Experiences and Challenges Scaling PFLOTRAN, a PETSc-based Code for Subsurface Reactive Flow Simulations, Towards the Petascale on Cray XT Systems

Richard Tran Mills
Oak Ridge National Laboratory, Oak Ridge, TN 37831

Vamsi Sripathi and G. (Kumar) Mahinthakumar
North Carolina State University, Raleigh, North Carolina 27695

Glenn E. Hammond
Pacific Northwest National Laboratory, Richland, WA 99352

Peter C. Lichtner
Los Alamos National Laboratory, Los Alamos, NM 87545

Barry F. Smith
Argonne National Laboratory, Argonne, IL 60439

ABSTRACT: *We describe our experiences running PFLOTRAN—a code for simulation of coupled hydro-thermal-chemical processes in variably saturated, non-isothermal, porous media—on the Cray XT series of computers, including initial experiences running on the petaflop incarnation of Jaguar, the Cray XT5 at the National Center for Computational Sciences at Oak Ridge National Laboratory. PFLOTRAN utilizes fully implicit time-stepping and is built on top of the Portable, Extensible Toolkit for Scientific Computation (PETSc). We discuss some of the hurdles to "at scale" performance with PFLOTRAN and the progress we have made in overcoming them on the Cray XT4 and XT5 architectures.*

KEYWORDS: PFLOTRAN, PETSc, groundwater, solvers

1 Introduction

Over the past several decades, subsurface (groundwater) 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 alternative energy sources and the efficacy of proposed remediation strategies for legacy waste sites. For years, traditional models—simulating single-phase groundwater flow and single-component solute transport within a single continuum, with basic chemical reactions such as aqueous complexing, mineral precipitation/dissolution and linear sorption to rock/soil surfaces and including radioactive decay—have been employed. 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 suite of biological and geochemical reactions at multiple scales. With this increased complexity, however, comes the need for more computational power, typically far beyond that of the average desktop computer. This is especially true when applying these sophisticated multiphase flow and multicomponent reactive transport mod-

els in three-dimensional problem domains. We contend, in fact, that in-depth understanding of many subsurface flow and transport problems will require simulations that demand petascale computing capabilities. In this paper, we briefly describe PFLOTRAN—a recently developed and highly-scalable code for simulations of reactive flows in geologic media—and discuss some of the challenges encountered and the progress made in scaling PFLOTRAN simulations towards the petascale on the Cray XT4 and XT5 architectures.

2 Problem formulation and discretization

PFLOTRAN solves a coupled system of continuum scale mass and energy conservation equations in porous media for a number of phases, including air, water, and supercritical CO_2, and for multiple chemical components.

The general form of the multiphase partial differential equations solved in the flow module of PFLOTRAN for mass and energy conservation can be summarized as [1]:

$$\frac{\partial}{\partial t}\left(\phi\sum_\alpha s_\alpha\rho_\alpha X_i^\alpha\right)+ \boldsymbol{\nabla}\cdot\sum_\alpha\left[\boldsymbol{q}_\alpha\rho_\alpha X_i^\alpha-\phi s_\alpha D_\alpha\rho_\alpha\boldsymbol{\nabla}X_i^\alpha\right] \;=\; Q_i^\alpha, \quad (1a)$$

and

$$\frac{\partial}{\partial t}\left(\phi\sum_\alpha s_\alpha\rho_\alpha U_\alpha+(1-\phi)\rho_r c_r T\right)+ \boldsymbol{\nabla}\cdot\left[\boldsymbol{q}_\alpha\rho_\alpha H_\alpha-\kappa\boldsymbol{\nabla}T\right] \;=\; Q_e. \quad (1b)$$

In these equations, α designates a phase (e.g. H_2O, supercritical CO_2), species are designated by the subscript i (e.g. $w = H_2O$, $c = CO_2$), ϕ denotes porosity of the geologic formation, s_α denotes the saturation state of the phase, X_i^α denotes the mole fraction of species i; ρ_α, H_α, U_α refer to the molar density, enthalpy, and internal energy of each fluid phase, respectively; $\boldsymbol{q}_\alpha$ denotes the Darcy flow rate defined by

$$\boldsymbol{q}_\alpha \;=\; -\frac{kk_\alpha}{\mu_\alpha}\boldsymbol{\nabla}(p_\alpha-W_\alpha\rho_\alpha g\boldsymbol{z}), \quad (2)$$

where k refers to the water saturated permeability, k_α denotes the relative permeability, μ_α denotes the fluid viscosity, W_α denotes the formula weight, and g denotes the acceleration of gravity. The source/sink terms, Q_i^α and Q_e, describe injection and extraction of mass and heat at wells, respectively.

The multicomponent reactive transport equations solved by PFLOTRAN have the form [1]:

$$\frac{\partial}{\partial t}\left(\phi\sum_\alpha s_\alpha\Psi_j^\alpha\right)+\boldsymbol{\nabla}\cdot\sum_\alpha\boldsymbol{\Omega}_j^\alpha \;=\; -\sum_m\nu_{jm}I_m, \quad (3)$$

for the jth primary species, and

$$\frac{\partial\phi_m}{\partial t} \;=\; \overline{V}_m I_m, \quad (4)$$

for the mth mineral. The quantities Ψ_j^α, $\boldsymbol{\Omega}_j^\alpha$ denote the total concentration and flux of the jth primary species in phase α (see [1] for more details). The mineral precipitation/dissolution reaction rate I_m is determined using a transition state rate law, and the qunatities ν_{jm} designate the stoichiometric reaction coefficients. These equations are coupled to the flow and energy conservation equations through the variable p, T, s_α, and $\boldsymbol{q}_\alpha$. In turn, chemical reactions may alter the material properties of the porous medium, such as porosity and permeability, which leads to coupling between the transport equations and flow and energy equations, albeit on a much slower time scale compared to the reverse coupling.

For both flow and transport, the governing partial differential equations all have the general form

$$\frac{\partial A}{\partial t}+\boldsymbol{\nabla}\cdot\boldsymbol{F} \;=\; \mathcal{S}, \quad (5)$$

with accumulation term A, source/sink term $\mathcal{S}$, and flux term $\boldsymbol{F}$ of the form

$$\boldsymbol{F} \;=\; \boldsymbol{q}\rho X-\phi D\rho\boldsymbol{\nabla}X, \quad (6)$$

PFLOTRAN utilizes a finite volume spatial discretization combined with backward-Euler (fully implicit) time stepping. Partitioning the computational domain into a set of finite volumes V_n and integrating the partial differential equations over each volume yields a discretized form of the mass conservation equations. Denoting the kth time step with superscript k, the residual equation for the discretized form of the partial differential equations is

$$R_n \;=\; \left(A_n^{k+1}-A_n^k\right)\frac{V_n}{\Delta t}+\sum_{n'}F_{nn'}A_{nn'}-\mathcal{S}_nV_n, \quad (7)$$

The flux $F_{nn'}$ across the $n-n'$ interface connecting volumes V_n and $V_{n'}$ is defined by

$$F_{nn'} = (q\rho)_{nn'} X_{nn'} - (\phi D \rho)_{nn'} \frac{X_n - X_{n'}}{d_n + d_{n'}}, \quad (8)$$

where the subscript nn' indicates that the quantity is evaluated at the interface, and the quantities d_n, $d_{n'}$ denote the distances from the centers of the control volumes V_n, $V_{n'}$ to the their common interface with interfacial area $A_{nn'}$. In general, R_n is a nonlinear function of the independent field variables. We use an inexact Newton method to solve the discretized equations for zero residual.

Within the flow and transport modules the equations are solved fully implicitly, but because transport generally requires much smaller time steps than flow, we couple these modules sequentially. A linear interpolation is used to obtain flow field variables within the transport solve. To account for changes in porosity and permeability due to mineral reactions, the transport solver is used to calculate an updated porosity over a flow time step and the revised porosity is passed back to the flow solver. Future implementations may explore independent grid hierarchies for flow and transport, as well as fully coupled schemes.

3 Architecture and Parallel Implementation

PFLOTRAN is has been written from the ground up with parallel scalability in mind, and can run on machines ranging from laptops to the largest massively parallel computer architectures. Through judicious use of Fortran 90/95 features, the code employs a highly modular, object-oriented design that can hide many of the details of the parallel implementation from the user, if desired. PFLOTRAN is built on top of the PETSc framework [2; 3; 4] and uses numerous features from PETSc including nonlinear solvers, linear solvers, sparse matrix data structures (both blocked and non-blocked matrices), vectors, constructs for the parallelism of PDEs on structured grids, options database (runtime control of solver options), and binary I/O.

PFLOTRAN's parallel paradigm is based on domain decomposition: each MPI process is assigned a subdomain of the problem and a parallel solve is implemented over all processors. Message passing (3D "halo exchange") is required at the subdomain boundaries with adjacent MPI processes to fill ghost points in order to compute flux terms (Equation 8). A number of different preconditioners from PETSc or other packages (PETSc provides interfaces to several) can be employed, but currently we usually use single-level domain decomposition preconditioners employed inside of a global inexact Newton-Krylov solver. Within the Krylov solver, gather/scatter operations are needed to handle off-processor vector elements in matrix-vector multiplies, and numerous *MPI_Allreduce()* operations are required to compute vector inner products and norms, making communication highly latency-bound.

4 Parallel Performance

In this section, we examine performance in a strong scaling context using a benchmark problem from a model of a hypothetical uranium plume at the Hanford 300 Area in southeastern Washington state, described in [5]. The computational domain of the problem measures $1350 \times 2500 \times 20$ meters (x,y,z) and utilizes complex stratigraphy (Figure 1) mapped from the Hanford EarthVision database [6], with material properties provided by [7]. The stratigraphy must be read from a large HDF5 file at initialization time. We examine aspects of both computation and I/O in this section.

In the benchmark runs here, we utilize an inexact Newton method with a fixed tolerance for the inner, linear solve. The linear (Jacobian) solves employ the `BCGS` stabilized bi-conjugate gradient solver (BiCGstab) in PETSc, with a block-Jacobi preconditioner that applies an incomplete LU-decomposition solver with zero fill-in (ILU(0)) on each block. (For linear solves with block-structured matrices, a block-ILU(0) solve is employed.) Our choice of preconditioner is a very simple one, but we have been surprised at how well it has worked at scale on large machines such as Jaguar. We have experimented with more sophisticated preconditioners such as the the Hypre/BoomerAMG and Trilinos/ML algebraic multigrid solvers, but we have yet to identify a preconditioner that is as robust and as scalable at high processor counts as block-Jacobi. Solvers such as BoomerAMG display excellent convergence behavior, but show unacceptable growth in set up time above 1000 cores. (Determining how to make good use of multi-level solvers is an area of active research for us.)

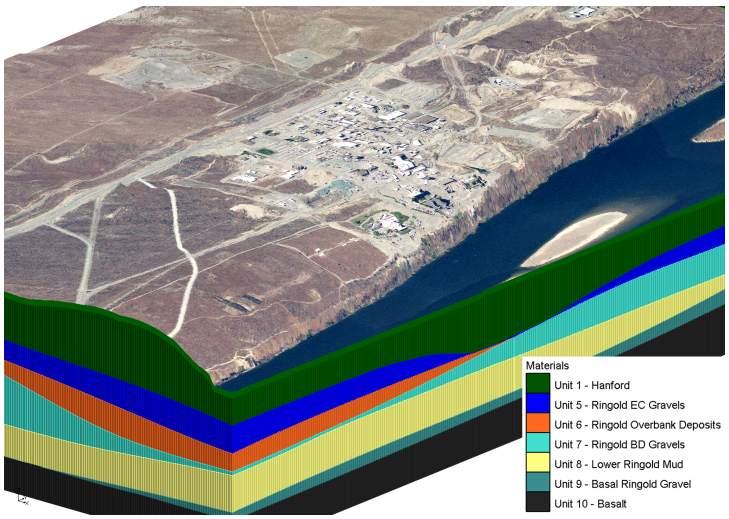

Figure 1: Hanford 300 Area stratigraphy (z scale = 20x, z axis ranges 70–130 meters).

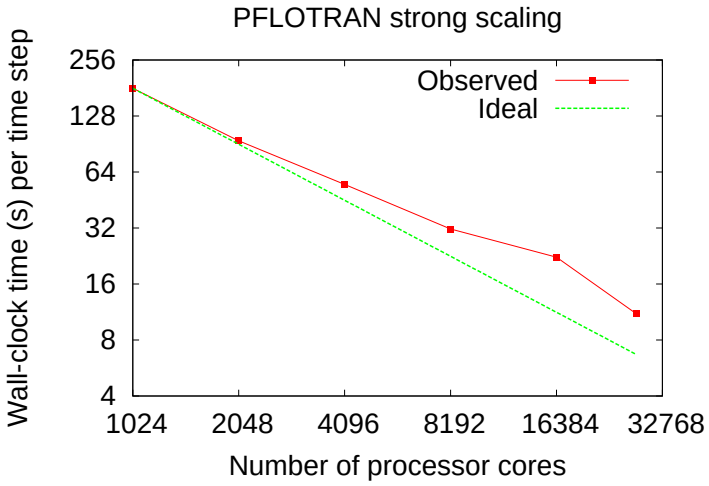

Figure 2: Strong-scaling performance on the Cray XT4 of the computation phase (no I/O) of PFLOTRAN for a variably-saturated flow problem with 270 million total degrees of freedom.

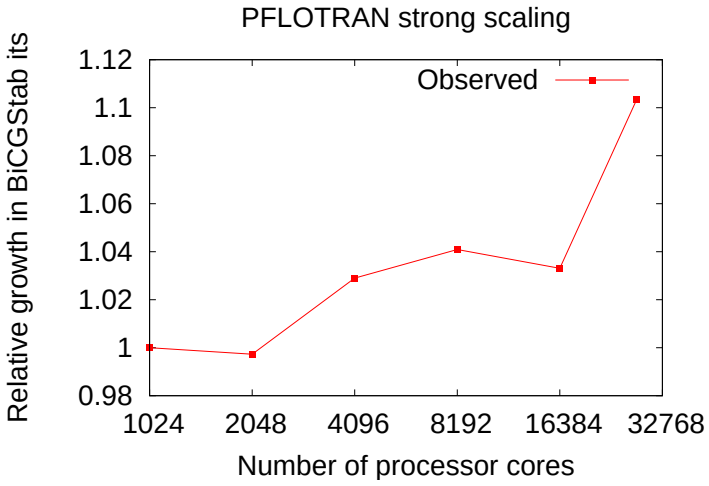

Figure 3: Growth of BiCGstab iterations for the strong-scaling case depicted in Figure 2.

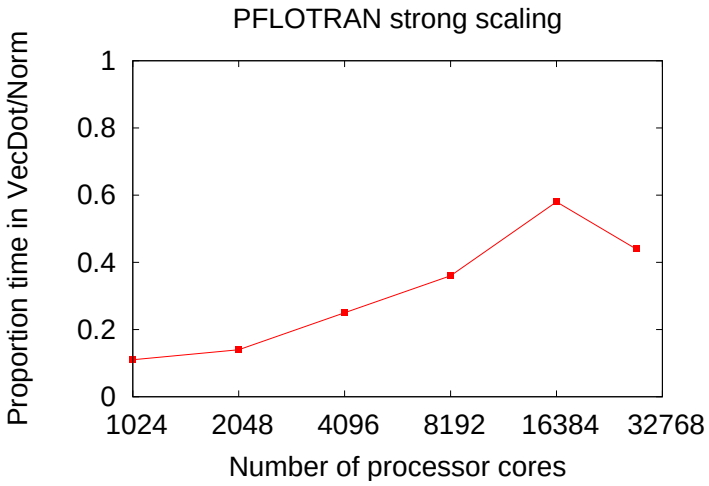

Figure 4: Proportion of time spent in vector dot product and norm computations for the strong-scaling case depicted in Figure 2.

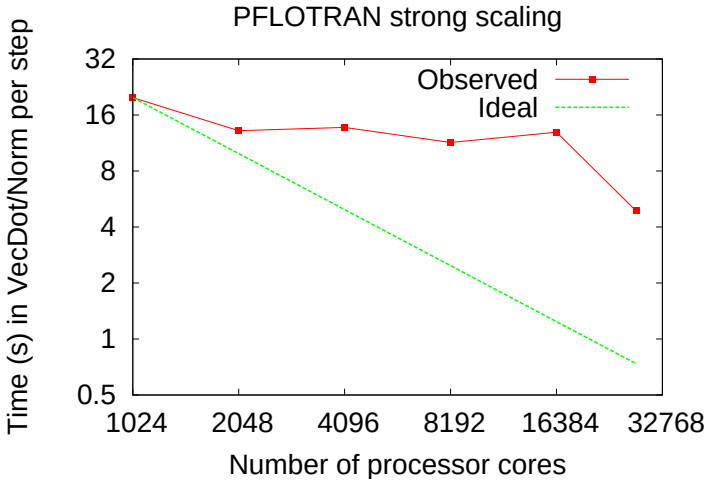

Figure 5: Combined cost of all vector dot product or norm calculations for the strong-scaling case depicted in Figure 2.

4.1 Computational Phase

4.1.1 Strong-scaling, flow, 270M DoF

Figure 2 depicts the strong scaling behavior of the flow solver for a 1350 × 2500 × 80 cell flow-only version of the problem (270 million total degrees of freedom) run on the Cray XT4 at ORNL/NCCS. We ran the problem for 50 time steps. The scaling behavior is good all the way out to 27580 processor cores (near the limit of the number of cores available on the machine with 800 MHz DDR2 memory—we did not wish to introduce load imbalance by mixing memory speeds), although some departure from linear scaling is seen. Because we employ a single-level domain decomposition preconditioner (block-Jacobi), it might be supposed that the loss of effectiveness of the precondition as the number of processor cores grows (and hence the size of each local subdomain shrinks) might be responsible for a significant portion of the departure from linear scaling. Figure 3 shows that the growth in the total number of BiCGstab iterations is actually very modest, with only approximately 10% more iterations required at 27580 cores than at 1024 cores. The departure from linear scaling appears to be almost entirely due to the large growth in the proportion of time spent in the computation of vector inner products (dot products) and norms (Figure 4), which is dominated by the cost of `MPI_Allreduce()` operations. In fact, if we examine the strong-scaling behavior of the cost of vector dot products and norms versus the cost of all other function calls (Figures 5 and 6, we see that the strong scalability of the dot products and norms is quite poor, while the scalability of everything else is essentially linear (and even slightly superlinear at 16384 cores).

The time spent in `MPI_Allreduce()` calls was further investigated using CrayPAT, which breaks the time into two components: *MPI_Allreduce* and *MPI_Allreduce (Sync)*. According to Cray, *MPI_Allreduce (Sync)* is due to the implicit synchronization operation in the `MPI_Allreduce()` call and is likely due to load imbalance. However, further analysis of the code indicated that only 5% of the processors incur any significant application load imbalance. To investigate this further, we developed a perfectly load balanced dot product kernel that performs approximately the same number of `MPI_Allreduce()` calls as PFLOTRAN and takes approximately the same execution time (accomplished via redundant local dot products) as PFLOTRAN. This kernel yielded nearly the same amount of *MPI_Allreduce (Sync)* time, indicating that the *MPI_Allreduce (Sync)* experienced by the application is not due to application load imbalance but likely due to system load imbalance (i.e., performance variability among cores).

To improve `MPI_Allreduce()` performance

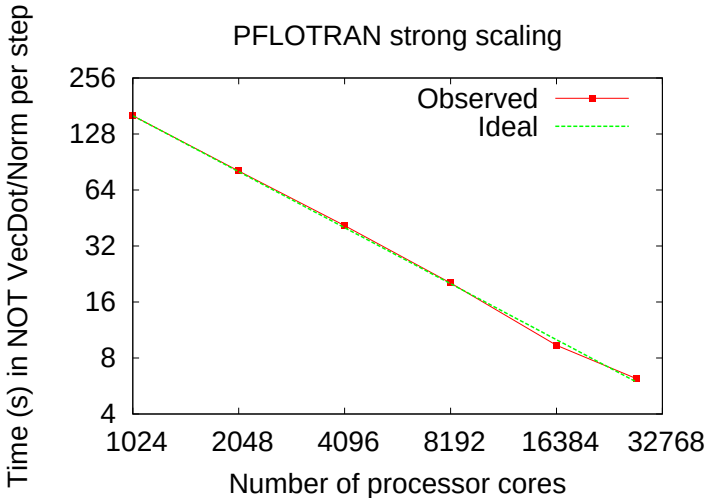

Figure 6: Combined cost of all routines that are not vector dot product or norm calculations for the strong-scaling case depicted in Figure 2.

on the Cray XT4, we experimented with a number of MPI communication related environment variables. Of these settings, `MPICH_PTL_MATCH_OFF` had the greatest impact as it eliminates a portal systems call that can add significant overhead to latency dominated applications such as PFLOTRAN (dominated by `MPI_Allreduce()`s of a single number in the flow solver). Other options such as `MPICH_FAST_MEMCPY` and `MPICH_COLL_OPT_ON` (which we note has become the default behavior in MPT 3) also had some impact. These optimizations reduced the `MPI_Allreduce()` time by approximately 18%.

As the `MPI_Allreduce()` calls performed during the BiCGstab solve dominate the overall execution time at large core counts on the XT systems, we implemented an alternative, "improved" version of algorithm [8] in a new `KSP` solver ("`IBCGS`") in PETSc. The standard BiCGstab method requires four `MPI_Allreduce()` calls per iteration (including the vector norm computation to check convergence). The restructured algorithm used in `IBCGS` requires only two `MPI_Allreduce()` calls, at the expense of a considerably more complicated algorithm, more local computation (additional vector operations), and computation of a transpose matrix-vector product at the beginning of the solve. By lagging the calculation of the residual norm, it is possible to further reduce communication and wrap everything into one `MPI_Allreduce` iteration, at the cost of doing one additional iteration. Despite the additional computation involved, the restructured solver is generally faster than the standard BiCGstab solver on the XT4/5 when using more than a few hundred processor cores. The table below displays the breakdown (as reported by CrayPAT) of time spent in *MPI_SYNC*, *MPI*, and *User* time when using the traditional ("BCGS") and restructured ("IBCGS") algorithms for a 30 time step run (including initialization time but no disk output) of the benchmark problem on 16384 cores of the XT4 version of Jaguar. The amount of time spent in *User* computations is larger in the IBCGS case, but the time spent in *MPI_SYNC* and *MPI* is far lower.

Group	Traditional	Restructured
MPI_SYNC	196	120
MPI	150	79
User	177	200

4.1.2 Strong-scaling, transport, 2B DoF

We have recently made some preliminary benchmark runs on the petaflop incarnation of Jaguar, the Cray XT5 at ORNL/NCCS, with a coupled flow and transport problem from the Hanford 300 Area. The problem consists of 850 $\times$ 1000 $\times$ 160 cells and includes 15 chemical components, resulting 136

million total degrees of freedom in the flow solve and 2.04 billion total degrees of freedom in the transport solve. This problem size is at the limit of what we can do with 32-bit array indexing. (Although PFLOTRAN and PETSc can support 64-bit indices, some work will be required to get our code working with some of the libraries we link with.) The problem was run for 25 time steps of both flow and transport. In many simulations, we would use a longer time step for flow than for transport, but in this case there is a time varying boundary condition (the stage of the adjacent Columbia River) that limits the size of the flow time step (to a maximum of one hour), and higher resolution for the transport step is not needed.

Figure 7 depicts the preliminary scaling behavior we have observed on up to 65536 processor cores on Jaguar. Figures 8 and 9 break the performance down into time spent in the flow and transport solves, respectively. It is not surprising that the scalability of the flow problem is poor: at only 136 million degrees of freedom, this is a very small problem to be running on so many processor cores. We employ the large number of processor cores for the benefit of the much larger (15 times) and computationally more expensive transport problem. We note that the coupled flow and reactive transport problems that we might typically run with PFLOTRAN will generally involve transport problems considerably larger than their flow counterparts, perhaps going as high as 20 chemical components. If kinetic rate laws are used for sorption, then the size of the transport problem becomes even higher because a kinetic sorption equation (without flux terms) must be added for each kinetic reaction. It is possible that at some point it may make sense to solve the flow problem redundantly on disjoint groups of MPI processes, much as parallel multi-level solvers do for coarse-grid problems. This may not be necessary, however, as the cost of the transport solves may be dominant, and in many cases it is possible to run the flow solves with a much larger time step size than the transport solve.

We have not yet had the opportunity to conduct further experiments to understand and improve the performance of this benchmark problem. We believe that running it with the restructured BiCGstab solver in PETSc will be beneficial, but we first need to add support in PETSc for the transpose matrix-vector product for blocked sparse matrices with large block size. The data we collected during the run indicate that there are some significant sources of load imbalance that do not derive from the partitioning of work among the processors. (We believe that the partitioning is good, as this problem employs a regular grid and the ratio of maximum to minimum times spent in some key routines is not high even at 65536 cores; for example, it is 1.2 inside the Jacobian evaluation routines.) More detailed evaluation (running experiments with `MPI_Barrier()` calls inserted before collective communications, for example) is needed to identify the sources of load imbalance.

4.2 I/O Phase

PFLOTRAN originally employed serialized I/O through MPI process 0. Because PFLOTRAN spends relatively little time in I/O, this actually proved workable up to about 8000 processor cores. To scale beyond this, we added parallel I/O in the form of 1) routines employing parallel HDF5 for reading input files and writing out simulation output files, and 2) employing direct MPI-IO calls in a PETSc Viewer backend (for checkpointing). This greatly increased the scalability of the code, but initialization costs still became unacceptable at when using more than about 16000 cores (roughly half the size of the Jaguar XT4 system). Here we detail how we have mitigated this problem.

The Cray XT architecture uses the Lustre file system (lfs) [9] to facilitate high bandwidth I/O for applications. Lustre is an object based parallel file system that has 3 main components namely Object Storage Servers (OSS's), Meta Data Servers (MDS) and File system clients. Each OSS can serve multiple Object Storage Targets (OST's) which act as I/O servers, the MDS manage the names and directories for the file system and the file system clients are the Cray XT compute nodes. The Cray XT5 system has 672 OST's, 1 MDS and 37,544 file system clients or compute nodes. Files are broken into chunks and stored as objects on the OST's in a round robin fashion. The size of the object stored on a single OST is defined as File Stripe Size and the number of OST's along which the file is striped is defined as File Stripe Count. The default value for stripe size is 1MiB and stripe count is 4 OST's. We have used the default lfs settings for our runs but these parameters can be modified by the user with the help of lfs commands. Depending upon the file access

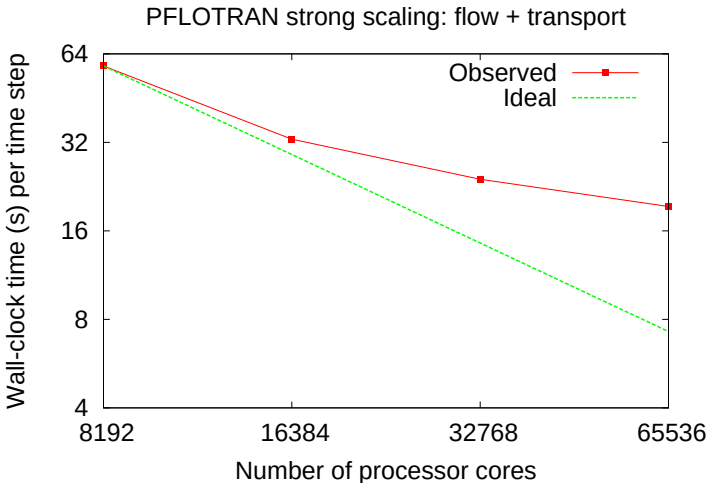

Figure 7: Combined strong-scaling performance of flow and transport solves for a coupled flow and reactive transport simulation on an $850 \times 1000 \times 160$ cell grid with 15 chemical components.

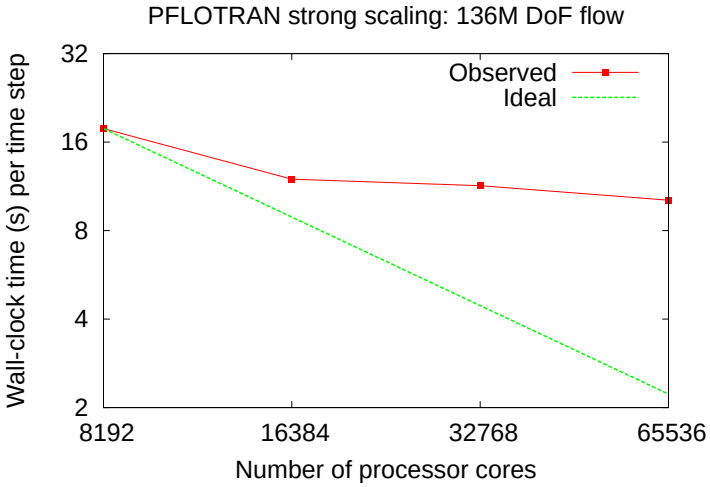

Figure 8: Strong-scaling performance of the flow solve component (136 million degrees of freedom) of the strong-scaling study depicted in figure 7.

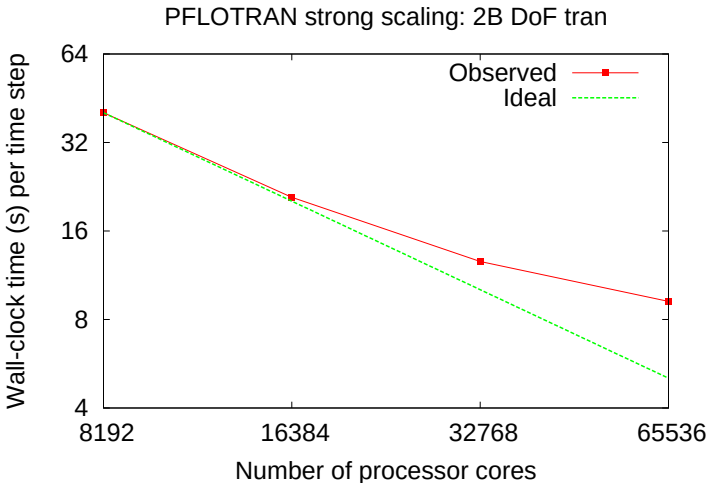

Figure 9: Strong-scaling performance of the transport solve component (2.04 billion degrees of freedom) of the strong-scaling study depicted in figure 7.

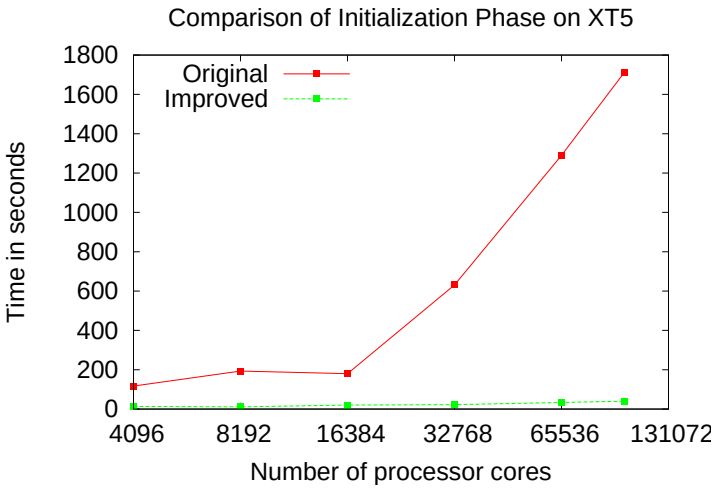

Figure 10: Performance comparison of the original and improved initialization routines on the 270 million degrees of freedom flow problem on the Cray XT5. The curve labeled "original" depicts the performance when all processes open and read from the HDF5 file, and the "improved" curve depicts performance when only a subset of processes read the data and then broadcast to the other members of their I/O group.

pattern, the lfs settings will have an impact on I/O performance.

From the scaling studies performed on Cray XT, it was observed that the initialization phase of PFLOTRAN does not scale well with increase in processor count. The first curve in Figure 10 shows the timing of the initialization phase at different processor counts on Cray XT5. The initialization phase almost entirely consists of HDF5 routines which read from a single large HDF5 input file. All processes participate in a parallel read operation which involves a contiguous region of HDF5 datasets and there are two different read patterns i.e., every process reads the dataset entirely and every process reads a chunk of the dataset starting from its own offset within the dataset. For this kind of read pattern the lfs settings (stripe size and stripe count) will have little impact on performance because most of the processors will be accessing a single OST at any given point of time during execution.

To investigate the reasons for the poor performance of the initialization phase at scale, we used CrayPAT to profile PFLOTRAN on up to 27572 cores of the Cray XT4. Figure 11 shows the percentage contribution of different routines to overall execution time. The profiling was done for the 270M DoF test problem with 30 time-steps and with no output. From the figure, it is evident that the MPI routines `MPI_File_open()`, `MPI_File_close()` and `MPI_File_read_at()` contribute significantly at higher processor count. It must be noted here that for the entire file system there is only 1 Meta data server (MDS) and every time a process needs to open/close a file it needs to poll the MDS. As the number of clients increases the file open/close become a much costlier operation and this setup would become a bottleneck for I/O performance. This has been identified as a problem for system scalability by others [10; 11].

To avoid this performance penalty at higher processor counts we have modified the read mechanism within PFLOTRAN. Instead of having every process participate in the read operations we implemented a mechanism where only a subset of processes execute the HDF5 routines and are involved in the read operations. The code example below describes the creation of new sub-communicators that are used in the improved I/O method.

```
group_size = HDF5_GROUP_SIZE (e.g., 1024)
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)

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

call MPI_Comm_split(global_comm,
                    reader_color,
                    reader_key,readers,
                    ierr)

if (reader_color==1) then
    call h5pset_fapl_mpio_f(prop_id,readers,
                            MPI_INFO_NULL,
                            hdf5_err)
    call h5dread_f(...,HDF_NATIVE_INTEGER,
                   integer_array,length)
endif
call MPI_Bcast(integer_array,length,
               MPI_INTEGER,0,iogroup,ierr)
```

Temporary buffers will have to be allocated at the reader processes to hold the data for their group. The reader processes collect the offset values and chunk sizes from the processes in their group by using `MPI_Gather()`. After the designated processes have completed reading the data from the file they would distribute the data using `MPI_Bcast()` or `MPI_Scatterv()` depending upon the offset values and chunk sizes to the rest of the processes in their respective groups. By careful selection of reader processes, we ensure that there is no network congestion at this point. Even though we did not face out-of-memory problems with our runs, we can avoid this situation in the future, if it arises, by using a smaller group size or by using multiple `MPI_Bcast()`/`MPI_Scatterv()`.

With these modifications, we were able to reduce the time spent in initialization phase from 1700 to 40 seconds at 98304 cores of Cray XT5. The second curve in Figure 10 shows the timing of the initialization phase with the improved I/O method. For the runs up to 16384, we have used a group size of 1024 and for runs above 16384 it is 8192.

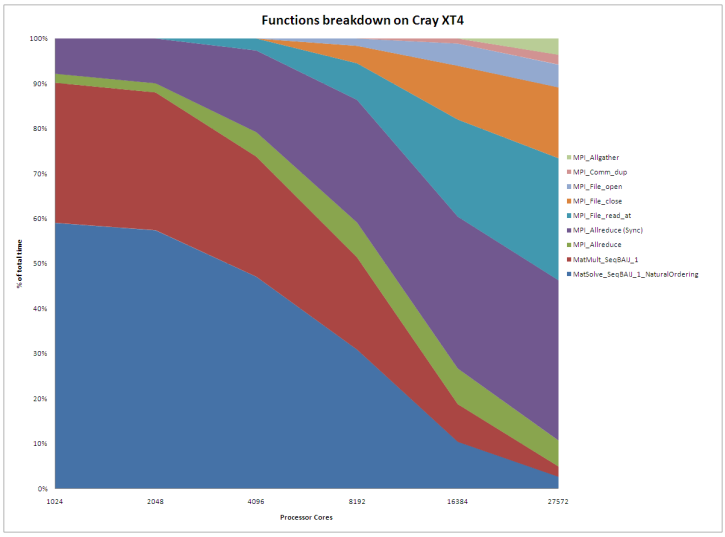

Figure 11: Breakdown of contributions of different routines to overall execution time of the 270 million degrees of freedom flow problem run on the Cray XT5 for 30 time steps, with no disk output.

5 Conclusions and Future Work

PFLOTRAN has only been under active development for a few years, but has already become a modular, object-oriented, extensible, and scalable code. We have demonstrated that it is currently able to make effective use of a significant portion of a petascale machine. It does not do so with anywhere near the efficiency that we would like, but it is possible to use the code to solve leadership-class computing problems right now.

Further development of PFLOTRAN continues at a rapid pace. One focus is on developing better solvers/preconditioners to further improve scalability. We are exploring several types of multi-level approaches, which we believe are ultimately necessary to achieve good weak scalability. These approaches will probably be combined with physics based ones [12] that will allow multi-level methods to be employed on linear systems to which they are well-suited. Though we did not discuss it in this paper, we are also making progress on adding structured adaptive mesh refinement capabilities to the code, which will greatly decrease the memory and computational requirements for simulations that must resolve certain fine-scale features. We are also adding support for unstructured meshes, which will allow static refinement of the grid around complex geologic features, around wells, etc. Support for multiple-continuum (subgrid) models will also be added to the code, which will dramatically increase the work associated with each grid cell and, incidentally, result in more efficient use of leadership-class machines, as much of the subgrid work can proceed in embarrassingly parallel fashion.

Acknowledgements

This research was partially sponsored by the Climate and Environmental Sciences Division (CESD) of the Office of Biological and Environmental Research (BER) and the Computational Science Research and Partnerships (SciDAC) Division of the Office of Advanced Scientific Computing Research (ASCR) within the U.S. Department of Energy's Office of Science (SC). This research used resources of the National Center for Computational Sciences (NCCS) at Oak Ridge National Laboratory (ORNL), which is managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Pacific Northwest National Laboratory is managed for the U.S. Department of En-

ergy by Battelle Memorial Institute under Contract No. DE-AC06-76RL01830. Argonne National Laboratory is managed by UChicago Argonne, LLC, for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357. Los Alamos National Laboratory is managed is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396.

About the Authors

Richard Tran Mills is a computational scientist in the Computational Earth Sciences Group of the Computer Science & Mathematics Division at ORNL. After dropping out of high school, he earned a B.A. in Geology and Geophysics from the University of Tennessee, Knoxville, and a Ph.D. in Computer Science from the College of William and Mary. His research interests include parallel and high-performance computing, computational science and scientific computing, software for the iterative solution of sparse algebraic systems of equations, computational geoscience applications, geospatiotemporal data mining, and execution context-aware scientific software. He can be reached at ORNL, MS 6015, Oak Ridge, TN 37831, E-Mail: rmills@ornl.gov.

Vamsi Sripathi is a graduate student in the department of Computer Science at North Carolina State University. His areas of interest include high performance computing, parallel I/O, and performance analysis and optimization. E-Mail: vamsi_s@ncsu.edu

G. (Kumar) Mahinthakumar is an Associate Professor of Civil Engineering at North Carolina State University. His research interests include computational modeling of subsurface flow and reactive transport processes, high performance computing, computational fluid dynamics, inverse problems, finite element methods, sparse matrix solvers, genetic algorithms, and biomedical computing applications. He holds a Ph.D. in Civil Engineering from the University of Illinois at Urbana-Champaign. E-Mail: gmkumar@ncsu.edu

Glenn Hammond is a computational hydrologist in the Hydrology Technical Group at Pacific Northwest National Laboratory. His research interests include high-performance computing and subsurface reactive transport. He holds a Ph.D. in Civil and Environmental Engineering from the University of Illinois at Urbana-Champaign. E-Mail: glenn.hammond@pnl.gov.

Peter Lichtner is a member of the technical staff in the Earth and Environmental Sciences division at Los Alamos National Laboratory. He is experienced in numerical modeling of fluid transport combined with chemical reactions of minerals, aqueous species and gases in porous media that has led to fundamental research in this field. He has an international reputation in this field with 60 publications of both theory and application to geochemical-hydrogeologic problems. In addition, he has 19 publications in the field of theoretical nuclear physics. He has co-edited a book on reactive transport entitled *Reactive Transport in Porous Media*, and currently is PI of a SciDAC (Scientific Discovery through Advanced Computing) project entitled *Modeling Multiscale-Multiphase-Multicomponent Subsurface Reactive Flows using Advanced Computing*, which funds the development of PFLOTRAN. E-Mail: lichtner@lanl.gov

Barry Smith is a Senior Computational Mathematician in the Mathematics and Computer Science Division at Argonne National Laboratory and is the leader of the PETSc project. He holds a Ph.D. in Mathematics from the Courant Institute of New York University and a B.S. in Mathematics from Yale University. E-Mail: bsmith@mcs.anl.gov

References

[1] P.C. Lichtner. Continuum formulation of multicomponent-multiphase reactive transport. In P.C Lichtner, C.I. Steefel, and E.H. Oelkers, editors, *Reactive Transport in Porous Media*, volume 34 of *Reviews in Mineralogy*, pages 1–81. Mineralogical Society of America, 1996.

[2] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2009. http://www.mcs.anl.gov/petsc.

[3] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2009.

[4] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[5] Glenn E. Hammond, Peter C. Lichtner, Richard T. Mills, and Chuan Lu. Towards petascale computing in geosciences: application to the hanford 300 area. In David Keyes, editor, *SciDAC 2008 Scientific Discovery through Advanced Computing*, volume 125 of *Journal of Physics: Conference Series*, page 012051, Seattle, Washington, 2008. IOP Publishing.

[6] P. D. Thorne, M. P. Bergeron, and V. L. Freedman. Groundwater data package for hanford assessments. Report PNNL-14753 Rev. 1, Pacific Northwest National Laboratory, Richland, WA, 2006.

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

[8] L. T. Yand and R. Brent. The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In *Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing*. IEEE, 2002.

[9] Lustre file system Web page. http://www.lustre.org.

[10] Mark R. Fahey, Jeff M. Larkin, and Joylika Adams. I/o performance on a massively parallel cray XT3/XT4. In *22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008*, pages 1–12, Miami, Florida, 2008.

[11] Weikuan Yu, Jeffrey S. Vetter, and Sarp Oral. Performance characterization and optimization of parallel i/o on the cray XT. In *22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008*, pages 1–11, Miami, Florida, 2008.

[12] Glenn E. Hammond, Albert J. Valocchi, and Peter C. Lichtner. Application of Jacobian-free Newton-Krylov with physics-based preconditioning to biogeochemical transport. *Advances in Water Resources*, 28(4):359–376, April 2005.