} + Cu^{2+} + 2\, HPO_4^{2-} + 8H_2O - 2H^+ \rightleftharpoons \text{Metatorbernite (s)}, \quad \textbf{(15c)}$$

$$Ca^{2+} + CO_2 - 2H^+ + H_2O \rightleftharpoons \text{Calcite (s)}. \quad \textbf{(15d)}$$

Mobility of uranium is also affected by aqueous complexation reactions with the two dominant species $CaUO_2\left(CO_3\right)_3^{2-}$ and $Ca_2UO_2\left(CO_3\right)_{3(aq)}$. It should be noted that Metatorbernite serves as surrogate source of non-labile U(VI), and although identified at the site in small quantities, the precise form of non-labile U(VI) is still unknown. Overall, 28,080,000 geochemical degrees of freedom are represented in the model.

The tight coupling of Hanford 300 Area hydrology to rapid fluctuations in the Columbia River stage necessitates the use of transient boundary conditions. For variably saturated flow, transient hydrostatic and seepage face/conductance boundary conditions are assigned to the western inland and river boundaries, respectively, with time-varying datums and gradients on a 1-hour time interval. The seepage face boundary condition calculates the boundary flux for river bank cells above the water surface as a function of internal cell and external atmospheric pressures. The conductance boundary condition replicates the lower permeability of the river hyporheic zone sediments for submerged cells. A constant specified surface recharge is assigned to the top of the model domain, while the north and south boundaries are no flow.

For geochemical transport, chemical compositions of background groundwater and river water are obtained from Hanford 300 Area IFRC and USGS data, respectively. The background groundwater concentrations are assigned to the inland boundary and river chemistry to the river seepage face/conductance boundary. Note that a slightly higher pH and lower carbonate concentration in the river relative to the groundwater significantly impact the sorption of U(VI) to Hanford Unit media.

Initial geochemical conditions are set by running a non-sorbing simulation of U(VI) transport based on the boundary conditions discussed above for several years to generate an initial U(VI) plume that reaches to the river. An intermediate initial condition is derived from a snapshot of the flow and transport solution at that time. Sorbed phase U(VI) concentrations are then equilibrated with the aqueous phase to generate the final initial condition.

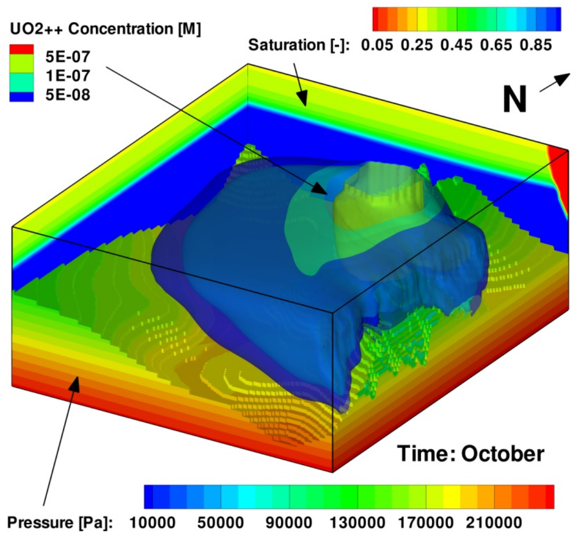

UO2++ Concentration [M]
5E-07
1E-07
5E-08
Saturation [-]: 0.05 0.25 0.45 0.65 0.85
N
Time: October
Pressure [Pa]: 10000 50000 90000 130000 170000 210000

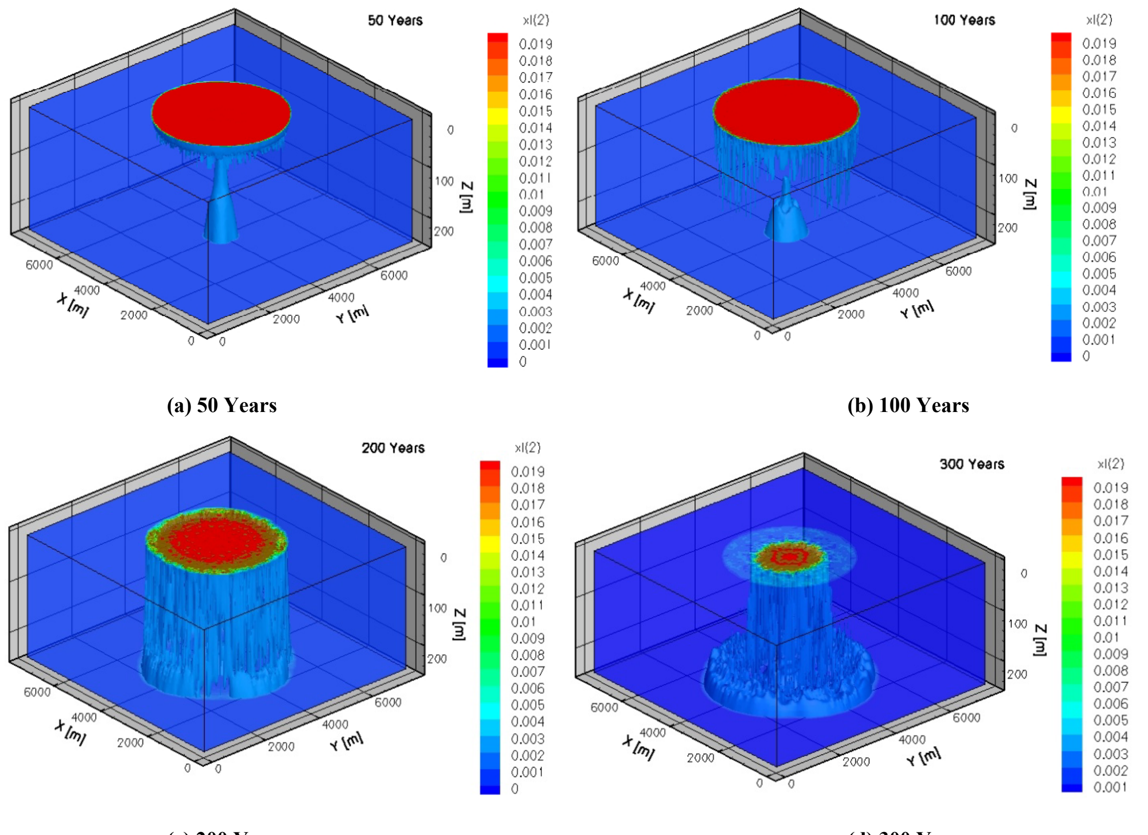
Figure 8: Isosurfaces of total aqueous U(VI) concentration. The Hanford Unit is hidden to better illustrate the U(VI) plume that resides above the Ringold Units. Piezometric pressure is draped on the top of the Ringold Units while water saturation is contoured on the posterior side surfaces of the domain.

Central to this work is the use of high performance computing carried out on the world's fastest open science computer, Jaguar, the Cray XT4/5 at ORNL. Use of HPC made possible the ability to capture the rapidly fluctuating Columbia River stage and multi-component U(VI) chemistry at a sufficiently fine 3D grid resolution. Simulations of the approximately 1.8 million flow and 28 million geochemical transport degrees of freedom were run on over 4k processor cores—runs that would have required several years of computation time on a conventional single processor work-station. Fig. **8** illustrates simulated concentration isosurfaces of a U(VI) plume at the Hanford 300 Area during the October time frame when the river stage is relatively low and groundwater flow is primarily toward the river. Several conceptual models based on this real-world application of PFLOTRAN to radionuclide migration have served as test problems for assessing the parallel performance of PFLOTRAN at the extreme scale. More details can be found in [5].

5.2. Supercritical CO_2 Sequestration

An example of modeling sequestration of supercritical CO_2 using PFLOTRAN is discussed in this section. The code is based on a variable switching approach to account for phase changes between CO_2 and H_2O. The Span and Wagner [18] equation of state is used to compute the density of supercritical CO_2 and correlations for the solubility of CO_2 in brine are taken from [19]. The density of the brine-CO_2 mixture is taken from [20]. To illustrate the code a 3D simulation was carried out for a sandstone host rock containing calcite cement [3].

An isotropic permeability of 2×10^{-12} m^2 is used in the simulation with a porosity of 15%. The nominal temperature and pressure is 50°C and 200 bars. The computational domain is 250 m thick and 7×7 km in lateral extent. CO_2 is injected at a depth of 50 m below the top of the domain. No flow boundary conditions are imposed at the top, bottom, front, and back of the domain with constant pressure at the left and right sides. An injection rate of 1 Mt/y for 20 years was used in the simulations. This corresponds to roughly 75% of the CO_2 produced by a 1000 MW gas-fired power plant in 20 years. Calculations were carried out with 256 processor cores for $160 \times 160 \times 25 \times 3$=1, 920, 000 degrees of freedom. A grid spacing of 47.35 m in the *x*-and *y*-directions and 10 m in the *z*-direction is used. The strong fingering observed in the figures is



(a) 50 Years **(b) 100 Years**

(c) 200 Years **(d) 300 Years**

Figure 9: Mole fraction of dissolved CO_2 plotted at the indicated times. Fingering is caused by a density instability as CO_2 dissolves into the aqueous phase.

a result of the high permeability of the formation and is caused by a density instability as CO_2 dissolves into the brine. The result is sensitive to the mixing rule used for the density of the CO_2-brine system. Ideal mixing, for example, results in the mixture density being lighter than the original brine causing the CO_2-brine mixture to move upwards rather than downwards. As can be seen in Fig. **9**(**d**), grid orientation effects appear in the solution as a result of the simple 7-point stencil employed. To remove these effects higher order discretization methods are being considered which honor the positive definiteness of the solution. This latter property is especially important to maintain in the reactive transport algorithms which solve for the logarithm of the concentration.

Further development of the CO_2 algorithm will involve adding methane and oil fluid phases, implementing higher order discretization methods to avoid grid orientation effects, and refining the time stepping algorithm to allow for larger time steps as phase changes take place. One possibility being investigated is the use of a kinetic formulation based on persistent variables, rather than the current basis switching algorithm. This formulation would also have the advantage of avoiding the problem of solving for different independent variables on different discretization levels when using multilevel solvers.

6. CONCLUSION & FUTURE DEVELOPMENT

A flexible, extensible, parallel computer code, PFLOTRAN, has been developed for modeling subsurface groundwater processes. The code has achieved petascale performance on ORNL's quad-core Cray XT5, Jaguar. PFLOTRAN is based on an object-oriented framework implemented in Fortran 9X, but it provides seamless integration with C and C^{++} packages such as SAMRAI for adaptive mesh refinement. Rapid development has been enabled through the use of PETSc as the parallel framework providing data structures, message passing and efficient solvers. A novel approach for running multiple realizations with each realization run on multiple processor cores was implemented. The code can be applied to various scenarios ranging from variably saturated media to CO_2 sequestration in deep geologic formations.

Future work will add capabilities for unstructured grids, Adaptive Mesh Refinement (AMR) based on the SAMRAI package, colloid-facilitated transport, Pitzer activity coefficient model, operator splitting algorithms, and multiple continua, among others. In addition, more work remains to be done to improve preconditioners and solvers at the extreme limit of hundreds of thousands of processor cores.

ACKNOWLEDGEMENTS

We would like to acknowledge the contributions of the PETSc development team and especially Barry Smith, Satish Balay, and Matt Knepley, without whose help PFLOTRAN would not have achieved its high degree of scalability. We are also indebted to G. (Kumar) Mahinthakumar and Vamsi Sripathi for their enhancement of PFLOTRAN's parallel I/O scalability. This research is supported under the U.S. Department of Energy SciDAC-2 program with funding provided by DOE Offices of Biological & Environmental Research (BER) and Advanced Scientific Computing Research (ASCR). Supercomputing resources were provided by the DOE Office of Science Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program with allocations on NCCS Jaguar at Oak Ridge National Laboratory.

REFERENCES

[1] C. I. Steefel, D. DePaolo, P. C. Lichtner, "Reactive transport modeling: An essential tool and a new research approach for the Earth sciences", *Earth and Planetary Science Letters*, vol. 240, no. 3-4, pp. 539-558, 2005.

[2] P. C. Lichtner, "Continuum formulation of multicomponent-multiphase reactive transport", in P. Lichtner, C. Steefel, E. Oelkers (Eds.), *Reactive Transport in Porous Media*, Vol. 34 of Reviews in Mineralogy, Mineralogical Society of America, pp. 1-81, 1996.

[3] C. Lu, P. C. Lichtner, "High resolution numerical investigation on the effect of convective instability on long term CO_2 storage in saline aquifers", in D. Keyes (Ed.), SciDAC 2007 *Scientific Discovery through Advanced Computing*, Vol. 78 of Journal of Physics: Conference Series, IOP Publishing, Boston, Massachusetts, p. 012042, 2007.

[4] P. Lichtner, "Continuum model for simultaneous chemical reactions and mass transport in hydrothermal systems", *Geochimica et Cosmochimica Acta*, vol. 49, pp. 779-880, 1985.

[5] G. E. Hammond, P.C. Lichtner, "Field-scale model for the natural attenuation of uranium at the Hanford 300 Area using high performance computing", *Water Resources Research*, (ISSN 0043-1397).

[6] S. Bea, J. Carrera, C. Ayora, F. Batlle, M. Saaltink, "Cheproo: A Fortran 90 object-oriented module to solve chemical processes in earth science models", *Computers & Geosciences*, vol. 35, pp. 1098-1112, 2009.

[7] J. C. Adams, W. S. Brainerd, R. A. Hendrickson, R. E. Maine, J. T. Martin, B. T. Smith, *The Fortran 2003 Handbook: The Complete Syntax, Features and Procedures*, Springer-Verlag, London, 2009.

[8] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. Melnes, B.F. Smith and H. Zhang. *PETSc users manual*, Tech. Rep. ANL95/11 -Revision 3.0.0, Argonne National Laboratory, 2009.

[9] The HDF Group, *HDF5 User's Guide: HDF5 Release 1.8.3*, NCSA, May 2009.

[10] J. E. Jones, C. S. Woodward, "Newton-Krylov-multigrid solvers for large-scale highly heterogeneous, variably saturated flow problems", *Advances in Water Resources*, vol. 24, pp. 763-774, 2001.

[11] G. E. Hammond, A. J. Valocchi, P.C. Lichtner, "Application of Jacobian-free Newton-Krylov with physics-based preconditioning to biogeochemical transport", *Advances in Water Resources*, vol. 28, no. 4, pp. 359-376, 2005.

[12] B. Lee, G. E. Hammond, "Parallel performance of preconditioned Krylov solvers for Richards equation", in preparation.

[13] EPA, *U. S. environmental protection agency soil cleanup criteria in 40 CFR part 192*, Tech. rep. 1998.

[14] J. M. Zachara, J. A. Davis, C. Liu, J. P. McKinley, N. Qafoku, D. M. Wellman, S. Yabusaki, "*Uranimum geochemistry in vadose zone and aquifer sediments from the 300 Area uranium plume*", Report PNNL-15121, Pacific Northwest National Laboratory, Richland, WA, 2005.

[15] D. Bond, J. Davis, J. Zachara, "Uranium(VI) release from contaminated vadose zone sediments: estimation of potential contributions from dissolution and desorption", *Adsorption of Metals by Geomedia II: Variables, mechanisms, and model applications*, pp. 375-416, 2008.

[16] S. B. Yabusaki, Y. Fang, S. R. Waichler, "Building conceptual models of field-scale uranium reactive transport in a dynamic vadose zone-aquifer-river system", *Water Resources Research*, vol. 44, pp. 1-24, 2008.

[17] M. D. Williams, M. L. Rockhold, P.D. Thorne, Y. Chen, "*Three-dimensional groundwater models of the 300 area at the hanford site, washington state*", Report PNNL-17708, Pacific Northwest National Laboratory, 2008.

[18] R. Span, W. Wagner, A new equation of state for carbon dioxide covering the fluid region from the triple-point temperature to 1100 K at pressures up to 800 MPa, *Journal of Physical and Chemical Reference Data* 25, pp. 1509-1596, 1996.

[19] Z. Duan, R. Sun, "An improved model calculating CO_2 solubility in pure water and aqueous NaCl solutions from 273 to 533 K and from 0 to 2000 bar", *Chemical Geology*, vol. 193, pp. 257-271, 2003.

[20] Z. Duan, J. Hu, D. Li, S. Mao, "Densities of the CO_2-H_2O and CO_2-H_2O-NaCl systems up to 647 k and 100 MPa", *Energy & Fuels*, vol. 22, no. 3, pp. 1666-1674, 2008.